7630 – Autonomous Robotics

Introduction

Cédric Pradalier

Today

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Introduction





Dr. Cédric Pradalier

- Software Engineer (Ingénieur ENSIMAG)
- Master&PhD in Robotics (INP Grenoble).
- Post-Doc: Field Robotics, CSIRO, Brisbane, Australia
  - Industrial Robots
  - Underwater Robots
- Until Dec 2012: Deputy Director @ ASL, ETH Zürich, Switzerland
  - Space Robotics
  - Autonomous Boats
  - Inspection Robotics
  - ...
- Since 2013: Associate Professor at GeorgiaTech Lorraine, Metz.

Objectives of the class

Conceptual

- Overview of robotic applications.
- Overview of (some of) the tools to consider when designing a robotic system.
- Ideas for robotic software architecture.

Practical

- Implementation of main-stream algorithms, in simulation and on real systems.
- Experience with middleware-based architecture.
- Experience Sensory-Motor systems.

Syllabus

- 01/09: Introduction to robotics
- 01/16: Perception and probabilities
- 01/23: Introduction to Control & Reactive Behaviours
- 01/30: Kinematics
- 02/6: Bayesian Filtering & Markov Localisation
- 02/13: Kalman filter: Localisation and Mapping
- 02/20: More on mapping and data association
- 02/27:Introduction to Planning
- 03/6: GTL recess
- 03/13: Mid-term quizz & Visual Servoing (TBC)
- 03/20: Obstacle Avoidance
- 03/27: Exploration and Coverage
- 04/3,10,17,24: Final Project

Class Setup

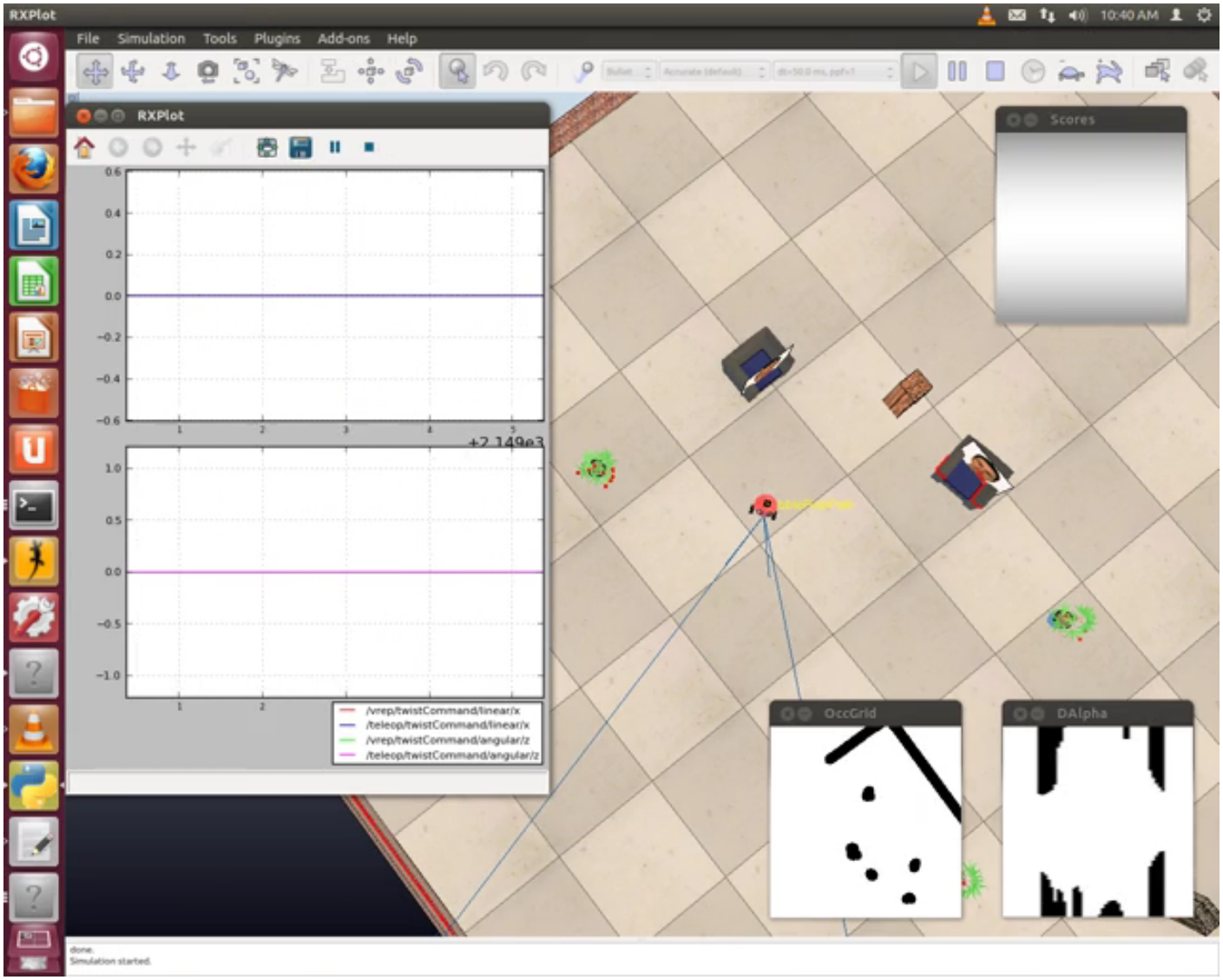
Tuesday

- Theoretical concepts
- Exercises
- Analysis of existing solutions (where appropriate)

Thursday

- Mini-project presentation (10-15 min with demo)
- Homework kick-off and assistance.

Homework environment: Simulation



Georgia Tech Lorraine

Homework environment: Turtlebots

Class Setup

Weekly Homework

- Simulation environment setup on GTL computer, Linux Ubuntu, ROS interface.
- Deployment on Turtlebots
- Mini-project in pair, with instructor support
- Designed to require 3 to 6 hours of work.
- Mandatory: simulation & turtlebot.

Deliverable

- Demo and Presentation: For **one** group.
- Code repository

Class Setup

Final Project

- 3 weeks, starting on April 3rd.
- Integration of component built during the semester
- Designed to require a lot of ~~work~~ fun.
- Final presentation instead of exam.

Deliverable

- Demo and Presentation
- Video
- Code repository

Syllabus: Evaluation

Course-work: 15%

- Mid-Term: Kinematics & Localisation – Date TBD after recess.

Projects: 85% credit

- Not returning a weekly project: -35%
- Not returning the final project: -50%
- Deliverable: Code, Report and demonstration, evtl. presentation.

Necessary conditions for an A

- Complete and working deliverable for **all** projects.
- Working deliverable on the turtlebot for **most** projects when appropriate.
- Don't crash the mid-term.

Syllabus: Office Hours

The class is designed for interactions

- Office 222, reliably 10:00 - 18:00, Mon-Fri.
- Robotic lab support otherwise.
- Don't hesitate to make an appointment.
- Don't hesitate to ask for support for technical questions.
- Don't come if you haven't even tried to solve your problem!
- Don't loose 2 hours on a technical issue.

Syllabus: Honor Code

- Students are required to follow the Georgia tech honor code which may be found at: `http://www.honor.gatech.edu/plugins/content/index.php?id=9`.
- Students are allowed to collaborate on out of class assignments but must include specific attribution to any help they received.
- Work turned in must be your own work not copied from anywhere else (including solution manuals) and you must state what type of assistance you received while completing the assignment.

In particular, white-board-level collaboration and exchange of ideas, as well as collaboration on solving technical programming challenges, are strongly encouraged. But **use these collaborations to build your own solution**.

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

First Autonomous Cars

CyCab

- PhD Thesis - INRIA
- SLAM, Navigation

ANU's SmartCar

- ANU/Nicta/CSIRO collaboration
- Parking assistance

1. Simultaneous localization and mapping using the geometric projection filter and correspondence graph matching, C Pradalier, S Sekhavat, Advanced Robotics, 2003
2. Autonomous Navigation of a Bi-steerable Car: Experimental Issues, J Hermosillo, C Pradalier, S Sekhavat, C Laugier, Machine Intelligence and Robotic Control Journal, 2004
3. Bayesian occupancy filtering for multitarget tracking: an automotive application, C Coué, C Pradalier, C Laugier, T Fraichard, P Bessière, The International Journal of Robotics Research, 2006

CSIRO ICT Centre: Autonomous Systems Lab

Hot Metal Carrier

Autonomous Tractor

- Low-level, navigation and control
- Mission execution
- Image-based load handling

- Autonomous Reversing

4. Vision-based operations of a large industrial vehicle: Autonomous hot metal carrier, C Pradalier, A Tews, J Roberts, Journal of Field Robotics, 2008
5. Robust trajectory tracking for a reversing tractor trailer, C Pradalier, K Usher, Journal of Field Robotics, 2008

More Field Robotics

Starbug (CSIRO)

CRAB (ETHZ-ASL)

- Low-level, navigation and control
- Mission execution
- Image-based Visual Servoing
- Synchronized wheel control
- Learning from experience

6. Robust vision-based underwater homing using self-similar landmarks, A Negre, C Pradalier, M Dunbabin, Journal of Field Robotics, 2008

7. Adaptive rover behavior based on online empirical evaluation: Rover-terrain interaction and near-to-far learning, A Krebs, C Pradalier, R Siegwart, Journal of Field Robotics, 2010

Design of Robotic Systems

Avalon (ETHZ-ASL)

Tartaruga (ETHZ-ASL)

- Designed and built from scratch
- Long-term autonomous navigation

- Designed and built from scratch
- Manouver optimization

8. Avalon, H Erckens, GA Busser, C Pradalier, R Siegwart, Robotics & Automation Magazine, 2010

Georgia Tech Lorraine

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

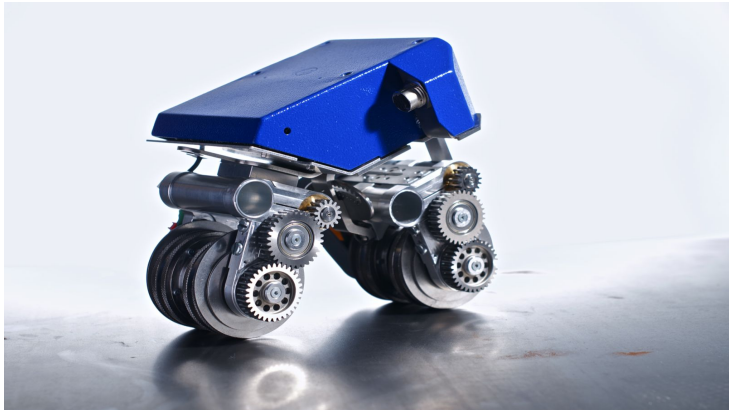
Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

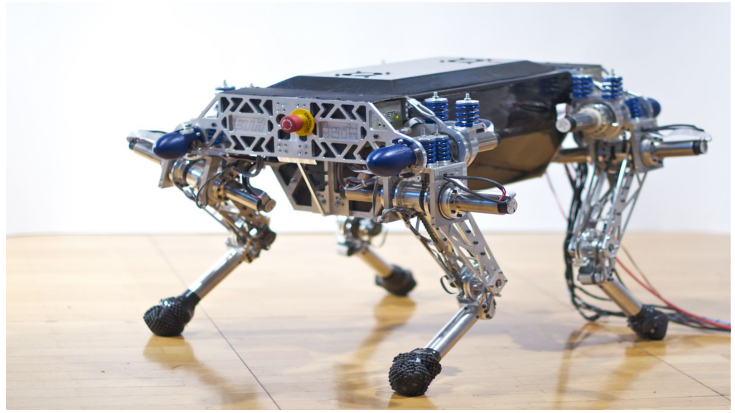
Summary

Unconventional mobility



Magnebike

- Pipe inspection
- 3D SLAM
- Path planning on manifolds



Starleth

- Walking with springs
- Ground perception
- Learning of footstep locations

Flying robots

sFly

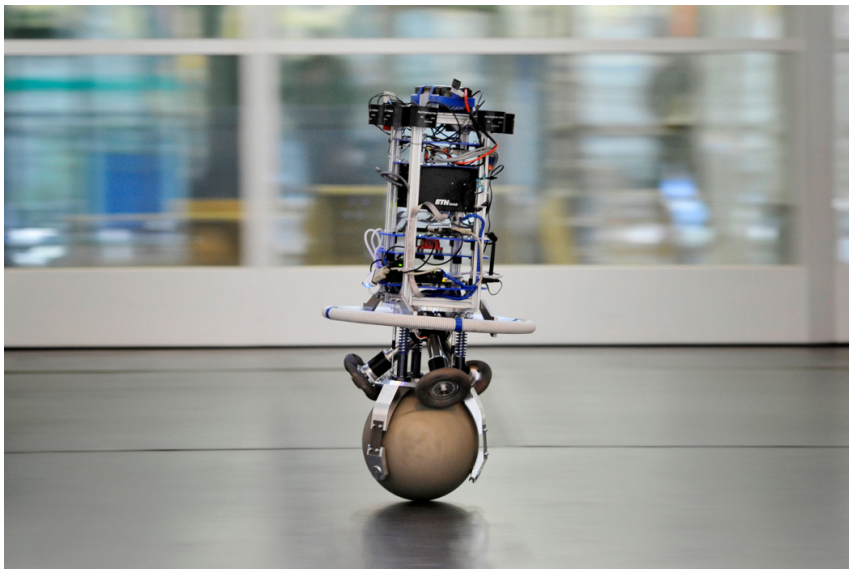
- Visual SLAM
- Sensor fusion

senseSoar

- Continuous solar flight
- Visual SLAM, sensor fusion

Education robots

Rezero



- Ballbot
- Autonomous control

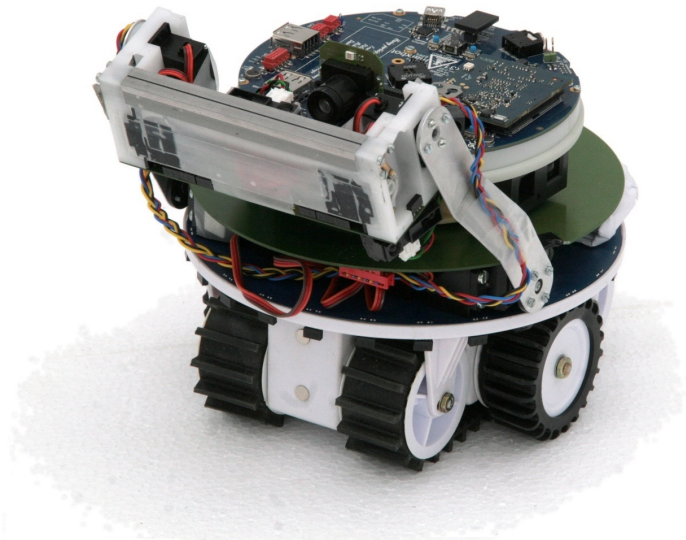
Hyraii

Video: `videos/hyraii.avi`

- Autonomous stabilisation
- Designed for high-wind and wave resistance

Miniature robots

Lonely builder



- Autonomous construction
- Block manipulation
- Task planning

Thymio II

- Open hardware for education
- $\approx$ 100 CHF
- `asebaros` bridge

Field robots

NIFTi

- 3D SLAM
- Navigation
- Semantic mapping

Limnobotics

- Lake monitoring
- Autonomous Navigation

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Designing Robotic Systems

Objective

- Understand the design process of a robotic software system.
- Get an overview of typical components
- Introduce middlewares and ROS.

Example of Application

Hot Metal Carrier Project, CSIRO ICT Center
Brisbane Australia

Example of Application

Information sources

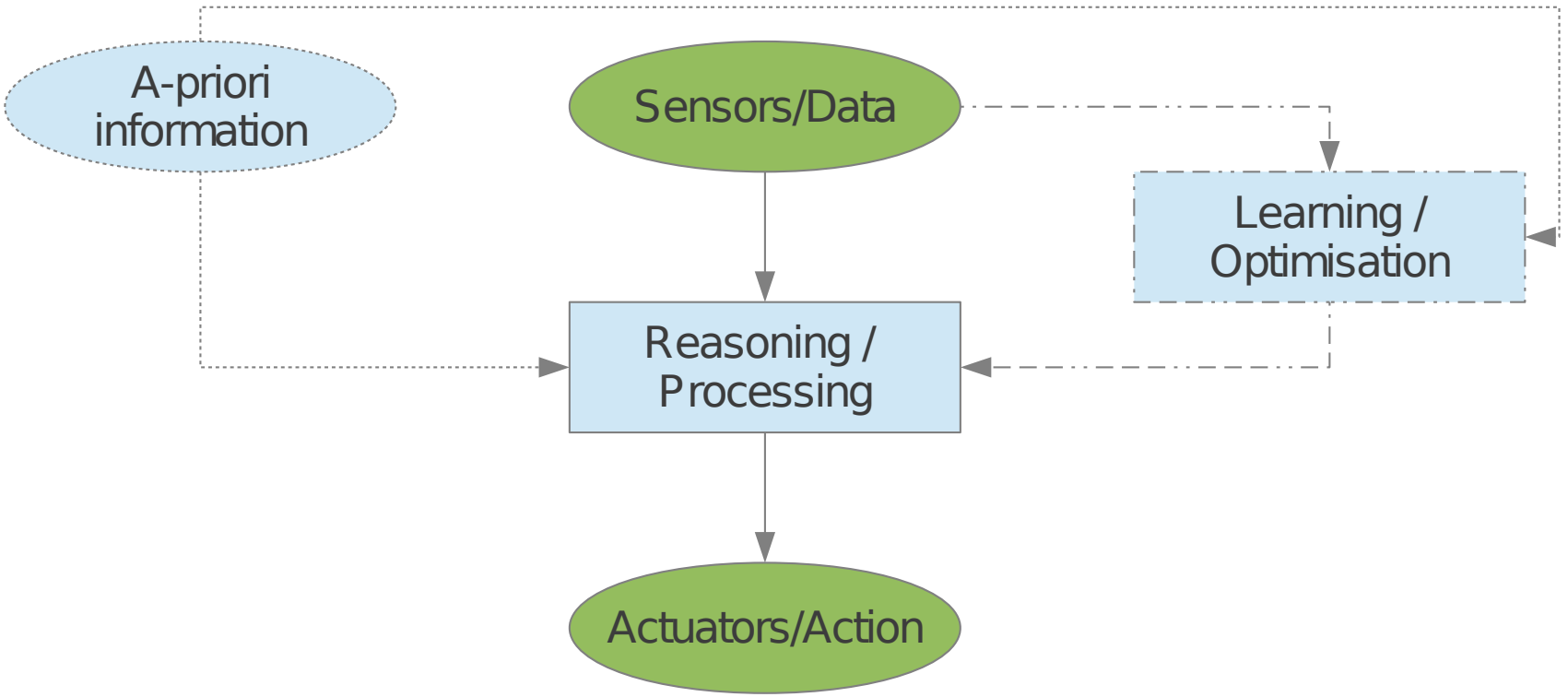
- Pan-tilt-zoom camera
- Laser scanner
- On-board sensors
- A-priori information:
  - maps,
  - vehicle model,
  - tuning parameters
- ...

Example of Application

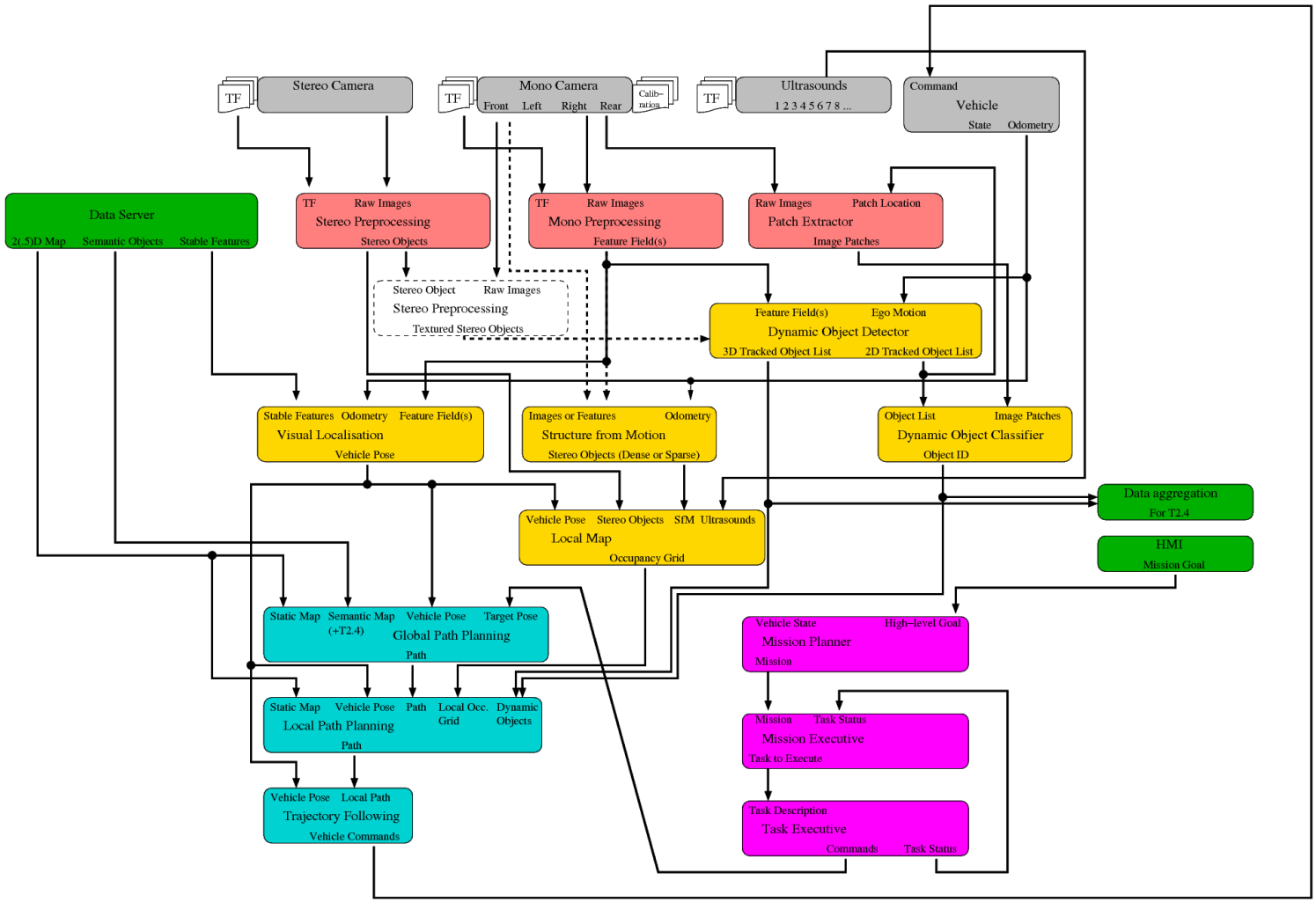
Processing

- Detection
- Estimation
- Tracking
- Control
- Decision
- Path Following
- Visual Servoing
- Obstacle Avoidance
- Localisation
- ...

Generic Design Pattern

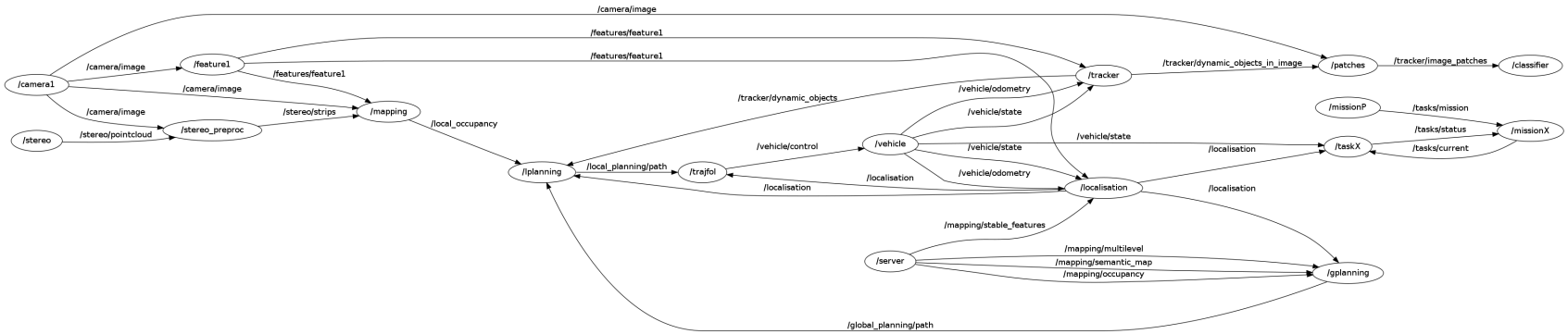


Applied to a Real System

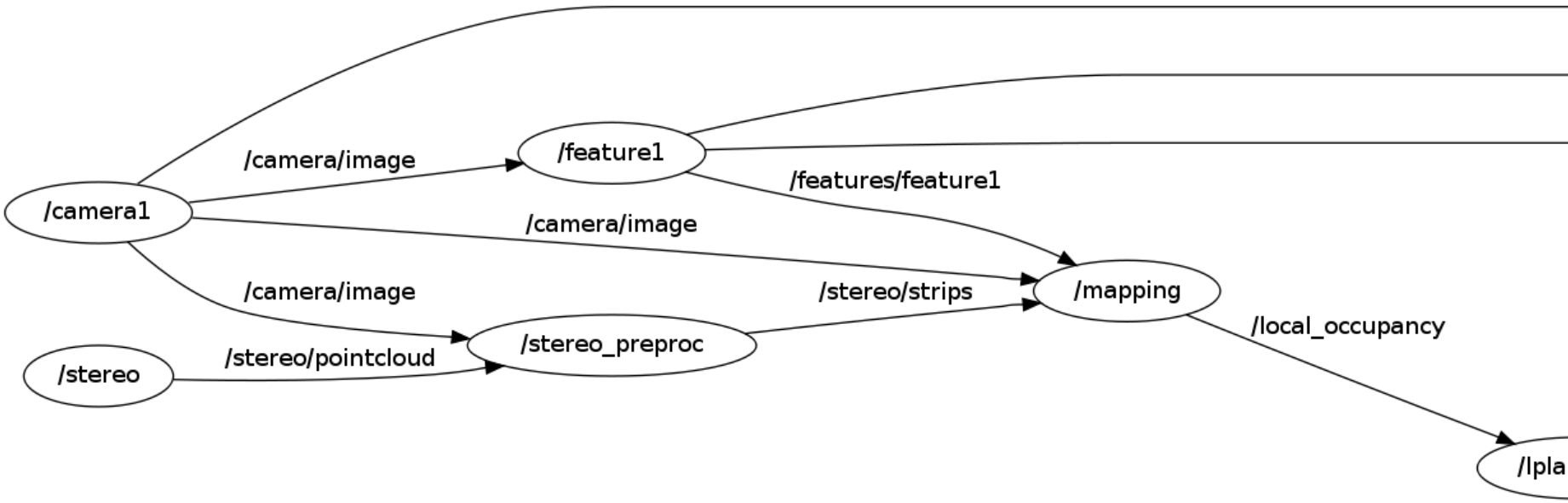


V-Charge: Autonomous cars driving in urban environment – Online architecture

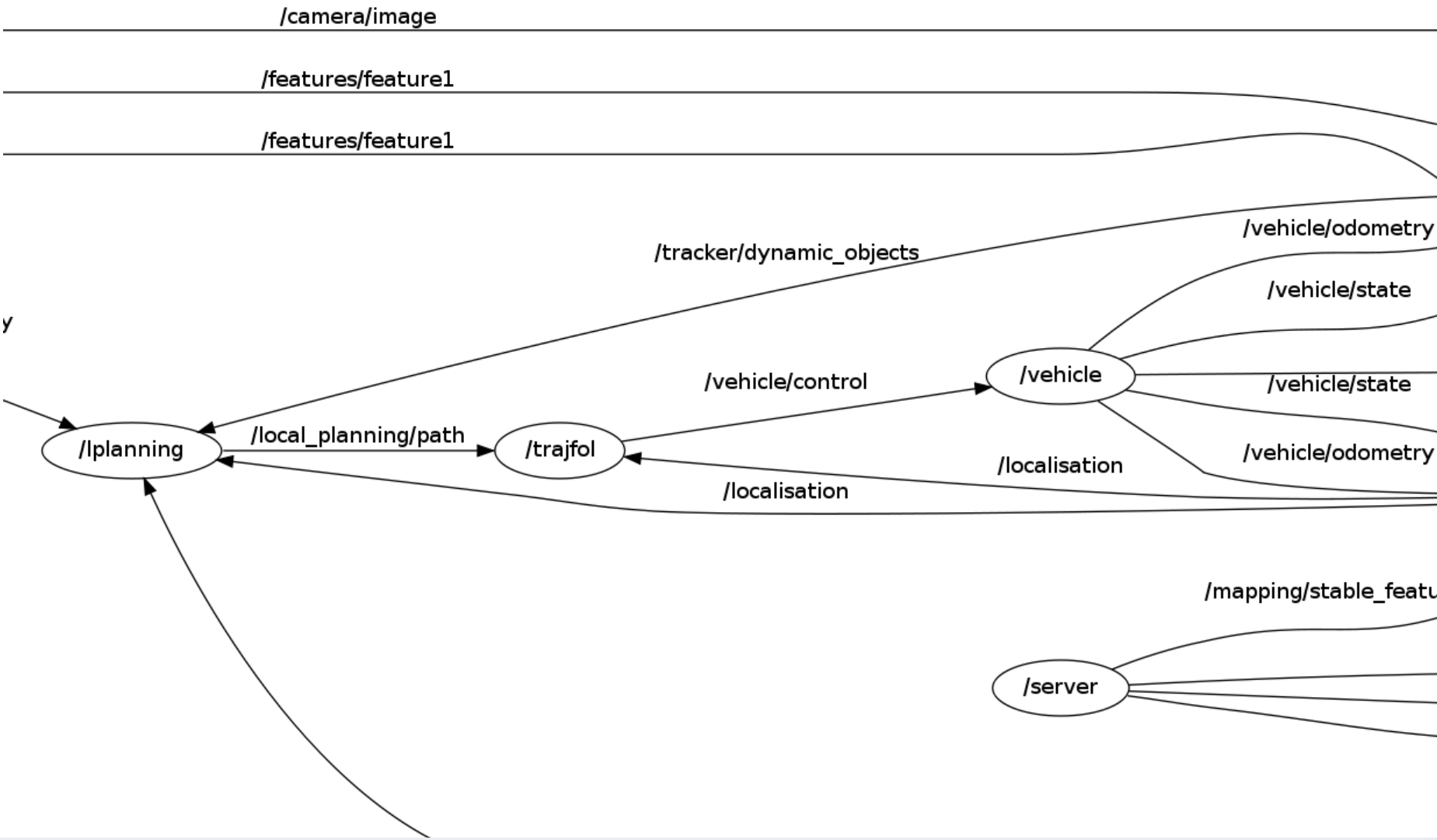
Data Flow and Interactions



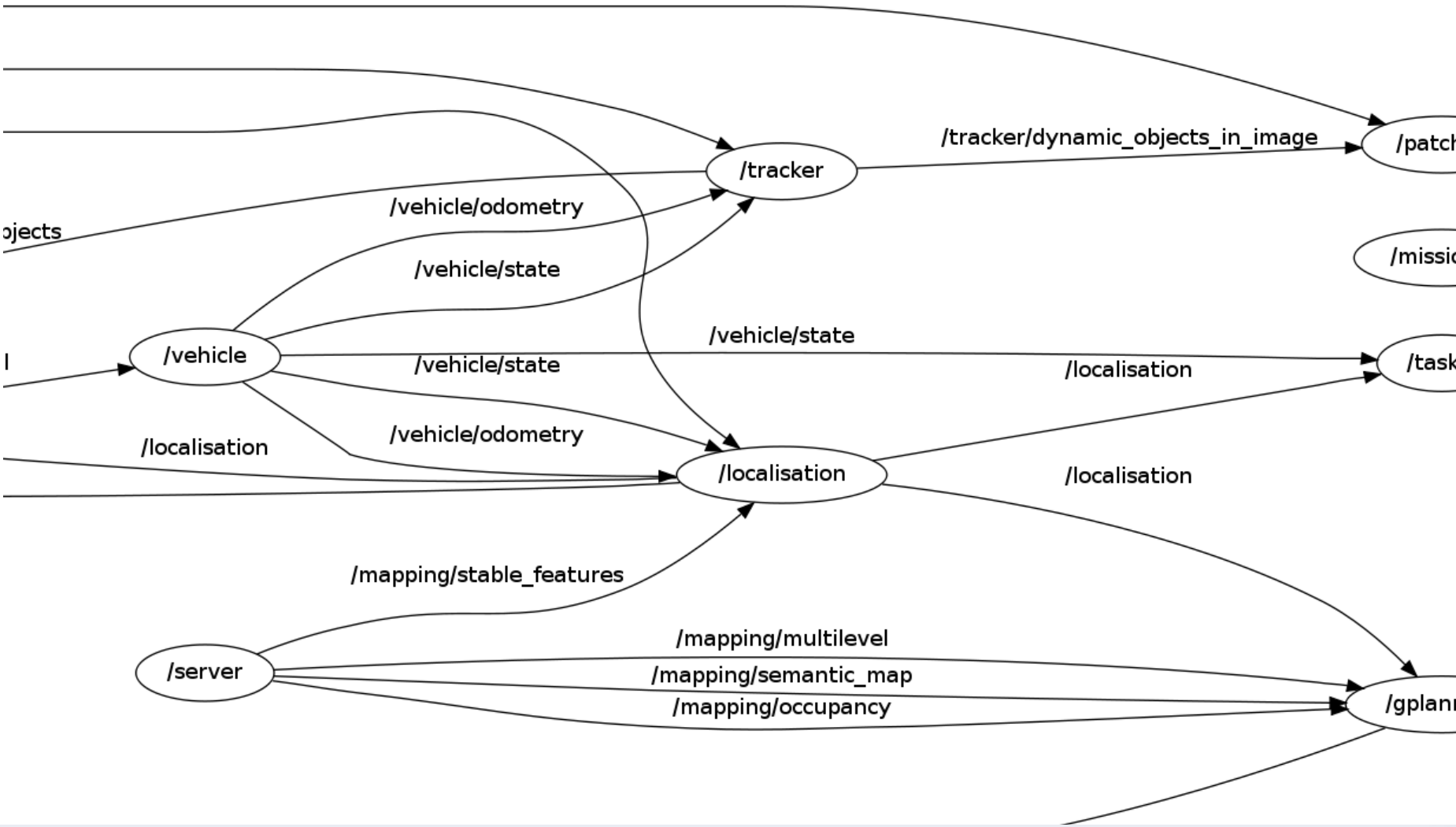
Data Flow and Interactions



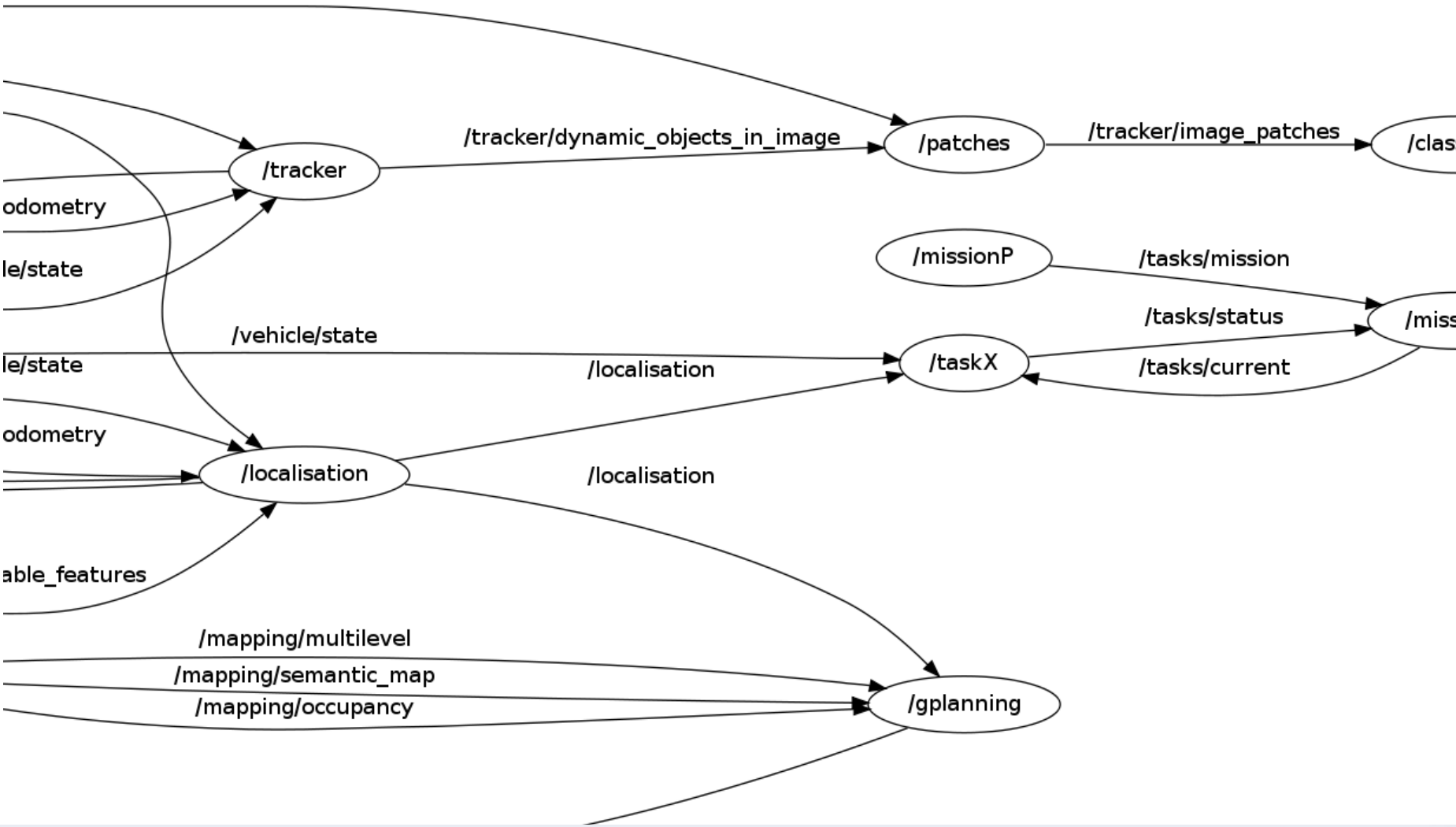
Data Flow and Interactions



Data Flow and Interactions



Data Flow and Interactions



Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Robotic Middlewares: Managing the Complexity

Examples from the Research Community

- DDX: CSIRO, Australia
- Player/Stage:
- Yarp:
- Carmen IPC:
- Genome: LAAS, France
- ROS: Open-Source, Willow-Garage, US
- ...

Similar frameworks from industry

- RTI DDS: Spin-off from Standford
- ADTF: German car industry
- Microsoft Robotic Studio

Common objectives

Common design pattern

- Independent modules (programs) with input/output
- Communication mechanism between module
- Possibility to distribute the modules on multiple computers
- More or less standardized program design patterns

Different implementation

- Communication transports: TCP, UDP, shared memory, ...
- Communication principles: data stream, blackboard, client/server, ...
- Real-time or not
- Cross-platform or not
- Multi-language support

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

ROS

- Stands for Robot Operating System, but runs on top of existing OS like Linux
- Originates from SAIL/Willow Garage
- Combines several features into a consistent project:
  - a software distribution with a dependency mechanism
  - an integrated build system
  - a communication middleware
  - helper tools: visualization, record/replay, etc.
- Provides client libraries for C++, Python, lisp; experimental ones for Java, Lua

ROS, the distribution platform

Software packages, organized in stacks, that cover many areas of robotics:

- Sensor interfaces:
  - Camera, laser, IMU, GPS, ...
  - Camera calibration, video streaming
- Visualization: 2D real-time plotting, 3D rendering, video display
- Data logging and replay
- 3D simulation
- Perception and Computer Vision (OpenCV)
- State estimation and Mapping
- Navigation and Planning
- Object detection
- Grasping
- Etc.

ROS, the robotics middleware

- Components (nodes) that may run on different computers
- Data exchanged through topics (event-based) and services (RPC), wrapper generated from descriptions
- TCP and UDP transport layers
- Standardized messages (images, laser scans, point clouds, IMU, GPS, joystick)
- Parameter server
- Visualization of data (rviz)
- Record and replay (bags)

What ROS is not:

An Operating System

- Needs a real operating system on a PC
- Linux Ubuntu preferred but not limited to.

A hard real-time environment

- ROS relies heavily on threads
- Does not offer scheduling or real-time guarantees

An OS for micro-controllers

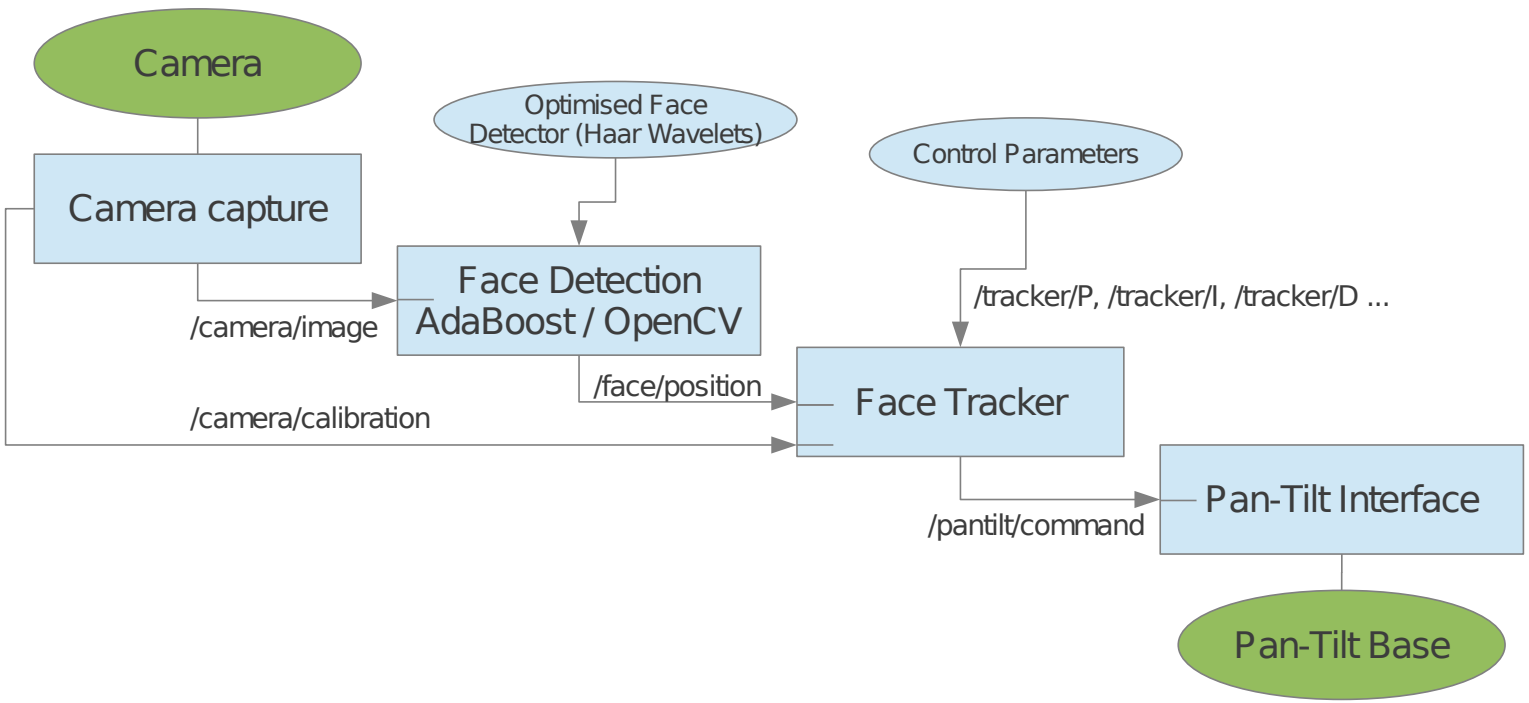
- Requires a significant amount of RAM, hard-drive and CPU

An Example

Face-tracking with a camera on a pan-tilt unit

An Example

Face-tracking with a camera on a pan-tilt unit



ROS Concepts

Node

- A program with some input and output
- Example: camera driver, face tracker, ...

Topic

- A (continuous) stream of data published by a node, or subscribed to by a node
- Example: /camera/image, /pantilt/command

ROS Concepts

Service

- An interface offered by a node in a request-answer form
- Example: /pantilt/reset, /camera/set_resolution

Parameter

- A constant value accessible at any time by any node
- Example: controller gains, /tracker/P ...

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

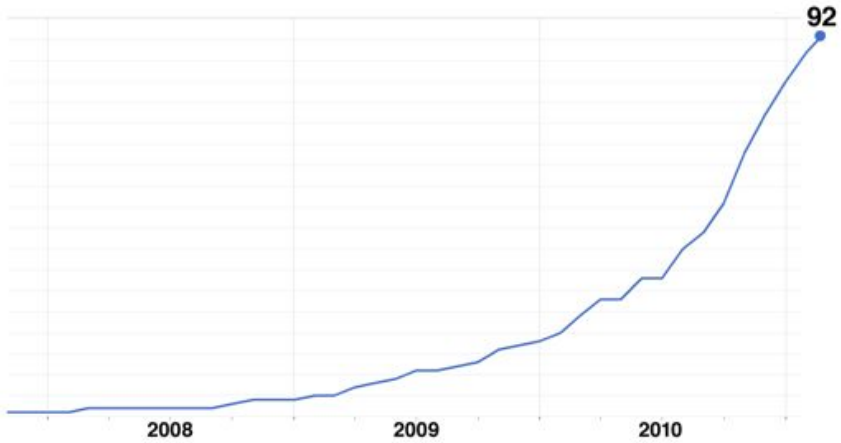
Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

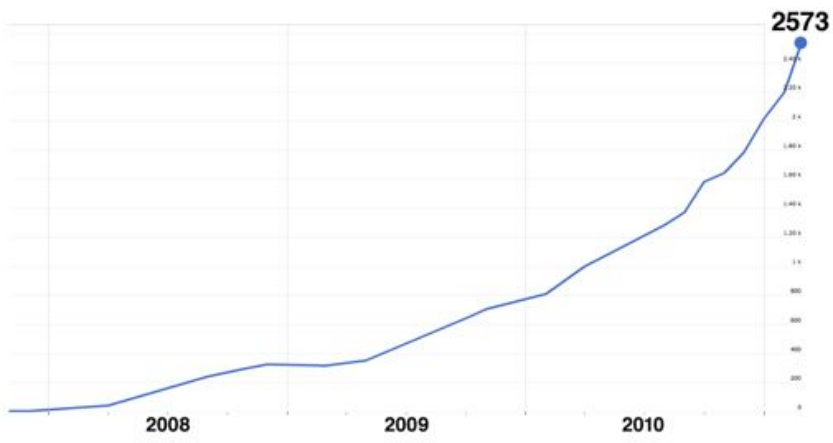
Case Study

Summary

ROS, status and future



Repositories



Packages

- Work force: currently 60 full-time positions at Willow Garage + many contributors from academia.
- Being a meeting point, ROS has lead to dramatic improvements in the standardization of robotics software.
- Challenges: maintenance of core, integration of third-party contributions, further standardization, application to industry.

Why do we need open-source in robotics?

- A functioning robot requires a lot of software:
  - A large part is similar from one robot to another, yet many details differ.
  - Closed-source software is expensive and certainly would not fit the needs.
- PhD students come and go, high-level knowledge is archived in papers, but detailed knowledge evaporates:
  - Released open-source software is of much higher quality than personal research software.
  - Open-source contributes to communication and collaboration within the community.
- Quality open-source code from research can easily contribute to applications:
  - This drives robotics forward.
  - This can motivate industry to invest money into research.

Why is robotics different than computer science?

- Very fragmented landscape:
  - various platforms, sensors and actuators
  - various mechatronic specificities
  - various environmental features
  - various research questions
- Most development is not done in the target environment:
  - need for monitoring, simulation and replay tools
  - test cost is high
- Information processing is complex:
  - data are noisy, uncertainty might be non-Gaussian
  - real-time constraints
- Common limitations of target computer:
  - computational power
  - network connectivity
  - access to keyboard/screen
  - processor might not be x86 or have latest features

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Use case: the NIFTi Project

NIFTi:

- Urban Search And Rescue.
- Firemen and rescue team as end-users.
- Yearly evaluation of the system.
- Scenarios of increasing complexity.

Natural Human-Robot Cooperation in Dynamic Environments

NIFTi: Experimental Scenario

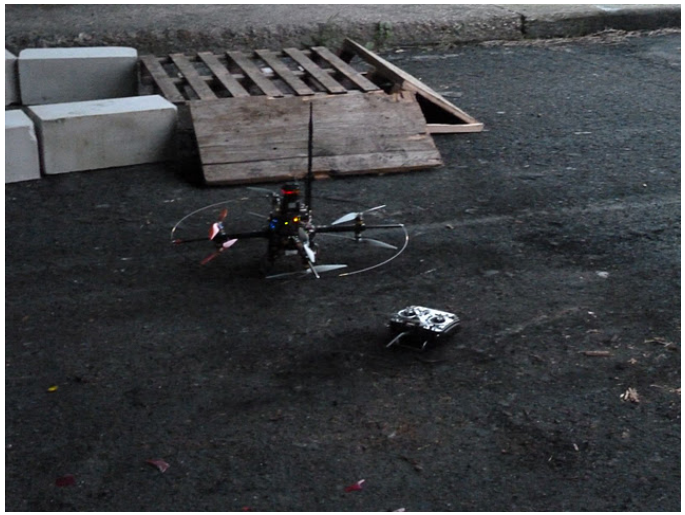
Tunnel at firemen school near Rome:

- car accident in a tunnel,
- truck unloaded some barrels and pallets,
- send the robot(s) to look for victims.

NIFTi: Role of the system

NIFTi system:

- UGVs and UAV(s),
- firemen operators,
- mission commander,
- UAV/UGV operators.



NIFTi: Software Components

ROS

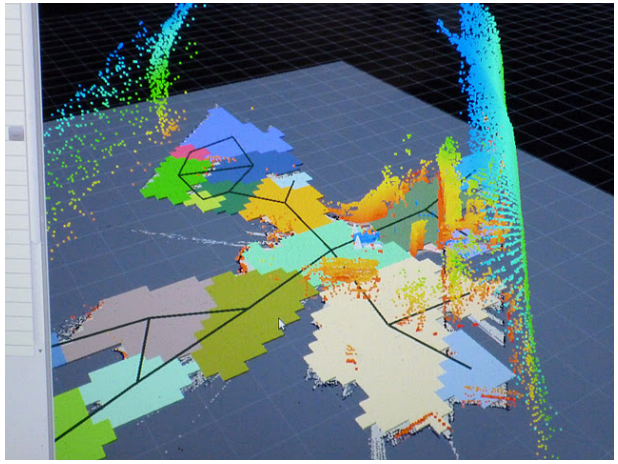
- mapping (hybrid: 2D, 3D and topological),
- vision (car and victim detectors),
- path-planning and execution (morphological adaptation).

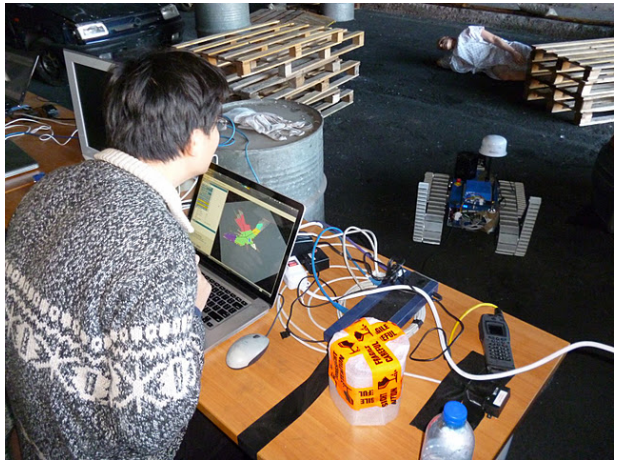
CAST

- GUI for the operators (situation awareness),
- ontological inference (to get the meaning),
- high-level planner (logical planning),

NIFTi: Role of the Autonomous Systems Lab

- 3D mapping with point clouds,
- topological segmentation of navigable space,
- making sure the robot works:
  - robot ROS driver,
  - networking setup,
  - 2D map and navigation,
  - handling the robot.

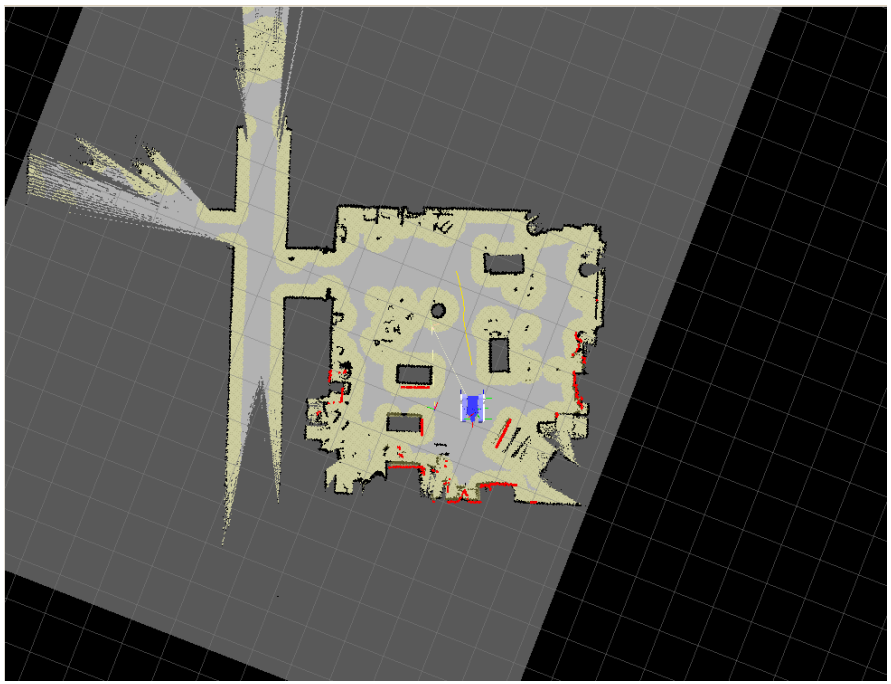


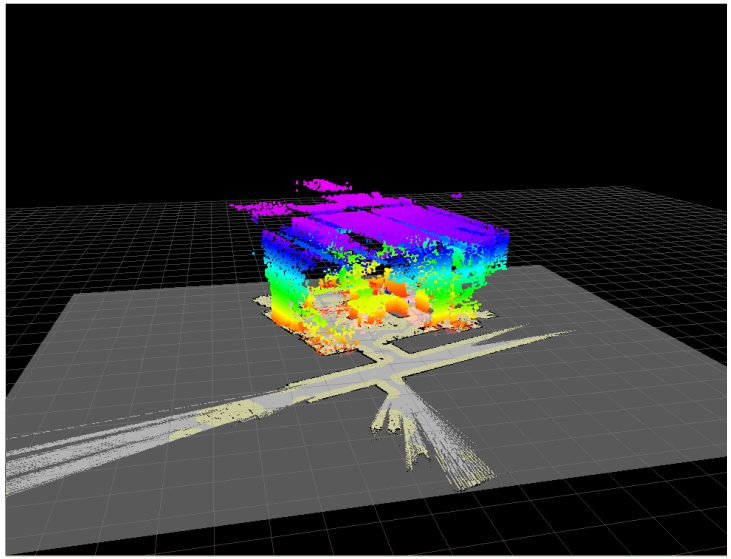


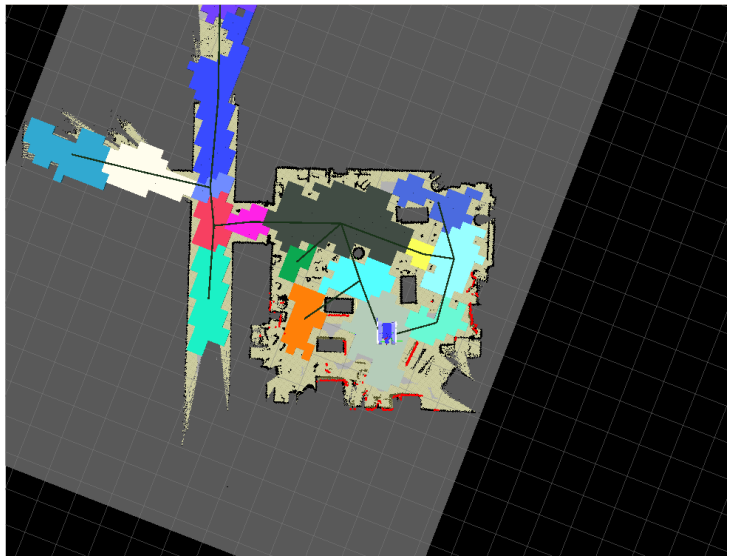
NIFTi: Mapping

Mapping layout:

- 2D mapping (karto_slam in ROS),
- 3D point cloud map,
- topological segmentation.





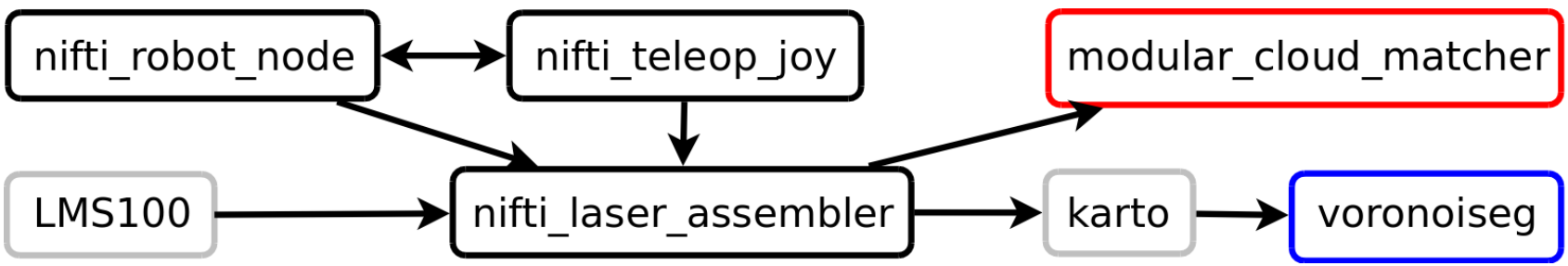


NIFTi: 3D environment mapping

NIFTi: Software architecture

Concretely:

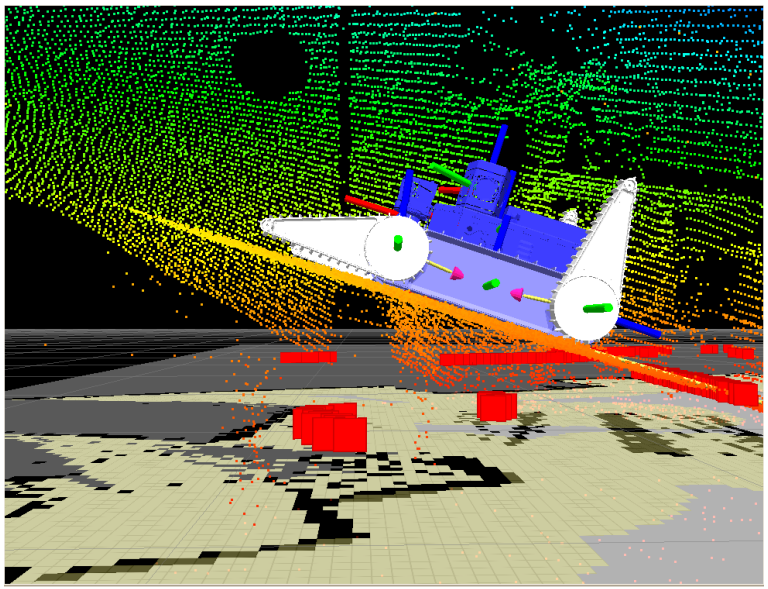
- `nifti_robot_node`: ROS driver for the robot,
- `nifti_teleop_joy`: joystick teleoperation,
- `nifti_laser_assembler`: filtering, dispatching and assembling laser scans,
- `modular_cloud_matcher`: ICP registration of point clouds,
- `voronoiseg`: topological segmentation.

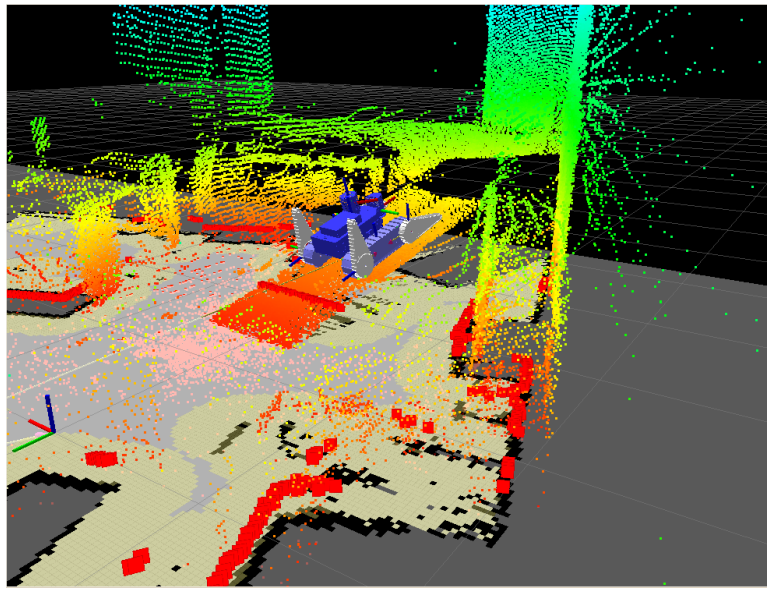


NIFTi: Going 3D

3D navigation:

- IMU integration.
- Path planning and execution using tensor voting.
- Morphological adaptation.





NIFTi: ROS Summary

ROS: out-of-the-box first layer of functionalities

- 2D Navigation and planning.
- Sensor interfaces (laser, cameras, IMU).
- 3D Point Clouds algorithms
- Logging, replay, monitoring, visualization, ...

ROS: enabler for complex system integration

- Decouples hardware interfaces from algorithms.
- Common interfaces shared by all partners across Europe
- Logging, data sharing, monitoring, visualization, ...

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Use case: the Limnobotics Project

Limnobotics:

- Regular lake monitoring.
- Algal bloom detection.
- Repeatable study of water parameters between 0 m and 20 m.
- Autonomous operation.

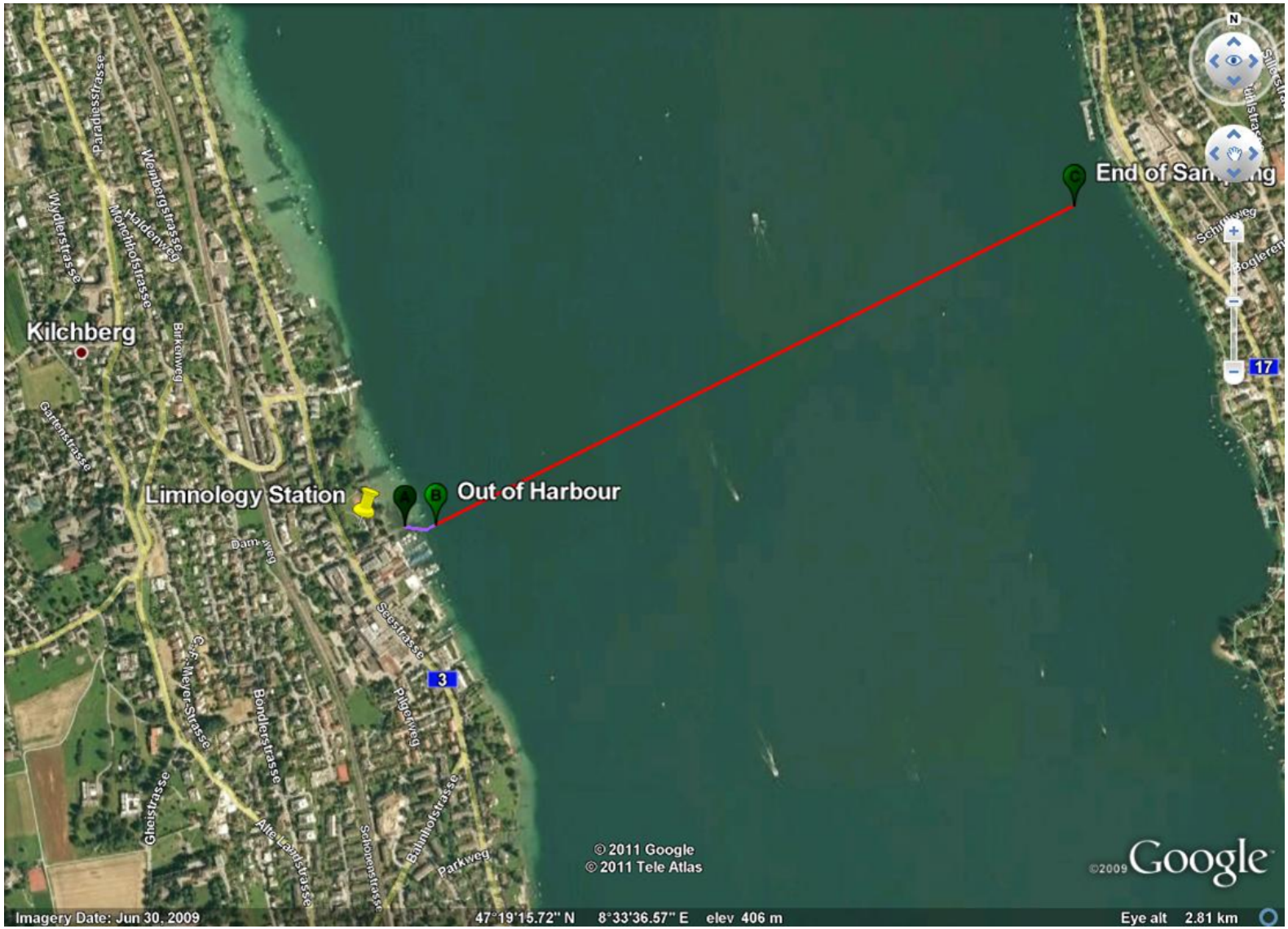
Funded by the Swiss National Fond (SNF).

Limnobotics: Lizhbeth sailing

Limnobotics: Lizhbeth details

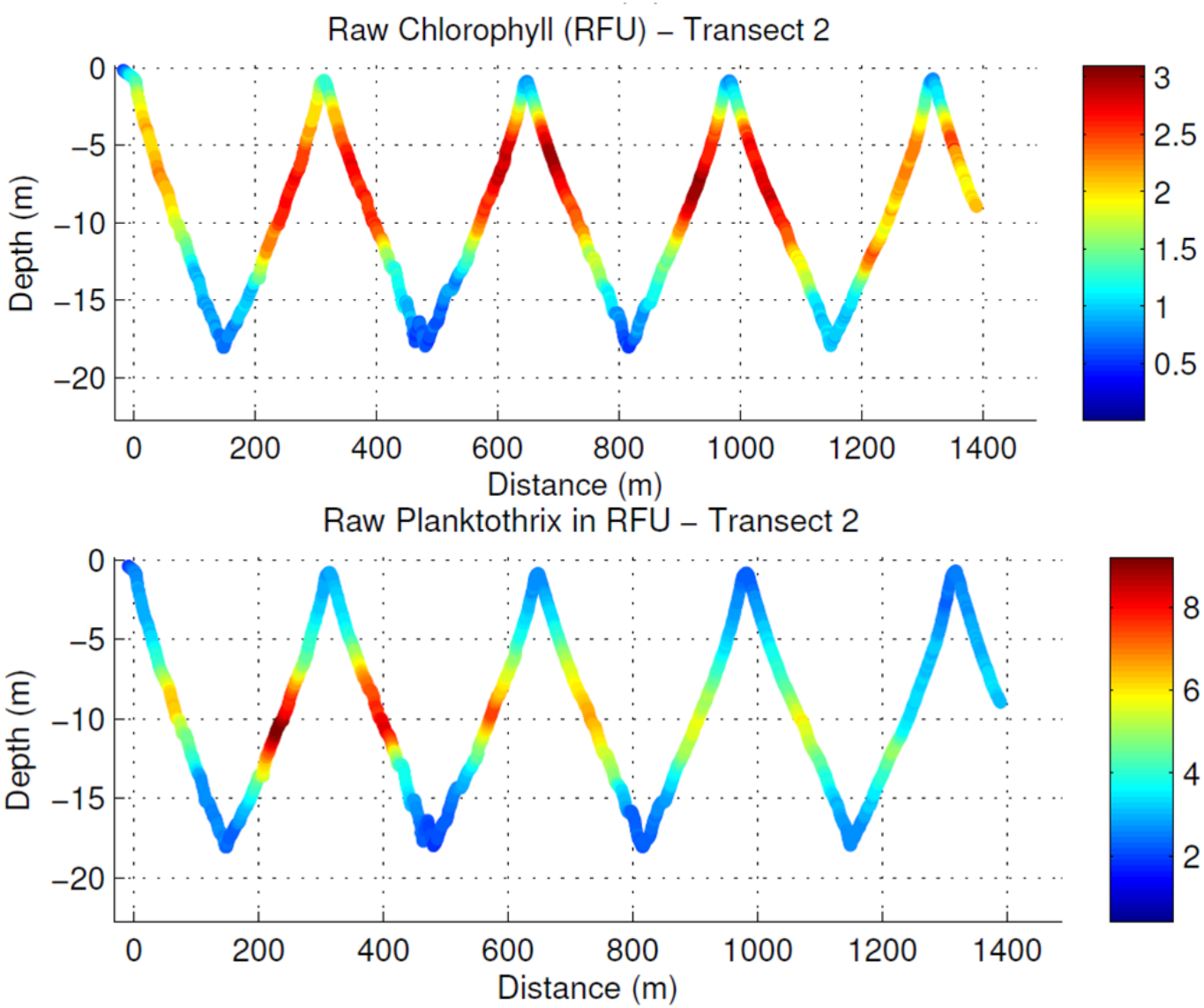
- Custom-made hull design
- Length: 2.5 m
- Width: 1.6 m
- Weight: 120 kg
- Electric motors and marine-grade batteries
- Winch with up to 120 m of cable (25 m used regularly).
- Biology sensor from YSI

Limnobotics: Typical navigation task: $2 \times 1.5\,\text{km}$

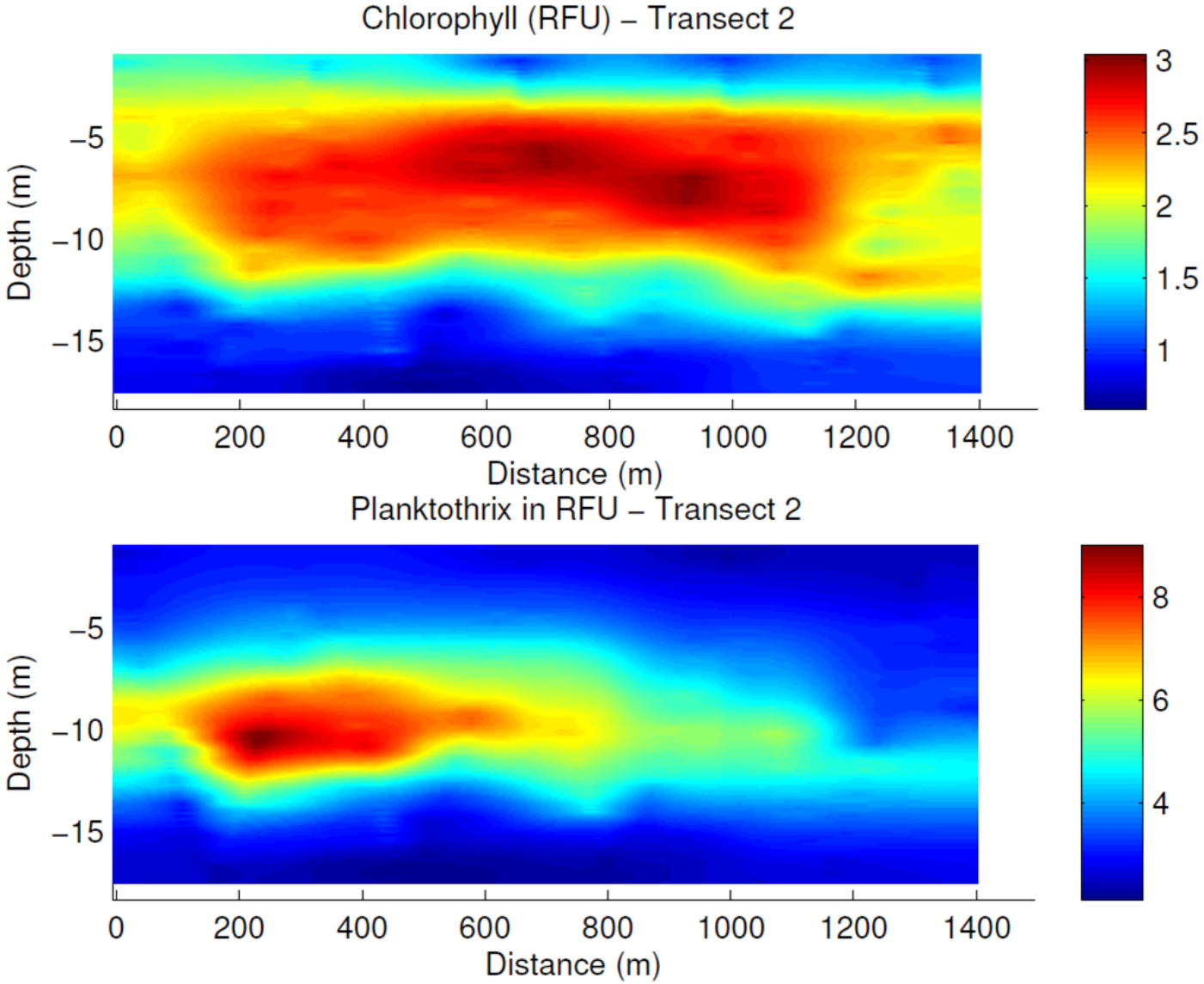


Georgia Tech Lorraine

Limnobotics: Measurements

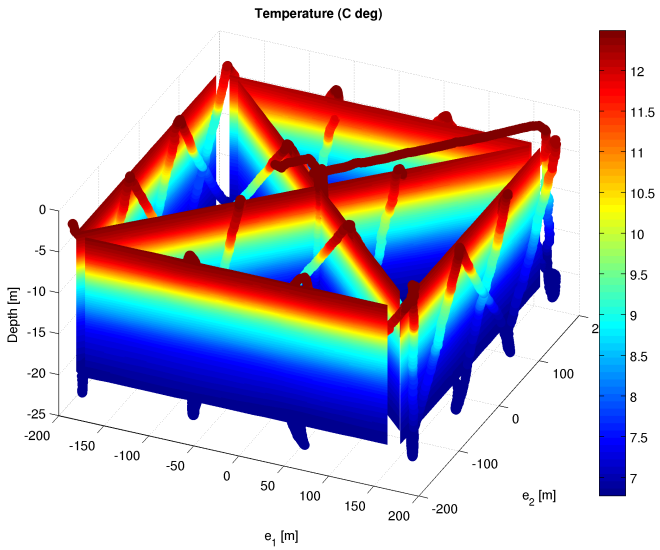


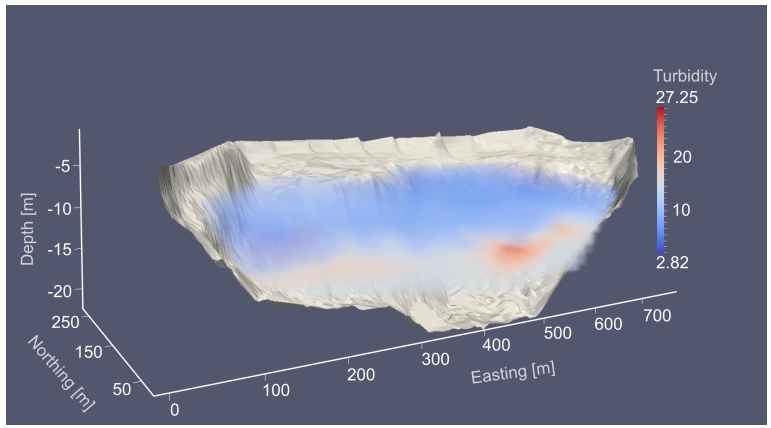
Limnobotics: Interpolated Measurements



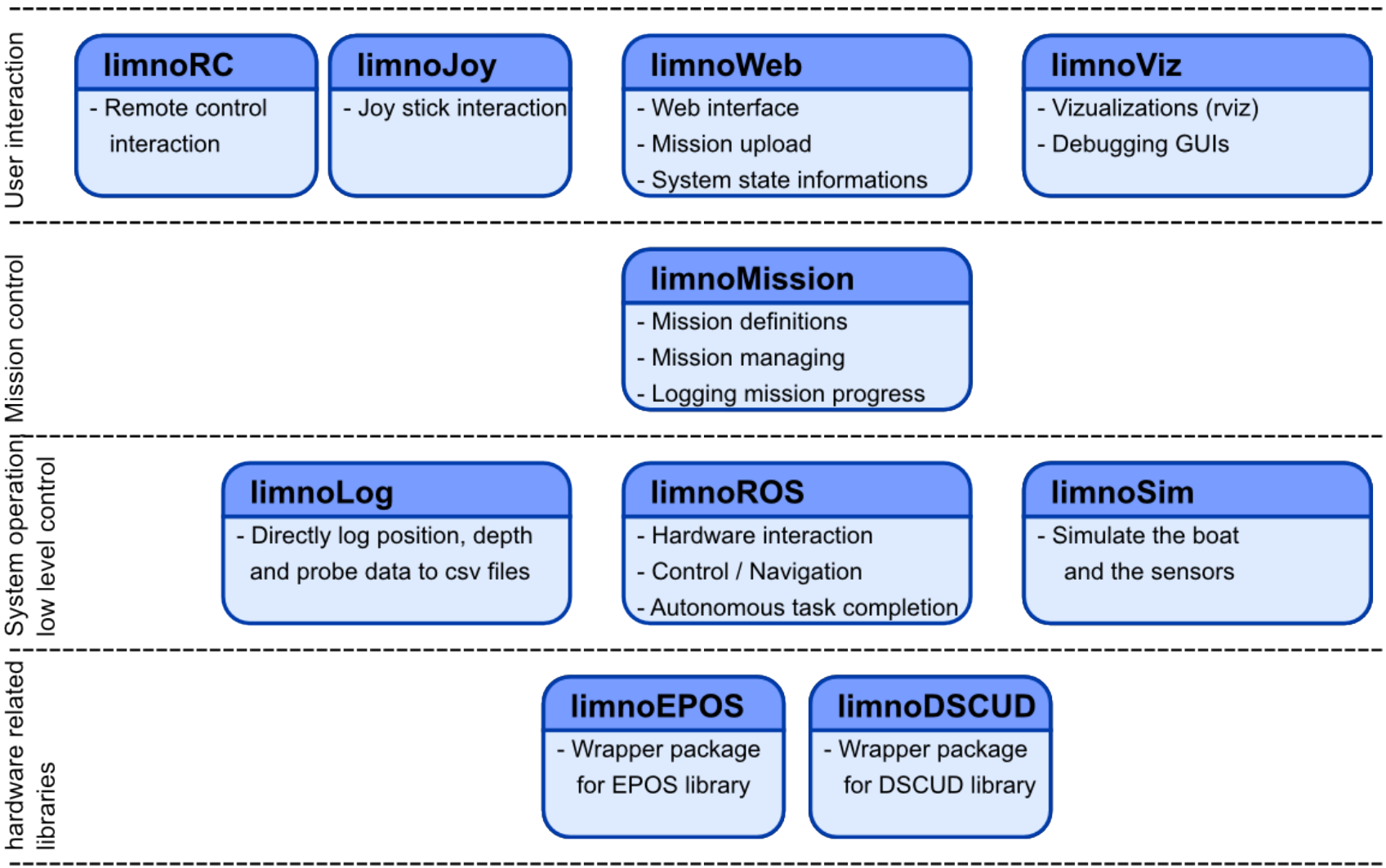
Georgia Tech Lorraine

Measuring a 3D Volume





Limnobotics: Software architecture



Limnobotics Summary

- Development of a robotic tool for science data acquisition
- Very mature robotic tool
  - Just draw the mission on Google earth
  - Load it inside the mission python script
  - Let it run
- Leveraging low-level tools and message definitions from ROS
  - Relatively few off-the-shelf solution for boat navigation
  - Few ROS packages for our sensors
- Converting to ROS made a huge increase in reliability and extendability.

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

Application Design

Draw the flow of data

- Identify nodes
- Identify topics

Define data-types

- Reuse standard messages whenever possible
- Define your own .msg files

Select coding language

- C++ for computation intensive process
- Python for glue or simple processes
- Also possible: java, octave, lisp, javascript indirectly

Debug, test and evaluation

Unit testing

- Test and validate every node independently
- Avoid debugging the complete system

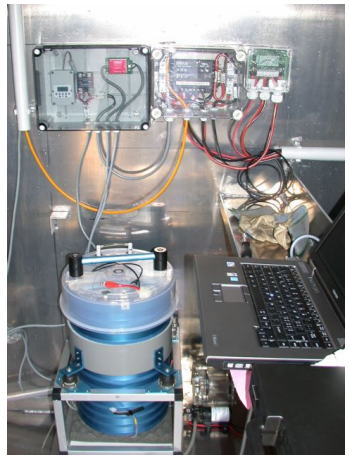
Debugging

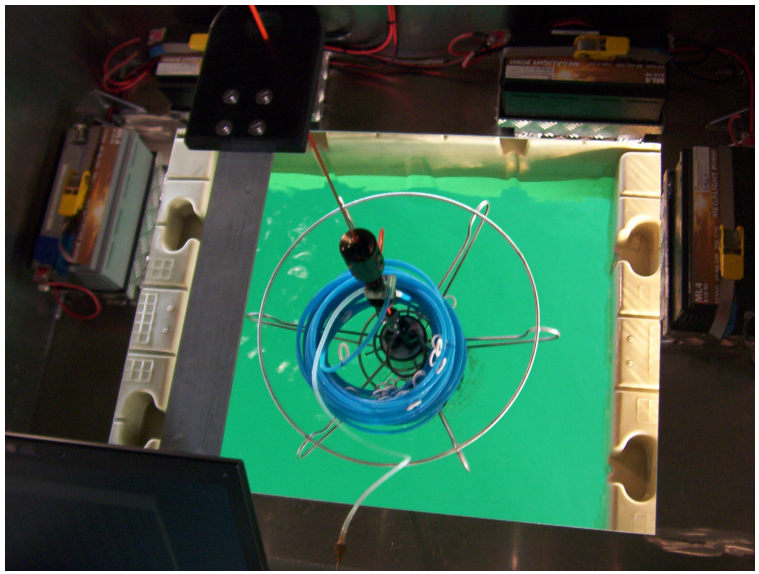
- Use fake input generator and check output
- Use pre-recorded bag-files
- Record the output and plot in matlab, or even in real-time
- Test on real data as soon as possible, but test on real system only at the end.

Case Study: Water Quality Monitoring

Automated Sampling for Water Quality Monitoring

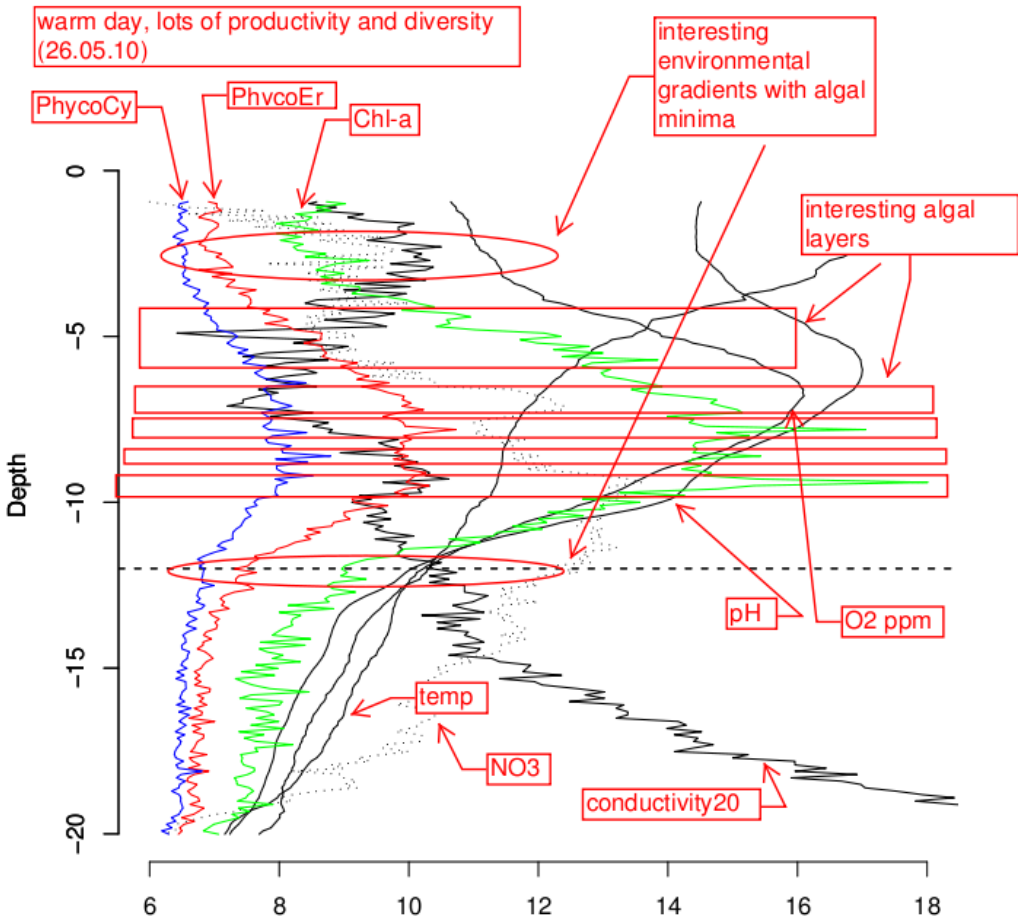
- Automated data acquisition
- Data processing
- Automated sample acquisition





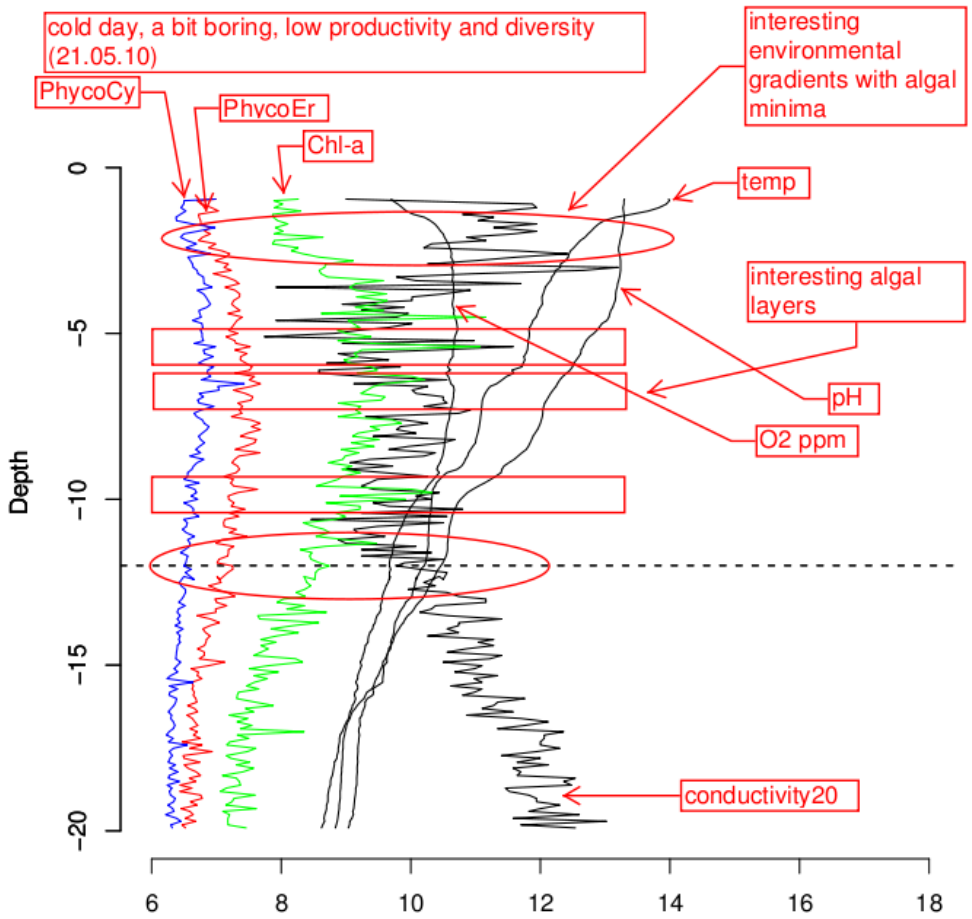
Picture courtesy of F. Pomati, EAWAG

Measurement Examples



Picture courtesy of F. Pomati, EAWAG

Measurement Examples



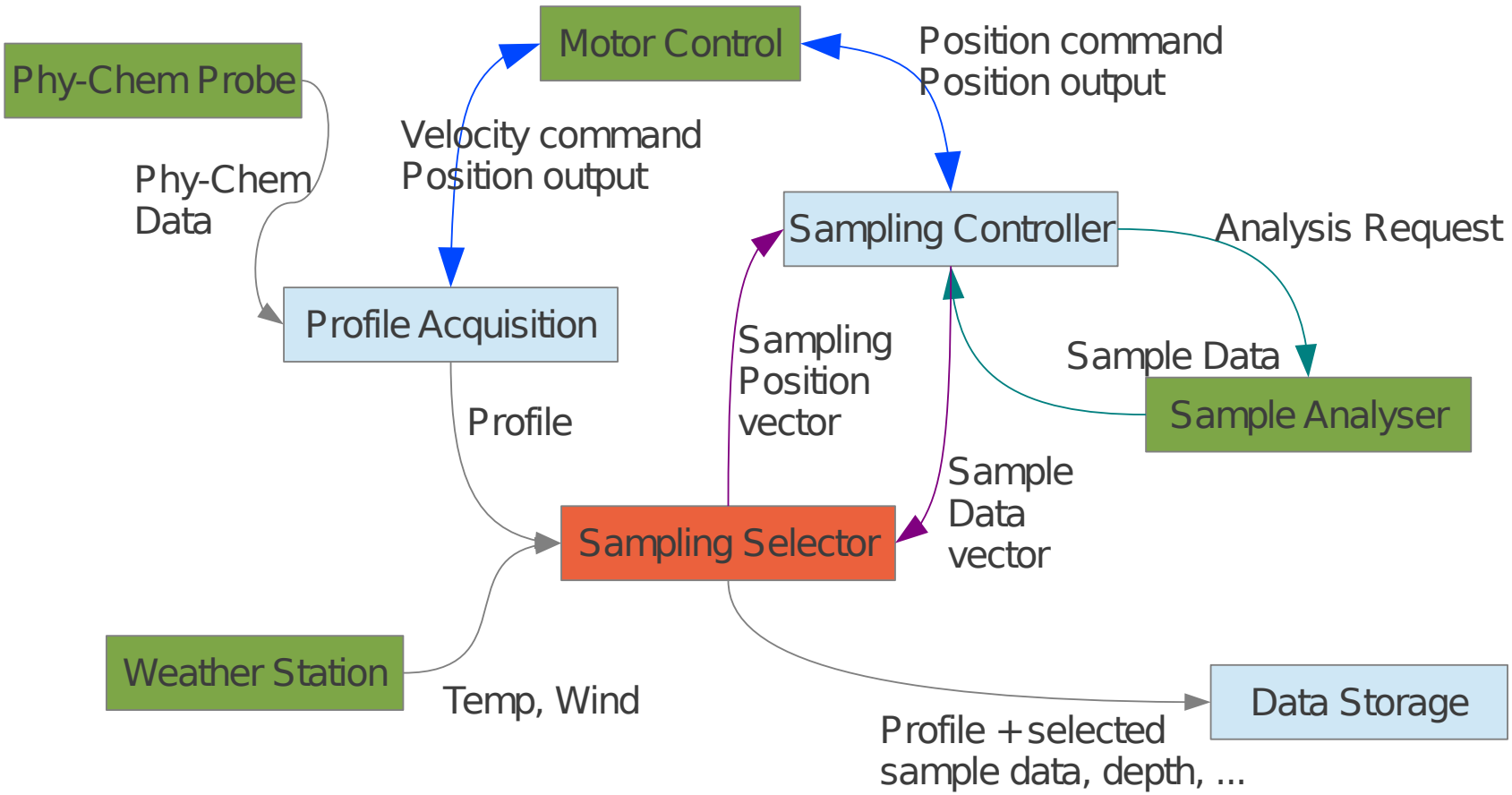
Picture courtesy of F. Pomati, EAWAG

System Components

Modules (or nodes)

- Motor controller
- Sampling controller
- Weather Station
- Sample Analyser
- Physico-chemical probe
- Sampling selector
- Profile Acquisition
- Data Storage

Data Flow


Motor Control
Phy-Chem Probe
Position command
Position output
Velocity command
Position output
Phy-Chem
Data
Sampling Controller
Analysis Request
Profile Acquisition
Sampling
Position
vector
Sample Data
Sample Analyser
Profile
Sample
Data
vector
Sampling Selector
Weather Station
Temp, Wind
Data Storage
Profile + selected
sample data, depth, ...

Component details

Motor controller

- Input: Position, Velocity, Idle → Topic.
- Output: Position, Velocity, Status → Topic.

Physico-chemical sensor

- Input: none
- Output: physico-chemical channels → Topic.

Weather station

- Input: none
- Output: temperature, wind, etc. → Topic.

Component details

Sample Analyser: service on request

- Returns: cell count, cell id.

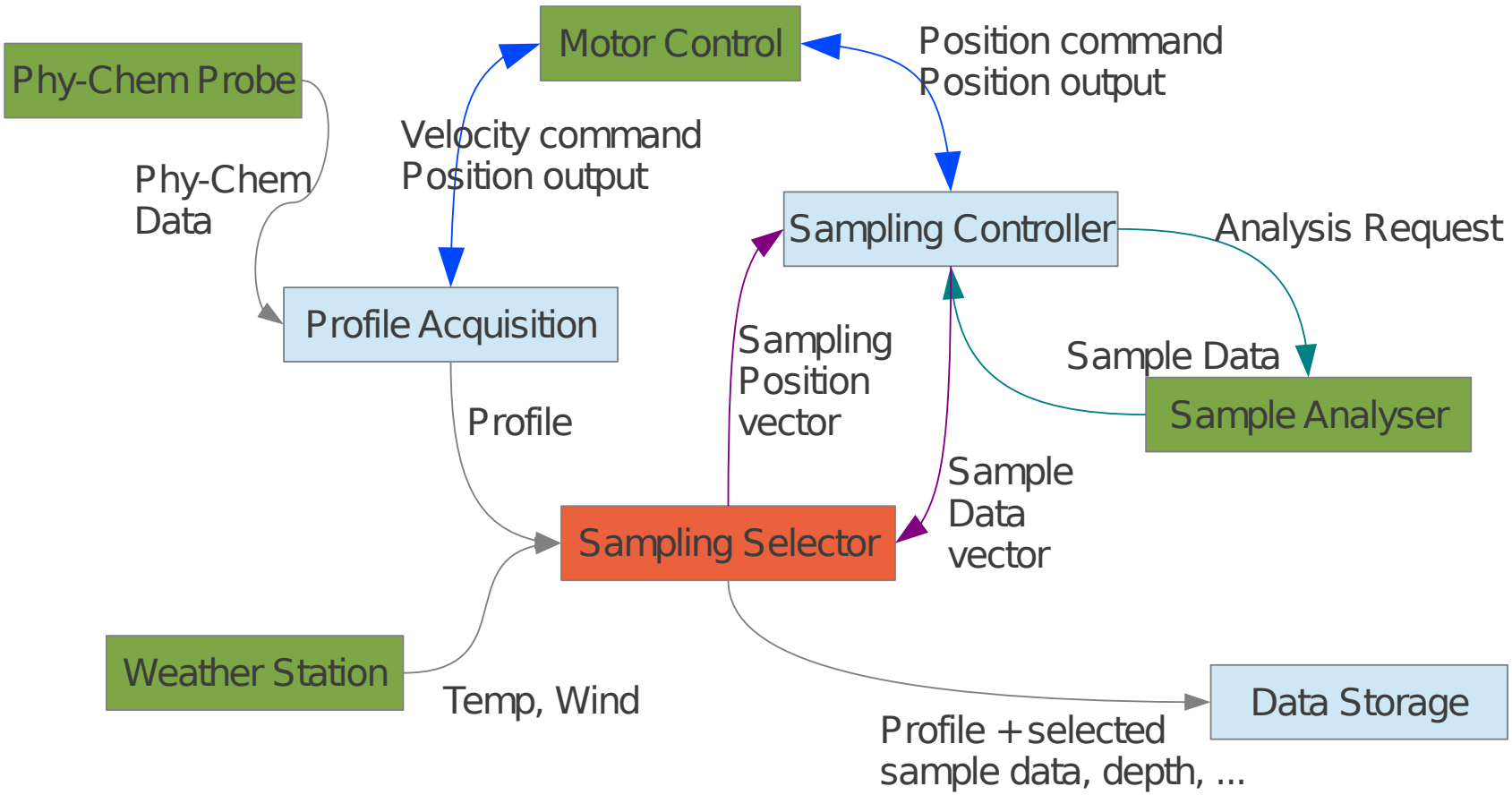
Sampling Controller: service (or action)

- Input: vector of sampling position
- Output: trigger for sample analyser

Sampling Selector

- Input: physico-chemical profile, weather data, ...
- Output 1: selection of sampling points
- Output 2: vector of sampling results

Data Flow


Motor Control
Position command
Position output
Phy-Chem Probe
Velocity command
Position output
Phy-Chem
Data
Sampling Controller
Analysis Request
Profile Acquisition
Sampling
Position
vector
Sample Data
Sample Analyser
Profile
Sample
Data
vector
Sampling Selector
Weather Station
Temp, Wind
Data Storage
Profile + selected
sample data, depth, ...

Example of Message Types (i.e. Topic Types)

PhyChemData

```
Header header
float32 depth
float32 temperature
float32 conductivity
float32 chlorophylA
...
```

SampleData

```
Header header
uint32 cellCount
uint32 cellId
...
```

PhyChemProfile

```
Header header
PhyChemData[] profile
```

WeatherData

```
Header header
float32 temperature
float32 windSpeed
float32 windAngle
...
```

Implementation

Getting started

- Follow the ROS tutorials on www.ros.org
- Start by creating all the ROS packages (roscreate-pkg).
- Continue with the message definitions (ideally in their own packages).

Going further

- Prepare and test nodes independently
- Start with hardware interfaces (green)
- ...

Outline

Introduction
- Some robots I worked with
- More Robots

Designing Robotic System

Robotic Middlewares
- ROS
- Open-source robotics and ROS

Use Cases
- The NIFTi Project
- The Limnobotics Project

Case Study

Summary

What did we learn?

On Robotics

- Many applications of robotics
- Many different sort of robots
- Not mentioned: humanoids, walking system, space robots, ...

On Robotic System Design

- Software challenges: complexity, reusability
- and also: reliability, robustness, precision, ...
- Middlewares make the complexity manageable
- ROS also helps for code sharing and publication

Where do we go from here?

On Robotics

- Understanding perception
- Designing controllers
- Understanding algorithm to understand the world

In Practice and Homework

- Analysis of existing systems.
- Simulation environment.
- Small projects to learn the concepts.

Home Work

Learn ROS

- Run through the tutorials: `http://www.ros.org/wiki/ROS/Tutorials`
- Tutorial 1 to 18, eventually 9, in C++ or Python.

Check the simulation environment

- Everything installed on GTL machines
- V-REP simulator (www.vrep.eu)

Home Work

Topic on Piazza, ressource tab.

Questions:

- What are the topic published/subscribed to by the simulator?
- How can you visualize the laser scans?
- How can you visualize the camera images?
- How can you make the robot move?
- Can you control the robot with a joystick?