A Semi-Autonomous Approach to Object Pose Estimation based on Object Templates

Master Thesis by Sebastian Müller, Department of Computational Engineering
February 1st, 2017
Supervisors: Prof. Dr. Oskar von Stryk, Dr. Alberto Romay
Simulation, Systems Optimization and Robotics Group

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Sebastian Müller
Matriculation number: 2916354
Major: M.Sc. Computational Engineering

Master-Thesis
Topic: A Semi-Autonomous Approach to Object Pose Estimation based on Object Templates

Submission date: February 1st, 2017

Supervisors: Prof. Dr. Oskar von Stryk, Dr. Alberto Romay

Prof. Dr. Oskar von Stryk
Fachgebiet Simulation, Systemoptimierung und Robotik
Fachbereich Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt

Abstract

Object pose estimation is a widely appearing and challenging problem and thus many different approaches have been proposed in recent decades. Using three dimensional registration of surfaces is a promising technique to solve this problem.
While the surface alignment problem has even been solved globally optimal by now, the issue of a high computational complexity remains.
In practice, surface registration often appears in applications that are partially solved by a human user. By making use of basic human skills, the computational complexity can be reduced in order to be applicable to time-dependent scenarios.
While object recognition is an easy task for humans, it remains difficult for computers. However, pose manipulation of objects is time-consuming for humans but can be solved efficiently by computers. By combining the strengths of both, the human and the computer, pose estimation can be solved semi-autonomously in a quick and robust manner.
This thesis proposes an approach that relies on an initial guess and prior knowledge of the shape of the object as input that can be given by a human supervisor or be available to the robot a-priori. Thereby, the complexity of the surface alignment that will be used for the pose estimation can be reduced.
In contrast to existing state-of-the-art methods that are mostly either only locally convergent or are based on complex optimization methods that guarantee global optimality, the proposed method implements a heuristic search strategy that evaluates samples from a search space and chooses the best sample in the end.
By combining this strategy with the initial guess, pose estimation can be solved rotation-invariant in feasible time which makes the algorithm promising for user-interactive real-time applications.
The algorithm is initialized with a rough estimate of the relative position between the two surfaces. After completion, it returns an estimate of the pose, consisting of rotation, translation and of the relative scale between the two surfaces. Although no optimality guarantee is incorporated, the user can easily give feedback on the result's correctness.
The algorithm operates on point clouds which can be easily obtained from any surface representation by a sampling procedure. The 3D-registration problem is then solved by defining a search space and creating a sophisticated version of the Iterative Closest Point (ICP) algorithm that is able to evaluate the search space locally, deal with partial overlap and integrate a scaling property. The algorithm aims to find the global optimum by using a custom-built search strategy.
As a test case scenario in this thesis, the algorithm is applied to pose estimation of objects in the environment of so-called avatar robots that are partially controlled by a human supervisor that takes control of high-level planning tasks.
Therefore, the algorithm is integrated in an existing user interface so that the supervisor is able to easily identify an estimate of the position of desired objects and to give feedback. In order to provide a point cloud, the objects have to be pre-known and ad-

ditionally, custom-built templates of the desired objects have to be available. Thereby, the supervisor is able to define specified object-templates and an initial estimate of the object's position in the 3D-sensor data of the robot.

Contents

1 Introduction

Object pose estimation is a highly important task for modern robotics which can be solved by various approaches. This thesis will utilize surface alignment in order to provide a semi-autonomous solution for the problem.
A novel approach will be presented that is based on prior knowledge, for example an initial guess of a human user. Thereby, the complexity of the surface registration problem can be reduced in order to be applicable in time-dependent scenarios.
The provided algorithm will then be applied to pose estimation for avatar robots that are partially user controlled. A rough initial guess of the supervisor will be used as prior knowledge.
Object recognition is an easy task for humans and by combining it with computational power for surface alignment, the efficiency of user-dependent pose estimation can be strongly improved.

1.1 Surface Alignment

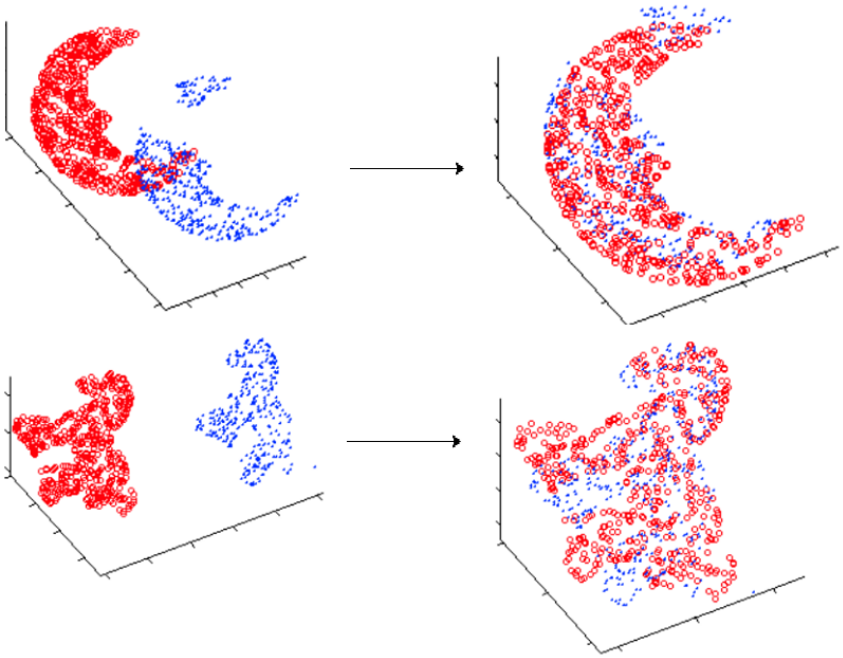


Figure 1.1.: Examples for Surface Registration
On both left images, two surfaces in point cloud representation can be seen. By estimating the transformation, one surface can be projected onto the other as depicted in the images on the right.
(Adapted from Xiong et al [1])

The registration of surfaces is a widely appearing and challenging problem that has its roots in early computer vision in the alignment of curves in two dimensional images. Although many different approaches have been proposed in recent decades, the solutions depend strongly on their application and are still subject to computational limitations.
The surface registration problem describes the search for a transformation that maps one surface in an arbitrary representation onto another surface. Mostly the two surfaces represent at least partly a common object and the resulting transformation projects corresponding parts of the object onto one another. Two examples for the surface registration problem are depicted in figure 1.1.
Efficient methods have existed for a long time that are capable of aligning two surfaces in a locally convergent manner. Therefore, the surface alignment problem can be solved efficiently and accurately if an initial estimate is known. Many approaches were made that used heuristics and other techniques to aim for global convergence until the problem was first solved globally in 2013. However, the computational costs of globally convergent algorithms are too high for many applications.
Some of the applications that depend on surface registration are interactive and partially user-controlled. This thesis presents an approach that makes use of known algorithms and extends them in a way that supervision and control by a human user is exploited in order to shorten computation time.
On the one hand, object recognition is a highly challenging task for machines while humans solve it even subconsciously. One the other hand, pose manipulation of objects remain costly and time-consuming for humans. By combining the advantages of computers and humans, efficient surface alignment can be achieved.

1.2 Context

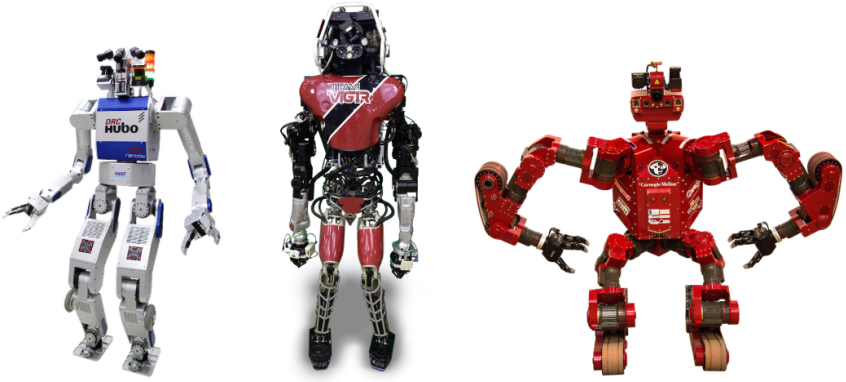


Figure 1.2.: Semi-Autonomous Avatar Robots
From left to right: DRC Hubo (DARPA), Atlas (Boston Dynamics), Chimp (DARPA)
(Source: A. Romay [2])

The proposed algorithm can be applied, for example, to object recognition for avatar robots.
Avatar robots are a category of robots that can be seen as a trade-off between purely-teleoperated and fully-autonomous robots. While the motion planning of body parts of humanoid robots can be solved autonomously by state-of-the-art techniques, tasks like perception, recognition and interaction with the environment remain difficult for fully-autonomous robots. Furthermore, communication with the robot is difficult in many cases due to a limited bandwidth or a high signal latency.
Avatar robots are an approach to deal with these limitations by using a human supervisor to adopt certain high-level planning tasks. The robot uses the relatively simple commands of the user to create complex behaviours and therefore reduces the communication between the robot and the supervisor. Three prevailing avatar robots are depicted in figure 1.2.
In order to make the interaction between the avatar robot and the human supervisor possible, an interface that allows to transfer the high-level commands to the robots has to be provided. This interaction allows the avatar robot to perform highly complex tasks that cannot be fulfilled completely autonomously and also permits the human supervisor to focus on the problem rather than monitoring and commanding relatively simple actions of the robot. Figure 1.3 illustrates the interaction between the supervisor and the robot.
In recent years, avatar robots have emerged as an important tool by allowing the execution of complex tasks in various remote environments, such as post-disaster and hazardous scenarios, medical assistance or planetary exploration. Proceeding progress in robotics allows for more complex applications which demand further improvement in matters of autonomy of the robot and reduction of supervisor tasks.
Object recognition is an important tool for the robot in order to interact with its environment that is currently at least partially solved by the human supervisor in most cases.
Interpreting this data autonomously is a non-trivial task. In order to interact with different objects in the environment, the robot has to recognize and identify the pose (consisting of rotation and translation) of the objects that should be interacted with. A common approach to solve this problem is to use pre-constructed templates of pre-known

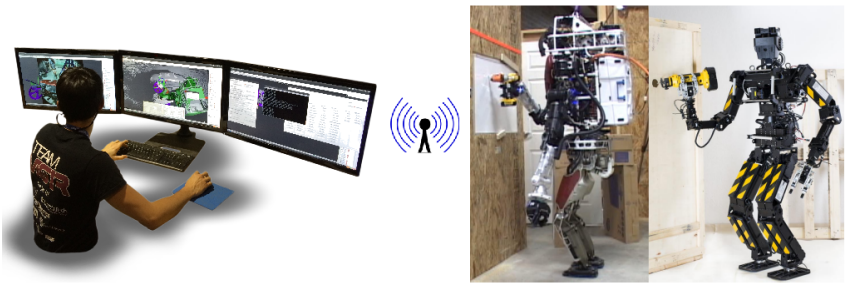


Figure 1.3.: Illustration of the interaction between the Human Supervisor and the Avatar Robot
(Source: A. Romay [2])

objects and align these with the scan data of the real object and thereby identifying the pose of the object.
These object templates can be seen as an abstract representation of the surface of an object. By aligning the template with the corresponding object in the sensor data of the robot, the object and its pose in the environment of the robot can be explicitly identified. The supervisor can solve the alignment between the template and the sensor data by hand.
The purpose of this thesis is to enhance pose estimation and reduce the necessary interaction of the supervisor by introducing a novel approach for semi-autonomous surface alignment and applying it to pose estimation of object templates.

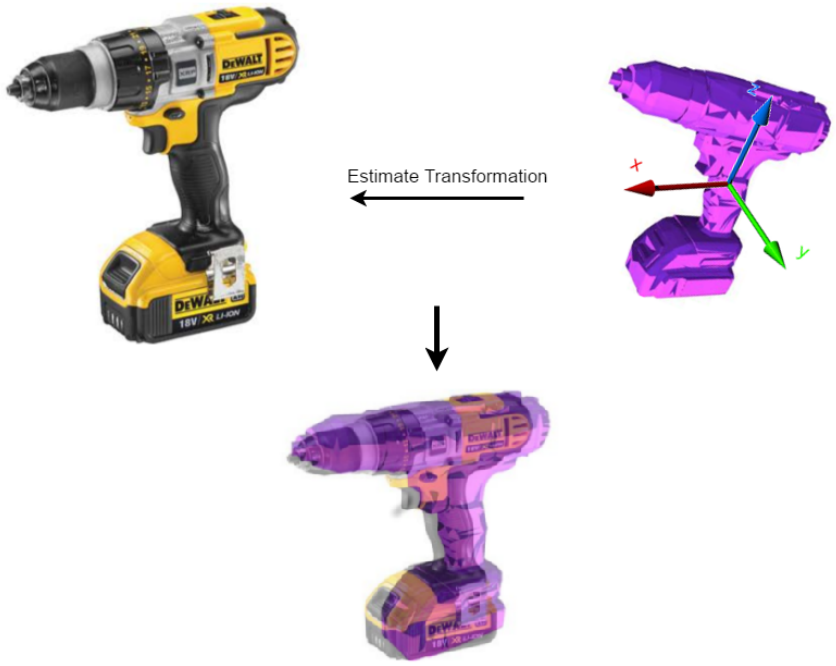


Figure 1.4.: Template-based Pose Estimation
Example of a template-based pose estimation on the basis of a drill: By estimating how to transform a pre-defined template of the drill (upper right) in order to match the real drill in the scan data (upper left), the pose of the drill in the environment of the robot is defined (bottom).
(Adapted from A. Romay [2])

1.3 Contribution

This thesis proposes a novel approach that extends state-of-the-art methods for surface alignment and aims for finding a correct registration estimation invariant of the initial rotation in feasible time.
The proposed algorithm will be specifically designed to allow for good applicability to pose estimation with object templates. By utilizing the object recognition abilities of the

supervisor, prior information about the position of the object template in the sensor data can be used to reduce the computational complexity of the surface alignment algorithm. Thereby, the effort of the human supervisor can be reduced and the pose of a pre-known object can be estimated semi-autonomously in a quick and robust manner.
The supervisor simply has to provide a rough guess of the position of the object and select the desired object template. The algorithm then calculates and returns the estimate of the pose of the object.
The extension of existing surface registration methods is crucial because no alternatives exist that allow for semi-autonomous alignment of surfaces in feasible time.

1.4 Challenges

This approach faces various challenges, such as non-perfect templates or noise caused due to the sampling procedure of the environment by the sensors. Furthermore, only a partial overlap of the template with the object in the scan data can be expected because in general, the sample view is only generated from a single perspective which leads to a hidden side of the sought-after object. The visible part of the object can also happen to be only a small portion of the surface of the object or consist of two separated sub-surfaces if the sensors are targeted towards the object from an unfavourable perspective. Occlusions by other objects are also common and may shadow a huge part of the object. Another problem to face are self-occlusions from the robot itself because interaction with an object often obscures a large portion of the very same object.

1.5 Outline

The rest of this thesis is structured as follows:

Section 2 gives a short overview of the most famous approaches for surface registration and reviews them in regard to their applicability to the presented problem. Afterwards, current solutions from some research groups for the semi-autonomous manipulation of objects will be presented.

Section 3 presents the algorithmic concept to solve the proposed problem.

Section 4 explains the requirements that must be satisfied in order to use the algorithm from section 3.

Section 5 describes implementation issues of the proposed algorithm.

Section 6 explains the integration of the implemented algorithm in the utilized set-up and presents the user interface.

Section 7 gives a performance evaluation of the implemented algorithm. It will not only be tested on different objects with simulation data from various perspectives, but also on sensor data captured by a real robot with different objects.

Section 8 will discuss the results, review fulfilment of the mentioned challenges and give an outlook on open problems and possible solutions.

2 State of Research

This section will present an overview of current literature and review famous approaches for surface alignment. Afterwards, a short insight to solutions for template alignment for object manipulation of robots groups will be given.

2.1 Surface Alignment Approaches

Registration of surfaces or curves in various representations is a challenging problem and has appeared in a broad range of applications and therefore many possible registration approaches have been proposed. The representation of the two surfaces may vary, whereas most approaches are easiest implemented with sampled point clouds from the two objects but representations like implicit functions or Computer Aided Design (CAD) models are also possible. Most algorithms work with dense objects as well as with surface representations, whereas the latter is much more common due to the surface sampling character of most sensors.
This thesis will focus on surfaces in point cloud representations because they are most common and other representations can easily be converted to point clouds by sampling. The reviewed methods aim to estimate the transformation parameters that project one point cloud onto the other. The point cloud to which the transformation is applied will be referred to as source point cloud, while the point cloud onto which the source cloud is projected will be called target point cloud.
The transformation parameters can be of different form but mostly appear in their easiest variant as a homogeneous transformation matrix that encodes rotation, translation and can for example be extended to include scaling.
In the following, different surface registration techniques will be described and reviewed with regard to their applicability to the presented problem.
The reviewed methods will be split into two categories: First, local surface alignment approaches that will only find a local optimum and thus rely on a rough initial estimate of the transformation parameters to work properly will be presented. Afterwards, more sophisticated approaches that aim for global convergence and thus work without an initial estimate of the transformation parameters will be reviewed.

2.1.1 Local Point Cloud Alignment

Iterative Closest Point (ICP) Algorithm

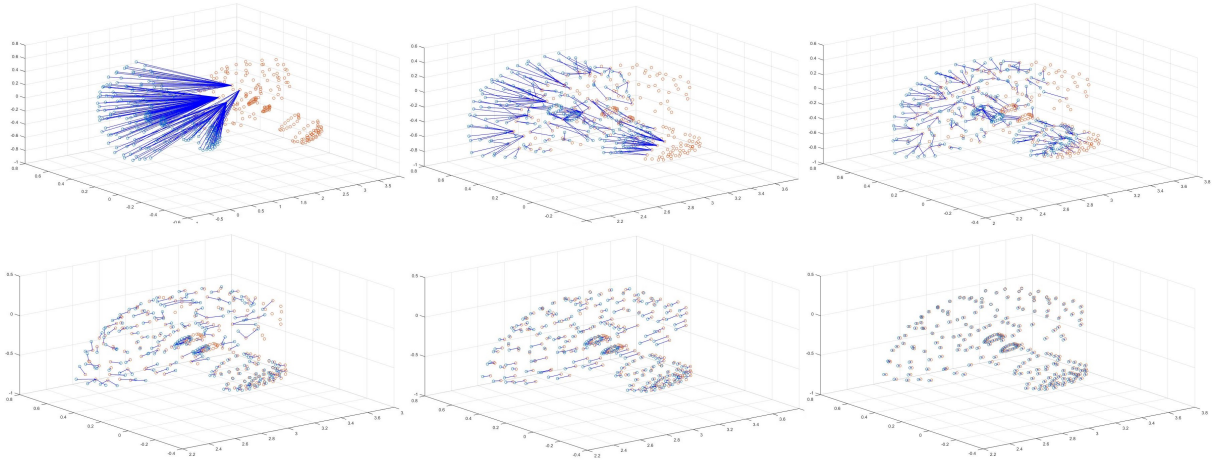


Figure 2.1.: Visualization of the ICP Algorithm
The ICP algorithm is applied to two mushrooms (red and blue points). The blue lines visualize the correspondences in the depicted steps. The images show the algorithm in iteration 1, 3, 8, 13, 20 and in the final iteration 27.

The most common approach for local point cloud alignment is the so-called Iterative Closest Point or Iterative Corresponding Point (ICP) algorithm that has been developed by Besl & McKay [3] and Chen & Medioni [4] independently and has since then appeared in many different versions.
The ICP algorithm defines a cost function that is iteratively minimized until a local optimum is reached. In each iteration, the nearest neighbour in the target point cloud is assigned as a correspondence for each point in the source point cloud. The cost function is defined as the sum-of-least-squares between the source points and their correspondences. The parameters are estimated by minimizing this cost function.
This can be understood as an Expectation-Maximization (EM) approach that consists of two steps. In the E-Step of the algorithm, the correspondences are assigned and in the M-step, the distance between the source points and their correspondences is minimized. These two steps are alternated until the ICP algorithm has reached a local minimum. A visualization of the ICP algorithm's execution can be seen in figure 2.1.
As the correct transformation and correspondences are both unknown, this resembles a chicken-and-egg problem because the transformation depends on the correspondences and the correct correspondences depend on the transformation. Solving one of these problems is easy if the other one is known but having to estimate both at the same time carries many problems due to a highly non-linear and non-convex cost function. Additional difficulties like noise or partial overlap of the two objects and occlusions cause the algorithm to be even more likely to fail and end up in a local minimum of the cost

function.
Over the years, numerous extensions have been proposed in order to make the ICP algorithm more robust and overcome some of its limitations. Most of these changes and improvements can be categorized as follows:

Selection The Selection of the points in both point clouds that are taken into account for the nearest-neighbour search. This is often regarded as a preprocessing step and thus not considered as part of the ICP algorithm. Subsampling as proposed in [5] can speed up the classical ICP algorithm by removing redundant information. This can for example be achieved by using the surface normals as a measure for the similarity of points as proposed by [6].

Matching The matching of the source points with their correspondences. In most cases, this is simply achieved by using the nearest neighbours but it is also possible to use additional information. In [7], the matching is carried out in a six dimensional space, where three dimensions represent the standard euclidean dimensions and the other three encode additional colour information. The distance between different points is still measured with the sum-of-least-squares. Since the most expensive part of the ICP algorithm is the correspondence-matching, data structures with fast look-up times like kd-trees are used in most implementations. It is also possible to use a sophisticated version of a kd-tree that exploits the properties of the ICP algorithm as described in [8].

Weighting Some variants of ICP weigh corresponding point pairs differently, for example by calculating the surface normals of points (by assuming locally planar surfaces and taking their neighbours into account) and by assigning correspondences with similar normals a higher weighting factor.

Rejection Rejection of correspondences (assign them a weight of zero) is especially important for outlier detection. It is possible to reject correspondences with a distance above a certain threshold or to reject a fixed percentage of correspondences with the biggest distances. Chetverikov et al [9] proposed a method in 2002 called *Trimmed ICP* that only takes a fixed percentage of source-correspondence pairs with the smallest distances into account. Zinsser et al [10] introduced an approach called *Picky ICP* that works with arbitrary overlapping percentages and that only uses pairs with a distance lower than a certain threshold for the minimization step and rejects the remaining pairs.

Error Metric While the standard ICP only uses the sum-of-squared-distances between the correspondences as error metric, more sophisticated approaches like the *point-to-plane* [11] and *Generaliced ICP* [12] algorithm exist that also take neighbouring points of the correspondences into account. The point-to-plane ICP algorithm minimizes the projection of a source point onto the approximated surface of the target cloud. The Generalized ICP algorithm regards all points as samples drawn from

an underlying probability distribution that are afflicted by noise. By using a full probabilistic approach, the Generalized ICP algorithm can be regarded as a plane-to-plane ICP that minimizes the distances between approximated planes of the source and target object.

Minimization Closed form solutions based on *Singular Value Decomposition* (SVD) for the standard ICP algorithm exist. More complex ICP variants require numerical optimization algorithms such as the *Levenberg-Marquardt*-algorithm to estimate the transformation parameters.

Rotation Representation Different rotation representations are possible, such as the Cardan, Euler or the Angle-Axis representation. Another famous representation are the so-called quaternions which avoid redundant angle representations.

Non-rigid extensions There are also approaches that allow the target point cloud to change shape (e.g. necessary for face detection with ICP).

Scaling Property Extensions which include a scaling parameter that allows to search for a target that is of different size than the source object have also been proposed. Approaches following this idea have been introduced by [13] and [14].

Performance Evaluation

Comparisons of different point cloud alignment approaches for locally convergent registration can be found for example in [15], [16], [17] and [18]. Many evaluations focus on the accuracy of the different approaches, whereas only few can be found that compare the convergence rate of the algorithms which is more relevant to the problem this thesis aims to solve. A short review of a convergence comparison of different algorithms from Salvi et al [18] that is visualized in 2.2 will be given here.
Five different versions of the ICP algorithm and their behaviour when executed on the same scene are depicted in the graph. The tested algorithms are the standard ICP algorithm, the Picky ICP algorithm (that only takes pairs into account whose distance is lower than a certain threshold) as well as the point-to-plane ICP algorithm. Salvi et al also evaluate two ICP versions that have not yet been mentioned here. The first one extends the ICP algorithm by using a genetic algorithm approach. The second one focuses on minimizing the least-median-of-squares instead of the sum-of-squared-distances.
The standard ICP converges slightly faster than the point-to-plane version, whereas the latter one is more accurate. The Picky ICP converges slower and ends up with a smaller error. The least-median-of-squares approach converges fastest, while the genetic method takes the longest time to converge. These two also end up with the highest mean-squared error.
It is important to note that this graphic only considers the number of iterations needed, not the duration of single iterations. Algorithms whose optimization step can be sped

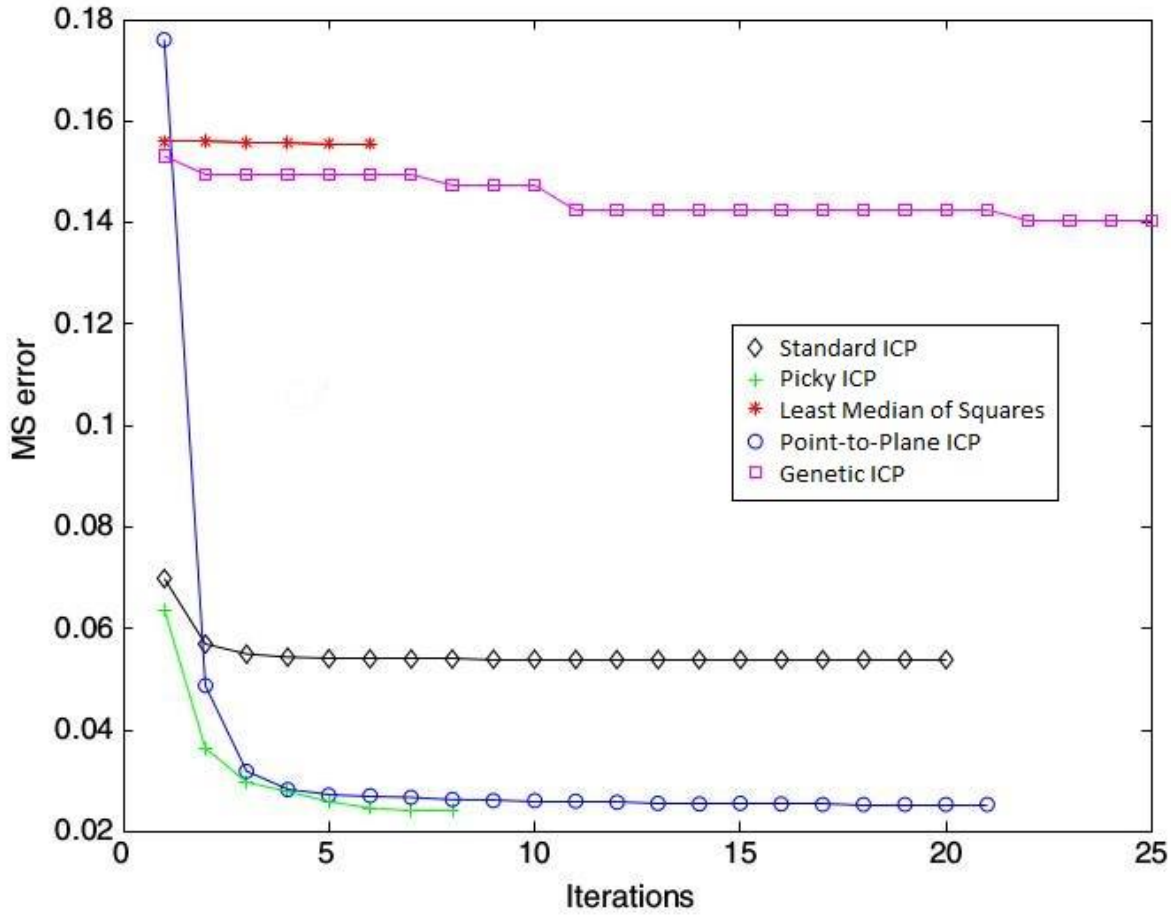


Figure 2.2.: Comparison of Different Local Point Cloud Alignment Algorithms
Convergence behaviour of different versions of the ICP algorithm. The number of iterations is depicted on the horizontal axis and the corresponding mean-squared-error on the vertical axis.
(Adapted from [18])

up by using closed-form solutions, can easily outperform algorithms where this is not possible in terms of execution time. In addition, algorithms with simpler cost-functions can also be extended easier, for example by adding a scaling property.

Limitations of the ICP Algorithm

As reviewed in the previous section, numerous variants of the ICP algorithm exist that improve different parts of the algorithm. Extensions for different problems, such as partial overlap of the source and target point clouds or the extension to a scaling property, have been proposed.
However, all these variants of the ICP algorithm have in common that they are extremely likely to get stuck in a local minimum. Therefore, a good initial estimate of the transformation parameters is necessary in order for the algorithm to work properly and converge

to the global optimum of the parameters.
The standard version of the algorithm defines a highly non-linear and non-convex cost function due to the two-way dependencies of the transformation and the correspondences which are both a-priori unknown. Extensions for dealing with a scaling parameter and partial overlap of the point clouds make the cost function even more difficult to solve and create more local minima.
Versions of the ICP algorithm, such as the Generalized [12] or the point-to-plane ICP algorithm [11], exist that try to improve the behaviour of the algorithm by using a more complex cost function but they come at the cost of higher optimization complexity because closed-form solutions do not exist for these extensions.
There are also solutions that aim to widen the basin of convergence by using multiple correspondences and then assign each one a possibility. The transformation can be estimated by maximizing the likelihood of these multiple correspondences. However, the resulting high computational cost makes them hard to use for real-time applications.
Different approaches have been proposed to solve the problem of avoiding local minima, the most important ones will be reviewed in the next section.

2.1.2 Non-local Point Cloud Alignment

Not all approaches covered here are fully globally convergent or incorporate a convergence guarantee. However, all presented methods have in common that they try to overcome local limitations of basic algorithms.

Correlation-Based Registration

The basic idea of correlation-based techniques is to identify some intrinsic properties in both point clouds and then align them. If these intrinsic properties are transformation-invariant, this is an easy way to estimate the transformation parameters.
A classical solution for fast and initialization-invariant point cloud registration following this idea is based on *Principal Component Analysis* (PCA) and was introduced by Yambor et al [19] in 2000. The used intrinsic parameters are the eigenbases of the point clouds. The alignment is carried out as follows: First, the covariance matrices of the normalized clouds are calculated. Afterwards, the eigenvectors and eigenvalues of these matrices are computed. The three eigenvectors represent a basis, the before-mentioned eigenbasis. This eigenbasis can be understood as the directions with the greatest variances of each point cloud. The eigenvector with the largest eigenvalue thus corresponds to the direction with the largest variance of the points. Figure 2.3 shows two example point clouds and their corresponding eigenbases.
The transformation matrix that projects one of the eigenbases onto the other is also the transformation matrix that can be used for projecting one point cloud onto the other.
If the source and target point cloud are identical, this approach will find the global optimum directly. However, in almost all cases the objects are sampled and consist of

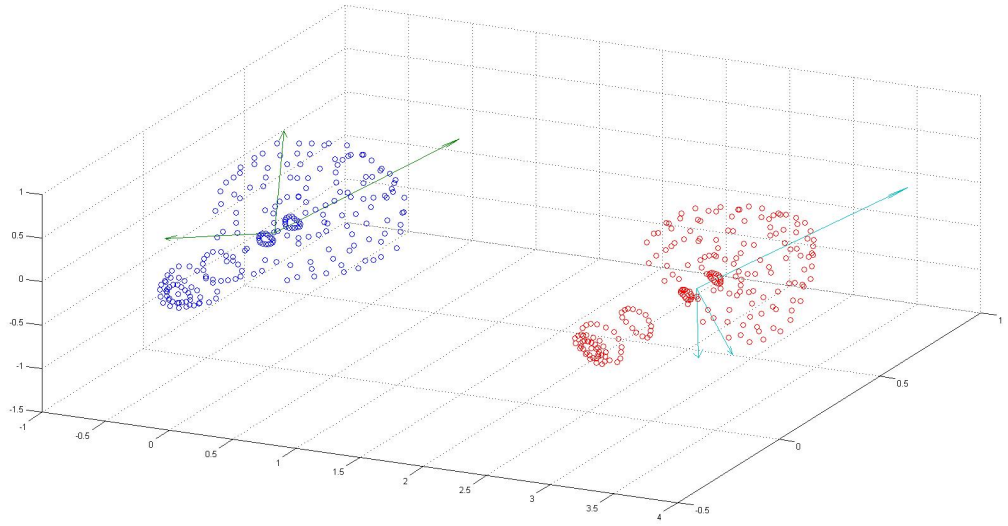


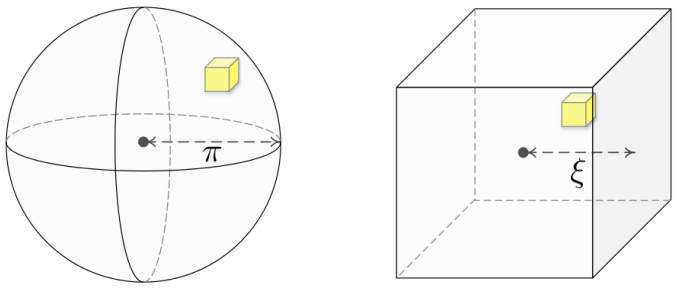
Figure 2.3.: Two Point Clouds and Their Eigenbases
The image depicts two example point clouds of a mushroom in dark blue and red. The corresponding eigenbases are visualized as three vectors in green and light blue respectively.

different points covered by noise. This approach may still be helpful when using the result to get a rough initial estimate of the transformation parameters and afterwards use them as initialization for a standard ICP algorithm.
However, this approach is still very sensitive to outliers because even a few outliers can have a severe impact on the covariance matrix. If the two point clouds only partially overlap each other and thus the two point clouds represent different parts of the surface of the object, the covariances do not have similar eigenbases any more.
The PCA-based approach can be understood as fitting each of the point clouds as a single Gaussian distribution and then aligning their means (translation) and covariances (rotation). To overcome the sensitiveness to outliers, Xiong et al [1] proposed a more robust extension of this approach in 2013 called *Kernel Feature Maps* that was inspired by kernel methods in machine learning.
Instead of fitting a Gaussian in the original three dimensional space, the points are implicitly mapped to a much higher dimensional Hilbert space, where a single Gaussian distribution can fit well which makes the technique more robust to outliers. The transformation is not computed explicitly, but is implicitly encoded in the applied kernel PCA. This method shows promising results and outlier robustness in feasible time.
However, if the point clouds only partially overlap each other and structures of outliers (e.g. parts of the environment) are part of the target point cloud, the Kernel Feature Maps method is still not very promising.

Another family of algorithms follows a probabilistic approach that regards the points of the clouds as samples drawn from an underlying probability distribution.
One of the algorithms following this idea is called *Soft Assign* and was introduced by Jian et al [20] in 2011.
Probably the most important approach from this family is the so-called *Coherent Point Drift* (CPD) [21]. It can be regarded as a probabilistic method that solves a density estimation problem. CPD approximates the two point clouds as a probability density with a *Gaussian Mixture Model* (GMM) and afterwards the distance, defined by the *Kullback-Leibler-Divergence*, between the two probability distributions is minimized in order to estimate the transformation.
Although the method has shown good results in case of outliers, noise and missing points with a higher accuracy than other approaches, its computation is very time consuming which makes it impractical to use for real-time applications. An improved version approximating the CPD for better runtime has been proposed by Lu et al [22] in 2016 but the improvements are not sufficient to regard the algorithm as a serious candidate for real-time applications.
In addition, the density estimation nature of the algorithm complicates the extension to point clouds with partial overlap.
A performance comparison of the CPD with other state-of-the-art techniques can be found in section 2.1.2.

Branch-and-Bound (BnB)-Based Registration

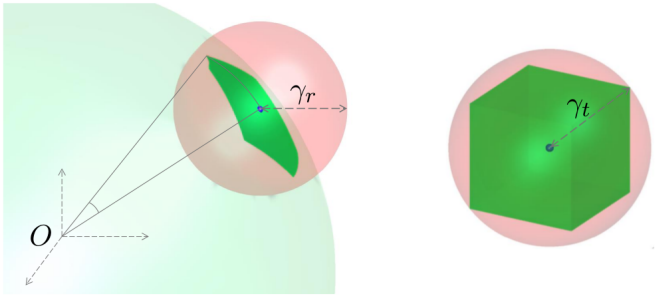
In recent years, a group of algorithms was developed that aims to give a convergence guarantee to the global optimum of the transformation parameters.
These algorithms are based on *Branch-and-Bound* (BnB)-optimization. Surveys on BnB-optimization can be found in [23] and [24].
The basic idea of BnB-optimization can be summarized as follows: If a higher and a lower bound on the derivative of the cost function is known, certain regions of the parameter domain can be excluded from the possibility that they may contain the global optimum and do not have to be searched in. Furthermore, if the function value of a point in the domain is known and a current best value exists, a certain area around the evaluated point must have smaller function values than the current best point. If all parts of the domain are excluded this way, the current best point is the global optimum.
BnB-optimization was first applied to point cloud registration by Li and Hartley [25] in 2007 but their approach made unrealistic assumptions (equally large point clouds, no outliers) which makes their method not very useful in practice.
Yang et al [26] extended the approach in 2013 and proposed a methoalled *Globally Optimal ICP* (GO-ICP). By deriving a higher and lower bound on the sum-of-least-squares cost function and using the BnB criterion, they developed an algorithm that is able to



(a) Rotation domain (b) Translation domain

Figure 2.4.: Parameter Space of the GO-ICP Algorithm
Left: Rotation parameter space in Angle-Axis representation: All possible rotation values lie inside the sphere with radius π. Right: Translation parameter space: All possible translation parameters lie inside the cube with edge length ξ. (Source: [26])

solve the point cloud registration problem globally in rotation and translation space and provide a convergence guarantee.
GO-ICP uses a standard version of ICP as function evaluation that takes the six rotation parameters as arguments.
The parameter domain is parametrized as follows: The translation space can be understood as a three-dimensional cube that contains all possible values of the translation parameters lower than a certain value. This value can simply be chosen as the maximum distance between two points from the two point clouds. The rotation space can be parametrized as a cube with radius π by using the Angle-Axis representation for rotation. The two parameter spaces are depicted in figure 2.4.
The rotation and translation space are both divided recursively with an octree and the sub-cubes at the branches of the tree are cut if the BnB-criterion is fulfilled. This is the case if the radius around the point that cannot contain a better point than the current best point completely surrounds a sub-cube of the octree. The mentioned radius is referred to as uncertainty radius. A visualization of this issue can be found in figure 2.5.
The GO-ICP algorithm uses a nested BnB-algorithm to avoid having to search a six-dimensional (three rotation and three translation parameters) parameter space. First, the global translation optimum is searched for a given rotation and afterwards, the optimal rotation is searched.
The GO-ICP algorithm works with a priority-queue that contains all cubes or branches of the tree that could contain the global optimum, sorted in increasing order by their lower bound on the cost function. If the cube with the lowest lower bound is closer to the current best found value than a certain threshold, the algorithm is stopped and the current estimate is returned as global optimum.
Otherwise, the first element of the list gets extracted, evaluated and divided into eight sub-cubes that then get inserted into the priority queue.
As mentioned before, the GO-ICP algorithm is able to solve the point cloud alignment registration problem globally optimally and it can be easily extended to other local



(a) Rotation uncertainty radius (b) Translation uncertainty radius

Figure 2.5.: BnB-Criterion of the GO-ICP Algorithm
The uncertainty radius (γ_r for the rotation and γ_t for the translation) depicts a region of the parameter space that has been excluded from the probability of containing the global optimum. If this radius completely surrounds a sub-cube of the octree that divides the parameter space, the corresponding branch can be cut. (Source: [26])

variants of ICP and thus deal with problems, such as partial overlap and occlusions. Unfortunately, GO-ICP shows long runtimes which will be evaluated in the following section.
The idea of GO-ICP has also be transferred to probabilistic models using GMMs as described in section 2.1.2. Campbell and Petersson [27] introduced a new method called *Globally Optimal Gaussian Mixture Alignment* (GOGMA) in 2016. The parameter space is parametrized in the same way as for the GO-ICP algorithm but the function evaluation is done with a Gaussian Mixture Alignment instead of an ICP execution. The upper and lower bounds are adapted accordingly.

Performance Evaluation

Campbell and Petersson [27] present a performance evaluation where they test their proposed GOGMA-algorithm and compare it to other point cloud alignment approaches. The results of their tests are depicted in figure 2.6.
In the table, six columns can be seen where each column represents the result of one of

Method	GOGMA	GOGMA+	GO-ICP	GO-ICP+	ICP	CPD
Translation	0.26	**0.04**	1.63	1.17	4.67	5.24
Rotation	1.25	**0.32**	30.9	19.4	107	88.8
Inlier % (C)	**100**	**100**	71.8	80.9	15.5	38.8
Inlier % (M)	**100**	**100**	48.5	51.9	13.4	28.6
Inlier % (F)	80.0	**99.7**	19.6	21.2	6.5	7.1
Runtime	49.6	71.2	31.6	103	**0.38**	4.2

Method	GOGMA	GOGMA+	GO-ICP	GO-ICP+	ICP	CPD
Translation	0.72	**0.13**	1.33	0.69	7.37	8.13
Rotation	3.09	**0.68**	9.66	5.19	109	90.7
Inlier % (C)	**100**	**100**	78.2	84.1	11.3	39.5
Inlier % (M)	75.0	**99.9**	36.6	64.5	10.8	19.3
Inlier % (F)	16.7	**99.9**	13.2	27.5	5.4	0.8
Runtime	29.5	49.6	26.2	77.7	**0.44**	4.2

Figure 2.6.: Comparison Between Different Point Cloud Alignment Approaches
(Adapted from [27])

the algorithms. The first two columns represent two different versions of the GOGMA algorithm, the third and fourth GO-ICP versions with $\epsilon = 1e^{-4}$ and $\epsilon = 1e^{-5}$ where ϵ denotes the tolerated error of the global optimum. The last two columns represent the standard ICP algorithm and the CPD algorithm.
The tests are executed on two point sets. The first one depicted on the left is called STAIRS, the second one on the right is called WOOD-SUMMER.
In the first two rows, the average absolute error of translation (in metres) and rotation (in degrees) can be seen. In the three rows beneath, the results are categorized in the three categories coarse (C), medium (M) and fine (F) that describe the percentage of executions that achieved a certain accuracy. Coarse allows an error of 2m/10°, medium an error of 1m/5° and fine an error of at least 0.5m/2.5°. In the last row, the average runtime is depicted.
It can be seen that in both cases the ICP algorithm performed poorly but has by far the shortest runtime. The CPD algorithm succeeds in more cases but has a runtime that is much higher than that of the standard ICP. GO-ICP solves about 80% of the executions coarsely and the GOGMA-algorithm even full 100% of the tests.
However, both GOGMA and GO-ICP suffer of a very high computational cost that makes them impractical for many time-dependent applications.

2.2 Pointcloud Alignment for Object Manipulation

In this subsection, current solutions for object manipulation will be presented.
Most semi-autonomous approaches for object manipulation are based on three dimensional models that are located in the environment by a supervisor. These models define different properties of the object, such as grasping or stance information that are used to create the trajectories of the robot. Symbolic level approaches that define manipulation instructions are also possible.
The first approach presented here is called the *Affordance Template (AT) ROS Package* and was published by TRACLabs and NASA in 2015 [28].
By providing an environment that allows a human user to modify the pose of these templates, the objects in the environment can be identified explicitly by manually adjusting the template to the according object in the sensor data.
An approach called *Online Affordance-based Perception* published by Fallon et al from the MIT [29] in 2014 implements an approach for the semi-autonomous alignment of templates, that was used at the DARPA Robotics Challenge (DRC).
The research group also developed another fully-autonomous approach for a different contest, but the required time for human input was short compared to the time lost due to planning the robot motion with a poorly fit affordance model.
The semi-autonomous approach relies on user input that provides a one- or a two-dimensional input (e.g. a point or a line) that specifies a search region. A chosen template is then aligned to the best fit in the local environment automatically. Figure 2.7 depicts an example for this procedure.

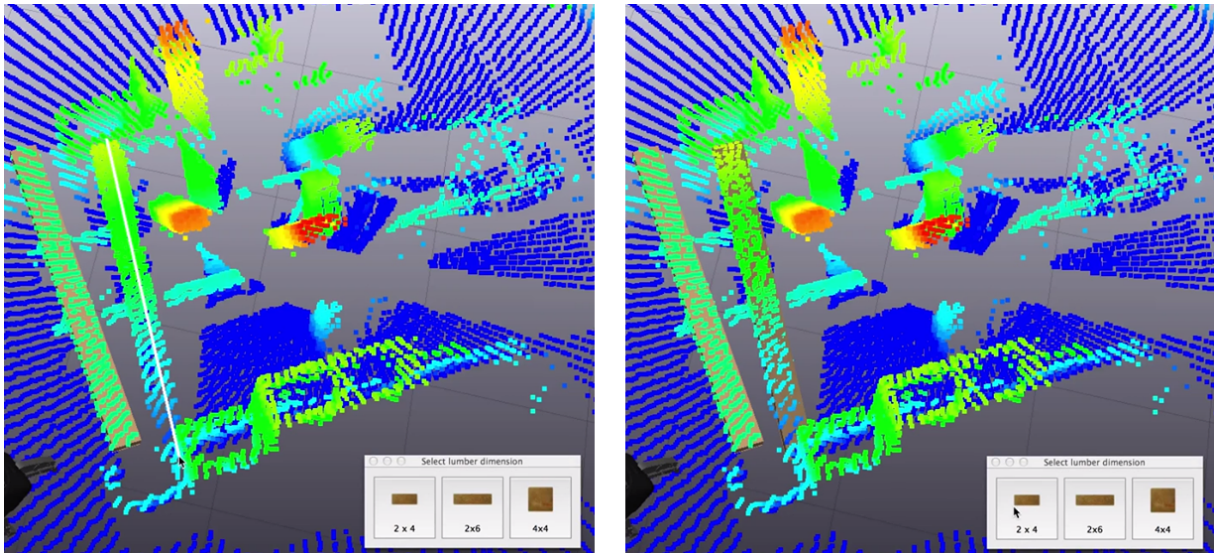


Figure 2.7.: MIT Perception Approach for the DRC
Semi-autonomous template matching: The operator specifies a template from the box in the bottom right corner and a search region by drawing a line (as it can be seen in the left image). Afterwards, the template is aligned automatically.
(Source: Screenshots taken from a youtube video [30])

2.3 Discussion

This thesis will present a semi-autonomous approach that makes use of techniques for surface registration in order to improve the template-based pose estimation used by state-of-the-art methods for object manipulation in robotics.
Affordance-based approaches rely on the registration of pre-build templates with their correspondences in sensor data. Designing a specifically adapted surface alignment algorithm that makes use of human object recognition abilities and computational efficiency can simplify and improve this alignment.
State-of-the-art methods for surface registration mostly feature only local convergence properties or suffer of high computational costs by aiming for global convergence in terms of translation and rotation.
For semi-autonomous template alignment, global translation convergence is not very important because the position of object templates in a three dimensional virtual environment can relatively easy be determined. The same accounts for the scaling factor. Rotation manipulation of objects on the other hand is time-consuming for humans.
This suggests to utilize a surface alignment technique that operates invariant of the rotation, but is only locally convergent in terms of translation and scaling in order to best trade-off human and computer abilities.
Therefore, a custom-built method which uses a search strategy that evaluates possible rotations in order to find the desired solution will be used. The evaluation can be achieved

by using a locally convergent algorithm, as it has been proposed in the literature.
The search strategy relinquishs convergence guarantees but features a high success probability. This is sufficient because if a human user monitors the results, falsely aligned surfaces can be identified and corrected easily.
This could not be achieved by using existing methods because locally convergent algorithms require time-consuming rotation manipulation by humans and techniques that offer a convergence guarantee are too computationally expensive.
The proposed method will trade-off both sides in order to improve the speed and quality of semi-autonomous template-based pose estimation.

3 Concept

This thesis presents an approach for object pose estimation that uses pre-known shape information of objects and an initial input of rough position and scale estimation in order to accurately estimate the pose and scale of an object. In this thesis, the problem is approached by the alignment of two surfaces in order to determine the pose of one of the surfaces relative to the other.
In general, surface alignment describes the problem of estimating a transformation that projects one surface onto another, so that parts of a common object are mapped together. To be applicable to pose estimation, the problem will be extended to deal with partial overlap, noise and to incorporate a scaling factor.
While various approaches for surface registration have been proposed, many applications require specifically adapted and optimized solutions. Many of these applications appear in user-interactive scenarios. Therefore, by restricting to these scenarios, the human ability for object recognition can be utilized in order to improve the performance of surface alignment algorithms.
The proposed algorithm follows this idea and will therefore be designed to rely on an initial guess and thereby shorten computation time.
In a human supervised approach, an appropriate interface should allow the user to be able to efficiently specify a rough estimate of the position and scaling factor. Since manipulation of the orientation of object templates in a three dimensional virtual environment is often difficult and time consuming, the algorithm will be designed to operate independently of the initial rotation, while relying on the initial guess and providing only locally optimal properties in the case of the translation and scaling factor.
The basic concept of the proposed algorithm can be understood as follows: A locally convergent algorithm will be constructed by using existing surface alignment techniques that will be able to deal with the defined requirements. By discretizing a rotation search space that encodes all possible orientations of a surface, the locally convergent algorithm can be used to evaluate samples from this search space.
After a time-dependent stop-criterion has been fulfilled, the algorithm will be terminated and the best evaluated sample will be chosen. The stop-criterion will depend on time to allow for autonomous adaptation to the difficulty of the problem. In cases where alignment between the surfaces is easy, the algorithm can be terminated quicker, while in more difficult cases, more samples will be evaluated to increase the certainty in the correctness of the estimate.
The procedure will be designed and optimized in order to provide a high success probability in feasible time.
With this approach, human object recognition abilities and computational power can be combined and the strengths of both, the computer and the human supervisor, can be utilized efficiently.

4 Requirements

In this section, the different requirements for properly using the proposed algorithm will be explained.
The algorithm takes two point clouds and an initial estimate of the position and scaling factor as input and returns the full pose estimate as result. After discussing the requirements for the source and target point clouds and the initial guess, the requirements for a necessary user interface will be presented. The interface allows the user to interact appropriately with the algorithm. An example for such an interface will also be presented.

4.1 Source and Target Clouds

The source and target objects have to be provided as discrete point clouds. If other representations are used, such as CAD-models or implicit functions, they have to be converted into point clouds, for example by using an appropriate sampling procedure.
The objects are supposed to be provided in a surface representation, whereas dense models are basically possible but will not be tested in this thesis.
The points from the source and target point cloud are usually represented in different coordinate systems, namely the source and target coordinate system. If both objects are represented in the same coordinate system, the initial position, that will be explained in detail in the following subsection, equals zero.
While the orientation of the target coordinate system does not matter directly, the orientation of the source coordinate system relative to the target coordinate system is important. The initial guess of the position is supposed to translate the source cloud so that it is projected onto the target cloud.
The proposed algorithm tries to avoid local minima of the cost function that are caused by symmetrical resemblances in the shape of an object. In order to increase the success probability, it is advised to align the symmetrical-axes of the source cloud with the axes of the source cloud's coordinate system's axes. Thereby, the algorithm will be able to check for better estimates with symmetrically rotated versions of the template.

4.2 Source Object

The proposed algorithm is generally able to deal with arbitrary surfaces but is designed to be applied to template-based pose estimation in particular. Therefore, the source cloud is supposed to resemble the surface of an object, while the target cloud contains a part of the surface of the desired object, as well as parts of the environment.

The quality of the template affects the success probability. It is possible to use imperfect templates, but they can significantly reduce the success rate in some cases.
Some applications also work if the used template only represents parts of the source object, e.g. the handle of a switch. For template-alignment, full template versions of the object will perform better because small parts of objects can often fit well with parts of the environment, too.
The bigger the represented part of an object, the more the desired solution differs from suboptimal solutions.
Then again, if a source object is deformable, e.g. a switch with different positions, it may be helpful to use only the handle of the switch because the template would only match the real switch in specific states.
In summary, templates representing complete object surfaces improve the performance of the algorithm, while deformable objects may be easier aligned with a template that resembles only a part of the object's surface.

4.3 Initial Estimate

The initial guess describes the rough location and scale of the source object relative to the target object. If the source and target objects are represented in the same coordinate system (if the source and target coordinate system are equal), the initial position is set to zero.
The position of the source object is defined as the point relative to which the rotation is applied, namely the origin of the source coordinate system.
If the symmetrical axes are aligned with the coordinate system's axes, as demanded in the previous subsection, this centre point is well defined. For an object with uniform density, this point is equal to its emphasis.
The question of the necessary accuracy of the initial position cannot be answered in general. It depends on many properties, such as the source and target object's shape, the perspective and quality of the sensors, the overlapping percentage, as well as on the environment.
It can be said that the more difficult the problem is, the more accurately the initial guess has to be set. Especially if the target cloud is separated into two sub-clouds due to a disadvantageous perspective, the initial position is required to be very precise. Noise and imperfect or very small templates further increase the problem.
A sophisticated analysis of the necessary accuracy can be found in section 7.2.

4.4 Operator Environment

The implemented algorithm is constructed to operate semi-autonomously and allow for fast initialization and feedback by a supervisor. This suggests that an environment has to be provided that allows a user to efficiently interact with the algorithm.
The environment is supposed to visualize the source and target point clouds and enable

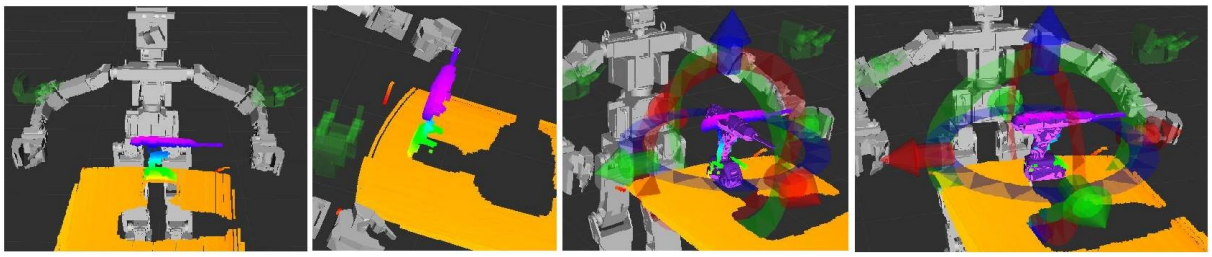


Figure 4.1.: Example of the Alignment Procedure with OCS
The User Interface of the Operator Control Station (OCS). In the first two images, a robot standing in front of a table with a drill on top can be seen from two different perspectives. The third image shows the same setup with a template of the drill. The drill can be inserted at an arbitrary position and the pose of the template can be changed by using the arrows and circles around the drill. By double clicking on the template the alignment algorithm is executed and the template is aligned with the scan data as depicted in the last image.

the operator to choose templates and modify their pose. If the scaling factor is also taken into account, the size of the object should also be manipulable. Manipulation of the rotation is not necessary, but can be helpful in order to allow the operator to use better initial positions of the source object in case the algorithm fails. However, this should only be required in very difficult cases, such as a target object that is split into two sub-clouds. The environment should also enable the operator to simply send a request to the template alignment server and visualize the result to allow for feedback. The operator can immediately see whether the result is correct and if not, start the template alignment again with a different initial position.
An example for an environment that incorporates these requirements is the so-called *Operator Control Station* (OCS). An example of the alignment process with OCS is shown in figure 4.1.
OCS visualizes both point clouds and allows the operator to insert different pre-defined object templates as source clouds to a specified position in the target cloud. Therefore, the source objects have to be pre-known and the corresponding templates have to be provided. The insertion point is chosen via ray-casting from the virtual camera to the surface of the target cloud.
The rotation and position of the template can be changed by using circles and arrows around the template. A support of the manipulation of the scale of objects has not yet been implemented.
If the template has been roughly aligned (which usually is the case directly after inserting the template because the position can be chosen before), the template alignment server can be called by double-clicking on the template. The algorithm is then initialized with the current position of the template. As soon as the server has returned, the result is visualized.

Remark: The OCS environment currently does not incorporate the functionality to change the scale of a template. In order to be able to use this feature in the future, the algorithm will be designed so that it also estimates the scaling property. In order to integrate this functionality, OCS will have to be adapted accordingly.

5 Implementation

In this section, the implementation of the proposed algorithm problem will be presented. The algorithm is supposed to solve the registration problem between two surfaces. The surfaces are represented as point clouds because they are most common, allow to define a good cost function easily and additionally, all other surface representations can be converted into point clouds by a sampling procedure.
The point cloud, which the estimated transformation is applied to, is referred to as source cloud, while the second point cloud will be denoted as target point cloud.
The proposed algorithm mainly consists of two parts: A local surface registration algorithm that is able to deal with the before-mentioned challenges, such as partial overlap, occlusions and an incorporated scaling property. This local algorithm is executed with a global search strategy from different initial positions and after a time-dependent stop-criterion has been satisfied, the best solution is chosen.
The final algorithm is supposed to be able to align two surfaces with only an initial estimate of the position of the source cloud relative to the target cloud and to operate completely independent of the rotation of the source cloud.
First, the input-output-interface of the algorithm will be defined. Afterwards, the local alignment algorithm and the global search strategy will be explained.

5.1 Interface

The source point cloud is defined as $X = \{\mathbf{x}_i\}_{i=1}^{n_x}$ and the target point cloud as $Y = \{\mathbf{y}_i\}_{i=1}^{n_y}$ with $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^3$. n_x denotes the number of points in X and n_y the number of points in Y accordingly.
The point clouds are represented in two different coordinate systems, namely the source and the target coordinate system.
In general, the two point clouds contain a different amount of points. In most cases, the source cloud will represent an object which can be partially found in the target cloud. The target cloud usually also contains a large portion of outliers because the surroundings of the object are scanned, too. Nevertheless, the algorithm will of course also work for other cases, such as point clouds with full overlap.
An initial position of the source object relative to the target object is also required. This position is represented by an initial translation vector $\mathbf{t}_{init} \in \mathbb{R}^3$. If the objects are of different size, an initial scaling parameter $s_{init} \in \mathbb{R}$ also has to be provided as input.
After successfully completing the registration problem, the algorithm returns the estimate of the source object's pose relative to the target object. The pose consists of the rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ of the source object. In the case the objects are of different size, the scaling parameter $s \in \mathbb{R}$ will also be estimated.
The output of the algorithm can be seen as the transformation matrix between the source

and the target point cloud. Therefore, applying the homogeneous transformation matrix to the source cloud, will project it onto the target cloud.

5.2 Data Preprocessing

Before executing the alignment algorithm on the data, some preprocessing steps are necessary to improve the performance and the success probability.
Computational power limits the number of points in the point clouds that can be processed efficiently. Different down-sampling methods exist, such as a *Voxel-grid* filter or algorithms that try to preserve important information in the data and down-sample the clouds non-uniformly. However, if the clouds are adequately large, a random-filter is generally sufficient and can even have advantages because it is possible to use different sub-sampled point clouds which can be helpful in order to avoid small local minima.
Another possibility to improve the success probability of the algorithm is to remove those points from the target cloud of which it can be assumed that they do not correspond to the source cloud.
The probably easiest but also very efficient form of data removal is to delete those points that can be assigned to a plane in the target cloud. If it is known that the source object does not contain large planar structures, it can be assumed that planes in the target cloud do not belong to the desired source object. This is advisable because many common objects do not contain planar structures while frequently appearing environments like tables or walls do contain easily detectable planes.
The plane removal procedure deletes the largest plane in the target cloud. Therefore, it is possible to also use the plane removal for partially-planar objects. If a part of a planar environment, that is larger than the planar structures in the object, is also contained in the target cloud, the plane will be removed without affecting the desired object.
If activated, the plane-removal is executed by using a RANSAC-based method.

5.3 Local Point Cloud Alignment

In this section, different parts of the literature from section 2 will be used in order to design an algorithm that can be initialized arbitrarily and will then converge to a local minimum.
The local algorithm is initialized with a point from the parameter space. The initial parameters are a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$, a translation vector $\mathbf{t} \in \mathbb{R}^3$ and a scaling factor $s \in \mathbb{R}$.
The transformation that is applied to the source point cloud is referred to as T.

5.3.1 Iterative Closest Point Algorithm

The best-known approach for the 3D-registration problem is the ICP algorithm that uses an iterative procedure to estimate the transformation matrix which projects the source point cloud onto the target cloud.
If the correct correspondences between the points of the two point clouds are known, the problem is relatively easy to solve. Unfortunately, the correct correspondences are a-priori unknown and depend on the transformation matrix. The correct transformation, however, depends on the correspondences.
The ICP algorithm splits this problem into two sub-problems by using a common approach to solve this kind of *chicken-and-egg* problem, the so-called Expectation-Maximization (EM)-algorithm. By using the EM-approach and the nearest neighbours in the target cloud as temporary correspondences for the points in the source cloud, the two-way dependency can be solved by iterating the following three steps:

Apply Transformation The current transformation matrix $T(X)$ is applied to the source point cloud.

E-Step: Find correspondences The nearest neighbour in the target point cloud is assigned as correspondence for each point in the projected source point cloud.

M-Step: Find transformation The transformation matrix is calculated so that the sum-of-least-squares between the source cloud and the assigned correspondences is minimized.

By iterating these steps, the algorithm is guaranteed to converge to a local minimum as shown by Besl and McKay [3].
A schematic visualization of the ICP algorithm can be found in figure 5.1.
The simple sum-of-least-squares cost function $\phi(X,Y)$ as it is used by the ICP algorithm will be defined as

$$\phi(X,Y) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \|T(\mathbf{x}_i) - \mathbf{y}_j\|^2 \cdot \omega_{i,j} \tag{5.1}$$

where $T(\mathbf{x})$ describes the affine transformation that is searched for and $\omega_{i,j}$ is defined as

$$\omega_{i,j} = \begin{cases} 1 & \text{if } j = \arg\min_{j' \in 1:n_y} \|T(\mathbf{x}_i) - \mathbf{y}_{j'}\| \\ 0 & \text{else} \end{cases} \tag{5.2}$$

For now, the transformation matrix is designed to only rotate and translate the points, so it is set as $T(\mathbf{x}_i) = \mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}$.
The rotation matrix has nine components but only three degrees of freedom. This means that additional constraints have to be made in order to receive a valid rotation matrix.

This can be achieved by fulfilling the equations $\det(\mathbf{R}) = 1$ and $\mathbf{R} \cdot \mathbf{R}^T = \mathbf{I}$ with $\mathbf{I}$ as the 3×3-identity matrix.
For simplicity, the nearest neighbours of the source points $\mathbf{x}_i$ will be denoted as $C = \{\mathbf{c}_i\}_{i=1}^{n_x}$, where $\mathbf{c}_i$ represents the target point in $\mathbf{y} \in Y$ that minimizes the euclidean distance to the source point $\mathbf{x}_i$.
The cost function can now be written as

$$\mathbf{R}^\star, \mathbf{t}^\star = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{n_x} \|(\mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}) - \mathbf{c}_i\|^2 \tag{5.3}$$

$$\text{s.t. } \mathbf{R}^T\mathbf{R} = \mathbf{I} \wedge det(\mathbf{R}) = 1 \tag{5.4}$$

with $\mathbf{R}^\star$ and $\mathbf{t}^\star$ as the (locally) optimal transformation parameters.

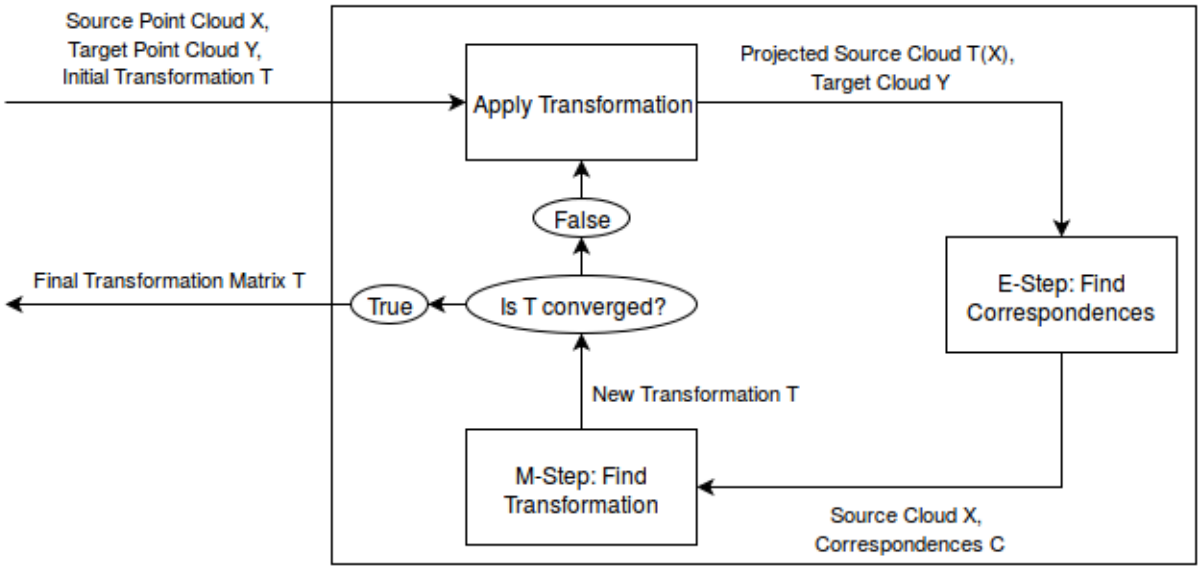


Figure 5.1.: Schematic Visualization of the ICP Algorithm
The ICP algorithm takes the source cloud X, the target cloud Y as well as an initial transformation matrix $T(X)$ as input. The algorithm then iteratively executes the following procedure until $T(X)$ is converged: First, the current transformation matrix is applied to the source cloud. Afterwards, for each point in the resulting projected source cloud the nearest neighbour in the target cloud is assigned as correspondence. Finally, the transformation matrix is recalculated so that the sum-of-least-squares between the source cloud and the correspondences is minimized.

5.3.2 E- and M-Step of the ICP Algorithm

The E-Step seems straightforward as for each point $\mathbf{x}_i$ a nearest-neighbour (NN)-lookup in the target cloud has to be performed and assigned as correspondence $\mathbf{c}_i$. In practice, however, this is often the bottleneck of the algorithm. This makes optimization for the NN-lookup, that will be discussed in section 5.5.2, necessary.
The M-Step operates on a set of hard-set correspondences and optimizes for the transformation so that the sum-of-squared-distances between the source-correspondence pairs are minimized. For this simple cost function, closed-form solutions exist which avoid using computational expensive numerical optimization methods.
Arun and Huang [31] proposed a method using SVD that can solve the described optimization problem in a fast and robust manner.
This can be done by calculating the matrix

$$\mathbf{H} = \sum_{i=1}^{x_n} \mathbf{x'}_i \mathbf{c'}_i^T \tag{5.5}$$

with $\mathbf{x}'_i$ and $\mathbf{c}'_i$ as the normalized source points and their normalized correspondences. These can be calculated easily by subtracting the mean $\bar{\mathbf{x}}$ of the source point cloud and the mean $\bar{\mathbf{c}}$ of the correspondences from each point

$$\mathbf{x}'_i = \mathbf{x}_i - \bar{\mathbf{x}}, \mathbf{c}'_i = \mathbf{c}_i - \bar{\mathbf{c}} \tag{5.6}$$

The Singular Value Decomposition of H can now be calculated as

$$\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T \tag{5.7}$$

The optimal rotation matrix $\mathbf{R}^\star$ and the optimal transformation vector $\mathbf{t}^\star$ can be computed as

$$\mathbf{R}^\star = \mathbf{V}\mathbf{U}^T, \mathbf{t}^\star = \bar{\mathbf{c}} - \mathbf{R}\bar{\mathbf{x}} \tag{5.8}$$

In some cases, this results in a reflection matrix $\mathbf{R}$ with a determinant equal to -1. This can be easily corrected in most cases by multiplying the last column of R with -1. In cases this fails, conventional least-squares is not appropriate and a RANSAC-like technique is proposed. However, this should only happen if the problem is ill-fitted and for the purpose of the proposed algorithm, it is sufficient to stop the execution of the ICP algorithm.

5.3.3 Scaling Property

In the previous sections, the computational background for the estimation of the rotation and the translation have been discussed but if the source and target point cloud are of different sizes, the scaling parameter also has to be taken into account for the algorithm. Another advantage of the SVD approach in the M-Step is that it allows to integrate the scaling parameter simply in the closed-form solution as shown by Zinssner [13] in 2005. The scaling factor $s \in \mathbb{R}$ can be integrated in the transformation by multiplying the rotation matrix with the parameter s

$$T(\mathbf{x}) = s \cdot \mathbf{R} \cdot \mathbf{x}_i + \mathbf{t} \tag{5.9}$$

An additional constraint has to be set because s must be greater than zero. Otherwise $s = 0$ would be a trivial but perfect solution and would simply translate the whole source cloud (that is shrunk to a single point) to one of the target cloud points.
The cost function now looks like the following:

$$\mathbf{R}^\star, \mathbf{t}^\star, s^\star = \arg\min_{\mathbf{R},\mathbf{t},s} \sum_{i=1}^{n_x} \|(s \cdot \mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}) - \mathbf{c}_i\|^2 \tag{5.10}$$

$$\text{s.t. } \mathbf{R}^T\mathbf{R} = \mathbf{I} \wedge det(\mathbf{R}) = 1 \wedge s > 0 \tag{5.11}$$

It can be shown that the rotation matrix $\mathbf{R}$ is not affected by the new scaling parameter s and can thus be computed the same way as before.
If the normalized source and target points are used as in the previous section, it is possible to solve for the scaling factor independently of the translation. Thus, the scaling factor is the only unknown in the following optimization problem as the rotation $\mathbf{R}^\star$ has already been estimated

$$\begin{aligned} s^\star = \ & \operatorname*{argmin}_{s} \quad \sum_{i=1}^{n_x} \|(s \cdot \mathbf{R}^\star \cdot \mathbf{x}_i') - \mathbf{c}_i\|^2 \\ & \text{s.t.} \qquad s > 0 \end{aligned} \tag{5.12}$$

By using the vectors $\mathbf{c}_i' = \mathbf{c}_i - \bar{\mathbf{c}}$ and $\mathbf{x}_i' = \mathbf{x}_i - \bar{\mathbf{x}}$ from the previous subsection, $s^\star$ can be calculated as

$$s^\star = \frac{\sum_{i=1}^{n_x} \mathbf{c}_i'^T \mathbf{x}_i'}{\sum_{i=1}^{n_x} \mathbf{x}_i'^T \mathbf{x}_i'} \tag{5.13}$$

In the last step, the transformation is computed as

$$\mathbf{t}^\star = \bar{\mathbf{c}} - s^\star \cdot \mathbf{R}^\star \cdot \bar{\mathbf{x}} \tag{5.14}$$

If these equations are put in the M-Step of the standard ICP algorithm, the estimation of the scaling property has been successfully integrated.
The algorithm does not increase the runtime very much in comparison to the standard ICP and works well and stable with relatively small size differences, which is sufficient due to the commitments that have been made. However, the algorithm still depends on a good initial estimate as it is only guaranteed to converge to a local minimum.

5.3.4 Partial-Overlap and Occlusion Robustness

The algorithm in its current form is able to find a local minimum for two point clouds with full overlap. It will now be extended so that it can also handle target clouds that only contain parts of the source object's surface.
As discussed in section 2, there are basically two famous variants to reach this goal. In both variants, all correspondences are assigned the same way as in the regular ICP algorithm and afterwards, some of the source-correspondence pairs are rejected. One possibility is to reject a fixed portion of the assigned pairs. The disadvantage of this approach is the fact that the inlier-outlier-rate has to be set to a fixed value before the execution of the algorithm. This restricts the possible applications of the final algorithm and limits the possible range of overlap-portions.
The second variant rejects all pairs whose distance is larger than a certain threshold. This is a much better solution because a good threshold can be experimentally evaluated (see section 5.6) and this approach will be able to deal with arbitrary inlier-outlier-proportions.
Other more complex solutions that assign weights (for example points with similar normals are assigned higher weights) or use multiple correspondences are not suitable as they require numerical optimization methods because no closed-form solutions exist for these approaches. This would conflict with the performance requirement that has been discussed at the beginning of the section.
The algorithm can be extended by inserting another step between the E-Step and M-Step in the iterative routine and by defining a distance threshold ζ that limits the distance of points that are taken into account. The modified routine now iterates the following steps:

Apply Transformation

E-Step: Find correspondences

Trim Point Cloud Remove all pairs with higher distance than the distance threshold ζ.

M-Step: Find transformation

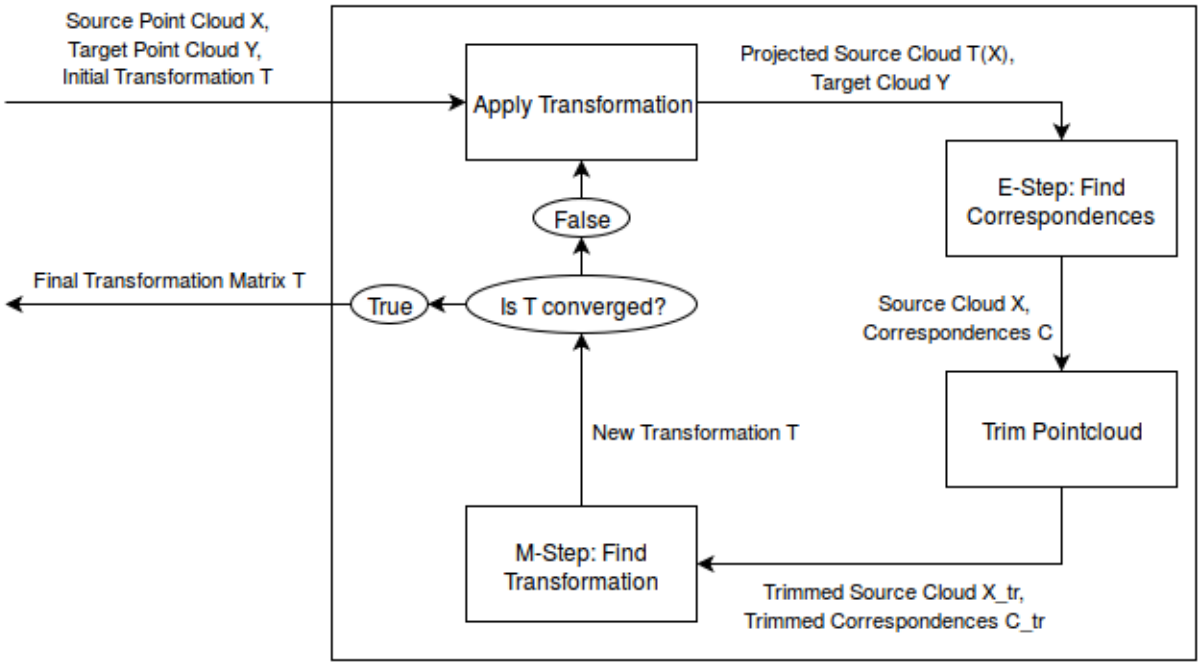


Figure 5.2.: Schematic Visualization of the Modified Algorithm
In contrast to the procedure in figure 5.1, another step between the E- and M- Step is added. In the Trim Point Cloud step, all source-correspondence pairs whose distance is greater than the distance threshold ζ are removed .

A schematic sketch of the modified algorithm can be found in figure 5.2.
The modified cost function can be written as

$$\mathbf{R}^\star, \mathbf{t}^\star = \arg\min_{\mathbf{R},\mathbf{t},s} \sum_{i=1}^{n_x} \|(s \cdot \mathbf{R} \cdot \mathbf{x}_i + \mathbf{t}) - \mathbf{c}_i\|^2 \cdot \omega_i \tag{5.15}$$

$$\text{s.t. } \mathbf{R}^T\mathbf{R} = \mathbf{I} \wedge det(\mathbf{R}) = 1 \wedge s > 0 \tag{5.16}$$

with

$$\omega_i = \begin{cases} 1 & \text{if } \|\mathbf{x}_i - \mathbf{c}_i\| < \zeta \\ 0 & \text{else} \end{cases} \tag{5.17}$$

A problem arising by only using pairs with a distance lower than a certain threshold is that if the source cloud is initially placed outside of the target cloud and all pairs have a higher distance than the threshold, no valid pairs will be found and the algorithm fails. This is especially problematic for target clouds that are split in the middle of the desired object (which may for example happen if the object is scanned from an unfavourable perspective). In these cases, the source cloud would be ideally placed in the middle of the target cloud. If the initial orientation of the source object is unfortunate, all source-correspondence pairs may have a higher distance from each other than ζ. This can be avoided by using a minimum overlapping percentage. If the portion of valid pairs is

lower than this minimum overlapping percentage, the number of pairs that are taken into account is increased so that the fixed minimum overlapping criterion is satisfied. Pairs with a lower distance are of course always preferentially used by the algorithm for the M-Step, instead of pairs with higher distances.

5.4 Global Rotation Search Strategy

This section will present a search strategy that aims for estimating the transformation of the 3D surface registration problem independently of the initial rotation and locally convergent in terms of translation and scaling factor. This will be done by constructing a search strategy that uses the locally convergent algorithm from the previous section to evaluate samples from a rotation search space. When the algorithm is terminated, the best evaluated sample is returned.
First, the rotation search space will be defined and discretized. Afterwards, the search strategy will be constructed so that a high success probability of finding the desired solution will be provided.
As discussed in section 2, different approaches have been made to achieve global convergence for the point cloud registration problem but real-time requirements limit complex strategies. This is why an iterative refinement procedure that aims to find the global optimum by using samples with a great variety is used. This strategy will be explained and presented in the following subsections.
Since no knowledge about the overlapping percentage of the source and target point cloud is available, it is hard to decide when to stop the search strategy and whether the desired solution has been found.
It seems reasonable to spend more time searching for a better sample in difficult cases than in easier ones. Therefore, a stop-criterion for the search strategy that depends on the quality of the current best solution, as well as on the passed execution time of the search strategy, will be used.
Thereby, the average runtime will be decreased by spending less time searching for a good estimate in easier cases than in more difficult ones.

5.4.1 Rotations in Angle-Axis-Representation

The search strategy will use the Angle-Axis representation to describe the rotation parameters which will be shortly explained here.
Rotations in Angle-Axis representation are defined by a vector $\mathbf{r} \in \mathbb{R}^3$. The rotation axis is simply given by $\mathbf{r}/\|\mathbf{r}\|$ and the rotation angle by $\|\mathbf{r}\|$.

By using the skew-symmetric matrix

$$[\mathbf{r}]_\times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}, \text{ with } \mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \tag{5.18}$$

the rotation matrix $\mathbf{R_r}$ that is defined by $\mathbf{r}$ can be calculated by using the matrix exponential of $[\mathbf{r}]_\times$

$$\mathbf{R_r} = \exp([\mathbf{r}]_\times) = \mathbf{I} + \frac{[\mathbf{r}]_\times \cdot \sin(\|\mathbf{r}\|)}{\|\mathbf{r}\|} + \frac{[\mathbf{r}]_\times^2 \cdot (1 - \cos(\|\mathbf{r}\|))}{\|\mathbf{r}^2\|} \tag{5.19}$$

The vector $\mathbf{r}$ can accordingly be computed by the matrix logarithm $[\mathbf{r}]_\times = \log(\mathbf{R_r})$.
The main advantage of the Angle-Axis representation for the purpose of this thesis is that it allows to divide the rotation space easily. All possible rotations can be represented by the values of $\mathbf{r}$ that lie inside a sphere with centre $\mathbf{0} \in \mathbb{R}^3$ and radius π.
Compare Yang et al [26].

5.4.2 Rotation Space Parametrization

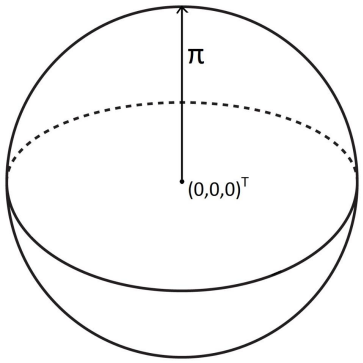


Figure 5.3.: Rotation Space
With the Angle-Axis representation, all possible 3D-rotations lie inside a sphere with radius π.

First, the search space will be defined. Since the translation and scaling are supposed to be locally convergent, only the rotation space has to be parametrized.
When following the same approach as Yang et al [26], all possible three dimensional rotations $\mathbf{r} \in \mathbb{R}^3$ can be represented as the points that lie inside a three dimensional sphere with centre $\mathbf{0} \in \mathbb{R}^3$ and radius π and thus satisfy the inequality $\|\mathbf{r}\| \leq \pi$. This is depicted in figure 5.3. Note that all points outside the sphere are also valid rotation parameters but can be seen as redundant representations of points inside the sphere.

5.4.3 Iterative Refinement Strategy

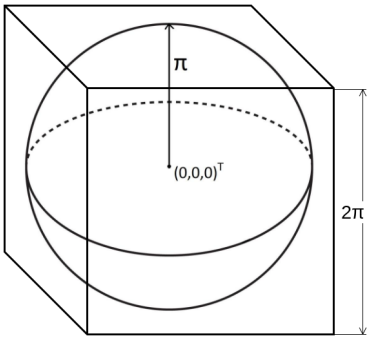


Figure 5.4.: Extended Rotation Space
The extended rotation space consists of a cube with edge length 2π that completely surrounds the sphere from figure 5.3.

Now that the search space has been parametrized, the search strategy can be defined. The strategy should aim for a fast location of the global optimum with a high certainty in the correctness of the result. The fact that the global optimum cannot be identified definitely complicates the decision when to stop the search strategy.
Since no prior knowledge of the optimal rotation is available, taking samples from the rotation space with a high variance seems reasonable. Therefore, the distances between the samples and to the boundaries should be maximal.
In order to make discretization of the parameter space easier, the rotation space will be extended to a cube with edge length 2π that imbeds the sphere that has been defined as the parameter space. This extended parameter space is depicted in figure 5.4.
By using this trick, the parameter space can be divided by an octree that defines the sample points.
The octree simply splits the cube into eight equally large sub-cubes and can thus divide the rotation space in arbitrary small pieces. Each cube stores its centre coordinates that can be used to initialize the local point cloud alignment algorithm. The root node contains the origin $(0,0,0)^T$ and thus represents an identity matrix as rotation matrix. The eight children of the root node can be obtained by permuting the signs of $(\pm\pi/2, \pm\pi/2, \pm\pi/2)^T$.
Fixed parameters define the maximum recursion-depth of the octree and also the minimum distance between two samples in rotation space.
In general, not the complete octree will be evaluated because the algorithm is terminated if a time-dependent stop criterion is fulfilled that will be defined in a later subsection.
The samples defined by the centres of the sub-cubes of the octree are stored in a priority queue that is straight-lined evaluated until the stop-criterion is fulfilled. When traversing the priority queue, the best sample evaluated yet will be stored and returned if the stop-criterion is fulfilled.
The best sample will be defined with a different cost function than the sum-of-least-squares of the locally convergent algorithm and will be presented in the following sub-

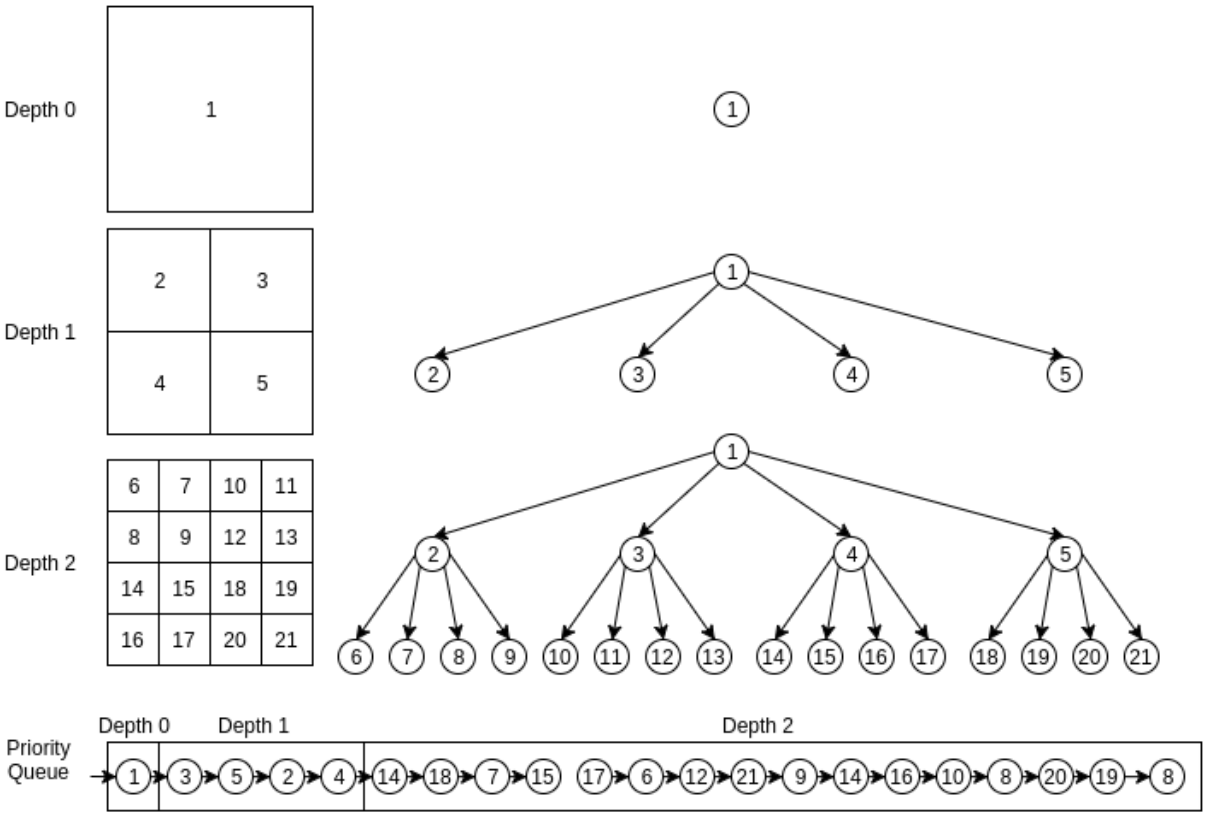


Figure 5.5.: Creation Process of the Priority Queue

2D example of the search strategy: The three boxes show how the space is divided by the quadtree (octree for the 3D case) where each node stores the centre of the corresponding cube. On the right side, the corresponding quadtree of each recursion-step can be seen. On the bottom, an example priority queue for the given depth is depicted. The nodes of the octree are stored in increasing level order but the order of the nodes of each level is estimated randomly.

section.

When traversing the queue, it is beneficial to try to explore the whole rotation space uniformly and refine the search steadily. This is achieved by a breadth-first search which means that each recursion level will be explored completely before the next one is taken into account.

The number of nodes in each level grows exponentially with basis eight, meaning that the third level already contains $8^3 = 512$ samples. If these samples were put in order into the priority queue, only a small portion of the search space would be evaluated thoroughly, while other areas would be neglected. This problem can be avoided simply by traversing each level of the octree in a random order. The whole process is depicted in figure 5.5 for the 2D-case for the simplicity of the visualization.

By extending the parameter space, some cubes of the octree represent redundant information. In order to reduce unnecessary computations, all cubes whose centre lies outside the sphere with radius π can be discarded. This is visualized in figure 5.6.

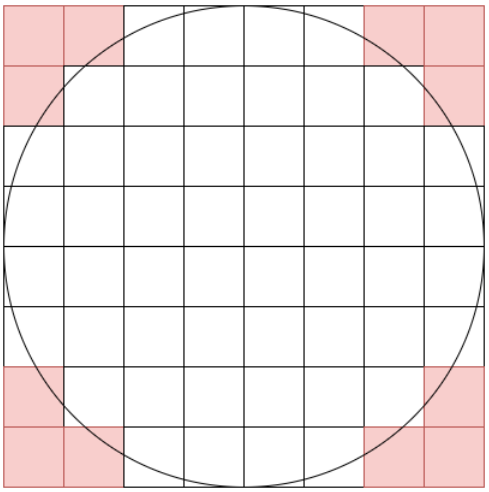


Figure 5.6.: Discarding of Redundant Cubes
Cubes in the quadtree (octree in the 3D-case) whose center lies outside the sphere with radius π represent rotation parameters which can also be found at a point inside the sphere and can thus be discarded.

5.4.4 Evaluation of the Sample Quality

As seen in the previous section, the rotation space is searched with a refinement strategy that aims to search the whole area relatively equally to avoid missing the global optimum by focussing on the wrong section of the rotation space.
In practice, the main difficulty is not to find the desired solution with one of the samples, but to identify and distinguish it from other suboptimal solutions. The global optimum of the cost function that is used by the local algorithm is often not identical to the desired solution.
Therefore, the best evaluated sample from the search strategy has to be identified based on different criteria.
The desired solution is expected to have a low distance between its correspondences and a high inlier percentage. This suggests to define a quality criterion based on these assumptions.
Each sample from the rotation space is optimized in terms of the parameters and all criteria regarded here are evaluated by using the optimized parameters $\mathbf{R}, \mathbf{t}$ and s, which get returned by the locally convergent algorithm.
The local algorithm maximizes a sum-of-squared-distances cost function that only takes correspondences whose distance is lower than the distance threshold ζ into account. This is problematic when comparing different samples because estimates with fewer inliers will have an advantage compared to samples with a higher amount of inliers.
This can be circumvented by dividing the sum-of-squared-distances by the number of points that are taken into account for the calculation and therefore using a per-point-error $\epsilon(\mathbf{R}, \mathbf{t}, s)$.

In order to reward high overlaps, the overlapping percentage $\Psi(\mathbf{R},\mathbf{t},s)$ can be calculated by computing the number of points whose correspondence is lower than an evaluation threshold ρ

$$\Psi(\mathbf{R},\mathbf{t},s) = \frac{\sum_1^{n_x} \omega_i}{n_x},\ \omega_i = \begin{cases} 1 & \text{if } \|(s\cdot\mathbf{R}\cdot\mathbf{x}_i+\mathbf{t})-\mathbf{c}_i\| < \rho \\ 0 & \text{else} \end{cases} \tag{5.20}$$

It has to be noted that the distance threshold ζ used to determine which points are taken into account for the M-Step of the ICP algorithm is not the same as the evaluation threshold ρ in general.
The per-point-error is defined as

$$\epsilon(\mathbf{R},\mathbf{t},s) = \frac{\sum_{i=1}^{n_x} \|(s\cdot\mathbf{R}\cdot\mathbf{x}_i+\mathbf{t})-\mathbf{c}_i\|^2\cdot\omega_i}{\Psi(\mathbf{R},\mathbf{t},s)\cdot n_x} \tag{5.21}$$

where $\mathbf{c}_i$ denotes the nearest neighbour of $\mathbf{x}_i$ in the target cloud.
Finally, an optional penalty for the deviation of the estimated translation from the initial position of the source cloud will be included.
This penalty can be useful if the template has a long shape and continues to "fly away" from the initial position or if an imperfect template is used and the global optimum can barely be designed to coincide with the desired solution. If the position penalty is taken into account, the operator can improve the success probability by cleverly placing the template at a position that is closer to the correspondence of the desired object than to the suboptimal solution.
The quality criterion $Q(\mathbf{R},\mathbf{t},s)$ for the samples will now be defined as

$$Q(\mathbf{R},\mathbf{t},s) = \log(1-\Psi(\mathbf{R},\mathbf{t},s)) + \lambda\cdot\log\epsilon(\mathbf{R},\mathbf{t},s) + \mu\cdot\log\|\mathbf{t}_{init}-\mathbf{t}\| \tag{5.22}$$

with the tuning parameters $\lambda,\mu\in\mathbf{R}$.
After the search strategy is stopped by a time-dependent stop-criterion that will be defined in the following subsection, the sample with the minimum value of Q is returned.

5.4.5 Stop Criterion

The algorithm could easily be stopped after a fixed number of iterations or after a sufficiently good estimate has been found. It turns out, however, that in practice good estimates for easier problems with a high overlapping percentage are found much quicker than in difficult cases, for example if the sensors are placed at an unfavourable perspective.
Therefore, it seems reasonable to let the stop-criterion depend on time. The more time has passed, the smaller the overlapping percentage of the best sample has to be. Otherwise, in order for the algorithm to terminate quickly, an estimate with a high overlapping

percentage has to be found.
This leads to a time-dependent stop-criterion $\Pi(t)$ that decreases the minimum requirements of the solution over time. A delayed and damped negative exponential function

$$\Pi(t) = \exp(-\alpha(t-\beta)) \cdot 100 \tag{5.23}$$

will be used, with the damping coefficient α and delay factor β. If the returned value of $\Pi(t)$ is smaller than the overlapping percentage $\Psi(\mathbf{R}, \mathbf{t}, s)$, the stop-criterion has been satisfied and the algorithm is terminated.
The overlapping percentage is computed according to equation 5.20 from the previous section.
An example plot of the time-dependent stop-criterion can be found in figure 5.7.

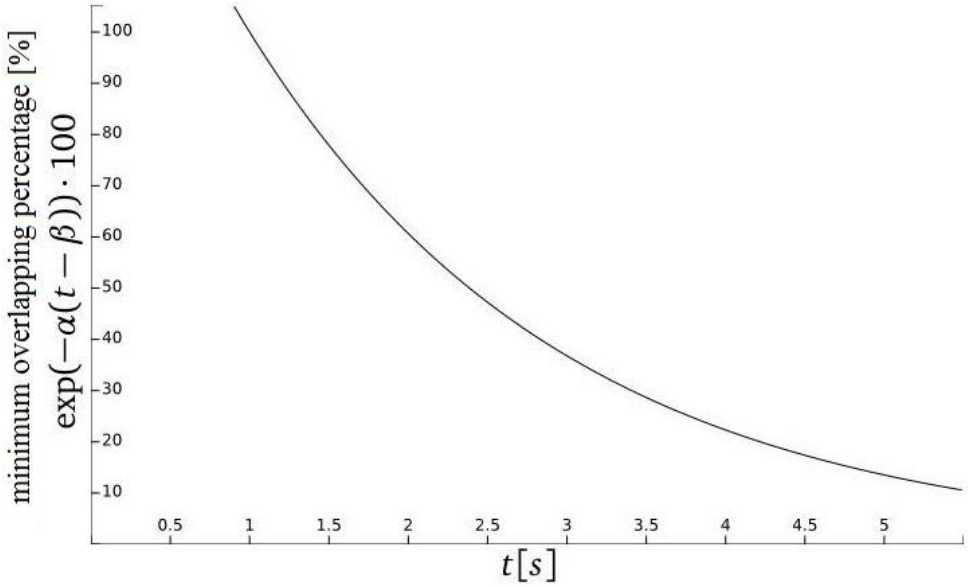


Figure 5.7.: Function Plot of the Time-Dependent Stop-Criterion
The vertical axis depicts the minimal overlapping percentage of the source with the target point cloud. Their relation is described by $y = \exp(-\alpha(x-\beta)) \cdot 100$ with damping coefficient α and delay factor β. The graph shows an example plot with $\alpha = 0.5$ and $\beta = 1$.

5.4.6 Symmetry Search and Refinement

When executing the described algorithm, it terminates with an estimate of the optimal rotation parameters. Sometimes, it still terminates in a local minimum, however, because of the heuristic background.
In practice, it turns out that some of these local minima appear more often than others and sometimes they can be exploited in order to reach the global optimum.

Many objects have a partly-symmetric shape which creates a large local optimum that may cause the algorithm to finish before the globally optimal parameters are found.
If the symmetries of the source cloud are known (due to axis-aligned symmetries of the source object), the locally convergent algorithm can be initialized with symmetrically rotated versions of the best found estimate so far.
In case a new result is assigned a higher quality value, it is more likely to be the desired solution and will replace the old best estimate.
Now that the final estimate has been chosen, it is reasonable to try to increase the accuracy of the result. The samples evaluated so far have been evaluated on small sub-clouds of the source and target point clouds in order to improve the performance.
As soon as the stop-criterion is fulfilled and the algorithm terminated, the locally convergent point cloud alignment algorithm can be initialized one last time with the best estimate of the transformation parameters.
Thereby, the accuracy of the result can be increased by using more points than before. Since the local point cloud algorithm is executed only once with a bigger number of points in the clouds, the accuracy can be increased without severely stretching the runtime of the global algorithm.

5.5 Implementation Details

An implementation of the presented algorithm in *C++* is enclosed to this thesis. This section will present and discuss implementation details, such as utilized environments, optimization strategies and data structures that are used to improve the performance of the algorithm.

5.5.1 ROS Action

The algorithm will be based on the *Robot Operating System* (ROS) library and implemented as a ROS *Action Server*, due to its application in robotics.
An action server in ROS takes a goal as input, can return feedback about the progress of the action and after completion returns the result.
This server takes two point clouds (source and target) and an initial position as arguments and returns the pose estimate of the corresponding registration problem.
Every time a new best estimate is found, the overlapping percentage and the per-point-error will be published as feedback.
A ROS action client will be used to send requests to the server and process the results. This client also represents the connection to the used interface OCS.

5.5.2 Optimization

The performance of the local ICP algorithm is extremely important because a faster execution results in more ICP evaluations and therefore, a higher overall success probability of the point cloud alignment.
The bottleneck of the ICP algorithm is the E-Step where the correspondences are assigned. A naive NN-search results in a runtime of $O(n_x \cdot n_y)$ but can be easily improved by using a KD-tree representation of the target point cloud. Thereby, the runtime can be reduced to $O(n_x \cdot \log n_y)$. The attached implementation uses a kd-tree implementation of the open source *Point Cloud Library* (PCL).
Even with a linear-logarithmic runtime, the performance of the ICP algorithm can be further improved by discarding a subset of points before the execution. Point clouds of the environment can get extremely large and there is little use of searching the complete point cloud for correspondences if the algorithm is supposed to be only locally convergent in case of the translation. Therefore, the maximum distance of the points in the source cloud from the origin of the coordinate system is calculated. Afterwards, each point in the target cloud that is further away from the initial position than this maximum radius times a fixed factor is discarded and the target cloud thereby downsized.
The computations of the algorithm are performed with the matrix data structure from the open source library *Eigen* that especially optimizes cache and memory efficiency.
As described, the search strategy evaluates as many ICP iterations as possible until the stop-criterion is fulfilled. Afterwards, the best estimate is returned. This can be improved by using multiple threads that process the priority queue in a parallel manner. If a result of one of the threads satisfies the stop-criterion, the other threads finish their current ICP evaluation normally and afterwards, the best estimate is chosen.
The symmetry checks of the best estimate are also executed in parallel. The parallelization process is implemented by using the open source library *OpenMP*.
The ICP algorithm operates only with sub-samples of the clouds that contain a fraction of the points (the number of used points is set by fixed parameters) that are contained in the original point clouds. This is done to achieve a higher performance and the lack of benefits that arise from a higher amount of points, as these mostly contain redundant information.
This sub-sampling can be used to avoid getting stuck in small minima which happens very often in practice. When the ICP algorithm is converged, this often happens only because of a very small local minima. By using a different sub-sample of the source cloud, the algorithm is often freed from this local minima and continues iterating. Therefore, using different sub-samples can eliminate small local minima.
In practice, this is done by creating different sub-samples of the source cloud at the beginning. If the ICP algorithm gets stuck, another sub-sample is used. If all samples converged without any change, the algorithm is terminated.

5.6 Important Parameter Values

Alongside the arguments for the action server, the user also provides the necessary parameters. These parameters are set by default but can be changed for specific requirements. The most important of these parameters will be discussed here and reasonable settings will be suggested and justified.
The first parameter that will be reviewed is the distance threshold of the ICP algorithm that defines which points are taken into account for the minimization step to estimate the transformation matrix. All source-correspondence pairs whose distance is higher than the evaluation threshold are discarded and only the points with a distance lower than the threshold are regarded.
It is advisable to assign the distance threshold according to the sensors that are used and the accuracy of the object template. A relatively easy way to estimate the distance threshold is by aligning a template by hand and assigning each point of the source cloud its nearest neighbour in the target cloud as correspondence. Afterwards, the distances between the source points and their correspondences are saved and a probability density is estimated, for example by using an arbitrary density estimation algorithm. Thereby, the frequency of the distances of the correct solution can be analyzed.
Figure 5.8 depicts an example of such a density distribution of a simulated drill with a high overlap. The target cloud is created by simulated *Light Detection and Ranging* (LiDaR)-scanners. In the image, the horizontal axis shows the distance of the source-correspondence pairs and the vertical axis their frequency of occurrence.
A part of a Gaussian distribution can be seen clearly on the left side of the plot. This

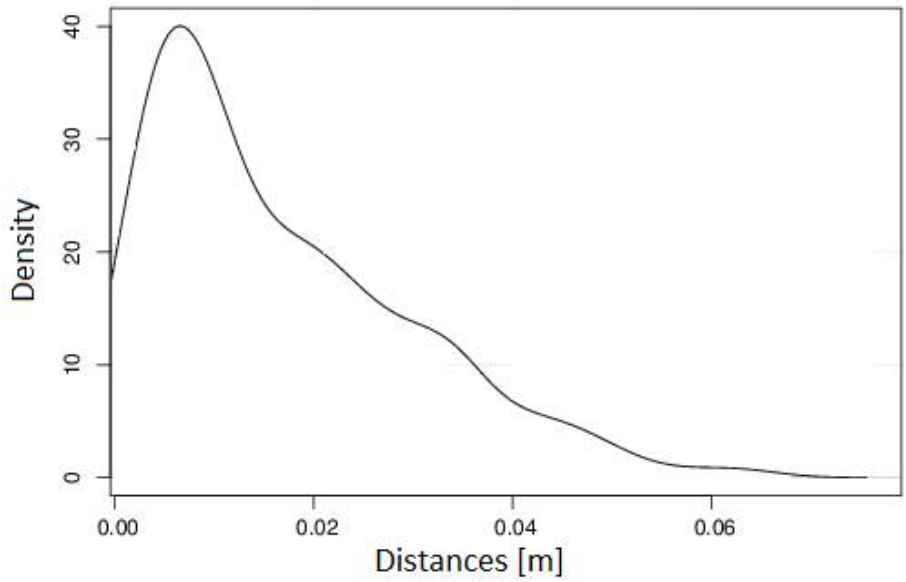


Figure 5.8.: Density Distribution of the Distances Between Source-Correspondence Pairs of a Simulated Drill

seems reasonable as measurement errors from the scanners are usually Gaussian distributed and arise as additive noise to the inlier points. Note that the mean of the Gaussian distribution will not lie at point zero of the horizontal axis, as there are no negative distances.
The outliers appear in a decreasing frequency with increasing distance.
When setting the distance threshold, a trade-off between missed inliers because of a too small threshold and outliers that are falsely taken into account due to a too high threshold has to be made. The algorithm is also not initialized optimally which causes higher distances to neighbouring points. This suggests to set the distance threshold a little bit higher than it would be done for a perfectly-aligned template. In the example depicted in figure 5.8, a distance threshold of about $\zeta = 0.02m$ would be a good value to choose.
The same figure is also helpful for setting the evaluation threshold ρ. This threshold defines which points are taken into account for the computation of the overlap of the two point clouds. In contrary to the distance threshold, it should be strictly avoided to take outliers into account. This suggests to set the evaluation threshold to the point on the x-axis where the Gaussian distribution stops being dominant and the influence of the outliers increases. In the depicted example, a good point to choose would be $\rho = 0.015m$.
Next, the two parameters α and β that create the shape of the time-dependent stop-criterion will be discussed. In figure 5.9, plots of the stop-criterion for different combinations of the two parameters can be seen.
These parameters can be set in a broad range. They depend strongly on user-preferences and on the difficulty of the alignment problem. Longer execution times result in a higher

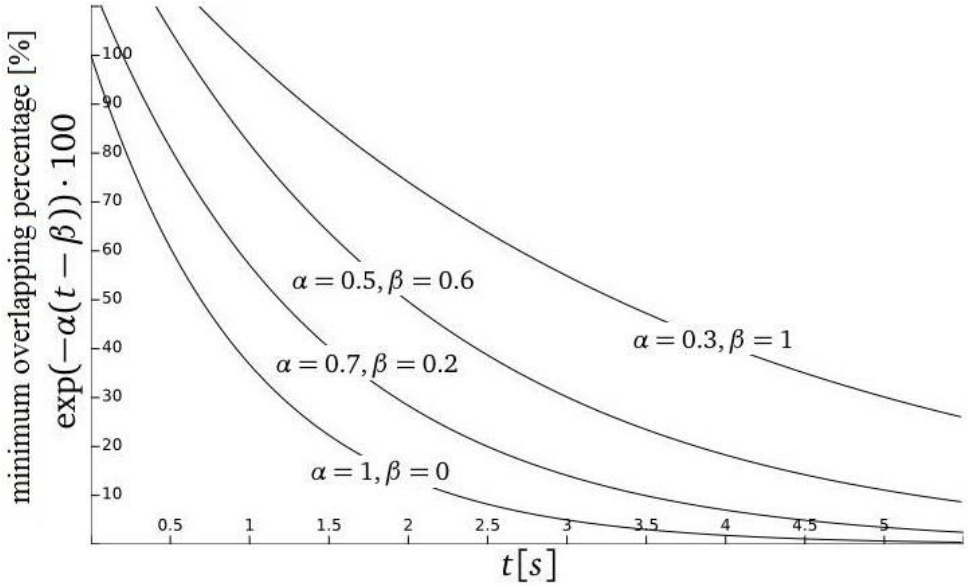


Figure 5.9.: Function Plot of the Time-Dependent Stop-Criterion with Different Parameters

success probability but it can also be helpful to use short execution times. If the algorithm fails, it can simply be executed again. If the time saved due to a shorter computation time is greater than the time lost due to multiple necessary executions of the algorithm, this can be a good strategy, too.
Various other parameters, such as the number of points in the sub-cloud or the maximum refinement depth of the search strategy, can also be changed by the user. Explanations of these parameters can be found in an documentation associated to the configuration file that contains the parameter value definitions.

5.7 Summary

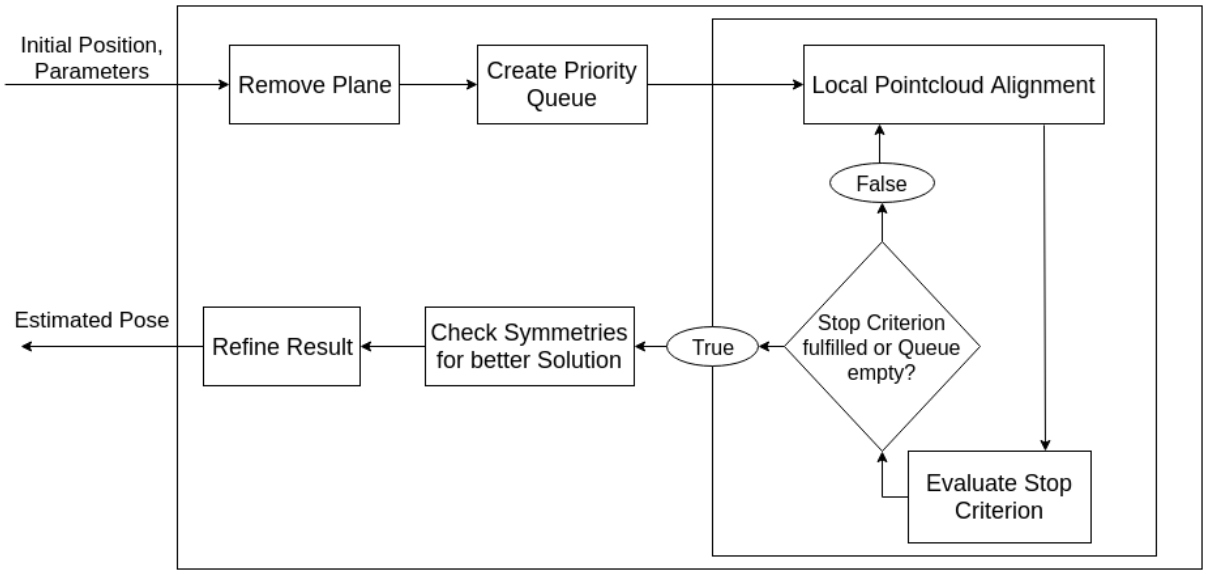


Figure 5.10.: A Schematic Visualization of the Final Algorithm

An algorithm that estimates a transformation matrix has been proposed, connecting two partially-overlapping point clouds, given only a rough estimate of the position, scale and of the size of the target object.
After preprocessing the data and, if applicable, removing a plane from the target cloud, a priority queue is created. The priority queue contains samples that iteratively refine a search space that defines all possible rotation parameters.
The points from the queue are evaluated by an locally convergent algorithm and the current best solution is stored until a time-dependent stop-criterion is fulfilled. Afterwards, symmetry checks for better estimates and a refinement step are executed.
An overview of the algorithm can be found in figure 5.10.

6 Integration

This section will describe the integration of the algorithm in the utilized set-up OCS.
It has already been mentioned that the point cloud alignment algorithm is implemented as a ROS *action server* that processes point clouds as input and returns the estimated pose. The requests are sent by a ROS *action client* accordingly.
The server will handle requests from arbitrary clients and can therefore be integrated in different set-ups. This can be seen as a separation into a model and a view part that allows for good adaptability.
The model part contains the algorithm with a provided interface that operates independently of the processes in the view part. The view part represents a client with an interface that visualizes data and allows the user to interact with the model part.
Figure 6.1 depicts the separation of the OCS environment and the processes inside the model and view parts.
The two parts will now be reviewed individually.

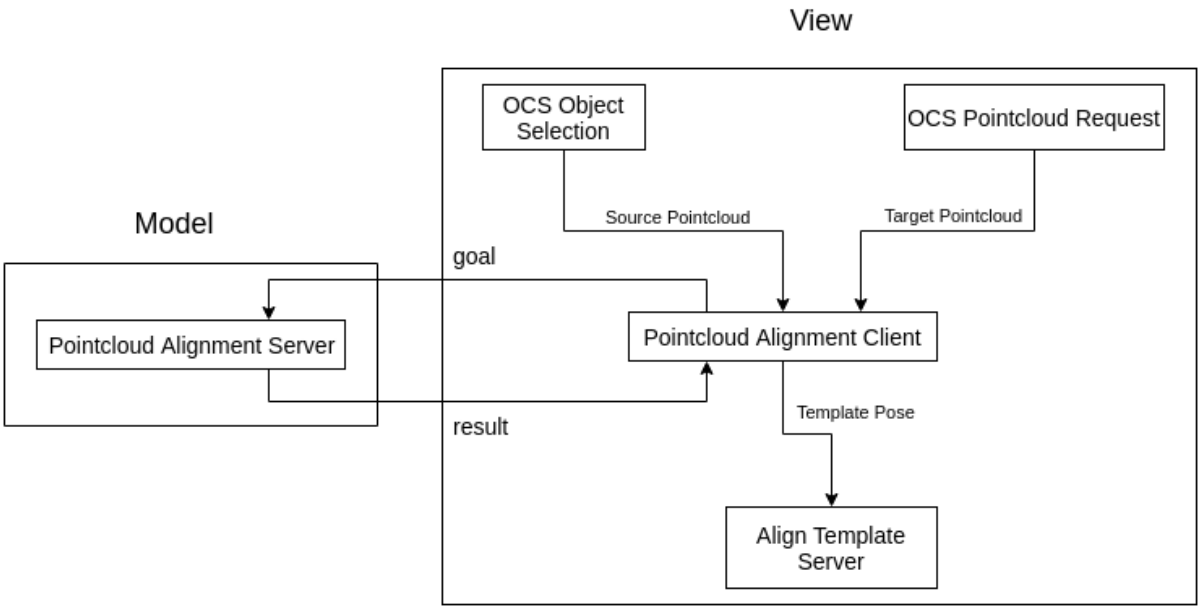


Figure 6.1.: Visualization of the Separation in Model and View Part

6.1 Model Part

The model part provides the action server that takes requests for the point cloud alignment. The implementation as a ROS action server denotes the interface structure, consisting of a goal, feedback and a result.

The goal consists of two point clouds represented as *PointCloud2*-messages from the PCL library and an initial pose stored as a *PoseStamped* data structure that contains the initial position and orientation. The orientation is only used to allow for different initial alignments but the algorithm does not depend on the initialization in terms of the rotation.
After a goal has been sent to the server, the initialization and preprocessing starts and afterwards the actual algorithm is executed. Every time a new best solution has been found, a feedback message is sent to inform the user of the quality of the current best estimate. The feedback consists of the overlapping percentage of the current best estimate and of the corresponding per-point-error of the inliers.
As soon as the algorithm is finished, the result is returned in the goal data structure. The goal consists again of a *PoseStamped*-message, containing the estimated position and orientation of the source cloud relative to the target cloud.

6.2 View Part

The basic task of the view part is to allow for the user's fast interaction with the model part from the previous section.
Different solutions are possible as long as the interface of the model part is respected. The OCS environment depicted in figure 6.1 will now be presented and can be regarded as an example for an interface for the user.
In the middle of the view part, the *Point Cloud Alignment Client* that stores the necessary data and sends it to the server as goal can be seen. The OCS environment visualizes the data and allows the user to adjust the pose of the template (see figure 4.1 in section 4).
The user can request point clouds from a chosen region of the robot's environment via the *OCS Point Cloud Request* plugin. These point clouds are sent via ROS messages, stored by the point cloud alignment server and are later provided as target cloud to the model part. The client always discards old point clouds and stores the latest one. After requesting a template, the user is able to chose a template and insert it at a chosen position in the target cloud. This insertion is ignored by the client. Only after the template has been placed at the desired position for the initialization of the server and the user double-clicks on it, a message is sent and the template is saved as the source cloud by the client. This message is sent by the *OCS Object Selection* plugin.
The client receives the pose of the template and stores it as initial pose. Now the goal message is sent to the server and the point cloud alignment server is started. Note that this happens right after the double-click on the template, no further alignment command has to be sent.
The client then waits for the server to return and after the result is received, a request with the estimated pose is sent to the *Align Template Server*. This server updates the position of the template to the estimated pose in the provided visualization and the user can immediately see whether the result is correct.

7 Evaluation

Examples for the execution of the algorithm have been uploaded on Youtube [32].
This section will discuss the performance of the proposed algorithm that will be tested on different data sets that are are generated from simulations or have been captured with LiDaR-scanners of a robot.
A detailed documentation of all test sets can be found in appendix A.2. An overview of all template is depicted in figure A.1.
Because of the missing functionality of the OCS environment to scale objects, the tests will be executed without taking the scale into account. However, the provided implementation contains an integrated scale estimation that has been tested for operational capability on independent data.
The tests will be executed on an AMD FX-8350 CPU with 8x4 GHz clock frequency and 16 GB of RAM. All tests will be using the same parameters that can be found in appendix A.4.
Each test set has been assigned the correct solutions manually. However, it has to be noted that these solutions are not a ground truth, but have been verified by hand. A test set may have multiple correct solutions if symmetries exist and all of them are correct.
The correctness of a result is evaluated by its deviation to the correct solution. If the norm of the translation is smaller than 5% of the template's size and the error of the rotation matrix is smaller than 0.05, it is regarded as correctly aligned. The size of the template is defined as the maximum distance between a point from the template and its mid-point.
The execution time measurement starts right before the server is called and is terminated right after it has returned.

7.1 Tests on Simulation Data

At first, the tests will be executed on different test sets that are generated by a simulation. The tests will be evaluated with regard to the success probability and the time needed for the execution. The target cloud is generated by using the same object that is used to generate the source cloud. This implicates a perfect template, which is not the case for real world tests.
Each configuration is executed 1000 times with random initial orientations and a small error in case of the translation. The initial translation is calculated by adding a random offset to the correct solution. This offset is sampled from the inside of a sphere with a radius that is 15% of the size of the template that is used.
If a test set contains a surface in the environment of the template, the algorithm is

executed with both, activated and deactivated plane removal.
Table 7.1 shows the results of the tests.

Test Set	Success Rate [%]	Min Time [s]	Avg Time [s]	Max Time [s]	Plane Removal
drill_ 1	99.6	0.03	2.19	2.46	on
	98.6	0.25	2.13	2.44	off
drill_ 2	90.8	2.10	2.26	2.53	on
	3	2.11	2.39	2.80	off
drill_ 3	99.5	2.10	0.02	2.38	on
	8.6	0.59	2.29	2.68	off
drill_ 4	95.5	2.10	2.40	2.82	on
	88.33	2.03	2.14	2.61	off
drill_ 5	99.3	2.05	2.18	2.33	on
	98	2.01	2.12	2.32	off
hammer_ 1	100	1.92	2.02	2.16	on
	100	1.85	1.90	2.22	off
hammer_ 2	100	1.89	2.04	2.13	on
	0	1.87	1.97	02.59	off
ladder_ 1	45	2.68	3.13	4.29	on
	46.33	2.68	3.06	4.97	off
ladder_ 2	65	2.70	3.05	6.67	off
fire_ hose	94.4	1.91	1.98	2.13	on
	91.6	1.91	1.99	2.22	off
cup_ 1	49.4	1.92	1.99	2.22	on
	81.2	1.93	2.02	2.23	off
cup_ 2	54	1.95	2.05	2.41	on
	63.4	1.93	2.07	2.34	off
cutting_ tool_ 1	0	2.07	2.24	2.61	on
	0	2.03	2.18	2.51	off
cutting_ tool_ 2	0.6	2.08	2.30	2.766	on
	8	2.02	2.22	2.55	off
valve	100	1.94	2.06	2.27	on
	100	1.93	2.04	2.66	off
stair_ 1	23.6	1.76	2.09	2.83	on
	14.6	1.72	2.11	3.10	off
stair_ 2	18.4	1.74	2.09	2.81	on
	20.4	1.64	1.99	2.60	off

Table 7.1.: Simulation Data Test Results

Discussion

Overall, the algorithm performed well on the tests. Some of the results will be discussed individually in the following:

drill The drill template was tested the most often and in different poses with varying difficulty. If plane removal was activated, all tests have been estimated with a very high success probability. If plane removal was deactivated, it still performed well but failed in the difficult cases.

hammer The hammer is rather flat, which is why the plane (if not removed) provided a better solution than the desired one if the LiDaR-scanners were placed in an unfortunate perspective. This is why the algorithm failed in all executions of the *hammer_ 2* set.

ladder Good initialization is important for the ladder because shifting it for one rug causes large local minima. In addition, the top part of the template (that averts functional symmetries) is barely seen by the LiDaR-scanners from unfortunate perspectives.

cup The results of the cup are interesting because the algorithm performed better without plane removal. This may be the case because parts of the cup were assigned as parts of the plane and were therefore removed. This eliminates the very small size differences of the two sides of the cup which are important for the correct estimation of the rotation.

cutting_ tool As it could be expected, the algorithm failed due to nearly perfect symmetries in the shape of the cutting tool but no functional symmetries.

7.2 Distance Tests

This section will evaluate the dependence of the success probability on the accuracy of the initial guess.
The tests will be executed on the *drill_ 1-3* test sets that represent surface alignment problems with different difficulties. The deviation from the initial guess will be calculated by adding an offset to the correct solution. In contrast to the previous section where the offset was sampled from the inside of a sphere, it will now be sampled only from the surface of a sphere in order to assign all tests the same deviation.
The distance from the correct solution will be evaluated in steps of $0.02m$ in a range from 0 to $0.4m$. Each setting will be executed 200 times. The results are plotted in figure 7.2.

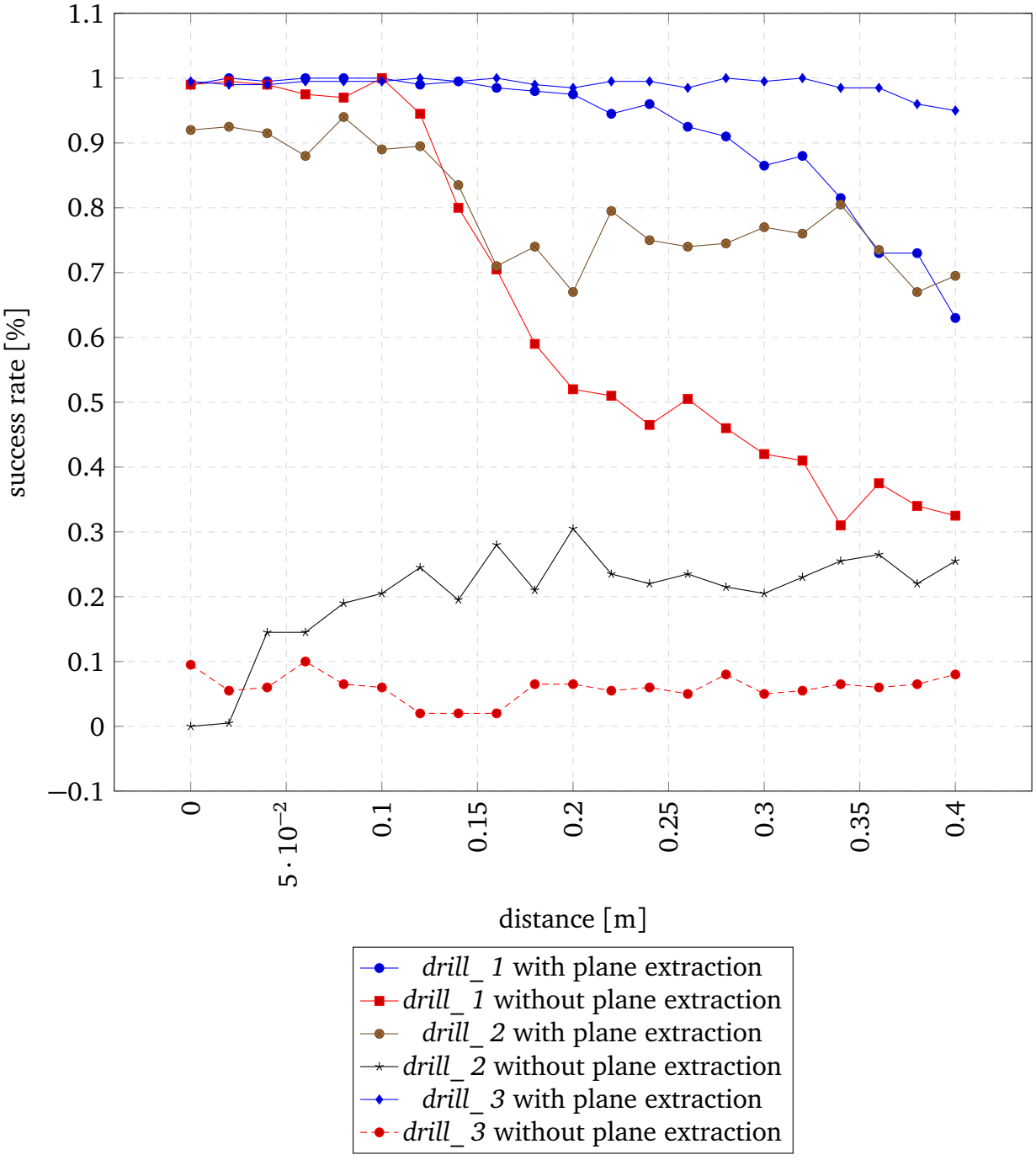


Figure 7.1.: Results of the Distance Tests

The horizontal axis depicts the deviation of the initial guess from the correct solution, while the vertical axis represents the achieved success rate of the tests.

Discussion

It can be seen that the algorithm performed well in the easy cases, when the plane has been removed, even if the position deviation is greater than the size of the drill. Only in the case of the *drill_ 2* test set the success rate decreases a little bit (which is supposedly due to the target cloud that is split into two) but remains stable.
Without plane removal, the *drill_ 2-3* test sets performed badly but the success rate increases with ascending distance. This may be due to some of the initializations that are sampled so that the source cloud is placed upon the drill in the target cloud. Thereby, the plane is not taken into account much for the alignment.
The performance of the algorithm decreased on the *drill_ 1* set from a certain distance on, which is not very surprising because some of the initializations are placed under the plane and therefore are not assigned to the drill in the target cloud.

7.3 Accuracy Tests

The tests in this section compare the accuracy of the proposed algorithm to templates that have been aligned by a supervisor using the OCS environment.
Each test will be executed ten times by the operator and ten times by the proposed algorithm. The accuracy will be measured by the achieved overlapping percentage (with an evaluation threshold $\rho = 0.015m$) and the per-point-error of all inliers.
The possibility of measuring the distance to the optimal transformation parameters has been rejected because no ground truth exists and therefore, the correct parameter would have had to be determined by hand or by the algorithm, which would have conflicted with the test procedure.
In order to avoid an advantage of the algorithm that optimizes a cost function which depends on the points, independent point cloud sets for the source and target point cloud will be used for the accuracy evaluation.
The results of the tests are shown in table 7.2.
It can be seen than the algorithm performed better than the supervisor on all test sets.

	Supervisor Results		Algorithm Results	
Test Set	**Overlapping percentage [%]**	**per-point-error [m]**	**Overlapping percentage [%]**	**per-point-error [m]**
drill_ 1	66.37	0.01272	72.16	0.01187
drill_ 2	42.14	0.01517	47.15	0.01519
drill_ 3	61.27	0.01151	63.91	0.01092

Table 7.2.: Accuracy Tests Results

7.4 Tests on Real Data

The following tests are executed on data that has been captured with the LiDaR-scanners of the robot Johnny #5, as depicted in figure 7.2, in order to evaluate the performance of the algorithm in real world scenarios. All test sets are depicted in appendix A.3.
The objects have been scanned in different poses and varying perspectives. The templates used to estimate the pose of the objects are imperfect, meaning they do not accurately represent the object's shape. Especially the drill, that can also be seen in figure 7.2, was tested with two different templates: The first one was a template of a drill that was used at the DRC challenge but which does not resemble the drill used for the scans. The other template used was a rough generic version of a drill. The other templates fit better but not perfectly.
The solutions of the test sets are generated by hand, just like it was done for the simulation data sets.
The offset is also computed the same way, by sampling a distance deviation from the inside of a sphere with a radius that is 15% of the template's size.
The results of the tests are shown in table 7.3.

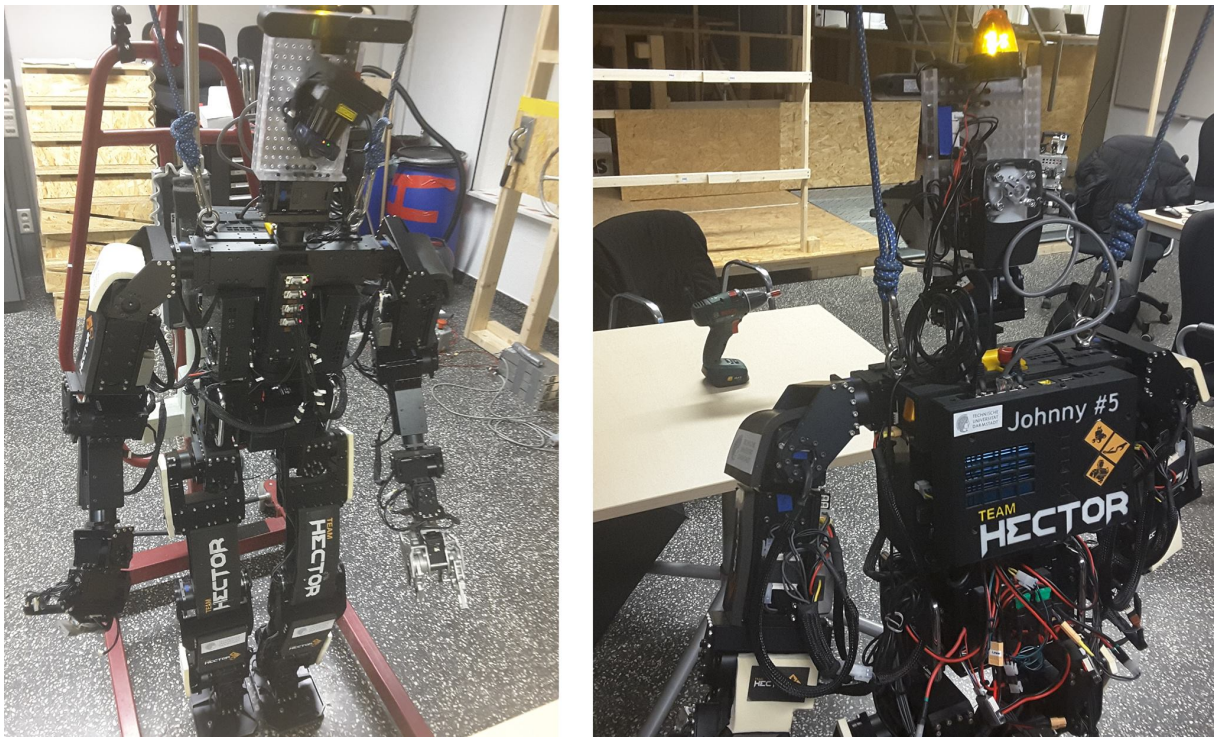


Figure 7.2.: Capturing Test Data with the Robot Johnny #5

Test Set	Template	Success Rate [%]	Min Time [s]	Avg Time [s]	Max Time [s]	Plane Removal
real_ drill_ 1	DRC_ drill	90.6 3.6	2.10 2.06	2.24 2.31	2.63 2.70	on off
	drill_ rough	83.6 79.6	2.11 2.11	2.25 2.26	2.47 2.71	on off
real_ drill_ 2	DRC_ drill	90.4 2.5	2.10 2.06	2.29 2.28	2.61 2.67	on off
	drill_ rough	76.67 0	2.08 2.07	2.26 2.24	2.64 2.70	on off
real_ drill_ 3	DRC_ drill	96.1 0.4	2.10 2.07	2.26 2.34	2.52 2.72	on off
	drill_ rough	88.7 0	2.07 2.10	2.21 2.26	2.49 2.64	on off
real_ drill_ 4	DRC_ drill	89.0 0.6	2.11 2.04	2.25 2.28	2.56 2.63	on off
	drill_ rough	95.64 0	2.05 2.01	2.17 2.32	2.60 2.67	on off
real_ drill_ 5	DRC_ drill	72.2 0	2.13 2.02	2.29 2.28	2.65 2.66	on off
	drill_ rough	90.04 62	2.02 2.02	2.14 2.21	2.62 2.74	on off
real_ valve	valve	100 100	2.00 1.93	2.12 2.04	2.39 2.66	on off
obstacle_ 1	obstacle	100 99	1.92 1.73	2.13 1.94	2.55 2.25	on off
obstacle_ 2	obstacle	95.3 72.6	2.14 1.67	2.41 1.93	2.78 2.33	on off
handle	handle	30 16.8	1.94 1.93	2.07 2.04	2.45 2.39	on off
cup_ 1	cup	38.5 78	1.89 1.88	1.97 1.99	2.31 2.69	on off
cup_ 2	cup	57.2 36.5	1.93 1.89	2.01 1.98	2.34 2.37	on off

Table 7.3.: Real Data Test Results

Discussion

Both templates used for the drill test sets performed well with activated plane removal. If planes were not removed, the performance broke down, especially when using the DRC drill. A detailed analysis showed that because of the imperfect templates, a wrong

solution with plane involved gave better results in both cases, per-point-error and overlapping percentage, than the desired solution.
The handle, however, did not perform very well. This may be due to noise that makes the handle in the target cloud hardly recognizable.
The difficulty when aligning the cup was again that the two sides of the template almost have the same extent, which is prone for errors.

8 Conclusion

Three dimensional pose estimation remains a challenging problem and no overall applicable solution exists. Surface alignment for pose estimation is a promising approach but different requirements demand customized and extended versions of common surface alignment techniques. Especially the computational complexity remains a big problem in many cases.

Contribution

This thesis proposed a novel approach for three dimensional surface alignment, specifically designed for scenarios where prior knowledge is available or can be obtained.
If an appropriate interface is available, it is easy for a human user to specify a rough guess of the position and scale of a surface, whereas manipulation of the orientation of objects turns out to be difficult and time-consuming in three dimensional applications.
Therefore, the proposed algorithm was designed so that it only depends on an easily providable initial guess in terms of translation and scaling factor, while operating completely independently in regard to the initial rotation of the surfaces.
The algorithm was applied to semi-autonomous pose estimation for avatar robots and therefore integrated in an existing visualization environment that allows to insert templates and to modify their pose.
By providing a good interface between user and the implemented algorithm, template-based pose estimation of object templates could be improved in terms of speed and accuracy.
The algorithm was evaluated on test sets that were designed according to the application in robotics to represent real robot scenarios.

Evaluation and Limitations

Overall, the proposed algorithm performed well on tests with simulation and real sensor data. It is also applicable in cases when only imprecise templates exist.
In practice, it turned out that the main problem is not to find the desired solution, but to differentiate it from suboptimal solutions. Different scenarios demand customized quality evaluation of the samples but generalizing the quality evaluation is a difficult task. Especially imperfect templates did not perform well on most of the criteria that can be used to identify desired solutions.
Large portions of outliers sometimes caused suboptimal solutions to have a better value of the quality evaluation function than the desired solution. Therefore, plane extraction,

which was first considered a small improvement, proved to be extremely important and to strongly improve some of the test cases, especially the ones with imperfect templates. This can be seen clearly in the tests of the real drill, when plane extraction could improve the result to have a better success rate of up to 95%.
Furthermore, the applicability of the algorithm is limited if the sought-after objects feature almost symmetrical shape but no functional symmetries. In these cases, the algorithm often fails due to a lack of information in the sensor data. In the tests, this can be seen when evaluating the sets of the cutting tool or the cup where the success rate was nearly down to zero percent.
Therefore, feedback from a human supervisor is essential to identify failed attempts. However, the presented interface allows to rotate object templates for each axis independently. Given that the non-symmetrical axes are correctly aligned, the error of the failed axis can be quickly corrected by the supervisor because only one rotational degree of freedom has to be changed, which can be done without influencing the other degrees of freedom. This can also be seen in the video where the manometer and the cutting tool were aligned by their non-symmetrical axes and afterwards, the last axis was adjusted manually [32].
In addition, specifically sampled point clouds can be used to improve behaviour in difficult cases. By sampling important regions of a point cloud (e.g. a switch of a tool) with a higher resolution and therefore assigning these areas more points, the modified areas also have a bigger impact on the cost function and have therefore a higher probability to be chosen as best drawn sample.

Usage in Practice

In the end, the supervisor should regard the algorithm as a helpful tool that has to be used according to the art of the problem. If possible, plane extraction should always be activated and the point clouds requested so that the plane can be easily detected.
In case of templates with symmetries, the algorithm should be used to align the non-symmetrical axes and afterwards the remaining axis can be aligned by hand.
In difficult cases, when the algorithm performs badly, the global search strategy can be avoided and only the local algorithm should be used. This can be also very helpful because the exact alignment is often the most time-consuming part and templates that are aligned by hand are less accurate.

Outlook

The proposed algorithm provides a good improvement to object manipulation approaches for avatar robots. However, various extensions are possible to further develop the autonomous abilities of avatar robots.
Since the plane removal proved to boost the algorithm's performance especially in diffi-

cult cases, further techniques for outlier removal could be used before the execution of the main algorithm.
The proposed algorithm could also be embedded in a procedure that uses state-of-the-art techniques for object recognition from computer vision. Thereby, regions of interest can be extracted from a camera of the avatar robot and be used to initialize the point cloud alignment algorithm with the template of the recognized object and its position. The operator's task would be to correct falsely identified objects or wrong pose estimations. With this method, a bigger part of object recognition and pose estimation could be relocated to the avatar robot to further reduce the interaction of the supervisor.

9 Bibliography

[1] XIONG, H., SZEDMARK, S., and PIATER, J. "Efficient, General Point Cloud Registration with Kernel Feature Maps". In: *Computer and Robot Vision (CRV), 2013 International Conference on*. 2013, pp. 83–90. DOI: 10.1109/CRV.2013.26.

[2] ROMAY, A. "An Object Template Approach to Manipulation for Semiautonomous Avatar Robots". In: *PHD Thesis* (2016).

[3] BESL, P. J. and MCKAY, H. D. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[4] CHEN, Y. and MEDIONI, G. "Object modeling by registration of multiple range images". In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. 1991, 2724–2729 vol.3. DOI: 10.1109/ROBOT.1991.132043.

[5] TURK, G. and LEVOY, M. "Zippered Polygon Meshes from Range Images". In: *In Computer Graphics Proceedings, Annual Conference Series* (1994).

[6] PALOMER, A. et al. "Bathymetry-based SLAM with difference of normals point-cloud subsampling and probabilistic ICP registration". In: *OCEANS - Bergen, 2013 MTS/IEEE*. 2013, pp. 1–8. DOI: 10.1109/OCEANS-Bergen.2013.6608091.

[7] KORN, M., HOLZKOTHEN, M., and PAULI, J. "Color supported generalized-ICP". In: *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*. Vol. 3. 2014, pp. 592–599.

[8] S. MARDEN, J. G. "Improving the Performance of ICP for Real-Time Applications unsing an Approximate Nearest Neighbour Search". In: *Proceedings of Australasian Conference on Robotics and Automation* (2012).

[9] CHETVERIKOV, D. et al. "The Trimmed Iterative Closest Point algorithm". In: *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. Vol. 3. 2002, 545–548 vol.3. DOI: 10.1109/ICPR.2002.1047997.

[10] ZINSSER, T., SCHMIDT, J., and NIEMANN, H. "A refined ICP algorithm for robust 3-D correspondence estimation". In: *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*. Vol. 2. 2003, II–695–8 vol.3. DOI: 10.1109/ICIP.2003.1246775.

[11] LOW, K.-L. "Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration". In: *Technical Report TR04-004, Department of Computer Science, University of North Carolina at Chapel Hill* (2004).

[12] SEGAL, A. V., HAEHNEL, D., and THRUN, S. "Generalized-ICP". In: *Proceedings of Robotics: Science and Systems* (2009).

[13] T. Zinsser J. Schmidt, H. N. “Point Set Registration with Integrated Scale Estimation”. In: *International Conference on Pattern Recognition and Image Processing* (2005).

[14] Liu, J. et al. “Fast and robust isotropic scaling probability iterative closest point algorithm”. In: *Chinese Automation Congress (CAC), 2015*. 2015, pp. 680–685. doi: 10.1109/CAC.2015.7382584.

[15] Rigid 3D Point Cloud Registration Algorithms, A. B. S. of. “Ben Bellekens, Vincent Spruyt, Rafael Berkvens, Rudi Penne, Maarten Weyn”. In: *International Journal on Advances in Intelligent Systems* (2015).

[16] Bellekens, B. et al. “A Survey of Rigid 3D Pointcloud Registration Algorithms”. In: *AMBIENT 2014: The Fourth International Conference on Ambient Computing, Applications, Services and Technologies* (2014).

[17] Rusinkiewicz, S. and Levoy, M. “Efficient variants of the ICP algorithm”. In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. 2001, pp. 145–152. doi: 10.1109/IM.2001.924423.

[18] Liying, W. and Weidong, S. “A review of range image registration methods with accuracy evaluation”. In: *2009 Joint Urban Remote Sensing Event*. 2009, pp. 1–8. doi: 10.1109/URS.2009.5137473.

[19] Wends S. Yambor Bruce A. Draper, J. R. B. “Analyzing PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures”. In: (2000).

[20] Jian, B. and Vemuri, B. C. “Robust Point Set Registration Using Gaussian Mixture Models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.8 (2011), pp. 1633–1645. issn: 0162-8828. doi: 10.1109/TPAMI.2010.223.

[21] Myronenko, A. and Song, X. “Point Set Registration: Coherent Point Drift”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.12 (2010), pp. 2262–2275. issn: 0162-8828. doi: 10.1109/TPAMI.2010.46.

[22] Lu, M. et al. “Accelerated Coherent Point Drift for Automatic Three-Dimensional Point Cloud Registration”. In: *IEEE Geoscience and Remote Sensing Letters* 13.2 (2016), pp. 162–166. issn: 1545-598X. doi: 10.1109/LGRS.2015.2504268.

[23] Boyd, S. and Mattingley, J. “Branch and Bound Methods”. In: *Notes for EE364b, Stanford University* (2007).

[24] Olsson, C., Kahl, F., and Oskarsson, M. “Branch-and-Bound Methods for Euclidean Registration Problems”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009), pp. 783–794. issn: 0162-8828. doi: 10.1109/TPAMI.2008.131.

[25] Li, H. and Hartley, R. “The 3D-3D Registration Problem Revisited”. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8. doi: 10.1109/ICCV.2007.4409077.

[26] Yang, J., Li, H., and Jia, Y. “Go-ICP: Solving 3D Registration Efficiently and Globally Optimally”. In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 1457–1464. doi: 10.1109/ICCV.2013.184.

[27] CAMPBELL, D. and PETERSSON, L. “GOGMA: Globally-Optimal Gaussian Mixture Alignment”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[28] HART, S., DINH, P., and HAMBUCHEN, K. “The Affordance Template ROS package for robot task programming”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 6227–6234. DOI: 10.1109/ICRA.2015.7140073.

[29] FALLON, M. et al. “An Architecture for Online Affordance-based Perception and Whole-body Planning”. In: *Computer Science and Artificial Intelligence Laboratory Technical Report* (2014).

[30] LU, M. et al. *Fitting Debris: Perception enabled by minimal human interaction*. 2015. URL: https://www.youtube.com/watch?v=_xBXgD7R0qo.

[31] K.S. ARUN T.S. Huang, S. B. “Least-Squares Fitting of two 3-D Point Sets”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1987).

[32] MÜLLER, S. *Semi-Autonomous Pose Estimation for Avatar Robots*. English. Jan. 2017. URL: https://www.youtube.com/watch?v=GinrJh4QCnI&list=PL03Is9em2eBaVlZj-8H-DFUPjlP_L71jx&index=1.

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
BnB	Branch and Bound
CAD	Computer Aided Design
CPD	Coherent Point Drift
CPU	Central Processing Unit
DRC	DARPA Robotics Challenge
EM	Expectation Maximization
E-Step	Expectation Step
ICP	Iterative Closest Point
GMM	Gaussian Mixture Models
GO-ICP	Globally Optimal ICP
LiDaR	Light Detection and Ranging
M-Step	Maximization Step
NN	Nearest Neighbour
M-Step	Maximization Step
MIT	Massachusetts Institute of Technology
OCS	Operator Control Station
OpenMP	Open Multi-Processing
PCA	Principal Component Analysis
PCL	Point Cloud Library
RAM	Random-Access Memory
RANSAC	Random Sample Consensus
ROS	Robot Operating System
SVD	Singular Value Decomposition

A Appendix

A.1 Templates

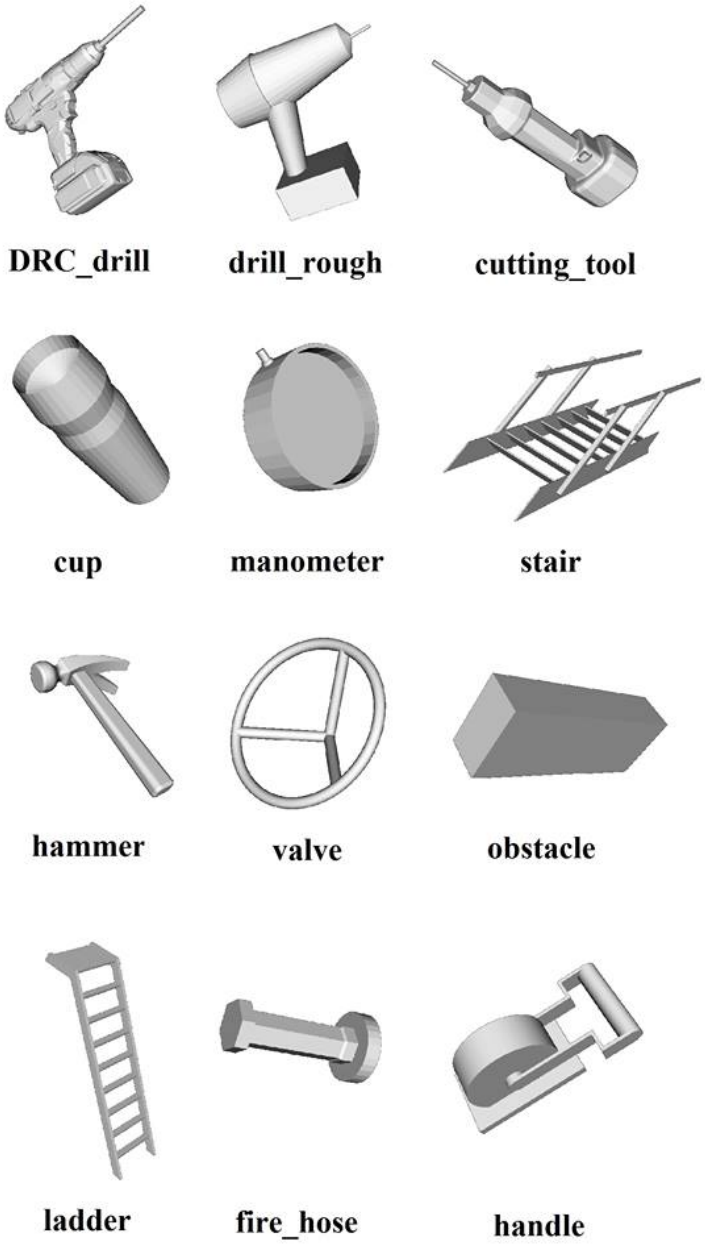


Figure A.1.: Overview of the Templates Used in this Thesis

A.2 Simulation Data Test Sets

The following data sets are depicted in increasing order from left to right and from top to bottom.

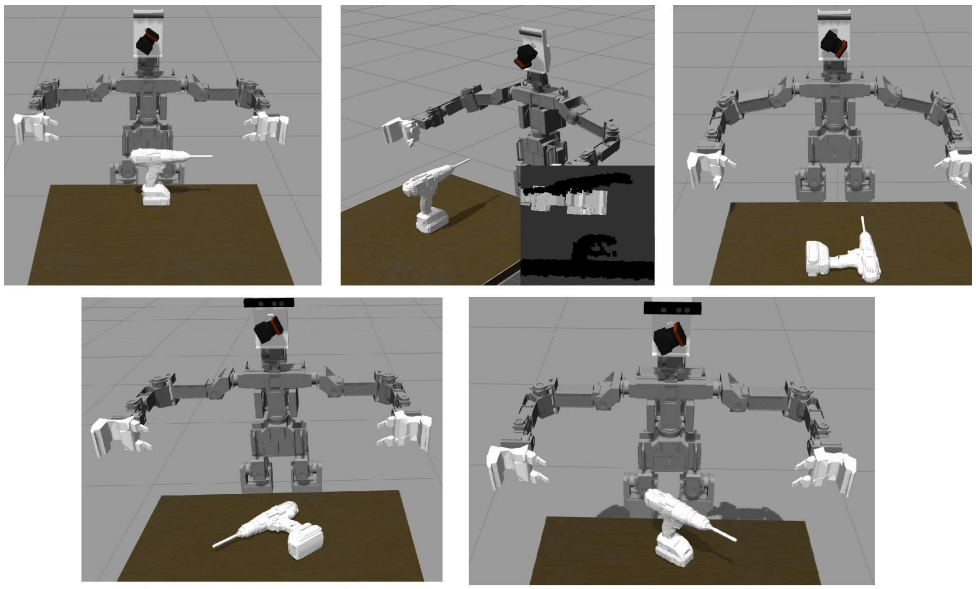


Figure A.2.: The *drill* Test Sets
In the bottom right of the second image the according point cloud of the set can be seen. Due to an unfavourable perspective, the cloud is split into two sub-clouds.

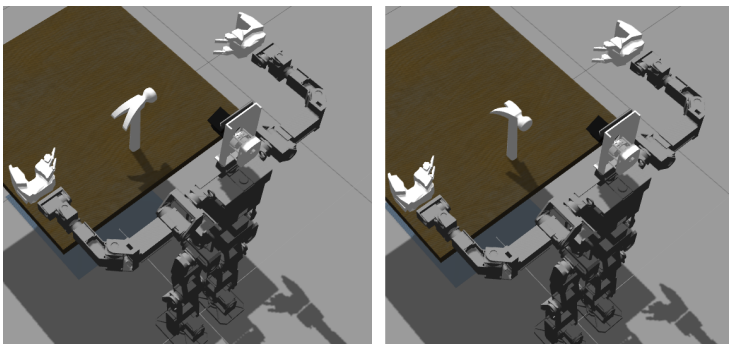


Figure A.3.: The *hammer* Test Sets

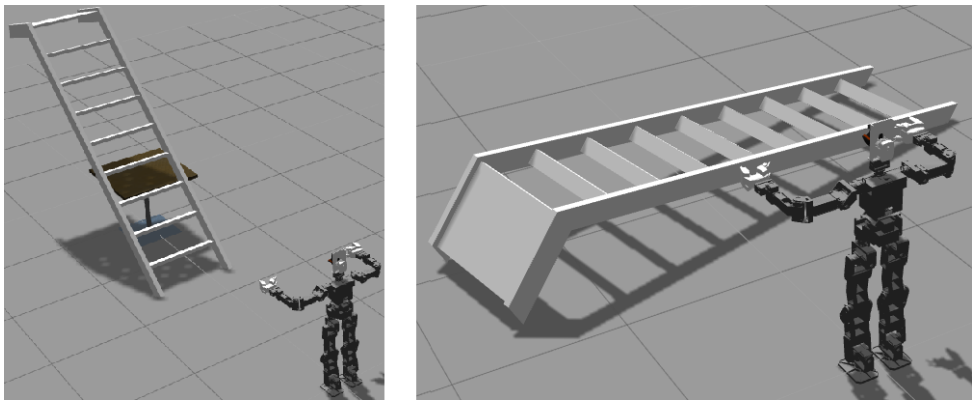


Figure A.4.: The *ladder* Test Sets

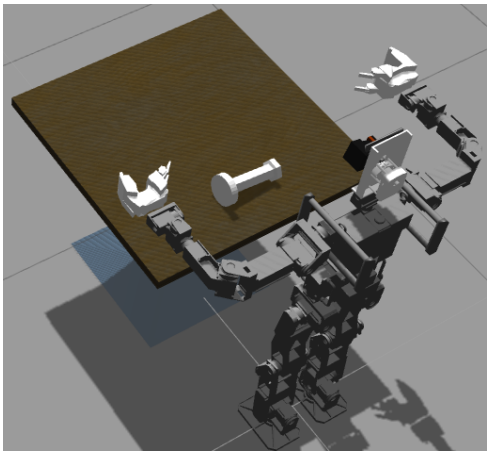


Figure A.5.: The *fire_ hose* Test Set

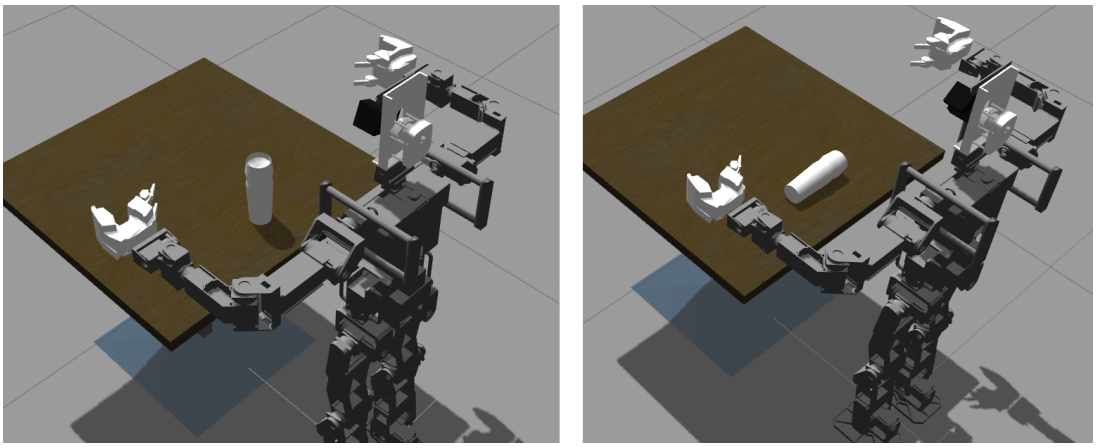


Figure A.6.: The *cup* Test Sets

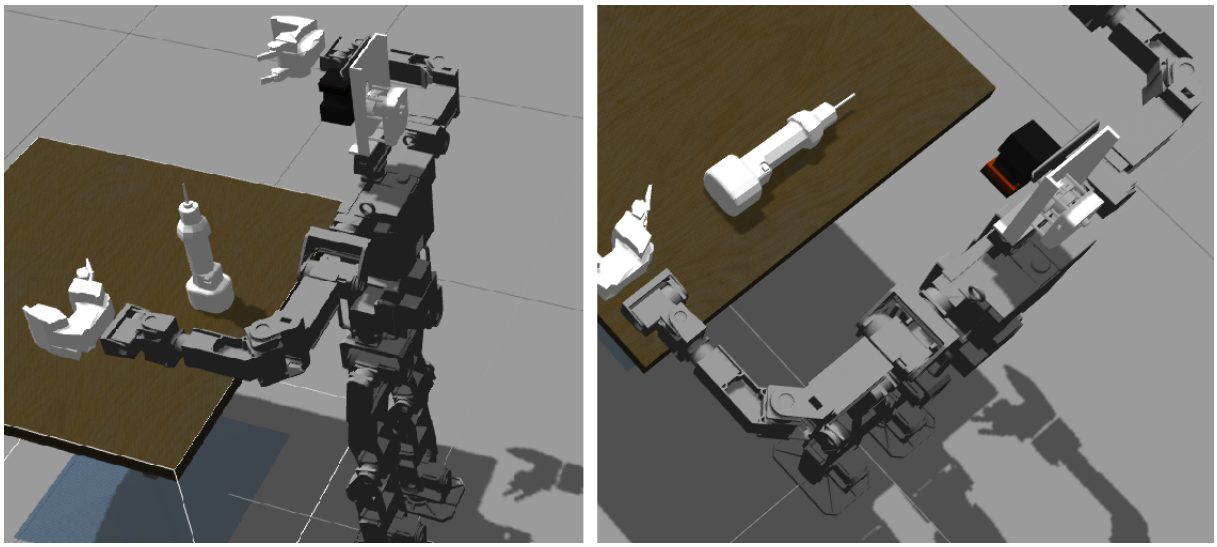


Figure A.7.: The *cutting_ tool* Test Sets

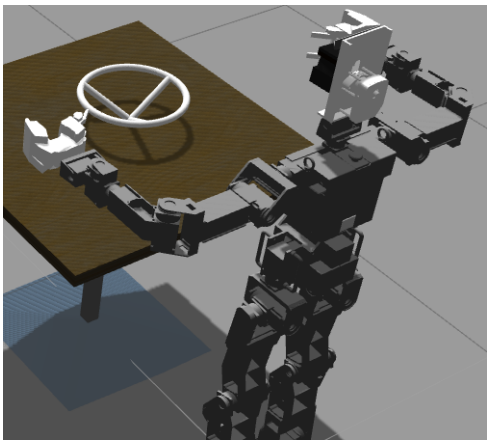


Figure A.8.: The *valve* Test Set

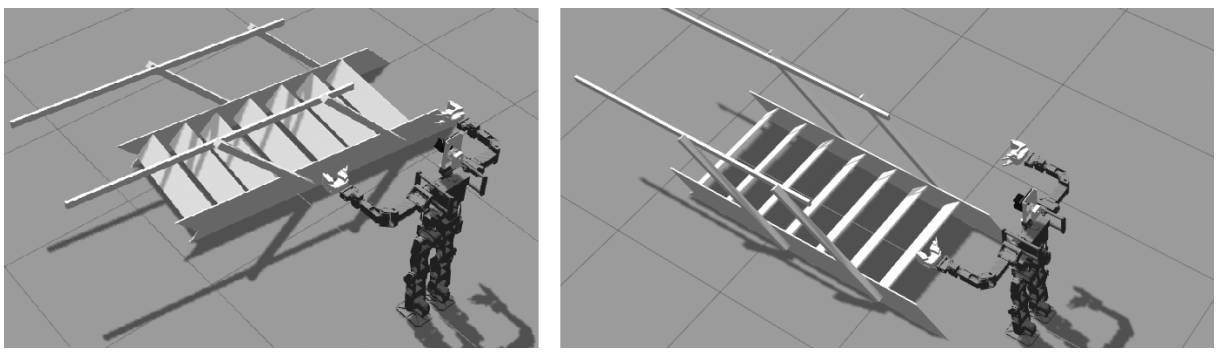


Figure A.9.: The *stair* Test Sets

A.3 Real World Data Test Sets

The following data sets are depicted in increasing order from left to right and from top to bottom.

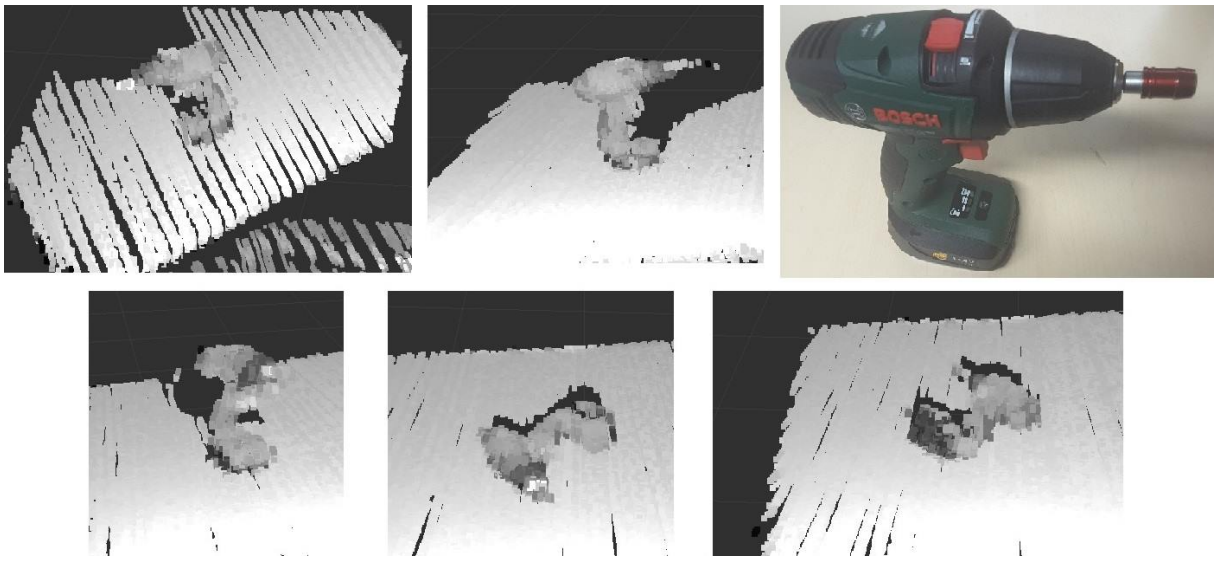


Figure A.10.: The *real_ drill* Test Sets

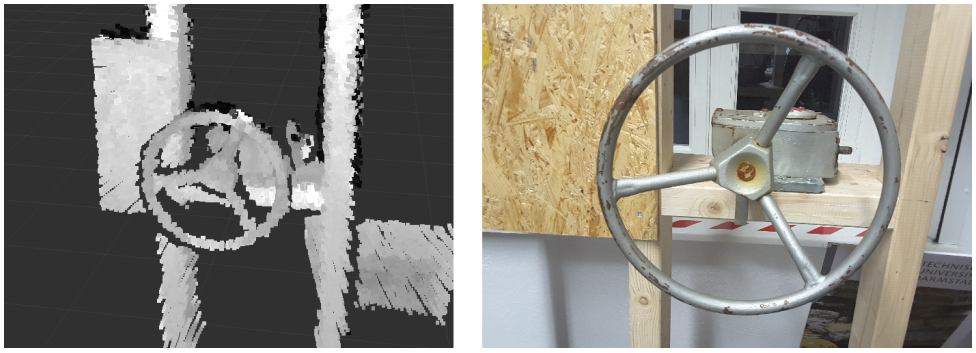


Figure A.11.: The *real_ valve* Test Set

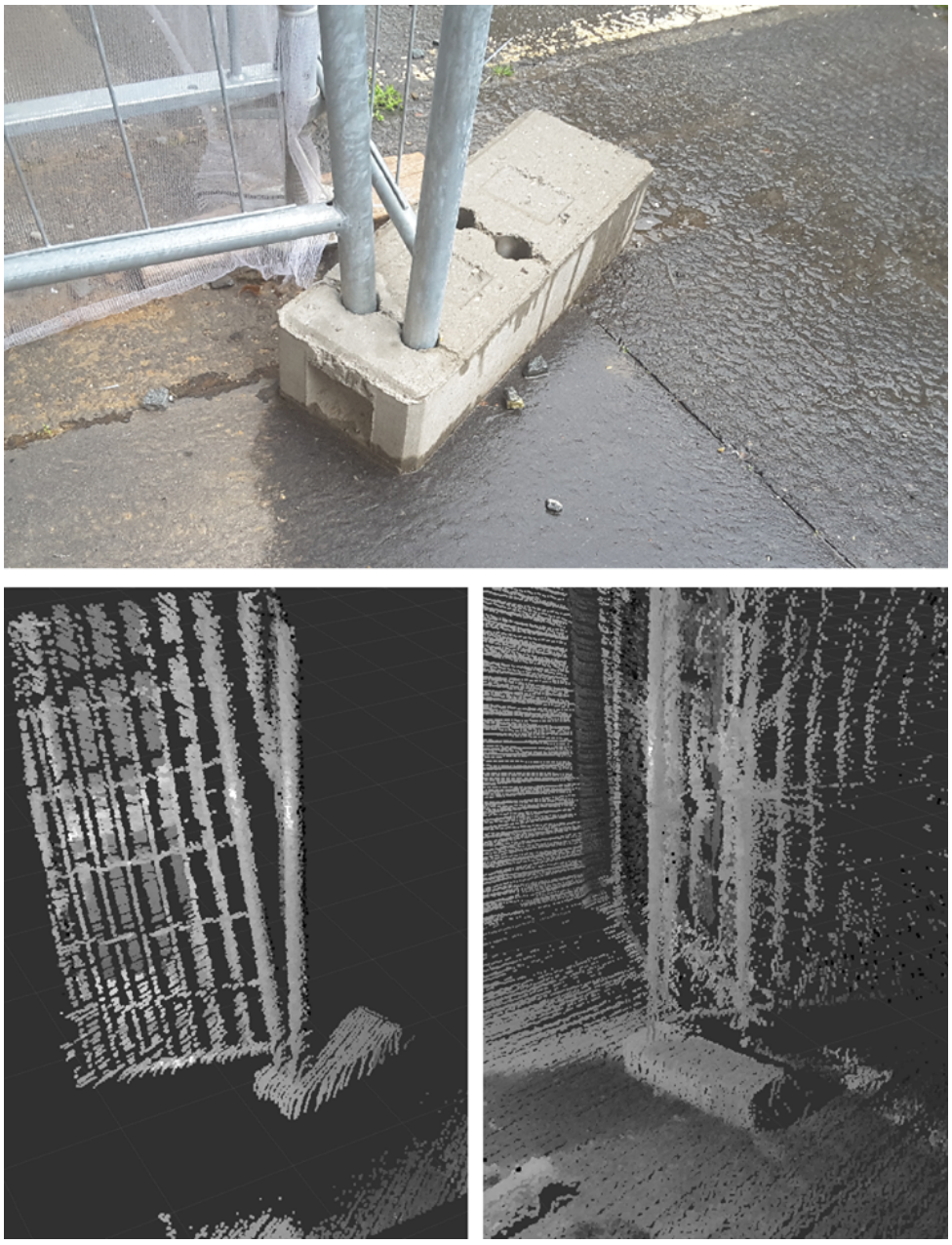


Figure A.12.: The *obstacle* Test Sets

Figure A.13.: The *handle* Test Set

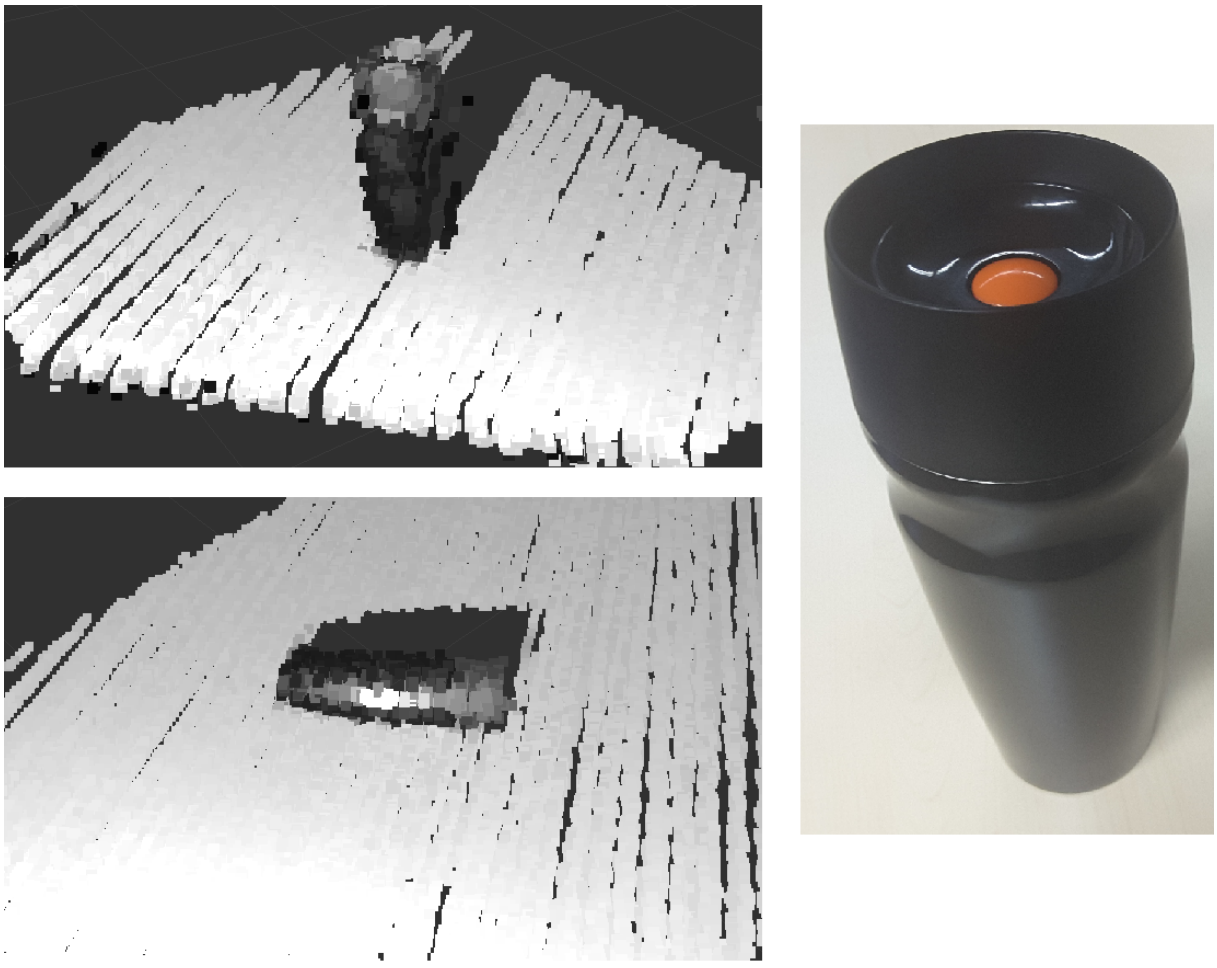


Figure A.14.: The *real_ cup* Test Sets

A.4 Test Parameter Values

Parameter	Symbol	Value
distance_ threshold	ζ	0.03
evaluation_ threshold	ρ	0.02
min_ overlapping_ percentage		0.15
target_ radius_ factor		2
number_ subclouds		5
size_ source		150
size_ target		2000
refinement_ icp_ source_ size		1000
refinement_ icp_ target_ size		5000
min_ plane_ portion		0.1
min_ plane_ distance		0.04
min_ scaling_ factor		0.8
max_ scaling_ factor		1.2
icp_ eps		$1.0e^{-5}$
icp_ eps2		$1.0e^{-3}$
max_ icp_ it		300
max_ numerical_ error		$1.0e^{-4}$
max_ percentage		0.85
damping_ coefficient	α	3
delay_ factor	β	0.2
max_ icp_ evaluations		600
lambda	λ	1
mu	μ	0
remove_ plane		0/1

Table A.1.: Parameters Used in the Tests

B Declaration of Academic Integrity

Thesis Statement pursuant to § 22 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Date: Signature: