Pose-Transformed Equivariant Network for 3D Point Trajectory Prediction

Supplementary Material

A. Proof of Theorem 1

Theorem 1. *With τ as Unique Pose-Normalization by PCA and F as Point Position Predictor, $\Psi(\mathcal{P})$ is equivariant under SE(3) transformation of $\mathcal{P}$.*

To prove this theorem, the notations are firstly introduced, based on which we present Lemma 1 on the uniqueness of our proposed Unique Pose-Normalization τ. We then present Lemma 2 to prove the invariance of $\tau(\mathcal{P})$. Finally we prove the equivariance of $\Psi(\mathcal{P})$.

Notations. Given point cloud $\mathcal{P}$ with point position $X_P \in \mathcal{R}^{N\times 3}$, velocity $V_P \in \mathcal{R}^{N\times 3}$, attribute $H \in \mathcal{R}^{N\times d}$ and edge attribute E, we only need to prove that $\Psi(\mathcal{P})$ is equivariant under $SE(3)$ transformation of X_P, V_P. The point attribute H and edge attribute E are both invariant under $SE(3)$ transformation of $\mathcal{P}$. Let $\mathcal{Q} = g(\mathcal{P}),\ \forall g \in SE(3)$, and $\mathcal{Q}$ is a transformed point cloud from $\mathcal{P}$ with its point position and velocity as $X_Q = X_P\overline{R} + \overline{T}, V_Q = V_P\overline{R}$, where $\overline{R} \in \mathcal{R}^{3\times 3}$ and $\overline{T} \in \mathcal{R}^{3\times 1}$ are rotation matrix and translation vector respectively.

Lemma 1. *The singular vector matrix computed by Unique Pose-Normalization τ is uniquely determined with only one direction for every singular vector.*

Proof. The singular vector matrix U is composed with three singular vectors $\{u_1, u_2, u_3\}$ corresponding to the largest three singular values sorted in descending order. Both u_i and $-u_i$ can be taken as singular vectors. In our unique pose-normalization τ, we determine the singular vector directions, with the uniqueness proven as follows.

We estimate the angle between $u_i,\ i = 1, 2, 3$ with a predefined anchor point y (e.g., the farthest point from centroid). The direction of u_i should be flipped if the corresponding angle is larger than $90°$. That is, we select the singular vector u_i with the inner product $\langle u_i, y\rangle \geq 0$. If $\langle u_i, y\rangle > 0$, we take u_i as the singular vector. If $\langle u_i, y\rangle < 0$, we take $-u_i$ as the singular vector which satisfies $\langle -u_i, y\rangle > 0$. If $\langle u_i, y\rangle = 0$, both u_i and $-u_i$ satisfy the condition $\langle u_i, y\rangle \geq 0$, which means they can both be taken as the singular vectors. In order to prevent this uncertainty, we take another point y' (e.g., the second farthest point from centroid) that satisfies $\langle u_i, y'\rangle \neq 0$ as new anchor point to determine the singular vector direction.

By this strategy, the directions of all the three singular vectors are uniquely determined, and the singular vector matrix U is uniquely determined. □

Lemma 2. *The uniquely pose-normalized point cloud $\tau(\mathcal{P})$ is invariant under $SE(3)$ transformation of $\mathcal{P}$.*

Proof. We prove this lemma by proving $\tau_P(\mathcal{P}) = \tau_Q(\mathcal{Q})$ in three steps, with τ_P, τ_Q as unique pose-normalization of $\mathcal{P}, \mathcal{Q}$ respectively.

Step1. Denote the center points of X_P, X_Q as μ, ν respectively, we prove the relationship between μ and ν in this step, i.e., $\nu = \mu\overline{R} + \overline{T}$. The center points μ and ν are respectively computed by

$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_i, \quad \nu = \frac{1}{N}\sum_{i=1}^{N} q_i, \tag{1}$$

with p_i, q_i as elements of X_P, X_Q. Recalling that $X_Q = X_P\overline{R} + \overline{T}$, $q_i = p_i\overline{R} + \overline{T}$, we have

$$\begin{aligned}\nu &= \frac{1}{N}\sum_{i=1}^{N} q_i \\ &= \frac{1}{N}\sum_{i=1}^{N} (p_i\overline{R} + \overline{T}) \\ &= (\frac{1}{N}\sum_{i=1}^{N} p_i)\overline{R} + \overline{T} \\ &= \mu\overline{R} + \overline{T}.\end{aligned} \tag{2}$$

Step2. In this step, we prove the relationship between singular vector matrices of centralized point clouds, i.e., $U_Q = \overline{R}^\top U_P$. Denote the singular vector decomposition utilized in our unique pose-normalization as

$$\begin{aligned}U_P\Lambda_P U_P^\top &= \text{SVD}[(X_P - \mu)^\top (X_P - \mu)], \\ U_Q\Lambda_Q U_Q^\top &= \text{SVD}[(X_Q - \nu)^\top (X_Q - \nu)].\end{aligned} \tag{3}$$

By Step 1 we have $\nu = \mu\overline{R} + \overline{T}$, and it is obvious

$$\begin{aligned}X_Q - \nu &= X_P\overline{R} + \overline{T} - \nu \\ &= X_P\overline{R} + \overline{T} - (\mu\overline{R} + \overline{T}) \\ &= X_P\overline{R} - \mu\overline{R} \\ &= (X_P - \mu)\overline{R}.\end{aligned} \tag{4}$$

Note that $\overline{R}$ is a rotation matrix, and Eqn. (4) means that $X_Q - \nu$ is an orthogonal transformation of $X_P - \mu$. Considering that Λ_P and Λ_Q are singular values computed by Eqn. (3), and orthogonal transformation do not change the

singular values, we have $\Lambda_P = \Lambda_Q$. On the other hand,

$$\begin{aligned} U_Q \Lambda_Q U_Q^\top &= (X_Q - \nu)^\top (X_Q - \nu) \\ &= [(X_P - \mu)\overline{R}]^\top [(X_P - \mu)\overline{R}] \\ &= \overline{R}^\top (X_P - \mu)^\top (X_P - \mu)\overline{R} \\ &= \overline{R}^\top U_P \Lambda_P U_P^\top \overline{R} \\ &= (\overline{R}^\top U_P)\Lambda_Q (U_P^\top \overline{R}), \end{aligned} \tag{5}$$

which means $U_Q \Lambda_Q U_Q^\top = (\overline{R}^\top U_P)\Lambda_Q (U_P^\top \overline{R})$ holds for any rotation matrix $\overline{R}$. By Lemma 1, U_P and U_Q are uniquely determined, so we have $U_Q = \overline{R}^\top U_P$.

Step3. We finally prove the invariance of point position and velocity for the uniquely pose-normalized point cloud in this step. Recalling that the normalized point positions of $\mathcal{P}, \mathcal{Q}$ are respectively computed by $(X_P - \mu)U_P$, $(X_Q - \nu)U_Q$, and with the conclusion of Step 1-2 we have

$$\begin{aligned} (X_Q - \nu)U_Q =& \{X_P \overline{R} + \overline{T} - (\mu\overline{R} + \overline{T})\}\overline{R}^\top U_P \\ =& (X_P - \mu)U_P. \end{aligned} \tag{6}$$

Above equation proves the invariance of point position for the uniquely pose-normalized point cloud. On the other hand, the velocities of uniquely pose-normalized point clouds for $\mathcal{P}, \mathcal{Q}$ are respectively computed as $V_P U_P, V_Q U_Q$, and the invariance is derived by

$$V_Q U_Q = V_Q \overline{R}^\top U_P = V_P U_P. \tag{7}$$

By Eqns. (6-7), we derive the invariance of point position and velocity for the uniquely pose-normalized point cloud. With the invariant point attribute and edge attribute, we finally arrive at $\tau_P(\mathcal{P}) = \tau_Q(\mathcal{Q})$, i.e., the uniquely pose-normalized point cloud $\tau(\mathcal{P})$ is invariant under $SE(3)$ transformation of point cloud $\mathcal{P}$. □

Proof of Theorem 1. The equivariance of $\Psi(\mathcal{P})$ is proven as follows.

Proof. Denote the predicted point position, velocity of $F \circ \tau_P(\mathcal{P})$ / $\Psi(\mathcal{P})$ as $\{X'_P, V'_P\}$ / $\{X^*_P, V^*_P\}$ respectively, denote the predicted point position, velocity of $F \circ \tau_Q(\mathcal{Q})$ / $\Psi(\mathcal{Q})$ as $\{X'_Q, V'_Q\}$ / $\{X^*_Q, V^*_Q\}$ respectively. According to the definition of Ψ, τ_P^{-1} and τ_Q^{-1}, we have

$$\begin{aligned} X^*_P &= X'_P U_P^\top + \mu, \quad V^*_P = V'_P U_P^\top, \\ X^*_Q &= X'_Q U_Q^\top + \nu, \quad V^*_Q = V'_Q U_Q^\top. \end{aligned} \tag{8}$$

On the other hand, by Lemma 2 we have $\tau_P(\mathcal{P}) = \tau_Q(\mathcal{Q})$ and $F \circ \tau_P(\mathcal{P}) = F \circ \tau_Q(\mathcal{Q})$, which means

$$X'_P = X'_Q, \qquad V'_P = V'_Q. \tag{9}$$

According to Steps 1-2 of Lemma 2, $\nu = \mu\overline{R} + \overline{T}$, $U_Q = \overline{R}^\top U_P$, therefore

$$\begin{aligned} X^*_Q &= X'_Q U_Q^\top + \nu \\ &= X'_P (\overline{R}^\top U_P)^\top + \mu\overline{R} + \overline{T} \\ &= X'_P U_P^\top \overline{R} + \mu\overline{R} + \overline{T} \\ &= (X'_P U_P^\top + \mu)\overline{R} + \overline{T} \\ &= X^*_P \overline{R} + \overline{T}, \\ V^*_Q &= V'_Q U_Q^\top \\ &= V'_P (\overline{R}^\top U_P)^\top \\ &= V'_P U_P^\top \overline{R} \\ &= V^*_P \overline{R}. \end{aligned} \tag{10}$$

With above equations, we have $X^*_Q = X^*_P \overline{R} + \overline{T}, V^*_Q = V^*_P \overline{R}$, which means the equivariance of $\Psi(\mathcal{P})$ for $\mathcal{P}$ under $SE(3)$ transformation. □

B. Algorithm of Global-Local Motion Estimation Layer (Φ)

The operations in Global-Local Motion Estimation layer (Φ) are summarized in Algorithm 1 as follows.

Algorithm 1: Global-Local Motion Estimation

Input: Point cloud with point position X, velocity V, attribute H, and edge attribute E.
Output: Point position X' and attribute H'.

Decompose global/local components by Eqn.(10) and learn global/local attribute by Eqn.(11).
for $i \leftarrow 1$ *to* N_S **do**
  Update global displacement Δx^g by Eqn.(12).
  Update local displacement Δx_i^l by Eqn.(13).
  Update global/local attribute h^g / h_i^l by Eqn.(14).
end
Compose position X' and attribute H' by Eqn.(15).

C. Details on Network and Training

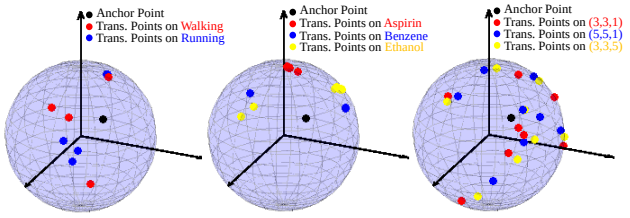
Details on network architecture. We design PT-EvNet with three pose-transformed points prediction blocks f_Γ in the point position predictor F. In each block, we set $N_K = 4$ pose transformations to transform the pose-normalized point cloud. In the global-local motion estimation layer Φ, we take $N_S = 4$ iterations to estimate global and local displacements. In the global-local motion estimation layer Φ, the functions of $\kappa, \beta, \zeta, \eta$ are all set as two-layer MLPs with output as three-dimensional vector, and ξ, ρ, γ are set as two-layer MLPs to learn 64-dimensional attribute. All the MLPs take 64-dimensional hidden neurons with SiLU as non-liner activation function. For PT-EvNet on CMU dataset, the MLPs in the Updating step including

$\kappa, \beta, \zeta, \eta, \xi, \gamma$ all take batch-normalization (BN) layer before the SiLU layer, while for networks on the MD17 and Simulation dataset, there is no BN utilized.

Details on network training. PT-EvNet is trained by Adam optimizer with Mean Squared Error (MSE) loss between the predicted point position and ground truth point position, and we set the batch-size to 12. On CMU and MD17 datasets, the learning rate and weight decay are set as $4e^{-4}, 1e^{-12}$ respectively. On the Simulation dataset, we set them as $4e^{-4}, 1e^{-4}$. All the networks are trained with early stopping of 50 epochs. We utilize the same codes as [3][1], [4][2], [7][3] for data pre-processing and data loading, and for the MD17 dataset we ignore particles with charges less than 1 as in GMN [4]. The velocity is calculated as the difference between adjacent frames. For the data that contains one trajectory sequence for one human / molecules / particle, the unique pose-normalization is computed on the first frame of the trajectory, and applied to all the frames in the trajectory.

D. Visualization of Learned Transformations

In this section, we visualize the learned transformations in PR-EvNet. Given one point (in black) with its position as $[1, 1, 1]$, we utilize the learned rotation in $\Gamma = \{\tau_k | k = 1, ..., 8\}$ in PT-EvNet to transform it, and display the transformed points (in different colors) in Figure 1. Subfigures (a-c) show the points with transformations learned on CMU, MD17 and Simulation dataset respectively, and these transformed points have different distributions. In every subfigure, the transformed points are highlighted with different colors for transformations learned on different subsets, which also differ across these subsets. There is more significant variation in the learned transformations on subsets of Simulation dataset.



(a) CMU (b) MD17 (c) Simulation

Figure 1. Illustration of learned pose transformations for CMU, MD17 and Simulation datasets. For the given anchor (black) point, the learned transformations transform it to different points across different dataset.

[1]https://github.com/hanjq17/EGHN
[2]https://github.com/hanjq17/GMN
[3]https://github.com/vgsatorras/egnn

E. Visualization of Points Prediction Progress

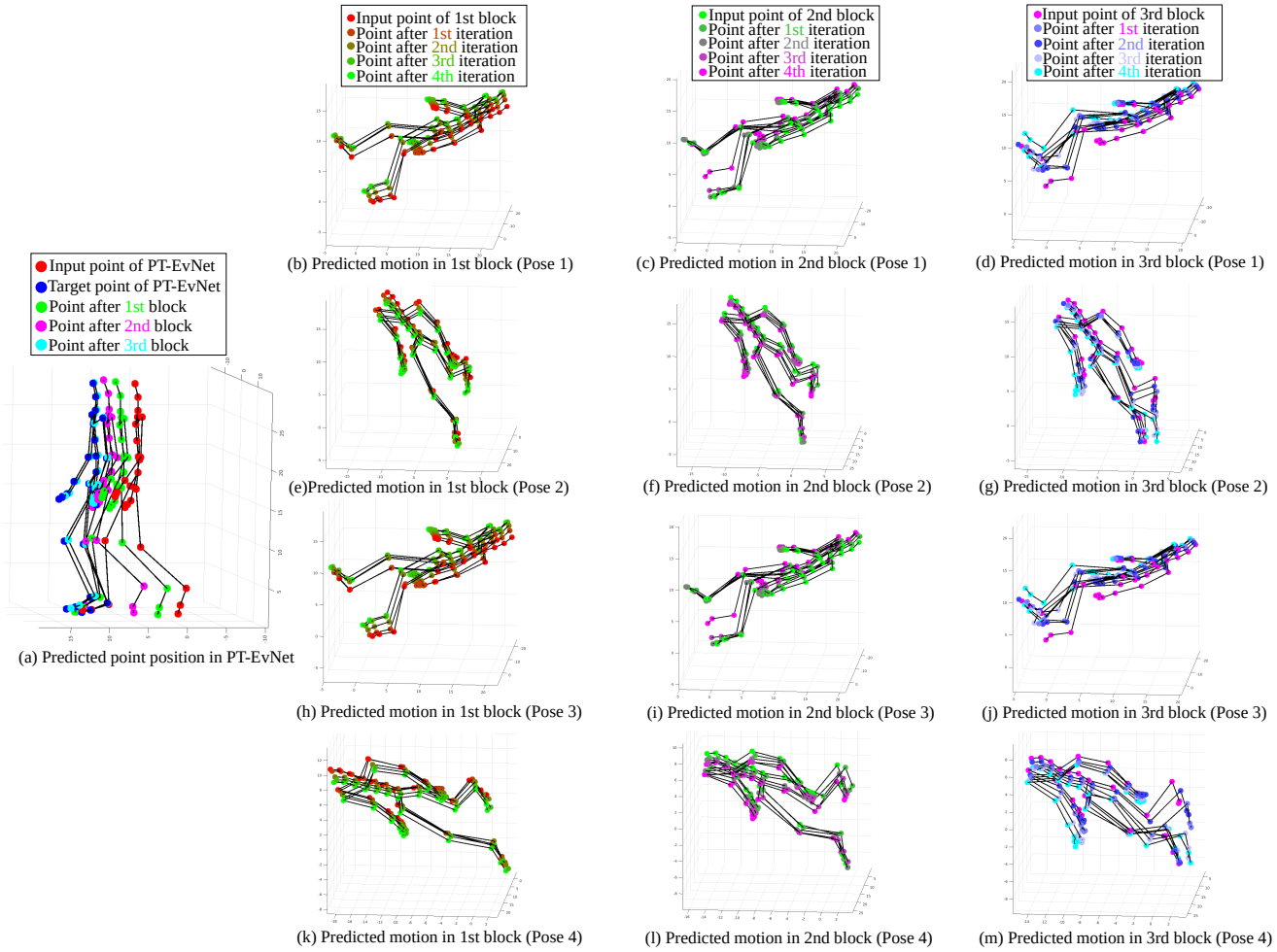
In this section, we first visualize the points prediction progress in our PT-EvNet on Walking dataset. In Figure 2 (a), we show the predicted points (in green/magenta /cyan) after three Pose-Transformed Points Prediction Blocks f_Γ, together with the input points (in red) and target points (ground truth, in blue). As illustrated in this subfigure, the predicted points within PT-EvNet move to the target ones step by step, with every block predicts a step of point motion. In every block of our PT-EvNet, the point motion is predicted from multiple pose-transformed point clouds, whose point-wise motions are estimated iteratively by Global-Local Motion Estimation Layer Φ. In subfigures (b-d), we present the point displacements in Φ for the first pose-transformed point cloud. As shown in these subfigures, the points move forward gradually, which agrees to our iterative displacement updating strategy in Φ. In subfigures (e-g), (h-j), (k-m), we display the predicted motion in Φ for other pose-transformed point clouds, and the points gradually move forward. We further present the points prediction progress in Figure 3 for the Benzene dataset, which exhibits similar phenomenons.

F. Details of Network Architectures in Table 5

Table 5 of the paper presents the results of modified PT-EvNet that replaces the Global-Local Motion Estimation Layer Φ with other motion prediction operations. These operations include Linear layer, MLP, and layers utilized in SOTA models, e.g., MPNN[2], EGNN[7], GMN[4] and EGHN[3]. The modified networks are respectively named as PT-EvNet-linear, PT-EvNet-mlp, PT-EvNet-mpnn, PT-EvNet-egnn, PT-EvNet-gmn, PT-EvNet-eghn.

For PT-EvNet-linear, we set the motion prediction layer as fully connected layer. Specifically, for pose-transformed point cloud, we concatenate the point position, attribute, velocity as input, and feed it to a fully connected layer for the attribute updating. Then the output is sent to SiLU and a fully connected layer for point displacement prediction. The displacement are added to the point position as estimation of the point positions. For PT-EvNet-mlp, we take the same architecture except that we replace the fully connected layers with two-layer MLPs. These MLPs have 64-dimensional hidden neurons and take SiLU as non-liner activation function.

For PT-EvNet-mpnn, PT-EvNet-egnn, PT-EvNet-gmn and PT-EvNet-eghn, we utilize the basic motion prediction layers (from codes of [3],[4],[7]) to estimate point motion for the pose-transformed point cloud. Specifically, we first utilize a two-layer MLP to learn point attribute, taking the concatenation of point position, attribute, velocity as input. This MLP has 64-dimensional hidden neurons and takes SiLU as non-liner activation function. Then we feed the



(a) Predicted point position in PT-EvNet
(b) Predicted motion in 1st block (Pose 1)
(c) Predicted motion in 2nd block (Pose 1)
(d) Predicted motion in 3rd block (Pose 1)
(e)Predicted motion in 1st block (Pose 2)
(f) Predicted motion in 2nd block (Pose 2)
(g) Predicted motion in 3rd block (Pose 2)
(h) Predicted motion in 1st block (Pose 3)
(i) Predicted motion in 2nd block (Pose 3)
(j) Predicted motion in 3rd block (Pose 3)
(k) Predicted motion in 1st block (Pose 4)
(l) Predicted motion in 2nd block (Pose 4)
(m) Predicted motion in 3rd block (Pose 4)

Figure 2. Visualization of points prediction progress in PT-EvNet on Walking dataset. Subfigure (a) displays the predicted points by the three pose-transformed points prediction blocks f_Γ (in green/magenta/cyan), as well as the input points (in red) and target points (ground truth, in blue). In every block, the points are predicted from multiple pose-transformed point clouds. Subfigures (b,e,h,k) present the points prediction progress in the first block of f_Γ on four pose-transformed point clouds. Subfigures (c,f,i,l) and subfigures (d,g,j,m) show the points prediction progress in the second and third blocks respectively. In every subfigure of (b-m), the points are iteratively updated.

point position, velocity, attribute, and edge attribute as input to four of their motion prediction layers to estimate point position and attributes. These estimations are aggregated by Eqns.(8-10) in the paper as our PT-EvNet.

G. Details of Compared Networks in Table 6

Table 6 of the paper presents the computational cost of MPNN [2], RF [5], TFN [8], SE(3)-Tr. [1], EGNN [7], GMN [4], EGHN [3], EqMotion [9], FA-GNN [6]. When reporting the computational cost, we utilize the hyper-parameters specified in the codes of [3], [4], [7] to tune the networks. For EqMotion [9][4], we set the parameter Frame Length as 1 to keep same setting as PT-EvNet, i.e., predicting one future frame taking one history frame as input. For our PT-EvNet, we take the hyper-parameters described in Sect. C. i.e., we utilize three pose-transformed points prediction blocks f_Γ, and in each block we set $N_K = 4$ pose transformations to transform the pose-normalized point cloud. In the global-local motion estimation layer Φ, we take $N_S = 4$ iterations to estimate global and local displacements. The batch size, learning rate, weight decay are set as $12, 4e^{-4}, 1e^{-12}$ respectively.

[4]https://github.com/MediaBrain-SJTU/EqMotion

References

[1] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *NIPS*, pages 1970–1981, 2020. 4

[2] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol

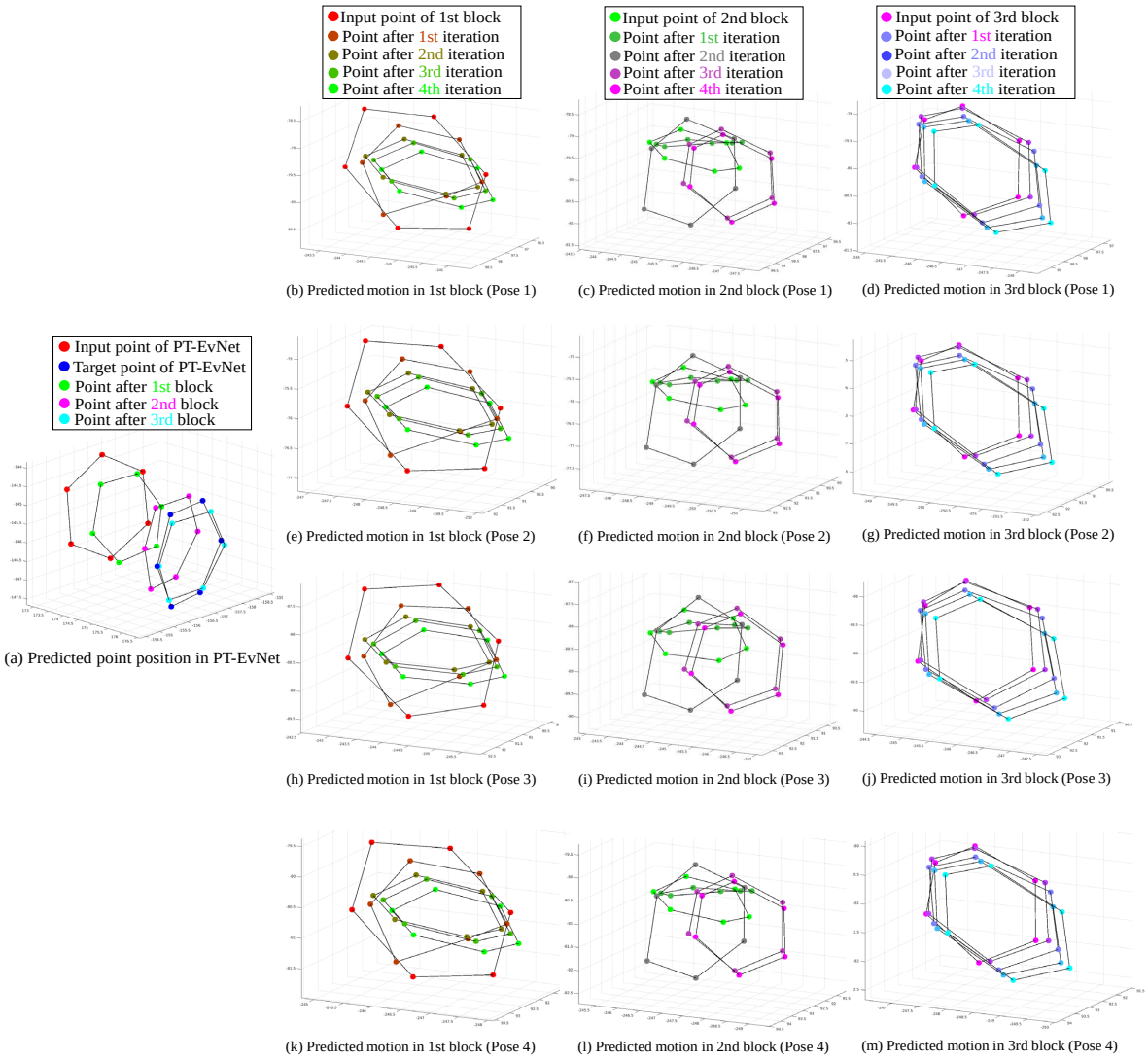


Figure 3. Visualization of points prediction progress in PT-EvNet on Benzene dataset. Subfigure (a) displays the predicted points by the three pose-transformed points prediction blocks f_Γ (in green/magenta/cyan), as well as the input points (in red) and target points (ground truth, in blue). In every block, the points are predicted from multiple pose-transformed point clouds. Subfigures (b,e,h,k) present the points prediction progress in the first block of f_Γ on four pose-transformed point clouds. Subfigures (c,f,i,l) and subfigures (d,g,j,m) show the points prediction progress in the second and third blocks respectively. In every subfigure of (b-m), the points are iteratively updated.

Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272, 2017. 3, 4

[3] Jiaqi Han, Wenbing Huang, Tingyang Xu, and Yu Rong. Equivariant graph hierarchy-based neural networks. In *NIPS*, 2022. 3, 4

[4] Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. In *ICLR*, 2022. 3, 4

[5] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019. 4

[6] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J. Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *ICLR*, 2022. 4

[7] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *ICML*, pages 9323–9332, 2021. 3, 4

[8] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. 4

[9] Chenxin Xu, Robby T Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *CVPR*, pages 1410–1420, 2023. 4