**课程编号：0521290X**                     **课程性质：必修**

**机器人技术大作业报告**



**学院：      计算机与信息学院**

**专业：      计算机科学与技术**

**班级：             17-1班**

**姓名：  王子翔, 叶岩宁, 张硕**

**学号： 2017217713, 2017217712, 2017217711**

**教师：        方宝富, 李俊照**

**2019年 4月 30日   至  2019年 5 月 9日**

**机器人技术大作业**

**1. 实验目标**

仿真 2D 机器人足球利用计算机模拟机器人进行足球比赛，由 RoboCup 仿真平台开发小组提供一个标准比赛软件平台，平台设 计充分体现了控制、通讯、传感和人体机能等方面的实际限制， 使仿真球队程序易于转化为硬件球队的控制软件。由于避免了现 实物理环境和当前机器人制造技术的限制，仿真机器人足球主要 把研究重点放在球队的高级功能的研究上，包括动态不确定环境 中的多主体合作、实时推理 -规划 -决策、机器学习和策略获取等 当前人工智能的热点问题。也正是由于摆脱了硬件限制，仿真组 比赛比较容易实现，成为 RoboCup 比赛中历史最老、参赛队最多 的一个项目，研究步伐也快于其他项目。

根据以前的实验，设计出一支具有一定战术策略的机器人足球队。

**2. 实验设备**

硬件环境：PC机

系统环境：Ubuntu 14.04 LTS

**3. 实验内容**

按照小组的方法完成一支完整的仿真机器人足球队伍。

**4. 实验过程和程序**

/\*

Copyright (c) 2000-2003, Jelle Kok, University of Amsterdam

All rights reserved.

Modified by ming gao Hfut..for server V14

1). dash dirction for not only forward and back but also left right and other 45\*(-3,-1,1,3) dirction,,

2). add Stamina.Capacity and the senceHandle analy systerm and worldmodel update

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this

list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,

this list of conditions and the following disclaimer in the documentation

and/or other materials provided with the distribution.

3. Neither the name of the University of Amsterdam nor the names of its

contributors may be used to endorse or promote products derived from this

software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE

FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*! \file PlayerTeams.cpp

<pre>

<b>File:</b> PlayerTest.cpp

<b>Project:</b> Robocup Soccer Simulation Team: UvA Trilearn

<b>Authors:</b> Jelle Kok

<b>Created:</b> 10/12/2000

<b>Last Revision:</b> $ID$

<b>Contents:</b> This file contains the class definitions for the

Player that are used to test the teams' high level

strategy.

<hr size=2>

<h2><b>Changes</b></h2>

<b>Date</b> <b>Author</b> <b>Comment</b>

10/12/2000 Jelle Kok Initial version created

</pre>

\*/

#include "Player.h"

/\*!This method is the first complete simple team and defines the actions taken

by all the players on the field (excluding the goalie). It is based on the

high-level actions taken by the simple team FC Portugal that it released in

2000. The players do the following:

- if ball is kickable

kick ball to goal (random corner of goal)

- else if i am fastest player to ball

intercept the ball

- else

move to strategic position based on your home position and pos ball \*/

///misol.gao fixed--->

///大家好！很高兴你们找到了这个函数！这个函数，就是我们球队决策的起点，所有的比赛策略都在这里，从这里出发，不会错的！

/// deMeer5 是球员的策略函数！ 在下面你们会发现 deMeer5\_goalie 那个是守门员的，目前基本可以忽略。

SoccerCommand Player::deMeer5( )

{

SoccerCommand soc(CMD\_ILLEGAL);// 定义一个命令对象，后面会频繁用它来构造一个球员动作命令

VecPosition posAgent = WM->getAgentGlobalPosition();// 一个点对象，表示Agent(可以理解成当前在执行这个代码的球员它自己)的位置

///下面我们把“我“就理解成当前执行这个代码的Agent，因为有11个球员在执行这个代码，所以对于每个球员 肯定有一个“我”存在。

VecPosition posBall = WM->getBallPos();// 球的位置

int iTmp;

if ( WM->isBeforeKickOff( ) ) /// 如果还没有开球 注意！ 这里只是还没有开球的情况！ 要做开球后的决策，向下面继续找另外一个 WM->isBeforeKickOff( )

{

if ( WM->isKickOffUs( ) && WM->getPlayerNumber() == 9 ) // 9 takes kick //就找9号去开球（判断了是不是我们开球，和我"Agent"是不是9号）

{

if ( WM->isBallKickable() )// 当球对于我来说是不是可踢！

{

VecPosition posGoal( PITCH\_LENGTH/2.0,

(-1 + 2\*(WM->getCurrentCycle()%2)) \*

0.4 \* SS->getGoalWidth() );// 计算一个射门点

soc = kickTo( posGoal, SS->getBallSpeedMax() ); // kick maximal 以最大速度把球踢向射门点！

Log.log( 100, "take kick off" );

}

else //如果球对于我不可踢，那我就去抢球！

{

soc = intercept( false );

Log.log( 100, "move to ball to take kick-off" );

}

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

return soc;

}

//下面不用管！

if ( formations->getFormation() != FT\_INITIAL || // not in kickoff formation

posAgent.getDistanceTo( WM->getStrategicPosition() ) > 2.0 )

{

formations->setFormation( FT\_INITIAL ); // go to kick\_off formation

ACT->putCommandInQueue( soc=teleportToPos( WM->getStrategicPosition() ));

}

else // else turn to center

{

ACT->putCommandInQueue( soc=turnBodyToPoint( VecPosition( 0, 0 ), 0 ) );

ACT->putCommandInQueue( alignNeckWithBody( ) );

}

}

else /// 这个else 大家应该可以知道，是开球以后的决策！ 也就是最重要的比赛正常模式下的决策！ 很重要，我们就从这里开始！！！！ Start！

{

formations->setFormation( FT\_433\_OFFENSIVE );//设置球队出场阵形！

soc.commandType = CMD\_ILLEGAL;//初始化soc命令对象

///test SoccerCommand(CMD\_DASH, 80, 45) for v14 ... fixed by misol.gao

if ( WM->getPlayerNumber() == 8 )

{

soc = SoccerCommand(CMD\_DASH, 80, 45);

ACT->putCommandInQueue( soc ); // 放入命令队列

ACT->putCommandInQueue( alignNeckWithBody() );

return soc;

}

// 2 号和 4 号一起去盯防对方拿球队员

if (WM->getAgentObjectType() == OBJECT\_TEAMMATE\_2 ||

WM->getAgentObjectType() == OBJECT\_TEAMMATE\_4)

{

ObjectT opponent = WM->getClosestInSetTo(OBJECT\_SET\_OPPONENTS, OBJECT\_BALL);

VecPosition opponentPos = WM->getGlobalPosition(opponent);

soc = moveToPos(opponentPos, PS->getPlayerWhenToTurnAngle());

ACT->putCommandInQueue(soc);

ACT->putCommandInQueue(turnNeckToObject(opponent, soc));

return soc;

}

// 在 playOn 模式下,9号带球前进,然后 5 号跟着 9 号一起前进

if (WM->getPlayerNumber() == 5)

{

VecPosition teamOpp = WM->getGlobalPosition(OBJECT\_TEAMMATE\_9);

VecPosition targetOpp(teamOpp.getX() + 5, teamOpp.getY() + 5);

soc = moveToPos(teamOpp, 20);

ACT->putCommandInQueue(soc); // move to strategic pos

ACT->putCommandInQueue(turnNeckToObject(OBJECT\_TEAMMATE\_9,soc));

return soc;

}

// 在 playOn 模式下,9 号带球前进,然后 8 号跟着 9 号一起前进

if (WM->getPlayerNumber() == 8)

{

VecPosition teamOpp = WM->getGlobalPosition(OBJECT\_TEAMMATE\_9);

VecPosition targetOpp(teamOpp.getX() + 5, teamOpp.getY() - 5);

soc = moveToPos(teamOpp, 20);

ACT->putCommandInQueue(soc); // move to strategic pos

ACT->putCommandInQueue(turnNeckToObject(OBJECT\_TEAMMATE\_9, soc));

return soc;

}

if ( WM->getConfidence( OBJECT\_BALL ) < PS->getBallConfThr() )//判断对球的可信度，如果小于某个阈值，则...也就是说，如果很多周期没有看到球在哪里了

{

ACT->putCommandInQueue( soc = searchBall() ); // if ball pos unknown //执行找球动作！并放入命令队列

ACT->putCommandInQueue( alignNeckWithBody( ) ); // search for it //同时把脖子随身体一起转

}

else if ( WM->isBallKickable()) // if kickable // 如果球已知，而且当前球在我脚下(可踢)

{

// 到对方禁区后以最大速度射向空隙大的球门一侧。

VecPosition pos = WM->getBallPos();

if (WM->isInTheirPenaltyArea(pos)) {

double posGoalieY = WM->getGlobalPosition(WM->getOppGoalieType()).getY();

if (posGoalieY > 0) {

soc = kickTo(VecPosition(52.5, -6.5), 2.7);

}

else {

soc = kickTo(VecPosition(52.5, 6.5), 2.7);

}

Log.log(100, "kick ball");

}

else { //未到禁区

// 有人逼抢,传球给最近的队友;否则向球门方向快速/慢速带球。

// 5, 8在9的前方, 因此多数情况下589会成为一个带球的团体, 不设置把球传给589也可以达到类似效果

ObjectT opponent = WM->getClosestRelativeInSet(OBJECT\_SET\_OPPONENTS);

if (opponent != OBJECT\_ILLEGAL && WM->getAgentGlobalPosition().getDistanceTo(WM->getGlobalPosition(opponent)) < 7) {

ObjectT mate = WM->getClosestRelativeInSet(OBJECT\_SET\_TEAMMATES);

soc = leadingPass(mate, 1);

}

else {

// 如果在本方半场,则朝前方快速带球,如果在对方半场,则朝球门慢速带球。

if (WM->getBallPos().get() < 0) {

VecPosition Goal = WM->getPosOpponentGoal();

AngDeg ang = (Goal - posAgent).getDirection();

soc = dribble(ang, DRIBBLE\_FAST);

}

else {

AngDeg ang = 0.0;

soc = dribble(ang, DRIBBLE\_SLOW);

}

}

}

ACT->putCommandInQueue(soc); // 放入命令队列

ACT->putCommandInQueue(turnNeckToObject(OBJECT\_BALL, soc)); // 把脖子转向球，也就是一直看着球

}

else if ( WM->getFastestInSetTo( OBJECT\_SET\_TEAMMATES, OBJECT\_BALL, &iTmp )

== WM->getAgentObjectType() && !WM->isDeadBallThem() ) // 如果球不在我的控制范围下，但是当前能最快抢到球的是我，那我就去执行抢球动作

{ // if fastest to ball

Log.log( 100, "I am fastest to ball; can get there in %d cycles", iTmp );

soc = intercept( false ); // intercept the ball

if ( soc.commandType == CMD\_DASH && // if stamina low // 这里是对体力的一个保护，体力过低就把Dash的dPower减小 保护体力

WM->getAgentStamina().getStamina() <

SS->getRecoverDecThr()\*SS->getStaminaMax()+200 )

{

soc.dPower = 30.0 \* WM->getAgentStamina().getRecovery(); // dash slow

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else // if stamina high

{

ACT->putCommandInQueue( soc ); // dash as intended

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

}

else if ( posAgent.getDistanceTo(WM->getStrategicPosition()) >

1.5 + fabs(posAgent.getX()-posBall.getX())/10.0) // 到了这里就是其他距离球相对远一点的人了，如果离自己的阵形点太远，就跑回 自己的阵形点去。

// if not near strategic pos

{

if ( WM->getAgentStamina().getStamina() > // if stamina high

SS->getRecoverDecThr()\*SS->getStaminaMax()+800 )

{

soc = moveToPos(WM->getStrategicPosition(),

PS->getPlayerWhenToTurnAngle());

ACT->putCommandInQueue( soc ); // move to strategic pos

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else // else watch ball

{

ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT\_BALL ) );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

}

else if ( fabs( WM->getRelativeAngle( OBJECT\_BALL ) ) > 1.0 ) // watch ball //其他剩下的球员呢，就看球。！

{///这里就是无球队员的跑位决策

ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT\_BALL ) );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else // nothing to do

ACT->putCommandInQueue( SoccerCommand(CMD\_TURNNECK,0.0) );

}

return soc;

}

/\*!This method is a simple goalie based on the goalie of the simple Team of

FC Portugal. It defines a rectangle in its penalty area and moves to the

position on this rectangle where the ball intersects if you make a line

between the ball position and the center of the goal. If the ball can

be intercepted in the own penalty area the ball is intercepted and catched.

\*/

SoccerCommand Player::deMeer5\_goalie( )

{

int i;

SoccerCommand soc;

VecPosition posAgent = WM->getAgentGlobalPosition();

AngDeg angBody = WM->getAgentGlobalBodyAngle();

// define the top and bottom position of a rectangle in which keeper moves

static const VecPosition posLeftTop( -PITCH\_LENGTH/2.0 +

0.7\*PENALTY\_AREA\_LENGTH, -PENALTY\_AREA\_WIDTH/4.0 );

static const VecPosition posRightTop( -PITCH\_LENGTH/2.0 +

0.7\*PENALTY\_AREA\_LENGTH, +PENALTY\_AREA\_WIDTH/4.0 );

// define the borders of this rectangle using the two points.

static Line lineFront = Line::makeLineFromTwoPoints(posLeftTop,posRightTop);

static Line lineLeft = Line::makeLineFromTwoPoints(

VecPosition( -50.0, posLeftTop.getY()), posLeftTop );

static Line lineRight = Line::makeLineFromTwoPoints(

VecPosition( -50.0, posRightTop.getY()),posRightTop );

if ( WM->isBeforeKickOff( ) )

{

if ( formations->getFormation() != FT\_INITIAL || // not in kickoff formation

posAgent.getDistanceTo( WM->getStrategicPosition() ) > 2.0 )

{

formations->setFormation( FT\_INITIAL ); // go to kick\_off formation

ACT->putCommandInQueue( soc=teleportToPos(WM->getStrategicPosition()) );

}

else // else turn to center

{

ACT->putCommandInQueue( soc = turnBodyToPoint( VecPosition( 0, 0 ), 0 ));

ACT->putCommandInQueue( alignNeckWithBody( ) );

}

return soc;

}

if ( WM->getConfidence( OBJECT\_BALL ) < PS->getBallConfThr() )

{ // confidence ball too low

ACT->putCommandInQueue( searchBall() ); // search ball

ACT->putCommandInQueue( alignNeckWithBody( ) );

}

else if ( WM->getPlayMode() == PM\_PLAY\_ON || WM->isFreeKickThem() ||

WM->isCornerKickThem() )

{

if ( WM->isBallCatchable() )

{

ACT->putCommandInQueue( soc = catchBall() );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else if ( WM->isBallKickable() )

{

soc = kickTo( VecPosition(0,posAgent.getY()\*2.0), 2.0 );

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else if ( WM->isInOwnPenaltyArea( getInterceptionPointBall( &i, true ) ) &&

WM->getFastestInSetTo( OBJECT\_SET\_PLAYERS, OBJECT\_BALL, &i ) ==

WM->getAgentObjectType() )

{

ACT->putCommandInQueue( soc = intercept( true ) );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else

{

// make line between own goal and the ball

VecPosition posMyGoal = ( WM->getSide() == SIDE\_LEFT )

? SoccerTypes::getGlobalPositionFlag(OBJECT\_GOAL\_L, SIDE\_LEFT )

: SoccerTypes::getGlobalPositionFlag(OBJECT\_GOAL\_R, SIDE\_RIGHT);

Line lineBall = Line::makeLineFromTwoPoints( WM->getBallPos(),posMyGoal);

// determine where your front line intersects with the line from ball

VecPosition posIntersect = lineFront.getIntersection( lineBall );

// outside rectangle, use line at side to get intersection

if (posIntersect.isRightOf( posRightTop ) )

posIntersect = lineRight.getIntersection( lineBall );

else if (posIntersect.isLeftOf( posLeftTop ) )

posIntersect = lineLeft.getIntersection( lineBall );

if ( posIntersect.getX() < -49.0 )

posIntersect.setX( -49.0 );

// and move to this position

if ( posIntersect.getDistanceTo( WM->getAgentGlobalPosition() ) > 0.5 )

{

soc = moveToPos( posIntersect, PS->getPlayerWhenToTurnAngle() );

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else

{

ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT\_BALL ) );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

}

}

else if ( WM->isFreeKickUs() == true || WM->isGoalKickUs() == true )

{

if ( WM->isBallKickable() )

{

if ( WM->getTimeSinceLastCatch() == 25 && WM->isFreeKickUs() )

{

// move to position with lesser opponents.

if ( WM->getNrInSetInCircle( OBJECT\_SET\_OPPONENTS,

Circle(posRightTop, 15.0 )) <

WM->getNrInSetInCircle( OBJECT\_SET\_OPPONENTS,

Circle(posLeftTop, 15.0 )) )

soc.makeCommand( CMD\_MOVE,posRightTop.getX(),posRightTop.getY(),0.0);

else

soc.makeCommand( CMD\_MOVE,posLeftTop.getX(), posLeftTop.getY(), 0.0);

ACT->putCommandInQueue( soc );

}

else if ( WM->getTimeSinceLastCatch() > 28 )

{

soc = kickTo( VecPosition(0,posAgent.getY()\*2.0), 2.0 );

ACT->putCommandInQueue( soc );

}

else if ( WM->getTimeSinceLastCatch() < 25 )

{

VecPosition posSide( 0.0, posAgent.getY() );

if ( fabs( (posSide - posAgent).getDirection() - angBody) > 10 )

{

soc = turnBodyToPoint( posSide );

ACT->putCommandInQueue( soc );

}

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

}

else if ( WM->isGoalKickUs() )

{

ACT->putCommandInQueue( soc = intercept( true ) );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

else

{

ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT\_BALL ) );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

return soc;

}

**5. 总结和心得**

本实验是机器人大作业, 也是机器人技术课程系列实验的最后一次实验. 这次的实验是要求设计出一支具有一定战略策略的机器人足球队. 即通过修改PlayTeams.cpp得到一个富有战术的球队. 通过前几次的实验课程，我们小组的成员了解了有关机器人技术的基础知识。并通过对于之前的基础技术的了解, 在本次实验中，我们才得以设计一个完整的球队策略.

**我们组在进行了讨论之后, 决定在源代码的基础上加入部分规则, 让机器人更具有战斗力. 我们决定加入如下规则:**

1. 默认9号带球前进, 同时5号8号球员在9号不远处跟随.
2. 在己方半场快速带球, 在对方半场慢速带球.
3. 有人逼抢, 传球给最近的球员, 否则朝前方带球.
4. 到禁区后射门, 并以最大速度射向空隙大的一侧射门.
5. 2, 4号球员夹防带球球员.

**我们组在这次试验的的分工如下: 我们都参与了制定规则的讨论. 王子翔(2017217713)负责对代码的整体架构, 逻辑制定和程序编写; 叶岩宁(2017217712) 参与了部分程序的编写和代码的鲁棒性测试; 张硕(2017217711) 负责对代码进行逻辑调试和校验. 我们三人齐心协力共同完成了本次实验.**

**我们加入的规则能够让多名机器人进行群体带球, 在遇到逼抢时即时向后面的跟随球员传球, 防止丢球. 通过这种方式能够显著的过半场改善丢球状况. 己方半场快速带球, 对方半场慢速带球能够更快的抵达对方半场, 节省体力.禁区射门射向空隙大的一边能够避免被守门员扑出. 同时多人带球的规则能够有效造成二次进攻. 同时己方半场2, 4号球员夹防带球队员, 给对方进攻制造威胁.**

**本次实验虽然难度较大, 但是通过讨论和查找资料, 我们成功的克服了困难, 完成了本次实验. 这次实验的完成过程加深了我们对所学知识的迁移应用, 同时也加深了我们对于源程序的理解, 让我们能够更好的运用所学知识.**

**希望我们设计出的机器人球队能够在此次比赛中大放异彩.**