Rework of the Coordination of Offensive Actions in the RoboCup Small Size League

Study Report

for the study program Information Technology at the
Cooperative State University Mannheim

by

Ulrike Leipscher

Matriculation number:	*6188730*
Course:	*MA-TINF15ITIN*
Supervisor:	Prof. Dr. Jochem Poller

Declaration

I hereby declare that the report submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

Oberpfaffenhofen, June 29th, 2018

Abstract

In the Small Size League (SSL) soccer competition of the Robot World Cup (RoboCup), fast passes between robots are the main tool to overcome the defence of the opponent team and allow for successful goal kicks. Most teams have developed tactics to intercept passes and gain ball control. Therefore, reliable analysis of the current situation and the decision if a pass is feasible are necessary. In this work, different pass rating functions are introduced and tested. Focusing on simplicity and reliability, the pass rater use distances of the opponent robots to the pass line as basis for their ratings. An analysis of simulated soccer games using the new pass raters shows similarly high success rates for passes. Finally, the implemented software allows simple exchange and configuration to adjust to different opponents.

Contents

List of Figures

Abbreviations

AI	Artificial Intelligence
RoboCup	Robot World Cup
SSL	Small Size League
TIGERs	Team Interacting and Game Evolving Robots
UI	User Interface
UML	Unified Modeling Language

1 Introduction

The work presented here is based on a project organized and maintained by students of the Cooperative State University Mannheim. In the project Team Interacting and Game Evolving Robots (TIGERs) Mannheim students develop and program robots that can play soccer in the Small Size League (SSL) of the RoboCup. In the following sections an insight is given to the project and the RoboCup and SSL in general.

1.1 RoboCup

The RoboCup or more precise the Robot World Cup Initiative was established in the 90th of the last century as the next big long term challenge in robotics and AI [2, 3]. Leading scientists proposed the following goal:

> By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup [3].

Using soccer as the motivating factor to improve the development of more advanced technologies in intelligent robotics, the first international RoboCup soccer games were successfully held in 1997 in Nagoya, Japan with over 40 participating teams in different leagues (real and simulated) [2]. Since then, RoboCup has grown into a well-established institution in the field of robotics, not only covering several leagues in (non-) humanoid robot soccer and simulated soccer, but also other fields like logistics, as well as robotics @Home, @Work and in the field of disaster rescue.

The RoboCup competitions and conference are held annually all over the world, still with the before mentioned goal in mind. Compared to previous challenges in AI, like developing a chess software that is able to beat the best chess players in the world, robot soccer offers great research potential in areas that are closer to real

world problems. As the game is highly dynamic and usually not all information about the environment is available, research in fields like vision, real-time sensor-fusion and planning, reactive behavior and motor control will be stimulated. Furthermore, strategy acquisition, learning, context recognition and strategic decision-making are just a few additionally named topics that are promoted through RoboCup all in the context of multi-agent systems [3].

1.2 SSL

Figure 1. A SSL soccer game at the RoboCup 2016 in Leipzig, Germany.

The SSL is one of the two non-humanoid, as well as one of the oldest soccer leagues at the RoboCup. Two teams with each eight driving robots that fit into a 180 cm diameter circle and with a maximum height of 15 cm play soccer against each other. Those robots can be engineered freely by each team with the only restriction in size, ball shoot speed and ball coverage. Figure 2 shows an example robot in close up developed by TIGERs Mannheim. The game is conducted according to the SSL rules (see [1]) by an autonomous referee (developed by various teams of the league) that is overseen in last authority by a human referee. It is played with an orange golf ball on a 12 m times 9 m green carpeted field (according to the latest rules for division A [1]).

In figure 1 a photo, taken during a soccer game at the RoboCup 2016 in Leipzig, Germany, is shown as example. Both teams receive vision data from a central vision system consisting of multiple cameras 4 m above the game surface [4]. The vision data of all cameras is beforehand processed by community maintained, open source software called SSL-Vision [4] that recognizes the color patterns on top of each robot as well as the orange ball and geometric data of the field. Therefore, each team receives information about the positions of all robots and the ball on the field. Next to the field, each team has a private computer system that is used to process the position data received from the vision, as well as sensory output from their own robots. Each team develops their own AI to control their robots and game strategy. The SSL is the only RoboCup soccer league with a common vision and a hybrid centralized/ distributed robot control system. It specializes in multi-agent coordination and strategy planning in a highly dynamic environment [4].

1.3 TIGERs Mannheim

Figure 2. A TIGERs soccer robot of the newest generation at the RoboCup 2016 in Leipzig, Germany.

TIGERs Mannheim is the robot soccer team of the Cooperative State University Mannheim, founded in 2009 by a group of information technology students. Engi-

neering their own robots (see fig. 2) and developing their own AI called *Sumatra* (see section 2.1 for a detailed description), they participate in the SSL of the RoboCup since 2011 with great success. Since 2014 the team belongs to the top 8 in the SSL world cup and became European champion in 2016. Several other honors were awarded during the recent years, for example for the best team description paper, open source and excellence. During the last RoboCup in 2018, the team could celebrate its biggest success in the worldwide competition so far with the 3rd place in the SSL soccer games.

1.4 Aim of the project

The robot soccer team TIGERs Mannheim playing in the RoboCup SSL strives to improve its AI every year. The participation in the competition of RoboCup 2017 showed deficits in the coordination of offensive and supportive robots. Offensive robots aim to shoot goals while outplaying the opponent team with passes between its team members. Supportive robots, that neither belong to the defense nor offense, search for optimal positions to receive passes by the current offensive robot. To evaluate the possible receive positions called pass targets various rating functions are applied and combined to decide if a pass could be received and if it will be a good position to shoot a goal after the ball was received.

During simulated and real soccer games, one of the two following scenarios can often be seen. On the one hand passes very often get intercepted by the opponent team as it reaches the ball faster than the team mate, or on the other, the robot receiving the ball can not handle it fast enough before an opponent reaches the ball. Therefore the evaluation of the pass targets needs improvement. This work aims to evaluate new rating functions and concepts that will allow better coordination of the offensive actions during game play in the SSL.

2 Current Software Architecture and Concepts

This project focuses on the improvement of the AI only and therefore is a pure software work. The central software of the TIGERs soccer team is called Sumatra, written in Java. In the following sections its architecture and basic concepts will be described.

2.1 Sumatra

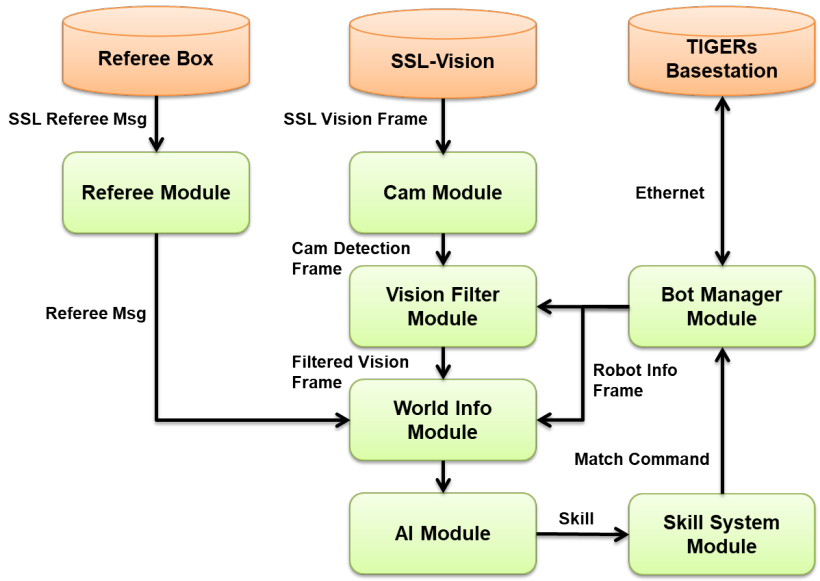


Figure 3. Main software components of Sumatra (green boxes) and their data exchange with external components (orange boxes) [6].

Sumatra is build of several modules with different tasks. In figure 3 the main components and their interactions are shown. All information available from external sources (referee box, SSL-Vision and our robots) is taken in and processed by the

corresponding modules (see fig. 3: Referee Module, Cam Module, Bot Manager Module). The output after processing the external information is collected in the world info module generating a `WorldFrame` which serves itself as input for the AI module. In the AI module the strategy and future actions are calculated which results in selecting the next skill for each robot on the field. The last step before sending the actual commands to each robot consists of calculating the next position, orientation and kicker and dribbler configurations for each robot according to the chosen skill in the skill system module.

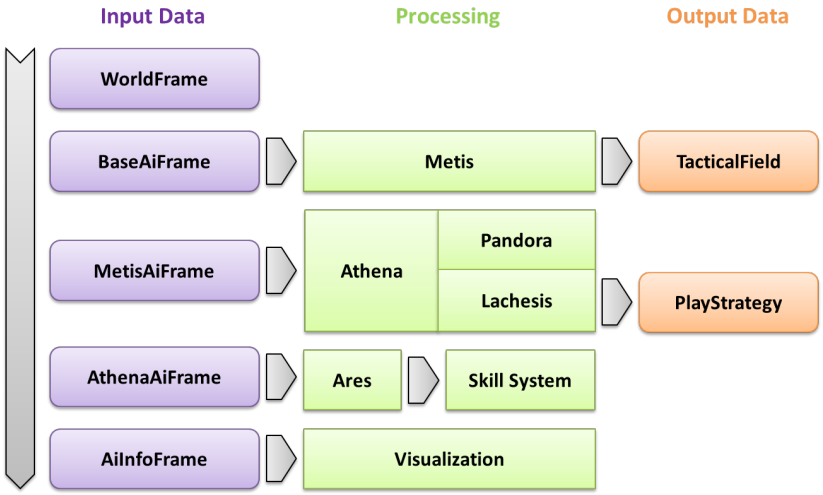


Figure 4. Data processing in the AI module of Sumatra [7]. Input data (violet boxes) is generated by the level above. Additional data output is displayed in orange boxes.

The AI of the TIGERs is based on a play - role - skill system. Skills are low level tasks like a goal kick, a pass, receiving the ball or simply moving to a position. These skills can be executed by any role. Roles basically define the actions of a single robot. They combine different skills into useful sequences of actions, for example a offensive role could do the following: receiving the ball, turn with the ball and shoot at the goal. They consist mainly of a state machine that decides which skill is the best to execute in the current situation. Information about the environment is taken from the `WorldFrame` which is further processed by the AI sub module `Metis` to analyze tactical information and collect it in the so called `TacticalField` (see fig.

4). Above the roles, Sumatra uses plays to coordinate several robots with similar roles and group them. Currently there exist mainly four different plays in the Sumatra:

- The **Keeper Play** has only one role, the keeper, and does protect the own goal from opponent goal shoots.
- The **Defense Play** coordinates the different defense roles according to a threat based system. Depending on the ball and opponent locations, the opponent robots pose different threats for our defense and will be blocked in front of the defense area.
- The **Offensive Play** usually has one or two roles that have to be coordinated especially for passes between the robots. Further details will be described in section 2.2 below.
- The **Support Play** coordinates all robots that do not belong to any other play mentioned before. It globally searches good positions on the field where these robots can best support the offensive robot in shooting a goal, either by receiving passes or distracting the opponent. See section 2.3 for further details.

In general the AI module consists of several sub modules as shown in figure 4. The first step to generate the strategy for the game is to use the collected world information from the `WorldFrame` and analyze the tactical information in `Metis`. This sub module consists basically of a list of calculators that process the position and velocity information of all robots and the ball to calculate different important aspects for the game, like ball possession or which robot can reach the ball the fastest. With the collected data, the sub module `Athena` decides which plays to execute and the specific role assignment. `Athena` is therefore further subdivided into `Pandora` which chooses the roles and plays and into `Lachesis` which assigns the chosen roles to specific robots. `Lachesis` is often simply called `RoleAssigner`. In the third step, the skill for each role is chosen and its specifics are decided in `Ares` and the skill system, respectively. These steps are calculated for each incoming vision frame with

a frequency of about 60 fps in the represented order. It allows for dynamic game play and fast reactions of the robots.

2.2 Offensive

The offensive play calculates a strategy with the objective to shoot a goal in the near future. The offensive strategy first considers each robot independently from its current role. For each robot the viability of all `OffensiveActionMoves` is calculated. `OffensiveActionMoves` are for example a forced pass during an indirect free kick (see official rules in [1]), a direct goal kick, a clearing kick near the own defense area or a standard pass (further description in [8]). First it is determined if an action is viable in which viability can be `TRUE`, `PARTIALLY` or `FALSE`. Actions with viability `TRUE` are the best choice for execution, `PARTIALLY` viable actions might be executed depending on their `viabilityScore`, actions with `FALSE` viability are not considered in any further calculation. In addition to the viability, actions are ordered according to their priority. This means an action that is higher in order has a higher priority and will be preferred to an action with same viability and lower priority. An actions is executed if it has the viability `TRUE` and the highest priority compared to other actions with viability `TRUE`. If only actions with viability `PARTIALLY` are available, all actions are ranked by a further calculated `viabilityScore` which determines according to predefined criteria how successful this actions might be on a scale between 0 and 1 [8].

To evaluate most of the actions (like passes and goal kicks) specific points on the field have to be rated according to their goal kick or pass chance. Therefore in a lot of situations, the offensive strategy uses so called pass targets (see section 2.4) to calculate the appropriate viability scores. To find the best possible option for the next strategy, the rating functions for passes and goal kicks are crucial.

2.3 Support

The support play tries to position all available robots with the objective to receive future passes and have a good goal chance. Positions are chosen randomly on the field and evaluated by two factors: first the chance to successful pass the ball from its current position to this support position and second by the chance to successful score a goal from this position. For both options, the best positions are chosen and robots with a supportive role are send there. To avoid grouping of robots, positions close to already chosen ones are removed from the generated support position list. To keep positions relatively stable, the chosen positions from the previous frame are kept and rated again in the next frame together with new candidates. If the old ones are as good as the new ones, the old positions are preferred [8]. Furthermore, the supportive roles are those that get additional pass targets as they are possible new pass receivers. These will be described in the next section.

2.4 Pass Targets

Generally spoken, pass targets are possible pass positions on the field that are rated by there score chance and by their pass chance. Theses chances are expressed as double values between 0 (bad) and 1 (good). In previous versions these scores were defined by the 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 of these factors were weighted by their own respective weighting factor [5]. Unfortunately, the results were not satisfying as many passes could not be received due to interceptions by the opponent team. Therefore new ways for the evaluation of possible pass positions are tested.

2.4.1 Pass Target Calculators

Pass targets are generated, rated and selected in three respective calculators in `Metis`:

1. Pass targets are randomly generated in a defined area around supportive robots in the **`PassTargetGenerationCalc`**. The area is defined by a moving circle with its center moved from the current robot position P_t by a vector defined from the current robot velocity v_t and its maximum brake acceleration a_{max}:

$$center = P_t + (v_t * (v_t/a_{max})/2) \tag{1}$$

   The radius is dynamically calculated by the time the ball needs to reach given center position t_{ball} and again the maximum acceleration a_{max} of the robot:

$$radius = a_{max} * (t_{ball}/2)^2 \tag{2}$$

   Each generated pass target candidate is verified by certain parameters, for example if the point lies in the field and outside the defensive areas. If it is a legal point it will be saved for the next step, the actual rating.

2. Each pass target that survived all previous inspections, will be rated in the **`PassTargetRatingCalc`**. There are mainly two rating functions at work: one to rate the actual pass chance and a second one to rate the score chance from the new position. Afterwards a combined score is calculated that allows for comparison between different pass targets.

3. In the last step, the **`PassTargetSelectionCalc`**, rated pass targets are sorted by their total score and for each available supportive robot a defined maximum number (usually 5) of pass targets is selected and stored in the `TacticalField` (see previous section 3).

2.4.2 Known issues

The aim is to find a simple rating method that can be applied in any situation. One problem might be that receiving a pass and shooting a goal are two totally different objectives and their rating can not simply be combined to a final score. Though choosing the best pass target can only be achieved, if pass chance as well as score chance can be combined. The need to also use the score chance when evaluating a pass position lies in the fact that passes should bring the robots closer to the opponent goal to allow for goal kicks. Therefore if the score chance is included in the pass target rating, pass positions automatically are driven to the opponent goal as score chances are usually higher the closer the position is to the goal.

Another issue comes from the two possible types of passes: straight and chipped kick. Both types need their own rating function as chipped passes can not be intercepted while the ball is above robot height (15 cm [1]). Again these two scores can not simply be combined to one final score though keeping both scores for the `OffensiveStrategy` to decide which to use, might be a possibility.

Last but not least the pass rating function itself did consider too many influencing factors that could not be easily combined to one score. So far rather complicated weightings were used to calculate the pass rating with unsatisfying results. For this work a few simpler methods were tested and compared which are further described in the next chapter.

3 Methods

In the following sections, three new pass rating methods are introduced which each consider chipped and straight passes. In addition a new way to combine pass and score chance will be put forward.

3.1 Pass Target Rating Functions

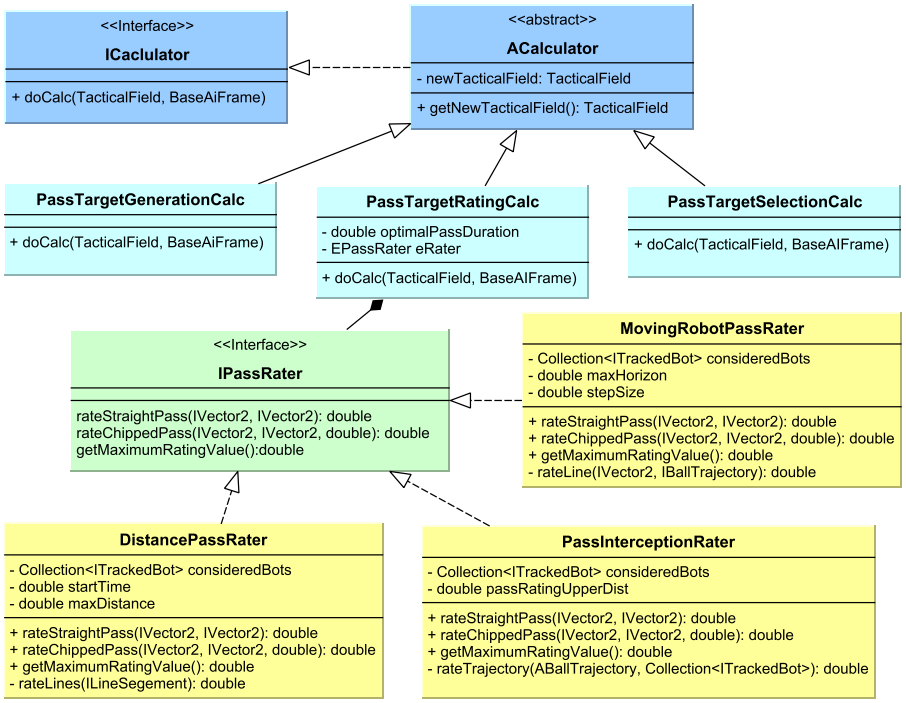


Figure 5. UML class diagram of the pass target calculators and their relation to the pass raters.

Pass targets define possible positions the current offensive robot could pass to. Therefore the ball should travel between its current position and the respective pass target without any disturbance of the opponent team. There are various ways to

determine if the opponent team has a chance to intercept the pass. Three different approaches were implemented as pass rater classes as shown in figure 5 (yellow boxes). The three `Metis` calculators for the pass targets (light blue boxes, see also section 2.4) share the calculated information by saving data in the `TacticalField`. The `PassTargetRatingCalc` then uses one of the pass rater classes which all implement the `IPassRater` interface for easy exchange, to rate the pass target candidates. Each pass target is rated once as straight pass and once as chipped pass. The three implementations are further described below.

In addition to the pass rating, the `PassTargetRatingCalc` also calculates the score chance for each pass target (not shown). The goal kick rating was maintained from the previous implementation with only slight adjustments. It considers all opponent robots between the goal and the shooting position. Stepping through discrete time steps, the movement of the robots is calculated from their current velocity. Their calculated position allows to determine how much of the goal each robot covers at that moment. The rating becomes worse the more the goal is covered if you look at the goal starting from the current shooting position. Often after receiving the ball, the receiver has to turn to the correct angle for a goal kick. In the previous version of this rater, the time to turn around was neglected. This was corrected by adding a time to turn that has the the turn angle as a factor. The additional time is 0 s if the ball can directly be redirected at the goal which is the case for angles smaller than 60°.

3.1.1 DistancePassRater

In a simple geometric approach the `DistancePassRater` calculates the distances of each opponent robot (keeper excluded) to the pass line (ball travel line). The rating can become 1 (best score) if all opponent robots are further from the pass line than a configurable distance (see fig. 5: `maxDistance`, default: 3 m). The calculation for chipped and straight passes is basically the same with the slight difference that the

considered ball travel line starts at the first touchdown position of the ball where it can be received again. This is a simplified view of the pass line. Though it is possible to calculate the correct line segments where the ball is below robot height, this would give several segments. Considering all line segments where it is possible to intercept the ball, makes further calculations too costly in terms of performance. Vision frames are received every 16 ms and with each frame all calculations in Sumatra are run again to adjust the game strategy to the latest information. Therefore time consuming calculations have to be kept to a minimum always considering the balance between performance and the error made by simplifying information. When a ball is chipped, the first part of the ball travel line is the highest (usually above robot height). Though the following jumps after the first touchdown might be above robot height, it is very likely that the ball can be intercepted during most of its remaining travel line. Therefore this approximation is made to simplify the rating.
Passes through the opponent defense area are favored by the rating because defending robots (besides the keeper) are not allowed to touch the ball inside the defense area. Therefore the pass line segments that lay inside the defense area are excluded from the distance rating. For straight kicked passes, an additional configurable parameter, called `startTime`, can be set to start the ball travel line from this time point. As the `DistancePassRater` uses such a simple geometric approach, all positions on the field would get a pass score near 0 when an opponent is close to the ball. To avoid this situation, either the rating for chipped passes has to be used or the start time of the pass can be set to a small value, artificially moving the pass line further away from the opponent. The default value for `startTime` is set to 0.1 s. This adjustment to the ball travel line can be done because a kick is the fastest when it was just executed. Any robot that is not already on the pass line can not reach the ball in the first few milliseconds as its reaction time would be a few vision frames.

3.1.2 MovingRobotPassRater

The `MovingRobotPassRater` uses a more advanced geometric approach, considering also time for movements and velocities of the robots. Similar to the `DistancePassRater` (see above, section 3.1.1), it determines distances to the travel line of the ball though not from the robot itself, but a moving circle calculated by the velocity and acceleration of the robot (see fig. 6). The calculation of the moving circle is very similar to the one used to calculate the area where pass targets can be generated (see section 2.4). This is done for each time point given the configurable `stepSize` (default: 0.1 s).

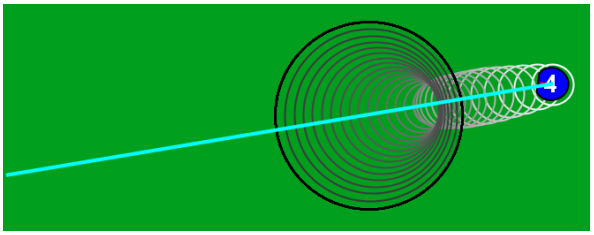


Figure 6. `MovingRobotPassRater` calculates circles around the moving robot for discrete time steps (here with `stepSize` = 0.025 s). Circles are drawn darker the further in time they are. Cyan line shows the velocity vector of the robot.

Listing 1 shows the basic rating algorithm with the ball trajectory and the target point as input. It steps through each time point searching for the smallest relative distance between the moving circles and the ball travel line. While it is stepping through the time points, the ball travel line is shortened as the ball moves forward. In contrast, the moving circles become larger because there is more time for the robot to accelerate and move in any direction (see fig. 6). The time frame in which moving circles are generated is capped to a maximal time, called `maxHorizon`. This value is also configurable with a default value of 0.7 s. So the moving circles do not get any larger than at 0.7 s even if the time to reach the target point is larger than this. The rating can become 1 (best score) if no moving circle cuts the ball travel line at any given time point. Otherwise the minimal relative distance (always

```
1  private double rateLine(IVector2 target, IBallTrajectory
    ballTraj)
   {
     double tMax = ballTraj.getTimeByPos(target);
4    IVector2 start = ballTraj.getTouchdowns().isEmpty() ?
       ballTraj.getTravelLine().getStart() :
       ballTraj.getTouchdowns().get(0);
7    ILineSegment passLine = Lines
       .segmentFromPoints(start, target);

10   double score = 1;
     for (double t = 0; t < tMax; t += stepSize)
     {
13     IVector2 curPos = ballTraj.getPosByTime(t);
       ILine curLine = passLine.isPointOnLine(curPos) ?
         Lines.segmentFromPoints(curPos, target) : passLine;
16     double dist = getLinesOutsidePenArea(curLine).stream()
         .mapToDouble(l -> getScore(movingRobots, l, t))
         .min().orElse(1);
19     score = Math.min(score, dist);
     }
     return score;
22 }
```

Listing 1. Method `rateLine` from `MovingRobotPassRater`.

compared to the given circle radius) becomes the pass score for this pass target. Similar to the `DistancePassRater`, the rating for chipped passes uses a ball travel line that starts at the first touchdown of the ball trajectory. For pass lines that cross the opponent defense area, the segment inside this area is excluded from the rating to favor passes through the defense area. In listing 1 in line 16, the method `getLinesOutsidePenArea` checks this condition and returns a list of line segments outside the defense area.

3.1.3 PassInterceptionRater

Compared to the other raters, the `PassInterceptionRater` uses the most advanced calculation. It rates how well each opponent robot could intercept the pass line by constructing a trajectory for the robot that assumes the robot wants to stop as fast as possible. Then the distance from the resulting brake point to the closest interception point on the pass line is used to rate the pass target. Similar to the `DistancePassRater` the best score can be reached when the brake points of all considered robots are further than a configurable distance from the pass line. The default value is again 3 m. As in the other raters, chipped passes only consider the ball travel line starting with the first touchdown point of the ball.

3.2 Calculation of final score

After each pass target candidate received a pass score for chipped and straight kicked passes, as well as a goal kick score, these separate scores need to be combined to one final score. This final score will determine the selection of the best pass target in the next game strategy.

So far all pass ratings only consider the opponent robots and calculate distances to the pass line in their own respective way. Usually this is done for a defined point in time or a maximal time span. As long passes across the field take more time, the opponents have more time to react and intercept the pass. Due to wanting a simple rating method, long passes are usually not rated any worse than short ones which does not mirror the actual situation in a game. To compensate for this misjudgment, the pass ratings gain a second factor: the pass duration. The pass duration is considered optimal if it is less or equal than a configurable value with a default of 1 s. Therefore these passes are multiplied with the factor 1 (best value) as only there previously calculated rating decides the score. Passes that would take two times this value (here

2 s) are multiplied by 0 and therefore their pass score will be 0 as well. If the pass duration lies in between 1 s and 2 s, the factor will be reduced linearly.

The decision, if the next pass will be chipped or kicked straight, comes from the current offensive role. Therefore the pass target calculators do not explicitly decide which pass score (chip or straight) is the right one. But as usually only one pass score is used for further calculations, the final pass score is set equal to the maximum score of chip and straight pass score times their respective duration factor. The last step is to calculate one final score with the pass and goal kick ratings. To receive this final score, different weighting functions can be applied. These are described in the section below.

3.2.1 Weighting functions

The selection of the pass targets can be done by sorting all rated positions by their final score. Pass score and goal kick score can be combined with different weighting concepts:

1. **Constant weight**: Pass score and goal kick score are added after each was multiplied with a constant factor, resulting in a score between 0 and 1 again. A factor of 0.8 for the pass score and 0.2 for the goal kick score was chosen. That way, the pass chance is prioritized over the goal kick chance to assure passes without interception by the opponent team.

$$score_i = passScore_i * 0.8 + goalKickScore_i * 0.2 \tag{3}$$

2. **Situation weight**: Pass score and goal kick score are multiplied with a factor that changes depending on the current situation, more precisely the x coordinate of the ball position. If the ball is close to the own defense area the pass score

gets a higher factor than the goal kick score and vise versa if it is closer to the opponent defense area.

$$\begin{aligned} score_i = passScore_i * (1 - weight(x_{ball})) \\ + goalKickScore_i * weight(x_{ball}) \end{aligned} \tag{4}$$

The two concepts are applied and compared by pass success rate and the amount of goal kicks and goals during 15 min of simulation in the Sumatra User Interface (UI). For comparison, the pass score is used as sole final score. This way, the most secure pass options should be selected as best pass targets, expecting more passes back to the own half and less towards the opponent goal.

3.2.2 Success Probabilities

As mentioned before, pass rating and goal kick rating are conceptually two very different scores. The combination of the two scores with the help of weighting functions as described above, does not guarantee that the chosen pass target is really the best one. A totally different approach might be to compare the pass scores (or goal kick scores respectively) of all pass targets between each other. A real probability p of success in comparison to the other pass targets can be calculated with the following formula:

$$p_i(score_i) = \frac{score_i}{\sum_{i=1}^{N} score_i} \tag{5}$$

The probability that a pass followed by a goal kick from the receive position (= pass target) succeeds, can than be calculated by simple multiplication of the two corresponding probabilities for pass and goal kick:

$$p_i(pass, goal) = p_i(pass) * p_i(goal) \tag{6}$$

The offensive strategy always needs to analyze if the current ball position has a good score chance compared to passing to another robot and shooting from the new position. For direct comparison, the current ball position or during an ongoing pass the expected receive position are added to the pass targets when calculating the success probabilities.

Though this approach allows for better comparison of the pass targets, it also comes with a few drawbacks. From equation 6 it is clear that having one of the single probabilities 0 will give a total success probability of 0. This could lead to further problems. For example, the ball is in the own field half and so are the pass targets of the supportive robots. In this situation all pass targets are far from the opponent goal, probably all having a goal kick score of 0. This would mean all total probabilities will become 0 and a sorting by pass chances is useless. To avoid this situation, a small value (default 0.005) is added to each goal kick score. Considering the pass score, it might be very unlikely that all pass targets have a pass score of 0 at the same time. Therefore no value is added as default. Only in the unlikely situation that really all pass targets have a pass score of 0, the pass success probability $p_i(pass)$ of each pass target i is set to $1/N$ with N as the total amount of considered pass targets.

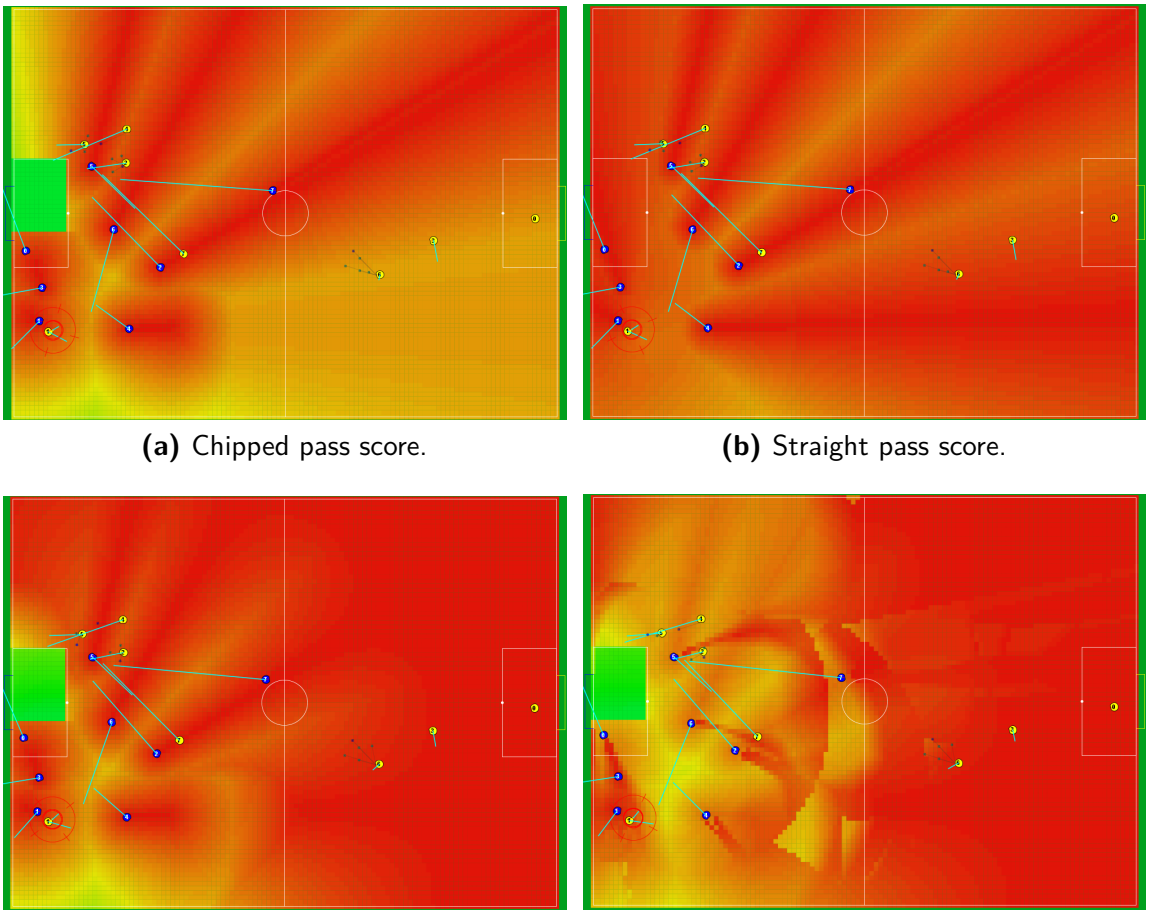
4 Results

In general new concepts in the AI of the game strategy are difficult to evaluate. The best way would be a statistical analysis of the new AI in a real game against other teams. Unfortunately this kind of data is not available so far. Sumatra comes with a simulation tool and a UI. It allows to collect simulated data playing against the own AI which should behave similar to the real world behavior. A small evaluation of the implemented pass raters is done with this tool.

The three different pass raters are compared under several conditions. Each pass rater is combined with each weighting method or the relative scoring (success probabilities). In the next section heatmaps over every position on the field displaying the different pass target ratings will be used for further comparison.

4.1 Visualization of Test Results

For visual comparison, heatmaps with 120 x 90 points on a simulated quatro-sized field (12 m x 9 m) were generated. Each rated point equals an area of 10 x 10 cm. Each point was rated with the respective function using the pass line from the current ball position (center of big red circle) to the rated point on the field. As an example, figure 7 shows ratings for the`DistancePassRater` from the point of view of the yellow team. In 7a and 7b pass ratings for chipped passes versus straight passes are shown. It can be seen that areas around and behind the opponent robots (blue) receive worse scores than areas distant from the opponents. For chipped passes, opponents close to the ball can be over chipped and therefore the scores get higher again with a certain distance behind those robots (see fig. 7a: behind the robot 4 blue). Similar, parts of the defense area receive the best score because the robots before can be over chipped and are not allowed to enter the defense area itself. For the straight passes this is not true as the straight passes can be intercepted



(a) Chipped pass score.

(b) Straight pass score.

(c) Final pass score.

(d) Final score with constant weight.

Figure 7. Heat maps of pass ratings from `DistancePassRater` for the yellow team. From green (best rating: 1) to red (worst rating: 0). (d) shows the final score including goal kick score calculated using constant weight.

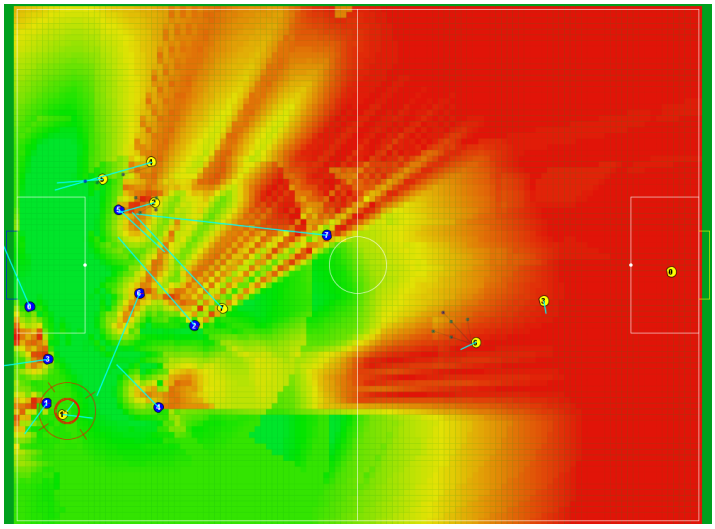
before they reach the safe zone of the defense area. Figure 7c shows the final pass score that is `max(chipPassScore * chipDurationScore, straightPassScore * straightDurationScore)`. With an optimal pass duration of 1 s, areas further away from the ball get worse ratings than when only considering the pass score (compare 7a and 7c). Finally, figure 7d shows the pass score and goal kick score put together to one score with the constant weighting function (see section 3.2.1, equation 3). The appendix containts further figures for comparison, for example

figure 10 showing the final score for the same situation calculated with the situation weight (see section 3.2.1, equation 4). The final score always has better ratings for positions close to the opponent goal because from there goal kicks have a better chance of success. This is due to adding the goal kick score to the rating. The situation weight ignores the pass score almost completely when the ball is close to the opponent goal (see appendix, fig. 10). The areas around the goal get really high scores (green areas) even if passes are not likely to be successful.

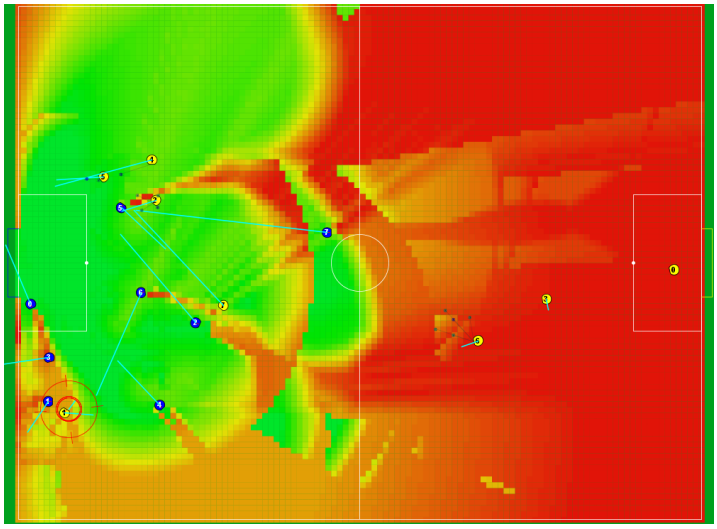
Compared to the `DistancePassRater`, the `MovingRobotPassRater` shows a very optimistic rating (see appendix, fig. 11 to 13). Most positions on the field have high sores (green areas). Only close to the opponents and behind them (looking from the ball), the ratings get bad. Discrete lines with bad scores are visible for robots with high velocity (cyan lines symbolize their velocity vector, see fig. 11, robot 2 blue) due to using discrete time steps to calculate the moving circles. This effect can be reduced by using a smaller step size. The shown figures were generated with a step size of 0.1 s. Reducing the step size to 0.025 s shows smoother results (not shown), but it increases the amount of iterations four times. As calculation times are relevant for the over all performance of the AI and robot control, a less accurate rating is tolerated.

Figure 8 compares the two weighting functions for the `MovingRobotPassRater`. The constant weighting function (fig. 8a) looks very similar to the final pass score (see appendix, fig. 13) as the goal kick score is only a minor part (with factor 0.2) in the total score. With the ball close to the opponent defense area, the situation weighting (fig. 8b) changes the scoring a lot more as the goal kick score influences the final score by 70 %.

The `PassInterceptionRater` behaves very similar to the `DistancePassRater` (see appendix, fig. 14 to 18). The main difference is that bad regions (red) are moved in the direction of the velocity vectors of the considered robots. Furthermore, the opponent defense area is not excluded from the rating. In theory, this rating function should be more realistic compared to the `DistancePassRater` because it



(a) Constant weight.



(b) Situation weight.

Figure 8. Heat maps of pass ratings from `MovingRobotPassRater` for the yellow team. The final score is shown using two different weighting functions.

also considers the current movement of the robots. Nevertheless, predicting the future behavior of opponent robots is risky as their AI and robot control is hardly known. Movements of robots are very dynamic in the SSL and therefore a simpler approach like the `DistancePassRater` might just be as good because it only considers the current situation.

For further comparison, another situation in the game is shown in figures 19 and 20 in the appendix. In this case, the ball lies in the own field half which changes the weight of the situation weight function towards the pass score. Therefore the difference between constant weight and situation weight gets less obvious. Again, the `MovingRobotPassRater` shows the most optimistic rating. The other two raters are quite similar, though the `PassInterceptionRater` is a bit more pessimistic.

4.2 Statistical Analysis

Though visual analysis of the pass rater gives a good first impression on the results, the behavior of the AI during an actual game is the important factor to decide which rater will be the best choice. To collect some data about passes and goal kicks while using the different pass raters, recordings of about 20 min were made in Sumatra for each pass rater and weighting function, respectively. Afterwards those recordings were analyzed from minute 1 to 16. For each recording, passes were counted as well as their outcome (successful, intercepted, missed or disturbed after reception). Furthermore, goal kicks and their success were counted.

Figure 9 shows the results of this analysis. The original counts are shown in table 1 in the appendix A.2. The upper bar graph displays the success rate of the passes. Counted were all passes of offensive robots to their team mates. The pass success rates for the different pass rater do not differ much when using the same weighting function. As expected, using only the pass score as sole criteria for the pass target selection, the success rates are the highest with over 75% for each rater. The more the goal kick rating weighs into the score, the less successful the passes are. For the constant weight with 20% goal kick score, pass success rates are between 67.9 to 73.7%. The situation weight can have a weight of up to 70% for the goal kick score which results in success rates of 61.0 to 68.2%. For the success probabilities, the relative goal kick score and the relative pass score are multiplied which basically means the relative goal kick score is the weighting factor itself for the pass score and

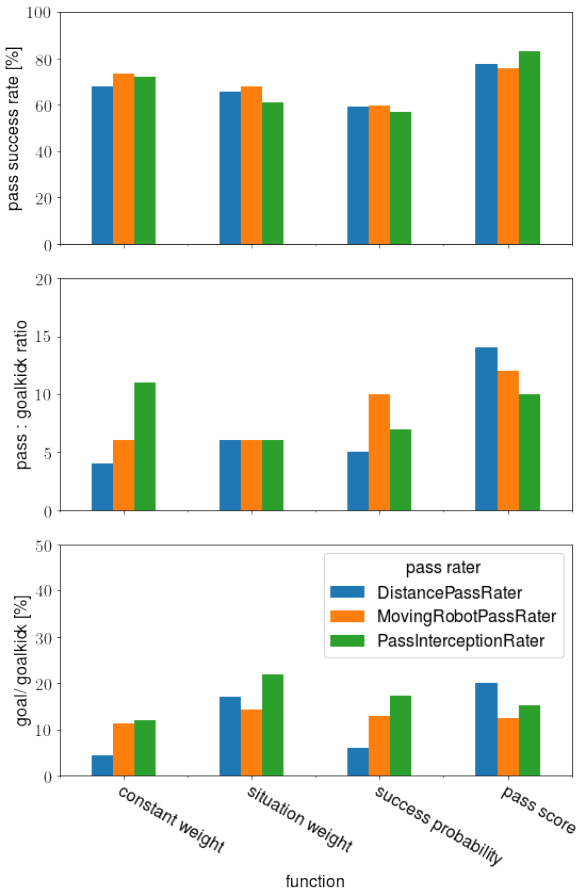


Figure 9. Statistical analysis of the three pass rater in combination with different weighting functions.

can be between 0 and 1. As a result, the pass success rate for the success probability is the lowest with 56.9 to 59.6%.

The second bar graph shows the ratio between passes and goal kicks. It is important that even if the focus of the pass rater lies in successful passes without interception or disturbance by the opponents, the main point for successful passes is to overcome the defense of the opponent team and shoot goals to win the game. Therefore, there should be a reasonable amount of goal kicks compared to pass actions. The ratio is displayed as pass : goal kick which means on x passes follows one goal kick (values are rounded). The ratios go from 4:1 (constant weight/ `DistancePassRater`) to

14:1 (pass score/ `DistancePassRater`). The results do not show a clear outcome. Though it is observable that using only the pass score instead of a weighting function gives the highest pass to goal kick ratios as passes are preferred over goal kicks. This behavior is not desirable.

Finally, the third bar graph shows the goal kick success rate. The values are calculated by dividing the actual goal count by the number of goal kicks. The results vary quite a bit and do not show a clear picture. Between the weighting functions, the situation weight seems the most stable and successful weighting function to achieve goals with up to 21.9% success. Interestingly, using only the pass score shows a good success rate as well with up to 20.0% for the `DistancePassRater`. Though looking at the absolute counts, this rater had the least goal kicks (20, see tab. 1) in combination with the pass score. In absolute numbers, the combination of `PassInterceptionRater` and situation weight could achieve the highest count with 7 goals in the recorded 15 min game. It is also noticeable that the new approach using success probabilities does not perform any better than the other more conservative weighting functions.

Looking once more at the absolute counts in table 1 (see section A.2), the main reason for pass failures is still the interception of the pass by the opponents. Though, one has to consider that the offensive strategy uses two different concepts for passes: the normal pass where the pass position is chosen with the help of pass targets and on the other hand redirected passes that only use the current robot positions as targets. This is done due to making fast decisions for rather fast redirects and the fact that not all pass target positions can be chosen for redirects as the angle needs to be rather small between the incoming pass and the redirect position. For this analysis, these two kinds of passes were not distinguished and therefore any pass failure, independent if it was a redirect or normal pass, is considered.

5 Discussion

In this work, three pass raters have been implemented and tested. All of them show decent results in the simulation with pass success rates up to 83.1%. All of them use rather simple approaches using position and velocity information of the opponent robots to calculate pass scores for the pass targets. Therefore these raters can be applied in any situation in the game.

In terms of performance, the `MovingRobotPassRater` needs the most time to calculate all desired ratings. The AI has maximal 16 ms to develop a strategy before the next vision frame containing new position information is received. The `MovingRobotPassRater` needs up to 4 ms alone, especially if its step size is lower than 0.1 s. Compared to that, the other two raters perform rather well with usually less than 1 ms calculation time. So to use the `MovingRobotPassRater` a compromise between performance and accuracy of the rating function has to be made. The chosen step size of 0.1 ms seems to work well as the shown results were as good as for the other pass raters.

Passes are only a tool to over come the opponent team in a soccer game. Therefore, additionally to the pass raters, new weighting functions and concepts were tested that combine the pass ratings with goal kick ratings. All of them were compared to using the pass score alone as final score. As expected, having a second factor that considers the scoring chance, more goal kicks are performed. Though the results were not consistent for the three pass raters in combination with the different weighting functions, the `PassInterceptionRater` did perform well with up to 21.9% success.

In the end, this analysis does not substitute for the data collected in real SSL soccer games. During the last RoboCup only the `PassInterceptionRater` together with the situation weight was extensively tested. The performance was good though only few goals were shot out of the game against stronger opponents. During RoboCup

another quite different approach also showed good results as it allowed for good scoring chances. The current goal kick rater was replaced with a different rater that especially works with redirect angles. One problem that still exists is that very often robots have to turn with the ball to pass to the next robot. This takes time which allows opponents to approach and disturb the next pass. To avoid this situation, robots that have good positions for redirected shots at the goal are preferred pass targets independent of possible interceptions from the opponent team. As mentioned earlier, soccer games in the SSL are very dynamic, so that calculated future behavior of the opponents is hardly correct and might as well be neglected in some situations. This approach still uses pass ratings for passes in the own field half and to bring the ball to the other half. The difference is that pass score and goal kick score are not combined at all, rather goal kick scores are used solely when any goal kick score above a defined value exists. In the end, to find the best combination of position raters and the way to apply them still needs further work and data analysis.

6 Conclusion

The goal of this study report was to improve the coordination of offensive and supportive robots in the SSL soccer competitions. The work focused on the establishment of reliable pass rater functions as well as weighting functions that combine pass ratings and goal kick ratings.

The next position the current offensive robot passes to is chosen via pass targets. These positions receive a pass score and a goal kick score, respectively. The pass score determines the success probability of a pass from the current ball position to this point. Whereas the goal kick score rates the success of a goal shot from this position. Three different pass raters were implemented and tested in combination with three weighting functions that combine the pass score and the goal kick score to one final score. The final rating is used to select the best pass target.

All raters and weighting functions showed decent results for pass success as well as goal kick success in simulated soccer games. Nevertheless a final decision for the best combination of pass rater and weighting function can not be made from such a small analysis. Three reliable pass raters were added to the AI that can be used and switched easily in real soccer games. All raters run over a newly implemented interface that allows easy exchange. The selection of the pass rater as well as weighting function is possible by simply changing a parameter in the configuration of the AI.

Bibliography

[1] Small Size League Technical Committee. *Laws of the RoboCupSoccer Small Size League 2018*. Tech. rep. RoboCup Federation, 06/2018.

[2] RoboCup Federation. *Website of the RoboCup - A Brief HIstory of RoboCup*. Accessed: June, 26th 2018. 2016. URL: http://www.robocup.org/a_brief_history_of_robocup.

[3] RoboCup Federation. *Website of the RoboCup - Objective*. Accessed: June, 26th 2018. 2016. URL: http://www.robocup.org/objective.

[4] RoboCup Federation. *Website of the RoboCup - RoboCupSoccer - Small Size*. Accessed: June, 26th 2018. 2016. URL: http://www.robocup.org/leagues/7.

[5] Mark Geiger et al. *Extended Team Description for RoboCup 2017*. Tech. rep. Baden-Wuerttemberg Cooperative State University, 2017.

[6] Malte Mauelshagen et al. *Team Description for RoboCup 2012*. Tech. rep. Baden-Wuerttemberg Cooperative State University, 2012.

[7] Nicolai Ommer. "AI Architecture and Standard Game Strategies in RoboCup SSL". Study report. 06/2011.

[8] Andre Ryll et al. *Extended Team Description for RoboCup 2018*. Tech. rep. Baden-Wuerttemberg Cooperative State University, 2018.

A Appendix

A.1 Heatmaps

The following heatmaps were generated by rating 120 x 90 points on a simulated quatro-sized field. All scores are between 0 (bad, red) and 1 (good, green) and define how well a pass between the ball (center of the big red circle) and the given point could be executed. The ratings shown are for the yellow team with the blue team as opponents.

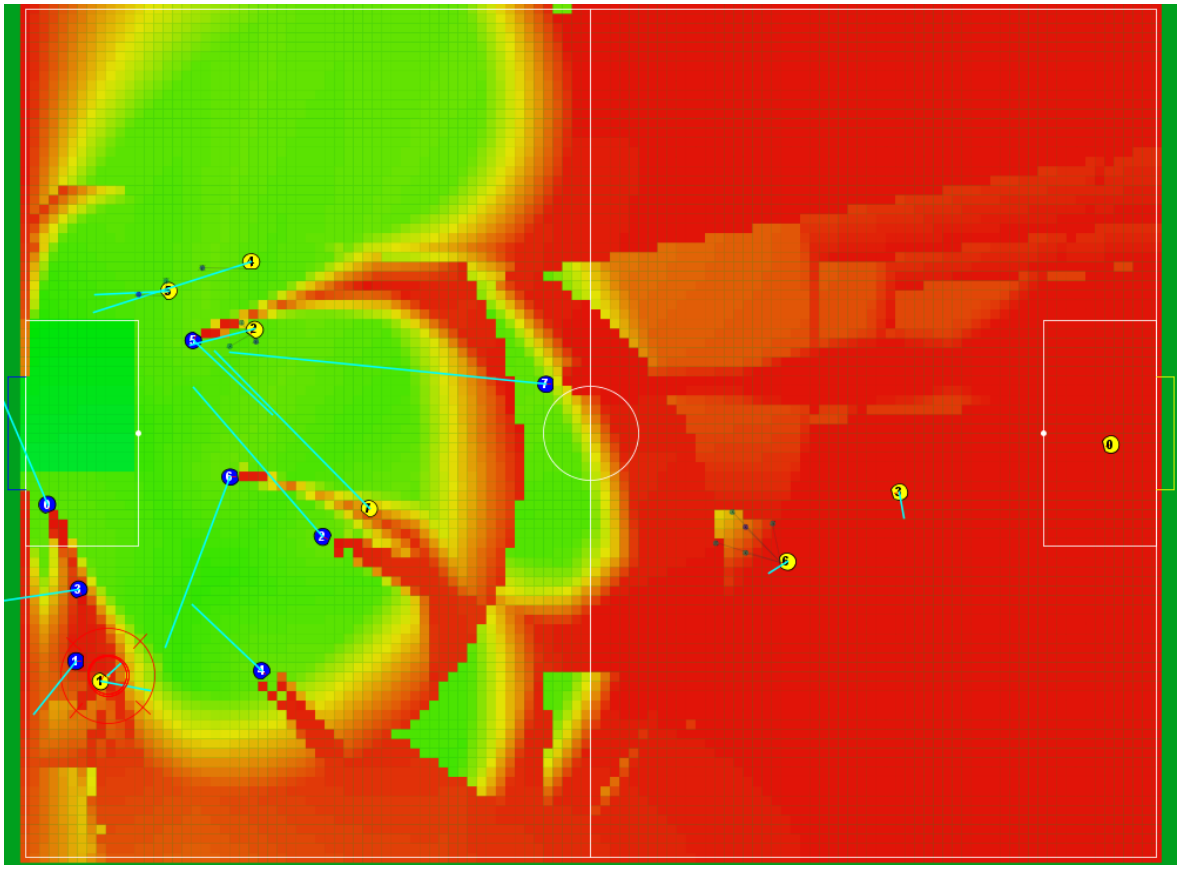


Figure 10. `DistancePassRater`: Final score with situation weight.

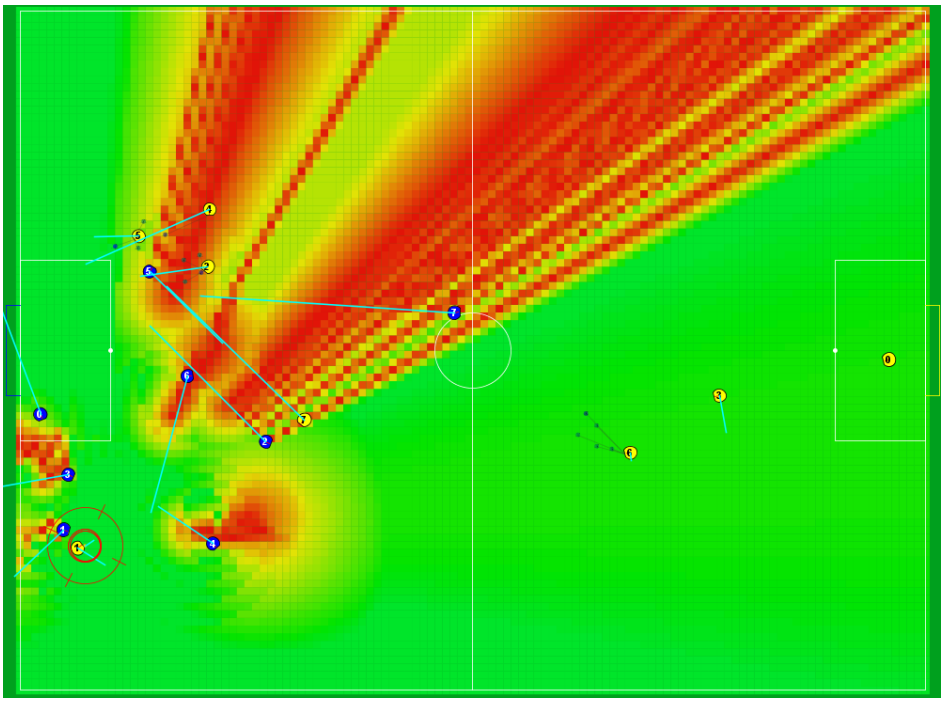


Figure 11. `MovingRobotPassRater`: Pass score for chipped kicks.

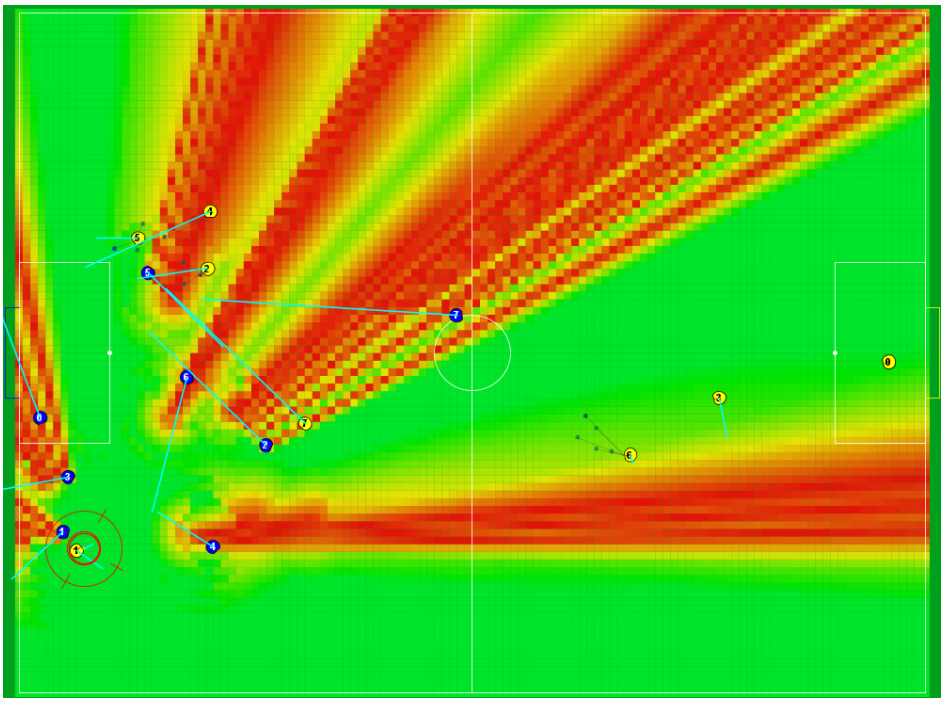


Figure 12. `MovingRobotPassRater`: Pass score for straight kicks.

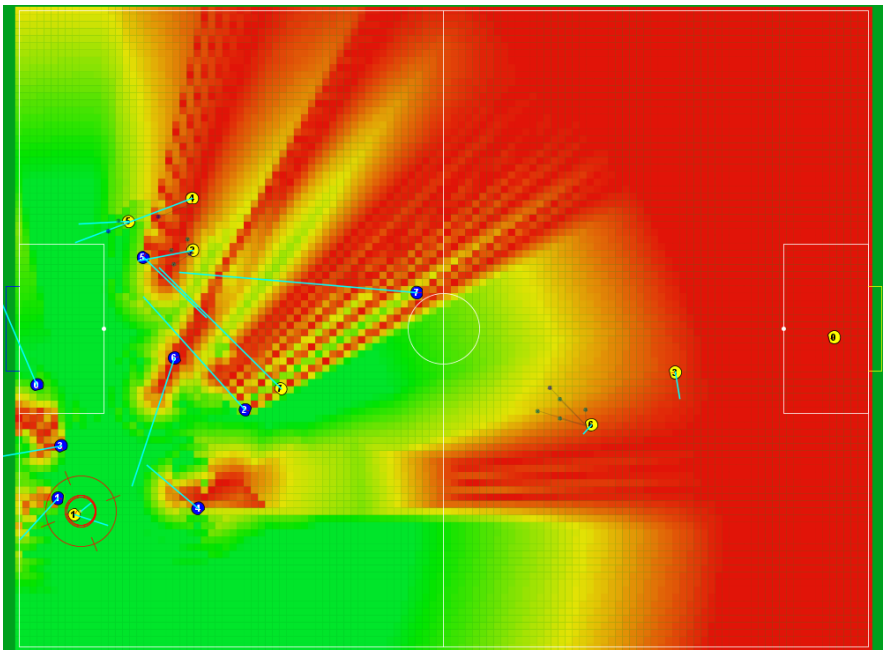


Figure 13. `MovingRobotPassRater`: Final pass score - maximum of chip and straight pass score times the respective pass duration score.

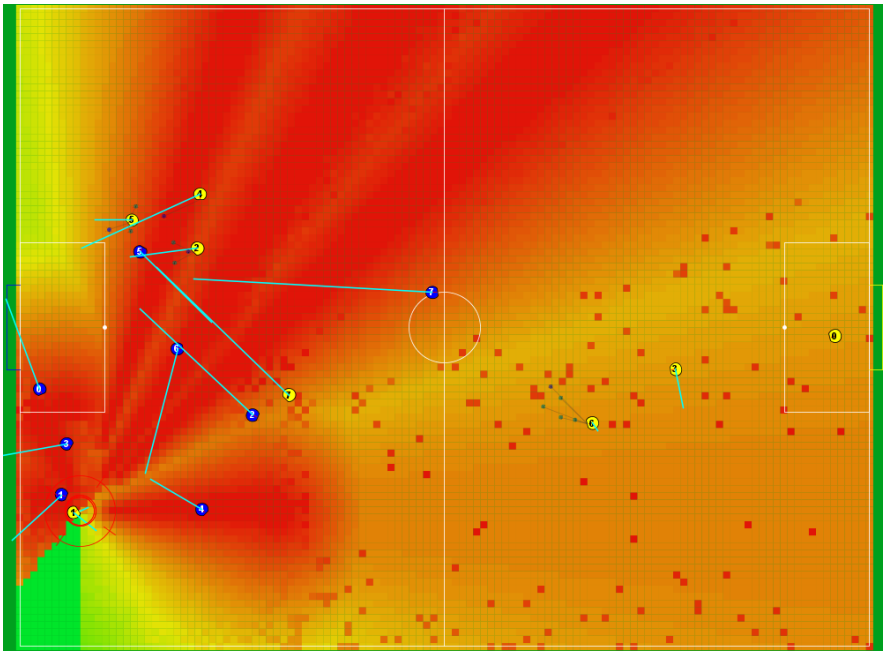


Figure 14. `PassInterceptionRater`: Pass score for chipped kicks.

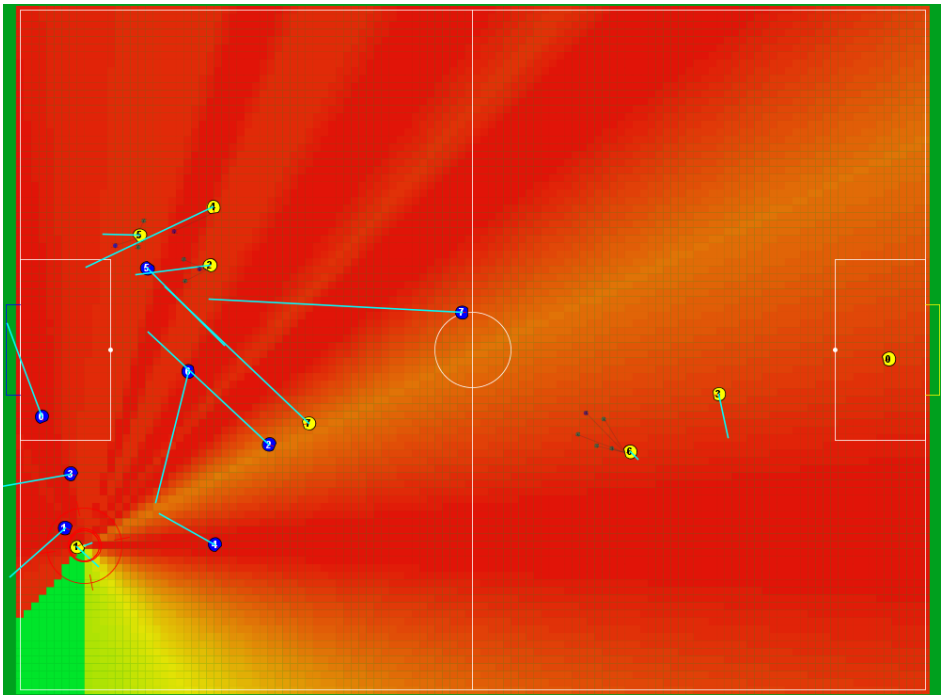


Figure 15. `PassInterceptionRater`: Pass score for straight kicks.

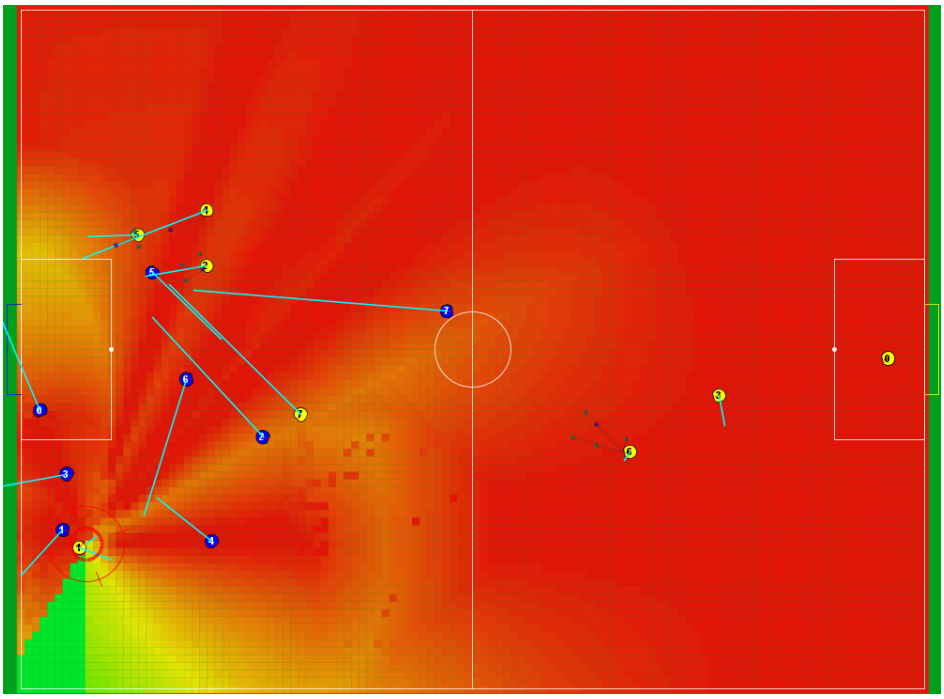


Figure 16. `PassInterceptionRater`: Final pass score.

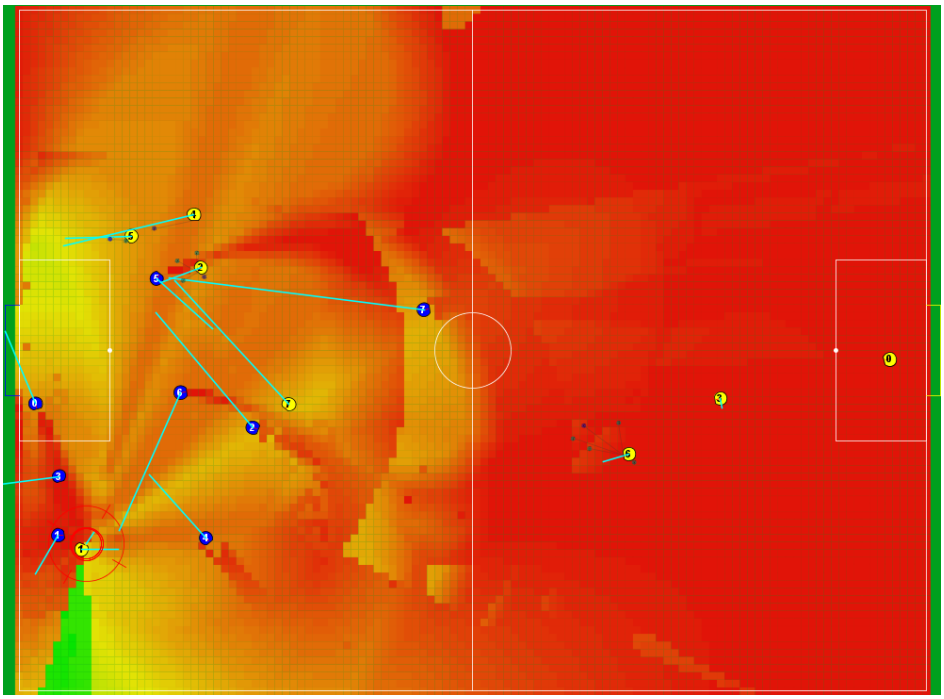


Figure 17. `PassInterceptionRater`: Final score with constant weight.

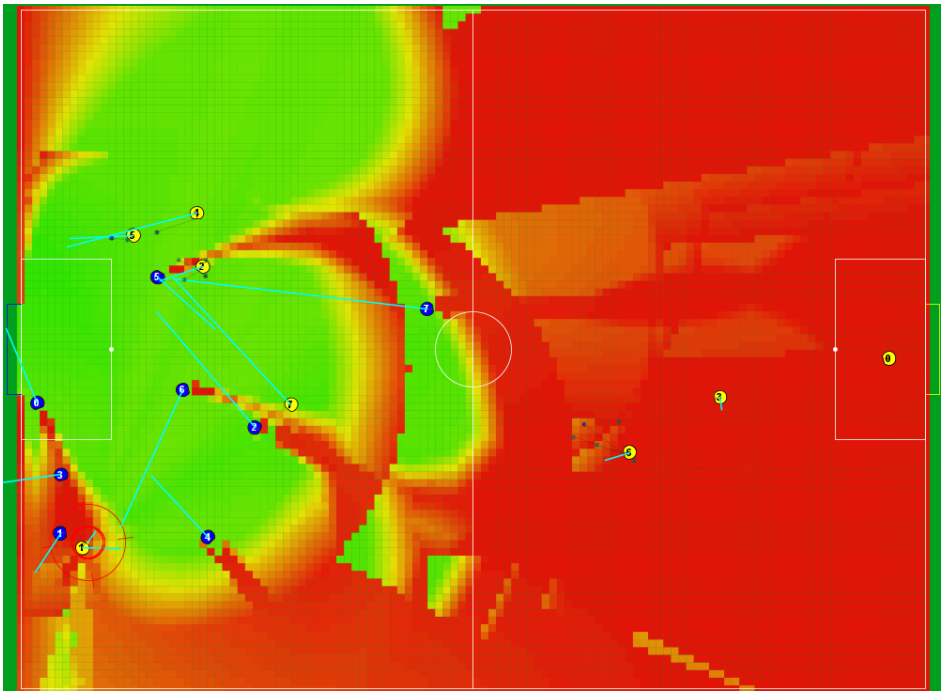
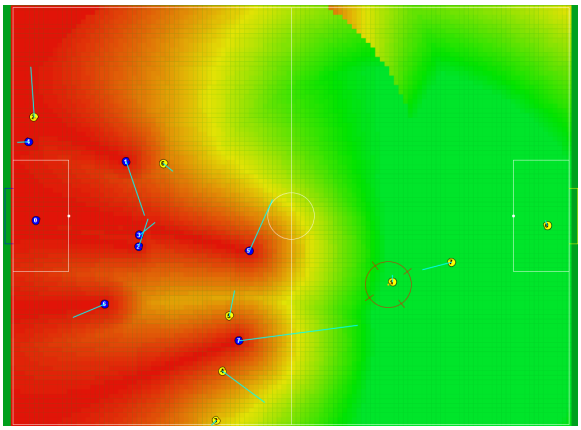
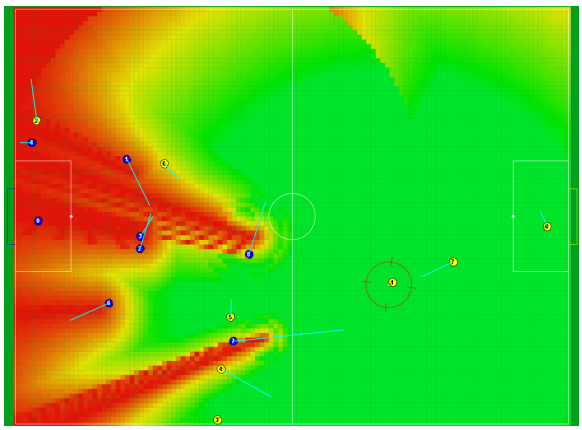


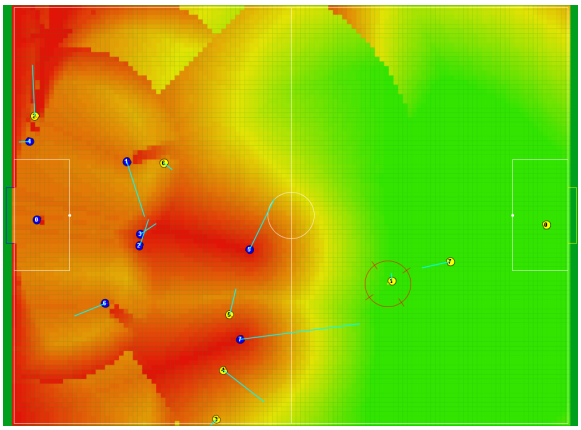
Figure 18. `PassInterceptionRater`: Final score with situation weight.



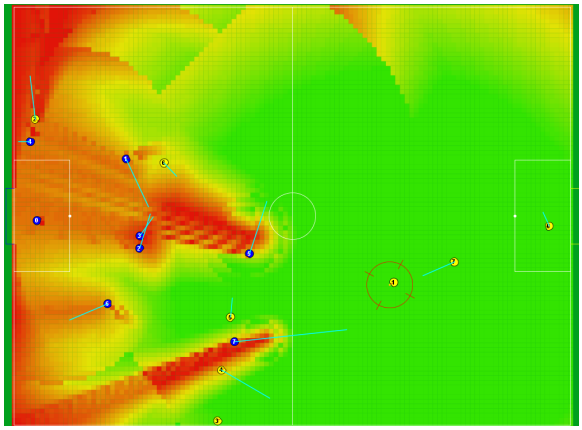
(a) Final pass score.



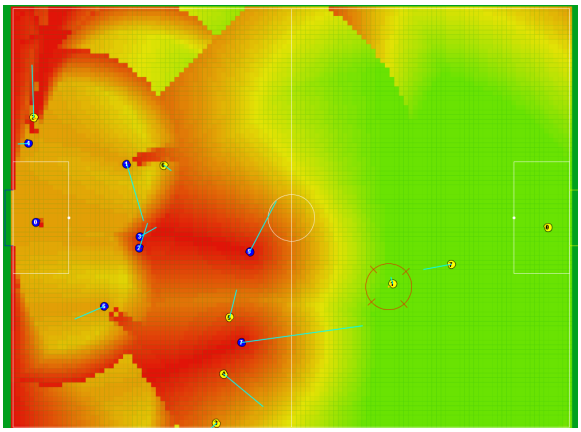
(b) Final pass score.



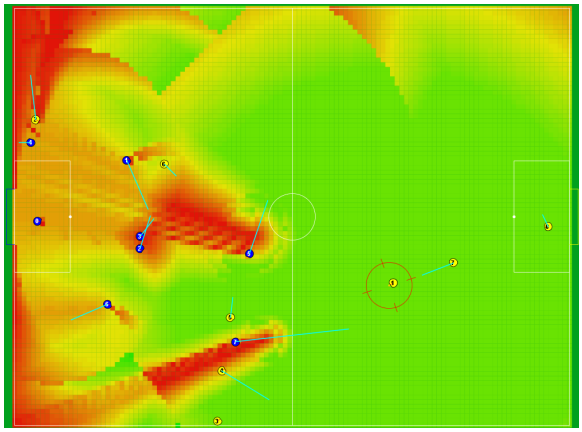
(c) Final score - constant weight.



(d) Final score - constant weight.

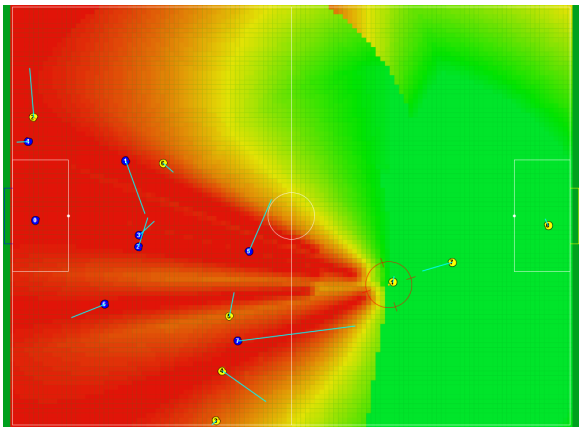


(e) Final score - situation weight.

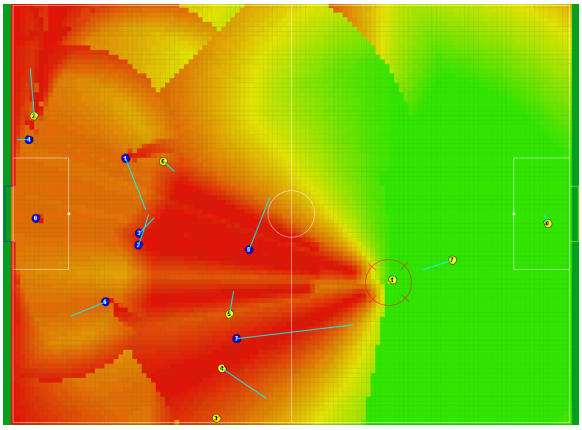


(f) Final score - situation weight.

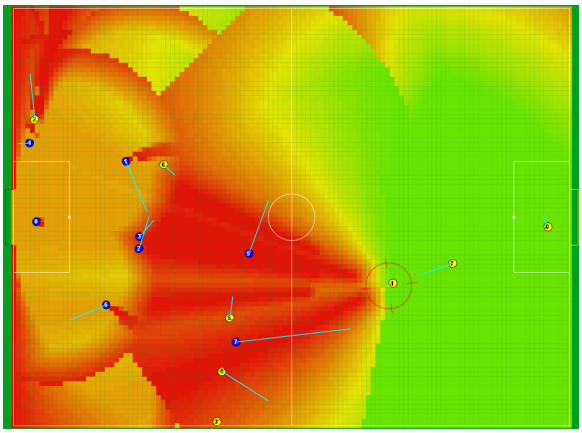
Figure 19. Heatmaps compared: Left) `DistancePassRater`; Right) `MovingRobotPassRater`.



(a) Final pass score.



(b) Final score - constant weight.



(c) Final score - situation weight.

Figure 20. `PassInterceptionRater`: Comparing the two weighting functions with the sole pass score.

A.2 Pass rater statistics

Table 1. Statistics for pass raters.

		passes				goalkicks	goals
function	rater	success	intercepted	disturbed	missed		
pass score	DPR	218	29	21	13	20	4
	MRPR	220	32	21	17	24	3
	PIR	217	15	15	14	26	4
constant w.	DPR	127	34	21	5	44	2
	MRPR	165	27	25	7	35	4
	PIR	201	44	21	12	25	3
situation w.	DPR	139	42	21	10	35	6
	MRPR	152	41	21	9	35	5
	PIR	114	50	14	9	32	7
probabilities	DPR	101	42	21	7	33	2
	MRPR	131	54	15	20	23	3
	PIR	115	51	33	3	29	5

Table 1 shows the counts of passes and goal kicks with their respective outcome for the different pass rater/ weighting function combinations. These counts are the results from analyzing 15 min of simulation recordings in Sumatra for each combination. The names of the pass raters are abbreviated: `DistancePassRater` (DPR), `MovingRobotPassRater` (MRPR), `PassInterceptionRater` (PIR).