

# Constraint Based Belief Modeling

Daniel Göhring, Heinrich Mellmann, and Hans-Dieter Burkhard

Institut für Informatik  
LFG Künstliche Intelligenz  
Humboldt-Universität zu Berlin, Germany  
{goehring,mellmann,hdb}@informatik.hu-berlin.de

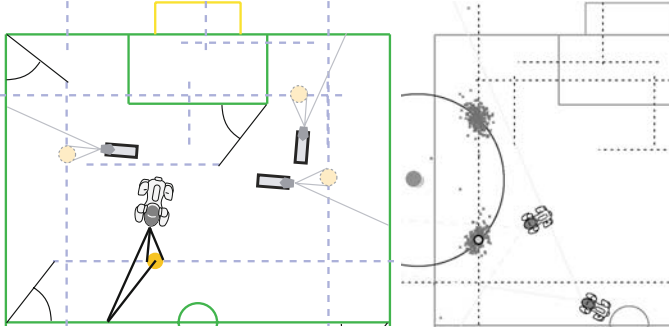
**Abstract.** In this paper we present a novel approach using constraint based techniques for world modeling, i.e. self localization and object modeling. Within the last years, we have seen a reduction of landmarks as beacons, colored goals, within the RoboCup domain. Using other features as line information becomes more important. Using such sensor data is tricky, especially when the resulting position belief is stretched over a larger area. Constraints can overcome this limitations, as they have several advantages: They can represent large distributions and are easy to store and to communicate to other robots. Propagation of a several constraints can be computationally cheap. Even high dimensional belief functions can be used. We will describe a sample implementation and show experimental results.

## 1 Introduction

Self localization and object tracking is crucial for a mobile robot. Especially when sensing capabilities are limited a short term memory about the surrounding is required. Thus modeling techniques have widely been researched in the past. Common approaches use Bayesian algorithms [5] as Kalman [6] or particle filters [2]. Under some circumstances, when sensor data is sparse and computational power is limited, those approaches can show disadvantages. Complex belief functions are hard to represent for Kalman filters which use Gaussians, particle filters don't have this limitation but need a high number for approximating the belief, resulting in high computational needs which often cannot be satisfied. We tackle this problem by using constraints for sensor data and belief representation. Constraint based modeling approaches have been proposed for localization in [8], or for slam map building in [7]. Constraint based approaches have several advantages: a) constraints are easy to create and to store. b) they have a high representational power, c) combining different constraints is computationally cheap. In this paper we discuss constraint propagation methods for solving navigation problems. The main difference to classical propagation is due to the fact that navigation tasks do always have a solution in reality.

### 1.1 Motivation

In many domains landmarks are very sparsely arranged. In RoboCup landmarks like beacons were more and more removed during the last years. Other sensor



**Fig. 1.** Robots are localizing a ball relative to field lines, resulting in large distributions  
a) a robot is seeing a ball next to a line, resulting possible ball positions, b) particle based multi-agent ball localization, as described in [3]

data like field line information has to be used for self localization. We found out that seeing one field line results in a complex belief function which is hard to represent by a Gaussian or by a small set of samples as in Monte-Carlo approaches. Therefore we developed this constraint based representation method.

## 1.2 Outline

We will show how a constraint based localization can be implemented within the RoboCup Legged league. Furthermore we will compare the constraint based approach to a Monte-Carlo Particle Filter. We will use real robot sensor data and will discuss thereby how noisy and inconsistent sensor data can be considered for constraint localization.

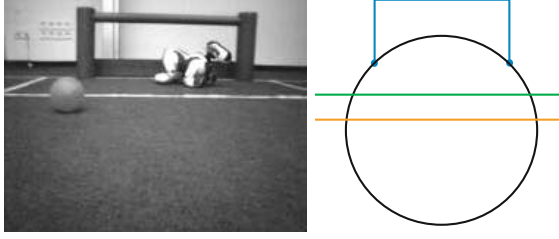
## 2 Perceptual Constraints

A constraint  $C$  is defined over a set of variables  $v(1), v(2), \dots, v(k)$ . It defines the values those variables can take:

$$C \subseteq \text{Dom}(v(1)) \times \dots \times \text{Dom}(v(k))$$

We start with an example from the Legged league where the camera image of a robot shows a goal in front and the ball before the white line of the penalty area (Figure 2). It is not too difficult for a human interpreter to give an estimate for the position  $(x_B, y_B)$  of the ball and the position  $(x_R, y_R)$  of the observing robot. Humans can do that, regarding relations between objects, like the estimated distance  $d_{BR}$  between the robot and the ball, and by their knowledge about the world, like the positions of the goalposts and of the penalty line.

The program of the robot can use the related features using image processing. The distance  $d_{BR}$  can be calculated from the size of the ball in the image, or from the angle of the camera. The distance  $d_{BL}$  between the ball and the penalty line



**Fig. 2.** Example from RoboCup (Four legged league): A robot is seeing a goal and the ball in front of a penalty line. The right picture shows the resulting robot positions represented by the periphery circle according to  $C_1$ , and the line of the Ball-Line-Constraint  $C_2$ .

can be calculated, too. Other values are known parameters of the environment:  $(x_{Gl}, y_{Gl}), (x_{Gr}, y_{Gr})$  are the coordinates of the goalposts, and the penalty line is given as the set of points  $\{(x, b_{PL}) | -a_{PL} \leq x \leq a_{PL}\}$ . The coordinate system has its origins at the center point, the y-axis points to the observed goal.

The relations between the objects can be described by constraints. The following four constraints are obvious by looking to the image, and they can be determined by the program of the observing robot:

$C_1$ : The view angle  $\gamma$  between the goalposts (the distance between them in the image) defines a circle (periphery circle), which contains the goal posts coordinates  $(x_{Gl}, y_{Gl}), (x_{Gr}, y_{Gr})$  and the coordinates  $(x_R, y_R)$  of the robot:

$$\{(x_R, y_R) | \arctan \frac{y_{Gl} - y_R}{x_{Gl} - x_R} - \arctan \frac{y_{Gr} - y_R}{x_{Gr} - x_R} = \gamma\}$$

$C_2$ : The ball lies in the distance  $d_{BL}$  before the penalty line. Therefore, the ball position must be from the set

$$\{(x_B, y_B) | x_B \in [-a_{PL}, a_{PL}], y_B = b_{PL} - d_{BL}\}$$

$C_3$ : The distance  $d_{BR}$  between the robot and the ball defines a circle such that the robot is on that circle around the ball:

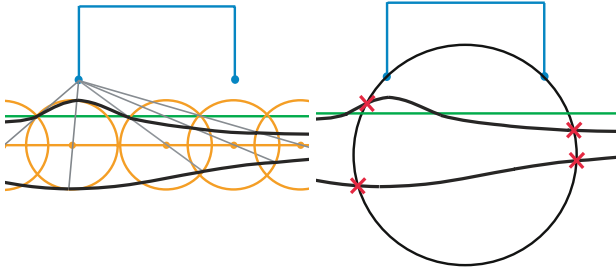
$$(x_B - x_R)^2 + (y_B - y_R)^2 = d_{BR}^2.$$

$C_4$ : The observer, the ball and the left goal post are on a line:

$$\frac{x_R - x_B}{y_R - y_B} = \frac{x_B - x_{Gl}}{y_B - y_{Gl}}$$

The points satisfying the constraints by  $C_1$  (for the robot) and by  $C_2$  (for the ball) can be visualized immediately on the playground as in Figure 2.

The constraint by  $C_3$  does not give any restriction to the position of the ball. The ball may be at any position on the playground, and then the robot has a



**Fig. 3.** Left: The picture shows the Constraint  $C_2$  for the ball, some of the circles according to constraint  $C_5$ , some of the lines according to  $C_4$ , and the resulting two lines for  $C_6$  (bold black lines). Right: Constraints according to  $C_7$ : The position of the robot is one of the four intersection points between the periphery circle ( $C_1$ ) and the lines according to  $C_6$ .

position somewhere on the circle around the ball. Or vice versa for reasons of symmetry: The robot is on any position of the playground, and the ball around him on a circle. In fact, we have four variables which are restricted by  $C_3$  to a subset of a four dimensional space. The same applies to constraint  $C_4$ .

The solution (i.e. the positions) must satisfy all four constraints. We can consider all constraints in the four dimensional space of the variables  $(x_B, y_B, x_R, y_R)$  such that each constraint defines a subset of this space. Then we get the following constraints:

$$C_1 = \left\{ \arctan \frac{y_{Gl} - y_R}{x_{Gl} - x_R} - \arctan \frac{y_{Gr} - y_R}{x_{Gr} - x_R} = \gamma \right\} \quad (1)$$

$$C_2 = \{(x_B \in [-a_{PL}, a_{PL}], y_B = b_{PL} - d_{BL})\} \quad (2)$$

$$C_3 = \{(x_B - x_R)^2 + (y_B - y_R)^2 = d_{BR}^2\} \quad (3)$$

$$C_4 = \left\{ \frac{x_R - x_B}{y_R - y_B} = \frac{x_B - x_{Gl}}{y_B - y_{Gl}} \right\} \quad (4)$$

Thus the possible solutions (as far as determined by  $C_1$  to  $C_4$ ) are given by the intersection  $\bigcap_{1, \dots, 4} C_i$ . According to this fact, we can consider more constraints  $C_5, \dots, C_n$  as far as they do not change this intersection, i.e. as far as  $\bigcap_{1, \dots, n} C_i = \bigcap_{1, \dots, 4} C_i$ . Especially, we can combine some of the given constraints.

By combining  $C_2$  and  $C_3$  we get the constraint  $C_5 = C_2 \cap C_3$  where the ball position is restricted to any position on the penalty line, and the player is located on a circle around the ball. Then, by combining  $C_4$  and  $C_5$  we get the constraint  $C_6 = C_4 \cap C_5$  which restricts the positions of the robot to the two lines shown in Figure 3 (left).

Now intersecting  $C_1$  and  $C_6$  we get the constraint  $C_7$  with four intersection points as shown in Figure 3 (right). According to the original constraints  $C_1$  to  $C_4$ , these four points are determined as possible positions of the robot. The corresponding ball positions are then given by  $C_2$  and  $C_4$ .

To find the real positions, we would need additional constraints from the image, e.g. that the ball lies between the robot and the goal (which removes one of the lines of  $C_6$ ), and that the robot is located on the left site of the field (by exploiting perspective).

### 3 Formal Definitions of Constraints

We define all constraints over the set of all variables  $v(1), v(2), \dots, v(k)$  (even if some of the variables are not affected by a constraint). The domain of a variable  $v$  is denoted by  $Dom(v)$ , and the whole universe under consideration is given by

$$U = Dom(v(1)) \times \dots \times Dom(v(k))$$

For this paper, we will consider all domains  $Dom(v)$  as (may be infinite) intervals of real numbers, i.e.  $U \subseteq \mathbb{R}^k$ .

**Definition 1.** (*Constraints*)

1. A **constraint**  $C$  over  $v(1), \dots, v(k)$  is a subset  $C \subseteq U$ .
2. An assignment  $\beta$  of values to the variables  $v(1), \dots, v(k)$ , i.e.  $\beta \in U$ , is a **solution** of  $C$  iff  $\beta \in C$ .

**Definition 2.** (*Constraint Sets*)

1. A **constraint set**  $\mathcal{C}$  over  $v(1), \dots, v(k)$  is a finite set of constraints over those variables:  $\mathcal{C} = \{C_1, \dots, C_n\}$ .
2. An assignment  $\beta \in U$  is a **solution** of  $\mathcal{C}$  if  $\beta$  is a solution of all  $C \in \mathcal{C}$ , i.e. if  $\beta \in \bigcap \mathcal{C}$ .
3. A constraint set  $\mathcal{C}$  is **inconsistent** if there is no solution, i.e. if  $\bigcap \mathcal{C}$  is empty.

The problem of finding solutions is usually denoted as solving a constraint satisfaction problem (CSP) which is given by a constraint set  $\mathcal{C}$ . By our definition, a solution is a point of the universe  $U$ , i.e. an assignment of values to all variables. For navigation problems it might be possible that only some variables are of interest. This would be the case if we are interested only in the position of the robot in our example above. Nevertheless we had to solve the whole problem to find a solution.

In case of robot navigation, there is always a unique solution of the problem in reality (the positions in the real scene). This has an impact on the interpretation of solutions and inconsistencies of the constraint system (cf. Section 4.1).

The constraints are models of relations (restrictions) between objects in the scene. The information can be derived from sensory data, from communication with other robots, and from knowledge about the world – as in the example from above. Since information may be noisy, the constraints might not be as strict as in the introductory example from Section 2. Instead of a circle we get an annulus for the positions of the robot around the ball according to  $C_3$  in the example. In general, a constraint may concern a subspace of any dimension

(e.g. the whole penalty area, the possible positions of an occluded object, etc.). Moreover, constraints need not to be connected: If there are indistinguishable landmarks, then the distance to such landmarks defines a constraint consisting of several circles. Further constraints are given by velocities: changes of locations are restricted by the direction and speed of objects.

## 4 Algorithms

In principle, many of the problems can be solved by grid based techniques. For each grid cell we can test if constraints are satisfied. This corresponds to some of the known Bayesian techniques including particle filters.

Another alternative are techniques from constraint propagation. We can successively restrict the domains of variables by combining constraints. We will discuss constraint propagation in the following subsection, later we will present experimental results for this approach.

### 4.1 Constraint Propagation

Known techniques (cf. e.g. [1] [4]) for constraint problems produce successively reduced sets leading to a sequence of decreasing restrictions

$$U = D_0 \supseteq D_1 \supseteq D_2, \supseteq \dots$$

Restrictions for numerical constraints are often considered in the form of  $k$ -dimensional intervals  $I = [a, b] := \{x \mid a \leq x \leq b\}$  where  $a, b \in U$  and the  $\leq$ -relation is defined componentwise. The set of all intervals in  $U$  is denoted by  $\mathcal{I}$ . A basic scheme for constraint propagation with

- A constraint set  $\mathcal{C} = \{C_1, \dots, C_n\}$  over variables  $v(1), \dots, v(k)$  with domain  $U = \text{Dom}(v(1)) \times \dots \times \text{Dom}(v(k))$ .
- A selection function  $c : \mathbb{N} \rightarrow \mathcal{C}$  which selects a constraint  $C$  for processing in each step  $i$ .
- A propagation function  $d : 2^U \times \mathcal{C} \rightarrow 2^U$  for constraint propagation which is monotonously decreasing in the first argument:  $d(D, C) \subseteq D$ .
- A stop function  $t : \mathbb{N} \rightarrow \{\text{true}, \text{false}\}$ .

works as follows:

**Definition 3.** (*Basic Scheme for Constraint Propagation, BSCP*)

- *Step(0) Initialization:*  $D_0 := U, i := 1$
- *Step(i) Propagation:*  $D_i := d(D_{i-1}, c(i))$ .
- *If  $t(i) = \text{true}$ : Stop.*
- *Otherwise  $i := i + 1$ , continue with Step(i).*

We call any algorithm which is defined according to this scheme a *BSCP-algorithm*.

The restrictions are used to shrink the search space for possible solutions. If the shrinkage is too strong, possible solutions may be lost. For that, backtracking is allowed in related algorithms.

**Definition 4.** (*Locally consistent propagation function*)

1. A restriction  $D$  is called **locally consistent w.r.t. a constraint  $C$**  if

$$\forall d = [d_1, \dots, d_k] \in D \quad \forall i = [1, \dots, k] \exists d' = [d'_1, \dots, d'_k] \in D \cap C : d_i = d'_i$$

*i.e. if each value of a variable of an assignment from  $D$  can be completed to an assignment in  $D$  which satisfies  $C$ .*

2. A propagation function  $d : 2^U \times \mathcal{C} \rightarrow 2^U$  is **locally consistent** if it holds for all  $D, C$ :  $d(D, C)$  is locally consistent for  $C$ .
3. The **maximal locally consistent** propagation function  $d_{maxlc} : 2^U \times \mathcal{C} \rightarrow 2^U$  is defined by  $d_{maxlc}(D, C) := \text{Max}\{d(D, C) | d \text{ is locally consistent}\}$ .

Since the search for solutions is easier in a more restricted search space (as provided by smaller restrictions  $D_i$ ), constraint propagation is often performed not with  $d_{maxlc}$ , but with more restrictive ones. Backtracking to other restrictions is used if no solution is found.

For localization tasks, the situation is different: We want to have an overview about **all** possible poses. Furthermore, if a classical constraint problem is inconsistent, then the problem has no solution. As already stated, for localization problems always exists a solution in reality (the real poses of the objects under consideration) so we must be careful not to lose solutions.

**Definition 5.** (*Conservative propagation function*)

A propagation function  $d : 2^U \times \mathcal{C} \rightarrow 2^U$  is called **conservative** if  $D \cap C \subseteq d(D, C)$  for all  $D$  and  $C$ .

Note that the maximal locally consistent restriction function  $d_{maxlc}$  is conservative. We have:

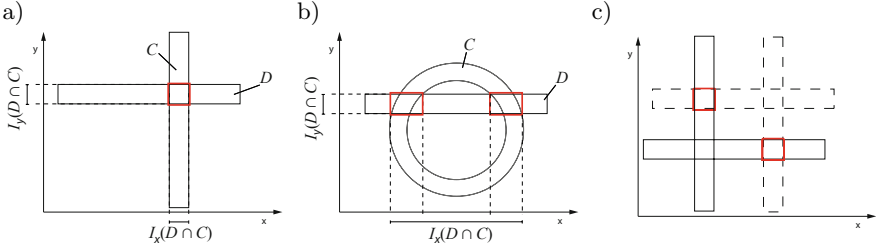
**Proposition 1.** *Let the propagation function  $d$  be conservative.*

1. Then it holds for all restrictions  $D_i : \bigcap \mathcal{C} \subseteq D_i$ .
2. If any restriction  $D_i$  is empty, then there exists no solution, *i.e.*  $\bigcap \mathcal{C} = \emptyset$ .

If no solution can be found, then the constraint set is inconsistent. There exist different strategies to deal with that:

- enlargement of some constraints from  $\mathcal{C}$ ,
- usage of only some constraints from  $\mathcal{C}$ ,
- computation of the best fitting hypothesis according to  $\mathcal{C}$ .

As already mentioned above, n-dimensional intervals are often used for the restrictions  $D$ , since the computations are much easier. Constraints are intersected with intervals, and the smallest bounding interval can be used as a conservative result. Examples are given in Fig. 4.



**Fig. 4.** Constraint propagation with intervals  $D$  for a) two rectangular constraints  $C$  b) a rectangular and a circular constraint  $C$ , resulting in a constraint consisting of two rectangular areas. *Intervals of Projection* w.r.t.  $C \cap D$  are illustrated. c) Two constraints consisting of two boxes each are intersected with each other, resulting constraints depicted as bold red squares.

While local consistency is the traditional approach (to find only some solutions), the approach with conservative intervals is more suited for localization tasks because it can be modified w.r.t. to enlarging constraints during propagation for preventing from inconsistency.

Now we want to present a constraint propagation scheme. The stop condition compares the progress after processing each constraint once.

---

**Algorithm 1.** Constraint Propagation with Minimal Conservative Intervals, MCI-algorithm

---

**Input:** constraint set  $\mathcal{C} = \{C_1, \dots, C_n\}$  with variables  $\mathcal{V} = \{v_1, \dots, v_k\}$  over domain  $U$  and a time bound  $T$

**Data:**  $D \leftarrow U$ ,  $s \leftarrow 1$ ,  $D_{old} \leftarrow \emptyset$

**Result:** minimal conservative  $k$ -dimensional interval  $D$

```

1 while  $s < T$  &  $D \neq D_{old}$  do
2    $D_{old} \leftarrow D$ ;
3   foreach  $C \in \mathcal{C}$  do
4     foreach  $v \in \mathcal{V}$  do
5        $D(v) \leftarrow I_v(D \cap C)$ ;
6     end
7    $D \leftarrow D(v_1) \times \dots \times D(v_n)$ ;
8   end
9    $s \leftarrow s + 1$ ;
10 end

```

---

Looking closer to the possible intersections of constraints (e.g. to the intersection of two circular rings or to the intersection of a circular ring with a rectangle like in Fig. 4a), the sets  $D \cap C$  might be better approximated by sets of intervals instead of a single interval (see Fig. 4 b). Thus, the algorithm was extended for implementation this way: The input and the output for each step are sets



of intervals, and all input intervals are processed in parallel. For such purposes the propagation function  $d$  of the BSCP could be defined over sets as well. As in other constraint propagation algorithms, it might lead to better propagation results if we split a given interval to a union of smaller intervals. In many cases, when using more constraints, the restrictions end up with only one of the related intervals anyway.

*Using Odometry data.* When the robot moves, in self-localization it shifts the constraint boundaries into to movement direction. The odometry noise results in an enlargement of the shifted constraints to pay tribute to slippery ground, collisions and walking noise. The appropriate constraint enlargement was found experimentally.

## 4.2 Handling Inconsistencies

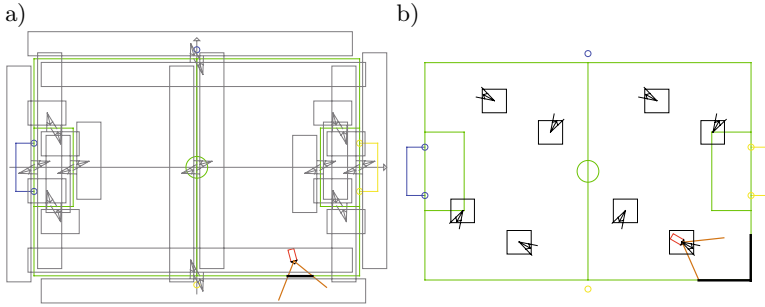
When performing real robot experiments, we realized that many constraints were based on noisy sensor data, resulting to inconsistencies. We dealt with that in the following way: We distinguished sensor data that was consistent to the current belief from inconsistent data, calling it the inconsistent data ratio  $IDR$ . When  $IDR$  was low, updating the belief used consistent sensor data only, resulting in a stable localization. But when new sensor data became inconsistent, e.g. as in kidnapped robot problem, resulting in a very large  $IDR$ , we added that inconsistent data to the belief constraints as well, enabling the robot belief to converge to the new position. For deciding what to do we used a threshold value.

## 5 Experimental Results

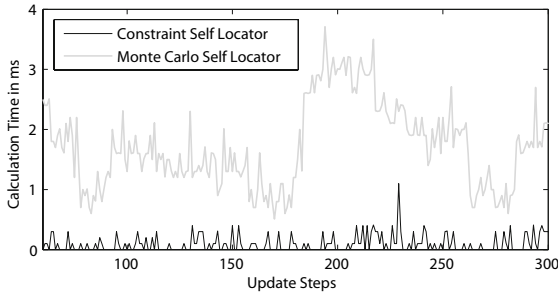
In our experiments within the RoboCup soccer domain (see section 2), we compared an implementation of a Monte-Carlo particle filter (MCPF) with the constraint based algorithm described above. We had our focus on calculation time and on localization accuracy.

We used constraints given by fixed objects like goalposts, flags and field lines identified in the images by the camera of the robot. The creation of the related constraints was done as follows: distances to landmarks are defined by circular rings, where only the distances derived from the vision system of the robot and the standard deviation of the measurement error have to be injected. Constraints given by observed field lines are defined by a set of rectangles and angles (Fig. 5 a)), the distances and the horizontal bearings are sufficient to define these constraints. All this can be done automatically. An example for constraints generated from lines and their propagation is given in Fig. 5 b).

During our first experiments we let a simulated robot move on a predefined path. Then we compared the modeled position with the ground truth position and calculated the localization error. Furthermore we measured in every time step the calculation time. As reference algorithm we used a Monte-Carlo particle filter.



**Fig. 5.** Simulated robot situated on a soccer field. Bold black lines depict the line segment seen by the robot. a) Gray boxes illustrate a constraint generated from only one seen line segment. b) Two constraints are generated from perceived lines (not in the figure), black boxes depict the resulting constraint after propagation of the two constraints.

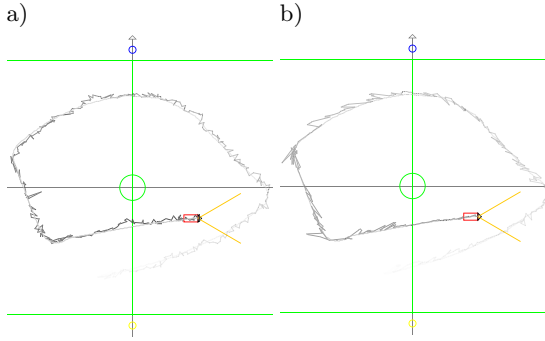


**Fig. 6.** Calculation time for one modeling step on a 1.5 GHz processor. Gray line: Monte Carlo particle filter, using 100 samples. Black line: Calculation time per step using the constraint based algorithm.

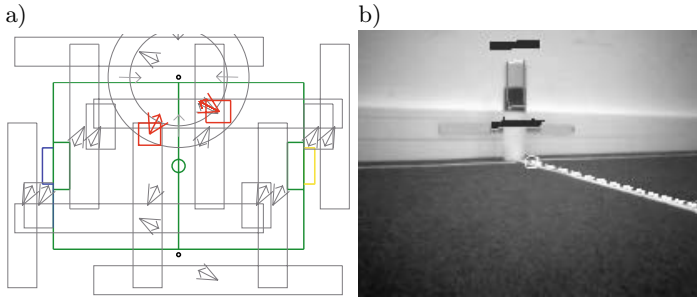
The time measurement data showed that the constraint based algorithm (MCI) algorithm works about 5-10 times faster than the particle filter (see Fig. 6). It also showed that the calculation time for the particle based approach is varying much more than for the constraint based approach.

In a further experiment we measured the localization accuracy for both approaches (Fig. 8). Most of the time the accuracies were comparable. Sometimes the constraint based approach was more sensible to noisy sensor data, which resulted in slightly jumping positions, as Fig. 8 b) shows. In future work we will investigate how the position can be more stabilized over time.

In another experiment we investigated more ambiguous data (i.e. when only few constraints are available as in Fig. 5). In this case, the constraint based approach provided a much better representation of all possible positions (all those positions which are consistent with the vision data). The handling of such cases



**Fig. 7.** Localization accuracy experiment. A robot is walking on the field in a circle (simulated) a) Monte-Carlo Particle filter based localization, the straight reference line is shown as well under the modeled localization trace. b) Constraint based localization.



**Fig. 8.** Real robot experiment: a) The constraints generated from scene b) Recognized flag and line depicted. The two bold rectangles (left) show that image data leaves two possible position areas.

is difficult for particle filters because many particles are necessary for representing large belief distributions. Related situations may appear for sparse sensor data and for the kidnapped robot problem.

## 6 Conclusion

Constraint propagation techniques are an interesting alternative to probabilistic approaches. They could help for better understanding of navigation tasks at all and permit the investigation of larger search spaces employing the constraints between various data. Therewith, the many redundancies provided by images and other sensor data can be exploited better.

We presented an algorithm for constraint propagation and discussed some differences to classical constraint solving techniques. In our experiments, the algorithm outperformed approaches like particle filters with regard to calculational needs. The localization quality of both algorithms was comparable when

many landmarks were seen. In case of fewer landmarks only the constraint based approach was able to represent the resulting belief properly, which was caused by the lack of samples within the particle filter.

Future work will include more investigations on algorithms and further comparisons with existing Bayesian techniques, applications to multiple target tracking with non-unique targets will be searched for as well.

## Acknowledgments

Program code used was developed by the GermanTeam. Source code is available for download at <http://www.germanteam.org>

## References

1. Davis, E.: Constraint propagation with interval labels. *Artificial Intelligence* 32 (1987)
2. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1322–1328. IEEE, Los Alamitos (1999)
3. Göhring, D.: Cooperative object localization using line-based percept communication. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) *RoboCup 2007: Robot Soccer World Cup XI*. LNCS (LNAI), vol. 5001, pp. 53–64. Springer, Heidelberg (2008)
4. Goualard, F., Granvilliers, L.: Controlled propagation in continuous numerical constraint networks. In: *ACM Symposium on Applied Computing* (2005)
5. Gutmann, J.-S., Burgard, W., Fox, D., Konolige, K.: An experimental comparison of localization methods. In: *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Los Alamitos (1998)
6. Kalman, R.: A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering* 82, 35–45 (1960)
7. Olson, E., Leonard, J., Teller, S.: Fast iterative alignment of pose graphs with poor initial estimates. In: *International Conference on Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE* (2006)
8. Stroupe, A., Martin, M., Balch, T.: Distributed sensor fusion for object position estimation by multi-robot systems. In: Bredendfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)* (2001)