A Dutch Trick

The aibo Blindfold Challenge

D.D.C. Knibbe

UNIVERSITEIT VAN AMSTERDAM

A Dutch Trick: the *aiBo* Blindfold Challenge

D.D.C. Knibbe

Artificial Intelligence, specialization Autonomous Systems
Intelligent Autonomous Systems group
Department of Natural Sciences, Mathematics and Computer Sciences
Universiteit van Amsterdam

August 2007

Examination thesis

Supervisor: A. Visser
Key words: multi-robot localization, shared world model, vision

Abstract

In an international robot competition known as *RoboCup*, mainly concerned with football playing robots, several leagues exist. One of these uses the SONY *aiBo*. In this league, special tasks or challenges are held as well. Once, one of these was the *Blindfold Challenge*. The main part of this thesis, is dedicated to this challenge. First I shall introduce the reader into the matter more extensively. Then I shall provide some background, before proceeding with the challenge itself.

At a certain moment during my project, I deemed it necessary, or at least useful, to also do something about head motions used with *aiBo*s for searching the field. In a separate chapter, a new approach to head motions is discussed.

In the appendix, the reader will find the achievements of the Universiteit van Amsterdam in RoboCup-competitions throughout the years.

Acknowledgements

My thanks go out to everyone who helped me in one way or another to reach my goal, in particular Arnoud Visser as my *afstudeerdocent* and supervisor. Special thanks go out to Moritz Schallaböck, Matthias Hebbel and Walter Nisticó of *Microsoft Hellhounds* for sharing their internal documents and part of their code with me which gave detailed insight into their approach.

Furthermore, I thank my parents and all other people who supported me all those years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 RoboCup

RoboCup is an international competition for academic teams of robot programmers and, in some leagues, robot designers, to test their programmes (and robots). Most of the leagues play a football tournament[1]. Their ultimate goal is to have a team of humanoid robots beat the human world football champion in the year 2050.

1.2 The Four-Legged League

Scientifically the most productive league, called the *Four-Legged League*, uses a standard robot known as the SONY *aiBo*, a small, dog-shaped robot.[2] These robots play on a field measuring 6 by 4 metres in total, or 5.4 by 3.6 metres within the outer lines. The goals, as used for the last time in 2006, consist of three panes, one back pane and two side panes, that are white on the outside and either yellow or sky-blue on the inside. The back pane is 80 cm wide, the side panes are 30 cm wide and all three are 30 cm high. The goal area is 1,3 m wide and 65 cm deep. For ease of self-localization, four beacons are used, standing just outside the lines half-way each field half, i.e. on 1,35 m from the centre line and 15 cm outside the outer lines. The beacons have an outer diameter of 10.3 cm and a height of 40 cm in total, the coloured parts measuring 10 cm each. Seen from the bottom of the field shown in figure 1.1, the upper colour of the close beacons is pink, the lower colour is that of the nearest goal. For the beacons on the other side, the order is reversed: the lower colour is pink and the upper colour is that of the nearest goal. The *aiBo*s are dressed in blue (darker than sky-blue) or red.

1.3 The *aiBo*

The *aiBo*s that *Dutch Aibo Team* use are white ones of type ERS-7 (see the picture on the cover). This type has three joints in each leg. The head is connected to the body by three joints: one below the neck and one in the head to tilt the head (the first between 3° backward and 80° forward, the second between 20° up and 50° down) and one to pan the head (between -93° and 93°). The head contains the colour camera, which has a resolution of 208 by 160 pixels. The opening angles of the camera are 56.9° horizontally and 45.2° vertically. The image frame rate is 30 Hz. The shutter speed, gain and white balance can be set to one of three (unspecified) values each. One motion cycle takes 8 milliseconds.[3]

1. A few others execute a (fictitious) rescue, whether or not virtual.
2. Because SONY stopped developing robots (esp. the *aiBo*), the league is now renamed to *Standard Platform League* and will gradually switch to another standard robot.
3. The values mentioned in this section come from the specifications by SONY (for the opening angles of the camera, some people found values different from SONY's, but also different from each other's).

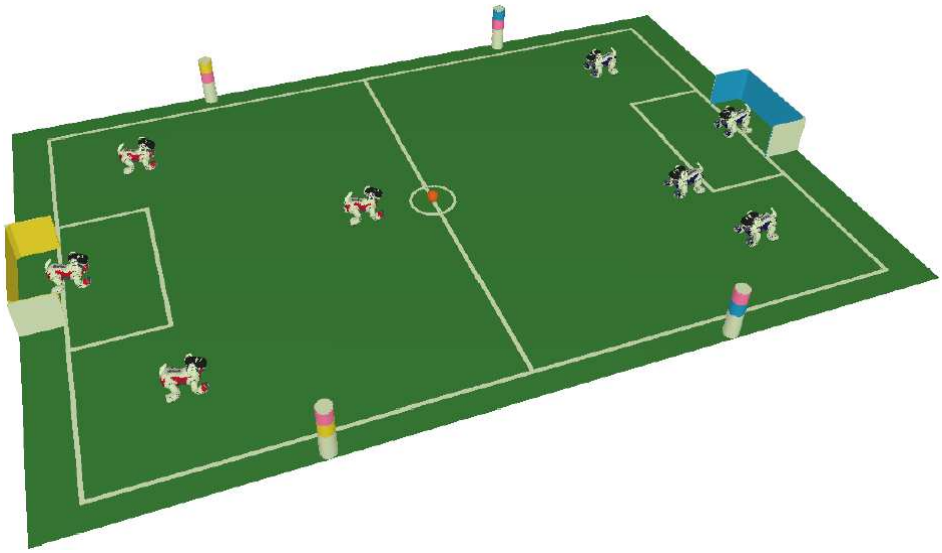


Figure 1.1: The field used in the RoboCup Four-Legged League, with goals (version of 2006 and before), beacons and *aiBo*s.
*(*RoboCup Four-Legged League Rule Book*, 2006, Fig. 4)*

1.4 The Blindfold Challenge

Besides the football tournament, some special tasks, called 'challenges', are executed. One of the proposed challenges once was the Blindfold Challenge. In 2006, *Microsoft Hellhounds* won the Technical Challenge with this.

With the Blindfold Challenge, a robot must walk across the field from one goal to the other, wearing a blindfold but with the aid of three other robots that can use their cameras freely. The 'blindfold' of the first robot is in fact a piece of opaque tape covering its camera.

Chapter 2

Background

The challenge involves two general problems: localization (both self-localization by the blindfolded robot and localization of the blindfolded robot by the other robots) and shared world model.

2.1 Self-localization

For the process of self-localization, several techniques can be used. In 2004, by far the most popular one was Monte Carlo localization, used by e.g. Röfer et al. (2004); Arai et al. (2004); Inoue et al. (2004). Some other teams combine it with a Kalman filter in some way.

2.1.1 Monte-Carlo Particle Filter

In the code used by *Dutch Aibo Team* and based on the code created by *GermanTeam* (Röfer et al., 2004, §3.3), robots self-locate using a Markov-localization method with the Monte-Carlo approach. This approach is probabilistic, modelling the current robot location as the density of a set of particles (see figure 2.1). These particles are formed by the position and the rotation of the robot:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

(the coordinates are in millimetres, θ is in radians). For the method to work, two models, an observation model and a motion model are needed. The former concerns the probability that a particular measurement is taken at a particular location. The latter concerns the probability that a particular action moves the robot into a particular pose.

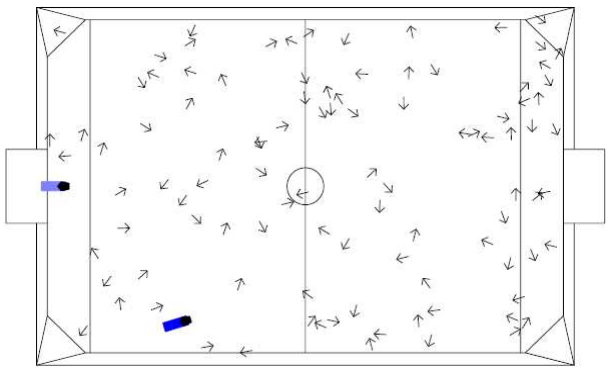


Figure 2.1: A set of particles with two positions of the robot: the real position (bright blue) and the estimated position (darker blue).
(Röfer et al., 2004, Fig. 3.17a)

In the localization process, first the particles are moved in the way that follows from the motion model of the previous action taken by the robot. Next, for all particles, the probabilities are calculated, based on the observation model for the data in the latest camera image. The resampling that takes place after that, based on these probabilities, moves particles to locations of highly probable samples. Then the probability distribution is averaged, giving the current robot pose that is most likely. The process now iterates.

Motion Model

For each sample, a new pose is calculated as

$$pose_{new} = pose_{old} + \Delta_{odometry} + \Delta_{error}, \tag{2.1}$$

where $\Delta_{odometry}$ is the odometry offset since the last localization from the odometry value, which represents the effects of the actions on the robot pose. Δ_{error} is a random error, defined as

$$\Delta_{error} = \begin{pmatrix} 0.1\,d & \times \text{ random}(-1..1) \\ 0.02\,d & \times \text{ random}(-1..1) \\ (0.002\,d + 0.2\,\alpha) & \times \text{ random}(-1..1) \end{pmatrix}, \tag{2.2}$$

where d is the length of the odometry offset (the distance the robot walked) and α the angle by which the robot turned (Röfer et al., 2004, §3.3.1).

Observation Model

The data used are the directions to the vertical edges of the flags and the goals and points on edges between the field and the field lines, the field wall and the goals.

For the calculation of the bearings on the left and right edges of a flag, besides the straightforward calculation of the one on the centre, the distance between the assumed camera pose and the centre of the flag, $distance_{flag}$, and the radius of the flag, r_{flag} are used as follows:

$$bearing_{left/right} = bearing_{flag} \pm \sin^{-1} \frac{r_{flag}}{distance_{flag}} \tag{2.3}$$

For the probabilities, the measured angles are compared with the expected angles, leading to a similarity s. For a measured angle $\omega_{measured}$ and an expected angle $\omega_{expected}$ for a certain pose, the similarity is determined as follows:

$$s(\omega_{measured}, \omega_{expected}) = \begin{cases} e^{-50d^2} & \text{if } d < 1 \\ e^{-50(2-d)^2} & \text{otherwise} \end{cases} \tag{2.4}$$

with $d = \frac{|\omega_{measured} - \omega_{expected}|}{\pi}$. The probability $q_{landmarks}$ of a certain particle is calculated thus:

$$q_{landmarks} = \prod_{\omega_{measured}} s(\omega_{measured}, \omega_{expected}) \tag{2.5}$$

For edge points, the similarity s is calculated from the measured angle ω_{seen}, the expected angle ω_{exp} and a constant σ:

$$s(\omega_{seen}, \omega_{exp}, \sigma) = e^{-\sigma(\omega_{seen} - \omega_{exp})^2} \tag{2.6}$$

Let α_{seen} and α_{exp} be vertical angles and β_{seen} and β_{exp} horizontal angles, then the overall similarity of a sample for a certain edge type becomes:

$$\begin{aligned} q_{edgetype} &= s(\alpha_{seen}, \beta_{seen}, \alpha_{exp}, \beta_{exp}) \\ &= s(\alpha_{seen}, \alpha_{exp}, 10 - 9\frac{|v|}{200}) \cdot s(\beta_{seen}, \beta_{exp}, 100) \end{aligned} \tag{2.7}$$

The filtered probability q' for a certain type is updated for each measurement of that type as follows:

$$q'_{new} = \begin{cases} q'_{old} + \Delta_{up} & \text{if } q > q'_{old} + \Delta_{up} \\ q'_{old} - \Delta_{down} & \text{if } q < q'_{old} - \Delta_{down} \\ q & \text{otherwise} \end{cases} \tag{2.8}$$

The value of $(\Delta_{up}, \Delta_{down})$ is equal to $(0.1, 0.05)$ for landmarks and to $(0.01, 0.005)$ for edge points.

The overall probability p of a certain particle is the product of the probabilities for bearings on landmarks and edges of field lines, the field wall and goals:

$$p = q'_{landmarks} \cdot q'_{field\ lines} \cdot q'_{field\ walls} \cdot q'_{goals} \tag{2.9}$$

(Röfer et al., 2004, §3.3.2).

Resampling

Three methods for calculating possible robot positions were implemented and used to fill a template buffer:

- Using a short term memory for the bearings on the three flags seen most recently.
- Employing only the current percepts.
- Drawing of candidate positions from all locations from which a particular measurement could have been made.
  A sample j is replaced by a candidate position if its probability is lower than the average of all samples, that is, if $\frac{rnd}{n} \sum_i^n p_i > p_j$, where *rnd* stands for a random number from the interval $[0, 1]$.

The probability of a sample in the distribution that is replaced by a posture from the template buffer is $1 - p'_i$. Every template is inserted once into the distribution. Random samples are used if not enough templates were calculated.

Depending on its probability, a sample may be moved locally: it is moved less if its probability is higher. The equation used is this:

$$pose_{new} = pose_{old} + \begin{pmatrix} 100\,(1 - p') \times \text{random}(-1..1) \\ 100\,(1 - p') \times \text{random}(-1..1) \\ 0.5\,(1 - p') \times \text{random}(-1..1) \end{pmatrix} \tag{2.10}$$

(Röfer et al., 2004, §3.3.3).

Robot Pose Estimation

To calculate the robot pose from the sample distribution, first the largest cluster is determined, for which all samples are assigned to a grid of $10 \times 10 \times 10$ cells, one dimension for each of the two coordinates and the rotation. From this grid, the $2 \times 2 \times 2$ subcube containing the most samples is used. The pose is then calculated as the average of all samples in that cluster. For the average value of θ_{robot}, the following formula is used:

$$\theta_{robot} = \tan^{-1} \frac{\sum_i \sin \theta_i}{\sum_i \cos \theta_i} \tag{2.11}$$

The value of θ_{robot} is adapted to fall in the range $[-\pi, \pi]$.

The validity or certainty c of the position estimate is the average probability of all n samples:

$$c = \frac{1}{n} \sum_i p'_i \tag{2.12}$$

(Röfer et al., 2004, §3.3.4).

2.1.2 Improvements

Sturm et al. (2005) describe several improvements over the code created by *GermanTeam*, added by *Dutch Aibo Team* (Sturm et al., 2005, §2.3):

- Decreasing the localization error by using the estimated distance to a landmark in the observation model.
- Increasing the convergence rate by using multiple subsequent landmark percepts in the resampling stage.
- Increasing the precision by checking the final pose estimate against the odometry, which would prevent sudden jumps.

Distances to Landmarks

The estimated distances to the landmarks were included in the observation model by using a Gaussian similarity formula to calculate the quality $q_{measured}$:

$$\begin{aligned} q_{measured} &= s(distance_{measured}, bearing_{measured}, distance_{expected}, bearing_{expected}) \\ &= \exp^{-25\,(1-b)^2 d^2} \end{aligned} \tag{2.13}$$

In this equation, $d = 1 + |\frac{distance_{expected} - distance_{measured}}{distance_{expected}}|$ and $b = |\frac{|bearing_{expected} - bearing_{measured}| - \pi}{\pi}|$. For $d = 1.4$, this equation is the same as equation 3.29 in (Röfer et al., 2004, §3.3.2.4). The quality $q_{measured}$ of each landmark observation is used to update the running estimate $quality_{landmarks}$ with equation 3.33 in (Röfer et al., 2004, §3.3.2.5). The probability p^i of a particle i, which represents a hypothesis of the posterior robot pose, is calculated by multiplying four independent quality estimates as follows:

$$p^i = q^i_{field\ lines} \cdot q^i_{border} \cdot q^i_{goal\ lines} \cdot q^i_{landmarks} \tag{2.14}$$

This time, q_{border} represents the measurement quality of the border outside the field, instead of that of the white wall that was used before.

Landmark Buffer

When it is difficult to see everything clearly, e.g. caused by bad colour calibration or during a game, it takes a long time for the particle filter to converge to the right position. This is caused by the stochastic nature of particle filtering, where upon detection of a landmark, unlikely particles are removed and likely particles are duplicated. If there are no subsequent landmark percepts within a few seconds, the position will get lost again. As a solution, seen landmarks are stored and their reliabilities are decreased in a few seconds. The parameter is adjusted such that roughly every landmark is used twice. For each type of landmark, a small buffer is maintained that contains the N most recent observations y_f and their time frames f. Each y_f has a probability $p(t, f) = Percept_{Change} \cdot Percept^{t-f}_{Decay}$, which is used to update the running estimate $q_{landmarks}$. By using the constant $Percept_{Change} = 0.15$, recent observations that are not subsequent will not dominate the running estimate. The constant $Percept_{Decay} = 0.99$ causes an observation to be forgotten after a few hundred frames.

Check against Odometry

Probably another effect of the stochastic nature of the particle filter and of wrong perceptions is that sometimes the particle filter diverges and suddenly jumps around across the field. Although most of the time the robot finds its position back within a second, this may cause undesired behaviour. Moreover, it is easy to detect such outliers by the extremely low validity of the position. Despite the possibility that the robot may have been kidnapped, it may be better to reject the strange value for the moment, until a good position (one with high validity) is found again, either (close to) the last good position or indeed further away, in case of a real kidnap. For the calculation of the position validity, equation 3.38 in (Röfer et al., 2004, §3.3.4.3) is used (where it is called 'certainty'). When the validity of

the filter using pure odometry data drops below that of the particle filter, the robot pose and validity of the particle filter are used to reinitialize the pure odometry filter. In other cases, the previous values are reused, but the validity will drop to zero within a few seconds. The filter producing the robot pose with the highest validity is chosen to provide the robot pose for the current frame.

Results

The average localization error achieved by the original module by *GermanTeam* was 21.3 cm and 8.27°. Recovering from kidnap on average took 4.97 s, with a standard deviation of 4.45 s.

The first modification by *Dutch Aibo Team*, using the distances to landmarks, resulted in an average error of 14.3 cm and a kidnap recovery time of 2.16 s.

Only using the second modification, with multiple subsequent landmark percepts, led to a kidnap recovery time of 2.09 s and an average position error of 17.3 cm.

The sole use of the third modification, concerning odometry filtering, gave an average position error of 12.8 cm and 4.64°.

Using all modifications together did not further increase performance. The results obtained were an average error of 13.5 cm and 5.02°. The higher error in comparison with the error from the third modification alone was due to occasional noise in the measurement procedure. Repeating the measurements several times will lead to more reliable results. The kidnap recovery time was 2.14 s on average with a standard deviation equal to 0.71 s. This indicates that the modifications led to a stable self-localization, but not at the cost of reaction time or kidnap recovery time.

2.1.3 Settings

In the code as I used it (and I did not change anything with respect to this[1]), the odometry filter was used, as well as the bearings to landmarks, but the landmark buffer and the distances to landmarks were not. A maximal number of 100 samples was used, which is not explicitly stated in (Röfer et al., 2004) as being *the* number of samples, but as an *example* value (Röfer et al., 2004, §3.3.2.3, §3.3.5).

2.2 Blindfolded Robot Localization

2.2.1 Player Localization

Currently, team-mates do not actually locate one another, but they use the data they receive from one another. Opponents only are seen as obstacles. There is a module for locating players, though, but it is not robust enough: as Schallaböck points out, there are many factors that worsen the quality of the localization, such as the colour of the tricot (esp. blue), a low detection rate beyond a certain visual range, the narrow field of view and the association problem caused by the fact that the robots in one team look identical (Schallaböck, 2006, p. 1). Mahdi et al. used the module in their project and found that it works best if the distance is at most 1.1 m. Above that, team-mates can no longer be perceived and recognized (Mahdi et al., 2006, §4.1). Hebbel et al. found, that even under ideal conditions, red robots close to an observing robot were detected only 50% of the time, while, if more than a metre away, they would typically not be detected at all and for blue robots, things were even worse (Hebbel et al., 2006, §3.3). In my experience, red pixels often are classified as pink or orange and blue pixels tend to be classified as sky-blue or black. I used the method for some time, the blindfolded robot being dressed in red, and I often was confronted with such or other errors, leading to something like ‘ghost robots’.

The method considers the number of pixels on a vertical image line that have one of the colours used for the tricots, being red and blue. If the number exceeds some threshold

1. I actually did not inspect that part of the code until after the experiments.

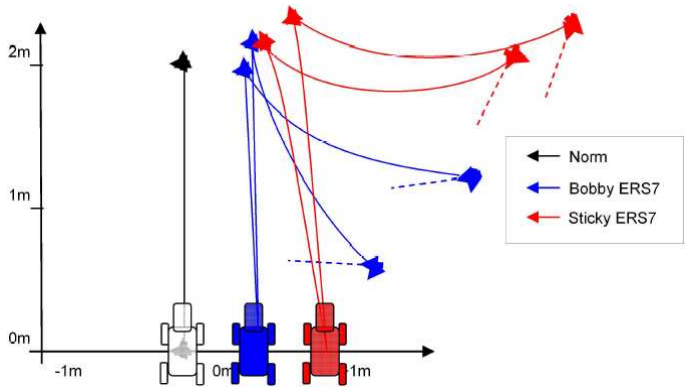


Figure 2.2: Effects of odometry on self-localization errors. *(Sturm, 2006, Fig. A.3)*

value, meaning the robot may be close, the distance is calculated from the intersection of the line through the upper tricot-coloured pixel and the camera with a horizontal plane on the height of where the robot tricot is to be expected. With a number not too far below the threshold (very small numbers are ignored), meaning a probably larger distance, the image line is followed downwards until a field-coloured (i.e., green) pixel is found. The distance to the robot is then calculated from the intersection of the line through the pixel and the camera with the field plane. In this case, the shortest distance in the cluster composed of the distances calculated from all vertical scan lines is used (Röfer et al., 2004, §3.2.8).

The positions of the detected robots, relative to that of the observing robot, are converted into absolute ones and, if outside the field, projected onto the field border. When at a certain location in the discretization of the field the number of perceived robots exceeds a threshold, that location is taken to really be the location of a robot. From this location an absolute field position is calculated. If a player has communicated its position, that position replaces the calculated one, unless it is too old. Otherwise, the calculated position is used, but, to avoid a team-mate being represented twice, it must have some minimal distance to all received positions (Röfer et al., 2004, §3.7).

2.2.2 Object Tracking

As noted above, one of the difficulties for a blindfolded (or blind) individual (robot or human (or (almost) any animal, for that matter.)), is to find out where it is. Probably the most likely method is to ask others about it, but, if the starting position is known, either by asking or by hard-coding it and carefully applying it, it is also possible to rely on odometry data for calculating the other positions. This is unreliable, because it is very sensitive to situations when the robot is moved by other means than walking (e.g. by hand) and, more often, to slippage and other noise in the mechanical system (e.g. when the robot is stuck).

Related to the latter source of problems is a test described by Sturm. In this test, a robot tried to walk forward along a straight line for two metres, turn 90° clockwise, walk sideways to the left for two metres and turn 90° anti-clockwise (which should make it turn back to its original position and orientation), using only its odometry data. With this test it was possible to evaluate forward and sideways walking as well as turning in both directions. As figure 2.2 shows, walking straight forward works fine, but turning and sideways walking result in large errors growing worse and worse the longer the walking types are continued (Sturm, 2006, §A.3).

In the method developed by Schallaböck (2006), multiple particle filters are used at the same time. Each instance of a particle filter models the position of one robot. In the update procedure, for every percept i and filter j a degree of association $D_{i,j}$ is calculated. If the sum of all degrees of association is below some threshold, the perceived robot is assumed to be new and a new filter is instantiated that represents that robot. Whether

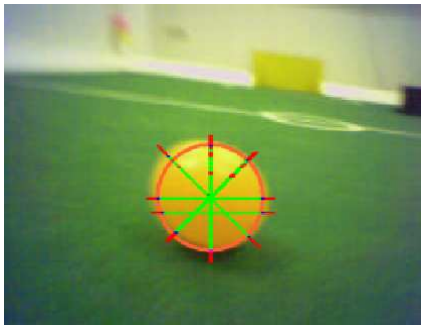


Figure 2.3: Ball percept with scan lines. *(Röfer et al., 2004, Fig. 3.10a)*

the existing filters update their measurements depends on $D_{i.j}$. The new weight w_{new} of a particle is based on the old weight w_{old} and the weight calculated from the percept w_{calc} through the equation

$$w_{new} = D_{i.j} \cdot w_{calc} + (1 - D_{i.j}) \cdot w_{old} \tag{2.15}$$

With very high degrees of association, the measurement update very strongly depends on the new weight, but with very low degrees of association, the new percept only hardly affects the particle filter. Filters whose particles share practically the same space are merged, assuming that they model the same robot. A filter whose particles have drifted very far apart because of a lack of percepts are considered to be lost and are deleted.

Because one $2N$-dimensional filter effectively is replaced by N two-dimensional filters, far less time is needed for computations, despite some additional complexity caused by the larger number of filters. Also, when there is a lack of input data for a subset of the filters, their particles are not pulled towards other robots, but drift apart in a Gaussian manner. This effectively increases the uncertainty of the affected filters.

To locate the blind robot, models that probably describe other team-mates are filtered out, although better results are obtained when the observers have a different colour or no colour at all. The position is stabilized further by using the median of some number of previously modelled positions.

For the rotation angle of the blind robot, the expected position, calculated from the odometry data, is compared to the perceived position. However, because the model is not accurate enough, the angle is not reliable, which at present makes it necessary that the robot is started with a known rotation.

In the Open Challenge, where this algorithm was tested, success was limited to about one third of the cases (Hebbel et al., 2006, §6.3.4).

2.2.3 Ball Detection

Following a suggestion made by Arnoud Visser, I finally decided to use the better detectable (orange) ball instead of the tricot for tracking the blindfolded robot more reliably. I did not conduct a quantitative experiment on the maximum distance from which the ball could still be detected, but based on the findings from a few simple tests I think it is somewhere around 4 m. The ball was mounted onto the front of the robot, hanging loosely between its front feet.

For the detection of the ball (Röfer et al., 2004, §3.2.5), the camera image is scanned in eight directions from the centre pixel in the longest run of orange pixels, being horizontally, vertically and diagonally (between top-left and bottom-right and between top-right and bottom-left), until in all directions the pixel does not resemble (true) orange or the image border is reached. This is shown in figure 2.3. In the latter case, the search continues along two lines parallel to the image border.

If the majority of the pixels that were scanned are classified as orange and the ma-

jority of the end pixels on the scan lines are not yellow, an attempt is made to calculate the centre and the radius. The first attempt is to use edge pixels lying next to a green pixel. In case there are not at least three of those or not all possible edge pixels are covered by the resulting circle, other possible edge pixels not lying on the image border are added, starting with all high-contrast pixels and if necessary continuing with the other pixels. If still unsuccessful, even the possible edge pixels on the image border are used.

2.3 Shared World Model

In order to maintain a good knowledge of the situation on the field, it often is useful to share some information with team-mates. Most teams include in their shared world models at least the position of, or with respect to, the ball (Veloso et al., 2005, §3; LeBlanc et al., 2004, §8; Röfer et al., 2004, §3.4.3, §3.7; TecRams, 2004, §1.1.2; Ruiz-del-Solar et al., 2004, §3; Stone et al., 2004, §2.1). In some cases, the positions of the robots themselves are shared, too (Veloso et al., 2005; Röfer et al., 2004; Stone et al., 2004).

Chapter 3

My Approach

3.1 Shared World Model

The seeing robots do not share additional information among themselves, but some special challenge data are sent to or received from the blindfolded robot.

At first, I considered having the seeing robots calculate the positions of and propose the actions for the blindfolded robot. The blindfolded robot then would have had to choose between the proposed actions, based on, e.g., the distances to the team-mates, because a shorter distance might imply a better judgement of its position. Another method would have had the seeing robots fuse in some way their positions of and action proposals for the blindfolded robot into one position and one action, which would have been known by all seeing robots.

As a third and definitive method, I decided that the blindfolded robot might as well itself fuse the position estimates and decide by itself on an action. In this case, with a proper fusion method, it is easier to choose between seeing robots and get to an action (first method) and only one robot (the blindfolded robot) does the work done in threefold by three (seeing) robots (second method). For the calculation of the position, see §3.4; for the action selection, see §3.6.

3.2 Blindfolded Robot Identification

For the identification of the blindfolded robot, I use the validities of the robot positions and their player numbers. When the validity of one robot's position does not reach 0.1[1] within the four first seconds of self-localization, it is assumed that the robot is the blindfolded robot. It then sends a special message to its team-mates. The first time they receive that message, the team-mates store the player number of the sending robot and use it where necessary to identify that robot as the blindfolded robot.

The only case where this might go wrong, is when another robot has such very serious problems with self-localization that the validity of its position will not exceed 0.1 in time. This would lead to more than one robot believing to be blindfolded. In the experiments, described in chapter 4, this situation did not occur.

In practice it can never happen that the real blindfolded robot will not conclude thus, as one should always verify that the 'blindfold' still covers the camera. If some light (colour) reaches the camera, some pixels in the image can lead to the false detection of a landmark and thus to the calculation of a (false) position. If the amount of light is too large, the validity may become higher than 0.1.

1. Remember from §2.1.1 that the validity is a probability, hence from the interval [0, 1].

3.3 Searching for the Blindfolded Robot

When a seeing robot has not seen the blindfolded robot (i.e., the ball) for one second, it starts searching for it. At first, it pans its head for 2.5 s. If that is not enough, it advances 1 dm (in 1 s) and repeats the pan. If that still does not help, it turns 90° (in 1 s). The sequence of pan-walk-pan-turn is now repeated as much as necessary. If the robot has turned around completely (360°) without success, it gives up searching, which it expresses by looking down. If, at any moment, the blindfolded robot is seen again, the searching process is reset.

When a seeing robot has not seen the blindfolded robot recently and is too far from the blindfolded robot, as computed from the estimate(s) done by the seeing robot(s) (believing to be) standing close enough to the blindfolded robot, the far robot does not even try searching, as it most probably will not succeed. This requires that, at the start of an experiment, every seeing robot be positioned so, that, if it is not too far away, it can see the blindfolded robot without turning its body.

3.4 Calculating the Position of the Blindfolded Robot

For every robot i out of N seeing robots, at time t, it is checked whether it has seen the blindfolded robot at most 0.2 s ago and if so, the position estimate $B_{t,i}$ of the blindfolded robot, reported by that seeing robot, is used by the blindfolded robot to calculate its position P_t, weighted by the position validity $v_{t,i}$ of that seeing robot. In that case, $v_{t,i}$ is used also to calculate the position validity v_t of the blindfolded robot, both as the validity itself and as a weight. It may be clearer to split these two uses and attach the name $w_{t,i}$ to the latter. The equations then become:

$$P_t = \frac{\sum_{i=0}^{N-1} B_{t,i} \cdot w_{t,i}}{\sum_{i=0}^{N-1} w_{t,i}} \tag{3.1}$$

$$v_t = \frac{\sum_{i=0}^{N-1} v_{t,i} \cdot w_{t,i}}{\sum_{i=0}^{N-1} w_{t,i}} \tag{3.2}$$

If the distance between the previous position P_{t-1} and the current position P_t is more than 5 mm, the orientation θ_t of the blindfolded robot at time t is calculated as follows:

$$\theta_t = \tan^{-1} \frac{P_{t,y} - P_{t-1,y}}{P_{t,x} - P_{t-1,x}} \tag{3.3}$$

If none of the seeing robots saw the blindfolded robot recently, the difference at time t between the previous odometry vector O_{t-1} and the current odometry vector O_t (that represent the positions calculated from the joint angles since the robot started) is considered. If the distance travelled, i.e. the length $||O_{t-1} - O_t||$, is more than 1 cm, the position of the blindfolded robot is calculated simply as follows[2]:

$$P_t = P_{t-1} + O_t - O_{t-1} \tag{3.4}$$

v_t is now reduced by 1% and θ_t is again calculated as in equation 3.3.

If the length of $O_{t-1} - O_t$ has been considered but does not exceed 1 cm, the blindfolded robot is assumed not to have moved and the previous position P_{t-1} and orientation θ_{t-1} are reused. The validity is reduced by 0.3% (because not moving does not influence the real position, the reduction can be lower).

3.5 Determining the Destination

In the determination of the destination, provisions are made for the blindfolded robot to cross the field in the y-direction.

2. By mistake, however, the equation $P_t = P_{t-1} + O_{t-1} - O_t$ was used in the experiments

The blindfolded robot waits until the validity of its position reaches 0.8. Then the x-coordinate is compared to the x-coordinate G_x of either the own or the opponent's ground line, whichever is nearer. The y-coordinate is compared to the y-coordinate G_y of either the left or the right ground line, whichever is nearer. If $|x - G_x| < |x - G_y|$, the destination becomes $(-G_x - l,\ y)$ if $-G_x < 0$ or $(-G_x + l,\ y)$ if $-G_x > 0$, l being the body length of the robot, to make 'sure' the robot will have crossed the far line completely.[3] If $|x - G_x| \geq |y - G_y|$, the destination becomes $(x,\ -G_y - l)$ if $-G_y < 0$ or $(x,\ -G_y + l)$ if $-G_y > 0$.

3.6 Planning the Next Move

If the validity of the position is below 0.5, meaning that the blindfolded robot has not been seen for quite some time, the robot will not move, waiting for the other robots to see the ball again.

In other cases, two aspects are considered: the deviation from the main track (the signed distance to the line from the starting position and the destination: negative if the robot is left of it, positive if the robot is right of it) and the orientation w.r.t. the intended direction. The algorithm goes as follows, with a forward speed of 1 dm s^{-1} and a turning speed, where applicable, of $\pm 15°\ s^{-1}$:

```
if the distance is more than 2 dm
    if the relative orientation is at most 15° towards the main tr ack
        turn towards the main track
    else if the relative orientation is at most 45° towards the ma in track
        move straight forward
    else
        turn away from the main track
else if the distance is more than 1 cm
    if the relative orientation is parallel to, or away from, the main track
        turn towards it
    else if the relative orientation is at most 15° towards the ma in track
        move straight forward
    else
        turn away from it
else
    if the relative orientation is more than 7.5° in either direc tion
        turn back towards the main track
    else
        move straight forward
```

3.7 The Process

I will now describe the full process of the Blindfold Challenge. It is split up into states (see figure 3.1), corresponding with the code. It consists of three parts, a general part, one for a seeing robot and one for the blindfolded robot. Unless stated otherwise, the robot transits from the state being described 'here' to the state described next.

3.7.1 Common states

wait until teammates present

This state is repeated until the robot is receiving messages from all team-mates. Already in this state it tries to self-locate.

3. When there is only one seeing robot (mainly for testing), the blindfolded robot will only cross half of the field, i.e. the destination becomes $(-G_x \pm l,\ y)$, because otherwise the seeing robot would not be able to see the blindfolded robot at all times.

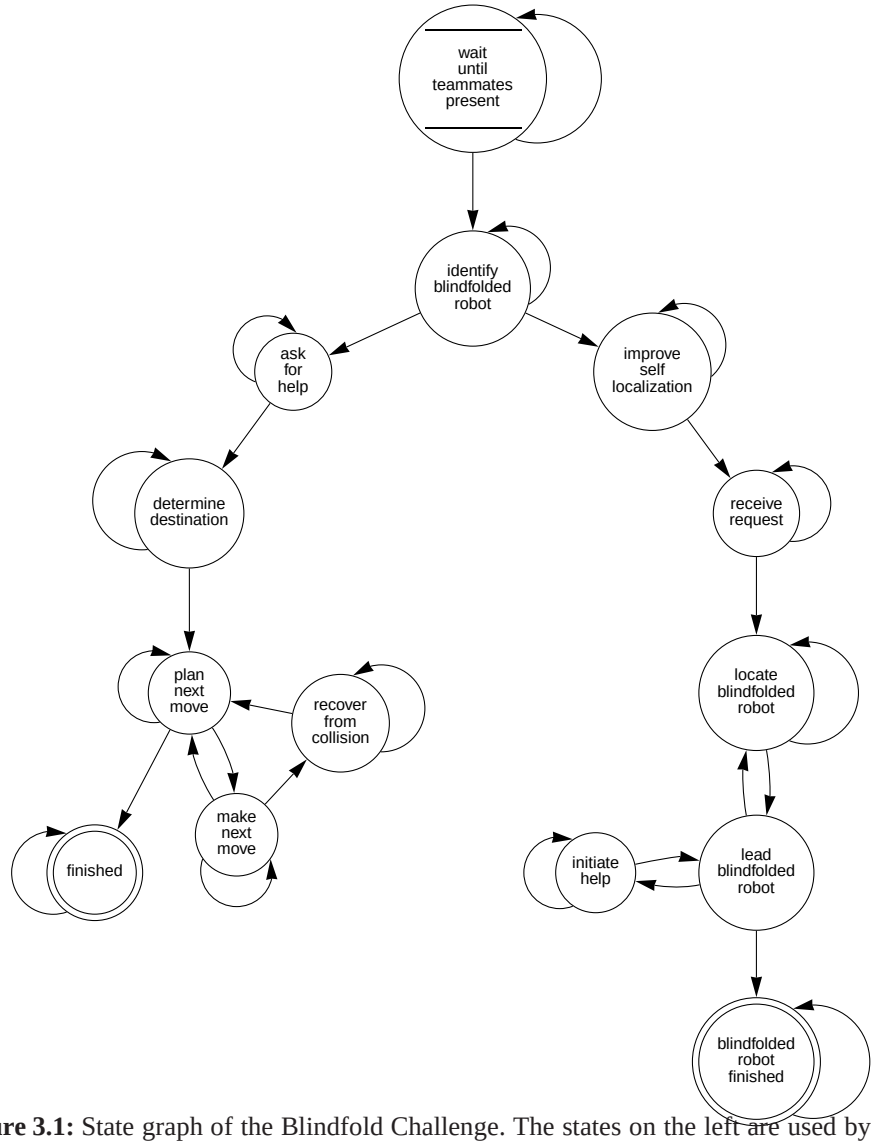


Figure 3.1: State graph of the Blindfold Challenge. The states on the left are used by the blindfolded robot; those on the right are used by the others.

identify blindfolded robot

The result of self-localization, which is still attempted, is used here to draw a conclusion about the identity of the robot, seeing or blindfolded, as described in §3.2.

3.7.2 States for a seeing robot

improve self localization

During at most thirty seconds, the robot tries to get its position validity to a minimum of 0.8 over the last 125 frames (which corresponds to one second).

receive request

The robot waits for a help request from one of the other robots, which is hoped (by the experimenter) to be the blindfolded robot.

locate blindfolded robot

The robot searches for the blindfolded robot, as is described in §3.3.

lead blindfolded robot

If help for the blindfolded robot has not yet been initiated, the process transits to state initiate help to do so. Otherwise, if a message is received from the blindfolded robot that (it believes) it has finished, this robot stops (transiting to state blindfolded robot finished).

initiate help

The robot sends a message (esp. to the blindfolded robot) that it has received a request for help, as long as it still receives such a request. After that, it transits back to state lead blindfolded robot.

blindfolded robot finished

The robot has finished its help and expresses this by sitting itself down.

3.7.3 States for the blindfolded robot

ask for help

The robot sends a request for help to its team-mates, until it has received a reply from all of them.

determine destination

The robot uses the method described in §3.5 to calculate the coordinates of where it wants to go to.

plan next move

If the robot believes it has finished, it transits to state finished. Otherwise, using the algorithm described in §3.6, the robot decides what to do next: nothing, turn left, turn right, or move straight forward.

make next move

If one of the front legs of the robot hits something, it transits to state recover from collision, otherwise it makes the move planned in state plan next move during half a second.

recover from collision

The robot uses the negative values of the speeds set in state plan next move, i.e. it walks backward. It does so for two seconds, then transits back to state plan next move.

finished

The robot expresses its belief that it has finished by sitting itself down.

Chapter 4

Experiments

4.1 Variables

In the experimental stage, I varied three aspects:

- the number of robots (one or three seeing robots),
- the method used for searching for the blindfolded robot and
- the lighting (whether or not using floodlight).

In case of one seeing robot, that robot was positioned such, that it should have been able to see the blindfolded robot at all times, if necessary by turning, if the latter would walk more or less in the right direction.

For the second variable, the variation is in the head movements used: either only panning the head ('search horizon'), or also raising or lowering it every now and then ('search field'). In the latter case, objects closer to and further from the robot can be found, but, because the head movements are more dynamic, at the cost of stability.

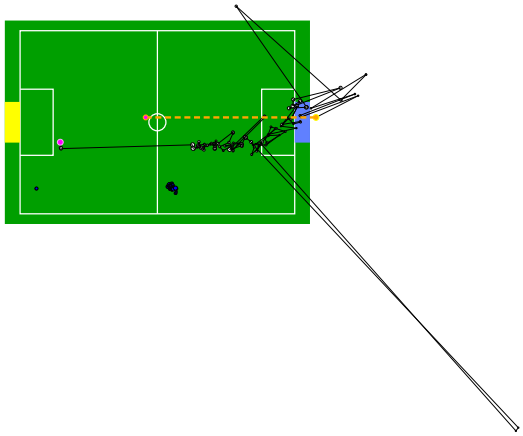
The value of the last variable influences the setting of the shutter speed: fast in case of floodlight, moderate otherwise. Experimenting with different lighting conditions was done to test the robustness.

4.2 Process Log

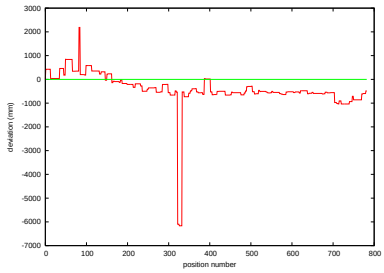
To be able to collect the results of the experiments, I made the blindfolded robot record several data at every moment. These data were written to a log file on the memory stick of the robot every 2500 lines or explicitly by a function call.

Each line of the log contains the following data:

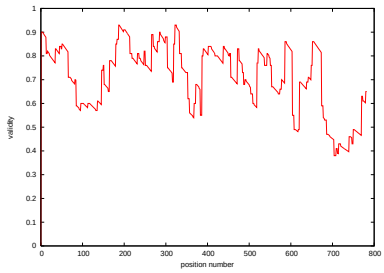
- From the blindfolded robot itself:
  - the *x*- and *y*-coordinates of the current position (mm);
  - the current rotation (rad);
  - the validity of the current position: negative if the position was calculated using odometry data, positive otherwise;
  - the *x*- and *y*-coordinates of the calculated destination (mm);
  - the deviation from the main track (mm).
- From each other robot:
  - the player number;
  - the *x*- and *y*-coordinates of the current position (mm);
  - the current rotation (rad);
  - the validity of the current position;
  - the *x*- and *y*-coordinates of the most recent estimate of the ball position.



a. The path that the blindfolded robot believes to have taken.



b. Deviation.



c. Position validity.

Figure 4.1: Results from experiment 1.

4.3 Results

In the field drawings in this section, the orange spot marks the starting position (i.e., the position from which the destination was calculated) of the blindfolded robot, the pink spot with orange ring (on the other end of the thick, dashed orange line) marks its destination and the plain pink spot marks its final position. The (black encircled) white and grey spots mark its other positions, where white is used for the newly calculated positions and grey for those involving odometry data. The black encircled red, yellow and blue spots mark the positions of the seeing robots. Red, yellow and blue spots without a black outline (used in figures 4.2b and 4.3d) mark the positions of the blindfolded robot as estimated by the corresponding seeing robot. The red, yellow, blue and white spots vary in size with the validity[1]. These spots and the grey spots in the drawings all belong to a position with validity ≥ 0.5.

4.3.1 Experiment 1 — One seeing robot, floodlight, search horizon

Observations

After only about 2 metres in more or less the right direction, the blindfolded robot thinks it has finished and acts correspondingly (see the description of state finished).

Analysis

In figure 4.1a, where the white spots in general show the same as what I saw, it looks very much as if the suddenly very wrong position estimation of the seeing robot itself is responsible for the misjudgement by the blindfolded robot: the position of the blue spot near the lower left corner of the field relative to the white and pink spots close to it is very similar to the other blue positions relative to the last few spots on the right half of the field.

The sudden drop in figure 4.1b corresponds to the points in the lower right corner of figure 4.1a, far outside the field. These are caused by a few erroneous measurements in

1. More precisely, the radius is directly proportional to half the inverse of the squared validity, or $r \propto \frac{0.5}{v^2}$.

blindfolded robot (i.e., ball) localization by the seeing robot, which themselves are probably caused by falsely positive detections of orange (the colour of the ball).

4.3.2 Experiment 2 — Three seeing robots, floodlight, search horizon

Observations

The blindfolded robot meanders towards the other goal and stops in the penalty area.

Analysis

Most white spots in figure 4.2a support my observations, but note the last positions: somewhere on the right half of the field. As can be seen from the large spread in estimated positions of the blindfolded robot in figure 4.2b, the seeing robots strongly disagree on the ball positions. However, averaging their estimates leads to quite reasonable results.

On the first part, the blindfolded robot is lead by the 'red' and 'blue' robots. After a while, the 'yellow' robot joins in and the 'blue' robot loses track. Somewhat further on, the 'red' robot loses track as well. Finally, the blindfolded robot is completely lost, as is indicated by the low position validity (see figure 4.2d).

At the point where the blindfolded robot suddenly seems to be on the right half of the field again, it just switched from the position provided by the seeing 'yellow' robot, to the one provided by the 'red' robot. Apparently, the latter just thought it saw the ball again and the former maybe did not, but the former still had better data and the latter did not. All seeing robots were approximately right about their own positions, though.

4.3.3 Experiment 3 — Three seeing robots, floodlight, search field

Observations

The first four metres the blindfolded robot walks reasonably well towards the other goal, but after that it walks into the upper border.

Analysis

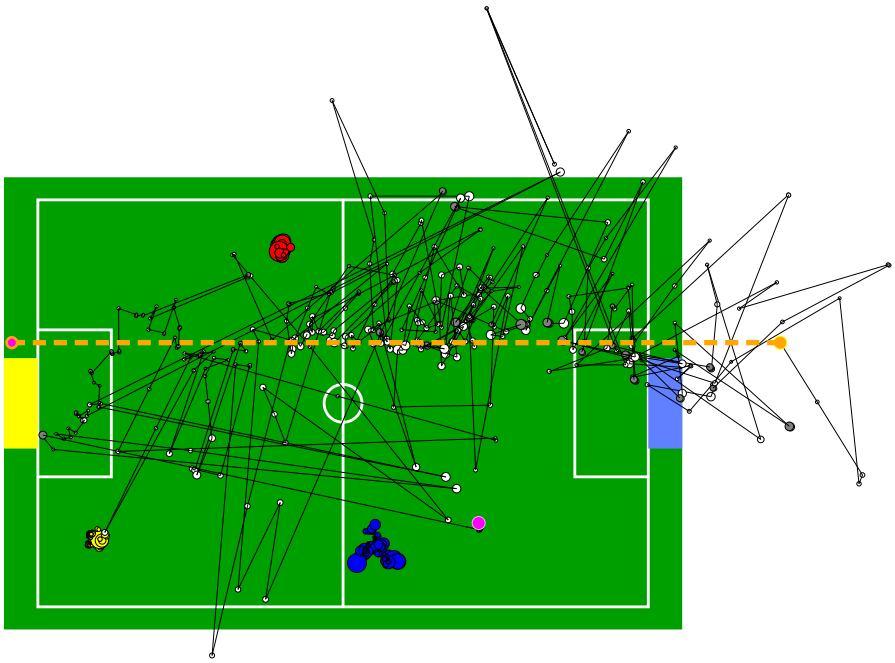
The white spots in figure 4.3a on average show the same pattern, more or less, as what I observed. As figure 4.3d shows, esp. the 'blue' and 'red' seeing robots have similar estimates of the ball positions on the right half of the field, but soon after that lose sight of it. Quite likely this is caused by the fact that all seeing robots were placed facing the sky-blue goal and have a maximum head pan of slightly over 90°. The point where at least the 'red' robot does not see the ball any longer, marks the situation that the maximum head pan is no longer sufficient.

The 'blue' robot, having turned 90° clockwise and now seeing the blindfolded robot from behind, seems to be looking under the blindfolded robot, bringing part of the ball back into view. The 'yellow' robot then takes over, soon losing sight of the ball. The blindfolded robot, carrying the ball, is walking away from the 'yellow' robot, which makes the latter see the blindfolded robot from behind. If the 'yellow' robot then does not see the blindfolded robot straight from behind, but a little from the side, the ball becomes practically or completely invisible.

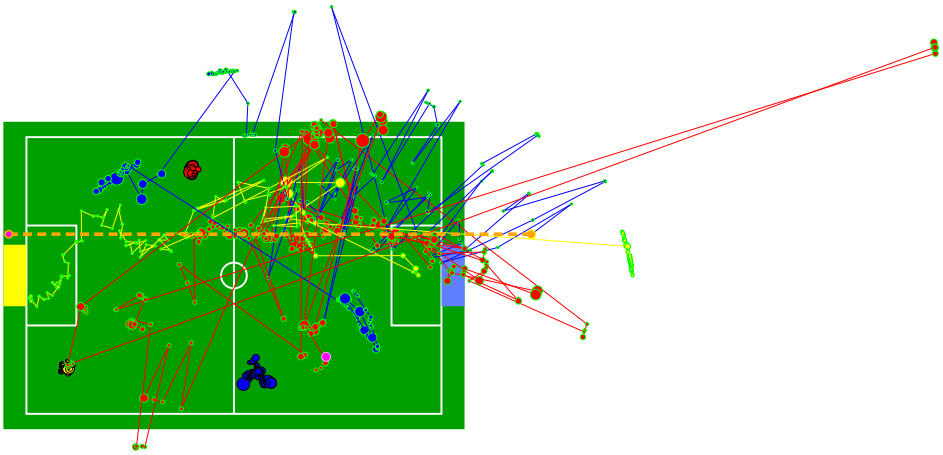
When the blindfolded robot has walked into the border and the 'red' robot has turned 90°, that robot can see the ball again near the upper left corner, but it does not help the blindfolded robot any more. Because of the conditions for recovering from a collision (see state make next move), it cannot get out of the situation in which it finds itself.

4.3.4 Experiment 4 — One seeing robot, no floodlight, search horizon

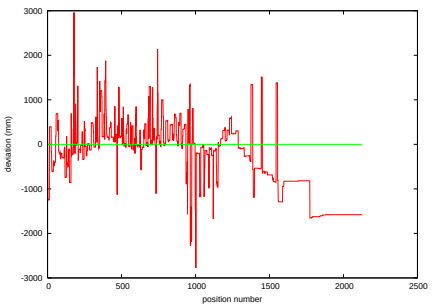
This is basically the same experiment as the one discussed in §4.3.1. The only difference is that this time the floodlight was not used. As the reader will see soon, that did not influence the results much.



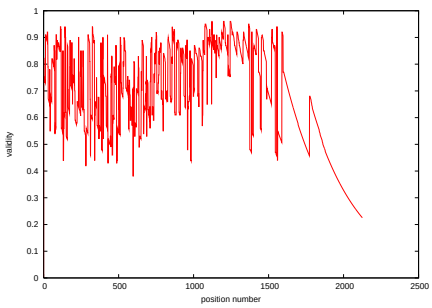
a. The path that the blindfolded robot believes to have taken.



b. The path of the blindfolded robot, according to the seeing robots; one path for each seeing robot.

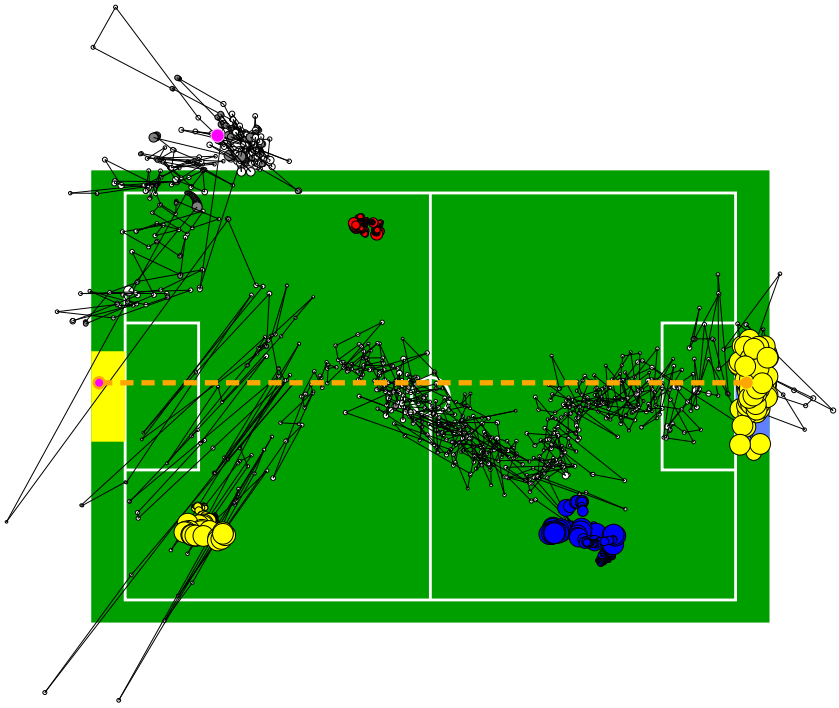


c. Deviation.

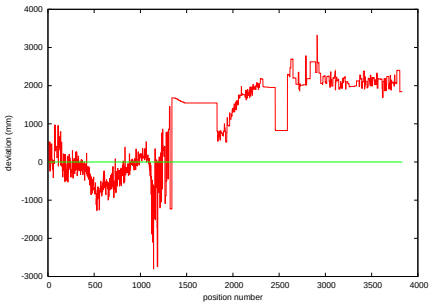


d. Position validity.

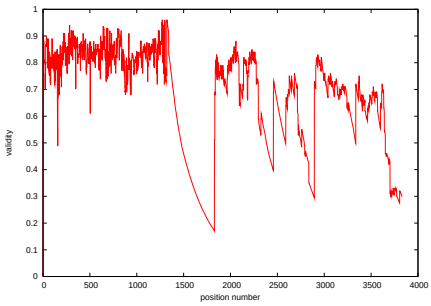
Figure 4.2: Results from experiment 2.



a. The path that the blindfolded robot believes to have taken.

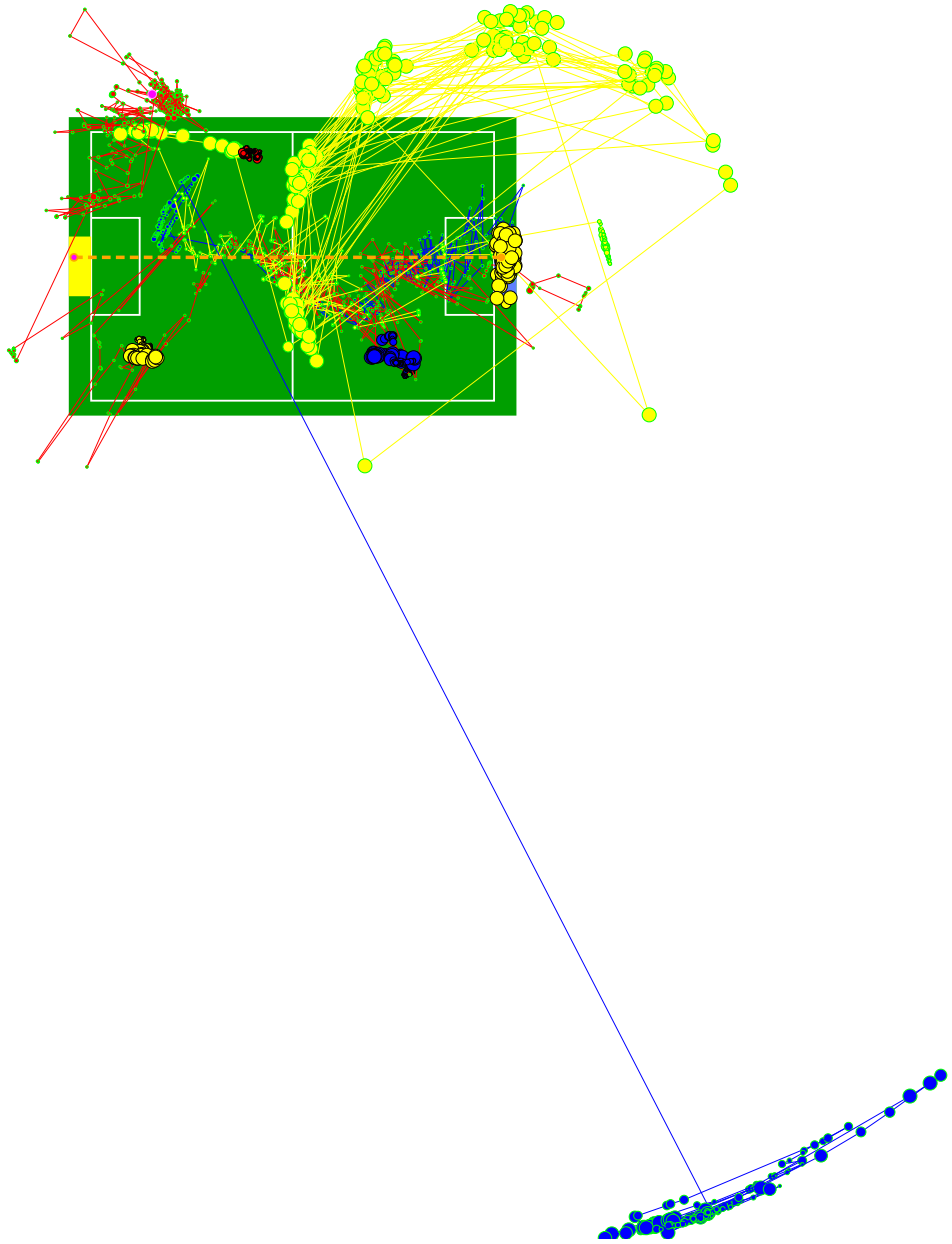


b. Deviation.



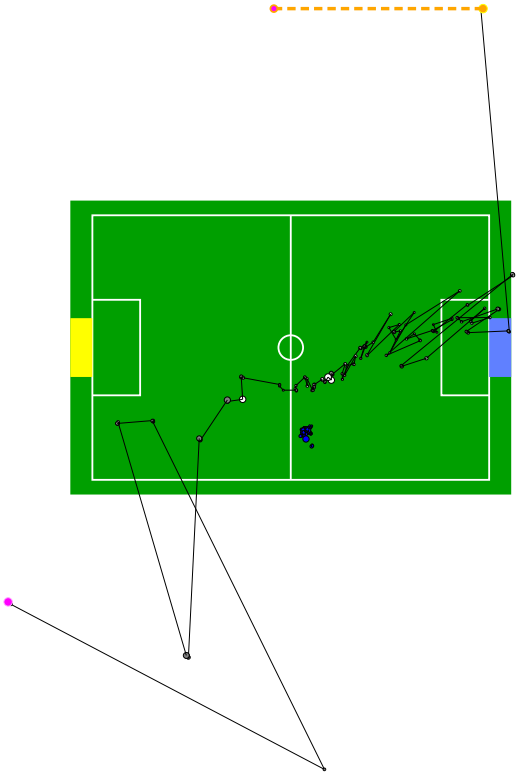
c. Position validity.

Figure 4.3: Results from experiment 3.

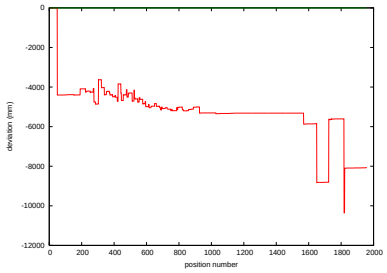


d. The path of the blindfolded robot, according to the seeing robots; one path for each seeing robot.

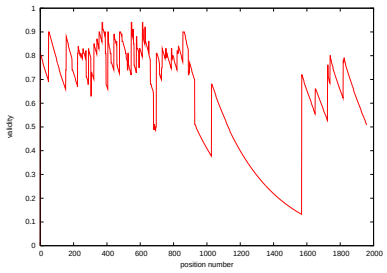
Figure 4.3: Results from experiment 3 (cont.).



a. The path that the blindfolded robot believes to have taken.



b. Deviation.



c. Position validity.

Figure 4.4: Results from experiment 4.

Observations

The blindfolded robot walks more or less in the right (x-)direction, but, when it comes close to the seeing robot, it starts a wide turn around the team-mate. This makes the blindfolded robot go past the destination too widely.

Analysis

Figure 4.4a shows reasonably well what happened in reality, as far as the white spots are considered. Figure 4.4c (too) shows that shortly after position 900, the completely fresh calculation of a position has become very rare (indicated by the small number of rises), meaning that the ball was hardly ever seen any more. This was due to parts of the blindfolded robot obscuring the ball for the seeing robot.

Chapter 5

Conclusions

In the experiments that I conducted, I demonstrated that a blindfolded robot can be guided across the field by multiple seeing robots. However, the success depends greatly on the quality of self-localization and ball localization by the seeing robots. Also the seeing robots must be positioned in a way that they can see the blindfolded robot without turning their bodies, if the distance is not too large. Whether or not using the floodlight did not appear to make much difference, provided that the shutter speed be adapted to the actual situation.

Inspection of the recorded data, combined with my observations of the behaviour of the blindfolded robot, reveals that in many cases a seeing robot has a good position estimate of the blindfolded robot that is not used, due to the upper limit of 0.2 s, used to decide between accepting or rejecting a ball observation, on the time since the last perception of the ball. Therefore, a higher limit may lead to better results. Another point where improvements can be made, is in the search for the blindfolded robot. Currently, the seeing robots do not move much and if they turn, they turn in a fixed direction. They may be made to move more, or to turn in the direction which gives the highest probability of finding back the blindfolded robot, based on the angle where it was lastly seen.

5.1 Future Research

Elaborating on the challenge, one may add the possibility of obstacle avoidance, or develop methods to better recognize one particular robot among a number of robots.

Chapter 6

New Head Motions

6.1 Motivation

At once, during my work on the Blindfold Challenge, the self-localization did not work properly any more. Not yet knowing the cause of that[1], it made me think about possibilities for improving the self-localization.

As with the normal head pan the colours tend to be vague, leading to many misclassifications, I first thought of lowering the speed, using an adapted version of the normal mode, but that did not work well: the fluctuation in the position estimates often was even larger and the validity lower.

My next idea was to only use images taken at a small number of head pan angles. These stills would have distinct colours, providing the best possible classification.

6.1.1 Saccades

The approach is based on the way our eyes move when we are reading. They do not move continuously, as is the case when tracking objects in motion, but jump from one piece of text to the next in *saccades*. Once our eyes have come to rest, cognitive processes do their work (Best, 1992, pp. 353–354).

6.2 Method

In the case of the *aiBo*, which has only one camera that cannot be moved independently from the head, the head does not turn back and forth continuously, like in the existing head control mode used for searching for landmarks. Instead, it has some rest points, between which it moves fast.

At first, I used one of the normal functions for the head movement, but before I got that working properly, something happened which drew my attention. When the process controlling the motions of the robot did not receive data from another process controlling the cognitive tasks of the robot for more than two seconds, the robot makes some jerky movements with its body, resulting in a kind of swinging behaviour. As the normal functions do not make the head turn very fast, I thought the technique behind these jerky movements might be better, if I could get this working.

6.2.1 Special Actions

The technique is used for many special actions, esp. several kicking motions and some entertaining actions, including the swing I mentioned above. It uses special files that can contain values[2] for every joint, to be set at a time. There may be more than one line of joint values, for complex actions. Other things can be put into such files as well, e.g. PID

1. A wrong setting causing a wrong colour table to be used.
2. Or a wildcard character, for joint values that are not important for the action.

data instead of joint values and transitions to other such files, allowing files to be used in different combinations for different actions (Röfer et al., 2004, §3.9.2).

Above, I already mentioned that there are (at least) two processes, one for motion and one for cognition. The main part of the saccadic search method involves the motion process. The algorithm of that part is as follows:

```
if the last head control mode was different from this one
    in the current pan direction,
        calculate the next rest point
        if it is one of the outer points
            reverse the pan direction for the next time
else
    if the current data are newer than the last and the head has been still since
        if the head has been still long enough
            if one of the outer points is reached
                reverse the direction
            take the next point
        else
            wait; reuse the last point
else
    wait; reuse the last point
turn to, or stay at, the specified point
```

In the cognition process, the following algorithm is applied for controlling the image processing:

```
if one of the saccadic search methods is in use
    if the current data are not newer than the last
        inhibit image processing
    else if the head has not moved
        allow image processing
    else
        inhibit image processing
else
    allow image processing
```

Later, I tried some versions with a normal function (the code in fact only differing in the absence of the special action related statements, as the normal function was already present to facilitate a good transition to another head control mode), but then too many images were unstable.

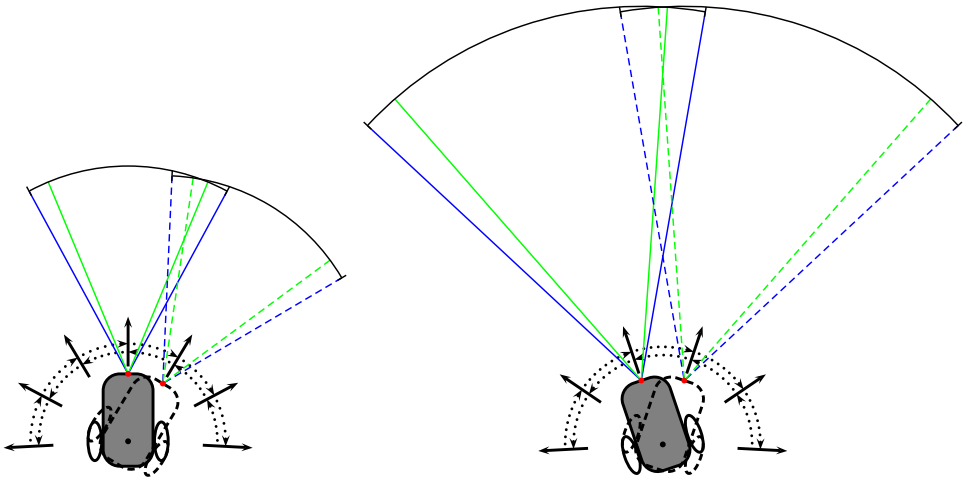
6.3 Experiment

6.3.1 Setting

In the experiments I conducted, I compared four methods:

- the standard, continuous movement
- saccadic movements with seven rest points and a double pause at each rest point
- saccadic movements with seven rest points and a single pause at each rest point
- saccadic movements with six rest points and a double pause at each rest point

Saccadic motion is depicted in figure 6.1, with seven rest points in figure 6.1a and with six rest points in figure 6.1b. The red dots mark the positions of the camera. The blue lines indicate the horizontal size and the green lines the vertical size of the field of view. I included the latter for the horizontal direction because of the radial distortion in the camera images (Röfer et al., 2004, §3.2.3), making the vertical angle more useful for the horizontal direction than the horizontal angle. The figure shows that, esp. when considering the green

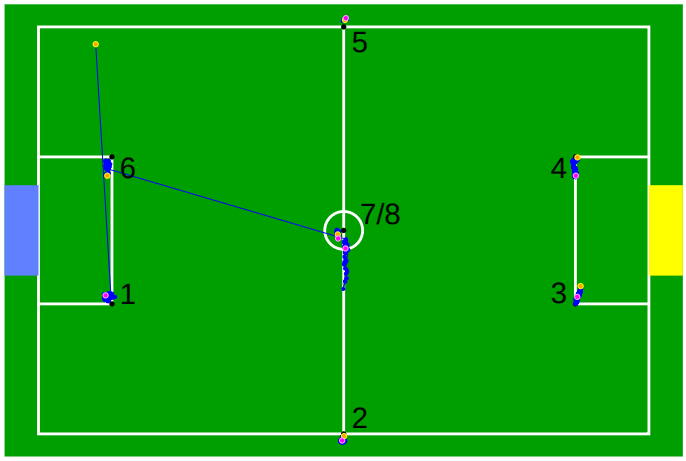


a. Seven rest points. **b.** Six rest points.

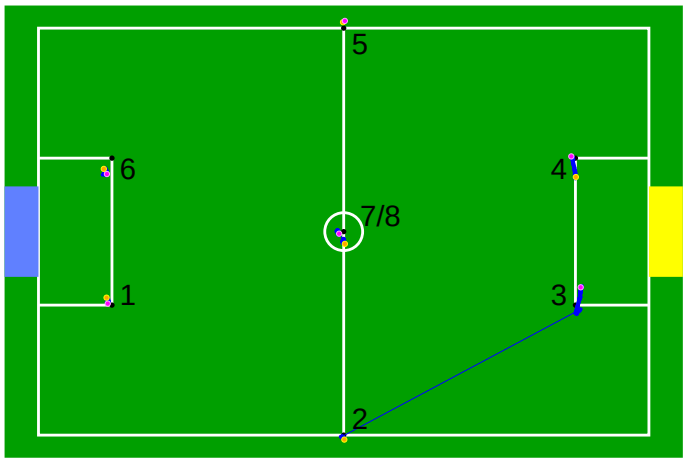
Figure 6.1: Saccadic motion.

Table 6.1: Positions, orientations and head modes used in the experiment.

x (mm)	*y* (mm)	*orient.* (deg)	*modes*
2050	650	-135	continuous → saccadic v1 → saccadic v2 → saccadic v3
0	1800	-90	saccadic v3 → continuous → saccadic v2 → saccadic v1
-2050	650	-45	saccadic v1 → continuous → saccadic v3 → saccadic v2
-2050	-650	45	saccadic v2 → continuous → saccadic v1 → saccadic v3
0	-1800	90	saccadic v3 → saccadic v1 → continuous → saccadic v2
2050	-650	135	saccadic v2 → saccadic v1 → saccadic v3 → continuous
0	0	0	continuous → saccadic v3 → saccadic v2 → saccadic v1
0	0	180	saccadic v1 → saccadic v2 → saccadic v3 → continuous



a. Continuous motion.



b. Saccadic motion, version 1.

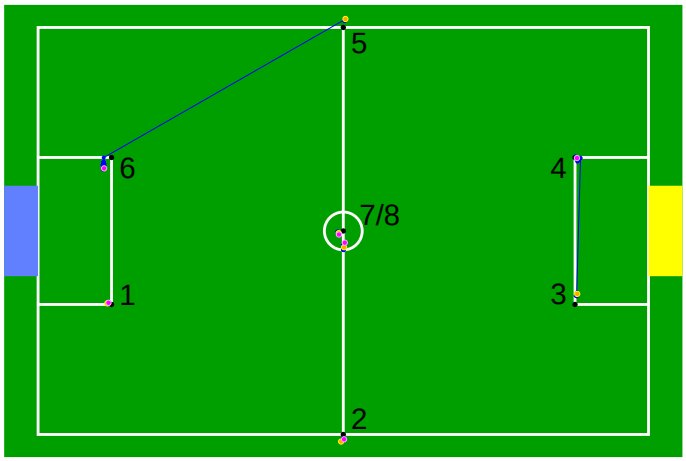
Figure 6.2: Results from the experiment.

lines, the nearest point of the overlap of the fields of view is much further from the camera in the 6-point method, where the angle between two head positions is 37.2°, than in the 7-point method, with an angle of 31° between two head positions.

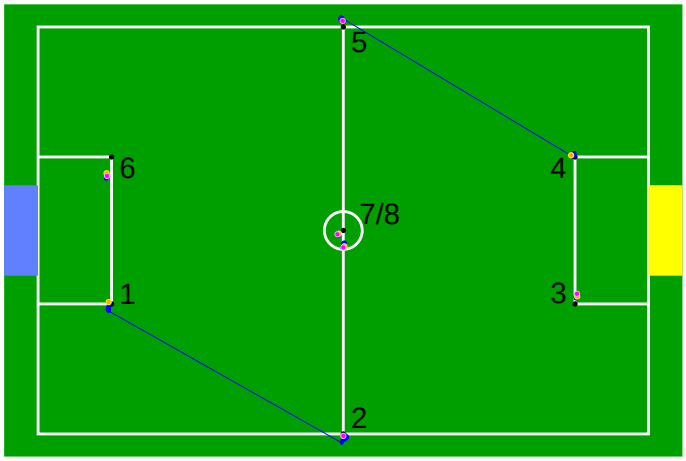
The robot was placed on each of the positions in table 6.1 in turn, with the corresponding orientations. The robot executed a mode for about one minute until I switched to another mode or moved it to another position. After such a kidnap and in the first mode on the first position, it was continuing the same mode for about one and a half minute (at most half a minute to recover). The order of modes was chosen such that as many combinations as possible would be used and such that the modes would equally often be the first on a new position, attempting to prevent the effects of one mode influencing those of another mode.

6.3.2 Results

Figure 6.2 shows the results of the experiment. The orange spots mark the first estimated position at each location, the pink spots mark the last and the blue spots mark all the other estimates. A blue line from one point to another indicates a kidnap, except the blue line to the first point in figure 6.2a, when the position was still completely unknown. The validities of the position estimates depicted are higher than 0.8.



c. Saccadic motion, version 2.



d. Saccadic motion, version 3.

Figure 6.2: Results from the experiment (cont.).

6.3.3 Analysis

All modes perform reasonably well on positions 2, 5 and 8. A bit more problematic in all modes appears to be position 6, where in most cases the spots lie close to one another, but never really close to the real position. Also a bit problematic is position 7, although when viewed in the correct order, there seems to be some progression from one image to the next.

The continuous motion performs worse than the others, as can be derived from the amounts of blue spots. The effect of the transition from position 6 to position 7 looks very bad, but because the robot more or less looks towards position 2 from position 6 and towards the sky-blue goal from position 7, there are but few landmarks that it can see from both positions: only the sky-blue goal and the flag positioned at $(1350, 1950)$ (i.e., just outside the middle of the line between the lower left corner and position 2), which may confuse the robot.

The first version with saccadic movements performs much better. The only real problems are with positions 3 and 4, but the results at position 3 may be influenced by the transition from position 2. One might expect that the effect of the 'transition' from position 7 to position 8 on the self-localization would be even worse than we saw in the continuous mode, as in this case there are no common landmarks at all. However, such effect does not appear here, but that may be because only the orientation changed, not the position.

The worst cases with the second saccadic version, but not really bad, are the positions 4 and (already mentioned) 6. Both positions were reached while this mode was active, so that must be the cause.

The third saccadic mode does not show new problems.

6.4 Conclusions

From the normal continuous mode and the saccadic modes, the third saccadic mode seems to give the best results. However, the landmarks always were at a rather large distance. Close objects may be missed.

On later occasions, when I tried to use one or more of the saccadic modes in the Blindfold Challenge, it (or they) did not work as well as before. Moreover, the continuous mode worked even better, judging by the position validities. A possible explanation may be in the longer time it takes to turn between the outer points than with the continuous mode. There may very well be a possibility to adapt the related parts to the longer time, although it may take some time itself.

Using the technique meant for special actions to create saccadic movements is in general a bad idea. It cannot be combined with normal actions. In the Blindfold Challenge, I once tried to use it for searching the ball, but when that led to a case where the robot had to turn and move its head at the same time, the robot started walking in a wide circle. If there is another way to reach a high head pan speed, a way that *can* be combined with other motion, it may be an improvement to use it. Also, because, while the head is moving, there is no image processing and everything that follows from it, the cognition process can do other or more time consuming tasks.

6.4.1 Future Research

To make the analogy with human eye movements stronger, the robot head could be made to pause at points of interest, like big changes in the colour histogram, instead of using fixed intervals.

Appendix A

Achievements of the Universiteit van Amsterdam at RoboCup

In the past years, the Intelligent Autonomous Systems group of the Universiteit van Amsterdam was involved in RoboCup-competitions in several ways. Here is an overview of the achievements.

1998 – Paris (France):
Emiel Corten won the third prize in the Soccer Simulation League with his team called *Windmill Wanderers*.

2001 – Seattle (U.S.A.):
- Matthijs Spaan and Bas Terwijn participated in the Dutch national team *Clockwork Orange* in the Midsize League, reaching the seventh place.
- In the Soccer Simulation League, team *UvA Trilearn* (Jelle Kok and Remco de Boer) became fourth.

2002 – Fukuoka (Japan):
Again, team *UvA Trilearn* (now only Jelle Kok) reached the fourth position in the Soccer Simulation League.

2003 – Padova (Italy):
- Jelle Kok won the competitions in the Soccer Simulation League with team *UvA Trilearn*.
- In the Rescue Agent Simulation League, team *UvA Rescue C2003* (Stef Post and Maurits Fassaert) became sixteenth.
- Team *UvA Zeppelin*, consisting of Arnoud Visser and Stijn Oomes, ended in the Rescue Robot League in the preliminary rounds.

2004 – Lisboa (Portugal):
- In the Soccer Simulation League, team *UvA Trilearn* reached the seventh place.

2005 – Osaka (Japan):
- Team *UvA Trilearn* (Soccer Simulation League) reached the tenth position.
- *Dutch Aibo Team* reached the ninth position in the Four-Legged League. From the UvA, Jürgen Sturm was present.

2006 – Bremen (Germany):
- In the Four-Legged League, *Dutch Aibo Team* became sixth in the football competitions and won the third prize in the Technical Challenge. The team coach was Arnoud Visser, Jürgen Sturm was also present.
- Bayu Slamet and Max Pfingsthorn won the third prize and the Mapping award in the new Rescue Virtual Robot League with team *UvA ResQ*.

2007 – Atlanta (U.S.A):
In the Rescue Virtual Robot League, the fourth position was reached by team *UvA Rescue*, with coach (and more) Arnoud Visser and with Tijn Schmits and Bayu Slamet as the other members.

(See also (Sturm, 2006, App. D) and the Dutch RoboCup website `www.robocup.nl`.)

Bibliography

Arai, T., Osumi, H., Umeda, K., Ueda, R., Kikuchi, T., Sakamoto, K., Jitsukawa, Y., Komura, M., Kato, H., Shirai, S., Takeshita, K. and Tsunetoh, H. (2004). Technical Report of Team ARAIBO 2004, *Technical report*, The University of Tokyo and Chuo University, Japan.

Best, J. B. (1992). *Cognitive Psychology*, third edn, West Publishing Company, St. Paul, MN, U.S.A.

Hebbel, M., Nisticó, W., Schwiegelshohn, U., Beuchel, T., Czarnetzki, S., Fiedler, D., Kaufmann, J., Kerkhof, T., Kruse, M., Meyer, M., Nasched, F., Neng, T., Rottberg, J., Schallaböck, M., Schmitz, B., Schwertfeger, B., Vincze, A., Wachter, M., Wege, J., Witte, M. and Zarges, C. (2006). Microsoft Hellhounds Team Report 2006, *Technical report*, Robotics Research Institute, Information Technology Section, Dortmund University, Germany.

Inoue, J., Aoyama, H., Ishino, A. and Shinohara, A. (2004). Jolly Pochie 2004 in the Four Legged Robot League, *Technical report*, Kyushu University, Japan.

LeBlanc, K., Johansson, S., Malec, J., Martínez, H. and Saffiotti, A. (2004). Team Chaos 2004, *Technical report*, Örebro University, Blekinge Institute of Technology and Lund University, Sweden and University of Murcia, Spain.

Mahdi, A., de Greef, M., van Soest, D. and Esteban, I. (2006). Technical Report On Joint Actions For An Aibo Team, *Technical report*, Universiteit van Amsterdam, The Netherlands.

RoboCup Four-Legged League Rule Book (2006).

Röfer, T., Laue, T., Burkhard, H.-D., Hoffmann, J., Jüngel, M., Göhring, D., Lötzsch, M., Düffert, U., Spranger, M., Altmeyer, B., Goetzke, V., von Stryk, O., Brunn, R., Dassler, M., Kunz, M., Risler, M., Stelzer, M., Thomas, D., Uhrig, S., Schwiegelshohn, U., Dahm, I., Hebbel, M., Nisticó, W., Schumann, C. and Wachter, M. (2004). GermanTeam RoboCup 2004, *Technical report*, Universität Bremen, Humboldt-Universität zu Berlin, Technische Universität Darmstadt and University of Dortmund, Germany.

Ruiz-del-Solar, J., Vallejos, P., Zagal, J. C., Lastra, R., Gortaris, C. and Sarmiento, I. (2004). UChile1 2004 Technical Report, *Technical report*, Universidad de Chile, Chili.

Schallaböck, M. (2006). Blind Robot Localisation, *Technical report*, Universität Dortmund.

Stone, P., Dresner, K., Fidelman, P., Jong, N. K., Kohl, N., Kuhlmann, G., Sridharan, M. and Stronger, D. (2004). The UT Austin Villa 2004 RoboCup Four-Legged Team: Coming of Age, *Technical report*, The University of Texas at Austin, U.S.A.

Sturm, J. (2006). *An appearance-based visual compass for mobile robots*, Master's thesis, Universiteit van Amsterdam, The Netherlands.

Sturm, J., Visser, A. and Wijngaards, N. (2005). Dutch Aibo Team: Technical Report RoboCup 2005, *Technical report*, Universiteit van Amsterdam, The Netherlands and DECIS Laboratory / Thales Research & Technology Netherlands, Delft, The Netherlands.

TecRams (2004). Team Report RoboCup 2004, *Technical report*, Tecnologico de Monterrey, Mexico.

Veloso, M., Rybski, P., Chernova, S., Vail, D., Lenser, S., McMillen, C., Bruce, J., Tamburrino, F., Fasola, J., Carson, M. and Trevor, A. (2005). CMPack'04: Team Report, *Technical report*, Carnegie Mellon University, U.S.A.