2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

DYNAMO: Dependency-Aware Deep Learning Framework for Articulated Assembly Motion Prediction

Mayank Patel[1], Rahul Jain[2], Asim Unmesh[2], and Karthik Ramani[1,2]

Abstract— **Understanding the motion of articulated mechanical assemblies from static geometry remains a core challenge in 3D perception and design automation. Prior work on everyday articulated objects such as doors and laptops typically assumes simplified kinematic structures or relies on joint annotations. However, in mechanical assemblies like gears, motion arises from geometric coupling, through meshing teeth or aligned axes, making it difficult for existing methods to reason about relational motion from geometry alone. To address this gap, we introduce MechBench, a benchmark dataset of 693 diverse synthetic gear assemblies with part-wise ground-truth motion trajectories. MechBench provides a structured setting to study coupled motion, where part dynamics are induced by contact and transmission rather than predefined joints. Building on this, we propose DYNAMO, a dependency-aware neural model that predicts per-part $\mathrm{SE}(3)$ motion trajectories directly from segmented CAD point clouds. Experiments show that DYNAMO outperforms strong baselines, achieving accurate and temporally consistent predictions across varied gear configurations. Together, MechBench and DYNAMO establish a novel systematic framework for data-driven learning of coupled mechanical motion in CAD assemblies. Project page: https://dynamo-web.pages.dev/**

I. Introduction

Articulated mechanical assemblies ranging from gear trains and linkages to cabinets are ubiquitous in engineered systems and everyday devices. For robots to interact with, manipulate, or even design such systems, it is crucial to understand how their constituent parts move. While modern CAD tools (e.g., SOLIDWORKS, Autodesk Fusion 360, Siemens NX) provide detailed 3D geometry, they rarely include executable motion specifications, limiting their utility for robotics applications [1] such as assembly automation, motion planning [2]–[4], and simulation [5], [6]. Bridging this gap requires methods that can infer physically valid part motions directly from static geometry.

To infer how parts move, a wide range of methods have been proposed that operate on inputs such as RGB-D images [7]–[9], 3D meshes [10]–[12], or point clouds [13]–[15]. These methods span supervised [8], [16], self-supervised [15], [17], and generative paradigms [18]–[20], and typically aim to predict either the mobility of individual parts [21] or the motion fields of articulated objects [4], [16], [22]. A wide range of methods have been proposed that rely on synthetic datasets with clean segmentation masks and joint information [12], while other methods operate on real-world scans [23], [24] to predict mobility, which is then used to derive motion. Broadly, these works estimates parts mobility parameters such as part axis, joint type, motion type, joint limits, learning deformation fields, or segmenting objects into movable components.

While prior works have explored various aspects of articulated object manipulation and motion understanding, most methods model kinematics by estimating the mobility of individual parts and then applying a fixed set of equations for motion generation [5], [14]. These approaches are effective when the motions of different parts are independent, where modelling individual mobilities often yields accurate predictions. However, in mechanical assemblies, part motions are inherently interdependent, and motion propagates from one component to another through complex coupling. Capturing this propagation is a fundamental yet underexplored aspect of articulated objects, and it is especially critical for accurately modelling and generating motions in mechanical assemblies.

To explore the problem of motion prediction under coupling in articulated assemblies, we focus on gear-based mechanisms such as rack-and-pinion systems and worm drives, which present unique challenges due to their interdependent part motions. In these systems, motion propagates through contact, meshing, and transmission, making it difficult for traditional models to reason about relational motion from geometry alone. Previous dataset have touched on coupling only in simple objects, as the construction of articulated CAD assemblies is labor intensive and requires detailed part modeling and articulation definitions [25]. To address this gap, we introduce the MechBench, a novel dataset consisting of 693 synthetic gear assemblies with diverse configurations, providing a structured benchmark for predicting rigid-body motion in complex assemblies. Beyond the dataset, we propose **DYNAMO**, a dependency-aware neural model designed to address the coupling problem by predicting per-part SE(3) trajectories using 6D Lie algebra vectors. DYNAMO learns motion patterns directly from static, segmented CAD geometry, without relying on joint annotations or predefined kinematic structures, and integrates PointNet++ for part-level feature extraction, a graph neural network for modeling inter-part relationships, and a temporal decoder for trajectory prediction, enabling end-to-end learning of coupled rigid-body motion.

Our key contributions are:

- We introduce MechBench, a novel benchmark dataset of 693 diverse synthetic gear assemblies with part-wise ground-truth motion labels, enabling systematic evaluation of coupled mechanical motion prediction.
- We present DYNAMO, a dependency-aware neural

[1]School of Mechanical Engineering, Purdue University, USA
[2]Department of Electrical and Computer Engineering, Purdue University, USA

model that jointly reasons over inter-part relationships to predict per-part SE(3) motion trajectories from static CAD geometry.
- We compare against baselines and conduct ablation studies, demonstrating that DYNAMO achieves accurate and temporally consistent motion predictions on MechBench across multiple evaluations.

II. Related Work

Research on articulated object motion prediction has largely focused on everyday objects, aiming to infer part mobility or deformation fields from static geometry. Early works such as Shape2Motion [21] proposed joint discovery frameworks, while later methods explored supervised [5], [16], weakly-supervised [14], and self-supervised paradigms [15], [26]. Other approaches model point-wise motion fields [4], [22], or track part poses given temporal cues [27]. These methods have driven progress but generally treat parts independently, assuming simple kinematic structures or relying on ground-truth joints.

To support these tasks, several synthetic datasets have been introduced. Early efforts such as Hu et al. [12] and RPM-Net [4] provided motion annotations at limited scale, while Shape2Motion [21] and PartNet-Mobility [28] expanded to thousands of objects across diverse categories. More recent datasets such as OPDSynth [29] and ACD [30] target more realistic variations. However, these resources primarily capture independent part mobility and rarely include coupled motion. Table I compares representative datasets, showing that MechBench is comparable in scale while uniquely addressing motion prediction under inter-part coupling in gear assemblies.

TABLE I
Comparison of synthetic articulated object datasets.

Dataset	#Objects	#Mov. Parts	#Categories	Coupling
Hu et al. [12]	368	368	–	–
RPM-Net [4]	969	1420	43	Few
Shape2Motion [21]	2440	6762	45	–
RBO [31]	14	21	14	–
PartNet-Mobility [28]	2440	6762	45	–
OPDSynth [29]	683	1343	11	–
ACD [30]	354	1350	21	–
MechBench (Ours)	**693**	**2445**	**13**	**Yes**

Mechanical assemblies present additional challenges beyond everyday articulated objects. Traditional kinematic modeling frameworks such as Mujoco [32] and Bullet [33] assume known joints, while analysis methods [2], [10] attempt to infer mobility from contact. More recent learning approaches predict kinematic graphs from 3D data [34], [35], but still rely on explicit joint representations. These assumptions break down for gears and transmissions, where motion propagates through meshing and contact rather than predefined joints. Addressing such coupled motion also depends on how rigid-body trajectories are represented: while quaternions or 6D continuous rotation formats [36], [37] are widely used, they mainly target single-part pose estimation, and flow-based methods [4] lack rigid consistency. In contrast, our work adopts a Lie algebra formulation of SE(3), predicting per-part motion trajectories that are both valid and continuous, and enabling dependency-aware reasoning over coupled motions in complex assemblies.

III. Dataset

Gear-based assemblies such as rack-and-pinion mechanisms, planetary trains, and worm drives present an ideal setting for studying *coupled motion* of parts, where the movement of one part directly induces motion in another through meshing and contact. Unlike everyday articulated objects (e.g., doors or laptops) with independent joints, gears capture the challenge of dependency-driven motion propagation. To enable learning in this setting, we construct **MechBench**, a benchmark dataset of synthetic CAD-generated gear assemblies with ground-truth motion labels.

A. Assembly Creation

We procedurally generate assemblies using FreeCAD by varying key design parameters, including gear type (spur, helical, double-helical, bevel, worm, rack-and-pinion, planetary), number of teeth, diameter, pressure angle, and tooth profile. Figure 1 illustrates examples across these categories. The number of parts per assembly is chosen randomly, ensuring valid meshing and mechanical compatibility. Each assembly is saved in two formats: (i) a `STEP` file preserving part hierarchy and geometry, and (ii) a segmented point cloud with up to $N_{\max} = 1024$ points per part, used for learning.

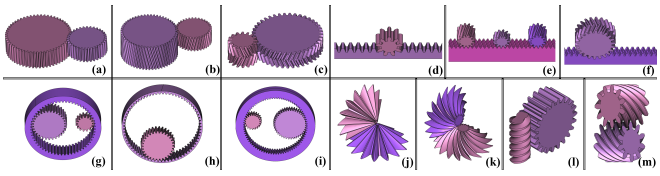

Fig. 1. Examples of diverse gear assemblies in our dataset, including: (a–c) spur and helical pairs, (d–f) planetary systems, (g–i) rack-and-pinion mechanisms, and (j–m) worm, bevel, and compound gear configurations.

B. Dataset Annotations

Our dataset consists of comprehensive annotations for each assembly, including 3D meshes, segmentation masks, part-wise point clouds sampled uniformly from surfaces, frame-wise per part rigid-body motion annotations, and mobility units describing motion type, axis, and degrees of freedom.

1) Motion Annotation: For motion labels, we represent rigid-body motion using a 6D Lie algebra vector (6DLAV) $\boldsymbol{\xi} \in \mathbb{R}^6$, also known as a twist vector, derived from the Lie algebra $\mathfrak{se}(3)$ of the transformation group $\mathrm{SE}(3)$.

$$\boldsymbol{\xi} = [\boldsymbol{\omega}^\top, \mathbf{v}^\top]^\top, \quad \boldsymbol{\omega} \in \mathbb{R}^3,\ \mathbf{v} \in \mathbb{R}^3$$

where $\boldsymbol{\omega}$ encodes angular velocity and $\mathbf{v}$ encodes linear velocity.

Rigid-body motions are traditionally represented as an $\mathrm{SE}(3)$ matrix, but direct regression of $R \in \mathrm{SO}(3)$ is unstable due to orthogonality and determinant constraints.

Instead, 6DLAV provides a compact, continuous, and valid representation, easily converted to $\mathrm{SE}(3)$ via the exponential map:

$$T = \exp\left(\hat{\boldsymbol{\xi}}\right) \in \mathrm{SE}(3)$$

For each assembly, one gear is selected as the driver, rotating at 10° per frame. The motion of other parts is determined by gear ratios. The ground-truth $\mathrm{SE}(3)$ transformation for part k at frame t is:

$$T_t^{(k)} = T_{\text{center}}^{-1} \cdot R(\theta_t) \cdot T_{\text{center}}, \quad \theta_t = \theta_0 + t \cdot \Delta\theta^{(k)}$$

2) Mobility Annotation: In addition, we store *mobility units* for each part:

$$\mu^{(k)} = (\tau^{(k)}, \mathbf{a}^{(k)}, \mathbf{c}^{(k)}, \mathbf{dof}^{(k)})$$

where $\tau^{(k)} \in \{\text{Rotation}, \text{Translation}\}$ is the motion type, $\mathbf{a}^{(k)}$ is the axis, $\mathbf{c}^{(k)}$ the center, and $\mathbf{dof}^{(k)} \in \{0,1\}^6$ the active degrees of freedom. These units are used only for evaluation and not provided as input. Figure 2 illustrates a sample two-part assembly, showing how point clouds evolve across frames together with their $\mathrm{SE}(3)$ transformations and twist vectors.

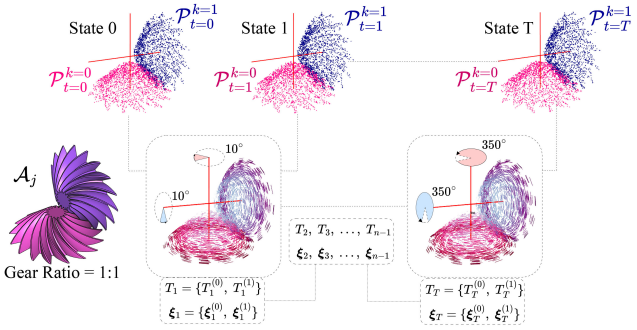

Fig. 2. Illustration of a sample gear assembly $\mathcal{A}_j$ from our dataset, showing two-part meshing motion over time. For each part $k = 0, 1$, we visualize the segmented point cloud $\mathcal{P}_t^{(k)}$ at initial (State 0), intermediate (State 1), and final (State T) frames. Corresponding rigid-body motion is described by SE(3) transformations $T_t^{(k)}$ and twist vectors $\boldsymbol{\xi}_t^{(k)}$ at each time step t. A full motion sequence is composed of T frames, where motion propagates according to gear ratio, and the dataset stores all $\{T_t\}_{t=1}^T$ and $\{\boldsymbol{\xi}_t\}_{t=1}^T$ for precise supervision.

C. Dataset Statistics

MechBench contains 693 assemblies, denoted by $\mathcal{A}$. Each assembly $\mathcal{A}_j$ has M_j parts, where $2 \leq M_j \leq 7$. The distribution is: 187 assemblies with 2 parts, 214 with 3 parts, 127 with 4 parts, 98 with 5 parts, 38 with 6 parts, and 29 with 7 parts. In total, the dataset provides 2445 movable parts. Each part is annotated across 36 frames of motion, corresponding to a complete rotation cycle that can be repeated for extended sequences.

IV. Method

A. Problem Definition

Given a CAD assembly of M parts, the task is to predict the motion of all parts over T frames. Input to the model is segmented part point clouds $\mathcal{P} = \{\mathcal{P}^k\}_{k=1}^M$, where each part k is represented as $\mathcal{P}^k = \{\mathbf{p}_i^k\}_{i=1}^N$ with $\mathbf{p}_i^k \in \mathbb{R}^3$ as input and outputs a sequence of 6D Lie algebra vectors $\{\boldsymbol{\xi}_t^{(k)}\}_{t=1}^T$, where each $\boldsymbol{\xi}_t^{(k)} \in \mathbb{R}^6$ represents the motion of part k at frame t in twist form.

We assume that clean part-level segmentation is available since assemblies are generated directly from CAD models, making this a reasonable assumption. Unlike prior works such as Shape2Motion [21] and RPM-Net [4], which must infer segmentation from raw scans, our setup focuses on learning motion reasoning rather than segmentation. While the CAD models also encode motion axes and centers, these quantities are withheld from the model and used only for evaluation.

To solve this task, we propose **DYNAMO** (Figure 3), a three-stage architecture consisting of (i) part-level point cloud feature extraction, (ii) coupling-aware inter-part relationship modeling, and (iii) temporal motion decoding. The following subsections describe each module in detail.

B. Part Feature Extraction

We extract geometric features for each part using a PointNet++ backbone with hierarchical Set Abstraction (SA) layers. The input tensor is $(P, 3, N)$, where P the number of parts, and N the number of points per part. Each part's point cloud is independently encoded through three SA modules that apply farthest point sampling, neighborhood grouping, and shared MLP layers, followed by a global pooling stage. This hierarchical process captures local-to-global geometric context, and a final SA module produces a global feature vector $\mathbf{f}^{(k)} \in \mathbb{R}^C$ for each part, yielding a feature tensor of shape (P, C) for downstream reasoning.

C. Inter-Part Relationship Modeling

In gear assemblies, motion of one part often directly induces motion in another due to meshing contact. These motion dependencies, referred to as *coupling*, are fundamental to mechanical assemblies but are not always reflected in CAD file hierarchies. While some CAD tools store parent–child trees, these do not necessarily encode true motion propagation paths. Instead, coupling arises from geometric constraints, such as tooth interlocking or shaft alignment, which must be inferred from part geometry.

To capture such relationships, we model each assembly as a fully connected part graph with M nodes, where each node corresponds to a part, and the edges encode mechanical coupling. Let $\mathbf{f}^{(k)} \in \mathbb{R}^C$ denote the feature vector of part k, obtained from the PointNet++ backbone, where $k = 1, \ldots, M$. We define a binary **coupling matrix** $\mathbf{C} \in \{0,1\}^{M \times M}$, where $c_{ij} = 1$ indicates that parts i and j are coupled, and $c_{ij} = 0$ otherwise.

DYNAMO estimates the coupling matrix directly from geometry using a simple contact heuristic. For each pair of parts

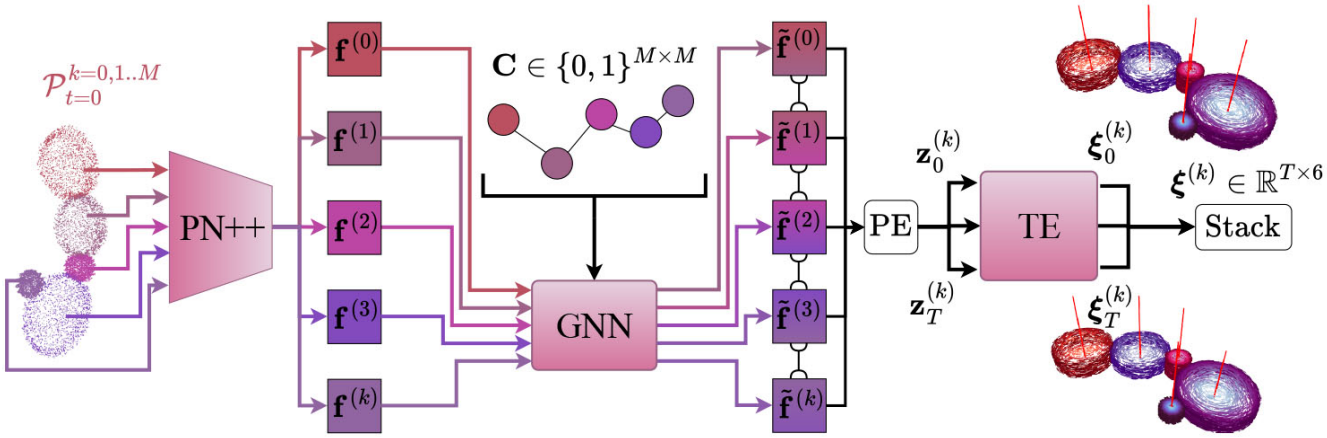

Fig. 3. Overview of the proposed DYNAMO architecture. Given segmented part-wise point clouds $\mathcal{P}_{t=0}^{k=0,\dots,M}$, a PointNet++ backbone extracts per-part features $\mathbf{f}^{(k)}$, which are refined by a coupling-aware GNN based on a binary coupling matrix $\mathbf{C} \in \{0,1\}^{M\times M}$. The final features $\tilde{\mathbf{f}}^{(k)}$ are temporally replicated and added to positional encodings (PE) to form $\mathbf{z}_t^{(k)}$, which are fed into a Transformer decoder to predict twist vectors $\boldsymbol{\xi}_t^{(k)}$ over T frames. The outputs are stacked to form the full motion trajectory $\boldsymbol{\xi}^{(k)} \in \mathbb{R}^{T\times 6}$. **Abbreviations:** PN++ = PointNet++, GNN = Graph Neural Network, PE = Positional Encoding, TE = Temporal Encoder (Transformer-based temporal module) $\mathbf{C}$ = Coupling Matrix.

(i, j), we compute all pairwise distances between their point clouds $\mathcal{P}^i$ and $\mathcal{P}^j$. If enough points lie closer than a distance threshold τ_d, we treat the parts as mechanically coupled; otherwise, they are considered independent. Formally,

$$c_{ij} = \begin{cases} 1, & \text{if } \#\{(m,n) : \|\mathbf{p}_m^i - \mathbf{p}_n^j\|_2 < \tau_d\} \;\geq\; \tau_c, \\ 0, & \text{otherwise,} \end{cases}$$

where τ_c is the minimum number of close contacts required. This process yields a symmetric binary coupling matrix $\mathbf{C} \in \{0,1\}^{M\times M}$ indicating which parts are in contact.

Given this matrix, we apply a coupling-aware graph neural network (GNN) where parts exchange information only along edges marked as coupled. At each layer, part i aggregates messages from its coupled neighbors j to update its representation:

$$\tilde{\mathbf{f}}^{(i)} \leftarrow \mathbf{f}^{(i)} + \sum_{j=1}^{M} c_{ij} \cdot \mathrm{MLP}(\mathbf{f}^{(i)}, \mathbf{f}^{(j)}).$$

Here $\tilde{\mathbf{f}}^{(i)}$ is the updated embedding of part i, enriched with information from its coupled neighbors. This process allows motion information to propagate through the contact graph.

D. Temporal Motion Decoder

The final part features $\tilde{\mathbf{f}}^{(k)} \in \mathbb{R}^C$ produced by the GNN are temporally replicated across T frames and combined with learnable positional encodings to form per-frame tokens. These sequences are processed by a Transformer encoder, which models temporal dependencies across frames and produces an updated sequence of latent vectors. Each latent vector is then passed through a small MLP head to predict a 6D twist vector $\boldsymbol{\xi}_t^{(k)} \in \mathbb{R}^6$, representing the angular and linear velocity of part k at frame t:

$$\boldsymbol{\xi}_t^{(k)} = \mathrm{MLP}\Big(\mathrm{Transformer}\Big(\{\mathbf{z}_\tau^{(k)}\}_{\tau=1}^{T}\Big)_t\Big).$$

where $\mathbf{z}_\tau^{(k)}$ denotes the positional-encoding–augmented token derived from $\tilde{\mathbf{f}}^{(k)}$. The final output ($\mathbb{R}^{P\times T\times 6}$) representing per-part, per-frame twist predictions. This design enables the model to learn smooth, temporally coherent motion trajectories using global self-attention over time.

E. Network Training and Loss Functions

The DYNAMO model is trained to predict physically plausible motion trajectories over time, expressed as sequences of 6D twist vectors $\boldsymbol{\xi}_t^{(k)} \in \mathbb{R}^6$ for each part k at frame t. During training, we supervise the model using ground-truth twist sequences derived from synthetic CAD assemblies, and optimize a multi-term loss function that balances trajectory accuracy and temporal consistency.

The training objective $\mathcal{L}_{\text{total}}$ consists of three loss components: L2 loss for translation, geodesic loss for rotation, and a consistency loss across frames. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{trans}}\mathcal{L}_{\text{trans}} + \lambda_{\text{rot}}\mathcal{L}_{\text{rot}} + \lambda_{\text{const}}\mathcal{L}_{\text{const}} \tag{1}$$

where each λ term is a scalar weight controlling the relative importance of each component.

a) Translation Loss: We use an L2 loss to supervise the predicted linear velocity components $\mathbf{v}_t^{(k)} \in \mathbb{R}^3$ of each part:

$$\mathcal{L}_{\text{trans}} = \frac{1}{M}\sum_{k=1}^{M}\frac{1}{T}\sum_{t=1}^{T}\left\|\mathbf{v}_t^{(k)} - \mathbf{v}_t^{(k,\text{gt})}\right\|_2^2 \tag{2}$$

where M_b is the number of valid (i.e., movable) parts in the b-th assembly and T is the number of motion frames.

b) Rotation Loss: To avoid ambiguities in angular velocity supervision, we adopt a geodesic rotation loss. Given angular velocity vectors $\boldsymbol{\omega}_t^{(k)}$, we form rotation matrices $R_t^{(k)} = \exp(\hat{\boldsymbol{\omega}}_t^{(k)})$ and $R_t^{(k,\text{gt})} = \exp(\hat{\boldsymbol{\omega}}_t^{(k,\text{gt})})$, and compute

the loss as:

$$\mathcal{L}_{\text{rot}} = \frac{1}{M}\sum_{k=1}^{M}\frac{1}{T}\sum_{t=1}^{T}\cos^{-1}\left(\frac{\text{Tr}\left(R_t^{(k)\top}R_t^{(k,\text{gt})}\right)-1}{2}\right) \tag{3}$$

c) LAV Consistency Loss: We encourage temporally smooth predictions by minimizing deviations in velocity and angular velocity over time. For each part k, we define frame-wise differences $\Delta\boldsymbol{\xi}_t^{(k)} = \boldsymbol{\xi}_{t+1}^{(k)} - \boldsymbol{\xi}_t^{(k)}$, and compute the variance-based loss

$$\mathcal{L}_{\text{const}} = \frac{1}{M}\sum_{k=1}^{M}\text{Var}_t\left(\Delta\boldsymbol{\xi}_t^{(k)}\right).$$

This penalizes abrupt changes in part motion and encourages smoother, physically plausible trajectories.

V. Evaluations

A. Experimental Setup

Models are trained on MechBench, with a train/test split of 90/10. We use the AdamW optimizer with an initial learning rate of 10^{-4}, weight decay of 10^{-5}, and cosine annealing schedule with a minimum learning rate of 10^{-6}. The model is trained with a batch size of 4, using gradient clipping at a norm of 1. All experiments are conducted on a single NVIDIA RTX A6000 GPU. All the models are trained with same hyperparameter for comparison. The loss function consists of three components: L2 translation loss, geodesic rotation loss, and a temporal consistency loss. We assign weights $\lambda_{\text{trans}} = 1$, $\lambda_{\text{rot}} = 1$, and $\lambda_{\text{const}} = 0.2$ to each component, which were empirically chosen via validation.

B. Evaluation Metrics

We evaluate predicted part motions using two metrics: *rotation error* and *translation error*. For both, we first transform each part's point cloud $\mathcal{P}^k = \{\mathbf{p}_i^k\}_{i=1}^N$ using the predicted or ground-truth $\text{SE}(3)$ trajectory $\{T_t^{(k)}\}_{t=1}^T$ to obtain $\mathbf{p}_{t,i}^k = T_t^{(k)} \cdot \mathbf{p}_i^k$. This yields frame-wise transformed point clouds $\mathbf{P}_t^{(k)}$ that allow direct comparison between predictions and ground truth.

a) Rotation Error: To quantify angular motion between frames, we compute the rigid rotation between the point clouds at consecutive frames using the Kabsch algorithm [38]. Specifically, let $\mathbf{P}_{t-1}^{(k)}$ and $\mathbf{P}_t^{(k)}$ be the transformed point clouds of part k at frames $t-1$ and t, respectively. We center both point sets and apply the Kabsch algorithm to extract the relative rotation matrix $R_t^{(k)} \in \text{SO}(3)$.

We compute the angular displacement in degrees as:

$$\theta_t^{(k)} = \arccos\left(\frac{\text{trace}(R_t^{(k)})-1}{2}\right)\cdot\left(\frac{180}{\pi}\right)$$

The *rotation error* is the absolute difference in angular displacement between predicted and ground-truth sequences:

$$\text{RotationError}_t^{(k)} = \left|\theta_t^{(k,\text{pred})} - \theta_t^{(k,\text{gt})}\right|$$

b) Translation Error: Translation error is computed as the mean Euclidean distance between the predicted and ground-truth transformed point clouds at each frame:

$$\text{TranslationError}_t^{(k)} = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{p}_{t,i}^{k,\text{pred}} - \mathbf{p}_{t,i}^{k,\text{gt}}\right\|_2$$

We evaluate both metrics per part $k = 1, \ldots, M$ and per frame $t = 1, \ldots, T$. We then report the average rotation and translation error over all movable parts and all frames across the test set. This evaluation framework is compatible with models that predict absolute $\text{SE}(3)$ transforms (e.g., via 6D Lie Algebra vectors $\boldsymbol{\xi}_t^{(k)}$) and those that predict frame-wise point displacements (e.g., RPM-Net).

C. Baselines

There are no existing methods designed for dependency-aware motion prediction in CAD assemblies. To provide fair comparisons, we introduce two simple yet strong baselines: **RigidNet-LSTM** and **RigidNet-Transformer**. Both take segmented part point clouds as input, extract features with PointNet++ (PN++), and use either an LSTM or Transformer temporal head to predict per-part 6D Lie algebra vectors over all frames. RigidNet-LSTM is inspired by RPM-Net's architecture, but instead of predicting dense displacement flows, it directly outputs rigid-body motions per part, making it a natural recurrent baseline. In contrast, RigidNet-Transformer leverages self-attention to capture long-range temporal dependencies, offering a stronger alternative to recurrent models. Unlike DYNAMO, these models have no coupling supervision, allowing us to isolate the effect of temporal modelling alone. They are trained with the same loss functions as DYNAMO (Section IV-E) for direct comparison.

We also compare with **RPM-Net** [4], which remains the only prior method capable of predicting multi-part motion trajectories from 3D geometry, for multiple frames of motion. Although RPM-Net jointly infers segmentation and motion, we disable its segmentation branch and use ground-truth part labels to focus on motion quality. Other existing methods target different settings (e.g., single-part mobility prediction, joint inference), making RPM-Net the most relevant baseline for our task.

D. Quantitative Results

We report rotation and translation errors for all baselines and DYNAMO. Table II summarizes the overall mean errors across the entire test set. DYNAMO achieves the lowest errors in both metrics, reducing rotation error by more than 70% compared to RPM-Net and consistently outperforming the simple rigid baselines.

We also break down performance by number of parts in the assembly (Table II). While all methods show increased difficulty for assemblies with more parts, DYNAMO maintains the lowest error across all settings, especially for challenging 5–6 part assemblies. The degradation for 6-part assemblies is likely due to increased graph complexity and longer dependency chains, where motion errors propagate

TABLE II

ROTATION AND TRANSLATION ERRORS ON **MECHBENCH**. OVERALL MEAN ERRORS AND BREAKDOWN BY NUMBER OF PARTS PER ASSEMBLY.

	Overall		2 Parts		3 Parts		4 Parts		5 Parts		6 Parts		7 Parts	
Method	Rot	Trans	Rot	Trans	Rot	Trans	Rot	Trans	Rot	Trans	Rot	Trans	Rot	Trans
RPM-Net [4]	15.96	0.46	10.52	0.64	13.63	0.53	17.79	0.33	25.99	0.33	7.14	**0.18**	3.81	**0.17**
RigidNet-LSTM	9.64	0.64	6.04	0.69	9.09	0.66	9.29	0.58	16.08	0.65	**5.04**	0.64	3.76	0.41
RigidNet-Transformer	5.90	0.18	1.41	0.11	5.18	0.19	7.44	0.20	9.11	0.22	10.28	0.28	16.72	0.27
DYNAMO (Ours)	**3.28**	**0.14**	**0.49**	**0.06**	**2.62**	**0.15**	**2.76**	**0.15**	**7.73**	**0.19**	5.79	0.23	**2.64**	0.16

across multiple coupled components. As assemblies grow, multi-hop coupling reasoning becomes more challenging.

a) Frame-wise stability.: For rotation, errors remained stable across the full 35 frames with negligible fluctuations. The standard deviation (SD) of frame-wise rotation errors was $0.007°$ for DYNAMO, $0.006°$ for RPM-Net, $0.005°$ for RigidNet-LSTM, and $0.062°$ for RigidNet-Transformer, confirming that the averages reported in Table II are consistent throughout the sequence.

For translation, small temporal variations are observed, DYNAMO achieved the lowest SD (0.058), followed by RigidNet-Transformer (0.051), RPM-Net (0.099), and RigidNet-LSTM (0.175). These trends reinforce that DYNAMO not only yields the lowest overall translation errors but also maintains stable performance across time, while the other baselines exhibit larger fluctuations that compound their higher mean errors.

E. Qualitative Results

Figure 4 shows that **DYNAMO** accurately predicts both rotational and translational motions across diverse assemblies. It captures synchronized rotation in coupled planar gears, infers linear translation in rack-and-pinion systems, and transfers motion across orthogonal or misaligned axes in screw and bevel configurations. In all cases, DYNAMO maintains tight alignment with the ground truth and produces smooth, consistent displacement trajectories, demonstrating robustness even in geometrically complex or symmetric setups where simpler models often fail.

The predictions preserve contact constraints and rotational directionality (clockwise vs. counterclockwise), even when the axes of rotation are orthogonal or misaligned, as in bevel and screw gears. Despite no supervision on gear ratios or explicit axis labels, DYNAMO learns to internalize speed and coupling constraints from geometry alone. This includes understanding how the size of interacting gears affects their relative motion: when a small gear meshes with a larger one, the smaller gear must rotate faster to maintain contact, this mechanical relationship is known as the gear ratio. For example, if one gear is twice the diameter of its neighbor, the smaller gear will rotate twice as fast. DYNAMO appears to infer such relationships directly from the part geometries, learning that larger differences in gear sizes imply faster relative speeds, whereas similar-sized gears (e.g., 1:1 ratio) rotate at similar speed. This emergent behavior indicates the model's capacity to implicitly capture physical constraints purely from static 3D geometry.

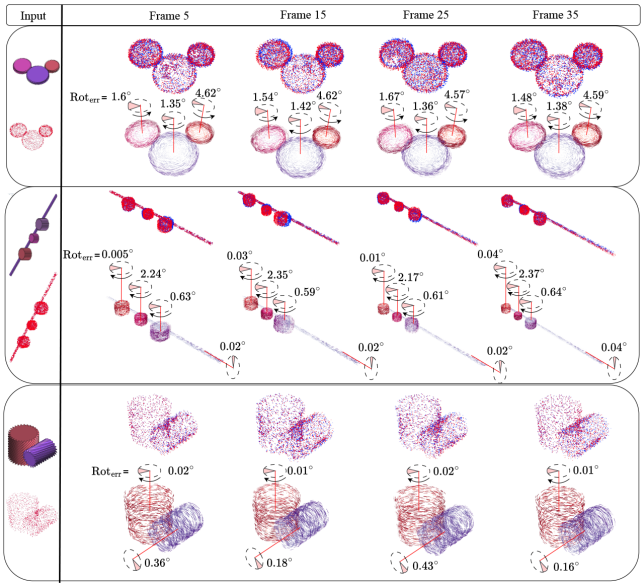

Fig. 4. Qualitative motion prediction results across different CAD assemblies. Each pair of rows shows a test assembly with its input (top-left: reference mesh + segmented input point cloud), followed by predicted motion at Frames 5, 15, 25, and 35. In each frame, **blue** denotes ground-truth point cloud, and **red** denotes the predicted point cloud overlaid. The bottom row for each frame shows the **point-wise displacement vectors** from the previous frame (e.g., Frame $t-1 \rightarrow t$), visualizing local motion trends. Annotations indicate the rotation direction of parts and report the exact angular error at each frame, highlighting prediction accuracy over time. Assemblies include spur, rack-and-pinion, and screw gears, demonstrating DYNAMO's generalization to diverse, tightly coupled kinematic structures.

Overall, these results show that DYNAMO not only predicts physically plausible part-wise motions but also captures coupling-induced behaviors in complex, multi-part CAD assemblies across 2 to 7 parts.

F. Failure Cases

Although **DYNAMO** performs robustly across diverse assemblies, it can fail when coupling constraints are not preserved. This typically occurs because the model learns motion patterns only from geometry, without explicit physical priors, and may therefore decouple translation from rotation or allow parts to drift, leading to unphysical behaviors such as overlaps or disconnected motion. Such errors remain uncommon: across the 70 held-out assemblies, only one

exceeded a $40°$ rotation error and just two crossed $10°$, with the vast majority below $5°$.

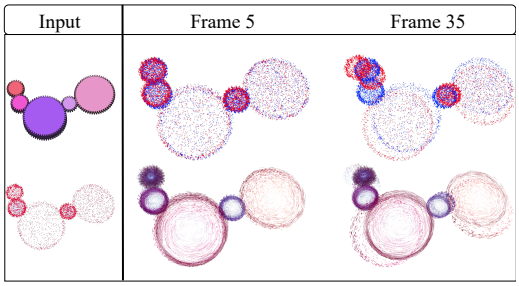

Fig. 5. Edge case example with the highest observed rotation error ($49.67°$). Red = predicted point cloud; Blue = ground truth. Bottom row shows displacement vectors (from previous frame) to visualize per-point motion direction.

Figure 5 illustrates the most severe failure. In this five-gear assembly, the prediction diverges from the ground truth, producing merged and drifting parts and a gear that translates without rotation. This example highlights the model's limitations in preserving strict coupling, suggesting future improvements such as incorporating contact-aware priors or explicit constraint enforcement during inference.

G. Ablation Studies

We conduct a series of ablation studies to assess the impact of key architectural and training choices in DYNAMO. Our goal is to quantify the contribution of different modules and design decisions toward accurate and stable motion prediction.

a) Model Architecture: Point Feature Extractor and Temporal Module.: We vary the point cloud encoder (PointNet++ [39] vs. Point Transformer [40]) and temporal head (LSTM vs. Transformer) while keeping the coupling-aware GNN fixed (Table III). DYNAMO (PN++ + Transformer) achieves the best rotation accuracy ($3.32°$) while remaining competitive in translation. Point Transformer with a Transformer head yields the lowest translation error (0.136), reflecting its strength in capturing local geometric detail, but performs worse in rotation, where PN++'s hierarchical abstraction better captures global context. LSTM-based variants consistently degrade performance, indicating that recurrent models struggle with long-horizon consistency compared to temporal self-attention. Overall, PN++ provides robust global features for rotation while the Transformer decoder ensures stable temporal trajectories. RigidNet-Transformer in Table II uses the same PN++ encoder and Transformer decoder but without coupling reasoning; the large gap between it and DYNAMO highlights the importance of coupling-aware modeling.

b) Effect of Point Cloud Density.: We evaluate DYNAMO under varying levels of point cloud sparsity by randomly subsampling each part to 128, 256, 512, or 1024 points. As shown in Table IV, increasing point density improves both rotation and translation accuracy, with performance stabilizing around 512–1024 points. Notably, even at low resolutions (e.g., 256 points), DYNAMO achieves lower errors than all baselines tested in Section V-C (Table II), demonstrating strong robustness to incomplete or partial point clouds. This makes our model suitable for real-world use cases where noisy or partial observations are common, such as depth sensing, occlusion-prone environments, or imperfect CAD imports.

TABLE III
ABLATION ON POINT FEATURE EXTRACTOR AND TEMPORAL MODULE.

Architecture Variant	Rotation (°)	Translation
Point Transformer + LSTM	4.3014	0.1597
Point Transformer + Transformer	3.4063	**0.1357**
PointNet++ + LSTM	3.3761	0.1897
PointNet++ + Transformer	**3.2822**	0.1427

TABLE IV
ABLATION ON NUMBER OF POINTS PER PART POINT CLOUD.

Points per Part	Rotation (°)	Translation
128	7.1212	0.4371
256	3.7709	0.2076
512	**3.1742**	0.1474
1024	3.2822	**0.1427**

c) Effect of Loss Terms.: We ablate each loss component from the training objective to assess its individual contribution. Removing the rotation loss leads to a dramatic degradation in both metrics, especially rotation ($12.89°$), confirming its critical role in orientational accuracy. Without translation loss, the model still learns rotational trends but suffers a large increase in translation error. Lastly, removing the temporal consistency loss results in moderate error increases, and more importantly, introduces temporal instability. The standard deviation (SD) of frame-wise translation error rises from 0.058 (full model) to 0.080, indicating less smooth trajectories over time.

TABLE V
ABLATION ON LOSS COMPONENTS. REMOVING ANY TERM DEGRADES PERFORMANCE.

Loss Configuration	Rotation (°)	Translation
w/o Translation Loss	3.7301	0.6043
w/o Rotation Loss	12.8936	2.5456
w/o Temporal Consistency Loss	3.4711	0.1586
All Losses (Ours)	**3.2822**	**0.1427**

VI. CONCLUSION AND FUTURE WORK

We presented **DYNAMO**, a dependency-aware neural model for predicting per-part rigid-body motion in CAD assemblies, together with **MechBench**, a benchmark of 693 synthetic gear assemblies annotated with ground-truth trajectories. Our results show that DYNAMO learns to infer coupled motions directly from static geometry, consistently outperforming strong baselines across multiple metrics.

Despite these advances, DYNAMO operates in a controlled synthetic CAD setting and does not model real-world perception noise, contact slip, or axis misalignment, and may produce implausible behaviors such as drifting or interpenetration. Future work will incorporate contact-aware physical constraints and extend evaluation to noisier or real-world assemblies, and expanding MechBench with additional mechanisms (e.g., linkages, cam–follower systems) to support broader and more generalizable motion prediction.

REFERENCES

[1] A. Gupta, M. Zhang, R. Sathua, and S. Gupta, "Opening articulated structures in the real world," *arXiv preprint arXiv:2402.17767*, 2024.

[2] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, M. Agrawala *et al.*, "Illustrating how mechanical assemblies work," *ACM Transactions on Graphics-TOG*, vol. 29, no. 4, p. 58, 2010.

[3] J. Chuyun and L. Mu, "Application of virtual assembly for complex mechanical structures based on digital twin technology," *Scientific Reports*, vol. 15, no. 1, p. 30306, 2025.

[4] Z. Yan, R. Hu, X. Yan, L. Chen, O. Van Kaick, H. Zhang, and H. Huang, "Rpm-net: recurrent prediction of motion and parts from point cloud," *arXiv preprint arXiv:2006.14865*, 2020.

[5] L. Fu, R. Ishikawa, Y. Sato, and T. Oishi, "Capt: Category-level articulation estimation from a single point cloud using transformer," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 751–757.

[6] L. Liu, J. Du, H. Wu, X. Yang, Z. Liu, R. Hong, and M. Wang, "Category-level articulated object 9d pose estimation via reinforcement learning," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 728–736.

[7] H. Li, G. Wan, H. Li, A. Sharf, K. Xu, and B. Chen, "Mobility fitting using 4d ransac," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 79–88.

[8] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to generalize kinematic models to novel objects," in *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[9] R. Wang, A. Gadi Patil, F. Yu, and H. Zhang, "Active coarse-to-fine segmentation of moveable parts from real images," in *European Conference on Computer Vision*. Springer, 2024, pp. 111–127.

[10] A. Sharf, H. Huang, C. Liang, J. Zhang, B. Chen, and M. Gong, "Mobility-trees for indoor scenes manipulation," in *Computer Graphics Forum*, vol. 33, no. 1. Wiley Online Library, 2014, pp. 2–14.

[11] X. Qiu, J. Yang, Y. Wang, Z. Chen, Y. Wang, T.-H. Wang, Z. Xian, and C. Gan, "Articulate anymesh: Open-vocabulary 3d articulated objects modeling," *arXiv preprint arXiv:2502.02590*, 2025.

[12] R. Hu, W. Li, O. Van Kaick, A. Shamir, H. Zhang, and H. Huang, "Learning to predict part mobility from a single static snapshot," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 6, pp. 1–13, 2017.

[13] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3706–3715.

[14] G. Liu, Q. Sun, H. Huang, C. Ma, Y. Guo, L. Yi, H. Huang, and R. Hu, "Semi-weakly supervised object kinematic motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 726–21 735.

[15] X. Liu, J. Zhang, R. Hu, H. Huang, H. Wang, and L. Yi, "Self-supervised category-level articulated object pose estimation with part-level se (3) equivariance," *arXiv preprint arXiv:2302.14268*, 2023.

[16] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas, "Deep part induction from articulated object pairs," *arXiv preprint arXiv:1809.07417*, 2018.

[17] S. Liu, S. Gupta, and S. Wang, "Building rearticulable models for arbitrary 3d objects from 4d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 138–21 147.

[18] R. Li, C. Zheng, C. Rupprecht, and A. Vedaldi, "Dragapart: Learning a part-level motion prior for articulated objects," in *European Conference on Computer Vision*. Springer, 2024, pp. 165–183.

[19] J. Liu, D. Iliash, A. X. Chang, M. Savva, and A. Mahdavi-Amiri, "Singapo: Single image controlled generation of articulated parts in objects," *arXiv preprint arXiv:2410.16499*, 2024.

[20] J. Liu, H. I. I. Tam, A. Mahdavi-Amiri, and M. Savva, "Cage: Controllable articulation generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 880–17 889.

[21] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2motion: Joint analysis of motion parts and attributes from 3d shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8876–8884.

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

[23] J. Liu, A. Mahdavi-Amiri, and M. Savva, "Paris: Part-level reconstruction and motion analysis for articulated objects," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 352–363.

[24] Y. Mao, Y. Zhang, H. Jiang, A. Chang, and M. Savva, "Multiscan: Scalable rgbd scanning for 3d environments with articulated objects," *Advances in neural information processing systems*, vol. 35, pp. 9058–9071, 2022.

[25] J. Liu, M. Savva, and A. Mahdavi-Amiri, "Survey on modeling of human-made articulated objects," in *Computer Graphics Forum*. Wiley Online Library, 2025, p. e70092.

[26] Y. Shi, X. Cao, and B. Zhou, "Self-supervised learning of part mobility from point cloud sequence," in *Computer Graphics Forum*, vol. 40, no. 6. Wiley Online Library, 2021, pp. 104–116.

[27] S. Qian, L. Jin, C. Rockwell, S. Chen, and D. F. Fouhey, "Understanding 3d object articulation in internet videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1599–1609.

[28] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 097–11 107.

[29] X. Sun, H. Jiang, M. Savva, and A. Chang, "Opdmulti: Openable part detection for multiple objects," in *2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 169–178.

[30] D. Iliash, H. Jiang, Y. Zhang, M. Savva, and A. X. Chang, "S2o: Static to openable enhancement for articulated 3d objects," *arXiv preprint arXiv:2409.18896*, 2024.

[31] R. Martín-Martín, C. Eppner, and O. Brock, "The rbo dataset of articulated objects and interactions," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1013–1019, 2019.

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

[33] E. Coumans, "Bullet physics simulation," in *ACM SIGGRAPH 2015 Courses*, 2015, p. 1.

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

[35] L. Liu, H. Xue, W. Xu, H. Fu, and C. Lu, "Toward real-world category-level articulation pose estimation," *IEEE Transactions on Image Processing*, vol. 31, pp. 1072–1083, 2022.

[36] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.

[37] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas, "Captra: Category-level pose tracking for rigid and articulated objects from point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 209–13 218.

[38] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Foundations of Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[40] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 259–16 268.