POLITECNICO DI TORINO
Repository ISTITUZIONALE

MaskPlanner: A Framework for 3D Learning-Based Object-Centric Motion Generation and Applications to Robotic Spray Painting

Original
MaskPlanner: A Framework for 3D Learning-Based Object-Centric Motion Generation and Applications to Robotic Spray Painting / Tiboni, G., Camoriano, R., Tommasi, T.. - In: IEEE TRANSACTIONS ON ROBOTICS. - ISSN 1552-3098. - 42:(2026), pp. 2576-2595. [10.1109/TRO.2026.3699455]

Availability:
This version is available at: 11583/3017200 since: 2026-10-29T12:06:40Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/TRO.2026.3699455

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

25 November 2026

MaskPlanner: A Framework for 3D Learning-Based Object-Centric Motion Generation and Applications to Robotic Spray Painting

Gabriele Tiboni, Raffaello Camoriano, *Member, IEEE*, and Tatiana Tommasi

***Abstract*—Object-centric motion generation (OCMG) plays a key role in a variety of industrial applications—such as robotic spray painting and welding—requiring efficient, scalable, and generalizable algorithms to plan multiple long-horizon trajectories over free-form 3D objects. However, existing solutions rely on specialized heuristics, expensive optimization routines, or restrictive geometry assumptions that limit their adaptability to real-world scenarios. In this work, we introduce a novel, fully data-driven framework that tackles OCMG directly from 3D point clouds, learning to generalize expert path patterns across free-form surfaces. We propose *MaskPlanner*, a deep learning method that predicts local path segments for a given object while simultaneously inferring "path masks" to group these segments into distinct paths. This design induces the network to capture both local geometric patterns and global task requirements in a single forward pass. Extensive experimentation on a realistic robotic spray painting scenario shows that our approach attains near-complete coverage (above 99%) for unseen objects, while it remains task-agnostic and does not explicitly optimize for paint deposition. Moreover, our real-world validation on a 6-DoF specialized painting robot demonstrates that the generated paths are directly executable and yield expert-level painting quality. We additionally provide empirical evidence that our approach remains complementary to downstream trajectory optimization methods, and applicable to tasks beyond spray painting.**

***Index Terms*—3D learning, deep learning, imitation learning (IL), motion generation.**

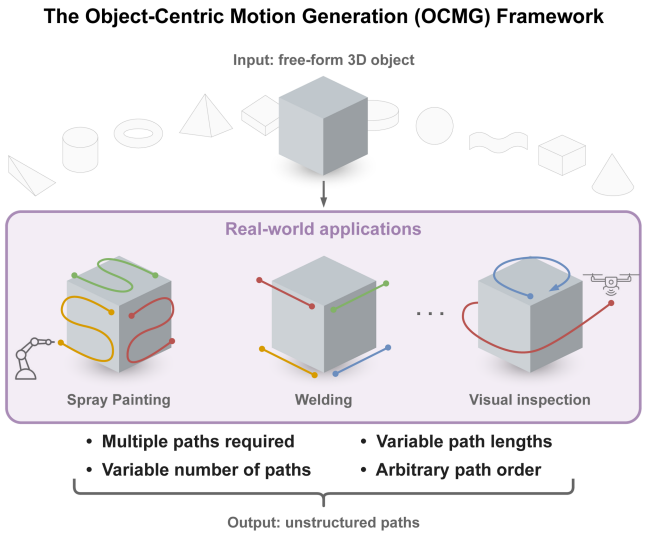

Fig. 1. OCMG framework unifies several robotic applications that share common assumptions on the desired output paths—referred to as unstructured paths.

I. Introduction

MOTION generation conditioned on 3D objects is fundamental to numerous industrial robotic applications, including spray painting, welding, sanding, and cleaning. Despite the different objectives, these tasks share key challenges arising from the complexity of free-form 3D inputs and the high-dimensional outputs required to define complete robot programs. In particular, the offline generation of long-horizon motions demands substantial computational resources for both optimization and planning. In addition, encoding human expert behavior into explicit optimization objectives remains extremely challenging for such complex tasks. To address these difficulties, robotics practitioners often rely on task-specific knowledge, impose strong simplifying assumptions regarding the object shapes, and develop heuristic algorithms to render each task individually more tractable. However, these solutions necessitate extensive re-engineering for each new product, making the process time-consuming, costly, and unable to efficiently adapt to new scenarios. In this context, establishing a unifying paradigm to tackle these tasks is a crucial step for the community to transition from handcrafted strategies to techniques that are scalable and capable of generalization.

To this end, we formalize the *object-centric motion generation* (OCMG) framework, a novel problem setting that unifies robotic tasks aiming to generate *multiple*, *long-horizon* paths

Received 31 August 2025; revised 27 January 2026; accepted 14 April 2026. Date of publication 2 June 2026; date of current version 29 June 2026. This study was carried out within the FAIR – Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (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. This article was recommended for publication by Associate Editor R. Detry and Editor J. Kober upon evaluation of the reviewers' comments. *(Corresponding author: Raffaello Camoriano.)*

Gabriele Tiboni and Tatiana Tommasi are with Politecnico di Torino, 10138 Turin, Italy (e-mail: gabriele.tiboni@polito.it; tatiana.tommasi@polito.it).

Raffaello Camoriano is with Politecnico di Torino, 10138 Turin, Italy, and also with the Istituto Italiano di Tecnologia, 16163 Genoa, Italy (e-mail: raffaello.camoriano@polito.it).

Code and data are available at https://github.com/vandal-lab/MaskPlanner.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2026.3699455, provided by the authors.

Digital Object Identifier 10.1109/TRO.2026.3699455

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

based on *static*, *free-form* 3D objects. For the sake of a general formulation, we consider output paths to be *unstructured:* no predefined path order is assumed, and paths can generally vary in number and length according to the input object. Notably, these properties are shared by a wide range of robotic applications—as illustrated in Fig. 1—where negligible dynamic interactions with the 3D objects are involved and global object geometric information is given.

Robotic spray painting is an industrially relevant example of the OCMG problem: multiple paths are necessary to paint a single object, and the resulting paint coverage (PC) is invariant with respect to the order of execution—e.g., separate paths can be executed in parallel across multiple robots. Furthermore, motion generation can occur offline, and no reactive planning is needed since global information of the surface geometry is available. Note that the pattern of the spray painting paths varies significantly with each object instance, making it difficult to codify general rules—e.g., human experts in the field rely on sophisticated, high-level reasoning and costly trial-and-error to determine robot programs based on the 3D geometry of the objects. Existing research studies propose decoupling the spray painting task into 1) partitioning the 3D object into convex components, and 2) performing offline trajectory optimization using either domain-specific heuristics [1], [2], [3], [4], [5], [6], or reinforcement learning (RL)-based methods [7]. Yet, such approaches make simplified premises on the structure of output trajectories, require expensive optimization routines, and are heavily tailored to specific shapes and convex surfaces only. These issues leave robotic spray painting solutions largely limited in flexibility and generalization capabilities, despite their relevance in product manufacturing.

Recently, interest in purely data-driven approaches has grown for extrapolating path patterns without the need to explicitly encode optimization objectives and task-specific constraints. Assuming that expert data is available, learning-based methods pave the way for solutions that are scalable, cheap to deploy at inference time, and generalizable to unseen scenarios. A number of successful applications demonstrate the potential of data-driven methods in related problems, such as motion forecasting in autonomous driving [8], [9], [10], [11], multiagent imitation learning (IL) [12], and socially compliant robot navigation [13], [14], [15]. Yet, these solutions consider a short prediction horizon, assume a fixed number of paths, and do not deal with 3D input data. In [16], the first approach is proposed to handle unstructured paths conditioned on 3D objects, predicting disconnected path segments across the object surface rather than directly inferring long-horizon paths. This approach allows for accurate local predictions, but crucially lacks a way to organize predicted segments into separate paths, and finally concatenate them to generate long-horizon paths.

In this work, we present *MaskPlanner*, a novel deep learning method to address OCMG tasks directly from expert data. Our pipeline breaks down the motion generation problem into the joint prediction of 1) path segments and 2) path masks, in a single forward pass. Particularly, we propose learning binary masks over predicted segments to identify which path each segment belongs to. This strategy allows the network to simultaneously make local (segment predictions) and global (mask predictions) planning decisions in one step. In turn, our method effectively infers the required *number* of paths and the *length* of each path according to a given input object.

We focus on spray painting as a representative application to conduct a thorough experimental evaluation. In this domain, MaskPlanner is capable of predicting segments and masks for 40 paths in only 100 ms, spanning a total length of 70 meters and 8 minutes of execution time, and achieving near-optimal PC on held-out 3D objects in simulation. Moreover, we successfully execute the generated paths on a real 6-DoF specialized painting robot for previously unseen object instances, achieving qualitative results that are indistinguishable from those obtained via human-expert trajectories.

Our novel contributions can be summarized as follows.

1) *OCMG framework*: We formalize the OCMG problem setting to provide a unifying ground for the community to devise solutions that target a wide range of robotic tasks.
2) *Joint segment and mask predictions*: We propose MaskPlanner, a novel deep learning method that predicts path segments along with a set of masks, identifying which segments belong to the same path.
3) *Segment concatenation*: A novel postprocessing step is designed to filter and concatenate the segments clustered within the same mask into long-horizon paths.
4) *Improved benchmarking*: We release a new public dataset extending that in [16] by more than three times in size. Two novel baselines are implemented ad hoc for comparison with methods that perform naïve autoregressive or one-shot predictions for OCMG.
5) *Real-world validation*: We assess the performance of MaskPlanner by executing the predicted paths on a 6-DoF spray painting robot, achieving expert-level painting quality on previously unseen object instances.
6) *Toward broader applications of OCMG*: We finally demonstrate the successful combination of our method with a downstream trajectory optimizer, and extend our empirical validation to tasks beyond spray painting.

II. Related Work

In this section, we provide a review of the literature related to the OCMG problem (see Sections II-A–II-D) with an in-depth focus on previous works in robotic spray painting (see Section II-F).

A. Learning-Based Motion Generation

Motion generation [17] encompasses a broad scope of problems aiming at generating output paths or trajectories, e.g., for robotic tasks that necessitate adherence to task-specific requirements, constraints, and patterns. A subfield of motion generation is *motion planning* [18], which involves finding low-cost, goal-conditioned trajectories in a given environment while accounting for kinematic or dynamic feasibility, safety, or smoothness. Classical motion planning methods typically work through search, sampling, or optimization over some representation of the state space. In this context, learning-based

approaches have been proposed to speed up planning times for robotic applications [19], e.g., by predicting the sampling distribution [20], [21], warm-starting traditional solvers [22], promoting kinodynamic feasibility [23], or performing end-to-end planning [24]. Yet, motion planning solutions only focus on reaching a specified goal state from a predefined start location, hence, they are not designed to generate complex path patterns that mimic expert behavior.

Therefore, motion generation works generally extend to include human demonstrations or task-specific objectives for learning complex tasks, beyond simple goal-conditioned planning. To this end, Sasagawa et al. [25] trained a recurrent neural network to tackle long-term motion generation in complex tasks such as writing letters, which requires separate sequential strokes. Saito et al. [26] adopted supervised deep learning to tackle long-horizon manipulation tasks and breaking the motion generation problem down into subgoals prediction. Neural networks are also effective in generating human-like whole-body trajectories by learning from human motion capture data, in both humanoid robotics [27] and character control [28]. The field of IL [29], [30], [31], [32], [33] particularly tackles motion generation assuming that a reward function is described implicitly through expert demonstrations, hence solving the task by learning from data. Within the field of IL, behavioral cloning (BC) [29], [30] consists of supervised learning techniques that directly find a mapping from the current state to the optimal action, e.g., through regression methods. Alternatively, inverse reinforcement learning (IRL) [34], [35] aims at learning a representation of the underlying reward function the human experts used to generate their actions. IRL has been successfully deployed to learn parking lot navigation strategies [36], human-like driving behavior [37], and long-term motion forecasting [38].

Notably, BC and IRL methods typically frame motion generation as a sequential decision making problem in an unknown dynamic environment with the Markov property—a Markov decision process. While our work is also fully data-driven, we address the challenge of global, long-horizon motion generation with complete state information (cf. [30, Sec. 3.5]). Notable examples of global trajectory learning works include distilling planning motions [12], [39], imitating patterns on complex 3D shapes and manifolds [40], or motion forecasting for autonomous driving [41]. Yet, although adaptations of trajectory-based IL to multiple agents [12] have been proposed, no method can manage unstructured paths—namely, they fail to model scenarios where the number of paths is unknown, and no temporal correlation exists among separate paths. Finally, our work focuses on learning paths that are generalizable across complex 3D shapes directly, a rarely addressed setting in robot IL [33], [42].

B. Motion Prediction

Motion prediction aims at anticipating the motion of multiple agents ahead in the future. To solve this problem, existing methods generally employ supervised learning techniques from observed trajectories, with applications to autonomous driving [8], [9], [10], [11], human motion forecasting [43], [44], [45], and socially compliant robot navigation [13], [14], [15]. Pfeiffer et al. [15] leveraged the maximum entropy principle to learn a joint probability distribution over the future trajectories of all agents in the scene from data, including the controllable robot. Gupta et al. [44] predict socially plausible human motion paths using a recurrent model and generative adversarial networks. Nayakanti et al. [11] adopted a family of attention-based networks for motion forecasting in autonomous driving, investigating effective ways to fuse scene information including agents' history, road configuration, and traffic light state.

Notably, motion prediction deals with output paths that are jointly executed through decentralized agents that move simultaneously over time, from a given starting state. In turn, numerous strategies to aggregate information across agents and across time can be employed, such as pooling layers [43], [44], independent self-attention for each axis [46], and joint attention mechanisms on multiple axes [8], [9], [11]. In contrast, the OCMG problem considers learning a set of disjoint paths that are uncorrelated in time—e.g., they may be executed separately at different times, in an arbitrary order, and from unknown starting states. In addition, the motion prediction literature assumes a known, fixed number of agents in the scene. Gu et al. [47] have recently introduced the first end-to-end approach to couple motion prediction with object detection and tracking, effectively dealing with a varying number of agents that is automatically inferred at test time. Yet, adapting these works to heterogeneous path lengths and long-horizon motions is an open problem—the literature focuses on fixed prediction horizons of only 3–8 s. Overall, the temporal correlation of predicted paths and assumptions on fixed, short-horizon forecasting render motion prediction methods unsuitable for direct application to the OCMG setting.

C. Set Prediction

Canonical deep learning models are not designed to directly predict sets, i.e., collections of permutation-invariant elements with varying cardinality. Early works addressing this issue in the context of multilabel classification include [48], where an unknown number of labels must be associated with a given input. For regression tasks of set prediction (e.g., object detection), the difficulty is instead to avoid generating near-duplicate outputs (i.e., near-identical bounding boxes) due to an unknown number of output elements. Classical methods to mitigate this issue comprise postprocessing techniques, such as nonmaximal suppression [49], [50]. More recent autoregressive recurrent models sequentially predict output sets [51], [52]. In turn, these are outperformed by transformer-based architectures [53], [54]. Transformers excel in such tasks by leveraging attention mechanisms to decode output elements in parallel and capture long-range dependencies, resulting in more robust and accurate set predictions.

Regardless of the architecture, the loss function designed for set prediction must always be invariant by a permutation of the predictions, or the ground truth. This can be achieved by *matching* predictions with ground truths before the loss computation, either implicitly, leveraging on a moving window across the

input image [50], or explicitly, by solving a bipartite matching problem [49], [52], [54]. The latter approach is widely adopted in object detection [53] and panoptic segmentation (PanSeg) [54] tasks using the Hungarian algorithm [55].

D. 3D Deep Learning from Point Clouds

3D deep learning architectures apply predictive models to process free-form 3D data [56], typically represented as voxel grids, meshes, or point clouds. In particular, point cloud representations describe objects as unstructured sets of 3D points and are successfully employed to perform tasks, such as 3D object classification [57], and segmentation [58], and shape completion [59], [60]. The latter task involves reconstructing missing parts of a 3D object or scene from incomplete input data and is, therefore, relevant for robotics applications [61], [62]. Similarly to the OCMG framework, the input is a free-form 3D shape and the output is unstructured, i.e., the unordered set of points that fill missing input regions. Inspired by these methods, our work leverages the expressive power of 3D deep learning architectures and adapts them to predict unstructured robotic paths that generalize to new objects.

E. Coverage Path Planning (CPP)

CPP focuses on determining a path that passes over all points of an area or volume of interest [67]. This is a core problem in many robotic applications, including lawn mowing [68], visual inspection [69], and spray painting [5]. The goal is typically to ensure full coverage while minimizing path length and/or time. Within the CPP literature, robotic applications for multiagent visual inspection of 3D objects share important similarities to the OCMG setting. Recent works propose optimization-based methods for planning multiple paths and successfully apply them to multi-UAV mission planning covering large 3D structures, such as [65] and HEDAC [66]. While these methods can effectively generate long-horizon paths around both convex and concave surfaces, they rely on sample-specific hyperparameters, incur high computational costs, and are unable to replicate desired expert patterns. In alternative, other recently proposed methods for optimization-based CPP extend *ergodic coverage* from simple domains, such as the Euclidean plane, to 3D objects, surfaces, and Lie groups (e.g., $\mathrm{SE}(3)$). In [70], ergodic coverage is employed to design a closed-loop impedance controller for online tactile exploration tasks including cleaning and inspection from 3D object point clouds. Ergodic trajectory optimization is shown to generalize to more complex domains by employing approximate basis functions [71], or by formulating a novel ergodic metric based on maximum mean discrepancy, requiring only samples from the search domain and reducing the need for domain knowledge [72]. Despite their strong performance on exploration and inspection tasks and being more efficient than FEM-based methods, such as HEDAC [66], ergodic coverage approaches still require to solve a costly optimization problem for each target object. Importantly, they exclusively focus on maximizing a coverage metric instead of learning from demonstrations, which might incorporate expert knowledge in many target applications. Finally, current approaches are limited to generating individual trajectories for each object.

F. Robotic Spray Painting

Autonomous robotic spray painting can be framed as an NP-hard CPP problem, further complicated by nonlinear paint deposition dynamics and hard-to-model engineering constraints. As a result of these challenges, existing approaches are largely heuristic methods that operate under simplifying assumptions about the output path structure (e.g., raster patterns) and object geometry (e.g., convex surfaces) [1], [2], [3], [4], [5], [6], [73]. Most methods further require a 3D mesh or the full CAD model of the object, while point clouds—which are easier to obtain in real-world scenarios through laser scanning—are only considered in [73]. Critically, all existing heuristics yet assume to work with objects that can be partitioned into convex or low-curvature surfaces. This renders them inapplicable for painting concave objects such as shelves and containers, where global reasoning and more complex path patterns are required. Another line of work relies on matching object parts to predefined geometric primitives stored in a database [63]. Each elementary part of the input object is then linked to a corresponding local painting stroke, and these strokes are later merged to form complete paths. While effective in some cases, this approach demands significant manual effort to encode object parts and their associated painting strategies for each object category, and it lacks the ability to generalize to arbitrary free-form shapes. Gleeson et al. [6] proposed a trajectory optimization procedure for spray painting that adapts an initial candidate trajectory externally provided.

RL is an alternative approach to train path generators by directly optimizing task-specific objectives, such as PC [7] and total variation [64]. RL-based stroke sequencing is also effective for 2-D image reconstruction tasks [74]. Despite their potential, these methods are limited to planar domains, and their applicability to long-horizon path generation on 3D objects remains unproven, mainly due to the complexity of high-dimensional state and action spaces. Furthermore, RL often suffers from poor generalization and requires accurate simulation environments.

We further highlight that all the aforementioned works are evaluated on a limited set of proprietary object instances, with neither datasets nor implementation details publicly available. This hinders reproducibility and fair benchmarking, and raises concerns on the generalization capabilities of these methods to unseen object instances and categories.

Table I provides a summary of the most relevant publications showing how existing settings and tasks in the literature relate to the OCMG problem.

III. Method

A. Problem Statement

We formalize the OCMG problem as finding a mapping from a 3D object point cloud to a set of unstructured output paths (see Fig. 2). Given expert demonstrations, we aim to generate accurate paths for previously unseen objects.

TABLE I
LITERATURE REVIEW WITH A SELECTED NUMBER OF EXEMPLARY WORKS IN FIELDS OF APPLICATIONS RELATED TO OCMG

Tasks	Works	Input	Output				Method	Pros (+) and Cons (–)		
			Multiple paths	Variable num. of paths	Variable path lengths	Long horizon paths		Fast Inference (+)	Ability to Generalize (+)	For Painting Applications
Spray Painting	[4], [5]	3D (convex only)	✓	✗	✗	✓	Task-specific Heuristics	✗	✗	(+) High paint coverage (–) High design costs and manual tuning
	[1], [2], [3], [63]		✓	✓	✓	✓				
	[7], [64]	2D	✗	✗	✗	✓	Reinforcement Learning	✗	✗	(+) Explicit paint coverage optimization (–) Requires accurate simulation
Multi-Agent Visual Inspection	[65], [66]	3D	✓	✗	✓	✓	Coverage Path Planning	✗	✗	(+) High inspection coverage (–) Sample-specific hyperparameters (–) Unable to model painting patterns
Multi-Agent Motion Prediction	[8], [9], [10], [11]	2D map + agent features	✓	✗	✗	✗	Imitation and Supervised Learning	✓	✓	(+) Learns painting patterns from data (+) Little domain knowledge required (–) Implicit paint coverage optimization
	[47]	3D	✓	✓	✗	✗				
3D Shape Completion	[59], [60]	3D	∼ Point-wise predictions							
Object Centric Motion Generation	**Ours**	3D	✓	✓	✓	✓				

We dissect each work to shed light on the differences and similarities with our problem setting.
The bold highlights the proposed framework to distinguish from tasks / methods / frameworks existing in literature.

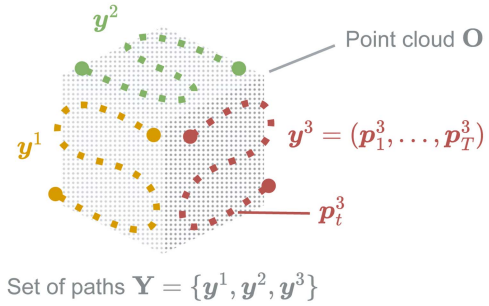

Fig. 2. Schematic illustration of a data sample $(\mathbf{O}, \mathbf{Y})$ describing input and output of the OCMG problem. The output paths are unstructured as they vary in number and length depending on the input object and can be executed in arbitrary order.

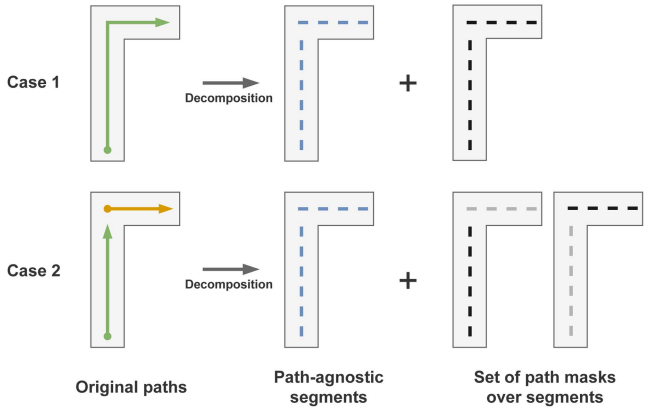

Fig. 3. Example of two arbitrary configurations of ground truth paths $\mathbf{Y}$ on an L-shaped 2-D object. Notice how MaskPlanner can naturally manage both cases by breaking down the learning problem into the prediction of path-agnostic segments and their associated path mask.

Let $\mathbf{O}$ represent the object geometry as a point cloud consisting of an arbitrary number of points in 3D space. Each object $\mathbf{O}$ is associated with a ground truth set of paths $\mathbf{Y}=\{\boldsymbol{y}^i\}_{i=1}^{n(\mathbf{O})}$, with object-dependent cardinality $n(\mathbf{O})$. Every path $\boldsymbol{y}^i = (\boldsymbol{p}_1^i, \ldots, \boldsymbol{p}_T^i) \in \mathcal{Y} \subseteq \mathbb{R}^{6 \cdot T}$ is encoded as a sequence of 6-D poses $\boldsymbol{p} \in \mathbb{R}^6$. For simplicity, we consider the dimension T fixed, with shorter paths zero-padded to reach such length.

Under this formulation, we consider the problem of finding a function $f : 2^{\mathbb{R}^3} \to 2^{\mathcal{Y}}$ mapping the set of points $\mathbf{O}$ describing the object geometry to the set of desired paths $\mathbf{Y}$. To do so, we parameterize f using a deep neural network, and train it through empirical risk minimization [75]. Specifically, we minimize a loss function $\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$, which quantifies the discrepancy between the predicted paths $\hat{\mathbf{Y}} = f(\mathbf{O})$ and the ground truth paths $\mathbf{Y}$ on the training data, using gradient descent to optimize the network parameters.

We highlight that this formulation does not make task-specific assumptions related to the spray painting problem, making our contribution applicable to a broad range of OCMG tasks (e.g., welding or cleaning).

B. Method Overview

We tackle OCMG with a tailored deep learning model that copes with unstructured input—3D point clouds—and unstructured output paths. More precisely, instead of directly predicting a set of paths, we decompose the problem into the prediction of unordered path segments, i.e., short sequences of 6-D end-effector poses (or *waypoints*). In addition, we concurrently predict a set of probability masks that identify which segments belong to the same path. We denote our method as *MaskPlanner*, and emphasize that all required path segments and masks are predicted by our network in parallel, with a single forward pass. Such approach induces our model to learn end-to-end global embeddings of the input object that allow for both local (segments) and global (masks) planning decisions in one step. Overall, by designing a joint segment and mask prediction pipeline, we conveniently deal with unstructured paths and address the following challenges.

1) *Long-horizon paths:* We do not impose constraints on the shape or length of each path. Diverse path configurations can be handled just as effectively, as exemplified in Fig. 3.
2) *Unordered paths:* By network design, predicted segments are invariant by permutation. Therefore, concatenating segments naturally yields an unordered set of output paths.
3) *Variable length and number of paths:* We predict a conservatively large number of path segments, allowing for the generation of redundant overlapping segments that can be easily filtered out.

Our method takes inspiration from the PanSeg literature [53], [54], where a variable number of class instances shall be predicted given a static environment with global information (an RGB image). Notably, recent works in the PanSeg field mostly focus on one-shot predictors as opposed to multiple-stage or autoregressive approaches. Similarly, we depart from sequential methods for OCMG, as we aim to reach real-time inference capabilities and avoid compounding errors on long-horizon predictions.

Throughout this work, we demonstrate that MaskPlanner is capable of predicting a large number of long-horizon paths with a single forward pass, that implicitly incorporate task-specific requirements without domain knowledge. We remark that ad hoc trajectory optimization methods [6] may still be applied to impose task-specific kinodynamic constraints (e.g., reachability and collision avoidance), which we consider subsequent and complementary to the scope of this work. Here, we directly interpolate the predicted sequence of 6-D waypoints for execution on a real robot, resulting in feasible trajectories out of the box.

In the following part of this section, we describe each step of our method in detail, breaking it down into segment predictions (see Section III-C), mask predictions (see Section III-D), and postprocessing (see Section III-E).

C. Segment Predictions

Let $\mathbf{S}$ be the set of ground truth path segments $\mathbf{S}=\{\boldsymbol{s}^j\}_{j=1}^{k(\mathbf{Y})}$ of an object $\mathbf{O}$ and its associated paths $\mathbf{Y}$. We define a segment as a sequence of $\lambda \in \mathbb{N}^+$ poses obtained from a path $\boldsymbol{y}^i \in \mathbf{Y}$. Namely, $\boldsymbol{s}^j = (\boldsymbol{p}_t^i, \dots, \boldsymbol{p}_{t+\lambda-1}^i) \in \mathbb{R}^{6\cdot\lambda}$ for some timestep $t=1,\dots,T-\lambda$. We derive all segments in $\mathbf{S}$ by striding along all paths with a step of $\lambda-1$, i.e., considering an overlap of one pose between consecutive segments. Notice that the total number of resulting segments $k(\mathbf{Y})$ depends on the number of paths and their lengths.

We design our model to take the object point cloud $\mathbf{O}$ as input, and predict a set of path segments $\widehat{\mathbf{S}}=\{\hat{\boldsymbol{s}}^j\}_{j=1}^{K}$ that approximate the true segments $\mathbf{S}$. We adopt the PointNet++ architecture [58] as basic backbone for global feature extraction from $\mathbf{O}$, followed by a fully connected 3-layer decoder that jointly outputs all path segments. A fixed number of segments K is predicted, corresponding to the maximum number of ground-truth segments per object across the training set.

Let $\mathbf{P}$ be the set of unordered ground truth waypoints, formally described as $\{\boldsymbol{p}_t^i \in \mathbb{R}^6 \mid i \in [1,\dots,n(\mathbf{O})],\ t \in [1,\dots,T]\}$ —equivalent to the set of segments for $\lambda=1$. Analogously, we define $\widehat{\mathbf{P}}=\{\hat{\boldsymbol{s}}_t^j \in \mathbb{R}^6 \mid j \in [1,\dots,K],\ t \in [1,\dots,\lambda]\}$ as the set of individual predicted waypoints, obtained by interpreting $\widehat{\mathbf{S}}$ as an unordered collection of waypoints.

MaskPlanner is trained with a novel loss function aimed at driving the prediction of path segments $\widehat{\mathbf{S}}$ close to the ground truth segments $\mathbf{S}$ by means of Euclidean distances in $\mathbb{R}^{6\cdot\lambda}$ space. To do this, our loss includes auxiliary point-wise terms that penalize prediction errors in the lower dimensional space $\mathbb{R}^6$, by disregarding how waypoints are arranged into segments. Overall, our *point-to-segment chamfer distance (CD)* is defined as

$$\mathcal{L}_{\text{p2s}}(\widehat{\mathbf{P}},\mathbf{P},\widehat{\mathbf{S}},\mathbf{S}) = w_p^f \cdot d_{\text{ACD}}(\widehat{\mathbf{P}},\mathbf{P}) + w_s^f \cdot d_{\text{ACD}}(\widehat{\mathbf{S}},\mathbf{S}) + \underbrace{w_p^b \cdot d_{\text{ACD}}(\mathbf{P},\widehat{\mathbf{P}})}_{\text{Point-wise}} + \underbrace{w_s^b \cdot d_{\text{ACD}}(\mathbf{S},\widehat{\mathbf{S}})}_{\text{Segment- wise}}. \quad (1)$$

Here, the *asymmetric chamfer distance* (ACD) *from* set A *to* set B is

$$d_{\text{ACD}}(A,B) = \frac{1}{|A|}\sum_{a\in A}\min_{b\in B}\|a-b\|_2^2 \quad (2)$$

and the parameters $w_p^f, w_s^f, w_p^b, w_s^b \in \mathbb{R}^+$ weight the computation of the four ACD terms (f: forward, b: backward, p: point-wise, s: segment-wise).

Based on this formulation, we propose an auxiliary point-to-segment curriculum that 1) first focuses on matching waypoints in some random permutation (point-wise terms), and then 2) gradually promotes local structure by computing distances among sequences of poses (segment-wise terms). To achieve this, we start the training with a dominant point-wise term ($w_p^b \gg w_s^b$) and progressively converge to equal point-wise and segment-wise contributions ($w_p^b=1, w_s^b=1$). Notably, this procedure is asymmetric and only applied on the backward terms: this ensures that predictions are pulled sufficiently close to the ground truth poses, which in turn enables effective global coverage of the desired poses, and limits the generation of clusters of poses [76]. On the other hand, we keep the forward weights fixed to $w_p^f=0$, $w_s^f=1$ throughout the training, promoting the generation of segments that are smooth and locally accurate. We refer to this process as the *asymmetric point-to-segment* (AP2S) curriculum, and report a schematic illustration of our overall loss function in Fig. 5.

D. Mask Predictions

Our model concurrently predicts a set of probability masks over predicted segments, indicating which segments belong to the same path. Defining a supervised learning objective for this task is not trivial, as it requires comparing predicted masks on generated segments with ground truth masks on generated segments. However, the latter do not exist. A reasonable choice is to match each predicted segment with the closest ground truth segment—as already proposed in Section III-C—and construct target masks accordingly. We refer to this strategy as *nearest-neighbour label association*, and use it to project masks over ground truth segments onto predicted segments. In particular, let $\boldsymbol{M}=\{\boldsymbol{m}^i\}_{i=1}^{n(\mathbf{O})}$ be the set of *target path masks* for some input pair $(\mathbf{O},\mathbf{Y})$. Each element $\boldsymbol{m}^i \in \{0,1\}^K$ encodes the set of predicted segments that belong to path i, that is

$$\boldsymbol{m}_j^i = \begin{cases} 1 & \text{if NN}(\hat{\boldsymbol{s}}^j) \text{ belongs to path } i \\ 0 & \text{otherwise} \end{cases}$$
$$\forall\, j = 1,\dots,K \quad (3)$$

where $\text{NN}(\hat{\boldsymbol{s}})=\arg\min_{\boldsymbol{s}\in\mathbf{S}}\|\boldsymbol{s}-\hat{\boldsymbol{s}}\|^2$. Our model is designed to predict a set of $N \geq n(\mathbf{O})$ probability masks $\widehat{\boldsymbol{M}}=\{\hat{\boldsymbol{m}}^i\}_{i=1}^{N}$, where N is the maximum number of paths associated to an object across the training set. All masks are predicted in parallel via a three-layer MLP decoder from the global features of the

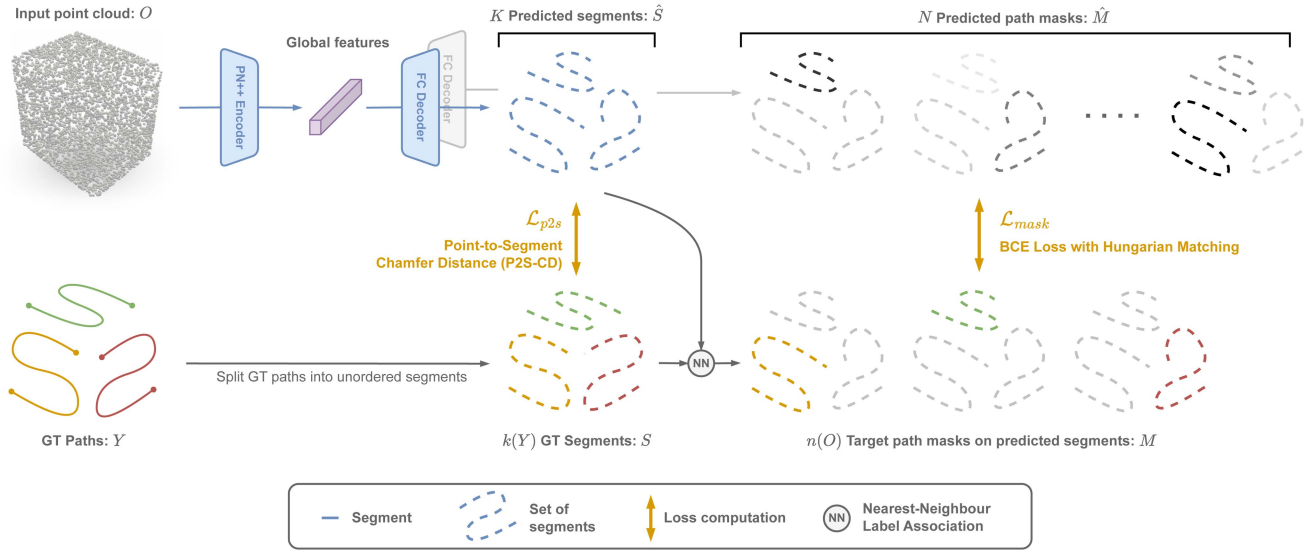

Fig. 4. Overview of the training pipeline of our method (MaskPlanner). Global features are learned from a point cloud representation of the input object, and used to concurrently predict path segments and path masks, in a single forward pass.

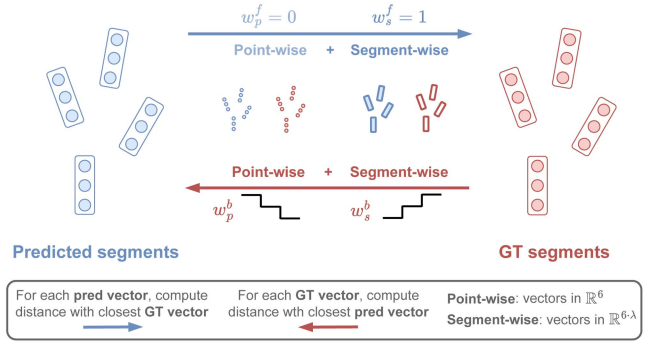

Fig. 5. Illustration of our AP2S curriculum for segment predictions (λ=3). The parameters w_p^b, w_s^b weighting the backward point-wise and segment-wise ACD terms vary during training.

object, followed by a sigmoid activation. Note that, as in [54], we do not force mask predictions to be mutually exclusive for a given segment $\hat{\boldsymbol{s}}^j$. Hence, we avoid using a softmax activation.

We drive the predicted path masks $\widehat{\boldsymbol{M}}$ toward the target path masks $\boldsymbol{M}$ through the binary cross entropy loss

$$\mathcal{L}_{\text{bce}}(\hat{\boldsymbol{m}}, \boldsymbol{m}) = -\sum_{j=1}^{K}\Big[\boldsymbol{m}_j \cdot \log(\hat{\boldsymbol{m}}_j) + (1-\boldsymbol{m}_j)\cdot\log(1-\hat{\boldsymbol{m}}_j)\Big]. \tag{4}$$

Importantly, we must consider the prediction of permutation invariant masks to cope with a set of unordered paths. Therefore, we assign each predicted path mask $\hat{\boldsymbol{m}}$ to a target mask $\boldsymbol{m}$ by finding a bijection $\sigma : \widehat{\boldsymbol{M}} \to \boldsymbol{M}$. Similarly to [53], [54], we do this by solving a bipartite matching problem between the two sets, where the assignment costs are computed using $\mathcal{L}_{\text{bce}}$. Particularly, we pad the target masks with a "no path" token $\varnothing$ to allow one-to-one matching. Ultimately, the training loss for mask prediction is as follows:

$$\mathcal{L}_{\text{mask}}(\widehat{\boldsymbol{M}}, \boldsymbol{M}) = \sum_{i=1}^{N}\Big[c^{\sigma(i)}\cdot\log(\hat{c}^i) + (1-c^{\sigma(i)})\cdot\log(1-\hat{c}^i) + c^{\sigma(i)}\cdot\mathcal{L}_{\text{bce}}(\hat{\boldsymbol{m}}^i, \boldsymbol{m}^{\sigma(i)})\Big] \tag{5}$$

where $\hat{c}^i \in [0,1]$ is a learned confidence score for each mask, and $c^i = \mathbb{1}_{\boldsymbol{m}^i \neq \varnothing}$ indicates the true masks. Overall, we train our model to minimize $\mathcal{L} = \mathcal{L}_{\text{p2s}} + \mathcal{L}_{\text{mask}}$. We display a schematic description of our training pipeline in Fig. 4.

At inference time, we derive the groups of segments belonging to the same path by assigning each predicted segment $\hat{\boldsymbol{s}}^j$ to one of the N predicted masks. Formally, the assignment of segment $\hat{\boldsymbol{s}}^j$ occurs as follows:

$$\underset{i\in[1,\ldots,N]}{\arg\max}\ \hat{\boldsymbol{m}}_j^i \ \text{ s.t. } \ \hat{c}^i \geq 0.5. \tag{6}$$

In other words, segments are assigned to the path mask with the highest predicted probability, discarding path masks that are predicted as "no path." Assigning all predicted segments according to (6) results in a final number of $\hat{n}(\mathbf{O}) \leq N$ paths predicted by our model given the input object, ideally equal to $n(\mathbf{O})$.

E. Postprocessing: Segment Concatenation

A final postprocessing step is applied to concatenate the subset of predicted segments $\widehat{\mathbf{S}}^i \subseteq \widehat{\mathbf{S}}$ that are assigned to the same path mask $\hat{\boldsymbol{m}}^i$, and produce an ordered sequence of 6-D waypoints, i.e., the executable path.

Viable solutions may include solving an open traveling salesman problem among segments or employing learning-based approaches for ranking. Here, we adopt a simple and effective

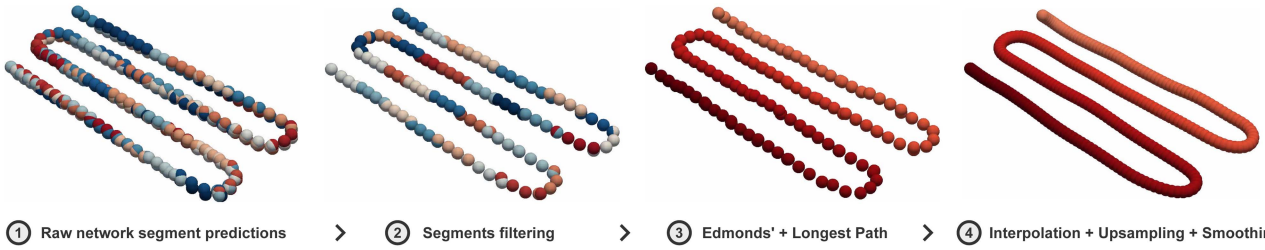

Fig. 6. Postprocessing: Concatenation of the set of predicted segments belonging to the same path mask. In step (1), the figure depicts raw network predictions with λ=4, with separate segments differentiated by color. Step (2) shows the effect of segment filtering. In step (3), the ordered sequence of waypoints is obtained by finding the longest path within a spanning arborescence of the predicted set of segments, displayed with a color gradient. Step (4) ultimately refines the obtained path.

concatenation strategy based on segment proximity and alignment.

First, predicted segments in excess are removed by discarding segment pairs whose distance falls below a predefined threshold, proceeding in ascending order of pairwise distances.[1] Then, we find an optimal path that connects the retained segments. Consider the set of segments in $\widehat{\mathbf{S}}^i$ as nodes of a directed graph. An edge among two segments is weighted based on the proximity in space and orientation between the starting and ending poses of the two segments, as well as the similarity in segment directions. More formally, the assigned cost to the edge from $\hat{\boldsymbol{s}}^j$ to $\hat{\boldsymbol{s}}^k$ is

$$C(\hat{\boldsymbol{s}}^j, \hat{\boldsymbol{s}}^k) = \|\hat{\boldsymbol{s}}^j_\lambda - \hat{\boldsymbol{s}}^k_1\|_2^2 + w_v \cdot \|(\hat{\boldsymbol{s}}^j_\lambda - \hat{\boldsymbol{s}}^j_{\lambda-1}) - (\hat{\boldsymbol{s}}^k_2 - \hat{\boldsymbol{s}}^k_1)\|_2^2. \tag{7}$$

Here, $w_v \in \mathbb{R}^+$ is a tradeoff weight between the two terms, and the segments' subscripts specify the index of a particular pose among the λ poses that make up each segment.

We find the optimal concatenation by employing the Edmonds' algorithm [77] to the k-nearest neighbor graph of segments $\widehat{\mathbf{S}}^i$ constructed using $C(\cdot,\cdot)$ and k=5. As a result, we obtain a directed acyclic graph (DAG) representing the spanning arborescence among predicted segments whose edge weights are minimized. Ultimately, we extract the longest path from the resulting DAG and obtain the final *ordered* sequence of predicted segments. At this point, filtering techniques, such as interpolation, upsampling, and smoothing, may be conveniently applied. The same process is repeated independently for all predicted path masks. For clarity, we illustrate each step of the postprocessing in Fig. 6.

IV. Extended PaintNet Dataset

PaintNet [16] has been the first dataset of expert demonstrations introduced to support the study of motion generation conditioned on free-form 3-D objects, specifically designed for the spray painting task. In this work, we expand PaintNet more than threefold, resulting in a new version of 3088 samples.

Every sample is a pair of a 3-D object and its corresponding spray painting paths. Each object is represented as a triangle mesh, with vertex coordinates expressed in real-world millimeter scale. The meshes are provided in an aligned and smoothed watertight [78] format with any private information (e.g., engraved logos) accurately anonymized. The number of spray painting paths associated to an object varies according to the geometry of the object. Each path is encoded as a sequence of end-effector configurations in task space, i.e., positions and orientations of the gun nozzle. More precisely, we record 3-D positions as the ideal paint deposition point, 12 cm away from the gun nozzle, and 3-D gun orientations as Euler angles. The data pairs represent realistic spray painting demonstrations that a real robot could directly execute. In other words, they encode feasible trajectories designed by experts to reach near-complete coverage of the whole surface of the 3-D objects. Each waypoint is collected by sampling the end-effector pose at a rate of 250 Hz during execution. Ground-truth paths associated to each object are produced by a set of proprietary heuristics based on long-standing experience in the field. Particularly, the heuristics are designed ad-hoc for synthetically generated objects shaped as cuboids, windows, or shelves. In addition, we collect a curated set of 88 real-world industrial containers, which we add to the dataset along with their associated painting paths via a manually guided process. Each category is presented in details as follows.

1) *Cuboids*: A basic class of 1000 rectangular cuboids tailored for testing models under minimal generalization requirements and relatively simple path patterns. Cuboids are generated with varying height and depth uniformly from 0.5 to 1.5 m, while having a fixed width of 1 m. Their volume ranges from 0.25 to 2.25 m^3. A fixed number of six raster-like paths is associated to each cuboid to paint the exterior faces, with gun orientations that are perpendicular to the surface at all times.
2) *Windows*: A set of 1000 window-like data pairs, generated with width and height varying uniformly from 0.4 to 1.8 m, and a fixed thickness of 4 cm. For each sample, up to 3 horizontal cross sections and up to 1 vertical cross section are randomly selected. Windows introduce harder challenges for motion generation, such as predicting a variable, high number of paths while handling nontrivial gun orientations and geometric patterns.
3) *Shelves*: A set of 1000 shelves featuring highly concave surfaces. The instances are generated to differ significantly in volume and number of inner compartments. Their

[1]Note that our loss $\mathcal{L}_{\text{p2s}}$ implicitly promotes overlapping if the number of predicted segments K is higher than the ground truth segments $k(\mathbf{Y})$.

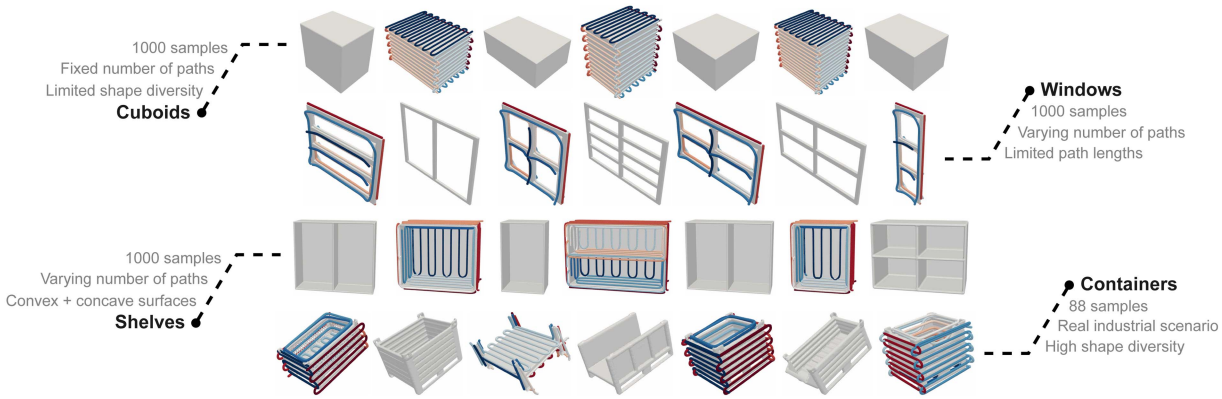

Fig. 7. Representative instances from the extended PaintNet dataset, featuring realistic spray painting demonstrations designed by experts. The dataset is divided into four categories of growing complexity. Different colors represent separate paths.

volume ranges from 18 to 160 dm^3, with up to 6 inner compartments for the larger samples.

4) *Containers*: A set of 88 industrial containers from a real industrial scenario including meshes with highly heterogeneous global and local geometric properties (e.g., wavy and grated surfaces). Here, the manually guided painting paths designed by experts show evident irregularities across instances. This class of objects is particularly challenging due to the limited number of samples.

See an overview of the extended PaintNet dataset in Fig. 7.

V. Experimental Evaluation

A. Implementation Details

1) Data Preparation*:* All experiments are carried out on the Extended PaintNet dataset introduced in Section IV. Input point clouds $\mathbf{O}$ are derived by sampling 5120 points from the surface of the available meshes through poisson disk sampling [79]. We further down-sample ground-truth paths so that adjacent poses are approximately 5 cm apart, avoiding to deal with needlessly dense waypoints. Each waypoint is represented as a 6-D vector with position described by (x, y, z) coordinates, and orientation encoded as a 3-D unit vector, rather than Euler angles. This simplification is permitted by our conic spray gun model, which is symmetric and invariant to rotations around the approach axis. Consequently, ground-truth Euler angles are converted into 3-D unit vectors (2-DoF), indicating where the gun nozzle is pointing. When computing Euclidean distances in (1) we rescale and adjust the relative importance of position and orientation components in the 6-D vector by weighting the latter by 0.25. Point clouds and path poses in the dataset are finally transformed so that each sample is centered around the origin, and down-scaled by a dataset-global factor. We create training and test sets by dividing each category in the dataset according to an 80%/20% split.

2) Architecture: We employ an encoder module based on PointNet++ [58] as the feature extractor mapping the input point cloud to a latent space of dimensionality 1024. The encoder is initialized with pretrained weights from an auxiliary shape classification task on ModelNet [80]. Segment and mask predictions are generated in parallel from the object features through separate dedicated three-layer MLPs decoders with hidden sizes (1024,1024) and output sizes $6 \times \lambda \times K$ and $K \times N$—respectively, for segments and masks. Confidence scores $\hat{c}^i$ for each path mask are learned through an additional linear output layer of size N, followed by a sigmoid activation. We set $\lambda{=}4$ throughout all our experiments as in [16], and report a sensitivity analysis on this hyperparameter in Appendix B.

3) Training Details: We minimize our loss function $\mathcal{L}$ using the Adam optimizer with learning rate 10^{-3} over 4800 epochs. Notably, we find it beneficial to activate the path mask loss $\mathcal{L}_{\text{mask}}$ only after 3200 epochs, so that masks are learned only when the predicted segments are sufficiently close to the ground truth. We implement the AP2S curriculum described in Section III-C by scaling w_p^b and w_s^b by factors of 0.1 and 10, respectively, at epochs 1000 and 2000. The learning rate is halved five times at intervals of 800 epochs throughout the training process. Training takes approximately 6 hours on a single NVIDIA RTX-4080 GPU for a training set of 800 objects and a batch size of 64. Each forward pass on 5120 input points takes approximately 100 ms on the same GPU, while postprocessing requires 100–500 ms on a single CPU core, depending on the number and length of output paths.

B. Baselines

The OCMG problem as described in Section I and formalized in Section III, is introduced in this work for the first time. The overview of previous literature in Section II highlights that none of the existing methods is suitable for handling OCMG tasks through deep learning. To provide a thorough benchmark comparison of MaskPlanner against alternative approaches, we devise a new set of baselines as ad hoc adaptations of existing works designed for different problem settings. A schematic overview of all baselines is depicted in Fig. 8.

1) Path-Wise: This model outputs a set of complete paths at once, where each path is a $6{\cdot}T$-dimensional vector. It predicts a predefined maximum number of paths, and identifies which paths to retain at inference time through learned confidence scores. The model also predicts an "end-of-path" probability

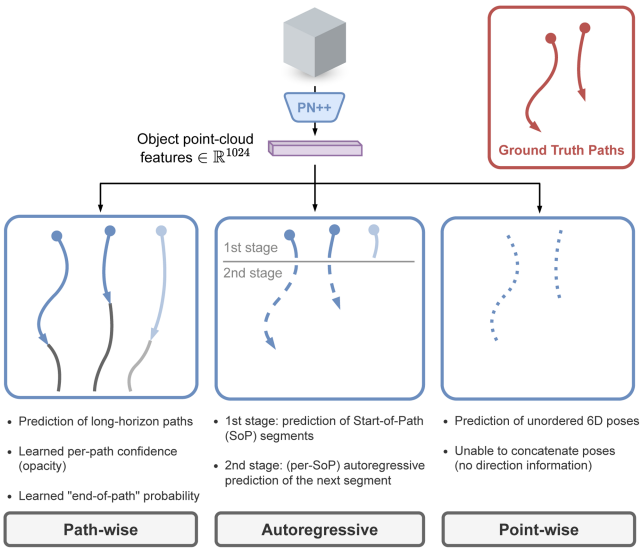

Fig. 8. Overview of the novel baselines implemented for comparison with MaskPlanner in the OCMG problem. All baselines share the same PointNet++ (PN++) encoder architecture for extracting object features.

for each individual path waypoint, thus, the length of each path can vary. This approach is inspired by the object detection literature, with paths treated analogously to bounding boxes. In particular, we adopt the same logic of one-shot set prediction as in DETR [53], but we design the network architecture with MLP layers as in MaskPlanner, instead of DETR's query-based modules. Overall, this implementation extends the *multipath regression* baseline introduced in [16], and reflects the attempt to directly predict long-horizon paths as opposed to breaking down the problem into segment predictions.

2) Autoregressive: This baseline allows comparing one-shot prediction methods—such as MaskPlanner and Path-wise—to an alternative autoregressive strategy for OCMG. We design a method for predicting the next pose and a termination probability given a history of previously generated poses, separately for each path. It consists of a first model trained to predict a set of start of path (SoP) poses given the object features (first stage). This is followed by the autoregressive model conditioned on the predicted SoP, the object features, and the 10 most recent predictions (second stage). The training procedure for the first stage is analogous to the path-wise baseline, i.e., the SoP model learns confidence probabilities for filtering out SoPs in excess. We then use teacher forcing [81] to train the autoregressive model and inject noise into the input history to mitigate compounding errors. Interestingly, we observe that predicting a sequence of λ output poses—i.e., segments—at each autoregressive step yields smoother output paths. Thus, we adopt this approach for this baseline. Finally, we find that jointly training the SoP and autoregressive models at the same time leads to training instabilities and does not improve predictive performance.

3) Point-Wise: Inspired by shape completion methods [59], the approach proposed in [16] directly predicts all output waypoints as a set of unordered 6-D poses, employing the standard symmetric CD [82]. We integrate this approach with the mask predictions pipeline of MaskPlanner in order to recognize how output waypoints are organized into separate paths. Notably, this implementation can be seen as an edge case of MaskPlanner with fully point-wise loss terms $w_p^f{=}1, w_s^f{=}0, w_p^b{=}1, w_s^b{=}0$. However, unlike MaskPlanner, this baseline is unable to concatenate output poses into long-horizon paths through postprocessing, rendering it practically inapplicable to a real-world scenario.

C. Evaluation Metrics

We introduce a collection of metrics for OCMG to assess the performance of different methods in predicting unstructured paths. In the following, we use $h = 1, \ldots, H$ as an index on the test object samples $\mathbf{O}_h$.

Point-wise chamfer distance (PCD) [82]: It compares paths as clouds of 6-D poses by computing $\frac{1}{H}\sum_{h=1}^{H} \text{PCD}(\widehat{\mathbf{P}}_h, \mathbf{P}_h)$, where $\text{PCD}(\widehat{\mathbf{P}}, \mathbf{P}){=}d_{\text{ACD}}(\widehat{\mathbf{P}}, \mathbf{P}) + d_{\text{ACD}}(\mathbf{P}, \widehat{\mathbf{P}})$. It accounts for the accuracy of all end-effector positions and orientations, while disregarding the order among poses. Lower is better.

Accuracy of number of paths (Acc-NoP): It measures the fraction of objects across the test set for which the predicted number of output paths matches the true number, i.e., $\frac{1}{H}\sum_{h=1}^{H} \mathbb{1}_{\hat{n}(\mathbf{O}_h)=n(\mathbf{O}_h)}$. Higher is better.

Mean absolute error of number of paths (MAE-NoP): This metric measures the average deviation of the predicted number of paths from the true number. It is obtained by computing the mean absolute error $\frac{1}{H}\sum_{h=1}^{H} |\hat{n}(\mathbf{O}_h) - n(\mathbf{O}_h)|$ across objects in the test set.[2] Lower is better.

PC: Although not explicitly optimized at training time, we aim to assess the percentage of object surface covered by the generated paths when executed in a spray painting simulator. To do so, we follow the same approach as in [16]. First, the 10th percentile of the ground truth paint thickness distribution across mesh faces is selected as the relative paint thickness threshold for measuring coverage. Next, we execute the generated paths and evaluate the percentage of faces whose paint thickness is larger than such threshold. We then average the percentages across all instances of the test set. Notice that the relative thickness threshold makes the metric independent of the specific spray gun model parameters used during simulation (e.g., paint flux), thus, renders it suitable for benchmarking purposes.

D. Results: Network Predictions

We evaluate the performance of models trained individually on each category, while keeping the same hyperparameters. This allows us to study the robustness of our design choices to variations in data distributions. The categories are considered in order of growing complexity of object shapes and path patterns: cuboids, windows, shelves, and containers.

Main qualitative results: We report the qualitative results of raw network predictions on a subset of test instances in Fig. 9. The results from the *path-wise* baseline indicate that one-shot methods inspired by object detection [50], [53], [54] provide a strong basis for addressing unstructured output paths. Although

[2]Both MAE-NoP and Acc-NoP are based on established metrics for ordinal regression problems in machine learning (e.g., see [83, Sec. 4.1.3]).

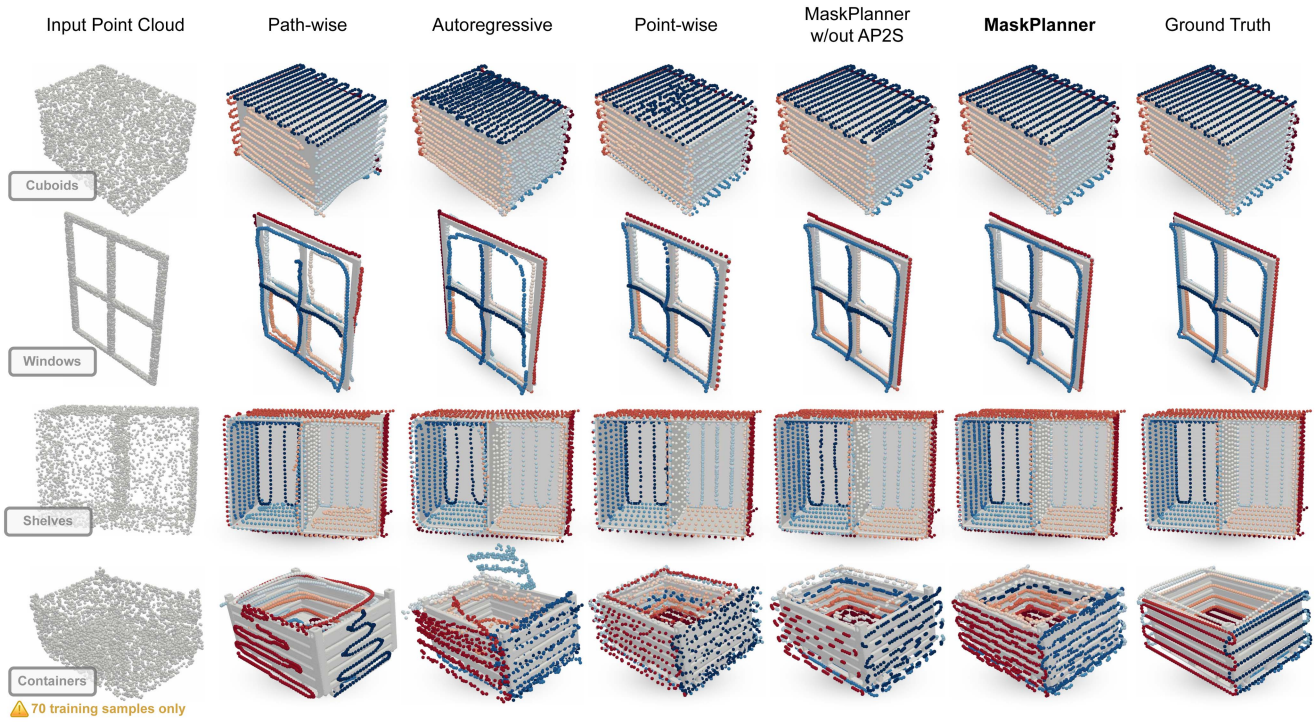

Fig. 9. Main qualitative results: the raw network predictions of all baselines are shown for a representative test sample of each object Category. Points displayed with the same color belong to the same path. Point orientations are not visible.

promising, the major limitation of this approach stems from the explicit computation of the loss between high-dimensional paths: the curse of dimensionality yields inaccurate predictions over long-time horizons (see the degenerate raster patterns for cuboids and containers).

The *autoregressive* approach aims at counteracting the described phenomenon, by reducing the dimensionality of the network output at the cost of additional forward passes. However, this baseline exhibits significant compounding errors, often predicting the wrong number of straight passes when attempting to match the long raster paths of the cuboids category. These findings highlight the importance of coping with the long-horizon nature of output paths alongside the other challenges of OCMG tasks.

Both *point-wise* and MaskPlanner approach the task by focusing on local path patterns and generating sets of poses or path segments, respectively. This novel paradigm shows significantly stronger generalization capabilities across all object categories. Still, the point-wise baseline provides predictions that fail to capture detailed path structures—e.g., see the top face of the cuboid, and the inner paths of the shelf. MaskPlanner successfully tackles this problem by introducing 1) segment-wise CD terms, and 2) the AP2S curriculum. Notice that both enable MaskPlanner's state-of-the-art performance: the paths obtained by MaskPlanner without AP2S exhibit errors similar to those of the point-wise baseline. In other words, incorporating both point-wise and segment-wise terms in $\mathcal{L}_{\text{p2s}}$ yields better performance than minimizing either one alone (see Appendix A for a thorough ablation of our loss function).

Finally, in Fig. 10, we present an illustration of MaskPlanner's network predictions as the training progresses. The picture highlights the effect of the AP2S curriculum: the model initially learns to generate sparse individual poses close to the target object surface; then, local structure is gradually promoted and smooth segment predictions are obtained.

Main quantitative results: The quantitative results of our experimental analysis are presented in Table II, considering all the metrics defined in Section V-C. It can be observed that the novel mask predictions paradigm is the most effective across all object categories. The path-wise and autoregressive baselines attempt to learn a confidence score for each path, but underperform in terms of NoP metrics. Instead, point-wise and MaskPlanner predict nonmutually exclusive masks over their own predictions to distinguish among different paths, a strategy that leads to significantly better NoP metrics at inference time. Notably, these findings align with those on the predicted number of instances from the PanSeg literature, where MaskFormer [54] improves over DETR [53].

Finally, we analyze the results on the containers category in greater detail. Here, the models are trained on only 70 data samples of heterogeneous object shapes, and tested on 18 unseen instances. This is a challenging setting often encountered in real-world industrial scenarios where the training set size is limited, while strong generalization capabilities are required. All methods exhibit low performance across the evaluated metrics, with high variance across independent training repetitions. Although point-wise achieves PCD scores comparable to our method, we observe that the former leads to globally sparse predictions that lack local consistency (cf. Fig. 9). We attribute this behavior to the properties of the CD, which is well-known for its insensitivity to mismatched local density [76]. Yet, MaskPlanner outperforms the other baselines in terms of PCD also

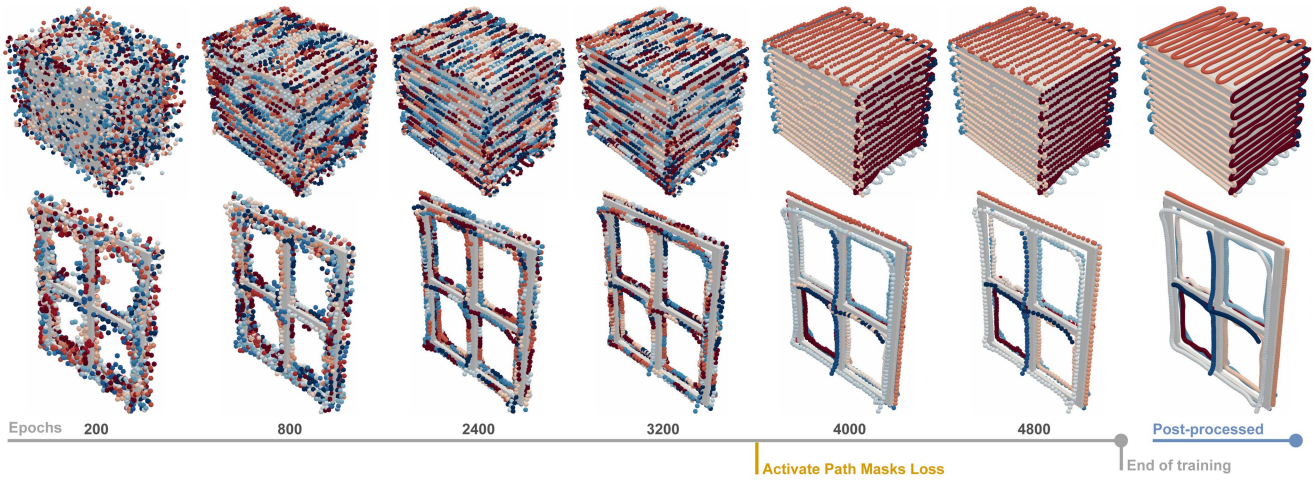

Fig. 10. Network is trained in a coarse-to-fine manner via the AP2S curriculum: first, output waypoints are positioned across the surface; then, local structure is promoted. Segments associated to the same path mask are shown with the same color.

TABLE II
MAIN QUANTITATIVE RESULTS ON THE TEST SET OF EACH CATEGORY

	Cuboids	Windows			Shelves			Containers		
	PCD (↓)	PCD (↓)	Acc-NoP (↑)	MAE-NoP (↓)	PCD (↓)	Acc-NoP (↑)	MAE-NoP (↓)	PCD (↓)	Acc-NoP (↑)	MAE-NoP (↓)
Path-wise	48.03 ± 5.34	64.10 ± 7.79	73.40 ± 8.09	0.40 ± 0.09	40.45 ± 5.34	70.70 ± 10.30	0.35 ± 0.15	556.39 ± 10.60	**20.00 ± 3.04**	2.70 ± 0.23
Autoregressive	33.20 ± 6.16	45.52 ± 12.75	70.40 ± 11.05	0.38 ± 0.14	46.70 ± 8.06	81.40 ± 8.13	0.22 ± 0.09	708.71 ± 49.98	**15.56 ± 8.24**	**2.26 ± 0.39**
Point-wise	**6.76 ± 0.29**	7.45 ± 0.42	94.00 ± 3.11	0.09 ± 0.03	9.40 ± 0.47	93.15 ± 4.16	0.07 ± 0.04	**216.07 ± 17.85**	**17.22 ± 7.61**	**2.28 ± 0.24**
MaskPlanner w/out AP2S	7.79 ± 0.34	7.18 ± 0.30	**98.11 ± 0.74**	**0.05 ± 0.01**	11.27 ± 1.02	**97.00 ± 1.31**	**0.03 ± 0.02**	**220.66 ± 18.22**	**20.00 ± 4.69**	**2.30 ± 0.26**
MaskPlanner	**6.52 ± 0.28**	**6.83 ± 0.26**	**97.50 ± 0.85**	**0.05 ± 0.01**	**7.43 ± 0.35**	**98.10 ± 1.10**	**0.02 ± 0.01**	248.19 ± 41.39	**17.78 ± 12.51**	**2.32 ± 0.41**

We report Mean ± St.dev for 10 training repetitions. Mean values that are not statistically significantly worse than any other are marked in bold (α=0.05). Remind that all methods predict a fixed number of six output paths for the cuboids category, hence, no NoP metrics are computed.

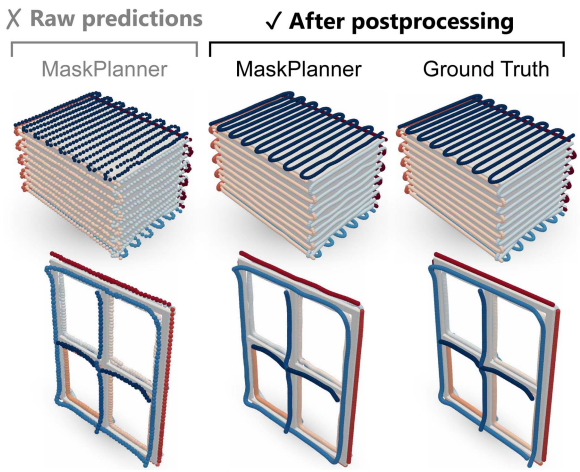

Fig. 11. Final paths when postprocessing is applied to segment predictions on cuboids and windows.

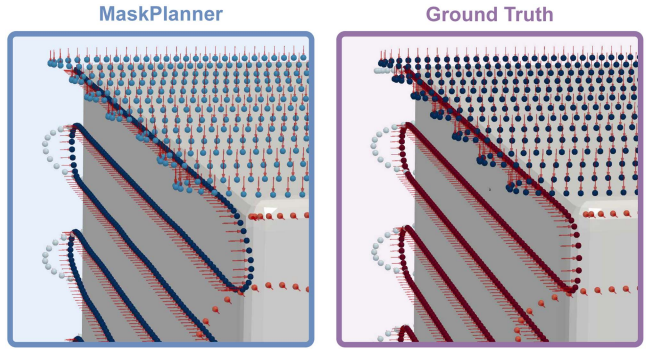

Fig. 12. Close up MaskPlanner results after postprocessing: both pose locations and orientations (red arrows) are effectively learned.

on the containers category. Generalization in such data-scarce conditions can be further improved by leveraging pretrained models or incorporating additional data across different categories, as discussed in Section V-E.

Postprocessing: The postprocessing step described in Section III-E aims at preparing the generated paths for execution on robotic systems. Its practical effect on concatenating the predicted segments can be observed in Figs. 11 and 12, leading to dense and smooth sequences of end-effector waypoints.

E. Results: Generalization Capabilities

By formalizing OCMG as a supervised learning task we are able to address key challenges commonly encountered in real-world industrial scenarios, such as the need to adapt to new data as it becomes available. To this end, data-driven approaches offer several advantages over ad hoc heuristics or traditional optimization-based methods. For example, models can be efficiently retrained without requiring process re-engineering, and one can leverage pretrained models to improve performance in cases of limited data or time constraints. In the following, we analyze how different training

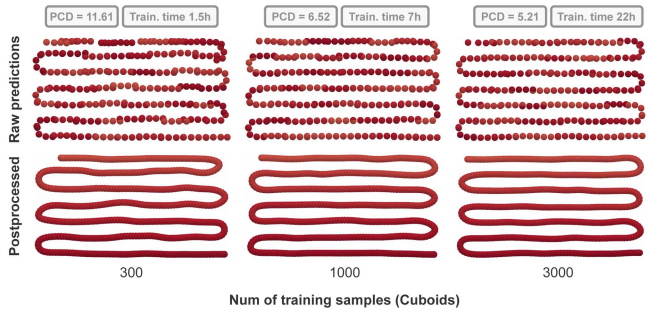

Fig. 13. Analysis of predictive performance over varying amounts of training data, on the Cuboids category. A representative predicted path at test time is depicted, before (top) and after (bottom) postprocessing.

configurations influence the generalization capabilities of our method.

Sensitivity to training set size: We investigate the performance trend of MaskPlanner when scaling up the amount of available training data. Specifically, we focus on the Cuboids category and report our findings for a model trained on 300, 1000, or 3000 samples. The results in Fig. 13 show that training on 300 Cuboids yields high PCD (i.e., high error) and paths with visible irregularities. These issues are largely mitigated when considering 1000 training samples and further improve when increasing the training set size to 3000.

Joint-category training: We extend the aforementioned analysis by studying the effect of jointly training MaskPlanner on multiple object categories with the goal of answering the following question: *"Does augmenting the training set with data from a different distribution but same task yield higher predictive performance?"* We illustrate the results in Fig. 15, where the basic model trained only on 70 containers is compared to a model jointly trained on cuboids, windows, shelves, and containers. Notably, the latter achieves significantly lower PCD on unseen containers. It is qualitatively evident that the containers-specific model is unable to preserve segments structure for novel shapes, showing more scattered predictions over the object surface. We conclude that experiencing higher shape and path diversity during training improves the model's generalization capability.

Finetuning on a novel category: Finally, we consider a knowledge transfer scenario in which MaskPlanner pre-trained on cuboids, shelves, and windows is subsequently finetuned on the containers category. We compare the resulting performance with that obtained by MaskPlanner in the vanilla setting, which leverages the task of shape classification on ModelNet as pretraining for backbone initialization. In particular, we inspect how performances vary across data and time constraints. The results in Fig. 16 show that exploiting pretrained models on task-specific data provides an advantage over the vanilla strategy in the few-shot scenario with scarce availability of training data. Similar conclusions can be drawn from the results in Fig. 17, where we analyze the convergence speed of the two approaches in terms of training epochs.

Generalization across highly-curved surfaces: The experiments presented so far primarily involve objects with flat or angular surfaces. To assess whether MaskPlanner can generalize to smoothly curved geometries—and whether the network is expressive enough to capture local curvature nuances—we introduce a novel benchmark category named Hills. We generate a large-scale dataset of 100 000 hilly surfaces by linearly combining 20 isotropic Gaussians, uniformly sampled with varying amplitudes and standard deviations over a 1 m $\times$ 1 m grid, holding out 1000 surfaces for testing. The results in Fig. 14 demonstrate that MaskPlanner yields smooth raster-like paths that precisely follow the undulatory surface geometry. In contrast, path-wise and autoregressive predictions struggle to adapt to the fine-grained variations in curvature, both qualitatively and quantitatively. Point-wise predictions, although locally accurate, remain an unordered set of poses that cannot be concatenated into executable paths. This experiment importantly indicates that, by predicting disconnected path segments, MaskPlanner can capture patterns that are highly sensitive to local curvature—without requiring complex surface reparameterizations or additional computational costs beyond processing the input point cloud.

Overall, the findings in this section provide a clear indication of the potential of supervised learning for future OCMG applications. In particular, the real-time inference capabilities of these models combined with an increasing amount of data can drastically reduce robot programming times.

TABLE III
QUANTITATIVE PC RESULTS, WITH MEAN AND ST.DEV

	PC % (↑)			
	Cuboids	Windows	Shelves	Containers
Path-wise	77.54 ± 11.89	81.50 ± 7.83	91.70 ± 3.58	58.37 ± 19.31
Autoregressive	88.49 ± 5.73	88.20 ± 6.17	89.10 ± 2.97	53.29 ± 17.06
Point-wise	**94.02 ± 8.16**	90.34 ± 7.30	92.34 ± 7.79	**88.29 ± 13.67**
MaskPlanner w/out AP2S	**99.87 ± 0.40**	**99.37 ± 0.73**	98.86 ± 2.13	**88.63 ± 17.16**
MaskPlanner	**99.82 ± 0.48**	**99.36 ± 0.62**	**99.76 ± 0.60**	**90.94 ± 12.93**

Across instances of the test set. Mean values that are *not* statistically significantly worse than any other are marked in bold (α=0.05).

F. Results: Spray Painting in Simulation

MaskPlanner and the proposed baselines aim at solving the spray painting task purely from expert data, i.e., achieving high PC despite not being explicitly optimized for a task-specific objective. To evaluate their capabilities on the downstream task, we run a spray painting simulation by placing the gun nozzle at each 6-D waypoint predicted by the network (as displayed in Fig. 9), at discrete time steps. By doing so, the resulting PC is invariant under permutation of the predicted waypoints and their arrangement into separate paths. Thus, we can fairly compare all the methods regardless of their point, segment, or path output structure. We execute the evaluation using a proprietary painting simulator, although similar tools would be equally suitable [4].

The results in Table III show two main trends: MaskPlanner achieves 1) substantially higher PC on average, and 2) exhibits the lowest variance across test samples on all object categories. Notably, over 99% of the object meshes are covered for test instances of cuboids, windows, and shelves when executing MaskPlanner's predictions.

Results on the containers provide further insights into the performance of MaskPlanner. In line with the PCD results in Table II, MaskPlanner attains a significant improvement in PC

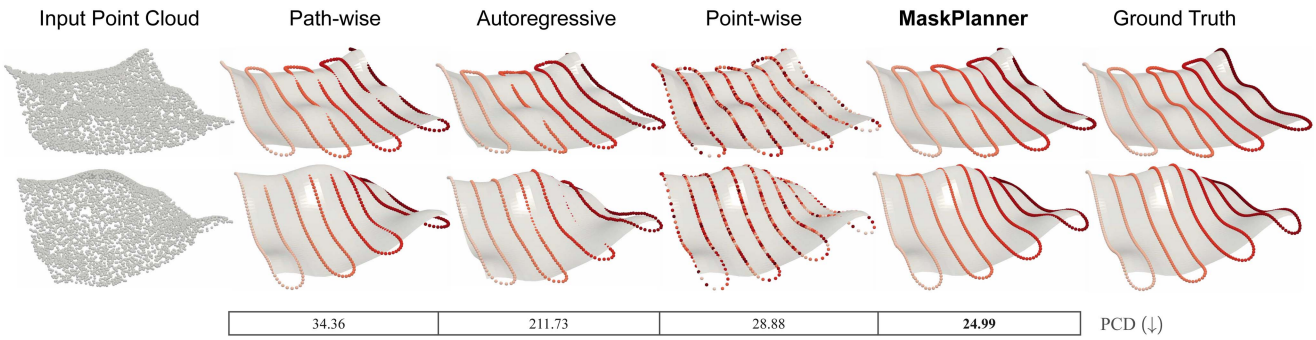

Fig. 14. Qualitative results on two representative test instances from the Hills benchmark. The reported PCD metric for each baseline is averaged over the entire test set of 1000 surfaces. A single raster-scan path is predicted for all surfaces.

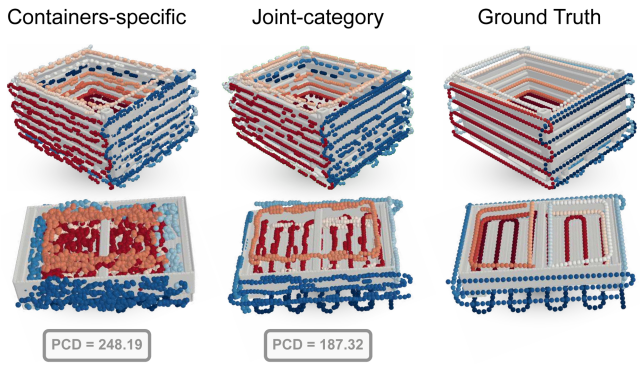

Fig. 15. Performance comparison on two test samples between a Containers-specific model versus a model trained on all categories combined (joint-category). Quantitative metrics refer to average over all 18 container test samples.

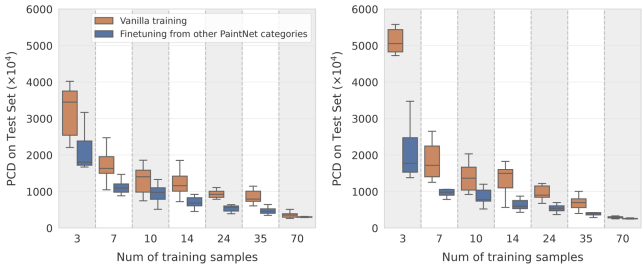

Fig. 16. Few-shot: A model jointly pretrained on cuboids, shelves, and windows generalizes better when finetuned on a subset of containers. Test results are shown after 600 (left) and 1200 (right) training epochs, averaged over 10 runs.

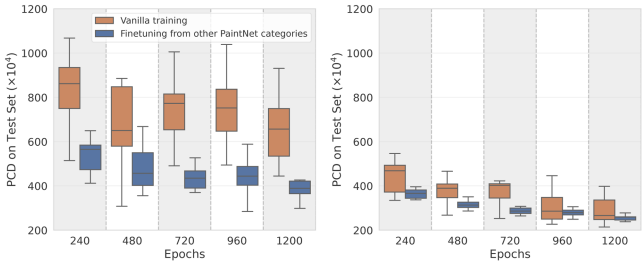

Fig. 17. Convergence speed: A model jointly pretrained on cuboids, shelves, and windows leads to faster convergence when finetuned on Containers. Test results after training with 50% (left) and 100% (right) of available containers, averaged over 10 runs.

compared to the path-wise and autoregressive baselines. We report qualitative results for a representative container in Fig. 18. Here, it is evident that path-wise and autoregressive predictions display issues in managing high dimensional paths or diverge due to compounding errors—as mentioned in Section V-D and highlighted in Fig. 9. As a result, large portions of the surface remain uncovered.

G. Results: Spray Painting Validation in the Real World

We conclude our experimental evaluation by demonstrating the successful application of MaskPlanner to a real-world industrial robotic system on the production line. Our validation aims to answer the following questions.

1) *Are paths predicted by MaskPlanner kinematically and dynamically feasible for direct execution on a real robot?*
2) *Does near-complete coverage in simulation translate to near-complete coverage of real objects?*

To this end, two representative object instances are drawn from the test set of Cuboids and Windows and hand-crafted by experts with their nominal dimensions. Note that, by doing so, the object point-cloud is readily available from our dataset and no domain shift between synthetic and real data is experienced at inference time. We design the experiment for powder coating, a widely adopted spray painting technique where paint is applied electrostatically to the object and subsequently cured under heat. We refer the reader to Fig. 19 for an illustration of the full real-world validation pipeline.

Inference and execution: We deploy the same category-specific models whose results have been discussed in Section V-D to predict output segments for the cuboid and window instances displayed in Fig. 19 (left). Predicted segments are then automatically concatenated through our postprocessing step to generate output paths. The entire path generation pipeline only takes 200 ms for each input object, requires a single forward pass of the network, and results in six paths for the cuboid (total length of 83 m) and 14 paths for the Window (total length of 17 m). The generated paths can be then executed on real robots by employing any approach of choice, such as tracking waypoints in Cartesian space. In our setting, we apply the Ramer–Douglas–Peucker algorithm—with thresholds of 1 cm for translational

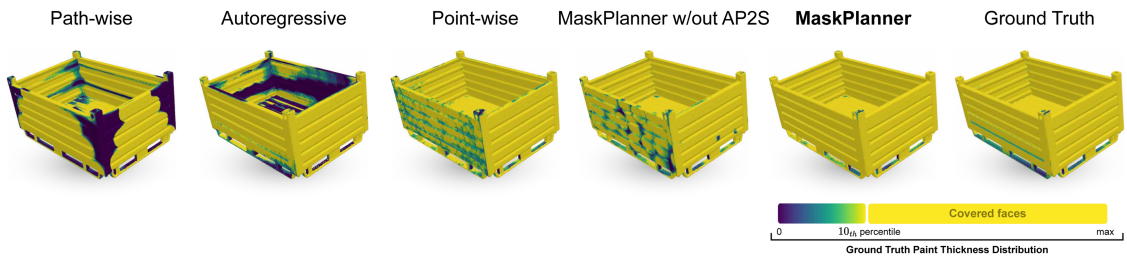

Fig. 18. Qualitative PC results in simulation on a representative test instance of the Containers category.

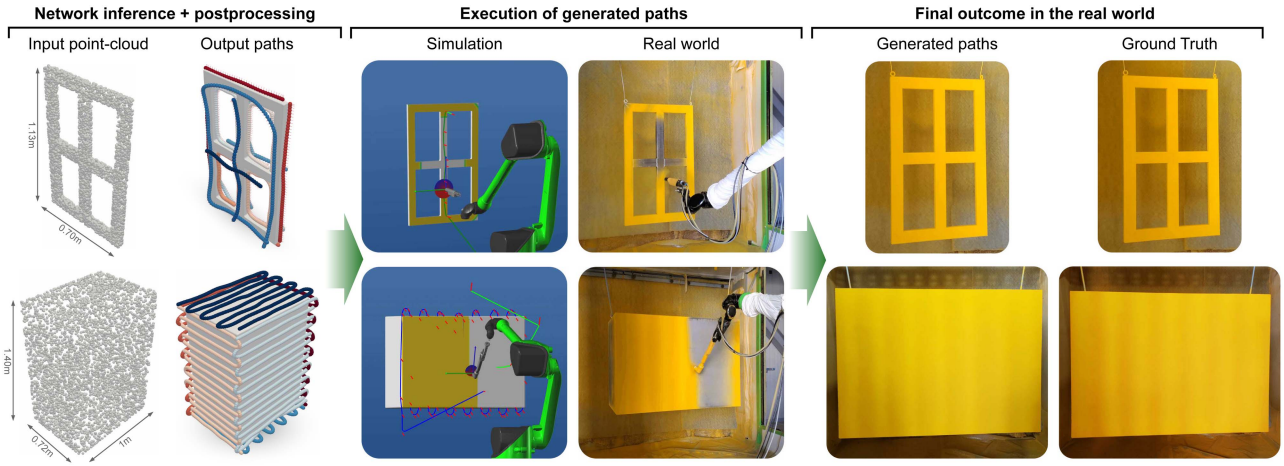

Fig. 19. Real-world validation of MaskPlanner on two test objects. A set of long-horizon paths is inferred given the object point clouds through a single forward pass (100 ms) and a postprocessing step (100 ms). Then, paths are checked for kinematic and dynamic feasibility in simulation and later executed on the real setup. The final paint result on the real objects is effectively equivalent to that produced by ground truth paths.

coordinates and 15° for rotational coordinates—and interpolate the subsampled waypoints through smooth junctions that preserve acceleration limits while targeting a desired velocity. We set the target velocity to 25 cm/s. A dedicated software automatically detects and manages reachability issues: in our physical setup it occasionally applied minor adjustments to the orientation normals of the paths (e.g., for the four inner and four outer edges of the window). Furthermore, motivated by the symmetries of the two objects, we avoid painting the back side of the window and cuboid without loss of generality. We execute paths in a random permutation, and reset the robot to a predefined starting configuration between each path execution. Both the generated and the ground-truth paths are then executed on a real-world Efort GR-680 6-DoF specialized painting robot, with their final outcome depicted in Fig. 19 (right).

Results and discussion: All paths generated by MaskPlanner are successfully executed on real hardware (see the accompanying video). Paths prove dynamically feasible for direct execution, with no collisions at any time. Importantly, the final spray painting outcome achieved with MaskPlanner trajectories prove indistinguishable from that of ground-truth expert trajectories across both objects (cf. Fig. 19). Recall that MaskPlanner includes no explicit paint optimization subroutine, rendering the final result a ground-breaking demonstration of its potential for addressing OCMG tasks through a pure data-driven perspective. Notably, we observe slight uneven paint deposition across both ground truth and predicted paths on the real Cuboid (see the vertical stripe trend in Fig. 19). This effect can be mitigated with application-specific trajectory optimization [6], or manual tuning of the target velocity by domain experts. Nevertheless, our real-world validation experiment demonstrates the effectiveness of MaskPlanner in quickly generating expert-level path patterns for previously unseen object instances, while requiring no domain knowledge and remaining applicable to a variety of OCMG tasks.

H. Results: Beyond Path Generation

So far, this work has focused on geometric path generation. In general, OCMG applications may demand the generation of ad hoc *trajectories*, as the velocity profile along a path may directly impact the quality of the final outcome. In our real-world validation (see Section V-G), we execute paths at a constant velocity of 25 cm/s informed by domain knowledge, which proves effective for the purpose of a PC analysis via powder coating. However, more complex objects or different paint types might require fine-grained control over the end-effector's speed to avoid issues like over- or under deposition of paint. We posit that MaskPlanner already provides a strong foundation for such scenarios by generating high-quality paths that can be quickly refined via downstream trajectory optimization with task-specific constraints.

To validate this claim, we extend our empirical analysis and implement the trajectory optimization step proposed by

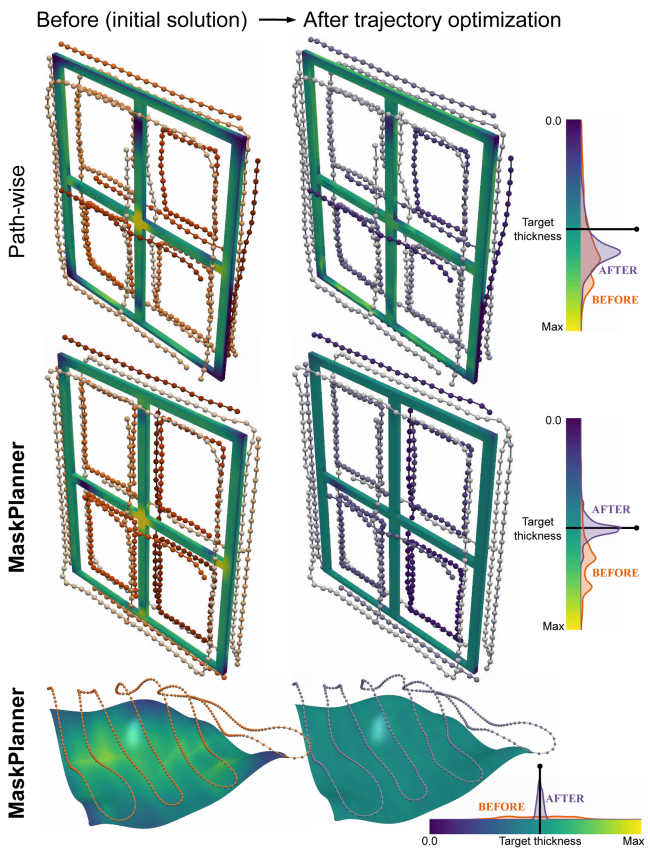

Fig. 20. Simulated paint deposition across the triangulated surface of a Window before (left) and after (right) applying a task-specific trajectory optimization step to the paths predicted by MaskPlanner and the path-wise baseline. The colorbars report the kernel density estimate of the paint thickness across mesh faces, before (orange) and after (purple) optimization.

Gleeson et al. [6], which includes a realistic model for the spatial distribution of paint particles emitted by a spray gun (cf. [6, Fig. 5]). This allows us to optimize MaskPlanner's predicted paths (after postprocessing) for a target paint thickness across the object, e.g., by triangulating the surface and computing the paint thickness deposited on each mesh face. Particularly, the optimization jointly refines the predicted 6-DoF waypoints and an additional per-waypoint time component, which determines the segment duration between consecutive waypoints. The objective function then minimizes the mean squared error between the amount of paint deposited on each mesh face and the target thickness. We set the target thickness to the midpoint between the minimum and the 98th percentile of the thickness distribution when executing the ground-truth paths at constant speed, allowing systematic testing and benchmarking.

We apply the aforementioned trajectory optimization step to the 14 paths predicted by MaskPlanner and the path-wise baseline on the same window instance analyzed in Fig. 9. The results are illustrated in Fig. 20, before and after running the optimization via the nonlinear solver IPOPT [84] for 100 iterations—equivalent to 3 h of wall-clock time on 8 CPU threads. The results clearly demonstrate the importance of a high-quality initial solution: when starting from MaskPlanner's paths, the downstream optimizer can successfully adjust the trajectories to produce a uniform paint thickness that closely matches the target value. In contrast, with the less accurate initial candidate solution provided by the *path-wise* baseline, the high-dimensional optimization problem is unable to correct for excessive geometric errors and converges to a suboptimal solution with significant thickness variation. We additionally report the results of trajectory optimization on a representative Hill instance (see Section V-E) in Fig. 20, importantly demonstrating the effectiveness of the proposed pipeline to obtain uniform paint deposition even on highly-curved surfaces. Overall, these findings crucially validate that MaskPlanner can be seamlessly integrated with optimization-based methods for spray painting applications and even scale, for the first time, to the joint optimization over multiple paths and full 3-D objects—rather than convex surfaces only. Finally, we note that such hybrid pipeline still requires substantial domain expertise to design the deposition model, and considerable computational time with respect to the path generation stage.

I. Results: Beyond Spray Painting

To further assess the general applicability of MaskPlanner within the realm of OCMG applications, we finally extend our evaluation beyond spray painting to novel tasks involving highly-curved surfaces and noncoverage path patterns. While applying learning-based methods to other industrial tasks—such as welding—is challenging due to the lack of publicly available expert data, we introduce two synthetic yet challenging tasks to systematically investigate the performance of MaskPlanner under these conditions. The first task, *ring-on-bottleneck*, requires predicting ring-shaped paths around the narrow sections of hourglass-like objects. We generate a dataset of 1000 instances with varying numbers, depths, and spacing among bottlenecks. The second task, *surface-tracking*, involves generating a single path along a highly-curved surface spanning 8 m in length. For this, we generate a large-scale dataset of 100 000 samples with randomly varying surface curvatures.

The results displayed in Fig. 21 confirm that our method is effectively applicable to these new domains. Particularly, MaskPlanner successfully identifies the correct number of paths for *all* test samples and accurately reproduces the curvilinear patterns from the expert data across both tasks. These findings suggest that the core principle of decomposing the problem into local path segments and global mask predictions enables learning arbitrary path structures that do not suffer from surface curvature, path length, path number, and scale of the training dataset. In contrast, we observe that autoregressive predictions are prone to divergence on long-horizon paths (cf. surface-tracking task), and that path-wise struggles in identifying the precise location and width of the narrow sections in the ring-on-bottleneck task.

VI. Conclusion

In this article, we address the core problem of robot motion generation conditioned on free-form 3-D objects. Particularly, we formalize the OCMG problem setting, aiming to unify several robotic applications under a common framework for the prediction of long-horizon, unstructured paths. To tackle this problem, we introduce MaskPlanner, a novel deep learning method capable of inferring smooth and accurate paths directly

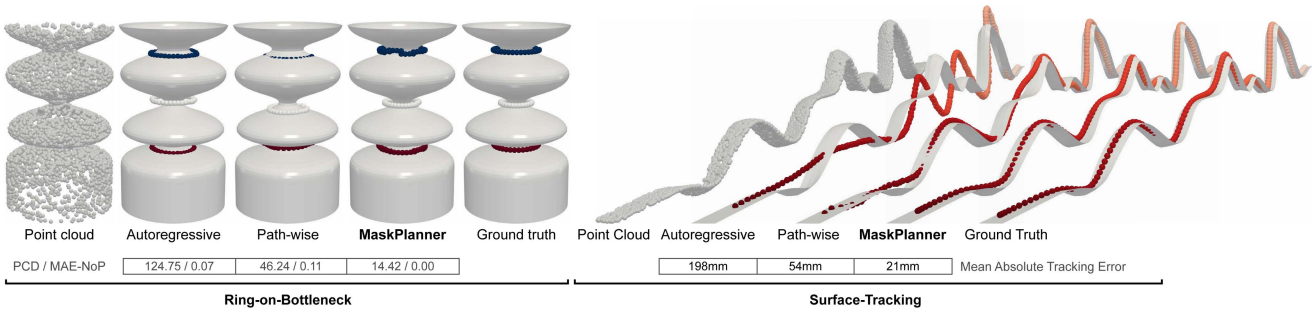

Fig. 21. Empirical results on previously unseen instances of two synthetically generated tasks: Ring-on-bottleneck (left) and surface-tracking (right). The reported quantitative metrics are averaged over the entire test set for each task.

from expert data. In simple terms, MaskPlanner breaks down motion generation into the joint prediction of local path segments and probability masks, followed by a postprocessing step that concatenates the segments predicted within the same mask. Our approach demonstrates generalization across both convex and concave 3-D objects (cuboids, windows, shelves), after being trained for only 6 hours on 800 samples on a single NVIDIA GeForce RTX 4080 GPU. We validate MaskPlanner in the context of robotic spray painting, demonstrating its ability to achieve near-complete PC on previously unseen objects, both in simulation and in real-world experiments. Our method remains task-agnostic and makes minimal assumptions on both the object geometry and the output path patterns, paving the way for future applications of deep learning to OCMG tasks beyond spray painting—such as welding, sanding, cleaning, and visual inspection. In fact, our findings showcase the potential of data-driven methods to achieve solutions to OCMG tasks that are scalable, generalizable, and cheap to deploy: models can do the following:

1) be pretrained with data from similar tasks;
2) boosted in performance when more samples become available;
3) leverage heuristics-based demonstrations;
4) be deployed at 10 Hz on large objects.

Remarkably, we also demonstrate a successful combination of our method with a downstream trajectory optimizer, and extend our empirical validation to tasks beyond spray painting featuring highly-curved surfaces. Finally, we note that combining our approach with ergodic coverage methods on complex domains, such as object meshes and point clouds, represents a promising future direction toward generating trajectories that both maximize coverage and imitate expert demonstrations. Extending our method to data modalities beyond 3-D objects in future work may also pave the way to tackle tasks, such as agricultural coverage planning from aerial images and multi-UAV search-and-rescue.

Limitations: Our current pipeline relies on a simple postprocessing strategy to concatenate predicted segments, which is not able to recover from erroneous predictions and may lead to inaccurate paths in such cases. Learning-based modules for concatenation will be investigated in future work to be robust to misplaced segment predictions. We also reckon that the choice of the segment length $\lambda = 4$ may not work equally well across different tasks and objects. This hyperparameter should be tuned accordingly to learn accurate isolated patterns that generalize well across the object surface. Alternatively, tuning the granularity of the sampled waypoints from the expert trajectories to achieve the desired degree of sparsity in the predicted poses would serve a similar purpose. Finally, our practical implementation simplifies gun orientations to unit vectors (from 3-DoF to 2-DoF), making the sole assumption that the task is invariant to gun rotations around the approach axis. Future work can extend the pipeline to the prediction of full orientation representations—e.g., Euler angles—to tackle tasks where the additional degree of freedom is necessary, such as motion generation for object grasping.

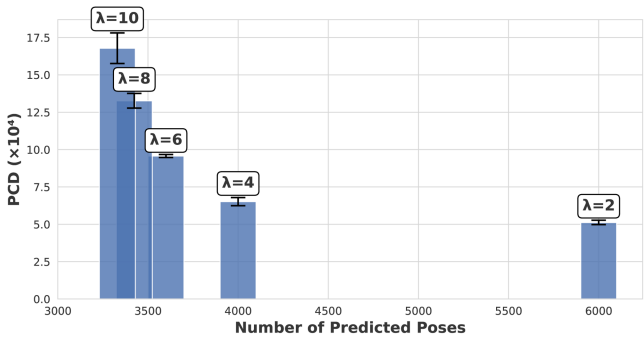

Fig. 22. Predictive performance on test Cuboids in terms of PCD, for varying segments length $\lambda \in \{2, 4, 6, 8, 10\}$. Notice that lower values of λ result in a larger number of poses to predict, as adjacent ground truth segments overlap by one pose.

APPENDIX A
AP2S CURRICULUM

When training MaskPlanner to minimize the proposed loss $\mathcal{L}_{\text{p2s}}$ (see Section III-C), various weighting schemes can be applied to the point-wise and segment-wise CD terms. In principle, an ideal learning algorithm would drive both point-wise and segment-wise matching simultaneously. In practice, however, gradient-based optimization can converge to substantially different local optima—e.g., while the CD is computationally efficient, it is known to be sensitive to outliers and insensitive to local mismatches in density [76]. In our ablation study (see Table IV), we evaluate the following four main weighting configurations.

1) *MaskPlanner w/out AP2S*: A baseline with a fully segment-wise loss function that leverages no auxiliary

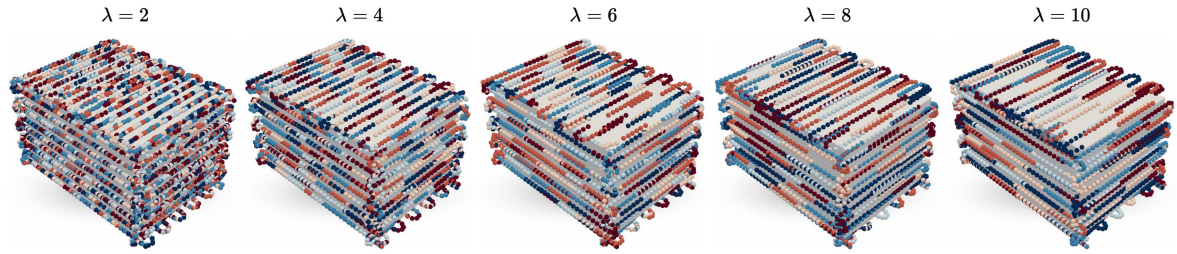

Fig. 23. Qualitative segment predictions for MaskPlanner models trained with varying segments length $\lambda \in \{2, 4, 6, 8, 10\}$ (colors differentiate among segments). Models trained with larger λ values display better segment structure and predict fewer overall output poses, but fail at predicting contiguous segments along the same path, resulting in higher PCD scores.

TABLE IV
PCD ON THE TEST SET OF ALL OBJECT CATEGORIES WHEN MODELS ARE TRAINED ON $\mathcal{L}_{\text{P2S}}$ WITH DIFFERENT WEIGHTING CONFIGURATIONS (10 SEEDS)

	MaskPlanner w/out AP2S	(1) Asymmetric	(2) P2S Curr.	(1) + (2) **MaskPlanner**
	$w_p^f = 0$ $w_s^f = 1$ $w_p^b = 0$ $w_s^b = 1$	$w_p^f = 0$ $w_s^f = 1$ $w_p^b = 1$ $w_s^b = 0$	$w_p^f = 100 \rightarrow 1$ $w_s^f = 0.01 \rightarrow 1$ $w_p^b = 100 \rightarrow 1$ $w_s^b = 0.01 \rightarrow 1$	$w_p^f = 0$ $w_s^f = 1$ $w_p^b = 100 \rightarrow 1$ $w_s^b = 0.01 \rightarrow 1$
Cuboids	7.79	11.81	9.24	**6.52**
Windows	7.18	12.55	11.89	**6.83**
Shelves	11.27	30.65	34.84	**7.43**
Containers	**220.66**	242.07	240.19	248.19

The "$a \rightarrow b$" notation indicates the value varies from a to b across training.
The bold text in the header row emphasizes the method in its final form, distinguishing it from its variants.

point-wise CD terms. This weighting scheme also resembles the loss function introduced in [16].

2) *Asymmetric*: Keeps a segment-wise forward CD, but uses a point-wise loss function for the backward CD.
3) *P2S curriculum*: Progressively assigns more weight to segment-wise predictions along training, but does so symmetrically for both forward and backward terms.
4) *1) + 2) MaskPlanner*: Our asymmetric curriculum that starts training with a higher weight on the backward point-wise term and gradually balances both point-wise and segment-wise backward terms. The forward term is fully segment-wise throughout training.

Quantitative results show that MaskPlanner converges to lower PCD scores when both asymmetric CD terms and a gradual point-to-segment curriculum are included. In particular, we observe that including the computation of auxiliary point-wise terms only helps if used in combination with the AP2S curriculum. A variety of additional configurations have also been tried, but led to no improvements. Moreover, we remind that naïvely optimizing for fully point-wise CD terms yields qualitative sparse predictions that fail in capturing detailed path structures (see point-wise baseline in Fig. 9). In turn, we conclude that the AP2S curriculum is crucial to promote effective convergence while ensuring smoothness and local consistency in the final predictions.

APPENDIX B
SENSITIVITY TO SEGMENT LENGTH

This section presents a sensitivity analysis on the segment length λ, which impacts the model performance and the structure of the output paths. In particular, decreasing λ also increases the total number of generated ground truth segments K, and thus, the dimensionality of the mask encodings $\boldsymbol{m}^i$. Notably, the overall number of output poses to be predicted also increases, as adjacent segments within the same path overlap by one pose. As a result, shorter path segments yield larger model sizes and demand higher network capacity. We investigate this relationship in Fig. 22. We notice that shorter segment lengths tend to reduce the PCD, while predicting a larger number of poses. We further inspect the qualitative predictions of these models in Fig. 23. We importantly find that a lower PCD does not necessarily imply better predictions, aligning with the results of our main experimental evaluation (see point-wise baseline in Table II and Fig. 9). On the other hand, it is evident from Fig. 23 that models trained with larger λ values progressively fail to maintain continuity along the same path, although display well structured segments.

Overall, while flexible, the choice of λ critically defines a tradeoff for learning local versus global geometric patterns between the input object point cloud and the output paths. Specifically, low values (e.g., λ=1) limit the ability of the model to directly predict long-horizon structured paths, and instead yield behavior akin to shape completion networks that generate unordered waypoints (cf. point-wise baseline). Conversely, excessively high values of λ push the model toward a path-wise prediction paradigm, which attempts to learn global mappings from object geometry to high-dimensional output and fails in capturing coherent expert patterns in the process.

ACKNOWLEDGMENT

The authors would also like to thank the support of the European H2020 Elise project (www.elise-ai.eu), the CINECA award under the ISCRA initiative (DRE-URL - HP10CF881 L) for the availability of HPC resources and support, and the EFORT group, providing the authors with support, domain knowledge, original object meshes, trajectory data, and access to the proprietary spray painting simulator and hardware used during the experiments.

REFERENCES

[1] W. Sheng, N. Xi, M. Song, Y. Chen, and P. MacNeille, "Automated CAD-guided robot path planning for spray painting of compound surfaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2000, pp. 1918–1923.
[2] H. Chen and N. Xi, "Automated tool trajectory planning of industrial robots for painting composite surfaces," *Int. J. Adv. Manuf. Technol.*, vol. 35, pp. 680–696 2008.

[3] X. Li, O. A. Landsnes, H. Chen, M.-V. Sudarshan, T. A. Fuhlbrigge, and M.-A. Rege, "Automatic trajectory generation for robotic painting application," in *Proc. 41st Int. Symp. Robot. (6th German Conf. Robot.)*, 2010, pp. 1–6.

[4] M. Andulkar and S. Chiddarwar, "Incremental approach for trajectory generation of spray painting robot," *Ind. Robot: Int. J.*, vol. 42, pp. 228–241 2015.

[5] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Uniform coverage of automotive surface patches," *Int. J. Robot. Res.*, vol. 24, no. 11, pp. 883–898, 2005.

[6] D. Gleeson et al., "Generating optimized trajectories for robotic spray painting," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1380–1391, Jul. 2022.

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

[8] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9813–9823.

[9] J. Ngiam et al., "Scene Transformer: A unified architecture for predicting future trajectories of multiple agents," in *Proc. Int. Conf. Learn. Represent.*, 2022. [Online]. Available: https://openreview.net/forum?id=Wm3EA5OlHsG

[10] B. Varadarajan et al., "Multipath : Efficient information fusion and trajectory aggregation for behavior prediction," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 7814–7821.

[11] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "WayFormer: Motion forecasting via simple & efficient attention networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 2980–2987.

[12] 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.

[13] M. K. H. Kretzschmar and C. S. W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Robotics: Science and Systems VIII*. Boca Raton, FL, USA: MIT Press, 2013, pp. 193–200.

[14] H. Kretzschmar, M. Kuderer, and W. Burgard, "Learning to predict trajectories of cooperatively navigating agents," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 4015–4020.

[15] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2096–2101.

[16] G. Tiboni, R. Camoriano, and T. Tommasi, "Paintnet: Unstructured multi-path learning from 3D point clouds for robotic spray painting," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 3857–3864.

[17] K. E. Bekris, J. Doerr, P. Meng, and S. Tangirala, "The state of robot motion generation," in *Proc. Int. Symp. Robot. Res.*, 2024.

[18] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.

[19] J. Wang et al., "A survey of learning-based robot motion planning," *IET Cyber- Syst. Robot.*, vol. 3, no. 4, pp. 302–314, 2021.

[20] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT∗: Learning-based optimal path planning," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1748–1758, Oct. 2020.

[21] R. Cheng, K. Shankar, and J. W. Burdick, "Learning an optimal sampling distribution for efficient motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7485–7492.

[22] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Sci. Robot.*, vol. 5, no. 48 2020, Art. no. eabd7710.

[23] P. Kicki et al., "Fast kinodynamic planning on the constraint manifold with deep neural networks," *IEEE Trans. Robot.*, vol. 40, pp. 277–297, 2024.

[24] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 3965–3972.

[25] A. Sasagawa, S. Sakaino, and T. Tsuji, "Motion generation using bilateral control-based imitation learning with autoregressive learning," *IEEE Access*, vol. 9, pp. 20508–20520, 2021.

[26] N. Saito et al., "Structured motion generation with predictive learning: Proposing subgoal for long-horizon manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 9566–9572.

[27] P. M. Viceconte et al., "Adherent: Learning human-like trajectory generators for whole-body control of humanoid robots," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2779–2786, Apr. 2022.

[28] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Trans. Graph. (ToG)*, vol. 37, no. 4, pp. 1–11, 2018.

[29] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist. JMLR Workshop Conf. Proc.*, 2011, pp. 627–635.

[30] T. Osa et al., *An Algorithmic Perspective on Imitation Learning*. Hanover, MA, USA: Now Publishers, 2018, ch. 3.

[31] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 4572–4580.

[32] Y. Pan et al., "Imitation learning for agile autonomous driving," *Int. J. Robot. Res.*, vol. 39, no. 2/3, pp. 286–302, 2020.

[33] 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 *Proc. Robotics: Sci. Syst.*, 2024, pp. 288–297.

[34] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 1–8.

[35] B. D. Ziebart et al., "Maximum entropy inverse reinforcement learning," in *Proc. 23rd Nat. Conf. Artif. Intell.*, 2008, vol. 8, pp. 1433–1438.

[36] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proc. IEEE RSJ Int. Conf. Intell. Robots Syst*, 2008, pp. 1083–1090.

[37] M. Wulfmeier, D. Z. Wang, and I. Posner, "Watch this: Scalable cost-function learning for path planning in urban environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2089–2095.

[38] F. Shkurti, N. Kakodkar, and G. Dudek, "Model-based probabilistic pursuit via inverse reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 7804–7811.

[39] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 819–826, Apr. 2017.

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

[41] M. Ye, T. Cao, and Q. Chen, "TPCN: Temporal point cloud networks for motion forecasting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11318–11327.

[42] J. Schulman, J. Ho, C. Lee, and P. Abbeel, *Learning From Demonstrations Through the Use of Non-Rigid Registration*. Cham, Switzerland: Springer, 2016, pp. 339–354.

[43] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2016, pp. 961–971.

[44] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2255–2264.

[45] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *Int. J. Robot. Res.*, vol. 39, no. 8, pp. 895–935, 2020.

[46] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, "Spatio-temporal graph transformer networks for Pedestrian trajectory prediction," in *Proc. 16 th Eur. Conf. Comput. Vis.*, 2020, pp. 507–523.

[47] J. Gu et al., "Vip3D: End-to-end visual trajectory prediction via 3D agent queries," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 5496–5506.

[48] S. H. Rezatofighi, V. K. Bg, A. Milan, E. Abbasnejad, A. Dick, and I. Reid, "Deepsetnet: Predicting sets with deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5257–5266.

[49] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2147–2154.

[50] J. Redmon, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2016, pp. 779–788.

[51] O. Vinyals, S. Bengio, and M. Kudlur, "Order Matters: Sequence to sequence for sets," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[52] R. Stewart, M. Andriluka, and A. Y. Ng, "End-to-end people detection in crowded scenes," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2016, pp. 2325–2333.

[53] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.

[54] B. Cheng, A. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 17864–17875.

[55] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1/2, pp. 83–97, 1955.
[56] E. Ahmed et al., "A survey on deep learning advances on different 3D data representations," 2018, *arXiv:1808.01462*.
[57] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
[58] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet : Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.
[59] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 728–737.
[60] A. Alliegro, D. Valsesia, G. Fracastoro, E. Magli, and T. Tommasi, "Denoise and contrast for category agnostic shape completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4627–4636.
[61] A. Rosasco, S. Berti, F. Bottarel, M. Colledanchise, and L. Natale, "Towards confidence-guided shape completion for robotic applications," in *Proc. IEEE-RAS 21st Int. Conf. Humanoid Robots*, 2022, pp. 580–586.
[62] S. S. Mohammadi et al., "3DSGRASP: 3D shape-completion for robotic Grasp," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 3815–3822.
[63] G. Biegelbauer, A. Pichler, M. Vincze, C. Nielsen, H. Andersen, and K. Haeusler, "The inverse approach of flexpaint [robotic spray painting], vol. 12, no. 3, pp. 24–34, Sep. 2005.
[64] A. Jonnarth, J. Zhao, and M. Felsberg, "Learning coverage paths in unknown environments with deep reinforcement learning," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, pp. 22491–22508. [Online]. Available: https://openreview.net/forum?id=nCZYRBK1J4
[65] W. Jing, D. Deng, Y. Wu, and K. Shimada, "Multi-UAV coverage path planning for the inspection of large and complex structures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1480–1486.
[66] S. Ivić, B. Crnković, L. Grbčić, and L. Matleković, "Multi-UAV trajectory planning for 3D visual inspection of complex structures," *Autom. Construction*, vol. 147, 2023, Art. no. 104709.
[67] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.
[68] M. Bosse, N. Nourani-Vatani, and J. Roberts, "Coverage algorithms for an under-actuated car-like vehicle in an uncertain environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 698–703.
[69] A. Bircher et al., "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Auton. Robots*, vol. 40, pp. 1059–1078, 2016.
[70] C. Bilaloglu, T. Löw, and S. Calinon, "Tactile ergodic coverage on curved surfaces," *IEEE Trans. Robot.*, vol. 41, pp. 1421–1435.
[71] D. Dong, A. Xu, G. Gutow, H. Choset, and I. Abraham, "Ergodic exploration over meshable surfaces," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 13088–13094.
[72] C. Hughes, H. Warren, D. Lee, F. Ramos, and I. Abraham, "Ergodic trajectory optimization on generalized domains using maximum mean discrepancy," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 1–7.
[73] W. Chen, X. Li, H. Ge, L. Wang, and Y. Zhang, "Trajectory planning for spray painting robot based on point cloud slicing technique," *Electronics*, vol. 9, no. 6, 2020, Art. no. 908.
[74] Z. Huang, W. Heng, and S. Zhou, "Learning to paint with model-based deep reinforcement learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8708–8717.
[75] V. Vapnik, "Principles of risk minimization for learning theory," in *Proc. Adv. Neural Inf. Process. Syst.*, 1991, vol. 4, pp. 831–838.
[76] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin, "Density-aware chamfer distance as a comprehensive metric for point cloud completion," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 29088–29100.
[77] J. Edmonds et al., "Optimum branchings," *J. Res. Nat. Bur. Standards B*, vol. 71, no. 4, pp. 233–240, 1967.
[78] J. Huang, H. Su, and L. Guibas, "Robust watertight manifold surface generation method for shapenet models," 2018, *arXiv:1802.01698*.
[79] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. Graph.*, vol. 5, no. 1, pp. 51–72, 1986.
[80] Z. Wu et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
[81] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[82] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2463–2471.
[83] P. A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martıńez, "Ordinal regression methods: Survey and experimental study," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 127–146, Jan. 2016.
[84] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.



Gabriele Tiboni received the M.Sc. degree in data science and engineering from Politecnico di Torino, Torino, Italy, in collaboration with Aalto University, Finland, in 2021.

He subsequently pursued the National Ph.D. AI Programme with Politecnico di Torino, with a research focus on deep learning-based methods for motion generation conditioned on 3-D objects. During his doctoral studies, he enrolled in the ELLIS Ph.D. and Post-Doc Programme and the ELIZA School of Excellence, collaborating with TU Darmstadt, Darmstadt, Germany, on the development of novel reinforcement learning approaches for robotic manipulation tasks. He later joined as a Postdoctoral Researcher the LiteRL Group, JMU Würzburg, Würzburg, Germany, and the Pearl Lab, TU Darmstadt.



Raffaello Camoriano (Member, IEEE) received the Ph.D. degree in bioengineering and robotics from the University of Genoa, Genoa, Italy, in 2017.

He has been a Research Fellow and a Postdoctoral Researcher with the Istituto Italiano di Tecnologia, Genoa, Italy, from 2014 to 2022. He is currently an Assistant Professor with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy, and an Affiliated Researcher with Istituto Italiano di Tecnologia. He is author of 25 international peer-reviewed articles in the fields of efficient learning from structured data, continual learning, reinforcement learning, and robotics.

Prof. Raffaello Camoriano was the recipient of the IEEE Computational Intelligence Society Italy Section Chapter's 2017 Best Ph.D. Thesis Award and is a Member of ELLIS. He is currently an Associate Editor for IEEE ROBOTICS AND AUTOMATION LETTERS and SPRINGER MACHINE LEARNING.



Tatiana Tommasi received the Ph.D. degree in electrical engineering from EPFL Lausanne, Lausanne, Switzerland, in 2013.

Subsequently, she held postdoctoral roles in both Belgium and the USA. She is currently a Full Professor with the Control and Computer Engineering Department, Polytechnic University of Turin, Turin, Italy, and the Director of the ELLIS Unit, Turin, Italy. Before her current position, she was an Assistant Professor with Sapienza University, Rome, Italy. Her publication record boasts more than 50 papers in top conferences and journals, specializing in machine learning and computer vision. She pioneered the field of transfer learning in computer vision and possesses extensive experience in areas, such as domain adaptation, generalization, multimodal learning, and open-set learning.

Prof. Tommasi is currently an Associate Editor for IEEE TRANSACTION ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and IEEE TRANSACTION ON EMERGING TOPICS IN COMPUTING.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement