Extraction and Reconstruction of Articulated Robots from Point Clouds of Manufacturing Plants

Kota Kawasaki[1], Kakeru Takeda[2] and Hiroshi Masuda[3]

[1] The University of Electro-Communications, k.kawasaki.uec@gmail.com
[2] The University of Electro-Communications, k.takeda@uec.ac.jp
[3] The University of Electro-Communications, h.masuda@uec.ac.jp

Corresponding author: Hiroshi Masuda, h.masuda@uec.ac.jp

Abstract. In manufacturing plants, many articulated robots are installed and operated. These robots are required to operate without collision with other objects. For collision detection, it is effective in simulating robot motion in a faithful virtual environment. In recent years, it has become possible to acquire dense point clouds of manufacturing plants where many robots operate in cooperation. However, since the point clouds include articulated robots themselves, it is necessary to exclude the point clouds of the robots for the simulation of robot motion. In this paper, we propose a method for extracting articulated robots in arbitrary postures from point clouds of manufacturing plants. In our method, the CAD model and structural information of each robot are used as prior knowledge. CAD models of robots are available from the manufacturer in most cases, and the connection relations of the robot links can be described using the unified robot description format (URDF). Based on the link relations, the robot links are then aligned to the point cloud to determine the robot's posture. Other moving objects that operate with the robot, such as end effectors, wire harnesses, and other accessories, can also be detected from the point cloud by identifying the robot links. We evaluated our method using actual point clouds in an automobile factory. Our experimental results showed that our method could identify the postures of articulated robots from point clouds and provide the 3D environment for robot simulation.

Keywords: Point-Clouds, Shape Reconstruction, Industrial Robots, Terrestrial Laser Scanner, Point Processing.
DOI: https://doi.org/10.14733/cadaps.2025.616-628

1 INTRODUCTION

In manufacturing plants, many articulated robots work together. Each robot must operate without colliding with other robots or equipment. In addition, the robot must operate with high precision, stability, and efficiency. Therefore, robot teaching to obtain the optimal motion plan requires an enormous amount of labor, time, and cost. In general, robot teaching is performed by actually moving the robot using a device such as a teaching pendant or by the operator directly moving the robot. However, robot teaching has several problems to be solved: operation time must be reduced because the production line must be stopped during the teaching process; skilled operators must be secured and trained because the quality and efficiency of the teaching depends on the technical skills and experience of the operators; and the robot needs to be carefully monitored for accidents caused by mishandling, such as contact with humans or damage caused by contact with objects in the robot's surroundings. For these reasons, off-line robot teaching is widely carried out before the teaching on actual robots.

Recent advances in laser scanners have made it possible to obtain 3D geometric information about the current state of manufacturing plants. By using a terrestrial laser scanner (TLS), 3D data of a manufacturing plant can be acquired as point clouds of hundreds of millions to billions of points. Point clouds are useful for constructing a faithful virtual environment and simulating production lines in the virtual environment. In particular, off-line robot teaching based on the virtual environment is very effective in assembly factories where many articulated robots work together. However, point clouds include various objects, such as floors, machine tools, safety fences, and robots. In order to use point clouds for the simulation of robot operation, it is necessary to detect and segment such objects from point clouds. Especially, it is important to classify moving objects and fixed objects; moving objects include robots, end effectors, wire harnesses, etc., and fixed objects include workbenches, stands, tool boxes, floors, fences, etc.

In robot motion simulation, if robots can be identified from the point cloud, the virtual environment can be created as a point cloud by excluding moving objects, and collision checks between the 3D models of robots and the virtual environment can be performed. Furthermore, points of attached parts and wire harnesses can be obtained as neighbor points of robot points. In most cases, these points are often not included in the CAD models of robots, but a precise collision check can be performed by adding these points to 3D models of robot links. It is desirable for precise and efficient motion planning to reconstruct faithful virtual environment of both moving and fixed objects.

Many shape reconstruction methods from point clouds have been proposed [2]. Schnabel et al. [12] proposed the random sampling consensus (RANSAC) based method to detect primitive shapes such as planes, spheres, and cylinders. Li et al. [7] refined the detected primitives considering symmetries and structural repetitions among them. These methods can be used for reconstructing the shapes of industrial parts. However, they are not suitable for articulated robots, which consist of moving parts and do not have fixed shapes.

For articulated robots used in manufacturing plants, their 3D CAD models are often available from the manufacturer. Therefore, the iterative closest point (ICP) [1] and the principal component analysis (PCA) [6] can be used for aligning each part of CAD models to the point cloud. Bey et al. [3] fitted CAD models to point clouds using Bayesian formulation. However, their method is not applicable when the object shape varies significantly with its posture, as in the case of a robot with articulated arms.

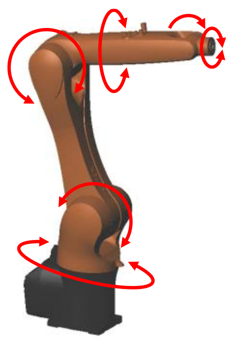
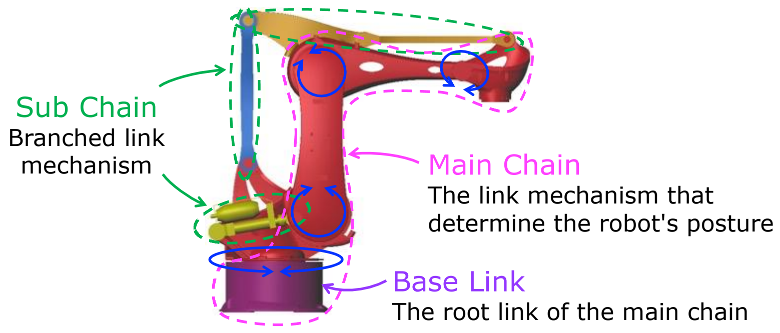
The 3D model of an articulated robot can be considered as an assembly model with a joint mechanism of links. Shah et al. [13] fitted CAD models to point clouds by using a simulated annealing method to determine the parameters of assembly models consisting of several parts. Hu et al. [5] used deep learning and pre-defined CAD model templates to recognize objects from point clouds and to estimate parameters associated with the objects. These methods based on parameter estimation are effective when assembly models can be determined by sequentially identifying components from point clouds. However, articulated robots used in manufacturing plants often have closed loops of link mechanisms, and the point clouds of robots include accessories such as wiring harnesses, which are not included in CAD models. Moreover, since robots in a large-scale manufacturing plant have a variety of end-effectors, it is not easy to train machine learning classifiers to recognize various combinations of robots and end-effectors. In addition, since the same type of robots are used repeatedly in combination with various end-effectors, it is desirable to be able to easily define models of link mechanisms for robots and end-effectors. Therefore, there are difficulties in applying existing methods to the point clouds of articulated robots.

In this paper, we propose a method for extracting articulated robots in arbitrary postures from point clouds of manufacturing plants by fitting CAD models of robots and end-effectors. 3D models of robots are available from the manufacturer in most cases, and the connection relations of the robot links can be described using the unified robot description format (URDF). The URDF is an XML format for representing robot models and is commonly used in robot simulations [14]. The 3D model of a robot is an assembly model in which each link is represented as an independent 3D CAD model. In our method, the 3D model and URDF data of each robot are used as prior knowledge to obtain the assembly model that fits the point cloud. Then, the obtained 3D model is used to extract the point cloud corresponding to each link and to detect surrounding objects of the robot, such as wire harnesses.

Our method covers vertical articulated robots with closed loops. For robots that have a serial link mechanism consisting of revolute joints, as shown in Figure 1(a), the parameters of the robot links can be determined sequentially. On the other hand, the large palletizing robot shown in Figure 1(b) has a branched link mechanism. In such vertically articulated robots, there are link mechanisms with closed loops due to rigidity and loading. Since a closed-loop link mechanism cannot be represented by URDF, we represent the link mechanism using multiple URDFs consisting of a *main chain* and *sub chains*. The main chain is a link mechanism that determines the robot's posture. We refer to the links in the main chain as *main links*. The sub chain is a link mechanism that moves in conjunction with the main chain. We refer to such additional links as the *sub links*.

To determine the robot's posture from the point cloud, the points corresponding to each robot link are extracted from the point cloud. First, a CAD model of the base link is aligned to the point cloud. The base link is defined as the root link installed on the floor or stand. Then, the main links are fitted by following the main chain from the base link, using the connection relations of the links described in the URDF. Since the sub-link moves in conjunction with the main chain, the CAD models of sub links are fitted after the main chain is determined, using inverse kinematics to calculate the posture. Finally, the points of all the links and the points of accessories such as wiring harnesses are extracted. The points of such moving objects are excluded from the point cloud in order to extract fixed objects.

In the next section, we explain the description of the joint mechanism using the URDF. In Section 3, we discuss the method of aligning the base link. In Section 4, we discuss the method of fitting main links. In Section 5, we discuss the fitting method of sub links. In Section 6, we discuss the extraction method of point clouds corresponding to the links. In Section 7, we describe experimental results, and finally, we conclude our research.



Sub Chain
Branched link mechanism
Main Chain
The link mechanism that determine the robot's posture
Base Link
The root link of the main chain

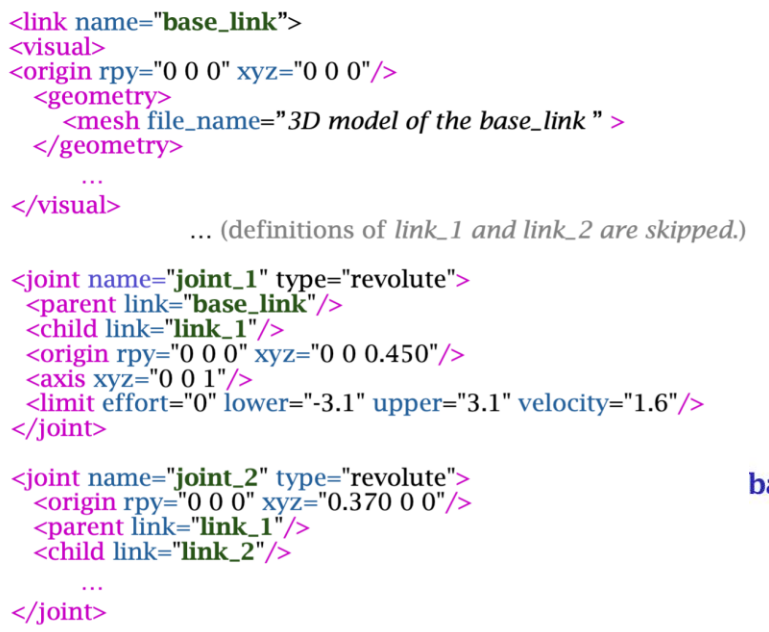
(a) (b)

Figure 1: Vertical articulated robots: (a) 6-axis small robots (KUKA KR 10 R1100 sixx), (b) Palletizing Robots (KUKA KR 700 PA).

2 URDF FORMAT FOR DESCRIBING LINKS AND JOINTS OF ROBOT

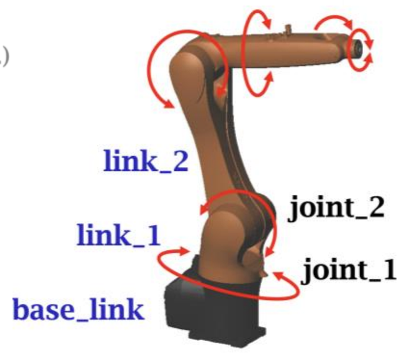
The URDF maintains 3D models of links and their joint data. Figure 2(a) shows an example of URDF description consisting of links and joints in Figure 2(b). Each link, such as *base_link*, maintains the 3D model of the link. Joint data, such as *joint_1*, maintains the relationships between the parent and child links, the rotation axis, and the rotation range. The relative position of the two links is represented as *xyz* and *rpy*, which indicates roll, pitch, and yaw angles.

The robot assembly model shown in Figure 3(a) can be created by combining 3D models of links. Since a URDF cannot describe a closed loop, the robot model is specified by describing the main chain and sub-chains in URDF, as shown in Figure 3(b, c). In our method, the 3D model of each link is converted into a point cloud by randomly sampling points on the 3D model. In this paper, point clouds generated from 3D models are referred to as CAD point clouds and point clouds obtained using a TLS are called a TLS point clouds. Figure 2(d) shows an assembly model consisting of CAD point clouds.



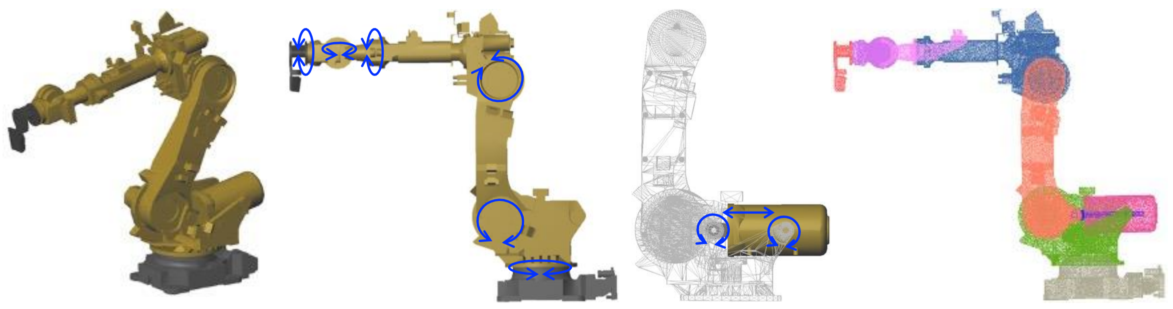
```
<link name="base_link">
<visual>
<origin rpy="0 0 0" xyz="0 0 0"/>
   <geometry>
      <mesh file_name="3D model of the base_link " >
   </geometry>
         ...
</visual>
                    ... (definitions of link_1 and link_2 are skipped.)

<joint name="joint_1" type="revolute">
  <parent link="base_link"/>
  <child link="link_1"/>
  <origin rpy="0 0 0" xyz="0 0 0.450"/>
  <axis xyz="0 0 1"/>
  <limit effort="0" lower="-3.1" upper="3.1" velocity="1.6"/>
</joint>

<joint name="joint_2" type="revolute">
   <origin rpy="0 0 0" xyz="0.370 0 0"/>
   <parent link="link_1"/>
   <child link="link_2"/>
         ...
</joint>
```

(a)



(b)

Figure 2: Example of URDF: (a) Description of links and joints, (b) Links and joints.



(a) (b) (c) (d)

Figure 3: Process of acquisition: (a) Robot model, (b) Main chain, (c) Sub chain, (d) CAD point clouds.

3 CALCULATION OF THE BASE LINK

Since most robots and machine tools are installed on the floor in manufacturing plants, objects on the floor can be separated by removing the floor points, which consists of large horizontal planes. Planar regions such as floors can be effectively detected from TLS point clouds by using the method proposed by Masuda et al. [8]. This method maps a point cloud onto a 2D image using the azimuth and elevation angles of the laser beam, and planar regions are detected on the 2D image. Then, each robot is selected from the segmented point clouds using common classification techniques. We applied this method to point clouds of assembly lines in an automobile factory. When the floor points were excluded from the point clouds, each robot could be easily selected from the segmented objects because there were no other objects of the same size as robots. In general, deep learning methods such as PointNet [10] or PointNet++ [11], can be used for the classification of separated objects.

After the point cloud of each robot is obtained, each link of the robot is detected from the point cloud. Since the main links of the robot are connected sequentially from the fixed base, it is important to detect the position of the base link as accurate as possible. This section discusses a method for calculating the position of the base link.

3.1 Extraction of Candidate Points of the Base Link

Figure 4(a) shows the point cloud of a robot. Each robot is installed on the floor or on a stand, which is a horizontal plane. To detect the installation plane, planes are detected from the point cloud of the robot, as shown in Figure 4(b). For the installation plane, a horizontal plane that is below the robot point cloud and larger than the size of the base link is selected. Depending on the posture of the robot, planes extracted from each robot link can be horizontal, but these planes can be distinguished from the installation plane because they are smaller than the size of the base link. In the example in Figure 4(c), the lowest plane is selected as the installation plane.

The point cloud is converted to a k-nearest neighbor graph by creating edges between neighbor points, as shown in Figure 4(d). Then, Dijkstra's algorithm is applied at each point to calculate the shortest path from the installation plane. When starting the search for connecting edges from the

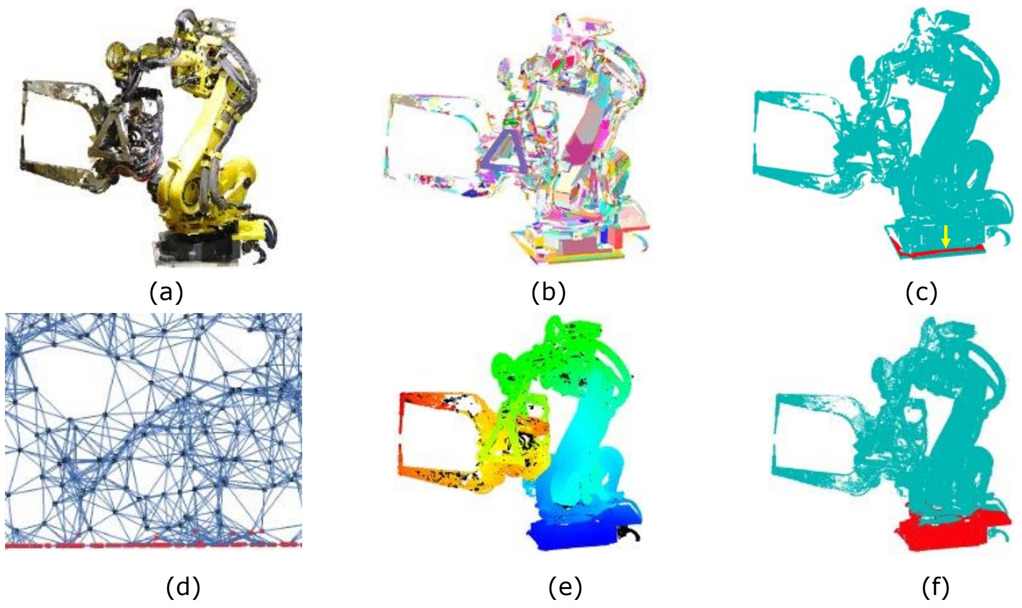


Figure 4: Detection of points of the base link: (a) Point cloud, (b) Plane detection, (c) Selection of an installation plane, (d) K-nearest neighbor graph, (e) Distance from the installation plane, (f)Candidate points of the base link.

installation plane, only edges above the installation plane are traversed, while edges below the installation plane are ignored to avoid searching downward for non-robot edges.

In Figure 4(e), the color of each point indicates the distance of the shortest path. The shortest-path distances are also calculated from the CAD point cloud of the base link, and the maximum distance from the installation plane is calculated for the base link points. The candidate points of the base link is selected as TLS points with distances shorter than the maximum distance, as shown in Figure 4(f).

3.2 Registration of the Base Link

The CAD point cloud of the base link is fitted to the candidate points of the base link. For rough registration, an oriented bounding box (OBB) is created from each of the CAD point clouds and the TLS point clouds, as shown in Figure 5(a, b). The OBB is created by mapping the points onto the installation plane and then performing 2D PCA to determine the orientation. Next, the CAD point cloud is moved to a position where the centers of the OBBs are coincident, and their heights from the installation plane are aligned. Then, the OBB of the CAD point cloud is rotated at equal intervals around an axis vertical to the installation plane so that the sum of the shortest distances between each point in the CAD point cloud to the TLS point cloud is minimized. As shown in Figure 4(c), the calculated position is used as the rough registration result. Finally, the sparse ICP algorithm [4] is applied for precise registration, as shown in Figure 5(d).

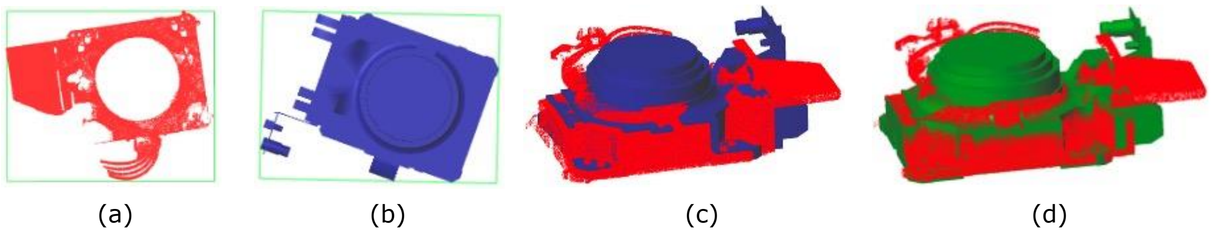


Figure 5: Process of base-link registration: (a) OBB of base-link TLS points, (b) OBB of the base-link CAD points, (c) Result of rough registration, (d) Result of sparse ICP registration.

4 CALCULATION OF MAIN LINKS

In our method, the CAD point cloud of each main link is fitted to the TLS point cloud sequentially from the base link, as shown in Figure 6. Since the rotation axis of each link is defined in the URDF, the rotation angle of the link around the axis can be calculated. Initially, the rotation angle of each link is determined roughly, and then neighbor points of the roughly fitted CAD point clouds are extracted from the TLS point clouds as the candidate points of the link. The exact rotation angle is calculated by constrained registration using the relationships between links.

To roughly determine the rotation angle at each joint, the CAD point cloud is rotated at equal intervals around the rotation axis defined in the URDF. The rotation angle is determined so that the sum of the shortest distances from each point in the CAD point cloud to the TLS point cloud is minimized. In order to improve computational efficiency, the shortest distances between the CAD and TLS point clouds are calculated on the 2D lattice created from the TLS point cloud. Each point of the TLS point cloud is measured by the laser beam emitted from the laser scanner, as shown in Figure 7(a). Therefore, the X, Y, Z coordinates of each point can be mapped on the 2D plane defined by the irradiation angles θ and ϕ. As shown in Figure 7(b), the nearest neighbor of each point in the CAD point cloud can be efficiently detected from the neighborhood of the pixel to which the point is mapped.

However, when the link shape is nearly rotationally symmetric, the rotation angle may not be calculated accurately using this method. For example, the link shape shown in Figure 8 is nearly rotationally symmetric, and the center of the point clod is close to the axis of rotation. Such links change little when rotated. In such cases, to avoid large registration errors, the next child link is also used to calculate the rotation angle.

Once the rotation angle is roughly determined, neighbor points of the roughly fitted CAD point clouds are extracted from the TLS point clouds as the candidate points of the link. The rotation angles of the link are calculated precisely by applying constrained registration to the candidate points. When the CAD point cloud is aligned to the TLS point cloud, the axis of rotation must coincide with the axis of rotation of its parent link. To satisfy this constraint, we introduce a constrained registration in which the two axes coincide.

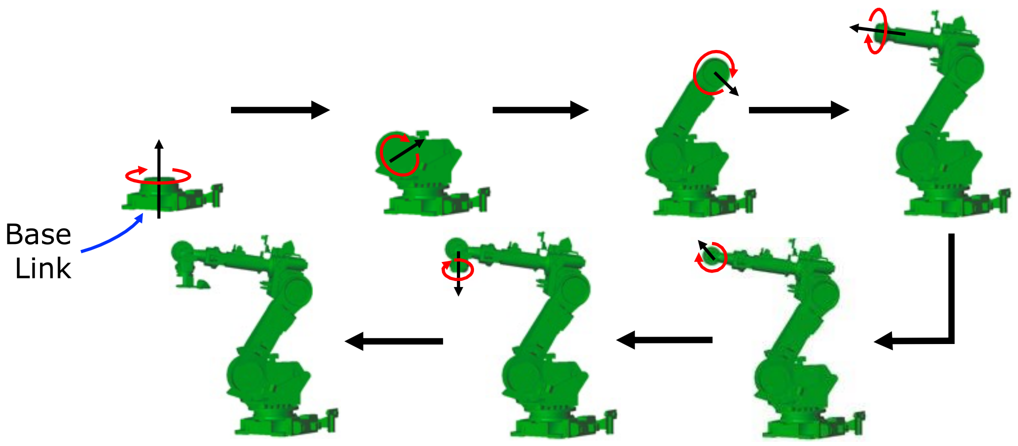
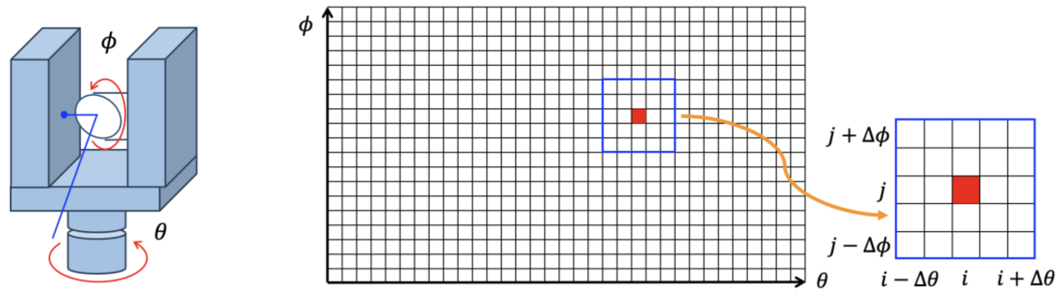


Figure 6: Fitting of main links.



(a) (b)

Figure 7: Neighbor search on 2D lattice of TLS point cloud: (a) Irradiation angle (θ, ϕ) of the laser beam, (b) Neighbor search on the 2D lattice defined by the irradiation angles.

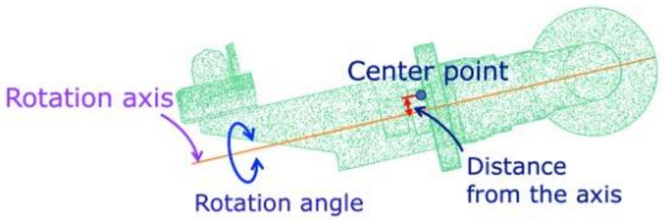
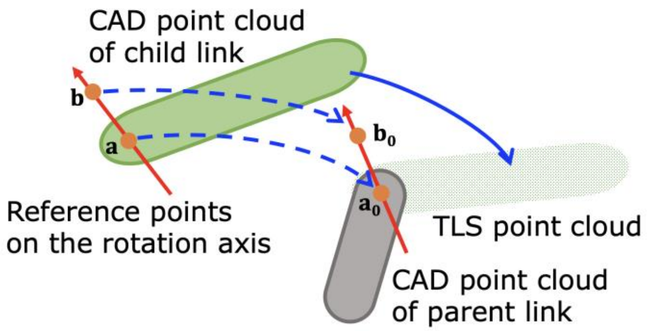


Figure 8: Example of rotationally symmetric link.

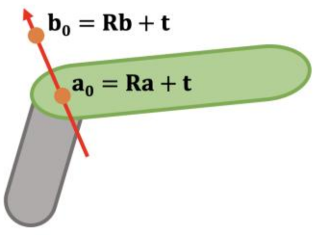
Each axis can be represented by two reference points on the axis. Let a and b be the reference points of the child link and $\mathbf{a_0}$ and $\mathbf{b_0}$ be the reference points of the parent link, as shown in Figure 9(a). To align the axes of the two links, the rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ are calculated so that the following equation is minimized:

$$E = \sum_{i=1}^{N} |\mathbf{Rp}_i + \mathbf{t} - \mathbf{q}_i|^2 + \lambda(|\mathbf{Ra} + \mathbf{t} - \mathbf{a_0}|^2 + |\mathbf{Rb} + \mathbf{t} - \mathbf{b_0}|^2) \tag{4.1}$$

Where $\mathbf{p}_i$ is a point in the CAD point cloud, $\mathbf{q}_i$ is the nearest TLS point from $\mathbf{p}_i$, and λ is a constant. $\mathbf{R}$ and $\mathbf{t}$ that minimizes Equation (4.1) are calculated by iteratively updating the corresponding points. Finally, the CAD point cloud of the child link is moved to the position that coincides with the TLS point cloud so that the axes of rotation coincide, as shown in Figure 9(b).


CAD point cloud of child link
b
a
$\mathbf{b_0}$
$\mathbf{a_0}$
Reference points on the rotation axis
TLS point cloud
CAD point cloud of parent link

(a)

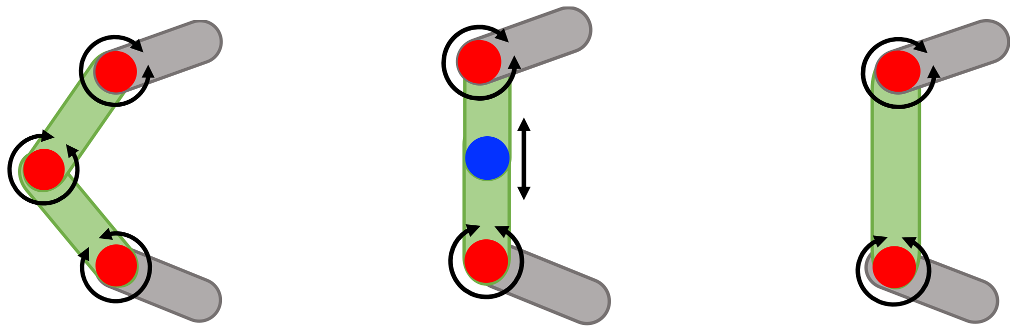

$\mathbf{b_0} = \mathbf{Rb} + \mathbf{t}$
$\mathbf{a_0} = \mathbf{Ra} + \mathbf{t}$

(b)

Figure 9: Constrained registration: (a) Constraints on revolute joints, (b) Registration result.

5 CALCULATION OF SUB-LINKS:

Once the rotation angles of the main links are determined, the parameters of the sub links are calculated using the positions and postures of the main links. Figure 10 shows link mechanisms commonly used in industrial robots. In these examples, since the positions of the links at both ends are determined by the fitted links, the positions and postures of the sub links can be computed as inverse kinematics solutions.



(a) (b) (c)

Figure 10: Link mechanism for sub-chain: (a) Revolute-revolute-revolute linkage, (b) Revolute-prismatic-revolute linkage, (c) Revolute-revolute linkage.

In some cases, the solution is not uniquely determined. For example, the link mechanism in Figure 11 has two solutions that satisfy the constraints of the main links. In such a case, the possible solution can be selected by considering the rotation range of the link and the interference with other links. In Figure 11(a), since the rotation angle is limited to a certain range, the solution closer to the initial angle is selected among the two solutions. In the case of Figure 11(b), one of the two solutions causes the connected links to separate, but such a situation does not occur in reality.

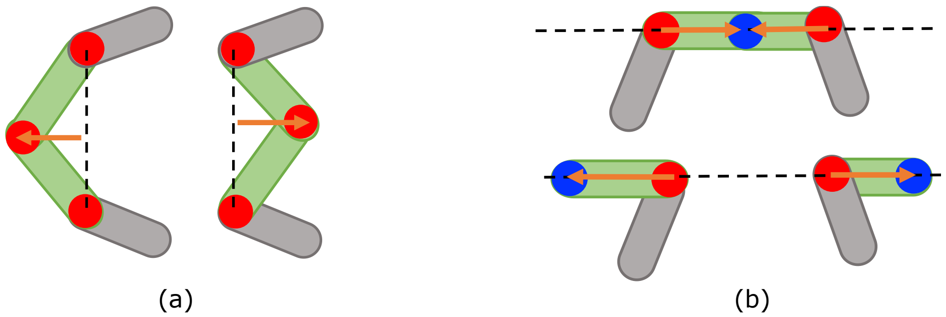


Figure 11: Inverse kinematics solutions: (a) Revolute-revolute-revolute linkage, (b) Revolute-prismatic-revolute linkage.

6 EXTRACTION OF ADDITIONAL OBJECTS

Once the positions and postures of all the links of the articulated robot are determined, the TLS point cloud for each link can be extracted by comparing the TLS point cloud with the fitted CAD point cloud. The remaining points that do not correspond to any link are considered end effectors, robot attachments, or wire harnesses.

End effectors also consist of multiple links, as shown in Figure 12. Since the end effector is connected to the end link in the main links, the position and posture of the base link of the end effector can be determined. From the base link of the end effector, the position and posture of each link of the end effector can be determined sequentially in the same manner as the main links.

Robot attachments move together with the links of the robot. Therefore, points of each attachment can be regarded as part of the links. This enhanced 3D model can be used for accurate collision detection in robot motion simulation.

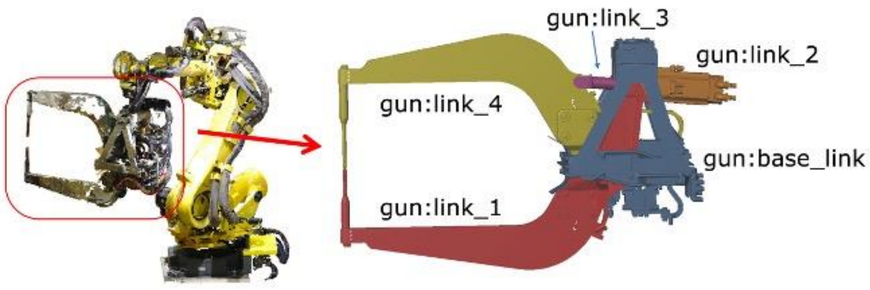


Figure 12: Link mechanism of an end effector.

7 EXPERIMENTAL RESULTS

We evaluated the proposed method using the actual point clouds captured by a TLS. In our evaluation, the point cloud shown in Figure 4(a) and CAD models in Figure 13 were used. This point cloud was captured from a FANUC industrial robot used in an automobile factory. The point cloud is measured from multiple positions to reduce missing points due to occlusion. The robot is mounted with an end effector for welding and is composed of a main chain consisting of seven revolute joints and two sub chains enclosed by red rectangles, as shown in Figure 13.

7.1 Evaluation with Actual TLS Point Clouds

We fitted the 3D model of each link of the robot and the end effector to the actual TLS point cloud. The results are shown in Figure 14. By applying constraint registration, precise fitting results could be obtained even for the links with missing points. The auxiliary links could also be positioned in appropriate positions by computing the parameters of the sub chains.

Next, we extracted the points on each link as neighbor points of the fitted CAD model. In Figure 15(a), the points on each link are shown in a different color. Figure 15(b) shows an assembly model of the articulated robot with the end effector, with the same position and posture of the point cloud. Attachments, such as wire harnesses, are shown in black. As shown in Figure 15, the point clouds corresponding to the links were successfully extracted by using fitted CAD models, and robot attachments and wire harnesses were also successfully separated from the point clouds.

In our method, since each link is aligned sequentially from the base link, registration errors may accumulate. In this evaluation, a misalignment of about 1 cm was observed on the end effector due to the accumulation of misalignment, as shown in Figure 16. In this research, only sequential registration was performed, but more accurate fitting would be possible by repeatedly refining registration results by using a multiview registration technique [9].

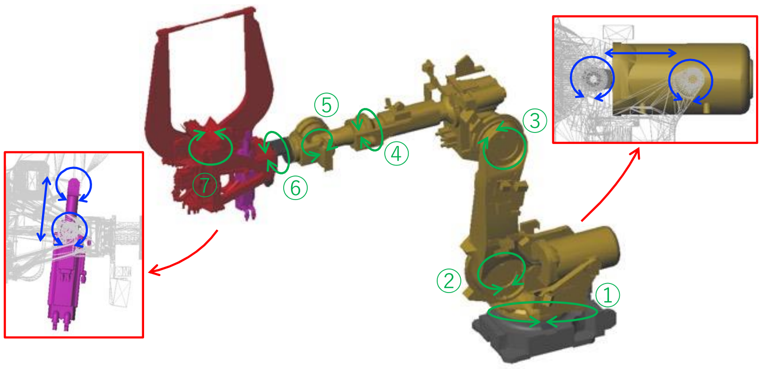


Figure 13: CAD models and the link mechanism of a FANUC articulated robot.

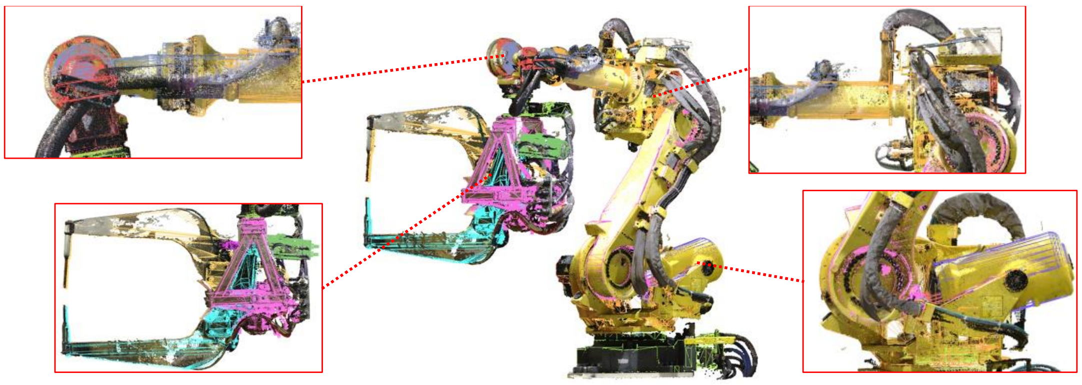


Figure 14: Fitting 3D models of links to TLS point cloud.

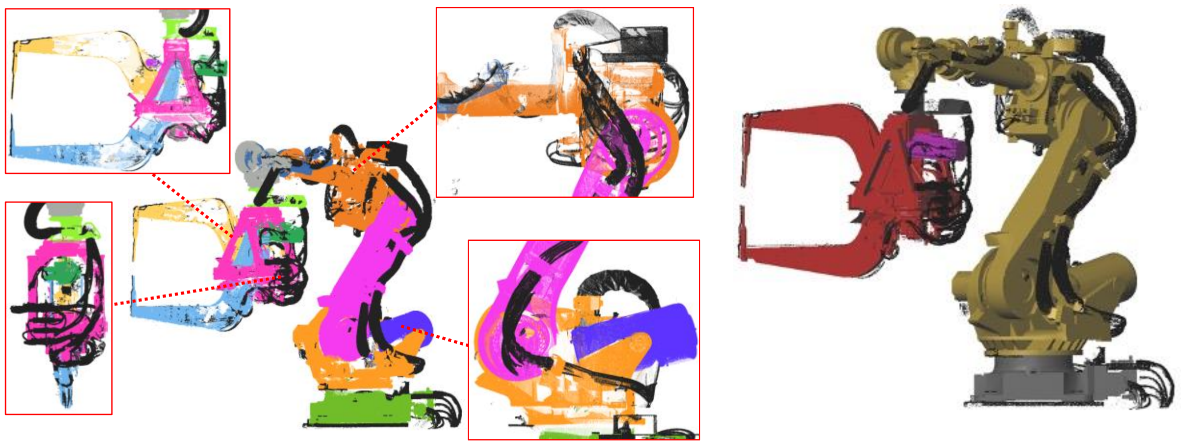

(a) (b)

Figure 15: Segmentation of TLS point cloud: (a) Segmented point cloud, (b) CAD models aligned to point cloud.

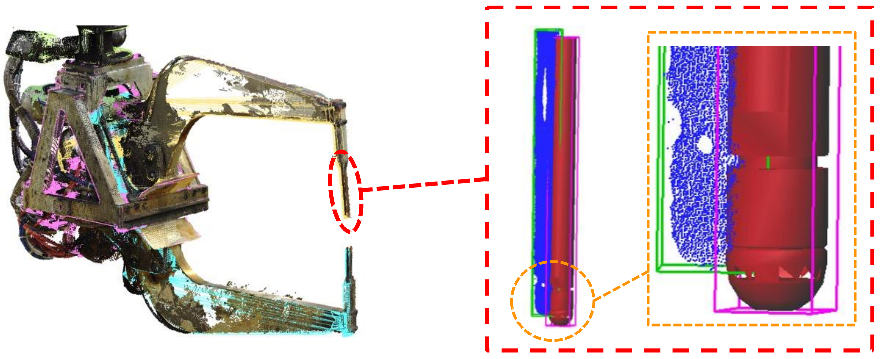


Figure 16: Accumulation of registration error at the tip of the end-effector.

7.2 Evaluation with Virtual TLS Point Clouds

It is difficult to quantitatively evaluate the accuracy of the positioning of robot links because the correct positions are not known. Therefore, we performed a quantitative evaluation by acquiring a point cloud from a CAD model using a virtual laser scanner and applying our method to the virtual point cloud. The virtual point cloud was captured from the robot with three different postures. In each posture, the assembly model of the robot was measured using a virtual laser scanner placed at six locations with equal angles on a circle with a radius of 5 m centered on the robot. In each measurement, 50 million virtual laser beams were irradiated, and the intersection points with the assembly model were calculated.

For the virtual point cloud, rough and precise registrations were performed using the same method as for the actual TLS point cloud. Table 1 shows the errors of rotation angles calculated using our method at different postures. These results show that rough registration could estimate nearly correct rotation angles, and constrained registration effectively improved the rotation angles to more correct angles. This experiment shows that our 2-step registration is effective in accurately aligning 3D models to a point cloud. By using the fitted CAD models, we segmented the virtual point clouds into individual links. Figure 17 shows that virtual point clouds were precisely segmented for robots in various postures.

Joint	*Posture-1* [deg]		*Posture-2* [deg]		*Posture-3* [deg]	
	Roughly	*Precisely*	*Roughly*	*Precisely*	*Roughly*	*Precisely*
1	0.5	0.002	-0.3	-0.068	-0.2	-0.022
2	-0.2	0.014	0.4	-0.032	0.3	0.014
3	-0.4	-0.058	-0.1	-0.086	-0.1	-0.037
4	0.3	0.212	0.5	0.297	-0.2	-0.131
5	-0.3	-0.241	-0.8	-0.189	-0.5	0.010
6	0.1	-0.002	-0.7	-0.097	0.3	0.139
7	0.2	0.193	-0.2	-0.243	-0.6	-0.539

Table 1: Fitting result of the difference between calculated and set values of rotation angle.

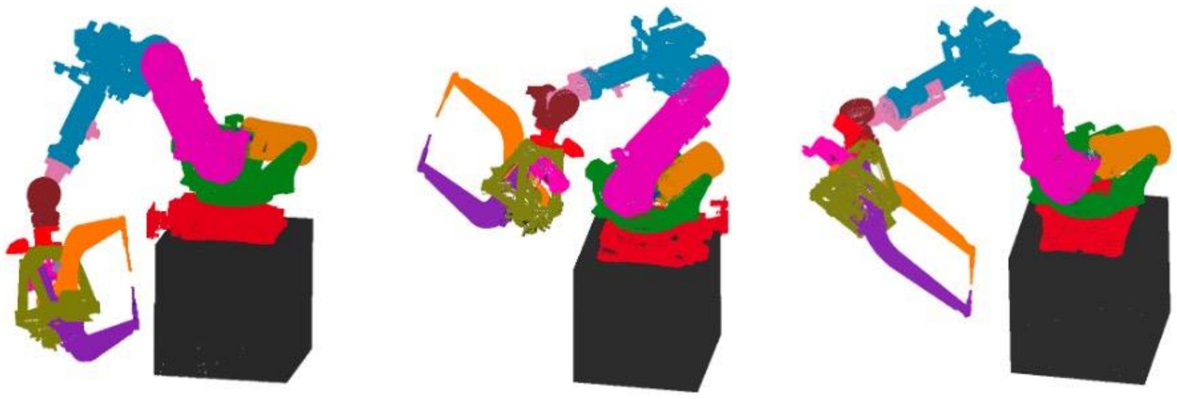


Figure 17: Segmentation of virtual point clouds into robot links.

8 CONCLUSIONS

This paper proposed a method to calculate the positions and postures of links of articulated robots with end-effectors from point clouds. The proposed method used the URDF for describing link mechanisms and fitted the 3D model of each robot link to a point cloud. Since the URDF was described using a combination of the main chain and sub chains, it could handle robot links with closed loops. For the main chain, the position and posture of each link were calculated sequentially from the base link, and for the sub chains, inverse kinematics was applied to determine their positions. By using the fitted 3D models, the point cloud of the robot could be segmented into individual links and attachments. Experimental results showed that our method could accurately determine the positions and postures of articulated robots, and point clouds of robots could be appropriately segmented.

In the future, we would also like to develop an integrated method that can extract and classify various types of robots, end effectors, and attachments from point clouds to support motion planning of articulated robots. We would also like to develop methods for fitting detailed link mechanisms of various types of end-effectors as well as a wide variety of robots by supporting cases where the main chain branches or contains prismatic joints. In order to apply our method to various types of robots, we would like to investigate ambiguous cases in determining the positions and orientations of links. Although rotational symmetry was discussed in this paper, other ambiguous situations may arise in actual production lines, depending on the relationship between links and the shape of the links. As shown in Figure 16, sequential registration causes misalignment in the end effector. We want to investigate more accurate fitting methods for articulated robots such as multiview registration. For improving accuracy, we might be able to use methods developed for human joints and other articulated objects.

Kota Kawasaki, https://orcid.org/0000-0001-5188-158X
Kakeru Takeda, https://orcid.org/0009-0003-1295-9796
Hiroshi Masuda, https://orcid.org/0000-0001-9521-6418

REFERENCES

[1] Arun, K. S.; Huang, T. S.; Blostein, S. D.: Least-Squares Fitting of Two 3-D Point Sets, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1987, PAMI-9(5), 698-700. https://doi.org/10.1109/TPAMI.1987.4767965

[2] Berger, M.; Tagliasacchi, A.; Seversky, L. M.; Alliez, P.; Guennebaud, G.; Levine, J. A.; Sharf, A.; Silva, C. T.: A Survey of Surface Reconstruction from Point Clouds, Computer Graphics Forum, 36(1), 2017, 301-329. https://doi.org/10.1111/cgf.12802

[3] Bey, A.; Chaine, R.; Marc, R.; Thibault, G.; Akkouche, S.: Reconstruction of Consistent 3D CAD Models from Point Cloud Data Using a Priori CAD Models, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-5/W12, 2011, 289-294. https://doi.org/10.5194/isprsarchives-XXXVIII-5-W12-289-2011

[4] Bouaziz S.; Tagliasacchi A.; Pauly M.: Sparse Iterative Closest Point, Computer Graphics Forum, 32(5), 2013, 113-123. https://doi.org/10.1111/cgf.12178

[5] Hu, S.; Polette, A.; Pernot, J.-P.: SMA-Net: Deep learning-based identification and fitting of CAD models from point clouds, Engineering with Computers, 38, 2022, 5467-5488. https://doi.org/10.1007/s00366-022-01648-z

[6] Ip, C. Y.; Gupta, S. K.: Retrieving Matching CAD Models by Using Partial 3D Point Clouds, Computer-Aided Design and Applications, 4(5), 2007, 629-638. https://doi.org/10.1080/16864360.2007.10738497

[7] Li, Y.; Wu, X.; Chrysathou, Y.; Sharf, A.; Cohen-Or, D.; Mitra, N. J.: GlobFit: Consistently Fitting Primitives by Discovering Global Relations, ACM Transactions on Graphics, 30(4), 2011, 1-12. https://doi.org/10.1145/2010324.1964947

[8] Masuda, H.; Niwa, T.; Tanaka, I.; Matsuoka, R.: Reconstruction of Polygonal Faces from Large-Scale Point-Clouds of Engineering Plants, Computer-Aided Design and Applications, 12(5), 2015, 555-563. https://doi.org/10.1080/16864360.2015.1014733

[9] Pulli, K.: Multiview Registration for Large Data Sets, IEEE Second International Conference on 3-D Digital Imaging and Modeling, October 1999, 160-168. https://doi.org/10.1109/IM.1999.805346

[10] Qi, C. R.; Su, H.; Mo, K.; Guibas, L. J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, July 2017, 77-85. https://doi.org/10.1109/CVPR.2017.16

[11] Qi, C. R.; Yi, L.; Su, H.; Guibas, L. J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, Conference on Neural Information Processing Systems, December 2017, 5105-5114. https://dl.acm.org/doi/10.5555/3295222.3295263

[12] Schnabel, R.; Wahl, R.; Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection, Computer Graphics Forum, 26(2), 2007, 214-226. https://doi.org/10.1111/j.1467-8659.2007.01016.x

[13] Shah, G. A.; Polette, A.; Pernot, J.-P.; Giannini F.; Monti M.: Simulated Annealing-Based Fitting of CAD Models to Point Clouds of Mechanical Parts' Assemblies, Engineering with Computers, 37, 2021, 2891-2909. https://doi.org/10.1007/s00366-020-00970-8

[14] Tola, D.; Corke, P.: Understanding URDF: A Survey Based on User Experience, International Conference on Automation Science and Engineering (CASE), IEEE, August 2023, 1-7. https://doi.org/10.1109/CASE56687.2023.10260660