BJUT仿真2D机器人足球队描述文档

队员：冯云贺，李战，唐震
指导老师：左国玉
北京工业大学电控学院
zuoguoyu@bjut.edu.cn

一、队伍简介

自2008年起，北京工业大学仿真2D机器人足球BJUT队积极参加Robocup中国公开赛，并取得了一定的成绩。从去年比赛开始，我们所用的底层由UVA换成了Agent2D，由于两个底层差别较大，移植的难度太高，所以我们在继承了前辈们成果思想的基础上加以自己的创新，经过一年多的分析和研究，改善了新的底层，强化了高层策略。以下是我们对新底层的理解及对底层所作的工作。

二、底层代码结构介绍

- librcsc v4.1.0　　Agent2D及相关工具的底层库
- soccerwindow2　RCSS浏览程序，既可作为monitor，也可作为logplayer
- fedit2　　球队阵型编辑器
- agent2d v3.1.0　官方给出的最原始的底层，下载地址为

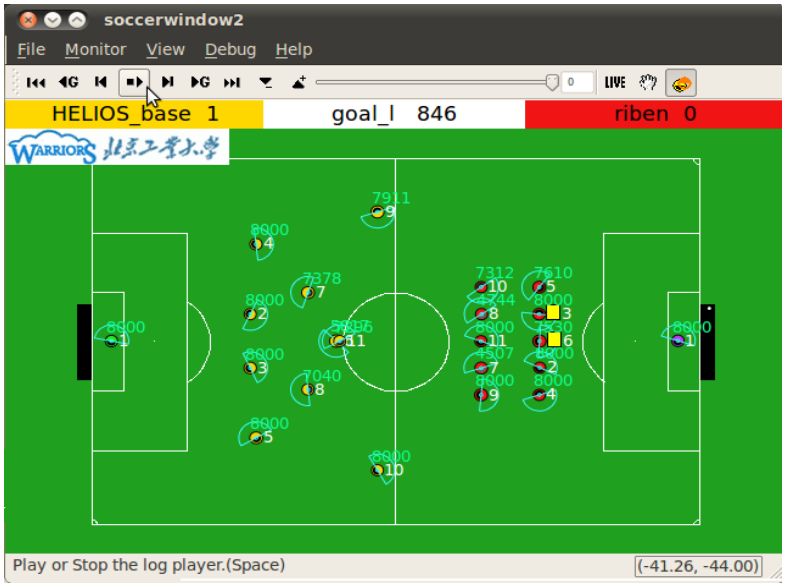
http://fr.sourceforge.jp/projects/rctools/downloads/51943/agent2d-3.1.0.tar.gz/

2.1 librcsc

librcsc 作为一个球队及相关工具的底层库，包含了几何运算库、网络连接库、球员的基本动作库、世界模型库、教练球员相关的库以及日志分析等库。

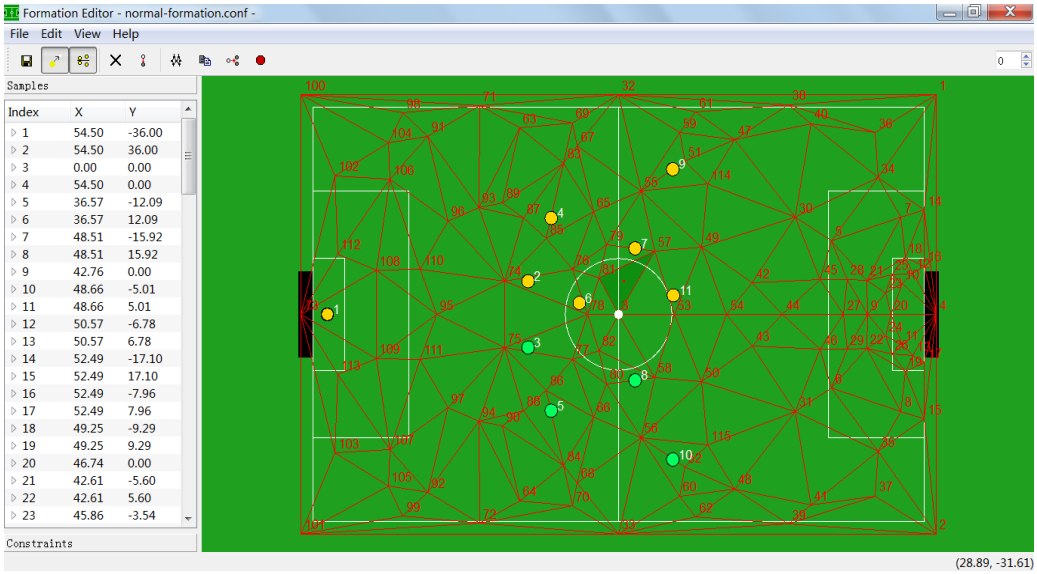
2.2 soccerwindow

使用 soccerwindow 调试工具，可以在 linux 平台下在线调试球队。以下图中的黄方 9 号球员为例，数字 9 是其号码，数字 7911 代笔其当前体力，半圆弧线代表其当前脖子方向（视野）。可以通过启动 Debug 菜单可以查看任意球员任意仿真周期的动作和状态。通过拖动工具栏的进度条，可以回放比赛。右下方显示当前鼠标位置的坐标。还可以在比赛中任意摆放球员的位置，观察效果。此外，调试工具还可以打开任意一场比赛的.rcg 文件，进行数据与策略分析。如下图所示：



2.3 fedit

fedit是一个在Windows下运行的阵型编辑器，可以直观的方便的设置自己的阵型，避免了在程序中写大量代码的麻烦，同时也不用一遍遍的测试坐标。通过改变阵型文件，我们可以对比赛当中的某个特殊的场景设定专门的阵型。如下图所示：



2.4 agent2d

agent2D 使用 librcsc 作为底层库，实现了一个比较简单的高层决策的模板。通过球员角色的分类和各自 Body 的动作决策，再配合球场区域的划分，完成整个决策的过程。

从main_player文件启动进入程序后，由sample_player执行一系列初始化工作，然后关联至PlayerAgent对像，调用Strategy文件进行角色的划分。最后根据当前的世界模型，不同角色的球员进入不同的函数。

以 role_XXX 为开头命名的文件定义着场上各个位置的球员角色（前锋、边卫、中后场及守门员等），bhv_XXX 文件定义球员和守门员在不同场景下的一些宏观动作（微观动作位于 chain_action 文件夹中），neck_XXX 文件定义了球员视角。在 strategy 类中，通过一个 role factory 数组结构生成和转换各个球员的角色类型，通过 getFormation 函数在不同的场景

下读取不同的阵型文件来调整场上各个球员的位置。

strategy.h和strategy.cpp文件的主要函数：

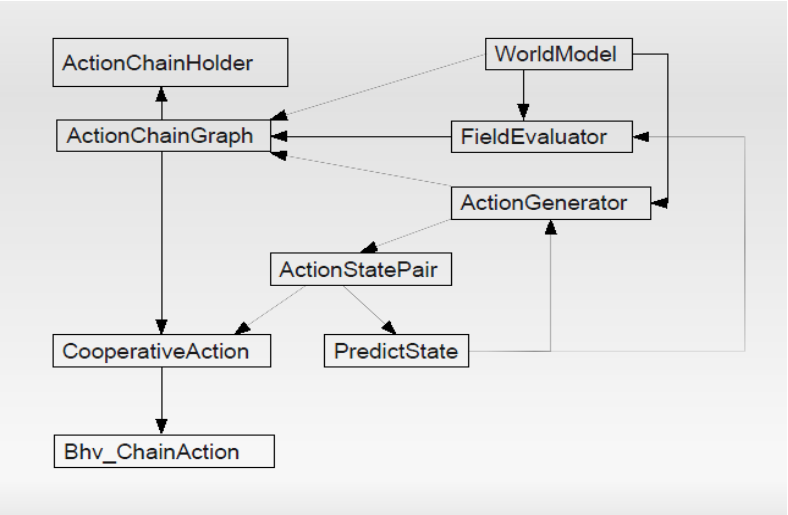
bool init(rcsc::CmdLineParser & cmd_parser); //初始化
bool read(const std::string & config_dir); //读阵型，调用readFormation等实现

void updateSituation(WorldModel & wm); //更新感知
void updatePosition(WorldModel & wm);//更新位置

BallArea get_ball_area(WorldModel & wm);//由世界模型获得球所在的区域
BallArea get_ball_area(rcsc::Vector2D & ball_pos);//由球的位置获得所在的区域

double get_normal_dash_power(WorldModel & wm);//计算带球力量，决定了速度
exchangeRole(const int unum0, const int unum1);//互换角色

本队目前基于比赛要求的agent2d 3.1.0底层进行设计，众所周知，agent2d 3.1.0版本与前代的最大区别就在于引入了动作链机制，通过预测接下来几个周期可能的情况来产生一系列状态。该机制执行所用的文件类都在chain_action文件夹中。在不同的场景下产生不同的战术意图（Intention），通过评估器打分评估 ，选择最优的动作链基本动作（Action_Generator）执行。如下图所示：



图中的核心部分是FieldEvaluator（场区评估器）和ActionGenerator（动作产生器），将上述两个模块产生的评估值和动作将送至ActionChainGraph去生成动作链图。CooperativeAction调用ActionChainGraph结合ActionStatePair去进一步产生动作的策略。最后，由Bhv_ChainAction产生具体的动作。

Action_Generator类

动作产生器，WorldModel和PredictState作为输入，输出为ActionStatePair，派生类为ActGen_XXX，可以通过继承自己来编写动作发生器。

CooperativeAction 类

抽象类，负责定义动作类型、起始位置、目标位置和球的初始位置。

PredictState 类

定义执行动作后的预测状态。

ActionStatePair 类

CooperativeAction 及其执行后的 PredictState 配对。

ActionChainGraph 类

负责执行搜索树的生成与评价。

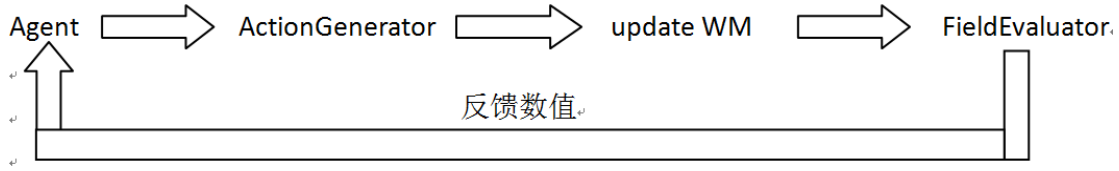
Bhv_ChainAction 类

生成一个 CooperativeAction 的命令。

Field_Evaluator 类

评估器，当使用不同的评价函数时，这个函数的返回值将改变。

Agent2d-3.1.0中的动作决策框架，如下图所示：



Agent 通过 ActionGenerator 选择需要执行的动作（shoot\pass\cross\dribble\selfpass），动作执行后 WorldModel(世界模型)发生改变,作用于 Field Evaluator,Field Evaluator 反馈一个 reward(回报)给 Agent。这是一个动作模拟执行的过程，5 种动作模拟执行后都会返回 reward，Agent 执行反馈数值最大的那个动作。

三、我们所做的主要工作与进展

3.1 场中球员角色的决策调整

底层把球队角色分成了8个，对应着场上各个位置的球员，但是每类球员（除了守门员）的策略基本是最原始的。基本都是判断自己是否持球，然后再做出移动还是传球的决策。而底层中只有有球策略，无球策略基本没有，比赛中经常出现跑位不准确的情况。我们对几类球员类型补充了一些的策略，在doMove()和doKick()中添加不同case下执行不同的behavior，改善了策略的单调空白。

进攻上主要是以下几点：

（1）前锋队员有球时，计算是否可射门，若可射门则射门，若不能射门，则传给附近队友，若附近没有可传球，则以相应的模式带球；

（2）后卫球员持球时，若在危险区，尽快传球，若没有符合的传到点，快速带球出危险区，若不在危险区域，传球或带球。

（3）加强了边路突破时中路队员的后排插上，并取得了一些实战效果。丰富球队的进攻策略，保证边路突破的同时，强调中路的渗透，记后腰（6、7、8号）、中锋11号球员之间的配合。当出现比较好的机会时，球员可以选择中路直接打门。这样可以分担边路球员的进攻压力，保障其体能。

（4）强调中路进攻，设立中锋和影子前锋角色。中锋在对方防守队员紧逼情况下，优先选择自己突破和进攻。在前场，影子前锋负责中场的调度和边路球员传中的接应，以缓解中锋的前场压力。在有较好射门机会的情况下，影子前锋优先选择射门，而不是传球给中锋。在后场，影子前锋更多得要参与球队中场的拦截。

防守上我们根据底层中的分区（在底层中已经分好区但是并没有使用），重点防守BA_DribbleBlockArea和BA_MinDefArea两个区域。而且我们修改防守的主要角色是side_back,center_back和defensive_half三个角色。我们修改的主要方针是让边路防守球员离球更近，然后根据自己、要防守的球员和球的具体情况采取不同的方法，让对方球员不

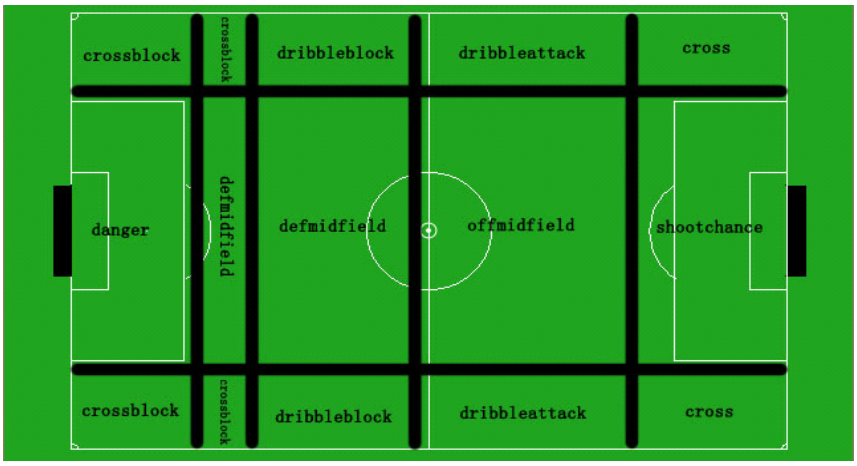
至于一路突到底线。在doMove()里面针对区域写不同的函数。

3.2 门将策略调整

通过对 agent2d 3.1.0 底层的研究以及一段时间的实际测试，我们发现，底层对于守门员站位的设置存在着一些不合理的地方。最突出的表现在于经常会出现边锋在底线附近小角度打近门柱破门这种进球方式。为了解决这一问题，我们微调了守门员的站位策略，包括在 Bhv_GoalieBasicMove 文件中，将守门员的初始位置往后挪，使得守门员的站位更加合理，防守效果更好。

3.3 球场区域的划分

Agent2d-3.1.0 场区划分如下图所示：



我们保留了底层将球场划分成的十个区域数量。agent2d底层进攻上大都由边路发起，边锋从侧路带球突破，下底后选择传中。为了增加中路的进攻，我们调整了区域的大小，扩大了边路区域的范围，以配合我们策略的执行。

同时，我们的球队坚持底层的优势强化了边路突破和禁区附近的防守。根据设计的阵型和跑位，横向扩大了dirbbleattack区域，这样边路球员有更多的空间发挥速度和突破技巧的优势。将danger区域扩大到大禁区外2m，当皮球进入该区域，我方将执行积极的防守策略。此外，我们改进了shootchance区域的范围，缩小到大禁区内2m处，这样可以在一定程度上减少远距离盲目射门的次数，从而提高射门命中率。

3.4 阵型的修改

在 agent2d 中，阵型文件是策略的重要组成部分。特别是 normal-formation、offense-formation、defense-formation 三个阵型文件，分别对应着中场、进攻、防守的策略。在对底层的攻防策略以及基本阵型进行研究过后，我们发觉其中存在部分需要强化的地方。我们针对边路的防守以及进攻中中场队员的跑动路径进行了微调，增减其中的一些节点，使球员在比赛时有更好的站位，加强了攻防的合理性。

A. 修改了before-kick-off开场阵型。

B. 为了增强本队的防守能力，大幅修改了defense-formation阵型，使得防守场景下的本方防线更靠后，回到后场投入防守的人员更多。

C. 发边线球和角球时，由于indirect-freekick-our-formation站位不合理，导致每次发边线球时都把球传给别人，经过改进后，问题得以解决。

D. 进攻时阵容没有战术配合，通过对Offense-formation的对方门前几个点的站位调整，形成了一个新战术。边锋9号、10号带球下底突破时，中锋11号的快速前插，接应打门。

E. 阵型调配结合强调中路进攻的策略，突出了中锋和影子中锋的作用。阵型中将中锋的位置设置的更加靠前，影子前锋紧跟其后。
F. 角球时设置了战术角球（接球队员接球后立即回传给发角球的队员）和直接角球。任意球防守时，人墙摆放更加严密。
G. 对方发门球时，站位紧贴对方后卫，增加前场抢断的可能性。

3.5 射门动作的改进

Agent2D 的 shoot_generator 文件中将对方球门线从最左到最右划分为 24 等份，这样在球门线上就形成了 25 个射门点，Agent2D 对这 25 个射门点进行复杂的打分评价，选取一个分数最高的点作为射门点。在此基础上我们进行了优化，改进了射门算法，在可以起脚射门时，增加了一步高速带球，使皮球有更大的速度，完成射门动作。与为改动之前的射门动作相比较，中短距离的射门成功率大幅度提高，但是远程射门被扑救和抢断的几率有小幅度上升。

3.6 动作链评估器的改进

我们意识到若要对战术配合进行深入改动必须要从动作链入手，场上球员所做的每一步决策都与动作链评估器有关。我们通过查找资料，参考了有关 1994-2006 年(15-18 届)世界杯全进球的射门区域分析。我们计划将球队的射门区域按照该分析对进球区域进行细化，不同的区域设置不同的权值。本方禁区内的防守也按照对方相应的射门区域进行调整。

我们在 action_chain_graph 文件中尝试修改了动作链评价函数，利用评估器对足球场场区进行了更为细致的重新划分，增加禁区防守人数，提高防守质量，当对方球员进攻到禁区时，本方球员积极抢断，在规则允许的情况下，压缩对方球员带球空间和射门角度。但加入新的复杂评估算法后会导致场上球员的决策行为过程变慢了许多，球队进攻运行不流畅，今后将重点对此进行改进。

四、关于录像

所提交的录像是BJUT队和底层的3000周期对决情况。具体的结果可以在提交的文件中看到。

五、总结与展望

虽然我们对agent2d进行了修改，但是我们发现BJUT队还是有很多缺陷。首先中路防守比较薄弱，并且当对方快速把球传向前场的的时候我方球员往往不能快速的反应，给对方提供了过多单刀的机会。其次进攻方面还不够强大，进攻配合战术不够丰富，传球效果还不够理想，与强队较量时容易被截获。我们未来将要研究和改进的地方还有很多。

我们希望今后尝试实现基于 Q 算法的射门，改进射门点的选择。就守门员的站位，我们计划实现守门员站在当前皮球的位置与两侧立柱所成的夹角的角平分线上，这样守门员可以以最短的平均距离扑到皮球。我们查阅资料，参考了首先由清华大学队引入的协防策略和 heilos-11 队伍中的协防的实现，计划加强后场的阻截和抢断。

我们还计划将对方球门两侧立柱到对方球门小禁区底线附近设置为禁区 passball 区域，如下图所示。由于前锋在此区域射门机会很小，皮球往往被门将没收，我们考虑当前锋带球到该区域时，将寻觅机会将皮球传到球门正前方的队友脚下，从而制造机会，完成射门。

回顾过去，我们也许没有取得骄人的成绩，但是从中我们学习到了很多，例如Linux、C++

和一些智能算法，同时也结交了很多朋友。我们坚信科技的力量，将继续努力钻研仿真机器人足球比赛，为后来的有志者铺好路，为Robocup迈向2050年目标的路上贡献出自己的力量！

参考资料

[1] RoboCup Official Site, http://www.robocup.org/

[2] Hidehisa Akiyama and Hiroki Shimora. HELLOS2011 Team Description. http://www.robocup.org/

[3] WrightEagle, 仿真机器人足球:设计与实现.

[4] RoboCup サッカー2D シミュレーション講習会，秋キャンプ2011，福岡大学 秋山英久