Deliverable Reference	:	D5.2
Title	:	PLANETARY RI AND ASSOCIATED EGSE DETAILED DESIGN
Confidentiality Level	:	PU
Lead Partner	:	MAGELLIUM
Abstract	:	The purpose of this document is to define the system architecture and detailed design to implement reference scenarios for the planetary track.
EC Grant N°	:	730014
Project Officer EC	:	Christos Ampatzis (REA)

InFuse is co-funded by the Horizon 2020
Framework Programme of the European Union

DOCUMENT APPROVAL SHEET			
	Name	Organization	Date
Prepared and cross-reviewed by:	Fabrice Souvannavong Vincent Bissonnette Jeremi Gancet Shashank Govindaraj	Magellium SPACEAPPS	30/03/2018
	Fabrice Souvannavong Vincent Bissonnette Arnaud Moura	Magellium	23/01/2018
	Simon Lacroix Quentin Labourey Andrea De Maio	CNRS/LAAS	
	Mark Post Alessandro Bianco Romain Michalec	Strathclyde University	
	Shashank Govindaraj Jeremi Gancet Xavier Martinez	SPACEAPPS	
	Nassir Oumer Michal Smisek	DLR	
	Raul Dominguez	DFKI	

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
0.1.0	20/10/2017	Vincent Bissonnette Fabrice Souvannavong	All	Initial release of template
0.2.0	30/11/2017	Vincent Bissonnette Fabrice Souvannavong Simon Lacroix Mark Post	Chapter 1 to 3	application scenario, system modelling
0.3.0	23/12/2017	All partners	Chapter 4 & 5	Adding DFN and DFPC descriptions
0.4.0	18/01/2018	Vincent Bissonnette	Chapter 4 & 5	Merging most contents of Section 5 to section 4
0.5.0	22/01/2018	Andrea De Maio	Chapter 4 & 5	Cross check of DFNs and DFPCs between D5.1 and D5.2
1.0.0	25/01/2018	Shashank Govindaraj Joseph Salini Vincent Bissonnette Jeremi Gancet	All Chapters	Final check of content and formatting the document
2.0.0	30/03/2018	Shashank Govindaraj Joseph Salini Vincent Bissonnette Jeremi Gancet	All Chapters	Addressing RIDs emitted by PSA.

Executive Summary

This document defines the system architecture and the detailed design to implement the reference scenarios of the planetary track. The system architecture focuses on the hardware architecture and the allocation of the software in the system, whereas the detailed design includes the definition of EGSE as well as the software detailed design which will rely on the CDFF advanced design described in D4.2.

The document addresses 2 reference implementations (i.e. integration and validation tracks):

- *RI-INFUSE*: it identifies scenarios at the consortium level whose objective is to demonstrate and evaluate the full capabilities of the CDFF, from space-compliant to state-of-the-art algorithms, from traditional to innovative sensors, and possibly including control in the loop;
- *RI-SRC-SR*: it identifies scenarios at the SRC Space Robotics level whose objective is to demonstrate that the CDFF is ready to be integrated with OG1, OG2, OG4 and OG6.

Table of Contents

1 Introduction **16**

1.1 Purpose 16

1.2 Document structure 16

1.3 Applicable Documents 17

1.4 Reference Documents 17

1.5 Acronyms 17

2 Demonstration and Validation Scenarios **19**

2.1 Introduction 19

2.2 Long Traverse Localisation 24

2.2.1 RI-INFUSE-LONG-TRAVERSE-LOC 24

2.2.1.1 Package diagram 25

2.2.1.2 Activity diagram 26

2.2.1.3 Expected results 27

2.2.1.4 Use Case 1 : Visual Odometry 27

2.2.1.5 Use Case 2 : Visual SLAM 28

2.2.1.6 Use Case 3 : LiDAR SLAM 28

2.2.2 RI-SRC.SR-LONG-TRAVERSE 29

2.3 Long Traverse DEM 29

2.3.1 RI-INFUSE-LONG-TRAVERSE-DEM 30

2.3.2 Package diagram 30

2.3.2.1 Activity diagram 31

2.3.2.2 Expected results 33

2.3.2.3 Use Case 1: Absolute Localisation 33
2.4 Rendezvous 33
2.4.1 RI-INFUSE-LONG-RANGE-TRACKING 34
2.4.1.1 Expected Results 34
2.4.1.2 Package Diagram 35
2.4.1.3 Activity Diagram 36
2.4.1.4 Use Case 1: Long-range Bearing-only Target Tracking 36
2.4.2 RI-INFUSE-RENDEZVOUS 37
2.4.2.1 Expected Results 38
2.4.2.2 Package Diagram 39
2.4.2.3 Activity Diagram 40
2.4.2.4 Use Case 1: RGB-D Model-based Detection and Tracking 41
2.4.2.5 Use Case 2: Dense Point Cloud Model-based Localisation 41
2.4.2.6 Use Case 3: 3D Feature Model-based Localisation 42
2.4.2.7 Use Case 4: 3D Reconstruction from Structure-From-Motion 43
2.4.3 RI-SRC.SR-RENDEZVOUS 43
2.5 Return to Base 44
2.5.1 RI-INFUSE-RETURN-TO-BASE 44
2.5.1.1 Expected Results 44
2.5.1.2 Package Diagram 45
2.5.1.3 Activity Diagram 46
2.5.1.4 Use Case 1 : Visual Map-Based Localisation 46
2.5.1.5 Use Case 2 : Point Cloud Map-Based Localisation 47
2.5.2 RI-SRC.SR-RETURN-TO-BASE 48

2.6 Requirements 48

3 System Modeling 49

3.1 Robotics System 49

3.2 RI-INFUSE 50

3.2.1 System Components 50

3.2.2 System Architecture 52

3.2.2.1 Visual localisation view 52

3.2.2.2 Autonomous navigation view 52

3.2.2.3 Relative Localisation view 53

3.2.2.4 Rover Data Product Manager 54

3.3 RI-SRC.SR 56

4 Detailed Architecture and Design 57

4.1 Flight software : DFPC Architecture and Design 57

4.1.1 DFPC: Wheel Odometry 59

4.1.2 DFPC : Visual Odometry 60

4.1.2.1 Flavor 1: MAG/CNES Visual Odometry Implementation 60

4.1.2.2 Flavor 2: LAAS Visual Odometry 63

4.1.3 DFPC : Visual SLAM 66

4.1.4 DFPC : Visual Map-based Localisation 70

4.1.5 DFPC: Long-range Tracking 74

4.1.6 DFPC : Mid-range 3D Model Detection 77

4.1.7 DFPC : Mid-range 3D Model Tracking 80

4.1.8 DFPC : Point Cloud Model-Based Localisation 82

4.1.8.1 Flavor 1: ICP Point Cloud Matching 83

4.1.8.2 Flavor 2: SHOT 3D Feature Matching and EKF 86
4.1.9 DFPC : Absolute Localisation 89
4.1.10 DFPC: DEM Building 92
4.1.11 DFPC: Lidar SLAM 94
4.1.12 DFPC : Lidar Map-based Localisation 97
4.1.13 DFPC: Navigation Map Building 100
4.1.14 DFPC: Path Planning 103
4.1.15 DFPC: Pose Fusion 103
4.1.16 DFPC : 3D Model Detection and Tracking 104
4.1.17 DFPC : Haptic scanning 105
4.2 Flight software : DFN Detailed Design 106
4.2.1 DFN: Template DFN 106
4.2.1.1 DFN Description 107
4.2.1.2 DFI: Template Implementation 107
4.2.1.3 DFN Description File 108
4.2.1.4 DFN Sequence Diagram 108
4.2.2 DFN Detailed Design 108
4.2.2.1 DFN: FeatAndSigExtractor 108
4.2.2.1.1 DFI: SIFT Feature Extractor 109
4.2.2.1.2 DFI: SURF Feature Extractor 110
4.2.2.1.3 DFI: ORB Feature Extractor 110
4.2.2.2 DFN: Feature Matching 110
4.2.2.2.1 DFI: FLANN Matcher 111
4.2.2.3 DFN: 3D Point Computation 111

4.2.2.3.1 DFI: Linear Triangulation (DLT) 112
4.2.2.4 DFN: 3D-3D Motion Estimation 112
4.2.2.4.1 DFI: SVD Rigid Body Transformation 113
4.2.2.5 DFN: 2D-3D Motion Estimation 113
4.2.2.5.1 DFI: PnP (Perspective from n-Points) 113
4.2.2.6 DFN: Point Cloud Construction 114
4.2.2.6.1 DFI: Point Cloud Builder 114
4.2.2.7 DFN: Bundle Adjustment 114
4.2.2.7.1 DFI: Bundle Adjustment 115
4.2.2.8 DFN: Fundamental Matrix Calculation 115
4.2.2.8.1 DFI: Fundamental Matrix Calculator 116
4.2.2.9 DFN: Estimation Filter 116
4.2.2.9.1 DFI: Extended Kalman Filter 116
4.2.2.10 DFN: Image Geometric Processing 117
4.2.2.10.1 DFI: Image Undistortion 117
4.2.3 Remaining DFNs 117
4.3 Ground software : DFPC Architecture and Design 118
4.3.1 DFPC: Camera calibration 118
4.3.2 DFPC: Stereo calibration 118
4.3.3 DFPC: Body/ Camera calibration 119
5 Detailed Description of EGSEs 120
5.1 DLR EGSE 120
5.1.1 HCRU 121
5.1.2 BB2 Rover 121

5.1.3 Planetary Exploration Lab 122
5.2 DFKI SherpaTT Rover 123
5.3 LAAS Rovers 125
5.3.1 Mana 126
5.3.2 Minnie 126
5.3.3 Ground station 127
5.3.4 CNES SEROM Mars Yard 127
6 Conclusion 132
7 Appendix 133
7.1 Definition of Reference Frames 133
7.1.1 Geo-referenced frames 133
7.1.2 Local Frames 134
7.1.3 Mathematical Conventions and Notations 136
7.1.4 References 136
7.2 Design Requirements 137
7.3 Technical Note on DFN and DFPC Specification 137
7.3.1 Scope of this Note 137
7.3.2 Definitions 137
7.3.2.1 DFN 137
7.3.2.2 DFPC 138
7.3.3 DFN Template 139
7.3.3.1 DFN Template Elements 139
7.3.3.2 Towards a Typology/Taxonomy of DFNs 139
7.3.3.2.1 DFN Characterization 140

7.3.3.2.2 DFI Characterization 140
7.3.4 DFPC Description Template 140
7.3.4.1 DFPC Data Flow Description 141
7.3.4.2 DFPC Data Product Management 142
7.3.4.2.1 Shared Data Structures 142
7.3.4.2.2 Processing Units - Queriers 142
7.3.4.3 DFPC Control Flow Description 143
7.4 Architecture of the Data Product Manager 144
7.4.1 Introduction 145
7.4.1.1 General concerns 145
7.4.1.2 Definition of the DPM 147
7.4.2 DPM use cases 147
7.4.2.1 Integrate estimation of past poses and motion estimates 147
7.4.2.2 DTM building 149
7.4.2.3 Exploitation of environment models produced by the CDFF 151
7.4.2.4 Science target localisation 153
7.4.2.5 Multi-robot use cases 154
7.4.2.6 Serve the operators 155
7.4.3 Synthesis: DPM functionalities 155
7.4.3.1 DPM services 155
7.4.3.2 Stored and managed data 156
7.4.3.3 DPM functions 156
7.4.4 Architecture and implementation design 157
7.4.4.1 Managing poses 157

7.4.4.2 Using Envire for managing poses 161

7.4.4.3 Managing environment models 162

7.4.4.4 Using Envire for the Management of Environment Models 162

7.4.4.5 Envire usage summary 162

List of of Figures

Figure 1: Illustration of the demonstration strategy. Reminder of the role of OG2 and OG4 with respect to InFuse.
Figure 2: Illustration of the validation and evaluation strategy.
Figure 3: rover suppliers (but 4 rovers) and 3 test fields are currently identified for testing and validation of InFuse.
Figure 4: Package diagram of the long range traverse scenario.
Figure 5: Activity diagram of the long range traverse scenario.
Figure 6: Package diagram of the long traverse scenario / DEM building.
Figure 7: Activity diagram of the long traverse scenario / DEM building.
Figure 8: Package diagram of the long range tracking scenario.
Figure 9: Activity diagram of the long range tracking scenario.
Figure 10: Package diagram of the rendez-vous scenario.
Figure 11: Activity diagram of the rendez-vous scenario.
Figure 12: Package diagram of the return to base scenario.
Figure 13: Activity diagram of the return to base scenario.
Figure 14: Components within a generic robotics system
Figure 15: Visual localisation view
Figure 16: Autonomous navigation view
Figure 17: Relative localisation view
Figure 18: Rover data product manager
Figure 19: Deployment of InFuse with CNRS/LAAS rovers.
Figure 20: Deployment of InFuse with MORSE simulator.
Figure 21: Deployment of InFuse for offline processing.
Figure 22: Deployment of InFuse with HRCU and SHERPA.
Figure 23: Wheel Odometry Data Flow Description
Figure 24: Visual Odometry - MAG/CNES Data Flow Description.
Figure 25: Visual Odometry - MAG/CNES Data Product Management.
Figure 26: Visual Odometry - MAG/CNES Control Description.
Figure 27: Visual Odometry - LAAS Data Flow Description
Figure 28: Visual Odometry - LAAS Data Product Management
Figure 29: Visual Odometry - LAAS Control Description
Figure 30: Visual SLAM Data Flow Description
Figure 31: Visual SLAM Data Product Management
Figure 33: Visual SLAM Control Description
Figure 33: Visual SLAM Control Description
Figure 34: Visual Map-Based Localisation Data Flow Description
Figure 35: Visual Map-Based Localisation Data Product Management
Figure 36: Visual Map-Based Localisation DFPC Control Description.

Figure 37: Long-range Tracking Data Flow Description.
Figure 38: Long-range Tracking Data Product Management.
Figure 39: Long-range Tracking Control Description
Figure 40: Mid-range 3D Model Detection Data Flow Description.
Figure 41: Mid-range 3D Model Detection Data Product Management
Figure 42: Mid-range 3D Model Detection Control Description
Figure 43: Mid-range 3D Model Tracking Data Flow Description
Figure 44: Mid-range 3D Model Tracking Data Product Management
Figure 45: Mid-range 3D Model Tracking Control Description
Figure 46: ICP Point Cloud Matching Data Flow Description
Figure 47: ICP Point Cloud Matching Data Product Management
Figure 48: ICP Point Cloud Matching Control Description
Figure 49: details the DFN component structure inside the DFPC.
Figure 50: 3D target tracking timing diagram
Figure 52: Absolute Localization DFPC Data Flow Description
Figure 53: Absolute Localization DFPC Data Product Management
Figure 54: Absolute Localisation Control Description
Figure 55: DEM Building Data Flow Description
Figure 56: DEM building DFPC Data Product Management
Figure 57: DEM building DFPC Control Description
Figure 58: LIDAR-SLAM Data Flow Description
Figure 59: LIDAR-SLAM-DFPC Data Product Management
Figure 60: LIDAR-SLAM DFPC Control Description
Figure 61: LIDAR Map-based Localization Data Flow Description
Figure 62: LIDAR map-based localization Data Product Management
Figure 63: LIDAR Map-based Localisation Control Description
Figure 64: Navigation Map Building Data Flow Description
Figure 65: Navigation Map Building Data Management
Figure 66: Navigation Map Building Control Description
Figure 67: 3D Model Detection and Tracking Process
Figure 69: Illustrations of DLR EGSE, from left to right: (1) HCRU basic setup, (2) HCRU sensoric setup, (3) BB2 in PEL.
Figure 69: Illustrations of DLR EGSE, from left to right: (1) HCRU basic setup, (2) HCRU sensoric setup, (3) BB2 in PEL.
Figure 70: ExoMars Phase B2 Breadboard (BB2)
Figure 71: PEL Pose Tracking and DSM Device
Figure 72: SherpaTT in two space analog scenarios
Figure 73: API Provided by OG6 to exchange data with SherpaTT
Figure 74: The robots Mana (right) and Minnie (left), pictured in autumn 2015.
Figure 75: Aerial View of CNES Mars Yard (Credits CNES)
Figure 76: Area with a high density of small rocks

Figure 77: Area with large diameter rocks, i.e. forbidden obstacles for the autonomous navigation mode
Figure 78: Flat and obstacle free area
Figure 79: Sandy dune on the left hand side of the yard
Figure 80: Valley on the left of the dune
Figure 81: Geo-referenced frames (ECEF)
Figure 82: Local terrain frame
Figure 83: Illustration of the RoverBodyFrame (RBF) (EADS-Astrium/ExoMars)
Figure 84: Illustration of the CameraFrame (CF) (left) and ImageFrame (IF) (right)
Figure 85: A simple schematic view of DFNs and DFPCs.
Figure 86: Example of a possible implementations of a DFN
Figure 87: LIDAR-PG-SLAM
Figure 88: Hierarchy of queriers for PG LIDAR-based SLAM
Figure 89: PG-LIDAR-based SLAM sequence diagram
Figure 90: Relations between the various kinds of DFPCs and the DPM
Figure 91: Interfaces with the DPM for the fusion of wheel and visual odometries
Figure 92: Fusion of two independent motion estimation processes, Wheel Odometry and Visual Odometry.
Figure 93: Role of the DPM with respect to the DTM Building DFPC
Figure 94: Role of the DPM with respect to the Absolute Localization DFPC when it produces the Total Rover Map
Figure 95: Role of the DPM with respect to the Map-Based Localization DFPCs
Figure 96: Role of the DPM with respect to the Absolute Localization DFPC
Figure 97: Timeline of the various pose estimates handled by the DPM.
Figure 51: The DPFC’s inputs are the following:

List of of Tables

Table 1: Table 1 provides a comparison of proposed validation approaches.
Table 2: Table 2 presents a summary of all DFPCs to be developed and the partners responsible for their description and development. Each of the following sections presents the architecture description of one of these DFPCs.
Table 3: Mid-range 3D Model Tracking Expected Accuracy Figures
Table 4: Typical performance ratio for the navigation map building function

1 Introduction

1.1 Purpose

The purpose of this document is to define the system architecture and detailed design to implement reference scenarios for the planetary track. The system architecture will mainly focus on the hardware architecture and the allocation of the software in the system. The detailed design will include the definition of EGSE as well as the software detailed design which will rely on the CDFF advanced design described in D4.2.

The document will address 2 reference implementations (i.e. integration and validation tracks):

- *RI-INFUSE*: the consortium level, in which the objective is to demonstrate and evaluate the full capabilities of the CDFF, from space-compliant to state-of-the-art algorithms, from traditional to innovative sensors, and possibly including control in the loop;
- *RI-SRC-SR*: the SRC Space Robotics level, where the objective is to demonstrate that the CDFF is ready to be integrated with OG1, OG2, OG4 and OG6.

1.2 Document structure

In brief, the document is structured as follows:

Section 1: This introductory material.

Section 2: Demonstration and validation scenarios. This chapter provides a detailed description of the scenario that will be implemented in the scope of InFuse.

Section 3: System modeling. This chapter describes the system architecture including EGSE, simulation tools and InFuse.

Section 4: Detailed architecture. This chapter details the software architecture of data fusion processing compounds (DFPC). It includes the decomposition of DFPC in data fusion nodes (DFN) and the corresponding sequence diagram.

Section 5: Detailed design. This chapter covers the software and hardware detailed design. It describes inputs, outputs and parameters of DFPC and DFN. It describes EGSE involved testing and validation, both internally to OG3 and with OG6.

Section 6: Conclusion and appendices. This chapter closes the document with a description of the work done and to pursue. In appendices, we find the template that is used to describe DFPC and DFN, and the definition of frames.

1.3 Applicable Documents

AD1 InFuse Grant Agreement

AD2 InFuse Consortium Agreement

AD3 InFuse internal management manual for project partners

1.4 Reference Documents

RD1 [InFuse_D3.1] Infuse Consortium, “Technological Review”, Jan 2017

RD2 [InFuse_D4.1] Infuse Consortium, “Technical trade-offs analysis”, Jun 2017

RD3 [InFuse_D4.2] Infuse Consortium, “Advanced CDFF architecture and ICD”, Jun 2017

1.5 Acronyms

DF: Data Fusion

RCOS: Robot Control Operating System

DFNCI: Data Fusion Node Common Interface

FPGA: Field-Programmable Gate Array

MW: Middleware

Fps: Frames per second

OG: Operational Grant

AHRS: Attitude and Heading Reference System

API: Application Program Interface

CDFF: Common Data Fusion Framework

DEM: Digital Elevation Map

DFN: Data Fusion Node

DFPC: Data Fusion Processing Compound

DOF: Degree Of Freedom

DPM: Data Product Manager

DSM: Digital Surface Model

EGSE: Electrical Ground Support Equipment

EKF: Extended Kalman Filter

FOG: Fiber Optics Gyroscope

HCRU: Handheld Central Rover Unit

IMU: Inertial Measurement Unit

KLT: Kanade-Lucas-Tomasi

LiDAR: Light Detection and Ranging

OBC: On Board Computer

ORB: Oriented FAST and Rotated BRIEF

PnP: Perspective n-Point

RGB: Red Green Blue

RGB-D : Red Green Blue and Depth

RI: Reference Implementation

ROI: Region of Interest

RTK: Real Time Kinematics

SLAM: Simultaneous Localisation and Mapping

SVD: Singular Value Decomposition

TRL: Technology Readiness Level

ZNCC: Zero-Norm Cross-Correlation

2 Demonstration and Validation Scenarios

2.1 Introduction

The goal of this chapter is to provide a set of use cases and requirements that will drive the design and implementation of InFuse. For this purpose, we start by describing the principles of the demonstration and validation strategies selected in InFuse. Considering this and identified operational scenarios and baseline solutions, we introduce and detail a set of use cases that will be implemented.

The key principle of the demonstration and validation strategies is to offer a dual approach that will allow to demonstrate, first, that InFuse is ready for integration at the system level to implement sophisticated scenarios in the SRC SPACE ROBOTICS; second that expected performances (ressources consumption; localisation, DEM, fusion accuracy) are met.

The first strategy, depicted in Figure 1, targets online demonstration with an autonomous rover (or hardware setup) and simulation tools. Simulation is required to prepare the work for integration with a rover and conduct preliminary evaluations. Thus, first we will work on the integration of InFuse with a simulator. And as soon as InFuse is operational with the simulator, we start the integration with a rover. Of course, we might have different simulators depending on the targeted rovers. The rover should have autonomous navigation capabilities in order to execute representative trajectories (i.e. relevant rotations and trajectory shapes). Moreover, it should be capable of using InFuse data products like the estimated localisation and DEMs. In this demonstration context, the main CDFF part involved is CDFF-Support which provides the interfaces with OG2 and OG4. The OG4 sensors and interfaces will be simulated with our own sensors. In some dedicated use cases, we will target control in the loop and simulate some interfaces and functions of OG2.

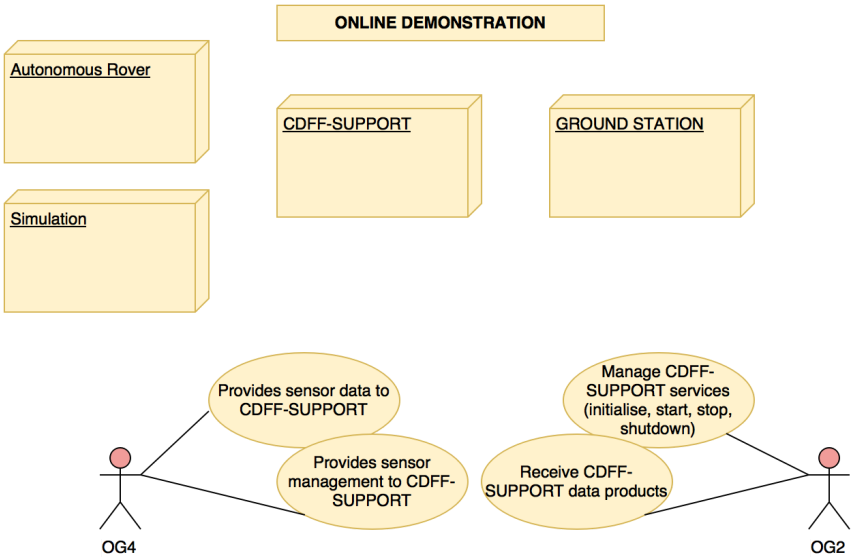


Figure 1: Illustration of the demonstration strategy. Reminder of the role of OG2 and OG4 with respect to InFuse.

The second strategy, depicted in Image 2, targets offline validation and evaluation to qualify InFuse performances. It will rely on recorded data provided by one or more acquisition setups like an Autonomous Rover or a HCRU; and possibly simulation tools. In this context, the main CDFF parts involved are CDFF-Core and CDFF-Dev.

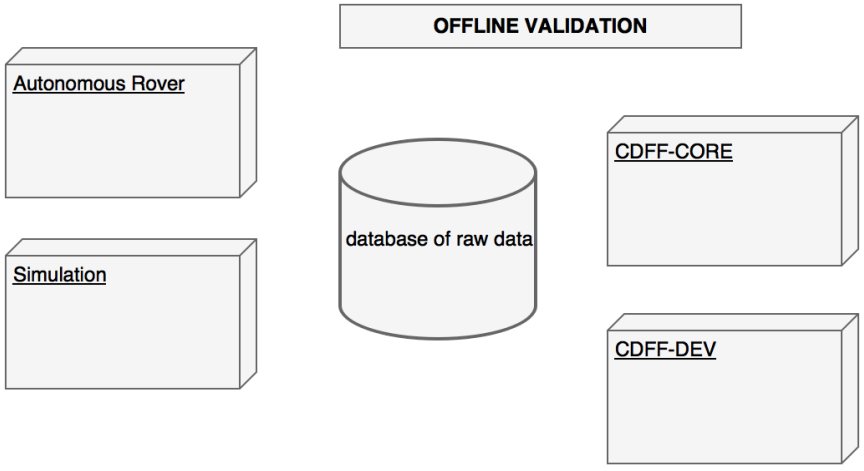


Figure 2: Illustration of the validation and evaluation strategy.

Three test fields are currently identified for demonstrations or to acquire data. CNES test field in Toulouse, DLR test field in Munich and a test site in Morocco.

Going to Morocco is a necessity. The EuroPlanet research infrastructure enables tests in a variety of Mars analogue terrains in Morocco, over large spatial scales, which has nothing to compare with planetary test fields such as the CNES SEROM or DLR test fields. These characteristics are necessary to test most of the scenarios addressed by InFuse, which imply the following functionalities: long range localisation, localisation while returning to base scenario, absolute localisation using orbital terrain maps.

Besides these obvious considerations, the benefits of deploying two robots on such terrains are:

- The possibility to gather datasets with an unprecedented completeness, in both the kind of terrains and sensors. Such datasets will foster future work, for instance on long range multi-robot setups.
- The thoroughness of the validations, brought be the variety and scale of the environments
- The impact, both within InFuse and in term of dissemination. Both the datasets and the obtained results will be publishable in high level conferences and journals (which is now becoming difficult to do when testing in non-realistic terrains).
- Last (and maybe not least), this goal has already shown to be a highly motivating one for the involved teams.

The hardware setups, ideally Autonomous Rovers, which will be used for demonstration and validation, are:

- CNRS/LAAS Minnie and Mana rovers deployed on the CNES SEROM test field and then in Morocco,
- DLR BB2 and HCRU deployed at DLR,
- DFKI Sherpa possibly with DLR HCRU in Morocco.

Table 1: Table 1 provides a comparison of proposed validation approaches.

	CNRS Mana (+ Minnie)	**DLR Exomars BB2 (+HCRU)**	**DFKI Sherpa (+HCRU TBC)**
Platform representativeness as a planetary rover	*	***	**
Test and validation scope	OG3 internal	OG6	OG6
Indoor / Outdoor (close vicinity) / Morocco	I/O/M	I	M
Closed / Open control loop	Partially closed	Open	Open
Middleware	GenoM (bridged with ROS)	ROS (bridged with GenoM)	ROCK (bridged with ESROCOS)
OG3 partners responsible	CNRS + MAG	SPACEAPPS + DLR	DFKI + USTRAT
CDFF coverage	Full	Near full (tbc)	Partial
Motivation and benefit	Extensive test setup (sensors wise) with partially closed loop control. The Mana and Minnie mobile robots from LAAS will be deployed in the Morocco outdoor analog environment with the same setup (sensors	Indoor tests in DLR facility (PEL) is representative of Martian soil terramechanics => realistic odometry can be expected. Furthermore, ground-truth data is available, which is useful for the consortium. Would reuse almost the	Would take benefit from the Sherpa system in place, with its own sensors, to test and validate InFuse in analogue conditions. Might be complemented by the HCRU, if attached to the Sherpa. OG3 would bring their own OBC to run the CDFF, with

	+ platform) used in OG3 internal testing. Advantages are - performing data fusion for long range navigation in a representative planetary environment and testing the CDFF with a multi-robot setup.	same CDFF software setup as the CNRS Mana one (tbc).	a ROCK flavor of the CDFF (tbc). No extra OG3 sensors would be added.
Potential concerns and constraints	Delta effort to make CNRS' platform ready for a Morocco campaign should be accommodated by the consortium. W.r.t Mana & Minnie, there is a low risk associated to the functional capabilities of the rovers and sensors as they are in a mature state and will be tested internally in OG3 with the CDFF software. The additional efforts and costs associated with the deployment in Morocco has to be analysed.	Need to equip DLR Exomars with a selection of OG3 sensors allowing to carry out an extensive test campaign. Physical interfacing to be agreed with DLR, and implemented (efforts to be assessed).	Availability of the Sherpa ahead of the Morocco's trials, for preparation purpose, may be very limited. Risk of interference with OG2.

Table 1: 3 Validation Approaches

After completion of WP4, in which the technical trade-offs analysis yielded the definition of a series of data fusion functionalities devoted to localisation and environment mapping for the planetary track [D4.1], we opted to use two robots instead of one:

- One robot will mostly be devoted to vision processes, with two large field of view stereoscopic benches rigidly mounted on the front and rear of the robot ("a la HazCam", and a third more resolved and smaller field of view stereoscopic bench mounted on an 2-axes orientable unit on the top of the robot ("à la NavCam"). A polarimetric and a hyperspectral camera will also be mounted on this unit.

- Another one will mostly be devoted to LiDAR processes, with a panoramic LiDAR (Velodyne HDL64 model) mounted on top, and a more precise and more resolved LiDAR mounted on a single orientable unit in front of the robot (this latter LiDAR having the ability to deliver multiple echoes).

The choice of using two robots instead of one has been made by realizing that integrating this whole sensor suite on-board a single robot was not possible. Furthermore, two robots will allow the testing of data fusion processes in the multi-robot case, which is one of the objective of InFuse.

The two platforms are already available at LAAS and fully equipped with a odometry, a high grade FOG gyro, a 6-axis low cost IMU, and cm-accuracy RTK GPS. Instrumenting the two robots will be made at no extra cost, as all other exteroceptive sensors are already available at LAAS (besides the procurement of some additional cameras and lenses, provisioned in the LAAS InFuse budget).

LAAS and MAG plan to integrate path and itinerary planning processes to demonstrate some of the scenarios depicted in [D4.1]. The objective is to assess the quality of the data fusion processes in realistic uses cases, in which the data fusion processes are integrated within a whole series of parallel processes that are required on-board. The experience of partners is that such an integration reveals issues that can not be detected (and hence neither addressed) when processing datasets offline: exploiting data products on-line will lead to more complete and thorough validation of the data fusion processes, and increase their TRL.

Figure 3: rover suppliers (but 4 rovers) and 3 test fields are currently identified for testing and validation of InFuse.

These test fields and rovers will be used to implement the envisioned demonstration and validation scenarios. Proposed scenarios relies on operational scenarios and baseline solutions identified in InFuse D3.1 and InFuse D4.1. Each scenario focuses on a specific operational concept and can be split into several use cases to address the different data products involved, the various sensors that can be used, or the different data fusion strategies that can be involved (DFPC). These variations of a DFPC are further detailed in section 4.1.

In this document we focus on the planetary track. Related operational concepts are assumed to be known by the reader, otherwise we invite him to first read InFuse D3.1. We address the following scenarios :

- Long Traverse Localisation,
- Long Traverse DEM,
- Rendezvous,
- Return to Base.

The long traverse scenario has been split into two scenarios, addressing respectively localisation and DEM data products. For each scenario we detail a reference implementation. RI-INFUSE addresses the reference implementation internal to InFuse that will be done on CNES premises, while RI-SRC-SR addresses the reference implementation with OG6 and possibly other OGs that will be done on DLR premises or in Morocco.

Discussions are still ongoing at the time of writing, but they tend to converge towards a new scheme where RI-INFUSE would first be executed at CNES premises and then reconducted in Morocco, while RI-SRC-SR will address a subset of RI-INFUSE at DLR premises for a precise evaluation of performances and in Morocco for an integrated demonstration with OG1 at least.

2.2 Long Traverse Localisation

The objective of the mission for the rover is to autonomously reach a target located about 1 km away, defined by its absolute coordinates. The localisation function is a key element for the success of the mission. Indeed, localisation data is used by three components of the system: trajectory control, fusion of navigation maps (or DEM), and localisation of the target with respect to the rover. It includes the rover pose and its uncertainties.

2.2.1 RI-INFUSE-LONG-TRAVERSE-LOC

The long traverse reference implementation in InFuse will be carried out on LAAS rovers and on the SEROM CNES site. Its objective is to conduct an online demonstration and offline validation of the localisation functions proposed in InFuse.

The offline validation will consist in two stages, first data collection, next data exploitation. The rover will execute a set of trajectories that will be defined later in the project. Each trajectory will be designed to address different complexity levels depending on the terrain, the surface type, the rock distribution,...

The online demonstration will consist of the execution of a limited set of localisation functions that will feed the locomotion system (trajectory control).

The online demonstration might include an autonomous navigation stack to validate the robustness of the function with respect to rover motions. Nevertheless, the navigation and locomotion stacks, will only use an RTK-GPS, an AHRS, and possibly wheel odometry to

implement their localisation function. In an extended demonstration scenario, the RTK-GPS could be removed and replaced by vision-based localisation.

We propose 4 use cases related to the localisation function to answer to localisation needs in a long traverse scenario :

- Visual Odometry : this implementation is compatible with current space-grade OBC and offers fairly good localisation accuracy,
- Visual SLAM : this implementation is designed for next generation space-grade OBC and offers state-of-the-art localisation accuracy,
- LiDAR SLAM : this implementation is designed for LiDAR sensors,
- Visual / LiDAR SLAM : this implementation explores the complementarity of visual and LiDAR SLAM.

2.2.1.1 Package diagram

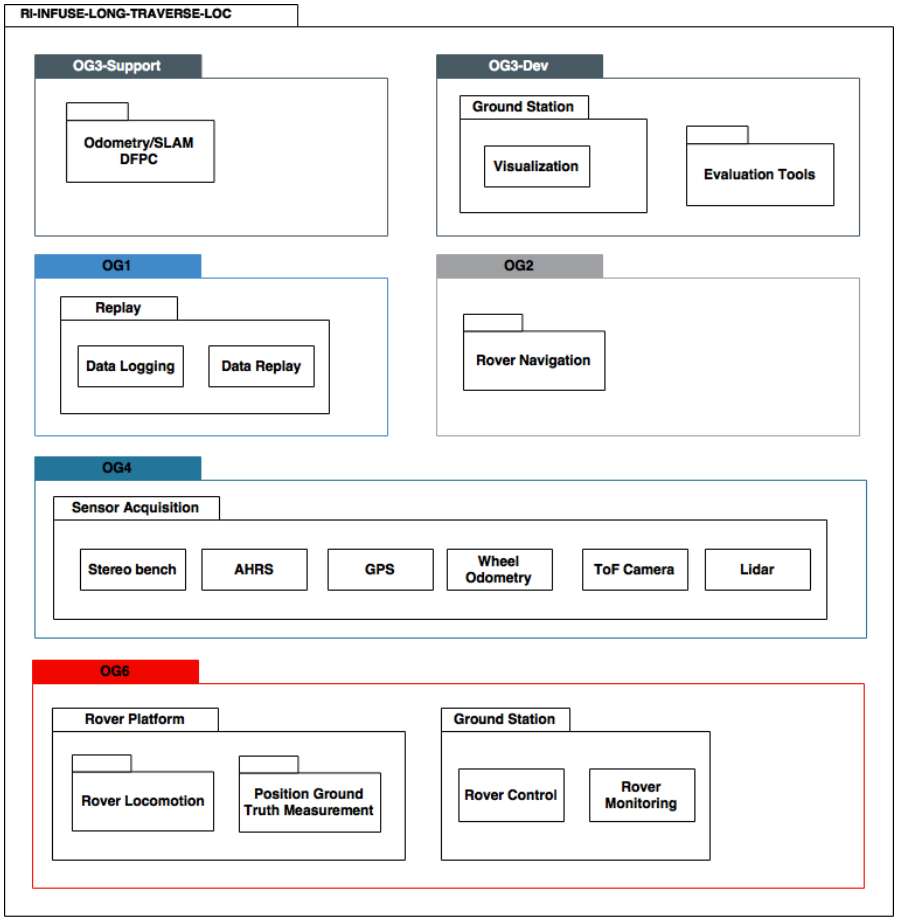


Figure 4: Package diagram of the long range traverse scenario.

The previous package diagram presents a summary of software and hardware elements foreseen to be required, in addition to the validated DFPC, in order to carry out all use cases of this reference implementation. The OG3, OG4 and OG6 labels are only specified here to emphasize the perimeter of the packages. **All elements will still be provided by InFuse or OG6 for this RI.**

2.2.1.2 Activity diagram

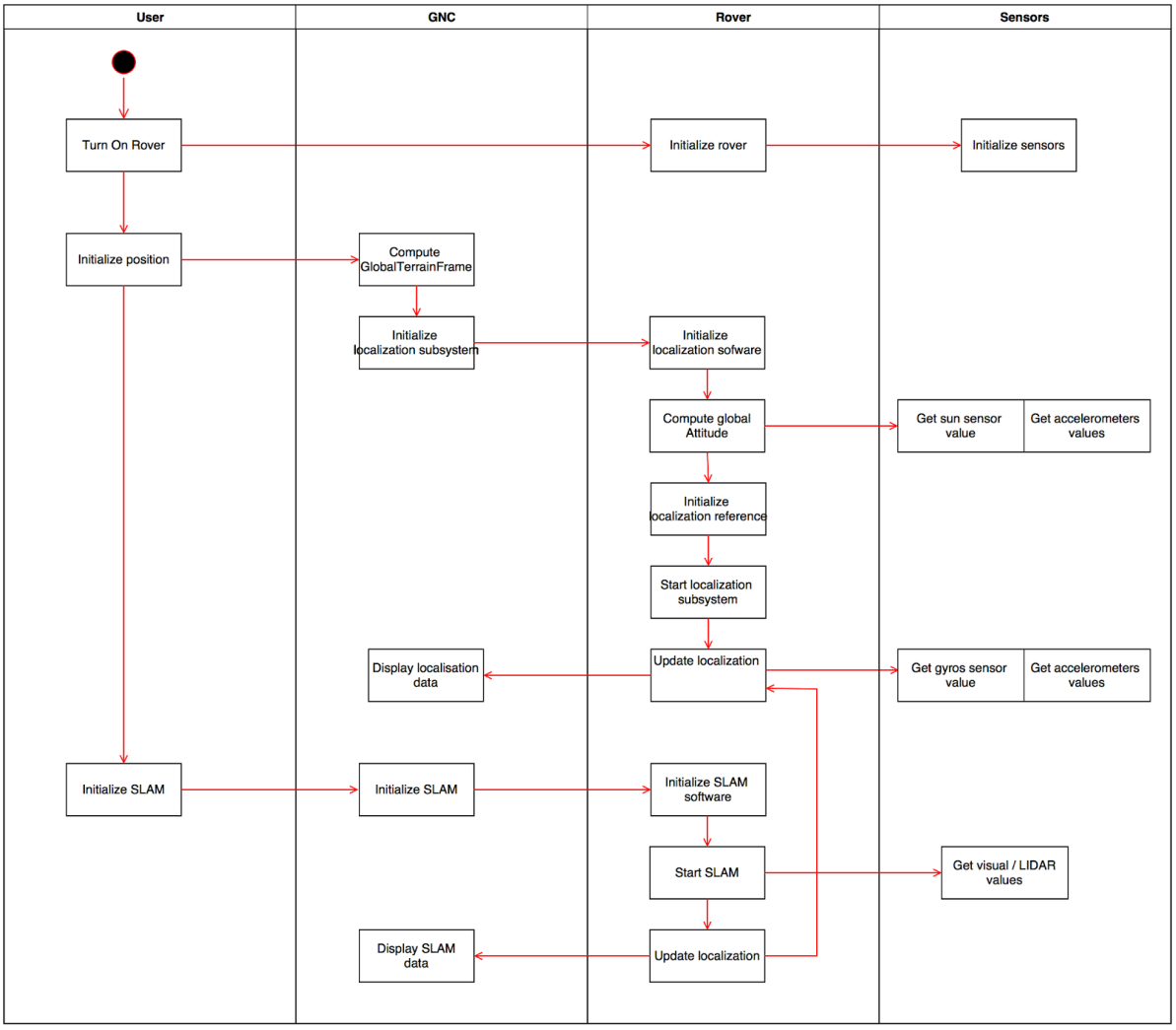


Figure 5: Activity diagram of the long range traverse scenario.

It is interesting to see that a localisation subsystem is required and it should be started very early and independently of the SLAM. The SLAM should update the localisation subsystem.

2.2.1.3 Expected results

For each of the proposed use cases, while the algorithm is running, we expect the following interactions with the user :

- related to the final product:
  - display the different frames (odometry estimate, ...),
  - display the orbital data,
  - see uncertainties (features and pose).
- related to the internal state of the DFPC:
  - display the feature map build with uncertainties,
  - browse images along the executed path : to see what happened in some points of the trajectory [optional],
  - display the DEM used to match orbital data.

Moreover, we want to evaluate the computation time, the memory footprint and the localisation accuracy (short and long range) of the various solutions, done offline by replaying data.

The metrics that will be used are the ones proposed in the Kitti Vision Benchmark[1].

2.2.1.4 Use Case 1 : Visual Odometry

This use case focuses on visual odometry only. It is the baseline solution for Mars exploration rovers. The proposed solution should support the following RGB-D sensors : stereo bench, Kinect, Xtion, TOF + Camera. It includes the following high-level functions :

- Wheel odometry,
- Visual odometry,
- Orbiter map-based localisation.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the RGB-D sensors and odometry,
- Rover navigation and locomotion control loop,
- GPS rover position measurement for ground truth determination,
- User interfaces for live monitoring and visualisation (if applicable),
- Data logging and replay functions,
- Localisation accuracy evaluation tools,

Challenges :

- sensors calibration,

[1] Andreas Geiger. 2012. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)(CVPR '12). IEEE Computer Society, Washington, DC, USA, 3354-3361.

- versatile solution (sensor support),
- delay management to provide the current estimate while taking into account delayed estimates,
- map-based localisation.

At least, this baseline solution involving a stereo bench, wheel odometry and visual odometry shall be demonstrated with an OBC representative of space missions.

2.2.1.5 Use Case 2 : Visual SLAM

This use case focuses on visual SLAM and targets next generation solutions, as it presents an increased complexity with regards to the first use case. Indeed, the visual SLAM function will create and maintain a long-term map of the explored terrain, which promises to improve localisation accuracy, especially if an area is revisited. Also, the map created during a first pass of the long traverse may be used for map-based localisation in the Return to Base scenario, to be described in the following chapters. The proposed solution should support the following RGB-D sensor : stereo bench, Kinect, Xtion, TOF + Camera. It includes following high-level functions :

- Wheel odometry,
- Visual SLAM,
- Orbiter map-based localisation.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the RGB-D sensors and odometry,
- Rover navigation and locomotion control loop,
- GPS rover position measurement for ground truth determination,
- User interfaces for live monitoring and visualisation (if applicable),
- Data logging and replay functions,
- Localisation accuracy evaluation tools.

Challenges :

- sensors calibration,
- versatile solution (sensor support),
- delay management to provide the most accurate estimate while taking into account delayed estimates,
- map-based localisation,
- satisfactory accuracy.

2.2.1.6 Use Case 3 : LiDAR SLAM

As it was said for Visual SLAM, SLAM represents a next generation of algorithms to be embedded on space vehicles, enabling the rover to build and maintain a long-term environment model of its surroundings. The map created by LiDAR SLAM can also be used

for map-based localisation in the return to base scenario. The proposed solution includes the following high-level functions:

- Wheel Odometry
- LiDAR SLAM,
- LiDAR Map-based localisation.

The functions required to integrate a full validation and demonstration chain are the same as the ones for the visual SLAM:

- Sensor acquisition for the RGB-D sensors and odometry,
- Rover navigation and locomotion control loop,
- GPS rover position measurement for ground truth determination,
- User interfaces for live monitoring and visualisation (if applicable),
- Data logging and replay functions,
- Localisation accuracy evaluation tools.

The foreseen challenges are the same as the ones for visual SLAM:

- sensor calibration,
- versatile solution (sensor support),
- map-based localisation,
- satisfactory accuracy.

2.2.1.7 Use Case 4: Fusion of LiDAR and Visual Data

Fusion of LiDAR and visual data can be achieved at two different levels:

- Directly, by incorporating LiDAR and visual data in the same data structure. With the defined DFPCs, the sole case of such fusion can be trivially done by the DEM building DFPC, which takes indifferently as inputs point clouds provided by stereovision or by a LiDAR. The fusion is here handled by the DEM building DFPC.
- Indirectly, by fusing position estimates provided by both kind of sensors. With the defined DFPCs, such a scheme can for instance be defined for the LiDAR Pose Graph SLAM DFPC, in which the initial pose at the time of the acquisition of the LiDAR scans is provided by the Visual Odometry DFPC. The fusion is here handled by the LiDAR Pose Graph SLAM DFPC.
- Note that other indirect schemes could be defined, for instance when exploiting Visual point clouds to build a DEM using positions estimates provided by LiDAR SLAM, or vice-versa. Such schemes however do not present a significant added value,

2.2.2 RI-SRC.SR-LONG-TRAVERSE

The use cases that will be retained with OG6 will only be a subpart of the use cases already identified. They will be selected depending on the set of sensors available, the level of autonomy of the platforms, and the available software interfaces.

2.3 Long Traverse DEM

The objective of the mission for the rover is to autonomously reach a target located about 1 km away and defined by its absolute coordinates, while producing 3 types of DEMs:

- The rover map: this is the map of the surroundings of the rover at each observation (One rover map is created for one observation). It is attached to the reference frame of the rover.
- The fused rover map: this is the fusion of rover maps over a given period of time. It is attached to a local reference frame (site frame).
- The total fused map: this is the fusion of all rover maps to create a complete map of the robot surrounding. It is attached to the planet's reference frame, and can be used in future journeys of the rover.

Those three types of produced maps inside InFuse are necessary for the autonomous navigation stack to analyse the rover surroundings and plan an optimal and safe trajectory towards its destination. What is "an optimal trajectory" shall not be discussed here as different criteria can be considered depending on the need of the mission.

2.3.1 RI-INFUSE-LONG-TRAVERSE-DEM

The long traverse reference implementation in InFuse will be carried out on LAAS rovers and on the SEROM CNES site. Its objective is to conduct an online demonstration and offline validation of the localisation functions proposed in InFuse.

The offline validation will consist in two stages: data collection, and data processing. The rover will execute a set of trajectories that will be defined later in the project. Each trajectory will be designed to address different complexity levels depending on the terrain (e.g. surface type, rock distribution).

For each trajectory, the following will be built:

- Rover maps and fused rover maps during trajectory execution, in simulation time,
- Fused total map at the end of trajectory execution, to create the final DEM,
- Localisation of the rover @10Hz w.r.t. the fused rover map, in simulation time.

Those three data products will be tested both with data acquired from a stereo camera bench and from a LiDAR sensor.

The consistency of the final DEMs can then be compared with the site ground truth provided by CNES SEROM.

The online demonstration will consist of two steps: first building rover maps, fused rover maps and fused total maps over a given trajectory, online, while localizing the rover at 10Hz. The fused total map produced can be then used as an orbiter map for absolute localization. The execution of the absolute localisation function will feed the locomotion system (trajectory control).

The online demonstration might include an autonomous navigation stack to validate the robustness of the function with respect to rover motions. Nevertheless, the navigation and locomotion stacks, will only use an RTK-GPS, an AHRS, and possibly wheel odometry to implement the localisation function.

2.3.2 Package diagram

The following package diagram presents a summary of software and hardware elements foreseen to be required, in addition to the validated DFPC, in order to carry out all use cases of this reference implementation. The OG3, OG4 and OG6 labels are only specified here to emphasize the perimeter of the packages. **All elements will still be provided by InFuse or OG6 for this RI.**

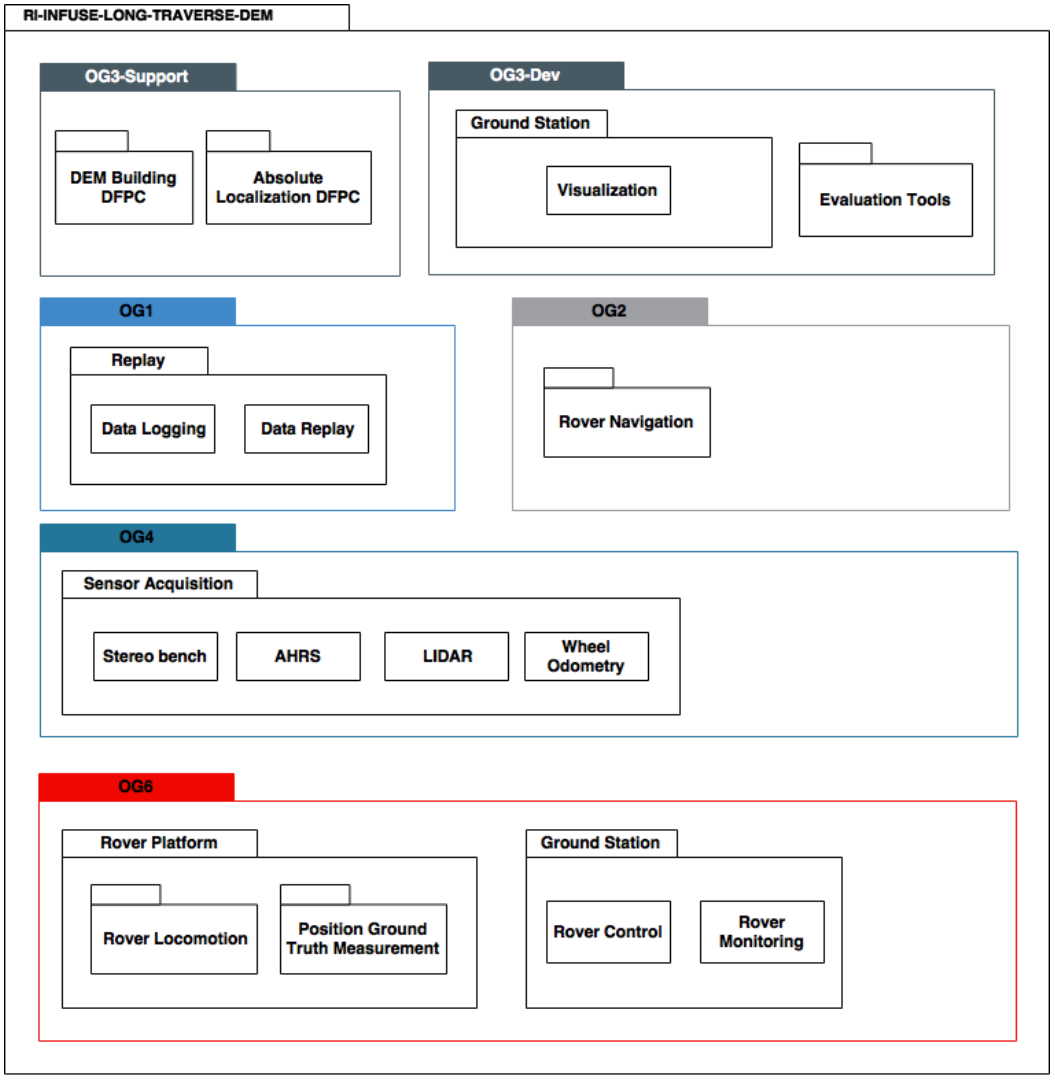


Figure 6: Package diagram of the long traverse scenario / DEM building.

2.3.2.1 Activity diagram

In this use case, for the online demonstration, the absolute localisation is performed with the following steps:

1. At the beginning of the mission, the rover position is initiated by the user
2. The user then initialises:
    a. the sensors;
    b. the rover navigation and locomotion subsystems;
    c. the localisation function:
        i. which sets an initial reference frame, loads the model of fused total map that serves as an orbital map to localize itself with, and a goal for the final pose in the target frame.
3. When the user starts the process, the rover begins its traverse:

a. Sensors data is acquired at a predefined rate;
b. The absolute localisation DFPC uses the data stream as input to estimate a relative pose;
c. Meanwhile, the user can monitor execution and receive estimated data;
d. The nominal navigation and locomotion cycle uses this pose estimate to control the movement and trajectory of the robot while avoiding hazards;

4. When the current estimated pose is reached, trajectory execution is stopped and the process finishes.

The following activity diagram gives an overview of the scenario execution.

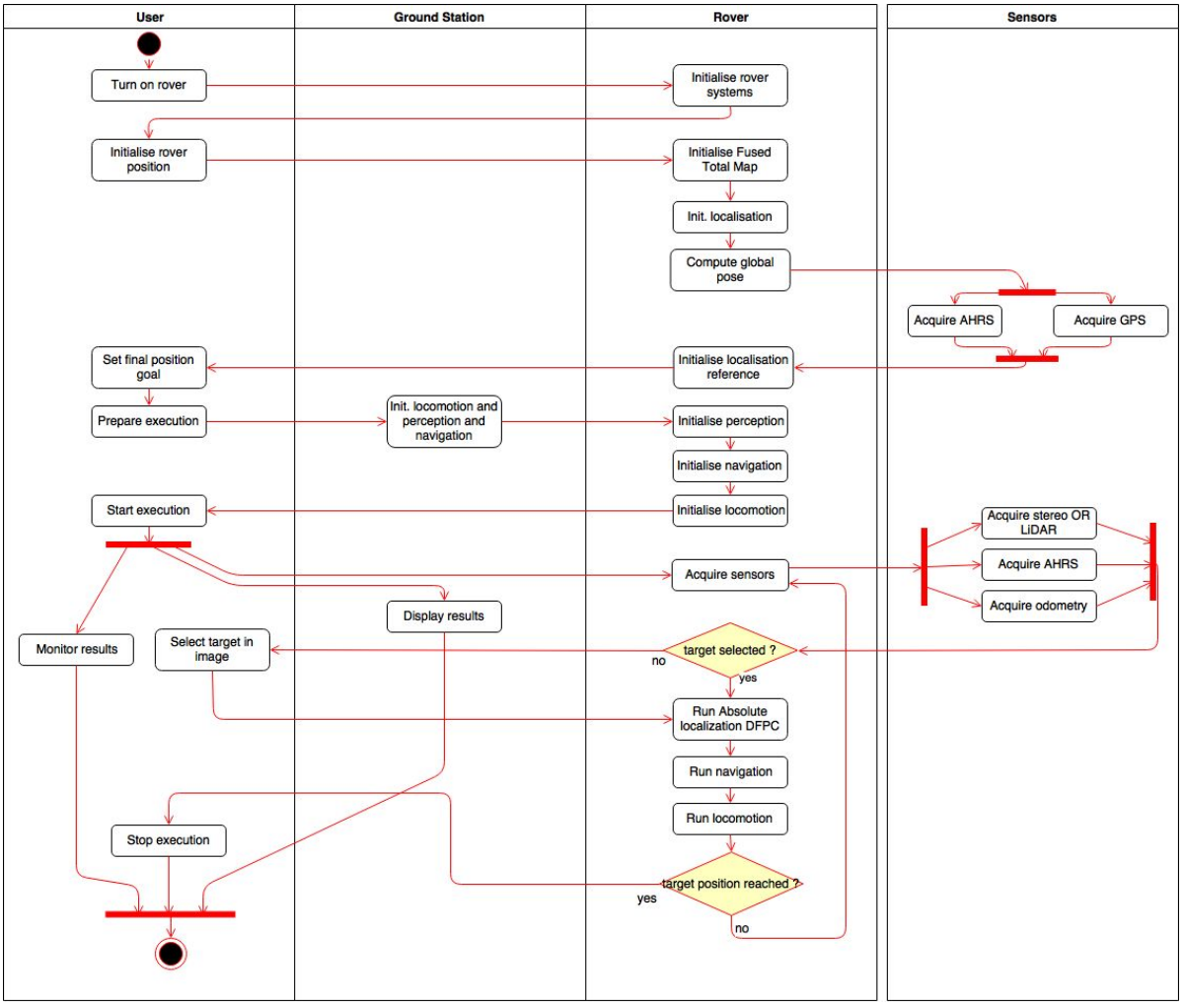


Figure 7: Activity diagram of the long traverse scenario / DEM building.

2.3.2.2 Expected results

Expected are the following, during the rover traverse:

- Related to the final product:
  - Access to the position of the rover in the fused total map frame of reference,
  - Computation and evaluation of error after full traverse,
- Related to the internal state of the DFPC:
  - Access to the rover map built in order to localize the rover,
  - Access to the point cloud provided by either the LiDAR or the stereoscopic images.

The computational resources used during the localisation process will also be evaluated.

In order to evaluate the accuracy of the DEM products, a reference DEM will be used and statistics on the errors computed.

2.3.2.3 Use Case 1: Absolute Localisation

This use case demonstrates both the capability to build DEM and to localize inside a DEM. Two subcases are possible and will be tested equally: building the DEM using LiDAR data or building the DEM using a stereoscopic image point cloud.

The proposed solution includes the following high-level functions:

- Wheel odometry
- Rover map building
- Absolute localisation

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the camera and inertial measurements,
- Rover navigation and locomotion control loop,
- GPS rover (and target) position measurement for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Localisation accuracy evaluation.

2.4 Rendezvous

The global mission objective consists in guiding the rover towards a precise position and orientation with respect to a man-made asset, e.g. the sample analysis module, for instance, to perform the transfer of a soil sample. The target is considered non-cooperative, as there is no direct communication between it and the rover. However, the proposed localisation algorithms can take advantage of the fact that its general geometry and appearance are very well known in advance.

To reach that final goal, we distinguish two mission phases, each with specific operational requirements and constraints:

- Long-range operations: from mission initialization to an approximate range of 20m. At this distance, rover sensors cannot resolve sufficient visual features on the target asset to perform a full relative pose estimation. Therefore, a simple bearing (and possibly range) tracking is performed and the result is used by the navigation system to guide the rover towards the general direction of the rover.
- Rendezvous: Once the range is short enough for the sensors to detect geometric features on the target, the rover switches to a Rendezvous operational mode, and localisation algorithms capable of giving a full relative pose are activated. The rendezvous phase lasts until the user-specified relative target pose is reached.

For a simpler validation process, these phases are decoupled in two distinct implementations : RI-INFUSE-LONG-RANGE-TRACKING and RI-INFUSE-RENDEZVOUS.

2.4.1 RI-INFUSE-LONG-RANGE-TRACKING

The long-range tracking scenario is expected to be validated on the CNES SEROM site with LAAS rovers. A previously chosen target asset (to be defined) will be placed on the site at a well-defined position. As with the RI-INFUSE-LONG-TRAVERSE-LOC implementation, both online demonstration and offline validations will be carried out on several trajectories with various levels of difficulty.

For offline execution, the baseline predefined trajectory will ensure the target remains within the field of view of the sensors as the rover approaches. Additionally, various operational variables, such as illumination direction, occlusions, and the target exiting from the field of view, can be tested to characterize the algorithm’s performance and working domain.

For online demonstration, we effectively close the loop by using the output of the long-range target tracking function as a guidance input for the navigation and locomotion functions. It will thus need to be executed in parallel with the nominal navigation loop in order to keep avoiding hazards on the way to the target.

Considering the need is only for a relatively coarse localisation at this stage, we propose a single type of reference implementation to address it: a camera-based 2D tracking.

2.4.1.1 Expected Results

While the algorithm is running, we expect the following interactions with the user :

- related to the final product:
  - display the InFuse localisation output vs GPS output on a map,
  - display the estimated target position (bearing and possibly range) on the map,
  - display uncertainties of estimated states.
- related to the internal state of the DFPC:

- provide the ability to browse images as the rendezvous progresses : to see what happened in some points of the trajectory,
- display the acquired images with an overlay of the reprojected target position.

Moreover, we aim to evaluate the computation time, the memory footprint and the final localisation accuracy.

The localisation accuracy will be evaluated with respect to the distance separating the target from the chaser. This evaluation will be completed by true / false detection analysis by using ROC curves.

2.4.1.2 Package Diagram

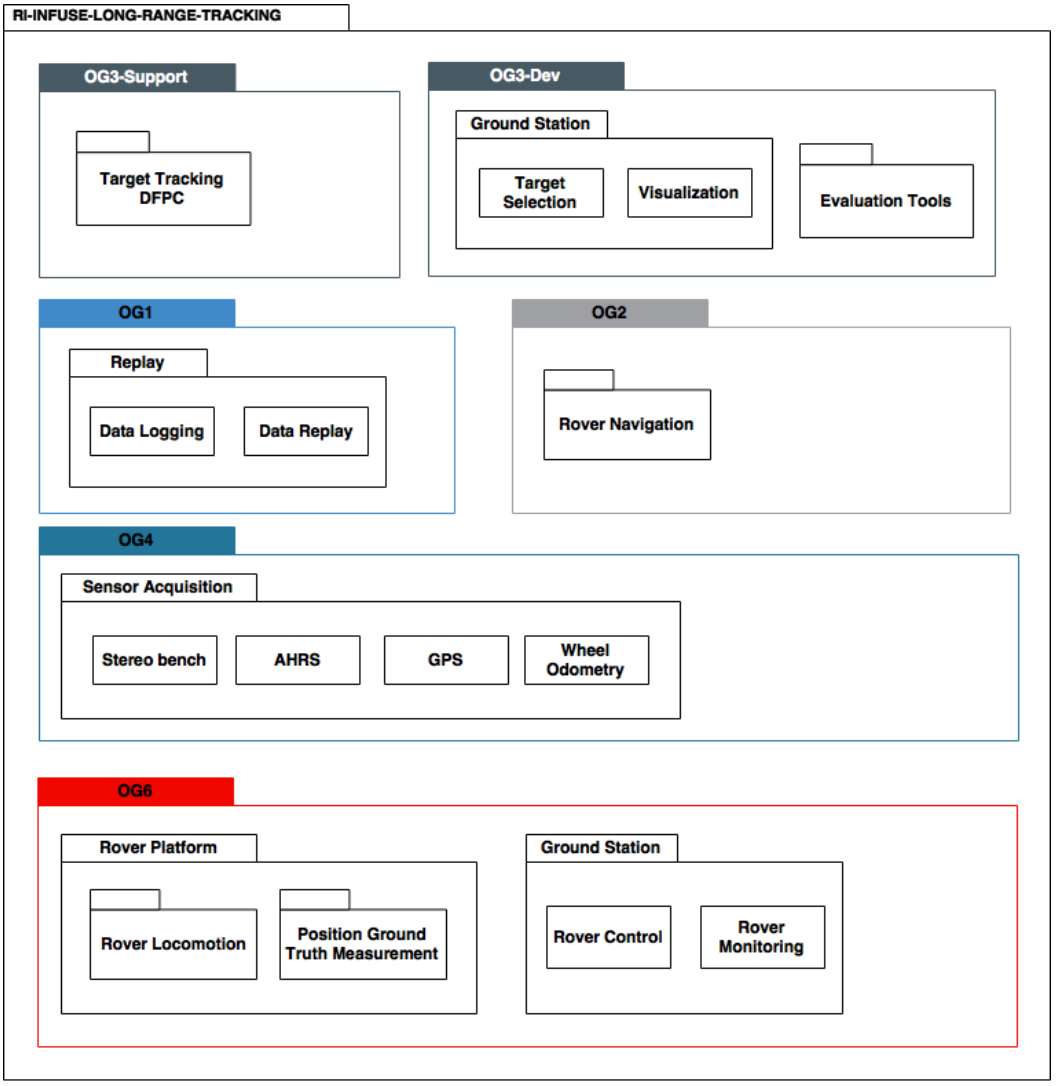


Figure 8: Package diagram of the long range tracking scenario.

The following package diagram presents a summary of software and hardware elements foreseen to be required, in addition to the validated DFPC, in order to carry out all use cases of this reference implementation. The OG3, OG4 and OG6 labels are only specified here to emphasize the perimeter of the packages. **All elements will still be provided by InFuse for this RI.**

2.4.1.3 Activity Diagram

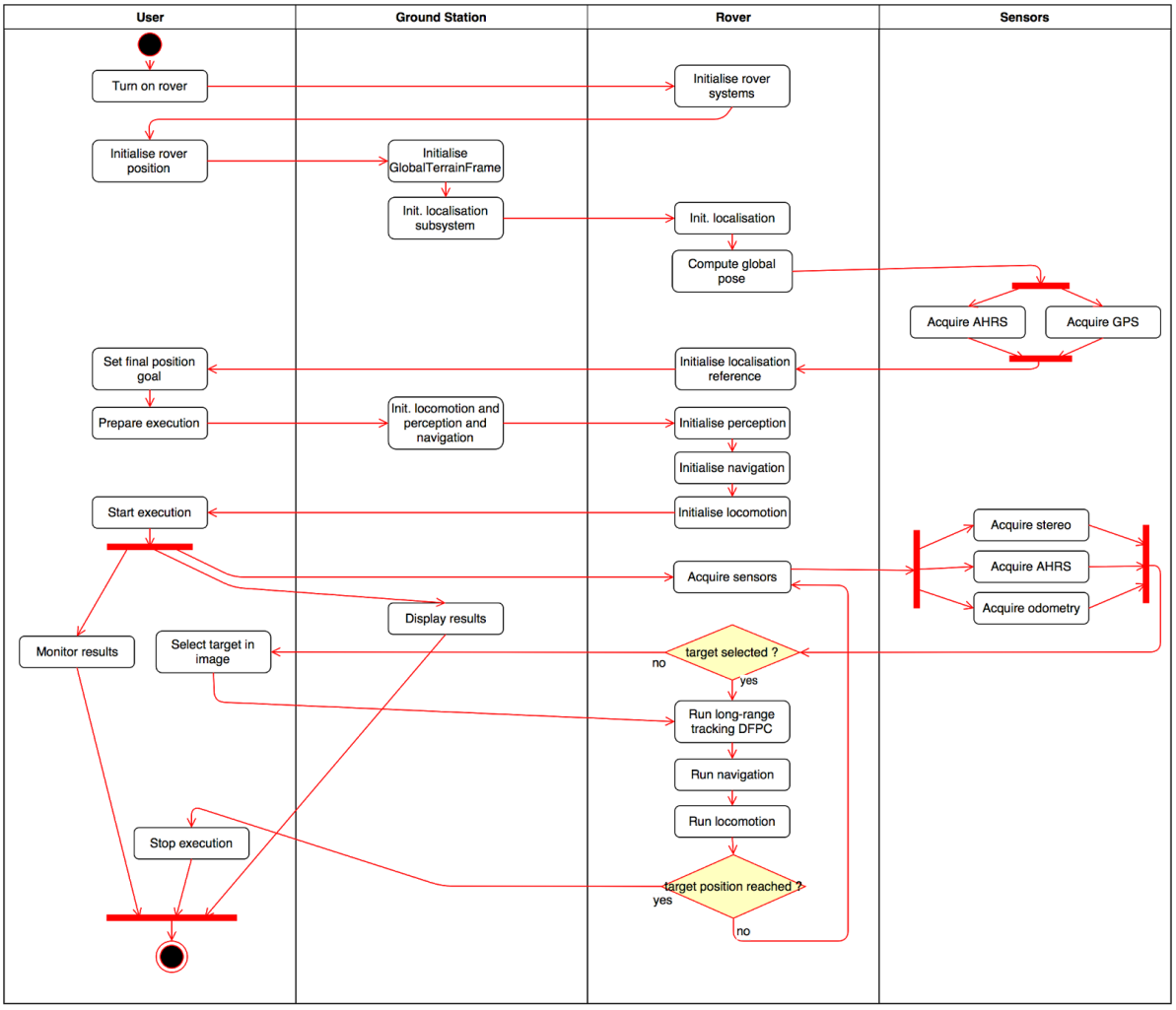


Figure 9: Activity diagram of the long range tracking scenario.

2.4.1.4 Use Case 1: Long-range Bearing-only Target Tracking

Since, at the considered range, it is only realistic to accurately estimate the target bearing, we focus this use case on a simple 2D camera. The target, which is assumed to be static, is first detected in the image by dense matching, then this bearing measurement is fed into a classic filtering function which uses a rover motion model to ensure a continuous and robust tracking. Additionally, the tracking filter would benefit from having access to inertial measurements of the rover, using it either for a simple state prediction, or as a

measurement for state correction. Finally, user interaction is needed to initialize the system by designating the target to be tracked in an acquired image.

The core data fusion implementation thus requires the following high-level functions:

- Dense image matcher,
- Tracking filter.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the camera and inertial measurements,
- Rover navigation and locomotion control loop,
- GPS rover (and target) position measurement for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Localisation accuracy evaluation.

2.4.2 RI-INFUSE-RENDEZVOUS

As a precondition detailed in [InFuse_D4.1], the rover is assumed, for this second phase of the mission, to be already positioned in the vicinity of the target asset.

This phase will also be carried out with the LAAS rovers, on the CNES SEROM site, with a target to be defined. Once again, several trajectories around the target asset, starting from the end point of the RI-INFUSE-LONG-RANGE-TRACKING and ending in a position and orientation suitable for a close-range sample transfer, will be executed manually while the sensors acquire and log the data for offline validation.

In an online demonstration, the user will specify position and orientation goals for the rover to reach with regards to the asset and the output of the localisation function will be used by the navigation function to reach this goal. It will thus need to be executed in parallel with the nominal navigation loop in order to keep avoiding hazards on the way to the target.

At medium to close range, we can choose relative localisation strategies that make use of richer RGB-D or point cloud data, such as a stereo bench, a structure-from-motion arrangement, a ToF camera or a 3D LiDAR, and harness our prior knowledge of the target's geometry. We thus propose four potentially complementary use cases to achieve this task:

- RGB-D Model-based detection tracking, which detects and tracks the target using RGB-D data and combined geometric and appearance-based models;
- LiDAR Model-based tracking, which detects and tracks with pure point cloud data based on a point cloud model of the target.
- Structure-from-motion reconstructions of the target based only on stereo vision cameras and the movement of the rover relative to the target,
- Haptic Scanning, for mapping and reconstruction of the target body.

2.4.2.1 Expected Results

While the algorithm is running, we expect the following interactions with the user :

- related to the final product:
  - display the InFuse localisation output compared to GPS output on a map,
  - the pose of the rover in the target reference frame and the confirmation that it accurately reaches the set goal,
  - display uncertainties of estimated states,
- related to the internal state of the DFPC:
  - provide the ability to browse images as the rendezvous progresses : to see what happened in some points of the trajectory,
  - display the acquired images with an overlay of the reprojected estimated model pose reprojected.

Moreover, we aim to evaluate the computation time, the memory footprint and the final localisation accuracy of the various solutions.

The localisation accuracy will be evaluated with respect to the distance separating the target from the chaser.

2.4.2.2 Package Diagram

The following package diagram presents a summary of software and hardware elements foreseen to be required, in addition to the validated DFPC, in order to carry out all use cases of this reference implementation. The OG3, OG4 and OG6 labels are only specified here to emphasize the perimeter of the packages. **All elements will still be provided by InFuse or OG6 for this RI.**

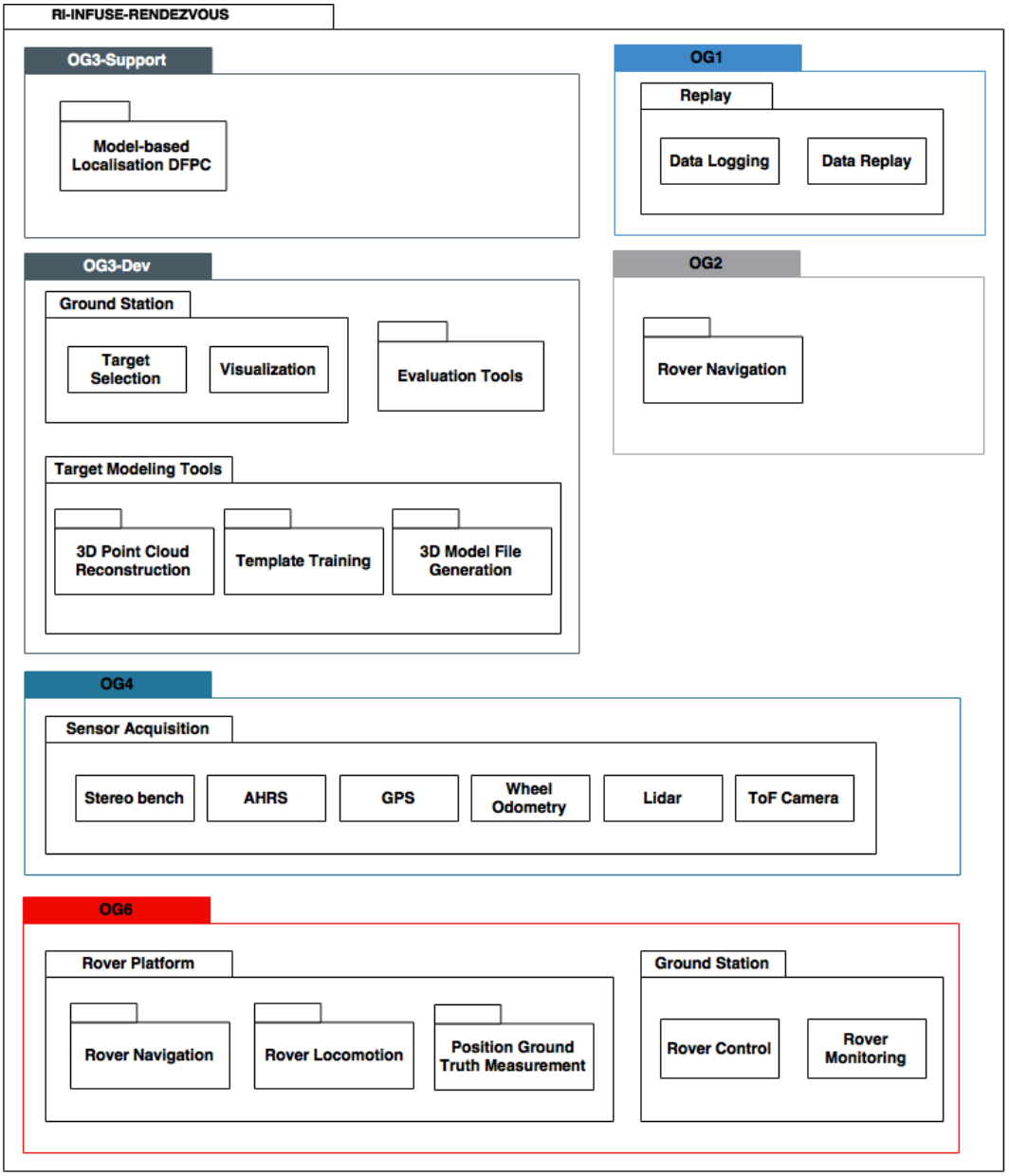


Figure 10: Package diagram of the rendez-vous scenario.

2.4.2.3 Activity Diagram

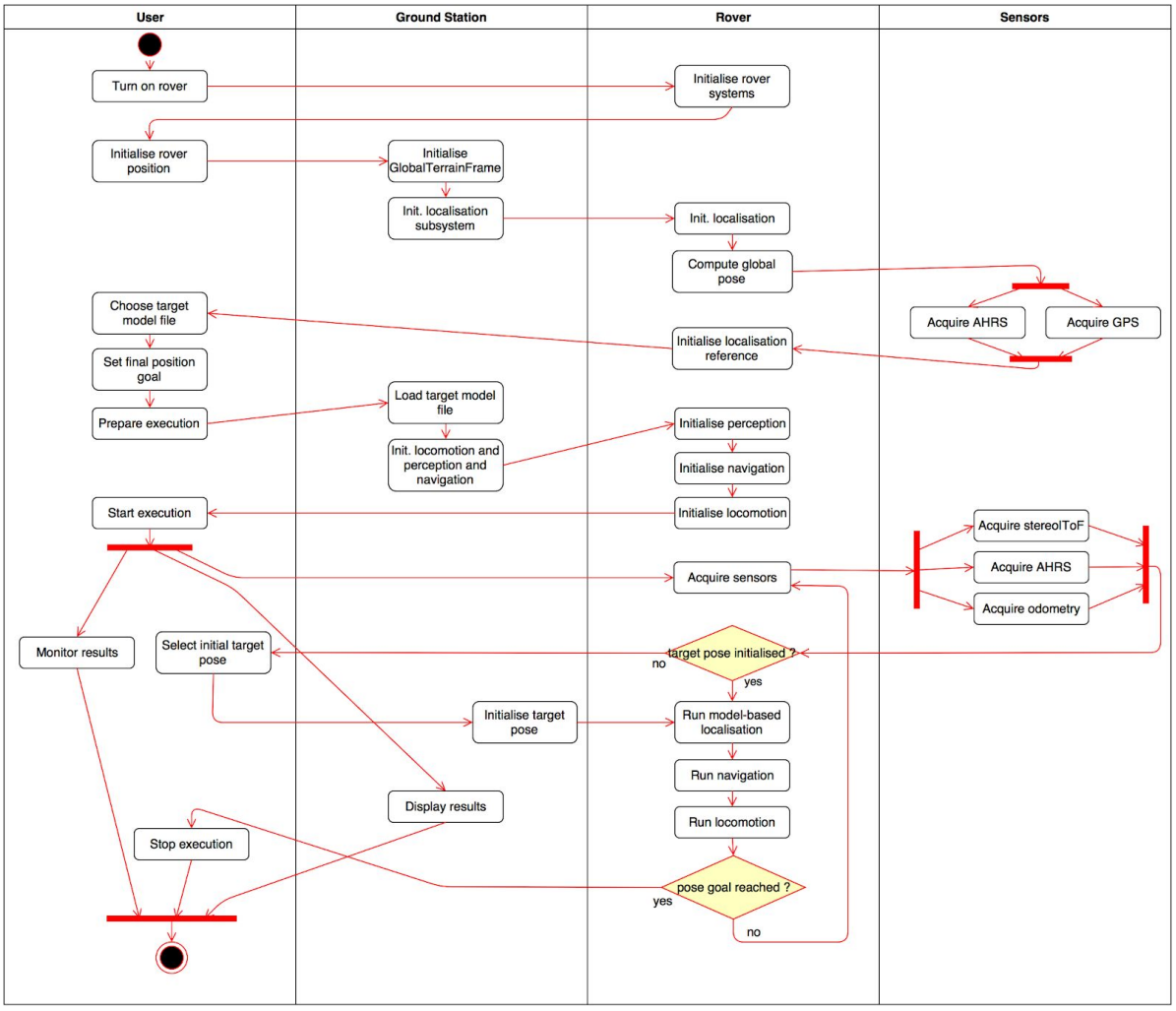


Figure 11: Activity diagram of the rendez-vous scenario.

In this use case, for the online demonstration, the rendezvous is performed in the following steps (The activity diagram gives an overview of the scenario execution):

1. At the beginning of the mission, the rover is located in the vicinity of the target, which is visible in the field of view of the camera;
2. The user initialises:
    a. the sensors;
    b. the rover navigation and locomotion subsystems;
    c. the localisation function:
        i. which sets an initial reference frame, loads the model of the target, a goal for the final pose in the target frame.
3. When the user starts the process, the rover begins its rendezvous operation.
    a. Sensors data is acquired at a predefined rate;

b. The model-based localisation DFPC uses the data stream as input to estimate a relative pose;
c. Meanwhile, the user can monitor execution and receive estimated data;
d. The nominal navigation and locomotion cycle uses this pose estimate to control the movement and trajectory of the robot while avoiding hazards;

4. When the current estimated pose is reached, trajectory execution is stopped and the process finishes.

2.4.2.4 Use Case 1: RGB-D Model-based Detection and Tracking

This use case focuses on localisation with regards a model of the target which combines visual features and geometric primitives, such as a CAD model. At close range, the 2D camera is able to detect and track visual features on the target body. To greatly enhance tracking performance and robustness, the algorithm can consider a user-provided 3D geometry (made of simple geometric primitives such as planes and cylinders) into its rigid-body motion model. However, the filter still needs an initialisation step. In order to reduce operator intervention, we propose to perform initialisation with a target detector. The detector uses an offline-trained template of the target and RGB-D measurements to provide a coarse first estimate of its pose to the tracking function.

The core data fusion implementation thus requires the following high-level functions:

- Model-based target detector,
- Visual feature detector and matcher,
- Model-based 3D tracking filter.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Template training tools based on simulated or real image data,
- Tools for model geometry definition and file generation,
- Sensor acquisition for the camera and inertial measurements,
- Rover navigation and locomotion control loop,
- GPS rover (and target) position measurement for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Localisation accuracy evaluation.

2.4.2.5 Use Case 2: Dense Point Cloud Model-based Localisation

This use case focuses on a simple implementation of a LiDAR-backed model-based localisation scheme. In this case, the target model consists of a reconstructed point cloud with a density high enough to allow for subsampling. We propose to perform a dense matching and rigid-body optimization between the acquired point cloud and the user-provided model. To enable a continuous tracking, pose filtering with a motion model is also included.

The core data fusion implementation thus requires the following high-level functions:

- Point cloud matcher,
- 3D pose filter.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the LiDAR and inertial measurements,
- Tools for target point cloud model reconstruction,
- Rover navigation and locomotion control loop,
- GPS rover (and target) position measurement for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Localisation accuracy evaluation.

Some challenges about this approach are foreseen, namely the issue of potentially large resolution differences between the model and acquired point clouds.

2.4.2.6 Use Case 3: 3D Feature Model-based Localisation

This use case focuses on a similar technique of matching and tracking between point clouds, with the particularity of working at the level of 3D features detected within them. Again, in this case, the target model consists of a reconstructed point cloud with a density high enough to allow for subsampling. We propose to first perform 3D keypoint detection and descriptor extraction in model and measured point clouds, then match these keypoints together to extract a rigid body transformation. The obtained measurement can then be fed to a filtering function to enable a continuous target pose estimation.

This use case can be implemented to support point clouds generated from stereo cameras, ToF cameras or 3D LiDAR sensors.

The core data fusion implementation thus requires the following high-level functions:

- 3D keypoint detector and descriptor extractor,
- Descriptor matcher,
- 3D pose filter.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the inertial and LiDAR or camera measurements,
- Tools for target point cloud model reconstruction,
- Rover navigation and locomotion control loop,
- GPS rover (and target) position measurement for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,

- Localisation accuracy evaluation,

Some challenges about this approach are foreseen, namely the issue of potentially large resolution differences between the model and acquired point clouds.

2.4.2.7 Use Case 4: 3D Reconstruction from Structure-From-Motion

This use case accomplishes detection of 3D features within a point cloud as in Use Case 3. However, the point cloud is generated using structure-from-motion methods with stereo cameras, which allows large-area point clouds to be reconstructed from multiple angles over time, using the relative motion of the rover and target to build a full model of all sides of a target from multiple views. While fusing point clouds generated from ToF cameras or 3D LiDAR sensors is possible, we focus on visual-only construction in this use case. Also, to identify models within the scene that is reconstructed, we use SHOT descriptors and Hough voting in place of ICP to provide additional control of computational efficiency and the resolution of keypoints by means of changing the descriptor radius and number of keypoints.

The core data fusion implementation thus requires the following high-level functions:

- 2D feature detector and descriptor extractor,
- 2D descriptor matcher (FLANN)
- Fundamental matrix calculation from triangulation of points
- Perspective-and-Point solver (RANSAC)
- 3D keypoint detector and descriptor extractor (SHOT),
- 3D descriptor matcher (RANSAC),
- 3D pose filter.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the inertial and camera measurements,
- Tools for target point cloud model reconstruction,
- Rover navigation and locomotion control loop,
- GPS rover (and target) position measurement for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Localisation accuracy evaluation,

This approach can also be evaluated for mitigating potentially large resolution differences between the model and acquired point clouds due to the tuneable SHOT descriptors.

2.4.2.8 Use Case 5: 3D reconstruction and mapping with Haptic Scanning

Haptic Scanning basely consists of taking benefit of information dealing with contacts established between a robotic manipulator and a target. In the planetary scenarios of InFuse, haptic scanning is identified as an opportunistic strategy to collect information about the environment: it is not foreseen to carry out dedicated haptic scanning actions or

series of actions (e.g. following a certain pattern of scanning along a structure), but rather to make use of information available while manipulation actions are being carried out, for other purposes. The assumption is that, when a contact takes place between the manipulator and a target (artificial or natural structure), a force is measured and is available as a piece of information. We propose to collect and integrate such contact information into a model, that will contain sparse, but accurate information on target points positioning (through information on encoders / kinematic chain of the manipulator) and associated force information. Such a model, that will translate in an augmented mesh (considering force information), may potentially be fused with other 3D models of the environment.

Besides an opportunistic usage, it could be envisaged to trigger dedicated haptic scanning actions (i.e. purposely establishing a contact) to disambiguate depth information in certain locations where other sensors may have been impaired, for various reasons (e.g. visual cameras may be dazzled by sun or reflect on shiny surface, Lidars may be misled by transparent or translucent materials, etc.). This capability may not be a fundamental one, but may occasionally be relevant, at limited cost. Similarly, in case of a failure with a primary sensor (Lidar, ToF camera, stereo…), pro-active haptic scanning may help ensuring that a basic (sparse) model of the environment may nevertheless be built - which may be useful to take decision and plan paths in a degraded mode, from OG2 / ERGO.

Note that we do not intend in InFuse to develop and provide guidance/control/servoing capability for a manipulator setup: the haptic scanning approach is considered a data fusion capability, from the InFuse DFPC point of view. Only opportunistic haptic scanning is therefore encompassed, not pro-active haptic scanning.

The core data fusion implementation thus requires the following high-level functions:

- Mesh data structure allocators :
    - allowing to populate for each position an associated normal,
    - allowing to query for each 3d point its associated normal if available,
    - 3D Distance query functions
    - Triangulation based on measured points

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Tools for target point cloud model reconstruction and visualisation,
- End effector force & positions generations

2.4.3 RI-SRC.SR-RENDEZVOUS

The use cases that will be retained with OG6 will only be a subpart of the use cases already identified. They will be selected depending on the set of sensors available, the level of autonomy of the platforms, and the available software interfaces.

2.5 Return to Base

The objective of the mission for the rover is to autonomously execute a trajectory that has been executed beforehand (*e.g.* after a long traverse, or after having fetched a sample, or performed a scientific analysis), but in the opposite direction. The localisation system can thus benefit from the use of maps created during the first pass and perform a relatively simpler pose estimation within this map instead of having to create and maintain a new one.

2.5.1 RI-INFUSE-RETURN-TO-BASE

The Return to Base implementation of InFuse will be carried out on LAAS rovers and on the SEROM CNES site. As with the RI-INFUSE-LONG-TRAVERSE-LOC implementation, an online demonstration and an offline validation of localisation functions proposed in InFuse will be conducted.

The offline validation will consist in two stages, first data collection, next data exploitation. Since this implementation is heavily dependent on the data acquired and produced during the RI-INFUSE-LONG-TRAVERSE-LOC implementation, we propose to carry out RI-INFUSE-RETURN-TO-BASE in tandem with this scenario. In this case, the data collection phase shall be executed immediately after finishing the Long Traverse, by having the rover perform the reverse trajectory back to the starting point. It may also be relevant to perform the return trajectory after some time delay, in order to take into account the changes in illumination conditions that could occur during a typical mission.

The online demonstration could also be executed immediately after the online demonstration RI-INFUSE-LONG-TRAVERSE-LOC, and will consist in replacing the Long Traverse localisation function with the map-based localisation function during the reverse trajectory, feeding the locomotion system, and effectively closing the loop.

In both cases, the LONG-TRAVERSE-LOC implementation must provide a way to save the SLAM map for reuse in further RETURN-TO-BASE scenarios.

We propose 2 use cases in order to answer to map-based localisation needs in a Return to Base scenario. Naturally, they follow the 2 types of localisation maps created in RI-INFUSE-LONG-TRAVERSE-LOC:

- Visual Map-based Localisation : this implementation uses the SLAM map based on visual features extracted from RGB-D data,
- Point Cloud Map-based Localisation: this implementation is designed for SLAM maps created using point cloud data.

2.5.1.1 Expected Results

For each of the proposed use cases, while the scenario is underway, we expect the following interactions with the user:

- related to the final product:
    - display the INFUSE localisation output vs GPS output on a map

- display uncertainties (rover pose).

- related to the internal state of the DFPC:
  - browse images along the executed path : to see what happened in some points of the trajectory [optional].

Moreover, we want to evaluate the computation time, the memory footprint and the localisation accuracy (short and long range) of the various solutions, done offline by replaying data. We will use the same metrics as for the long range navigation localisation scenario.

2.5.1.2 Package Diagram

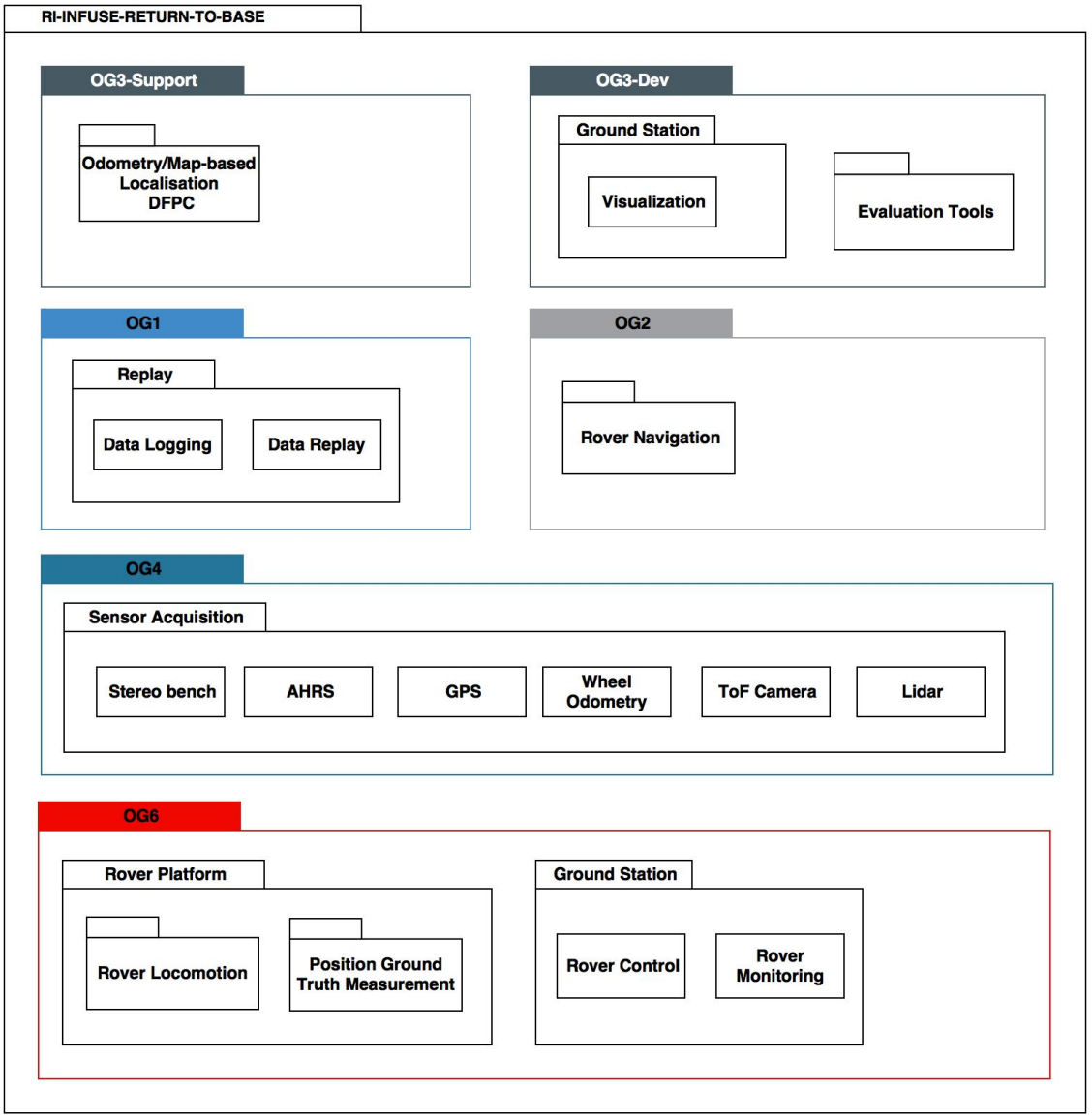


Figure 12: Package diagram of the return to base scenario.

This package diagram presents a summary of software and hardware elements foreseen to be required, in addition to the validated DFPC, in order to carry out all use cases of this reference implementation. The OG3, OG4 and OG6 labels are only specified here to emphasize the perimeter of the packages. **All elements will still be provided by InFuse or OG6 for this RI.**

2.5.1.3 Activity Diagram

For the online demonstration, the scenario is performed in the following steps:

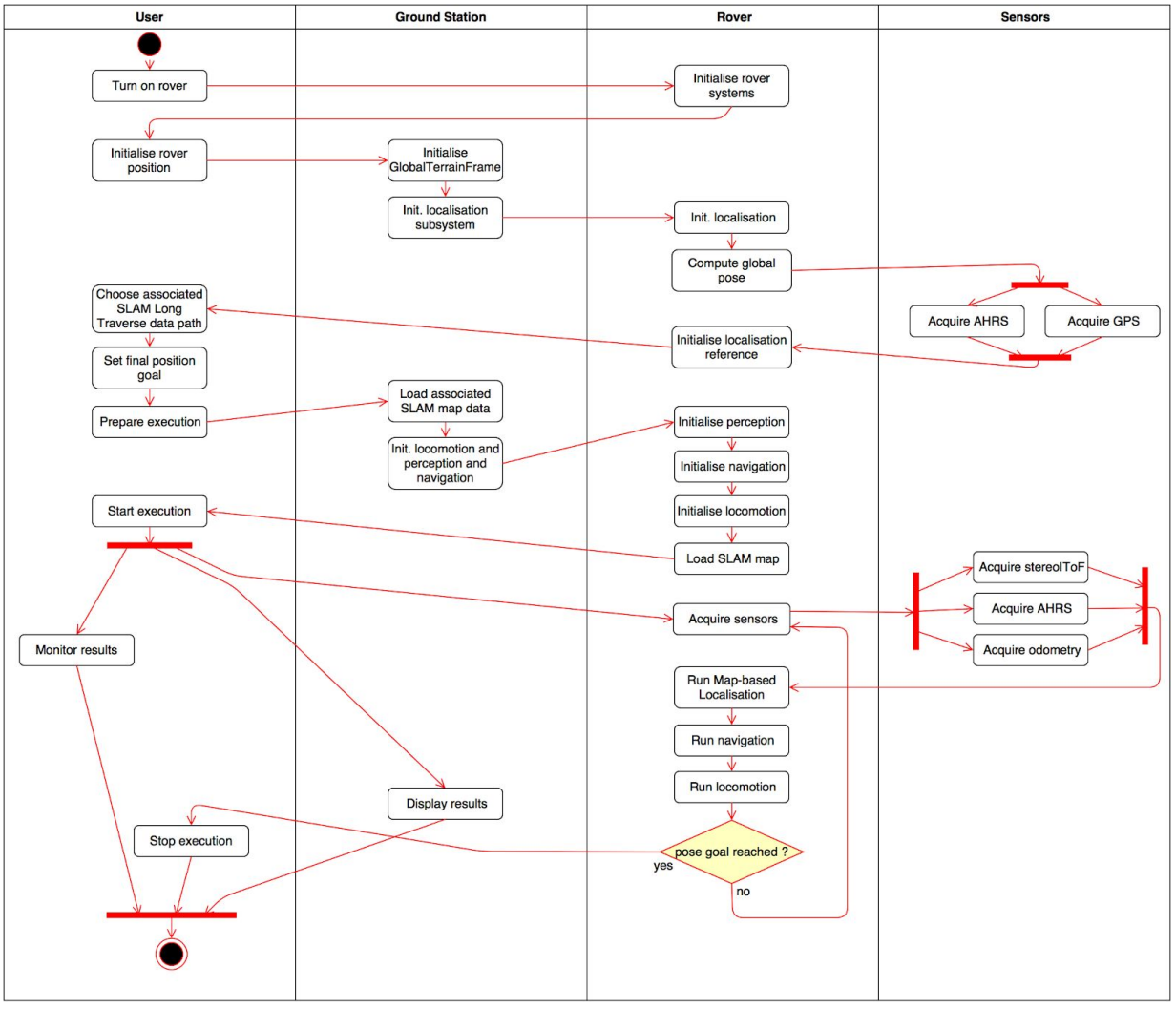


Figure 13: Activity diagram of the return to base scenario.

2.5.1.4 Use Case 1 : Visual Map-Based Localisation

This use case focuses on localisation with regards to a previously-built SLAM map using visual features, as detailed in Use Case 2 : Visual SLAM. The basic odometry process is thus augmented with a local map tracking process for a refinement of the estimated

position with regards to the map. This process detects keypoints in the acquired image, and matches them with known keypoints from the map. In order to select relevant keypoints in the map, two techniques are available: if tracking has been successful in the previous frame, we simply select keypoints in the closest keyframe. Otherwise, a bag-of-words matching technique attempts to find the most likely keyframe to extract keypoints from. After an initial pose estimation, the tracking function extracts keyframes in the neighbouring frames and performs a full rover pose optimization.

As with the Visual SLAM use case, this solution should support the following sensors: stereo bench, Kinect, Xtion, and other ToF + RGB camera setups.

This use case includes the following high level-functions:

- Visual feature detection and extraction,
- Wheel odometry,
- Relocalisation: to provide a first pose estimate,
- Tracking: to provide an optimized rover pose.

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the RGB-D sensors and odometry,
- Rover navigation and locomotion control loop,
- GPS rover position measurement for ground truth determination,
- User interfaces for live monitoring and visualisation (if applicable),
- Data logging and replay functions,
- Localisation accuracy evaluation tools.

Notably, data logging and replay functions must not only be able to save and restore sensor data, but also the complete SLAM map created during LONG-TRAVERSE-LOC.

2.5.1.5 Use Case 2 : Point Cloud Map-Based Localisation

In this case we focus on localisation with regards to a previously built map using point clouds, such as in Use Case 2 : LiDAR SLAM. This use case is symmetric with the visual map-based localisation: it improves the odometry localisation by matching LiDAR observation with a previously built LiDAR map, thanks to the LiDAR-SLAM.

This use case includes the following high level-functions:

- Point cloud matching
- Wheel odometry
- Pose estimation

Additional ad-hoc functions are also required to integrate a full validation and demonstration chain:

- Sensor acquisition for the RGB-D sensors and odometry,

- Rover navigation and locomotion control loop,
- GPS rover position measurement for ground truth determination,
- User interfaces for live monitoring and visualisation (if applicable),
- Data logging and replay functions,
- Localisation accuracy evaluation tools.

Notably, data logging and replay functions must not only be able to save and restore sensor data, but also the complete SLAM map created during LONG-TRAVERSE-LOC.

2.5.2 RI-SRC.SR-RETURN-TO-BASE

The use cases that will be retained with OG6 will only be a subpart of the use cases already identified. They will be selected depending on the set of sensors available, the level of autonomy of the platforms, and the available software interfaces.

2.6 Requirements

Following the objectives, platforms and scenarios described in this document, a list of requirements applicable to the Planetary Track reference implementation has been produced. This list will serve as a tool to evaluate how our final implementation responds to the expressed needs.

See associated requirements spreadsheet : InFuse_5.2_APPENDIX_REQUIREMENTS.

3 System Modeling

This chapter addresses the modeling of InFuse, integrated in our validation and demonstration scenarios. It somehow describes the system architecture including InFuse EGSE as well as Facilitators, ESROCOS and ERGO products.

The objective is thus to identify all components of the system, their interfaces and relationships. For this, the architecture corresponding to RI-INFUSE and RI-SRC.SR scenarios are presented in dedicated sections.

The system is defined as all hardware and software parts that together allows to implement scenario described in Chapter 2. Depending on the scenario that will be demonstrated, all parts of the overall system might not be required.

To provide a comprehensive description, we adopt a top down approach. We start by presenting generic components composing a robotics system, then we list all components that could be used, and finally we explain how they will be assembled.

3.1 Robotics System

Usually a robotics system is composed of robots, sensors, actuators, on-board computers, environments, ground stations, communication links and software. In this document we adopt the following definitions which might change in other contexts.

- robot system : the robot in our case is an exploration rover including all actuators, sensors, controllers, its on-board computer, batteries and software which together provide a basic mobile platform that can be controlled in speed and direction,
- sensor system : it includes all sensors that are not included in the robot and which are required to improve its autonomy, to perform science or to interact with the environment,
- actuator system : it include all actuators that are not included in the robot and that are required to improve its autonomy, to perform science or to interact with the environment,
- on-board computer system : it includes all the processing units except the ones included in the robots and dedicated to their operation,
- environment system : it includes the environment where the robot will evolve,
- ground station system : it is the set of computers that allow the end-user to interact with the rover,
- communication link system : it consists of all communications links that are required by ground stations, on-board computers, sensors, actuators, microcontrollers to communicate and exchange data,
- embedded software system : it is composed of all the necessary software. It could be decomposed at a finer functional level like localisation software, navigation software, etc.

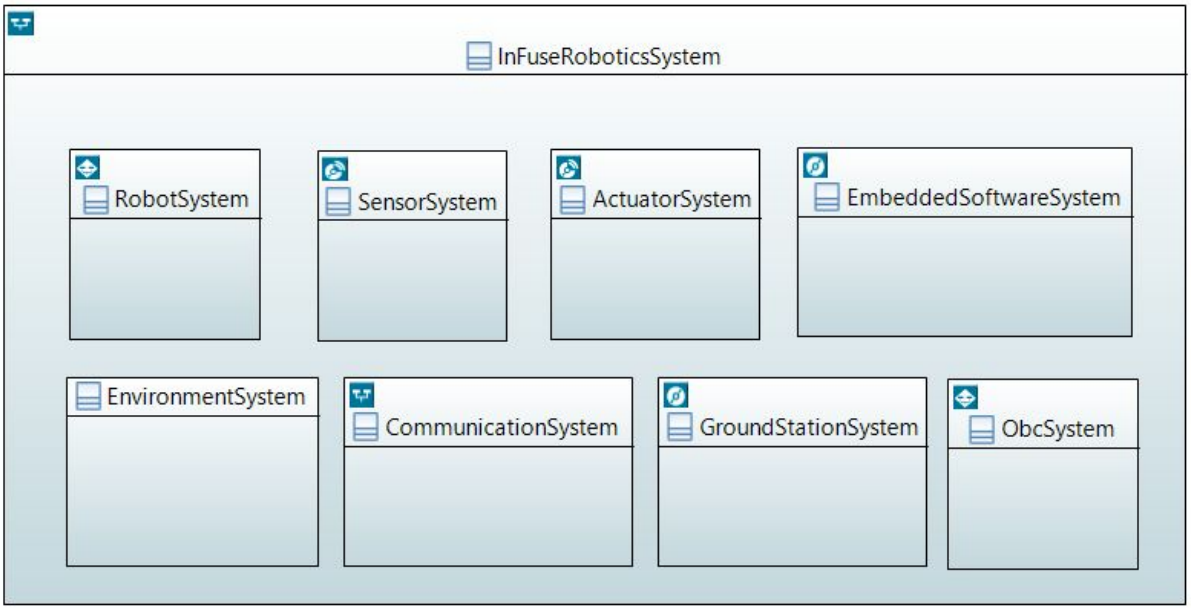


Figure 14: Components within a generic robotics system

3.2 RI-INFUSE

This section describes the system architecture retained to implement RI-INFUSE scenarios.

3.2.1 System Components

The following system components are part of the InFuse robotics system :

Robot system : The robot system is implemented with two rovers, namely Mana and Minnie provided by CNRS/LAAS. They should have a mast to mount navigation cameras.

Sensor system : The sensor suite required by RI-INFUSE scenarios includes :

- two RTK-GPS (one per rover),
- two AHRS,
- three stereo benches (one for the navigation on top of a mast NavCam), one for visual odometry or hazard avoidance (FrontCam), and one for back hazard avoidance or return to base scenario (RearCam),
- one LIDAR for navigation and localisation.

Actuator system : The actuator suite required by RI-INFUSE scenario includes :

- a pan-tilt turret for autonomous navigation and target tracking (rendez-vous operation),
- a robotic arm for sample exchange.

On-board computer system: We want to cover the various categories of processing units that could be available for space systems from now to 2050 and more, thus we propose :

- LEON 4 FT: this processor is available for space applications,
- ARM CORTEX-A9: this type of processor architecture should soon be available,
- x86: it can be considered representative of future processors,
- Some functions could also be implemented on a FPGA.

Environment system: As presented in the introduction of validation and demonstration scenario, InFuse will work in following environments :

- SEROM test field in Toulouse,
- DLR PLE test field in Munich,
- Erfoud test field in Morocco.

Ground station system: For now, one ground station per rover is envisaged. They shall be transportable. Each also includes its own software and user interfaces.

Communication link system: For now, the following communication links have been identified:

- Wireless communication link between rovers and the ground station,
- Wired communication link between on-board computers,

Embedded software system : The overall software system can be decomposed in 7 sub-systems.

- Locomotion system : It is responsible for trajectory execution,
- Localisation system : It is responsible for providing the robot pose in real-time or in the past (short term and long term), and the relative pose wrt a target,
- Perception system : It is responsible of acquisitions, turret management (used to make active acquisitions like a panorama or target tracking) and some low-level processing like stereo-vision,
- Hazard avoidance system : It is responsible for raising alerts when unexpected objects are detected in the close vicinity of rover,
- Autonomous navigation system : It is responsible for building / fusing dem and navigation maps, finding and executing a safe trajectory towards a given goal.
- Autonomous controller system : It is responsible for managing requests from the ground station and all kinds of events that can arise : errors, hazard detection, etc.
- Rover state data product manager : It is responsible of tracking the rover configuration and state : actuator state and status, sensor models, current localisation, …

3.2.2 System Architecture

In order to validate and demonstrate InFuse, DFPCs will be integrated in an existing Robotics middleware that will allow to use either simulation tools or real rovers. For now, we focus on the identification of relevant interfaces between systems. Then, depending on the deployment strategy, the appropriate robotics middleware will be chosen. In order to simplify the presentation of the architecture, we focus on some key functions in a multiview approach. We focus on image processing solutions and the same logic can be applied to process LIDAR data.

3.2.2.1 Visual localisation view

This diagram defines interfaces with the localisation and perception systems in the scenario of Long Traverse Localisation.

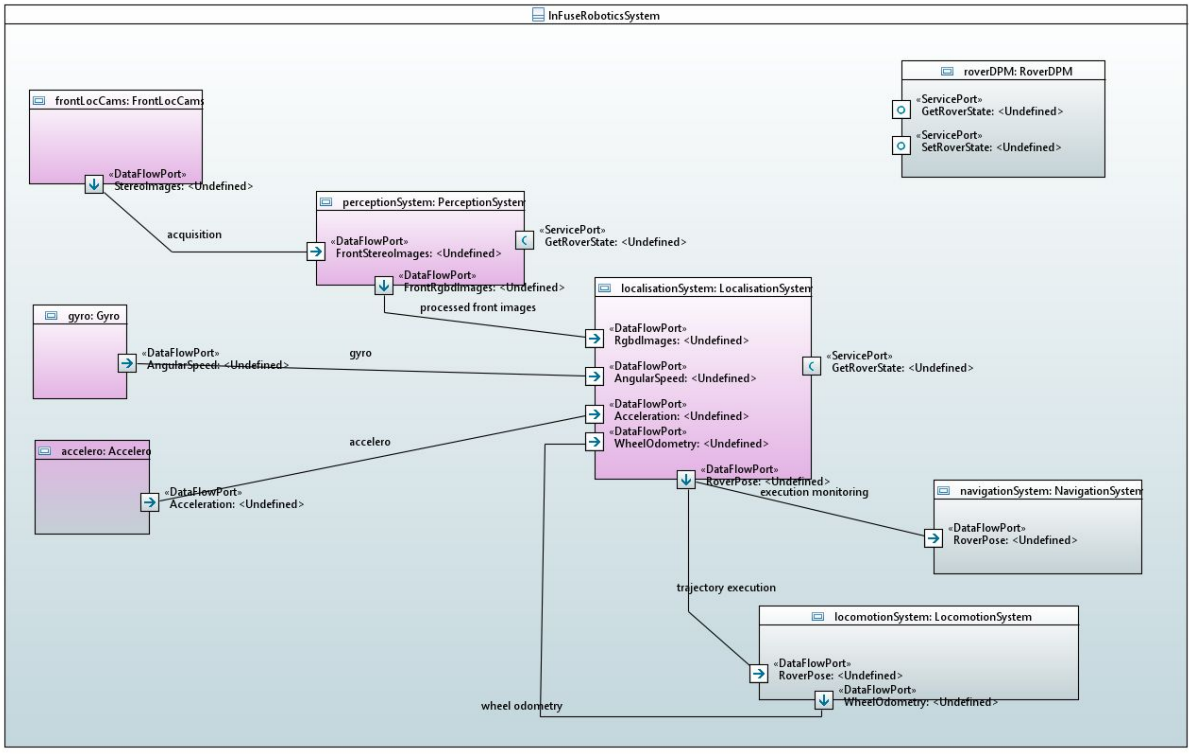


Figure 15: Visual localisation view

3.2.2.2 Autonomous navigation view

This diagram defines interfaces with the localisation and navigation systems in the scenario of Long Traverse DEM. The localisation triggers navigation cameras (NavCam) by asking to the perception system to cover a specific area (either a panorama or a specific area on the ground). From depth images, it computes navigation maps, possibly by fusing multiple acquisitions. It plans a path and sends the result to the locomotion system. The current localisation of the rover is provided by the localisation system.

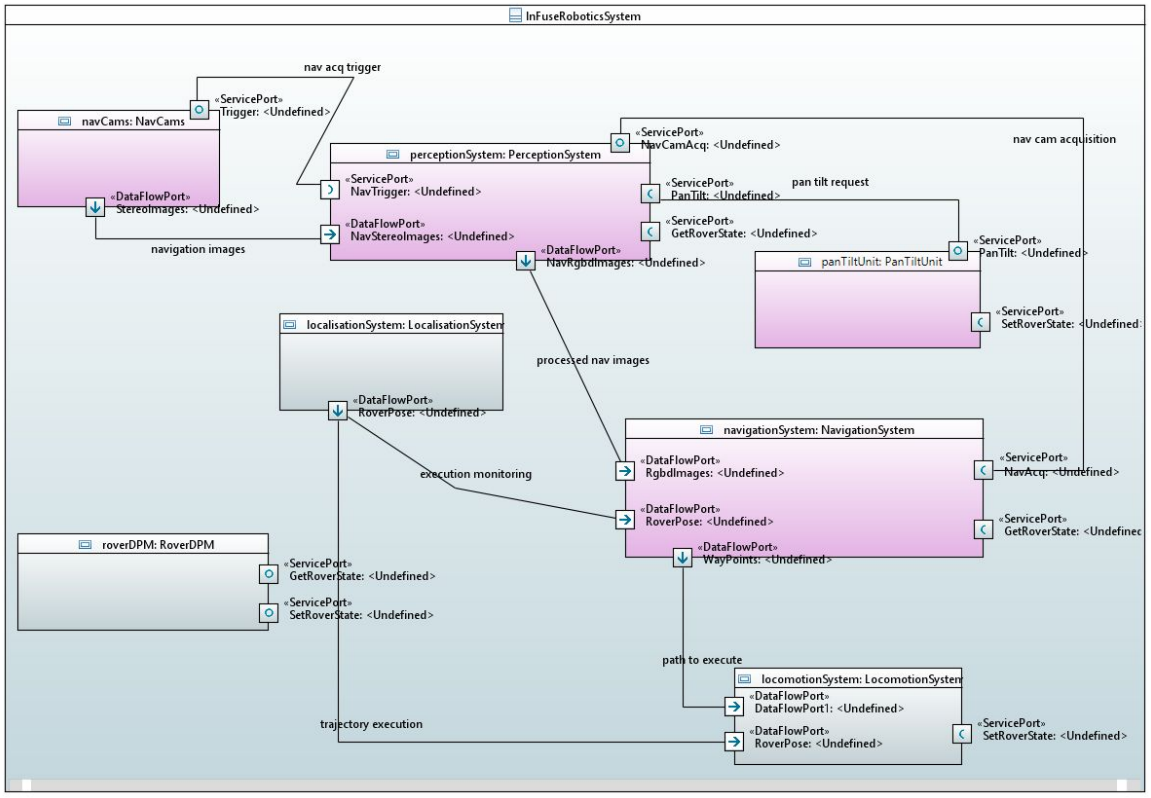


Figure 16: Autonomous navigation view

3.2.2.3 Relative Localisation view

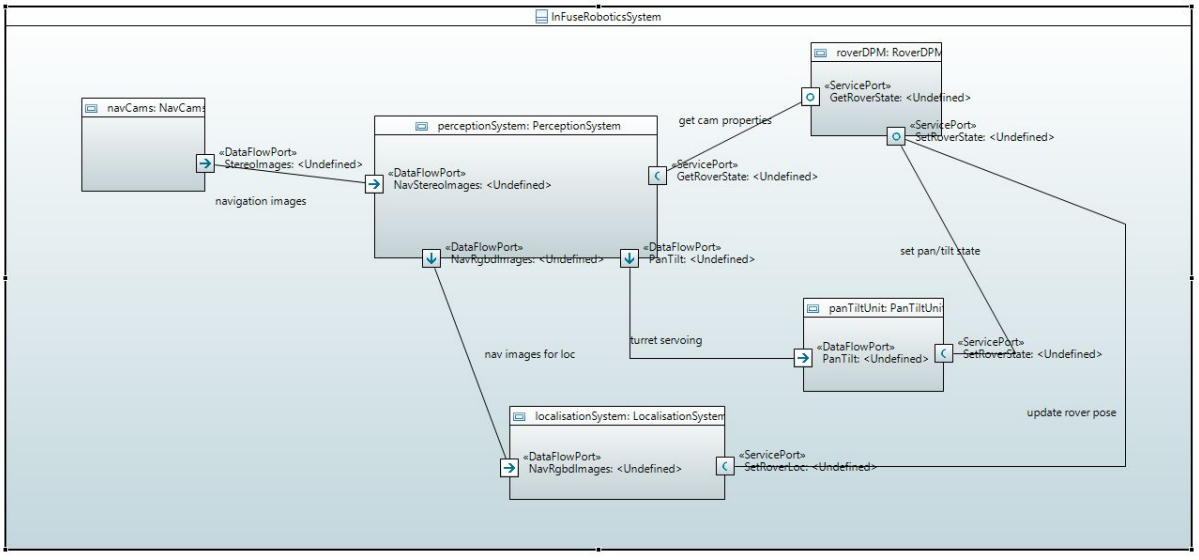


Figure 17: Relative localisation view

This diagram defines interfaces with the localisation system when relative localisation is activated. It emphasizes the role of the perception system in the management of turret to follow a target.

3.2.2.4 Rover Data Product Manager

This diagram defines interfaces with the Rover Data Product Manager system. This allows to centralise all the rover states and share them with all other systems. Data are exchanged through services on demand.

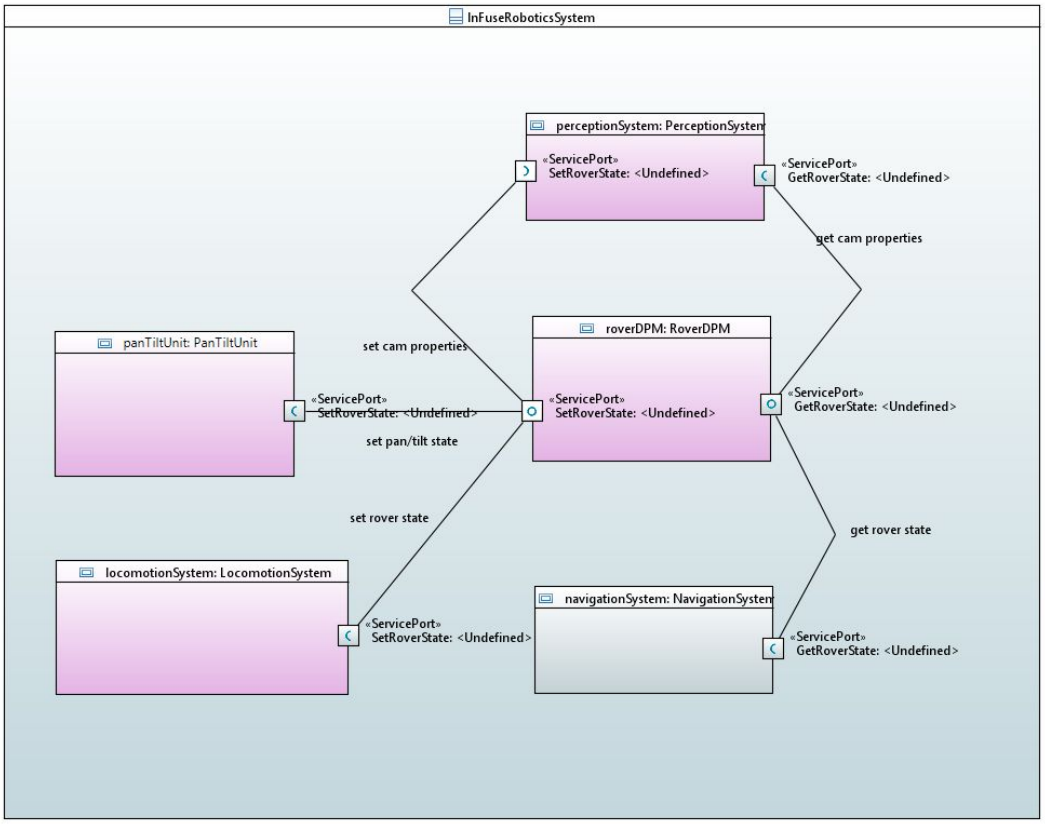


Figure 18: Rover data product manager

To conclude this chapter, we now present two examples of deployment, one for CNRS/LAAS rovers and the other with the HRCU / SHERPA setup.

For CNRS/LAAS rovers, the sensor, actuator and locomotion systems are in MANA. They communicate through ROS with InFuse deployed on a dedicated OBC. Finally, the ground station interacts with InFuse through ROS or another robotics middleware.

The MANA rover can be simulated with MORSE in a transparent way. It will ensure a fluid transition from the development phase to analog tests.

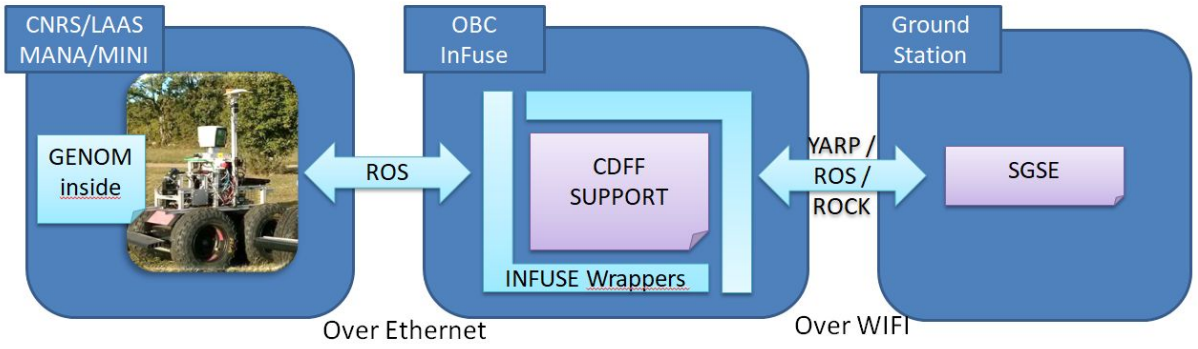


Figure 19: Deployment of InFuse with CNRS/LAAS rovers.

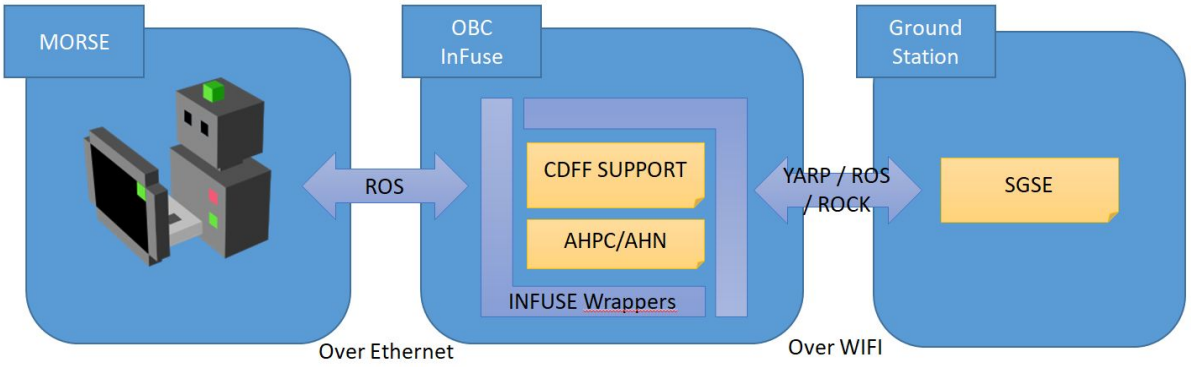


Figure 20: Deployment of InFuse with MORSE simulator.

Moreover, offline processing will be enabled for analysis and benchmarking.

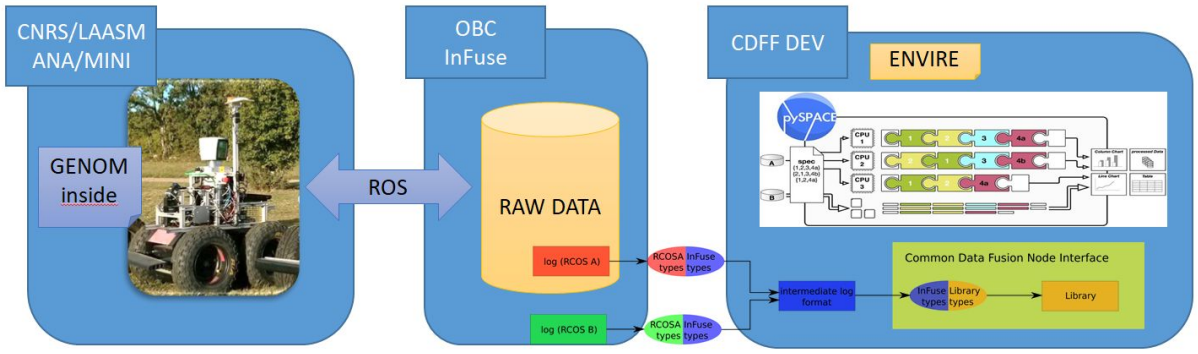


Figure 21: Deployment of InFuse for offline processing.

For DLR/DFKI setup, the sensor, actuator and locomotion systems are in the HRCU and SHERPA. They communicate through ROS with InFuse deployed on a dedicated OBC. Finally, the ground station interacts with InFuse through ROS or another robotics middleware.

In this setup, stereo vision is computed on board on a FPGA by the HRCU.

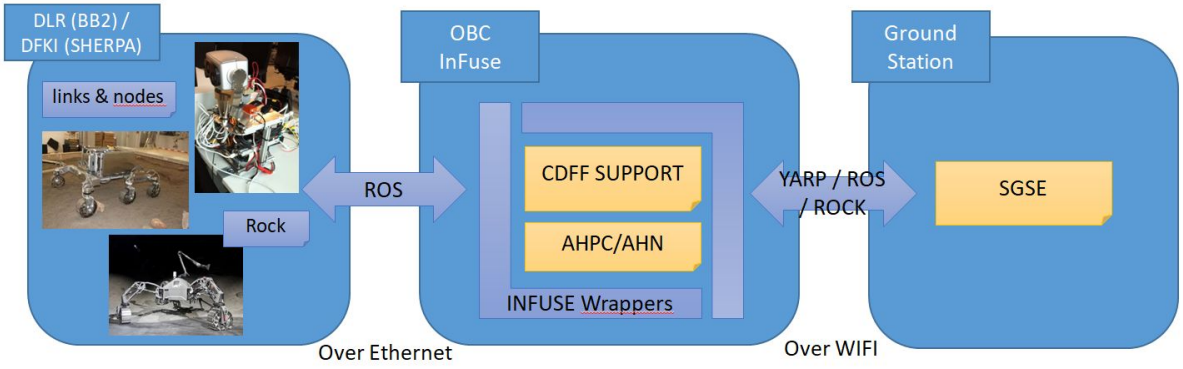


Figure 22: Deployment of InFuse with HRCU and SHERPA.

3.3 RI-SRC.SR

The work has not been done explicitly for RI-SRC.SR scenarios as they are already covered by RI-INFUSE scenarios. Indeed, RI-SRC.SR is a subset of RI-INFUSE in which less sensors will be available, loop closure will not be addressed and a different rover system will be used.

If any major difference appear between the different system architecture, they will be included in this document.

4 Detailed Architecture and Design

The goal of this chapter is to expose a first, high-level description of the DFPCs which will be developed by partners to respond to the scenarios and use cases identified in Chapter 2.

4.1 Flight software : DFPC Architecture and Design

At this level, we are focusing on the architecture of the DFPC. The list of DFNs in each DFPC, their internal interfaces, and the DFPCs external data interfaces are identified and described.

Each DFPC description follows the approach detailed in the Technical Note on DFN and DFPC Specification, included in Appendix 7.3. The DFPC architecture is presented in three parts:

- *Data Flow Description*: A functional description of the DFNs that compose a DFPC and their relations, seen only from a data-flow point of view. The goal of this description is to identify the list of required DFNs to build a DFPC.

- *Data Product Management*: A description of the shared data between the DFNs in the given DFPC, and the interfaces between this data and the various DFNs.

- *Control Description*: A description of the control flow within a DFPC: the order in which DFNs are called, DPM access to shared data, synchronicity of timestamped data. The control flow will be achieved by the DFPC Controller for implementation. This description takes the form of a sequence diagram.

In addition to this three-step architecture definition of each DFPC, a detailed design description is also foreseen. This detailed design will present the implementation specifics of each DFPC for each Reference Implementation, and will include the following elements:

- List of DFNs used
- Definition of the chosen data structures for:
    - DFPC-level inputs and outputs,
    - Shared data structures between DFNs,
    - Data managed by the DPM.
- DFPC-level parameters available to the user.
- For RI-INFUSE, since the CDFF and ESROCOS are not available and operational at the beginning of the development work, a given DFPC may be implemented with existing RCOS, middleware and tools. We thus detail:
    - RCOS and middleware choice, if applicable,
    - Implementation of the DFPC Controller,
    - Alternate data types and structures,
    - Deployment scheme on the EGSE,
    - Interfaces with external AHPCs, e.g. sensor acquisition, data display, model creation.

- Integrated tests to be performed at DFPC level.

This detailed design work is planned to be carried out in parallel with the development of DFPCs, continuously being updated with the latest design choices.

Table 2: Table 2 presents a summary of all DFPCs to be developed and the partners responsible for their description and development. Each of the following sections presents the architecture description of one of these DFPCs.

DFPC Name	Partner Responsible
Wheel Odometry	LAAS
Visual Odometry - MAG/CNES	Magellium
Visual Odometry - LAAS	LAAS
Visual SLAM	Magellium
Visual Map-based Localisation	Magellium
Long-range Tracking	Magellium
Mid-range 3D Model Detection	Magellium
Mid-range 3D Model Tracking	Magellium
Point Cloud Model-Based Localization	USTRATH, Magellium
3D Model Detection and Tracking	SPACEAPPS
Haptic Scanning	SPACEAPPS
Absolute Localization	LAAS
DEM Building	LAAS
LIDAR Pose Graph SLAM	LAAS
LIDAR Map-based Localisation	LAAS
Navigation Map Building	Magellium
Pose Fusion	LAAS

Table 2: List of DFPCs and partners responsible for each partner

4.1.1 DFPC: Wheel Odometry

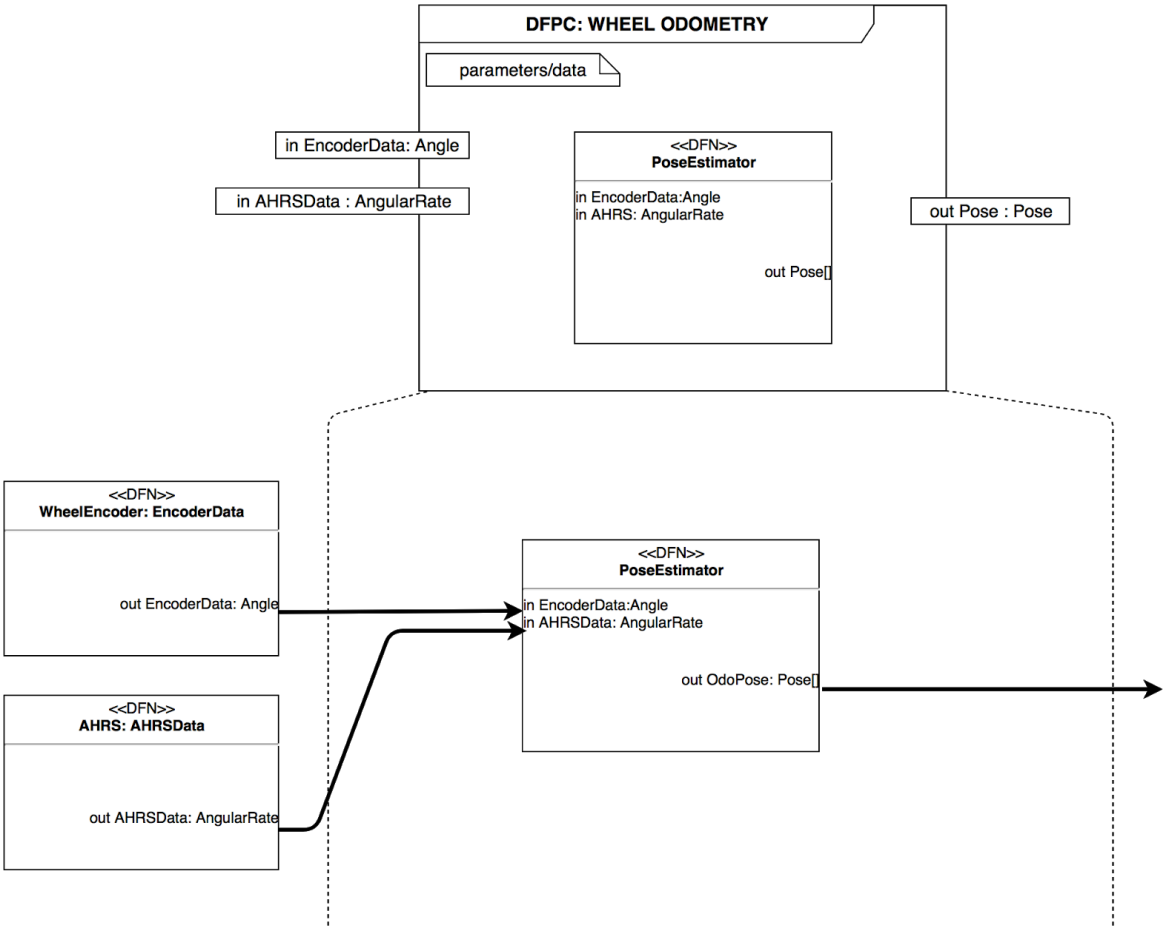


Figure 23: Wheel Odometry Data Flow Description

This figure shows the data flow in that simple DFPC. This DFPC is a core functionality which will be used by all the following reference implementation scenarios:

- RI-INFUSE-LONG-TRAVERSE-LOC
- RI-INFUSE-LONG-TRAVERSE-DEM
- RI-INFUSE-RENDEZVOUS
- RI-INFUSE-RETURN-TO-BASE

This DFPC inputs are :

- Wheel encoder data, with associated metadata
- AHRS sensor data, with associated metadata

The DFPC output is:

- Estimated rover pose

The DFPC is composed of only one simple DFN that computes the pose from raw data. This DFPC is conceptually among the simplest ones of the InFuse projet.

4.1.1.1 DFPC Expected Performance

The precision of odometry depends considerably on the type of the traversed terrain, on the rover locomotion structure and on the executed trajectories. On straight lines executed on cohesive soils, it can be precisely estimate distances with errors below 1%, whereas on sloppy sandy areas the estimate distance can be off by a few tens of percent. No reliable error model not precise performance can be proposed for odometry.

4.1.2 DFPC : Visual Odometry

This set of DFPCs responds to the following reference implementation scenarios:

- RI-INFUSE-LONG-TRAVERSE-LOC
  - Use Case 1 : Visual Odometry

Visual odometry, based around a stereo camera, provides an estimate of the pose of the rover by computing its displacement between two consecutive frames, without any memory of the frames on a longer horizon. Since it is well tested and understood, and some implementations already provide good performance on space-compatible targets, it represents a baseline solution to the Long Traverse scenario.

For this DFPC, two different implementations are considered, one from MAG/CNES, and one from LAAS.

4.1.2.1 Flavor 1: MAG/CNES Visual Odometry Implementation

The following figures detail the DFN component structure inside the DFPC.

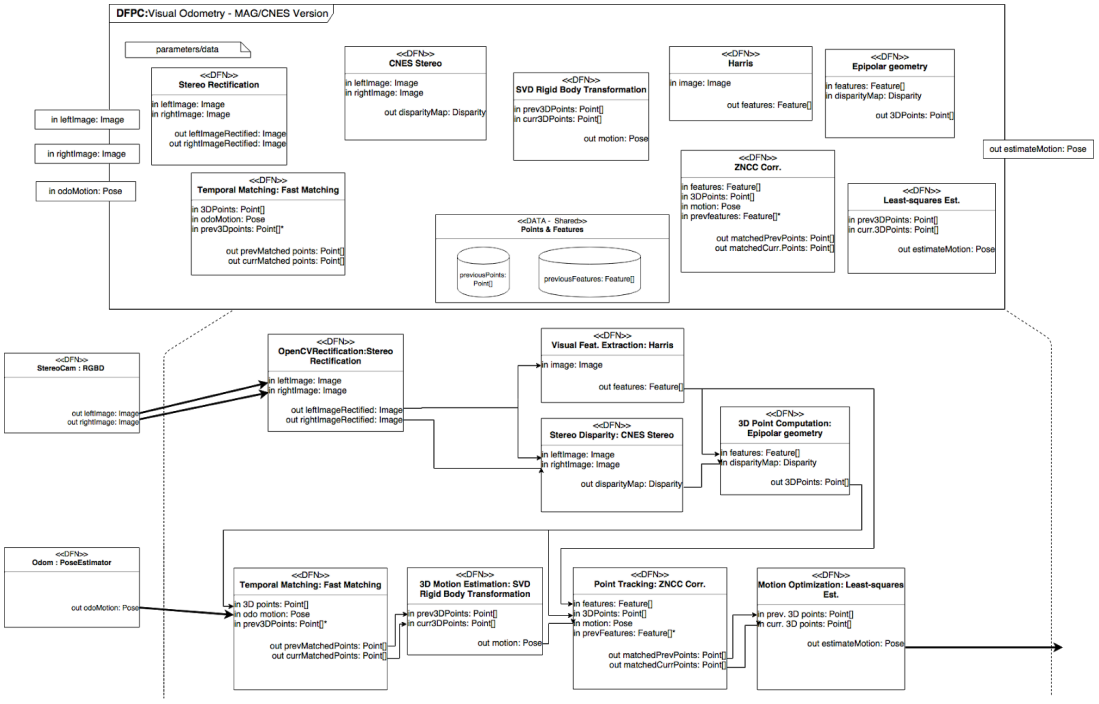


Figure 24: Visual Odometry - MAG/CNES Data Flow Description.

This visual odometry flavor supports a stereo camera, and has already been tested on space-compliant targets with satisfactory performance.

DFPC Inputs:

- Left and right stereo images with associated metadata
- Rover pose estimated from wheel odometry.

DFPC Outputs:

- Estimated rover pose.

The DFPC will be composed of the following DFNs:

- CNES Stereo Rectification: Performs a rectification of both cameras in the stereo bench, using their calibration parameters, to prepare for stereo disparity computation,
- CNES Stereo Disparity: Computes, refines and filters a disparity map from the left and right rectified images,
- Harris Visual Features Extraction: Detects and extracts Harris corners from the left image,
- 3D Point Computation: Computes the 3D position of the detected Harris features from epipolar geometry using the previously created disparity map,

- Temporal Matching: Efficiently matches the 3D points detected in the current image with the 3D points of the previous image, using wheel odometry as a prior data.
- 3D Motion Estimation: Efficiently computes a first estimate of the displacement between consecutive frames from the matched 3D points, using an SVD decomposition algorithm,
- Point Tracking: Matches features between the previous frame and the current frame by performing a dense Zero Norm Cross Correlation over their neighbourhoods,
- Motion Estimation: Provides a final pose estimation of the rover pose with a least-squares optimization of the previously matched features.

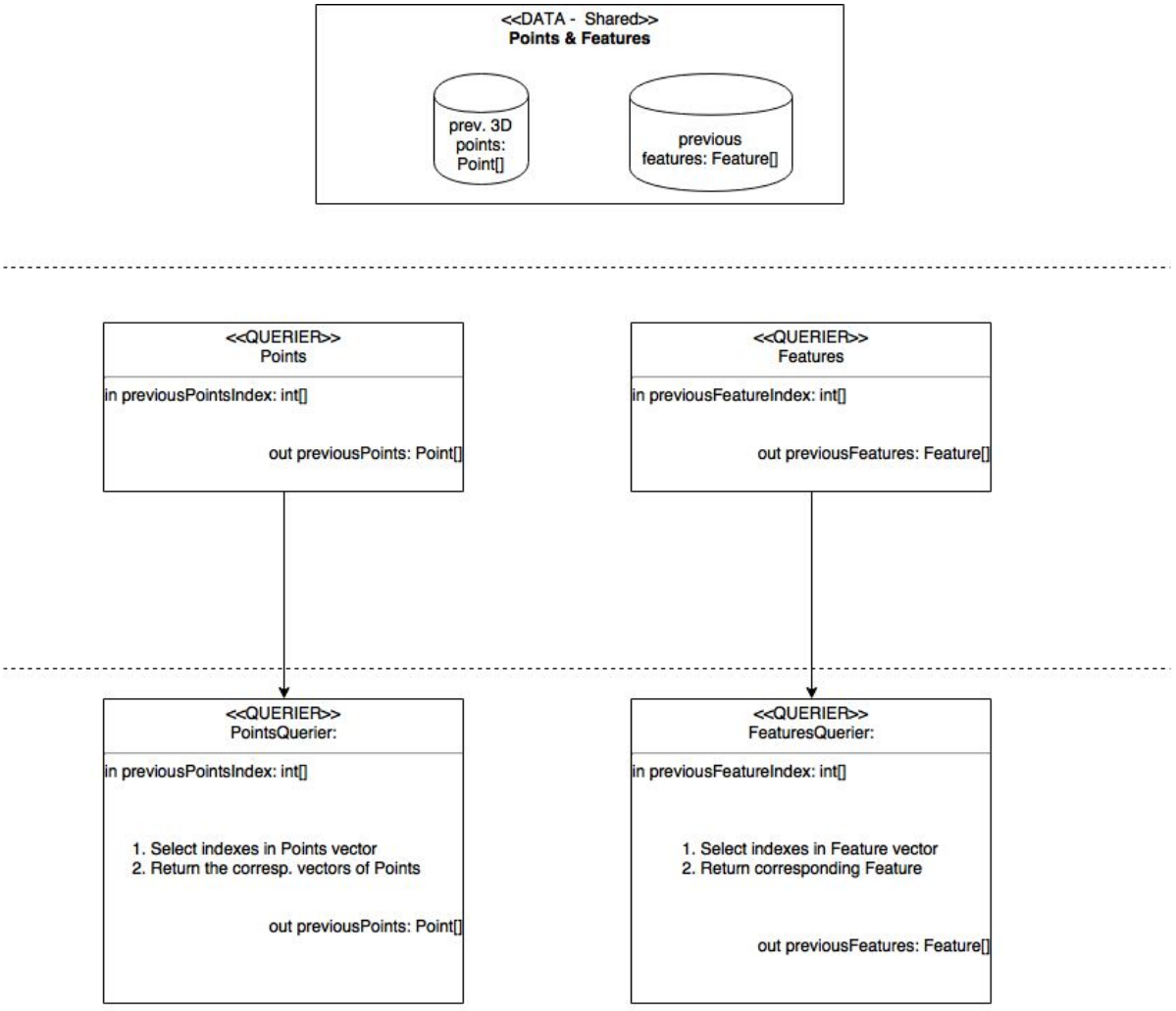


Figure 25: Visual Odometry - MAG/CNES Data Product Management.

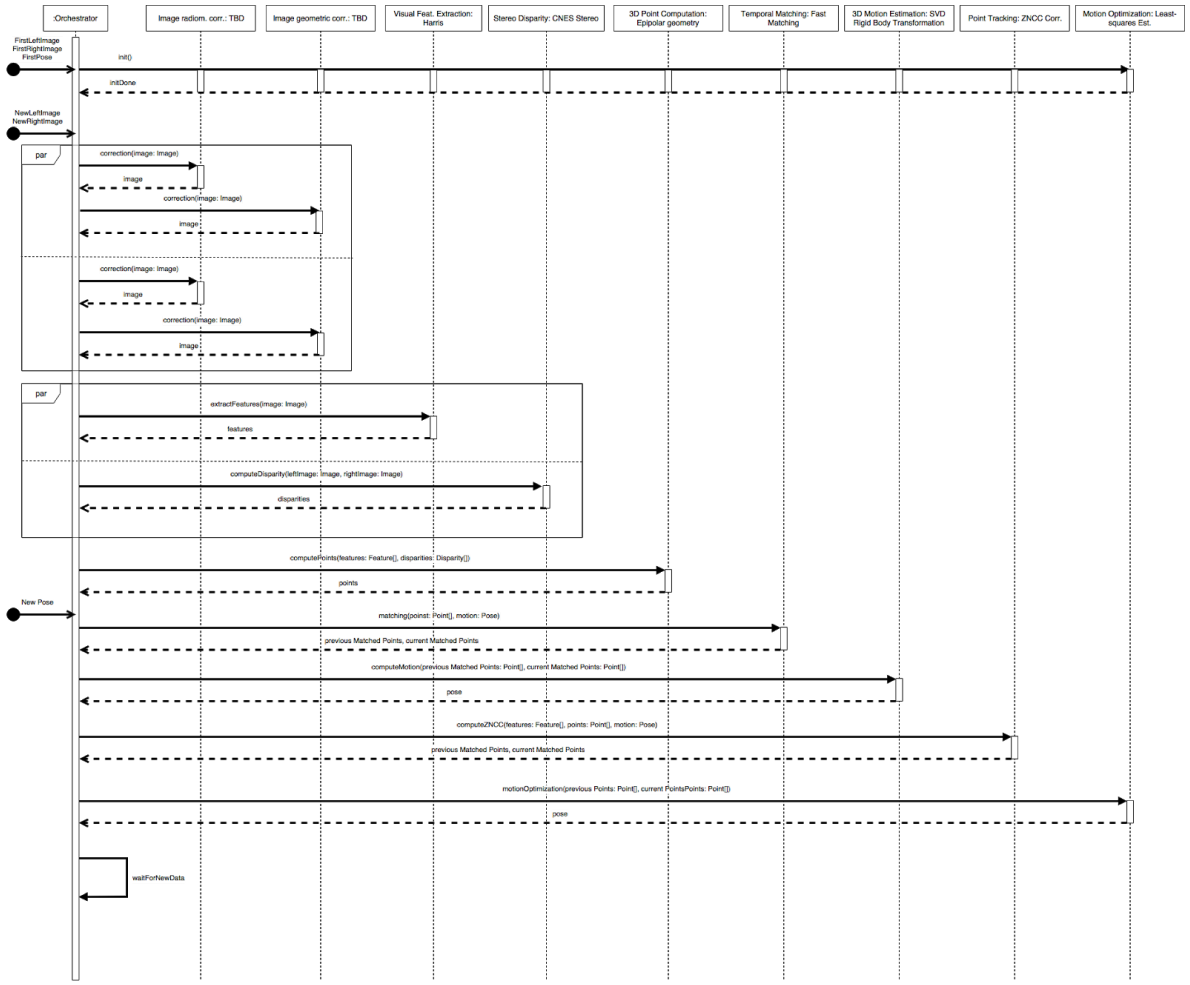


Figure 26: Visual Odometry - MAG/CNES Control Description.

4.1.2.2 DFPC Expected Performance

From the literature and consortium expertise, we can state that the localisation accuracy of a pure visual odometry function is in the order of magnitude of 2% of the travelled distance in rough terrain. In [MAIMONE2007], authors report an error of approximately 2% in some cases for MER rovers in ground-based testing conditions. In [SOUVA2008], authors report an error around 5% without integrating AHRS measurements. In [SUNDERHAUF2005], authors also report an error of 2% using sparse bundle adjustment.

A localisation error of 2% of the travelled distance is a reasonable target figure for planetary exploration rovers, knowing that in the KITTI benchmark, that address cityscapes only few participants reach a performance level below 1%.

The target localisation accuracy is 2% of the travelled distance.

[MAIMONE2007] Mark Maimone et al., Two Years of Visual Odometry on the Mars Exploration Rovers, 2007

[SOUVA2008] F. Souvannavong et al., VISUAL ODOMETRY FOR AUTONOMOUS LOCALIZATION ON MARS, Astra 2008

[SUNDERHAUF2005] Sünderhauf, N., Konolige, K., Lacroix, S. & Protzel, P. (2005). Visual Odometry using Sparse Bundle Adjustment on an Autonomous Outdoor Vehicle. Tagungsband Autonome Mobile Systeme, 2005

[KITTI] http://www.cvlibs.net/datasets/kitti/eval_odometry.php

4.1.2.3 Flavor 2: LAAS Visual Odometry

This visual odometry flavor supports a stereo camera.

DFPC Inputs:

- Left and right stereo images with associated metadata.

DFPC Outputs:

- Estimated rover pose.

This DFPC will be composed of the following DFNs:

- Stereo Rectification (Optional): In case the input is not rectified, the DFN will perform a rectification of the two cameras using their relative calibration parameters,
- FeatAndSigExtractor: Detects and extracts SIFT visual features from the images,
- Feature Matching: This DFN is used twice (with different sets of parameters). Once for left-right matching, and a second time for t/t+1 (left-left) matching,
- 3D Point Triangulation: Performs triangulation based on the matches to obtain a set of 3D points,
- (3D-3D) or (2D-3D) Motion Estimation: One of the two DFNs can be used to estimate motion between two time steps. 3D-3D can use SVD while 2D-3D solves a PnP problem.

The following figures detail the DFN component structure inside the DFPC.

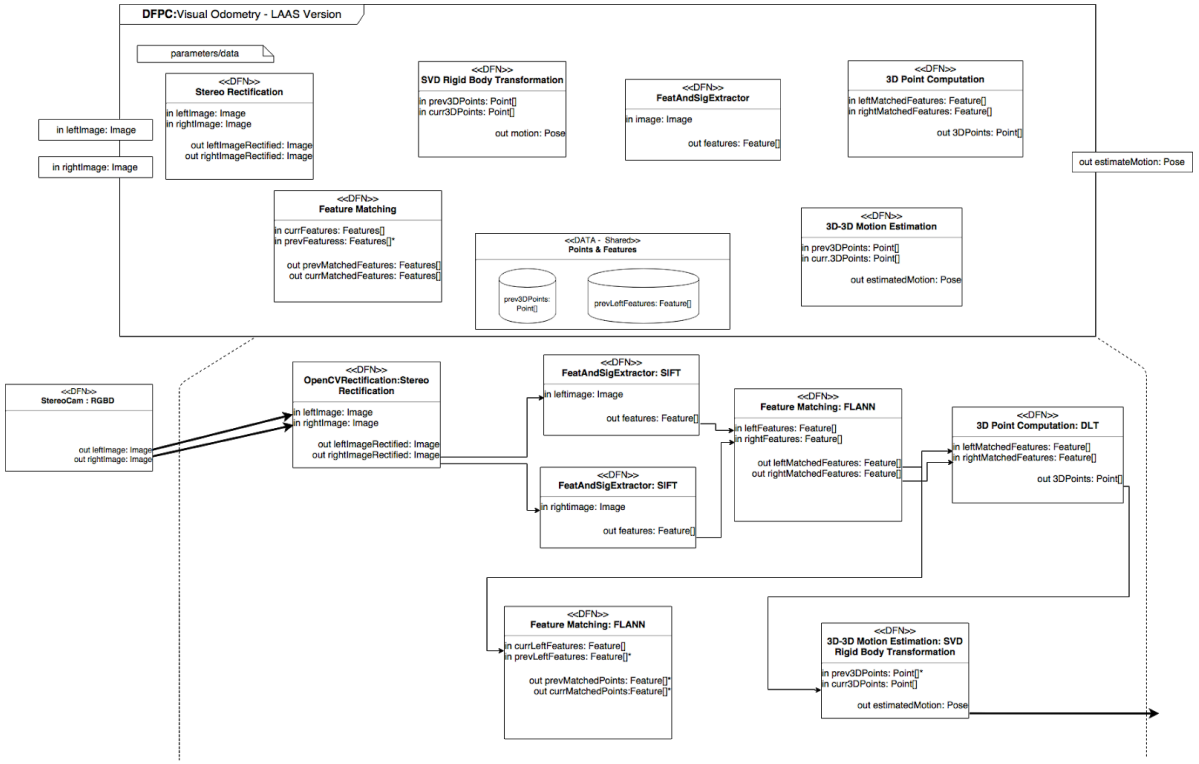


Figure 27: Visual Odometry - LAAS Data Flow Description

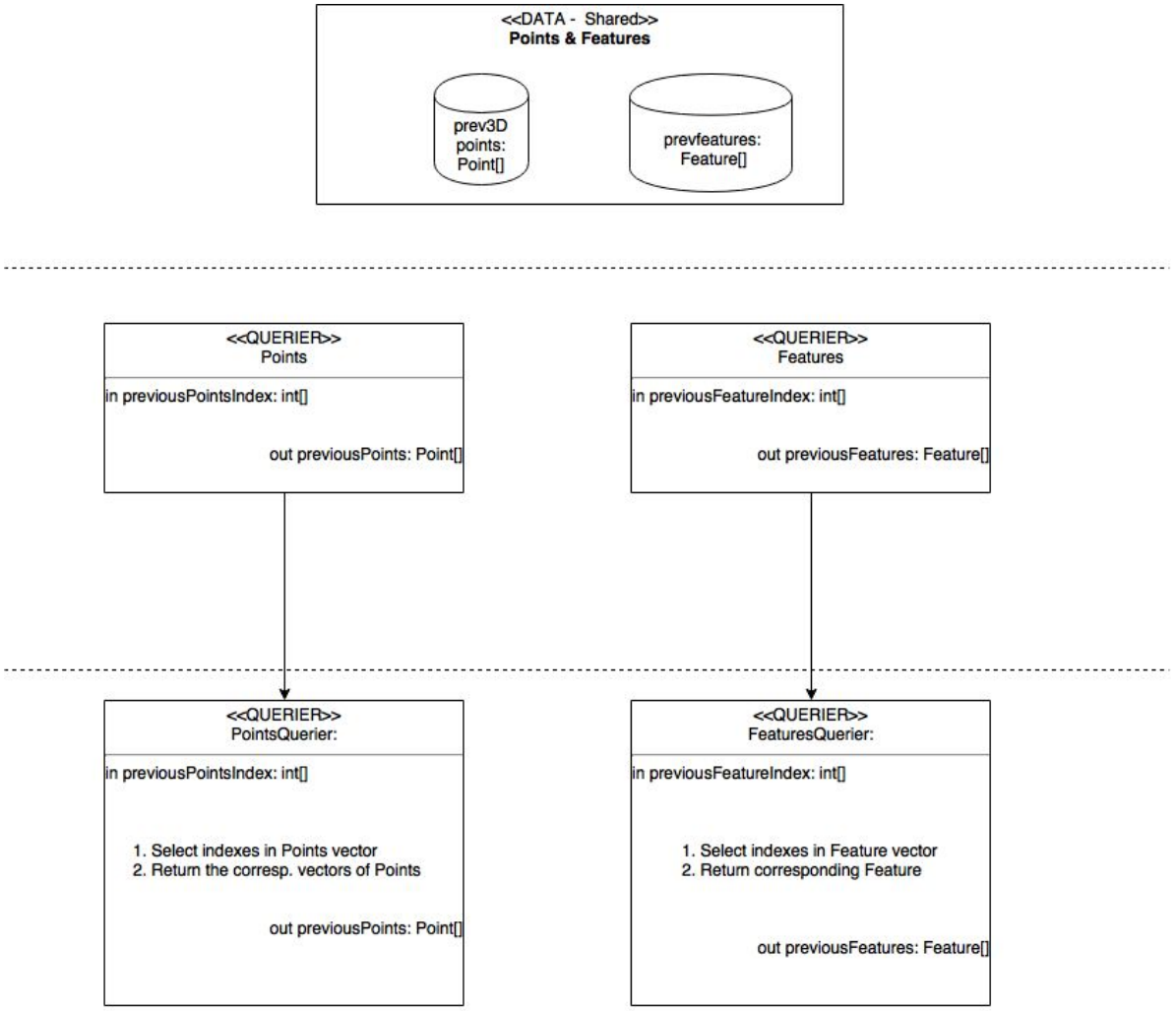


Figure 28: Visual Odometry - LAAS Data Product Management

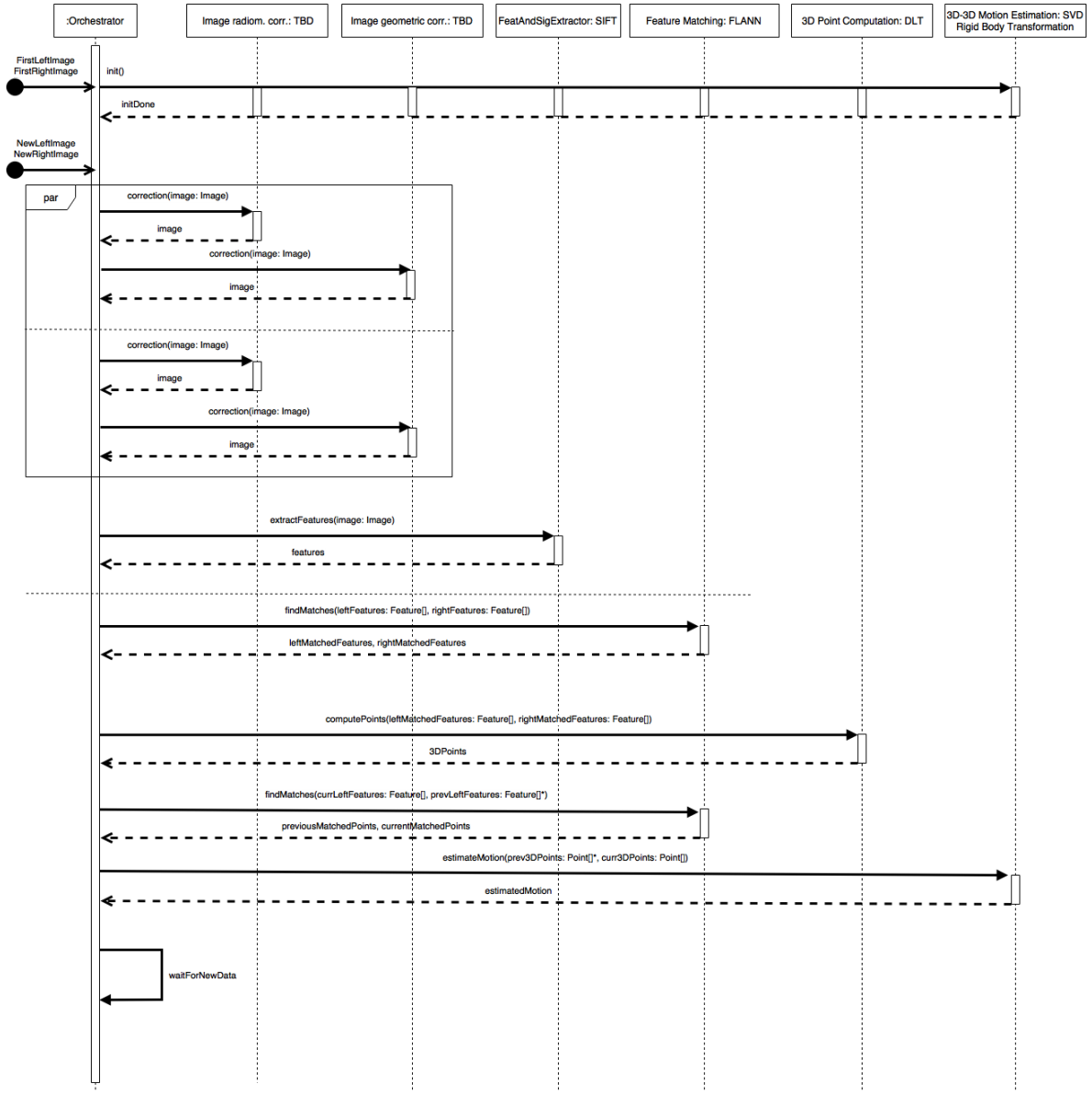


Figure 29: Visual Odometry - LAAS Control Description

4.1.2.4 DFPC Expected Performance

Expected performances detailed in 4.1.2.1 are applicable here.

4.1.3 DFPC : Visual SLAM

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-LONG-TRAVERSE-LOC

- Use Case 2 : Visual SLAM

It is also implicitly required to implement RI-INFUSE-RETURN-TO-BASE.

The Visual SLAM DFPC extends the concept of Visual Odometry to target next generation localisation solutions. Indeed, the function creates and maintains a long-term map of the explored terrain, which promises to improve localisation accuracy, especially if an area is revisited and a loop closure can be computed.

Since this DFPC performs a rather large and diverse set of functions, it was chosen to decompose it in a smaller number of higher-level DFNs. This simplifies the DFPC architecture, but still retains the atomic nature of DFNs, following our definition in Appendix 7.3.

DFPC Inputs:

- Stereo images with associated metadata,
- Depth image with associated metadata,
- RGB Image with associated metadata,
- Rover pose estimated from wheel odometry.

DFPC Outputs:

- Estimated rover pose in map.

The DFPC will be composed of the following DFNs:

- Feature and Signature Extractor: Detects keypoints in the left and right RGB images and returns associated descriptors. ORB features are proposed for an initial implementation, but one could opt for a different type while keeping an essentially identical DFPC architecture,
- Simple Predictor Pose Estimator: Performs a first prediction of the current rover pose using the odometry estimated pose input and a simple motion model,
- Relocaliser: Attempts to estimate an initial rover pose using 2 alternative methods:
    - If tracking has been successful in the previous frame, it matches the current keypoints with the previous frame's keypoints and performs a motion-only optimisation to give a first estimate of the rover pose,
    - Otherwise, if tracking has failed, the DFN uses a bag-of-words method to find the map keyframe which is most likely to be the closest, and if this is successful, uses a Perspective n-Point method to estimate a new initial rover pose.
- Loop Closure: Runs in parallel with the rest of the process and uses a bag-of-words method to attempt to find map keyframes which share a given amount of keypoints with the current frame. If a closure is detected, it performs an optimization on a wider window of keyframes, and propagates the correction throughout the whole map.

- Tracker: From the initial pose estimate given by the Relocaliser DFN, this DFN selects several more connected neighbouring keyframes and performs an additional matching step, followed by an full optimization of related keyframes and keypoints. This DFN produces the final DFPC rover pose output.
- Mapper: Updates the shared SLAM map by adding the current frame and map points to it, and performing a strict keyframe culling to remove past keyframes which are identified as redundant according to various criteria.

The following figure details the DFN component structure inside the DFPC.

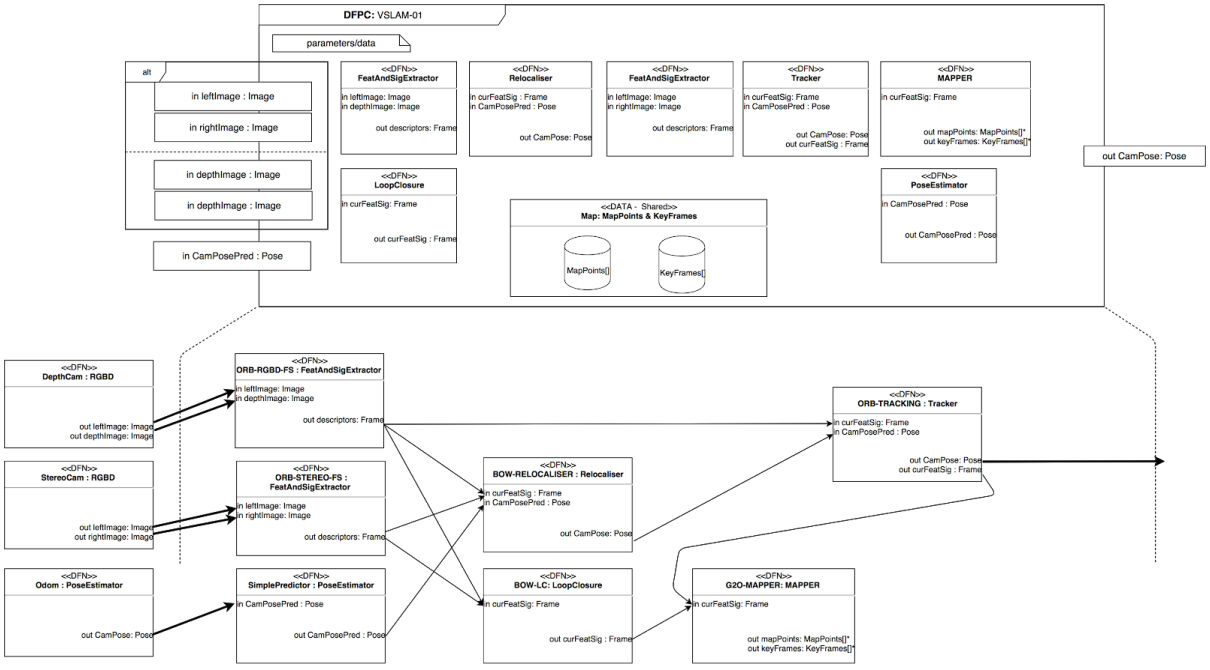


Figure 30: Visual SLAM Data Flow Description

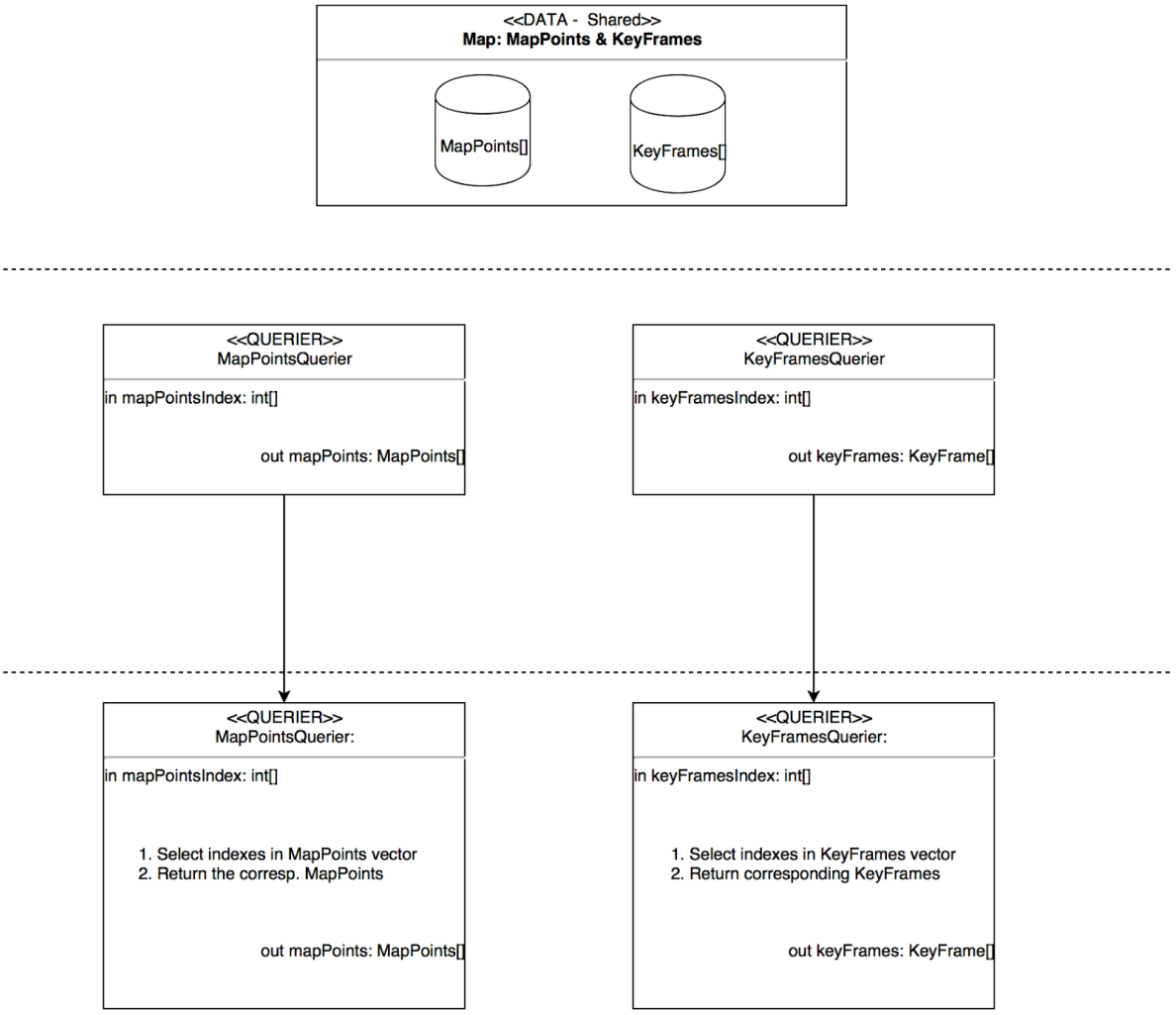


Figure 31: Visual SLAM Data Product Management

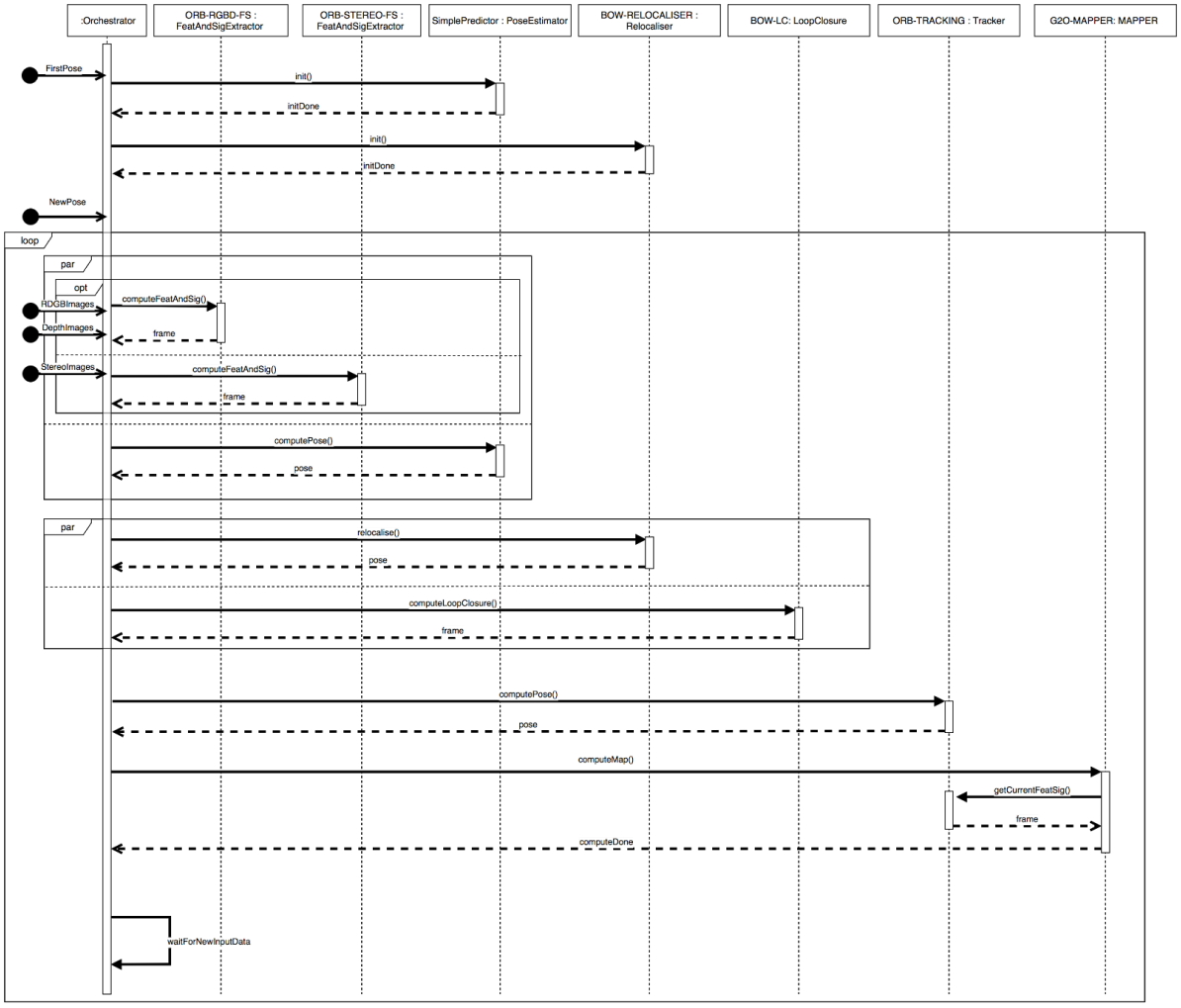


Figure 33: Visual SLAM Control Description

4.1.3.1 DFPC Expected Performance

Expected performances detailed in 4.1.2.1 are applicable here.

4.1.4 DFPC : Visual Map-based Localisation

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-RETURN-TO-BASE
  - Use Case 1 : Visual Map-Based Localisation

This DFPC, centered around ORB features, partly reuses the Visual SLAM DFPC, however excluding all DFNs pertaining to map creation and management, as well as loop closure.

We therefore only aim to localise the rover within a fixed map created previously with the Visual SLAM DFPC.

DFPC Inputs:

- Left and right stereo images with associated metadata,
- Depth image with associated metadata,
- RGB image with associated metadata,
- Rover pose estimated from wheel odometry
- Previously created SLAM landmark map.

DFPC Outputs:

- Estimated rover pose in map

The DFPC will be composed of the following DFNs:

- Feature and Signature Extractor: Detects keypoints in the left and right RGB images and returns associated descriptors. ORB features are proposed for an initial implementation, but one could opt for a different type while keeping an essentially identical DFPC architecture,
- Simple Predictor Pose Estimator: Performs a first prediction of the current rover pose using the odometry estimated pose input and a simple motion model,
- Relocaliser: Attempts to estimate an initial rover pose using 2 alternative methods:
    - If tracking has been successful in the previous frame, it matches the current keypoints with the previous frame's keypoints and performs a motion-only optimisation to give a first estimate of the rover pose,
    - Otherwise, if tracking has failed, the DFN uses a bag-of-words method to find the map keyframe which is most likely to be the closest, and if this is successful, uses a Perspective n-Point method to estimate a new initial rover pose.
- Tracker: From the initial pose estimate given by the Relocaliser DFN, this DFN selects several more connected neighbouring keyframes and performs an additional matching step, followed by an full optimization of related keyframes and keypoints. This DFN produces the final DFPC rover pose output.

The following figure details the DFN component structure inside the DFPC.

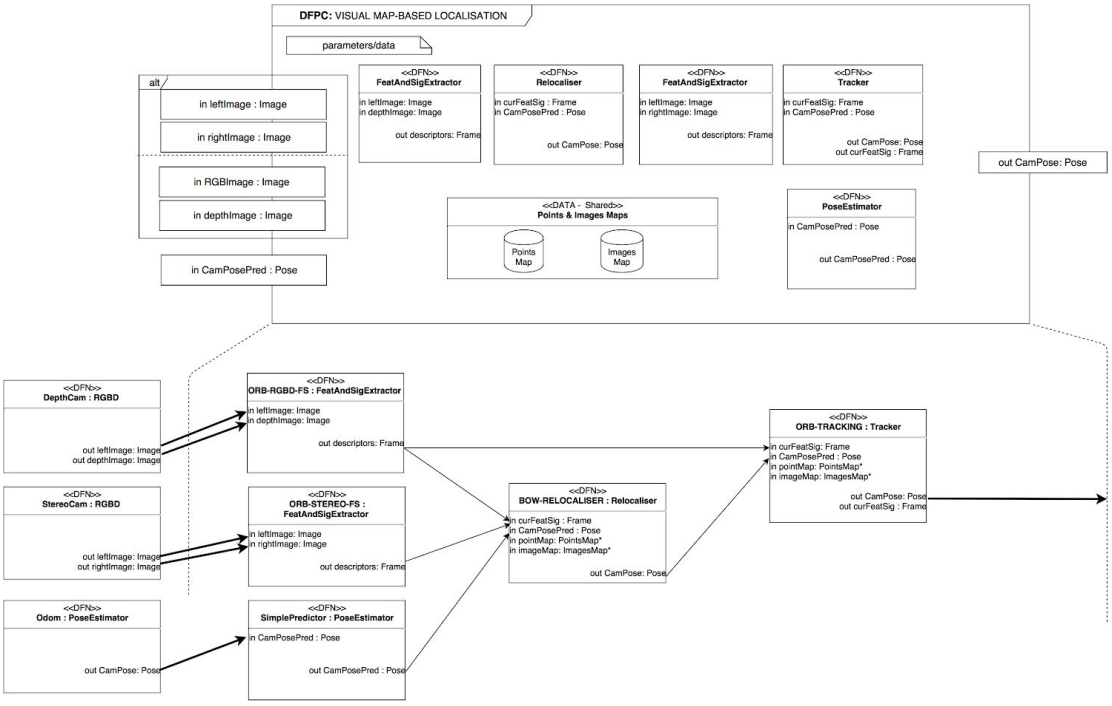


Figure 34: Visual Map-Based Localisation Data Flow Description

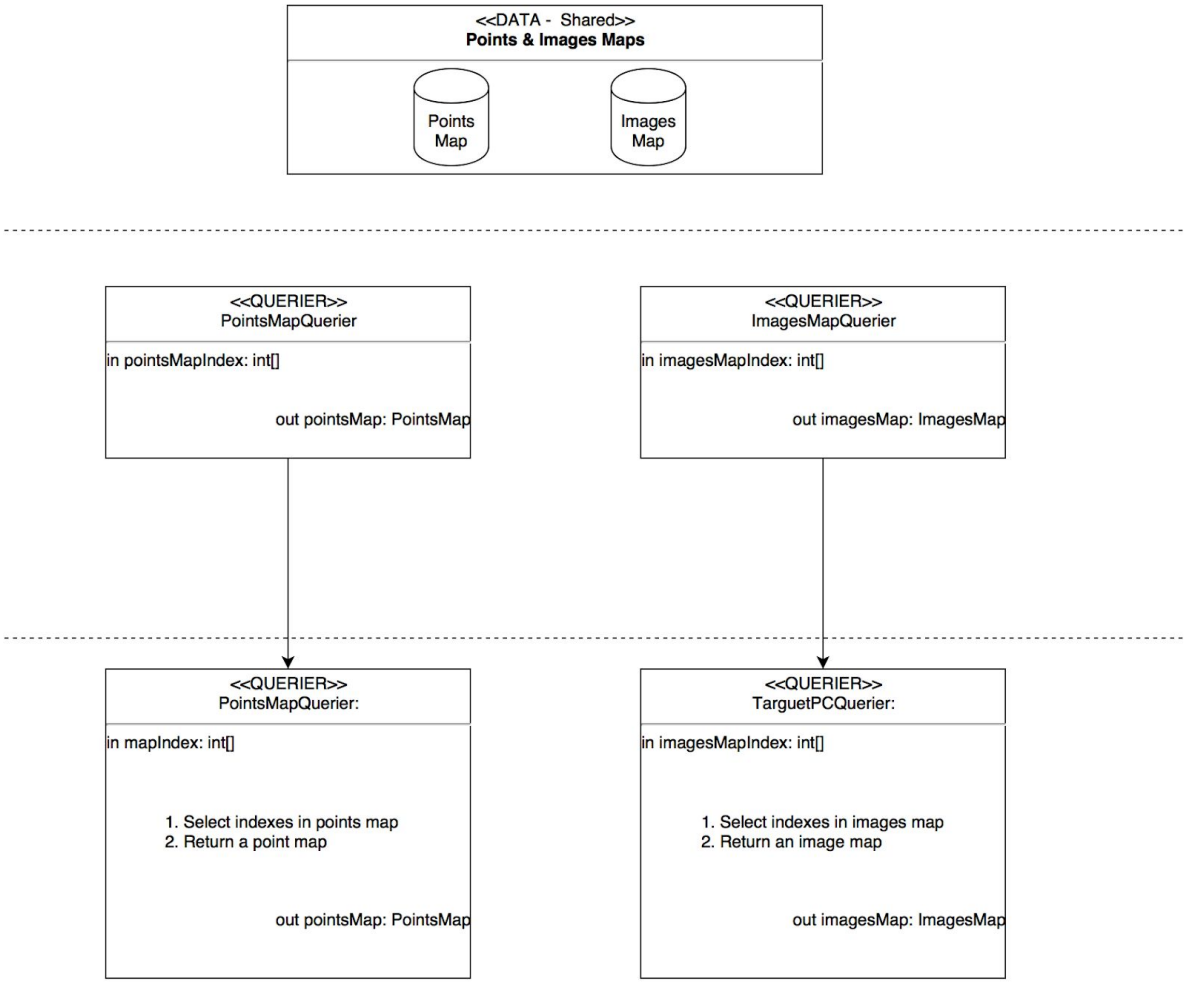


Figure 35: Visual Map-Based Localisation Data Product Management

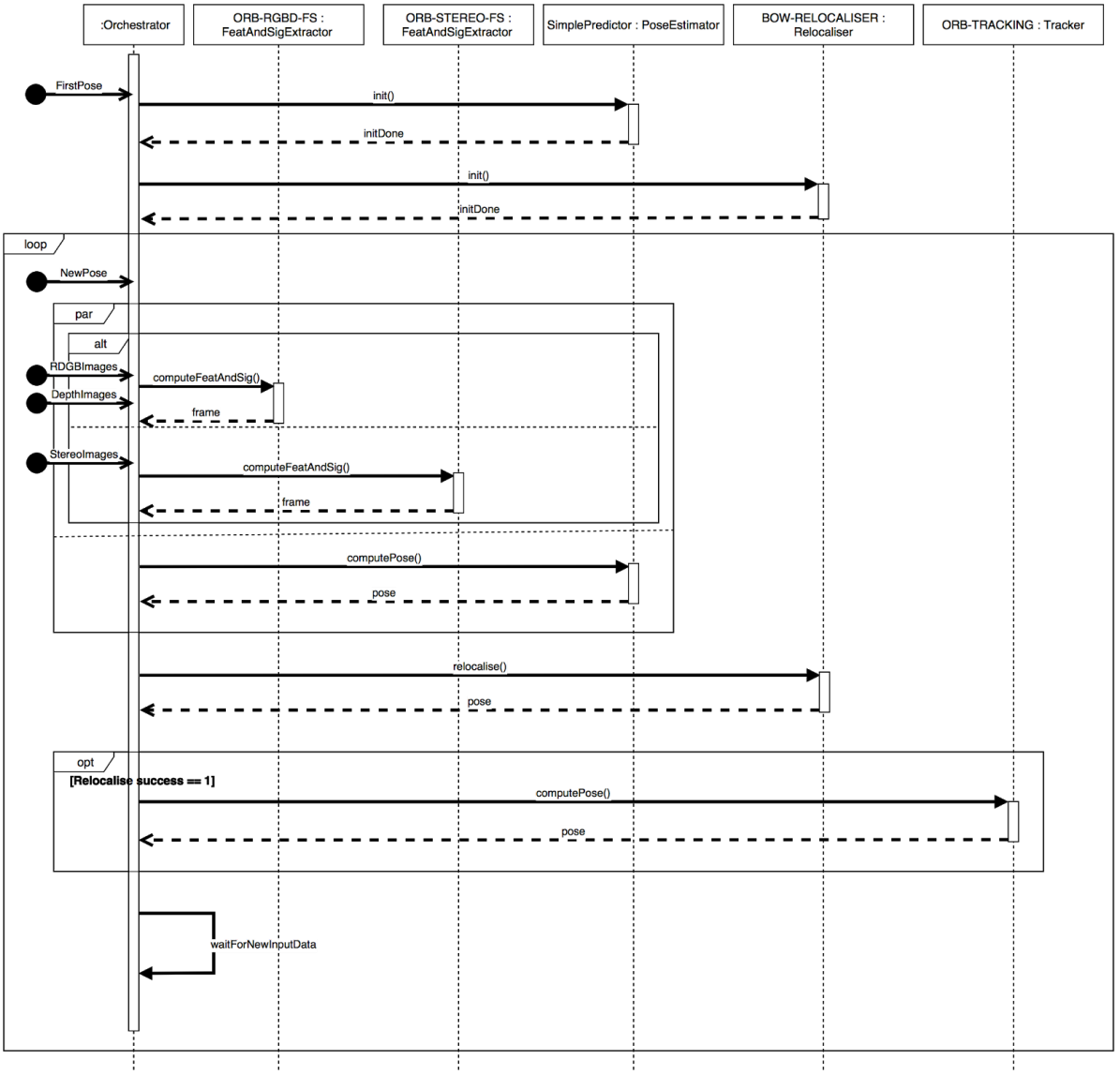


Figure 36: Visual Map-Based Localisation DFPC Control Description.

4.1.4.1 DFPC Expected Performance

Map-based localisation is a kind of absolute localisation solution as it provides a pose estimate with respect to past observations. This kind of approach is well known for autonomous cars in cityscapes, but less common for exploration rovers. In [YEOM2017], this technology leads to low deviations (≤ 0.5m) from the planned mission. In [KELLY2017], authors present the Visual Teach and Repeat (VT&R) system that is capable of autonomously repeating kilometer-scale routes with centimeter-scale accuracy in rough terrain, using only monocular vision.

The target localisation accuracy is 0.5m of the closest and detected taught image.

[YEOM2017] Brian Yeomans and al., MURFI2016 - FROM CARS TO MARS: APPLYING AUTONOMOUS VEHICLE NAVIGATION METHODS TO A SPACE ROVER MISSION, Astra 2017
[KELLY2017] Clement L, Kelly J, and Barfoot T D. “Robust Monocular Visual Teach and Repeat Aided by Local Ground Planarity and Colour-Constant Imagery”. Journal of Field Robotics, special issue on “Field and Service Robotics”, 2017

4.1.5 DFPC: Long-range Tracking

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-LONG-RANGE-TRACKING
    - Use Case 1: Long-range Bearing-only Target Tracking

It is used to provide a relatively simple bearing-only relative localisation of the target asset when its distance with regard to the rover is too great to allow for a full pose estimation. Localisation is performed through tracking, in an RGB camera input, of a visual feature previously initialized by the user.

DFPC Inputs:

- RGB image with associated metadata,
- Rover attitude from AHRS.

DFPC Outputs:

- Estimated rover pose relative to target.

The DFPC will be composed of the following DFNs:

- User Interface: Provides a way for the user to see the input images and initialize the position of the target by selecting a ROI within them,
- EKF Prediction: Performs a prediction of the expected position of the target feature in the image using a rover motion model and the current image timestamp,
- ZNCC Matching: Matches the saved target feature ROI within the new acquired image,
- EKF Correction: Uses the results of the matching DFN as an observation to update the filter and compute an estimation of the relative pose of the target with regard to the rover. This is the DFN which provides the final pose output of the DFPC.

Some architecture choices have been made for this DFPC:

- The ZNCC matcher optimizes a homography to represent feature ROI position in the input images,

- The features are represented by a ROI image, a homography with regard to its source image, and the camera pose of the source image,
- The EKF prediction is separated from the correction in order to support the case where images are not acquired at a constant frequency. It thus needs a timestamp input. It returns a predicted homography with regard to the current image,
- We currently propose to use AHRS data in the EKF correction step. However, in a different implementation, it could be used to perform EKF prediction,

The following figures detail the DFN component structure inside the DFPC, the shared data structures, and its DFN calling sequence.

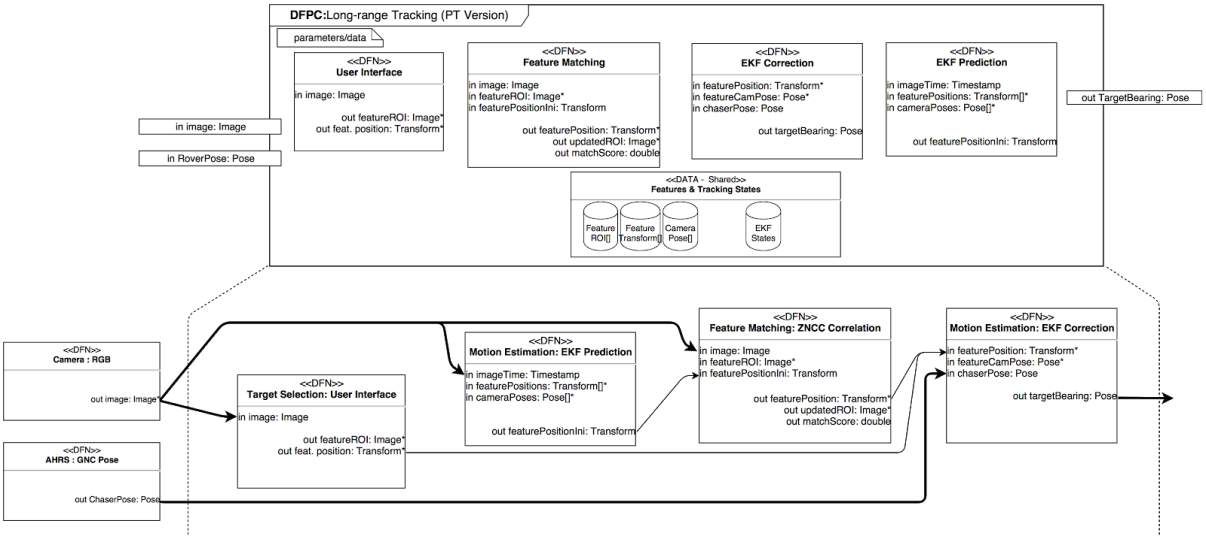


Figure 37: Long-range Tracking Data Flow Description.

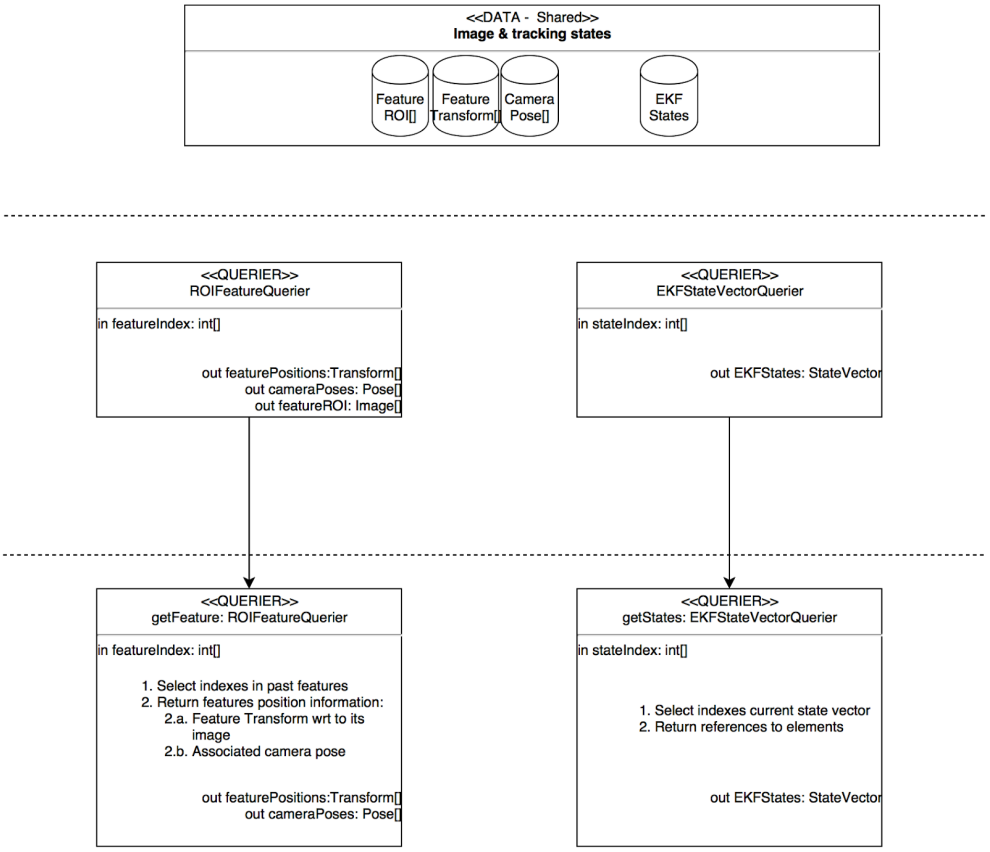


Figure 38: Long-range Tracking Data Product Management.

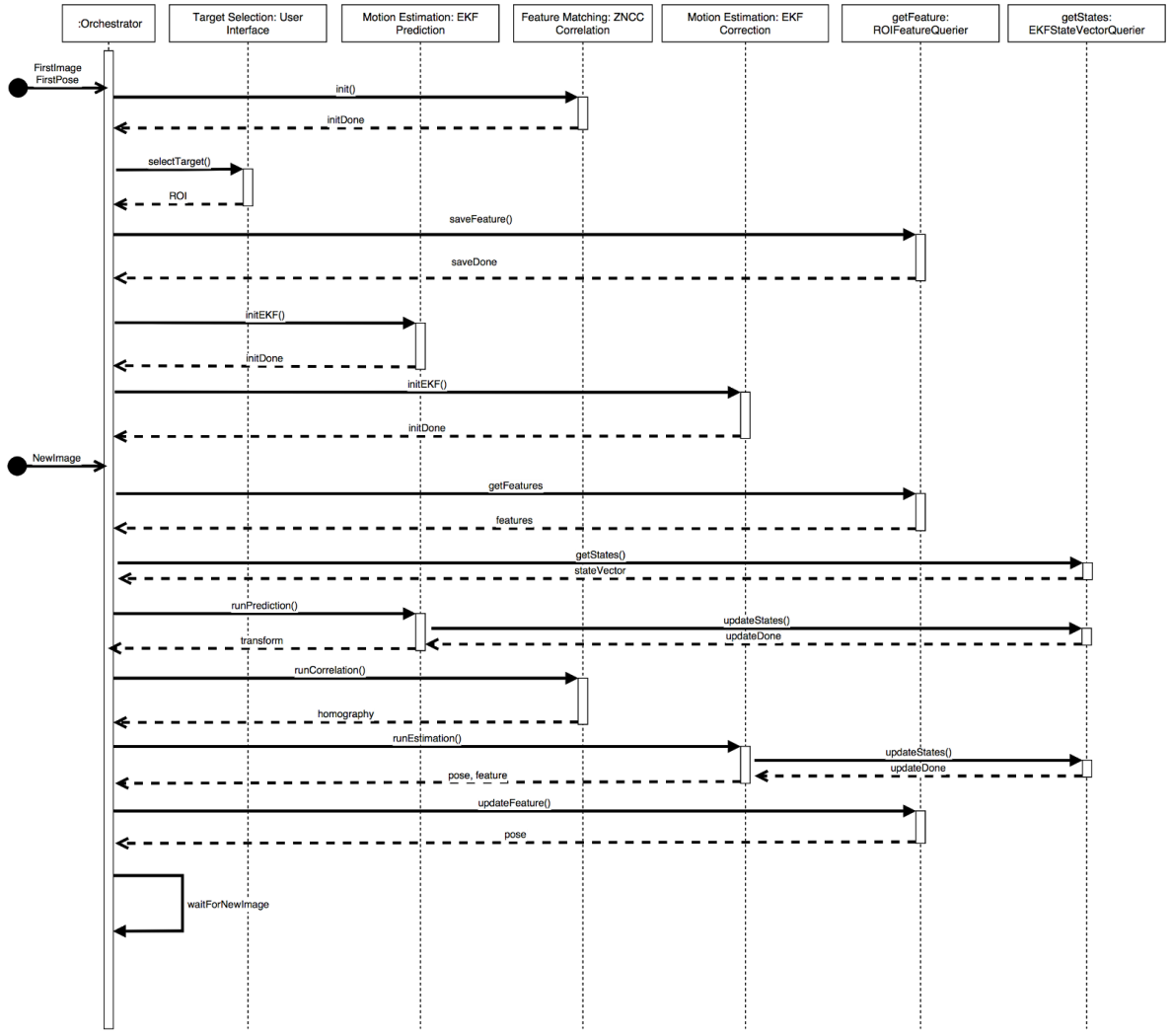


Figure 39: Long-range Tracking Control Description

4.1.5.1 DFPC Expected Performance

Since the long range tracking DFPC only operates in bearing to guide the rover towards its target, we expect, from benchmarking and litterature figures, the tracking accuracy of the center of the target to be in the order of magnitude of 1 pixel on the sensor, which would be sufficient to allow for further rendezvous operations. Tracking rate is expected to be sufficiently fast to perform autonomous navigation at 1Hz. Another measure of success is the guarantee that tracking can be successful over the whole approach trajectory.

4.1.6 DFPC : Mid-range 3D Model Detection

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-RENDEZVOUS

- Use Case 1: RGB-D Model-based Localisation

The processing compound attempts to detect a known target within its input stereo image pair and, if successful, returns a coarse estimated relative pose. The detection process is based on the LINEMOD template detection algorithm, which requires a 3D CAD model of the target. A training step is first performed offline with the model, and the resulting template is then loaded by the detection DFN. The template consists in a large database of the object's most discriminant features in various modalities, including gradients and surface normals, from all possible point of views. Each input image is then efficiently compared to the template, and the function signals a successful detection if the computed similarity exceeds a given threshold.

As this is only a detection DFPC, we do not include any long term tracking nodes such as a filter, but instead this DFPC could be used in conjunction with Mid-range 3D Model Tracking as a pose initialization step.

DFPC Inputs :

- Left and right stereo images with associated metadata.

DFPC Outputs:

- Estimated rover pose with regard to target.

The DFPC will be composed of the following DFNs:

- Stereo Rectification: Performs a rectification of both cameras in the bench using their calibration parameters,
- OpenCV Stereo Correlation: Computes, refines and filters a disparity map from the left and right rectified images. Computes the associated depth map,
- LINEMOD Template Detection: Loads the target template and performs a detection using an input RGB image and its depth map. This DFN provides the final DFPC estimated target pose signalling a successful detection.

The following figure details the DFN component structure inside the DFPC.

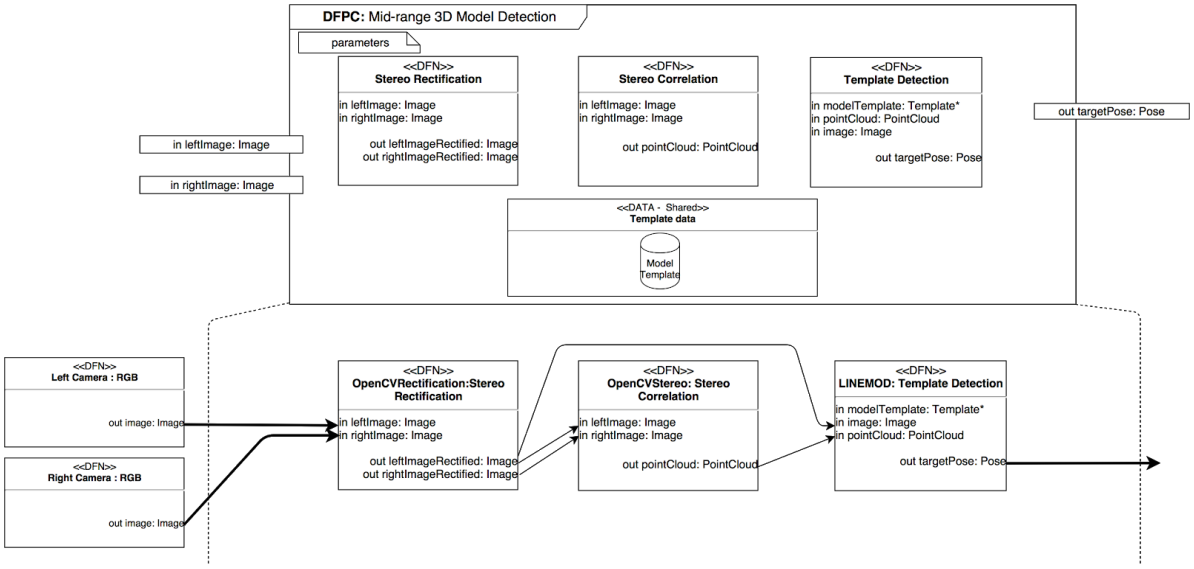


Figure 40: Mid-range 3D Model Detection Data Flow Description.

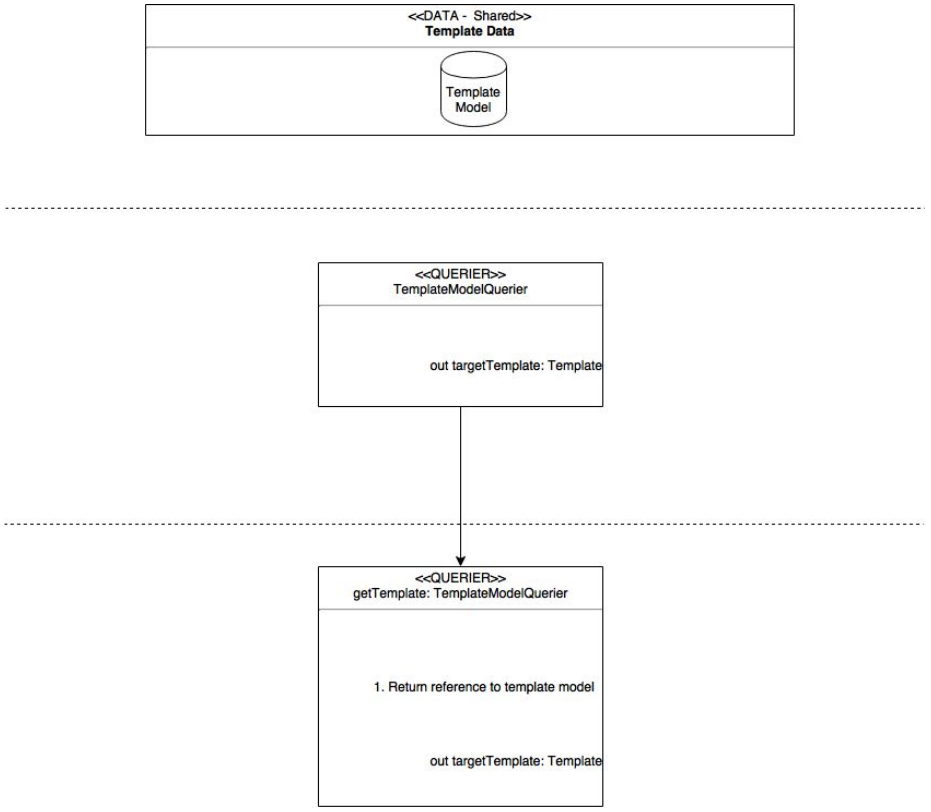


Figure 41: Mid-range 3D Model Detection Data Product Management

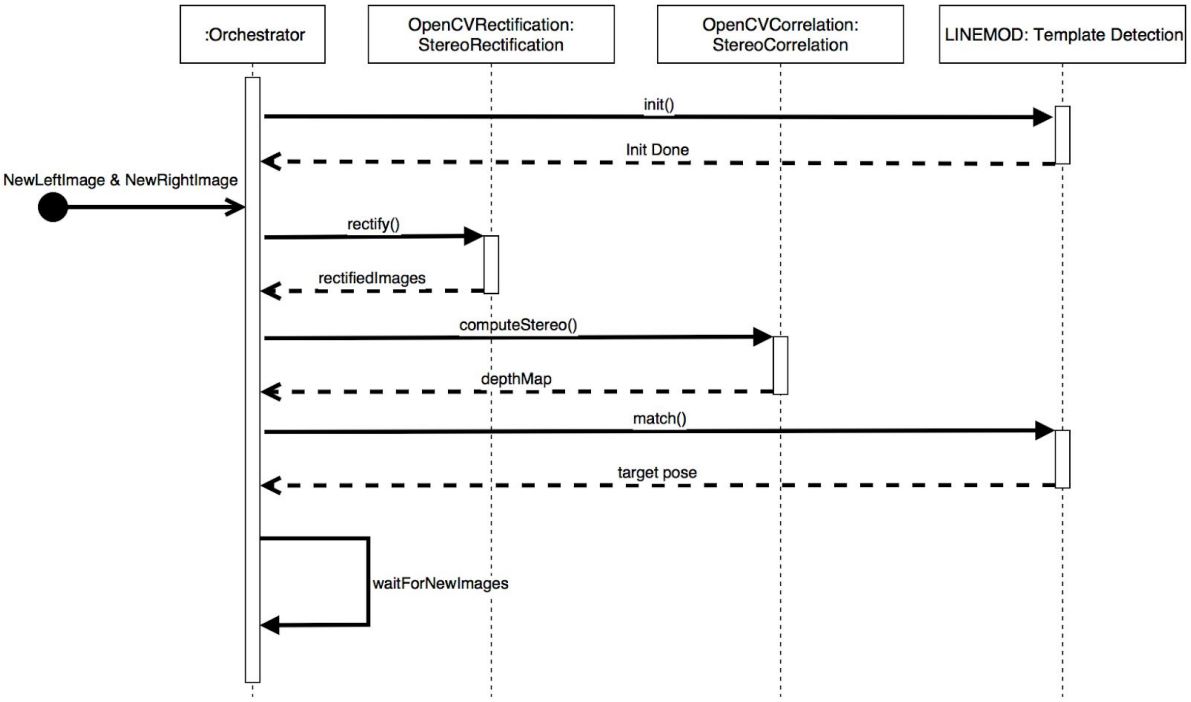


Figure 42: Mid-range 3D Model Detection Control Description

4.1.6.1 DFPC Expected Performance

This DFPC performs detection of a known object and only a coarse first estimation of its relative pose. Successful detection of the object is determined by verifying that the estimated pose falls within a given tolerance of ground truth. The final pose estimation accuracy is then only dependent on the spatial resolution of the trained template (i.e. the number of discrete angular and linear camera positions used to perform the training). The higher the number of vertices in the training, the better the accuracy, but with the cost of a longer computation time.

Performance figures available in the litterature mirror results obtained by benchmarks carried out during the tradeoff analysis phase. In [HINTERST2012], detection is performed on a selection of objects with a template trained with a spatial sampling of 15 degrees in rotation and 10 cm in scale. In these conditions, the target is successfully detected, on average, between 83% and 93% of the time, with an average rate of 8Hz on a desktop computer.

Our benchmarks in simulation indicate similar performance, but the spatial sampling of the training will need to be refined, as the size and range of the target is around an order of magnitude larger, further impacting the pose estimation accuracy.

[HINTERST2012] Hinterstoisser, Stefan, et al. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes." *Asian conference on computer vision*. Springer, Berlin, Heidelberg, 2012.

4.1.7 DFPC : Mid-range 3D Model Tracking

This DFPC responds to the following reference implementation scenario:

- RI-INFUSE-RENDEZVOUS
    - RGB-D Model-based Detection and Tracking

This DFPC is activated once the rover is close enough for the camera to resolve geometric features on the target. It is based on the existing VISP Model-based Tracker library, which is able to track a known 3D target using two types of features (and their combination) : visible edges and corners, and KLT keypoints. The tracking function is thus adapted for textured or untextured objects, with visible edges or not.

The target needs to be described with an input CAD model file in order to specify its geometric primitives. It also already includes its own sub functions such as keypoint extraction, edge visibility computation, and real-time tracking filter, therefore the DFPC is quite simple, as it is composed of self-contained DFNs.

DFPC Inputs:

- RGB image with associated metadata.

DFPC Outputs:

- Estimated rover pose with regard to target.

The DFPC will be composed of the following DFNs:

- User interface - Pose Initialization: Displays input images, and provides an interface for the user to (optionally) click to initialize the target pose,
- VISP Template Tracking: Implements the full tracking chain, with keypoint extraction and matching, edge visibility computation, and pose estimation. This DFN provides the final DFPC pose output.

The following figures detail the DFN component structure inside the DFPC.

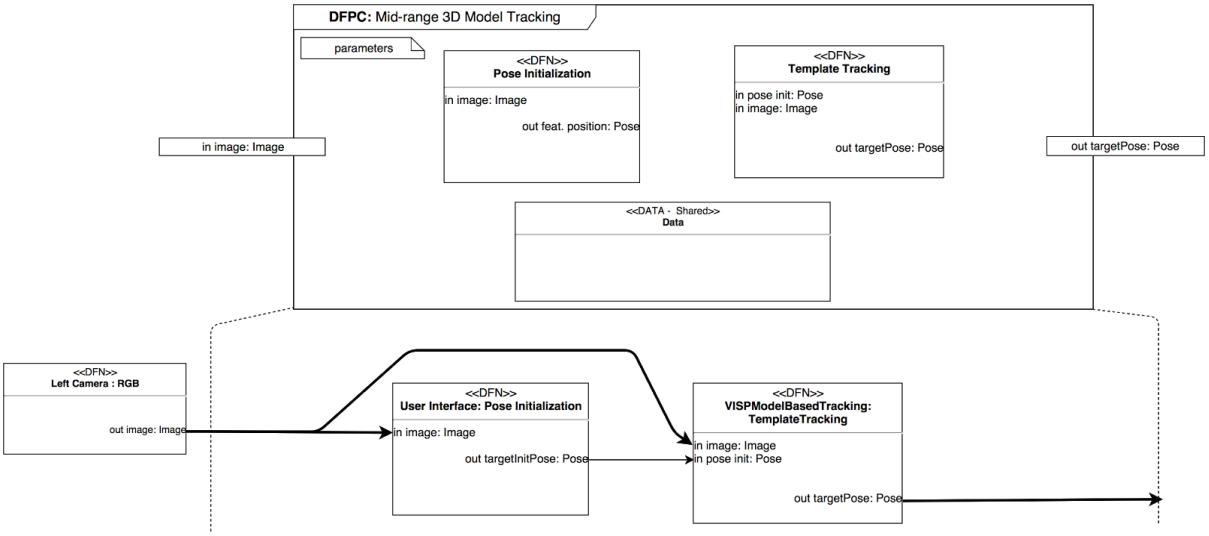


Figure 43: Mid-range 3D Model Tracking Data Flow Description

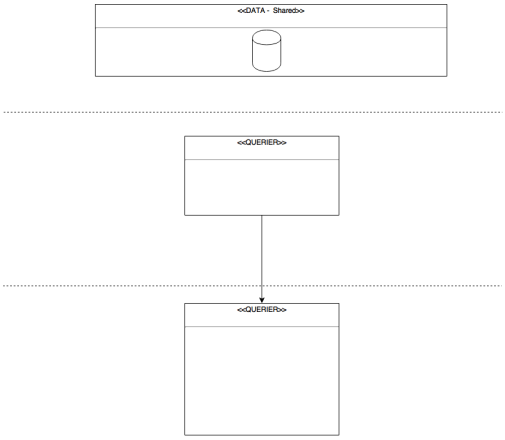


Figure 44: Mid-range 3D Model Tracking Data Product Management

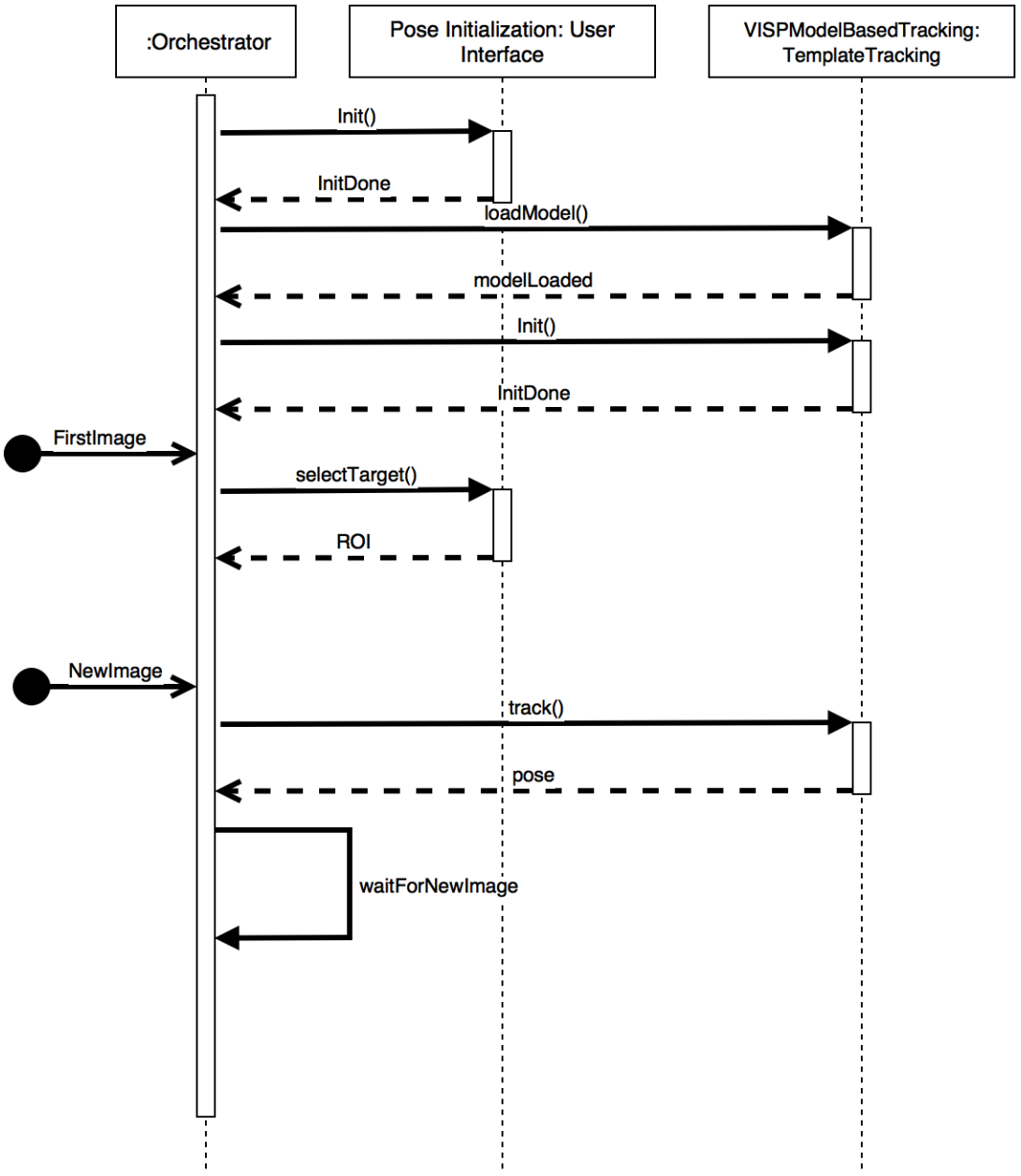


Figure 45: Mid-range 3D Model Tracking Control Description

4.1.7.1 DFPC Expected Performance

From tests and preliminary benchmarks performed during the tradeoff analysis, we expect the tracking accuracy to be affected mostly by scene conditions (e.g. lighting, background), target geometry, and the approach trajectory. As a comparison baseline, the following accuracy intervals, with variations due to the environment conditions, have been obtained with simulated rendezvous sequences (1024x1024 camera resolution, focal length 35mm):

Table 3: Mid-range 3D Model Tracking Expected Accuracy Figures

Range (m)	Position RMS Error (m)	Angle RMS Error (deg)
10	0.01 to 0.05	0.01 to 0.02
25	0.1 to 0.5	0.02 to 0.05
50	3.5 to 6	0.5 to 1

4.1.8 DFPC : Point Cloud Model-Based Localisation

This set of DFPCs responds to the following implementation scenarios:

- RI-INFUSE-RENDEZVOUS
  - Use Case 2: Dense Point Cloud Model-based Localisation
  - Use Case 3: 3D Feature Model-based Localisation

The general goal of this set of DFPCs is to perform a robust tracking of a known target described by a point cloud, in data either obtained directly from LiDAR sensors or ToF cameras, or computed from stereo camera or mono camera images. The point clouds should be sampled or resampled to be at similar resolutions but point clouds can be generated from any of these relevant sensors.

Two distinct, but similar methods are proposed to respond to Use Case 2 and Use Case 3 respectively. The first is a relatively simpler implementation aiming to perform a dense point cloud matching between an acquired one and the provided model. The second attempts to make use of 3D features detected within the input point cloud in order to increase matching performance and robustness. Both methods require that a sufficiently dense point cloud model of the target is provided in advance by the user.

4.1.8.1 Flavor 1: ICP Point Cloud Matching

In this specific DFPC flavor, we propose a naive implementation of point cloud tracking built around an EKF with a simple motion model. The measurements are provided to the filter by performing an ICP matching step between the input point cloud and the provided model. The ICP algorithm is aided by an initial prediction of the target pose, and the EKF correction step is enhanced by measurements coming from the rover's AHRS sensor.

DFPC Inputs:

- Rover attitude from AHRS,
- Point cloud with associated metadata from LiDAR sensor or stereo camera or ToF camera.

DFPC Outputs:

- Estimated rover pose with respect to target.

The DFPC is composed of the following DFNs:

- ICP Point Cloud Registration: Using an initial pose estimation, applies an ICP algorithm to determine the transform that minimizes the distance (euclidean or other) between 2 input point clouds,
- EKF Prediction: Performs a prediction of the expected position of the target point cloud using the rover motion model and the current image timestamp,
- EKF Correction: Uses the results of the ICP matching DFN as an observation to update the filter and compute an estimation of the relative pose of the target point cloud with respect to the rover. This is the DFN which provides the final pose output of the DFPC.

Some specific architecture choices have been made when defining this DFPC:

The Kalman filter tracks 12-DOF system states which contains the target pose (6-DOF) and frame-to frame local velocity (6-DOF). Here we assume a constant velocity motion model, i.e the frame to frame relative motion of the camera and the target is constant. The filter inputs: process noise, measurement noise and initial covariance are tracker parameters and have to be provided by the user. Moreover, the Kalman filter requires initial states which could be provided by an external means such a detection DFPC. After reasonable initialization, the filtering rate is obviously higher than actual point cloud processing time. Hence, the filter latency with low dimensional state vector is not our concern in this particular case where computational burden is highly related to the image-based pose estimation.

The EKF prediction is separated from the correction in order to support the case where images are not acquired at a constant frequency. It thus needs a timestamp input.

We currently propose to use AHRS data in the EKF correction step. However, in a different implementation, it could be used to perform EKF prediction.

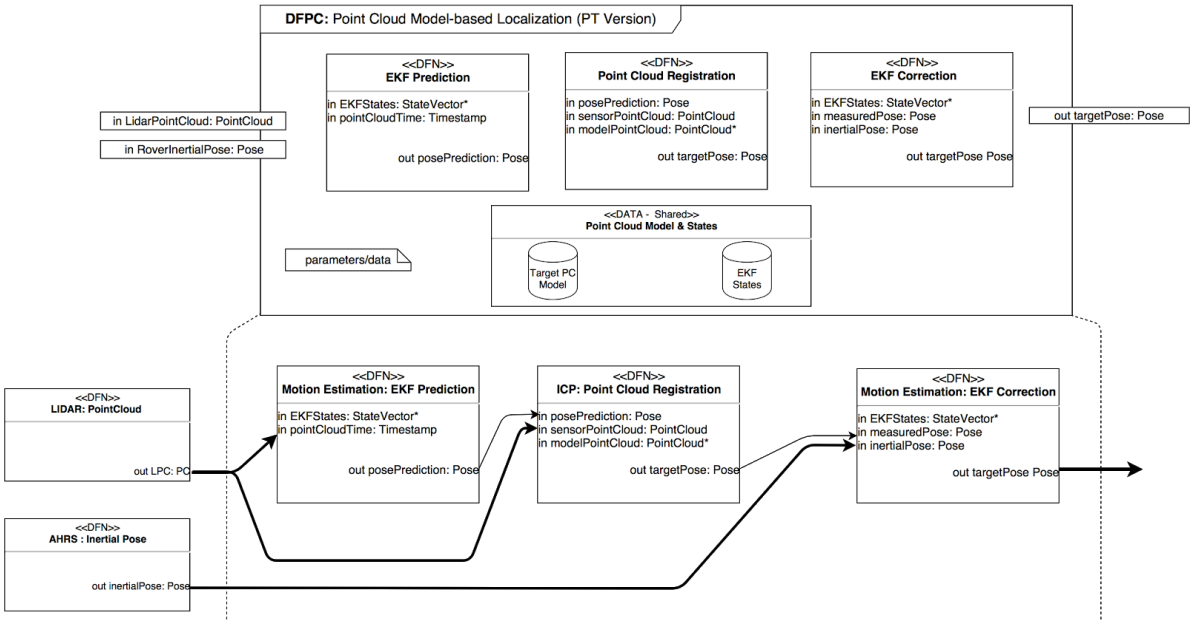


Figure 46: ICP Point Cloud Matching Data Flow Description

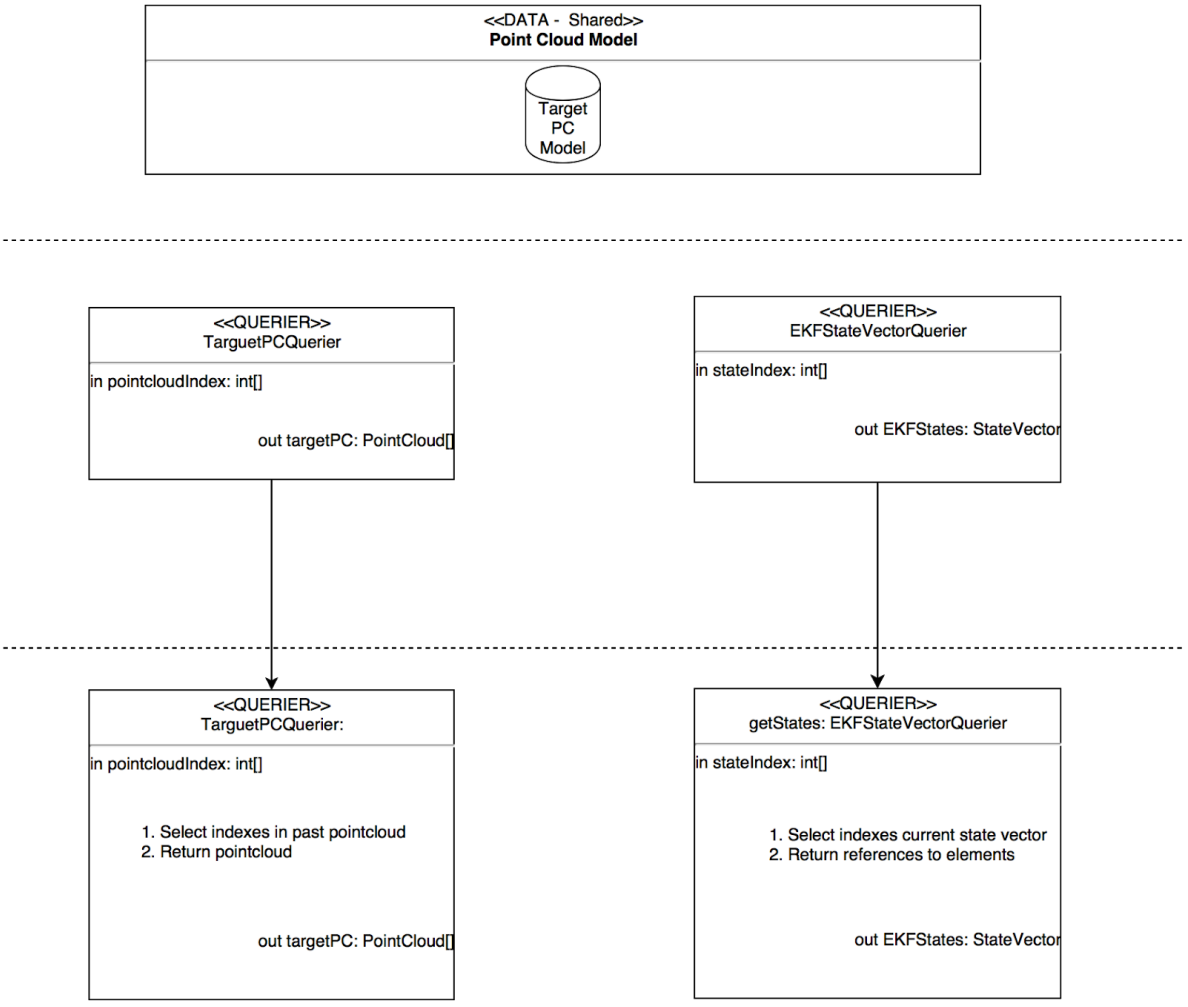


Figure 47: ICP Point Cloud Matching Data Product Management

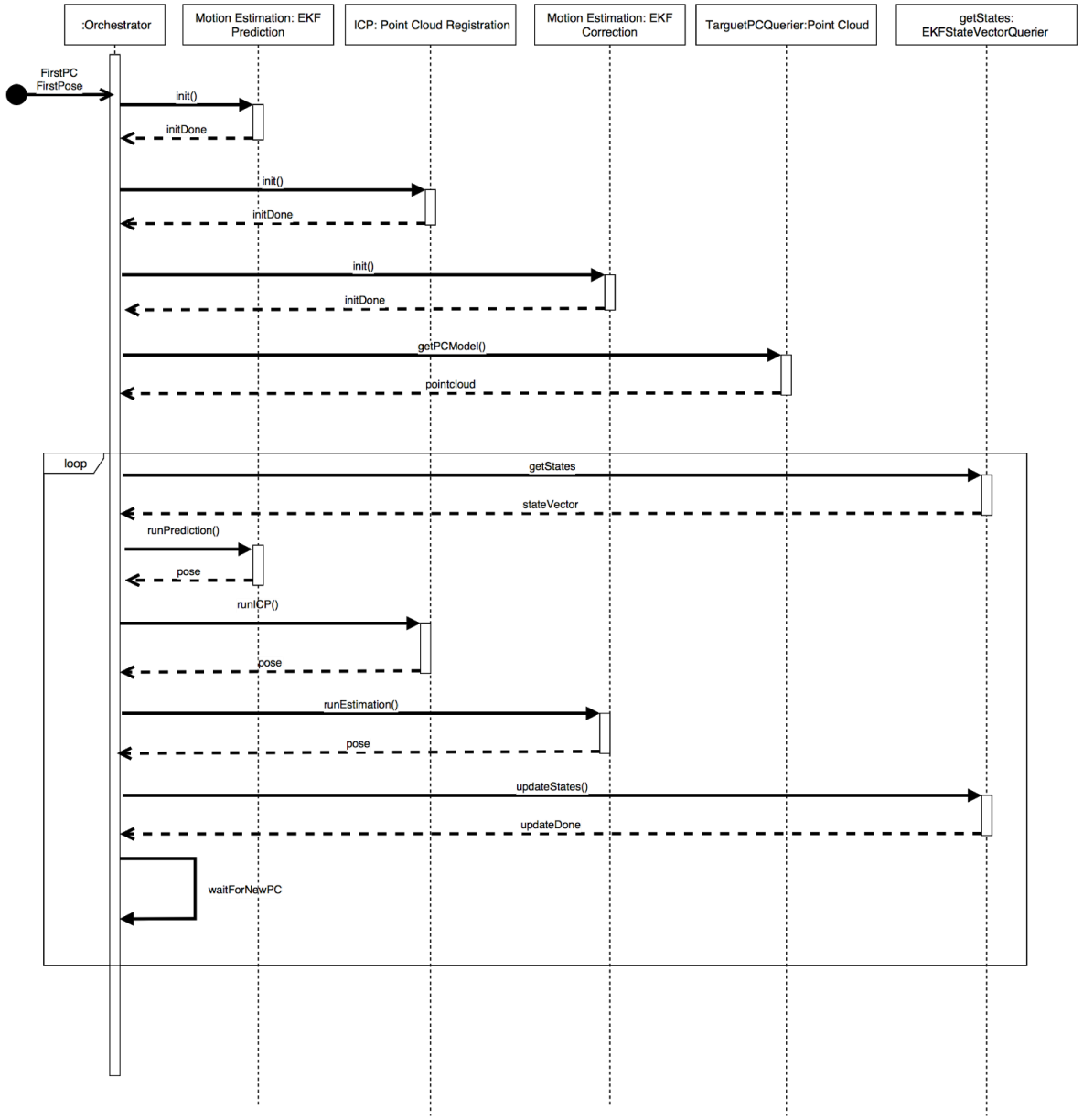


Figure 48: ICP Point Cloud Matching Control Description

4.1.8.1.1 DFPC Expected Performance

Since this is a similar, yet simplified implementation of the SHOT 3D Feature Matching flavor of this DFPC (see Section 4.1.8.2) working with a complete, dense point cloud, we expect a lower general accuracy and robustness, but a possible increase in execution rate. We can thus expect euclidean distance to be below 5% of R, where R is the maximum operational distance of the camera, and a final angular distance to be below 10°.

4.1.8.2 Flavor 2: SHOT 3D Feature Matching and EKF

This DFPC is a modified implementation of the 3D reconstruction DFPC specified for the Orbital Track. This DFPC assumes that an environment point cloud is already available as a computational results of other DFPCs, and directly performs a 3D descriptor matching on the provided target point cloud.

A set of three-dimensional keypoints are chosen from both the scene and the model by picking individual points from the cloud separated by a given sampling radius. Normals are calculated for these keypoints relative to nearby points so that each keypoint has a repeatable orientation. The keypoints are then associated with a three-dimensional descriptor. An example 3d descriptor is SHOT (Signature of Histograms of OrienTations) descriptors [Salti et al., 2014]. SHOT descriptors are calculated by grouping together a set of local histograms over the volumes about the keypoint, where this volume is divided into by angle into 32 spatial bins. Point counts from the local histograms are binned as a cosine function of the angle between the point normal within the corresponding part of the structure and the feature point normal. This has the beneficial effects of creating a general rotational invariance since angles are relative to local normals, accumulating points into different bins as a result of small differences in relative directions, and creating a coarse partitioning that can be calculated fast with small cardinality. Additionally, the BOrder Aware Repeatable Directions algorithm for local reference frame estimation (BOARD) is used to calculate local reference frames for each three-dimensional SHOT descriptor [Petrelli and Di Stefano, 2011] to make them independent of global coordinates for rotation and translation invariance.

Comparing the scene keypoint descriptors with the model keypoint descriptors to find good correspondence matches is done with a matching algorithm. An example matching algorithm is the FLANN search on a k-dimensional tree (k-d tree) structure, similar to the FLANN matching of image keypoints. The results of a matching algorithm is a set of local matching candidates, that need to be further verified globally upon the overall model, by means of a correspondence grouping algorithm. One example of correspondence grouping algorithm is Hough voting that makes recognition of shapes more robust to partial occlusion and clutter [Tombari and Di Stefano, 2010]. When Hough voting is used, evidence of a particular pose and instance of the model in the scene is initialized before voting by obtaining the vector between a unique reference point and each model feature point and transforming it into local coordinates by the transformation matrix from the local x-y-z reference frame unit vectors. This precomputation can be done offline for the model in advance.

For online pose estimation, Hough voting is performed by each scene feature that has been found by FLANN matching to correspond with a model feature, casting a vote for the position of the reference point in the scene. The transformation that makes these points line up can then be transformed into global coordinates. The votes cast are thresholded to find the most likely instance of the model in the scene, although multiple peaks in the Hough space are fairly common and can indicate multiple possibilities for model instances. Due to

the statistical nature of Hough voting, it is possible to recognize partially-occluded or noisy model instances, though accuracy may be lower.

DFPC Inputs :

- Environment point cloud, representing an area of the world visible by the platform,
- Model point cloud file if model-based ID required (PCD, PLY, A3D),
- [Parameter] Additional optional parameters.

DFPC Outputs:

- Estimated orientation and matching of model in scene.

The DFPC will be composed of the following DFNs, with various flavor options available in each:

- Point cloud descriptor extraction: to find keypoints in scene and model point clouds,
    - Harris 3D detector + SHOT feature descriptors,
- Descriptor matching: to match descriptors between scene and model,
    - FLANN descriptor matcher (alternately: RANSAC or brute force matcher)
- Correspondence grouping: to find correspondences between scene and model,
    - Hough voting (alternately: ICP or RANSAC)
- Pose estimation calculation,
    - Movement averaging (alternately: Extended or Cubature Kalman filter)

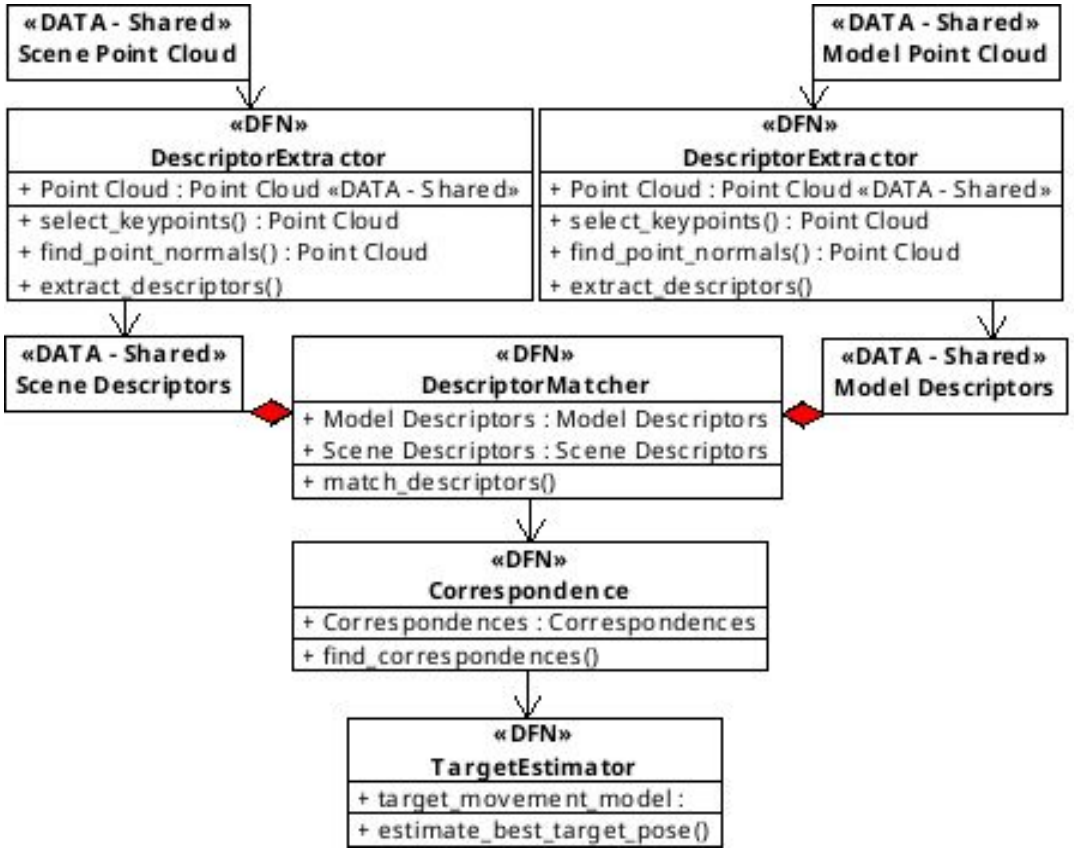


Figure 49: details the DFN component structure inside the DFPC.

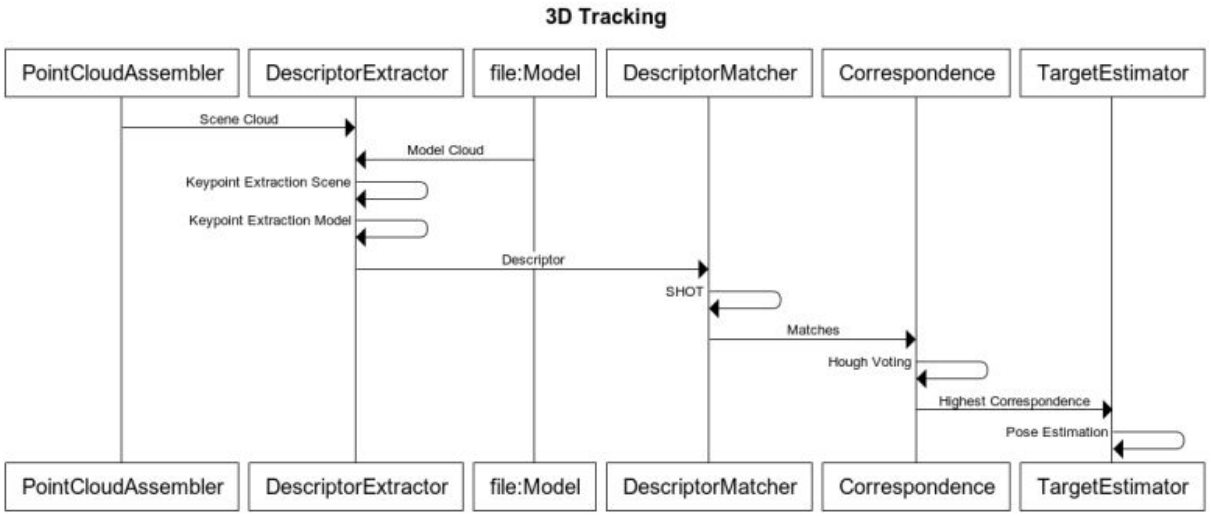


Figure 50: 3D target tracking timing diagram

4.1.8.2.1 Algorithm Description

A set of three-dimensional keypoints are chosen from both the scene and the model by picking individual points from the cloud separated by a given sampling radius. Normals are calculated for these keypoints relative to nearby points so that each keypoint has a repeatable orientation. The keypoints are then associated with three-dimensional SHOT (Signature of Histograms of OrienTations) descriptors. SHOT descriptors are calculated by grouping together a set of local histograms over the volumes about the keypoint, where this volume is divided into by angle into 32 spatial bins. Point counts from the local histograms are binned as a cosine function of the angle between the point normal within the corresponding part of the structure and the feature point normal. This has the beneficial effects of creating a general rotational invariance since angles are relative to local normals, accumulating points into different bins as a result of small differences in relative directions, and creating a coarse partitioning that can be calculated fast with small cardinality. This method generalizes to the descriptor

$$D(p)=\left(\bigcup_{i=1}^{m}\right)SH^{i}_{g,f}(p) \quad (20)$$

which can also be used for color texture descriptions.

Comparing the scene keypoint descriptors with the model keypoint descriptors to find good correspondence matches is done using a FLANN search on a k-dimensional tree (k-d tree) structure, similarly to the matching of image keypoints. Additionally, the BOrder Aware Repeatable Directions algorithm for local reference frame estimation (BOARD) is used to calculate local reference frames for each three-dimensional SHOT descriptor to make them independent of global coordinates for rotation and translation invariance.

Once a set of nearest correspondences and local reference frames is found, clustering of correspondences is performed by pre-computed Hough voting to make recognition of shapes more robust to partial occlusion and clutter.

Evidence of a particular pose and instance of the model in the scene is initialized before voting by obtaining the vector between a unique reference point C^M and each model feature point F_i^M and transforming it into local coordinates by the transformation matrix $R^M_{GL}=[L^M_{i,x}, L^M_{i,y}, L^M_{i,z}]^T$ from the local x-y-z reference frame unit vectors $L^M_{i,x}$, $L^M_{i,y}$, and $L^M_{i,z}$. This precomputation can be done offline for the model in advance and is performed by calculating for each feature a vector

$$V^M_{i,L}=[L^M_{i,x},L^M_{i,y},L^M_{i,z}]\cdot(C^M-F^M_i). \quad (21)$$

For online pose estimation, Hough voting is performed by each scene feature F^S_J that has been found by FLANN matching to correspond with a model feature F^M_I, casting a vote for the position of the reference point C^M in the scene. The transformation $R^M S_L$ that makes these points line up can then be transformed into global coordinates with the scene reference frame unit vectors, scene reference point F^S_j and scene feature vector $V^S_{i,L}$ as

$$V^S_{i,G}=[L^S_{j,x},L^S_{j,y},L^S_{j,z}]\cdot V^S_{i,L}+F^S_j. \quad (22)$$

The votes cast by $V^s_{i,g}$ are thresholded to find the most likely instance of the model in the scene, although multiple peaks in the Hough space are fairly common and can indicate multiple possibilities for model instances. Due to the statistical nature of Hough voting, it is possible to recognize partially-occluded or noisy model instances, though accuracy may be lower.

4.1.8.2.2 DFPC Expected Performance

Four different tests were performed in the laboratory on image sequences produced from robotic movement of a camera in equidistant arcs about a 1U CubeSat engineering model to obtain the scene and model shown in Figure 51.

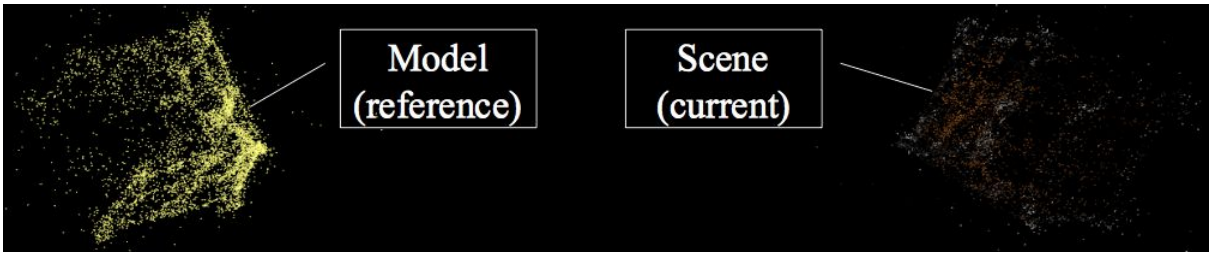


Figure 51: High-resolution CubeSat model (left) and scene from sensors (right)

The SHOT descriptor radius and cluster size parameters were varied to test the relationship of these variables to the resulting matches. Table 4 shows results from these tests.

Table 4: Parameters for 3D model-based tracking

Test Number	Number of Images	Number of scene features	Number of keypoints	Number of matches	Descriptor Radius (m)	Cluster Size (m)
1	220	5584	167	63	0.05	0.1
2	220	5584	632	594	0.1	0.5
3	32	1816	77	28	0.05	0.1

4	32	1816	77	70	0.1	0.5

The process of reconstruction and tracking was profiled running on the ARM core of a Xilinx Zynq Z7020 SoC microcontroller (667MHz ARM-Cortex A9). Table 5 shows timing results (in seconds) for the 3D model-based identification and tracking process[2]. It can be seen from this that the majority of time is spent on keypoint production and FLANN search during the identification and tracking process.

Table 5: Timing Results for 3D model-based tracking

Test	Model Normals	Scene Normals	Model Sampling	Scene Sampling	Model Keypoints	Scene Keypoints	FLANN Search	Clustering	TOTAL (s)
1	0.17	0.15	0.027	0.020	1.26	0.84	107.7	0.92	112.1
2	0.17	0.15	0.029	0.024	3.37	2.19	118.0	2.00	127.2
3	0.17	0.043	0.031	0.0083	3.31	0.37	42.5	0.63	48.4
4	0.17	0.041	0.031	0.0078	3.31	0.37	42.6	1.36	49.1

The accuracy of ego-motion estimation (effectively the tracking of the relative position of the target) during the tracking process was additionally profiled using another test using a 3U CubeSat engineering model, shown in Figure 52. Figure 53 shows plots of the pose estimation accuracy in translation and Figure 54 in rotation. The total RMS error in translation was 7mm in X, 8mm in Y, and 7mm in Z. The total RMS error in rotation was 0.14rad about X, 0.11rad about Y, and 0.19rad about Z.

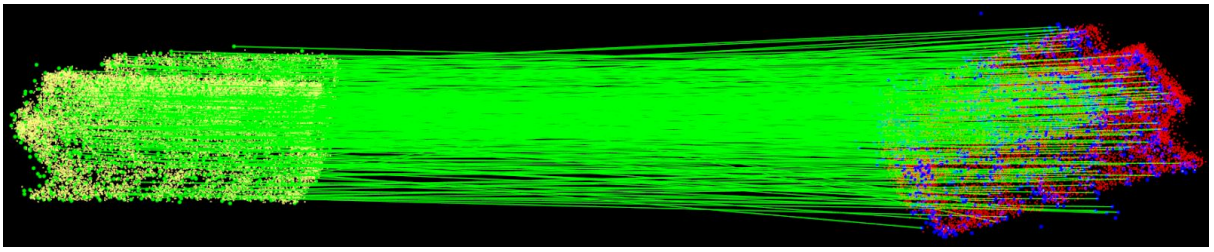


Figure 52: 3U CubeSat model (left) and match with scene (right)

[2] M.A. Post, J. Li, C. Clark, X. Yan. “Visual Pose Estimation System for Autonomous Rendezvous of Spacecraft”. ESA Astra 2015: 13th Symposium on Advanced Space Technologies in Robotics and Automation. ESA/ESTEC, Noordwijk, the Netherlands, 11-13 May 2015.

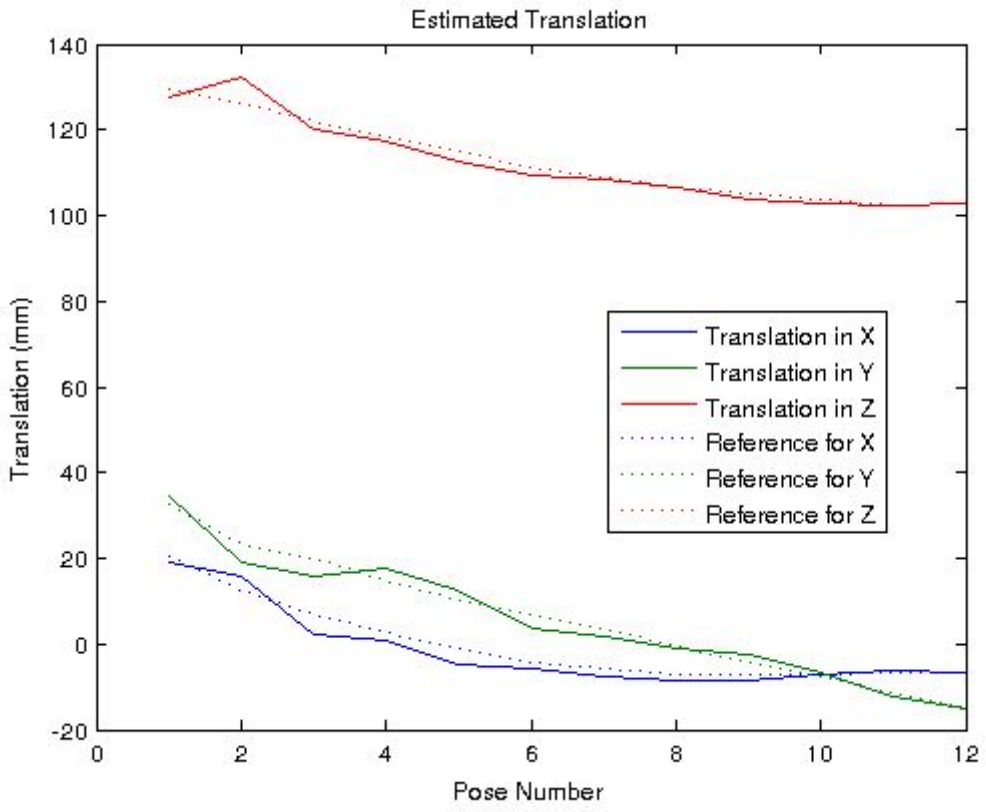


Figure 53: Tracking accuracy of 3U CubeSat model in translation

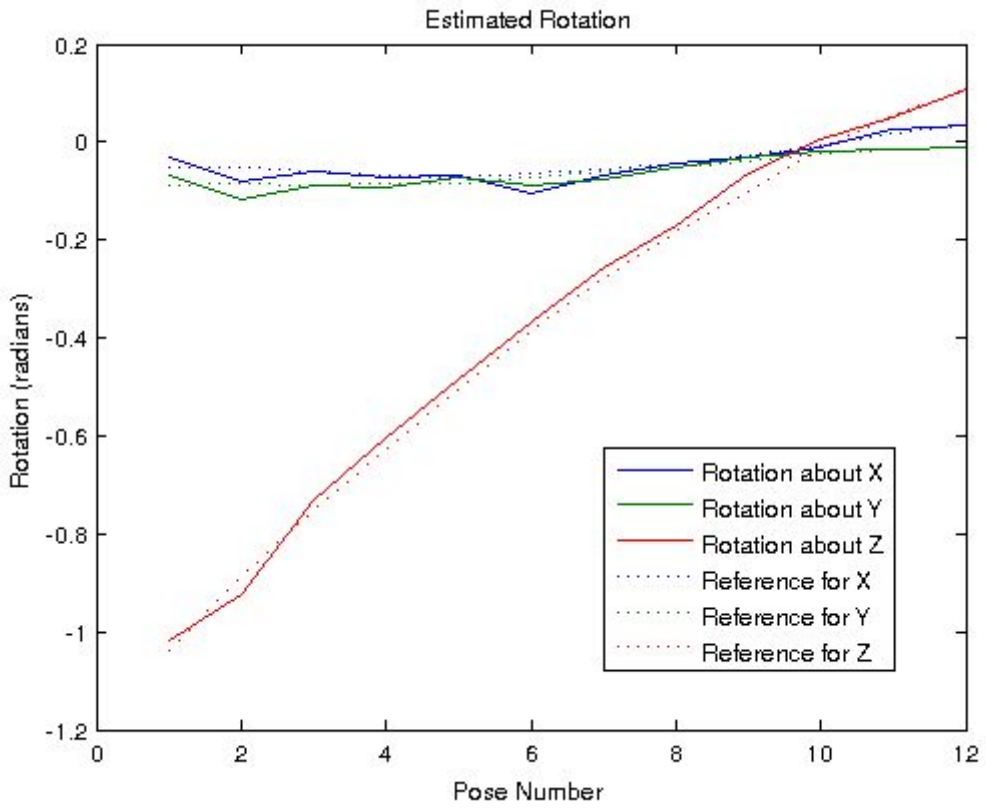


Figure 54: Tracking accuracy of 3U CubeSat model in rotation

Some initial estimates of pose estimation accuracy under partial occlusion of a 3U CubeSat target were also performed. Shadowing the target by 25% resulted in an additional ~1% error in translation and ~2% error in rotation, shadowing the target by 50% resulted in an additional ~7% error in translation and 3% error in rotation, and with 75% shadowing no correspondence with the model was found.

From the testing results given, initial parameters for the DFPC are suggested as follows:

- Descriptor Radius and Cluster Size should be a fraction (1%-10%) of the size of the object to be detected
- Descriptor Radius and Cluster Size should be the same order of magnitude
- Descriptor radius may be tuned to improve the speed of the descriptor selection
- Cluster size may be tuned to increase the speed of the matching process
- The model point density should be no more than one order of magnitude different from the scene point density (subsampling is possible)

Expected performance in general cases for this DFPC are shown below.

Test	Reference	Output	Measure

Object Detection	Ground truth object pose	Estimated object pose	euclidean distance and angular distance less than 1% of R, where R is the maximum operational distance of the camera.

4.1.9 DFPC : Absolute Localisation

This DFPC responds to the absolute localisation from orbital data objective expressed in the implementation scenario :

- RI-INFUSE-LONG-TRAVERSE-DEM.

This processing compound provides a pose estimate of the rover in the orbital image frame, using orbital imagery, a DEM generated by the rover and an orthophoto also generated by the rover. The dual type of data is used to ensure good performances in rocky or outcrop areas. In presence of outstanding rocks, matching can be performed through rock extraction in the DEM. Whereas, in presence of outcrops a keypoint extraction and matching procedure can also be applied.

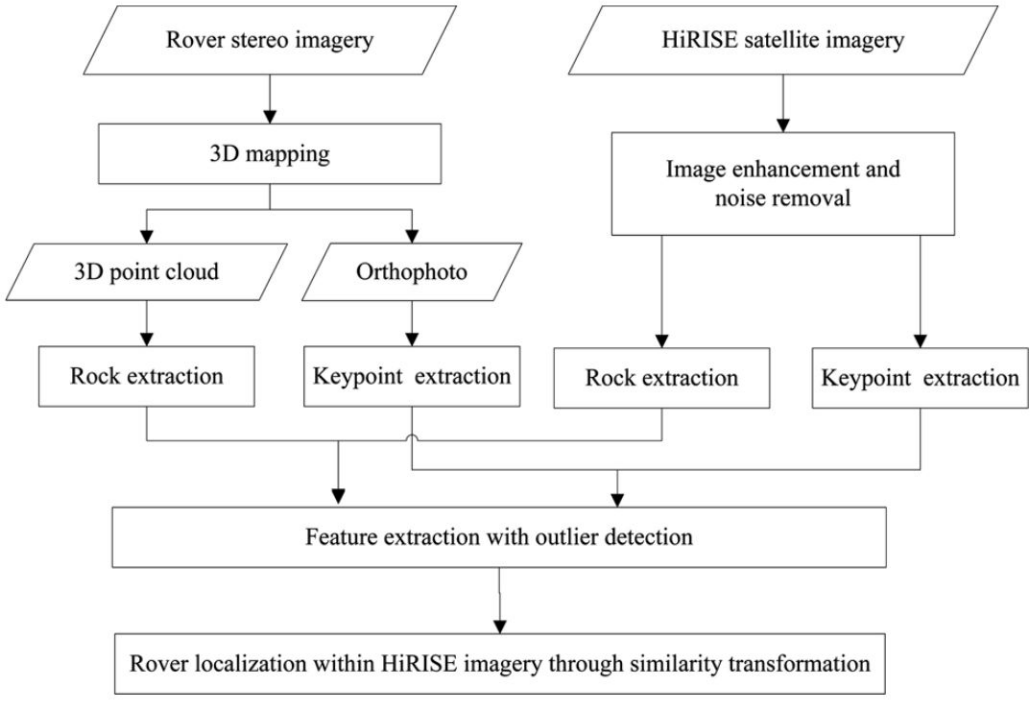


Figure 51: Diagram of Absolute Localization DFPC. HiRISE represents the orbital imagery

The DPFC's inputs are the following:

- Point cloud for rover DEM generation, and associated metadata.
- Orthoimage produced by the rover, and associated metadata.
- Orbital image and associated metadata.

The DFPC's output is the following:

- Pose estimate in the orbiter map frame of reference.

The DFPC is composed of the following DFNs:

- PCTransform: Enables the transformation of a given point cloud in a given frame of reference, of which we know the transform. This is used to transform the LIDAR point cloud in the fused map frame of reference,
- Rasterizer: Transforms the point cloud into a Rover map by projecting points. Each layer of the raster contains information on the Rover map,
- FeatureExtraction: Extracts relevant points of interest in both the rover map and the orbiter map. Those points of interest are related to peaks of altitude in the elevation map or to visual features in the case of the orthoimage,
- FeatureMatching: Matches features extracted from the rover map and the orbiter map,
- PoseEstimator: From the feature-matching DFN, this DFN computes the pose of the rover in the orbiter map frame of reference.

The orbiter input map is considered to have the characteristics of the best maps currently produced by the Hirise instrument on-board Mars Reconnaissance Orbiter: DEM resolution of the order of 1.0m and elevation precision of the order of 0.1 m, and orthoimage resolution in the visible spectrum of the order of 0.25m.

The following figure details the DFN component structure inside the DFPC.

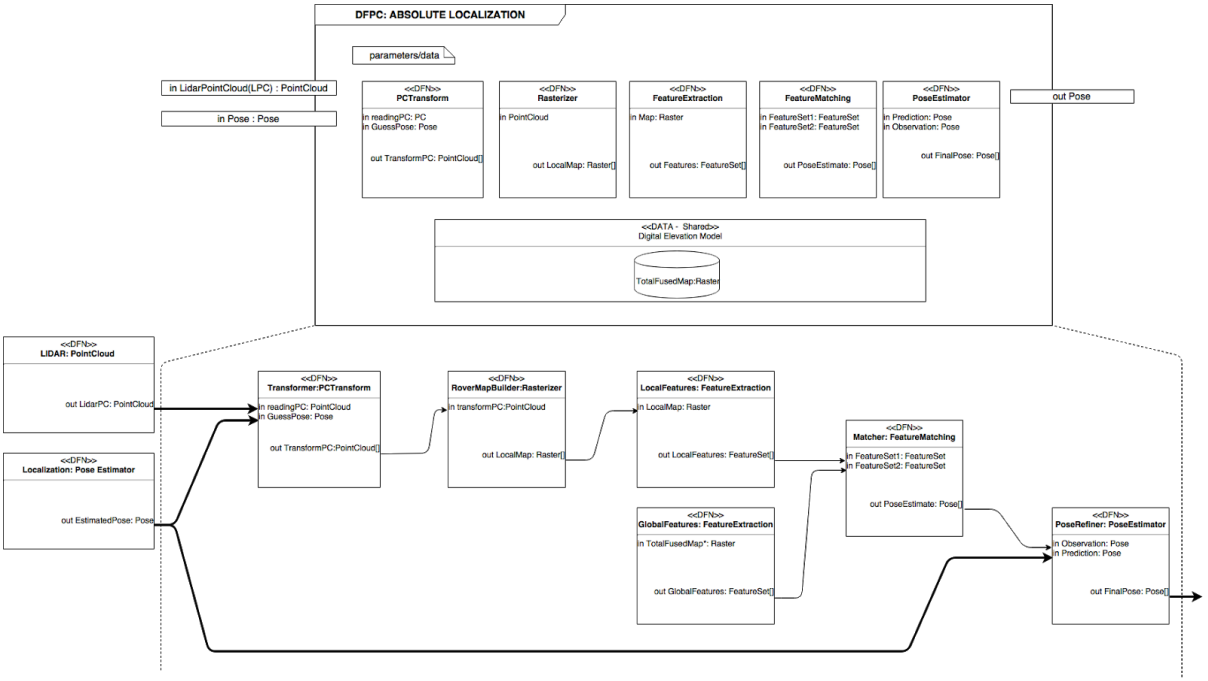


Figure 52: Absolute Localization DFPC Data Flow Description

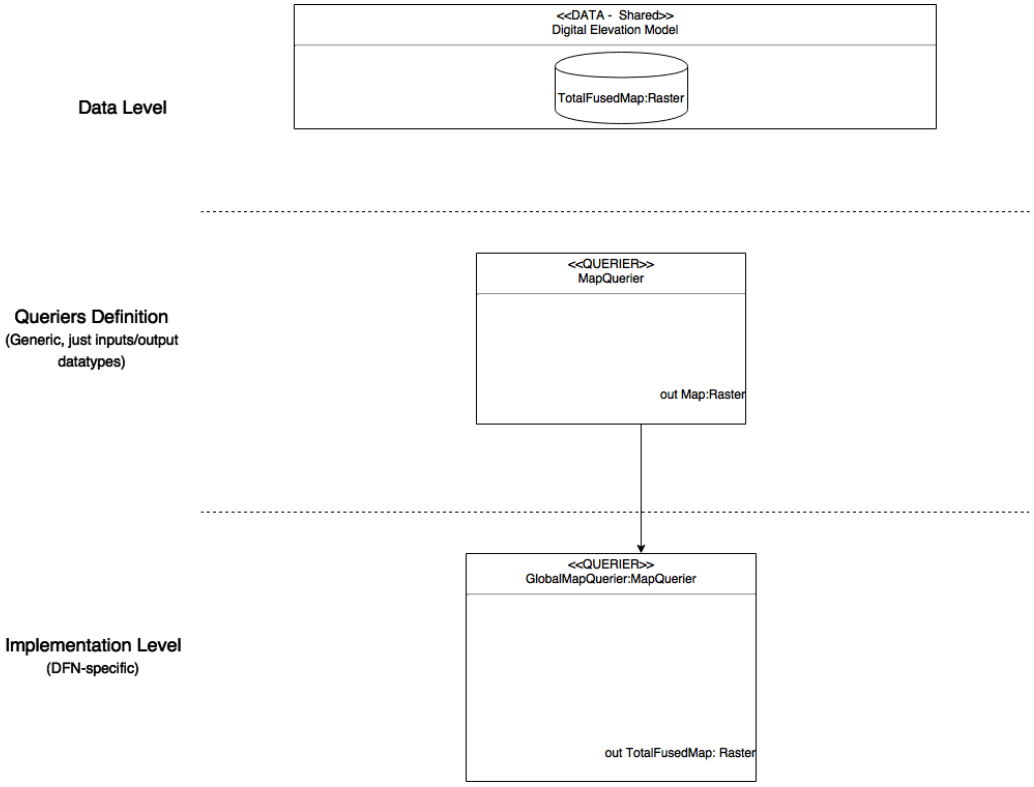


Figure 53: Absolute Localization DFPC Data Product Management

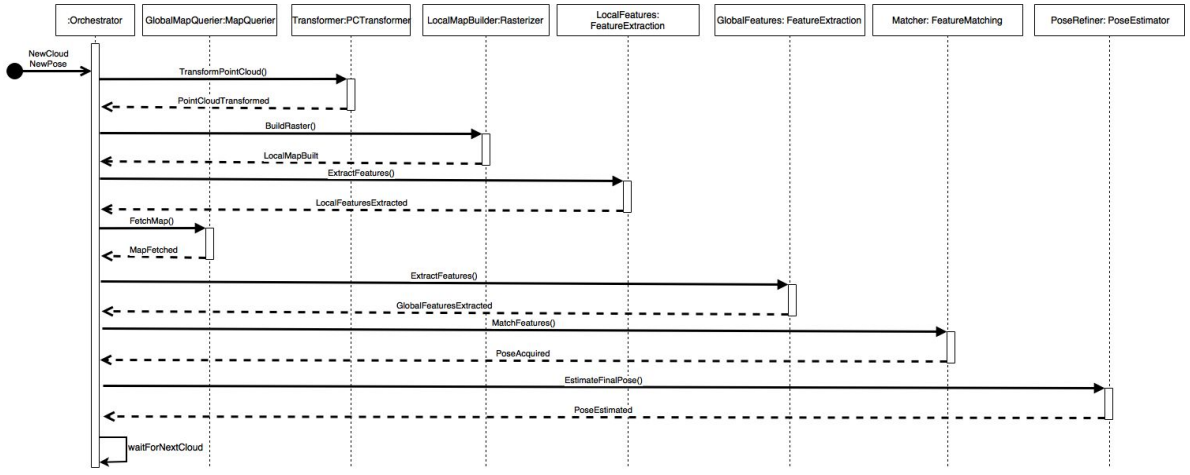


Figure 54: Absolute Localisation Control Description

4.1.9.1 DFPC Expected Performance

The proposed approach from [DI2011] shows tests performed on MER data for localization near the respective (MER-A and MER-B) landing sites. It presents a RMS error of less than one pixel on average, which are sampled at 0.25m resolution. However, the algorithm uses SIFT features, which are far more reliable in terms of scale invariance, while it is not clear yet if an alternative keypoint may yield comparable performances. In terms of computation time, a time between 1 and 3 minutes should be expected. Note that this is an acceptable processing speed as the method is not expected to be run onboard in real time.
The target accuracy is set at the equivalent in meters of 1-2 pixel, depending on the resolution of the orthoimage.

[DI2011] K. Di, Z. Liu, and Z. Yue, “Mars Rover Localization based on Feature Matching between Ground and Orbital Imagery,” Photogrammetric engineering and remote sensing, vol. 77, no. 8, 2011.

4.1.10 DFPC: DEM Building

This DFPC responds to the DEM building objective expressed in the implementation scenario :

- RI-INFUSE-LONG-TRAVERSE-DEM.

This processing compound builds a Digital Elevation Map (DEM) from point clouds provided by LIDAR or stereoscopic imaging sensors.

The DPFC’s inputs are the following:

- LIDAR or stereo point cloud, and associated metadata
- Pose estimate from the Data Product Manager (DPM), and associated metadata

The DFPC’s output is the following:

- Fused Rover Map: the DEM built from the start of the rover trajectory.

The DFPC is composed of the following DFNs:

- PCTransform: Enables the transformation of a given point cloud in a given frame of reference, of which we know the transform. This is used to transform the LIDAR point cloud in the fused map frame of reference,
- Rasterizer: This DFN transforms the point cloud into a Rover map by projecting points. Each layer of the raster contains information on the Rover map.
- Map builder: This DFN integrates the Rover Map inside the Fused Rover map
- The following figure details the DFN component structure inside the DFPC.

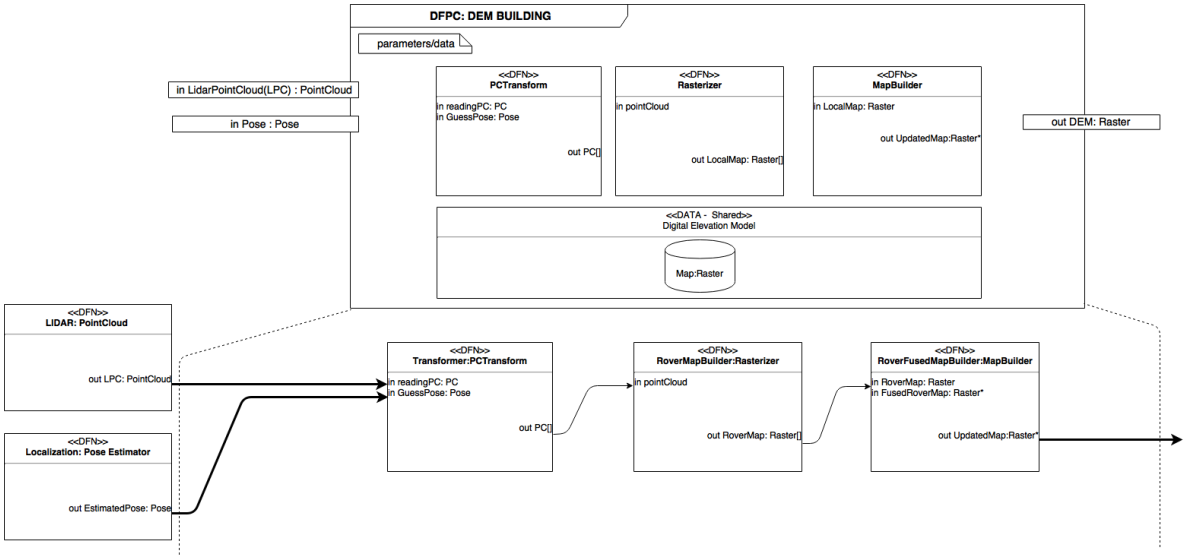


Figure 55: DEM Building Data Flow Description

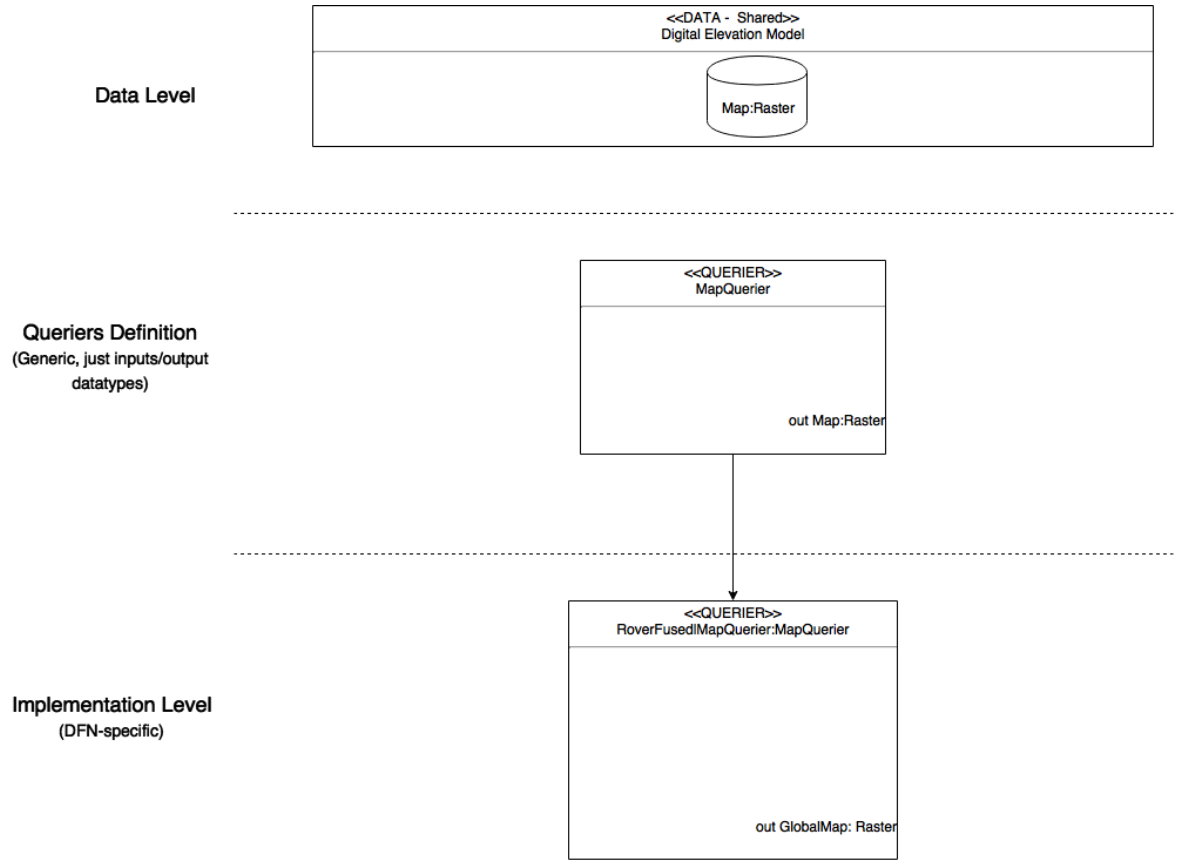


Figure 56: DEM building DFPC Data Product Management

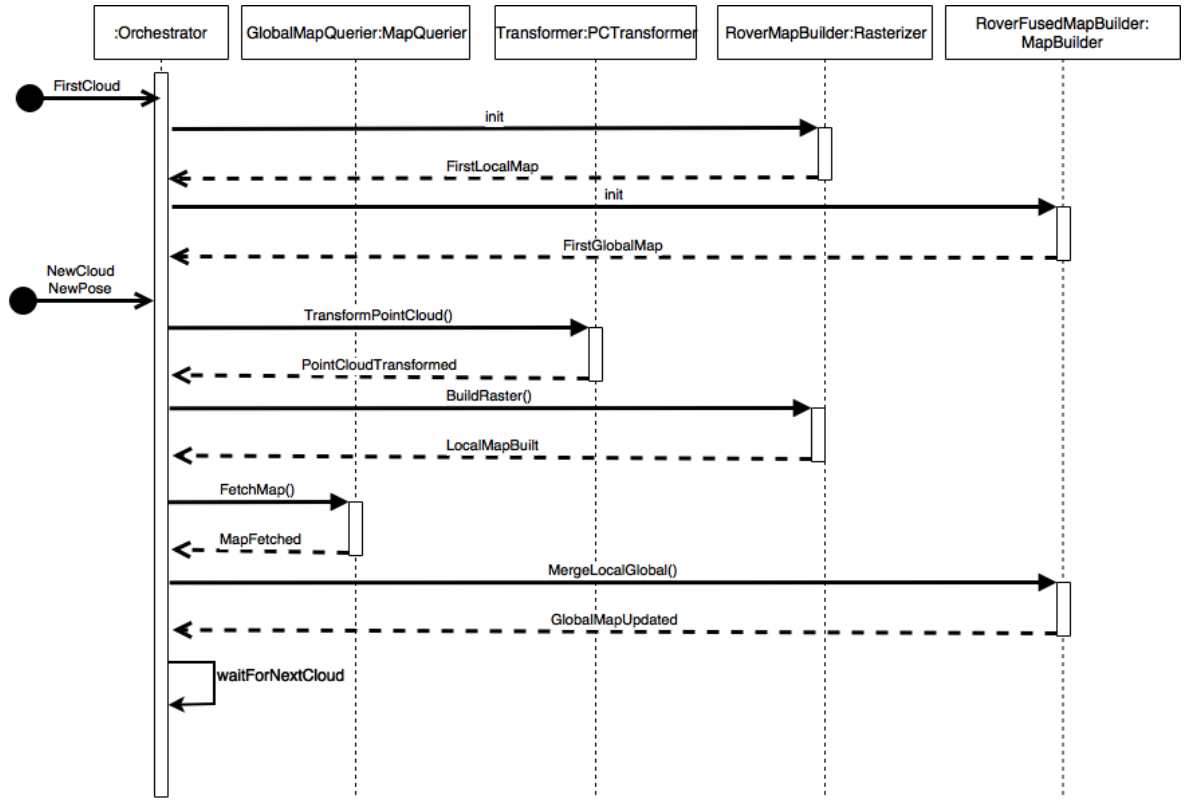


Figure 57: DEM building DFPC Control Description

4.1.10.1 DFPC Expected Performance

As DEM does not feature stochastic processes, the introduced error is only dependent from the sensor noise and an eventual downsampling for memory/computation constraints. Given a target platform composed as follows:

CPU : Intel Core i7-6700HQ @ 2.60GHz

RAM : 16GB DDR4

the expected run time and memory usage for delivering one scan DEM of 120x120 meters with a resolution of 0.1m per cell are:

DFPC	Input type	Single thread – Memory in MB	Single thread – CPU time in ms
DEM Building	Posestamped point clouds	18	<500

Of course the memory usage can be affected by many factors such as size and resolution of the map, data encoding and, number of exported layers.

4.1.11 DFPC: Lidar SLAM

This DFPC responds to the LIDAR mapping objective expressed in RI-INFUSE-LONG-TRAVERSE-LOC, Use Case 3: LIDAR SLAM.

This processing compound simultaneously builds a environment model composed of a series of LIDAR point clouds and provides pose estimates for the rover.

The DPFC's inputs are the following:

- LIDAR point cloud, and associated metadata,
- Pose estimate from wheel odometry (or the DPM) and associated metadata.

The DFPC's outputs are the following:

- Pose estimate of the rover in the pose-graph frame of reference,
- Pose graph: an environment model containing keyframes associated to successive poses.

The DFPC is composed of the following DFNs:

- PCMatcher: Matches two point clouds and gives a percentage of overlap and the transformation between the point clouds. The most common PCMatcher is ICP. It

will be used two times: once to decide if the current point cloud should be added to the pose graph, and once to decide if the current point cloud calls for a loop closure.

- MapBuilder: This DFN is in charge of adding keyframes inside the pose graph, depending on different criteria. It is also used twice: once in case a keyframe is added in the pose graph, and once when a loop is closed.
- MapOptimizer: This DFN is in charge of optimizing the graph when a loop is closed by the rover.

The following figure details the DFN component structure inside the DFPC.

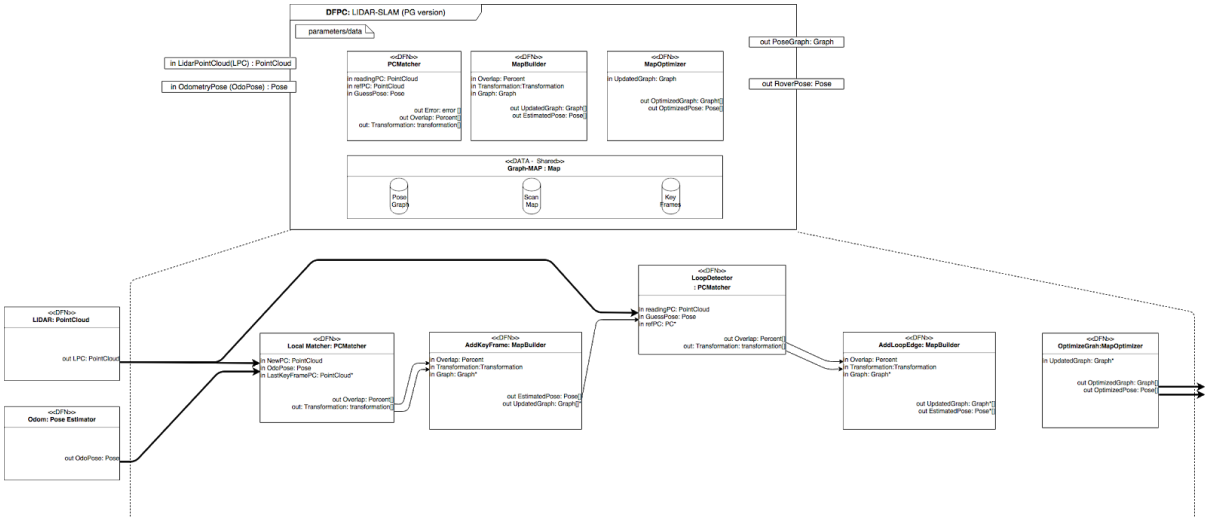


Figure 58: LIDAR-SLAM Data Flow Description

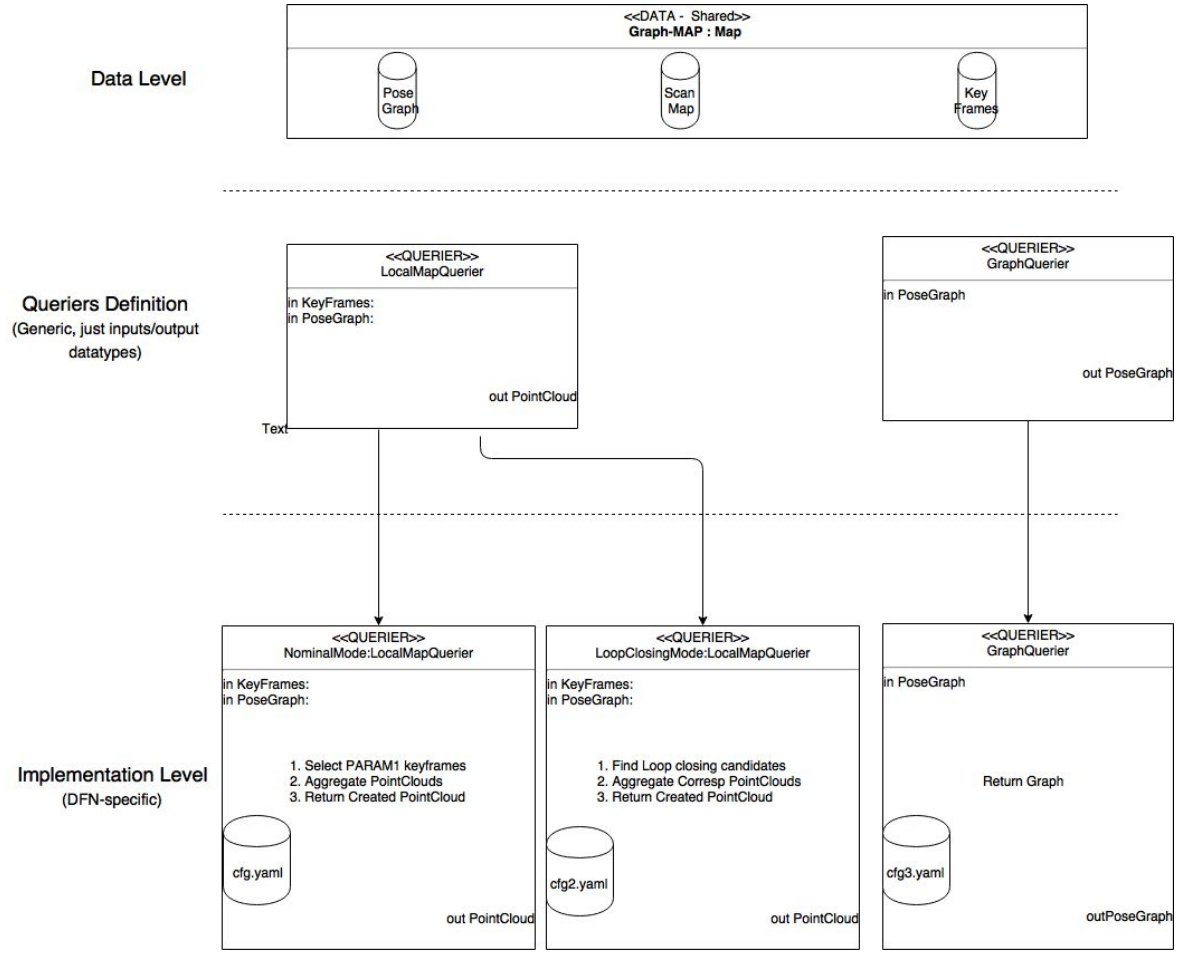


Figure 59: LIDAR-SLAM-DFPC Data Product Management

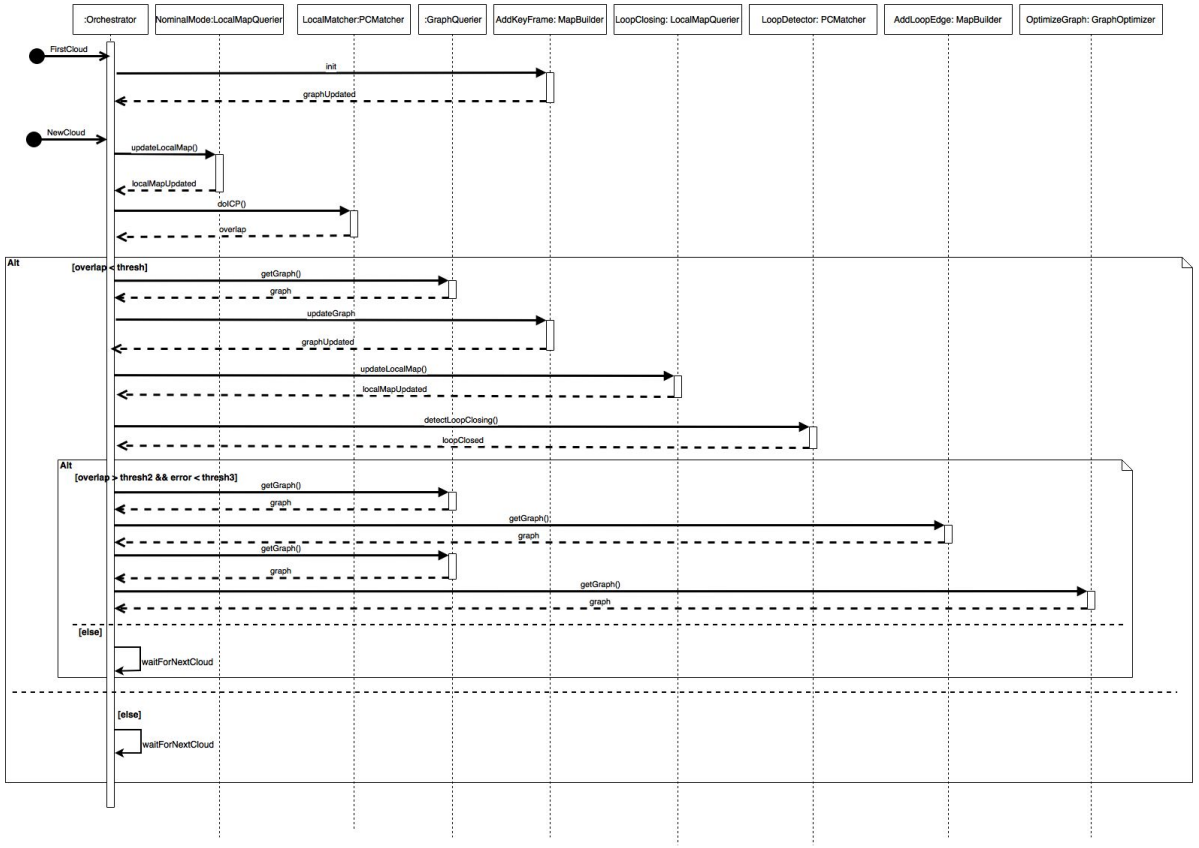


Figure 60: LIDAR-SLAM DFPC Control Description

4.1.11.1 DFPC Expected Performance

A localisation error is applicable to LIDAR-based SLAM. From previous benchmarks we can reasonably expect a translation error which is set around 2% of the travelled distance (path length), with a standard deviation of +/- 1% [MEND2011]. Nonetheless, the environment shall allow the localisation process, i.e. being sufficiently rich in features such as rocks, boulder, etc.
The target rotation error is set at 0.04 deg/m.

[MEND2011] Mendes, Ellon Paiva. "Study on the use of vision and laser range sensors with graphical models for the slam problem." PhD diss., Institut National des Sciences Appliquées de Toulouse (INSA Toulouse), 2017.

4.1.12 DFPC : Lidar Map-based Localisation

This DFPC responds to the LIDAR mapping objective expressed in RI-INFUSE-LONG-TRAVERSE-LOC, Use Case 3: LIDAR SLAM.

This processing compound provides a pose estimate for the rover, considering a previously built LIDAR Pose-GRAPH (with the help of LIDAR-SLAM, described above).

The DPFC's inputs are the following:

- LIDAR point cloud, and associated metadata
- Pose estimate from the Odometry (or the DPM) and associated meta data

The DFPC's outputs are the following:

- Pose estimate of the rover in the pose-graph frame of reference

The DFPC is composed of the following DFNs:

- PCMatcher: Matches two point clouds and gives a percentage of overlap and the transformation between them. The most common PCMatcher is the ICP,
- PoseEstimator: Once the matching is performed between the LIDAR observation and the graph, a pose is estimated in the Graph frame of reference.

The following figure details the DFN component structure inside the DFPC.

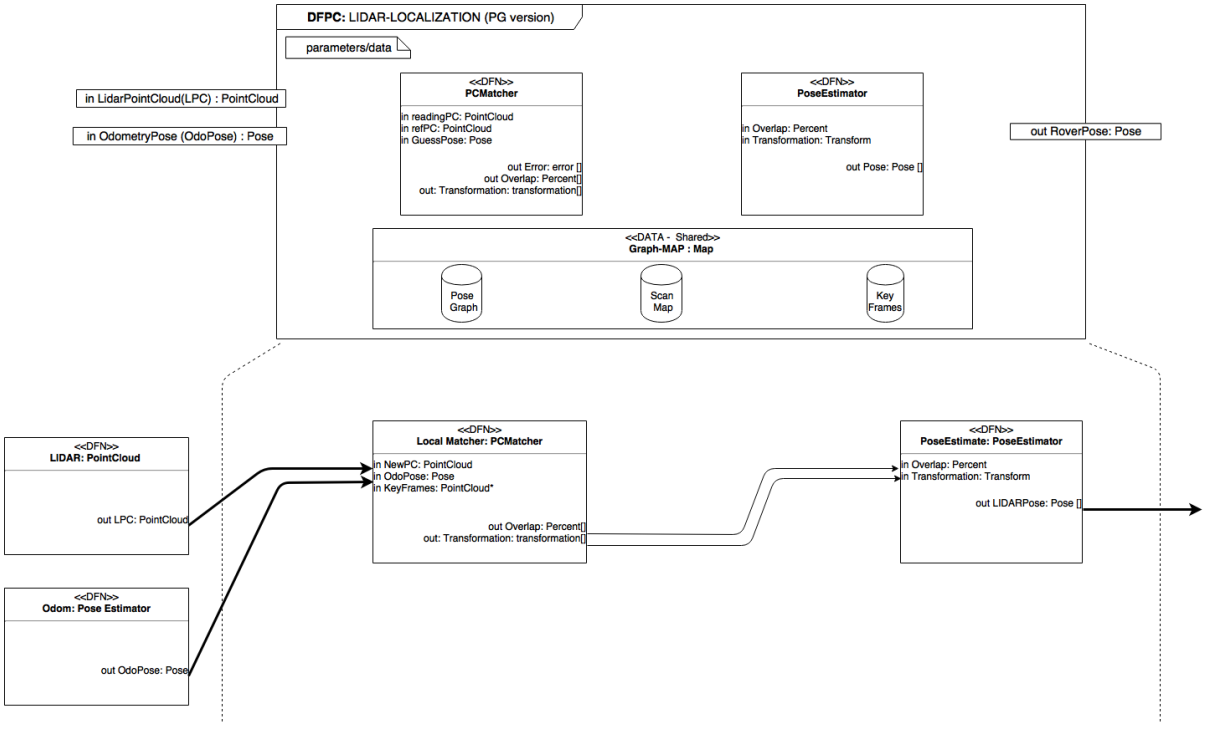


Figure 61: LIDAR Map-based Localization Data Flow Description

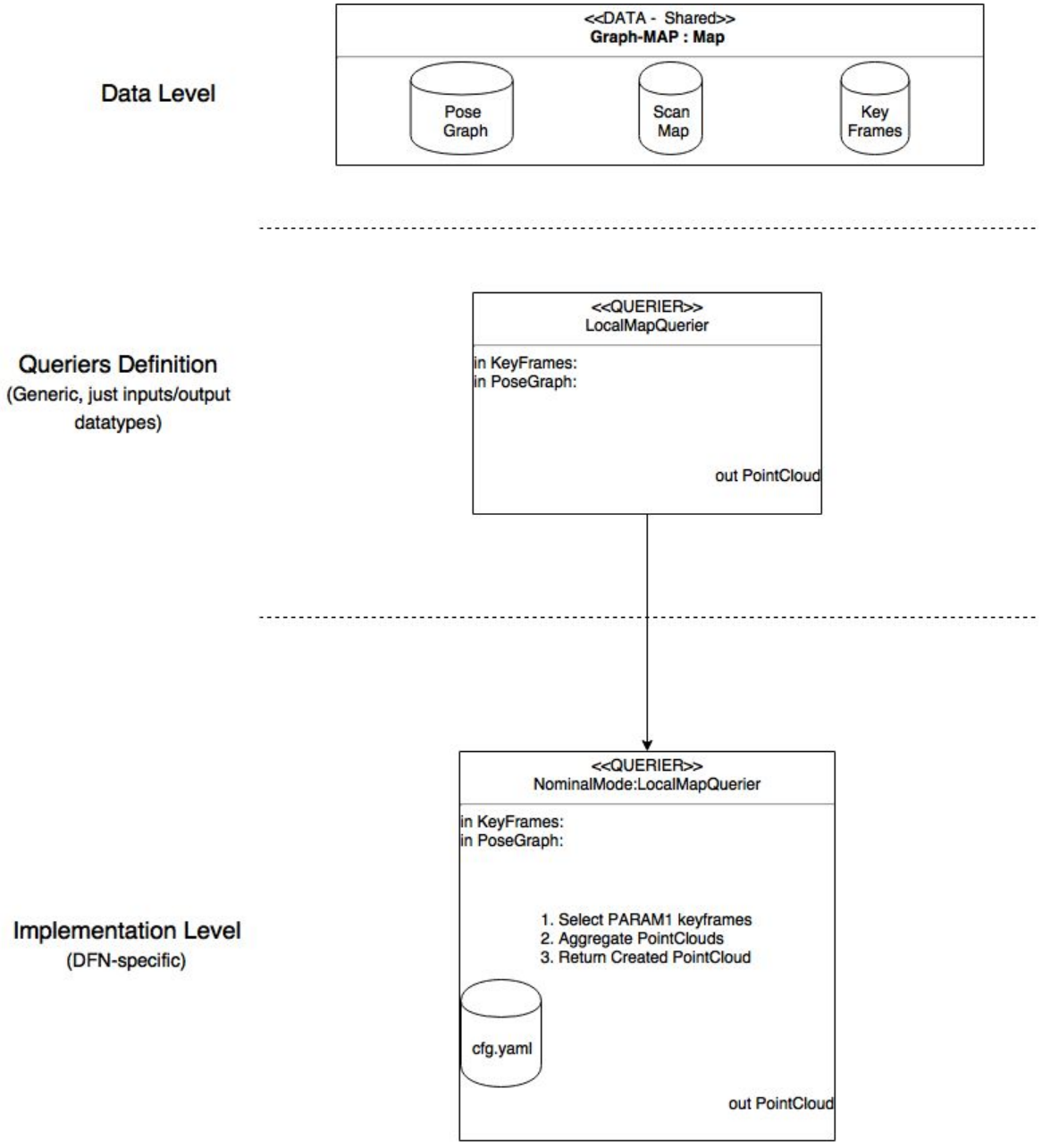


Figure 62: LIDAR map-based localization Data Product Management

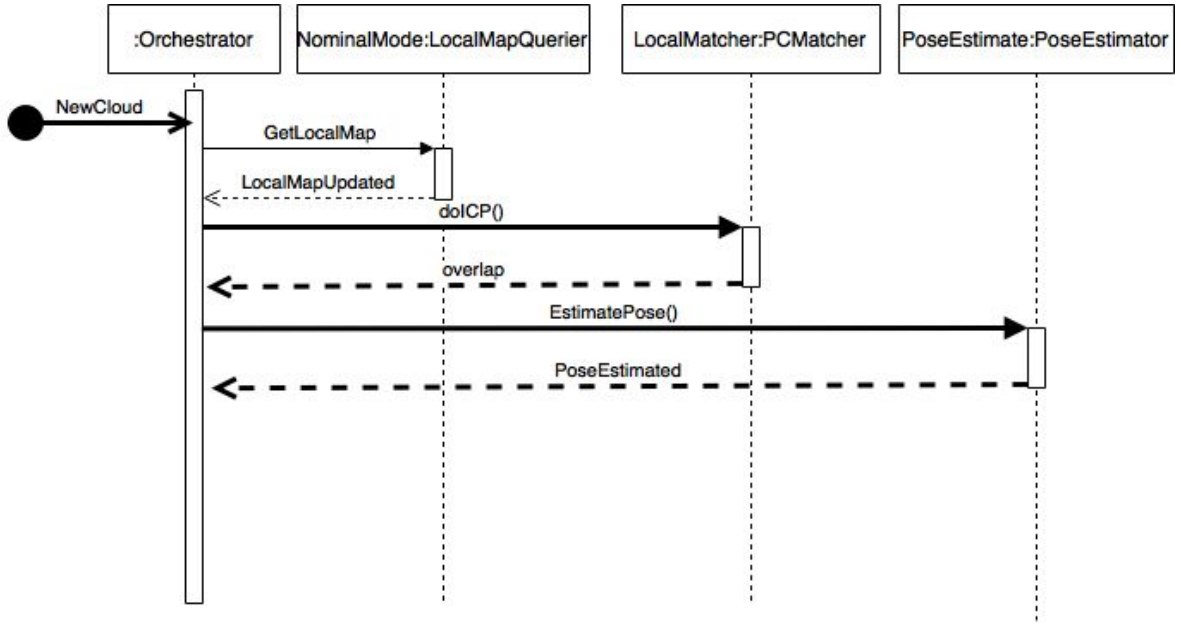


Figure 63: LIDAR Map-based Localisation Control Description

4.1.12.1 DFPC Expected Performance

The error on the localisation produced by this DFPC has two sources:

- The precision of the localisation with respect to the Lidar Pose-Graph map
- The precision of the localisation of the Lidar Pose-Graph map itself.

The precision of the localisation with respect to the Lidar Pose-Graph map is expected to be of the order of a few cm in translation, and below 0.5 degree in orientation. This precision depend strongly on the type of environment that is mapped.

The precision of the Lidar Pose-Graph map inherits the precision of the SLAM process that builds it: the map localisation precision depends on its distance with respect to the starting point - see expected performance of the LIDAR SLAM DFPC.

4.1.13 DFPC: Navigation Map Building

This DFPC responds to the reference implementation scenarios described in :

- RI-INFUSE-LONG-TRAVERSE-LOC
  - Use Case 1 : Visual Odometry,
  - Use Case 2 : Visual SLAM,
  - Use Case 3 : LiDAR SLAM.
- RI-INFUSE-LONG-TRAVERSE-DEM
- RI-INFUSE-LONG-RANGE-TRACKING
- RI-INFUSE-RENDEZVOUS
- RI-INFUSE-RETURN-TO-BASE

- Use Case 1: Visual Map-based Localisation
- Use Case 2: Point Cloud Map-based Localisation

This DFPC is an essential part of multiple implementation scenarios, as it is required to enable a safe autonomous rover demonstration. Indeed, the navigation maps produced by this DFPC will be used to plan and guarantee a safe path for our rovers within the chosen experimentation terrain.

Our implementation leverages existing CNES assets, and will attempt to extend their capabilities to continuous operation, as opposed to a stop-and-go approach which was required until now.

DFPC Inputs :

- Left and right stereo images with associated metadata.

DFPC Outputs:

- Navigation Map.

The DFPC will be composed of the following DFNs:

- Image Radiometric Correction: Depending on input parameters, applies various operations on input images such as normalisation, thresholding, vignetting correction, etc,
- Image Geometric Correction: Using provided intrinsic and extrinsic camera calibration parameters, perform a geometric correction by correcting distortion and rectifying input images to prepare for stereo disparity computation,
- CNES Stereo Disparity: Computes, refines and filters a disparity map from the left and right rectified images,
- Build DEM,
- Fuse DEM,
- Build Navigation Map,
- Fuse Navigation Map.

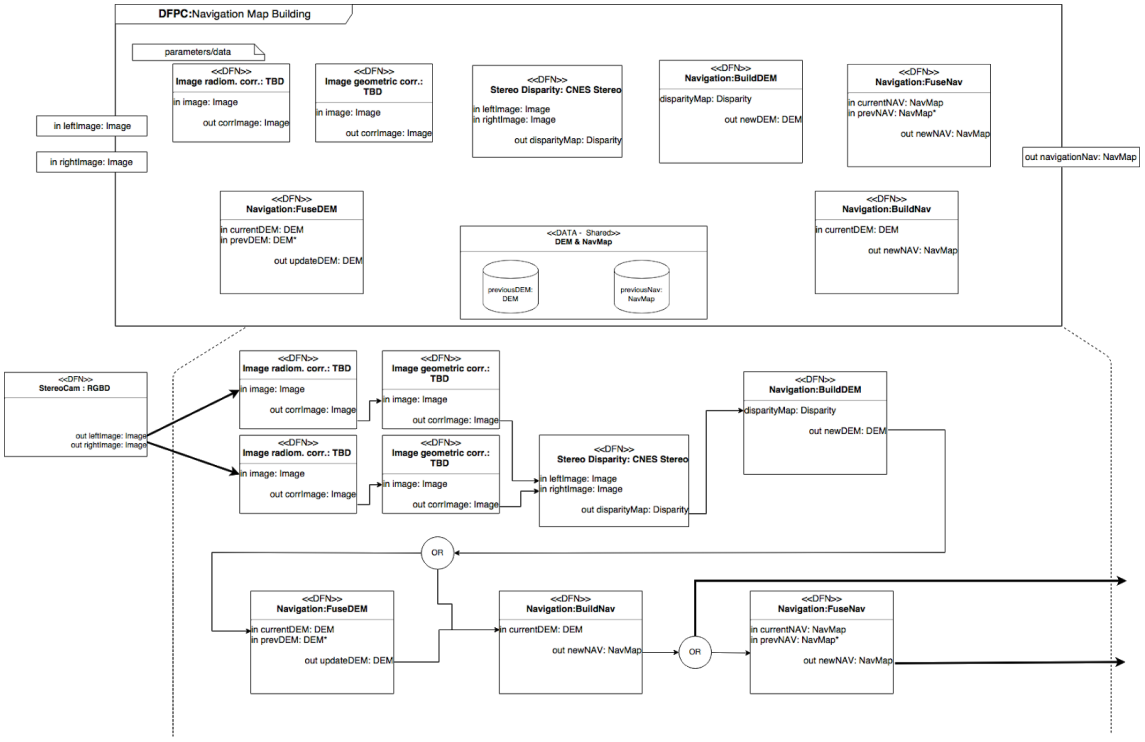


Figure 64: Navigation Map Building Data Flow Description

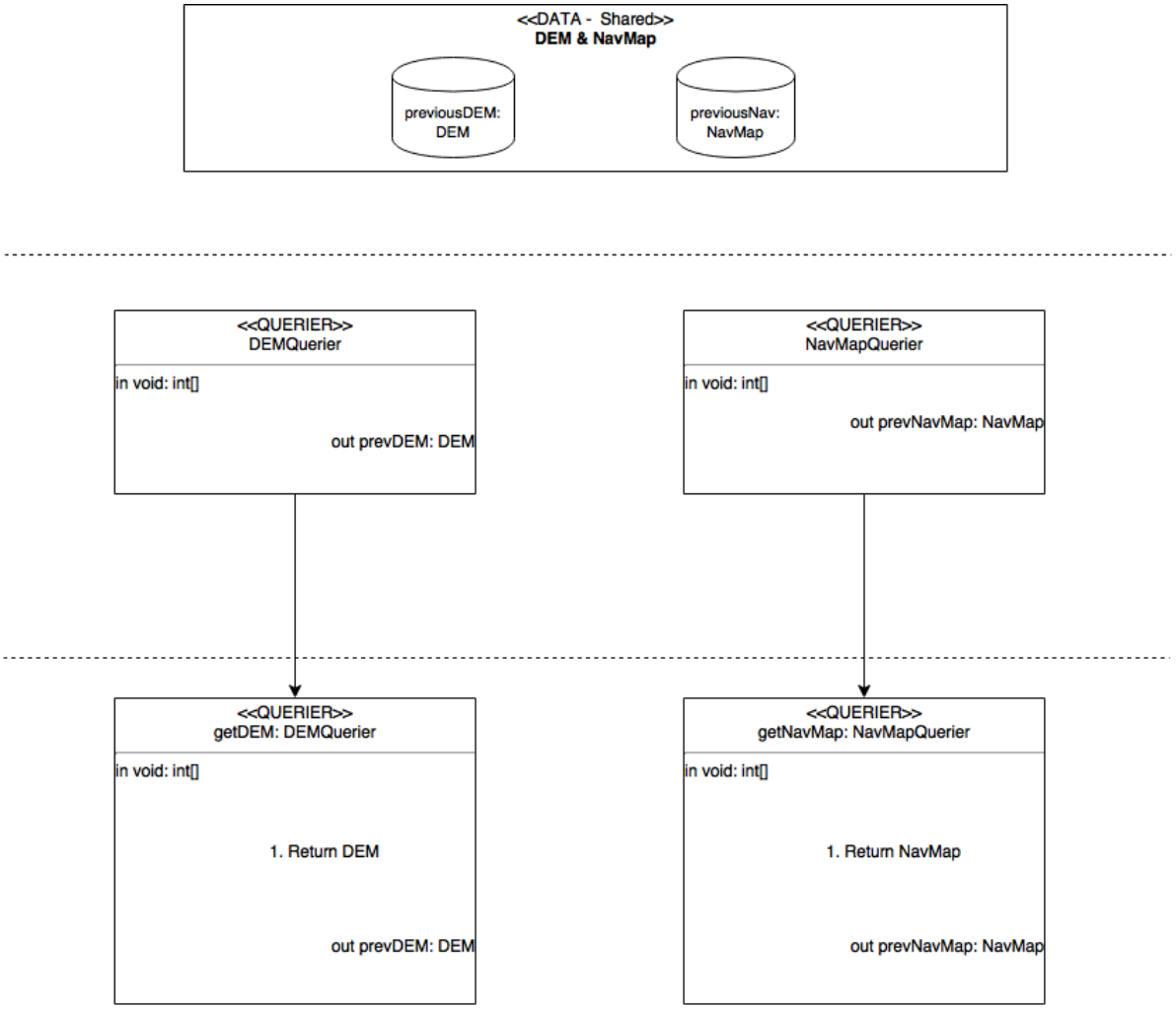


Figure 65: Navigation Map Building Data Management

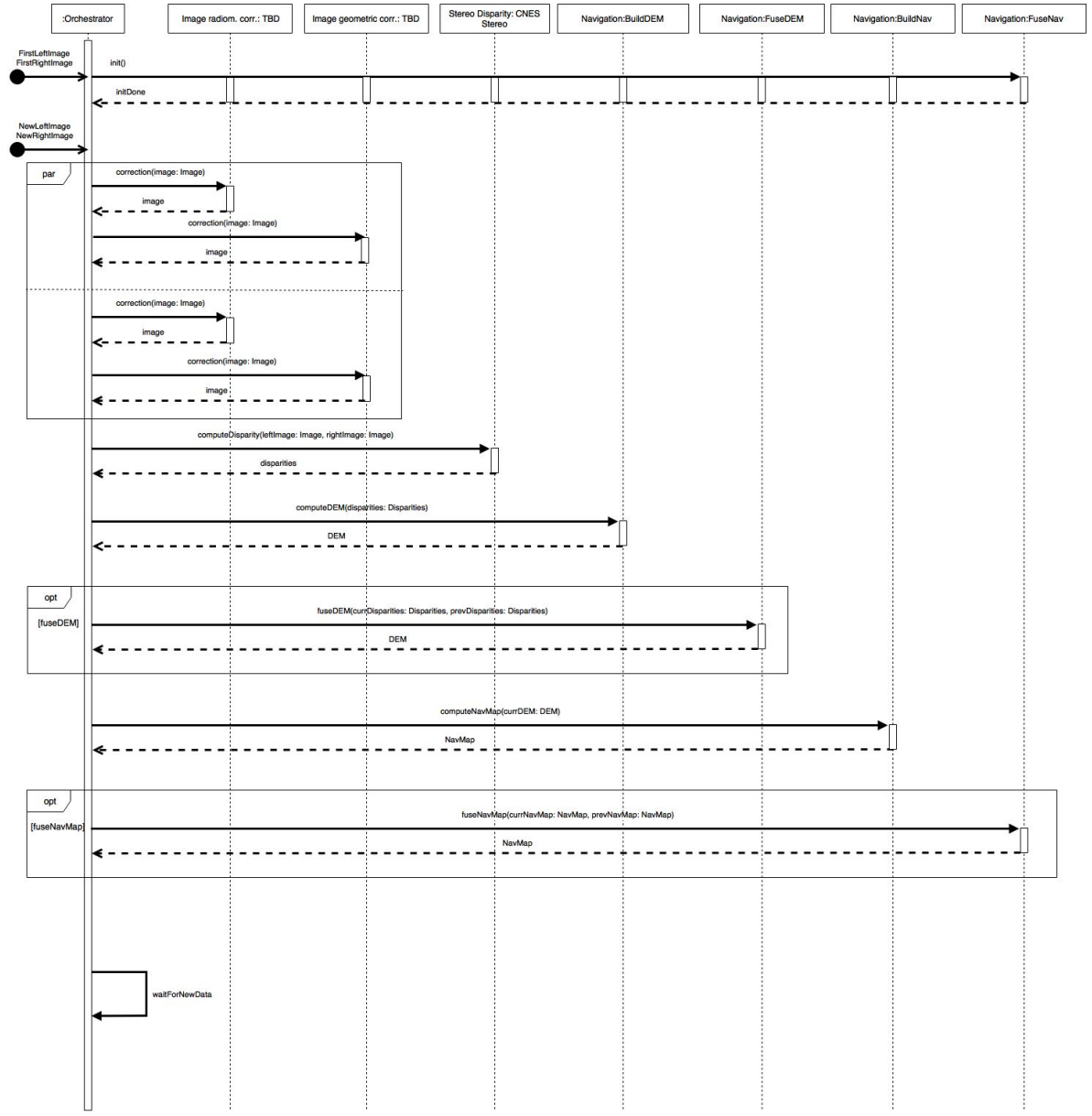


Figure 66: Navigation Map Building Control Description

4.1.13.1 DFPC Expected Performance

A typical way of evaluating the global performance of the navigation map building function is to compare the ratio of the travel distance of the reference trajectory obtained with an A* algorithm applied to the ground truth DEM of the test area, and the planned trajectory which is autonomously executed by the rover using the navigation DFPC. The obtained ratio can be under 100% (executed distance > reference distance), and this behavior is expected, as the A* algorithm is optimal. A ratio under 100% is also possible, as the A*

algorithm is optimal for its cost function, but does not necessarily produces the shortest path.

The final performance of the algorithm is obviously dependent on the type of terrain, the trajectory goals and the complexity of the path. Previous experiments on the CNES mars yard allow us to determine expected performance figures for this criterion:

Table 4: Typical performance ratio for the navigation map building function

	Value
Min ratio	70%
Max ratio	110%
Mean	90%
Median	90%
Stdev	10%

4.1.14 DFPC: Path Planning

This DFPC will be a wrapper over CNES assets. It is presently relevant mainly to implement demonstration and validation scenarios.

4.1.15 DFPC: Pose Fusion

This DFPC is the part of the DPM that manages the poses (Position Manager DFPC, see section 8.4).

4.1.16 DFPC : 3D Model Detection and Tracking

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-RENDEZVOUS
  - Use Case 1: Model-based Localisation
  - Use Case 2: Dense point-based Localisation
  - Use Case 3: 3D Feature Model-based Localisation

This DFPC will support the detection and tracking of known rigid deformable objects. A robotic arm with multiple joints is such an object. An object with broken or missing parts will also be dealt in this DFPC. This DFPC is intended to be used on close range applications.

The detection process will be based on the following graph :

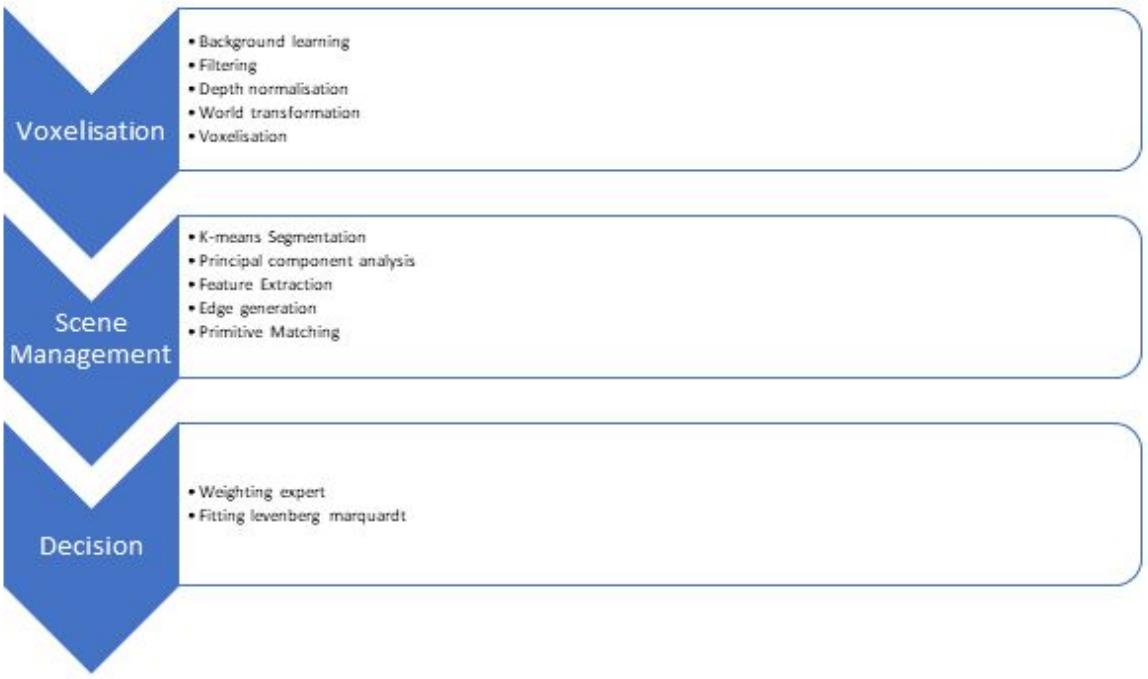


Figure 67: 3D Model Detection and Tracking Process

The detection process is based on a multi-criteria algorithm. The first part of the algorithm will clean the data points, the second part will execute a series of classical image processing algorithms, the last part will weight the results from the previous pass into the decision algorithm.

DFPC Inputs :

- Point cloud with associated metadata from LiDAR sensor or stereo camera or ToF camera.
- Estimated rover pose relative to target.

DFPC Outputs:

- Target object ID
- Estimated Target pose with regard to Rover.

The DFPC will be composed of the following DFNs:

- Background learning : removes static data from the input
- Filtering : Will remove noise and outliers from the input data
- Depth Normalisation : normals are extracted from data
- World Transformation : will if available apply estimated ROV pose to the input data

- Voxelization : will bin input data in a sparse data structure for easier manipulation and also create a point cloud
- K-means : will segment the point cloud
- Feature Extraction : will use SURF feature to generate regions of interest
- Edge Generation : will extract edges present in the pointcloud
- Primitive Matching : will match various planes and ellipses to segmented regions
- Weighting Expert : this multicriteria DFN will accept previous results and follow a voting scheme to generate most plausible pose estimation about subparts, e.g. the gripper, the elbow and the base of an articulated arm.
- Fitting Levenberg Marquardt : will fit classified results of known subparts into possible known objects.

4.1.16.1 DFPC Expected Performance

The target platform is a standard computer made of :

CPU : Intel Core i7-6700HQ @ 2.60GHz

RAM : 16GB DDR4 - 15-15-15

From these hypothesis the expected run time are :

DFN	Input type	Single thread – Memory in MB	Single thread – CPU time in ms
Background learning	2D array (640*480*8bit)	1	1
Filtering	2D array (640*480*8bit)	1	1
Depth Normalisation	2D array (640*480*8bit)	1	1
World Transformation	2D array (640*480*8bit)	1	<1
Voxelization	2D array (640*480*8bit)	1	1
K-means	2D array (640*480*8bit)	2	1
Feature Extraction	2D array (640*480*8bit)	7	1

Edge Generation	2D array (640*480*8bit)	2	1
Primitive Matching	Octomap: 307 000 points	20	2
Weighting Expert	Octomap: 307 000 points	20	Remaining Time
Fitting Levenberg Marquardt	Octomap: 307 000 points	20	6
TOTAL		66	16.66 (~ 60fps)

4.1.17 DFPC : Haptic scanning

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-RENDEZVOUS
  - Use Case 1: Model-based Localisation

This DFPC will support the scanning of objects by using a force measuring sensor. This DFPC is intended to be used on close range applications, and will complement 3D imagery devices. The detection process is based on a force profile algorithm coupled with odometry.

For this DFPC, a robotic arm with 7 degrees of freedom is foreseen to be used.

The detection process will only gather data without requiring arm to be actuated.

DFPC Inputs :

- Desired end-effector position
- End-effector force measurements
- Estimated rover pose relative to target.

DFPC Outputs:

- 3D point cloud representing touched shapes

The DFPC will be composed of the following DFNs:

- Octomap generator : Will merge force normals data into a spatial representation
- Force Mesh Generator : Will exploit the octomap data to generate meshes representing touched objects.

4.1.17.1 DFPC Expected Performance

The target platform is a standard computer made of :

CPU : Intel Core i7-6700HQ @ 2.60GHz

RAM : 16GB DDR4 - 15-15-15

From these hypothesis the expected run time are :

DFN	Input type	Single thread – Memory in MB	Single thread – CPU time in ms
Force Mesh Generator	cartesian Pose	<1	<1
TOTAL		1	>1000fps

4.1.18 DFPC : 3D Reconstruction

This DFPC responds to the following reference implementation scenarios:

- RI-INFUSE-RENDEZVOUS
  - Use Case 1: Model-based Localisation

This DFPC is used for the reconstruction of an environmental point cloud as a preliminary step for model detection.

DFPC Inputs :

- Stream of mono camera images OR Stream of stereo camera images OR stream of point cloud from 3D Lidar OR stream of point clouds from ToF cameras
- Sensors parameters

DFPC Outputs:

- 3D point cloud representing the environment
- stream of camera poses with respect to the first camera pose.

The DFPC will be composed of the following DFNs:

- Image filter (like undistortion filter);
- 2d Features Extractor (like Orb detector, or Harris detector);
- 2d Features Descriptor (like Orb Descriptor);
- 2d Features Matcher (like Flann matcher);

- Fundamental Matrix Computation (like Ransac based estimation);
- Camera Pose Estimation from fundamental matrix (like decomposition of the essential matrix)
- Triangulation of 2d correspondences.

4.1.18.1 Algorithm Details

Due to the complexity and the number of algorithms used in the 3D Reconstruction DFPC, a detailed description of the algorithms is provided as follows.

4.1.18.1.1 Feature Matching

We use ORB (Oriented FAST and Rotated BRIEF) point descriptors for 2-D feature matching. First, a method of keypoint detection must be used to obtain keypoints from a sequence of images. The FAST keypoint detector (Features from Accelerated Segment Test) is frequently used for keypoint detection due to its speed, and is used for quickly eliminating unsuitable matches in ORB. Starting with an image patch p of size 31x31, each pixel is compared with a Bresenham circle centred on that point (built 45 degrees at a time by). The radius of the surrounding circle of points is nominally 3, but is 9 for the ORB descriptor, which expands the patch size and number of points in the descriptor. If at least 75% of the pixels in the circle are contiguous and more than some threshold value above or below the pixel value, a feature is considered to be present. The ORB algorithm introduces an orientation measure to FAST by computing corner orientation by intensity centroid, defined as

$$C=\left(\frac{m_{10}}{m_{00}},\frac{m_{01}}{m_{00}}\right) \text{ where } m_{pq}=\sum_{x,y} x^p y^q I(x,y). \quad (1)$$

The patch orientation can then be found by . Since the FAST detector does not produce multi-scale features, a Harris filtered scale pyramid is used to compare several scales of features.

4.1.18.1.2 ORB Keypoint Description

The feature descriptor provided by BRIEF is a bit string result of binary intensity tests τ, each of which is defined from the intensity p(a) of a point at a relative to the intensity p(b) at a point at b:

$$\tau(p;a,b)=\begin{cases}1: p(a)<p(b)\\ 0: p(a)\geq p(b)\end{cases} \quad (2)$$

and

$$f_n(p)=\sum_{1\le i\le n} 2^{i-1}\tau(p;a_i,b_i).$$

(3)

BRIEF descriptors can be referred to as BRIEF-k, where k is the number of bytes needed to store the descriptor. The descriptor is very sensitive to noise, so Gaussian smoothing is applied to the image patch that the descriptor acts on. The more smoothing, the more matches can be obtained. Also, the basic BRIEF descriptor falls in accuracy quickly with rotations of more than approximately 10 degrees. To make BRIEF invariant to in-plane rotation, it is steered according to the orientations computed for the FAST keypoints. The feature set of points (a_i,b_i) in 2xn matrix form is rotated by multiplication by the rotation matrix R_f corresponding to the patch orientation Θ to obtain the rotated set F:

$$F=R_f\begin{pmatrix} a_1 \cdots a_n \\ b_1 \cdots b_n \end{pmatrix}.$$

(4)

The steered BRIEF operator used in ORB then becomes:

$$g_n(p,\Theta)=f_n(p)\forall(a_i,b_i)\in F$$

A lookup table of steered BRIEF patterns is constructed from this to speed up computation of steered descriptors in subsequent points.

4.1.18.1.3 Matching Process

The first step is to match the keypoints with descriptors generated by BRIEF between two images taken from slightly different positions, attempting to find a corresponding keypoint a' in the second image that matches each point a in the first image. Brute-force matching of all combinations of points is the simplest method which generally involves an XOR operation between each descriptor and a population count to obtain the Hamming distance. This is an $O(N^2)$ algorithm, and takes relatively long to complete. However, The FLANN (Fast Library for Approximate Nearest Neighbor) search algorithm built into OpenCV is used in current work.

4.1.18.1.4 The Fundamental Matrix

To obtain depth in a 3-D scene, an initial baseline for 3-D projection is first required, which for the case of monocular images requires the calculation of the Fundamental Matrix F, which is a the general 3x4 transformation matrix that maps each point in a first image to another second image. It is generally preferable to use stereoscopic vision for point cloud reconstruction because the baseline can be obtained with two cameras a known distance apart at each location. As a result, the fundamental matrix is constant and can be calculated relatively easily. For monocular vision, the fundamental matrix must be estimated using homographies. The set of "good" matches M_g is used to obtain the fundamental matrix for the given scene. The fundamental matrix is the matrix F that maps every point on

the first image to its corresponding location in the second image, based on the assumption of linear geometry between two viewpoints. Consequently, each keypoint a_i in the first image will map to a corresponding keypoint a_i' on the epipolar line (the line of intersection at a_i' of the second image plane with the camera baseline) in the second image by the relation

$$a_i'^T \mathbf{F} a_i = 0,\ i = 1,\ldots,n. \quad (6)$$

For three-dimensional space, the matrix F has nine unknown coefficients and Equation 6 is linear and homogeneous, so F can be uniquely solved for by using eight keypoints with the method of Longuet-Higgins. However, image noise and distortion inevitably cause variation in points that make it difficult to obtain a single "correct" F for all points. Therefore, for practical calculations, a linear estimation method such as linear least squares or RANSAC must be used. RANSAC (RANdom SAmple Consensus) is an efficient algorithm designed for robust model fitting that can handle large numbers of outliers, and is commonly used with OpenCV and other algorithms. We use RANSAC for its speed to estimate F for all matches and estimate the associated epipolar lines. Outliers (defined as being keypoints more than the tolerance 0.1 from the estimated epipolar line) are then removed from M_g to yield a final, reliable set of keypoint matches M_h. If no keypoint matches remain by this point, then there are too few features in common between the two images and no triangulation can be created.

4.1.18.1.5 The Essential Matrix

To perform a three-dimensional triangulation of points from two-dimensional feature planes and a transformation F between them, it is necessary to take into account any transformations and projective ambiguity caused by the cameras themselves. A camera matrix is defined as C=K[R|t], being composed of the calibration matrix K, the rotation matrix R and the translation vector t. We also need to locate the position of the second camera C2 in real space with respect to the first camera C1. The cameras can be individually calibrated using a known pattern such as a checkerboard, but fairly good results have been achieved by estimating the camera calibration matrix as

$$\mathbf{K} = \begin{pmatrix} s & 0 & w/2 \\ 0 & s & w/2 \\ 0 & 0 & 1 \end{pmatrix}. \quad (7)$$

For real-world point localization, we can use the so-called essential matrix that relates two matching normalized points $\hat{x}$ and $\hat{x}'$ in the camera plane as:

$\hat{a}_i'^T \mathbf{E} \hat{a}_i = 0,\ i=1,\ldots,n.$

(8)

In this way, E includes the "essential" assumption of calibrated and is related to the fundamental matrix by E

4.1.18.1.6 Orientation

After calculating E, we can find the location of the second camera C2 by assuming for simplicity that the first camera is uncalibrated and located at the origin (C1=[I|0]). We decompose E=t×R into its component R and t matrices by using the singular value decomposition of E. We start with the orthogonal matrix W and its transpose , where

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(9)

and the singular value decomposition of E is defined as

$$SVD(\mathbf{E}) = \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}.$$

(10)

The matrix W does not directly depend on E, but provides a means of factorization for E. There are two possible factorizations of R, namely $R=UW^TV^T$ and $R=UWV^T$, and two possible choices for t, namely $t=U(0,0,1)^T$ and $t=-U(0,0,1)^T$. Thus when determining the second camera matrix C2=K[R|t], we have four choices in total.

4.1.18.1.7 Triangulation

Given the essential matrix E, and a pair of matched keypoints, it is now possible to triangulate the original point positions in three dimensions using least-squares estimation. The algorithm described by Hartley and Sturm for iterative linear least-squares triangulation of a set of points is used as it is affine-invariant and performs quite well without excessive computation time. A point in three dimensions x when written in the matrix equation form Ax=0 results in four linear nonhomogeneous equations in four unknowns for an appropriate choice of . To solve this, singular value decomposition can again be used, or the method of pseudo-inverses. An alternate method is to simply write the system as Ax=B, with A and B defined as

$$\mathbf{A}=\begin{pmatrix} a_x\mathbf{C1}_{2;0}-\mathbf{C1}_{0;0} & a_x\mathbf{C1}_{2;1}-\mathbf{C1}_{0;1} & a_x\mathbf{C1}_{2;2}-\mathbf{C1}_{0;2} \\ a_y\mathbf{C1}_{2;0}-\mathbf{C1}_{1;0} & a_y\mathbf{C1}_{2;1}-\mathbf{C1}_{1;1} & a_y\mathbf{C1}_{2;2}-\mathbf{C1}_{1;2} \\ b_x\mathbf{C2}_{2;0}-\mathbf{C2}_{0;0} & b_x\mathbf{C2}_{2;1}-\mathbf{C2}_{0;1} & b_x\mathbf{C2}_{2;2}-\mathbf{C2}_{0;2} \\ b_v\mathbf{C2}_{2:0}-\mathbf{C2}_{1:0} & b_v\mathbf{C2}_{2:1}-\mathbf{C2}_{1:1} & b_v\mathbf{C2}_{2:2}-\mathbf{C2}_{1:2} \end{pmatrix} \quad (11)$$

and

$$\mathbf{B}=\begin{pmatrix} -a_x\mathbf{C1}_{2;3}-\mathbf{C1}_{0;3} \\ -a_y\mathbf{C1}_{2;3}-\mathbf{C1}_{1;3} \\ -b_x\mathbf{C2}_{2;3}-\mathbf{C2}_{0;3} \\ -b_v\mathbf{C2}_{2:3}-\mathbf{C2}_{1:3}. \end{pmatrix} \quad (12)$$

Solution of the resulting system of equations (in this case, using singular value decomposition) yields x, which can be transformed into undistorted “real” coordinates by x=KC1x. This assumes that the point is neither at 0 nor at infinity, so very distant points may have to be removed before this process. Because solutions are possible for either direction of the translation vector t between the cameras, or for a rotation of π radians about the vector t, so this triangulation must be performed four times, once for each possible combination of R and t, and each resulting point set checked to verify it lies in front of the camera. We use a simple perspective transformation using C1 and a test to ensure $x_z>0$. Triangulation produces a point cloud in local (camera) coordinates with points p_i.

4.1.18.1.8 Pose Estimation

The last step is to find the object pose from the 3D-2D point correspondences and consequently the egomotion of the camera relative to the feature points, commonly know as the Perspective & Point (PnP) problem. Bundle adjustment can also be performed to optimize the point cloud after triangulation, but works best on a large number of points and images for, while we are focused on relatively fast triangulation over a few frames. For this, we apply the OpenCV implementation of the EPnP algorithm. Four control points denoted as are used to identify the world coordinate system of the given reference point cloud with n points $p_1 \ldots p_n$, chosen so that one is located at the centroid of the point cloud and the rest are oriented to form a basis with the principal directions of the data. Each reference point is described in world coordinates as a normalized, weighted sum of the control points with weightings α_{ij}. As this coordinate system is consistent across linear transforms, they have the same weighted sum in the camera coordinate system, effectively creating a separate basis

$$\mathbf{p}_i^w = \sum_{i-1}^{4} \alpha_{ij} \mathbf{c}_j^w, \mathbf{p}_i^c = \sum_{i-1}^{4} \alpha_{ij} \mathbf{c}_j^c, \sum_{i-1}^{4} \alpha_{ij} = 1. \quad (13)$$

The known two-dimensional projections of the reference points can be linked to these weightings with the camera calibration matrix K considering that the projection involves scalar projective parameters as

$$\mathbf{K}\mathbf{p}_i^c = w_i \begin{pmatrix} \mathbf{u}_i \\ 1 \end{pmatrix} = \mathbf{K} \sum_{i-1}^{4} \alpha_{ij} \mathbf{c}_j^c. \quad (14)$$

The expansion of this equation has 12 unknown control points and n projective parameters. Two linear equations can be obtained for each reference point, and concatenated together to form a system of the form Mx=0, where the null space or kernel of the matrix gives the solution x to the system of equations, which can be expressed as

$$\mathbf{x} = \sum_{i-1}^{m} \beta_i \mathbf{v}_i \quad (15)$$

where the set is composed of the null eigenvectors of the product corresponding to m null singular values of M. The method of solving for the coefficients β depends on the size of m. Given perfect data from at least six reference points, m should be 1, but in practice, m can vary from 1 to 4 depending on the camera parameters, reference point locations with respect to the basis, and noise. Hence, four different methods are used in the literature for practical solution, but the methods are complex and not summarized here.

4.1.18.2 Expected Performance

The same set of test parameters as in Table 4 from Section 4.1.8.2.2 were used for 3D reconstruction testing with a 1U CubeSat model. To gain an estimate of time required for processing, the process of reconstruction was profiled running on the ARM core of a Xilinx Zynq Z7020 SoC microcontroller (667MHz ARM-Cortex A9). Table 5 shows the timing results for each part of the 3D reconstruction process[3]. It can be seen from this that the

[3] M.A. Post, J. Li, C. Clark, X. Yan. "Visual Pose Estimation System for Autonomous Rendezvous of Spacecraft". ESA Astra 2015: 13th Symposium on Advanced Space Technologies in Robotics and Automation. ESA/ESTEC, Noordwijk, the Netherlands, 11-13 May 2015.

majority of time is spent on keypoint production and FLANN search during the tracking process.

Table 5: Timing Results for CubeSat 3D reconstruction from image sequences

Test	Feature Detection	Feature Matching	Feature Selection	Fundamental Matrix	Essential Matrix	Triangu-lation	PnP RANSAC	Ego-Motion	TOTAL (s)
1-2	0.12	0.058	0.015	0.083	0.0017	0.038	0.0033	0.0005	0.32
3-4	0.12	0.061	0.010	0.048	0.0014	0.025	0.0026	0.0004	0.27

The accuracy of the reconstruction was evaluated through dimensional analysis of the resulting point cloud. Table 6 shows the dimensions of separate components reconstructed from images.

Table 6: Dimensional Analysis Results for CubeSat 3D reconstruction

Component	Size X (m)	Size Y (m)	Size Z (m)	Real X (m)	Real Y (m)	Real Z (m)
Left Solar Panel	0.307	0.103	0.025	0.300	0.100	0.002
Right Solar Panel	0.309	0.117	0.032	0.310	0.100	0.002
Body	0.336	0.112	0.120	0.315	0.100	0.100
Satellite	0.337	0.230	0.115	0.315	0.264	0.100

These results show a lower noise limit for point cloud reconstruction, such that thin components such as solar panels may exhibit up to 3cm deviation of their points with the camera within the close range domain as has occurred in these tests. For larger components of a 0.3m sized target, this translates into an estimated up to 10% error in dimensional analysis, where a maximum error of 7% was observed in these tests. It should be noted that surface detection and filtering may be used to reduce this error in additional DFNs or DFPCs.

Test	Reference	Output	Measure

Features Detection and Matching	Ground Truth matches on images	estimated matches	90% similarity
Features Detection, Matching and Pose Estimation	Ground Truth camera poses of images	estimated poses	euclidean distance and angular distance less than 1% of R, where R is the maximum operational distance of the camera.
Triangulation and 3D reconstruction	Ground Truth point cloud	estimated environment cloud	number of estimated scene outliers less than 10%

4.2 Flight software : DFN Detailed Design

Here we present the detailed design of each data fusion node (DFN) identified in the DFPCs. A DFN is an atomic processing entity that fulfills a given basic function. It is the smallest unit of a complex task defined by its function, input and output. However, a DFN can be defined by a combination of elementary functions which may not expose their input/output. A DFN exhibits at least two control interfaces:

- configure() and
- process()

The configure() sets all the configuration parameters of the DFN while the process() function calls library functions to compute the outputs of the DFN.

This section presents the detailed design of each DFN identified in the DFPCs described previously. This should be the last step before code-level description. The section introduces two templates: one for DFNs (Sec 4.2.1.1) and one for DFIs (Sec 4.2.1.2).

Given the current stage of the project, it is not possible to have a complete exhaustive description of all DFNs and even more of DFIs. Additionally, some fields, e.g. “Diagnostic capabilities” and “Unit test”, can be filled only after extensive testing and validation and not before the implementation. However a partial list is provided as it is helpful to create some instances to see how they fit to the proposed template.

The application of part of the DFNs to this template will enforce an auspicable node standardisation for the future development phase, during which design details will be progressively updated.

4.2.1 DFN: Template DFN

The following section and its subsections propose a template for DFNs.

4.2.1.1 DFN Description

One DFN may have multiple DFI.

DFN Name	**DFN_Example**
DFN element	**Remarks**
Generic description	Must be present
Input(s) and Ouput(s) data	Data here means both: • Actual data (e.g. Image16bit) • Metadata (e.g. CameraParameters, Timestamp) A DFN cannot work without these data structures.
Input Parameters	Can be provided by: • Configuration file (e.g. Threshold, Size of patch) • Output of another DFN (e.g. KF reinitialization)
Performance and cost estimation methods	A cost/performance estimation method which is common to all DFI of this DFN.
Unit test	This must be provided with the code, along with the dataset used for validation.

4.2.1.2 DFI: Template Implementation

This template applies to any DFI. As a DFN can have multiple DFIs, there can be several instances of this template under the same DFN.

DFI Name	**TemplateNameImplementation1**
DFI element	**Remark**
Est. performance and cost	Possibly represented, in an adequate cost/performance space. This information should make it possible to define a performance measure and a cost measure for a resulting DFPC.
External library dependencies	List of external library dependencies (e.g Opencv, PCL)
Input Parameters	DFI-specific input parameters. For example: • Feature thresholds, • Descriptor length
Diagnostic capacities	Includes:

	• Errors/warnings at runtime (e.g. unexpected datatype, out-of-range parameter...). • Log capabilities (e.g. try/catch results written in a log file) • Output reports (e.g. "if the image is all black…") Fault Detection and Identification is the responsibility of the DFN. However its Recovery (when possible) is made at the DFPC level and is part of the Orchestrator.

4.2.1.3 DFN Description File

The DFN description file is a human readable artifact that describes the DFN based on the DFN template. A code generator produces C++ code and corresponding python bindings from the DFN description file.

Here we include the DFN YAML description file following our CDFF-Support specification.

File: TemplateDFN.yml

```
name: Template
input_ports:
   - name: templatePort1
     type: base::samples::template
     doc: Template port 1
   - name: templatePort2
     type: base::samples::template
     doc: Template port 2
output_ports:
   - name: templatePort3
     type: base::samples::template
     doc: Template port 3
```

4.2.1.4 DFN Sequence Diagram

What happens inside the configure() and process() calls of this DFN.

4.2.2 DFN Detailed Design

The next section describes a first part of the detailed implementation of specific DFNs which can be used in certain DFPC presented above. The DFN elements and description are provided for each data fusion node below.

4.2.2.1 DFN: FeatAndSigExtractor

DFN Name	FeatAndSigExtractor
DFN element	**Remarks**
Generic description	Detects and extract a visual point feature from an image. The feature is represented by a detector choosing keypoints of interest in the image and by an array of descriptors describing the region around the keypoint.
Input(s) and Ouput(s) data	Input: A grayscale image (e.g. cv::Mat) Output: A vector of keypoints (e.g. cv::Keypoint) and an array of descriptors (e.g. cv::Mat) for each keypoint
Input Parameters	Number of maximum desired features.
Performance and cost estimation methods	Detection time can be used to estimate the cost of any feature extractor. For some instances, performance evaluation methods may exist.
Unit test	

4.2.2.1.1 DFI: SIFT Feature Extractor

The use of SIFT has to be confirmed due to patent pending on the algorithm.

DFI Name	SIFT Feature Extractor
DFI element	**Remark**
Est. performance and cost	Generally slower than other extractors. Very robust.
Input Parameters	- Number of octave layers - Edge Threshold - Contrast Threshold - Sigma
External library dependencies	OpenCV
Diagnostic capacities	N/A
Unit test	

4.2.2.1.2 DFI: SURF Feature Extractor

The use of SURF has to be confirmed due to patent pending on the algorithm.

DFI Name	SURF Feature Extractor
DFI element	**Remark**
Est. performance and cost	It is generally outperformed by SIFT but significantly faster.
Input Parameters	- Hessian Threshold - Number of pyramid octaves - Number of octave layers - Flags for extended descriptor and/or orientation computation
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.1.3 DFI: ORB Feature Extractor

DFI Name	ORB Feature Extractor
DFI element	**Remark**
Est. performance and cost	Very fast and computationally efficient. Less reliable than SIFT/SURF.
Input Parameters	- Pyramid decimation ratio (scale factor) - Number of pyramid levels (similar to SIFT/SURF) - Edge threshold - Number of points to produce for BRIEF - Patch size used by BRIEF
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.2 DFN: Feature Matching

DFN Name	Feature Matching
DFN element	**Remarks**
Generic description	Given two sets of visual point features returns a set matches. Each match associate two features.
Input(s) and Ouput(s) data	Input: Two vectors of keypoints and their relative descriptors Output: A vector of matches (e.g. cv::DMatch)

Input Parameters	Distance threshold to accept matches.
Performance and cost estimation methods	Matching time. Percentage of outliers in the matches. Min/Max/Avg distance in the matches.
Unit test	Comparison with known matches in two images

4.2.2.2.1 DFI: FLANN Matcher

DFI Name	**FLANN Matcher**
DFI element	**Remark**
Est. performance and cost	Main alternative in literature to a brute force matcher. Best choice in terms of computation time.
Input Parameters	
External library dependencies	OpenCV
Diagnostic capacities	TBD

4.2.2.3 DFN: 3D Point Computation

DFN Name	**3D Point Computation**
DFN element	**Remarks**
Generic description	Given two sets of visual point features and the calibration matrix of the camera with which the images were taken returns a point cloud or more generally a set of 3D points.
Input(s) and Ouput(s) data	Input: Two vectors of keypoints, their pairings and a calibration matrix Output: A vector of 3D points (e.g. cv::Point3f) or a point cloud
Input Parameters	
Performance and cost estimation methods	Computational time.
Unit test	

4.2.2.3.1 DFI: Linear Triangulation (DLT)

DFI Name	**Epipolar geometry**
DFI element	**Remark**
Est. performance and cost	Fast but dependent on number of points
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	

4.2.2.4 DFN: 3D-3D Motion Estimation

DFN Name	**3D-3D Motion Estimation**
DFN element	**Remarks**
Generic description	Given two sets (p,q) of corresponding 3D points (point clouds) estimates a rigid transformation (R, t) minimizing the reprojection error |(Rp+t)-q|
Input(s) and Ouput(s) data	Input: 2 vectors of corresponding 3D points, Output: a rigid transformation aligning the 2 sets of points
Input Parameters	None
Performance and cost estimation methods	Complexity. Computational time.
Unit test	

4.2.2.4.1 DFI: SVD Rigid Body Transformation

DFI Name	**SVD Rigid Body Transformation**
DFI element	**Remark**
Est. performance and cost	The algorithm has a complexity of $O(d^3)$, where d is the dimension of the input (typically small).

Input Parameters	Optionally the DFI can utilise a weight array for all the point pairs
External library dependencies	None
Diagnostic capacities	TBD

4.2.2.5 DFN: 2D-3D Motion Estimation

DFN Name	**2D-3D Motion Estimation**
DFN element	**Remarks**
Generic description	Given a set of 3D points and a their corresponding image projections. Computes a rigid transformation minimizing the reprojection error.
Input(s) and Ouput(s) data	Input: a vector of 3D points, a vector of image points, a camera matrix and an array of distortion coefficients Output: a rigid transformation
Input Parameters	None
Performance and cost estimation methods	Complexity. Computational time. Percentage of inliers.
Unit test	

4.2.2.5.1 DFI: PnP (Perspective from n-Points)

DFI Name	**PnP**
DFI element	**Remark**
Est. performance and cost	Dependent from the algorithm parameterisation
Input Parameters	- Solving method (e.g. EPNP, Iterative, P3P) - Apply ransac or not + ransac parameters - Use extrinsic guess (for Iterative method)
External library dependencies	OpenCV
Diagnostic capacities	Depending of the chosen method. E.g. P3P requires exactly 4 matches or will return error.

4.2.2.6 DFN: Point Cloud Construction

DFN Name	**Point Cloud Construction**
DFN element	**Remarks**
Generic description	This DFN combines a new point cloud with an existing point cloud by only adding points that have not been already triangulated
Input(s) and Ouput(s) data	Input: Point cloud, re-projected points, pose estimates Output: Point cloud
Input Parameters	None
Performance and cost estimation methods	Computational time
Unit test	N/A

4.2.2.6.1 DFI: Point Cloud Builder

DFI Name	**Point Cloud Builder**
DFI element	**Remark**
Est. performance and cost	Fast overall
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.7 DFN: Bundle Adjustment

This DFN is optional for use in environment reconstruction

DFN Name	**Fundamental Matrix Calculation**
DFN element	**Remark**
Generic description	This DFN optimizes point clouds so that they are a better match to images

Input(s) and Ouput(s) data	Input: Point cloud, feature descriptors (type, cv::Mat), pairings of features Output: Point cloud
Input Parameters	Camera parameter matrix, distortion coefficients
Unit test	N/A

4.2.2.7.1 DFI: Bundle Adjustment

DFI Name	**Bundle Adjustment**
DFI element	**Remark**
Est. performance and cost	High computational load for large point clouds
Input Parameters	None
External library dependencies	Ceres-solver
Diagnostic capacities	N/A

4.2.2.8 DFN: Fundamental Matrix Calculation

DFN Name	**Fundamental Matrix Calculation**
DFN element	**Remark**
Generic description	This DFN calculates a fundamental matrix given feature positions and their matches
Input(s) and Ouput(s) data	Input: feature descriptors (type, cv::Mat); Pairings of features, good triangulations for these features Output: Fundamental Matrix (type, cv::Mat)
Input Parameters	None
Unit test	Calculate a known matrix from known points

4.2.2.8.1 DFI: Fundamental Matrix Calculator

DFI Name	**Fundamental Matrix Calculator**
DFI element	**Remark**

Est. performance and cost	Fast calculation
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.9 DFN: Estimation Filter

DFN Name	**Estimation Filter**
DFN element	**Remark**
Generic description	Predicts the state based on a state motion model and corrects it with a measurement
Input(s) and Ouput(s) data	Input: current state, motion model, measurement, measurement model Output: predicted and updated state
Input Parameters	Process noise, measurement noise, initial covariance
Performance and cost estimation methods	Error w.r.t. ground truth.
Unit test	

4.2.2.9.1 DFI: Extended Kalman Filter

The Kalman filter consists of the functions: init for initialization, predict and correct for update of the predicted state with the measurement.

DFI Name	**Extended Kalman Filter**
DFI element	**Remark**
Est. performance and cost	Dependent on use case
Input Parameters	See following table
External library dependencies	OpenCV
Diagnostic capacities	TBD

Kalman Filter parameters and design specifics are tailored to the use case. Some possible parameters of the DFN are gathered in the table below.

Parameter	Parameter type
Max number of sample points	Tracker parameter
Search distance [pixel integer]	Tracker parameter
Maximum LSE iteration	Tracker parameter
Minimum incremental update to declare convergence [deg, mm]	Tracker parameter
Maximum update parameters with respect to initial prediction to declare divergence[deg, mm]	Tracker parameter
Threshold on percentage of inlier matches	Tracker parameter
Kalman filter process noise	Tracker parameter
Kalman filter measurement noise	Tracker parameter
Kalman filter initial covariance	Tracker parameter

Table 5: DFPC Parameters of the model-based visual tracking

4.2.2.10 DFN: Image Geometric Processing

DFN Name	**Image Geometric Processing**
DFN element	**Remark**
Generic description	Corrects the distorted image geometrically
Input(s) and Ouput(s) data	Input: gray scale image (type, cv::Mat) Output: gray image (type, cv::Mat)
Input Parameters	Camera parameter matrix, distortion coefficients
Performance and cost estimation methods	N/A

4.2.2.10.1 DFI: Image Undistortion

DFI Name	Image Undistortion
DFI element	**Remark**
Est. performance and cost	N/A
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	N/A
Unit test	

4.2.3 Remaining DFNs

The remaining DFNs, which have been listed in the various DFPCs, but not in this detailed design section, as well as the further details needed to complete the description templates of DFNs currently provided as examples in the previous sections, will be continuously completed and updated during the future development phase.

4.3 Ground software : DFPC Architecture and Design

4.3.1 DFPC: Camera calibration

This is a supporting DFPC, performed in preparations for the actual test run. Its task is to obtain optimal camera parameters - perspective projection and distortion parameters (radial, tangential, thin-prism) - which are used to provide mapping between points in image space and rays in real-word space. Correct camera calibration is a prerequisite to any DFPC depending upon single camera data and corresponding metadata as its input.

Calibration hardware: Cameras will be calibrated using images of 2D checkerboard-like pattern with origin denoted by three dots. We recommend manufacturing a pattern of size that fills in the whole field of view when positioned at the focus distance of the cameras, but in practice a smaller one, placed closer to the camera to still fill in roughly the whole field of view, can be used as well. A solid alu-dibond finishing or similar is recommended.

Calibration software: DLR aims to provide its background-IPR camera calibration softwares CalDe and CalLab. CalDe is a tool used to extract checkerboard corners from images of calibration pattern. This tool can be used either in a fully automatic mode, or semi-automatically with user pointing to the pattern origin. CalLab is a tool to perform a non-linear least-squares optimization of correspondences between image-space and real-world-space positions of the extracted corners.

This DFPC inputs are :

- Several camera images
- Calibration pattern configuration file

The DFPC output is:

- Estimated camera calibration parameter file

4.3.2 DFPC: Stereo calibration

This is a supporting DFPC, performed in preparations for the actual test run. Its task is to obtain optimal stereo camera bench parameters - perspective projection, distortion (radial, tangential, thin-prism) and stereo-camera-transformation parameters - which are used to provide mapping between points in stereo image space and points in real-word space. Correct stereo camera calibration is a prerequisite to any DFPC depending upon stereo camera bench data and corresponding metadata as its input.

Description of hardware, software and input/output breakdown is identical to previous chapter.

4.3.3 DFPC: Body/ Camera calibration

This is a supporting DFPC, performed in preparations for the actual test run. Its task is to obtain optimal 6DOF transformation between robot body frame and camera frame. Correct body-to-camera calibration is a prerequisite to any DFPC, which needs to obtain robot body egomotion from visual odometry inputs.

5 Detailed Description of EGSEs

The goal of this chapter is to provide the detailed design of EGSE and DFPCs. It includes the description of the hardware that is identified to conduct analog tests, completed by the software API they offer and test sites. The software detailed design should describe all data types used by DFPCs and DFNs as well as their integration into robotics middlewares.

The chapter is organised as follows :

- presentation of the EGSE detailed design, including DLR, CNRS/LAAS, DFKI and CNES EGSE,
- presentation of the InFuse framework, the objective here is to provide an overview of its key principles,
- presentation of the detailed design of DFPCs,
- presentation of the detailed design of DFNs.

This work is still going on and this chapter will live during the development phase of the projects.

This section presents all the EGSE that will be involved in InFuse to demonstrate and validate InFuse components. It includes EGSE provided by DLR, CNRS/LAAS and DFKI.

5.1 DLR EGSE

Ground support equipment provided by DLR is threefold:

- The Handheld Central Rover Unit (HCRU), with various setups,
- The BB2 rover, to carry the HCRU and its own sensors,
- The Planetary Exploration Lab (PEL).

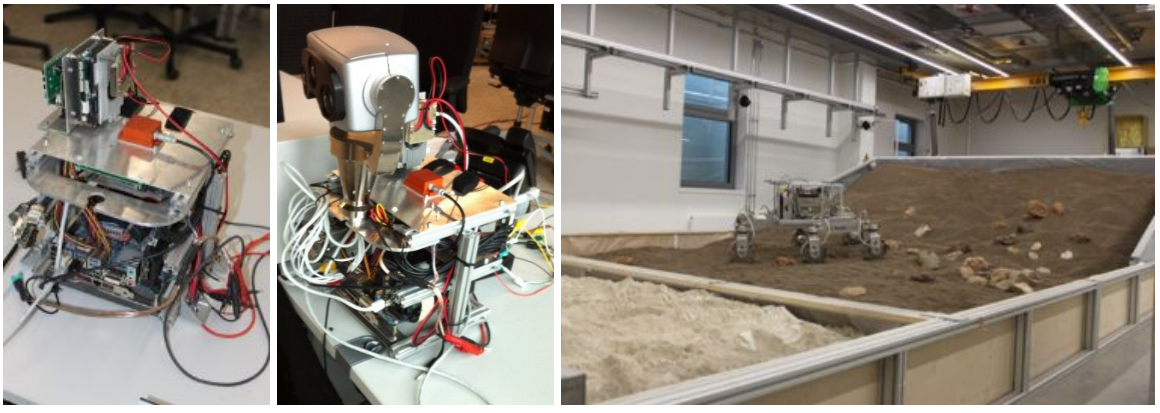


Figure 69: Illustrations of DLR EGSE, from left to right: (1) HCRU basic setup, (2) HCRU sensoric setup, (3) BB2 in PEL.

5.1.1 HCRU

The Handheld Central Rover Unit presents the following characteristics:

- mobile and portable
- hardware and software is equivalent to ExoMars BB2 and LRU rover
- EtherCAT Bus System with control frequency 1kHz

Sensor suite:

- IMU: XSens MTi-10
- B/W camera: Guppy Pro F-125 B
- Color camera: Guppy Pro F-125 C
- Camera lens: Schneider/Kreuznach Cinegon 1.8/4.8

Software setup and APIs:

- OS : OpenSuse Leap 42.1 64bit,
- Runtime-configurable hardware abstraction framework: robotkernel (DLR proprietary),
- IMU integration via xsens_imu_drivers,
- Camera drivers and software integration: SensorNet (DLR proprietary),
- Robot control operating system: ROS,
- Module deployment: Links and Nodes real-time framework (DLR proprietary).

5.1.2 BB2 Rover

The ExoMars rover BB2 has a total mass of approximately 100 kg and a footprint of approximately 136 cm x 131 cm and allows mounting relatively large payloads. The rover is equipped with various sensors (e.g. six 6DOF force-torque sensors) and both the high level rover control as well as the individual actuators can be commanded independently.

Figure 70: ExoMars Phase B2 Breadboard (BB2)

BB2 has currently been refurbished and is now in line with the “DLR mobility standard”. A uniform software and hardware architecture has been implemented to facilitate testing and integration. Now, all foreseen systems (e.g. handheld mock-up or optional LRU rover) share

a common architecture which allows mutual usage and exchange of software and hardware modules or the implementation of optional sensors or algorithms on any of the devices.

Hardware:

- Motor controller: 18x Elmo Gold DC Whistle,
- Fieldbus: EtherCAT,
- PC: Intel Core i7-2720QM, 8 GB RAM, Kontron KTQM67/mITX,
- Encoder: 18x Maxon MR128 CPT 225771,
- Potentiometer: 15x NOVO WAL305 (Steering, Deployment, Bogies),
- 6 Force/Torque-Sensors: ATI Mini85 S-1000-50.

Software:

- OS: openSUSE Leap 42.1 64 bit,
- Runtime-configurable hardware abstraction framework: robotkernel (DLR proprietary),
- Module deployment: Links-and-Nodes real-time framework (DLR proprietary)

5.1.3 Planetary Exploration Lab

The DLR-RM PEL test facility consists of a 5.5 m x 10 m indoor soil bin. It is used for testing the short range planetary scenarios foreseen for OG3 validation and testing.

The terrain can be composed by different materials, e.g. soft soil or stones and rocks of different size and shape, to investigate the mobility for different terrain set-ups. A wide variety of planetary soil simulants as well as obstacles and rocks is available.

Figure 71: PEL Pose Tracking and DSM Device

For ground truth generation, the PEL is equipped with an optical position tracking system and a fully automated Digital Surface Model (DSM) device.

The pose tracking system allows tracking all 6 degrees of freedom and the trajectory of the rover or the position of obstacles can be measured precisely.

Pose tracking characteristics:

- Real time tracking of 6 DoF rover motion @ 60 Hz,
- Allows measuring pose of reference landmark precisely,
- 8 cameras,
- Marker on rover or reference landmark,
- Accuracy:
    - up to approx. < 1mm,
    - up to approx. < 1°.

The DSM device consists of five cameras that are mounted on a bar which is moved over the test bed by a linear axis. From these coloured images, a 2.5 dimensional DSM with a resolution of a few millimetres per pixel can be computed via Semi-Global Matching. The DSM directly forms a precise reference map of the terrain and can also be used to document the test setup and results (e.g. wheel tracks).

DSM characteristics:

- Cameras: 1360 x 1024 pixels,
- Accuracy up to :
    - Vertical 2.23 mm,
    - Horizontal: 3.13 mm.

5.2 DFKI SherpaTT Rover

SherpaTT will be used in a Mars analog demonstration scenario in Morocco. On the rover, the HRCU described in section 5.1.1. will be integrated. The rover will be teleoperated and the data generated by both system will be used to demonstrate the the CDFF-Dev features offline.

OG6 will provide an API to send motion commands to SherpaTT rover as well as a control layer to perform the corresponding motions. The sensor data, odometry and joint states will be made available via an API, exactly as it will be done in the OG2 demonstration. During SherpaTT functioning, Rock logs with the sensor data will be generated. A joypad will be also provided allowing in this way the remote operation of the SherpaTT rover by a human operator. The ground truth position of the robot will be given thanks to a differential GPS.

The HCRU provided by OG6 will be mounted physically to the robot allowing the acquisition of a richer set of sensor values. An API will be made available by OG6 to allow the

acquisition of sensor data similar to the one provided to access SherpaTT. The sensor data generated will, as well as for the rover case, be logged.

At a later stage the logged data will be loaded and fused offline using InFuse CDFF-Dev to provide maps and poses of the robot. Any existing DFPC provided by InFuse and suitable for such scenario can then be tested offline. In this demonstration, features of CDFF-Dev will be validated such as loading of logged data from multiple RCOSs, and testing of various data fusion solutions on such data.

SherpaTT sensors:

- Lidar: Velodyne HDL-32E,
- Laser range finder: Hokuyo UST-20LX,
- Camera: Basler Ace (2048 x 2048 px, 25 fps),
- IMU: Xsens MTi-28A AHRS.

SherpaTT actuators:

- RoboDrive BLDC-motor kits, HarmonicDrive gears and partially equipped with linear drive kits.
- Suspension system: 4 identical, aluminium-casted units with 5 DoF. Allows active ground adaptation and independent body attitude control.
- Force linear joints: 3500 N,
- Wheel torque: 74 Nm (nominal).

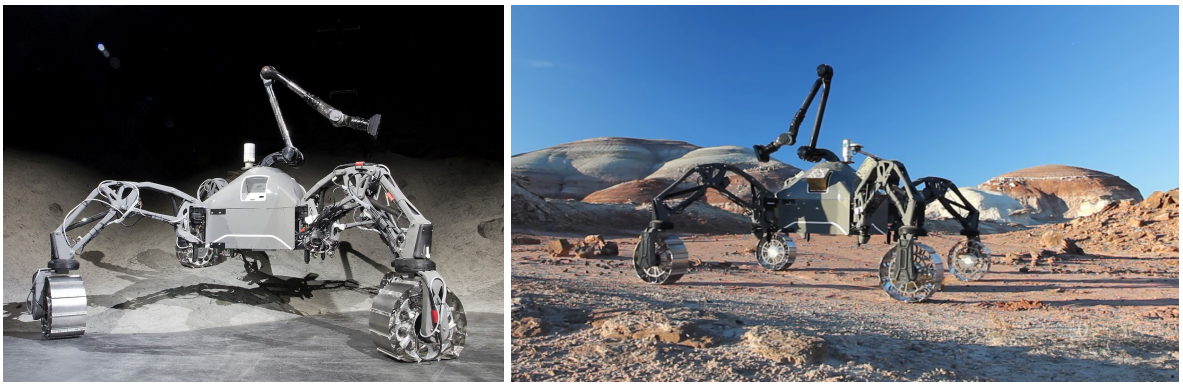


Figure 72: SherpaTT in two space analog scenarios

SherpaTT Software API:

- Interface between OG6-Sherpa and HRCU :

SherpaTTTeleClient
+getRoverPose() : Pose +getManipulatorJointState() : Joints +getMobileBaseJointState() : Joints +getFrontalCameraImage() : Image +getGripperCameraImage() : Image +getIMUData() : IMU +getDEM() : DEM (TBC) +getDGPS() : DGPS +sendMotion2D(eing. command : Motion2D) +sendManipulatorJointCommand(eing. command : Joints)

Figure 73: API Provided by OG6 to exchange data with SherpaTT

5.3 LAAS Rovers

The multi-robot EGSE is made of the two robots Mana and Minnie, each equipped with a full suite of sensor that allow them to perform autonomous navigation tasks. Both robots have been conceived as configurable research platforms, to which additional sensors and computing units can be added.

Figure 74: The robots Mana (right) and Minnie (left), pictured in autumn 2015.

5.3.1 Mana

Platform: Mana is a Segway RMP400 platform, with four non-orientable 0.53 m diameter wheels. The platform dimensions are w x l x h = 76 x 112 x 61 cm, and its weight is 110 kgs.

The platform is controllable through a USB interface with linear and angular speed commands sent at 50 Hz. Each wheel rotation speed and is returned as the same rate.

Proprioceptive sensors: apart from the platform sensors, Mana is equipped with:

- A six-axis IMU (Xsens MTI-100) made of 3 accelerometers, 3 gyrometers and 3 magnetometers. The device can return raw data or attitude and heading information.
- A fiber-optic gyro (KVH DSP-3000), mounted so as to measure the yaw speed. After correction of the Earth rotation (defined by the operating latitude), the drift of the integrated heading speed, delivered at 100 Hz, is of the order of 1 degree per hour.
- A RTK GPS (Novatel Flex6), that emits measured positions and speeds at 100 Hz with a cm accuracy (provided corrections emitted from a nearby base are received at a rate of 0.5 Hz)

The data of these three sensors are accessed through a RS232 serial link.

Exteroceptive sensors: Mana is equipped with 2 Lidars:

- A panoramic Lidar (Velodyne HDL64), composed of 64 independent telemeters, delivers panoramic scans at rates from 5 to 20 Hz. The vertical field of view is -24 / +2 degrees, the data throughput is 1.3 million points per second
- An automotive Lidar (Sick LD-MRS 400001) mounted on a 2-axis orientable turret allows to deliver high resolution point clouds in a H x V = 110 x 90 degrees.

Both Lidars export their data through an Ethernet interface, the turret that rotates the Sick Lidar is controllable through a serial link.

Computing unit: The computing unit is an Advantech ARK-3440F fanless embedded PC, with a i7 2.53GHz processor, 8 GB of memory and a 256 Go SSD hard drive, with placeholders for PCI extensions. The PC is running Ubuntu 16.04.

Communications: The robot is equipped with WiFi communication, and an independent remote emergency stop.

Supporting software: Mana software is composed of Genom3 components, that exploit the ros_comm middleware stack. The components are supervised by means of TCL scripts.

5.3.2 Minnie

Minnie is a robot very similar to Mana, with the following differences:

- The **platform** is a RMP440 model, with the same physical characteristics as Mana (the only difference being the on-board motor control electronics, which exhibits the same interface)
- The **exteroceptive sensor** suite is currently made of a Velodyne HDL 32 panoramic Lidar, composed of 32 independent telemeters, delivers panoramic scans at a rate from 10 Hz. The vertical field of view is -30 / +10 degrees, the data throughput is 700,000 points per second.

The rest of this suite is currently being procured. It will be composed of three stereoscopic benches: two fixed wide angle benches at the front and rear of the platform, and one orientable stereoscopic bench mounted on a mast on top of the robot.

- The fixed benches are similar to the HazCam configuration of the MER HazCams. They are made of FLIR USB3 1/3 inch 1 MPixel cameras and Theia 1.7 mm lenses yielding 120 degrees horizontal field of view. The orientable bench is similar to the NavCam configuration of the MER. It is made of FLIR USB3 cameras and Cinegon 4.8 mm lenses, yielding a 90 degrees horizontal field of view.
- The computing unit is made of two Advantech ARK-3440F fanless embedded PCs.

5.3.3 Ground station

The ground station is made of:

- A WiFi access point with 200m range, which can be extended with an additional remote access point connected via a directional antenna,
- A routing PC,
- A GPS base, which corrections are broadcasted through WiFi.

5.3.4 CNES SEROM Mars Yard

The SEROM facilities at CNES will also be made available to perform a subset of our validation and demonstration scenarios.

For safety reasons, the Mars yard is bounded by slopes steeper than the authorised threshold for navigation, so that the rover will remain within this field when moving autonomously.

The Mars yard is roughly 80m long and 50m wide. The aerial view of the terrain shown in the following figure was taken from a plane in August 2011.

Figure 75: Aerial View of CNES Mars Yard (Credits CNES)

The outdoor terrain is made of pozzolan and of stones of various sizes and shapes. It consists of areas and features with variable density and geometry to meet various requirements, depending on the purpose of the tests (perception tests, locomotion tests, localisation tests, etc.). It will therefore be possible to perform tests on :

- Areas with various density of stones,
- Areas with surmountable or insurmountable obstacles,
- Flat areas,
- Areas with slopes,
- Sandy areas.

The following figures illustrate the various types of terrains available within the yard.

Figure 76: Area with a high density of small rocks

Figure 77: Area with large diameter rocks, i.e. forbidden obstacles for the autonomous navigation mode

Figure 78: Flat and obstacle free area

Figure 79: Sandy dune on the left hand side of the yard

Figure 80: Valley on the left of the dune

6 Conclusion

The present document covered all the items required to support the development phase : detailed scenarios and use cases, system modeling, detailed architecture of DFPCs and detailed design of the whole.

It presented the detailed scenarios and validation strategy that should be implemented to first validate InFuse is ready for integration in futur systems in the scope of the SRC SPACE ROBOTICS, next to qualify InFuse performances in terms of resources consumptions, localisation and DEM accuracy. Scenarios were designed to really focus on key data products that InFuse should build by introducing use cases dedicated to each of them. The validation strategy relies on offline processing to conduct the study of algorithms and on on-line processing to validate their integration and real-time behaviour. The online processing the either done thanks to a simulated rover o a real rover, in order to ease the transition from the development phase to field testing.

Following the definition of key scenarios, the system modeling was presented to give a global view of all entities involved to implement our reference scenarios. The various systems and subsystems were defined, as well as their relationships. It exposed the principle of the integration process of InFuse with EGSE that will mainly rely on the robotics middleware ROS. Moreover, the system modeling emphasized the links between InFuse and other parts of the system, like the turret, sensors and the locomotion system.

The document went on with a description of DFPCs that have to be implemented. This detailed architecture described the content of DFPCs that were decomposed into DFNs. The idea here was to identify common functions shared among DFPCs as well as their internal behaviour.

The last chapter will gather the detailed design of DFPCs and DFNs. The work is still on progress and this chapter will be kept updated during the development phase.

Finally, the document includes 4 appendices. The first one deals with the definition of frames and mathematical conventions to manipulate rotations and transformations. The second appendix lists requirements that should be verified or tested. The next one is the template proposed to describe DFNs and DFPCs. And the last one proposes a first design of the data product manager.

7 Appendix

7.1 Definition of Reference Frames

This appendix sets out the various frames that are required to manipulate and exchange data in InFuse and over its external interfaces. We propose to follow aerospace convention or stick to common conventions as much as possible.

The coordinate systems are right-hand Cartesian and the positive rotational direction is defined counter clockwise around the selected axis, when this axis points toward the observer. Units must be expressed in the International System.

We now describe frames, starting from geo-referenced ones down to local ones.

7.1.1 Geo-referenced frames

ECEF : Earth Centered Earth Frame. This frame and its associated cartesian coordinates system allow to describe any point and transformation on earth. It can be used to define the **AbsoluteFrame (AF)**. By definition the AbsoluteFrame is the inertial frame attached to the planet.

WGS84 : World Geodetic System 1984. This frame and its associated polar coordinates system allow to localise any point on Earth.

IAU2000:49900 : Mars 2000. This and its associated polar coordinates system allow to localise any point on Mars.

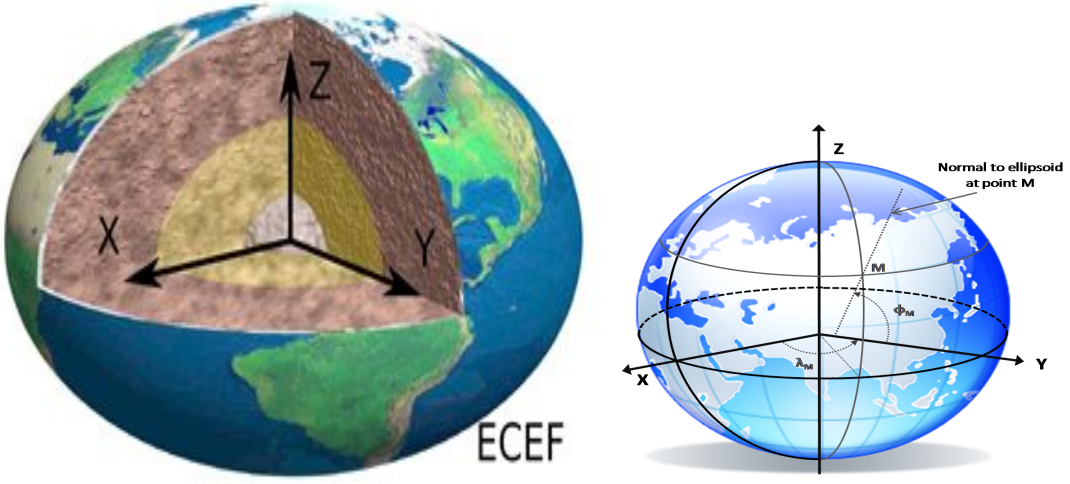


Figure 81: Geo-referenced frames (ECEF)

In order to use a most common convention of the robotics community with Z upwards, we define and recommend the use of the following convention for coordinates frames with a global reference :

ENU : East North Up. This convention is used to set up a cartesian frame on a local plane tangent to the ellipsoid. It can be used to define the GlobalTerrainFrame (GTF) and the LocalTerrainFrame (LTF). By definition, the GlobalTerrainFrame is a reference frame in which is defined a whole navigation mission (a long term path).

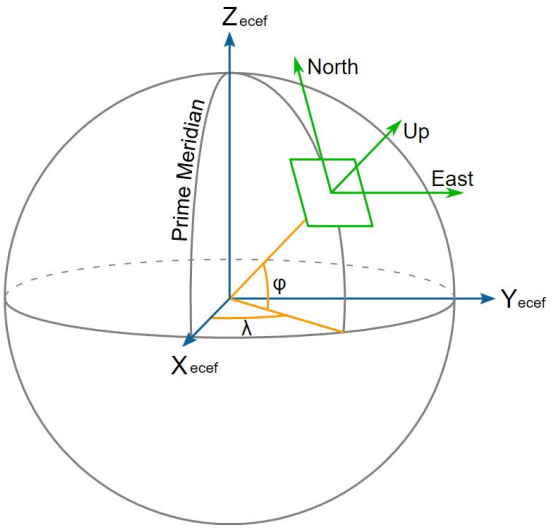


Figure 82: Local terrain frame

The LocalTerrainFrame is a reference frame in which are defined the local trajectories. During a travel sequence, this frame remains fixed with the planet, but becomes reset at the start of a new travel sequence. The rationale for this is that a travel sequence requires a stationary reference frame in which to measure position and navigation maps.

7.1.2 Local Frames

Now we address local frames attached to some hardware components.

RoverBodyFrame (RBF) : This is the frame associated to a given stable and salient reference point on the rover chassis. The frame origin is fixed with the rover, and thus moves as the rover moves. This is an intermediate frame to ease length measurements between the various parts constituting the rover.

As it is widely used on the field of ground-based mobile robotics, the following convention has been chosen for the rover body frame:

- The x-axis, X_RB, lies towards the front of the rover in the nominal direction of travel.

- The z-axis, Z_RB, lies vertically upwards, antiparallel to the gravity vector when the rover is on flat, horizontal terrain.
- The y-axis, Y_RB, completes the orthogonal right-handed set, and will lie to the left of the rover.

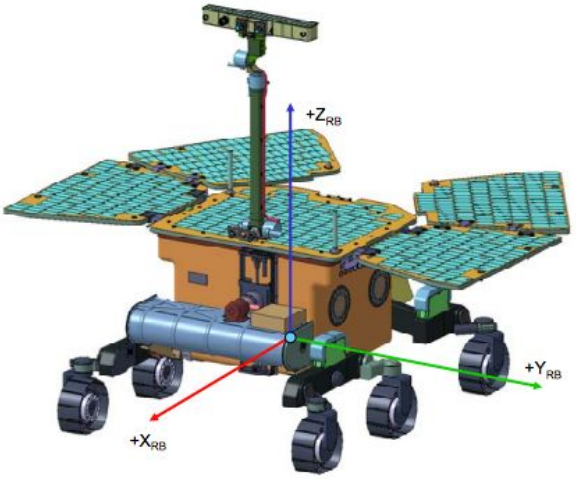


Figure 83: Illustration of the RoverBodyFrame (RBF) (EADS-Astrium/ExoMars)

RoverNavFrame (RNF) : This is the frame associated to the rover body for navigation. The frame origin is fixed with the rover, located at the nominal Planetary surface, centered under the rover turn-in-place rotation axis.

In order to keep the same convention all along the transformation chain from a sensor to the AbsoluteFrame, we have chosen the convention Z upward, X forward and Y leftward.

CameraFrame (CF) and **ImageFrame (IF)** : These are the common frames attached to a camera and its sensor that allow to use the simplest projection matrix while respecting the common convention on image coordinates. This frame definition can be extended to many kind of sensors, like stereo bench, LIDAR, TOF camera, etc.

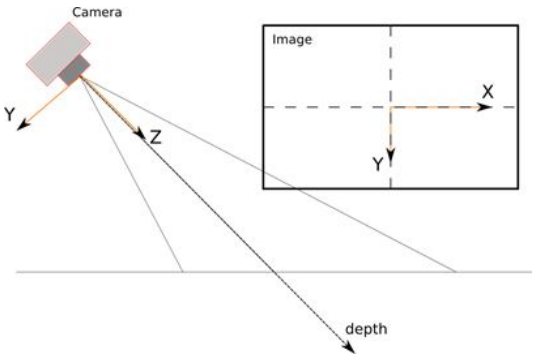


Figure 84: Illustration of the CameraFrame (CF) (left) and ImageFrame (IF) (right)

The current set of reference frames may be complemented with additional ones when applicable, depending on the type of rover and the final set of sensors and actuators used in a given implementation. These additional frames shall be included in the implementation description.

7.1.3 Mathematical Conventions and Notations

This section describes the mathematical conventions used to express rotation, translation and transformation.

We assume here, that we will only use transformations. The transformation from frame A to frame B converts the coordinates of a point P expressed in A to its coordinates expressed in B. We propose the following notation :

P_A = T_A2B * P_B

Transformations can be represented either by a transform matrix (using homogeneous coordinates) or a quaternion plus a translation. Given the expression above, there is no ambiguity for T_A2B as a transform matrix. Therefore, we will only detail the quaternion case.

T_A2B = (T_o_A, T_q_A2B), where T_o_A is the origin of A in B and T_a_A2B is the transform quaternion from A to B. TO BE FURTHER DETAILED.

7.1.4 References

[1] Astrium, "ExoMars Rover Vehicle: Coordinate Systems Specification", http://emits.sso.esa.int/emits-doc/ASTRIUMLIM/Exomars_Rover_Cameras/EXM-RM-SYS-ASU-00101_IssB_Coordinate-Systems-Specification.pdf

[2] NASA JPL, 2003, "Mars Exploration Rover Coordinate Systems Relevant to Imaging and Rover Motion Counter", https://scholar.google.com/scholar?cluster=11909632597052415878

[3] Titov et al, 2009, "Venus Express: Highlights of the Nominal Mission", https://scholar.google.com/scholar?cluster=2440509023330886703

[4] Glassmeier et al, 2007, "The Rosetta Mission: Flying Towards the Origin of the Solar System", https://scholar.google.com/scholar?cluster=11732370026057599015

[5] Koschny et al, 2007, "Scientific Planning and Commanding of the Rosetta Payload", https://scholar.google.com/scholar?cluster=10780063107272110226

[6] Diaz del Rio, 2016, "Trace Gas Orbiter (TGO) Spacecraft Frames Kernel", https://naif.jpl.nasa.gov/pub/naif/EXOMARS2016/kernels/fk/em16_tgo_v01.tf

[7] Justin Maki, 2013, "MSL Coordinate Systems for Science Instruments", https://an.rsl.wustl.edu/msl/mslbrowser/helpPages/MSL_Coordinate_Frames_Mar5_2013.pdf

7.2 Design Requirements

Design requirement are provided in the spreadsheet InFuse_D5.2_APPENDIX_REQUIREMENTS that accompanies this document.

7.3 Technical Note on DFN and DFPC Specification

7.3.1 Scope of this Note

This appendix sets out a proposed template for the definition of Data Fusion Nodes (DFN), and another one for the description of the various Data Fusion Processing Compounds (DFPC) outlined in D4.1. It sits at a "pre-implementation" description level, the lowest level before code lines.

7.3.2 Definitions

Data Fusion Nodes (DFN) and Data Fusion Processing Compounds (DFPC) have already been defined in D4.2 (see e.g. appendix 1 Glossary). Here we give some more details.

7.3.2.1 DFN

Atomicity A Data Fusion Node is an atomic processing entity, in the sense that it fulfills a single given basic function. It is the smallest brick, for the purpose of the CDFF, into which we break a more complex processing task. At a conceptual level, a DFN is completely defined by:

- Its purpose
- The data types of its input(s) and output(s)

Interfaces The only control interfaces exhibited by a DFN are *configure* and *process* (e.g. see file LaserFilterDFN.pdf).

Internal makeup A DFN may be made up of several smaller functions, but these functions and their output/input are not exposed. For instance, an ImageLineSegmentExtractor DFN is made up of the sequence ComputeImageGradient, ThresholdImageGradient, ChainThresholdedGradients, ChainLinearApproximation, but this sequence, which may include some controls, remains completely internal to the DFN and is not exposed.

7.3.2.2 DFPC

A DFPC always generates at least one data product. It is an organized set (a compound) of DFNs, with determinate data and control flows controlled by the Orchestrator. It may additionally maintain an internal data structure, under the responsibility of the Data Product Manager.

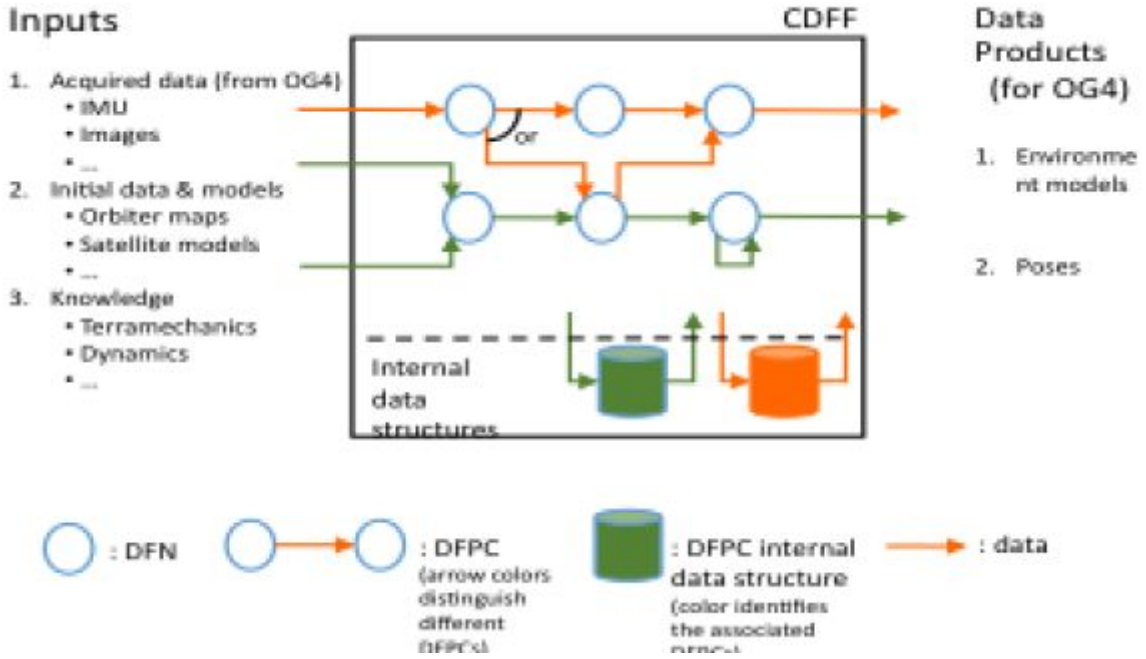


Figure 85: A simple schematic view of DFNs and DFPCs.

Figure 84 shows how DFNs can be put together to form DFPCs (two DFPCs on this figure). A few comments:

- A DFPC always links input data to one (or more) data product
- The control scheme of a DFPC is not “hard-wired”, in the sense that the sequence of DFN calls can vary, depending on the context (input parameters of the DFPC, intermediary results of DFNs). The control is implemented by the Orchestrator.
- The interfaces with OG2 have been defined in D4.1 and D4.2, but the associated data structures still need to be defined

The interfaces with OG4 have been defined in D4.2, they comprise the “acquired data”. The possible additional inputs (Initial Data and Models) still need to be defined - note the “knowledge” input are not and will not be explicit, and are actually implicitly considered in the implementation of the DFNs.

- A DFPC may theoretically be made up of a single DFN (although we don't have such a case in our list of DFPCs)

A table providing a detailed view of all interfaces and workflow between OG2-OG3-OG4 can be found in Appendix 7.5.

7.3.3 DFN Template

7.3.3.1 DFN Template Elements

Template element	Remarks
1. Generic description	Not much to say: must be present
2. Input(s) and Ouput(s) data	Data here means both: ● Actual data (e.g. Image16bit) ● Metadata (e.g. CameraParameters, Timestamp) A DFN cannot work without these data structures.
3. Input(s) Parameters	Can be provided by: ● Configuration file (e.g. Threshold, Size of patch) ● Output of another DFN (e.g. KF reinitialization)
4. Estimated performance and cost	This should be represented, if possible, in an adequate cost/performance space. This information should make it possible to define a performance measure and a cost measure for a resulting DFPC.
5. External library dependencies	Straightforward
6. Diagnostic capacities	Includes: ● Errors/warnings at runtime (e.g. unexpected datatype, out-of-range parameter...). ● Log capabilities (e.g. try/catch results written in a log file) ● Output reports (e.g. “if the image is all back…”) Fault Detection and Identification is the responsibility of the DFN. However its Recovery (when possible) is made at the DFPC level and is part of the Orchestrator.
7. Unit test	This must be provided with the code, along with the dataset used for validation.

7.3.3.2 Towards a Typology/Taxonomy of DFNs

It may be interesting to categorize DFNs, from both a description/documentation point of view, and mostly from an implementation point of view. Being entirely defined by their input and output data types, the DFN categorisation naturally induces a categorisation of data types.

This categorization is yet to be done, and will lead to an object-oriented implementation of DFNs.

7.3.3.2.1 DFN Characterization

A DFN is characterized by:

- A dictionary of Data Fusion implementations **(DFI)** fitting the DFN definition
- A cost/performance space representation **(if possible)** of each implementation, enabling the use to choose which DFI to use (5.)
- A set of validation tests (Added after implementation) (7.)

7.3.3.2.2 DFI Characterization

There can be different implementations of the same Data Fusion Node (e.g. a Visual Point Feature extractor DFN can be implemented with Harris Features or SURF). The different implementations are called Data Fusion Implementations (DFI) and characterize the given DFN.

Each DFI is characterized by:

- Its input parameters (3.)
- Its external library dependencies (5.)
- Its diagnostic capacities (6.)

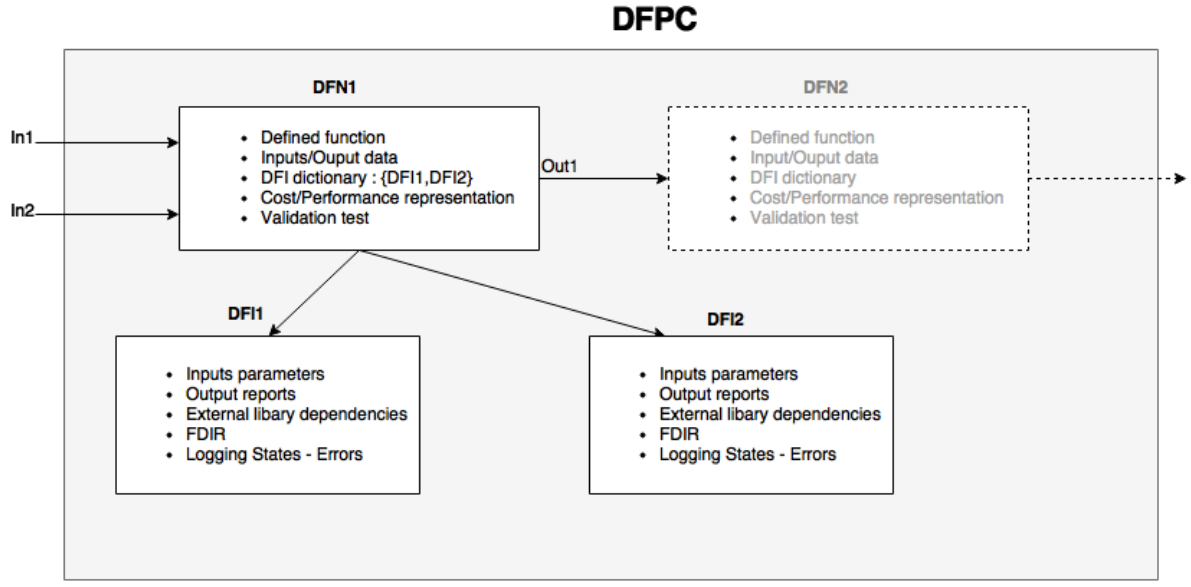


Figure 86: Example of a possible implementations of a DFN

7.3.4 DFPC Description Template

The specification of a DFPC is split into three main parts:

- Data Flow description: this is a purely functional description of the elementary processes (the DFNs) that compose a DFPC and their relations, seen only from a data-flow point of view. The goal of this description is to identify the list of required DFNs to build a DFPC.
- Data Product Management: this part describes the shared data between the DFNs in the given DFPC, and the interfaces between this data and the various DFNs: memory calls, data cropping, etc…
  The section ends with further considerations about the Data Product Management, which in particular can handle data in-between DFPCs.
- Control description: a pure data-flow description is not operational, and in particular does not depict the sequence and logic of a DFPC. This section proposes a way to depict the control flow within a DFPC: order in which DFNs are called, DPM access to shared data, synchronicity of timestamped data, etc. The control flow will be achieved by the Orchestrator for implementation.

7.3.4.1 DFPC Data Flow Description

The DFPC data flow provides a layout and ordering of the different DFNs. It defines:

- Inputs/Outputs of the DFPC
- Inputs/Outputs of each DFN
- DFN types used
- Shared data between DFNs: even though data product management is not represented, it provides data for DFNs as inputs.

Find below example for LIDAR-PG-SLAM:

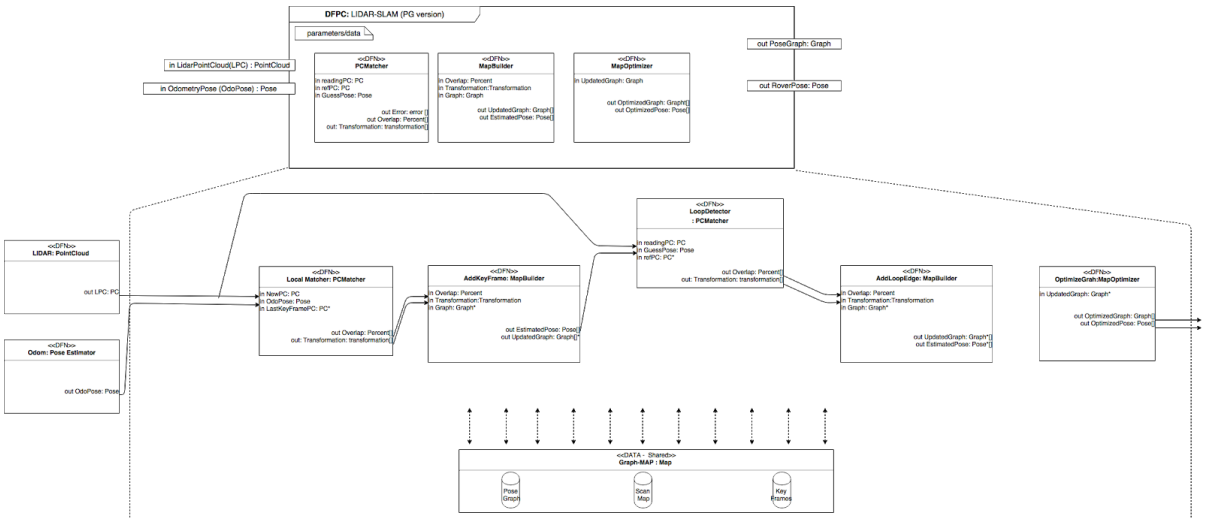


Figure 87: LIDAR-PG-SLAM

In this example, all inputs that are provided by DPM and outputs that are inserted in the DPM are represented with a * after their name.

7.3.4.2 DFPC Data Product Management

The Data Product Management of DFPC specifies:

- The shared data structures amongst the different DFNs
- The processing units that query and insert those data structures
- Specific managing processes

Note the DPM will also comprise *inter-DFPC* relations -- this will be further developed.

7.3.4.2.1 Shared Data Structures

The shared data structures are a list of data types that need to be accessed by different DFNs in the course of a DFPC execution, e.g. in SLAM, the map is updated several times at each different observation times by successive DFNs.

As an example, the shared data structure for Pose-Graph LIDAR-based SLAM would be the Map, that contains (note these will further be depicted using data types) :

- Pose Graph: List of poses and constraints
  - Pose: {x,y,z,t1,t2,t3} coordinates of the rover + frame in which they are defined
    - Coordinates: float
    - Frame: string defining the reference frame
  - Constraints: Transformation, and error
    - Transformation: rotation and translation obtained by ICP
      - Rotation: 3x3 Matrix
      - Translation 1x3 Vector
    - Constraint
      -
- KeyFrames: List of PointClouds and Poses
  - Pose: defined above
  - PointCloud: List of Points
    - Points: {x,y,z} coordinates of points in space
      - Coordinates: float

7.3.4.2.2 Processing Units - Queriers

A querier is a processing unit that accesses the shared data structure and crop it in order to feed the right input to a given DFN. They are non-generic (each DFN requiring a particular part of data), and are defined by their inputs and outputs.

Processing units for Pose-Graph LIDAR-based SLAM would be the following ones:

- Local Map querier:
  - Input : KeyFrames and Graph
  - Output: LocalMap (PointCloud)
- Graph Querier:
  - Input: PoseGraph
  - Output:PoseGraph

The hierarchy of queriers for PG LIDAR-based SLAM can be seen in the following figure:

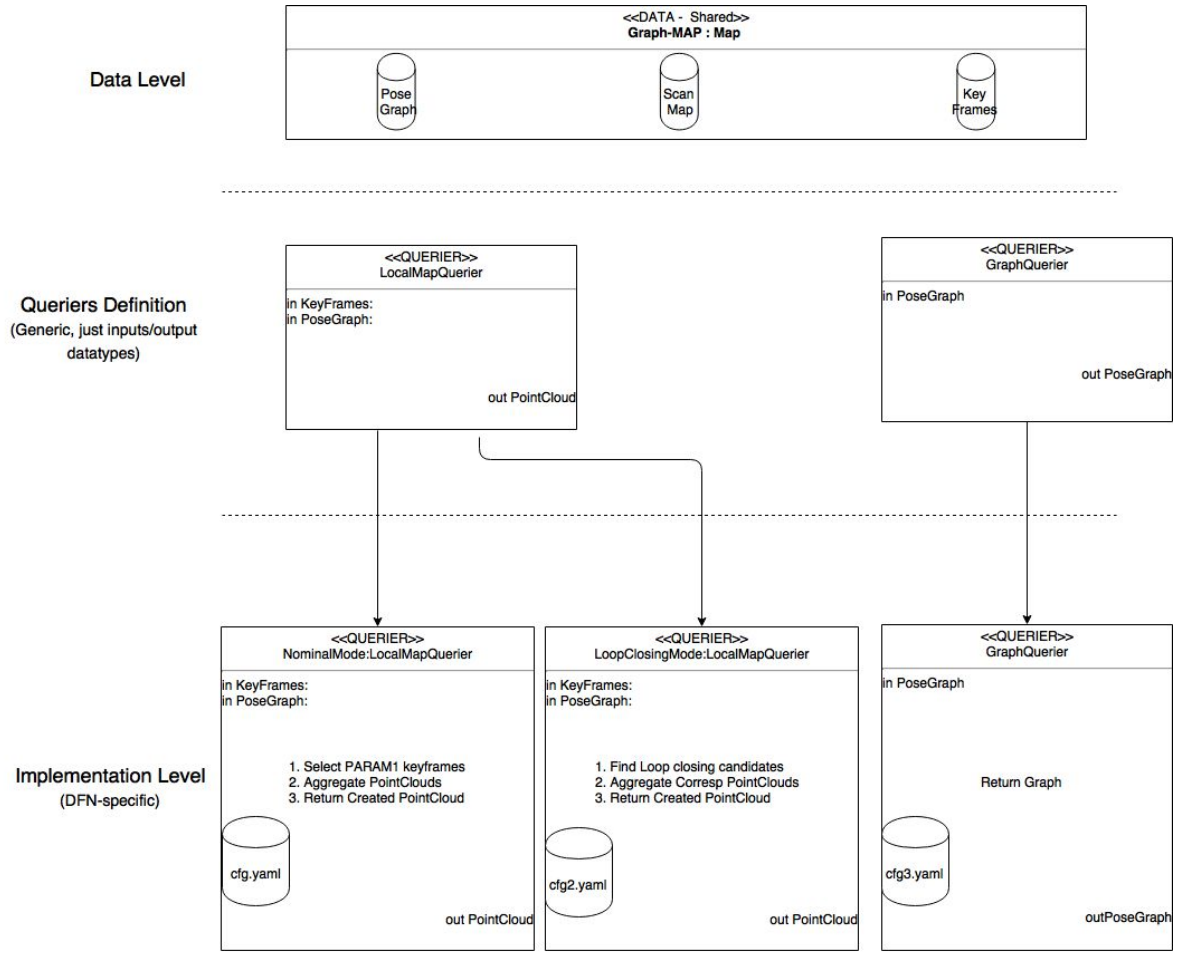


Figure 88: Hierarchy of queriers for PG LIDAR-based SLAM

7.3.4.3 DFPC Control Flow Description

The orchestrator is responsible for temporal rightful execution of a DFPC. Each DFN/Querier can be considered as a single process/thread, and the orchestrator organizes the execution of those DFN.

The control flow can be represented as an UML sequence diagram. Elements appearing on this diagram include:

- Objects appearing in the data flow
- Data product management units (queriers)
- The DFPC orchestrator itself is represented as an object sending signals the the different DFNs

The aim of the sequence diagram is to represent the temporal course of the execution of a DFPC. An example of such sequence diagram can be found below for the PG-LIDAR-based SLAM:

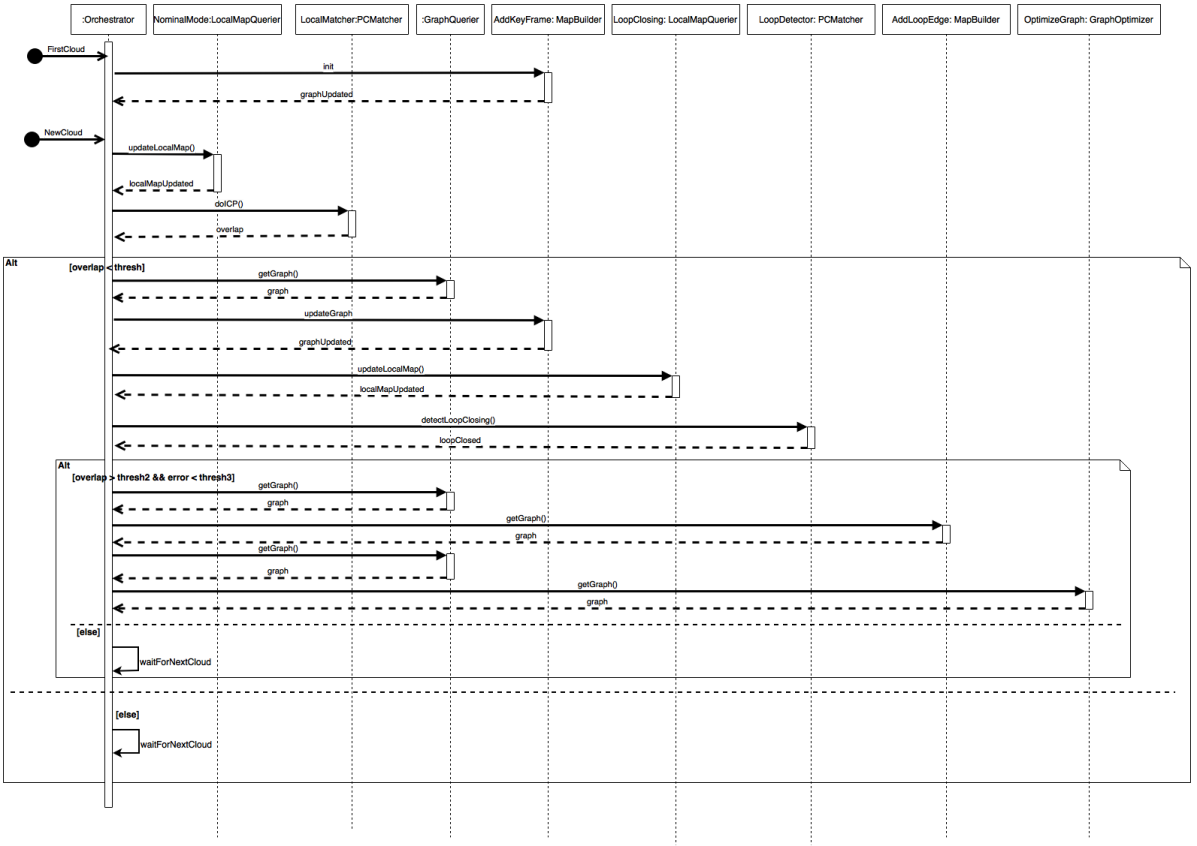


Figure 89: PG-LIDAR-based SLAM sequence diagram

In this diagram, the temporal aspect is vertical, and messages (corresponding to events triggered by the orchestrator) are represented by the arrows.

7.4 Architecture of the Data Product Manager

This section details the design of the DPM, along the following structure:

1. Introduction: general concerns, definition of the DPM

2. Use cases: the list of situations in which the DPM is at play. This will help to specify the data that have to be stored and managed by the DPM, as well as the associated processes.
3. List of functions to be ensured by DPM: it is a synthesis made by analysing the various use cases.
4. Implementation: this is the detailed design of the structure of the DPM and the associated functions.

7.4.1 Introduction

7.4.1.1 General concerns

The introduction of the Data Product Manager (DPM) within the InFuse CDFF comes from the following facts:

- The CDFF comprises various localization DFPCs that estimate the robot pose, in various reference frames and at various frequencies. A core function of the CDFF is the ability to fuse these estimates to enhance the precision of the robot pose estimate: this is handled by the DPM.
- There are numerous clients of the robot pose data product: OG2-ERGO first, and also some DFPCs of the CDFF, *e.g.* the ones that build terrain models. These clients should have a unified access to the robot pose estimate, whatever the selection and configuration of the localisation DFPCs: it is desirable to have a single interface for these clients to access the robot pose. The DPM provides this interface.
- Similarly, there are numerous clients of the terrain model data products: OG2-Ergo first, and also some localisation DFPCs that exploit an environment model: the DPM is the interface through which the environment models are accessed by the processes that need them.
- Like robot pose estimates, environment models can evolve over time (when past pose estimates have been refined for instance), and there may be dependencies between the various models (a request of the Digital Terrain Map can be made to build a Navigation Map upon a Digital Terrain Map for instance): there is a need to maintain the consistency of the built environment models, which is an additional role of the DPM.

In mobile robotics, there are strong interdependencies between the localisation and navigation processes, and it has been early reckoned that these two functionalities are two facets of the same problem that must be be solved concurrently: this leads to the development of SLAM approaches. However SLAM approaches mostly consider environment models dedicated to localisation, whereas other environments models are required for the autonomy of mobile robots (in the InFuse case, the DTM and the Navigation Map). Furthermore, localisation estimates can be produced using dedicated environment models that are not produced by a SLAM solution (in the InFuse case, Absolute Localisation

using the Orbital Map for instance). The DPM explicits these relations. Figure 89, shows these relations between the two kinds of data products built within the CDFF, and the four following kinds of DFPCs:

- *Localization DFPCs*, which produce position estimates without resorting to any environment models. These DFPCs are:
  - Wheel Odometry,
  - Visual Odometry,
  - Point Cloud Model-based Localization
- *SLAM Localization DFPCs,* which produce both position estimates and dedicated environment models. These DFPCs are:
  - Visual SLAM,
  - Lidar Pose-Graph Localisation

  which respectively produces the Visual Map and the Lidar Map.
- *Map-based Localisation DFPCs,* which produces positions estimates on the basis of existing environment models. These DFPCs are:
  - Absolute Localisation
  - Visual Map-based Localisation
  - Lidar Map-based Localization

  which respectively exploits the Orbital Map, the Visual Map and the Lidar Map.
- *Environment Modelling DFPCs*, which exploits the robot position estimates and produces environment models. These DFPCs are:
  - DTM building
  - Navigation Map building

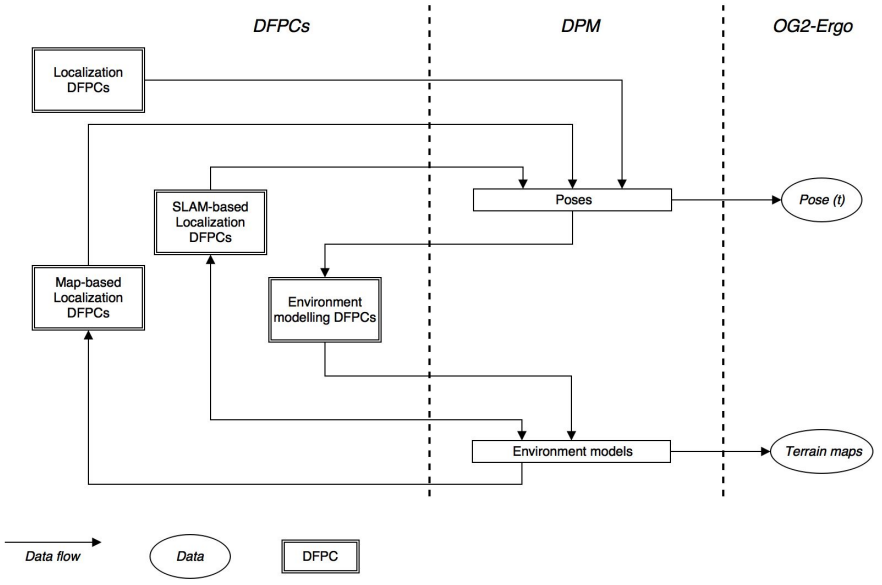


Figure 90: Relations between the various kinds of DFPCs and the DPM

7.4.1.2 Definition of the DPM

The role of the DPM is to handle the selection, structuring and storage of all the data processed or produced by the CDFF that may be re-used, either internally by InFuse processes or to satisfy OG2-ERGO requests. Additionally, it is the interface through which robots expose and retrieve the CDFF data products in multi-robot scenarios, and also the interface through which ground operators can access the CDFF data products.

The DPM can be seen a robotics-dedicated Geographic Information System (GIS). With respect to the activated DFNs and DFPCs in the CDFF, the DPM processes the data insertion requests. Internally, it manages all the spatial related data by implementing insertion, deletion or update functions, aiming at satisfying future needs for data products and storage constraints.

7.4.2 DPM use cases

This section depicts the cases in which the DPM is involved in the various scenarios depicted in D4.1 (Technical Trade-Offs Analysis). Each use case is first qualitatively presented, the involved DFPCs and their relation to the DPM are listed, and the required DPM functions and the data to store / manage within the DPM are then listed. When needed, a picture representing the data structures and functions at play is provided.

7.4.2.1 Integrate estimation of past poses and motion estimates

Case 1: fusion of wheel and visual odometries

- Description: Visual odometry (VO) processes two stereoscopic frames acquired at times t and $t+\Delta t$, and estimates the motion $M^{VO}_{t\rightarrow t+\Delta t}$, which is made available at $t+\Delta t+\delta t$, where δt is the visual odometry processing time. Meanwhile, the robot is continuously moving, and higher frequency motion estimates are provided by the wheel odometry (WO). At the time $t+\Delta t+\delta t$ of the production of the VO estimate, the VO and WO estimates $M^{VO}_{t\rightarrow t+\Delta t}$ and $M^{WO}_{t\rightarrow t+\Delta t}$ are fused, and this past information has to be propagated up to the present time, by re-integrating the WO motion estimates provided between $t+\Delta t$ and $t+\delta t$.

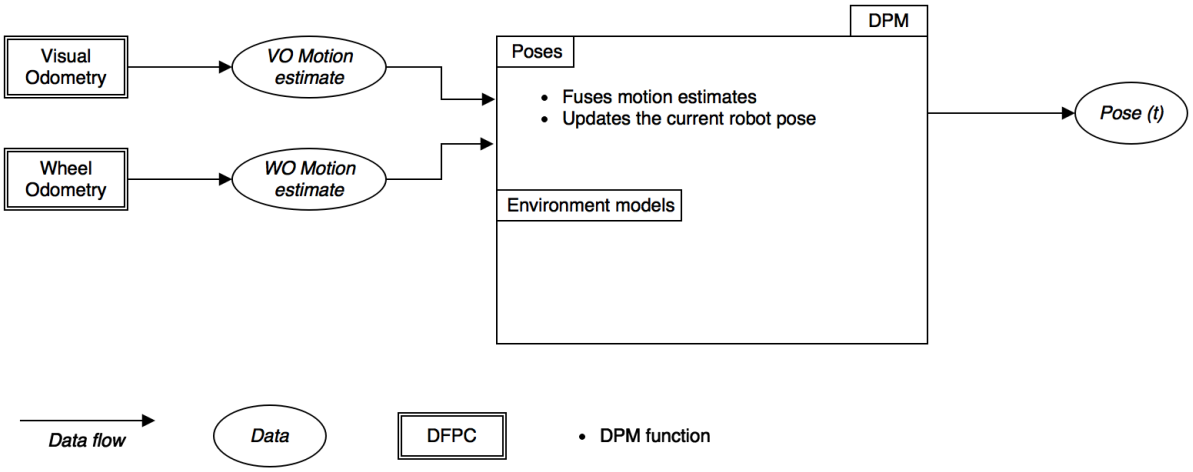


Figure 91: Interfaces with the DPM for the fusion of wheel and visual odometries

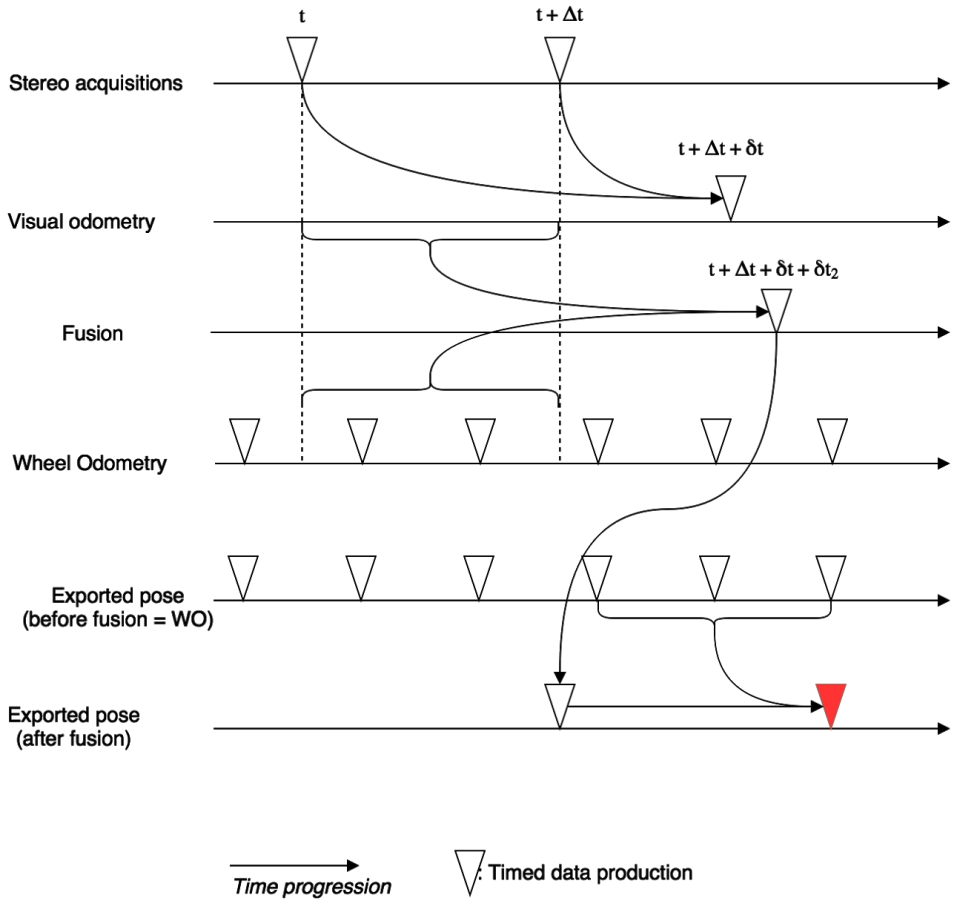


Figure 92: Fusion of two independent motion estimation processes, Wheel Odometry and Visual Odometry.

Because of computing time, VO can only estimate the motions between t and $t+\Delta t$ at a later time $t+\Delta t+\delta t$, and this motion estimate is fused with the associated motion estimated by WO at time $t+\Delta t+\delta t+\delta t2$. Once fused with the WO estimate of the same motion, the exported pose (in red) is updated by composing the newly estimated position at past time $t+\Delta t$ and the motions measured by odometry since then.

- Involved DFPCs: Visual Odometry and Wheel Odometry serve the DPM with motion estimates (more generally, any localization DPFC that produces a motion estimate between two positions)
- Data to store / manage inside the DPM:
  - History of motion estimates produced by the independent DFPCs
- Required functions
  - Store the history of poses of independent un-synchronized estimators
  - Fusion of two independent motion estimates
  - Recomputation of current pose after the fusion of past motion estimates

Case 2: Absolute localization

- Description: an absolute position estimate is produced. It allows to re-estimate the whole trajectory made since the last absolute position has been produced (this function is mainly launched at the end of a mission, to build a Total Rover Dtm, see following use case)
- Involved DFPCs: Absolute Localisation, which serves the DPM with pose estimates (similarly, also Vision Map-based Localization, Lidar Map-based Localisation serve the DPM with pose estimates in the associated reference frames).
- Data to store / manage:
  - A history of the poses that can be revised after an absolute localisation estimation
- Required functions:
  - Fuse the absolute pose estimate with the current pose estimate
  - Re-estimate the history of poses

Note that the two other map-based localisation DFPCs (Vision Map-based Localisation and Lidar Map-based localisation) interact with the DPM in a strictly similar manner.

7.4.2.2 DTM building

Case 1: Rover Map and Fuse Rover Map building

- Description: this is the nominal operation of the DTM Building DFPC. Point Clouds produced either by stereovision or a Lidar are fused into a DTM structure. The Rover Map DTM is expressed in the Rover Frame at the time of the Point Cloud production,

the Fused Rover Map is expressed in a dedicated frame (most likely corresponding to the frame at the time of the first Point Cloud production?)

- Involved DFPCs:
  - DTM building is a client of the DPM, to obtain the pose associated to the incoming Point Clouds
  - DTM building serves the DPM with its products
- Data to store / manage:
  - History of poses associated to Point Clouds acquisitions: some point clouds acquisitions are made without a pose computation associated to them (e.g. in the case of scientific area localization). We need to make sure that we are able to retrieve a pose each time an observation is performed.
- Required functions (prerequisites):
  - Deliver the pose associated to a Point Cloud acquisition
  - Store the produced DTMs

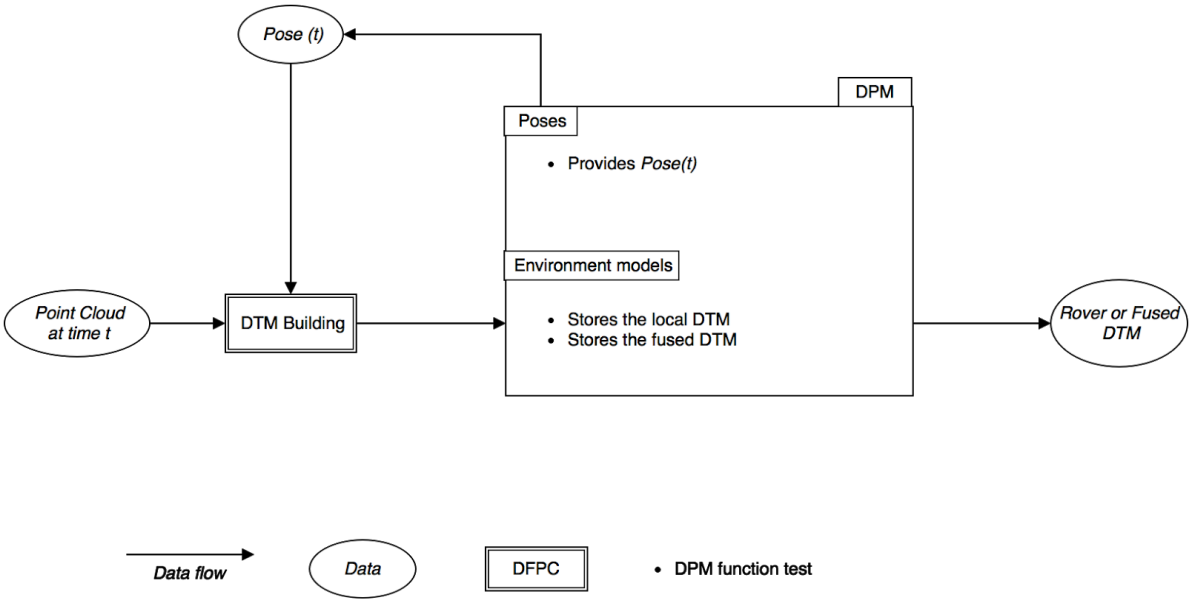


Figure 93: Role of the DPM with respect to the DTM Building DFPC

Case 2: Total Rover Map building

- Description: this is one of the service provided by the DTM Building DFPC, which consists in building a DTM that integrates a large set of Point Clouds (typically, to prepare a Getting-Back scenario). While the the Total Rover Map building could be done in-line as Point Clouds are produced, the benefits of building a Total Rover Map comes after the estimate of an absolute pose (or a loop closing event detected by a SLAM DFPC, or a map-based position estimate produced by one of the Map-based

Localisation DFPCs), which will yield the production of a spatially more consistent DTM.

- Involved DFPCs: DTM Building, which is a client of the DPM, to retrieve the poses associated to each Point Cloud acquisition.
- Data to store / manage:
  - the Point Clouds (or, most likely, the associated Rover Maps) acquired during a given time period
- Required functions:
  - Deliver the pose associated to a series of Point Cloud acquisitions

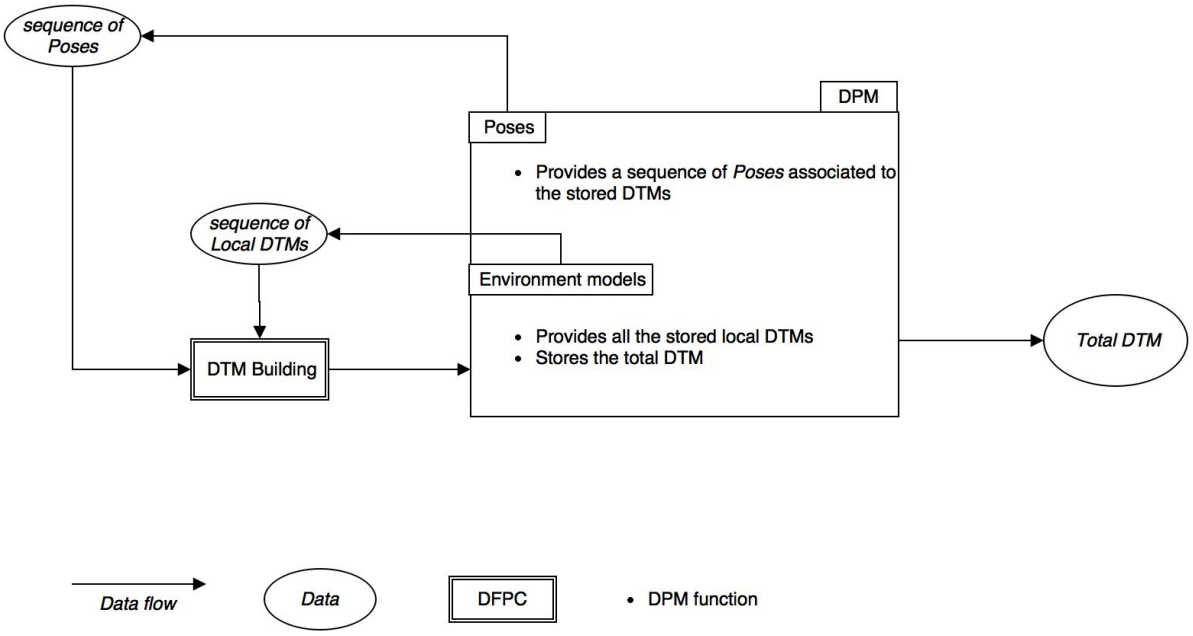


Figure 94: Role of the DPM with respect to the Absolute Localization DFPC when it produces the Total Rover Map

7.4.2.3 Exploitation of environment models produced by the CDFF

Case 1: localization with respect to localization maps

- Description: a rover pose is estimated relatively to a localisation map, which can either be:
  - The Visual Map (produced by the Visual SLAM DFPC) for the Visual Map-based Localisation DFPC
  - The Lidar Map (produced by the Lidar SLAM DFPC) for the Lidar-based Localisation DFPC
- Involved DFPCs:
  - The DFPCs that build maps serve the DPM with these maps: Visual SLAM and Lidar SLAM

- The DFPC that estimate poses with respect to maps are clients of the DPM, which provides them access to the required maps

- Data to store / manage:
  - The localisation maps: Visual Map, Lidar Map
- Required functions:
  - Provide the stored localisation maps (or a subpart of it)
  - Integrate the position estimates provided by the Map-based Localisation DFPCs

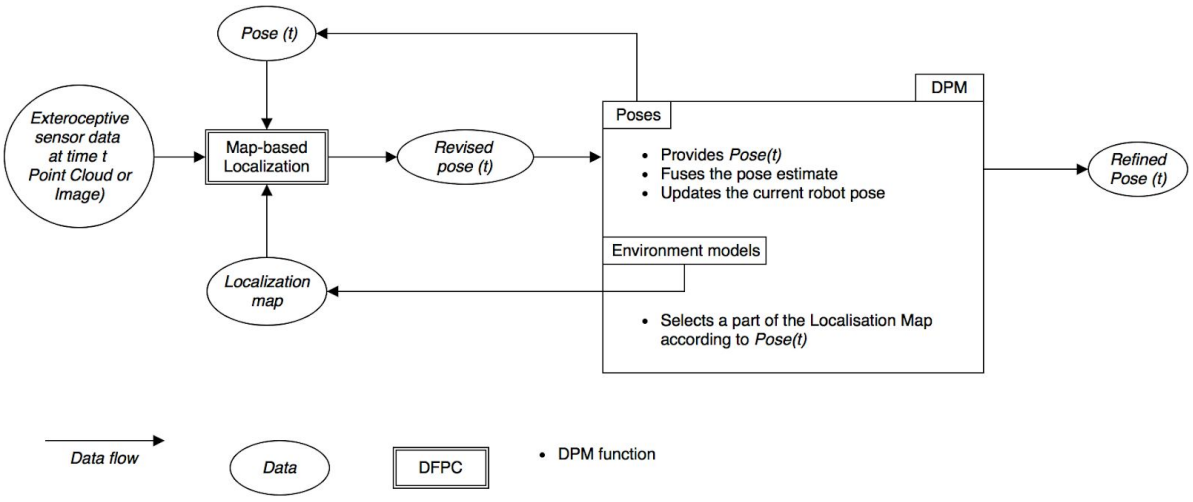


Figure 95: Role of the DPM with respect to the Map-Based Localization DFPCs

Case 2: Absolute localisation

- Description: the Absolute Localization DFPC requires a DTM (possibly with associated luminance information) to assess matches with the Orbital Map. Most often this DTM is a fused one, as a DTM built from a single Point Cloud acquisition does not cover a surface large enough to be matched with the Orbital Map.
- Involved DFPCs:
  - the Absolute Localisation DFPC is a client of the DPM : to retrieve a DTM, and to retrieve a subpart of the Orbital Map.
  - The Absolute Localisation DFPC serves the DPM with an absolute pose estimate
- Data to store / manage:
  - The DTM
- Required functions:
  - Provide a DTM
  - Integrate the position estimate provided by the Absolute Localisation DFPC

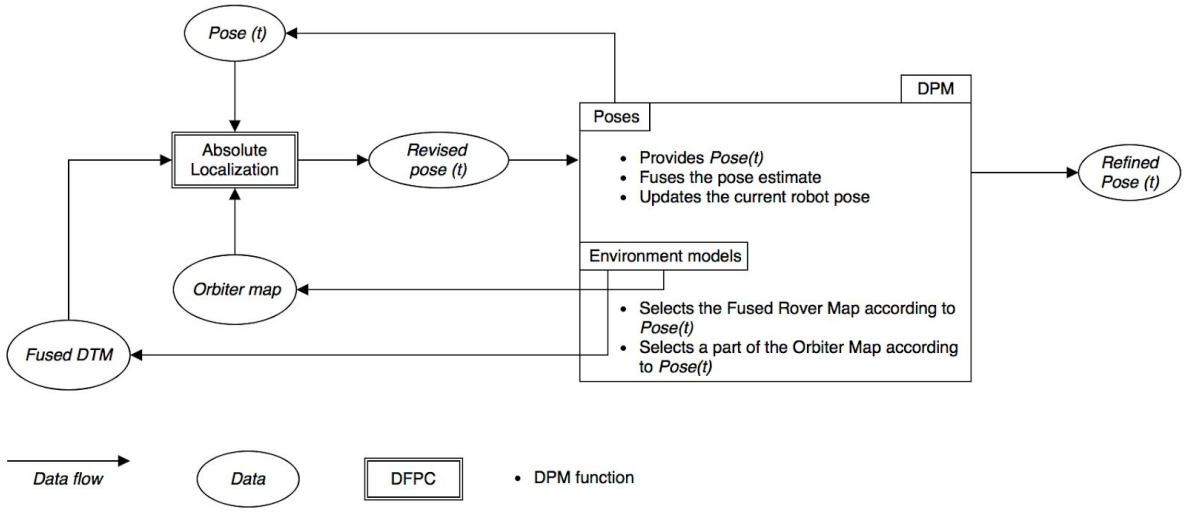


Figure 96: Role of the DPM with respect to the Absolute Localization DFPC

Note these two cases are very similar, with the slight difference that the DPM serves the Absolute Localization DFPC with the DTM, whereas the two other Map-based Localisation DFPCs exploit raw data.

7.4.2.4 Science target localisation

- Description: a scientific target is detected in an image by a process that is not within the InFuse CDFF. The detection corresponds to an area (most likely rectangular) in the image: its localisation in 3D must be provided by the DPM. This can be done:
  - by associating a bounding box in a memorized point cloud that has been acquired at a position close (or rather, equal) to the image acquisition position,
  - alternatively, by reprojecting the image area in the DTM corresponding to the area in which the target has been detected

The returned position is a 3D bounding box localized in the reference frame associated to the image acquisition.

- Involved DFPCs:
  - Science Target Localisation
- Data to store/ manage:
  - Point Cloud data associated to the image on which the science target detection is applied
  - Alternatively, DTM covering the area of the image on which the science target detection is applied
- Required functions:
  - Retrieve the Point Cloud data or the DTM associated to the area of the image on which the science target detection is applied

Note: to be applicable, the Science Target Localisation DFPC requires that 3D data are acquired along with the image that is processed to detect the science target.

7.4.2.5 Multi-robot use cases

In multi-robot setups, robots may benefit one from each other for both localisation and environment modelling purposes.

Case 1: inter-robot localisation

- Description: One robot A perceives and localizes a robot B with respect to its current pose. This relative localization information is used to refine both robots position.
- Involved DFPCs:
  - Point Cloud Model-based localisation, which estimates the relative robot's position
- Data to store / manage:
  - The history of both robots poses
- Required functions:
  - Re-estimate the history of the robots' poses

Case 2: one robot exploits environment models produced by an other one for localization purposes.
This case is very similar to the "localization with respect to localization maps" case, with the sole difference that the maps used are provided by another robot.

- Description: One robots exploit the Localisation Map produced by an other robot to localize itself.
- Involved DFPCs:
  - Map-based robot localisation
  - The DFPC that estimate poses with respect to maps (Visual Map-based Localisation and Lidar-based Localization) of one robot are clients of the DPM of the other robot, which provides them access to the required maps
- Data to store / manage:
  - The localisation maps: Visual Map, Lidar Map
- Required functions:
  - For the robot providing the localization map: provide the stored localisation map (or a subpart of it)
  - For the robot exploiting the localisation map: Integrate the position estimates provided by the Map-based Localisation DFPCs

Case 3: one robot exploits environment models produced by an other one for navigation purposes

- Description: to plan an itinerary towards a goal in an unmapped area, one robot queries the other one with the Navigation Map on a given area
- Involved DFPC: none
- Data to store / manage:
    - The navigation map
- Required functions
    - Assess whether a navigation map is available on the requested area, deliver it

Note: this use case is tightly associated to the itinerary planning functionality: this functionality should state which unknown areas are to be crossed, so that the DPM can be queried the associated parts of the navigation map.

7.4.2.6 Serve the operators

- Description: The DPM being in charge of memorizing and structure the DFPC data products and a selection of the acquired raw data, it is the component that is in charge of delivering the data requested by the operators -- e.g. at the end of a traverse mission.
- Involved DFPCs: None
- Data to store / manage:
    - The history of the robot poses
    - A selection of the raw exteroceptive data acquired by the rover
    - The built environment models
- Required functions:
    - Deliver poses, raw data or parts of environment models corresponding to either a time interval or a spatial area

7.4.3 Synthesis: DPM functionalities

7.4.3.1 DPM services

The core services the DPM is offering are the following:

1. Produce at high frequency a continuous position estimate, expressed in any specified frame
2. Integrate the localisation estimates produced by all the localization DFPCs
3. Associate a position estimate to any exteroceptive raw data (image, point cloud) or derived product (point cloud obtained from stereovision), expressed in any specified reference frame
4. Serve both the OG3-Ergo and CDFF processes with environment models

7.4.3.2 Stored and managed data

The data stored and managed by the DPM are the following:

- History of rover poses:
  - A short term history of motion estimates produced by the Wheel Odometry and Visual Odometry DFPCs. This is required to fuse the motion estimates of these DFPCs and the pose estimates produced by any other localisation DFPC (both SLAM and Map-based Localisation DFPCs), and to propagate the fused information up to the current time.
  - A long term history of the poses associated to Point Cloud acquisitions. This is required to update the DTM and Navigation Maps after the revision of the past poses associated to Point Cloud acquisitions, the revision of these poses being triggered by any of the SLAM-based and Map-based localisation DFPCs.

- History of acquired data and built environment models:
  - History of the Point Cloud data exploited to build the DTM and Navigation Maps. This is required to update the DTM and Navigation Maps after the revision of the past poses, triggered by any of the SLAM-based and Map-based localisation DFPCs (note that for the sake of memory storage, the Rover Map associated to each Point Cloud are memorized, not the Point Clouds).
  - History of the built environment models. This is required to serve the DFPCs that exploit environment models, other robots in the multi-robot case, and OG2-Ergo. The Localisation Maps are simply stored as they are produced by the SLAM DFPCs. As for the DTM and Navigation Maps, they can be built on demand on the basis of the history of data acquisition.

7.4.3.3 DPM functions

To achieve the DPM services, the functions the DPM must achieve are the following:

- Select the motion estimates and poses to store
- Fuse independent motion estimates and pose estimates
- Update the current rover pose after the fusion of motion or pose estimates
- Re-estimate the rover poses stored in the long term history of poses (after a pose fusion has been performed)
- Associate a pose to raw data acquisitions or derived product -- either at the time of the acquisition, or for past acquisitions

- Provide the environment models and/raw data upon a request that specifies the needed area or time interval (Point Clouds, Localisation Maps, DTM and Navigation Map)

7.4.4 Architecture and implementation design

The DPM manages pose estimates and environment models: although these two kind of information are closely interrelated, for the sake of clarification and modularity we propose to separate them in two components.

7.4.4.1 Managing poses

The following figure illustrates the various poses that are produced by the Localisation DFPCs, as well as the poses corresponding to data acquisition. They must all be memorized, so as to allow the pose fusion processes to integrate data corresponding to past poses, and to produce updated past poses after the application of a pose fusion process (e.g. to allow the re-building of a Total Rover DTM after a Map-based Localisation DFPC updates a rover pose).

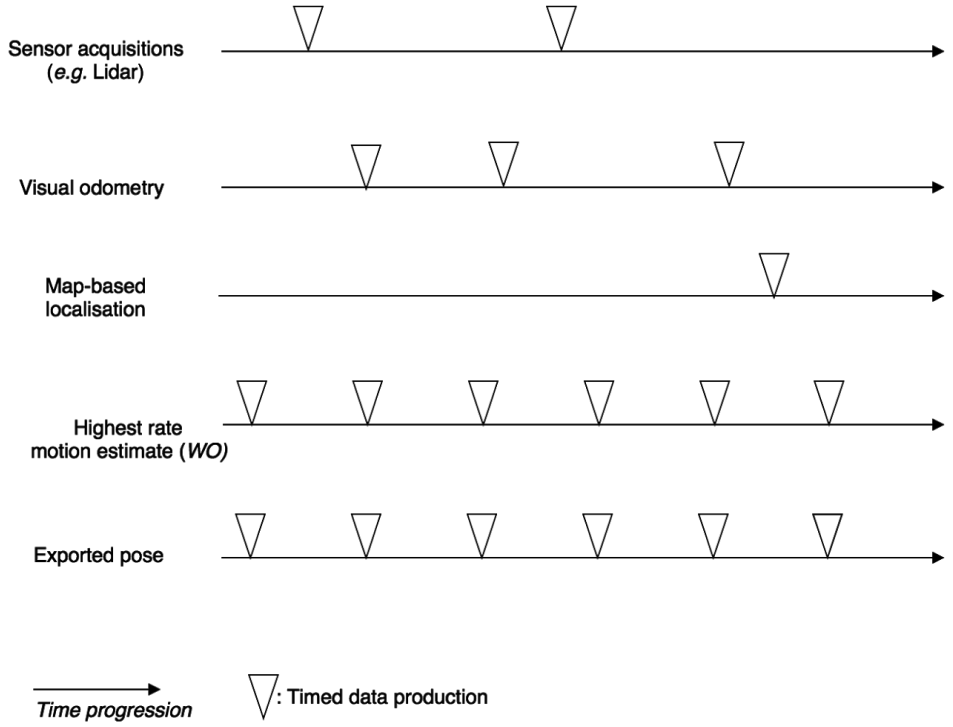


Figure 97: Timeline of the various pose estimates handled by the DPM.

The integration of the fusion processes, as well as the managing of the past poses history is illustrated by the next set of figures.

Position manager

Time line of a small trajectory.All references frame here are Z-up. 3 localization modules.

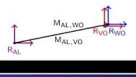
- Wheel Odometry (WO) in blue, reference frame R_{WO}
- Visual Odometry (VO) in red, reference frame R_{VO}: SAME AS R_{WO}
- Absolute localization (AL) in violet, reference frame R_{AL}

Time

Position manager

- Initial transforms known at the start t_0 of the mission: $M_{AL,VO}$ and $M_{AL,WO}$: THEY ARE FIXED, AND THE SAME
- A t_0 visual v_0 observation is made, and a first IMU reading is done

(t_0,v_0) Time



Position manager

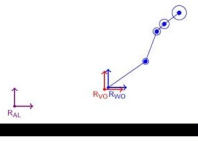
- Highest frequency DFPC starts producing transform : WO

(t_0,v_0) t_1 Time

Position manager

- Highest frequency DFPC starts producing transform : WO

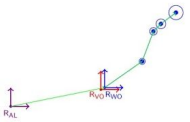
(t_0,v_0) t_1 t_2 t_3 t_4 Time



Position manager

- Highest frequency DFPC starts producing transform : WO
- As we have only one source, at t4, the fused history of poses is the same as WO (green)

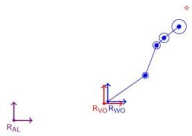
(t_0,v_0) t_1 t_2 t_3 t_4 Time



Position manager

- At time t5, a new video observation is acquired to produce visual odometry (red diamond, not related to a pose), BEFORE the next WO transform estimate

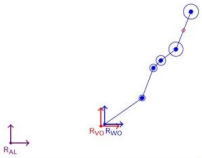
(t_0,v_0) t_1 t_2 t_3 t_4 (t_5,v_1) Time



Position manager

- At time t5, a new video observation is acquired to produce visual odometry (red diamond, not related to a pose), BEFORE the next WO transform estimate
- Some more WO estimations are made while the first VO estimation is computed

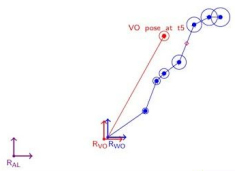
(t_0,v_0) t_1 t_2 t_3 t_4 (t_5,v_1) t_6 Time



Position manager

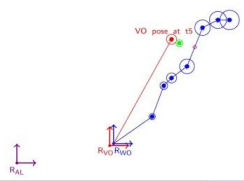
- At time t9, the result from the first VO is made available (in red)

(t_0,v_0) t_1 t_2 t_3 t_4 (t_5,v_1) t_6 t_7 t_8 t_9 Time



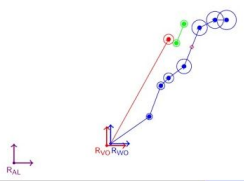
Position manager

- We can use this new estimate to estimate a pose at time t9, which is the fusion of the VO pose and interpolation of WO poses



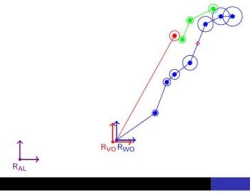
Position manager

- We can use this new estimate to estimate a pose at time t9, which is the fusion of the VO pose and interpolation of WO poses



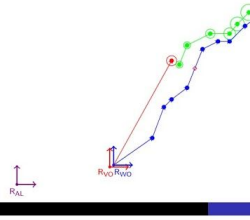
Position manager

- We can use this new estimate to estimate a pose at time t9, which is the fusion of the VO pose and interpolation of WO poses
- ... and correct subsequent observation thanks to the "prediction" step from KF



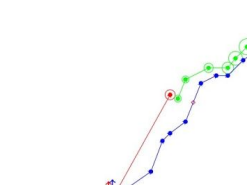
Position manager

- At time t13, a new visual observation is made (red diamond)



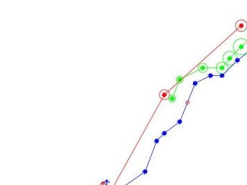
Position manager

- At time t13, a new visual observation is made (red diamond)
- Some more estimations are produced by WO (and corrected)



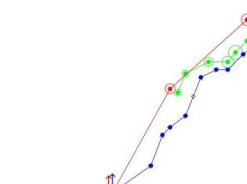
Position manager

- At time t16, the VO transform is estimated for t13



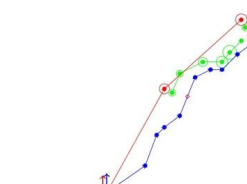
Position manager

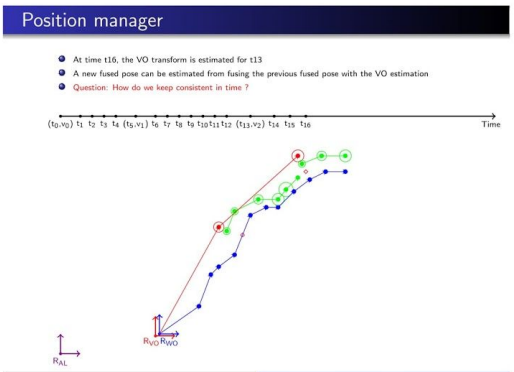
- At time t16, the VO transform is estimated for t13
- A new fused pose can be estimated from fusing the previous fused pose with the VO estimation

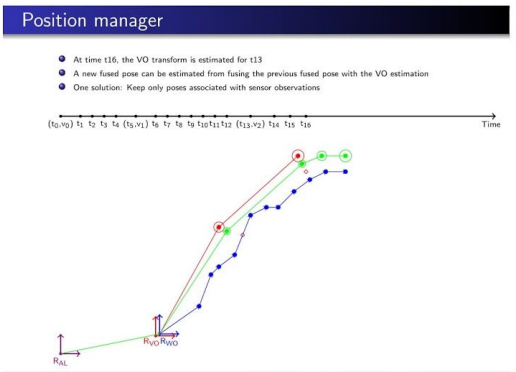


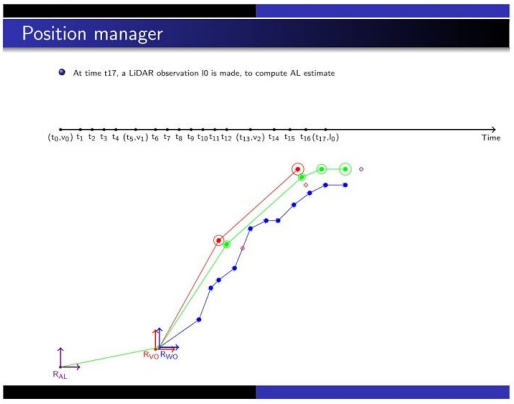
Position manager

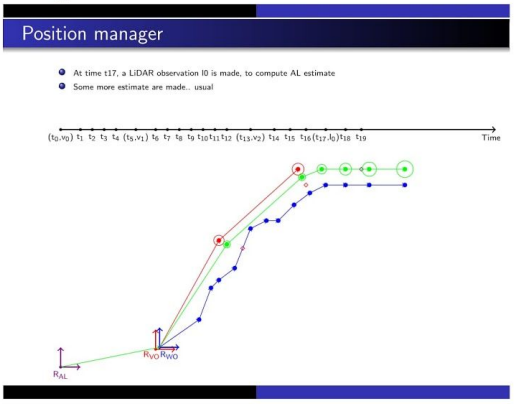
- At time t16, the VO transform is estimated for t13
- A new fused pose can be estimated from fusing the previous fused pose with the VO estimation
- As before, we propagate through KF prediction

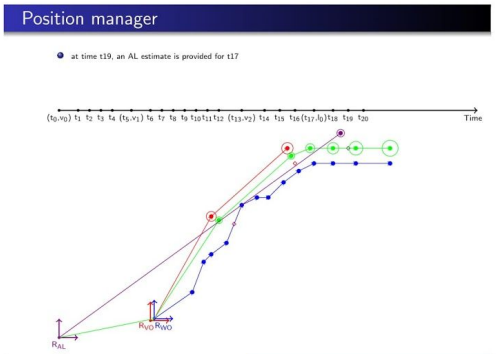


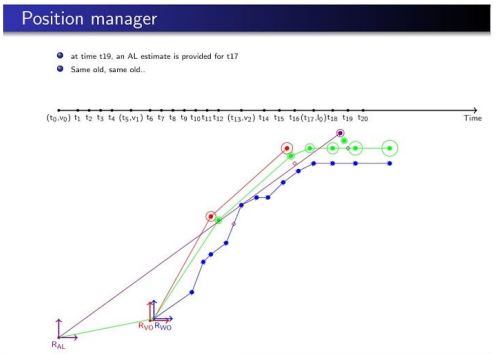

Position manager
At time t16, the VO transform is estimated for t13
A new fused pose can be estimated from fusing the previous fused pose with the VO estimation
Question: How do we keep consistent in time ?
Time

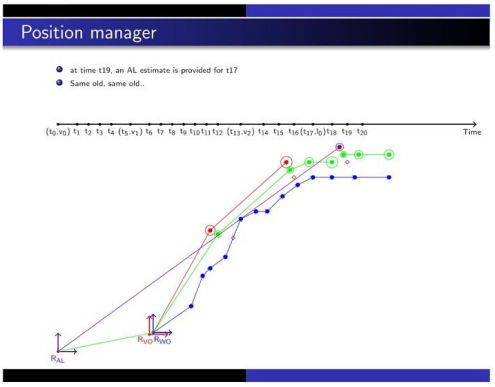

Position manager
At time t16, the VO transform is estimated for t13
A new fused pose can be estimated from fusing the previous fused pose with the VO estimation
One solution: Keep only poses associated with sensor observations
Time

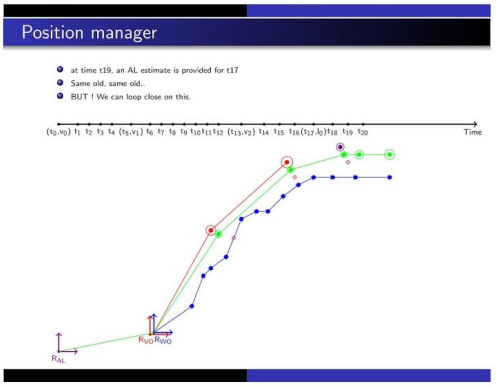

Position manager
At time t17, a LiDAR observation l0 is made, to compute AL estimate
Time


Position manager
At time t17, a LiDAR observation l0 is made, to compute AL estimate
Some more estimate are made.. usual
Time


Position manager
at time t19, an AL estimate is provided for t17
Time


Position manager
at time t19, an AL estimate is provided for t17
Same old, same old..
Time


Position manager
at time t19, an AL estimate is provided for t17
Same old, same old..
Time


Position manager
at time t19, an AL estimate is provided for t17
Same old, same old..
BUT ! We can loop close on this.
Time

The poses are managed within a single centralized component, the Position Manager (PoM), that takes as inputs all the pose and motion estimates produced by the localisation DFPCs, and that fuses these information resorting to two kinds of fusion DFNs:

- A Kalman Filter DFN is used to fuse the individual pose estimates as they are produced
- A pose graph-based non linear optimization DFN is used to update the memorized past poses, when a pose estimate allows such an optimization (namely, when a loop of transforms is identified in a graph of memorized poses).

The latter DFN is introduced here to propagate the information brought by the fusion of Map-based Localisation DFPCs to past poses. It is the same DFN that is used in the Lidar Pose-Graph Localisation DFPC.

7.4.4.2 Using Envire for managing poses

The Envire library provides already a set of utilities that can be very useful for the implementation of the DPM. This section briefly introduces some of them.

The Envire Graph can be populated with Frames connected by transformation with covariance corresponding to each of the different frames internal to the robot (e.g. sensor_i frame) as well as external to the robot (e.g. marker_i). These frames can be stored connected for instance to the previous position of the frame at any frequency. So far, the uses cases that have been covered with the Envire Graph only incorporate a frame for each position to be represented and the corresponding transformation has been updated. Envire Graph imposes currently unique naming of the frames. For covering the DPM requirements an approach would be to declare frames with an index associated to them or to modify the library so that frames with same name could be included in the same graph. Some utility functions could be implemented to access quickly all frames created after certain timestamp to update its value

One very useful utility of the Envire Graph that can also be used in this case is the subscription to events. Any object can be implemented and a subscriber to a certain type of events taking place on an Envire Graph of the same library. Events to which a class can be subscribed are for instance, the update of the position of an existing frame, the addition of a new frame to graph or the addition of an object of certain type to a particular frame of the graph or to any frame in general. This function can become very useful to update the poses of a high frequency pose estimation system with high covariance based on a lower frequency and more accurate pose estimation method.

Another advantage of using Envire is to reduce the use of additional dependencies in the project because Envire will be integrated as the visualization tool for the CDFF-Dev tools.

Indeed, this visualization tool can become very useful in the development of the DPM as well as for debugging,

7.4.4.3 Managing environment models

The environment models are produced by the SLAM and Environment Modeling DFPCs. There are no dependencies between the various models, and managing them in a unique centralized component would require high bandwidth exchanges with these DFPCs. Therefore, they are managed within the DFPCs where they are produced, and the DFPCs and other client processes (OG2-Ergo, other robots in multiple robot setup, and the operators) that require environment models have access to them through requests addressed to the proper DFPC.

Within each of these DFPCs, the environment models and memorized raw data are accessed through requests that specify the area or time interval corresponding to the desired information. For this purpose,

As it is done in most geographic information systems, the environment models are stored within a structure of tiles that paves the space along a regular Cartesian discretization. This simple structure indeed yields straightforward spatial hashing, and allows simple data management policies to load, save, allocate, export and transmit chunks of data. A tile is a container, it is not information per se. The tile structure is geo-referenced once and for all: it is their information content which is updated as information are gathered by the robot.

7.4.4.4 Using Envire for the Management of Environment Models

Envire provides serialization methods tested for maps. It is implemented to allow the storage of any kind of object in its frames. Though the correspondent serialization methods would have to be implemented.

Both approaches distributed as well as centralized could be covered using the Envire Graph. In the first case only one Envire Graph would have to be maintained and in the second one for each DFPC requiring serialization.

7.4.4.5 Envire usage summary

Envire provides at this stage well tested code that makes efficient management of the sensor data and associated data products. It provides serialization mechanisms that should ease the integration of the tool in different RCOS and an event system that would be very useful for the DPM for instance to trigger the update of old poses bases on the acquisition of a more accurate one.

Envire missed a couple of features that would be needed basically these are two queries: (1) Deliver all sensor data and data products of the requested type(s) associated to a time interval (2) Deliver all sensor data and data products of the requested type(s) associated to a given volume.

Finally, it is important to mention that Envire is at this stage not thread safe. At this point it has not been discussed whether the DPM will require concurrency.

7.5 Detailed Interfaces Between OG2/OG3/OG4

The following table presents the the detailed interfaces between OG2-OG3-OG4.

Interface	Call Direction	Synch/ Asynch	Params	Possible Values	Return value	Comments
setCDFFState	OG2 -> OG3	Synch	State	Initialize, idle, reset, stop	Success, Error, invalid States	
getCDFFState	OG2-> OG3	Synch	NULL	N/A	Runtime state or error	
getDFPCStatus	OG2-> OG3	Synch	Type	DEM or Pose	Runtime state or error	
getRoverMap	OG2 -> OG3	Synch	List of sensors, accuracy, update rate, resolution, area of coverage	sensor names, expected accuracy values, Hz, pixel to cm coverage, in sq. mts	DEM map or error state	Map produced with information gathered by sensors on the rover itself at the last sensing capture
getFusedRoverMap	OG2 -> OG3	Synch	List of sensors, accuracy, update rate, resolution, area of coverage	sensor names, expected accuracy values, Hz, pixel to cm coverage, in sq. mts	DEM map or error state	Map produced with information gathered by sensors on the rover itself at the last and previous sensing captures.

getFusedTotalMap	OG2 -> OG3	Synch	List of sensors, accuracy, update rate, resolution, area of coverage	sensor names, expected accuracy values, Hz, pixel to cm coverage, in sq. mts	DEM map or error state	Map produced with information from any sensing sources at any capturing time, e.g. rover, orbital, other mobile or static devices on the surface.
getLocalPose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	Produces the LocalPose Pose of the BodyFrame in the LocalTerrainFrame
getGlobalPose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	Produces the GlobalPose Pose of the BodyFrame in the GlobalTerrainFrame
getAbsolutePose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	Produces the AbsolutePose Pose of the BodyFrame in the AbsoluteFrame
getTargetRelativePose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	relative pose (3 axes position and 3 axes attitude) of the target Body Frame expressed in the chaser Body Frame, with associated uncertainties
getTargetRelativeVelocity	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	twist + uncertainty	relative speed (3 axes translation speeds and 3 axes rotation speeds) of the target Body Frame expressed in the chaser Body Frame, with associated uncertainties
getModelOfTarget	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	3D Model	This interface produces the 3D model of the target spacecraft.

initDFPC	Orch -> DFPC	Synch	DFPC ID	DFPC Name	Success, Error States	
stopDFPC	Orch -> DFPC	Synch	DFPC ID	DFPC Name	Success, Error States	
getDFPCStatus	Orch -> DFPC	Asynch	DFPC ID, Frequency, Callback function ptr	Name, Hz, Function ptr	N/A	
getDFPCPose	Orch -> DFPC	Asynch	DFPC ID, Frequency, Callback function ptr	Name, Hz, Function ptr	N/A	
getDFPCDEM	Orch -> DFPC	Asynch	DFPC ID, Frequency, Callback function ptr	Name, Hz, Function ptr	N/A	
initICU	OG3 -> OG4	Synch	NULL	N/A	Success, Error States	
setOperatingMode	OG3 -> OG4	Synch	OpModeID	ID Number	Success, Error, invalid States	
selectSensorConfiguration	OG3 -> OG4	Synch	SensorID, Configuration ID	ID number, ConfigID number	Success, Error, invalid States	
getOpModeSensorStatus	OG3 -> OG4	Synch	OpModeID	ID Number	Runtime or error states	
getStereoCamDepthMap	DFPC -> OG4	Synch	NULL	N/A	Depth map or error state	
getStereoCamDisparityMap	DFPC -> OG4	Synch	NULL	N/A	Disparity Map or error state	
getStereoCamPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getStereoCamImages	DFPC -> OG4	Synch	NULL	N/A	Images or error state	
getToFPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getIMUData	DFPC -> OG4	Synch	NULL	N/A	Linear acceleration & angular velocity or error state	
getLidarPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	

getLaserScan	DFPC -> OG4	Synch	NULL	N/A	Planar 2D PC or error state	
getRadarScan	DFPC -> OG4	Synch	NULL	N/A	2D or 3D PC or error state	
getHRCameraImage	DFPC -> OG4	Synch	NULL	N/A	Image or error state	
getTIRCameraImage	DFPC -> OG4	Synch	NULL	N/A	Image or error state	
getForceTorque	DFPC -> OG4	Synch	NULL	N/A	Wrench data or error state	
getStructuredLightPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getStarTrackerOrientation	DFPC -> OG4	Synch	NULL	N/A	Orientation data or error state	