Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Learning Grasp Evaluation Models Using Synthetic 3D Object-Grasp Representations

Minh Nguyen

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Alex Mitrevski
Maximilian Schöbel

August 2018

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Minh Nguyen

Abstract

This project considers the problem of generating data for training grasp evaluation models. Recent advances are reviewed for four main aspects most relevant to labeled grasp data synthesis, namely feature extraction from perceptual data, object-grasp representation, grasp evaluation techniques, and data generation techniques. A completed object grasping pipeline is integrated, from object detection to grasp pose detection and grasp execution. Two set of experiments are performed for two pose estimation methods using this grasping pipeline.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivation

Manipulation is an important functionality for applied robotics. Manipulating objects involves moving an end effector to the desired object's location and using this end effector to interact with this object to execute a specific task. The task can be simply moving the object to another location or can require more complicated maneuvers of the object (i.e. moving a handle to open doors or moving a bottle to pour water). Traditionally the end effector in the industry is a two-fingered gripper. Any object movement, in this case, requires manipulating the whole arm, which is effective for transferring objects over large motion ranges but can be intractable for tasks requiring more precise movements. The recent vacuum based effectors face similar problems in object manipulation. To handle finer motions, the common solutions in the industry are either using custom designed end-effector or multi-finger robot hands. The problem of robotic grasping typically refers to the use of grippers and multi-finger end effectors to manipulate objects. 70 years after the conception of the first robot arm, a general solution to finding a suitable grasp for objects is still a challenging problem and an area of active research.

Figure 1.1: Typical objects in the Robocup@Home competition [24].

1.2 Use Case

This project focuses on the use cases of robotic grasping in the Robocup @Home [1] league. The tasks in this competition are designed to evaluate the performance of autonomous robots in service robotic applications in a domestic environment. The objects to be manipulated are ones that can be found in a typical household setting. Some examples of such objects can be seen in figure 1.1. These objects can be placed on tables, shelves or in a dishwasher, whose locations can be anywhere in a simulated apartment. Grasping tasks in such an environment is challenging because of various factors, including the variety of object shapes, the uncertainty of the robot manipulator and objects' locations, and the many noise/disturbance sources which can interfere with the robot's perceptual sensors such as lighting condition or object occlusion.

The platform used for the project's experiments is the robot chosen for the

[1] http://www.robocupathome.org/

standard platform for the Robocup@Home league, the Toyota Human Support Robot (HSR) [1] [24]. The robot is equipped with an RGB-D sensor (Xtion PRO LIVE [2]), stereo camera and a wide-angle camera on the robot head, as well as a wide angle camera mounted on the two-finger gripper. The gripper also has a potentiometer and a force sensor.

1.3 An overview of robotic grasping research

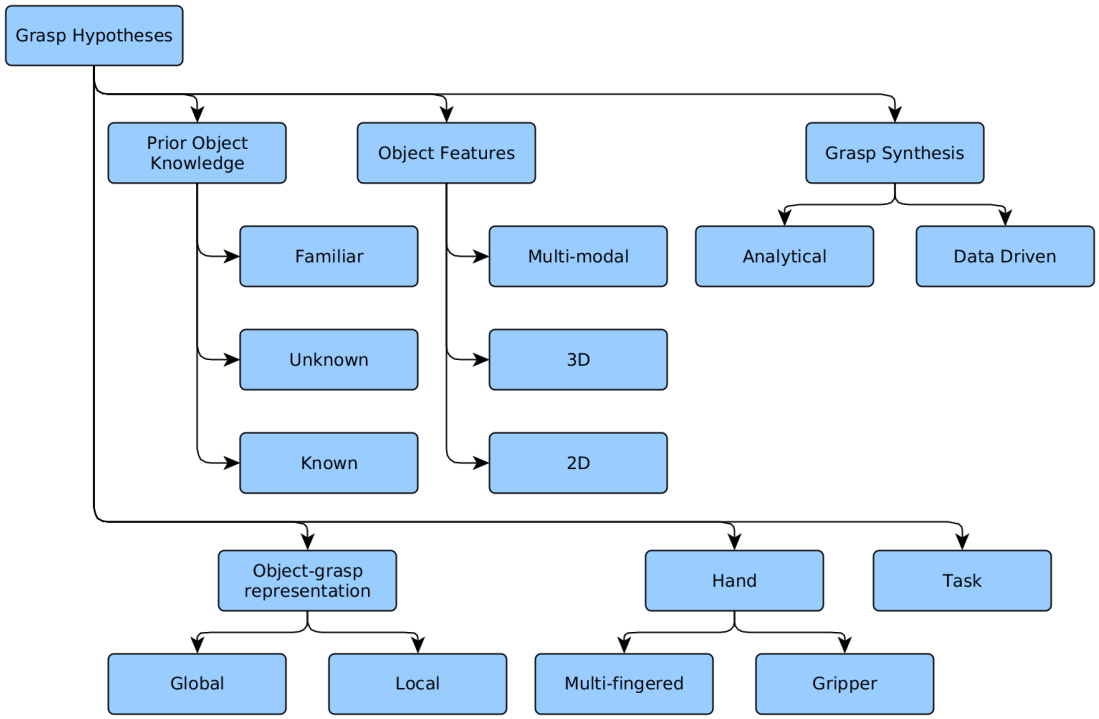


Figure 1.2: Aspects which may influence generation of grasp hypotheses [3].

Synthesizing an optimal grasp from perceptual data is a challenging problem and is frequently addressed in robotic research, resulting in a myriad of approaches. Sahbani et al. [31] classify grasp synthesis into analytical and empirical approaches. Analytical methods consider mechanical properties of the contact points between the gripper's fingers and an object [29, 31, 33], namely:

[1] https://www.toyota-global.com/innovation/partner_robot/robot/#link02
[2] https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/

- *Disturbance resistance*: the ability of the grasp to resist disturbances when the object is immobile, either by form-closure (finger positions) or force-closure (forces applied by fingers).
- *Dexterity*: the ability of the robot hand to move the object to perform a specified task, or in any direction if no task is specified.
- *Equilibrium*: the combined forces and torques applied to the object is null.
- *Stability*: any displacement of an object caused by a disturbance will self-correct over time.

Instead of analyzing mechanical properties, empirical approaches rely on some form of grasp experience to synthesize candidates. Sahbani et al. [31] group empirical approaches based on whether they focus on observing humans or objects. The first group of methods teaches a robotic system to observe a human operator via different forms of descriptors, then to reproduce the same grasp. These techniques are also known as learning by demonstration. Methods focusing on object observation generally learn the association between object characteristics and gripper configurations. The survey by Bohg et al. [3] argue instead to classify data-driven methods based on how much information is assumed about the query object, specifically:

- Approaches dealing with *known objects* rely on a grasp experience database which contains suitable grasps for each encountered object. These approaches, therefore, focus on object recognition and pose estimation.
- Approaches for *familiar objects* focus on finding a similarity metric and an object representation, from low (i.e. shape, color, texture) to high (i.e. category) levels, to match query objects with encountered objects.
- Approaches for *unknown objects* identify local or global features from sensory data to for generating and evaluating grasp candidates.

The authors argue that this classification can better capture the diversity of empirical approaches as well as the importance of perception in the process. In addition to the above classification, Bohg et al. [3] also identify principal factors that may influence a grasp hypothesis, as shown in the mind map in figure 1.2.

Analytical approaches to synthesizing grasps generally rely on knowledge of the object, the gripper model and the contact location of the fingers. This information is often inaccurate or unavailable in the real environment because of noisy sensors as well as imprecise manipulator/gripper actuation. Indeed, analytical methods have been shown to be unreliable in synthesizing stable grasps when applied on real robots [19, 30, 37]. Empirical approaches, however, as with other supervised machine learning methods, demand labeled data. Generating a sufficiently large grasp experience knowledge base is often time-consuming and expensive, while grasp simulation may not be close enough to the real world.

Data augmentation, the process of generating new samples by transforming real data, and data synthesis are popular solutions to supplementing limited training data [9, 34], especially when applying deep learning methods to solve image recognition and detection tasks. As grasp synthesis algorithms need 3D information synthesizing the gripper configuration, RGB-D is often preferred over images as training data. The research in [8, 16] proposes approaches to augmenting RGB-D data, whereas Mahler et al. [23] generates synthetic data to train a grasp quality predictor.

In order to leverage the successes of deep learning for various perception and grasping tasks, which are in general supervised techniques, this project shall focus on data generation and augmentation for empirical approaches to grasp synthesis using labeled data.

The next chapter of this report will examine recent advances in aspects relevant to generating data for a grasp quality model. Next, the methodology of the grasping pipeline's components implemented will be described in chapter 3. Chapter 4 will describe the experiments performed using this pipeline, and chapter 5 will conclude by discussing the contributions and potential future extensions of the project.

2

State of the Art

Empirical grasp synthesis techniques based on labeled data vary mainly in what type of perceptual data is taken into consideration, how object-grasp representation is formulated based on such perceptual data, and how grasp candidates are evaluated. This section will review recent approaches to handling these topics, as well as recent data augmentation and synthesis methods for training a grasp evaluation model.

2.1 Extracting features from perceptual data for grasping

As RGB-D cameras become more accessible and affordable, they are often chosen in robotic systems as the primary perception sensor. While RGB-D data has been shown to provide richer features compared to pure 2D images for various perception tasks [20, 8, 16, 18], it is often unclear how to handle the multi-modality of color and depth information.

Bo et al. [1] build sparse coding dictionaries for RGB-D data using the K-SVD algorithm and proposed to use a hierarchical matching pursuit (HMP) algorithm to compute a feature hierarchy for new RGB-D images. Each entry in the dictionaries contain 8 channels calculated from RGB-D data: grayscale intensity, RGB, depth and surface normals in three axes.

Lenz et al. [20] use deep auto-encoders to build a representation for each feature channel, reducing its data dimensions. To handle the multi-modality of RGB and depth data, the authors also introduced a structured regularization technique. Each modality (i.e. RGB and depth/surface normal) has a regularization term which

will be added to each hidden unit. In case of a p-norm, with K hidden units, R modalities, and N visible units, the regularization term would be

$$f(W) = \sum_{j=1}^{K} \sum_{r=1}^{R} \left(\sum_{i=1}^{N} S_{r,i} |W_{i,j}^{p}| \right)^{1/p} \tag{2.1}$$

, where $S_{r,i}$ is 1 if feature i belongs to modality r and 0 otherwise.

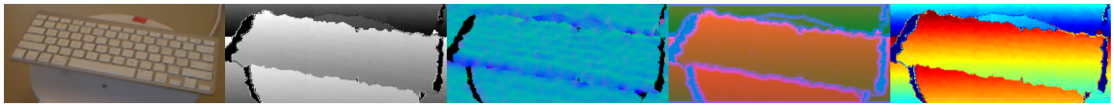


Figure 2.1: Different encodings of depth information [8]. From left: RGB, grayscale depth, surface normals [1], HHA [16], and the color mapping of depth values proposed by Eitel et al. [8].

Building upon the success of Convolutional Neural Networks (CNN) in capturing 2D features [14], several multi-modal approaches convert depth data into three-channel images during the preprocessing step [8, 16]. Gupta et al. [16] propose the HHA representation, which encodes the depth each pixel of the depth image with horizontal disparity, height above ground, and the angle between the local surface normal and direction of gravity. Eitel et al. [8] propose to use convert the depth value directly into RGB values using a jet color mapping. Examples of the different depth encodings can be seen in figure 2.1. Two CNN models of the same architecture are then trained independently using depth and RGB images to classify objects. Finally, features learned from the two models (the networks up until before the classification layer) are concatenated as inputs to a fully connected layer which learns to combine them to recognize objects. The paper showed that while surface normals proved to give better classification performance using depth information alone, they require additional calculation on the input data and does not improve on results when combining both RGB and depth images compared to the proposed color map encoding.

More recently, Porzi et al. [27] introduce a novel convolutional block called *DaConv* to learn scale awareness using depth information. Each *DaConv* block consists of two networks, namely *PredictionNet* and *DepthNet*. In *PredictionNet*

different scales of the input features are simulated using dilated convolutions [39] with different dilation factors $\{\ell_1, ..., \ell_d\}$. An ℓ-dilated convolution can be obtained by adding $\ell - 1$ zeros between adjacent elements of the convolutional kernel. *Depth-Net* use only the depth information to produce a vector of probabilities $\{a_1, ..., a_d\}$ representing how likely a dilation factor is chosen for a specific spatial location, captured by each convolutional kernel. The outputs of the dilated convolutions are then linearly combined using this probability vector to produce the final output of the *DaConv* block. The authors evaluate their networks on three robot perception tasks and achieve state-of-the-art performance. The three perception tasks examined are object pose estimation, object affordance detection, and contour detection.

Above methods deal with data acquired from a single RGB-D sensor, which does not provide complete 3D information, as occluded regions are not included in point clouds. Techniques generating representations for object models, however, work with true 3D information and can give insights into extracting features from a single view RGB-D camera. Recent techniques for generating representations of 3D object models generally use either volumetric CNN's (where the kernels have three dimensions instead of two like regular CNN's) or applying well known CNN models to 2D views rendered from the 3D model. Qi et al. [28] develop two novel volumetric architectures, which significantly improve previous volumetric CNN results. The first approach introduces auxiliary learning tasks in the final layers, training parts of the network only on sub-volumes of the full objects. The second approach introduces anisotropic probing kernels to project the 3D model onto 2D views, then applies a 2D CNN model on the projections. The authors also extend and improve upon a highly successful multi-view CNN technique proposed by Su et al. [35] by introducing multi- resolution filters to capture information at multiple scales. These techniques may allow sampling occluded points in a single-view RGB-D image, which would provide more information for planning grasps.

Efforts have also been spent to reconstruct objects from the incomplete view of a single RGB-D camera. Bohg et al. [2] search for a symmetric plane perpendicular to workspace surface (i.e. table) and mirror the incomplete point cloud across this plane to fill in the missing information. Varley et al. [36] train a CNN to take in an occupancy grid computed from a 2.5D point cloud and output an occupancy

grid of the same shape which predicts if the obscured points are occupied. This output voxel grid is either directly run through a marching cubes algorithm to quickly create an object mesh and add it to the manipulation planning scene, or post-processing is applied to create a higher resolution mesh for grasp planning.

2.2 Object-Grasp representation

Bohg et al. [3] parameterize grasps by (1) the *grasping point* of the object where the end-effector should be aligned, (2) an *approach vector* from which the robot hand shall approach the *grasping point*, (3) the *wrist orientation* of the hand, and (4) an *initial finger configuration*. The article also categorized object-grasp representation approaches into ones which extract local (i.e. curvature, contact area with the hand) or global features (center of mass, bounding box).

2.2.1 Techniques based on global features

As illustrated in figure 2.2, Mahler et al. [23] represent grasps in 2D images by aligning the image center to the gripper central point and the image's middle row to the grasp axis. The grasp is assumed to have an approach vector perpendicular to the table and hence is characterized by the gripper center and the angle of the gripper axis with respect to the table.

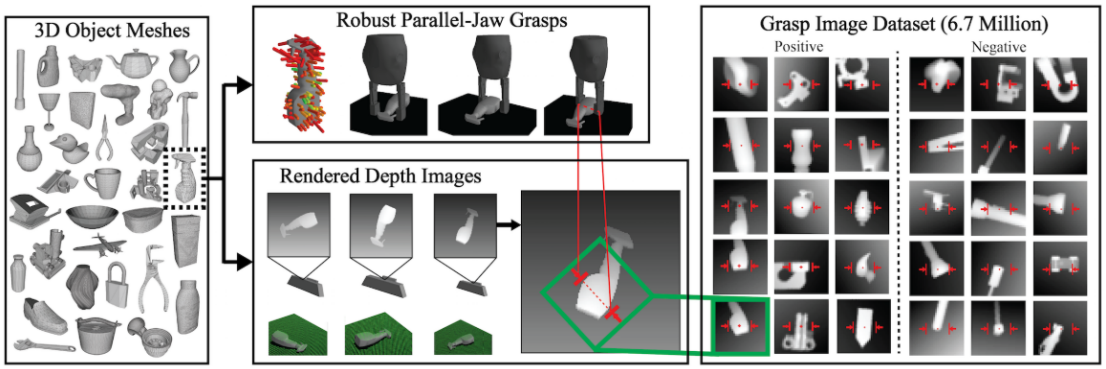


Figure 2.2: Dex-Net 2.0 data generation pipeline [23].

Several methods use eigengrasps to represent the object-grasp relation in free space [12, 4]. Ciocarlie and Allen [4] introduced eigengrasps and defined them

as principal components of the dataset of human hand configurations. Using this low-dimensional representation, the robot hand receive adjustments from a human operator during grasp execution. The proposed Eigengrasp planner uses a simulated annealing algorithm to maximize a quality metric Q calculated from the robot hand's posture and pose, where posture is represented using an eigengrasp. Specifically,

$$Q = \Sigma_i(1 - \delta_i) \tag{2.2}$$

for all desired contacts i, where

$$\delta_i = \frac{|\mathbf{o_i}|}{\alpha} + \left(1 - \frac{\hat{\mathbf{n}}_\mathbf{i} \cdot \mathbf{o_i}}{|\mathbf{o_i}|}\right) \tag{2.3}$$

, in which the local surface normal $\hat{\mathbf{n}}_\mathbf{i}$ and the distance between the desired contact location and the object $\mathbf{o_i}$ is related to the eigengrasp and hand position. α is a scaling factor which make the two terms comparable in equation 2.3. Goldfeder and Allen [12] used the Eigengrasp planner to generate the Columbia Grasp Database [1]. Grasp planning is then done by matching the query object to one in the database, then executing the best indexed grasp for the matched object.

2.2.2 Techniques based on local features

Detry et al. [5] attempt to apply grasp experience demonstrated on known objects to novel ones by finding similar graspable regions. As illustrated in figure 2.3, part candidates are point clouds extracted from applying a set of predefined regions of interest (ROI) boxes on a known grasp from the database. Then the authors defined a distance metric for clustering similar candidate parts, and a dictionary of candidate parts is created using only the central points of these clusters. Part candidates for new object are then matched against this dictionary to find possible grasps for the familiar object part.

Several methods represent grasp candidates as rectangles positioned in RGB and/or depth images [20, 18], corresponding with a bipedal gripper. The RGB-D image region within these rectangles is then used to score the respective grasp using

[1] http://grasping.cs.columbia.edu

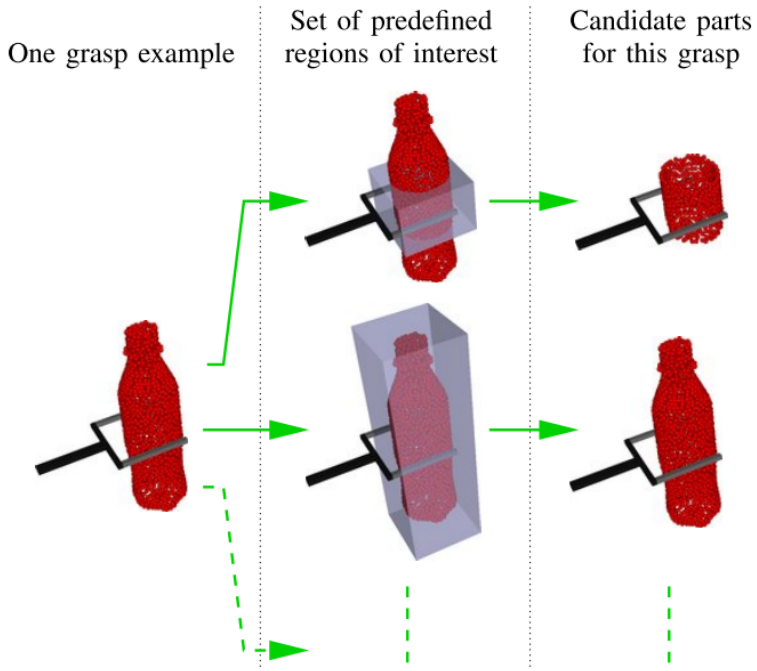


Figure 2.3: Generating part candidates with box-shaped region of interests [5].

machine learning methods. These approaches are naturally limited in their ability to represent the robot hand's approach vector since rectangles in images can only correspond to grasps roughly parallel with the RGB-D camera's optical axis [15].

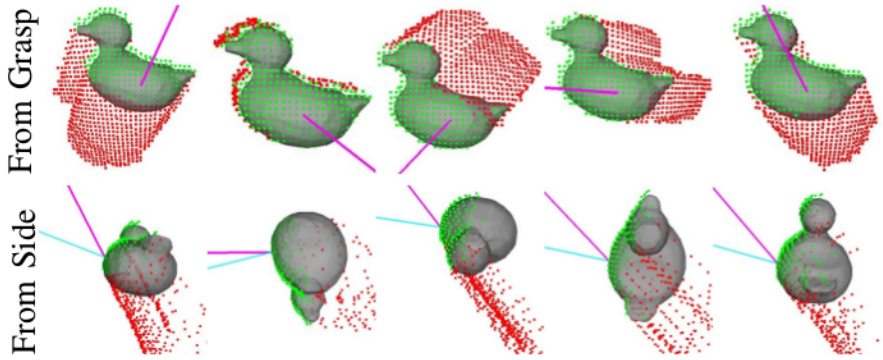


Figure 2.4: Local shape representations with different camera viewpoints of the same grasp. Cyan lines represent the approach vectors while pink lines represent the camera viewpoint [19].

Kappler et al. [19] extract local shape representations (or templates) of objects from point clouds generated from object meshes at various camera viewpoints. The templates are grids with predefined resolution aligned with the plane tangent to the object surface at the intersection point between the grasp approach vector and the object. Each point cloud will be projected onto the grid, where the grid cells with corresponding projected points "contain the distance to that point as a value," and the other grid cells are considered either occlusion (with height depends on the camera angle) or free space (with a fixed height). Figure 2.4 illustrates how the local shape representations capture the different camera viewpoints.

Gualtieri et al [15] propose to represent a grasp candidate by the cuboid swept out by a two-fingered gripper as it closes on the object, extracting the observed points from RGB-D clouds and sampling points unobserved by all sensors within the region. In order to reduce dimensionality, the points in this cuboid region are then voxelized into $60 \times 60 \times 60$ grid, then projected onto the three planes corresponding with the cuboid's principal axes. For each projection, three images are generated: the average height maps of the occupied points (60×60) and unobserved region (60×60), as well as the average surface normals ($60 \times 60 \times 3$), whose three dimensions are interpreted as three channels in the image. These result in a total of 15 channels representing each candidate grasp. In their experiments, this approach performs better than the above approach proposed by Kappler et al [19], but does not improve significantly over their previous approach, which used only three channels from the surface normals for each projection. This approach did not consider RGB data, arguing that the improvement would be insignificant considering the results from [20].

2.3 Grasp evaluation

2.3.1 Background and terminology

Many classical grasp evaluation approaches follow Murray et al [25]'s definition of grasp planning as the problem of finding a set of contact locations between a rigid-body object and the end-effector's fingers, mounted on a rigid-link robot. The model of contact between the object's surface and robot fingertips affects

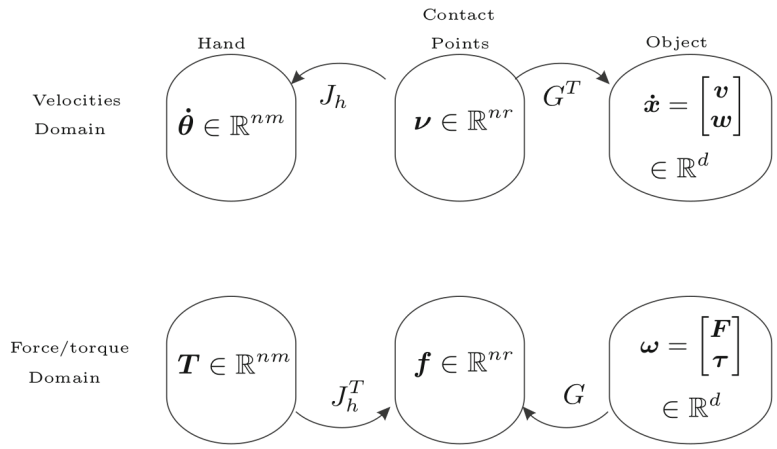


Figure 2.5: Relationships between velocities and forces, torques for multi-fingered grasping [29].

the mechanical analysis of grasps. Contact types are generally grouped into three categories:

- *Frictionless point contact*: applied forces are assumed to always be normal to the object's surface.
- *Point contact with friction*: applied forces are assumed to have a normal and a tangential component, the latter representing friction between the fingers and the object surface. This contact is commonly modeled by the Coulomb's friction cone.
- *Soft contact*: in addition to the forces in the friction cone described in the previous model, torques around the object's surface normal are also taken into consideration.

In their survey of grasp metrics, Roa and Suárez [29] adopt the following notations for concepts defined in [25]. Force $\boldsymbol{F}_i$ applied on the object at contact point $\boldsymbol{p}_i$ generate a torque $\boldsymbol{\tau}_i$, and are modeled by a wrench vector $\boldsymbol{\omega}_i = (\boldsymbol{F}_i, \boldsymbol{\tau}_i/\rho)^T$, in which ρ is a constant that defines the metrics of the wrench space. Contact type dictates the number of independent components r of $\boldsymbol{\omega}_i$: $r = 1$ for frictionless contacts, $r = 2$ or 3 for contacts with friction for 2D and 3D objects, and $r = 4$ for soft contacts.

The object's movement is modeled by a twist $\dot{\boldsymbol{x}} = (\boldsymbol{v}, \boldsymbol{w})^T$, where $\boldsymbol{v}$ is the translational velocity of the object's center of mass (CM), and $\boldsymbol{w}$ is the object's rotational velocity around its CM.

The robot hand is assumed to have n fingers, each with m joints. The force $\boldsymbol{f}_i$ applied at fingertip i, $i = 1, ..., n$, is caused by the torques in each joint $\boldsymbol{T}_{ij}$, $j = 1, ..., m$. $\boldsymbol{\theta}_{ij}$ denotes the velocities in these finger joints. Vectors $\boldsymbol{f}$ and $\boldsymbol{\nu}$ denote the forces and velocities at the fingertips after grouping all the r independent components together.

The relations between variables at the finger joints and at contact points are captured by the hand Jacobian J_h:

$$\boldsymbol{\nu} = J_h \dot{\boldsymbol{\theta}} \tag{2.4}$$

$$\boldsymbol{T} = J_h^T \boldsymbol{f} \tag{2.5}$$

, whereas the relationship between the object and the contact points are captured by the grasp matrix G:

$$\boldsymbol{\nu} = G^T \dot{\boldsymbol{x}} \tag{2.6}$$

$$\boldsymbol{\omega} = G \boldsymbol{f} \tag{2.7}$$

. These relationships can be seen visualized in figure 2.5. The fundamental grasping constraint can then be derived from equations (2.4) and (2.6):

$$J_h \dot{\boldsymbol{\theta}} = G^T \dot{\boldsymbol{x}} \tag{2.8}$$

. The hand-object Jacobian H represents the transformation between the high-dimensional hand joint space and the low-dimensional object space:

$$\dot{\boldsymbol{x}} = H \dot{\boldsymbol{\theta}} \tag{2.9}$$

, where $H = (G^T)^+ J_h$, and $(G^T)^+$ denotes the pseudoinverse of G^T.

Via approximating the friction cones at contact points $\boldsymbol{p}_i$ as pyramids, the applied wrench can be represented by a linear combination of primitive wrenches. The convex hull $\mathcal{P}$ containing all primitive wrenches is called the *Grasp Wrench*

Space (GWS).

2.3.2 Analytical grasp metrics

Roa and Suárez [29] group grasp quality measures into approaches focusing on contact point position, hand configuration, and ones which combine both metric types.

Contact point grasp quality measures focus on the object's properties, friction constraints, and form/force closure conditions. Here the space of all force-closure grasps is referred to as *force-closure space* (FCS). Several of these approaches analyze the algebraic properties of the grasp matrix G, most commonly optimizing some function of its singular values. Other approaches may measure grasp quality based on geometric relations between the robot hand and the object, such as shape or area of grasp polygon for 2D objects, distance between the contact polygon's centroid and the object's CM, how well the hand aligns with the object's principal axis, or how far away the finger positions are from the boundaries of the FCS. Another factor which may affect grasp quality is the limits of finger forces. Methods concerning this factor may consider the geometric properties of the GWS $\mathcal{P}$, or formulate the metrics directly from the applied forces and torques. Task-oriented measures may focus on analyzing wrenches within a *task wrench space* (TWS), also known as task polytope, which includes wrenches needed for a specific task to be executed or disturbance wrenches expected during task execution.

Approaches focusing on hand configuration measures focus on studying the properties of the hand-object Jacobian H, many of which apply a similar analysis of the algebraic properties of the grasp matrix G to H. Some approaches in this category also consider the physical limits of the end-effector fingers, ranking grasps based on how close each joint is to its operating range.

Grasp measures can be combined serially or in parallel to capture different aspects of a grasp's quality. In the serial case, one measure is first applied to find several good grasps, then another is used to choose the optimal candidate from these grasps. A parallel approach combines the different measures in a single global index, i.e. simply calculating a normalized sum of these measures or creating a ranking table for each quality criterion, then summing the rankings for each grasp.

Most popular among analytical grasp evaluation methods is the ϵ-metric. Ferrari and Canny [10] formulated the metric from the maximum external wrench that a grasp can resist in any direction. In wrench space $\mathcal{P}$ this value is represented by the distance from the space's origin to its convex hull. The metric, therefore, is geometrically characterized by the radius of the largest sphere that can be contained in $\mathcal{P}$ and centered at its origin [29]. Weisz and Allen [37] extend the ϵ-metric with probabilistic methods to better deal with uncertainties in object pose. The method assumes the object to be fixed on a table and limit their noise model to the 2D coordinates and an angle with respect to this table, $[x, y, \theta]$. Grasp simulations are run for all perturbations of the object pose within predefined error ranges for these three parameters. The extended grasp metric is defined as the probability of that the evaluated grasp achieves an ϵ score over a certain value, for all perturbations. Evaluation using this metric shows that grasps with the best ϵ score is much more likely to have low ϵ score when pose variations are introduced, as compared to grasps which perform well under the proposed metric. However, the authors also admit that the calculation of the metric is intractable in real time, taking hours in worst cases.

2.3.3 Learning to predict grasp quality

Table 2.1 lists recent prominent empirical approaches to predicting grasp quality. Earlier techniques often rely on carefully designed features. Jiang et al.'s [18] calculate spatial histograms for 17 different filters of each pixel in each RGB-D input image. These features are linearly combined using a weight matrix, creating a score value for each pixel. Three such weight matrices are initialized, creating three pixel-wise score matrices. For each bipedal grasp, the aforementioned rectangle representation is divided into three regions. For each region, the respective score values are extracted from one of the three score matrices. The final score for grasp is the sum of all the extracted pixel scores. A Support Vector Machine (SVM) algorithm is used to learn the weight matrices. Lenz et al [20] train auto-encoders to extract features from the rectangles representing grasps in RGB-D images. Weights from these auto-encoders are used to initialize multilayer-perceptron (MLP) models, which are then trained to predict the probability of success of given grasps.

Approach	Object-grasp representation	Feature extraction & learning method	Dataset generation
Jiang et al. [18]	Rectangle in RGB-D image	Histograms of hand-crafted filters of RGB-D images Model: SVM	Rectangles manually annotated on object images
Lenz et al. [20]	Rectangle in RGB-D image	Auto-encoders used to initialize weights, structured regularization used to combine depth and RGB data Model: MLP	Extended version of the dataset used in Jiang et al. [18]
Kappler et al. [19]	Template grid at intersection of object surface and approach vector	RGB rendering of the template grid Model: LeNet CNN	Grasps and quality values are generated for object meshes in simulation then verified by crowd-sourcing
Gualtieri et al. [15]	Rectangle cuboid region swept out by the gripper fingers, containing observed depth data and points sampled from the occluded volume	Different filters of the cuboid region are projected onto three orthogonal planes, creating 15 image channels Model: LeNet CNN	Grasp candidates sampled for object models are labeled using force-closure property
Mahler et al. [23]	Two gripper plates visualized in depth images	Depth images are cropped and aligned to the gripper Model: CNN combined with single layer neural networks	Grasp candidates sampled for object models are labeled using a variant of the robust ϵ-metric from [37]

Table 2.1: Recent machine learning approaches to grasp quality prediction. Details of the different object-grasp representations and feature extraction techniques can be found in subsection 2.2.2, while data synthesis methods are examined more closely in subsection 2.4.1.

More recent approaches project local features at contact locations onto 2D representations then apply CNN methods in a similar manner with RGB images to learn the grasp rating directly [15, 23, 19]. These methods mainly differ in the

representations their CNN models take as input, as described in subsection 2.2.2. While [15, 19] implement a pure LeNet CNN model [14] to learn the grasp quality directly, Mahler et al [23] also include a single-layer fully connected network which takes the distance between the camera and the gripper as input. The outputs of the CNN and the fully-connected network are then combined using more fully-connected layers to produce a final grasp success probability prediction.

Methods trying to learn grasp quality are highly dependent on the dataset on which their models are trained. Humans are still the most versatile and dexterous manipulators known, but data containing human grasp experience is difficult and costly to collect. Of the approaches listed in table 2.1, Jiang et al.'s [18] train their model on a human-labeled dataset, which only has around 300 data points for both training and testing. The extended version of this dataset, used by Lenz et al. [20], also has only around 1000 data points. Because of this scarcity, synthesizing data for training grasp evaluation models has become popular in the more recent approaches [19, 15, 23]. A review of data synthesis techniques for grasping will be included in the next section. The introduction of a new large-scale human grasp experience dataset by Saudabayev et al. [32], however, may open up new possibilities for direct training of grasp evaluation models on human grasp experience. The database contains RGB data from a head-mounted camera, depth data from a hand-mounted sensor, and kinematic data from an inertial motion capture suit. 3826 grasps of 35 different types are recorded.

2.4 Generating data for grasp success prediction

2.4.1 Data synthesis for robot grasping

Most approaches to synthesize data for grasp evaluation models generate training samples from object meshes. Earlier methods then learn to match detected objects with these object models to find suitable grasps to execute. Notably Goldfeder et al. [11] generate a database of form-closure grasps for 3D object models, in which each grasp entry is represented by a pregrasp in the form of a two-dimensional eigengrasp [4], and the final grasp pose. Grasps are sampled from a subspace of each robot hand's degree-of-freedom (DOF) space. The authors then used a nearest-neighbor

algorithm to match each detected object to a model in this database [12] to find grasp candidates.

Several grasp synthesis approaches generate perceptual data from object meshes at different viewpoints [23, 15, 19] for constructing a grasp knowledge database. Mahler et al. [23] use 3D mesh models from Dex-Net 1.0 [22], rescaling them to fit within the gripper's physical constraints. Stable poses are computed for each object, and poses with a probability of occurrence above a threshold are stored. For each stable pose, a set of grasps is sampled perpendicular to the table and collision-free. Rejection sampling is used during grasp generation to ensure coverage of the object surface. Grasps are then ranked and selected using a trained grasp quality metric. Finally, 2D images are generated from these grasp candidates as training data with the antipodal axis aligned to the middle row of the image. Figure 2.2 provides a visualization of this data generation process.

Kappler et al. [19] generate data using the OpenRAVE [1] [6] simulator. Reference grasps are sampled using the geometric strategy available in OpenRAVE, where the contact points are approach points, and the surface normals are approach vectors. For each approach point, 8 wrist rotations around the approach vector and two translation offsets are applied, resulting in 16 candidate grasps per reference grasp. Data points are then generated for each grasp using the local shape representation described in subsection 2.2.2. A set of metrics is then applied to evaluate candidate grasps, creating the data labels.

All these approaches, however, still rely on analytical metrics to label grasp candidates. These metrics, as mentioned in subsection 1.3, can be unreliable when tested on real systems. As robots can rarely acquire the comprehensive model of their environment from perceptual sensors, training data for robot learning should reflect what is normally available to the robot's perception pipeline. As a result, this project will focus on data synthesis approaches which generate grasp knowledge databases synthesizing and labeling perceptual data.

[1] http://openrave.org/

2.4.2 Data augmentation

For learned models to become robust to noise in real-world sensors and to alleviate the scarcity of real data, augmentation techniques are often applied to available data before training [8, 19, 16]. As defined in [14], data augmentation refer to transforming existing data "without altering their natures." Which features exactly should be preserved during the augmentation process depend on the data type.

With the relatively recent popularization of RGB-D sensors, few approaches exist to augment depth data. Eitel et al. [8] sample noise patches of fixed size from real RGB-D data and divide them into five groups depending on the number of missing depth reading in each patch. To create a final noise pattern, a pair of patches are sampled at random from two different groups, then randomly added or subtracted with each other and optionally inverted. This process is repeated until a specific number of patterns is generated. During training, at a predefined probability, a noise pattern will be randomly selected from this set and applied to the depth sample to create the noised input. Kappler at al [19] introduces noise to the object poses while sampling for reference grasps, before generating each data point. Specifically, at the grasp approach point, the object is rotated around a random axis, then a random offset is added to the surface pose. These variations of the object pose are intended to capture the errors in calculating the surface normals and the surface noise during the reconstruction as described in subsection 2.2.2. Gupta et al. [16] have a simpler approach, only adding a low-frequency white noise to the disparity images to augment the depth information.

In contrast, many augmentation approaches have been developed for images. Several involve simple geometric transformations such as mirroring, rotating, shifting [14] or photometric such as color scaling, contrasting [7]. Hauberg et al. [17] propose an elegant method to learn these transformations using diffeomorphisms. The authors first estimate the transformations for image pairs in each class. These transformations are then used as observations to build a class-wise statistical model of transformations. New transformations are then sampled from this model and applied to input images.

2.4.3 Generative Adversarial Networks

Recently, Generative Adversarial Networks (GAN) are introduced as a powerful model for synthesizing images. As described in the original paper by Goodfellow et al. [13], a minimax algorithm is used to optimize two competing networks: one is the generator G which learns the generator's distribution p_g over data $\boldsymbol{x}$ to map from a noise signal $\boldsymbol{z}$ to the data space, and the other is the discriminator D which learns to predict whether an input is from data or from p_g. The minimax game is formulated for the value function $V(D, G)$ as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}\left[\log(1 - D(G(\boldsymbol{z})))\right] \quad (2.10)$$

. In addition to applications in image generation, GANs have also recently been applied for reconstructing 3D objects from a single depth image [38], which in turn can be used to introduce controlled permutations to object models during grasp data synthesis, or to provide additional information about the object for the grasp planner during execution.

3

Methodology

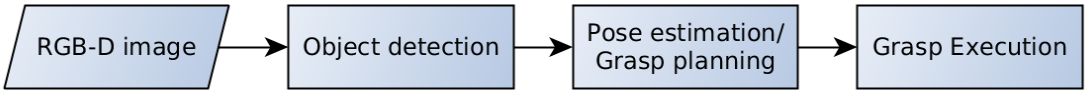


Figure 3.1: Pipeline for object grasping.

In order to perform the experiments described in the next chapter, it is necessary to integrate a full object grasping pipeline on the HSR. Figure 3.1 illustrates the main components of this pipeline. Details on the approaches considered and implemented for each component will be discussed in this chapter.

3.1 Object detection

Originally object detection in the Robocup@Home team's code base [1] is done using existing algorithms available in the Point Cloud Library (PCL) [2]. First, a random sample consensus (RANSAC) algorithm is executed to find a plane and its inlaying points using point clouds from the RGB-D camera (multiple clouds are accumulated before processing to alleviate noise from the sensor). Only planes whose normals are parallel to the z-axis of the world frame, or in other words, parallel to the ground, are considered. The convex hull of the plane and a given

[1] GitHub link for the source code: `https://github.com/b-it-bots/mas_perception/`
[2] http://docs.pointclouds.org/

Figure 3.2: Example of objects that are partially invisible in point clouds.

height are then used to form a 3D polygonal prism and extract points inside this prism. Clusters are then segmented from the extracted points as objects. In practice, this method proves to be unreliable for Robocup@Home objects as the SAC algorithm does not always find a plane, especially when several objects are present. Relying solely on the depth information can also prove inadequate for object detection, as some materials of domestic objects can be noisy or even invisible for the depth sensor (two examples of which are shown in figure 3.2). The 26 parameters required for tuning the detection pipeline also make it inflexible in adapting different surfaces and objects. Finally, execution of the algorithm is slow, taking several seconds for plane detection and cluster extraction.

The HSR from Toyota provides a built-in segmentation solution based on an existing Robot Operating System (ROS) [1] package called `tabletop_object_detector` [2]. ROS is a software framework designed for robots, containing drivers for various robotic hardware devices as well as libraries commonly used in robotic applications. The detection package assumes objects to have a rotationally symmetric shape, to have a fixed orientation on a horizontal surface (e.g. a table), and to be no more than 3cm apart from each other. This package follows a similar segmentation algorithm to the aforementioned method, first detecting a plane then clustering points above this plane as objects. While the

[1] http://www.ros.org
[2] http://wiki.ros.org/tabletop_object_detector

package is better optimized and performs faster than the previous implementation, the source code of the modified version is not released by Toyota, and the algorithm suffers from limitations similar to the ones described above.

3.1.1 Single Shot MultiBox Detector

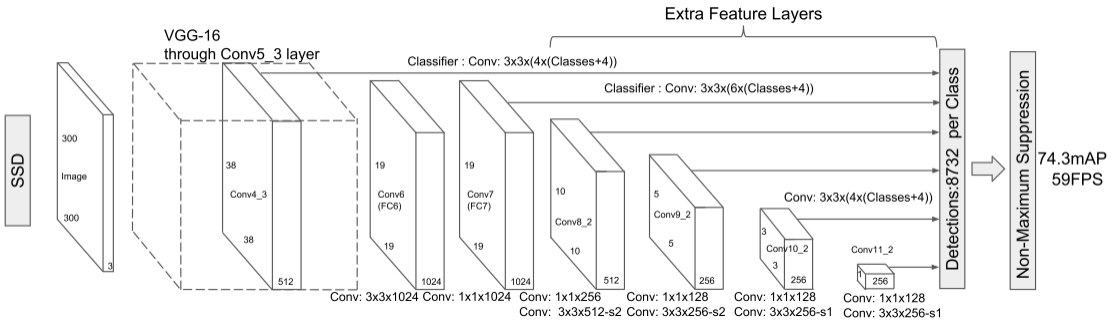


Figure 3.3: SSD architecture [21].

Because the above detection implementations prove not reliable enough for performing grasp experiments, the Single Shot MultiBox Detector (SSD) algorithm [21] was integrated into the grasping pipeline for this project. In recent years, the success of CNN in image recognition and the introduction of large-scale object detection datasets (i.e. Microsoft COCO [1] and Pascal VOC [2]) has bolstered research interests in detecting objects directly from RGB images [14]. Gu et al. [14] include a review of recent CNN-based object detection architectures in their survey of CNN. Among these approaches, SSD stands out as being able to detect objects accurately at a high frame rate. The architecture introduces a set of "default boxes" for each training image and select ones which has a certain amount of overlapping with the ground truth boxes. The network then learns the offsets between the ground truth and default boxes, specifically between their center coordinates, widths, and heights, alongside with a confidence score for the object class detected. For N selected default boxes, the overall loss function is the sum of the localization loss

[1] http://cocodataset.org/
[2] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

(L_{loc}) and confidence loss (L_{conf}):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + L_{loc}(x, l, g))$$

, where x is a chosen default box, c is the predicted confidence, l is the predicted box, and g is the ground truth box. Detection training is applied to multiple resolutions of the feature map (illustrated in figure 3.3) for the network to learn scale awareness.

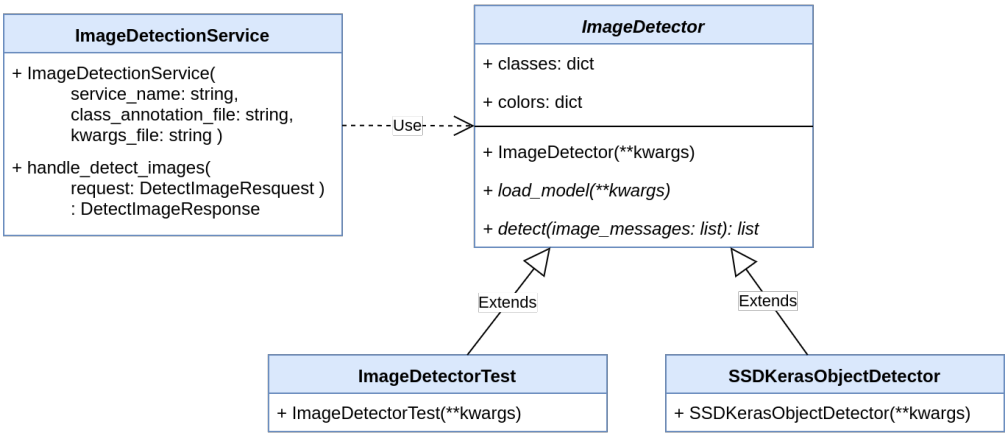


Figure 3.4: Class diagram for the image detection implementation. In italics are abstract classes and methods.

The SSD implementation by Pierluigi Ferrari [1] is extended to work with ROS and the Robocup@Home perception software structure. To allow for easy adaptation to different image detection architectures and implementations in the future, an abstraction layer for the image detection service was created, as illustrated in figure 3.4. The `ImageDetectionService` class loads the specific realization of `ImageDetector` class and its configurations at launch via constructor parameters and uses its instantiation to handle detection service requests from other ROS nodes during runtime. Figure 3.5 shows a sample detection result from the integrated SSD architecture. For detecting RoboCup@Home objects, we use a provided model trained on the COCO dataset.

[1] GitHub link for source code: `https://github.com/pierluigiferrari/ssd_keras`

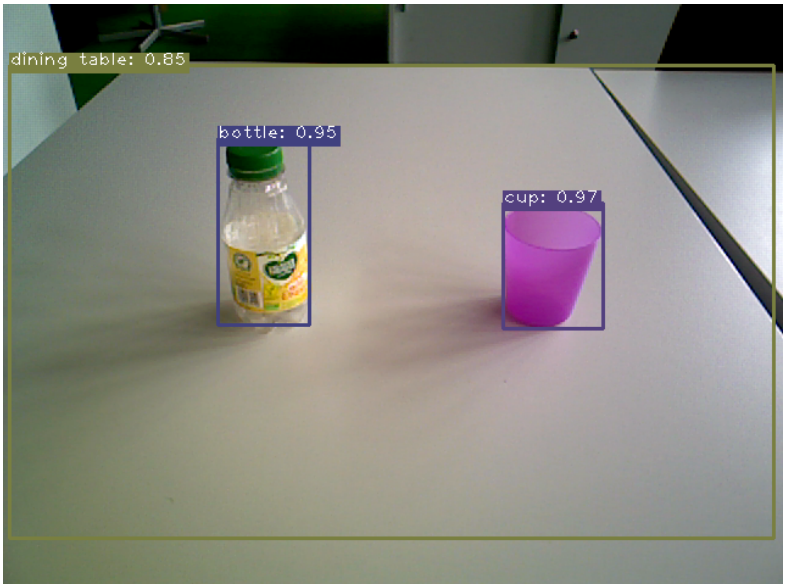


Figure 3.5: Example of SSD detection using a model trained on the COCO dataset. The bounding box label contains the detected object class and the prediction confidence.

3.2 Pose estimation and grasp planning

This section describes how grasp poses are inferred from the objects detected in RGB images. First, simple pose estimation algorithms are implemented as the baseline grasp planning method. Next, the GQCNN grasp planner introduced by Mahler et al. [23] is integrated for comparison.

3.2.1 Pose estimation for detected objects

For the baseline method, the grasp approach vector is assumed to align with the x-axis of the robot base coordinate frame, as illustrated in figure 3.6. This assumption reduces the 6-DOF grasp-pose-finding problem to finding the 3D coordinates of the object. As shown in figure 3.7, the RGB image is extracted from the point cloud for detection. After the image detection service responded with 2D bounding boxes of the objects, point clouds are transformed to the robot's base frame, and points within the detected regions are extracted from the transformed clouds. This step relies on the second assumption that the point clouds are

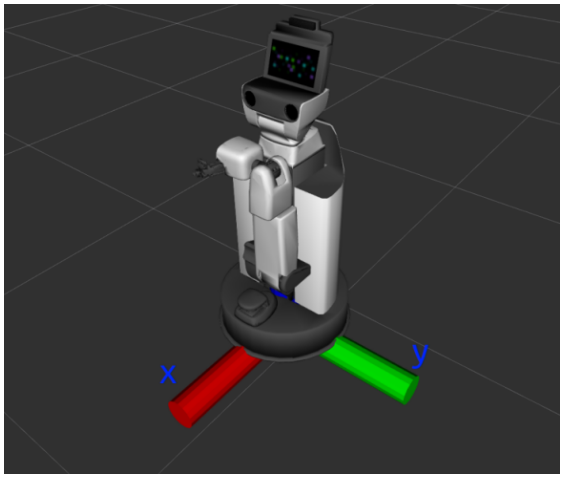


Figure 3.6: `base_link` coordinate frame. The z-axis points upward into the robot.

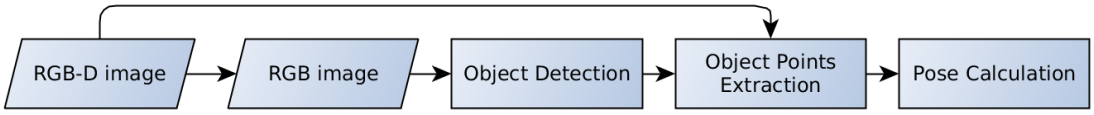


Figure 3.7: Flowchart for the pose estimation pipeline.

organized: the points resemble an image or a 2D matrix where data is organized into rows and columns. This allows direct projection of pixel coordinates to points in the point cloud. As PCL is implemented in `C++`, while the SSD implementation is written in `Python`, interfaces between the two languages are developed for functions processing point clouds using the `Boost.Python` [1] library. The grasping position $\mathbf{p} = (x, y, z)$ is calculated from the extracted 3D coordinates. If matrix B of shape $M \times 3$ contains all the 3D coordinates extracted using the detected object's bounding box, we either take the closest point along the x-axis of the base frame:

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \min_{i=1}^{M} B_{i,1} \\ \frac{1}{M} \sum_{i=1}^{M} B_{i,2} \\ \frac{1}{M} \sum_{i=1}^{M} B_{i,3} \end{pmatrix} \tag{3.1}$$

[1] https://www.boost.org/doc/libs/1_68_0/libs/python/

, or the mean along all three axes:

$$p_j = \frac{1}{M} \sum_{i=1}^{M} B_{i,j} \tag{3.2}$$

, where $j = 1, 2, 3$.

3.2.2 Grasp detection with GQCNN

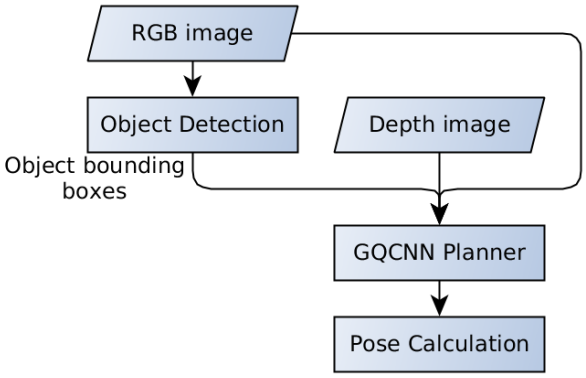


Figure 3.8: Flowchart for the GQCNN pipeline.

The GQCNN grasp planner introduced by Mahler et al. [1] [23] is integrated into ROS as a grasp detection service, which takes in a color, a depth image, and the detected object's bounding box. After planning it returns a pregrasp pose and a probability of grasp success. The planner first combines the image pair into a single 4-channel matrix and uses the 2D bounding box to crop out the relevant region. Bipedal grasp candidates, characterized by the pixel coordinates of the grasp center and the gripper's angle in the image plane, are then sampled for the cropped matrix. Corresponding local representations (as described in section 2.2.2) are extracted for each grasp as input for the GQCNN model. The output success probability scores are compared to find the optimal grasp. To calculate the 3D grasp pose, the provided planner assume that the approach vector to align with

[1] GitHub link: `https://github.com/BerkeleyAutomation/gqcnn`

the camera axis. The returned pose is 10cm away from the object's surface along this axis.

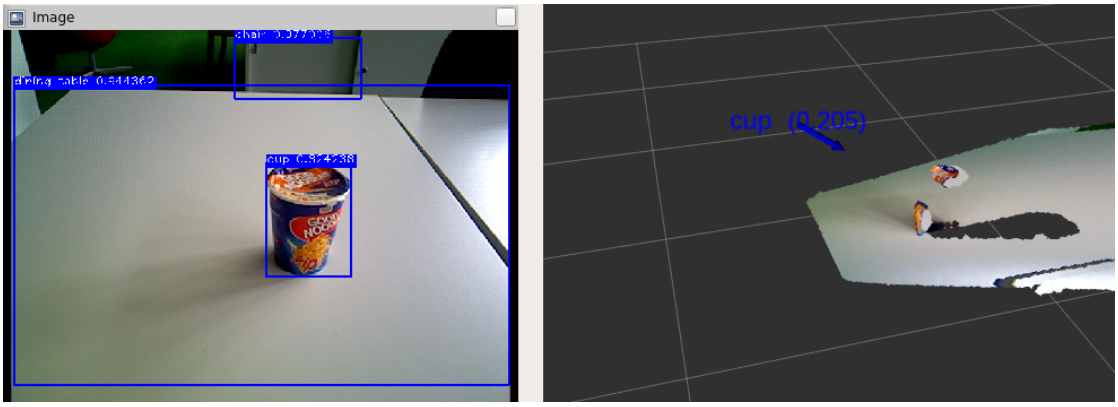


Figure 3.9: The blue arrow represents an example grasp pose returned from the GQCNN grasp planner. The number in parentheses is the probability of grasp success returned by the GQCNN model. On the left is the visualized object detection result.

For this project, the GQCNN model trained on the full Dex-Net 2.0 dataset containing 6.7 million synthetic data points [23] is integrated. Figure 3.9 shows an example of grasp poses returned from the grasp planner using this model. Out of the four objects selected for experiments (shown in figure 4.2), the integrated planner was only able to detect grasps for the noodle box at very low confidence. The planner's unreliability in grasp detection and its long planning time (from 10 to 20 seconds per detected object) discourage from including them in the grasp experiments performed. This can be attributed to the experimental setup, which is different from the statically mounted RGB-D camera with its principal axis perpendicular to the grasping surface (e.g. a table) as described in the original paper. However, for a mobile robot like the HSR, such a setup is often impractical.

3.3 Grasp execution

Originally manipulation planning is done using the MoveIt! [1] mobile manipulation library. Currently, MoveIt! only fully supports planners available in the Open Motion Planning Library (OMPL) [2], which are primarily randomized planners.

[1] http://moveit.ros.org
[2] http://ompl.kavrakilab.org

This kind of planners samples a set of joint trajectories and choose the first plan that is physically feasible. Because of their stochastic nature, the planners may give unnatural, unexpected manipulation plans, such as one shown in figure 3.10.

Figure 3.10: MoveIt! failing to plan a simple grasp.

In order to have a more reliable setup for grasp experiments, manipulation motion is planned using the integration and extension of Dynamic Motion Primitives (DMP) libraries for a parallel project by Padalkar [26]. DMP models demonstrated trajectories as a series of differential equations, allowing the robot to learn human motions and calculate kinematic commands via solving these equations to perform new manipulation tasks in a similar way. The work by Padalkar extends the DMP libraries to also use navigation commands to perform manipulation tasks that can't be reached by the learned motion alone.

4

Experiments

The implementation described in the previous chapter was used to perform two sets of experiments. This chapter will provide details on the experimental setup and discuss the experiments' results.

4.1 Setup

Figure 4.1: Experiment setup: the robot assumes the same pose before each grasp.

During the experiments, before each grasp, the robot is moved to a predefined location (marked on the floor) facing the dining table in the C069 laboratory as shown in figure 4.1. Two sets of experiments are performed for the two pose estimation methods described in section 3.2.1, for the objects shown in figure 4.2. As mentioned previously, the SSD model used for object detection is trained on the COCO dataset. Since this does not match the RoboCup@Home objects completely, we allow less confident detections to prioritize evaluating the pose estimation methods as a baseline for grasp planning approaches. For example, the salt box or the noodle box can be detected either as a bottle or a cup, whereas the duct tape can be detected as a bowl.

(a) A white styrofoam ball

(b) A roll of duct tape

(c) A cup noodles box

(d) A salt container

Figure 4.2: Objects selected for the experiments.

For both experiments, the robot first aligns itself with the estimated pose's y-coordinate with respect to the `base_link` frame (shown in figure 3.6). The grasp is then executed using the learned primitives, and the robot moves forward if the x-coordinate is too far away. Also with respect to the `base_link` frame, objects detected more than 90cm away in the x direction and lower than the table height of 75cm in the z direction are ignored. Objects' positions on the table are arbitrary for each grasp, but kept within a 35cm distance from the table's edge to ensure reachability. The calculated x coordinate is offset by 4cm to take into account the distance between the gripper coordinate frame and the fingertips. Pose detected further than 0.8m in the x-axis or lower than 0.78m in the z axis are adjusted to these values to avoid severe collisions. The robot arm still collides with the table using these adjusted coordinates. Only one object is grasped at a time.

For each object, the entire grasping pipeline is executed 20 times for 20 different positions. A trial is counted as successful if the object stays in the gripper after the arm moves back close to the robot's body. If the object slips off the gripper or the arm collides with the table, the execution is counted as a failure.

4.2 Results

Object	Mean x		Minimum x	
	Success	Failure	Success	Failure
Salt	17	3	16	4
Ball	8	12	15	5
Noodle box	16	4	12	8
Duct tape	7	13	13	7

Table 4.1: Results of the grasp experiments. On the left are results from using the mean x coordinates for estimating the grasp pose, and on the right are results from using the min coordinates along the x-axis.

Table 4.1 shows the grasp success/failure results of the experiments. During the experiments, many of the failures while grasping the styrofoam ball is caused by the gripper pushing on the ball and making it roll forward (an example is shown in figure 4.3). For this object, grasping using the minimum coordinate along the

x-axis proves to be much more reliable. Low objects like the roll of duct tape give low estimates for the z coordinate, causing many failures via collisions between the arm and the table. This suggests that a different approach vector from above may perform better for such objects. Many of the failures also occur because of slipping while the arm is moving back, which suggests that verification from the force sensors may boost grasp reliability.

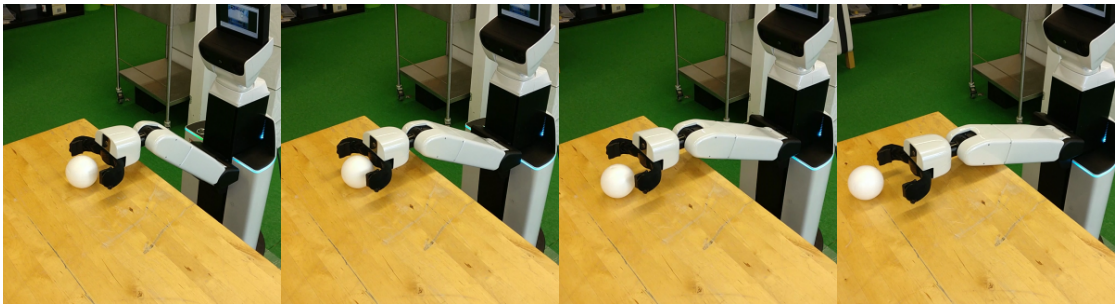


Figure 4.3: A failure instance where the gripper pushes on the ball and it rolls forward.

5

Conclusions

5.1 Contributions

This report reviews recent advances in aspects most relevant to generating data for training a grasp evaluation models, namely feature extraction from perceptual data, object-grasp representation, grasp evaluation metrics, and data generation techniques. Additionally, five recent, prominent approaches to data synthesis for grasp evaluation are examined, and their solutions for each of the four aspects mentioned above are summarized in table 2.1. Two of these approaches utilize human labeled data, but their dataset is small because human grasping datasets are time-consuming and costly to collect. The other three approaches rely on analytical grasp metrics to label their data, either built into simulators like OpenRAVE or GraspIt!, or calculated directly f or each grasp candidates. As discussed in section 1.3, analytical metrics have been shown to be unreliable when applied to real systems, and simulators often fail short in capturing all the details of the real world.

The second contribution of this project is that, in collaboration with another Research and Development project by Padalkar [26], a full grasping pipeline is implemented, from perceiving objects to grasp execution. Two pose estimation methods are implemented, serving as baselines for experimenting and comparing with more advanced grasp planning techniques.

5.2 Future work

Several extensions and optimizations for the current implementation of the grasping pipeline are possible. First, the grasp execution implementation can be extended to allow the robot to grasp from different directions. Next, the detection model for the SSD architecture can also be fine-tuned to perform better in detecting RoboCup@Home objects. Nearest neighbor algorithms can also be used to improve object pose estimation from the points extracted from RGB-D clouds. Surface normal calculations can also be implemented to replace the grasp pose calculation provided with the GQCNN software. Specifically, the pixel coordinate of the detected can be directly transformed to its corresponding 3D coordinate, and the surface normal around this point can be used as the grasp's approach vector. This surface normal can also be used as the approach vector for the implemented pose estimation algorithms.

Furthermore, several of the approaches reviewed in chapter 2 can be integrated. Particularly, the shape completion technique introduced by Varley et al. [36], can be used as a prior for Gualtieri et al. [15] sampling of the occluded points for their 12-channel 2D representation. Training on the new human grasp experience dataset by Saudabayev et al. [32] can also be examined for a direct comparison with training on data synthesized using analytical grasp metrics and simulation. Techniques to introduce task awareness into data generation can also be examined, i.e how the knowledge of the task to be performed with the manipulated object can be encoded into the data synthesis process.

A

Experiment Reports

Experimental report 28.07.2018

Date & Time	28.07.2018, 17:00 - 21:00
Experimenter	Minh Nguyen, Alex Mitrevski
User Account	lucy@192.168.50.201
Hardware configuration	Toyota HSR
Tested feature	Domestic object grasping
Test setup / environment	HBRS, C069 lab

Experiment description

The purpose of this experiment is to evaluate the performance of a baseline grasp planner for grasping common domestic objects, which uses a fixed grasping orientation and a position determined by averaging the points of an object's point cloud.

Assumptions

- All robot components are working correctly (no internal faults)
- The experiment is performed under normal room lighting conditions, which are potentially changing during the experiment (e.g. a light going off or the weather changing between sunny and cloudy)
- The position of an object is kept fixed during grasp planning and grasp execution (i.e. there is no malicious external agent that moves the object during an experimental trial)

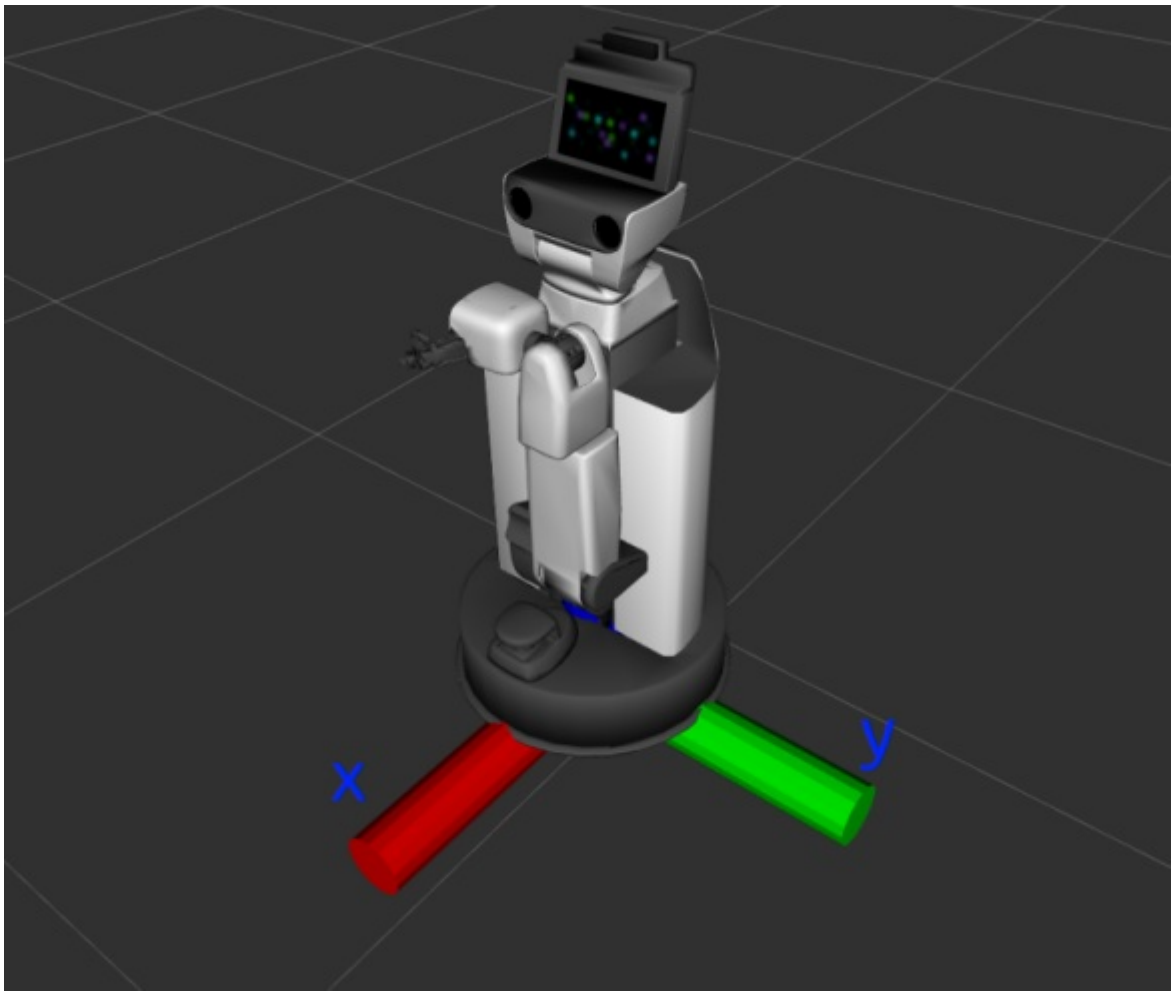
Experimental procedure

The experimental setup is as follows:

- The robot grasps objects from the dining room table in C069
- In the experiment, there is a single object on the table at a time that the robot needs to grasp

- Only the following objects are used for the experiment: a salt box, a ball, a noodle box, and duct tape (shown below)

- For recognising objects, an SSD detector trained on the COCO dataset is used; since the detector is not trained on our particular object set, the salt box is detected as either a bottle or a cup, the ball is detected as either an orange or an apple, the noodle box is detected as a cup, and the duct tape is detected as a bowl.
- Before the experiment, the distance from the robot to the nearest table edge is fixed; however, the robot aligns itself with the position of an object before grasping it and may move forward to approach the object if it is too far away.
- The position of the object on the table is arbitrary, but is kept within 35cm from the edge nearest to the robot so that reachability is always ensured.
- We ignore object detections if the estimated object position is further than 90cm along x and below 75cm along z with respect to the `base_link` frame. The `base_link` frame is shown in the figure below.



- For each object, we repeat a grasp 20 times (once for 20 different positions of the object). A trial is considered successful if the robot picks an object and the object stays within the gripper after the arm is moved back. Trials in which the arm collides with the table or the object slips out of the gripper are considered unsuccessful. Grasping motions are performed using a prerecorded dynamic motion primitive.

Results

Object	Success	Slip	Fail	Comments
Salt	17	0	3	failures due to the object slipping out of the gripper
Light ball	8	0	12	failures due to the ball being pushed away while grasping or the arm colliding with the table
Noodle box	16	0	4	failures due to collisions with the table
Duct tape	7	0	13	failures due to collisions with the table; in five of the failed trials, the object was successfully grasped

Observations

- The minimum height (with respect to the `base_link` frame) is set to the height of the table (0.78m); any calculated pose lower than that is adjusted to this height.
- The maximum allowed distance along x from the robot to an object is 80cm; any calculated position larger than that is adjusted to 80cm. In addition, the x position is offset 4cm to account for the gripper frame being at the back of the gripper.
- Objects sometimes slip through the robot's gripper; it is thus necessary to integrate grasp verification into the grasping procedure.
- Over time, the pose estimate from the camera seems to degrade; this might be due to the camera overheating or going to autofocus mode after a drastic light change

(e.g. the light turns off).
- When an object is placed closer to the robot, the pose estimate is consistently worse than when the object is further away

Media

A video of the noodle box grasping trials can be seen here.

Experimental report 31.07.2018

Date & Time	31.07.2018, 15:00 - 17:15
Experimenter	Minh Nguyen, Alex Mitrevski
User Account	lucy@192.168.50.201
Hardware configuration	Toyota HSR
Tested feature	Domestic object grasping
Test setup / environment	HBRS, C069 lab

Experiment description

The purpose of this experiment is to evaluate the performance of a baseline grasp planner for grasping common domestic objects, which uses a fixed grasping orientation and a position determined by averaging the points of an object's point cloud along the y and z axes, but the minimum position along the x axis with respect to the `base_link` frame.

Assumptions

- All robot components are working correctly (no internal faults)
- The experiment is performed under normal room lighting conditions, which are potentially changing during the experiment (e.g. a light going off or the weather changing between sunny and cloudy)
- The position of an object is kept fixed during grasp planning and grasp execution (i.e. there is no malicious external agent that moves the object during an experimental trial)

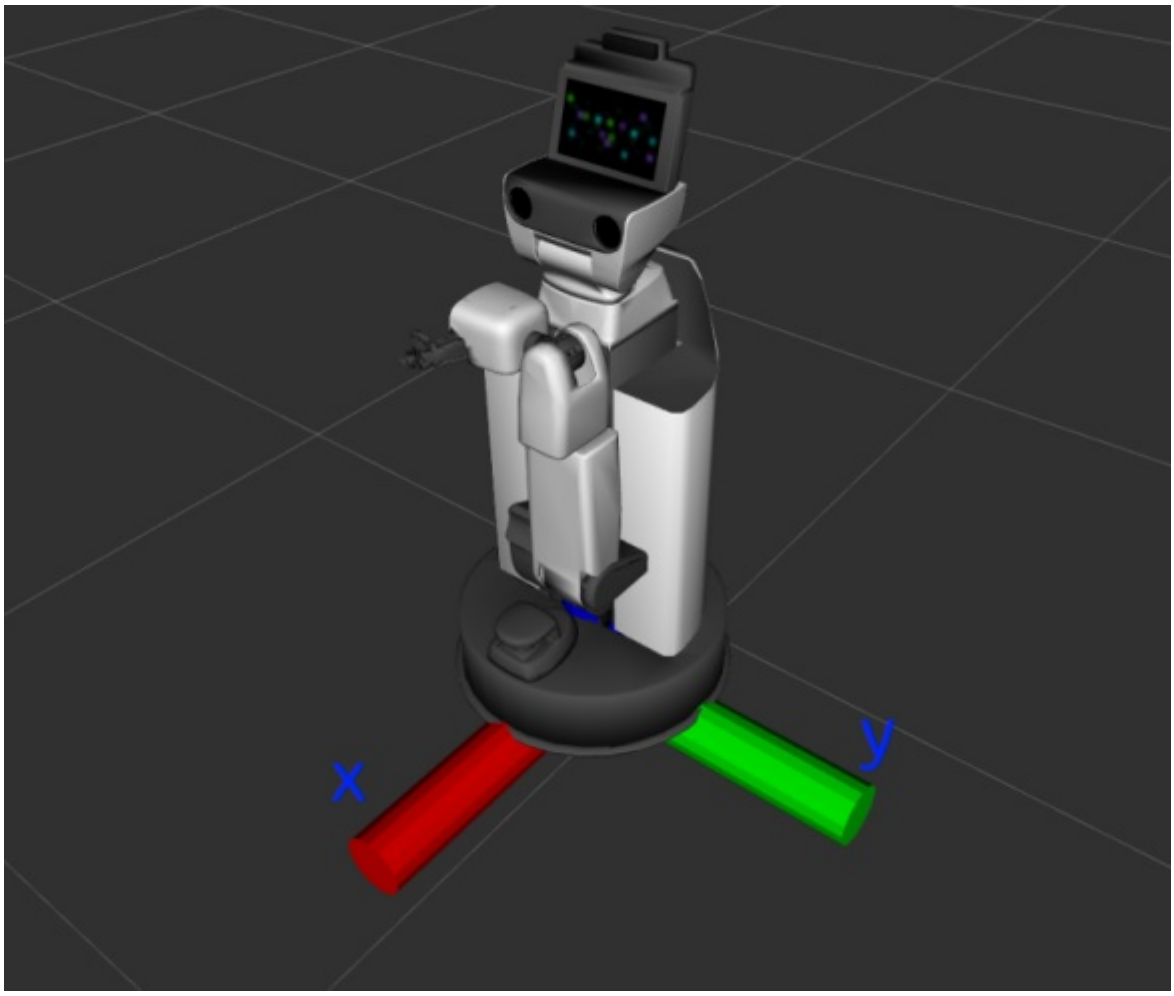
Experimental procedure

The experimental setup is as follows:

- The robot grasps objects from the dining room table in C069
- In the experiment, there is a single object on the table at a time that the robot needs to grasp

- Only the following objects are used for the experiment: a salt box, a ball, a noodle box, and duct tape (shown below)

- For recognising objects, an SSD detector trained on the COCO dataset is used; since the detector is not trained on our particular object set, the salt box is detected as either a bottle or a cup, the ball is detected as either an orange or an apple, the noodle box is detected as a cup, and the duct tape is detected as a bowl.
- Before the experiment, the distance from the robot to the nearest table edge is fixed; however, the robot aligns itself with the position of an object before grasping it and may move forward to approach the object if it is too far away.
- The position of the object on the table is arbitrary, but is kept within 35cm from the edge nearest to the robot so that reachability is always ensured.
- We ignore object detections if the estimated object position is further than 90cm along x and below 75cm along z with respect to the `base_link` frame. The `base_link` frame is shown in the figure below.



- For each object, we repeat a grasp 20 times (once for 20 different positions of the object). A trial is considered successful if the robot picks an object and the object stays within the gripper after the arm is moved back. Trials in which the arm collides with the table or the object slips out of the gripper are considered unsuccessful. Grasping motions are performed using a prerecorded dynamic motion primitive.

Results

Object	Success	Slip	Fail	Comments
Salt	16	0	4	failures due to collisions with the table and incorrect object pose estimates
Light ball	15	0	5	failures due to collisions with the table
Noodle box	12	4	4	failures due to collisions with the table
Duct tape	13	1	6	failures due to collisions with the table and object slips; in one of the failed trials, the object was successfully grasped

Observations

- The minimum height (with respect to the base_link frame) is set to the height of the table (0.78m); any calculated pose lower than that is adjusted to this height.
- The maximum allowed distance along x from the robot to an object is 80cm; any calculated position larger than that is adjusted to 80cm. In addition, the x position is offset 4cm to account for the gripper frame being at the back of the gripper.
- Objects sometimes slip through the robot's gripper; it is thus necessary to integrate grasp verification into the grasping procedure.
- Over time, the pose estimate from the camera seems to degrade; this might be due

to the camera overheating or going to autofocus mode after a drastic light change (e.g. the light turns off)

- When an object is placed closer to the robot, the pose estimate is consistently worse than when the object is further away

References

[1] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. *Unsupervised Feature Learning for RGB-D Based Object Recognition*, pages 387–402. Springer International Publishing, Heidelberg, 2013. ISBN 978-3-319-00065-7. doi: 10.1007/978-3-319-00065-7_27. URL https://doi.org/10.1007/978-3-319-00065-7_27.

[2] J. Bohg, M. Johnson-Roberson, B. Len, J. Felip, X. Gratal, N. Bergstrm, D. Kragic, and A. Morales. Mind the gap - robotic grasping under incomplete observation. In Yuan F. Zheng, editor, *2011 IEEE International Conference on Robotics and Automation*, pages 686–693, May 2011. doi: 10.1109/ICRA.2011.5980354.

[3] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis: A survey. *IEEE Transactions on Robotics*, 30(2):289–309, April 2014. ISSN 1552-3098. doi: 10.1109/TRO.2013.2289018.

[4] Matei T. Ciocarlie and Peter K. Allen. Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28(7):851–867, 2009. doi: 10.1177/0278364909105606. URL https://doi.org/10.1177/0278364909105606.

[5] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic. Generalizing grasps across partly similar objects. In Antonio Bicchi, editor, *2012 IEEE International Conference on Robotics and Automation*, pages 3791–3797, May 2012. doi: 10.1109/ICRA.2012.6224992.

[6] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, 2010.

[7] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In Katsushi Ikeuchi, Christoph Schnrr, Josef Sivic, and Ren Vidal, editors, *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, Dec 2015. doi: 10.1109/ICCV.2015.304.

[8] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In Wolfram Burgard, editor, *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687, Sept 2015. doi: 10.1109/IROS.2015.7353446.

[9] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Adaptive data augmentation for image classification. In Lina Karam, editor, *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692, Sept 2016. doi: 10.1109/ICIP.2016.7533048.

[10] C. Ferrari and J. Canny. Planning optimal grasps. In Giuseppe Menga Giuseppe Menga Giuseppe Menga Giuseppe Menga Giuseppe Menga, editor, *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295 vol.3, May 1992. doi: 10.1109/ROBOT.1992.219918.

[11] C. Goldfeder, M. Ciocarlie, Hao Dang, and P. K. Allen. The columbia grasp database. In Antonio Bicchi, editor, *2009 IEEE International Conference on Robotics and Automation*, pages 1710–1716, May 2009. doi: 10.1109/ROBOT.2009.5152709.

[12] Corey Goldfeder and Peter K. Allen. Data-driven grasping. *Autonomous Robots*, 31(1):1–20, Jul 2011. ISSN 1573-7527. doi: 10.1007/s10514-011-9228-1. URL `https://doi.org/10.1007/s10514-011-9228-1`.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing*

Systems 27, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[14] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354 – 377, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.10.013. URL http://www.sciencedirect.com/science/article/pii/S0031320317304120.

[15] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt. High precision grasp pose detection in dense clutter. In Wolfram Burgard et al., editor, *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605, Oct 2016. doi: 10.1109/IROS.2016.7759114.

[16] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 345–360, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10584-0.

[17] Sren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John Fisher, and Lars Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 342–350, Cadiz, Spain, 09–11 May 2016. PMLR. URL http://proceedings.mlr.press/v51/hauberg16.html.

[18] Yun Jiang, S. Moseson, and A. Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In Antonio Bicchi, editor, *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3304–3311, May 2011. doi: 10.1109/ICRA.2011.5980145.

[19] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In Allison Okamura, editor, *IEEE International Conference on Robotics and*

Automation (ICRA), pages 4304–4311, May 2015. doi: 10.1109/ICRA.2015.7139793.

[20] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. doi: 10.1177/0278364914549607. URL `https://doi.org/10.1177/0278364914549607`.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

[22] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In Allison Okamura, editor, *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[23] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR*, abs/1703.09312, 2017. URL `http://arxiv.org/abs/1703.09312`.

[24] Mauricio Matamoros, Caleb Rascon, Justin Hart, Dirk Holz, and Loy van Beek. Robocup@home 2018: Rules and regulations. `http://www.robocupathome.org/rules/2018_rulebook.pdf`, 2018.

[25] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994. ISBN 0849379814.

[26] Abhishek Padalkar. Dynamic motion primitives. Technical report, Hochschule Bonn-Rhein-Sieg, 2018.

[27] L. Porzi, S. R. Bul, A. Penate-Sanchez, E. Ricci, and F. Moreno-Noguer. Learning depth-aware deep representations for robotic perception. *IEEE Robotics and Automation Letters*, 2(2):468–475, April 2017. doi: 10.1109/LRA.2016.2637444.

[28] C. R. Qi, H. Su, M. Niener, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In Lourdes Agapito, Tamara Berg, Jana Kosecka, and Lihi Zelnik-Manor, editors, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, June 2016. doi: 10.1109/CVPR.2016.609.

[29] Máximo A. Roa and Raúl Suárez. Grasp quality measures: Review and performance. *Autonomous Robots*, 38(1):65–88, January 2015. ISSN 0929-5593. doi: 10.1007/s10514-014-9402-3. URL `http://dx.doi.org/10.1007/s10514-014-9402-3`.

[30] C. Rubert, D. Kappler, A. Morales, S. Schaal, and J. Bohg. On the relevance of grasp metrics for predicting grasp success. In Tony Maciejewski, editor, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 265–272, Sept 2017. doi: 10.1109/IROS.2017.8202167.

[31] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, March 2012. ISSN 0921-8890. doi: 10.1016/j.robot.2011.07.016. URL `http://dx.doi.org/10.1016/j.robot.2011.07.016`.

[32] Artur Saudabayev, Zhanibek Rysbek, Raykhan Khassenova, and Huseyin Atakan Varol. Human grasping database for activities of daily living with depth, color and kinematic data streams. *Scientific Data*, 5:180101, May 2018. URL `http://dx.doi.org/10.1038/sdata.2018.101`. Data Descriptor.

[33] K.B. Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996. doi: 10.1177/027836499601500302. URL `https://doi.org/10.1177/027836499601500302`.

[34] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In Lisa OConner, editor, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, July 2017. doi: 10.1109/CVPR.2017.241.

[35] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In Katsushi Ikeuchi, Christoph Schnrr, Josef Sivic, and Rene Vidal, editors, *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, Dec 2015. doi: 10.1109/ICCV.2015.114.

[36] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In Tony Maciejewski, editor, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2442–2447, Sept 2017. doi: 10.1109/IROS.2017.8206060.

[37] J. Weisz and P. K. Allen. Pose error robust grasping from contact wrench space metrics. In Antonio Bicchi, editor, *2012 IEEE International Conference on Robotics and Automation*, pages 557–562, May 2012. doi: 10.1109/ICRA.2012.6224697.

[38] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3d object reconstruction from a single depth view with adversarial learning. In R. Cucchiara, Y. Matsushita, N. Sebe, and S. Soatto, editors, *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[39] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Hugo Larochelle, Oriol Vinyals, and Tara Sainath, editors, *ICLR*, 2016.