Supplementary Materials for Generalizable task representation learning from human demonstration videos: a geometric approach

Jun Jin† and Martin Jagersand†

I. Representing geometric constraints as graphs

We propose to use graphs as structural priors to encode the geometric constraints. More specifically, given a set of geometric features $\mathbf{X} = \{x_i\}$, we define an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a representation of the innerconnections between geometric features, where the nodes $\mathcal{V}$ are variables that take input of feature descriptors $x_i \in \mathbf{X}$ and edges $\mathcal{E}$ define their association relationships. Furthermore, suppose a graph neural network g is used to encode the graph structure $\mathcal{G}$ along with the node feature inputs $\mathbf{X}$ and output a latent embedding $\mathbf{z}$ representing this geometric constraint. We have:

$$\mathbf{z} = g(\mathbf{X}|\mathcal{G}) \tag{1}$$

A. Representing complex geometric features using keypoints

The above equation is the basic idea of geometric constraint graphs. Commonly, any geometric constraint is a binary relationship on two geometric features, such as two points, a point and a line, two lines, a point and a plane. To generally describe complex geometric features using keypoints, as proposed above, a geometric constraint can be defined as a multiple-entity relationship that operates on a set of feature points.

For convenience, we use $[x_i, ..., x_m]$ to define a complex geometric feature, where x_i are the keypoints used for its representation. Given this defintion, eq. 1 is rewritten as below:

$$\mathbf{z} = g(\mathbf{X} = \{[x_1, ..., x_m], [x_{m+1}, ..., x_n]\}|\mathcal{G}) \tag{2}$$

, where x_i are keypoints, $[x_1, ..., x_m]$ and $[x_{m+1}, ..., x_n]$ represent the two geometric features, the square brackets define a complex feature composed from a set of keypoints. For simplicity, we denote eq. 2 as:

$$\mathbf{z} = g([x_1, ..., x_m], [x_{m+1}, ..., x_n]) \tag{3}$$

Since we've introduced an extra composition structure to represent a complex geometric feature, the graph structure $\mathcal{G}$ should satisfy the below two properties. For convience, let us give the line-to-line constriant as an example, its graph representation is $g([x_1, x_2], [x_3, x_4])$ where each line is represented using two points:

- **Permutation-invariant**, which means an arbitrary order of the two complex features and the keypoints inside each feature should contribute to an invariant graph structure, namely:

$$g([x_1, x_2], [x_3, x_4]) = g([x_3, x_4], [x_1, x_2]) \tag{4}$$

$$g([x_1, x_2], [x_3, x_4]) = g(\pi([x_1, x_2]), \pi([x_3, x_4])) \tag{5}$$

  , where π is the permutation operator.
- **Non-inner-associative**, which means changing the inner association rules of the complex features will change the graph structure, namely

$$\begin{aligned} g([x_1, x_2], [x_3, x_4]) &\neq g([x_1, x_4], [x_3, x_2]) \\ g([x_1, x_2], [x_3, x_4]) &\neq g([x_1, x_3], [x_2, x_4]) \end{aligned} \tag{6}$$

Eq. 4 is naturally satisfied by the graph definition and graph neural network [1] since a graph neural network is required to be permutation-invariant to the input orders of node features. However, to satisfy eq. 5 and eq. 6, specially designed $\mathcal{G}$ is required for each geometric constraint representation, which is discussed below.

a) Graph structure of basic geometric constraints:: Now we give the graph structure $\mathcal{G}$ of the three basic geomeric constraints as shown in Fig. 1: (1) point-to-point; (2) point-to-line; (3) line-to-line.

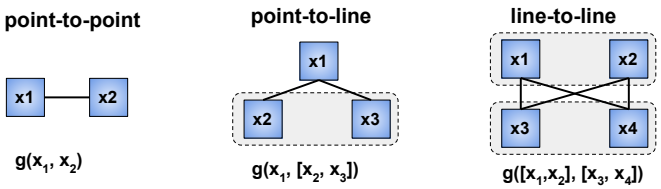

Fig. 1: The select-out graph structure of three basic geometric constraints. The grey shaded region means the two points represent a line. The selection process is shown in Fig. 2

Each graph structure is derived by enumerating all possible node connections and filtering out graphs that do not satisfy the "permutation-invariant" or "non-inner-associative" property. To list all possible node connections inside a graph $\mathcal{G}$, we require any node $v \in \mathcal{G}$ that its degree should satisfy $deg(v) \geq 1$ and the intermediate connection nodes with $deg(v) \geq 2$, since the graph needs organize all the points without any node or edge isolations. For example, consider the line-to-line constraint using four keypoints. There are a total of 38 possible structures of $\mathcal{G}$. After filtering out, the structure shown in Fig. 1 is selected out. Fig. 2 shows the selection process.

†Authors are with the Department of Computing Science, University of Alberta, Edmonton AB., Canada, T6G 2E8. {jjin5,mj7}@ualberta.ca

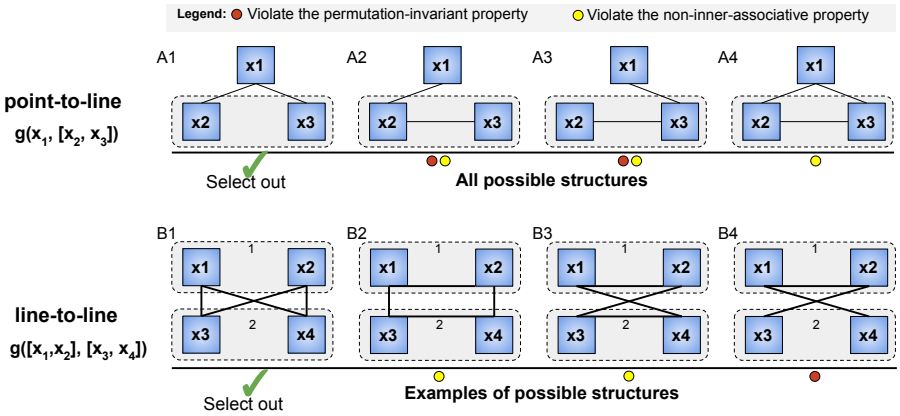

Fig. 2: Examples of how the graph structure of point-to-line and line-to-line constraints are selected out. We firstly enumerate all possible node connections without any node or edge isolation, wherein the point-to-line has 4 candicates and the line-to-line has 38 candidates. Then we filter out candicates that violate the above mentioned two properties. As shown above, red dot indicates violation of the permutation-invariant property and yellow dot for the non-inner-associative property. To give examples of such violations, in B3, non-inner-associative property is violated since $g([x_1, x_2], [x_3, x_4]) = g([x_1, x_4], [x_3, x_2])$.In B4, permutation-invariant property is violated since $g([x_1, x_2], [x_3, x_4]) \neq g([x_3, x_4], [x_1, x_2])$.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.