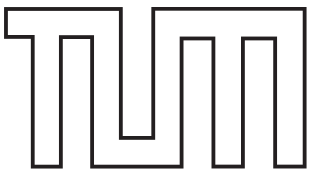


Technische Universität München
Chair of Media Technology
Prof. Dr.-Ing. Eckehard Steinbach

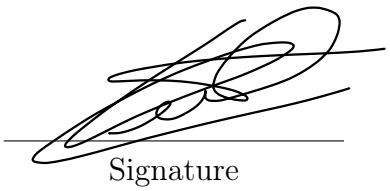
Bachelor's Thesis

Design of a Grasp Metric for Robotic Task-oriented Grasping Based on Human Demonstrations and Point Cloud Processing

Author:	Kevin Zien Qu
Matriculation Number:	03730587
Address:	Felsennelkenanger 21 80937 München
Advisor:	Hasan Furkan Kaynar
Begin:	02.11.2022
End:	22.03.2023

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, March 21, 2023
Place, Date


Signature

München, March 21, 2023
Place, Date


Signature

Kurzfassung

Roboter könnten unsere Lebensqualität verbessern, indem sie uns in unserem Zuhause unterstützen und bei alltäglichen Aufgaben helfen. Allerdings stellt die Fähigkeit, Haushaltsgegenstände zu greifen und zu manipulieren, eine enorme Herausforderung für einen Roboter dar, da es eine große Anzahl von Gegenständen sowie möglichen Aufgaben gibt und die Haushaltsumgebung von unstrukturierter Natur ist. Vor allem ist es wichtig, dass der Roboter ein Objekt so ergreift, dass es anschließend für eine gewünschte Aufgabe verwendet werden kann (sogennantes aufgabenorientiertes Greifen). In dieser Arbeit wird eine Greifmetrik für aufgabenorientiertes Greifen eingeführt, die angibt, wie geeignet ein vorgeschlagener Greifkandidat für eine bestimmte Aufgabe ist. Zu diesem Zweck wird eine Punktwolkenverarbeitungspipeline entwickelt, die die Ähnlichkeit zwischen dem vorgeschlagenen Griffkandidaten und einer Reihe von menschlichen Demonstrationsgriffen schätzt. Die menschlichen Demonstrationen werden über eine Teleoperations-Benutzerschnittstelle gewonnen und dienen als Basis für aufgabenorientierte Griffe. Auf der Grundlage der geschätzten Ähnlichkeit wird die Metrik für die Aufgabeneignung berechnet. Die experimentelle Auswertung zeigt, dass die eingeführte Eignungsmetrik eine starke Beschreibungskraft hat und sehr genaue Ergebnisse für Anwendungen wie die Rangfolge und Aufgabenklassifizierung von Griffen auf der Grundlage der berechneten Metrikwerte liefert.

Abstract

Robots could improve our quality of life by assisting us in our homes and helping with everyday tasks. However, the key ability to grasp and manipulate household objects poses an enormous challenge to a robot due to the large number of objects, possible tasks and the unstructured nature of the household environment. Most importantly, it is essential that the robot grasps an object in such a way that it can subsequently be used for a desired task (i.e. task-oriented grasping). This thesis introduces a grasp metric for task-oriented grasping which indicates how suitable a proposed candidate grasp is for a given task. For this purpose, a point cloud processing pipeline is developed that estimates the similarity between the proposed candidate grasp and a set of human demonstration grasps. The human demonstrations are obtained via a teleoperation user interface and serve as ground-truth task-oriented grasps. Based on the estimated similarity, the task-suitability metric score is computed. The experimental evaluation shows that the introduced suitability metric has strong descriptive power and provides highly accurate results for applications such as ranking and task classification of grasps based on the computed metric scores.

Contents

Chapter 1

Introduction

Robots are versatile machines that have the potential to improve many aspects of our lives. With the rapid advances in robotics research, the capabilities of a robot and its possible application areas have greatly expanded in recent years. In many industries, such as manufacturing or agriculture, robots already play an essential role in completing a variety of tasks in a safe and efficient way.

In households, robots could support and help people (e.g. the elderly with limited dexterity) with everyday tasks. However, robots are currently used there much less frequently due to their lack of reliability. While most household tasks appear straightforward and easy for humans, many of them are too complex for robots. In particular, grasping and manipulating an object, one of the key tasks for household service robots, remains a challenge [XKZD09]. Industrial robots are excellent at repetitively grasping and moving the same object because the object's shape, size, weight and position are well-known beforehand and the robot's motion can therefore be pre-programmed. Industrial robots are normally application specific, isolated from human workforce and possess their own structured working space [SAI20]. In contrast, household environments are highly unstructured and the robot has to have more flexible skills as the objects often are very close to or on top of each other, which makes perception and successful grasping even more difficult. Moreover, the objects can differ significantly in their physical properties and functionalities.

Another challenge for household service robots is the fact that facilitating a stable grasp itself is not sufficient in most cases. When planning a grasp, the robot must also consider the affordance of the object and the post-grasp action (i.e. the task) that should be executed with it. The concept of "affordance" was first introduced by Gibson [Gib79] who described how features (i.e. parts) of an object may be linked to a functional goal. Based on this affordance information and the desired task, the robot has to select a suitable grasp (so-called semantic or task-oriented grasping [LDC20]). For example, a tea mug has a handle (affords holding) and a bowl-like shaped body (affords containing liquid). Possible tasks that a user/robot could perform are carrying the mug or pouring out of it. If the

desired task is pouring out, the robot should not grasp by the rim since this would most probably impede the pouring process (the liquid would spill onto the gripper). In this particular example, a grasp by the handle would be much more suitable. Understanding object affordances and incorporating task orientation into the grasp planning process are therefore essential for an autonomous robot to successfully interact with objects and assist humans in various daily tasks [DNR18].

A question that now arises is: Given the desired task to perform, how should the robot be able to autonomously make the right decision on how and where to grasp an object? Since grasping is easy and comes naturally for humans, it would be ideal to incorporate human knowledge into the grasping process. More specifically, the human could demonstrate how to grasp the object for a given task, and in this way, teach the robot new skills. This "learning from human demonstration" approach has the great advantage that even non-experts with limited robotics knowledge are able to teach proper grasps and that explicit and tedious programming can be minimized or eliminated [BCDS08].

Most state-of-the-art methods for task-oriented grasping use deep learning approaches that show reasonable performance for a few specific object classes. However, their major drawback is the fact that they require enormous amounts of training data in order to generalize to the large number of different objects and possible tasks in the household environment. Data acquisition may require human annotators and is usually tedious and time-consuming. Moreover, adding new objects and tasks to an existing model is not straightforward as the model has to be re-trained from scratch.

In this thesis, an analytical approach is presented to generate a suitability metric for task-oriented grasping based on human demonstrations. The computed metric score indicates the task-suitability of a candidate grasp for a given task. A point cloud processing pipeline is developed to estimate the similarity between a proposed candidate grasp and a set of previously recorded human demonstration grasps. The human demonstrations serve as ground-truth task-oriented grasps and are obtained through a robotic teleoperation user interface that allows operation by non-experts due to its simplicity. Based on the estimated similarity, the suitability metric is computed which is the final output of the pipeline and can be further used for several applications. For instance, it can be used for ranking candidate grasps in terms of their task-suitability or classifying grasps into certain task categories. The presented approach is learning-free and thus does not need to be trained on large-scale datasets. Moreover, it is object class-agnostic and can therefore generalize to all kinds of (household) objects and possible tasks. Additionally, the point cloud processing pipeline works with single-view partial point clouds and does not require a multi-view reconstruction.

The thesis is organized as follows: Chapter 2 reviews the background and related work in the fields of 3D point cloud processing and robotic grasping of household objects. The methodology, pipeline structure and metric design are presented in Chapter 3. In Chapter 4, an experimental evaluation is conducted. Finally, Chapter 5 summarizes the thesis with a conclusion and thoughts about future work.

Chapter 2

Background and Related Work

2.1 3D Point Cloud Processing

The representation and reconstruction of the physical world in three-dimensional (3D) space with accurate metric information is essential for robotic perception. A very prominent type of 3D representation is a point cloud, which is a discrete and unstructured set of data points in space described by their Cartesian coordinates (x, y, z). Point clouds are often converted into 3D surfaces or mesh models for further operations. However, further conversion to other 3D representations might induce errors or information loss. Therefore, it is preferable to work directly with point clouds as this type of data format can be obtained from the raw output of 3D data sensors.

This chapter presents the theoretical foundations of point cloud creation from RGB-D image data, the FPFH point cloud feature descriptor and the DBSCAN clustering algorithm. In addition, three different point cloud registration algorithms are explained.

2.1.1 Point Cloud Creation from RGB-D Images

In order to capture 3D metric information of scenes and objects, various types of data sensors can be used. While laser scanners such as LIDAR (Light Detection And Ranging) scanners provide high resolution and accuracy, their high acquisition costs prevent them from becoming a commodity available to a larger user base. In robotics research, RGB-D depth cameras are most frequently used thanks to their lightweight design and affordability [ZSG+18]. RGB-D cameras are sensing systems that capture RGB images along with per-pixel depth information at real-time rates. To obtain depth information, there are two main approaches: Triangulation and Time-of-Flight (ToF). Triangulation can be realized as a passive (stereo vision) or active (structured light) approach. While stereo vision computes the disparity between two images taken at different positions, structured light cameras project an infrared light pattern onto the scene and estimate the disparity by the perspective distortion of the pattern due to the varying scene depth. Time-of-flight

cameras, on the other hand, measure the time it takes for light emitted from an illumination unit to travel to an object and back to a detector.

After obtaining the depth data, every pixel in the RGB-D image can be converted to a point in 3D space. Doing this for every image pixel yields the point cloud of the scene/object. In order to achieve this, the camera's geometric properties (i.e. the intrinsic camera parameters) must be first determined. The intrinsic camera parameters include:

- Focal lengths f_x,f_y: distance between the lens and the film/sensor
- Optical centre c_x,c_y: principal point where the optical axis intersects with the image plane
- Skew s: skew between the axes of the image plane

The intrinsic camera parameters form the intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3x3}$

$$\mathbf{K} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

After determining the intrinsic matrix $\mathbf{K}$ and knowing that every RGB-D image pixel's depth d corresponds to the point's z-coordinate in 3D space, i.e. $d = z$, the 3D point coordinates can be easily computed by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = z\mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{2.2}$$

where $(x, y, z) \in \mathbb{R}^3$ are the Cartesian point coordinates in 3D space and $u, v \in \mathbb{N}_{\nu}^{+}$ are the pixel coordinates in the depth image.

Since $\mathbf{K}$ is an upper triangular matrix, there exists an analytical solution for its inverse:

$$\mathbf{K^{-1}} = \begin{pmatrix} 1/f_x & -s/(f_x f_y) & (sc_y - c_x f_y)/(f_x f_y) \\ 0 & 1/f_y & -c_y/f_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

And plugging in the analytical solution for $\mathbf{K}^{-}1$ into Equation 2.2, we obtain:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = z \begin{pmatrix} 1/f_x & -s/(f_x f_y) & (sc_y - c_x f_y)/(f_x f_y) \\ 0 & 1/f_y & -c_y/f_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{2.4}$$

2.1.2 FPFH Point Cloud Feature Descriptor

Fast Point Feature Histograms (FPFH) is a multi-dimensional feature descriptor that characterizes the local geometry around a point p in a 3D point cloud. FPFH was first introduced by Rasu et al. [RBB09] and is heavily based on their previously proposed feature descriptor Point Feature Histograms (PFH) [RBMB08]. FPFH serves as a run-time optimized version of PFH as it drastically reduces the computational complexity from $O(n \cdot k^2)$ to $O(n \cdot k)$ (where n is the number of points in the point cloud P and k is the number of neighbors for each point p in P) by reformulating mathematical terms and caching previously computed values.

In order to understand the principle of FPFH, it is important to first look at the working mechanism of PFH. The computation of a PFH feature at a query point p is based on certain geometric relations between points in its k-neighborhood (points whose distance to the query point p is less than a given radius r) and their estimated surface normals. PFH features are thus pose-invariant features that represent the underlying surface model properties at a point p. The computation works as follows:

(1) For each point $\mathbf{p} \in \mathbb{R}^3$, all of $\mathbf{p}$'s neighbors enclosed in the sphere with a given radius r are selected (k-neighborhood)

(2) For every pair of points $\mathbf{p}_i$ and $\mathbf{p}_j$ $(i \neq j)$ in the k-neighborhood of $\mathbf{p}$, the relative difference between the two points and their associated surface normals $\mathbf{n}_i$ and $\mathbf{n}_j \in \mathbb{R}^3$ should be computed. To do this, a fixed Darboux uvw reference coordinate frame is first defined at one of the points (see Figure 2.1). The coordinate axes are determined by

$$\mathbf{u} = \mathbf{n}_i \tag{2.5}$$

$$\mathbf{v} = \mathbf{u} \times \frac{(\mathbf{p}_j - \mathbf{p}_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|} \tag{2.6}$$

$$\mathbf{w} = \mathbf{u} \times \mathbf{v} \tag{2.7}$$

(3) Using the uvw frame, the Euclidean distance between the points $\mathbf{p}_i$ and $\mathbf{p}_j$ and the angular variations between their point normals $\mathbf{n}_i$ and $\mathbf{n}_j$ can be computed:

$$\alpha = \mathbf{v}^T \cdot \mathbf{n}_j \tag{2.8}$$

$$\phi = \mathbf{u}^T \cdot \frac{(\mathbf{p_j} - \mathbf{p}_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|} \tag{2.9}$$

$$\theta = \arctan\left(\mathbf{w}^T \cdot \mathbf{n}_j, \mathbf{u}^T \cdot \mathbf{n}_j\right) \tag{2.10}$$

$$d = \|\mathbf{p}_j - \mathbf{p}_i\| \tag{2.11}$$

This quadruplet $\langle \alpha, \phi, \theta, d \rangle$ is computed for each pair of two points in the k-neighborhood.

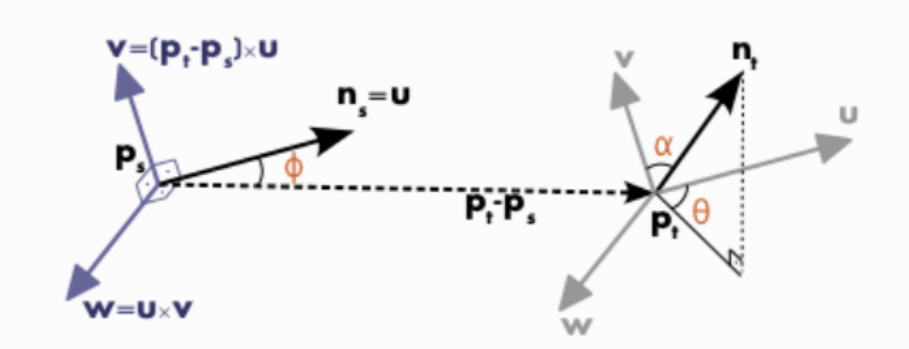


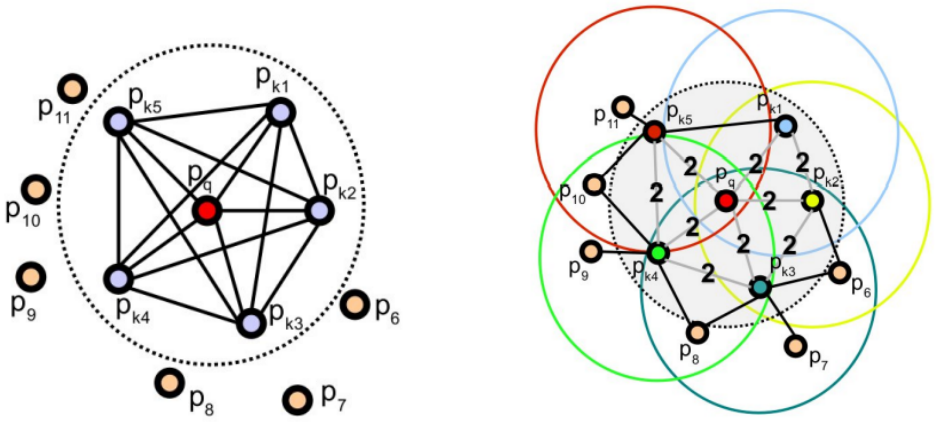
Figure 2.1: Darboux uvw reference coordinate frame defined at point $\mathbf{p}_s$. Note that the two example points in this figure are defined as $\mathbf{p}_s$ and $\mathbf{p}_t$ (instead of $\mathbf{p}_i$ and $\mathbf{p}_j$). Taken from [HIT+15].

(4) In the last step, the set of quadruplets $\langle \alpha, \phi, \theta, d \rangle$ is binned into a histogram. The binning process divides each feature's value range into b subdivisions and counts the number of occurrences in each subinterval to obtain the Point Feature Histogram (PFH) representation.

The computationally optimized version Fast Point Feature Histograms (FPFH) uses a simplified histogram feature computation. Different from FPH, where the set of quadruplets $\langle \alpha, \phi, \theta, d \rangle$ was computed for all pairs of points in the query point k-neighborhood, FPFH computes only the relationship between the query point p and its direct neighbors. The histogram obtained by this simplified computation is called Simplified Point Feature Histogram (SPFH). In the final step, the Fast Point Feature Histogram for a query point p (FPFH) can be computed by considering the SPFH values of its k neighbors to weight the final histogram of p:

$$FPFH(p) = SPF(p) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{w_k} \cdot SPF(p_k) \tag{2.12}$$

where the weight w_k represents the distance between the query point p and a neighboring point p_k in a metric space. As already mentioned above, the main difference between PFH and FPFH is that FPFH does not fully interconnect all k neighbors of the query point during computation (i.e. does not compute the geometric relationship between all pairs). The influence region diagrams in Figure 2.2 illustrate the two different approaches. Although the simplified FPFH computation is missing some value pairs which might contribute to capturing the geometry around p_q, FPFH features retain most of the discriminative power of the PFH.



(a) PFH influence region diagram (b) FPFH influence region diagram

Figure 2.2: Influence region diagrams for feature computation at a query point p_q (red). In the PFH computation (a), p_q and its k-neighbors are fully interconnected in a mesh. In the FPFH computation (b), the query point (red) is only connected to its direct k-neighbors (enclosed by the gray circle). Each direct neighbor itself is connected again to its own neighbors (enclosed by the colored circles) and the resulting histograms are weighted together with the histogram of the query point to form the FPFH. Taken from [RBB09]

2.1.3 DBSCAN Clustering

Clustering algorithms can process a set of data points (such as point clouds) in a way that similar points are grouped together. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [EKSX96] is a density-based clustering algorithm that can detect arbitrarily shaped clusters from a data set containing noise and outliers. It is based on the main idea that a point belongs to a cluster if it is in spatial proximity to other points in the cluster. There are two key parameters of DBSCAN:

- *eps*: the distance that specifies a neighborhood. Two points are considered to be neighbors if the distance between them is less than or equal to *eps*.
- *minPts*: minimum number of data points to define a cluster.

Based on these two parameters, every point in the data set can be classified as a core point, border point or outlier which is illustrated in Figure 2.3. They are defined as follows:

- **Core point** has at least *minPts* number of points (including the point itself) in its neighborhood defined by the distance *eps*.
- **Border point** is within the neighborhood of a core point and has less than *minPts* number of points within its own neighborhood.
- **Outlier** is not a core point nor in the neighborhood of any other core points.

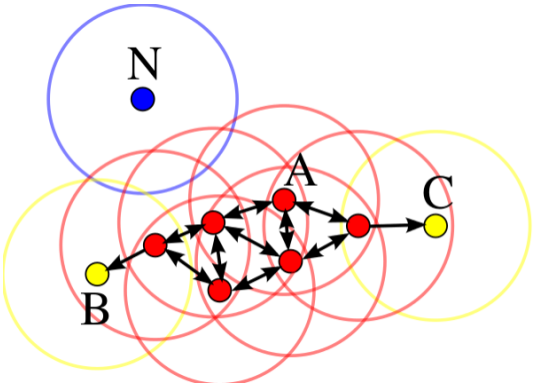


Figure 2.3: Core points (A, red), border points (B & C, yellow) and outlier (N, blue). The core points and border points form a cluster. The circles represent the neighborhood defined by the radius *eps* of the respective points. In this example, *minPts* is set to 4. Taken from [Yil20].

After the parameters are set, the algorithm proceeds by randomly selecting a point in the dataset. If there are at least *minPts* number of points within a radius of *eps* to the point, it is marked as a core point and a cluster formation starts. If not, the point is marked as outlier/noise. Once a cluster formation begins, all points in the neighborhood of the initial point become a part of that cluster. If these new points are also core points, their neighborhood points are also added to the cluster. The DBSCAN algorithm will continue to randomly select new points until all points have been visited. As a final result, clusters with a high density of points are separated from low-density regions. Figure 2.4 shows clustering results by DBSCAN on three different point sets.

Figure 2.4: DBSCAN clustering results on three different point sets. Each segmented cluster is assigned a different color. Taken from [EKSX96].

2.1.4 Point Cloud Registration

Point cloud registration (also known as point set registration) is a fundamental task in 3D computer vision with extensive applications in robotics, such as object detection, pose estimation and simultaneous localization and mapping (SLAM). Given two or more sets of point clouds in different coordinate systems, the aim of registration is to find the spatial transformation (e.g. scaling, rotation and translation) that best aligns them into a common coordinate system. Mathematically speaking, given two sets of 3D point clouds with n points $\mathbf{M}, \mathbf{S} \in \mathbb{R}^{nx3}$, the registration goal is to find the optimal transformation T^* such that the distance between the transformed point cloud $T(\mathbf{S})$ and the target point cloud $\mathbf{M}$ is minimized:

$$T^* = \arg\min_T ||T(\mathbf{S}) - \mathbf{M}||^2 \tag{2.13}$$

Although point cloud registration has been studied for a long time, there are still several challenges that can prevent a good alignment. For instance, partial or incomplete point clouds with little overlap are difficult to align [HGU+21]. A very common registration approach is to first find a coarse alignment using a global registration method and to then refine that coarse alignment using a local registration method. Global registration methods do not require an initial alignment but produce less tight results, while local methods rely on rough initial alignments to produce tighter results.

2.1.4.1 RANSAC-based Global Registration Method

Random Sample Consensus (RANSAC) is an iterative parameter estimation algorithm introduced by Fischler and Bolles [FB81] that can deal with a set of observed data points containing outliers. RANSAC uses a resampling technique to estimate the model parameters and generates candidate solutions by using only a small subset of points. The algorithm is summarized by [Der05] as follows:

Algorithm 1 RANSAC Algorithm

1: Select randomly the minimum number of points required to determine the model parameters.
2: Solve for the parameters of the model.
3: Determine how many points from the set of all points fit the proposed model with a predefined tolerance ϵ
4: If the fraction of the number of inliers over the total number of points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5: Otherwise, repeat steps 1 through 4 (maximum of N times).

The principle of the RANSAC algorithm and variations of it are also commonly used in global point cloud registration methods [AMB17]. In this thesis, a RANSAC-based global

registration method based on the Open3D (an open source library for 3D data processing) [ZPK18] implementation is used to coarsely align a source point cloud to a target point cloud. In the first step, FPFH features are computed for each point in the two point clouds. Then, in each RANSAC iteration, a predefined number of n random points is selected from the source point cloud. Their corresponding points (i.e. geometrically most similar points) in the target point cloud are determined by querying the nearest neighbor in the 33-dimensional FPFH feature space. After determining the sampled points and their correspondences, a transformation to align them is computed. The estimated transformation is then validated on the entire point cloud. The registration algorithm terminates if the iterations reach the predefined maximum iteration number N or if the fitness (measurement of overlap) between the two point clouds after transformation indicates that the algorithm can be terminated earlier with some predefined confidence p. An early termination takes place if the number of iterations reaches the value k:

$$k = \frac{\log(1-p)}{\log(1-f^n)} \tag{2.14}$$

where p is the confidence, f is the fitness and n is the number of sampled points in the RANSAC algorithm. The fitness measures the overlapping area and is defined as [# inlier correspondences / # of points in the target point cloud] (the higher the fitness, the better the registration result).

2.1.4.2 ICP Local Registration Method

The Iterative Closest Point (ICP) algorithm introduced by Besl and McKay [BM92] is the most widely used method for rigid point cloud registration due to its simplicity and low computational complexity. ICP is considered a local registration method as it requires a rough initial alignment (usually obtained by a global registration method) to tightly align two point clouds. Many variants and extensions of ICP have been proposed that involve all phases of the algorithm from point selection and matching up to the minimization strategy. The basic ICP variant of Besl and McKay [BM92] iteratively assigns point correspondences based on the closest distance criterion and finds the least-squares rigid transformation $\mathbf{T} \in \mathbb{R}^{4x4}$ relating the points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^4$ (homogenous coordinates) in the correspondence set K

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in K} ||\mathbf{p} - \mathbf{T}\mathbf{q}||^2 \tag{2.15}$$

where E is the objective function to be minimized with respect to the transformation matrix $\mathbf{T}$.

A very popular variant, the Point-to-Plane ICP, first introduced by Chen and Medioni [CM91], uses a different error metric. Instead of minimizing the sum of squared distances

between a transformed point $\mathbf{q}$ and its correspondence point $\mathbf{p}$ (point-to-point error metric) as proposed by Besl and Mckay (Equation 2.15), this variant minimizes the sum of squared distances between the transformed point $\mathbf{q}$ and the tangent plane at its correspondence point $\mathbf{p}$ (point-to-plane error metric)

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in K} ||(\mathbf{p} - \mathbf{Tq}) \cdot \mathbf{n}_p||^2 \tag{2.16}$$

where $\mathbf{n}_p \in \mathbb{R}^3$ is the normal vector of point p. Rusinkiewicz and Levoy [RL01] have shown that the point-to-plane error metric has a substantially faster convergence speed than the point-to-point metric. Figure 2.5 geometrically visualizes the two different error metrics.

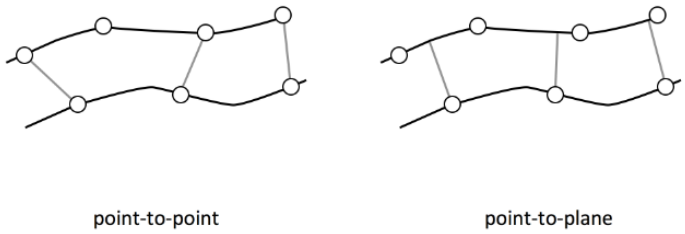


Figure 2.5: Geometric visualization of the point-to-point (left) and point-to-plane error metric (right). Taken from [Sta20].

2.1.4.3 CPD Registration Method

The Coherent Point Drift (CPD) algorithm proposed by Myronenko and Song [MS10] uses a probabilistic approach to register two sets of point clouds. The CPD algorithm is based on the idea of modeling one point cloud as Gaussian Mixture Model (GMM) centroids and then finding a transformation that maximizes the likelihood of the other point cloud under this mixture model. Crucial for this method is to force the GMM centroids to move coherently as a group to preserve the topological structure of the point clouds. The CPD algorithm can be used for affine, rigid and non-rigid registrations.

The CPD algorithm works in two steps:

(1) **Initialization**: In this step, the algorithm computes an initial transformation between the two point sets. This can be done using an initial guess, such as the identity transformation, or by using a feature-based method to find correspondences between the two sets.

(2) **Iterative refinement**: In this step, the EM optimization algorithm [DLR77] is applied to iteratively update the mixture model parameters and the transformation until convergence is reached. At each iteration, the algorithm computes the posterior

probability of each point belonging to each Gaussian component in the mixture model and then uses these probabilities to update the mean and covariance of each Gaussian component and the transformation between the two point sets.

As an example for the rigid registration case: given two sets of 3D point clouds with n points $\mathbf{M}, \mathbf{S} \in \mathbb{R}^{nx3}$ and the registration goal to find a rigid transformation such that $\mathbf{S}$ aligns to $\mathbf{M}$, the transformed point cloud $T(\mathbf{S})$ is:

$$T(\mathbf{S}) = s\mathbf{S}\mathbf{R}^T + 1\mathbf{t}^T \tag{2.17}$$

where $s \in \mathbb{R}$ is the scale, $\mathbf{R} \in \mathbb{R}^{3x3}$ is the rotation matrix and $\mathbf{t} \in \mathbb{R}^{3x1}$ is the translation vector.

2.2 Robotic Grasping of Household Objects

With the desire and demand for household service robots that can support and assist humans in their daily lives, significant research efforts have been invested into the key abilities of grasping and manipulation of household objects. Households are particularly challenging for robots as the environments are highly unstructured and contain a wide variety of objects with diverse physical properties and functionalities. For many household objects such as kitchen tools, the function of the tool and the desired task that should be executed with it must be considered when planning a grasp. Simply selecting a random stable grasp is often a poor decision as the grasp could interfere with the post-grasp action. Consequently, the robot must not only detect and classify an object but also "understand" its functionality to maximize the task's success.

This chapter introduces the concept of object affordances, which is an essential concept for understanding task-oriented grasping. Following this, state-of-the-art methods for task-oriented grasping and the learning from human demonstration approach are discussed.

2.2.1 Object Affordances

The ability to understand object affordances has been studied for a long time [Gib79, Mas90, NN$^+$04]. The American ecological psychologist James Jerome Gibson [Gib79] was the first to introduce the concept of affordance and argued that the goal of perception is primarily to understand the environment and act in it. He famously claimed that "to see things is to perceive what they afford". In contrast to other visual or physical properties, which usually describe only the object itself, affordances indicate functional interactions of object parts with humans [DNR18]. In essence, an affordance is a potential action that a user can perform in relation to an object. In Figure 2.6, affordance examples for three objects and their object parts are illustrated. When having to choose between different objects for a given task, it is thus important to know which one possesses an affordance that is needed for the specific task. For instance, when cutting something, one would choose an object that has a sharp part that affords "cutting" (e.g. the blade of a knife or scissors). Describing it in simple terms, an affordance facilitates the decision on "what" to grasp.

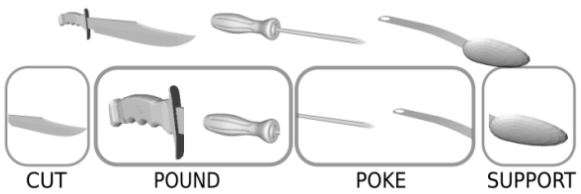


Figure 2.6: Top row: knife, screwdriver and spoon. Bottom row: detected affordances of their object parts. Taken from [KSHK17].

Figure 2.7: Object class and affordance detection on RGB images with AffordanceNet [DNR18]. The different colors indicate specific affordances of the object parts.

In robotics, some approaches followed the principle of interactive perception to explore object affordances. Interactive perception conducts forceful interaction with the environment to simplify and enhance perception [BHS+17]. Similar to biological systems, this creates rich and informative sensory signals that are concurrent with the actions. For example, a robot can understand the functionality of a certain tool by randomly experimenting, i.e., interacting with the tool on other objects and observing the effects of its actions [Sto05]. Although this is very intuitive, learning the affordances through experimentation is quite time-consuming. Moreover, some affordances might remain unexplored if the robot does not perform certain actions. Consequently, this interactive approach alone does not generalize well to novel objects and actions on a larger scale.

More recently, with the rapid advances in computer vision, numerous vision-based approaches to detect object affordances were introduced. Do et al. [DNR18] proposed the deep learning-based network "AffordanceNet" that simultaneously detects objects and their affordances from single RGB images. They first detect the object and localize it by putting a rectangular bounding box around it. Inside the bounding box, the affordances are encoded at every pixel of the object, i.e., a group of pixels that shares the same object functionality is assigned the same affordance. The detection results are shown in Figure 2.7. The bounding box indicates the class of the object (hammer, cup or bowl), while the color of the object part represents its affordance (e.g. blue hammer handle affords holding, brown hammer top part affords pounding). Similarly, Thermos et al. [TDP20] proposed a deep learning approach that infers pixel-wise affordance predictions from 2D data (i.e. images and video sequences). Different from AffordanceNet, their model surpasses the need for object-related information such as object labels and bounding boxes. Although the two approaches show promising results, their disadvantage is that detection and affordance prediction are output in 2D form while a robot perceives and operates in a 3D world.

2.2.2 Task-oriented Grasp Estimation

After determining the object affordances and thus answering the question of "what" to grasp, the task itself and therefore the question of "where" to grasp has to be considered too before executing the grasp. This leads us to the concept of task-oriented grasping (or semantic grasping) which addresses the problem of selecting stable grasps that are functionally suitable for specific object manipulation tasks [DA12]. The grasp location (i.e. the robot gripper's contact point on the object) has to be correctly chosen for the successful execution of a given task. Continuing the example at the beginning of Chapter 2.2.1: given the task "cutting", we could determine that knives or scissors are most suitable for this task as they both possess the affordance "cutting". If we now, for instance, select the knife to cut open a paper box, it is necessary to grasp in a task-oriented manner. Since only the sharp blade can be used to cut properly, the robot should grasp the handle and not block the blade with its gripper.

While general robotic grasping has experienced enormous progress in recent years, task-oriented grasping has not advanced as far and researchers have been working on solutions to bridge the gap. Dang and Allen [DA12] presented an example-based planning framework to generate task-specific (semantic) grasps. They use a semantic affordance map that links local geometry to a set of predefined semantic grasps suitable for different tasks. The semantic affordance map for each object class is built using a representative object of that class. Given the map, the pose of a robot hand and its most similar predefined grasp with respect to an object can be estimated for a particular task. However, their method is limited in its capability to deal with objects with large shape variances within the class. Detry et al. [DPM17] introduced a task-oriented grasp model consisting of two components: a geometric model and a semantic model. From a depth image as input, the geometric model computes a distribution of 6D grasp poses for which the gripper shape matches the object surface. The model relies on a dictionary of grasp prototypes which consists of object shapes and corresponding grasp parameters. When observing a new object, the dictionary is matched against the visual input and the grasp parameters from the dictionary are transferred to the new object. The semantic model uses Convolutional Neural Networks (CNNs) to partition the scene into areas that are task-compatible. In order to create training data for the semantic model, they manually label images of synthetic objects by assigning a binary value (suitable/unsuitable) for each appropriate task.

To avoid manual annotation, Fang et al. [FZG+20] developed a simulated self-supervision framework to automatically collect the training data. They leveraged a real-time physics simulator that allows a simulated robot to perform trial and error for the two tasks of "hammering" and "sweeping" in millions of trails. The grasps are then scored based on the task-execution success in each trial and recorded. In Figure 2.8, task-oriented grasps for "sweeping" and "hammering" are compared to a task-agnostic grasp which received a lower score for task success. Their task-oriented grasping network consists of a task-oriented grasping model and a manipulation policy that are coupled and jointly trained in a deep neural network using the recorded data. This presented procedure provides a natural way

Figure 2.8: Comparison between task-agnostic and task-oriented grasps. The task-agnostic grasp (left) can lift up the hammer but it is not suitable for the tasks of "sweeping" and "hammering". In order to maximize task success, different grasps (middle and right) are required. Taken from [FZG+20].

for data collection, however, given the complexities of designing manipulation tasks in simulation, it is still difficult to scale this approach to multiple tasks.

Liu et al. [LDC20] introduced the Context-Aware Grasping Engine (CAGE), which combines semantic representations of grasp contexts with a neural network structure to infer task-oriented grasps. First, the engine receives as input the point cloud of the object, a spectral reading from an SCiO sensor for material recognition and a label specifying the object state. Additionally, they used the affordance detection model "AffordanceNet" [DNR18] to extract part affordances. This information, together with the manually specified task, form the object's semantic representation. The engine's second input is a set of grasp candidates obtained from an antipodal grasp sampler. Each grasp gets assigned the affordance and material of the object part closest to the grasp as the grasp's semantic representation. They then apply a neural network based on the Wide & Deep model [CKH+16] as a reasoning module to infer the relations between the extracted semantic representations of object and grasps. The final output is a ranking of all grasp candidates ordered by their suitability for the given context. Figure 2.9 shows an exemplary execution of the highest-ranked grasp based on the following context information: pouring task, ceramic object material, handle part affordance and hot object state.

Murali et. al [MLM+21] have made two contributions to the field of task-oriented grasping. The first contribution was the creation of the "TaskGrasp" dataset which is not only larger in magnitude but also significantly more diverse in terms of objects and tasks than previously published datasets. TaskGrasp contains 250K task-oriented grasps in total for 56 tasks and 191 household & kitchen objects from 75 object categories along with their

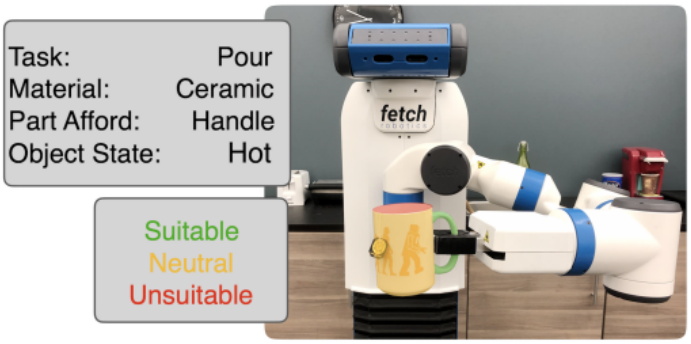


Figure 2.9: Most suitable grasp selected by CAGE [LDC20] based on the context information.

RGB-D information. During dataset collection, each object was captured with an RGB-D camera from multiple viewpoints. The resulting point clouds were then registered with the iterative closest point algorithm to obtain a more complete object model. Then, 600 object-centric stable grasps were sampled on each object point cloud. In order to receive labels for the grasps in terms of their suitability or non-suitability for a specific task, the authors utilized crowdsourcing on an online platform where regular people were paid to carry out the data annotation. The authors' second contribution was the "GCNGrasp" framework, which is based on a Graph Convolutional Network (GCN) that evaluates the task-suitability of candidate grasps from geometric observations and semantic knowledge. First, grasp and object point cloud are encoded by an encoder built on a PointNet++ architecture [QYSG17]. Then, the semantic relationship between object category, task and hierarchies is encoded in a knowledge graph. The graph is constructed from the relationship between task & object class in the TaskGrasp dataset and the object hierarchy from the large lexical database WordNet [Mil95], which groups words into sets of cognitive synonyms (synsets). Following this, a Graph Convolutional Network takes the encoded information and knowledge graph as input and outputs node-level embeddings. Finally, a grasp evaluator utilizes the node embeddings to predict a final grasp score for a given task. In Figure 2.10, the grasp scores for all stable candidate grasps are visualized (low scores in red, high scores in green).

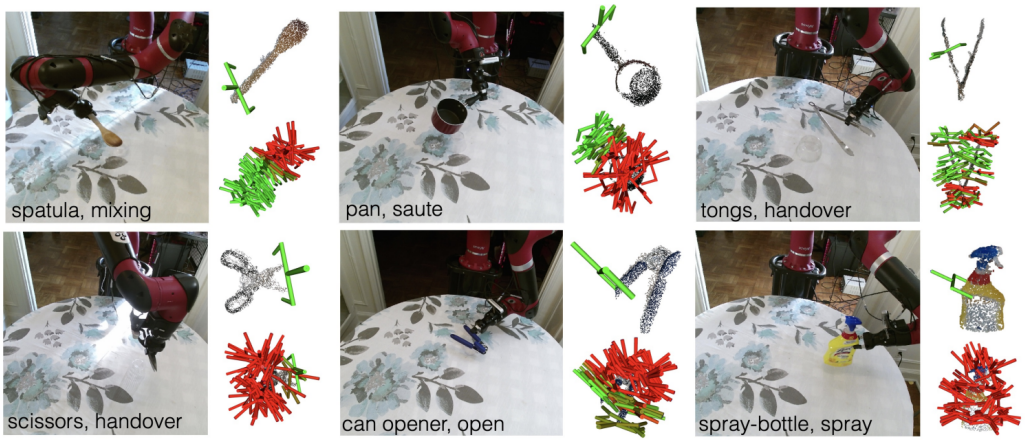


Figure 2.10: Task-oriented grasp evaluation by GCNGrasp [MLM+21]. For each execution, the top 3D visualization shows the grasp that was executed (which had the best evaluation score) and the bottom shows all the stable grasp candidates colored by their scores (green means higher score).

2.2.3 Learning from Human Demonstration

In the field of robotics and automation, learning from human demonstration, or often just called learning from demonstration (LfD), is the concept of robots acquiring new skills by learning to imitate a demonstrator [BCDS08, ACVB09]. An advantageous property of LfD is that even non-experts who are not proficient with robot programming can execute demonstrations and in this way, teach a robot. Demonstrations can be carried out through, for instance, robotic teleoperation, in virtual reality or in real life. Compared to other robot learning methods, LfD is particularly compelling in cases where the desired behaviour cannot be specified in a script or where it cannot be simply defined as an optimization of a known reward function (as it is done in reinforcement learning) [RPCB20]. A possible disadvantage of the LfD technique is that learning only from demonstrations could limit the performance to the teacher's skills. If the teacher him/herself is not able to demonstrate correctly, the robot's performance will also suffer from this. However, in the context of grasping and manipulating household objects in a task-oriented manner, humans are ideal teachers due to their lifelong built-up experience, superior cognitive abilities and what they perceive as the low difficulty of the tasks.

Aleotti and Caselli [AC11] proposed a part-based grasp planning method learned from human demonstrations. In their method, an interactive 3D virtual reality environment including a data glove is used in which the user can perform multiple demonstrations of the same task. After the demonstrations are captured, a grasp map and a task precedence graph (TPG) augmented with information about the parts of the objects that should be grasped are created. They also present a 3D shape retrieval algorithm based upon the Reeb graph data structure and for each object class in the database, a prototype Reeb graph with annotated parts is predefined. When observing a new object, the shape retrieval algorithm

classifies the object as a member of a particular object class and automatically labels the object parts according to the predefined Reeb graph. In the robotic manipulation phase, a manipulation planner then selects candidate grasps from the object grasp map, which contains sampled stable grasps for each object part of interest.

Kokic et al. [KSHK17] presented a convolutional neural network approach for learning task-oriented grasping from real-world human activity datasets. The dataset contains RGB images of humans holding and manipulating various household objects. To first process the dataset, a CNN is trained that predicts hand pose, object pose and object shape from an RGB image. When the CNN is then run on the whole human activity dataset, statistics on the relative hand poses with respect to the objects can be obtained. This yields distributions of task-oriented grasps on objects which can be used to teach a robot how to grasp novel objects of the same class in a task-oriented manner. The knowledge is transferred to the robot by training another CNN that receives an object point cloud, the desired task and a candidate grasp as input and outputs a prediction of whether the candidate grasp is suitable for the given task or not. Figure 2.11 shows the pipeline from processing the human activity dataset to a real robot executing the most suitable grasp.

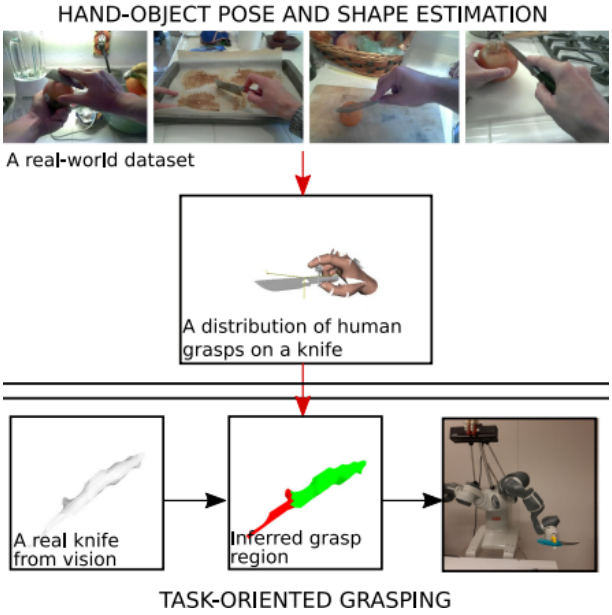


Figure 2.11: Top to bottom: RGB images of human grasps on knives for the task cutting from the real-world dataset; the distribution of human grasps on a knife as a result of processing RGB images; a knife recorded with an RGB-D camera, suitable grasping area, a task-oriented grasp with a robot. Taken from [KSHK17].

Simeonov et al. [SDT+22] introduced Neural Descriptor Fields (NDFs), an object representation that encodes both points and relative poses between an object and a target via category-level descriptors. With only a small number ($5 \sim 10$) of human demonstrations guiding a robot arm, the NDF generalizes the demonstrated manipulation task (here: pick

and place task) to novel object instances of the same object class in any 6-DoF configuration. NDFs are trained self-supervised for the surrogate task of 3D reconstruction, do not require labeled key points, and are SE(3)-equivariant, guaranteeing generalization to unseen object configurations. They first use a neural network to represent an object point cloud P as a continuous function $f(x|P)$ that maps any 3D coordinate x to a spatial point descriptor that encodes the relationship between x and the object's salient geometric features. Then, a local coordinate frame with a rigid set of query points is introduced. The configuration of these points is represented as an SE(3) pose with respect to a canonical pose in the world frame. For each object instance in the demonstration, the query point set is converted into a set of feature descriptors by concatenating point descriptors of all the points. The feature representation resulting from the evaluation of query points at different SE(3) configurations forms the pose descriptor field. When planning a grasp on a new object instance, they estimate the correct gripper pose by optimizing for the SE(3) transformation of the query point set that minimizes the difference between the descriptors of demonstration and test pose. A major drawback of the method is the amount of data that is needed to train the neural network. For training, a dataset of 100,000 objects was generated consisting only of 3 object categories (mugs, bowls, and bottles) at random tabletop poses. Considering the large number of different object categories in a household environment, their method would need a huge amount of data to generalize to them all.

Chapter 3

Methodology

3.1 Pipeline Overview

This chapter presents an overview of the pipeline for generating a grasp metric for task-oriented grasping. Based on similarity estimation to human demonstration grasps through point cloud processing, a candidate grasp's task-suitability metric score can be computed. The pipeline structure diagram is illustrated in Figure 3.1. First, given a task, multiple human demonstrations of task-oriented grasps on a household object are obtained through a teleoperation user interface. For every demonstrated grasp, the object point cloud of the object to be grasped is first isolated (1) and then transformed into gripper frame coordinates (2). Doing this for every demonstration results in a set of isolated and transformed object point clouds. This process can be repeatedly carried out for various objects and tasks (every object and task will result in its own set of isolated and transformed demonstration point clouds) and is referred to as the human demonstration phase. After finishing this phase, the demonstrations can be used to estimate the task-suitability metric score of a novel candidate grasp. When a candidate grasp is proposed, its object point cloud is again isolated (3) and transformed into gripper frame coordinates (4). Then, the candidate object point cloud is pairwise registered to all demonstration object point clouds in the set for the given object and task (5). Based on the registration results, the similarity between the candidate and the demonstrations is estimated. As the demonstrations serve as task-oriented ground-truth grasps for a given task, high similarity indicates high task-suitability of the candidate grasp. The estimated similarity is then used to compute a task-suitability metric score (6) which is the final output of the pipeline. The presented approach is learning-free and does not require any training as it solely uses an analytical analysis to compare the candidate grasp to the demonstrations. Moreover, the pipeline is object class-agnostic, and thus expanding it for new objects and tasks can be easily achieved by simply collecting new demonstrations in the human demonstration phase. Moreover, the objects do not have to be captured from multiple viewpoints and reconstructed into a 3D model as the pipeline operates on single-view partial point clouds. Note that our

approach does not necessarily need human demonstrations to compute the task-suitability metric score. Instead of human demonstrations, any task-oriented grasps (e.g. from a task-oriented grasping dataset) can be used as ground-truth grasps for a given task.

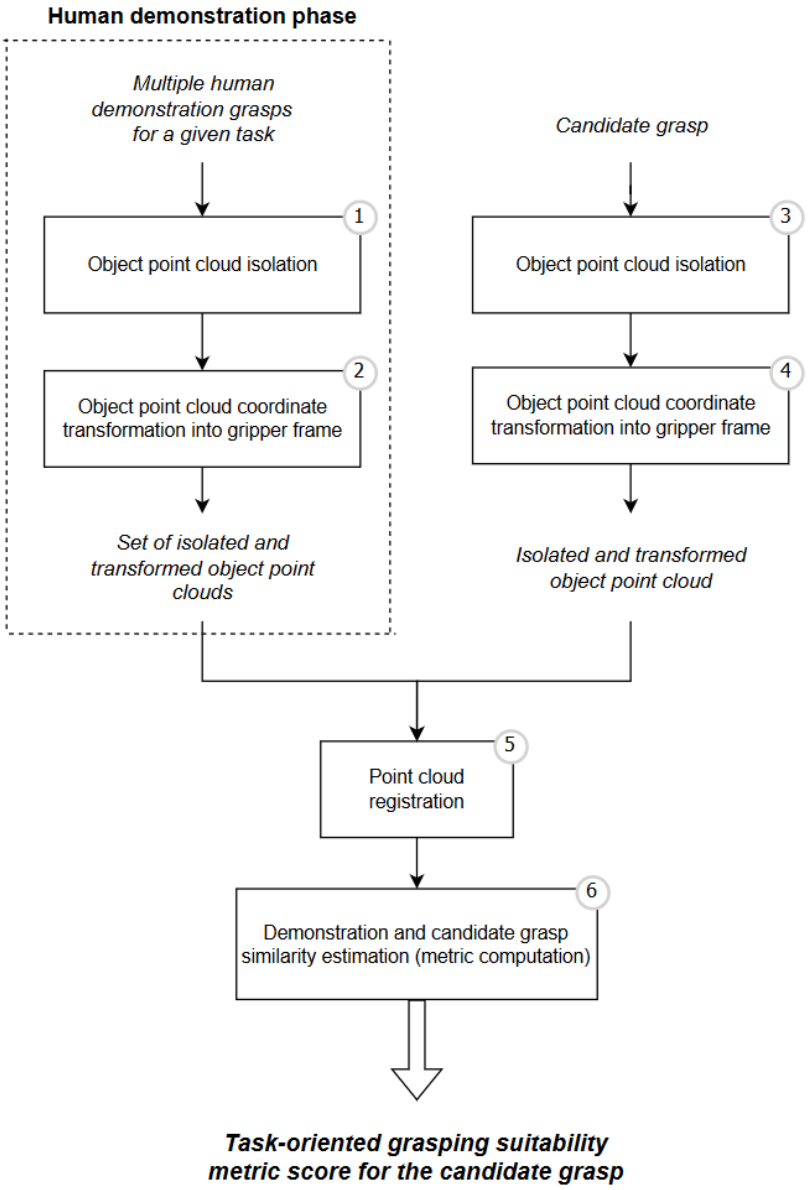
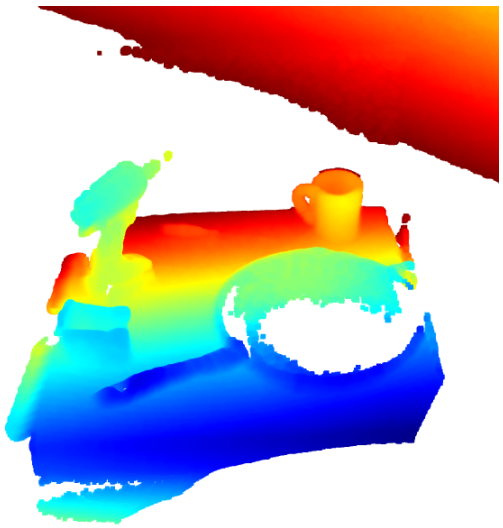


Figure 3.1: Pipeline structure diagram. Given the task, the pipeline will output a metric score that indicates the task-suitability of a proposed candidate grasp.

3.2 Scene Capturing and Point Cloud Creation

The scenes are captured with an Azure Kinect RGB-D camera which is mounted on a tripod and positioned in front of a table. The scenes contain various household objects that are placed on the table surface. Object positions are randomly set with the condition that the objects should not occlude each other from the camera's point of view. The scene capturing setup is shown in Figure 3.2a. The corresponding 3D scene point cloud is then created using the captured depth data and shown in Figure 3.2b. As only one camera from a single viewpoint is used to capture the scene, the resulting point cloud is not complete but partial. Moreover, imperfect perception and camera noise can lead to missing depth values (as observed on the inside of the pan).

(a) RGB-D camera (top right) mounted on a tripod and capturing the scene.



(b) Corresponding 3D scene point cloud created from the captured depth data.

Figure 3.2: Scene capturing setup and corresponding scene point cloud.

3.3 Obtaining Grasps from Human Demonstration

Given a specific task, multiple human task-oriented demonstration grasps should be collected. In the context of grasping and manipulating household objects in a task-oriented manner, humans are excellent demonstrators/teachers due to their lifelong built-up experience, superior cognitive abilities and what they perceive as the low difficulty of the tasks. Therefore, the collected demonstrations serve as ideal (i.e. ground-truth) task-oriented grasps.

The grasp demonstrations are collected through a remote robot teleoperation user interface. The interface used in this thesis was created by Lejiri [Lej21] in the Unity game-engine environment and is based on the idea from Kent et. al [KSC20]. By using the interface,

a human teleoperator can guide the robot gripper and choose what and how the robot should grasp. The procedure to record a grasp is as follows:

(1) **Scene capturing**: The scene containing the object to be grasped is captured by the RGB-D camera and displayed on the interface.

(2) **Grasp contact point selection**: The user selects the grasp contact point (i.e. the point where the gripper should grasp the object) by clicking on the desired position. Once the grasp contact point is selected, a sphere is rendered around it.

(3) **Gripper pose selection**: The user can then set the desired gripper pose (i.e. the approach direction/angle). The gripper is displayed on the sphere and can be moved around on the sphere surface by rotating a joystick on the interface. Once the desired pose is reached, the grasp is saved.

Figure 3.3, shows the teleoperation user interface in the Unity game engine environment.

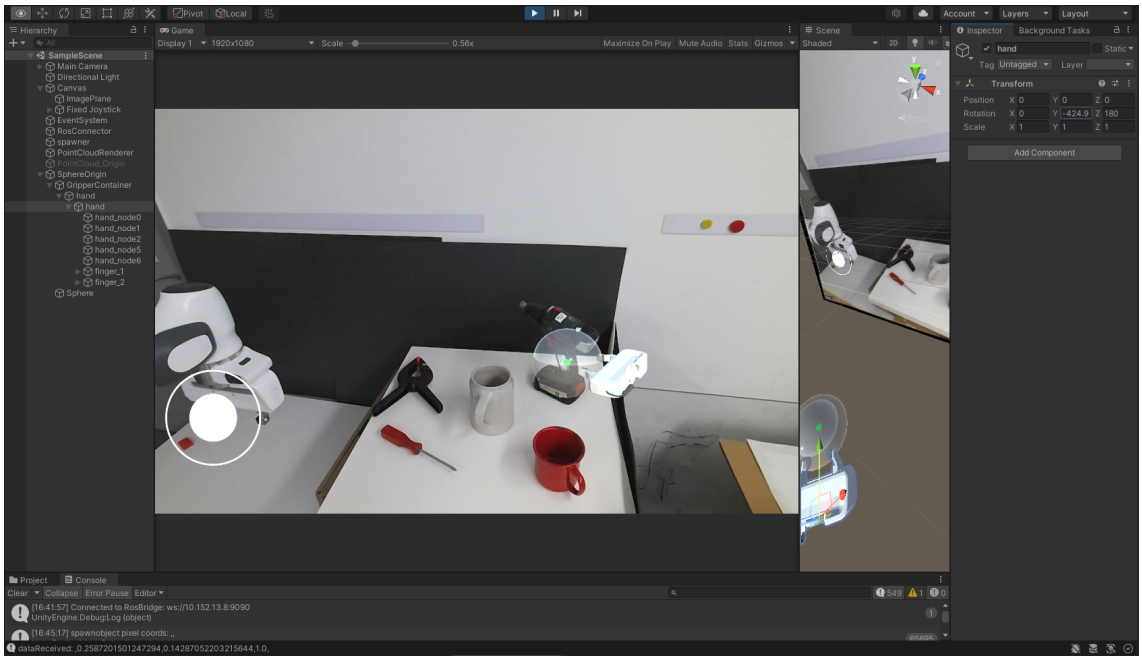


Figure 3.3: Teleoperation user interface: after selecting the grasp contact point on the drill machine (green point), the gripper pose can be specified by rotating the joystick (bottom left) to move the gripper around on the white sphere.

For each object and task, n demonstrations are collected which form the demonstration set $D = \{d_1, \ldots, d_n\}$. The set D is labeled with the object and task name. Between every few demonstrations in a set, the object is moved and rotated to obtain different object poses. However, the poses are chosen so that the object part to be grasped is always visible from the camera's point of view.

For every recorded demonstration grasp, the following information is saved:

- **Scene point cloud**: the scene point cloud containing the object to be grasped.
- **Gripper Pose**: the gripper pose for the demonstration grasp as $4x4$ matrix
- **Grasp contact point**: the grasp contact point which represents the 3D point cloud coordinates of the point where the robot's gripper grasps the object.

3.4 Point Cloud Pre-Processing

Since we are interested in the object to be grasped and its physical properties (e.g. size and shape), it is necessary to obtain the object point cloud by isolating it from the rest of the scene. Furthermore, we want to incorporate the grasp information (i.e. information about gripper pose and grasp contact point) into the object point cloud. Thus, spatial transformations to the object point cloud are applied. For every grasp (either demonstration or candidate grasp) and its corresponding grasp information, the object point cloud is isolated and then transformed.

This chapter describes the point cloud processing steps to obtain the isolated and transformed object point cloud.

3.4.1 Object Point Cloud Isolation

In order to obtain the isolated point cloud of the object to be grasped, multiple operations on the scene point cloud are executed. The whole isolation process is visualized in Figure 3.4 and explained in the following:

(1) **Sphere isolation**: firstly, a radius r is defined. Given a (proposed) grasp and the gripper's contact point on the object, all points with a distance to the contact point larger than the radius r are removed from the scene point cloud. This step serves as a coarse isolation of the object as the remaining points represent a sphere containing the object and the close neighborhood around it.

(2) **Plane removal**: secondly, the planar surface underneath the object is removed (assuming that the object is placed on a planar surface such as a table or the floor). To find the plane, a RANSAC-based plane segmentation algorithm is used. The algorithm receives three parameters as input:

- distance threshold: defines the maximum distance a point can have to an estimated plane to be considered a plane inlier
- number of sampled points: defines the number of points that are randomly sampled to estimate a plane
- iteration number: defines how often a random plane is sampled and verified.

Given the input parameters, the algorithm then returns the plane parameters $a, b, c, d \in \mathbb{R}$ such that each point $(x, y, z) \in \mathbb{R}^3$ on the plane fulfills the plane equa-

tion:

$$ax + by + cz + d = 0 \tag{3.1}$$

The plane points are then removed from the coarsely isolated point cloud from step 1.

(3) **Clustering**: next, the DBSCAN clustering algorithm is run on the point cloud resulting from step 2. The algorithm groups the points into a number of clusters (the number does not have to be predefined as the algorithm will determine it itself). To obtain the cluster that represents the object to be grasped, a query of the grasp contact point is executed and the cluster that contains the grasp contact point is determined as the object. All other clusters are discarded.

(4) **Camera Noise Removal**: as the final step, part of the camera noise is removed. Since the scene is captured in real-life with an RGB-D camera and not in a simulation environment, the resulting point cloud will contain some camera noise. Particularly at the rear sides of the isolated object point cloud, the camera noise can be observed (see Figure 3.5a). In order to remove a large part of this noise, a plane is inserted into the object point cloud that "cuts" through the object. Points that lie behind the plane are then discarded ((see Figure 3.5b). The plane equation parameters $a, b, c, d \in \mathbb{R}$ from Equation 3.1 and the plane normal vector $\mathbf{n} \in \mathbb{R}^3$ are computed as follows:

Given the camera position $\mathbf{p}_{camera} \in \mathbb{R}^3$ at the origin of the coordinate frame ($\mathbf{p}_{camera} = (0, 0, 0)^T$) and the grasp contact point $\mathbf{p}_{cp} \in \mathbb{R}^3$, the normalized plane normal vector $\mathbf{n}$ is:

$$\mathbf{n} = \frac{\mathbf{p}_{cp} - \mathbf{p}_{camera}}{||\mathbf{p}_{cp} - \mathbf{p}_{camera}||} = \frac{\mathbf{p}_{cp}}{||\mathbf{p}_{cp}||} \tag{3.2}$$

The parameters (a, b, c) can now be easily derived from $\mathbf{n}$ with the relation:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \mathbf{n} \tag{3.3}$$

In order to compute d, the coordinates of a random point on the plane and the normal vector $\mathbf{n}$ have to be known. The initial position of the plane is set so that the contact point $\mathbf{p}_{cp}$ is the point on the plane (i.e. $\mathbf{p}_{cp}$ fulfills the plane equation (Equation 3.1)). Then, d is:

$$d = \mathbf{n}^T \cdot \mathbf{p}_{cp} \tag{3.4}$$

Now, after knowing how to compute the plane equation parameters (a, b, c, d), the number of points behind the plane can be determined. Inserting the coordinates of a point (x, y, z) into the plane equation (Equation 3.1), the numerical value on the

right side of the equation (here called $result$) indicates where the point lies in relation to the plane:

- $result = 0$: point lies on the plane
- $result > 0$: point lies behind the plane
- $result < 0$: point lies in front of the plane

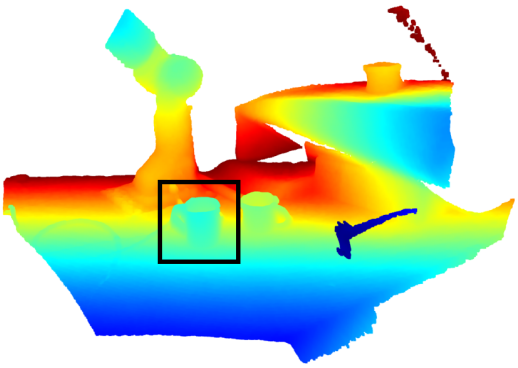
The main idea is to continuously shift the plane towards the rear of the object until only a predefined number of points t is located behind the plane. These t number of points then get removed from the object point cloud as they originate from camera noise. The algorithm is shown in Algorithm 2. First, the plane gets initialized with the normal vector n and the starting position that should go through the contact point p_{cp} (lines 1 - 2). Then, for every point p in the point cloud P, it is checked whether the point lies behind the plane (inserting the coordinates into the plane equation and checking the result, lines 7 - 8). If it lies behind the plane, the counter (counts the number of points behind the plane) gets incremented and the point itself gets appended to the list of points behind the plane (lines 9 - 11). If the counter exceeds t (predefined number of points that are allowed to be behind the plane), the current iteration stops (as there are already too many points behind the plane and consequently, the plane's current position is not correct, lines 13 - 14) and the plane is shifted towards the rear of the object (the point on the plane that influences the calculation of d is moved along the direction of the normal vector n, line 20). If the position of the plane is correct in a way that the number of points behind the plane is equal to or lower than t, the algorithm stops (lines 17 - 18) and returns the point cloud without the points behind the plane (lines 23 - 24).

Algorithm 2 Camera Noise Removal Algorithm

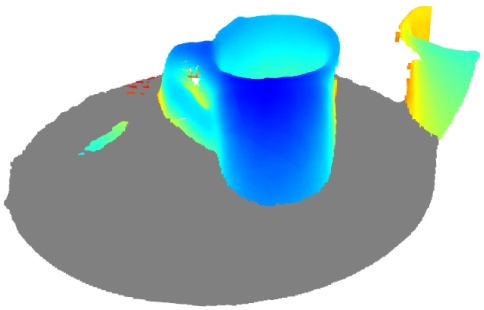
Input: Point cloud $\mathbf{P}$, contact point $\mathbf{p}_{cp}$, plane normal vector $\mathbf{n}$, number of points behind plane that should be discarded $\mathbf{t}$

Output: Point cloud $\mathbf{P}^*$ without points behind plane

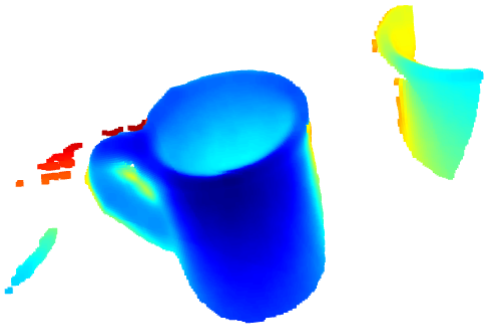
$a, b, c \leftarrow n_x, n_y, n_z$
$point\_on\_plane \leftarrow p_{cp}$
while $True$ **do**
    $list \leftarrow \emptyset$
    $counter \leftarrow 0$ ▷ counts points behind plane
    $d \leftarrow n^T \cdot point\_on\_plane$
    for every point **p** in **P** **do**
        $result \leftarrow (a, b, c, d) \cdot (p_x, p_y, p_z, 1)^T$ ▷ plane equation
        if $result > 0$ **then**
            $counter \leftarrow counter + 1$
            $list.append(p)$
        end if
        if $counter > t$ **then**
            $break$
        end if
    end for
    if $counter <= t$ **then**
        $break$
    else
        $point\_on\_plane \leftarrow point\_on\_plane - 0.001 \cdot n$
    end if
end while
$P^* \leftarrow P \setminus \{list\}$
return P^*



(a) Whole scene. Object to be grasped marked by black rectangle.

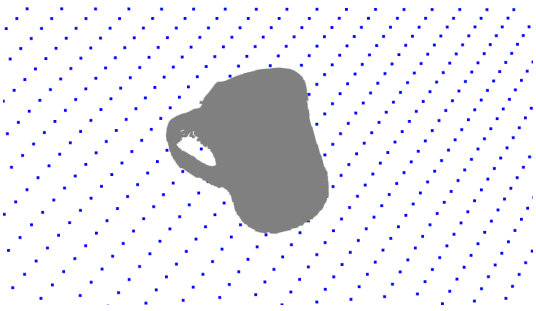


(b) Sphere isolated object. Segmented plane (tabletop) in grey color.



(c) Sphere isolated object with segmented plane removed.

(d) DBSCAN clustering result. The cluster representing the object in blue color.

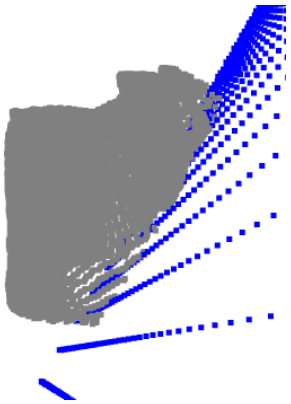


(e) Plane for noise removal. The plane (blue color) cuts through the rear of the object.

(f) Final isolated object.

Figure 3.4: Visualization of the individual steps executed during the object isolation process. In this example, a mug to be grasped is isolated from the rest of the scene point cloud.

(a) Side view of the object point cloud with camera noise at the rear sides (circled in red).



(b) Final plane position. Points behind the plane are already removed.

(c) Side view of the final isolated object point cloud with removed camera noise.

Figure 3.5: Camera noise removal (step 4 in the object isolation process).

3.4.2 Object Point Cloud Transformation

The next step after isolating the object point cloud is to transform it. The reason for this is that we want to incorporate the grasp information (i.e. information about gripper pose and grasp contact point) into the coordinates of the object point cloud. In order to achieve this, two operations are executed:

1) **Coordinate transformation into gripper frame coordinates**: Initially, the point coordinates are with respect to the camera coordinate frame where the camera position is the origin $(0, 0, 0)^T$. To include the gripper pose information of a (proposed) grasp, the point cloud gets transformed into the gripper coordinate frame where the gripper itself is the origin. Given the transformation matrix $\mathbf{T}_C^G \in \mathbb{R}^{4x4}$ that transforms from camera frame to gripper frame, every point $\mathbf{p_i}^C \in \mathbb{R}^3$ in the camera frame of the object point cloud $\mathbf{P}$ gets multiplicated with the transformation matrix to obtain the corresponding point $\mathbf{p_i}^G \in \mathbb{R}^3$ in the gripper frame:

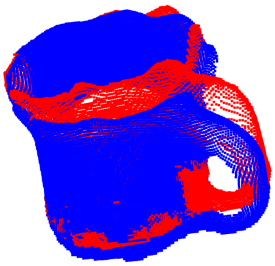
$$\mathbf{p}_i^G = \mathbf{T}_C^G \mathbf{p}_i^C \qquad \forall \quad \mathbf{p}_i \in \mathbf{P} \tag{3.5}$$

2) **Grasp contact point normalization**: In order to also include the grasp contact point information, every point $\mathbf{p}_i$ of the object point cloud $\mathbf{P}$ gets translated by the grasp contact point $\mathbf{p}_{cp}$. This results in the new set of points $\mathbf{p}_i^{'}$ and the new contact point $\mathbf{p}_{cp}^{'}$ at the coordinate frame origin:

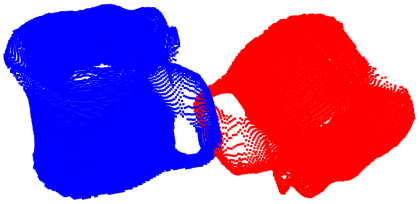
$$\mathbf{p}_i^{'} = \mathbf{p}_i - \mathbf{p}_{cp} \qquad \forall \quad \mathbf{p}_i \in \mathbf{P} \tag{3.6}$$

$$\mathbf{p}_{cp}^{'} = \mathbf{p}_{cp} - \mathbf{p}_{cp} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.7}$$

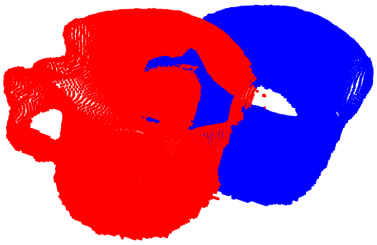
Figure 3.6 shows examples of object point cloud poses after transformation (i.e. transformation into gripper frame and contact point normalization). In each of the four subfigures, two mug point clouds are visualized that were transformed according to two different grasps. The grasps in (a) are very similar (similar gripper pose and grasp contact point) as the two object point clouds are well aligned. The grasps in (b) have a similar grasp contact point on the handle (point clouds are aligned at the handle) but different gripper poses. In (c), the grasps have similar gripper poses (object orientation is similar) but different grasp contact points (rim for the red one and handle for the blue one). Finally, in (d), the grasps are very different in terms of both grasp contact point and gripper pose.



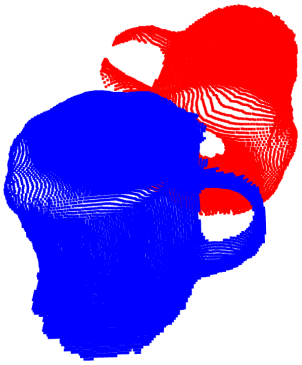
(a) Similar grasp contact points, similar gripper poses.



(b) Similar grasp contact points, different gripper poses.



(c) Different grasp contact points, similar gripper poses.



(d) Different grasp contact points, different gripper poses.

Figure 3.6: Visualization of object point cloud poses after transformation. The poses represent the corresponding grasps that were executed on the mugs (i.e. gripper pose and grasp contact point). The more similar the grasps are, the more aligned the transformed object point clouds are. Here, 4 different cases of (non-) similarity between two grasps are shown.

3.5 Candidate and Demonstration Grasp Similarity Estimation

The similarity between a candidate grasp and the set of human demonstration grasps for a given task should now be estimated. The demonstrations serve as task-oriented ground-truth grasps. Therefore, a high similarity indicates high task-suitability of the candidate grasp. As previously described in the Point Cloud Pre-Processing Chapter 3.4, the grasp information (i.e. gripper pose and grasp contact point) is now incorporated into the coordinates of the transformed object point clouds. In order to estimate the grasp similarity, point cloud registration is executed. Given a specific task, the candidate grasp's transformed object point cloud is pairwise registered to every transformed object point cloud in the demonstration set. Based on the registration results, a task-oriented grasping suitability metric score (or just task-suitability metric score) is computed. The score is the final output of the pipeline and indicates how suitable a candidate grasp is for a specific task.

This Chapter describes the point cloud registration process and the theory behind the computation of the task-suitability metric.

3.5.1 Object Point Cloud Registration

The goal of the point cloud registration is to align the transformed object point cloud of the candidate grasp (source point cloud) to the transformed object point cloud of a demonstration grasp (target point cloud). The registration process involves three registration algorithms (the algorithms are described in detail in the background Chapter 2.1.4) and is built up upon the three following steps:

(1) First, a RANSAC-based global rigid registration algorithm is applied to obtain a coarse alignment of the source and target point cloud.

(2) Secondly, the ICP local rigid registration algorithm is used to refine the alignment.

(3) Finally, the CPD rigid registration algorithm is applied. Although tight alignment should already be achieved after the first two steps, the RANSAC-based and the ICP registration algorithms have only applied rigid rotations and translations. Executing the CPD algorithm as the last step brings in an advantageous feature: while also computing the rotation and translation, it additionally considers possible scale differences between the two point clouds and introduces a scaling transformation accordingly. This is particularly useful if the candidate and demonstration objects differ in their size. This way, the pipeline can also process new object instances with varying sizes inside an object class and is not restricted to exactly the same object.

3.5.2 Point Cloud Registration-based Metrics

After executing the point cloud registration of the candidate and the demonstration grasp object point clouds, the registration result and the applied transformation are evaluated. For this purpose, three error/distance metrics are introduced that are explained in the following subchapters.

3.5.2.1 Registration Distance Error Metric

We compute a registration distance error metric to determine how good the final registration result is. The higher the alignment and overlap between the two point clouds after registration, the lower the registration distance error $d_{reg} \in \mathbb{R}$. The error metric is computed as follows:

(1) Given two point clouds $\mathbf{P}_1$ and $\mathbf{P}_2$ with n points that were registered (i.e. after registration). For each point $\mathbf{p}_i$ in the point cloud $\mathbf{P}_1$, the average euclidean distance to its k-nearest neighbors from the other point cloud $\{\mathbf{p}_1, \ldots, \mathbf{p}_k\} \in \mathbf{P}_2$ is computed:

$$d(\mathbf{p}_i) = \frac{1}{k} \sum_{j=1}^{k} ||\mathbf{p}_i - \mathbf{p}_j|| \qquad \forall \quad \mathbf{p}_i \in \mathbf{P}_1 \tag{3.8}$$

(2) After computing the average euclidean distance to the k-nearest neighbors from the other point cloud for every point $\mathbf{p}_i$ in the point cloud $\mathbf{P_1}$ (average with respect to a single point $\mathbf{p}_i$), this distance value is then averaged over all n points in $\mathbf{P_1}$ (average with respect to the whole point cloud $\mathbf{P_1}$):

$$d(\mathbf{P}_1) = \frac{1}{n} \sum_{i=1}^{n} d(\mathbf{p}_i) \tag{3.9}$$

(3) To obtain the final registration distance error value, the distance value from step 2 is computed for both point clouds (for the source point cloud $\mathbf{P}_s = \mathbf{P}_1$ as well as for the target point cloud $\mathbf{P}_t = \mathbf{P}_2$) and summed up:

$$d_{reg} = d(\mathbf{P}_s) + d(\mathbf{P}_t) \tag{3.10}$$

3.5.2.2 Absolute Rotation Distance Metric

The absolute rotation distance is a metric for the amount of rotation that was applied to the source point cloud (candidate) to register it to the target point cloud (demonstration). Since the initial object point cloud poses before the registration represent the respective gripper poses, the absolute rotation distance metric can be seen as a similarity metric between the candidate and demonstration grasp gripper poses. The lower the absolute rotation distance is, the more similar the grasp gripper poses are. Since three point cloud

registration algorithms (RANSAC-based, ICP and CPD) are applied to the point clouds (and every algorithm will apply a rotation), the final rotation matrix of the whole registration process $\mathbf{R}_{total}$ can be computed as the multiplication of the rotation matrices of each registration algorithm:

$$\mathbf{R}_{total} = \mathbf{R}_{CPD}\mathbf{R}_{ICP}\mathbf{R}_{RANSAC} \tag{3.11}$$

The absolute rotation distance $d_{rot} \in \mathbb{R}$ is then obtained by computing the norm of the difference between the identity matrix and the final rotation matrix:

$$d_{rot} = ||\mathbb{1} - \mathbf{R}_{total}|| \tag{3.12}$$

where $\mathbb{1} \in \mathbb{R}^{3x3}$ is the identity matrix.

3.5.2.3 Grasp Contact Point Distance Metric

The grasp contact point distance is a metric for the euclidean distance between the two grasp contact points on the object point clouds after registration. Assuming a good registration result (i.e. good alignment), closer contact points indicate more similar grasp regions. This a very important metric for task-oriented grasping since for most task-oriented grasps, a specific object part with a suitable affordance should be selected. If the candidate and demonstration object point clouds are now well-registered and the candidate contact point has a small distance to the demonstration contact point, there is a high probability that they are on the same object part. This indicates that the candidate grasp's contact point is in a region suitable for the given task. The grasp contact point distance $d_{cp} \in \mathbb{R}$ is computed as follows:

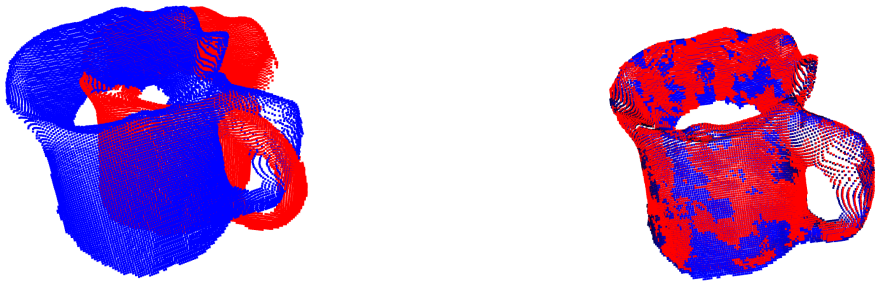
$$d_{cp} = ||\mathbf{p}_{cp}(d) - \mathbf{T}\mathbf{p}_{cp}(c)|| \tag{3.13}$$

where $\mathbf{p}_{cp}(d)$ and $\mathbf{p}_{cp}(c)$ are the grasp contact points on the demonstration and candidate point cloud, respectively and $\mathbf{T}$ is the transformation that is applied to the candidate point cloud for registration.

Figure 3.7 shows an example of two point clouds of the same mug before and after registration. Before the registration, the point clouds' initial poses (Figure 3.7a) represent their respective grasps (i.e. gripper pose and grasp contact point). Both point clouds are initially aligned at their grasp contact points on the handle (both contact points are at the coordinate frame origin because of the grasp contact point normalization) but have slightly different orientations (i.e. the gripper poses for their grasps were slightly different). The point cloud registration algorithms compute a spatial transformation to align them (Figure 3.7b). Based on the registration result and the computed transformation, the three metrics described previously in this Chapter can then be computed.

3.5.2.4 Metric Normalization

Computing the metrics results in corresponding metric scores (i.e. numerical values). However, the value ranges can differ significantly. To overcome this, normalization is



(a) Before registration (b) After registration

Figure 3.7: Candidate (red) and demonstration (blue) object point cloud registration.

applied (this will be useful when combining the metrics in the next step). In this thesis, two types of normalization are used. Given a dataset $D = \{d_1, \ldots, d_n\}$ with n data values, the normalizations work as follows:

- **min-max normalization**: performs a linear transformation on the original data. This technique gets all the normalized data into the range [0,1]:

$$d^* = \frac{d - d_{min}}{d_{max} - d_{min}} \tag{3.14}$$

  where d^* is the normalized value of d, and d_{max} and d_{min} are the maximum and minimum value in the dataset D, respectively.

- **z-score normalization**: refers to the process of normalizing every value in a dataset such that the mean μ of the dataset D is 0 and the standard deviation σ is 1:

$$d^* = \frac{d - \mu}{\sigma} \tag{3.15}$$

  where d^* is the normalized value of d, μ is the mean and σ is the standard deviation of the unnormalized values in the dataset D.

In the next step, we want to normalize the metrics *absolute rotation distance* d_{rot} and *grasp contact point distance* d_{cp}. The parameters needed to compute the normalized metrics d^*_{rot} and d^*_{cp} (max and min for min-max normalization; μ and σ for z-score normalization) are solely derived from the set of demonstrations for a given task and object. This means that the normalization parameters $\langle max, min, \mu, \sigma \rangle$ of a metric are different for every demonstration set and independent from the candidate grasp.

An example is presented to make it more clear: assume that we demonstrate task-oriented grasps on the object for a task (*task 1*) and obtain the set of demonstrations $D = \{d_1, d_2, \ldots, d_n\}$. Then, on the same object, we demonstrate task-oriented grasps for a different task (*task 2*) and get the demonstration set $D' = \{d'_1, d'_2, \ldots, d'_n\}$. We want the normalization to be set-specific, so D_1 has parameters $\langle max, min, \mu, \sigma \rangle$ for d^*_{rot} and d^*_{cp} while

D_2 has its own different parameters $\langle max^{'}, min^{'}, \mu^{'}, \sigma^{'} \rangle$. The data values (multiple values of d_{rot} and d_{cp}) are obtained through pairwise registration and metric computation inside a demonstration set. Thus, for n demonstrations, $\frac{n(n-1)}{2}$ pairs are possible. Exemplary for the normalization parameter computation for the demonstration set $D = \{d_1, d_2, \ldots, d_n\}$:

(1) Compute the metric *absolute rotation distance* d_{rot} for all demonstration pairs (d_i, d_j) in D with $(i \neq j)$: The first demonstration pair yields $d_{rot}(d_1, d_2)$. The second demonstration pair yields $d_{rot}(d_1, d_3)$ and so on until reaching the last pair that yields $d_{rot}(d_{n-1}, d_n)$

(2) Now, we have $\frac{n(n-1)}{2}$ data values for d_{rot} and can easily obtain its respective parameters $\langle max, min, \mu, \sigma \rangle$. These parameters are set-specific (as they are computed for a specific set of demonstrations).

(3) The same procedure is also performed with *absolute grasp contact point distance* d_{cp} metric to obtain its respective parameters.

If a candidate grasp c is registered to a demonstration d in the demonstration set D, the resulting metric values $d_{rot}(c, d)$ and $d_{cp}(c, d)$ are normalized to obtain $d^*_{rot}(c, d)$ and $d^*_{cp}(c, d)$ by using the set-specific parameters $\langle max, min, \mu, \sigma \rangle$ computed from the demonstration set D for each metric. If a different demonstration set $D^{'}$ is queried, the parameters $\langle max^{'}, min^{'}, \mu^{'}, \sigma^{'} \rangle$ computed from the set $D^{'}$ are used for normalization.

3.5.3 Task-oriented Grasping Suitability Metric

The absolute rotation distance and grasp contact point distance are good metrics to roughly estimate the similarity between two grasps (as they indicate gripper pose and contact region similarity, respectively). However, it would be beneficial if we could combine the information of the two metrics. Therefore, we introduce a grasp similarity distance metric which combines the information of two aforementioned metrics to create a more descriptive one. It is the sum of the normalized absolute rotation distance and the normalized grasp contact point distance:

$$d^*_{sim} = d^*_{rot} + d^*_{cp} \tag{3.16}$$

Now, if a specific task is given and a candidate grasp is proposed, the goal is to determine whether the grasp is suitable for the task or not. As we have obtained human demonstrations that serve as ground-truth task-oriented grasps, it is desirable to know how similar the candidate is to the demonstrations. If it is very similar, it can be inferred with high certainty that the candidate is also a task-suitable grasp. Therefore, we compute the grasp similarity distance between the candidate and all demonstration grasps that were recorded for this task. If the set of grasp demonstrations $D = \{d_1, \ldots, d_n\}$ for a task consists of n demonstrations in total, the similarity distance is pairwise computed between candidate and all n demonstrations and then averaged. This results in the task-oriented grasping suitability metric m_{suit} that is the final output of the pipeline:

$$m_{suit}(c, D) = \frac{1}{n} \sum_{i=1}^{n} d^*_{sim}(c, d_i) = \frac{1}{n} \sum_{i=1}^{n} d^*_{rot}(c, d_i) + d^*_{cp}(c, d_i) \tag{3.17}$$

where $m_{suit}(c, D)$ is the task-oriented grasping suitability metric for candidate grasp c computed from comparison to the set of demonstrations D, and $d^*_{sim}(c, d_i)$ is the normalized grasp similarity distance between the candidate grasp c and a single demonstration $d_i \in D$.

The lower the metric score of d_{sim}, the more similar the grasps are. In the context of comparing a candidate grasp to the ground-truth task-oriented demonstration grasps, a high similarity indicates a high task-suitability of the candidate. As m_{suit} is the averaged sum of d_{sim}, a low metric score of m_{suit} is desired for a high task-suitability. The suitability metric is the final output of the pipeline and can be further used for several applications. For instance, it can be used to rank multiple candidate grasps in terms of their task-suitability or to classify them into certain task categories.

Chapter 4

Experimental Evaluation

This chapter describes the experiments that are conducted to evaluate the descriptiveness of the task-oriented grasping suitability metric. Two possible applications of the metric are tested: firstly, a task-suitability ranking of multiple proposed candidate grasps is generated based on the computed metric scores. Secondly, the metric scores are used to classify candidate grasps into a certain task category.

Both the demonstration and candidate grasps are collected by a human via the teleoperation user interface. For each object and task, 30 task-oriented grasps are recorded. From these 30 grasps, 18 grasps are used as demonstration set and the remaining 12 are used as candidate set for testing. In order to create more diversity within the sets, the demonstration/test split is executed 3 times in a random manner, resulting in 3 different demonstration sets and 3 different test sets for each object and task. During the experiments, the metric computations are performed three times on the different sets and the results are then averaged to obtain a final result.

For each object, two tasks that can be executed with it are defined. Each task has a specific object part assigned to it that is selected for a task-oriented grasp. The relations between the object, task, and object part to be grasped are shown in Table 4.1. Taking the drill machine as an example, the handle is always grasped for the drilling task and the machine top is always grasped for the handover task.

Grasps were executed on the objects shown in Figure 4.1. For each object class (except for the drill machine class), multiple instances were used. Although we recorded grasps for all objects, only the grasps on the 3 mugs, 3 shampoo bottles and 1 drill machine were used for the experimental evaluation. For the remaining objects, grasp collection proved to be challenging due to inaccuracies of the user interface and noisy point clouds caused by imperfect perception of the RGB-D camera. Consequently, these grasps were not used in the evaluation as they were not seen as reliable enough.

Object	Task	Object part to be grasped
Mug	Pour	Handle
Mug	Handover	Rim
Shampoo bottle	Squeeze	Body (side grasp)
Shampoo bottle	Handover	Top (top grasp)
Drill machine	Drill	Handle
Drill machine	Handover	Top (top grasp)

Table 4.1: Relations between the object, task and corresponding object part to be grasped for the given task.

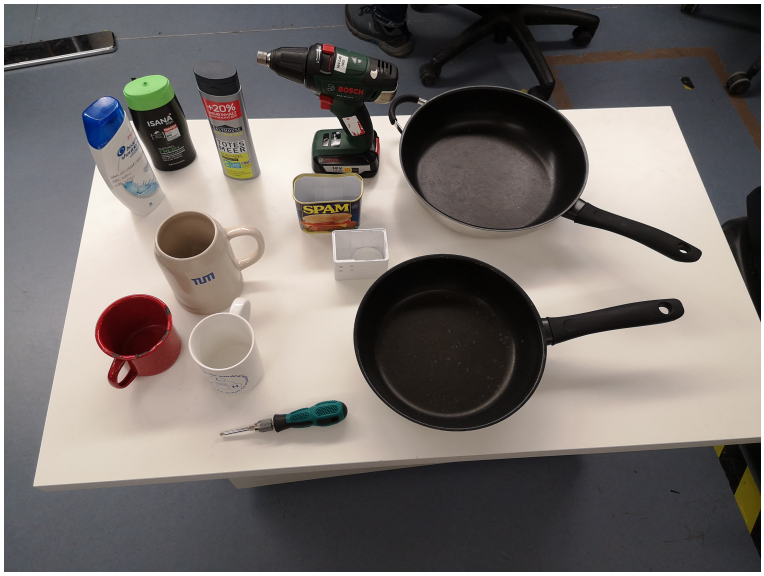


Figure 4.1: Objects used for grasp collection. The mugs (bottom left), shampoo bottles (top left) and drill machine (top middle) were used for the final experimental evaluation.

4.1 Task-Suitability Ranking of Candidate Grasps

One of the possible applications of the task-oriented grasping suitability metric m_{suit} is the ranking of multiple candidate grasps in terms of their suitability for a given task. In order to do this, each candidate grasp is input into the pipeline and compared to the set of human demonstrations recorded for this task. The pipeline then outputs the task-suitability metric score for each candidate grasp. Based on the scores, the task-suitability ranking of the candidate grasps is created. The lower the metric score, the more suitable the grasp is for a given task. Thus, the lower the metric score, the higher the ranking position is.

If the ranking is ideal (i.e. correct), then the candidate grasps that were labeled as task-suitable should be at the top of the task-suitability ranking and all unsuitable candidate grasps should be at the bottom. In order to evaluate and express the accuracy of the

ranking, the percentage value $p_{correct}$ is computed. Given n task-suitable candidate grasps, $p_{correct}$ is the percentage of task-suitable grasps that are placed in the top n positions of the ranking:

$$p_{correct} = \frac{\#\text{ task-suitable grasps in the top } n \text{ ranking places}}{n} \cdot 100 \tag{4.1}$$

The ranking for each task & object consists of 24 grasps in total: 12 task-suitable candidate grasps and 12 non-task-suitable candidate grasps (i.e. $n = 12$). Since we have defined two tasks per object, the non-task-suitable grasps are those from the task that is not currently being evaluated. This means, for instance, when creating the task-suitability ranking for the mug and the pouring task, the task-suitable candidate grasps are taken from the mug/pour candidate set and the non-task-suitable candidate grasps are taken from the mug/handover candidate set.

Another very useful property that can be derived from the task-suitability ranking is that the top-ranked grasp is the most task-suitable among all candidates. When a robot has to choose between multiple candidates, it would select the top-ranked one. For this reason, we want to check whether the top-ranked grasp is actually a task-suitable one. If it is task-suitable, the 1st rank box is ticked with a ✓, otherwise with a ✗.

A high $p_{correct}$ value and a task-suitable top-ranked candidate grasp would verify the descriptiveness of our introduced task-oriented grasping suitability metric m_{suit}. As the metric itself is a combination of two other metrics (*absolute rotation distance* d_{rot} and *grasp contact point distance* d_{cp} (the asterisk * to denote the normalization is omitted here)), we would also like to test their individual descriptiveness and see if their combination to form m_{suit} is actually bringing an improvement. Therefore, the task-suitability rankings are also created based on the computed scores of d_{rot} and d_{cp}.

4.1.1 Metric Computation Based on all Demonstrations

First, the task-oriented grasping suitability metric score m_{suit} is computed as described in Equation 3.17. Therefore, if the set of grasp demonstrations $D = \{d_1, \ldots, d_n\}$ for a task consists of n demonstrations in total, the registration and metric are pairwise computed between the candidate and all n demonstrations (same for d_{cp} and d_{rot}). Since all demonstrations are considered, this computation is here referred to as "metric computation based on all demonstrations". After computing the metric scores for all candidate grasps, the rankings are created. The ranking results are evaluated by the $p_{correct}$ percentage and the 1st rank tick/cross and illustrated in Figure 4.2. Min-max normalization was used for the metric computation in Table 4.2a, while z-score normalization was used for the metric computation in Table 4.2b. The evaluation shows that the choice of normalization does not have a major impact on the results. The metrics d_{rot} and d_{cp} show reasonable individual performance and their combination into m_{suit} proves to be very beneficial as the ranking accuracy is further improved. For all three metrics, the top-ranked grasp is always a task-suitable one.

	$\mathbf{p_{correct}}(\%)$			$\mathbf{1^{st}}$ **rank**		
Object/Task	d_{rot}	d_{cp}	m_{suit}	d_{rot}	d_{cp}	m_{suit}
Mug/pour	72.2	63.9	83.3	✓	✓	✓
Mug/handover	83.3	83.3	86.1	✓	✓	✓
Shampoo bottle/squeeze	77.8	100	100	✓	✓	✓
Shampoo bottle/handover	94.5	97.2	100	✓	✓	✓
Drill machine/drill	63.9	97.2	97.2	✓	✓	✓
Drill machine/handover	91.7	94.5	100	✓	✓	✓

(a) Min-max normalization

	$\mathbf{p_{correct}}(\%)$			$\mathbf{1^{st}}$ **rank**		
Object/Task	d_{rot}	d_{cp}	m_{suit}	d_{rot}	d_{cp}	m_{suit}
Mug/pour	72.2	63.9	86.1	✓	✓	✓
Mug/handover	83.3	83.3	86.1	✓	✓	✓
Shampoo bottle/squeeze	77.8	100	100	✓	✓	✓
Shampoo bottle/handover	94.5	97.2	100	✓	✓	✓
Drill machine/drill	63.9	97.2	97.2	✓	✓	✓
Drill machine/handover	91.7	94.5	100	✓	✓	✓

(b) Z-score normalization

Table 4.2: Evaluation of the task-suitability ranking results. The metric score computation was based on all demonstrations.

4.1.2 Metric Computation Based on a Subset of Demonstrations

The computation of the metric scores strongly relies on a good registration result when registering the transformed object point clouds. If the registration does not provide a good result (i.e. poor alignment between the point clouds), the metrics *absolute rotation distance* and *contact point distance* could deliver erroneous values and consequently, the task-oriented suitability metric score would also not be accurate either. As described in Chapter 3.5.2.1, we introduced the registration distance error metric d_{reg} which determines how accurate the registration result is. A low score for the registration error distance metric indicates a good registration result.

A poor registration result is not necessarily caused by failure of the registration algorithm (e.g. the algorithm got stuck in a local minimum and did not find the optimal global solution). The scenes that we capture are obtained from a single camera and a single viewpoint. Therefore, the object point clouds are only partial point clouds. When testing the registration algorithms, it could be observed that the partiality of the object point

clouds caused the algorithms to align fully observable object parts to fully observable object parts and non-fully-observable to non-fully observable ones. For example, if a mug was captured in two different poses (a pose where the handle was on the right side and a pose where the handle was on the left side from the camera's point of view), the registration result was a well-aligned object body but the handles were still on different sides. This is certainly not ideal but the registration distance error metric d_{reg} can detect such cases by outputting a higher distance error score.

We now want to test whether only considering the demonstrations that yield a better registration result leads to more accurate task-suitability rankings. For this purpose, the candidate object point cloud is pairwise registered to all demonstration point clouds and the registration distance error metric score d_{reg} is computed for all registration results. Then, instead of considering all demonstrations from the demonstration set $D = \{d_1, \ldots, d_n\}$, only the subset S consisting of the k demonstrations with the lowest d_{reg} score is further used for the metric computations of m_{suit}, d_{rot} and d_{cp}. Thus, this type of computation is here referred to as "metric computation based on a subset of demonstrations". Table 4.3a (min-max normalization) and Table 4.3b (z-score normalization) show the ranking evaluation if only the best 50% of demonstrations (in terms of d_{reg} score) are considered. As we have 18 demonstrations per object and task, 50% means $k = 9$. As previously, the evaluation again shows that the choice of normalization does not affect the ranking results and for all three metrics, the top-ranked grasp is always a task-suitable one. Compared to the "metric computation based on all demonstrations" from before, improvements for d_{cp} and d_{rot} are observed. The performance for d_{rot} remains about the same. Interpreting the results, we can assume that the "metric computation based on a subset of demonstrations" yields slightly more accurate ranking results.

	$\mathbf{p_{correct}}(\%)$			$\mathbf{1^{st}}$ **rank**		
Object/Task	d_{rot}	d_{cp}	m_{suit}	d_{rot}	d_{cp}	m_{suit}
Mug/pour	75	88.9	83.4	✓	✓	✓
Mug/handover	75	86.1	91.7	✓	✓	✓
Shampoo bottle/squeeze	69.4	100	100	✓	✓	✓
Shampoo bottle/handover	80.5	100	100	✓	✓	✓
Drill machine/drill	66.7	100	100	✓	✓	✓
Drill machine/handover	94.5	100	100	✓	✓	✓

(a) Min-max normalization

	$\mathbf{p_{correct}}(\%)$			$\mathbf{1^{st}}$ **rank**		
Object/Task	d_{rot}	d_{cp}	m_{suit}	d_{rot}	d_{cp}	m_{suit}
Mug/pour	75	88.9	83.4	✓	✓	✓
Mug/handover	75	86.1	91.7	✓	✓	✓
Shampoo bottle/squeeze	69.4	100	100	✓	✓	✓
Shampoo bottle/handover	80.5	100	100	✓	✓	✓
Drill machine/drill	66.7	100	100	✓	✓	✓
Drill machine/handover	94.5	100	100	✓	✓	✓

(b) Z-score normalization

Table 4.3: Evaluation of the task-suitability ranking results. The metric score computation was based on the best 50% of demonstrations (in terms of d_{reg} score).

Finally, the suitability rankings are computed for the case that only the best 3 demonstrations (in terms of d_{reg} score) are considered (i.e. $k = 3$). The results are illustrated in Table 4.4a (min-max normalization) and Table 4.4b (z-score normalization). It can be seen that the results are very similar to those in Table 4.3 when the metric score computation was based on the best 50% of demonstrations (in terms of d_{reg} score). Therefore, further downsizing of the considered demonstration subset does not show significant improvement.

	$\mathbf{p_{correct}}(\%)$			$\mathbf{1^{st}}$ **rank**		
Object/Task	d_{rot}	d_{cp}	m_{suit}	d_{rot}	d_{cp}	m_{suit}
Mug/pour	83.4	88.9	83.4	✓	✓	✓
Mug/handover	77.8	91.7	91.7	✓	✓	✓
Shampoo bottle/squeeze	75	100	97.2	✓	✓	✓
Shampoo bottle/handover	77.8	100	100	✓	✓	✓
Drill machine/drill	66.7	100	100	✓	✓	✓
Drill machine/handover	91.7	100	100	✓	✓	✓

(a) Min-max normalization

	$\mathbf{p_{correct}}(\%)$			$\mathbf{1^{st}}$ **rank**		
Object/Task	d_{rot}	d_{cp}	m_{suit}	d_{rot}	d_{cp}	m_{suit}
Mug/pour	83.4	88.9	83.4	✓	✓	✓
Mug/handover	77.8	91.7	91.7	✓	✓	✓
Shampoo bottle/squeeze	75	100	97.2	✓	✓	✓
Shampoo bottle/handover	77.8	100	100	✓	✓	✓
Drill machine/drill	66.7	100	100	✓	✓	✓
Drill machine/handover	91.7	100	100	✓	✓	✓

(b) Z-score normalization

Table 4.4: Evaluation of the task-suitability ranking results. The metric score computation was based on the best 3 demonstrations (in terms of d_{reg} score).

4.2 Task Classification of Candidate Grasps

Another possible application of the task-oriented grasping suitability metric is the classification of a candidate grasp into a task category (assuming that the object is known). In order to do this, the candidate's task-oriented grasping suitability metric score m_{suit} is computed for both tasks that were defined for an object. The candidate is then classified as task-suitable for the task, for which the metric score was lower. For instance, if a candidate grasp on a mug should be classified, the candidate's metric score m_{suit} is computed for both the pouring task and the handover task. If m_{suit} is lower for the pouring task, the candidate is classified as task-suitable for the pouring task. The same process is also executed for d_{rot} and d_{cp} to investigate their individual performance in classification.

For each object & task, there are 12 task-suitable candidate grasps. For each of the candidates, classification is executed. To validate the classification success, the percentage value $c_{correct}$ is introduced. It is the percentage of candidates that were correctly classified for a given object & task:

$$c_{correct} = \frac{\#\text{ correctly classified candidate grasps}}{\#\text{ total number of candidate grasps}} \cdot 100 \tag{4.2}$$

Same as for ranking computation, the classification is performed for three different cases: metric computation based on all demonstrations (Table 4.5a), metric computation based on the best 50% of demonstrations (Table 4.5b) and metric computation based on the best 3 demonstrations (Table 4.5c). Only min-max normalization was used as the previous evaluations showed that the normalization choice does not have an impact on the results.

The evaluation results show that all three metrics perform really well for task classification. The combination of d_{rot} and d_{cp} to form m_{suit} proves to be again beneficial as the accuracy improves. The metric score computation based on the best 50% of demonstrations delivers slightly better results than the metric score computation for all demonstrations/best 3 demonstrations.

	$\mathbf{c_{correct}}(\%)$		
Object/Task	d_{rot}	d_{cp}	m_{suit}
Mug/pour	91,7	83.4	94.5
Mug/handover	66.7	97.2	86.1
Shampoo bottle/squeeze	97.2	97.2	100
Shampoo bottle/handover	83.3	100	100
Drill machine/drill	88.9	100	100
Drill machine/handover	91.7	100	100

(a) Metric score computation based on all demonstrations

	$\mathbf{c_{correct}}(\%)$		
Object/Task	d_{rot}	d_{cp}	m_{suit}
Mug/pour	94.5	100	100
Mug/handover	80.6	94.4	94.4
Shampoo bottle/squeeze	97.2	100	100
Shampoo bottle/handover	91.7	100	100
Drill machine/drill	86.1	100	100
Drill machine/handover	91.7	100	100

(b) Metric score computation based on the best 50% of demonstrations (in terms of d_{reg} score)

	$\mathbf{c_{correct}}(\%)$		
Object/Task	d_{rot}	d_{cp}	m_{suit}
Mug/pour	100	100	100
Mug/handover	83.3	100	86.1
Shampoo bottle/squeeze	75	100	100
Shampoo bottle/handover	86.1	100	100
Drill machine/drill	80.6	100	100
Drill machine/handover	97.2	100	100

(c) Metric score computation based on the best 3 demonstrations (in terms of d_{reg} score)

Table 4.5: Evaluation of the task classification results.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, a task-oriented grasping metric was introduced which indicates how suitable a proposed candidate grasp is for a given task. The metric is the final output of a point cloud processing pipeline that estimates the similarity between the candidate and a set of human demonstrations that serve as task-oriented ground-truth grasps. In order to collect the human demonstrations, a teleoperation user interface is used that allows operation by non-experts. For each grasp, the pipeline isolates the captured object point cloud and transforms it into the gripper frame. When comparing a candidate to a demonstration grasp, point cloud registration is performed on the respective transformed object point clouds. Based on the registration results, two metrics are computed that indicate the similarity between the grasp's gripper poses and grasp contact points. The combination of these two metrics yields the task-oriented grasping suitability metric. A major advantage of our approach is that the pipeline and metric computations are object class-agnostic and therefore generalizable to all kinds of object classes and possible tasks. Moreover, our approach is learning-free and does not require training on large-scale datasets. In addition, the objects do not need to be captured from multiple viewpoints as the pipeline is able to process single-view partial point clouds. The experimental evaluation has shown that the task-oriented grasping suitability metric has strong descriptive power and provides highly accurate results when used for the applications of ranking and classifying grasps in terms of their task-suitability.

5.2 Future Work

The experimental evaluation has shown that the introduced task-oriented grasping suitability metric has a high descriptiveness and performs well in its applications. Nevertheless, there are things that could be further improved in future studies.

The candidate grasps for our experimental evaluation were collected by a human via the teleoperation user interface. Future studies could use a grasp estimator to sample candidate grasps based on the perceived geometry of the object. However, a human would still be required in the process to label the candidates as task-suitable or non-task-suitable if the same experimental evaluation is to be conducted.

Furthermore, the object poses could be varied more during the grasp collection (e.g. the object pose could be changed after every recorded grasp). Although there was some object pose variation during grasp collection in our experiments, time constraints and issues with the interface prevented a greater variety of poses. In addition, the number of different object classes with multiple instances within every class could be increased.

In our experiments, only partial point clouds obtained from one camera and a single viewpoint were processed. It could be investigated whether the evaluation performance improves if fully reconstructed 3D models of the objects to be grasped are used. The full object models can be created by capturing the objects from multiple viewpoints and fusing the obtained object point clouds into a global object model.

Finally, when computing the absolute rotation distance metric d_{rot}, the gripper symmetry could be taken into account. Around the gripper approach vector towards the contact point, a gripper rotation of 180° does not make a difference for the grasp itself (because the gripper is symmetric around its wrist). However, 180° of rotation does affect the value of d_{rot}. Therefore, when comparing two grasps, d_{rot} could be computed twice (once for the actually selected grasp and once for the same grasp rotated by 180° around the approach vector) and only the grasp with the smaller value of d_{rot} is kept.

List of Figures

List of Tables

Bibliography

[AC11] Jacopo Aleotti and Stefano Caselli. Part-based robot grasp planning from human demonstration. In *2011 IEEE International Conference on Robotics and Automation*, pages 4554–4560, 2011.

[ACVB09] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[AMB17] David Avidar, David Malah, and Meir Barzohar. Local-to-global point cloud registration using a dictionary of viewpoint descriptors. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 891–899, 2017.

[BCDS08] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. *Robot Programming by Demonstration*, pages 1371–1394. 01 2008.

[BHS+17] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[BM92] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[CKH+16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[CM91] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2724–2729 vol.3, 1991.

[DA12] Hao Dang and Peter K. Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *2012 IEEE/RSJ In-*

ternational Conference on Intelligent Robots and Systems, pages 1311–1317, 2012.

[Der05] Konstantinos G. Derpanis. Overview of the ransac algorithm. 2005.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[DNR18] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5882–5889, 2018.

[DPM17] Renaud Detry, Jeremie Papon, and Larry Matthies. Task-oriented grasping with semantic and geometric scene understanding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3266–3273, 2017.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226â231. AAAI Press, 1996.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381â395, jun 1981.

[FZG+20] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020.

[Gib79] James J. Gibson. *The Ecological Approach to Visual Perception: Classic Edition*. Houghton Mifflin, 1979.

[HGU+21] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 4267–4276, 2021.

[HIT+15] Dirk Holz, Alexandru E. Ichim, Federico Tombari, Radu B. Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine*, 22(4):110–124, 2015.

[KSC20] David Kent, Carl Saldanha, and Sonia Chernova. Leveraging depth data in remote robot teleoperation interfaces for general object manipulation. *The International Journal of Robotics Research*, 39(1):39–53, 2020.

[KSHK17] Mia Kokic, Johannes A. Stork, Joshua A. Haustein, and Danica Kragic. Affordance detection for task-specific grasping using deep learning. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 91–98, 2017.

[LDC20] Weiyu Liu, Angel Daruna, and Sonia Chernova. CAGE: Context-aware grasping engine. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2550–2556. IEEE, 2020.

[Lej21] Khaled Lejri. Unity-based user interface for robotic grasp teleassistance. Bachelor's thesis, Chair of Media Technology, Technical University of Munich (TUM), 2021.

[Mas90] Dominic W. Massaro. *The American Journal of Psychology*, 103(1):141–143, 1990.

[Mil95] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39â41, nov 1995.

[MLM+21] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In *Conference on Robot Learning*, pages 1540–1557. PMLR, 2021.

[MS10] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

[NN+04] Alva Noë, Alva Noë, et al. *Action in perception*. MIT press, 2004.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[RBMB08] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391, 2008.

[RL01] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[RPCB20] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.

[SAI20] F. Sherwani, Muhammad Mujtaba Asad, and B.S.K.K. Ibrahim. Collaborative robots and industrial revolution 4.0 (ir 4.0). In *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pages 1–5, 2020.

[SDT+22] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400, 2022.

[Sta20] Cyrill Stachniss. Lecture notes in mobile sensing and robotics, 2020.

[Sto05] Alexander Stoytchev. Behavior-grounded representation of tool affordances. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3060–3065, 2005.

[TDP20] Spyridon Thermos, Petros Daras, and Gerasimos Potamianos. A deep learning approach to object affordance segmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2358–2362. IEEE, 2020.

[XKZD09] Zhixing Xue, Alexander Kasper, J. Marius Zoellner, and Ruediger Dillmann. An automatic grasp planning system for service robots. In *2009 International Conference on Advanced Robotics*, pages 1–6, 2009.

[Yil20] Soner Yildirim. Towards data science. `https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556`, 2020. Accessed on 20-02-2023.

[ZPK18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

[ZSG+18] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the Art on 3D Reconstruction with RGB-D Cameras. *Computer Graphics Forum (Eurographics State of the Art Reports 2018)*, 37(2), 2018.