

# Solving Constraints with Trigonometric Functions Occurring in the Workspace of a Mobile Robot by Methods of Machine Learning

Fritz Wysotzki      Sylvia Wiebrock      Carsten Gips

Methods of Artificial Intelligence, Dept. of Computer Science, Technical University of Berlin  
FR 5-8, Franklinstraße 28/29, D 10587 Berlin, email: wysotzki@cs.tu-berlin.de

## Abstract

*The paper represents first results on solving constraint nets consisting of equations and inequations containing trigonometric functions by methods of Machine Learning. Constraints of this type occur for example in planning movements of a mobile robot on a symbolic level in a workspace where obstacles or other “dangerous regions” have to be avoided. Another application area is the automatic generation of layouts and diagrams from textual input. Learning proceeds by “active” exploration of the workspace by generating training data classified by constraint satisfaction vs. non satisfaction. Results obtained with both decision tree learning and a neuronal net of the Perceptron type demonstrate good approximation and generalization properties depending on the number of constraints and training data.*

## 1 Introduction<sup>1</sup>

In the AI-research on robotics during the seventies there were two main lines: 1. planning of global optimal action sequences on a symbolic (relational) level and 2. the development of “higher level languages” for instruction of robot movement instead of point-to-point-teaching. The problem with 1. was that in order to get executable plans the semantics of the symbolic (e.g. predicate logic) descriptions in the workspace of the robot had to be provided, and with 2., where the semantics of relations was defined by the relative positions and orientations of the objects involved, that complicated constraint satisfaction problems arose (cf. [1, 6]). Since relative positions and orientations are defined by homogeneous transformations (matrices), the semantics of relations includes the satisfaction of con-

straints consisting of equations and inequations which contain trigonometric functions. Additional problems arise in the presence of obstacles or dangerous regions which must be avoided and their transformation in the configuration (motor-) space of the robot. There are no general analytical methods to solve sets of constraints of this kind. Numerical solutions would be adapted to special environments (e.g. special assembly or navigation tasks) only and are therefore of no general interest even if real-time solutions were possible. Despite these problems it would be desirable to program robots on a symbolic textual level. For instance, one could instruct a mobile robot moving in a room by “Go to the cupboard, the cupboard is right of you”. This would roughly constrain the search for the cupboard, which can then be identified accurately by the sensory system of the robot. If we have a set or chain of instructions or even a text describing a more complex scenario, higher level “cognitive skills” like spatial reasoning to infer relations not explicitly mentioned would be necessary. As a main problem there remains constraint satisfaction of the kind mentioned above.

In this paper we describe how methods of concept (classification) learning can be used to solve constraint satisfaction problems in the spatial domain, for example if a set of numerical constraints defines a complicated region in the physical or feature (configuration) space. In this case it is in general not easy or even not possible to find explicit boundary descriptions by some kind of formula manipulation system. Formally, the task is learning a decision function  $A(x_1, \dots, x_n)$  which decides whether a vector  $x = (x_1, \dots, x_n)$  of the configuration space belongs to a region where the predicate  $A$  is true, i.e. the corresponding constraints are satisfied. In the examples given below, the constraints consist of equations and inequations containing trigonometric functions which lead to computational difficulties well known in robotics [1, 6]. Seen from the cognitive point of view, our claim is that at least parts of human knowledge (facts and general rules) are learned by experience and directed exploration using the type of in-

<sup>1</sup>This research was supported by the Deutsche Forschungsgemeinschaft (DFG) in the project “Modelling Inferences in Mental Models” (Wy 20/2–1, Wy 20/2–2) within the priority program on spatial cognition (“Raumkognition”).

ductive inference introduced in this paper. Learning rules (logical implications) of the form  $\forall x(A(x_1, \dots, x_n) \rightarrow B(x_1, \dots, x_n))$  is the more general case which can be handled by our method, too (see Sect. 2.1).

In the following we identify a region where a constraint  $A$  is satisfied with a class (concept)  $A$ . Now algorithms of classification learning (e.g., decision tree learning or Neuronal Nets) by means of a training set construct classifiers which — by inductive generalization — decide the class membership of an arbitrarily chosen  $x$  (not necessarily contained in the training set). This decision is very fast compared to using the system of (in)equations for direct computation of the class membership and therefore especially suited for on-line tasks. Usually, there is a generalization error due to the unavoidable approximation of the boundaries between the  $A$ -region and the not- $A$ -region. The generalization error can be measured using a test set of classified example vectors different from the training set. Decision tree learners approximate the class boundaries piecewise linearly by axis-parallel hyperplanes, generalized Perceptrons by hyperplanes in general position. In the case of rule learning, the implication  $\forall x(A(x) \rightarrow B(x))$  can be “decided” within the limits of approximation error by testing whether every  $x$  taken from the test set and classified as belonging to the  $A$ -region also belongs to the  $B$ -region. This procedure will be demonstrated by means of an example problem taken from the area of spatial reasoning using the decision tree learner Cal5 [3, 4] and the neuronal net Dipol [5] to learn the semantics of spatial relations like *right* or *in front*. In our case the training set is not given by a “teacher” but *must be constructed appropriately by exploiting the given constraint description*. This is what we will call “learning by exploration” or “active learning”. Some aspects of cognitive modelling of exploration under risk, i.e. if there are forbidden regions (for example in physical space or in the configuration space describing a process) which must not be entered, are discussed in Sect. 2.2. Furthermore, we have used the learned decision trees to generate depictions for scenes described by a set of relations. This method is described in Sect. 3.3.

## 2 Learning Spatial Relations with Cal5 and Dipol

The following examples are taken from constraint satisfaction problems arising in a new algorithm for spatial inference [2].

### 2.1 Rule learning, experiments with a bar

In the first example, it has to be determined whether a bar is *right* of a person or robot (indicated by a point  $S$ ). An object is *right* of  $S$  iff every point of it is in the area bounded by the first and second diagonal (see Fig. 1). We define that a bar is in the “ $A$ -region” in the feature (configuration) space defined by the parameters of the problem, iff its corners are *right* of  $S$ . The bar is in the “ $B$ -region” iff any point of it is *right* of  $S$ . Without loss of generality we used for the “ $B$ -region” only the centre of the bar.

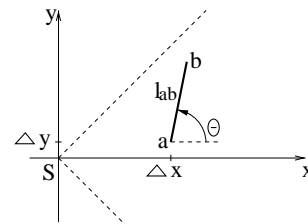


Figure 1: Bar *right* of  $S$

Let  $\Delta_x$  and  $\Delta_y$  be the coordinates of end point  $a$  in the  $S$ -coordinate-system, and  $\Delta_l = (\Delta_x - \Delta_y)/l_{ab}$ ,  $l_{ab}$  being the length of the bar. Then the  $A$ -region is determined by

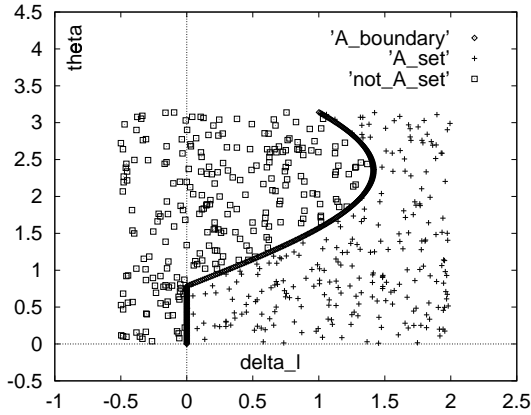
$$\begin{aligned} \Delta_l &> 0 \quad \text{and} \\ \Delta_l &> \sin \theta - \cos \theta \end{aligned}$$

and the  $B$ -region by

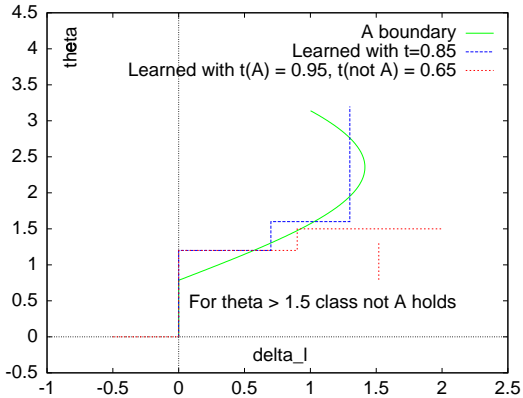
$$\Delta_l > (\sin \theta - \cos \theta)/2$$

In order to get a mapping into a 2-dimensional feature space defined by  $\Delta_l$  and  $\theta$  so that we can depict the class boundaries, we simplified the problem by considering the case  $\Delta_y > 0$  only. With this restriction, in fact the relation “right or behind” bounded by the first diagonal is considered. With the abbreviation  $x = \sin \theta$ , the boundary between the  $A$ -region and not- $A$ -region can be computed in this simple case and is given by  $x_A = \Delta_l/2 \pm \sqrt{1/2 - \Delta_l^2}/4$ , analogous for the  $B$ -region  $x_B = \Delta_l \pm \sqrt{1/2 - \Delta_l^2}$ . We use it for comparison with the *learned boundaries*.

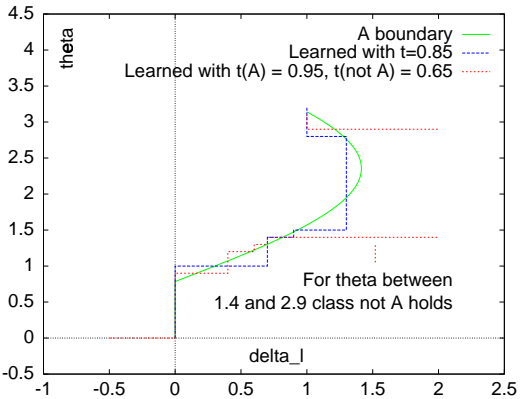
The two parameters  $\theta$  and  $\Delta_l$  span a region of the  $\mathcal{R}^2$ . We generated finite training and test sets of points equally distributed over a subset  $\theta \in [0, \pi]$  and  $\Delta_l \in [-0.5, 2.0]$  of  $\mathcal{R}^2$  and computed the truth values of  $A$  and  $B$  for each vector  $(\theta, \Delta_l)$  of the training and test set, respectively. The decision tree learning algorithm Cal5 constructed classifiers for the  $A$ -region and  $B$ -region using the training sets with a threshold  $t = 0.85$ , i.e. a maximum decision error of 15% was allowed for each class ( $t$  is an input parameter of Cal5 to be specified by the user; for using class



(a)  $A$ -region is right of *computed* boundary



(b) Boundaries for 1000 points



(c) Boundaries for 2000 points

Figure 2: *Learned* vs. *computed* boundaries

dependent threshold values, see Sect. 2.2). The generalization error of both classifiers was measured with the test sets. Finally we used the  $B$ -region classifier to classify the  $A$ -region test set in order to check the implication  $\forall \theta, \Delta_l (A(\theta, \Delta_l) \rightarrow B(\theta, \Delta_l))$ .

Fig. 2(a) shows the  $A$ -region with computed boundary vs. the  $\text{not}_A$ -region. In Fig. 2(b) the solid real (computed) boundary of the  $A$ -region is compared with the dashed learned boundary for a training set of 1000 points. There is a relatively large classification error. In Fig. 2(c) it can be seen that the error shrinks with an increasing number of points for learning. The same holds for the  $B$ -region covering the  $A$  region (not shown in the figures). Furthermore the error for classifying the  $A$ -region test set by the classifier of the  $B$ -region was near 0%. This means we “proved” the implication above inductively.

## 2.2 Exploration under risk

One feature of the concept learning algorithm Cal5 is the ability to decide for a class  $k$  using a class dependent threshold  $t_k$ . Low  $t_k$  for some class  $k$  means that the learning algorithm is “more sensitive” with respect to this class or “pays more attention” to it. Since it can be shown that

$$t_k \approx \text{const}/C_k$$

where  $C_k$  is the cost for not recognizing class  $k$ , the algorithm tries to detect the “important” or dangerous classes (i.e. those with low thresholds) more carefully in order not to overlook one of them. In general this leads to a higher mean error for important classes but to lower costs of misclassification.

We performed experiments with the bar and considered  $\text{not}_A$  to be the important class which must not be overlooked. For demonstration purposes we set the threshold for  $\text{not}_A$  to the very low value of 0.65. Fig. 2(b) and Fig. 2(c) show that the system tries to establish the  $A$ -region with a small error (i.e. if a point is classified as  $A$ , it is indeed in  $A$ ) allowing a higher error for the  $\text{not}_A$  region.

While the overall errors are large for this small number of training data, it is obvious that the computed boundaries change so that the probability of erroneously deciding for  $A$  when a point is in  $\text{not}_A$  is reduced.

Another cognitive aspect connected with our experiments is skill learning if we got the training samples for our learning problem from generating trajectories along which the objects move. Imagine for example a rat getting a negative reinforcement (e.g. a painful electric shock) if it runs into the  $\text{not}_A$ -region. Then it will learn by trial and error and negative reinforcement a decision function for  $A/\text{not}_A$ -discrimination, i.e. implicitly the boundary between the classes (without constructing an analytical description in its head!). The  $\text{not}_A$ -region will then be detected more carefully (for example by sensory features if there are any or by using a memory for orientations and positions of the rat) in order to avoid it. High cost means pain in this case. Another example is the learning of ob-

stacle avoidance by a child or robot. Thus we can easily extend our example discussed above to prototypical cases of skill acquisition where learning takes place in the configuration (motor-) space into which the original 2- or 3D-space together with the obstacles is transformed. Finally let us note that in organismic information processing only approximations of class boundaries can be computed as in our case, i.e. some “fuzziness” is unavoidable. Learning fuzzy relations will be a task of future research.

The results for increasing training sets show that the decision region for  $\text{not}_A$  shrinks with increasing training sets, i.e. the consequences of high risk can be – to some extent – compensated by more experience.

In a students project, we compared Cal5 and Dipol [5], a hybrid (statistical/neural) algorithm that computes, for a given training set and given numbers of clusters for every class, the discriminating hyperplanes for each pair of clusters belonging to different classes. (Dipol automatically decomposes each class into a set of clusters whose number is an input parameter). We applied the programs to the more complicated case of two rectangles, where we get the additional constraint that the objects must not overlap. Some results using Dipol are described below.

### 2.3 Learning “right” for two rectangles using Dipol

The relation was quite simple: one rectangle  $B$  is *right* of another rectangle  $A$  iff it is in the region bounded by the diagonals starting from the two right corners of  $A$  (see Fig. 3). The configuration (feature-) space has seven dimensions. In our problem domain (objects located in a room), we assume that we know the ranges for the dimensions of objects and the maximum distances between them. We also assume that the sizes of objects are not distributed uniformly, i.e. there are more objects of medium size than very large or very small objects. Equally, the distance values will mostly fall into the medium range. To capture this background knowledge, we have given preferred intervals for the values, where the density of points is higher than in the rest of the intervals. For a more detailed description compare [7].

This “natural” distribution of data (called REF in the following) was compared with an approach where only data near the border between the two classes (“critical region”) were used. Using that method (called EPS), only those vectors were considered, where changing one value by  $\varepsilon$  or  $-\varepsilon$  changed the class. Comparing these training sets, the classification errors on a test set (100 000 points distributed according to REF) were much larger for the classifiers learned from a REF training set than from an EPS training set. This means that an extremely good

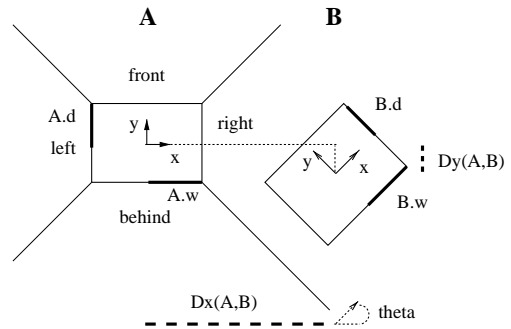


Figure 3: Possible relation *right* for two rectangles

Data gen.	# Clusters		$\varepsilon$	train error	test error
	A	not_A			
1000 points for training					
REF	50	50	/	0.50%	6.64%
EPS	50	50	1.5	1.20%	3.24%
2500 points for training					
REF	50	50	/	1.36%	5.20%
EPS	50	50	1.5	5.28%	2.66%
25000 points for training					
REF	50	50	/	3.04%	3.75%
EPS	50	50	1.5	8.10%	1.93%

Table 1: Results for the *right* relation for rectangles

generalization was obtained with EPS, i.e. by learning in the critical region only. Some sample errors are shown in Tab. 1. The higher errors on the training samples for the EPS method result from the fact that only training data near the boundary are used.

## 3 Foundations for Instruction Chains and Automatic Planning

For any application, to consider relational propositions as isolated entities is insufficient. To be able to build a visualization or to navigate according to a textual instruction, we have to integrate the propositions and their corresponding constraints into a (conceptual) representation of the whole scene. In our approach this “mental model” consists of a labelled graph, where nodes represent objects and directed arcs denote relations between the nodes. For each object we have constraints on the shape and extensions, for each arc we have the constraints for the relative positions and orientations represented by a transformation

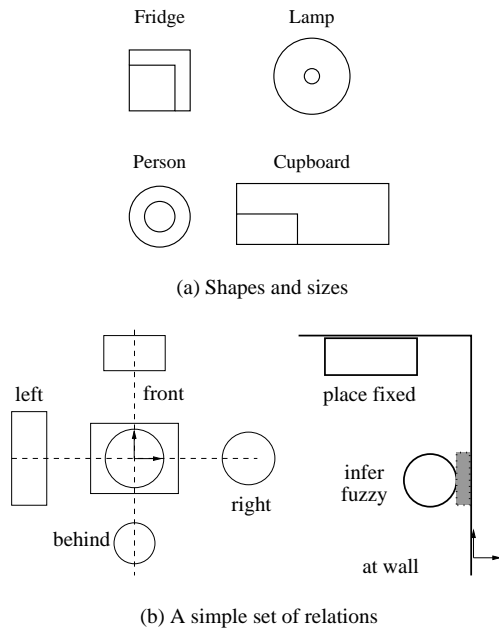


Figure 4: Example for objects and relations

matrix (homogeneous transformation, cf. [1]). Our problem domain is the description of objects in a room. This background knowledge is incorporated in the model.

As a running example for this section consider the very much simplified 2D problem where objects are defined as in Fig. 4(a), where the shape and the minimal and maximal sizes of objects are shown. Persons, fridges, and cupboards have intrinsic front and right sides<sup>2</sup>, while the orientation of the lamp’s coordinate system is arbitrary. The only defined relations are those for the four basic directions, where the center points of the located object must be on the corresponding axis in the coordinate system of the relatum. For the additional relation *at\_wall*, the default is to place an object directly against the wall, but inference of the relation is also possible when the distance is smaller than 0.5. The default orientation for objects standing at the wall is to orient their front sides (if they have an intrinsic front) away from it. The coordinate system for the walls are chosen such that the  $y$ -axes are oriented counterclockwise along the walls, while the positive  $x$ -axes always point outside the room (see Fig. 4(b)).

Suppose we are given the propositions

- (1) *right(S,C)* (“Cupboard  $C$  stands right of Stefanie  $S$ .”)
- (2) *at\_wall(W1,C)* (“Cupboard stands at the wall  $W1$ .”), (1) and (2) are the linearization of “The cupboard stands at the wall to the right of Stefanie.”
- (3) *front(S,F)* (“Fridge  $F$  stands in front of Stefanie.”)
- (4) *at\_wall(W2,F)* (“The fridge stands at the wall  $W2$ .”),

<sup>2</sup>The question of handedness is left out in this example.

- (3) and (4) are the linearization of “The fridge stands at the wall in front of Stefanie.” Thus we know that  $W1 \neq W2$ .
- (5) *left(F,L)* (“The lamp  $L$  stands left of the fridge.”)
- (6) *right(C,L)* (“The lamp stands right of the cupboard.”)

### 3.1 Integrating Spatial Propositions

From the given spatial propositions, the mental model (graph) is built up incrementally (see Fig. 5). Initially, the graph contains only nodes for the room and its four walls. For the first proposition, two new nodes are created. The background knowledge that all objects are in the room constrains the positions of both Stefanie and the cupboard. When the relation *right(S,C)* is added to the graph, the corresponding constraints are propagated and ensure that Stefanie keeps away from the right wall by the size of the cupboard, which conversely cannot stand at the left wall. With the second proposition, only the constraints for the cupboard’s position have to be updated. On the other hand, the proposition *front(S,F)* restricts the positions of the fridge in both directions (because Stefanie can’t be at the right wall, the fridge can’t either, and from the relation it cannot be at the wall behind Stefanie), and also the possible positions of Stefanie and the cupboard. Proposition (5) implicitly asserts (because of the object sizes) that the lamp must stand at the front wall, and proposition (6) places it at the exact position where the left/right axis of the cupboard meets the left/right axis of the fridge.

### 3.2 Inference of Spatial Relations

Due to constraint propagation, the relations between any two objects within the room are implicitly represented in the graph with its constraints. Inferring a relation serves to make this relation explicit. This involves finding a path in the graph – which is always possible, because the graph is connected –, multiplying the transformation matrices along this path, and then checking, for every defined relation, whether the computed transformation matrix satisfies its constraints. Depending on the defined relations, it may be impossible to find such a relation. For instance, in Fig. 5, if the fridge can have a depth larger than 0.5 plus the minimal lamp size, we cannot infer that the lamp stands at the front wall, even if we know that its distance from the wall is exactly the depth of  $F$  minus its radius. That means, we know the relation between the wall and the lamp, but cannot verbalize it. Therefore, in contrast to qualitative reasoning where not-being-able-to-infer-a-relation is equivalent to not knowing their relative positions, in our approach the inference is still useful. For example, if we infer all positions of objects with respect to the room coordinate system, we can generate a depiction

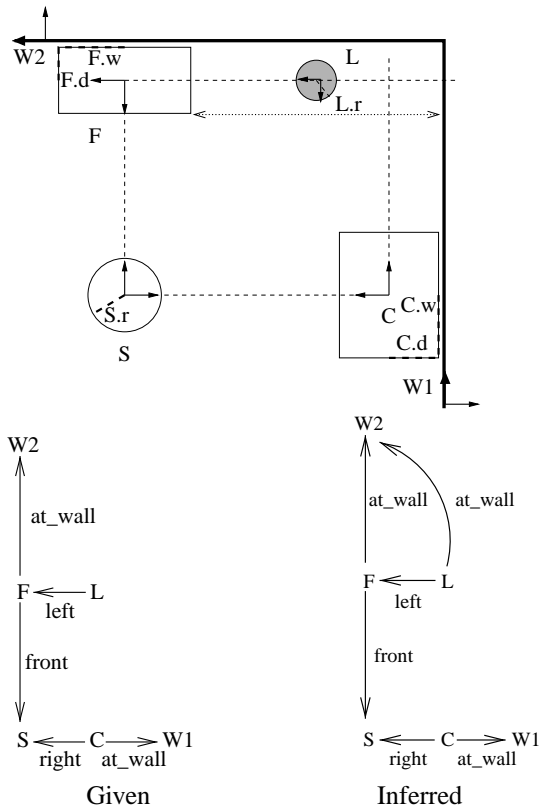


Figure 5: Situation after propositions (1) to (5)

for the scene. In the next section, an indirect approach, using not (only) the computed constraints, but those learned for the relations, is described.

### 3.3 Generating Depictions

One major challenge is the generation of visualizations for a given set of relations. We have implemented an algorithm based on the Cal5 decision trees. This means that for every relation (in this case *right/front/left/behind* with the areas shown in Fig. 3, and *at\_wall* as above), and for every pair of object types (cuboid or cylinder), the relations have to be learned. The resulting decision trees are then “pruned”, i.e. for each leaf in the tree corresponding to a region in the parameter space, where the relation holds, the admissible intervals for every parameter are extracted. This considerably reduces the storage space necessary for the trees. Each region is assigned a weight relative to the number of training points at that leaf. In the algorithm sketched below, these regions are used to generate depictions. All defined relations are binary. Thus, if one of the objects in a relation is already placed in the room, we can pick a leaf of the tree and compute the size, relative position and relative orientation of the second object by assigning values to the

remaining variables.

Since the decision trees only represent the general constraints for two objects of the specified forms, we have to include additional constraints for the depiction: object descriptions may restrict the extensions of objects, during the inference process we may have narrowed the admissible regions for an object, and most of all we have to check that the object is still within the room and doesn’t overlap any other object.

The algorithm works without backtracking. As we cannot guarantee that every scene produced can be augmented in such a way that the next relation(s) also hold, we work on a specified number  $v$  of object constellations in parallel. For every new relation, the vector containing the positions, extensions, and orientations of objects must be updated. In this step, we try up to  $k$  times to find a leaf such that the new object(s) can be placed within the room without overlapping other objects. In the case that both objects are already positioned, we only have to test whether there is a leaf consistent with the relative position already generated.

#### Generating visualization algorithm:

INPUT: number  $v$  of initial vectors and number  $k$  of trials relations  $r$  that have to hold

file containing object descriptions

OUTPUT: up to  $v$  scenes where all relations  $r$  hold

ALGORITHM:

For each relation  $r$ :

- Identify objects and object types
- Find corresponding pruned tree
- For each vector  $v$ :
  - if** both objects placed **then** check whether relation holds
  - else**
    - **if** both objects new **then** place first randomly in room
    - Pick area (random according to weight of leaf)
    - Assign values to variables within intervals<sup>3</sup>
    - Check nonoverlapping with other objects
    - **repeat** up to  $k$  times if check fails
    - **if** no success **then** drop vector  $v$

Generate depictions for remaining vectors

## 4 Conclusions

We have presented an approach to spatial inference with transformation matrices and constraints. While the inference of spatial relations is more complex and less efficient than in the case of qualitative reasoning, we can express

<sup>3</sup>The intervals are the intersection between the information from the decision tree and the specific constraints for the object extensions and/or rotations

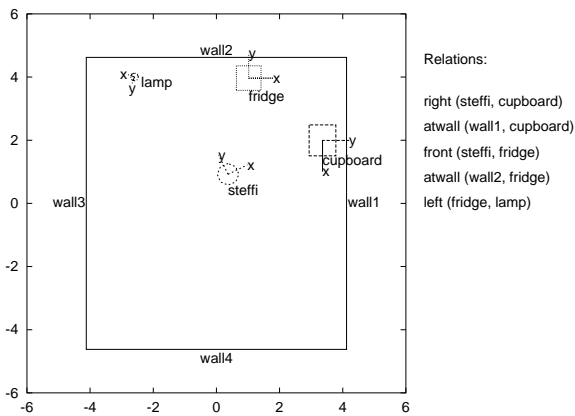


Figure 6: Depiction generated for five relations

more complicated relations. For problem domains where a numerical representation is necessary, as for generating visualizations, our approach is more natural. The inherent problem of checking nonlinear symbolic constraints can often be avoided by the application of background knowledge and by default rules.

For the generation of depictions, we have described a ML based algorithm that uses decision trees for the relations. Several scenes are augmented in parallel with the next relation. Preferred object sizes or positions can be implicitly provided by generating more training data for those values.

## References

- [1] A. P. Ambler and R. J. Popplestone. Inferring the Positions of Bodies from Specified Spatial Relationships. *Artificial Intelligence*, 6:157–174, 1975.
- [2] B. Claus, K. Eyferth, C. Gips, R. Hörnig, U. Schmid, S. Wiebrock, and F. Wysotzki. Reference Frames for Spatial Inferences in Text Comprehension. In Christian Freksa, Christopher Habel, and Karl F. Wender, editors, *Spatial Cognition - An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, LNAI 1404. Springer Verlag, Berlin, 1998.
- [3] W. Müller and F. Wysotzki. Automatic Construction of Decision Trees for Classification. *Annals of Operation Research*, 92:231–247, 1994.
- [4] W. Müller and F. Wysotzki. The Decision-Tree Algorithm CAL5 Based on a Statistical Approach to Its Splitting Algorithm. In G. Nakhaeizadeh and C. C. Taylor, editors, *Machine Learning and Statistics - The Interface*, pages 45–65. Wiley, 1997.
- [5] B. Schulmeister and F. Wysotzki. DIPOL – A Hybrid Piecewise Linear Classifier. In G. Nakhaeizadeh and C. C. Tay-

lor, editors, *Machine Learning and Statistics - The Interface*, pages 133–152. Wiley, 1997.

- [6] R. H. Taylor. A synthesis of manipulator control programs from task level specifications. Technical Report Memo AIM-282, Stanford Artificial Intelligence Laboratory, 1976.
- [7] S. Wiebrock, L. Wittenburg, U. Schmid, and F. Wysotzki. Inference and visualization of spatial relations. In C. Freksa, W. Brauer, C. Habel, and K. Wender, editors, *Spatial Cognition II – Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, volume 1849 of *LNCS*. Springer Verlag, 2000.