# RoboCup 2022 SSL Champion TIGERs Mannheim - Ball-Centric Dynamic Pass-and-Score Patterns

Mark Geiger, Nicolai Ommer, Andre Ryll

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

**Abstract.** In 2022, TIGERs Mannheim won the RoboCup Small Size League competition with individual success in the division A tournament, the blackout technical challenge and the dribbling technical challenge. The paper starts with an outline of the robot's dribbling hardware and ball catching computations, followed by a high level summary of the AI used in the tournament. Given 62 scored goals and no conceded goals at RoboCup 2022, the focus is on describing the used attack and support behaviors and how they are selected. The paper concludes with a statistic of the tournament backing the efficiency of our employed strategies.

## 1 Robot Dribbling Hardware and Ball Interaction

As in all other RoboCup soccer leagues ball handing and control is a key factor to success. It has gained more importance recently as our offensive employs an increasing number of actions to steal the ball from opponents, to move with the ball, or to protect it from opponents (see Section 2). Section 1.1 gives an overview of the robot hardware which is in direct contact with the ball and recent updates applied to it. Section 1.2 describes how to approach the ball to actually make use of the hardware.

### 1.1 Dribbling Device

In the SSL a golf ball is used, which is the most rigid game ball of all leagues. Hence, the ball itself provides only very little damping during reception and dribbling. It also has a low friction coefficient, complicating ball control even further.

Consequently, damping and a high friction coefficient must be provided by the robots controlling the ball. This is done by a unit which is called the dribbling device. It is depicted in Fig. 1 for our v2022 robot generation.

We decided to use a design with two degrees of freedom, as we did in our v2016 robots [1]. We combined the v2016 2-DoF dribbler with ZJUNlict's additional

dampers [2]. The top damper is mainly used to absorb impact energy of incoming passes. As soon as the ball is actively controlled the exerted backspin on the ball can push the whole dribbler upwards on the sideward sliders. The additional load is absorbed by flexible elements and the motor is current and temperature controlled to prevent overstress. If the dribbler drops during the dribbling process (either due to a skirmish or an uneven ground) it is damped via the small bottom dampers. All dampers are 3D printed from a flexible TPE material with a 70A shore hardness. The damping properties of the top damper can be adjusted by changing its shape (mainly by varying the branch thickness).

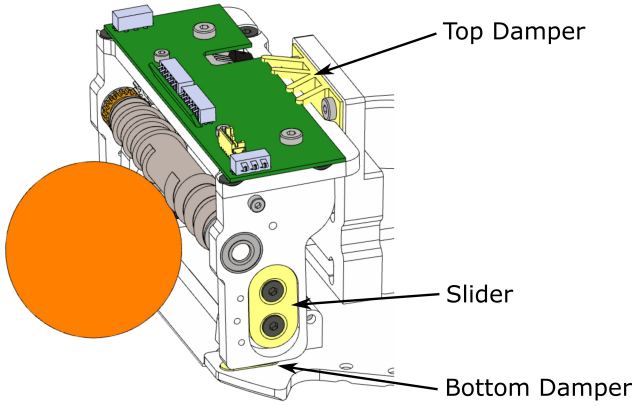



**Fig. 1:** Dribbling device with highlighted damping elements.

Compared to the version used in the 2021 hardware challenges some additional improvements were made [3]. The material of the dribbling bar has been changed to a soft silicone, which is much less abrasive than the previously used polyurethane. Due to the complex shape of the dribbling bar it is molded by using two 3D-printed half shells as a mold and pouring in the liquid silicone from the top. Furthermore, the gear modulus has been changed from 0.5 to 0.7 as the small gears tended to break under heavy load.

With the updated dribbling device, we achieve excellent damping properties and can stop an incoming pass directly at the robot. This was tested with another robot kicking the ball so that it reaches the dribbling device with 5 m/s. The rebound was assessed visually and no separation of the ball from the dribbling device could be identified. To retain ball control the dribbler can run at up to 25000 rpm. Depending on carpet friction it consumes between 2 A and 8 A of current. A higher current corresponds to a better grip of the ball. This current is also reported back to our central AI which uses it to asses if a difficult move with the ball can be executed. A detailed description of our v2020 hardware can be found in [4,5].

### 1.2 Catching a rolling ball

When a ball is rolling on the field, the robots have to stop or catch the ball. We use two different approaches to catch such a ball. If it is possible to intercept the ball by moving onto the ball travel line just in time, we try to intercept. Otherwise, we try to approach the ball from behind and stop it with the dribbler.

*Intercepting the ball* The approach of intercepting the ball is based on a method from CMDragons [6]. It samples multiple points along the ball travel line. A robot trajectory is then planned to each point and the resulting travel time is associated with the respective point. Then, we calculate the time that the ball needs to reach each point by using our internal ball model [7]. This gives us the slack time that a given point has. A negative slack time means that the robot reaches the position before the ball. Plotting the slack times results in the graph shown in Fig. 2. In most situations in which a ball is rolling towards the robot, there are two time slots (interception corridors) where the robot can actually catch the ball. Usually one small time window to catch the ball close to the robots current position and one large time window far in future, when the ball is getting so slow that the robot can overtake the ball again.

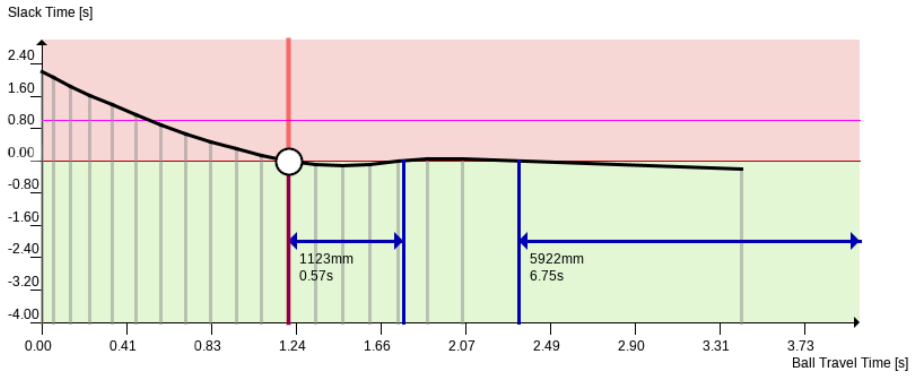



**Fig. 2:** Ball interception calculation.

The robot will try to move towards the first reachable interception corridor (negative slack time) that meets some requirements (corridor width $> 0.2\,\mathrm{s}$ and min slack-time $<$ -0.2 s). The selected corridor begins at a given ball travel time (x-axis), which we can use to feed the ball model to calculate a target position where to actually catch the ball. This will be done for each robot on the field. The robot that can catch the ball most rapidly will be selected as the primary offensive robot. If it is uncertain that the primary robot is able to catch the ball, then multiple robots may try to intercept the ball.

*Approach and stop ball* The fallback, when intercepting the ball is not feasible is to approach the ball from behind by moving onto the ball travel line and then approaching the ball until it hits the spinning dribbler. As soon as the ball is on the dribbler, the robot brakes as quickly as possible without loosing the ball. If tuned well, this is quite an effective approach to quickly gain back ball control.

## 2 Offensive Strategies

This section introduces the basic foundation of the offensive decision making. One key aspect of the offensive strategy are the *OffensiveActionMoves*. An *OffensiveActionMove* represents a specific action a robot can execute. An *OffensiveActionMove* can be a simple pass, a kick on the opponents goal, or a special behavior in close engagements with robots from the opponent team. Currently, we have ten *OffensiveActionMoves*. There are three methods that each *OffensiveActionMove* has to implement. The method *isActionViable* determines the viability of an *ActionMove*. The viability can either be *TRUE*, *PARTIALLY* or *FALSE*. The method *activateAction* controls the actual execution of the move. The method *calcViabilityScore* will determine a score between 0 and 1 for the current situation. This score should be connected to the likelihood, that this action can be executed successfully. The viability and its score are calculated in a unique way for each OffensiveActionMove. For example, the viability of a GOAL_SHOT is determined mainly by the open angle through which the ball can enter the opponent's goal. The viability of a PASS is mainly determined by the pass target rating (see section 3.2). The different scores are made comparable by additional weights set by hand, based on an educated guess. In addition, a self-learning algorithm is used that takes into account the successes and failures of past strategies to fine-tune these weights during a match. This algorithm was first presented in our 2018 TDP[1].

Algorithm 1 shows how the best *OffensiveActionMove* out of one given *OffensiveActionMoveSet* is determined. It is important to note that the *OffensiveActionsMoves* inside a given set have a specific ordering, which represents the priority. The *OffensiveActionMove* in the first position of the set has the highest priority. An *OffensiveActionMove* will be activated if its viability returns *TRUE* and it has a higher priority than all other *OffensiveActionMoves* that return a *TRUE* viability. Actions that return the viability *FALSE* will be ignored in any further processing. All actions that are *PARTIALLY* viable are sorted by their *viabilityScore* and if there is no action that has a *TRUE* viability, then the action with the highest *viabilityScore* will be activated. In case all actions have a *FALSE* viability then a default strategy will be executed.

**Algorithm 1** Pseudocode - Find the best OffensiveActionMove

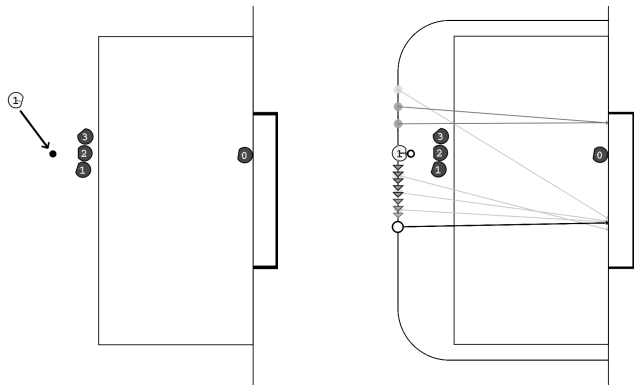
```
for (var action : actionsSet) {
  var viability = action.isActionViable();
  if (viability == TRUE) {
    // activate first move that got declared as viable
    return action.activateAction();
  } else if (viability == PARTIALLY)
    partiallyMoves.add(action);
}

partiallyMoves.sort(); // sort by viabilityScore
if (!partiallyMoves.isEmpty()) {
  // choose best partially viable move to be activated
  return partiallyMoves[0].activateAction();
}
return defaultMove.activateAction()
```

The separation into viable and partially viable actions, combined with priorities leads to a very stable and easily modifiable/extendable algorithm for the offensive strategy. For example, the *OffensiveActionMove* that controls direct kicks on the opponent goal will return a *TRUE* viability if there is a high chance to score a goal. If there is a extremely low chance to score a goal it will return *FALSE*. Otherwise, if the hit chance is reasonable but not really high, it will return *PARTIALLY*. Additionally, this action has a high priority. Thus, the robot will surely shoot on the goal if there is a good opportunity to score a goal. However, if the viability is *PARTIALLY* the action will be compared with the other actions and based on the *viabilityScores* the robot will decide whether it should shoot on the goal or execute another action, e.g. a pass to another robot.

### 2.1 Offensive Dribbling

Another offensive action that the robot may choose is the so called *DribbleKick*, which is one of the dribbling actions the robot can do. Figure 3a shows a typical scenario of a ball located in front of the opponent goal. In this case the robot chooses to do a DribbleKick. The robot approaches the ball and tries to bring it onto its dribbler. The strength of the dribble contact can be estimated from the power drawn by the dribble motor (see Section 1.1). The robot will wait until the ball has a strong contact and also checks if it is possible to score a goal from another position on a curve around the opponents penalty area. Multiple points on the curve are sampled and evaluated for their chance to score a goal (white = high chance to score, gray = low chance to score). The robot will drive along the curve towards the best point, while keeping the ball on the dribbler. As soon as the target is not blocked anymore the robot will kick the ball as shown in Fig. 3b.



**(a)** Attacker approaching ball **(b)** Evaluation of scoring chances

**Fig. 3:** Execution of a DribbleKick

The entire sample-curve can move closer or further away from the opponent goal, depending on the behavior of the defending robots. In general the robot will try to avoid coming to close to opponent robots. The robot must also adhere to the maximum dribble distances allowed. Therefore, it does not sample positions farther away than the maximum allowable dribble distance (1 m) to avoid dribble rule violations. The robot tries to move laterally and shoot the ball while it is still in the acceleration phase. Since the opposing robot only reacts to the measured position of our robot, it will always have a disadvantage due to overall system latency. As the robot tries to shoot during acceleration it may not be possible to change movement anymore if a dribbling violation is imminent. In such a case, the robot will simply shoot the ball to avoid a violation, even if there is no good chance to score a goal.

The calculations are done on every AI frame. Meaning that there is no *plan* that the robot follows. Each frame the destination or the kick target can change. This is important, because we need to react fast to the opponents movement and re-evaluate our strategy constantly. In order not to lose the ball while dribbling, the robot balances its orientation so that the rotational force of the ball points in the direction of our robot. When a dribbling robot changes its orientation, the force vector of the rotating ball also changes. However, it lags behind the robots movement. If the orientation or the direction of movement is changed too quickly, ball control may be lost. The robot will give priority to ball control during the movement. However, if the robot sees that it could score a goal, it will quickly align itself towards the target and shoot. For the final shot, the robot will take into account its current velocity to calculate the final alignment towards the target to make an accurate shot.

### 2.2 Defensive Dribbling

Our AI distinguishes between defensive and offensive dribbling. Defensive dribbling is concerned with getting the ball and protecting it from the opponent robots while always adhering to the dribbling rule constraints. Our attacking robot will remain in the defensive dribbling state until a good enough offensive strategy has been found.

Figure 4 shows a common situation. The ball is located in front of the translucent robot and an opponent robot is about to attack us. Our robot has ball control, but no offensive action with a good enough viability score. Thus, the robot will enter the defensive dribbling mode. The robot will then try to protect the ball from the opponent robots. Multiple points within the allowed dribbling radius are sampled and evaluated. The robot will then dribble the ball towards the position that is rated to be the safest from opponent robots. At the same time the robot will try to turn the ball away from the opponent robots.

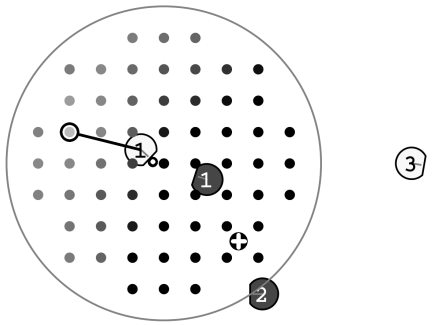



**Fig. 4:** Defensive dribbling calculations.

## 3 Support Strategies

Robots which are not assigned to any attacking or defending role become supporting robots. They are supposed to run free, look for good positions on the field from where they can safely receive a pass and ideally also have a good chance to score a goal. Section 3.1 gives an overview of the high-level behaviors a supporting robot may get assigned. They define where a robot should go. Section 3.2 outlines where our robots may receive passes and forms the connection between support and attack strategies.

### 3.1 Supporting Robots

Given the fast-paced nature of the Small Size League, planning too far in the future is not advisable. Situations change within fractions of a second. So instead of finding good positions globally on the field, we focus on optimizing current robot positions first, while still observing the global supporter distribution. With the increasing number of robots in the league (2012-2017: 6, 2018-2019: 8, 2021-2022: 11) more and more robots take over the supporting roles. During a free kick in the opponent half, we may use up to 9 supporters, while 5 years ago, it were 4 at most.

Over the previous years, we developed different supporting behaviors. Each supporter is assigned a behavior. There can be limits on the number of robots having a certain behavior and behaviors may be disabled based on situation, tactics or game state. For each robot, the viability and a score between 0 and 1 of each behavior is determined and the best rated behavior is assigned. The viability algorithm is similar to the one described in Section 2. The following sections describe some of the most important behaviors.

*Direct Goal Redirector* Find a position from where a goal can be scored, optimizing for the redirect angle, namely the angle between the current ball position, the desired supporter position and the goal center. A small redirect angle is better, because receiving it is more reliable and precise.

*Fake Pass Receiver* If a supporter is near an ongoing or planned pass, it pretends to receive this pass by standing close to the passing line, but without actually receiving the ball. Opponents will need to figure out which is the right receiver or need to defend all potential receivers. This behavior could often be observed quite clearly in matches[1].

*Penalty Area Attacker* Position the robot as close as possible to the opponent penalty area to prepare it for a goal kick. Passing through the penalty area and scoring from that position will leave the defense few chances to block the goal kick.

*Repulsive Attacker* Bring the supporter to a good attacking position without interfering with other supporters using a force field with several force emitters. For example field boundaries, other robots and a general trend towards the opponent goal. The desired position is determined by following the forces in the field a fixed number of iterations. Figure 5 shows such a force field fully visualized for the team playing towards the left goal. In the own half, forces are directly towards the opponent half, while in the opponent half, forces are directly towards the middle of the left or right side of the opponent half and away from opponents and the ball.

[1] https://youtu.be/W8Z_2a2Ieak?t=80

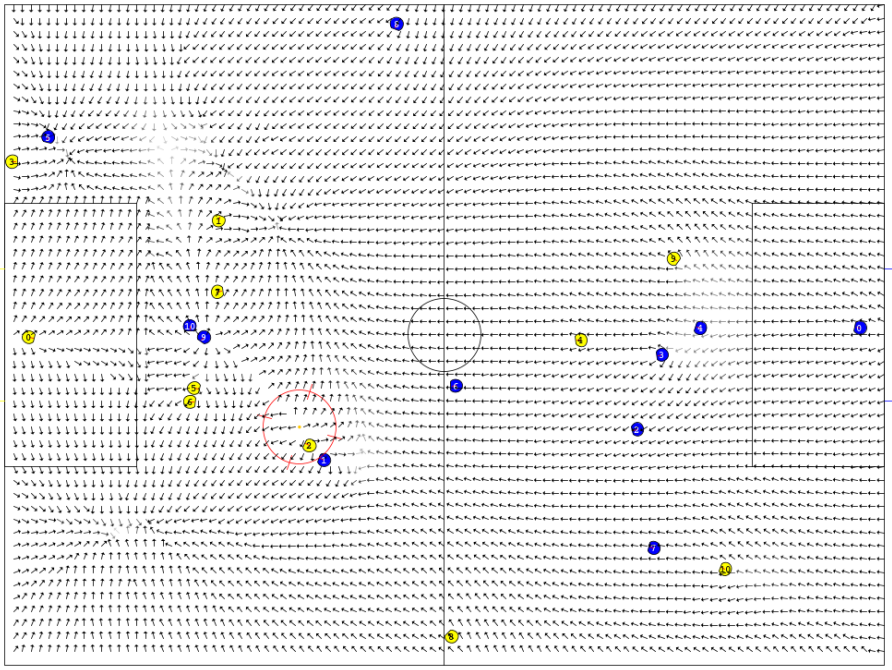


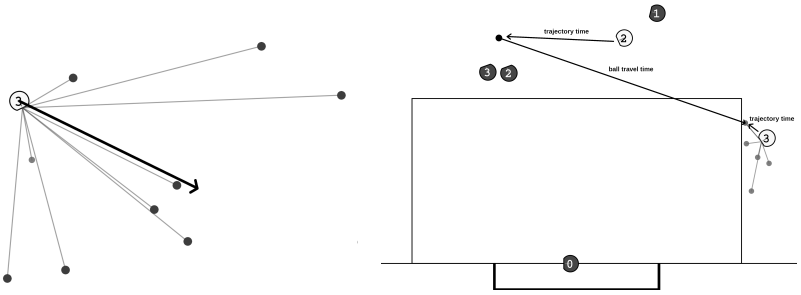
**Fig. 5:** Repulsive force field for Repulsive Attacker behavior

*Repulsive Pass Receiver* Based on the same repulsive principal as the Repulsive Attacker behavior, a position with a certain distance to the ball that is not covered by opponents is targeted.

### 3.2 Pass Targets

Pass targets are potential positions where the ball can be received and are calculated for each friendly robot on the field, except for the keeper. Multiple pass targets exist for each robot. Each one having a rating which is based on its passability, chance to score a goal from the pass targets location and the chance of an opponent intercepting the ball on its way to the pass target. The pass targets are calculated in a circle around the robot, where the circle is additionally shifted in the current motion path of the robot. The radius of the generation cirlce is determined by the current velocity of the robot. A fixed number of unfiltered targets is generated in each frame. The list of targets is then filtered to ensure that all targets can be reached in time by the designated robot until the scheduled pass arrives. Different times are taken into account: The time until the attacking robot is able to shoot the ball, the travel time of the ball and the time needed for the pass receiving robot to reach its passing target. The best pass targets

from the past are reused to efficiently optimize the targets over time. Figure 6a shows the calculated pass targets for a single robot.




**(a)** Pass targets generated for a single robot.

**(b)** Pass targets generated for a single robot.

**Fig. 6:** Illustration of pass target generation and selection.

Once an offensive robot gets close to the ball, it will choose the currently calculated offensive action. In case of a pass, the offensive strategy will take over the robot with the best rated pass target as the pass receiver shortly before the pass is executed. This allows a tighter and more stable coordination between the robots. Figure 6b shows a typical pass situation generated by our AI. Robot 2 plans to pass the ball to robot 3. Rather than passing to the robot's position, it passes to a pass target near robot 3. Since we use trajectories to control our robot movement, we can calculate the time at which robot 2 needs to kick the ball. Furthermore, we can use the initial ball velocity and our ball model [7] to calculate the time at which the ball needs to reach the pass target. Also, we can calculate the time robot 3 needs to reach his pass target. By combining these numbers we can synchronize the time in which the ball and robot 3 will reach the pass target.

## 4 Conclusion

At RoboCup 2022, TIGERs Mannheim played 10 official matches during the group and elimination phase of the division A tournament and scored 62 goals in total, while not conceding any goal throughout the tournament. Every second attempted goal shot was successful and two thirds of passes between TIGERs robots succeeded on average. These numbers support the focus and strength of the team: A fast paced and dynamic attack strategy.

The numbers were extracted from the official log files [2] using the TIGERs log analyzer from the technical challenge 2019 [3] and the leagues match statistics [4]. Table 1 shows the full set of gathered statistics. Only shots with a duration of at least 300 ms were considered. The ball possession specifies the amount of time that the team uniquely possessed the ball relative to the time that either team uniquely possessed the ball. So a value larger than 50% means that this team possessed the ball more often than the other team.

**Table 1:** Tournament statistics from RoboCup 2022 (Division A)

| Team | Goals scored | Goal shots | Goal shot success | Ball possession | Passes | Pass success |
|---|---|---|---|---|---|---|
| TIGERs Mannheim | 62 | 113 | 54.9% | 63.2% | 648 | 66.6% |
| ER-Force | 13 | 95 | 13.7% | 59.3% | 619 | 53.3% |
| RoboTeam Twente | 3 | 28 | 10.7% | 37.6% | 255 | 34.9% |
| KIKS | 1 | 29 | 3.4% | 52.1% | 268 | 34.3% |
| RoboDragons | 1 | 127 | 0.8% | 35.5% | 270 | 29.3% |

The number of goals scored indicates a good positioning by the supporters and also a reliable and precise execution of kicks by the actual robots on the goal. The outstanding goal shot success ratio underlines very well the offensive action selection based on viabilities. Our robots do not blindly force kick towards the opponent goal on every opportunity but carefully decide if this action would have a chance to score at all. Alternative actions like the defensive dribbling ensure a high ball possession rate in case no other reasonable offensive strategy is available. The high number of passes and the pass success ratio show that our supporters are in good positions to receive passes and the offense often selects a passing action to get in a good position to score.

## 5 Publication

Our team publishes all their resources, including software, electronics/schematics and mechanical drawings, after each RoboCup. They can be found on our website[5]. The website also contains several publications with reference to the RoboCup, though some are only available in German.

[2] https://ssl.robocup.org/game-logs/
[3] https://ssl.robocup.org/robocup-2019-technical-challenges/
[4] https://ssl.robocup.org/match-statistics/
[5] Open source / hardware: https://tigers-mannheim.de/publications

## References


1. A. Ryll, M. Geiger, C. Carstensen, and N. Ommer. TIGERs Mannheim - Extended Team Description for RoboCup 2018, 2018.
2. Z. Huang, L. Chen, J. Li, Y. Wang, Z. Chen, L. Wen, J. Gu, P. Hu, and R. Xiong. ZJUNlict Extended Team Description Paper forRoboCup 2019, 2019.
3. N. Ommer, A. Ryll, and M. Geiger. TIGERs Mannheim - Extended Team Description for RoboCup 2022, 2022.
4. A. Ryll and S. Jut. TIGERs Mannheim - Extended Team Description for RoboCup 2020, 2020.
5. A. Ryll, N. Ommer, and M. Geiger. RoboCup 2021 SSL ChampionTIGERs Mannheim - A Decade of Open-SourceRobot Evolution. In R. Alami, J. Biswas, M. Cakmak, and O. Obst, editors, *RoboCup 2021: Robot World Cup XXIV*, 2022.
6. J. Biswas, J. P. Mendoza, D. Zhu, and M. Klee, S. Veloso. CMDragons 2014 Extended Team Description, 2014.
7. A. Ryll et al. TIGERs Mannheim - Extended Team Description for RoboCup 2015, 2015.