# Decomposing the Configuration of an Articulated Object via Graph Neural Network

Seunghyeon Lim[1*], Kisung Shin[2*], Nakul Gopalan[3], Jun Ki Lee[4†], Byoung-Tak Zhang[1, 2, 4, 5†]

[1]Interdisciplinary Program in Cognitive Science, Seoul National University, Korea
[2]Interdisciplinary Program in Artificial Intelligence, Seoul National University, Korea
[3]School of Computing and Augmented Intelligence, Arizona State University, USA
[4]AIIS, Seoul National University, Korea
[5]Department of Computer Science and Engineering, Seoul National University, Korea
[*]Equal Contribution, [†]Co-corresponding authors
{shlim, ksshin, btzhang}@bi.snu.ac.kr, junkilee@snu.ac.kr, nakul.gopalan@asu.edu

**Abstract:** Configuring links, joints, and their combinations is a critical skill for robots to manipulate complex articulated objects. In this paper, we employ a graph-based representation to model an articulated object, where the parts (i.e., links) are treated as nodes and the joints connecting them as edges. We train a Graph Neural Network (GNN) to learn the language embedding of the individual parts, determine the connectivity between different links, and predict the joint state values and types. Our model demonstrates superior performance across four key metrics, highlighting its applicability to robotic manipulation tasks. Then, we conducted initial experiments on the effectiveness of using joint information that our model can provide in learning the manipulation skills for articulated objects and presented its results. This emphasizes the potential of our model to offer significant advancements in reinforcement learning for robotic manipulation in the near future.

**Keywords:** Articulated Objects Manipulation, Message Passing Neural Network, Language Embeddings

## 1 Introduction

Articulated objects are composed of multiple joints and links. Estimating the configuration of the articulated object is crucial for precise robotic object manipulation. Most of the research focuses on configuring articulated objects in a limited condition, such as a limited number of joints and links, or supports only a single type of joint [1, 2]. This limitation can make it difficult to generalize their model to real-world applications, which are filled with objects that have varying numbers and types of links and joints. To address this issue, we formulate an articulated object in a graph structure. Graph is one of the easiest and effective solutions to demonstrate the combinations of links and joints of an articulated object.

In this paper, we propose a graph neural network (GNN) designed to decompose the diverse information of articulated objects, which is essential for robotic manipulation. We utilize the Message Passing Neural Network (MPNN), a widely adopted framework in GNNs, which efficiently aggregates information from both nodes and edges to update node features through our edge update networks [3].

The model takes as input a 360-degree point cloud of a single articulated object along with class-agnostic part labels. It then outputs four types of information: 1) language embeddings of part names, 2) confidence scores indicating the possibilities that two parts are actually connected, 3) probabilities for joint types (i.e., prismatic or revolute), and 4) joint state values, which describe the current configuration or position of each joint, scaled between 0 and 1. The details of the model output are described in Figure 1.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

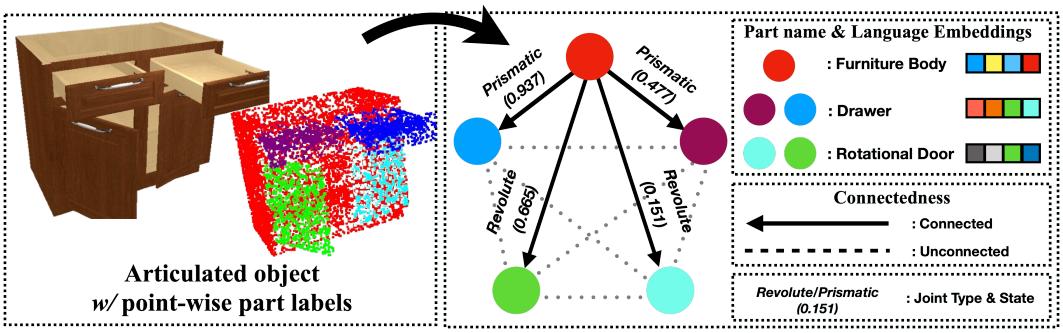


Figure 1: **An example of decomposing the configuration of an articulated object.** An articulated object can be decomposed into parts and joints. Our model represents the object as a graph and outputs: 1) language embeddings of part names, 2) confidence scores (i.e., connectedness), 3) joint types, and 4) joint state values scaled between $[0, 1]$.

In this experiment, we evaluate our proposed model on a range of articulated object instances with different joint configurations of a single category. We evaluated the performance of the model on all four criteria mentioned above with accuracy metrics exceeding 90% on several measures, demonstrating outstanding quantitative results. Furthermore, we train a robot arm agent with the SAC algorithm [4] within a simulation environment [5, 6]. We incorporate joint state information of the object into the observation inputs and employ it for reward shaping. This approach aims to demonstrate our insight that our proposed model can significantly enhance the performance of robotic manipulation tasks in future work, particularly for real-world applications.

## 2 Graph Neural Network for Articulated Object Decomposition

The model takes as input a 360-degree point cloud view of an articulated object, along with point-wise class-agnostic part labels that represent the object's structure. The model outputs the following:

- A **language embedding** for the name of each part of the articulated object.
- A **joint confidence score** representing the probability that two parts (i.e. links) are connected by a joint.
- **Joint type probabilities** indicating whether a specific joint is prismatic or revolute.
- **Joint state values** representing the positional data of a joint, which describe its current orientation or displacement within its allowable range of motion, scaled between 0 and 1.

Figure 2 illustrates the overall framework of our model. First, our model extracts shape embeddings for each part using a discrete Variational Autoencoder (dVAE) [7], following the approach outlined in Point-BERT [8]. The detailed procedure for extracting partwise shape embeddings, $\{\psi^{p_k}\}_{k=1}^{\mathcal{G}}$, for $\mathcal{G}$ parts $\{p_k\}_{k=1}^{\mathcal{G}}$ from Point-BERT embeddings $\{\phi^{g_j}\}_{j=1}^{G}$ of $G$ groups $\{g_j\}_{j=1}^{G}$ can be found in Appendix B.

We utilize MPNN to obtain language embeddings for each part name and information vectors for each joint between two connected parts. The shape embeddings for each part, denoted as $\{\psi^{p_k}\}_{k=1}^{\mathcal{G}}$ are obtained from a dVAE model and serve as the initial node features $\{X_i\}_{i=1}^{\mathcal{G}}$, while the edge features are initialized as zero vectors. Each joint information vector has five dimensions, representing the confidence score, the probabilities of joint type, and the values of the joint state, as detailed in Figure 2. We denote the joint information vector connecting nodes $i$ and $j$ as $\chi_{ij}$.

For message passing, node features and adjacent edge features are utilized to update node features. We can use the structure of the MPNN model as [9, 10, 11]. Additionally, we design an edge update network comprising three linear layers, each followed by a batch normalization and an activation function. The MPNN is used to update the node features $\{X_i\}_{i=1}^{\mathcal{G}}$ by aggregating information from neighboring nodes and connected edges. Subsequently, the edge update network is applied to update the edges $E = \{\chi_{ij} | 1 <= i, j <= \mathcal{G},$ if parts $p_i, p_j$ are connected$\}$.

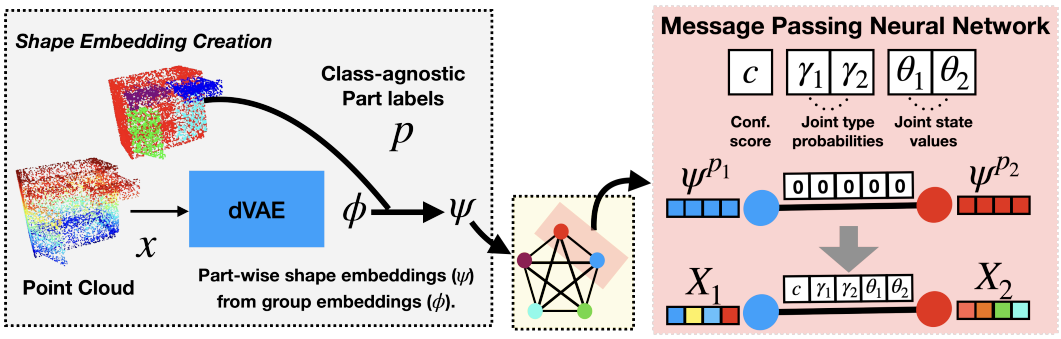


Figure 2: **Overall framework of our model.** Our model processes a 360-degree point cloud of an articulated object using the discrete Variational Autoencoder (dVAE) from Point-BERT to generate part-wise shape embeddings, $\psi$. These embeddings serve as initial node features in the graph, with edge features initialized as zero vectors. After the Message Passing Neural Network (MPNN), node features $X$ are transformed into a language embedding for part names and a joint information vector.

To train MPNN, we first freeze the weight of dVAE which is trained in advance following the method in Point-BERT. After MPNN, the node features $\{X_i\}_{i=1}^{\mathcal{G}}$ are transformed into language embeddings for each part $p_i$, denoted as $X^{p_i}$.. We employ an L2 loss function to train the node features, which is formulated as $L_{lang} = \frac{1}{\mathcal{G}} \sum_{i=1}^{\mathcal{G}} ||X_i - X^{p_i}||_2$.

The five-dimensional joint information vector $\chi_{ij}$ is decomposed into the confidence score $c_{ij}$, the probabilities of the joint type $\boldsymbol{\gamma}_{ij}$, and the values of the joint state $\boldsymbol{\theta}_{ij}$. Unlike the confidence score $c_{ij} \in \mathbb{R}$, the vectors $\boldsymbol{\gamma}_{ij}, \boldsymbol{\theta}_{ij} \in \mathbb{R}^2$ represent the probabilities that the joint is prismatic or revolute and the joint states for the prismatic and revolute joints, respectively. The corresponding ground truth for the joint information is denoted as $c'_{ij}$, $\boldsymbol{\gamma}'_{ij}$, and $\boldsymbol{\theta}'_{ij}$. The confidence score loss function $L_{conf}$ is defined as a binary cross entropy function as $L_{conf} = \frac{1}{|E|} \sum_{\chi_{ij} \in E} \text{BCELoss}(c_{ij}, c'_{ij})$.

The model update joint-type probabilities $\boldsymbol{\gamma}_{ij}$ uses a cross-entropy loss, conditioned on $\boldsymbol{\gamma}'_{ij} \neq \mathbf{0}$ (i.e., an edge with a valid joint type) as follows:

$$L_{type} = \frac{1}{\sum_{\chi_{ij} \in E} \mathbb{1}(\boldsymbol{\gamma}'_{ij} \neq \mathbf{0})} \sum_{\chi_{ij} \in E} \mathbb{1}(\boldsymbol{\gamma}'_{ij} \neq \mathbf{0}) \cdot \text{CELoss}(\boldsymbol{\gamma}_{ij}, \boldsymbol{\gamma}'_{ij}) \tag{1}$$

Finally, the joint state loss is defined using an L2 loss. This loss is computed only for joints with a finite state range, with joints of infinite state ranges masked by $\mathcal{M}_{state}$ and conditioned on $\boldsymbol{\gamma}'_{ij} \neq \mathbf{0}$:

$$L_{joint\_state} = \frac{1}{\sum_{\chi_{ij} \in E} \mathbb{1}(\boldsymbol{\gamma}'_{ij} \neq \mathbf{0}) \cdot \mathcal{M}_{state}} \sum_{\chi_{ij} \in E} \mathbb{1}(\boldsymbol{\gamma}'_{ij} \neq \mathbf{0}) \cdot \mathcal{M}_{state} \cdot ||\boldsymbol{\theta}_{ij} - \boldsymbol{\theta}'_{ij}||_2) \tag{2}$$

The total loss $L$ is then defined as $L = L_{lang} + L_{conf} + L_{type} + L_{joint\_state}$.

## 3 Experimental Results

In this section, we evaluate the performance of our proposed model in interpreting complex joint configurations of articulated objects. The experimental setup and dataset details referred to in Section 3.1 are provided in Appendix D. Also, the setup for the initial experiment conducted in the simulation environment (Section 3.2) is in Appendix E.

### 3.1 Inference on Complex Joint Configurations

In this experiment, we measured the accuracy of the predicted language embeddings by matching them to the nearest ground-truth embeddings from the part name list of each object instance. Furthermore, we derived the predicted confidence scores and joint types from the joint information vectors and calculated their respective accuracies. Furthermore, we observed errors in the joint states by scaling the predicted joint state values according to the actual range of the joints provided.

Table 1: Overall performance of our models with varying MPNN iterations (Iteration) on articulated object instances with complex joint configurations.

| Iteration | Lang. acc. | Joint conf. acc. | Joint type acc. | Rev. err. (°) ↓ | Pris. err. (cm) ↓ |
|---|---|---|---|---|---|
| 1 | 0.985 | 0.999 | 0.930 | **15.942** | **14.281** |
| 3 | 0.986 | 0.999 | **0.939** | 16.057 | 14.323 |
| 5 | **0.987** | 0.999 | 0.930 | 16.195 | 14.282 |

Table 1 presents the performance of our models in all four outputs, comparing different numbers of MPNN iterations. This experiment involves objects with varying joint types and a diverse number of parts. We conducted our experiments on the *StorageFurniture* category, which consists of objects with complex part structures and connectivity. Specifically, the average number of parts in the test data set is 3.149, while the average number of prismatic and revolute joints is 1.148 and 1.000, respectively.

Our models achieved a better accuracy 90% in predicting language embeddings, joint confidences, and joint types. This demonstrates the effectiveness of the MPNN in aggregating information for articulated objects. Moreover, our model offers a significant advantage over recent models for articulated object inference [1, 2, 12, 13], which lack the ability to handle objects with varying connectivity and joint types in a single model or are unable to infer part names, an essential feature for enabling human-robot interaction in real-world applications. Finally, our models exhibit consistent performance across different MPNN iterations for joint state errors. As demonstrated in Appendix C, our model achieves joint-state error rates that are on par with the state-of-the-art approach presented by [1], despite of its broader applicability beyond joint state error inference.

### 3.2 Initial Experiment for Learning Manipulation Skill

We conducted reinforcement learning (RL) experiments in a simulation environment using the Robosuite framework [6]. In this experiment, the goal of the SAC agent is to manipulate the Franka-Emika-Panda robot arm to rotate the handle of the door sufficiently to release the latch and open the door so that the angle of the hinge of the door becomes a specific angle. Please note that the information used for observations and reward shaping is not obtained through our proposed model. The mean reward curves during training are shown in Figure E.2.

Table 2: Performance of reinforcement learning agent in door opening task using Soft Actor-Critic algorithm (SAC). We compared the model using the provided joint state values as observations (Obs.) and reward shaping (Reward).

| Setting | Prop. | Obs. | | Reward | | Success Rate (%) |
|---|---|---|---|---|---|---|
| | | Handle | Hinge | Handle | Hinge | |
| *w/o Arti-info* | ✓ | × | × | × | × | 0 |
| *w/ Arti-info* | ✓ | ✓ | ✓ | ✓ | ✓ | 49.4 |

Our experimental results are summarized in Table 2. Both settings incorporate proprioceptive information (Prop.) as part of the observations, which includes the 7-DoF joint positions of the robot, the joint positions of the gripper, and the position and orientation of the end effector. The agent utilizing articulated object information (*w/ Arti-info*) has a 49.4% higher average task success rate than the agent without the information (*w/o Arti-info*).

## 4 Discussion

Considering that the ground-truth information for articulated objects is not inherently available in the real world, the results of this experiment demonstrate that the configurations of articulated objects generated by our proposed model can significantly enhance the manipulation capabilities of real-world robotic agents. Although not addressed in the initial experiment in Section 3.2, the ability of our proposed model to extract language embeddings and estimate part labels represents a significant advantage. In particular, combining our model with large language model can extend the robot to learn more diverse and general manipulation skills.

**Acknowledgments**

This work was partly supported by the IITP (RS-2021-II212068-AIHub/10%, RS-2021-II211343-GSAI/15%, 2022-0-00951-LBA/15%, 2022-0-00953-PICA/20%), NRF (RS-2024-00353991/20%, RS-2023-00274280/10%), and KEIT (RS-2024-00423940/10%) grant funded by the Korean government. Also, this work was supported by Artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT(MSIT, Korea)&Gwangju Metropolitan City.

## References

[1] J. Mu, W. Qiu, A. Kortylewski, A. Yuille, N. Vasconcelos, and X. Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.

[2] N. Heppert, M. Z. Irshad, S. Zakharov, K. Liu, R. A. Ambrus, J. Bohg, A. Valada, and T. Kollar. Carto: Category and joint agnostic reconstruction of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21201–21210, 2023.

[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[5] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[6] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[7] J. T. Rolfe. Discrete variational autoencoders. *arXiv preprint arXiv:1609.02200*, 2016.

[8] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ. accepted as poster.

[10] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

[11] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[12] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13670–13677. IEEE, 2021.

[13] Q. Yu, J. Wang, W. Liu, C. Hao, L. Liu, L. Shao, W. Wang, and C. Lu. Gamma: Generalizable articulation modeling and manipulation for articulated objects. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5419–5426. IEEE, 2024.

[14] B. Eisner, H. Zhang, and D. Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. *arXiv preprint arXiv:2205.04382*, 2022.

[15] V. Zeng, T. E. Lee, J. Liang, and O. Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE, 2021.

[16] A. Jain, S. Giguere, R. Lioutikov, and S. Niekum. Distributional depth-based estimation of object articulation models. In *Conference on Robot Learning*, pages 1611–1621. PMLR, 2022.

[17] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6813–6823, 2021.

[18] C. Ning, R. Wu, H. Lu, K. Mo, and H. Dong. Where2explore: Few-shot affordance learning for unseen novel categories of articulated objects. *arXiv preprint arXiv:2309.07473*, 2023.

[19] R. Wu, K. Cheng, Y. Zhao, C. Ning, G. Zhan, and H. Dong. Learning environment-aware affordance for 3d articulated object manipulation under occlusions. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] N. Nie, S. Y. Gadre, K. Ehsani, and S. Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arXiv preprint arXiv:2207.08997*, 2022.

[21] S. Patki and T. M. Howard. Language-guided adaptive perception for efficient grounded communication with robotic manipulators in cluttered environments. In *19th Annual SIGdial Meeting on Discourse and Dialogue*, 2018.

[22] E. Fahnestock, S. Patki, and T. M. Howard. Language-guided adaptive perception with hierarchical symbolic representations for mobile manipulators. *arXiv preprint arXiv:1909.09880*, 2019.

[23] T. M. Howard, S. Tellex, and N. Roy. A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659. IEEE, 2014.

[24] M. Liu, Y. Zhu, H. Cai, S. Han, Z. Ling, F. Porikli, and H. Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21736–21746, 2023.

[25] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[26] J. Lei, C. Deng, W. B. Shen, L. J. Guibas, and K. Daniilidis. Nap: Neural 3d articulated object prior. *Advances in Neural Information Processing Systems*, 36:31878–31894, 2023.

[27] J. Liu, H. I. I. Tam, A. Mahdavi-Amiri, and M. Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17880–17889, 2024.

[28] H. Abdul-Rashid, M. Freeman, B. Abbatematteo, G. Konidaris, and D. Ritchie. Learning to infer kinematic hierarchies for novel object instances. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8461–8467. IEEE, 2022.

[29] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[30] X. Li and J. Li. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*, 2023.

[31] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[32] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

# A Related Works

## A.1 Estimating a Configuration of an Articulated Object

Numerous methods have been proposed for estimating the configuration of articulated objects. [14] learns 3D articulation flow from point clouds, facilitating robotic manipulation of objects. [15] utilizes RGB-D and object part segmentation images to estimate node types and part connectivity, aiming to classify the articulation type (i.e., prismatic or revolute). [12, 16] leverage effective representations, such as screw theory and Stiefel manifolds, to model the configuration of articulated objects. [17, 18, 19] focus on estimating the affordances for manipulating articulated objects. [13, 20] propose learning the joint parameters through robot interaction with articulated objects.

Despite the strengths of previous work, many do not fully encompass the range of articulation information that our model outputs: part name embeddings, joint connectivity confidence, joint types, and joint position states. In particular, our model's ability to generate part-name embeddings offers a significant advantage in deducing the configuration of articulated objects. [21, 22] employ Distributed Correspondance Graph (DCG) [23] to parse narrations and derive language-based inferences in robotic tasks. However, our model is more suitable for application to articulated objects with complex relationships between parts and joints.

PartSLIP [24] utilizes a 360-degree point cloud of an object to ground the language into its constituent parts. In contrast, our model primarily focuses on the relationships between the parts of articulated objects and joint configuration, grounding the language using a provided list of part names. Although our model requires class-agnostic part labels as input, these can be easily obtained through image segmentation models such as Segment Anything [25].

## A.2 Graph Structure for the Configuration of an Articulated Object

In recent years, many studies have leveraged graphs to model articulated objects. In particular, cutting-edge works like [26, 27] utilize graph structures for object generation tasks, demonstrating that such structures can effectively encode the information necessary for successful object generation. Similarly, our model harnesses the power of graphs through the use of a Message Passing Neural Network (MPNN), which outputs critical information for articulated object manipulation tasks.

[28] employs a Graph Neural Network (GNN) to infer the kinematic hierarchies of articulated objects, predicting joint types and the existence of edges. Like our approach, they represent object parts as nodes and joints as edges. However, our model extends this by also outputting joint states and part names, in addition to joint type and edge existence. These added features make our model particularly applicable to real-world tasks requiring precise joint state configurations for robotic perception and high-level reasoning (e.g., language understanding).

# B Obtain Shape Embeddings from a discrete Variational Autoencoder

We utilize a Discrete Variational Autoencoder (dVAE) to obtain shape embeddings of object parts, which will later serve as node features of the graph at initialization. Point clouds are first processed through the dVAE in Point-BERT, which outputs the shape embedding for each local group. The dVAE at Point-BERT divides $N$ points $\{x_i\}_{i=1}^{N}$ into $G$ groups $\{g_j\}_{j=1}^{G}$, with each group containing $H$ points (i.e., $N = G \times H$). Each group is represented by an embedding generated by the dVAE, denoted as $\{\phi^{g_j}\}_{j=1}^{G}$.The $i$th point of the $j$th group of the point cloud can be denoted as $x_i^{g_j}$. Thus, the point cloud can be represented as $\{x_i^{g_j}\}_{i=1}^{H}$, for $j = 1, 2, \ldots, G$. Simultaneously, points from the point cloud $\{x_i\}_{i=1}^{N}$, sharing the same part label $\{p_k\}_{k=1}^{\mathcal{G}}$ can be denoted as $\{x_i^{p_k}\}_{i=1}^{\mathcal{H}}$ where $\mathcal{G}$ represents the total number of part labels and $\mathcal{H}$ the number of points for a given part label.

To construct part-wise shape embeddings $\{\psi^{p_k}\}_{k=1}^{\mathcal{G}}$ from the group embeddings $\{\phi^{g_j}\}_{j=1}^{G}$, we first calculate the chamfer distance between the part labels and the group embeddings. This will be

referred to as the chamfer distance matrix $M$. Next, we take the reciprocal of $M$ and apply the softmax function. The value for the $j$th group of the point cloud and the $k$th part of the original point cloud is given by the following:

$$M_{jk} = \text{softmax}\left(\frac{1}{\text{ChamferDist}\left(\{x_i^{g_j}\}_{i=1}^{H}, \{x_i^{p_k}\}_{i=1}^{\mathcal{H}}\right) + \epsilon}\right) \tag{3}$$

Finally, we obtain the shape embeddings of the $k$th part of the point cloud by multiplying the chamfer distance matrix with group embeddings. This process can be simply written as $\psi^{p_k} = \sum_{j=1}^{G} M_{jk} \cdot \phi^{g_j}$.

## C Joint state values Inference

In this experiment, we compare our model's ability to infer joint state values with a decoder of A-SDF, a recent method for obtaining joint state. Before examining the results, we would like to emphasize that A-SDF requires a fixed joint number to be set during each training instance, which contrasts with the flexibility of our model. Furthermore, A-SDF cannot infer the connectivity between parts. These limitations reduce the generalizability of the model, which is crucial for robots interacting with objects with varying numbers and types of joints. Additionally, A-SDF requires further optimization of shape codes and joint angles during inference, which increases its runtime and renders it impractical for dynamic robotic environments.

We utilize A-SDF for both prismatic and revolute joint inference, referencing the implementation from [2]. For fairness, since A-SDF does not require the actual state range to infer the joint state, we clamp its predictions to the minimum and maximum values of the object instance.

Table 3: Joint state errors for articulated objects with varying joint configurations. Results are compared across different iterations (#It.) of the MPNN. The units for revolute errors and prismatic errors are degrees (◦) and centimeters (cm), respectively.

| Method | #It. | Double Revolute ↓ | | Single Revolute ↓ | | Single Prismatic ↓ | |
|---|---|---|---|---|---|---|---|
| | | *StorageFurniture* | *Eyeglasses* | *Door* | *Laptop* | *TrashCan* | *Table* |
| A-SDF [1] | - | **11.762** | **8.771** | **6.543** | 18.298 | 12.973 | **9.788** |
| Ours | 1 | 12.412 | 10.678 | 6.653 | 18.227 | **5.921** | 10.766 |
| Ours | 3 | 12.375 | 10.617 | 6.835 | 17.749 | 5.980 | 10.724 |
| Ours | 5 | 12.516 | 10.631 | 7.042 | **17.395** | 6.108 | 11.147 |

Table 3 demonstrates the joint state errors between the predicted and ground truth values. We grouped the data set based on the number and type of joints into six categories for comparison with A-SDF. Both models are trained on a per-category basis. As shown in the table, our model achieves a performance comparable to that of A-SDF **despite the advantages for robotic applications**. For objects with revolute joints, both models exhibit less than a difference $2^\circ$ in all categories. For objects with prismatic joints, our model outperforms the baseline by 7cm in the *TrashCan* category, while showing slightly inferior performance in the *Table* category.

We also investigated the performance of our model with varying numbers of iterations for the MPNN and the subsequent edge update network. We found that the optimal number of iterations varied between categories; however, the difference in performance within a single category was less than $1^\circ$ (or cm), indicating a minimal impact on overall performance.

## D Experimental Setup and Datasets

We conducted experiments using the PartNet-Mobility dataset, which contains 46 categories of articulated objects [29]. For each object, we capture 360-degree point clouds across various joint states. The data set comprises 2,312 object instances, which we split into training and test sets. First, we

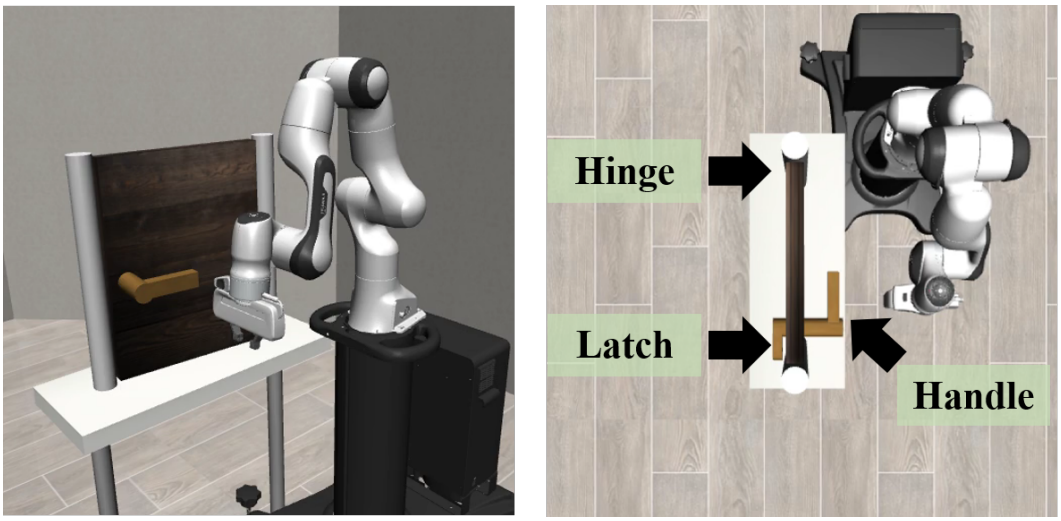


Figure E.1: **Simulation environment for reinforcement learning experiment.** Scene capture and detailed configuration of door object in the simulator for the door opening task. The door object includes the handle, latch, and hinge.

train a dVAE using 10 poses with different joint configurations per instance. This enables us to obtain 768-dimensional shape embeddings $\{\psi^{p_k}\}_{k=1}^{\mathcal{G}}$ for each part of the articulated objects. We acquire ground truth language embeddings $\{X^{p_i}\}_{i=1}^{\mathcal{G}}$ using a pre-trained BERT model, optimized following the approach of [30].

Unlike dVAE, the MPNN is trained per object category, selecting object instances depending on the specific experiment. We utilize GINEConv [10], implemented via [31]. MPNN training is conducted using 100 poses with different joint configurations for each instance.

## E Details of the RL Experiment

**Door Opening Task** Figure E.1 illustrates the simulation environment within the Mujoco-based Robosuite framework. The objective of this experiment is to control a Franka-Emika-Panda robot arm to adjust the position of the door hinge to 0.3 radians. The door is equipped with two revolute joints: the hinge and the handle. The robot's goal is to open the door by rotating the handle to unlock the latch located at the rear of the handle.

**Reward Function** Both settings in Table 2 yield a reward value of 1 if the position of the door hinge exceeds 0.3 radians. We also receive a reward of 0.25 for both handle and hinge positions in the *w/ Arti-info* setting.

$$R_{handle} = \min\left(0.25, 0.25 \times \left|\frac{q_{handle}}{0.5\pi}\right|\right)$$

$$R_{hinge} = \min\left(0.25, 0.25 \times \left|\frac{q_{hinge}}{0.5\pi}\right|\right)$$

$$R_{total} = \begin{cases} 1 & \text{if } q_{hinge} > 0.3\,rad \\ R_{handle} + R_{hinge} & \text{else} \end{cases}$$

Here, $R_{handle}$ represents the handle reward, $R_{hinge}$ represents the hinge reward, $q_{handle}$ denotes the handle position value, $q_{hinge}$ denotes the handle position value, and $R_{total}$ indicates the total reward.

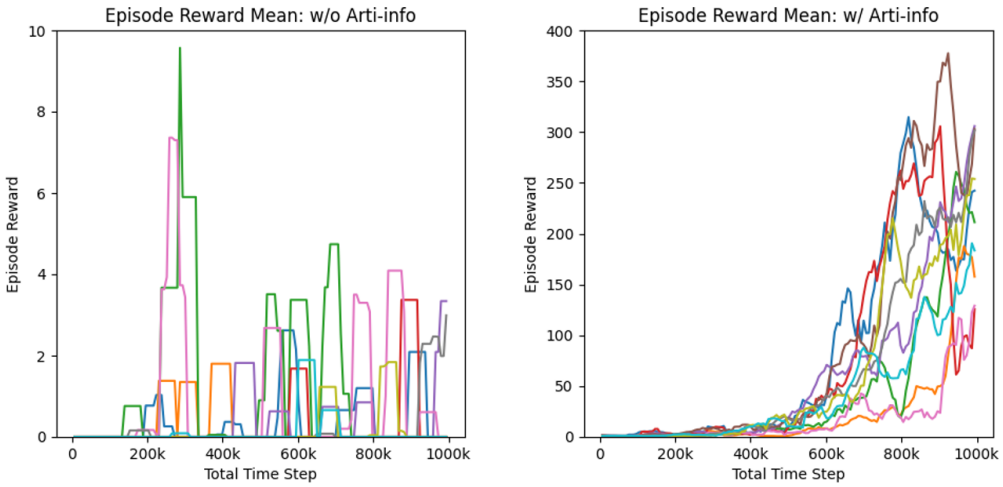


Figure E.2: **Mean of episode rewards in different reinforcement learning settings.** Plot of mean episode rewards of with and without *Arti-info* settings during training. Each line color represents a different random seed used for training.

**Training & Evaluation** Agents are implemented using the Stable-Baselines3 [32] packages. The actor and policy networks of the model architecture are constructed with two-layer MLPs, trained using the Adam optimizer. Each layer consists of 256 dimensions. The total number of samples for training and replay buffer size is set to 1 milion with a learning rate of 0.00075 and a discount factor of 0.99. We trained for 2,000 episodes using 14 processors, with each episode consisting of 500 steps. During the evaluation, we randomize the position and orientation of a door object within 2 cm and 0.25 radians, respectively. We calculate the mean success rate by measuring the number of successful trials divided by the number of total trials (i.e., 100 trials).