Dutch AIBO Team at RoboCup 2006

Arnoud Visser[1], Paul van Rossum[2], Joost Westra[3],
Jürgen Sturm[1], Dave van Soest[1], and Mark de Greef[1]

[1] Universiteit van Amsterdam
[2] Technische Universiteit Delft
[3] Universiteit Utrecht

http://www.dutchaiboteam.nl/

Abstract. The Dutch AIBO Team is a multi-institute team which competes in the 4-legged robot league of RoboCup since 2004. This team description paper briefly outlines the approach taken for the Passing and Open Challenge of the RoboCup 2006.

1 Introduction

The Dutch AIBO Team is this year a small group of experienced students from the Universities of Amsterdam, Delft and Utrecht. This group gets the support of the Decis Lab, the Universities of Groningen, Twente and Eindhoven, as well as Saxion University of Professional Education. Our unified efforts are intended to foster our individual and joined research interests in collaborative autonomous systems.

Because of the limited size of the team, the focus of our contribution this year will be directed to the Technical Challenges. That the Dutch Aibo Team has a good basis for the Soccer competition was demonstrated at the Dutch Open, where the semi-finals were reached. The competition code has been developed by the Dutch Aibo Team for the RoboCup in Osaka [1]. The team-strategy can be characterized as a strong goalie and aggressive chasing in the center of the field. With this strategy the Dutch Aibo Team received at the Dutch Open overall less goals than the later champion. Yet, this year the progress in the strategy will be minor, most of the current effort is directed to the perception part of the robots. This is less visible during a competition, but can be seen in the short calibration time and the ease of localization during the setup-phase.

As part of our research plan, the Universities of Groningen and Utrecht experiment with another development environment which can be used for educational purposes and research purposes outside the soccer competitions. The New Goal challenge will be implemented in the Tekkotsu framework[4] [2] which may be a stepping stone to a contribution in the RoboCup @ Home competition.

In the remainder of this document we briefly outline our approach for two other Technical Challenges.

[4] http://www.Tekkotsu.org

2 Passing Challenge

As demonstrated in the movie provided with our qualification material[5], the Passing Challenge was used as a benchmark to study the effect of cooperation between the robots [3]. For this benchmark three robots should pass the ball to each other. This sounds simple, but is in practice hard. On the field, both the observations and the actions have only a limited chance on success. Improvements can be made to enlarge the chance on success, but the decision process has to be inherently robust to the unreliability of the real world. In other words, this system is a Partial Observable Markov Decision Process (for an overview see part IV of [4]). To estimate the probability inside the decision process, we have constrained the experimental conditions as far as possible, until we had a simple test with reproducible success rates. This simple test was scaled up step by step towards the Passing Challenge as described in the rules, by eliminating the constraints one at a time. The final step will be to include estimates of the absolute location into the decision process.

Central in our cooperation study is the decision which robot will block and grab the ball. A ball was kicked on a line precisely between two robots, as depicted in figure 1. Because the ball is relatively light and the field is relatively rough, the actual trajectory will not follow the line precisely in the middle of the robots, but will deviate towards one of the robots. The two robots have to decide who will attempt to grab the ball. When two robots make the same decision, we will call this confusion. The decision to grab the ball can be based on own observations (namely ball speed, position and direction of the movement relative to robots local coordination system) or on information distributed via a communication channel.

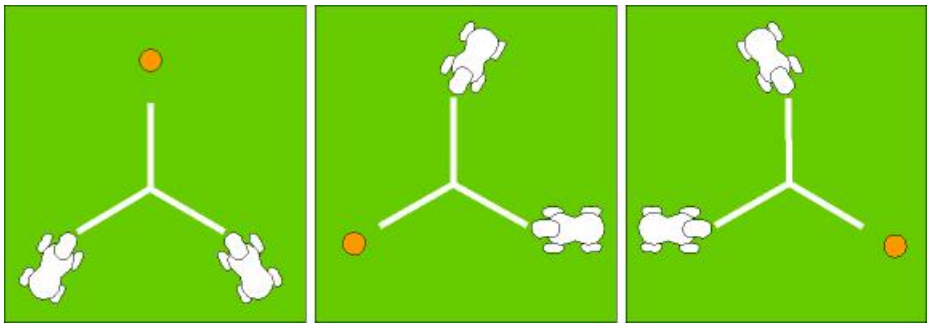

Fig. 1. Experimental setup to measure confusion

Experiments indicated that using extremely simple status messages to share the intentions between the two robots reduced the confusion level. Although the

[5] http://www.dutchaiboteam.nl/robocup/robocup2006/qualification/

observations were correct most of the time (90% success-rate), these observation are repeated so many times that on a certain moment the robots make the same decision. Depending on their own observations robots, the tests showed an average confusion level of 33%. When the two robots shared their intentions via extremely simple status messages the confusion level dropped to 3%. The success of the message driven behavior indicates that the two robots make their decision to grab the ball at a different moment. If there is enough time difference, the status message will be received by the second robot and an equivalent decision can be prevented. It is interesting though to notice that this result is not trivial due to the lack of message synchronization and reliable message transport protocols.

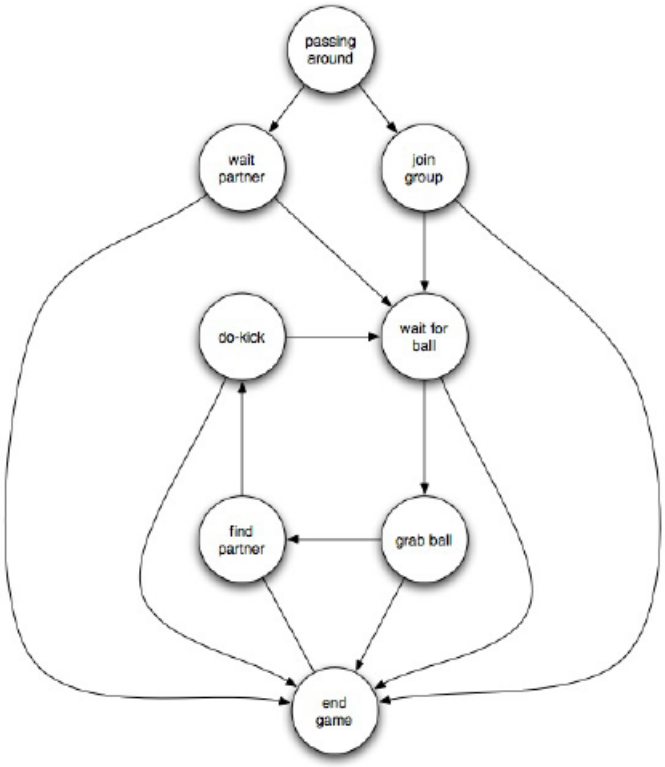

Fig. 2. Behavior design for the Passing Challenge

Based on this result a behavior was designed which allows to pass the ball around, which makes use of the messages when communication channels are available, and otherwise relies in its own observations. The experiments described before concentrated on the success rate of the behaviors **do-kick** combined with

wait-for-ball (see figure 2). To pass around the behaviors where alternated with two other behaviors:

find-partner In order to pass the ball to another teammate, the robot possessing the ball first needs to turn around to face one of its teammates. This means that it is not only necessary to detect whether a teammate is in the field but also to have somewhat of an accurate estimation of the other teammate's bearing. Locating other teammates by turning the robot's head to scan the environment is not a good option, because the robot has no control of the ball in the mean time. Instead the robot turns the body until the robot comes to face the teammate.

wait-for-ball In order to receive the ball, the other robots have to keep a certain distance from the ball if it is possessed. This is done to avoid the situation where the robot grabbing the ball cannot perform a kick because it doesn't detect or cannot reach other team players. The robots that do not possess the ball will try to stay at a distance range of 70 cm to 110 cm. The maximum threshold of 110 cm is determined by the distance that the teammates can be perceived and recognized. By determining the minimum threshold of the distance range, it was important to choose a distance not too close to give the robots the time to block the ball.

By combining those four behaviors, the robots were able to play the ball around. In terms of performance, the robots were able to play around while keeping the ball inside the field (a mere 95% of the total game play). A video of this performance is available at the team site[6].

For the competition in Bremen, the current behavior will be extended with a behavior that will direct the robot in possession of the ball back to initial position. To do this, a good absolute localization is needed. Although the performance of our localization algorithm is adequate when the robot is not in possession of the ball (accuracy of 6 cm) [1], this performance drops fast when the robot is handling the ball.

Mantz reported in his thesis [5] that the localization performance can increase considerable when the context of the behavior is taken into account (demonstrated for the behaviors of the goalie). Depending on the context different parameters where used in the localization algorithm. The coming period a study is performed if the same performance increase can be generated for field players that are passing .

The Passing Challenge seems to be applicable for building behavior specific localization algorithms, because in this Challenge there is a default location and heading for the robots. Therefore it is makes sense to build behavior-specific routines that only take one or two percepts as input.

Further, the behavior specific localization algorithms do not have to be limited to small modifications and parameter settings. Behavior specific localization makes it also possible to include complete different localization methods from the currently used Monte Carlo approach (such as SIFT-algorithms [6] or panoramic

[6] http://www.dutchaiboteam.nl/robocup/robocup2006/qualification/video

localization [7]) in specific circumstances, without making the general applicable localization algorithm more and more complex.

3 Open Challenge

In the RoboCup symposium article [7] the Panoramic Localization algorithm is described to get a reliable estimate of the current bearing of the robot. The algorithm distinguishes the appearance of the surroundings in different directions by learning the frequency of the many random color-transitions above the horizon. To be able to learn these color-transitions in natural environments, an automated color clustering algorithm is needed. The details of this algorithm were omitted in the article, but are described in this report. The power of this automated color clustering algorithm was demonstrated during the Dutch Open.

At the competition in Bremen we will demonstrate that the algorithm cannot only be used for a reliable estimate of the bearing, but also for a reliable estimate of the current position. To do this, the frequency of color-transitions has to be learned on multiple spots distributed on the field. Already with 5 learned spots the robot can estimate its position with an accuracy of less than 25 cm, without the use of any artificial landmark. The extension of the Panoramic Localization algorithm to estimate the current position is described in the section after the automated color clustering.

3.1 Automated Color Clustering

Autoshutter To be able to operate in natural environments with a wide variety of lighting conditions, the camera of the Sony Aibo can adjust its hardware settings for the shutter time, the camera gain and the white balance. While our approach has no need for setting the white balance (as we find the most important color by clustering anyway, whatever the white balance might be), the camera gain and the shutter time influence greatly the quality of the images. Too dark images contain noise in all three channels, while too bright images tend to saturate and therefore loose information.

Choosing the right setting is always a compromise. On one hand, we want the dynamic range of the brightness to be as big as possible. To a certain extend, this can be reached by increasing the camera gain, but beyond that we have to increase the shutter time. This, on the other hand, leads to more motion blur in the images and therefore decreases the sharpness.

We experimentally determined that the dynamic range (measured as the distance between the 15-percentile and the 85-percentile of the distribution of the Y channel) of normal images taken under optimal light circumstances is around 100. We now want to determine for each camera setting the dynamic range (starting with slow shutter, high gain and ending with fast shutter, low gain). We require that the dynamic range is at least 80, and of all remaining valid camera settings we choose the one with the highest shutter speed and lowest camera gain (in that order). As it takes a while until new camera settings get

active and the automatic white balance gets stable, we wait for 10 frames before we estimate the dynamic range. Therefore, selecting the best camera settings takes approximately 3 seconds to complete.

Color clustering on the field Now we would like to divide the color space of the camera into 10 characteristic color classes of more or less equal size and distribution. Therefore, the Aibo starts collecting colors by scanning its surroundings. For the localization the Aibo selects colors above the horizon, but for the Dutch Open we showed the result when the colors on the field are collected. If we used all seen colors directly as input for a clustering algorithm, we would end up being capable of discriminating between all shades of green, brown and grey, but we would miss the rather rare but much more characteristic color clusters. So, we use Monte Carlo filter to select characteristic colors. A coarse 3D color space histogram is generated (consisting of 32^3 bins) to estimate the distribution over the color space. We filter the color buffer again by randomly picking colors using the point-wise inverse of the occurrence frequency as probability:

$$P_{use-color-in-clustering}\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \frac{1}{freq_{bin}(Y, Cb, Cr)} \tag{1}$$

We finally end up with (depending on the structure of the color space) approximately 2.000 colors hopefully characteristic and uniformly spread colors from the environment. We feed them into a standard implementation of an Expectation-Maximization algorithm (as described in [8]) in order to find the strongest 10 color clusters (assuming an underlying distribution of a mixture of Gaussians; each one defined by its center, covariance matrix and weight). After 10 iterations we assume the clusters to have emerged and stabilized, and we compute a 6bit color class lookup table [9] for faster access.

$$\begin{aligned} &Expectation: \\ &q_{ns} \leftarrow p(s|x_n) = \tfrac{\pi_s p(x;\theta_s)}{\sum_{s'} p(x;\theta_{s'})} = \tfrac{\pi_s N(x;\mu_s,\Sigma_s)}{\sum_{s'} N(x;\mu_{s'},\Sigma_{s'})} \end{aligned} \tag{2}$$

$$\begin{aligned} &Maximization: \\ &\pi_s \leftarrow \tfrac{1}{N} \sum_{n=1}^{N} q_{ns} \\ &\mu_s \leftarrow \tfrac{1}{N\pi_s} \sum_{n=1}^{N} q_{ns} x_n \\ &\Sigma_s \leftarrow \tfrac{1}{N\pi_s} \sum_{n=1}^{N} q_{ns} \left(x_n - \mu_s\right)\left(x_n - \mu_s\right)^T \end{aligned} \tag{3}$$

As the color class evaluation of the 64^3 voxels is pretty time consuming (because for each cube we need to compute the values of 10 Gaussian distributions), we recursively fill the lookup table from big to small scale and thereby skip the evaluation of presumably homogeneous regions. This means that whenever we encounter that all 8 corners of a (sub-) cube have the same color class assignment,

then we assume this color class for the whole cube. Experimental comparisons have shown no significant difference to a completely evaluated lookup table, but the gain in computation time is enormous: by this approximation we could speed up the evaluation process by factor 20. The result is an unsupervised learned color-table of the 10 most characteristic color clusters in the 3D color space.

When applied to the pixels collected on the field (below the horizon), it became clear that the characteristic colors on the soccer-field are in the normal color space of the Aibo (luminance/chrominance system YCbCr) to close to each other to be automatically clustered. Instead, we converted to another color space (Hue, Saturation, Intensity system HSI). In this color space the pixels collected on the field are nicely distributed, as shown in figure 3.

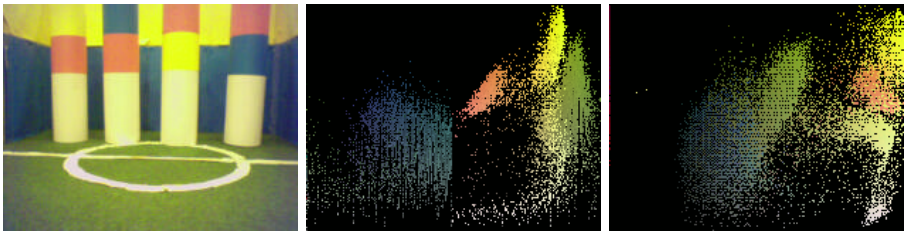

Fig. 3. The pixels collected from a training image (left) in respectively the HS en SI color space

In the HSI color space the characteristic colors of the field could be automatically clustered, as demonstrated in figure 4.

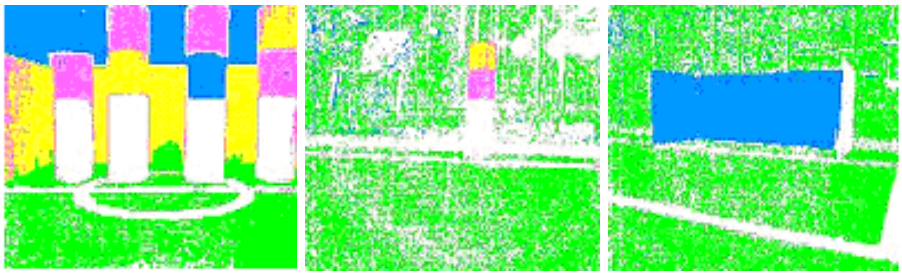

Fig. 4. The segmentation of the color for a training image (left) and two test images on the soccer field

The left image showed the segmentation of the original image used to learn the clusters. The middle and right image shows that the automatically learned cluster can be applied to images not used in the training set. The middle image shows one of the beacons, the right image shows the blue goal. Because no background was visible in the left image, no clusters are available for the background.

Color clustering off the surroundings To learn characteristic colors in the background, the normal YCbCr is sufficient. Yet, the solid surfaces in the background have typically slowly transitioning colors, because the lighting conditions are not as good as on the field, and shadows are quite common. For that reason we apply an additional filter over the pixels before they are offered to the Monte Carlo filter:

When scanning an image, we only store colors whose distance to the next neighboring color (in vertical direction) exceeds a certain threshold value. The metric we use is a Manhattan distance [10] modified in such a way that blue-green and white-yellow transitions in the YCbCr color-space are amplified (which are naturally hard to discriminate with on Aibo images). The threshold value is the dynamic estimate of the 95-percentile of all measured color distances, therefore assuming that approximately one of 20 pixels is situated on an edge. In this way, only the (hopefully clearer) near-edge colors are used instead of the slowly transitioning colors of solid surfaces (due to shadows etc).

$$\left\| \begin{matrix} Y_1 & Y_2 \\ Cb_1\,, & Cb_2 \\ Cr_1 & Cr_2 \end{matrix} \right\| = \|Y_1 - Y_2\| + \|Cb_1 - Cb_2\| + \|Cr_1 - Cr_2\| + \|\|Cb_1 - Cb_2\| - \|Cr_1 - Cr_2\|\| \tag{4}$$

Figure 5 shows exemplary the result of the automatic color clustering process when only color-transitions above the horizon were taken into account for clustering. The strongest 6 clusters contain different shades of the background color (different shades of brown), while the remaining 4 contain (at least from the point of view of a human observer) contain the interesting colors (blue, yellow, pink and dark blue). Note the green of the field is excluded because that only colors above the horizon are taken into account; otherwise green would have been the strongest cluster.

3.2 Position Estimation with Panoramic Localization

The algorithm described in [7] can be used to get a robust bearing estimate together with a confidence value for a previously trained spot. As we finally want to use this algorithm to obtain full localization we extended the approach to support multiple training spots. The main idea is that the robot determines to which amount its current position resembles with the previously learned spots and then uses interpolation to estimate its exact position. As we think that this approach could also be useful for the RoboCup @ home-league (where robot localization in complex environments like kitchens and living rooms is required)

Fig. 5. Unsupervised color clustering.

it could become possible that we finally want to store a comprehensive panorama model library containing dozens of previously trained spots (for an overview see part II of [4]).

However, due to the computation time of the feature space conversion and panorama matching, per frame only a single training spot and its corresponding panorama model can be selected. Therefore, we developed a grid-based Monte Carlo filter that randomly selects one of the stored panorama models for image matching.

$$P(select = Spot_i) = \frac{min(0.05, confidence_i)}{\sum_j confidence_j} \tag{5}$$

Every panorama model is associated with a gradually changed confidence value representing a sliding average on the confidence values we get from the per-image matching. To compute the final position estimate, we simply weight each training spot with its corresponding confidence value:

$$position_{robot} = \sum_i position_i \frac{confidence_i}{\sum_j confidence_j} \tag{6}$$

To prove the validity of this idea, we trained the robot on five spots on regular 4-Legged field in our robolab. The first training spot was in the center of the field, while the other four were located along the axes approximately 1.5m away from the center. The training itself was performed fully autonomously by the Aibo and took less than 10 minutes. After training was complete, the Aibo walked back to the center of the field. We recorded the found position and kidnapped the robot to an arbitrary position around the field and let it walk back again. Figure 3.2 shows the result of this experiment. It can be seen that the localization is not as accurate as traditional approaches, but can still be useful for some applications (bearing in mind that no field or landmarks are required). We recorded repeatedly a derivation to the upper right that we think can explain by the fact that different learning spots don't produce equally strong confidence value; we believe to be able to correct for that by means of confidence value normalization in the near future.

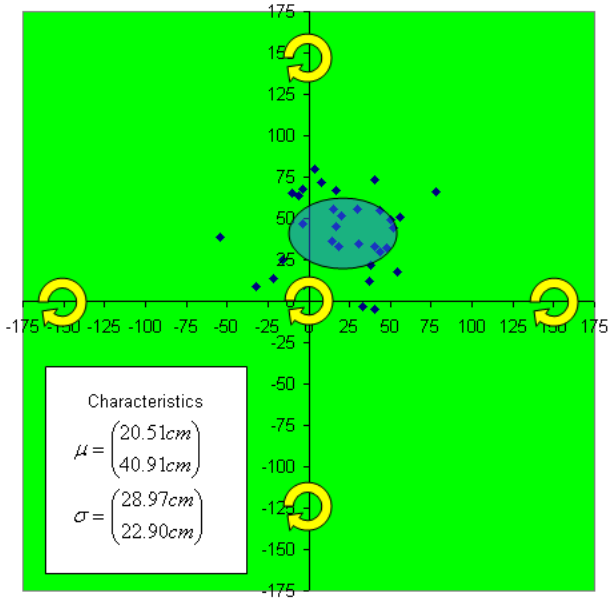

Fig. 6. The position estimation after learning the appearance of the surroundings on 5 spots (indicated with yellow arrows). The small dots indicate the distribution of positions reached every time the robot tried to return to the central spot after been placed randomly somewhere on the field.

4 Conclusion

We look forward to the RoboCup 2006 to demonstrate our ideas and learn from our competitors. We hope that our effort to operate under a wide variety of lighting conditions combined with the use of natural landmarks facilitates the advancement of mobile robots - and thereby robotics research itself - into more natural environments.

Acknowledgements

The Dutch AIBO Team is financially supported by DECIS Lab, University of Amsterdam, Technical University of Delft, Saxion University of Professional Education, University of Groningen and University of Utrecht. We are grateful for the contributions by all of our team members, working on RoboCup related research and other research. Special thanks to Floris Mantz, Bayu Slamet, Isaac Esteban, Areej Mahdi, Rutger Vlek, Gert Kootstra, Andrew Koster, Jeff Ouwerkerk, Rico Slagmolen and Niek Wijngaards.

References

1. Sturm, J., Visser, A., Wijngaards, N.: Dutch aibo team: Technical report robocup 2005. Technical report, Dutch Aibo Team (2005)
2. Touretzky, D., Tira-Thompson, E.: Tekkotsu: a sony aibo application development framework. The Neuromorphic Engineer **1**(2) (2004) 12
3. Mahdi, A., de Greef, M., van Soest, D., Esteban, I.: On joint actions for an aibo team. Technical report, Universiteit van Amsterdam (2006)
4. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. Intelligent robotics and autonomous agents. The MIT Press, Cambridge, MA (2005)
5. Mantz, F.: A behavior-based vision system on a legged robot. Master's thesis, Delft University of Technology (2005)
6. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (2004) 91–110
7. Sturm, J., van Rossum, P., Visser, A.: Panoramic localization in the 4-legged league. In: Proc. 10th RoboCup International Symposium, Bremen (2006) To be published in the Lecture Notes on Artificial Intelligence series, Springer Verlag, Berlin.
8. Verbeek, J.: Mixture models for clustering and dimension reduction. PhD thesis, Universiteit van Amsterdam (2004)
9. Nisticó, W., Röfer, T.: Improving percept reliability in the sony four-legged league. In: RoboCup 2005: Robot Soccer World Cup IX. Lecture Notes in Artificial Intelligence, Springer (2006)
10. Sridharan, M., Stone, P.: Real-time vision on a mobile robot platform. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. (2005)