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`

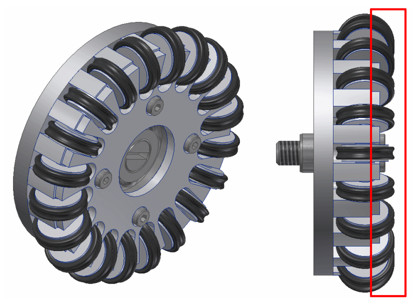
Abstract. This paper presents a brief overview of the main systems of TIGERs Mannheim, a Small Size League (SSL) team intending to participate in RoboCup 2017 in Nagoya, Japan. This year's ETDP focuses primarily on the artificial intelligence used during RoboCup 2016, especially strategic planning of 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 the previous version, the robots are faster, more robust, easier to maintain, and easier to develop on. Details of the 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 from rolling 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 a comparison of the first and second version of the wheels' cover plates.

The second issue 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. The covers can be seen in figure 2.


(a) First version of the wheel's cover plate. Sub-wheels are exposed (red).

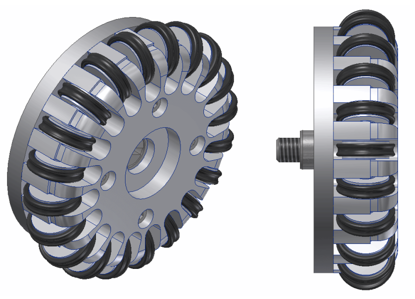

(b) Updated version of the cover plate. Sub-wheels are completely covered to prevent direct impacts.

Fig. 1: Comparison of wheel's cover plate.

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	14mm
Kicker charge topology	Flyback Converter (up to 230V)
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

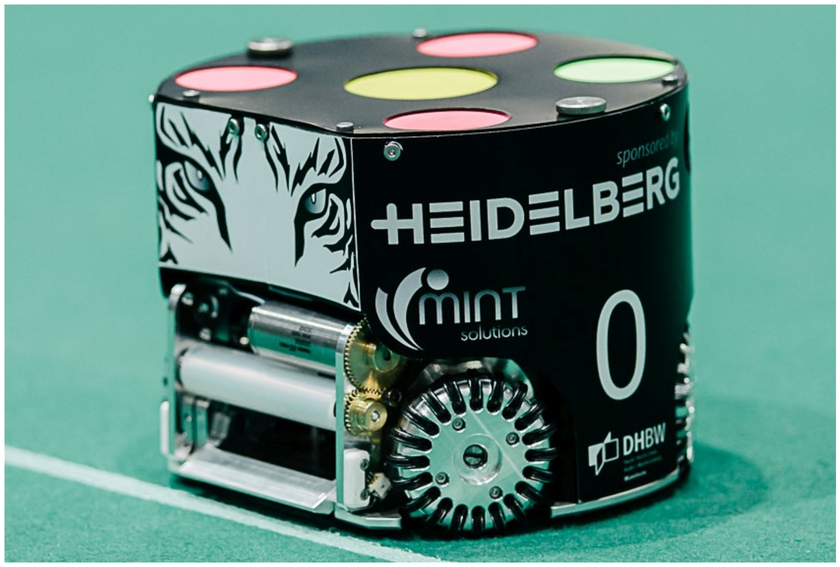


Fig. 2: New robot with cover on the field (v2016). Left and right to the dribbling device the new infrared barrier protection can be seen (white).

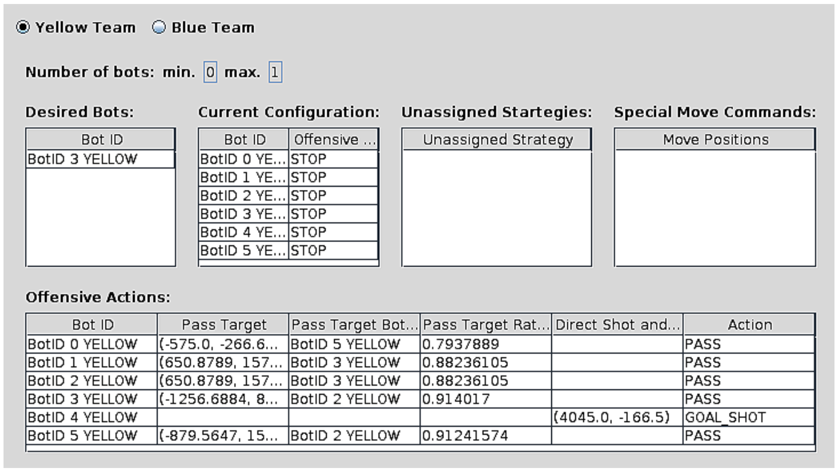
2 Artificial Intelligence

At RoboCup 2016, we were rated the most improved team by the community. We believe that this voting is not least 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 performed 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 present how we implemented dynamic passing situations in non-standard game situations at the RoboCup in 2016. With each calculation frame (60fps) an offensive strategy is calculated. The offensive strategy is a complex data structure. A visualization can be seen in figure 3. For each robot, the desired offensive state (explained in more detail below) is calculated. It is shown in column *Current Configuration* in figure 3. Additionally, the number of minimum, maximum and desired robots, which 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 the situation shown in figure 3, robot 4 should attempt a kick on goal. All other robots should do a pass. The field *Special Move Commands* are used to store specific move positions for offensive robots. These are used for passing situations. A pass receiving bot 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. 3: Visualization of offensive strategy data.

Offensive states The offensive states reflect the different actions an offensive robot can perform. For example, an offensive robot can kick the ball, do some dribbling moves or prevent opponents from acquiring the ball. In 2016, we had 7 different offensive states which are explained here:

KickState This is the most commonly used state. It is used to kick the ball in case of passes or goal kicks.

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 themselves 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 target of the pass receiving robot, the robot will decide whether it can redirect the ball without stopping it or if it has to stop the ball first. In case of a catch, the robot will then automatically switch to the KickState and fetch a new offensive action from the central module.

InterceptionState The interception state is used during opponents' standard situations. It tries to prevent a successful pass of the kicking robot by intercepting its path.

ProtectionState In the ProtectionState the robot tries to prevent opponent robots from obtaining ball control. It will position itself between the ball and the opponent 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. Consequently, 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 achieve a fast 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 well be performed 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 of our processing. Consequently, nearly the entire supportive strategy had to be rewritten.

To reduce computational effort we have split the field into sectors. Approximately 50 sectors were calculated in each frame during RoboCup 2016. The calculation is simple and only consists of two values: the distance to the ball and the actual score chance from the center of each sector.

Distance to the Ball Having the best distance between the pass sender 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 would lose the ball. This two constrains forms a torus around the ball, which could be described as the body of rotation of an exponential function. Here *actualDistance* describes the distance between the ball and a point, which should be evaluated. The *desiredDistance* represents the radius of the torus and *distanceSigma* the width of the ring.

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

Score Chance Basically, we want to score as many goals as possible. On the occasion of a pass, the chance of scoring a goal is quite high, because the whole situation on the field is changing for the defending team. Therefore, when we pass to another robot, this robot should have a good chance of scoring a goal from its position. The calculation of the actual score chance for a goal is more complicated. In short, we simulate that the position of the robot kicking 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. Then we put 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. A visualization of the calculation is shown in figure 4. As previously mentioned, the ball, represented as orange dot in this figure, simulates the light source. All other robots cast shadows away from that position. As you can see a part of the goal is covered with the shadow, visualized by the red bar in front of the goal. The green bar next to it represents the free space in the goal. The size of this gap put into relation to the actual goal size represents the score chance, without taking into consideration the distance and the angle to the goal.

After that, we sort the sectors by their calculated values and assign the best sector to the nearest support robot. Often, nearby sectors are valued equally high. 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 each support robot has a position. When a robot enters his assigned sector it tries to see the ball by moving in his sector to be ready for receiving an incoming pass.

Pass Targets Pass targets are potential pass positions which are calculated for each friendly robot on the field (except for the keeper). Multiple pass targets exist for each robot, with each one 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 the distance to the goal. All these values have their own configurable weight. Figure 5 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.

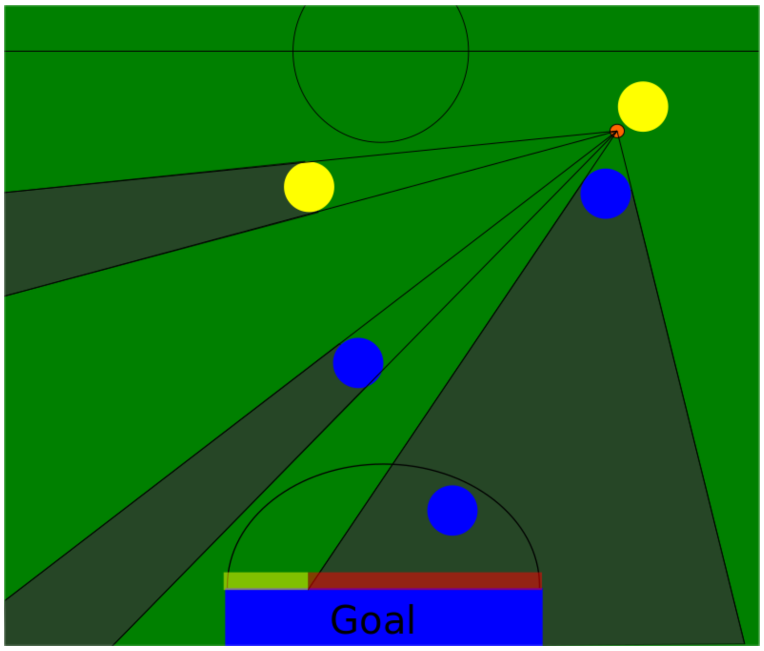


Fig. 4: Visualization of the score chance for estimating support positions. The ball is a virtual light source that casts shadows behind robots. The shadows are then used to evaluate the visibility of the goal.

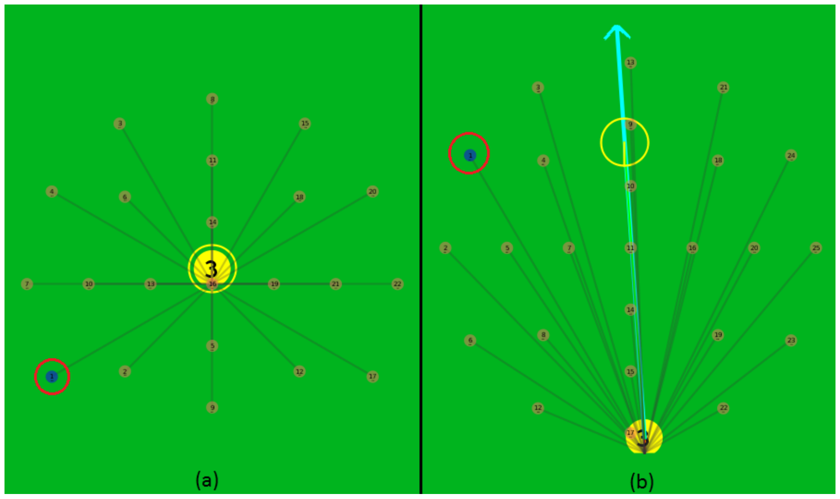


Fig. 5: Pass targets generated for a single robot. The best rated target is marked with a red circle. (a) shows the pass targets generated around a steady root. (b) shows the pass targets of a moving robot, where the cyan arrow indicates the velocity. The targets have an offset along the robots movement 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 one. This means that the pass receiving robot will also become part 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 6 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's position, it passes to a pass target near robot 4. Since we use trajectories to control our robot movement, we can calculate the time at which robot 3 needs to kick the ball. Furthermore, we can use the initial ball velocity and our ball model [5] to calculate the time at which 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.

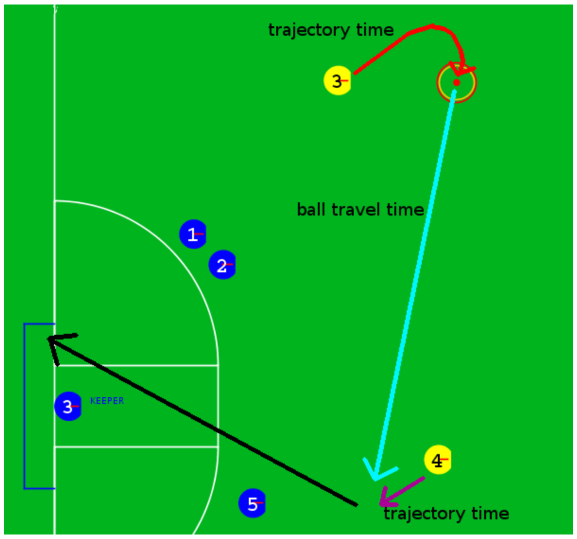


Fig. 6: Robot 3 (yellow) is about to pass the ball to one of the pass targets of robot 4. Robot 4 will wait on its current position until the ball travel time plus the trajectory time of robot 3 is smaller than its own trajectory time.

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 intended to be played to one specific robot, another robot of our team could receive this ball more efficiently. Figure 7 presents such a situation.

The pass was kicked towards robot number 4, but clearly it is not in a suitable position 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.

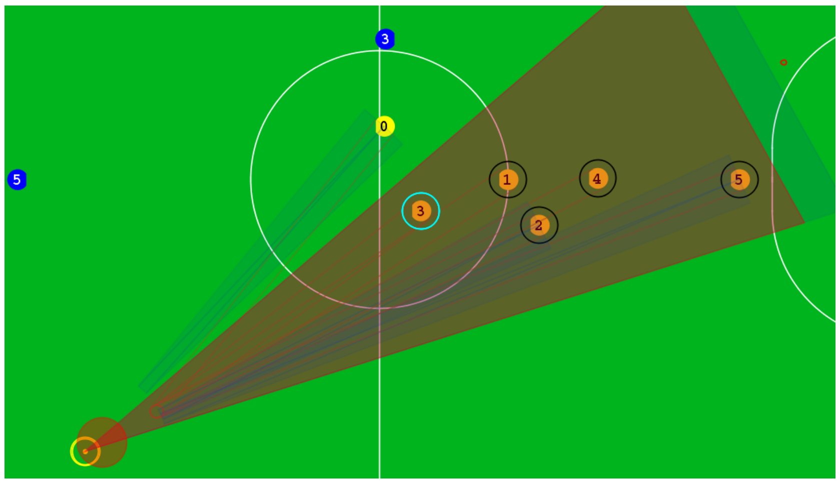


Fig. 7: Choosing the best robot to catch a rolling ball. The red triangle visualizes the direction of the pass. All robots with a black circle are potential receivers. The best robot is marked with a cyan circle.

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

– Needed calculations for pass synchronization have not been accurate enough. We had to use quite big thresholds to work against this inaccuracy. Consequently, 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 the future. We are convinced that in 2017 our offensive will be able to compete head on with the best teams in the SSL.

2.3 Defensive

The defense described in [5] has been about optimizing defense positions and selecting the correct 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 based on their distance to our goal. Therefore, the opposing team could pile as many robots in one corner of the field as there are defenders, 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 it receives the ball.

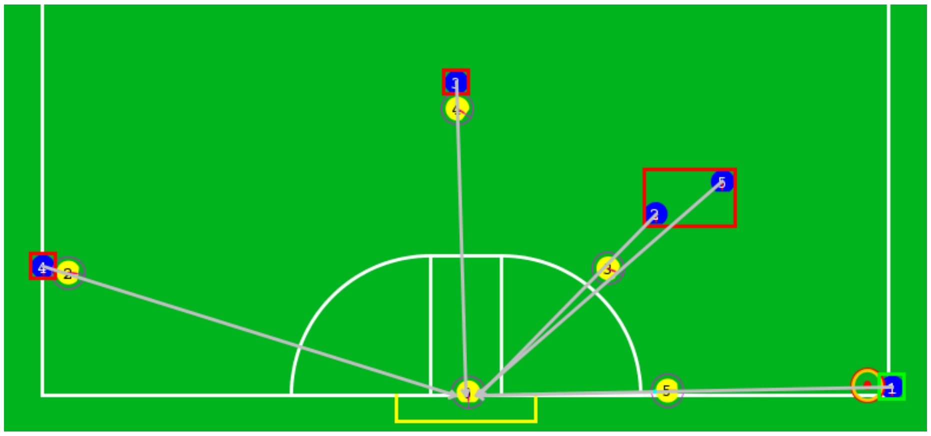


Fig. 8: Distributing defenders using groups

Group concept To address the weakness described above, a new group concept was implemented. When analyzing offensive positioning of the opposing team, robots are grouped, if they have a similar angle to our goal center. The angle necessary to create a group is smaller than the one needed to dissolve it. This avoids oscillating creations and dissolutions of groups.

When distributing defenders, each group is assigned a defender before another group gets a second one. Figure 8 shows a typical corner situation, visualized in our software. Robots in the same group are enclosed by a rectangle. By default the rectangle is red. It is green if the group is nearest to the ball. The previous implementation would have assigned the defender with 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 bot to interfere with his positioning and intercept passes.

More defensive concepts Besides 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 intercepting the kick 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 our defense in replays.

Strengths and weaknesses The defense has 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 can potentially still toggle. This leads to holes in the defense because the defenders never reach their target position.
- It is often difficult to decide how many defenders to use in which situations.
- The defense only considers bot positions. In many situations it would be helpful if the defense tried to interrupt passes, or block kicks actively.

2.4 Keeper

In the last two years, the keeper's 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 shots on goal can happen, the keeper stays 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 balls slightly faster and might prevent scoring a goal.

Move into penalty area To save time, the keeper almost 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 has returned to the penalty area. Consequently, this state is active if the keeper is outside of the penalty area and moving 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 of 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 throw itself into the ball travel 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 ball travel line in time with the robot stopping there, it tries to overaccelerate 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 into 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 in possession of the ball while being positioned 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 while still being 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, a free-kick is awarded to our team.

Chipping ball out of penalty area After successfully stopping the ball inside the penalty area, the keeper has to chip it away. To achieve this, 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.

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.

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