Entropy-Based Active Vision for a Humanoid Soccer Robot

Andreas Seekircher[1], Tim Laue[2], and Thomas Röfer[2]

[1] University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146 USA
aseek@mail.cs.miami.edu
[2] Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de

Abstract. In this paper, we show how the estimation of a robot's world model can be improved by actively sensing the environment through considering the current world state estimate through minimizing the entropy of an underlying particle distribution. Being originally computationally expensive, this approach is optimized to become executable in real-time on a robot with limited resources. We demonstrate the approach on a humanoid robot, performing self-localization and ball tracking on a RoboCup soccer field.

1 Introduction

Many robot localizations use passive vision systems. Path planning and navigation tasks are often based on a single given robot pose. While acting in an environment to achieve a goal, the measurements used for localization are made independently from the current belief in the state estimation. However, there are several approaches showing that an active vision can be used to reduce uncertainties in localizations. An active vision system controls the robot's actuators or sets sensor parameters to receive measurements that provide as much information as possible. Active localization is often divided into active navigation and active sensing. In active navigation even the target position of the robot is chosen to minimize uncertainties. In this paper an active sensing system is described that controls the head of a humanoid robot and thereby optimizes the state estimation. The focus of the work presented here is on computational efficiency. i. e. to spend only a few milliseconds of the computation time on an embedded system to implement the active sensing, leaving enough computational resources for the other modules running on the robot that, in our case, realize playing soccer.

This paper is organized as follows: The humanoid robot platform and its environment are shortly described in Sect. 2. The active vision approach is presented in Sect. 3, enhancements regarding its computational efficiency are presented in Sect. 4. Section 5 presents the experiments that have been carried out together with the results achieved.

J. Ruiz-del-Solar, E. Chown, and P.G. Ploeger (Eds.): RoboCup 2010, LNAI 6556, pp. 1–12, 2011.

1.1 Related Work

There are many approaches using mutual information. Fox, Burgard, and Thrun realize active Markov localization by an entropy-minimizing action selection [1,3]. Denzler and Brown use a similar method for choosing camera parameters (pan, tilt, and zoom) for object recognition [2]. Vidal-Calleja et al. use an active localization for SLAM [16]. However, active vision systems have also been used in other domains, not only for localization. For instance, Schill et al. use active sensing for the exploration of spatial scenes [12]. Except for the latter, all approaches are using active navigation and active sensing, because the localization controls all robot actions. However, in some cases the robot has to perform a global task and the localization can only be improved by active sensing. Seara and Schmidt use an entropy-based method for gaze control to optimize simultaneous localization and obstacle detection [13]. A different approach by Zhou and Sakane uses Bayesian networks to represent relationships between the environment, control actions, and state estimates [17].

There are also task-oriented approaches to learn camera control strategies for robots playing soccer. In [7] the camera control for shooting goals is learned by reinforcement learning using the success rate for rewards. The camera control of a goalkeeper is learned in [5].

2 Robot Platform and Environment

Our experimental platform is the RoboCup edition of the Nao robot [4] made by Aldebaran Robotics as shown in Fig. 1. The robot is equipped with 21 degrees of freedom, a 500 MHz processor, and a camera as main sensor[1]. In addition, it provides measurements of joint angles which are combined with accelerometer and gyroscope data for body posture estimation. Image processing and self-localization are performed at the camera's frame rate, i. e. 30 Hz.

The experimental environment is a RoboCup Standard Platform League field. The field size is $7.4m \times 5.4m$. The only unique features are two colored goals and one center circle. In addition, the field contains several different lines which are non-unique.

The robot runs the software framework of the RoboCup team *B-Human* [11] in the same configuration as it is used during real competitions, e. g. during the team's win of the RoboCup German Open 2009 and the RoboCup 2009. The self-localization component is an improved version of the one described in [8] that is based on [10], additionally including the *Augmented MCL* resampling approach of [6]. The robot's vision software is able to perceive a variety of features on the field. In addition to the aforementioned ones, also field line crossings, a ball, and single non-unique goal elements can be extracted from images and be used by the self-localization component.

[1] In fact, the robot has two cameras but only the lower one has been used for our purposes.

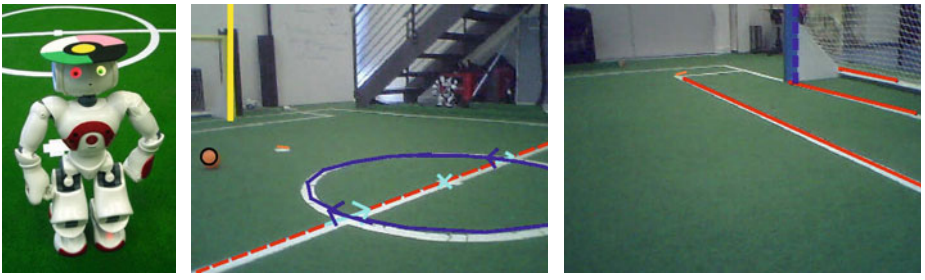

Fig. 1. Left: Nao RoboCup robot used for the experiments. A colored pattern for global tracking has been attached to its head. Middle and right: the robot's perceptions. Images taken by the robot's camera, perceived objects – a yellow right goal post, a blue goal post (side unknown), the center circle, the ball, and several lines – are labeled.

3 Active Vision

Self-localization and ball tracking is based on the perceptions retrieved from camera images. The result of the state estimation is highly dependent on the number and type of the visible features. The perceptions extracted from camera images give information about the state of the robot and the ball. By calculating particle weightings from observations, the uncertainty in the belief distribution is reduced. In many cases moving the camera in a fixed pattern does not provide as much information as possible. A camera control system choosing the pointing direction that gives the highest information gain should improve the state estimation.

3.1 Information and Uncertainty

The uncertainty of a belief b about a state x is given by the Shannon entropy:

$$H_b(x) = -\int b(x) \log b(x) dx \tag{1}$$

The information gain (the mutual information) of an action a is equal to the decrease in uncertainty and it can be calculated by the difference between the entropies in cycles:

$$I(x, a) = H_b(x) - H_b(x'|x, a) \tag{2}$$

In our system, state estimation is realized using particle filters. Therefore the belief is represented as a set of samples. These samples only approximate the actual probability density function. The entropy cannot be calculated directly and has to be estimated. There are many approaches for entropy estimation, for example maximum-likelihood or m-spacing estimators. One of these methods is to create partitions in the state space and to calculate the discrete entropy over the sum of particle weightings (probability mass) in each partition (e. g. as

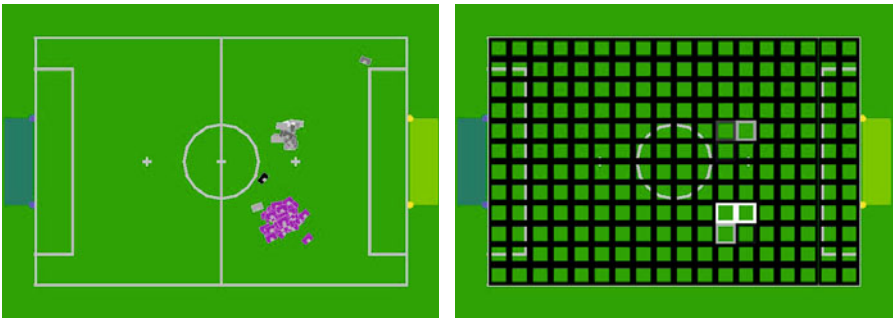

Fig. 2. An example for a belief with samples and the grid used for entropy estimation. The grid dimensions are $(18, 12, 16)$ on the x, y, and rotational axes (the latter is not shown).

in [14]). Similar to that method, simple histogram-based entropy estimation is used in this paper. The belief is approximated by using a grid on the x, y, and rotation axes (cf. Fig. 2). The discrete entropy over all cells can be calculated very efficiently, because the number of cells with a probability greater than 0 is limited by the number of samples. Therefore most cells have no probability and can be ignored. The entropy estimation over a constant grid gives comparatively poor results with a high bias. Nevertheless it is used here because of the limited computational resources.

3.2 Entropy-Based Camera Control

The camera control has to choose the direction with the highest mutual information expected. Thus the entropy expected after executing a given action a in state x is needed for an entropy-minimizing action selection. Equation 3 gives the entropy expected in an probabilistic environment (e. g. used for active localization in [1]). A set of observations is denoted as z. So $p(z|x)$ gives the probability of perceiving z in the state x. The probability of reaching the state x' by executing the action a in state x is written as $p(x'|a, x)$.

$$\begin{aligned} H_b(x'|a) &\approx E_z[H_b(x'|z, a)] \\ &= \int\int\int H_b(x'|z, a)p(z|x')p(x'|a, x)b(x)dzdx'dx \end{aligned} \tag{3}$$

An action selection based on minimizing the entropy only for the immediate next step only produces a greedy exploration. The camera movement created in this way is not the optimal strategy. Due to local maxima, a sequence of other actions could be necessary to produce a higher information gain. However, planning the camera control for more than a single step will result in much higher computational costs due to exponential complexity. Therefore, a camera control system optimizing the information gain only for a single step can be calculated more frequently. Thus, such a control system is more reactive and

will possibly perform better in a dynamic environment. The camera control has to react to unexpected changes in the belief, for example when the robot's position is changed by collisions.

The policy for controlling the camera is given by equation 4. The action with the lowest entropy expected and lowest costs is selected. The costs for each action a in the state x is given by $r(x, a)$ as negative values. The weighting between costs and the entropy expected is given by α.

$$\begin{aligned} \pi(b) &= \underset{a}{\operatorname{argmax}}\ \alpha(H_p(x) - E_z[H_b(x'|z,a)]) + \int r(x,a)b(x)dx \\ &= \underset{a}{\operatorname{argmax}} \int r(x,a)b(x)dx - \alpha(E_z[H_b(x'|z,a)]) \end{aligned} \tag{4}$$

An implementation of this strategy is the Monte Carlo exploration in algorithm 1:

Algorithm 1. Monte carlo exploration (described in [15])

```
1: Monte_Carlo_Exploration(b):
2: set p_a = 0 for all actions a
3: for i = 1 to N do
4:    sample x ~ b(x)
5:    for all control actions a do
6:       sample x' ~ p(x'|a, x)
7:       sample z ~ p(z|x')
8:       b' = Bayes_filter(b, z, a)
9:       p_a = p_a + r(x, a) − αH_b'(x')
10:   end for
11: end for
12: return argmax_a p_a
```

By sampling from the various probability distributions, the method approximates the expected entropy for $N \to \infty$. As the belief is already represented as samples, line 4 only means to choose one sample of the particle filter. In fact a state x consists of the belief and the head joint angles given by sensor readings. Drawing samples from $p(x'|a, x)$ in line 6 can be simplified by the assumption that the state change from x to x' is deterministic. The actions available do only affect the head movement. Many approaches for active localization have to consider uncertainty and the probability of collisions when determining the next state x' for an action a. However by limiting the actions to head movements, the next state x' consist of the same belief for the poses only with changed head joint angles. The state x' is given by the action a.

The last probability function needed for calculating the expected entropy is $p(z|x')$ in line 7. There, the expected observations for a given state x' are created to be used as measurements in the state estimation in line 8. In this paper, all particle weightings are calculated using the expected observation z. These weightings produce an information gain by changing the entropy calculated in line 9.

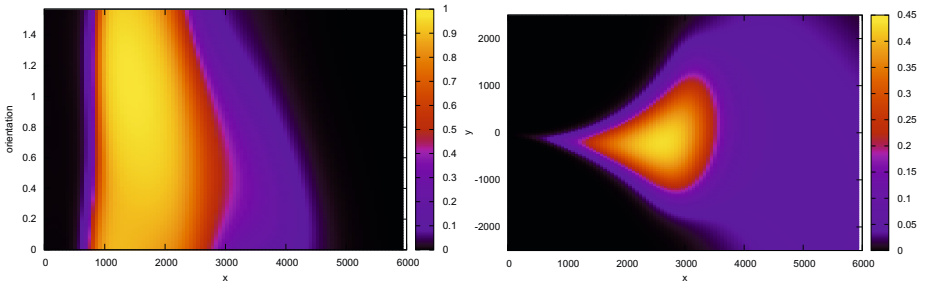

Fig. 3. Sensor model for lines and the right goal post. The color denotes the probability to recognize a line depending on its orientation and distance to the robot (left) and the right goalpost depending on its relative position to the robot (right).

3.3 Expected Observations

The probability function $p(z|x')$ needed for the Monte Carlo Exploration gives the probability for a measurement z in a given state x'. It is necessary to predict the measurements for different actions to calculate utility values for action selection. The raw measurement used for state estimation is the camera image. However the vision modules of the RoboCup team *B-Human* extract features from the camera image. Therefore, the self-localization and the ball tracking are feature-based localizations using the features recognized as observations. These features are field lines, goal posts, the middle circle, corners, and the ball. An observation z consists of a set of visible features. Predicting an observation means to create a set of features that could be perceived in state x'. Due to the known environment, the expected observations can be created from the known field model by iterating through all features and checking which features can be seen. The decision whether a feature is visible or not depends on the feature's relative position (x, y) and orientation ϕ to the camera and the current camera tilt angle θ:

$$(x, y, \phi, \theta) \rightarrow [0, 1] \tag{5}$$

The simplest solution would be to check whether an object lies inside the image and is not too far away. But each feature can have special properties that have an influence to the probability of actually being perceived. For instance, horizontal field lines cannot be seen in as large distances as vertical lines. In general, a better solution for solving this problem is to create sensor models by learning the function in equation 5 with examples created from real perceptions. In this approach, the function was approximated with a feed-forward neural network with 20 hidden nodes and by using backpropagation to minimize the output error. With these sensor models it is possible to decide where certain features are visible. For example, Fig. 3a shows the probability of perceiving a field line depending on the distance and the orientation. The maximum distance varies for different orientations as expected. As another example, Fig. 3b shows the probability for seeing a goal post depending on its x and y coordinates. Here,

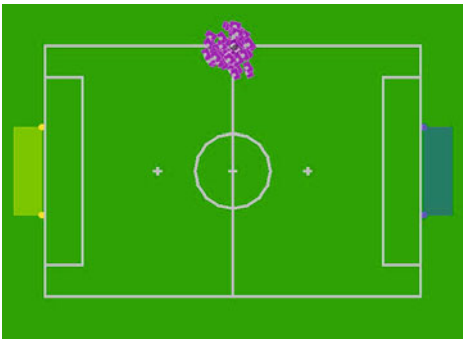

Fig. 4. An example belief. The grey drawing in the middle is the ground truth robot pose.

the orientation is always set to the direction to the right goal post. There must be a small part of the upper bar of the goal visible to detect the side of the goal post. As a result, the colored area is slightly shifted to the right side. Now the expected observations z for a given state x' including any pan/tilt-angles can be created using these sensor models. For the sample set in the self-localization in Fig. 4, the actual mutual information and the expected information gain for the belief and head joint angles are compared in Fig. 5. In that example, the camera was moved from pan angle $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ and back with a constant tilt angle of 20°. The robot is already localized well, so the prediction of the information gain is similar to the actual information gain produced by the perceptions.

The expected information gain for a given belief can be calculated for any camera direction by calculating the average information gain caused by the expected observations for all sample poses. The expected information gain for all reachable camera angles is shown in Fig. 6. The best information is provided by the goals in the upper part. The large red area on the left side is the direction to the center circle.

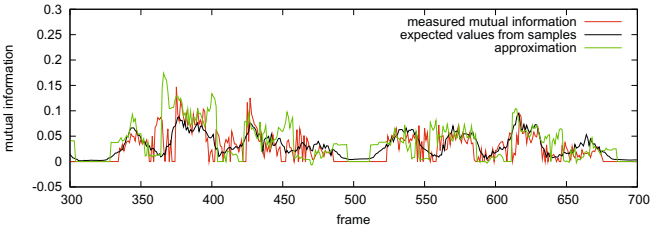

Fig. 5. The mutual information (red) and the expected information gain (black). The values are very similar, because most samples are near the correct robot pose. The actual information gain is only affected by noise. The approximated information gain (green) as described in Sect. 4.

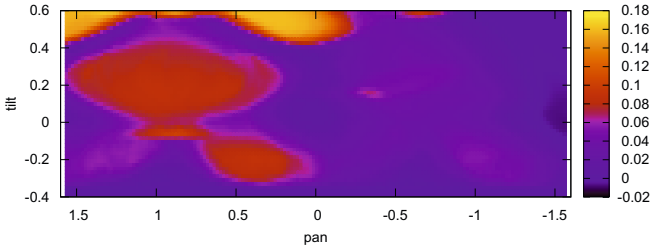

Fig. 6. The entropy expected for all reachable camera angles in one frame from the example in Fig. 4

These values are calculated only for a fixed number of head joint angles defined as actions. The action selection chooses one of these directions as a target position for the head control. The cost of an action is given by the angular distance between the target camera position and the current camera position.

3.4 Feedback

The calculation of the expected information gain is based only on the expected observation for the current belief. However, the expected observations can be wrong due to occlusions or bad sample positions. In Fig. 7, the robot was moved in frame 700. In the following frames the expected values do not match the measured values, so the measured values at an action's position are stored as feedback values and used for some seconds to adjust the expected information gain for this action. This way the action selection does not choose an action with a high expected value, where actually no information has been retrieved a short time before.

3.5 Ball

The calculation of the entropy expected in the localization of the ball is realized with the same method. Because there is only a single type of perception, i. e. a visible ball, an expected observation for a given sample is simply a ball percept

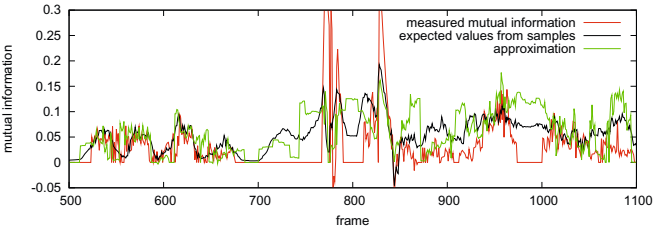

Fig. 7. At the beginning the same data as in Fig. 5, but the robot has been moved in frame 700

at this position. The weightings produced by an observation at a given sample position can directly be obtained by the sample positions and the differences between their distances and angles to the robot. Thus the entropy expected can be calculated directly from the particle positions.

The entropy-based camera control has to optimize the results of the self-localization and ball tracking simultaneously. This is realized by minimizing the sum of both entropies, weighted by a factor for the importance of the ball.

4 Optimizations

The calculation of the action utility values used for action selection described so far is too expensive to be performed for each camera image. The most complex part is to create the expected observations and to calculate the hypothetical particle weightings in the self-localization. In the experiments presented in the next section, these weightings are stored in a look-up-table. This table stores the weightings for a given sample pose produced by the observation at any hypothetical robot pose and tilt angle. By combining the pan angle with the robot rotation, there are seven dimensions left (two poses and the tilt angle). Although the resolution used for the look-up-table is quite low due to the number of dimensions, the approximated entropy values can still be used for action selection (see Fig. 5). There is a high error in the weighting, but the calculation of the expected entropy for an action has been reduced to reading values from the weightings table and calculating the entropy of these values. The calculation of the entropy has been optimized by using fixed-point arithmetics and an approximation of the logarithm using a table for the logarithm to the base 2.

The full calculation of the entropy expected for only one robot pose hypothesis and a given camera direction needs an average time of 41 ms on the Nao. The largest part of this time is needed to create the expected observations (38.5 ms). By using the weights table and the optimized entropy calculation this time has been reduced to less than 0.07 ms.

In the experiments in the following section the active head control uses 44 different actions. Considering that there are 44 actions and 100 particle poses that are possible robot poses, there are 4400 expected entropies to be calculated in every cycle (4400×0.07 ms $= 308$ ms). Consequently the number of robot pose hypotheses has to be decreased significantly. In practice, samples rarely distribute over the state space but cluster at a few positions, as, e. g., shown in Fig. 2 where the probability distribution has only two major modes. Through extracting the distribution's n_m most significant modes from n_s samples and using them as robot hypotheses for entropy calculation, the time necessary for computing can be reduced by a factor of $\frac{n_s}{n_m}$. Of course, this assumes that the mode extraction is computationally comparatively inexpensive. Thus this task is realized by the *Particle History Clustering* algorithm [9] which robustly provides good results in linear time.

The calculation of a utility value in the ball localization needs 0.015 ms on average. In most cases the belief for the ball is a unimodal distribution. Therefore

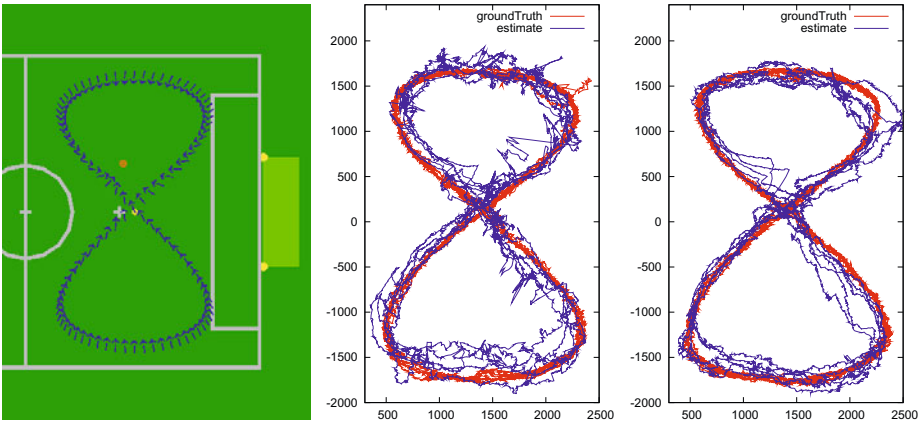

Fig. 8. Left: path used for the experiments. Middle: walked path and the estimated robot position using the passive head control. Right: same for entropy-based head control.

the hypotheses positions used are reduced only to the average sample position, similar to the cluster positions in the self-localization.

Considering an average number of 1.5 clusters in the self-localization and 44 different actions, all utility values are calculated in 5.28 ms. Thus enough processing time is left for the other modules that have to run on a soccer-playing robot (at 30 Hz). All these timings have been measured on a Nao with the operating system and a separate joint control thread running in parallel.

5 Experimental Results

In the experiment, the new active head control is compared to the former head control, used by the striker of the RoboCup team *B-Human*. The former head control points the camera at the ball, at a goal, and at some look-around positions on the left and the right side in regular intervals. The directions to the ball and the goal are based on the current state estimates.

Figure 8 shows the path the robot walks along several times to measure the average errors in the state estimation. The motion control uses the ground truth pose to walk along the path independently from self-localization. As source for ground truth data, a global tracking system has been used. For this purpose, a unique marker has been fixed on the robot's head (cf. Fig. 1) and been tracked by a camera hanging above the field, the software for this purpose is the standard vision system of the RoboCup Small Size League [18]. The system provides the position as well as the rotation (which is fused with the robot's head rotation) of the robot on the field. The self-localization module, described in Sect. 2, has been configured to use 100 samples.

The robot's ground truth position and the estimated state during the experiments are shown in Fig. 8 for the old head control and for the active method.

Table 1. Errors in self-localization and ball tracking

	Self-localization			Ball tracking		
Head control	avg. in mm	stdv. in mm	avg. in rad	avg. in mm	stdv. in mm	seen
passive	163.915	94.8412	0.144	483.547	782.94	37.6%
active	130.062	79.1946	0.105	271.419	360.043	51%

The robot walked along the path 5 times for each experiment. The overall errors in the self-localization and ball tracking are summarized in table 1. The ball importance was quite high in this experiment. By decreasing this value, the self-localization gets better, but the error in ball-tracking will increase again.

6 Conclusions and Future Work

In this paper, we present an approach for an active head control. The method is based on a commonly used entropy-minimizing action selection, but the computationally expensive operations have been executed in a preprocessing step. This allows applying active head control on a robot with low computing power. Thus the action selection takes only a few milliseconds. The error in state estimation in the RoboCup scenario used for the experiments was reduced by 20% in self-localization and by 44% in ball tracking. So both competing state estimations are significantly improved.

A better approximation of the weights with another data structure could improve the results. But nevertheless the computational costs should not be increased too much and an efficient access on the stored weightings is essential for this method to become applied on robots such as the Nao.

Acknowledgements

The authors would like to thank Armin Burchardt for his work at the ground truth system and all members of the team B-Human for providing the software base for this work.

References

1. Burgard, W., Fox, D., Thrun, S.: Active mobile robot localization by entropy minimization. In: Proc. of the Second Euromicro Workshop on Advanced Mobile Robotics, pp. 155–162 (1997)
2. Denzler, J., Brown, C.: Optimal selection of camera parameters for state estimation of static systems: An information theoretic approach. Tech. rep., University of Rochester (2000)
3. Fox, D., Burgard, W., Thrun, S.: Active Markov localization for mobile robots. Robotics and Autonomous Systems 25, 195–207 (1998)

4. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: The nao humanoid: a combination of performance and affordability. CoRR abs/0807.3223 (2008)
5. Guerrero, P., Ruiz-del-Solar, J., Romero, M.: Explicitly task oriented probabilistic active vision for a mobile robot. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 85–96. Springer, Heidelberg (2009)
6. Gutmann, J.S., Fox, D.: An experimental comparison of localization methods continued. In: Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002), Lausanne, Switzerland, vol. 1, pp. 454–459 (2002)
7. Kwok, C., Fox, D.: Reinforcement learning for sensing strategies. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan (2004)
8. Laue, T., Röfer, T.: Particle filter-based state estimation in a competitive and uncertain environment. In: Proceedings of the 6th International Workshop on Embedded Systems, VAMK, University of Applied Sciences, Vaasa, Finland (2007)
9. Laue, T., Röfer, T.: Pose extraction from sample sets in robot self-localization - a comparison and a novel approach. In: Petrović, I., Lilienthal, A.J. (eds.) Proceedings of the 4th European Conference on Mobile Robots - ECMR 2009, Mlini/Dubrovnik, Croatia, pp. 283–288 (2009)
10. Röfer, T., Laue, T., Thomas, D.: Particle-filter-based self-localization using landmarks and directed lines. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 608–615. Springer, Heidelberg (2006)
11. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-human team report and code release 2009 (2009), `http://www.b-human.de/download.php?file=coderelease09_doc`
12. Schill, K., Zetzsche, C., Parakrama, T.: A hybrid architecture for the sensorimotor exploration of spatial scenes. In: Bloch, I., Petrosino, A., Tettamanzi, A.G.B. (eds.) WILF 2005. LNCS (LNAI), vol. 3849, pp. 319–325. Springer, Heidelberg (2006)
13. Seara, J.F., Schmidt, G.: Intelligent gaze control for vision-guided humanoid walking: methodological aspects. Robotics and Autonomous Systems 48(4), 231–248 (2004)
14. Stowell, D., Plumbley, M.: Fast multidimensional entropy estimation by k-d partitioning. Signal Processing Letters 16, 1930–1936 (2009)
15. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
16. Vidal-Calleja, T., Davison, A.J., Andrade-Cetto, J., Murray, D.W.: Active control for single camera slam. In: ICRA, pp. 1930–1936. IEEE, Los Alamitos (2006)
17. Zhou, H., Sakane, S.: Sensor planning for mobile robot localization - a hierarchical approach using a bayesian network and a particle filter. IEEE Transactions on Robotics 24(2), 481–487 (2008)
18. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS(LNAI), vol. 5949, pp. 425–436. Springer, Heidelberg (2010)