Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

Deep Reinforcement Learning for Bipedal Locomotion

Oberseminar TAMS

Marc Bestmann

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

04. December 2018

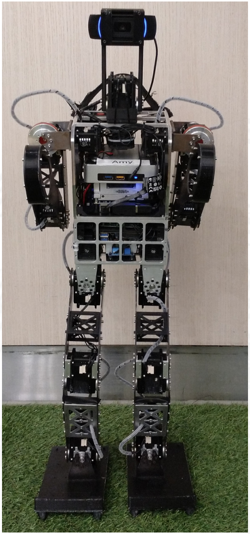
Table of Contents

What I did till now

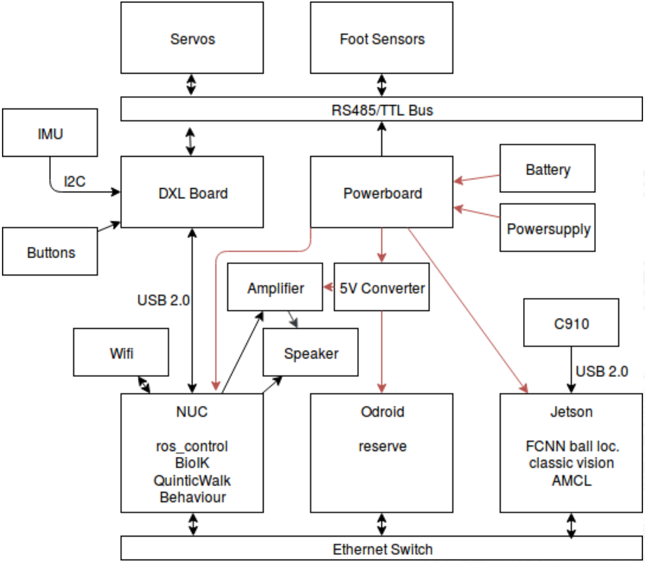
- Creating new humanoid platform (Wolfgang)
- Basic open-loop walking
- Parameter learning

Wolfgang - Components

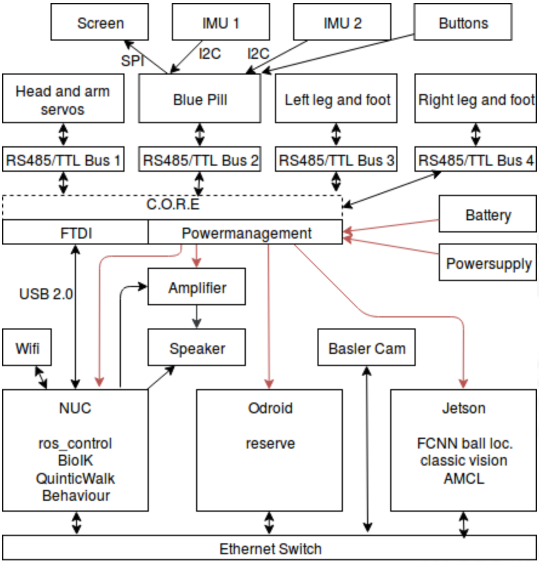
- Successor of a successor of Nimbro-OP
- Material: carbon, steel, aluminium, PLA
- Joints: 20 DOF using Dynamixels
  - Spring protectors for shoulder joints
- Sensors
  - Logitech C910 / (Basler camera)
  - (2x) IMU
  - Foot Pressure sensors
- Computers
  - Intel NUC i5
  - Nvidia Jetson TX2
  - Odroid XU-4
- Electronics
  - Network switch
  - Powerboard
  - DXL Board
  - Speaker, (small Display)
  - 2 Buttons, servo power switch



Current Structure



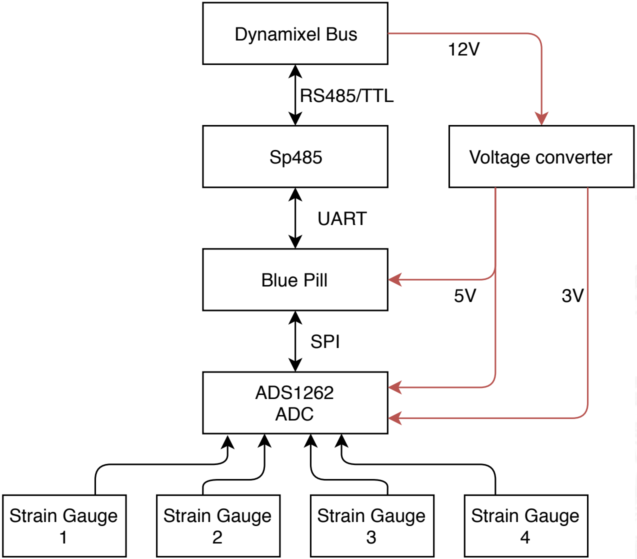
Goal Structure



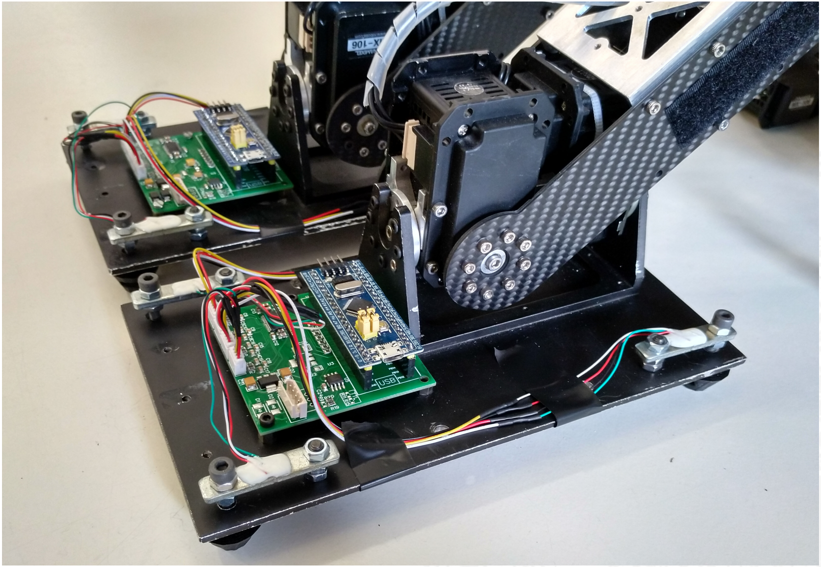
Performance

- Camera
  - Current: USB2 10Hz 640*480
  - Next version: GigE 30Hz 3MP
- Servos
  - Reading: position, velocity, torque
  - Control: position, velocity, current, current based position
- Foot pressure sensors
  - 4 strain gauges per feet
  - Max force: 40 kg
  - Resolution: ~24bit
- Dynamixel bus read/write all
  - Current: 2Mbaud, ~250Hz
  - Next version: 3 x 4.5Mbaud, ~1kHz

Foot Pressure Sensors



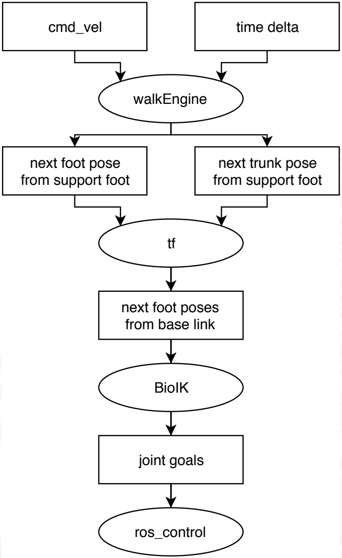
Foot Pressure Sensors



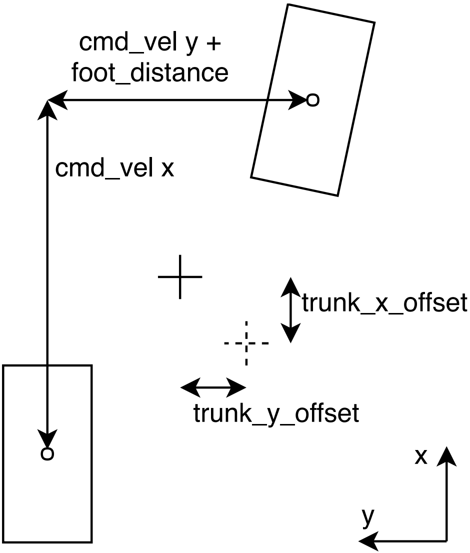
Outline

QuinticWalk

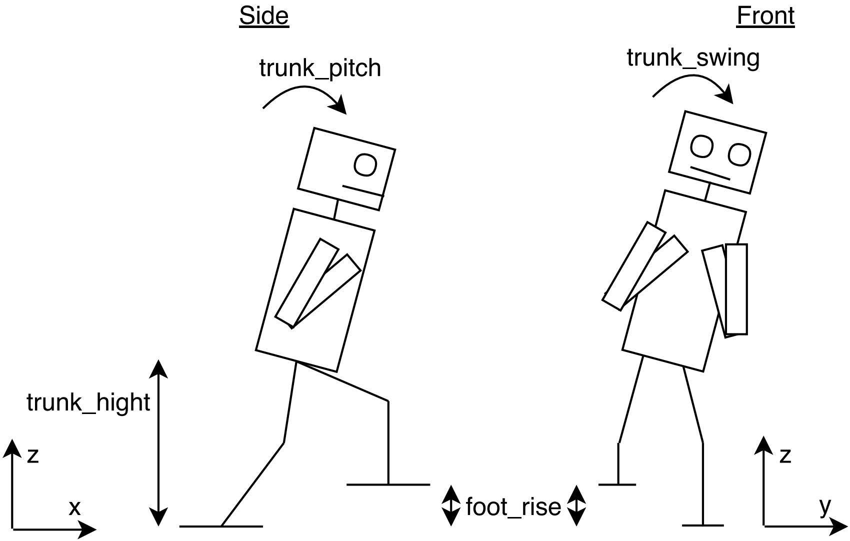
- Based on work from Rhoban
- Holonomic
- Fixed phase
- Open loop
- Direct parameter based
  - Less magical
- Generalization to any bipedal robot
  - Successfully used on two different robot types
- Control like a wheeled robot



QuinticWalk Engine - Next Step

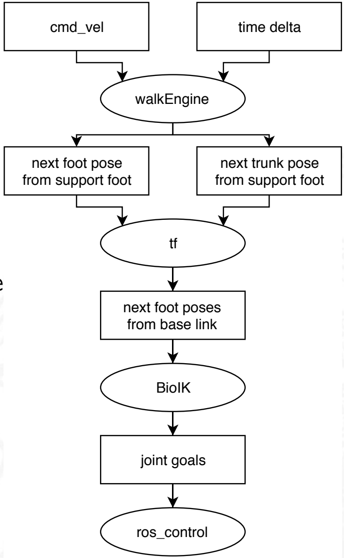


QuinticWalk Engine - Parameters



QuinticWalk Engine - Splines

- Given
  - start and end pose of flying foot
  - start and end pose of trunk
  - key values in between
- Searched
  - flying foot pose at any point of time
  - trunk pose at any point of time
- Solution
  - spline interpolation
  - 6 splines for foot
  - 6 splines for trunk

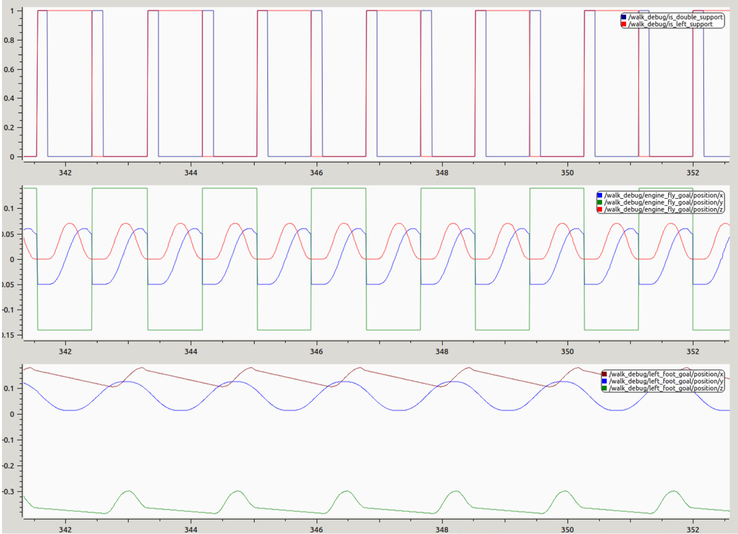


Quintic Splines

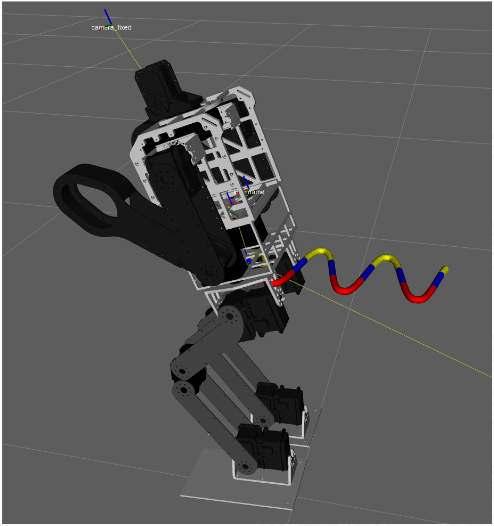
- define a set of spline points with
  - time (x)
  - position (y)
  - velocity (y')
  - acceleration (y'')
- fit polynoms of degree 5 between two points
- get values at time point t by solving corresponding polynom

QuinticWalk Engine

QuinticWalk Engine



QuinticWalk Engine



Parameter GUI

QuinticWalk Video

Video from RoboCup 2018 in Montreal
Push recovery video

QuinticWalk Engine - Advanced Features

- Performance can be increased by introducing overshoot
  - For x and y direction
  - defined by a overshoot_ratio and overshoot_phase
- Small kicks can easily be introduced
  - Adding conditional spline points
  - kick_length for additional movement in x
  - kick_phase for the timing

QuinticWalk Engine - Falling

- Robot still falls sometimes
  - Very stable sagital
  - Problems in lateral plane
  - Robot builds up lateral errors
- Simple solutions
  - Pausing when unstable
    - Gives the robot time to lose lateral energy
    - (previous) IMU not precise/fast enough
    - Try with new foot sensors
  - Phase reset when foot touches ground
    - Reduces build up
    - Good results for Rhoban
  - Using balance goal of BioIK
    - Closing loop by computing CoP
    - Choosing trunk pose based on this
  - Finding perfect set of parameters
    - See next section

Outline

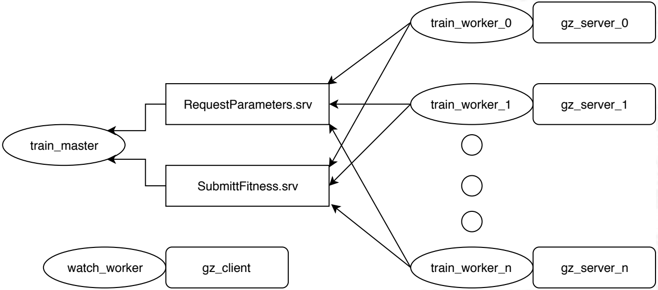
1. Previous Work
   - Wolfgang Platform
   - Quintic Walk
   - Parameter Learning
   - Side Projects
2. Deep Reinforcement Learning

Parameter Search

- Idea: Find optimal parameters automatically
- Problem: a lot of possible combinations
- Assumption: Small changes on a parameter have small effect
- Still a lot of parameter combinations
- Idea: Limit parameter values
- Information needed
  - Correlation between parameter value and fitness
  - Correlation between multiple parameters and fitness
- Getting this by random sampling the parameter space
- Many samples needed -> not possible with real hardware
- Running single simulation also to slow

Parameter Search - Simulation Setup

- Start master node
- Start n worker nodes on any computers
- For worker in workers until number of evaluations reached
  - Get set of parameters from master using a service
  - Evaluate the performance of these parameters
  - Return the performance for this set of parameters to the master
- Results are written in a .csv file

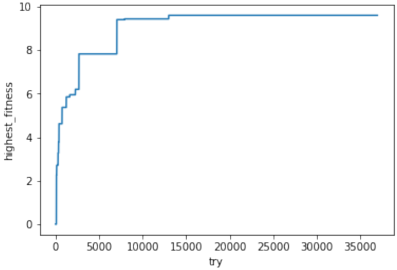


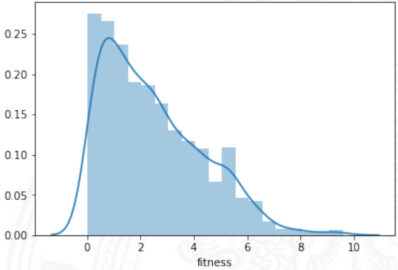
Simulation Run

- Reset robot
  - Set robot joints to start position
  - Set robot to start pose
- Do forward walk
- Measure distance
- Reset robot
- Do sideward walk
- Measure distance
- Reset robot
- Measure angle
- Compute fitness by adding distances and angle
- Break with fitness 0 if robot falls

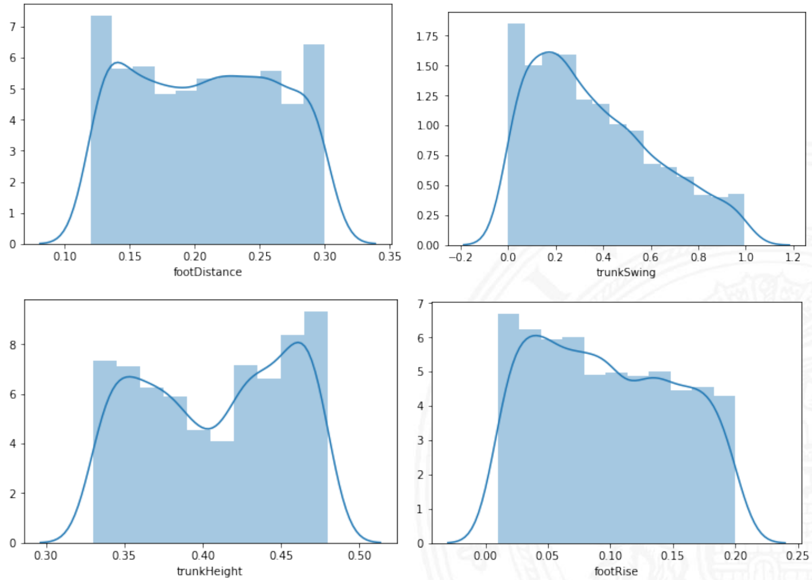
Results

- 36958 sets tested
- 1237 fitness > 0
- 269 fitness > 4
- 55 fitness > 6

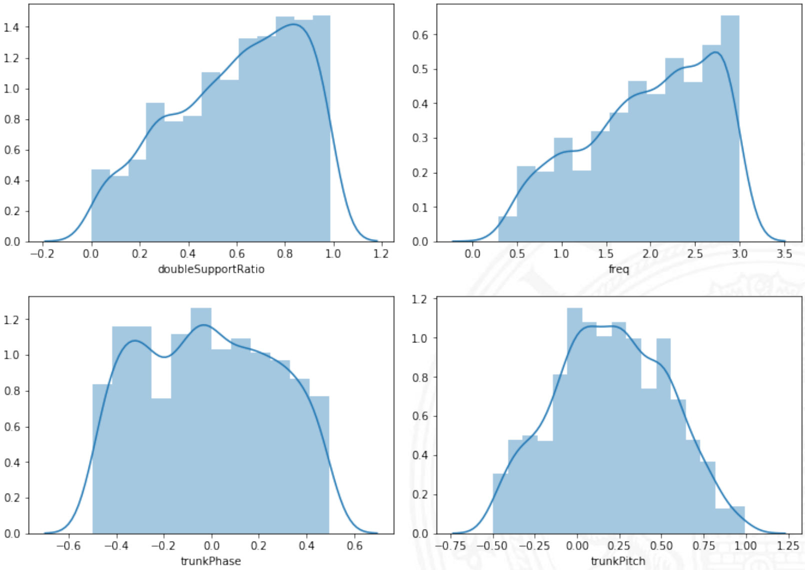




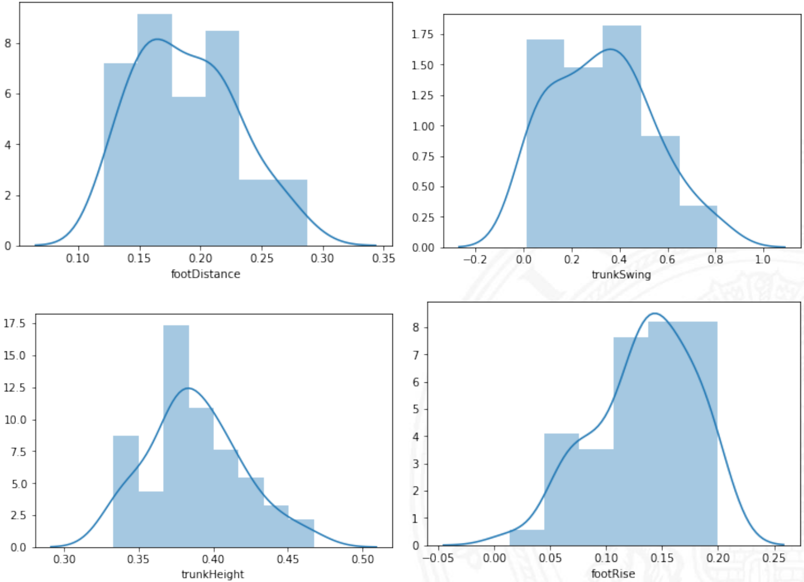
Results - Higher Zero



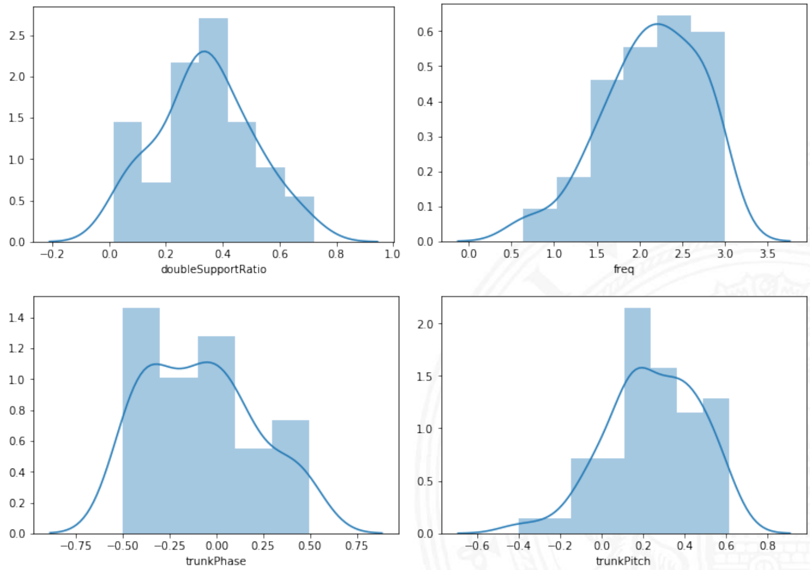
Results - Higher Zero



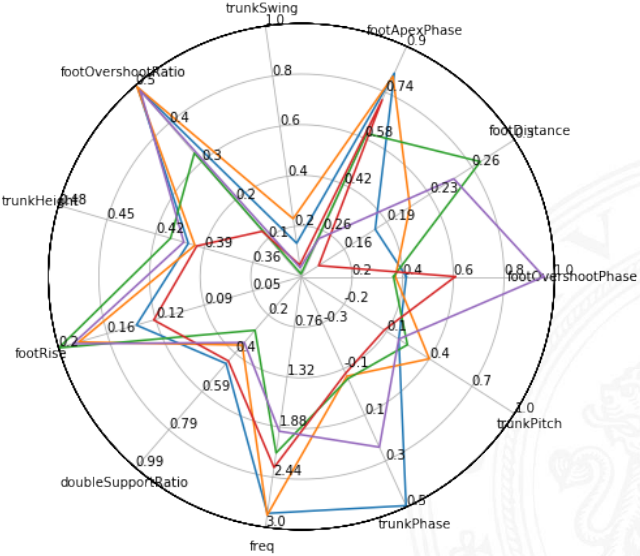
Results - Higher Six



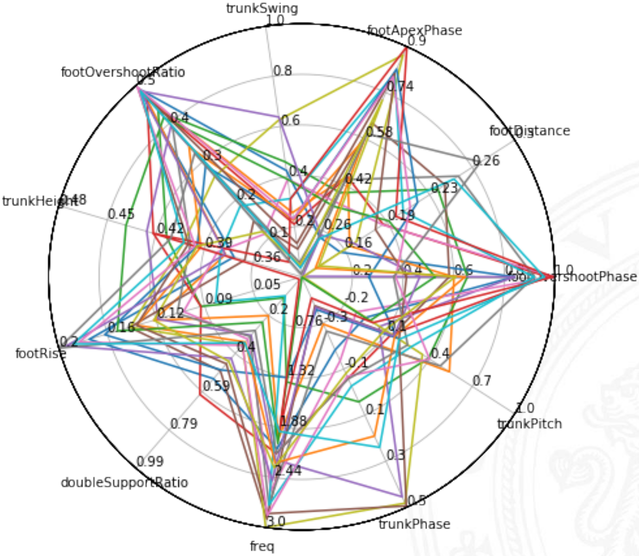
Results - Higher Six



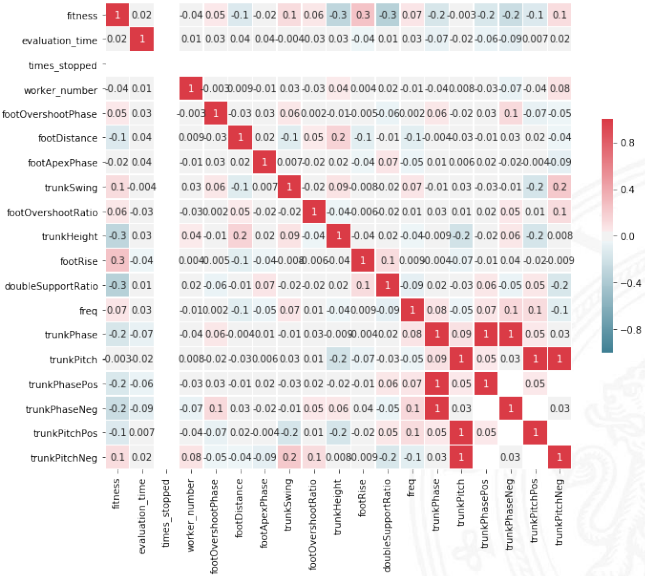
Results


trunkSwing
footApexPhase
footDistance
footOvershootPhase
trunkPitch
trunkPhase
freq
doubleSupportRatio
footRise
trunkHeight
footOvershootRatio

Results


trunkSwing
1.0
0.8
0.6
0.4
0.2
footApexPhase
0.9
0.74
0.58
0.42
0.26
footDistance
0.26
0.23
0.19
0.16
footOvershootPhase
1.0
0.6
0.4
0.2
trunkPitch
1.0
0.7
0.4
0.1
-0.2
trunkPhase
0.5
0.3
0.1
-0.1
-0.3
freq
3.0
2.44
1.88
1.32
0.76
doubleSupportRatio
0.99
0.79
0.59
0.4
0.2
footRise
0.2
0.16
0.12
0.09
0.05
trunkHeight
0.48
0.45
0.42
0.39
0.36
footOvershootRatio
0.5
0.4
0.3
0.2
0.1

Results - Parameter Pairwise Correlation



Conclusion

- Problem to easy
  - Introduce disturbances
  - More different speeds
  - Control from move_base
- Best parameters somewhat realistic
- Most parameters independent
  - Important to know which are not
  - Improvements on algorithm for more independence
- Dependency between parameter, cmd_vel and fitness
- Use another learning approach

Outline

Side Projects

- Besides the walking, I did some side projects
- I will only present them shortly
- Ask me if you want more information

Dynamic Stack Decider (DSD)

- Lightweight behavior framework
- Flixble like a behavior tree and simple like a FSM
- Based on building a stack of decisions and actions
- Concept developed in the RoboCup team in 2013
  - Used for body and head behavior
  - Previously called stack machine
- Improved version this year together with Martin Poppinga
- Was also used for the Tiago bartender challenge
- Paper for IROS workshop was rejected due to topic change
  - New paper will follow

Dynamixel ros_control

- Allows controlling of the Dynamixel servos via ros_control
- Same interface on robot as on Gazebo simulator
- Usage of standard controllers possible
- Implementation of current based position control
- Usage of sync read/write for optimal bus usage
- Generic implementation for any robot

Hardware Control Manager V2.0

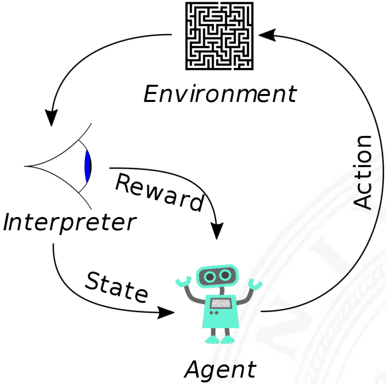
- Four functions
  - Regulate access to joint control
  - Implement basic (reflex like) behavior
  - Provide semantic status of the robot
  - Handle hardware problems
- Enabling easier writing of high level behavior
- Allows control of a humanoid robot like a wheeled one
- First version was done during my masters thesis
- Replaced old state machine with DSD
- Added further functionalities
  - Recognition of kidnapped robot
  - Better hardware error handling
- Paper in planning

Supervised Projects

- ImageTagger
  - Online platform for collaborative image labeling
  - Paper at RoboCup Symposium 2018, second author
- FCNN real-time ball localization
  - Improvement on previous approach
  - More data due to ImageTagger platform
  - Paper at RoboCup Symposium 2018, second author
- Particle filter world model
  - Filtering on relative FCNN heatmap
  - Submitted to IEEE MFI 2019, second author
- Speech recognition in the RoboCup domain
  - Bachelor thesis by Thomas Walther together with SP
  - Giving commands to a player in natural language
  - Training Kaldi model on specific trainer
  - Increasing robustness against noise
  - Not real time capable

Outline

Deep RL - Basic Idea



https://en.wikipedia.org/wiki/Reinforcement_learning

Current Baseline in Walking

- Good results in simulation
  - RoboSchool Flagrunn
  - RoboSchool Atlas
  - DeepMimic
- Not many applications on actual robots

Current RL Frameworks

- OpenAI gym interface
  - reset, step, render
- Environments
  - (Atari)
  - (Classic Control)
  - (Box2D)
  - Mujoco
  - RoboSchool (Bullet)
  - PyBullet
  - openai_ros (the Construct)
- Baselines
  - OpenAI baselines
  - INRIA Flowers stable-baselines

Available Baseline Algorithms

- A2C
- ACER
- ACKTR
- DDPG
- DQN
- GAIL
- HER
- PPO
- TRPO

Deep RL on Wolfgang

- How can we apply simulator results on Wolfgang?
- Input
  - Command velocity
  - IMU
  - Feet pressure sensors
  - Phase
  - Current feet poses / velocities
  - Joint efforts
- Output
  - Goal poses of feet
- Some parts of the robot are not actuated
  - Head moves independently to look at the world
  - Arms should stay in a safe position

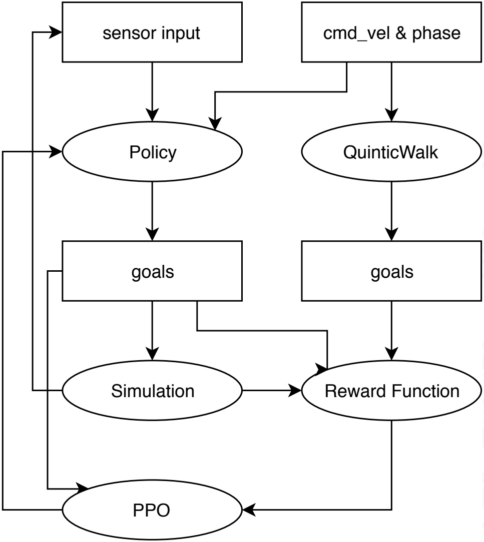
Parts Needed

- Robot ✓
- Robot model ✓
- Training environment (✓)
  - Roboschool URDF
  - Maybe training in multiple simulators
- Learn algorithm (✓)
  - PPO2 from stable-baselines
- Policy network
  - Simple fully connected should work
  - Central Pattern Generators could improve results
- Reward function
  - Next slide
- Real world training/evaluation
  - HCM to track falling and stand up again
  - Use april tag + camera to get odometry error
  - Give commands so that robot does not run into walls

Reward Function

- Very crucial to shape what the robot is learning
- Goals
  - Stability > speed
  - Usability in real world
  - Odometry error not to big
- Use "mocap" data from QuinticWalk
  - Better transfer than human mocap data
  - Robot has only 20DOF
  - Similar reward term as in DeepMimic
- Punishing term for large error in odometry
- Punish falling a lot

Learning Overview



From Simulation to Reality

- Main challenge: get it to work in real world
- Evaluate difference between simulator and real robot (BA)
- Initial state randomization
- Physic parameter randomization
- Learning in different simulators
- Introducing disturbances
- Adding noise to simulated sensor values

Future Work

- Stand up
- Small kick while walking
- Omnidirectional kick engine

References

Questions?