Deep Learning for Robot Perception and Cognition

Chapter 1

Robotic Grasping in Agile Production

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

ABSTRACT

Recent developments in robotics and deep learning have enabled high-level robotic tasks to be learned from simulated or real data. In this chapter, the task of robot grasping is covered, where a robot manipulator learns a grasping model from perceptual data, such as RGB-D or point clouds. The chapter is presented in context of robotics for agile production, thereby providing requirements and limitations that are relevant for deep learning in robotics. An overview of different approaches is given with special attention to the evaluation of robotic object grasping and the potential follow-step of object manipulation. In addition, a list of datasets is provided that utilize simulation to generate training data for object grasping.

KEYWORDS

Robot object grasping and manipulation, Object pose estimation, Grasp representation, Deep learning, Agile production

1.1 INTRODUCTION

Agile production refers to robotic production systems that operate quickly and adaptively in dynamically changing work environments. This means robots should adapt to variations in tasks, the local production environment and its contents such as work pieces. Moreover, recent advances in robotics allow for the safe interaction between humans and robots towards a common shared goal [1, 2]. In such scenario, a human operator and the robot share the same workspace to complete a task in the dynamic environment. The contrast to traditional industrial production is rather evident as most commonly production environments exclude human operators and production processes have little variation in tasks and work pieces for the same production line. Besides the

[1] Cognitive Robotics group, Automation Technology and Mechanical Engineering, Tampere University, 33720, Tampere, Finland.

[2] Computer Vision group, Automation Technology and Mechanical Engineering, Tampere University, 33720, Tampere, Finland.

trend of agility in production, other developments towards smart manufacturing refer to the utilization of advanced data analytics to improve system performance and decision making. This ongoing effort to digitize industry and collect relevant data, enables the processing of data by machine and deep learning techniques [3]. As a step-by-step procedure, traditional machine learning performs feature extraction and model construction in a separated manner, as compared to deep learning, which learns features and the model in an end-to-end manner. In the context of smart manufacturing and its generation of big data [4], deep learning is therefore regarded as a great advantage, as it avoids the complex step of feature engineering. Different levels of data analytics can then extract knowledge, identify patterns and make decisions on various processes. That is, models can act on individual production processes as well as on the factory as a whole, for example, by tracking parts and their production progress, monitoring equipment and its usage and evaluation of a process' performance and quality.

In this chapter, we narrow our focus towards one common task for robots in agile production, i.e., object grasping, and its utilization of deep learning. First, we introduce in Section 1.1 robot tasks and deep learning, in context to the requirements and limitations the agile production environment sets. Following, the robot tasks of grasping and object manipulation are described in Section 1.2, where different techniques are classified according to the sensor modality (RGB-D or point clouds) and the objects to be grasped. Evaluation of grasping and benchmarking of manipulation is discussed in Section 1.3 and Section 1.4, respectively. Finally, a brief overview of grasp datasets is given in Section 1.5.

1.1.1 Robot tasks in agile production

Robot manipulators in agile production environments are tasked with repetitive actions that require high precision. Typical examples of repetitive tasks are bin picking and packaging in warehouses and assembly for the automotive industry. Most of these industrial robots are position controlled, to ensure a high accuracy and high precision of end-effector motion [5], which has implications to their tasks and capabilities. Practically speaking, a position-controlled robot is designed to execute repetitive and predefined motions without any external disturbances. Tasks such as object pick and placement, spot welding and painting are suitable for such robot control, as the task is well-defined and external disturbances are unlikely to occur. That is, consecutive objects are known, path motion does not change and contact to a rigid surface is either planned or not included.

In case of robot tasks that include contact, other control modalities are common, such as force/torque and compliance control, which enables a more adaptive behavior of the robot to external disturbances and sensors [6]. Typical tasks that rely on such control methods are for example assembly (peg-in-hole, bolt screwing) and surface finishing (deburring, sanding, polishing). In these cases, external force/torque sensors provide information for the controller to

act upon during task and motion execution. Drawbacks of this approach are the complexity of the control structure and how uncertain and noisy sensor measurements should be taken into account.

While in both cases the robots work independently, they are no longer stand-alone devices, but connected as a general effort towards Industry 4.0 [7]. This trend towards digitization offers several advantages, such as predictive maintenance and optimal factory resource deployment, which ultimately lead to a higher efficiency and productivity. Digitization also offers integration solutions, as a so-called digital twin can be utilized for the programming of robot tasks, by programming and verifying motion and task models in simulation and deploying them on the real robot [8].

Recent advances in robotics also allows for the collaboration between humans and machines in performing shared tasks in a safe way. In this context, where robot and human share the same workspace, adaptation from both the human and the robot is required such that they could work together as smoothly as possible meanwhile capturing the best capabilities from each side [2]. Interfaces to ease robot programming are an inherent part of this, by techniques such as learning from demonstration [9, 10], web-based interfaces [11], Augmented Reality [12] and many others [13]. User experience design should then ensure efficient collaboration and acceptance from the human operator [14].

1.1.2 Deep learning in agile production

Within an industrial production environment, deep learning can be utilized in data analytics in different ways and on different processes [4]. With respect to robot manipulators and related sensor systems, deep learning has been utilized for perception, action and control [15].

As perception tool, deep learning can detect and classify objects, and estimate their pose [16]. Robot systems can then utilize this information for grasping and manipulation tasks, towards goals such as pick and placement, assembly [9] and bin picking [17]. A broad overview and discussion on deep learning-based object pose estimation and robot grasping is given in the body of this chapter. Besides the perception of objects, visual processing is also utilized for monitoring persons in a workspace, by detection of a person and the estimation of his/her body pose. Actions of the person, by body movement and gesture recognition, can guide robot motion or serve for monitoring safety in a human-robot collaborative scenario [18] (i.e., by estimating the distance between a person and robot, according to ISO 15066 [19]). Chapter 14 in this book is dedicated to deep learning for human activity recognition.

Deep learning offers an additional approach towards robot motion and task generation, by collecting data (real or simulated), training motion and task models based on this data and deployment of the models on a real robot. In context of agile production, robot (deep) learning has been demonstrated for tasks such as bin picking [17], grasping [20], assembly [9] and peg-in-hole

insertion [21]. Moreover, the interaction and collaboration between human and robot is under investigation as well, for example, by imitation learning of motion trajectories [22] and deep reinforcement learning for a collaborative packaging task [23].

1.1.3 Requirements in agile production

Development and integration of robots and robot systems in production and manufacturing environments need to follow standards on safety requirements as set by ISO 10218-1/2 [24]. In addition, if human operators are collaborating with robots or in its close vicinity, additional standard ISO 15066 [19] needs to be taken into account. Compliance to these standards can be done in several ways, for example by vision-based safety systems [25]. As the grasping approaches in this chapter act as stand-alone systems, only the functional requirements are described and not the safety requirements as defined by the standards.

One crucial factor for the functionality of a robot grasping action in agile production is timing, for both its implementation and execution. Implementation of the grasping model implies the generation of a new grasp detection model by training a Deep Neural Network (DNN) with real or simulated data. The collection of such data can be time-consuming and difficult in situations were no accurate object models (e.g., CAD) are available or when no appropriate imaging system can be utilized. Moreover, the training of a DNN model requires computational resources that are often not present in small to medium-sized enterprises (SMEs) and external computation clusters need to be utilized. In addition, inference time of a grasp model should not be the bottleneck of the robotic system and, ideally, real-time control should be possible. This means that a grasp detection model should be light-weight ($<$1 GB), fast ($>$25 FPS) and have low latency ($<$100 ms), while ensuring a grasp detection or pose estimation accuracy such that parts can be picked up.

In short, as the requirements set by industrial agile production implies a fast implementation and reconfiguration of different robotic tasks, this should be reflected in the generation of robot tasks that utilize deep learning as well. A careful trade-off between inference time and input resolution should therefore be made, while considering suitable computational hardware.

1.1.4 Limitations in agile production

The industrial agile production environment sets limitations on several aspects for utilizing deep learning. Practical limitations concern the environment itself and the difficulty on capturing reliable models and data, such as sensor models with noise and accurate CAD models of objects. Following, a brief explanation is given of relevant hardware and software limitations that affect both the integration of advanced robotics in agile production and how it impacts the utilization of deep learning.

Grasping hardware

Most common industrial robot grippers have a parallel gripping mechanism that only allows one contact patch per gripper side [26]. Gripper finger design can then be utilized to customize the gripper for individual parts [27] and create a more robust grip by shape matching. Alternatively, tool change mechanisms enable multiple (custom) grippers per robot. Nevertheless, the number of parts that a robot can handle is still limited, especially when an accurate grasp pose is required for subsequent object manipulation. Multi-fingered grippers [28], or compliant gripping surfaces [29] are under development as well, however, in most cases, accurate sensing is still required. This gripper limitation affects deep learning, as the gripper design has to be included in the grasping model. Naturally, a more complex gripper can complicate the grasp model, while a too simple gripper design is incapable of grasping the industrial parts. Moreover, the reality that often industrial part models are unavailable, hinders the learning and modelling process as well.

Grasping software

Recent state of the art in robot grasping utilizes or develops open source software that is distributed free of charge, but without warranty or liability to its use. In context to industrial applications this might pose difficulties as the understanding of the software and its integration in a production line can take considerable effort in resources and time. Moreover, support from the initial developers is not guaranteed and reliance on other software packages and hardware systems, such as sensors or processing platforms, has most likely not been tested. This affects the compliance to required safety standards even if risk assessments are carried out. Even if an industrial company would integrate a (deep learning-based) grasping model, pretrained models might not be suitable for the custom objects that a company's robot should handle. Training new models or extending existing models might be too time-consuming or simply ineffective due to an inaccessibility to computational resources (e.g., computing clusters). The need for a standard toolkit with respect to robotics and deep learning is therefore evident, effectively bridging the gap between academic research and industrial applications.

1.2 GRASPING AND OBJECT MANIPULATION

Robotic object grasping and manipulation are commonplace in industrial production. For high-throughput assembly lines, fast and accurate grasping is typically established by custom-made bulk feeders that position and align the part to a specific pose [30]. A robot then grasps the object, based on a predefined grasp pose, with gripper or end-effector design that is tailored to the shape of the object. For low-volume manufacturing, this approach is however unsuitable, as it requires flexibility and reconfigurability following task changes. Ongo-

ing research efforts towards (bin) picking therefore disregard feeders and utilize sensing to determine object and grasp pose for object handling [31], for example by deep convolutional neural networks (CNNs) trained on large datasets of grasp attempts in a simulator or on a physical robot [17, 20]. As autonomous grasping and object manipulation with robots sees many similarities to the computer vision problem of pose estimation, we first introduce the problem statement of grasp representations used in grasping literature (see Fig. 1.1 and Fig. 1.2). Following, an overview of different grasp detection methods is given, divided by their sensor modality; RGB-D or point clouds.

1.2.1 Problem statement

To a robot, the task of grasping is the successful determination of an end-effector or gripper pose $\mathbf{P}_g$, that leads to a secure and stable object grip and subsequent object lift-off from a surface. In this, the gripper pose is defined as $\mathbf{P}_g = (\mathbf{R}_g, \mathbf{t}_g) \in SE(3)$, where the rotation $\mathbf{R}_g \in SO(3)$ defines the grasp axis and direction of approach and the translation $\mathbf{t}_g \in \mathbb{R}^3$ defines the center location in between the gripper fingers (see Fig. 1.2). Sahbani et al. [32] give a detailed overview of terminology conventionally used in work related to robotic grasping, which defines the stability conditions such that the sum of all external forces and moments acting on a grasped object are zero. Furthermore, a stable grasp is defined as one that can withstand minor disturbance forces to the object or the end-effector and allows the system to restore to its original configuration. Other than stability, task-compatibility and adaptability to unseen objects are important parameters as well [32, 33]. In literature, a wide variety of popular grasp representations are presented, some which combine both image and depth information [33]. Earlier works defined a grasp as 2D point coordinates $[x, y]$ in an image (see Fig. 1.1) or as 3D point coordinates $[x, y, z]$ in the robot workspace. The obvious limitation is the inability to address gripper orientation, opening width and angle of approach. Therefore, later approaches [34] used 7D oriented rectangular-box representations in robot workspace $[x, y, z, roll, pitch, yaw, width]$ and 5D on the image plane $[x, y, \theta, width, height]$ (see Fig. 1.1). These approaches are analogous to object detection and localization frameworks and hence can be translated to grasping, as grasping is a detection problem in itself. Some of the later works introduced depth besides image data as input and hence used a 5D representation $[x, y, z, \theta, width]$ that dropped the height of the gripper [33]. In addition, other approaches regress grasps over point clouds [35–37], parametrized in object or camera coordinate frame, i.e., grasps are defined as 6-DOF poses of the gripper $\mathbf{P}_g$ relative to the camera $\mathbf{P}_c = (\mathbf{R}_c, \mathbf{t}_c) \in SE(3)$, by transformation of an (estimated) object pose $\mathbf{P}_o = (\mathbf{R}_o, \mathbf{t}_o) \in SE(3)$. These approaches propose grasps that are not constrained in a single plane relative to the camera, and can be directly used as goal poses for the robot. However, these are difficult to regress directly, require high dimensional features for learning and usually require post-refinement steps.

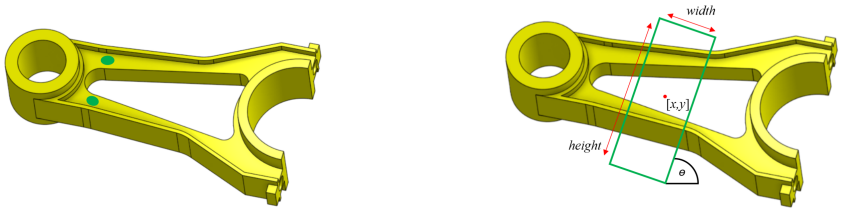

FIGURE 1.1 Grasp representation in image space: pixel coordinates $[x, y]$ (left, green circles) and oriented bounding box $[x, y, \theta, width, height]$ (right, green square). The yellow object is a connecting rod for a piston engine.

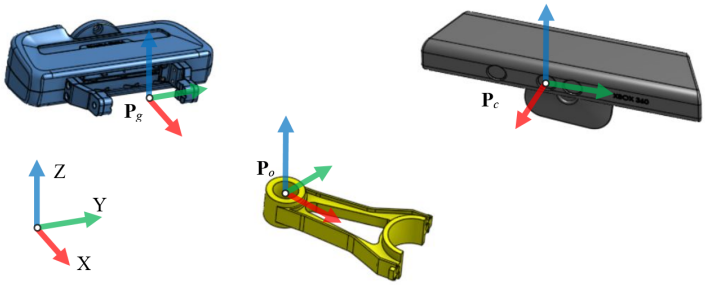

FIGURE 1.2 Grasp representation in 3D Cartesian space, with 6D poses for the gripper of the robot $\mathbf{P}_g$, the camera $\mathbf{P}_c$ and the object to be grasped $\mathbf{P}_o$.

1.2.2 Analytical vs. data-driven approaches

Early research in robotic grasping utilized analytical approaches of calculating robot kinematics and dynamics to deal with the constraints on the 3D geometry of the object to be grasped [16, 33]. These techniques aim to satisfy force-closure, form-closure or task-specific geometric constraints to find feasible contact points for a particular object and manipulator configuration. The majority of these works is concerned with finding and parameterizing the surface-normals on various flat faces of an object and then testing the force-closure condition by subjecting the angles between these normals to be within certain thresholds. Generally, a force-closed grasp is one in which the end-effector can apply required forces on the object in any direction, without letting it slip or rotate out of the gripper. Later analytical methods argued on the optimality criterion of the grasp quality [32]. This means that a metric should decide on the quality of the force-closure achieved with a certain grasp, i.e., how close the grasp is to losing its force-closure, given a particular object geometry and hand configuration. Recently, it has been studied that these analytical modeling methods and metrics alone were not good measures of grasps as they do not adapt well to the challenges that are faced during execution, such as uncertainties in dynamic behavior and the unstructured environment [38]. Therefore, in the last decade, there was a general push towards machine-learning approaches which present a more abstract, easy-

to-test and indirect approach of evaluating grasp success on a large variety of objects, grippers and environmental conditions. The shift from complex mathematical modeling of the grasping itself, towards the indirect mapping of perceptual features with grasp-success was made possible by the availability of high quality 3D cameras and depth sensors, increasingly powerful computational resources and a substantial amount of invaluable research in deep neural networks and transfer learning [33]. The biggest advantage these methods provided were the vast possibilities for testing in simulation and real execution, using real and synthesized data. These methods don't explicitly guarantee equilibrium, dexterity or stability but the premise of testing all criteria, based solely on sensor data, object representation and carefully designed object features provides a convenient way for studying grasp synthesis [38].

1.2.3 Grasp detection with RGB-D

Known objects

Grasping known objects has been researched extensively, as it is a direct extension of object detection and object pose estimation. Different methods are used to extract information from the objects such as 2D point correspondences (e.g., SIFT, SURF, ORB), templates (e.g. HOG, surface normals, moments) or patches (e.g., image regions, super-pixels) to represent objects in intermediate representations for correspondence matching [39]. Hinterstoisser et al. [40] pioneered the research in template matching by providing a complete framework for creating robust templates from existing 3D models of objects by sampling a full-view hemisphere around an object. In addition, they introduced the Linemod dataset, consisting of 1100+ frame video sequences of 15 different household objects varying in shape, color and size along with their registered meshes and CAD models. Other research includes PoseCNN [41], which combines feature and template methods, ConvPoseCNN [42] that improves PoseCNN by a fully-convolutional architecture, HybridPose [43] and PVNet [44].

Similar objects

This class of grasp-detection methods are aimed at objects that are similar i.e., a different/unseen instance of a previously seen category of objects. For example, all cups with slight intraclass variation belong to a single category of objects and all shoes belong to another. These methods learn a normalized representation of an object category and transfer grasps using sparse-dense correspondence between normalized 3D representation of the category and the partial-view object in the scene [16]. NOCS [45] presents an initial benchmark in this category by formulating a canonical-space representation per category using a vast collection of different CAD models for each class. A color-coded 2D perspective projection of this space (NOCS map) is then used to train a Mask-RCNN based network [46], in order to learn correspondences from RGB-D images of unseen instances

to this NOCS map. These correspondences are later combined with a depth-map to estimate the 6D pose and size of multiple instances per class.

Novel objects

Regarding novel objects, there is no existing knowledge of object geometry and grasps are estimated directly from image and/or depth data. The majority of these methods use geometric properties inferred from input perceptual data, as a measure of grasp success [16]. Most methods are developed in an end-to-end fashion, learning from a database of grasps on a large number of different object models. These grasps are sampled exhaustively around the objects and are either evaluated on classical grasp metrics, such as the epsilon quality metric [47], manually annotated with their success measures by humans or tested with real execution [48, 49]. The premise of these methods lies in the later training stage, where a deep neural network learns to produce robust grasps in general. The emphasis is on learning a robustness function that ranks a candidate grasp for various quality metrics.

Dex-Net 1.0 [48] and Dex-Net 2.0 [49] are two pioneering works that utilized this strategy and created large datasets of 3D object models for learning objective functions that minimize grasp failure, in presence of object and gripper position uncertainties and camera noise. Enforcing constraints like collision avoidance, approach-angle threshold and gripper-roll threshold, these methods provide a baseline for correlating object geometry from RGB-D images [48] or point clouds [49] with grasp robustness. Extensions of Dex-Net include suction grasping in Dex-Net 3.0 [50] and ambidextrous grasping (i.e., two or more heterogeneous grippers) in Dex-Net 4.0 [51]. Finally, [52] and [53] are excellent examples that use a fully-convolutional architecture for grasp detection.

PVN3D

To obtain a better understanding of the complexity of deep learning-based pose estimation, one network is explained in more detail. In particular, PVN3D [54] is a recent, deep point-wise 3D key-points-based voting network for 6-DOF pose estimation and utilizes single RGB-D images for inference. The network consists of the following separate blocks and their functionalities:

1. **Feature-extraction** contains two separate steps as follows:
   - PSPNet-based [55] CNN layer for feature-extraction in RGB images.
   - PointNet++-based [56] layer for geometry extraction in point clouds.

   The output features from these two are then fused by:
   - DenseFusion [57] layer for a combined RGB-D feature-embedding.
2. **3D key-point detection** comprises of shared Multi-Layered Perceptrons (MLP) with a semantic-segmentation block and uses the features extracted by the previous block to estimate an offset of each visible point from the target

key-points in Euclidean space. The points and their offsets are then used for voting candidate key-points. The candidate key-points are then clustered using Meanshift clustering [58] and cluster-centers are casted as key-point predictions.

3. **Instance semantic-segmentation** contains two modules sharing the same layers of MLPs as those of the 3D key-point detection block. These are, a semantic segmentation module that predicts per-point class label and a center-voting module to vote for different object centres in order to distinguish between object instances in the scene. The center-voting module is similar to the 3D key-point detection block in that it predicts per-point offset, which in this case votes for the candidate center of the object rather than the key-points.
4. **6 DOF pose estimation** finally executes a least-squares fitting between key-points predicted by the network and the corresponding key-points.

1.2.4 Grasp detection with point clouds

The detection of grasps with point cloud data presents a unique set of challenges uncommon to RGB and RGB-D-based methods, such as the small scale of available datasets, the high dimensionality and the unstructured nature of point clouds. Nevertheless, point clouds are a richer representation of geometry, scale and shape as they preserve the original geometric information in 3D space without any discretization [59]. Perhaps the most widely used point cloud aggregation method in applications of grasp detection and pose estimation are PointNet [60] and PointNet++ [56]. These two methods revolutionized geometry-encoding in point clouds by preserving permutation invariance. Point clouds are an inherently unordered data-type and any kind of global or local feature representation should not change the way they are ordered. To deal with this problem, most techniques convert point clouds into other discrete and ordered forms, for example, voxel-grids, height-maps, octomaps, surface-normals or 2D-projected gradient-maps, before aggregating into a final compact representation. PointNet, PointNet++ and their later modifications overcame this and paved way for direct useage of point clouds in order to be used with other deep-learning frameworks.

Due to a better scene understanding and geometry aggregation, point clouds processed with deep neural networks have led to a whole new line of pose estimation methods. These methods filled the undesirable gap that existed in RGB/RGB-D techniques, that required data collection, training and evaluation in image coordinates in the form of 2D or 3D bounding boxes. Moreover, the need to transfer poses or grasps from image to world coordinates was removed, as these techniques could recover full 6-DOF pose of the object without having to employ any post-processing on the depth channel or without learning to estimate depth. In this regard, methods have been developed that follow the same categorization as RGB/RGB-D based pose estimation, such as correspondence-based [61–63], template-based [64] or voting-based [44, 54] techniques.

Methods that generate grasps directly from point cloud data, circumvent the need for estimating the pose of an object in the scene or any prior knowledge about the shape or canonical representation of a class of objects. With point clouds, these grasps can be recovered in 6-DOF without constraining the gripper to move along the image plane, as with the RGB-D-based detectors. Noteworthy examples of such networks are [35], [65], GG-CNN [66], PointNetGDP [36] and 6-DOF GraspNet [37].

6-DOF GraspNet

To obtain a better understanding of the complexity of deep learning-based grasp detection, one network is explained in more detail. In particular, 6-DOF GraspNet [37] generates grasps directly as output of the network, from 3D point clouds observed by a depth camera. The network has two main components and a third refinement step as follows:

1. **Grasp-sampling** utilizes a Variational Auto-encoder (VAE) [67] with PointNet++ encoding layers, which learns to maximize the likelihood of ground-truth positive grasps given a point cloud.
2. **Grasp-evaluation** utilizes an adversarial network to measure the probability of success of a grasp. Negative examples are included from the training data and by generating hard-negatives through random perturbation of positive grasp samples.
3. **Iterative grasp-pose refinement**: The grasps rejected by evaluation are close to success and can undergo an iterative refinement step. This transformation is found by taking a partial derivative of the success function with respect to the closest successful grasp.

1.3 GRASP EVALUATION

As many different approaches towards robotic grasping exist, in this section we evaluate two methods that are the state of the art in their respective approach; pose estimation with PVN3D and grasp detection with 6-DOF GraspNet. In addition, we address the metrics by which grasping should be evaluated.

1.3.1 Metrics

From an industrial perspective, the evaluation and measures for grasping can be simplified to either a successful grasp, lift and put-down, within a specified placement and pose range. With the traditional industry approach of pre-specified grasp locations and object feeders, this protocol provides a suitable evaluation. However, with the novel approach of learning-based grasp generation without object-designed grippers, multiple factors influence the success of grasping. Here, we aim to address these different factors and cover the most popular metrics for grasp evaluation. This includes pose estimation metrics for grasping

objects at a pre-defined pose, grasp candidate selection for grasp detection models, grasp evaluation metrics for grasp detection models and the influence of the gripper on grasping.

For evaluating pose-estimation different pose-error functions can be utilized [68]. One popular metric is the Average Distance of Model Points (ADD) and the Average Closest Point Distance (ADD-S) [40]. Given a ground truth object rotation $\mathbf{R}$, translation $\mathbf{T}$, predicted rotation $\tilde{\mathbf{R}}$ and translation $\tilde{\mathbf{T}}$, ADD computes the average of the distances between pairs of corresponding 3D points $\mathbf{x}$ on the model transformed according to the ground truth pose and the estimate pose:

$$ADD = \underset{\mathbf{x} \in M}{\text{avg}} \|(\mathbf{R}\mathbf{x} + \mathbf{t}) - (\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{T}})\|_2, \tag{1.1}$$

where M denotes the set of 3D model points and $\| \cdot \|_2$ denotes the $L2$-norm.

For objects that are symmetric in the plane along the principal axis of rotation, the rotation along this axis in the predicted pose could be ambiguous by 180 degrees since similar points would repeat after every 180 degrees and the correspondences are spurious. This rotational ambiguity is by-passed by the ADD-S metric where only the minimum of all the distances between a pair of points is considered and the average of this distance then gives the error:

$$ADD-S = \underset{\mathbf{x}_1 \in M}{\text{avg}} \min_{\mathbf{x}_2 \in M} \|(\mathbf{R}\mathbf{x}_1 + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x}_2 + \tilde{\mathbf{T}})\|_2. \tag{1.2}$$

Based on an estimated object pose, a robot grasp pose is defined to which the robot moves for grasping. This approach assumes that the pre-defined grasp pose is suitable for grasping the object with the available gripper. Novel learning-based approaches for grasp detection generate a grasp pose directly from object perception data, thereby bypassing the suitability check of the grasp pose. Moreover, typically a large number of grasp candidates are generated per object and only a single best grasp should be selected for execution (see Fig. 1.5). One open question in grasping research is therefore how the grasp candidates are sampled for testing and how they should be evaluated. In order to generate training data for a robot that is large enough in quantity, varied enough for generalization and provides an accurate enough representation of task constraints, some efficient heuristic measures are needed to search through a space of thousands of potentially viable grasps [69]. Even after the initial selection of these candidates, effective evaluation of the grasps and the metrics for a robot's performance on them determines the usefulness of the data and the robustness of the grasp algorithm being trained on [70]. The commonly used grasp sampling techniques are analyzed by Eppner et al. [69] and are broadly categorized into object geometry-guided sampling, (non-)uniform sampling [71], approach-based sampling and antipodal sampling [72]. Eppner et al. [69] then devised a few intuitive metrics of comparing these sampling methods, i.e., grasp coverage, robustness and precision. In simulation more than 317 billion grasps on 21 YCB-dataset objects [73] are generated and the successful 1 billion grasps

out of these are evaluated. It is concluded that uniform samplers have better grasp coverage because of minimal constraints, with the trade-off for efficiency. On the other hand, heuristics like approach-based sampling or antipodal sampling are efficient but might not entirely capture all possible grasps. Moreover, it was also found that antipodal grasps have higher coverage and find more robust grasps. Precision is quite low for both uniform and approach-based methods, while being significantly higher for antipodal methods. Non-uniform or geometry-based approaches, consistently perform poor on all three metrics.

In both cases, i.e., grasping based on pose estimation and grasp detection, grasping is evaluated by a grasp success rate for a single object or an average grasp success rate for an object set, while including the number of attempted and failed grasps. Another common method of evaluation is an offline metric using the Cornell grasping dataset [34] or Jacquard [74] dataset, where a predicted grasp is successful if it has an intersection-over-union (IoU) greater than 25% and is within 30° angle, as compared to the ground-truth grasp. However, this metric is susceptible to producing large numbers of false-positive and false-negative detections, due to the sparse labelling of the dataset, and low requirements for considering a match. Recent work has included resistance to disturbances to the evaluation metrics, by expected wrench resistance or the ability to resist task-specific forces and torques, such as gravity, under random perturbations [51]. In addition, it reports the mean picks per hour that the physical robot system achieves (i.e., the number of successful grasps per hour).

It is crucial to note the importance of the gripper used for the evaluation. A perfect pose estimate or grasp detection does not imply a perfect grasp, nor does it suggest which type of gripper would provide best results. Even though many works integrate the gripper as part of the grasp model, usually these are limited to only modelling the gripper width and depth [35, 36, 75], thereby only ensuring that the generated grasps comply to the gripper's dimensions and configuration. Compliant grippers [29] could provide an alternative and potentially improve grasp success rate, however, at the expense of a higher placement uncertainty, as, similar to many grasp detection methods, the object pose inside the gripper would be unknown.

1.3.2 Pose estimation with PVN3D

Data collection and training

Data collection utilizes simulation only, as it provides easy fine-tuning of a variety of parameters, i.e., object placement and pose, lighting, etc. A set of ten objects are selected, consisting of parts from a (3D printed) Cranfield assembly benchmark [76] and from a Diesel engine assembly use case (see Fig. 1.3). A Kinect v1 camera is simulated in Gazebo and publishes RGB and depth images (i.e., 640 by 480 pixels) and all objects are simulated using their standard polygon mesh files generated from CAD models. The background is left out to

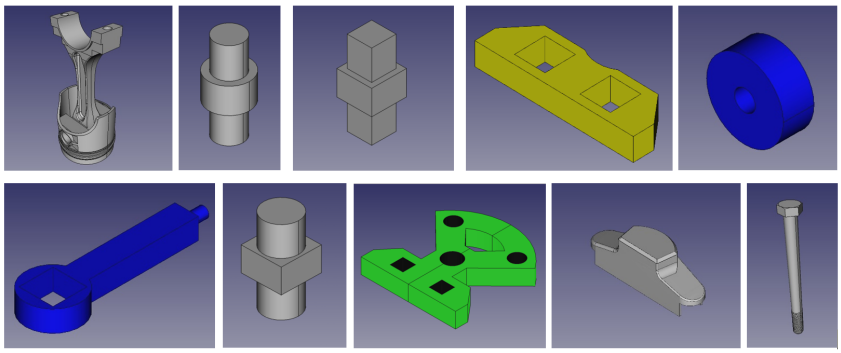

FIGURE 1.3 CAD dataset consisting of parts from the (3D printed) Cranfield assembly benchmark [76] and from a Diesel engine assembly use case. From top left to top right: piston, round peg, square peg, separator, pendulum head. From bottom left to bottom right: pendulum, shaft, face plate, valve tappet, shoulder bolt.

be a brightly-lit plain-grey room with no walls and the objects always rest on the floor in their most stable equilibrium pose. No dense background clutter or non-dataset objects are added to the environment. A moderately dense clutter of all the objects in the dataset is created around the origin in the simulation environment. Hemisphere-sampling, as described in [40], is carried out around each clutter-set, with steps in yaw, pitch and scale for the simulated camera. For each sample, the dataset records RGB and depth images of the scene, a grey-scale image with binary mask of each object with the respective class label and the ground truth pose of each object in camera- and world coordinates.

This generates a total of 2304 collected data samples, from which a 75%-25% train-test split is applied, in order to evenly cover all possible pitches, yaws and scales in both test and training dataset. A joint multi-task training is carried out for 3D key-point detection and instance semantic-segmentation blocks. Firstly, the semantic-segmentation module facilitates extracting global and local features in order to differentiate between different instances, which results in accurate localization of points and improves the key-point offset reasoning procedure. Secondly, learning for the prediction of key-point-offsets indirectly learns size-information as well. This helps distinguish objects with similar appearance but different size. Three key-point variations (i.e., 8, 12 and 16 key-points) were trained for a total of 70 epochs with batch size of 24 as recommended by the authors. The training was carried out on 4 Nvidia V100 GPUs and takes around 5-7 hours for the given batch-size, number of epochs and training-dataset.

Results

Results of object pose estimation are reported in terms of area under the accuracy-threshold curve where the threshold for the ADD and ADD-S metric (see Eqs. 1.1 and 1.2) is varied from 0 to 10 cm. These results are reported in Table 1.1.

TABLE 1.1 Area under curve (AUC) for accuracy-threshold curve for the ADD and ADD-s metric on the CAD dataset.

Object	16 key-points		12 key-points		8 key-points	
	ADD	**ADD-s**	**ADD**	**ADD-s**	**ADD**	**ADD-s**
Piston	**97.76**	**98.16**	97.57	98.02	97.21	97.85
Round peg	97.35	97.35	**97.4**	**97.4**	97.05	97.05
Square peg	**97.12**	**97.12**	96.92	96.92	96.32	96.32
Pendulum	**96.09**	**96.09**	95.68	95.68	95.31	95.31
Pendulum-head	**97.27**	**97.27**	97.05	97.05	96.73	96.73
Separator	**96.4**	**96.4**	96.24	96.24	95.95	95.95
Shaft	**97.93**	**97.93**	97.68	97.68	97.41	97.41
Face-plate	**96.16**	**96.16**	96.15	96.15	95.88	95.88
Valve-tappet	91.7	91.7	**92.48**	**92.48**	91.58	91.58
Shoulder-bolt	**93.5**	**93.5**	93.08	93.08	91.4	91.4
Average	**96.13**	**96.17**	96.03	96.07	95.48	95.55
Inference time [sec]	1.98		1.75		1.5	
GPU memory [MB]	2775		2544		2343	

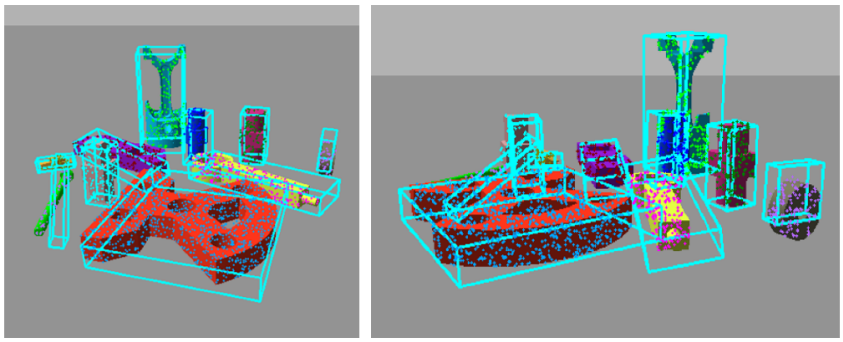

FIGURE 1.4 Images from the pose estimation result with PVN3D [54]. The 3D poses are shown as bounding boxes projected on the RGB image.

Fig. 1.4 depicts the pose estimation results for the object set, where each object is overlaid with an estimated pose, depicted by a projected bounding box (light-blue). These results demonstrate that accurate object pose estimation can be achieved with PVN3D in simulation. Unfortunately, both the inference time and model size are large (>1.5 seconds and >2 gigabytes, see Table 1.1), meaning that real-time feedback for control is not possible.

1.3.3 Grasp detection with 6-DOF GraspNet

Data collection and training

The grasp data is collected in a physics simulation, based on grasps done with a free-floating parallel-jaw gripper and objects in zero gravity. Objects retain a uniform surface density and friction coefficient and the grasping trial consists of closing the gripper in a given grasp-pose and performing a shaking motion. If the object stays enclosed in the fingers during the shaking, the grasp is labelled as positive. Grasps are sampled based on object geometry, sampling random points on object mesh surface to align approach axis with the normal at each of these points. Grasps are performed on a total of 206 objects from six categories in ShapeNet [77]. From a total of over 10 million candidate grasps, 2 million successful grasps (19.4%) are generated. This data collection and training is done as part of the original work in [37], by utilizing 8 NVIDIA V100 GPUs.

Results

For evaluation, grasp detection and execution is simulated on a Franka Emika Panda[1] robot with Kinect v1 camera in Gazebo using ROS[2]. MoveIt[3] and ros_control[4] are utilized as they provide a generic and robot-agnostic framework to interface with any real or simulated robot. Results of grasp detection is reported in terms of total number of grasps generated per object and the percentage of grasps that passed the initial planning step and grasp execution test (Grasp test in Table 1.2). The initial planning step excludes grasps that are infeasible, by limiting the yaw and pitch angles of grasp candidates to $[-90, 90]$ and $[-90, 45]$ degrees, respectively. Fig. 1.5 depicts the detected grasps before (middle figure) and after (right figure) grasp candidate filtering for an industrial part (piston). Table 1.2 reports results on grasp generation and grasp testing for a subset of objects. These results demonstrate that, while grasps can be generated with 6-DOF GraspNet in simulation, stable grasping is difficult to achieve. Reasons for this are likely due to the simple candidate filtering approach and the fact that simulation is a poor representation of the real world. Object and gripper properties, such as friction and inertia, are hard to model, leading to an unrealistic simulation environment and therefore unreliable results.

1.3.4 Pick-and-place results

While previous sections reported results on object pose estimation and grasp detection and execution, here results are reported on robotic pick-and-place actions. Again, grasping is simulated on a Franka Emika Panda robot in Gazebo. A Kinect v1 camera and PVN3D are chosen to estimate an object pose and

1. https://franka.de
2. https://www.ros.org
3. https://moveit.ros.org
4. https://wiki.ros.org/ros_control

TABLE 1.2 Results from grasp simulation experiments from grasp detection with 6-DOF GraspNet [37], reported based on the number of generated grasps and grasp success: % of grasps that ended in successful holding of the object.

Object	Generated grasps	Grasp test [%]
Piston	40	42.5
Round peg	22	77.3
Square peg	19	47.4
Pendulum	36	8.3
Separator	39	5.1
Shaft	21	42.9

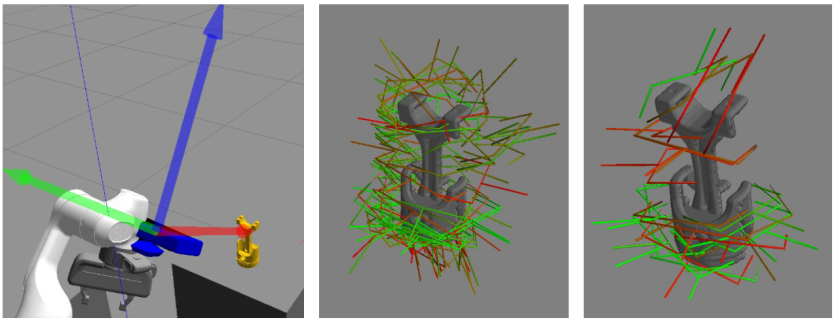

FIGURE 1.5 Simulation (left) of robot (Franka Emika Panda), camera (Kinect) and industrial part (piston). Detected grasp poses before (middle) and after (right) grasp candidate filtering are detected from point clouds by 6-DOF GraspNet [37].

execute a grasp action, followed by a placement action in a different location (see Fig. 1.6). A grasp pose is therefore predefined for each object and transformed from the estimated object pose. Table 1.3 reports results for the pick and place actions, for a selection of objects, based on the number of generated grasps, the percentage of successful grasps and the percentage of grasps that could be accurately placed upright, with placement error < 5cm. These results demonstrate that, while an object pose can be accurately estimated (see Section 1.3.2), this does not necessarily lead to a high grasp success (Grasp test in Table 1.3) or a high placement success (Placement test in Table 1.3). A likely cause of this is, again, that simulation is a poor representation of the real world, where physical properties (friction, inertia, etc) are hard to model correctly.

Nevertheless, from all (simulated) evaluations of the different grasping approaches, valuable conclusions can be drawn. For example, the selection of a suitable grasp detector depends on several factors, such as the object models and their availability for training, suitability of the grasp detector for real-time control and effort needed to select a single optimal grasp. This information can be obtained from simulated experiments, with realistic objects similar to their real world cases.

TABLE 1.3 Results from pick-and-place simulation experiments from pose estimation with PVN3D [54]. The experiments are reported based on the number of generated grasps, grasp test: % of grasps that ended in successful holding of the object, placement test: % of grasps that were able to place the object upright with an (x, y) error < 5cm and the average placement error in [cm], [deg].

Object	Generated grasps	Grasp test [%]	Placement test [%]	Placement error [cm], [deg]		
				x	y	yaw
Piston	88	83.0	60.2	0.9	2.5	25
Round peg	108	76.9	53.7	0.2	1.3	-
Square peg	108	72.2	42.6	0.9	1.4	46
Pendulum	54	87.0	61.1	0.4	2	2
Separator	58	50.0	36.2	1.1	3.8	2
Shaft	108	74.1	51.9	0.4	1.5	44

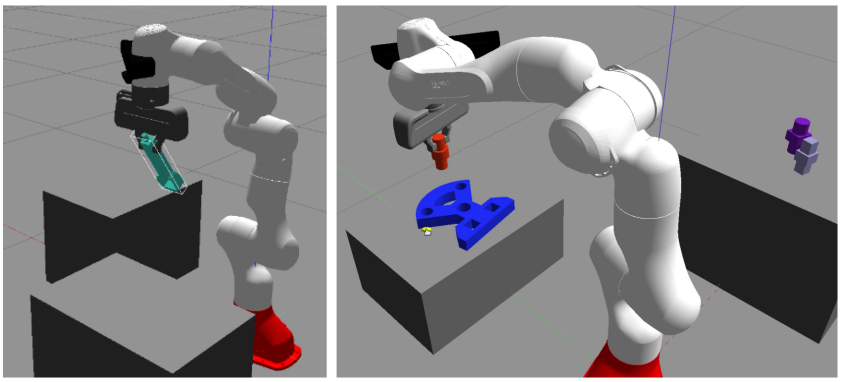

FIGURE 1.6 Simulation of robot grasping for a pick and place task (left) and a peg-in-hole manipulation task (right). Object pose is estimated from point cloud data by PVN3D [54].

1.4 MANIPULATION BENCHMARKING

Object manipulation represents the step after object grasping, where an object in the gripper or hand is manipulated towards a certain task or goal (see Fig. 1.6). Accurate grasps and grasp poses are crucial for this, which implies that neither the object falls out of the gripper nor slips too much inside the gripper. With an accurate estimate of initial object-pose, the in-hand pose of the object is assumed to be within certain constraints and the final placement of an object can be done with a reasonable certainty [78]. Robotic object manipulation tasks other than placement, such as peg-in-hole, hole-on-peg and bolt screwing require additional sensors and advanced control methods to be successfully executed [6]. State of the art works that provide a general framework and benchmark for the

aforementioned assembly tasks, utilizing motion-priors like spiralling around the hole for peg-insertion tasks and back-and-forth spinning for screwing tasks, can be found in [79] and [80]. These methods are thoroughly tested on various combinations of both compliant-arm and compliant-hand and fingers, both with and without contact sensors. The benefits of using various force-profiles as cues for driving the manipulation towards a more accurate and robust assembly is described in detail. Recent work also combines visual and force sensing with deep reinforcement learning to teach policies for contact-rich manipulation tasks [21]. By self-supervision, a compact and multimodal representation of the sensory inputs is learned, which can then be used to improve the sample efficiency of policy learning. Evaluation of the method is done on a peg insertion task, which shows that it generalizes over varying geometries, configurations, and clearances, while being robust to external perturbations. Bekiroglu et al. [81] developed a benchmark protocol specifically for the evaluation of robot grasping algorithms, describing in detail how to setup the workspace, object choices and placements, grasp execution and scoring. Robustness of grasps is taken into account by describing a series of motions after an object is grasped. The ACRV picking benchmark [82] is another benchmark and consists of a set of 42 common objects, a widely available shelf, and exact guidelines for object arrangement using stencils. A well-defined evaluation protocol enables the comparison of complete robotic systems, including perception and manipulation, instead of subsystems only. A benchmark for robot learning is RLBench [83], which features 100 completely unique, hand-designed tasks aimed to accelerate progress in a number of vision-guided manipulation research areas, including reinforcement learning, imitation learning, multi-task learning, geometric computer vision and few-shot learning.

Besides scientific works that push benchmarking forward in robotics, international organized challenges are another effort that drives scientific progress in robot object manipulation. Well-known initiatives are RoboCup@Work[5], Amazon Picking Challenge (APC), European Robotic League (ERL) - Professional[6] and the ADvanced Agile ProducTion (ADAPT) competition[7], and competitions organized at international events, such as the Robotic Grasping and Manipulation Competition[8] at the IEEE/RSJ International Conference on Intelligent Robots and Systems and the Assembly Challenge[9] at the World Robot Summit.

5. https://atwork.robocup.org
6. https://www.eu-robotics.net/robotics_league/erl-professional
7. https://metricsproject.eu/agile-production
8. https://rpal.cse.usf.edu/competition_iros2020
9. https://worldrobotsummit.org/en/wrs2020/challenge/industrial/assembly.html

1.5 DATASETS

Robotic grasping is a popular topic in research and several surveys can be found that compare different approaches [16, 32, 33, 38]. Most works present a large variation in objects such that the learned grasp model can handle most if not all objects in the dataset. In fact, the importance of the training data and its generation should not be underestimated as it has a crucial role in the learning of a suitable grasp model. Generating grasps for objects that are unknown should thus be captured by the variance in the dataset. Several works have tried to accommodate this by either a dataset with large number of objects [74, 84] or by creating a dataset where all objects cover a statistical large variation in geometry [85]. This variation of objects in a dataset can also be identified from their type, as in many cases household objects, kitchen utensils and tools are represented. Few works target specifically other part categories, for example, industrial parts as in [86]. The choice of object naturally needs to follow the capabilities of a robot gripper (e.g., object weight, size and shape) and its configuration (e.g., parallel gripper or multi-finger) which affects the model and representation of a grasp as well.

One important property to distinguish between different datasets is their method of generating training data; real or simulated. Early approaches used real objects and high definition sensor systems to record datasets leading to vast amounts of data (typically few gigabytes) that are freely available to utilize [20, 35, 40, 73, 75]. This implies that the dataset is fixed and a pretrained grasp model only takes the objects in the dataset into account, unless more images of real objects are added. From an industrial perspective, the question whether pretrained models are suitable for a company's particular industrial parts, is difficult to answer. Moreover, training grasp models based on the industrial parts themselves might not be possible simply due to a lack of resources or due to the unavailability of (digital) part models. The resources required to commit to such effort are, for example, human effort and the required technical skills, and the computational infrastructure. Simulated datasets are a more recent approach, where digital tools (e.g., CAD or physics simulators) are utilized to generate object models. Training data can then be prepared by importing a CAD model into the simulation environment and generating different images or point clouds under varying conditions (e.g., viewpoint, lighting, color, noise, etc.). The advantage is that only a CAD model is required and the type, properties and number of objects can be tuned by changing simulation parameters. Table 1.4 lists grasping datasets that utilize simulation to generate objects, instead of images or point clouds from real objects.

Noteworthy examples in Table 1.4 are the following. The approach of Zhao et al. [86] particularly focused on industrial parts and the industrial applications of assembly and palletizing. The dataset includes more than 1000 CAD models of industrial parts, such as gears, brackets and screws, collected from an online hardware shop. The Evolved Grasping Analysis Dataset (EGAD) comprises over

TABLE 1.4 Overview of grasping datasets that utilize simulation for the generation of object training data. The variables in the **Grasp representation** column are: for **EGAD**; object position [x, y, z], rotation around the vertical axis θ, desired width of the gripper w and grasp quality q, for **Jacquard**; center of a rectangle [x, y], its height and width [h, w] and its orientation relative to the horizontal axis of the image θ, for **Zhao et al**.; location of the grasp relative to the object's geometric center [x, y, z] and planar rotation of the gripper about an axis orthogonal to the table surface θ.

Dataset (**year**)	**Parts**	**Categories** (**number**)	**Simulator**	**Modality**	**Grasp representation**	**Gripper**
ACRONYM [84] (2020)	8872	household (2331)	FleX [89] PyRender [90]	Depth	6D	parallel
Dex-Net 4.0 [51] (2019)	1664	basic shapes, household	CAD	Depth	6D	suction, parallel
EGAD [84] (2020)	2000	geometrically diverse	-	Depth	[x, y, z, θ, w, q]	parallel
6-DOF GraspNet [37] (2019)	206	household (6)	FleX [89]	Depth	6D	parallel
Jacquard [74] (2018)	11000	household	PyBullet [91], Blender [92]	RGB-D	[x, y, h, w, θ]	parallel
Kappler et al. [87] (2015)	700	household, tools (87)	openRAVE [93]	RGB-D	6D	Barrett hand [28]
Veres et al. [88] (2017)	-	household (64)	V-REP [94]	RGB-D	6D	Barrett hand [28]
VR-grasping [95] (2018)	101	household (7)	PyBullet [91]	RGB-D	6D	2-finger
Zhao et al. [86] (2019)	1011	industrial parts	V-REP [94], PyBullet [91]	Depth	[x, y, z, θ]	parallel

2000 generated objects that are geometrically diverse and range from simple to complex shapes, with varying grasp complexity. A set of 49 diverse 3D-printable evaluation objects within the dataset are selected to encourage reproducible testing of robotic grasping systems. Dexterity Network (Dex-Net[10]) has seen several iterations that have focused on cloud-based data generation and training (Dex-Net 1.0 [48] with 10000 3D object models), gripper-specific models (Dex-Net 2.0 [49] and Dex-Net 3.0 [50]) and ambidextrous grasping (i.e., two or more heterogeneous grippers in Dex-net 4.0 [51]). As can be seen in Table 1.4, most works focus on parallel grippers and only few datasets consider variations, such as the BarettHand [87, 88] or a suction gripper [51]. Other differences between the datasets are the sensor modality (RGB-D vs. depth perception), the representation of a grasp and the simulation environment and physics engine utilized for realistic dynamic grasp simulation.

10.https://berkeleyautomation.github.io/dex-net

1.6 CONCLUSION

In this chapter we have provided an overview of state of the art approaches in robotic grasping, with particular emphasis on deep learning. In context to the application area of agile production, the overview addresses several requirements and limitations to be taken into account (Section 1.1), such as data collection and computational resources. The principal part of the chapter (Section 1.2) presents different techniques for grasp detection, divided by grasp detection with RGB-D data and point cloud data. Evaluation of robotic grasping and the potential follow step of manipulation is covered in Section 1.3 and Section 1.4, respectively, where two methods are selected for evaluation on custom objects. The simulated grasp experiments demonstrate the capabilities and limitations of current state of the art grasping approaches. Regarding object pose estimation, while an accurate object pose can be estimated, high resolution object (CAD) models are required and objects should not be (partially) occluded or in a cluttered scene. Regarding grasp detection, as a large number of grasp poses will be generated for each object, their suitability still needs to be assessed in a filtering step, to remove unwanted grasps or select a most suitable grasp. In both cases, similar drawbacks can be identified regarding data collection and computation time. Large datasets of object models are required for training, leading to long computation times for model generation. In addition, the inference time of the trained models is high, implying real-time control based on them is not possible. Finally, the chapter describes different measures used to evaluate pose estimation and grasp detection actions, as well as several protocols in benchmarking grasps that are common in the field. Following, in Section 1.5, we provide a list of different datasets that generate object models by utilizing simulation.

The state of the art in robot grasping is improving rapidly, with most algorithms and datasets provided open source. The benefits of deep learning, compared to traditional approaches in grasping, is eminent, as the manual step of feature engineering is avoided and, instead, a learning based approach provides the generation of a grasping model. Despite this, different aspects require further research to tackle their limitations. First, methods need to be investigated that can better grasp unknown objects. Potential solutions to this are the automatic generation of object datasets, based on the types of objects to be expected or where object models include a wide variation of object geometries. Second, the computation time for both training a grasping model and its inference, should be suitable for real world applications. Current state of the art models are heavy to train and execute, and typically utilize computing clusters. Considering the application area of agile production, such computation power cannot be accessed. Potential solutions to this could, again, target the data generation step, where objects to be grasped are only included in the dataset, or the object models contain wide variety in object geometry. To conclude this robotic grasping overview and analysis, deep learning as tool for robotic grasping offers great benefits, compared to traditional approaches in generating grasping models.

BIBLIOGRAPHY

[1] V. Villani, F. Pini, F. Leali, C. Secchi, Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications, Mechatronics 55 (2018) 248–266.

[2] E. Matheson, R. Minto, E. G. Zampieri, M. Faccio, G. Rosati, Human–robot collaboration in manufacturing applications: A review, Robotics 8 (4) (2019) 100.

[3] J. Wang, Y. Ma, L. Zhang, R. X. Gao, D. Wu, Deep learning for smart manufacturing: Methods and applications, Journal of Manufacturing Systems 48 (2018) 144–156.

[4] A. Kusiak, Smart manufacturing must embrace big data, Nature 544 (7648) (2017) 23–25.

[5] T. Brogårdh, Robot control overview: An industrial perspective, Modeling, Identification and Control 30 (3) (2009) 167.

[6] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: modelling, planning and control, Springer Science & Business Media, 2010.

[7] IFR, How connected robots are transforming manufacturing, Tech. rep., International Federation of Robotics, `https://ifr.org/papers` (2020).

[8] C. Zhang, W. Xu, J. Liu, Z. Liu, Z. Zhou, D. T. Pham, Digital twin-enabled reconfigurable modeling for smart manufacturing systems, International Journal of Computer Integrated Manufacturing (2019) 1–25.

[9] Z. Zhu, H. Hu, Robot learning from demonstration in robotic assembly: A survey, Robotics 7 (2) (2018) 17.

[10] E. De Coninck, T. Verbelen, P. Van Molle, P. Simoens, B. Dhoedt, Learning robots to grasp by demonstration, Robotics and Autonomous Systems 127 (2020) 103474.

[11] A. Angleraud, R. Codd-Downey, M. Netzev, Q. Houbre, R. Pieters, Virtual teaching for assembly tasks planning, in: IEEE International Conference on Human-Machine Systems (ICHMS), 2020, pp. 1–6.

[12] A. Hietanen, R. Pieters, M. Lanz, J. Latokartano, J.-K. Kämäräinen, AR-based interaction for human-robot collaborative manufacturing, Robotics and Computer-Integrated Manufacturing 63 (2020) 101891.

[13] S. El Zaatari, M. Marei, W. Li, Z. Usman, Cobot programming for collaborative industrial tasks: An overview, Robotics and Autonomous Systems 116 (2019) 162–180.

[14] A. Chowdhury, A. Ahtinen, R. Pieters, K. Vaananen, User experience goals for designing industrial human-cobot collaboration: A case study of franka panda robot, in: Nordic Conference on Human-Computer Interaction (NordiCHI), 2020, pp. 1–13.

[15] H. A. Pierson, M. S. Gashler, Deep learning in robotics: a review of recent research, Advanced Robotics 31 (16) (2017) 821–835.

[16] G. Du, K. Wang, S. Lian, Vision-based robotic grasping from object localization, pose estimation, grasp detection to motion planning: A review, arXiv preprint arXiv:1905.06658 (2019).

[17] J. Mahler, K. Goldberg, Learning deep policies for robot bin picking by simulating robust grasping sequences, in: Conference on Robot Learning, 2018, pp. 515–524.

[18] P. Wang, H. Liu, L. Wang, R. X. Gao, Deep learning-based human motion recognition for predictive context-aware human-robot collaboration, CIRP annals 67 (1) (2018) 17–20.

[19] ISO-15066:2016, Robots and robotic devices — collaborative robots, Standard, International Organization for Standardization (2016).

[20] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, The International Journal of Robotics Research 37 (4-5) (2018) 421–436.

[21] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, J. Bohg,

Making sense of vision and touch: Learning multimodal representations for contact-rich tasks, IEEE Transactions on Robotics 36 (3) (2020) 582–596.

[22] J. Bütepage, A. Ghadirzadeh, Ö. Ö. Karadag, M. Björkman, D. Kragic, Imitating by generating: Deep generative models for imitation of interactive tasks, Frontiers in Robotics and AI 7 (2020) 47.

[23] A. Ghadirzadeh, X. Chen, W. Yin, Z. Yi, M. Björkman, D. Kragic, Human-centered collaborative robots with deep reinforcement learning, arXiv preprint arXiv:2007.01009 (2020).

[24] ISO-10218-1/2:2011, Robots and robotic devices – safety requirements for industrial robots – part 1: Robots/part 2: Robot systems and integration, Standard, International Organization for Standardization (2011).

[25] R.-J. Halme, M. Lanz, J. Kämäräinen, R. Pieters, J. Latokartano, A. Hietanen, Review of vision-based safety systems for human-robot collaboration, Procedia CIRP 72 (2018) 111–116.

[26] L. Birglen, T. Schlicht, A statistical review of industrial robotic grippers, Robotics and Computer-Integrated Manufacturing 49 (2018) 88–97.

[27] M. Honarpardaz, M. Tarkian, J. Ölvander, X. Feng, Finger design automation for industrial robot grippers: A review, Robotics and Autonomous Systems 87 (2017) 104–119.

[28] Barrett Technology, BarrettHand - Multi-fingered programmable grasper, `https://advanced.barrett.com/barretthand`, accessed on 23.11.2020 (2020).

[29] M. Netzev, A. Angleraud, R. Pieters, Soft robotic gripper with compliant cell stacks for industrial part handling, IEEE Robotics and Automation Letters 5 (4) (2020) 6821–6828.

[30] V. Limère, H. V. Landeghem, M. Goetschalckx, E.-H. Aghezzaf, L. F. McGinnis, Optimising part feeding in the automotive assembly industry: deciding between kitting and line stocking, International Journal of Production 50 (15) (2012) 4046–4060.

[31] A. Hietanen, J. Latokartano, A. Foi, R. Pieters, V. Kyrki, M. Lanz, J.-K. Kämäräinen, Benchmarking 6D object pose estimation for robotics, arXiv preprint arXiv:1906.02783 (2019).

[32] A. Sahbani, S. El-Khoury, P. Bidaud, An overview of 3D object grasp synthesis algorithms, Robotics and Autonomous Systems 60 (3) (2012) 326–336.

[33] S. Caldera, A. Rassau, D. Chai, Review of deep learning methods in robotic grasp detection, Multimodal Technologies and Interaction 2 (3) (2018) 57.

[34] Y. Jiang, S. Moseson, 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.

[35] A. ten Pas, M. Gualtieri, K. Saenko, R. Platt, Grasp pose detection in point clouds, The International Journal of Robotics Research 36 (13-14) (2017) 1455–1473.

[36] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, J. Zhang, PointNetGPD: Detecting grasp configurations from point sets, in: IEEE International Conference on Robotics and Automation (ICRA), 2019, pp. 3629–3635.

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

[38] J. Bohg, A. Morales, T. Asfour, D. Kragic, Data-driven grasp synthesis—a survey, IEEE Transactions on Robotics 30 (2) (2013) 289–309.

[39] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

[40] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, 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.

[41] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes, arXiv preprint arXiv:1711.00199 (2017).
[42] C. Capellen, M. Schwarz, S. Behnke, ConvPoseCNN: Dense convolutional 6D object pose estimation, arXiv preprint arXiv:1912.07333 (2019).
[43] C. Song, J. Song, Q. Huang, Hybridpose: 6D object pose estimation under hybrid representations, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 431–440.
[44] S. Peng, Y. Liu, Q. Huang, X. Zhou, H. Bao, PVNet: Pixel-wise voting network for 6DOF pose estimation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4561–4570.
[45] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, L. J. Guibas, Normalized object coordinate space for category-level 6D object pose and size estimation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2642–2651.
[46] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961–2969.
[47] F. T. Pokorny, D. Kragic, Classical grasp quality evaluation: New algorithms and theory, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 3493–3500.
[48] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, K. Goldberg, Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards, in: IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1957–1964.
[49] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, K. Goldberg, Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, arXiv preprint arXiv:1703.09312 (2017).
[50] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, K. Goldberg, Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning, in: IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1–8.
[51] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, K. Goldberg, Learning ambidextrous robot grasping policies, Science Robotics 4 (26) (2019).
[52] J. Redmon, A. Angelova, Real-time grasp detection using convolutional neural networks, in: IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 1316–1322.
[53] D. Guo, T. Kong, F. Sun, H. Liu, Object discovery and grasp detection with a shared convolutional neural network, in: IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 2038–2043.
[54] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, J. Sun, PVN3D: A deep point-wise 3D keypoints voting network for 6DOF pose estimation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 11632–11641.
[55] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2881–2890.
[56] C. R. Qi, L. Yi, H. Su, L. J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, in: Advances in Neural Information Processing Systems, 2017, pp. 5099–5108.
[57] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, 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.
[58] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.

[59] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3D point clouds: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

[60] C. R. Qi, H. Su, K. Mo, L. J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 652–660.

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

[62] Q.-H. Pham, M. A. Uy, B.-S. Hua, D. T. Nguyen, G. Roig, S.-K. Yeung, LCD: Learned cross-domain descriptors for 2D-3D matching., in: AAAI Conference on Artificial Intelligence, 2020, pp. 11856–11864.

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

[64] A. Hertz, R. Hanocka, R. Giryes, D. Cohen-Or, PointGMM: a neural GMM network for point clouds, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 12054–12063.

[65] B. S. Zapata-Impata, P. Gil, J. Pomares, F. Torres, Fast geometry-based computation of grasping points on three-dimensional point clouds, International Journal of Advanced Robotic Systems 16 (1) (2019).

[66] D. Morrison, P. Corke, J. Leitner, Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach, in: Robotics: Science and Systems (RSS), 2018.

[67] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[68] T. Hodaň, J. Matas, Š. Obdržálek, On evaluation of 6D object pose estimation, in: European Conference on Computer Vision (ECCV), 2016, pp. 606–619.

[69] C. Eppner, A. Mousavian, D. Fox, A billion ways to grasp: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set, arXiv preprint arXiv:1912.05604 (2019).

[70] F. Bottarel, G. Vezzani, U. Pattacini, L. Natale, GRASPA 1.0: GRASPA is a robot arm grasping performance benchmark, IEEE Robotics and Automation Letters 5 (2) (2020) 836–843.

[71] A. Yershova, S. Jain, S. M. Lavalle, J. C. Mitchell, Generating uniform incremental grids on SO(3) using the hopf fibration, The International Journal of Robotics Research 29 (7) (2010) 801–812.

[72] A. Bicchi, V. Kumar, Robotic grasping and contact: A review, in: IEEE International Conference on Robotics and Automation (ICRA), Vol. 1, 2000, pp. 348–353.

[73] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, A. M. Dollar, The YCB object and model set: Towards common benchmarks for manipulation research, in: international Conference on Advanced Robotics (ICAR), 2015, pp. 510–517.

[74] A. Depierre, E. Dellandréa, 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.

[75] H.-S. Fang, C. Wang, M. Gou, 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. 11444–11453.

[76] K. Collins, A. Palmer, K. Rathmill, The development of a european benchmark for the comparison of assembly robot programming systems, in: Robot Technology and Applications, Springer, 1985, pp. 187–199.

[77] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., ShapeNet: An information-rich 3D model repository, arXiv preprint

arXiv:1512.03012 (2015).

[78] C. Mitash, R. Shome, B. Wen, A. Boularias, K. Bekris, Task-driven perception and manipulation for constrained placement of unknown objects, IEEE Robotics and Automation Letters 5 (4) (2020) 5605–5612.

[79] K. Van Wyk, M. Culleton, J. Falco, K. Kelly, Comparative peg-in-hole testing of a force-based manipulation controlled robotic hand, IEEE Transactions on Robotics 34 (2) (2018) 542–549.

[80] J. Watson, A. Miller, N. Correll, Autonomous industrial assembly using force, torque, and RGB-D sensing, Advanced Robotics 34 (7-8) (2020) 546–559.

[81] Y. Bekiroglu, N. Marturi, M. A. Roa, K. J. M. Adjigble, T. Pardi, C. Grimm, R. Balasubramanian, K. Hang, R. Stolkin, Benchmarking protocol for grasp planning algorithms, IEEE Robotics and Automation Letters 5 (2) (2019) 315–322.

[82] J. Leitner, A. W. Tow, N. Sünderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, et al., The ACRV picking benchmark: A robotic shelf picking benchmark to foster reproducible research, in: IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 4705–4712.

[83] S. James, Z. Ma, D. R. Arrojo, A. J. Davison, RLBench: The robot learning benchmark & learning environment, IEEE Robotics and Automation Letters 5 (2) (2020) 3019–3026.

[84] C. Eppner, A. Mousavian, D. Fox, ACRONYM: A large-scale grasp dataset based on simulation, arXiv preprint arXiv:2011.09584 (2020).

[85] D. Morrison, P. Corke, J. Leitner, EGAD! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation, IEEE Robotics and Automation Letters (2020).

[86] J. Zhao, J. Liang, O. Kroemer, Towards precise robotic grasping by probabilistic post-grasp displacement estimation, arXiv preprint arXiv:1909.02129 (2019).

[87] D. Kappler, J. Bohg, S. Schaal, Leveraging big data for grasp planning, in: IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 4304–4311.

[88] M. Veres, M. Moussa, G. W. Taylor, An integrated simulator and dataset that combines grasping and vision for deep learning, arXiv preprint arXiv:1702.02103 (2017).

[89] M. Macklin, M. Müller, N. Chentanez, T.-Y. Kim, Unified particle physics for real-time applications, ACM Transactions on Graphics 33 (4) (2014) 1–12.

[90] M. Matl, Pyrender, `https://github.com/mmatl/pyrender` (2019).

[91] E. Coumans, Y. Bai, PyBullet, a python module for physics simulation for games, robotics and machine learning, `http://pybullet.org` (2016–2019).

[92] Blender Online Community, Blender - a 3D modelling and rendering package, `http://www.blender.org` (2016).

[93] R. Diankov, J. Kuffner, OpenRAVE: A planning architecture for autonomous robotics, Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34 79 (2008).

[94] E. Rohmer, S. P. Singh, M. Freese, V-REP: A versatile and scalable robot simulation framework, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 1321–1326.

[95] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, H. Lee, Learning 6-DOF grasping interaction via deep geometry-aware 3D representations, in: IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1–9.