In Proceedings of RoboCup International Symposium, Singapore, June 2010.

Utilizing the Structure of Field Lines for Efficient Soccer Robot Localization

Hannes Schulz[⋆], Weichao Liu, Jörg Stückler, Sven Behnke

University of Bonn, Institute for Computer Science VI,
Autonomous Intelligent Systems, Römerstr. 164,
53117 Bonn, Germany

Abstract. The rules in RoboCup soccer more and more discourage a solely color-based orientation on the soccer field. While the field size increases, field boundary markers and goals become smaller and less colorful. For robust game play, robots therefore need to maintain a state and rely on more subtle environmental clues. Field lines are particularly interesting, because they are hardly completely occluded and observing them significantly reduces the number of possible poses on the field.

In this work we present a method for line-based localization. Unlike previous work, our method first recovers a line structure graph from the image. From the graph we can then easily derive features such as lines and corners. Finally, we describe optimizations for efficient use of the derived features in a particle filter. The method described in this paper is used regularly on our humanoid soccer robots.

1 Introduction

On its way to realistic soccer environments, RoboCup started out with small-sized, color-coded scenarios. Gradually, artificial markers, colors and special lighting are removed and the soccer field size increases to encourage teams to build reliable vision systems which can compete under real-world conditions. While other leagues, like the MidSize league, already went a long way, the humanoid league is still at the beginning of this transition. Especially the small available computational power and noisy observations due to mostly unmodelled motion models prevented large steps so far. However, from the experience in MidSize-league we can learn that the removal of colored landmarks emphasizes the importance of field lines. For precise positioning, for example during setup phase of the game, use of lines and crossings is already mandatory since the uncertainties of other landmarks are prohibitively large. Finally, the restricted field of view in humanoid robots – as opposed to "omnivision" commonly used in other leagues – can be partially compensated for using field lines.

In this work we present a system for detecting field-lines and employing them for localization. This is by far not the first method presented for the purpose; however, our approach has several advantages. The line structure is determined using algorithms inspired from work on analysis of handwritten digits [1]. This method employs local and

[⋆] {schulz,liu,stueckler,behnke}@ais.uni-bonn.de

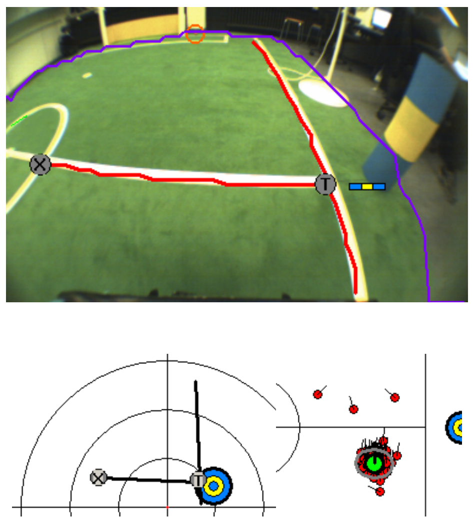

Fig. 1. Localization using line and corner features. The top-figure shows an image taken from the robot's front camera. The purple line denotes the detected field boundary, red (green) lines show field lines (not) used for localization. Detected corners are marked as "X" or "T". Bottom left: egocentric view with everything used for localization. Bottom right: resulting localization using the particle filter.

global cues to determine a graph which captures the essential structure of lines in an image taken by the robot's camera. The graph has a particularly nice structure: nodes are candidates for corners where the field lines meet with the branching factor determining the corner type. Edges in the graph correspond to field lines. Most importantly, the estimates of line parameters are not influenced by spurious segments or noisy locally estimated orientations.

Notably, the favorable properties of the algorithm come at little cost. We incorporated line and corner features into a particle filter which runs online on our humanoid soccer robots. This is possible, because the costly per-particle association decision of observed features to landmarks can be simplified.

The remainder of the paper is organized as follows. The next section reviews previous work on line-based localization. Section 3 and 4 describe preprocessing and feature extraction, respectively. In Section 5 we describe how line and corner features can be used in a particle filter. We provide qualitative and quantitative experimental results using motion capture as ground truth in Section 6.

2 Related Work

Work on field-lines for localization in RoboCup environments can be described on three axes. First, how candidate line segments are found; second, how the line segments are merged to lines; and third, how detected lines are used for localization. Naturally, the literature describes only combinations of all three approaches and this work is no exception. Nevertheless, this work contributes to all three areas.

Finding candidate line segments is usually performed using green-white-green transition filters [2] or scan-lines [3, 4] on the image. However, convolving the image is an expensive operation and scan-lines ignore context. Our method simply makes use of all white pixels and rejects those which are not part of the field lines.

Candidate line segments are then processed to find actual lines. Hough-space techniques are commonly used for this purpose (e.g. [5], [6]). These methods need to calculate a large accumulator array and require double book-keeping of line-pieces for post-processing the found lines. Also, estimated line orientations of small segments tend to be quite noisy. The same is true for the approach followed by the NAO team NUManoid [2], where lines are determined by adding and removing points to candidate lines which requires tuning of numerous parameters. In the approach presented here, candidates for lines emerge naturally from the determined line structure graph.

Finally, candidate line segments or lines can be used for localization. Lauer et. al. [7] use all candidate line segments for performing a gradient descent from the currently estimated pose. This technique relies on the favorable properties of omnidirectional cameras, which are not allowed in the humanoid league, and on stable distance estimates, which are hard to determine for humanoid robots with unknown camera pitch and roll angles. Röfer et. al. [4] also make use of all candidate line segments using a pre-calculated lookup table for the observation model. With this method, segments which are observed on a line can be associated with different lines in the world, supporting improbable poses. In our experience, this approach also requires a large floor and standard deviations in the observation model such that spurious line segments do not completely destroy the belief. For precise localization, this approach is not helpful. Furthermore, the computational load is high if it's used in combination with particle filters, where each particle has to integrate information from all candidate line segments. Consequently, in humanoid and AIBO teams, field line extraction from candidate line segments prevails for localization purposes. The resulting long lines can then either be used as pose constraints [8] or directly used in a particle filter. Pose constraints rule out many possible poses and need to be relaxed iteratively in the case where no allowed poses are left. Particle filters can represent much more complex beliefs. However, to achieve real-time performance, speed optimization is critical. In this work, we describe optimizations used to acquire a high frame-rate even when many features are detected in an image.

Program Part	Time (Standard Deviation) in μs			
Color classification	1578.5	(33.2)		
Pre-processing	421.6	(3.7)		
Object detection	661.2	(94.5)		
Lines preprocessing			128.3	(6.9)
Lines connect nodes			107.0	(30.3)
Split/merge lines			101.5	(35.4)
Verify crossings			35.0	(20.5)
Particle filter	3068.3	(1601.4)		

Table 1. Timing of line-detection code in relation to other visual processing tasks

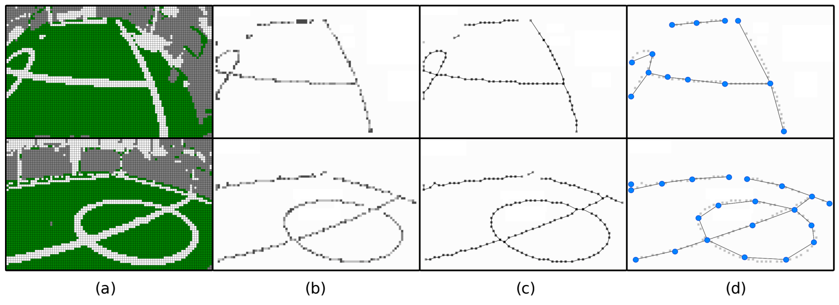

Fig. 2. Visualization of vectorization process. (a) subsampled image of pixels classified as "white" and "green", with cluttered background. (b) skeleton inside field boundary. (c) all nodes with connections. (d) keynode structure graph

3 Vectorization

The first step of our algorithm is the vectorization of the field lines. Our robots are equipped with three color cameras (IDS uEye 1226 LE) which capture images with a WXGA (752×480) resolution in YUV 4:2:2 color space. The individual pixels are classified to color classes as follows. First, the Y-component is compared to luminance thresholds for classification of black and white. For pixels with intermediate luminance, the color class is defined by a lookup-table for the U and V values. Color classes are described by ellipses in the UV plane. In addition, each color class is restricted to an interval in the Y dimension.

The pixels classified to a color class are counted in a separate 94×60 grid (one eighth of the our camera resolution in each dimension). Each pixel in this color class image represents the number of occurrences of its color in a 8×8 window of the original image. While this subsampling reduces spatial resolution, it allows for quick access to the density of the corresponding color in image regions. All algorithms below are performed on these subsampled color class images, albeit we use subpixel precision when possible.

In the next processing step, simple pre-processing operations are performed on the color images. We reduce unwanted effects such as Bayer-pattern induced orange and cyan colors next to sharp contrasts. In unreliable corner-regions of our wide-angle lens we delete all classified colors. In Fig. 2(a), we exemplarily show the result of classification for green and white.

The vectorization of lines is now implemented in two major steps: We first extract the field boundary, and second, determine the structure of lines within the field boundary. The following sections describe the two steps in detail.

3.1 Field Boundary Extraction

In robot soccer games, everything of interest is located on the green carpet. While the area outside the field is undefined and cluttered, objects on the field can be clearly distinguished by color and structure. Consequently, as a first step we segment the field region, thereby further reducing the area of interest.

We use a three-step boundary scanning algorithm. This algorithm first integrates the subsampled color-images into a gray-level image in a manner that it can better represent the field region. Then, it retrieves a histogram of field height in the image. In a final postprocessing step, we smooth and fill local gaps in the histogram.

Merging Color-Images We want to determine the boundary of the green field. However, objects on the field might occlude some parts of the field. Therefore, we create a new 94×60 gray-level image which is composed by a weighted sum of previously classified colors according to their probability of being part of the field or occluding it.

Binarization For a pixel to be part of the field, it must exceed a minimum threshold itself, be part of a largely green line or have enough field pixels below. After merging, we use three thresholds t_{pix}, t_{row} and t_{win} to determine whether a pixel g_{xy} is likely to be inside the field. The threshold tests are arranged in order of complexity to deal with the obvious cases first. First, we check the value of the pixel itself. If $g_{xy} \geq t_{pix}$, the pixel undergoes further examination. If the pre-calculated row-sum $\sum_{x'} \sum_{\{y'=-1,0\}} g_{x',(y+y')}$ is larger than some threshold t_{row}, the pixel is binarized to 1. The row-sum acts here as a prior which biases weak pixels in rows with many high-valued pixels to be part of the field. Pixels which do not pass t_{row} are examined by the most expensive test, by examining the sum s_{xy} of their neighbors below in an 8×4 neighborhood and comparing s_{xy} to t_{win}.

Retrieving Field Image Height Histogram Assuming that the robot should always be located somewhere on the field, the field in the image always starts at the bottom and reaches up to some level. This simplified view can be captured in a histogram of image heights.

We count consecutive binary pixels with value 1 and 0 respectively in each column from bottom-up. Once we encounter more than four consecutive pixels with value 0, the algorithm sets the corresponding bin to be the y-coordinate of the last 1-valued pixel and proceeds to the next column.

Smoothing and Gap Filling The histogram so far is a rough field boundary with gaps and peaks caused by uneven illuminations and imprecise color classification. We consequently smooth the histogram using a 1D Gaussian filter for slightly uneven bins and a median filter for small gaps and peaks.

Finally, we employ a local convex corner algorithm based on [9] to only fill in the remaining gaps but not include unwanted regions like the opponent robot's body or white goal posts attached to the field. This algorithm simply calculates the local convexity:

$$v_{xy} = (R_x - L_x) \cdot (P_y - L_y) - (P_x - L_x) \cdot (R_y - L_y)$$

where P_x is the number of the current bin and P_y is its value; L_x, L_y, R_x and R_y are the respective values of the neighboring left and right bins. If v_{xy} is 0, the top point P_y of the current x is collinear with its left and right neighbors; if v_{xy} is positive, the current bin is above the line segment between the top points of its left and right neighbors and it is below that line segment if v_{xy} is negative. The locally convex hull of the field is then determined by the heights of bins with associated non-negative results. Iterating this scheme will eventually lead to a globally convex hull, but for our purposes one iteration suffices. The result of the field boundary extraction is shown in Fig. 1 for an example image.

3.2 Preprocessing and Skeletonization

With the information of the field boundary, we only process the classified white pixels inside this boundary. Then, a low-pass filter is applied to make sure that each line cross-section has only a single maximum-valued pixel. Skeletonization [1] is used to reduce the line width to approximately one pixel. Unlike morphological methods which start from the border of the line and iteratively erode it, we use a ranking operator, which directly finds a skeleton in the middle of a line. The operator observes 3×3 pixel regions to decide if the central pixel belongs to the skeleton. Pixel having gray-level zero do not belong to the skeleton, but to the background. For all other pixels, the number $c(x, y)$ of neighboring pixels (8-neighborhood) having an equal or higher gray-level is computed and if this number is less than three the central pixel is added to the skeleton. Fig. 2(b) visualizes skeletons resulting from two soccer-field images.

3.3 Placement and Connection of Nodes

Nodes are placed starting at peaks of the skeleton ($c(x, y) = 0$). This emphasizes crossings, which tend to appear as peaks in the skeleton. Note however, that crossings detection does not depend on correct node placement at this stage. Then nodes are placed at least two pixels apart at pixels belonging to ridges ($c(x, y) = 1$ and $c(x, y) = 2$).

The nodes now need to be connected to represent field lines. First, the connection structure of the skeleton is reconstructed by inserting connections where 3×3 regions of nodes overlap or touch on the skeleton. In order to insert the few remaining connections necessary to recover the original field lines, more global information of adjacent connections is used. Lines are stretched by moving end-nodes to the last pixel of the skeleton and degree-0 nodes are split to fill in the gaps. Finally, candidate connections are created and evaluated according to continuity, closure and simplicity. Specifically, the distance between nodes should be small and the grayness on the line between nodes should be similar to grayness of nodes. Furthermore, we place restrictions based on the degree of nodes, ruling out crossings of degree greater than 4 and crossings which do not result in continuous lines. Examples are shown in Fig. 2(c); we refer the interested reader to [1] for details.

4 Feature Extraction

We can now easily extract features such as crossings or lines from the node structure and verify them in the image.

4.1 Line Crossing Detection

The node connections produced so far are the original structures of field lines in the image with many details and distortions. To extract the key structure, the lines are smoothed and nodes are removed at locations of low curvature. Short lines ending in junctions are eliminated and junctions that are close together and connected are merged to form a crossing. When merging and moving nodes, we make sure that the new nodes are placed on the skeleton to guarantee an undistorted world view. The result is shown in Fig. 2(d).

Crossing detection here is now dramatically simplified due to the key node structure representation. A degree-2 node with certain angle of edges connected to it is an L-crossing candidate, a degree-3 node is a T-crossing candidate, and a degree-4 node is an X-crossing candidate. To verify whether one candidate represents a real crossing, we first use a state machine to check out the green-white and white-green color transition along a circle centered at the crossing. Then, we check whether there is a white path from the crossing to the next neighbors in each direction. Both checks are performed in the sub-sampled color-images.

4.2 Field Line Extraction

Starting with the fine-grained, connected graph of observed line segments (Fig. 2(c)) we can extract field lines. Here, a field line is a connected set of nodes with degree two and nodes connected directly to the set with different degree.

The coordinates of the nodes can be approximately connected by a straight line. Consequently, we first traverse the graph to extract connected components of degree two nodes. Because we rely on wide-angle lenses, straight lines in the world do not result in straight lines in the image. Before proceeding, we therefore calculate undistorted coordinates of the nodes. Each component c_i is then processed using the split-and-merge algorithm: we fit a line l_i to c_i using least squares regression [10] on the coordinates of the nodes. The node $n^* = \arg\max_{n \in c_i} \operatorname{dist}(n, l_i)$ with the largest distance to the fitted line defines a splitting point. We split c_i into two components $c_i^{1/2}$ if the node-line distance is sufficiently large and recursively process the resulting components. Components containing less than three nodes are discarded. During the merging phase, we merge components c_i and c_j if the parameters of l_i and l_j are sufficiently similar.

The final field line in image coordinates is determined by projecting the end-points of the component onto the fitted line. We do not use the end-points directly, since these can represent part of another line which barely did not pass the splitting threshold. If c_i contains many nodes, its (possibly wrongly matched) end-points will have limited influence on l_i, resulting in a better approximation. Lastly, by projecting both end-points to the floor, we can infer the line parameters (distance and angle) in a egocentric coordinate frame.

5 Integration of Features

As described in [6], we use Monte-Carlo localization (MCL, [11]) to estimate the current 3D pose of our soccer robots. The pose is a tuple (x, y, θ), where (x, y) denotes the position on the field and θ is the orientation of the robot. The belief is updated recursively with:

$$p(x_t|z_{1:t}, u_{1:t-1}) = \eta \cdot p(z_t|x_t) \cdot \int p(x_t|x_{t-1}, u_{t-1}) \cdot p(x_{t-1}|z_{1:t-1}, u_{0:t-2}) dx_{t-1}, \quad (1)$$

where η is a normalization constant resulting from Bayes' rule, $u_{0:t-1}$ is the sequence of all motion commands executed by the robot up to time $t-1$ and $z_{1:t}$ is the sequence of all observations. The term $p(x_t|x_{t-1}, u_{t-1})$ is called motion model and denotes the probability that the robot ends up in state x_t given it executes the motion command u_{t-1} in state x_{t-1}. The observation model $p(z_t|x_t)$ is the likelihood of making the observation z_t given the robot's current pose is x_t. MCL uses a set of random samples to represent the belief of the robot about its state at time t. Each sample consists of the state vector $x_t^{(i)}$ and a weighting factor $\omega_t^{(i)}$ that is proportional to the likelihood that the robot is in the corresponding state. The update of the belief is carried out according to the sampling importance resampling particle filter. First, the particle states are predicted according to the motion model. For each particle, a new pose is drawn given the executed motion command since the previous update. In the second step, new individual importance weights are assigned to the particles. Particle (i) is weighted according to the likelihood $p(z_t|x_t^{(i)})$. Finally, a new particle set is created by sampling from the old set according to the particle weights. Each particle survives with a probability proportional to its importance weight. This step is also called resampling.

In order to allow for global localization, e.g. in case of the "kidnapped robot problem", a small amount of the particles is replaced by particles with randomly drawn poses. Additional particles are used if pose certainty suddenly drops (Augmented MCL, [12]).

5.1 Crossing Observations

We treat crossing observations similar to other point features on the field (e.g., center of goal, goal posts, marker poles). However, in contrast to the other point features, crossings are not unique. To calculate the likelihood of an observation in the particle filter, we have to make an association decision: for a line crossing observation o, each particle i and all crossings C of the same type (X/T/L), we must calculate the most likely association

$$o' := \arg\max_{c \in C} p\left(c \,\middle|\, x_t^{(i)}\right). \quad (2)$$

While the result of the calculation can be re-used in the second step of the sampling importance resampling, evaluating the observation model repeatedly is expensive and limits the number of particles. However, we can considerably reduce the size of C by considering the egocentric orientation of T and L-crossings. Consider, for example, a particle at the centerline looking towards the yellow goal and observing an L-crossing oriented towards the particle. This particle is quite unlikely to observe the L-crossings next to the blue goal, which are oriented exactly opposite allocentrically. It is also unlikely to see the L-crossings next to the yellow goal which point towards the yellow

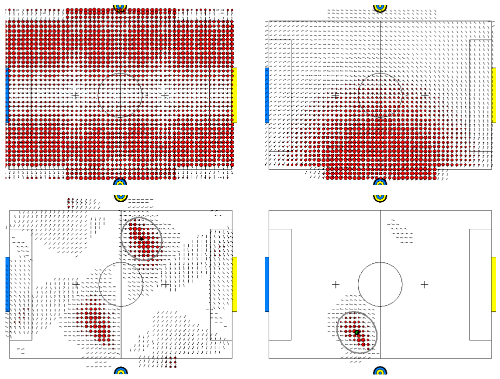

Fig. 3. Particle filter belief visualization based on observed landmarks. We used particles on a 3D-grid and show sizes and orientations based on the likelihood. The features used for localization are the same as in Fig. 1. Top left: only lines, top right: only pole, bottom left: only corners, bottom right: poles, corners and lines.

goal. Consequently, we split the set $\mathcal{L}$ of L-crossings into four sets $\mathcal{L}_{45^\circ}, \ldots, \mathcal{L}_{315^\circ}$ containing two L-crossings of equal global orientation each. A pre-calculated, coarse lookup table $\mathbb{R}^2 \times \{45^\circ, 135^\circ, 225^\circ, 315^\circ\} \mapsto \mathcal{L}$ then associates an observation position on the field and an allocentric, oriented L-crossing observation with the closest L-crossing. We proceed similarly with the T-crossings, but since the distance between different T crossings is large, a lookup table mapping positions to closest T-crossings is sufficient. For X-crossings, we calculate the most likely crossing on a per-particle basis according to Eqn. (2) using the observation model. For an egocentric observation (d_o, β_o) and an expected egocentric position $(d_e^{(i)}, \beta_e^{(i)})$ relative to the current particle i, we define the observation model to be

$$p_{\text{point}}\left(o \,\middle|\, x_t^{(i)}\right) \propto \exp\left(-\frac{\left\|d_e^{(i)} - d_o\right\|^2}{2(\sigma_d + \lambda d_o)^2} - \frac{\left\|\beta_e^{(i)} - \beta_o\right\|^2}{2\sigma_\beta^2}\right), \tag{3}$$

where σ_d and σ_β represent the uncertainty of a range and bearing measurement, respectively. Note that the uncertainty of distance measures increases for far away objects to compensate for the unknown pitch and roll angle of our camera. Fig. 3 (top right) shows the belief resulting from the observation of a point-landmark.

5.2 Line Observations

In our current system, we ignore lines which are short and far away in terms of distance of the line to the origin. For each remaining line o represented by length of dropped perpendicular l_o, distance to closest observed point d_o, and expected angle β_e, we evaluate the observation model

$$p_{\text{line}}\left(o \,\middle|\, x_t^{(i)}\right) \propto \exp\left(-\frac{d^2\left(l_e^{(i)}, l_o\right)}{2(\sigma_l + \lambda d_o)^2} - \frac{\left\|\beta_e^{(i)} - \beta_o\right\|^2}{2\sigma_\beta^2}\right). \tag{4}$$

Here, $d(\,\cdot\,,\,\cdot\,)$ depends on the oriented distance: in contrast to point landmarks discussed above, the orientation β_o represents the angle of the observed line, not the angle of a polar coordinate. As a result a simple observation model which does not take into account oriented distance, would assign equal likelihood to the situation where the robot observes the line behind itself and in front of itself, although the position of the line in front of or behind the robot can be directly inferred from the observation. We therefore set in Eqn. (4)

$$d(l_e, l_o) = \begin{cases} \|l_e - l_o\|_2 & \text{if } \langle l_e, l_o \rangle > 0 \\ \infty & \text{else,} \end{cases} \tag{5}$$

which eliminates high likelihood of the implausible situation. In contrast to corner observations, we cannot make a distinction between the seven long lines in our world. We use Eqn. (2) to determine the most likely match on a per-particle basis. Note, however, that due to monotonicity of $\exp$, for the $\arg\max$ computation in Eqn. (2) it is not necessary to evaluate the likelihood in Eqn. (5) completely. Instead, we apply $\exp(\,\cdot\,)$ after the association has been made and rely on the minimum argument of $\exp$ for the association decision itself. In Fig. 3 (top left) we visualize the belief resulting from the observation of the two lines in Fig. 1.

5.3 Combined Observation Model

For each observation o_j, we ensure that its observation likelihood is larger than some uniform threshold. We further incorporate confidence values c_j from our vision system such that unconfident observations o_j have less influence on the final distribution than confident observations in the same frame:

$$p\left(o_j \,\middle|\, x_t^{(i)}\right) = \alpha_{\text{uni}} p_{\text{uni}}\left(o_j | x_t^{(i)}\right) + \alpha_{\text{normal}} p\left(o_j | x_t^{(i)}\right),$$

where $\alpha_{\text{uni}} = \alpha_{\text{base}} + (1 - \alpha_{\text{base}}) \cdot (1 - c_j)$ and $\alpha_{\text{base}} \in \,]0, 1[$ is a uniform floor. We further set $\alpha_{\text{uni}} + \alpha_{\text{normal}} = 1$ and $p_{\text{uni}}(o_j | x_t^{(i)})$ to be the Lebesgue measure of the observation range. Assuming independence of observations, as typically done in MCL, the observation likelihood of a particle then amounts to the product of all single observations

$$p_{\text{comb}}\left(z_t \,\middle|\, x_t^{(i)}\right) \propto \prod_{l \in \mathcal{L}} p_{\text{line}}\left(l \,\middle|\, x_t^{(i)}\right) \prod_{o \in \mathcal{P}} p_{\text{point}}\left(o \,\middle|\, x_t^{(i)}\right), \tag{6}$$

where $\mathcal{P}$ includes corners and other point landmarks such as goal centers and poles marking the end of the center line. The combined belief resulting from observations of point and line landmarks is shown in Fig. 3 (bottom right).

6 Results

Our 1.3 GHz system runs at about 24 frames per second using between 250 and 1000 particles (depending on the certainty of the current pose). Tab. 1 shows the relative timing results. Soccer robots profit enormously from line-based localization. Consider the pose certainty of the robot in our running example. We illustrate the beliefs in Fig. 3 by

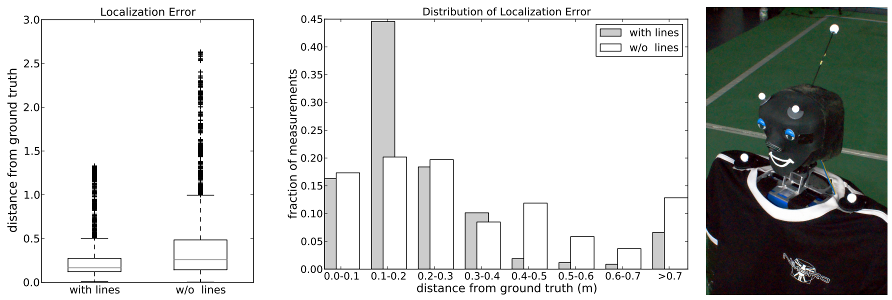

Fig. 4. Left: Localization error with lines (without lines). The median accuracy is $17\ \mathrm{cm}$ ($26\ \mathrm{cm}$). Center: Distribution of distances between estimated and ground truth pose. Right: TeenSize Robot "Dynaped" with attached infrared-reflective markers.

equally distributing particles in a regular 3D-grid. Besides lines and corners, the robot only observes a pole at the side of the field. With color-based localization only, the position on the field is not at all clear from the image. Using only corners, the belief is reduced to two positions. The only colored landmark observed is then enough to further reduce the pose belief to an almost unimodal distribution. For a given frame, we run a particle filter update and select the most likely orientation z for each planar position p_{xy}. We then show this particle sized in proportion to its likelihood at position (x, y).

To further quantify the performance of our algorithm, we let our TeenSize robot "Dynaped" walk for about 5 minutes across the field using remote-control. Robot and field are configured to be rule conformant to the rules of 2010, namely, the robot only uses a single camera, only the posts of the goal are colored and center line pole sizes are reduced. We fix the movable head of the robot to look straight at all times to eliminate effects due to active vision. In addition to estimated poses we record ground truth positions using an Optitrack motion capture system and infrared-reflective markers attached to the robot's torso and head. The experiment was performed twice, once using line and crossing detection and once without, with similar trajectories. Both trajectories contain about 10000 poses. The average estimated speed was $20\ \mathrm{cm/s}$ (with lines) and $22\ \mathrm{cm/s}$ (without lines); we removed 10% (7%) of the recorded frames where the motion capturing tracking failed. We then determined the percentage of recorded frames for which the distance between the robot's estimated pose to ground truth was in some interval. The result is depicted in Fig. 4. Note that without lines only 66% of the recorded poses are within $40\ \mathrm{cm}$ of the ground truth while with line and crossing detection enabled 89% are in this range. With lines, the median localization error was $17\ \mathrm{cm}$, which is less than the average step length. This error is also within the range of the robot's trunk movements which are due to walking-based weight shifts, which can be seen as an error of the motion capturing process.

In the TeenSize league, robots play on a field of the same size as the KidSize field, but the robots are – by definition – much larger. The structural information which is visible from the perspective of a TeenSize robot is consequently very informative. Our TeenSize robot Dynaped, the champion of the Graz 2009 world cup could therefore mainly rely on allocentric information from the particle filter for its decisions instead of previously used egocentric information.

7 Conclusion

In this work, we introduced a line-detection method based on methods developed for the identification of structure in handwritten digits. We first described how to find the boundary of the soccer field. White points within this region are then skeletonized and a simple graph structure is retrieved from the skeleton. The graph structure has advantageous properties: field-line corner candidates are represented by nodes and field-lines are represented by edges. This graph structure is then used to verify linear components and crossings and to determine their parameters. Due to the graph representation, clustering or averaging of noisy locally estimated line parameters can be avoided. Finally, we showed how lines and crossings can be used in Monte-Carlo localization. The paper describes observation models and optimizations to speed up association decisions. Our observation models were exemplarily demonstrated using visualizations of resulting particle filter beliefs. Evaluation of our localization on the real robot showed smaller differences to ground truth when lines and corners were used. The algorithms described were used on our KidSize and (winning) TeenSize soccer robots during RoboCup 2009.

Acknowledgements: This project is supported by the German Research Foundation (DFG), grant BE2556/2-2 and BE2556/4.

References

1. S. Behnke, M. Pfister, and R. Rojas, “Recognition of handwritten digits using structural information,” in *Proc. of ICNN*, 1997.
2. N. Henderson, P. Nicklin, A. Wong, J. Kulk, K. Chalup, R. King, H. Middleton, S. Tang, and A. Buckley, “The 2008 NUManoids Team Report.”
3. M. Sridharan and P. Stone, “Real-time vision on a mobile robot platform,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005)*, 2005, pp. 2148–2153.
4. T. Rofer and M. Jungel, “Fast and robust edge-based localization in the sony four-legged robot league,” *Lecture Notes in Computer Science*, vol. 3020, pp. 262–273, 2004.
5. A. Bais, R. Sablatnig, and G. Novak, “Line-based landmark recognition for self-localization of soccer robots,” in *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on*, 2005, pp. 132–137.
6. H. Strasdat, M. Bennewitz, and S. Behnke, “Multi-cue localization for soccer playing humanoid robots,” *Lecture Notes in Computer Science*, vol. 4434, p. 245, 2007.
7. M. Lauer, S. Lange, and M. Riedmiller, “Calculating the perfect match: an efficient and accurate approach for robot self-localization,” *Lecture Notes in Computer Science*, vol. 4020, p. 142, 2006.
8. A. Borisov, A. Ferdowsizadeh, D. Gohring, H. Mellmann, M. El Bagoury, F. Kohler, O. Welter, and Y. Xu, “NAO-Team Humboldt 2009.”
9. R. Graham, “An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set,” in *Information Processing Letters*. Elsevier, 1972, pp. 132–133.
10. J. Kenney and E. Keeping, *Mathematics of Statistics, Linear Regression and Correlation*. Van Nostrand, 1962, ch. 15.
11. F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” in *Proc. of ICRA*, 1999, pp. 1322–1328.
12. S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.