Opening Doors: An Initial SRL Approach

Bogdan Moldovan[1], Laura Antanas[1], and McElory Hoffmann[1,2]

[1] Department of Computer Science, Katholieke Universiteit Leuven, Belgium
[2] Department of Mathematical Sciences, Stellenbosch University, South Africa

Abstract. Opening doors is an essential task that a robot should perform. In this paper, we propose a logical approach to predict the action of opening doors, together with the action point where the action should be performed. The input of our system is a pair of bounding boxes of the door and door handle, together with background knowledge in the form of logical rules. Learning and inference are performed with the probabilistic programming language ProbLog. We evaluate our approach on a doors dataset and we obtain encouraging results. Additionally, a comparison to a propositional decision tree shows the benefits of using a probabilistic programming language such as ProbLog.

Keywords: robotics, mobile manipulation, door opening and grasping, logical rules, ProbLog

1 Introduction

In the context of the EU-project on *Flexible Skill Acquisition and Intuitive Robot Tasking for Mobile Manipulation in the Real World*[3] (FIRST-MM), one of the goals of autonomous robots is to perform mobile manipulation tasks in indoor environments. In this setting, an essential condition is that the robot can operate doors during navigation. A complete solution to this general problem requires a system that can solve several tasks: detecting and localising the door and its handle, recognising the grasping points, finding the right actionable point and action and finally, performing the action on the handle. In this work we focus on two of these tasks: detecting the actionable point and the action movement itself. We assume that the door and handle are detectable by the robot. This is an object detection problem that has been addressed previously in the literature using several approaches, using either 2D [1] or 3D information [3, 4, 19].

Detecting the action and action points is a challenging manipulation task. It depends on the sensorimotor control of the robot, the gripper, the type of handle and the properties of the door. To be opened, each door requires different actions depending on the side of the door that the robot is approaching. The action also depends on the type of the handle, its relative position to the door and sometimes even the objects around the door. Usually if hinges are detected on the side of the door and the light switch on the opposite side next to the handle, the door

[3] More information available at: http://www.first-mm.eu

needs to be pulled. Similarly, while the shape of the handle can be quite a good indicator of a suitable action point (i.e., knob), sometimes it cannot be detected reliably, for example when an L-shaped handle is confused with a horizontal one. This may directly influence valid graspable points, given the robot hand type, and limit the actionable handle points. In this case, the relative position of the contact points with the door, the relative positions of the candidate action point to the door sides or other points in the point cloud of the handle may play a key role for grasping and performing the action [18, 11].

In the cases mentioned above, generalisations over opening directions, point positions in the point cloud and types of handles are needed. Additionally, it is highly likely that a robot will encounter a door or handle that it has not seen before. Hence it is not possible to enumerate all possible doors and handles. Moreover, the objects in the surrounding environment, such as the hinges, play a role in predicting the correct action and action point for opening the door. Following this idea, Rosman and Ramamoorthy [16] introduced approaches to learn the symbolic spatial relationships between objects which allow performing various robotic tasks. Thus, they take the relations in the environment into account. Similarly, Bereson et al. [2] takes contextual information into account when planning grasps.

In this work we propose the use of statistical relational learning (SRL) [5, 14] to generalise over doors and handles in order to predict door opening actions and handle actionable points. SRL combines logical representations, probabilistic reasoning mechanisms and machine learning and thus is suitable for our task. Although several existing SRL approaches could be used to solve our problem [10, 15], in this work we consider *probabilistic logic programming languages* (PPLs). Our choice is mainly motivated by the fact that PPLs are specially designed to describe and infer probabilistic relational models or, more generally, logic theories. In particular, we use Problog [6], a PPL, that allows us to specify our problem declaratively, using symbolic rules, while being able to deal with noisy data due to its probabilistic language. Other SRL frameworks could be used as well, however ProbLog is very convenient for encoding world knowledge due to: i) its declarative nature; ii) the fact that weights can be directly viewed as probabilities; iii) its capability to handle numerical values, which is something to consider when dealing with point clouds (for future developments of this work).

Thus, the main contribution of this paper is the use of a ProbLog theory to predict the actions to open doors and well as the action points. We use as input solely extracted properties of door images. In our logical representation of the domain, every visual scene is mapped to a logical interpretation. However, ProbLog could also model the noisy nature of the detection aspects and the uncertainty of the environment where the robot operates. As a key element, we encode background knowledge about the domain, in a natural way, using a set of logical rules which reduce the number of parameters of the theory. Finally, we use a learning from interpretations setting [7] to learn the parameters of our ProbLog theory. The approach is general enough to be able to deal with point clouds as well as 2D visual images.

We evaluated our approach on a dataset containing 60 images of doors. The results are promising and motivate us to continue this work with a real robot scenario. Additionally we compare against a random classifier, a majority classifier and a propositional decision tree. We report superior results using our relational approach, which shows the role of background knowledge when solving the opening door problem. Some work on predicting how to best open a door by a robot setting exists [12]. However, it does not make use of logical representations as we do.

The outline of this paper is as follows: in Section 2 we introduce the problem and the approach used to solve it, and in Section 3 we present our learning and inference setting. We show experimental results in Section 4, after which we conclude and mention future directions in Section 5.

2 Problem Modeling

2.1 Problem Description

We first introduce an initial setting for a high-level logical and relational reasoning system that can be used by a robot for opening doors. We assume the robot is able to detect doors and door handles, so we assume to have access to bounding boxes in the image for both the door frame and the door handle. Figure 1 presents two such examples of detected door frame (in red) and handle (in blue) with their bounding boxes. Later, we can add prior probabilities on the positions of the frame and handle for a more realistic scenario which involves object detection uncertainty. The setting can be expanded to include other detected objects in the environment to help us identify the action needed to open the door by providing additional relational contextual cues.

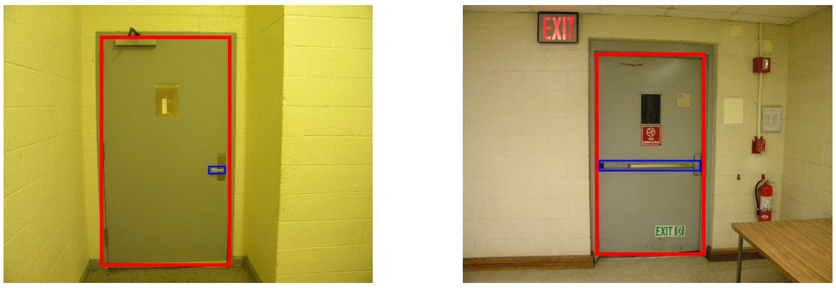


Fig. 1. Annotated doors: **(a)** push down action, **(b)** push in.

In this setting, we are interested in predicting the high-level (initial) action the robot needs to perform in order to open the door, and where this action should be applied (action point). Once these are determined, the robot can grasp the handle and execute the action. In a more advanced setting it can be imagined that we can generalise over possible grasping points depending on the specific robot hand. Here, we are just interested in predicting the action/action point pairs.

We assume that the robot can open a door by pushing it in any of the six 3D directions, labelled as *in*, *out*, *left*, *right*, *up*, *down*, and turning the handle in two directions: *clockwise* and *counterclockwise.* In total, there are eight possible high-level actions. At a high-level, we think of the action point in terms of which end of the handle needs to be acted upon, so we will discretise this into 5 different values: *up*, *down*, *left*, *right* and *centre.* For future work, we plan to upgrade our model with continuous actionable points in the handle point cloud.

Although currently the proposed model does not include explicit relations between objects or object points in the scene, it can be easily extended to incorporate such information. For example, if the object is represented by a point cloud, knowing that two points are very close spatially allows us to exploit symetries and redundancies in the domain. By using such a relation we can generalize over points in the cloud, and thus produce a more compact model.

2.2 Approach

From the bounding boxes for the door frame and handle we obtain a set of positions in the $x - y$ plane for both the door and handle: $(x_{min}, y_{min}, x_{max}, y_{max})$. Based on these we define five features $(F_1, ..., F_5)$, namely: the handle aspect ratio, the handle width relative to the door width (or handle relative width), the handle height relative to the door height (or handle relative height), the position of the centre of the handle relative to the door frame in the x-axis and in the y-axis (or handle relative width/height positions).

We assume that these features are independent and additionally, we discretize them. Handle aspect ratio can take the values {*big-width*, *small-width*, *square*, *small-height*, *big-height*}. Handle relative width and height can take the values {*small*, *medium*, *large*}. Finally, the position of the handle relative to the door can take the values {*center-pos*, *up-pos*, *down-pos*, *left-pos*, *right-pos*} on the x-axis or y-axis.

Action prediction An initial intuition is that we can use a Naive Bayes classifier [17] in order to predict the action based on these features. The Bayesian Network for our Naive Bayes model is illustrated in Figure 2.2. Given our computed features $F_1, ..., F_5$ from the observed x and y positions of the bounding boxes of the door frame and handle and using our independence assumption, we can compute the conditional probability [17] of an action A as:

$$P(A|F_1, ..., F_5) = \frac{P(A) * P(F_1, ..., F_5|A)}{P(F_1, ..., F_5)} = \frac{P(A) * P(F_1|A) * ... * P(F_5|A)}{P(F_1) * ... * P(F_5)}.$$

Then, in order to predict A, we compute the maximum a posteriori (MAP) probability estimate as: $\arg\max_A P(A) * \prod_{i=1}^{5} P(F_i|A)$. However, in a fully propositional setting this requires the learning of many parameters, even in such a small domain with five features, taking values from a small discretised set. We propose to go towards a relational setting, where background knowledge can be used as a set of logical rules to reduce the number of parameters that need to be learnt, and thus the number of learning examples that need to be used, and at a later stage to generalise over our setting.

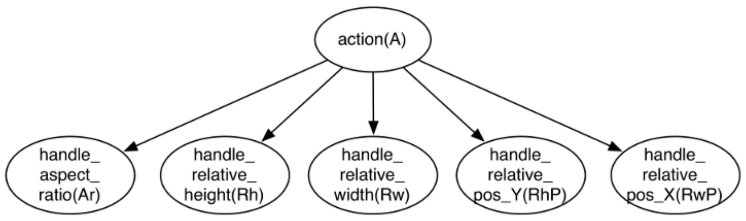

action(A)
handle_ aspect_ ratio(Ar)
handle_ relative_ height(Rh)
handle_ relative_ width(Rw)
handle_ relative_ pos_Y(RhP)
handle_ relative_ pos_X(RwP)

Fig. 2. The Bayesian Network used for the Naive Bayes classifier for action prediction.

Action point prediction The action point can be determined both by the type of the already predicted action and the features $F_1, ..., F_5$. Since the action and these features are not independent, we cannot use our previous approach that we used for predicting the action. For the purpose of predicting the action point we define a Bayesian Network and learn its parameters. The Bayesian Network is illustrated in Figure 3. In order to predict the position, we need to compute: $\arg\max_{Pos} P(Pos|A, F_1, ..., F_5)$.

Similarly to the action prediction task, we augment our model with background knowledge in the form of logical rules which constrain the action point based on the action and related features. For example, a *push in* or *out* on a handle with a big width or height (or big aspect ratio) should be done at the centre of the handle. This also helps us reduce the number of parameters that we need to learn. In practice, our experiments show that the action point prediction is influenced only by the action, relative position of the centre of the handle on the x-axis and handle relative height.

For both tasks we can use ProbLog to compute the probabilities. In the next section we describe our learning and inference setting with ProbLog.

3 Learning and Inference with ProbLog

ProbLog is a probabilistic extension of the Prolog programming language. We first review the main principles and concepts underlying ProbLog. Afterwards, we explain how we perform learning and inference with our model within the ProbLog language.

In ProbLog – as in Prolog – an atom is an expression of the form $a(t_1, \ldots, t_k)$ where a is a predicate of arity k with $t_1, \ldots, t_k$ terms. A term can be a variable, a

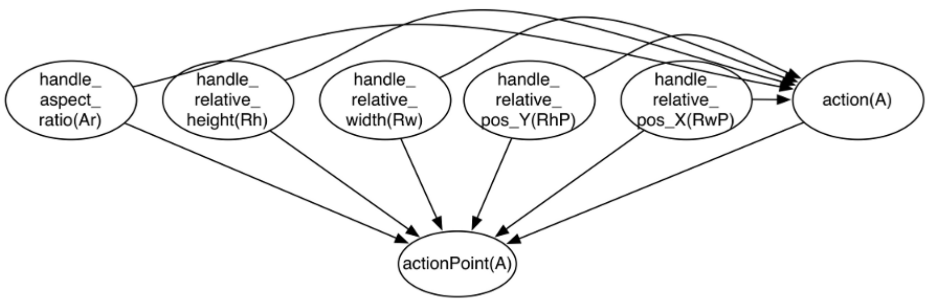

handle_ aspect_ ratio(Ar)
handle_ relative_ height(Rh)
handle_ relative_ width(Rw)
handle_ relative_ pos_Y(RhP)
handle_ relative_ pos_X(RwP)
action(A)
actionPoint(A)

Fig. 3. The Bayesian network for action point prediction.

constant, or a functor applied to other terms. An expression is called ground if it does not contain variables. Definite clauses are universally quantified expressions of the form $h : -b_1, \ldots, b_n$, where h and b_i are atoms. A fact is a clause without the body. Additionally, in ProbLog a clause c_i may be labeled with a probability p_i in the form $p_i :: h : -b_1, \ldots, b_n$. Similarly, a ProbLog labeled fact signifies that the fact is true with probability p_i.

A ProbLog theory T consists of a set of facts F and a set of definite clauses BK which express the background knowledge. The semantics of a set of definite clauses is given by its least Herbrand model, the set of all ground facts entailed by the theory. If $T = \{p_1 :: c_1, \ldots, p_n :: c_n\}$, ProbLog defines a probability distribution over logic theories $L \subseteq L_T = \{c_1, \ldots, c_n\}$ in the following way: $P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i)$. Additionally, in ProbLog we are interested in computing the probability that a query q succeeds in the theory T. The success probability $P(q|T)$ corresponds to the probability that the query q has a proof, given the distribution over logic programs.

For our prediction tasks, we first build a ProbLog theory of the Bayesian Networks, which we augment with logical rules reflecting our background knowledge. The theory then supports inference for predicting both tasks by answering probabilistic queries. In our case the end goal query is the conditional probability of a particular action (or action point) given the set of observations made about the world, that is about the handle in relation with the door. The logical rules are (general) background knowledge and they are able generalize over different properties of the handle relative to the door, actions or action points. In this way, we generalise by reducing the number of parameters of a fully ground model. Because our goal is to solve both action prediction and action point prediction, our ProbLog theory will contain two models, one for each task.

We next present an excerpt from our ProbLog theory used to answer probabilistic queries.

```
%theory for action prediction

%parameters for the Naive Bayes model
```

```
0.0:: action(push_up).
0.43:: action(push_in).
0.05:: action(push_out).
...
0.74:: har(small_height,left).
0.11:: har(big_height,left).
...
0.31:: hrw(small,in).
0.13:: hrw(medium,in).
0.56:: hrw(large,in).
...
0.998:: hrh(small,turnc).
0.997:: hrh(small,turncc).
0.001:: hrh(large,turnc).
...
0.60:: hrwp(left_point,down).
0.004:: hrwp(center_pos,down).
0.24:: hrwp(center_pos,right).
...
0.017:: hrhp(down_pos,up).
0.34:: hrhp(center_pos,up).
0.001:: hrhp(up_pos,out).
...
%definite clauses for the Naive Bayes (action prediction)
handleAspectRatio(square):- action(turnc), har(square,turnc).
handleAspectRatio(square):- action(turncc), har(square,turncc).
handleAspectRatio(Ar):- action(Ac), har(Ar,Ac), Ar\=square.
handleAspectRatio(Ar):- action(Ac), har(Ar,Ac),
                        Ac\=turnc, Ac\=turncc.

handleRelativeWidth(large):- action(in), hrw(large,in).
handleRelativeWidth(large):- action(out), hrw(large,out).
handleRelativeWidth(Rw):- action(Ac), hrw(Rw,Ac), Rw\=large.
handleRelativeWidth(Rw):- action(Ac), hrw(Rw,Ac), Ac\=in, Ac\=out.

%similar relations for relative height
...
handleRelativePosition\_X(RwP):- action(Ac), hrhp(HrWp,Ac).
handleRelativePosition\_Y(RhP):- action(Ac), hrhp(HrHp,Ac).

%extended theory with action point prediction

%parameters for the Bayesian Network model
0.09:: ap(center,up,right_pos).
0.06:: ap(center,left,right_pos).
```

```
0.17:: ap(center,up,left_pos).
...
0.09:: ap(down,left,left_pos).

%definite clauses for the Bayesian Network
%(action point prediction)
actionPoint(center):- handleRelativeWidth(large).
actionPoint(center):- handleRelativeHeight(large).
actionPoint(center):- handleRelativeWidthPosition_X(center_pos).
actionPoint(center):- action(turnc).
actionPoint(center):- action(turncc).
actionPoint(center):- action(in).
actionPoint(center):- action(out).
actionPoint(center):- action(Ac), ap(center,Ac,HrWp),
                    handleRelativeWidthPosition_X(RwP), Ac\=turnc,
                    Ac\=turncc, Ac\=in, Ac\=out, RwP\=center_pos.
actionPoint(left):- action(Ac), ap(left,Ac,HrWp),
                  handleRelativeWidthPosition_X(RwP), Ac\=turnc,
                  Ac\=turncc, Ac\=in, Ac\=out, RwP\=center_pos.
...
%similarly for the rest of action points
```

The Naive Bayes model for action prediction can be directly encoded in ProbLog using rules such as:
$\mathtt{handleAspectRatio(Ar)} \leftarrow \mathtt{action(Ac), har(Ar, Ac)}$.
$\mathtt{handleRelativeWidth(Rw)} \leftarrow \mathtt{action(Ac), hrw(Rw, Ac)}$.
where $\mathtt{har(Ar, Ac)}$ is a probabilistic fact representing the conditional probability of the handle aspect ratio given the action and $\mathtt{hrw(Rw, A)}$ is a probabilistic fact representing the conditional probability of the handle relative width given the action. The probabilities of its groundings (e.g., $0.74 :: \mathtt{har(small\_height, left)}$, $0.11 :: \mathtt{har(big\_height, left)}, \dots$) are parameters of the model. However, to reduce them, we augment the rules with additional background knowledge. For example, the rule encoding the handle relative width feature can be extended to encode that if the handle relative width or height is large, the action that needs to be performed is either a push *in* or a push *out* (as illustrated in Figure 1(b)). We encode this in ProbLog using two rules the following way:
$\mathtt{handleRelativeWidth(large)} \leftarrow \mathtt{action(Ac), Ac = in, hrw(l, Ac)}$.
$\mathtt{handleRelativeWidth(large)} \leftarrow \mathtt{action(Ac), Ac = out, hrw(l, Ac)}$.
Similarly, if the handle is a knob, it needs to be turned in one of the two directions *clockwise* or *counterclockwise*. In this case the handle is characterized by a handle aspect ratio close to 1 (i.e. a bounding box close to a *square*) and we encode this case in ProbLog using two other rules:
$\mathtt{handleAspectRatio(square)} \leftarrow \mathtt{action(Ac), Ac = turn\_clock, har(square, Ac)}$.
$\mathtt{handleAspectRatio(square)} \leftarrow \mathtt{action(Ac), Ac = turn\_clockwise,}$
$\mathtt{har(square, Ac)}$.

Similar rules are defined for the action prediction task for:
`handleRelativeHeight(Rh)`
`handleRelativePosition_Y(RhP)`
`handleRelativePosition_X(RwP)`.

The model can also be extended later with background contextual knowledge gathered from the environment, like the presence of other objects near the door which could give an indication about how to open it. Furthermore, ProbLog allows us to add priors (e.g., Gaussian) on the x and y axis positions of the detected door frame and handle to model uncertainty in object detection.

The theory defined for the action prediction task can be extended with extra rules to encode the ProbLog model of the BN associated with the action point task. Similarly, it will contain the respective background knowledge in the form of logical rules, which enables us to generalise.

The BN model for action point prediction can be encoded in ProbLog using a rule such as:

```
actionPoint(center) : −action(Ac), ap(center, Ac, RwP),
                          handleRelativeWidthPosition_X(RwP).
```

where `ap(Point, Ac, RwP)` is a probabilistic predicate representing the conditional probability of the action point given the action and the relative position of the centre of the handle on the x-axis. These probabilities are the model parameters. Similarly, to reduce their number, we augment the rules with additional background knowledge. From experience we know that any turn action requires the robot to perform a caging grasp of the knob, which means grabbing the handle at the centre. This can be encoded as definite rules as follows:

```
actionpoint(centre) ← action(Ac), A = turn_counterclock.
actionpoint(centre) ← action(Ac), A = turn_clockwise.
```

Once these background rules are added, we would not need the model parameters `ap(center, turn_counterclock, RwP)` and `ap(center, turn_clockwise, RwP)` anymore. Similar rules one can define for the left, right, up and down action points. A clear advantage of such models is in the context of predicting exact points in a point cloud. Then, such rules will generalize over similarly spatially displayed points in the cloud.

Now our model is encoded via probabilistic facts and logical clauses. ProbLog can be used both to learn the parameters or answer probabilistic queries [9].

3.1 Learning

In order to perform either task, additionally to the theory we need examples, which in this case are facts characterizing the world, and thus each example defines an interpretation of the image. Interpretations are used for parameter learning in the training phase and as evidence to query the theory in the inference phase. An interpretation and example for learning the parameters of the door action prediction model is illustrated in Example 1.

Example 1.
`example(1).`

```
known(1,handleAspectRatio(small_height),true).
known(1,handleRelativeHeight(long),true).
known(1,handleRelativeWidth(medium),true).
known(1,handleRelativePosition_Y(up_point),true).
known(1,handleRelativePosition_X(left_point),true).
known(1,action(in),true).
```

We are in a learning from interpretation setting, thus we learn the parameters of our models using ProbLog LFI (or the learning from partial interpretations setting within ProbLog) [7]. Given a ProbLog program $T(p)$ where the parameters $p = \langle p_1, ..., p_n \rangle$ of the probabilistic labeled facts $p_i :: c_i$ in the program are unknown, and a set of M (possibly partial) interpretations $D = I_1, ..., I_M$, known as the training examples, ProbLog LFI estimates using a Soft-EM algorithm the maximum likelihood probabilities $\hat{p} = \langle \hat{p_1}, ..., \hat{p_n} \rangle$ such that $\hat{p} = \arg\max_P P(D|T(p)) = \arg\max_P \prod_{m=1}^{M} P_w(I_m|T(p))$, where $P_w(I)$ is the probability of a partial interpretation $I = (I^+, I^-)$ with the set of all true atoms I^+ and the set of all false atoms I^-. ProbLog LFI is also able to learn parameters in the case of partial observations, which is useful to generalise over the cases when the door or handle is not fully observed.

In essence, the algorithm constructs a propositional logic formula for each interpretation that is used for training and uses a Soft-EM to estimate the marginals of the probabilistic parameters. For more details please see [7][4].

3.2 Inference

After the learning phase setting, we performed inference in order to do either action recognition or action point prediction. It is assumed that the robot can observe the properties of the world (door and handle in our case) and it needs to infer which action needs to be performed to open the door and which point to action. This resumes to querying the ProbLog theory for the conditional probabilities $P(A|F_1, \ldots, F_5)$ for each of the 8 possible actions and $P(Pos|A, F_1, \ldots, F_5)$ for each of the 5 possible discretized action points. We then compute the MAP probability estimate of the action, and afterwards of the action point.

4 Experiments and Results

For the purpose of our initial experimental setup, we collected a set of 60 door images. Most of these were taken from the Caltech and Pasadena Entrances 2000 dataset[5] and from the CCNY Door Detection Dataset for Context-based Indoor Object Recognition [20]. To increase variation in the different types of doors and handles we also added a few images from a Google image search. The images were manually annotated with bounding boxes for the door and handles, as well

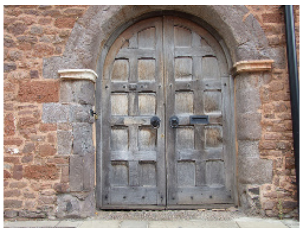
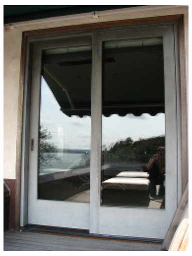
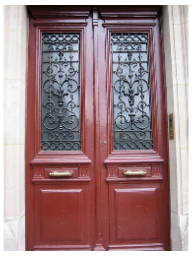
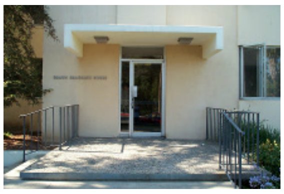
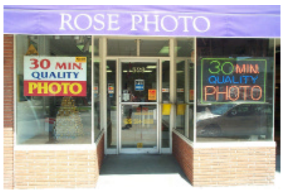
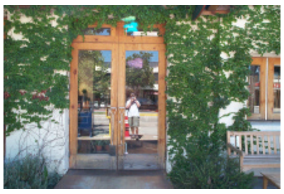
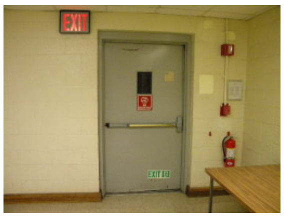
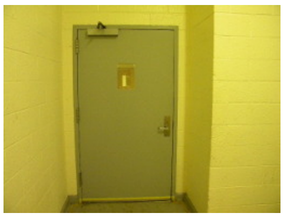
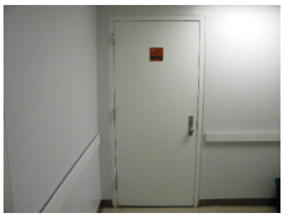
[4] In practice we have used the implementation available at: http://dtai.cs.kuleuven.be/problog/tutorial-learning-lfi.html.

[5] Available at: http://www.vision.caltech.edu/html-files/archive.html

as the action needed to open the door and the action point. We randomly split this dataset into two sets of 30 images, one to be used for training the ProbLog model by running ProbLog LFI to learn parameters, and one for testing by running inference to make first predictions about the action and then about the action point for the predicted action. Each experiment was run five times with different train and test sets and the results averaged.

In Table 1 we can see some examples of doors and the predictions produced by our logical approach. The correct predictions are shown in green, while the incorrect ones in red, with the ground truth value shown between brackets.

Table 1: Example doors and predictions: correct predictions are in green, incorrect ones in red. The ground truth value is between brackets.

 **Action:** turncc (out) **Position:** center (down)	 **Action:** in (right) **Position:** center (up)	 **Action:** in (in) **Position:** center (center)
 **Action:** out (out) **Position:** center (right)	 **Action:** in (in) **Position:** center (center)	 **Action:** in (in) **Position:** center (center)
 **Action:** in (in) **Position:** center (center)	 **Action:** down (down) **Position:** left (left)	 **Action:** turncc (turncc) **Position:** center (center)

On this dataset, we compared our approach against three other baseline approaches. We compared against the following approaches on our collected door dataset: a random classifier, a majority classifier, and a propositional learner setting, for which we used a decision tree. These approaches are intended as a baseline to justify the benefit of using our logical approach. We will show that we obtain better prediction rates for the two prediction tasks of action prediction and action point prediction on the dataset.

We first set up a random classifier. We obtained the prior probabilities for the action and action point for each of the five training sets of 30 images, and then use a random classifier for the two prediction tasks. The results are summarized in Table 2 for the action prediction task, and Table 3 for the action point prediction task.

Similarly, we then set up a majority classifier. We determine from the training sets which are the action and action point majority classes and use this information in a majority classifier on the test sets. The majority class for the action is *out* in three of the training data sets, and *in* in the other two. The majority class for the action point is *center* in all the training datasets. The results of the two prediction tasks are also summarized in Table 2 and Table 3.

For the propositional learner setting baseline, we modelled a decision tree classifier in Weka [8]. We used the J48 decision tree algorithm in Weka, a version of the C4.5 [13] decision tree algorithm. We trained the decision tree on the training sets data. An example of a learned decision tree from a training set for predicting the action needed to open the door can be seen in Figure 4. An example of a learned decision tree from a training set for predicting the action position needed to open the door can be seen in Figure 5.

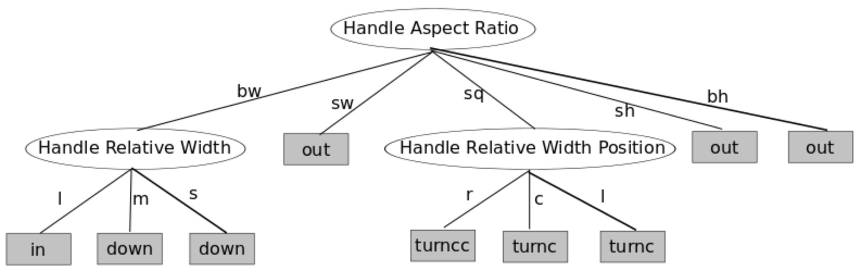


Fig. 4. Example obtained decision tree for predicting the action.

We used the learned decision trees on the tests sets. The results of the two prediction tasks are also summarized in Table 2 and Table 3.

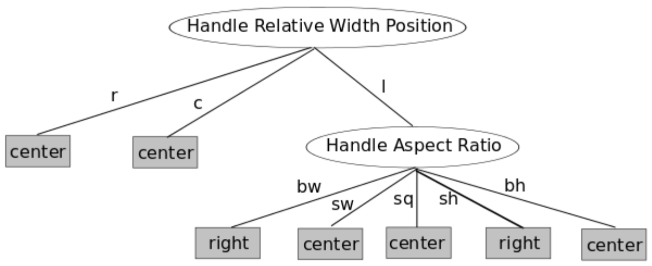


Fig. 5. Example obtained decision tree for predicting the action position.

Table 2: Action prediction.

Method	Total experiments	Avg. Success	Percentage
Random Classifier	30	8.6	28.67%
Majority Classifier	30	9.2	30.67%
Decision Tree	30	21.4	71.33%
Relational Approach	30	23.6	78.67%

The results of our preliminary experiments against the three mentioned baselines are promising. The results show that our relational approach outperforms the three baseline approaches and so that even our initial prior rules add a benefit and thus justify the benefit of using a relational approach. We plan to extend our model with the ideas suggested in this paper and perform more extensive experiments that would extend the relational domain of our initial setting.

Table 3: Action point prediction.

Method	Total experiments	Avg. Success	Percentage
Random Classifier	30	16	53.33%
Majority Classifier	30	21.2	70.67%
Decision Tree	30	22.6	75.33%
Relational Approach	30	23	76.67%

5 Conclusion and Future Work

We described an initial approach that uses SRL, and in particular ProbLog, to predict the action a robot needs to perform in order to open doors and the point it needs to action. The experiments showed that our relational approach produced better results than a random classifier, a majority classifier, and a J48 decision tree.

Although the initial model is limited, there are future ideas on adding more context by considering other objects in the environment and extending the action point prediction to consider multiple interest points on the handle which are relationally related. These can be obtained by automatically detecting different grasping points for the handle [11]. Additionally, we plan to include probabilistic priors on the doorframe and handle positions to model real-world detection uncertainty. Furthermore, our model can be extended with a temporal relational aspect to generalise over opening doors that need a sequence of actions (e.g., first push the handle down, then pull the door).

Our final goal is to test the algorithms with a realistic simulator – a common evaluation setup in the robotic community, as well as with real robots. However, this is a rather challenging task, given that even obtaining a dataset for training is difficult and error-prone. However, some solutions exist. One of them is introduced by Saxena et al. [18], where they employ an automated technique to generate a realistic, synthetic dataset. In particular, the use 3D-models of objects together with computer graphics algorithms to construct 2D-images with known ground truth seems feasible and works well in practice. Therefore, as feature work we plan to extend our work with synthetic generated datasets to perform experiments with real-world data. Finally, we will use a realistic simulator to extend our work to multiple robotic hands opening different doors.

Acknowledgements. Bogdan Moldovan is supported by the IWT (agentschap voor Innovatie door Wetenschap en Technologie). Laura Antanas is supported by the European Community's 7th Framework Programme, grant agreement First-MM-248258.

References

1. L. Antanas, M. van Otterlo, J. M. Oramas, T. Tuytelaars, and L. De Raedt. A relational distance-based framework for hierarchical image understanding. In *ICPRAM (2)*, pages 206–218, 2012.
2. D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner. Grasp planning in complex scenes. In *7th IEEE-RAS International Conference on Humanoid Robots*, pages 42–48. IEEE, 2007.
3. M. Blum, J. T. Springenberg, J. Wulfing, and M. Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1298 –1303, May 2012.
4. L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *CVPR*, pages 1729–1736, 2011.
5. L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *Prob. Ind. Log. Progr.*, pages 1–27, 2008.
6. L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007.
7. B. Gutmann, I. Thon, and L. De Raedt. Learning the parameters of probabilistic logic programs from interpretations. In *ECML*, 2011.
8. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1), 2009.

9. A. Kimmig, B. Demoen, L. De Raedt, V. S. Costa, and R. Rocha. On the implementation of the probabilistic logic programming language ProbLog, 2011.
10. N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating Naïve Bayes and FOIL, 2005.
11. P. Moreno, J. Hrnstein, and J. Santos-Victor. Learning to grasp from point clouds. *VisLab Technical Report*, 2011.
12. M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. V. Le, A. Wellman, and Andrew Y. Ng. High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening. In *ICRA*, pages 2816–2822. IEEE, 2009.
13. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.
14. L. De Raedt. *Logical and Relational Learning.* Springer, 2008.
15. L. De Raedt and I. Thon. Probabilistic rule learning. In *ILP*, pages 47–58, 2010.
16. B. Rosman and S. Ramamoorthy. Learning spatial relationships between objects. *The International Journal of Robotics Research*, 30(11):1328–1342, 2011.
17. S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach.* Prentice-Hall, 1995.
18. Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Y. Ng. Robotic grasping of novel objects. In *In Neural Information Processing Systems*, pages 1209–1216, 2006.
19. B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3D range scans taking into account object boundaries. In *ICRA*, pages 2601–2608, 2011.
20. X. Yang, Y. Tian, C. Yi, and A. Arditi. Context-based indoor object detection as an aid to blind persons accessing unfamiliar environments. In *ACM Multimedia*. ACM, 2010.