An Introduction to Team ViGIR's Open Source Software and DRC Post Mortem

TECHNISCHE UNIVERSITÄT DARMSTADT

Stefan Kohlbrecher, Alberto Romay, Alexander Stumpf, Achim Stein, Oskar von Stryk

Simulation, Systems Optimization and Robotics (SIM) Group, TU Darmstadt, Germany

Philipp Schillinger

Robert Bosch GmbH, Corporate Research, Department for Cognitive Systems. Stuttgart, Germany

Spyros Maniatopoulos, Hadas Kress-Grazit

Verifiable Robotics Research Group, School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, USA

David C. Conner

Christopher Newport University, Newport News, United States

Outline

- Intro
  - The DRC
  - Team ViGIR
- (ROS based) Infrastructure
- System Overview
- DRC Finals
- Lessons Learned

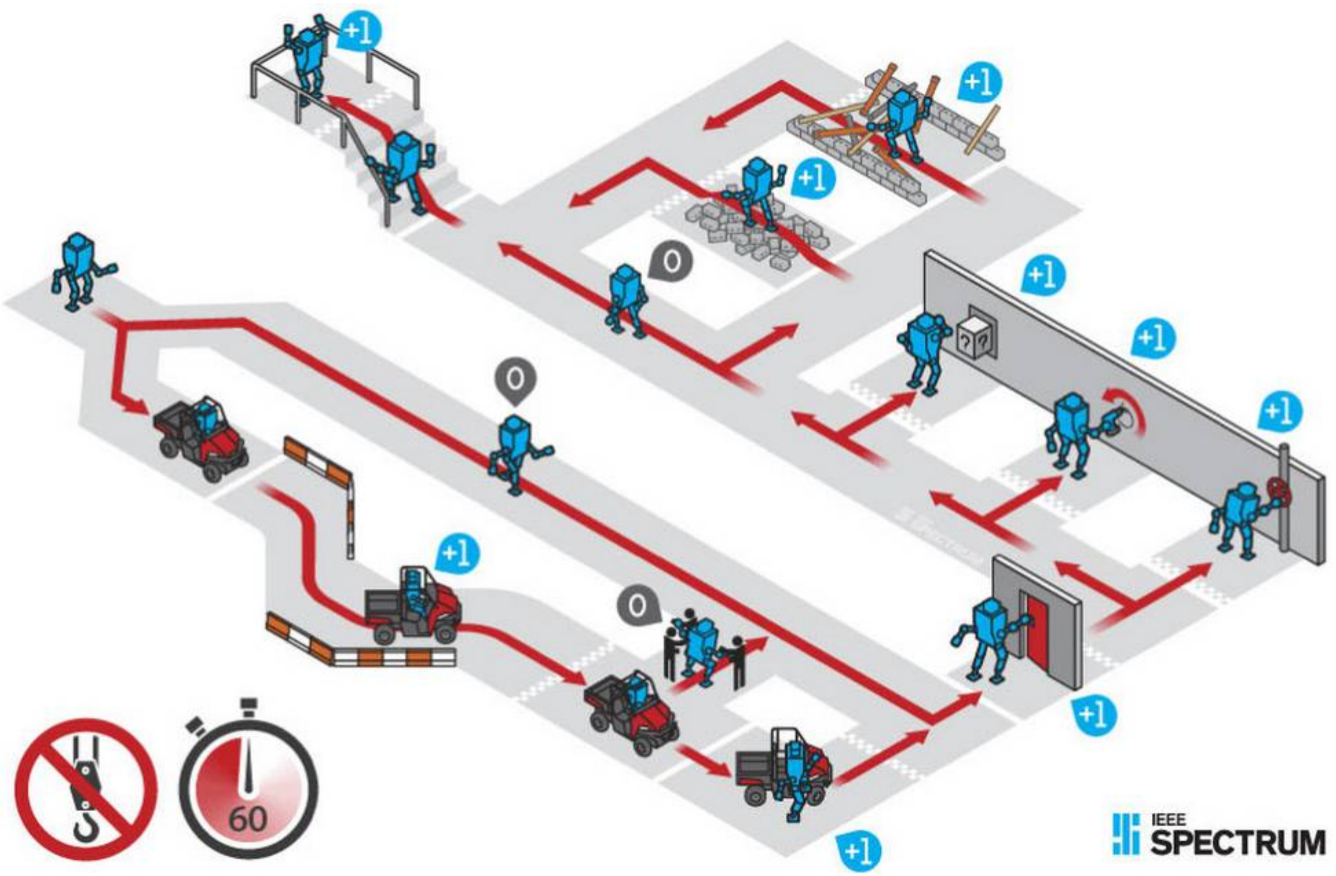
The DARPA Robotics Challenge

TECHNISCHE UNIVERSITÄT DARMSTADT

...close study of the disaster's first 24 hours, before the cascade of failures carried reactor 1 beyond any hope of salvation, reveals clear inflection points where minor differences would have prevented events from spiraling out of control.

IEEE Spectrum, November 2011 pg. 36. (online version)

DRC – Tasks and Rules



IEEE Spectrum

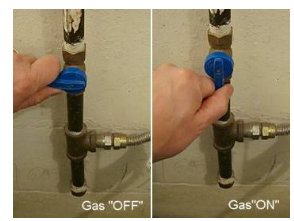
The “C” in DRC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Uneven terrain, stairs and ladders
- Motions with multiple contacts (e.g. getting out of a vehicle)

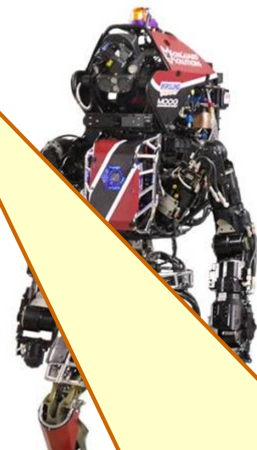
Versatile and robust

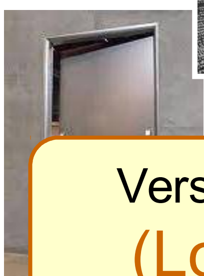
(Loco-)Motion



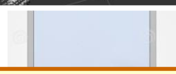
The “C” in DRC

Versatile and robust
Perception





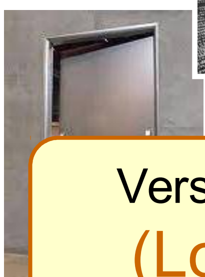
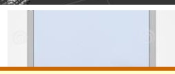
Versatile and robust
(Loco-)Motion



- Perceive Environment for locomotion
- Perceive objects for manipulation
- Ability to acquire new objects and their potential purposes on the fly
- Robustness to different lightning conditions

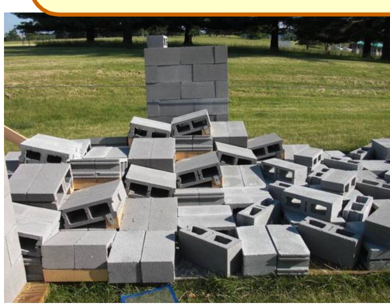
The "C" in DRC

TECHNISCHE UNIVERSITÄT DARMSTADT

Versatile and robust
Perception

Versatile and robust
(Loco-)Motion

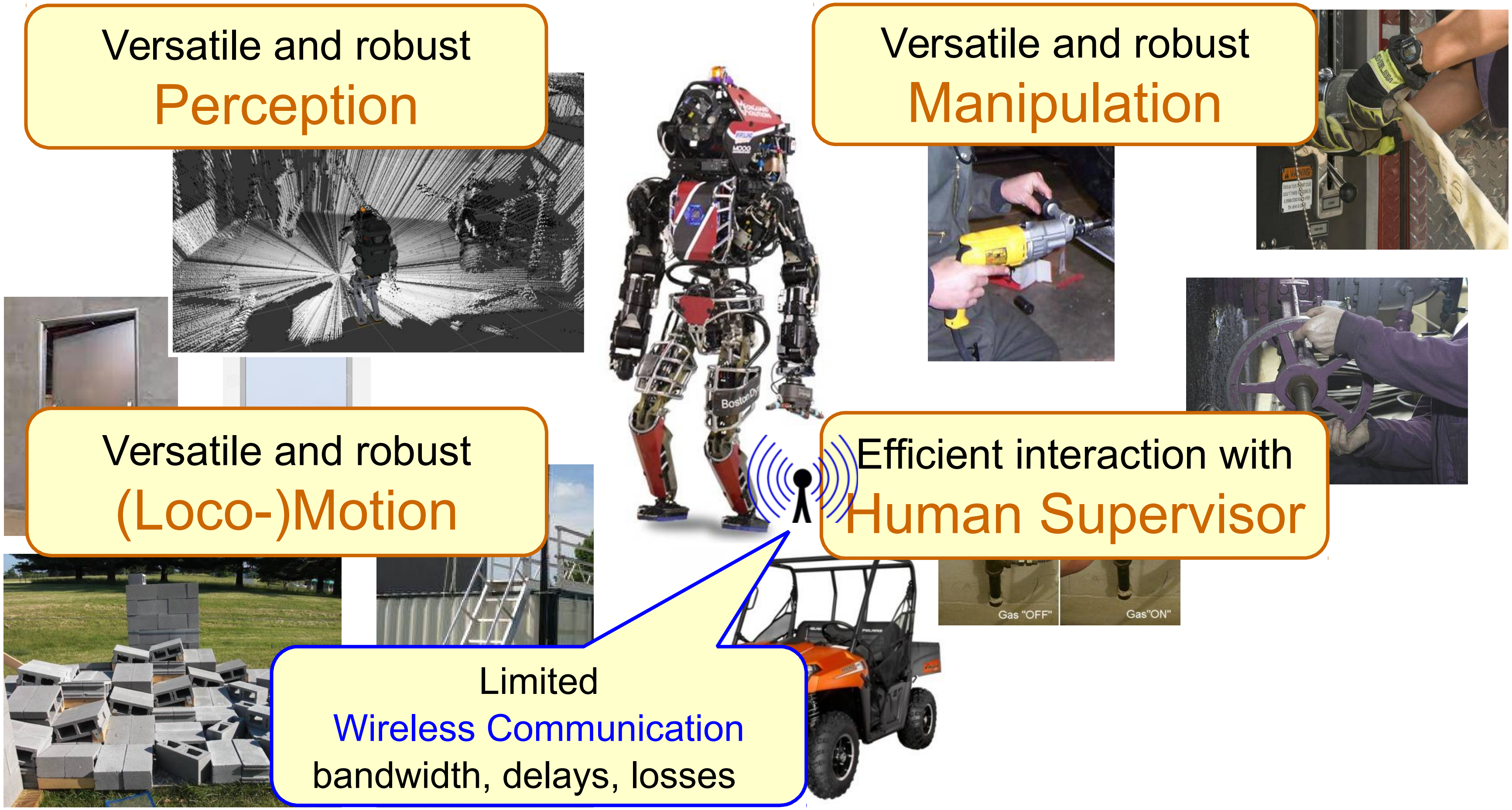



Versatile and robust
Manipulation



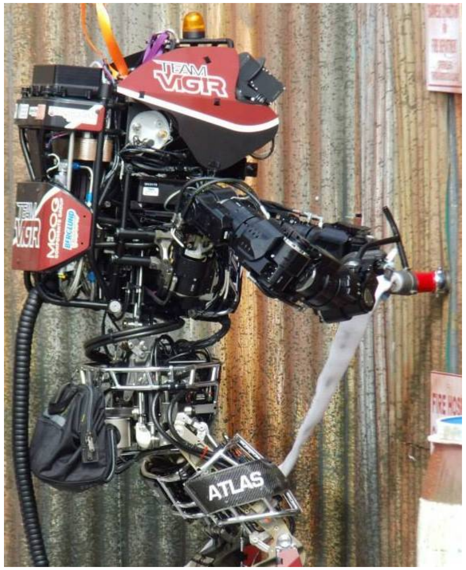
- Many different tools, only few exactly known in advance
- Acquiring new manipulation modes
- Ability to coordinate manipulation, locomotion & active perception

The “C” in DRC


Versatile and robust
Perception
Versatile and robust
Manipulation
Versatile and robust
(Loco-)Motion
Efficient interaction with
Human Supervisor
Gas "OFF"
Gas"ON"
Limited
Wireless Communication
bandwidth, delays, losses

DRC - The Meta-Challenges

- Highly compressed timeline
- Multiple competition events
  - VRC (June 2013)
  - Trials (Dec 2013)
  - Finals (June 2015)
- Systems integration
- High reliabilty
  - Only few attempts at tasks



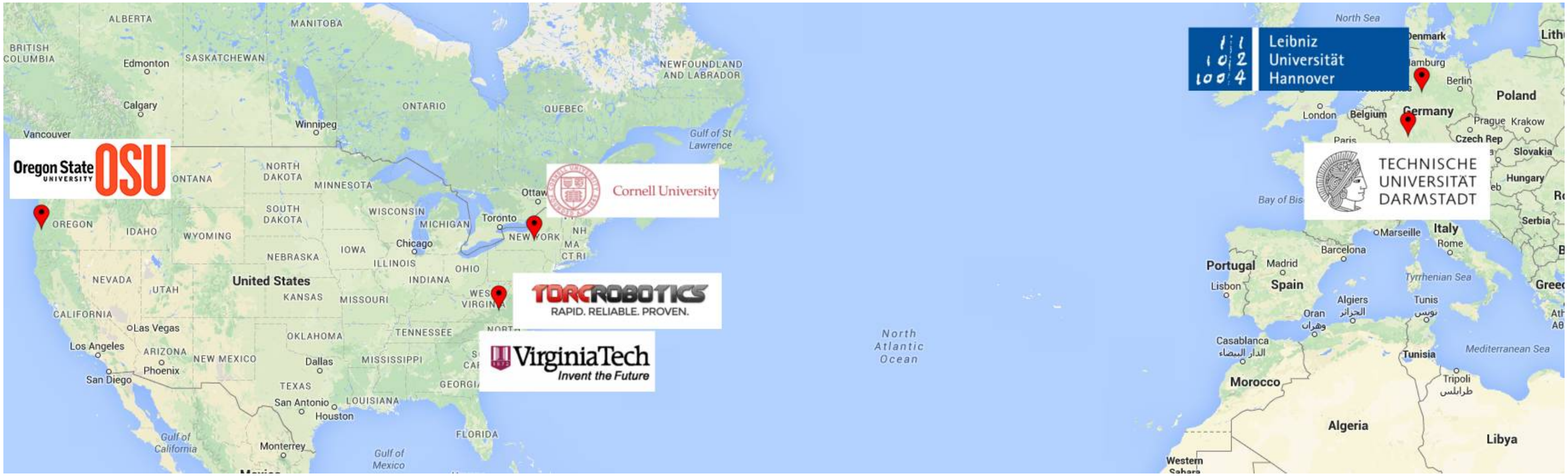
Team ViGIR

International collaboration, Track B Atlas team. VIrginia Germany Interdisciplinary Robotics

- TORC Robotics (Blacksburg, VA)
- TU Darmstadt (Darmstadt, Germany)
- Virginia Tech (Blacksburg, VA)
- Cornell University (Ithaca, NY)
- Leibnitz Universität Hannover (Hannover, Germany)
- Oregon State University (Corvallis, OR)

Team ViGIR

TECHNISCHE UNIVERSITÄT DARMSTADT



Team ViGIR

TECHNISCHE UNIVERSITÄT DARMSTADT

- Track B team, DRC participation from day one
  - Virtual Robotics Challenge (VRC)
  - DRC Trials
  - DRC Finals
- Software available: github.com/team-vigir
  - Exceptions:
    - Robot controller
    - Comms bridge
- Other teams using ViGIR software at DRC Finals
  - HECTOR (SIM, TU Darmstadt)
  - VALOR (TREC, Virginia Tech)

Open Source Efforts by other DRC competitors

TECHNISCHE UNIVERSITÄT DARMSTADT

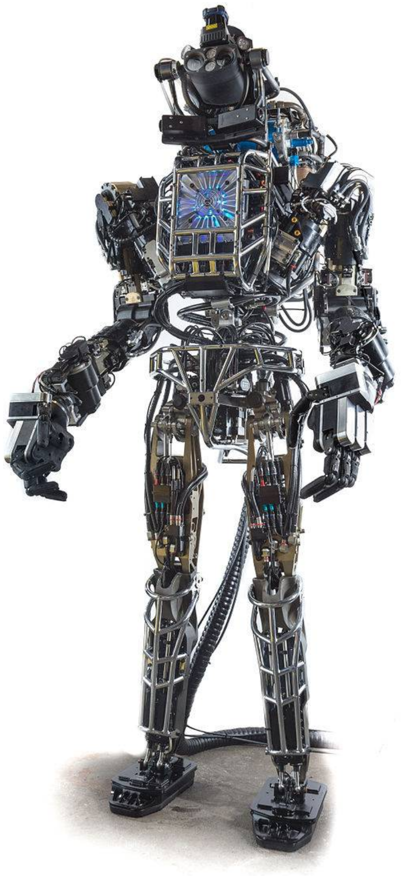
- MIT:
  - Pronto State Estimator (pronto-distro github)
  - Drake Planning and Control (drake github)
  - Director UI (director github)
- IHMC:
  - IHMC Controller
  - SCS Simulator

  (ihmc_ros bitbucket)
- JSK:
  - Extensive ROS-based Software (jsk-ros-pkgs github)

Hardware

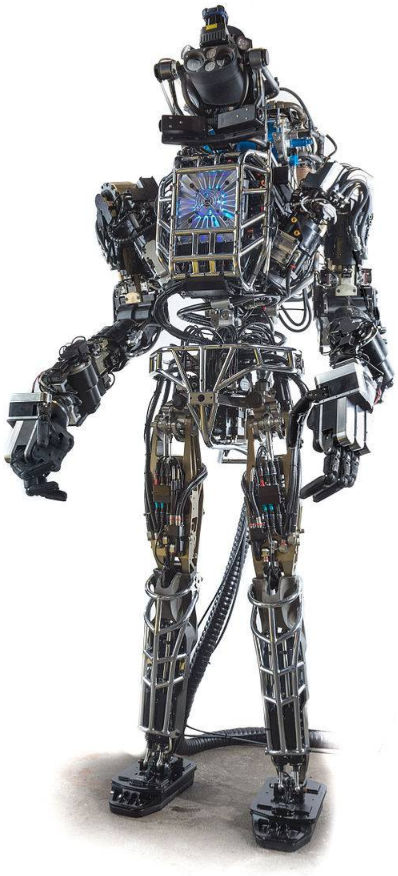
TECHNISCHE UNIVERSITÄT DARMSTADT

- Boston Dynamics (BDI) Atlas robot
  - Hydraulically actuated
- Our Atlas nicknamed "Florian" (after patron saint of firefighters)
- API provided by BDI
  - Walking/Stepping
  - Balancing
- → Upper body planning decoupled from low level balance control



Hardware – Atlas Versions

- Atlas V3 (2013–Nov 2014)
  - Tethered
  - 6DOF arms



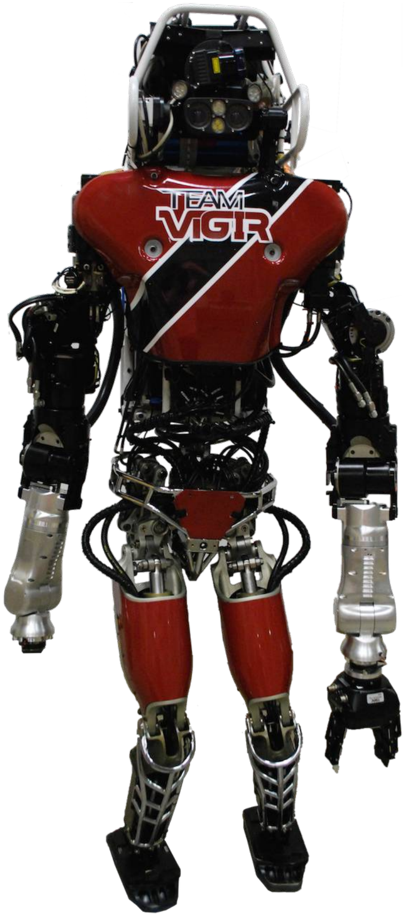
Hardware – Atlas Versions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Atlas V3 (2013–Nov 2014)
  - Tethered
  - 6DOF arms
- Atlas V4 (Feb 2015-Mar 2015)
  - Untethered
  - Onboard Computing
  - 6DOF arms

Hardware – Atlas Versions

- Atlas V3 (2013–Nov 2014)
  - Tethered
  - 6DOF arms
- Atlas V4 (Feb 2015-Mar 2015)
  - Untethered
  - Onboard Computing
  - 6DOF arms
- Atlas V5 (Apr 2015-Aug 2015)
  - As V4, but 7DOF arms (lower 3 joints electric)



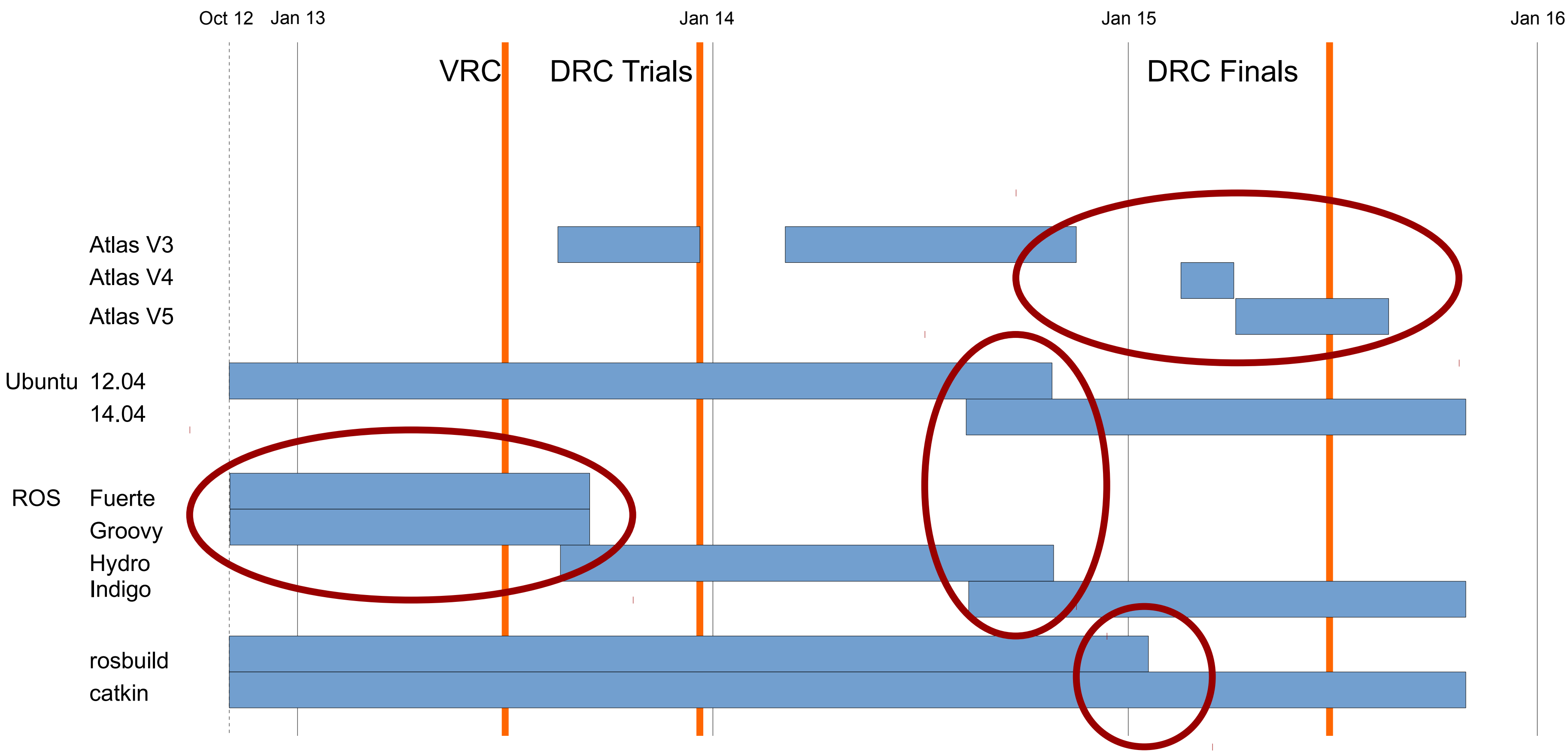
Infrastructure

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use of ROS from the beginning
  - Prior experience
  - Great community
  - A lot of useful software
  - Integration with DRCsim
- Private git(lab) repos
  - Now moved to github
- Project management via Redmine
  - Every task in issue tracker
  - Hundreds of Wiki-pages

Timeline with a Focus on Infrastructure

TECHNISCHE UNIVERSITÄT DARMSTADT



Infrastructure – Managing Robot Variability

- Many variants:
  - 3+ Atlas versions
  - 4 Hand types
- Could use args/params
  - Unwieldy to forward through launch files
- Use environments variables
- Generate robot model (and onboard software setup) at launch-time

```
filename="$(find atlas_description)/urdf/$(optenv VIGIR_ATLAS_ROBOT_TYPE atlas_v5)_simple_shapes.urdf"/>
filename="$(find atlas_description)/urdf/$(optenv VIGIR_ATLAS_ROBOT_TYPE atlas_v5).$(optenv VIGIR_SIM_TYPE
filename="$(find atlas_description)/urdf/$(optenv VIGIR_ATLAS_ROBOT_TYPE atlas_v5).transmission" />

filename="$(find atlas_description)/robots/multisense/$(optenv VIGIR_ATLAS_MULTISENSE_TYPE sim)_multisense

filename="$(find atlas_description)/robots/hands/$(optenv VIGIR_ATLAS_LEFT_HAND_TYPE l_stump).urdf.xacro" ,
filename="$(find atlas_description)/robots/hands/$(optenv VIGIR_ATLAS_RIGHT_HAND_TYPE r_stump).urdf.xacro"
```

github.com/team-vigir/vigir_atlas_common/blob/master/atlas_description/robots/vigir_atlas.urdf.xacro

Infrastructure – Deployment to multiple machines

- Complex system
  - 3 Onboard Computers
  - 1 Field Computer
  - 4 OCS Computers
- Fast development cycles
  - Build and deploy quickly and consistently
- Remotelaunch scripts
  - Build using catkin (install)
  - Deploy using rsync
  - Start using ssh/screen

github.com/team-vigir/remotelaunch

Infrastructure – Deployment to multiple machines

TECHNISCHE UNIVERSITÄT DARMSTADT

- 01_theoden_robot.sh
- 02_boromir_robot.sh
- 03_eowyn_robot.sh
- 04_legolas_robot.sh
- 05_pippin_robot.sh
- env.sh

Launch scripts for each machine

```
#theoden
roslaunch vigir_atlas_bringup  common_parameters.launch
roslaunch vigir_atlas_controller atlas_robot.launch
roslaunch pgr_camera sa_cameras.launch
```

Common environment setup executed on each machine

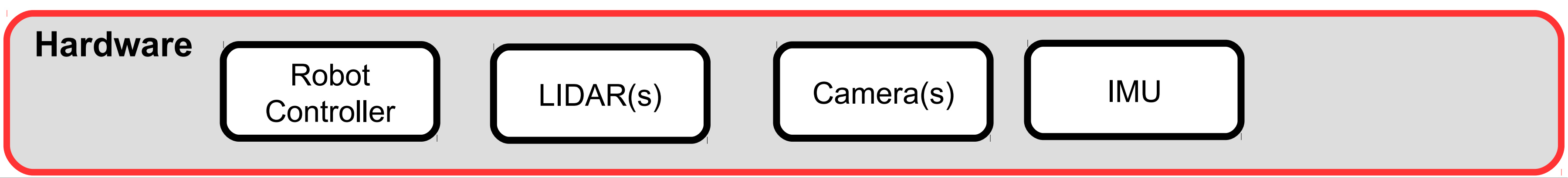
```
#!/bin/bash
# This code will be run in every screen on the remote machine. It is typically
# used to source ros and setup any other environment variables you need.
cmdline=("$@")
if [ $# == 0 ]; then
    cmdline=($SHELL -i)
fi

export VIGIR_ROOT_DIR=/home/user/vigir_repo
echo "sourcing catkin_ws................"
source "/home/user/vigir_repo/catkin_ws/install/setup.bash"
echo "sourcing scripts setup.bash.................."
source "/home/user/vigir_repo/scripts/setup/setup.bash"
shopt -s expand_aliases

exec "${cmdline[@]}"
```

Infrastructure – Simulation Options

- No single solution that can do everything currently available (as open source)
  - IHMC controller/Atlas with Gazebo integration to be released this fall

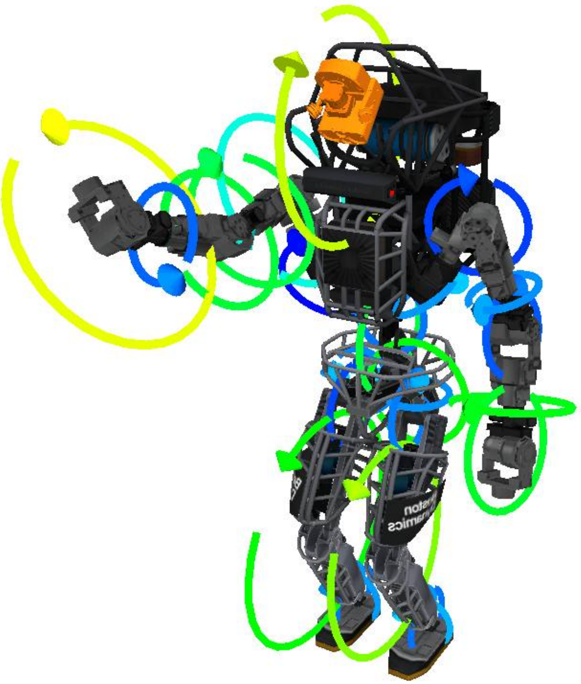
Simulator/Robot	Locomotion	Manipulation	Remarks
Atlas/BDI/DRCsim	(Yes)	No	Only with BDI lib
Atlas/IHMC/SCS	Yes	No	
Atlas/IHMC/DRCsim	(Yes)	(Yes)	Coming soon :)
Valkyrie/IHMC/DRCsim	(Yes)	(Yes)	Coming soon
Thor-Mang/Gazebo4	No	Yes	

Components - Controls

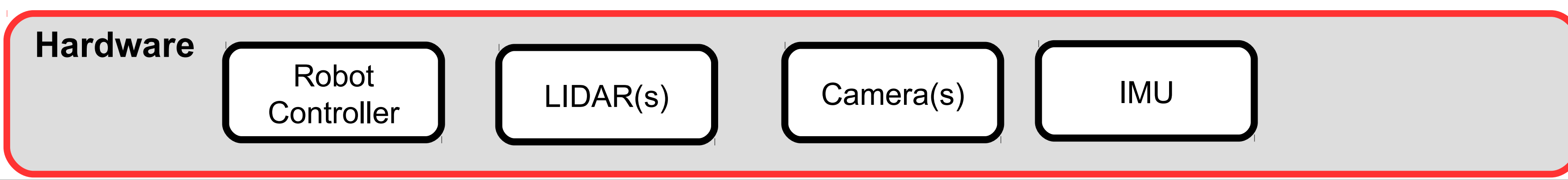


Controls

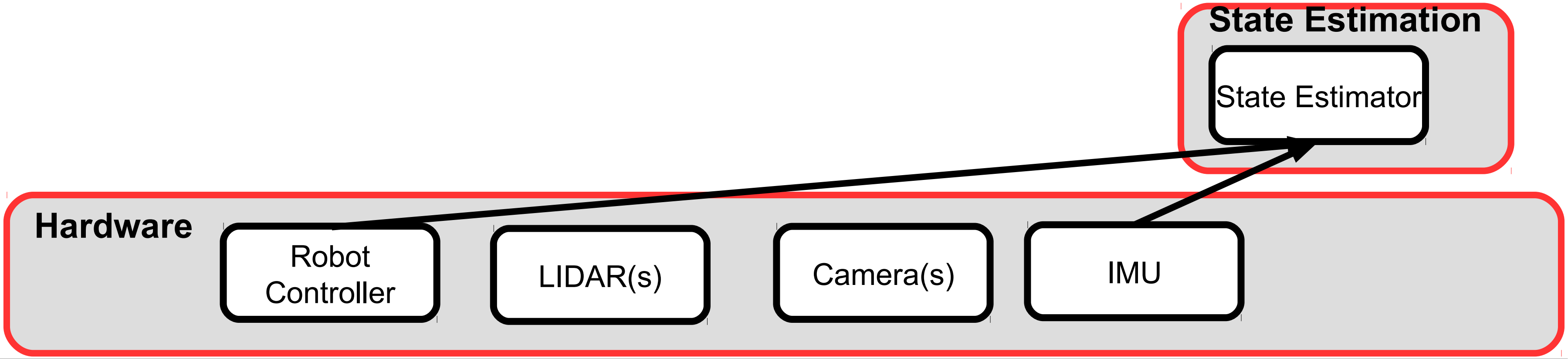
- Use of BDI supplied library
  - Walk (dynamic stability)
  - Step (static stability)
  - Manipulate (balance while standing)
- Provided as binary
  - Black box, no source (also for DRC teams)
  - Not available to general public :(
- Effort to integrate IHMC Whole Body controller
  - Use in competition prevented by time constraints/delays
  - Coming soon



Components – State Estimation



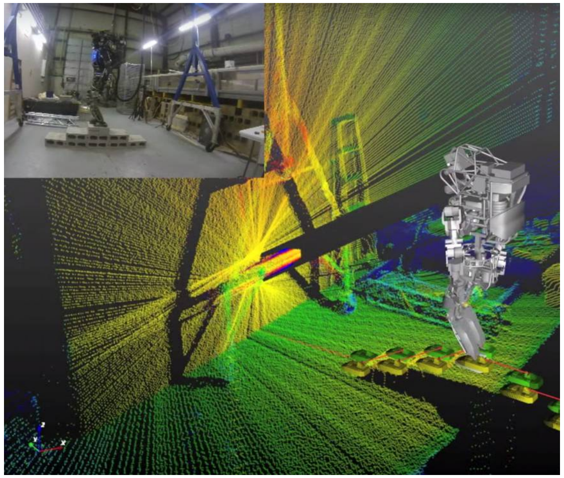
Components – State Estimation



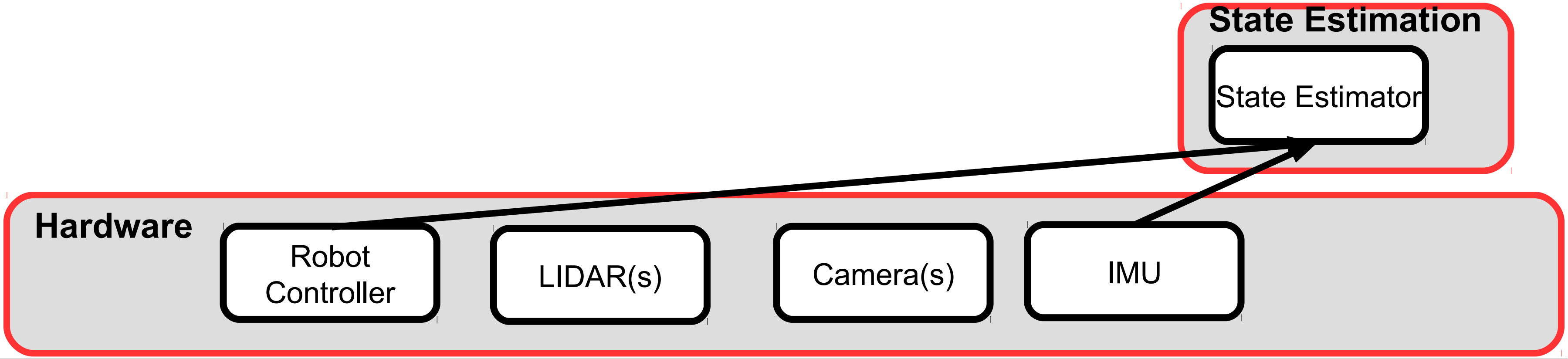
State Estimation

- Provide state (pose) estimate for robot
- Fuse
  - Leg Kinematics
  - IMU
- Continuous but drifting estimate
  - Low drift with good sensors
- Use MIT's pronto
  - Tuned for Atlas system
  - pod build system
  - LCM communications
- LIDAR use dangerous in non-static environment

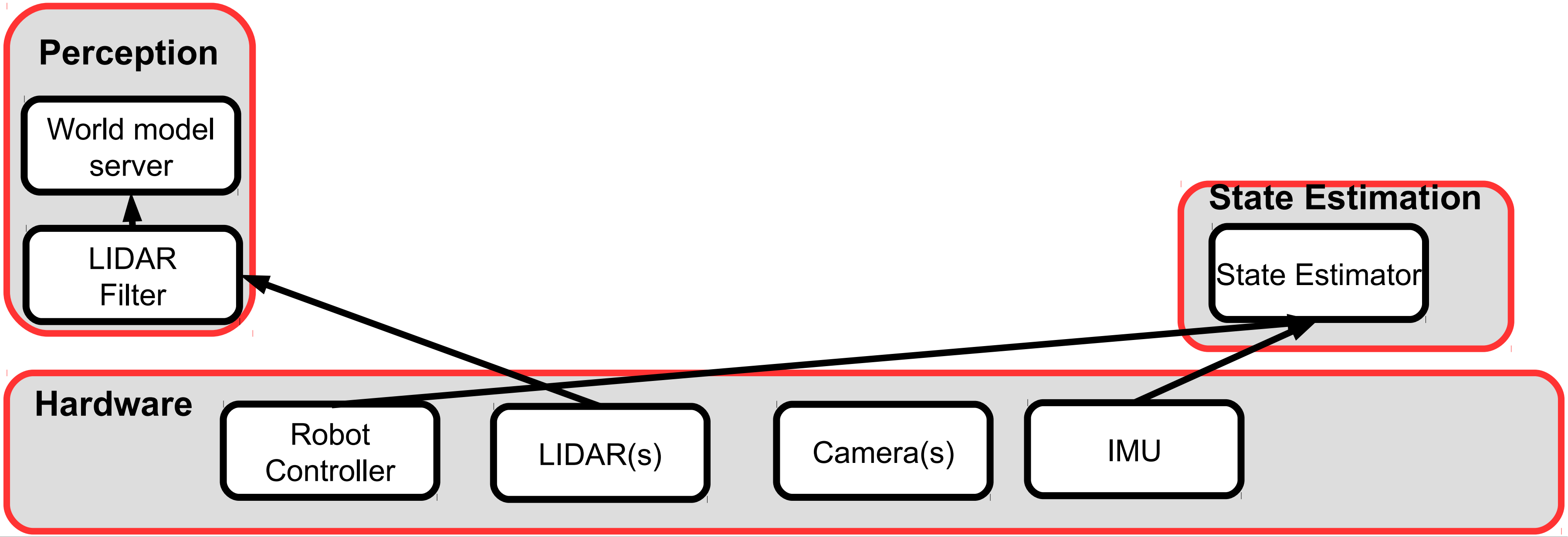
pronto-distro (ViGIR fork)



Components – Perception

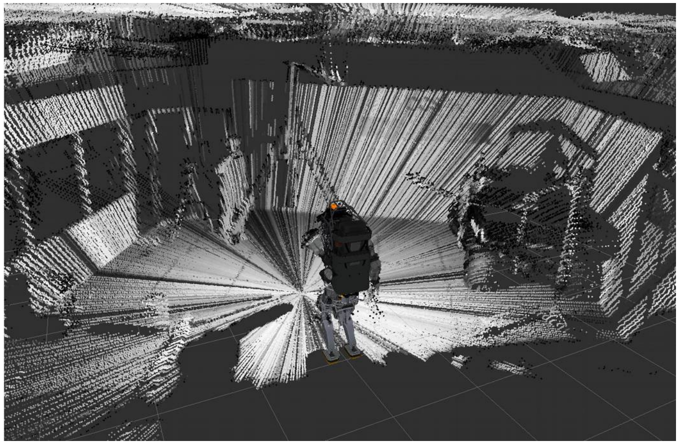


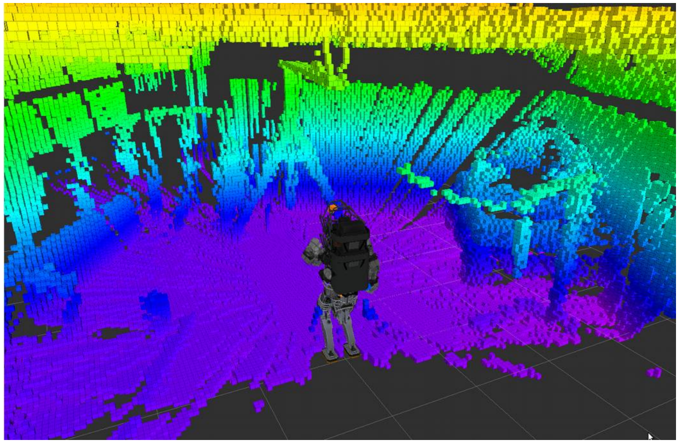
Components - Perception



Perception

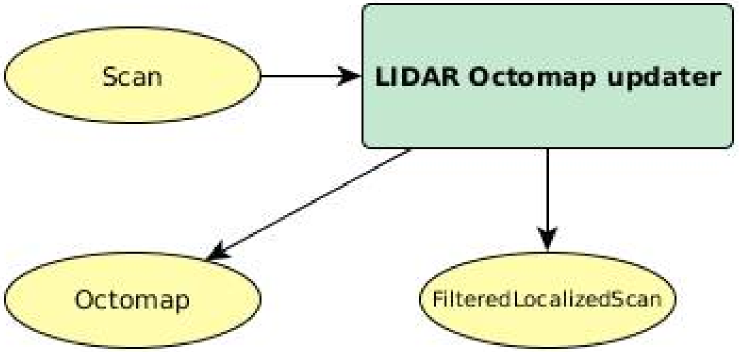
- Provide situational awareness for operator(s)
- Provide world state estimate for robot
  - Footstep planning
  - Manipulation





Perception - LidarOctomapUpdater

- Environment octomap updated in real-time
- Provide collision model for planner
- Also provide filtered LIDAR data for overall system
  - Annotate with transform information as tf prohibitive over constrained comms



github.com/team-vigir/vigir_manipulation_planning/tree/master/vigir_lidar_octomap_updater

Perception – Compressing LIDAR Data

TECHNISCHE UNIVERSITÄT DARMSTADT

- Standard scan too big for 1500 Byte UDP limit

github.com/team-vigir/vigir_perception/tree/master/vigir_filtered_localized_scan_utils

Perception – Compressing LIDAR Data

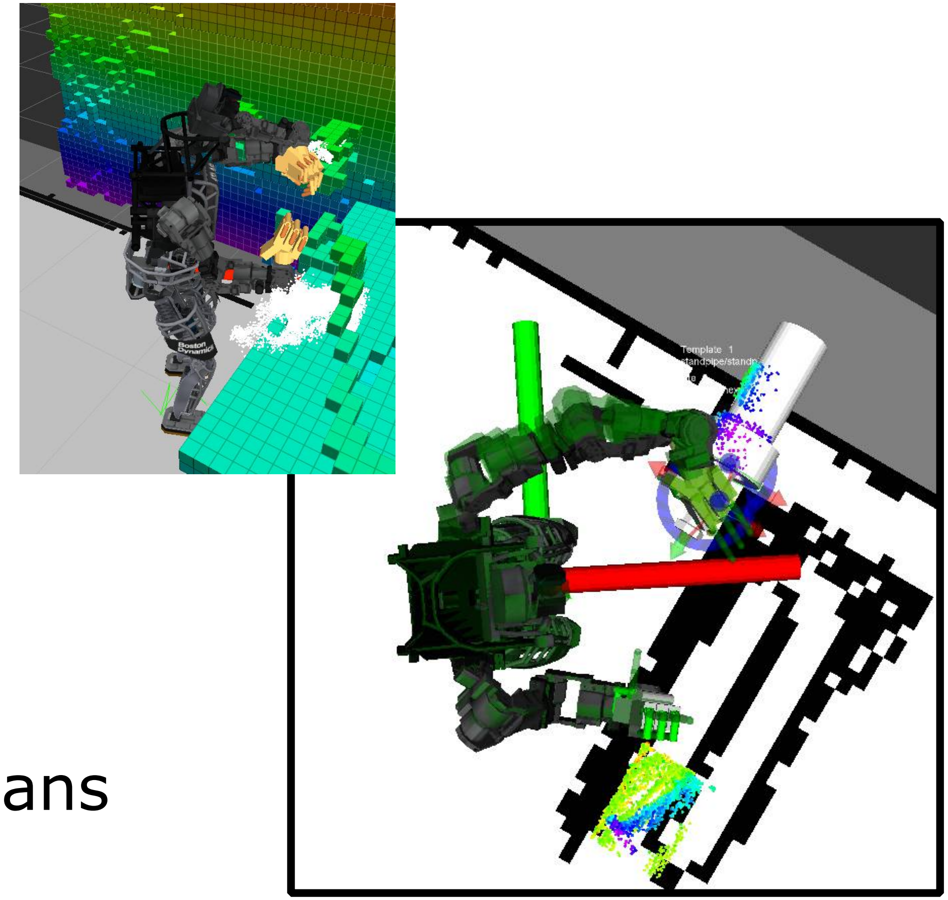
TECHNISCHE UNIVERSITÄT DARMSTADT

- Standard scan too big for 1500 Byte UDP limit
- Compress:
  - Split (3 separate scans)
  - Distances to uint_16
  - Intensities to uint_8
  - Self filter bit
- Add start/end global transform info
- Can reconstruct on OCS side
  - Every compressed scan usable standalone

github.com/team-vigir/vigir_perception/tree/master/vigir_filtered_localized_scan_utils

Perception – World Model Server

- Collect LIDAR data
- Provide services
  - Pointcloud ROIs
  - Octomap ROIs
  - Gridmap slice ROIs
  - Distance queries
- Two instances
  - Onboard
  - OCS
- Sync via compressed scans



github.com/team-vigir/vigir_perception/tree/master/vigir_worldmodel_server

Situational Awareness using Fisheye Cameras

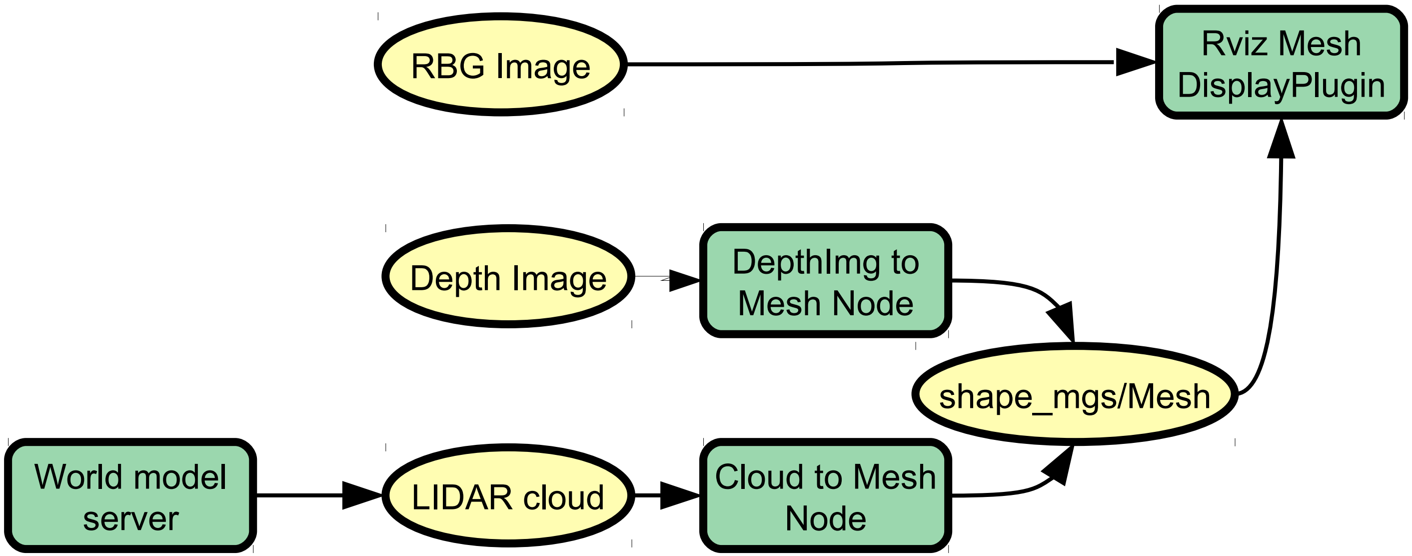
TECHNISCHE UNIVERSITÄT DARMSTADT

- Fisheye cameras provide high FOV
- Hard to interpret for humans
- Calibrate Fisheye cam using the ocamlib toolbox
- Virtual pinhole camera that follows tf frames

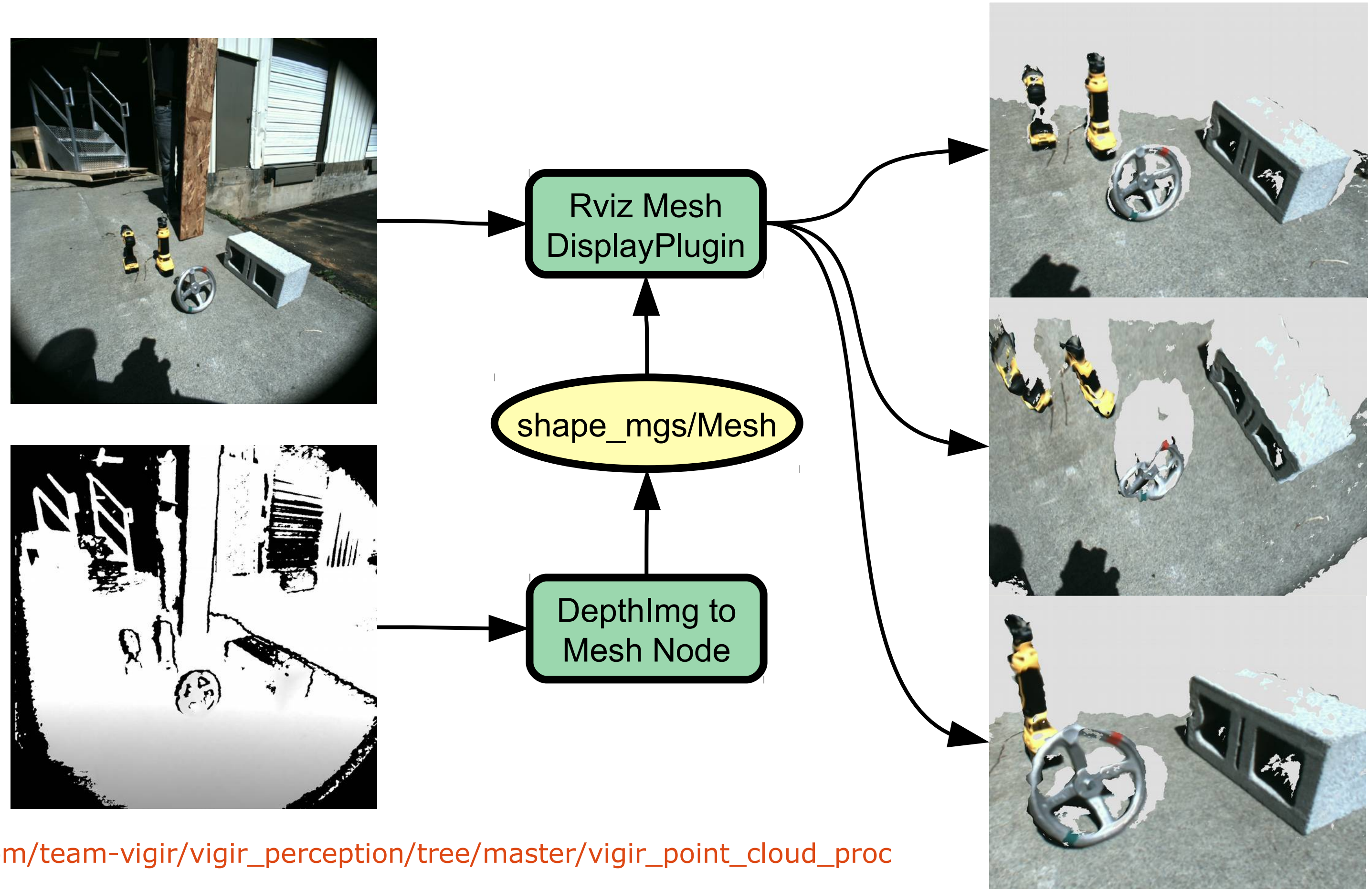
github.com/team-vigir/vigir_wide_angle_image_proc

Mesh Visualization

- Latest image data texture mapped onto mesh
  - Depth image-based: Fast update rate, low range
  - LIDAR-based: Low update rate, high range



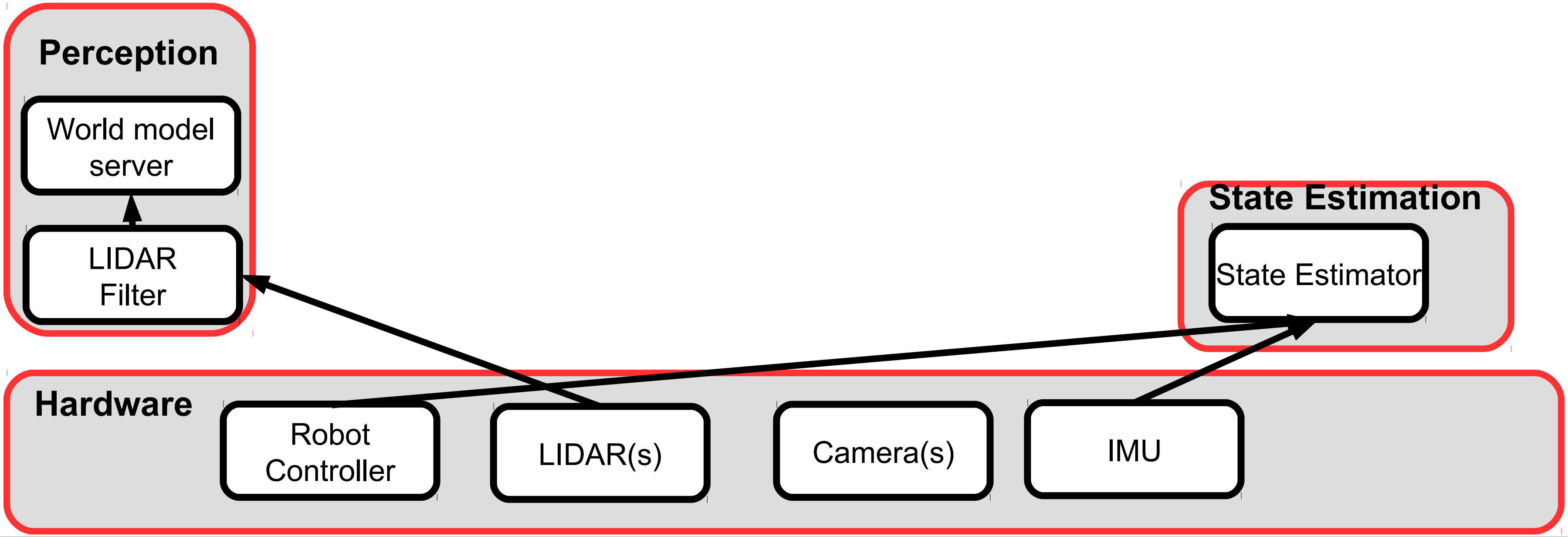
Mesh Visualization



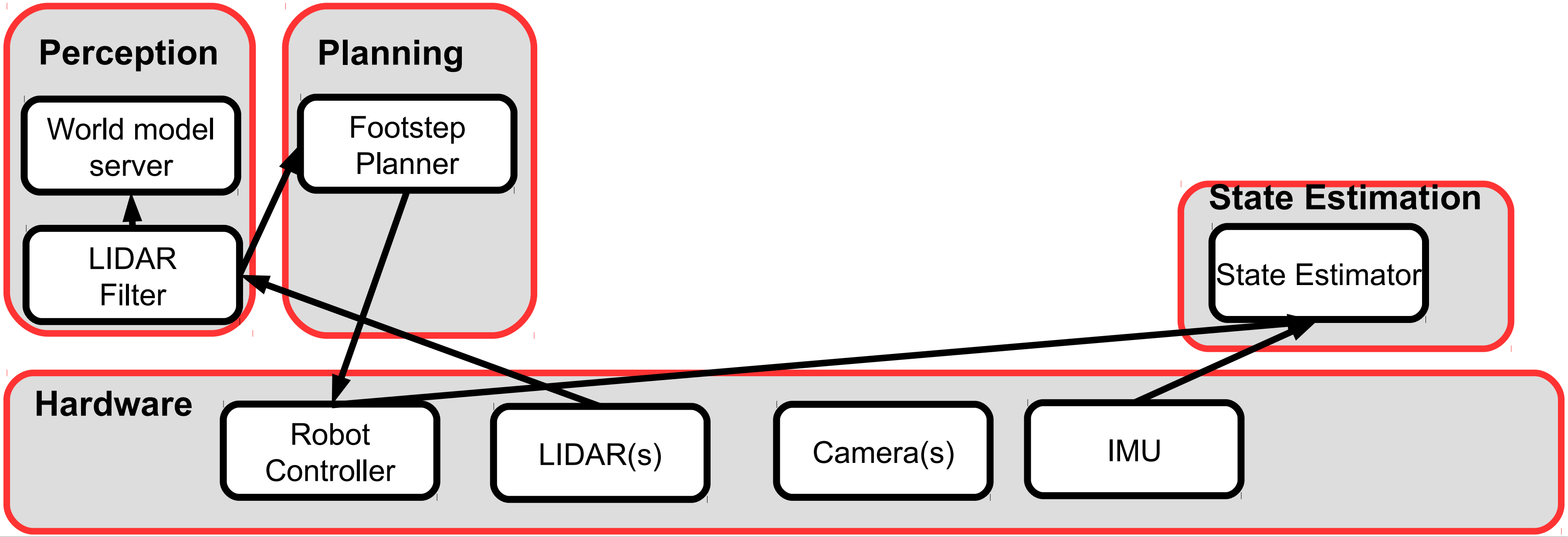
github.com/team-vigir/vigir_perception/tree/master/vigir_point_cloud_proc

github.com/team-vigir/vigir_ocs_common/tree/master/vigir_ocs_rviz_plugins/vigir_ocs_rviz_plugin_mesh_display_custom

Components – Footstep Planner

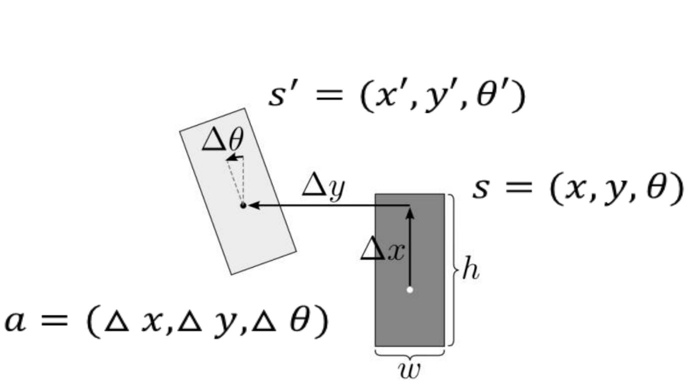


Components – Footstep Planner

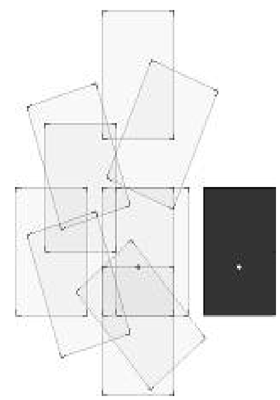


Footstep Planner

- Based on work by Hornung et. al. [1]
  - A*-search-based planning approach



Discrete Foot Placements



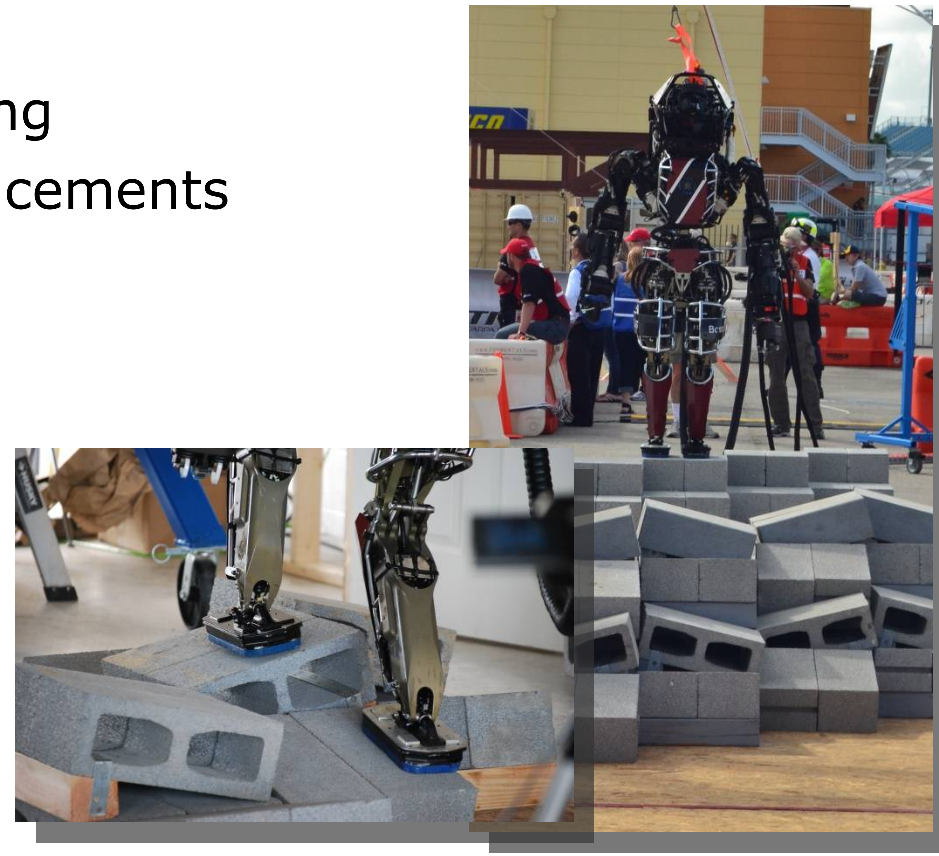
Successor Set

[1] Hornung et.al. Anytime Search-Based Footstep Planning with Suboptimality Bounds, Humanoids 2012

Footstep Planner

TECHNISCHE UNIVERSITÄT DARMSTADT

- Complex Locomotion:
  - 3D perception and modeling
  - Safe sequences of foot placements
  - 6DOF foot placements
  - Obstacle avoidance
  - Balance control
- Divide and conquer
  - Terrain Model Generator
  - 3D Footstep Planning
  - Robot Controller



github.com/team-vigir/vigir_footstep_planning_core

Terrain Model Generator

TECHNISCHE UNIVERSITÄT DARMSTADT

- Only point clouds required
  - Octree as back-end
  - Incremental updates
  - Stand-alone ROS package
    - Usable in other domains

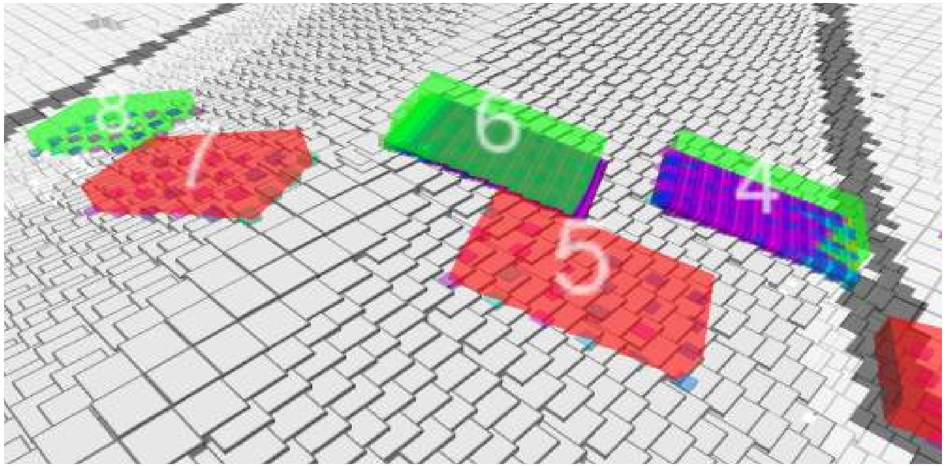
github.com/team-vigir/vigir_terrain_classifier

Terrain Model Generator

- Online Generation
  - Surface Normals (left)
  - Height Map (right)

Footstep Planner: 3D Planning

- Extension to 3D

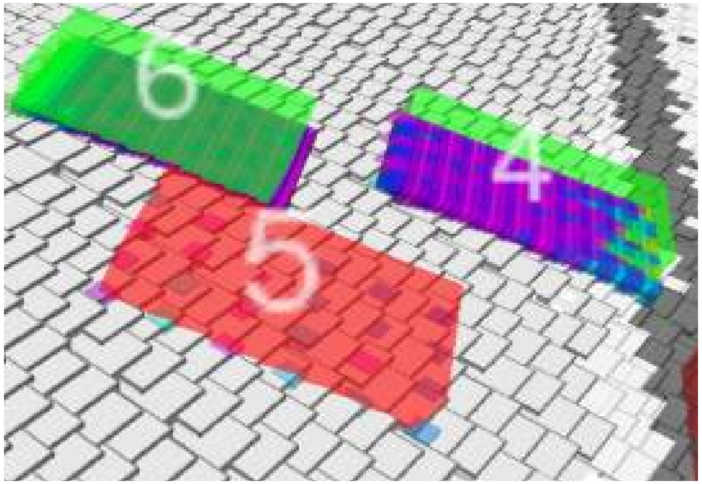


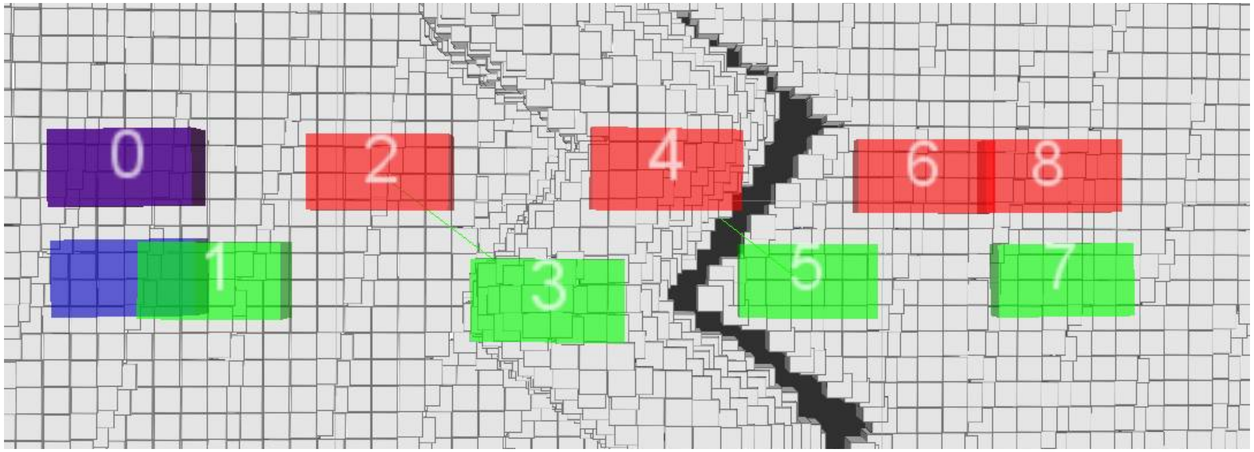
- States: Become full 6 DOF
- Actions: Remain the same
- Roll, pitch and step height are constrained by underlying terrain
- Search space does not enlarge
- No expensive branching tree!

Footstep Planner: 3D Planning

TECHNISCHE
UNIVERSITÄT
DARMSTADT

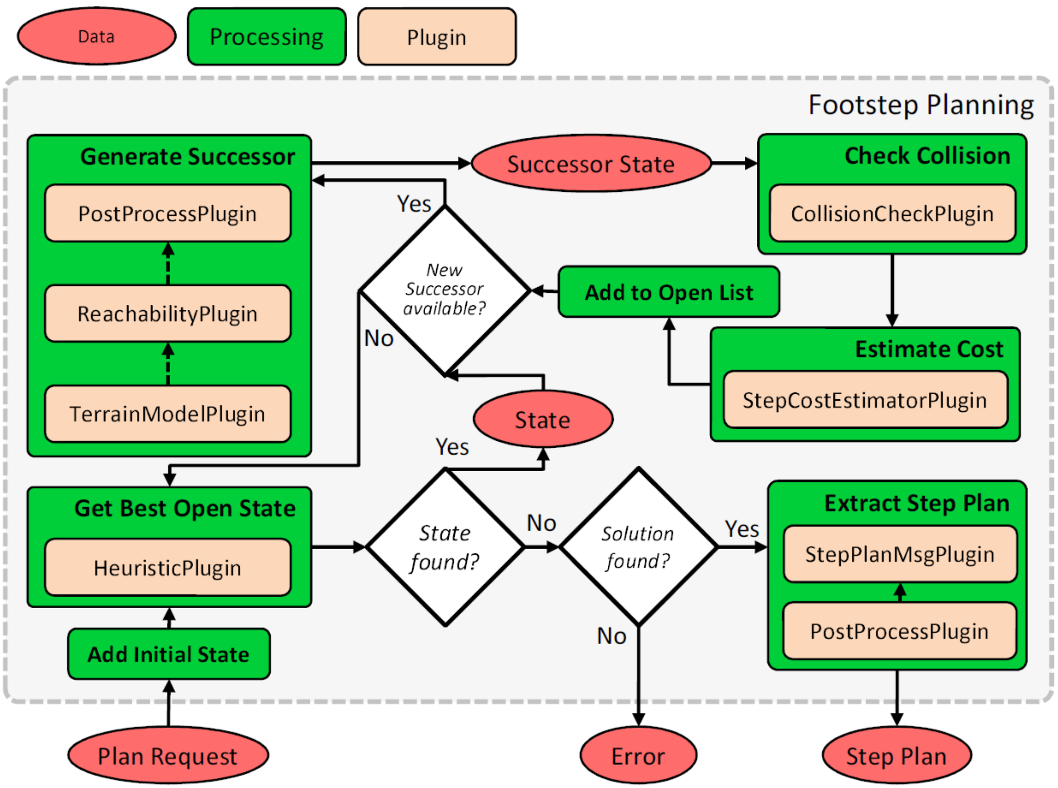
- Ground contact estimation
  - Sampling of foot surface
  - Estimate contact situation of each sample using height map
  - More flexible collision checking model
  - Allows overhanging steps



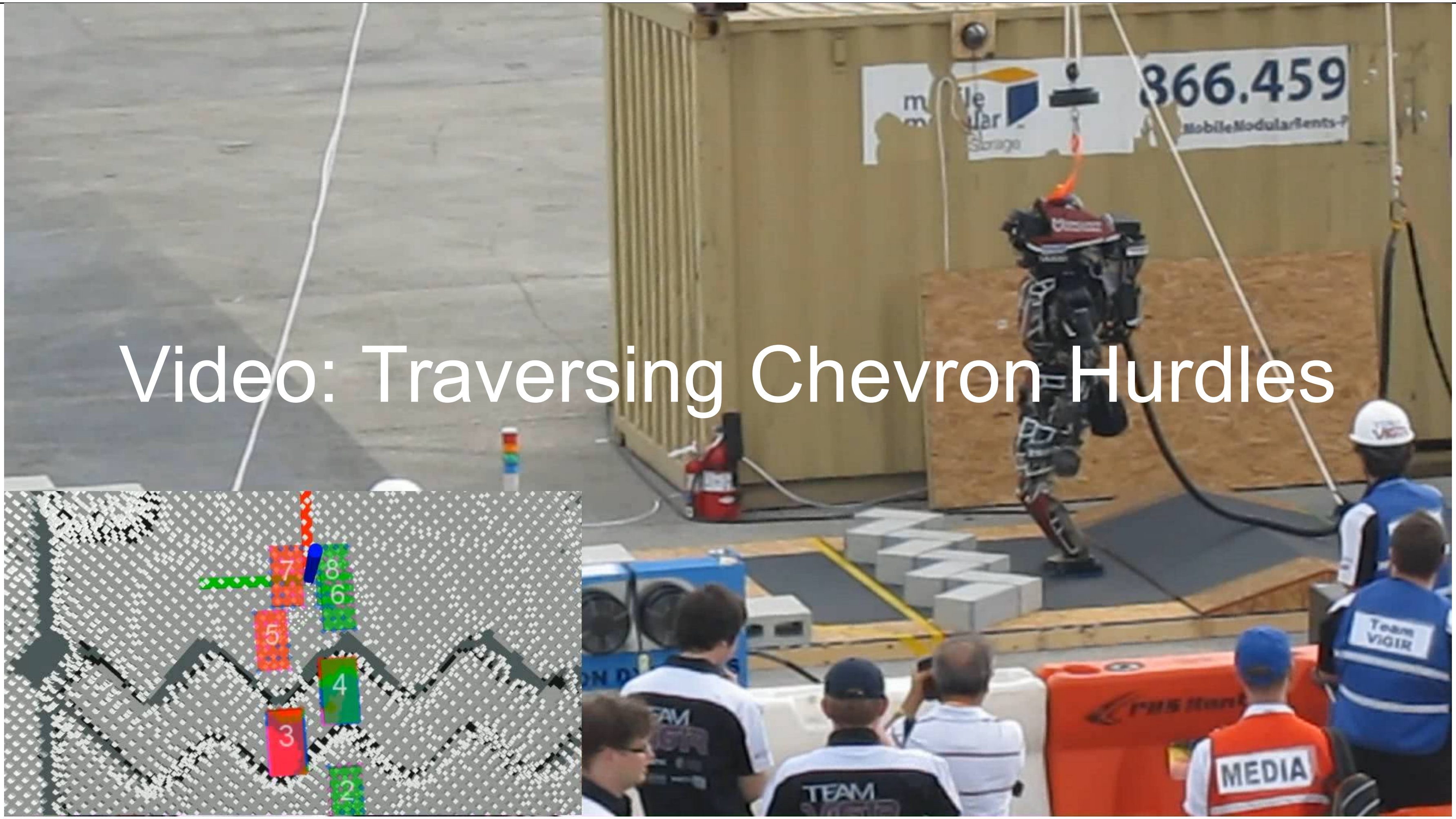


Footstep Planner: Plugins

- Plugins used for customization of all relevant system aspects
- Setups for 3 robots already available:
  - Atlas
  - Thor-Mang
  - ESCHER

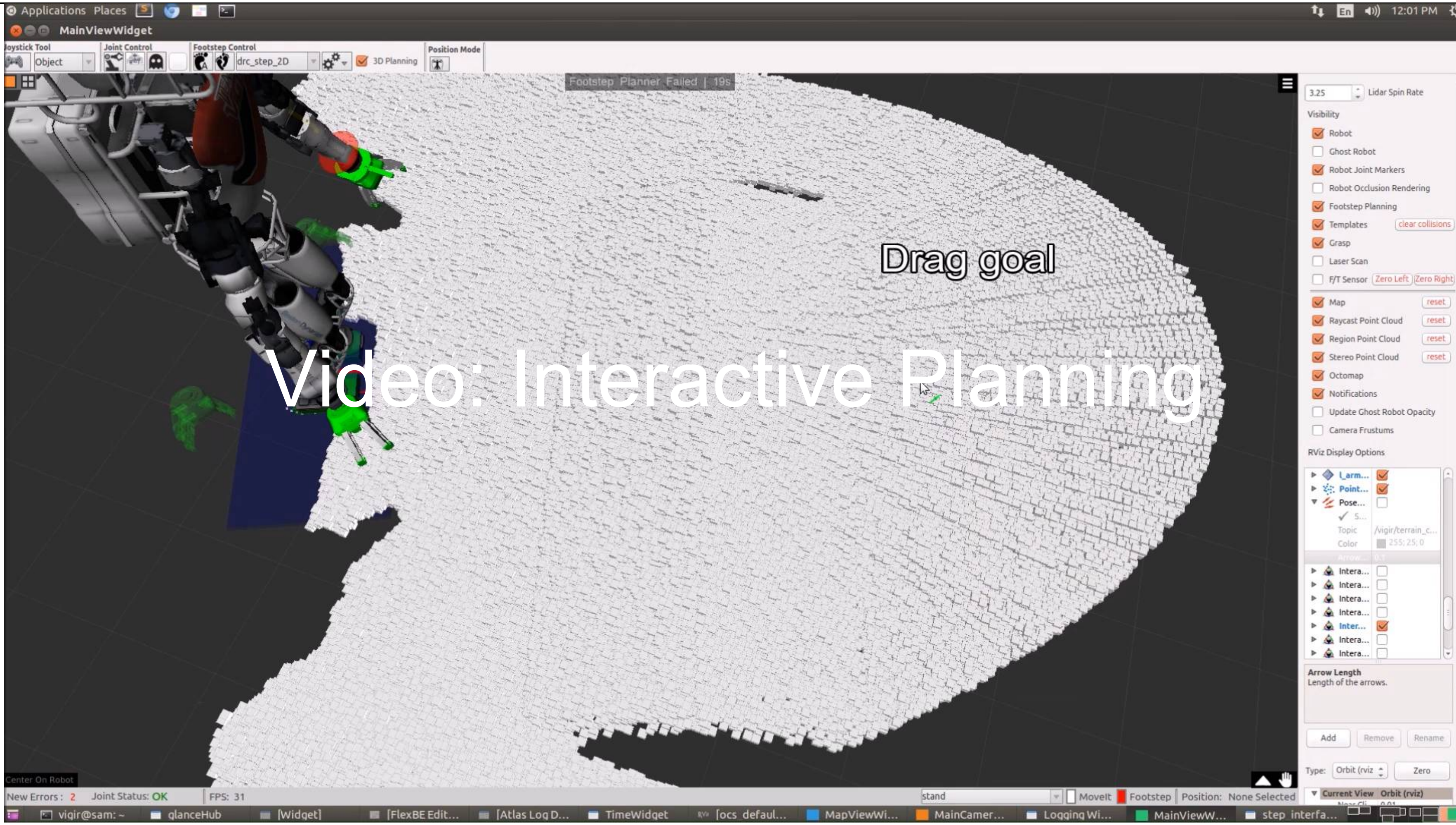


Footstep Planner: Example



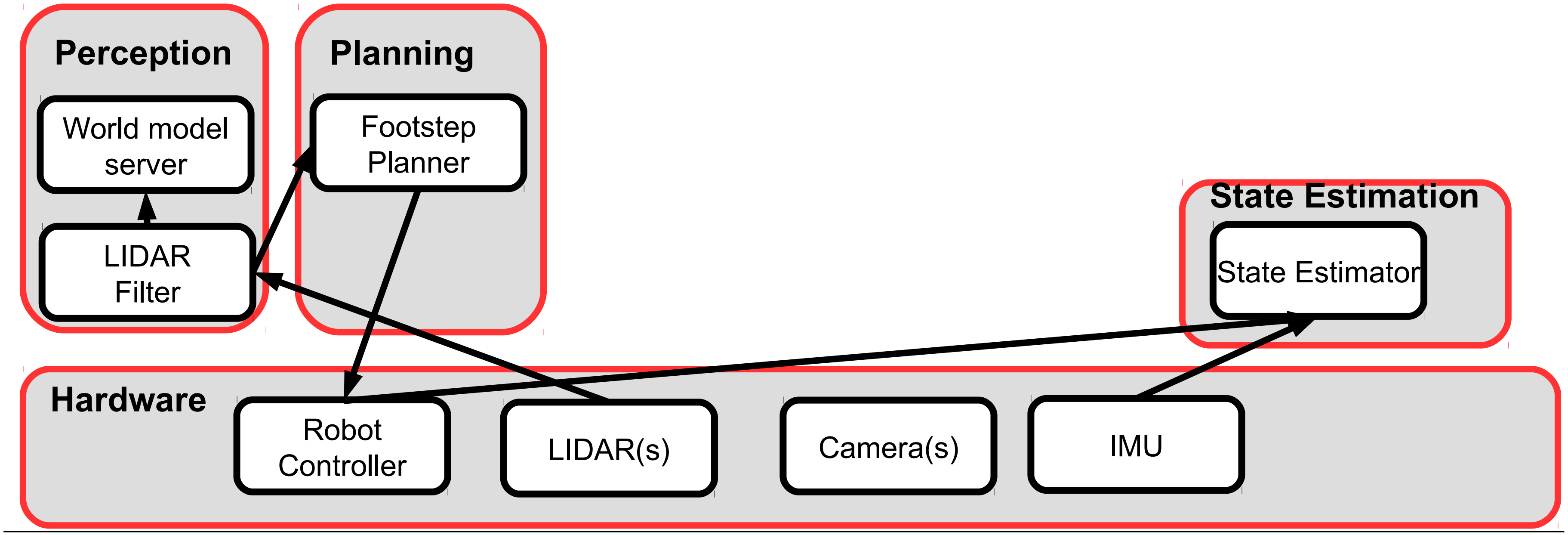
Footstep Planner: Interactivity

TECHNISCHE UNIVERSITÄT DARMSTADT

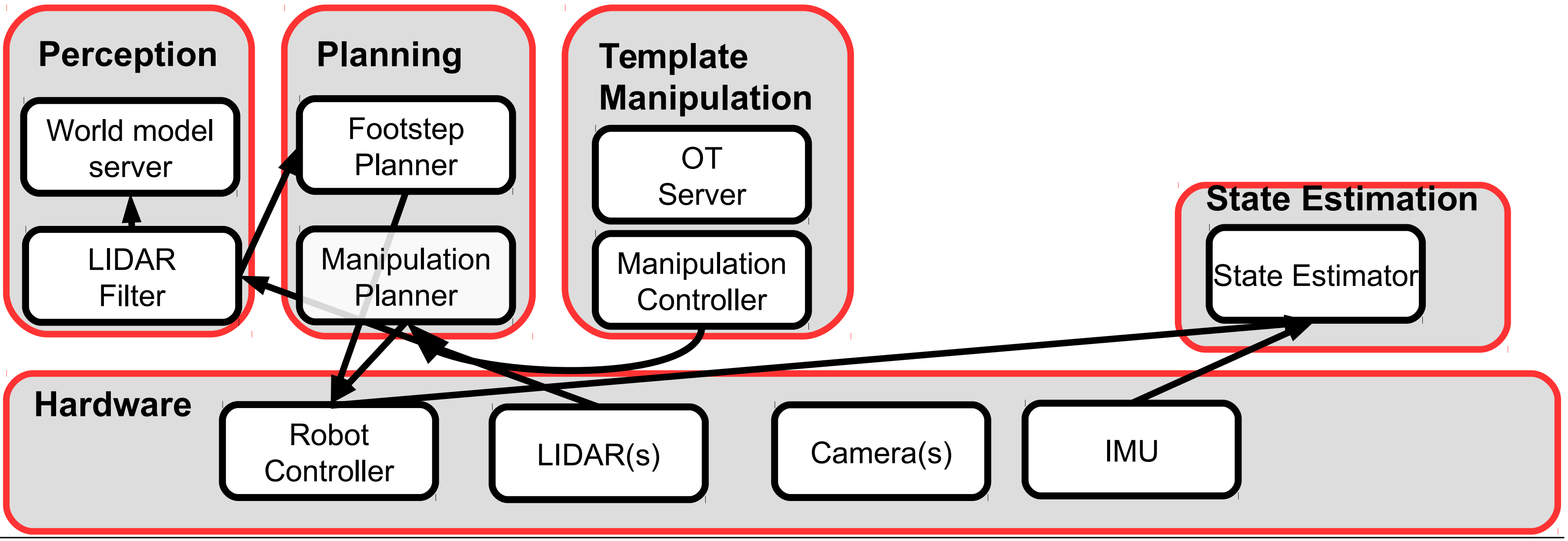


VIGIR SIM

Components – Footstep Planner



Components - Manipulation



Motion Planning - Requirements

TECHNISCHE UNIVERSITÄT DARMSTADT

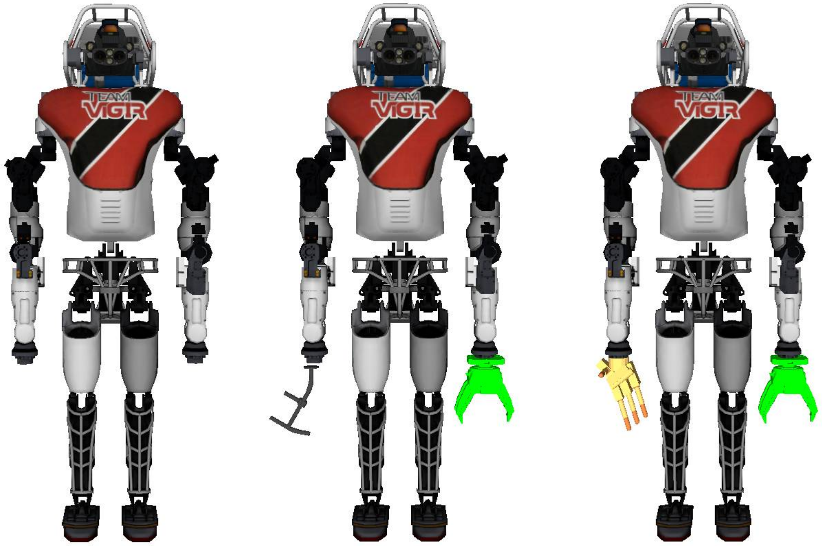
- Manipulation
  - Collision free planning
  - Cartesian Paths
  - Manipulation in contact with environment
  - Maintain stability
- Sliding Autonomy:
  - Operator/OCS-based (Teleop)
  - Operator/Object template based (Task level)
  - Behavior Executive (Autonomous)

→ Use MoveIt! as back-end

Motion Planning – Robot Setup

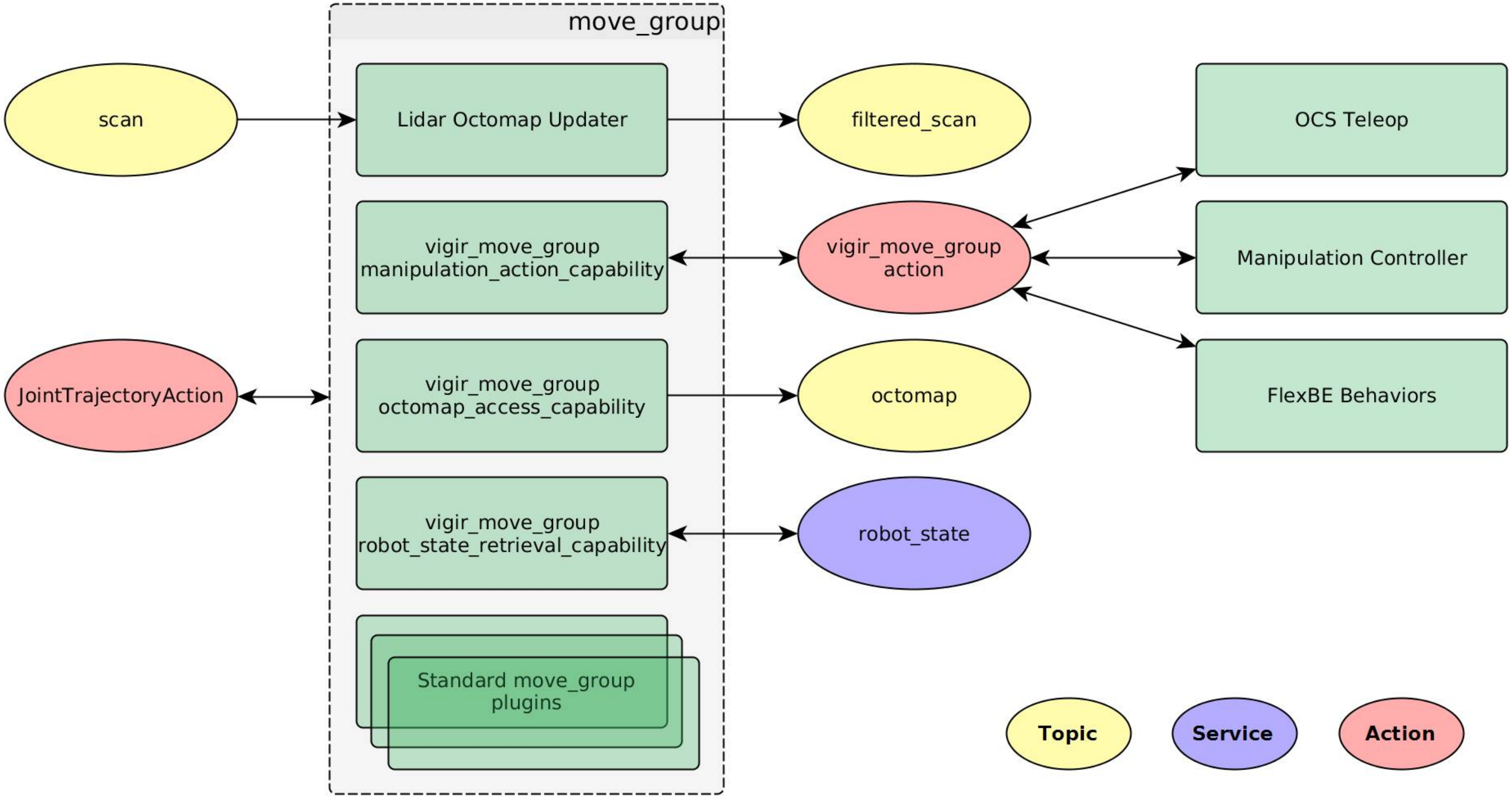
TECHNISCHE UNIVERSITÄT DARMSTADT

- Different robot variants
- Different hand variants
- Combinatory explosion of configs
  - Do not want to run setup assistant for every (possible) combination
- Solution:
  - Use of xacro macros to change configs



github.com/team-vigir/vigir_atlas_planning/tree/master/vigir_atlas_moveit_config

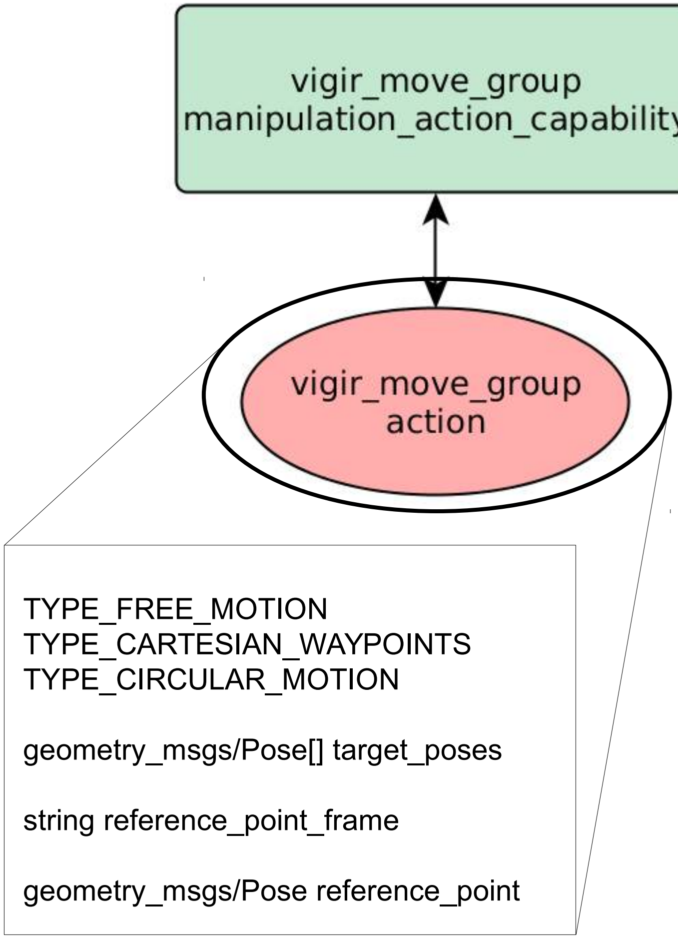
Motion Planning - Overview



Planning - Capabilities

TECHNISCHE UNIVERSITÄT DARMSTADT

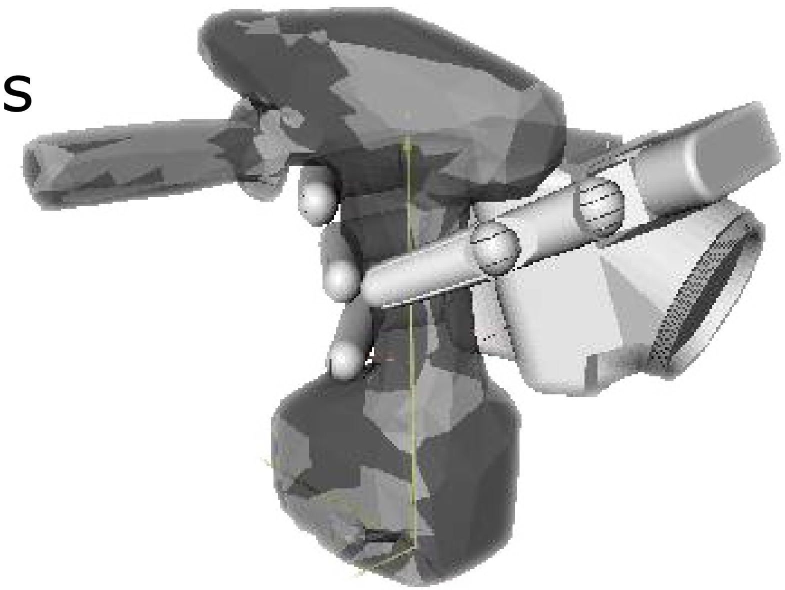
- Additional move_group capability
  - Different types of motion requests
    - Joint goal
    - Cartesian goal
    - Cartesian Path (waypoints)
    - Circular motion
  - Specify planning reference pose relative to endeffector
  - Constrain joint limits selectively at run-time



github.com/team-vigir/vigir_manipulation_planning/tree/master/vigir_move_group

Planning – Object Templates

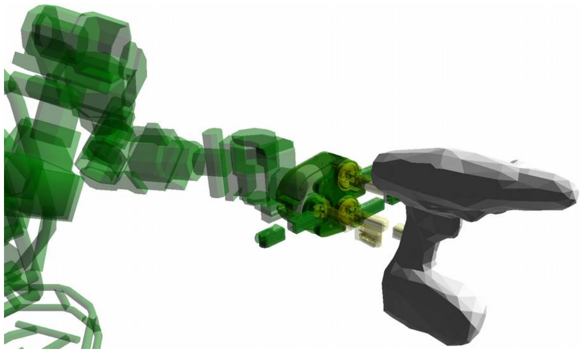
- On top of vigir_move_group
- Operator places objects
- Planning relative to instantiated objects templates
- Object template library
  - Geometry
  - Mass/Inertia
  - Grasps
  - Stand poses

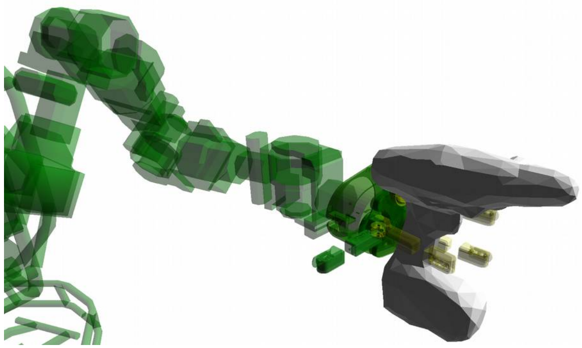


github.com/team-vigir/vigir_object_template_manager

Planning - "Ghost" robot

- Pre-plan motions with virtual "Ghost Robot"
- Additional capabilities compared to start/goal state visualization in MoveIt! Rviz plugin
  - Snap endeffectors to objects
  - Move to stand poses relative to object templates
  - Constrain IK joint limits
  - Send low-bandwidth planning request directly from OCS





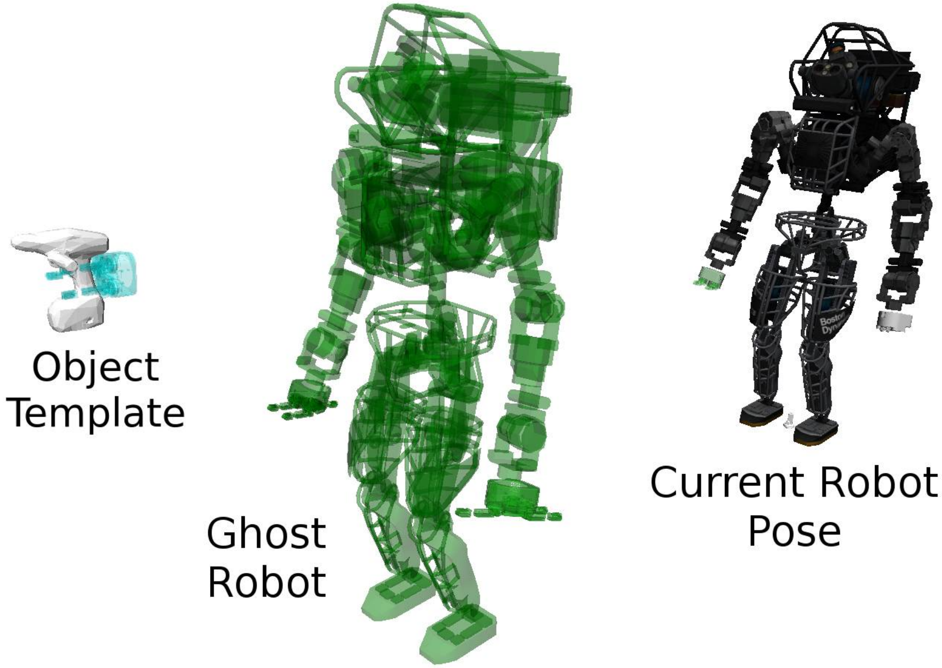
github.com/team-vigir/vigir_manipulation_planning/tree/master/vigir_ocs_robot_model

Manipulation Pipeline Example

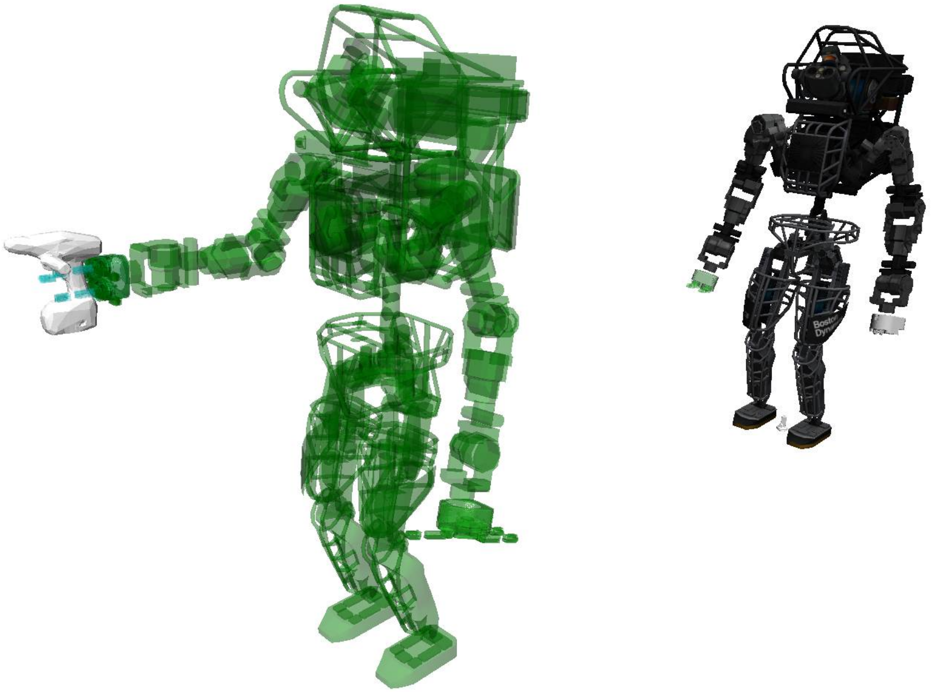
Object
Pointcloud

Current Robot
Pose

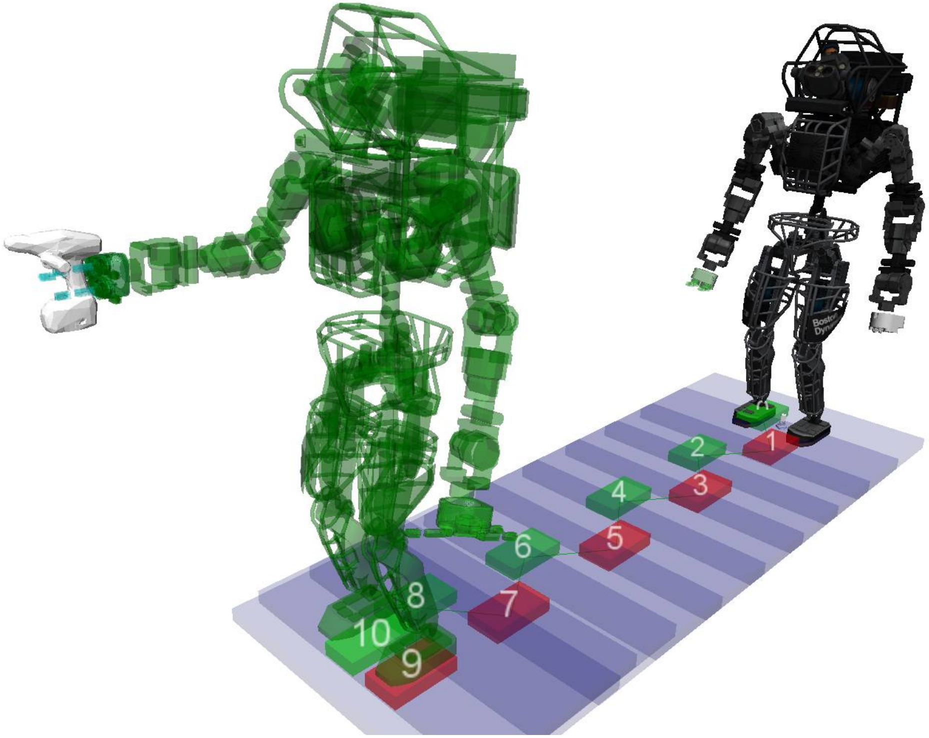
Manipulation Pipeline Example



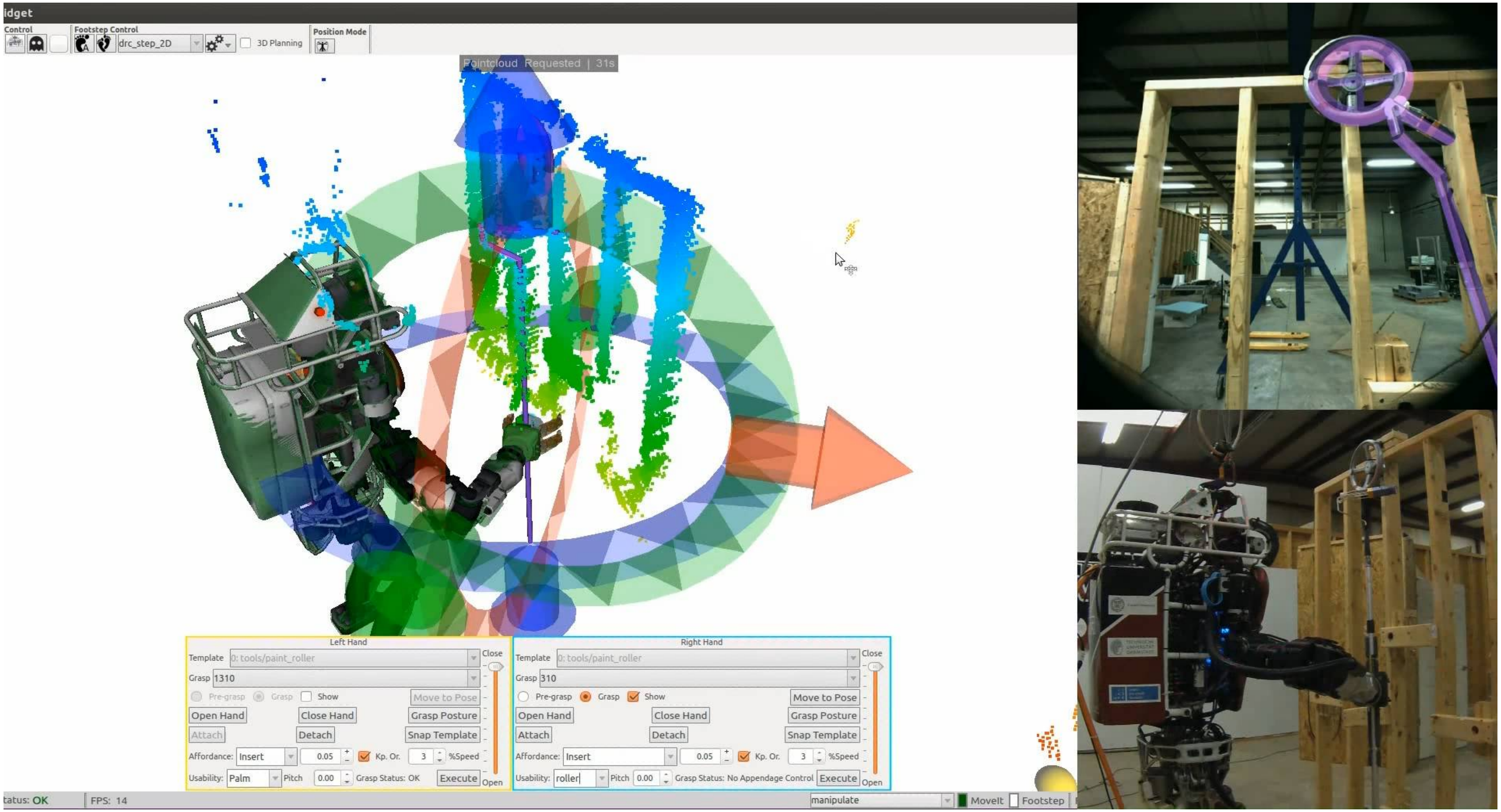
Manipulation Pipeline Example



Manipulation Pipeline Example

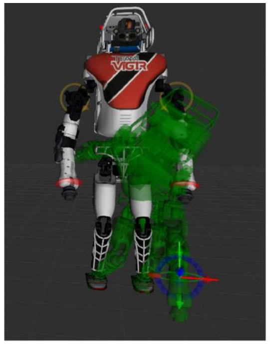


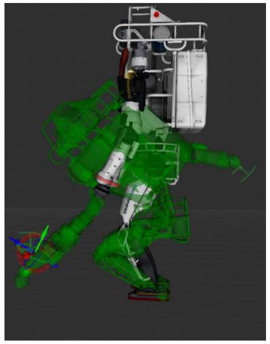
Manipulation example



Manipulation – Drake Integration

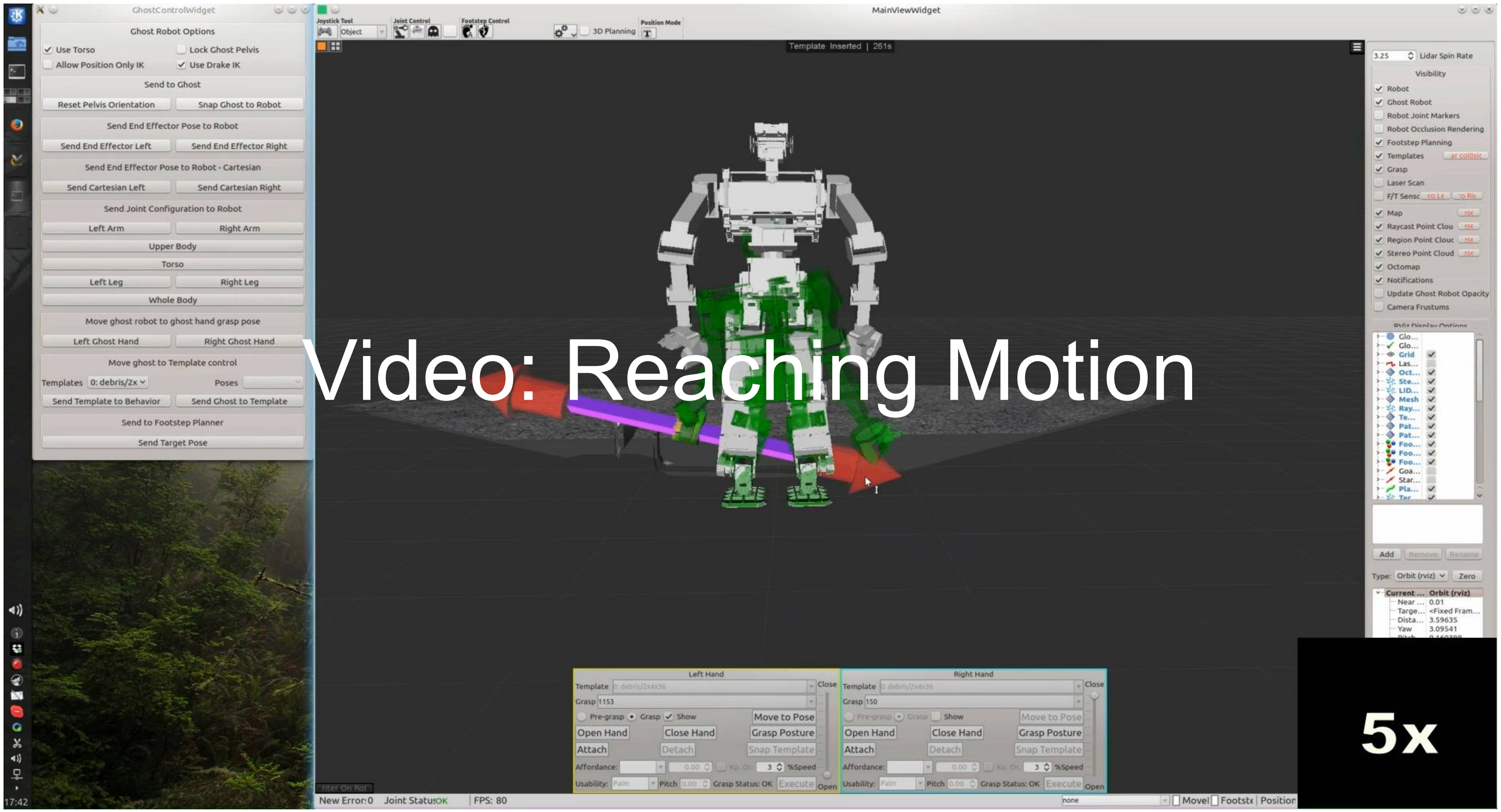
- Switch between MoveIt! and MIT's Drake planning framework on a per plan request basis
  - Whole Body Motions
  - Using github.com/tu-darmstadt-ros-pkg/rosmatlab





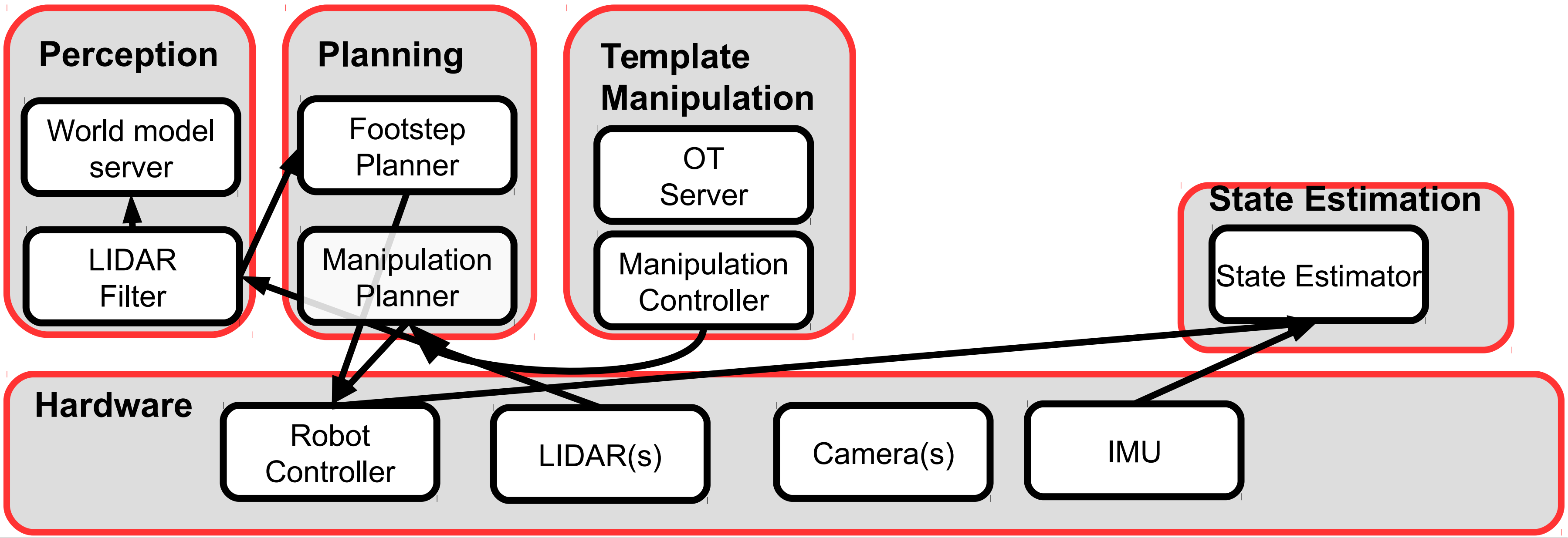
Manipulation – Reaching motion using Drake Integration

TECHNISCHE UNIVERSITÄT DARMSTADT

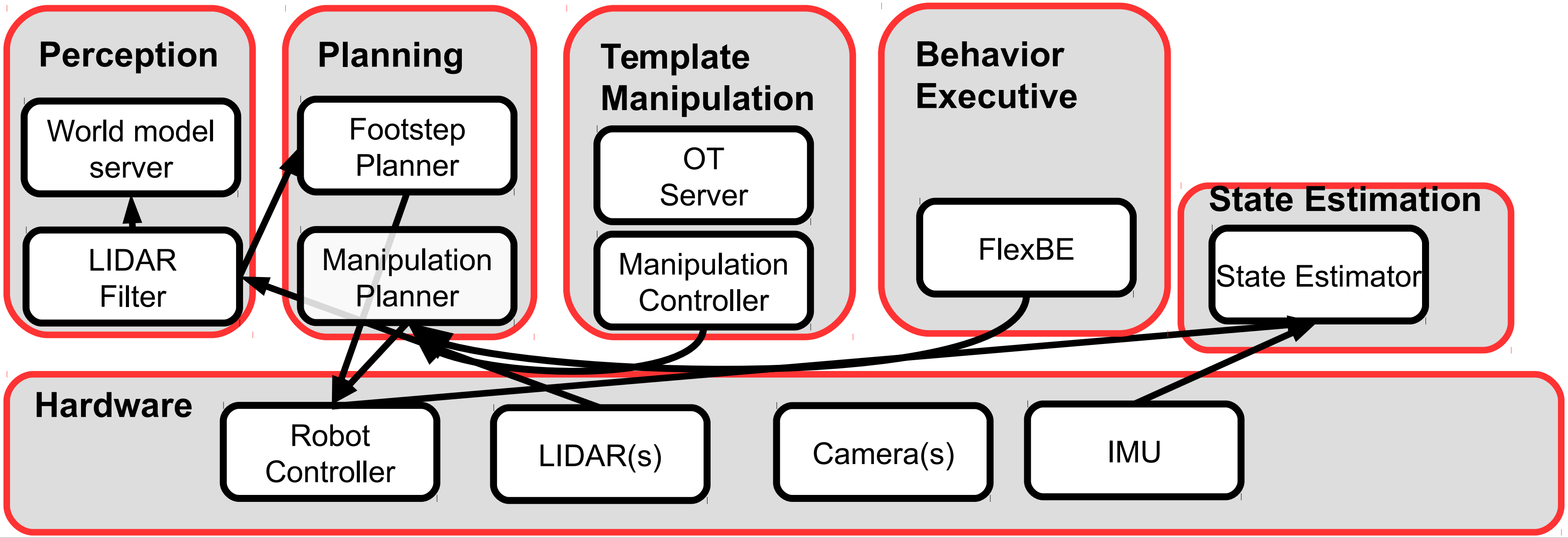


VIGIR SIM

Components – FlexBE Behavior Executive



Components – FlexBE Behavior Executive



Behavior Executive - High-Level Approach

TECHNISCHE UNIVERSITÄT DARMSTADT

- Communication constraints
- Limited time
- Complex robot system

Motivates high degree of **robot autonomy**

Flexible Robot-Operator Collaboration

Motivates high degree of **operator support**

- Unstructured environment
- Complex tasks
- Robustness important

VIGIR SIM

Behavior Executive - High-Level Approach

TECHNISCHE UNIVERSITÄT DARMSTADT

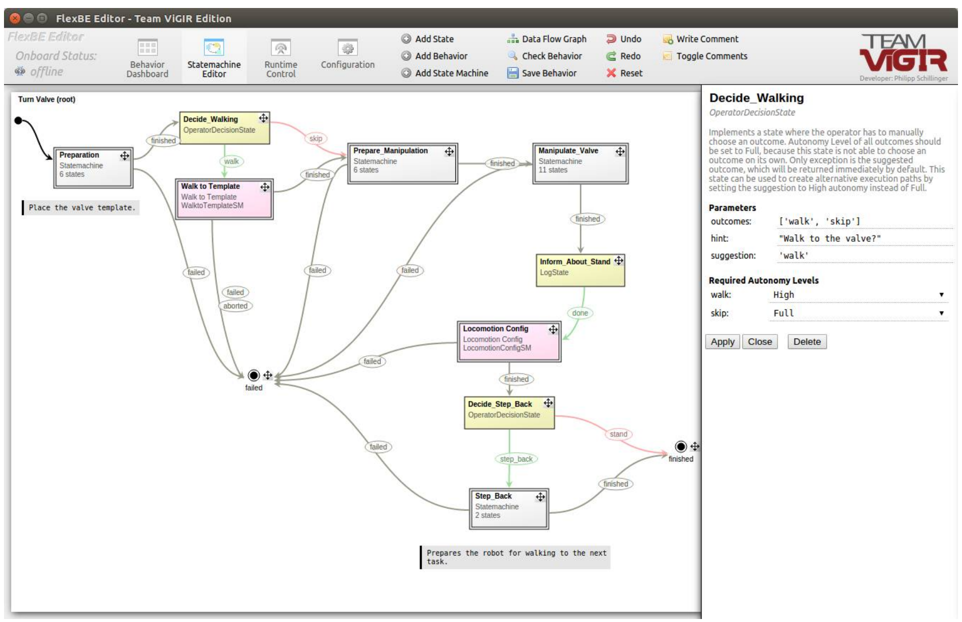
- SMACH, XABSL, etc.
  - Focused on pure autonomy
  - Pre-defined robot behavior

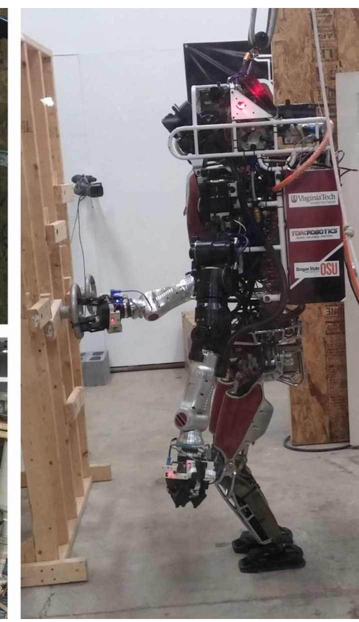
- Required features:
  - Allow multiple degrees of autonomy
  - Support and restrict robot when in low autonomy
  - Adapt behavior to unforeseen situations
  - Abstraction of complex behavior design
  - Robust against runtime failure

Behavior Executive - FlexBE

TECHNISCHE UNIVERSITÄT DARMSTADT

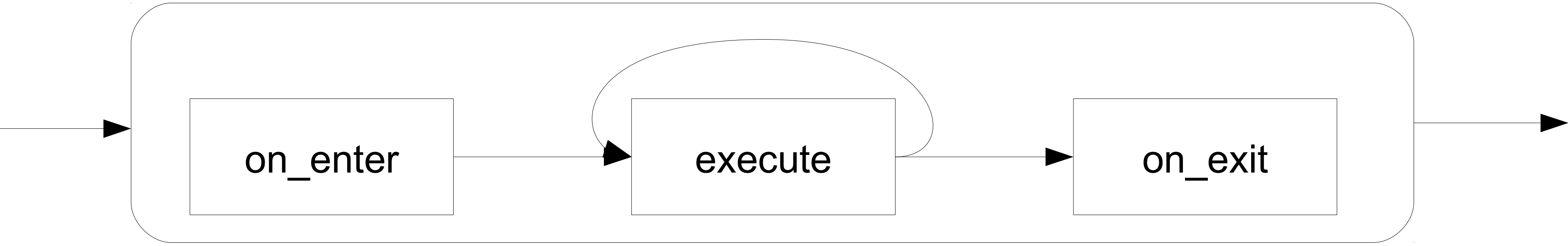
- “**Flex**ible **B**ehavior **E**ngine”
  - Based on SMACH → Hierarchical state machines
  - Adds robot-operator collaboration
  - Available on GitHub: *github.com/team-vigir/flexbe_behavior_engine*





FlexBE – States

- Interface basic robot capabilities / actions
- Executed periodically
- Event-based lifecycle (simplified):



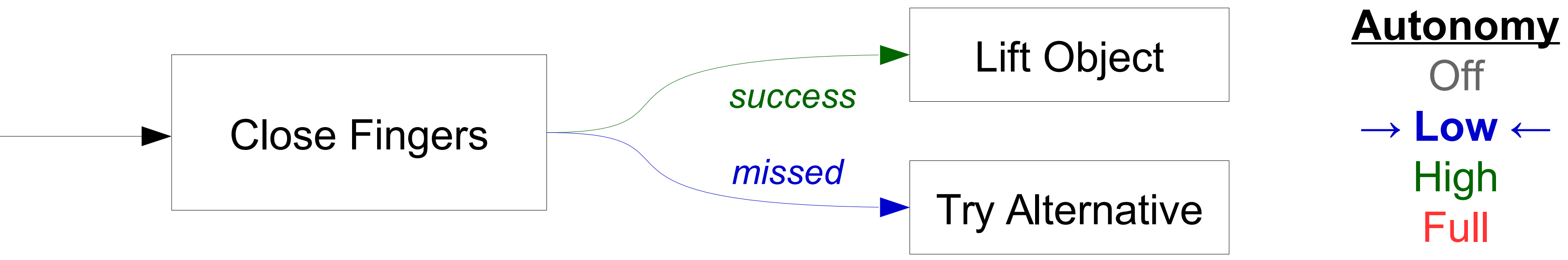
Send command(s), eg.
- publish message
- actionlib call

Check conditions and evaluate results
→ Determine outcome

Clean up

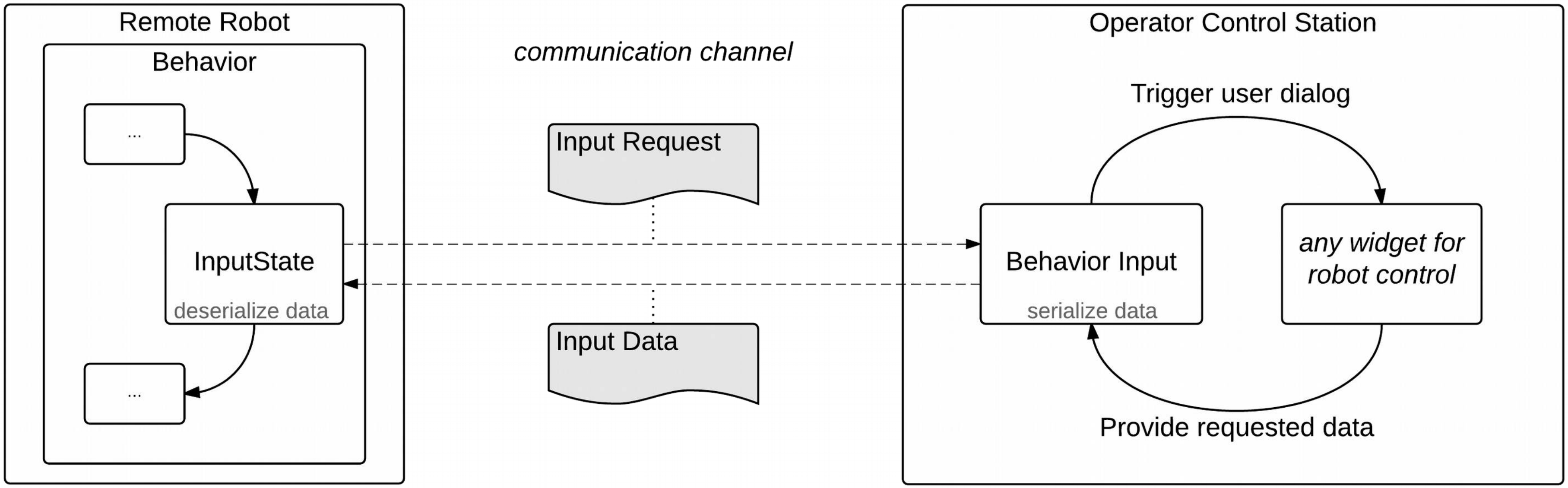
FlexBE – Autonomy Level

- Behavior runs with explicit *Autonomy Level*
  - Can be changed any time during execution
- State outcomes define required autonomy
  - High enough → Autonomous execution
  - Too low → Operator confirms or rejects
- Operator can force outcomes any time



FlexBE – Data Input

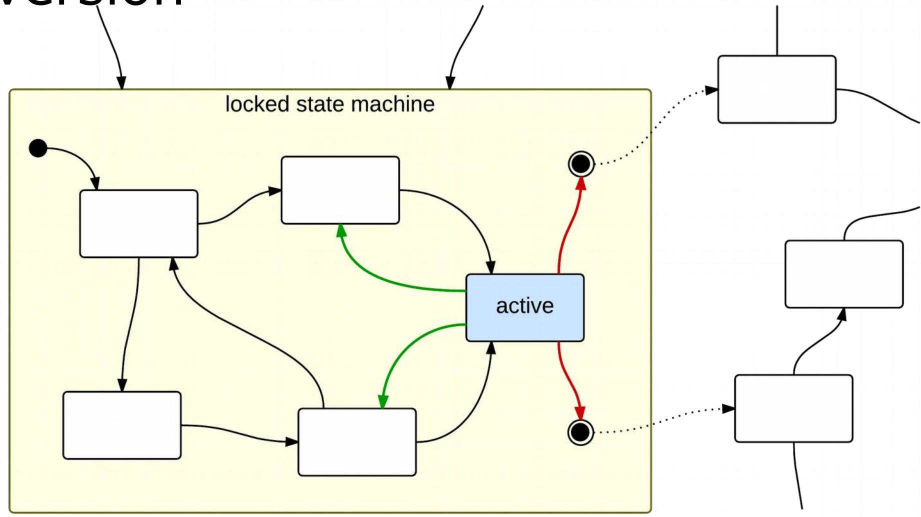
- Behavior can request required data from operator
- Integrated into operator control station



FlexBE – Runtime Changes

- Behavior is locked in a specific state
- Modifications are sent to the onboard executive
- New version is generated and imported
- Active state is transferred
  - Extracted from old, running version
  - Integrated into new version
- Old version is stopped
- New version is executed

→ Arbitrary adaptation



FlexBE – User Interface

TECHNISCHE UNIVERSITÄT DARMSTADT

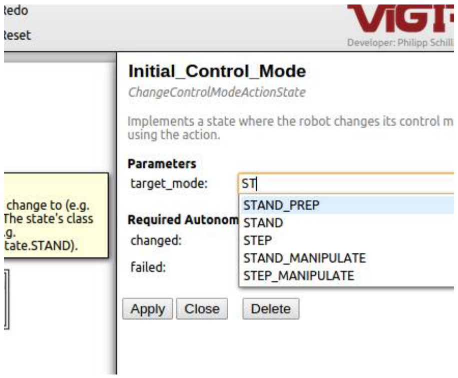
- Facilitates behavior development
- Automated code generation
- Integrated operator interaction

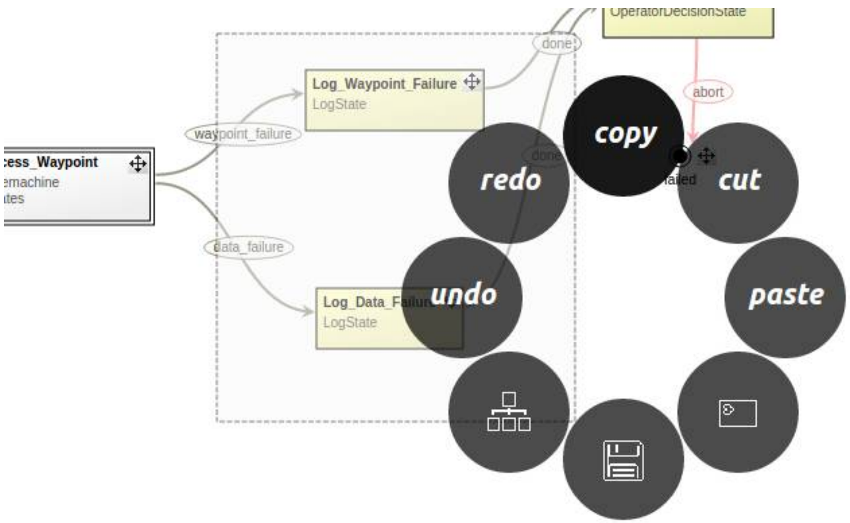
→ Is prerequisite for operator-robot collaboration

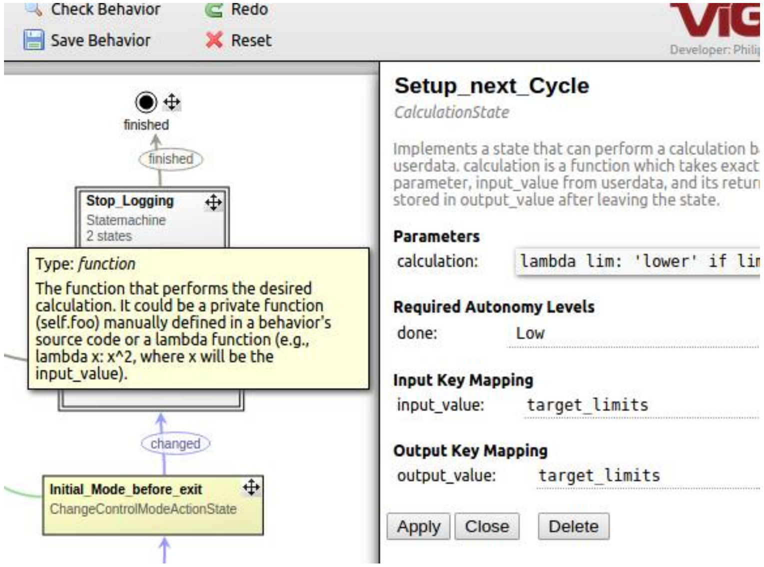
- Behavior re-definition during runtime feasible
- Transparent robot decision-making
- Send context-dependent high-level commands

FlexBE – Editor

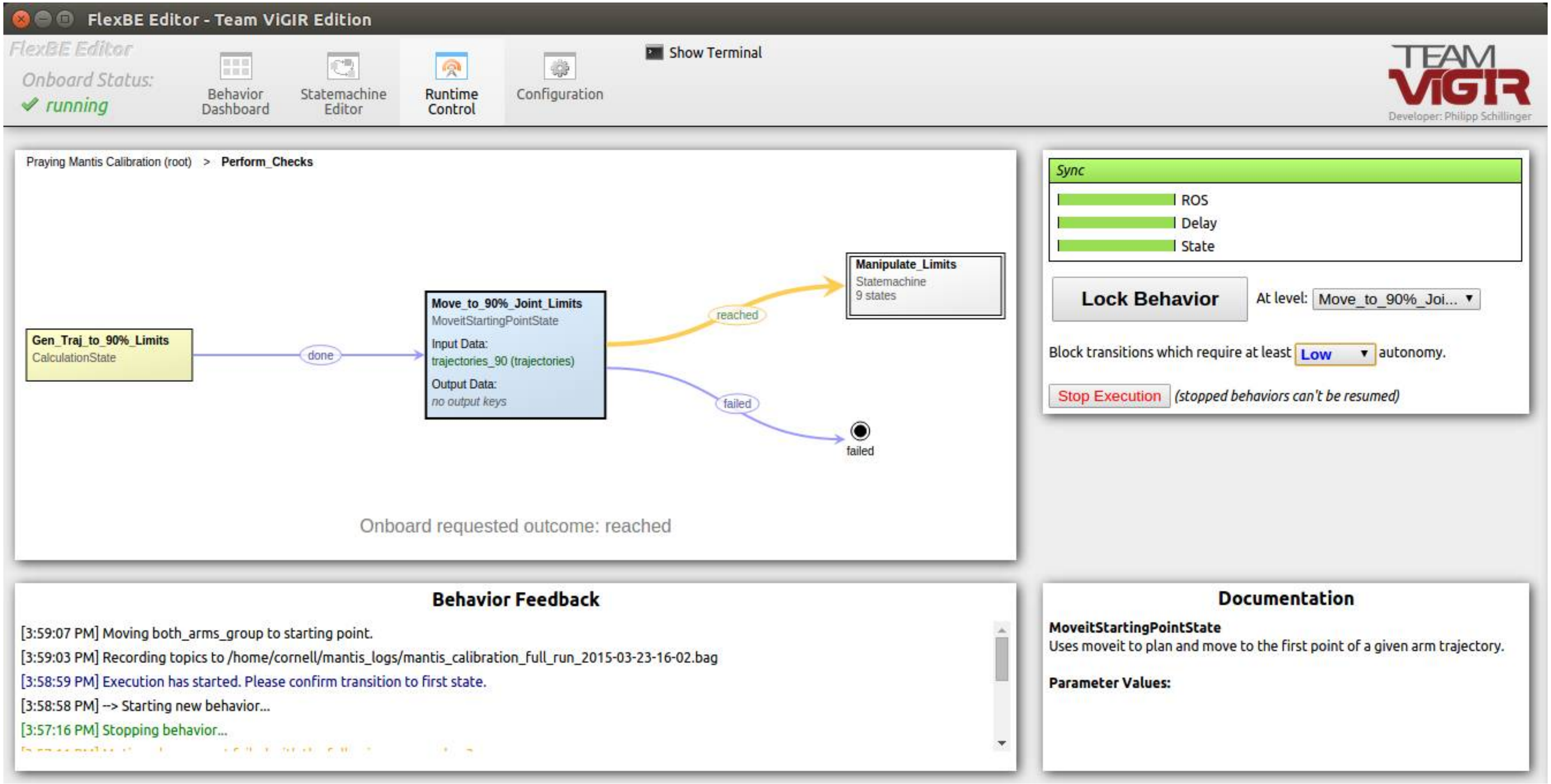
- Drag&Drop state composition
- Configuration of state properties
- Detailed documentation of states
- Dataflow graph and verification



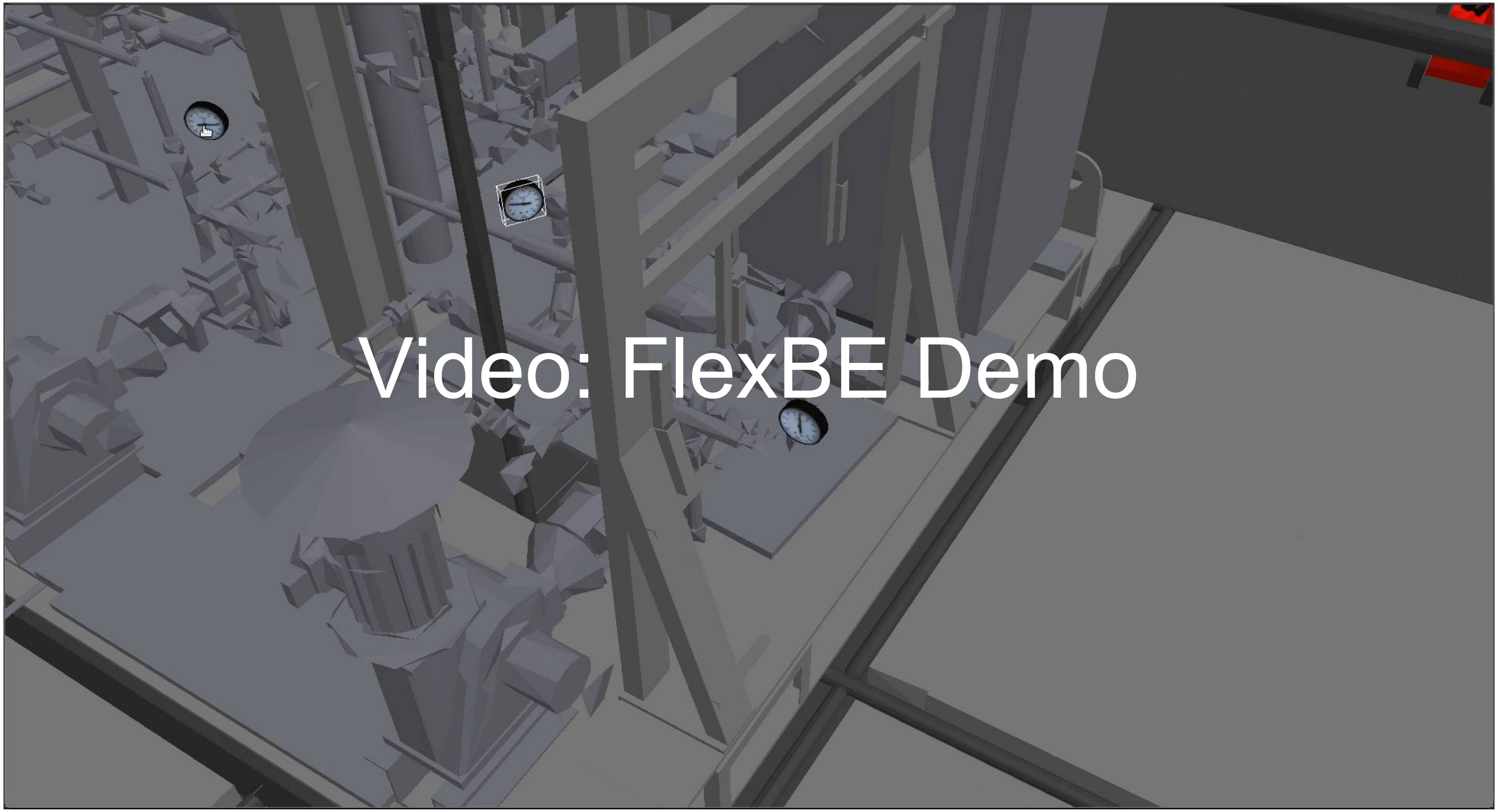




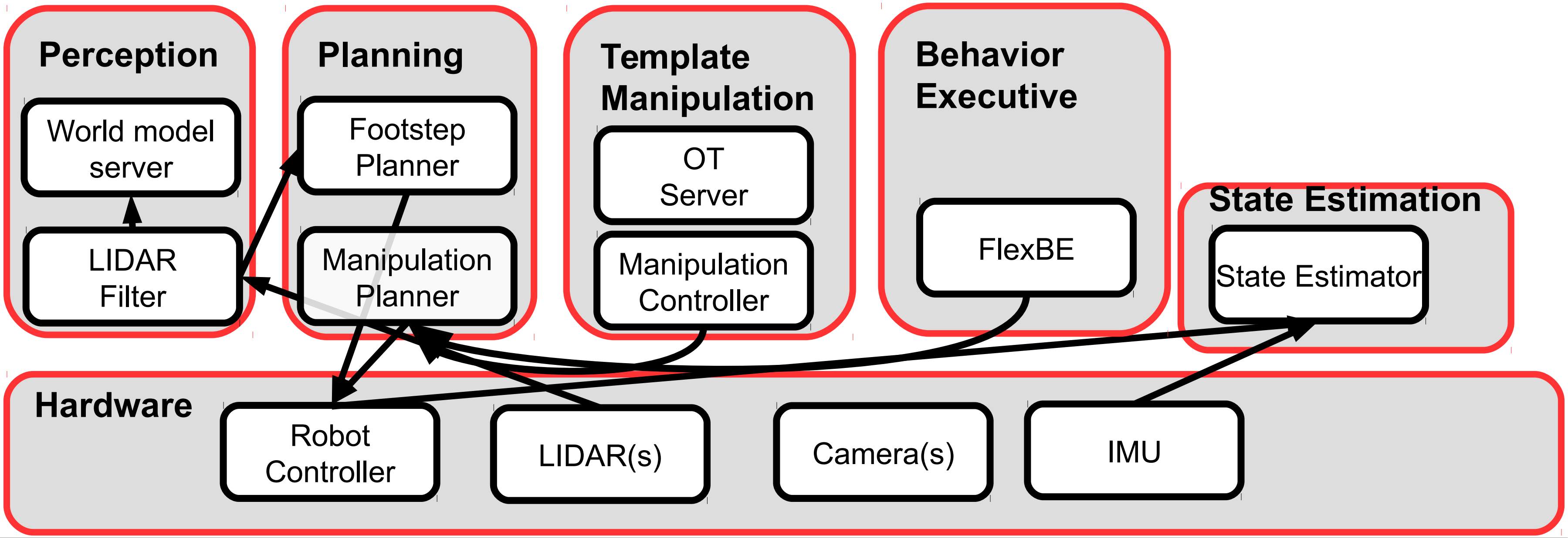
FlexBE – Runtime Control



FlexBE – Beyond DRC application

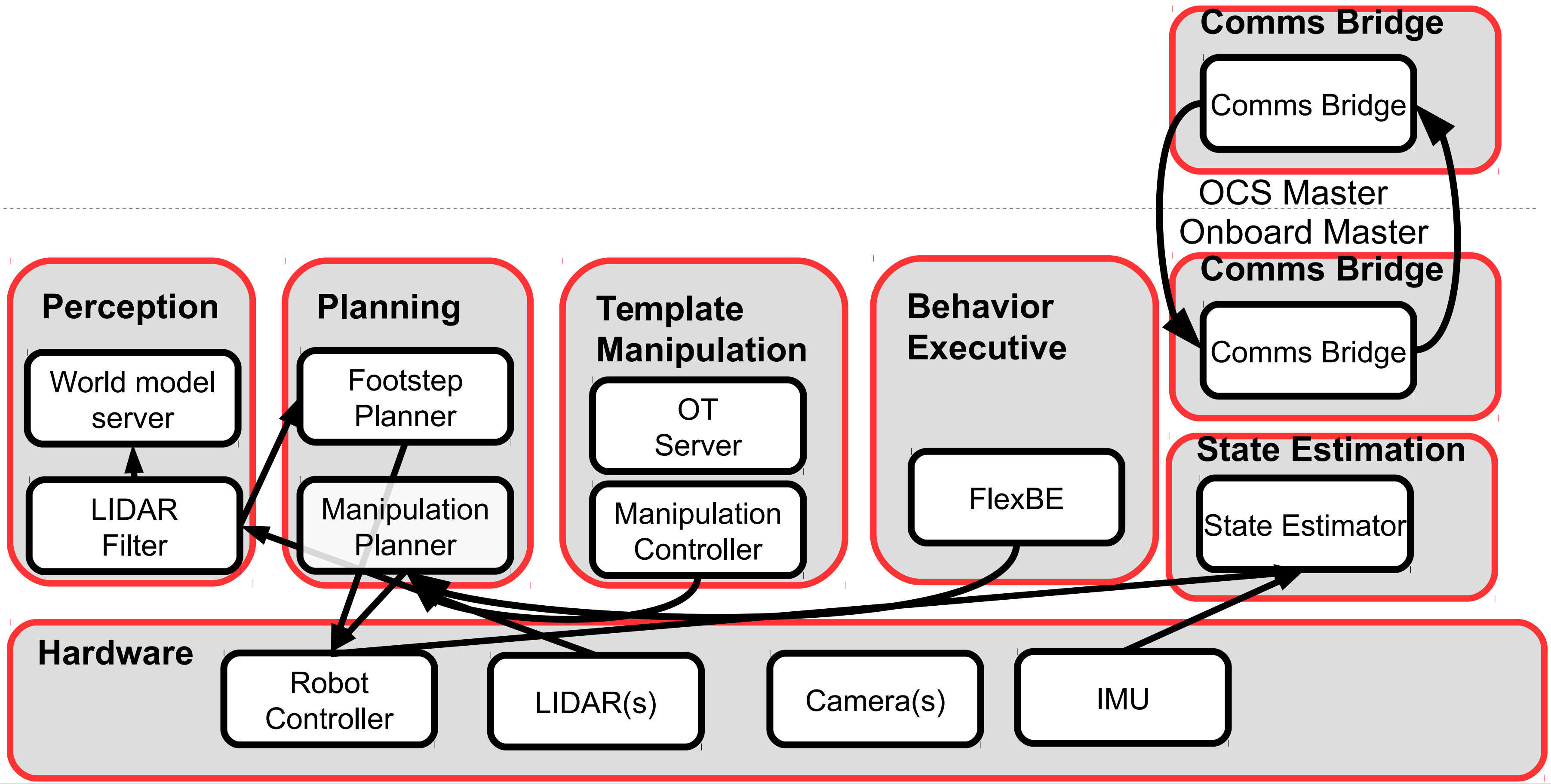


Components – Comms Bridge



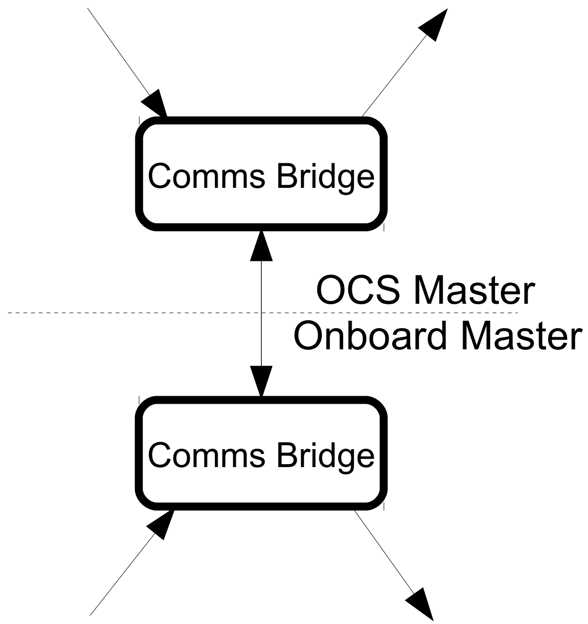
Components – Comms Bridge

TECHNISCHE UNIVERSITÄT DARMSTADT

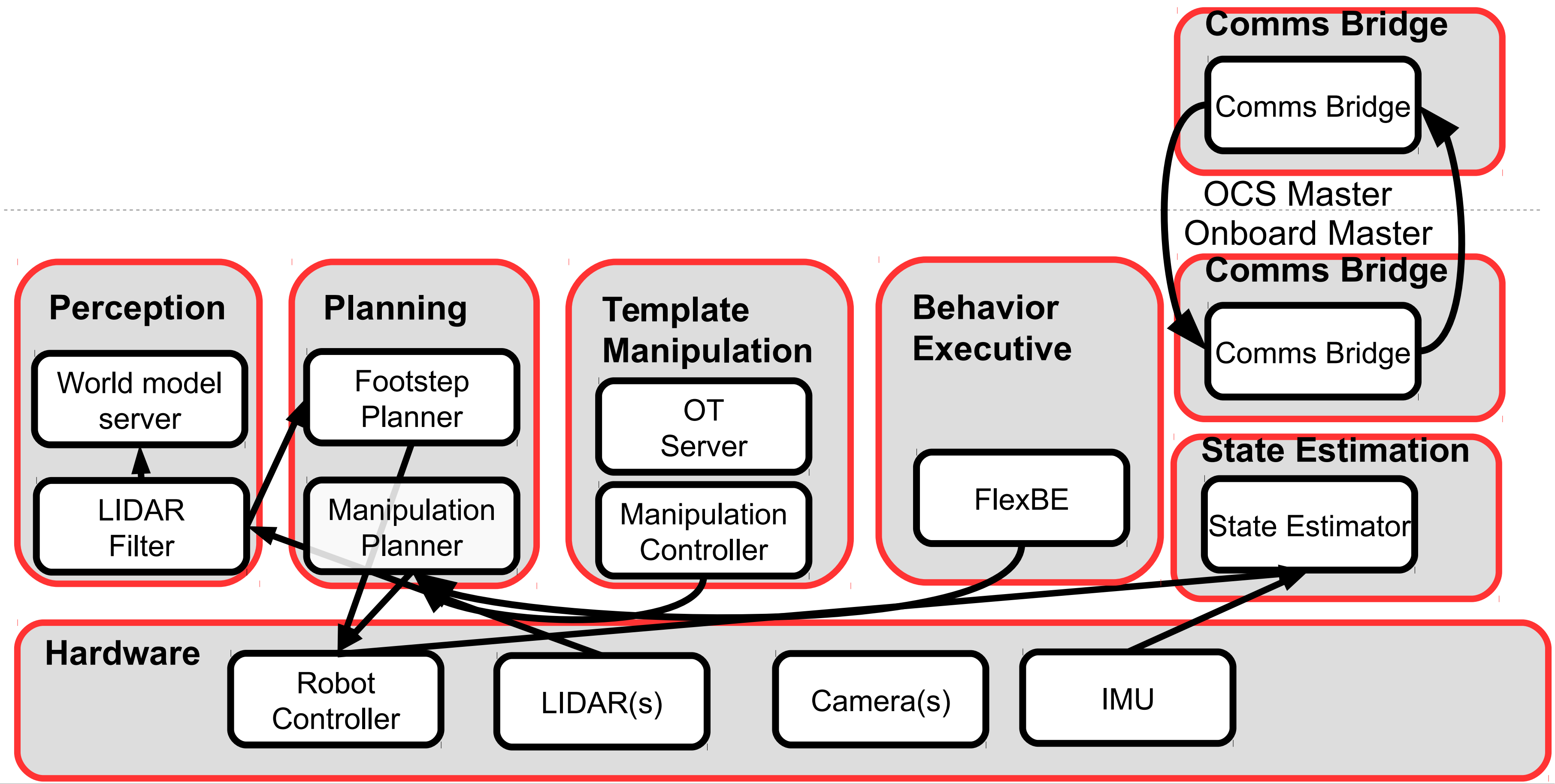


Comms Bridge

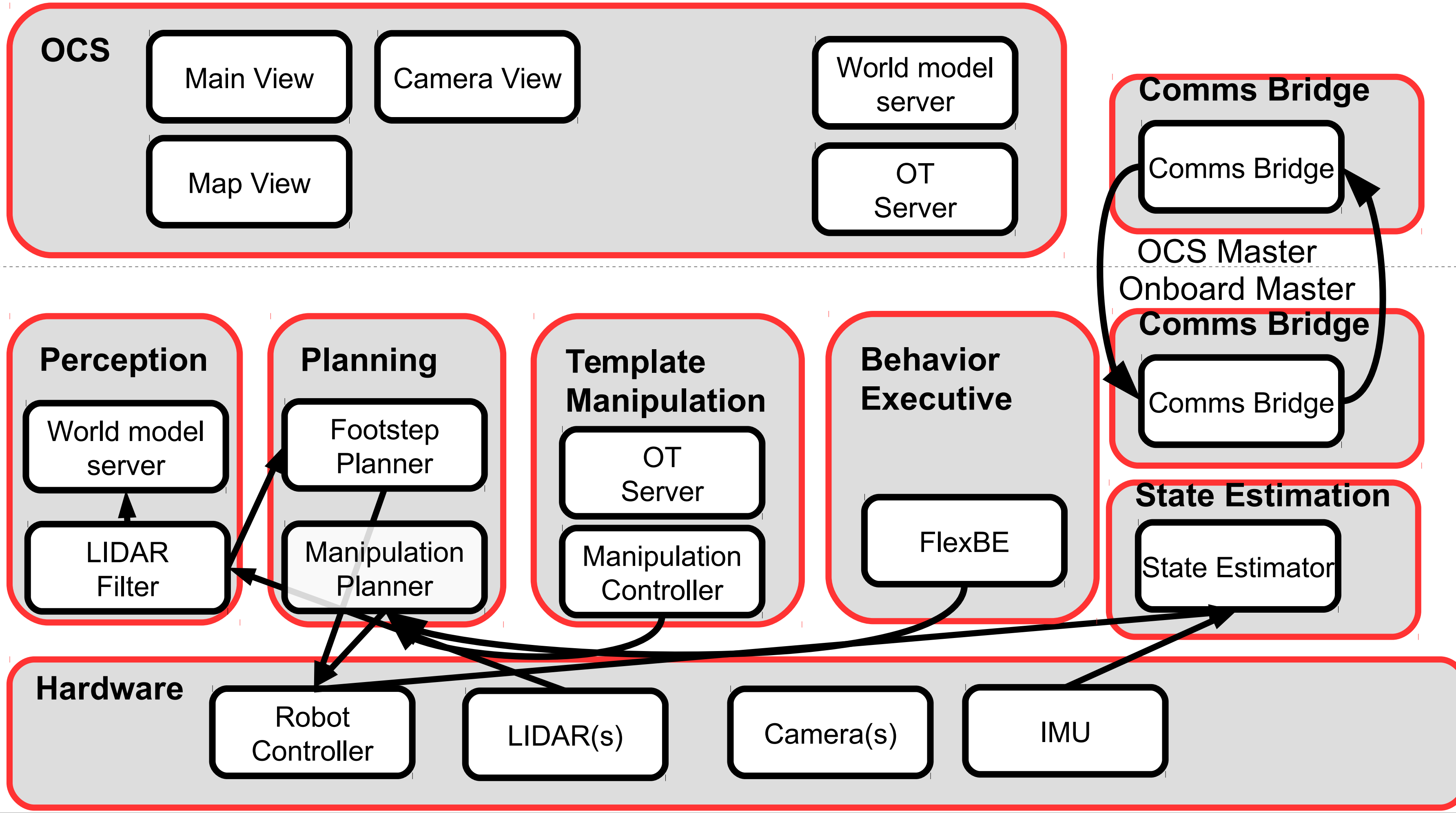
- Single ROS Master infeasible
  - Unreliable connection between operator and robot
- Dual Master approach
  - OCS
  - Onboard
- Prioritization
- Special treatment of high rate state data
  - Compress using domain knowledge
- Other data compressed using blob_tools
  - Bz2 compression per default



Components - OCS



Components - OCS



OCS

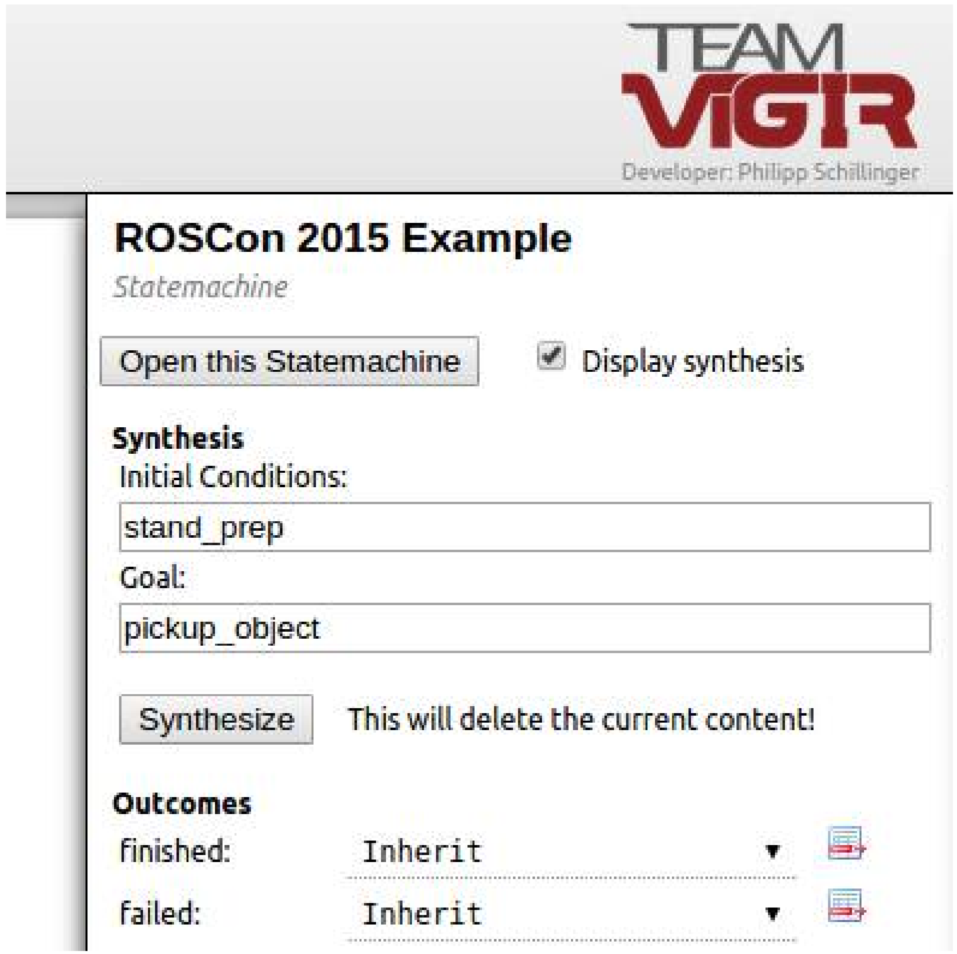
- 3D visualization based on librviz
  - Map View (Top Down)
    - Rectangle selection (query sensor data ROI)
  - Main View
  - CameraView
    - Camera data visualization
- Multiple Qt widgets for general controls
  - "Ghost Control"
  - Pre-canned joint configurations

Components – Install

- Install instructions for complete setups: github.com/team-vigir/vigir_install/wiki
  - Waiting for Atlas IHMC/Gazebo integration for full capability (walk/manipulate) Atlas example
  - Thor-Mang example available for manipulation demo: github.com/team-vigir/vigir_install/wiki/Install-thor-mang-vigir-gazebo

Components – Tutorial video

Work in Progress - Behavior Synthesis



Work in Progress - Behavior Synthesis

TECHNISCHE UNIVERSITÄT DARMSTADT

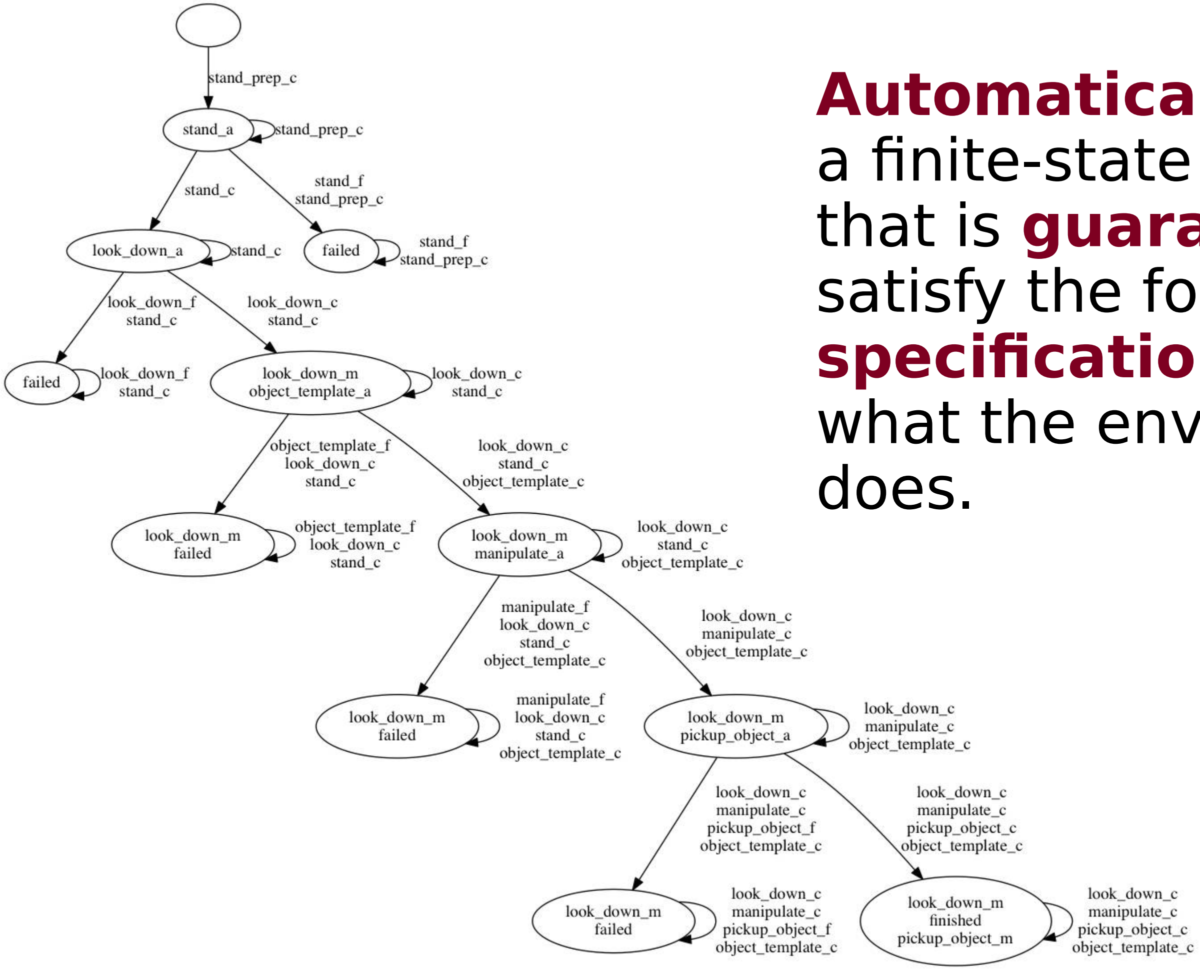
- Compile formal Linear Temporal Logic (LTL) specification from:
  - High-level task (goals and initial conditions)
  - Abstract description of the robot-plus-software system, defined a priori (think config files)

- The formalism treats the outcomes of actions as an adversarial environment

github.com/team-vigir/vigir_behavior_synthesis

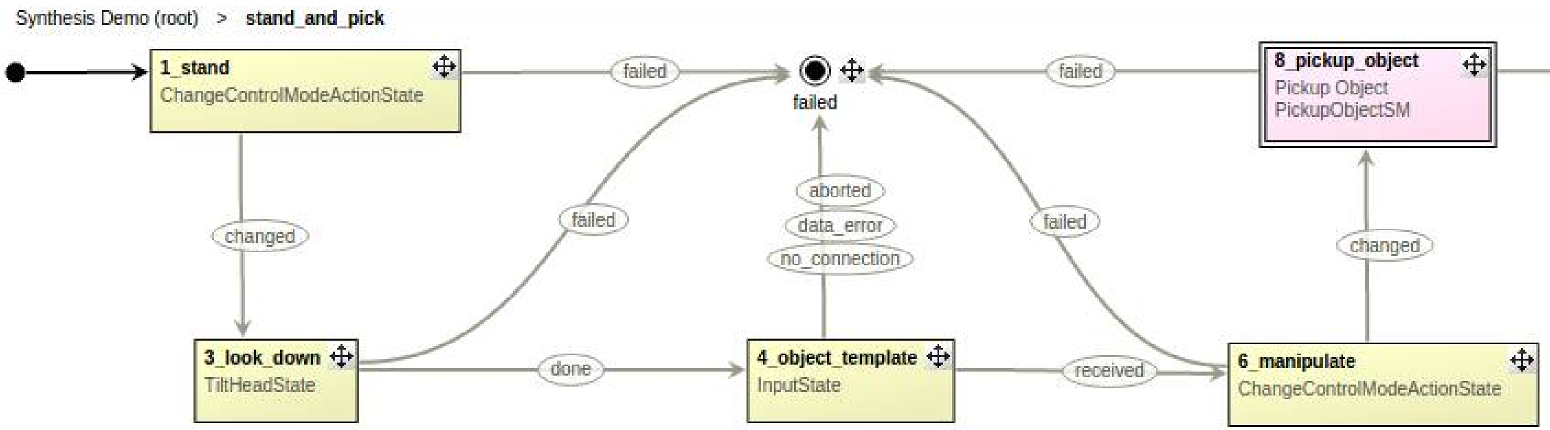
Work in Progress

TECHNISCHE UNIVERSITÄT DARMSTADT



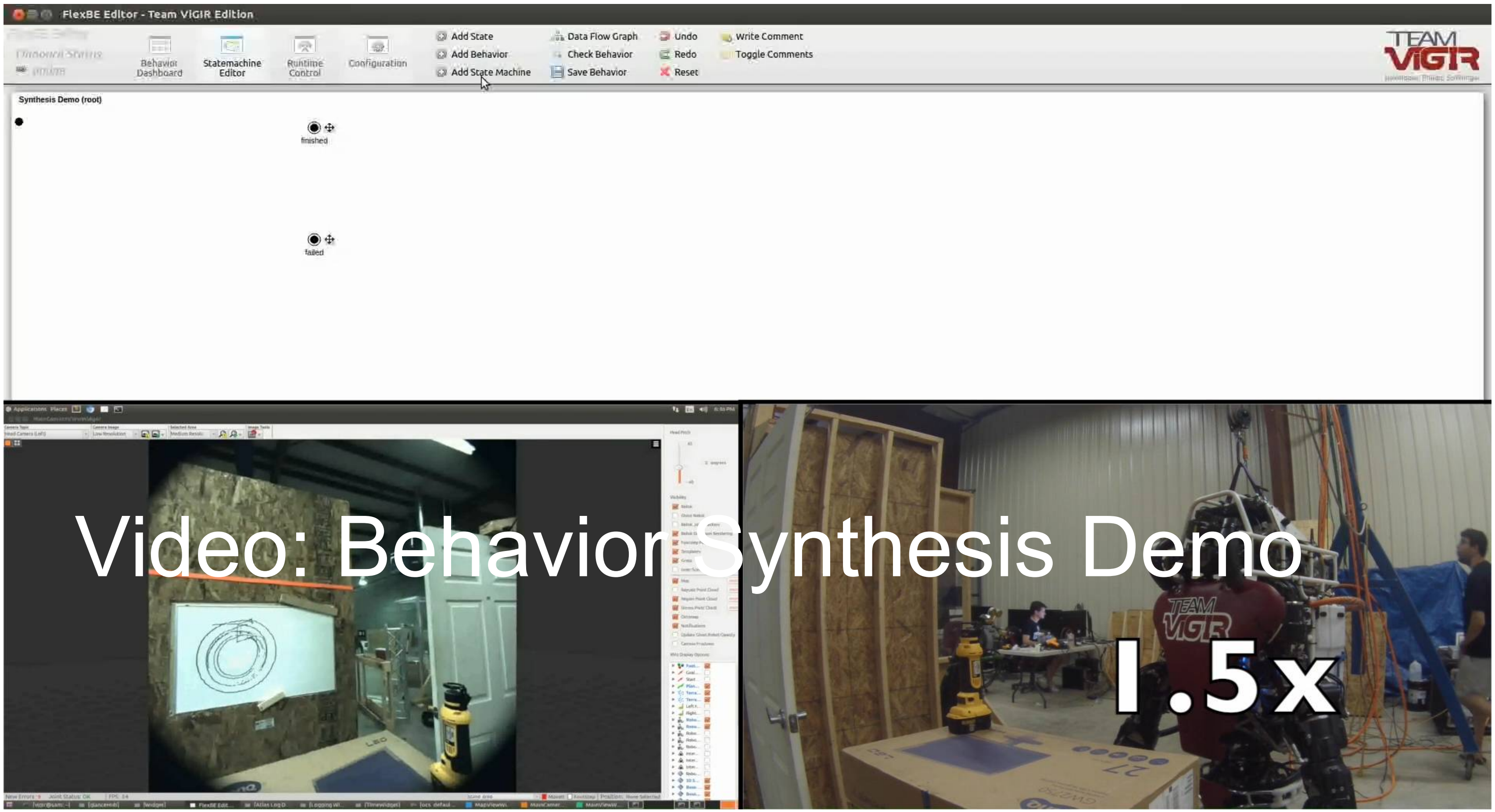
Automatically synthesize a finite-state automaton that is **guaranteed** to satisfy the formal LTL **specification** no matter what the environment does.

Work in Progress



- Mapping from abstract symbols to low-level system components (here, FlexBE states)
- Instantiation of symbolic automaton as an executable state machine in FlexBE

Behavior Synthesis - Example



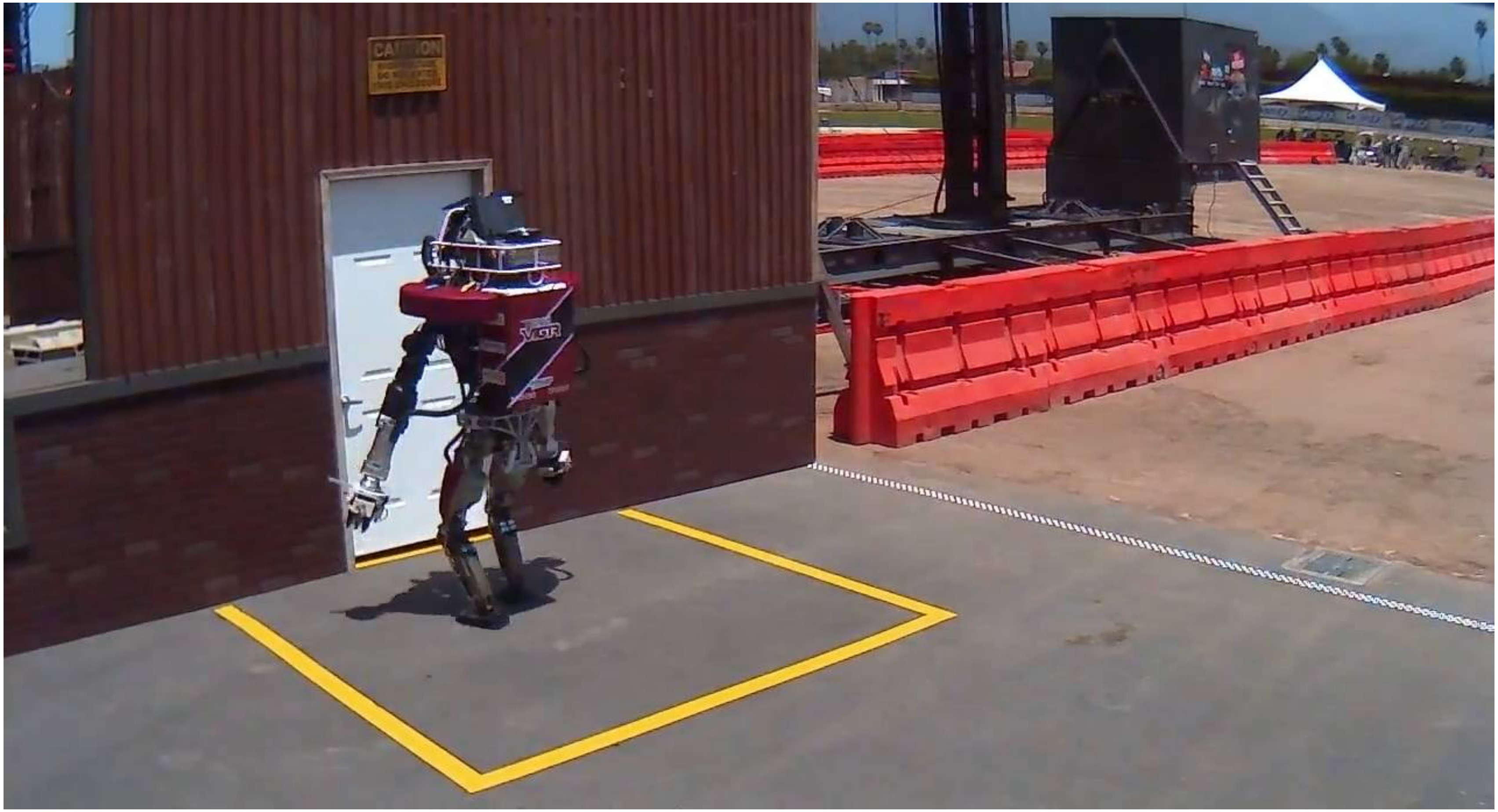
DRC Finals

- Decision not to do egress
  - Significant development effort
  - Risk of (catastrophic) damage to robot
- Limited testing under degraded comms conditions

DRC Finals – Day 1

DRC Finals – Day 1

DRC Finals – Day 1



DRC Finals – Day 1

DRC Finals – Day 1

DRC Finals – Day 1

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Flawless Driving
- Comms bridge setup issue
  - Behavior control
  - Footstep planning
- Switch to teleop mode
- Slow but reliable

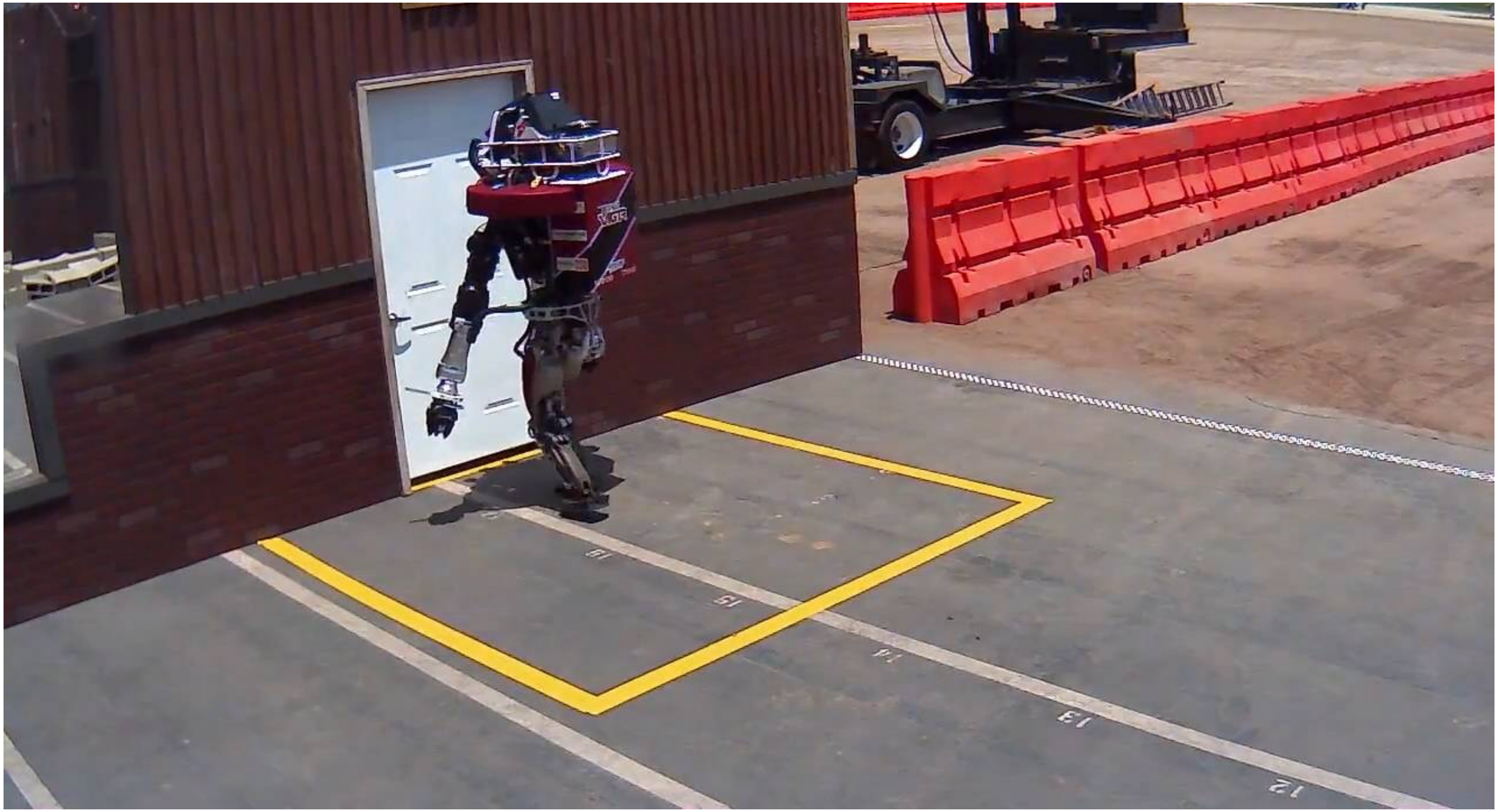
DRC Finals – Day 2

DRC Finals – Day 2

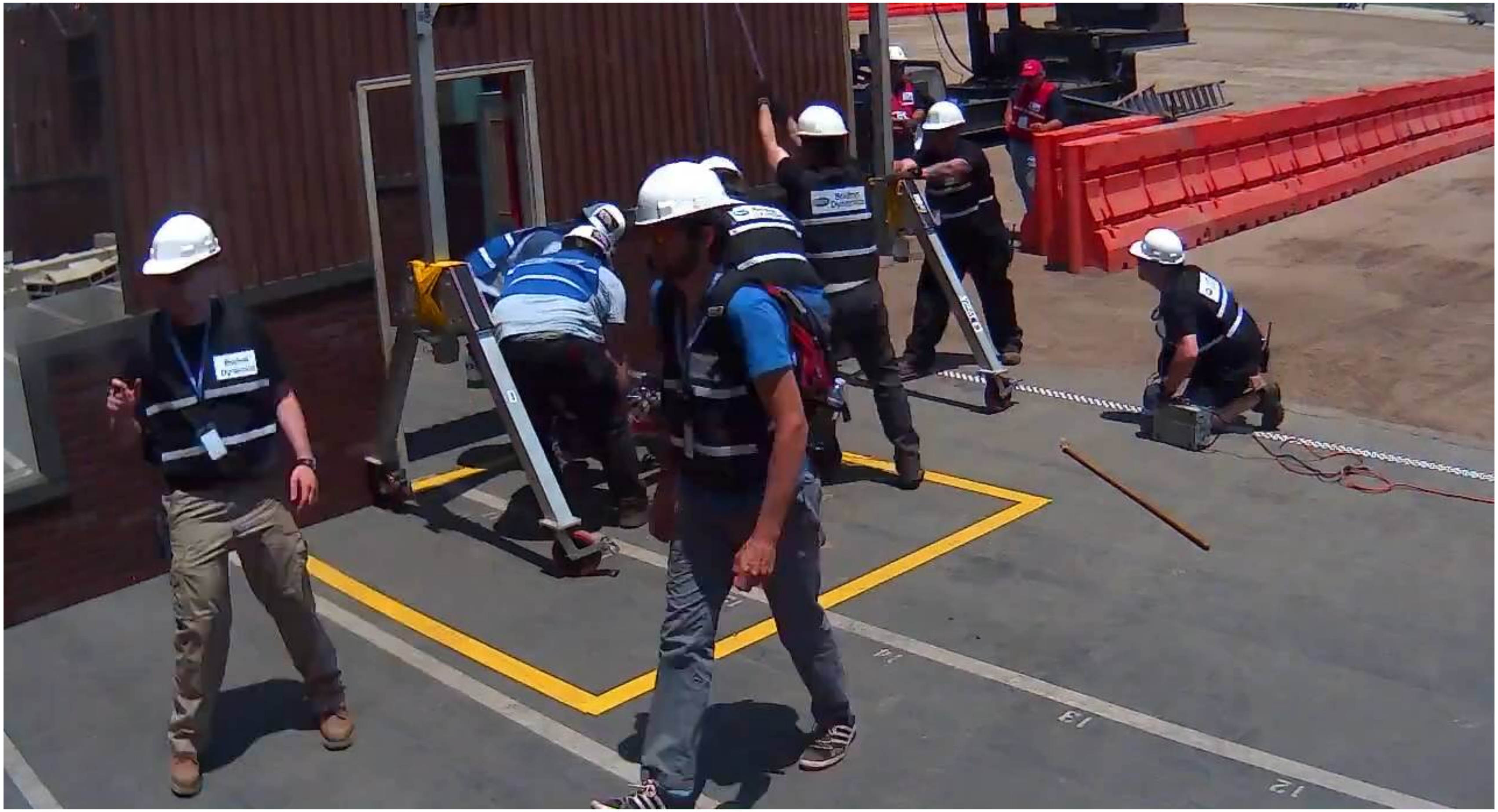
DRC Finals – Day 2

DRC Finals – Day 2

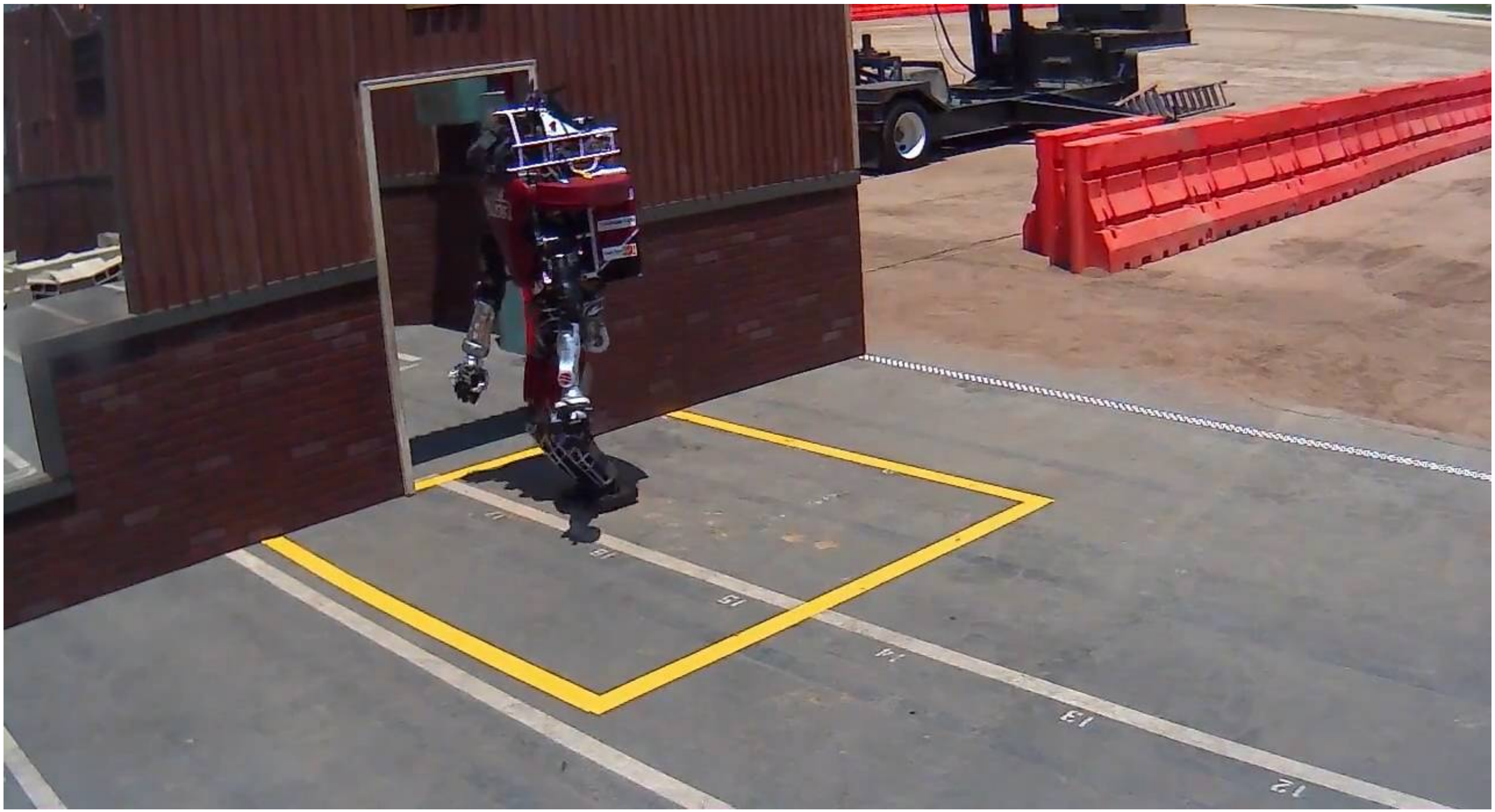
DRC Finals – Day 2



DRC Finals – Day 2



DRC Finals – Day 2



DRC Finals – Day 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Start delay due to arm hardware failure
- (Too) fast driving
- Reset after touching barrier
- Successful driving
- Door opened
- Pump shutdown
  - Possibly due to overheating
- Reset
- Fall while walking through door

DRC Finals Results

- 3 Points (Day 1)
- Scored lower than would have been achievable and expected
  - Achievable: 7 points (No egress)
- Missed chance at Day 1 due to comms issues
- Unknown cause for pump shutdown at Day 2

- Driving approach worked well on both robots that used it
  - ViGIR Florian (Atlas)
  - HECTOR Johnny (Thor-Mang)

Lessons Learned - ROS

TECHNISCHE UNIVERSITÄT DARMSTADT

- Workspace setup using wstool works well
  - Few convenience scripts helpful
- Keeping pace can be painful
  - From rosbuild to catkin
  - From hydro to indigo (switching ROS distro and Ubuntu version simultaneously)
- Using plain “catkin_make” in large projects bad idea
  - Use catkin_tools
- Limited constrained comms capability
- Supporting different configurations feels more involved than it should
  - Environment variables?

Lessons Learned – Big Picture

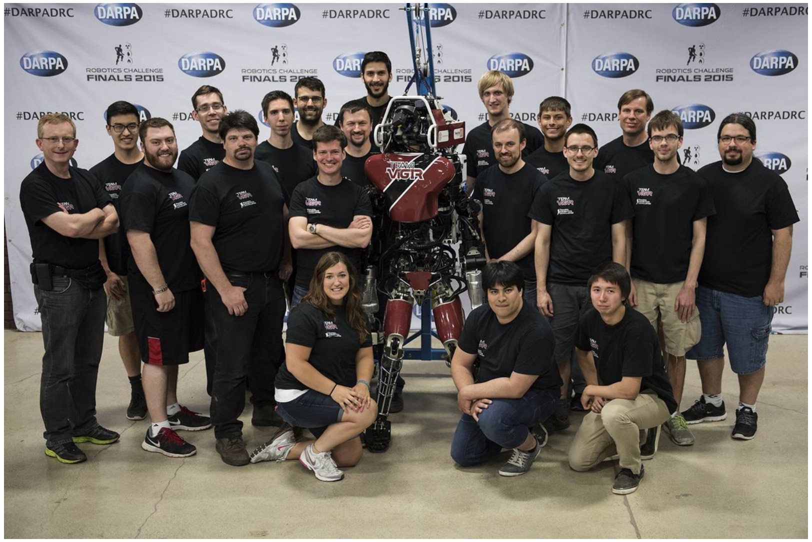
TECHNISCHE UNIVERSITÄT DARMSTADT

- Having a transatlantic, nine time zone team works
  - Right mindset and people
  - Tools
- DRC showed what is possible
  - Brilliant display of state of the art capabilities
  - Still a long way to go till robots can be useful for real DRC-like tasks
- Continuous Integration
  - Simulation-in-the-loop testing desirable
- Everybody wins
  - Leap across wide range of capabilities
  - Open source developments (Gazebo, code releases..)
  - Incredible sportsmanship and cooperation across DRC teams

Conclusions

TECHNISCHE UNIVERSITÄT DARMSTADT

- DRC overview
- ROS infrastructure discussion
  - Useful tools
- Intro to open source components
  - Let us know what you're interested in
- DRC results discussion
- Lessons learned



Questions?

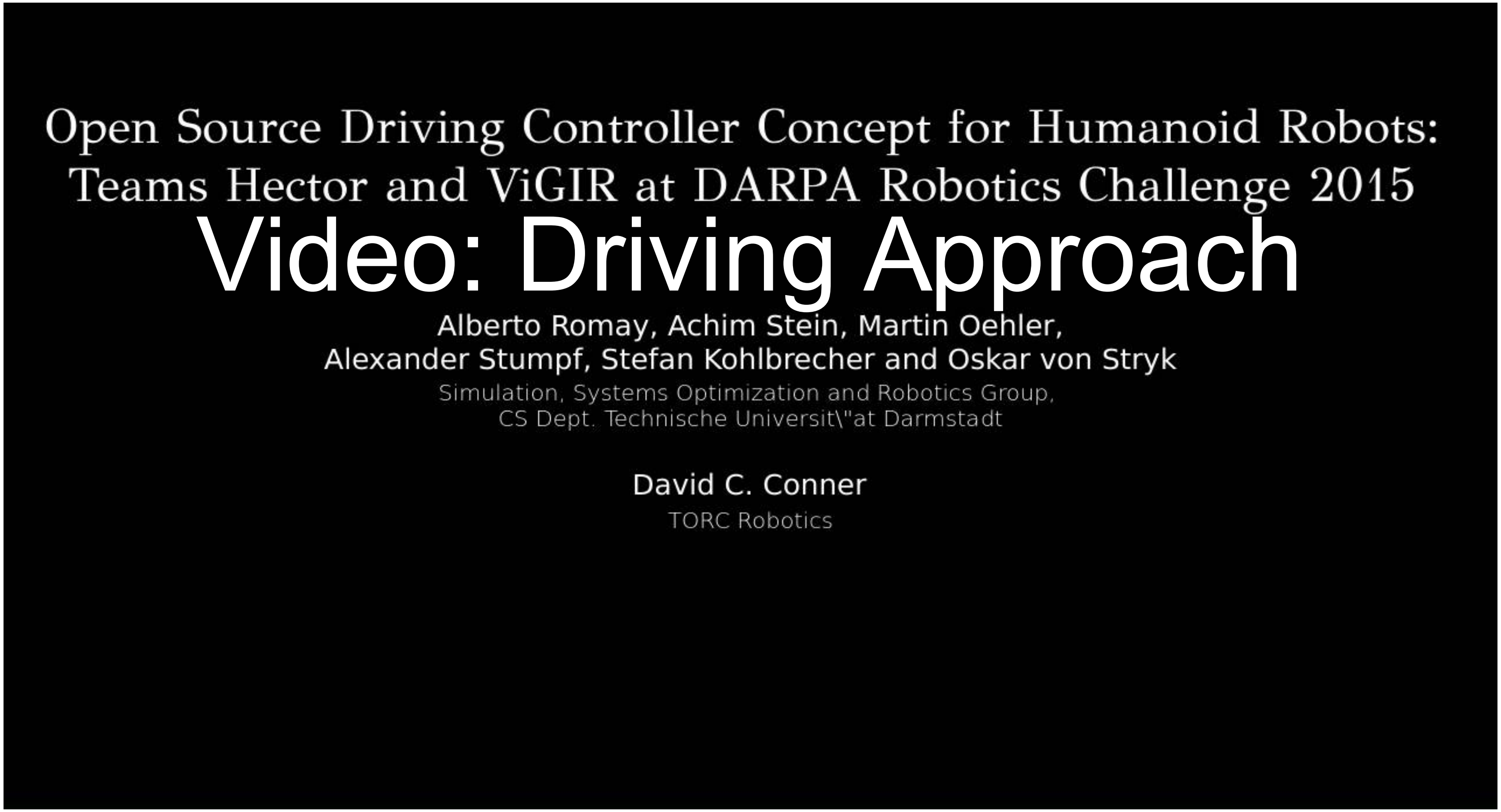
References

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kohlbrecher et al. “Overview of team ViGIR's approach to the Virtual Robotics Challenge”, IEEE SSRR 2013
- Kohlbrecher et al. “Human-Robot Teaming for Rescue Missions: Team ViGIR´s Approach to the 2013 DARPA Robotics Challenge Trials” Journal of Field Robotics, 2014
- Romay et al. “Template-Based Manipulation in Unstructured Environments for Supervised Semi-Autonomous Humanoid Robots”, IEEE Humanoids 2014
- Stumpf et al. “Supervised Footstep Planning for Humanoid Robots in Rough Terrain Tasks using a Black Box Walking Controller”, IEEE Humanoids 2014

- YouTube playlist with manipulation examples

Additional Material - Driving Approach



Additional Material – DRC Communication constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

	Uplink (to OCS)	Downlink (to robot)	Remarks
VRC	Total ~115 kB for 30 minutes. 500 ms latency	Total, ~7 MB for 30 minutes. 500 ms latency	Worst case (20% of scenarios)
Trials	1 MB/s, 50ms latency	1 MB/s, 50 ms latency	Good comms — Switch every minute
	100 kB/s, 500 ms latency	100 kB/s, 500 ms latency	Bad comms
Finals	1.2 kB/s	1.2 kB/s	
	300 Mbit/s		Outages of 1-30 seconds after robot traverses door

VRC Rules (pdf)

Trials Rules (pdf)

Finals Rules (pdf)