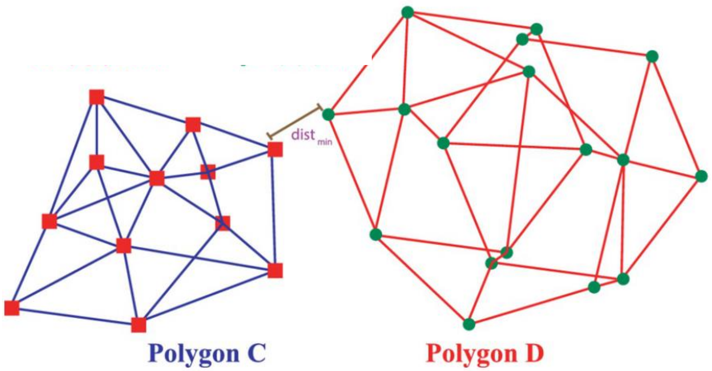
CS686: Paper presentation 2

GraphDistNet: A Graph-based Collision-distance Estimator for Gradient-based Trajectory Optimization

Yeeun Lim (임예은)

KAIST

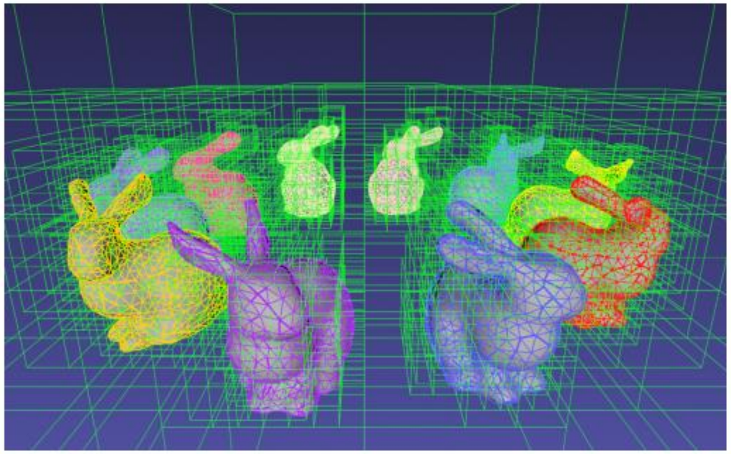
Motivation



- **Collision detection**
  - **Geometric Algorithm**

    ex) GJK algorithm

    time and space consuming



  - **Data-driven Algorithm**

    ex) configuration-based, point cloud - based

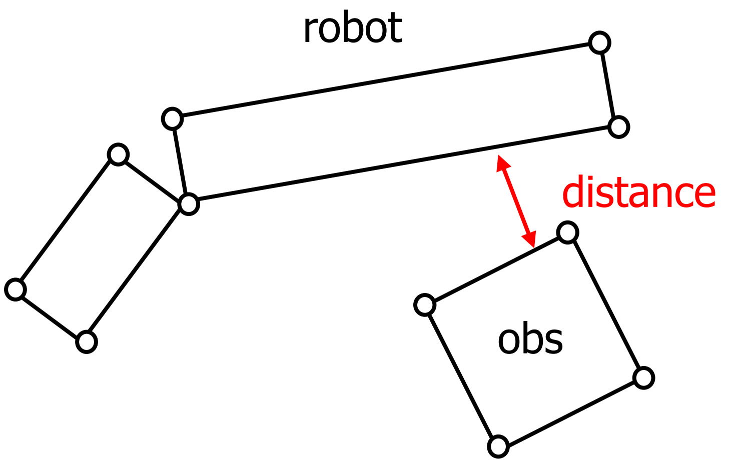
    scalability issuses

https://www.researchgate.net/figure/The-figure-explains-the-working-of-the-GJK-algorithm-where-two-arbitrary-bodies-are_fig2_348539875
https://www.kitware.com/octree-collision-imstk/

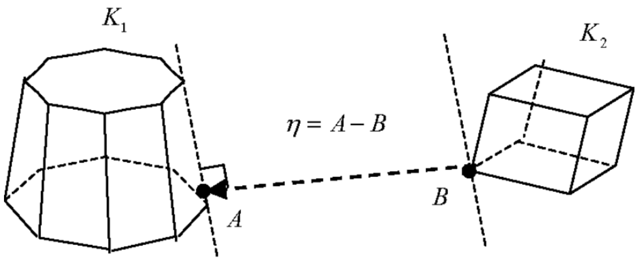
Motivation

GraphDistNet

: Graph neural networks-based collision distance estimator for trajectory optimization

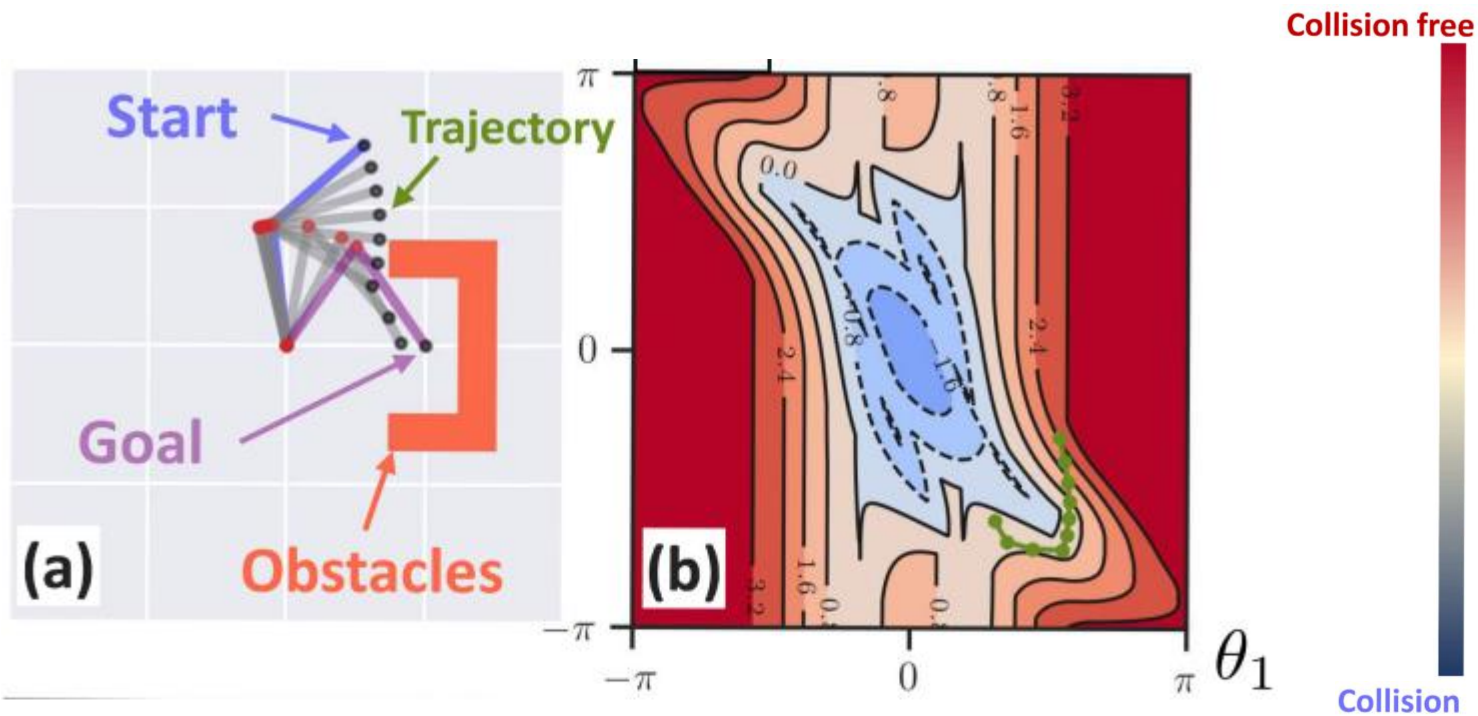


- **Estimated distance**
- **Calculating collision gradient**



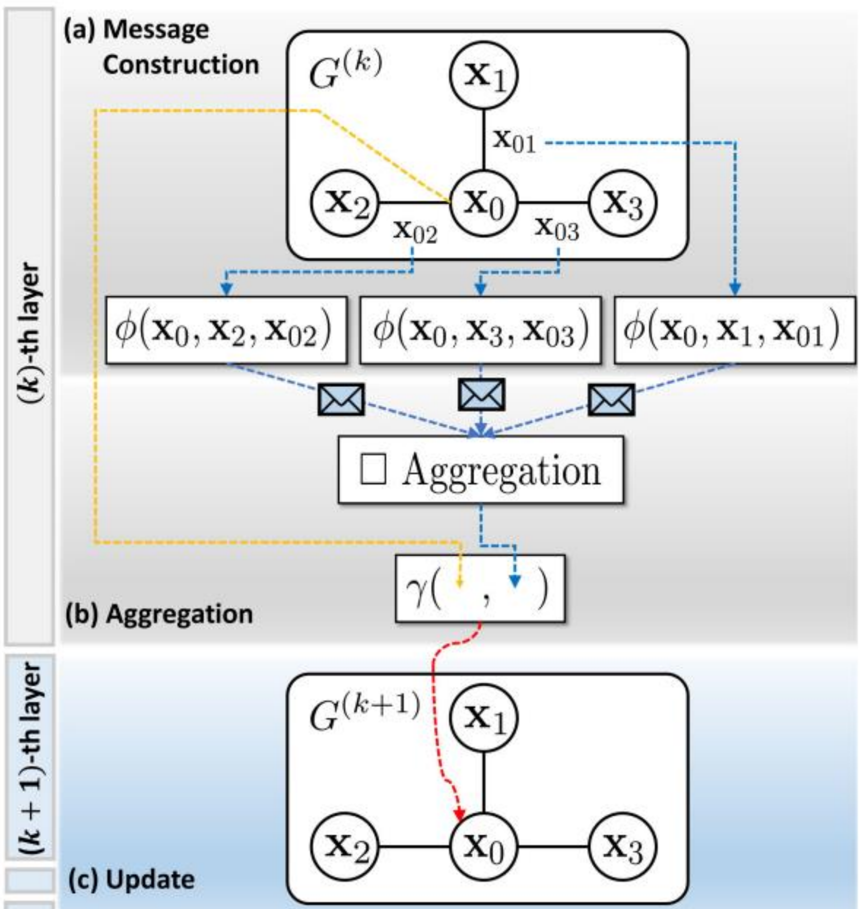
Background

- **Trajectory optimization**



Background

- **Graph Neural Network**



$$G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$$

$$+ \ edge\ features\ \mathbf{x}_{ij} = h_\theta(\mathbf{x}_i, \mathbf{x}_j)$$

Aggregation method
: ***message passing***

1) Each node v_i computes a message ϕ
2) Each node aggregates the messages $\square$ using sum or average operations
3) Each node updates the node feature γ using aggregated message and current

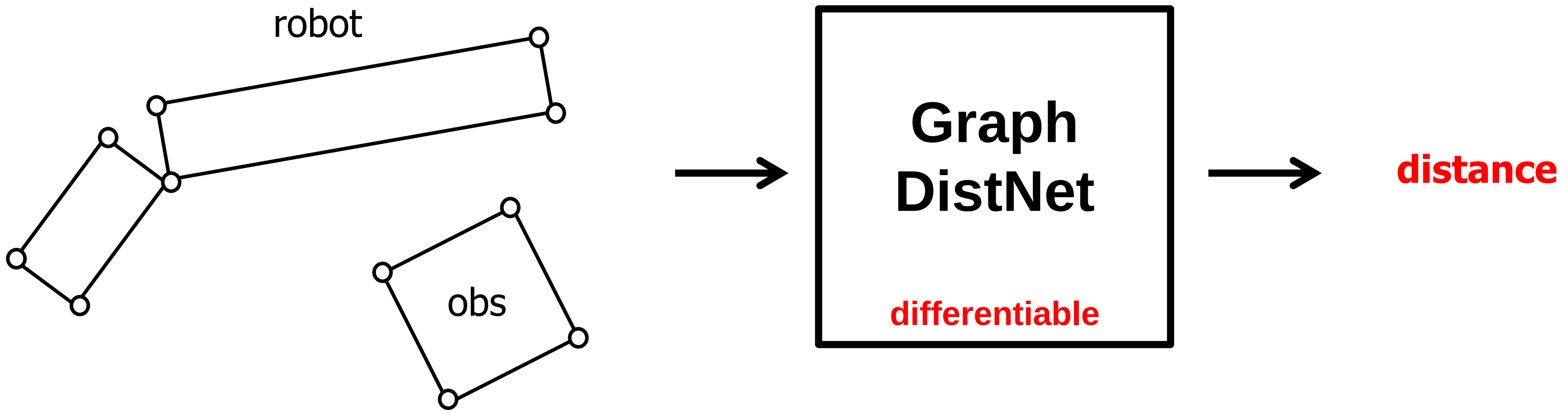
$$\mathbf{x}_i^{(k+1)} = \gamma\left(\mathbf{x}_i^{(k)}, \square_{j:(i,j)\in\mathcal{E}} \phi\left(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(k)}, \mathbf{x}_{ij}^{(k)}\right)\right)$$

γ, ϕ : MLPs

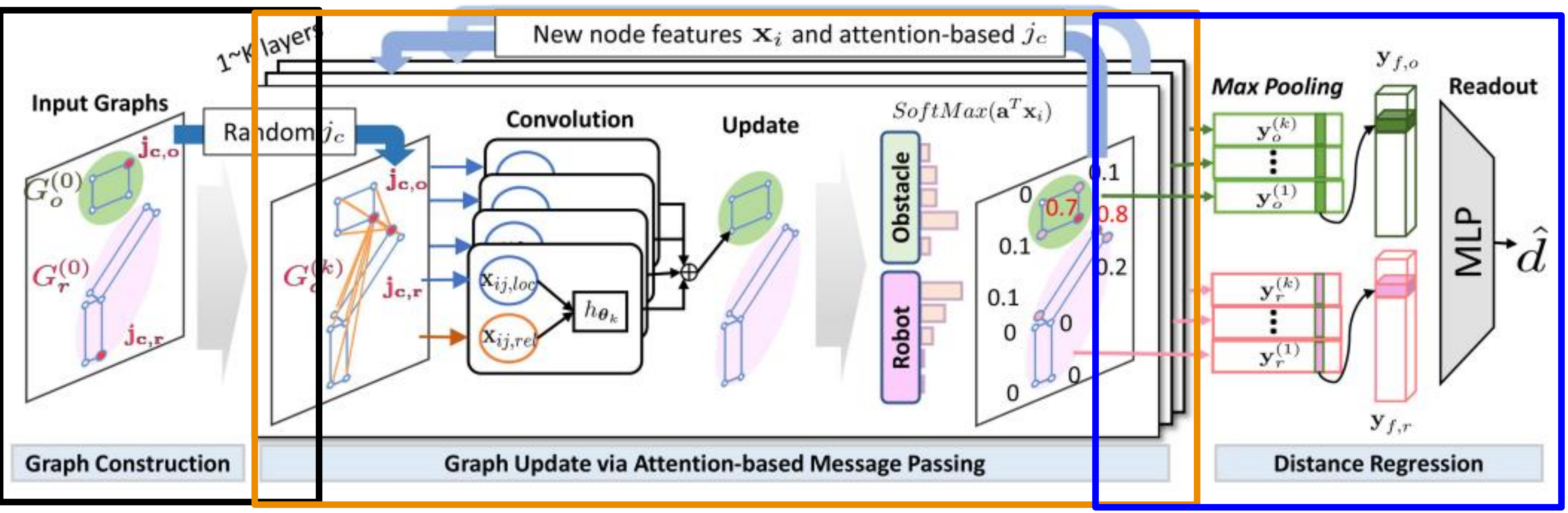
Method

GraphDistNet

: Graph neural networks-based collision distance estimator for trajectory optimization



Method : GraphDistNet



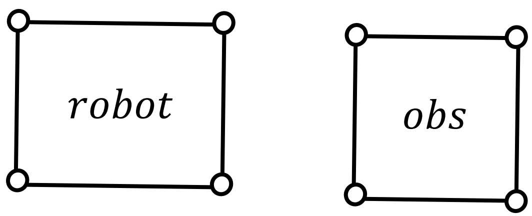
1. Initial Graph Construction

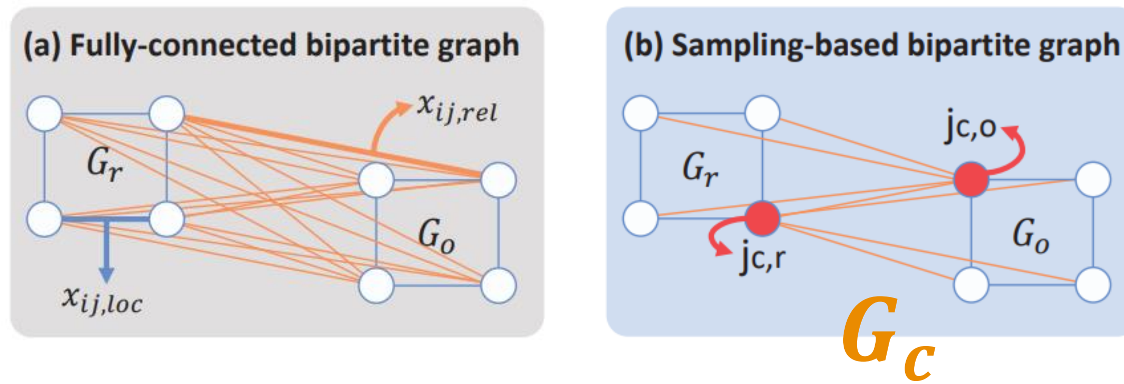
2. Graph Update via Attention-based Message Passing

3. Collision-distance and gradient Estimation

Method

1. Initial Graph Construction



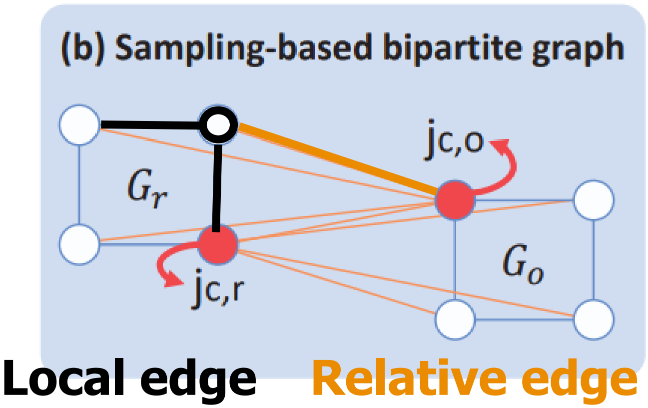


Randomly select j_c

j_c **: informative node**

Method

2. Graph Update via Attention-based Message Passing



Update graph : $G_u^{(1)}, G_u^{(2)}, G_u^{(3)} \ldots. G_u^{(k)}$

$$\mathbf{x}_i^{(k+1)} = \sum_{j \in \mathcal{N}_{ro}(i) \cup \{i\}} h_{\boldsymbol{\theta}_k}^{(k)} \left(\mathbf{x}_{ij,loc}^{(k)}, \mathbf{x}_{ij_c,rel}^{(k)} \right)$$

$h_{\theta_k}^{(k)}$: MLP-based encoder

* Node feature x_i : cartesian coordinate.

* Edge feature $$\mathbf{x}_{ij}^{(k)} = \begin{cases} \left(\mathbf{x}_j^{(0)} - \mathbf{x}_i^{(0)}\right) & \text{if } k = 0, \\ \left(\mathbf{x}_j^{(0)} - \mathbf{x}_i^{(0)} \parallel \mathbf{x}_j^{(k)} - \mathbf{x}_i^{(k)}\right) & \text{otherwise,} \end{cases}$$

Method

After each convolution,

$$\overset{\text{New } j_c^2}{\curvearrowright} \quad \overset{\text{New } j_c^3}{\curvearrowright}$$

$$G_u^{(1)},\ G_u^{(2)},\ G_u^{(3)} \ldots\ldots$$

Reselect most informative j_c by introducing **attention-based selection method.**

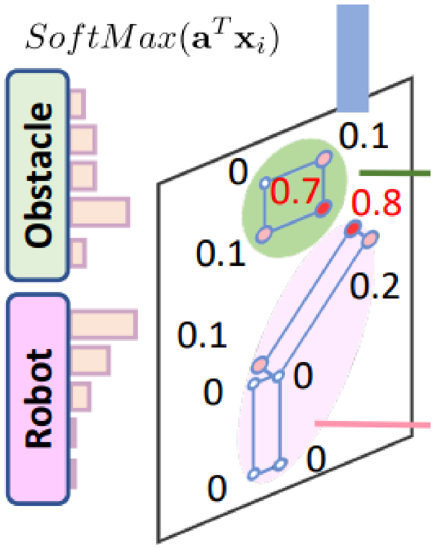
Attention score $\mathbf{a}^{(k)T}\mathbf{x}_i^{(k)}$

Vector of trainable parameter in attention mechanism

Attention weights

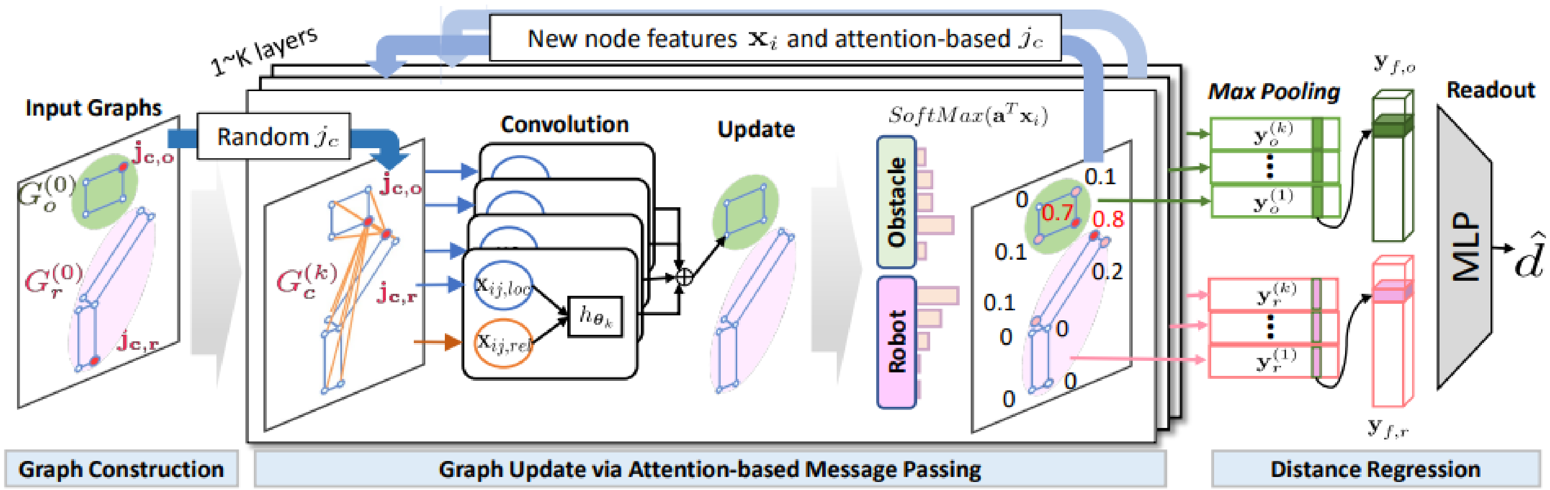
$$\alpha_{i_r}^{(k)} = \frac{\exp(\mathbf{a}^{(k)T}\mathbf{x}_{i_r}^{(k)})}{\sum_{j_r}\exp(\mathbf{a}^{(k)T}\mathbf{x}_{j_r}^{(k)})}$$

$$\alpha_{i_o}^{(k)} = \frac{\exp(\mathbf{a}^{(k)T}\mathbf{x}_{i_o}^{(k)})}{\sum_{j_o}\exp(\mathbf{a}^{(k)T}\mathbf{x}_{j_o}^{(k)})}$$



Method

3. Collision-distance and gradient Estimation



Attention weighted feature

$$\mathbf{y}_r^{(k)} = f_{LR}\left(\sum_{i_r} \alpha_{i_r}\mathbf{x}_{i_r}^{(k)}\right) \in \mathbb{R}^{d_h},$$

$$\mathbf{y}_o^{(k)} = f_{LR}\left(\sum_{i_o} \alpha_{i_o}\mathbf{x}_{i_o}^{(k)}\right) \in \mathbb{R}^{d_h},$$

Leacky-ReLU

$$\hat{d} = MLP\Big(\underbrace{\big\Vert_{i=1}^{d_h} \max_k \mathbf{y}_r^k(i)}_{\mathbf{y}_{f,r}} \,\Vert\, \underbrace{\big\Vert_{i=1}^{d_h} \max_k \mathbf{y}_o^k(i)}_{\mathbf{y}_{f,o}}\Big),$$

Evaluation

- **We can use this as …**

- Binary Collision Checker

$$\text{Is collision?} = \begin{cases} \text{True} & \text{if } \hat{d} \leq d_{margin}, \\ \text{False} & \text{otherwise,} \end{cases}$$

- Collision-gradient estimator

$$\partial GraphDistNet(\boldsymbol{\theta}) / \partial \boldsymbol{\theta}$$

-> Gradient-based trajectory optimization

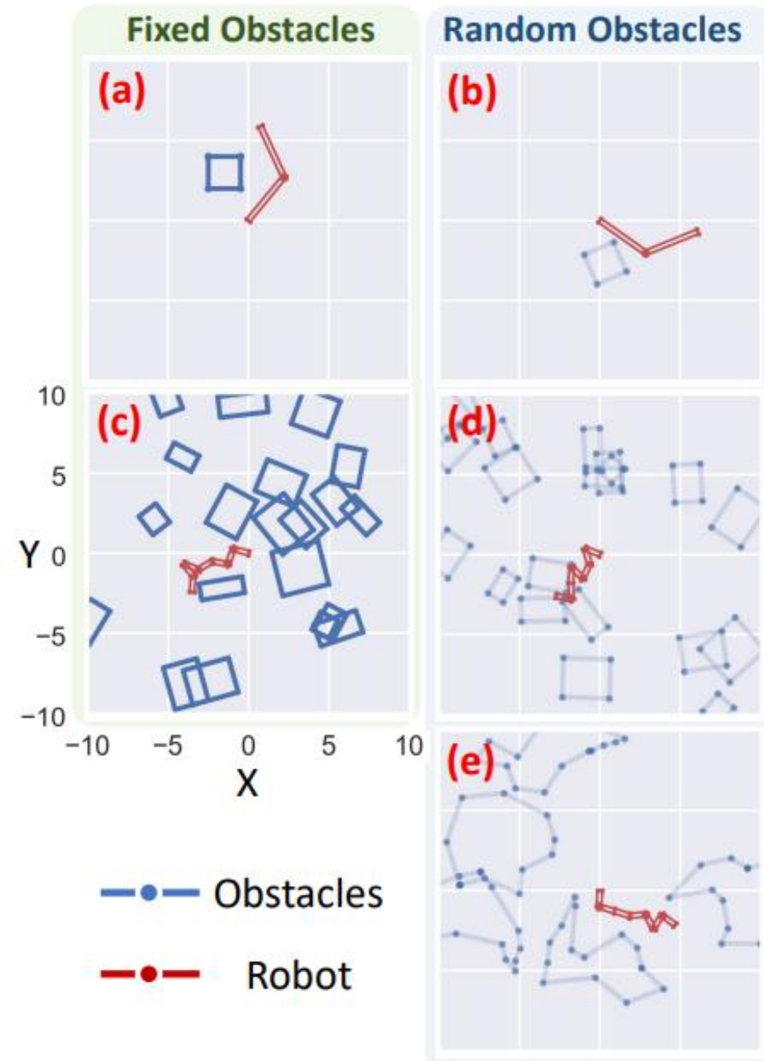
Evaluation

- **environments.**

- 2-DOF & 7-DOF

- Fixed Obstacles & Random Obstacles

- Shape of obstacle



- **Baselines.**

- DiffCo

- ClearanceNet

- FCL use as ground truth

Evaluation

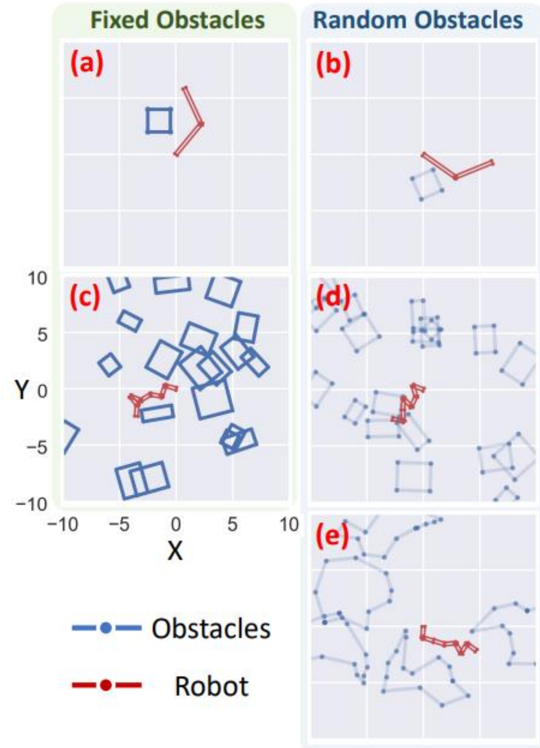
- **Estimation performance**

MAE : mean absolute error

AUC : area under the ROC (receiver operating characteristic)

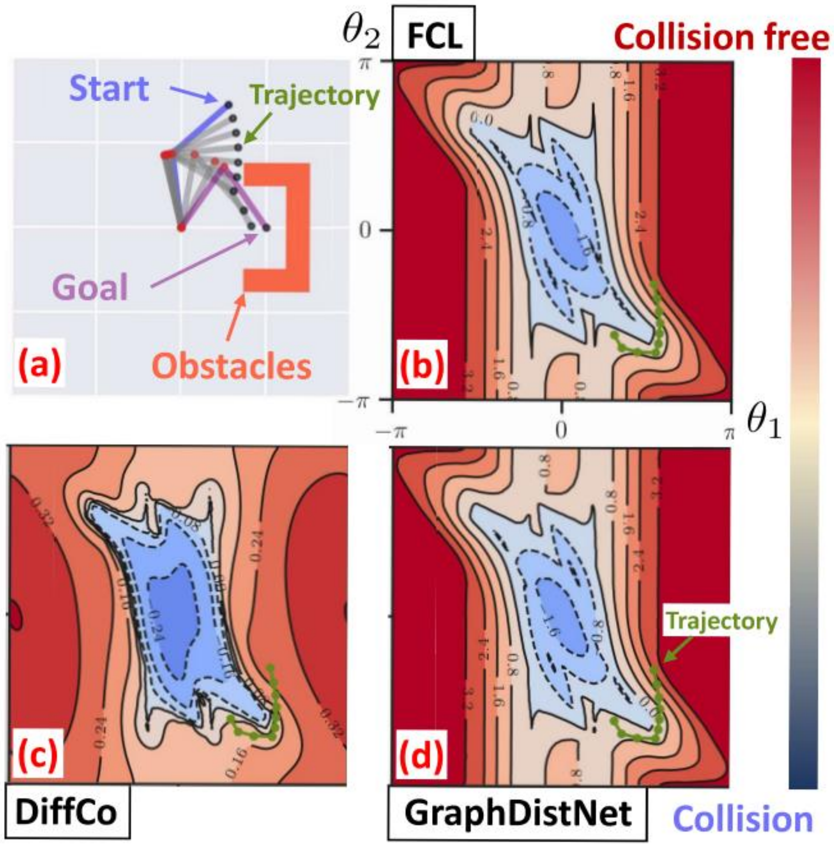
ACD : cosine distances of estimated gradient fields.

Env.	Method	Distance Regression & Classification: Elapsed Time (s)	MAE with p-value		AUC	Gradient Est.: ACD
Fig. 5 (a)	FCL	1.0653	-		-	-
	DiffCo	0.0048	N/A		1.0000	0.3575
	ClearanceNet	**0.0009**	0.0082	6×10^{-20}	1.0000	0.0255
	GraphDistNet	0.0115	**0.0031**		**1.0000**	**0.0054**
Fig. 5 (b)	FCL	1.1822	-		-	-
	DiffCo (w/o active learning)	0.0054	N/A		0.5031	1.0270
	DiffCo (w/ active learning)	2058.2	N/A		0.9948	0.4221
	ClearanceNet	**0.0009**	0.1732	2×10^{-12}	0.9986	0.2853
	GraphDistNet	0.0115	**0.0390**		**0.9999**	**0.1748**
Fig. 5 (c)	FCL	3.1573	-		-	-
	DiffCo	0.0740	N/A		0.9843	0.3202
	ClearanceNet	**0.0012**	0.2360	6×10^{-60}	0.9111	0.5933
	GraphDistNet	0.0574	**0.0253**		**0.9990**	**0.1458**
Fig. 5 (d)	FCL	4.4467	-		-	-
	DiffCo (w/o active learning)	0.0742	N/A		0.5239	0.9450
	DiffCo (w/ active learning)	75129.5	N/A		0.9798	**0.3968**
	ClearanceNet*	**0.0012**	0.7725	1×10^{-10}	0.5691	0.9491
	GraphDistNet	0.0546	**0.1614**		**0.9842**	0.4152
Fig. 5 (e)	FCL	6.8432	-		-	-
	DiffCo (w/o active learning)	0.0350	N/A		0.5200	0.9612
	DiffCo (w/ active learning)	(Failure: Out of Memory)				
	ClearanceNet*	**0.0011**	0.7479	3×10^{-2}	0.8882	0.9767
	GraphDistNet	0.1251	**0.2446**		**0.9925**	**0.8421**



Evaluation

- **Gradient fields in 2-DoF**



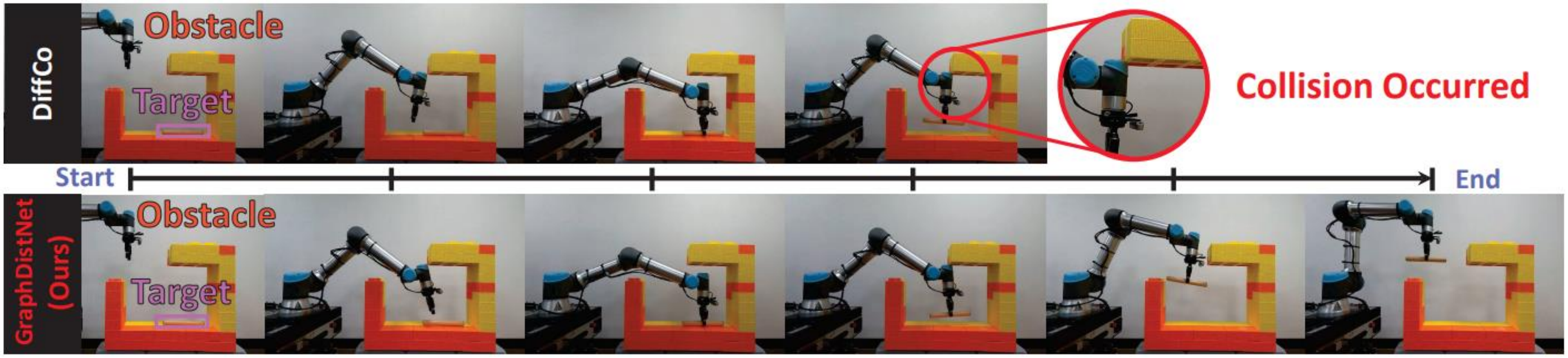
Evaluation

- **Trajectory Optimization**

	Env.	Collision Checker	Avg. Elapsed Time (s)	Avg. Path Cost	Success Rate (%)
Simple env	Fig. 5 (b)	DiffCo	0.9304	**23.7657**	0.95
		ClearanceNet	1.7300	24.9018	**1.0**
		GraphDistNet	1.9522	24.4297	**1.0**
Complex env	Fig. 5 (d)	DiffCo	28.8247	99.0850	**0.95**
		ClearanceNet	10.9816	72.3153	0.4
		GraphDistNet	16.2682	**65.0297**	0.9
	Fig. 1	DiffCo	0.6364	3.9×10^5	0.4
		ClearanceNet	1.9603	5.5×10^5	0.3
		GraphDistNet	3.2982	9.0×10^5	**0.7**

Evaluation

- **Demonstration**



Conclusion

- **Contribution**

- graph-based collision-distance estimation network,
  that precisely regresses the collision distance between objects.

- accurate gradients and batch computation,
  improving trajectory optimization

- robust to various to various environmental changes and unseen environments

Q & A

- **Thank you for listening ☺**

Quiz

Q1. List the sequence of GraphDistNet

a) Graph updating via message passing

b) Graph construction with j_0

c) Distance regression

Q2. Which is NOT possible with GraphDistNet?

a) estimating collision distance

b) generating trajectory

c) calculating gradient of distance