Adaptive Methods to Improve Self-Localization in Robot Soccer

Ingo Dahm[1], Jens Ziegler[2]

[1] Computer Engineering Institute
E-mail: ingo.dahm@uni-dortmund.de

[2] Dept. of Computer Science
E-mail: jens.ziegler@uni-dortmund.de

University of Dortmund, D-44227 Dortmund, Germany

Abstract. This paper shows adaptive strategies to improve the reliability and performance of self-localization in robot soccer with legged robots. Adaptiveness is the common feature of the presented algorithms and has proved essential to enhance the quality of localization by a new classification technique, essential to increase the confidence level of internal information about the environment by extracting reliability information and by communicating them via parameterizable acoustic communication, and essential to circumvent manual implementations of walking patterns by evolving them automatically.

1 Introduction

The autonomous mobile robot teams in the different leagues of RoboCup face very special requirements. One of the most important problems that these teams have to solve are navigation and self-localization. Therefore, algorithms are needed that in fast changing, dynamical environments give reliable information about the actual state. Cooperation in teams of robots is a topic of recent research, indicating that reliable and fast communication is crucial for successful control.

In order to improve the quality of navigation, it is necessary to increase the performance of the process that extracts information from the sensory input. Here, an approach is presented that adds a meta level of information to the sensor data: a reliability factor is calculated from visual information that represents the accuracy of the data. The internal representation of the actual state of the game – called *world model* – of each robot is then communicated to other team robots via acoustic communication in order to enhance the local reliability information.

If the robots' localization module depends on odometry data, which is the fact with the robots of our part of the German team, it is crucial for them to have robust locomotion modules. Robustness is used here in terms of returning reliable information about the effective covered distance during a given time with a given walking pattern. Robustness, speed and reliability are all facets of the overall quality of a walking pattern and are used within an Evolutionary Algorithm to automatically develop well performing walking programs in the above mentioned sense.

The rest of the paper is organized as follows. In Section 2, we describe how reliability information can be extracted during classification. The principle of improving accuracy and speed of the robots walking patterns is illustrated in Section 3. Thereafter, we briefly present a robust method of acoustical communication. This is followed by a discussion, how the suggested methods improve the accuracy of self-localization.

2 Vision-Based Navigation

Each legged robot is equipped with a camera. The hardware-based vision processor provides a robust eight-color differentiation [1]. Since main objects are characterized by color [2], a basic object classification can be done by using that module. Nevertheless, observed objects have more properties that can be used for classification. Some of them (e.g. shape, size) can be estimated efficiently or are extracted during image processing anyway [3, 4]. Thus, additional features can be included in the classification process with negligible increase of processing time.

Robot actions are mainly affected by the observed dynamic environment. Accidentally false classified objects can impair the robots' behavior dramatically. Thus, enhancements in classification can improve the overall robots' performance. To improve the accuracy of object detection under the aspects of the real-time limits, we suggest a technique that classifies objects with M properties and assigns a reliability information to every decision. This reliability information can be used for further optimizations in navigation as shown in Section 5.

2.1 Object Classification by Signal Space Detection (SSD)

For that, all extracted M properties of an object are used to determine the classifiers output. Without noisy components, these properties can be viewed as coordinates of fixed points in a M-dimensional signal space (Fig. 1 for $M = 2$). Each of these so-called admissible signal points is associated with a fixed classification. In Fig. 1, an admissible signal point $\underline{c}_i$ is associated with the class i. Given this constellation, the signal space can be partitioned so that decision regions are formed which are bounded by hyperplanes (so-called decision planes).

In typical implementations, the properties of objects cannot be identified ideally. A set of inexactly estimated properties can also be represented as a point in signal space. The classification is determined by the position of this so-called observed signal point $\underline{r}_j$ relative to the bounding hyperplanes. In Fig. 1, $\underline{r}_0$ is associated with class 2 and $\underline{r}_1$ is associated with class 1.

If inexact property estimation can be modeled as exact values disturbed by average white gaussian noise, then the decision planes are easy to estimate. For

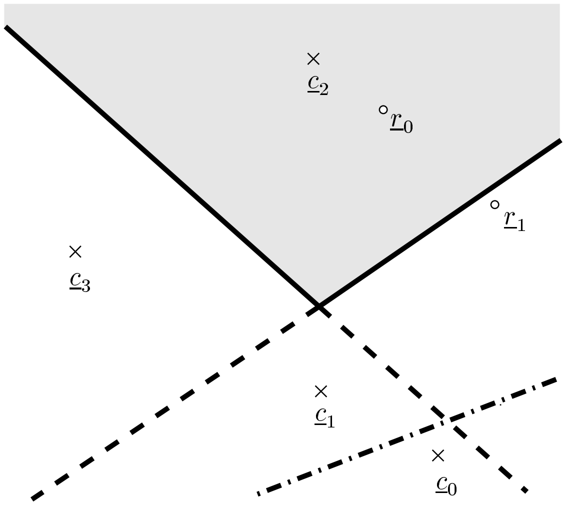

Fig. 1. Illustrated concept of object classification using the Signal Space Detection approach for $M = 2$.

that, the signal space must be partitioned into Voronoi Regions to get classifications of highest accuracy [5].

For colored noise, the euklidic distance is not a good measure to estimate the correct class of an observed object. In some cases, the signal space can be transformed into a white-noise domain [6, 7]. Besides, the hyperplanes can be set using a neural network approach. In the next section, we show how reliability information can be extracted using conventional networks like multilayer perceptrons (MLP) [8, 9].

2.2 Extracting Reliability Information

In [10], a method to extract reliability information using a special SSD detector was introduced. An observed signal point, which is positioned close to an admissible signal point, represents a very confident decision (e.g. $\underline{r}_0$ in Fig. 1). On the other hand, an observed signal point which lies close to a decision plane marks an unreliable decision (e.g. $\underline{r}_1$ in Fig. 1). That is, the distance from an observed signal point to the admissible signal points and to the decision planes is a measure for the reliability of the classification. The consideration of these distances allows the calculation of the individual probabilities for the possible detector outputs: to be member of a certain class ($\underline{r} \in C$) or not ($\underline{r} \in C^*$).

Assuming that the observed signal point is $\underline{r}$, the soft-output for the classification i, which is called $L(i)$, can be calculated with the so-called log-likelihood ratio after [11]:

$$L(i) = ln\frac{P(i = C|\underline{r})}{P(i = C^*|\underline{r})} \tag{1}$$

Note, that a positive value $L(i)$ indicates $\underline{r}$ is classified to belong to C, a negative value means $\underline{r}$ is classified *not* to belong to C. The absolute value of $L(i)$ is the reliability of the decision. In [10], the exact procedure for the calculation of this reliability information using the SSD approach is presented in detail for practical implementations.

Unfortunately, that approach uses predefined admissible signal points. Thus, decision planes can be precalculated and implemented easily. For typical object classifications, neither properties nor noise are known exactly. Therefore, classification is often done by neural networks, typically by multi-layer perceptrons [8, 9].

A single perceptron multiplies the input vector with its weight vector. The dot-product is weighted by the activation function $A(x)$ afterwards as done in eq. (2) [8].

$$A\left((w_1, w_2, \ldots w_{M+1}) \cdot (\underline{r}^T, 1)\right) \tag{2}$$

If $A(x)$ is chosen to be $A(x) = x$, the perceptron represents a hyperplane H as illustrated in eq. (3). Choosing the normal vector of the hyperplane to have a length of $|(w_1, w_2, \ldots w_M)| = 2b/\sigma^2$, the reliability information of $\underline{r}$ is implicitly given by the perceptrons output $H(\underline{r})$ [10]. For that, b is the distance from the admissible signal point to the plane and σ^2 represents the noise variance [10].

$$H : \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_M \end{pmatrix} \underline{r} + w_{M+1} = 0. \tag{3}$$

Since the variance typically changes for different sensor data, it must be measured at run-time or approximated by an estimate of typical channel characteristics. The admissible signal point can be approximated by the center of all observed signal points that are classified to be a member of the same class. This can be done efficiently after the training phase.

Unfortunately, due to simplified learning rules, typically a sigmoid activation function (see eq. (4)) is used in common neural networks [9]. Thus, the distance from H to $\underline{r}$ cannot be calculated directly using conventional perceptrons. To extract the reliability information, the inverse function $A^{-1}(x)$ after eq. (5) must be applied.

$$A : x \rightarrow \frac{1}{1 + e^{c(t-x)}} \tag{4}$$

$$A^{-1} : x \rightarrow t - \frac{1}{c} \cdot ln\left(\frac{1}{x} - 1\right) \tag{5}$$

2.3 Implementation Issues

In the signal space, the decision regions are typically bounded by a set of several hyperplanes. Not all planes contribute to the determination of the reliability information. Thus, we have to choose the appropriate bounding hyperplane for calculating the soft information. In Fig. 1 for example, the hyperplane, which is visualized by the dash-dot line, is obviously not decisive for $\underline{r}_0$. Therefore, we have to define a set of decisive hyperplanes for each output vector. That k-dimensional vector gives the position of the observed signal point relatively to all decision planes. The classification can be done by assigning a class to every possible output vector. This can be implemented using the MLP network approach. On the other hand, the algebraic sign of the extracted reliability can be used with a boolean algebra for classification. This approach is very efficient, especially under run-time constraints.

An implementation of the supposed methodology is very similar to conventional neural networks: M property extraction units (e.g. sensors) are connected with k perceptrons. These perceptrons represent the decision planes. After the training phase, the weights must normalized so that $|(w_1, w_2, \dots w_M)| = 2b/\sigma^2$. To calculate the reliability (resp. the distance of $\underline{r}$ to the closest decision plane), the outputs of the perceptrons must be weighted by A^{-1}.

This calculated additional information leads to a significantly increased accuracy of self-localization. The way how reliability information improves the localization process is presented in Section 5. Nevertheless, the performance of this approach crucially depends on the accuracy of sensory information.

3 Movement

A reliable estimation of the actual robot position even with insufficient sensor data can be done only with high-quality odometry data. Therefore, the development of robust walking is necessary.

3.1 Evolution of Walking

The progress in the development of faster, more robust and computationally more powerful robots is mirrored in the Sony Legged League: this young league of the RoboCup federation has now reached the second generation of four-legged robots and the challenging edutainment robot market is expanding with increasing speed which will have a great influence on the future development of the Legged League robots. A major drawback of this tendency is that with almost every new robot architecture a redesign of the walking programs is necessary. Programming and control of walking robots is difficult, because of the high dimensionality of the movements and the complex sensory and motor limitations, let alone the various uncertainties that arise during the operation. Even if the original program proves satisfactory, parameter changes of the hardware (e.g. a slight change of the position of the center of mass of the robot, changes in maximum acceleration or speed of joints, or changes in weight or length of limbs) may cause unwanted defects in terms of stability, robustness against slackness or speed. In order to avoid this effects and to circumvent additional and expensive work, adaptive methods should be used that automatically generate programs for walking.

The evolution of robot control programs has been the topic of recent publications [12–18] and especially the field of walking robots becomes more and more important. Gait patterns of stick insects have been analyzed to gain more detailed information on natural gait coordination algorithms [19]. Many researchers have often been inspired by biology to build legged robots. An overview can be found in [20]. Nevertheless, the above-mentioned approaches deal with a special instance of an autonomous robot (or walking agent), on which the architecture of the developed control system heavily depends.

It was one of the main goals of this work to make the evolution of robot controllers as independent as possible from morphology specific information. So morphology-related information, although available, will not be used. If, for example, a joint looses the ability to reach certain positions, the outcome of an inverse transformation, which depends on the correct working joint, will be useless. Additionally, the algorithm for the inverse transformation is correct only for a single robot. A machine-learning algorithm, however, should be able to cope with changing hardware and environmental conditions.

A first step towards control of movements of a legged robot is to move the single legs according to a desired trajectory. This trajectory depends on the desired behavior and requires very well coordinated synchronous movements of all joints involved. Thus, describing a movement of a robot requires to give the time dependent values of the acting forces for each involved joint during the motion. The next sections shall now explain the experimental setup and preliminary results from the evolution of walking with Genetic Programming.

3.2 The Evolutionary Algorithm

A first step toward control of movements of a legged robot is to move single joints according to a desired movement of more complex parts of the robot, e.g. a limb. Coordinating the movement results in the necessity to give a time series of motive forces or nominal angles for each joint of the robot, which sums up to 3 joints for a single leg and an overall of 12 joints for the whole four-legged robot at each discrete time step t during the motion. This sequence has to be coordinated in time to achieve the desired movement in sufficient quality. The Evolutionary Algorithm now has to fulfill certain requirements:

– The structure of the individual has to be interpreted as a robot control program.
– Therefore it is necessary to have operations in the global operator set that allow to control motors.
– The quality of the executed individual has to be measured and fed back into the algorithm.
– Better individuals must have a higher probability to spread their genetic information into the next generation.

Parameters	Values
Objective:	Evolve 52 parameters that make the robot walk
Terminal set	Real numbers
Selection scheme	Roulette wheel selection
Population size	20
Crossover probability	0.3
Mutation probability	0.1
Random replacement probability	0.09
Termination criterion	No. of generations
Maximum size of individual	4 walking phases
Initialization method	Random

Table 1. *Koza tableau with parameter settings for the Evolutionary Algorithm.*

– The genetic information must be varied to get an evolutionary drift towards better and better walking programs.

Representation An individual in the current setting is a set of vectors

$$I = \{i_1, i_2, i_3, i_4\} \tag{6}$$

with each i_k being a 14-tuple

$$i_k = (x_1, x_2, \ldots, x_{12}, b, t). \tag{7}$$

The values $x_1, \ldots, x_{12}$ represent motor angles for all 12 motors of the legs of the robot, the value b is a boolean variable indicating whether the movement of the joints to the nominal values x_k shall be a linear movement or a free point-to-point movement. The value t gives the time after which all motors $m_k, k \in \{1, \ldots, 12\}$ have to have reached the desired position x_k. Walking is a cyclic movement and here one loop is separated into different phases. The robot is supposed to be in the first phase i_1 at the beginning of the walking. After a certain time given by t_1, the next phase starts and the motors move to the positions given in i_2. After time t_2, the desired positions are reached and the next phase starts. After reaching the positions in phase i_4, the next phase is again phase i_1, so that the walking is divided into four phases. This rotation scheme continues as long as the robot walks. The evolutionary algorithm now has $4 \times 14 = 52$ degrees of freedom and several additional parameters which are given in Table 1.

In other experiments a different representation was implemented. To avoid possible erratic movements, a variable walking pattern was used that could be adjusted with a set of ten parameters which are variables for a mathematical model of a set of superimposed oscillations that are used to calculate the leg movements. Preliminary experiments used this representation with reduced complexity (see Section 3.2).

Measuring Fitness There are two ways of measuring the fitness of an individual in the current algorithm. The first one is an realization of an interactive evolutionary process: two individuals are executed on a robot and a human observer has to manually decide which of both individuals has a higher quality. This approach circumvents numerical values as the fitness criterion, and the experimenter is relieved of implementing a fitness function that includes 'soft' criteria such as smoothness, grip, and elegance of movements, criteria which can easily be observed and rated by any human experimenter. On the other hand, a comparison of individuals from different generations is difficult because of the lack of objective numerical data. This has led to the implementation of an automatic fitness evaluation module which uses a camera to track the robot during the motion. The distance between start and end point divided by the time used gives the speed of the movement. This setting has some disadvantages that makes it difficult to use, e.g. a robot falling over is probably considered as a fast forward moving robot.

The main problem of both approaches is the time consuming evaluation of individuals, because each individual of each generation has to be down-loaded to the robot, executed and evaluated. Due to the fact that evolution is a 'blind watchmaker', movements are likely to emerge that make the robot stumble and fall, which has the effect that a human experimenter is needed to observe the evaluations *all the time*.

New approaches are currently investigated that try to learn from the user's decisions to develop a heuristic function that can be used instead of the time consuming evaluation with real robots[1]. Artificial Neural Networks will be trained with pairs of individuals along with the observers decision which of them represents the better walking. Once this heuristic is established in sufficient quality, it will be used to evaluate individuals *offline*. The potential effect is less long lasting experiments and a better quality.

Selection Scheme Individuals are compared pairwise and assigned a static fitness value. This method is known as tournament selection [14], but in this special case, a tournament is only needed to discriminate between good and bad performing individuals. The better individual is assigned a better—and static—fitness value that in turn is used in subsequent steps. After all individuals are evaluated, a fitness proportional selection takes place (called roulette-wheel-selection). Selected individuals are either mutated, recombined with other individuals or replaced by randomly initialized new individuals.

[1] The execution of evolved walking programs during long lasting experiments is a strenuous process for both experimenter and robot hardware. It is thus desirable to increase the speed of the evolutionary process (and therewith to decrease the total number of evaluations) in order to minimize the wear-out of expensive hardware.

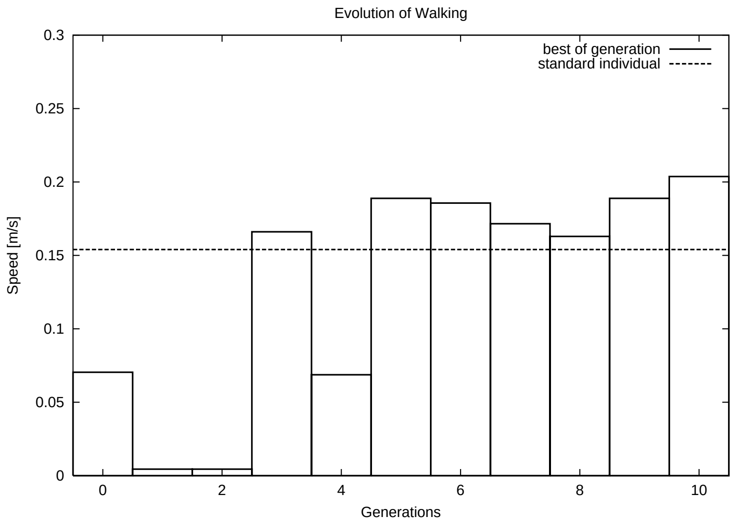

Fig. 2. Development of the speed of the fastest individual per generation. The horizontal line indicates the speed of the standard walking used for soccer.

Mutation One of the 52 parameters of an individual is randomly selected and mutated by adding a standard (0,1)-Gaussian distributed random variable.

Crossover Two individuals are recombined by randomly exchanging single parameters or sequences of parameters. The special parameters b, t are recombined separately due to their different domains.

Results Fig. 2 shows preliminary results of an experiment. Here, the above mentioned simplified representation is used. The speed of the standard parameter setting is shown as the horizontal line at $0.154\frac{m}{s}$. The best individual per generation are presented. It is astounding that, even with a very small population size of four individuals, an increase of maximum speed of nearly 30% could be observed. The only fitness criterion was linear speed of the movement. Neither robustness nor smoothness of the walking pattern were taken into account for fitness calculation. Deviations to both sides during the walking, stumbling or crawling were frequently observed phenomena. This leads to the assumption that additional criteria must be taken into account for fitness evaluation, resulting in a multi-objective optimization [21]. Multiple criteria can be adopted easily in our Evolutionary Algorithm by varying the decision criterion of the tournaments. This leads to a selection pressure on the individuals that is proportional to the frequency of the corresponding criteria. This experiments are promising and indicate that the objective function needs to be defined very carefully in order to yield optimal results.

Having improved odometry data gained from robust walking and reliable vision-based self-localization, the robots position can be estimated even more exactly, if individual information is shared with other team members. Therefore, a communication channel is implemented.

4 Robot Agent Communication

Communication is treated as an enabling technology for efficient distributed work in multi robot teams. A robust communication channel can be used to replace the robots internal world models by a distributed global model. This can be done be broadcasting the calculated model and to receive the other robots observations. Thus, a more reliable localization can be performed.

To evaluate the impact of the given approach, we implemented a robust acoustic communication. The main problem when dealing with that channel is the heavy noise. Therefore, the transmission quality is mainly affected by the audience speech, echos, and ambience noise. The bandwidth is limited due to quality of sensors, run-time constraints and complicated channel features. As the channel parameters vary over time due to robots movement, time dependency of noise, and echos, the communication must be adapted to the actual channel parameters.

The second complicated constraint is the run-time limited processing time. Since world models use to age very fast and the transmitted data is very error-sensitive, the data transmission must be effective and almost error free. Therefore, a fast error correction code (ECC) must be established. The two most common approaches for error correction are the use of Block Codes (e.g. so-called Reed Solomon Codes [22, 23]), and the utilization of convolutional codes as for so-called Turbo Codes [24]. Due to the real-time constraints it is impossible to use a complex code for error detection in robot communication. Such codes are characterized by a time intensive decoding phase. Thus, a specialized parity check code is used for communication by our team.

The relevant information is stored symbolically in a $M \times M$-matrix. Every row and every column is protected by a parity check bit [25]. We use even parity, that means the number of ones in each row and column including the parity bit is even. This is illustrated in Fig. 3. If one bit error inside the $M \times M$ array occurs, the parity check of the corresponding row and column fails. Thus, the error position is easy to detect as the intersection of the erroneous row and column. The error value is given implicitly, since we handle boolean information.

The situation changes, if more errors occur or if parity bits become corrupted. To solve this problem, we use an n-dimensional parity check, for $n = 3$ a parity cube. When all three parity checks fail (column, row, depth), the erroneous check lines intersect at the bit, that is most likely corrupted. If the checks of only two parities fail, the probable error position can be estimated at the intersection of the corresponding row and column. Nevertheless, it is not sure if this bit is really a corrupted one or not: for example, two erroneous bits in a line cause a successful parity check for the corrupted bits, two corrupted parity bits cause unwanted corruption of a correct bit.

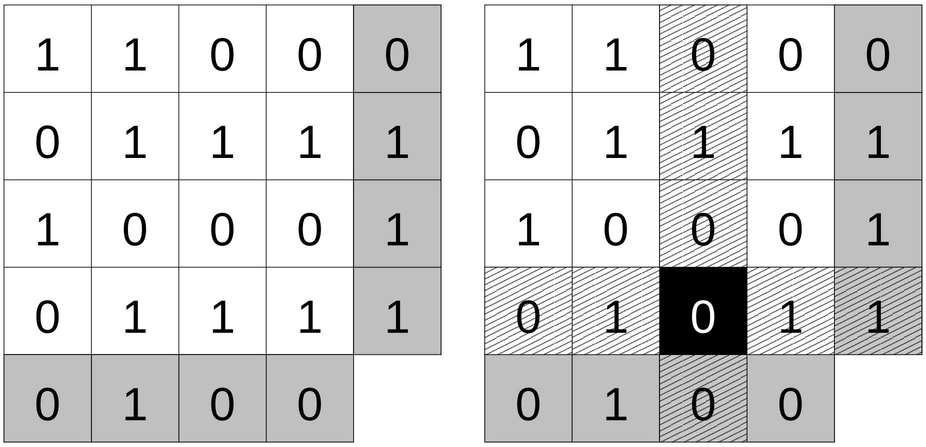

Fig. 3. Parity check code using a two-dimensional concept. Received parities are compared with calculated information. Intersection of defect line and row points to erroneous position.

To improve the error correction capability, the soft information about every bit is very useful. To extract the reliability of a bit, we perform a discrete Fast Fourier Transform (FFT) [26, 27]. The FFT data is then classified using the method given in Section 2. If the reliability of a possible erroneous bit is low, then it becomes inverted. This idea is illustrated in Fig. 4. There, 8 bits are parity checked. The error position cannot be calculated. Since reliability information about every bit is calculated, the symbol with the lowest soft information becomes corrected. The calculation of soft information leads to enhancements in

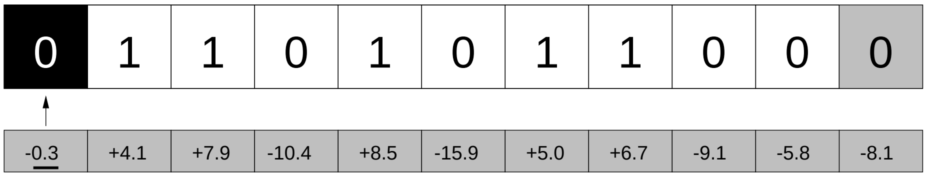

Fig. 4. Enhanced parity check code using soft information to estimate possible error position by examining single bit reliabilities.

error correction capability. Furthermore, we use soft information as a measure of overall channel quality. So we are able to trade code rate resp. bandwidth against error correction capability. The code rate can be adjusted to the actual channel capacity.

The suggested method is very fast, because it uses bitwise operations. This can be done parallel on the robots, because multi-bit operations are possible with our architecture. With that ECC, a code rate of $c(n)$ for n-dimensional parity check constructs after eq. (8) can be obtained. The code rate can be increased by larger M, error correction capability rises with n.

$$c(n) = \frac{M^n}{M^n + nM^{(n-1)}} = 1 - \frac{1}{\frac{1}{n}M + 1} \tag{8}$$

5 Bayesian Based Probabilistic Localization

The additional information about the decision reliability leads to an improved localization and navigation concept. The playing field is separated into discrete grid locations. We create a state-space with these positions. A position in this state space is called s_i. Observations O of landmarks, goals, and walls are combined with a priori information of the actual state and of the robots' movement. To do this, we use Bayes' Theorem [28]

$$P(s_i|O) = \frac{P(s_i)P(O|s_i)}{\Sigma_j P(s_j)P(O|s_j)} \tag{9}$$

where $P(s_i)$ is the a priori probability that the robot is in state s_i. $P(s_i|O)$ is the posterior probability that the robot is in state s_i given that it has just observed O and $P(O|s_i)$ is the probability of observing O in state s_i.

Since $P(s_i)$ is the probability to be in state s_i (without knowledge gained from visual input), a more accurate walking leads to a better a priori estimation of the actual position. Thus, the probability distribution $P(s_i)$ can be estimated more exactly. Providing reliability information of classified objects leads to enhanced observations O. Therefore, the variance of $P(O|s_i)$ can be estimated more exactly. Consequently, $P(s_i|O)$ provides more information with higher reliability.

The impact of the suggested approach is illustrated in Fig. 5. There, a typical scene is shown: The keeper stands inside the opponents goal. Since it hides the right border of the goal, the width of the goal dx is estimated incorrectly at t_0. As distance is calculated by applying intercept theorems, this leads to an incorrect self localization $s_i^{t_0}$ based on the observed situation (bold line in Fig. 5, A^{t_0}). At t_1, both robots have moved. The observation O now shows the goal in its full width. With the conventional approach, the a priori probability of being in one of the states $s_i^{t_1}$ (bold line in Fig. 5, A^{t_1}) is relatively high. A move to the posterior correct position (marked by a cross in Fig. 5, A^{t_1}) seems rather unlikely due to odometry data.

Using reliability information, the observation at t_0 and thus a calculated possible position $s_i^{t_0}$ is marked as low reliable[2] (grey area in Fig. 5, B^{t_0}). In contrast, the reliable observation O at t_1 gives a posterior position $s_i^{t_1}$ (grey circle in Fig. 5, B^{t_1}) with high reliability. Reliable odometry data supports this calculated decision.

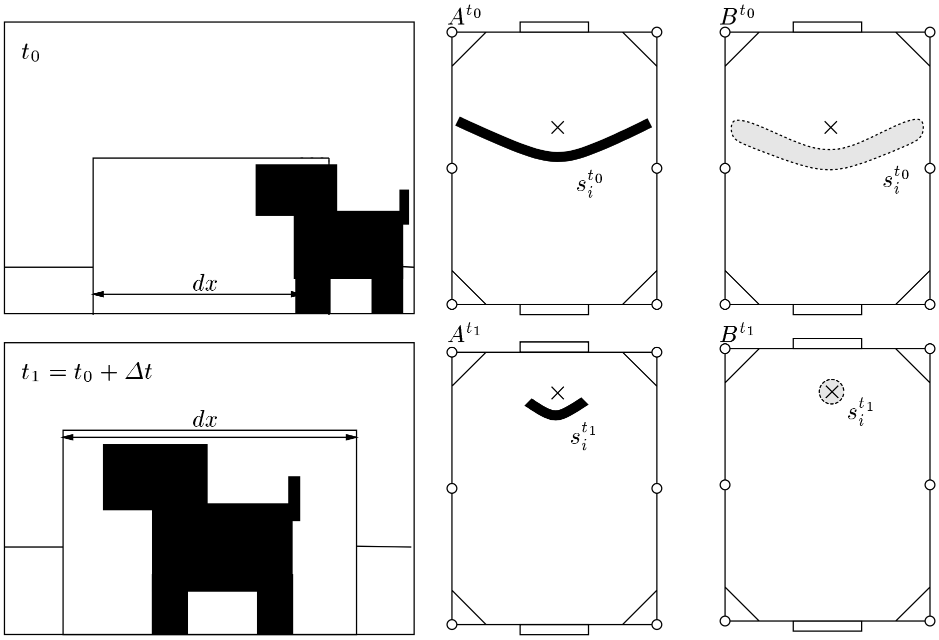

Fig. 5. Observation of a keeper standing inside the goal. Object borders are partially hidden. World model due to conventional concept is given in the middle. Our concept is illustrated right.

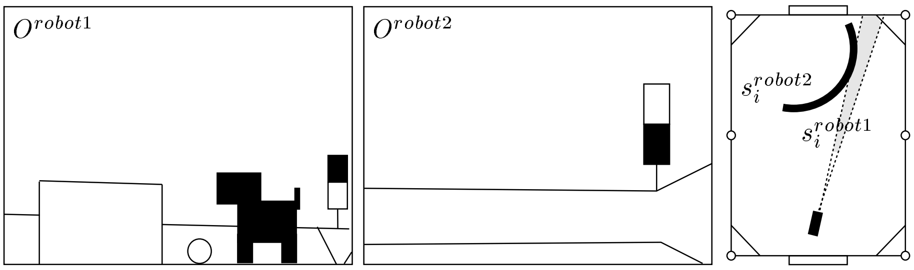

Fig. 6. Observations of two cooperative robots. The left illustration presents the observation of robot 1, the sketch in the middle shows the observation of robot 2. On the right, the impact combining these observations is demonstrated.

The acoustic communication can be used to further increase the precision of the robots world model. Therefore, every robot broadcasts its observed data. For each robot, the received data is synchronized with its internal world model. This

[2] As shape is used to classify objects, the non-rectangular shape of the goal at t_0 indicates a low reliability.

approach is demonstrated in Fig. 6: Since its disadvantageous position, robot 2 cannot localize itself exactly (bold line in Fig. 6). Robot 1 observes the second robot under a certain angle (gray area in Fig. 6). The observations of both robots are combined. Thus, the position of robot 2 can be estimated with significantly improved accuracy.

6 Conclusion

In this paper, we presented a method for increasing the quality of self-localization based on three main improvements: (i) reliability information is extracted from visual sensor input by using an enhanced object classification. This leads to a more reliable observation-based navigation. (ii) the development of robust walking patterns with an Evolutionary Algorithm leads to increasing accuracy and speed of the movements. (iii) the individual knowledge of the world model is shared among team members. This is done by establishing a stable acoustic communication channel. Thus, the advantages of both approaches are combined to diminish uncertainty in special situations.

The RoboCup tournament will show if the presented theoretical improvements in self-localization do positively affect the overall performance of soccer playing robot teams.

Acknowledgments

This work has been done in cooperation with the members of the German team: Humboldt Universität Berlin, Technische Universität Darmstadt and Universität Bremen. Thanks to the team *Ruhrpott Hellhounds*: Matthias Hebbel, Denis Piepenstock, Holger Hennings, Simon Fischer, Jens Rentmeister, Oliver Giese, Ingo Mierswa, Markus Meier, Arthur Cesarz, Patrick Matters, Christian Neumann, Marc Malik, and Lars Schley. Jens Ziegler has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Ba 1042/6-2.

References

1. Manuela Veloso, William Uther, Masahiro Fujita, Minoru Asada, and Hiroaki Kitano, “Playing soccer with legged robots,” in *Intl. Conference on Intelligent Robots and Systems IEEE/RSJ*, October 1998, pp. 437–442.
2. SONY, *Sony Four Legged Robot Football League Rule Book*, SONY, 2nd edition, 2000.
3. J. Bruce, Tucker Balch, and Maria Manuela Veloso, “Fast and inexpensive color image segmentation for interactive robots,” in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ’00)*, October 2000, vol. 3, pp. 2061 – 2066.
4. Francis Quek, “An algorithm for the rapid computation of boundaries of run-length encoded regions,” *Pattern Recognition Journal*, vol. 33, pp. 1637–1649, 2000.

5. T.Jeon and J.Moon, "A Systematic Approach to Signal Space Detection," *IEEE Transactions on Magnetics*, vol. 33, no. 5, pp. 2737–2739, September 1997.
6. G.Stromberg, M.Hassner, and U.Schwiegelshohn, "Signal Space Detection in Colored Noise," *IEEE Transactions on Magnetics*, vol. 36, no. 3, pp. 604–612, May 2000.
7. G.Stromberg, *Signal Space Detection with Application to Magnetic Recording*, Ph.D. thesis, University of Dortmund, September 2000.
8. James A. Anderson, *An Introduction to Neural Networks*, Number ISBN 0-262-01144-1. MIT Press. Boston, 1995.
9. Simon Haykin, *Neural Networks: A Comprehensive Foundation*, Number ISBN 0-02-352761-7. Macmillan College Publishing Company Inc., 1994.
10. S. Schmermbeck and G. Stromberg, "Soft-output signal space detectors (S^3D)," Tech. Rep. 0102, Computer Engineering Institute (CEI), University of Dortmund, 2002.
11. J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," vol. IT-42, pp. 429–445, March 1996.
12. J. R. Koza, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.
13. P. J. Angeline, "Genetic programming and emergent intelligence," in *Advances in Genetic Programming*, Kenneth E. Kinnear, Jr., Ed., chapter 4, pp. 75–98. MIT Press, 1994.
14. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*, Morgan Kaufmann, dpunkt.verlag, 1998.
15. P. Dittrich, A. Bürgel, and W. Banzhaf, "Learning to control a robot with random morphology," in *Proceedings First European Workshop on Evolutionary Robotics*, P. Husbands and J.-A. Meyer, Eds. pp. 165–178, Springer, Berlin, 1998.
16. M. Olmer, W. Banzhaf, and P. Nordin, "Evolving real-time behavior modules for a real robot with genetic programming," in *Proceedings of the International Symposium on Robotics and Manufacturing (ISRAM-96)*, M. Jamshidi, F. Pin, and P. Dauchez, Eds. Robotics and Manufacturing, pp. 675 – 680, Asme Press, New York, 1996.
17. M. A. Lewis, A. H. Fagg, and A. Solidum, "Genetic programming approach to the construction of a neural network control of a walking robot," in *Proceedings of the 1992 IEEE InternationalConference on Robotics and Automation*, Nice, France, 1992, pp. 2618–2623, Electronica Bks.
18. Graham F. Spencer, "Automatic generation of programs for crawling and walking," in *Advances in Genetic Programming*, Kenneth E. Kinnear, Jr., Ed., chapter 15, pp. 335–353. MIT Press, 1994.
19. H. Cruse, "Coordination of leg movement in walking animals," in *From animals to animats. Intl. Conf. on Simulation of Adaptive Behavior*, J.-A. Meyer and S.W. Wilson, Eds. pp. 105–119, MIT Press, Cambridge, MA, 1991.
20. K. Kleiner, "Look to the insect," *New Scientist, No. 1951, 12 Nov. 1994*, vol. 144, pp. 27–29, 1994.
21. Kalyanmoy Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," Technical Report of the Collaborative Research Center 531 *Computational Intelligence* CI–49/98, University of Dortmund, October 1998.
22. I.S.Reed and G.Solomon, "Polynomial Codes over certain Finite Fields," *J Soc.Indust.Appl.Math.*, vol. 8, no. 2, June 1960.

23. W.Blanz, C.E.Cox, G.Fettweis, M.Hassner, and U.Schwiegelshohn, "A Key Equation Solver for variable Block Reed-Solomon Decoders," *IBM Technical Bulletin*, 1995.
24. C.Berrou, A.Glavieux, and P.Thitimajshima, "Near shannon limit error correcting coding and decoding: Turbo-codes (1)," in *International Conference on on Communications*, Geneva, Switzerland, 1993, pp. 1064–1070.
25. F. J. MacWilliams, N. J. A. Sloane, *The Theory of Error–Correcting Codes*, North-Holland, Amsterdam, 1977.
26. J.W. Cooley and J.W.Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.
27. E. Oren Brigham, *The Fast Fourier Transform and Its Applications*, Prentice Hall, 1st edition, October 1997.
28. G. Larry Bretthorst, "An introduction to model selection using probability theory as logic," *Maximum Entropy and Bayesian Methods*, 1993.