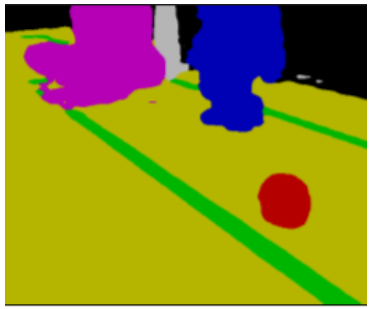
决策的重要性及地位

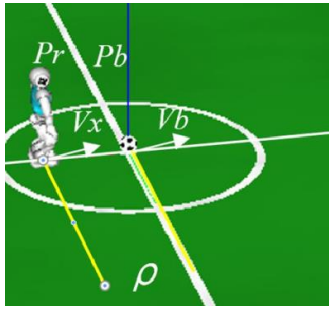
Standard Platform League

机器人系统实现 & 制胜关键

关键技术

❑ 感知　❑ 行为　❑ **决策**



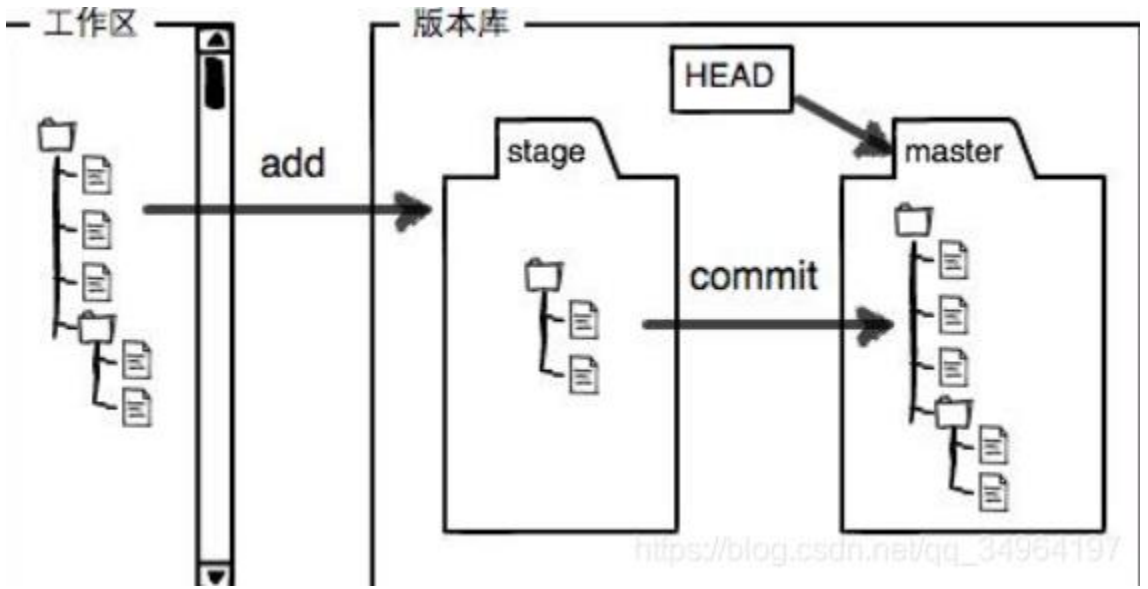


- 围绕球的进攻及防守—与决策好坏直接相关
- “球商”的重要体现

Git使用

- **工作区**（Working Directory）：指的是在电脑里能看到的目录，比如mymenu文件夹就是一个工作区
- **版本库**（Repository）：.git目录，Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。

  - git add是把需要提交的文件添加到暂存区
  - git commit是把暂存区的所有内容提交到当前分支

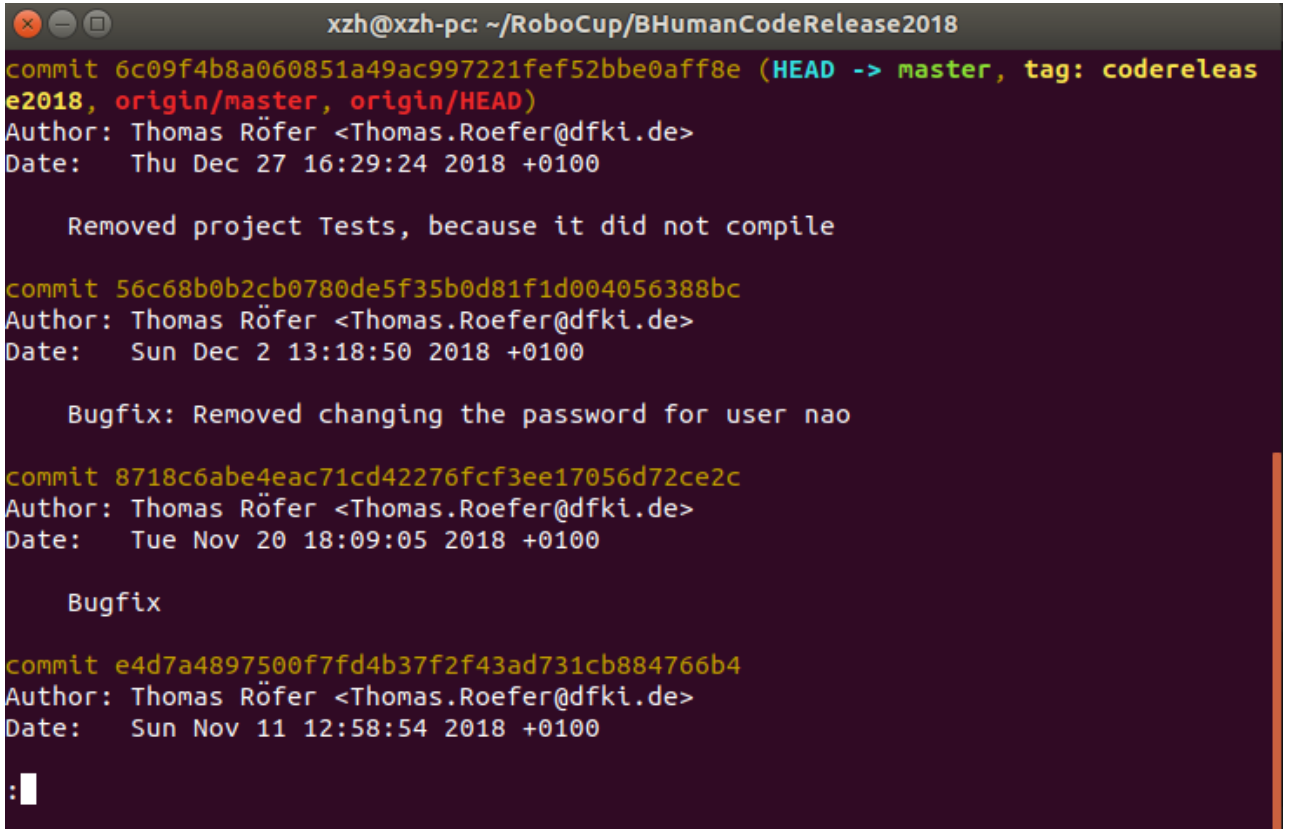


Git使用

- 下载bhuman的代码

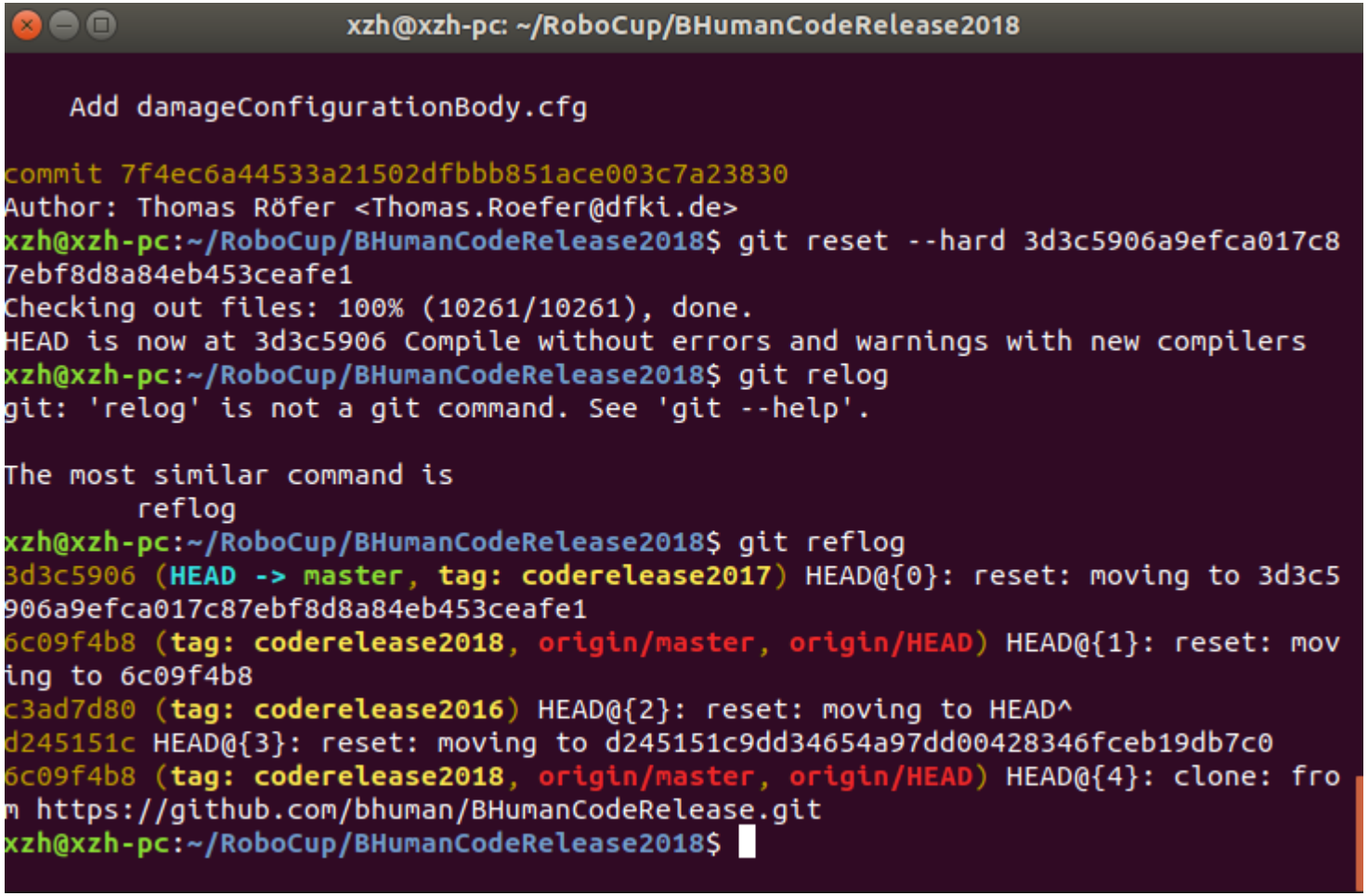
```
git clone https://github.com/bhuman/BHumanCodeRelease
```

- 使用git log 查看commit的信息



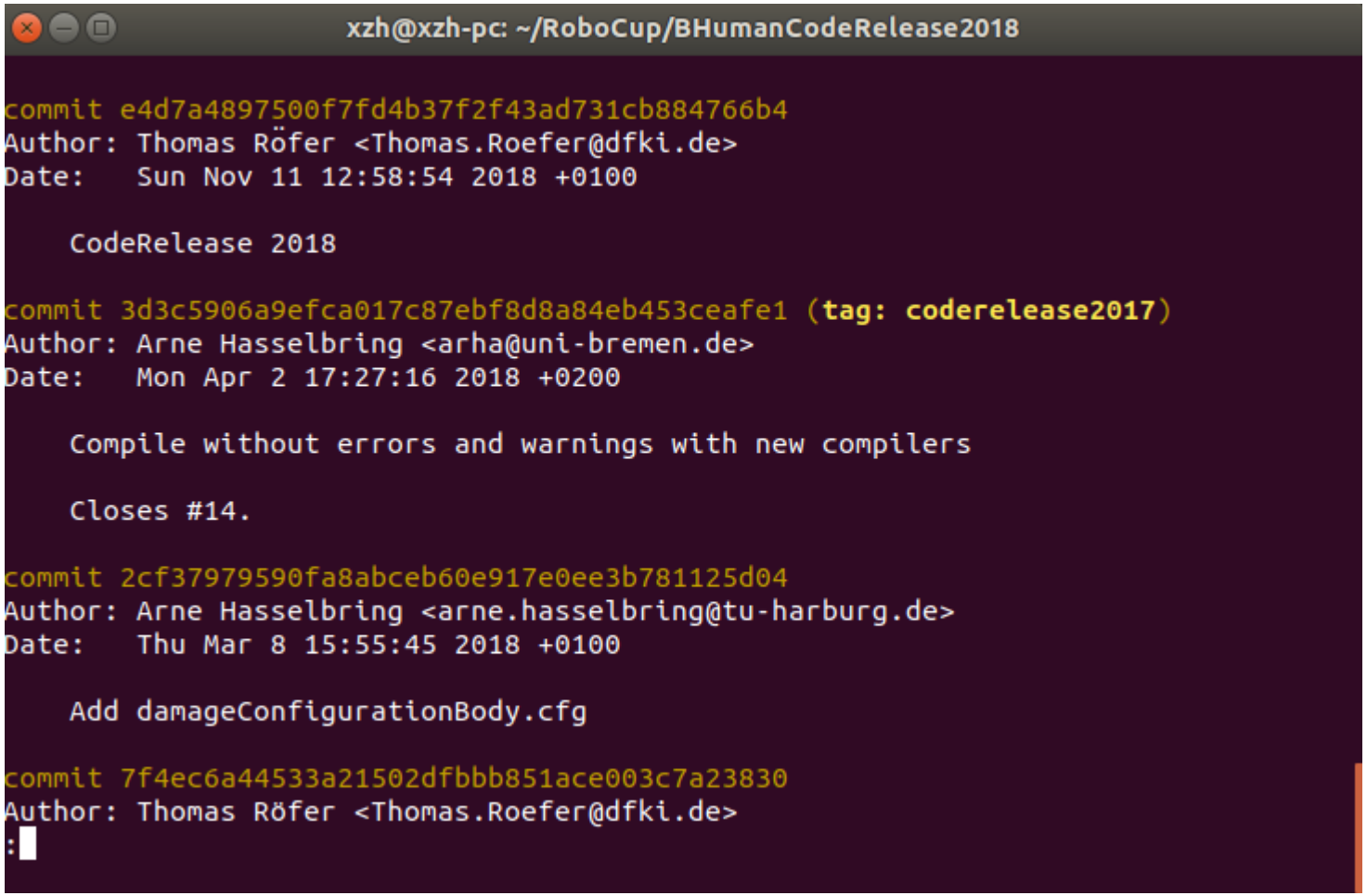
Git使用

`git reflog` 可以查看所有分支的所有操作记录(包括已经被删除的 commit 记录和 reset 的操作)



Git使用


```
git reset --hard 3d3c5906
```



Git使用


```
git reset --hard 3d3c5906
```

	git reset产生影响			表现	
选项	HEAD	索引（暂存区）	工作目录	原有文件内容的变更	目录结构的变更（增加或者删除文件）
--soft	是	否	否	修改内容还在，变成未add的状态	新增文件：还存在，变成未add的状态(目录结构中文件变成绿色，可以再次执行git commit)； 删除文件：目录结构中还是没有，可以直接执行git commit
--mixed	是	是	否	修改内容还在，变成未add的状态	新增文件： 还存在，变成未add的状态(目录结构中文件变成红色，需要执行命令git add . 再执行git commit) 删除文件：目录结构中还是没有，可以直接执行git commit
--hard	是	是	是	修改内容丢失，修改的代码不会变成未add的状态	新增文件丢失、删除的文件相当于没删

比赛规则

RoboCup Standard Platform League (NAO) Rule Book

RoboCup Technical Committee

进攻

- ❑ Kick-off Shot
- ❑ Player Pushing
- ❑ Leaving the Field

防守

- ❑ Illegal Defender

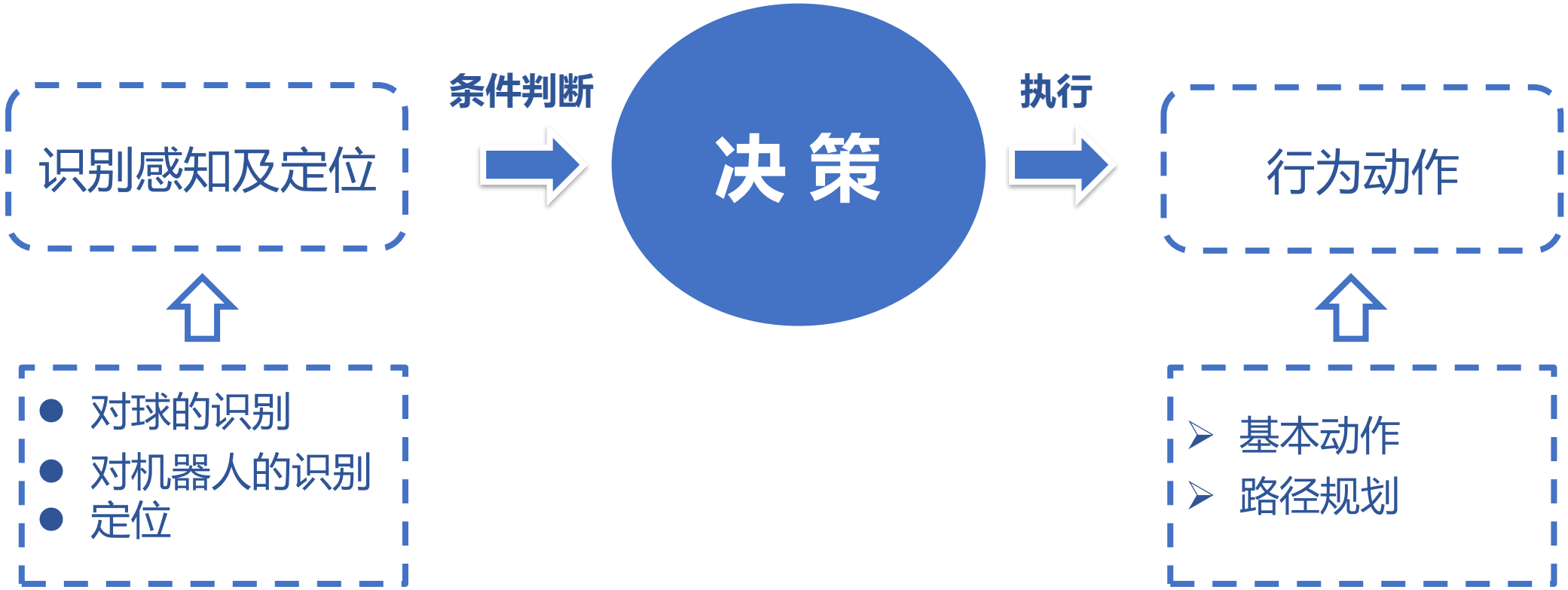
争取更多进球的同时减少犯规
标准移除处罚（Standard Removal Penalty）处罚时间会逐渐增加

比赛流程及环节

- Initial Kick-off
- Kick-off
- Kick-in
- Free Kick
- Penalty Kick
- 挑战赛

——全面考虑各环节并进行针对性策略编写

决策



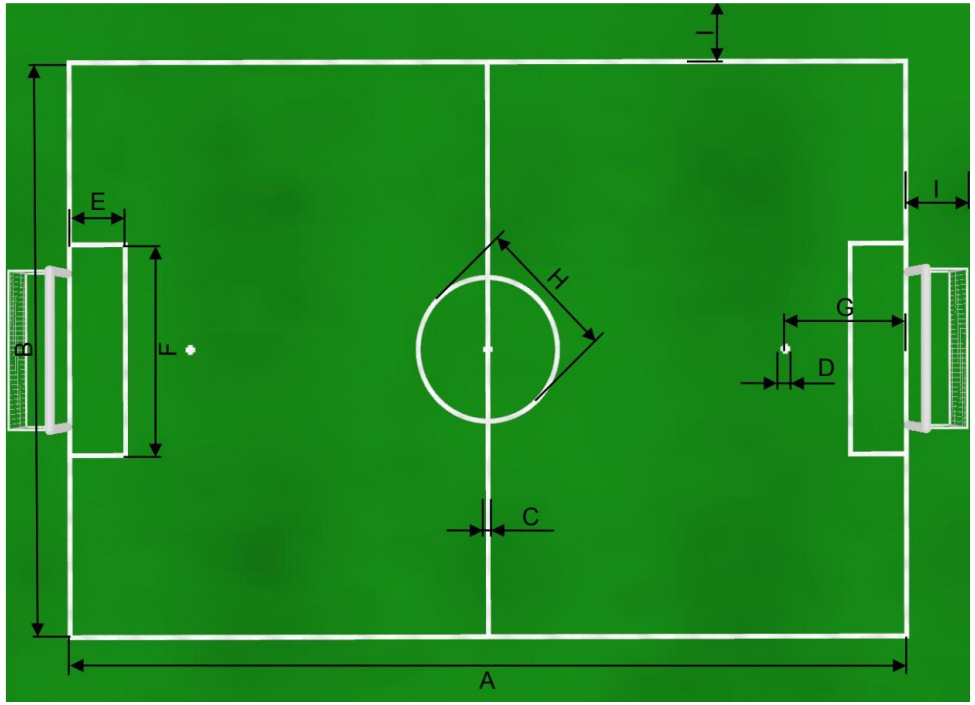
决策关注信息——输入

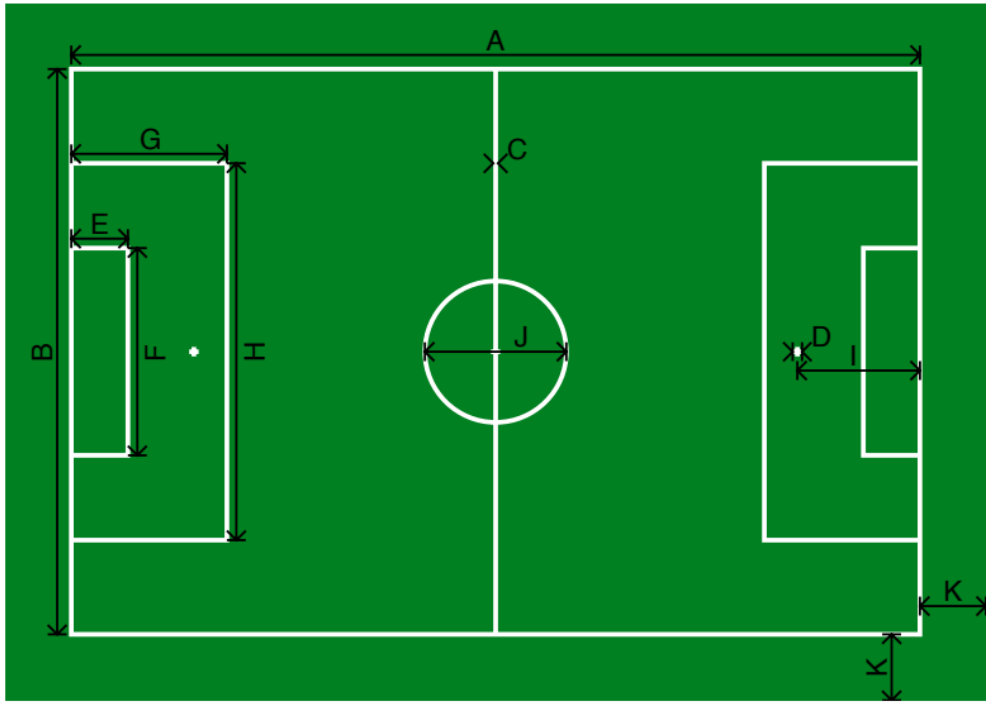
主要依靠感知及定位

- 机器人自身场上位置
- 机器人与球及球门的位置关系
- 机器人与队友的位置关系
- 机器人与对手的位置关系

决策关注信息——输入

机器人的位置信息结合场地信息：





- 判断半场位置
- 球门位置
- 边界位置

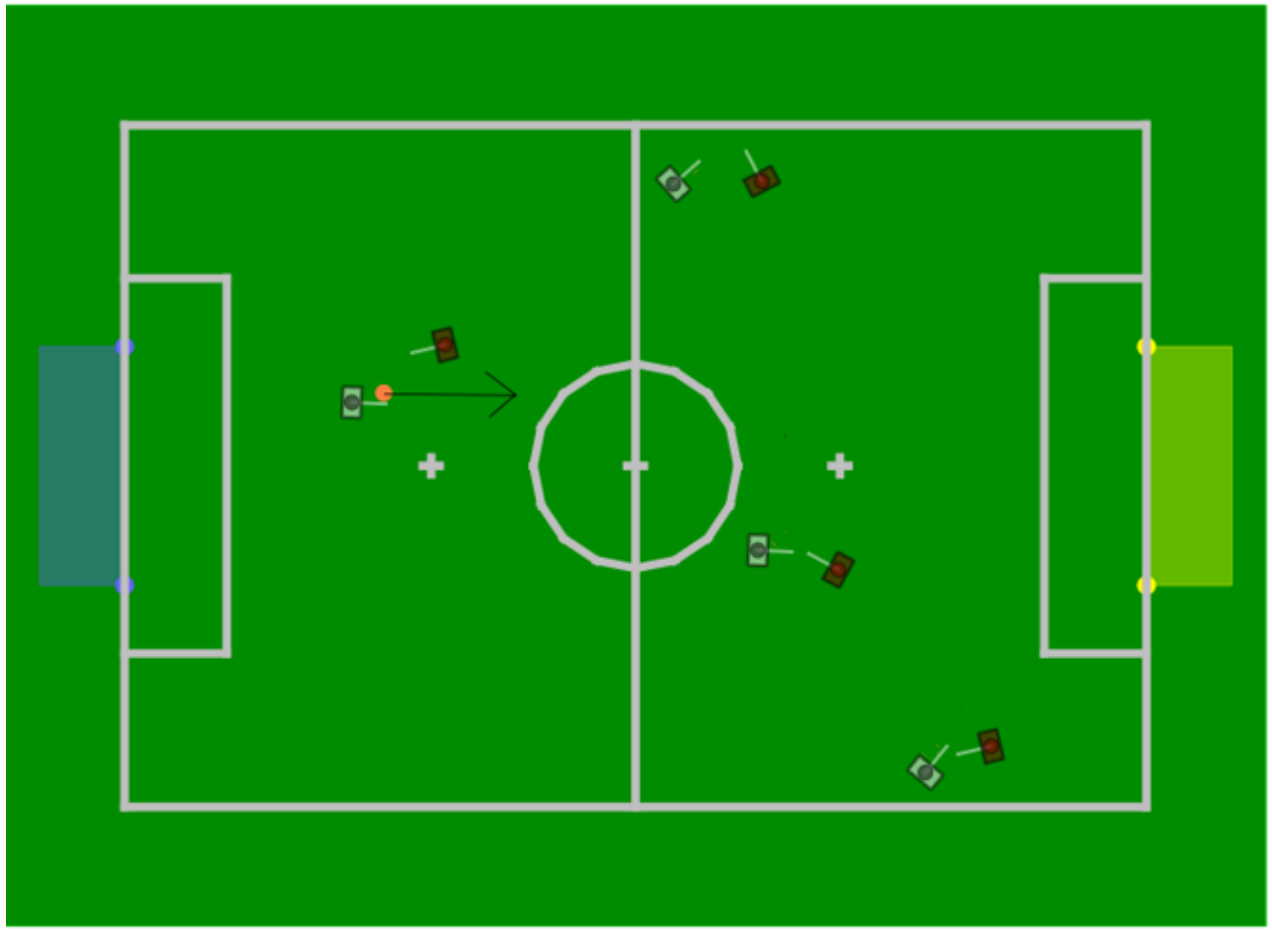
决策关注信息——输入

球的位置信息结合场地信息+机器人位置信息：

- 机器人是否控球判断
- 踢球参数判断
- 目标路径规划

决策关注信息——输入

机器人位置信息(己方+对手方)



- 传球判断
- 过人判断

决策输出——行为

- 动作
  - 走（速度，目标位置）
  - 踢球（力度，方向，目标位置）
  - 带球（方向，目标位置）
  - 找球（转头，移动旋转）
  - 扑救（动作选择）

目的：

- 围绕球的进攻及防守
- 考虑场上站位——提供更好的视角并获取信息

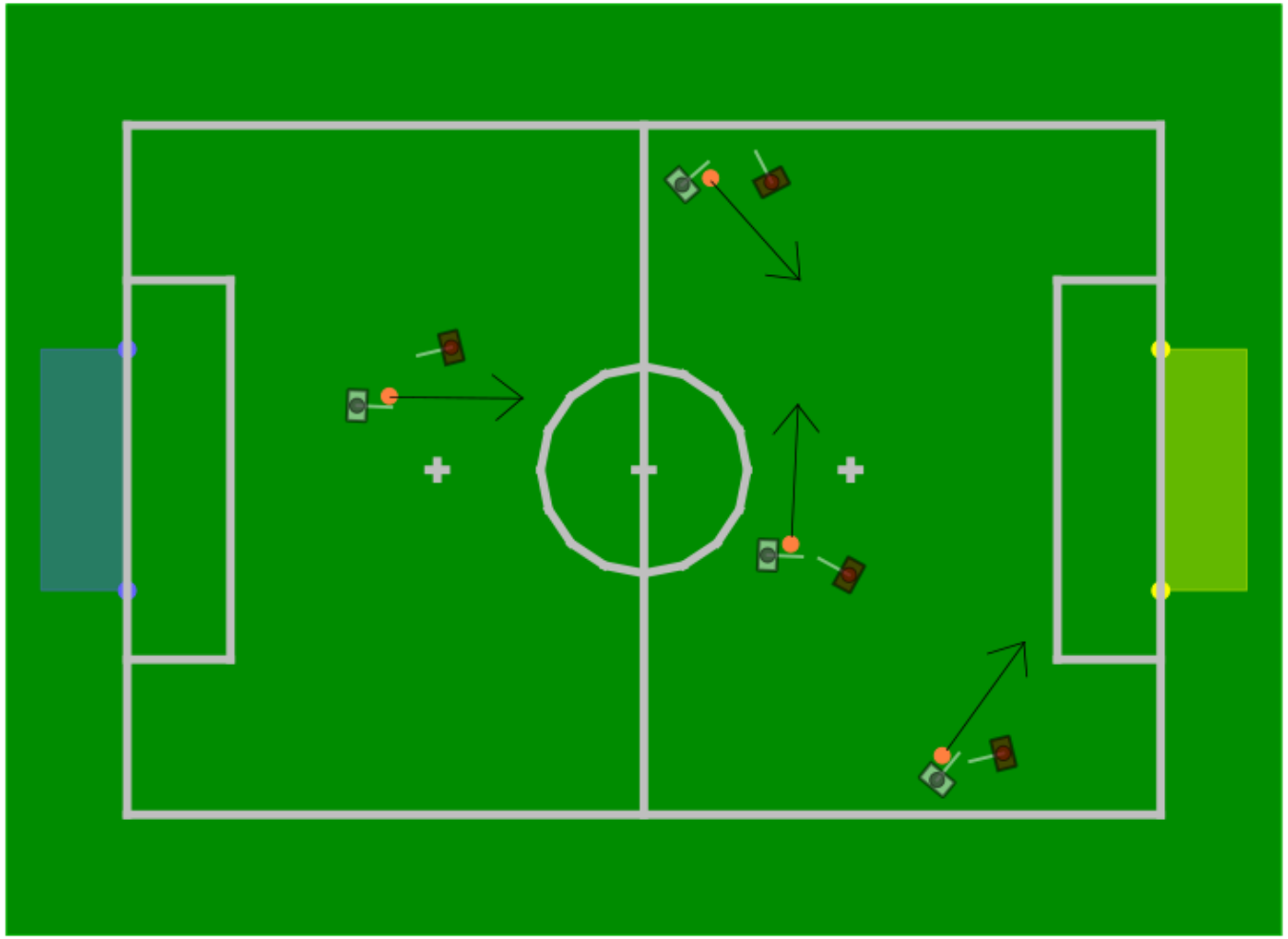
决策输出——行为

走（速度，目标位置）

- 行走算法的选择
- 相对位置 & 绝对位置
- 世界坐标系 & 机器人坐标系
- 不同的路径规划算法
- 是否有避障

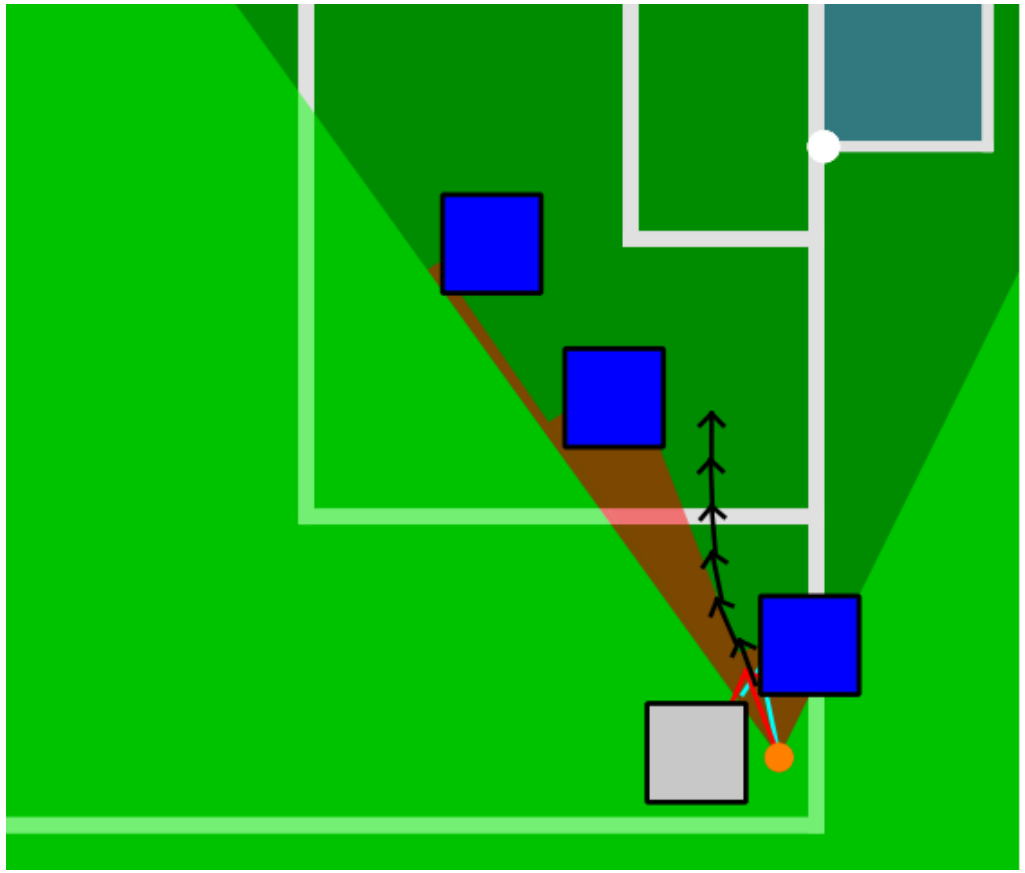
决策输出——行为

踢球（力度，方向，目标位置）



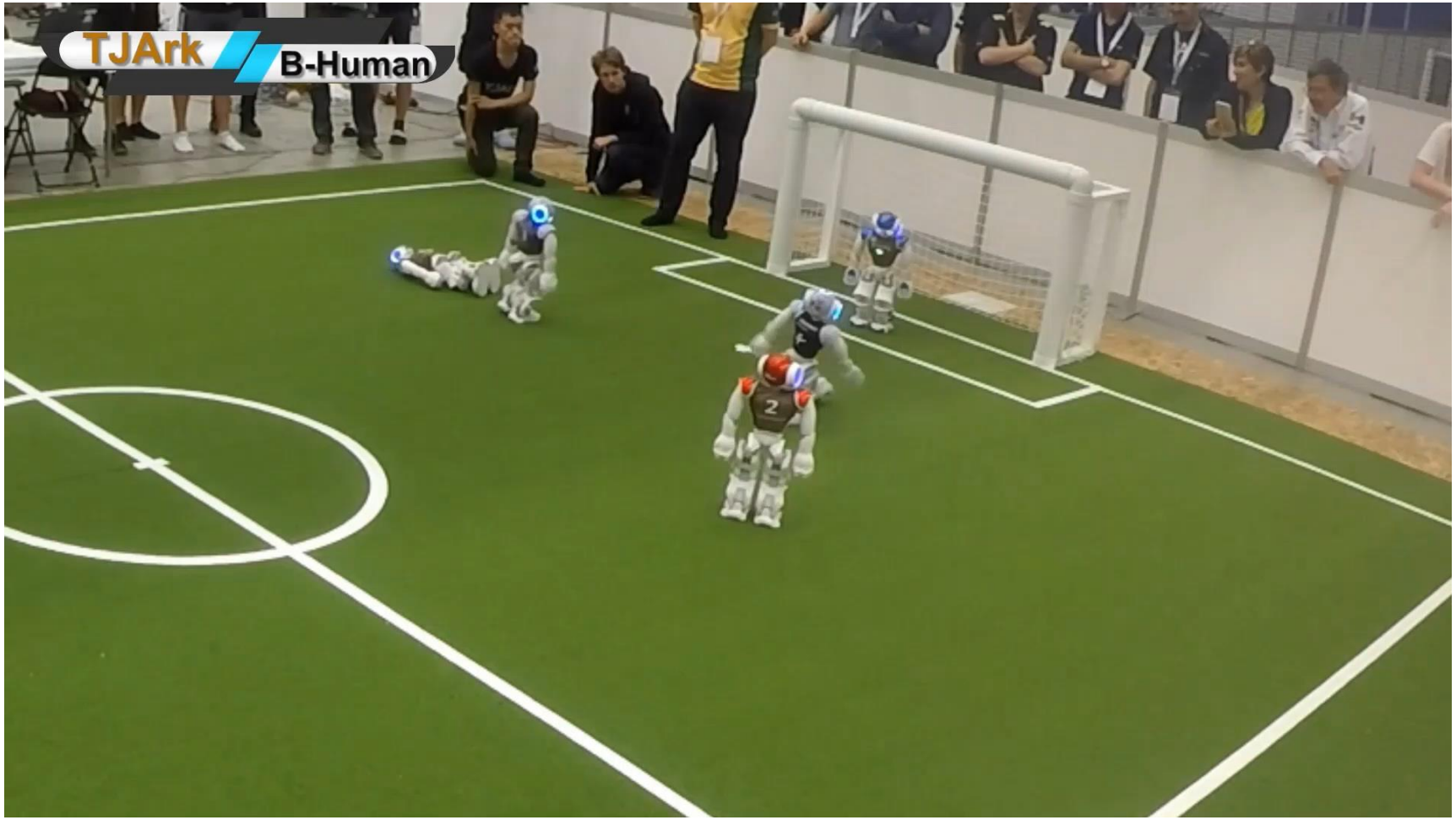
决策输出——行为

带球（方向，目标位置）DribbleToGoal



决策输出——行为

扑救（动作选择）



进攻决策

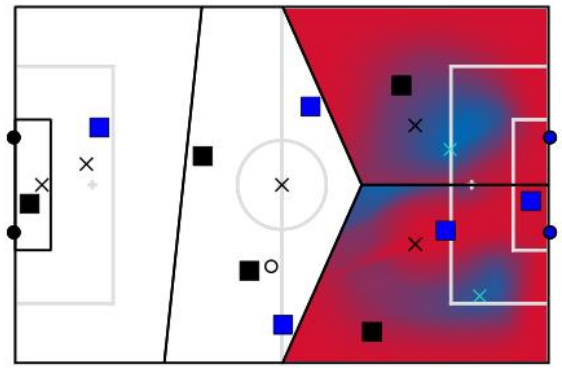
- **Behavior讲解** B-Human https://docs.b-human.de/coderelease2024/

- B-Human为行为控制Behavior Control开发了一个**新的体系结构**，将在B-Human Wiki中深入描述。本章重点介绍2022年旧版本代码用于提高机器人skills技能的方法。

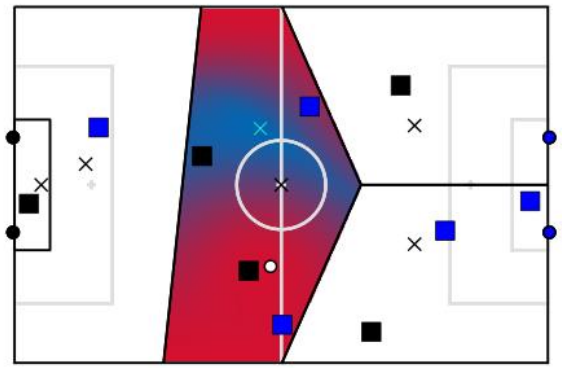
- **角色Roles--Forward前锋** 文件路径：/Src/Modules/BehaviorControl/StrategyBehaviorControl/PositionRoles/Forward.cpp

- 前锋球员的职责主要是为球队**射门**。当队友正在踢球时，前锋必须接到传球才能射门。为了确定实现该任务的最佳位置，搜索空间被限制在角色当前分配给的区域，即由团队战术产生的相应**Voronoi单元（维诺图）**。策略通过为每个角色指定一个**基本位置**来定义团队的一般结构，该位置可以用作生成Voronoi图的种子seed。
- 我们已经有一个传球和射门的**评级函数rating function**，基于此构造了一个新的评级函数，将它们与基本姿势Pose周围的正态分布结合起来，**以鼓励靠近该位置**。它们通过乘法组合并与Voronoi区域进行剪切。机器人的理想位置应在**评级函数的最大值处，如下图（a）所示。**
- 搜索使用梯度上升，可能会陷入局部极大值。然而，这个过程在计算上很轻量，特别是当我们在第一帧开始搜索直到最后一帧才找到最佳位置的时候。

2个Forward前锋



(a) Rating of positions for both forward players.



(b) Rating of positions for the midfielder.

1个Midfielder中场

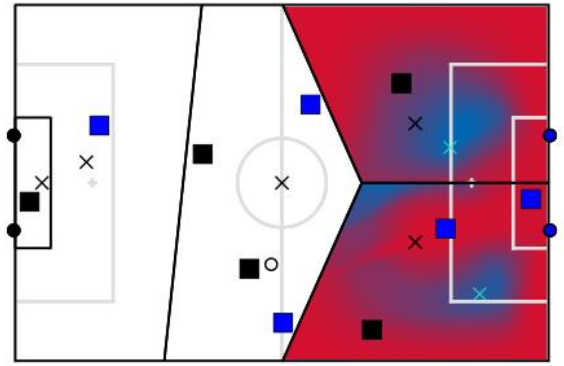
➢ 位置角色在对应Voronoi区域内的归一化**评级函数**(黑色分割线)。十字标记的基础位姿(黑色)和发现极大值处(绿色)。球位于白色圆圈。方块表示队友(黑色)和对手(蓝色)。门柱用各自球队颜色的圆圈表示（黑和蓝）。

进攻决策

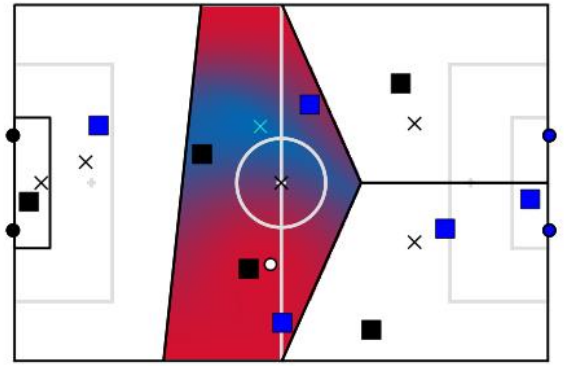
- 角色Roles--Midfielder中场

- 中场球员**既要进攻又要防守**，所以其行为的描述比前锋球员复杂一些。中场球员总是**试图远离队友**，以确保**较高的场地覆盖范围**，并成为一个潜在的**传球目标**，同时也可以快速**夺回控球权**。
- 特殊情况下被特殊考虑，如**与场地边界的距离**和**与基础位姿的距离**等。当球在对方半场的情况下，我们也会将基础位置稍微**向对方移动**，以便在对手夺回球时**快速施压**。由此得到的评级如下图（b）所示。据此，我们同样构造了一个**评级函数 Rating Function**，并搜索其最大值。
- 特殊情况：针对**任意球FreeKick**，该状态可以因为是处于进攻状态，因此其评级函数类似于前锋Forward的评级函数，但是**对直接射门的概率的权重较小**。

2个Forward前锋



(a) Rating of positions for both forward players.



(b) Rating of positions for the midfielder.

1个Midfielder中场

进攻决策

- **角色Roles--PlayBall带球**

- 这个角色实现默认的踢球PlayBall行为。它是由行为分配的目标是可以最快地**到达对手射门的目标方向**，**从而快速获得球权。**
- 该角色的意图由一个**SmashOrPass算法**计算得出，该算法可以决定是**射门**还是**传球给队友。**这些行为成功的概率是通过一个利用人工特征工程的**评级函数**来估计的，**评级函数**之后会进行具体介绍。
- 一些遗留的算法仍然被保留，它在做出决策时不会考虑环境中的障碍物，这个可以通过一个bool变量控制是否激活。

进攻决策

- **技能Skills—PassToTeammate传球**

- 当Strategy Behavior Control策略行为控制设置了传球技能请求，且当前**没有在处理特殊情况时**，在**路过的机器人**上会执行传球技能。
- 首先，根据收到的团队信息，**确定传给队友的位置**，即技能请求参数中相应号码的机器人的位置。
- 然后，利用**障碍物模型obstacle model**，从感知到的球的位置处计算出一个**有角度的扇形**，包括**自由扇形**和**被障碍物阻挡的扇形**。由此计算出**传球角度**和**target目标位置**，该位置尽可能**靠近队友的位置**，同时减少在传球路线上**障碍物阻挡球的可能性**。
- 在**与传球距离匹配**的踢球类型中，我们基于**踢球的距离**和机器人**到达踢球位置**的时间，选择一种能**最快速、最准确**地踢到传球目标点的踢球方式。备选的踢球类型可以根据情况进行限制。例如，尝试实现乌龙球可以使用**力量最强**的一脚射门，它可以滚到8米远，深入对方半场，立即**从防守转变为进攻**。在常规比赛中，这种踢法**不能用于传球**，主要是因为距离太远。但它的假设是所有机器人站着不动，因此当对手在附近时它的执行速度会变慢。
- 注意，在set状态开球时，规则限制在球周围**75厘米的半径内**不允许有对手球员，并且有足够的时间给执行机器人。因此，球员会等到**set状态结束后**才会执行传球。这将增加所有队友到达当前常规比赛中指定的目标位置的概率。在球前的对球点和对球角度，是根据**规划出的轨迹**和所选定的**踢球类型**来提前计算得到的，以减少在执行踢球动作时临时的调整，减少对球时的**踱步**。然而，为了避免不必要的延迟比赛，机器人只有在**障碍物扇区**挡住队友位置时才会进入等待状态，这种情况下没有足够宽的传球路径可用，并且预估出传球不太可能成功。否则，在正式的比赛中，只有在**无需球前的等待**时才会执行传球。

进攻决策

- **技能Skills—ReceivePass接球**

- 该技能在机器人知道队友准备传球给它时执行，即收到的球队消息team message包含了一个**与该球员球衣上的数字相匹配的传球目标。**
- 传球的队友计划的踢球目标，就是接球手应该走向的**目标位置。**
- 并且将接球的机器人的方向设置为**面向正在传球的队友**，这样接球者在球滚动到达之前就能看到球，以便进行必要的调整步骤来**预判接球。**

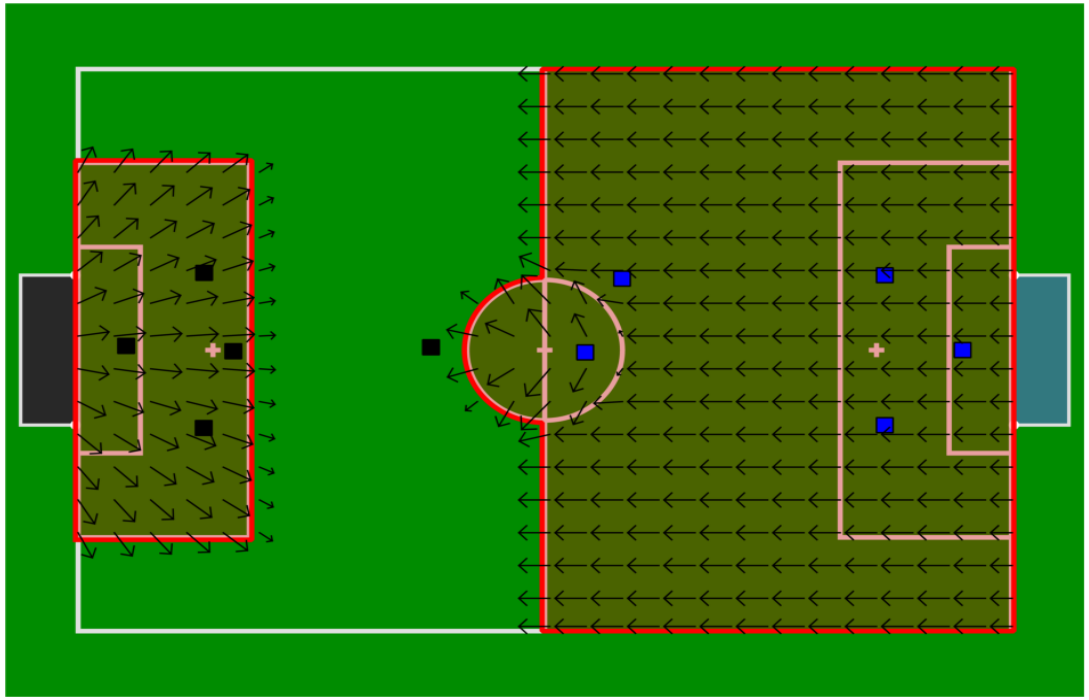
进攻决策

- **技能Skills—WalkPotentialField-行走势场**

- 根据规则，在某些情况下，机器人进入一些特定的区域是非法的。因为，进入这些区域会**导致时间损耗**。这些**非法区域**包括：
  - 有**3个本队机器人在本方禁区内时，**本方禁区不得进入。在**点球**时，本方禁区将成为**除门将外**所有机器人的禁区。
  - 有3**个本队机器人在对方禁区内时**，对方禁区不得进入。
  - 在对方球队**开球时的对方半场**。在本队开球时，除整个中圈外，对方半场为非法区域。
  - 对手开球时的**中圈**。
  - **任意球**时，**球周围半径为75厘米的球区**。
- 表示（Representation）**IlegalAreas**提供了一个函数来检查给定的位置是否在(或将在) 非法区域内，并且可以通过一个参数向外扩展该非法区域。此外，增加了25厘米的边线外缓冲区。
- 当机器人即将进入一个非法区域时，**HandlellegalAreas**选项Option将通过让机器人**站住**并**观察非法区域内的既定目标**来防止机器人进入内部(即成为非法区域)。一旦既定目标周围的区域**恢复合法**，机器人将继续**直接走向**目标。例如，在对手任意球期间，防守机器人会**在球区边界等待**，直到裁判宣布“无球-Ball Free”。

进攻决策

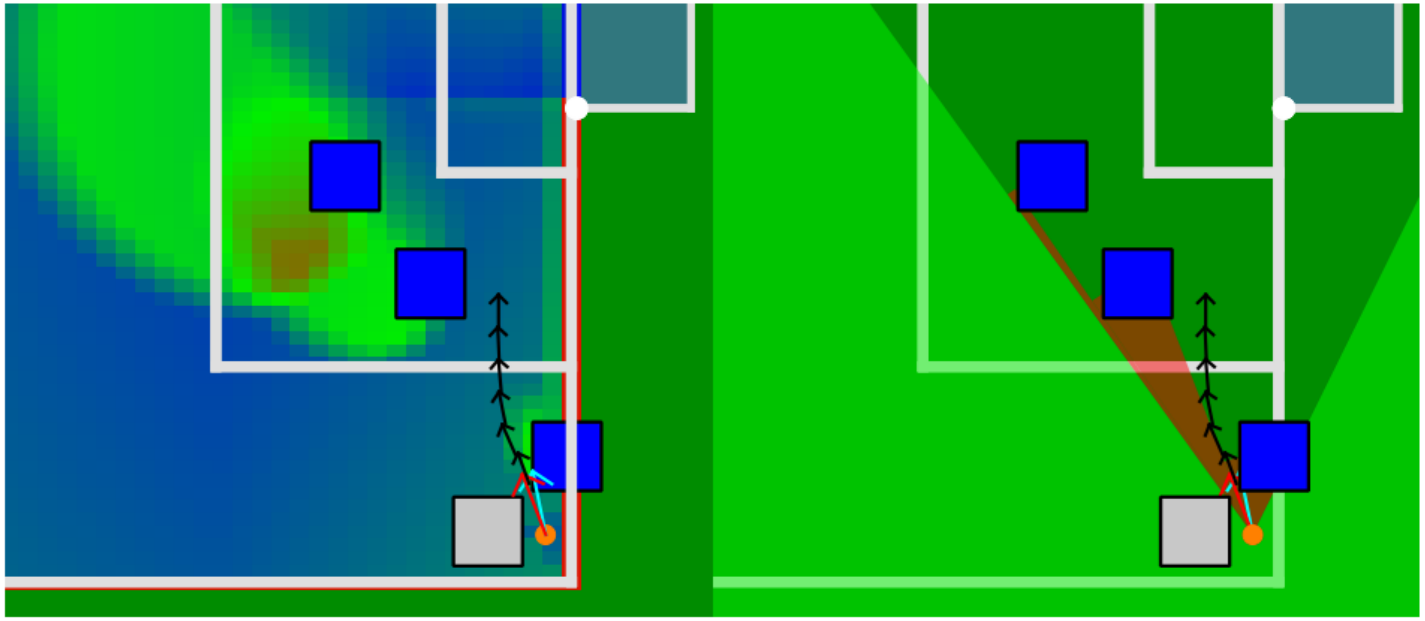
- **技能Skills—WalkPotentialField-行走势场**
- 当机器人**已经进入**一个非法区域时，**WalkPotentialField**技能将被调用。
- 该技能使用**FieldDimensions表示**来计算当前在字段坐标中的非法区域。然后，通过在非法区域内**施加最大排斥力**来计算**势场（Potential Field）**。
- 从非法区域的边界开始，排斥力**与向外的最大距离成正比**。如果机器人检测到它当前的位置在一个非法区域内，机器人通过调用WalkToPoint技能，使用**由势场给出的相对角度**离开该区域。此外，当机器人向非法区域外行走时，其**朝向**可以设置为**看向目标区域**(例如任意球期间的球)。该技能提供的调试图如下图所示。



➢ 从黑队(左)的角度看对手开球时机器人的**非法区域**(红色矩形)和**势场**(箭头)。本方禁区是非法的，是因为**内线有三个以上的队友**。

进攻决策

- **技能Skills—DribbleToGoal-运球到球门**
- DribbleToGoal是运球到球门技能，但会尽量**避开障碍物**，场地边界处则保证**不运球到外面。**
- 使用了一个**FieldRating表示**中的势场。在最新方向的基础上，势场方向在每一个循环中**向前确定1厘米**，直到达到1米的距离。
- 然后使用**结束时的位置**来计算新的运球方向。然后，利用**扇形区域**和**到场地边界的距离**，确定实时的最佳运球方向，如下图所示。



这项技能效果不错，但仍有一些不足之处:

1. 在对方门柱附近，结果往往是**直接带球进入门柱。**
2. Walk Kick行走带球时，**没有使用不同的Kick踢球距离范围**。原因来自于踢球Kick本身，因为它们有很大的范围偏差。
3. 靠近球场边界的踢球是通过**与球场边界的最小距离**来处理的。此外，使用最小的角度可以进一步降低踢到外面的风险（沿着边线的方向踢球，而不是垂直于边线踢）。

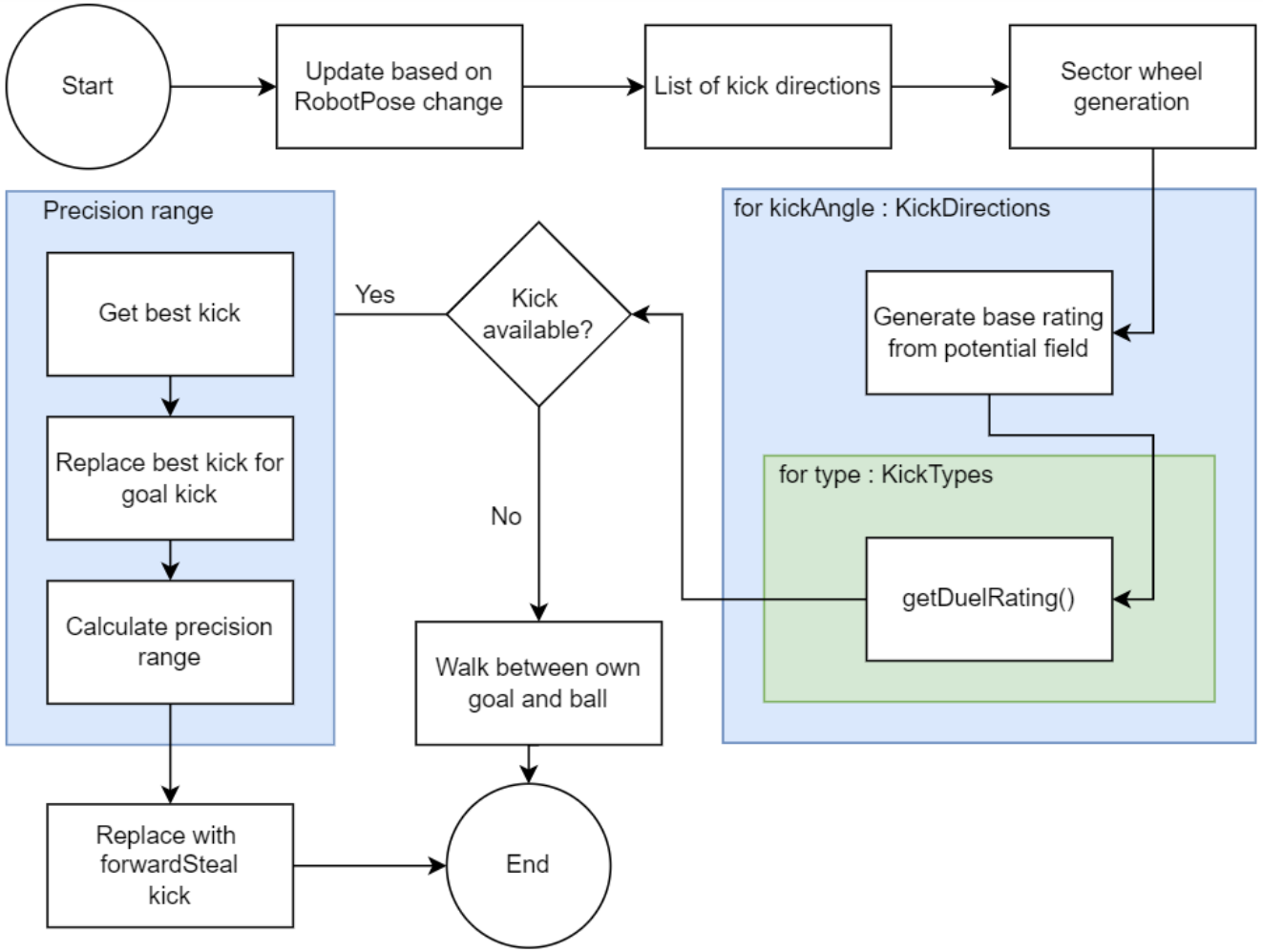
➢ 在运球技能中使用的势场(左)和扇形区域(右)。绿色箭头表示从势场出发的运球方向，红色箭头表示实际执行的方向。黑色箭头表示迭代坐标点时的中间步骤。

进攻决策

- **技能Skills—Zweikampf-对抗过人**
- 当一个机器人想要**踢球Kick（射门）**，而另一个对方机器人离得很近时，它就会使用Zweikampf 技能。
  - Zweikampf是一个德语单词。在足球语境中，它指的是**两名球员都靠近球**，想要获得或**保持球权**的情况。这包括有重大身体接触的情况，如铲球。
- 根据最后一次的踢球方向生成一个可能的**踢球方向的子集**。
- 然后，我们检查每个踢球方向的一些**约束**，和由**表示FieldRating**提供的**势场的评级子集**。然后我们为每一个方向**确定每种踢球类型的最终评级**。并且通过一些约束过滤掉部分踢法。此外，降低踢**向其他机器人**的踢球类型的得分，**提高射门**和与之前的踢球类型相似的踢法的得分。
- 现在我们有了一个带有**方向**、**距离**和**等级**的踢类型列表。它们被分为四类：**进球goal、抢断steal、传球pass、其他other**。进球类型的踢法（射门）总是比其他类型的踢法要好，即使评分差很多。
- 最后，计算精确的距离。这个信息被Motion线程所使用，从而提前开始准备踢球，因为它知道踢球的方向从而进行方向的适当调整，同时**仍然执行自己的行走动作**。
- 如果所有的踢球Kick（射门）都被过滤掉了，机器人只会**在己方球门和球之间**行走。在短暂等待后，开始使用DribbleToGoal技能。

进攻决策

- **技能Skills—Zweikampf-对抗过人**
- 当一个机器人想要**踢球Kick（射门）**，而另一个对方机器人离得很近时，它就会使用Zweikampf 技能。



茨维坎夫Zweikampf技能真的很好，但仍然有一些问题，计划在未来解决：

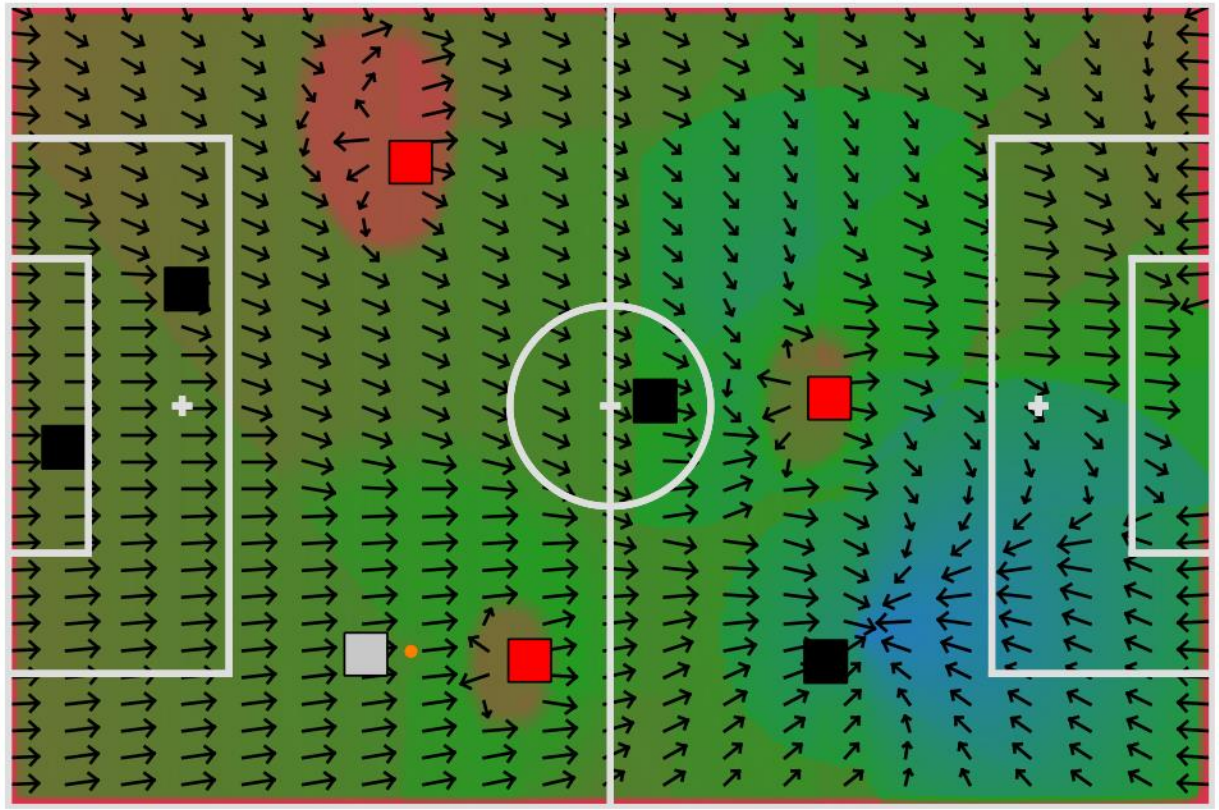
1. 该技能仍然需要**太多的计算时间**。势场花了大部分时间。许多范围并不需要计算，因为大部分的踢法会被过滤掉。
2. **需要更多的迟滞效应**。例如，射门Goal Kick在一个计算周期中被过滤掉后，但在下一个计算周期中再次被使用。因为踢球方向在每个循环中都是调整的，当踢球方向**在障碍物区域的边界上**时容易来回跳变。
3. **传球技能的请求仅仅被用于让势场提高队友的评级**。如果距离太远，茨维坎夫Zweikampf技能不会运球到传球目标，此时射门会优先于传球，而回传也会完全被过滤掉。
4. **靠近球场边界的踢球是通过与球场边界的最小距离来处理的。**此外，使用最小的角度可以进一步降低踢到外面的风险（沿着边线的方向踢球，而不是垂直于边线踢）。

进攻决策

- **模块Modules—ExpectedGoals-射门期望**
- 这个新模块负责**估计从一个位置射门得分的概率。**
- 提出了两种不同的方法，主要区别是，一个考虑了外部环境的已知障碍，另一个则忽略，从而提高适应性和速度。
- 这两种方法都考虑了**到对手球门的距离**来评估射门机会。
- 第一种方法目前大部分behavior行为使用，它计算踢球点到对方两个门柱之间的连线所形成的扇形区域的角度范围，来估计射门成功的概率，且不能被障碍物阻挡。

进攻决策

- **模块Modules—FieldRatingProvider-场地线提供器**
- FieldRatingProvider模块为Zweikampf和DribbleT oGoal技能提供了**计算势场**的函数。
- 它们都使用了基本的线性缩放，从排斥点或者吸引点开始进行计算，得到各种势场，然后基于他们的评级**加权求和**，他们为场上的每个点给出一个得分值，作为机器人的目标点。不同的势场求和的一个例子如下图所示。



当前使用的FieldRating的问题:

1. 有些势场是**有明显边界**的，而不是平滑的插值。
2. Pass**传球势场可以更大**。
3. 我们的球衣检测还不够好，**无法区分对手和队友**。
4. Facing机器人的**面前势场可能会被删除**。

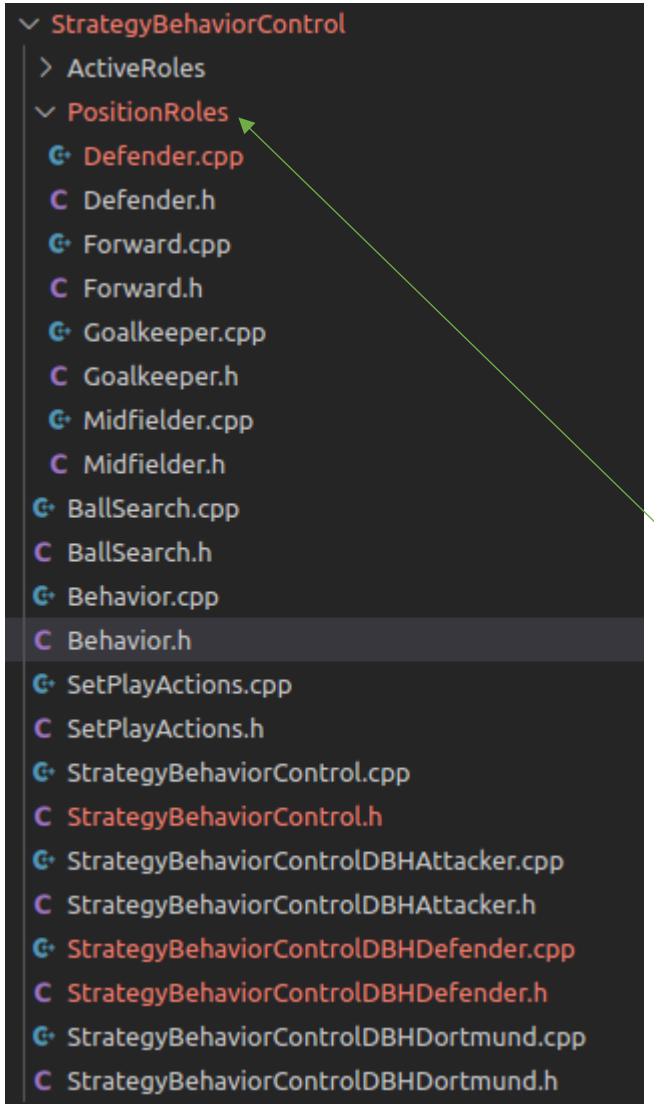
➢ FieldRatingProvider中的所有势场的和。黑色矩形是队友，红色矩形是对手，白色矩形是我们的带球机器人。橙色的圆圈是球。蓝色区域被评为**好**，棕红色区域被评为**坏**。箭头表示势场方向。

进攻决策

- ## 模块Modules—Ball Search-找球

- **角球corner kick**和球门球**goal kick**，这两种战术采用了Ball Search找球行为。
- 这两种情况包含了球场上8个可能的位置，球可能的位置：**球场的四个角**和**球门区域的四个角**。目标是有效地搜索这两个位置，而不是让所有球员四处走动。
- 第一步是查看当前哪个比赛状态，以及是哪个团队进行的。根据是在己方半场还是对方半场的情况，**只有在这两个半场的球员会检查**，如果这两个位置中的一个在他们的Voronoi**维诺图**区域内。
- 例如，在发生角球时，防守球员不需要检查是否在自己的区域内，因为在这种情况下，两个位置都在对方半场。
- 如果两个位置都在同一名球员的Voronoi区域，一名中场球员会移动去检查角球，角球离该球员更远。其他人移动到预定的位置，直到球被找到或设定的比赛结束。
- 守门员特别不会主动移动去寻找球，即使球在自己的一半。因为守门员不会让球门毫无防备。
- 出于同样的原因，**守门员把Kick踢球的任务交给后卫（守门员不会踢球）**，并给他们腾出空间。
- 由于处罚，并不是所有的球员都在场上，因此定义了维持防守的最低人数。
- 在机器人世界杯的正常比赛中，每个位置只有一名球员被派去，而其他球员在进球和角球情况下都转移到他们的初始基本位置（**根据区域分工**）。
- 不幸的是，这种走到基础位置也发生在比赛开球期间，针对性的球搜索技能还没有实现。这种情况在机器人世界杯上发生了两次，结果球队丢了球。

代码阅读 文件路径： /Src/Modules/BehaviorControl/



Strategy（策略/决策）

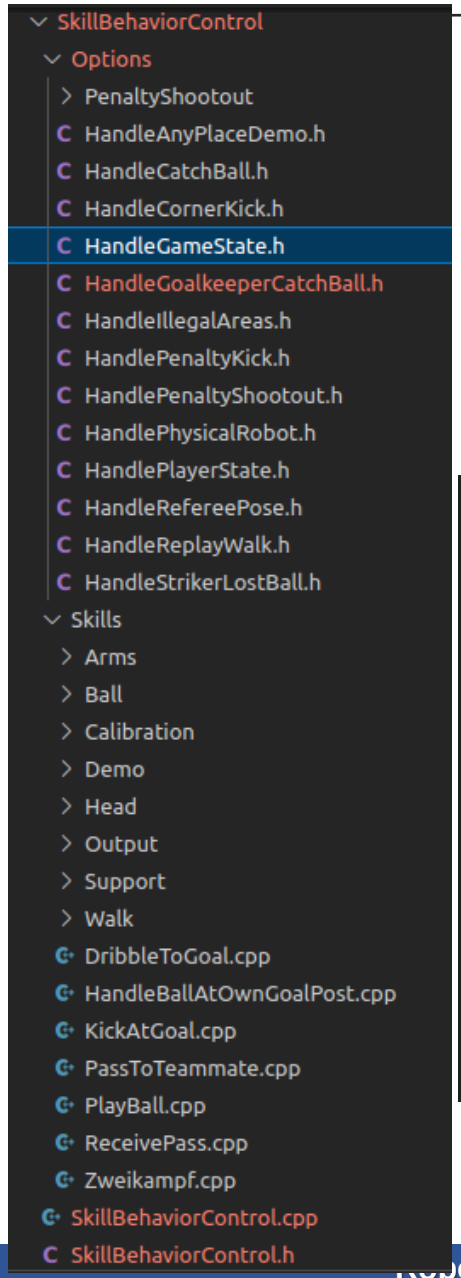
Behavior框架：/StrategyBehaviorControl/Behavior.cpp

- **载入比赛的初始策略配置文件：**
  - /Strategies/game.cfg（阵型变换条件，如球在我方半场选择t211，球在对方半场选择t112 ）
  - Config/Behavior/Tactics/t112.cfg t211.cfg（初始站位阵型）
- **进入update函数循环， 不断执行：**
  - **根据阵型分配站位assignPositions**
  - **根据站位分配角色assignRoles**
  - **根据角色执行相应skill技能请求execute() → SkillRequest**

```
79    /**
80     * Executes the selected role for an agent.
81     * @param agent The agent for which to do the calculations.
82     * @param otherAgents All other agents (excluding \c agent).
83     * @return The resulting skill request.
84     */
85    SkillRequest execute(const Agent& agent, const Agents& otherAgents);
```

代码阅读

文件路径： /Src/Modules/BehaviorControl/



Skill（技能）

/SkillBehaviorControl/**SkillBehaviorControl.cpp**

```
72    select_option(options);
```

Config > Scenarios > Default > skillBehaviorControl.cfg

```
1   options = [
2     HandlePhysicalRobot,
3     HandlePlayerState,
4     HandlePenaltyShootout,
5     HandlePenaltyKick,
6     HandleGameState,
7     HandleIllegalAreas,
8   ];
9
10  playingOptions = [
11    HandleGoalkeeperCatchBall,
12    HandleStrikerLostBall,
13    HandleCatchBall,
14  ];
```

ExecuteRequest函数主要内容：

1. 如果处于射门shoot、传球pass、运球dribble三种行为时，调用thePlayBallSkill()
2. 否则，判断是否有人要传球给我，如果有，调用theReceivePassSkill()
3. 同时，判断自己此时的SkillRequest请求类型：站立stand、行走walk、封堵block、盯防mark、观察observe

状态跳转入口：SkillBehaviorControl/Options/**HandleGameState.h**

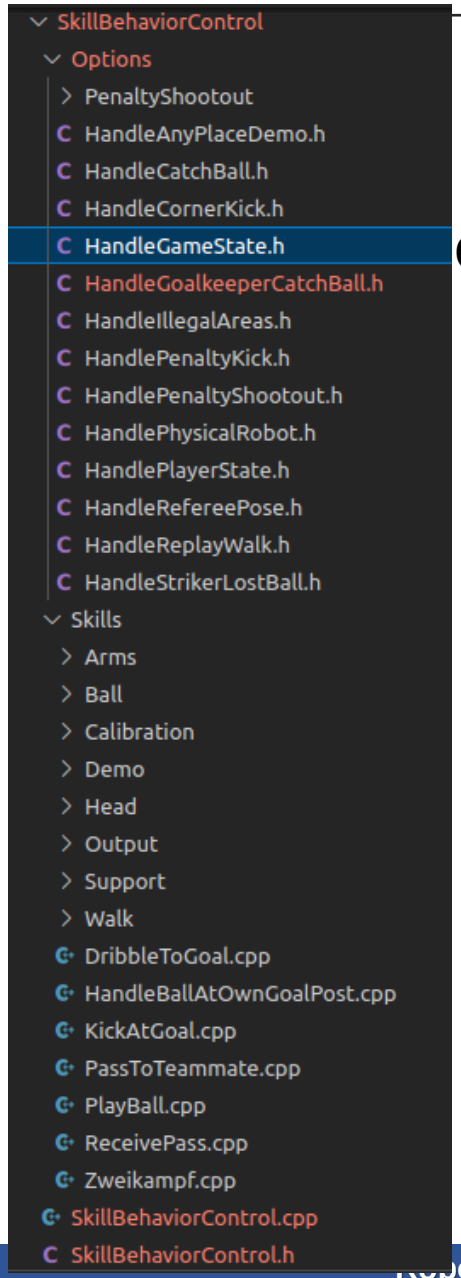
```
18  // The default state is "playing".
19  initial_state(playing)
20  {
21    action
22    {
23      theArmContactSkill();
24      if(theStrategyStatus.role != PositionRole::toRole(PositionRole::goalkeepe
25        theArmObstacleAvoidanceSkill();
26      if(!select_option(playingOptions))
27        executeRequest();
28    }
29  }
```

执行主要**技能**函数/SkillBehaviorControl/**SkillBehaviorControl.cpp**，函数executeRequest()

Options/HandleGoalKeeperCatchBall.h：守门员扑救相关

代码阅读

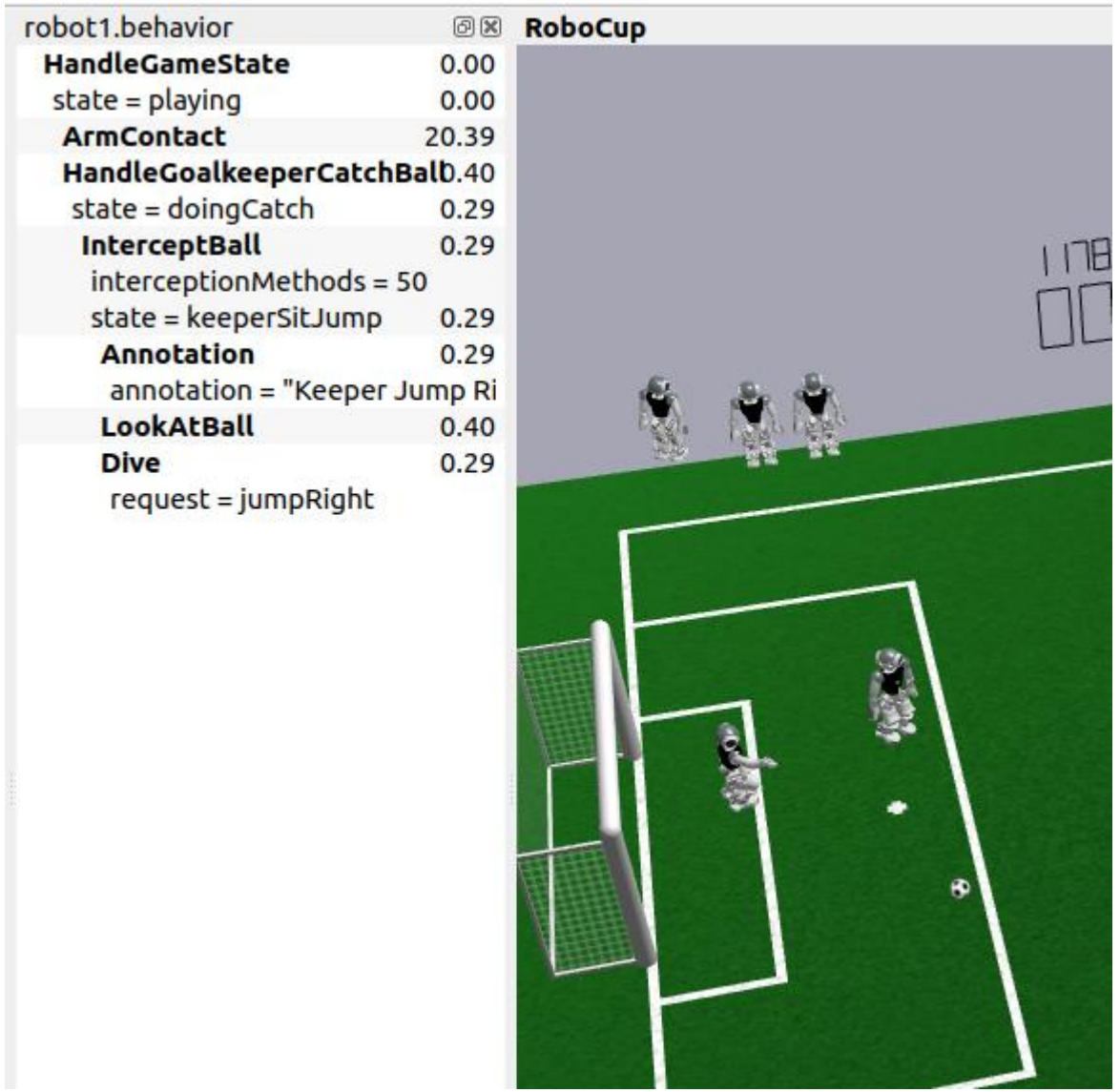
文件路径： /Src/Modules/BehaviorControl/



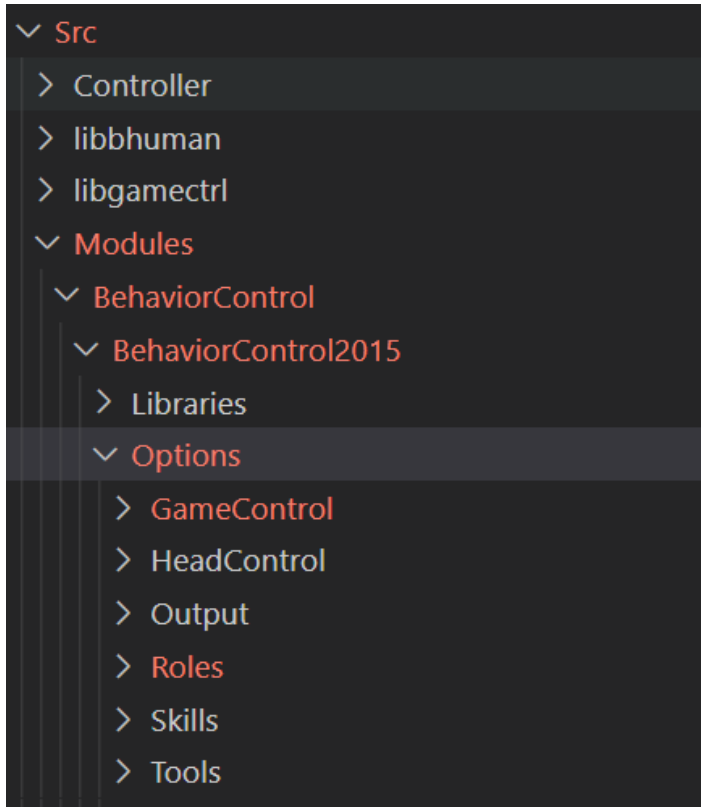
Skill（技能）

Options/HandleGoalKeeperCatchBall.h：
守门员扑救相关

```
13  namespace Interception
14  {
15    ENUM(Method,
16    {,
17      stand,
18      walk,
19      genuflectStandDefender,
20      genuflectStand,
21      jumpLeft,
22      jumpRight,
23    });
24  }
```



主要关注代码位置



Build：文件夹中包含编译好的程序，包括bush，simRobot等
Config：配置和脚本文件，如simRobot中读取的场景文件以及连接机器人的网络配置等等。
Install、Make：实体Nao机器人创建、安装bhuman，编译程序
Src：所有源码
Util：工具集
Modules 与 Representations关系（repre添加对象，module实现对对象的更新）调用Representations中的函数，编写每一个角色策略文件;
Modules是实现Representations的模块,Modules中的函数具体实现representation的相关定义函数并执行相关action
Roles Skills Libraries
Roles文件夹下面包含5个角色的策略文件
Skills文件夹下面包含了写角色策略时需要的一些集成的技能模块(自己针对特殊或某一特定情况写一个skills文件)，包括所有角色都通用的一些技能
Libraries文件夹下面包含了很多库文件，包含了如何实现角色的一些功能函数，针对特定的类型创建的库文件