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

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
...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

- 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

• 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

Versatile and robust (Loco-)Motion
The “C” in DRC

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 (Loco-)Motion
- Versatile and robust Manipulation
- Efficient interaction with Human Supervisor
- 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 reliability
  - 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)
- Leibniz Universität Hannover (Hannover, Germany)
- Oregon State University (Corvallis, OR)
Team ViGIR

- Track B team, DRC participation from day one
  - Virtual Robotics Challenge (VRC)
  - DRC Trials
  - DRC Finals

- Software available: [github.com/team-vigir](http://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

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

- IHMC:
  - IHMC Controller (ihmc_ros bitbucket)
  - SCS Simulator

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

- 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

  - Tethered
  - 6DOF arms
Hardware – Atlas Versions

  - Tethered
  - 6DOF arms

- **Atlas V4 (Feb 2015–Mar 2015)**
  - Untethered
  - Onboard Computing
  - 6DOF arms
Hardware – Atlas Versions

  - Tethered
  - 6DOF arms
- **Atlas V4 (Feb 2015-Mar 2015)**
  - Untethered
  - Onboard Computing
  - 6DOF arms
  - As V4, but 7DOF arms (lower 3 joints electric)
Infrastructure

- 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

- **Atlas V3**
- **Atlas V4**
- **Atlas V5**
- **Ubuntu 12.04**
- **Ubuntu 14.04**
- **ROS Fuerte**
- **ROS Groovy**
- **ROS Hydro**
- **ROS Indigo**
- **rospack**
- **catkin**

**Timeline**:
- **Jan 14**: VRC, DRC Trials
- **Jan 15**: DRC Trials, DRC Finals
- **Jan 16**:
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](https://github.com/team-vigir/remotelaunch)
Infrastructure – Deployment to multiple machines

Launch scripts for each machine

```
# theoden
roslaunch vigir_atlasbringup 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

<table>
<thead>
<tr>
<th>Simulator/Robot</th>
<th>Locomotion</th>
<th>Manipulation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas/BDI/DRCsim</td>
<td>(Yes)</td>
<td>No</td>
<td>Only with BDI lib</td>
</tr>
<tr>
<td>Atlas/IHMC/SCS</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Atlas/IHMC/DRCsim</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>Coming soon :)</td>
</tr>
<tr>
<td>Valkyrie/IHMC/DRCsim</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>Coming soon</td>
</tr>
<tr>
<td>Thor-Mang/Gazebo4</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Components - Controls

Hardware

Robot Controller  LIDAR(s)  Camera(s)  IMU
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

Hardware

- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU

State Estimation

State Estimator
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

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU

State Estimation
- State Estimator
Components - Perception

- Perception
  - World model server
  - LIDAR Filter

- State Estimation
  - State Estimator

- Hardware
  - Robot Controller
  - LIDAR(s)
  - Camera(s)
  - IMU
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

- Standard scan too big for 1500 Byte UDP limit

[github.com/team-vigir/vigir_perception/tree/master/vigir_filteredlocalized_scan_utils]
Perception – Compressing LIDAR Data

- 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
  
  [Link](https://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

- 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

Perception
- World model server
- LIDAR Filter

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
Components – Footstep Planner

Perception
- World model server
- LIDAR Filter

Planning
- Footstep Planner

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
Footstep Planner

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

\[ s' = (x', y', \theta') \]
\[ s = (x, y, \theta) \]
\[ a = (\Delta x, \Delta y, \Delta \theta) \]

Discrete Foot Placements

Successor Set

Footstep Planner

- 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 Link: github.com/team-vigir/vigir_footstep_planning_core]
Terrain Model Generator

- 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)

Video: Terrain Model Generator
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

- 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

[Diagram of Footstep Planner: Plugins]
Footstep Planner: Example

Video: Traversing Chevron Hurdles
Footstep Planner: Interactivity

Video: Interactive Planning

Drag goal
Components – Footstep Planner

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU

Perception
- World model server
  - LIDAR Filter

Planning
- Footstep Planner

State Estimation
- State Estimator
Components - Manipulation

Perception
- World model server
- LIDAR Filter

Planning
- Footstep Planner
- Manipulation Planner

Template Manipulation
- OT Server
- Manipulation Controller

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
Motion Planning - Requirements

- **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

- 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

scan

Lidar Octomap Updater

filtered_scan

OCS Teleop

JointTrajectoryAction

vigir_move_group_manipulation_action_capability

vigir_move_group_octomap_access_capability

vigir_move_group_robot_state_retrieval_capability

Standard move_group plugins

octomap

Manipulation Controller

FlexBE Behaviors

robot_state

Topic

Service

Action
### Planning - Capabilities

- **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

---

**Python Code: vigir_move_group**

```python
# Example:

import vigir_move_group

# Initialize

move_group = vigir_move_group.VigirMoveGroup()

# Add move_group capability

move_group.add_capability(
    type='type
    geometry_msgs/Pose[] target_poses
    string reference_point_frame
    geometry_msgs/Pose reference_point

# Start

move_group.start()
```

---

[github.com/team-vigir/vigir_manipulation_planning/tree/master/vigir_move_group](https://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 Link](https://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](https://github.com/tu-darmstadt-ros-pkg/rosmatlab)
Manipulation – Reaching motion using Drake Integration

Video: Reaching Motion
Components – FlexBE Behavior Executive

Perception
- World model server
- LIDAR Filter

Planning
- Footstep Planner
- Manipulation Planner

Template Manipulation
- OT Server
- Manipulation Controller

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
Components – FlexBE Behavior

Executive
Behavior Executive - High-Level Approach

- Communication constraints
- Limited time
- Complex robot system

Flexible Robot-Operator Collaboration

- Unstructured environment
- Complex tasks
- Robustness important

Motivates high degree of robot autonomy

Motivates high degree of operator support
Behavior Executive - High-Level Approach

- 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
“Flexible Behavior Engine”

- 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):

```
- on_enter
- execute
- on_exit
```

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

![Diagram](image)

- Close Fingers → Lift Object
  - *success*
  - *missed*

- Lift Object → Try Alternative

**Autonomy**
- Off
- → Low
- → High
- → Full
FlexBE – Data Input

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

---

**Diagram:**
- Remote Robot
  - Behavior
    - InputState
      - deserialize data
    - ...
- Communication channel
  - Input Request
  - Input Data
- Operator Control Station
  - Behavior Input
    - serialize data
  - any widget for robot control
  - Provide requested data
  - Trigger user dialog
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

- 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 Editor - Team ViGIR Edition

Onboard Status: running

Behavior Dashboard StateMachine Editor Runtime Control Configuration

Show Terminal

Praying Mantis Calibration (root) ➔ Perform Checks

Gen_Tail to 50% Limits
CalculationState

Move to 90% Joint Limits
MoveStartingPointState

Input Data:
trajectories_00 (trajectories)

Output Data:
no output keys

Manipulate_Limits
StateMachine
3 states

done

-reached

-failed

Onboard requested outcome: reached

Stop Execution (stopped behaviors can't be resumed)

Lock Behavior
At level: Move_to_90%_Joi...

Sync
ROS
Delay
State

Behavior Feedback

[3:59:07 PM] Moving both Arms group to starting point.
[3:59:03 PM] Recording topics to /home/cornel/mantis_logs/mantis_calibration_full_run_2015-03-23-16-02.bag
[3:58:59 PM] Execution has started. Please confirm transition to first state.
[3:58:58 PM] → Starting new behavior...
[3:57:16 PM] Stopping behavior...

Documentation

MoveToStartingPointState
Uses moveIt to plan and move to the first point of a given arm trajectory.

Parameter Values:
FlexBE – Beyond DRC application

Video: FlexBE Demo
Components – Comms Bridge

Perception
- World model server
- LIDAR Filter

Planning
- Footstep Planner
- Manipulation Planner

Template Manipulation
- OT Server
- Manipulation Controller

Behavior Executive
- FlexBE

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
Components – Comms Bridge

Perception
- World model server
- LIDAR Filter

Planning
- Footstep Planner
- Manipulation Planner

Template Manipulation
- OT Server
- Manipulation Controller

Behavior Executive
- FlexBE

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU

Comms Bridge
- Comms Bridge
- OCS Master
- Onboard Master
- Comms Bridge
- Comms Bridge
- Comms Bridge
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

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU

Perception
- World model server
- LIDAR Filter

Planning
- Footstep Planner
- Manipulation Planner

Template Manipulation
- OT Server
- Manipulation Controller

Behavior Executive
- FlexBE

State Estimation
- State Estimator

Comms Bridge
- Comms Bridge
- OCS Master
- Onboard Master

FlexBE
- Comms Bridge
- Comms Bridge
- Comms Bridge
- Comms Bridge
- OCS Master
- Onboard Master

Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
Components - OCS

OCS
- Main View
- Camera View
- World model server
- OT Server

Perception
- World model server
- LIDAR Filter
- Footstep Planner
- Manipulation Planner
- LIDAR Filter
- Flexible Operation Planner
- World model server

Planning
- Footstep Planner
- Manipulation Planner
- OT Server
- Manipulation Controller

Template Manipulation
- OT Server
- Manipulation Controller

Behavior Executive
- FlexBE

State Estimation
- State Estimator

Hardware
- Robot Controller
- LIDAR(s)
- Camera(s)
- IMU
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
Components – Tutorial video

Manipulation Control Approach for Remote Humanoid Robots under Human Supervision

Video: Open Source Tutorial
Open Source Tutorial

Team ViGIR’s software using Team Hector’s robot "Johnny" in Gazebo Simulator
Work in Progress - Behavior Synthesis

ROSCon 2015 Example

Statemachine

Open this Statemachine  Display synthesis

Synthesis
Initial Conditions:

stand_prep

Goal:
pickup_object

Synthesize  This will delete the current content!

Outcomes
finished:  Inherit
failed:  Inherit
Work in Progress - Behavior Synthesis

- 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
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

Video: Behavior Synthesis Demo
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

Video: Day 1 Pt.1
Video: Day 1 Pt.2
DRC Finals – Day 1
DRC Finals – Day 1
DRC Finals – Day 1
DRC Finals – Day 1

- Flawless Driving
- Comms bridge setup issue
  - Behavior control
  - Footstep planning
- Switch to teleop mode
- Slow but reliable
DRC Finals – Day 2

Video: 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

- 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

- 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

- 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

- 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


- YouTube playlist with manipulation examples
Open Source Driving Controller Concept for Humanoid Robots: Teams Hector and ViGIR at DARPA Robotics Challenge 2015

Video: Driving Approach

Alberto Romay, Achim Stein, Martin Oehler, Alexander Stumpf, Stefan Kohlbrecher and Oskar von Stryk
Simulation, Systems Optimization and Robotics Group, CS Dept. Technische Universität at Darmstadt

David C. Conner
TORC Robotics
## Additional Material – DRC Communication constraints

<table>
<thead>
<tr>
<th></th>
<th>Uplink (to OCS)</th>
<th>Downlink (to robot)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VRC</strong></td>
<td>Total ~115 kB for 30 minutes. 500 ms latency</td>
<td>Total, ~7 MB for 30 minutes. 500 ms latency</td>
<td>Worst case (20% of scenarios)</td>
</tr>
<tr>
<td><strong>Trials</strong></td>
<td>1 MB/s, 50ms latency</td>
<td>1 MB/s, 50 ms latency</td>
<td>Good comms</td>
</tr>
<tr>
<td></td>
<td>100 kB/s, 500 ms latency</td>
<td>100 kB/s, 500 ms latency</td>
<td>Bad comms</td>
</tr>
<tr>
<td><strong>Finals</strong></td>
<td>1.2 kB/s</td>
<td>1.2 kB/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300 Mbit/s</td>
<td></td>
<td>Outages of 1-30 seconds after robot traverses door</td>
</tr>
</tbody>
</table>

VRC Rules (pdf)
Trials Rules (pdf)
Finals Rules (pdf)