第五章　球员智能体世界模型

本章概要

- 世界模型概述
- 主要信息属性及获取方式
- 属性值的更新方法
- 世界模型的预测方法
- 世界模型的高级处理方法

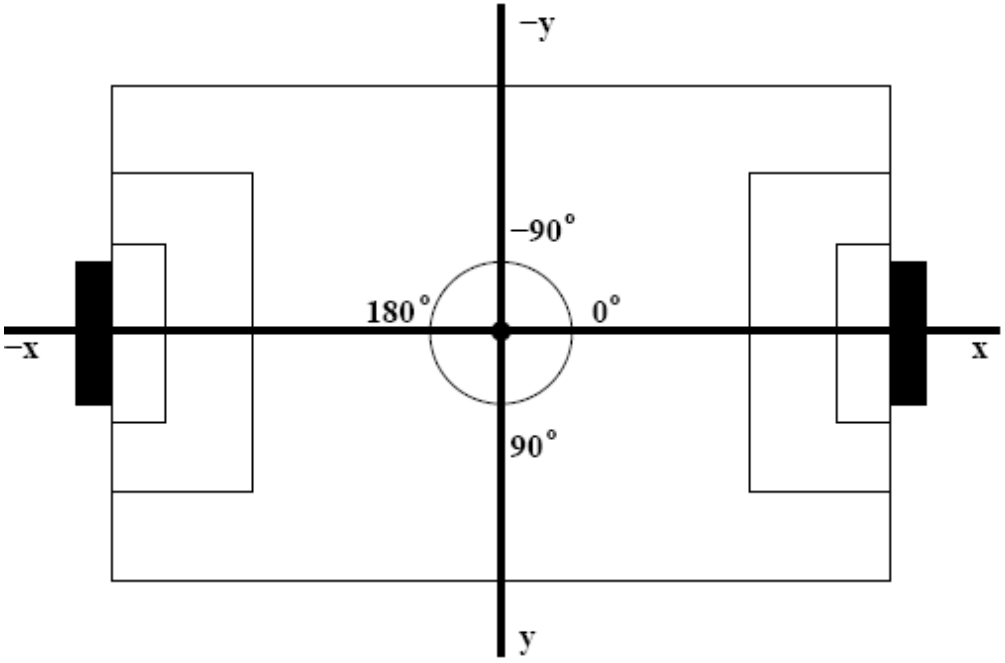
5.1世界模型概述

- **容器(类似STL)**
  - 存储信息并对信息进行进一步组织加工
- 具体信息内容
  - 配置信息，包含：Sever参数，异构球员取值范围，每一种异构球员的参数。
  - 比赛信息，包含：时间, 球员号码, 边线, 队名, 比赛模式,比分差。
  - 对象信息，包含：各类对象且有继承信息。
  - 动作信息，包含：每次动作执行的次数，动作队列，已执行的动作。

5.1世界模型概述

- 主要处理方法：
  - 环境信息获取方法，从球员智能体的世界模型中重新得到对象信息。
  - 环境信息更新方法，根据来自Sever的感觉信息更新球员智能体的WorldModel。
  - 环境信息预测方法，根据先前的感觉信息预测世界的未来状态。
  - 环境信息高级处理方法，从基本的WorldModel信息得出高层的结论。

5.1世界模型概述



5.2主要信息属性及获取方式

- 配置信息
- 比赛信息
- 对象信息
- 动作信息

5.2主要信息属性及获取方式 -配置信息

- ServerSettings *SS;
- PlayerSettings *PS;
- HeteroPlayerSettings pt[MAX_HETERO_PLAYERS];
- Formations *formations;
- 通过SS指针可以获得Server端配置的各种参数，如：球员大小、最大速度、比赛仿真周期等。具体为标准球员（0类型）的信息。
- PS指针主要存储决策临界变量，一些比较关键的临界值。
- Pt存储所有异构球员种类及该异构类型对应的球员属性在本场比赛的取值。注意，在一场比赛开始时，球员初始化后这些值才能得到确定，确定以后整场比赛的取值保持不变。

5.2主要信息属性及获取方式
-比赛信息

- Time timeLastSeeMessage; /*!< server time of last see msg */
- Time timeLastSenseMessage; /*!< server time of last sense msg */
- bool bNewInfo; /*!< indicates new info from server */
- Time timeLastCatch; /*!< time of last catch by goalie */
- Time timeLastRefMessage; /*!< time of last referee message */
- char strTeamName[MAX_TEAM_NAME_LENGTH]; /*!< Team name */
- int iPlayerNumber; /*!< player number in soccerserver */
- SideT sideSide; /*!< side where the agent started */
- PlayModeT playMode; /*!< current play mode in the game */
- int iGoalDiff; /*!< goal difference */

5.2主要信息属性及获取方式
-对象信息

- BallObject Ball; /*!< information of the ball */
- AgentObject agentObject; /*!< information of the agent itself */
- PlayerObject Teammates[MAX_TEAMMATES];/*!< information of all teammates */
- PlayerObject Opponents[MAX_OPPONENTS];/*!< information of all opponents */
- PlayerObject UnknownPlayers[MAX_TEAMMATES+MAX_OPPONENTS];
- int iNrUnknownPlayers;

- FixedObject Flags[MAX_FLAGS]; /*!< all flag information */
- FixedObject Lines[MAX_LINES]; /*!< all line information */

5.2主要信息属性及获取方式
-动作信息

- SoccerCommand queuedCommands[CMD_MAX_COMMANDS];

/*!< all performed commands,*/

- bool performedCommands[CMD_MAX_COMMANDS];

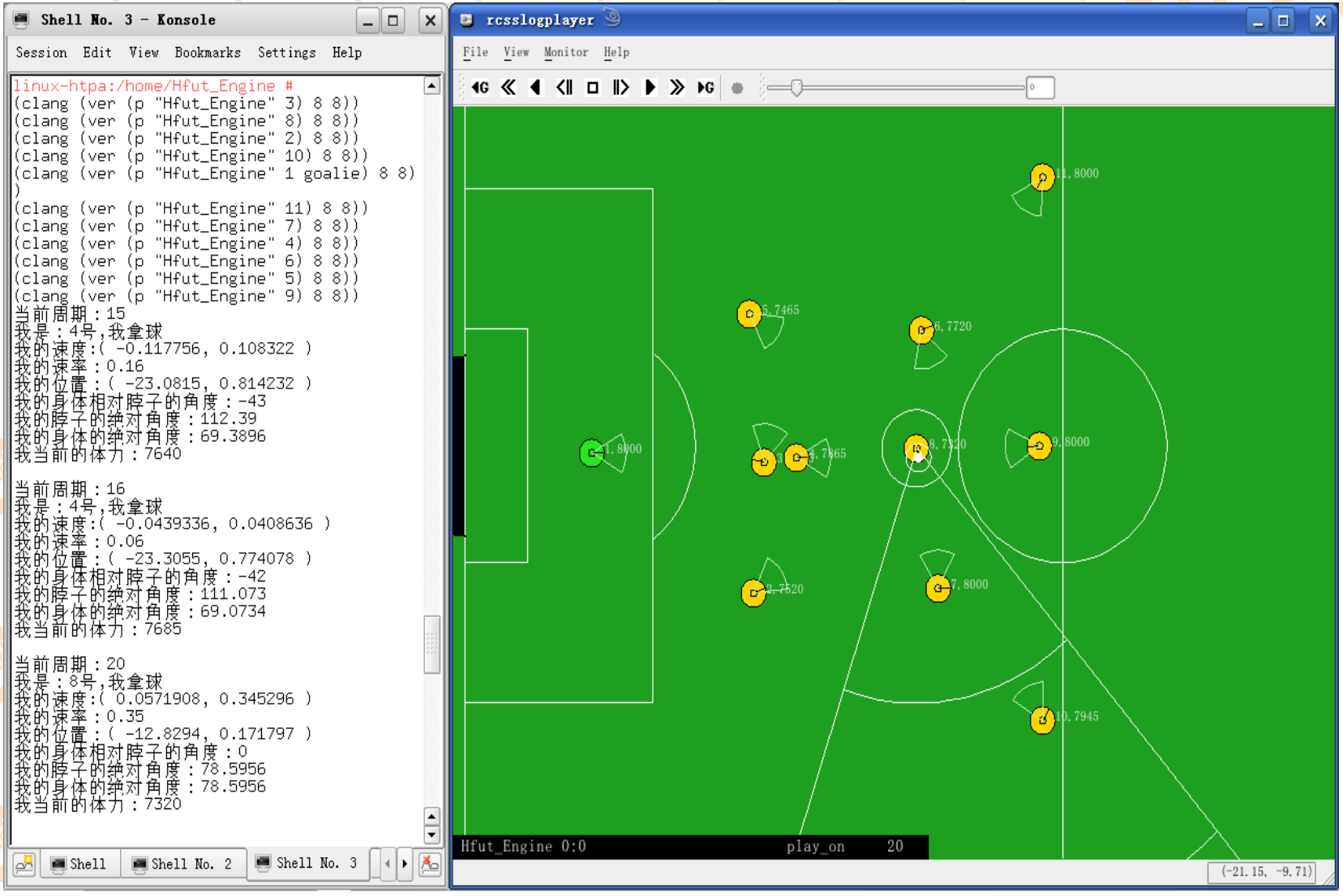
/*!< commands performed in last cycle, index is CommandT */

- int iCommandCounters[CMD_MAX_COMMANDS];

/*!< counters for all performed commands */

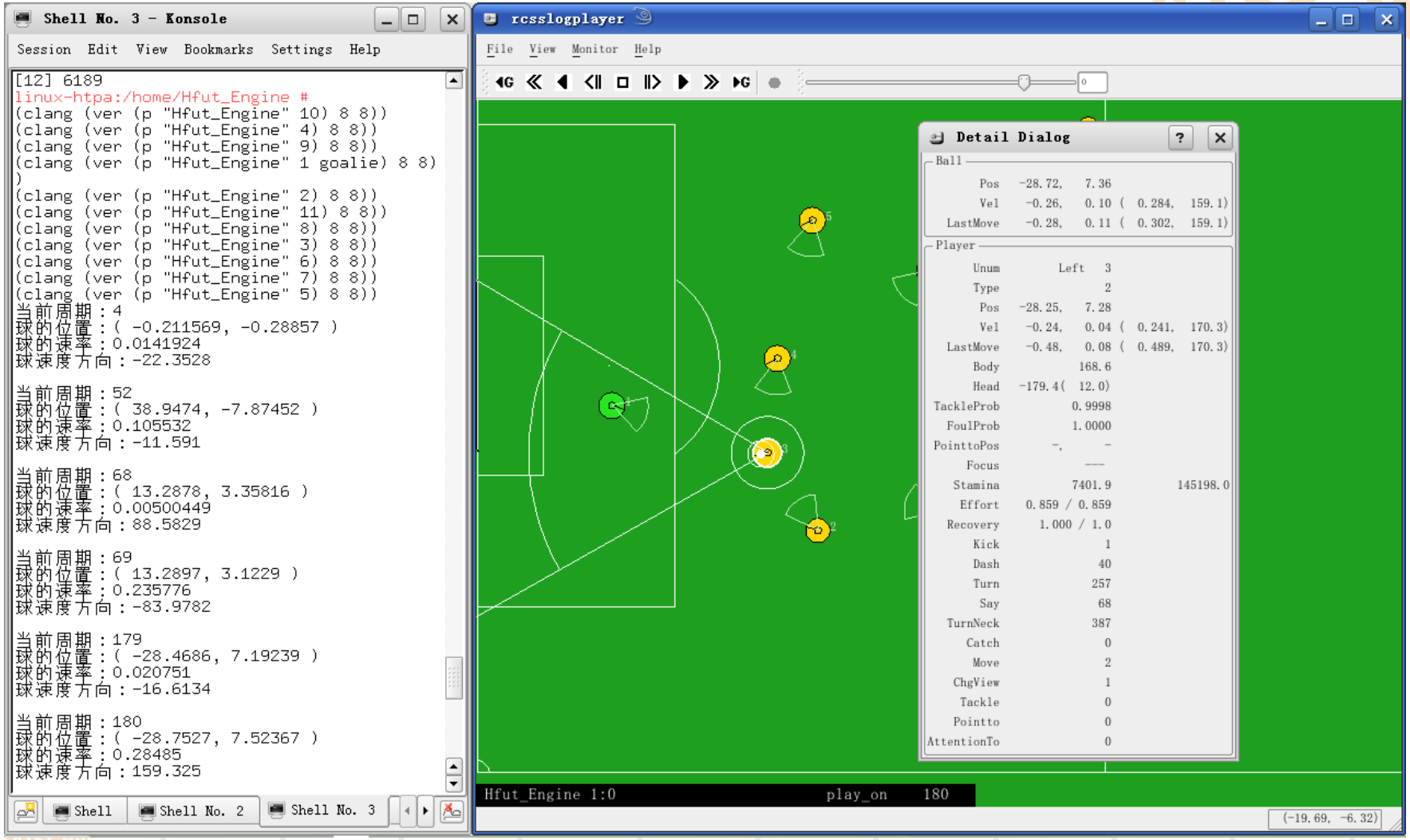
5.2主要信息属性及获取方式

-属性值的获取



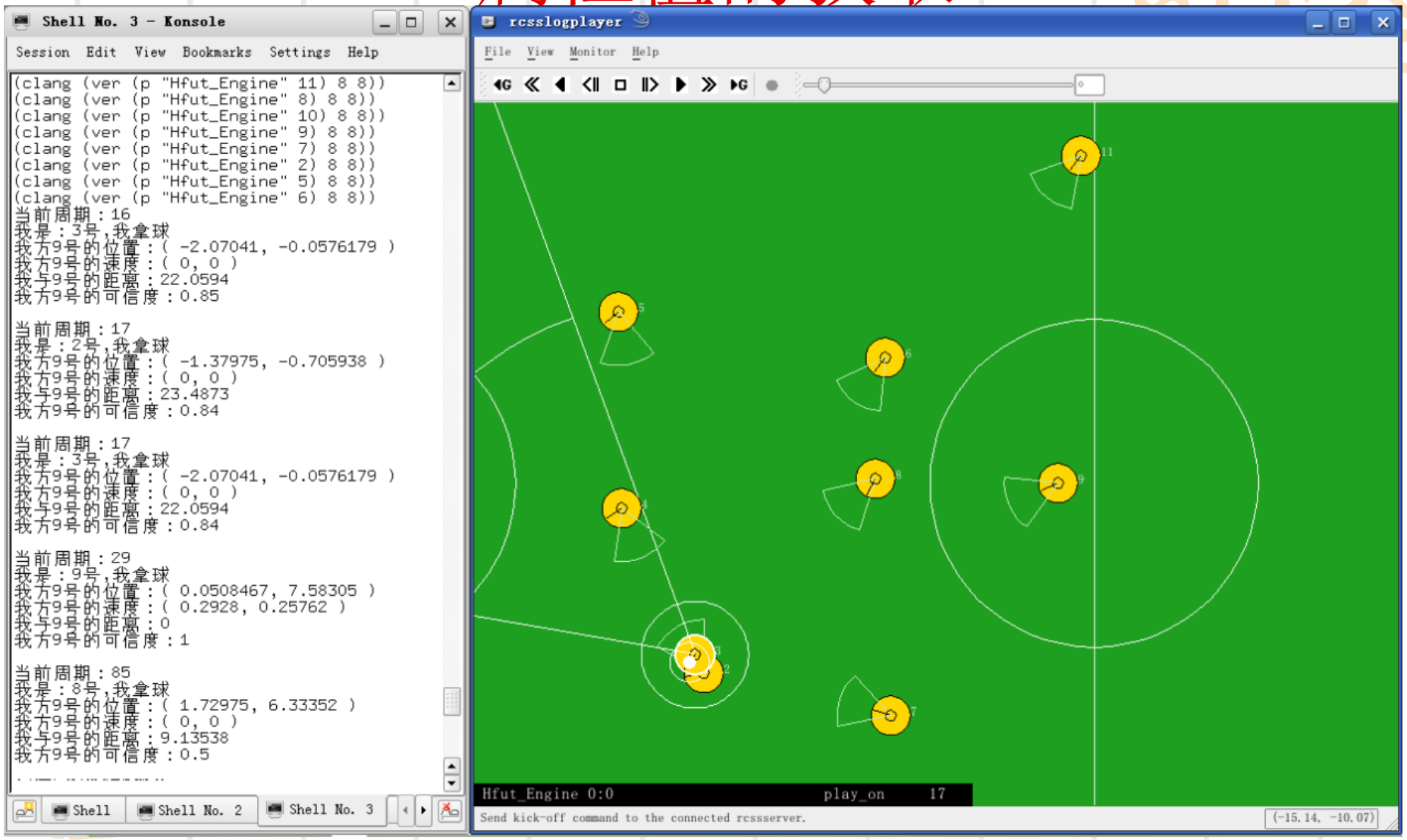
5.2主要信息属性及获取方式

-属性值的获取



5.2主要信息属性及获取方式

-属性值的获取



5.2主要信息属性及获取方式
-属性值的获取

- 比赛模式信息的获取
  - bool isFreeKickUs (PlayModeT pm=PM_ILLEGAL)
  - bool isKickInUs (PlayModeT pm=PM_ILLEGAL)
  - ……
- 球场标志（线）信息的获取
  - **VecPosition getPosOpponentGoal ()**
  - ……
- 其他
  - **int getTimeSinceLastCatch ()**
  - bool isLastMessageSee () const
  - ……..

5.3属性值的更新方法

- 根据自身感知信息更新世界模型
- 根据视觉感知信息更新世界模型
- 主要更新方法举例

5.3.1根据自身感知信息更新-更新自己

- 当前周期t
- 球员智能体的视觉宽度和视觉质量
- 球员智能体的体力
- 球员智能体过去累计执行的动作及其数量
- 相对于脖子方向的速度 $(\tilde{v}_x^t, \tilde{v}_y^t)$
- 脖子相对与身体方向 $\tilde{\theta}^t$

求解：

全局速度： v_x^t, v_y^t

头部全局方向： θ^t

5.3.1根据自身感知信息更新

$$(v_x^{t-1}, v_y^{t-1}) = (u_x^t, u_y^t) = \frac{(v_x^t, v_y^t)}{Decay}$$

$$\alpha = \frac{(1.0 + \tilde{r}) \cdot Moment}{1.0 + inertia\_moment \cdot \| (v_x^{t-1}, v_y^{t-1}) \|}$$

$$\theta^t = normalize(\theta^{t-1} + \alpha + \gamma)$$

α表示上一周期球员实际转身角度

γ 表示球员转头角度，它等于上一周期执行turn_neck 命令时使用的角度参数

$(\tilde{v}_x^t, \tilde{v}_y^t)$ 旋转 θ^t 得到(v_x^t, v_y^t)

5.3.1根据自身感知信息更新-更新其他球员和球的信息

- 针对其他队员的位置和速度的更新，除了接下来可以用到的视觉感知以外，因为无法得到类似于自身动作执行次数的实际数据，只能做加速度为0的假设，并在此假设下根据运动模型公式进行更新。
- 对于球，如果在自己的脚下，可以根据是否执行了自己的Kick指令（根据本周期Kick指令的数量是否增加1来确认）来得到球的加速度，如果没有提到就假设加速度为0，踢到了就下式计算：

$$(a_x^t, a_y^t) = act\_pow \times \mathbf{kick\_Power\_rate} \times (\cos(\theta_{ball}^t), \sin(\theta_{ball}^t))$$

5.3.2根据视觉感知信息更新

```
bReturn = updateAfterSeeMessage( );

processLastSeeMessage( );

updateAgentObjectAfterSee( );

for( ObjectT o = iterateObjectStart( iIndex, OBJECT_SET_PLAYERS, dConfThr );
     o != OBJECT_ILLEGAL;
     o = iterateObjectNext ( iIndex, OBJECT_SET_PLAYERS, dConfThr ) )
 {
   if( getTimeLastSeen( o ) == getTimeLastSeeMessage() &&
       o != getAgentObjectType() )
     updateDynamicObjectAfterSee   ( o );
   else
     updateObjectRelativeFromGlobal( o );
 }
 iterateObjectDone( iIndex );
```

updateAgentObjectAfterSee()

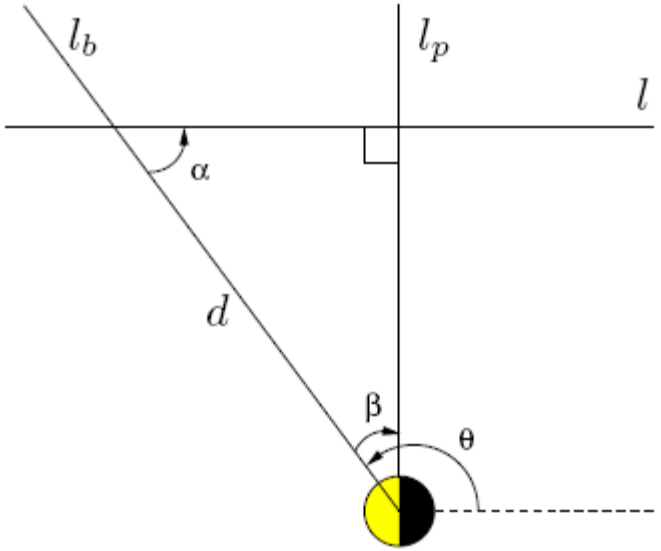
calculateStateAgent(&posGlobal, &velGlobal, &angGlobal);

5.3.2根据视觉感知信息更新

- 视觉信息：
  - 视觉信息所处的仿真周期t。
  - 球场上可见物体的信息，这些信息是相对于球员的视角的，根据不同情况可能包含以下内容中的几种：
    对象名称标识
    - 自身到对象的距离 r
    - 对象的相对角度 ϕ
    - 对象距离的改变 Δr
    - 对象角度的改变 $\Delta \phi$
    - 对象的身体角度 $\tilde{\theta}^t_{body}$
    - 对象的头部角度 $\tilde{\theta}^t_{neck}$

球员自身定位（p_x，p_y）

- 方法1：基于一条边线和一个标记定位
  - 边线的角度可以用来计算球员的全局头部角度，然后根据球员已知的标记的全局位置来计算球员的全局位置。



$$\beta = -sign(\alpha)(90 - |\alpha|)$$

边线	全局角度
OBJECT_LINE_R	0
OBJECT_LINE_B	90
OBJECT_LINE_L	180
OBJECT_LINE_T	-90

$$\theta = orientation(l_p) - \beta$$

$$(p_x, p_y) = (f_x, f_y) + \pi(\tilde{f}_r, \tilde{f}_\phi + \theta)$$

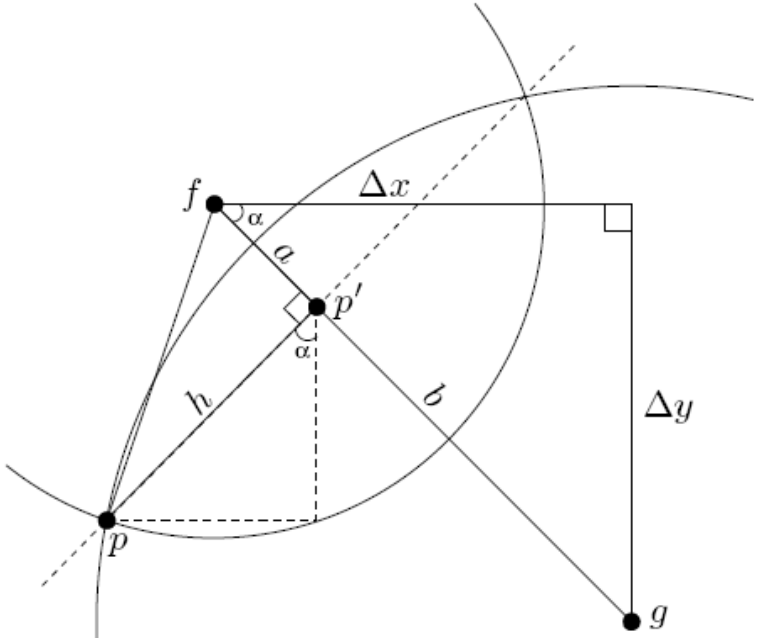
$$(x, y) = \pi(r, \phi) = (r \cdot \cos(\phi), r \cdot \sin(\phi))$$

球员自身定位（p_x，p_y）

- 使用2个标记来定位

标记 f 的相对极坐标$(\tilde{f}_r, \tilde{f}_\theta)$

标记 g 的相对极坐标$(\tilde{g}_r, \tilde{g}_\phi)$



f 和 g 的全局坐标(f_x, f_y)，(g_x, g_y)是已知的，因此球员智能体知道这两个标记点的全局位置(f_x, f_y)和(g_x, g_y)。结合标记点的全局位置以及与此点的距离，可得到一个智能体可能的位置圆。这个圆以此标记点为圆心，与标记点的距离为半径。同理，利用另外一个标记点也可得到一个位置圆。那么，智能体正确的位置必须同时位于两个圆上。一般情况，两个圆有两个交点，那么其中的一个交点必然是智能体的实际位置。

动态对象定位
-位置

得到了自身全局位置(p_x, p_y)和头部全局角度θ，球员就可以使用这些信息来计算视觉信息中包含的动态对象（球或者球员）的属性，这些需要计算的属性主要包括：全局绝对位置(q_x, q_y)，绝对速度(v_x, v_y)，身体全局角度θ^t_{body}，头部全局角度θ^t_{neck}，其中后两个属性只属于球员。

动态对象的全局位置(q_x, q_y)可以结合球员自身的全局位置和对象的相对视觉信息来获得。视觉直接包含了对象极坐标表示的相对位置$(\tilde{q}_r, \tilde{q}_\phi)$，而这个位置是相对与球员的头部视角的。具体公式为：

$$(q_x, q_y) = (p_x, p_y) + \pi(\tilde{q}_r, \tilde{q}_\phi + \theta)$$

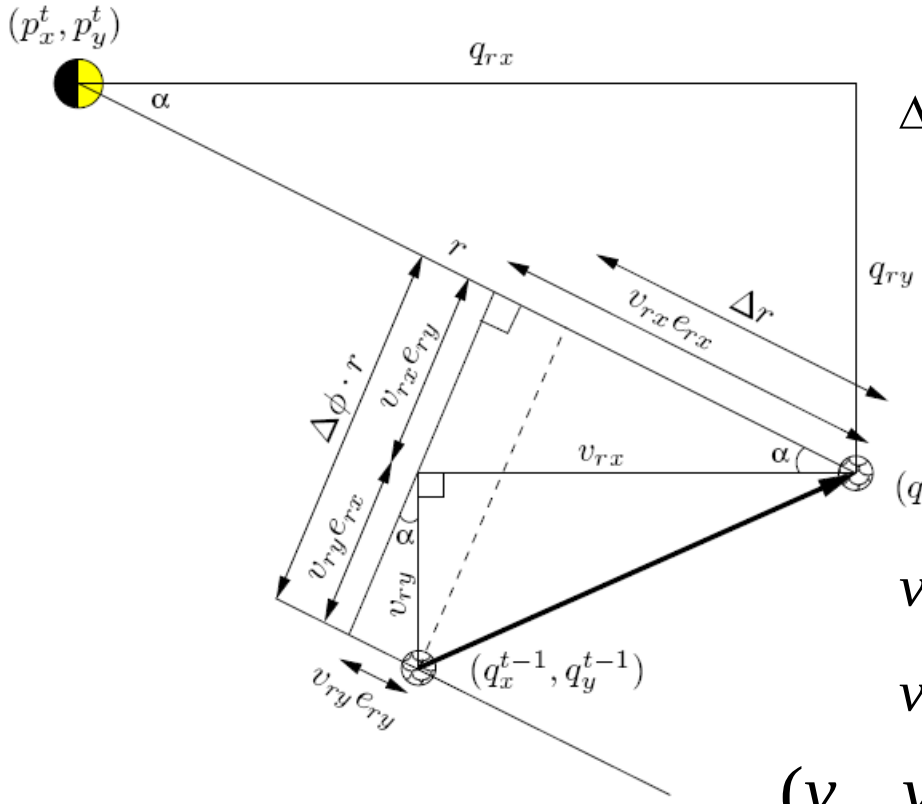
动态对象定位
-角度

$$\theta_{body}^{t} = \tilde{\theta}_{body}^{t} + \theta$$

$$\theta_{neck}^{t} = \tilde{\theta}_{neck}^{t} + \theta$$

动态对象定位
-速度

- 方法一：使用1个对象距离改变量和1个角度改变量来进行速度估计（以球为例）



$$\Delta r = (v_{rx} \cdot e_{rx}) + (v_{ry} \cdot e_{ry})$$

$$\Delta\phi = [(-(v_{rx} \cdot e_{ry}) + (v_{ry} \cdot e_{rx}))/r] \cdot (180/\pi)$$

$$e_{rx} = q_{rx}/r$$

$$e_{ry} = q_{ry}/r$$

$$(q_{rx}, q_{ry}) = (q_x, q_y) - (p_x, p_y)$$

$$v_{rx} = \Delta r \cdot e_{rx} - \Delta\phi \cdot (\pi/180) \cdot r \cdot e_{ry}$$

$$v_{ry} = \Delta r \cdot e_{ry} + \Delta\phi \cdot (\pi/180) \cdot r \cdot e_{rx}$$

(v_{rx}, v_{ry}) 旋转 θ (v_x, v_y)

动态对象定位
-速度

- 基于位置的速度估算

已知球的上个周期位置(q_x^{t-1}, q_y^{t-1})和当前的位置(q_x^t, q_y^t)，可计算出球的运动速度为：

$$(v_x^t, v_y^t) = ((q_x^t, q_y^t) - (q_x^{t-1}, q_y^{t-1})) \cdot ball\_decay$$

其中，$ball_decay$为球速衰减率。

5.3.3根据听觉感知信息更新

- 喊话机制
- 解析获得信息
- 转化成WorldModel

5.3.4主要更新方法举例

- void processSeeGlobalInfo (ObjectT o, Time time, VecPosition pos, VecPosition vel, AngDeg angBody, AngDeg angNeck)
- bool processNewAgentInfo (ViewQualityT vq, ViewAngleT va, double dStamina, double dEffort, double dSpeed, AngDeg angSpeed, AngDeg angHeadAngle)
- void processNewObjectInfo (ObjectT o, Time time, double dDist, int iDir, double dDistChange, double dDirChange, AngDeg angRelBodyAng, AngDeg angRelNeckAng, bool isGoalie, ObjectT objMin, ObjectT objMax)

5.4世界模型的预测方法

- 根据球员执行了某一特定动作来预测球员自身未来的状态
  - VecPosition predictAgentPos (int iCycles, int iDashPower=0)
  - bool predictStateAfterCommand (SoccerCommand com, VecPosition *pos, VecPosition *vel, AngDeg *angGlobalBody, AngDeg *angGlobalNeck, Stamina *sta=NULL)
  - void predictStaminaAfterDash (double dPower, Stamina *sta)
  - ……

5.4世界模型的预测方法

- 预测动态对象未来的状态
  - **VecPosition predictPosAfterNrCycles (ObjectT o, int iCycles, int iDashPower=100, VecPosition *vel=NULL)**
  - int predictNrCyclesToObject (ObjectT objFrom, ObjectT objTo)
  - ……

5.4世界模型的预测方法

- 预测球员到达某一位置所需时间
  - predictNrCyclesToPoint (ObjectT o, VecPosition posTo, AngDeg ang)
  - ……..

5.4世界模型的预测方法

5.5世界模型的高级处理方法

- 特定区域有多少球员
- 距离某目标或者点最近的球员
- 场上形势的简单判断条件，如是否我方控球等
- 返回特定区域的球员之间的一些角度
- 计算一些动作的实际发送参数

5.5世界模型的高级处理方法

- int getNrInSetInCircle (ObjectSetT objectSet, Circle c)
- int getNrInSetInCone (ObjectSetT objectSet, double dWidth, VecPosition start, VecPosition end)
- ObjectT getClosestInSetTo (ObjectSetT objectSet, ObjectT o, double *dDist=NULL, double dConfThr=-1.0)
- ObjectT getClosestRelativeInSet (ObjectSetT set, double *dDist=NULL)
- ObjectT getFastestInSetTo (ObjectSetT objectSet, ObjectT o, int *iCycles=NULL)
- ……

5.5世界模型的高级处理方法