Robot Standup System - Learning Guide

机器人起立系统 - 学习指南

System Overview

系统概述

The Hamburg Bit-Bots robot standup system (`bitbots_dynup`) enables humanoid robots to recover from falls using **IMU-based closed-loop control** with **quintic spline trajectories**. The system achieves **2.1-2.7 second recovery times** and operates on artificial turf conditions similar to RoboCup competitions. 汉堡比特机器人团队的机器人起立系统（`bitbots_dynup`）使用**基于IMU的闭环控制**和**五次样条轨迹**使人形机器人能够从跌倒中恢复。该系统实现了**2.1-2.7秒的恢复时间**，并在类似RoboCup比赛的人工草坪条件下运行。

Key Technical Components

关键技术组件

1. **Spline-Based Motion Generation**

1. 基于样条的动作生成

- **Quintic polynomial splines** generate smooth trajectories in Cartesian space
- **五次多项式样条**在笛卡尔空间中生成平滑轨迹
- **Continuous position, velocity, and acceleration** profiles prevent jerky movements
- **连续的位置、速度和加速度**轮廓防止突然的运动
- **4 end-effectors controlled**: left/right hands and feet
- **控制4个末端执行器**：左/右手和脚
- **6 DOF per end-effector**: x, y, z position + roll, pitch, yaw orientation
- **每个末端执行器6自由度**：x, y, z位置 + 滚转、俯仰、偏航方向

2. **Closed-Loop Stabilization**

2. 闭环稳定化

- **IMU-based PD controllers** using fused angles representation
- **基于IMU的PD控制器**使用融合角度表示
- **Real-time error correction** during dynamically unstable phases
- **实时误差校正**在动态不稳定阶段
- **Pitch and roll stabilization** prevents falls during transition
- **俯仰和滚转稳定化**防止过渡期间跌倒
- **295ms minimum lead time** for fall detection and recovery initiation
- **最小295毫秒前置时间**用于跌倒检测和恢复启动

3. **Multi-Direction Recovery**

3. 多方向恢复

- **Front standup**: Arms push forward, legs pull under body, transition to feet
- **前向起立**：手臂向前推，腿部拉到身体下方，过渡到脚部
- **Back standup**: Arms push backward, simultaneous leg positioning, roll-to-feet
- **后向起立**：手臂向后推，同时腿部定位，滚转到脚部
- **Side recovery**: Roll to back position, then execute back standup procedure
- **侧向恢复**：滚转到背部位置，然后执行后向起立程序

4. **Parameter Optimization**

4. **参数优化**

- **MOTPE/TPE algorithms** for automatic parameter tuning
- **MOTPE/TPE算法**用于自动参数调优
- **15-24 free parameters** per direction (timing + pose parameters)
- **每个方向15-24个自由参数**（时序+姿态参数）
- **Multi-objective optimization**: balance speed vs stability
- **多目标优化**：平衡速度与稳定性
- **Sim-to-real transfer** validated on multiple robot platforms
- **仿真到现实转换**在多个机器人平台上验证

How the System Works

系统工作原理

Motion Pipeline

动作管道

```
Fall Detection → Spline Generation → Stabilization → Inverse Kinematics → Motor Commands
跌倒检测 → 样条生成 → 稳定化 → 逆运动学 → 电机命令
```

1. **Spline Generation** (`dynup_engine.cpp:20-45`)

2. **样条生成**（`dynup_engine.cpp:20-45`）

   - Creates quintic polynomial trajectories for each end-effector
   - 为每个末端执行器创建五次多项式轨迹
   - Defines motion phases with semantic timing parameters
   - 用语义时序参数定义动作阶段
   - Initial pose = current robot position for smooth start
   - 初始姿态 = 当前机器人位置以实现平滑启动

3. **Stabilization** (`dynup_stabilizer.cpp`)

4. **稳定化**（`dynup_stabilizer.cpp`）

   - Uses IMU fused angles (avoids gimbal lock)
   - 使用IMU融合角度（避免万向节锁）

- PD controllers adjust foot placement to correct trunk orientation
- PD控制器调整脚部放置以校正躯干方向
- Only active during dynamically unstable phases
- 仅在动态不稳定阶段激活

5. **Inverse Kinematics** (BioIK solver)

6. **逆运动学**（BioIK求解器）

    - Converts Cartesian end-effector poses to joint angles
    - 将笛卡尔末端执行器姿态转换为关节角度
    - Handles non-parallel kinematic chains in humanoid robots
    - 处理人形机器人中的非平行运动链
    - Approximates unreachable poses for low-DOF arms
    - 为低自由度手臂近似不可达姿态

Core Configuration Parameters (`dynup_config.yaml`)

核心配置参数（`dynup_config.yaml`）

Timing Parameters (Front): **时序参数（前向）：**

- `time_hands_side`: 0.3s - Move arms to sides
- `time_hands_side`：0.3秒 - 手臂移动到侧面
- `time_hands_rotate`: 0.3s - Rotate arms forward
- `time_hands_rotate`：0.3秒 - 手臂向前旋转
- `time_hands_front`: 0.3s - Push arms forward
- `time_hands_front`：0.3秒 - 手臂向前推
- `time_foot_ground_front`: 0.132s - Place feet on ground
- `time_foot_ground_front`：0.132秒 - 脚部接地
- `time_torso_45`: 0.462s - Lift torso to 45°
- `time_torso_45`：0.462秒 - 躯干抬升至45°
- `time_to_squat`: 0.924s - Transition to squat position
- `time_to_squat`：0.924秒 - 过渡到蹲姿

Pose Parameters: **姿态参数：**

- `leg_min_length_front`: 0.244m - Minimum leg extension
- `leg_min_length_front`：0.244米 - 最小腿部伸展
- `max_leg_angle`: 71.71° - Maximum leg angle during recovery
- `max_leg_angle`：71.71° - 恢复期间最大腿部角度
- `trunk_overshoot_angle_front`: -5.0° - Torso overshoot compensation
- `trunk_overshoot_angle_front`：-5.0° - 躯干超调补偿

Getting Started Guide

入门指南

Step 1: Understand the Architecture

第1步：理解架构

1. **Read the research paper**: `01_wb_works/01.02_papers/02_md/05 Fast and Reliable Stand-Up Motions...md`
2. **阅读研究论文**：`01_wb_works/01.02_papers/02_md/05 Fast and Reliable Stand-Up Motions...md`
3. **Study the README**: `bitbots_motion/bitbots_dynup/README.md`
4. **学习README**：`bitbots_motion/bitbots_dynup/README.md`
5. **Examine configuration**: `bitbots_motion/bitbots_dynup/config/dynup_config.yaml`
6. **检查配置**：`bitbots_motion/bitbots_dynup/config/dynup_config.yaml`

Step 2: Explore the Code Structure

第2步：探索代码结构

```
bitbots_dynup/
├── include/bitbots_dynup/
│   ├── dynup_engine.hpp      # Main motion generation engine
│   ├── dynup_engine.hpp      # 主要动作生成引擎
│   ├── dynup_stabilizer.hpp  # IMU-based stabilization
│   ├── dynup_stabilizer.hpp  # 基于IMU的稳定化
│   ├── dynup_ik.hpp          # Inverse kinematics wrapper
│   ├── dynup_ik.hpp          # 逆运动学包装器
│   └── dynup_node.hpp        # ROS2 node interface
│   └── dynup_node.hpp        # ROS2节点接口
├── src/
│   ├── dynup_engine.cpp      # Spline generation and control logic
│   ├── dynup_engine.cpp      # 样条生成和控制逻辑
│   ├── dynup_stabilizer.cpp  # PD controller implementation
│   ├── dynup_stabilizer.cpp  # PD控制器实现
│   └── dynup_node.cpp        # ROS2 integration
│   └── dynup_node.cpp        # ROS2集成
└── config/
    ├── dynup_config.yaml     # Main configuration parameters
    ├── dynup_config.yaml     # 主配置参数
    └── dynup_sim.yaml        # Simulation-specific parameters
    └── dynup_sim.yaml        # 仿真特定参数
```

Step 3: Key Files to Study First

第3步：首先研究的关键文件

1. `dynup_engine.cpp:20-100` - Understand spline initialization
2. `dynup_engine.cpp:20-100` - 理解样条初始化
3. `dynup_config.yaml:58-228` - Learn parameter meanings
4. `dynup_config.yaml:58-228` - 学习参数含义
5. **Research paper pages 287-440** - Grasp theoretical foundation
6. **研究论文第287-440页** - 掌握理论基础
7. `dynup_stabilizer.cpp` - Study closed-loop control implementation

8. `dynup_stabilizer.cpp` - 研究闭环控制实现

Step 4: Hands-On Learning Path

第4步：实践学习路径

> ⚠ **IMPORTANT: Start with Simulator Environment** ⚠ **重要：从仿真环境开始**
>
> Always begin learning with simulation before moving to physical hardware. This approach is: 在转向物理硬件之前，始终从仿真开始学习。这种方法是：
>
> - **Safer**: No risk of damaging expensive robot hardware during learning
> - **更安全**：在学习过程中没有损坏昂贵机器人硬件的风险
> - **Faster**: Rapid iteration cycles for parameter testing and modification
> - **更快**：参数测试和修改的快速迭代周期
> - **Proven**: Sim-to-real transfer validated on multiple robot platforms
> - **已验证**：仿真到现实转换在多个机器人平台上得到验证
> - **Recommended**: BitBots team uses separate `dynup_sim.yaml` for simulation learning
> - **推荐**：BitBots团队为仿真学习使用单独的`dynup_sim.yaml`

Beginner (Week 1-2) - SIMULATION ONLY: **初级（第1-2周）- 仅仿真：**

- **Start with simulation**: `ros2 launch bitbots_dynup test.launch`
- **从仿真开始**：`ros2 launch bitbots_dynup test.launch`
- Run existing standup motions in simulation environment
- 在仿真环境中运行现有起立动作
- Modify timing parameters in `dynup_sim.yaml` and observe effects
- 修改`dynup_sim.yaml`中的时序参数并观察效果
- Study spline generation mathematics (quintic polynomials)
- 学习样条生成数学（五次多项式）
- Learn fused angles representation for orientation
- 学习方向的融合角度表示
- Practice with debug monitoring: `ros2 topic echo /dynup_engine_debug`
- 练习调试监控：`ros2 topic echo /dynup_engine_debug`

Intermediate (Week 3-4) - SIMULATION + THEORY: **中级（第3-4周）- 仿真+理论：**

- Implement custom pose sequences in simulation
- 在仿真中实现自定义姿态序列
- Test parameter optimization with MOTPE in simulation
- 在仿真中测试MOTPE参数优化
- Study stabilization PD controller tuning principles
- 学习稳定化PD控制器调优原理
- Compare open-loop vs closed-loop performance in simulation
- 在仿真中比较开环与闭环性能
- Master simulation before considering hardware
- 在考虑硬件之前掌握仿真

Advanced (Week 5-8) - HARDWARE TRANSITION: **高级（第5-8周）- 硬件过渡：**

- **Only after mastering simulation**: Transition to physical robot
- **仅在掌握仿真后**：过渡到物理机器人
- Port optimized parameters from `dynup_sim.yaml` to `dynup_config.yaml`
- 将优化参数从`dynup_sim.yaml`移植到`dynup_config.yaml`
- Test on physical robot with validated parameters
- 使用验证参数在物理机器人上测试
- Implement multi-objective optimization objectives
- 实现多目标优化目标
- Add new motion phases or recovery strategies
- 添加新的动作阶段或恢复策略
- Optimize for specific competition conditions
- 针对特定比赛条件进行优化

Step 5: Development Environment Setup

第5步：开发环境设置

Required Dependencies: **必需依赖项**：

```
# Install ROS2 Iron
# 安装ROS2 Iron
sudo apt install ros-iron-desktop

# Install optimization libraries
# 安装优化库
pip install optuna matplotlib

# Clone and build workspace
# 克隆并构建工作空间
git clone <bitbots-repo>
cd bitbots
make install
colcon build --packages-select bitbots_dynup
```

Test Commands: **测试命令**：

```
# SIMULATION FIRST (recommended starting point)
# 仿真优先（推荐起点）
ros2 launch bitbots_dynup test.launch

# Monitor debug output during simulation
# 仿真期间监控调试输出
ros2 topic echo /dynup_engine_debug

# Physical robot launch (only after simulation mastery)
# 物理机器人启动（仅在掌握仿真后）
ros2 launch bitbots_dynup dynup.launch
```

Environment-Specific Configuration: 环境特定配置：

- **Simulation**: Uses `config/dynup_sim.yaml` with adapted parameters
- **仿真**：使用`config/dynup_sim.yaml`与适配参数
- **Physical Robot**: Uses `config/dynup_config.yaml` with hardware-tuned parameters
- **物理机器人**：使用`config/dynup_config.yaml`与硬件调优参数
- **Parameter Transfer**: Validated sim-to-real transfer methodology ensures smooth transition
- **参数转换**：验证的仿真到现实转换方法确保平滑过渡

Key Learning Resources

关键学习资源

Primary Sources

主要来源

1. **Research Paper**: "Fast and Reliable Stand-Up Motions for Humanoid Robots Using Spline Interpolation and Parameter Optimization" (2021)
2. **研究论文**："Fast and Reliable Stand-Up Motions for Humanoid Robots Using Spline Interpolation and Parameter Optimization"（2021）
3. **Implementation**: `bitbots_motion/bitbots_dynup/` package
4. **实现**：`bitbots_motion/bitbots_dynup/`包
5. **Configuration Guide**: Parameter meanings in `dynup_config.yaml`
6. **配置指南**：`dynup_config.yaml`中的参数含义

Theoretical Background

理论背景

- **Quintic Spline Mathematics**: Continuous acceleration/velocity trajectories
- **五次样条数学**：连续加速度/速度轨迹
- **Fused Angles**: Singularity-free orientation representation
- **融合角度**：无奇点方向表示
- **BioIK**: Non-linear inverse kinematics for humanoid robots
- **BioIK**：人形机器人的非线性逆运动学
- **MOTPE**: Multi-objective Bayesian optimization
- **MOTPE**：多目标贝叶斯优化

Performance Benchmarks

性能基准

- **Speed**: 2.1-2.7s recovery time (optimized vs 3-4s manual)
- **速度**：2.1-2.7秒恢复时间（优化后与3-4秒手动相比）
- **Success Rate**: 85-95% on artificial turf
- **成功率**：人工草坪上85-95%
- **Platforms**: Wolfgang-OP, Darwin-OP, Sigmaban robots tested
- **平台**：Wolfgang-OP、Darwin-OP、Sigmaban机器人已测试

- **Competition Validation**: RoboCup Humanoid League since 2015
- **比赛验证**：自2015年以来RoboCup人形机器人联赛

Next Steps for Deeper Learning

深度学习的下一步

1. **Implement parameter optimization** for custom robot platform
2. **为自定义机器人平台实现参数优化**
3. **Study multi-contact dynamics** in complex recovery scenarios
4. **研究复杂恢复场景中的多接触动力学**
5. **Explore reinforcement learning** alternatives and comparisons
6. **探索强化学习**替代方案和比较
7. **Investigate sim-to-real transfer** challenges and solutions
8. **调查仿真到现实转换**挑战和解决方案
9. **Design new recovery strategies** for specific failure modes
10. **为特定故障模式设计新的恢复策略**

This system represents a mature, competition-tested approach to humanoid robot recovery that balances theoretical rigor with practical performance requirements. 该系统代表了一种成熟的、经过比赛测试的人形机器人恢复方法，平衡了理论严谨性与实际性能要求。