Behavior-Based Vision on a 4 Legged Soccer Robot

Floris Mantz, Pieter Jonker, 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 as can be found in the RoboCup [1] are *the* playground to gain experience with embodied intelligence. The software architectures of those robots - that can autonomously survive in a niche of the real physical world; with limited rules necessary to survive, limit physical circumstances to account for and simple goals to achieve [2] - can very well serve as an example for more complex industrial machines such as photocopiers, wafer steppers, component placement machines, CT and MRI scanners. The problem with all those machines is that they are built around a sense-think-act cycle which uses the machine to operate in the physical world to perform its task, but that it is quite common that several discipline groups with various expertises cooperate in the design. As a consequence, the most obvious basic architecture is the one as presented in figure 1a, in which for instance 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 engineers and mechanical engineers take the responsibility over the overall software and mechanical hardware design and maintainability. After an initial limited architecture phase the interfaces are quickly established and all groups retract to their own lab to locally optimize their part of the problem, thereby often making assumptions what is or should be done by the other group. In the end, the data is "thrown over the wall" to the other groups who have to cope with it. As those embedded machines increase in complexity over the years, as well as the demands from the world they are operating in, the software and hardware complexity grows, and all groups start to make their vision/motion/AI system most versatile and

robust and hence increasingly complex and, to partly cope with that, they start to locally optimize their parts, making their sources even more obscure. Already from 1991 onward it was suggested [3,4,5] that a different architectural concept should be followed in the sense that a layered modular architecture should be set-up in which higher layers control the lower layers, either by invocation actions from the lower layers or by promoting or suppressing behaviors from the lower layers that in principle act in parallel on their turn on modules lower in the abstraction hierarchy, e.g., on the lowest level on hardware devices such as the robot's motor currents.

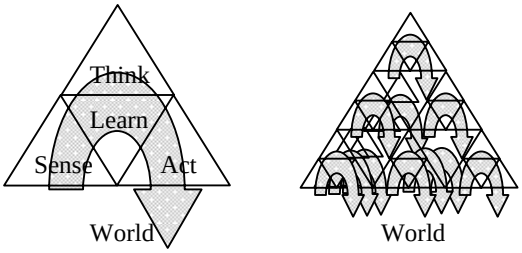


Fig. 1. architecture based on: a) scientific disciplines b) required behaviors

To program all behaviors of the system that - without tedious recalibration of the system - 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. For this, many started to use learning, e.g. based on reinforcement [6, 7] to overcome this situation. However, when the dimension of input-action space becomes too high (>8) this approach becomes cumbersome [8]. Hence, we should also aim for a layered modular architecture in which we can learn basic behaviors such as dribbling with the ball, collision avoidance or a shoot to goal reflex, in combination with meta level learning to coordinate this [9]. So this strengthens our argument.

RoboCup software is mostly "studentware". With growing complexity of the modules and the limited time students have for their graduation, students mostly do not entirely understand the full complexity of 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. Beware; in industrial projects e.g. in embedded systems, this is often not different! To cope with these problems, we have tried to set-up a new software architecture for our Dutch AIBO Team (DAT) based on a hierarchy of modules in which each module is a separate relatively simple sense-think-act loop. See figure 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 an architecture of a hierarchy of simple sense-think-act modules (fig. 1b) performs better than our old architecture based on monolithic discipline based modules (fig1.a). Note, that we omitted learning here, as this complicates our proof. To prove that the new architecture is better, we changed the software of the goalkeeper of the Dutch AIBO team 2004 software (DT2004) [10] and measured the differences with the NEW software for our goalie. Basically, as benefits we expect:

1. That each (sense-think-act) module is simpler and hence can be better understood and be used to design other behaviors (copy-past-modify) by new students. Although we found this to be true, this cannot be proven easily.
2. That effectively less code, and less complicated code is running in the new situation then it was in the old situation. We proved that this is true and that we made CPU cycles free, with which it is possible to make our behaviors more robust, e.g. by using more complex algorithms. The crux in this is that in the old situation the code that was running not always contributed to the behavior, but was merely there for "general-purposeness".
3. That our goalkeeper performs better and more robust. We found that this was true. The new software prevented more goals then the old software and, moreover, under a variation of lighting conditions. That it did so 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, or: if you know what you are doing, ignore landmarks that have nothing to do with your task and/or consider them at a lower pace.
    b. Behavior specific image processing, or:
        i. Only search objects that you need to see for your task and ignore the rest
        ii. Only search for those objects at the place you expect them to see; e.g. the ball (near) or a corner-post (at the horizon)
    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

In order 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 role of a 4 legged soccer robot. For the goalkeeper role of the robot we can identify his basic behaviors. These behaviors are controlled by a meta-behavior (*Goalie* in fig. 2) that may invoke them. We will call this meta-behavior the goalie's governing behavior.

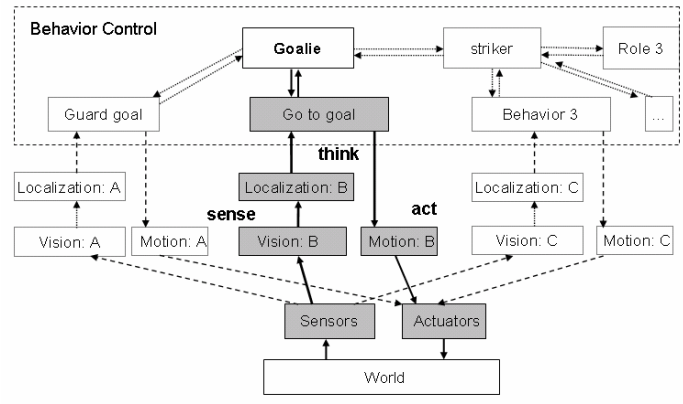


Fig. 2. Cut-out of the hierarchy of behaviors of a soccer robot, with emphasis on the goalkeeper role. Each behavior (e.g. go to goal) is an independently written sense-think-act loop.

Fig. 2 also shows that the cognition system of the robot is split into a vision system using grids and color tables [12] and a Monte Carlo self locator [13, 14, 15] or particle filter. We have implemented our behavior-based vision on a Sony Aibo ERS-7. We used the code of the Dutch Aibo Team (DT2004) [10], which was adapted in 2004 from the code of the German Team of 2003 (GT2003) [11]. These codes feature grid-based image processing, Monte Carlo self localization [19], an XABSL state machine for behavior control [16], an inverse kinematics walking engine, and special actions for motion control. Based on the DT2004 code, we have implemented a behavior based architecture [17, 18] now including behavior-based vision by adding the following features:

- ***Behavior-specific self localization***

With particle filters for self localization, one always has to make a trade-off between *robustness* and *speed*. If the particles are updated slowly on new sensor inputs, the system is more robust against false sensor inputs. If the particles are updated fast, the system can be accurate despite un-modeled movements, such as uncertainty in odometry evaluation, collisions, or a pickup (kidnap) by the referee. In our set-up, 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.

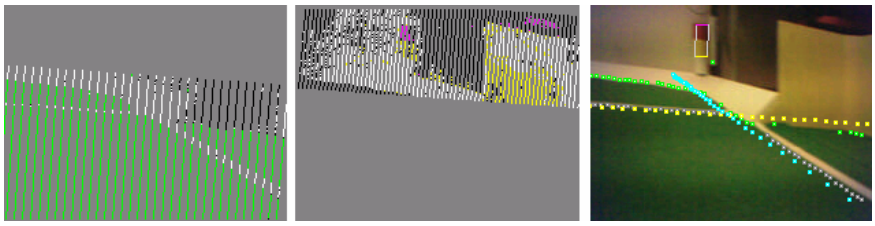
- ***Behavior-specific image processing***

What the robot can see depends on the robot's location. The robots location is strongly correlated with its behavior. 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. For example, a goalie guarding its goal, can most accu-

rately and likely detect the lines of the goal area and its own corner flags. These objects are near (i.e. accurate) and it is highly likely that they will not be occluded by opponents, which is the case for the opponent's goal and flags. The drawback of using behavior-coupled location dependence is that the robot will possibly enter local loops, when making a wrong assumption about its position. A guarding background process (at a lower pace) must detect these loops.

•***Object-specific image processing***

In other to enable behavior-dependent image processing, we have split up the vision system into a separate function per object to detect. Within this, we use as parameters the types 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 example, for detecting a yellow/pink flag (fig. 3), the image is scanned only above the horizon, limiting the used processing power and reducing change on error. For each object we use a *specific color-table* (CLUTs). In general CLUTS have to be calibrated [19]. Here we only calibrated the CLUT for the 2 or 3 colors necessary for segmentation. This procedure greatly reduces the problem of overlapping colors. Moreover, for each object we know a *preferred size* that it should have 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.

2.2. The basic behaviors of a goal keeper and its behavior based processing

For the goalkeeper role of the robot we can identify three basic behaviors. These behaviors are controlled by a meta- or governing behavior that may invoke them. The three basic behaviors of the goal keeper role 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 with its head-camera. 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 in this behavior mainly relies on the detection of the own goal, detected line-points and detected border-points (no Hough-transform is used). To overcome the situations when the

own goal is not visible from some places in the field, the two own corner flags are also used for localization. All sensor input is processed in a particle filter, almost identical to the one used in the old DT2004 software, except that a detected own goal is used twice for updating the particles. When the goalie returned in his goal, he is put (via its governing behavior) into the behavior *goalie-position*.

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- position. The goalie stands in the centre of his goal when no ball is near. It moves its head around and thus is likely to see the field-lines of the goal area very often and at least one of the two nearest corner flags once in a while. Hough transforms are used on detected line-points to calculate the distance and angle (R,θ) to the field lines. An image processing algorithm using a grid above the horizon, and a 3-color look-up table while rejecting candidates that are too small, was used for detecting the own flags. An orange/white/green color table was used for ball- and line detection. A particle filter was used in such a way that it localized only on the detected lines and flags. As all these measurements heavily rely on the assumption that the goalie is in his goal. Hence, a background process on a slower pace keeps verifying if this assumption is still valid. For this, the average number of detected lines and flags is evaluated. If this value is too low, the behavior signals to the governing that the pre-condition is false (the goalie is likely to be located outside the goal area), and the governing behavior switches, e.g., to the behavior *goalie-return-to-goal.*

-Goalie-clear-ball. When the ball comes clearly into sight of the goalie, the *goalie-clear-ball* behavior is switched on too by the goalies governing behavior. This means that the goalie is defending the line between ball and the center of the goal. It keeps its nose towards the ball and its rear towards where it thinks the center of the goal is. 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 really is. The goalie will gradually rely more on the odometry than on the vision. If the ball enters the goal area, the goalie will clear the ball from the area, return to the center of the goal and return (through the governing behavior) to the behavior *goalie-position*. While walking to the ball the goalie's head is aimed at the ball; when controlling the ball, the head is positioned over the ball. In these situations the quality of the vision input is very low. Although the same image processing solution is used as for *goalie-position-*

behavior, detecting the lines and near flags, the particles in the particle filter of the self-locater are updated much slower. Flags and lines detected at far off angles or distances are totally ignored. The algorithm that detects whether the premises for "I am in the goal" is still valid, runs on a lower pace; so the robot longer works with the assumption that he is standing between ball and goal, without actually verifying this.

3. Performance Measurements

3.1. General setup of the measurements

In order to prove our hypothesis 3 (section 2.1), we have performed measurements on the behavior of our new goalie. The performance is commonly evaluated in terms of accuracy and/or reactiveness of localization, in test environments dealing with noisy (Gaussian) sensor-measurements [15, 19]. We, however, are interested also in terms of the system's reliability when dealing with more serious problems such as large amounts of false sensor data input, or limited amounts of correct sensor input.
Then the ultimate test is how much goals does the new goalie prevent under game conditions in comparison with the old goalie? Due to the hassle and chaotic play around the goal when there is an attack, the goalie easily loses track of where he is. So our ultimate test is now 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 have compared the original DT2004 image processing with a general version of our NEW image processing; meaning that the latter does not (yet) use behavior specific image processing nor self-localization. In contrast with the DT2004 code, the NEW approach does use object specific grids and color tables. Our tests consisted of, searching for the 2 goals, the 4 flags, and all possible line- and border-points. The images sequences were captured with the robot's camera, under a large variety of lighting conditions. See fig. 5. A few images from all but one of these lighting condition sequences were used to calibrate the Color-Lookup Tables (CLUTs). For the original DT2004 code, a single general CLUT was calibrated for all colors that are meaningful in the scene, i.e.: blue, yellow, white, green, orange and pink. This calibration took three hours. For the NEW image processing code we calibrated five 3-

color CLUTs (for the white-on-green lines, blue-goal, blue-flag, yellow-goal, yellow-flag respectively). This took only one hour for all tables, so 30% of the original time.

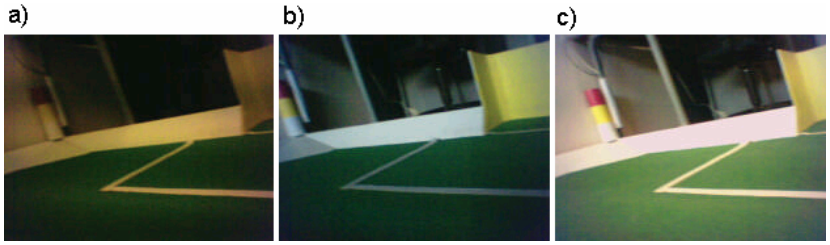


Fig. 5. Images taken by the robots camera under different lighting conditions: a) Tube-light; b) Natural light; c) Tube light + 4 floodlights + natural light

For all image sequences that we had acquired, we have counted the number of objects that were detected correctly (*N true*) and detected falsely (*N false*). We have calculated also 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 due to using object specific grids and color tables, 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. Moreover, it takes less time to calibrate the color tables. 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. The old DT2004 image processor uses a general grid and a single color table, the NEW modular image processor uses object-specific grids and color-tables per object. The calculation of the correctly accepted rate is based on 120 flags hat were in principle visible in the first 5 image sequences and 360 flags in principle visible in the set where no calibration settings were made for. The image sequences for line detection each contained on average 31-33 line-points per frame.

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

In the previous tests we have shown the improvement due to the use of object specific grids and color tables. Below we show the performance improvement due to behavior

based switching of the vision system **and** the self localization algorithm (the particle filter). We used the following scenarios which results can be found in figs. 6-9:

- 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 manually put in the penalty area every time the robot is in the return spot. It has to clear the ball as often as possible in 2 minutes.
- Clear ball with obstacles on the field. We have repeated the clear ball tests but then with many strange objects and robots placed in the playing field, to simulate a more natural playing environment.

In order to be able 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.

– *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.

The results can be found in the figures below:

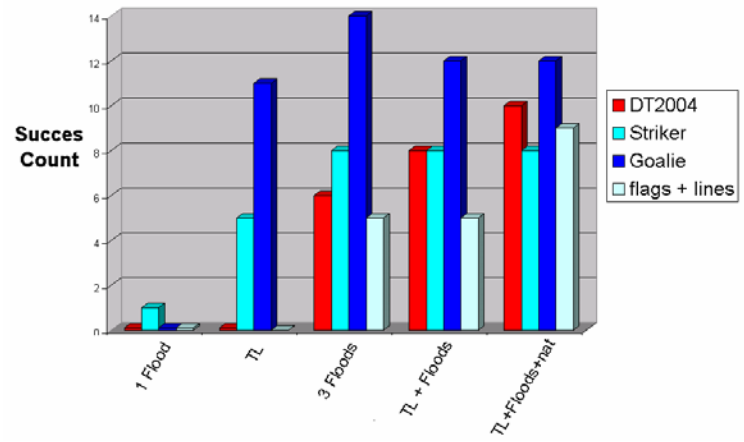


Fig. 6. Results for localization in the penalty area. The number of times the robot can re-localize in the penalty area within 2 minutes. The old DT2004 vision system cannot localize when there is little light (TL). The performance of the object specific image processing is shown by the "flags and lines" 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.

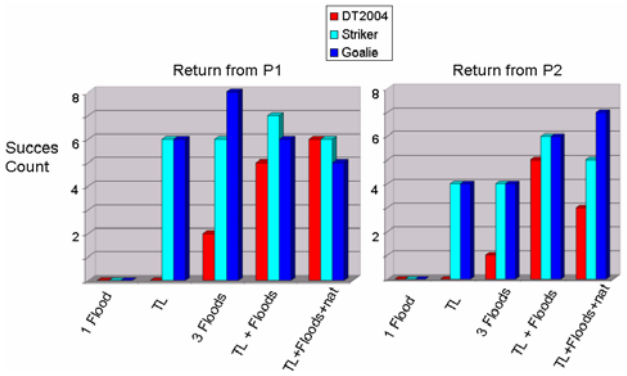


Fig. 7. Results of the return to goal test. The robot has to return to its own goal as many times as possible within 3 minutes. The striker vision systems works significantly better than the DT2004 vision system. There is not a very significant difference in overall performance between the striker (no behavior-dependence) and the goalie (behavior dependence)

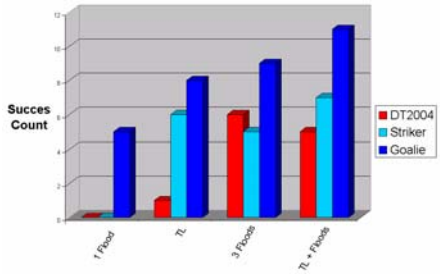


Fig. 8. Results of the clear ball test. The robot has to clear the ball from the goal area as often as he can in 2 minutes. Both the striker and the goalie vision systems are more robust in a larger variety of lighting conditions than the DT2004 vision system (that uses a single color table). The goalie's 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.

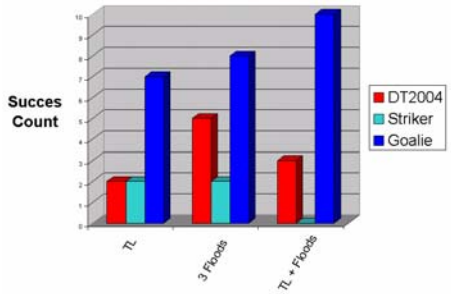


Fig. 9. Results of the clear ball with obstacles on the field test. The goalie vision system, which uses location information to disregard blue flags/goals and only detects large yellow flags, is very robust when many unexpected obstacles are visible in or around the playing field.

3.4. Results

- The impact of object-specific image processing on the system's reliability can be clearly seen from all tests. Under tube (TL) light, which is quite dark and thus giving rise to much overlap between colors, the old DT2004 system can not localize at all, or at most very poorly
- The impact of behavior-specific image processing and self localization on the accuracy of localization can clearly be seen from the localization test in the penalty area. The vision system of the goalie, with new behavior based vision and self-localization, specifically designed for localizing in the penalty area, performs about 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 most clearly be seen from the clear ball behavior test with obstacles on the field where there is a dramatic difference in performance between the goalie and the striker version of the vision system. With 1 flood light, where none of the robots can detect flags, the goalie can still clear a number of balls before being lost.
- A final word on hypothesis 2: 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 even possible to decrease the CPU load to about 10% of the load we had in the old DT2004 code.

4. Conclusions

In this paper we have presented a behavior-based vision system. Several modifications have resulted in improvements of the performance. 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%. Using behavior-specific vision and localization for the goalie leads to an additional 50% increase in performance under normal conditions (due to gained accuracy) and to an increased reliability in difficult environments, such as when (strange) obstacles are visible in the playing field. The negative impact of wrongful staying in local loops because the original assumptions (such as "I'm in the goal") became invalid, once in a while, is far outweighed by these positive results. As general-purposeness is traded for a goal directed approach, the CPU load is drastically reduced for the same task from 50 to 10% of the original load. And last but not least, the contribution of this paper is not only in the improvement of the *performance* due to a different software architecture. Far more important is the fact that these improvements where realized in a time-frame of only a few person months programming effort of an inexperienced programmer. So one can also conclude that breaking down the system into small, understandable (sense-think-act) problems, makes it possible for interested students to make significant contributions to robot research in a short span of time, thereby keeping their enthusiasm.

5. References

1. http://www.robocup.org
2. R.Pfeifer and C.Scheier. Understanding Intelligence. The MIT Press, Cambridge, Massechussets, 1999, ISBN 0-262-16181-8.
3. R.A. Brooks. Intelligence without Representation. Artificial Intelligence, Vol.47, 1991, pp.139-159.
4. R.C. Arkin. Behavior based robotics, MIT press 19989, ISBN 0-262-01165-4
5. Parker, L. E. (1996). On the design of behavior-based multi-robot teams. Journal of Advanced Robotics, 10(6).
6. Richard S. Sutton, Andrew G. Barto; Reinforcement learning – an introduction., MIT press, 1998. ISBN 0-262-19398-1
7. Y. Takahashi, M. Asada. Modular Learning Systems for Soccer Robot (Takahashi04d.pdf). 2004, Osaka, Japan.
8. P.P. Jonker, Bas Terwijn, Jev Kuznetsov, and Bram van Driel, The Algorithmic foundation of the Clockwork Orange Robot Soccer Team, WAFR '04 (Proc. 6th Int. Workshop on the Algorithmic Foundations of Robotics, Zeist/Utrecht, July), 2004, 1-10.
9. T.G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Arti cial Intelligence Research, 13:227-303, 2000
10. 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
11. Thomas Rofer, Oskar von Stryk, Ronnie Brunn, Martin Kallnik and many other. German Team 2003. Technical report (178 pages, only available online: http://www. German-team.org/GT2003.pdf)
12. 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.
13. S.Thrun. Particle filters in robotics. In The 17th Annual Conference on Uncertainty in AI (UAI), 2002
14. 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
15. 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.
16. Lötzsch, M., Back, J. Burkhard, H-D., Jüngel, M (2004). Designing agent behavior with the extensible agent behavior specification language XABSL. In: 7th International Workshop on Robocup 2003 (Robot World Cup Soccer Games and Conferences in Artificial Intelligence, Padova, Italy, 2004. Springer (to appear).
17. 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.
18. 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).
19. Mohan Sridharan and Gregory Kuhlmann and Peter Stone. Practical Vision-Based Monte Carlo Localization on a Legged Robot. In IEEE International Conference on Robotics and Automation, April 2005. To appear.