

# Unsupervised identification of the body parts of an unknown articulated object

Anna M. Maureder



## MASTERARBEIT

eingereicht am  
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Februar 2017

© Copyright 2017 Anna M. Maureder

This work is published under the conditions of the Creative Commons License *Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0)—see <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

# Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, February 28, 2017

Anna M. Maureder

# Contents

<b>Declaration</b>	<b>iii</b>
<b>Abstract</b>	<b>vi</b>
<b>Kurzfassung</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>2</b>
2.1 Marker, User input . . . . .	2
2.2 Non-rigid Registration . . . . .	2
2.2.1 EM-algorithm . . . . .	2
2.2.2 LRP . . . . .	2
2.2.3 Symmetrization . . . . .	2
<b>3 My contribution</b>	<b>3</b>
3.1 Goal and approach . . . . .	3
3.2 Assumptions . . . . .	3
3.3 Challenges/restrictions . . . . .	4
3.4 Approach . . . . .	4
3.4.1 Divide and conquer . . . . .	4
3.4.2 Removing outliers . . . . .	4
3.4.3 Subdividing into clusters . . . . .	4
3.4.4 Merging neighboring clusters to rigid parts . . . . .	6
3.4.5 Joint/skeleton estimation . . . . .	7
3.4.6 Implementation . . . . .	7
3.4.7 Implementation Steps . . . . .	8
3.4.8 Results . . . . .	9
3.4.9 Possible Improvements . . . . .	9
3.5 LRP as initial alignment . . . . .	9
3.5.1 Overview . . . . .	9
3.5.2 Algorithm . . . . .	10
3.5.3 Steps . . . . .	10
3.6 Other approaches . . . . .	11

3.6.1	Points-to-Ellipse fitting . . . . .	11
3.6.2	Algorithm . . . . .	11
3.6.3	Steps . . . . .	11
3.6.4	Results . . . . .	12
3.6.5	Reusing detected shapes . . . . .	12
3.7	General Results . . . . .	12
3.8	Improvements . . . . .	12
3.9	Future work . . . . .	12
<b>4</b>	<b>Conclusion</b>	<b>13</b>
4.1	Future work . . . . .	13
	<b>References</b>	<b>14</b>
	Literature . . . . .	14

# Abstract

The proposed work describes a method for pose estimation of articulated objects without any prior knowledge of their body parts. Most existing pose estimation methods take advantage of trackers, and user inputs to estimate the joint positions. However, a completely unsupervised method constitutes an enormous potential but also a great challenge. A main solution to that is proposed by template matching associated with the non-rigid registration of meshes, which requires two poses of the same object and applies the Expectation-Maximization algorithm to segment the object into its rigid parts. This is done iteratively by assigning mesh points on body parts and finding transformations that perfectly match those body parts on both meshes. Based on this approach, a segmentation method is developed to obtain the rigid parts of an object and consequently estimate its joints.

# Kurzfassung

An dieser Stelle steht eine Zusammenfassung der Arbeit in Deutsch.

## Chapter 1

# Introduction



## Chapter 2

# Related Work

Here comes the State-of the Art. An overview of related methods to non-rigid registration for detecting rigid body parts of an articulated object are mentioned by Chang [3] and Tam [7] which mainly use the ICP (iterative closest point) and the PCA (Principal component analysis) to find corresponding body parts. This paper is based on the Correlated Correspondance algorithm [2] [1] and Symmetrization [6]. A following work to [3] is [4]. Other methods include temporal coherence, markers and user inputs. Another method is from [5].

### 2.1 Marker, User input

Here are methods that take advantage of user inputs and markers and are therefore supervised methods. I will focus my work on non-rigid registration methods, which are unsupervised.

### 2.2 Non-rigid Registration

Here are methods that focus on non-rigid registration to recover the rigid part of an object.

#### 2.2.1 EM-algorithm

#### 2.2.2 LRP

#### 2.2.3 Symmetrization

## Chapter 3

# My contribution

This chapter focuses on the implementation of a new segmentation approach by taking the existing methods as reference (see chapter 2).

### 3.1 Goal and approach

The goal is to segment an articulated mesh  $M$  into its unknown number  $n$  of rigid parts  $P = \{p_0, \dots, p_n\}$  and extract the joints  $J = \{j_0, \dots, j_{n-1}\}$  linking those parts in form of a skeleton structure. In general, this is done by non-rigid registration of the points clouds  $S_0$  and  $D_0$  of an object in two different poses.  $S_0$  is thereby used as a *template* to be registered with  $D_0$ . The main task is to determine a part assignment  $p_i$  and the corresponding transformation  $t_i$  for all points of the *template* that aligns them with all points of  $D_0$ . Basically, a divide and conquer approach (see section 3.4) is implemented to recursively subdivide  $S_0$  and  $D_0$  into clusters trying to be matched (reduce the computation steps of the correlated correspondence algorithm [1]). Furthermore, the LRP approach [5] is employed as an initial registration step to align the two poses of the object.

### 3.2 Assumptions

The input mesh  $M$  is assumed to solely consist of rigid parts that can not be deformed or stretched (e.g. rigid parts of a human). Those parts are linked by joints and no matter what kind of pose is adopted by the articulated object, the geodesic distances between mesh points always stay the same (see Figure SKIZZE). Thereby, it is taken advantage of the knowledge that points located on a rigid part  $p_i$  have the same transformation  $t_i$ . Furthermore, it is assumed that the two poses  $S_0$  and  $D_0$  of  $M$  are oriented the same direction.

### 3.3 Challenges/restrictions

There are many challenges regarding the non-rigid registration of point clouds in 2D, as well as in 3D. First off, the input data can be noisy by means of points not belonging to the object. Furthermore, the approach is computationally expensive and time-consuming, as the corresponding body parts of two meshes need to be detected iteratively. Additionally, the inevitable difficulty of finding the global optimum, related to ambiguous body parts, has to be overcome.

### 3.4 Approach

#### 3.4.1 Divide and conquer

The algorithm starts with two sets of point clouds  $S_0$  and  $D_0$  of an object  $M$  in different poses (3.3). The two point clouds are iteratively subdivided into point clusters  $C = \{c_0, \dots, c_m\}$  through their centroids. In each iteration step two related clusters of  $S_0$  and  $D_0$  are matched by applying the ICP (iterative closest point) resulting in a matching error  $e$ . In case of  $e < \tau$ , two clusters are assumed to match. Otherwise, the algorithm is applied recursively and the clusters are again subdivided into further clusters. The algorithm terminates if all resulting clusters of  $S_0$  can be matched to all clusters of  $D_0$ . All neighboring clusters are then checked to be merged, in case of having divided a rigid part. After that step the remaining clusters are assigned to a rigid part  $part_i$ .

#### 3.4.2 Removing outliers

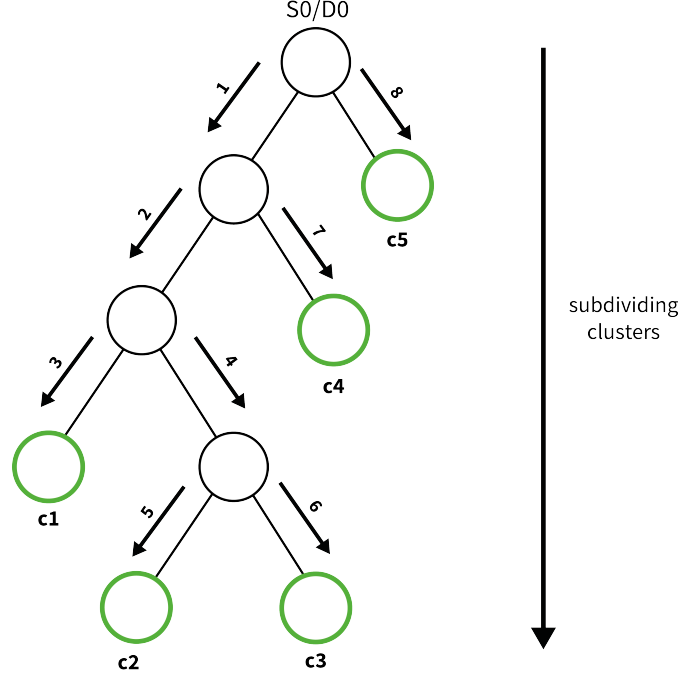
As a first step the outliers of the two point clouds  $S_0$  and  $D_0$  are removed. This is done, by finding clusters and just keeping the biggest one, assuming it is the main object

#### 3.4.3 Subdividing into clusters

As a next step, the two point clouds  $S_0$  and  $D_0$  are assigned to the initial clusters  $c_0$ . If the matching between them does not succeed, they are subdivided into two clusters and the whole procedure is repeated recursively until all sub clusters  $C = \{c_1, \dots, c_m\}$  of  $S_0$  can be matched to the clusters of  $D_0$ .  $S_0$  and  $D_0$  are initially divided into a left and right cluster.

#### Divider position

To determine where to divide a cluster into two other clusters, PCA is applied on the clusters is applied. The resulting dividers  $d_0$  are realised by



**Figure 3.1:** Subdividing of  $S_0$  and  $D_0$  into matching clusters by the depth-first approach

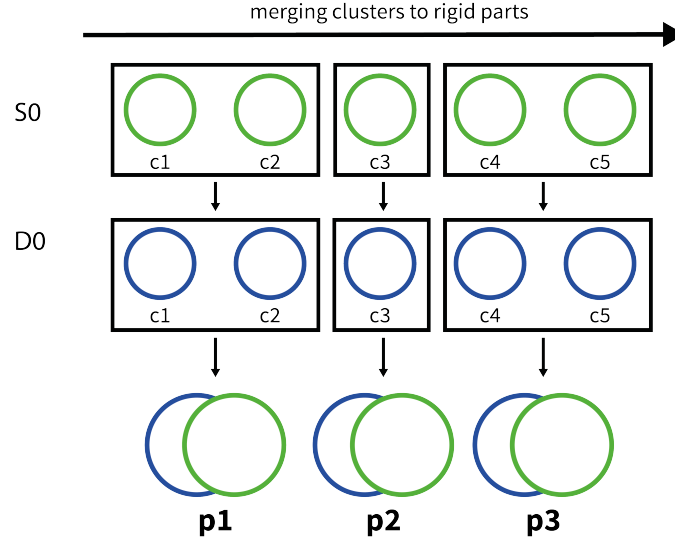
computing the principal axes  $p_0$  through the centroids  $ct_0$  and taking the perpendicular secondary axes  $s_S$  and  $s_D$  through the centroids.

#### Cluster tree

The subdividing of the clusters  $c_0$  is realised by a depth-first approach in a binary tree. Consequently,  $S_0$  and  $D_0$  are subdivided from the left to the right. A node  $N$  of the tree contains two related clusters  $c_i$  of  $S_0$  and  $D_0$  and in case of subdividing it, a Node *left* and *right*, containing the subdivided clusters  $c_{i+1}$  and  $c_{i+2}$ . If two associated clusters in a Node  $n_i$  match, no further subdividing is performed. The resulting leaves of the tree are stored as matching clusters  $C = \{c_1, \dots, c_m\}$  3.1. By applying the depth-first approach, the neighboring clusters in the list are also neighbouring clusters in the object  $S_0$  or  $D_0$ . As a result, the skeleton structure from an object can be extracted, where parts are connected by joints.

#### Declaring the matching condition

By applying the ICP and the nearest neighbour approach, usually a certain matching error is computed between  $P = \{p_0, \dots, p_n\}$  and its associated points  $Q = \{q_0, \dots, q_n\}$ . To declare when an object is matching, it is important



**Figure 3.2:** Detecting rigid parts of S0 and D0 by iteratively merging neighboring clusters of S0 and matching them with the merged clusters of D0.

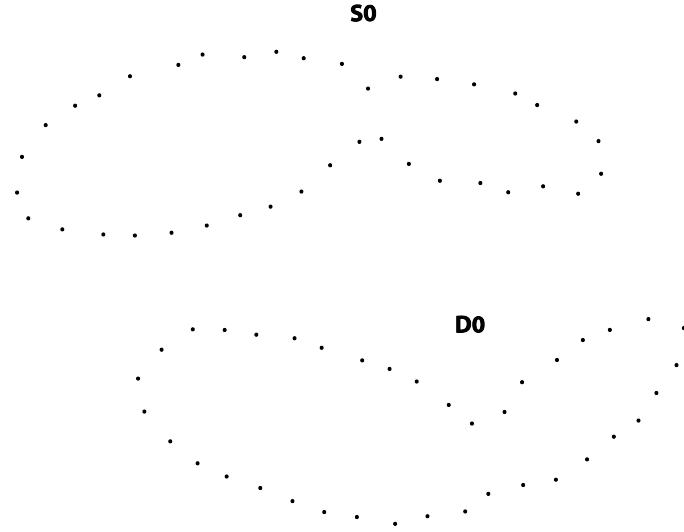
to find the right maximum matching error  $\tau$ . If it is too high, two clusters are not easy to be matched, which will result in more clusters. If the value is too low, clusters are more likely to be matched and there won't be enough subdividing. The matching threshold is compared to error per point

$$e_{avg} = \frac{\sum_{i=0}^m \|p_i - q_i\|^2}{|P|} \quad (3.1)$$

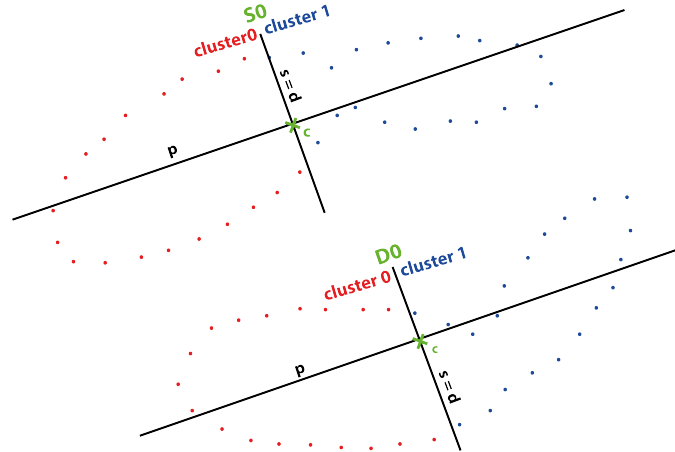
which is the average error of a point contributing to the total error. By using the average error instead of the total error, region growing is enabled as the matching error is independent of the amount of points.

#### 3.4.4 Merging neighboring clusters to rigid parts

As a next step, neighboring clusters from the matchedList are iteratively merged and checked for another match. This is done to reduce the found clusters to the rigid parts of the object. Again, the merge of one cluster is done until no further merge for neighboring parts can be done. Subsequently, the cluster is assumed to be a rigid part and saved in a new list. The resulting rigid parts in the list are again neighboring (see figure 3.2).



**Figure 3.3:** Taking a mesh  $M$  in two different poses  $S_0$  and  $D_0$  as input.

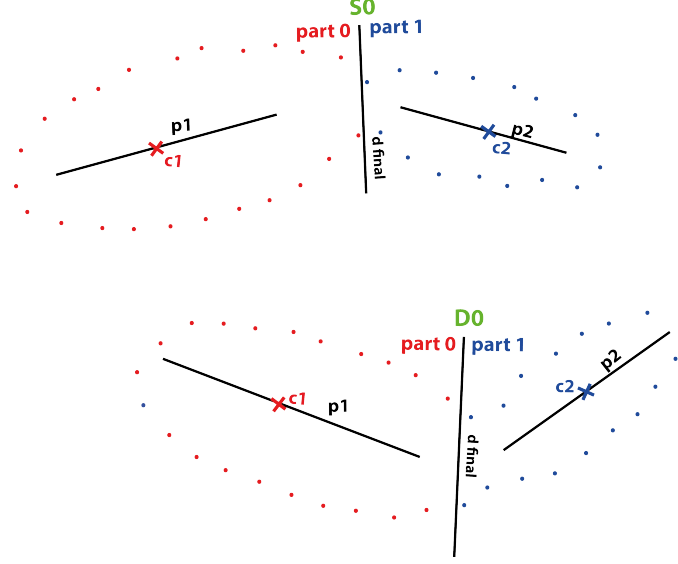


**Figure 3.4:** Dividing  $S_0$  and  $D_0$  into clusters by the divider  $d$  to match them with ICP.

### 3.4.5 Joint/skeleton estimation

### 3.4.6 Implementation

using of a binary tree to recursively segment clusters into smaller clusters, until they match. To be continued until all leaves of the tree can be matched with other clusters.



**Figure 3.5:** Assigning of the rigid parts  $P = (part_0, part_1)$  after termination of segmentation process.

### 3.4.7 Implementation Steps

1. The centroids  $c_S$  and  $c_D$  of  $S_0$  and  $D_0$  are computed.
2. The principal axis  $p_S$  and  $p_D$  are computed through  $c_S$  and  $c_D$  in order to orient the point clouds horizontally around their centroids.
3. The secondary axis  $s_S$  and  $s_D$  perpendicular to  $p_S$  and  $p_D$  through  $c_S$  and  $c_D$  are computed.
4. The dividers  $d_S$  and  $d_T$  to segment  $S_0$  and  $D_0$  into its assumed two rigid parts are initialized with the secondary axis  $s_S$  and  $s_D$ .
5. The points  $P_{0...N}$  of  $S_0$  are either allocated to  $S_{left}$  or  $S_{right}$  depending on its position to  $d_S$ . The same procedure is done with all points of  $D_0$ .
6. ICP is computed between the rigid parts  $S_{left}$  and  $D_{left}$  as well as  $S_{right}$  and  $D_{right}$ .
7. An error distance  $e_{left}$  and  $e_{right}$  is obtained. The part with the most error per point is assumed to be not rigid which gives back an indicator where to divide  $S_0$  and  $D_0$ .
8. The dividers  $d_S$  and  $d_D$  are shifted to the direction of the highest error. To be continued from step 5 until the total error  $e_{total}$  doesn't get smaller.

### 3.4.8 Results

Results from easy examples. Not working for human, as by dividing of one cluster, breaking down into single clusters (see Figure X). Another approach, e.g. using LRP as an initial alignment to then recursively segment the clusters linked to the LRP. Clusters not matching, as they don't have the same number of points, each point can only have one neighboring point. Or dividing clusters that they all have the same amount of points. In case of a more complex object where one rigid part can be linked with a high number of rigid parts (upper body of a human) another approach has to be found, as the skeleton structure is different than from a chain. (see Figure 3.5).

### 3.4.9 Possible Improvements

Same amount of points of associated clusters

Check clusters after subdividing

After subdividing, only two clusters should come up.

LRP as initial alignment

As only objects with rigid parts arranged like a chain are possible with this approach, another improvement/algorithm has to be pursued (see section 3.5).

## 3.5 LRP as initial alignment

Instead of cutting the object initially in half, as an initial step the largest rigid part is found and recursively from there all other linked parts can be detected.

### 3.5.1 Overview

As an initial step, the LRP algorithm tries to find the most reliable correspondences, the so-called largest rigid part (LRP), subsequently all other parts are detected that are linked to the LRP. The initial alignment stage tries to find sparse correspondences between two point clouds by applying a single rigid transformation to detect the largest subsets of points in two point clouds. Starting from the LRP all other parts are detected recursively.

### 3.5.2 Algorithm

Finding the LRP

The algorithm also takes two point clouds  $S_0$  and  $T_0$  of the same object in different configurations as input. The goal is to find a single rigid transfor-



mation  $T_{init}$  for all points of  $S_0$  to get potential corresponding points  $C_0 = \{(s_i, t_j)\}$  in  $T_0$ . For that, local descriptors of  $S_0$  and  $T_0$  are computed. The requirement for a sparse correspondance between two points  $s_i$  and  $t_j$  is that they are *reciprocal*, which means that the Euclidean distance  $d(s_i, t_j)$  between them is the smallest in both directions. Some of the sparse correspondances are asumed to be wrong. Therefore, RANSAC is used on the sparse correspondances  $C_0$  to estimate a rigid alignment that is supported by the largest number of points  $n$  from  $S_0$  and  $T_0$ . To assign the LRP in  $S_0$  and  $T_0$ , the biggest point clusters  $C_s$  and  $C_t$  of the overlapping area  $G_s = \{C_1, \dots, C_n\}$  and  $G_t = \{C_1, \dots, C_n\}$  are detected.

#### Part discovery

The remaining clusters from  $S_0$  and  $T_0$  that have not been registered yet are matched recursively by starting with clusters connected to already matched parts. First, all matched parts are excluded from the input point clouds  $G_{s(l+1)} = S_0 - C_{sl}$  and  $G_{t(l+1)} = T_0 - C_{tl}$  defining  $l$  as the number of already matched parts  $\{1, \dots, n\}$ ,  $C_{sl}$ . For that clusters are formed, using region taking into account that they are attached to already registered parts. The algorithm explained is applied until all body parts have been discovered.

#### 3.5.3 Steps

1. The centroids  $c_s$  and  $c_t$  of  $S_0$  and  $T_0$  are computed.
2. The principal axis  $p_s$  and  $p_t$  are computed through  $c_s$  and  $c_t$  in order to horizontally orient the objects around their centroids.
3. The ICP is conducted as a first guess to find a transformation  $T_{init}$  for all points from  $S_0$  that results in the highest number of corresponding points  $n$  in  $T_0$ , given the threshold  $T$ .
4.  $C_0$  contains the corresponding points from  $S_0$  and  $T_0$ , resulting from  $T_{init}(S_0)$ .
5. The RANSAC approach is applied on  $C_0$  to find a  $T_f$  that results in the highest number of corresponding points  $n$  between  $T_f(S_0)$  and  $T_0$ .
6. The LRP is assigned to  $C_s$  and  $C_t$  from the resulting point clusters  $G_s$  and  $G_t$ .
7. Starting from parts that are connected to the LRP, corresponding points  $C_i$  for unmatched points from  $S_0$  and  $T_0$  are seeked. The clusters are given as a input from Step 5.

### 3.6 Other approaches

#### 3.6.1 Points-to-Ellipse fitting

#### 3.6.2 Algorithm

This algorithm only requires one point cloud containing  $m$  points  $\{pt_0, \dots, pt_m\}$ . The basic idea is to segment the non-rigid object  $S_0$  into its rigid parts  $part_1$  and  $part_2$  by fitting ellipses to its rigid parts.  $S_0$  is divided perpendicular to its principal axis  $p_0$  into two assumed rigid parts  $S_{left}$  and  $S_{right}$ , initially defining the divider  $d$  with the secondary axis  $s_0$ . The points of  $S_{left}$  and  $S_{right}$  are verified to form an ellipse by using its formular

$$\frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} = 1 \quad (3.2)$$

Assuming to verify  $S_{left}$  forming an ellipse,  $r_1$  is half the length of the principal axis  $p_{left}$  of  $S_{left}$  through its centroid  $c_{left}$ . Furthermore,  $r_2$  is half the length of the secondary axis  $s_{left}$  of  $S_{left}$ . Thereby, the centroid  $c_{left}$  needs to be located in the origin (0,0). Now, to check whether a point  $pt_i$  of  $S_{left}$  is located on the ellipse, the formular is remodeled and its x values is applied.

$$\left(1 - \frac{x^2}{r_1^2}\right) \cdot r_2^2 = y^2 \quad (3.3)$$

The resulting y-value of the ellipse is compared to the points actual y-value. Given a certain threshold  $\tau$  a point either accounts to the number of total points lying on the ellipse  $n$ , or not.

$$n = \sum_{i=0}^m \begin{cases} 1 & \text{if } \|pt_i.y^2 - y^2\| < \tau \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

The algorithm is repeated by sliding  $d$  in the direction of the highest error  $e$ . To be continued until the total error  $e_{total} = e_{left} + e_{right}$  reaches its minimum.

#### 3.6.3 Steps

1. The centroid  $c_0$  of  $S_0$  is computed.
2. The principal axis  $p_0$  is computed through  $c_0$  and  $S_0$  horizontally oriented.
3. The secondary axis  $s_0$  perpendicular to  $p_0$  through  $c_0$  is computed.
4. The divider  $d$  is initialized with the secondary axis  $s_0$  to segment  $S_0$  into two assumed rigid parts .
5. The points of  $S_0$  are either allocated to  $S_{left}$  or  $S_{right}$  depending on its position to  $d_0$ .

6. The ellipse formular is applied on  $S_{left}$  and  $S_{right}$ .
7. An error  $e_{left}$  and  $e_{right}$  is obtained implying how many points of  $S_{left}$  and  $S_{right}$  form an ellipse.
8. The divider  $d$  is shifted to the direction of the highest error. To be continued from step 5 until the total error  $e_{total}$  doesn't get smaller.

#### 3.6.4 Results

#### 3.6.5 Reusing detected shapes

After termination of the algorithm, one point cloud can be segmented into its rigid parts  $P \{part_1, ..., part_n\}$ . Their variables like the ellipses' centroid  $c_i$  and radii  $r_1, r_2$  can be used to segment similar point clouds in different configurations. As the shapes to be matched are already known, e.g. how they are linked, finding the position to be segmented is a lot easier.

### 3.7 General Results

### 3.8 Improvements

### 3.9 Future work

The approaches implemented in 2D are then implemented in 3D using the PCL.

## Chapter 4

# Conclusion

To conclude, I proposed... The results are ...

### 4.1 Future work

Future developments can be done by ....

# References

## Literature

- [1] Dragomir Anguelov et al. “Recovering Articulated Object Models from 3D Range Data”. In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. UAI '04. Banff, Canada: AUAI Press, 2004, pp. 18–26. URL: <http://dl.acm.org/citation.cfm?id=1036843.1036846> (cit. on pp. 2, 3).
- [2] Dragomir Anguelov et al. “The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces”. In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 33–40. URL: <http://papers.nips.cc/paper/2601-the-correlated-correspondence-algorithm-for-unsupervised-registration-of-nonrigid-surfaces.pdf> (cit. on p. 2).
- [3] Will Chang and Matthias Zwicker. “Automatic Registration for Articulated Shapes”. *Computer Graphics Forum (Proceedings of SGP 2008)* 27.5 (2008), pp. 1459–1468 (cit. on p. 2).
- [4] Will Chang and Matthias Zwicker. “Range Scan Registration Using Reduced Deformable Models”. *Computer Graphics Forum (Proceedings of Eurographics 2009)*, to appear () (cit. on p. 2).
- [5] Hao Guo, Dehai Zhu, and Philippos Mordohai. “Correspondence estimation for non-rigid point clouds with automatic part discovery”. *The Visual Computer* 32.12 (2016), pp. 1511–1524 (cit. on pp. 2, 3).
- [6] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. “Symmetrization”. *ACM Trans. Graph.* 26.3 (July 2007). URL: <http://doi.acm.org/10.1145/1276377.1276456> (cit. on p. 2).
- [7] Gary KL Tam et al. “Registration of 3D point clouds and meshes: a survey from rigid to nonrigid”. *IEEE transactions on visualization and computer graphics* 19.7 (2013), pp. 1199–1217 (cit. on p. 2).