Constraint Based World Modeling in Mobile Robotics

Daniel Göhring, Heinrich Mellmann and Hans-Dieter Burkhard

Department of Computer Science
Artificial Intelligence Laboratory
Humboldt-Universität zu Berlin
10099 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 such as beacons or 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 several constraints can be computationally cheap. Even high dimensional belief functions can be used. We will describe a sample implementation and show experimental results.**

I. 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 investigated in the past. Common approaches use Bayesian algorithms [4] 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 do not 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 [10], or for SLAM map building in [8]. A localization method using ultrasonic sensors combined with bounded-error state estimation was introduced for a small truck or vehicle in [7], [9] where interval mathematics described in [5] was applied.

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.

A. Motivation

In many domains landmarks are very sparsely arranged. In RoboCup, landmarks like beacons have been being eliminated gradually during the last years. Other sensor 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.

B. Outline

We will show how a constraint based localization can be implemented within the RoboCup Standard Platform League (SPL). 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.

II. Perceptual Constraints

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

$$C \subseteq Dom(v(1)) \times ... \times Dom(v(k))$$

We start with an example from the SPL where the camera image of a robot shows a goal in front and the ball before the white line of the penalty area (Figure 1). 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 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

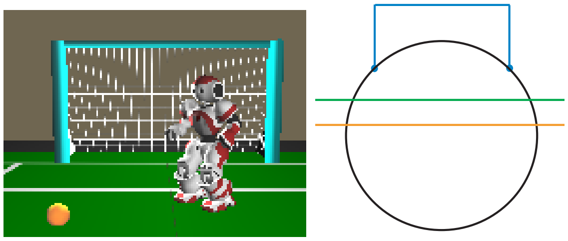

Fig. 1. Image from RoboCup (Standard Platform League): A robot is seeing a goal and the ball in front of a penalty line. The right picture shows the resulting robot positions represented by the periphery circle according to C_1, and the line of the Ball-Line-Constraint C_2. In collaborative navigation, the robot seen in the image could provide further constraints.

looking to the image, and they can be determined by the program of the observing robot:

C_1: The view angle γ 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:

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

C_2: The ball lies in the distance d_{BL} before the penalty line. Thus, 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:

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

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

$$\{(x_R, y_R, x_B, y_B)|\frac{x_R - x_B}{y_R - y_B} = \frac{x_B - x_{Gl}}{y_B - y_{Gl}}\}$$

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 1.

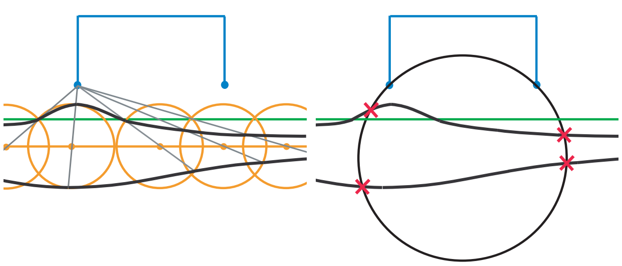

Fig. 2. Left: The picture shows the Constraint C_2 for the ball, some of the circles according to constraint C_5, some of the lines according to C_4, and the resulting two lines for C_6 (bold black lines). Right: Constraints according to C_7: The position of the robot is one of the four intersection points between the periphery circle (C_1) and the lines according to C_6.

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 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:

$$\begin{aligned}
C_1 &= \{(x_R, y_R)|\text{atan}\frac{y_{Gl}-y_R}{x_{Gl}-x_R} - \text{atan}\frac{y_{Gr}-y_R}{x_{Gr}-x_R} = \gamma\} \\
C_2 &= \{(x_B, y_B)|(x_B \in [-a_{PL}, a_{PL}], y_B = b_{PL} - d_{BL}\} \\
C_3 &= \{(x_R, y_R, \ldots)|(x_B - x_R)^2 + (y_B - y_R)^2 = d_{BR}^2\} \\
C_4 &= \{(x_R, y_R, x_B, y_B)|\frac{x_R - x_B}{y_R - y_B} = \frac{x_B - x_{Gl}}{y_B - y_{Gl}}\}
\end{aligned}$$

Thus the possible solutions (as far as determined by C_1 to C_4) are given by the intersection $\bigcap_{1,\ldots,4} C_i$. According to this fact, we can consider more constraints $C_5, \ldots, C_n$ as far as they do not change this intersection, i.e. as far as $\bigcap_{1,\ldots,n} C_i = \bigcap_{1,\ldots,4} C_i$. 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 2 (left).

Now intersecting C_1 and C_6 we get the constraint C_7 with four intersection points as shown in Figure 2 (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).

III. Formal Definitions of Constraints

We define all constraints over the set of all variables $v(1), v(2), ..., v(k)$ (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

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

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 3.1: (Constraints)

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

Definition 3.2: (Constraint Sets)

1) A **constraint set** $\mathcal{C}$ over $v(1), ..., v(k)$ is a finite set of constraints over those variables: $\mathcal{C} = \{C_1, ..., C_n\}$.
2) An assignment $\beta \in U$ is a **solution** of $\mathcal{C}$ if β 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 were 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 the 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 IV-A).

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 II. 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.

IV. 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.

A. Constraint Propagation

Known techniques (cf. e.g. [1] [3]) 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 $I = [a, b] := \{x | a \leq x \leq b\}$ 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

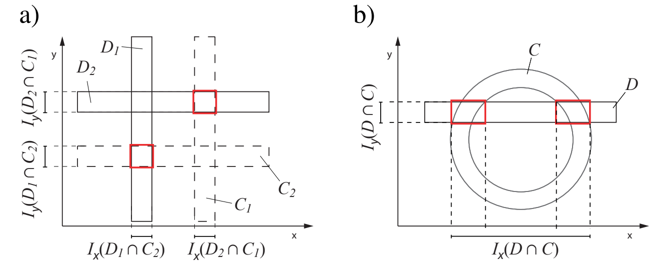

Fig. 3. Constraint propagation with intervals: a) Two constraints consisting of two boxes each $C = C_1 \cup C_2$ and $D = D_1 \cup D_2$ are intersected with each other, resulting constraints depicted as bold red squares. b) a rectangular constraint D and a circular constraint C resulting in a constraint consisting of two rectangular areas. *Intervals of Projection* w.r.t. $C \cap D$ are illustrated.

- A constraint set $\mathcal{C} = \{C_1, ..., C_n\}$ over variables $v(1), ..., v(k)$ with domain $U = Dom(v(1)) \times ... \times Dom(v(k))$.
- 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 $t : \mathbb{N} \rightarrow \{true, false\}$.

works as follows:

Definition 4.1: (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 $t(i) = true$: 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 4.2: (Locally consistent propagation function)

1) A restriction D is called **locally consistent w.r.t. a constraint** C if $\forall d = [d_1, ..., d_k] \in D \quad \forall i = [1, ..., k] \exists d' = [d'_1, ..., d'_k] \in D \cap C \quad : d_i = d'_i$ 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 $d_{maxlc}(D, C) := Max\{d(D, C) | d \text{ is locally consistent}\}$.

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 situations 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 loose solutions.

Definition 4.3: (Conservative propagation function) A propagation function $d : 2^U \times \mathcal{C} \to 2^U$ 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 4.1: 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. 3.

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. 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.

Input: constraint set $\mathcal{C} = \{C_1, ..., C_n\}$ with variables $\mathcal{V} = \{v_1, ..., v_k\}$ 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

while $s < T$ & $D \neq D_{old}$ **do**
  $D_{old} \leftarrow D$;
  foreach $C \in \mathcal{C}$ **do**
    foreach $v \in \mathcal{V}$ **do**
      $D(v) \leftarrow I_v(D \cap C)$;
    end
    $D \leftarrow D(v_1) \times \cdots \times D(v_n)$;
  end
  $s \leftarrow s + 1$;
end

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

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 an rectangle like in Fig. 3a), the sets $D \cap C$ might be better approximated by sets of intervals instead of a single interval (see Fig. 3 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.

a) 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 constraint borders consider slippery ground, collisions, and/or walking noise. The appropriate constraint enlargement has to be found experimentally.

B. Inconsistency Treatment

Noisy robot data, especially from real robots, can lead to inconsistent constraints, i.e., no global solution can be found. There are many possibilities to tackle this problem. At first, we have to consider which kind of constraints we have to propagate with each other. Firstly we have the odometry predicted constraint representing our current belief $\hat{C}_t^B$ at time t. Furthermore we have the constraints generated from sensor data $C_t^{z_1}, ..., C_t^{z_n}$, whereas $z_1, ..., z_n$ depict the different sensor data. Now we have to decide which constraints to propagated with each other - and which constraints to relax. A greedy approach is to propagate the current belief with the sensor data iteratively while the constraint result is not empty.

Input: $\hat{C}_t^B, C_t^{z_1}, ..., C_t^{z_n}$
Result: C_t^B

$C_t^B \leftarrow \hat{C}_t^B$;
for $i = 1$ *to* n **do**
  $S \leftarrow C_t^B \cap C_t^{z_i}$;
  if $S \neq \emptyset$ **then**
    $C_t^B \leftarrow S$;
  end
end
return C_t^B

Algorithm 2: Greedy propagation

The advantage of this approach is its simplicity. The disadvantage is that constraints generated from noisy sensor data can lead to very small constraint sets, and then other constraints might not be taken into account because of inconsistencies. The order of constraints used for propagation here affects the resulting constraint C_t^B. In other words, there can be other consistent subsets when changing the propagation order.

Another approach is to find a maximal subset of the sensor constraints. The sensor constraints are propagated with each other at first and propagated with the belief constraint at second.

Input: $\hat{C}_t^B, C_t^{z_1}, ..., C_t^{z_n}$
Result: C_t^B

$S \leftarrow C_t^{z_1}$;
for $i = 2$ *to* n **do**
  $S \leftarrow S \cap C_t^{z_i}$;
end
if $S \cap \hat{C}_t^B \neq \emptyset$ **then**
  $C_t^B \leftarrow \hat{C}_t^B \cap S$;
else
  $C_t^B \leftarrow increaseBoundaries(\hat{C}_t^B)$
end
return C_t^B

Algorithm 3: Sensor Constraints Propagation

If S and the resulting constraint $\hat{C}_t^B \cap S$ are not empty, they will be propagated resulting in the new belief constraint C_t^B at time t. On the other hand, if the sensor data constraint result S is empty, or has no common elements with $\hat{C}_t^B$, the boundaries of $\hat{C}_t^B$ are increased and C_t^B is assigned to the new constraint. Experimental data showed a much better convergence using this approach.

V. Experimental Results

In our experiments in the RoboCup soccer domain, 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.

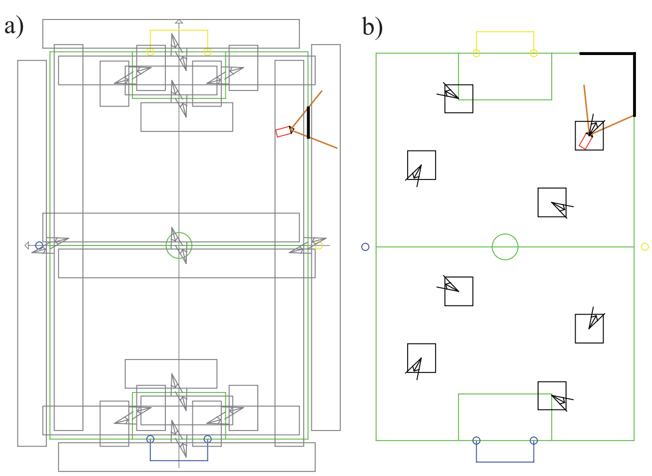

Fig. 4. Robot situated on a soccer field. Bold black lines depict the line segment seen by the robot. a) Gray boxes illustrate a constraint generated from only one seen line segment. b) Two constraints are generated from perceived lines (not in the figure), black boxes depict the resulting constraint after propagation of the two constraints.

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. 4 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 4 b).

During our experiments we let a robot move on a pre-defined 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.

The time measurement data showed that the constraint based algorithm (MCI) algorithm works about 5-10 times faster than the particle filter (see Fig. 5). 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 of both approaches (Fig. 6). Most of the time the accuracies were comparable. Sometimes the constraint based approach was more sensible to noisy sensor data, which resulted in jumping positions, as Fig. 6 b) shows. In future work we will investigate how the position can become more stable over time.

In Fig. 7 we investigated more ambiguous data (i.e. when only few constraints are available as in Fig. 4). 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) than a Monte-Carlo particle filter using 100 samples. The handling of such cases 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.

In Fig. 8 we analyzed, how different constraints affect the localization accuracy, measured as the area covered by the belief constraint (upper row). As percepts served the left and right goal post and the field lines. We further focussed on how consistent and inconsistent sensor constraints (with regard to the belief constraint) affect the localization accuracy. It can be seen, that the covered area of the belief (upper row) decreased whenever consistent

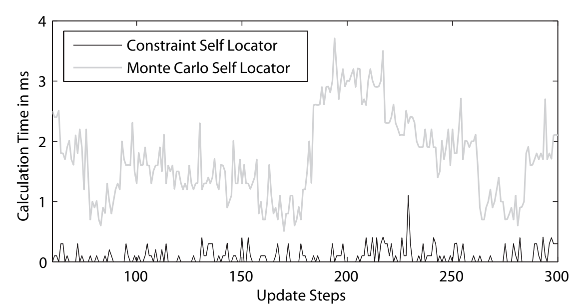

Fig. 5. Calculation time for one modeling step on a 1.5 GHz processor. Gray line: Monte Carlo particle filter, using 100 samples. Black line: Calculation time per step using the constraint based algorithm.

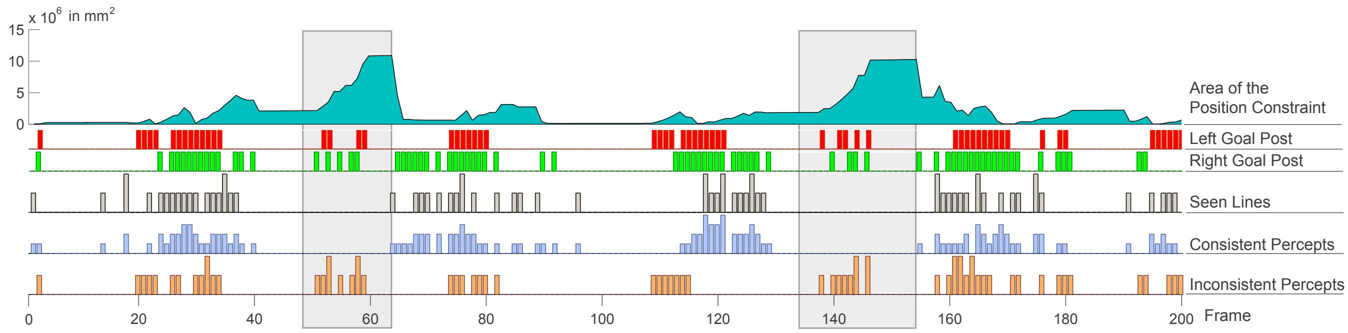

Fig. 8. From the top to the bottom: (1) area of the calculated position constraint as a measure of quality; (2),(3) indicate whether the left or right goal post were seen respectively; (4) denote the number of seen lines; (5),(6) illustrate the overall number of perceptual constraints which were consistent or inconsistent with the position constraint respectively. Gray boxes mark the areas when wrong goal percept were seen. At this time all the seen percepts are inconsistent and the area of position constraint grows until correct data is perceived.

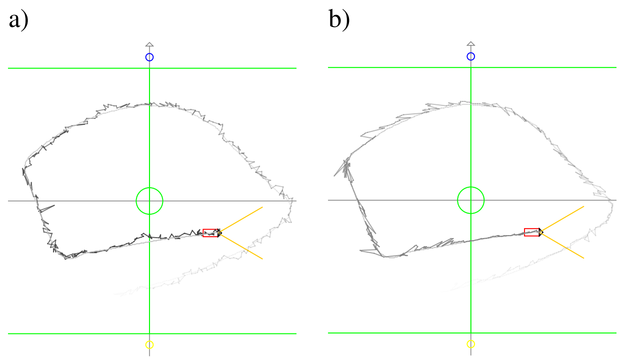

Fig. 6. Localization accuracy experiment. A robot is walking on the field in a circle a) Monte-Carlo Particle filter based localization, the straight reference line is shown as well under the modeled localization trace. b) Constraint based localization.

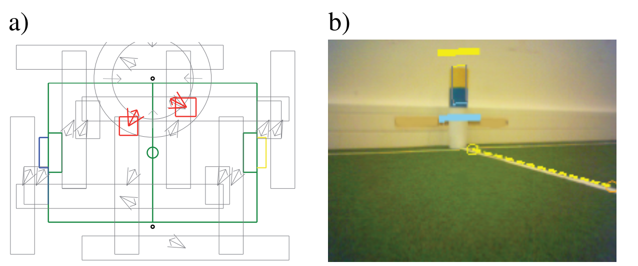

Fig. 7. Real robot experiment: a) The constraints generated from scene b) Recognized flag and line depicted. The two bold rectangles (left) show that image data leaves two possible position areas, because the seen line could match with the center line or with the border line, respectively.

constraints were seen, regardless of inconsistent constraints. When sensor constraints were inconsistent - and whenever consistent constraints were not available, the area increased - thus the localization accuracy decreased. The figure also gives hint about which percepts the robot perceives when scanning across the field while standing on the center circle and it gives an idea about perception breaks, i.e., time slots in which the robot perceives nothing.

VI. Conclusion

Constraint propagation techniques are an interesting alternative to probabilistic approaches. This paper has shown how sensor data can be transformed into constraints. 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 computational needs. Future work will include more investigations on algorithms and further comparisons with existing Bayesian techniques. In addition we want to check how constraint based techniques can be applied to multiple target tracking with non-unique targets.

References

[1] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32, 1987.

[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328. IEEE, 1999.

[3] F. Goualard and L. Granvilliers. Controlled propagation in continuous numerical constraint networks. *ACM Symposium on Applied Computing*, 2005.

[4] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1998.

[5] L. Jaulin, M. Kieffer, Didrit, and E. Walter. *Applied Interval Analysis*. Springer Verlag, London, 2001.

[6] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[7] M. Kieffer, L. Jaulin, Éric Walter, and D. Meizel. Robust autonomous robot localization using interval analysis. *Reliable Computing*, 6(3):337–362, August 2000.

[8] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *International Conference on Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE*, 2006.

[9] E. Seignez, M. Kieffer, A. Lambert, E. Walter, and T. Maurin. Experimental vehicle localization by bounded-error state estimation using interval analysis. In *Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS), pages 1084–1089*, 2005.

[10] A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA-01)*, Lecture Notes in Artificial Intelligence, pages 154–165. Springer, 2001.