The 6th Workshop on Humanoid Soccer Robots @ Humanoids Conference, Bled (Slovenia), October 2011.

Coordinated Pitch Observation for a Humanoid Robot Soccer Team

Felix Wenk
Department for Mathematics and Informatics
Universität Bremen
Email: fwenk@informatik.uni-bremen.de

Thomas Röfer
Sichere Kognitive Systeme
Deutsches Forschungszentrum für Künstliche Intelligenz
Email: Thomas.Roefer@dfki.de

***Abstract*—While the quality of matches between the teams in the RoboCup Standard Platform League has increased a lot, there are still certain situations that prevent the game from progressing. One of the most severe ones is when a team loses track of the ball, because it cannot score goals or prevent the opponent team from scoring goals without knowing where the ball is. In this paper a method is presented to quickly find the ball again by searching the least-recently observed parts of the pitch. A consistent model shared by all robots of the team to identify these parts of the field is explained, as well as the procedure to coordinate the observation among the teammates, such that a varying number of robots can participate in the process.**

I. Introduction

The RoboCup Standard Platform League (SPL)[1] is certainly one of the most exciting research settings. It poses a number of general robotics problems to be solved, e.g. detecting certain features in images, self-localization, multi-agent cooperation, etc., while being constrained enough to allow a fair comparison of the results achieved to solve the aforementioned problems (cf. Fig. 1). The key element of the league is the fact that all teams participating use the same standard platform, i.e. the robot *Nao* manufactured by Aldebaran Robotics [1]. Thus, the teams can focus on the software side of robotics problems, rather than constructing robots on their own. As a result, the league has quickly progressed in its four years of existence, which made the games interesting to watch even for those, whose main interest is not robotics but soccer itself.

Despite all the advances in developing a software to let a group of *Nao* robots play soccer, it becomes noticeable during pretty much every game that the scene is only partially observable by a single robot. This results in robots losing track of certain features, which may be more or less severe, depending on the feature being lost. The most severe case is losing track of the ball, since this defeats nearly all of the behaviors of a soccer player, given that the overall task is both to prevent the ball from rolling into the player's own goal and to score goals for his own team.

To avoid this situation, numerous measures are taken, such as maintaining a combined, team-wide model of the current state of the ball or implementing multi-agent cooperation to watch the motion of the ball [2]. While this significantly reduces the chance of losing track of the ball, it does not

[1]http://www.tzi.de/spl

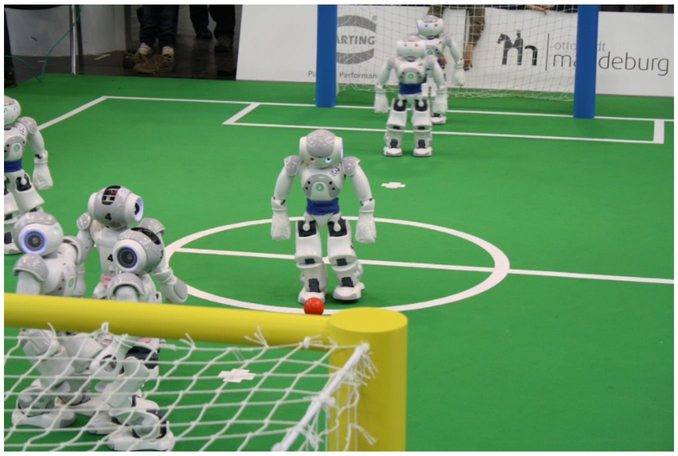
Fig. 1. RoboCup Soccer Standard Platform League: Nao robots autonomously playing a 4 on 4 soccer match in a color-coded environment.

completely prevent it from occurring. Therefore, it is crucial to regain to find the ball again as quickly as possible to continue the normal, ball-aware behavior.

In this paper a method called *field coverage* is introduced to quickly find the ball again after it has been lost. It has been implemented in the robot soccer software of the team B-Human, the SPL world champion of the past three years. Prior to using this method, the reaction of B-Human's robots to losing track of the ball position was to just spin around while scanning the pitch with the camera, hoping that the ball would show up again in a camera image while scanning for the ball [3]. If the latter did not happen, the robot would walk to a few different, fixed positions on the field, hoping to find the ball along the way.

This procedure, called *patrolling* in the B-Human context, changed considerably as a fourth robot was allowed playing in an SPL team. A searching robot now takes into account which parts of the field are actually visible and cooperates with the other robots during patrolling. The idea is that if the team as a whole knows some part of the field well, i.e. one robot of the team looked at that part recently, but the ball has not been seen, then the ball is probably somewhere else.

Since the introduction of the team-wide ball model, a robot only needs to actively search for the ball if all team members lost knowledge of the ball position. Consequently if a robot is searching, it can be sure its team members are search for the ball, too. Knowing which parts of the field are visible to the team members, patrolling can happen dynamically in a much more organized way.

In the remaining paper related work is discussed in the next section, the field coverage method is introduced in section III. The results are briefly discussed in section IV before the paper is concluded and possible future work is discussed in the last section.

II. Related Work

Mazda Ahmadi and Peter Stone developed a method to perform sweeping tasks with multiple robots [4]. To carry out such a task, a "robot must repeatedly visit all the points in its environment in an effort to detect and react to different types of events" [4]. While their method solves a problem that is more general than the scope of the method presented in this paper, they interestingly chose an orange ball appearing on a green pitch as an event type in one of their experiments.

Similar to the field coverage presented here, Ahmadi and Stone divided the robot's environment into a grid made up of cells, for each of which they recorded when the robot visited the cell in question the last time. They then learned a policy mapping the current state of the robot, i.e. its pose and the grid, to points in the environment the robot should visit next. This policy is subject to a function calculating the cost of a policy with respect to different events which possibly happen in different parts of the field and are of different importance.

This approach is extended to multiple robots by adding a negotiating procedure for the region of responsibility for each robot, in which the robot then carries out the continuous sweeping. Due to the negotiation, these regions can change dynamically and the addition and removal of other robots can be handled. Notably the negotiation procedure does not require communicating the poses of the robots, with the addition of a robot as an exception.

Although it shares the grid component, the approach presented in the next section is somewhat different. It does not take different types of events into account, so the only criterion is the time passed since a certain part of the field has been visited. In addition neither a policy function nor a negotiation is involved. Instead, the robots communicate their state and do all the calculations independently, which converge roughly to the same result on each robot. The action to be carried out then depends on the locations of the regions, as not the entire field is partitioned but only the least-recently visited parts of the field. The last visit to such a cell probably was quite some time ago, since most of the time the robots do not continuously sweep the environment, but ignore large parts of the pitch being busy trying to score goals.

For coordinated *exploration* of an environment using a team of robots, a very different method of partitioning a map into segments is used in [5]. To carry out the partitioning, the Voronoi Graph of the map of the environment is computed. That graph is then separated at nodes, the distance of which

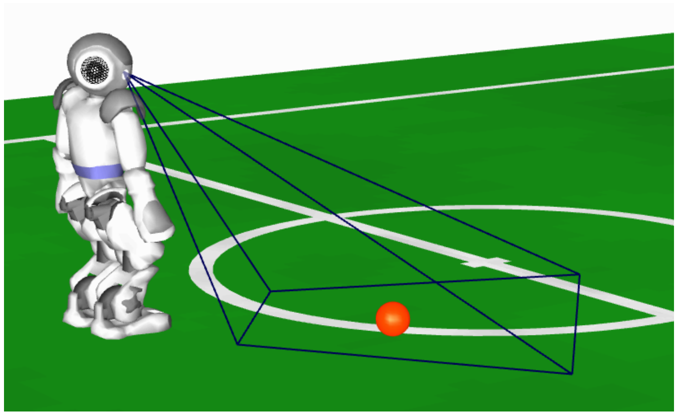
Fig. 2. Projecting the area visible by the camera onto the field.

"to the closest obstacle is at a local minimum" [5], which are subject to certain constraints. These nodes are called *critical points*. The resulting segments are then assigned to the robots which are exploring the environment. Thereby the intrinsic structure of the environment is taken into account when the targets for the exploring robots are generated. While such a structure is missing from a plain RoboCup soccer pitch, this approach may be used as a basis for future work.

Kohlbrecher and von Stryk also represent the field as a grid that is filled with visual perceptions of a humanoid robot, but not for coordinated ball searching but for obstacle avoidance [6]. Therefore, it accumulates measurements of obstacles and free space, rather than timestamps of the last observation of a grid cell. The grid has a much higher resolution than the one used in the work presented here, because it is not communicated between the teammates.

III. Field Coverage

The main part of the method presented in this section is to maintain a model of the knowledge each robot has about the field. This model is split into two parts, the *local field coverage* and the *global field coverage*. A part of the field is considered to be covered, if a robot looked at it recently. The local field coverage of robot r_1 contains which parts of the field the robot r_1 has looked at recently, whereas the global field coverage of any robot contains which parts of the field have been seen recently by any robot of the team.

A. Local Field Coverage

To keep track of which parts of the field are visible to a robot, the field is divided into a very coarse grid of cells, each cell being a square that has a size of $\frac{1}{4}m^2$ (cf. Fig. 3). To determine which of the cells are currently visible, the current image is projected onto the field (see Fig. 2). Then all cells the centers of which lie within the projected image are candidates for being marked as visible, unless either robots are obstructing the view to that cell (cf. Fig. 4) or the cell is so far away (2 m) that other robots would not be recognized safely by the vision system [7]. Having determined the set of visible cells, each of

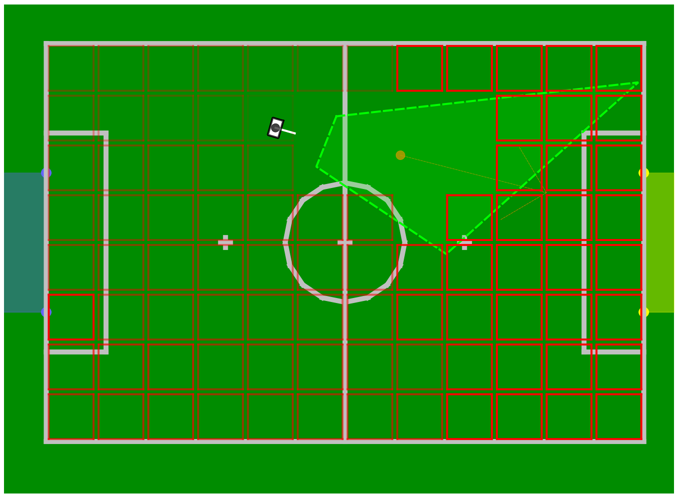
Fig. 3. A local field coverage grid of one robot. The more intense the red color around a cell is, the further in the past is the last time the robot looked at the cell. The area bounded by the dashed line indicates the projection of the field of view of the robot onto the pitch.

those cells is timestamped. These timestamps are later used to build the global field coverage model and to determine the least-recently-seen cell that can be used to generate the head motion to scan the field while searching for the ball.

A special situation arises when the ball goes out. If this happened, the timestamps of the cells are reset to values depending on where the ball is likely to be put back onto the field. This way, the least-recently-seen cell of the grid – the cell which the robot has the most outdated information about – is now the cell in which the ball is most likely to be put back. This cell is determined by the last intersection of the trajectory of the ball with an outer field line before the referee computer sent the signal that indicates that the ball is out. Of course, this grid resetting can only work well if the ball motion was estimated accurately and if the referees put the ball on the correct position on the field. However, without resetting, the information stored in the grid would not be useful anyway. One possible result of this grid resetting is shown in Figure 5.

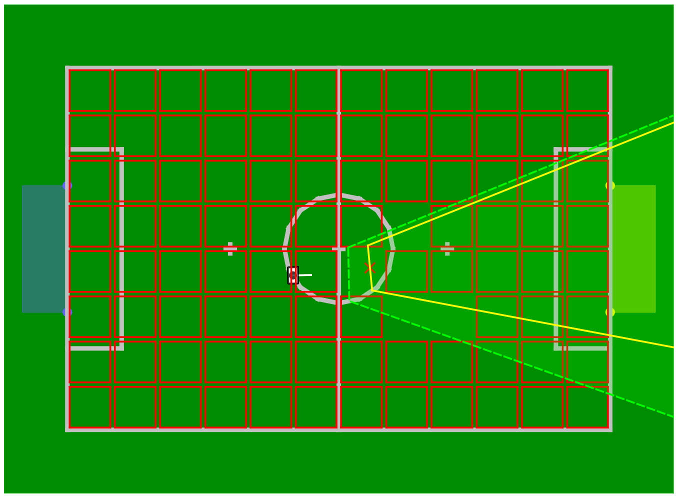
Fig. 4. The local field coverage for a robot standing on the center circle with an opponent robot in front. The detected robot is marked by a small red cross. The detected robot obstructs a large part (yellow lines) of the field of view (light green dashed lines) and thus prevents the cells behind him to be marked as visible.

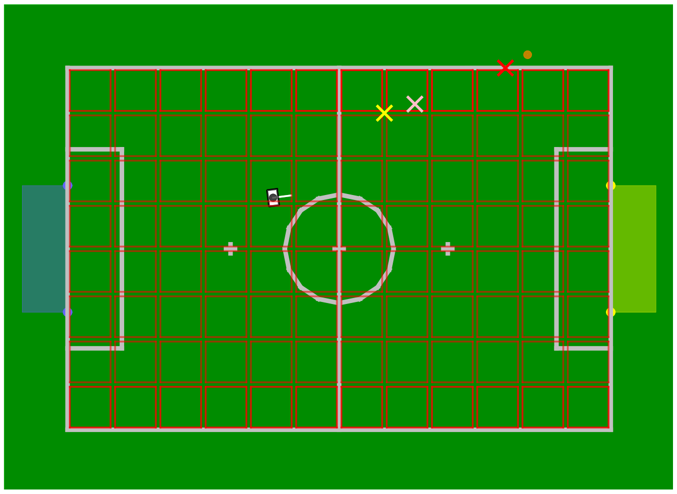
Fig. 5. Grid resetting: if the referee computer signals that the ball went out, the local field coverage grid is reset. The position where the ball crossed the border of the pitch is marked by a red cross. The grid has been adjusted such that the least-covered cells are at the field border where the ball is most likely being placed back into the game. The least-covered cell is the cell containing the throw-in position according to the rules, given that the estimate of the position the ball went out is accurate. In this case this is one meter further to the own goal, marked by the pink cross. The lower left corner of the throw-in cell is marked by a yellow cross.

B. Global Field Coverage

In addition to its own local field coverage grid, each robot maintains the field coverage grids of its teammates, which are incrementally updated in every team communication cycle. To make use of the field coverage grids of the other robots, each robot has to communicate its grid to its team mates. Given this year's field dimensions, 4-byte-timestamps and that each cell is a square with $\frac{1}{2}m$ edge length, there are $4bytes \times \frac{4m \times 6m}{(\frac{1}{2}m)^2} = 384bytes$ which have to be sent in each team communication cycle in addition to the other data the robots exchange do during gameplay. Since the resulting bandwidth requirement would be beyond the bandwidth limit set by SPL's rules, the timestamps are 'compressed' to a single byte and the grid is not sent as a whole but in slices. For each cell c which is part of the slice which is to be sent to the other robots, a one byte coverage value $v(c)$ is computed, such that $time - (255 - v(c)) \times tick$ roughly matches the timestamp stored for cell c, with $time$ being the reference, i.e. the current timestamp. Setting $tick = 300ms$ leads to coverage values which reach up to $76.5seconds$ into the past. With $n = 3$ being the number of slices the grid is divided into, the field coverage only adds $\frac{96}{3} + 4bytes = 36bytes$ to the team communication. The 4

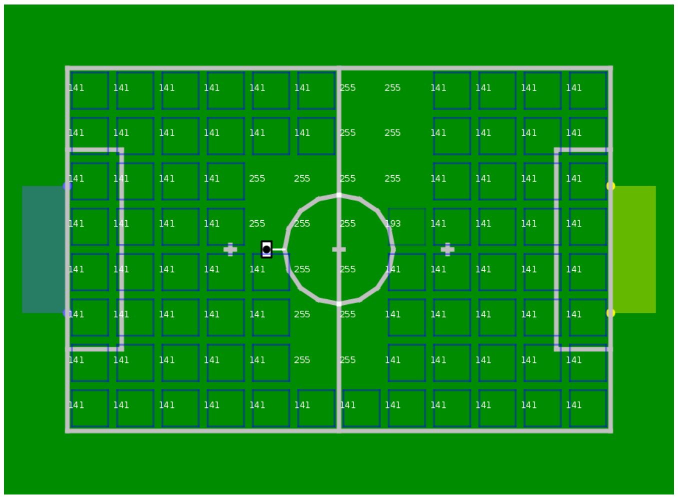

Fig. 6. The global field coverage grid. This grid is obtained by merging the communicated local grids. Note that there are covered cells more than 2 meters away from the robot (black circle within a white rectangle with a black border), so these cells are covered by other robots.

extra bytes are for the reference timestamp which has to be sent with the coverage values.

As a result, each robot has its own local field coverage grid and the ones it received from its teammates at its disposal. All these grids have to be merged into a single global grid that looks roughly the same for all teammates so that calculations based on the grid come to sufficiently similar results for all team mates. The value of each cell $g(i)$ of the global coverage grid is determined by calculating the maximum of all values $c(i)_r$ stored by each individual robot r for that cell:

$$g(i) = \max_{r=1}^{N} c(i)_r \tag{1}$$

Such a merged grid is shown in Figure 6. Note that in the shown grid cells are covered well, although they are clearly more than 2 meters away from the robot.

Based on the values in the global field coverage grid, it has to be decided which parts of the field are covered by the robots and which parts are not, i.e. which parts are unknown to the team as a whole. Therefore, a threshold is required to separate the two classes. It has to be determined dynamically, because a fixed threshold could result in the entire field being considered uncovered or covered, although there are still significant differences in the coverage of the cells. The problem has some similarities to determining which parts of a gray scale image are black or white. Therefore, we applied the Otsu algorithm [8] to compute the threshold.

The idea is as follows: The ideal situation to separate the coverage values into *covered* and *uncovered* would be the coverage grid containing only two different coverage values v_{low} and v_{high}. In this case, we could just choose the coverage threshold $t = (v_{low} + v_{high})/2$ to be in between them. This situation will not occur very often. Therefore, we reverse the process and calculate for every possible threshold $0 \leq t \leq 255$, how well the resulting model with only two coverage values fits the coverage grid if we choose optimal v_{low} and v_{high}. To do this, we need an error function that determines how well a model fits the actual grid.

$$e(v_{low}, v_{high}) = \sum_{c \in Grid} \min\left((v(c) - v_{low})^2, (v(c) - v_{high})^2 \right) \tag{2}$$

Instead of summing up all cells in the grid, we build a histogram h of all coverage values, so we can get rid of the minimum and rewrite eq. (2) as

$$e(v_{low}, v_{high}) = \underbrace{\sum_{v=0}^{t} h(v)\,(v - v_{low})^2}_{e_{v_{low}}} + \underbrace{\sum_{v=t+1}^{v_{max}} h(v)\,(v - v_{high})^2}_{e_{v_{high}}} \tag{3}$$

In eq. (3), t is the threshold as defined above, v is a coverage value and v_{max} is the maximum coverage value. Now for a given t we can find the optimal values for v_{low} and v_{high} by minimizing both $e(v_{low})$ and $e(v_{high})$. By taking the derivatives and solving for v_{low} and v_{high} respectively, the optimal values turn out to be the average coverage value of all coverage values below t for v_{low} and above t for v_{high}:

$$v_{low} = \frac{\sum_{v=0}^{t} h(v) * v}{\sum_{v=0}^{t} h(v)} \quad v_{high} = \frac{\sum_{v=t+1}^{v_{max}} h(v) * v}{\sum_{v=t+1}^{v_{max}} h(v)} \tag{4}$$

By substituting eq. (4) into eq. (3), we get an error function that only depends on the choice of the threshold t. After some simplification this is:

$$e(t) = \sum_{v=0}^{v_{max}} h(v)v^2 - \frac{\left(\sum_{v=0}^{t} h(v)v\right)^2}{\sum_{v=0}^{t} h(v)} - \frac{\left(\sum_{v=t+1}^{v_{max}} h(v)v\right)^2}{\sum_{v=t+1}^{v_{max}} h(v)} \tag{5}$$

With eq. (5), we calculate the optimal threshold by just trying every possible threshold and then using the threshold with the minimal error. Eq. (5) looks very computationally expensive. Note, however, that in each iteration over t the summations either change by only a single summand or not at all if the threshold t is increased or decreased monotonically.

After it has been determined which cells are the uncovered ones, each cell has to be assigned to a robot that will look at it. This is done using k-means clustering. k is set to be the number of robots that are able to cover a certain part of the field, i.e. to be included, a robot must not be fallen down or penalized and must be reasonably confident in its self-localization.

The clusters are initialized with the current positions of the robots and each uncovered cell is assigned to its closest cluster. After that, the new cluster means are computed based on the center positions of the cluster's cells. This process is repeated until the assignments do not change anymore. A simplified version of the approach is shown as pseudo code in Algorithm 1.

Handling the addition of a robot to patrolling is accomplished by setting the mean of the region of the added robot

Algorithm 1 Assigning cells to robots. Each robot is associated to a region which is considered *valid*, if the robot is able to patrol.

```
$threshold$ ←highest 'uncovered' value
$Grid$ ←global field coverage grid
$converged \leftarrow false$
while $\neg converged$ do
    $converged \leftarrow true$
    for all $cell \in Grid$ do
        if $coverage(cell) > threshold$ then
            $r \leftarrow region(cell)$
            Remove cell from region r
        else
            $r$ ←clostest valid region to cell
            if $region(cell) \neq r$ then
                Move cell to r
                $converged$ ←false
            end if
        end if
    end for
    for all $r$ ∈valid regions do
        Compute new mean of r
    end for
end while
```

to the robot's current position on the field and by marking the region as valid, so that cells can be assigned to that region. To remove a robot from patrolling, the corresponding region is marked as invalid and all cells currently belonging to the region are removed from that region.

Using four-way floodfill on each cell of each cluster, the connected components of each cluster are computed and the largest connected component of each cluster is retained. So for each robot currently able to patrol there is now one connected component of uncovered cells. Now for each robot the geometric center of the component is calculated, and used as a patrol target for that robot, such that each robot knows the patrol targets of all robots of the team, including its own. Figure 7 shows the result of applying the procedure on the global field coverage grid depicted in Figure 6.

As the robots move over the field, the patrol targets change with the field coverage, such that each robot that can patrol has its patrol target in its closest, largest region of uncovered cells. Notice that besides leading to meaningful patrol targets this procedure also has the nice property that the ways of different robots to their patrol target do not cross.

IV. Results

The field coverage method was used during the RoboCup 2011 competition in Istanbul. Together with many other improvements in B-Human's soccer system, the field coverage method significantly contributed to B-Human winning the world cup for the third time in a row. To qualify the successfulness of the approach, we performed a simulation experiment. Using our simulator SimRobot [9], we ran a four-against-four

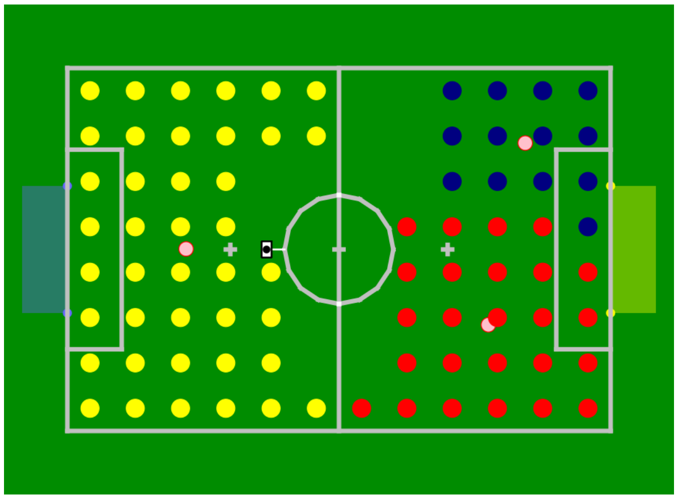

Fig. 7. The largest connected components of each cluster of uncovered cells. Each cell of such a component is marked by a yellow, blue or red circle to indicate the cluster assignment. The pink circles are the resulting patrol targets.

game, where the B-Human code played against itself. The simulator provides a limited refereeing system. Thereby, the game can run completely automatic. Figure 8 shows the results of this game that represent five hours of effective game time, in which the ball was "beamed" to a random location every 20 s. The data was recorded when the robots were actually playing soccer, not when they prepared for the next kick-off, because they do not try to track the ball during that time. The graph shows a histogram over all image processing frames of a single player, for how long *the whole team* has not seen the ball (for estimating the shared ball model, cf. [2]). Note that usually, the frequencies should continuously decrease towards longer absences of the ball, since, e.g., a ball that was absent for two seconds must have been absent for one second before. However, there are a few exceptions, because the time of absence is computed for the whole team, and sometimes perceptions from teammates are ignored, because they fell down, and one of their earlier perceptions is used for the computation instead. This might let the time "jump". The results show that nearly 90% of the time, ball perceptions are less than 300 ms old. Nearly 95% of the time, they are less than a second old. The longest absence of the ball measured was 18.7 seconds during a game time that is the equivalent of 15 regular SPL (final) games.

The improvement of the approach over the manual placement of patrolling points is hard to quantify, since B-Human never implemented a patrolling behavior with fixed positions for a team of four robots. Doing so only for this paper would not have been a realistic comparison.

As a part of the B-Human software, all calculations have to be carried out in real time. Measurements on a single Nao of a team playing soccer showed that the execution time of all global field coverage operations combined is consistently below one millisecond. With one robot within the field of view

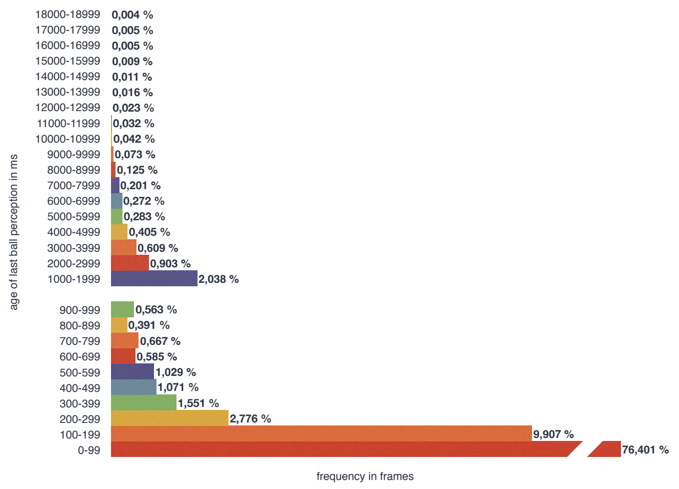

Fig. 8. The distribution of the time of how long the whole team has not seen the ball in a simulated game.

of the Nao, the local field coverage operations run in less than half a millisecond on average.

V. Conclusion & Future Work

In this paper, the field coverage method to quickly explore the least-recently visited parts of the pitch in order to regain knowledge of the ball position is presented. Part of the method are a model to dynamically identify these parts of the pitch and a procedure to dynamically assign those parts to a varying number of robots, such that the assignments are known to all teammates and the robots can observe these parts of the pitch in a coordinated way. In contrast, using fixed target positions for patrolling has the disadvantage that the manually placed points are static and do not adapt themselves to the game situation in which the ball is lost. As a consequence, the paths to the static patrolling points are likely to cross, increasing the time to reach the targets. Using the field coverage these disadvantages could be removed. In addition, the number of robots is not fixed. Therefore, the need to develop different patrolling strategies for the team's behavior for different numbers of patrolling robots could be removed as well. So besides an improved performance on the pitch, the programmed behavior of the soccer team could be greatly simplified.

The method is implemented in the B-Human software system and has been used successfully during the RoboCup 2011 SPL competition. It reduced game delays in situations in which the entire team lost track of the ball and could not find it by just spinning around, either because the view of the ball was obstructed or the ball was too far away to be reliably detected in camera images. After all, the field coverage method was one of the main building blocks for B-Human's continuing success, not only to win all games and the world cup, but also doing so at least five goals ahead in each game.

Currently the partitioning of the pitch only takes into account the coverage values of the cells of the global field coverage grid and the positions of the robots taking part in the patrolling process. The positions of the robots of the opponent team are only considered implicitly as they affect the coverage values of the cells that they obstruct. Taking opponent robots into account may be the subject of future work on this topic. As already suggested in section II, the approach to segmenting an environment taken in [5] could be applied to the segmentation of the pitch as well. The structure of the environment would be given by the boundaries of the pitch and the actual obstacles, i.e. the robots of the opponent team. Instead of the constraints geared towards exploration, the critical points used to separate regions could be constrained such that at least one of its closest obstacles is a genuine obstacle, i.e. not a pitch boundary.

While this approach might increase the performance of the patrolling, it has to be tried whether this approach is practicable given that obstacles change their positions fairly quickly and the limiting computing power of the Nao.

Acknowledgment

We would like to thank the team B-Human for providing the software basis for this work.

References

[1] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "The NAO humanoid: a combination of performance and affordability," *CoRR*, vol. abs/0807.3223, 2008.

[2] T. Röfer, T. Laue, J. Müller, A. Fabisch, K. Gillmann, C. Graf, A. Härtl, A. Humann, and F. Wenk, "B-Human Team Description for RoboCup 2011," 2011, available online: http://www.b-human.de/downloads/bhuman11_tdp.pdf.

[3] T. Röfer, T. Laue, J. Müller, A. Burchardt, E. Damrose, A. Fabisch, F. Feldpausch, K. Gillmann, C. Graf, T. J. de Haas, A. Härtl, D. Honsel, P. Kastner, T. Kastner, B. Markowsky, M. Mester, J. Peter, O. J. L. Riemann, M. Ring, W. Sauerland, A. Schreck, I. Sieverdingbeck, F. Wenk, and J.-H. Worch, "B-human team report and code release 2010," 2010, only available online: http://www.b-human.de/file_download/33/bhuman10_coderelease.pdf.

[4] M. Ahmadi and P. Stone, "A multi-robot system for continuous area sweeping tasks," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, may 2006, pp. 1724 –1729.

[5] K. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 1160 –1165.

[6] S. Kohlbrecher and O. v. Stryk, "Modeling observation uncertainty for soccer playing humanoid robots," in *Proceedings of the Fourth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots*, C. Zhou, E. Pagello, S. Behnke, E. Menegatti, T. Röfer, and P. Stone, Eds., 2010.

[7] A. Fabisch, T. Laue, and T. Röfer, "Robot recognition and modeling in the robocup standard platform league," in *Proceedings of the Fourth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots*, C. Zhou, E. Pagello, S. Behnke, E. Menegatti, T. Röfer, and P. Stone, Eds., 2010.

[8] N. Otsu, "A threshold selection method from grey level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[9] T. Laue, K. Spiess, and T. Röfer, "SimRobot - A General Physical Robot Simulator and Its Application in RoboCup," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence, A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds., vol. 4020. Springer, 2006, pp. 173–183.