文档里可能有错别字！！！

**实验2 Demeer5基本动作1**

**1，实验目的**

（1）了解Demeer5的工作原理

（2）学会对Demeer5进行简单的修改

**2，实验设备**

硬件环境：pc。

软件环境：Linux。

**3，实验内容**

(1)Demeer5的工作原理

Demeer5函数是整个球的核心，它最终返回一个可执行的动作，底层的模块负责将此动作发送给server,然后由Server执行。可以说,Demeer5就是我们想法的体现，是一支球队的大脑。在Demeer5中有一系列的判断来决定每个周期的动作。下面对

Demeer5进行简要的分析。

Demeer5是一个决策函数，在策略上使用的是下面这个简单的策略：

．如果球可踢，则用最大力量踢球；

．如果球不可踢且我是队友中最快到达球的队员，则去截球；

．其他情况按战略点跑位。

我们现在只要看一下球可踢时的代码：

Else if(WM—>isBaIlKickable()）

{

//如果球可踢

//确定踢向的点

VecPositionposGoaI(PITCH\_LENGTH/2.0，

（-1+2\*(WM—>getCurrentCycle()％2）)\*0·4\*SS—>getGoalWidth()）；

//调用踢球的动作

soc=kickTo(posGoaI，SS—>getBallSpeedMax()）；//kick maximal

//将动作放人命令队列中

ACT—>putCommandInQueue(soc）；

//将脖子转向球

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

//记录调试信息

Logelog(100，"kickball"）；

}

这一小段函数决定了当球在球员智能体的可踢范围之内时应当做的动作，这里是一

个简单地把球向前踢而不考虑任何其他情况的方法。该程序段的条理是很清晰的。

（2）对Demeer5进行简单的修改

现在我们对Demeer5进行简单的修改，让它在球可踢的时候进行带球的动作。带球

就是kick和dash动作序列的结合。带球的函数在BasicPlayer中，函数为dribble()·它

接收两个参数，第一个参数为带球的方向，第二个参数为带球的类型。

带球类型解释如下：

．DRIBBLE\_FAST快速带球；

．DRIBBLE\_SLOW慢速带球；

．DRIBBLE\_WITHBALL安全带球。

所以，对dribble的一种调用形式为：dribble(ang,DRIBBLE\_FAST）

其中ang为AngDeg(是一个double)类型。

该函数的返回值是一个类型。

知道了如何调用dribble，我们来对Demeer5进行修改：

else if(WM—>isBallKickable()）

{

AngDegang=0.0；

soc=dribble(ang，DRIBBLE-FAST）；//进行带球

//将动作放人命令队列中

ACT—>putCommandlnQueue(soc）；

//将脖子转向球

ACT—>putCommandInQueue(turnNeckToObject(OBJECTBALL，soc））；

//记录调试信息

Log.log(100，"kickball"）；

}

这样，在球可踢的时候，球员智能体将把球带向前方，这里取的是'0度角，即沿x轴

一直向前快速带球。

我们再次对Demeer5函数进行修改，这次是让球员Agent将球踢向各个不同的地方。这将调用kickTo（）函数来完成。下面简要说明一下kickTo()函数的使用方法：这个函数有两个参数，第一个参数是目标点的坐标，第二个参数是球到达目标点时的速度，返回一个踢球的动作。可以使用下面的形式来调用：

VecPosition pos(x,y）；

Double speed=1.0；

kickTo（pos，speed）；

kickTo()函数在其内部将会决定踢球时所用力量的大小，并且会判断是否能够将球踢到该点，并作出相应的调整。比如，将球踢向（0，0）点，1·0的末速度：

else if(WM—>isBallKickabIe()）

{

kickTo(VecPosition(0，0），1.0）；

//将动作放人命令队列中

ACT—>putCommandInQueue(soc）；

//脖子转向球

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

//记录调试信息

Logelog(100，"kickball"）；

}

根据上述操作，完成以下踢球操作：

．将球踢向对方的球门。

．将球踢向距离自己最近的队友。

．尝试不同的踢球点。

以下是一些可能用到的函数：

VecPosition getPosOpponentGoal(）；//得到对方球门的坐标

调用：WM一>getPosOpponentGoaI（）；

ObjectT getClosestInSetTo(）；//得到距离自己最近的队员

调用：WM-> getClosestInSetTo(OBJECT\_SET\_TEAMMATES，posAgent）

VecPosition getGlobaIPosition(ObjectT o）//得到一个对象的坐标

调用：WM一>getGloba1Position(o）；

更多的函数请查找教材，或者查看源程序中的WorldModel.CPP，WorldModel-

HighLevel.cpp，WorldModeIPredict.CPP等文件。

根据上述内容，完成以下带球的操作：

．用不同的带球模式进行带球，并观察效果，比较异同；

．将球向对方的球门方向带；

．尝试不同的带球组合。

根据上述内容，完成以下综合练习：

带球与踢球的结合：让球员智能体一直向对方球门的方向带球，在进人对方禁区后

以最大力量踢向球门（末速度最大为2·7）。涉及的具体函数请查看教材及源程序。

**实验3 Demeer5基本动作2**

**1,实验目的**

熟悉demeer5并学会demeer5的基本使用方法，具体内容如下：

（1）能理解UVA程序中，原来的demeer5中的内容；

（2）能通过修改中的具体函数内容实现对场上球员的控制；

**2，实验设备**

硕件环境：pc机。

软件环境：操作系统Linux

3．实验内容

（1）在球队程序中找到playerTeams.cpp并打开；

（3）阅读此段程序，并结合Monitor观察球员的具体行为（你将发现可以踢到球的球

员会将球朝球门的方向踢去，而不能踢到球的队员中如果是离球最近的队员就去截球

否者则按阵形跑位）；

（4）修改demmer5函数改变队员的行为具体步骤如下：

1，在demeer5函数中找到

if(WM—>isBallKickable()）//isBallKickable()函数用来判断球是否可踢

{

Vec PositionposGoal(PITCHLENGTH/2.0，

（-1+2\*(WM—>getCurrentCycle()％2））0，4SS—>getGoalWidth()）；

//设定射门位置坐标

soc=kickTo(posGoal，SS—>getBallSpeedMax()）；//朝球门方向将球以最大力度踢出

ACT—>putCommandInQueue(soc）；//只有把命令放人命令队列动作才会执行

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

//做动作的同时改变脖子的方向

}

2，「控球」将此函数修改为

if(WM—>isBallKickable()）

{

soc=kickBaIlCloseToBody(45）；

ACT—>putCommandInQueue(soc）；

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

}

然后编译运行程序，观察球员的行为我们会发现当球可踢时，球员不再朝着球门的

方向踢了，而是将球绕自己身体转动（UVA的这个底层动作经常把球转丢！）

3，【带球」将此函数修改为

if(WM—>isBaIIKickabIe()）

{

soc=dribble(0.0，DRIBBLE\_SLOW)；//其中dribble函数中第一个参数表示带球的方向可以是-180、180之间，不一定是0·0

ACT—>putCommandInQueue(soc

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

}

然后编译运行程序，观察球员的行为我们会发现当球可踢时，球员不再朝着球门的

方向踢了，而是朝我们指定的方向执行带球。

4，【传球】将此函数修改为

if(WM—>isBallKickabIe()）

{

soc=leadingPass(OBJECT\_TEAMMATE\_9,1）；

//其中leadingPass中第一个参数表示传球的对象，本实验中我们将球直接传给制定

的球员，不一定是OBJECT\_TEAMMATE\_9

ACT—>putCommandInQueue(soc）；

ACT\_>putcommandInQueUe(turnNeckToObject(OBJECT\_BALL,soc)

}

然后编泽运行程序，观察球员的行为我们会发现当球可踢时，球员不再朝着球门的

方向踢了，而是将球传给我们指定号码的队员

5，[配合]将此函数修改为

if(WM—>isBallKickabIe())

{

if(WM-->getAgentObjectType(）-OBJECT\_TEAMMATE\_9)

soc=dribble(0.0,sDRIBBLE\_SL()W)；//带球

else

leadingPass(OBJECT\_TEAMMATE\_9,1)，//传球

ACT—>putCommandInQueue(soc)

ACT—>putCommandInQueue(turnNeckToObject(OBJECT\_BAt.L，卜“）

}

编译程序，观察球员行为，我们会发现，当9号队员得到球后会朝前方带球，其他队

得到球后会将球传给9号（不管9号是不是越位）。

根据以上描述完成以下实验内容：

1，如果在对方禁区内就射门，否则，如果是7，8，9号队员就朝前带球，其他队员将球传给9号（用WM一>isInTheirPenaltyArea()M一>gctBallPos(））来判球是否在对方禁区）；

2，如果队员的位置在自己半场就将球朝对方球门踢去，否者就朝前方带球（用WM->getBallPos().getX()来得到球的x坐标）；

3，当有人来抢球时（离自己很近），就将球传给离自己最近的队友，舌则就自己带球

調用WM一>getCloscstRelativeInSet函数来得到离自己最近的己方或对方球员，通过posl.getDistanceTo(pos2)来得到两位置之间的距离）。（选作）

第1，2题希望大家仔细思考一下，第．题有兴趣的同学可以思考一下，更多基本动作请查阅BasicPlayer.CPP函数，或其他有关书籍，也希望人家能做出更多的动作

**实验4 复杂的动作决策**

1. **实验目的**

进一步了解demecr5并能熟练的修改dcmccrfi的内容以达到对场上球的制

（1)能理解UVA程序中原来的dernccr5中的全内容'

（2）能通过修改demcer5中的具体数内容实现场上球员的制'

（3）能通过底层动作的简单组合控制场上队员做出一些愆杂动作决策'

（4）对WorldModcl有初步的认识，学会在WorldModel,basicplayer里加函数

**2，实验设备**

硬件环境：pc机。

软件环境：操作系统Linux

**3·实验内容**

〈1)在球队程序中找到playerTeams.CPP并打开；

（2）在player.CPP中找到demmer5函数；

（3）阅读此段程序，并结合monitor观察球员的具体行为；

（4）修改demmer5函数改变队员的行为具体步骤如下：

1，在demeer5函数中找到

Else if(WM—>getFastestInSetTo(OBJECT\_SET\_TEAMMATES，OBJECT\_BALL，&iTmp）

==WM—>getAgentObjectType()&&！WM—>isDeadBallThem()）

//如果是最快到达球的队员

Log.log(100，"Iamfastesttoball；cangettherein%dcycles"，iTmp）；

//截球

soc==intercept(false）；

if(soc.commandType=CMD\_DASH&&

WM-->getAgentStamina().getStamina())<

SS—>getRecoverDecThr() && SS—>getStaminaMax()+200）

{

//当体力低时

soc.dPower=30\*WM—>getAgentStamina().ppgetRecovery()；//慢速移动

ACT—>putCommandlnQueue(soc）；

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

//脖子转向球

}

Else

{

ACT—>putCommandlnQueue(soc）；

//当体力高时

//正常移动速度

ACT—>putCommandlnQueue（turnNeckToObject(OBJECT\_BALL，soc））；

//脖子转向球

}

}

此函数的内容是，当球不可踢时，如果是离球最近的队员就执行截球命令（截球函数

intercept（）在BasicP1ayer•CPP中定义）观察在队员体力值小于多少时，带球速度会变慢。

2，现在我们通过修改函数来改变非持球队员的决策，在以上函数之前加上此段

代码：

Else if(WM—>getAgentObjectType()==OBJECT\_TEAMMATE\_9）//如果是9号认员

{

soc—SoccerCommand(CMD\_TURN，60）；//转身体

ACT—>putCommandInQueue(soc）；

ACT—>putCommandInQueue(alignNeckWithBody(））；//脖子随着身体一起转动

}

然后编译运行程序'观察球员的行为我们会发现当9号不持球时'身体在一直的转

动（此动作可用来找球）

（5）在WorldMoled里添加状态函数

1，打开WorldModel.h，在里面预定义函数，即写人

bool isOpponentAtAngleEx(AngDegangA，AngDegangB，doubledDist)，该函数用来判断前球员角度在angA、angB之间距离小于的范围内是否有对方队员

2，找到并打开WorldModel.cpp在里面添加一个新函数

Bool WorldModeI::isOpponentAtAngleEx(AngDegangA,AngDegangBdouble,dDist）

{

VecPosition posAgent = getAgentGIobalPosition(）；

VecPosition posOpp；

AngDeg angOpp；

Int index；

for(ObjectT o：iterateObjectStart(ilndex，OBJECT\_SET\_OPPONENTS)

o!=OBJECT\_ILLEGAL；

o=iterateObjectNext（ilndex，OBJECT\_SET\_OPPONENTS））

{

posOpp=getGlobalPosition(o）；

angOpp=（POSOPP-posAgent）.getDirection()；

if(angA<=angOpp&&angOpp<：angB&&posAgent•getDistanceTo（POSOPP）<dDist）

return true；

iterateObjectDone(ilndex）；

return false；

}

3，将if（WM一>isBaIlKickable()）内的内容修改为：

if（WM—>isBalIKickable(））

{

double ang=（VecPosition(52.5，0）,-posAgent)·getDirection(）；

if（WM->isOpponentAtAngleEx(ang-45，ang，6））

ang+=45;

else if（WM—>OpponentAtAngleEx(ang,ang+45，6））

ang-=45;

SoccerCommand soc=dribble（ang，DRIBBLE\_SLOW)；

ACT—>putcornmandlnQueUe(soc）；

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

}

然后编译运行程序，观察球员的行为，试分析球员的行为。

根据以上描述完成练习：通过基本动作的组合实现球员的以下行为:

1,判断守门员的位置，朝球门空隙较大的一方射门。（通过在worldmodel里建立新状态来判断，哪一方空隙较大，守门员的位置为VecPosition posGoaile=WM->getGlobalPosition(WM->getOppGoalieType())；球门位置为（52,5,0）可尝试朝（52.5,6.5）和(52.5,-6.5)两点射门）

1. 在BasicPlay里添加一个带球函数，要求如果无人阻挡（带球将要经过多路线附近没有对方球员）就朝球门方向带球，否则想办法避开对方球员带球前进（要求只要作出闪避的动作即可，不要求效果）
2. 尝试修改视觉函数使得球员使得球员能更多的获得场上信息（要求不影响球员动作）。

**实验5 特殊比赛模式的设计**

1.实验目的

（1）掌握仿真机器人足球比赛中特殊比赛模式发生的条件；

（2）掌握仿真机器人足球比赛特殊比赛模式的规则要求，

（3）了解仿真机器人足球比赛特殊比赛模式的战术设计思想；

（4）进一步熟悉WorldModel类。

2．实验设备

硬件环境：PC机。

软件环境：操作系统Linux。

&实验内容

（1)概述

Robocup仿真机器人足球比赛特殊比赛模式包括角球（corner\_kick）、界外球（kick\_

In)、定位球/任意球（free\_kick)以及球门球(goal\_kick)。

（2）角球(corner—kick)

当防守方球员将球踢出底线时，由进攻方开角球。Server一旦接收到发球队员发出

的kick命令后，就将比赛模式设为正常的play\_on模式。

注意：与国际足联的规则类似，发球队员在其他球员接触球之前不能再触球，否

则判犯规。此时，另一方球员须在一定时间（根据Server手册的规定为300周期）内将球开出，否则算发球失误，由对方发定位球。

在UVA-Base-2003的原代码中，球员是不管这些特殊比赛模式的，统统是最近的球

员跑向球，一脚踢向球门。这就需要我们手工编码完成这些细节地方。一种简单的设计思路是：．如果比赛进人角球模式，则离球最近的球员A跑向球，而次近的球员B跑到某个接应点，等待A将球传过来；．A跑到球跟前（即进人可踢范围）时，不必立即将球开出，可以先看看场上环境，等B跑到预定位置并且体力恢复得差不多时再开球。

以上设计思路同学们在实验过程中只须完成．即可，对于学有余力的同学可以考l虑的实现。下面列出几个可能用到的函数：

playModeT WorldModel：：getPlayMode(）const

//返回比赛模式（或者直接用bool WorldModeI：：isCornerKickUs()判断是不是我方开角球）

ObjectT WorldModel：：getCIosestInSetTo（ ObjectSetT objectSet，ObjectT o，double· \*dDist=NULL，double dConfThr=-1.0）

//返回在对象集合objectSet中距离对象o最近的对象，只有当对象的可信度高于给定的值才被考虑，如果没有给出阈值则使用PlayerSettings中定义的阈值，同时dDist返回距离

ObjectT WorldModel：：getSecondClosestInSetTo(ObjextSetT objectSet.ObjectT o，double dDist=NULL，double dConfThr=-1.0)

//返回在对象集合objectSet中距离对象o次近的对象，只有当对象的可信度高于给定的阈值才被考虑，如果没有给出國值则使用Playersettings中定义的阈值，同时dDist返回距离

下面具体描述一下实现过程：

首先，我们知道demeer5()主要是围绕三句话展开的，即

1如果球可踢，则用最大力踢球；

2，如果球不可踢且我是队友中最快到球的队员，则去截球；3其他队友则按战略点跑位。

我们可以围绕这三句话来实现角球策略。即在角球模式下

1. 如果球可踢，则传球给接应球员；2，
2. 如果球不可踢且我是队友中最快到球的队员，则去发球；3，
3. 其他情况下，如果我是离球次近的队员，则跑到预定位置接应。

接着，有了这个基本思想后，我们开始编写代码。在demeer5()中找到

Else if（WM—>isBalIKickabIe()）{ //ifkickable

在里面加入我们的角球代码：

If（WM—>isCornerKickUs()）{

ACT—>putCommandInQueue(kickTo(pointToKickTo（），SS—>getBallSpeedMax()\*0.8））；

}

这一部分代码是完成开球动作。函数pointToKickTo()是player类的成员函数，返

一个类型的参数，代表要将球踢向的坐标。下面给出一段参考代码：

VecPosition Player：：pointToKickTo() {

VecPositionPOS，temp；

Double x，y；

pos=WM—>getBallPos()；

x=pos.getX()；

Y=pos.getY0；

temp.setX(—x/fabs(x)\*5+x)；

temp.setY(一y/fabs(y)\*12+y）；

return temp；

}

再在demeer5()中找到

Else if(posAgent.getDistanceTo(WM->getStrategicPosition())>1.5+fabs(posAgent,

getX()-posBall.getX()/10.0) //if not nearstrategic POS

POsBall.getX()）/10，0）

在里面加入如下代码：

If (WM—>isCornerKickUs()）{

ObjectT o

SoccerCommand sctemp;

o=WM->getSecondCloseInSetTo(OBJECT\_SET\_TEAMMATES,OBJECT\_BALL);

If(o=WM->getAgentObjectType()){

sctemp=ACT一>putCommandInQueue（moveToPos（pointToKickTo0，PS一>getPlayWhenToTurnAngle()）；

}

}

ACT=putCommandInQueue(turnNeckToObject(OBJECT\_BALL，sctemp));

}

这段代码是实现接应球员的跑位。

这样，一个简单的角球策略就实现了。同学们可以编译运行看看效果。

（3）界外球(kick\_in)

当一方球员将球踢出边线时，比赛即进人界外球模式。比赛的规则要求基本同“角

球"模式，简单的设计思路和实验内容与角球也基本相同，在此不再赘述。

要求：要求同学们编写一段程序实现界外球策略，要求能够根据发球点的位置给出

合适的接应球员的接应位置。

可能用到的函数：

Bool WorldModel：：isKickInUs()//返回是否是我方开界外球

（4）定位球/任意球(free\_kick)

当一方球员犯规或违例时，由对方开任意球。与国际足联的规定略有不同的是，

Robocup的任意球没有直接任意球跟间接任意球之分。比赛的规则要求基本同界外球。

简单的设计思路和实验内容与界外球也差不多，在此不再赘述。略有不同的是，如

果条件允许，开球队员可以选择直接射门。另外，有能力的同学可以在这里考虑实现开

球队员等待一段时间再开球的策略（提示，要用到锁定机制）。

要求：要求完成一种任意球的开球动作。

（5）球门球(goal—kick)（选做）

当对方进攻球员将球踢出底线或我方守门员截住对方射门的球时，由我方守门员开

球门球。注意，如果守门员截住的是己方后卫的回传球，那么会由对方球员在离截球点

最近的禁区角点发任意球，因为这是违例行为。

如果守门员在300周期内不能将球开出，那么由对方球员在离开球点最近的禁区角

点发任意球。

在UVA-trilearn的源代码中，守门员开球也是直接将球踢向对方球门。而开门球

是一个比较复杂且非常重要的细节，许多强队，诸如清华、科大，都有不错的开门球策略。

我们这次实验只完成一个简单的策略，即守门员沿某一直线开球，该直线是由最近对方

球员、守门员、次近对方球员三点所构成角的角平分线。