Lessons Learned During **TRADR**

Search & Rescue Robotics Using ROS

M. Pecka, S. Caccamo*, R. Dubé, V. Kubelka, D. Reuter*

TRADR GitHub

tradr-project.eu

September 21st-22nd, 2017
Vancouver, Canada

Outline

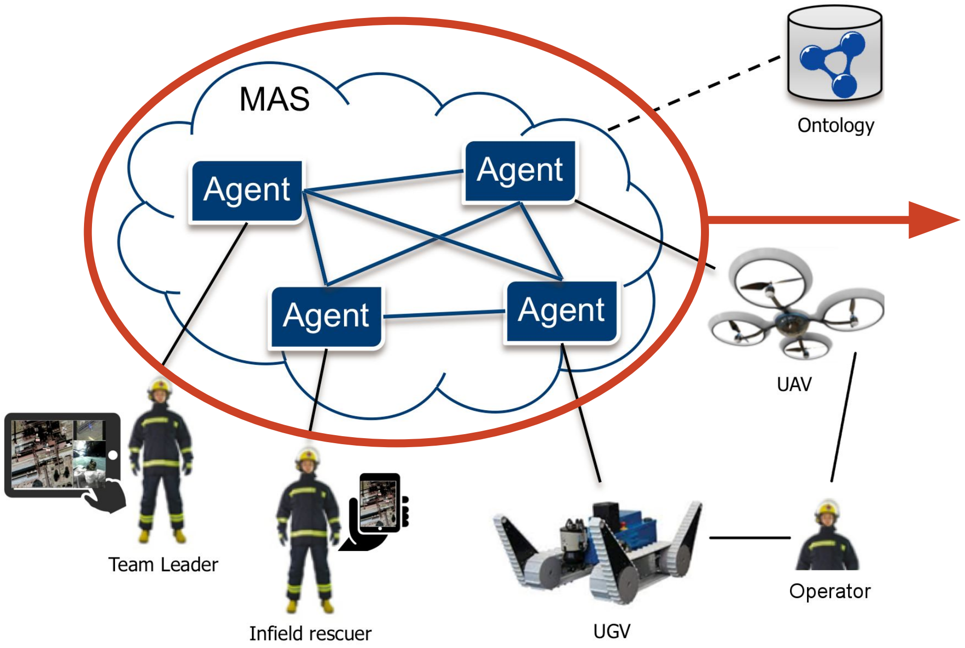
1. **TRADR** project intro
2. Our ROS tools and packages
3. Advanced functionalities of our robots
4. Our hardware experience
5. Bonus

TRADR Project

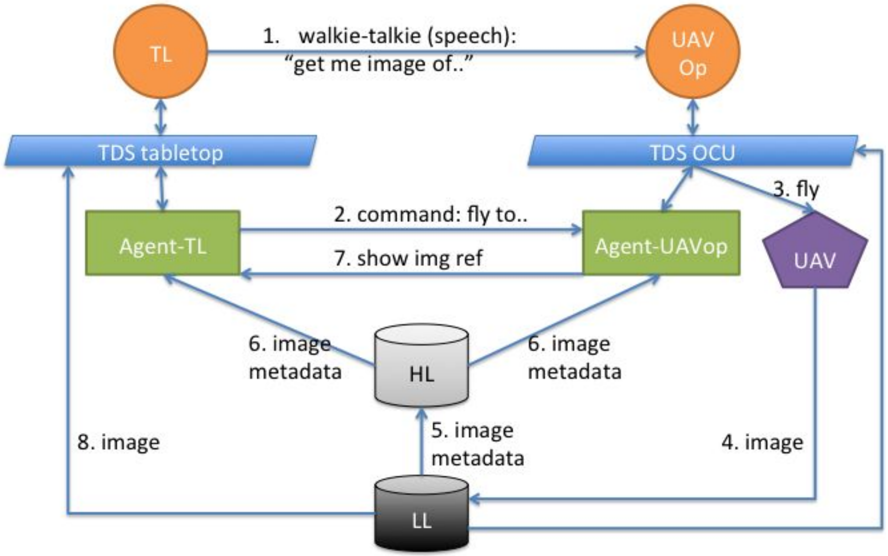
(EU - FP7 Framework)

- **UGVs** and **UAVs** as a part of human-robot rescue teams
- **Exploration, detection** and **monitoring** in a disaster area, sample pickup missions
- Strong end-user orientation
- **Human** ↔ **Robot** cooperation, every team member does what he’s better at
- Robots need to work in (unfriendly) real world without markers, pretrained models etc.

Team Members



System architecture



UGVs

- Custom-made tracked vehicles
- Ca. 25 kg (55 lbs)
- Wi-Fi connection (Ubiquiti Bullet, 5 GHz)
- Multiple cameras, tilting lidar, tactile/smoke/gas/radioactivity/.. sensors
- No GPS (intentionally)

UAVs

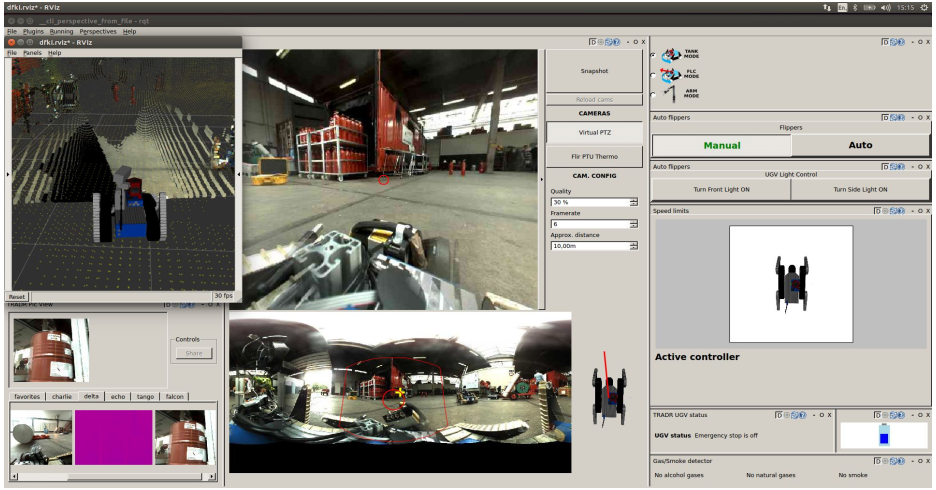
- More models, all commercial
- Live streaming of RGB and IR cameras
- D-GPS

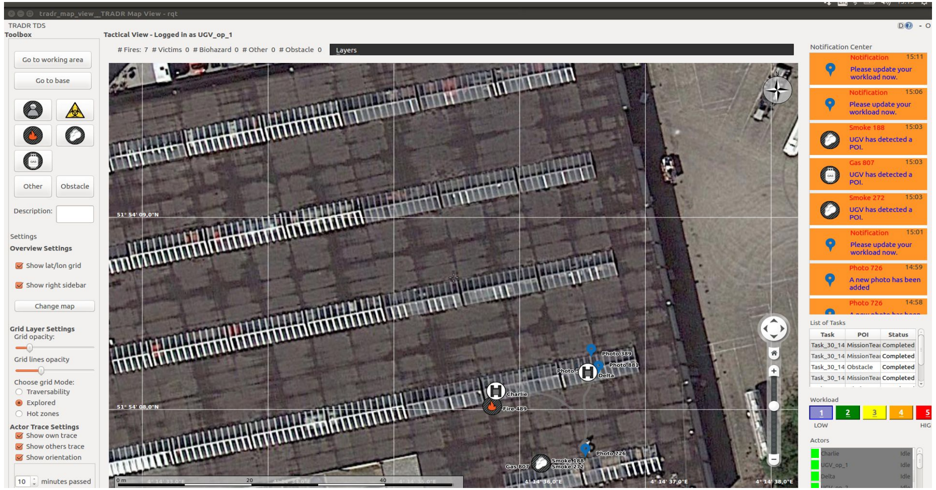
OCU (Operator Control Unit)

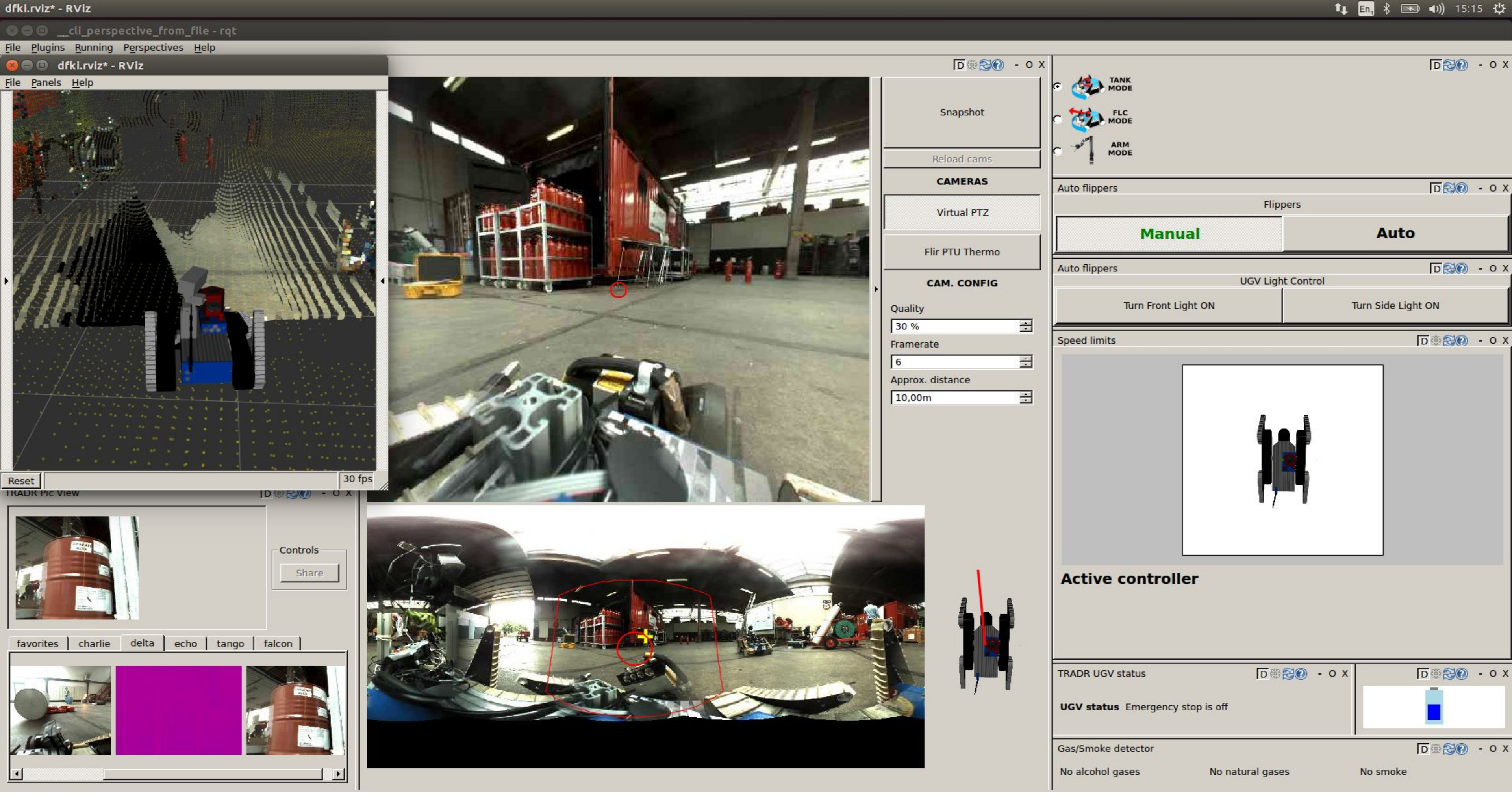
- Multiple RQT plugins
- Modular design for different UGV configurations (e.g. presence of a manipulator)

TDS (Tactical Display System)

- Multiple RQT plugins
- Interface between high-level reasoning and humans





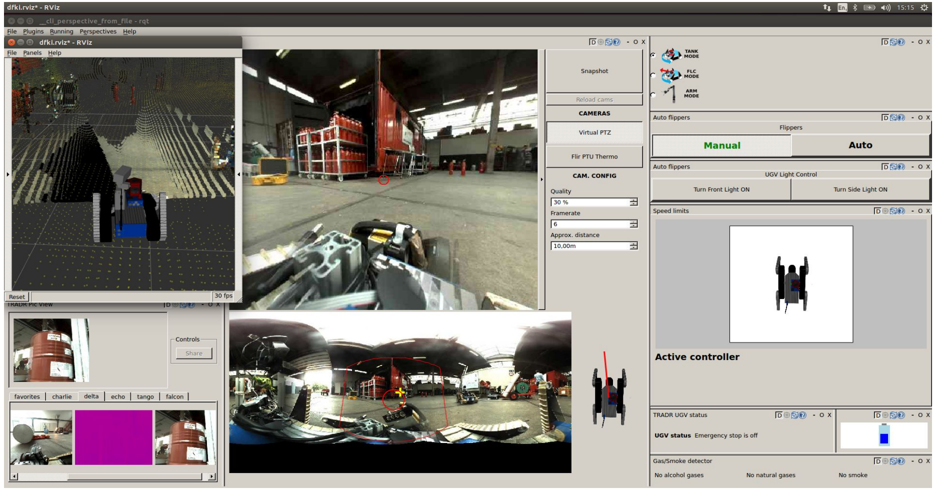

dfki.rviz* - RViz
15:15
__cli_perspective_from_file - rqt
File Plugins Running Perspectives Help
dfki.rviz* - RViz
File Panels Help
Reset
30 fps
TRADR Pic View
Controls
Share
favorites
charlie
delta
echo
tango
falcon
Snapshot
Reload cams
CAMERAS
Virtual PTZ
Flir PTU Thermo
CAM. CONFIG
Quality
30 %
Framerate
6
Approx. distance
10,00m
TANK MODE
FLC MODE
ARM MODE
Auto flippers
Flippers
Manual
Auto
Auto flippers
UGV Light Control
Turn Front Light ON
Turn Side Light ON
Speed limits
Active controller
TRADR UGV status
UGV status Emergency stop is off
Gas/Smoke detector
No alcohol gases
No natural gases
No smoke

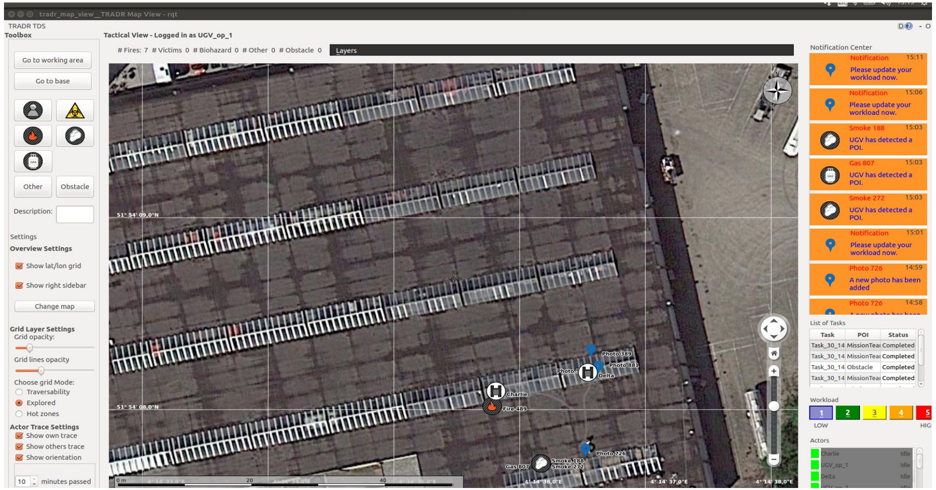
OCU (Operator Control Unit)

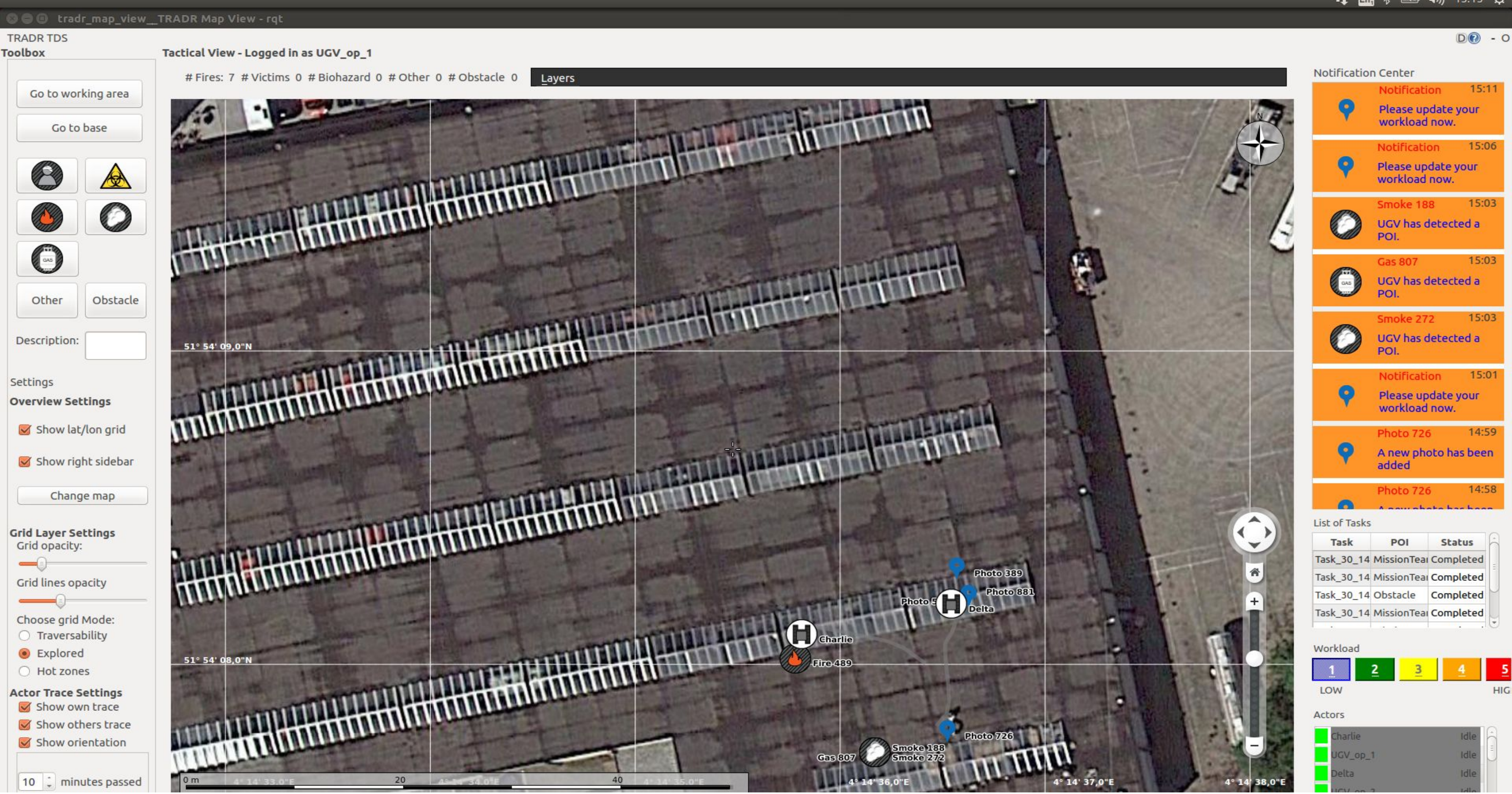
- Multiple RQT plugins
- Modular design for different UGV configurations (e.g. presence of a manipulator)

TDS (Tactical Display System)

- Multiple RQT plugins
- Interface between high-level reasoning and humans







tradr_map_view__TRADR Map View - rqt
TRADR TDS
Toolbox
Tactical View - Logged in as UGV_op_1
Fires: 7 # Victims 0 # Biohazard 0 # Other 0 # Obstacle 0
Layers
Go to working area
Go to base
Other
Obstacle
Description:
Settings
Overview Settings
Show lat/lon grid
Show right sidebar
Change map
Grid Layer Settings
Grid opacity:
Grid lines opacity
Choose grid Mode:
Traversability
Explored
Hot zones
Actor Trace Settings
Show own trace
Show others trace
Show orientation
10 minutes passed
51° 54' 09,0"N
51° 54' 08,0"N
0 m
20
40
4° 14' 36,0"E
4° 14' 37,0"E
4° 14' 38,0"E
Photo 389
Photo 881
Delta
Charlie
Fire 489
Photo 726
Gas 807
Smoke 188
Smoke 272
Notification Center
Notification 15:11
Please update your workload now.
Notification 15:06
Please update your workload now.
Smoke 188 15:03
UGV has detected a POI.
Gas 807 15:03
UGV has detected a POI.
Smoke 272 15:03
UGV has detected a POI.
Notification 15:01
Please update your workload now.
Photo 726 14:59
A new photo has been added
Photo 726 14:58
List of Tasks
Task POI Status
Task_30_14 MissionTea Completed
Task_30_14 MissionTea Completed
Task_30_14 Obstacle Completed
Task_30_14 MissionTea Completed
Workload
1 2 3 4 5
LOW
Actors
Charlie Idle
UGV_op_1 Idle
Delta Idle


UGV operators
UAV operator
Team leader

ROS Tools & Packages

Workspace Layout and Versioning

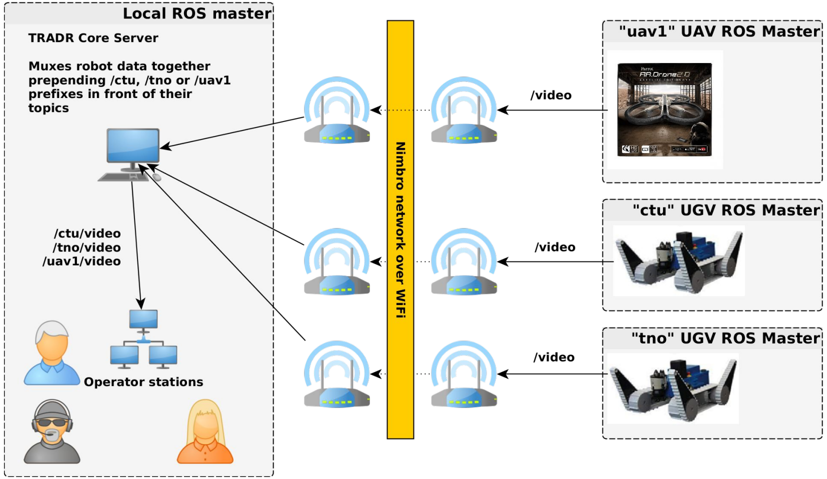
- **Problem**: >100 ROS packages
  - they need to be organized and separable
- We use git, but not submodules
- Ca 15 git repos
- Some machines only need some repos

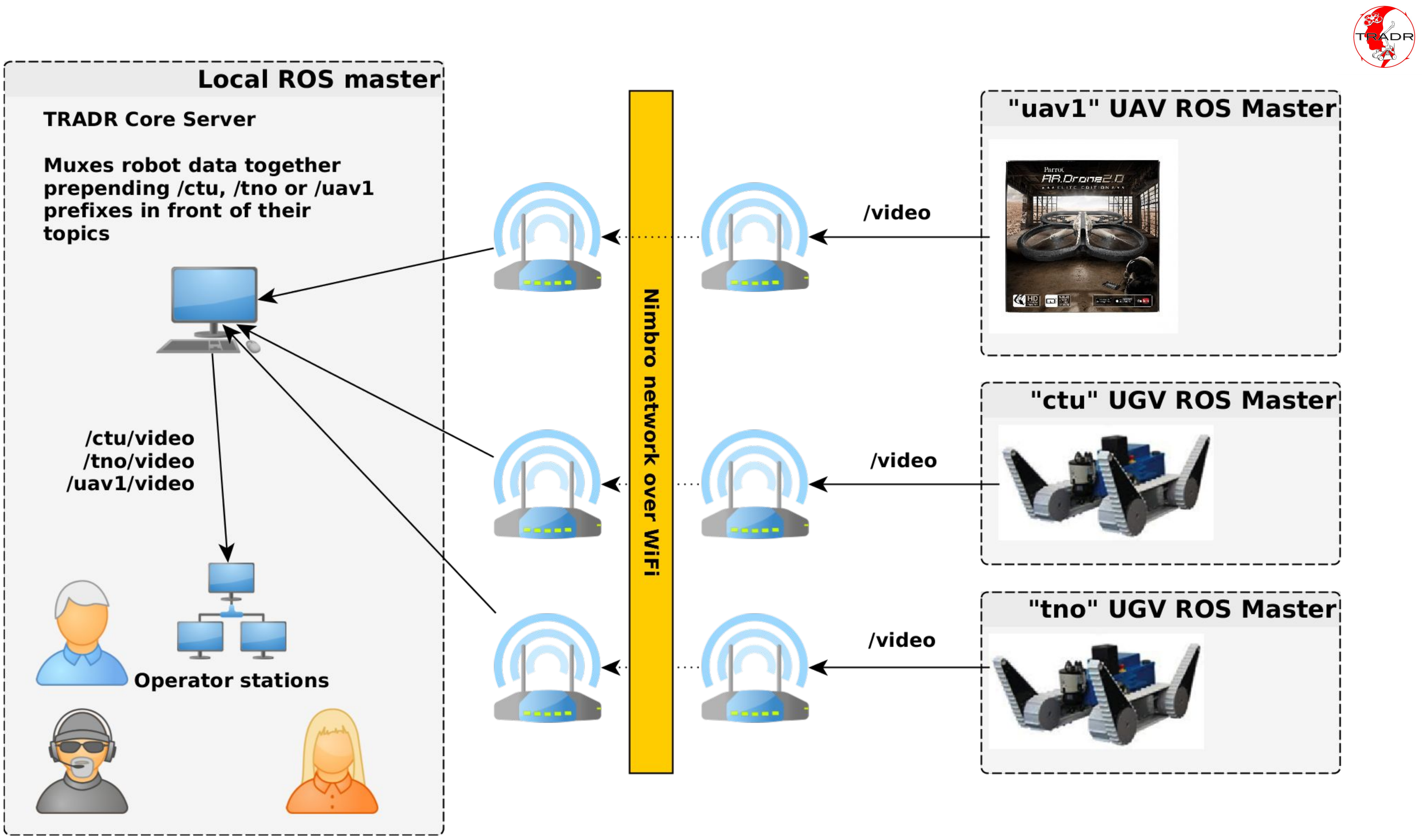
Still not there (do better than us!)

- Autoinstall scripts for developers were offered, but nobody uses them
  - Complicates adding non-ROS dependencies and repo-repo relations
  - They're used for monthly integration tests
- We partly use rosinstall
  - Still too confusing for some developers
- Free choice of build tool
  - catkin_make / catkin tools / catkin simple
- We use our own Gitlab installation
  - Has both pros and cons

Multimaster (customizations released)

- **Problem:** Too much data transmitted over wireless, TCP congestion
- Nimbro_network with customizations
  - Support latched and bi-directional topics
- Problems: no auto connect/disconnect, network resilience still isn't perfect

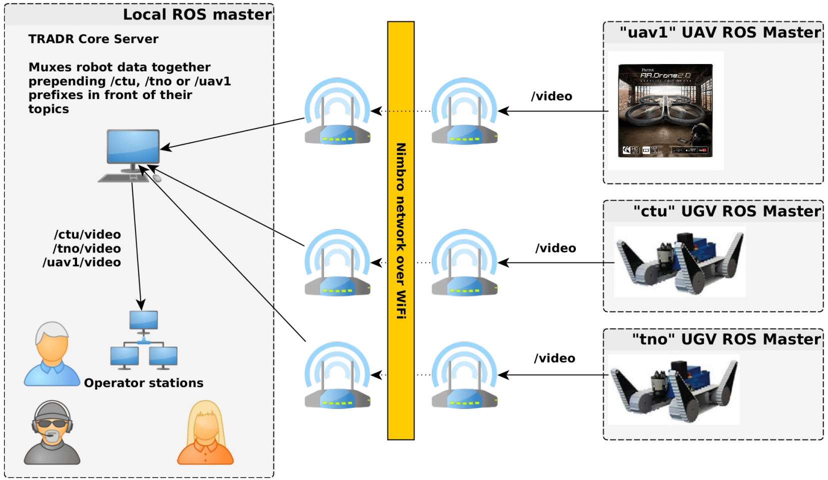




Local ROS master
TRADR Core Server
Muxes robot data together prepending /ctu, /tno or /uav1 prefixes in front of their topics
/ctu/video
/tno/video
/uav1/video
Operator stations
Nimbro network over WiFi
"uav1" UAV ROS Master
/video
Parrot
AR.Drone 2.0
"ctu" UGV ROS Master
/video
"tno" UGV ROS Master
/video
TRADR

Multimaster (customizations released)

- **Problem:** Too much data transmitted over wireless, TCP congestion
- Nimbro_network with customizations
    - Support latched and bi-directional topics
- Problems: no auto connect/disconnect, network resilience still isn't perfect



- YAML configs and launch file generator (not released, but planned)
- Easy configuration of video, dynamic reconfigure and actions

ctu-robot.yaml

```yaml
robot:
  hostname: ctu-robot
  prefix: ugv1

ports:
  to_core:
        tcp: 17011
        udp: 17012
        service: 17013
  to_robot:
        tcp: 17014
        udp: 17015
        service: 17016

packs:
  - base
  - teleop
  - axis_ptz
```

packs/axis_ptz.yaml

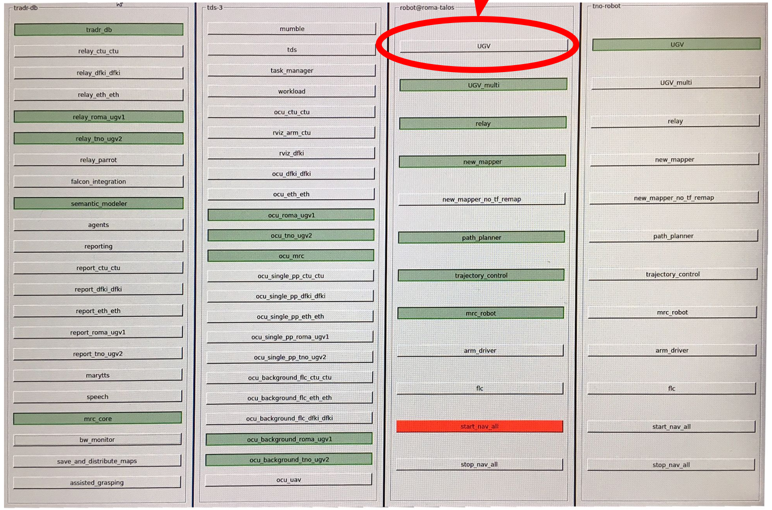
```yaml
to_robot:
  udp:
    - name: "axis/control/look_at"
    - name: "axis/control/zoom/relative"

to_core:
  video:
    - name: "axis"

  dynamic_reconfigure:
    - name: "axis/axis"

  udp:
    - name: "axis/camera/ptz"
```

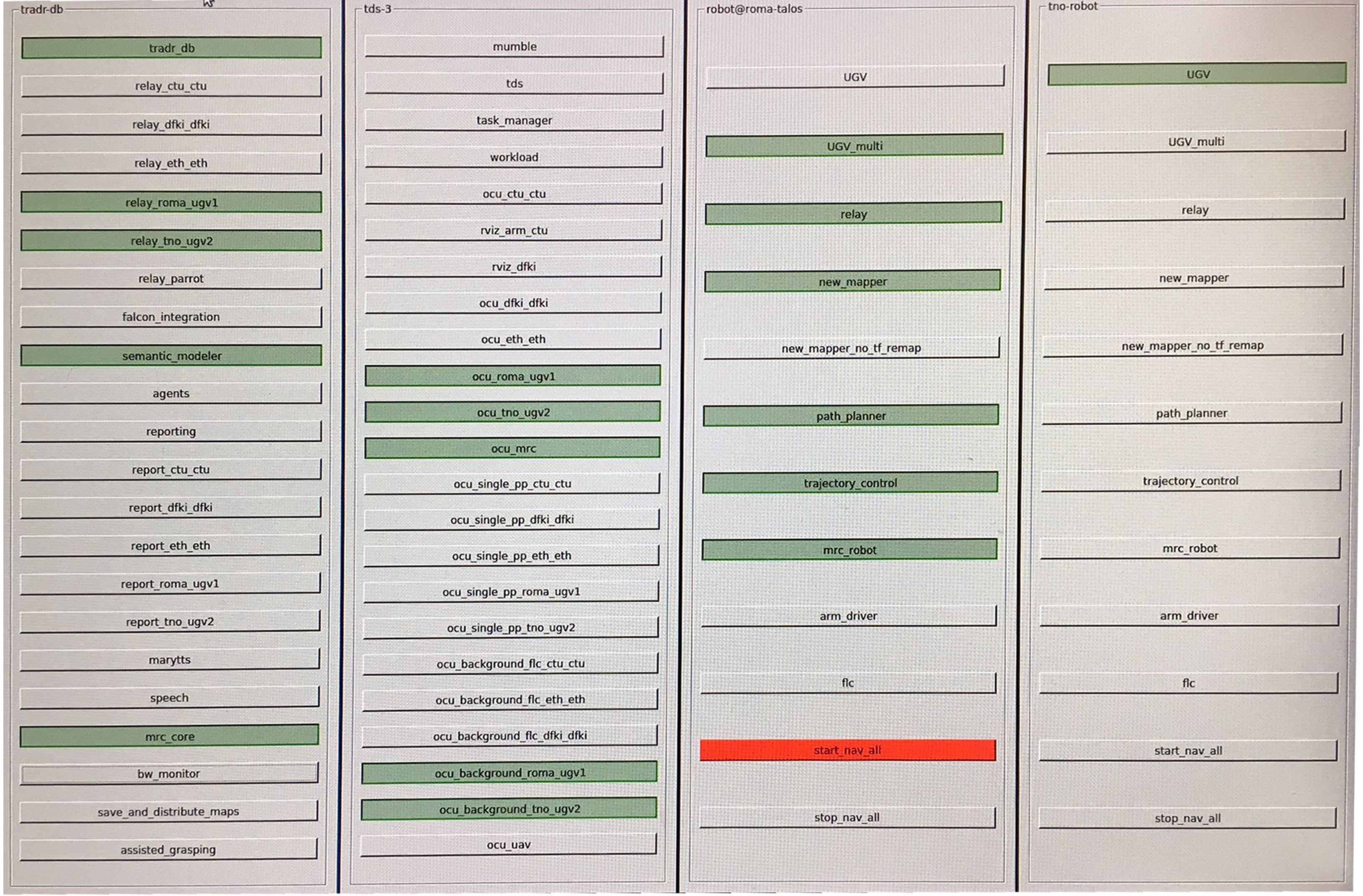
TRADR Orchestra (not released)

- **Problem:** Too many parts of the systems to be launched, some only sometimes

- Visual control over a set of scripts/launch files that run everything needed during the mission
- No “one launch file that rules them all” → it’s easy to restart parts of the system
- Status reporting (running, dead)
- Scripts run in separate `screen` sessions
- Unified deployment script (rsync)

```
main:
  {{ hostname }}:
    - name: UGV
      shell: ugv.sh
    - name: UGV_multi
      shell: ugv_multi.sh {{ ugv_prefix }}
    - name: relay
      shell: relay.sh {{ ugv_prefix }} {{ core_hostname }}
    - name: new_mapper
      shell: new_mapper.sh {{ ugv_prefix }} {{ localize_against_map }}
```



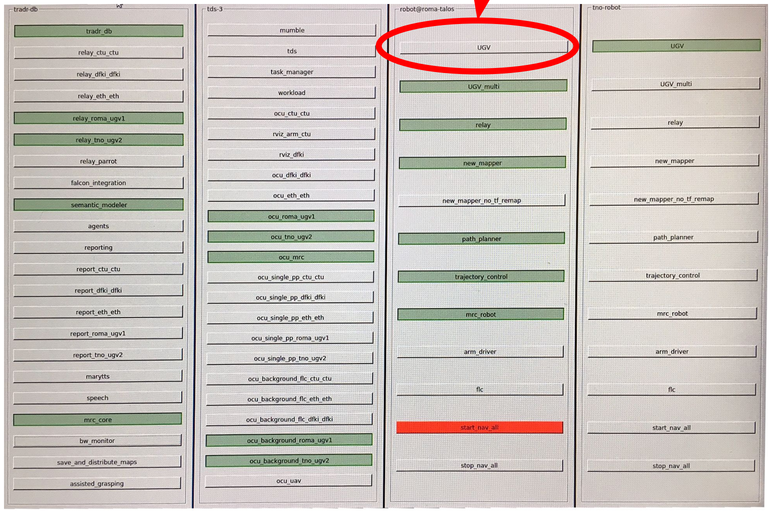

tradr-db
tradr_db
relay_ctu_ctu
relay_dfki_dfki
relay_eth_eth
relay_roma_ugv1
relay_tno_ugv2
relay_parrot
falcon_integration
semantic_modeler
agents
reporting
report_ctu_ctu
report_dfki_dfki
report_eth_eth
report_roma_ugv1
report_tno_ugv2
marytts
speech
mrc_core
bw_monitor
save_and_distribute_maps
assisted_grasping
tds-3
mumble
tds
task_manager
workload
ocu_ctu_ctu
rviz_arm_ctu
rviz_dfki
ocu_dfki_dfki
ocu_eth_eth
ocu_roma_ugv1
ocu_tno_ugv2
ocu_mrc
ocu_single_pp_ctu_ctu
ocu_single_pp_dfki_dfki
ocu_single_pp_eth_eth
ocu_single_pp_roma_ugv1
ocu_single_pp_tno_ugv2
ocu_background_flc_ctu_ctu
ocu_background_flc_eth_eth
ocu_background_flc_dfki_dfki
ocu_background_roma_ugv1
ocu_background_tno_ugv2
ocu_uav
robot@roma-talos
UGV
UGV_multi
relay
new_mapper
new_mapper_no_tf_remap
path_planner
trajectory_control
mrc_robot
arm_driver
flc
start_nav_all
stop_nav_all
tno-robot
UGV
UGV_multi
relay
new_mapper
new_mapper_no_tf_remap
path_planner
trajectory_control
mrc_robot
arm_driver
flc
start_nav_all
stop_nav_all

TRADR Orchestra (not released)

- **Problem:** Too many parts of the systems to be launched, some only sometimes

- Visual control over a set of scripts/launch files that run everything needed during the mission
- No “one launch file that rules them all” → it’s easy to restart parts of the system
- Status reporting (running, dead)
- Scripts run in separate `screen` sessions
- Unified deployment script (rsync)

```
main:
  {{ hostname }}:
    - name: UGV
      shell: ugv.sh
    - name: UGV_multi
      shell: ugv_multi.sh {{ ugv_prefix }}
    - name: relay
      shell: relay.sh {{ ugv_prefix }} {{ core_hostname }}
    - name: new_mapper
      shell: new_mapper.sh {{ ugv_prefix }} {{ localize_against_map }}
```

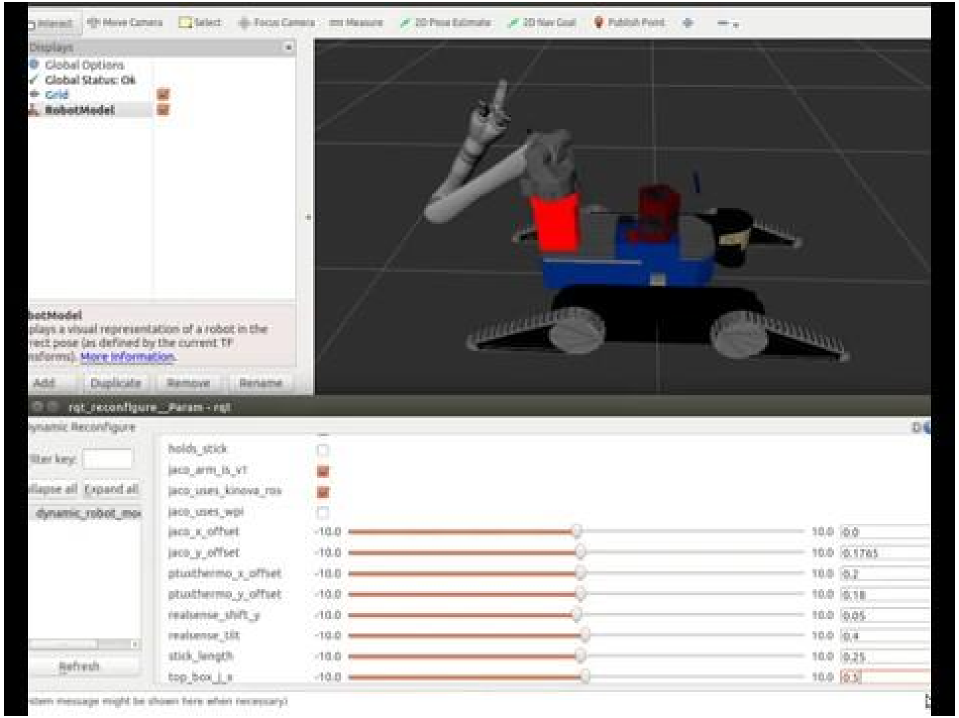


M. Pecka, S. Caccamo, R. Dubé , V. Kubelka, D. Reuter

Dynamic URDF Model Generation (not released)

- **Problem:** Additional sensors/equipment differ for each mission
- Laser filtering, camera image masking, useful for teleoperation, coarse interactive calibration of sensor positions
- Implemented using Xacro and dynamic_reconfigure
- No need to restart drivers to add/remove sensors/equipment

```
<robot name=”NIFTi”>
  <xacro:arg name="has_jaco_arm" default="0" />
  <xacro:if value="$(arg has_jaco_arm)">
    <xacro:include filename="$(find kinova_description)/
       urdf/j1n6a300.xacro"/>
    <xacro:j1n6a300 base_parent="jaco_base_helper" />
    ....
```



Tracked Robot Simulation in Gazebo (released)

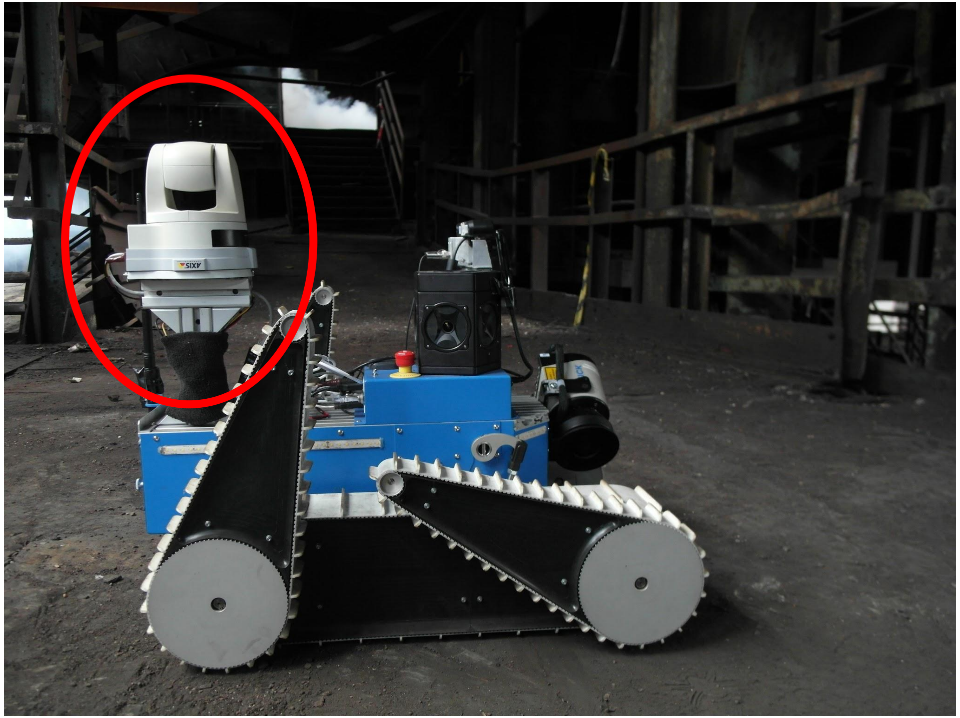
- **Problem:** (Proper) tracked vehicle simulation hasn't been available in Gazebo

- Teaser for our IROS presentation "Fast Simulation of Vehicles with Non-deformable Tracks" (WeCT4.2)
- Enables us doing Reinforcement learning with the UGV
- The simulation model is also dynamically generated from Xacro template

camera_info_manager for Zoom Cameras (released)

- **Problem:** We have a zoom camera and we need its calibration data

- Python only (good for e.g. IP cameras)
- Basic support for zoom cameras in camera_info_manager_py
- Two approaches:
  - Calibrate at several zoom levels
  - A trigonometry-only solution
- Integration into general ROS packages is complicated



Multiple PTZ Camera RQT Widget

- **Problem:** Need to be controlled by 1 operator
- Automatic detection of running cameras (by getting # of publishers)
- Abstraction over multiple pan-tilt-zoom control APIs
- 360° panoramic view for quick orientation and visualization of field of view
- Basic control over video stream and camera properties
- The thermo camera view allows setting color palette properties

M. Pecka, S. Caccamo, R. Dubé , V. Kubelka, D. Reuter

High-level and Low-level Databases

Low-level DB

- MongoDB (NoSQL) + GridFS for large files
- New database for each mission
- ROS integration using catkin-generated `get*` and `put*` services for various data types

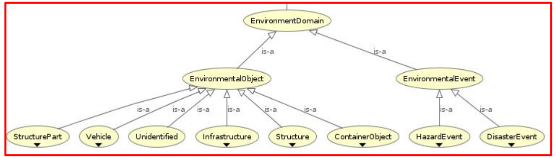
```
CMakeLists.txt:
add_tradrdb_msg(Image sensor_msgs Image)

autogen_srv/getImage.srv:
string object_id
---
sensor_msgs/Image data
```

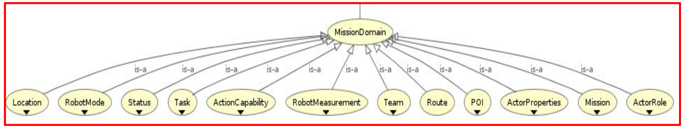
Semantic modelers

- Transform low-level (raw) data into high-level (semantic) information
  - Fire/victim detector with spatial clustering
  - Actor poses
  - Operator workload
- Should not work with “live” ROS streams, but with data from MongoDB

High-level DB

- SparQL semantic database with ontology
- Used by “agents” in GOAL framework and tactical displays





Easy-to-use C++ Tensorflow (released)

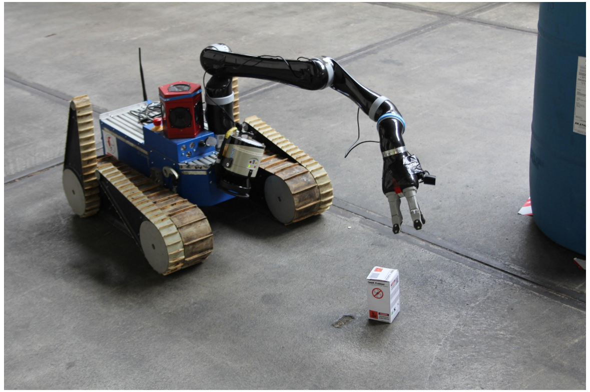
- Our hacky way to easily use Tensorflow in ROS and C++
  - No need to use/install Bazel - compilation is done using catkin
  - Requires C++11 support in compiler (not in Trusty by default)
- (Mis)uses the pip-installed version of Tensorflow for Python
- It isn't optimized to recent CPUs, so performance is not good enough for training neural networks, but sufficient enough for running simple ones

package.xml (format 2)

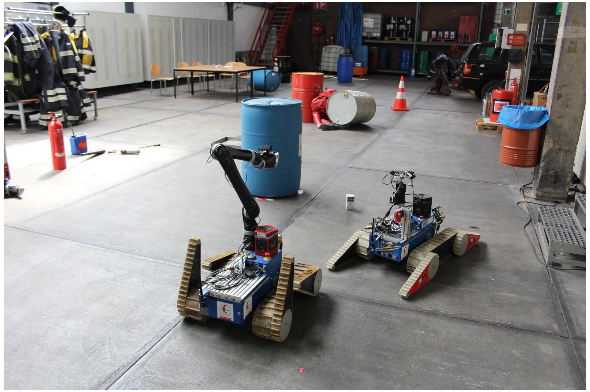
```
...
<build_depend>tensorflow_ros</build_depend>
<exec_depend>tensorflow_ros</exec_depend>
...
```

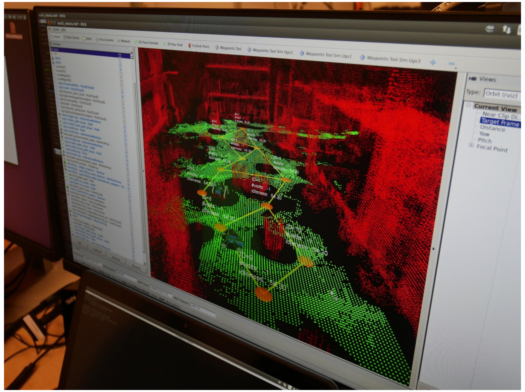
CMakeLists.txt

```
find_package(catkin REQUIRED COMPONENTS
tensorflow_ros)

add_library(tensorflow_executor
src/TensorflowGraphExecutor.cpp)

target_link_libraries(tensorflow_executor
${catkin_LIBRARIES})

set_target_properties(tensorflow_executor PROPERTIES
CXX_STANDARD 11 CXX_STANDARD_REQUIRED YES
CXX_EXTENSIONS NO)
```



TRADR Functionalities

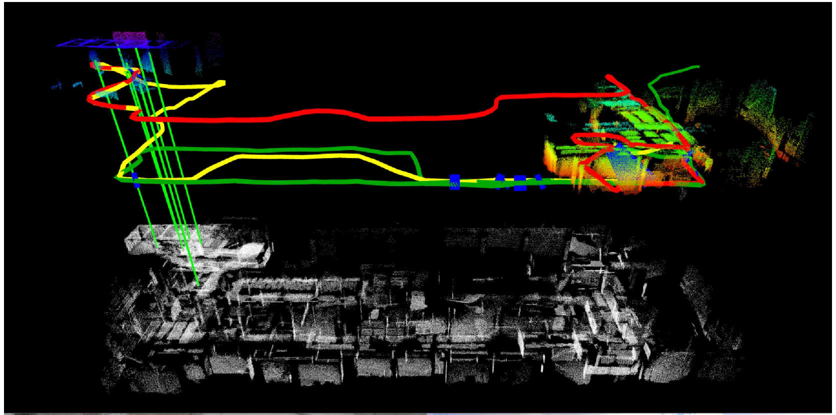




3D Laser Mapping (released)

SLAM system for multiple robots equipped with 3D LiDARs

- Back-end: Incremental pose-graph optimization
- ICP-based LiDAR odometry constraints
- Loop-closures via SegMatch - a segment extraction and matching approach
- Framework available open-source with demonstrations
- More information at IROS SLAM 1 session MoBT7.2

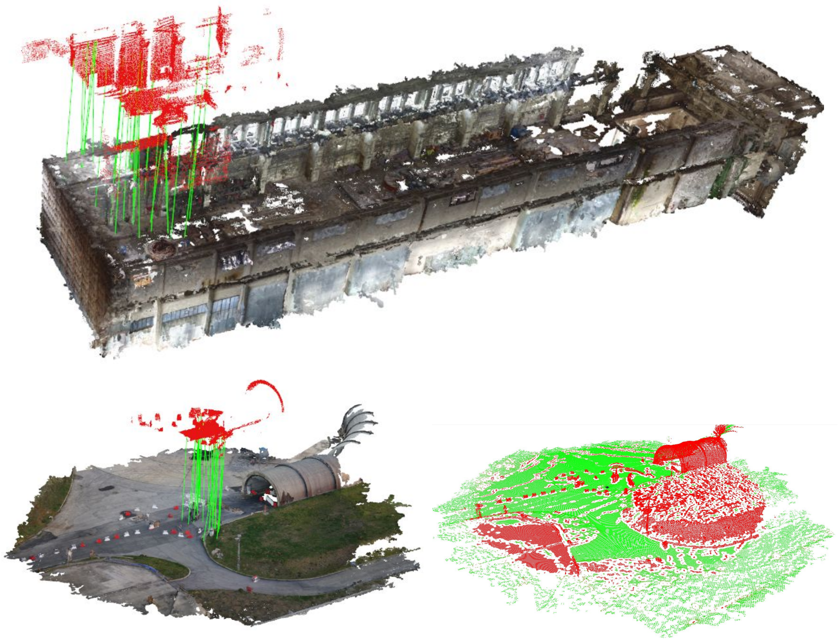


https://github.com/ethz-asl/segmatch

Heterogeneous 3D Map Merging

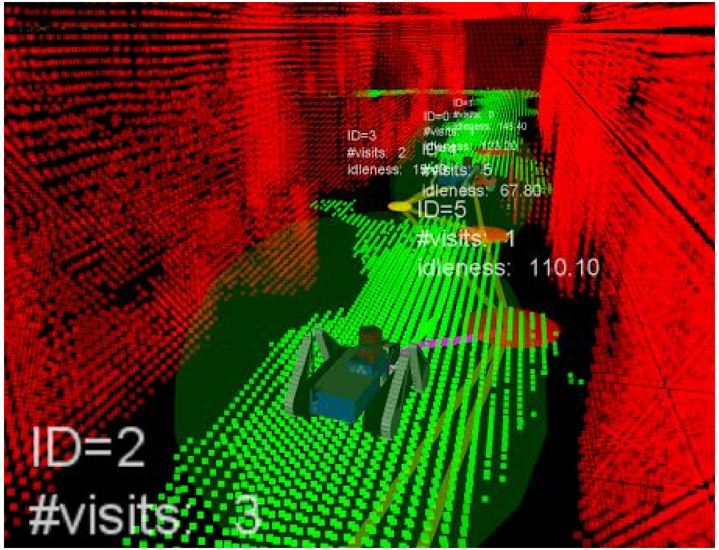
Global Registration of maps from 3D ground LiDAR and aerial reconstruction:

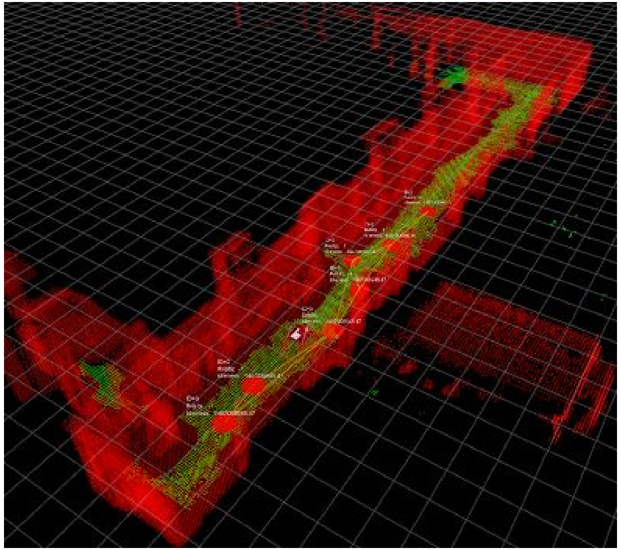
- Inputs aerial 3D reconstructions from camera and LiDAR sub-map from UGV
- Key-point descriptor-based global registration
- Registered aerial map enables continuous localization, traversability analysis, and path planning

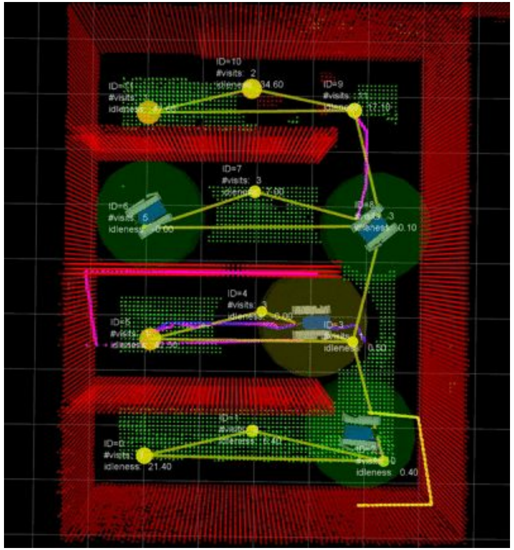


Gawel, A., Dubé, R., Surmann, H., Nieto, J., Siegwart, R., & Cadena, C.. 3D Registration of Aerial and Ground Robots for Disaster Response: An Evaluation of Features, Descriptors, and Transformation Estimation." *SSRR* (2017).

3D Multi-robot Path Planning, Patrolling







More info @ **https://sites.google.com/a/dis.uniroma1.it/3d-cc-patrolling/**

Resilient Communication-Aware Motion Planner

- Mobile robots **require stable communication** with the base station during a USAR mission.
- Stochastic elements in radio signal propagation and possibility of unpredictable events or hardware failures often lead to **signal loss**.

Disconnected robots are either **abandoned** or attempt to autonomously **back-trace their way** to the base station.

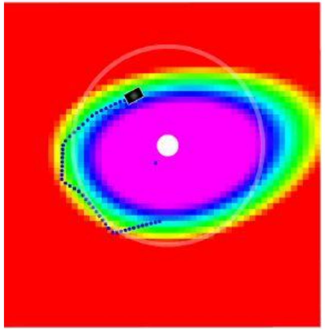
Resilient Communication-Aware Motion Planner

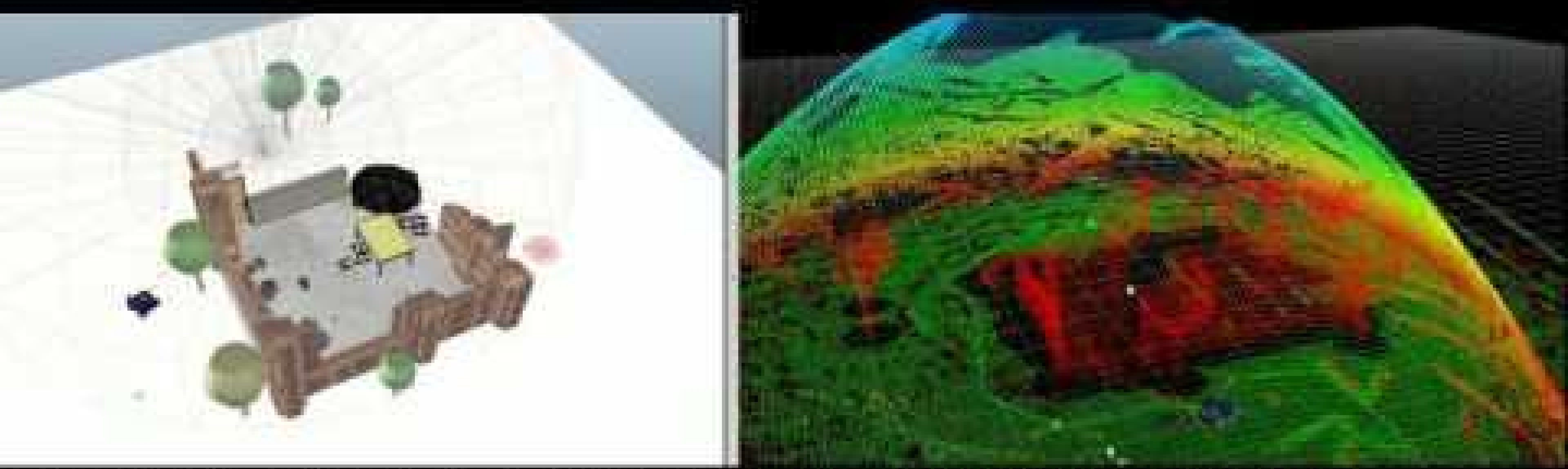
State-of-the-art methods:

- Uses a priori knowledge of the network structure (e.g. access point locations)
- Missing an effective self-repair strategy
- Do not handle complex scenarios and **rewuires some offline processing (fingerprinting/training, etc.)**

What is needed?

A system that allows to dynamically map the RSS distribution and autonomously moves the robot to a connection safe location in case of signal loss avoiding obstacles.



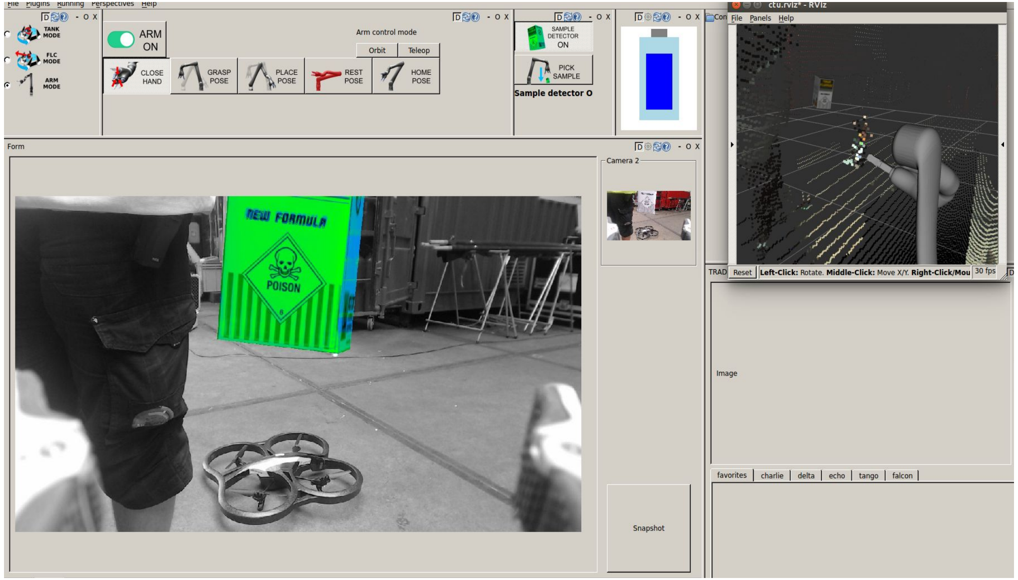


The user selects a GOAL position and the **RCAMP** promptly generates a path on that traversable area that avoids obstacles and connection loss.
The generated path is NOT the shorter distance to the GOAL but ensures good connectivity.

Assisted Grasping

- ❑ Collecting a sample during a USAR mission is a difficult task (time constraints, limited fov, delays, many DOF to control)

➔ **Solution:** vision aided cartesian arm control (using a SIFT based sample detector)



TRADR Hardware Experience

* The following slides are not objective reviews of products. They represent our experience with mentioned products.
  In most cases, we did not do comprehensive practical comparisons of available alternatives.

USB Cameras

- We were quite unlucky in our selection of USB cameras for various purposes
- When selecting a new USB camera, we would:
  - Thoroughly check its support in Linux kernel
  - Check if it has a decent ROS support
  - Run a series of tests on different battery levels of the robot
- When designing a new robot, we would:
  - have a lot of USB connectors directly on the motherboard (with screws)
  - not use unnecessary cable extenders
  - provide at least 1.0 A even to USB 2 ports
  - have enough USB 3 ports
  - have a separate USB bus for every port

- Asus Xtion Pro
  - Problems with longer cables and cable extenders
  - Problems when more cameras in one USB hub (even with external power)
    - Micro-Epsilon thermoIMAGER TIM 160
- IDS uEye
  - Picky about input power “shape” (on notebook OK, on robot 0.2 FPS)
  - Panicks with our 4.4 Linux kernel when connected at boot time
- Intel RealSense R200
  - Doesn’t work with cable extenders
- Logitech C920
  - Bulky but reliable

Wireless Communication

- We have tried several wireless communication options to achieve operational radius of the robots about 200 meters in urban areas
- WiFi 2.4 GHz
  - Not enough bandwidth for our system
  - 2.4 spectrum usually occupied
- WiFi 5 GHz 802.11a/ac/n
  - Good real bandwidth (~100 Mbps)
  - Nicely surprised in a furnace made of steel
- Military grade radio-based mesh network
  - Long range, real bandwidth ~32 Mbps
  - Illegal if you’re not government
- Custom-made 1:1 866MHz radio link
  - Range up to 500 m, bandwidth ~200 bps
  - For emergency in case fast link is lost

- Ubiquity Bullet M2/M5
  - We use it on the robots
  - Allows using non-standard 2.4 GHz bands and proprietary AirMAX protocol which gives higher bandwidth
  - Can be fixed to one AP by MAC address
- Zyxel AC1200 (USB 3 dongle)
  - Good support in Linux, but latency was higher
  - Quite powerful despite its size
- Netgear Nighthawk AC1900 (router)
  - Good performance on 5 GHz 802.11ac
  - Has problems when 2 clients are too close

Lidar / Depth Sensors

- We use them for several purposes - metric mapping (and 3D planning), virtual bumper, reactive control and visual aid for human operators
- Each use requires a different quality/type of laser scans (speed, density, completeness, coarseness of self-filtering)
- We would like to remove the mechanically tilted Sick laser, but still haven't found a comparable non-moving alternative
  - We're preparing for solid-state lidars*
- Sensors do not work in smoke, laser scanner has problems with water, wet roads and linoleum

* K. Zimmermann et al., Learning for Active 3D Mapping, ICCV 2017

- Sick LMS-300
  - Rotating ToF laser scanner
  - Range 0.2-50 m, both indoor and outdoor
  - Works reliably, but there is a lot of noise, shadows and distortions close to the scanner
  - Needs to be actively tilted for full 3D scans
- Asus Xtion Pro
  - Structured light
  - Range 0.5-5 m, only indoor
  - Good support, but probably overcome by RealSense
- Intel RealSense R200
  - Structured light + onboard IR stereo
  - Range 0.5-20 m, both indoor and outdoor
  - Indoor/outdoor switching not perfect
  - Noise with high variance in depth
  - Linux drivers not perfectly stable
- ZED camera
  - RBG stereo, computations out-of-board, requires CUDA
  - Range 0.3-20 m, both indoor and outdoor
  - Good results, but difficult to integrate (both HW and SW)

Real Deployment in Amatrice, Italy

Real Deployment in Amatrice, Italy, Sep. 2016

- Strong earthquake destroyed a big part of Amatrice
- TRADR UGVs and UAVs were part of an assessment mission in two damaged churches
- Robots provided detailed 3D models of the buildings without risking human lives

That's all folks!

Released TRADR SW

- Available on github.com/tradr-project/tradr-doc
  - Axis camera driver (Axis network camera driver, written in Python)
  - SegMatch (Loop-closure detection algorithm)
  - LaserSlam (End-to-end system to laser-based graph SLAM user laser point clouds)
  - Tensorflow ROS (An easy way how to link Tensorflow C++ API to ROS programs)
  - Tracked vehicles in Gazebo (Added support for tracked vehicles to Gazebo)

Project Resources

Web: tradr-project.eu
Youtube channel: goo.gl/F1Q4MG
Datasets (video+laser+TF, indexed by bagbunker): goo.gl/fC6sbo

Learning for Active 3D Mapping with Solid-State Lidar