Feedback on the use of ROS in the InFuse project

Ellon PAIVA MENDES
Pierre NARVOR

Journée ROS - robots mobiles terrestres
Clermont-Ferrand – 4 Juillet 2018

emendes@laas.fr
pnarvor@laas.fr

The InFuse project

"*Infusing Data Fusion in Space Robotics*"

- One of six projects of Space Robotics Technologies SRC (Horizon 2020)
- Aims to develop of a **Common Data Fusion Framework (CDFF)** building block
  - To serve through all SRC upcoming activities
- LAAS is only involved with the planetary rovers (not the orbital track)
  - CDFF development
  - Absolute map-based localization
  - Alternative perception techniques (hyper spectral cameras, lidars, etc)

What is CDFF?

- **Common Data Fusion Framework**
- Defines a functional architecture to integrate the data fusion process
  - that is flexible and generic
  - with clear inner and outer interfaces
  - expose products (maps and positions)
  - and algorithm models (to allow their control)

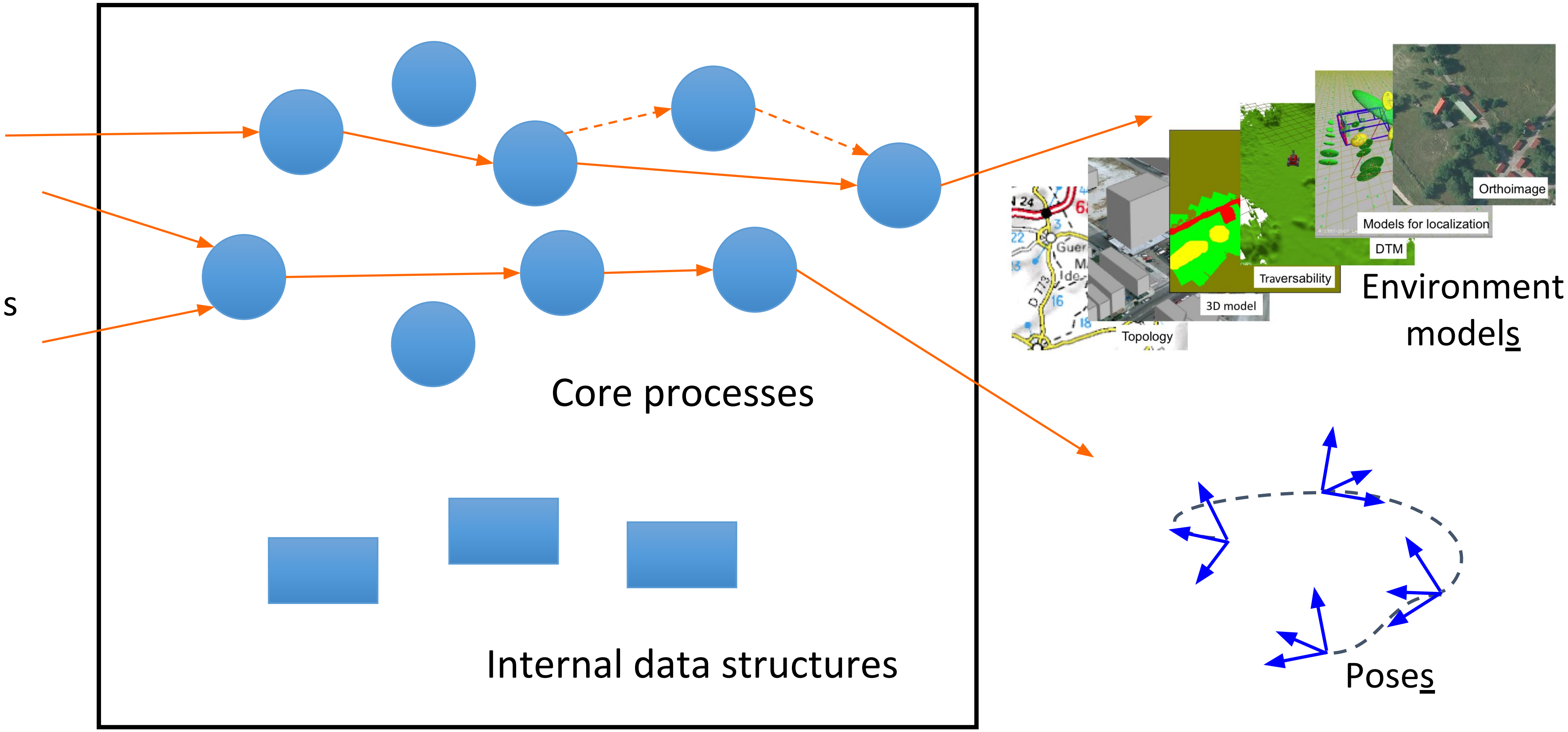
CDFF functional architecture

(inputs)

1. Acquired data
   - IMU
   - Images
   - Point clouds
   - …
2. Initial data & models
   - Orbiter maps
   - Satellite models
   - …
3. Knowledge
   - Terramechanics
   - Dynamics
   - …

CDFF

(outputs)
CDFF products



Project constraints

- ESROCOS should be used as Robot Control Operational System (RCOS)
  - Space oriented RCOS being developed in a parallel project
  - **Problem:** ESROCOS is not ready yet! → We decided to use ROS instead.

- Data exchanged between nodes should be described using ASN.1
  - Interface description language for defining data structures that can be serialized and deserialized in a standard, cross-platform way.
  - We decided to user ASN.1 over ROS messages.

ASN.1 Communication interface

- ASN.1 (Abstract Syntax Notation One) widely used communication standard.
  - Exchanged types defined in high level .asn abstract description files.

```
TASTE-BasicTypes DEFINITIONS ::=
BEGIN

-- Set of TASTE predefined basic types
T-Int32 ::=  INTEGER (-2147483648 .. 2147483647)
T-UInt32 ::= INTEGER (0 .. 4294967295)
T-Int8 ::= INTEGER (-128 .. 127)
T-UInt8 ::= INTEGER (0 .. 255)
T-Boolean ::= BOOLEAN

END
```

ASN.1 Communication interface

- ASN.1 (Abstract Syntax Notation One) widely used communication standard.
  - Exchanged types defined in high level .asn abstract description files.
- ASN1SCC : ESA's ASN.1 compiler for safety-critical embedded systems.
  - .asn compiled into C files with ready to use serialization functions

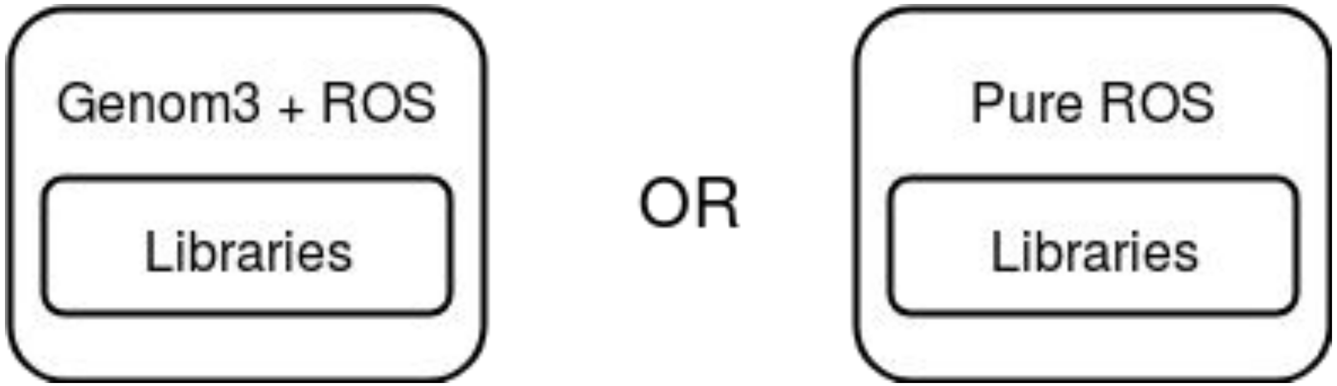
```
TASTE-BasicTypes DEFINITIONS ::=
BEGIN

-- Set of TASTE predefined basic types
T-Int32 ::=  INTEGER (-2147483648 .. 2147483647)
T-UInt32 ::= INTEGER (0 .. 4294967295)
T-Int8 ::= INTEGER (-128 .. 127)
T-UInt8 ::= INTEGER (0 .. 255)
T-Boolean ::= BOOLEAN

END
```

```
#ifndef GENERATED_ASN1SCC_TASTE_TYPES_H
#define GENERATED_ASN1SCC_TASTE_TYPES_H

typedef int T_Int32;

#define T_Int32_REQUIRED_BYTES_FOR_ENCODING      4
#define T_Int32_REQUIRED_BITS_FOR_ENCODING      32
#define T_Int32_REQUIRED_BYTES_FOR_ACN_ENCODING  4
#define T_Int32_REQUIRED_BITS_FOR_ACN_ENCODING  32
#define T_Int32_REQUIRED_BYTES_FOR_XER_ENCODING  39

flag T_Int32_Encode(const T_Int32* val, BitStream*
pBitStrm, int* pErrCode, flag bCheckConstraints);
flag T_Int32_Decode(T_Int32* pVal, BitStream* pBitStrm,
int* pErrCode);
...
```

Serialization functions

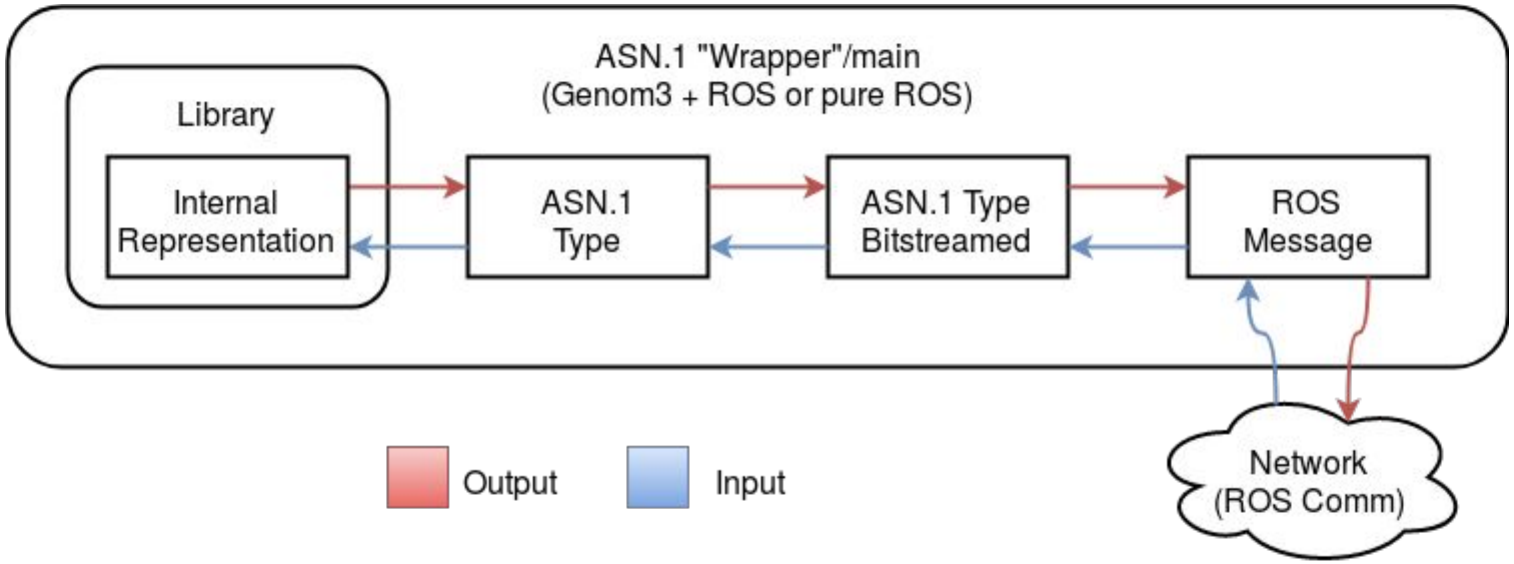
Separation between "core" and ASN.1

- Core functionalities are being written into libraries
- Libraries are wrapped in/used by a "main" program that may be written in Genom3 + ROS or pure ROS
- ASN.1 related code stays outside the libraries
- ROS is used as implementation middleware



ASN.1 and ROS

- Libraries use any internal representation for data (pcl, eigen, etc)
- Data is converted from/to an according ASN.1 type
- The ASN.1 type is converted to/from bitstream using asn1scc encode/decode functions
- Bitstream is stored in a ROS message for output to/input from the middleware



A single ROS message type

- Only <u>one</u> type of ROS message is exchanged between the modules

```
# ROS common header
std_msgs/Header header

# Identification of the type : ASN.1 name
string type

# Serialisation method : 0 (uPER), 1 (BER), 2 (XER)
uint8 serialization_method

# Buffer with ASN serialised data
uint8[] data
```

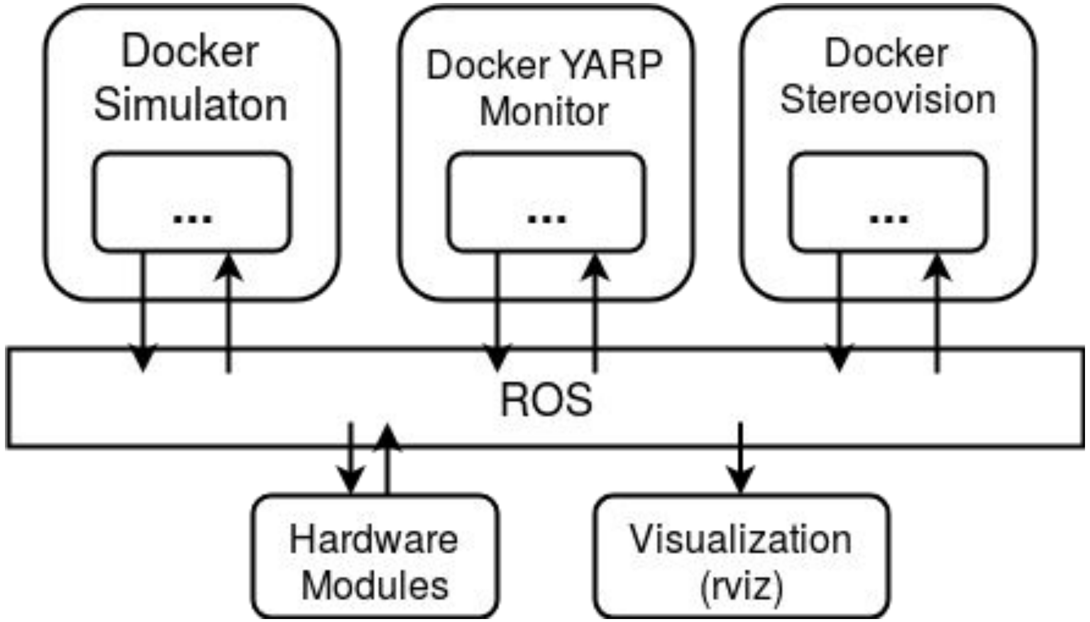
ASN.1 Tradeoffs

- Pros
  - International widely used standard, mature technology
  - Simple text notation for type definition with physical encoding rules
  - Independent from programming languages
  - ASN1SCC compiler : free, open-source, ready to use

- Cons
  - ASN.1 ROS message used are not human readable
  - Extra computation overhead (to be assessed)
  - Some encoding limitations (e.g. NaN or Inf encoding missing)

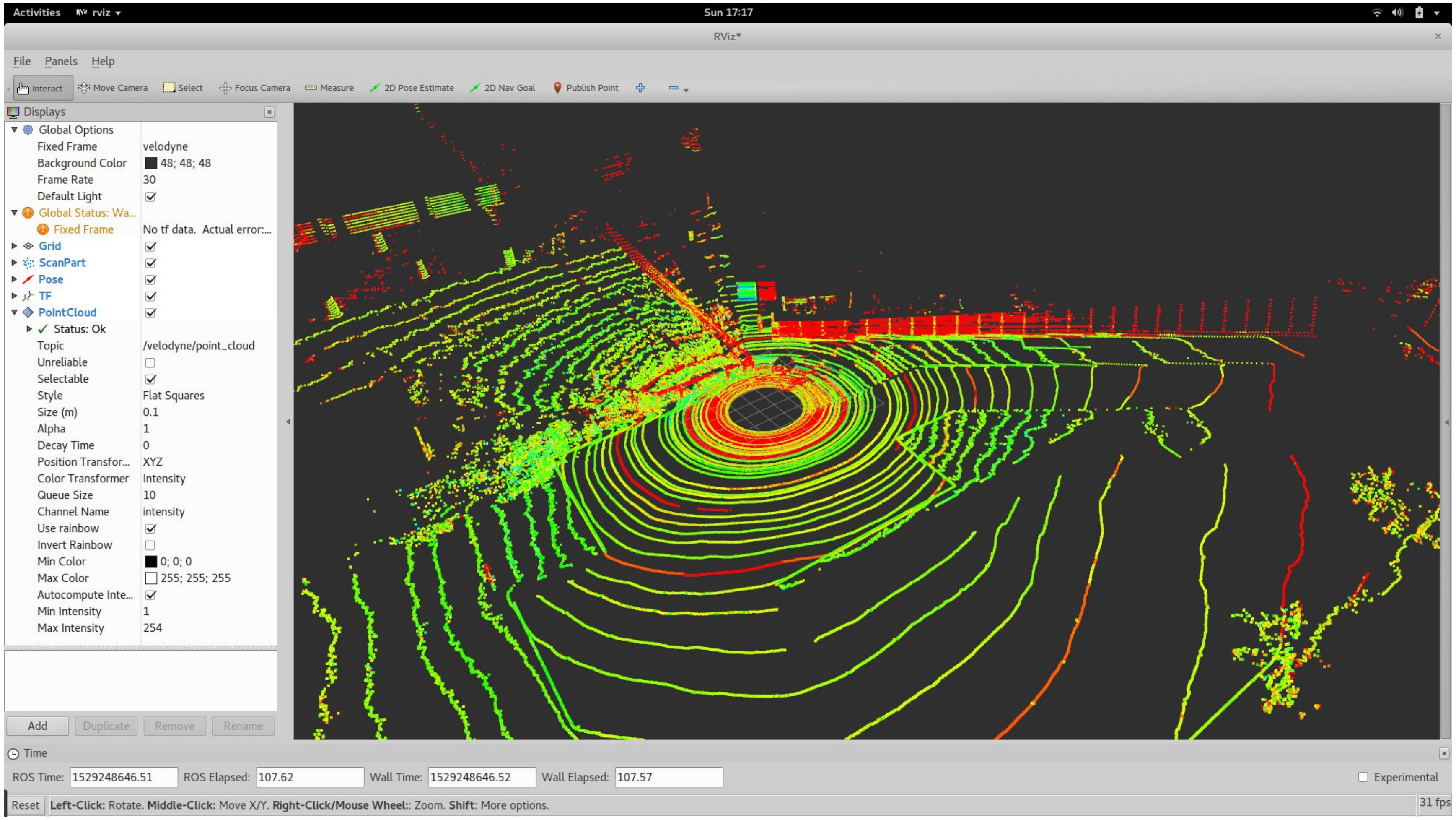
Integration with docker

- Part of the project may be supplied as dockers images
  - Allow easy use of software with closed source code
- Dockers communicate through ROS using the same type of ROS message

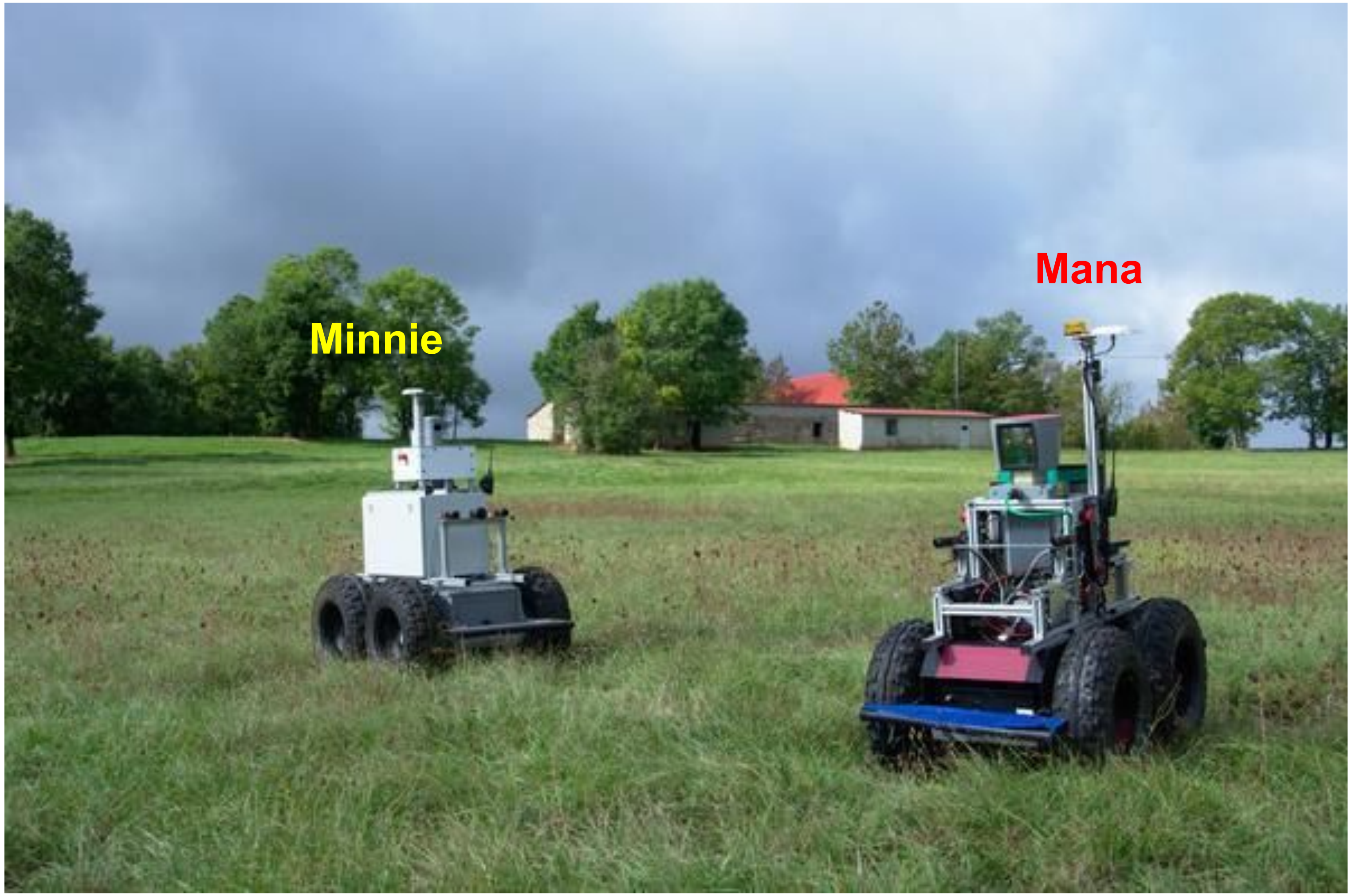


Visualization with rviz

- Rviz plugin implemented to decode and display data



LAAS Rovers *(not on Mars)*



LAAS Rovers Hardware

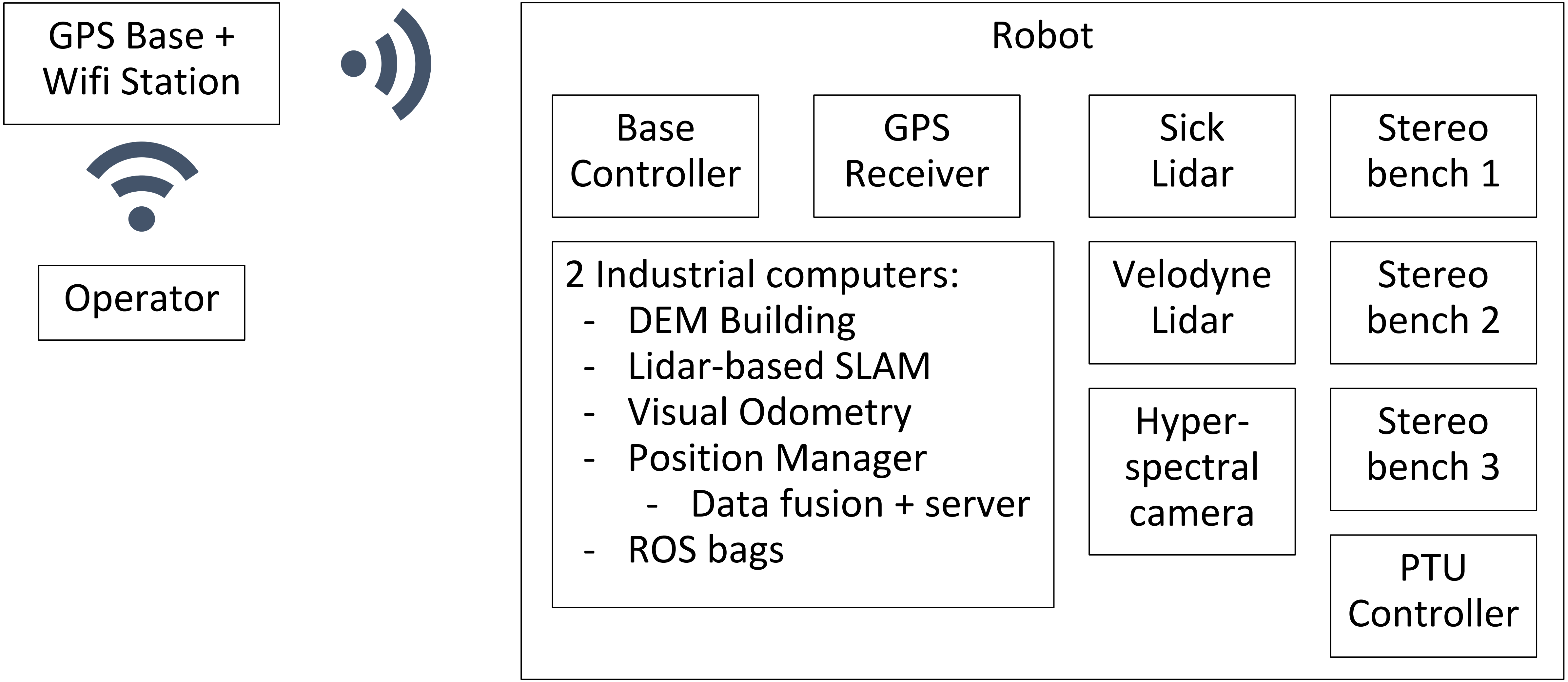
- Segway rmp400 and rmp440 platforms.
- Six-axis IMU
  - (accelerometer, gyrometer, magnetometer).
- One axis Fiber-optic gyro.
  - (100Hz, drift of 1 deg/hour after correction).
- RTK GPS.
  - (20Hz, cm accuracy => corrections provided by a nearby base).

LAAS Rovers Hardware

- Lidar rover (Mana):
  - Panoramic Lidar velodyne HDL64.
    - (Scan rate 5-20Hz, vertical FOV -24/+2deg, 1.4M points per second)

- Vision rover (Minnie):
  - Panoramic Lidar velodyne HDL32.
    - (Scan rate 10Hz, vertical FOV -30/+10deg, 700k points per second)
  - 1 NavCam stereo bench on a PTU.
  - Automotive Lidar
    - High resolution point cloud, FOV 110x90deg.
  - 2 HazCam fixed stereo benches (front and rear)

LAAS Rovers System Diagram



Feedback from trials at CNES

- First tests performed during the last two weeks
  - (in fact mostly integration work…)
  - Data acquisition in ROS bags with joystick controlled robots

- Needed to have some ASN.1 translator nodes here and there...
- Single ROS message + ASN.1 helped integration with partners (Magellium)
- Detected some latency/bandwidth problems
  - Less point clouds in the bags than expected (need to investigate why)
- Acquired datasets still being analysed.

Conclusion

- Not an optimal solution
- ROS being used only as communication layer + debugging
- Still a work in progress…

- Some questions:
    - Could we generate .msg from .asn files?
    - Should we use nodelets to reduce message passing?
    - Are we doing something very wrong here? :)