WonderPines 仿真 2D 机器人足球队描述文档

张凯旋，刘继多，朱世林，邵健
安徽省黄山学院创新实验室

1 黄山学院 2D 仿真足球队简介

黄山学院 2D 仿真足球于 2009 年建立，WonderPines 采用 agent2d 作为开发球队的底层，通过队员们共同努力，在底层的基础之上，我们做了很大的改进。球队水平较底层已经有了明显的提高。

2 底层简介

（1）底层版本

底层版本：agent2d-3.0.0 库版本：librcsc-4.0.0 下载地址：http://wrighteagle.org/2D/

（2）底层框架

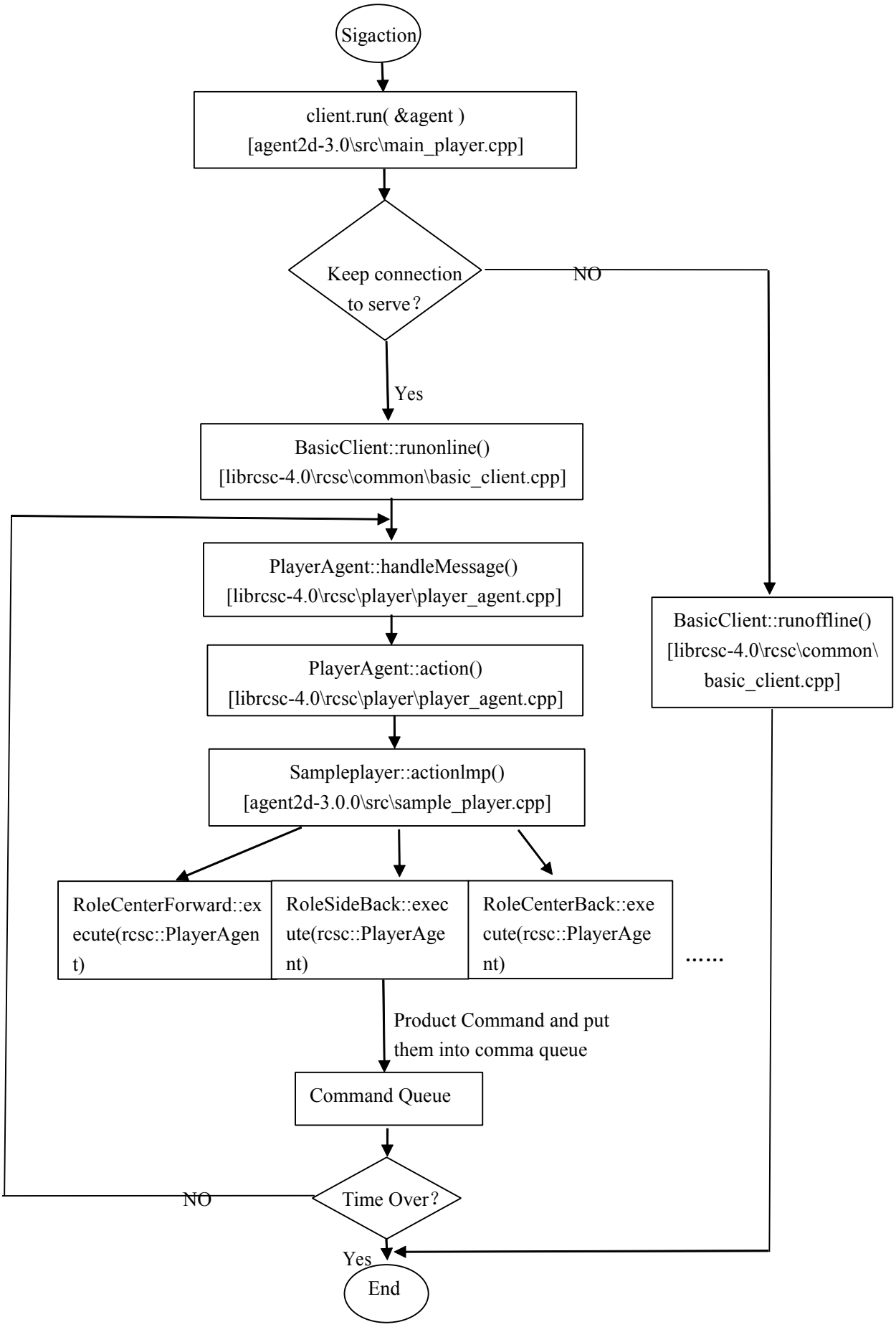
agent2d-3.0.0 是基于角色的决策系统，主要有两大基类：rcsc::SoccerBehavior 和 SoccerRole。

rcsc::SoccerBehavior 是对球员的行为进行抽象，封装成类。此类可以派生出很多的子类如 Bhv_BasicMove，Bhv_BasicOffensiveKick，当然也有我们自己添加的类如 Hsu_Center-ForwardMove，Hsu_HalfOffensiveMove2011 等等。这些子类通过有参数或者无参数的构造函数来生成对象，而子类的成员函数中“执行此动作”的函数是 bool execute (rcsc ::PlayerAgent * agent)，参数为指向某个 agent 的指针；

SoccerRole 类是抽象出来的球员类型基类，即球员在球队中扮演的角色。其中底层提供了由其派生出来的子类分别代表着不同类型的球员。如RoleSideBack（边前锋）、RoleSide Half（边中卫）。SoccerRole 共派生出了 1 个守门员，2 个边前锋，1 个中前锋，2 个边中卫，1 个中中卫，2 个边后卫，2 个中后卫。除此之外，底层还提供了用于训练球员个人技能的两个子类进攻球员与防守球员。

每个角色的决策，主要是由 bhv_chain_action.cpp 控制，bhv_chain_action.cpp 又向下树形展开了很多文件，这些文件主要完成对当前形势的评估，然后产生一个候选动作，将此动作放入队列，及 queue_command, 然后按照先进先出的原则以此向服务器发送这些命令。当队列为空时，底层中就默认将一个 hold_ball 持球动作，送入队列。由于所有的球员均采用同一个决策标准，因此对于球员的异构便不是很明显。尤其在进攻时，表现的最为突出。进攻时，无论哪个球员控球，都会把球往拐角里带，而且都是一样的长时间持球, 这也就是底层的一个 bug。

（3）agent 执行流程


Sigaction
client.run(&agent)
[agent2d-3.0\src\main_player.cpp]
Keep connection to serve?
NO
Yes
BasicClient::runonline()
[librcsc-4.0\rcsc\common\basic_client.cpp]
PlayerAgent::handleMessage()
[librcsc-4.0\rcsc\player\player_agent.cpp]
PlayerAgent::action()
[librcsc-4.0\rcsc\player\player_agent.cpp]
Sampleplayer::actionlmp()
[agent2d-3.0.0\src\sample_player.cpp]
RoleCenterForward::execute(rcsc::PlayerAgent)
RoleSideBack::execute(rcsc::PlayerAgent)
RoleCenterBack::execute(rcsc::PlayerAgent)
……
Product Command and put them into comma queue
Command Queue
Time Over?
NO
Yes
End
BasicClient::runoffline()
[librcsc-4.0\rcsc\common\basic_client.cpp]

3 在 agent2d 底层上所做的改进

3.1 底层上所添加的文件（相应的头文件不再赘述）

hsu_shoot_area_kick.cpp\
hsu_pass.cpp\
hsu_center_forward_move.cpp\
hsu_half_defensiver_move.cpp\
hsu_side_forward_corner_kick.cpp\
hsu_half_defensiver_kick2011.cpp\
hsu_basic_defensive_kick.cpp\
hsu_half_offensive2011.cpp\
hsu_half_offensive_move2011.cpp\
hsu_mark_dribble.cpp\
hsu_block_side_attack.cpp\
hsu_fast_mark.cpp\
hsu_exact_shoot.cpp\

这里的每个文件相应的会创建一个新类，这些类都是继承基类 rcsc::SoccerBehavior，是球员行为动作的子类的扩展，也是我们的高层决策的核心。

3.2 所做工作

底层中的 strategy.cpp 文件是统领整个高层决策的指挥官，该文件中实现了创建角色（create role）、创建阵型（create formation）、评估当前场上形势包括：{ Normal _Situation,Offense_Situation,Defense_Situation,OurSetPlay_Situation,OppSetPlay_Situation,PenaltyKick_Situation}

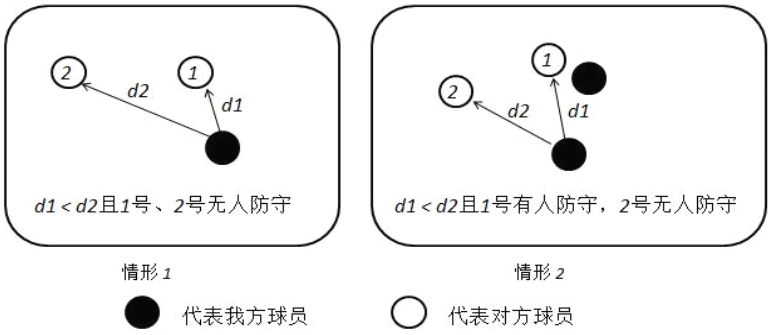
根据不同的形势来创建不同的阵型、对球场区域进行了划分，将球场区域划分为：{BA _CrossBlock,BA_DribbleBlock,BA_DribbleAttack,BA_Cross,BA_Stopper,BA_DefMidField,BA _OffMidField, BA_ShootChance, BA_Danger } 九大区域。以便于进行基于场地划分的决策系统、更新世界模型（update worldmodel）等。

我们的高层决策是基于角色和区域划分的混合决策。下面依次简介决策内容：

(1) 快速盯防策略

hsu_fast_mark.cpp：这是“快速盯防”策略，主要在非正常比赛模式下，如球门球，角球，任意球等模式下，对方发球时实施的有效防守动作。核心是对对手实行一对一盯防，迫使对方无法发球。为我方赢得断球机会，实验证明这是非常有效的防守方式。

快速盯防是适用于特殊模式下的快速有效的防守。在比赛暂停和发球之间有几个仿真周期可供场上球员跑位和防守，为了阻止对方球员将球发给其它球员，我方球员可以采用快速盯防的策略对对手进行防守，快速盯防的思想在于防守时选择距离自己较近（注意：不一定是距离最近的!）且无人防守的对手进行盯防，这样既可以实现快速防守，也可以节省体力。在比赛时常见的情形大致可分为以下两种：如下图（1）所示。



图（1）

针对情形 1：由于快速盯防的核心思想在于选择距离自己较近且无人防守的对手盯防，因而在扫描对方球员时，当 1 号和 2 号球员都无人防守时，要根据对手与自己距离的大小选择距离相对较小的对手并上前盯紧。同样对于对方所有球员，就需要对所有对方球员一一扫描并从中找到一个距离自己较近的对手看防。

针对情形 2：当扫描时对方 1 号球员有人防守，对方 2 号球员无人防守但 2 号离自己的距离大于 1 号，这时就要考虑应该盯防哪个对手了。由于比赛规则中不允许多于两个队员防守相同的对手，因此即使 2 号离自己较 1 号远，当前 agent 也该选择 2 号而不是 1 号进行盯防。同样这种情形下还可能是：对方 1 号、2 号球员离自己较近但都有人防守而 3 号距自己较远无人防守，这时 agent 就需要选择 3 号对手进行防守，所以说我们盯防的对手不一定是距离自己最近的。

（2）新增射门算法

hsu_exact_shoot.cpp：此文件的作用是实现快速射门。在 Robocup2D 比赛中，场上的形势瞬息万变，球员的站位也都充满了随机性，很难保证能找到一条安全的射门路径，但却可以找到其中一条成功概率较大的路径射门。因此我们做的射门决策是基于几何概率模型的，即在较大射门成功概率的区域中选取较优射门路径。

射门时决策的实现步骤如下：

1，根据对手守门员的位置选择成功概率较大的进攻区域 S1；

2，计算进攻区域 S1 的内切圆的圆心 O1；

3，以 O1 为圆心扫描周围离 O1 最近的对手 OPP；

4，如果 OPP 存在，则根据 OPP 的位置选取有效进攻区域 S2；

5，计算有效进攻区域 S2 的内切圆的圆心 O2；

6，根据 O2 计算出成功率较大的射门路径并判断是否安全，如果安全，则射门，否则将球传给队友；

通过计算 s1，s2 的面积可得 P(s1/s)和 P(s2/s)的大小，即可确定进攻区域 S1。

这种方法相对于底层中的射门决策的优点是复杂度低，执行效率高，因为比赛中球员所获取的信息都是不断变化的，如果不能在较短时间内做出决策，就可能会错失很多进攻机会。但这种决策也有一些缺陷，比如决策过程不够准确，因为任何事情都具有随机性，只是以概率的方式进行决策的话，成功射门的情况会带有一定的随机性。也可能因为场上环境的原因（比如噪声对球员的干扰、风向对球运动轨迹的影响等）而忽略了一些安全性低但却很可能成功的射门路径，所以这种算法还有于改进。另外，这种决策还有一个特点就是它符合一个真实球员在射门时的决策思想（找空挡的思想），所以也具有一定的仿真性。

通过计算 s1，s2 的面积可得 P(s1/s)和 P(s2/s)的大小，即可确定进攻区域 S1。

至于第 6 步，最后选取的路径虽然是成功概率较大的，但也只是相对其它路径而言，所

以还需要判断该路径是否安全。

这种方法相对于底层中的射门决策的优点是复杂度低，执行效率高，因为比赛中球员所获取的信息都是不断变化的，如果不能在较短时间内做出决策，就可能会错失很多进攻机会。但这种决策也有一些缺陷，比如决策过程不够准确，因为任何事情都具有随机性，只是以概率的方式进行决策的话，成功射门的情况会带有一定的随机性。也可能因为场上环境的原因（比如噪声对球员的干扰、风向对球运动轨迹的影响等）而忽略了一些安全性低但却很可能成功的射门路径，所以这种算法还有于改进。另外，这种决策还有一个特点就是它符合一个真实球员在射门时的决策思想（找空挡的思想），所以也具有一定的仿真性。

（3）优化传球模拟数据搜集

hsu_pass.cpp：为了能够预测传球的路线，以及预期的接球点，接球点周围环境的预测。我们在底层的基础之上，添加了一个判断传球路径的函数。此函数返回 pass_route“传球路径”这种结构体数据类型。很容易获取模拟传球的传球点，接球位置等信息，用来判断什么时候传球才是最好的传球时机。

（4） 进攻决策

组织有效的进攻，是我们决策的核心。WonderPines 采用“边线进攻”优先，但不是死板的非要将球带到底线才传中，在中前锋与边前锋的配合上应当灵活一些。如果有好的传球机会，尽量还是传球，因为传球更容易甩开对手创造射门机会。其实无论边线或者中路进攻，目的都是将球带至对方球门前有效射门区域附近，以实施自己的进攻阵型。由于边线进攻相对中场进攻容易一些，所以 WonderPines 还是优先考虑从边线突破。

选择合适的机会传球以及无球队员的跑位将是难点，对于 agent2d-3.0.0 底层而言，球员的决策就两种：有球决策和无球决策，有球策略主要由函数 Bhv_ChainAction(). execute (agent) 控制，无球决策均运用了同一个策略，就是跑本位点，由函数 Bhv_BasicMove(). execute (agent) 控制。并没有主动的进攻跑位和防守跑位。因此鲁棒性较差，没有灵活性。WonderPines 的进攻决策部分已经删除了底层中的策略，取而代之的是我们自己的进攻策略。

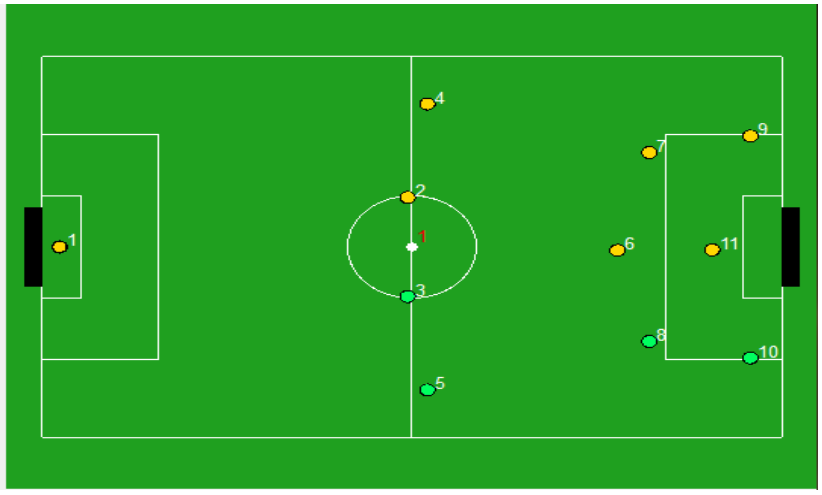
WonderPines 进攻思路为：

I：由边前锋优先从边线附近带球进攻，在带球过程中，时刻考虑有没有好的传球机会，若反馈回来的模拟传球结果不是最有利的，那么将不会冒险去传球。而选择继续从较为安全的边线继续进攻。若发现有好的传球机会，比如能够甩掉对手，创造射门机会，那么就传球。由文件 hsu_side_forward_corner_kick.cpp 完成；

II：当边前锋带球边线进攻时，其余队员跑位要及时跟上。（底层中的中卫经常是跑位滞后）由文件 hsu_center_forward_move.cpp　hsu_half_defensiver_move.cpp　hsu_half_ offensive _move2011.cpp 完成跑位；

III：若没有好的传球机会，边前锋将球带至较为靠近底线的位置后，中卫与前锋站位成一个弧形围绕在对方球门前，中前锋将站在对方球门的中点前方，随时准备接传中的球，由 hsu_shoot_area_kick.cpp 文件完成；

IV：站成弧形后，通过短而快的传球，打出空档获得射门机会。控球球员的有球决策由 hsu_half_defensiver_kick2011.cpp\hsu_half_offensive2011.cpp\hsu_side_forward_corner_kick.cpp\hsu_shoot_area_kick.cpp 完成。如下图（2）所示，中场配合前锋站成圆弧形是我们的进攻策略。



图（2）

圆弧形的站位加上短而快的传球，是我们最终要做成的进攻站位效果

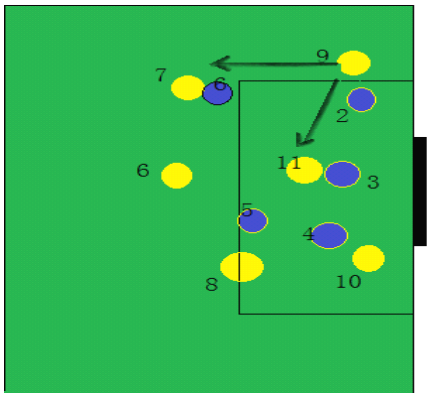
在进攻决策中，球员之间的配合时很重要的，而跑位则是重中之重。底层中用 CDT（Concstrained Delaunay Triangulation and Posioning Mechanism ）即约束三角形定位与线性差值算法来模拟大量的本位点。这种跑位有别于国内的其他主流球队通过对对球的吸引力（或者称对球的敏感度），最大 x，最大 y 的算法。CDT 将球场划分为若干个三角形区域，通过对其三角形的外接圆作为边缘建立了约束三角网来实现跑位。然而，这种算法也有缺陷，一是跑位滞后，二是，很难对其进行修正。有时场上的变化较大，但是跑位点是固定了的，在修正过程中非常复杂。目前，WonderPines 通过对部分本位点的修改以及用函数的方法对局部跑位点进行修正。

局部跑位点的修正算法：

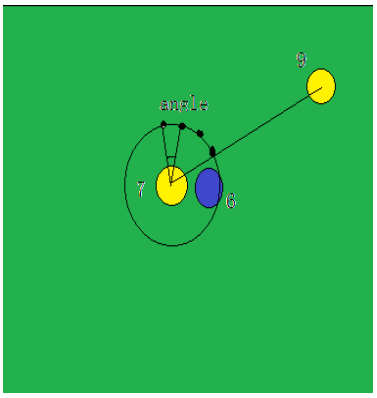
底层跑位均调用 bhv_basic_move.cpp，此函数的核心就是每个球员都跑事先确定的本位点。而本位点的选取，是通过大量模拟球落在球场的位置而事先确定。将写好的阵型文件保存在 src/formations-dt/文件夹中，文件的数据结构为：球的位置坐标，以及相应的每一个球员所要的跑位点坐标构成一个数据类型。

当 self 不是控球时，首先判断自己是不是所有队友中要接球的队员，如果不是，则进行简单的跑本位点。如果是，那么就要找到一个点去接球，这个点是以自己当前位置为圆心，R_dist（大小自定）为半径的的圆弧上，可以以与控球队友之间的角度为角度的中心，以正负 30 度为上下界限。具体的角度选择，还应考虑到对手的身体朝向和位置。每隔一定角度 angle，落在圆弧上的点。把这些点放进容器内，遍历的同时，进行筛选。最终筛选出符合条件的局部修正跑位点。

如下图（3）所示，7 号和 11 号是最有可能接到 9 号(当前控球)传球的球员（可用自己距离拿球队员的距离简单判定自己是否是最有可能接球球员），那么 7 号和 11 号都要执行局部跑位，如图（4）黑点则代表每个候选的跑位点。



（图 3）



（图 4）

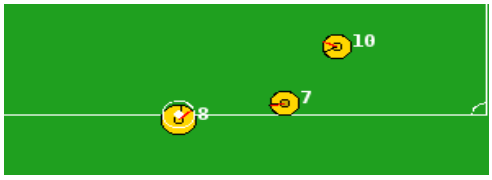
除了此修正部分跑位点的算法外，WonderPines 当前场进攻时，对于一些特殊的情况也作出了相应的处理，这里不再说明。

（5）特殊模式

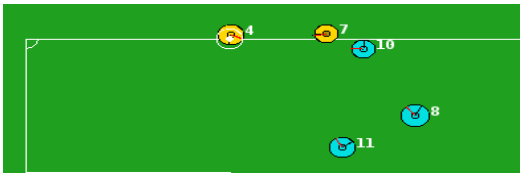
1，在我方发角球、边界球模式时边线处对方球员防守较弱，而且为了进一步完善我们球队的边线进攻策略。在角球、边界球模式时我们采取了：接球球员的边线站位法（如下图（5）所示）。

2，由于现在大多数球队在对方球队发角球、边界球等时对对方接球球员实行“一对一防守”策略。

我们在边线站位法的基础上我们采取了：让我们的接球球员站在了 1 米内，而根据大赛的规则，对方球员不能站在该范围内，从而有效的避开了对方的防守，让球顺利的发给了我方接球球员。如下图（6）所示。



图（5）



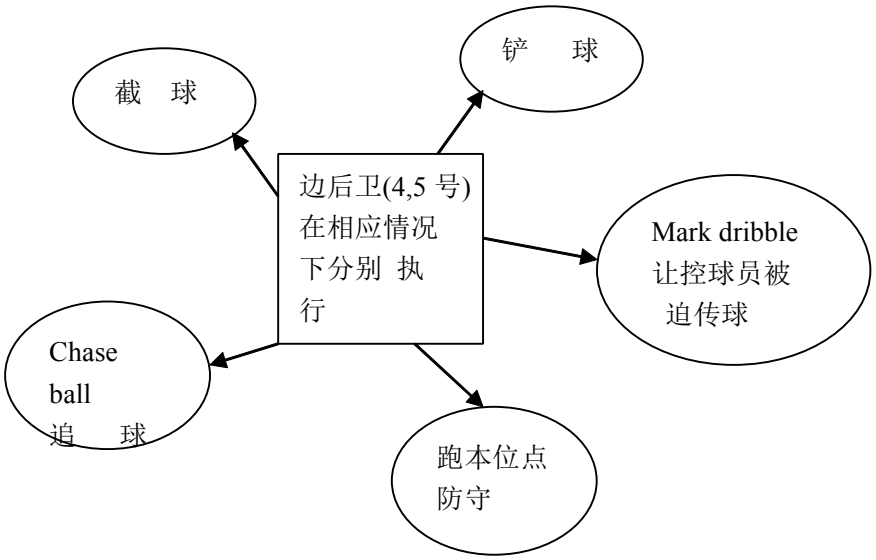
图（6）

（6）边线防守策略

该策略写在我们添加的 hsu_mark_dribble.cpp\hsu_block_side_attack.cpp\ 两文件中。

为了阻止对方的边线进攻，我们实施了“一盯一”封堵对方控球队员，封堵不成功，就会一直紧随对方控球球员，或者阻止对方运球（mark dribble）以及迫使他传球给他的队友，如果对方已控球至我方球门附近，则跑至自己的本位点进行防守。

大致的思路是：我们采用几何算法来预测边线对手带球进攻路线，然后通过在预测带球路线上寻找封堵点（block_point），评价封堵点的参数主要是自己和对手达到封堵点所花费的时间之差、自己与封堵点的距离和对手与封堵点的距离。



图(6)

（7）短传

即短距离的传球，是相对于后场反攻或者解围时的长传。这种短距离的传球，有以下几点要求：1，快速定位要传给球的那个球员；2，基于角色的传球，因为这样会减少模拟的时间；3，直接将球传到球员脚下；4，出球速度尽量快；5，以最小代价的跑位，即接到球就传，快速转移球，打出空档。

如下图（7）所示，若按照进攻的阵型站好位后，事实短传时检索接球员号码，在对方球门前站成半圆形阵型然后能够快速进行传球组织进攻。在传球过程中检索对手与队友确定目标点和安全区域。在进攻到对方球场时，队友组成半圆形阵型然后执行快速短传，扫描固定的球员与目标点。

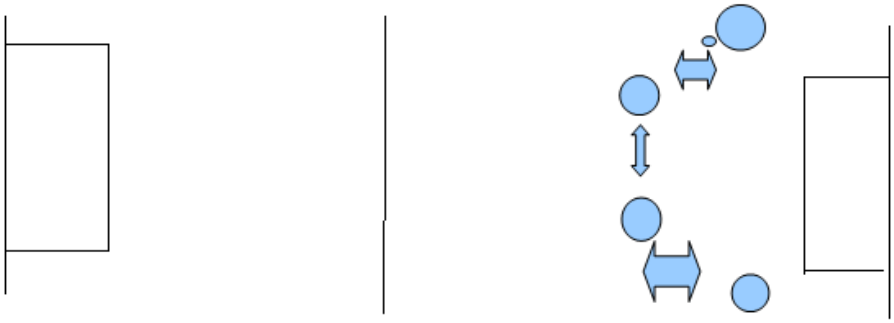


图 7

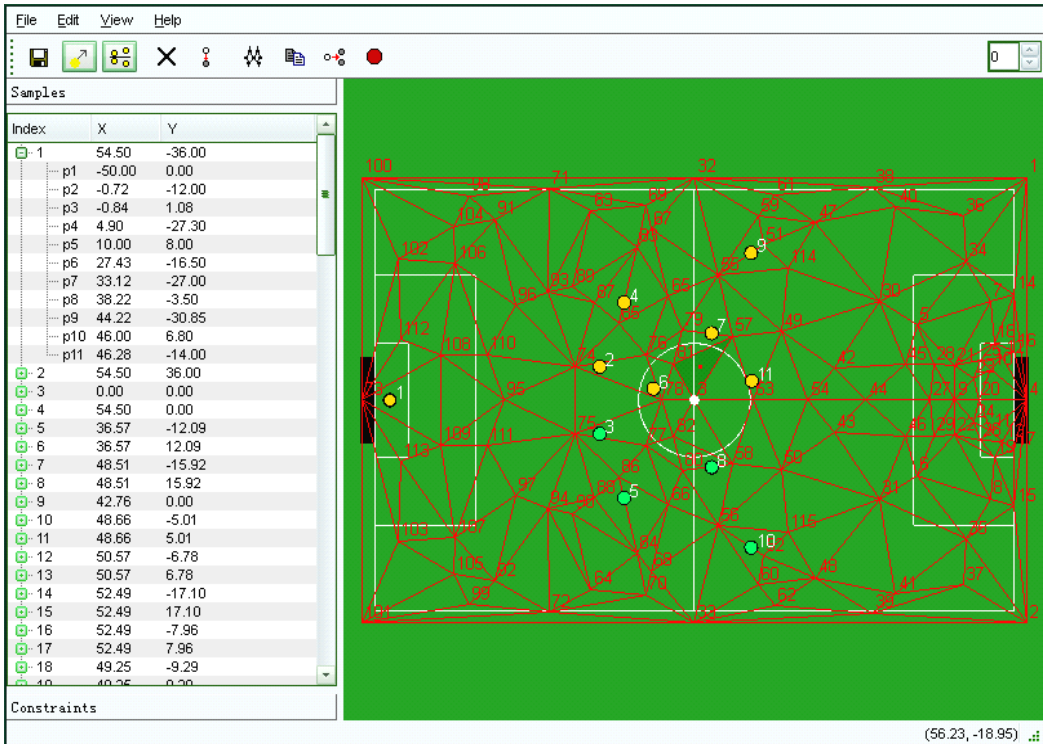
hsu_short_pass.cpp 这一策略也是为了更好的实现我们的进攻策略。由于时间原因，WonderPines 短传策略效果还不理想。从比赛中也可看出，WonderPies 丢球率很高，而且传球不灵活。因此也制约了我们的前场进攻，这也是我们未来要继续做的工作。

（8）其他

添加自己的队标 hsu_logo 

另外，我们采用配合底层发布的本位点编辑软件 fedit（即 formation edit），完成了对上

述部分进攻本位点和防守本位点修正和优化，以及开球前阵型的修改（before_kick_off）。也是为了弥补用函数法来修正本位点可能被各种异常情况所忽略。下面图（8）为该软件截图，通过该软件可以将所有本位点标出来，可以很直观地观察到本位点的分布。当然，更改本位点和创建阵型也变得非常简单。



图（8）

4 未来的工作

我们计划用几年或者更长时间的努力，最终将黄山学院 2D 仿真足球队建设成一只强队。目前，我们的弱点是：进攻时，传球有差错、丢球，带球时候容易被铲球。进攻跑位点、带球点选取优化与评价方法，各个权值的分配、精炼本位点形成的进攻网络等不太完善，希望通过队友们的努力，逐步完善上述不足。

由于我们选用的底层偏向于防守型，进攻比较薄弱。因此短期内我们的主要改进在于进攻决策，其后的改进重点为防守，未来，我们计划引入 AI 人工智能算法。现在，我们已经着手研究遗传算法，蚂蚁算法，鱼群算法，BP 神经网络等主流 AI 人工智能算法，希望通过努力，将 WonderPines 建设成一个自主学习型的球队。

5 总结

通过我们的努力，从进攻，防守，特殊模式等多方面做，WonderPines 球队水平已有明显的提高。但是，由于时间短，知识水平有限，很多细节和想法都没有能够实现。如，进攻决策达不到预期的效果，同时，鉴于 agent2d 底层本身的难度，要想把队伍建设成为国内一流的强队，决不是一朝一夕能够实现，需要我们多届同学和指导教师的努力，需要利用一切机会向其他高校的赛队学习，向其他高校的同学学习。