**机器人足球实验二至五**

**班级：**计算机科学与技术21-1班  **实验日期:**2022.10.24

**小组成员：**

2021218067 王帅杨

2021218070 冀国鹏

2021218071 韩浩天

**实验二 Demeer5基本动作1**

**一、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

**二、实验设备**

硬件环境：PC机

软件环境：操作系统linux

**三、实验内容**

**1.将球踢向对方球门**

取得对方球门的位置作为kickTo参数，并指定速度，踢向对方球门。

|  |
| --- |
| VecPosition pos1 = WM->getPosOpponentGoal();  double speed = 2.0;  soc = kickTo( pos1, speed );  ACT->putCommandInQueue ( soc );  ACT->putCommandInQueue ( turnNeckToObject ( OBJECT\_BALL, soc ) ); |

**2.将球踢向距离自己最近的队友**

取得队友集合中距自己最近的队友的位置作为kickTo参数，踢向这个队友的位置。

|  |
| --- |
| ObjectT clstm;  clstm = WM->getClosestInSetTo( OBJECT\_SET\_TEAMMATES, posAgent);//最近的队友坐标  VecPosition clspos = WM->getGlobalPosition( clstm );  soc = leadingPass( clstm , 1.0 );  ACT->putCommandInQueue ( soc );  ACT->putCommandInQueue ( turnNeckToObject ( OBJECT\_BALL, soc ) ); |

**3.用不同的带球模式进行带球，并观察效果，比较异同**

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 ) ); |

**4.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 ) ); |

**5.让球员智能体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）**

先求球门方向，向该方向带球，进入禁区后射门。

|  |
| --- |
| if( WM->isInTheirPenaltyArea(WM->getBallPos()) )//进入敌方禁区  {  VecPosition posGoal(PITCH\_LENGTH/2.0,(-1 + 2\*(WM->getCurrentCycle()%2)) \* 0.4 \* SS->getGoalWidth());  soc = kickTo( posGoal, 2.7 );// 朝球门方向将球以最大力度踢出  }  else  soc = dribble((VecPosition(52.5,0) - posAgent).getDirection(), DRIBBLE\_FAST);//向敌球门带球  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  Log.log( 100, "kick ball" ); |

**实验三 Demeer5基本动作2**

**一、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 demeer5 中的具体函数内容实现对场上球员的控制

3.能通过底层动作的简单组合控制场上队员做出一些复杂动作

**二、实验设备**

硬件环境：PC机

软件环境：操作系统linux

**三、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->isInTheirPenaltyArea()来判断球是否在对方禁区）**

判断球员编号，并进行相应的决策。

|  |
| --- |
| if( WM->isInTheirPenaltyArea(WM->getBallPos()) )//进入敌方禁区  {  VecPosition posGoal(PITCH\_LENGTH/2.0,(-1 + 2\*(WM->getCurrentCycle()%2)) \* 0.4 \* SS->getGoalWidth());  soc = kickTo( posGoal, 2.7 );// 朝球门方向将球踢出  }  else//不在禁区  {  if( WM->getAgentObjectType()==OBJECT\_TEAMMATE\_9 ||  WM->getAgentObjectType()==OBJECT\_TEAMMATE\_8 ||  WM->getAgentObjectType()==OBJECT\_TEAMMATE\_7 )  {  soc = dribble(0.0,DRIBBLE\_SLOW); //带球  }  else  soc = leadingPass(OBJECT\_TEAMMATE\_9,1);//传球  } |

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

按球的位置判断是否在敌方半场，进行相应决策。

|  |
| --- |
| if( WM->getBallPos().getX() < 0 )//进入敌方半场  {  VecPosition posGoal(PITCH\_LENGTH/2.0, (-1 + 2\*(WM->getCurrentCycle()%2)) \* 0.4 \* SS->getGoalWidth());  soc = kickTo( posGoal, 2.7 );// 朝球门方向将球踢出  }  else//不在敌方半场  {  soc = dribble(0.0,DRIBBLE\_SLOW); //带球  }  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  Log.log( 100, "kick ball" ); |

**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，否则就自己带球（调用WM->getClosestRelativeInSet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）**

判断自身周围一定范围内对方球员的数目。若该范围内有球员，则传给最近队友。

|  |
| --- |
| Circle cir(posAgent,7);//判断周围7m内是否有敌方球员  int num = WM->getNrInSetInCircle(OBJECT\_SET\_OPPONENTS,cir);  AngDeg ang = (VecPosition(52.5,0) - posAgent).getDirection();  if(num>0)  soc =leadingPass(WM->getClosestInSetTo(OBJECT\_SET\_TEAMMATES , posAgent) ,1);//传给最近队友  else  soc = dribble(ang, DRIBBLE\_SLOW);//带球  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  Log.log( 100, "kick ball" ); |

**实验四 复杂的动作决策**

**一、实验目的**

进一步了解 demeer5 并能熟悉的修改 demeer5 的内容以达到对场上的球员的控制：

1.能理解 UVA 程序中原来的 demeer5 中的全部内容

2.能通过修改 demeer5 中的具体函数内容实现对场上球员的控制

3.能通过底层动作的简单组合控制场上队员做出一些复杂动作决策

4.对 WorldModel 有初步的认识，学会在 WorldModel,basicplayer 里添加新函数

**二、实验设备**

硬件环境：PC机

软件环境：操作系统linux

**三、实验内容**

**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，球门哪一方空隙较大，守门员的位置为VecPosition posGoalie = WM->getGlobalPosition(WM->getOppGoalieType())；球门位置坐标为（52.5，0），可尝试朝（52.5，6.5）（52.5，-6.5）两点射门）**

这里应该使用角度来比较空隙大小。将自身和守门员、球门上界、球门下界分别连线，比较夹角。将球向比较大的夹角踢。

|  |
| --- |
| else if ( WM->isBallKickable())//if kickable//如果球已知，而且当前球在我脚下(可踢)  {  if(WM->isInTheirPenaltyArea(WM->getBallPos())) {//朝空隙大的一方射门  VecPosition posGoalie;  posGoalie = WM->getGlobalPosition(WM->getOppGoalieType());    if( WM->maxAngShoot( posGoalie, posAgent ))  soc = kickTo(VecPosition(52.5,6.0),SS->getBallSpeedMax());  else  soc = kickTo(VecPosition(52.5,-6.0),SS->getBallSpeedMax());  }  else  soc = dribble(0.0,DRIBBLE\_SLOW); //带球  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  Log.log( 100, "kick ball" );  } |

射门判断函数maxAngShoot

/\*以下这个函数可以向空隙较大的一方射门，采用夹角比较

\* 参数：守门员位置，自己位置

\* 返回值：bool型，true为向上踢，false为向下踢

\* 需要在worldmodel.h里添加内容

\*/

|  |
| --- |
| bool WorldModel::maxAngShoot( VecPosition posGoalie, VecPosition posAgent )//朝较大的夹角射门  {  //VecPosition posGoalie;  AngDeg ang\_goalie, angup, angdown;    //posGoalie = WM->getGlobalPosition(WM->getOppGoalieType());  ang\_goalie = (posGoalie - posAgent).getDirection();  angup = (VecPosition(52.5,6.0) - posAgent).getDirection();  angdown = (VecPosition(52.5,-6.0) - posAgent).getDirection();    if(fabs(angup - ang\_goalie) > fabs(angdown - ang\_goalie))  return true;  else  return false;  } |

**2.在BasicPlay里添加一个带球函数，要求如果无人阻挡（带球将要经过的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）**

首先添加函数isOpponentAtAngleEx(),判断在视野中指定夹角和距离构成的扇形中是否存在对手球员。若没有则向前带球。闪避带球则是分别判断当前视野的左45°和右45°范围内有无对方球员。如有则向另一方向带球以实现躲避。

位置：worldmodelhighlevel.cpp

/\*

这个函数判断在扇区内有没有敌方球员，若有则ture 否则false

\*/

|  |
| --- |
| bool WorldModel::isOpponentAtAngleEx( AngDeg angA, AngDeg angB, double dDist )  {  VecPosition posAgent = getAgentGlobalPosition();  VecPosition posOpp;  AngDeg angOpp;  int iIndex;    for( ObjectT o = iterateObjectStart(iIndex, OBJECT\_SET\_OPPONENTS );  o != OBJECT\_ILLEGAL;  o = iterateObjectNext( iIndex, OBJECT\_SET\_OPPONENTS ))  {  posOpp = getGlobalPosition(o);  angOpp = ( posOpp - posAgent ).getDirection();  if ( angA <= angOpp && angOpp <= angB && posAgent.getDistanceTo( posOpp ) < dDist)  return true;  }  iterateObjectDone( iIndex ) ;  return false;  } |

位置：basicplayer.cpp

|  |
| --- |
| SoccerCommand BasicPlayer::dribble\_Dodge(VecPosition posAgent)//闪避带球  {  SoccerCommand soc;  //Circle cir(posAgent,7);  //ObjectT o = WM->getClosestInSetTo(OBJECT\_SET\_OPPONENTS,posAgent);  //VecPosition p = WM->getGlobalPosition(o);  //int num = WM->getNrInSetInCircle( OBJECT\_SET\_OPPONENTS, cir );  AngDeg ang = (VecPosition(52.5,0) - posAgent).getDirection();  soc = dribble(ang, DRIBBLE\_WITHBALL );  if( WM->isOpponentAtAngleEx( ang - 45, ang , 7 ) )  ang += 45;  else if( WM->isOpponentAtAngleEx( ang, ang + 45 , 7 ) )  ang -= 45;  soc = dribble( ang , DRIBBLE\_SLOW );  return soc;  } |

**实验五 特殊比赛模式的设计**

**一、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

**二、实验设备**

硬件环境：PC机

软件环境：操作系统linux

**三、实验内容**

**1.角球。如果比赛进入角球模式，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**

角球是要求向场内踢。编写pointKickTo()返回在不同角时应该踢向的位置。

当球在自己半场时，将位置的X坐标调大，尽量向对方半场踢；而在对方半场时，X设置得比较小，为夺球攻门创造机会。

位置：basicplayer

|  |
| --- |
| VecPosition BasicPlayer::pointKickTo()//计算角球应该踢向的点  {  VecPosition posBall, posKickTo;  double x, y;  posBall = WM->getBallPos();  x = posBall.getX();  y = posBall.getY();    posKickTo.setX(- x / fabs(x) \* 5 + x );  posKickTo.setY(- y / fabs(y) \* 12 + y );    return posKickTo;  } |

位置：demeer5

|  |
| --- |
| else if( WM->isCornerKickUs())//我方角球  {  if(WM->isBallKickable() )  soc = kickTo(pointKickTo(), SS->getBallSize());  else if(WM->getFastestInSetTo( OBJECT\_SET\_TEAMMATES, OBJECT\_BALL, &iTmp)  == WM->getAgentObjectType())  soc = moveToPos(posBall, PS->getPlayerWhenToTurnAngle());  else if(WM->getSecondClosestInSetTo(OBJECT\_SET\_TEAMMATES, OBJECT\_BALL)  == WM->getAgentObjectType())  soc = moveToPos(pointKickTo(), PS->getPlayerWhenToTurnAngle());  else  soc = moveToPos(WM->getStrategicPosition(), PS->getPlayerWhenToTurnAngle());    ACT->putCommandInQueue( soc ); // 放入命令队列  ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );  } |

**2.界外球。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，在此不在赘述。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。**

界外球和角球类似，区别在于比赛中很少有机会在对方底线发界外球。所以大部分情况都将接应点设置得更加靠近敌方半场，增加进攻机会。

位置：demeer5

|  |
| --- |
| else if( WM->isKickInUs() )//我方界外球  {  if( WM->isBallKickable() )  soc = kickTo(pointKickToK(), SS->getBallSize());  else if(WM->getFastestInSetTo( OBJECT\_SET\_TEAMMATES, OBJECT\_BALL, &iTmp)  == WM->getAgentObjectType())  soc = moveToPos(posBall, PS->getPlayerWhenToTurnAngle());  else if(WM->getSecondClosestInSetTo(OBJECT\_SET\_TEAMMATES, OBJECT\_BALL)  == WM->getAgentObjectType())  soc = moveToPos(pointKickToK(), PS->getPlayerWhenToTurnAngle());  else  soc = moveToPos(WM->getStrategicPosition(), PS->getPlayerWhenToTurnAngle());    ACT->putCommandInQueue( soc ); // 放入命令队列  ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );  } |

位置：basicplayer

/\* 此函数返回界外球应该踢向的点

\* 策略是把球尽可能往敌方半场踢

\*/

|  |
| --- |
| VecPosition BasicPlayer::pointKickToK()//计算界外球应该踢向的点  {  VecPosition posBall, posKickTo;  double x, y;  posBall = WM->getBallPos();  x = posBall.getX();  y = posBall.getY();    posKickTo.setX( x + 12 );  posKickTo.setY(- y / fabs(y) \* 12 + y );    return posKickTo;  } |

**3.定位球/任意球。简单的设计思路和实验内容与界外球差不多，在此不在赘述。略有不同的是，如果条件允许，开球队员可以选择直接射门。要求：要求完成一种任意球的开球动作。**

基本思路同界外球，都是尽可能向对方半场踢。但当条件合适（比如X比较大或者在禁区内），我们选择向较大的夹角直接射门。

|  |
| --- |
| else if( WM->isFreeKickUs() )//我方任意球  {  if( WM->isBallKickable() )  {  if(WM->getBallPos().getX() > 35 || WM->isInTheirPenaltyArea())//x大于35直接射门  soc = maxAngShoot( posAgent );  else  soc = kickTo(pointKickToK(), SS->getBallSize());  }  else if(WM->getFastestInSetTo( OBJECT\_SET\_TEAMMATES, OBJECT\_BALL, &iTmp)  == WM->getAgentObjectType())  soc = moveToPos(posBall, PS->getPlayerWhenToTurnAngle());  else if(WM->getSecondClosestInSetTo(OBJECT\_SET\_TEAMMATES, OBJECT\_BALL)  == WM->getAgentObjectType())  soc = moveToPos(pointKickToK(), PS->getPlayerWhenToTurnAngle());  else  soc = moveToPos(WM->getStrategicPosition(), PS->getPlayerWhenToTurnAngle());    ACT->putCommandInQueue( soc ); // 放入命令队列  ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );  } |