TIGERs Mannheim

(Team Interacting and Game Evolving Robots)

Extended Team Description for RoboCup 2017

Mark Geiger, Chris Carstensen, Andre Ryll, Nicolai Ommer, Dominik Engelhardt, Felix Bayer

Department of Information Technology
Baden-Wuerttemberg Cooperative State University,
Coblitzallee 1-9, 68163 Mannheim, Germany
management@tigers-mannheim.de
`https://tigers-mannheim.de`

Abstract. This paper presents a brief overview of the main systems of TIGERs Mannheim, a Small Size League (SSL) team intending to participate in RoboCup 2017 in Nagoya, Japan. This year's ETDP focuses primarily on the artificial intelligence used during RoboCup 2016, especially strategic planning of offensive and defensive actions. Furthermore, the new robot design that was introduced at RoboCup 2016 is summarized.

1 Mechanical and Electrical System

Last year, we decided to perform the second major redesign of our robots since our first participation in 2011. Specifications of the new robots are listed in table 1. The new robots were ready just in time for RoboCup 2016 and performed exceptionally well. Compared to their previous version the robots are faster, more robust, easier to maintain and easier to develop on. Details of mechanical and electrical changes can be found in [1].

During development and testing of the new robots in official RoboCup games at Iran Open 2016 two minor mechanical issues were found. Firstly, the small sub-wheels experienced major damage during games even from only light impacts and hence prevented them to roll properly. Root of this problem is that the wheels were moved further apart from the robot's center to accommodate for larger motors and thereby cut-outs in the robot's cover were necessary. Without the protection of the cover the wheels are directly exposed to the environment. The problem was solved by mounting thicker cover plates on the wheels so that the sub-wheels do not stick out anymore. Hence, they cannot be hit directly. Figure 1 shows the new robot cover and the wheels' cover plates. The second issues pertains the infrared barrier electronics mounted directly at the dribbling unit of the robot. During tackle situations they were regularly damaged (e.g. LED ripped off of the PCB). This issue was mitigated by mounting thick rapid prototyping covers on the PCBs to absorb impacts.

Robot version	v2016
Dimension	Ø179 x 146mm
Total weight	2.5kg
Max. ball coverage	19.7%
Driving motors	Maxon EC-45 flat 50W
Gear	18 : 60
Gear type	Internal Spur
Wheel diameter	57mm
Encoder	US Digital E8P, 2048 PPR [2]
Dribbling motor	Maxon EC-max 22, 25W
Dribbling gear	50 : 30
Dribbling bar diameter	17mm
Kicker charge topology	SEPIC
Chip kick distance	approx. 2.5m
Straight kick speed	max. 8m/s
Microcontroller	STM32F746 [3]
Used sensors	Encoders, Gyroscope, Accelerometer
Communication link	nRF24L01+ @2MBit/s, 2.400 - 2.525GHz [4]

Table 1. Robot Specifications

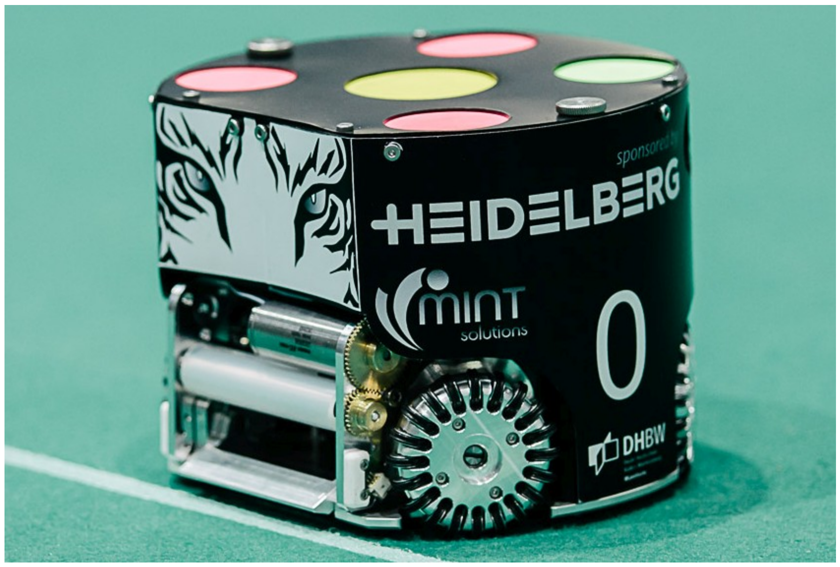


Fig. 1. New robot with cover on the field (v2016). Sub-wheels are completely covered by each wheels top plate to prevent direct impacts.

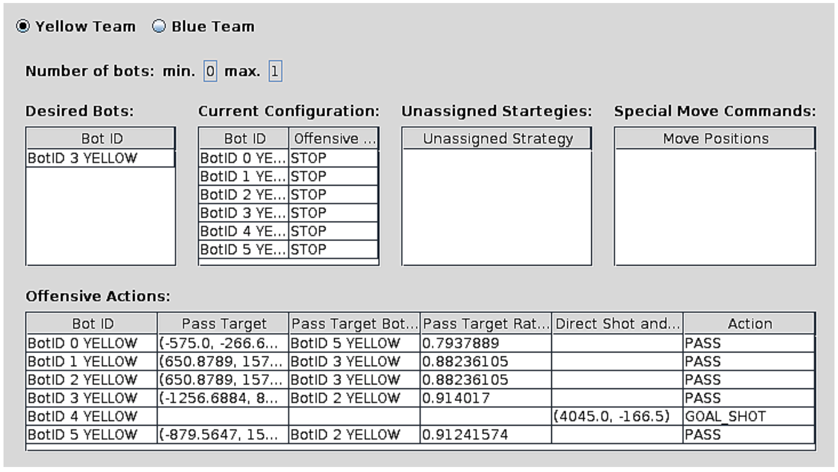
2 Artificial Intelligence

At RoboCup 2016, we have been rated as the most improved team by the community. We believe that this voting is also based on the capabilities of our dynamic AI which does not rely on different statically encoded strategies. Each move in a match is a unique result of a sequence of individual decisions. In this chapter, we are going to explain how our AI worked at RoboCup 2016, especially how it performs dynamic decisions instead of relying on static plays.

The strategy planning is divided into four groups: Offensive, Support, Defensive and Keeper. We already explained this concept in our TDP from 2015 [5]. The number of roles and the robot assignments are calculated using the algorithm introduced in 2015 as it proved to be stable and easy to maintain.

2.1 Offensive

Passing is almost inevitable to score goals against a strong defense. In this section, we show how we implemented dynamic passing situations in non standard game situations in RoboCup 2016. In each calculation frame (60fps) an offensive strategy is calculated. The offensive strategy is a complex data structure. A visualization can be seen in figure 2. For each robot the currently desired offensive state (explained in more detail in section 2.1) is calculated (Current Configuration). Additionally, the number of minimum, maximum and desired robots, the offensive group can currently handle, is determined. This is the input to the role assigner [5]. Furthermore, an offensive action is calculated for each robot. The offensive action basically tells the robot what it should do in the current situation. In this figure, robot 4 should shoot on the goal. All other robots should do a pass. The field "special move commands" is used to store specific move positions for offensive robots. These are used for passing situations. A pass receiving robot can use these commands to move to its pass receiving position before the pass was carried out by the pass sending robot.



[✓ Yellow Team] Blue Team

Number of bots: min. 0 max. 1

Desired Bots:

Bot ID
BotID 3 YELLOW

Current Configuration:

Bot ID	Offensive ...
BotID 0 YE...	STOP
BotID 1 YE...	STOP
BotID 2 YE...	STOP
BotID 3 YE...	STOP
BotID 4 YE...	STOP
BotID 5 YE...	STOP

Unassigned Startegies:

Unassigned Strategy

Special Move Commands:

Move Positions

Offensive Actions:

Bot ID	Pass Target	Pass Target Bot...	Pass Target Rat...	Direct Shot and...	Action
BotID 0 YELLOW	(-575.0, -266.6...	BotID 5 YELLOW	0.7937889		PASS
BotID 1 YELLOW	(650.8789, 157...	BotID 3 YELLOW	0.88236105		PASS
BotID 2 YELLOW	(650.8789, 157...	BotID 3 YELLOW	0.88236105		PASS
BotID 3 YELLOW	(-1256.6884, 8...	BotID 2 YELLOW	0.914017		PASS
BotID 4 YELLOW				(4045.0, -166.5)	GOAL_SHOT
BotID 5 YELLOW	(-879.5647, 15...	BotID 2 YELLOW	0.91241574		PASS

Fig. 2. Visualization of offensive strategy data.

Offensive states The offensive states reflect the different actions an offensive robot can do. For example, an offensive robot can simply shoot the ball, do some dribbling moves or prevent enemies from getting to the ball. In 2016, we had 7 different offensive states which are explained here:

KickState This is the most common used state. It is used to kick the ball.

StopState This state is used in stop situations. The robot should not move too close to the ball in this state.

DelayState The delay state is used in standard situations. It will wait for supporting robots to position on the field and then automatically switch to the KickState.

SpecialMoveState This state is used to prepare for receiving passes. Once the pass was initialized (ball kicked), the robot will switch to the RedirectAndCatchState. This state will also wait before moving to its desired destination to synchronize the robots arrival time with the balls arrival time to the pass target (see section 2.2) .

RedirectAndCatchState This state is used to redirect or catch incoming balls. Depending on the angle between the incoming ball and the shoot target of the pass receiving robot, the robot will decide whether he can redirect the ball without stopping it or if it has to stop the ball first.

InterceptionState The interception state is used in enemy standard situations to prevent the enemy robot from shooting the ball by intercepting its path.

ProtectionState In the ProtectionState the robot tries to hinder enemy robots from obtaining ball control. It will position itself between the ball and the enemy robot, waiting for the supporting robots to provide good passing positions.

The desired offensive state for each offensive robot is calculated by a central module. Once an offensive robot is doing a pass it will store this information in the OffensiveStrategy. The central module will then use this information to calculate the offensive strategy in the next frame. Therefore, the number of desired robots for the offensive strategy will be raised. This also means that the state of an offensive robot depends not only on its own situation, but also on the active states of other offensive robots.

2.2 Support

Our AI is mainly based on passing between our robots. This enables us to gain a faster game play. The main task of our supporting robots is to drive to a position where they can safely receive a pass and also have a good chance to score a goal. In recent years we calculated many different scores over the complete field to find the best support positions. By using the GPU, the calculations could be performed easily within reasonable time. The major disadvantage of using a GPU is to actually get it to work on everybody's computer during development. Thus we decided to go back to the CPU for all our processing. Therefore, nearly everything had to be rewritten in the supportive strategy.

To reduce computational effort we have split the field into sectors. Round about 50 sectors where calculated for each frame during RoboCup 2016. This calculation is quite simple and only consists of two values: the distance to the ball and the actual score chance from the center of each sector.

The Distance to the Ball Having just the right distance between the pass shooter and receiver is very important for a smooth gameplay. If we are too close to the ball a pass is quite useless and we would create a crowd on the field. If we are too far away from the ball, opponents could intercept and we lose the ball. So we used an exponential function to calculate the ballDistanceScore representing this:

$$ballDistanceScore = \frac{e^{-(actualDistance-desiredDistance)^2}}{distanceSigma^2}$$

The Score Chance Basically, we want to shoot as many goals as possible. Right after a pass, the chance of shooting a goal is quite high, because the whole situation on the field is changing for the defending team. So when we pass to another robot, this robot should have a good chance of shooting a goal from its position. The calculation of the actual score chance for a goal is a little more tricky though. In short, we are pretending that the position of the robot shooting the ball is a light source. The shadows of the surrounding robots created by this light source partially fall onto the goal. We search for the longest continuous spot of light in the goal, putting its length into relation to the goal size. Additionally, we take the distance from our position to the goal and the angle to the goal in consideration to calculate the score chance.

After that, we sort the sectors by their calculated values and assign the best sector to the nearest support robot. Often, nearby sectors are equally high valued. To avoid a crowd of supportive robots we recalculate the other sectors, lowering the values surrounding the best sector after each assignment. This will be repeated until every supportive robot has its position. When a robot enters his assigned sector it is trying to see the ball by moving in his sector to be ready receiving an incoming pass. Figure 3 shows a visualization of the implemented score chance. The ball, represented as orange dot in this figure, is the light source mentioned before. All other robots throwing shadows away from that position. As you can see a part of the goal is covered with the shadow, visualized with the red bar in front of the goal. The yellow bar next to it represents the free space in the goal. The size of that yellow gap put into relation to the actual goal size represents the score chance, without the distance and the angle to the goal.

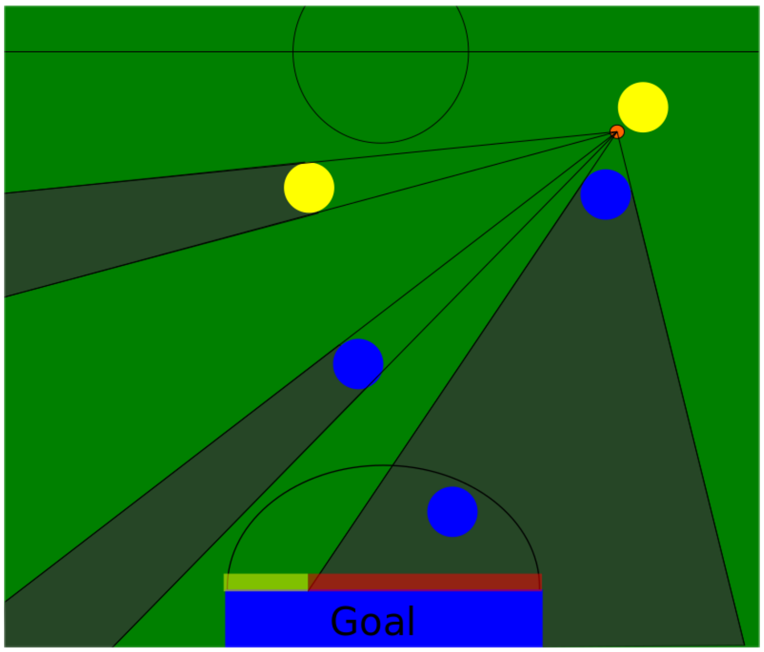


Fig. 3. Visualization of the score chance for estimating support positions. The ball is a virtual light source that creates shadows behind robots. The shadows are than used to evaluate the visibility of the goal.

Pass Targets Pass targets are potential pass positions which are calculated for each friendly robot on the field (except the keeper). Multiple pass targets exist for each robot, with each having a rating based on its visibility to the ball, the score chance from this position, the time difference for the nearest opponent and the robot to the pass target, the angle and distance to the goal. All these values have a own configurable weight. Figure 4 shows the calculated pass targets for a single robot. The pass targets are calculated around the robot, additionally offsetting it in the current movement path of the robot.

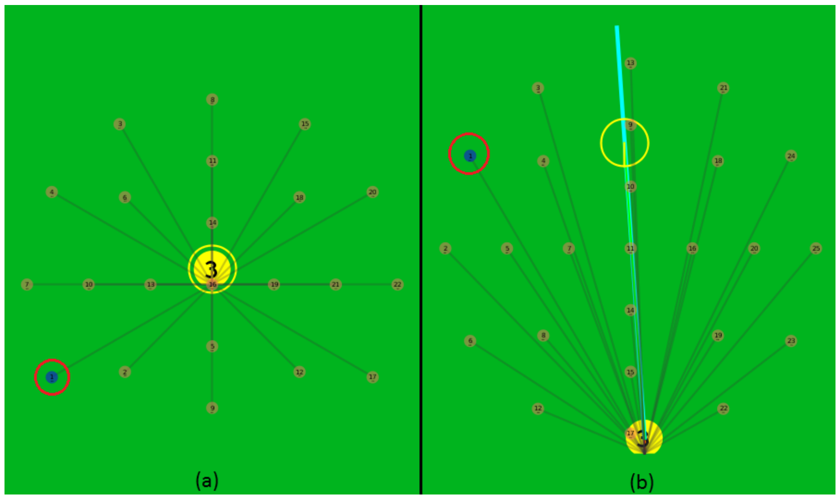


Fig. 4. Pass targets generated for a single robot. The best rated target is marked with a red circle. (a) shows the pass targets generated around a steady root. (b) shows the pass targets of a moving robot, where the cyan line is the velocity. The targets have an offset along the robots movement path.

Dynamic Passing and Synchronization Once an offensive robot gets close to the ball it will choose the currently calculated offensive action. In case of a pass, the desired number of robots will be raised by 1. This means that the pass receiving robot will also become a role of the offensive group. This happens before the pass has been played. The pass sending robot chooses a pass target which belongs to the pass receiving robot. The pass target position is not equal to the pass receiving robot position as explained in 2.2. Figure 5 shows a typical pass situation generated by our AI. Robot 3 plans to pass the ball to robot 4. Rather than passing to the robot position, it passes to a pass target near robot 4. Since we use trajectories to control our robot movement, we can calculate the time that robot 3 needs to kick the ball. Furthermore, we can use the initial ball velocity and our ball model [5] to calculate the time the ball needs to reach the pass target. Also, we can calculate the time robot 4 needs to reach his pass target. By combining these numbers we can synchronize the time in which the ball and robot 4 will reach the pass target. Robot 4 will simply wait until the trajectory time of robot 3 and the ball travel time are smaller than its own trajectory time.

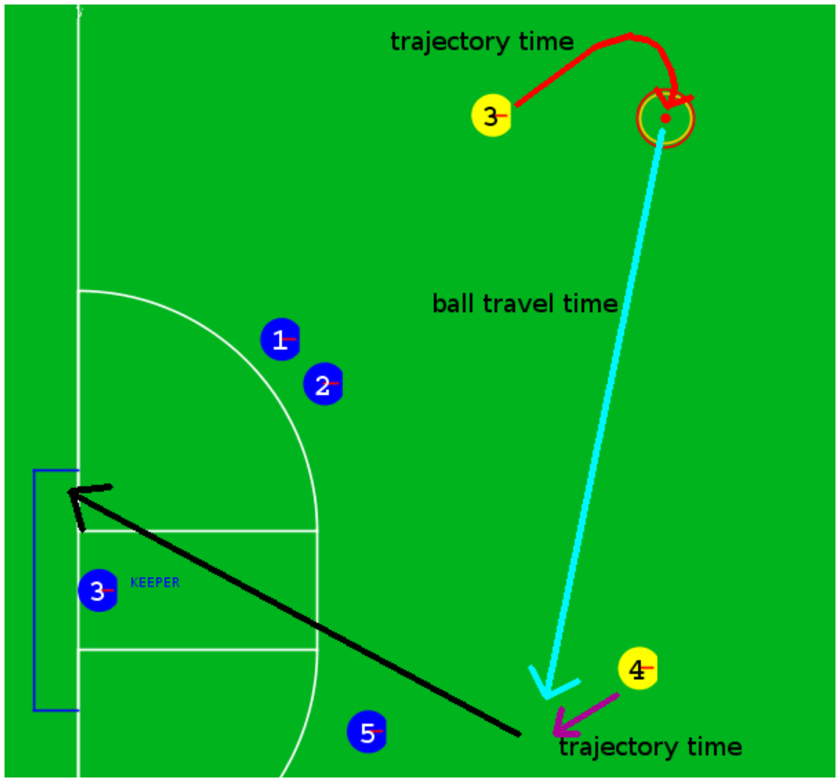


Fig. 5. Robot 3 (yellow) is about to pass the ball to one of the pass targets of robot 4. Robot 4 will wait on its current position until the ball travel time + the trajectory time of robot 3 is smaller as its own trajectory time.

Once the ball is rolling towards the pass receiving robot, it will automatically switch to the RedirectAndCatchState, preparing itself to redirect the ball. However, it can happen that even though the ball was played to one specific robot, another robot of our team could receive this ball more efficiently. Figure 6 shows one of this situations.

The pass was kicked at robot number 4, but clearly it is not suitable to receive this pass. In this case a triangle around the ball's movement path will be drawn, ending at the location where it will stop rolling. All robots inside the triangle can potentially redirect the ball. The robots inside the triangle are rated by their distance to the incoming ball and the distance to the ball's movement path. In this case, the best robot to redirect the ball would be robot number 3.

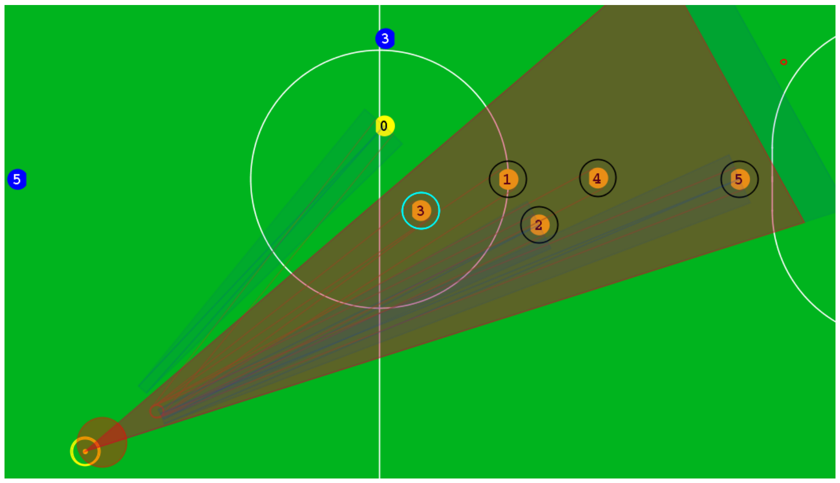


Fig. 6. Choosing the best robot to catch the rolling ball. The red triangle visualizes the direction of the pass. All robots with a black circle are potential receivers. The best robot its marked with a cyan circle.

Conclusion The described concept of the offensive we used at RoboCup 2016 has proven to work really well. Nonetheless, there are still some drawbacks:

- The needed calculations for the pass synchronization have not been accurate enough. We had to use quite big thresholds to work against that inaccuracy. The consequence of this was that the pass receiving robot reached his pass target way before the incoming ball.
- In standard situations we do not use any hard coded plays. We use the exact same strategy as previously described. However, some of the best teams in SSL use hard coded plays in standard situations. And they have shown to be really powerful. More powerful than our strategy in 2016.

However, we think that our dynamic strategies will enhance further in future. And we are convinced that in 2017 our offensive will be able to compete head on with the best teams in SSL.

2.3 Defensive

The defense described in [5] has been about optimizing the defense positions and selecting the right robot as defender. It showed its strengths in defending against the most dangerous robots selected by the defense. The robots to defend against were selected mainly by their distance to our goal. Therefore, the opposing team could pile, as much as there are defenders, robots in one corner of the field and put another pass receiving robot slightly farther away. This would lead our defense to only defend against the pile, but not consider the lonely robot with a very high chance to score a goal if he receives the ball.

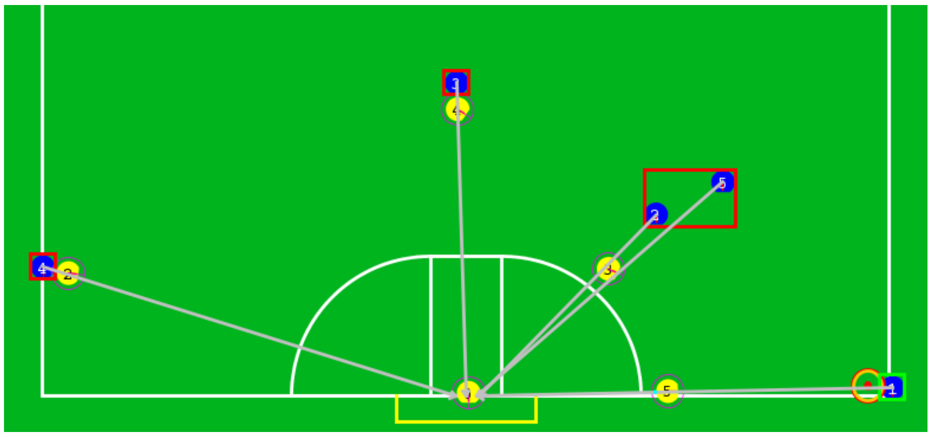


Fig. 7. Distributing defenders using groups

Group concept To address the weakness described in 2.3 a new group concept was implemented. When analyzing the offensive positioning of the opposing team, robots are grouped, if they have a similar angle to our goal center. To avoid the quick creating and resolving of groups the angle necessary to create a group is smaller than the one needed to resolve it.

When distributing defenders, each group gets assigned a defender before another group gets a second one. Figure 7 shows a typical corner situation, visualized in our software. Robots in the same group are enclosed by a rectangle. The rectangle is red by default and green, if the group is the group nearest to the ball. The previous implementation would have assigned the defender with the number two to one of the offensive bots two or five, while the offensive bot with the number four would have been uncovered. One can also recognize the positioning of the defenders on the vector between the offensive bots and the goal. The bot which covers more than one offensive bot has positioned itself closer to the goal to be able to quickly switch its position. The defenders covering a single offensive bot have moved as close to their respective covered bot to interfere with his positioning and intercept passes.

More defensive concepts Next to the group concept there were many small tweaks which we added to our defense:

Assignment At RoboCup 2015 we assigned the defenders to the defense position by selecting the assignment with the shortest cumulated time for the bots to reach their position. When a new offensive bot needed to be covered this could lead to every defender swapping its position with each other, leaving the goal uncovered for some amount of time. The distribution was improved by first covering high priority targets with the fastest defender and covering subsequent targets afterwards.

Positioning When interrupting the shoot vector of an offensive bot the defense begins to adjust its position on this vector. We implemented the possibility to quickly change between different goals when adjusting the position. A defender can additionally block the pass path, harass an offensive bot or be as close to the penalty area as possible.

Pathplanning To calculate path timings the defense did use a simple path planning without taking obstacles into account. This was replaced by our default pathplanning to get timings as realistic as possible.

Visualization The different possibilities and concepts are visualized in our AI software. This helps to understand different game states and analyze the behavior of the defense in replays.

Strengths and weaknesses The defense has be proved effective against some of the world´s most successful SSL teams. Besides preventing many goals against us, the defenders are available as pass targets for the offense and do not hinder the buildup of our offensive game.

Nevertheless, the defense has shown some weaknesses:

- When many groups are created and removed the defender still can be made toggling. This leads to holes in the defense because the defenders never get to their target position.
- How many defenders to use in which situations?
- The defense only does bot positioning. In many situations it could be helpful if the defense tries to interrupt passes, or blocks shots actively.

2.4 Keeper

In the last two years, the keepers logic has grown from a simple skill to a quite complex state machine. In RoboCup 2016 there were seven states covering all the necessary game situations for the keeper. The different states are described below.

Default To minimize the distance between the keeper and all potential positions, from where goal kicks could happen, the keeper is staying on the line between the ball and the center of its goal, effectively driving on a circle around the center of the goal. Furthermore, the robot is turned by 90° instead of facing the ball directly, because our robots can accelerate slightly faster in the forward and backward direction compared to left and right. This lets it intercept an incoming ball slightly faster and might prevent conceding a goal.

Move into penalty area To save time the keeper nearly never uses path planning, because there are no other robots in the penalty area. Just in case the keeper is outside of the penalty area path planning is activated until it reaches it. So this state is active if the keeper is outside of the penalty area and will move it into it.

Game stopped When the referee sends a stop message some special rules have to be considered. This state is nearly the same as the default state with the difference that the motion speed is reduced and the robot keeps a sufficient distance to the ball, as defined in the rules.

Defending ball velocity is directed to goal This is the most important state in the whole keeper state machine. If the velocity of the ball is directed to the goal and is moving away from the robot that previously was in possession of that ball, then the keeper tries to through itself into the shooting line. If the keeper has enough time to reach the receive-position and stop there, it faces the ball to stop and control the ball properly. However, if it is impossible to reach the shooting line in time with the robot stopping there, it tries to over-accelerate such that it cannot come to a full halt where the ball would be received but still has a chance of stopping the ball.

Defend redirect A very dangerous situation during the game is when the opponent team passes from one side of the field to the other and tries to redirect the ball directly to the goal. The keeper is capable of detecting those kinds of redirects. If a redirect is detected the keeper does not try to follow the ball, instead moving directly in the way of the receiving opponent to block it's sight of the goal.

Go out In case an opponent robot is with the ball near the penalty area the keeper drives just as far toward the outer edge of the penalty area so it can cover the whole goal but still is able to quickly defend the other side of the goal if necessary. If the opponent then tries to push the ball into the penalty area by force and touches the keeper in the process it's a free-kick for our team.

Chipping ball out of penalty area After successfully stopping the ball inside the penalty area the keeper has to chip it away. Therefore it uses the same pass targets as our offensive strategy does, filtering those out that are in front of a configurable line on the field. This will prevent the keeper from passing to a defender in our half.

2.5 Estimation of Robot and Ball Behavior

A good estimation of the behavior of robots and balls is essential for planning. Describing everything in detail would be out of scope of this paper, so this section will only give a rough overview of how we currently estimate behavior of robots and the ball. In recent years, we made our robots move more and more precise. Since 2013, our robots are able to move independently of our software framework, given a target position. The independent base station, that is responsible for wireless communication, is connected to the Ethernet at the field. It receives data from the SSL Vision and redirects positional data to the individual robots. The first version of this architecture is described in [6]. In 2016, we used bang bang trajectories based on [7] within the control loop of the robot. While the trajectories are not optimal, they are sufficient and fast to compute. Our path planning algorithm is based on the same trajectories, so it will find paths that the robot is able follow. Additionally, the path is actually an aggregation of trajectories and provides the position and velocity of the robot by time. These trajectories are used for obstacle avoidance of the other robots, for ahead of time predictions in the AI and other estimations like ball interception. The algorithms were tuned continuously over recent years and showed to be effective and very useful in 2016.

In 2015, we introduced a ball prediction model based on regression [5] which we have not changed in 2016. The model produced good estimations for rolling balls, but could not generalize well for fast balls. For 2017, we are developing a new two phase model based on [8], which will be more accurate and less error prone.

3 Publication

Our team publishes all their resources, including software, electronics/schematics and mechanical drawings, after each RoboCup. They can be found on our website[1]. The website also contains several publications[2] about the RoboCup, though some are only available in German.

References

1. A. Ryll, M. Geiger, N. Ommer, A. Sachtler, and L. Magel. TIGERs Mannheim - Extended Team Description for RoboCup 2016, 2016.
2. US Digital. E8P OEM Miniature Optical Kit Encoder, 2012. `http://www.usdigital.com/products/e8p`.
3. STmicroelectronics. STM32F745xx, STM32F746xx Datasheet, December 2015. `http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00166116.pdf`.
4. Nordic Semiconductor. nRF24L01+ Product Specification v1.0, 2008. `http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P`.
5. A. Ryll, M. Geiger, N. Ommer, J. Theis, and F. Bayer. TIGERs Mannheim - Extended Team Description for RoboCup 2015, 2015.
6. A. Ryll, N. Ommer, M. Geiger, M. Jauer, and J. Theis. TIGERS Mannheim - Team Description for RoboCup 2014, 2014.
7. O. Purwin and R. D'Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54(1):13 – 22, 2006.
8. C. Lobmeiner, P. Blank, J. Buehlmeyer, D. Burk, M. Eischer, A. Hauck, M. Hoffmann, S. Kronberger, M. Lieret, and M. Eskofier. ER-Force - Extended Team Description for RoboCup 2016, 2016.

[1] Open source / hardware: https://tigers-mannheim.de/index.php?id=29
[2] publications: https://tigers-mannheim.de/index.php?id=21