Learning Spatial Preconditions of Manipulation Skills using Random Forests

Oliver Kroemer and Gaurav S. Sukhatme

Abstract— **Robots working in everyday and unstructured environments will need to perform manipulation skills using different sets of objects. To determine if a manipulation skill can be executed in a given situation, a robot will need to learn the preconditions of the skill. The robot will need to check both that the required objects are present in the scene and that they are arranged in a suitable manner for the skill to be executed.**

We propose a random forest approach to learn the set of spatial configurations of objects that fulfill a skill's preconditions. We also explore how parts of objects and interactions between parts can be incorporated into the scene models to improve the generalization performance. The proposed approach was evaluated on the preconditions of six manipulation skills. The experiments show that using the ensemble approach, and including the parts and interactions, results in an increase in accuracy of 16.4%.
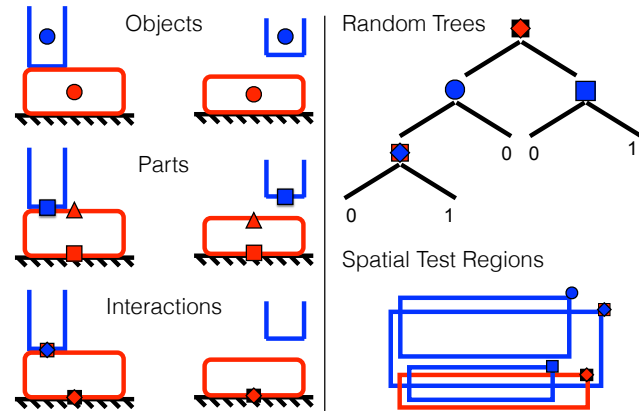
Fig. 1. The figure presents an overview of the proposed approach. (Left) The illustrations show the three types of scene elements used to model the robot's surroundings. (Right) Configurations of scene elements are classified as valid or invalid for executing a given skill by using random forests. Each random tree generates a hierarchical set of test regions to detect the presence of specific scene elements in certain absolute or relative regions of the elements' feature space.

I. INTRODUCTION

The conditions required to execute a skill and achieve its predicted effects, are known as the skill's preconditions. A robot can learn the preconditions of a skill to determine in which situations the skill is applicable. It can also use the preconditions to sequence skills, with the goal state of one skill being within the preconditions of the next skill [1].

The preconditions of manipulation skills depend both on the types of objects present in the scene as well as the spatial configuration in which these objects are arranged. For example, a cutting skill cannot be applied to an apple and an orange because an orange, unlike a knife, does not afford cutting [2]. The cutting skill also cannot be executed with an apple and a knife if the edge of the knife is not placed in contact with the fruit. Our work focuses on learning the latter spatial preconditions, i.e., the set of of object configurations in which the skill can be executed.

Since robots will need to manipulate a variety of objects, the learned preconditions should generalize between different sets of objects, e.g., the robot should adapt the position of the knife relative to the fruit when using a larger knife. The learned preconditions also need to generalize to scenes with missing or additional objects, e.g., some scenes will not include a knife. We assume that the correspondences between the objects in the different scenes are known and of the same valid type, e.g., a knife may be replaced with a smaller knife in another scene, but not with an orange. Our focus is on how the variations in the objects' shapes and sizes affect the set of spatial preconditions.

We propose a method for learning the spatial preconditions of manipulation skills using random forests. We frame the precondition learning as a classification problem, wherein the robot has to classify a configuration of objects as either affording or not affording the skill. To better generalize between objects with different shapes and sizes, the robot models the task-relevant parts of objects and the interactions between these parts, as illustrated in Fig. 1. We refer to the objects, parts, and interactions jointly as the *scene elements*. The robot learns the spatial preconditions over the scene elements using random forests [3], [4]. Each non-leaf node of a random tree evaluates if a selected scene element is in a specific region in the workspace, e.g., whether the cup object is located within a 3D volume above the saucer object. By combining multiple of these tests, each tree provides an estimate of the skill's spatial preconditions. There may be multiple sets of conditions that can explain the differences between the valid and invalid scenes when given a limited amount of training data. By using an ensemble of multiple trees, the robot captures multiple sets of possible preconditions and thus reduce the chances of overfitting.

The proposed method was evaluated on the preconditions of six manipulation skills, including placing, cutting, and wiping. The results of the experiments show that using the ensemble approach and including the parts and interactions results in a 16.4% increase in accuracy. The experiments are described in Section III.

A. Related Work

Learning the preconditions of actions is closely related to symbol and predicate learning, as the agent learns discrete abstract states [5], [6], [7], [8], [9]. The robot may learn symbols for spatial relationships such as *on* or *next to*. The symbols are usually learned either by clustering states or in a supervised manner. The learned symbols can subsequently be used by a symbolic planner to generate a sequence of skills for performing tasks. The robot can also directly learn the preconditions and effects of individual skills to sequence them [10], [1]. The precondition learning is treated as a binary classification problem. Most of the work in predicate learning has focused on learning the conditions for a fixed set of objects. As the part-object transformations are therefore constant, the relative poses between parts can be implicitly modeled by the relative poses between objects. We explore the potential benefits of incorporating object parts and interactions into the preconditions model when generalizing between different objects.

If an object can be used to perform an action, then the object is said to *afford* the action [2]. In this manner, objects can be seen as resources for enabling agents' actions. Research into affordance learning has explored a wide range of manipulations and interactions including grasping [11], [12], pushing [13], [14], stacking [15], [16], rolling [17], and containing [18]. Interactions between objects in the scenes will result in additional environmental constraints that may change the set of afforded actions [19]. Most of the work in this research area has focused on learning affordances for single objects or pairs of objects.

Tasks can usually be segmented into a series of skills, which the robot can then resequence using a high-level policy [20], [21], [22]. Unlike the learned preconditions, the high-level policy determines when a skill *should* be executed and not when it *can* be executed to achieve a particular manipulation. As a result, the policy is specific to a certain task while the preconditions can be reused between tasks.

Our approach to learning preconditions is based on random forest classifiers. Random forests have been widely used in computer vision applications, including image classification [23], human pose recognition [24], and image segmentation [25]. In the field of robotics, random forests have been used for tasks such as robot pose estimation [26], contact classification [27], and grasp classification [28]. Our approach uses random forests to classify configurations of assorted scene elements to learn the preconditions of manipulation skills.

II. Learning Preconditions of Manipulations

In Section II-A, we explain how to segment objects into parts and extract interactions between parts. The resulting scene elements are then represented by a set of labels and features, as described in Section II-B. Learning to classify configurations of scene elements using random forests is explained in Section II-C, and an illustrative example of the training process is described in Section II-D.

A. Extracting Object Parts and Interactions

The first step of the proposed approach is to decompose scenes into their task-relevant elements. Manipulations are usually associated with specific affordance-bearing parts of objects [29], [30], [31]. Hence, when generalizing between objects with different shapes, the positions of the parts are often more informative than the locations of the objects. Our framework explicitly models the affordance-bearing parts of objects, as well as the interactions between them.

We assume that the robot is provided with coarse 3D point cloud models of the objects as well as example scenes that demonstrate the objects interacting with other objects. These scenes are obtained from demonstrations of the task. We define a part as a subset of the points in the object model and, similarly, an interaction as a subset of a part's points.

We begin by extracting the interactions from the example scenes. Most interactions involve close proximity or direct physical contact between objects [32]. We therefore consider a point to be interacting if it is within a given threshold of another object's point cloud in the scene. We used a position threshold of 2 cm and required the inner product between the normals to be less than -0.75. In addition to comparing each pair of objects in the scene, we also tested for interactions between the objects and the table.

A part of an object is considered to be relevant if it is used to interact with other objects. We therefore use the extracted interaction points to initialize the part detection process. We use the GrabCut algorithm to segment the part from the rest of the object [33]. GrabCut iterates between modeling the local geometric features of the part and non-part regions of the object, and segmenting the point cloud based on these features. We compute the position, normal, curvature, and spectral features to represent the local geometry of each point in the object model [34], [35]. We generate a Markov random field for the segmentation by creating a 10-nearest neighbour graph. The unary potentials of the Markov random field are modeled using Gaussian mixture models with three Gaussians. The pairwise potentials are modeled using a Potts model [36]. Using the GrabCut approach, the robot detects parts of objects that have similar local geometries and afford the observed interactions.

Once the robot has extracted the affordance-bearing parts of an object, it can use them to represent novel scenes that contain the object. Parts are included in the set of scene elements even if the part is not currently interacting with another object. Thus, the parts can capture the object's potential for future interactions. For each part in a novel scene, we detect the set of points that are interacting with other parts in the scene. These points are then combined into one interaction scene element.

B. Representing Scene Elements

A scene model $\mathcal{S}_i$ consists of a set of n_i scene elements $\mathcal{S}_i = \{\varepsilon_{i1}, ..., \varepsilon_{in_i}\}$ which includes all of the objects, parts, and interactions in the scene. We define a set of features $\phi(\varepsilon_{ij}) = \phi_{ij} \in \mathbb{R}^d$ to represent the individual elements. In our experiments, all of the scene elements were represented

by the mean positions of their respective point clouds, resulting in a $d = 3$ dimensional feature space. The features thus model the spatial configuration of the elements in the scene. The set of features could be extended to include other properties of the scene elements, e.g., size and orientation.

To evaluate the preconditions, the robot tests if selected scene elements are present in automatically generated regions of the feature space, e.g., a test may evaluate if the apple is present in a certain volume above the plate. To generalize these tests between scenes with different objects, the robot needs to know the correspondences between the objects, e.g., the tests applied to the apple in the first scene should be applied to the corresponding pear in the second scene.

We predefine the object correspondences by labelling the elements in the scenes. The labels capture correspondences between individual objects and groups of objects, as well as indicate when a corresponding object is missing. We identify corresponding scene elements by defining m label sets $L_{ik} \subset S_i \forall k \in \{1, ...m\}$. A label set that contains at most one element from any scene defines a correspondence between individual objects, e.g., L_{12} may contain the large knife from the first scene S_1 and L_{22} contains the corresponding small knife from the second scene S_2. We assume that corresponding scene elements have the same types of affordances and can be used to perform similar manipulations. If there is no corresponding object in a scene, e.g., there is no knife, then the label set is empty.

By including multiple elements in a label set $|L_{ik}| > 1$, we can define correspondences between groups of objects. For example, we define one label set that includes all of the object scene elements. Using this label set, the robot can create precondition tests based on the number of objects in a region. This label set allows the robot to test for obstacles in a region without having to create a separate test for each object. In the future, we plan on using object recognition methods to automatically determine the corresponding elements between different scenes.

Each scene model S_i and corresponding label sets L_{ik} is assigned a binary precondition label $\mathcal{L}_i \in \{0, 1\}$ that indicates if the preconditions of the skill are fulfilled $\mathcal{L}_i = 1$ or not $\mathcal{L}_i = 0$. These precondition labels are known for the training data, and must be predicted for the test data.

C. Random Forest for Learning Spatial Preconditions

Learning the spatial preconditions can be framed as a classification problem, wherein the robot learns a mapping from the scene elements S_i and label sets L_{ij} to the binary precondition estimates $\mathcal{L}_i$. This mapping is not trivial to learn, due to the limited amount of training data and the variations in objects between scenes. We propose to use random forests to classify the configurations of scene elements. The random forests evaluate if the required scene elements are present in the scenario and if the scene elements are positioned in a configuration that affords the manipulation.

A random forest is an ensemble approach that consists of T random trees [3], [4]. Each tree starts as a single node that contains all of the training data. A node is considered a leaf node if it contains less than $a_{\min}$ samples, or if the precondition labels $\mathcal{L}_i$ of all of its samples are the same. A node is split by generating a set of κ random tests to evaluate the samples in the node. Each test divides the node's samples into two subsets: the samples for which the test returns true $S_i \forall i \in \mathbb{S}_T$, and the samples that return false $S_i \forall i \in \mathbb{S}_F$. The tests are evaluated using a score function, and the test with the highest score is used to split the node. We used the variance reduction score [3], [28] to evaluate the tests

$$\frac{-|\mathbb{S}_T|}{|\mathbb{S}_T| + |\mathbb{S}_F|} \mathrm{Var}(\mathcal{L}_i \forall i \in \mathbb{S}_T) - \frac{|\mathbb{S}_F|}{|\mathbb{S}_T| + |\mathbb{S}_F|} \mathrm{Var}(\mathcal{L}_i \forall i \in \mathbb{S}_F).$$

When a node is split, two child nodes are created. The samples for which the test was false $\mathbb{S}_F$ are passed to the first node, and the samples for which the test was true $\mathbb{S}_T$ are passed to the second node. This process continues until all of the samples are in leaf nodes. Each leaf is then assigned a prediction label corresponding to the mode of the precondition labels of its training samples.

The tree training process uses bagging to help avoid overfitting [4]. Bagging creates a separate training set for each random tree. Each tree's training set is sampled with replacement from the full training data set. In our experiments, the tree training sets include the same number of samples as the full training set. However, by sampling with replacement, the trees' training sets will usually include duplicate samples.

When classifying a new scene, the sample starts at the root node. It then moves along the edges of the tree depending on the outcomes of the tests associated with each node. When it reaches a leaf node, it is assigned the binary value corresponding to the leaf's prediction label. The sample is evaluated by each tree in the forest. The final prediction corresponds to the mode of the predictions of the trees.

We adapt the random trees to classify configurations of scene elements by modifying the tests used to split the nodes. The trees are designed to test for the presence of specific scene elements in automatically generated regions of the elements' feature space $\mathbb{R}^d$. A test consists of a label set index $l \in \{1, ..., m\}$, a test region $\mathcal{R} \subset \mathbb{R}^d$, and a threshold $\tau \in \mathbb{Z}^+$. The test is evaluated by computing the number of elements ε_{ij} in the selected label set L_{il} for which $\phi(\varepsilon_{ij}) \in \mathcal{R}$. The test is passed if the number of elements in the region is greater or equal to the threshold τ. The threshold τ is always one for label sets L_{il} that define correspondences between individual objects. The threshold may be greater for groups of corresponding objects, e.g., the robot may test if at least $\tau = 2$ objects are present in the region $\mathcal{R}$.

The tests, as described above, only capture the absolute locations of elements in the feature space. However, the relative positions of two elements are often more important. The robot can model relative locations by including reference elements ρ in the tests. For example, to evaluate the position of the knife relative to the fruit, the fruit would be the reference object. The reference element is given by a label set for an individual scene element, i.e., $\rho_i = L_{ij}$ where $|L_{ij}| \leq 1 \forall i$. If the reference element is present in the scene, then the robot evaluates the test by computing the number

| Training Set | | | | Test Set |

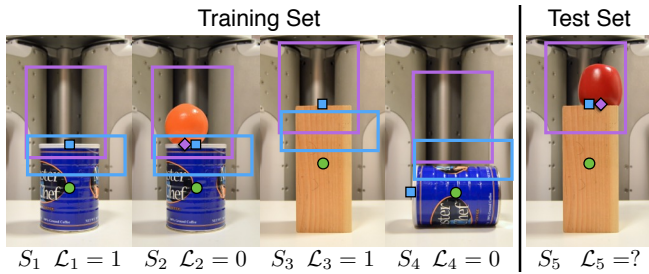| S_1 $\mathcal{L}_1 = 1$ | S_2 $\mathcal{L}_2 = 0$ | S_3 $\mathcal{L}_3 = 1$ | S_4 $\mathcal{L}_4 = 0$ | S_5 $\mathcal{L}_5 = ?$ |

Fig. 2. The figure shows example subscenes for the placing task. The grasped object for placing, as well as other objects in the scenes, are not shown. The supporting object L_{i1} is indicated by the green circle. The object's supporting surface L_{i2} is marked by the small blue square. The supporting surface's interaction with other objects L_{i3} is shown by the purple diamond. The label set L_{i3} is empty for scenes S_1, S_3, and S_4. Other elements in the scene are not marked for clarity. The regions $\mathcal{R}$ marked by the blue and purple squares correspond to two tests for the L_{i2} and L_{i3} label sets respectively, each with a threshold value of $\tau = 1$. Both of the regions are defined relative to the supporting object $\rho_i = L_{i1}$.

of elements $\varepsilon_{ij} \in L_{il}$ for which $\phi(\varepsilon_{ij}) - \phi(\rho) \in \mathcal{R}$ and compares this value to the threshold τ. If the reference element is not in the scene $\rho = \emptyset$, then the test returns false.

The random trees combine the tests of the individual nodes to evaluate the configuration of multiple elements in the scene. Deeper trees can encode more complicated preconditions with additional scene elements and test regions. However, by greedily selecting the test with the best score, the random trees are more likely to create simpler sets of tests to explain the data. Random forests reduce the chance of overfitting by merging the predictions from multiple random trees. Given a limited amount of training data, there may be multiple sets of preconditions that could explain the data. In the case of a random forest, each random tree can capture a separate set of preconditions and the forest will only label the scene as valid $\mathcal{L}_i = 1$ if the majority of the trees predict that the preconditions have been fulfilled.

D. Example of Training Process

In this section, we present an illustrative example of the tree training process. A set of training examples for a placing skill is shown in Fig. 2. The pictures focus on the support object and do not show the grasped object that should be placed nor other objects in the scene. The support objects L_{i1} are indicated by green circles, and the supporting surface parts L_{i2} are marked by blue squares. An interaction between the supporting surface and another object is marked by a purple diamond L_{i3}. The surface part is only interacting in scene S_2, i.e., $L_{i3} = \emptyset \forall i \in \{1, 3, 4\}$. The other scene elements, such as the orange obstacle in S_2 and its corresponding parts and interactions, are not marked. The robot can perform the placing skill in scenes S_1 and S_3, but not in the other two training scenes as the supporting surface is blocked in S_2 and on the side in S_4.

The robot has randomly generated two tests as part of the tree training process. The first test evaluates if there are $\tau = 1$ or more elements from L_{i2} in the region $\mathcal{R}$ indicated by the large blue square, i.e., is the supporting surface (blue icon) in

the blue region? Similarly, the second test evaluates if there are $\tau = 1$ or more elements from L_{i3} in the purple region $\mathcal{R}$, i.e., is there an interaction with the supporting surface (purple diamond) in the purple region? Both of these tests' regions are defined relative to the supporting object $\rho_i = L_{i1}$ (green circle). Hence, both tests would return false if the supporting object were not present in the scene. The location and sizes of the test regions $\mathcal{R}$ were sampled from uniform distributions. The threshold values τ did not need to be sampled as the test regions included at most one test element.

The first test returns true for scenes S_1 and S_2, and false for scenes S_3 and S_4. The variance reduction score for this test is therefore $-0.5\text{var}(\{1, 0\}) - 0.5\text{var}(\{1, 0\}) = -1$. The second test returns true for S_2 and false for the other three scenes. Hence, the variance reduction for test two is $-0.25\text{var}(\{0\}) - 0.75\text{var}(\{1, 1, 0\}) = -0.25$.

Since the score is greater for the second test $-0.25 > -1.0$, the robot selects the second test and divides the data such that $\mathbb{S}_T = \{2\}$ and $\mathbb{S}_T = \{1, 3, 4\}$ for this node. The child node with samples $S_i \forall i \in \mathbb{S}_T$ would become a leaf node as all of its samples have the same class $L_i = 0 \forall i \in \mathbb{S}_T$. The other child node, which contains the three samples $S_i \forall i \in \mathbb{S}_F$, would become a leaf node with output $L = 1$ if $a_{\min} \geq 3$. Otherwise, the impure node would be split again by generating a new set of tests. These tests could separate samples S_1 and S_3 from S_4 and create two final leaf nodes.

When presented with a test scene, the robot would evaluate only the selected test. For the example scene S_5, the robot would evaluate the test as false and the tree would correctly assign the precondition label $L_5 = 0$. The robot trains multiple trees in this manner and selects the precondition label that the majority of the trees predicted.

III. EXPERIMENTS

We performed four experiments to evaluate the proposed method. The first experiment evaluated the performance when using different types of scene elements as well as individual random trees versus random forests. The second experiment compared the performance of the random forest approach to a Gaussian naive Bayes classifier. The third and fourth experiments demonstrated how the learned preconditions can be used to select skill-relevant objects and goal states for the previous skill in a sequence.

A. Scene Data

To evaluate the proposed approach, the robot was provided with initial scenes for six different tasks: placing, pushing, tilting, pouring, cutting, and wiping. The tasks are shown in Fig. 3. For each task, the robot was given 30 scenes with 15 positive samples $\mathcal{L}_i = 1$, and 15 negative samples $\mathcal{L}_i = 0$. The scenes' preconditions $\mathcal{L}_i$ and the scene elements L_{ik} were manually labelled. Each scene contained between 4 and 7 objects. Depending on the task, only 2 to 4 objects were required for the manipulation.

The robot was also provided with three scenes for each task demonstrating the affordances of the objects. These scenes were used to extract the parts of the objects, as

Fig. 3. The pictures show example executions of the PR2 performing the six manipulation tasks: placing, pushing, tilting, pouring, cutting, and wiping.
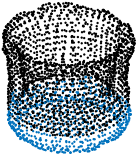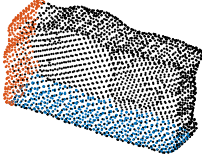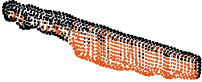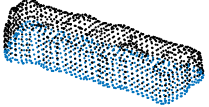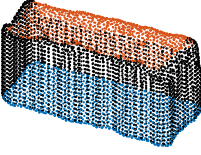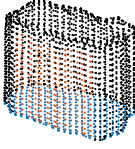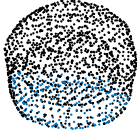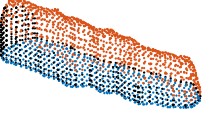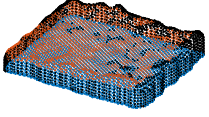
Place	Push	Tilt	Pour	Cut	Wipe
Plastic Cup	Marker	Cardboard Box	Bottle	Knife	Eraser
Wooden Block	Metal Can	Apple	Cup	Cucumber	Book

Fig. 4. The figure shows examples of the object parts (colored points) that were extracted using GrabCut. The grasped parts have been omitted for clarity.

explained in Section II-A. Examples of the extracted object parts are shown in Fig. 4.

The precondition learning experiments were performed using 15-fold cross validation, with each test set containing one positive and one negative sample. The classifiers were trained on 10 positive and 10 negative training samples, randomly sampled from the pool of 28 remaining samples. Each fold of the cross validation was repeated five times with different training samples. We report the mean accuracies across these repetitions.

When generating the tests for the random trees, the robot selected an object as a reference element ρ with 95% probability. The remaining tests were defined relative to the world coordinate frame. The six boundaries of the test regions were uniformly sampled from a $50\text{cm} \times 20\text{cm} \times 55\text{cm}$ region centered above the table in the robot's workspace. The test label set indices l were sampled uniformly $p(l = k) = m^{-1} \forall k \in \{1, ..., m\}$. The reference elements were limited to the object elements. The threshold values were sampled uniformly from the range of values observed in the node's training set. A potential test was resampled if it did not split the data samples, i.e., $|\mathbb{S}_T| = 0$ or $|\mathbb{S}_F| = 0$. A node was not split if it contained $a_{\min} = 3$ or fewer samples. Due to the large space of tests and the limited amount of training data, most of the leaf nodes had pure sample labels and, hence, many leaf nodes contained more than three samples.

B. Precondition Learning

The first experiment compared the precondition learning performance when using different types of scene elements.

We evaluated using objects, parts, or interactions individually, or all three together (OPI). For the OPI case, a third of the κ tests were assigned to each type of scene elements. In addition to the scene element types, we also compared the performance of a single random tree to a random forest of $T = 100$ trees to explore the effects of using an ensemble approach. The random forests' trees were trained by generating $\kappa = 30$ potential tests for splitting each node. The single random tree was trained with $\kappa = 1000$ potential tests, which resulted in better performance for the single tree. The single tree was trained without bagging.

The results of the experiments are shown in Fig. 5. Despite using relatively few training samples, the random forest approach performed well and achieved high accuracies. The highest overall accuracy of 87.2% was achieved by the OPI random forest approach. In contrast, the object random forests achieved an accuracy of 79.6%. Hence, including the parts and interactions into the set of scene elements increased the expected accuracy by 7.67%. The use of an ensemble approach also increased the classification performance. The accuracy improved on average by 7.17% overall, and 7.22% for the OPI case. When combined, using an ensemble approach and including the parts and interactions resulted in an increase in accuracy of 16.4%.

Of the three types of scene elements, the interactions were the most discriminative and achieved the highest accuracies in five of the six tasks. For pouring, which does not involve direct physical contact between objects, the parts were the most discriminative elements. Interactions give the most
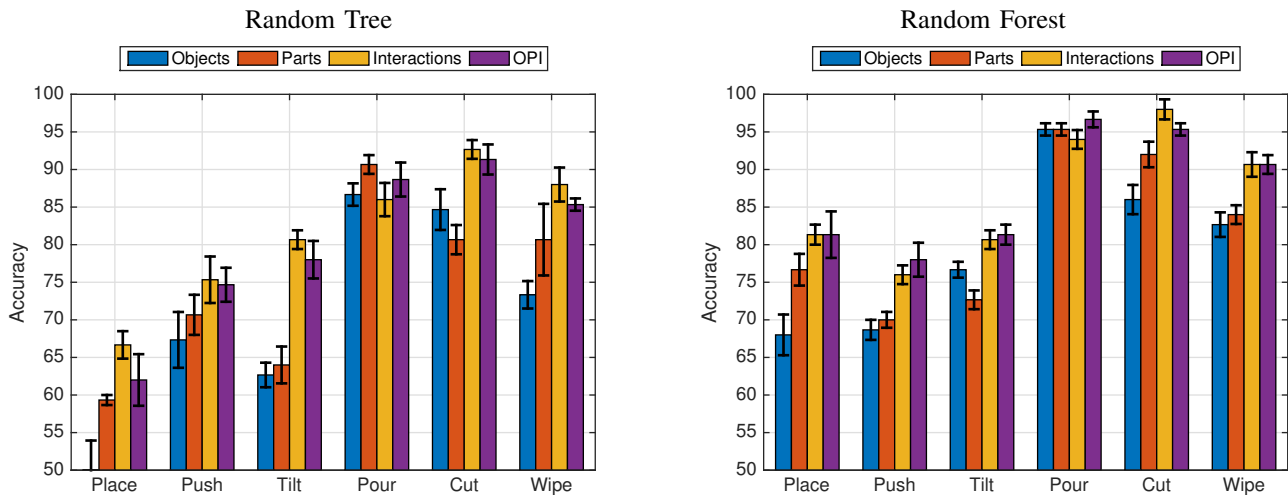
Fig. 5. The figure shows the results of the precondition learning. (Left) The accuracies when using a single random tree trained with $\kappa = 1000$ tests per split. (Right) The accuracies when using a random forest of 100 random trees trained with $\kappa = 30$ tests per split. Error bars indicate +/- one standard error.

detailed information of the three element types and they represent the locations of active object affordances in the scene. A random tree requires many small test regions to implicitly learn if two objects are touching using only the object and part elements. When the interactions are explicit, a single large test region can be used, which makes the learning easier. Unfortunately, interactions are the most difficult scene elements to detect. It is not trivial to determine if two objects are in contact based on only vision data. The results therefore emphasize the need for tactile methods to verify if contact interactions between objects have occurred [37], [38].

The results indicate that more specific preconditions (e.g., for pouring and cutting) are easier to learn. One reason for this result is that a random tree is more likely to generate a test region that covers all of the positive samples if the test elements are close together across the scenes. Another reason is that the movements of these skills are also smaller and less likely to be blocked. The placing, pushing, tilting, and wiping skills all included multiple scenes where the skill was blocked by another object. Given the limited amount of training data, the training set may only include a couple of examples of an object blocking the skill execution. The robot would need to use these samples to determine the region that needs to be empty, and that any object could be an obstacle and not just the ones in the training set. For these skills we observe large performance boosts when using interactions. Interactions are the result of any object being near the object part. Hence, they can also capture the presence of any obstacle object in the proximity of the part. Obstacles that are not in close proximity will not be captured by the interaction elements.

Overall, the experiments have shown that explicitly incorporating interaction and part elements leads to increased accuracy. Using an ensemble of random trees allows the classifier to capture multiple possible sets of preconditions. The ensembles therefore result in higher accuracies and more robust predictions.
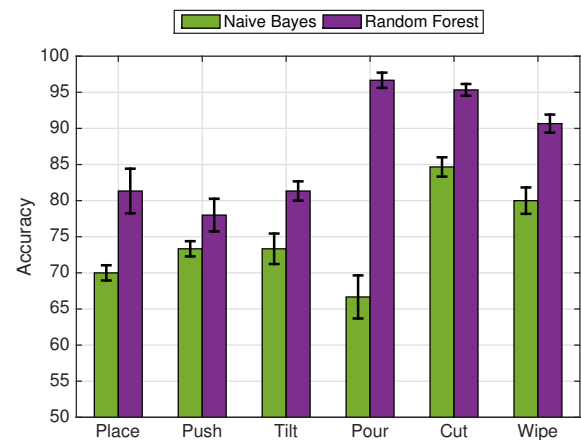

Fig. 6. Results for comparison of random forest to naive Bayes classifier.

C. Comparison to Naive Bayes Classifier

In the second experiment, we compared the performance of the OPI random forest to a Gaussian naive Bayes classifier (NBC). The NBC does not require many training samples, and it can handle missing features. The features correspond to the 3D positions of all of the OPI scene elements in the world coordinate frame, as well as relative to each of the objects in the scene. Including the object-relative features significantly increases the number of features, but omitting these features resulted in worse performance. Features with less than two training samples per class, or that were not defined for the test sample, were omitted for the purpose of classifying the sample.

The results of the experiment are shown in Fig. 6. The NBC achieves a mean accuracy of 74.7% across the different tasks, which is 12.6% less than the mean accuracy for the OPI forest. This result suggests that the random forest's ability to select subsets of scene elements, capture coarse correlations between features, and specify different regions for detecting elements results in better performance.
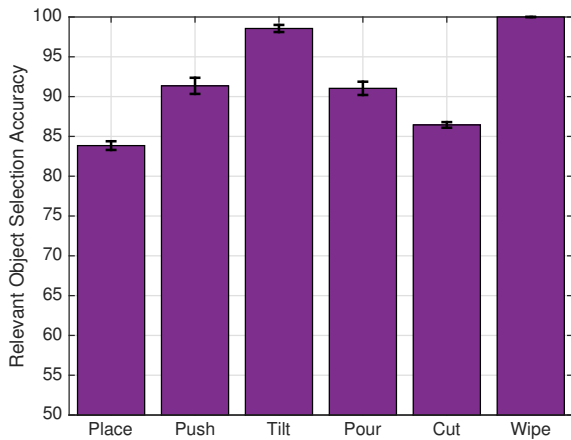
Fig. 7. The figure shows the results of selecting the set of relevant objects based on the learned preconditions. The relevant objects were selected based on individual test scenes. Errorbars indicate +/- one standard error.

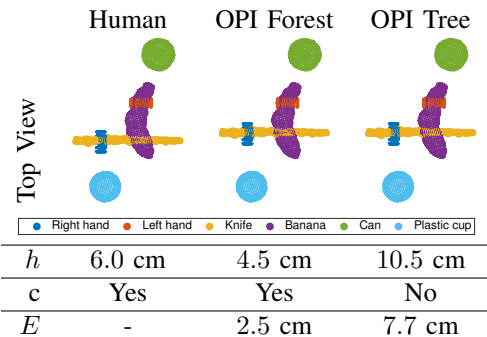|	Human	OPI Forest	OPI Tree
h	6.0 cm	4.5 cm	10.5 cm
c	Yes	Yes	No
E	-	2.5 cm	7.7 cm

Fig. 8. The figure shows the results of estimating the goal state for placing the knife using preconditions learned for the cutting skill. The columns show the results for different classifiers. The first column shows the goal state selected by a human. The top row shows a view of the scene with the highest score. The second row (h) indicates the height of the knife above the table plane. The third row (c) indicates if the knife and the banana are in contact in the scenes. The bottom row (E) indicates the average error relative to the human scene over all of the scenes with the highest score.

D. Relevant Object Selection

If an object is required to perform a manipulation skill, then removing the object from the scene should invalidate the skill's preconditions. For example, the preconditions of a cutting skill will no longer be fulfilled if the knife is removed from the scene as it is relevant to the task. By contrast, the predicted precondition labels should not be affected if an irrelevant object is removed, e.g., a rock. The robot can thus use the learned preconditions to estimate which objects are relevant for performing the skill.

In this experiment, the robot estimated the set of relevant objects, by re-evaluating the preconditions when individual objects were omitted from the model of the scene. This approach can only be applied to scenes that the robot estimated as valid $\mathcal{L}_i = 1$. If the re-evaluation results in an invalid condition $\tilde{\mathcal{L}}_i = 0$, then the omitted object is considered to be relevant to the scene.

Using the OPI random forests trained in the previous experiment, the robot was able to select the relevant objects with an accuracy of 91.9% based on individual test scenes. The accuracies for the individual skills are shown in Fig. 7. The set of test scenes included both true positive and false positive samples. The accuracy can be improved by taking the mode of the estimates from multiple scenes. The accuracy increases to 95.2% when the estimates from all of the test scenes were combined.

The majority of the errors correspond to relevant objects being excluded, rather than irrelevant objects being included. These errors are often the result of a lack of diversity in the training data. For example, the majority of the scenes for the cutting task included the right hand grasping the knife, even in the negative scenes $\mathcal{L}_i = 0$. As a result, the right hand element was not discriminative and, hence, it was often omitted from the set of relevant objects.

The experiment has shown that the robot can use the preconditions to accurately determine the set of relevant objects for the skill. A similar approach could potentially be used to identify obstacles in negative scenes $\mathcal{L}_i = 0$.

E. Preconditions for Estimating Goals

When performing a sequence of skills, the goal state of one skill must lie within the set of preconditions of the next skill [10], [1]. Preconditions can thus be used to determine goal states for the previous skill in the sequence. This experiment demonstrate how the robot can use the learned preconditions to select where to place the knife for the cutting task. We selected the cutting task as it has the most specific requirements. The locations of the other objects in the scene, including the grasp of the knife, are fixed.

To select a suitable location for the knife, the robot performed a grid search over a $30\text{cm} \times 50\text{cm} \times 16\text{cm}$ region at 2 cm increments. For each node in the grid, the robot extracted the set of scene elements and evaluated the preconditions using a classifier trained on all 30 scenes. Rather than using the binary output from the random forest, the robot assigned a score from 0 to 100 based on the number of random trees that consider the location to be valid.

We compared an OPI random forest and a single OPI tree classifier. The number of trees and tests are the same as the ones used in the first experiment. The results of the experiment are shown in Fig. 8. We included the location that a human would place the knife as the ground truth. The OPI forest found a single location with the best score. Due to its binary score, the OPI tree found 164 locations with the max score of 1. These locations span a $12\text{cm} \times 6\text{cm} \times 16\text{cm}$ region and include the location selected by the random forest. The figure shows the location that is the closest to the mean of all of the locations.

The OPI forest was closer to the manually selected goal with an offset of only 2.5 cm. Both classifiers resulted in a horizontal shift that would result in a larger portion of the fruit being cut off. Unlike the ensemble approach, the majority of the locations selected by the OPI tree did not create contact between the knife and the fruit. These locations therefore do not fulfill the preconditions for the cutting skill. This result shows the benefit of using an ensemble approach to capture multiple sets of possible preconditions and ranking the potential goal states accordingly.

IV. Conclusion

We proposed an approach to learning the preconditions of skills using random forests. The robot first decomposes scenes into objects, parts, and interactions between parts. The parts are segmented using the GrabCut algorithm and the interactions are detected by the proximity of the objects. The configurations of the scene elements are classified by a random forest to determine if the preconditions have been fulfilled. Each random tree creates a set of spatial tests that determine if certain scene elements are present in absolute or relative regions of the elements' feature space. In this manner, the trees can detect if a required object is not present in the scene or if it is in the wrong position.

The proposed approach was evaluated on the preconditions of six manipulation skills. The results of the experiments show that using an ensemble of random trees, rather than one random tree, and including the parts and interactions in the scene models increases the classification accuracy by 16.4%. The experiments also demonstrated how the robot can use the learned preconditions to accurately select the set of task-relevant objects and the goal state for the previous skill in a sequence of skills.

References

[1] G. D. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[2] J. J. Gibson, *The Ecological Approach To Visual Perception*. Lawrence Erlbaum Associates, 1986.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth International Group, 1984.

[4] L. Breiman, "Random forests," *Machine Learning*, 2001.

[5] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *International Journal of Robotics Research*, 2011.

[6] S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Kruger, and F. Guerin, "Learning spatial relationships from 3d vision using histograms," in *International Conference on Robotics and Automation (ICRA)*, 2014.

[7] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *International Conference on Robotics and Automation (ICRA)*, 2015.

[8] J. Kulick, T. Lang, M. Toussaint, and M. Lopes, "Active Learning for Teaching a Robot Grounded Relational Symbols," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2013.

[9] K. Zampogiannis, Y. Yang, C. Fermuller, and Y. Aloimonos, "Learning the spatial semantics of manipulation actions through preposition grounding," in *International Conference on Robotics and Automation (ICRA)*, 2015.

[10] G. D. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *International Journal of Robotics Research (IJRR)*, vol. 31, no. 3, March 2012.

[11] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krueger, and J. Piater, "Learning grasp affordance densities," *Paladyn. Journal of Behavioral Robotics*, vol. 2, no. 1, pp. 1–17, 2011.

[12] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *International Journal of Robotics Research (IJRR)*, 2008.

[13] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor coordination to imitation," *Transactions on Robotics (TRo)*, vol. 24, no. 1, pp. 15–26, 2008.

[14] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick, "Learning contact locations for pushing and orienting unknown objects," in *International Conference on Humanoid Robots (Humanoids)*, 2013.

[15] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *International Conference on Development and Learning and on Epigenetic Robotics (ICDL EPI-ROB)*. IEEE, 10 2014, pp. 476–481.

[16] O. Kroemer and J. Peters, "Predicting object interactions from contact distributions," in *International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[17] A. Gonçalves, G. Saponaro, L. Jamone, and A. Bernardino, "Learning Visual Affordances of Objects and Tools through Autonomous Robot Exploration," in *International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2014.

[18] S. Griffith and A. Stoytchev, "Interactive categorization of containers and non-containers by unifying categorizations derived from multiple exploratory behaviors." in *International Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2010.

[19] C. Eppner and O. Brock, "Planning grasp strategies that exploit environmental constraints," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 4947–4952.

[20] J. Butterfield, S. Osentoski, G. Jay, and O. Jenkins, "Learning from demonstration using a multi-valued function regressor for time-series data," in *International Conference on Humanoid Robots (Humanoids)*, Dec 2010, pp. 328–333.

[21] S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto, "Incremental semantically grounded learning from demonstration," in *Robotics: Science and Systems (R:SS)*, 2013.

[22] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *International Conference on Robotics and Automation (ICRA)*, 2015.

[23] B. Yao, A. Khosla, and L. Fei-Fei, "Combining randomization and discrimination for fine-grained image categorization," in *International Conference on Computer Vision and Pattern Recognition*, 2011.

[24] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *International Conference on Computer Vision and Pattern Recognition(CVPR)*. IEEE, June 2011.

[25] F. Schroff, A. Criminisi, and A. Zisserman, "Object class segmentation using random forests," in *British Machine Vision Conference*, 2008.

[26] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," in *International Conference on Robotics and Automation (ICRA)*, 2014.

[27] S. Cabras, M. E. Castellanos, and E. Staffetti, "A random forest application to contact-state classification for robot programming by human demonstration," *Applied Stochastic Models in Business and Industry*, 2015.

[28] S. Leischnig, S. Luettgen, O. Kroemer, and J. Peters, "A comparison of contact distribution representations for learning to predict object interactions," in *International Conference on Humanoid Robots*, 2015.

[29] J. Sung, S. H. Jin, and A. Saxena, "Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d point-clouds," in *International Symposium on Robotics Research*, 2015.

[30] S. R. Lakani, M. Popa, A. J. Rodríguez-Sánchez, and J. H. Piater, "CPS: 3d compositional part segmentation through grasping," in *Conference on Computer and Robot Vision (CRV)*, 2015.

[31] M. Tenorth and M. Beetz, "Representations for robot knowledge in the knowrob framework," *Artificial Intelligence*, 2015.

[32] F. Worgotter, E. E. Aksoy, N. Kruger, J. Piater, A. Ude, and M. Tamosiunaite, "A simple ontology of manipulation actions based on hand-object relations," *Transactions on Autonomous Mental Development (TAMD)*, vol. 5, no. 2, pp. 117–134, 2013.

[33] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut -interactive foreground extraction using iterated graph cuts," *International Conference and Exhibition on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2004.

[34] A. Boularias, O. Kroemer, and J. Peters, "Learning robot grasping from 3d images with markov random fields," in *International Conference on Intelligent Robot Systems (IROS)*, 2011.

[35] D. Munoz, N. Vandapel, and M. Hebert, "Onboard contextual classification of 3-d point clouds with learned high-order markov random fields," in *International Conference on Robotics and Automation*, 2009.

[36] R. B. Potts, "Some generalized order-disorder transformations," *Mathematical Proceedings of the Cambridge Philosophical Society*, 1952.

[37] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011.

[38] Z. Su, O. Kroemer, G. Loeb, G. Sukhatme, and S. Schaal, "Learning to switch between sensorimotor primitives using multimodal haptic signals," in *International Conference on Simulation of Adaptive Behavior (SAB)*, 2016.