Visually Grounded Task and Motion Planning for Mobile Manipulation

Xiaohan Zhang[1], Yifeng Zhu[2], Yan Ding[1], Yuke Zhu[2], Peter Stone[2,3], Shiqi Zhang[1]

Abstract— Task and motion planning (TAMP) algorithms aim to help robots achieve task-level goals, while maintaining motion-level feasibility. This paper focuses on TAMP domains that involve robot behaviors that take extended periods of time (e.g., long-distance navigation). In this paper, we develop a visual grounding approach to help robots probabilistically evaluate action feasibility, and introduce a TAMP algorithm, called GROP, that optimizes both feasibility and efficiency. We have collected a dataset that includes $96,000$ simulated trials of a robot conducting mobile manipulation tasks, and then used the dataset to learn to ground symbolic spatial relationships for action feasibility evaluation. Compared with competitive TAMP baselines, GROP exhibited a higher task-completion rate while maintaining lower or comparable action costs. In addition to these extensive experiments in simulation, GROP is fully implemented and tested on a real robot system.

I. Introduction

Task and motion planning (TAMP) algorithms and systems have been used for robot planning at both discrete and continuous levels [1], [2]. Task planners sequence symbolic actions for guiding the robot's high-level behaviors [3], and motion planners calculate low-level motion trajectories in continuous spaces [4]. TAMP algorithms aim to bridge the gap between task planning and motion planning towards enabling robots to fulfill task-level goals and maintain motion-level feasibility at the same time [5]–[13].

One way to categorize TAMP domains is based on if a problem domain requires robot actions that take relatively short time (e.g., seconds, such as picking up and putting down objects) or relatively long time (e.g., minutes or even hours as navigating from one location to another) [14]. In the former type of domains, action feasibility is much more important to consider than plan efficiency, since extra plan steps do not add much time to the expected execution time. On the other hand, this paper is motivated by the latter type of TAMP domains, wherein it is advantageous to incorporate both *efficiency* and *feasibility* into the evaluation of plan qualities. Some existing TAMP research incorporates both efficiency and feasibility into task-motion planning [9], [14]. However, those methods evaluate feasibility in a deterministic way, and rely on predefined "state mapping functions" for mapping each symbolic state into feasible poses in continuous space. For instance, to unload an object to a table, a robot needs to move to the table first, i.e., `beside(table)=true`, where previous TAMP research that we are aware of relies on predefined feasible poses that
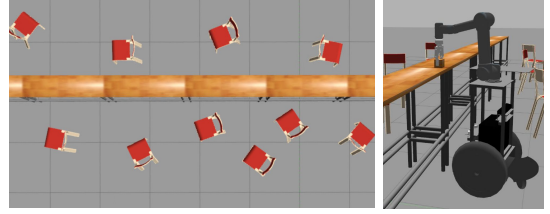

Fig. 1. Our mobile manipulation domain that includes a long banquet table surrounded by chairs. Given a target location (on the table) to place an object, the robot needs to navigate to a location from which it can successfully perform the manipulation action, ideally as quickly as possible (thus preferring the near side of the table when feasible).

are spatially close to the table to evaluate the truthfulness of the "beside table" statement.

Such predefined state mapping functions that assume deterministic action feasibility have at least two deficiencies. First, a predefined state mapping function is not robust to dynamic obstacles (e.g., people seated around the table). Second, not all "feasible" behaviors are equally preferred, e.g., standing far and stretching out to place an object may be less preferred than standing close to do so. Those observations motivate this work that learns to evaluate action feasibility for robot task-motion planning.

The main contribution of this work is a visually grounded TAMP algorithm, called **G**rounded **RO**bot Task and Motion **P**lanning (GROP), that probabilistically evaluates action feasibility, and incorporates both feasibility and efficiency towards maximizing long-term utility. Inspired by the concept of "symbol grounding" [15], we use "visual grounding" to refer to methods that use computer vision techniques to help an agent interpret abstract symbol tokens and connect them to the real world.

We have applied GROP to a domain of a mobile manipulator setting "dinner tables," as illustrated in Fig. 1. The robot needs to decide how to approach a table at the task level (e.g., from which side of the table), compute 2D navigation goals (connecting task and motion levels), and plan motion trajectories for navigation and manipulation behaviors. We have collected a dataset that includes $96,000$ instances of a robot conducting mobile manipulation tasks where in each instance, a robot unloads an object with dynamic obstacles surrounding a table. An instance is labeled "successful" if the robot is able to compute and execute a task-motion plan that includes both navigation and manipulation actions. We use fully convolutional networks (FCNs) [16] to learn to visually ground spatial relationships and evaluate action feasibility. GROP is summarized in Fig. 2.

Compared with baselines from the literature [14], [17], GROP performed better in success rate while maintaining

[1] Department of Computer Science, The State University of New York at Binghamton {xzhan244; yding25; zhangs}@binghamton.edu
[2] Department of Computer Science, The University of Texas at Austin {yifeng.zhu@; yukez@cs.; pstone@cs.}utexas.edu
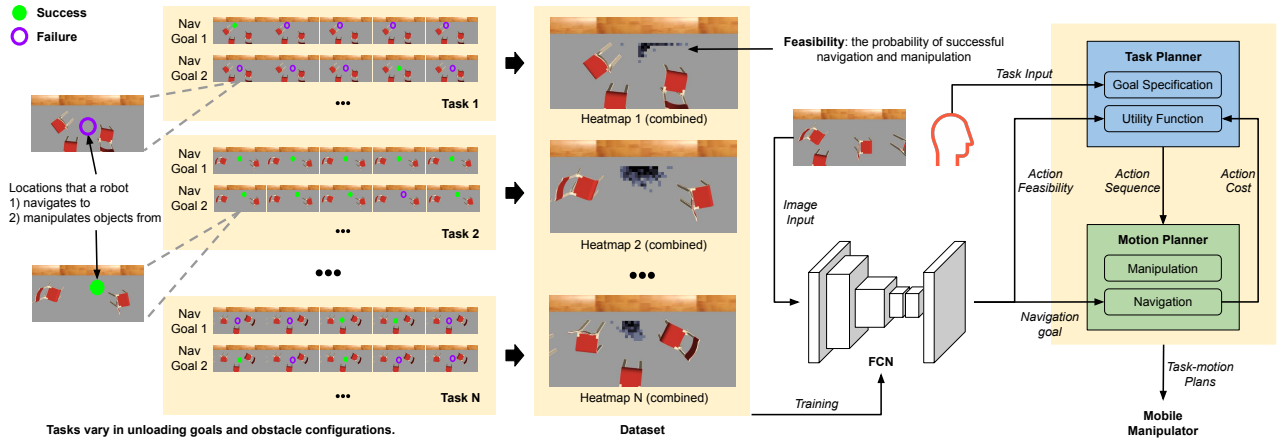[3] Sony AI

Fig. 2. An overview of this work, including an FCN-based feasibility evaluation approach, and GROP, our grounded TAMP algorithm. A *task* corresponds to one "unloading goal" on the table, as well as a configuration of obstacles (chairs in our case). Given a task, every pixel is considered a navigation goal – the robot attempts to navigate there, and unload an object from there. This navigation-manipulation process is referred to as a *trial*. The robot performs multiple trials for each navigation goal, which yields a *feasibility* value for that particular location. The feasibility values together form one *heatmap* for each task. In our *dataset*, each instance is a top-down view image, whose label is the corresponding heatmap. The "Dataset" box shows a few "combined heatmaps" where heatmaps are overlaid onto the corresponding images. Training with the dataset generates an FCN that is used for two purposes: 1) evaluating the feasibility of task-level actions, and 2) selecting motion-level navigation goals. Finally, GROP incorporates both efficiency (measured by action costs) and feasibility to compute task-motion plans for a mobile manipulator.

lower (or comparable) cumulative action costs. Finally, we demonstrate GROP with real-world robot hardware.

II. RELATED WORK

TAMP methods aim to compute plans that fulfill task-level goals while maintaining motion-level feasibility, as reviewed in recent articles [1], [2]. Several TAMP algorithms have been introduced in recent years (e.g., [6], [7], [18]–[29]. Within the TAMP context, we distinguish a few subareas of TAMP that are closest to this research on learning to visually ground symbolic spatial relationships towards planning efficient and feasible task-motion behaviors under uncertainty.

A. TAMP for Efficient and Feasible Behaviors

When high-level actions only take a few seconds, TAMP algorithms can focus mostly on action feasibility constraints without fully optimizing high-level plan efficiency. However, when there are actions that take significant time to execute (e.g., long-distance navigation), task-completion efficiency cannot be overlooked. Some recent methods have considered efficiency in different aspects of TAMP, such as planning task-level optimal behaviors in navigation domains [14], integrating reinforcement learning with symbolic planning in dynamic environments [8], computing safe and efficient plans for urban driving [30], and optimizing robot navigation actions under the uncertainty from motion and sensing [9]. In contrast to those methods that do not have a perception component, GROP visually grounds symbols (about spatial relationships) to probabilistically evaluate action feasibility for task-motion planning.

B. TAMP under Uncertainty

While most TAMP methods assume a fully observable and deterministic world [2], some have been developed to account for the uncertainty from perception and action outcomes [31]–[36]. For instance, the work of Kaelbling

and Lozano-Pérez extended the "hierarchical planning in the now" approach to address both current-state uncertainty and future-state uncertainty [31]. Going beyond those methods that aim to maintain plan feasibility to complete tasks under high-level uncertainty, we consider uncertainty in the robot motion and also incorporate task-completion efficiency into the optimization of robot behaviors. As a result, our GROP algorithm is particularly suitable for TAMP domains that require robot operations over extended periods of time, such as long-distance navigation.

C. TAMP with Visual Perception

Recently developed methods have shown that visual information can be used to help robots predict plan feasibility, including task-level feasibility [6], [37], and motion-level feasibility [10], [17]. Those methods were developed to maximize task completion rate in manipulation domains, and actions that take relatively long time (such as long-distance navigation) were not included in their evaluations. Focusing on robots that operate over extended periods of time, GROP (ours) incorporates efficiency into plan optimization. For instance, when highly feasible plans have very high costs, GROP supports the flexibility of executing slightly less feasible plans with much lower costs. GROP achieves this desirable trade-off between feasibility and efficiency by probabilistically evaluating plan feasibility, which is not supported by the above-mentioned methods.

III. PROBLEM STATEMENT

We consider a mobile manipulation domain that includes N objects *Obj*. There are obstacles (tables and chairs in our case) that prevent the robot from navigating to some positions in the domain. Location l is a symbolic concept that corresponds to a set of obstacle-free 2D poses (X), where each pose $(x \in X)$ specifies a 2D position and an orientation.

The robot needs to move each object $o \in Obj$ from its initial location to a goal position.

Actions: The robot is equipped with skills of performing a set of symbolic (task-level) actions denoted as $A : A^n \cup A^m$, where A^n and A^m are *navigation* actions and *manipulation* actions respectively. A navigation action $a^n_{l,l'} \in A^n$ is specified by its initial and goal locations, $l, l' \in L$, where L includes a set of symbolic locations. A manipulation action, $a^m_{o,l} \in A^m$, is specified by an object to be manipulated, $o \in Obj$, and a symbolic location, $l \in L$, to which the robot navigates and performs the manipulation action. We consider two types of manipulation actions of loading and unloading, represented by a^{m+} and a^{m-} respectively. Actions are defined via preconditions and effects. For instance, the action load(o_1) has preconditions of at(robot,l_1) and at(o_1,l_1), meaning that to load the object o_1, the object must be co-located with the robot at the location l_1. The effects of load(o_1) include o_1 being moved into the robot's hand, i.e., inhand(o_1).

Perception: The robot visually perceives the environment through top-down views over the areas where manipulation and navigation actions are performed. We use *IM* to represent a 2D image that captures the current obstacle configuration, as shown in the "Image Input" of Fig. 2 (bottom right). To facilitate robot learning, we provide a dataset (as illustrated in the "Dataset" box of Fig. 2). Each instance includes a top-down view image, and a target object with a predefined position, while each label is in the form of a heatmap. Each pixel of a heatmap is associated with a 2D position, and has a "feasibility" value that represents the success rate of the robot navigating to the 2D position, and manipulating the target object from there.

A map is generated in a pre-processing step, and provided to the robot as prior information for navigation purposes using rangefinder sensors.

Uncertainty: The outcome of performing navigation action $a^n_{l,l'}$ to goal pose x is deterministic at the task level, but is non-deterministic at the motion level. In other words, the robot will end up in position x', which is not necessarily the same as x. This setting captures the fact that a mobile robot never achieves its exact 2D navigation goal (due to its imperfect localization and actuation capabilities), though successfully navigating to an area (l) is generally possible.

We focus on the interdependency between navigation and manipulation actions. For instance, the execution-time uncertainty from navigation actions results in different standing positions of the robot, which makes the outcomes of manipulation actions non-deterministic. This challenge generally exists in mobile manipulators. We assume no noise in the execution of manipulation actions (loading and unloading) to objects within a reachable area.

Format of Solution: A solution is in the form of a task-motion plan $p = \langle p^t, p^m \rangle$, where task plan p^t is of the form $\langle a^n_0, a^m_0, a^n_1, a^m_1, ... \rangle$, indicating that navigation and manipulation actions are interleaved. Motion plan p^m is of

the form $\langle \xi^n_0, \xi^m_0, \xi^n_1, \xi^m_1, ... \rangle$, and ξ^n_i (or ξ^m_i) is a trajectory in continuous space for implementing symbolic action a^n_i (or a^m_i). The quality of task-motion plan p is evaluated using a utility function $\mathcal{U}(p)$, which considers both feasibility and efficiency of plan p:

$$\mathcal{U}(p) = \mathcal{R} \cdot \mathcal{F}(p) - \mathcal{C}(p), \tag{1}$$

where $\mathcal{F}(p) \in [0, 1]$ is the plan feasibility (i.e., the probability that p can be successfully executed), $\mathcal{C}(p)$ is the overall plan cost of executing p, and $\mathcal{R} \to \mathbb{R}$ is a success bonus reflecting the reward from a successful execution. An optimal algorithm reports a task-motion plan of the highest utility:

$$p^* = \arg\max_p \ \mathcal{U}(p)$$

Next, we present an algorithm that computes such task-motion plans through visually grounding spatial relationships while considering both efficiency and feasibility.

IV. THE GROP ALGORITHM

In this section, we introduce the paper's main contribution, an algorithm called **G**rounded **RO**bot Task and Motion **P**lanning, or GROP for short.

A. Algorithm Description

Algorithm 1 presents the GROP algorithm. Implementing GROP requires a task planner *Plnr*^{t}, a motion planner *Plnr*^{m}, a success bonus $\mathcal{R} \to \mathbb{R}$, and a cost function *Cst* that evaluates the cost of any motion trajectory generated by *Plnr*^{m}. Inputs of GROP include a rule-based task description T, a robot initial 2D position x^{init}, and a provided dataset D. GROP outputs a task-motion plan p in the form of $\langle p^t, p^m \rangle$.

GROP starts with training an FCN-based feasibility evaluator Ψ using provided dataset D in Line 1. Then it initializes an empty set of task-motion plans **P** in Line 2. *Plnr*^{t} takes T as input and outputs a set of task-level satisficing plans, denoted as **P**^{t} in Line 3. The outer for-loop (Lines 4-21) iterates over each task-level satisficing plan. In each iteration, GROP evaluates the utility value of one task plan $\mathcal{U}(p)$, which incorporates both plan feasibility $\mathcal{F}(p)$ and plan efficiency $\mathcal{C}(p)$. Aiming to evaluate $\mathcal{F}(p)$ and $\mathcal{C}(p)$, each iteration in the first inner for-loop (Lines 7-13) considers a pair of navigation and manipulation actions in the task plan, and evaluates its feasibility and cost. In the second inner for-loop of Lines 14-17, GROP calls *Plnr*^{m} to compute one motion plan for each task-level action. Line 18 puts together task plan p^t and motion plan p^m to form a task-motion plan p. In the same line, p is added to task-motion plan set **P**. Lines 22-23 are the final steps to select and return the optimal task-motion plan from **P** given utility function $\mathcal{U}(p)$.

B. Feasibility Evaluation

In this subsection, we discuss how to evaluate action feasibility at task and motion levels (Line 10 in Algorithm 1), where the feasibility evaluation at the task level relies on the feasibility evaluation at the motion level.

Motion-Level Feasibility: In our mobile manipulation domain, motion-level feasibility $Fea^m(x, y)$ is a function

---

Algorithm 1 GROP

Require: Task planner $Plnr^t$, motion planner $Plnr^m$, success bonus $\mathcal{R}$, and cost function Cst
Input: Task description T, robot initial position x^{init}, dataset D
 1: Train a motion-level feasibility evaluator Ψ using dataset D (detailed in Section IV-B)
 2: Initialize a set of task-motion plans $\mathbf{P} \leftarrow \emptyset$
 3: Compute a set of task-level satisficing plans: $\mathbf{P}^t \leftarrow Plnr^t(T)$
 4: **for** each plan $p^t \in \mathbf{P}^t$ **do**
 5:   Initialize a motion-level position sequence: $X^{seq} \leftarrow [x^{init}]$
 6:   Initialize $tmp^f \leftarrow 0$ and $tmp^c \leftarrow 0$
 7:   **for** each action pair $\langle a_{l,l'}^n, a_{o,l'}^m \rangle$ in p^t **do**
 8:     Capture IM of location l'
 9:     Predict heatmap $h = \Psi(IM)$, using Eqn. 3
10:     $tmp^f \leftarrow tmp^f + Fea^t(a_{l,l'}^n, a_{o,l'}^m)$, using Eqn. 4
11:     $x' \leftarrow Smp(l', h)$, and append x' to X^{seq}
12:     $tmp^c \leftarrow tmp^c + Cst\big(Plnr^m(a_{l,l'}^n)\big) + Cst\big(Plnr^m(a_{o,l'}^m)\big)$
13:   **end for**
14:   **for** each $(x_i, x_{i+1}) \in X^{seq}$ **do**
15:     Compute motion-level trajectory $\xi \leftarrow P^m(x_i, x_{i+1})$
16:     Append ξ to motion plan p^m
17:   **end for**
18:   Generate task-motion plan $p \leftarrow \langle p^t, p^m \rangle$, and append p to the task-motion plan set $\mathbf{P}$
19:   Update $\mathcal{F}(p) \leftarrow tmp^f$ and $\mathcal{C}(p) \leftarrow tmp^c$
20:   $\mathcal{U}(p) \leftarrow \mathcal{R} \cdot \mathcal{F}(p) - \mathcal{C}(p)$ (Eqn. 1)
21: **end for**
22: Compute optimal task-motion plan: $p^* = \arg\max_{p \in \mathbf{P}} \mathcal{U}(p)$
23: **return** p^*

---

of 2D positions x and y, and is the probability of a robot successfully navigating to x and manipulating an object that is in position y. $Fea^m(x, y)$ can be extracted from gray-scale heatmap image h^y that is centered around y:

$$Fea^m(x, y) = h^y[x] \tag{2}$$

We use a FCN-based feasibility evaluator Ψ to generate heatmap h^y, given a top-down view image IM^y captured right above unloading position y ("Image Input" in Fig. 2):

$$h^y = \Psi(IM^y) \tag{3}$$

Data Collection and Learning Ψ with FCN: Here we discuss how to learn Ψ in Equation 3. A *task* specifies an obstacle configuration and a position y that a robot wants to unload objects to. In each *trial* of our data collection process, a robot attempts to navigate to position x, and then unload an object to position y. Such a trial produces a data point in the following format:

$$(IM^y, x) : r$$

where IM^y is a top-down view image captured right above y, and r is either *true* or *false* depending on if the robot succeeds in both navigation and manipulation actions. The robot repeated the same process for N times ($N = 5$ in our case), and we used the results $(r_0, r_1, \cdots, r_{N-1})$ to compute a success rate for positions x and y, which determines a gray-scale color for one pixel of a heatmap: $h[x]$.

Iterating over all possible positions of x in an area of $Width \times Height$ (24 pixels by 8 pixels in our case) in image IM, we were able to generate one full heatmap h for the current task. Here we assume this area is large enough to cover all positions, from which the robot can unload objects to y. To diversify the instances, we randomly placed obstacles (chairs in our case) to generate ten different "environments," and then randomly sampled unloading positions to generate a total of 100 tasks. As a result, our dataset contains 100 instances, each in the form of a top-down view image (64×32). Each instance has a label that is in the form of a heatmap. The size of our dataset is $96,000$, i.e., $100 \times N \times Width \times Height$.

Task-Level Feasibility: $Fea^t(a_{l,l'}^n, a_{o,l'}^m)$ evaluates the feasibility (in the form of a probability) of a robot successfully performing both task-level navigation action $a_{l,l'}^n$ and task-level manipulation action $a_{o,l'}^m$.

$$Fea^t(a_{l,l'}^n, a_{o,l'}^m) = \frac{\sum\limits_{i=0 \cdots N-1} Fea^m\big(Smp_i(l', h), y\big)}{N} \tag{4}$$

where function $Smp_i(l', h)$ samples the ith 2D position from location l'. The positions are weighted by heatmap h that is centered around object o. Intuitively, positions of higher motion-level feasibility are more likely to be sampled.

V. EXPERIMENTS

We conducted extensive experiments in simulation, where a mobile manipulator performs navigation and manipulation actions to set "dinner table." We also demonstrate GROP using a real robot system that includes a mobile platform and a robot arm. Our main hypothesis is that GROP outperforms existing TAMP algorithms in task completion rate without introducing additional action costs.

Baselines: GROP is evaluated through comparisons with the following baselines. All baselines are TAMP algorithms, and they vary in whether efficiency is considered in plan optimization, and whether feasibility is considered. All baselines select navigation goals by randomly sampling an obstacle-free position that is close to the unloading position.

- **Satisficing** (weakest baseline): Action costs are not considered, so it does not avoid long-distance navigation. All actions share the same feasibility (FCN not used).
- **PETLON** [14]: It considers plan efficiency, but does not quantitatively evaluate action feasibility.
- **DVH** [17]: It does not consider plan efficiency, but quantitatively evaluates action feasibility.
- **FCN-Planning** (most competitive): The same as GROP except that the heatmap (Line 11 in Algorithm 1) is not used for selecting 2D navigation goals.

It should be noted that, we cannot authentically implement DVH [17] for evaluation, because they used convolutional neural networks (CNNs) for task-level action feasibility evaluation, and we do not have a dataset from our domain for training the CNNs. We did the best we could by replacing their CNN-based visual component with our FCN-based feasibility evaluator.

Experiment Setup: The mobile manipulator includes a UR5e robot arm, a Robotiq 2F-140 gripper, an RMP 110
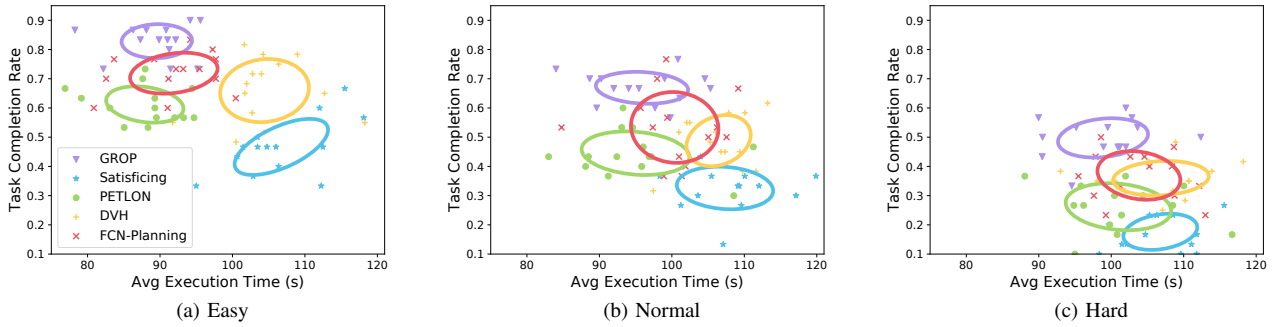
Fig. 3. Overall performances of GROP and four baseline methods in efficiency (x-axis) and task completion rate (y-axis). Tasks are grouped based on their difficulties. The ellipses represent the means and 2D standard variances of each approach. GROP produced the highest task completion rate, while maintaining smaller or comparable execution time. This observation is consistent over tasks of different difficulties.

mobile base, and a Velodyne VLP-16 lidar sensor. We used the Building-Wide Intelligence (BWI) codebase [38] to construct our simulation platform, which relies on the Gazebo physics engine [39]. We use a Rapidly exploring Random Tree (RRT) approach [40] to compute motion-level manipulation plans. The navigation stack was built using the `move_base` package of Robot Operating System (ROS) [41]. The robot's task planner is ASP-based [42], [43] and we used the Clingo solver for computing task plans [44].

The dataset described in Section IV-B was fed into an FCN for training Ψ. We adapted the FCN-VGG16 model [16] and trained it with batch size 4 and learning rate e^{-3}. We used a machine equipped with an Intel 3.80GHz i7-10700k CPU and a GeForce RTX 3070 GPU on a Ubuntu system.

The test environment contains two tables, one for loading and the other (a long banquet table) for unloading. Obstacles (chairs) are randomly placed near the unloading table. Positions and the number of chairs are dynamically changed for different environments. An RGB camera is attached to the ceiling to capture overhead images of environments. A mobile manipulator is tasked with moving three objects from the loading table to three different positions on the unloading table, where the robot can hold multiple objects at the same time. There is a tolerance of $0.1m$ for unloading actions, and an unloading action is considered unsuccessful if the object is more than this distance away from the specified unloading position. Task completion is evaluated based on whether each "seat" of the table is set up. Reward $\mathcal{R}$ has a value of 40 in our utility function defined in Equation 1.

GROP vs. Baselines: Fig. 3 shows the main results from experiments of comparing GROP to the four baselines. There were a total of 420 different tasks in 30 different environments. Each data point in the figure represents an average of 10 tasks. We grouped the tasks based on their difficulties: Easy, Normal, and Hard. A task's *difficulty* is measured by the total area that a robot can navigate to and unload an object from. For instance, a task with all unloading positions being surrounded by obstacles has a high difficulty. After sorting the tasks based on their difficulties, we evenly placed them into the three groups.

GROP consistently performed better in task completion rate (y-axis) in all three settings, while maintaining high plan efficiency (x-axis). We also see that GROP performed

TABLE I

TASK COMPLETION RATE / AVERAGE EXECUTION TIME IN ONE OF THE ENVIRONMENTS WITH DIFFERENT ROBOT'S NAVIGATION VELOCITIES.

	GROP	PETLON	DVH
Slow	0.80 / 166.16	0.63 / 166.46	0.73 / 204.04
Medium	0.82 / 95.42	0.63 / 93.23	0.73 / 112.02
Fast	0.88 / 59.56	0.63 / 56.62	0.73 / 66.01

particularly well in hard tasks where it produced the highest completion rate and the lowest action costs. While PETLON generated efficient plans (comparable to GROP), it does not reason about feasibility, resulting in low completion rate. DVH generates feasible plans (like FCN-Planning), but it does not consider action costs, resulting in long execution time in task completions. Results support our hypothesis that GROP improves plan efficiency without introducing additional action costs.

Robot Velocities: In this experiment, we used three robots that move at different velocities: 0.2 m/s (*Slow*), 0.4 m/s (*Normal*), and 0.8 m/s (*Fast*). Results are shown in Table I. Here we compare GROP to only the two baselines that are available from the literature (PETLON and DVH). We see that GROP outperforms the two baselines in task completion rate. What is interesting is that when the robot moves fast, GROP automatically weighted feasibility more, because the robot will not take too long to complete a navigation task anyway. As a result, GROP produced the highest task completion rate of 0.88 on a fast robot, while the baselines are not adaptive to the robot's velocity.

Illustrative Trials in Simulation: Fig. 4 shows three illustrative trials using GROP (ours) and two baselines (PETLON and DVH), where GROP produced the highest completion rate (3/3), while the baselines succeeded in at most two tasks. PETLON does not evaluate plan feasibility, and planned to unload objects to the middle and right positions from the south. In particular, unloading to the middle unloading position from the south is very difficult (with a feasibility value of 0.223). PETLON does not take such factors into consideration, which produced failures in unloading to the middle position. DVH does not consider efficiency in plan optimization, and generated a plan with long-distance
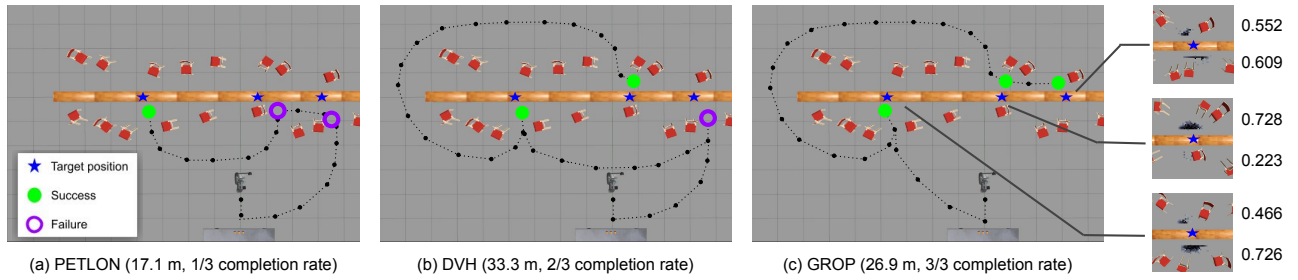
Fig. 4. Three illustrative trials using GROP and two baselines (PETLON and DVH). The robot needs to move three objects from the loading table (bottom) to three unloading target positions marked by blue stars, where the robot can hold multiple objects. Green dots (or purple circles) represent a robot successfully (or unsuccessfully) navigating to the position and unloading an object to the corresponding target position. Three heatmaps are overlaid onto overhead images, as shown on the right, indicating the feasibility values of navigating to and unloading from different positions. The numbers on the very right represent task-level action feasibility values of unloading from one side of the table. Under each subfigure, we present the total navigation distance and task completion rate, where we see GROP produced the highest completion rate, and performed better than DVH in efficiency.
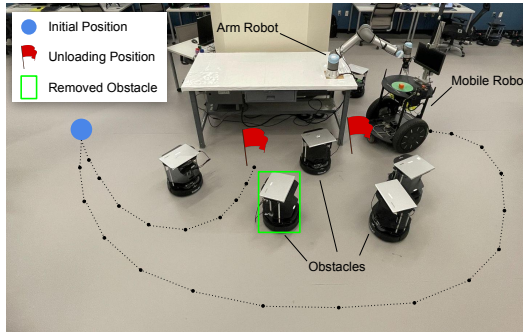

Fig. 5. The arm robot is placing an object onto the mobile robot in trial T_1. There are two loading positions on the south and east sides of the table, marked by red flags. The mobile robot's initial position is shown as the blue dot. The green box highlights the obstacle that was removed in T_2.

navigation actions. GROP incorporates both efficiency and feasibility, and produced the best overall performance.

Real Robot Demonstration: We demonstrate two trials of T_1 and T_2 using GROP on a real-robot platform. Instead of using a mobile manipulator, we used a robot system that includes two robots of a Segway-based mobile platform and a UR5e robot arm. The mobile robot started from an initial position, and was tasked with loading a distant object (an orange cube in our case) from the arm robot. The object was on the same table as the arm robot is, where the arm robot could pick the object, and place it onto the mobile robot to complete a loading behavior.

In trial T_1, the system computed the task-level feasibility values of loading from the south and east sides: 0.377 and 0.721. The corresponding costs were 7.5 and 19.3 respectively (distances of $4.5m$ and $11.6m$), where the robot moved at speed $0.6m/s$. GROP evaluated the utility values (7.6 and 9.5 in this case), and decided to load from the east side (less efficient but more feasible), as shown in Fig. 5.

In trial T_2, the robot system worked on the same task, while one obstacle (green box in Fig. 5) was removed from the environment. The obstacle removal changed the feasibility: Loading from the south has higher feasibility of 0.520, and a higher utility of 13.3. Accordingly, the mobile robot decided to load the object from the south, where there existed little chance of failing in the loading behavior but the overall efficiency was significantly improved. In both demonstration trials, the robot system succeeded in loading the object to the mobile platform.

VI. CONCLUSION AND FUTURE WORK

This paper introduces an algorithm, called **G**rounded **RO**bot Task and Motion **P**lanning (GROP), that considers both efficiency and feasibility for robot task-motion planning. GROP visually grounds spatial relationships to probabilistically evaluate action feasibility, and is particularly suitable for TAMP domains with long-term robot operations (e.g., long-distance navigation). We have extensively evaluated GROP in simulation using a mobile manipulator, and demonstrated it using a real robot system that includes a mobile robot and an arm robot. Results showed that GROP outperformed competitive baselines from the literature in plan efficiency without introducing additional action costs.

In this paper, we empirically evaluated the performance of GROP, while there is room to improve the evaluation through formal analysis, e.g., about its completeness and optimality. The difficulty comes from the different problem representations of task planning and motion planning. Also, the FCN-based feasibility evaluator is data-driven, where formal analysis is difficult. The current implementation of GROP relies on top-down views. It would be interesting to investigate the feasibility of applying egocentric vision to GROP. Due to the various viewpoints, we expect GROP to require a greater amount training data in this setting.

REFERENCES

[1] F. Lagriffoul, N. T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L. E. Kavraki, "Platform-independent benchmarks for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3765–3772, 2018.

[2] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[3] M. Ghallab, D. Nau, and P. Traverso, *Automated planning and acting*. Cambridge University Press, 2016.

[4] H. M. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, and R. C. Arkin, *Principles of robot motion: theory, algorithms, and implementation*, 2005.

[5] M. Toussaint, "Logic-geometric programming: an optimization-based approach to combined task and motion planning," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 1930–1936.

[6] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[7] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[8] Y. Jiang, F. Yang, S. Zhang, and P. Stone, "Task-motion planning with reinforcement learning for adaptable mobile service robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[9] A. Thomas, F. Mastrogiovanni, and M. Baglietto, "Mptp: Motion-planning-aware task planning for navigation in belief space," *Robotics and Autonomous Systems*, vol. 141, p. 103786, 2021.

[10] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *IEEE Robotics and Automation Letters*, 2019.

[11] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 844–851, 2020.

[12] J. McMahon and E. Plaku, "Robot motion planning with task specifications via regular languages," *Robotica*, 2017.

[13] Y. Zhao, Y. Li, L. Sentis, U. Topcu, and J. Liu, "Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments," *arXiv preprint arXiv:1811.04333*, 2018.

[14] S.-Y. Lo, S. Zhang, and P. Stone, "The petlon algorithm to plan efficiently for task-level-optimal navigation," *Journal of Artificial Intelligence Research*, vol. 69, pp. 471–500, 2020.

[15] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 335–346, 1990.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[17] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9563–9569.

[18] F. Gravot, S. Cambon, and R. Alami, "asymov: a planner that deals with intricate symbolic and geometric problems," in *The Eleventh International Symposium of Robotics Research*, 2005.

[19] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning." in *Robotics: Science and Systems*, 2007, pp. 326–333.

[20] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *IEEE International Conference on Robotics and Automation*, 2011.

[21] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 639–646.

[22] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *The International Journal of Robotics Research*, vol. 33, no. 14, pp. 1726–1747, 2014.

[23] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, "Guided search for task and motion plans using learned heuristics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 447–454.

[24] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Active model learning and diverse action sampling for task and motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4107–4114.

[25] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 793–812, 2019.

[26] R. Chitnis, L. P. Kaelbling, and T. Lozano-Pérez, "Learning quickly to plan quickly using modular meta-learning," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[27] Y. Ding, X. Zhang, X. Zhan, and S. Zhang, "Learning to ground objects for robot task and motion planning," in *IEEE Robotics and Automation Letters*, 2022.

[28] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Conference on Robot Learning*. PMLR, 2020, pp. 955–968.

[29] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[30] Y. Ding, X. Zhang, X. Zhan, and S. Zhang, "Task-motion planning for safe and efficient urban driving," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[31] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[32] D. Hadfield-Menell, E. Groshev, R. Chitnis, and P. Abbeel, "Modular task and motion planning in belief space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[33] C. Phiquepal and M. Toussaint, "Combined task and motion planning under partial observability: An optimization-based approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[34] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5678–5684.

[35] A. Nouman, V. Patoglu, and E. Erdem, "Hybrid conditional planning for robotic applications," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 594–623, 2021.

[36] A. Akbari, M. Diab, and J. Rosell, "Contingent task and motion planning under uncertainty for human–robot interactions," *Applied Sciences*, vol. 10, no. 5, p. 1665, 2020.

[37] D. Driess, J.-S. Ha, and M. Toussaint, "Deep Visual Reasoning: Learning to Predict Action Sequences for Task and Motion Planning from an Initial Scene Image," in *Proceedings of Robotics: Science and Systems*, 2020.

[38] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, V. Lifschitz *et al.*, "Bwibots: A platform for bridging the gap between ai and human–robot interaction research," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 635–659, 2017.

[39] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[40] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[41] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[42] M. Gelfond and Y. Kahl, *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[43] V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence*, vol. 138, no. 1-2, pp. 39–54, 2002.

[44] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, "Clingo= asp+ control: Preliminary report," *arXiv preprint arXiv:1405.3694*, 2014.