# B-Human 2011 – Eliminating Game Delays*

Tim Laue[1], Thomas Röfer[1], Katharina Gillmann[2],
Felix Wenk[2], Colin Graf[2], and Tobias Kastner[2]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de
[2] Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany
{kathy,fwenk,cgraf,dyeah}@informatik.uni-bremen.de

**Abstract.** After having won the Standard Platform League competitions in 2009 and 2010, the B-Human software already included sophisticated solutions for most relevant subtasks, such as vision, state estimation, and walking. Therefore, the development towards RoboCup 2011 did not focus on replacing specific low-quality components, but was guided by an overall goal: eliminating game delays by more efficient actions and faster reactions to game state changes. This required several changes all over the system. In this paper, we present some of the developments that had the most impact regarding our goal: different ball models and corresponding cooperative ball tracking and retrieval strategies, a path planner as well as new approaches for tackling situations.

## 1 Introduction

*B-Human* is a joint RoboCup team of the Universität Bremen and the German Research Center for Artificial Intelligence (DFKI). The team consists of numerous undergraduate students as well as three researchers. The students participate in the team in the form of a two-year project course. Afterwards, some of them also write their thesis in the team's context. The researchers have already been active in a number of other RoboCup teams, such as the GermanTeam and the Bremen Byters (both Four-Legged League), B-Human and the BreDoBrothers (Humanoid Kid-Size League), and B-Smart (Small-Size League). Due to this particular continuity, the team always has a significant number of experienced members and we have been able to incrementally improve the overall software performance without major breaks during the past years.

Since its start in the Standard Platform League in 2009, the team B-Human has won every tournament it participated in. The status of the team's software after RoboCup 2010 can be considered as complete regarding solutions for most of the league's major challenges, such as a robust vision system, precise self-localization, and fast and stable walking. However, the overall game performance

* The authors would like to thank all B-Human team members for providing the software base for this work.

T. Röfer et al. (Eds.): RoboCup 2011, LNCS 7416, pp. 25–36, 2012.

indicated a significant lack of reactivity in some regularly occurring situations, such as retrieving a lost ball, winning a tackle against a dribbling opponent, or avoiding a walking obstacle.

This situation has led to the overall goal for 2011: eliminating game delays. In this paper, we describe some of the most significant developments that contributed to achieve this goal. To overcome any delays resulting from failed ball tracking, a cooperative ball model as well as corresponding cooperative ball tracking and retrieval strategies have been developed. Obstacle avoidance has become more efficient by the implementation of an RRT-based path planner. Finally, tackling situations can now be handled more successfully due to a new perception of the opponent's feet and the ability to carry out kicking motions within the walking pattern.

This paper is organized as follows: Section 2 briefly summarizes the Standard Platform League's state of the art, focusing on previous works of the B-Human team. The different ball models are presented together with the cooperative tracking and retrieval strategies in Sect. 3. The path planner is described in Sect. 4, followed by the developments regarding tacklings in Sect. 5.

## 2 State of the Art

As aforementioned, the current major challenges of the Standard Platform League can be considered as more or less solved, not only by B-Human, but also by a number of other teams. In this section, we briefly describe the currently used approaches, with a focus on developments related to the works presented in this paper.

Vision is confined mostly by NAO's limited computing resources. Therefore, most teams rely on manual color classification and detect cues by a combination of heuristics and grid-based or blob-based clustering approaches. However, some teams such as HTWK [16] already have systems that perform an automatic color classification. Whereas it is obviously common to reliably detect major objects such as the ball and the goals, only a few teams perform visual obstacle detection. Current solutions for this task include a detection based on the robots' waistbands [4] as well as the usage of color histogram features [12]. In Sect. 5.1, we present a new approach for detecting nearby robots in tackle situations.

For self-localization, probabilistic state estimation approaches such as (different variations of) Kalman filters [8], Monte-Carlo localization [5], or a combination of both are successfully applied by almost all teams. Several teams also estimate the ball's velocity in a sophisticated manner, as indicated by a number of effectively jumping goal keepers. Furthermore, it can be assumed that most teams communicate the ball state among their robots and possibly also perform a fusion of the different measurements. In Sect. 3, we present the cooperative ball tracking approach employed by our team.

Probably due to the limited computational resources of the NAO platform, elaborated planning approaches for action selection or motion planning are currently not common in the Standard Platform League. Another reason might be

the fact that most problems in this domain can still be handled by simpler approaches, such as finite state machines. However, Steffens *et al.* [15] have already presented an A* path planner for the SPL. In Sect. 4, we describe the advantages of using an RRT-based path planning approach.

Although Aldebaran Robotics already provides a robust walking implementation for the NAO robot, all successful teams rely on their own walking approaches which are able to reach higher speeds. Recent works have been published by Czarnetzki *et al.* [3] and Graf and Röfer [6]. Kicking implementations are in many cases based on static sequences of key frames. However, in recent years, some teams introduced dynamic kicking motions, such as HTWK [17], NaoDevils [2], and B-Human [13]. An approach for combining walking and kicking is presented in Sect. 5.2.

## 3 Ball Models

The ball is the most important object in a soccer game as its state determines the behaviors of all robots at any moment during a game. Therefore, it is highly advantageous for a team if all robots know its correct position as often as possible. In addition, knowing the velocity of the ball is important, because this allows predicting future ball positions. For instance, for the goal keeper to decide when to dive, it must know when the ball would cross the goal line. Together with a friction model, it can also be predicted, where a ball will come to a halt, allowing robots to directly head to that position instead of chasing a moving target.

Keeping track of the ball was one of our major goals for 2011. The *local ball model* estimates the ball's position and velocity for each individual robot. The *global ball model* fuses the local ball models of all players to a team-wide estimate of the ball's position and velocity. The *synchronized head control* tries to make sure that the team does not miss unexpected ball movements. Finally, the *field coverage model* is used to coordinate the search for the ball in case it has been lost.

### 3.1 Local Ball Model

The local ball model uses Kalman filters [8] to derive the actual ball motion from the perceptions of the ball delivered by the vision system. Since ball motion on a RoboCup soccer field has its own peculiarities, as for instance instantaneous speed changes due to kicks and ball repositioning due to referee interventions, the belief about the ball state is inherently multimodal. Since a single Kalman filter cannot represent a multimodal belief state, we use multiple multivariate Gaussian probability distributions (currently twelve) to represent the belief concerning the ball. Each of these distributions is used independently for the prediction step and the correction step of the filter. Only one of those distributions is used to generate the actual ball model. That distribution is chosen depending on how well the current measurement, i. e. the position the ball is currently seen at, fits to the mean of the distribution and how small the variance of that

distribution is. That way we get a pretty accurate estimate of the ball motion while being able to quickly react on displacements of the ball, for example when the ball is moved by the referee after being kicked off the field.

To further improve the accuracy of the estimation, the distributions are equally divided into two sets, one for rolling balls and one for balls that do not move. Both sets are maintained at the same time and get the same measurements for the correction steps. Since the perceived position and motion of the ball can change rapidly at almost any time, the worst distribution of each set in each frame gets reset to effectively throw one filter away and replace it with a newly initialized one.

There are some situations in which a robot changes the motion of the ball. After all, we filter the ball position to finally get to the ball and kick it. The robot influences the motion of the ball either by kicking it or by just standing in the way of a rolling ball. To incorporate these influences into the ball model, the mean value of the best probability distribution from the last frame gets clipped against the robot's feet. In such a case, the probability distribution is reset, so that the position and the velocity of the ball get overwritten with new values depending on the motion of the foot the ball is clipped against. Since the vector of position and velocity is the mean value of a probability distribution, a new covariance matrix is calculated as well. The covariance matrix determining the process noise for the prediction step is fixed over the whole process. Contrary to that, the covariance for the correction step is derived from the actual measurement; it depends on the distance between robot and ball.

### 3.2 Global Ball Model

Unlike some other domains, such as the Small Size League, the robots in the SPL do not have a common and consistent model of the world, but each of them has an individual world model, estimated on the basis of its own limited perception. However, the rules allow the robots to communicate with each other over WLAN, using a limited bandwidth of 500 kbit/s per team. Since a shared model is a necessity for creating cooperative behavior, we implemented a combined world model that lets all robots of a team have an estimate of the current state of the world, even if parts of it were not seen by each individual robot. This estimate is consistent among the team of robots (aside from delays in the WLAN communication) and consists of three different parts (*global ball model*, *positions of the teammates*, and *positions of opponent players*).

In this paper, we focus on the global ball model. It is calculated locally by each robot, but takes the ball models of all teammates into account. This means that the robot first collects the last valid ball model of each teammate, which is in general the last received, except for the case that the teammate is not able to play, for instance because it is penalized or fallen down. In this case, the last ball model computed before that incident is used. Since a ball model might already be impeded by misreadings when a robot detects, for instance, that it is falling, the last valid ball model is the one that was received 500 ms before the robot

reports that it is incapacitated. To still be able to access this model, each robot buffers the ball models from its teammates for the last 500 ms in a ring buffer.

The only situation in which a teammate's ball model is completely ignored is when the ball was seen outside the field, which is considered a false perception. After the collection of the ball models, they are combined in a weighted sum calculation to get the global ball model. The ball model of each individual robot is weighted by the product of different factors. The first factor is the validity of the self-localization estimate (pose). The validity reflects how unimodal the belief about the robot's pose is. We use a combination of particle filter and Kalman filter for self-localization. Basic self-localization is provided by an Augmented Monte-Carlo localization [7]. However, when the belief state about the pose seems to be unimodal, the Kalman filter takes over and tracks the pose provided by the particle filter, until the particle filter suggests a completely different pose, for instance because the robot was kidnapped. When the Kalman filter is in charge, the validity of the pose is set to 1. Otherwise, it is the ratio between the number of particles in the largest cluster of the particle filter, i. e. the one from which the pose is calculated, and the overall number of particles. The second factor is the period of time $\Delta n$ in seconds since the ball was last seen by the robot, i. e. how long it has not been seen. The third factor is the period of time $\Delta m$ in seconds for which the ball is missing, i. e. how long the ball was not seen although it should have appeared in the robot's camera image according to the ball model. The final factor is the approximate deviation of the distance to the ball based on the bearing. Technically, $\sigma$ is computed as the expected change of the distance measurement if the vertical bearing of the ball would be wrong by 1°. From these factors, a weight $w_r$ is calculated for each robot $r$. While a higher validity results in a bigger weight, larger values for the other three factors reduce that weight:

$$w_r = validity_{pose_r} \left(1 - \frac{1}{1 + e^{\alpha - \beta \Delta n_r}}\right) \left(1 - \frac{1}{1 + e^{\gamma - \delta \Delta m_r}}\right) \frac{1}{\sigma_r} \qquad (1)$$

The values used for the constants in the equation are $\alpha = 5, \beta = 1, \gamma = 4, \delta = 4$, which means that missing a ball that should have appeared in the camera image is significantly worse than not having seen the ball because the robot looked somewhere else. Based on the weights for all $N$ robots considered, a common ball model is calculated that contains an approximate position and velocity of the ball.

### 3.3 Synchronized Head Control

Although the ball is the most important object in a soccer game, even the robot approaching or having possession of the ball has to regularly look away from it to perceive field lines and goal posts to update its self-localization. In particular, this is necessary when the robot is close to the ball, since in this case, camera images showing the ball will usually not contain any objects that support self-localization. While the robot is looking away from the ball, the

risk of missing unexpected ball movements is rather high. On the one hand, an opponent robot might kick the ball away, and on the other hand, the robot itself might inadvertently touch the ball and thereby move it away. Since searching for the ball delays the game, the teammates are responsible for keeping track of the ball when the robot closest to the ball (i. e. the *striker*) looks away.

To accomplish this, all robots continuously broadcast whether their current head motion will show them the ball in their camera image according to their ball model. If it will not and if the sending robot is currently the striker, its teammates will look at the ball instead, independently of what their original intention for a head motion was. It is not guaranteed that each of them will actually be able to see the ball, because other robots could be blocking the view. However, the fact that all of them are looking at the ball increases the chance that at least one of them will actually be able to see it. But even if none of them does, it is still advantageous to observe the region where the ball would appear eventually when the striker has lost it. Note that there is no negotiation between the robots involved. In particular, the robot having possession of the ball is not limited in any way by this behavior.

Experiments have shown that as a result of this synchronization, the periods of time in which no robot of the team sees the ball are below 100 ms most of the time and very rarely climb above one second.

### 3.4 Field Coverage Model

All the methods described above cannot completely prevent the team from losing track of the ball, for instance when it is hidden by an opponent robot or it has been moved by a referee. Therefore, it is also important to optimize the process of finding the ball again after it has been lost. In general, searching for the ball is a three-step process. First, a robot just moves its head and sweeps the area it can see without moving its body. Then, it starts turning on the spot. Finally, it starts walking around the field to get a different perspective. Where the robot looks at during the search can be optimized by managing both the areas it has already covered and the areas that have been covered by its teammates.

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 $0.5m \times 0.5m$ (cf. Fig. 1a, b). To determine which of the cells are currently visible, the current image is projected onto the field. 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 or the cell is so far away (2 m) that other robots would not be recognized safely. Having determined the set of visible cells, each of 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 cell in which the ball most likely has been put back is determined by the last intersection of the trajectory of the ball with an outer field line before the referee

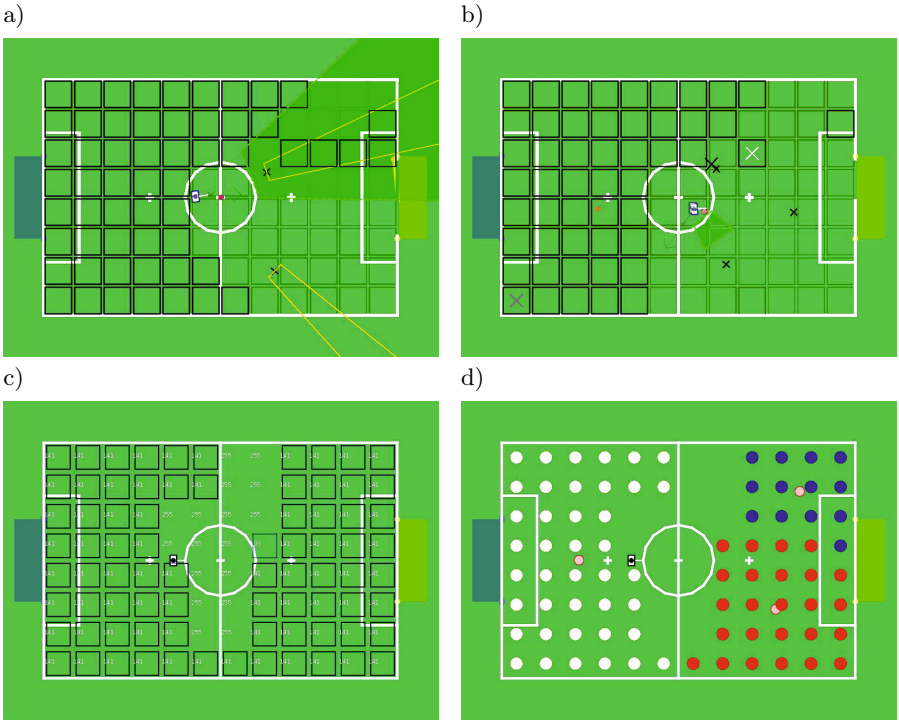


**Fig. 1.** Local and global field coverage models. a) The visible area of a single robot projected on the field. Uncovered cells are black. Other robots (each denoted by a black X) cast shadows over the visible area. b) The model generates the best camera targets to search for the ball. The white X marks the best target that is reachable without turning. The gray X marks the overall best target. c) The global grid obtained by merging the communicated local grids. d) The largest connected components of each cluster of uncovered cells. Each cell of such a component is marked by a circle in a different color to indicate the cluster assignment. The three extra circles mark the resulting search targets.

computer sent the signal that indicates that the ball is out. Knowing this cell, the timestamps of the entire grid are reset such that this cell, in which the ball most likely is, appears to be the most outdated one and the cells at the left and right field borders appear to be more outdated than the rest of the grid. 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.

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. 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 of the global coverage grid is determined by calculating the maximum of all values stored by each individual robot for that cell.

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 [14] to compute the threshold.

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. Using four-way flood-fill on each cell of each cluster, the connected components of each cluster are computed and the largest connected component of each cluster is retained. This results in a connected area of uncovered cells for each robot (cf. Fig. 1c, d). The geometric center of that area is calculated and it is used as a target position for that robot from where it will search for the ball.

## 4 Path Planning

In recent years, we have used a reactive obstacle avoidance approach based on ultrasonic measurements. This approach has not been effective as the robot only reacted on recently measured obstacles in its immediate vicinity, a behavior that caused multiple problems (cf. Fig. 2a). First and foremost, avoiding an obstacle whilst keeping the original walking target in focus requires walking sidewards, which is exceedingly slow on the NAO. Additionally, reactive obstacle avoidance recurrently guides the robot into situations that thwart fluent play. The most prominent example for such a situation are local minima, which occur often in presence of multiple robots (especially after the last increase of the number of players), but also overtaking an opponent robot during a footrace towards the ball is a hard task.

To overcome these problems, we developed a path planner that is based on the established *Rapidly-Exploring Random Tree* algorithm [10]. This non-optimal algorithm is based on the random exploration of the search space and works on continuous values. It builds up a tree that quickly expands in few directions of the search space, as in each step of the algorithm the tree is enlarged by one

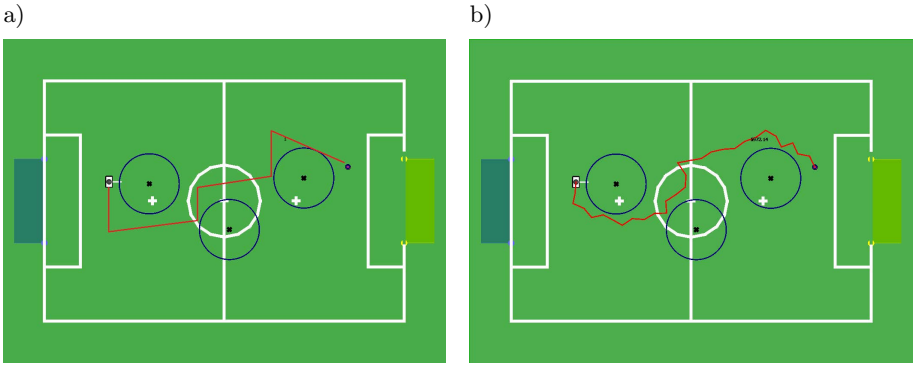


**Fig. 2.** Paths resulting from different approaches for obstacle avoidance: a) The previously used reactive behavior and b) the RRT planning algorithm

edge in a random direction (cf. Fig. 2b). For this general algorithm, different variants such as *RRT-Extend* [10], *RRT-Connect* [9] and *RRT-Bidirectional* [11] exist. We use *Extend* and a slightly modified variant of *Bidirectional*. Using the former variant restricts the expansion towards the random position to a given distance that is the same for each expansion, which has a direct influence on the expansion of the tree, whereas the latter variant has no influence on the tree itself but decreases the runtime. This is achieved by creating two separate trees, one beginning from the start point and one from the end point. Another modification is to replace the random position by the target position or a waypoint of the last found path with a given probability. Using this modifications helps to avoid oscillations of the path, for instance if there is an obstacle on the direct way to the target.

One advantage of using a path planner is its ability to consider a number of distant obstacles to prevent the aforementioned situations. For this purpose, the path planner not only uses the local ultrasonic measurements, but also integrates a combined world model, which is similar to the global ball model as described in Sect. 3.2, including the positions of all teammates and of the robots tracked by them. Moreover, having a planned path, the time for avoiding an obstacle can be decreased since the robot walks on a circular path around the obstacles, preferring a fast forward walk and rotations over slow sidewards motions (cf. Fig. 2b).

## 5 Efficient Tacklings and In-walk Kicks

Even a fast reaction regarding ball state changes and an efficient path planning approach cannot always prevent a situation that regularly occurs in each game: the presence of an opponent robot near the ball. Experiences from previous competitions showed that the resulting situations require fast actions as the robot that loses the tackling has only a minor chance of regaining ball possession.

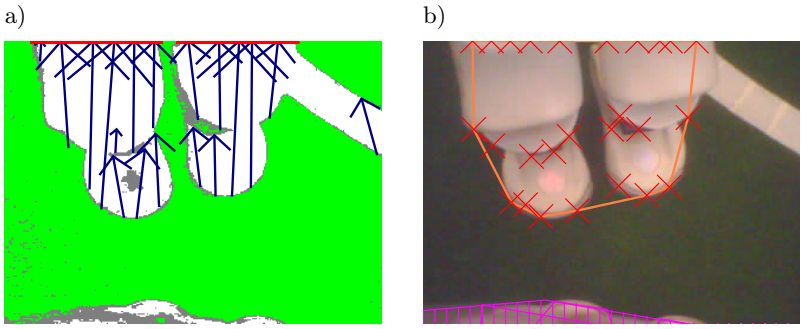


**Fig. 3.** Perception of robot feet. a) Segmented image. The arrows indicate white line segments detected in a preprocessing step but rejected by the field line detection algorithm. For being considered as robot parts, at least some of these segments must start at the upper image border. Therefore the single arrow on the right side of the image will be ignored. b) Raw image with the convex hull around the start and end points of the accepted segments is computed. The closest (thick circle), the leftmost, and the rightmost point (marked by the smaller circles) are determined to describe the feet.

Therefore, we spent much efforts not only in behavior tuning but also in the development of two approaches that strongly contributed to winning a majority of all tacklings: the perception of a nearby robot's feet as well as the ability to carry out kicks within the normal walking pattern.

### 5.1 Foot Perception

To successfully solve tackle situations by dribbling around an opponent, knowledge about the position of the opponent is necessary. For short ranges, the NAO's ultrasonic sensors provide reasonable distance measurements (and thus allow the detection of the presence of an obstacle) but quite poor angular information, including many false positives in case of centered obstacles.

To overcome these problems, a simple but yet effective solution has been found: the visual detection of the opponent's feet. This approach has two advantages: Firstly, when looking at a nearby ball, a blocking robot's feet are, in general, in the field of view and no additional search motions of the head are needed. Secondly, to dribble a ball around a robot, its feet are the only body parts of interest as other parts, such as upper body and arms, are obviously not able to block the ball. In addition, the perception of these parts is probably less precise and might distort the overall position estimate.

However, the position of nearby feet can be determined quite reliably and straightforward by clustering white image segments that have previously been rejected by the line detection algorithm, similar to the input used by Metzler *et al.* [12]. An example is shown in Fig. 3. The resulting perceptions are only used in tackle situations, where they override all other, comparably imprecise obstacle information. For path planning, the position of feet is not useful, as for this task, the upper bodies of other robots are the main obstacles.

### 5.2 In-walk Kicks

As already mentioned in Sect. 2, the tasks of walking and kicking are often treated separately, each solved by different approaches. In presence of opponent robots, such a composition might waste precious time as certain transition phases between walking and kicking are necessary to ensure stability. Direct transitions between walking and kicking are likely to let the robot stumble or, in the worst case, to fall over. Therefore, the B-Human walking implementation [6] is able to carry out sidewards and forward kicks within the walk cycle.

Such an in-walk-kick is described by a number of parameters. On the one hand, the sizes and speeds of the step before the kick and the step during the kick are defined. On the other hand, a 6-D trajectory (three degrees for translation and three degrees for rotation) relative to the original trajectory of the swinging foot is defined that overlays the original trajectory and thereby describes the actual kicking motion. The kick retains the start and end positions and speeds of a normal step. The instability resulting from the higher momentum of the kick is compensated by the walk during the steps following the kick.

## 6 Conclusions and Future Works

In 2011, B-Human showed again a strong overall performance and won every match at the German Open as well as at RoboCup 2011 in Istanbul. The developments presented in this paper significantly contributed to these achievements such that almost all tackles have been won and all robots were able to quickly react on any ball state changes. The latter, in combination with the new path planner, strongly decreased the number of situations in which an opponent robot was alone at the ball, having the chance to score.

Several components of B-Human's current system require a proper calibration to perform as desired. Therefore, the focus of future work is probably not the development of new soccer features – except for necessary adaptations regarding major rule changes – but the integration of approaches and tools that reduce calibration efforts and enable a more efficient testing. Currently ongoing theses works already include a new vision approach that does not rely on manual color calibration as well as two new approaches for robot behavior specification, a scripting language based on the programming concept of generators and a strategy definition based on the playbook approach [1] respectively, both leading towards more compact and faster ways to adapt behavior definitions.

## References

1. Bowling, M., Browning, B., Veloso, M.: Plays as effective multiagent plans enabling opponent-adaptive play selection. In: Proc. of the 14th Int. Conf. on Automated Planning and Scheduling (ICAPS), Whistler, BC, Canada, pp. 376–383 (2004)
2. Czarnetzki, S., Kerner, S., Klagges, D.: Combining Key Frame Based Motion Design with Controlled Movement Execution. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS (LNAI), vol. 5949, pp. 58–68. Springer, Heidelberg (2010)

3. Czarnetzki, S., Kerner, S., Urbann, O.: Applying Dynamic Walking Control for Biped Robots. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS (LNAI), vol. 5949, pp. 69–80. Springer, Heidelberg (2010)
4. Fabisch, A., Laue, T., Röfer, T.: Robot recognition and modeling in the RoboCup Standard Platform League. In: Pagello, E., Zhou, C., Behnke, S., Menegatti, E., Röfer, T., Stone, P. (eds.) Proc. of the Fifth Workshop on Humanoid Soccer Robots at the 2010 IEEE-RAS Int. Conf. on Humanoid Robots, Nashville, TN, USA (2010)
5. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte-Carlo localization: Efficient position estimation for mobile robots. In: Proc. of the Sixteenth National Conf. on Artificial Intelligence, Orlando, FL, USA, pp. 343–349 (1999)
6. Graf, C., Röfer, T.: A center of mass observing 3D-LIPM gait for the RoboCup Standard Platform League humanoid. In: Röfer, T., Mayer, N.M., Savage, J., Saranli, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 102–113. Springer, Heidelberg (2012)
7. Gutmann, J.S., Fox, D.: An Experimental Comparison of Localization Methods Continued. In: Proc. of the 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2002), Lausanne, Switzerland, vol. 1, pp. 454–459 (2002)
8. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME-Journal of Basic Engineering 82(Series D), 35–45 (1960)
9. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. In: Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation (ICRA 2000), San Francisco, CA, USA, vol. 2, pp. 995–1001 (2000)
10. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Tech. Rep. TR 98-11, Computer Science Dept., Iowa State University (1998)
11. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proc. of the 1999 IEEE Int. Conf. on Robotics and Automation (ICRA 1999), Detroit, MI, USA, vol. 1, pp. 473–479 (1999)
12. Metzler, S., Nieuwenhuisen, M., Behnke, S.: Learning Visual Obstacle Detection Using Color Histogram Features. In: Röfer, T., Mayer, N.M., Savage, J., Saranli, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 149–161. Springer, Heidelberg (2012)
13. Müller, J., Laue, T., Röfer, T.: Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots. In: Ruiz-del-Solar, J., Chown, E., Ploeger, P.G. (eds.) RoboCup 2010. LNCS (LNAI), vol. 6556, pp. 109–120. Springer, Heidelberg (2010)
14. Otsu, N.: A threshold selection method from grey level histograms. IEEE Transactions on Systems, Man, and Cybernetics 9(1), 62–66 (1979)
15. Steffens, R., Nieuwenhuisen, M., Behnke, S.: Multiresolution path planning in dynamic environments for the standard platform league. In: Pagello, E., Zhou, C., Behnke, S., Menegatti, E., Röfer, T., Stone, P. (eds.) Proc. of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS Int. Conf. on Humanoid Robots, Nashville, TN, USA, pp. 59–64 (2010)
16. Tilgner, R., Reinhardt, T., Borkmann, D., Seering, S., Kalbitz, T., Fritzsche, R.: Nao-Team HTWK – team description paper. In: Röfer, T., Mayer, N.M., Savage, J., Saranli, U. (eds.) RoboCup 2011: Robot Soccer World Cup XV Preproceedings, RoboCup Federation (2011)
17. Xu, Y., Mellmann, H.: Adaptive Motion Control: Dynamic Kick for a Humanoid Robot. In: Dillmann, R., Beyerer, J., Hanebeck, U.D., Schultz, T. (eds.) KI 2010. LNCS, vol. 6359, pp. 392–399. Springer, Heidelberg (2010)