LEARNING GEOMETRIC REASONING NETWORKS FOR ROBOT TASK AND MOTION PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Task and Motion Planning (TAMP) is a computationally challenging robotics problem due to the tight coupling of discrete symbolic planning and continuous geometric planning of robot motions. In particular, planning manipulation tasks in complex 3D environments leads to a large number of costly geometric planner queries to verify the feasibility of considered actions and plan their motions. To address this issue, we propose Geometric Reasoning Networks (GRN), a graph neural network (GNN)-based model for action and grasp feasibility prediction, designed to significantly reduce the dependency on the geometric planner. Moreover, we introduce two key interpretability mechanisms: inverse kinematics (IK) feasibility prediction and grasp obstruction (GO) estimation. These modules not only improve feasibility predictions accuracy, but also explain why certain actions or grasps are infeasible, thus allowing a more efficient search for a feasible solution. Through extensive experimental results, we show that our model outperforms state-of-the-art methods, while maintaining generalizability to more complex environments, diverse object shapes, multi-robot settings, and real-world robots.

1 INTRODUCTION

Task and Motion Planning (TAMP) (Garrett et al., 2021) is a robotics problem in which the goal is to find a sequence of robot actions and their corresponding motions to transition an environment from an initial state to a goal state. In most cases, the order of actions, their grounded parameters, and the feasibility of the corresponding motions are tightly coupled, requiring a careful combination of symbolic task planning and continuous geometric planning. However, the resulting large search space leads to a combinatorial explosion. Moreover, every action considered must be validated geometrically to ensure feasibility, leading to numerous costly queries to the geometric planner. Thus, geometric planning can become a bottleneck for TAMP (Bouhsain et al., 2023b).

Previous works (Wells et al., 2019; Driess et al., 2020a; Khodeir et al., 2023a) demonstrate that learning methods can help accelerate TAMP by providing fast geometric feedback to the task planner. In particular, action feasibility prediction offers an efficient alternative to geometric planning during the TAMP process. In offline manipulation planning, it can answer critical questions such as which object can be picked, how to grasp it, where to place it, or how to free access to it. However, action feasibility prediction presents several challenges. A suitable representation of 3D environments is needed, as these may contain an arbitrary number of objects, along with action representations that capture varying parameters such as grasps. Furthermore, predictions must not only be fast and accurate but also interpretable to understand why an action is infeasible and how to rectify it (e.g. another object is blocking access to the desired one). Finally, action feasibility prediction must generalize to environments with numerous objects, objects of varying shapes, and multi-robot settings.

Existing approaches to action feasibility prediction often struggle with interpretability, scalability, and generalization across diverse environments. To address these limitations, we propose a novel approach that leverages a GNN-based model for robot action and grasp feasibility prediction. Our method constructs a graph representation of 3D environments, where fixed and movable objects are represented as nodes, and edges capture spatial relationships and interaction constraints. Through this graph-based structure, we leverage an Edge-Enhanced Graph Attention Network (EGAT) to predict action and grasp feasibility for each movable object. A unique aspect of our approach is the introduction of two interpretability mechanisms: inverse kinematics (IK) feasibility predictions,

which determine whether the robot can feasibly manipulate an object from different sides, and Grasp Obstruction (GO) predictions, which quantify how neighboring objects restrict access to grasps from different sides of the object. These interpretable features not only allow us to predict action infeasibility but also to explain why a specific action fails, enabling more efficient planning.

The contributions of this paper are threefold: (1) We propose a novel GNN-based model for efficient and accurate action and grasp feasibility prediction in complex 3D environments. (2) Our method introduces two mechanisms, inverse kinematics (IK) feasibility and grasp obstruction (GO) predictions, that improve action feasibility prediction while enhancing its interpretability and hence task planning efficiency. (3) We provide comprehensive experiments showcasing our method's performance in comparison to state-of-the-art (SOTA) approaches, including evaluations of its interpretability and generalization capabilities.

2 Related Works

2.1 Task and Motion Planning

Task and Motion Planning (TAMP) combines discrete symbolic task planning with continuous geometric motion planning to achieve robotic manipulation goals. Early approaches (Alami et al., 1990; Koga & Latombe, 1994; Ahuactzin et al., 1998; Siméon et al., 2004) view the problem primarily from a geometric perspective. These approaches build manipulation graphs that map valid grasps, placements, and connecting motions. Although, they suffer from a combinatorial explosion as the number of objects increases. Later, multi-modal motion planning (Hauser & Latombe, 2010; Hauser & Ng-Thow-Hing, 2011) generalized these methods using constraint-based graphs, but the complexity of constructing these graphs for cluttered environments remains a challenge. Modern methods integrate symbolic task planning with geometric planners, using backtracking when actions are infeasible (Cambon et al., 2009; Srivastava et al., 2014; Lagriffoul et al., 2014; Dantam et al., 2016; Garrett et al., 2018). However, in complex environments, the large number of queries to the geometric planner causes significant computational overhead, leading to slow planning processes.

2.2 Learning for Task and Motion Planning

Many recent works leverage learning methods to provide heuristics for TAMP. Chitnis et al. (2016) apply reinforcement learning (RL) to learn policies for task refinement. Xu et al. (2021) model both immediate and future affordances in a learned latent space. Some methods leverage GNNs to identify the smallest set of objects for solving a planning problem (Silver et al., 2021), learning inter-robot relations to verify subgoal satisfaction (Huang et al., 2023), or predict dynamic object interactions (Chen et al., 2023). Eisner et al. (2024) tackles the problem of precise relative object placement using an SE(3)-equivariant learning. Agia et al. (2023) learn task-agnostic policies for various robot skills and verify their feasibility. Other works learn to fully solve TAMP problems. Zhu et al. (2021) leverage a two-level scene graph for neuro-symbolic task planning and graph-based motion generation. McDonald & Hadfield-Menell (2022) propose an imitation learning method that mimics a TAMP solver. Lin et al. (2022) introduce a GNN-based policy architecture trained on expert demonstrations. Haramati et al. (2024) propose a goal-conditioned RL framework using an object-centric image representation of the 3D environment. These methods tend to be problem-dependent, needing further training on unseen manipulation problems. Wang et al. (2024), Lin et al. (2023); Huang et al. (2024b) leverage large language models (LLM) to perform TAMP, incorporating feedback from geometric planning failures or predicted geometric infeasibility into the LLM to iteratively refine the solution. Although, these methods require large resources for running LLMs.

2.3 Action And Grasp Feasibility Prediction for Manipulation Planning

Action and grasp feasibility prediction is a fast growing research topic, which aims at accelerating common TAMP algorithms by reducing their dependency on the geometric planner. Wells et al. (2019) pioneered this problem by proposing to use Support Vector Machines (SVM) as action and grasp feasibility classifiers. Though, this method is limited to simple environments with a fixed number of objects. To tackle this issue, Driess et al. (2020b;a); Xu et al. (2022) propose to use top-view depth images as input to a convolutional neural network (CNN). However, this approach is limited

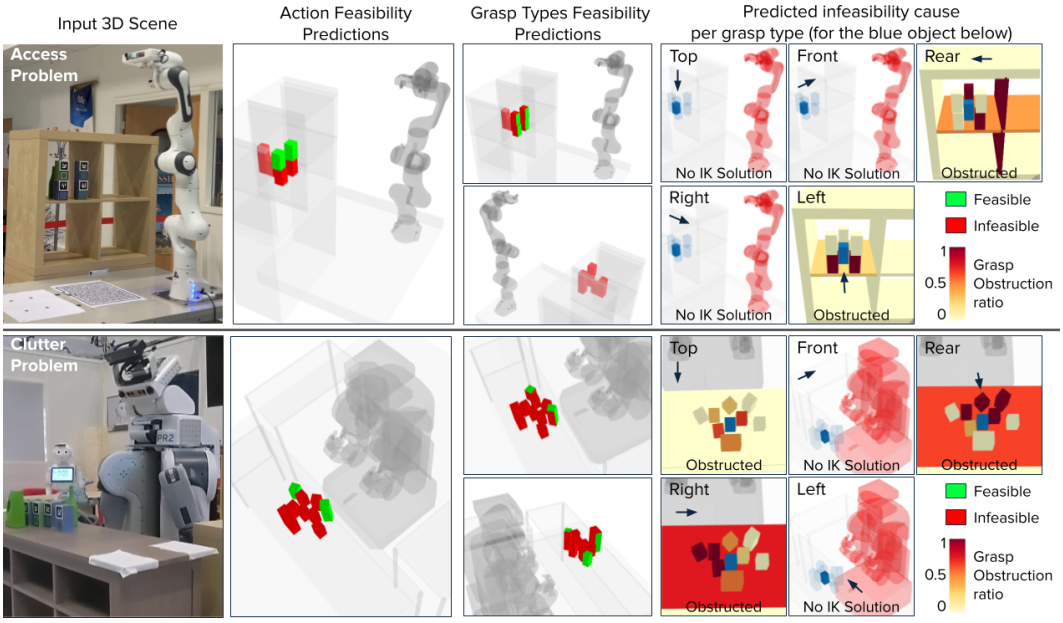


Figure 1: Visualization of **GRN** predictions on two manipulation problems, **Access** (Panda arm) and **Clutter** (PR2 Robot, predictions shown for its right arm). A single query to **GRN** outputs 3 predictions for each movable object in the environment: Action feasibility, grasp types feasibility (two views), and the predicted infeasibility causes for each grasp type. For clarity, we show the predicted infeasibility cause for one object only (shown in blue), and distinguish two cases: (1) *No IK solution*: robot shown in red, (2) *Grasp type obstructed*: we show all obstructing objects in a color gradient representing the obstructions ratio. Arrows show approach directions of grasp types.

to tabletop problems. Bouhsain et al. (2023a;b) generalize this method to 3D environments by using 5 depth images from different scene views. Bouhsain et al. (2024) extend this method to mesh objects and multi-robot problems. Yang et al. (2022) also use multiple views, combined with text descriptions of actions and predicates as input to a transformer. Such image-based representations suffer from occlusions, which can hurt feasibility prediction accuracy. Some methods use 3D voxel grids (Park et al., 2022) or pointclouds (Huang et al., 2024a) to represent 3D scenes. Though, they suffer from high inference times and are, hence, limited to environments with few objects. Other works such as Kim et al. (2022); Khodeir et al. (2023a;b); Sung et al. (2023) tackle this issue by representing 3D environments using scene graphs, leveraging GNNs to predict the success of geometric planning steps from prior search experiences. These methods, however, lack interpretability and can not provide feedback on why actions are infeasible.

3 Problem Description

In this work, we address the problem of predicting action and grasp feasibility for offline manipulation planning in 3D environments. The goal is to determine whether a robot can successfully plan actions, precisely picking or placing objects, and which grasps allow their collision-free motions, while accounting for inverse kinematics constraints and potential grasp obstructions.

3.1 Offline Manipulation Planning Context

Our approach is tailored for offline manipulation planning tasks in 3D environments, containing fixed objects and movable objects. We assume the shape, dimensions, and pose of all objects are fully known, and that all objects remain static unless moved by the robot. A pick action involves grasping a movable object, while a place action refers to placing it at a target pose. From a motion planning perspective, these actions are symmetrical (Wells et al., 2019; Bouhsain et al., 2023b). Assuming they start and end at the same home configuration of the robot, a place action at a specific

pose is the reversed pick action from that pose. Throughout this work, the term "action" refers to both pick and place actions. Planning an action first requires sampling grasps. For each grasp, an inverse kinematics (IK) solver computes the corresponding robot configurations. If one exists, it is checked for collisions before a motion planner computes the robot's full trajectory. This process involving grasp sampling, IK solving, collision checking, and motion planning is computationally expensive and incurs significant overhead during TAMP, especially in cluttered 3D environments.

We define $\mathcal{G} = \{Top, Front, Rear, Right, Left\}$ as a set of 5 grasp types, each one representing a subspace of grasps related to the side from which the object is grasped. We focus on axis-aligned grasps, such that each type is the continuous set of grasps for which the end-effector's axis is parallel to one of the object's principal axes in a specific direction. This representation is similar to those introduced in previous works (Wells et al., 2019; Driess et al., 2020b;a; Bouhsain et al., 2023a;b).

3.2 Feasibility Prediction

This work aims to reduce the dependency on the complex geometric planning process involved in offline manipulation planning. Given an environment E and an object of interest $\mathcal{O} \in E$, the goal is to predict both the feasibility $F_a(\mathcal{O}, E) \in \mathbb{R}$ of picking or placing $\mathcal{O}$ at its pose in E, and the feasibility of each grasp type, $\mathbf{F}_{\mathcal{G}} = [F_g(\mathcal{O}, E), \forall g \in \mathcal{G}] \in \mathbb{R}^5$.

Moreover, we aim to estimate the cause of infeasibility for each grasp type. We focus on two primary factors contributing to infeasibility: (1) the absence of a valid inverse kinematics (IK) solution for all grasps within a grasp type $\boldsymbol{\kappa}_{\mathcal{G}} = [\kappa_g(\mathcal{O}, E), \forall g \in \mathcal{G}] \in \mathbb{R}^5$, and (2) the obstruction of grasps by each neighboring objects, represented as the ratio $\boldsymbol{\rho}_{\mathcal{G}}$ of obstructed grasps:

$$\boldsymbol{\rho}_{\mathcal{G}} = [\rho_g(\mathcal{O}, \mathcal{O}'), \quad \forall g \in \mathcal{G}, \quad \forall \mathcal{O}' \in \mathcal{N}(\mathcal{O})] \quad \in \mathbb{R}^{5 \times |\mathcal{N}(O)|} \tag{1}$$

where $\mathcal{N}(\mathcal{O})$ denotes the distance-based neighborhood of $\mathcal{O}$ and $|\mathcal{N}(O)|$ its cardinality. In addition to helping explain infeasibility and providing insights into the constraints imposed by the environment, these predictions are also used in the feasibility prediction process. In summary, the task at hand is to learn two classification functions f_F, f_κ, and a regression function f_ρ s.t.:

$$\begin{bmatrix} F_a \\ \boldsymbol{F}_{\mathcal{G}} \end{bmatrix} = f_F(\mathcal{O}, E, \boldsymbol{\kappa}_{\mathcal{G}}, \boldsymbol{\rho}_{\mathcal{G}}) \qquad \text{where} \qquad \boldsymbol{\kappa}_{\mathcal{G}} = f_\kappa(O, E) \quad \text{and} \quad \boldsymbol{\rho}_{\mathcal{G}} = f_\rho(O, E) \tag{2}$$

4 Geometric Reasoning Networks

We propose **Geometric Reasoning Networks (GRN)**, a three-module GNN-based neural network which takes as input a graph representation of the environment, and outputs the action and grasp types feasibility for each movable object, as well as inverse kinematics feasibility (IK) and grasp obstruction (GO) predictions (cf. Figure 1). The main challenge of learning methods in a manipulation planning context is finding an appropriate representation of the 3D environment which can contain an arbitrary number of objects. Previously proposed image-based representations suffer from occlusions as the number of objects in the environment increases, which can significantly impact performance. In this paper, we tackle this issue by representing 3D scenes as graphs where nodes represent fixed and movable objects and edges represent geometric relationships between objects.

4.1 3D Scene Representation

Given an enviroment E, we construct a directed graph $(\mathcal{V}, \mathcal{E})$ where each node corresponds to an object in E, fixed or movable. Nodes have a feature vector $\mathbf{x} = [l, w, h, x, y, z, \theta]^T$ where (l, w, h) are the length, width and height of the object's bounding box, (x, y, z) represent the position of the object in the environment, and θ is the object's orientation w.r.t its z axis. The position and orientation of the object are expressed in the reference frame of the base of the robot. This allows a straightforward generalization to multi-robot settings simply by switching the reference frame to the base of the robot of interest. For each node u corresponding to a movable object $\mathcal{O}$, we add a self-loop edge $(u \to u) \in \mathcal{E}$, as well as a directed edge $(v \to u) \in \mathcal{E}$ from each node v corresponding to a neighboring (fixed or movable) object $\mathcal{O}' \in \mathcal{N}(\mathcal{O})$. Two objects are considered neighbors if the euclidian distance between their closest points along the (x, y) plane is lower than a user-defined threshold. Since our approach is GNN-based, where edges are used to determine

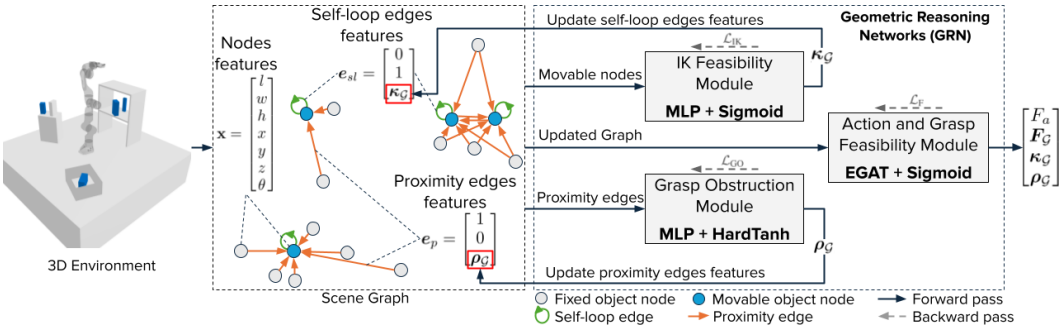


Figure 2: **Complete GRN architecture**. A scene graph is constructed from the input 3D environment. Node features of movable objects are given to IK feasibility module which outputs are used to update self-loop edge features. The concatenated features of nodes linked through a proximity edge are fed to the GO module to get grasp obstruction estimations, which are appended to proximity edge features. Finally, the updated graph is given to the AGF module to predict the action and grasp types feasibility for each movable object in the environment.

which features to aggregate, the more edges are present in the input graph, the higher the number of computations is. Hence, the permissiveness of this threshold creates a tradeoff between accuracy and inference speed. Finally, self-loop edges and proximity edges are differentiated using a one-hot encoded feature vector such that $\boldsymbol{e}_{sl} = [0, 1]^T$ and $\boldsymbol{e}_p = [1, 0]^T$, where $\boldsymbol{e}_{sl}$ and $\boldsymbol{e}_p$ represent the feature vectors for a self-loop edge and a proximity edge respectively. The obtained graph is given as input to **GRN**, shown in Figure 2, which is comprised of three modules.

4.2 Inverse Kinematics Feasibility Prediction

The first module is the Inverse Kinematics (IK) feasibility prediction module. This submodel is a binary classifer, composed of a 4-layer Multi-Layer Perceptron (MLP) with ReLU activation functions, followed by a Sigmoid activation function. It takes as input the feature vector $\mathbf{x}_u$ of each node $u \in \mathcal{V}$ corresponding to a movable object, and simultaneously outputs the predicted inverse kinematics feasibility for each of the five grasp types in $\mathcal{G}$:

$$\boldsymbol{\kappa}_{\mathcal{G}}(u) = \text{Sigmoid}(\mathbf{MLP}_{\text{IK}}(\mathbf{x}_u)) \quad \in [0, 1]^5 \tag{3}$$

Each prediction κ_g where $g \in \mathcal{G}$ corresponds to the presence of a valid inverse kinematics solution for at least one grasp in g. This can be viewed as predicting whether the robot can reach a specific side of the object at its pose in the environment, without taking into account any other object. Since this information is valuable for action and grasp types feasibility prediction, we incorporate it to the previously constructed graph by concatenating the obtained predictions $\boldsymbol{\kappa}_{\mathcal{G}}$ to the features of self-loop edges resulting in $\boldsymbol{e}_{sl}^{+} = [\boldsymbol{e}_{sl} \parallel \boldsymbol{\kappa}_{\mathcal{G}}] \in \mathbb{R}^7$, where $\parallel$ denotes the concatenation operator.

4.3 Grasp Obstruction Estimation

The second stage is Grasp Obstruction (GO) estimation, which is a regression module consisting of 4-layer MLP with ReLu activation functions, followed by a HardTanh activation function which bounds the output between 0 and 1. It takes as input the concatenated features of each pair of nodes $(v \to u)$ linked through a proximity edge. For each grasp type $g \in \mathcal{G}$, it outputs the estimated ratio of grasps of u that are obstructed by the object corresponding to v, which is defined as the number of obstructed grasps divided by the total number of grasps of a specific type:

$$\boldsymbol{\rho}_{\mathcal{G}}(u, v) = \text{HardTanh}(\mathbf{MLP}_{\text{GO}}([\boldsymbol{x}_u \parallel \boldsymbol{x}_v])) \quad \in [0, 1]^5 \tag{4}$$

This submodel not only predicts whether an object obstructs grasps of another, it also gives insight into how much it blocks access to it. In a manipulation planning context, this allows to rank obstructing objects and plan accordingly. Similarly to IK feasibility predictions, this information impacts directly the feasibility of actions and grasp types. Thus, we concatenate the grasp obstruction estimations $\boldsymbol{\rho}_{\mathcal{G}}$ to the edge features of proximity edges to obtain $\boldsymbol{e}_p^{+} = [\boldsymbol{e}_p \parallel \boldsymbol{\rho}_{\mathcal{G}}] \in \mathbb{R}^7$.

4.4 Action and Grasp Feasibility Prediction

Once our graph is constructed and enriched through IK feasibility and GO predictions, it is given to the action and grasp feasibility (AGF) prediction module. As mentioned previously, IK feasibility and grasp obstructions, which constitute our edge features, have a direct impact on action and grasp type feasibility. However, the classic graph attention network (GAT) (Veličković et al., 2018) uses edge features simply for computing attention weights, and does not take them into account during the aggregation process. This causes the model to lose valuable information when applied to tasks where edge features are as important as node features. In this work, we propose to use the Edge-Featured Graph Attention Network (EGAT) (Wang et al., 2021) which, in addition to using edge features to compute attention, leverages both node and edge features during the aggregation process.

The third submodel first computes embeddings $\boldsymbol{h}_u$ and $\boldsymbol{e}^*_{\boldsymbol{uv}}$ of node and edge features using two fully-connected layers such that $\boldsymbol{h}_u = \boldsymbol{W_h}.\boldsymbol{x}_u$ and $\boldsymbol{e}^*_{\boldsymbol{uv}} = \boldsymbol{W_e}.\boldsymbol{e}^+_{uv}$, where $\boldsymbol{W_h}$, $\boldsymbol{W_e}$ are weight matrices. Then, EGAT computes the multi-head attention of each edge as:

$$\alpha_{uv} = \frac{\exp(\boldsymbol{a}^T \text{LeakyReLU}([\boldsymbol{h}_u \parallel \boldsymbol{h}_v \parallel \boldsymbol{e}^*_{uv}]))}{\sum_{k \in \mathcal{N}(u) \cup \{u\}} \exp(\boldsymbol{a}^T \text{LeakyReLU}([\boldsymbol{h}_u \parallel \boldsymbol{h}_k \parallel \boldsymbol{e}^*_{uk}]))} \tag{5}$$

Note that this formulation is slightly different from the one proposed by Wang et al. (2021). We adapt the attention computation introduced by Brody et al. (2022), which fixes the static attention problem of the standard GAT. Once the multi-attention weights computed, our model computes a weighted average of the concatenated node and edge embeddings, followed by a LeakyReLU activation:

$$\boldsymbol{h}'_u = \text{LeakyReLU}\left(\sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} [\boldsymbol{h}_v \parallel \boldsymbol{e}^*_{uv}]\right) \tag{6}$$

Finally, the obtained vector is passed through a 2-layer MLP followed by a sigmoid activation function that outputs the action and grasp types feasibility predictions for movable object node u:

$$\begin{bmatrix} F_a(u) \\ \boldsymbol{F}_{\mathcal{G}}(u) \end{bmatrix} = \text{Sigmoid}(\mathbf{MLP}_F(\boldsymbol{h}'_u)) \tag{7}$$

4.5 Training Strategy

Our proposed **GRN** model is trained in a supervised manner by first pre-training each module separately. The IK feasibility prediction submodel is trained using a binary cross entropy loss denoted $\mathcal{L}_{\text{IK}}$ while the GO estimation module is trained using the mean square error loss $\mathcal{L}_{\text{GO}}$. Regarding the AGF prediction module, it is trained using a binary cross entropy loss $\mathcal{L}_{\text{F}}$, using the ground truth of IK feasibility and grasp obstructions as edge features. The complete **GRN** model is then fine-tuned using a weighted sum of the previously defined losses such that:

$$\mathcal{L} = \mathcal{L}_{\text{F}} + \mathcal{L}_{\text{IK}} + \eta.\mathcal{L}_{\text{GO}} \tag{8}$$

where η is a weighting factor allowing a balanced order of magnitude across classification and regression. This training strategy is inspired by the one proposed by Chen et al. (2020).

During training, our 3D scene representation allows two data augmentation methods: dimensions switch and rotation. They both take advantage of the symmetry of bounding boxes. Indeed, switching the length and width of a bounding box, then applying a $\frac{\pi}{2}$ rotation around the z axis, keeps the geometric properties of the environment unchanged, resulting in a new node feature vector $\boldsymbol{x} = [w, l, h, x, y, z, \theta \pm \frac{\pi}{2}]^T$. The same goes for applying a π rotation without any dimensions switch such that $\boldsymbol{x} = [l, w, h, x, y, z, \theta \pm \pi]^T$. These can be applied to fixed and movable objects. For the latter case, the labels associated with grasp types need to be interchanged, since switching the dimensions and/or rotating the object changes which side is considered as the front for example.

5 Experiments

We conduct a series of experiments in order to evaluate the performance of our proposed method compared to existing approaches, and showcase the generalization capabilities of our approach. We

conduct our experiments using mainly the Franka Emika Panda, which is a 7 degrees-of-freedom (DOF) robotic arm with a parallel jaw gripper. Since our model is robot-centric, meaning that it is specific to the robot it is trained for, we showcase the applicability of our method to other robotic manipulators by running experiments on the Willow Garage PR2 robot as well. The latter is a dual-arm robot where each arm has 7 DOFs and a parallel jaw gripper, and a telescoping spine.

5.1 DATASETS

Our dataset is trained and evaluated on fully synthetic data. We generate a number of datasets following the method described in Appendix B. Each one characterized by a number of movable objects as well as a minimum and maximum numbers of structures (e.g, rack, counter, basket) and obstacles. They are also characterized by the robot used during data annotation.

Panda-3D-4: This is dataset is composed of 3D environments containing 4 movable objects, 1 to 4 structures and 0 to 4 obstacles and is annotated using a Panda robot.

Panda-Tabletop-4: In order to conduct a fair comparison to tabletop methods Wells et al. (2019); Driess et al. (2020b), we generate a dataset consisting of tabletop environments with 0 structures, 4 movable objects and up to 4 obstacles, all placed on the same support surface as the robot's base.

PR2-3D-4: We generate this dataset to showcase the applicability of our approach to other robotic arms as well as multi-robot settings. It is generated using the same parameters as the **Panda-3D** dataset, and annotated for the right arm of PR2 robot[1]. We consider that the base of robot is fixed to the ground, and that the telescopic spine is one of the DOFs of the arms.

In order to quantitatively measure the generalizability of our approach to environments containing a higher number of objects than training environments, we generate three additional test sets denoted **Panda-3D-10**, **Panda-3D-15**, **Panda-3D-20**, each composed of 1'000 environments containing 10, 15 and 20 movable objects respectively. These environments also contain a higher number of fixed objects, with a number of structures ranging from 4 to 8, as well as 2 to 4 obstacles. Moreover, we increase the dimensions' range used during data generation resulting in larger objects.

5.2 BASELINES

We compare our proposed approach to multiple baselines and methods proposed or adapted from previous works on action and grasp feasibility prediction for manipulation planning.

MLP: This is a simple baseline which uses a 4-layer MLP that takes as input the feature vector $\boldsymbol{x}$ of an object to predict action and grasp feasibility, without considering the rest of the environment.

Feasibility-SVM (F-SVM): Introduced by Wells et al. (2019), this method uses multiple SVMs to predict the action and grasp types feasibility prediction in tabletop environments containing 2 movable objects represented using hand-crafted feature vectors.

Deep Visual Heuristics (DVH): This method, proposed by Driess et al. (2020b), represents environments using top-view depth images, then uses a CNN to predict the feasibility of an action using a grasp type. We adapt **DVH** to output the feasibility of the action and all grasp types simultaneously.

Action and Grasp Feasibility Prediction Network (AGFP-Net): Bouhsain et al. (2023b) propose this method as an extension of **DVH** to 3D environments, using 5 depth images corresponding to different views of the scene as input to the CNN.

Feasibility-GAT (F-GAT): This baseline is an adapted version of the methods proposed by Silver et al. (2021) and Khodeir et al. (2023a;b). It represents environments as graphs where nodes represent objects, and edges represent symbolic relationships (e.g object on table). Nodes features are the dimensions and pose of objects and edge features are one-hot-encodings of the different relationship types. GAT is then used to predict action and grasp feasibility.

Feasibility-GCN (F-GCN): This baseline uses the same scene representation as **F-GAT**, except that GAT is replaced with a Graph Convolution Network (GCN), which does not use edge features.

[1]As explained is Section 4.1, our model can be applied to different arms of the same type simply by expressing the objects' poses in the frame of reference of the considered arm.

Table 1: Comparison with SOTA methods trained and tested on different datasets. For grasp types feasibility prediction, the mean (± standard deviation) of F1 scores of the 5 grasp types are reported.

Dataset	Panda-3D-4		Panda-Tabletop-4		PR2-3D-4	
Task	**Action (F1)**	**Grasp (F1)**	**Action (F1)**	**Grasp (F1)**	**Action (F1)**	**Grasp (F1)**
F-SVM	-	-	0.884	0.415 (± 0.220)	-	-
MLP	0.784	0.558 (± 0.089)	0.911	0.696 (± 0.121)	0.750	0.574 (± 0.104)
DVH	0.840	0.718 (± 0.108)	0.961	0.865 (± 0.073)	0.808	0.622 (± 0.179)
AGFPNet	0.882	0.806 (± 0.065)	0.964	0.916 (± 0.032)	0.836	0.655 (± 0.238)
F-GCN	0.836	0.721 (± 0.057)	0.955	0.879 (± 0.042)	0.791	0.680 (± 0.075)
F-GAT	0.867	0.796 (± 0.052)	0.961	0.904 (± 0.034)	0.827	0.764 (± 0.061)
GRN (Ours)	**0.939**	**0.940 (± 0.009)**	**0.976**	**0.976 (± 0.004)**	**0.908**	**0.903 (± 0.013)**

5.3 Evaluation Metrics

We use the **F1** score as the evaluation metric for classification tasks, namely action, grasp types and IK feasibility predictions. For or Grasp Obstruction (GO) predictions, we use the **Mean Absolute Error** to measure performance. For clarity, we report the mean and standard deviation of predictions across the different grasp types.

6 Results

6.1 Comparison to Prior Work

Table 1 shows that our proposed model outperforms all prior works on both action feasibility and grasp types feasibility predictions, and on all datasets. CNN-based methods, **DVH** and **AGFP-Net**, fall short compared to our approach, with a difference in F1 score on the **Panda-3D-4** of 10% (resp. 5.7%) for action feasibility prediction, and 22.6% (resp. 13.7%), for grasp type feasibility prediction. Indeed, image-based scene representation suffers from occlusions due to the 3D nature of the environment, resulting in inaccurate predictions for occluded objects. GNN-based methods, on the other hand, represent 3D environments using scene graphs. However, the performance of our approach compared to **F-GCN** and **F-GAT** shows that a careful design of the graph and its connectivity is needed. Silver et al. (2021) and Khodeir et al. (2023a;b) connect nodes using symbolic facts. Our method uses geometric relationships between objects to connect the graph. This allows our model to achieve an F1 score up to 10.3% higher than other GNN-based baselines on action feasibility prediction, and up to 21.8% higher on grasp types feasibility prediction on **Panda-3D-4**.

Comparing the standard deviations across F1 scores of each grasp type shows that our proposed method has a more consistent performance across the different grasp types than other models. This is due to our data augmentation method and the two interpretation mechanisms. Switching the dimensions then/or rotating an object implies interchanging grasp types annotations, which ensures balanced labels across grasp types. This method is not applicable to CNN-based methods [2]. Additionally, the use of IK feasibility predictions and GO estimations as edge features allows each grasp type feasibility prediction to be informed, which is not the case for other GNN-based methods.

In robotic manipulation planning, feasibility prediction must not only be accurate, it must also have a low inference time and memory footprint. Table 2 reports the number of parameters in previous models compared to **GRN**, as well as the inference time of each model on a 3D scene containing a Panda robot, 4 movable objects and 15 fixed object. The inference time incorporates the complete prediction process from the model's input construction to the output, for each movable object in the environment. For reference, we also report the planning time of an off-the-shelf geometric planner.

Table 2: Comparison of the number of parameters and inference time on a 3D environment with 4 movable objects and 15 fixed objects (4 queries).

Model	Inference time (ms)	Nb Parameters
Geometric planner	1500	-
MLP	0.6	269'830
DVH	25	11'225'282
AGFPNet	150	34'585'350
F-GCN	4.25	1'057'798
F-GAT	5.0	2'636'806
GRN (Ours)	5.5	2'259'472

[2] Switching the dimensions then rotating an object results in the same input images.

Table 3: Ablation Study on the **Panda-3D-4** dataset. For each task related to grasp types, we report the mean ($\pm$ standard deviation) across all grasp types.

Task	Action (F1) ↑	Grasp (F1) ↑	IK (F1) ↑	GO (MAE) ↓
w/o IK, w/o GO	0.868	0.811 ($\pm$ 0.043)	-	-
w/o GO	0.872	0.811 ($\pm$ 0.046)	**0.995 ($\pm$ 0.001)**	-
w/o IK	0.937	0.933 ($\pm$ 0.011)	-	0.029 ($\pm$ 0.003)
Full model w/ GAT	0.928	0.924 ($\pm$ 0.013)	**0.995 ($\pm$ 0.001)**	0.029 ($\pm$ 0.003)
Full model w/o data aug.	0.915	0.903 ($\pm$ 0.013)	0.990 ($\pm$ 0.001)	0.044 ($\pm$ 0.002)
Full model (Trained from scratch)	0.932	0.925 ($\pm$ 0.011)	0.994 ($\pm$ 0.001)	0.038 ($\pm$ 0.002)
Full model (Ours)	**0.939**	**0.939 ($\pm$ 0.009)**	**0.995 ($\pm$ 0.001)**	**0.028 ($\pm$ 0.003)**

Results show that, compared to previous works, our method yields the most accurate predictions while being one the models with the lowest inference times and memory footprints. Furthermore, **GRN** has a 99.6% lower inference time than traditional geometric planning. In an offline manipulation planning context, where the number of feasibility checks can reach tens of thousands of queries, this difference in computational cost can significantly reduce planning time.

6.2 Ablation Study

In order to justify the choices behind our neural network architecture and training strategy, we conduct an ablation study on the **Panda-3D-4** dataset. Results reported in Table 3 showcase the importance of the proposed interpretation modules. Our full model shows a 7.1% gain in performance compared to the one without IK feasibility and GO predictions. A more in-depth analysis shows that the grasp obstruction estimation module is the most important, while IK feasibility prediction yields a slight improvement in performance. Moreover, the improved performance obtained using EGAT instead of classical GAT Veličković et al. (2018) shows that incorporating edge features in the attention computation and the aggregation process helps action and grasp types feasibility prediction. We conduct two ablations to demonstrate the effectiveness of our training strategy. Train the model without the proposed data augmentation method, yields a lower performance on all tasks, particularly on grasp types feasibility prediction and GO estimation with a 3.6% difference in F1 score, and 1.6 % in MAE. Finally, training the full model from scratch, rather than pre-training each module before fine-tuning the complete network, yields a slightly lower performance across all tasks.

6.3 Generalizability Evaluation

Applicability to other robots. Table 1 shows our model's performance on the **PR2-3D-4** compared to prior methods. The results show that **GRN** achieves a better performance than the state-of-the-art on robots with various kinematics. A consistent decrease in performance can be noticed across all methods (ours included), compared to when trained on the **Panda-3D-4** dataset. This is due to the smaller number of training data of the PR2 dataset and the harder kinematics of the PR2 robot.

Generalizability to more complex environments. Table 4 reports the performance of our method and the baselines on the **Panda-3D-10**, **Panda-3D-15** and **Panda-3D-20** test sets, when trained on the **Panda-3D-4** dataset. Results show that, although there is a decrease in F1-scores compared to the one obtained on **Panda-3D-4**, **GRN** maintains a good performance on 3D environments with a higher number of fixed and movable objects, with an F1-score on **Panda-3D-20** of 0.89 for action feasibility prediction, and 0.903 for grasp types feasibility prediction. Particularly, our model shows a better generalization capability on the latter than previous CNN-based or GNN-based methods.

6.4 Application to Task and Motion Planning

Although the integration of our method into a task and motion planner is outside the score of this paper, we develop a simple planner (cf. Appendix C.1) to showcase the planning performance gain yielded by our approach. From a single query to **GRN**, it proposes a geometrically feasible plan to a class of manipulation planning problems, where the goal is to move a single object initially placed in a complex setting. We compare this algorithm to the non-informed TAMP planner proposed by Bouhsain et al. (2024) on two different problems shown in Figure 1. The first is the **Access** problem,

Table 4: Evaluation of the generalizability to 3D environments with a higher number of objects compared to SOTA methods, when trained on the **Panda-3D-4** dataset.

Test set	**Panda-3D-10**		**Panda-3D-15**		**Panda-3D-20**	
Task	**Action (F1)**	**Grasp (F1)**	**Action (F1)**	**Grasp (F1)**	**Action (F1)**	**Grasp (F1)**
MLP	0.773	0.624 (± 0.028)	0.766	0.616 (± 0.046)	0.768	0.609 (± 0.047)
DVH	0.820	0.697 (± 0.115)	0.819	0.686 (± 0.127)	0.825	0.676 (± 0.140)
AGFPNet	0.858	0.770 (± 0.079)	0.862	0.768 (± 0.078)	0.864	0.755 (± 0.087)
F-GCN	0.794	0.633 (± 0.036)	0.771	0.595 (± 0.044)	0.764	0.565 (± 0.055)
F-GAT	0.829	0.738 (± 0.026)	0.826	0.725 (± 0.032)	0.825	0.715 (± 0.038)
GRN (Ours)	**0.894**	**0.909 (± 0.012)**	**0.891**	**0.906 (± 0.013)**	**0.890**	**0.903 (± 0.012)**

Table 5: Performance of **GRN** planner compared to a non-informed planner on the **Access** and **Clutter problems**. Results are average over 10 runs on 10 different instances of each problem.

Problem	**Method**	**Success Rate (%)**	**Planning time (s)**	**Nb Geometric Planner Calls**
Access	Bouhsain et al. (2024)	100%	26.5	41.1
	GRN planner	90%	3.17	6
Clutter	Bouhsain et al. (2024)	100%	558.9	89.5
	GRN planner	100%	14.8	7.2

where a single Panda robot has to move a small bottle, to which a number of fixed and movable objects block access. The second is a multi-robot **Clutter** problem, in which both arms of the PR2 robot can collaborate to pick an object surrounded by grasp-obstructing objects. Objects used in these problems are a mix of box-shaped and mesh objects. Table 5 shows that despite its simplicity, the **GRN**-based planner not only achieves a 90% and 100% success rate but also reduces planning time by 88% on the **Access** problem, and 97% on the **Clutter** problem. This is possible thanks to the two interpretation mechanisms, which allow the planner to reason over why an action or a grasp type is infeasible, and recursively decide which objects should be moved to rectify that. We also test **GRN** planner in real-world setups, on both a Panda and a PR2 robots. Objects' poses are estimated using a separate perception module. The proposed model is able to accurately predict action and grasp feasibility, as well as reasons of infeasibility, from estimated objects' poses in both setups, allowing the planner to compute feasible solutions that are successfully executed by the robots.

7 Discussion and Future Work

In this work, we propose a framework for action and grasp feasibility prediction in 3D environments. Leveraging a GNN-based neural network and two interpretation mechanisms, our model predicts the feasibility of pick or place actions, different grasp types, and the infeasibility cause for each grasp type from a scene graph representation. Results demonstrate that our approach outperforms state-of-the-art methods, generalizing better to complex environments and robots. Additionally, through IK feasibility and grasp obstruction predictions, a class of manipulation planning problems can be solved with a single query to our model, significantly reducing planning time compared to traditional TAMP planners. Our design also enables straightforward extensions to diverse object shapes via bounding boxes and multi-robot feasibility predictions due to the robot-centric nature of **GRN**.

Currently, our method is limited to infeasibility caused by missing IK solutions or grasp obstructions, and it struggles with problems stemming from infeasible robot motion. Future work will include graph pooling layers to evaluate motion infeasibility across the entire scene graph. Additionally, grasp obstructions are currently estimated as a blocked ratio, which does not indicate which grasps are obstructed. Representing obstructions as blocked object regions could improve interpretability. While bounding boxes work well for common objects, they face challenges with large or complex shapes. A potential extension is using multi-bounding box representations, treating sub-boxes as separate nodes. Alternatively, embeddings from off-the-shelf shape encoders (e.g., meshes or point clouds) could enhance node features, though computational efficiency remains a challenge.

REFERENCES

Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7951–7958. IEEE, 2023.

Juan Manuel Ahuactzin, Kamal Gupta, and Emmanuel Mazer. Manipulation planning for redundant robots: a practical approach. *The International Journal of Robotics Research*, 1998.

Rachid Alami, Thierry Simeon, and Jean-Paul Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In *The fifth international symposium on Robotics research*, 1990.

Smail Ait Bouhsain, Rachid Alami, and Thierry Simeon. Learning to predict action feasibility for task and motion planning in 3d environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023a.

Smail Ait Bouhsain, Rachid Alami, and Thierry Simeon. Simultaneous action and grasp feasibility prediction for task and motion planning through multi-task learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023b.

Smail Ait Bouhsain, Rachid Alami, and Thierry Simeon. Extending task and motion planning with feasibility prediction: Towards multi-robot manipulation planning of realistic objects. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Stephane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 2009.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pp. 66–75. PMLR, 2020.

Haonan Chen, Yilong Niu, Kaiwen Hong, Shuijing Liu, Yixuan Wang, Yunzhu Li, and Katherine Rose Driggs-Campbell. Predicting object interactions with behavior primitives: An application in stowing tasks. In *7th Annual Conference on Robot Learning*, 2023.

Rohan Chitnis, Dylan Hadfield-Menell, Abhishek Gupta, Siddharth Srivastava, Edward Groshev, Christopher Lin, and Pieter Abbeel. Guided search for task and motion plans using learned heuristics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and systems*, volume 12, pp. 00052. Ann Arbor, MI, USA, 2016.

Didier Devaurs, Thierry Siméon, and Juan Cortés. Enhancing the transition-based rrt to deal with complex cost spaces. In *2013 IEEE international conference on robotics and automation*, pp. 4120–4125. IEEE, 2013.

Danny Driess, Jung-Su Ha, and Marc Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Robotics: Science and Systems 2020*, 2020a.

Danny Driess, Ozgur Oguz, Jung-Su Ha, and Marc Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020b.

Ben Eisner, Yi Yang, Todor Davchev, Mel Vecerik, Jonathan Scholz, and David Held. Deep se (3)-equivariant geometric reasoning for precise placement tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Ffrob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, 2018.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.

Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. Moveit! task constructor for task-level motion planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 190–196. IEEE, 2019.

Dan Haramati, Tal Daniel, and Aviv Tamar. Entity-centric reinforcement learning for object manipulation from pixels. In *The Twelfth International Conference on Learning Representations*, 2024.

Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 2010.

Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, 2011.

Haojie Huang, Owen Lewis Howell, Dian Wang, Xupeng Zhu, Robert Platt, and Robin Walters. Fourier transporter: Bi-equivariant robotic manipulation in 3d. In *The Twelfth International Conference on Learning Representations*, 2024a.

Yixuan Huang, Adam Conkey, and Tucker Hermans. Planning for multi-object manipulation with graph neural network relational classifiers. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1822–1829. IEEE, 2023.

Yixuan Huang, Christopher Agia, Jimmy Wu, Tucker Hermans, and Jeannette Bohg. Points2plans: From point clouds to long-horizon plans with composable relational dynamics. *arXiv preprint arXiv:2408.14769*, 2024b.

Mohamed Khodeir, Ben Agro, and Florian Shkurti. Learning to search in task and motion planning with streams. *IEEE Robotics and Automation Letters*, 2023a.

Mohamed Khodeir, Atharv Sonwane, Ruthrash Hari, and Florian Shkurti. Policy-guided lazy search with feedback for task and motion planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3743–3749. IEEE, 2023b.

Beomjoon Kim, Luke Shimanuki, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Representation, learning, and planning algorithms for geometric task and motion planning. *The International Journal of Robotics Research*, 2022.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yoshihito Koga and J-C Latombe. On multi-arm manipulation planning. In *1994 IEEE International Conference on Robotics and Automation*, 1994.

Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 2014.

Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.

Yixin Lin, Austin S Wang, Eric Undersander, and Akshara Rai. Efficient and interpretable robot manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):2740–2747, 2022.

Michael James McDonald and Dylan Hadfield-Menell. Guided imitation of task and motion planning. In *Conference on Robot Learning*, 2022.

Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866. IEEE, 2012.

Suhan Park, Hyoung Cheol Kim, Jiyeong Baek, and Jaeheung Park. Scalable learned geometric feasibility for cooperative grasp and motion planning. *IEEE Robotics and Automation Letters*, 2022.

Guillaume Sarthou. Overworld: Assessing the geometry of the world for human-robot interaction. *IEEE Robotics and Automation Letters*, 8(3):1874–1880, 2023.

Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2021.

Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 2004.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, 2014.

Yoonchang Sung, Zizhao Wang, and Peter Stone. Learning to correct mistakes: Backjumping in long-horizon task and motion planning. In *Conference on Robot Learning*, pp. 2115–2124. PMLR, 2023.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. Llmˆ 3: Large language model-based task and motion planning with motion failure reasoning. *Multi-modal Foundation Model meets Embodied AI Workshop in the 41 st International Conference on Machine Learning*, 2024.

Ziming Wang, Jun Chen, and Haopeng Chen. Egat: Edge-featured graph attention network. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part I 30*, pp. 253–264. Springer, 2021.

Andrew M Wells, Neil T Dantam, Anshumali Shrivastava, and Lydia E Kavraki. Learning feasibility for task and motion planning in tabletop environments. *IEEE robotics and automation letters*, 2019.

Danfei Xu, Ajay Mandlekar, Roberto Martín-Martín, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Deep affordance foresight: Planning through what can be done in the future. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Lei Xu, Tianyu Ren, Georgia Chalvatzaki, and Jan Peters. Accelerating integrated task and motion planning with neural feasibility checking. *arXiv preprint arXiv:2203.10568*, 2022.

Zhutian Yang, Caelan Reed Garrett, and Dieter Fox. Sequence-based plan feasibility prediction for efficient task and motion planning. *arXiv preprint arXiv:2211.01576*, 2022.

Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6541–6548. IEEE, 2021.

A IMPLEMENTATION DETAILS

Scene Graph Construction. Scene graphs are built using the graph representation from Pytorch Geometric (Fey & Lenssen, 2019). Nodes correspond to fixed and movable objects from the environment. Self-edge loops are added for nodes corresponding to movable objects. When adding proximity edges, since evaluating the distance between two objects closest points can be computationally expensive, we consider two objects $\mathcal{O}$ and $\mathcal{O}'$ as neighbors if the euclidian distance between their centers is lower than a threshold $\epsilon = r_{\mathcal{O}} + r_{\mathcal{O}'} + K$, where $r_{\mathcal{O}}$ and $r_{\mathcal{O}'}$ are the bounding cylinder radii of $\mathcal{O}$ and $\mathcal{O}'$ respectively, and K is constant set to 0.6. This constant is chosen as twice the length of the last link of the robot. Using this definition, the neighborhood of an object is easy to compute and captures enough of its surroundings for accurate action/grasp feasibility prediction.

Model Architecture. The IK feasibility prediction and GO estimation modules are 4-Layer MLPs with a hidden size of 512. The AGF module, on the other hand, has a hidden size of 256, 4 attention heads and one messsage-passing step. Exceptionally, when training on the **PR2-3D-4** dataset, we use a hidden size of 256 for the GO module as it yields better results.

Training Details. The three modules are implemented in Pytorch Geometric (Fey & Lenssen, 2019) and trained using the Adam optimizer (Kingma, 2014). During the pre-training stage, each module is trained for 100 epochs. We use a batch size of 8192 and a learning rate of 0.001 for the IK feasibility classifier and the GO estimator. For AGF classifier, we set the batch size to 2048 and the learning rate to 0.0001. During the fine-tuning stage, the complete **GRN** model is trained for 100 epochs with a batch size of 2048 and a learning rate of 0.0001. The model is trained on an Intel(R) Xeon(R) W-2223 CPU @ 3.60GHz workstation, with an NVIDIA RTX A5000 GPU. The full training process takes approximately 15 hours.

Inference Time Decomposition. Table A shows the inference time decomposition of our proposed model. The total inference time of **GRN** is 5.5 ms in average, with the most significant portion spent on scene graph construction with an average time cost of 3 ms. During this step, time is mostly spent on evaluating the distance between pairs of objects to add proximity edges between their corresponding nodes. The IK feasibility and GO modules take each a computation time of 0.5 ms. Finally, the AGF module takes 1.5 ms to output the action and grasp feasibility prediction modules.

Table 6: Decomposition of the inference time of **GRN** on a 3D environment with 4 movable objects and 15 fixed objects.

Step	Inference Time
Scene Graph Construction	3 ms
IK Feasibility Module	0.5 ms
GO Module	0.5 ms
AGF Module	1.5 ms
Total	5.5 ms

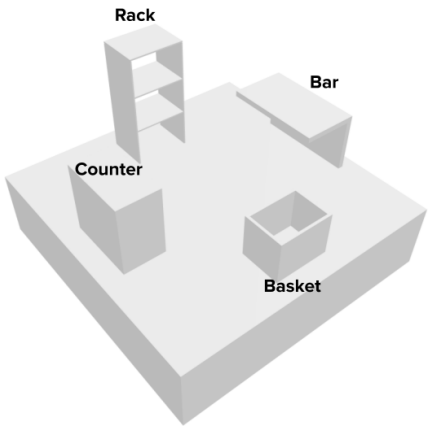


Figure 3: Visualization of the different types of structures used during data generation.

B DATA GENERATION AND ANNOTATION METHOD

Data Generation. Our data generation process consists of generating a number of 3D environments containing a random number of support surfaces, movable objects and obstacles within a specified range. We define 4 types of structures as shown in Figure 3: (1) a rack which is a structure containing a varying number of shelves separated by a gap, and a holder on each side or each corner, (2) a bar which is a L-shaped structure with one support surface and one holder, (3) a basket which consists of a support surface and 4 sides surrounding it, and (4) a counter which is a large block. When generating an environment, we first randomly sample a number of structures within the specified range. We then choose one of the predefined structures, before randomly sampling its dimensions and pose. After collision checking using the FCL libarary (Pan et al., 2012), all objects composing the structure are added to the environment as fixed objects. These can be either support surfaces or obstacles (e.g holders). This process is repeated until the number of structures sampled is reached.

Given the fact that our approach represents objects using their bounding boxes, it is sufficient to generate a dataset consisting of box-shaped objects only. After sampling a number of movable objects to add, we randomly sample the dimensions of the object with a specified range. In order to incorporate difficulties encountered in real-world scenarios, we define three methods for object placement sampling: (1) the first is random placement in which a support surface and a pose within its bounds are randomly sampled, (2) proximity placement in which the object is placed in the neighborhood of a randomly chosen object, (3) the third is the underneath placement in which the object is placed underneath a randomly chosen support surface. We check the sampled object placement for collisions before adding it to the environment. We then repeat the process until the sampled number of movable objects is reached. Finally, a number of fixed box-shaped obstacles is randomly added to the scene, before storing it.

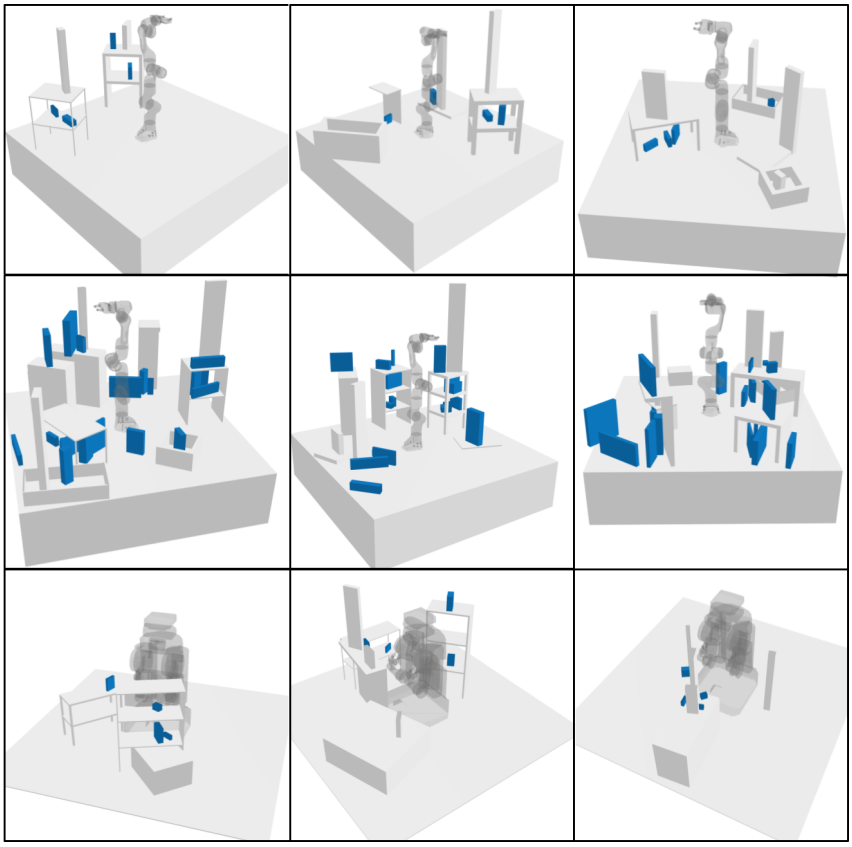


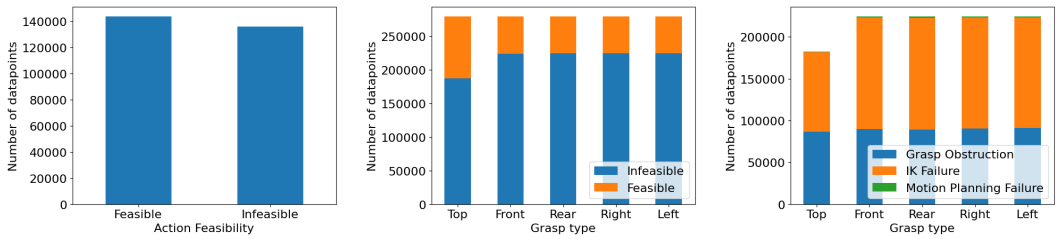
Figure 4: Visualization of generated environments from the (top) **Panda-3D-4**, (middle) **Panda-3D-20**, and (bottom) **PR2-3D-4** test sets

Using this method, we generate the datasets presented in Section 5.1. The **Panda-3D-4** dataset consists of a training set containing 70'000 scenes, a validation set of 10'000 scenes, and a test set of 20'000 scenes, each one generated using a different random seed to ensure that the environments are different across all three sets. The **Panda-Tabletop-4** and **PR2-3D-4** are each composed of 25'000 training scene, 5'000 validation scenes and 10'000 test scenes. Figure 4 shows environments from different datasets.

Data Annotation. Data annotation can be done using any off-the-shelf geometric planner. In this work, we use an adapted version of Moveit Task Constructor (Görner et al., 2019), with a Bidirectional Transition-Based Rapid Random Tree motion planner Devaurs et al. (2013). Since pick and place actions are symmetrical, they can be considered as equivalent. Thus, we only annotate pick actions. Also, we do not focus on trivial infeasibility cases due to the object being larger than the gripper. Hence, during annotation, we allow collisions between the object to pick and the gripper fingers. This also allows the handling of mesh objects such as a mug or a wine glass, for which feasible grasps exist even if the bounding box is larger than the gripper's maximum width.

For each movable object in a generated environment, we query the geometric planner to plan a pick action from its placement in the environment. If a solution is found, the action is annotated as feasible. In parallel, we set the feasibility label of each grasp type $g \in \mathcal{G}$ to 1 if at least one grasp in g is feasible. Similarly, we set the IK feasibility label of a grasp type to 1 if at least one grasp belonging to it has a valid inverse kinematics solution. Finally, we record all grasp-obstructing objects as well as the ratio of grasps obstructed per grasp type. Data annotation takes approximately 68 hours to complete on the **Panda-3D-4** dataset.

Figures 5 and 6 show the distributions of obtained labels for the **Panda-3D-4** and **PR2-3D-4** training sets. For the former, the action feasibility annotations are balanced, while grasp types feasibility is imbalanced with more infeasible cases. Regarding reasons of infeasibility, Figure 5c shows a balanced distribution between failures due to IK feasibility and grasp obstructions. Cases where infeasibility is due to motion planning failure, on the other hand, appear rarely in the dataset. The **PR2-3D-4** dataset has a more pronounced imbalance towards infeasibility, the latter being caused more often by the absence of inverse kinematics solutions than grasp obstructions, showcasing the more complex kinematics of the PR2 robot.

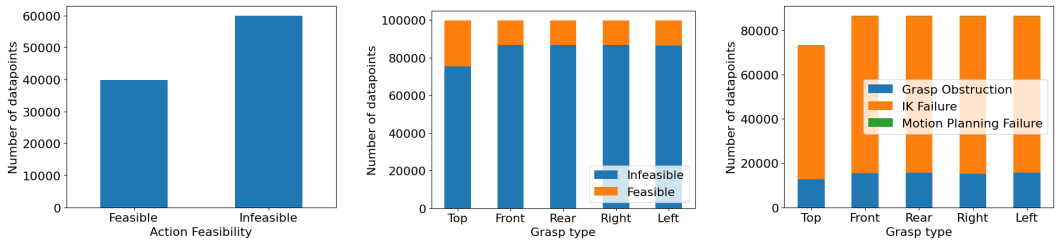


(a) Number of feasible and infeasible actions

(b) Number of feasible and infeasible cases per grasp type

(c) Distribution of failure causes per grasp type

Figure 5: Annotations statistics for the **Panda-3D-4** training set.



(a) Number of feasible and infeasible actions

(b) Number of feasible and infeasible cases per grasp type

(c) Distribution of failure causes per grasp type

Figure 6: Annotations statistics for the **PR2-3D-4** training set.

C APPLICATION TO TAMP

C.1 GRN PLANNING ALGORITHM

Algorithm 1 shows the pseudo-code for the planner we develop to test our approach on offline manipulation problems. Given the **GRN** predictions on the initial state of the environment, a feasibility threshold and a grasp obstructions threshold, it recursively builds a task plan by moving objects when its feasible, or trying to free access to grasp types based on grasp obstruction information otherwise. Once a task plan is found, we query the geometric planner used during data annotation to verify its geometric feasibility, and plan the corresponding robot motions. For simplicity, Algorithm 1 shows the planner used for the single robot problem only. For the multi-robot **Clutter** problem using both arms of the PR2 robot, we choose the robot with the higher feasibility prediction to execute feasible actions. If an action is infeasible for both arms, we consider both of them when trying to free grasp types.

Algorithm 1 GRNPlanner

Input: $F_a, \boldsymbol{F}_{\mathcal{G}}, \boldsymbol{\kappa}_{\mathcal{G}}, \boldsymbol{\rho}_{\mathcal{G}}, \mathcal{T}_F, \mathcal{T}_{GO}, \mathcal{O}$ ▷ **GRN** predictions, feasibility threshold, grasp obstruction threshold, desired object

$plan \leftarrow []$
if $F_a(\mathcal{O}) > \mathcal{T}_F$ **then**
  $feasibleGrasps = \emptyset$
  for each $g \in \mathcal{G}$ such that $F_g(\mathcal{O}) > \mathcal{T}_F$ **do**
    $feasibleGrasps \leftarrow feasibleGrasps \cup g$
  end for
  $plan.append(Move(\mathcal{O}, feasibleGrasps))$
  return $plan$
else
  $BlockersPerGraspType \leftarrow [\emptyset \quad \forall g \in \mathcal{G}]$
  $freeableGraspTypes \leftarrow \mathcal{G}$
  for each $g \in \mathcal{G}$ **do**
    if $\kappa_g(\mathcal{O}) < \mathcal{T}_F$ **then**
      $freeableGraspTypes \leftarrow freeableGraspTypes \setminus g$
    else
      for $\mathcal{O}' \in \mathcal{N}(\mathcal{O})$ such that $\rho_g(\mathcal{O}, \mathcal{O}') > \mathcal{T}_{GO}$ **do**
        if $\mathcal{O}'$ is fixed **then**
          $freeableGraspTypes \leftarrow freeableGraspTypes \setminus g$
          break
        else
          $BlockersPerGraspType[g] \leftarrow BlockersPerGraspType[g] \cup \mathcal{O}'$
        end if
      end for
    end if
  end for
  if $freeableGraspTypes$ is empty **then**
    return []
  else
    for each $g \in freeableGraspTypes$ **do**
      $subplan \leftarrow []$
      $graspTypeFreed \leftarrow True$
      for each $\mathcal{O}' \in BlockersPerGraspType[g]$ **do**
        $result \leftarrow GRNPlanner(F_a, \boldsymbol{F}_{\mathcal{G}}, \boldsymbol{\kappa}_{\mathcal{G}}, \boldsymbol{\rho}_{\mathcal{G}}, \mathcal{T}_F, \mathcal{T}_{GO}, \mathcal{O}')$
        if $result$ is empty **then**
          $graspTypeFreed \leftarrow False$
          break
        else
          $subplan.append(result)$
        end if
      end for
      if $graspTypeFreed$ is True **then**
        $plan.append(subplan)$
        return $plan$
      end if
    end for
    if $graspTypeFreed$ is False **then**
      return []
    end if
  end if
end if

Figure 7: Rollout of the solution found by the **GRN**-based planner on the **Access** problem.

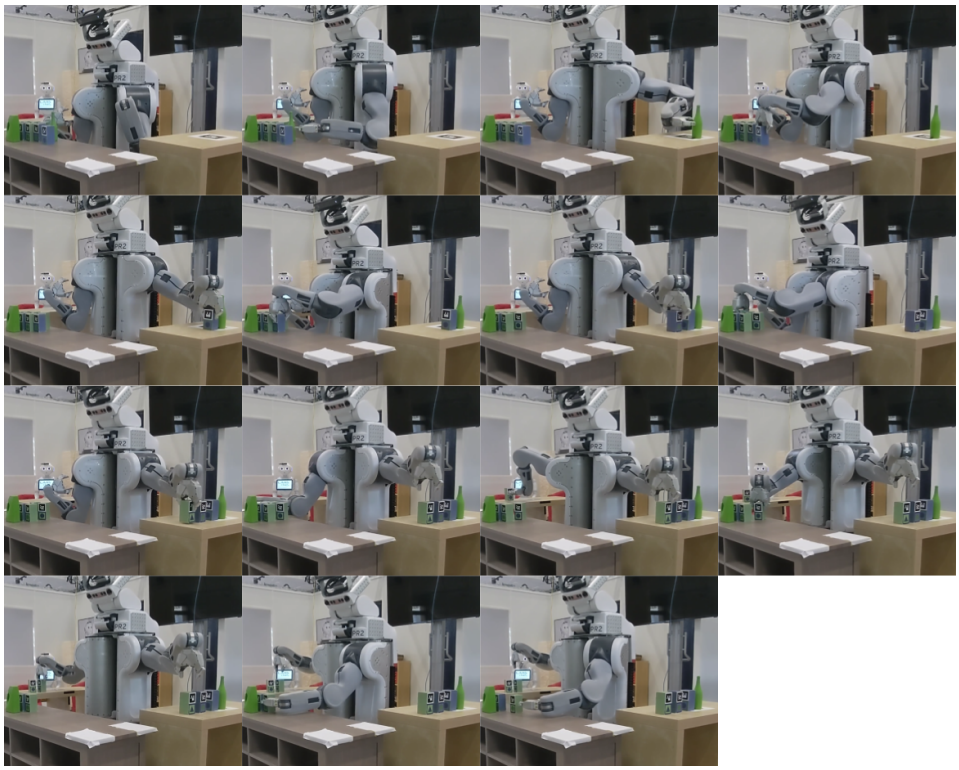


Figure 8: Rollout of the solution found by the **GRN**-based planner on the **Clutter** problem.

C.2 REAL-WORLD EXPERIMENTS

We test the GRN-based planner in two real-world setups, using a Panda arm and a PR2 robot. We use a perception module based on Overworld (Sarthou, 2023) to identify objects and estimate their poses. The shapes of the object are known and are not subject to detection. The GRN-based planner was able to generate geometrically feasible plans on both the **Access** and **Clutter** problems, which were successfully executed by the robots. Figures 7 and 8 show the rollouts of both executions. In the **Access** problem, the desired task is simply to pick and place the bottle object, the planner has to find the solution plan that needs to move all obstructing objects to get the bottle. In the **Clutter** problem, the goal is to simply pick the object in the middle of the clutter and place it at the center of the table. In the same way, the planner has to find a solution that frees one of the grasp types of the object first. Although these tasks might seem simple, they represent challenging TAMP problem with high combinatorial complexity. Indeed, in both problems, multiple objects not only block access to the goal object, they also block access to each other. The number of geometrically feasible solutions is very limited, requiring the planner to find the specific order in which to move the objects and which grasp to choose at each step. As shown in Figures 7 and 8, the **Access** problem requires 6 Pick-Place actions to be solved, while the **Clutter** problem requires 7 Pick-Place actions to be solved.

C.3 ADDITIONAL SIMULATION EXPERIMENTS

Access problem with a large number of objects. We conduct an additional experiment in simulation to test the scalability of our method to problems requiring long-horizon planning. As in the previous problems, the goal is to move a single object (shown in pink in Figure 9), to which a total of 27 objects block access. This task requires removing at least 18 objects to access the desired one, resulting in a total of 19 necessary Pick-Place actions. Figure 9 shows **GRN**'s predictions on the initial state of this problem. The model accurately predicts that, initially accessing the desired object is infeasible, and that only two blockers can be moved, using a single grasp type each. For simplicity, we show the obtained reasons of infeasibility of the pink object only. However, after the same initial query, our model also predicts the reasons of infeasibility for all the blockers as well. The non-informed TAMP algorithm from Bouhsain et al. (2024) completely fails to solve this problem, even after 2 hours of planning, while the **GRN**-based planner find a feasible solution in under 15s.

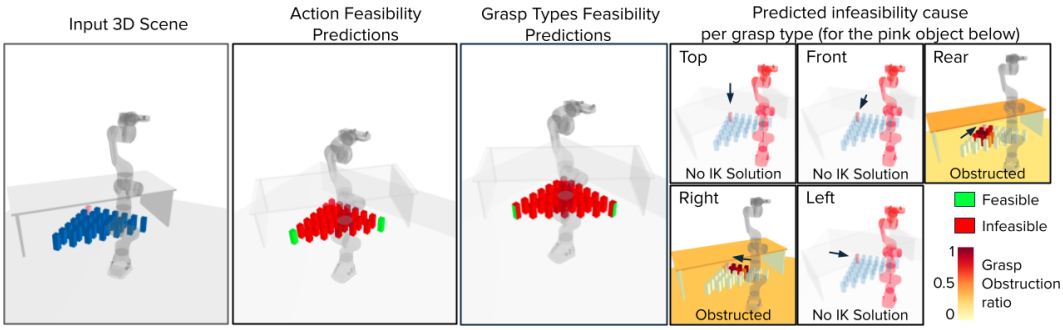


Figure 9: Visualization of **GRN** predictions on the initial state of a larger version of the **Access** problem.

Inter-robot Handover. Our proposed approach can also be used to predict the feasibility of handover actions between two robots. Indeed, given the robot-centric nature of our model, we can divide a handover action into two separate Pick-Place actions, and query **GRN** for each robot individually to predict the feasibility of each sub-action. One of the main challenges of handover tasks is to determine where to perform the object exchange. Our model can be used to efficiently find feasible handover positions by evaluating different sampled positions. In order to showcase this ability, we define the **Handover** problem, in which two Panda robots have to perform a handover task for a single object. A wall separates the two robots leaving only 3 "windows" through which the object can be exchanged. We sample a number of potential handover positions that lie in the intersection between the two robots workspaces, and query **GRN** to evaluate their feasibility. As shown in Figure

10, our model is able to accurately predict that the wall blocks most of the sampled positions, and that only those in front of the windows are feasible. One key advantage of our method is that all the sampled positions can be evaluated simultaneously by adding a node to the input scene graph for each one, while making sure that proximity edges are not created between these nodes. This enables an efficient search for feasible handover positions, requiring a single query to **GRN**. In contrast, traditional methods need to query the geometric planner for each sampled position independently, resulting in long planning times.

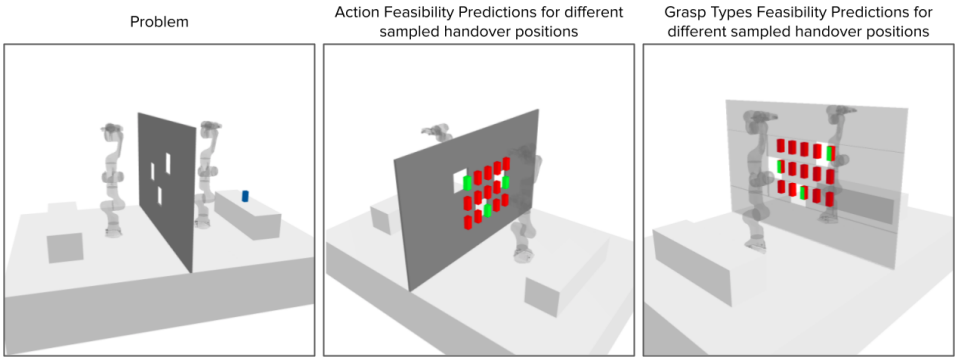


Figure 10: Visualization of **GRN** predictions on different sampled handover positions in the **Handover** problem.

D VISUALIZATIONS

In this section, we present a visual comparison between the predictions of our model, GRN, and the ground truth. Figure 11 illustrates GRN's predictions in a test environment from **Panda-3D-4**, while Figure 12 showcases predictions from **Panda-3D-10**. In these figures, action feasibility predictions and ground truth are represented through object colors in the top images: green indicates feasibility for pick or place actions, while red denotes infeasibility. The feasibility of different grasp types is visualized through the coloration of the corresponding object's faces. Finally, the reasons of infeasibility are shown for one of the objects in the environment (in blue), for each grasp type. There are two types of infeasibility causes, IK infeasibility (with the robot depicted in red) and grasp obstructions (where obstructing objects are displayed in full opacity, with their colors reflecting obstruction ratios). These comparisons highlight that GRN accurately predicts action and grasp feasibility for most objects. It also effectively identifies the reasons for infeasibility and estimates grasp obstruction ratios with minimal error. A notable exception occurs in the 10-object environment, where the *Rear* grasp type of one object is incorrectly classified as infeasible. Figure 12 details the predicted infeasibility reasons for this misclassification. A closer examination of the corresponding *Rear* grasp type plot reveals a slight overestimation of the obstruction ratio for a large block, which likely contributes to the error. However, GRN correctly predicts the feasibility of other grasp types for the same object, including the feasible *Top* grasp type, ensuring accurate action feasibility predictions overall.

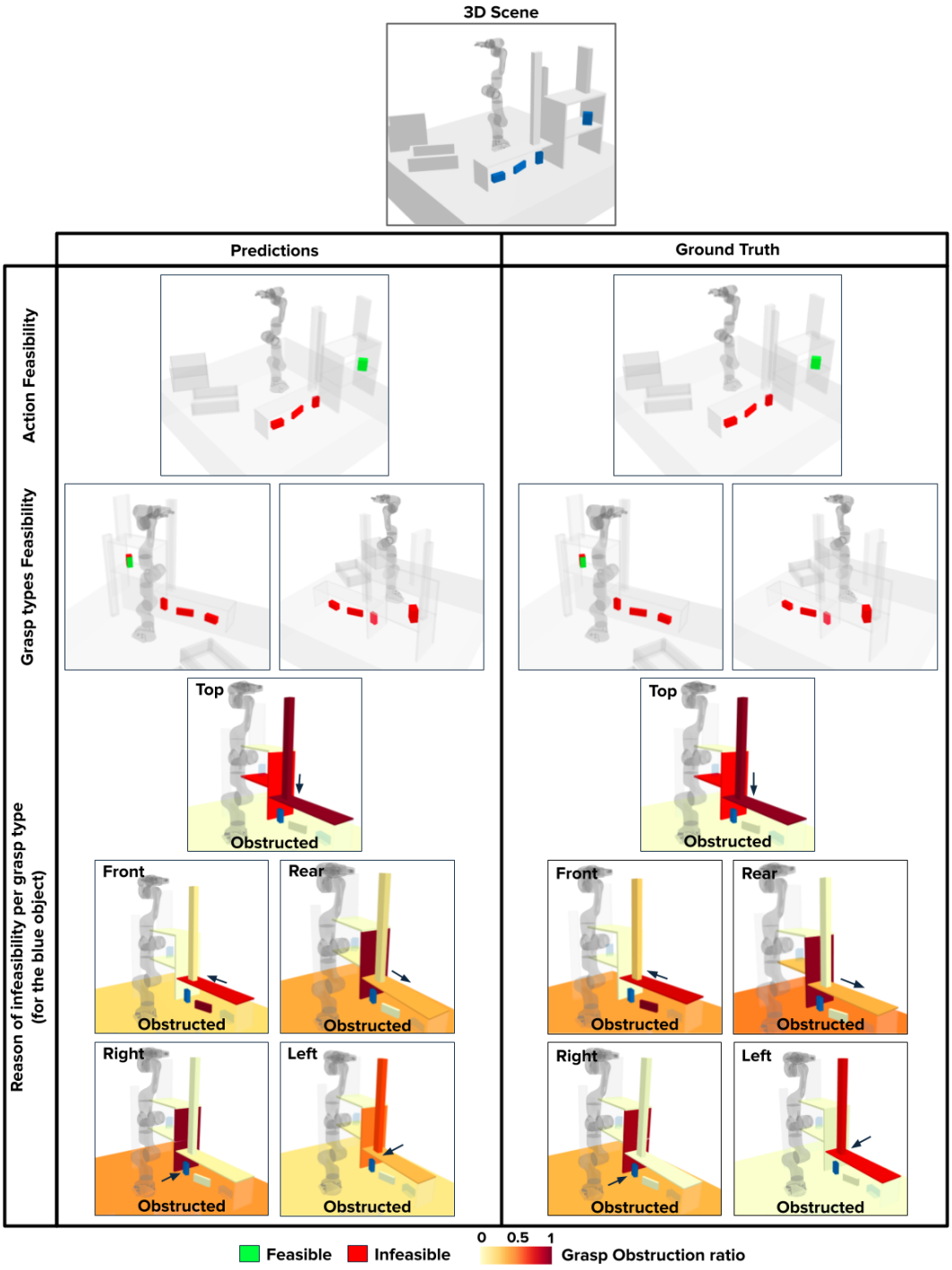


Figure 11: Visualization of **GRN** prediction compared to the ground truth on a test environment from **Panda-3D-4**.

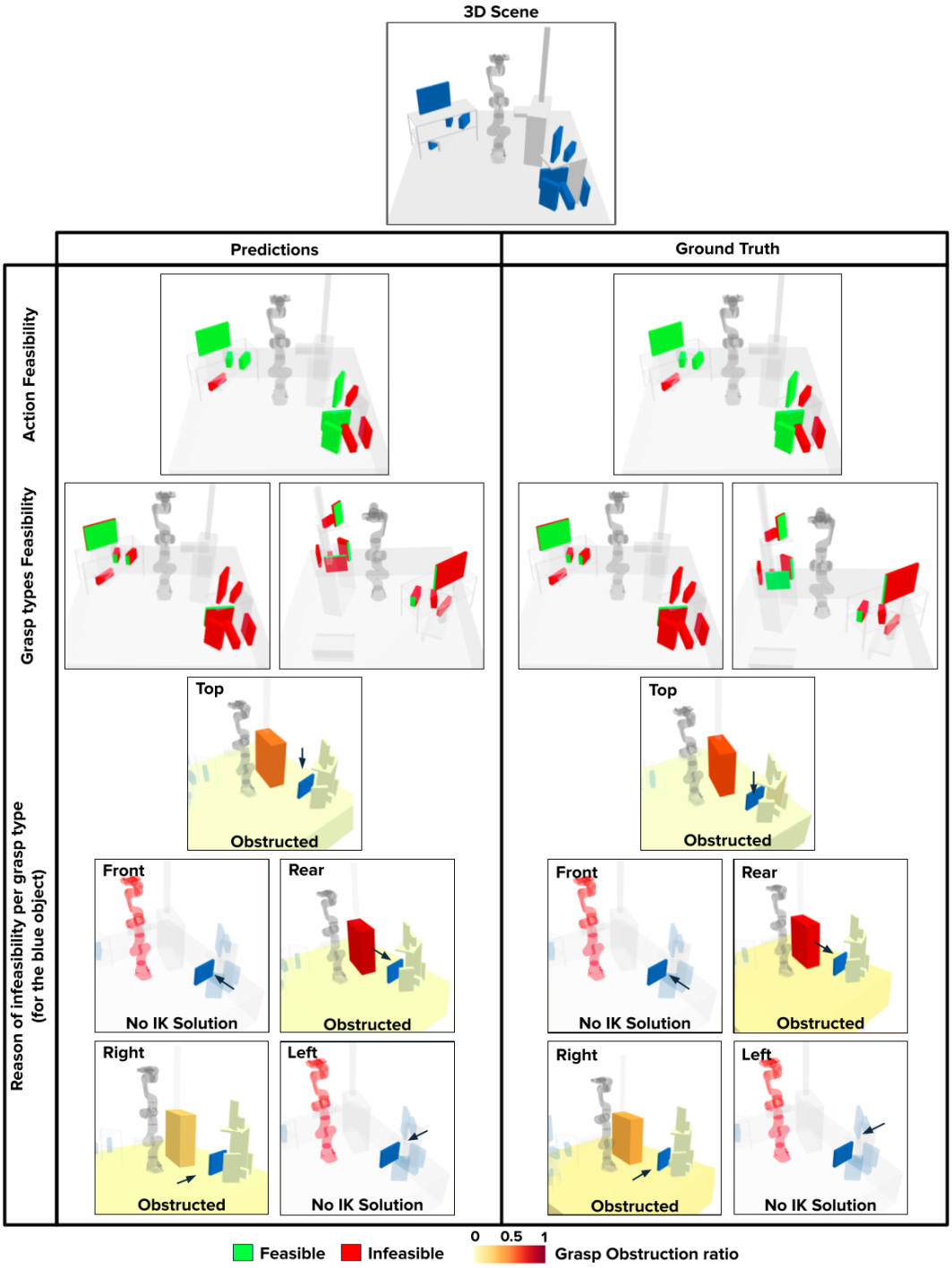


Figure 12: Visualization of **GRN** prediction compared to the ground truth on a test environment from **Panda-3D-10**.