**实验二 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 函数是整个球队的核心，它最终返回一个可以执行的动作，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->isBallKickable()) // 如果球可踢

{

//确定踢向的点

VecPosition posGoal( PITCH\_LENGTH/2.0,

(-1+2\*(WM->getCurrentCycle()%2))\* 0.4 \* SS->getGoalWidth());

//调用踢球的动作

soc = kickTo(posGoal, SS->getBallSpeedMax()); // kick maximal

//将动作放入命令队列中

ACT->putCommandInQueue(soc);

//将脖子转向球

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

//记录调试信息

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

}

这一小段函数决定了当球在球员Agent的可踢范围之内时应当做的动作，这里是一个简单的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 SoccerCommand 类型。

知道了如何调用 dribble，我们来对 Demeer5 进行修改：

else if( WM->isBallKickable())

{

AngDeg ang = 0.0;

soc = dribble（ang，DRIBBLE\_FAST ）; // 进行带球

//将动作放入命令队列中

ACT->putCommandInQueue( soc );

//将脖子转向球

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

//记录调试信息

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

}

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是的速度，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，并做出相应的调整。比如，将球一直踢向（ 0， 0）点, 1.0 的末速度：

else if( WM->isBallKickable())

{

soc = kickTo( VecPosition( 0, 0 ),1.0);

//将动作放入命令队列中

ACT->putCommandInQueue( soc );

//将脖子转向球

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

//记录调试信息

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

}

任务：

根据上述操作，完成以下踢球操作：

（1）将球踢向对方的球门：

VecPosition pos=WM->getPosOpponentGoal; soc=kickTo( pos,1.0 ); ACT->putCommandInQueue ( soc );

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

（2）将球踢向距离自己最近的队友：

ObjectT o=WM->getClosestInSetTo( OBJECT\_SET\_TEAMMATES,posAgent);

soc=kickTo( WM->getGlobalPosition( o ) ,1.0 );

ACT->putCommandInQueue ( soc );

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

（3）尝试不同的踢球点。

else if(WM->isBallKickable())

{

//x、y确定球场上不同点 -52.5<=x<=52.5 -34<=y<=34

soc=kickTo(VecPosition(x,y),1.0);

ACT->putCommandInQueue(soc);

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

Log.log(100,”kick ball”);

}

根据上述内容，完成以下带球的操作：

（1）用不同的带球模式进行带球，并观察效果，比较异同。

观察和测试：

SLOW和WITHBALL带球比较慢，FAST显著加快。

当拦截对手从后面追来时，SLOW和WITHBALL护球的效果比较好，而当拦截对手进行正面拦截，FAST可以较快的把球带走，护球成功率高于SLOW和WITHBALL。

AngDeg ang = 0.0;//向正前方带

soc = dribble（ang，DRIBBLE\_SLOW）;//在这里更改慢速、安全、快速带球

ACT->putCommandInQueue( soc );

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

（2）将球向对方的球门方向带。

double ang=( VecPosition(52.5,0)-posAgent).getDirection();//求与球门间角度

soc=dribble(ang,DRIBBLE\_SLOW ); ACT->putCommandInQueue( soc );

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

1. 尝试不同的带球组合。

else if(WM->isBallKickable())

{

AngDeg angDribble = (WM->getPosOpponentGoal()-posAgent).getDirection();

//soc=dribble(angDribble,DRIBBLE\_WITHBALL);

//soc=dribble(angDribble,DRIBBLE\_FAST);

soc=dribble(angDribble,DRIBBLE\_SLOW);

ACT->putCommandInQueue(soc);

}

根据上述内容，完成以下综合练习：

带球与踢球的结合：让 agent 一直向对方球门的方向带球，在进入对方禁区后以最大力量踢向球门( 末速度最大为 2.7 )。涉及到的具体函数请查看教材。

先判断自己是否在禁区，若在禁区则射门，不在则求出球门的方向，向该方向带球。

VecPosition posGoal ( 52.5 , 0 );

if ( WM->isInTheirPenaltyArea ( WM->getBallPos() ) ) //判断是否在对方禁区

{

soc=kickTo ( posGoal , 2.7 );

}

else

{

double ang=( posGoal - posAgent ).getDirection();//求与球门间角度

soc=dribble( ang,DRIBBLE\_FAST );

}

ACT->putCommandInQueue( soc );

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