Unsupervised identification of the rigid part of an unknown articulated object

Anna M. Maureder

27th January, 2018

Contents

Chapter 1

Pose estimation

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

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 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 handels the registration process by first detecting the largest rigid part [5].

1.3.1 EM-algorithm

Here comes an explanation of Anguelov with pictures.

1.3.2 LRP

Here comes the LRP algorithm with pictures.

1.3.3 Symmetrization

Here comes an explanation of 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 $P = \{p_1, ...p_n\}$ and extract the joints $J = \{j_1, ...j_{n-1}\}$ 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.

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.

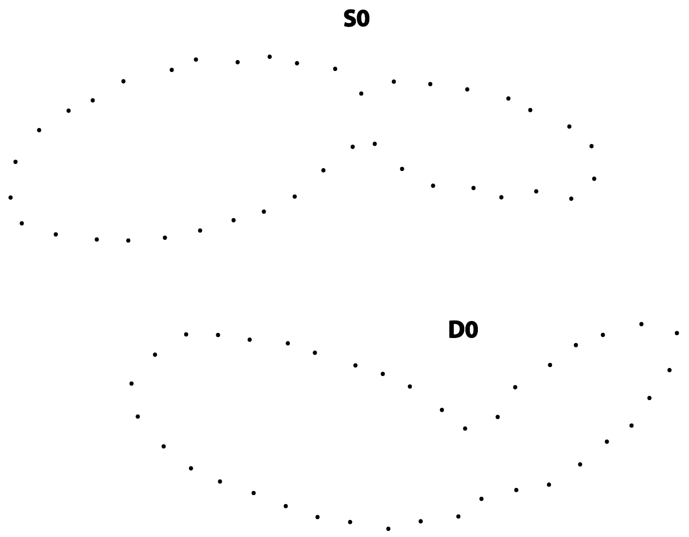

Figure 2.1: Taking a mesh M in two different poses S_0 and D_0 as input.

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

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 $P = \{p_1, ..., p_n\}$.

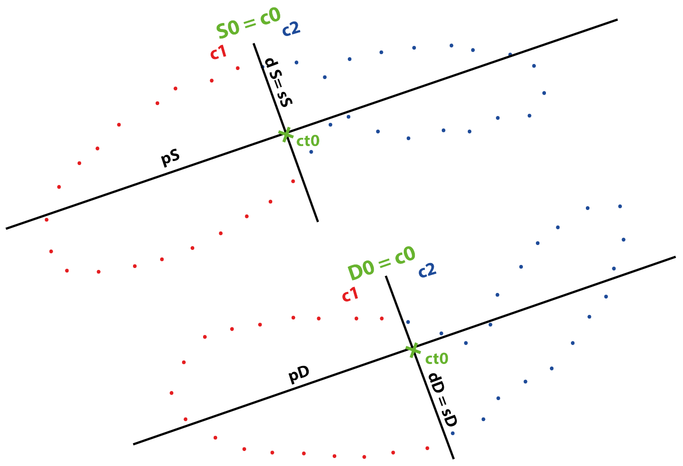

Figure 2.2: Dividing S_0 and D_0 into clusters by the divider d to match them with ICP.

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 $C = \{c_0, ..., c_m\}$. A cluster c_i starts with an unclustered point p_i of S_0 or D_0. Another point p_j is then added to the cluster c_i if the euclidean distance between them $d_e(p_i, p_j)$ is under a certain threshold τ. If p_j can not be assigned to any cluster of C a new cluster c_{m+1} is created. Once all points of S_0 and D_0 are assigned to clusters, the biggest cluster c_0 is selected.

2.4.3 Subdividing into clusters

As a next step, the two main clusters c_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 S_0 until they can be matched to all clusters of D_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_i of S_0 and D_0, a certain matching error is computed between the cluster points $CP = \{cp_0, ..., cp_n\}$ and the associated points $QP = \{qp_0, ..., qp_n\}$. To declare when two clusters are matching, it is important to find 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 too many rigid parts. To make the matching error independent of the amount of cluster points, an average error per point

$$e_{avg} = \frac{\sum\limits_{i=0}^{m} \|cp_i - cq_i\|^2}{|CP|} \tag{2.1}$$

is computed, assuming that the two clusters $c_{S,i}$ and $c_{D,i}$ contain the same amount 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.

Version 1:

Version 2:

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 three 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 object S_0 or D_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

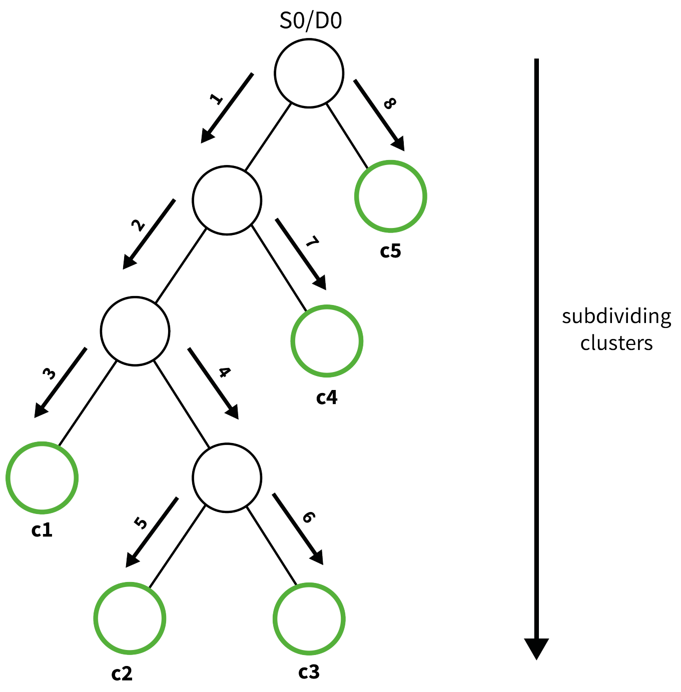

Figure 2.3: Subdividing of S0 and D0 into matching clusters by the depth-first approach

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 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, the 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.

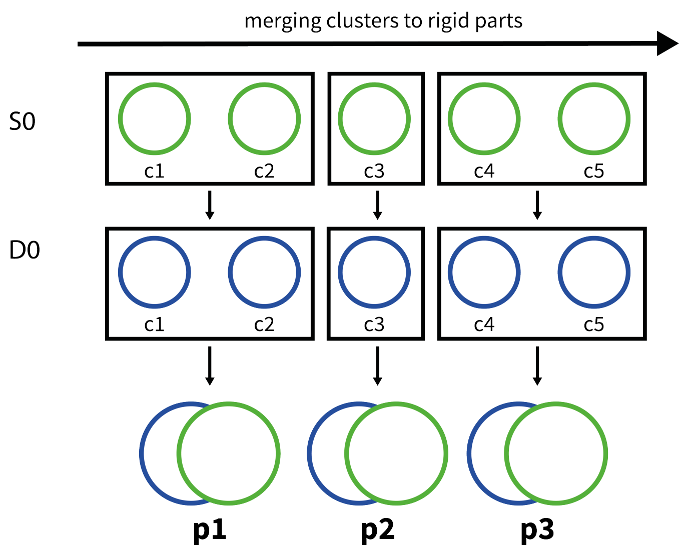

Figure 2.4: Detecting rigid parts of S0 and D0 by iteratively merging neighboring clusters of S0 and matching them with the merged clusters of DO.

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/remove Clusters. Furthermore, 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 $c_{S,i} = c_{S,0}$ and $c_{D,i} = c_{D,0}$.
2. The centroids $ct_{S,i}$ and $ct_{D,i}$ of $c_{S,i}$ and $c_{D,i}$ are computed.
3. The principal axis $p_{S,i}$ and $p_{D,i}$ are computed through $ct_{S,i}$ and $ct_{D,i}$ in order to orient and align $c_{S,i}$ and $c_{D,i}$.
4. The secondary axis $s_{S,i}$ and $s_{D,i}$ perpendicular to $p_{S,i}$ and $p_{D,i}$ through $ct_{S,i}$ and $ct_{D,i}$ are computed.
5. The ICP between $c_{S,i}$ and $c_{D,i}$ and an error distance per point e_{avg} are computed.
6. In case of $e_{avg} > \tau$, the dividers $d_{S,i}$ and $d_{D,i}$ to subdivide $c_{S,i}$ and $c_{S,i}$ are initialized with the secondary axis $s_{S,i}$ and $s_{D,i}$.
7. The cluster points $CP = \{cp_0, ..., cp_n\}$ of $c_{S,0}$ and $c_{D,0}$ are either allocated to $c_{S,i+1}$ or $c_{S,i+2}$ depending on its position $cp_i.x$ to $ct_{S,i}.x$ and $ct_{D,i}.x$.
8. The algorithm continues with $c_{S,i} = c_{S,i+1}$ from step 2.
9. If $e_{avg} \leq \tau$

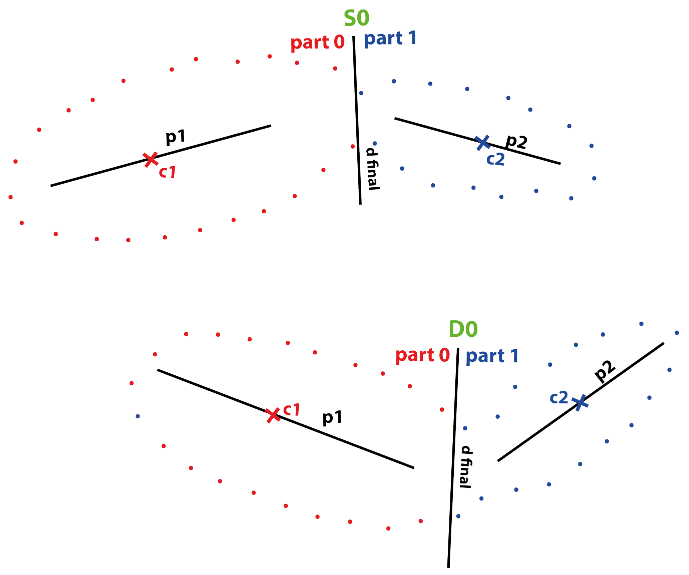

Figure 2.5: Assigning of the rigid parts P =($part_0$, $part_1$) after termination of segmentation process.

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.
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

First on, the implementation was tested with two point clouds of an articulated object with only two rigid parts. Figure XX shows the clusters of the object after subdividing, figure XX shows the resulting rigid parts. The results are directly dependent of error threshold τ. In case of XXX too many segments are detected, in case of XXX no segmentation is conducted. (see Figure 2.5).

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 humen) another approach has to be found, as the skeleton structure is different than from a chain. (see Figure 2.5).

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).

2.6 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.

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 $G_s = \{C_1, \dots, C_n\}$ 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, ..., 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.

2.7 Other approaches

2.7.1 Points-to-Ellipse fitting

2.7.2 Algorithm

This algorithm only requires one point cloud containing m points $\{pt_0, ..., pt_m\}$. 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

$$\frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} = 1 \tag{2.2}$$

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.

$$(1 - \frac{x^2}{r_1^2}) \cdot r_2^2 = y^2 \tag{2.3}$$

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

$$n = \sum_{i=0}^{m} \begin{cases} 1 & if \quad \|pt_i.y^2 - y^2\| < \tau \\ 0 & otherwise \end{cases} \tag{2.4}$$

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

2.7.3 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.

2.7.4 Results

2.7.5 Reusing detected shapes

After termination of the algorithm, one point cloud can be segmented into its rigid parts P $\{\mathrm{part}_1, ..., \mathrm{part}_n\}$. 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.

2.8 Results

2.9 Future work

There are still improvements regarding the algorithm in 2D, therefore the focus in the next semester will be intensive testing and possibly taking different approaches into account. The next main step is to implement the segmentation method in 3D using the PCL.

Chapter 3

Conclusion

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 p. 4).

[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 pp. 3, 4).

[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. 4).

[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 p. 4).

[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. 4).

[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. 3).