Learning Symbolic Failure Detection for Grasping and Mobile Manipulation Tasks

Patrick Hegemann, Tim Zechmeister, Markus Grotz, Kevin Hitzler and Tamim Asfour

***Abstract*— The ability to detect failure during task execution and to recover from failure is vital for autonomous robots performing tasks in previously unknown environments. In this paper, we present an approach for failure detection during the execution of grasping and mobile manipulation tasks by a humanoid robot. The approach combines multi-modal sensory information consisting of proprioceptive, force and visual information to learn task models from multiple successful task executions, in order to detect failures and to externalize them for humans in an interpretable way. To this end, we define symbolic action predicates based on multi-modal sensory information to allow high-level state estimation based on action-specific decision trees. To allow symbolic failure detection, we then learn task models that are represented as Markov chains. We evaluated the approach in several pick-and-place and mobile manipulation tasks performed by a humanoid robot in a decommissioning and a household scenario. The evaluation shows that the learned task models are capable of detecting failure with an F1-score of 93%.**

I. Introduction

Mobile manipulation of unknown objects is an essential and difficult robot task that requires the integration of object detection, grasp hypothesis extraction and selection, as well as the sensory based execution. During such complex tasks, failures can occur at any time due to inaccuracies in perception and execution. To increase efficiency and autonomy, robots must be able to detect these failures based on sensory feedback and recover from them to successfully complete the task. In addition, the externalization of these failures to a human operator contributes significantly to the explainability of the robot's behavior. Thus it is important to develop methods that go beyond anomaly detection based on sensory data and extend to failure detection on a symbolic level. Given a task model with different actions, as well as predicates and states that describe them, the robot should not only be able to detect failures, but also recognize which actions failed in which state.

Figure 1 shows the humanoid robot ARMAR-6 grasping unknown objects in a cluttered scene, and provides an overview of how failures are detected on a symbolic level during execution. The task consists of grasping these objects located in one box and navigating to a second box, in which the objects should be placed. The ability to recover from failures allows the robot to continue the execution in case of failure and hence drastically reduces the overall time needed to perform the task. By introducing a failure detection we can detect *if* and *when* the task execution should be stopped and restarted to achieve the goal. Furthermore, different types of failure can occur during task execution. For example, the robot might fail to grasp an object due to an inaccurate grasp hypothesis, or a grasped object might slip out of the robot's hand while lifting, holding or carrying the object. Thus, relying on a single sensory modality is not sufficient to detect all types of failure and to distinguish between them.

In this paper, we present an approach for failure detection based on multi-modal sensory information combining proprioceptive, force and visual input to detect failures in the context of mobile manipulation tasks. Our main contributions are (1) a framework for learning symbolic task models from multi-modal data and (2) a method for detecting failures based on the learned models in an interpretable way.

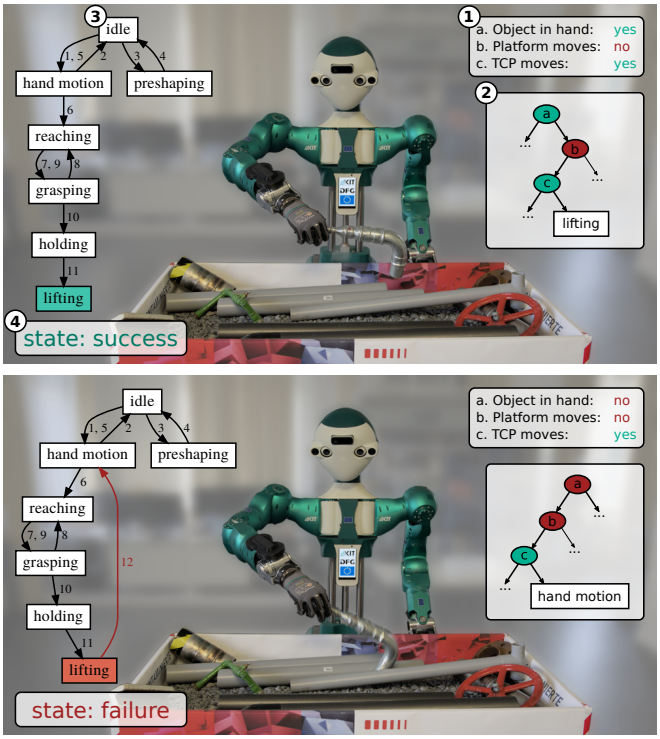


Fig. 1. The humanoid robot ARMAR-6 grasping an object from a cluttered scene in a nuclear decommissioning scenario with unknown objects of varying shape and weight. Failures such as an object slipping out of the robot's hand while lifting it are detected. Symbolic action predicates are extracted ① from multi-modal sensory data, to estimate the high-level state of the robot using a decision tree ②. Based on the transition between states, a task graph is constructed ③ and classified into success or failure ④ according to a task model learned from previous executions.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project ROBDEKON (13N14678).

The authors are with the High Performance Humanoid Technologies Lab, Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany. {patrick.hegemann, asfour}@kit.edu

To this end, we define symbolic action predicates from multi-modal sensory data (Section III-A) and build a decision tree for the underlying actions to determine the current state of the robot (Section III-B). Based on the detected states and transitions we learn a task model from multiple executions that serves as a reference to detect success and failure of the task execution (Section III-C). We evaluate the approach by conducting several pick-and-place and mobile manipulation experiments with a humanoid robot (Section IV). The results show that our system is able to detect failures with an F1-score of 93% during online execution and visualize the decision making process to a human operator.

II. Related Work

In the context of failure detection for complex manipulation tasks, we distinguish between model-based and model-free approaches. Model-based methods usually estimate the robot state from observations, create a reference model for task execution and then detect failures based on the model [1]–[10]. On the other hand, model-free methods learn to classify success and failure cases directly from observations [11]–[15]. This is usually done by training various deep learning architectures in an end-to-end manner, such as convolutional neural networks (CNN) [14], [15], or long short term memories (LSTM) [13], [15]. In the same way, classical machine learning models such as k-Nearest-Neighbours (kNN) and Support Vector Machines (SVM) can be trained for failure detection [11]. Classical machine learning approaches can also be combined with deep learning approaches [12] to improve feature extraction for failure detection. While these approaches can achieve very high performance in anomaly detection and failure classification, the underlying decision making process can not easily be externalized and made explainable for a human.

Model-based approaches can improve the explainability of the decisions in several ways. In [1], the state of the robot is estimated on a symbolic level. The hidden states of a Hidden Markov Model (HMM) are learned using a clustering procedure based on Kohonen networks, and labeled with high-level descriptions [16]. Inceoglu *et al.* [6] define a set of symbolic rule-based predicates that can be extracted from proprioceptive, auditory and visual input. The predicates are then used as observations for training several HMMs to detect failures. Similar to [6], we define a set of action predicates that describe manipulation actions based on multi-modal sensory data. For state estimation on a symbolic level, decision trees based on object relations have been used in the context of labeling human demonstrations [17] and skill transfer to humanoid robots [18]. Rojas *et al.* [19] propose a taxonomy based on relative change in force signatures to detect low-level and high-level behaviors in the context of an assembly task, which can be used to detect failures [3]. In our work, we define action-specific decision trees based on the proposed multi-modal action predicates to determine the current high-level state of the robot. Thus, the state estimation process becomes more transparent and explainable to humans.

Another way to improve explainability is by sequencing separate models of each different skill or action within a complex manipulation task [2], [4], [7], [9], [10], [20], [21]. This allows anomaly detection on the level of transitions between skills. Kappler *et al.* [4] use a manually constructed manipulation graph that models both normal and anomalous behavior, where transitions are detected using an associative skill memory [22]. HMMs with different emission processes for state estimation based on multi-modal sensory data are commonly used to model individual skills [2], [5], [7]–[10], [20], [21]. To allow failure detection for complex manipulation tasks with various sub-goals achieved by different actions, we learn a model for each action by automatically segmenting the observations according to the state of task execution. Since we assume the states to be fully observable in our approach, it is sufficient to represent these models as Markov Chains.

The learned task models can then be used to detect anomalies and failure cases. Even though anomaly detection and failure detection are not generally the same problem, we assume that a deviation from the model of successful behavior means that a failure has occurred. Most methods using HMMs detect failures based on applying a threshold to the (log-)likelihood of the estimated hidden state space trajectory [1], [2], [5], [7]–[10]. Since we are using Markov chains in our approach, we can calculate a likelihood and detect failures in the same way. In [1], the minimum and maximum duration of hidden states is considered to calculate error scores for each state. A different approach proposed in [6] defines an HMM with two hidden states for success and failure respectively. Failures are then detected based on the estimated posterior probability of these hidden states. When using multiple HMMs for different actions within a task, the model with the highest likelihood is selected for failure detection [2], [5], [7]–[10]. In this paper, we assume that the currently executed action is known and given by the task execution plan. Thus, we only evaluate the corresponding model for the current action to detect failures.

Additionally, the approaches presented in [4] and [10] show how task models can be extended to include recovery behavior. However, this is out of scope of this work, and we implement a recovery strategy which restarts the task from the beginning.

III. Learning Symbolic Failure Detection

We propose a failure detection system for grasping and mobile manipulation tasks that makes use of multi-modal sensory information to detect and externalize failures. Figure 2 shows an overview of the system. By incorporating prior knowledge into the high-level state estimation process, a task model is learned on a symbolic level, and the decision making process can be visualized to the user. To this end, we first ground sensor values into symbolic action predicates, also taking into account the current execution context, such as the target grasping pose. In the next step, we define a decision tree to determine the robot's current state based on the detected predicates. Since complex manipulation tasks

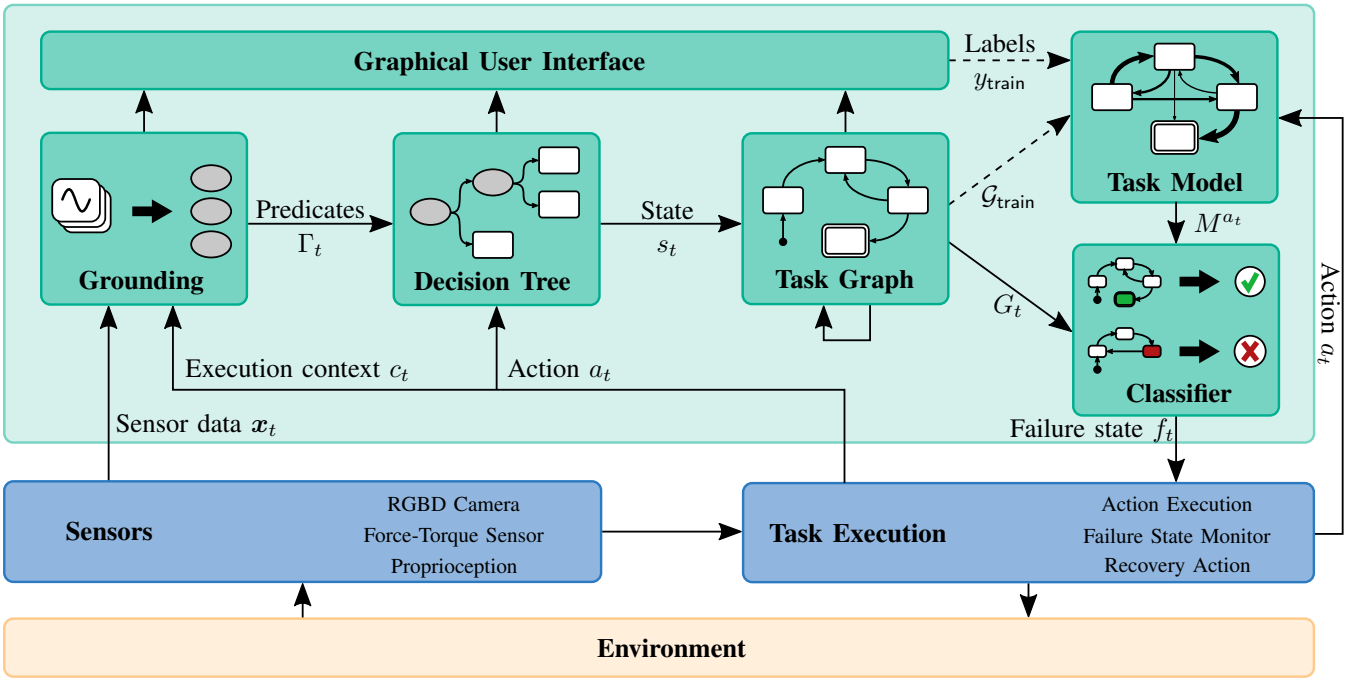


Fig. 2. Overview of the system architecture for failure detection, providing a high-level feedback loop for a task execution plan. In the first step, multi-modal sensory data is grounded into symbolic predicates. The current state of the robot is determined using an action-specific decision tree based on the detected action predicates. A task graph is constructed from the resulting series of states. Based on the experience from previous executions, a task model is learned and then used to detect failures during task execution. Dashed arrows mark the flow of training data and are only relevant for learning the task model.

consist of multiple actions, such as *pick*, *place*, etc., we define a separate decision tree for each action. During task execution, we continuously determine the current state, and construct a task graph from the states and the transitions between them. Next, several task graphs of successful executions are used to learn a task model, which is then used to detect failures during task execution.

Formally, we define multi-modal sensory data as a series of observations $\boldsymbol{x}_{1:t} \in \mathcal{X}^t$, where $\mathcal{X}^t$ is the observation space over t time steps. We denote the probability of a series of observations originating from a successful execution with $p(\boldsymbol{x}_{1:t})$. In our approach, we further take into account the execution context $c_{1:t}$ and the executed action $a_{1:t}$ at each time step. The execution context c_t is a set of variables containing high-level information, such as the tool center point (TCP) target pose for the current grasp. a_t denotes the current high-level action as defined by the task execution plan, such as *pick* or *place*. Therefore, we define f_t, indicating the failure state at time t, as

$$f_t = f(\boldsymbol{x}_{1:t}, c_{1:t}, a_{1:t}) = \begin{cases} 1, & \text{if } p(\boldsymbol{x}_{1:t}|c_{1:t}, a_{1:t}) \leq \tau \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where τ is the threshold for a successful execution. The main challenge now is to determine the probability p, using a learned symbolic task model. Therefore, we must first estimate the high-level symbolic state of the robot. To this end, we define a set of action predicates Γ, and a rule-based grounding function $g(\boldsymbol{x}_t, c_t)$ based on the current observation $\boldsymbol{x}_t$ and execution context c_t. From this data, we extract features z^γ for predicates $\gamma \in \Gamma$. By applying a threshold τ_γ, the detected predicates for the current time step $\Gamma_t \subset \Gamma$ are given as

$$\Gamma_t = g(\boldsymbol{x}_t, c_t) = \{\gamma \in \Gamma \mid z_t^\gamma \geq \tau_\gamma\}. \quad (2)$$

The predicates, features and grounding function are described further in Section III-A. Next, we define a decision tree T^a for each action a, in order to determine the current state $s_t \in \mathcal{S}$ based on the detected predicates Γ_t with

$$s_t = T^{a_t}(\Gamma_t). \quad (3)$$

The definition of actions, states and decision trees is described in Section III-B. This yields a series of high-level symbolic states $s_{1:t}$. We can model this series as a task graph G_t, which is the directed graph

$$G_t = (\mathcal{S}, \{(s_i, s_{i+1}) \mid i \in [1, t)\}), \quad (4)$$

defined by the states $\mathcal{S}$ and transitions in $s_{1:t}$. This task graph can be visualized in order to externalize the current execution progress and failure state to a human operator. Further, we use a set of task graphs $\mathcal{G}_{\text{train}}$ recorded from executions of the entire task to learn task models M^a, represented as Markov Chains, for each action a. To this end, each task graph $G \in \mathcal{G}_{\text{train}}$ is labeled by a human operator such that $y_{\text{train}}(G) = 1$ if G models a failed execution, and 0 otherwise. The appropriate sub-graphs used for training each task model are determined by $a_{1:t}$. Finally, during the task execution, the appropriate symbolic task models are used to

estimate the failure probability

$$p(\boldsymbol{x}_{1:t}|c_{1:t}, a_{1:t}) \approx p(s_{1:t}|M^{a_{1:t}}), \tag{5}$$

and the resulting failure state is determined using (1). We describe the learning of task models and failure detection further in Section III-C.

A. Predicate Definition

General action predicates are defined based on features extracted from multi-modal sensory input to allow failure detection on a symbolic level. Table I provides a list with the defined action predicates. The used input modalities include motor encoders, wrist-mounted 6D force-torque sensors, and mobile platform pose determined based on laser scanner data. Following (2), we denote a predicate $\gamma \in \Gamma$ as *detected* at time step t, if the corresponding feature $z_t^\gamma(\boldsymbol{x}_t, c_t)$ exceeds a threshold τ_γ , and thereby define $g(\boldsymbol{x}_t, c_t)$. In the first step of feature extraction, a median filter with fixed window size is applied to the measured sensor values. If not denoted otherwise, all variables refer to the measurement at the current time step t after filtering. Some of the defined features depend on the change in a certain measurement, e.g. to detect end-effector movement relative to the target pose. For scalar quantities $\theta \in \mathbb{R}$, we define the difference of that measurement between two time steps as $\Delta\theta = \theta_t - \theta_{t-1}$. Similarly, for vectors $\boldsymbol{v} \in \mathbb{R}^n$, we define $\Delta\boldsymbol{v} = \|\boldsymbol{v}_t - \boldsymbol{v}_{t-1}\|$. For rotation matrices $\boldsymbol{R} \in \mathrm{SO}^3$, we calculate the axis-angle representation of $\boldsymbol{R}_t\boldsymbol{R}_{t-1}^T$, and set $\Delta\boldsymbol{R}$ to the resulting angle. In this work, we define the following predicates and their grounding rules:

- `Object_in_hand`: The norm of the force vector $\boldsymbol{f} \in \mathbb{R}^3$ measured by the force-torque sensor in the wrist exceeds a threshold with respect to a reference force $\boldsymbol{f}_{\mathsf{ref}} \in \mathbb{R}^3$. The reference force is recorded shortly before grasping and is captured in the current execution context c_t.
- `TCP_moves`: The absolute change in TCP position $\boldsymbol{x}_{\mathsf{TCP}} \in \mathbb{R}^3$ or TCP orientation $\boldsymbol{R}_{\mathsf{TCP}} \in \mathrm{SO}^3$ exceeds a threshold.
- `TCP_near_target`: The distance between the TCP position $\boldsymbol{x}_{\mathsf{TCP}}$ and the target position $\boldsymbol{x}_{\mathsf{target}} \in \mathbb{R}^3$ exceeds a threshold. Here, $\boldsymbol{x}_{\mathsf{target}}$ is the grasp pose captured in the current execution context c_t.
- `TCP_approaches` and `TCP_retreats`: The change in distance of the TCP position to the target position exceeds a threshold. If the TCP is neither approaching nor retreating from the target, the relative TCP direction is indeterminate.
- `Hand_opens` and `Hand_closes`: The change in joint angle of the fingers $\theta_{\mathsf{fingers}} \in [0, 1]$ or thumb $\theta_{\mathsf{thumb}} \in [0, 1]$ exceed a threshold. In ambiguous cases, closing takes precedence over opening.
- `Platform_moves`: The change in platform position $\boldsymbol{x}_{\mathsf{platform}} \in \mathbb{R}^2$ or orientation $\boldsymbol{\omega}_{\mathsf{platform}} \in (-\pi, \pi]$ exceeds a threshold.

With the exception of `platform_moves`, all predicates are defined for both arms of the humanoid robot ARMAR-6 and can be grounded independently from each other. Thresholds can be determined separately for both arms, in order to deal with slight differences in hardware.

TABLE I
PREDICATE DEFINITION

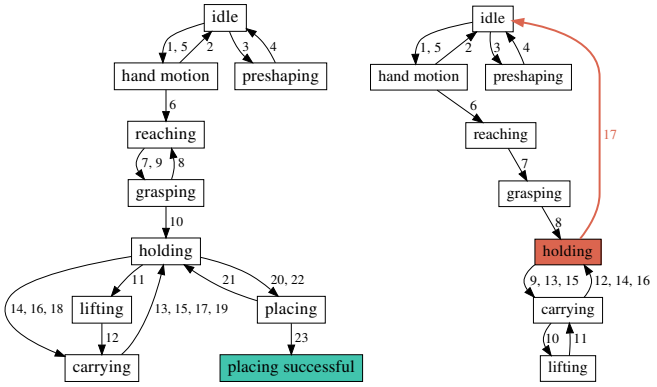
Feature	Condition	Predicate
$\|\boldsymbol{f}\|$ - $\|\boldsymbol{f}_{\mathsf{ref}}\|$	$\geq \tau_{\mathsf{force}}$	`Object_in_hand`
$\|\Delta\boldsymbol{x}_{\mathsf{TCP}}\|$ $\|\Delta\boldsymbol{R}_{\mathsf{TCP}}\|$	$\geq \tau_{\mathsf{tcp\_move}}$ $\geq \tau_{\mathsf{tcp\_rotate}}$	`TCP_moves`
$\|\boldsymbol{x}_{\mathsf{TCP}} - \boldsymbol{x}_{\mathsf{target}}\|$	$\leq \tau_{\mathsf{near}}$	`TCP_near_target`
$\Delta(\|\boldsymbol{x}_{\mathsf{TCP}} - \boldsymbol{x}_{\mathsf{target}}\|)$	$\leq \tau_{\mathsf{approach}}$ $\geq \tau_{\mathsf{retreat}}$	`TCP_approaches` `TCP_retreats`
$\Delta\theta_{\mathsf{fingers}}$ or $\Delta\theta_{\mathsf{thumb}}$	$\leq \tau_{\mathsf{open}}$ $\geq \tau_{\mathsf{close}}$	`Hand_opens` `Hand_closes`
$\|\Delta\boldsymbol{x}_{\mathsf{platform}}\|$ $\|\Delta\boldsymbol{\omega}_{\mathsf{platform}}\|$	$\geq \tau_{\mathsf{move}}$ $\geq \tau_{\mathsf{rotate}}$	`Platform_moves`
$\|V_{\mathsf{diff}}\|$	$\geq \tau_{\mathsf{voxels}}$	`Object_placed`

Furthermore, we define the vision-based predicate `Object_placed`, which is evaluated after the execution of a pick-and-place task. To this end, a point cloud of the scene is captured before and after placing the object. The former point cloud is denoted as P_{before}, the latter as P_{after} respectively. The captured point clouds are cropped to only include the relevant part of the scene, e. g., points above a given threshold. Then, the iterative closest point algorithm (ICP) is applied to cancel slight inaccuracies in robot placement and head positioning. P_{before} and P_{after} are then voxelized, resulting in the respective point clouds V_{before} and V_{after}. The voxel size can be adjusted depending on the geometry of the used objects.

Finally, the difference between V_{before} and V_{after} is computed by recursive comparison, resulting in the spatial change of point clouds, denoted as V_{diff}. If the total number of voxels in V_{diff} exceeds a threshold, i. e., the scene has changed significantly, then the predicate `Object_placed` is detected. Thus, the exact placement pose of the object within the scene is not considered, but only if it is placed within a certain area.

B. States and Decision Tree Definition

During the different phases of the *pick-and-place* task, the robot has various goals that can be achieved by different actions. This means that the states and transitions indicating a successful or failed execution vary between different actions. Therefore, we define a set of actions $A = \{a_{\mathsf{pick}}, a_{\mathsf{carry}}, a_{\mathsf{place}}\}$ and thus distinguish between the three actions *pick*, *carry*, and *place*. For each action, an individual task model will be learned. Furthermore, we define the possible states $\mathcal{S}^a$ for each action $a \in A$, and construct a decision tree for determining the robot state based on the detected predicates. Formally, each decision tree T^a maps the current set of predicates $\Gamma_t \subset \Gamma$ to the current state $s_t \in \mathcal{S}^a$. The functions T^{a_t} in (4) are then defined by the respective decision trees for actions a_t.



(a) a single successful execution (b) a single failed execution

Fig. 3. Examples for recorded task graphs of the pick-and-place task. Starting from the *idle* state, edges are labeled following the chronological order of detected transitions, ignoring self-transitions. The final state of a successful execution is marked in green. A failed transition and the respective state where the failure occurred are marked in red.

Due to the complexity of the *pick* action, we use all proprioceptive predicates described in Section III-A to define the relevant states and the decision tree. On the top level of this decision tree, a decision is taken for whether an object is in the hand or not. In the case that an object is in the hand, the three states *holding*, *lifting* and *carrying* can be detected, depending on whether the platform and the TCP are moving. Otherwise, i. e., an object is not in the hand, the decision tree distinguishes between the states *platform moves* and other manipulation action related states such as *reaching*, *grasping*, etc.

For the action *carry*, we use a reduced version of this decision tree that contains only the states *idle*, *holding*, *carrying* and *platform moves*. This means, that all TCP movement and finger motion is ignored, since the only relevant factors are whether an object is being held in the hand and whether the robot is moving.

For the action *place*, it is sufficient to detect that the object was released and to check whether it was successfully placed at the target location. Thus, we define a decision tree that depends only on the predicates `Object_in_hand` and `Object_placed`. Depending on whether an object is still in the hand, the active state is either *holding* or *placing*. The visual predicate `Object_placed` will determine the final state of the task graph as either *placing successful* or *placing failed*.

C. Task Modeling and Failure Detection

To decide whether the execution of a task was successful or failed, we first learn a task model. To this end, the robot state is continuously monitored during the execution of a task based on the predicates and decision trees as defined in Section III-A and Section III-B. The states and the transitions between them are used to construct a task graph. It is worth mentioning that a transition between two states can occur multiple times, and thus we label each transition with a list of the respective time steps, as shown in Figure 3. Using the recorded task graphs from several successful task executions as a training set, we learn a task model which is represented as a set of Markov Chains $M = \{M^a \mid a \in A\}$, where each Markov Chain M^a corresponds to one action a as defined in Section III-B. Accordingly, we define $\mathcal{S}^a$ as the set of states that can occur during the execution of action a. We assume that it is known which action a_t is executed at any time t. Thus, the task execution module automatically selects the appropriate task model based on the currently executed action. The recorded task graphs $\mathcal{G}_{\text{train}}$ and state sequences $\mathcal{S}_{\text{train}}$ for training are segmented accordingly into separate training sets $\mathcal{S}^a_{\text{train}}$. Furthermore, we define $n^a_{ij}(\mathcal{G}_{\text{train}})$ as the number of times that a transition from state i to state j has occurred within $\mathcal{S}^a_{\text{train}}$. The models M^a are then given as

$$M^a = (\mathcal{S}^a, \Pi^a), \quad \Pi^a = \begin{pmatrix} \pi^a_{11} & \dots & \pi^a_{1m} \\ \vdots & \ddots & \vdots \\ \pi^a_{m1} & \dots & \pi^a_{mm} \end{pmatrix} \tag{6}$$

$$\pi^a_{ij} = \frac{n^a_{ij}(\mathcal{G}_{\text{train}})}{\sum_{k \in \mathcal{S}^a} n^a_{ik}(\mathcal{G}_{\text{train}})}. \tag{7}$$

The transition matrices thus give the probability of each transition between two states, assuming a successful execution. Finally, the learned task model can be used to detect failures online during task execution. Following (5), the current execution is evaluated by calculating the likelihood of the current state trajectory $s_{1:t}$ given the model M^a as

$$p(s_{1:t}|M^a) = \prod_{t' \in [1,t)} \pi^a_{s_{t'} s_{t'+1}}. \tag{8}$$

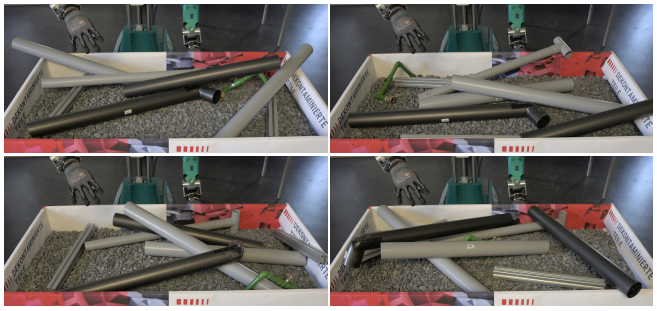
Due to the Markov property, we can further simplify this to $p(s_{1:t}|M^a) = \pi^a_{s_{t-1} s_t}$, i. e., transitions only depend on the current state, not the entire history of states. In the case that the action is not changed between two time steps, i. e., $a_t = a_{t-1}$, the failure state is determined using (1). However, failures can also occur when switching between actions. In this case, the failure state is determined by whether the current state is well-defined in the models for both actions, and we set

$$f_t = \begin{cases} 0, & \text{if } s_{t-1} = s_t \in \mathcal{S}^{a_{t-1}} \cap \mathcal{S}^{a_t} \\ 1, & \text{otherwise.} \end{cases} \tag{9}$$

If a failure is detected, the task execution enters a failure state and a recovery action is triggered to restart the task.

IV. Experiments and Evaluation

To evaluate our approach, we conduct pick-and-place experiments using the humanoid robot ARMAR-6 [23]. We learn a task model using the task graphs of several successful executions of manipulation actions in the context of 1) a decommissioning scenario and 2) a household scenario. Furthermore, we conduct an ablation study to analyze how our approach performs with and without visual input.



(a) Scenes from the decommissioning scenario

(b) Scenes from the household scenario

Fig. 4. Random cluttered scenes from two scenarios. Unknown objects are randomly placed in a box or on a table, where they have to be picked from and placed into another box or table. We assume no prior knowledge about the objects.

A. Experimental Setup

For the decommissioning scenario, eight unknown objects with different shape and weight are randomly placed into a box as shown in Figure 4a. The goal is to empty the box by grasping the objects in the box, carrying them and placing them at a target location. For the household scenario, seven household objects are placed on a table as shown in Figure 4b, and the robot has to clear the table by grasping the objects and placing them at a target location. During task execution, the failure detection is running continuously and a recovery action is triggered when a failure is detected. Furthermore, each execution of the pick-and-place tasks is labeled by a human operator. The task execution is considered successful if and only if an object was grasped, carried to a target location and successfully placed there.

Both experiments were conducted using ARMAR-6. Grasp candidates for unknown objects are generated and executed using our previous work described in [24], [25]. The robot can use its right or left arm for grasp execution, depending on the reachability of the detected grasp candidates. The motions for reaching objects and for recovery are learned from kinesthetic teaching and executed using via-point movement primitives [26]. The system is implemented in the robot software framework ArmarX [27]. Experiments for both scenarios are shown in the accompanying video.

B. Data-Driven Task Model Learning

We execute a pick-and-place task in the decommissioning scenario and record the generated task graphs such as shown in Figure 3. The task was executed 50 times in total, of which 18 attempts were labeled successful. Next, the task graphs of the 18 successful executions are used to create Markov chains by counting the frequency of each transition in the task graph across all executions, as described in Section III-C. The task execution module automatically selects a separate model for the actions *pick*, *carry* and *place*, resulting in one learned model per action.

Figure 5 shows the resulting transition probabilities between states in the learned Markov chain for the *pick* action. Since self-transitions occur most of the time, some transitions have a very low probability in the learned task model. Notably, the crucial transition between *grasping* and *holding* occurs at most once per execution, while the transition of *grasping* to *idle* occurs more often due to noisy predicate detection. Another critical aspect is that the states *holding*, *lifting* and *carrying* only transition between each other, and never to other states. This means that once an object is in the hand, it must stay in the hand for the action *pick* to be successful. An example is shown in Figure 6, where the object is pulled out of the robot's hand while carrying it to a target location. In this case, a transition from *carrying* to *platform moves* is detected, which has a probability of 0 in $M^{a_{\text{carry}}}$ and is thus classified as a failure.

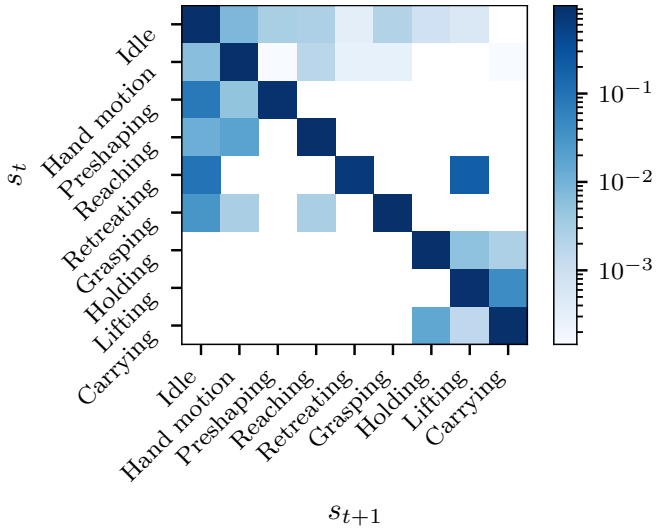


Fig. 5. Learned transition probability matrix from recordings of the *pick* action as described in Section IV-B, visualized as a heat map with logarithmic scaling. The probabilities are normalized per row, which represents the transitions from a source state s_t to any consecutive state s_{t+1}.

C. Experimental results

We conducted two experiments in a decommissioning and a household scenario.

In the first experiment, we executed 103 grasp attempts in the decommissioning scenario. The results are shown as confusion matrices in Figure 7. Overall, 40 task executions were successful, of which 31 were correctly labeled by the failure detection as success, and 9 were labeled as

Fig. 6. An example for a failure case from the real-world experiments. The grasped object is pulled out of the robot's hand while carrying it to the target location. Since there is no longer an object in the hand, a transition from *carrying* to *platform moving* is detected. The robot detects this as a failure according to the task model and restarts the task by lowering the arms and navigating back to the box. A video showing the full experiment is attached to this contribution.

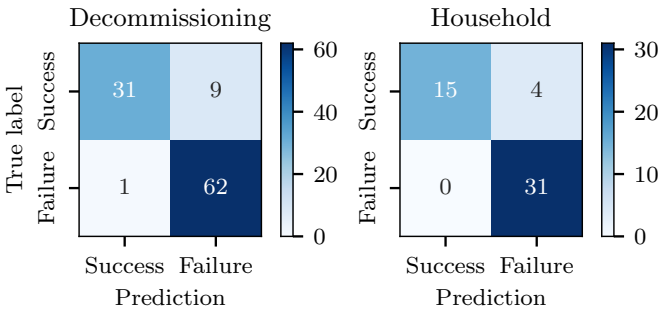


Fig. 7. Confusion matrices for the decommissioning task (left) and the household task (right).

TABLE II
ABLATION STUDY

Scenario	Vision	Precision	Recall	F1 Score
Decommissioning	✗	**0.897**	0.968	**0.931**
	✓	0.873	**0.984**	0.925
Household	✗	0.882	0.968	0.923
	✓	**0.886**	**1**	**0.939**
All	✗	**0.892**	0.968	0.928
	✓	0.877	**0.989**	**0.93**

failure, i.e., false positives. In most cases, this error occurs because the measured force by the wrist-mounted force-torque sensor was under the threshold of the predicate `Object_in_hand`, which results in a failure being detected according to the learned task model. Furthermore, placing the smaller objects did not always lead to sufficient visual change in the scene to exceed the threshold for the predicate `Object_placed`. Out of the 63 failed executions, only one was incorrectly classified as a success. In this case, the predicate `Object_in_hand` was wrongly detected after a failed grasp execution.

In the second experiment, we executed 50 grasp attempts in the household scenario. We used the same task model that was learned for the decommissioning scenario without changes. Most of the predicates' thresholds were left unchanged, with the only exception being the predicate `Object_placed`. For this predicate, we adjusted the threshold for spatial change because the used household objects are generally smaller than the pipes from the decommissioning scenario. In this scenario, our approach reached an F1-score of 93.9%. All failure cases were correctly identified by the failure detection, while four success cases were incorrectly labeled as failure. Like in the first scenario, this error can be explained by the faulty detection of the predicate `Object_in_hand`. However, the visual placement detection was more reliable in this scenario, because the household objects are all of similar size, unlike the objects used in the decommissioning scenario. The results show similar performance in both scenarios, thus showing that the learned task model can be transferred to different scenarios.

Finally, we analyze the impact of including visual input for detecting object placement in an ablation study. We reuse the data from the previous experiments, and discard the output of the visual predicate `Object_placed`. Table II shows the precision, recall and F1 score of both configurations (vision enabled and disabled) in each scenario. In both scenarios, incorporating visual information increases the recall, i. e., fewer failed executions are wrongly labeled as successful. However, the precision slightly decreases in the decommissioning task, as successfully placed objects were not always detected as such, thus increasing the false positive rate.

V. Conclusion

In this paper, we proposed a novel approach for learning multi-modal symbolic success and failure detection for grasping and mobile manipulation tasks. We defined a set of action predicates that are extracted from multi-modal sensory data and used action-specific decision trees to estimate the high-level symbolic state of the robot. This enabled learning task models for failure detection, while also externalizing the decision making process to a human operator. To evaluate our approach, we first learned an execution model of a pick-and-place task from repeated execution of the task and manual labeling of successful cases. The learned task model was then used in two different scenarios, reaching an F1-score of 93%. The experimental results showed that the system is capable of detecting failures during task execution, thus reducing the time to complete the task. We showed that including visual input for verifying the object placement improves the performance in some cases.

In future work, we plan to extend our approach in several ways. Firstly, we want to improve the detection of force-torque- and vision-based predicates, as these were the limiting factor in our evaluation. By integrating Bayesian filters and deep learning methods into the framework presented in

this paper, we hope to achieve robust high-level symbolic state estimation while maintaining the aspect of explainability. To further increase the autonomy of humanoid robots, we want to extend our system to predict failures and plan recovery actions in mobile manipulation tasks.

REFERENCES

[1] M. Fox, J. Gough, and D. Long, "Detecting execution failures using learned action models," in *Proceedings of The National Conference on Artificial Intelligence*, vol. 22, no. 2, 2007, p. 968.

[2] E. Di Lello, M. Klotzbücher, T. De Laet, and H. Bruyninckx, "Bayesian time-series models for continuous fault detection and recognition in industrial robotic tasks," in *IEEE/RSJ international conference on intelligent robots and systems*, 2013, pp. 5827–5833.

[3] W. Luo, J. Rojas, T. Guan, K. Harada, and K. Nagata, "Cantilever snap assemblies failure detection using svms and the rcbht," in *IEEE International Conference on Mechatronics and Automation*, 2014, pp. 384–389.

[4] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wüthrich, and S. Schaal, "Data-driven online decision making for autonomous manipulation." in *Robotics: science and systems*, 2015.

[5] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 407–414.

[6] A. Inceoglu, G. Ince, Y. Yaslan, and S. Sariel, "Failure detection using proprioceptive, auditory and visual modalities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2491–2496.

[7] S. Luo, H. Wu, H. Lin, S. Duan, Y. Guan, and J. Rojas, "Fast, robust, and versatile event detection through hmm belief state gradient measures," in *27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2018, pp. 1–8.

[8] D. Park, H. Kim, and C. C. Kemp, "Multimodal anomaly detection for assistive robots," *Autonomous Robots*, vol. 43, no. 3, pp. 611–629, 2019.

[9] X. Zhou, H. Wu, J. Rojas, Z. Xu, and S. Li, *Nonparametric Bayesian Learning for Collaborative Robot Multimodal Introspection*. Springer Nature, 2020.

[10] S. Luo, H. Wu, S. Duan, Y. Lin, and J. Rojas, "Endowing robots with longer-term autonomy by recovering from external disturbances in manipulation through grounded anomaly classification and recovery policies," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 1–40, 2021.

[11] E. Moreira, L. F. Rocha, A. M. Pinto, A. P. Moreira, and G. Veiga, "Assessment of robotic picking operations using a 6 axis force/torque sensor," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 768–775, 2016.

[12] D. Park, H. Kim, Y. Hoshi, Z. Erickson, A. Kapusta, and C. C. Kemp, "A multimodal execution monitor with anomaly classification for robot-assisted feeding," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5406–5413.

[13] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[14] G. R. Moreira, G. J. Lahr, T. Boaventura, J. O. Savazzi, and G. A. Caurin, "Online prediction of threading task failure using convolutional neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2056–2061.

[15] V. Arapi, Y. Zhang, G. Averta, M. G. Catalano, D. Rus, C. Della Santina, and M. Bianchi, "To grasp or not to grasp: an end-to-end deep-learning approach for predicting grasping failures in soft hands," in *3rd IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 653–660.

[16] M. Fox, M. Ghallab, G. Infantes, and D. Long, "Robot introspection through learned hidden markov models," *Artificial Intelligence*, vol. 170, no. 2, pp. 59–113, 2006.

[17] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Automatic segmentation and recognition of human activities from observation based on semantic reasoning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 5043–5048.

[18] ——, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artificial Intelligence*, vol. 247, pp. 95–118, 2017.

[19] J. Rojas, K. Harada, H. Onda, N. Yamanobe, E. Yoshida, K. Nagata, and Y. Kawai, "A relative-change-based hierarchical taxonomy for cantilever-snap assembly verification," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 356–363.

[20] O. Kroemer, H. Van Hoof, G. Neumann, and J. Peters, "Learning to predict phases of manipulation tasks as hidden states," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4009–4014.

[21] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[22] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal, "Towards associative skill memories," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 309–315.

[23] T. Asfour, M. Wächter, L. Kaul, S. Rader, P. Weiner, S. Ottenhaus, R. Grimm, Y. Zhou, M. Grotz, and F. Paus, "ARMAR-6: A High-Performance Humanoid for Human-Robot Collaboration in Real World Scenarios," *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 108–121, 2019.

[24] C. Pohl, K. Hitzler, R. Grimm, A. Zea, U. D. Hanebeck, and T. Asfour, "Affordance-based grasping and manipulation in real world applications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9569–9576.

[25] R. Grimm, M. Grotz, S. Ottenhaus, and T. Asfour, "Vision-based robotic pushing and grasping for stone sample collection under computing resource constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[26] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter-and extrapolation capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4301–4308.

[27] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, "The robot software framework armarx," *it-Information Technology*, vol. 57, no. 2, pp. 99–111, 2015.