ReassembleNet: Learnable Keypoints and Diffusion for 2D Fresco Reconstruction

Adeela Islam[1,2] Stefano Fiorini[1] Stuart James[3] Pietro Morerio[1] Alessio Del Bue[1]
[1]Fondazione Istituto Italiano di Tecnologia [2]University of Genova [3]Durham University
{adeela.islam, stefano.fiorini}@iit.it

Abstract

The task of reassembly is a significant challenge across multiple domains, including archaeology, genomics, and molecular docking, requiring the precise placement and orientation of elements to reconstruct an original structure. In this work, we address key limitations in state-of-the-art Deep Learning methods for reassembly, namely i) scalability; ii) multimodality; and iii) real-world applicability: beyond square or simple geometric shapes, realistic and complex erosion, or other real-world problems. We propose ReassembleNet, a method that reduces complexity by representing each input piece as a set of contour keypoints and learning to select the most informative ones by Graph Neural Networks pooling inspired techniques. ReassembleNet effectively lowers computational complexity while enabling the integration of features from multiple modalities, including both geometric and texture data. Further enhanced through pretraining on a semi-synthetic dataset. We then apply diffusion-based pose estimation to recover the original structure. We improve on prior methods by 55% *and* 86% *for RMSE Rotation and Translation, respectively.*

1. Introduction

Reassembly requires placing each element in its correct position and orientation to form the original shape as a whole – whether it be a 2D or 3D object. This ability is a form of spatial intelligence, which refers to the capacity to accurately perceive the visual-spatial environment and manipulate that perceived space [9]. This skill is typically evaluated through reassembly tasks, where individual components must be arranged and connected to form a coherent, functional whole—such as solving 2D jigsaw puzzles or assembling 3D structures with LEGO blocks.

Since the advent of the first puzzle solver [5], reassembly tasks have posed a significant challenge to the machine learning community due to their inherent combinatorial complexity. These challenges are further underscored by their wide range of applications, including genomics [21], assistive technologies [39], and molecular docking [3].

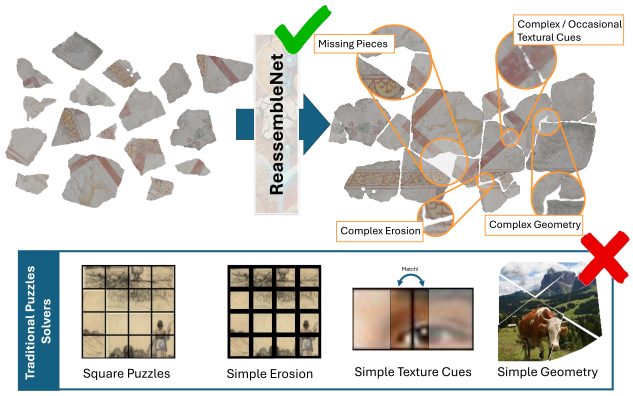


Figure 1. We introduce **ReassembleNet** as a method for fresco reassembly. Our ReassembleNet addresses key challenges that have been ignored by traditional methods, including complex geometry, texture and erosion, missing pieces, often in a data-scarcity scenario.

In recent years, AI techniques for reassembling objects have gained increasing traction in the heritage field, particularly in fresco reconstruction [38]. This growing application of AI is driven by the fundamental challenge archaeologists face in reconstructing the past. After extensive and painstaking work involving site surveys and excavations, they are often confronted with the daunting task of reassembling countless fragments of varying sizes, shapes, and appearances to recreate ancient artifacts or artworks. Depending on the complexity of the artifact, this reassembly process can span months, years, or even decades. In cases involving a vast number of pieces, the task may become nearly insurmountable, regardless of the skill of the experts. Despite recent advances, state-of-the-art methods for solving this task remain far from achieving a satisfactory accuracy [36]. The challenges persist not only in the 3D domain but also in the 2D space, where the reduced

degrees of freedom do not significantly alleviate the complexity of the problem. Reduction to 2D has been used for a long time to aid in finding the solutions as many heritage objects, e.g. frescos, are planar on the dominant surface [6]. Traditional approaches use keypoints or patches [4, 22, 25, 35] benefiting from optimization strategy allowing them to scale to complex irregular shapes, in contrast to deep learning approaches. Three primary factors contribute to the difficulty of fresco reassembly for deep learning approaches. First, the inherent complexity of irregular pieces and the diverse textures present in the fragments (Figure 1). Secondly, as a byproduct of the first, complex polygonal shapes are difficult to model in deep learning approaches, where comparisons need to be made across the points of one fragment edge to another fragment. Finally, deep learning methods typically require large-scale datasets to effectively learn meaningful patterns [32], a requirement that is difficult to meet in most applications domains of reassembly.

To address these challenges, we propose ReassembleNet, a scalable deep learning method that integrates a diffusion process with multi-modal feature fusion. Our approach enables scalability on irregular 2D shapes through keypoint selection, a crucial capability for highly fragmented structures that existing methods struggle to process [18]. We introduce a semi-synthetic dataset specifically designed to highlight the characteristics of fresco pieces. Through finetuning, we demonstrate that pre-training on a semi-synthetic dataset significantly enhances model performance on real-world frescos. We combine the keypoint selection with geometry and texture representation to constrain the matching problem modeled through an inter and intra-piece attention block. Finally, we wrap this model in a diffusion process to iteratively denoise the pose of pieces into the final locations.

The main contributions of this work are: *i)* We propose a scalable end-to-end deep learning method designed to solve reassembly tasks with 2D irregular input shapes. *ii)* We introduce a 2D learnable keypoint selector that identifies the most significant keypoints along the borders of 2D irregular shapes. This approach reduces complexity, enhances scalability, and overcomes the computational limitations of SOTA models that struggle when using all points of the contours. *iii)* We use multimodal features that incorporate geometric features, local and global texture features, to provide a richer representation and aid solving the task. *iv)* We address the limitation of scarce training data by introducing a novel pipeline with a reduced sym-to-real gap that generates a semi-synthetic dataset for pre-training. This strategy enhances overall performance when solving real puzzles.

2. Related Works

We review key literature on reassembly tasks for both square and irregular shape puzzles.

2.1. Reassembly of Square-based puzzles

Square-based jigsaws have been extensively studied in the literature, with early approaches using greedy methods. For instance, [24] introduced a three-phase greedy approach involving placement, segmentation, and shifting, relying on compatibility functions and estimation metrics. While effective for large puzzles, it requires predefined orientations and does not handle missing pieces. Similarly, [7] proposed a tree-based algorithm for puzzles with unknown orientations and locations, using Mahalanobis Gradient Compatibility (MGC) for boundary analysis. However, it struggles with color-based similarity and missing pieces. [23] presented a greedy strategy for puzzles without prior information on piece sizes or orientations, refining the initial configuration. It works with mixed or missing pieces but struggles with noisy images. Other methods, such as [19], combine edge and content similarity using MGC, but face challenges with large-scale puzzles. [29] used a genetic algorithm based on color similarity to solve puzzles with unknown locations and orientations.

In recent years, several deep learning approaches have been developed for puzzle reconstruction. One prominent direction leverages generative models to reconstruct a complete image from an unordered set of puzzle pieces [33, 34]. These methods aim to infer the global structure of the puzzle, effectively generating missing spatial relationships. However, they are not applicable when the input pieces are rotated, as they assume a fixed orientation. Another line of research explores diffusion-based processes to iteratively refine the placement of puzzle pieces step by step [10, 26]. These approaches have demonstrated effectiveness in solving the oriented puzzle problem, where piece orientations are known. However, they are restricted to regular puzzle settings, where pieces conform to a structured grid. For real-world applications, square-based approaches are often impractical, as broken objects typically have irregular shapes, noisy visual content, and eroded or missing edges between pieces. A naive solution is to create a regular patch that encloses the broken object; however, this drastically hinders the effectiveness.

2.2. Reassembly of irregular pieces

To overcome the limitations of approaches restricted to square-based shapes, several recent studies have introduced methods designed explicitly for irregular-shaped object pieces.

Before the adoption of deep learning, various techniques were developed to address this problem. For instance, [25] proposed an artifact reassembly method using inpainting, texture synthesis, and FFT-based image registration. Their approach aligns fragments by maximizing correlation through FFT shift theory, effectively handling damaged edges. Similarly, [22] uses a divide-and-conquer strategy,

grouping pieces based on texture and color similarity. The method utilizes RGB and HIS color spaces for color matching and employs co-occurrence matrices to extract texture features. In [35], the authors introduced a computer-aided method consisting of four key steps. Their approach reconstructs fragmented images by identifying adjacent pieces, matching contours using a Smith-Waterman-based method with color similarity, refining alignment through Iterative Closest Point, and assembling the final image based on optimized alignment angles. More recently, [4] proposed a patch-based optimization method for artifact reconstruction, leveraging color, gradients, and geometric transformations to match fragments.

In recent years, significant progress has been made in applying deep learning to handle irregular input shapes. The first work in this area is [30], which introduced DNN-Buddies, a deep neural network model for evaluating piece compatibility. In [20], the authors enhanced puzzle piece matching using CNNs and adaptive boosting. They proposed a multi-graph search algorithm to replace greedy strategies, enabling the handling of missing pieces and low-texture areas. [1] developed a classification network for evaluating the compatibility of fragment pairs and matching irregular puzzle pieces. However, their approach requires square-shaped pieces and cannot accommodate irregular outputs. PuzzleFusion [17] is based on a diffusion model and treats puzzle pieces as simple polygons with a limited number of vertices. This method is designed to work only on toy problems, as it processes all keypoints of the input polygons. As a result, it becomes impractical for real-world datasets, where a large number of keypoints lead to exponential memory consumption. Finally, PairingNet [40] is a learning-based image fragment pair-searching and -matching approach. Their method employs a graph-based network for feature extraction, a linear transformer-based module for fragment pair-searching, and a weighted fusion module for pair-matching. By formulating a similarity matrix to infer adjacent segments.

Unlike previous works, which have several limitations for real-world applications, our solution is designed for practical scenarios and is capable of handling irregular inputs without scalability issues, thanks to the keypoint selection module. It also distinctively leverages multi-modal features by including both geometry and texture during training and inference.

3. ReassembleNet Pipeline

In this paper, we focus on reassembling fragmented 2D objects where our method input is a set of images representing the fragments with irregular shapes. Following [18], our method takes as input m images of unordered pieces. We extract keypoints from the images by applying Harris Corner Detector [13]. Points act as a helpful cue in the arrangement of the pieces as subsets of the points on each piece can be aligned to recover the overall piece pose.

For each of the m pieces, let the set of keypoints be defined as $K^m = \{\mathbf{k}_i^m\}_{i=1}^{d^m}$, where d^m is the number of keypoints in each piece m. As this results in a significant number of points, direct comparison between all pieces is infeasible; therefore, our method applies an end-to-end learnable keypoint selector, which is trained to identify the most informative keypoints from the fragments. These keypoints are crucial for guiding the reassembly process.

Next, we aim at extracting a multi-modal keypoint feature designed to capture both the geometric and textural properties of each fragment. This allows us to fully embed the shape and visual characteristics of the pieces, which are essential for proper alignment. Finally, we utilize a combination of attention blocks to model the relationships between the fragments. This solution helps the method to better fit the pieces together by analyzing their spatial and contextual connections, ensuring they are aligned in the correct configuration. The model is wrapped within a diffusion process to progressively learn the correct alignment in terms of translation $\mathbf{s}_i^m \in \mathbb{R}^2$ and rotation $\mathbf{r}_i^m = [\cos(\theta_i^m), \sin(\theta_i^m)]^\top$ [41].

3.1. KeyPoint Feature Representation

Each keypoint $\mathbf{k}_i^m$ is represented by a feature vector $\mathbf{h}_i^m$ that combines several components. In challenging reassembly tasks, extracting meaningful characteristics is crucial for providing the network with valuable prior knowledge and improving its performance. Therefore, we focus on extracting multi-modal features from the input - specifically, geometric and texture attributes.

Geometric Features. We focus on two specific geometric features: *Curvature*, which measures the curvature of the boundary at the keypoint. Curvature indicates whether a corner is a sharp bend or a more gentle curve, and it can be approximated by fitting a curve (e.g., a circle or spline) around the keypoint and estimating its curvature. It is analyzed by second-order derivatives and is represented as a single scalar value per keypoint. *Edge Angles* involve calculating the angle of the tangent to the edge at each boundary keypoint. It is a 1D feature (one angle per keypoint) and is measured in degrees. This angle helps distinguish between different types of boundary segments, such as straight edges versus curved ones.

Texture Features. We have also focused on extracting features specifically related to the textures of the pieces. In particular, we extracted global texture features that summarize the general characteristics of the element, as well as local texture features that capture more detailed, localized information. For global texture features, we leveraged a pre-trained ResNet18 [15] to capture high-level texture representations across the entire element. For local texture

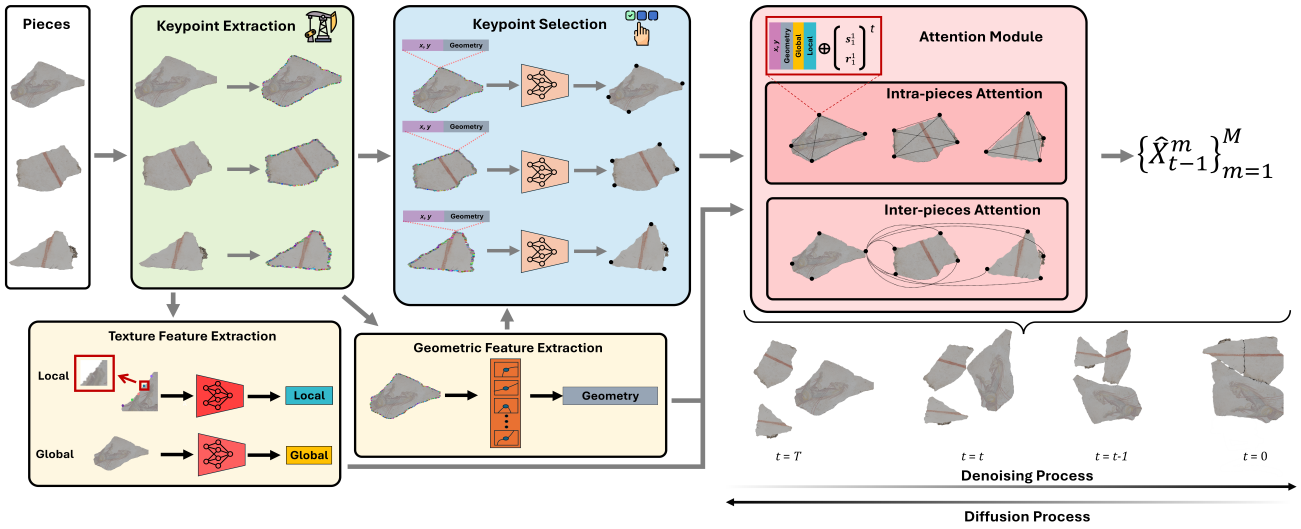


Figure 2. Framework of our proposed ReassembleNet. We begin by extracting keypoints from the input pieces, followed by computing global and local texture features alongside geometric features. Using the geometric features and keypoint coordinates, we then select the most relevant k keypoints. To model the reassembly process, we employ a Diffusion Probabilistic Model, formulating a Markov chain that gradually injects noise into the keypoints' positions and orientations. At timestep $t = 0$, the pieces are correctly aligned, whereas at timestep $t = T$, their keypoints are randomly translated and rotated (note that for visualization purpose, we compute the average translation and rotation of keypoints within each piece at every step in the chain). At each timestep t, our attention module processes the keypoints—incorporating their coordinates, orientations, and extracted features—to predict a less noisy version of their positions and orientations, $\{\hat{X}^m_{t-1}\}^M_{m=1}$, iteratively refining them toward the correct configuration.

features, we adopted a similar approach using the same pre-trained ResNet18. Specifically, we extract 32×32 patches centered on each selected keypoint. These localized patches are processed through ResNet18, where we remove the final classification layer and utilize the penultimate layer for feature extraction. This combination of global and local texture features allows our model to capture both the overall texture characteristics of the element as well as more detailed texture variations, which are crucial for accurately solving reassembly tasks.

3.2. Keypoint Selector Module

A key component of our pipeline is the selection of $k \in \mathbb{R}$ keypoints, which act as anchors to ensure scalability. To maintain consistency and simplicity, we select the same number of k keypoints for all pieces (ablated on in Sec. 5). Identifying the most relevant keypoints is essential for handling irregular shapes with an arbitrary number of points. To achieve this, we explore two strategies: *i)* a non-learnable pre-selection method and *ii)* a learnable approach.

Non-Learnable Algorithm. For the non-learnable strategy, we employ the heuristic Farthest Point Sampling (FPS) [11] to perform an initial pre-selection of the k-points. FPS is a well-established technique in the field of 3D point cloud processing and geometric sampling. It iteratively selects the point that is furthest from the current set, ensuring maximum dispersion throughout the entire data space. This approach guarantees that the selected keypoints are evenly distributed, capturing the underlying structure and geometry of the input data effectively. By leveraging FPS, our method can robustly cover the entire image or spatial domain with a fixed number of representative points.

Learnable Algorithm. We frame the keypoint selection task as a graph sparsification problem [14]. In our formulation, the set of keypoints K^m corresponds to the set of vertices V^m of a complete graph G^m (i.e. fully connected). Graph sparsification is then defined as the process of selecting a subset of nodes—i.e., keypoints—or edges from G^m to produce a sparser graph $\hat{G}^m$. In other words, the elements of $\hat{G}^m$ are a subset of those in G^m.

To address this problem, we designed an architecture that learns to extract k keypoints in a data-driven manner. As illustrated in Figure 3, the architecture comprises three main components:

Projection Layer. The input features—comprising the coordinates and geometric attributes —are first projected into a higher-dimensional space. We rely solely on coordinates and geometric features, excluding texture information, as the pre-training is specifically designed to focus on the input shape (see Equation (2)).

Graph Transformer. The projected features are then processed by a Graph Transformer [28] to aggregate and refine the input information.

Pooling Layer. Finally, we apply a pooling layer [8], where

we select k nodes from the original graph. The selection of nodes to drop is guided by a projection score computed against a learnable vector $\mathbf{p}$. To ensure that gradients propagate into $\mathbf{p}$, these projection scores also serve as gating values, allowing nodes with lower scores to retain fewer features. Defined as:

$$\mathbf{y} = \frac{D\mathbf{p}}{\|\mathbf{p}\|} \qquad \mathbf{j} = \text{top-k}(\mathbf{y}, k)$$
$$\hat{D} = (D \odot \tanh(\mathbf{y}))_j \qquad \hat{A} = A_{j,j} \tag{1}$$

where D represents the feature matrix of the input graph G^m, $\hat{D}$ denotes the output feature matrix of the subset graph $\hat{G}^m$, $\|\cdot\|$ denotes the L2 norm, top-k selects the top k indices from a given input vector, $\odot$ represents element-wise multiplication, and $._j$ is an indexing operation that extracts slices at the indices specified by $\mathbf{j}$. This operation involves only a point-wise projection and slicing of the original feature and adjacency matrices, thereby preserving sparsity.

The algorithm benefits from a pre-training phase to enable the model to identify, based on a general criterion, which keypoints to retain and which to discard in an unsupervised manner. This is helpful because we have no prior knowledge of the most relevant keypoints and aim for a starting point that is not task-specific. Therefore, we train the model to select k nodes while maximizing the preservation of both the perimeter and the area. The keypoints are chosen to best maintain the overall shape of the object. For this reason, we employ the following two losses:

$$\mathcal{L}_{\text{area}} = \left(\frac{A_{\text{total}} - A_{\text{sel}}}{A_{\text{total}}}\right)^2; \; \mathcal{L}_{\text{per}} = \left(\frac{P_{\text{total}} - P_{\text{sel}}}{P_{\text{total}}}\right)^2, \tag{2}$$

where A_{total}, A_{sel}, P_{total}, and P_{sel} represent the area and perimeter measurements. Specifically, A_{total} and P_{total} correspond to the area and perimeter computed using all initial keypoints, while A_{sel} and P_{sel} refer to the area and perimeter computed using only the selected k keypoints. The final loss function is defined as:

$$\mathcal{L} = \lambda_{\text{area}}\mathcal{L}_{\text{area}} + \lambda_{\text{per}}\mathcal{L}_{\text{per}}, \tag{3}$$

where λ_{area} and λ_{per} are the regularizing parameters. This architecture allows us to effectively select representative keypoints by leveraging both the geometric information and the relational structure captured by the graph.

3.3. Estimating position and rotation with diffusion

We adopt Diffusion Probabilistic Models as defined in Denoising Diffusion Implicit Models (DDIM) [31]. For each piece m, we apply an initial transformation, consisting of a translation $\mathbf{s}^m$ and a rotation $\mathbf{r}^m$, that is replicated identical for all its keypoints. To represent this initial transformation compactly, we define a concatenated vector for each keypoint $\mathbf{x}_{i0}^m = [\mathbf{s}_{i0}^{m\top}, \mathbf{r}_{i0}^{m\top}]^\top$, where $\mathbf{s}_{i0}^m$ and $\mathbf{r}_{i0}^m$ denote the initial translation and rotation applied to all keypoints of the piece. Since this transformation is applied globally, all keypoints within the same piece share these initial transformation parameters.

At training time, we iteratively add noise sampled from a Gaussian distribution $\mathcal{N}(0, I)$ to the poses of each keypoints (Forward Process). Following that, for the denoising process, we train ReassembleNet to reverse the noising process (Reverse Process) and predict the initial poses of all keypoints for every piece. We denote the initial poses of all the keypoints as $X_0 = \{X_0^m\}_{m=1}^M$, where for each piece, $X_0^m = \{\mathbf{x}_{i0}^m\}_{i=1}^K$, represents the set of initial keypoint poses. Additionally, we denote the predicted keypoint poses at timestep $t-1$ as $\hat{X}_{t-1} = \{\hat{X}_{t-1}^m\}_{m=1}^M$. For the final result at time $t = 0$, we take the average polygon Euclidean center position and the polygon rotation. Further details on the diffusion process we use can be found in Supplementary Material C.

The Architecture. To process the task, we aim to capture both intra-piece information and inter-piece (i.e. global information between pieces). The key idea is to handle these two types of information separately but in parallel. To achieve this, we employ a combination of attention layers. Intra-piece information is processed using a sparse self-attention block [2], which helps the network focus on obtaining consistent positions and rotations, across different keypoints within the same piece. While, inter-piece information is handled using a standard self-attention block [37] between pieces.

Losses. Following [16] and standard practice in Diffusion Models, we train ReassembleNet to directly predict $\hat{X}_0$ rather than $\hat{X}_{t-1}$. We employ two loss functions to reconstruct the initial pose of each piece's keypoints.

Translation Loss. This loss measures the average discrepancy between the ground truth translation vectors and the predicted translations $\hat{\mathbf{s}}_{i0}^m$:

$$\mathcal{L}_{tr} = \frac{1}{M}\frac{1}{K}\sum_{m=1}^{M}\sum_{i=1}^{K}\|\mathbf{s}^m - \hat{\mathbf{s}}_{i0}^m\|_2^2,$$

where $\|\cdot\|_2^2$ denotes the squared L2 norm.

Rotation Loss. This loss quantifies the average discrepancy between the ground truth rotation and the predicted rotations $\hat{\mathbf{r}}_{i0}^m$:

$$\mathcal{L}_{rt} = \frac{1}{M}\frac{1}{K}\sum_{m=1}^{M}\sum_{i=1}^{K}\|\mathbf{r}^m - \hat{\mathbf{r}}_{i0}^m\|_2^2.$$

3.4. Semi-Synthetic Dataset creation and Fine-Tuning

To increase training data while reducing the sym-to-real gap of the synthetic data we introduce a large-scale

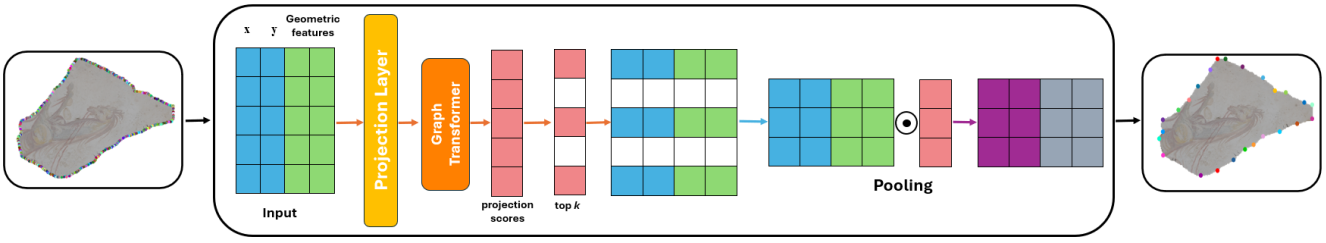


Figure 3. An illustration of the Learnable KeyPoint Selector Module where keypoints are projected into a high dimensional space, then use a graph transformer to predict scores which is then used to identify the top-k and pooled to identify the most important keypoints.

semi-synthetic dataset based on RePAIR [36]. We build from [40], where frescoes are divided into fragments with varied break patterns. The segmentation begins by selecting two points on the fresco's circumscribed circle. The segmentation line is then randomly divided into straight or curved segments, with curves created using Fourier bases for realistic edges. The generated pieces of this algorithm are not capable of emulating the complexity of fresco datasets. To overcome this limitation, we enhance the approach by incorporating two additional features: applying erosion operation on the piece and applying a slight random rotations and translations. Rather than introducing noise, these adjustments are essential for accurately modeling realistic fragmentation (see Section 5 and Supp. Mat. H for more details).

Based on this new semi-synthetic dataset, we pre-train our model on it and fine-tune the method to a real-world dataset with fewer samples. This approach leverages the knowledge acquired from a large-scale dataset to reduce the need for extensive supervised training, thereby enhancing sample efficiency and accelerating convergence, while reducing overfitting and enhancing generalization.

4. Evaluation

We evaluate our method on the RePAIR benchmark dataset [36], following the same experimental protocol. The dataset evaluation procedures are detailed in Section 4.1, while the performance of all methods is assessed in Section 4.2. In Section 5, we present an ablation study on the scalability, the selection of the number of keypoints and the semi-synthetic dataset creation. Additional ablation studies on ReassembleNet's configuration options Supp. Mat. G and results on the semi-synthetic dataset Supp. Mat. H.

4.1. Dataset, Metrics and Baseline Methods

RePAIR Dataset. The dataset serves as a challenging benchmark for testing modern computational and data-driven puzzle-solving methods. It features realistic 2D and 3D fragments of frescos that, due to natural and human-made impacts undergone over time, exhibit erosion, missing pieces, and irregular shapes. The dataset is multi-modal, including high-resolution images and archaeologist-annotated ground truth and metadata. The RePAIR 2D dataset consists of 100 puzzle samples, with 79 for training and 21 for testing. The total number of fragments in RePAIR is 809, with an average of 8.09 fragments per puzzle.

Semi-Synthetic Dataset The semi-synthetic dataset has 5000 samples with a total of 45200 pieces, resulting in an average of around 9 pieces per puzzle. Following the RePAIR 2D train-test split (80:20 ratio), we divided the semi-synthetic dataset into 4000 training samples and 1000 testing samples.

Evaluation Metrics. Following [10], we evaluated the methods using the Root Mean Square Error (RMSE) for both translation (in millimeters) and rotation (in degrees), computed with respect to the ground truth. Specifically, we predict the final pose of each keypoint and compute the average polygon translation and rotation as $\boldsymbol{\mu}_{\hat{t}}^{m} = \frac{1}{K}\sum_{i=1}^{K}\hat{\mathbf{s}}_{i0}^{m}$ and $\boldsymbol{\mu}_{\hat{r}}^{m} = \frac{1}{K}\sum_{i=1}^{K}\hat{\mathbf{r}}_{i0}^{m}$, respectively. We also evaluated the performance of the methods using the Q_{pos} metric [36], which quantifies the overlap between the ground truth fragment poses (translation and rotation) and the reconstructed solution. More details about these metrics can be found in the Supplementary Material E.

Baseline Methods. We compare ReassembleNet against both non-learnable and learnable SOTA approaches.

Non-learnable approaches: *i)* the Archaeological Puzzle Solver [4], which applies a greedy "next best piece" algorithm based on texture. However, they use an outdated extrapolation process, which was replaced by [12] with the stable-diffusion extrapolation method. *ii)* The Genetic Algorithm reconstruction uses a fitness function based on geometry, specifically the area of the puzzle's bounding rectangle and the intersection area of overlapping pieces. While perfect solutions minimize both values, this does not guarantee an optimal solution. *iii)* Greedy geometric matching, which, starting from a random seed fragment, iteratively extends the fragment pose in a greedy fashion based on geometry, in contrast to [4], which relies on texture compatibility.

Learnable approaches: *i)* DiffAssemble [26] is a GNN-based architecture designed to tackle reassembly tasks using a diffusion model formulation. In this approach, pieces

are treated as elements within a set, represented as nodes in a spatial graph. In the 2D scenario, these pieces are modeled as regular patches. *ii)* PairingNet [40] is a learning-based approach for fragment pair-searching and matching. It uses a graph-based network to extract features, integrates them via a linear transformer module, and employs contrastive loss for global encoding. A weighted fusion module then computes similarity scores to infer adjacent segments.

4.2. RePAIR Dataset Evaluation

Details. We compare ReassembleNet with both non-learnable and learnable methods. We train our model with Adafactor as the optimization algorithm [27] and initialize the learning rate with 0.001. We set a batch size of 4. Our method involves selecting k keypoints (sec. 3.2, which is set to $k = 20$. For the learnable keypoint selection module, we pre-train to enhance its performance. Further details on the pre-training process are provided in the Supp. Mat. F. We assess the performance of ReassembleNet in the three different configurations for keypoint selection: (i) a strategy based on the no-learnable keypoint selection, (ii) a frozen learnable module, and (iii) a trainable keypoint selection module optimized for the given task.

Results. Table 1 presents the results on the RePAIR dataset. ReassembleNet with learnable keypoint selection and fine-tuning outperforms the other methods. This result emphasizes that representing irregular pieces as points, while selecting the best keypoints as done by ReassembleNet, is effective. It leads to improvements over the second best performing method, *Greedy Geom Match* [36], by 55% and 86% for RMSE rotation and translation, respectively.

Regarding the keypoint selection comparison, as shown in the table, ReassembleNet with the learnable keypoint selector achieves the best performance. These results demonstrate the benefits of using a learnable module instead of a non-learnable one (i.e. Furthest Point Sampling), as it can adapt to extract the most significant keypoints.

Additionally, when comparing the performance of the learnable models with and without fine-tuning, we observe a clear benefit from applying fine-tuning. All methods show improved performance in rotation and translation, while their performance in Q_{pos} decreases. These results emphasize the effectiveness of our dataset construction, as it successfully preserves the patterns from the original dataset.

Regarding the results of other methods, DiffAssemble approximates irregular shapes as squared pieces, making it challenging to accurately identify the correct matches. Furthermore, in the original 2D configuration reported in [26], the method relies on a regular grid to arrange the pieces in the output. Since no such grid is present in our case, this introduces additional limitations. PairingNet performs poorly because it employs a pair-matching loss to align contour points, which is unsuitable for puzzle pieces with erosion or gaps between them. Non-learnable approaches, as also stated in [36], achieve low performance, reinforcing the advantages of learned strategies.

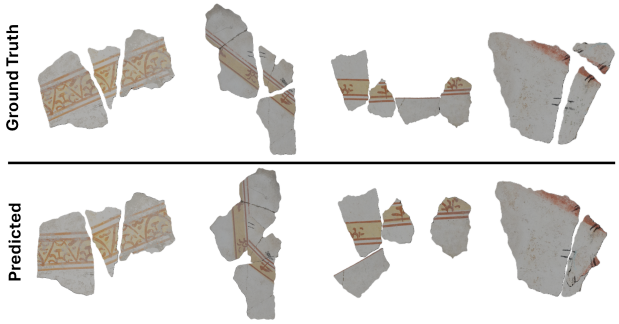


Figure 4. Qualitative results on RePAIR, showing the reassembly outcomes on four frescoes.

Figure 4 reports qualitative results of ReassembleNet on four example frescos. More quantitative are presented in Supplementary Material I.

5. Ablation Study

Scalability. In this experiment, we aim to demonstrate the scalability of ReassembleNet. To achieve this, we generate n different datasets, where the number of pieces grows exponentially, i.e., 2^n. We set $n = 7$ and evaluate both models.

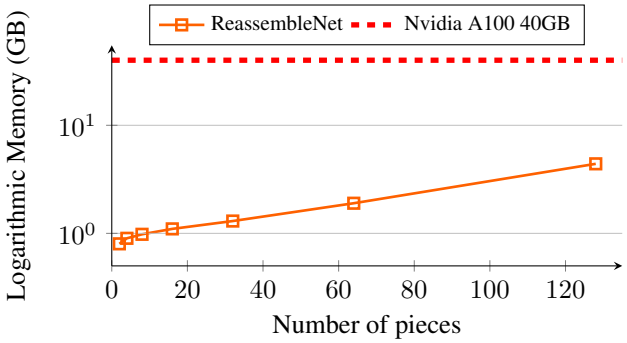


Figure 5. GPU memory consumption as a function of the number of puzzle pieces on RePAIR dataset.

In Figure 5, we present the memory consumption trends of ReassembleNet as the number of puzzle pieces increases. As shown, ReassembleNet successfully scales with larger puzzle sizes. This highlights the robustness and efficiency of our method in handling more complex puzzle configurations, making it suitable for larger-scale realistic problems.

The Number of Keypoints. We evaluate the impact of the number of keypoints by conducting experiments with ReassembleNet, using learnable keypoint selection on the RePAIR dataset without fine-tuning.

Category	Method	$Q_{pos} \uparrow$	RMSE ($\mathcal{R}^\circ$) $\downarrow$	RMSE ($\mathcal{T}_{mm}$) $\downarrow$
Non-learnable	Derech *et al*. [4]	0.04	80.96	139.49
	Genetic Optimization [36]	0.05	85.63	151.71
	Greedy Geom Match [36]	0.02	76.99	135.95
Learnable & No Fine-Tuning	DiffAssemble [26]	0.10	131.36	283.55
	PairingNet [40]	0.16	98.45	390.43
	ReassembleNet w/ no Learnable KP selection	0.34	57.11	16.58
	ReassembleNet w/ Frozen Learnable KP selection	0.38	55.72	27.01
	ReassembleNet w/ Learnable KP selection	0.22	49.42	21.79
Learnable & Fine-Tuning	DiffAssemble [26]	0.10	123.42	280.76
	PairingNet [40]	0.13	91.54	364.62
	ReassembleNet w/ no Learnable KP selection	0.19	43.51	18.82
	ReassembleNet w/ Frozen Learnable KP selection	0.16	40.60	18.39
	ReassembleNet w/ Learnable KP selection	0.19	34.28	17.90

Table 1. Results on RePAIR dataset [36] using Q_{pos} from [36] for groundtruth overlap, Root Mean Square Error (RMSE), in terms of, rotation ($\mathcal{R}^\circ$) and translation ($\mathcal{T}_{mm}$). Comparing against non-learnable optimization methods [4, 36] and learning (i.e., deep learning) methods [26, 40]. We do not include PuzzleFusion [18] because it runs out-of-memory due to highly irregular shape.

KeyPoints	Memory (MB)	$Q_{pos} \uparrow$	RMSE ($\mathcal{R}^\circ$) $\downarrow$	RMSE ($\mathcal{T}_{mm}$) $\downarrow$
3	22630	0.07	103.26	50.89
5	24791	0.10	86.74	37.42
10	28434	0.14	78.91	23.37
15	34209	0.17	67.61	24.25
20	36701	0.22	49.42	21.79
25	[OOM]	[OOM]	[OOM]	[OOM]

Table 2. Comparison of the parameter k for the number of Keypoints selected by the Keypoint selector (sec. 3.2)

Table 2 reports the results of the keypoint number evaluation. It is clear that setting $k = 20$ yields the best performance while efficiently maximizing the available resources (40 GB of GPU memory).

Semi-Synthetic Dataset Creation Process. As discussed in Section 3.4, we modify the algorithm proposed by [40]. In this experiment, we compare our semi-synthetic dataset creation method with the original algorithm introduced by [40].

Strategy	$Q_{pos} \uparrow$	RMSE ($\mathcal{R}^\circ$) $\downarrow$	RMSE ($\mathcal{T}_{mm}$) $\downarrow$
[40]	0.18	56.54	26.42
Our	0.19	34.28	17.90

Table 3. Results on the impact of the Semi-Synthetic Dataset Creation process in contrast to [40].

Table 3 reports the results, showing that ReassembleNet trained on our semi-synthetic dataset outperforms ReassembleNet trained on the dataset generated using [40]. This validates the effectiveness of the modifications we introduced to the original algorithm.

6. Conclusion

In this work, we introduce ReassembleNet, a scalable deep learning approach for reassembly tasks. We propose the first 2D keypoint selector module designed to identify the most relevant keypoints that represent the pieces contour. Additionally, we integrate multimodal features, including both geometric and texture-based information, to better capture the characteristics of the pieces. We demonstrate that pre-training and fine-tuning on a semi-synthetic dataset, specifically created for this study, enhance ReassembleNet's ability to generalize to real-world fresco datasets, such as RePAIR.

Our experiments show that ReassembleNet outperforms existing methods in the fresco domain and is suitable for real-world applications. Through an extensive ablation study, we highlight the benefits of using ReassembleNet in terms of memory consumption, as well as the importance of incorporating a learnable keypoint selector. This integration allows the network to adapt its selection based on the task and input dataset, improving overall performance.

The limitations of ReassembleNet include the use of a fixed number of keypoints, whereas the selection should ideally vary depending on the complexity of the pieces, as well as the use of a non-rotation equivariant backbone for extracting texture features.

Our ReassembleNet demonstrates a move towards usable reassembly for real-world problems, enabling the possibility for archaeologists and other application domains to integrate automatic assembly into workflows.

References

[1] Yangjie Cao, Zheng Fang, Hui Tian, and Ronghan Wei. 2d irregular fragment reassembly with deep learning assistance. *IEEE Access*, 2024. 3

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 5

[3] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *International Conference on Learning Representations (ICLR)*, 2023. 1

[4] Niv Derech, Ayellet Tal, and Ilan Shimshoni. Solving archaeological puzzles. *Pattern Recognition*, 119:108065, 2021. 2, 3, 6, 8

[5] Herbert Freeman and L Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers*, (2):118–127, 1964. 1

[6] Thomas Funkhouser, Hijung Shin, Corey Toler-Franklin, Antonio García Castañeda, Benedict Brown, David Dobkin, Szymon Rusinkiewicz, and Tim Weyrich. Learning how to match fresco fragments. *Journal on Computing and Cultural Heritage (JOCCH)*, 4(2):1–13, 2011. 2

[7] Andrew C Gallagher. Jigsaw puzzles with pieces of unknown orientation. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 382–389. IEEE, 2012. 2

[8] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019. 4

[9] Howard E Gardner. *Frames of mind: The theory of multiple intelligences*. Basic books, 2011. 1

[10] Francesco Giuliari, Gianluca Scarpellini, Stuart James, Yiming Wang, and Alessio Del Bue. Positional diffusion: Ordering unordered sets with diffusion probabilistic models. *arXiv preprint arXiv:2303.11120*, 2023. 2, 6

[11] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985. 4

[12] Peleg Harel, Ofir Itzhak Shahar, and Ohad Ben-Shahar. Pictorial and apictorial polygonal jigsaw puzzles from arbitrary number of crossing cuts. *International Journal of Computer Vision*, 2024. 6

[13] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, pages 10–5244. Citeseer, 1988. 3

[14] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B. Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8058–8066. International Joint Conferences on Artificial Intelligence Organization, 2024. Survey Track. 4

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 5, 1

[17] Sepidehsadat Hosseini, Mohammad Amin Shabani, Saghar Irandoust, and Yasutaka Furukawa. Puzzlefusion: Unleashing the power of diffusion models for spatial puzzle solving. 3

[18] Sepidehsadat Hosseini, Mohammad Amin Shabani, Saghar Irandoust, and Yasutaka Furukawa. Puzzlefusion: Unleashing the power of diffusion models for spatial puzzle solving, 2023. 2, 3, 8

[19] Sou-Young Jin, Suwon Lee, Nur Aziza Azis, and Ho-Jin Choi. Jigsaw puzzle image retrieval via pairwise compatibility measurement. In *2014 International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 123–127. IEEE, 2014. 2

[20] Canyu Le and Xin Li. Jigsawnet: Shredded image reassembly using convolutional neural network and loop-based composition. *IEEE Transactions on Image Processing*, 28(8): 4000–4015, 2019. 3

[21] William Marande and Gertraud Burger. Mitochondrial dna as a genomic jigsaw puzzle. *Science*, 318(5849):415–415, 2007. 1

[22] Ture R Nielsen, Peter Drewsen, and Klaus Hansen. Solving jigsaw puzzles using image features. *Pattern Recognition Letters*, 29(14):1924–1933, 2008. 2

[23] Genady Paikin and Ayellet Tal. Solving multiple square jigsaw puzzles with missing pieces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4832–4839, 2015. 2

[24] Dolev Pomeranz, Michal Shemesh, and Ohad Ben-Shahar. A fully automated greedy square jigsaw puzzle solver. In *CVPR 2011*, pages 9–16. IEEE, 2011. 2

[25] Mahmut Samil Sagiroglu and Aytül Erçil. A texture based matching approach for automated assembly of puzzles. In *18th International Conference on Pattern Recognition (ICPR'06)*, pages 1036–1041. IEEE, 2006. 2

[26] Gianluca Scarpellini, Stefano Fiorini, Francesco Giuliari, Pietro Moreiro, and Alessio Del Bue. Diffassemble: A unified graph-diffusion model for 2d and 3d reassembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28098–28108, 2024. 2, 6, 7, 8, 1

[27] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018. 7

[28] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 4

[29] Dror Sholomon, Omid David, and Nathan Netanyahu. A generalized genetic algorithm-based solver for very large jigsaw puzzles of complex types. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014. 2

[30] Dror Sholomon, Omid E David, and Nathan S Netanyahu. Dnn-buddies: A deep neural network-based estimation met-

ric for the jigsaw puzzle problem. In *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*, pages 170–178. Springer, 2016. 3

[31] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 5

[32] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 2

[33] Davide Talon, Alessio Del Bue, and Stuart James. Ganzzle: Reframing jigsaw puzzle solving as a retrieval task using a generative mental image. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 4083–4087. IEEE, 2022. 2

[34] Davide Talon, Alessio Del Bue, and Stuart James. Ganzzle++: Generative approaches for jigsaw puzzle solving as local to global assignment in latent spatial representations. *Pattern Recognition Letters*, 187:35–41, 2025. 2

[35] Efthymia Tsamoura and Ioannis Pitas. Automatic color based reassembly of fragmented images and paintings. *IEEE Transactions on Image Processing*, 19(3):680–690, 2009. 2, 3

[36] Theodore Tsesmelis, Luca Palmieri, Marina Khoroshiltseva, Adeela Islam, Gur Elkin, Ofir I Shahar, Gianluca Scarpellini, Stefano Fiorini, Yaniv Ohayon, Nadav Alali, et al. Reassembling the past: The repair dataset and benchmark for real world 2d and 3d puzzle solving. *Advances in Neural Information Processing Systems*, 37:30076–30105, 2025. 1, 6, 7, 8

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5

[38] Ariana M Villegas-Suarez, Cristian Lopez, and Ivan Sipiran. Matchmakernet: Enabling fragment matching for cultural heritage analysis. *Preprint*. Accessed: No DOI available at the time of retrieval. 1

[39] E Yurteri, Sabri Derin, Fatma Polat, Murat Canpolat, Zekiye Barman, C Polat, Emel Kaya, and Selim Güllü. Obstacle maps. *New Era Journal*, 1(1):1–15, 2022. 1

[40] Rixin Zhou, Ding Xia, Yi Zhang, Honglin Pang, Xi Yang, and Chuntao Li. Pairingnet: A learning-based pair-searching and-matching network for image fragments. In *European Conference on Computer Vision*, pages 234–251. Springer, 2024. 3, 6, 7, 8, 1, 2

[41] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 3

ReassembleNet: Learnable Keypoints and Diffusion for 2D Fresco Reconstruction

Supplementary Material

A. Introduction

In this supplementary martial, we present details on: the experimental details (sec. B; a detailed description of the diffusion process (sec. C); the evaluation over the semi-synthetic dataset (sec. D), metric formulation (sec. E); the keypoint selector (sec. F); an ablation study on the features (sec. G); and qualitative results for synthetic (sec. H) and RePAIR (sec. I) datasets.

B. Experiment Details

Hardware. The experiments were conducted on four machines, each equipped with an NVIDIA A100 GPU (40GB), 380GB of RAM, and two Intel(R) Xeon(R) Silver 4210 CPUs (2.20GHz, Sky Lake architecture).

C. The Diffusion Process

Forward Process. We define the forward process as a fixed Markov chain that adds noise following a Gaussian distribution to each input, i.e., each keypoints, $\mathbf{x}_{i0}^m$ to obtain a noisy version, $\mathbf{x}_{it}^m$, at timestep t. Following [16], we adopt the variance β_t according to a cosine scheduler and define $q(\mathbf{x}_{it}^m|\mathbf{x}_{i0}^m)$ as:

$$q(\mathbf{x}_{it}^m|\mathbf{x}_{i0}^m) = \mathcal{N}(\mathbf{x}_{it}^m; \sqrt{\overline{\alpha}_t}\mathbf{x}_{i0}^m, (1-\overline{\alpha}_t)\mathbf{I}), \tag{4}$$

where $\overline{\alpha}_t = \prod_{c=1}^{t}(1-\beta_c)$ and $\mathbf{I}$ is the identity matrix.

Reverse Process. The reverse process iteratively recovers the initial poses for the set of elements $\hat{X}_{t-1}$ using the current (noisy) poses $X_t = \{X_t^m\}_{m=1}^M$ and the features $H = \{H^m\}_{m=1}^M$, where each $H^m = \{\mathbf{h}_i^m\}_{i=1}^K$ is the set of features for the keypoints in each piece. The recovered poses $\hat{X}_{t-1}$ are computed as:

$$\hat{X}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(X_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(X_t, H, t)\right), \tag{5}$$

where $\alpha_t = 1-\beta_t$, and $\epsilon_\theta(X_t, H, t)$ is the estimated noise output by ReassembleNet that has to be removed from $\hat{X}_t$ at timestep t to recover $\hat{X}_{t-1}$.

D. Semi-Synthetic Dataset Evaluation

We compare ReassembleNet on this dataset with learnable methods. We train ReassembleNet using geometric, local, and global features in three different configurations: (i) ReassembleNet-conf. 1, which has *no Learnable KP selection*, (ii) ReassembleNet-conf. 2, which uses *Frozen Learnable KP selection*, and (iii) ReassembleNet-conf. 3, which incorporates *Learnable KP selection*.

Method	RMSE ($\mathcal{R}^\circ$) ↓	RMSE ($\mathcal{T}_{mm}$) ↓
DiffAssemble [26]	122.92	73.79
PairingNet [40]	60.11	266.84
ReassembleNet-conf. 1	40.43	16.91
ReassembleNet-conf. 2	36.02	14.69
ReassembleNet-conf. 3	35.79	15.58

Table 4. Results on Semi-Synthetic dataset.

Results. Table 4 presents the results on the Semi-Synthetic Dataset. As shown, ReassembleNet outperforms the second-best method across all the metrics. This result demonstrates that representing irregular objects as 2D points, as done by ReassembleNet, is more effective than treating them as squared images with padding, as done by DiffAssemble, to achieve a regular shape.

E. Metrics Explanation

To evaluate the performance of the methods, we use three different metrics: RMSE for translation, RMSE for rotation and the Q_{pos}.

The RMSE for translation and rotation are defined as:

$$\text{RMSE}(\mathcal{T}_{mm}) = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\|\mu_{\hat{\mathbf{t}}}^m - \mu_{\mathbf{t}}^m\|_2}, \tag{6}$$

$$\text{RMSE}(\mathcal{R}^\circ) = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\| \mu_{\hat{r}}^m - \mu_r^m\|_2}, \tag{7}$$

where μ_t^m denotes the mean ground truth translations, and μ_R^m denotes the corresponding mean ground truth rotations for the m-th piece.

We also evaluate the performance of the methods using the Q_{pos} metric [36], which quantifies the overlap between the ground truth fragment poses (translation and rotation) and the reconstructed solution. To ensure that the metric is invariant to rigid motions—preventing good solutions from being penalized due to differing global rotations—we first apply a rigid transformation to align the largest reconstructed fragment (referred to as the *anchor*) with its corresponding ground truth fragment in both translation and

Category	Method	Global Feats	Local Feats	Geom Feats	$Q_{pos}\uparrow$	RMSE ($\mathcal{R}°$)$\downarrow$	RMSE ($\mathcal{T}_{mm}$)$\downarrow$
No Transfer Learning	no Learnable KP selection	X	X	X	0.18	64.51	80.19
	Frozen Learnable KP selection	X	X	X	0.23	62.45	33.82
	no Learnable KP selection	X	X	V	0.17	59.76	17.76
	Frozen Learnable KP selection	X	X	V	0.22	43.11	22.03
	no Learnable KP selection	V	X	X	0.14	63.24	32.30
	Frozen Learnable KP selection	V	X	X	0.22	62.95	24.42
	no Learnable KP selection	X	V	X	0.27	64.08	19.75
	Frozen Learnable KP selection	X	V	X	0.28	49.58	23.11
	no Learnable KP selection	V	V	V	0.34	57.11	16.58
	Frozen Learnable KP selection	V	V	V	0.38	55.72	27.01
	Learnable KP selection	V	V	V	0.22	49.42	21.79
Transfer Learning	no Learnable KP selection)	X	X	X	0.27	53.47	28.74
	Frozen Learnable KP selection)	X	X	X	0.15	51.95	21.63
	no Learnable KP selection	X	X	V	0.21	56.27	17.76
	Frozen Learnable KP selection	X	X	V	0.15	41.74	20.92
	no Learnable KP selection	V	X	X	0.19	59.07	23.43
	Frozen Learnable KP selection	V	X	X	0.13	58.97	23.26
	no Learnable KP selection	X	V	X	0.20	61.83	17.64
	Frozen Learnable KP selection	X	V	X	0.15	45.46	16.68
	no Learnable KP selection	V	V	V	0.19	43.51	18.82
	Frozen Learnable KP selection	V	V	V	0.16	40.60	18.39
	Learnable KP selection	V	V	V	0.19	34.28	17.90

Table 5. Ablation on ReassembleNet settings.

rotation. To compute Q_{pos}, we first define the area of a fragment, denoted as $A(m)$. In 2D, the shared area can be determined in two different ways: *i)* by comparing the non-transparent pixels of two large canvases containing all fragments, or *ii)* by computing the area intersection of the registered 2D point clouds. Additionally, fragments are weighted based on their area, emphasizing the impact of errors on larger fragments. The metric is formally defined as:

$$Q_{pos} = \sum_{m=1}^{M} w_m \cdot \frac{|A(m \cap \tilde{m})|}{|A(\tilde{m})|}, \tag{8}$$

where $w_m = \frac{|A(m)|}{\sum_{k=1}^{M}|A(k)|}$ represents the weight of each fragment, and $A(\tilde{m})$ denotes the area of the fragments with predicted rotation and translation.

F. Keypoints Selector

As detailed in Section 3.2, our approach involves selecting k keypoints. To achieve this, we employ our learnable keypoint selection module, which is pre-trained to improve its effectiveness. During the pre-training phase, we utilize the RePAIR dataset, treating each piece independently. This dataset enables the model to learn to identify salient keypoints in a diverse and representative context.

We then optimize the module using the two loss functions defined in Equation (2), with $\lambda_{area} = 1$ and $\lambda_{per} = 1$. These losses work together to enforce geometric precision and structural consistency, while also mitigating selection bias toward task-specific nodes.

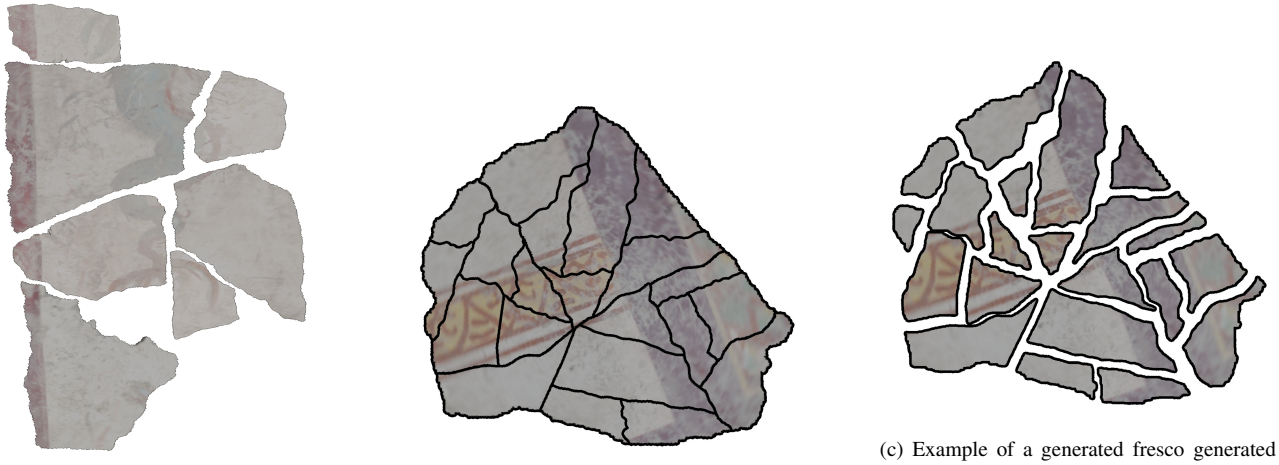
G. Ablation Study on Multimodal Features

Table 5 presents a comprehensive ablation study assessing the impact of the final configuration used for ReassembleNet. The results clearly demonstrate that incorporating all features and leveraging transfer learning are crucial for tackling this challenging task. By utilizing the full set of features, our model gains both geometric awareness of the object and semantic understanding through local and global image representations. This injected bias enhances the network's ability to learn effectively.

H. Qualitative comparison on Semi-Synthetic Dataset Creation Process

In this section, we are reporting a visual representation of the semi-synthetic dataset created following [40] and the final results of the semi-synthetic dataset we were able to create by adding the random erosion of the borders with a slight random rotations and translation.

Figure 6 shows the visual differences in the creation of the semi-synthetic dataset. As can be seen, our proposed algorithm (Figure 6c) exhibits a certain similarity to Figure 6a, which is taken from the real-world dataset RePAIR. In contrast, Figure 6b clearly shows that the puzzle generated using the algorithm in [40] deviates significantly from the characteristics present in RePAIR: the pieces are assembled to align perfectly without gaps, ensuring a seamless matching between the pieces.



(a) (b) (c) Example of a generated fresco generated using our modified algorimth

Figure 6. An illustration of (a) an example of a RePAIR fresco, (b) a synthetic fresco generated using the algorithm proposed by [40], and (c) a synthetic fresco generated using our modified algorithm. The black contour is intentionally added to highlight the borders of the pieces in b and c.

I. More Qualitative Results on RePAIR Dataset

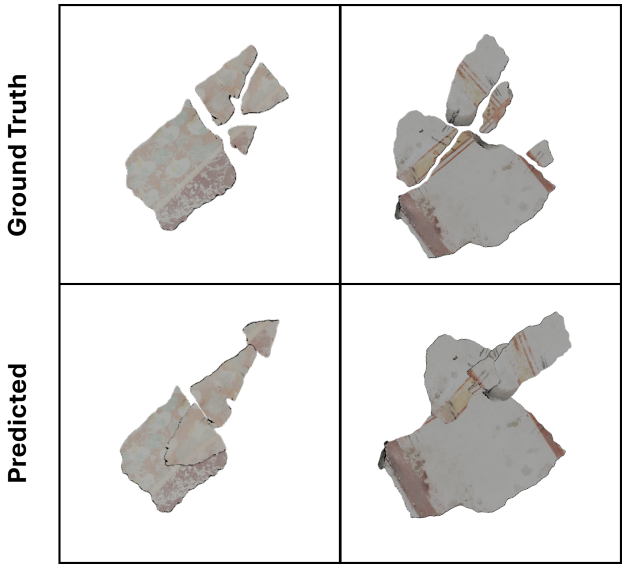


Figure 7. Qualitative results

We report some more qualitative results on the RePAIR dataset. In particular, we report with Figure 7 some failure cases where it can be seen that the model is learning the complexity of groundtruth data.