Tampere University

Kulunu Samarawickrama

RGBD BASED DEEP LEARNING METHODS FOR ROBOTIC PERCEPTION AND GRASPING

Master of Science Thesis
Faculty of Information and Technology
Examiners: Prof. Roel Pieters
Prof. Esa Rahtu
May 2021

ABSTRACT

Kulunu Samarawickrama: RGBD BASED DEEP LEARNING METHODS FOR ROBOTIC PERCEPTION AND GRASPING
Master of Science Thesis
Tampere University
Master's Programme in Information Technology
May 2021

The recent advancements in robotic perception have vested robotic grasping with learning capabilities. During the past decade, empirical methods on grasp detection have been preempted by the data-driven methods highlighting the potential of deep learning and computer vision. Majority of the studies are focused on 2D information from RGB and depth cameras as the input. The ability to learn on 3D point clouds has been a breakthrough which overcome crucial challenges in 2D learning methods. Point cloud based learning endows robustness to occlusions, lack of texture and propagation of projection errors, in grasp detection algorithms. In modern state of the art, 3D model-based learning methods feature 6DoF pose estimation and instant semantic segmentation while other methods utilize deep generative methods. In this context, this thesis studies on synthetic dataset generation, implementation of algorithm and benchmarking grasp manipulation of point cloud based deep learning methods. Two grasp detection pipelines featuring each of the model-based and model-free approaches are implemented and comparatively analyzed in this thesis. The analysis provides an overview on 6DoF grasp pose detection on cluttered multi-object scenes and implicates on adaptation to real-world grasp manipulation tasks such as robotic assembly.

Keywords: Pose-Estimation,6DoF-grasp,Semantic segmentation, deep generation, Grasp Manipulation, Perception, RGBD, Point clouds, Synthetic data

PREFACE

It has been a privilege to experience the rich research culture at Tampere University. Many thanks to my mentors Professor Roel Pieters and Professor Esa Rahtu for the guidance and advises. Two years ago, I took a decision to give up work and move on to a different country to pursue higher studies in search of true motivation. At the completion of this thesis I have found my passion in research area of Robotics and AI. I express my immense gratitude to my parents for supporting me and backing me up to this point!

Tampere, 18th May 2021

Kulunu Samarawickrama

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

3D	3 Dimensional
ADD	Average Distance Metric
AUC	Area Under the Curve
CNN	Convolution Neural Network
DNN	DEEP Neural Network
DoF	Degrees of freedom
GPU	Graphics Processing Unit
MLP	Multi Layer Perceptrons
ROS	Robotic Operating System

1. INTRODUCTION

The effort to endow human-like capabilities to robots has resulted in a new branch in robotics called intelligent robotics. The ability to perceive and interact with the environment is a demanding quality in modern robot manipulation. Robotic grasping is a key element in robot manipulation with great utility. Though it is a trivial task for humans to grasp an object, achieving same dexterity in robotics has been a challenging task. The recent development of visual sensors such as depth cameras and machine learning algorithms has led to development of several novel grasp estimation methods. Though each method has become an integral part of the solution for grasp estimation, they address distinct problems and present diverse approaches. Therefore, a comparison study and establishment of a state of the art is a timely concern.

Sahbani et al. provides a wider perspective of the grasp estimation mechanisms by categorizing under analytical methods and Data-driven methods. Analytical methods use human expertise to analyze the geometric constrains of the object and manually formulate grasps. But these methods are tedious, resource consuming and inflexible. With the rise of machine learning, data driven methods have become the most prominent. These learning-based methods learn both appearance and geometric features and produce grasps robust to change in pose of the object.

Kleeberger et al. on their survey establish two main branches in learning-based grasping methods namely model-based grasping and model-free grasping.

1. Model-based robotic grasping follows pose estimation, grasp definition and path planning for known objects.
2. Model-free gasping produces grasps without pose estimation or prior knowledge on the object, thereby emphasizing the capacity to generalize to unseen objects.

Both model-based and model-free methods can incorporate point clouds for grasp estimation. These point cloud based learning methods are becoming a modern trend. RGBD sensors such as Microsoft Kinect and Intel Realsense generate detailed RGBD images which can be transformed to point clouds. The learning algorithms learn by sampling key-points from the point clouds. A model learning on 3D key-points have an edge over a model learning on 2D key-points with minimal error propagation.Therefore, RGBD based methods come as an extension to the RGB-based methods for more accurate grasp estimation.

The learning-based grasp estimation methods heavily rely on data. Specifically, model-based learning methods require annotated image or video data-sets. Various data generation techniques and ready made image datasets are available for this purpose. But a pipeline for customized automatic data generation is essential to extend these grasp manipulation techniques to applications in various industries.

Furthermore, Grasp Estimation is only a single aspect in robotic grasping. The grasps generated on a single object could be further constrained in a multi-object scenario. Grasp execution has to consider the constraints imposed by the robot kinematics , working environment and object scenario. These constraints are inherently tested in grasp execution for each manipulation task.This leads to a necessity in benchmarking protocols for grasping activities such as pick and place, bolt-screwing,peg and hole etc.

This thesis outlines achievement of the following objectives in line with the aforementioned problem statement

- To survey and compare modern grasp estimation methods
- To analyze class agnostic and multi-class learning-based grasp estimation methods
- To formulate a synthetic customized data generation pipeline for industrial mechanical components
- To train, evaluate and compare 6DoF grasp estimation on both model-based and model-free learning approaches on generated dataset
- To simulate the grasp manipulation using Franka Emika Panda robotic manipulator and perform benchmark tests

The thesis is structured as follows :

1. Chapter 2 is an overview of the literature, definitions and state of the art grasp detection methods
2. Chapter 3 presents the methodology of implementation of two novel grasp detection methods on a simulation environment
3. Chapter 4 presents an analysis of the results of implemented grasp detection methods
4. Chapter 5 is a discussion and comparison of the two methods which were implemented for grasp detection
5. Chapter 6 provides a summary and conclusion on grasp detection approaches with suggestions for further improvement.

2. BACKGROUND

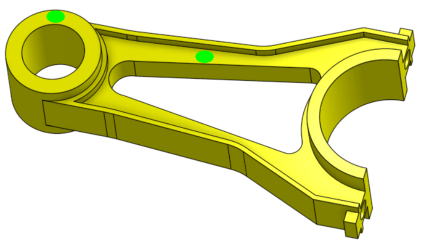
2.1 Grasp Representation

Grasp representation is the visual interpretation of parameters required by the robot to position its end effector to perform a successful grasp. A successful grasp is a stable and collision free lift of an object with an optimal grip . The grasp should maintain the force equilibrium on object and should be tolerant to minor disturbances. A grasp should also be pertaining to robot's kinematic constraints and task related queries.[1] [3] [4].

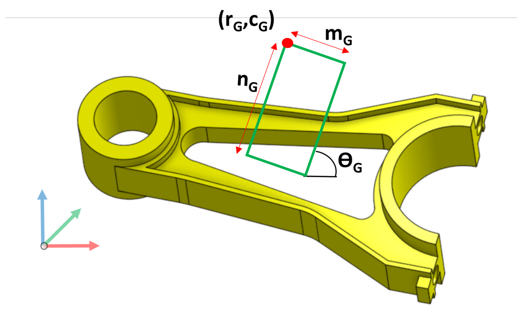
The pose of an end effector is defined using 3D position and 3D orientation. Sahbani et al. introduces 2 main branches of grasp forms as 2D planar grasps and 6doF grasps. 2D planar grasps are restricted to a single plane while 6DoF grasps represent the 6D orientation of the gripper in camera coordinate frame. The representation of grasp pose is different for each of them. For example, Caldera et al. provides an analysis and comparison on the planar grasp representation methods. Which are namely point representation, pose with orientation representation and rectangular box representation.

Point representations use points (x,y) or (x,y,z) on a 2D image to represent grasping points. These representations are simple but they do not provide 3D orientation, grasp width and height information for effective end effector planning. This was partially overcome when combined representation using both location and orientation (x,y,z, theta) were introduced later [5] [6] [7].

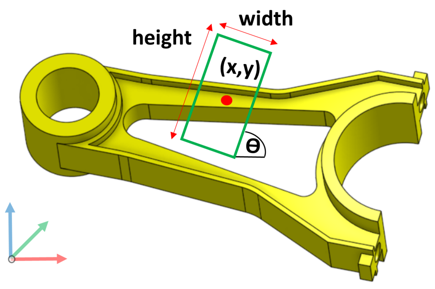
Rectangular box representations which feature grasp point, orientation and gripper width are a more popular and informative presentation. These methods were well complimented by the improvement of object detection methods in computer vision. Some examples are 5D representations (x, y, theta, width, height), and 7D representations (x, y, z, roll , pitch, yaw, width) where grasp point was the top vertex of the rectangle with respect to robot workspace [8] . Later, Redmon and Angelova introduced a simplified 5D grasp representation by altering the top-vertex defined grasp rectangle to a center defined grasp rectangle. These grasp representations are illustrated in figure below.



***(a)** Point representation*



***(b)** Rectangle box representation with Top vertex (r_G, c_G), length m_G, width n_G and its angle from the x-axis, θ_G*



***(c)** Simplified representation with centre at (x, y) oriented by an angle of θ from its horizontal axis*

***Figure 2.1.** Planar-grasp representation methods*

Modern grasp manipulation setups consist of a RGBD camera which provides the visual perception to the robot [3].The depth cameras has the ability to generate full or partial point clouds of objects which are used by the algorithms to generate grasps on . Often these setups work with known objects and grasp detection is sub divided in to tasks of object detection, pose estimation and grasp estimation. Hence, 6DoF grasp representation is the 3D translation and 3D rotation of the gripper pose with reference to the goal pose defined in camera coordinate system of the setup. [10] [11]

This representation is not constrained to a single plane. The figure below illustrates the 6DoF grasp representation on a point cloud of an observed object. It is important to note that the object world coordinate frame is parallel to the camera world coordinate frame.

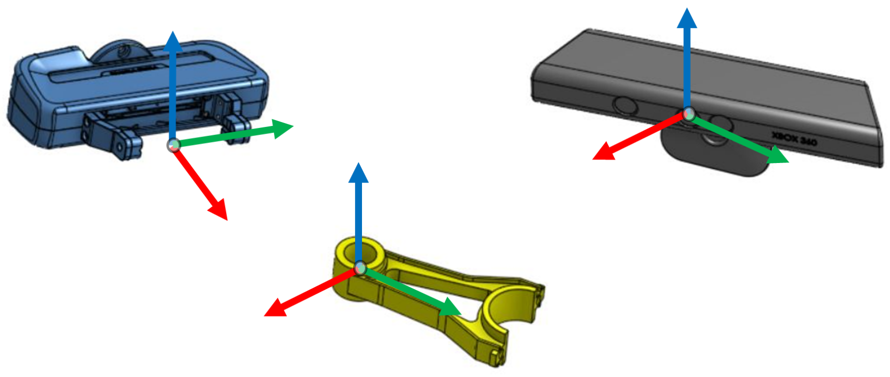


Figure 2.2. *6DoF Grasp representation*

2.2 Grasp Detection Methods

2.2.1 Analytical approaches

Analytical methods were the most prominent approach in the early stages. The core idea of analytical methods were based on geometric, kinematic, or dynamic formulations. With heavy human intervention, mathematical modelling and hard-coding these methods were lacking the sense of autonomous behaviour. Shimoga [12] describes this approach as the process of generating force-closure grasps which characterize dexterity, stability, equilibrium and compliance to a certain dynamic behaviour.

Force-closure grasps are a sub category of equilibrium grasps which are discussed in depth by Howard and Kumar [13]. Force-closure is an equilibrium grasp which can counteract against the disturbance forces using a linear combination of forces exerted on a solid object by the gripper. In the process of force-closure grasping for a 3D object, surface normals on flat faces of an object are detected. Then the grasps are tested for satisfying force-closure conditions by testing among different combinations of surface normals [1] [3].

However, there are more downsides of analytical approaches. Bicchi and Kumar [14] in their work highlighted that these approaches were not robust to positioning errors. The analytical methods often resulted in uncertainties for end-effector positioning and the urge for optimizing the grasps stood up [15] [16] [17]. Kirkpatrick et al. introduced a grasp wrench space for grasp optimization. The task of optimization is to find contact points and suitable approach vectors which maximize the resistance to grasp wrenches applied on objects. This quantifies the grasps according to efficiency by indicating as the radius of the largest sphere which can be constrained in a convex hull containing all grasp wrenches, hence known as convex optimization.

The analytical methods involve heavy computations and search algorithms which are not favourable in practical deployments. They featured methods which use heuristics to avoid the tedious case-by-case calculations in filtering the grasp candidates. But the techniques were not flexible and task specific. The analytical approaches were further constrained by their limitations to a specific environment. The modelling of actuator noise, sensor noise is another aspect which should be modeled case-by-case. Furthermore, these techniques were heavily reliant on the precision of geometric and physical models of objects. But the precision among different objects of the same class is not often uniform in reality. The variations in parameters such as friction, weight, center of mass were directly affecting the quality of grasps produced. Thereby, analytical approaches were tend to become more complex and time consuming. Further, in order to accommodate different grippers, object and workspaces, a complete re-modelling of the grasping algorithm was required.

2.2.2 Data-driven approaches

In contrast to analytical methods which formulate force-closure grasps to satisfy certain conditions as stated in previous section, data-driven methods exploit empirical inputs such as human demonstrations, perceptual information and semantics. For this reason, they are also known as empirical methods. These methods however do not guarantee the same grasping attributes such as dexterity, equilibrium, stability and dynamic performance as analytic approaches. These attributes have to be measured and verified empirically.

The data-driven approaches slowly transformed as an extension from analytic methods during the first decade of 2000s. In 2004, when Graspit! [19] was introduced it took a different path in grasp detection compared to the existing analytical methods. Graspit was a grasping simulator which analyzed, planned and evaluated grasps. The simulator was used in highly cited work which used different grasping techniques [20] [21] [22]. These methods sampled grasp candidates and used different metrics to rank the grasps. Although the grasp sampling followed data-driven approaches grasp ranking approaches were predominantly analytical. The grasps were ranked using classical metrics such as $\epsilon - metric$ featured in the work by Ferrari and Canny [23]. The simulation based grasp detection provided more control over the environment and planning for researchers. But, Weisz and Allen pointed out the huge gaps in these simulation based work in the deployment on a real robot.

In 2008, Saxena et al. featured supervised learning for their work in computer vision based grasp detection with synthetic data. The Machine learning techniques continued to become more popular with the development of computing technology and software packages. These methods focused more on object representation and perceptual processing. The objectives included feature extraction, object detection, classification and segmentation and pose estimation [26]. In the following years deep learning and probabilistic machine learning redefined many fields of science.This sudden surge in artificial intelligence methods completely took over the data-driven approaches in grasp detecting. The diversity in techniques created many branches in data-driven approaches for grasp detection. Categorizing the techniques according to similarity is essential task for understanding the state of the art.

Sahbani et al. made an early attempt to categorize the existing approaches based on the learning approach;

- **Human based learning** which is the process of learning a human demonstration and replicating the grasping task. Magnetic trackers and Datagloves and vision based human hand gesture tracking and learning are examples in this category. Their learning is based on human hand joint angles, trajectory and shape.
- **Object based learning** is the second type of learning which studies the object features and deploy supervised learning to detect grasp points or grasp regions.

But this categorization was not effective in capturing many different kinds of approaches that followed. Bohg et al. presented a different overview on categorizing different techniques based on the types of objects grasped. The main factor to consider is whether there is existence of previous knowledge *(a priori)* about the object.

- **Known Objects** In this case the object has been observed and grasps are predefined. In the task of grasp manipulation, robot extracts the suitable grasp depending on the object type and transforms the grasp to the estimated pose of the goal object in world coordinate frame.
- **Similar Objects** refers to objects that belong to a previously seen object category. There has to exist a measure of degree of similarity to previously seen object. Therefore, a normalized object representation per category of object is used as the similarity measure. Then the grasps from previously seen instances are transferred to the familiar object.
- **Novel Objects** This instance refers to objects never seen before. This method focus on identifying structure and extracting global or local features of object using sensors and generating and ranking grasp candidates.

However in the modern era the problem of grasp manipulation is observed as a machine learning task. The topic machine learning is wide and numerous learning methods have been published over past few years. Deep learning, reinforcement learning and Probabilistic Machine learning are key aspects in current state of the art. The fundamentals of this approach is to learn on a model developed using certain knowledge acquired about the object. There are methods which do not learn on such models as well. Therefore current learning based tasks can be broadly sub-categorized under two main categories of model-based learning and model-free learning [2]. This is a sensible approach to categorize the existing data-driven approaches given that current research in subject area is being published under topics of machine learning principles in journals and conferences.

The learning based approaches require large datasets and control over external factors affecting data generation. The simulation based synthetic data generation has well complemented this task. The grasp detection pipeline from data acquisition to grasp execution is emphasized in the following figure.

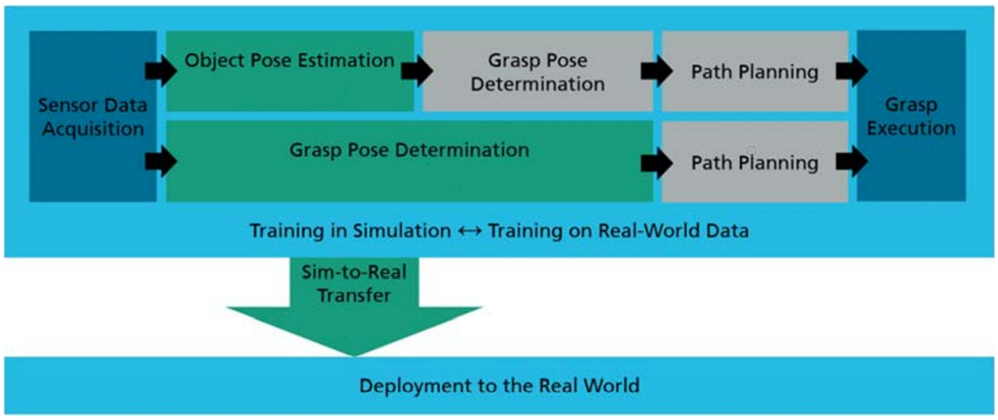


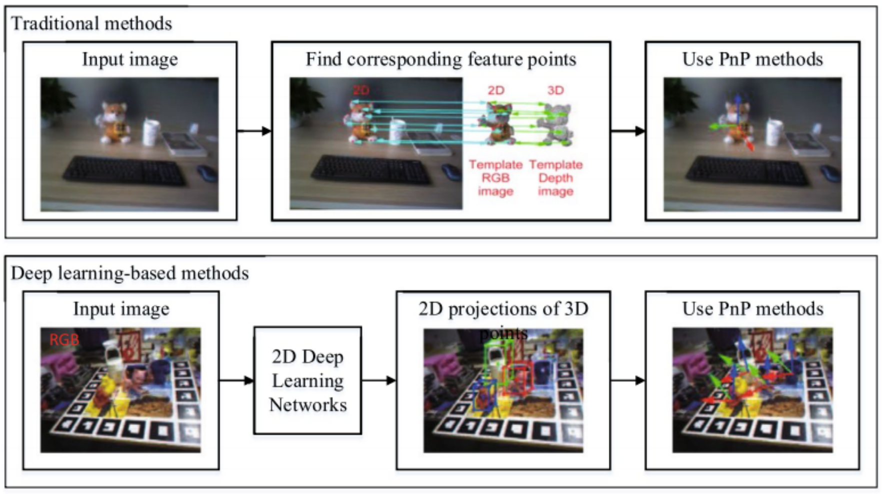
Figure 2.3. *Typical pipelines to robotic grasping - image from [2]*

2.3 Model-based Grasp Estimation

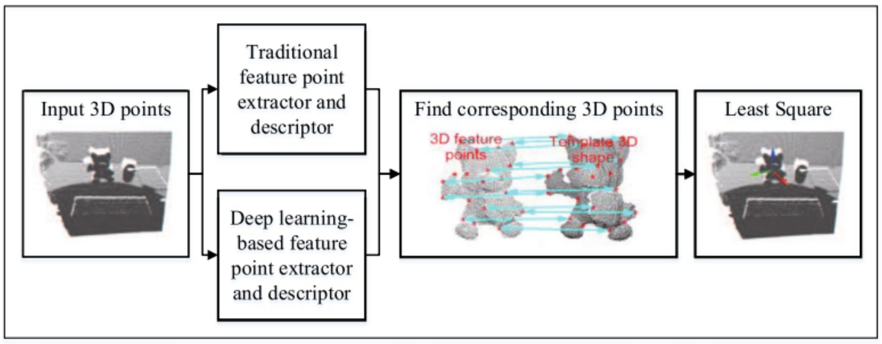
Model-based techniques are applicable where accurate CAD models of the target objects are available for learning. These methods can be extended to similar objects as well. Generally, given a model of an object the grasp detection is achieved via pose estimation. The fact that objects are known or familiar means the grasp handles on the object can be predefined on object coordinate frame. Then pose estimation allows to transform the grasps on to the camera coordinate frame in order to generate the grasps on target object. There exists several notable work which are robust to occlusions and varying lighting conditions and on some occasions scale invariant [27] [28] [29]. The methods can be sub-categorized according to the techniques as below

2.3.1 Correspondence based methods

These methods recover the pose of an existing 3D model from 2D images of the model from different view angles. The correspondences are established between the 2D image pixels and 3D points of the image. The extracted features are known as descriptors. And the pose is recovered using perspective-n-point algorithms [30]. There exists several classical correspondence-based methods such as SIFT [31], SURF [32] and ORB [33].



***(a)** 2D-3D correspondence methods*



Input 3D points
Traditional feature point extractor and descriptor
Deep learning-based feature point extractor and descriptor
Find corresponding 3D points
3D feature points
Template 3D shape
Least Square

***(b)** 3D-3D correspondence methods*

***Figure 2.4.** An overview of correspondence based methods - figure from [3]*

With recent popularity in depth cameras, these methods have extended to 3D-3D correspondence matching using point clouds. Correspondences can be derived from partial-views of point clouds and the 3D model. And unlike in 2D-3D correspondence methods, 3D-3D correspondence methods estimate the pose using least square algorithm. Some of the well known 3D Descriptors are FPFH [34] and Shot [35].

The above classical methods tend to be less effective when the objects lack rich texture. This is further challenging when objects are in clutter and exposing to light variations. Pham et al. introduced a method by mapping 2D and 3D input features to a shared latent space. It has showed that this discriminative approach in cross domain descriptors are more efficient than 2D-3d traditional descriptors.

With development of deep learning a new approach towards extracting representative feature with CNN were introduced. Yi et al. work suggests extracting disciminative feature points and matching them using representative CNN features. Hu et al. introduced a method which uses segmentation to extract descriptors. In achieving this, the segmented features were filtered and descriptors were ranked with a confidence value.This method proved to be robust to occlusions and lack of texture in the objects.

There are correspondence methods that build a normalized representation for familiar objects. In this type of work, template is build from multiple images. NOCS is an example where they transform a 3D model in a normalized coordinate plane with a diagonal of unit bounding-box located at the center. Then a Mask-RCNN [27]is used to learn the correspondence between the RGBD images and similar model.

2.3.2 Template based methods

Template based methods retrieve the most similar template from a database of templates which have ground truth pose information prerecorded. The templates in this context are the 2D images which are projected from the 3D models of the objects. Therefore, in these methods the task is rather interpreted as an image retrieval problem. Template-based methods are more suited for texture-less objects while it proves less effective to infer pose using correspondences.

Some template-based methods use the 3D model information directly without projecting to a 2D image. In this case, the template is the full point-cloud of the target model. An observed partial point cloud is matched against the database of existing point clouds and the one which aligns the best is chosen to derive the pose. These methods are further explained in the next section on point-cloud based methods.

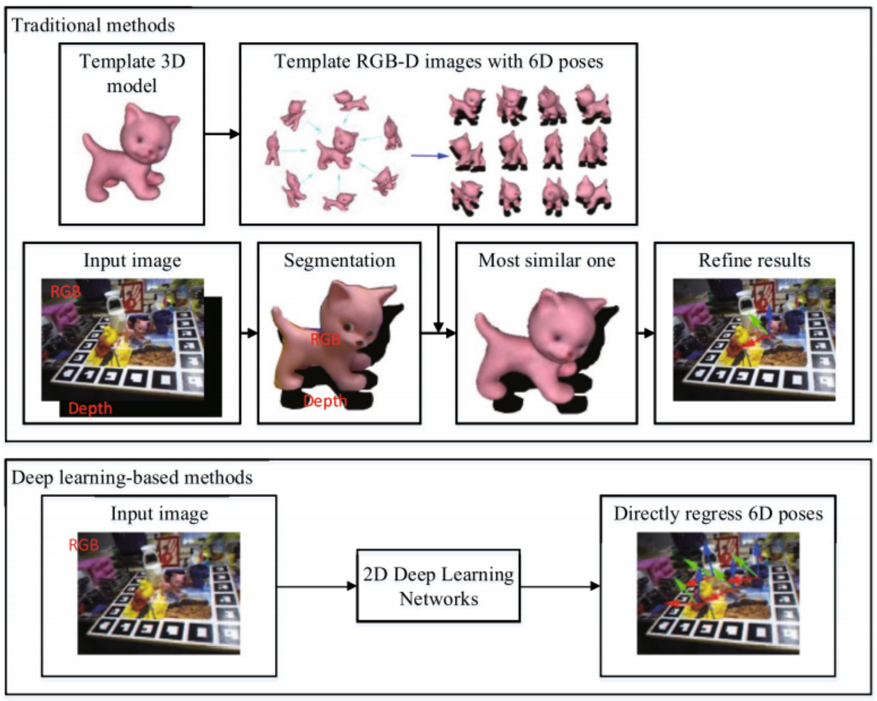


Figure 2.5. *An overview of Template-based Pose Estimation- Image from - [3]*

There has been a lot research in this area to improve the strategy. One challenge of this method is to obtain large number of templates. Hinterstoisser et al. came up with a method by using image gradients sparsely scattered on 2D image. This procedure worked with a limited number of templates. The pose estimation for multiple objects scenarios were another important aspect. Hodaň et al. proposed a method which uses object detection and localization which was successful in texture-less multi-objects.

Like every other method, the classical approaches were improved by deep learning approaches. These methods assisted in extracting effective templates without manual coordination.Gordo et al. work features an image retrieval step based on deep learning-based methods. However this work did not prove to produce enough number of templates to learn representative and discriminative features.

PoseCNN [42] is a notable work which perfoms direct pose estimation on objects. This method calculates the 3D translation of an object projected on a 2D image. It is done by localizing the center of the object and predicting the distance to the center from camera. 3D rotation is estimated by regressing a quaternion representation.ConvPoseCNN [43] proposed RoI by pooling the extracted features in fully-convolutional network structure. In this process the regions with interested regions are extracted. This method simplifies the prediction of rotation and translation by unifying them into one regression problem. This was less complex and time efficient in inference compared to PoseCNN.

HybridPose [44] is another method which utilize hybrid intermediate representations such as edges, key-points, pixel-wise symmetry correspondences keypoints. Uses hybrid intermediate representations such as key-points, edge-vectors, dense pixel-wise symmetry correspondences. This method is effective when one type of template feature fails to produce accurate results.

There are correspondence methods that propose a latent representation for familiar objects. Park et al. in their work LatentFusion, describes network that back project a latent 3D representation of an object by using only small number of reference views during the inference. A natural renderer is used to reconstruct the object using arbitrary views. This avoids the large number of templates required to predict pose like other methods and also provides the opportunity to predict estimation on unseen and familiar objects. These methods were competitive against other supervised learning methods.

Latentfusion does not specifically require information about the model to predict the pose. But in the process of grasp estimation the knowledge about the object is necessary to produce grasp handles. Therefore this approach is considered under model-based grasp estimation approaches which could utilize template-based techniques for pose estimation.

2.3.3 Voting based methods

In voting based methods each pixel or 3D point makes a contribution towards object pose estimation by voting. There are two main types of voting. Indirect voting contributes towards pose estimation via 2D-3D correspondences or 3D-3D correspondences. Direct voting assumes directly voting to certain pose ground truth rather than correspondences.

In indirect voting methods seed points are used to vote key-points which are utilized in creating 2D–3D correspondences or 3D-3D correspondences. If the input is a 2D image, 2D key-points are voted and 2D-3D correspondences are achieved. If the input is a 3D image 3D key-points are voted to find correspondences between the observed partial point cloud and the full point cloud of the target object. This procedure is illustrated in the figure 2.6

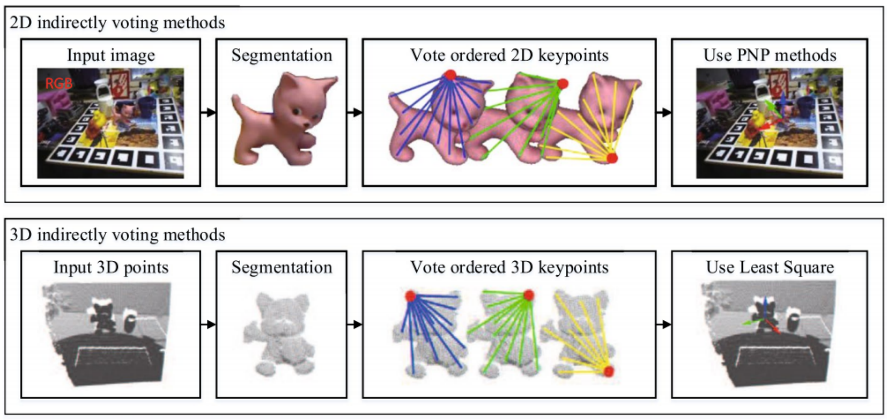

2D indirectly voting methods
Input image
RGB
Segmentation
Vote ordered 2D keypoints
Use PNP methods
3D indirectly voting methods
Input 3D points
Segmentation
Vote ordered 3D keypoints
Use Least Square

Figure 2.6. *An overview of indirect voting for pose Estimation- Image from - [3]*

PVNet [46] is a network which follows indirect voting from 2D RGB images. This work is known for its robustness to occlusions. This is because the work adopts patch-centers as the key-points which are being voted. Further they calculate an uncertainty level for each vote which is an effective application to localize occluded or truncated keypoints. This method was extended to 3D domain by Y. He et al. in PVN3D ,where 3D seed points are used to vote 3D keypoints. These 3D-3D correspondence voting methods will be further explained in section 2.5

Direct voting methods can be interpreted as a template-based voting system if the voting scheme is accustomed to vote for the most similar template. An overview of this method is illustrated in the figure 2.7 Brachmann et al. proposed 6D pose estimation method which uses a single RGBD image and a learned, intermediate representation combining 3D object coordinate labels and dense class labels. More examples of this approach include work presented by C. Wang et al. , Wada et al.

2.4 Model-free Grasp Estimation

This type of grasp estimation is capable of producing grasps for novel objects. Unlike in model-based grasp estimation these methods do not require any object specific knowledge. Therefore, these methods do not include pose estimation step. Hence grasp ma-

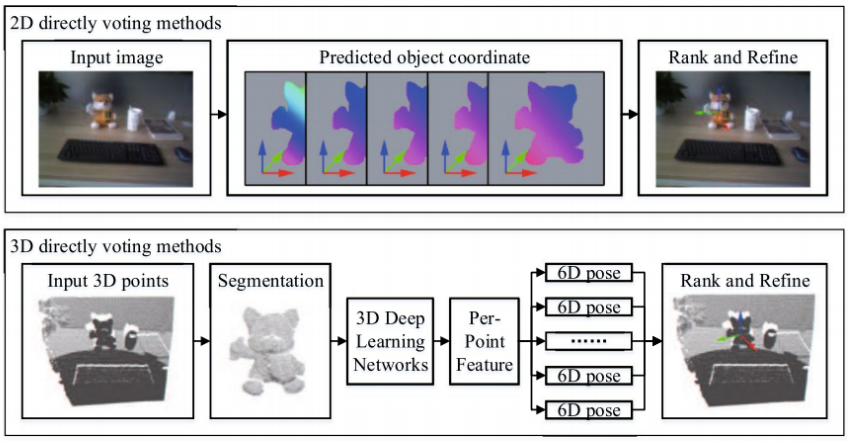


***Figure 2.7.** An overview of indirect voting for pose Estimation- Image from - [3]*

nipulation tasks using these methods do not complement well in object placement or assembly.

Model-free approaches takes geometric properties as the input and directly estimate grasps on the object. Generally, these networks are trained in an end-to-end fashion and optimized to produce robust grasps. The evaluation of these networks are often done using classical metrics which were explained in 2.2.1. Typical pipeline follows grasp sampling and ranking processes. Model-free approaches can be further categorized as Discriminative approaches and Generative approaches [2]

2.4.1 Discriminative approaches

Discriminative approaches exhaustively sample grasp candidates and separately run a procedure to evaluate and rank them using a neural network. The highest ranked grasp is then is selected as the grasp pose output. These methods inherit high inference time due to the forward pass in their network structure.

Levine et al. proposed a learning based approach to assist hand-eye coordination for robot grasping with a single RGB input. The task of grasp sampling was carried with 14 robots to sample over 800,000 grasps for a span of 2 months. But it is stated that a change in robot environment setup requires retraining of the whole network. Nevertheless, these methods have a benefit, which is the ability to evaluate multiple grasps. The evaluation is done using CEM [52]. Sundermeyer et al. in their work Graspnet, introduces gradient-based refinement to save grasps which are rejected but still feasible in their grasp evaluation step.

Dex-Net 1.0 [53] and Dex-Net 2.0 [54] uses a physics simulation to capture randomized grasp poses. They created a large dataset with grasp pose output and aligned crop of depth image. This dataset also incorporated grasp failure, gripper and object position uncertainties, collision avoidance, approach angle limitations, gripper-roll limitation and camera noises to the dataset to predict robust grasps.

2.4.2 Generative approaches

A grasp configuration is directly output in generative approaches. Therefore this type of approach provides a grasp detection rather than a grasp estimation. Generally, the grasps are represented as oriented rectangles which was explained in the figure 2.1. Therefore this approach can be further simplified to a task of object detection [55] [56] with the additional parameter of gripper orientation. Redmon and Angelova proposed a method called SingleGrasp which directly predict grasp rectangle and carry classification simultaneously on a RGB input. This work followed by their MultiGrasp work as there exists multiple ways to grasp an object in 3D space.

2.5 Grasp Estimation with Point Clouds

An RGB-D image can be interpreted as a RGB image combined with its corresponding depth image. A point cloud is the result of 3D projection of a RGB-D image [3]. This is illustrated in the figure 2.8. The introduction of RGB-D sensors such as Microsoft Kinect and Intel RealSense enables the easy access to point clouds of objects when camera insintrics. Similar to 2D methods point cloud based methods can also be classified as model-based or model-free approaches. In the case of model-based approach, grasp estimation is analogous to 6DoF pose estimation. And model-free approach, grasps are directly detected on visible full or partial point cloud. The grasps detected on point clouds are different from the 2D approaches as they detect 6DoF grasps as illustrated in figure 2.2.

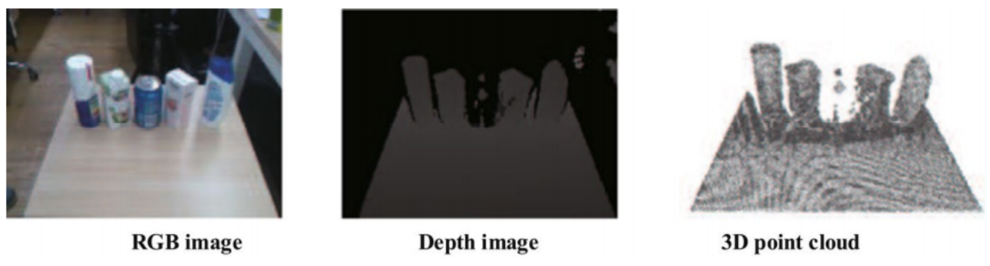


Figure 2.8. *RGB image, RGB-D image and Point cloud of the same scene - Image from - [3]*

Generally, most of the point cloud based grasp estimation work is based on deep learning. There has been a number of research in grasp estimation which exploit RGB-D images. But they are categorized as 2D methods which incorporate depth as an additional parameter. The most popular 3D space pose estimation methods based on 2D key-points using RGB-D images include Pointnet [58], Frustrum pointenets [59] and Votenet [60]. In contrast, point cloud based methods become much beneficial in 6DoF grasp detection [47]. Due to more informative representation of the object geometry, point clouds are more effective in extending grasp detection to novel or unseen objects. The existing 2D key-point based 6DoF pose estimation tend to produce errors in 3D projection which may lead to large errors in 3D space. This can adversely affect the accuracy of estimated pose and there by affecting the quality of the grasps detected.

Point cloud based methods are able to learn 3D point-wise features. Therefore these methods are able to handle texture-less images much better compared to classical methods discussed in previous sections. Point cloud based methods can over come the problem of key-point overlapping in 2D image based techniques without loosing important features for learning.The point cloud based methods are facing challenges when it comes to model-based approaches in grasp detection due to scarcity of available datasets. The recent work introduced by Fang et al. introduced a dataset with 87,040 RGBD images with more than 370 million grasp poses. Another challenge faced by point cloud based methods is their high dimensional nature which is consuming lot of computation power.

2.5.1 Feature extraction from point clouds for Deep learning

The first step in model-based approaches is to localize and classify the point cloud assuming the target object. Therefore object detection and object instance semantic segmentation modules are included in the network pipelines for pose estimation [47]. This is favourable in cases where multiple objects are in clutter. In model-free approaches the object detection is sufficient to detect grasps.

Guo et al. presents several 3D shape classification methods for feature extraction from point clouds which are featured in several object detection and segmentation work. In these methods, all embedded points in the point cloud are absorbed and a global shape is extracted as the feature. These features are then utilized as input to the neural networks which perform different tasks.

Multi-view representation is a technique where multiple views are obtained around the point cloud on their surface normals and view-wise features are extracted. Volumetric representation is a Voxelized representation which extracts features in a 3D grid. There are two other representation known as spherical and Lattice representation. An overview of the 3D representations are illustrated in the figure 2.9

Point based deep learning methods are becoming more popular in modern point cloud based grasp detection research. In these methods the features extracted in differnt ways. Guo et al. and Shi et al. in their work establish Point-wise multilayer perceptrons, graph based methods and convolutional methods as the main variations of methods used for

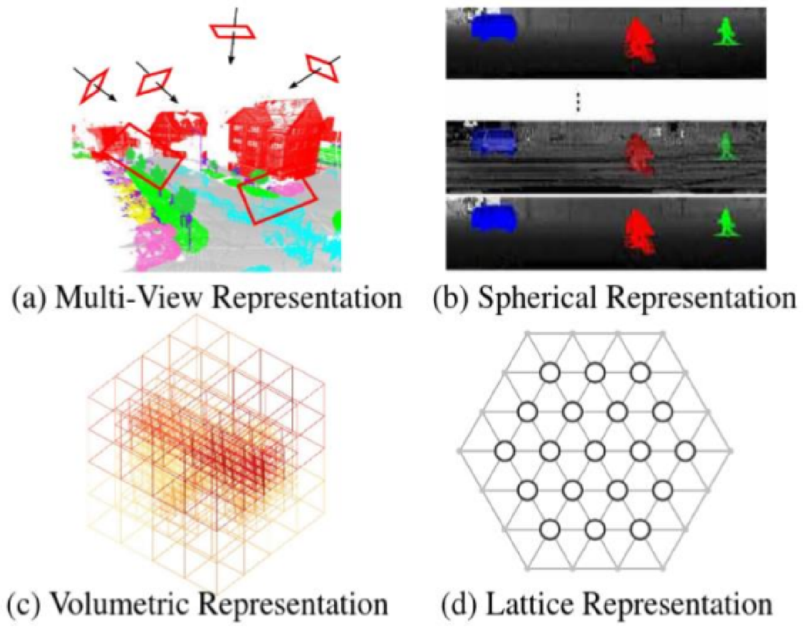

(a) Multi-View Representation
(b) Spherical Representation
(c) Volumetric Representation
(d) Lattice Representation

Figure 2.9. *Representation methods for 3D Shape classification in point clouds - Image from - [62]*

feature extraction.

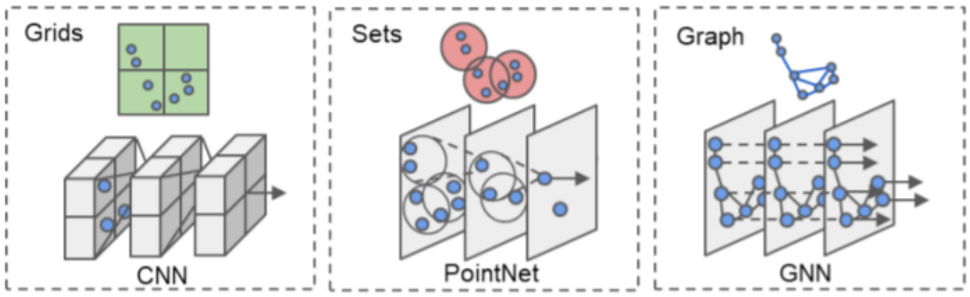


Figure 2.10. *Feature extraction methods - Image from - [63]*

Convolution in both RGB and RGB-D images can be defined with 2D and 3D kernels. However this is not directly applicable to point clouds as the points are randomly located. This can be overcome by adopting continuous convolutional kernels instead. Another method is to apply clustering to define kernels. Discrete convolution is the third option where you define kernels as usual grids while assigning weights for neighbouring points with respect to the center point. convolution based methods are negatively impacted by the sparsity and irregulars of points.

Graph neural networks uses graph theory in defining the point cloud. All neighbouring points in the cloud are connected via edges. Each node is assign either a coordinate, color or intensity are listed as a feature. The edges are featuring geometric attributes such as distance etc. For learning, these methods adopt MLPs with usual operations such as pooling in spatial domain. Shi et al. and Y. Wang et al. are two of the most recent work based on graph convolution. These methods have an edge over other methods due to their non repetitive , one-time feature extraction and the ability to adapt to the pattern of the point cloud. The graph based feature extraction procedure is explained in the figure 2.11

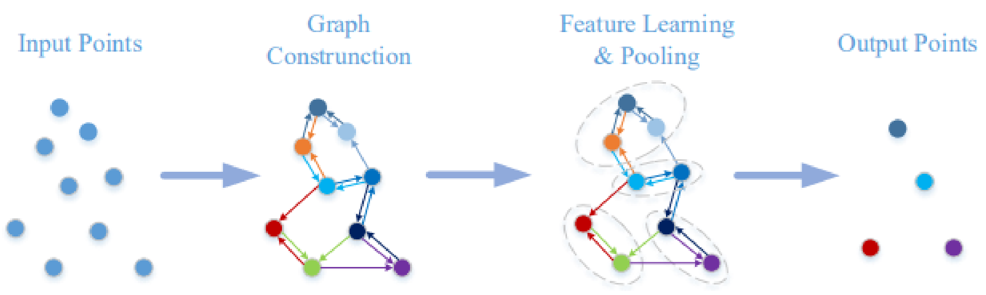


Figure 2.11. *Graph-based Key point extraction procedure - Image from - [63]*

Point-wise methods model each individual point with several shared MLPs. PointNet [65] and Pointnet++ [66] are two notable work that directly absorb a point clouds and achieve permutation in-variance by formulating as a symmetric function. In the case of PointNet point-wise features are learnt with several MLP layers and global features are extracted with a max-pooling layer.

PointNet fails to capture local features (point-wise eucledean distance etc) between points since it learns each point individually. This is a loss of important information in feature extraction step. PointNet++ overcomes this shortcoming by introducing a hierarchical network which learn the geometric structure of neighbouring points. PointNet++ abstracts point sets using 3 layers. The layers are namely sampling layer, grouping layer and PointNet learning layer. The abstraction threshold is varied as it learns local features layer by layer. The set abstraction threshold is defined by a neighbourhood ball in euclidean space. Therefore the final output feature is a combination of both local and global features. Furthermore, a mini feature-alignment network that facilitate features which are transformation invariant was introduced. The figure 2.12 provides an illustration of the network architecture of PointNet. These methods apply in mostly in 3D point cloud base object detection, pose estimation. There are number of methods in recent past which adopts the techniques introduced by PointNet and PointNet++ [67] [68]. These methods are also feature in backbone networks PVN3D [47]. The only unfavourable characteristic of these methods is that they require repetitive sampling and grouping to be done hence involving heavy computations.

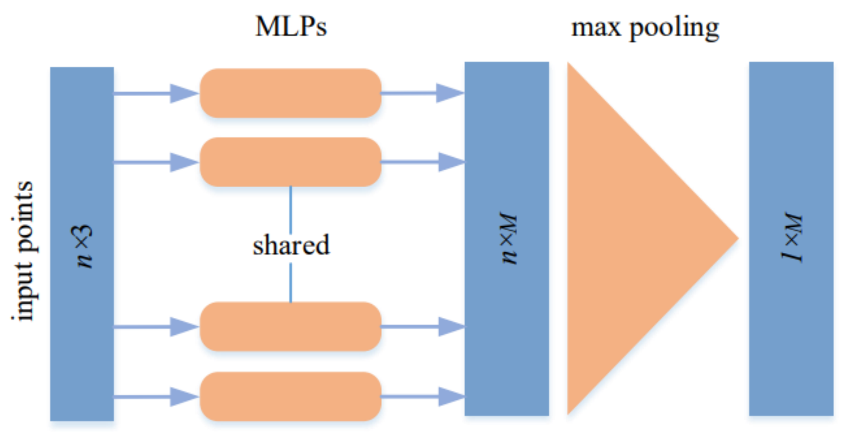


Figure 2.12. *Lightweight PointNet Network Architecture - Image from - [62]*

2.5.2 Point cloud based pose estimation

The rich presentation of a 3D scene using point clouds led to more effective pose estimation methods which are based on deep learning. The 2D to 3D projection which accumulate errors in 3D space could be avoided using point cloud based pose estimation methods. Technically, point cloud based methods eliminated all the post processing steps from RGB and RGB-D image based methods. Point cloud based methods follow the same classification as RGB/RGB-D model-based approaches

i). Correspondence-based methods:

To estimate 6DoF pose using correspondence based methods the observed partial-view of the point cloud is aligned with the ground truth point cloud in order to find the corresponding 3D points. Th 3D local shape decriptors such as SHOT [35], FPFH [34] are utilized to find the correspondences. Once the correspondences are known pose can be estimated via least square regression etc. 3DMatch [69] is one example which learns on volumetric patch descriptor to detect correspondences between observed partial views of an object in 3D space. LCD [36] is another method which combined both 2D and 3D methods by mapping features into a shared latent-space using a dual auto-encoder.

ii) Template-based methods

Template based methods are oriented at matching the partial -point cloud that best aligns with a prefigured template point cloud. Global registration methods Mellado et al. and Yang et al. which are utilized in these methods are robust to large variations of initial poses and noise. The time efficiency in these methods is considerably low. The deep learning-based methods approach the partial registration task by consuming a pair of point clouds respectively to extract discriminative and representative features using 3D deep learning networks, and regress the relative 6D transformations [72]

iii) Voting-based Methods

This is fairly a new research area with very few publications. Y. He et al. in their work PVN3D introduce a point-wise 3D key-point voting network for 6DoF pose estimation as a novel approach. This is an 3D extension of 2D pixel-wise hough-voting scheme . It learns the 3D offsets from predefined set of 3D keypoints point-wise. Their network comprises of a key-point detection module and semantic instance segmentation module.

2.5.3 Point cloud based grasp detection

An alternative method compared to section 2.5.2 for grasp detection is directly regressing the grasps on object rather than estimating the pose of the object. This is a favorable approach for grasp detection on unseen objects. Pas et al. introduced a method which adapts anti-podal sampling of grasp poses on object point cloud. Anti-podal sampling are based on the constraints of force-closure. The method is a two staged approach where grasps are randomly sampled on the point cloud at the first step and refined using the constraints in the second step.

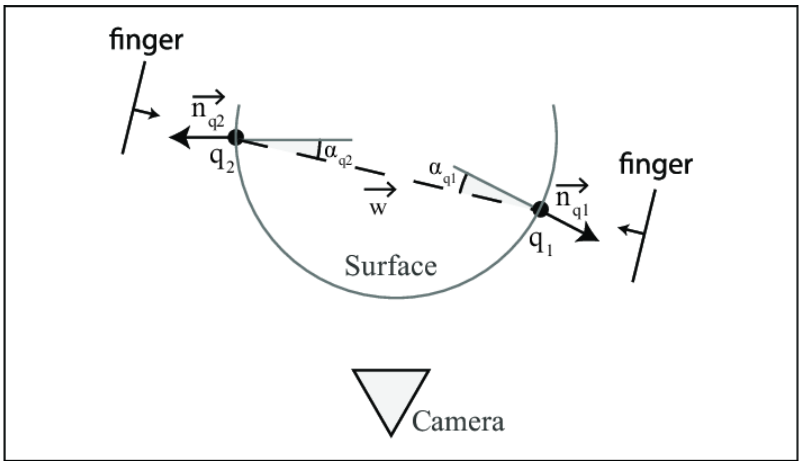


Figure 2.13. *Antipodal configuration presented by Zapata-Impata et al. [74]*

Zapata-Impata et al. Introduced another method which calculate a dual of feasible grasp points on a partially observed point cloud. The points are calculated based on a predefined volume intersecting the origin and principle axis of the object. The ranking of grasps generated by this method are based on the distance from the origin to the plane including the principle axis and the curvature of points around the neighborhood. The ranking also considers antipodal criteria and the angles between the line adjoining two grasp points and the cutting plane.

PointentGPD [11] implemented a network utilizing the pointnet architecture which learns directly on the geometry of the 3D points. The work introduces a continuous grasp quality metric using the coefficient of friction and grasp wrench radius on the point cloud. This metric is utilized in producing ground truth labels in the YCB dataset.

Sundermeyer et al. introduced Graspnet which is featuring deep generative methods for graspnet. The core idea of this network is to train an encoder which learns to enxtract useful features from the point cloud to a latent space and a decoder which learns to reconstruct the grasps from the latent space. This method is different from the implementation of the methods discussed previously [74] [73] as it follows a two stage approach where the grasps are sample and evaluated in first step and grasps are refined in the second step. The advantage of this method over other methods is its capability to convert near failure grasps to successful grasps. This results in better grasp coverage in graspnet compared to other methods.

3. METHODOLOGY

As presented in the previous section, modern robotic grasp manipulation is based on two main approaches as model-based and model-free. Among the plethora of research on grasp detection, this thesis work performs a comparative research on 3D point cloud based grasp detection. The methodology adapts two state of the art work from each of the two main branches for grasp detection in the literature.

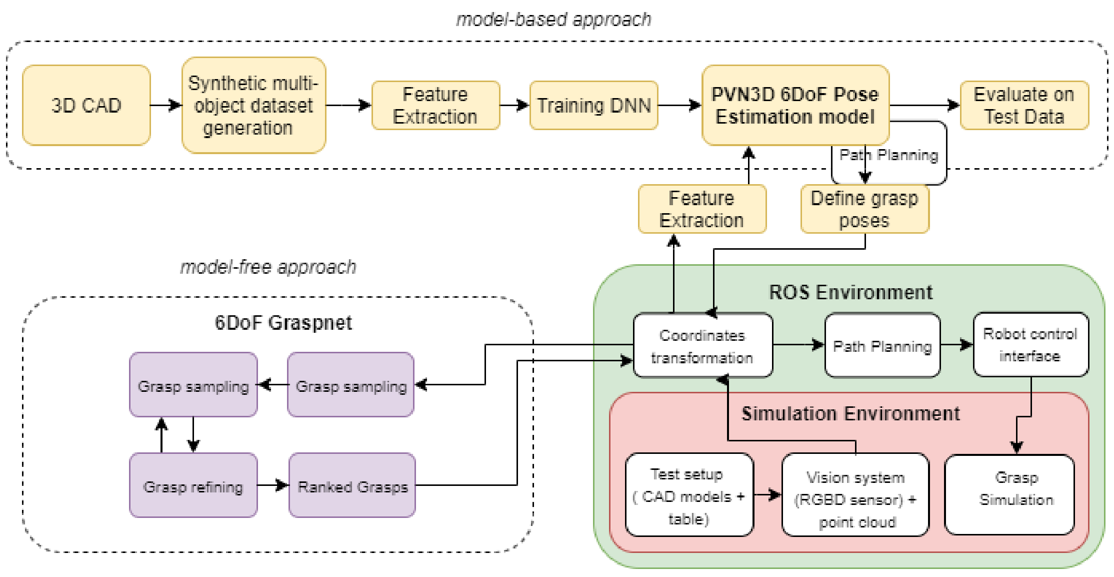


Figure 3.1. *Overall methodology*

As illustrated in figure 3.1, PVN3D [47] is the model-based grasp detection approach adopted in this work. The core machine learning approach in this method is supervised learning. The network is trained on a synthetic dataset. The generation of the dataset is discussed in the section 3.1. A concise description of the network architecture and training is presented in the section 3.2. Graspnet [10] is the model-free approach which follows unsupervised learning. Therefore it directly detects grasps on novel objects. The network architecture is introduced in the section 3.3. Simulation based testing is a convenient method for research due to low-cost setup,less complexity and flexibility. The section 3.4 presents the simulation of grasp manipulation tests. The simulation environment contains three main elements, namely vision system, test setup and robot setup. The setup runs on ROS platform which supplies python libraries for robot controller and path planning.

3.1 Synthetic Data Generation

Obtaining data for training has been a bottleneck in developing DNNs aimed at object detection, segmentation, classification, pose estimation etc. With the development of software capabilities several methods to render photo-realistic synthetic data has surfaced. In this work, an approach based on gazebo simulation environment and CAD models is introduced.

3.1.1 Gazebo Simulation environment

Gazebo is an open source simulation environment which is compatible with ROS. It supports simulation of various sensors, actuators and models. Gazebo provides a wide range of resources for testing , training AI algorithms and rendering datasets. Gazebo utilizes OGRE which is an object-oriented graphics rendering engine. This utility enables producing datasets with lighting variations, shadowing and surface textures. The 3D cad models types currently supported by gazebo are as follows

- **Wavefront files (.obj)**

  Wavefront files presents the 3D geometry using a mesh. The mesh is defined using the coordinates of the vertices, vertex normals, polygonal faces which connect neighbouring vertices. In addition to the details about the mesh geometry, .obj files also include information about the surface texture

- **3D Systems files (.stl)**

  .stl files have a similar representation of the mesh to that of .obj files. The exception is that .stl donot include the information about surface texture. This is disadvantageous if the algorithm performances is affected by the object texture. A .stl file imported to gazebo must be assigned with surface colours in model editor if the algorithm is associated with the semantics.

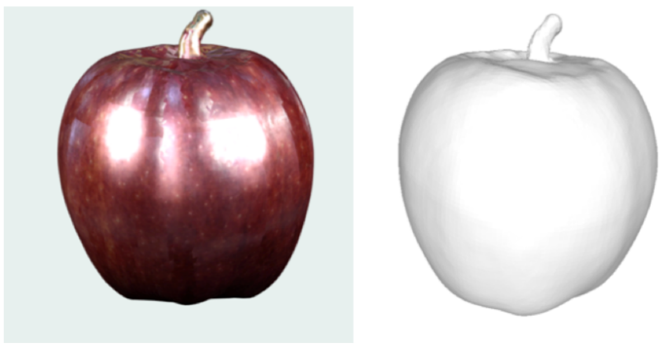


Figure 3.2. *An .obj file on left and .stl file on right - Image from [75]*

- **Collada files (.dae)**

.dae is a more interactive and sophisticated 3D shape presentation which supports number of platforms. The information included are geometry, texture, animation and physics etc

- **Scalable Vector Graphics files (.svg)**

  .svg is a 2D image representation based on vectors. These file types are based on XML files which allows them to be edited using text editors.

A dataset of 3D CAD models consisting of 6 parts was chosen for the implementation from the following source: `https://www.thingiverse.com/thing:3629243`. The dataset is identified by the name Planetary-gear dataset in the preceding sections of this report. The figure 3.3 illustrates the CAD models and the relevant names.

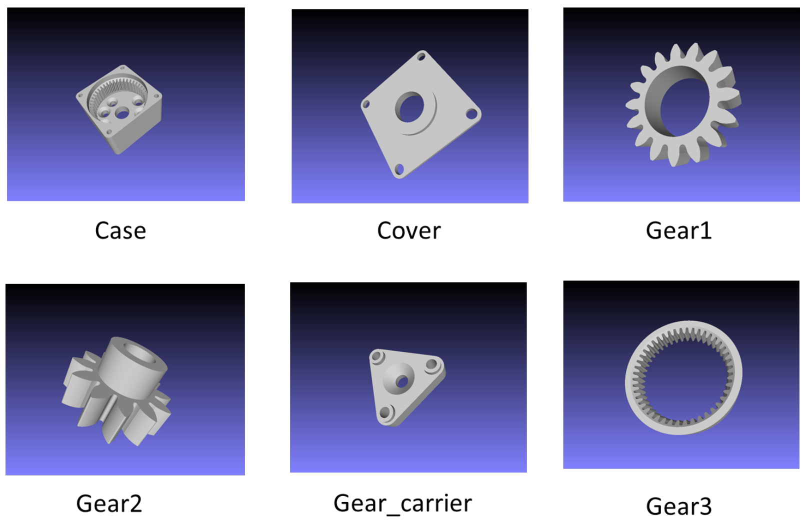


Figure 3.3. *The CAD models contained in the datasets*

3.1.2 View point Sampling

The CAD models were arranged in the gazebo simulation as illustrated below in figure 3.3. The objects are arranged in slight clutter. No special lighting variations or shading used. The scene contains only the objects belonging to dataset and objects lie on their most stable position around origin. As the CAD files used in this implementation were .stl files, a unique color was assigned to each object from the options available in gazebo. View point sampling aims at capturing view points as much as possible from the upper hemisphere of gazebo environment around the origin. The repetitive and approximately similar instances of images must be eliminated in order to prevent over learning in training.

For this purpose a ROS depth camera integrated with a gazebo depth camera plugin is simulated to move in a predefined sequence of view points to capture images. The sensitivity of depth image of a kinect-camera ranges upto 3m.

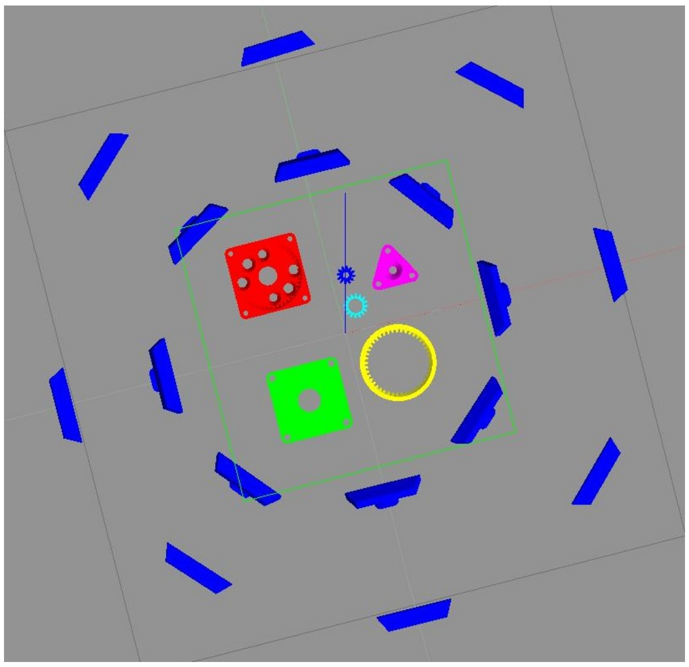


Figure 3.4. *View point sampling Algorithm*

Hinterstoisser et al. presents a view point sampling technique called hemisphere sampling. A similar pattern is used in positioning the camera in this implementation. The camera changes its pose in the world coordinate system while pointing out the X-axis of the camera coordinate system towards the origin of the world coordinate system. The camera iterates through the following increments in the process.

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

At each sampling point the camera captures and saves an RGB image I_{RGB}, depth image I_D a binary mask I_M, Ground truth labels, Ground truth poses in both camera and world coordinate frames T_O^C, T_O^W $(4 \times 4$ homogeneous transformation matrices). Figure 3.5 illustrates a sample of images captured

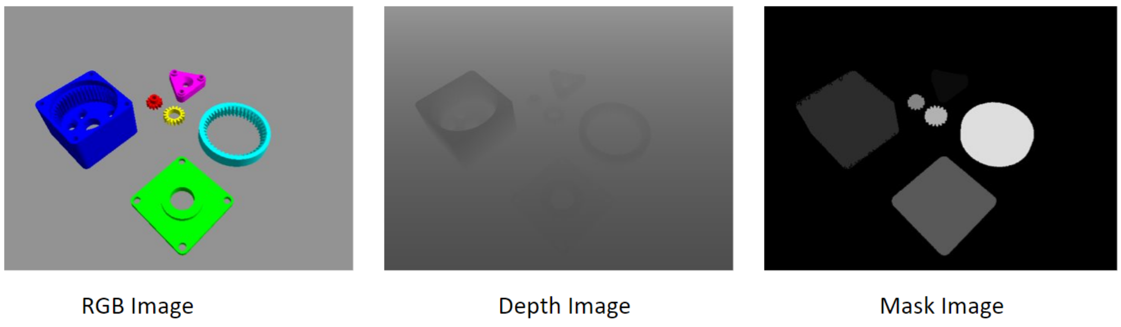


Figure 3.5. *View point sampling Algorithm*

The RGB and depth images can be accessed directly by subscribing to respective ROS topics. But the grayscale binary masks are produced using a sub-process. An artificial point cloud is sampled on the mesh files using open3d python library. These points are back projected on to the gazebo environment while assigning grayscale values which ranges in intensity according to the distance from the camera. The assigned grayscale value is saved as the ground truth label for each object class. The ground truth pose of each object is recorded with reference to camera coordinate frame T^{camera}_{object}. The transformations T^{world}_{camera} and T^{world}_{object} can be directly obtained from the simulation. Then the T^{camera}_{object} can be obtained as follows;

$$^{camera}T_{object} = \left(^{world}T_{camera}\right)^{-1} \times {}^{world}T_{object} \tag{3.1}$$

The overall process of dataset generation can be concluded as follows

Algorithm 1: Automatic dataset generation

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

3.2 Model-based Multi-class Grasp detection

3.2.1 Pose Estimation

Pose of a 3D object in space can be defined as its location(x,y,z) coordinates and orientation (α,β,γ) angles, a total of 6 parameters in world coordinate system. Pose estimation process involves 3 coordinate frames, namely object coordinate frame, world coordinate frame and the camera coordinate frame. Hence , the task of pose estimation is defined as estimation of rigid transformation of an object from its world coordinate system to camera coordinate system. This is illustrated in the equation 3.1. The4×4 homogeneous transformation matrix consists of Rotation in 3D (**R**) and translation (**t**)

3.2.2 Network Architecture

PVN3D network [47] is an open source deep neural network based on PyTorch platform. The following implementation is adapted from the github repository at : `https://github.com/ethnhe/PVN3D`

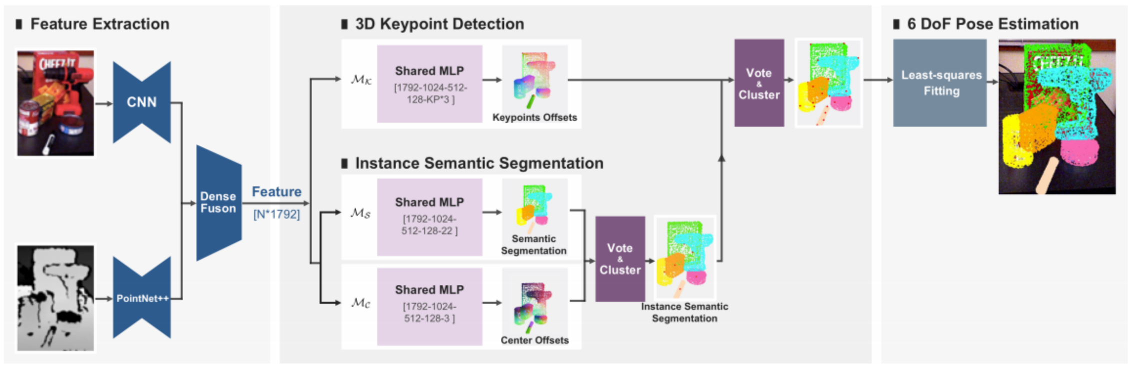


Figure 3.6. *PVN3D network Architecture - Image from [47]*

- **Feature-Extraction**

  PVN3D network utilizes two previous work, PSPNet [76] and Pointnet++ [66] for feature extraction. PSPNet extracts features required for semantic segmentation from RGB images. Pointnet++ extracts the geometric features from the input point cloud. A Densefusion [49] network fuses the two feature channels to produce one feature vector as the input to MLPs.

- **3D-keypoint detection**

  This module predicts the offsets from visible seed points to target key-points. The offsets are measured in terms of per-point euclidean distance metric. Each object class in the dataset has a predefined set of target key-points for each instance. The visible seed points and the predicted offsets vote for the target key-points. The voted points are gathered and clustered using clustering algorithms and cluster-centers are noted as detected key-points.

- **Instance semantic-segmentation**

  Instance semantic segmentation includes two MLPs for semantic segmentation and center offsets voting module. The semantic segmentation and key-point detection module are jointly optimized and they enhance the performance of each other. Both local and global features are extracted in this procedure to predict semantic labels of the objects. Center offset voting network distinguishes object instances.

- **6 DoF Pose-estimation**:

  A least-squares fitting algorithm implemented on keypoints predicted by the network in the camera coordinate frame and ground truth keypoints in object coordinate frame.

3.2.3 Optimization problem

The training of the network is aimed at optimizing the 3 MLPs described in the previous section. The corresponding trained weights of the network are saved for later when the model is used for testing. The optimization problem defines a loss function for each module. The MLPs are trained to optimize the loss functions with the help of PyTorch, CUDA and GPU support.

Keypoint offset detection module $\mathcal{M}_k$ minimizes the per-point Euclidean distance offset from *n* visible seed points to *m* selected key-points from the point cloud (of^n_m), against the per-point ground truth key-point offset. This is an optimization of the mean absolute error (MSE). The relevant loss function can be formulated as below

$$\mathcal{L}_{keypoints} = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} of^n_m - of^{*n}_m \mathbb{I}(p_i \in I) \tag{3.2}$$

where N is the total number of seed points, M is the total number of selected key-points , $\mathbb{I}$ is an indicating function when point p_i belongs to instance *I* and of^{*n}_m is the ground truth offset.

Instance Semantic Segmentation Module $\mathcal{M}_f$ is dissimilar to previous module from its output. In classification and segmentation MLPs output is a probability for class label. Hence, instead of a regression loss function , a focal loss is implemented.

$$\mathcal{L}_{semantic} = -\alpha(1 - q_n)^{\gamma} \log(q_n) \tag{3.3}$$

where γ is focusing parameter and α the balancing parameter. The output q_n is a vector of the product of class confidence and one hot coded representation of class label which always corresponds to a value between 0 and 1.

Center Point Detection module $\mathcal{M}_\rfloor$ uses a similar regression loss function to the $\mathcal{M}_\int$ module. This module calculates the Euclidean distance offset of each seed point to the centers of the objects. The loss function can be simply formulated as follows.

$$\mathcal{L}_{center} = \frac{1}{N}\sum_{n=1}^{N} \delta \mathsf{x}_i - \delta \mathsf{x}_i * \mathbb{I}(p_i \in I) \tag{3.4}$$

3.2.4 Training the network

A combined Multi task loss can be supervised by combining the loss functions in equations 3.2, 3.3,3.4 as follows.

$$\mathcal{L}_{Multitask} = \gamma_1 \mathcal{L}_{keypoints} + \gamma_2 \mathcal{L}_{semantic} + \gamma_3 \mathcal{L}_{center} \tag{3.5}$$

In the above multi-task loss the parameter values are set as $\gamma_1 = \gamma_2 = \gamma_3 = 1$ and all the 3 MLPs are optimized simultaneously. In the data generation step a total of 3052 images were obtained. The dataset was preprocessed to a 75%-25% split between the training, evaluation and testing. The input images are of the resolution 640×480. The authors of the original paper recommends sampling 12288 points from the point cloud as seed points. A Wrap padding is applied in the sampling procedure to limit the seed points to above recommendation. The 3 shared MLPs were trained for a total of 70 epochs in batches of 24. The network was trained using 4 Nvidia v100 GPUs.

3.3 Model-free Grasp detection

Graspnet [10] introduced a model-free framework for successfully generating multiple sets of stable grasps for novel objects. As discussed in chapter 2 , model-free methods do not require an accurate 3D model of the objects to be grasped. Graspnet features a variational auto-encoder (VAE) which maps a partial point cloud of a visible object to a set of grasps. This method produces more diverse grasps along object geometry while covering a large area of grasp-space compared to other grasp detection methods [54], [73], [11]. The majority of grasps produced are functional and feasible while the number of failure grasps are minor. This is due to the grasp evaluator embedded in their network. The grasp evaluator maps the object point cloud and point cloud of the robot gripper in to the same space to assess the quality of the detected grasps. Graspnet produces a success probability rather than a binary label (fail-success) which protects feasible grasps. The refinement step further protects the near-failure grasps by searching for a transformation that transforms the grasps to a successful grasp pose.

3.3.1 Variational Autoencorders

Variational Autoencorders are an element of deep generative models. These follow an unsupervised deep learning model where high dimensional data are mapped into a low dimensional latent space in order to learn a representation or an encoding of some known data. This called a generative network. The generative network is followed by a discriminative network where the learned representations are reconstructed as close as possible to the learned encodings. The two components are known as the encorder and decorder. The encorder is trained extract useful features from the latent space while decorder is trained to decode the encodings and reconstruct the original data.

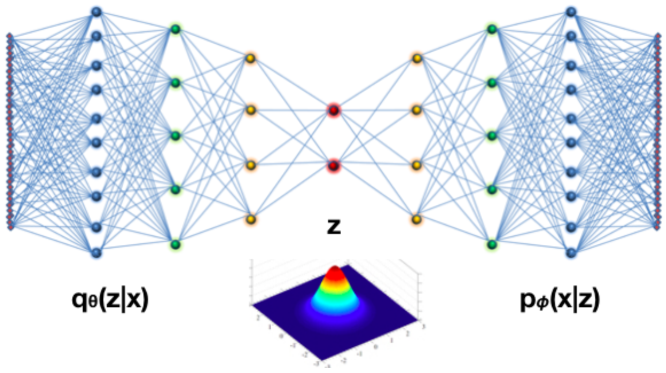


Figure 3.7. *VAE Illustration by Stephen G. Odaibo, M.D. [77]*

3.3.2 Network Architecture

Graspnet adopts a similar network architecture as VAE explained in figure 3.7 but includes additional features to suit the purpose. Graspnet features 3 main sections of their grasp detection network. The grasp sampler which reflects the functionality of VAEs is capable of producing grasps of novel objects on partially observed point clouds. The sampled grasps are then evaluated in the second step. A third step performs iterative refinement of the evaluated grasps which ensures the advantages discussed about graspnet previously.

Graspnet aims at learning the posterior distribution $P(G|X)$ where G represents the space of successful grasps and X represents the partial point clouds. Each grasp $g \in G$ is represented by $(R,T) \in SE(3)$ where $R \in SO(3)$ and $T \in \mathbb{R}^3$. The grasps are detected in object coordinate frame and origin in located at the center of the observed point cloud. The axes of the object coordinate frames are defined parallel to the camera coordinate frame.

3.3.3 Optimization problem

- **Variational Grasp Sampler**

  The observed point cloud and the grasps are transformed to a latent space *Z* by the encoder. The latent space variable is defined *z* and the respective probability distribution is *P(z)* $\sim \mathcal{N}(0, I)$. Hence, the posterior distribution of sampled grasps can be interpreted as follows:

$$P(G|X) = \int P(G|X, z; \Theta)P(z)dz \tag{3.6}$$

  The integration over all positive grasps *g* in equation 3.6 yields to complexities. Therefore each grasp *g* and point cloud *X* is mapped to a subset *Q* of the latent space *z* as *Q(z|X,g)* by the encoder. The reduced grasps are reconstructed by the decoder $\hat{g}$. In the process, both encoder and decoder learns to optimize the reconstruction loss $\mathcal{L}(g, \hat{g})$. To calculate the reconstruction loss in terms for transformation parameters (Orientation and translation) a predefined set of points on the gripper point cloud are noted and the respective transformation from ground truth grasp to reconstructed grasp is defined as $\mathcal{T}(.; p)$

$$\mathcal{L}(g, \hat{g}) = \frac{1}{n}\sum \|\mathcal{T}(g,p) - \mathcal{T}(\hat{g},p)\| \tag{3.7}$$

  Further, the KL-Divergence between the subspace distribution and normal distribution is optimized. The loss function is formulated as follows;

$$\mathcal{L}_{vae} = \sum_{z\sim Q, g\sim G} \mathcal{L}(g, \hat{g}) - \alpha \mathcal{D}_{KL}[Q(z|X,g), \mathcal{N}(0, I)] \tag{3.8}$$

- **Grasp Evaluator**

  While grasp sampler works as a generative network , grasp evaluator acts as a dis-

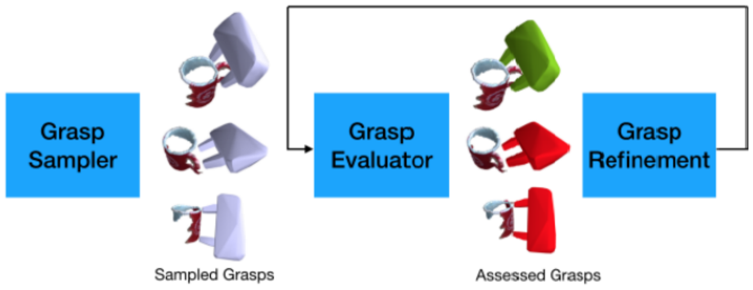


Figure 3.8. *Overview of Graspnet Network Architecture [10]*

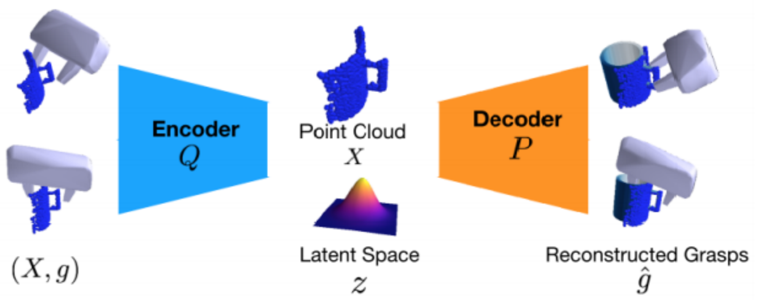


Figure 3.9. *Grasp sampling procedure [10]*

criminative network. Since grasp sampler learns only on predefined positive grasps ,the generated grasps tend to include false-positive grasps. The grasp evaluator assigns a probability of success for each sampled grasp *P(S|g,X)*. A combined point cloud of both the gripper X_g and the object X is utilized for classifying the grasps. A cross entropy loss function is formulated combining the success probability prediction *s* and a binary ground truth label *y* for each grasp as fail or success.

$$\mathcal{L}_{evaluator} = -(y \log(s)) + (1 - y) \log(1 - s) \tag{3.9}$$

The authors have included hard-coded negative grasps by slightly transforming the positive grasps to either collide with the object or to live too far from grasp distance.

- **Iterative refinement of grasps**

  The grasp evaluation step discards grasps which are close to being successful grasps. A grasp refinement step is introduced to retain such grasps. Graspnet defines a transformation Δg, that converts a near success grasp to a successful grasps. To filter the most effective set of grasps and refine them, the gradient $(\delta S/\delta g)$ of transformation against success probability is utilized. Hense, this is a point-wise operation, a rigid transformation cannot be ensured for X_g. Therefore , X_g is defined to a function of orientation and translation $\mathcal{T}_g$ = (R_g , T_g). The Δg can be calculated using chain rule as in equation 3.10. The hyper parameter η is a scaling parameter which prevents refinements scaling over 1cm.

$$\Delta g = \frac{\Delta S}{\delta g} = \eta \times \frac{\Delta S}{\delta \mathcal{T}(g;p)} \times \frac{\delta \mathcal{T}(g;p)}{\delta g} \tag{3.10}$$

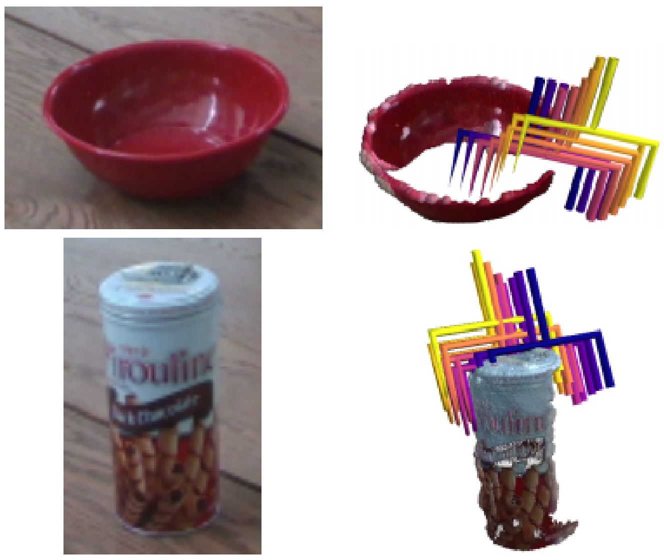


Figure 3.10. *Iterative grasp refinement procedure [10]*

3.3.4 Training and evaluation of the network

The implementation of 6doF GraspNet is adapted from the following github repository : `https://github.com/NVlabs/6dof-graspnet`

The data required for training are collected using on a physics simulator. 206 objects of 6 different categories from ShapeNet[78] has been used in training. The grasp candidates are sampled randomly on the object meshes. For sampling, the z-axis of the gripper has been oriented perpendicular to object surfaces. And grasps have been uniformly sampled from zero to the length of the gripper fingers. The grasps are tested for validity in the same simulation environment. While maintaining a constant friction between the gripper and the objects, a parallel jaw gripper has been used to perform pick up and oscillation. The sampled grasps are then classified as successful and failed. The procedure has sampled over 10,000,000 grasps and over 7,000,000 grasps have been simulated and tested. Ultimately, 2,104,894 successful grasps has been extracted for training the network.

Pointnet++ [66] is being used by both the grasp sampler and evaluator. Each batch of training contains 64 grasps sampled on a random view from each object. And the grasps sets contain 30% positive grasps, 30% negative grasps and 40% hard negative grasps. The KL-Divergence loss has been assigned a weight of 0.01 in equation 3.8. Adam optimizer is used for optimizing the network with a learning rate of 0.0001.

For evaluation, graspnet adopts two metrics; success rate and coverage rate. The proportion of the successful grasps out of total grasp candidates are define as the success rate. A single grasp is not enough for real life application where as there are chances that the produced grasps are not feasible when tested with other external factors. Therefore, the term grasp coverage defines the amount of diversely sampled positive grasps out of the generated grasps. This success rate and coverage rate represents the precision and recall in binary classification.

3.4 Simulation of Grasp Manipulation

The simulation set up is based on ROS and consists of 3 main elements. ROS-control, Robot Simulation and Motion planner. The robot used in the simulation is FRANKA EMIKA Panda robot. Franka panda provides ROS package which connects the robot with ROS control. Franka Panda also provides a ROS based simulation package which is compatible with robot-agnostic motion planning package called Moveit.

3.4.1 ROS Control

ROS Control (`ros.org`) provides a functional ecosystem to control joints or actuators of real robots or simulations. A controller takes an input value as a feedback and calculates the error against a set value. Then the output is controlled in a way that the error is minimized. Generally, the controller is a PID controller. ROS control provides several controllers

- **Effort controllers** : To control torque/force applied to joints of the robot
- **Joint state controllers** : To publish the state of the joints of the robot
- **Position controllers** : To control the position of one or multiple joints of the robot
- **Velocity controllers**: To control one or multiple joint velocities at once
- **Joint trajectory controllers** : To control the trajectory of a joint combining all the above controller types

The ROS-control is unable to read the necessary inputs directly. They communicate with hardware interfaces which obtain the necessary inputs from robot hardware. The type of input depends on the type of controller. few of the main hardware interfaces available in ROS control are as follows;

- **Joint command Interface** : An interface that can deliver effort/position/velocity commands top multiple joints
- **Joint state interface** :An interface to read the state of multiple joints
- **Actuator state interface**: An interface to read the state of multiple actuators

In the case of a joint position being controlled , the joint state interface reads the state of the joint by communicating with the robot hardware. Generally, the joint state is read via the encoder embedded to robot hardware. The encoder reading is then fed into a PID controller which acts as the joint position controller. The PID controller produces a responsive output by calculating the error between read position and the set position by the ROS interface. The generated output is transferred to joint command interface which commands an effort (a torque) to the embedded robot controllers which control the actuators. This is explained in the figure 3.11

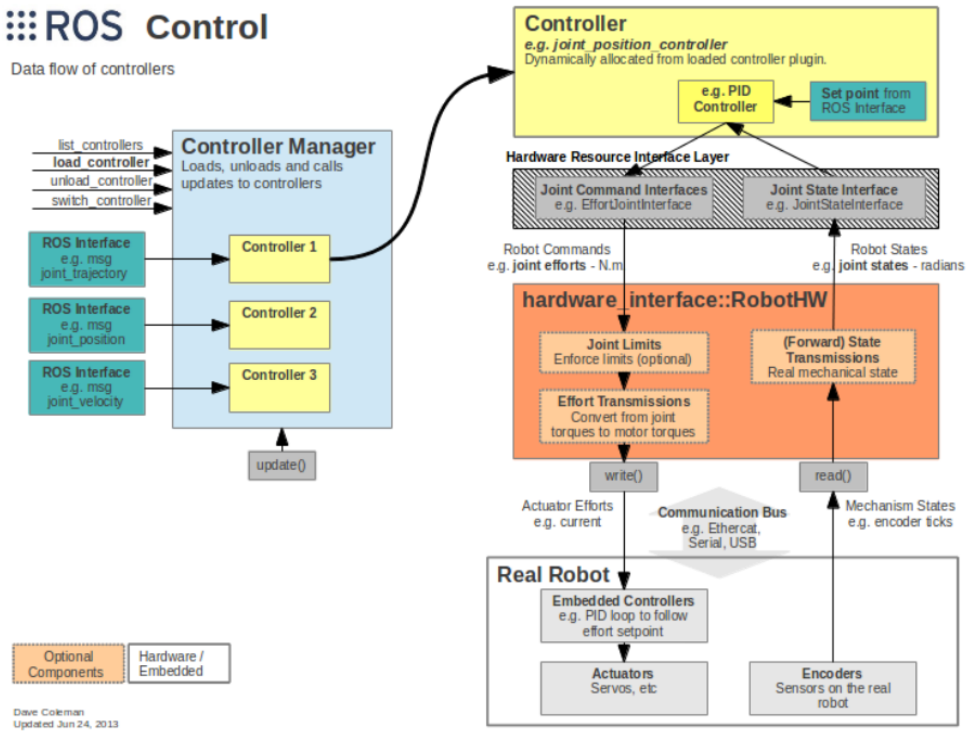

ROS Control
Data flow of controllers
list_controllers
load_controller
unload_controller
switch_controller
Controller Manager
Loads, unloads and calls updates to controllers
ROS Interface e.g. msg joint_trajectory
ROS Interface e.g. msg joint_position
ROS Interface e.g. msg joint_velocity
Controller 1
Controller 2
Controller 3
update()
Controller
e.g. joint_position_controller
Dynamically allocated from loaded controller plugin.
e.g. PID Controller
Set point from ROS Interface
Hardware Resource Interface Layer
Joint Command Interfaces e.g. EffortJointInterface
Joint State Interface e.g. JointStateInterface
Robot Commands e.g. joint efforts - N.m
Robot States e.g. joint states - radians
hardware_interface::RobotHW
Joint Limits Enforce limits (optional)
(Forward) State Transmissions Real mechanical state
Effort Transmissions Convert from joint torques to motor torques
write()
read()
Actuator Efforts e.g. current
Communication Bus e.g. Ethercat, Serial, USB
Mechanism States e.g. encoder ticks
Real Robot
Embedded Controllers e.g. PID loop to follow effort setpoint
Actuators Servos, etc
Encoders Sensors on the real robot
Optional Components
Hardware / Embedded
Dave Coleman
Updated Jun 24, 2013

Figure 3.11. *ROS Control - Image from ros.org*

3.4.2 Frank-Panda Simulation

An open source FRANKA EMIKA Panda simulation package based on Gazebo environment is provided at : `https://github.com/erdalpekel/panda_simulation`.

The package requires the following dependencies;

- **libfranka** : C++ package provides low level control for FRANKA EMIKA products.
- **franks-ros** : The package for integrating the Robot hardware interface (Frank Control Interface) and ROS control
- **panda moveit configuration package** : The package for motion planning of FRANKA EMIKA Robots

In the simulation, the robot joints are assigned a damping coefficient and each link has defined its inertia matrix based on the geometrical features of its mesh file. Each link has a mass assumed by the volume of each link and density corresponding to mild Aluminium as the material. The friction coefficient is set to 0.61 corresponding to the material. The motion planning is altered to follow joint trajectory which corresponds to the controller simulated by the package.

The controllers used by the implementation are effort controllers and joint trajectory controllers. The inputs to this controller are joint position and velocities. The controller regulates the force/torque to the actuators of the simulated robot. The joint states are published via a joint states controller. The hardware interface type is effort-joint control, which permits control of the gripping via a torque/force application which enables a better gripping than velocity-joint hardware based controls. The friction between the object and the gripper is maintained by exerting a constant force on the object by the gripper.

The visual input is required for grasping algorithms discussed in the previous sections. An Xbox Kinect camera is connected to a link of the robot and the camera is simulated using a plugin which enables the functionality of a real kinect depth sensor. Amendments are made to cancel collisions between certain links of the robot and the kinect camera for simulation. To ensure the grasp force , a force torque is connected to the gripper. The force sensor also provides the functionality of a real sensor via a ROS plugin. A pregrasp link is connected to link 7 of robot, with an offset of 18cm in z-direction. This link is configured as the tool.

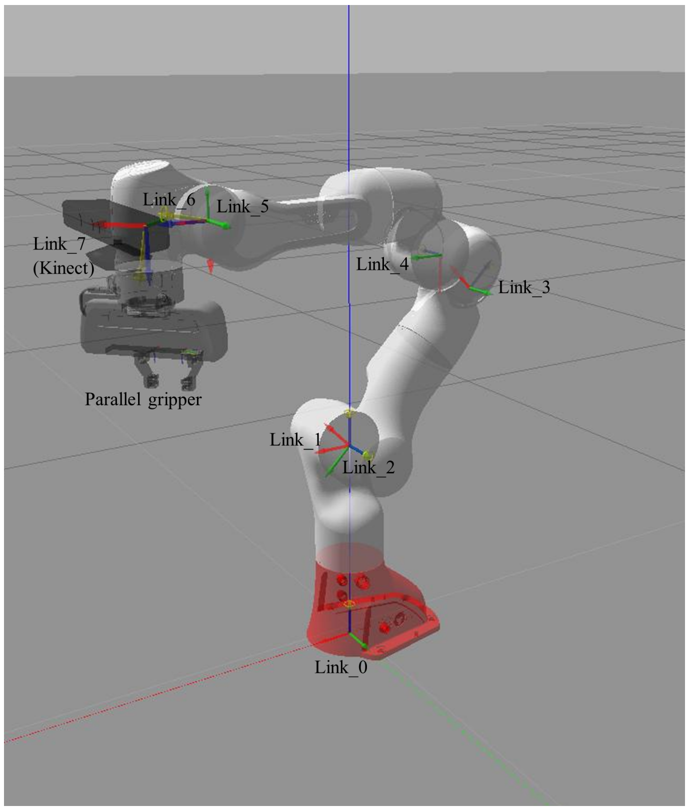


Figure 3.12. *FRANKA EMIKA Panda Simulation Configuration*

3.4.3 Grasp manipulation setup

The grasp manipulation is simulated in the same gazebo environment using the FRANKA EMIKA Panda Robot simulation, Object CAD files, picking and placement platforms.

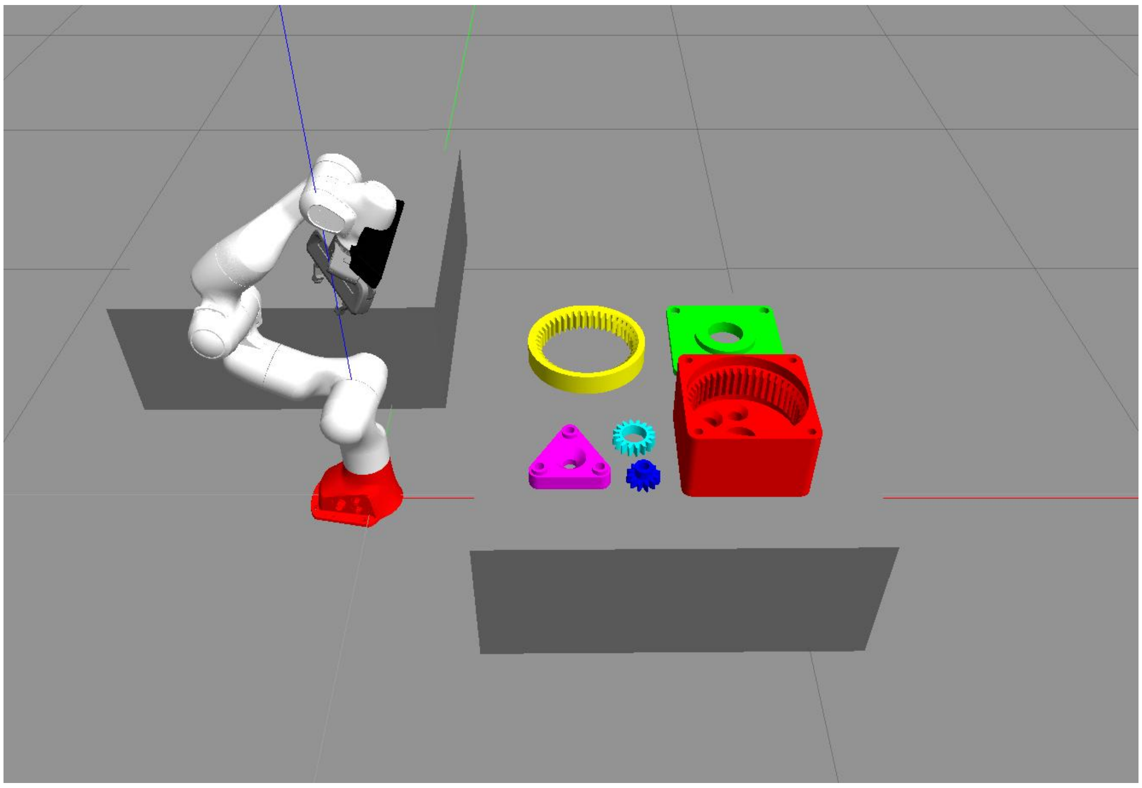


Figure 3.13. *Grasp Manipulation setup based on FRANKA EMIKA Panda robot*

3.4.4 Experiments

This section explains a set of grasp manipulation experiments performed using the gazebo simulation set up explained in figure 3.13

The grasp detection methods using both model-free and model-based approaches are deployed for grasp manipulation experiments in this section. However, the two methods have different characteristics in performing grasp manipulation. The model-based technique is a form of pose aware grasping. The robot has an idea about the object being grasped. Therefore the robot is able to perform the grasp manipulation to place the oibject according to a predefined object pose. The model-free approach performs grasp manipulation without any information about the pose. Therefore, placement of the object to a predefined pose is not an option. Considering the limitations , Testing for grasp success using a pick up and oscillation procedure for model-free grasping method and model-based grasping method was carried out seperately.

- **Testing for pose-unaware grasping**

  The experimental setup tests for the grasping success using the approach described in section 3.3. As the test begins, robot moves to home position and performs grasp detection. Grasp detection follows a procedure where point clouds are clustered and sorted according to the distance from the camera. Then for each cluster the generated grasps are ranked and filtered. Each of the sorted grasp poses are approached by the robot gripper based on robot motion planning. The object is lifted up and tested for grip using the force sensor in gripper. If the object remains in grip the grasp test is recorded as success.It follows a stability test where the gripped object is oscillated by a predefined motion and object being checked again if remains in grip.

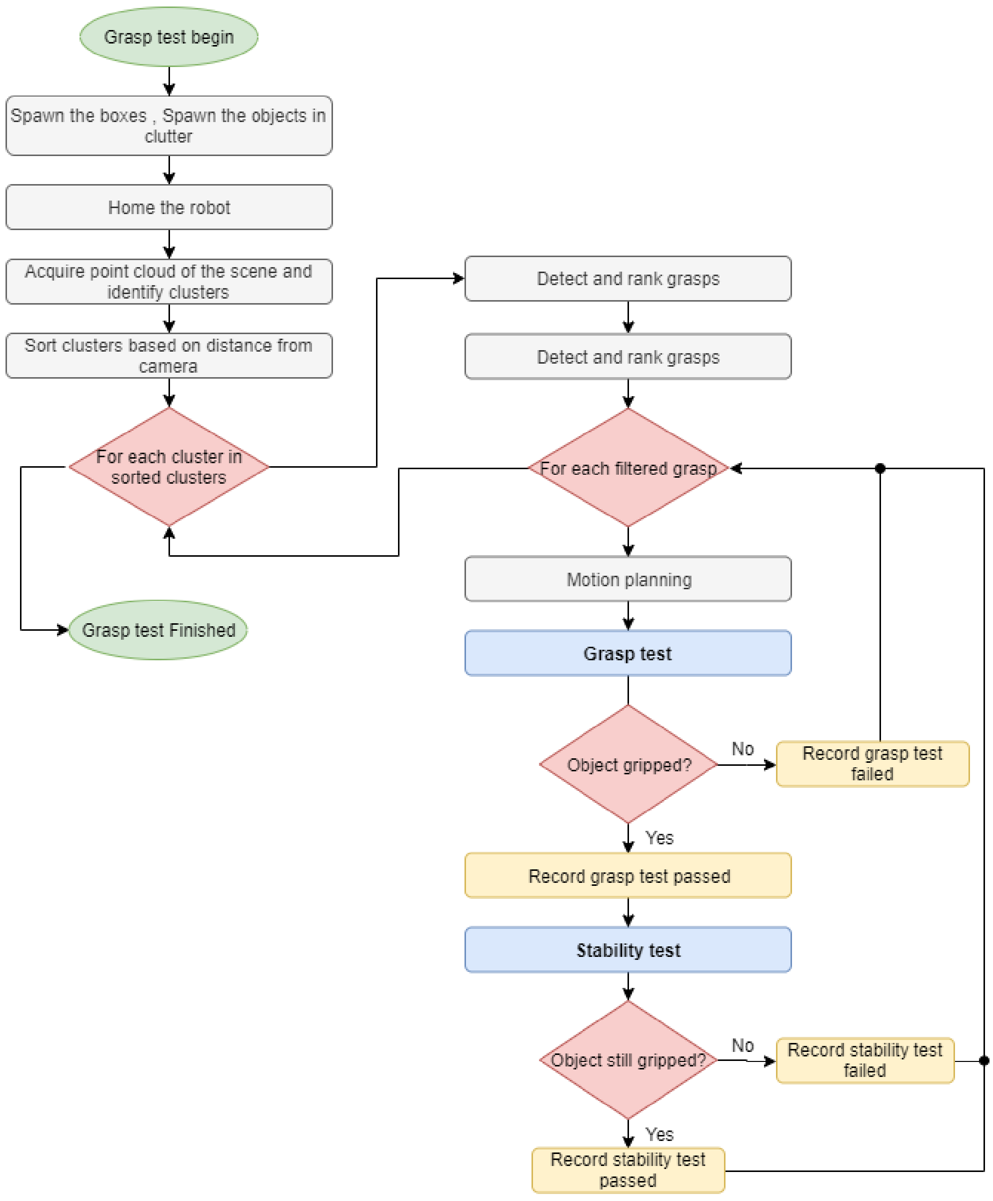


Figure 3.14. *Pose unaware grasp test*

- **Grasp Filtering process**

The grasp detection algorithm generates multiple grasps scattered arbitrarily in 3D space of the object. Some of the instances are not feasible to be approached by the robot gripper due to kinematic restrictions. Two most evident types of grasps that must be pruned are illustrated in figure 3.15.

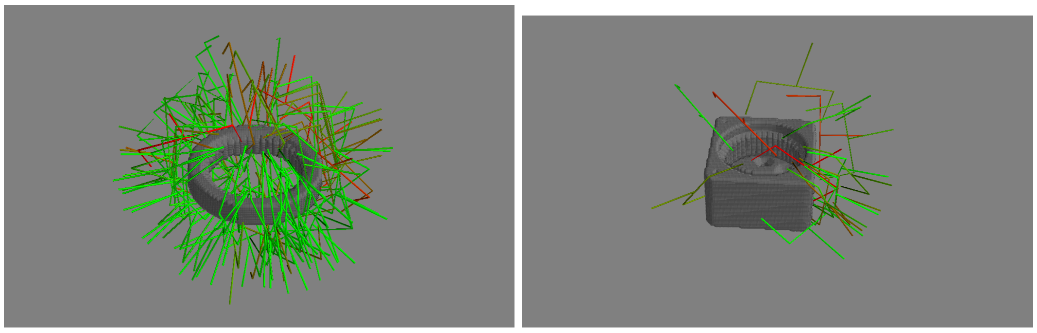

(a) *Grasp samples including bottom-up oriented instances* ***(b)*** *Grasp samples including instances where approach vector oriented towards camera*

Figure 3.15. *ill-posed grasp samples*

The orientation of the detected grasps are presented in euler angles (roll-pitch-yaw). There exists an ambiguity when considering the order of angles the grasp would follow. Therefore, these angles are not directly helpful in the grasp filtering process. The approaching vector of the grasps is more useful representation for filtering grasps. The unit approaching vector can be deduced from the homogeneous transformation matrix of grasp pose. Then, the approaching vector is decomposed to orthogonal projection vectors in right handed Cartesian coordinate system. An example of calculating XY plane projection vector is presented in equation 3.11.

$$I_{XY} = N_{XY} \times (A.N_{XY}) \tag{3.11}$$

$$\theta_{XY} = \tan^{-1} \frac{I_{XY}.\hat{\imath}}{I_{XY}.\hat{\jmath}} \tag{3.12}$$

I_{XY} is the projection of the unit approaching vector of the detected graph in XY plane. N_{XY} is unit the Normal vector for XY plane. A is the unit approaching vector. θ_{XY} is the angle of the projection vector on each plane. considering $\hat{\imath}, \hat{\jmath}, \hat{k}$ as the unit vectors of XYZ Carteesian plane, the projection angles θ_{YZ}, θ_{XZ} can be calculated following the equation 3.12.

The Z axis is the approaching axis of the gripper. Grasp filtering refers to filtering the unfavourable projection angles $(\theta_{XY}, \theta_{YZ}, \theta_{XZ})$ with reference to approaching axis of the gripper in order to avoid the instances illustrated in figure 3.15. The projection angle θ_{XY} is restricted between $-90°$ and $90°$ to filter out instances in figure 3.15a. The instances in figure 3.15b can be filtered by restricting the smaller angle out of θ_{YZ}, θ_{XZ} between $-90°$ and $45°$. The figure 3.16 illustrates the result after filtering unfavorable grasps

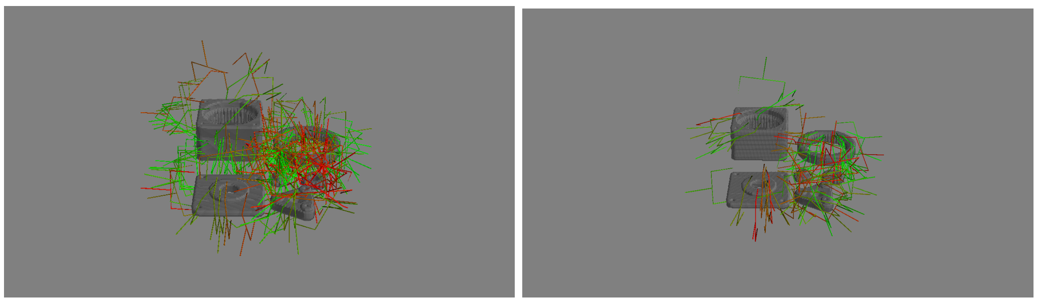

(a) *Grasps before filtering process* ***(b)*** *Grasps after filtering process*

Figure 3.16. *Grasp filtering procedure*

- **Testing for pose-aware grasping**

Pose aware grasping adapts the model-based approach presented in section 3.2. The method estimates the pose of the object with reference to camera coordinate frame as the initial step. The arrangement of the objects on picking box replicated the arrangement used for training. This is to improve the accuracy of semantic segmentation module of the method. The experimental procedure is similar to the grasp unaware-procedure with the exception of grasp filtering. Model-based grasping approach transfers predefined set of grasps to the detected object pose. The requirement for filtering grasps is eliminated as the defined grasps utilize expert knowledge about object geometry and constrains in the grasping scene.

The robot moves to the defined home position and observes the object scene using the depth sensor. The pose detection algorithm estimates the pose of each identified object. The corresponding predefined grasps are transformed to the object coordinate frame with reference to the pose of the object. Each of the defined grasps are carried out sequentially starting from the object closest to the camera. Robot gripper moves to each grasp pose and grips the object by closing the parallel grippers. The object is lifted and tested for grasping success using the torque sensor. If the test is passed , the oscillation tests for the stability of the grasp. The flowchart for the procedure is illustrated in figure 3.17

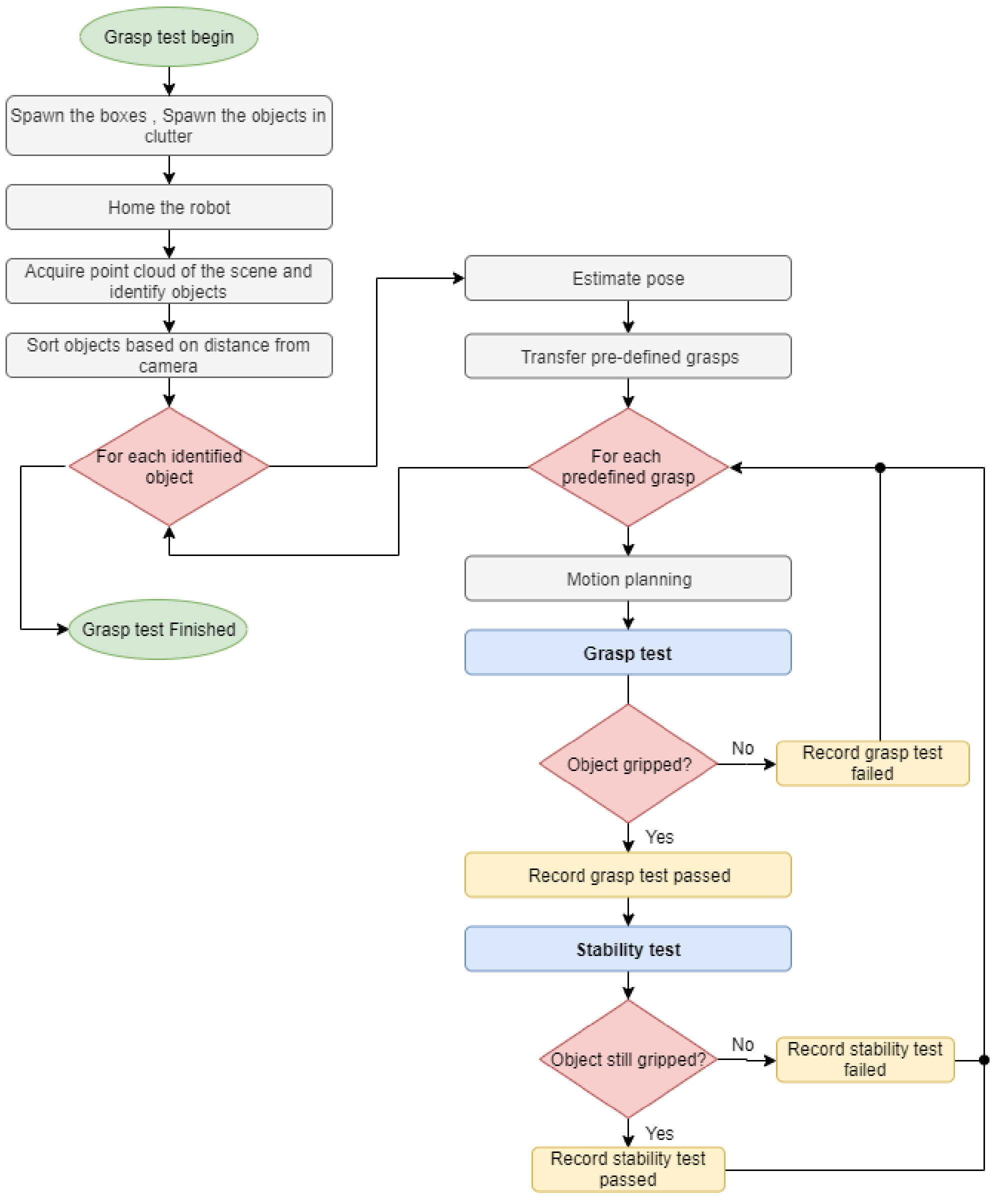
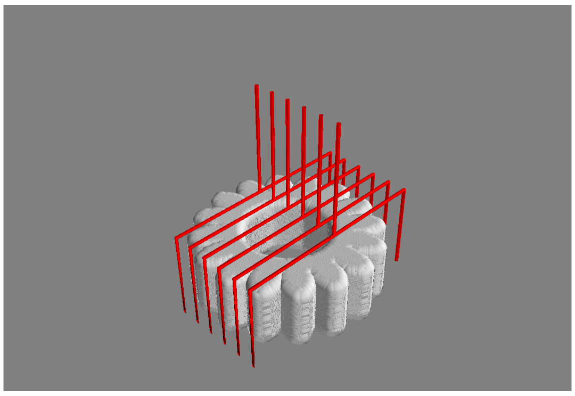

Grasp test begin
Spawn the boxes , Spawn the objects in clutter
Home the robot
Acquire point cloud of the scene and identify objects
Sort objects based on distance from camera
For each identified object
Grasp test Finished
Estimate pose
Transfer pre-defined grasps
For each predefined grasp
Motion planning
Grasp test
Object gripped?
No
Record grasp test failed
Yes
Record grasp test passed
Stability test
Object still gripped?
No
Record stability test failed
Yes
Record stability test passed

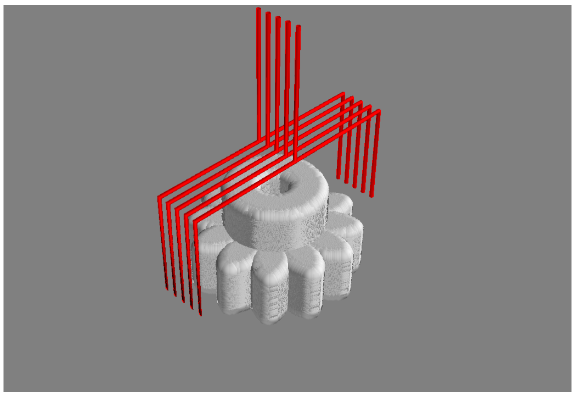
Figure 3.17. *Pose aware grasp test*

- **Predefined grasps**

  The predefined grasps in pose-aware grasping are sampled manually. The grasps are sampled uniformly on each of the objects in the dataset. The grasps are defined with respect to the object coordinate frame. The grasp sampling follows a set of guidelines to protect the feasibility of grasp manipulation. The grasps closer to the ground plane lesser than 3cm are avoided. The grasps oriented bottom-up are avoided. In other cases the sampling of grasp poses completely rely on the object geometry and stable pose. The figure below illustrates the grasps defined for each of the objects 3.18



(a) *Gear1*



(b) *Gear2*

Figure 3.18. *Predefined grasps*

4. RESULTS

4.1 Pose Estimation and Semantic Segmentation

The trained model explained in section 3.2 is tested on the allocated 25% of the test data. The performance of the model on testing data is presented below.

Average Distance Metric

The performance of pose evaluation is based on **ADD** (Average Distance metric) [79]. given a set of 3D points belonging to a certain object category as *x*, estimated rotation **R** and estimated translation **T**, ground truth rotation $\tilde{\mathbf{R}}$ and translation $\tilde{\mathbf{T}}$, the average point wise distance is calculated as illustrated in the equation 4.1

$$ADD = \frac{1}{m}\sum_{x\in M}||(\mathbf{R}x+\mathbf{T})-(\tilde{\mathbf{R}}x+\tilde{\mathbf{T}})|| \tag{4.1}$$

M is the set of points and *m* is the number of points. In symmetrical objects the equation 4.1 could produce repetitive entries due to the ambiguity of pose. **ADD-S** metric eliminates the repetitive entries by selecting only the minimum of the distances associated with a point being considered.

$$ADD-S = \frac{1}{m}\sum_{x_1\in M} min_{x_2\in M}||(\mathbf{R}x+\mathbf{T})-(\tilde{\mathbf{R}}x+\tilde{\mathbf{T}})|| \tag{4.2}$$

ADD(S) is a metric which selectively performs ADD and ADD-S on objects depending on its symmetry.

Area Under Curve (AUC)

The above ADD metric is not a direct indication of the performance of pose estimation. The accuracy of the predictions depend on the level of threshold defined. Therefore, different threshold values assume different performance levels. An accuracy-threshold plot is created by deviating the threshold value and noting down the estimation. The area under the curve (AUC) indicates an aggregate measure of performance across all the threshold values. The value of AUC varies between 0 and 1. The predictions are performing well when the AUC value is close to 1.

Table 4.1. *Inference results on test data*

Model	**ADD**		**ADDS**		**ADD(-s)**	
	8kps	**16kps**	**8kps**	**16kps**	**8kps**	**16kps**
Case	74.68	76.21	85.36	85.46	85.36	85.46
Cover	81.82	80.91	91.81	90.74	91.81	90.74
Gear1	82.07	82.37	92.71	92.90	92.71	92.90
Gear2	69.36	66.41	80.23	79.50	69.36	66.41
Gear3	66.09	63.27	79.42	78.30	79.42	78.30
Gear_Carrier	83.55	83.89	91.88	91.82	83.55	83.89
Total average	76.26	75.38	86.90	86.33	83.70	82.84

The resulting images of bounding box representation of the object pose estimation are illustrated in figure 4.1

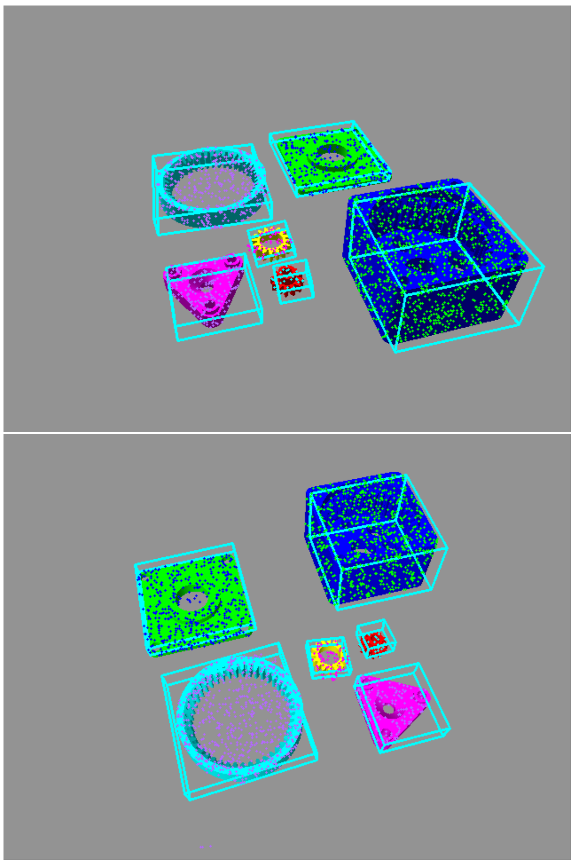


Figure 4.1. *Instance segmentation with 3D Bounding box representation of pose*

4.2 Pose aware grasping results

The table 4.2 presents a summary of pose-aware grasping tests with predefined grasps tested for stability test and grasping test as described in the section 3.4.4. The results are analyzed and discussed in the section 5.3

Table 4.2. *Pose aware grasp test results*

Object	**Total Grasps Generated**	**Grasp test**	**Stability test**
Case	25	32%	20%
Cover	0	0%	0%
Gear1	17	70.6%	58.8%
Gear2	18	72.2%	50%
Gear_carrier	24	33.3%	29.2%
Gear3	21	81%	66.7%

4.3 Pose unaware grasping results

The table 4.3 presents a summary of pose-aware grasping tests using class agnostic grasp detection described in section 3.4.4. The results are analyzed and discussed in the section 5.3

Table 4.3. *Pose aware grasp test results*

Object	**Total Grasps Generated**	**Grasp test**	**Stability test**
Case	32	12.5%	6.5%
Cover	9	11.1%	0%
Gear1	21	66.7%	33.3%
Gear2	24	66.7%	33.8%
Gear_carrier	31	54.8%	29%
Gear3	27	59.3%	18.5%

5. DISCUSSION

5.1 Discussion on Data generation

Training data generation is a crucial step in blending deep learning methods with robotic grasping. In addition, elimination or alteration made to objects subjected to pose estimation requires complete regeneration of data and retraining of the model. Performing view sampling with a real depth sensor and physical objects is a complex task. Synthetic data generation using graphics rendering engines is a fast and convenient alternative. The proposed method extracts details from the CAD files and does not depend on the availability of the physical models. Hence, synthetic data generation is a progressive step towards addressing the bottleneck in acquiring training data efficiently.

The proposed method produced more than 3000 samples of images within a duration of 2-3 hours. An aspect often overlooked is the balance between quality and quantity. Therefore optimizing the view sampling parameters according to the object scene is an essential step in the implementation of the proposed method. The number of files generated has to be concerned as it directly impacts the time consumed by complete process. 3000 samples of RGBD,RGB and Image masks approximate to more than 9000 individual files. The feature extraction from each of the 9000 individual files during the training proved to be highly time consuming. An in-depth analysis on data preprocessing is highly recommended for models associated with large number of input files. Utilizing data compression tools for data migration and storing is more convenient. Furthermore,the proposed method follows a number of sequential preprocessing tasks. sampling point cloud on mesh files, 3D point cloud back-projection, key-point sampling, surface normals extraction are examples of distinct tasks associated with multiple python libraries. Therefore, data preprocessing requires a considerable amount of computing power and scripting.

5.2 Discussion on Training the DNNS

The training requirements for PVN3D proved rather costly in terms of computing resources. Harnessing the power of cloud computing clusters is highly beneficial for training large DNNs. Puhti supercomputer from csc.fi was used for training PVN3D DNN in this thesis work. The important features available in CSC platform are as follows

- CSC allows users to utilize the power of their super computers via virtual login nodes. The users can allocate required number of CPU cores and GPU's required for the task. Puhti supercomputer delivers 700 login nodes, 80 GPU nodes from 320 Nvidia volta V100 GPUs.
- Users can interact with the resources via Linux bash shell. Each user gets the access to a specified disk space for data storage. There storage types are categorized according to the span of time data is stored inorder to prevent accumulation of unwanted files.
- To accommodate high user demand CSC introduces a batch job system where each user has to queue to access resources. The time and resources required for the job must be pre-calculated and request has to be send in advance for resource allocation.
- CSC supports Anaconda package management system, python libraries and machine learning libraries. Some widely used software packages based on the field of research can be loaded as pre-installed modules making it convenient for the developers.

  In the implementation of PVN3D, the virtual environment required for PVN3D has to be created first. Pointnet++ has to be built from source codes and important packages such as point cloud library has to be installed. The training data has to be compressed and uploaded to the workspace. CUDA GPU support can be load as a module along with relevant gcc compilers. Once the environment is successfully created the training can be implemented via a batch job or an interactive session. The trained modules can be downloaded and shared easily.

5.3 Discussion on grasp manipulation test results

The table presents the percentage of attempts passed by each object in grasp test and stability test explained in the section 3.4.4.

The grasp manipulation test results presented in section 4.1 compares the accuracy of grasp poses generated by both model-based and model-free approaches. Model-based approach has access to a predefined set of grasps. The predefined grasps are sampled on the object using expert knowledge on object geometry. The object class 'cover' is a flat object with low depth and a surface area larger than gripper width. Therefore a feasible grasp pose cannot be defined for this kind of an object. However, the deep generative model generates some grasps on the object as emphasized on table 4.3, which slips on the attempts made for grasping. Hence none of the instances pass the stability test as evident from table 4.2.

The object class 'case' is slightly out of the range of the robot. Therefore, the results include grasps failure due to failure in motion planning for both graspnet and PVN3D. The object classes ’gear1’,’gear2’ and ’gear3’ yields comparatively good results (more than 50%) with PVN3D model. It is evident from the results that stability test is challenging more than the grasp test. The table 4.1 shows that accuracy of models trained with synthetic data for pose estimation is more than 75%. However, the results indicate the performance in simulation and the implications on real world performance remains untested in this thesis work.

5.4 Discussion on simulation environment

The external factors which can be controlled in simulation such as lighting variance, texture etc have to be dealt with in real world implementations. The simulation environment provides extended support to perform grasp manipulation testing as described in section 3.4. Simulation of grasp manipulation can foresee constraints and save time in designing and implementing a real world grasp manipulation setup. Furthermore, the simulations uplift flexibility in improvising the simulated grasp manipulation to different types of robots.

The simulation at this phase of work runs as several individual concurrent scripts executed in different python virtual environments. The reason is the lack of support for different python versions for ROS packages. The gazebo environment lacks support for python3 where as modern machine learning python libraries only supports python3. However, the open-source development is continuous and compatibility issues can expected to be solved in future. The computing requirements for the implementation is high due to multiple concurrent running processes. There is ample of room for improvement in this work such as optimizing the performance. Simulation provides the support to spawn the objects in scene, but a real-world application requires human intervention. The assumptions made on the coefficient of friction, camera noise and differences between the simulated torque sensor and real torque sensor has to be noted. Therefore, the results in simulation cannot be directly used to predict the performance in real-world scenario.

5.5 Deliverable work and future improvements

At the current status of the thesis, PVN3D along with data generation step can be delivered as an end-to-end pose estimation network. The data generation pipeline can serve many of the existing research for their needs in acquiring data. The simulated method can be adapted to work with a real RGBD sensor and a robot upon further optimizations. The grasp manipulation step can be extended to a robotic assembly with quality control embedded in the system.

6. CONCLUSION

It can be concluded that all of the intended goals of the thesis work has been met and achievements can be summarized as follows;

- The objective of surveying on pose estimation methods and their adaptation towards grasp estimation was achieved via the literature study. Pose estimation was identified as the key task in of model-based grasp detection. Point-wise voting-based pose estimation methods highlights among 2D learning methods due to the robustness to occlusions and lack of texture.
- The second objective of analyzing class-agnostic, multi-object grasp detection was achieved via studying the literature on model-free grasp detection methods. The application of deep generation towards grasp detection proved to be a key factor in extending grasp detection towards novel objects.
- A dataset of 3152 images has been created using the proposed synthetic data generation method. The implementation of Grasp detection algorithms on synthetic data yielded successful results in simulation. The process of synthetic data generation can be concluded as a complex but more efficient method of generating training data. Gazebo simulation environment which was utilized for data generation can act as a multi-tasking tool in data generation, testing and grasp simulation steps. But the lack of support with machine learning python libraries is a downside at current moment.
- In achieving the forth objective, A comparative study of the two state of the art grasp detection methods exposing the structural differences was done. PVN3D is the most complex and lengthy implementation at development stage. It requires multiple models to be trained in order perform pose estimation on different objects. Model-based approaches has to perform the additional task of predefining grasps which is an unfavourable characteristic in mass production environments. But the quality of predefined grasps are high compared to model-free approach.Model-based approach is less flexible to alterations in the dataset. It does not facilitate pose estimation to familiar objects. Therefore model-based approach can be recommended for precision grasp manipulation tasks which allow minimum room for errors and less number of trials. GraspNet was easy to implement compared to model-based approaches. The implementation is short and simple. But it comes

with an additional step to filter grasps unfavorable for the grasping scenario. As a generative method, model-free approach can work with only a partial view of the object. Although, model-based method tolerates the occlusions in a similar way,its performance relies on the instance segmentation performance which identifies the object class. Hence, model-free approach is more effective on objects in clutter. Due to large number of grasps generated, model-free methods are more suitable for applications where the number of grasp trial attempts do not matter.

- The benchmarking of grasp manipulation was attempted via a simulation setup. A grasp-success benchmarking setup was initiated in gazebo physics simulation environment.The lack of support between machine learning libraries and ROS complicates the setup. However, the grasp-success tests emphasized the advantages and disadvantages between the two approaches studied.

REFERENCES

[1] Sahbani, A., El-Khoury, S. and Bidaud, P. An overview of 3D object grasp synthesis algorithms. *Robotics and autonomous systems* 60.3 (2012), pp. 326–336.

[2] Kleeberger, K., Bormann, R., Kraus, W. and Huber, M. A Survey on Learning-Based Robotic Grasping. *KleebergerKilian2020ASoL* (2020).

[3] Sahbani, A., El-Khoury, S. and Bidaud, P. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *The Artificial intelligence review* 54.3 (2012), pp. 1677–1734.

[4] Caldera, S., Rassau, A. and Chai, D. Review of Deep Learning Methods in Robotic Grasp Detection. eng. *Multimodal technologies and interaction* 2.3 (2018), p. 57.

[5] Pinto, L. and Gupta, A. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. *IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 3406–3413.

[6] Viereck, U., Pas, A. t., Saenko, K. and Platt, R. Learning a visuomotor controller for real world robotic grasping using simulated depth images. (2017).

[7] Mahler, J. and Goldberg, K. Conference on robot learning. (2017), pp. 515–524.

[8] Jiang, Y., Moseson, S. and Saxena, A. Efficient grasping from rgbd images: Learning using a new rectangle representation. *2011 IEEE International conference on robotics and automation* (2011), pp. 3304–3311.

[9] Redmon, J. and Angelova, A. Real-Time Grasp Detection Using Convolutional Neural Networks. (2014).

[10] Sundermeyer, M., Mousavian, A., Triebel, R. and Fox, D. Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes. eng. (2021).

[11] Liang, H., Ma, X., Li, S., Görner, M., Tang, S., Fang, B., Sun, F. and Zhang, J. PointNetGPD: Detecting Grasp Configurations from Point Sets. (2018).

[12] Shimoga, K. Robot Grasp Synthesis Algorithms: A Survey. *The International journal of robotics research* 15.3 (1996), pp. 230–266. ISSN: 0278-3649.

[13] Howard, W. and Kumar, V. On the stability of grasped objects. *IEEE transactions on robotics and automation* 12.6 (1996), pp. 904–917. ISSN: 1042-296X.

[14] Bicchi, A. and Kumar, V. Robotic grasping and contact: a review. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* 1 (2000), pp. 348–353.

[15] Roa, M. and Suarez, R. Computation of Independent Contact Regions for Grasping 3-D Objects. *IEEE transactions on robotics* 25.4 (2009), pp. 839–850. ISSN: 1552-3098.

[16] Rodriguez, A., Mason, M. T. and Ferry, S. From caging to grasping. *The International journal of robotics research* 31.7 (2012), pp. 886–900.

[17] Nguyen, V.-D. Constructing Force- Closure Grasps. *The International journal of robotics research* 7.3 (1988), pp. 3–16. ISSN: 0278-3649.

[18] Kirkpatrick, D., Mishra, B. and Yap, C.-K. Quantitative steinitz's theorems with applications to multifingered grasping. *Discrete & Computational Geometry* 7.3 (1992), pp. 295–318.

[19] MILLER, A. T. and ALLEN, P. K. Graspit. *IEEE robotics and automation magazine* 11.4 (2004), pp. 110–122.

[20] Miller, A., Knoop, S., Christensen, H. and Allen, P. Automatic grasp planning using shape primitives. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)* 2 (2003), pp. 1824–1829.

[21] Pelossof, R., Miller, A., Allen, P. and Jebara, T. An SVM learning approach to robotic grasping. eng. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004* 4 (2004), pp. 3512–3518. ISSN: 1050-4729.

[22] Borst, C., Fischer, M. and Hirzinger, G. Grasping the dice by dicing the grasp. Vol. 4. IEEE, 2003, pp. 3692–3697. ISBN: 0780378601.

[23] Ferrari, C. and Canny, J. Planning optimal grasps. *Proceedings 1992 IEEE International Conference on Robotics and Automation* 3 (1992), pp. 2290–2295.

[24] Weisz, J. and Allen, P. K. Pose error robust grasping from contact wrench space metrics. (2012), pp. 557–562.

[25] Saxena, A., Driemeyer, J. and Ng, A. Y. Robotic Grasping of Novel Objects using Vision. *The International journal of robotics research* 27.2 (2008), pp. 157–173.

[26] Bohg, J., Morales, A., Asfour, T. and Kragic, D. Data-Driven Grasp Synthesis-A Survey. *IEEE transactions on robotics* 30.2 (2014), pp. 289–309. ISSN: 1552-3098.

[27] He, K., Gkioxari, G., Dollar, P. and Girshick, R. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988. ISSN: 2380-7504.

[28] Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. eng. *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 0162-8828.

[29] Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. Feature Pyramid Networks for Object Detection. eng. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 936–944. ISSN: 1063-6919.

[30] Lepetit, V. and Fua, P. *Monocular model-based 3D tracking of rigid objects: a survey.* eng. Vol. 1. Foundations and trends in computer graphics and vision 1. Hanover, Mass: Now Publishers, 2005, pp. 1–89. ISBN: 1933019530.

[31] Lowe, D. Object recognition from local scale-invariant features. eng. *Proceedings of the Seventh IEEE International Conference on Computer Vision* 2 (1999), pp. 1150–1157.

[32] Bay, H., Tuytelaars, T. and Van Gool, L. SURF: Speeded Up Robust Features. eng. *Computer Vision – ECCV 2006*. Lecture Notes in Computer Science 3951 (2006), pp. 404–417. ISSN: 0302-9743.

[33] Mur-Artal, R., Montiel, J. M. M. and Tardos, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. eng. *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163. ISSN: 1552-3098.

[34] Rusu, R. B., Blodow, N. and Beetz, M. Fast Point Feature Histograms (FPFH) for 3D registration. eng. (2009), pp. 3212–3217. ISSN: 1050-4729.

[35] Salti, S., Tombari, F. and Di Stefano, L. SHOT: Unique signatures of histograms for surface and texture description. eng. *Computer vision and image understanding* 125 (2014), pp. 251–264. ISSN: 1077-3142.

[36] Pham, Q.-H., Uy, M. A., Hua, B.-S., Nguyen, D. T., Roig, G. and Yeung, S.-K. LCD: Learned Cross-Domain Descriptors for 2D-3D Matching. eng. *Proceedings of the ... AAAI Conference on Artificial Intelligence* 34.7 (2020), pp. 11856–11864. ISSN: 2159-5399.

[37] Yi, K. M., Trulls, E., Lepetit, V. and Fua, P. LIFT: Learned Invariant Feature Transform. eng. *Computer Vision – ECCV 2016*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 467–483. ISBN: 3319464655.

[38] Hu, Y., Hugonot, J., Fua, P. and Salzmann, M. Segmentation-driven 6D Object Pose Estimation. eng. (2018).

[39] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K. and Navab, N. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. eng. *Computer Vision – ACCV 2012*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 548–562. ISBN: 3642373305.

[40] Hodaň, T., Michel, F., Brachmann, E., Kehl, W., Buch, A. G., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., Sahin, C., Manhardt, F., Tombari, F., Kim, T.-K., Matas, J. and Rother, C. BOP: Benchmark for 6D Object Pose Estimation. eng. *Computer Vision – ECCV 2018*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 19–35. ISBN: 3030012484.

[41] Gordo, A., Almazán, J., Revaud, J. and Larlus, D. Deep Image Retrieval: Learning Global Representations for Image Search. eng. *Computer Vision – ECCV 2016*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 241–257. ISBN: 3319464655.

[42] Xiang, Y., Schmidt, T., Narayanan, V. and Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. eng. (2017).

[43] Capellen, C., Schwarz, M. and Behnke, S. ConvPoseCNN: Dense Convolutional 6D Object Pose Estimation. eng. (2019).

[44] Song, C., Song, J. and Huang, Q. HybridPose: 6D Object Pose Estimation under Hybrid Representations. eng. (2020).

[45] Park, K., Mousavian, A., Xiang, Y. and Fox, D. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. 2020, pp. 10710–10719.

[46] Peng, S., Zhou, X., Liu, Y., Lin, H., Huang, Q. and Bao, H. PVNet: Pixel-wise Voting Network for 6DoF Object Pose Estimation. eng. *IEEE transactions on pattern analysis and machine intelligence* PP (2020), pp. 1–1. ISSN: 0162-8828.

[47] He, Y., Sun, W., Huang, H., Liu, J., Fan, H. and Sun, J. PVN3D: A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation. eng. (2019).

[48] Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J. and Rother, C. Learning 6D Object Pose Estimation Using 3D Object Coordinates. eng. *Computer Vision – ECCV 2014*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 536–551. ISBN: 9783319106045.

[49] Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L. and Savarese, S. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. eng. (2019).

[50] Wada, K., Sucar, E., James, S., Lenton, D. and Davison, A. J. MoreFusion: Multi-object Reasoning for 6D Pose Estimation from Volumetric Fusion. eng. (2020).

[51] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J. and Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. eng. *The International journal of robotics research* 37.4-5 (2018), pp. 421–436. ISSN: 0278-3649.

[52] *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. eng. Information Science and Statistics. New York, NY: Springer New York, 2004. ISBN: 9781475743227.

[53] Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards. eng. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1957–1964. ISBN: 1467380261.

[54] Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A. and Goldberg, K. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. eng. (2017).

[55] You Only Look Once: Unified, Real-Time Object Detection. eng. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 779–788. ISBN: 9781467388511.

[56] Redmon, J. and Farhadi, A. YOLO9000: Better, Faster, Stronger. eng. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 6517–6525. ISBN: 1538604574.

[57] Redmon, J. and Angelova, A. Real-Time Grasp Detection Using Convolutional Neural Networks. eng. (2014).

[58] Xu, D., Anguelov, D. and Jain, A. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. eng. (2017).

[59] Qi, C. R., Liu, W., Wu, C., Su, H. and Guibas, L. J. Frustum PointNets for 3D Object Detection from RGB-D Data. eng. (2017).

[60] Votenet. eng. *Association management* 52.13 (2000), pp. M38–. ISSN: 0004-5578.

[61] Fang, H.-S., Wang, C., Gou, M. and Lu, C. GraspNet: A Large-Scale Clustered and Densely Annotated Dataset for Object Grasping. eng. (2019).

[62] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. and Bennamoun, M. Deep Learning for 3D Point Clouds: A Survey. eng. *IEEE transactions on pattern analysis and machine intelligence* PP (2020), pp. 1–1. ISSN: 0162-8828.

[63] Shi, W., Ragunathan and Rajkumar. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. eng. (2020).

[64] Wang, Y., Sun, Y., Liu, Z., Sarma, S., Bronstein, M. and Solomon, J. Dynamic Graph CNN for Learning on Point Clouds. eng. *ACM transactions on graphics* 38.5 (2019), pp. 1–12. ISSN: 0730-0301.

[65] Charles, R. Q., Su, H., Kaichun, M. and Guibas, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. eng. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 77–85. ISBN: 1538604574.

[66] Qi, C. R., Yi, L., Su, H. and Guibas, L. J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. eng. (2017).

[67] Joseph-Rivlin, M., Zvirin, A. and Kimmel, R. Momen(e)t: Flavor the Moments in Learning to Classify Shapes. eng. (2018).

[68] Zhao, H., Jiang, L., Fu, C.-W. and Jia, J. PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5560–5568. DOI: 10.1109/CVPR.2019.00571.

[69] Zeng, A., Song, S., Niessner, M., Fisher, M., Xiao, J. and Funkhouser, T. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. eng. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 199–208. ISBN: 1538604574.

[70] Mellado, N., Aiger, D. and Mitra, N. J. Super 4PCS Fast Global Pointcloud Registration via Smart Indexing. eng. *Computer graphics forum* 33.5 (2014), pp. 205–215. ISSN: 0167-7055.

[71] Yang, J., Li, H., Campbell, D. and Jia, Y. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. eng. *IEEE transactions on pattern analysis and machine intelligence* 38.11 (2016), pp. 2241–2254. ISSN: 0162-8828.

[72] Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R. A., Lucey, S. and Choset, H. PCRNet: Point Cloud Registration Network using PointNet Encoding. eng. (2019).

[73] Pas, A. ten, Gualtieri, M., Saenko, K. and Platt, R. Grasp Pose Detection in Point Clouds. eng. *The International journal of robotics research* 36.13-14 (2017), pp. 1455–1473. ISSN: 0278-3649.

[74] Zapata-Impata, B. S., Gil, P., Pomares, J. and Torres, F. Fast geometry-based computation of grasping points on three-dimensional point clouds. eng. *International journal of advanced robotic systems* 16.1 (2019), pp. 172988141983184–. ISSN: 1729-8814.

[75] *Matter and Form*. 2015. URL: `https://mafstore.blob.core.windows.net/media/5501a796497fac0021d82386.png`.

[76] Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J. Pyramid Scene Parsing Network. eng. (2016).

[77] Odaibo, S. G. *Variational Inference and Derivation of the Variational Autoencoder (VAE) Loss Function: A True Story*. 2020. URL: `https://medium.com/retina-ai-health-inc/variational-inference-derivation-of-the-variational-autoencoder-vae-loss-function-a-true-story-3543a3dc67ee`.

[78] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L. and Yu, F. ShapeNet: An Information-Rich 3D Model Repository. eng. (2015).

[79] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K. and Navab, N. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. eng. *Computer Vision – ACCV 2012*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 548–562. ISBN: 3642373305.