Goal-Conditioned Dynamic Graph Model for Task and Motion Planning

M. Clara De Paolis Kaluza[1], Christopher Paxton[3], Animesh Garg[2,3], Anima Anandkumar[4,3], Rose Yu[1]

[1] Northeastern University, [2] Univ. of Toronto, [3] NVIDIA Co., [4] CalTech

Problem: Task and motion planning

Task planning: reasoning about the environment and goal task at high level (at the object level),

Motion planning: solves for low-level controls for robot arm motors to move between configurations at each step of task

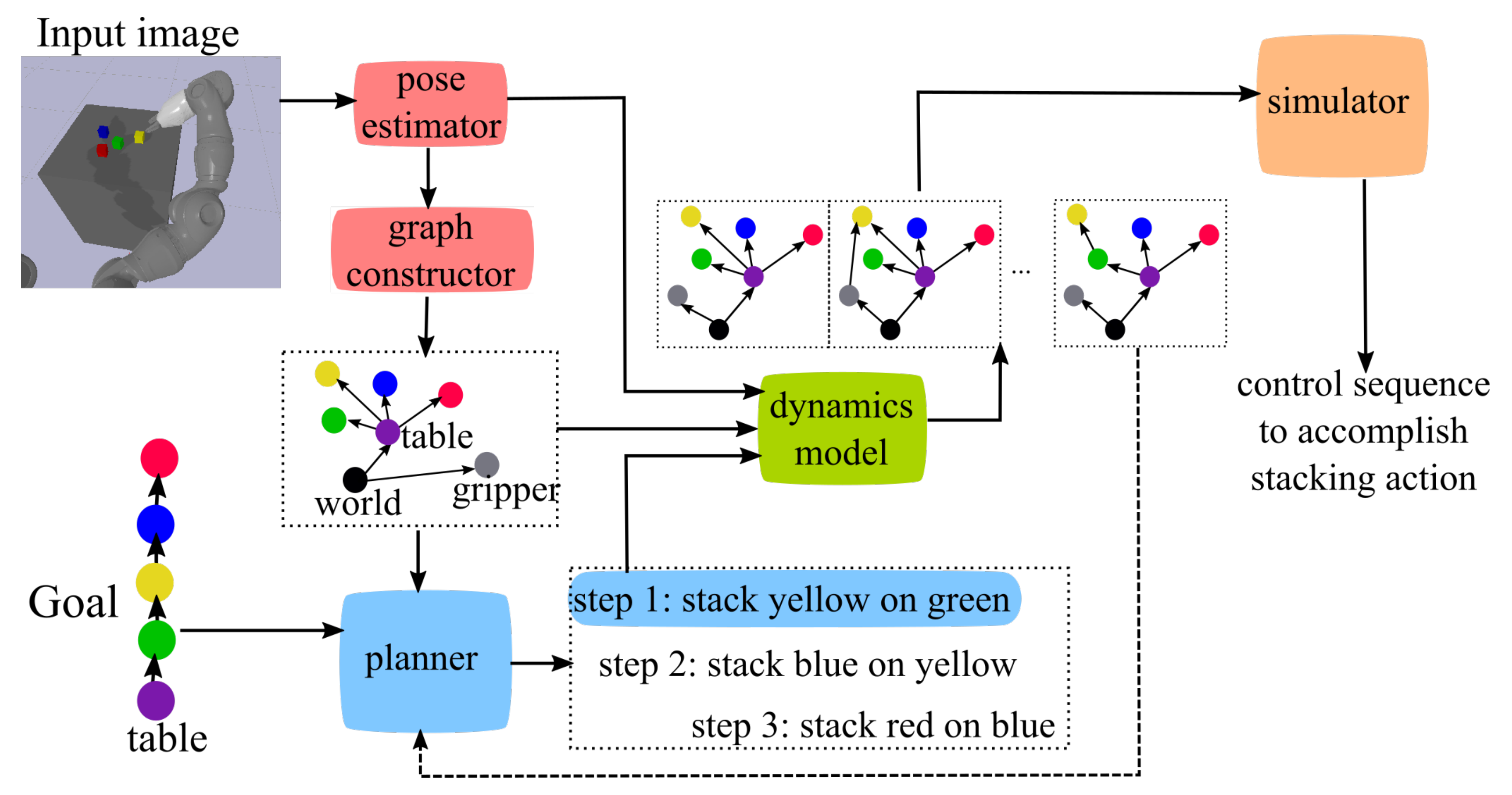
Graphs encode a sparse *high-level abstraction for a task*, such as how objects interact with each other, as well as *low-level information* about the individual objects (location and rotation). Dynamics model to predict a dynamic graph for a achieving a goal can be used to validate **task planning** and execute **motion planning**.

Our Solution

Graph-based dynamics model: predicts sequence of graphs conditioned on goal. Yields high-level object representation for planning and lower-level pose prediction for task execution.

End-to-end planning pipeline: Planner suggests sub-goals for dynamics model. Dynamics model validates plan and provides pose predictions for simulator to calculate control signals.

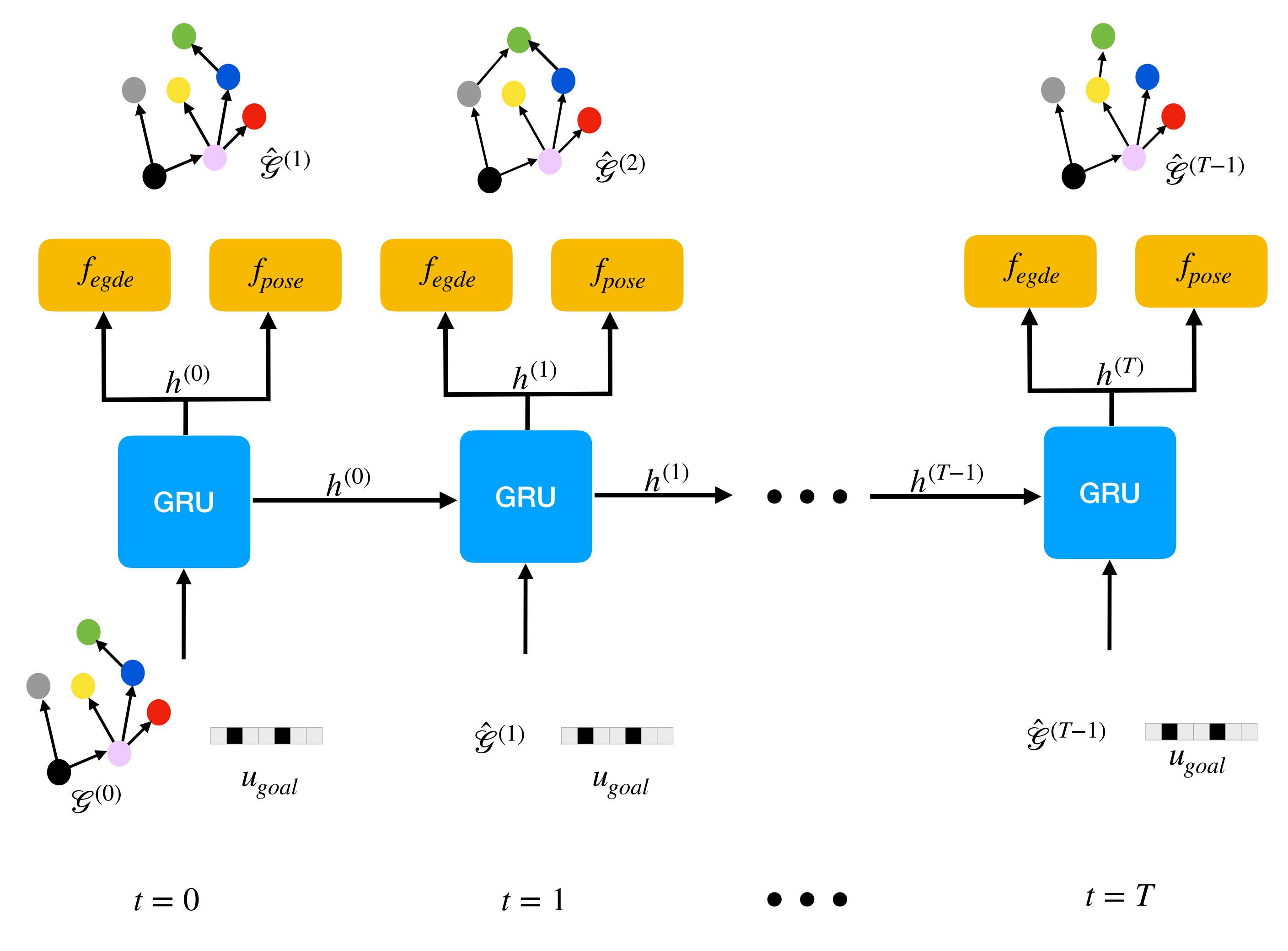
Goal-conditioned Planning Pipeline


Input image
pose estimator
graph constructor
table
world
gripper
dynamics model
simulator
control sequence to accomplish stacking action
Goal
table
planner
step 1: stack yellow on green
step 2: stack blue on yellow
step 3: stack red on blue
...

Defining graphs: Nodes for each object in scene (blocks, surface, robot arm), plus a "world" node to make graph connected. Directed edges indicate physical support, *e.g.* edge from table to block indicates takes is supporting block.
Planner: Takes input graph and goal configuration and outputs sequence of sub-goals, *i.e.* single stacking actions like `stack yellow on green`. We use simple A^* search to generate plan.
Dynamics model: Outputs sequence of graphs with object poses. We use these graphs to validate the proposed plan and to calculate the low-level control signals for robot gripper to execute plan.

Dynamics Model: Graph sequence generation

Idea: From initial graph $\mathcal{G}^0$ with poses $\mathbf{X}^{(0)}$ and constructed adjacency $\mathbf{A}^{(0)}$ and (sub)goal vector $\mathbf{u}_{goal}$, produce sequence of poses and graphs for achieving (sub)goal.



Decode graph from recurrent state $\mathbf{h}^{(t)}$

$$p\left(\mathbf{X}^{(t)}|\mathbf{h}^{(t)}, \mathbf{u}_{\text{goal}}\right) = f_{node}\left(\mathbf{h}^{(t)}\right) \qquad p\left(\mathbf{A}^{(t)}|\mathbf{h}^{(t)}, \mathbf{u}_{\text{goal}}\right) = f_{node}\left(\mathbf{h}^{(t)}\right)$$

Supervised Loss Function: $\mathcal{L} = \mathcal{L}_{pose} + \gamma\mathcal{L}_{graph}$
Low-level node information

$$\mathcal{L}_{pose} = \frac{1}{T}\sum_{t=1\ldots T} \|\mathbf{X}^{(t)} - \hat{\mathbf{X}}^{(t)}\|_2$$

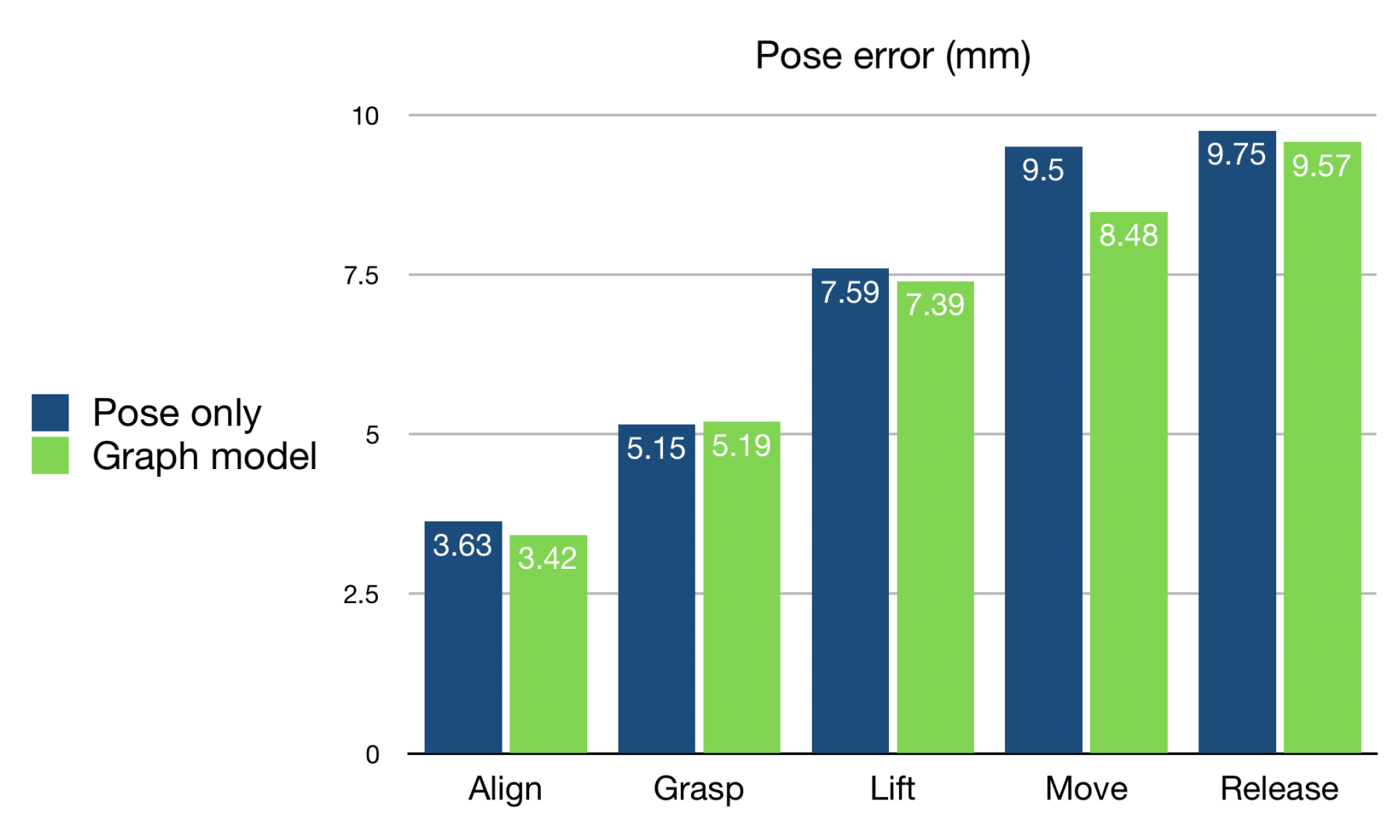
High-level graph structure

$$L_{graph} = \sum_{e_{i,j}\in\mathcal{E}} -\mathbf{A}^{(t)}_{i,j}\log(e_{i,j}) + (1-\mathbf{A}^{(t)}_{i,j})\log(1-e_{i,j}))$$

Experiments

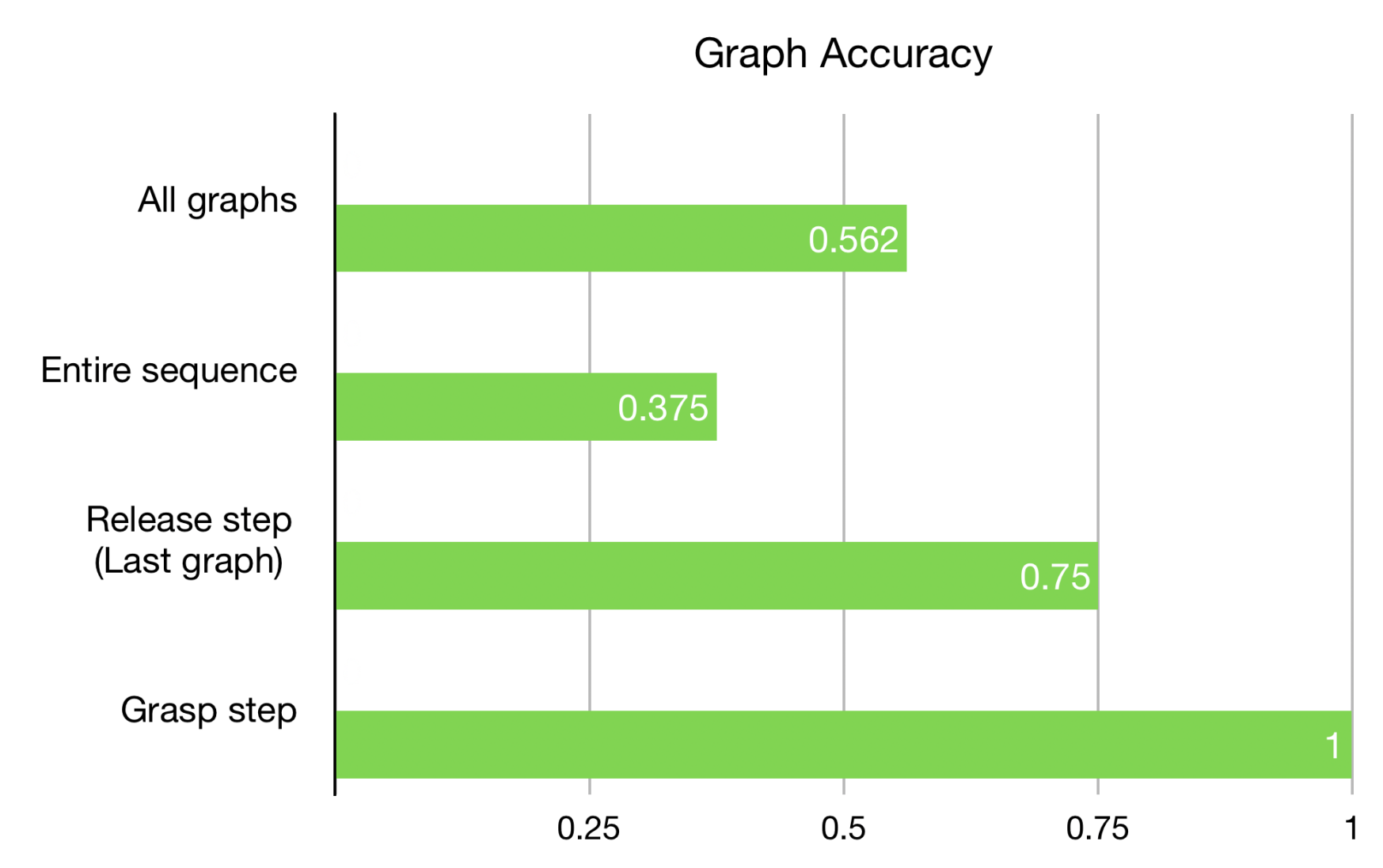
Evaluating Dynamics Model

Dynamics model predicts poses for all blocks and graph structure: we can validate the proposed plan by checking that blocks end up at desired locations and graph edges follow physics of the task.
We use 7d poses: three (x, y, z) Cartesian coordinates in 3d space and a quaternion (a, b, c, d) to specify the rotation.



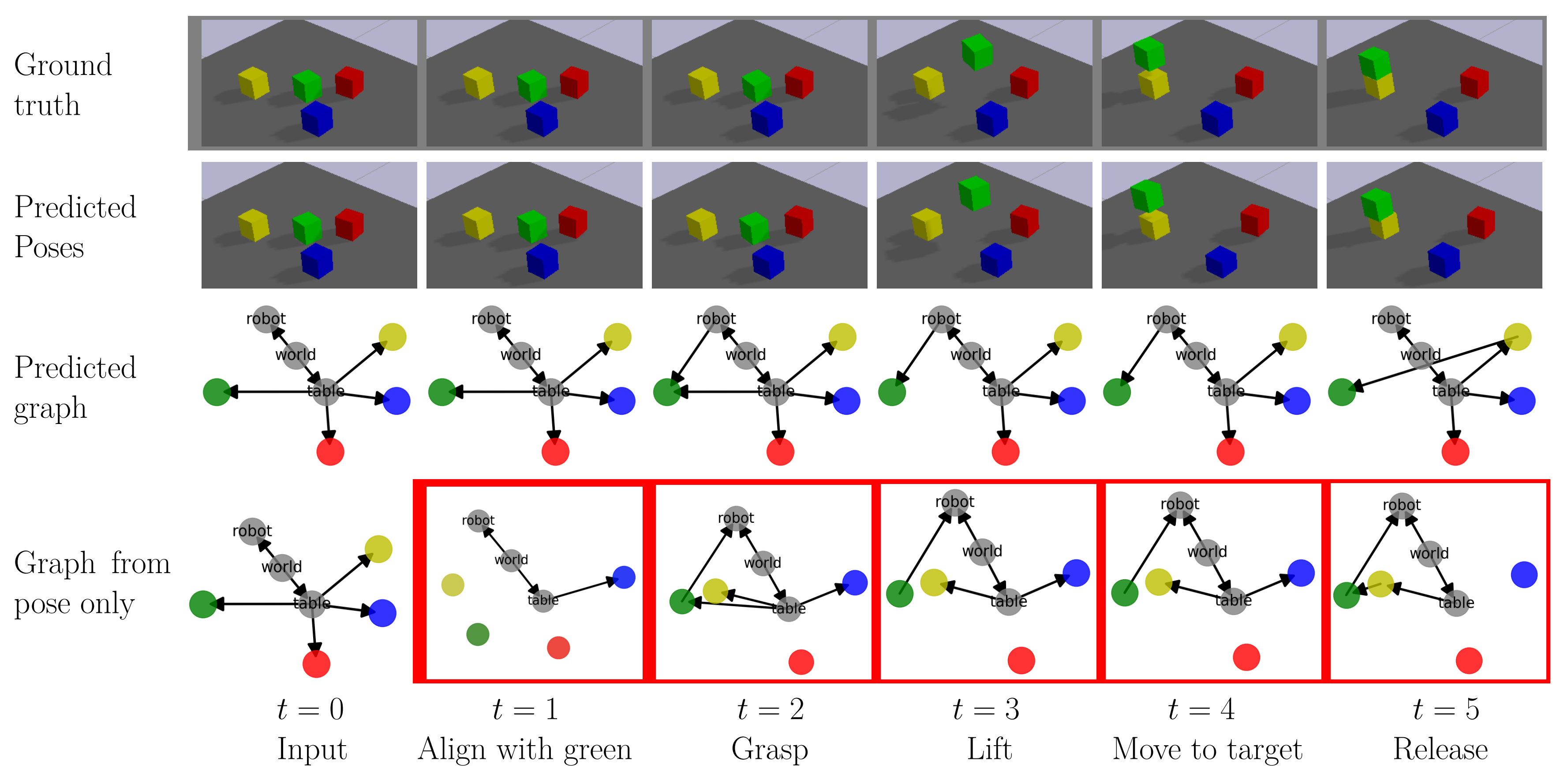
	Align	Grasp	Lift	Move	Release
Pose only	3.63	5.15	7.59	9.5	9.75
Graph model	3.42	5.19	7.39	8.48	9.57

Euclidean error for pose estimation (lower is better)



	Graph Accuracy
All graphs	0.562
Entire sequence	0.375
Release step (Last graph)	0.75
Grasp step	1

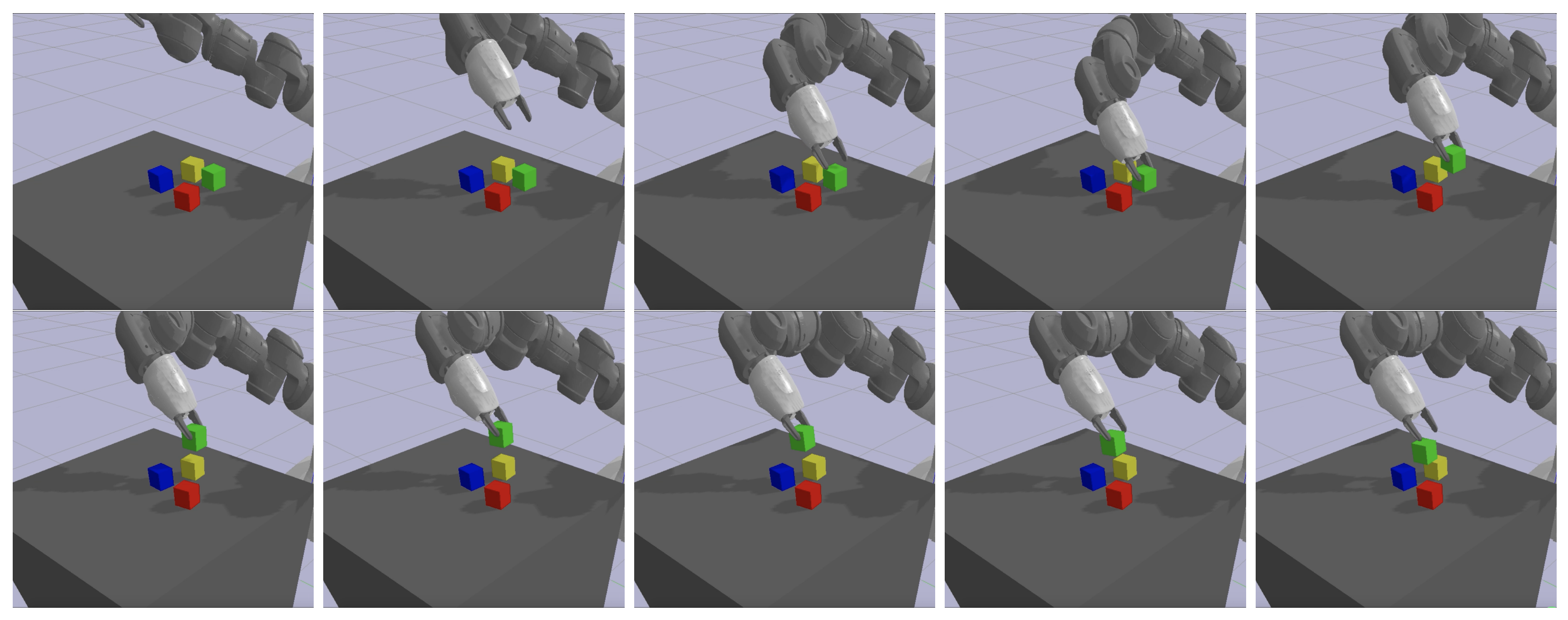
Percent of graphs correctly predicted



Ground truth (top row) and predicted (second row) block poses for first goal action, "stack green on yellow." Predicted graph structure (third row) agrees with predicted poses; edges exist only when blocks are in contact. The bottom row show the graphs constructed heuristically from the predicted poses. The red highlighted graphs disagree with the model-predicted graphs, showing the robustness of the predicted graph structure to small errors in the predicted poses.

Control Signals From Predicted Poses

Dynamics model provides prediction of poses for sequence. We use the Inverse Kinematics solver in PyBullet [2] on the predicted poses of robot gripper to compute control signals for the robot arm joints to execute goal of stacking block.



Frames from example simulation using control signals from Inverse Kinematics in PyBullet

Next Steps

- Improvements to dynamics model for more settings
- Refinement of Inverse Kinematics calculations for smoother movement and fewer collisions
- Long term predictions by rolling out entire plan

References

[1] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.