1. **获取方式**

实验室代码仓库（private）：[https://github.com/SDNURoboticsAILab/BHuman\_for\_CRAIC/tree/RoboCup2024#](https://github.com/SDNURoboticsAILab/BHuman_for_CRAIC/tree/RoboCup2024)（里面有bhuman框架代码）

<https://github.com/SDNU-Robotics-and-AI-Lab/historicalCompetitionCode>（新仓库）

两版代码接口略有改动，脚本不通用

Git地址提供：（git clone --recursive）

实验室代码仓库是24版bhuman框架：git clone --recursive <https://github.com/SDNURoboticsAILab/BHuman_for_CRAIC.git>分支：RoboCup2024  
需要ssh密钥方式获取：git clone –recursive [git@github.com:SDNURoboticsAILab/BHuman\_for\_CRAIC.git](mailto:git@github.com:SDNURoboticsAILab/BHuman_for_CRAIC.git)  
  
官方仓库更新至25版：

git clone --recursive https://github.com/bhuman/BHumanCodeRelease.git

官方文档：<https://docs.b-human.de/coderelease2024/getting-started/initial-setup/>

1. **镜像刷机**

Bhuman镜像制作教程：（一系列教程介绍一共有四篇。）

<https://blog.xiaoqi.work/index.php/2024/05/23/bhuman%e4%ba%8c-bhuman%e9%95%9c%e5%83%8f%e5%88%b6%e4%bd%9c/>

Bhuman使用Cmake集成，依靠Cmake，生成部署工具Deploy和仿真平台simrobot，（官方文档上有工具介绍）

1. **结构介绍：**

概论：<https://sisterchen.github.io/2018/06/30/B-Human-framework/?utm_source=chatgpt.com>

B-Human 框架是一个基于模块化、线程化和行为树设计的 NAO 机器人软件架构，主要分为以下几个核心部分：

**1. 总体架构**

* **基于 C++ 的模块化设计**：  
  所有功能由独立的 **Module** 实现，通过数据结构（Representation）进行信息交换。
* **多线程运行**：  
  各传感器、感知模块、运动控制模块可分布在不同线程中，提高实时性。
* **行为树（Options）控制逻辑**：  
  使用层级化的行为树描述机器人动作逻辑，如点球、巡逻、攻防转换等。

**2. 核心组成**

| **模块** | **作用** |
| --- | --- |
| **Representations** | 定义数据类型，如球位置、障碍物感知、机器人姿态等 |
| **Modules** | 功能模块，例如视觉识别、定位、运动控制等 |
| **Cognition** | 感知与决策线程，负责视觉处理、定位、策略决策 |
| **Motion** | 运动控制线程，接收 Cognition 的决策并执行 |
| **BehaviorControl** | 行为控制模块，管理高层策略和状态机（如 PenaltyTaker） |
| **SimRobot** | 仿真平台，模拟真实比赛环境 |
| **Deploy** | 部署工具，将代码编译并上传到 NAO 机器人 |

**3. 数据流**

**传感器输入**（摄像头、陀螺仪、关节编码器等）

→ **感知处理模块**（如球门识别、球检测、障碍物检测）

→ **表示层（Representations）** 存储处理结果

→ **行为控制（BehaviorControl）** 根据感知结果决策

→ **运动模块（Motion）** 执行实际动作（行走、射门等）

→ **执行器输出**（舵机、发声器等）

**4. 线程结构**

* **Cognition 线程**：高频处理感知数据和策略计算
* **Motion 线程**：高频执行运动指令
* **Debug/Visualization 线程**：用于调试和可视化，如 SimRobot

1. **点球思路，25版规则**  
   见：第二十七届中国机器人及人工智能大赛比赛规则.txt

三轮，每轮三个，第一轮权重最大，每轮中第一球权重大于第二球大于第三球，全进加时。

思路：

RoboCup魔改；调用里面较为成熟的的点球模块Penalty，用来完成机器人点球。

1. **两版代码**

老代码是写死，机器人根据朝向转向。优点：进球率高。缺点：缺乏灵活性

新代码是利用上摄像头识别球门位置，与机器人所在位置进行向量坐标计算得出偏转角度，对球门柱内侧点射，参照物是球门柱。优点：具有一定灵活性，点球位置，可以小幅调整，但机器人必须正面面向球门。

完整代码应该是对球门柱和障碍物进行统一识别，但是以我们现在的水平还做不到。写不出来。

1. **摄像头：**

上摄像头使用神经网络识别（Src\Modules\Perception\PlayersPerceptors\RobotDetector.cpp）：

训练方法是compileNN  
Bhuman团队自研，轻量化。  
训练数据集（开源下载）：https://b-human.de/datasets-en.html?utm\_source

下摄像头传统识别方式：（"Src\Modules\Perception\PlayersPerceptors\PlayersDeeptectorFeatBOPLower.cpp"）

根据YUYV图像（灰度-色相-饱和度），框出识别内容，从相机坐标系转化到二维坐标系。

1. **改进思路：**

点球大概看了一下，应该重新训练一个模型就可以。

大家可以看看脚本，其中有一些是可以优化的，而且优化后的效果相当不错，可以好好研究一下

**八、基于脚本的修改**

路径：  
BHumanCodeRelease/Src/Modules/BehaviorControl/SkillBehaviorControl/Options/PenaltyShootout/PenaltyTaker.cpp

PenaltyTaker.cpp 是控制 NAO 机器人点球的重要脚本，由 PenaltyShootout 选项在点球状态下激活。

**1. 行为树相关宏**

**(1) option(...)**

定义一个行为选项（Option）。

示例：

option((SkillBehaviorControl) PenaltyTaker, ...)

表示该选项属于 SkillBehaviorControl 模块下的 PenaltyTaker。

**(2) 状态定义**

使用 initial\_state / state / target\_state 定义状态机各状态。

常见状态：

initial\_state(initial)：初始状态

state(goToBallAndKickLeft)：踢左球

state(goToBallAndKickRight)：踢右球

state(goBehindPenaltyMark)：回到点球点后方

每个状态包含：

transition：状态跳转条件

action：该状态的具体行为

**(3) common\_transition**

定义所有状态共享的跳转条件。

if(timeSincePenaltyShootoutStarted > 5000 && !theFieldBall.ballWasSeen(5000))

goto goBehindPenaltyMark;

含义：  
点球开始 5 秒后仍未看到球，则回到点球点后方寻找球。

**2. Lambda 表达式**

**示例：**

auto isPenaltyShootoutObstacleInGoal = [&](const bool left)

{

return theFrameInfo.getTimeSince(obstacleCellTimestamps[left ? 0 : 2]) < obstacleMaxAge;

}; // 判断障碍物是否在球门内

解析：

[&]：按引用捕获当前作用域所有变量，可直接在 Lambda 内访问 theFrameInfo、obstacleCellTimestamps、obstacleMaxAge 等。

(const bool left)：参数列表，用于判断是否检查左侧门柱的障碍物。

返回类型未显式声明 → 编译器自动推导为 bool（因函数体返回 bool 值）。

**3. 脚本主要功能**

定位球门与足球

规划射门动作

控制运动执行射门

**4. 点球参数配置**

配置文件：  
Config/Scenarios/Default/behaviorParameters.cfg

声明文件：  
Src/Representations/Configuration/BehaviorParameters.h

**九，配置文件**

**点球参数**

* 文件：Config/Scenarios/Default/behaviorParameters.cfg
* 声明：Src/Representations/Configuration/BehaviorParameters.h

**障碍物感知**

* Src/Representations/Perception/ObstaclesPercepts/ObstaclesImagePercept.cpp
* ObstaclesImagePercept.h
* ObstaclesFieldPercept.cpp
* ObstaclesFieldPercept.h

**线程配置**

* 文件：Config/Scenarios/Default/threads.cfg
* 功能：配置上下摄像头线程调用