Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Improving the Reliability of Service Robots in the Presence of External Faults by Learning Action Execution Models

Alex Mitrevski, Anastassia Kuestenmacher,
Santosh Thoduka, and Paul G. Plöger

May 31, 2017

Motivation

While releasing objects, service robots are prone to execution failures

Insufficient knowledge about the preconditions of release actions is one cause of such failures

Focus

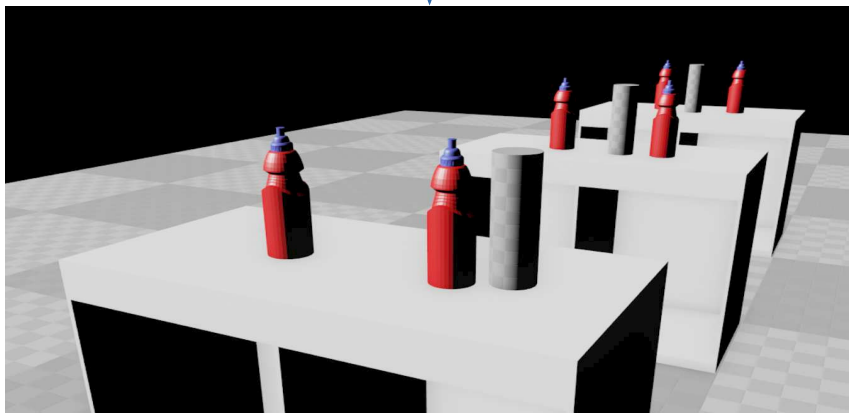
Our work deals with failures caused by physical object properties

We particularly address the following questions:

- How to represent action execution knowledge
- How to generalise execution knowledge
- How to update the learned models

Learning Execution Models: General Approach

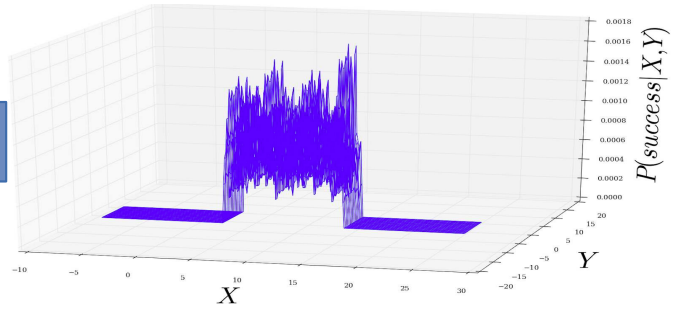
Detect fault → Import scene → Create training set → Learn execution model → Generate execution poses → Execute action



Symbolic model

```
rightOf(ChipsCan, Bottle1) & dc(ChipsCan, Bottle1) &
leftOf(ChipsCan, Bottle2) & dc(ChipsCan, Bottle2) &
leftOf(Bottle1, Bottle2) & dc(Bottle1, Bottle2)
```

Geometric model



Action Execution Model Representation

We represent execution models as task description templates that we call δ-models

δ-models have two components:

1. *symbolic model* - used for template matching
2. *geometric model* - used for sampling execution poses

δ-Models Primer: Global vs. Local Geometric Reasoning

Considering a configuration of static objects

- global reasoning attempts to find a geometric solution by analysing the configuration as a whole
- local reasoning finds a solution by considering local regions between different static objects

Local solutions can be identified using pairwise object relations

This idea has already been explored [1,2]

[1] K. Zampogiannis, Y. Yang, C. Fermller, and Y. Aloimonos, "Learning the Spatial Semantics of Manipulation Actions through Preposition Grounding," in *Robotics and Automation (ICRA), 2015 IEEE Int. Conf., May 2015*, pp. 1389-1396.

[2] R. Dearden and C. Burbridge, "Manipulation Planning using Learned Symbolic State Abstractions," Robotics and Autonomous Systems, vol. 62, no. 3, pp. 355-365, 2014.

Symbolic Model Representation

The first component of our representation is a symbolic model:

Symbolic template model

$$S(C^{\delta}) = (P_{1:k}(O_1), P_{k+1:p}(O_1, O_2), P_{1:k}(O_2))$$

where C^{δ} is a constrained configuration with two static objects O_1 and O_2 and P is a predefined predicate set

Symbolic template model: special case

$$S(C^{\delta^{\varnothing}}) = (P_{1:k}(O))$$

Symbolic Template Matching

$S(C^{\delta})$ is a template that can be searched for in an arbitrary configuration

We perform symbolic template matching to decide if a model is applicable for execution

Template configuration

New configuration

Template Configurations

Not every configuration of objects makes a good template

Badly chosen template

Configuration not covered by the template

Geometric Model Representation

The second component of our representation is a geometric sampling model (GSM)

A GSM encodes the success distribution of a set of sample action executions:

Geometric sampling model

$$GSM(C^{\delta}) = \mathrm{hist}_{\mathsf{success}}(C^{\delta})$$

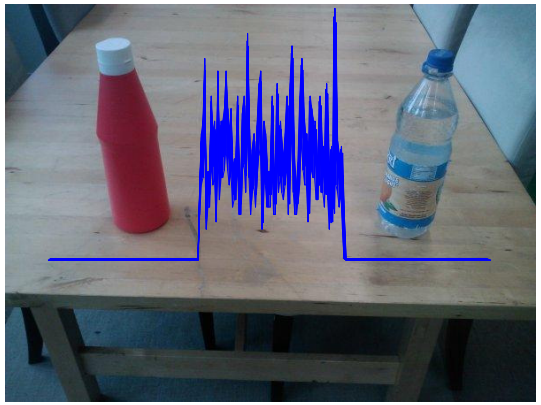
Geometric sampling model: special case

$$GSM(C^{\delta^{\varnothing}}) = \mathrm{hist}_{\mathsf{success}}(C^{\delta^{\varnothing}})$$

Learning Geometric Models

We create a GSM in a configuration symbolically equivalent to the template configuration

Template configuration



Symbolically equivalent configuration

GSM scaled back to the template configuration

Geometric Pose Sampling

GSMs can be stretched or compressed in order to match new configurations

We use GSMs for sampling execution poses proportional to their success probability:

Execution pose sampling

$$\mathbf{x} \sim GSM(C^{\delta})$$

Learning δ-Models

Learning is a four-step procedure:

1. Simulating execution in the template configuration and collecting a set of learning poses
2. Learning a symbolic model
3. Creating a geometric mapping of the symbolic data
4. Creating a GSM

Learning δ-Models

Symbolic learning

$$c(\bar{P}) = \sum_{i=1}^{R} \sum_{j=1}^{L} |\bar{P}_j - D_{ij}^{+}|$$

$$f(\bar{P}) = 1 - \frac{c(\bar{P})}{c_{\#}}$$

$$\bar{P} = \arg\max_{\bar{P}} f(\bar{P})$$

- $\bar{P}$: candidate symbolic model
- D^{+}: symbolic set of positive executions in C^{δ}
- $c\#$: possible bit error count

Geometric mapping[3]

$$\mathrm{P}(\bar{P}_i|X) = \frac{\mathrm{P}(X|\bar{P}_i)}{\mathrm{P}(X|\bar{P}_i) + \mathrm{P}(X|\neg\bar{P}_i)}$$

$$\mathrm{P}(X|\bar{P}_i) = \frac{1}{d_{\bar{P}_i+}} \sum_{i=1}^{d_{\bar{P}_i+}} e^{-\frac{\left\|x_i^{+} - x\right\|^2}{h^2}}$$

$$\mathrm{P}(X|\neg\bar{P}_i) = \frac{1}{d_{\bar{P}_i-}} \sum_{i=1}^{d_{\bar{P}_i-}} e^{-\frac{\left\|x_i^{-} - x\right\|^2}{h^2}}$$

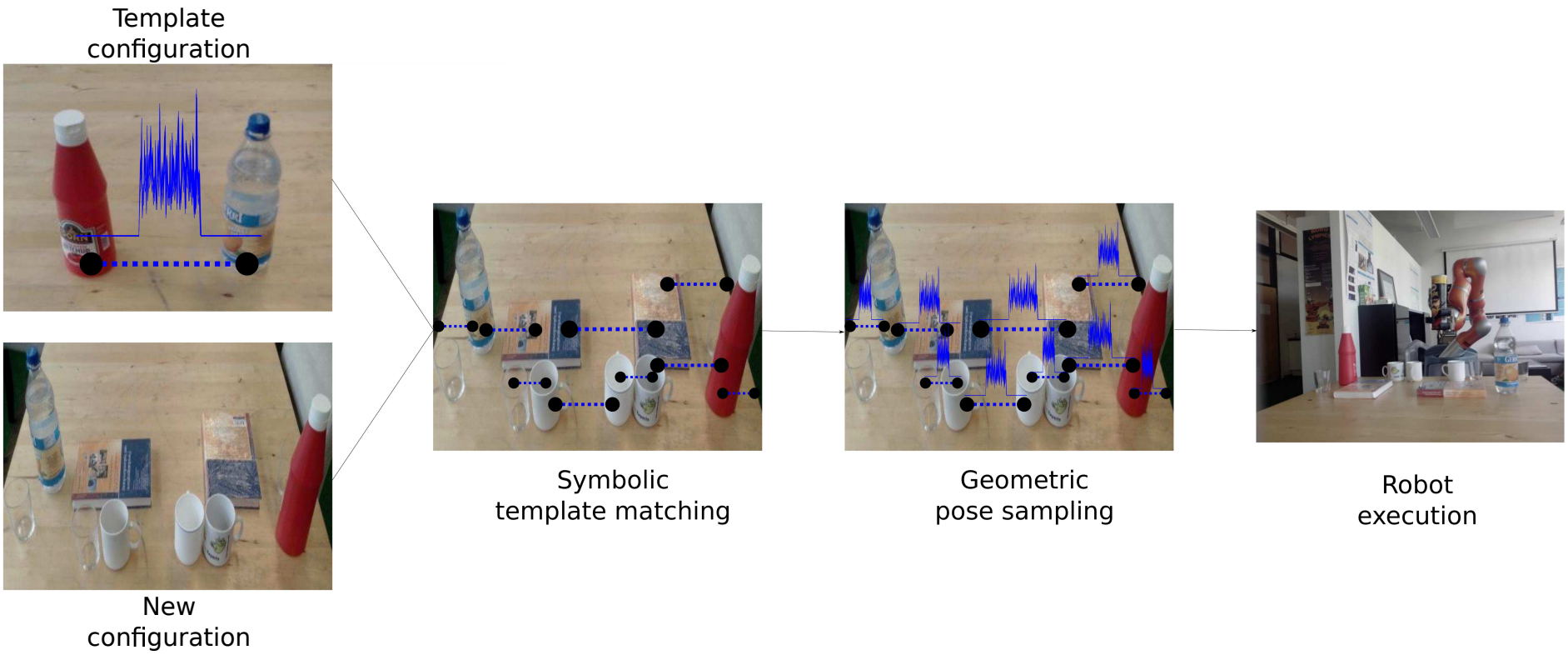
[3] R. Dearden and C. Burbridge, "Manipulation Planning using Learned Symbolic State Abstractions," Robotics and Autonomous Systems, vol. 62, no. 3, pp. 355-365, 2014.

Applying δ-Models

Applying these models is a four-step procedure as well:

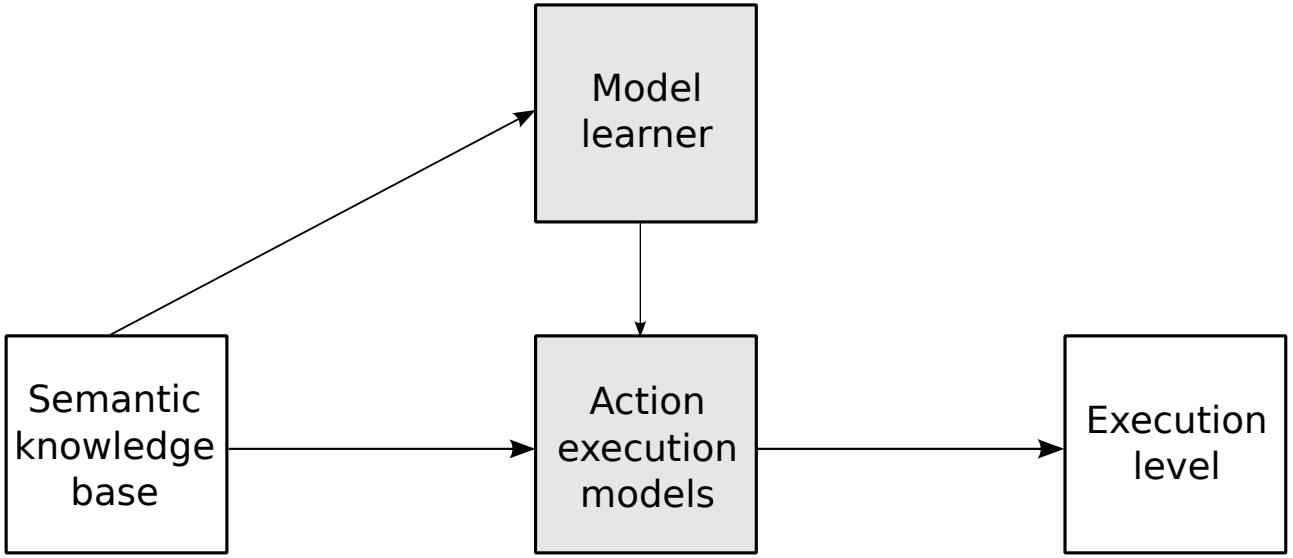
1. Recognising objects and instantiating a robot-independent simulation
2. Identifying the symbolic template in the current configuration
3. Collecting a set of robot-independent successful execution poses by sampling from the GSM
4. Sorting the poses based on robot-dependent criteria and selecting one of them for execution

Applying δ-Models



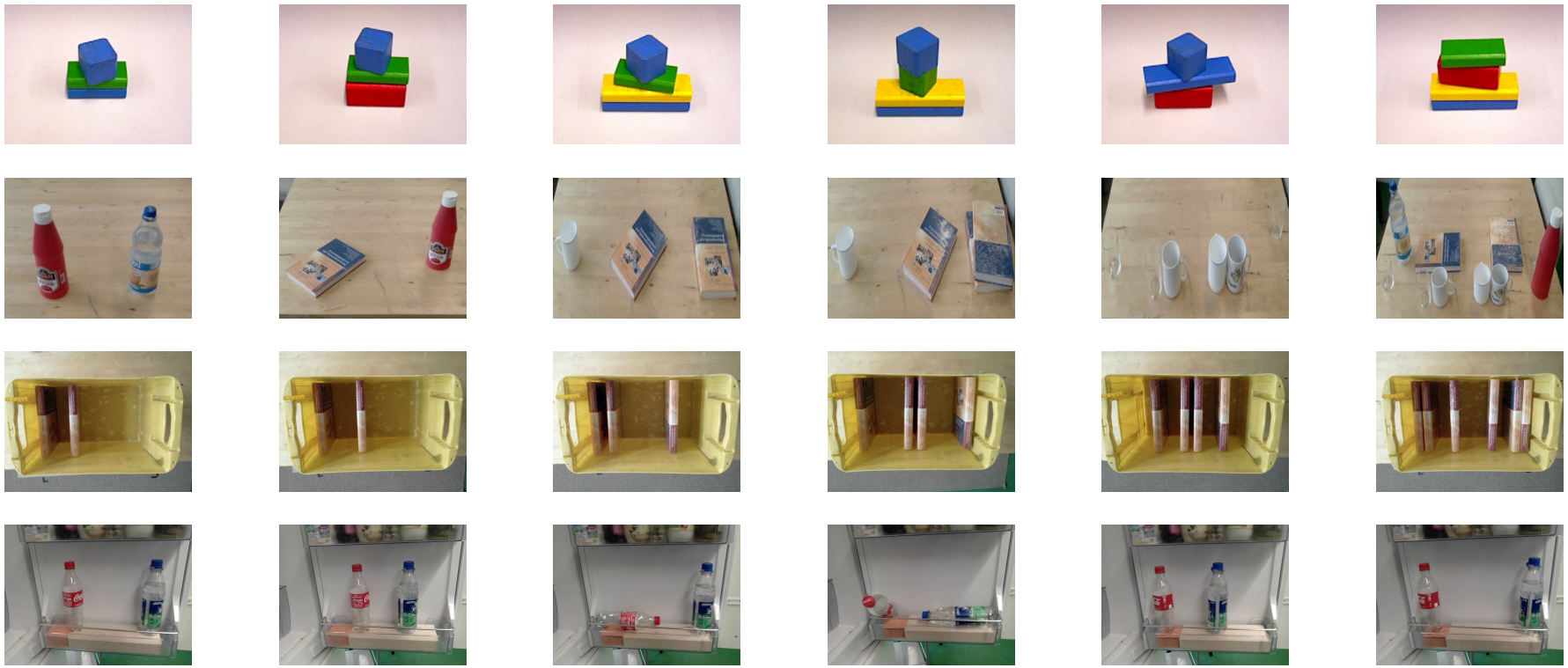
Level of Operation

δ-models are partially grounded by definition and can only be used after a high-level plan has been generated



For a fully autonomous agent, a semantic knowledge base should provide information about the nature of a given problem, which determines whether a δ- or a $\delta^{\varnothing}$-model should be used

Experiments: Scenarios



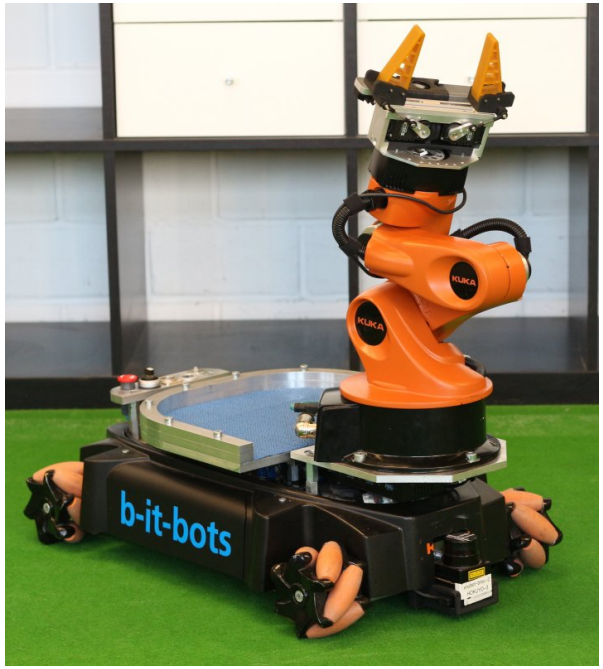
A B C D E F

Use cases from top to bottom: block tower, table, container, fridge

Experiments: Robots

We performed our experiments with two different robots[4]

Care-O-bot 3



KUKA youBot

The youBot was used for the block tower use case; we used the Care-O-bot for the other three use cases

[4] https://mas-group.inf.h-brs.de

Experiments: Predicate Set

$$P = \{on/2, leftOf/2, rightOf/2, above/2, \\ below/2, behind/2, inFrontOf/2, ntpp/2, \\ ntpp_i/2, ec/2, dc/2, eq/2, po/2\}$$

This set combines predicates that have been used elsewhere[5,6]

We assume that all predicates are expressed with respect to a right-handed coordinate system that is fixed at one of the edges of the destination surface and has its z-axis pointing up

[5] K. Zampogiannis, Y. Yang, C. Fermller, and Y. Aloimonos, "Learning the Spatial Semantics of Manipulation Actions through Preposition Grounding," in *Robotics and Automation (ICRA), 2015 IEEE Int. Conf., May 2015*, pp. 1389-1396.

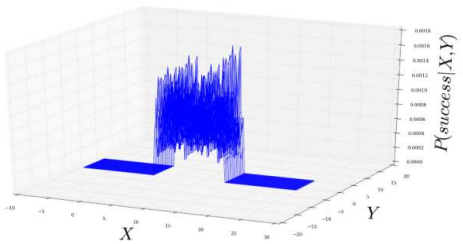
[6] N. Akhtar, A. Kuestenmacher, P. G. Plöger, and G. Lakemeyer, "Simulation-based approach for avoiding external faults," in *Int. Conf. Advanced Robotics (ICAR)*, 2013.

Experiments: Analysis

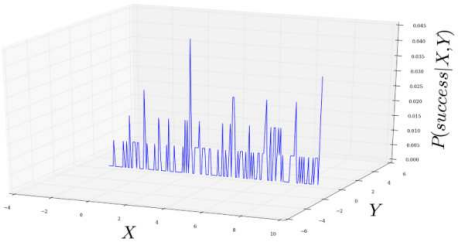
Learned symbolic models

Fridge	$rightOf(O_M, O_1) \wedge dc(O_M, O_1) \wedge$ $leftOf(O_M, O_2) \wedge dc(O_M, O_2) \wedge$ $\mathbf{leftOf(O_1, O_2) \wedge dc(O_1, O_2)}$
Table	$rightOf(O_M, O_1) \wedge dc(O_M, O_1) \wedge$ $leftOf(O_M, O_2) \wedge dc(O_M, O_2) \wedge$ $\mathbf{leftOf(O_1, O_2) \wedge dc(O_1, O_2)}$
Container	$rightOf(O_M, O_1) \wedge$ $above(O_M, O_1) \wedge dc(O_M, O_1) \wedge$ $leftOf(O_M, O_2) \wedge$ $above(O_M, O_2) \wedge dc(O_M, O_2) \wedge$ $\mathbf{leftOf(O_1, O_2) \wedge dc(O_1, O_2)}$
Block tower	$above(O_M, O_1) \wedge po(O_M, O_1)$

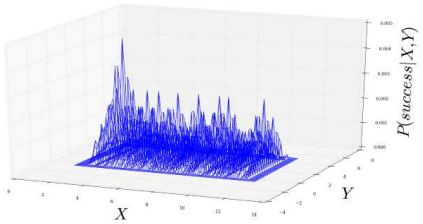
Geometric sampling models



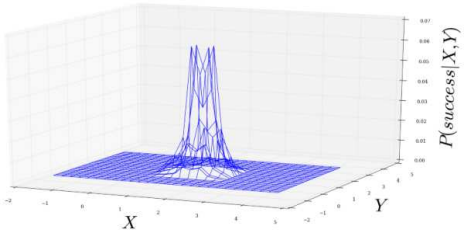
Table



Container



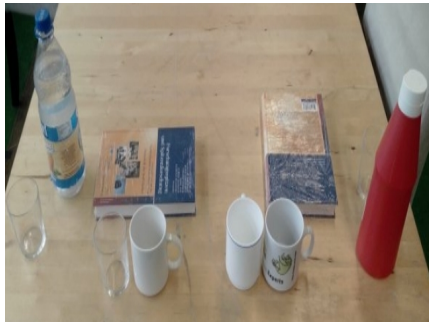
Fridge



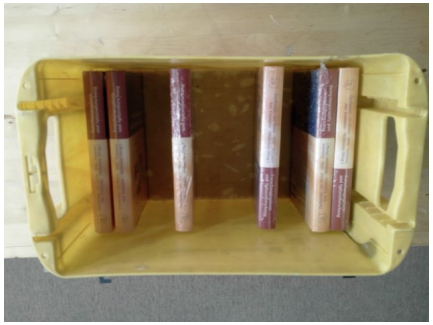
Block tower

Experiments: Results

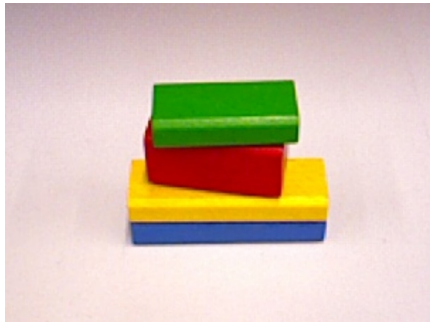
Fridge



Table



Container



Block tower

Successful runs (out of 10)

	A	B	C	D	E	F
Fridge	9	10	9	10	8	8
Table	10	10	10	10	10	8
Container	10	10	10	10	10	10
Block tower	9	8	8	8	8	10

Assumptions and Limitations

Assumptions:

- Reliable perception, navigation, grasping, and trajectory planning
- Known action effects
- Known objects (3D models of the objects are available)

Assumptions and Limitations

Assumptions:

- Reliable perception, navigation, grasping, and trajectory planning
- Known action effects
- Known objects (3D models of the objects are available)

Limitations:

- Deciding whether a δ- or a $\delta^{\varnothing}$-model should be used is difficult
- Generalisation of the models to configurations with significantly different passive objects cannot be guaranteed
- Rotation of passsive objects not explicitly taken into account during pose generation

Future Work

- Improving the formal description of template configurations
- Relaxing the main assumptions
- Defining a strategy for biasing the model learning process with semantically related model representations
- Expanding the framework so that action effects can be learned as well