DiffAssemble: A Unified Graph-Diffusion Model for 2D and 3D Reassembly

Supplementary Material

A. Experiment Details

Hardware. The experiments were conducted on 2 different machines: four NVIDIA Tesla V100 16GB, 380 GB RAM, and 2x Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz Sky Lake CPU, and one NVIDIA RTX 4090 GPU, 64 GB RAM, and 12th Gen Intel(R) Core(TM) i9-12900KF CPU @ 3.20GHz CPU.

Model Settings. We train DiffAssemble with a learning rate of 10^{-4} and Adagrad as the optimization algorithm [11]. During our training process, we set a maximum of 1000 epochs, but we stop the training earlier to prevent unnecessary iterations when the loss no longer decreases.

B. Equivariant Feature Representation

As we presented in Section 3.2, one of key point of our proposal lies in its ability to work with element features $\mathbf{h}^m$, which can be extracted by any pre-trained encoders. In particular, we discover the importance to extract rotation-equivariant features.

A function ϕ is equivariant to the action of a group G if $\phi(S_g(\cdot)) = S_g^{'}(\phi(\cdot))$ for all $g \in G$, where S_g and $S_g^{'}$ are linear representations related to the group element g [38]. This means that applying ϕ to the codomain of $S_g(\cdot)$ is equivalent to applying $S_g^{'} \in G$ to the codomain of ϕ. In this work, the transformation S_g and $S_g^{'}$ are rotations. As a result, the equivariant function $\phi(\cdot)$, i.e. the backbone, ensures the consistency of the rotational effect irrespective of whether it is applied before or after the function. Consequently, DiffAssemble associates a specific rotation $\mathbf{r}^m$ (in the input space) to the features vector $\mathbf{h}^m$.

C. Diffusion process and Rotation in 3D

We provide a more detailed description of how we introduce Gaussian Noise with 3D rotations. Following [25], we use a specific procedure to scale the rotation matrices $f_r(\mathbf{r}_t^m)$ by *i)* converting the rotation matrix to values in the Lie algebra $\mathfrak{so}(3)$, *ii)* multiplying them element-wise with t-dependent scalars, and *iii)* converting back to a rotation matrix through matrix exponentiation. Analogous to an addition in Euclidean space, the composition of rotations is done through matrix multiplication in $SO(3)$ as:

$$\lambda(\gamma_t, \mathbf{r}_t^m) = \exp(\gamma_t \log(f_r(\mathbf{r}_t^m))),$$

where $\lambda(\dots)$ is the geodesic distance flow from $\mathbf{I}$, the identity matrix, to $\mathbf{r}_t^m$ by an amount γ_t.

In particular, for the Forward Process, we rewrite Equation (1) to inject noise into $\mathbf{r}_0^m$:

$$q(\mathbf{r}_t^m | \mathbf{r}_0^m) = IG_{\text{SO(3)}}(\lambda(\sqrt{\overline{\alpha}_t}, \mathbf{r}_0^m), (1 - \overline{\alpha}_t)),$$

where $IG_{\text{SO(3)}}$ is the isotropic Gaussian distribution (IG) that is compatible with $SO(3)$ rotation directly. The IG distribution is parameterized in an axis-angle form by sampling uniformly an axis and rotation angle $\omega \in [0, \pi]$ as:

$$f(\omega) = \frac{1 - \cos\omega}{\pi} \sum_{l=0}^{\infty} (2l+1) e^{-l(l+1)\epsilon^2} \frac{\sin((l+0.5)\omega}{\sin(\omega/2)}.$$

For the Reverse Process, letting $R_t = \{\mathbf{r}_t^m\}_{m \in [1,\cdots,M]}$ and $H = \{\mathbf{h}^m\}_{m \in [1,\cdots,M]}$, we rewrite Equation (2) as follows:

$$\hat{R}_{t-1} = \lambda\left(\frac{\sqrt{\overline{\alpha}_{t-1}}}{\alpha_t}, R_t\right) \lambda\left(\frac{1 - \overline{\alpha}_{t-1}}{\sqrt{\overline{\alpha}_t}}, \epsilon_\theta^{\text{rot}}(R_t, t, H)\right)^T,$$

where $\epsilon_\theta^{\text{rot}}(R_t, t, H)$ is the estimated noise that has to be removed from R_t to recover $\hat{R}_{t-1}$.

D. Additional Ablations

Missing Fragments in 3D Objects Reassembly We assess the performance of DiffAssemble and the baselines in scenarios involving missing 3D pieces. We consider a setting where each object is composed of 10 to 20 parts. We test the methods in four different scenarios: *i)* without missing pieces, *ii)* 10% of missing pieces, *iii)* 20% missing pieces, and *iv)* 30% of missing pieces. We do not retrain the models with missing pieces, but instead, we use the same method and weights as in the main paper experiment described in Section 4.1. To account for potential variations in fracture sizes within each object, we report the experiment five times using different seeds. This methodology helps alleviate potential biases introduced by excluding fractures with differing levels of complexity. Mean and standard deviation for each metric provide an indication of the overall behavior of the compared methods.

Table 2 reports the results, demonstrating that in all four scenarios, DiffAssemble outperforms the baseline in 2 out of 3 metrics. There is a decrease in performance when we increase the number of missing pieces, even if this reduction is minimal.

2D Jigsaw Puzzle. Table 6 reports further ablation results for the puzzle setting, which we could not include in the main paper due to space constraints.

Missing	0%			10%		
Method	RMSE (R) ↓ degree	RMSE (T) ↓ $\times 10^{-2}$	PA ↑ %	RMSE (R) ↓ degree	RMSE (T) ↓ $\times 10^{-2}$	PA ↑ %
Global	83.00	18.74	7.02	83.86	<u>18.76</u>	6.78
DGL	84.56	**18.26**	<u>9.72</u>	84.74	18.98	<u>8.42</u>
LSTM	88.26	19.64	4.78	88.40	19.74	4.96
SE(3)-Equiv	<u>81.82</u>	<u>18.50</u>	6.74	<u>82.96</u>	**18.54**	6.58
DiffAssemble	**80.13**	19.02	**11.61**	**80.32**	19.32	**11.20**

Missing	20%			30%		
Method	RMSE (R) ↓ degree	RMSE (T) ↓ $\times 10^{-2}$	PA ↑ %	RMSE (R) ↓ degree	RMSE (T) ↓ $\times 10^{-2}$	PA ↑ %
Global	84.20	<u>18.86</u>	6.66	84.76	**18.96**	<u>6.62</u>
DGL	85.01	19.80	<u>7.34</u>	85.64	20.68	6.56
LSTM	88.72	19.90	4.88	88.96	20.01	4.36
SE(3)-Equiv	<u>82.52</u>	**18.72**	6.54	<u>82.88</u>	<u>19.48</u>	6.51
DiffAssemble	**80.37**	19.52	**10.67**	**80.46**	19.84	**10.43**

Table 5. Results for DiffAssemble on BB's objects with 8-20 pieces when 0%/10%/20%/30% of the pieces are missing pieces. Our approach is robust even in the hardest scenario where 30% of the pieces are missing.

STAGE	CHANGES	PuzzleCelebA				PuzzleWikiArts			
		6x6	**8x8**	**10x10**	**12x12**	**6x6**	**8x8**	**10x10**	**12x12**
Representation	Non-Equivariant Enc.	96.12	71.62	91.98	64.15	25.31	14.63	8.19	4.96
	Invariant Enc.	22.97	20.01	16.87	13.63	7.64	4.64	2.79	1.66
Diff. Process	No Diff. process	99.43	79.84	99.05	91.28	73.07	54.70	22.68	18.27
GNN	Standard GCN [24]	85.03	54.35	71.19	45.56	30.12	22.07	10.77	1.08
DiffAssemble	Base Implementation (Tab. 3)	**99.51**	**84.94**	**99.30**	**97.76**	**90.65**	**72.79**	**63.33**	**53.08**

Table 6. We conduct an ablation study to evaluate the impact of each component of DiffAssemble for Jigsaw puzzle solving on *PuzzleCelebA* and *PuzzleWikiArts*. The base implementation corresponds to our proposed approach, as reported in Table 3 of the main paper.

We assess the benefit of employing rotation-equivariant features, instead of invariant and non-equivariant ones. These two last representations lead to worse performance in both datasets. In particular, these differences are more evident with the WikiArt dataset. DiffAssemble obtains an average improvement of 94.41% and 82.43% compared to DiffAssemble with invariant and non-equivariant features. This result highlights, one more time, the importance of employing rotation-equivariant features to solve reassembly tasks when rotation is involved.

We aimed at demonstrating that the adoption of the diffusion process is well-founded and effective. For this reasons, we experiment DiffAssemble wihtout the diffusion process. The results show that predicting the pose without the diffusion process, i.e., in 1 step, leads to worst performance, which serves as strong justification for the inclusion and use of the diffusion process in our approach.

Finally, we conducted an ablation for the *GNN architecture* adopted in DiffAssemble. Specifically, we assess the Graph Convolutional Network (GCN) [24] against UniMP. The goal is to investigate the impact of the attention mechanism on information propagation. For this purpose, we define the adjacency matrix $A \in \mathcal{R}^{M \times M}$ of the GCN as an all-ones matrix. Tables 6 reports the results of this comparison in the 2D and 3D scenarios, respectively. DiffAssemble with the use of UniMP consistently outperforms DiffAssemble with GCN, showing a remarkable improvement. These results highlight the importance of employing a mechanism that can effectively capture relationships among nodes.

D.1. Effect of Edge Pruning

Figure 7 presents an ablation study on *PuzzleCelebA*, where we vary the pruning rate during training. Increasing the pruning, i.e., reducing the graph size, has a minor effect on the final results.

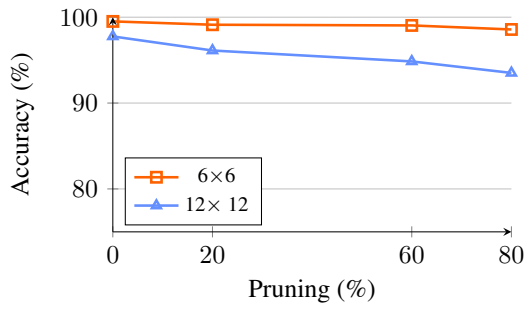

Figure 7. Ablation Sparse Attention Mechanism for Jigsaw puzzle solving on *PuzzleCelebA*.

E. Dataset Details

3D Reassembly Task. A 3D reassembly involves aligning fragments of a broken object into its original form, an essential task with applications in artifact preservation, digital heritage archiving, computer vision, robotics, and geometry processing. Despite its practical importance, the field has faced challenges due to the lack of suitable datasets for studying the natural fracture process. Existing datasets, such as PartNet [33], AutoMate [23], and JoinABLe [49], rely on semantic segmentation, failing to represent objects broken under natural, physically realistic conditions. Breaking Bad (BB) [37] fills this gap by simulating fractures using an algorithm that accounts for an object's most geometrically natural breaking patterns, thus creating a dataset that more realistically represents the challenges faced in fragments reassembly. **BB** contains approximately 10,000 meshes sourced from PartNet and Thingi10k. Each mesh includes 80 fractures, resulting in a total of 1,047,400 breakdown patterns. The dataset is divided into three subsets: *everyday*, *artifact*, and *other*. In this work, we focus on the *everyday* subset, as it is the commonly used dataset for evaluation in previous literature [50]. Qualitative examples can be found in the video attached to this Supplementary Material.

2D Reassembly Task. In this task, we evaluated DiffAssemble on two datasets: PuzzleCelebA and PuzzleWikiArts. Figure 8 shows some examples of inputs and reconstructions. More examples can be found in the video attached as Supplementary Material.

- *PuzzleCelebA* is based on CelebA-HQ [26] which contains 30K images of celebrities in High Definition (HD). Despite its superficial simplicity, this dataset poses significant challenges for puzzle-solving algorithms due to the inherent symmetry in human faces and often indistinct backgrounds. The dataset is divided in 80-20% train-test split, with 6,000 test puzzle permutations and randomly rotated patches.
- *PuzzleWikiArts* is based on WikiArts [46], and contains 63K images of paintings in HD. This dataset is particularly challenging due to very different content, artistic styles, and intricate patterns, which test the limits of puzzle-solving algorithms. The dataset is split into an 80-20% train-test ratio, resulting in 50k training images and 13k test puzzles across various grid sizes. It represents a more challenging dataset for puzzle solving as the paintings do not have a common pattern as in PuzzleCelebA (i.e. portraits).

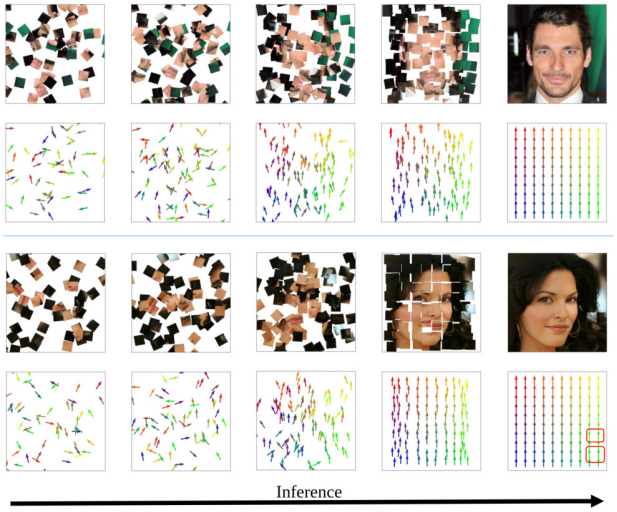

Figure 8. Qualitative results showing the diffusion process from random to solved puzzle. Each arrows correspond to one piece of the puzzle and its orientation indicate the orientation of the piece.

METHOD W/ % DEGREE	PuzzleCelebA							
	6x6	**8x8**	**10x10**	**12x12**	**14x14**	**16x16**	**18x18**	**20x20**
Degree 20%								
Classical dropout	91.60	57.08	82.32	50.18	74.43	25.40	61.45	28.35
Sparse Attention Mechanism	**92.37**	**59.45**	**87.67**	**54.07**	**83.11**	**31.07**	**73.88**	**32.97**
Degree 60%								
Classical dropout	**99.17**	72.93	**98.56**	94.07	98.53	46.48	**98.35**	92.51
Sparse Attention Mechanism	99.04	**73.91**	98.43	**94.35**	**98.70**	**48.26**	97.75	**93.29**
Degree 80%								
Classical dropout	99.15	76.45	98.75	95.87	98.58	51.51	98.14	94.93
Sparse Attention Mechanism	99.15	78.18	98.77	95.71	98.69	52.28	97.80	94.34

Table 7. Ablation Sparse Attention Mechanism for Jigsaw puzzle solving on *PuzzleCelebA*.