Unsupervised identification of the rigid part of an unknown articulated object

Anna M. Maureder

6th February, 2018

Abstract

This document is a simple template for a typical term or semester paper (lab/course report, "Übungsbericht", etc.) based on the HagenbergThesis LaTeX package.[1] The structure and chapter titles have been formulated to provide a good starting point for a typical *project report*. This document uses the custom class hgbreport which is based on LaTeX's standard report document class with `chapter` as the top structuring element. If you wish to write this report in German you should substitute the line

```
\documentclass[english]{hgbreport}
```

at the top of this document by

```
\documentclass[german]{hgbreport}.
```

Also, you may want to place the text of the individual chapters in separate files and include them using `\include{..}`.

Use the abstract to provide a short summary of the contents in the document.

[1] See https://github.com/Digital-Media/HagenbergThesis for the most current version. This repository also provides a good introduction and useful hints for authoring academic texts with LaTeX.

Contents

Chapter 1

Introduction

1.1 Initial idea

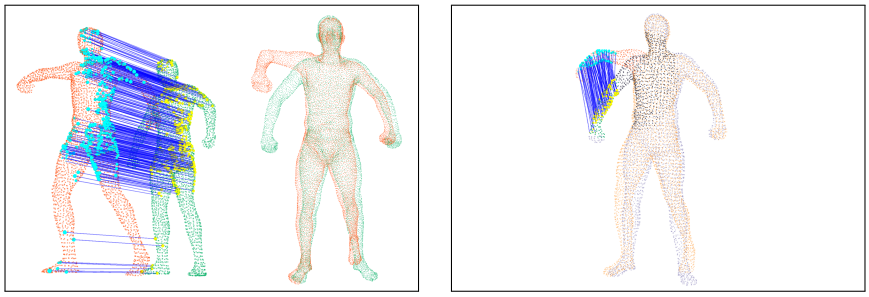
The initial project idea was to create a tool for 3D animation purposes using a small puppet to capture its poses in real-time. However, the idea addressed many different challenges, like 3D reconstruction, segementation, joint and skeleton estimation as well as creating a interface with a 3D animation program. As the implementation of these tasks would go beyond the scope of a thesis project, it was indispensible to break it down into its main areas. While doing researches on pose estimation, the issue of segmenting an articulated object into its rigid part frequently emerged and for this reason the thesis project will focus on this field.

1.2 Motivation

Pose and motion estimation of objects is an active field of research due to the growing digitalization of day-to-day processes. A vast majority of existing pose estimation methods take advantage of sensors and markers as an indicator for the joints of an object. Additionaly, the rigid parts and of an object and its joints are already known. However, unsupervised methods that are completly independent of user input and detect the pose of an unknown object, constitute a great challenge. Among those methods, the non-rigid registration is a well-known approach [7].

1.3 Related Work

A main approach for non-rigid registration is proposed by Anguelov [1] using the correlated correspondence algorithm [2]. A different approach is reached by Symmetrization [6] and the recursive detection of body parts by the LRP –



(a) (b)

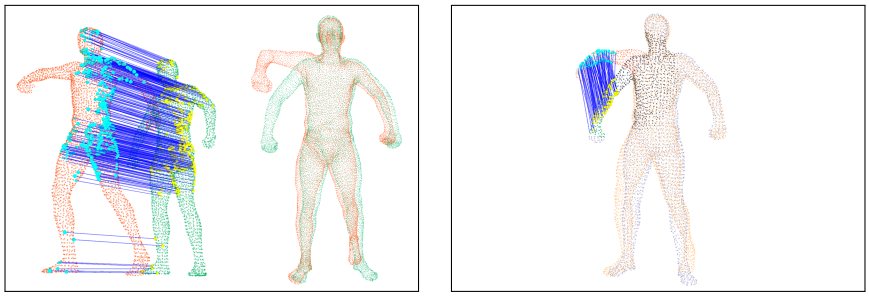
Figure 1.1: Taking a mesh in two different poses as input (a), removing noise of the input point clouds (b)

"largest rigid part" algorithm [5]. Based on [1] and [6], Chang et al developed a closely related approach [3] [4].

Mainly referenced are following approaches:

1.3.1 Correlated Correspondence

This algorithm takes a template Mesh and other Meshes in different configurations as input. It performs an probabilistic framework and Expectation-Maximization (EM) to iterate between finding a decomposition of template into rigid parts and finding the part in the other meshes.



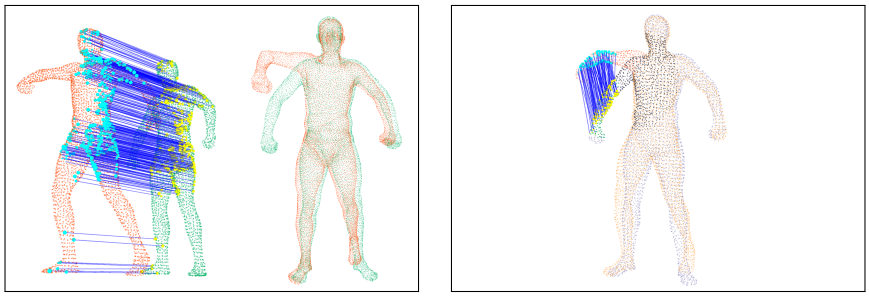
(a) (b)

Figure 1.2: Expectation-Maximization algorithm (a), to iteratively detect rigid parts (b)

1.3.2 LRP

The LRP algorithm discovers the articulated parts of two objects in different configurations by initially detecting the largest rigid part. This would be the

biggest point cluster by applying a single rigid transformation. To reach that, sparse correspondences in combination with RANSAC are implemented. From there, the linking parts are recursively detected by growing clusters from the LRP and reapplying the algorithm.



(a) (b)

Figure 1.3: Detecting the largest rigid part of an object (a), and align the object to recursively detect linking parts to the LRP (b)

1.3.3 Symmetrization

Chapter 2

My contribution

This chapter focuses on the implementation of a new segmentation approach by taking the existing methods as reference (see section 1.3). The algorithm has been first implemented in 2D, in order to be able to focus solely on developing and testing. Subsequently, it will be implemented in 3D using the PCL.

2.1 Goal and approach

The goal is to segment an articulated mesh M into its unknown number n of rigid parts $P = \{p_1, \ldots, p_n\}$ and extract the joints $J = \{j_1, ..., j_m\}$ linking those parts in form of a skeleton structure. In general, this is done by non-rigid registration of the point 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 alligns them with all points of D_0. Basically, a divide and conquer approach (see section 2.4) is implemented to recursively subdivide S_0 and D_0 into clusters trying to be matched.

Problem?

2.2 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) and are linked by joints. Comparing two poses S_0 and D_0 being adopted by the articulated object, the geodesic distance $g_{i,j}$ between two mesh points p_i and p_j remains constant. 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 in the same direction.

Problem?

2.3 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.

2.4 Approach

Problems of other approaches

2.4.1 General idea

The algorithm starts with two sets of point clouds S_0 and D_0 of an object M in different poses (see figure 2.1). The two point clouds are iteratively subdivided into point clusters $C = \{c_1, ..., c_m\}$. 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 rigid parts $P = \{p_1, ..., p_n\}$.

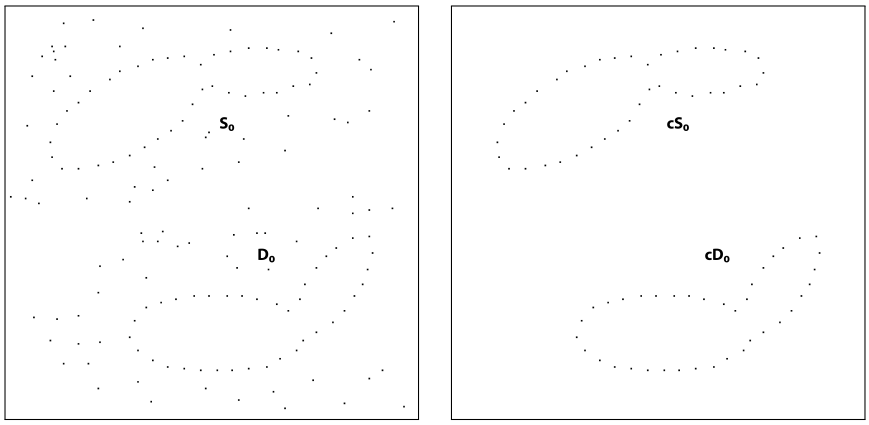


Figure 2.1: Taking a mesh in two different poses as input (a), removing noise of the input point clouds (b)

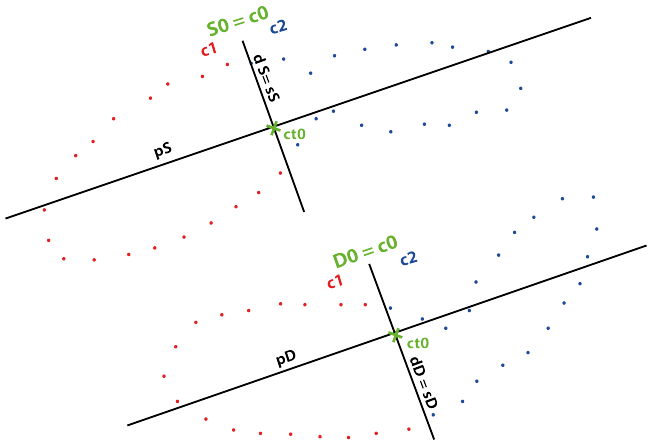


Figure 2.2: Dividing S_0 and D_0 into clusters by the divider d to match them with ICP.

2.4.2 Removing outliers

As a first step, the outliers of the two point clouds S_0 and D_0 are removed. This is done, by detecting point clusters $\mathcal{C} = \{C_0, \ldots, C_m\}$. A cluster c_i is grown from an unclustered point $\boldsymbol{p}_i$ of S_0 or D_0. Another point p_j is added to the cluster c_i if the euclidean distance between them $d_e(pt_i, pt_j)$ is below a pre-defined threshold τ. All points of c_i are then iteratively compared to the remaining unclustered points to allow the cluster to grow. Once, all points of c_i have been treated, another unclustered point is used as a seed. If there are no unclustered points left, the clusters with the highest number of points n are selected as input clusters cS_0 and $C_{1,0}$ (see figure 2.1).

2.4.3 Subdividing into clusters

As a next step, the two main clusters cS_0 and cD_0 are taken as input for further computation steps. If the matching between the two clusters does not suceed, they are subdivided into two clusters. In the other case, no subdividing is done and the procedure is repeated recursively for all clusters $C = \{c_1, ..., c_m\}$ of cS_0 until they can be matched to all clusters of cD_0

Divider position

To determine where to divide a cluster, it is taken advantage of the PCA (principal component analysis). The resulting dividers d_S and d_D are realised by computing the principal axes p_S and p_D through the centroids ct_0 and taking the perpendicular secondary axes s_S and s_D through the centroids (see figure 2.2).

Declaring the matching condition between two clusters

By applying the ICP and the nearest neighbour approach on two associated clusters C_p and C_q, a certain matching error e is computed between the cluster points $C_p = \{cp_0, ..., cp_m\}$ and the associated points $C_q = \{cq_0, ..., cq_m\}$. To make the matching error independent of the amount of cluster points, the average error per point

$$e_{\text{avg}} = \frac{\sum_{i=0}^{m} \|cp_i - cq_i\|^2}{|C_p|} \tag{2.1}$$

is computed, assuming that the two clusters C_p and C_q contain the same number of cluster points m. In case of different point amounts, the segmentation needs to be carried out at that clusters are always the same amount of points or some points are not considered during error amount calculation. To declare when two clusters match, it is quite essential to determine an appropriate threshold τ. In case of being overvalued, clusters are more likely to be matched which could result in insufficient subdividing. On the other hand, the clusters are difficult to be matched, which will result in further subdividing and the detection of too many rigid parts. The two clusters C_p are matching, if $e_{avg} < \tau$.

Cluster tree

The subdividing of the clusters cS_0 and cD_0 is realised by a depth-first approach in a tree. Consequently, cS_0 and cD_0 are subdivided from the left to the right. A node N of the tree contains two related clusters cS_i of S_0 and cS_i of D_0 and in case of subdividing it, a Node *left*, containing the subdivided clusters cS_{i+1} and cD_{i+1}, as well as a Node*right*, containing the subdivided clusters cS_{i+2} and cD_{i+2}. If two associated clusters $\{cS_{i+1}, cD_{i+1}\}$ in a Node N_i match, no further subdividing is performed. The resulting leaves of the tree are stored as matching clusters $C_l = \{c_{l1}, ..., c_{lm}\}$ (see figure 2.3). By applying the depth-first approach, the neighboring clusters in the list are also neighbouring clusters in the main clusters cS_0 or cD_0. As a result, in the following steps the skeleton structure from an object can be extracted, where parts are connected by joints.

2.4.4 Merging neighboring clusters to rigid parts

As a next step, neighboring clusters from C_l are iteratively merged and subsequently verified to match. This process is required to rejoin, if necessary, detected sub clusters to the rigid parts of the object. This is the case, if during the subdividing process, a rigid part was subdivided. The merging initiates with the first cluster $c_{li} = c_{l1}$ and its adjacent cluster c_{li+1}. If the resulting merged clusters $c_{ri} = c_{r1}$ can be matched, the merging proceeds with c_{ri} and the adjacent cluster c_{li+2}. If not, the merging is not executed and c_{li} is stored in a list of resulting clusters C_r. The whole procedure then starts with c_{li+1}. The process

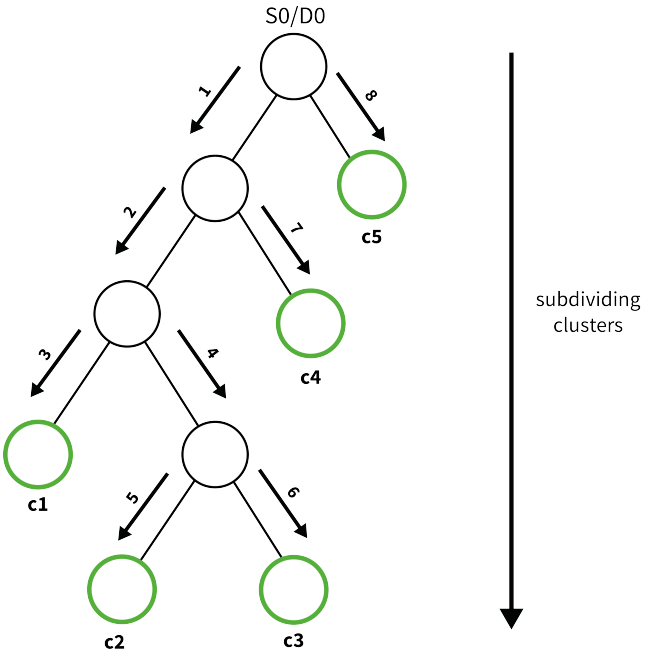


Figure 2.3: Subdividing of S0 and D0 into matching clusters by the depth-first approach

terminates if all clusters of C_l are verified and consequently the clusters of C_r are assigned to rigid parts $P = \{p_1, ...p_n\}$ (see figure 2.4).

2.4.5 Joint/skeleton estimation

After detecting the rigid parts $P = \{p_1, ...p_n\}$, they are linked with joints and the principal axes are for now assumed to form the skeleton.

2.5 Implementation

My approach was implemented in Java, using ImageJ as processing library, to focus on implementing and testing the algorithm in 2D. Another implementation is planned for 3D with the PCL to bring the attention to segmentation in 3D, and visualization. A class Cluster was implemented to store cluster points, its centroid, the orientation as well as the principal and secondary axes. For the subdividing, a cluster tree was implemented to simultainiously divide the initial clusters $c_{S,0}$ and $c_{D,0}$ into sub clusters 2.1. Each node N contains thereby two associated clusters $c_{S,i}$ and $c_{D,i}$. For the clustering and merging of clusters a List of Clusters was used to dynamically add and remove Clusters. Furthermore,

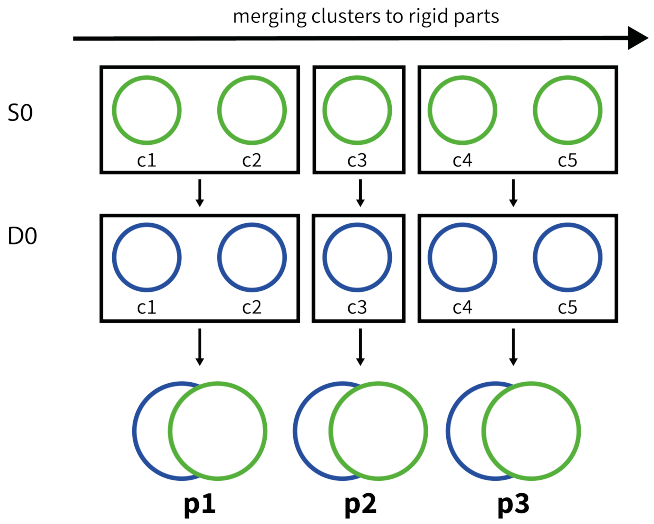


Figure 2.4: Detecting rigid parts of S0 and D0 by iteratively merging neighboring clusters of S0 and matching them with the merged clusters of DO. Genau erklären!

a class ICP was created, which takes two clusters to be matched as input.

2.5.1 Implementation Steps

1. The point clouds S_0 and D_0 are taken as input and clustering is computed to get the biggest clusters $cS_i = cS_0$ and $cD_i = cD_0$.
2. The centroids ctS_i and ctD_i of cS_i and cD_i are computed.
3. The principal axis pS_i and pD_i are computed through ctS_i and ctD_i in order to orient and align cS_i and cD_i.
4. The secondary axis sS_i and sD_i perpendicular to pS_i and pD_i through ctS_i and ctD_i are computed.
5. The ICP between cS_i and cD_i and an error distance per point e_{avg} are computed.
6. In case of $e_{avg} > \tau$, the dividers dS_i and dD_i to subdivide cS_i and cD_i are initialized with the secondary axis sS_i and sD_i.
7. The cluster points $CP = \{cp_0, ..., cp_n\}$ of cS_0 and cD_0 are either allocated to cS_{i+1} or cS_{i+2} depending on its position $cp_i.x$ to $ctS_i.x$ and $ctD_i.x$.
8. The algorithm continues with $cS_i = cS_{i+1}$ from step 2.
9. If $e_{avg} \leq \tau$
10. 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.

Rigid parts	τ	detected clusters	detected rigid parts
2	10	cell3	cell4
	5	cell3	cell4
	4	cell3	cell4
	3	cell3	cell4
	1	cell3	cell4
3	10	cell3	cell4
	5	cell3	cell4
	4	cell3	cell4
	3	cell3	cell4
	1	cell3	cell4

Table 2.1: Segmentation results

11. 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.

2.5.2 Intermediate results

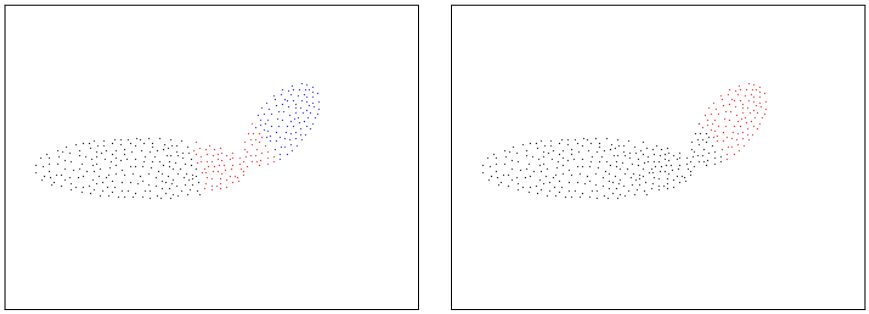
At first, the implementation was tested on two point clouds of an articulated object with only two rigid parts. The segmentation results are directly dependent on the matching error threshold τ which can bee seen on table 2.1. The higher the threshold τ, the less clusters and subsequently rigid parts can be detected, as two clusters are more likely to match and are not further subdivided. The lower τ, the more clusters and rigid parts will be detected, as clusters require further subdividing in order to match. Figure 2.5 shows the clusters of the object after subdividing them with $\tau = 4$ and the final rigid parts, figure 2.6 shows the result with three parts with $\tau = 5$. In case of $\tau = 3$ too many segments are detected, in case of $\tau = 1$ no segmentation is conducted (see Figure **??**). Figure XXX shows the two registered point clouds, in case of no segmentation and in case of a right detection of rigid parts.

Open Problems!

Algorithm 2.1: Subdividing of clusters

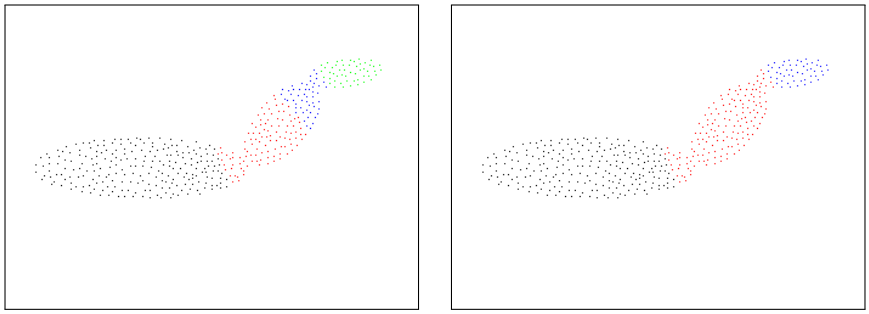
```
1: ClusterTree(a)
2:     j ← j − 1.
3: end
```

The algorithm works satisfactory for simple objects, in which the rigid parts are aligned like in a chain. More complex object, e.g. a human, where one rigid part is linked to more than two rigid parts, improvements have to be implemented, for example detecting the largest rigid part and apply the algorithm for all remaining clusters (arms, legs). In case of dividing checking for multiple clusters, e.g. in case of a human. Clusters not matching, as they don't have the same number of points, each point can only have one neighboring point.



(a) (b)

Figure 2.5: Taking a mesh in two different poses as input (a), removing noise of the input point clouds (b)



(a) (b)

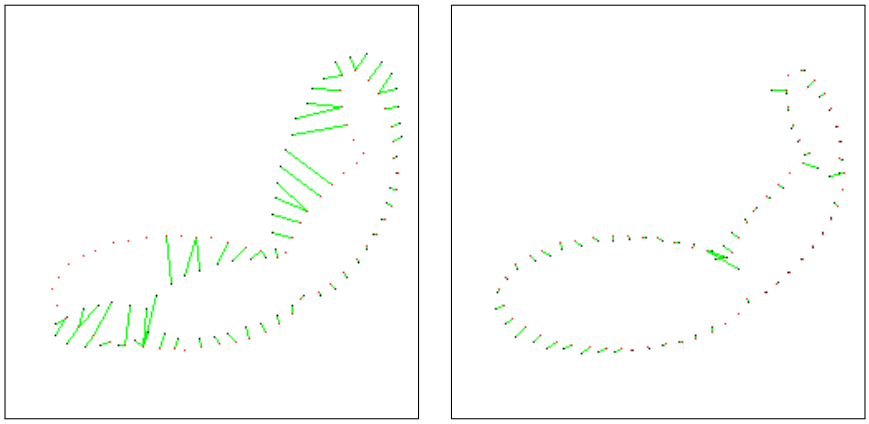
Figure 2.6: Taking a mesh in two different poses as input (a), removing noise of the input point clouds (b)

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 humen) another approach has to be found, as the skeleton structure is different than from a chain. (see Figure **??**).

2.5.3 Possible Improvements

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 2.6).



(a) (b)

Figure 2.7: Registration of two articulated objects before segmenting them into their rigid parts (a) and after the segmentation procedure (b).

Nearest neighbour

Another improvement would be to ensure that each point only has one nearest neighbour and considering the case, that there are not always the same amount of points in two clusters.

Checking for multiple clusters

Also a quite similar approach to [5], in case of dividing or detecting a cluster, for the remaining clusters, region growing is conducted.

2.6 LRP

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.

2.6.1 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.

2.6.2 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 transformation 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 $G_s = \{C_1, \ldots, C_n\}$ and $G_t = \{C_1, \ldots, C_n\}$ 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 $\{1, \ldots, n\}$, 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.

2.6.3 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.

Chapter 3

Conclusion

In the beginning of the project, intense research was done in order to focus on one main problem. For that, it was quite essential to find the state-of-the-art regarding motion and pose estimation, in order to find a relevant topic as master thesis project. During research, the topic unsupervised pose estimation appeared. By focusing research in this direction, the non-rigid registration became a major indicator for possible optimizations. Taking existing methods as reference (see Chapter 1.3), an own approach for 2D point clouds was developed, which should reduce the computation steps of detecting the rigd parts of an articulated non-rigid object. To do that, a divide-and-conquer approach was developed, which recurisively divides two point-clouds of the same object in different poses into matching clusters. The subdividing was realised with a tree and depth-first traversal to divide the point-clouds from left to right. As a next step all neighboring cluster were verified to be merged, in case of having subdivided a rigid part. After the mergin the rigid parts from left to right of the objects are detected.

3.1 Future work

The focus in the next semester will be intensive testing of the current implementation and adding possible improvements. Eventually other existing approaches are taken into account to be able to segment more complex objects. The next main step is then, to implement the finished segmentation algorithm in 3D using the PCL.

References

[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. 4, 5).

[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. 4).

[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. 5).

[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. 5).

[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. 5, 15).

[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 pp. 4, 5).

[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. 4).