CAD-based Pose Estimation - Algorithm Investigation

Annette Lef
Master of Science Thesis in Electrical Engineering

**CAD-based Pose Estimation - Algorithm Investigation:**
Annette Lef
LiTH-ISY-EX--19/5239--SE

**Supervisor:**  **Tuan Pham**  
IMT, Linköpings universitet  
**Kevin Kjellén, Filipe Marreiros, Anders Moe**  
SICK IVP

**Examiner:**  **Maria Magnusson**  
ISY, Linköpings universitet

---

*Computer Vision Laboratory*  
*Department of Electrical Engineering*  
*Linköping University*  
*SE-581 83 Linköping, Sweden*

Copyright © 2019 Annette Lef
Abstract

One fundamental task in robotics is random bin-picking, where it is important to be able to detect an object in a bin and estimate its pose to plan the motion of a robotic arm. For this purpose, this thesis work aimed to investigate and evaluate algorithms for 6D pose estimation when the object was given by a CAD model. The scene was given by a point cloud illustrating a partial 3D view of the bin with multiple instances of the object. Two algorithms were thus implemented and evaluated. The first algorithm was an approach based on Point Pair Features, and the second was Fast Global Registration. For evaluation, four different CAD models were used to create synthetic data with ground truth annotations.

It was concluded that the Point Pair Feature approach provided a robust localization of objects and can be used for bin-picking. The algorithm appears to be able to handle different types of objects, however, with small limitations when the object has flat surfaces and weak texture or many similar details. The disadvantage with the algorithm was the execution time. Fast Global Registration, on the other hand, did not provide a robust localization of objects and is thus not a good solution for bin-picking.
Acknowledgments

I would like to thank SICK IVP for the opportunity to perform my thesis work there. An extra big thanks to my supervisors Kevin Kjellén, Filipe Marreiros, Anders Moe and Ola Petersson at SICK for their valuable inputs and help. In addition, I would like to thank Martin Lundberg, also doing his master thesis at SICK, for helpful discussions during the thesis.

Last but not least, I would like to thank my examiner Maria Magnusson at Linköping university for help and feedback on the thesis.

Linköping, Juni 2019
Annette Lef
# Contents

Notation ix

1 Introduction 1
   1.1 Motivation ............................................ 1
   1.2 Problem formulation ................................... 2
   1.3 Limitations ........................................... 2
   1.4 Related work ........................................... 2

2 Theory 5
   2.1 Feature Extraction ...................................... 5
      2.1.1 Point Pair Feature (PPF) ......................... 6
      2.1.2 Point Feature Histograms (PFH) ................. 6
      2.1.3 Fast Point Feature Histograms (FPFH) .......... 7
   2.2 The Point Pair Feature approach ....................... 8
      2.2.1 Global model description ......................... 9
      2.2.2 Voting scheme ................................... 9
      2.2.3 Pose clustering .................................. 11
   2.3 Extension of the Point Pair Feature approach .......... 12
      2.3.1 Preprocessing .................................... 12
      2.3.2 Voting scheme, modification 1 .................... 12
      2.3.3 Voting scheme, modification 2 .................... 12
      2.3.4 Voting scheme, modification 3 .................... 13
      2.3.5 Postprocessing .................................. 13
   2.4 Fast Global Registration ................................ 14
      2.4.1 Correspondences .................................. 14
      2.4.2 Objective and optimization ....................... 14
   2.5 Iterative closest point (ICP) .......................... 17

3 Method 19
   3.1 Synthetic data creation ................................ 19
      3.1.1 CAD model ...................................... 22
   3.2 Algorithm implementation .............................. 23
      3.2.1 Preprocessing .................................... 23
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2 The Point Pair Feature approach</td>
<td>25</td>
</tr>
<tr>
<td>3.2.3 Extensions of the Point Pair Feature approach</td>
<td>26</td>
</tr>
<tr>
<td>3.2.4 Fast Global Registration</td>
<td>28</td>
</tr>
<tr>
<td>3.2.5 Parameter tuning</td>
<td>29</td>
</tr>
<tr>
<td>3.3 Evaluation</td>
<td>29</td>
</tr>
<tr>
<td>3.3.1 Top-1 recall</td>
<td>30</td>
</tr>
<tr>
<td>3.3.2 Total recall</td>
<td>30</td>
</tr>
</tbody>
</table>

| 4 Results                               | 33   |
| 4.1 The Point Pair Feature approach    | 33   |
| 4.2 Fast Global Registration           | 39   |

| 5 Discussion                            | 45   |
| 5.1 Results                             | 45   |
| 5.1.1 The Point Pair Feature Approach   | 45   |
| 5.1.2 Fast Global Registration          | 46   |
| 5.2 Method                              | 46   |
| 5.2.1 The Point Pair Feature Approach   | 46   |
| 5.2.2 Fast Global Registration          | 47   |

| 6 Conclusion                            | 49   |
| 6.1 Future work                         | 50   |

| Bibliography                            | 51   |
### Notation

#### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>Computer-Aided Design</td>
</tr>
<tr>
<td>PPF</td>
<td>Point Pair Feature</td>
</tr>
<tr>
<td>PFH</td>
<td>Point Feature Histogram</td>
</tr>
<tr>
<td>FPFH</td>
<td>Fast Point Feature Histogram</td>
</tr>
<tr>
<td>SPFH</td>
<td>Simplified Point Feature Histogram</td>
</tr>
<tr>
<td>FGR</td>
<td>Fast Global Registration</td>
</tr>
<tr>
<td>ICP</td>
<td>Iterative Closest Point</td>
</tr>
</tbody>
</table>
Introduction

3D registration has become a central part in computer vision and robotics. The problem lies in determining the transformation that best aligns two data sets to bring the data into the same reference system. Since technologies for 3D scanning are making progress and robots are being more involved in industrial processes, efficient and robust registration techniques are needed.

One fundamental task in robotics where registration techniques are used is random bin-picking. To plan the motion of a robotic arm it is necessary to be able to detect the object, given by a CAD model, in a partial 3D view of the scene and estimate its 6D pose, i.e. 3D translation and 3D rotation. To estimate the pose of an object is however a challenging problem since lighting conditions, clutter and occlusion affect the appearance of the objects in the image.

This thesis was performed at SICK IVP and aims to investigate and evaluate algorithms for object localization that can be used for the bin-picking problem.

1.1 Motivation

This thesis aims to investigate and evaluate algorithms for 6D pose estimation for the bin-picking scenario. The scene is given by a point cloud that illustrates a partial 3D view of a bin with multiple instances of an object for the robot to grasp. The object to be found is defined by a CAD model. The goal is to find the poses of the instances in the bin, such that a robot can grasp one of them. An example of how the scene might look and what type of object it can be is shown in figure 1.1. To the left is the scene as a point cloud and to the right is the CAD model of the object.
1.2 Problem formulation

The thesis aims to answer the following questions

- Can the Point Pair Feature approach [1] by Vidal et al. or the Fast Global Registration [2] by Zhou et al., described in chapter 2, achieve sufficiently robust localization of objects, defined by a CAD model, in point cloud data so that it can be used in the bin-picking scenario?

- Which one of the algorithms is preferred for the problem?

- Can the algorithms handle objects of different types or what structure of an object is needed for the solution to be robust?

1.3 Limitations

There exist a lot of different registration algorithms. In this thesis only two of them were chosen to be investigated if they can be used for bin-picking. This thesis also focuses on finding a solution with high accuracy and the computational speed was therefore not a decisive factor.

1.4 Related work

Many methods exist for 6D object pose estimation. Roughly they can be categorized as template matching methods, feature-based methods and learning-based methods.

In template matching methods [3, 4], a template is usually constructed by rendering the 3D model of an object. The template is then moved over the input image and a similarity score is computed at each location. By comparing these similarity scores the best match is obtained. These methods are useful for detecting objects with weak texture, but do not work as well when there is occlusion between the objects. If the object is occluded, the similarity score for the template is low.
In feature-based methods [5–7], features on the 3D model are matched with features in the image. These features can either be extracted from points of interest or from every pixel in the image. In contrast to template-based methods, feature-based methods can often handle occlusion between objects better. However, to compute the features there needs to be sufficient texture on the objects. Feature-based methods are commonly divided into two categories, local and global methods. The workflow typically consists of a global registration followed by a local refinement. The global registration computes an initial estimate of the rigid transformation between two surfaces and the local registration thereafter refines the estimate to obtain a tighter registration. Commonly used local registration methods are the Iterative Closest Point (ICP) and its variants [8].

Recently learning-based methods have become popular for pose estimation [9–11]. Some use machine learning techniques to learn feature descriptors. Others use convolutional neural networks on color images, RGB, or color and distance images, RGB-D, to estimate the object pose. Learning-based methods seem to be a powerful tool for pose estimation. However, they are limited by issues such as generalizability, learning of geometric invariance and computational efficiency [12].

At the 3rd international workshop on recovering 6D object pose at ICCV 2017, the SIXD challenge was organized [13]. The goal of the challenge was to evaluate methods for 6D object pose estimation and the results submitted to the challenge were published in the BOP benchmark paper [14]. The benchmark includes eight data sets in a unified format, a comprehensive evaluation of 15 recent methods and an online evaluation system for continuous submissions. The task that was evaluated reflects the bin-picking scenario and the conclusion was that for this task, methods based on point pair features perform best. The top-performing method in the challenge was the method by Vidal et al. [1]. It is based on the point pair feature approach by Drost et al. [6]. The benchmark also concludes that occlusion is a big problem for current methods. In addition, it shows that methods that use RGB images also have problems with varying lightning conditions, which methods that only use depth images are more robust against.
This chapter presents the theory needed to understand the methods used in the thesis. First, the features that are used are explained, and then the registration methods.

The registration problem lies in determining the transformation that best aligns two data sets to bring the data into the same reference coordinate system. Here, one data set is the model, i.e. the CAD model of the object converted to a point cloud, and the other data set is the scene, a point cloud of a bin with multiple instances of the object. Points in the model are denoted as $m_i \in M$, where $i = 1, ..., N_m$, and points in the scene are denoted as $s_i \in S$, where $i = 1, ..., N_s$. $N_m$ and $N_s$ are the number of points in the model and the scene respectively. The problem is therefore to find a transformation that fits the model to the scene so that an object in the scene can be found.

### 2.1 Feature Extraction

A way to describe an image is to use features. They can either be extracted from points of interest, such as edges or corners, or from every point in the image. A common approach is to use the neighborhood of the points to create a description of the feature. Such a descriptor should be descriptive, compact and robust to a set of nuisances.

In this thesis, Point Pair Features, PPF, and Fast Point Feature Histogram, FPFH, are used. These are the features suggested by the registration algorithms that are going to be investigated. To understand the Fast Point Feature Histograms, the Point Feature Histograms, PFH, are explained first.
2.1.1 Point Pair Feature (PPF)

The Point Pair Feature, PPF, [6, 15] uses oriented points to represent the scene and the model. The four-dimensional feature $F$ for two points $p_1$ and $p_2$ is defined as

$$F(p_1, p_2) = (F_1, F_2, F_3, F_4) = (\|d\|_2, \angle(n_1, d), \angle(n_2, d), \angle(n_1, n_2)),$$

where $n_1$ and $n_2$ are the normals, $d = p_2 - p_1$ and $\angle(a, b) \in [0; \pi]$ is the angle between the vectors $a$ and $b$. This is illustrated in figure 2.1. The feature is asymmetric and invariant to rigid motions.

![Figure 2.1: The Point Pair Feature of two oriented points. Inspired by [6].](image)

2.1.2 Point Feature Histograms (PFH)

Point Feature Histograms, PFH, [16] uses the neighborhood of a point to create a histogram of values to describe the point in a scene. The goal when the feature was constructed was that the feature space should have high discriminating power, be invariant to 3D rotations and translations, and be insensitive to point cloud density and noise.

The Point Feature Histograms are computed from a set of points, $P = \{p_1, ..., p_N\}$, and their normals are denoted as $n_i$, $i = 1, ..., N$. For each point $p_i \in P$, all other points enclosed in a sphere with radius $r$ centered in the point $p_i$ are selected, i.e. the $k_i$ nearest neighbors. This is illustrated in figure 2.2 where the dotted circle illustrates the sphere. All points in the sphere are paired with each other as in the figure. For every pair, $p_{j_1}$ and $p_{j_2}$ ($j_1 \neq j_2, j_2 < j_1$), the point with the smaller angle between its normal and the line between the points, is set to be the source point, $p_s$, and the other the target point, $p_t$. From the source and the target, a Darboux frame, with the origin in the source point, is defined as

$$u = n_s, \quad v = \frac{(p_t - p_s) \times u}{\|p_t - p_s\|}, \quad w = u \times v.$$

After this, a set of 4 features are calculated from the point pair and their normals. The features are given by
Figure 2.2: The k-neighborhood of a point $p_i$ for calculation of the PFH, where all pairs of points are connected with a line. Inspired by [17].

\[ f_0 = v \cdot n_t, \]
\[ f_1 = \|p_t - p_s\|, \]
\[ f_2 = \frac{u \cdot (p_t - p_s)}{f_1}, \]
\[ f_3 = \arctan(w \cdot n_t, u \cdot n_t), \]

which represent angles between normals and the distance vector between the points. The four features are then used to calculate an index of the bin in the histogram which is increased by one. According to the implementation in PCL, in contrast to [16], the index is calculated with

\[ idx = \sum_{i=0}^{i \leq 3} \left\lfloor \frac{(f_i - f_{i\text{min}})d}{f_{i\text{max}} - f_{i\text{min}}} \right\rfloor d^i, \]

where $f_{i\text{max}} - f_{i\text{min}}$ is the maximal theoretical range between features, $d$ is the number of bins each feature is quantized in and $\lfloor \rfloor$ is the floor function. When all point pairs in the sphere have been processed, each bin is normalized with the total number of point pairs in the sphere.

By using these four features, the number of histogram bins is $d^4$. So, quantizing each feature into 2 bins, $d = 2$, would give 16 bins in total. Since the number of dimensions increases exponentially by the power of four, increasing $d$ results in a large number of extra dimensions for each point.

### 2.1.3 Fast Point Feature Histograms (FPFH)

For the Point Feature Histograms, the theoretical computational complexity is $O(Nk^2)$, where $N$ is the number of points in the point cloud and $k$ is the number of neighbors for each point. In dense point neighborhoods, the computation of PFH can be a bottleneck in real-time applications. Therefore, Rusu et al. proposed the Fast Point Feature Histograms, FPFH, [17] which is a simplification of
the PFH with a computational complexity of $O(Nk)$. Even though the computational complexity is reduced, most of the discriminative power of the PFH is preserved.

For the computation of the Fast Point Feature Histograms, the Simplified Point Feature Histograms, SPFH, is introduced. The SPFH is calculated in the same way as the PFH except that point pairs is only created between $p_i$ and its neighbors. This is illustrated in figure 2.3 which can be compared to figure 2.2 for the PFH. When the SPFH is calculated for all points in the point cloud, the FPFH is calculated by re-determining the points k-neighborhood using the neighboring SPFH values as follows

$$FPFH(p_i) = SPFH(p_i) + \frac{1}{k} \sum_{j=1}^{k} \frac{1}{\omega_j} SPFH(p_j),$$

(2.5)

where $\omega_j$ is the distance between point $p_i$ and a neighboring point $p_j$.

![Figure 2.3: The k-neighborhood of a point $p_i$ for calculation of the SPFH, where the point pairs of $p_i$ and its neighbors are connected with a line. Inspired by [17].](image)

When calculating the PFH four features are used. However, experiments show that excluding $f_1$, the distance between the points, didn’t decrease the robustness significantly. Therefore, FPFH excludes this feature and only uses the other three. By doing this the number of bins in the histogram is reduced to $d^3$ compared to $d^4$ for PFH. However, when using this fully correlated feature space, some of the bins in the histogram will be empty. A simplification, that optimizes the computational complexity of the PFH further, is to instead use 3 separate histograms, one for each feature. By concatenating them, a histogram with $3d$ bins is then created.

### 2.2 The Point Pair Feature approach

The Point Pair Feature approach by Drost et al. [6], is based on the idea that the scene and the model are represented by finite sets of oriented points. Given a
point cloud, such representations are easily computed. The approach can be divided into an offline phase and an online phase and is described in the following sections which are based on [6].

As a first step, both the model and the scene are preprocessed. The point clouds are subsampled such that there is a minimum distance between all points. This distance, denoted as $d_{\text{dist}}$, is set relative to the diameter of the model and is given by

$$d_{\text{dist}} = \tau_d \text{diam}(M),$$  \hspace{1cm} (2.6)

where $\tau_d$ is the sampling rate, and $\text{diam}(M)$ is the diagonal of a bounding box around the model. Drost et al. set the default sampling rate to be 0.05.

### 2.2.1 Global model description

In the offline phase, a global description of the model is created with the use of Point Pair Features described in section 2.1.1. For all pairs of points $m_i, m_j \in M$ on the surface of the model a feature vector is calculated with (2.1). The feature vectors are discretized by sampling the distances in steps of $d_{\text{dist}}$, given by (2.6). The angles are discretized in steps of $d_{\text{angle}}$, which is given by

$$d_{\text{angle}} = 2\pi/n_{\text{angle}},$$  \hspace{1cm} (2.7)

where $n_{\text{angle}}$ was chosen to be 30 by Drost et al. Equal discrete feature vectors are then grouped together and the point pairs are stored in a hash table indexed by the quantized PPF.

### 2.2.2 Voting scheme

In the online phase, a set of approximately evenly distributed points in the scene are selected as reference points. If we assume that one of the reference points, $s_r \in S$, lies on the searched object in the scene, then there is a point on the model, $m_r \in M$, that corresponds to $s_r$. By aligning $m_r$ and $s_r$, and their normals, the model can be aligned to the scene by rotation of the object around the normal of $s_r$. A simple example of this is illustrated in figure 2.4 where both the scene and the model are the conrod. The rigid motion between the model and the scene can thus be described by the point on the model and the angle of the rotation, $(m_r, \alpha)$. This pair is defined as the local coordinates of the model with respect to the scene.

By using the local coordinates of the model, the transformation between a point pair on the model, $(m_r, m_i)$, and on the scene, $(s_r, s_i)$, that have similar feature vectors can be defined as

$$s_i = T_s^{-1}R_x(\alpha)T_m m_i.$$  \hspace{1cm} (2.8)

$T_m$ and $T_s$ translates $m_r$ and $s_r$ respectively to the origin and rotates their normals $\mathbf{n}_m^r$ and $\mathbf{n}_s^r$ onto the x-axis. $R_x(\alpha)$ is a rotation around the x-axis with the angle $\alpha$ to align $s_i$ and $m_i$. This is illustrated in figure 2.5.
Figure 2.4: An illustration of how the model and the scene can be aligned with two corresponding points. (a) Scene and model with their normals from corresponding reference points. (b) After aligning the reference points and their normals the model can be rotated around the normal to align the model to the scene.

Figure 2.5: Transformation between model and scene pairs where the two point pairs have similar features F. Inspired by [6].

To find the optimal local coordinates for a fixed reference point a voting scheme, similar to the generalized Hough transform [18], is used. A two-dimensional accumulator array representing the discrete space of local coordinates is created. The size of the accumulator array is the number of model sample points times the number of sample steps of the rotation angle $\alpha$.

Figure 2.6 illustrates the voting process and the steps are as follows.

a) The fixed reference point, $s_r$, is paired with all other points in the scene, $s_i \in S$, and their Point Pair Features $F(s_r, s_i)$ are calculated according to (2.1).
b) The feature \( F(s_r, s_i) \) is used as a key to the hash table of the global model description.

c) Model pairs, \((m_r, m_i)\), with similar features are retrieved from the hash table.

d) The rotation angle, \( \alpha \), is calculated according to (2.8) for all model pairs matched with the point pair in the scene.

e) Votes are cast in the accumulator array for the local coordinates \((m_r, \alpha)\).

The optimal local coordinate is thereafter found as the peak in the accumulator array and a global rigid transform can be calculated.

**Figure 2.6:** Illustration of the voting process for a reference point \( s_r \). Inspired by [6].

To speed up the calculations of the rotation angle, \( \alpha \) is divided into two parts, \( \alpha = \alpha_m - \alpha_s \), where \( \alpha_m \) is related only to the point pair on the model, and \( \alpha_s \) is related only to the point pair on the scene. The rotation angle \( \alpha \) is defined as the rotation around the x-axis. Therefore, by projecting \( T_m m_i \) on the yz-plane, \( \alpha_m \) can be defined as the rotation angle between the projection and the positive y-axis. It is thus possible to calculate \( \alpha_m \) in the offline phase and store it in the global model descriptor. \( \alpha_s \) can, in the same way, be defined as the rotation angle between the projection of \( T_s s_i \) on the yz-plane and the positive y-axis.

### 2.2.3 Pose clustering

To ensure that a reference point lies on the surface of the searched object the voting is done for multiple reference points. Drost et al. used 1/5th of the points in the subsampled scene as reference points. From every reference point, a possible object pose is retrieved. To remove incorrect poses and increase the accuracy, the poses are clustered such that the differences in translation and rotation between the poses in the clusters are less than a predefined threshold. By adding the number of votes in the voting scheme, figure 2.6 e), for each pose in each cluster, the score of the clusters can be calculated. For the cluster with the maximum score,
an average of the poses is calculated giving the resulting pose. By averaging the poses in several clusters, several resulting poses can be returned, which is desired if the scene contains multiple instances of the object.

2.3 Extension of the Point Pair Feature approach

Many extensions to the point pair feature approach by Drost et al. exist. One extension is [15] by Hinterstoisser et al. Vidal et al. followed their analysis and proposed new improvements to the PPF approach [1]. Thereafter Vidal et al. extended the preliminary work and proposed more improvements to the approach [19]. The following sections describe some of the extensions of the approach and are based on [1, 15, 19].

2.3.1 Preprocessing

In the original PPF approach, during the preprocessing, the 3D points are subsampled such that there is a minimal distance between all points. However, when points close to each other have different normals, this leads to loss of useful information. To avoid this, points where the angle between the normals is larger than 30 degrees are kept even if the distance between them is smaller than the minimal distance.

2.3.2 Voting scheme, modification 1

In the online phase, Drost et al. pairs the reference points in the scene with all other points. To improve the run-time, the reference points are instead paired only with points that are closer than the model diameter. Point pairs are then created only with points that can belong to the same object. This is done by using a KD-tree structure, which is a data structure for organizing points in a $k$-dimensional space [20].

2.3.3 Voting scheme, modification 2

In the original PPF approach, all Point Pair Features are discretized so that the search in the hash table can be done in constant time. However, this can prevent features from being matched correctly, since sensor noise can cause similar features being discretized into different bins. A first approach to avoid this was to spread the point pairs and store them in both the bin indexed by the discretized feature vector and the 80 neighboring bins, i.e. $3^4 - 1$. However, this significantly increases the running time. In addition, this increases the correspondence distance, if $d_{dist}$ is kept, introducing pairs with less similarity being voted for. By instead spreading the point pairs only to neighbors that are more likely to be affected by noise, the worst case scenario is to spread the pairs to 15 neighboring bins, i.e. $2^4 - 1$. This is solved by checking the quantization error, $e_q = \left( \frac{F_i}{d_{dist}} - \left\lfloor \frac{F_i}{d_{dist}} \right\rfloor \right)$, for each dimension of the feature vector in (2.1). Which
neighbors that are likely to be affected by noise are then determined according to

\[
N(e_q) = \begin{cases} 
-1, & e_q < \frac{4}{3} \\
1, & e_q > (1 - \frac{1}{3}) \\
0, & \text{otherwise}
\end{cases}
\] (2.9)

The result should be interpreted as −1 indicating that the left neighbor could be affected, 1 indicating that the right neighbor could be affected, and 0 indicating that no neighbor is likely to be affected.

### 2.3.4 Voting scheme, modification 3

A drawback with the discretization and spreading is that it introduces bias in the votes. When similar scene pairs have the same model correspondence and similar scene angle \(\alpha_s\), giving the same discretized \(\alpha\), the local coordinates get multiple superfluous votes in the accumulator array. This leads to a deviation in the results. To avoid this an array indexed by the quantized PPFs is created. Every element in the array is a 32-bit integer where each bit corresponds to a quantized rotation angle \(\alpha_s\). The integer is initialized to 0 and a bit is set to 1 the first time the corresponding quantized PPF and rotation is voted for. Only when a bit is 0, voting is allowed for the corresponding quantized PPF and rotation.

### 2.3.5 Postprocessing

The score from the clustered poses may not be a good representation of how well the object pose fits the scene and there are two problems that can reduce the robustness of the score. The first problem is that in the scene some model points are self-occluded from the camera view which causes a deviation. The second problem is that there can be an alignment error between the model and the scene. To mitigate these problems, the object is rendered according to the most voted clustered poses. This is done by first transforming the model point cloud according to the pose and then removing points that are not visible from the camera. The poses are then refined by performing ICP, see section 2.5.

Thereafter, the scores of the poses are re-calculated. The new score is computed by counting the number of points in the rendered model cloud that are closer to the scene than some threshold. Vidal et al. set the threshold to \(d_{dist}/2\).

After the re-scoring, two filtering steps are applied to reject poses that do not correspond to an object. In the first step, the rendered view of the object is compared with the scene, and all points in the model cloud are classified as close to the scene, further away from the camera or closer to the camera. Points further away from the camera are points that may be occluded and points closer to the camera are non-consistent with the scene. If the percentage of occluded or non-consistent points is too high the pose is rejected. Vidal et al. reject a pose if more than 15% of the points are non-consistent or more than 90% of the points are occluded.

In the second filtering step, the silhouette of the object is extracted from the rendered view and compared to edges extracted from the scene. The edges are
obtained by identifying variation in depth and normals. If the average distance between silhouette points and edge points is larger than a threshold, the pose is rejected. Vidal et al. use 5 pixels as threshold.

2.4 Fast Global Registration

Fast Global Registration by Zhou et al. [2], is a registration method that doesn’t involve iterative sampling, model fitting or local refinement thus making the algorithm faster than many other global registration methods. It estimates a rigid transformation $T$ that aligns the model to the scene by minimizing an objective on correspondences. The following sections describe the method and are based on [2].

2.4.1 Correspondences

As a first step, correspondences between the model and the scene are calculated. This is done by using the FPFH described in section 2.1.3. Let the FPFH feature, given by (2.5), for a point $p \in P$ be $F(p)$ and let the set of FPFH for all points in the point cloud be $F(P)$. Correspondences are created by first going through all points in the model, $m \in M$, and finding the nearest neighbor of $F(m)$ among $F(S)$, and then going through all points in the scene, $s \in S$, and finding the nearest neighbor of $F(s)$ among $F(M)$.

However, this may lead to that the correspondence set includes a lot of outliers. Therefore a new correspondence set is created, for which a correspondence pair $(m, s)$ is added only if the points are mutually nearest. That means that the nearest neighbor for $F(m)$ among $F(S)$ is $F(s)$ and the nearest neighbor for $F(s)$ among $F(M)$ is $F(m)$. In addition, from the correspondence set with mutually nearest correspondences, a new correspondence set is created with only correspondences that are compatible. To test this, three correspondence pairs are picked randomly, $(m_1, s_1)$, $(m_2, s_2)$, $(m_3, s_3)$. If they meet the condition given by

$$\forall i \neq j, \quad \tau < \frac{|m_i - m_j|}{|s_i - s_j|} < 1/\tau,$$  \hspace{1cm} (2.10)

where $\tau$ is a threshold for the comparison, they are added to the set. Correspondences are picked randomly for $100 \times \text{size(correspondence set)}$ iterations. This gives the final correspondence set, $K$.

2.4.2 Objective and optimization

The objective of Fast Global Registration is to estimate the rigid transformation $T$ that minimizes the distances between corresponding points. The error function to be minimized is given by

$$E(T) = \sum_{(m, s) \in K} \rho(|s - Tm|),$$  \hspace{1cm} (2.11)
where \( \rho(\cdot) \) is a robust penalty and \( K \) is the correspondence set described in section 2.4.1. The robust penalty is used to disable spurious correspondences. It is therefore important to use an appropriate robust penalty function that automatically performs validation and pruning without an additional computational cost. The robust penalty that is used is the scaled Geman-McClure estimator,

\[
\rho(x) = \frac{\mu x^2}{\mu + x^2},
\]

that is shown in figure 2.7a for different \( \mu \). The parameter \( \mu \) controls which residuals that significantly effect the objective.

To simplify the minimization of the objective in (2.11), Black and Rangarajan's duality between robust estimation and line processes is used [21]. Line processes were first introduced to model discontinuities to for example be able to recover piecewise smooth surfaces. The main goals of robust estimation is to find a structure that best describes the data and to identify outliers, i.e. deviating points, or deviating substructures. Unifying line processes and robust estimation makes it possible to incorporate assumptions on the nature of discontinuities into the objective function.

By denoting a line process over the correspondences as \( L = \{l_{s,m}\} \), where \( 0 \leq l_{s,m} \leq 1 \) and indicates the presence or absence of a discontinuity, the objective can be written as

\[
E(T, L) = \sum_{(m,s) \in K} l_{s,m} \|s - Tm\|^2 + \sum_{(m,s) \in K} \Psi(l_{s,m}),
\]

where \( \Psi(l_{s,m}) \) is a prior. The prior can be thought of as a penalty for introducing a discontinuity, and is given by

\[
\Psi(l_{s,m}) = \mu \left( \sqrt{l_{s,m}} - 1 \right)^2.
\]

Thus, when there is no discontinuity, \( l_{s,m} \rightarrow 1 \), the penalty function goes to 0, and when \( l_{s,m} \rightarrow 0 \) and there is a discontinuity, \( \Psi(l_{s,m}) \rightarrow \mu \). When the objective is written as in (2.13), the partial derivate with respect to each \( l_{s,m} \) must be zero for \( E(T, L) \) to be minimized.

\[
\frac{\partial E}{\partial l_{s,m}} = \|s - Tm\|^2 + \mu \frac{\sqrt{l_{s,m}} - 1}{\sqrt{l_{s,m}}} = 0
\]

\[
\iff l_{s,m} = \left( \frac{\mu}{\mu + \|s - Tm\|^2} \right)^2.
\]

By substituting \( l_{s,m} \) into \( E(T, L) \) in (2.13), it becomes (2.11). Thus, the solution when optimizing \( E(T, L) \) is also optimal for \( E(T) \).

The main benefit of formulating the objective as in (2.13) is that by alternating between optimizing \( T \) and \( L \), the optimization can be performed efficiently.
By fixing \( \mathbb{L} \) when optimizing \( \mathbf{T} \), and vice versa, both steps optimize the error function and thus the algorithm guarantees convergence.

When \( \mathbf{T} \) is fixed, the error function is minimized when (2.15) is satisfied. When \( \mathbb{L} \) is fixed, the objective becomes a weighted sum of squared distances between corresponding points. To solve this, \( \mathbf{T} \) is linearized to \( \xi = (\omega, \mathbf{t}) = (\alpha, \beta, \gamma, a, b, c) \), where \( (\alpha, \beta, \gamma) \) is the rotational component \( \omega \) and \( (a, b, c) \) is the translation component \( \mathbf{t} \). \( \mathbf{T} \) can then be iteratively updated by

\[
\mathbf{T}^k \approx \begin{pmatrix} 1 & -\gamma & \beta & a \\ \gamma & 1 & -\alpha & b \\ -\beta & \alpha & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{T}^{k-1},
\]

where \( k \) is the current iteration and \( \mathbf{T}^{k-1} \) is the transformation estimated in the previous iteration. Equation (2.13) then becomes a least-squares objective on \( \xi \). By using the Gauss-Newton method, and defining \( \mathbf{r} \) as the residual vector of (2.13), and \( J_\mathbf{r} \) as its Jacobian, i.e. the matrix of all first-order partial derivatives, \( \xi \) can be computed by solving

\[
J_\mathbf{r}^T J_\mathbf{r} \xi = -J_\mathbf{r}^T \mathbf{r}.
\]

(2.17)

\( \mathbf{T} \) is then updated with (2.16).

---

**Figure 2.7:** (a) The Geman-McClure estimator for \( \mu = 0.25, 1, 4, 16 \). (b) Example of an objective function and how it is affected by varying \( \mu \).

The objective in (2.13) is non-convex and the parameter \( \mu \) controls its shape. It is used to create a convex approximation to the objective function that can easily be minimized, and it balances the effect of the prior term and alignment term. As \( \mu \) is adjusted, the minimum is tracked so that the objective function increasingly approximates the original non-convex estimation problem. The effect of varying \( \mu \) is illustrated in figure 2.7. The optimization starts with a large \( \mu \) to allow many correspondences and estimate a rough alignment. Then \( \mu \) is decreased during the
optimization to obtain a tighter alignment. It is decreased until $\mu = \delta^2$, where $\delta$ is a distance threshold.

2.5 Iterative closest point (ICP)

Iterative Closest Point [8] is a local registration algorithm often used to refine a solution given by a global registration method. The classical ICP estimates a rigid transformation, $T = (R, t)$, by minimizing the error function given by

$$E(T, M, S) = \sum_{i=1}^{N_m} \| (Rm_i + t) - s_j \|^2,$$

(2.18)

where $N_m$ is the number of points in the model and $(m_i, s_j)$ are corresponding points. The algorithm starts with an initial alignment, $T = (R, t)$, of the model. Correspondences are then found by calculating the closest point $s_j$ in the scene to each model point $m_i$. The closest point is given by

$$j = \arg \min_{j \in \{1, \ldots, N_s\}} \| (Rm_i + t) - s_j \|^2,$$

(2.19)

where $N_s$ is the number of points in the scene. A new transformation is then computed from the current set of correspondences and applied to the model.

Establishing closest-point correspondences, with (2.19), and recomputing the transformation, with (2.18), are then repeated until the change in the error function between two iterations is lower than some threshold.

ICP and its variants are very popular due to its simple concept, high usability and good performance. However, the algorithm requires a good initialization to avoid being trapped in a local minimum. Another issue of the method is the speed of computation. The basic ICP algorithms become very slow when there are a high number of points.
This chapter describes how the master thesis work was performed. First, the creation of synthetic data is described and then the implementation of the algorithms. At last, the evaluation of the algorithms is described.

As mentioned in the introduction, the goal of the thesis was to investigate and evaluate algorithms for 6D pose estimation that can be used for bin-picking. Two methods were thus chosen to be analyzed. The first was the Point Pair Feature approach [1] which extends a previous method using Point Pair Features [6]. This method was the top-performing method in the BOP benchmark that evaluated methods for the bin-picking task. The second method was the Fast Global Registration [2]. The article demonstrated that this algorithm is more than one order of magnitude faster than other global registration algorithms. It hasn't been tested for bin-picking. However, when aligning two scenes, seen from different viewpoints, it matched the accuracy of local refinement algorithms, such as ICP.

The Point Pair Feature approach returns a list with possible poses of objects, while Fast Global Registration only returns one pose. Thus, an object needs to be removed from the scene for the method to be able to find a second object.

### 3.1 Synthetic data creation

To be able to evaluate the algorithms, synthetic data with ground truth annotations were needed. The data were created by using software developed by SICK. It uses a physics engine to simulate objects being dropped in a bin. Both the object and the bin were given by CAD models.

A scene image was created by placing a random number of objects, within a predefined limit, above the bin. The objects were placed according to the following steps:
1. Define a volume above the bin such that if an object is placed inside the volume it will fall in the bin when being dropped.

2. Generate a random position and orientation inside the volume.

3. If the position and orientation are such that the object will be placed partly within another object, go back to step 2. Otherwise, place the object according to the position and orientation.

4. Repeat step 2 - 3 until all objects are placed in the volume above the bin.

After starting the simulation, the objects fell according to physical laws and ended up on the floor of the bin. A synthetic range camera was used to take an image of the scene and the output was a point cloud over the scene. Noise was added to the point cloud to make it more similar to real data. The positions and rotations of the objects after they had been dropped were then saved and used as ground truth.

Multiple scene images were created by repeating the procedure for one image. Every scene image then contained a various number of objects placed randomly to illustrate different situations that can occur in reality.

The CAD models that were used to create synthetic data for evaluation of the results are shown in figure 3.1. The models were chosen because of their differences in size and texture, to be able to see what types of objects the algorithms could handle. A data set with scene images was created for each of these objects. Examples of scene images are shown in figure 3.2. In these examples, the conrod was used. Every data set contained 50 scene images. The volume above the bin, explained above, and the maximum number of objects to be dropped were defined such that most parts of all objects were seen from the camera. This was done to simplify the evaluation by making sure that all objects would possibly be found by the algorithms. The maximum number of objects was dependent on the size of the object. The diameter of the object, the maximum number of objects there could be in a scene, and the total number of objects in the entire data set are shown in table 3.1. The diameter of the object was the diagonal of a bounding box around the object.

<table>
<thead>
<tr>
<th>Object</th>
<th>Diameter of object (mm)</th>
<th>Max number of objects in one scene</th>
<th>Total number of objects in data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conrod</td>
<td>213</td>
<td>30</td>
<td>676</td>
</tr>
<tr>
<td>Brake disc</td>
<td>342</td>
<td>15</td>
<td>370</td>
</tr>
<tr>
<td>Pipe</td>
<td>173</td>
<td>30</td>
<td>687</td>
</tr>
<tr>
<td>Crankshaft</td>
<td>555</td>
<td>10</td>
<td>232</td>
</tr>
</tbody>
</table>

*Table 3.1: The diameter of the object, the maximum number of objects a scene could contain, and the total number of objects in the entire data set.*
3.1 Synthetic data creation

(a) Conrod.  (b) Brake disc.

(c) Pipe.  (d) Crankshaft.

Figure 3.1: The CAD models used to create synthetic data.

Figure 3.2: Examples of scenes.
3.1.1 CAD model

The CAD models were also converted to point clouds. This was done as follows, where step 1 and 2 were done by using functions in FreeCAD [22] and the rest were done using functions in MeshLab [23]:

1. Convert the CAD model to solid.

2. Export the CAD model to a ply file. This step converts the model to a triangular mesh, which can be opened with MeshLab. The triangular mesh comprises a set of triangles, connected by their common edges and corners. The triangles are called faces and the corners are called vertices.

3. Re-orient all faces coherently. This step is to make sure that all face and vertex normals are pointing outwards from the object. For this step, it is important that the CAD model is solid.

4. If the vertices are sparse, use subdivision surfaces: midpoint, to create a denser point cloud. It substitutes each triangle with four smaller triangles by splitting every edge on its midpoint. See figure 3.3 for an example, where (a) is before subdivision, and (b) is after.

5. Re-compute face normals and then re-compute vertex normals. It is important that the face normals are calculated first, since vertex normals are calculated from the face normals.

6. At last, normalize the vertex normals.

The point cloud was then given by the vertices and their normals.

![Figure 3.3: Example of the density of a point cloud. The CAD model showing the object and the black dots are the vertices in the triangular mesh. (a) The brake disc when it is converted to a triangular mesh. (b) The brake disc when subdivision of the surfaces is performed.](image-url)
3.2 Algorithm implementation

The algorithms were implemented in C++ using the open source library PCL (point cloud library) [24]. PCL has functions for point cloud processing and visualization of point clouds, among those, implementation of PPF, FPFH, and ICP, thus facilitating the implementation. The experiments were performed on an Intel Xeon E5-1620 @ 3.60 GHz with 16 GB of RAM. The following sections describe how the implementation was done.

The inputs to the Point Pair Feature approach and Fast Global Registration were the point cloud of the scene and the point cloud of the model. Figure 3.4 shows an example of the inputs.

![Model point cloud.](image1)

![Scene point cloud.](image2)

Figure 3.4: Example input to the algorithms.

3.2.1 Preprocessing

The point clouds of the scene and the model were loaded from ply files. When the model was loaded both the points and the normals were read, thus creating a point cloud with normals. The scene from the synthetic data contains only the points and thus, the normals needed to be estimated. However, before the normals were estimated, points outside the bin and the sides of the bin were discarded to increase the speed of the algorithm.

The normals were then estimated by using PCL [25]. Estimating the normal of a plane tangent to the surface gives an approximation of the normal to a point on the surface. Estimation of the surface normal is thus solved by analyzing the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbors of the point. The covariance matrix for each point $p_i$ is defined as

$$ C = \frac{1}{k} \sum_{i=1}^{k} (p_i - \bar{p})(p_i - \bar{p})^T, \quad C v_j = \lambda_j v_j, \quad j \in \{0, 1, 2\} \quad (3.1) $$

where $k$ is the number of points in the neighborhood of $p_i$, $\bar{p}$ is the 3D centroid of the nearest neighbors, $\lambda_j$ is the $j$-th eigenvalue of the covariance matrix, and $v_j$ the $j$-th eigenvector. The value of $k$ was determined by testing different values, and was thereafter set to 5.
As a last preprocessing step, both point clouds were subsampled following the subsampling in the extended Point Pair Feature approach described in section 2.3.1. First, both point clouds were downsampled such that the densities of the point clouds were the same. The subsampling was then done using a voxel grid with voxel size as $d_{\text{dist}}$, see (2.6), and explained in algorithm 1. If the input to the algorithm was the model or scene cloud in figure 3.4 and $\tau_d = 0.05$, the output was as shown in figure 3.5a or 3.5b respectively. This subsampling was used for both the PPF approach and Fast Global Registration since it keeps useful information on the objects, while removing more points on planar surfaces, for example, the floor in the bin, which contain less useful information.

![Model point cloud after subsampling.](image1)

![Scene point cloud after subsampling.](image2)

*(a) Model point cloud after subsampling.* *(b) Scene point cloud after subsampling.*

**Figure 3.5:** Output of algorithm 1, given the model or scene point cloud in figure 3.4 as input.

---

**Algorithm 1:** Subsampling of the point clouds.

**Input:** Point cloud $P$ and $d_{\text{dist}}$

**Output:** Subsampled cloud

Save all points $p \in P$ in a voxel grid with voxel size $d_{\text{dist}}$

**for each voxel in the grid do**

**for each point $p$ in the voxel do**

**for each cluster in the voxel do**

if $\angle(n_p, n_{\text{point\_in\_cluster}}) < 30^\circ$ then

Add point to the cluster

Break

end

end

if point not added to any cluster then

Create new cluster with the point

end

end

Take the centroid of each cluster and add to the output cloud

end
3.2.2 The Point Pair Feature approach

The Point Pair Feature approach, described in section 2.2, was implemented using PCL [26] and the classes PPFEstimation, PPFHashMapSearch and PPFReg- istration. PPFEstimation was used in the offline phase to calculate the features and creating the global model description together with PPFHashMapSearch. PPFRegistration was used in the online phase to perform the matching of the point clouds. However, small parts of the classes were modified since the implementation did not match the documentation [27–29]. A summary of the algorithm is found in algorithm 2.

### Algorithm 2: The Point Pair Feature approach.

**Input:** Model and scene point cloud \((M, S)\), see figure 3.4  
**Output:** Transformations \(T\), called poses, that aligns \(M\) to \(S\)

Preprocessing of point clouds, see section 3.2.1  
Calculate the global model description  
for every reference point in the scene, \(s_r\), do  
  for every other point in the scene, \(s_i\), do  
    Calculate the Point Pair Feature \(F(s_r, s_i)\) according to (2.1)  
    Get model pairs with similar features from the hash table  
    for every model pair do  
      Calculate \(\alpha\) according to (2.8)  
      Vote for local coordinate \((m_r, \alpha)\)  
    end  
  end  
end  
Calculate possible pose from optimal local coordinate  
Cluster possible poses  
Average poses in clusters  
**return** poses with highest score

Several parameters in the algorithm could be adjusted to increase the performance. These were as follows

- **Sampling rate** \(\tau_d\): Multiplied with the diameter of the model to determine \(d_{\text{dist}}\) according to (2.6). \(d_{\text{dist}}\) was used both as the voxel size during sub-  
sampling of the point cloud and for the discretization of the distance in the feature vectors.
- **\(n_{\text{angle}}\)**: Determines \(d_{\text{angle}}\) according to (2.7). \(d_{\text{angle}}\) was used for discretiza-  
tion of the angles in the feature vector.
- **Scene reference point sampling interval**: Determines the number of points in the scene to be used as reference points.
- **Translation clustering threshold**: Maximum difference in translation be-  
tween two poses for them to belong to the same cluster.
• Rotation clustering threshold: Maximum difference in rotation between two poses for them to belong to the same cluster.

• Number of poses to return.

3.2.3 Extensions of the Point Pair Feature approach

The implementation of the PPF approach was used as a base to implement the extensions described in section 2.3. The classes were modified and new functions were added. Algorithm 3 shows a summary of the extended PPF approach, where & and | are the logical and, and or, respectively. The algorithm can be compared with algorithm 2 which is the original PPF approach.

**Algorithm 3:** The Extended Point Pair Feature approach.

**Input:** Model and scene point cloud \((M, S)\), see figure 3.4

**Output:** Transformations \(T\), called poses, that aligns \(M\) to \(S\)

Preprocessing of point clouds, see section 3.2.1

Calculate the global model description

for every reference point in the scene, \(s_r\), do

- Create array with the size of total number of discretized PPF, \(b\)
- Set each element in \(b\) to a 32 bit int, see section 2.3.4

for every other point in the scene closer than \(\text{Diam}(M)\), \(s_i\), do

- Calculate the Point Pair Feature \(F(s_r, s_i)\) according to (2.1)
- Calculate which neighbors could be affected by noise with (2.9)
- Get model pairs with similar features from the hash table and from neighboring bins that are likely to be affected by noise, see section 2.3.3

for every model pair do

- Calculate \(\alpha\) according to (2.8)
- Let \(\alpha\) be a 32 bit int with the bit corresponding to \(\alpha = 1\)

if \(b[F(s_r, s_i)] \& \alpha = 0\) then

- Vote for local coordinate \((m_r, \alpha)\)
- \(b[F(s_r, s_i)] = b[F(s_r, s_i)] | \alpha\)

end

end

Calculate possible pose from optimal local coordinate

end

Cluster possible poses

Average poses in clusters

Postprocessing, see algorithm 4

return poses with highest score

At the end of the algorithm, before the poses were returned, the extended PPF approach includes some postprocessing steps. First, the model cloud was
rendered according to every clustered pose such that only the points that were seen from the camera were kept. Then, using the rendered cloud, a refinement with ICP was performed by using the implementation from PCL. A re-scoring was performed on the rendered cloud to give a better representation of how well the pose fitted the scene. At last, filtering was performed to reject poses that didn’t correspond to an object. The postprocessing is summarized in algorithm 4.

Algorithm 4: Postprocessing.

Input: Model and scene point cloud \((M, S)\), list with possible poses \(T\)
Output: List with possible poses \(T\)

for every possible pose do
  // Render model point cloud
  Transform model cloud according to pose
  for each point in model cloud do
    // Remove points not visible from the camera
    if point normal \cdot (camera position - point) > 0 then
      Keep point
    else
      Erase point
    end
  end
  Refine pose with ICP
  // Re-scoring
  Score = 0
  for each point in model cloud do
    Find closest point in scene cloud
    if distance between scene point and model point < \(d_{\text{dist}}\) then
      Score++
    end
  end
  // Filtering
  Classify all points in model as inlier, occluded, non-consistent depending on the distance in z-axis to the scene
  if occluded points > 90% or non-consistent points > 15% then
    Erase pose
  end
end
Sort poses by score
return poses

The parameters for the algorithm were the same as for the PPF approach, explained in section 3.2.2. In addition, a filtering threshold, for determining
whether a point was classified as inlier, occluded or non-consistent, were added.

### 3.2.4 Fast Global Registration

The Fast Global Registration, described in section 2.4, was implemented using an open source implementation by Zhou et al. available on Github [30]. For the implementation of the FPFH feature, an implementation done at SICK was used. However, since the model was a 3D object, where there are points on all sides, and the scene only show points on the object from one viewpoint, the code was slightly modified. When calculating the FPFH feature, only points on the same side as the query point on the model should be included as neighbors. Therefore, only points with a positive scalar product between the normals were included as neighbors. A summary of the algorithm is found in algorithm 5.

#### Algorithm 5: The Fast Global Registration.

**Input:** Model and scene point cloud \((M, S)\), see figure 3.4  
**Output:** Transformation \(T\) that aligns \(M\) to \(S\)

- Preprocessing of point clouds, see section 3.2.1  
- Compute FPFH features \(F(M)\) and \(F(S)\), see section 2.1.3  
- Create correspondence set, \(K\), by computing nearest neighbors between \(F(M)\) and \(F(S)\), see section 2.4.1  
- Create new \(K\) only with the correspondences that are mutually nearest  
- Create new \(K\) only with the correspondences that are compatible  

\[
k = 0, \quad T^k = I, \quad \mu = 0.5 \text{diam}(S)
\]

while not converged and \(k < \text{max iterations}\) do

\[
J_r = 0, \quad r = 0
\]

for \((m, s) \in K\) do

\[
\text{Compute } l_{s,m} \text{ according to (2.15)}
\]

Update \(J_r\) and \(r\) of (2.13)

end

Solve (2.17) and update \(T^k\) according to (2.16)

if \(k \mod 4 = 0\) and \(\mu > \delta^2\) then

\[
\mu = \mu/df
\]

end

end

return \(T^k\)

Several parameters in the algorithm could be adjusted to increase the performance. These were as follows:

- Sampling rate \(\tau_d\): Multiplied with the diameter of the model to determine \(d_{\text{dist}}\) according to (2.6). \(d_{\text{dist}}\) was used as the voxel size during subsampling of the point cloud.
• Sphere radius $r$: Radius used for selecting neighbors when calculating FPFH features. [30] recommend to use a sphere radius that is five times larger than the voxel size used for subsampling of the point cloud. Therefore $r$ was set to $5d_{dist}$.

• Quantization parameter $d$: The number of bins each feature was quantized in. PCL uses $d = 11$ as default in the implementation of the FPFH features [31]. This seems like a common value to use and was thus used here, giving a 33-dimensional feature vector.

• Tuple scale $τ$: Used to determine if correspondences are compatible.

• Division factor $df$: Determines how fast $µ$ decreases, and thereby how fast the robust penalty in (2.12) narrows.

• Distance threshold $δ$: Determines when to stop decreasing $µ$ and consequently when to stop the optimization.

• Max iterations: The maximum number of iterations to perform optimization.

3.2.5 Parameter tuning

To find the parameters for the algorithms that gave the best result, a grid search was performed for both of the algorithms and for all objects. All parameters mentioned above, where no value of the parameter is set, were varied during the search.

3.3 Evaluation

The output from the algorithms were the estimated transformations that should align the model to the scene. The transformation was rigid, consisting of a rotation part, $R$, and a translation part, $t$. The transformations were compared with the ground truth transformations to evaluate the performance. Both the difference in translation and rotation between the transformations needed to be small enough for the estimated transformation to be considered correct. Therefore, a translation error and a rotation error were defined. The translation error was defined as

$$ε_{trans} = \|t_{gt} - t_{est}\|,$$  \hspace{1cm} (3.2)

where $gt$ denotes the ground truth and $est$ denotes the estimated. The distance between two rotations is the angle of the difference rotation, given by the rotation matrix $R = R_{gt}R_{est}^*$, where $^*$ is the matrix transpose. The angle of the difference rotation can then be retrieved from the trace of $R$,

$$tr(R) = 1 + 2\cos θ.$$  \hspace{1cm} (3.3)
Thus the rotation error was given by

$$\varepsilon_{rot} = \arccos\left(\frac{tr(R) - 1}{2}\right).$$

(3.4)

Since some of the objects have symmetries, rotations were applied to the transformation such that all poses that were considered correct were covered and the smallest error from these poses was chosen. A transformation was then defined to be correct if

$$\varepsilon_{trans} < \tau_{trans} \quad \text{and} \quad \varepsilon_{rot} < \tau_{rot},$$

(3.5)

where \(\tau_{trans}\) and \(\tau_{rot}\) are two thresholds. From this, a recall was defined as

$$\text{recall} = \frac{\text{correct matches}}{\text{all matches}}.$$  

(3.6)

The Point Pair Feature approach returns a list with multiple transformations, while Fast Global Registration returns only one transformation. Due to this, two different measures were defined and used to evaluate the performance. The top-1 recall, which measures the performance on finding one object, and the total recall, which measures the performance on finding all objects in the scene.

### 3.3.1 Top-1 recall

To measure the top-1 recall the best transformation from each algorithm was used. Thus, from Fast Global Registration, this was the output and from the PPF approach, this was the transformation with the highest score.

However, since there could be multiple objects in the scene, the estimated transformation needed to be assigned to a ground truth transformation. A cost, according to algorithm 6, was therefore calculated between the estimated transformation and all ground truth transformations. The transformation with the minimum cost was then used as the actual ground truth.

Then, the translation error and the rotation error were calculated according to (3.2) and (3.4), and the estimated transformation was classified as a correct match or not according to (3.5). This was done for every scene in the data set and a recall was calculated according to (3.6).

### 3.3.2 Total recall

Since the output from the algorithms differs, the total recall was calculated in two different ways. As mentioned before, Fast Global Registration returns only one transformation. This was assigned to a ground truth transformation and classified as correct or not in the same way as when calculating the top-1 recall. Points in the scene that were close to the model, after the ground truth transformation was applied, were then removed, simulating that a robot removed the object in the bin. Then the algorithm was executed again. This way, one object was removed from the scene for every iteration and it was repeated until there were no
3.3 Evaluation

**Algorithm 6**: The cost between the ground truth transformation and the estimate transformation.

**Input**: Model cloud $M$, ground truth and estimated transformations $T_{gt}, T_{est}$

**Output**: Cost between the two transformations

1. $M_{gt} = T_{gt}M$, $M_{est} = T_{est}M$
2. $cost = 0$
3. for every point in $M_{est}$ do
   4. Find closest point in $M_{gt}$
   5. cost += distance to closest point
4. end
5. cost /= number of points in $M_{est}$

return Cost

more objects left in the scene. After doing this on all scenes in the data set the recall for finding all objects in the scene was calculated.

The PPF approach returns multiple transformations. By making sure that the number of transformations it returned was more than the number of objects in the scene, each ground truth transformation could be assigned to an estimated transformation. The cost between every ground truth transformation and every estimated transformation was calculated according to algorithm 6. The assignment problem could then be solved by using the Hungarian method, which minimizes the total cost to find an optimal assignment [32]. The cost between transformations was limited to a maximum to avoid that objects without any match got assigned to a transformation that would be a correct match for another object.

When the estimated transformations were assigned to ground truth transformations, the translation errors and the rotation errors were calculated according to (3.2) and (3.4). The estimated transformations were then classified as correct matches or not according to (3.5) and the total recall was calculated according to (3.6), when every scene image was processed.
The following chapter presents the results from the algorithms. 15 images from each data set were used for the parameter search as a tradeoff between speed and performance. The resulting parameters were then used for evaluation on the entire data set. The evaluation was performed as explained in section 3.3. As postprocessing, ICP can be performed on both algorithms to refine the estimated transformation. The algorithms were evaluated both with and without ICP.

4.1 The Point Pair Feature approach

The parameter search for the Point Pair Feature approach was performed in two steps. First, the parameters in the algorithm without the postprocessing steps were optimized. Second, the postprocessing steps, and the parameters used during postprocessing, were optimized with the parameters found in the first step.

The parameters, see section 3.2.2, that resulted in the highest recall in the initial parameter search are listed in table 4.1. The search showed that the parameters that affected the result the most, were the sampling rate, \( \tau_d \), and the scene reference point sampling interval. Smaller \( \tau_d \) and smaller scene reference point sampling interval, usually resulted in a higher recall. However, it also resulted in increased execution time. Therefore, these values were chosen as a tradeoff between speed and accuracy.

The number of poses to return affected the total recall. Sometimes, two poses in the result are approximately the same, that is approximating the pose of the same object. Therefore, returning more poses than the maximum number of objects in the scene, resulted in a higher total recall. Thus, to be sure that it is possible to find all objects in the scene, the number of poses to return was chosen to be 1.5 times the maximum number of objects in the scene.
The results from the optimization of the postprocessing steps, see section 3.2.3, are listed in table 4.2. The effect of the postprocessing steps were evaluated both when refinement with ICP was included and not. As can be seen, the postprocessing steps sometimes increased the performance and sometimes not.

<table>
<thead>
<tr>
<th>Postprocessing parameters</th>
<th>Conrod</th>
<th>Brake disc</th>
<th>Pipe</th>
<th>Crankshaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICP = False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-score poses</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Filter poses</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Filter threshold</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td>ICP = True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-score poses</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Filter poses</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Filter threshold</td>
<td>-</td>
<td>15</td>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.2: The postprocessing parameters used for evaluation of the PPF approach. True means the postprocessing step should be included, and False that it should not, to achieve the highest recall.

The pose estimation was first performed with the parameters listed in table 4.1 and 4.2 with ICP equal to false on all data sets. Figure 4.1 shows histograms over the translation and rotation error, described in section 3.3, for all objects where \( \varepsilon_{\text{trans}} < 100 \text{ mm} \) and \( \varepsilon_{\text{rot}} < 45^\circ \). Figure 4.2 shows the total recall and the top-1 recall for different values of the translation error threshold, \( \tau_{\text{trans}} \), and the rotation error threshold, \( \tau_{\text{rot}} \). In the left figures, \( \tau_{\text{rot}} \) was fixed to \( 30^\circ \) and \( \tau_{\text{trans}} \) was varied between 0 mm and 100 mm. In the right figures, \( \tau_{\text{trans}} \) was fixed to 40 mm and \( \tau_{\text{rot}} \) was varied between \( 0^\circ \) and \( 45^\circ \).

Then, the pose estimation was performed with the same parameters except with ICP equal to true. Figure 4.3 and figure 4.4 show the histograms over the translation error and rotation error, and the total and top-1 recall.
4.1 The Point Pair Feature approach

**Figure 4.1:** Histogram over the translation error, $\varepsilon_{\text{trans}}$, and rotation error, $\varepsilon_{\text{rot}}$, for all objects in the data sets for the PPF approach without ICP.

**Figure 4.2:** Total recall and top-1 recall for the PPF approach without ICP. The threshold for the translation error, $\tau_{\text{trans}}$, are varied while keeping the threshold for the rotation error, $\tau_{\text{rot}}$, fixed, and vice versa.
Figure 4.3: Histogram over the translation error, $\varepsilon_{\text{trans}}$, and rotation error, $\varepsilon_{\text{rot}}$, for all objects in the data sets for the PPF approach with ICP.

Figure 4.4: Total recall and top-1 recall for the PPF approach with ICP. The threshold for the translation error, $\tau_{\text{trans}}$, are varied while keeping the threshold for the rotation error, $\tau_{\text{rot}}$, fixed, and vice versa.
Figure 4.5 and figure 4.6 show examples of the results for the different objects when using the parameters listed in table 4.1 and 4.2 without and with ICP respectively. The scene and the models are plotted before the subsampling for easier visualization. The models were transformed with the estimated transformations and shown in green.

Figure 4.5: Examples of results with the Point Pair Feature approach without ICP.
By comparing figure 4.5 and 4.6, it can be seen that with ICP, for example, the conrod in the lower right corner in (b), was turned correctly, while the crankshaft to the left in (h) was turned wrong. Some of the conrods and the pipes also got a tighter alignment.

*Figure 4.6: Examples of results with the Point Pair Feature approach with ICP.*
4.2 Fast Global Registration

The parameters, see section 3.2.4, that resulted in the highest recall in the parameter search are listed in table 4.3. The search showed that the parameters that affected the result were mainly the sampling rate and the tuple scale. A smaller sampling rate, yielding denser point clouds, increased the performance. The tuple scale, i.e. the threshold for considering correspondences to be compatible, was dependent on the value of the sampling rate. The distance threshold and the maximum number of iterations did not have a significant effect on the result.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Conrod</th>
<th>Brake disc</th>
<th>Pipe</th>
<th>Crankshaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling rate, $\tau_d$</td>
<td>0.04</td>
<td>0.03</td>
<td>0.04</td>
<td>0.01</td>
</tr>
<tr>
<td>Tuple scale, $\tau$</td>
<td>0.6</td>
<td>0.75</td>
<td>0.6</td>
<td>0.9</td>
</tr>
<tr>
<td>Division factor, $df$</td>
<td>1.2</td>
<td>2</td>
<td>2</td>
<td>1.2</td>
</tr>
<tr>
<td>Distance threshold, $\delta$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Max iterations</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

*Table 4.3: The parameters used for evaluation of Fast Global Registration.*

The registration was performed with the parameters listed in table 4.3 on all data sets. Figure 4.7 shows a histogram over the translation error and a histogram over the rotation error, described in section 3.3, for all objects where $\varepsilon_{\text{trans}} < 100$ mm and $\varepsilon_{\text{rot}} < 45^\circ$. Figure 4.8 shows the total recall and the top-1 recall for the data sets for different values of the translation error threshold, $\tau_{\text{trans}}$, and the rotation error threshold, $\tau_{\text{rot}}$. In the left figures, $\tau_{\text{rot}}$ was fixed to $30^\circ$ and $\tau_{\text{trans}}$ was varied between 0 mm and 100 mm. In the right figures, $\tau_{\text{trans}}$ was fixed to 40 mm and $\tau_{\text{rot}}$ was varied between 0$^\circ$ and 45$^\circ$.

Figure 4.9 and figure 4.10 show the histograms over the translation error and rotation error, and the total recall and top-1 recall, when ICP was performed as a last step.

Figure 4.11 and figure 4.12 show examples of the results for the different objects when using the parameters listed in table 4.3 without and with ICP respectively. The scene and the models are plotted before the subsampling for easier visualization. The models were transformed with the estimated transformations and shown in green. To get estimated transformations for all objects in the scene, the registration was performed multiple times for each scene in the same way as when calculating the total recall. As described in section 3.3.2, the points close to the model, transformed with the ground truth transformation assigned to the estimated transformation, were removed in each iteration until no objects were left.

By comparing figure 4.11 and 4.12, it can be seen that with ICP, for example, the conrod in (a) and the crankshaft in (g), got a tighter alignment. ICP also managed to turn the lowest brake disc in (c), the two brake discs in (d), the right pipe in (e), and two pipes on the right side in (f), correctly.
Figure 4.7: Histogram over the translation error, $\varepsilon_{\text{trans}}$, and rotation error, $\varepsilon_{\text{rot}}$, for all objects in the data sets for Fast global Registration without ICP.

Figure 4.8: Total recall and top-1 recall for Fast Global Registration without ICP. The threshold for the translation error, $\tau_{\text{trans}}$, are varied while keeping the threshold for the rotation error, $\tau_{\text{rot}}$, fixed, and vice versa.
4.2 Fast Global Registration

Figure 4.9: Histogram over the translation error, $\varepsilon_{\text{trans}}$, and rotation error, $\varepsilon_{\text{rot}}$, for all objects in the data sets for Fast Global Registration with ICP.

Figure 4.10: Total recall and top-1 recall for Fast Global Registration with ICP. The threshold for the translation error, $\tau_{\text{trans}}$, are varied while keeping the threshold for the rotation error, $\tau_{\text{rot}}$, fixed, and vice versa.
Figure 4.11: Examples of results with Fast Global Registration without ICP.
Figure 4.12: Examples of results with Fast Global Registration with ICP.
5 Discussion

In the following sections, the algorithms and the results are discussed.

5.1 Results

The algorithm that showed the best performance was the Point Pair Feature approach, providing robust localization of the objects. Fast global Registration, on the other hand, performed worse than expected.

5.1.1 The Point Pair Feature Approach

The PPF approach provided a robust localization of the objects as can be seen in figure 4.1 and 4.2. The algorithm either estimated a pose close to an object or did not find an object at all. When small translation and rotation errors were allowed, the top-1 recall was maximum, i.e. equal to 1, for all objects, except for the crankshaft. This could indicate that the crankshaft was harder to find because of the the many similar details on the object. The total recall was slightly lower than the top-1 recall for all objects. This could be explained by the fact that the algorithm had a hard time finding objects that were partly hidden by other objects, as seen in figure 4.5b. However, this would not be a problem in a real-time application, since when an object is found, it is removed and a new image of the scene is created.

Figure 4.5e and 4.5f shows that the algorithm had a hard time finding pipes that were standing up. An explanation could be that the only points visible on the object was a flat surface, where the features would be the same as the features on the floor. This also explains the low total recall in comparison with the high top-1 recall in figure 4.2.
By comparing figure 4.1 and 4.2 with figure 4.3 and 4.4, it can be seen that ICP made a tighter alignment for the conrod and the crankshaft. However, for the pipe, the alignment was tighter for some objects, while yielding a larger error for other objects. In addition, for the brake disc, the results were worse when performing ICP compared to the results without ICP. The reason for this might be that the parameters of ICP needed to be adjusted.

5.1.2 Fast Global Registration

Fast Global Registration did not achieve robust localization of the objects as can be seen in figure 4.7 and 4.8. Even though a set of parameters were evaluated to find a good solution, both the total recall and the top-1 recall were low. However, the algorithm is non-deterministic in the creation of the correspondence set. During the parameter search, it was initialized with a random seed, making the algorithm deterministic, to be able to compare the parameters. Thereby, other parameters could give a better result by allowing the algorithm to be non-deterministic.

When ICP was added as a last step, the performance was slightly better, see figure 4.9, 4.10 and 4.12. However, the increase in performance was small, which could be explained by the fact that when the initial transform to ICP is far away from the object, ICP has a hard time refining the solution. When an object was found, the translation error was usually smaller than 60 mm, which can be decreased by adding ICP as a last step. However, for most of the estimated transformations, the translation error was larger than 100 mm, for which a robot would have a hard time picking up the object.

5.2 Method

In the following sections, the two algorithms and their advantages and disadvantages are discussed.

5.2.1 The Point Pair Feature Approach

Overall, the Point Pair Feature approach was a good solution for bin-picking. It returned a list with poses, giving a robot several chances to pick an object. In addition, the results indicated that it can handle different types of objects, yet some better than others.

The largest disadvantage with the algorithm was the execution time. The more points the scene contained, i.e. the smaller $d_{\text{dist}}$, and the smaller scene reference point sampling interval that was used, the slower the matching, yet also the better the result. Therefore, these two parameters must be chosen as a tradeoff between speed and accuracy. On the other hand, it would be possible to speed up the implementation. This could, for example, be done by implementing it on a GPU or divide the scene into multiple sections and perform pose estimation on several sections at the same time. When using the parameters listed in table 4.1
and the computer specified in section 3.2, the matching took between 10 and 30 seconds.

The extensions that were implemented appeared to increase the performance of the algorithm. The largest effect was given by avoiding multiple superfluous votes in the accumulator array, explained in section 2.3.4. This extension resulted in that the algorithm could handle the floor of the bin. Before it was included, many of the estimated transformations were on the floor.

The spreading of the point pairs to decrease the effect of the noise, explained in section 2.3.3, was not implemented according to the documentation because of issues with PCL. Instead of spreading the point pairs in the offline phase, point pairs were retrieved from neighboring bins in the online phase. This resulted in a slight increase in the performance, however, it also increased the execution time, giving an execution time about 1.5 times the execution time when spreading of point pairs were excluded. If this instead had been implemented as in the documentation, the execution time would probably not have increased as much, since it partly could be done in the offline phase. Worth to consider is, however, whether or not the slight increase in performance is worth the increase in execution time.

The postprocessing steps did not appear to have a significant effect on the result. For some objects, as can be seen in table 4.2, they even lead to a worse result. Because of time constraint, the second filtering step, where the silhouette of the object and the edges in the scene were compared, explained in section 2.3.5, was never implemented. This step would probably have removed incorrect poses that were far away from the objects. However, since it is likely that these poses would have a lower score, it would probably not affect the top-1 recall.

### 5.2.2 Fast Global Registration

One explanation of the bad result of Fast Global Registration, could be that the algorithm is not suited for this kind of problem. The point clouds and the normals to the points, were created in different ways. The point cloud of the scene was created with a synthetic range camera and the normals were then estimated by using the neighborhood of the points. The point cloud of the model, on the other hand, was created from the CAD model and the normals were estimated from the faces in the triangular mesh. This may lead to a deviation between the point clouds. Since all points in a neighborhood are used to calculate the FPFH feature, this could result in a deviation between the FPFH features as well. A deviation in the features, could in turn yield a bad correspondence set. Since the optimization heavily relies on a good correspondence set, this would lead to bad optimization. In many cases, the correspondence set was small, containing only a couple of correspondence pairs, which could indicate that the correspondence set was bad.

Another reason for the bad result could be that the algorithm had a hard time finding a solution when there were multiple instances of the same object in the scene. This would lead to a lot of similar details in the scene, creating many similar features. The examples in figure 4.11 indicates that when there was only
one object in the scene, the estimated transform was close to the object. Yet, when
there were many objects in the scene, the results were worse.

One advantage with fast global registration was the low execution time. Reg-
istration of one scene image took less than 2 seconds when using the computer
specified in section 3.2. A disadvantage was however, that it only returned one
transformation. If this is incorrect and a robot does not succeed with picking an
object, the same scene will be processed again. This may yield the same incor-
rect transformation. However, since the algorithm was non-deterministic during
the creation of the correspondence set, a new transformation could be estimated.
Although, a few experiments showed that the returned transformation was ap-
proximately the same when processing the same scene multiple times.
Conclusion

In this thesis algorithms for 6D pose estimation for bin-picking were investigated and evaluated to find a solution that achieves sufficiently robust localization of objects. Two algorithms were implemented for this, the Point Pair Feature approach and Fast Global Registration. The best result was provided by the Point Pair Feature approach. As a conclusion the questions in section 1.2 are answered.

- **Can the Point Pair Feature approach [1] by Vidal et al. or the Fast Global Registration [2] by Zhou et al., described in chapter 2, achieve sufficiently robust localization of objects, defined by a CAD model, in point cloud data so that it can be used in the bin-picking scenario?**

  The Point Pair Feature approach provided a robust localization of objects that could be used for bin-picking. The disadvantage with the algorithm was the execution time. However, if the execution time could be decreased it would be a good option in a real-time application.

  Fast global Registration did not provide sufficiently robust localization of objects for it to be used for bin-picking. The only advantage with the algorithm was the low execution time.

- **Which one of the algorithms is preferred for the problem?**

  The preferred algorithm is thereby the Point Pair Feature approach.

- **Can the algorithms handle objects of different types or what structure of an object is needed for the solution to be robust?**

  The Point Pair Feature approach appears to be able to handle objects of different types. It had a hard time finding the pipe when it was standing up, which indicates that it is not the best solution for objects with flat surfaces and weak texture. The results for the crankshaft were also slightly worse.
than for the other objects. The crankshaft was the object with most texture and similar details over the object. Thus, many similar details on the object might be a limitation for the algorithm as well.

Fast Global Registration provided slightly higher top-1 recall for the conrod than for the other objects. However, none of the objects had a structure that resulted in a robust localization with Fast global Registration.

### 6.1 Future work

The algorithms were in this thesis only evaluated on synthetic data. As a next step, it would be interesting to test it on real data to see how the result would be affected. Also to see if the Point Pair Feature approach still achieves sufficiently robust results.

Two improvements could be done in the Point Pair Feature approach. The first one is to modify the spreading of point pairs to match the documentation, i.e. spread the point pairs in the offline phase to decrease the execution time. The second, is to implement the second filtering step in the postprocessing to remove bad matches.

In addition, ways to decrease the execution time could be tested to make the approach more suitable for a real-time application. One example could be to implement it on a GPU or with multithreading. Another example could be to divide the scene images in multiple sections and perform the pose estimation on several sections at the same time. A third example could be to follow the analysis by Abbeloos and Goedemé [33], using an additional subsampling procedure to select the scene reference points. Their assumption is that the highest objects will be the easiest to detect and grasp for a robot. Therefore, the range image is low pass filtered and iteratively the highest point is selected as scene reference point and the local neighborhood of the point is excluded.


[22] “FreeCAD: Your Own 3D Parametric Modeler.”
https://www.freecadweb.org/.


[27] “Pcl::PFPEstimation class uses PFH features instead of PPF features · Issue #1131 · PointCloudLibrary/pcl.” https://github.com/PointCloudLibrary/pcl/issues/1131.


https://github.com/IntelVCL/FastGlobalRegistration.

