FoldPath: End-to-End Object-Centric Motion Generation via Modulated Implicit Paths

FoldPath: End-to-End Object-Centric Motion Generation via Modulated Implicit Paths

Paolo Rabino[1,2], Gabriele Tiboni[1], Tatiana Tommasi[1]

***Abstract*— Object-Centric Motion Generation (OCMG) is instrumental in advancing automated manufacturing processes, particularly in domains requiring high-precision expert robotic motions, such as spray painting and welding. To realize effective automation, robust algorithms are essential for generating extended, object-aware trajectories across intricate 3D geometries. However, contemporary OCMG techniques are either based on ad-hoc heuristics or employ learning-based pipelines that are still reliant on sensitive post-processing steps to generate executable paths. We introduce *FoldPath*, a novel, end-to-end, neural field based method for OCMG. Unlike prior deep learning approaches that predict discrete sequences of end-effector waypoints, FoldPath learns the robot motion as a continuous function, thus implicitly encoding smooth output paths. This paradigm shift eliminates the need for brittle post-processing steps that concatenate and order the predicted discrete waypoints. Particularly, our approach demonstrates superior predictive performance compared to recently proposed learning-based methods, and attains generalization capabilities even in real industrial settings, where only a limited amount of expert samples are provided. We validate FoldPath through comprehensive experiments in a realistic simulation environment and introduce new, rigorous metrics designed to comprehensively evaluate long-horizon robotic paths, thus advancing the OCMG task towards practical maturity.**

I. INTRODUCTION

Automated robotic spray painting is an essential component of modern manufacturing, involving ad-hoc motion generation to replicate expert strategies while optimizing material usage, paint coverage, and engineering costs. However, planning efficient trajectories over complex 3D objects remains challenging, often requiring *tens* of long-horizon paths per object, with execution times reaching 8–10 minutes.

Traditional methods for robotic spray painting predominantly rely on heuristic strategies and rule-based motion generation [1], [2], [3], [4]. These approaches typically involve partitioning 3D objects into convex regions and applying coverage planning techniques, such as raster-based paths or predefined tool orientations. While effective in controlled environments, these heuristics have limited adaptability to new object shapes, often require manual tuning, and are computationally expensive for complex surfaces.

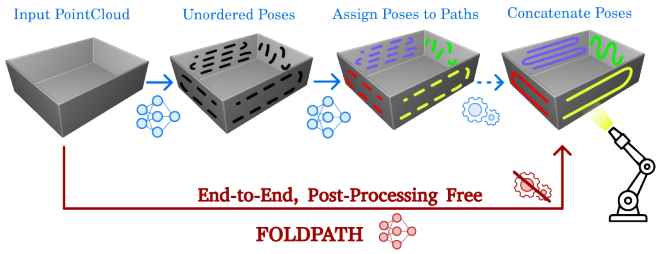

Fig. 1. FoldPath tackles Object-Centric Motion Generation with an end-to-end deep learning pipeline for the first time. Differing from previous works that require multiple learning and post-processing stages, our model directly predicts long-horizon output paths that generalize across free-form 3D objects.

To overcome these limitations, learning-based approaches have been recently explored to predict painting trajectories directly from expert demonstrations [5], [6], given a point cloud encoding of the 3D object. Such methods formulate the problem as the prediction of unordered end-effector poses, which are then post-processed into coherent paths. As a result, these approaches are limited to discrete path representations and yet depend on brittle post-processing steps that are highly sensitive to variations in network predictions, often leading to unfeasible trajectories and discontinuities.

In this work, we build on existing learning-based methods to investigate strategies that can quickly generalize motion generation for previously unseen free-form 3D objects. Yet, we depart from multi-stage post-processing-based techniques and propose *FoldPath*, the first end-to-end training pipeline for direct prediction of multiple, long-horizon paths that attain smoothness by design (see Fig. 1). Particularly, we learn to decode the output paths from the input object geometry and leverage neural fields [7] to formalize the paths as continuous functions of 6D end-effector poses. This allows to sample the paths by inputting scalars interpretable as relative time-steps indicating instants along the path execution. This paradigm makes our pose predictions ordered and smooth, working around the shortcomings of recent approaches.

Our data-driven model contribute to the solution of Object-Centric Motion Generation (OCMG) [6] tasks beyond robotic spray painting, as it does not rely on domain-specific knowledge or task-specific optimization objectives. Importantly, we also introduce a new set of metrics for the OCMG problem, combining set-to-set and curve similarity measures. We adapt the standard AP metric [8] for arbitrary-length curves, ensuring a mature evaluation of path properties like order, location and orientation precision, and uniqueness.

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union NextGenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.
The EFORT group provided the authors with original object meshes, trajectory data, and access to the proprietary spray painting simulator used during the experiments.

[1] Politecnico di Torino, Italy `first.last@polito.it`
[2] Italian Institute of Technology, Genova

In summary, our contributions are the following:

- We introduce FoldPath, a novel deep learning approach for generating multiple smooth, long-horizon paths conditioned on 3D point cloud;
- We formalize a new set of evaluation metrics, bringing the task of OCMG to maturity for benchmark comparison within the community;
- We empirically find that our model achieves state-of-the-art predictive performance on the recently introduced PaintNet dataset [6], including object categories with high shape variability and a limited amount of training samples.

The results obtained in spray painting simulation confirm FoldPath robustness and effectiveness, indicating a high technology readiness level and its potential for real-world deployment.

II. Related Works

Spray-Painting. Finding the optimal trajectory for paint distribution across arbitrary 3D shapes exemplifies the NP-hard coverage path planning (CPP) problem. Existing solutions mainly rely on hand-crafted heuristics designed by expert engineers, which are costly and inherently limited in generalization [1], [2], [3], [4]. A few works restrict the task to simplified object geometries and apply reinforcement learning [9] or propose to adapt externally provided trajectory candidates without directly handling motion generation [4]. More recently, data-driven deep learning based methodologies demonstrated that it is possible to model expert trajectories without the need for domain-specific assumptions [5], [6]. This paves the way to more general advancements in a set of related OCMG tasks as automated surface finishing [10], and welding [11]. Still, the proposed methods combine multiple learning and post-processing stages that may hinder the smoothness and applicability of the predicted trajectories.

Robot Programming by Demonstration. The Programming by Demonstration (PbD) [12] paradigm enables robots to acquire task-specific motion skills from expert demonstrations, eliminating the need for manual trajectory design. Early PbD methods relied on kinesthetic teaching or teleoperation, wherein an operator physically guides the robot along the desired trajectory. Later, learning-based PbD methods were proposed to achieve generalization to new conditions or tasks, while attaining fast motion generation. In this context, Imitation Learning (IL) [13] methods frame PbD as a supervised learning problem where optimal actions are learned from a dataset of ground-truth data pairs, *e.g.* through regression techniques [14]. Although recent adaptations of IL investigate trajectory learning [15], multiple path generation [16], and point cloud data [17], no previous IL work copes with the OCMG problem setting—the prediction of an unknown number of output paths, of varying lengths, and uncorrelated in time (cf. Sec. II.A in [6]).

3D Deep Learning. Deep learning models can elaborate on 3D data in various forms, *e.g.* multi-view images, depth-maps, voxel grids, meshes, or point cloud. The latter are largely adopted in robotics applications thanks to their direct availability from common sensors. Dedicated network modules have been developed to manage the unordered and non-Euclidean nature of point clouds starting from point convolution [18], [19], [20] to the more recent transformers that leverage the self-attention mechanism for point cloud processing [21], [22]. Classification, segmentation, and reconstruction are among the key tasks explored in 3D deep learning. In particular, reconstruction involves processing 3D data as both input and output. Early approaches in point cloud reconstruction utilized latent codes to produce fixed-size point clouds [23]. However, these methods proved challenging to optimize and were surpassed by techniques that learn to deform point grids [24], [25]. Notably, FoldingNet [25] proposed to iteratively deform a fixed 2D grid, lifting it into higher dimensions until the desired shape is achieved. More recently, combinations of multiple small Folding-Nets and transformers demonstrated highly accurate point cloud reconstructions [26], [27].

Neural Fields. 3D signals can also be represented as continuous functions parametrized by neural networks. Neural fields are such networks that elaborate on discrete coordinate-based data and implicitly encode the underlying shape function, which can then be queried at any point [7]. These models have found applications in diverse areas, including scene reconstruction [28], grasping [29], and physics simulation [30], [31]. While deep Multilayer Perceptrons (MLPs) are commonly employed as neural fields, the incorporation of Fourier features alongside coordinate inputs has demonstrated improved performance for high-frequency signals [32]. This insight has led to the development of architectures utilizing sine [33] or other periodic activations [34]. Neural fields have traditionally represented individual instances, with each shape or scene requiring its own parameterized model. To address this limitation, research has explored conditioning neural fields on latent variables to encode multiple fields. Early approaches involved simple feature concatenation [35], while subsequent work introduced modulation techniques [36].

III. Method

Given a 3D point cloud as input, our goal is to generate the paths a spray painting robot should execute to complete the painting task. Since the number and length of paths vary significantly depending on the object geometry, a flexible and adaptive solution is required.

To address this, we propose a novel deep learning model that encodes the object's point cloud into a compact representation, which is then decoded into a predefined set of path embeddings. These embeddings act as codewords for dedicated path heads, guiding the generation of points in a 6D vector field defining the spray gun nozzle position and orientation using a conditioned neural field. Each codeword is leveraged multiple times to produce an ordered set of poses that collectively form a path. Additionally, the model predicts a confidence score for each path, enabling the selection of the most relevant ones.

The intuition at the basis of our approach is that the obtained codewords implicitly contain all the information

needed to define a continuous path curve, and each path head operates as a *sampler* and *neural field generator* conditioned by the codeword.

The sampling is guided by the user who decides on the *arbitrary number of 6D points* needed to quantize and execute the path, while the network progressively *fold* the curve composed by these points and their embeddings to fit the ground truth path. Remarkably, by feeding the network with a list of relative curve locations (interpretable as time-steps along the path execution), the output will seamlessly describe the corresponding path through its ordered components, avoiding the post-processing steps needed in previous approaches. By modeling the path head with MLP layers, we leverage their biases to generate predictions that are already smooth and ordered by design. Our approach significantly differs from current LLM-based planners, such as those described by [37], which are limited to predicting a single path in discrete steps and necessitate billions of parameters. In contrast, our simpler architecture can predict a variable number of complete paths, each with an adaptable number of points. We name our method FoldPath and present its schematic visualization in Fig. 2. In the following we formally define the model together with its optimization objective (Sec. III-A), and describe the adopted architecture (Sec. III-B). Finally, we provide details on the waypoint sampling (Sec. III-C).

A. FoldPath Overview

Let $\boldsymbol{O}$ represent the object geometry as a point cloud consisting of an arbitrary number of points in 3D space. Each object $\boldsymbol{O}$ is associated with a ground truth set of paths $\{\boldsymbol{y}_i\}_{i=1}^{n(\boldsymbol{O})}$, with object-dependent cardinality $n(\boldsymbol{O})$. FoldPath employs an encoder to map $\boldsymbol{O}$ to the visual features $\boldsymbol{z} \in \mathbb{R}^{256\times C}$. A transformer decoder then processes $\boldsymbol{z}$ alongside $N > n(\boldsymbol{O})$ path query vectors $\boldsymbol{Q}_{j=1,\dots,N} \in \mathbb{R}^C$ and outputs $\boldsymbol{P}_{j=1,\dots,N} \in \mathbb{R}^C$, which define specific path prototypes. Each of them is further elaborated by a tailored network head, which also takes as input a scalar value x_j and outputs one of the 6D points of the path $\hat{\boldsymbol{y}}_j$. By considering T scalars $s_{t=1,\dots,T} \in [-1, 1]$ and assigning $x_{j,t} = s_t$, the output set $\hat{\boldsymbol{y}}_{j,t=1,\dots,T}$ will represent 6D points sampled from the predicted j-th path. Specifically, we define $\hat{\boldsymbol{y}}_{j,t} = \{\hat{\boldsymbol{p}}_{j,t}, \hat{\boldsymbol{v}}_{j,t}\}$ with the two sub-vectors respectively representing 3D position and orientation of each 6D point.

Considering the difference in cardinality between the predicted paths and ground truth ones, a matching procedure is needed. The ground truth paths are sampled by using the same set $s_{t=1,\dots,T}$ adopted for the predictions. Here the scalars are interpreted as relative positions along the path: -1 corresponds to the first point, 0 to the middle, and 1 as the last one, with other intermediate points derived via linear interpolation. Thus, we obtain the set of reference ground truth points $\boldsymbol{y}_{i,t=1,\dots,T}$ and assign a confidence score $f_{i=1,\dots,n(\boldsymbol{O})} = 1$ to all the corresponding paths.

To perform a bipartite matching, we add extra $N - n(\boldsymbol{O})$ zero-padded paths with score $f_{i=n(\boldsymbol{O})+1,\dots,N} = 0$ and we search among all permutations of N elements $\mathfrak{S}_N$ via the Hungarian algorithm by relying on the 3D point locations:

$$\hat{\sigma} = \arg\min_{\sigma\in\mathfrak{S}_N} \sum_i^N \mathcal{L}_{\text{match}}(\boldsymbol{y}_{\sigma(i)}, \hat{\boldsymbol{y}}_i) \ , \tag{1}$$

$$\text{where} \quad \mathcal{L}_{match} = \mathbb{1}_{f_i\neq 0}(||\boldsymbol{p}_{\sigma(i)} - \hat{\boldsymbol{p}}_i||_2) \ . \tag{2}$$

Finally, the overall objective function of FoldPath is $\mathcal{L} = \mathcal{L}_{points} + \mathcal{L}_{conf}$ defined by leveraging the optimal matching $\hat{\sigma}$ and the predicted confidence score $\hat{f}_i$ with

$$\mathcal{L}_{points} = \textstyle\sum_i^N \mathbb{1}_{f_i\neq 0} \left\{||\boldsymbol{p}_{\hat{\sigma}(i)} - \hat{\boldsymbol{p}}_i||_2 + 1 - \frac{\boldsymbol{v}_{\hat{\sigma}(i)}\cdot\hat{\boldsymbol{v}}_i}{||\boldsymbol{v}_{\hat{\sigma}(i)}||_2\cdot||\hat{\boldsymbol{v}}_i||_2}\right\} \tag{3}$$

$$\mathcal{L}_{conf} = \sum_i^N -(1-\hat{f}_i)^\gamma \log \hat{f}_i \ . \tag{4}$$

In short, the *points* loss evaluates the 6D points set similarity by combining the Euclidean distance between their 3D location and the angular (cosine) distance between the corresponding orientation vectors. The *confidence* prediction correctness is measured by the focal loss [38] with $\gamma = 2$.

B. Architecture

We design FoldPath to process 3D point clouds of objects and learn an implicit representation of path curves, enabling arbitrary sampling of the generated paths. An initial encoder downsamples the point cloud by using point-convolution layers and then upsamples it via feature propagation layers to output $\boldsymbol{z}$ that we interpret as visual features. A transformer decoder operates on $\boldsymbol{z}$, and on the path queries $\boldsymbol{Q}_j$ by alternating layers of self-attention between the path queries and cross-attention between the path queries and visual features to generate the path propotypes $\boldsymbol{P}_j$. The queries $\boldsymbol{Q}_j$ are randomly initialized and then learned during training.

Each path prediction head takes as input a single path prototype $\boldsymbol{P}_j$ and elaborates the initial scalar value which represents the progress along the path via L stacked modulated [36] MLP blocks. So, starting from $\boldsymbol{x}_{j,t}^{l=0} = x_{j,t}$ we obtain $\boldsymbol{x}_{j,t}^{l=1}, \dots, \boldsymbol{x}_{j,t}^{l=L}$. More precisely, each block receives an embedding $\boldsymbol{x}_{j,t}^{l}$ and creates the next one $\boldsymbol{x}_{j,t}^{l+1}$ as follows

$$\boldsymbol{x}_{j,t}^{l+1} = \boldsymbol{h}_j^l \odot act(\boldsymbol{A}^l \cdot \boldsymbol{x}_{j,t}^l) \ , \tag{5}$$

where l is the block index, act is a non-linear activation function that will be discussed later, and $\boldsymbol{A}^l$ is a linear layer with dimension $1 \times H$ for $l = 0$ and $H \times H$ otherwise. Specifically, $\boldsymbol{h}_j^l$ is the hidden representation that modulates the l-th MLP block depending on the input path prototype $\boldsymbol{P}_j$, defined by

$$\boldsymbol{h}_j^0 = \text{ReLU}(\boldsymbol{w}^0\boldsymbol{P}_j), \quad \boldsymbol{h}_j^{l+1} = \text{ReLU}(\boldsymbol{w}^{l+1}\left[h_j^l\boldsymbol{P}_j\right]^\top). \tag{6}$$

Here $\boldsymbol{w}^0$ has dimension $H \times C$, while for $l \neq 0$, $\boldsymbol{w}^l$ has dimension $H \times (H + C)$. The last MLP block outputs $\boldsymbol{x}_{j,t}^L$ which is then fed to a final linear layer $\boldsymbol{A}^{L+1}$ that maps it to the 6D prediction $\hat{\boldsymbol{y}}_{j,t}$.

Regarding the choice of act, we follow the neural field literature, which also inspired our use of modulated blocks for continuous visual signal representation. Previous work [33], [36], [34] demonstrated that activation functions belonging to the sinusoidal family are particularly suitable

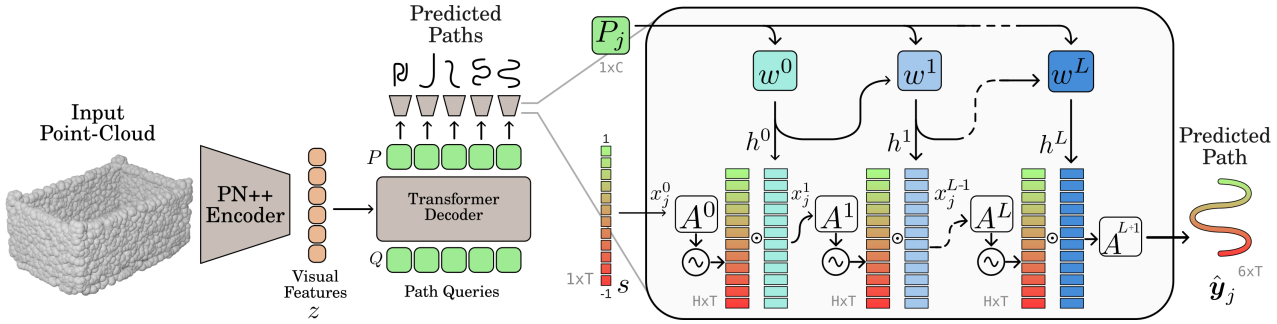

Fig. 2. Schematic visualization of FoldPath. It first encodes the input point cloud using a PointNet++ backbone [19], then it elaborates the obtained visual features $\boldsymbol{z}$ with a transformer decoder together with the learned queries $\boldsymbol{Q}_j$ to obtain path embeddings $\boldsymbol{P}_j$. Finally, every path-specific neural field inspired head is fed with $\boldsymbol{P}_j$ and a scalar $x_{j,t}$ to yield the t-th 6D pose $\hat{\boldsymbol{y}}_{j,t}$ of the output path. The whole path is generated by sampling T scalars $s_{t=1,\dots,T} \in [-1, 1]$.

when dealing with signals with high-frequency components. Thus, we adopted siren activations [33] and finer activations [34] in addition to the standard ReLU.

Finally, the path head further elaborates on $\boldsymbol{P}_j$ via a two-layer Feed Forward Network ending with a sigmoid activation that outputs the confidence score $\hat{f}_j$.

C. Waypoint Sampling

We described the model flow by considering the path quantized in $t = 1, \dots, T$ waypoint components. Still, we highlight that the network can elaborate a whole point sequence in parallel without any downside other than memory consumption. Indeed, to let the model reason on path curves during training, we interpolate the ground truth path points to then re-sample the underlying curve at an arbitrary rate. We adopt a uniform random sampling in training as it facilitates the learning dynamics (see Sec. V-E) and at test time we use an equidistant point list by setting $s_t = -1 + t\frac{2}{T}$.

IV. Metric

Spray painting paths are inherently continuous curves. Measuring the performance of a path prediction model that outputs discrete sequences of 6D poses requires applying an analogous sampling on the ground truth paths and the adoption of set-to-set metrics as the point-wise chamfer distance (PCD) [39]. This is the evaluation strategy used in previous work [5][6] that, however, depends on the sampling rate, disregards the point order as well as the identity of each path, and focuses on a preliminary stage before the refinement needed to determine the spray plan for execution. Moreover, the PCD exhibits a critical sensitivity to outliers, with even a few inaccurate points leading to a skewed PCD mean. Thus, the assessment may become unreliable, particularly in scenarios requiring fine-grained comparisons.

To address these limitations, we propose a novel Average Precision (AP) metric based on the F-score and leveraging the Dynamic Time Warping (DTW) distance [40]. Calculating the F-Score is a commonly used strategy in point cloud prediction tasks [41], [26] with the chamfer distance used to obtain the point matching, but we adopt the DTW to ensure the correct point ordering.

More formally, given the ground truth and the predicted 3D point lists $\boldsymbol{p} \in \mathbb{R}^{K\times 3}$ and $\hat{\boldsymbol{p}} \in \mathbb{R}^{M\times 3}$, we find the optimal match between each component point such that the DTW distance is minimized. This distance encodes both the point locations and their index order in the list. In this way, similar paths with low distances will have the same shape and point sorting, while the metric remains agnostic to different sampling rates as each point from one set can match multiple points from the other and vice versa. For simplicity, we use the wildcard $*$ to indicate both the DTW ground truth match for a predicted point (precision) and the DTW predicted point match for a ground truth point (recall). So when comparing the two 6D path poses corresponding to the 3D point lists, we have F-score$= 2(pre \times rec)/(pre + rec)$, where

$$rec = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}(||\boldsymbol{p}_k - \hat{\boldsymbol{p}}_*||_2 < \delta, \measuredangle(\boldsymbol{v}_k, \hat{\boldsymbol{v}}_*) < \theta) \tag{7}$$

$$pre = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}(||\hat{\boldsymbol{p}}_m - \boldsymbol{p}_*||_2 < \delta, \measuredangle(\hat{\boldsymbol{v}}_m, \boldsymbol{v}_*) < \theta)\ . \tag{8}$$

Here we use two thresholds for the Euclidean and angular distances with δ set to 0.025 in normalized space, and $\theta = 10°$, similarly to [26]. In summary, the F-score provides values in $[0, 1]$ that express the similarity between two paths of arbitrary length. By using this score, we can calculate the precision-recall curve and compute the AP, thus evaluating the capability of any path prediction model. For the spray painting task, path execution does not have a preferential direction as long as the points along a path are correctly ordered. Hence, we calculate the F-score considering both the order with which the path points are generated and its reverse, keeping the highest value. For the experimental evaluation, we report two AP metrics. AP^{50}_{DTW} is obtained by considering that a path is correct when the F-Score is at least 0.5, and represents the case when prediction and ground truth share at least 50% of the points. AP_{DTW} is the mean of APs calculated with a threshold ranging from 0.5 to 0.95 with 0.05 step increments: this represents a comprehensive metric accounting for the overall quality of the predictions.

V. Experiments

In our experimental analysis, we assess the capabilities of our method in predicting paths that are ready for direct execution on a robot. Indeed, FoldPath does not require a

post-processing phase on the output paths, which is instead essential for existing approaches. We consider the spray painting task as a relevant industrial use case and thoroughly evaluate the performance of FoldPath, further conducting ablation studies to gain deeper insights into its inner workings.

A. Dataset and Baselines

We align with previous work [5], [6] and adopt the PaintNet dataset [6] for our experimental benchmark, covering four object categories of increasing complexity: cuboids, windows, shelves, and containers. Each data sample includes the input object as a 3D point cloud of 5120 points, and the set of associated ground-truth paths encoded as a discrete sequence of 6D waypoints. More precisely, 6D poses describe the ideal paint deposition location in (x, y, z) coordinates, and the end-effector orientation of the gun nozzle as a 3D unit vector (2-DoF representation).

We consider three reference baselines introduced in [6], briefly described below for clarity. *Path-wise* directly outputs a set of paths, as fixed sequences of 6D end-effector poses. For each pose and each path, a confidence score is simultaneously predicted. As a result, this model attempts to autonomously retain the correct number and lengths of output paths in end-to-end fashion, similarly to our approach. *Autoregressive* starts by predicting a set of initial configurations for each output path. Then, each path is independently generated in an autoregressive manner, by predicting 6D poses and a termination probability. *MaskPlanner* [6] predicts a set of path segments, *i.e.* short sequences of 6D poses. Its key strength lies in generating path segments as local sub-goals for the robot to follow. However, MaskPlanner relies on a final post-processing step for sub-goal concatenation, which may ultimately lead to unfeasible paths.

B. Implementation Details

We train FoldPath separately for each object category for 200 epochs with a batch size of 24. We use the Adam optimizer [42] and cosine annealing scheduling (with learning rate from 3e-4 to 1e-8). The encoder is a PointNet++ backbone [19], the same used in [5], [6], with the last feature propagation layer removed. The encoder is trained from scratch. The dimension of the visual features $\boldsymbol{z}$ is $256 \times C$ with $C = 384$. The transformer decoder has 4 layers with 4 heads, an hidden dimension of C and takes as input $N = 40$ learned path queries. The path head has $L = 4$ layers and an hidden dimension of H=512. The waypoint cardinality is set to T=64 in training and T=384 at testing.

We implement three different versions of our model: FoldPath (*ReLU*), FoldPath (*Siren*), and FoldPath (*Finer*). Inspired by the blocks in the decoder of FoldingNet [25], every layer in our path head is composed of linear, ReLU, and layer norm. The Siren and Finer versions include respectively sine activation [33] and finer activation [34] instead of ReLU and don't use normalization due to the constrained nature of the activation functions.

TABLE I

COMPARISON BETWEEN BASELINES AND FOLDPATH ON THREE CATEGORIES OF THE PAINTNET BENCHMARK [6]. MASKPLANNER SHOWS TOP PCD BUT FALLS OFF WHEN POST-PROCESSING IS APPLIED. THE FULL PATH AP INDICATES SUPERIOR RESULTS FOR FOLDPATH.

CUBOIDS			
Model	$PCD \downarrow$	$AP^{50}_{\text{DTW}} \uparrow$	$AP_{\text{DTW}} \uparrow$
Autoregressive	33.2	2.1	0.0
Path-Wise	48.0	32.9	10.8
MaskPlanner (W/O Post)	6.5	-	-
MaskPlanner (W Post)	19.9	80.0	50.3
FoldPath (*ReLU*)	11.2	59.8	35.2
FoldPath (*Siren* [33])	9.2	97.5	60.3
FoldPath (*Finer* [34])	**5.5**	**99.2**	**91.1**
WINDOWS			
Model	$PCD \downarrow$	$AP^{50}_{\text{DTW}} \uparrow$	$AP_{\text{DTW}} \uparrow$
Autoregressive	64.1	10.5	4.0
Path-Wise	45.5	28.0	11.1
MaskPlanner (W/O Post)	**6.8**	-	-
MaskPlanner (W Post)	336.0	70.4	50.4
FoldPath (*ReLU*)	106.7	91.4	71.8
FoldPath (*Siren* [33])	114.3	91.3	71.9
FoldPath (*Finer* [34])	104.6	**91.9**	**75.0**
SHELVES			
Model	$PCD \downarrow$	$AP^{50}_{\text{DTW}} \uparrow$	$AP_{\text{DTW}} \uparrow$
Autoregressive	40.45	23.6	8.5
Path-Wise	46.7	9.7	2.5
MaskPlanner (W/O Post)	**7.4**	-	-
MaskPlanner (W Post)	1470	35.1	26.0
FoldPath (*ReLU*)	65.8	88.4	75.4
FoldPath (*Siren* [33])	64.7	89.5	78.0
FoldPath (*Finer* [34])	65.9	**91.3**	**84.3**

C. Results on Cuboids, Windows, Shelves

We run a first set of experiments on the three object categories with synthetic spray painting paths: cuboids, windows, and shelves with the same train/test splits of [6]. For MaskPlanner we consider both the final prediction after post-processing (W Post) and its raw output without post-processing (W/O Post). The latter consists of short sequences of 6D points that are grouped as part of specific paths but remain unconnected. As a result, they cannot be directly evaluated as paths using the AP metric.

The results in Tab. I indicate that for the cuboids MaskPlanner provides much lower PCD than the Autoregressive and Path-Wise baselines. In terms of AP, MaskPlanner (W Post) outperforms the competitor baselines and also FoldPath with ReLU activations. Still, when passing to the Siren version, FoldPath improves and gets to top performance when adopting Finer activations. High AP^{50}_{DTW} values indicate a high number of correct paths, while high AP_{DTW} means that the paths generated by the model are overall accurate.

For windows and shelves the AP results show a trend similar to that observed on cuboids, but the best PCD is that provided by MaskPlanner (W/O Post). On one hand, filtering and concatenating the predicted pose segments during post-processing significantly affect the PCD, and on the other provide paths with better AP than the baseline competitors (Autoregressive and Path-Wise), but largely worse than FoldPath in all its versions.

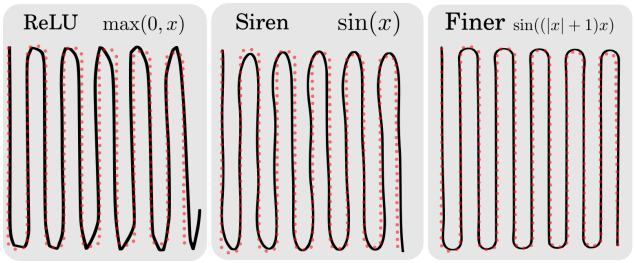

Fig. 3. Qualitative comparison of a single path from the cuboids test set between various activation functions for our FoldPath model. In black the predictions, in light red the ground truth. ReLU activation yields sharp corners, undesirable for robotic applications. Siren activation yields good curves but lacks precision. Finer performs best overall.

In general, we can state that the AP metric reflects a model's ability to generate coherent and smooth paths—aspects on which FoldPath excels. The Autoregressive and Path-Wise baselines generate entire paths directly, but their performance is limited by optimization challenges: Path-Wise struggles with the complexity of high-dimensional outputs, whereas Autoregressive faces difficulties in making accurate long-horizon predictions, particularly for cuboids.

Overall, the optimization challenges are also evident from the training time needed by these methods as well as by MaskPlanner, as they all require six times the number of epochs to converge in respect to FoldPath.

Regarding FoldPath, we can further discuss the role of the chosen activation functions. ReLU achieves top results on tasks with a prevalence of straight paths (windows and shelves) but fails when the paths are highly complex and require multiple curves (cuboids). The Siren activation performs well overall but tends to favor wavy paths, leading to reduced precision in straight path sections. Finer activations combine both characteristics and perform well on all object categories, confirming the suitability of sinusoidal activations for high-frequency predictions (see Fig. 3).

In Fig. 4, we present qualitative results for all baselines and the best version of FoldPath. Path-Wise exhibits unprecise paths that deviate significantly from the ground truth. Autoregressive predictions are qualitatively good but are subtly imprecise, confirming what already observed from the quantitative results. MaskPlanner benefits significantly from post-processing, but small errors in the original prediction cause significant deviations in the final paths. FoldPath generates smooth and precise paths, thanks to its ability to implicitly encode the path curve rather than restricting its focus to discrete poses.

D. Results on Containers

Here, we zoom in on the challenging container class of the PaintNet dataset, which is limited to 88 samples (70 train, 18 test) collected from a real industrial scenario. Container instances particularly show high shape variability (*e.g.*, wavy and grated surfaces), and are paired with spray painting paths that are irregular due to the recording from real-world manual executions. To fairly evaluate the performance of the path generative models on this testbed, we adopt a version of the AP metric that is more lenient to errors. We define

TABLE II

RESULTS FOR THE CHALLENGING CONTAINER CLASS IN PAINTNET [6]. MASKPLANNER OBTAINS THE BEST PCD BUT IT IS NEGATIVELY AFFECTED BY POST-PROCESSING. FOLDPATH SHOWS TOP AC RESULTS THAT CORRELATE WITH THE EFFECTIVENESS IN PAINT COVERAGE.

Model	$PCD \downarrow$	$AP^{easy}_{\text{DTW}} \uparrow$	Paint Cov. % $\uparrow$
Path-Wise	556.0	9.5	58.4
Autoregressive	708.7	9.2	53.3
MaskPlanner (W/O Post)	**220.7**	-	90.9
MaskPlanner (W Post)	1483.0	9.6	29.5
FoldPath	584.1	**13.7**	**91.1**

TABLE III

AP_{DTW} ON THE PAINTNET BENCHMARK FOR DIFFERENT DESIGN CHOICES OF FOLDPATH. IN BOLD THE ADOPTED CONFIGURATION.

Training Sampling Strategy			
Sampling Strategy	CUBOIDS	WINDOWS	SHELVES
Equispaced	88.0	74.6	85.3
Noisy Equispaced	90.9	75.4	86.1
Uniform Noise	91.1	75.0	84.3
Decoder Size			
Hid. Dim / Num. Layers	CUBOIDS	WINDOWS	SHELVES
Tiny (256/2)	86.6	58.6	79.7
Small (256/4)	89.0	60.2	83.9
Medium (512/4)	91.1	75.0	84.3
Large (512/6)	91.4	66.5	85.6
Conditioning Strategy			
Strategy	CUBOIDS	WINDOWS	SHELVES
FoldingNet Concat. [25]	14.4	46.6	56.4
Modulation [36]	91.1	75.0	84.3

AP^{easy}_{DTW} that sweeps the DTW threshold from 0.05 to 0.5 instead of 0.5 to 1.0. This change maintains all the desired metric properties described in Sec. IV, but with a lower number of points needed to match between the ground truth path and the model output to be considered as a correct prediction. Following the protocol defined in [5], we also use a proprietary simulator developed by EFORT to test the actual ability of FoldPath to paint the real objects. We report the paint coverage (as defined in [6]) in percentage points averaged across all test instances.

We assess our model's performance by considering its version with finer activations and training for 200 epochs, a sixth of the baseline's training. Tab. II reveals that, despite the limitations highlighted by the PCD results, our method achieves improved AP^{easy}_{DTW} scores compared to baselines. The obtained paint coverage percentage further confirms the effectiveness of FoldPath. The inverse trend between the paint coverage and the PCD results confirms that the latter metric is not really informative on the model's effectiveness for the spray painting task. The qualitative results in Fig. 5 indicate that, even if the generated paths do not fully match expert-defined optimal paths, the predictions of FoldPath capture essential path qualities for practical applications.

E. Ablation Studies

The design of FoldPath requires several model choices that can impact the final results. We already discussed the role

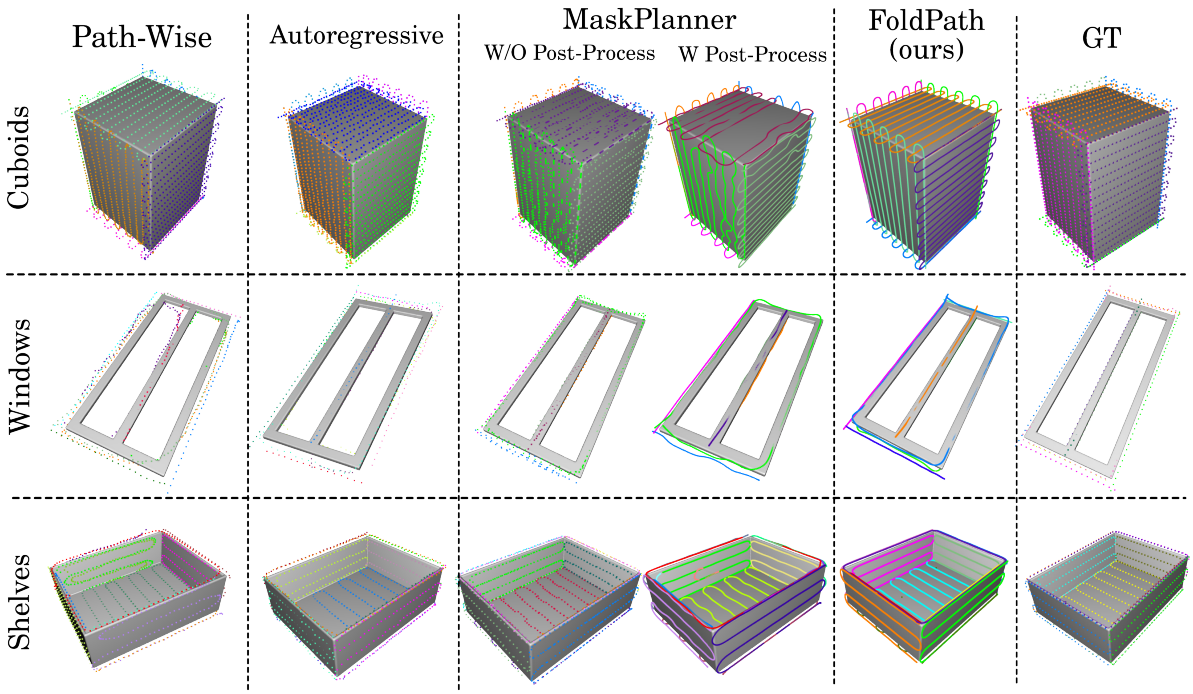

Fig. 4. Qualitative results on randomly chosen samples from the three main categories of the PaintNet benchmark. Path-wise struggles with long paths. Autoregressive paths are coherent but may miss details. MaskPlanner is able to generate highly precise paths but the post-processing plays a key role in producing accurate results. FoldPath overall outputs the clearest paths without need of any post-processing step.

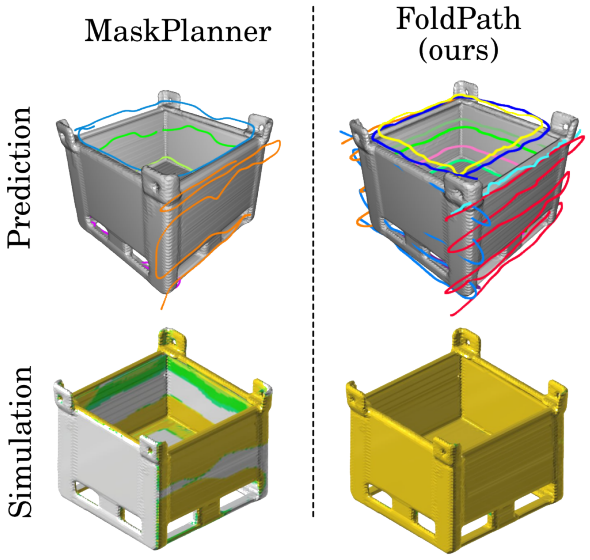

Fig. 5. Qualitative evaluation in simulation for a random sample from the container test set. The colormap ranges from green (low) to yellow (high) paint deposition. Some paths can be hard to see due to compenetration with object mesh, but in practice the nozzle is 12 cm far from the object so even compenetrating paths yield correct painting executions.

of the activation functions, we focus here on the waypoint sampling strategy, on the size of the path head and the technique used to condition the path head with the path codewords obtained from the transformer decoder. In Tab. III we present the results obtained while investigating these aspects. We focus on cuboids, windows, and shelves where we have a sizable amount of samples in training that allows to properly assess the model choices. Moreover, we consider only the AP metric that better reflects the accuracy of the predicted paths over the whole test set.

When selecting the T scalars $s_{t=1,\dots,T} \in [-1, 1]$ and assigning $x_{j,t} = s_t$ at training time, the exact selection procedure has minimal effect on the model's outcome. This is the conclusion we can draw from the top part of Tab. III, where we consider simple equally-spaced points (corresponding to the same strategy used for sampling at test time), a uniform random selection, and an intermediate case where random Gaussian noise is added to the equispaced values. As there isn't a clearly superior single strategy, all the choices are equally valid and we adopt the uniform noise everywhere.

Regarding the size of the path head, we experiment with different values of the hidden dimension H and the number of layers L. The results in the middle part of Tab. III highlight that a small network module has limited capacity, while a large one shows diminishing returns or may under-perform due to high optimization complexity. This supports the choice of the Medium setting for our experiments.

Finally, we examine the role of the modulation strategy inspired by [36] that we use to condition the path heads with the path codewords. It differs from the solution adopted in FoldingNet [25], according to which the path codeword $\boldsymbol{P}_j$ is simply concatenated to $\boldsymbol{x}_j$ as input to every layer. From the bottom part of Tab, III it is evident that the modulation is much more effective than the concatenation solution which suffers especially when dealing with long paths as for the cuboids. The reason lies in the heavy dimensionality reduction applied on the codewords to maintain a manageable

dimensionality of the concatenated features, consequently limiting the expressive abilities of the path head.

VI. Conclusions

In this work we propose FoldPath, a novel deep learning model for end-to-end Object-Centric Motion Generation from 3D point clouds. Inspired by concepts from the neural field literature, our approach successfully predicts a large number of smooth, long-horizon paths in a single forward pass, without the need for costly heuristics or sensitive post-processing steps. Moreover, we introduce tailored evaluation metrics for OCMG that provide useful insights into the real functioning of the model and support fair benchmark comparisons. Extensive experimental evaluations demonstrated the effectiveness of FoldPath for the spray painting task.

We believe this contribution is highly relevant to the broader OCMG problem and plan to extend our findings to other tasks that feature geometrically complex paths, such as robotic welding and automated multi-UAV visual inspection.

References

[1] N. Xi and Y. Chen, "Multi-objective optimal robot path planning in manufacturing," in *IROS*, 2003.

[2] G. Biegelbauer, A. Pichler, M. Vincze, C. L. Nielsen, H. J. Andersen, and K. Häusler, "The inverse approach of flexpaint [robotic spray painting]," *IEEE RAM*, vol. 12, pp. 24–34, 2005.

[3] X. Li, O. A. Landsnes, H. Chen, S. M-V, T. A. Fuhlbrigge, and M.-A. Rege, "Automatic trajectory generation for robotic painting application," in *ROBOTIK*, 2010.

[4] D. Gleeson, S. Jakobsson, R. Salman, F. Ekstedt, N. Sandgren, F. Edelvik, J. S. Carlson, and B. Lennartson, "Generating optimized trajectories for robotic spray painting," *IEEE TASE*, 2022.

[5] G. Tiboni, R. Camoriano, and T. Tommasi, "Paintnet: Unstructured multi-path learning from 3d point clouds for robotic spray painting," in *IROS*, 2023.

[6] ——, "Maskplanner: Learning-based object-centric motion generation from 3d point clouds," 2025.

[7] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural fields in visual computing and beyond," in *Computer Graphics Forum*, vol. 41, no. 2, 2022, pp. 641–676.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[9] J. Kiemel, P. Yang, P. Meißner, and T. Kröger, "Paintrl: Coverage path planning for industrial spray painting with reinforcement learning," in *RSS WS*, 2019.

[10] A. Caggiano, R. Teti, V. Alfieri, and F. Caiazzo, "Automated laser polishing for surface finish enhancement of additive manufactured components for the automotive industry," *Production Engineering*, vol. 15, no. 1, pp. 109–117, 2021.

[11] P. Zhou, R. Peng, M. Xu, V. Wu, and D. Navarro-Alarcon, "Path planning with automatic seam extraction over point cloud models for robotic arc welding," *IEEE RAL*, vol. 6, no. 3, pp. 5002–5009, 2021.

[12] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot programming by demonstration*, 2008, pp. 1371–1394.

[13] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, *An algorithmic perspective on imitation learning*. Now Publishers, Inc., 2018, ch. 3.

[14] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[15] A. Duan, I. Batzianoulis, R. Camoriano, L. Rosasco, D. Pucci, and A. Billard, "A structured prediction approach for robot imitation learning," *IJRR*, vol. 43, no. 2, pp. 113–133, 2024.

[16] M. Srinivasan, A. Chakrabarty, R. Quirynen, N. Yoshikawa, T. Mariyama, and S. D. Cairano, "Fast multi-robot motion planning via imitation learning of mixed-integer programs," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 598–604, 2021.

[17] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.

[20] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *ICCV*, 2019.

[21] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *ICCV*, 2021.

[22] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler faster stronger," in *CVPR*, 2024.

[23] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *ICML*, 2018.

[24] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *CVPR*, 2018.

[25] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *CVPR*, 2018.

[26] X. Yu, Y. Rao, Z. Wang, J. Lu, and J. Zhou, "Adapointr: Diverse point cloud completion with adaptive geometry-aware transformers," *TPAMI*, vol. 45, no. 12, pp. 14 114–14 130, 2023.

[27] F. Duan, J. Yu, and L. Chen, "T-corresnet: Template guided 3d point cloud completion with correspondence pooling query generation strategy," in *ECCV*, 2024.

[28] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[29] K. Karunratanakul, J. Yang, Y. Zhang, M. J. Black, K. Muandet, and S. Tang, "Grasping field: Learning implicit representations for human grasps," in *3DV*, 2020.

[30] S. L. Cleac'h, H. Yu, M. Guo, T. A. Howell, R. Gao, J. Wu, Z. Manchester, and M. Schwager, "Differentiable physics simulation of dynamics-augmented neural objects," *RA-L*, vol. 8, pp. 2780–2787, 2022.

[31] M. Z. Irshad, M. Comi, Y.-C. Lin, N. Heppert, A. Valada, R. Ambrus, Z. Kira, and J. Tremblay, "Neural fields in robotics: A survey," 2024.

[32] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *NeurIPS*, 2020.

[33] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *NeurIPS*, 2020.

[34] Z. Liu, H. Zhu, Q. Zhang, J. Fu, W. Deng, Z. Ma, Y. Guo, and X. Cao, "Finer: Flexible spectral-bias tuning in implicit neural representation by variable-periodic activation functions," in *CVPR*, 2024.

[35] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019.

[36] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker, "Modulated periodic activations for generalizable local functional representations," in *ICCV*, 2021.

[37] M. K. et. al, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *CVPR*, 2017.

[39] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *TPAMI*, vol. 43, no. 12, 2020.

[40] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[41] L. P. Tchapmi, V. Kosaraju, S. H. Rezatofighi, I. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *CVPR*, 2019.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014.