Behavior-Based Vision on a 4 Legged Soccer Robot

Floris Mantz, Pieter Jonker, and Wouter Caarls

Quantitative Imaging Group, Faculty of Applied Sciences,
Delft University of Technology, Lorentzweg 1, 2628 LC Delft,
The Netherlands
{floris, pieter, wouter}@ph.tn.tudelft.nl

Abstract. In this paper the architecture of a 4 legged soccer robot is divided into a hierarchy of *behaviors*, where each behavior represents an independent sense-think-act loop. Based on this view we have implemented a *behavior-based vision system,* improving performance due to *object-specific* image processing, *behavior-specific* image processing and *behavior-specific* self localization. The system was tested under various lighting conditions, off-line using sets of images, and on-line in real tests for a robot in the role of goalkeeper. It appeared hat the performance of the goalie *doubled*, that it could play under a wider range of lighting and environmental conditions and used less CPU power.

1 Introduction

Soccer playing robots are *the* playground to gain experience with embodied intelligence. The software architectures of those robots can serve as an example for more complex embedded systems. Those designs are often built around a single sense-think-act cycle, as presented in Fig. 1a. Here, e.g. an image processing group solves the *sense* task, the control theory group solves the *act* task, an AI group solves the *think* task, whereas software and mechanical engineers take the responsibility over the overall software and mechanical hardware design and maintainability. After an initial architecture phase, interfaces are quickly established and all groups retract to their own lab to solve their part of the problem, thereby often making assumptions what is or should be done by the others. Data is "thrown over the wall" to the others who have to cope with it. As those embedded systems increase in complexity over the years, the software and hardware complexity grows, and all groups start to make their system part versatile and robust and hence complex. Then, to cope with that, they start to locally optimize their parts, making sources even more obscure. From 1991 onward it was suggested [1,2,3] that a different layered modular architecture should be followed in the sense that higher layers control the lower layers, by invocation actions from the lower layers or by promoting or suppressing behaviors from the lower layers, i.e., on the lowest level on the hardware devices. To program all behaviors of a system that - without tedious recalibration - functions under al circumstances on any setting, is like maintaining a house of cards. One cannot foresee all possible states in which the system might end up in the future.

With growing complexity of the modules and the limited time students have for their graduation, RoboCup students mostly do not understand the full complexity of

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 480 – 487, 2006.

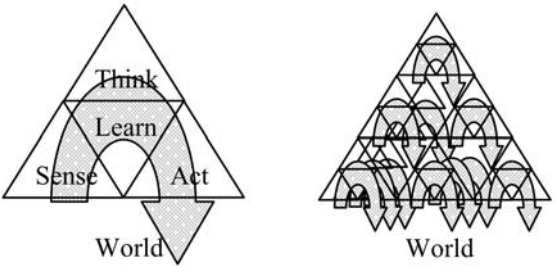


Fig. 1. Architecture based on: a) scientific disciplines b) required behaviors

the existing modules; so often their new software ruins or hinders good functionality elsewhere in the code, they start anew from scratch forgetting past lessons, or they build extra features from which the impact on the total system is not and cannot be overseen. In industrial embedded system projects, this is often not different! To cope with this, we have set-up a new software architecture for the Dutch AIBO Team (DAT) based on a hierarchy of modules in which each module is a separate relatively simple sense-think-act loop. See Fig. 1b. In the end we aim for a pluggable architecture in which it is relatively simple for students to understand the modules, to replace them with better ones or to add functionality by adding modules. The drawback of this approach is that we need students that need to understand the principles of image processing; mechanical engineering, control theory, AI *and* software engineering.

2 Behavior Based Vision

2.1 Advantages and Expected Performance Improvements

In this paper we show that a hierarchy of simple sense-think-act modules (Fig. 1b) performs better than an architecture based on monolithic modules (Fig1.a). To prove this, we changed the software of the goalkeeper of the Dutch AIBO team 2004 (DT2004) [10] and measured the differences with the NEW software for our goalie. As benefits we expect:

1. That each (sense-think-act) module is simpler and hence can be better understood and used to design new behaviors (copy-past-modify).
2. That effectively less and less complicated code is running in the new situation. We proved that this is true and that we made CPU cycles free.
3. That our goalkeeper performs better and more robust. The new software prevented more goals then the old software and, moreover, under a variation of lighting conditions. This was due to the capability of the goalie to localize itself better and more robust. And this was due to:
    a. Behavior specific self localization.
    b. Behavior specific image processing, or:
        i. Only search objects that you need to see for your task
        ii. Only search for objects at the place you expect them to see

c. Object specific image processing, or:
   i. process them using the shape knowledge over that object
   ii. process them using the color knowledge over that object.

2.2 Behavior Specific Processing Tools

To prove hypotheses 3 we used behavior-based vision to realize a goalie that can perform better and more robust. Fig. 2 is an implementation of Fig. 1b applied to the goalkeeper. For this role we can identify its basic behaviors. These are controlled by a meta-behavior that may invoke them. We will call this the governing behavior.

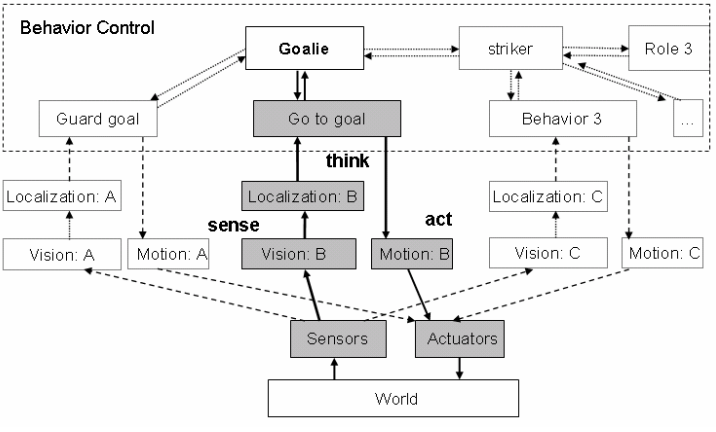


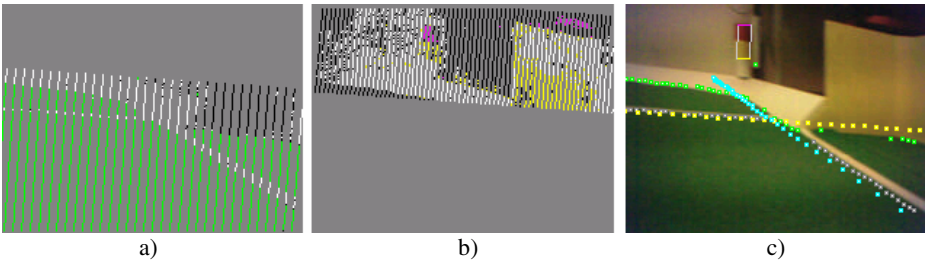
Fig. 2. Cut-out of the hierarchy of behaviors of a soccer AIBO, with emphasis on the goal-keeper role. Each behavior (e.g. go to goal) is an independently written sense-think-act loop.

Fig. 2 also shows that the cognition system of the robot is split into a vision system using grids and color tables [5] and a Monte Carlo self locator [6, 7] or particle filter. We have implemented our behavior-based vision on a Sony AIBO ERS-7 of the Dutch AIBO Team (DT2004) [4].We have implemented a behavior based architecture [8] by adding the following features:

- ***Behavior-specific self localization***
  When a robot is positioning (e.g. a goalie standing in the goal, or a field player walking around), the sensor input is qualitatively high and accurate localization is our aim; hence we use a fast update of the particles. When a robot is handling a ball, the sensor input is qualitatively low and the updating of the robot's pose is less urgent; hence we use a slow update of the particles.
- ***Behavior-specific image processing***
  Per behavior we only execute the image processing algorithms for detecting objects that are most likely to be seen, or that can be measured with high accuracy. A goalie guarding its goal, can most accurately and likely detect the lines of the goal area and its own corner flags. These objects are near and it is likely that they will not be occluded by opponents, which is the case for the opponent's goal and flags.

The drawback of this is that the robot can get stuck in a local loop, when making a wrong assumption about its position. A background process (at a lower pace) must detect this.

- ***Object-specific image processing***
  We use the type of objects, (goals, flags), color of objects (blue/yellow goal), and size of objects (far/near flag). For each type of object we define a *specific grid*, *color-table* and *specific size*. For each object we use a *specific color look-up table* (CLUT). CLUTs have to be calibrated [9]. Here we only calibrated the CLUT for the 2 or 3 colors necessary for segmentation. This greatly reduces the problem of overlapping colors. For each object we know its *preferred size* in order to be of interest. E.g., we might in some circumstances not be interested in a corner flag far away.



a) b) c)

Fig. 3. Object-specific image processing: a) for line detection we scan the image below the horizon, using a green-white color table; b) for yellow flag detection we scan above the horizon using a yellow-white-pink color table; c) 2 lines and 1 flag detected in the image

2.3 The Basic Behaviors of a Goal Keeper and Its Behavior Based Processing

For the goalkeeper role we can identify 3 basic behaviors. They are:

- ***Goalie-return-to-goal.*** When the goalie is not in his goal area, he has to return to it. The goalie walks around scanning the horizon. When he has determined his own position on the field, the goalie tries to walk straight back to goal - avoiding obstacles - keeping an eye on his own goal. The localization relies on the detection of the own goal, detected line-points and detected border-points (no Hough-transform is used). The two own corner flags are also used for localization. All sensor input is used in a particle filter in which a detected own goal is used twice when updating the particles.
- ***Goalie- position.*** The goalie is in the centre of its goal when no ball is near. It sees the field-lines of the goal area often and at least one of the two nearest corner flags regularly. Hough transforms on detected line-points are used to calculate the distance and angle to the field lines. Detection of the own flags is based on a grid above the horizon, a 3-color LUT, and rejecting candidates that are too small. An orange/white/green color table was used for ball- and line detection. A particle filter was used that localized only on the detected lines and flags. A background process verifies the “in goal” assumption on the average number of detected lines and flags.

a) b) c)

Fig. 4. Basic goalie behaviors: a) **Goalie-return-to goal,** b) **Goalie-position**, c) **Goalie-clear ball**. For each behavior a different vision system is used and a different particle filter setting.

- ***Goalie-clear-ball.*** When the ball comes near, the *goalie-clear-ball* behavior is switched on, meaning that the goalie is defending the line between ball and the center of the goal. The closer the ball comes to the goal area, the less time there is for the goalie to verify with the vision system where the center of the goal is. The goalie will gradually rely more on odometry than on vision. If the ball enters the goal area, the goalie will clear the ball. Walking to and controlling the ball, the quality of the vision input is very low. Although the same image processing is used as in *goalie-position-behavior,* the particles in the self-locater are updated much slower. Flags and lines detected at far off angles or distances are ignored.

3 Performance Measurements

3.1 General Setup of the Measurements

To prove our hypothesis 3, we have performed measurements on the behavior of our new goalie. Our test is twofold:

1. How fast can the new goalie find back his position in the middle of the goal on a crowded field in comparison with the old goalie
2. How many goals can the new goalie prevent on a crowded field within a certain time slot in comparison with the old goalie

As our improvements are due to three measures we have taken, we would like to know the contribution of each of the measures to the final result, i.e.;

a. Object-specific image processing
b. Behavior-specific image processing
c. Behavior-specific self localization.

3.2 Influence of Object-Specific Image Processing

We compared the DT2004 code with a general version of our NEW code. The latter does not (yet) use behavior specific image processing nor self-localization. However, it does use object specific grids and color tables. The tests consisted of searching for

the goals, the flags, and all possible line- and border-points. The images were captured with the robot's camera, under a large variety of lighting conditions. For the DT2004 code, a single general CLUT was calibrated for all colors that are meaningful in the scene (blue, yellow, white, green, orange and pink) taking 3 hours. For the NEW image processing code we calibrated five 3-color CLUTs (white-green lines, blue-goal, blue-flag, yellow-goal, yellow-flag). This took 1 hour for all tables.

For all image sequences we counted the number of objects detected correctly (*N true*) and falsely (*N false*). We calculated the correctly accepted rate (CAR), being the number of objects that were correctly detected divided by the number of objects that were in principle visible. Table 1 shows the results on detecting flags and lines. It shows that the performance of the image processing largely increased. The correctly accepted rate (CAR) goes up from about 45 % to about 75%, while the number of false positives is reduced. The correctly accepted rate of the line detection even goes up to over 90%, also when a very limited amount of light is available.

Table 1. The influence of object-specific algorithms for goal, flag and line detection

Goals and flags	DT2004			NEW			DT2004	NEW
	N true	CAR (%)	N false	N true	CAR (%)	N false	Lines (%)	Lines (%)
1 flood light	23	19	0	65	54	0	18	94
Tube light	54	45	9	83	83	1	58	103
4 flood lights	86	72	0	99	99	0	42	97
Tube +flood lights	41	34	1	110	92	0	24	91
Tube,flood+natural	39	33	0	82	68	0	42	91
Natural light	47	39	0	68	57	0		
Non calibration set	131	44	28	218	73	16		

3.3 Influence of Behavior Based Vision

Below we show the performance improvement due to behavior based switching of the vision system **and** the self localization algorithm. We used the following scenarios:

- *Localize in the penalty area.* The robot is put into the penalty area and has to return to a predefined spot as many times as possible within 2 minutes.
- *Return to goal.* The robot is manually put onto a predefined spot outside the penalty area and has to return to the return-spot as often as possible within 3 minutes.
- *Clear ball.* The robot starts in the return spot. The ball is put in the penalty area each time the robot returns. It has to clear the ball as often as possible in 2 minutes.
- *Clear ball with obstacles on the field.* As above but with many strange objects and robots placed in the playing field, to simulate a more natural playing environment.

To distinguish between the performance increase due to object-specific grids and color-tables, and the performance increase due to behavior-dependent image processing and self localization, we used 3 different configurations, resulting in Figs 5 and 6:

– *DT2004:* The old image processing code with the old general particle filter.
– *Striker:* The new object-specific image processing used in combination with the old general particle filter of which the settings are not altered during the test.
– *Goalie*: The new object-specific image processing used in combination with object-specific algorithms as well as with a particle filter of which the settings are altered during the test, depending on the behavior that is executed.

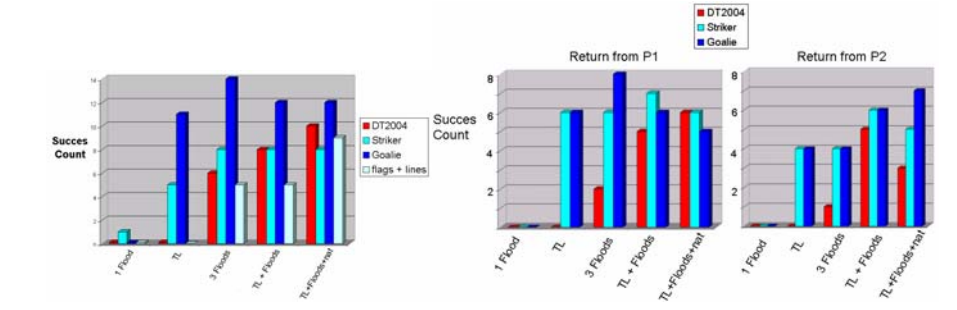


Fig. 5. Left. *Results for localization in the penalty area*. The number of times the robot can relocalize in the penalty area within 2 minutes. The old DT2004 vision system cannot localize when there is little light (TL). The performance of the object specific image processing is shown by the "flags and lines" bars. In contrast with the DT2004 code, the striker uses object specific image processing. The goalie uses both object specific image processing *and* behavior based vision processing *and* behavior based self localization. **Right.** Results of the return to goal test. The robot has to return to its own goal as many times as possible within 3 minutes. The striker vision systems works significantly better than the DT2004 vision system. There is not a very significant difference in overall performance between the striker (no behavior-dependence) and the goalie (behavior dependence).

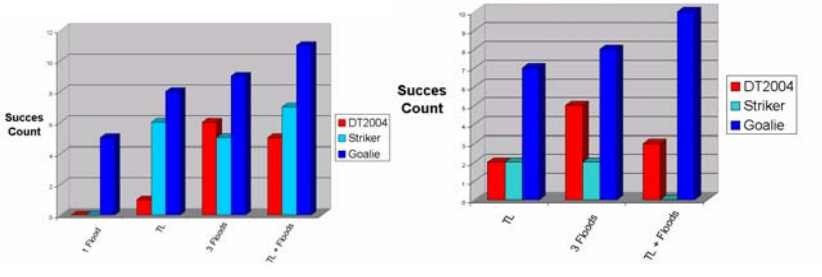


Fig. 6. Left. Results of the clear ball test. The robot has to clear the ball from the goal area as often as he can in 2 minutes. Both the striker and the goalie vision systems are more robust in a larger variety of lighting conditions than the DT2004 vision system (that uses a single color table). The goalie's self locator, using detected lines and the yellow flags, works up to 50 % better than the striker self locator, which locates on all line-points, all flags and goals. **Right.** Results of the clear ball with obstacles on the field test. The goalie vision system, which uses location information to disregard blue flags/goals and only detects large yellow flags, is very robust when many unexpected obstacles are visible in or around the playing field.

4 Conclusions

- With object-specific grids and color-tables, the performance of the image processing (reliability) under variable lighting conditions has increased with 75-100%, while the color calibrating time was reduced to 30%.
- The impact of behavior-specific image processing and self localization can be seen from the localization test in the penalty area. The vision system of the goalie, with behavior based vision and self-localization, performs 50 % better on the same task as a striker robot with a vision system without behavior based vision and self localization. Note that both do use object specific image processing in this case.
- The impact of behavior-specific image processing on the reliability of the system can clearly be seen from the clear ball behavior test with obstacles on the field. With 1 flood light, where none of the robots can detect flags, the goalie can still clear a number of balls before being lost.
- Using object specific image processing reduced the CPU load with 50%. Using only specific algorithms as is done in behavior based vision and localization, then it is possible to decrease the load to about 10% of that in the old DT2004 code.
- Due to the new architecture, the code is more clean and understandable; hence better maintainable and extendable. The price is that one has to educate system engineers instead of sole image processing, software, AI, and mechanical experts.

References

1. R.A. Brooks. Intelligence without Representation. Artificial Intelligence, Vol.47, 1991, pp.139-159.
2. R.C. Arkin. Behavior based robotics, MIT press 19989, ISBN 0-262-01165-4
3. Parker, L. E. (1996). On the design of behavior-based multi-robot teams. Journal of Advanced Robotics, 10(6).
4. Stijn Oomes, P.P. Jonker, Mannes Poel, Arnoud Visser, and Marco Wiering, The Dutch AIBO Team 2004, Proc. Robocup 2004 Symposium (July 4-5, Lisboa, Portugal, Instituto Superior Tecnico, 2004, 1-5. see also http://aibo.cs.uu.nl
5. J. Bruce, T.Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00), volume 3, pages 2061-2066, 2000.
6. S.Thrun, D.Fox, W. Burgard, and F. Dellaert, Robust monte carlo localization for mobile robots. Journal of Artificial Intelligence, Vol. 128, nr 1-2, page 99-141, 2001, ISSN:0004-3702
7. T. Rofer and M. Jungel. Vision-based fast and reactive monte-carlo localization. In The IEEE International Conference on Robotics and Automation. In The IEEE International Conference on Robotics and Automation, pages 856-861, Taipei, Taiwan, 2003.
8. Scott Lenser, James Bruce, Manuela Veloso, A Modular Hierarchical Behavior-Based Architecture, in A. Birk, S. Coradeschi, and S. Tadokoro, editors, RoboCup-2001: The Fifth RoboCup Competitions and Conferences, Springer Verlag, Berlin, 2002.
9. Jüngel, M. (2005). Using Layered Color Precision for a Self-Calibrating Vision System. In: 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences, lecture Notes in Artificial Intelligence. Springer (to appear).