Deep Visual Heuristics: Learning Feasibility of Mixed-Integer Programs for Manipulation Planning

Danny Driess Ozgur Oguz Jung-Su Ha Marc Toussaint

***Abstract*— In this paper, we propose a deep neural network that predicts the feasibility of a mixed-integer program from visual input for robot manipulation planning. Integrating learning into task and motion planning is challenging, since it is unclear how the scene and goals can be encoded as input to the learning algorithm in a way that enables to generalize over a variety of tasks in environments with changing numbers of objects and goals. To achieve this, we propose to encode the scene and the target object directly in the image space.**

Our experiments show that our proposed network generalizes to scenes with multiple objects, although during training only two objects are present at the same time. By using the learned network as a heuristic to guide the search over the discrete variables of the mixed-integer program, the number of optimization problems that have to be solved to find a feasible solution or to detect infeasibility can greatly be reduced.

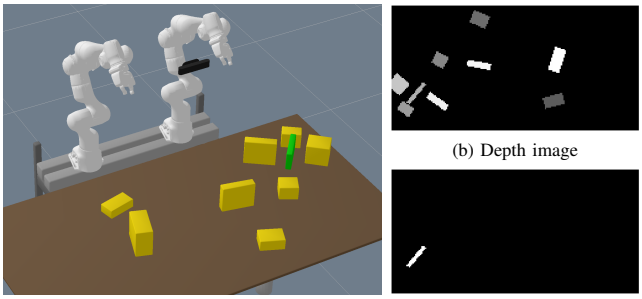
I. Introduction

In joint Task and Motion Planning (TAMP), it is common to combine discrete search on a symbolic, logical level with a geometric planner, which, given the discrete decisions, tries to find motions that fulfill the requirements induced by the high level task plan or returns that there is no feasible motion to realize this plan. Due to the combinatorial complexity of possible discrete decisions, a high number of geometric problems have to be solved to find an overall feasible plan. Indeed, for many typical TAMP problems, the majority of the computation time is spent in trying to generate motions or returning infeasibility of a decision [1], [2], [3], [4].

To overcome this, we seek to accelerate task and motion planning by learning a classifier that predicts the feasibility of the resulting geometric problem for a discrete decision. This way, by using the classifier as a heuristic for the search over discrete decisions, the hope is that only a very small subset of promising motion planning problems have to be solved to find a feasible motion.

One of the major challenges in integrating learning/experience into TAMP is the question of how the problem setting, e.g. the objects in the scene and goals, can be encoded as input to the learning algorithm [1], [5], [6]. Often, machine learning methods are integrated into robotic problems for a single task only. In this case, a fixed size feature representation might be sufficient. However, a remarkable property of TAMP approaches is that they generalize over a large variety of tasks in different environments with changing numbers of objects and goals, which makes fixed sized features not directly suitable.

In the present work, we argue that the feasibility of a geometric problem should be decided directly in the sensor space of the robot, instead of in a feature space. The sensor space not only has a fixed dimensionality and hence is suitable for standard learning algorithms. The sensory information also contains what the robot can extract from the current world state (not taking into account history or other priors). Therefore, this paper aims to answer the two following questions. First, whether it is possible to learn the feasibility of a geometric problem from sensory input like vision. Secondly, we investigate if the learned classifier can generalize to new problem instances that, for example, have a varying number of objects in the scene, although the classifier has been trained on only a small and fixed number of objects. This is especially interesting, since, as mentioned before, the combinatorial complexity of our world prohibits to generate a rich enough dataset that covers sufficiently large property combinations of multiple objects.



(a) Scene (b) Depth image (c) Object mask

Fig. 1. Generalization test scenario with 9 objects. The task is to grasp and pick up the green box in (a). (b) and (c) show the input image to the neural network, which is captured by the black camera in (a).

More specifically, we propose a deep neural network that predicts the feasibility of mixed-integer programs, which are one way to realize TAMP, where the discrete action (integer) variable represents the abstract decision and the resulting nonlinear trajectory optimization problem the geometric part. The input to our neural network is a multi-channel image and the discrete action. The image consists of a depth image of the scene and object centric masks in the image space, which enable to encode the objects that are involved in the action. In the experiments, we consider the problem of grasping box-shaped objects in a scene with two robot arms (Fig. 1). The discrete actions encode how and with which robot a box should be grasped. Infeasibility of an action can occur due to the fact that a box is kinematically out of reach for the chosen grasping or that other objects obstruct this action.

In our experience, feasible problems usually converge quickly, whereas detecting infeasibility reliably takes longer. At the same time, for the problems we considered, the majority of geometric problems are actually infeasible. Therefore, we propose two methods how the learned classifier can

Machine Learning and Robotics Lab, University of Stuttgart, Germany.
Max Planck Institute for Intelligent Systems, Stuttgart, Germany.
Danny Driess is supported by the IMPRS for Intelligent Systems.

guide the discrete search: On the one hand as an admissible heuristic that cannot produce false infeasibles, but does not reduce the number of to-be-solved optimization problems for an infeasible problem. On the other hand as a non-admissible heuristic that greatly reduces the number of motion planning problems in case of an overall infeasible scene, but can prevent finding a solution to a feasible problem. In the experiments, we discuss the implications of both methods.
To summarize, our main contributions are:

- We propose a deep neural network that predicts the feasibility of a mixed-integer program from visual input for manipulation planning in a tabletop scenario.
- We show that the network generalizes to scenarios with more objects in the scene than it has been trained on.
- We develop how to use the network to guide the discrete search as an admissible and non-admissible heuristic.

II. Related Work

A. Task and Motion Planning

Many TAMP approaches combine logic search for task planning with sampling based motion planners [7], [8], [9], [10] or rely on a discretization of the configuration/action space in order to utilize constraint satisfaction methods [11], [12], [13]. Another recent method for TAMP is so-called Logic Geometric Programming (LGP) [2], an optimization based approach where the logic imposes a skeleton of active constraints on a nonlinear program, which showed to be able to integrate physical reasoning into TAMP [14], [15]. Our mixed-integer approach to TAMP is a simplified version of this formulation and uses the same solver as in [2].

B. Mixed-Integer Programs in Robotics

Mixed-integer programs play an important role in robotics to formalize the hybrid nature of contacts and interaction modes [16], [17]. The aforementioned LGP [2] can also be understood as a generalization of a mixed-integer program in the context of TAMP. One major problem with such formulations is the combinatorial complexity that is introduced through the discrete variables. Hogan et al. [16] try to solve this problem by learning a classifier that predicts the integer assignments of hybrid MPC for planar pushing of the same object. There is no generalization to different scenarios.

C. Learning to Plan

Advances in deep neural networks have enabled to create learning systems that plan from (raw) sensory input such as images. One line of research focuses on learning compact representations suitable for planning or control directly from raw sensory inputs [18], [19], [20], [21], [22], [23], [24]. The compactness of the learned representation enables to utilize reinforcement learning or local planners efficiently.

Another approach to planning from raw sensory input is to learn an action conditioned predictive model [25], [26], [27], [28], [29]. Here the main idea is that the learned model predicts the future state of the environment in the sensor space after an action has been taken. This way, planning can be performed by imagining the outcome of an action.

The major difference of these approaches to our work is that they attempt to learn low level actions for similar tasks, while we are interested in accelerating the search over high level decisions for a mixed-integer program, generalizing to scenes with different numbers of objects.

D. Learning for Task and Motion Planning

Integrating learning as a heuristic has the potential to greatly speed up the discrete search in TAMP [1], [30], [31], [5]. As mentioned in the introduction, finding the right encoding of the scene and goals is challenging if similar generalization capabilities of TAMP are desired. For example, [5] propose so-called score-spaces to compare problem instances to reuse experience to guide TAMP.

Similarly to the methods of the last subsection, in [32] a predictive model of a box-stacking scenario is learned that can be utilized for planning sequences of high level actions directly in the image space. However, a main limitation of their work is that the scene always contains the same four objects with the same colors, which allows them to have separate actions for each object. We aim at a more general formulation that is not restricted to a predefined number of objects, however, we do not consider action sequences.

The work of Wells et al. [1] is most closely related to our work. They propose to learn a classifier that predicts the feasibility of finding a motion path (via RRT) to a prism that should be grasped. The input to the classifier is a feature representation of exactly two objects in the environment (size and relative position). A major contribution of their approach is the generalization to new environments by approximating all objects with prisms and performing pair-wise feasibility checks between all objects. In contrast, our proposed classifier can directly be evaluated on multiple objects simultaneously. Furthermore, we consider also rotated objects and the classifier has to learn reachability for two robot arms. In the experimental section, we compare our image space based approach with the feature based approach of [1].

Since in the experiments we consider grasping, there are connections to Dex-Net [33] or others [34]. Dex-Net [33] aims at learning the probability of success for grasping from a depth image. On the one hand, their approach is more general, since they consider grasping arbitrarily shaped objects. On the other hand, our approach is more general, since Dex-Net neither considers reachability of objects nor is able to directly handle other objects in the vicinity of the object that should be grasped. Furthermore, our approach takes 12 different graspings from different sides into account. In [34], object target masks are used, similar to our work.

To our knowledge, the present work is the first that learns the feasibility of a mixed-integer program within TAMP from scene images, while being able to generalize to multiple objects in the scene.

III. Mixed-Integer Program for Robot Trajectory Planning

We start by describing the trajectory optimization framework for which we want to predict its feasibility. Let $\mathcal{X} \subset \mathbb{R}^n \times SE(3)^{n_O}$ be the configuration space of all objects and articulated structures (e.g. robot arms) in the scene S with initial condition $\tilde{x}_0$. The goal is to find a path $x : [0, T] \to \mathcal{X}$,

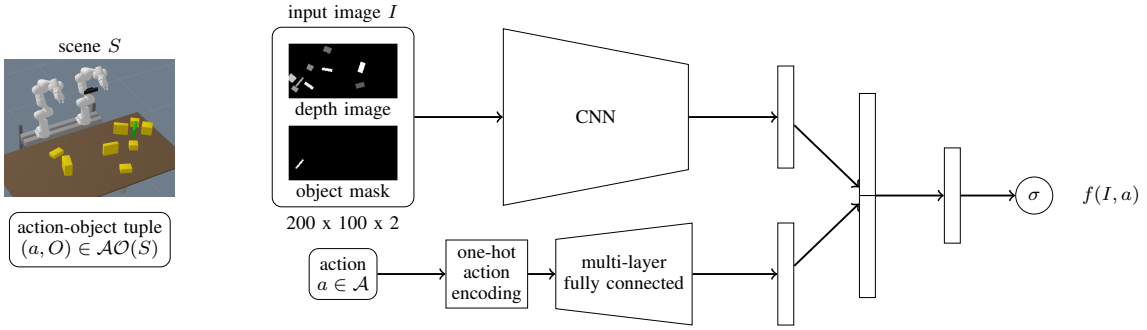


Fig. 2. Proposed neural network architecture to predict the feasibility of an action-object tuple $(a, O) \in \mathcal{AO}(S)$ in the scene S.

$x \in C^2$ in the configuration space that minimizes

$$P(S,O) = \min_{\substack{x:[0,T]\to\mathcal{X} \\ a\in\mathcal{A}}} \int_0^T c(t, x(t), \dot{x}(t), \ddot{x}(t), a, O)\,\mathrm{d}t \quad \text{(1a)}$$

$$\text{s.t.} \qquad x(0) = \tilde{x}_0 \quad \text{(1b)}$$

$$\forall_{t\in[0,T]} \;:\; h(t, x(t), \dot{x}(t), a, O) = 0 \quad \text{(1c)}$$

$$\forall_{t\in[0,T]} \;:\; g(t, x(t), \dot{x}(t), a, O) \le 0 \quad \text{(1d)}$$

$$(a, O) \in \mathcal{AO}(S). \quad \text{(1e)}$$

The costs c, equality constraints h and inequality constraints g are parameterized by an action-object tuple $(a, O) \in \mathcal{AO}(S) \subset \mathcal{A} \times \mathcal{O}(S)$, where $a \in \mathcal{A}$ is a discrete action variable of the categorical set $\mathcal{A}$ and $\mathcal{O}(S)$ are the objects in the scene S. The set $\mathcal{AO}(S)$ contains all possible action-object combinations. Note that it is straightforward to generalize this formulation to actions that involve more than one object, but since in the experiments we only consider actions involving a single object like grasping, we simplify the description to action-object pairs. Of course, the scene could still contain multiple additional objects.

When fixing a specific action-object tuple (a, O), the constraints h, g are smooth functions in $x, \dot{x}$, yielding a remaining smooth nonlinear program (NLP), which we denote by $P(S, a, O)$. Ideally, formulating the problem as a mixed-integer program is realized in a way that renders the resulting NLP $P(S, a, O)$ for fixed discrete variables solvable with (local) iterative constrained optimization methods such as augmented Lagrangian or interior point methods. In the context of TAMP, the action-object tuple (a, O) assignment corresponds to a decision on the symbolic level, the resulting NLP $P(S, a, O)$ is the geometric planner that tries to optimize a motion that fulfills the requirements induced by the action-object tuple. For example, a could be the decision that a certain robot arm should grasp an object O. The NLP $P(S, a, O)$ then attempts to optimize a collision free trajectory such that the robot grasps the object with a certain strategy imposed by a.

For a specific action-object tuple in the scene S, we define the feasible set of (1) as

$$\mathcal{F}_S(a, O) = \big\{ x \in C^2([0,T], \mathcal{X}) \;:\; \text{(1b)}, \text{(1c)}, \text{(1d)} \big\}. \quad (2)$$

Feasibility of the specific NLP $P(S, a, O)$ is expressed by the function

$$F_S(a, O) = \begin{cases} 0 & \text{if } \mathcal{F}_S(a, O) = \emptyset \\ 1 & \text{otherwise} \end{cases}, \quad (3)$$

whereas the complete scene S with target object O is considered feasible, if there exists at least one feasible action

$$F(S, O) = \begin{cases} 0 & \text{if } \forall_{a\in\mathcal{A}} \;:\; F_S(a, O) = 0 \\ 1 & \text{if } \exists_{a\in\mathcal{A}} \;:\; F_S(a, O) = 1 \end{cases}. \quad (4)$$

Note that feasibility as defined here is a property of the nonlinear program. However, in practice, the ground truth feasibility is determined by whether the numerical optimizer finds a feasible solution, which is not always the case for a feasible problem, since it is, even after introducing the discrete variables, still non-convex. We use the same augmented Lagrangian trajectory optimization method as in [2] to solve the NLP for a fixed discrete action. Indeed, a generalized version of the trajectory optimization framework of the present work is so-called Logic Geometric Programming [2], where a sequence of action-object tuples is considered.

IV. Deep Visual Heuristics

The goal of the present work is to learn a classifier f that predicts the feasibility of the nonlinear program $P(S, a, O)$ for an action-object tuple $(a, O) \in \mathcal{AO}(S)$ in the scene S, i.e. f should approximate $F_S(a, O)$ as defined in (3) for (1).

In order to realize this, the question is how the scene S, the action $a \in \mathcal{A}$ as well as the object $O \in \mathcal{O}(S)$ that is involved in the action can be encoded as input to the classifier in a way that not only enables to learn an accurate classifier, but also generalizes to new scenes with, for example, changing numbers of objects. As mentioned in the introduction, we argue that the sensor space, in this case vision, has the potential to fulfill these requirements, while being a fixed size input. Formally, we assume that there is a generative process that produces a depth image $I_{\text{depth}} \in \mathbb{R}^{w\times h}$ from the scene S, either via a rendering engine in simulation or a depth camera in the real world. Since feasibility of the problems we consider mainly depends on geometric information, a depth image is a natural choice. If there are multiple objects in the scene, a crucial question is how the target object O of an action a is encoded, since a depth image alone would show all objects. Separating the object from the action is also important if the classifier should generalize to scenes with different objects.

The idea is to add a binary mask $I_{\text{mask}} \in \{0, 1\}^{w\times h}$ of the target object as a second image to the input of the classifier in the same image space as I_{depth}. This way, an attention mechanism to the target object is created and the actions are separately represented from the concrete object. The object mask alone, however, would also not be sufficient, since the

feasibility of an action-object pair (a, O) can depend on other objects in the scene or on the exact shape (like height) of the target object, which is not provided by the mask alone, but through the depth image. Being able to generate such object masks is a reasonable assumption, since there are many methods dealing with this task (e.g. [35]).

Therefore, the input to the classifier $f(I, a)$ is the action $a \in \mathcal{A}$ and the stacking of I_{depth} and I_{mask} as the two channel image $I = I(S, O) \in \mathbb{R}^{w \times h \times 2}$, which itself is a function of the scene S and the involved object O. Fig. 1 visualizes these two input images. We parameterize $f(I, a)$ as a deep neural network, whose architecture is illustrated in Fig. 2. The network is trained to approximate the probability that the action-object pair (a, O) in the scene S is feasible

$$f(I(S, O), a) = p(F_S(a, O) = 1 \mid I(S, O), a) \tag{5}$$

on the standard weighted binary cross-entropy loss

$$L(w) = -\sum_{(S,O,I)\in\mathcal{D}} \sum_{a\in\mathcal{A}} \eta \; F_S(a, O) \log(f(I, a; w)) + (1 - F_S(a, O)) \log(1 - f(I, a; w)). \tag{6}$$

The training data $\mathcal{D} = \{(S_i, O_i, I(S_i, O_i))\}_{i=1}^{d}$ consists of scenes S_i with target objects O_i. All actions are taken into account in the loss (6) for each scene-object sample. Since in our experiments the majority of actions is infeasible, the weighting factor $\eta \geq 1$ in (6) turned out to be very important to balance the loss, otherwise the network could achieve high accuracies by always predicting infeasibility. If the prediction $f(I(S, O), a) > \beta$ is higher than the feasibility threshold $\beta > 0$, the network decides that $F_S(a, O) = 1$. In the experiments, we discuss the influence of this threshold.

The specific network architecture used in the experiments is as follows. The CNN part are two convolutional layers with a filter size of 5 x 5, followed by max-pooling of size 2 and stride 2. The first convolution layer has 5 filters, the second one 10. The output of the CNN is flattened and passed through a fully connected layer to produce a feature size of 500. The action encoder consists of two fully connected layers with a final feature size of 500. Both features are concatenated and passed through one additional fully connected layer with size 100, before the final linear layer with one sigmoid output. All hidden layer use ReLUs.

V. Guiding Motion Planning using Learned Visual Heuristics

The learned classifier (5) can be utilized not only to predict the feasibility of a given action-object tuple in a scene, but also to guide the optimization of the mixed-integer program (1) as a heuristic, since it outputs a feasibility probability. The goal of this heuristic is to solve $P(S, O)$, i.e. to find an action $a \in \mathcal{A}$ for a scene S and a target object $O \in \mathcal{O}(S)$ such that the NLP $P(S, a, O)$ is feasible or return that the complete scene is infeasible $F(S, O) = 0$, while reducing the number of NLPs that have to be solved as much as possible.

A. Admissible Heuristic

One way is to use the classifier as an admissible heuristic. Admissible in this context denotes that classification errors cannot prevent finding a solution to a feasible problem $P(S, O)$. This is realized by first evaluating $f(I, a) = p_a$ for all $a \in \mathcal{A}$. Then first the NLP $P(S, a, O)$ with the highest predicted feasibility p_a is solved. If it is feasible, a solution is found, if not, the next largest p_a is chosen and so on. This implies that for a feasible problem, only one NLP would have to be solved for a perfect classifier. However, for an infeasible problem, i.e. a problem where none of the action choices lead to a feasible NLP ($F(S, O) = 0$), all actions would be tested and one therefore does not gain anything when using the learned network as an admissible heuristic.

B. Non-Admissible Heuristic

As mentioned in the introduction, in our experience, many actions or even the complete scene is infeasible. In addition, it takes usually more time for the optimizer to detect infeasibility than to solve a feasible problem. Therefore, we also propose how the learned classifier can be used in a way that enables to detect infeasibility of a scene very quickly, hence being able to save a lot of computation time. The principal algorithm is the same as in the case of an admissible heuristic (see last section). However, the network predicts that the scene S with target object O is infeasible, i.e. $F(S, O) = 0$, if it holds for the remaining actions $\mathcal{A}' = \mathcal{A} \backslash \{a_1, \ldots, a_j\}$ after having tested the actions $a_1, \ldots, a_j$ that

$$\max_{a\in\mathcal{A}'} f(I(S, O), a) < \beta \tag{7}$$

with the feasibility threshold $0 < \beta < 1$. Therefore, if, for example, the highest predicted probability of all actions is below this threshold, not a single NLP has to be solved to detect infeasibility of the scene. Of course, in this case classification errors can lead to a situation where a feasible action is not evaluated and therefore the scene is erroneously classified as infeasible (false infeasible rate). In the experiments, we evaluate and discuss the false infeasible rate and the savings with respect to the feasibility threshold. For $\beta = 0$, this method becomes equivalent to the admissible heuristic of Sec. V-A.

VI. Experiments

We demonstrate our approach on the problem of grasping box-shaped objects with two robot arms that have parallel grippers. Fig. 1 shows a typical scene, where the task is to grasp the green box. To model grasping box-shaped objects as a mixed-integer program, the 6 faces of a box introduce 6 fundamentally different ways, i.e. integer assignments or actions a, how a parallel gripper can grasp the box, in the sense that the solution sets are not connected without violating the constraints during a transition (re-grasping). Furthermore, for each of the 6 faces we additionally have 4 alignment constraints of the end-effector orthogonally to a chosen face. The optimizer still has two degrees of freedom to choose the exact grasping location. Since in the experiments we only consider objects on a table, we can exclude grasping actions where the gripper would always collide with the table, which leaves 12 different actions for one robot arm, i.e. 24 in total. Fig. 3 visualizes these 12 grasping actions, where one can also see the remaining degrees of freedom. The path costs in (1) are squared accelerations and there are collision and joint limit inequality constraints.

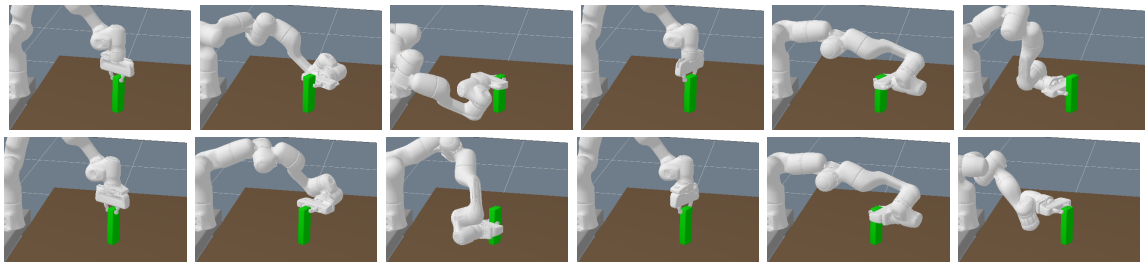


Fig. 3. Visualization of the 12 different discrete grasping actions for one robot arm for which the feasibility for a target object should be predicted.

A. One Object

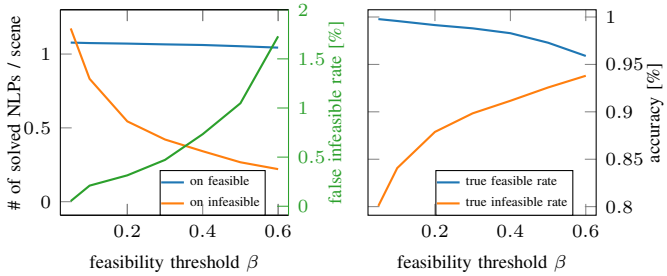
We first consider scenes with only one object.

1) Training and Test Data Generation: The training data consists of 19278 scenes where the position of the box, its rotation and size are chosen on a grid. Furthermore, Gaussian noise is added to all of those parameters. 11.2% of the 462672 actions in total were feasible, 88.8% infeasible. This imbalance imposes a challenge to the learning algorithm. To account for this imbalance in the training dataset, we use $\eta = 10.0$. The network is trained with the ADAM optimizer (learning rate 0.0005, batch size 10). We generate two different test datasets consisting of 3672 scenes each. The first one, test data 1, uses the same grid points as for the training data, but a different random seed. Test data 2 is generated by uniformly sampling the parameters of the box (position, rotation, size). About half of the test scenes are completely infeasible, the feasible-infeasible ratio of all actions is approximately 11% to 89%.

2) Accuracy and Saved NLPs: Tab. I lists the results for both test datasets. Looking at the true feasible and true infeasible rate over all actions, choosing $\eta = 10.0$, the network not only resolves the imbalance in the training data, but is also biased towards a low false infeasible rate, which allows us to use the network as a non-admissible heuristic. Depending on the feasibility threshold, a true feasible accuracy of over 98% and over 91% true infeasible one can be achieved, even for the off-distribution test data 2. When using the network to guide the search as discussed in Sec. V, the network reduces the number of to-be-solved NLPs to find a feasible solution by a factor of 4.8 compared to random selection. Indeed, only 1.1 NLPs on average have to be solved for a feasible problem, which is very close to the optimum of 1. Also note the very large standard deviation for random selection compared to the low one with the learned classifier. As one can see in Fig. 4a, by adjusting the feasibility threshold β, a low false infeasible rate can be achieved while still being able to detect infeasibility very quickly, e.g. speedup of 48 with false infeasible rate of 0.3%, which justifies the use of the network as a non-admissible heuristic.

B. Comparison to Feature-Based Approach

Here we compare to an SVM based approach similar to [1] with features instead of images as input. As in [1], a separate SVM for each of the 24 actions is trained with LIBSVM [36]. The input is a 6 dimensional vector containing the x, y position, the rotation and the size of the box. The test and training scenes are the same as for the neural network. Due to the imbalanced dataset, like for the neural network, a weighting in the loss of the SVM is necessary, otherwise it performed extremely poorly. For a fair comparison, we spent a decent amount of time to tune the parameters of the SVM. As one can see in Tab. I, an SVM feature based approach is competitive with respect to the accuracies on all actions, although our network performs slightly better. However, if one considers the number of solved NLPs, one can see that using the SVM as an admissible heuristic takes nearly twice the amount of NLPs to find a feasible solution or 1.6 times more in the non-admissible case. If $\beta = 0.1$ for the network, the SVM has a 15 times higher false infeasible rate for the off-distribution test data 2, although the network still requires less NLPs to detect infeasibility. Therefore, our approach in image space is significantly better in this case. Although we focus on grasping boxes in this work, another advantage of image based approaches is that they could be utilized for arbitrarily shaped object, which is not obvious for a feature-based approach that relies on a specific parameterization of the object shape.



(a) Number of solved NLPs (b) Accuracy on all actions

Fig. 4. Test data 2 evaluation. (a) shows the number of NLPs necessary to find a feasible solution or to detect infeasibility.

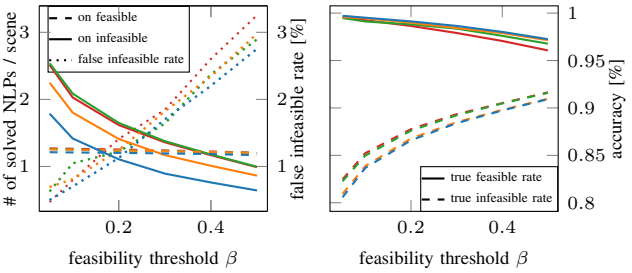
C. Multiple Objects

Now we consider scenarios where there are additional objects in the scene which might or not obstruct a certain grasping action. As mentioned in the introduction, it is not realizable to generate a rich enough dataset with combinations of different numbers of additional objects with different sizes and positions. Therefore, we generate a training dataset with only *one* additional object, again box-shaped. Half of the scenarios are generated by randomly sampling the position and orientation of the second box. For the other half, the second box is randomly positioned in the vicinity around the first object that should be grasped. This way, we ensure that the dataset contains enough samples where the presence of the second object actually creates additional infeasibilities. The size of the second object is always uniformly sampled. In total, the training data contains 22490 scenes, again only 10% of all actions are feasible.

1) Generalization to more than Two Objects: To test the generalization capabilities of our network, we test on scenes

TABLE I ACCURACIES ON ALL ACTIONS AND NUMBER OF SOLVED NLPS PER SCENE FOR SCENARIOS WITH A SINGLE OBJECT. TRUE (IN)FEASIBLE MEANS CLASSIFYING AN (IN)FEASIBLE ACTION AS (IN)FEASIBLE. FALSE INFEASIBLE MEANS CLASSIFYING A FEASIBLE SCENE AS INFEASIBLE.

		accuracy on all actions		number of solved NLPs per problem instance						
				on feasible problems			on infeasible problems			false infeas.
	β	true feas.	true infeas.	random	admissible	non-admissible	random	admissible	non-admissible	(scenes)
test data 1	DVH 0.05	99.9%	79.2%	5.1±4.4	1.1±0.6	1.1±0.6	24	24	1.0±2.7	0.1%
	DVH 0.1	99.7%	83.1%	5.1±4.4	1.1±0.6	1.1±0.6	24	24	0.8±2.2	0.2%
	DVH 0.2	99.5%	86.9%	5.1±4.4	1.1±0.6	1.1±0.5	24	24	0.5±1.5	0.3%
	DVH 0.3	99.0%	88.9%	5.1±4.4	1.1±0.6	1.1±0.5	24	24	0.4±1.3	0.6%
	DVH 0.4	98.5%	90.3%	5.1±4.4	1.1±0.6	1.1±0.5	24	24	0.3±1.1	0.9%
	DVH 0.5	97.3%	91.8%	5.1±4.4	1.1±0.6	1.1±0.5	24	24	0.2±0.8	1.5%
	SVM	98.0%	91.6%	5.1±4.4	1.9±1.8	1.8±1.2	24	24	0.9±2.0	1.7%
test data 2	DVH 0.05	99.8%	80.1%	5.3±4.7	1.1±0.5	1.1±0.5	24	24	1.2±2.7	0.1%
	DVH 0.1	99.6%	84.1%	5.3±4.7	1.1±0.5	1.1±0.4	24	24	0.8±2.1	0.2%
	DVH 0.2	99.1%	87.9%	5.3±4.7	1.1±0.5	1.1±0.4	24	24	0.5±1.5	0.3%
	DVH 0.3	98.8%	89.8%	5.3±4.7	1.1±0.5	1.1±0.4	24	24	0.4±1.3	0.5%
	DVH 0.4	98.3%	91.2%	5.3±4.7	1.1±0.5	1.1±0.4	24	24	0.3±1.1	0.7%
	DVH 0.5	97.3%	92.6%	5.3±4.7	1.1±0.5	1.1±0.4	24	24	0.3±0.9	1.0%
	SVM	96.6%	91.5%	5.3±4.7	2.1±2.3	1.8±1.2	24	24	0.9± 2.0	3.0%



(a) Number of solved NLPs (b) Accuracy on all actions

Fig. 5. Generalization to multiple objects. Blue 1, orange 2, green 5 and red 8 additional objects to the one that should be grasped.

that contain 1 (as in training), 2, 5 and 8 additional objects to the target box in the scene. Fig. 5 shows the accuracy and number of solved NLPs. As one can see, although the false infeasible rate and also the number of NLPs required to detect infeasibility is slightly higher with an increasing number of additional objects, the network is still very accurate and a useful heuristic. Finding a feasible solution is even nearly independent on the number of additional objects, which is remarkable. As a sanity check, if 7 of the 8 additional objects are in the scene but not rendered, the true infeasible rate drops by 8% and about twice as much NLPs have to be solved to detect infeasibility of an infeasible scene in the non-admissible case. Therefore, it really generalizes. However, comparing the results to the case of a single object only, one can see that the false infeasible rate on the complete scene is higher when the network is trained or tested on multiple objects. Still, the value is low enough to justify using it as a non-admissible heuristic. As a side note, if the network is trained only on a single object as in the previous section, then the predictions get totally confused if an additional object is placed in the scene (70% false infeasibles). Therefore, training with at least one additional object is necessary.

2) Cylinder-shaped Object: Although the training data set only consists of box-shaped objects, we were interested what happens if shapes other than boxes are present in the scene. We generate 2000 test scenes where cylinders of different sizes (height, radius) are placed together with the box that should be grasped. 983 scenes contain at least one feasible action, 1017 are completely infeasible, because either the cylinder prevents grasping the box or the box is out of reach. For a feasibility threshold of 0.5, the true feasible accuracy is 98.3%, the true infeasible rate is 89.4%. To show that this is not a statistical effect of the test data distribution, we generated the same test data, but this time the cylinder is not rendered in the depth image, although present in the scene. Depending on the feasibility threshold, the true infeasible rate drops between 2 and 3%. Hence, the network can generalize to cylinders to some extend. More importantly, the network does not get confused if other shapes are present.

D. Runtime Improvements

So far, we have focused on how many NLPs have to be solved to find an overall feasible solution or decide that the complete scene is infeasible, since this is a metric that is invariant to the concrete solver implementation or its stopping criteria. For a typical feasible action, solving one NLP takes between 0.5 and 1.1 seconds. For an infeasible action, it can take up to 17 seconds, which means a significant runtime improvement in the non-admissible case. Querying the SVMs for all actions takes about 5 ms, the network 2 ms on GPU (10 ms on CPU), again for all actions, including computing the image feature encoding. Compared to solving an NLP, the querying times of the network and the SVM are neglectable.

VII. CONCLUSION

In the present work, we have shown that it is possible to predict the feasibility of a mixed-integer program from visual input with high accuracy. Although being trained on only two objects present at the same time in the scene, the network generalizes to multiple additional objects and still achieves a high accuracy and significantly reduces the number of NLPs that have to be solved to find a feasible solution to the mixed-integer program. The object mask is a crucial component of the system to allow this generalization to multiple objects.

While an SVM based approach with geometric features as input is competitive in the accuracy of the classifier, it performed inferior as a heuristic to guide the search and therefore saved about twice less NLPs than our proposed method in the image space. Furthermore, this feature based approach cannot directly generalize to multiple objects.

One limitation of our work is that the considered scenes in the experiments have to be captured well through visual input without occlusions or other ambiguities.

REFERENCES

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

[2] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proc. of Robotics: Science and Systems (R:SS)*, 2018.

[3] D. Driess, O. Oguz, and M. Toussaint, "Hierarchical Task and Motion Planning using Logic-Geometric Programming (HLGP)," RSS Workshop on Robust Task and Motion Planning, 2019.

[4] I. Rodriguez, K. Nottensteiner, D. Leidner, M. Kasecker, F. Stulp, and A. Albu-Schäffer, "Iteratively refined feasibility checks in robotic assembly sequence planning," *Robotics and Automation Letters*, 2019.

[5] B. Kim, L. Kaelbling, and T. Lozano-Perez, "Learning to guide task and motion planning using score-space representation," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[6] C. Garrett, L. Kaelbling, and T. Lozano-Perez, "Learning to rank for synthesizing planning heuristics," in *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2016.

[7] S. Alili, A. K. Pandey, E. A. Sisbot, and R. Alami, "Interleaving symbolic and geometric reasoning for a robotic assistant," in *ICAPS Workshop on Combining Action and Motion Planning*, 2010.

[8] L. de Silva, A. K. Pandey, M. Gharbi, and R. Alami, "Towards combining HTN planning and geometric task planning," *arXiv preprint 1307.1482*, 2013.

[9] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. J. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[10] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *Int. Journal of Robotics Research (IJRR)*, 2018.

[11] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson, "Constraint propagation on interval bounds for dealing with geometric backtracking," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[12] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *Int. Journal of Robotics Research (IJRR)*, 2014.

[13] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[14] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," arXiv:2002.12780, 2020.

[15] J.-S. Ha, D. Driess, and M. Toussaint, "Probabilistic framework for constrained manipulations and task and motion planning under uncertainty," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[16] F. R. Hogan, E. R. Grau, and A. Rodriguez, "Reactive planar manipulation with convex hybrid MPC," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[17] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Proc. of the Int. Conf. on Humanoid Robots*, 2014.

[18] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. of the Conf. on Neural Information Processing Systems*, 2015.

[19] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[20] B. Boots, S. M. Siddiqi, and G. J. Gordon, "Closing the learning-planning loop with predictive state representations," *Int. Journal of Robotics Research (IJRR)*, 2011.

[21] S. Lange, M. A. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, 2012.

[22] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, *et al.*, "The predictron: End-to-end learning and planning," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2017.

[23] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *Robotics and Automation Letters*, 2019.

[24] J.-S. Ha, Y.-J. Park, H.-J. Chae, S.-S. Park, and H.-L. Choi, "Adaptive path-integral autoencoders: Representation learning and planning for dynamical systems," in *Advances in Neural Information Processing Systems*, 2018.

[25] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[26] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," in *Proc. of the Conference on Robot Learning (CoRL)*, 2017.

[27] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2017.

[28] R. Pascanu, Y. Li, O. Vinyals, N. Heess, L. Buesing, S. Racanière, D. P. Reichert, T. Weber, D. Wierstra, and P. W. Battaglia, "Learning model-based planning from scratch," *arXiv preprint 1707.06170*, 2017.

[29] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, *et al.*, "Imagination-augmented agents for deep reinforcement learning," in *Advances in neural information processing systems*, 2017.

[30] 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 *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[31] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning feasibility constraints for multicontact locomotion of legged robots," in *Proc. of Robotics: Science and Systems (RSS)*, 2017.

[32] C. Paxton, Y. Barnoy, K. D. Katyal, R. Arora, and G. D. Hager, "Visual robot task planning," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2019.

[33] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. of Robotics: Science and Systems (RSS)*, 2017.

[34] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[35] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "Mask R-CNN," in *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2017.

[36] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.