Deliverable Reference	:	D5.1
Title	:	Orbital RI AND ASSOCIATED EGSE DETAILED DESIGN
Confidentiality Level	:	PU
Lead Partner	:	DLR
Abstract	:	This deliverable presents the detailed design of the orbital reference implementation and associated EGSE, addressing the implementation details and interfaces of data fusion nodes and data fusion processing compound of the CDFF framework.
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:	Nassir Oumer Jeremi Gancet Shashank Govindaraj Nassir Oumer Michal Smisek Shashank Govindaraj Jeremi Gancet Xavier Martinez Fabrice Souvannavong Vincent Bissonnette Arnaud Moura Mark Post Alessandro Bianco Romain Michalec Simon Lacroix Quentin Labourey Andrea De Maio Raul Dominguez	DLR SPACEAPPS SPACEAPPS DLR SPACEAPPS Magellium Strathclyde University CNRS/LAAS DFKI	30/03/2018 23/01/2018
PA Checked by:			

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
0.1.0	16/08/2017	Nassir Oumer InFuse team	All	Document contents template preparation
0.2.0	13/12/2017	Nassir Oumer	All sections	First Release
0.3.0	20/12/2017	Xavier Martinez	Chapter 4 & 5	DFN and DFPCs for mid-range tracking and detection
0.4.0	23/12/2017	Nassir Oumer Mark Post Michal Smisek Vincent Bissonnette Fabrice Souvannavong	Chapter 4 & 5	Adding DFN and DFPC descriptions
0.5.0	18/01/2018	Nassir Oumer	Chapter 4 & 5	Merging most contents of Section 5 to section 4
0.6.0	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 Nassir Oumer Jeremi Gancet	All Chapters	Final check of content and formatting the document
2.0	30/03/2018	Nassir Oumer, Shashank Govindaraj Jeremi Gancet All	All Chapters	Addressing RIDs emitted by the PSA.

Executive Summary

This document presents the detailed design of the InFuse CDFF. It consists of an implementation detail of CDFF components in orbital track. The document provides the detail design, including internal and external interfaces of Data Fusion Node (DFN), Data Fusion Processing Compound (DFPC), and also exposes interfaces to an external such as OG4 and OG2. Several use cases are identified and related detail design of DFPCs and DFNs are extensively described. Consequently, this document serves as a guideline for the software development of the orbital track associated to the CDFF framework.

Table of Contents

1 Introduction 10
1.1 Purpose 10
1.2 Document Structure 10
1.3 Applicable Documents 10
1.4 Reference Documents 11
1.5 Acronyms 11
2 Reference Scenarios Implementation 13
2.1 Introduction 13
2.2 Detection, Reconstruction and Tracking of a Target 17
2.2.1 RI-INFUSE: Detection, Reconstruction and Tracking 17
2.2.2 Use Case 1 : Far-range Target Detection and Tracking 18
2.2.2.1 Description 19
2.2.3 Use Case 2 : Mid-range 3D Model Detection and Tracking 19
2.2.3.1 Activity Diagram 20
2.2.4 Use Case 3: LIDAR-based Tracking of a Target 21
2.2.4.1 Activity Diagram 22
2.2.5 Use Case 4: Visual Tracking of a Target and Estimation of Robot Relative State 23
2.2.5.1 Description 24
2.2.5.2 Activity Diagram 25
2.2.6 Use Case 5: 3D Reconstruction 26
2.2.6.1 Description 26
2.2.6.2 Activity Diagram 27

2.2.7 Requirements 28
2.2.8 RI-SRC.SR-Detection, Reconstruction and Tracking 28
3 System Modeling 29
3.1 Satellite and Robotic System 29
3.2 RI-INFUSE 30
3.2.1 System Components 30
3.2.2 On-orbit Servicing Simulator 33
3.3 RI-SRC.SR 35
4 Detailed Architecture and Design 36
4.1 DFPC Architecture View 36
4.1.1 DFPC : Far-range Object Tracking 38
4.1.2 DFPC : Mid- and Close-range Target Detection 41
4.1.3 DFPC : Mid- and Close-range Target Tracking 44
4.1.4 DFPC: LIDAR-based Tracking of a Target 48
4.1.5 DFPC : Mid- and Close-range Visual Tracking of a Target 52
4.1.6 DFPC: 3D Reconstruction 57
4.1.7 DFPC: 3D Tracking 59
4.2 DFN 60
4.2.1 DFN Template 61
4.2.1.1 DFN Description 61
4.2.1.2 DFI: Template Implementation 62
4.2.1.3 DFN Description File 62
4.2.1.4 DFN Sequence Diagram 63
4.2.2 DFN Detailed Design 63

4.2.2.1 DFN: Image Geometric Processing 63
Image Geometric Processing 63
4.2.2.1.1 DFI: Image Undistortion 63
Image Undistortion 63
4.2.2.2 DFN: Edge Detection 64
4.2.2.2.1 DFI: Canny Edge Detector 64
4.2.2.3 DFN: Estimation Filter 65
4.2.2.3.1 DFI: Extended Kalman Filter 65
Extended Kalman Filter 65
4.2.2.4 DFN: FeatAndSigExtractor 65
4.2.2.4.1 DFI: ORB Feature Extractor 66
4.2.2.5 DFN: Feature Matching 66
4.2.2.5.1 DFI: FLANN Matcher 67
FLANN Matcher 67
4.2.2.6 DFN: Fundamental Matrix Calculation 67
4.2.2.6.1 DFI: Fundamental Matrix Calculator 68
Fundamental Matrix Calculator 68
4.2.2.7 DFN: Bundle Adjustment 68
4.2.2.7.1 DFI: Bundle Adjustment 68
Bundle Adjustment 68
4.2.2.8 DFN: 3D Point Computation 69
4.2.2.8.1 DFI: Linear Triangulation (DLT) 69
Epipolar geometry 69
4.2.2.9 DFN: 2D-3D Motion Estimation 70

4.2.2.9.1 DFI: PnP (Perspective from n-Points) 70
PnP 70
4.2.2.10 DFN: Point Cloud Construction 71
4.2.2.10.1 DFI: Point Cloud Builder 71
Point Cloud Builder 71
4.2.2.11 DFN: 3D Keypoint Descriptor Extraction 71
4.2.2.11.1 DFI: SHOT 3D Keyoint Extractor 72
SHOT 3D Keyoint Extractor 72
4.2.2.12 DFN: Correspondence Grouping 72
4.2.2.12.1 DFI: Hough Correspondence Grouping 73
Hough Correspondence Grouping 73
4.2.2.13 DFN: Target Pose Estimation 73
4.2.2.13.1 DFI: Target Pose Estimator 73
Target Pose Estimator 73
4.3 Data Types 74
5 Detailed Description of EGSE 75
5.1 Introduction 75
5.2 EGSE 75
5.2.1 Hardware and Software Interface 76
5.2.2 Functional Interface 77
5.2.3 Operational Interface 77
6 Conclusion 78
7 Appendix 79
7.1 Requirements 79

7.2 Technical Note on DFN and DFPC Specification **80**

7.2.1 Scope of the Note **80**

7.2.2 Definitions **80**

7.2.2.1 DFN **81**

7.2.2.2 DFPC **81**

7.2.3 DFN Template **82**

7.2.3.1 DFN Template Elements **82**

7.2.3.2 Towards a Typology/Taxonomy of DFNs **83**

7.2.4 DFPC Description Template **84**

List of of Figures

Figure 1: Earth background to a target image.
Figure 2: Earth and deep space background to the target
Figure 3: Illustration of a spacecraft (a servicer) rendezvous to a target from far-range to close-range and states that should be estimated.
Figure 4: Package diagram of the demonstration and validation scenario in orbital track.
Figure 5: Use Case 2 Mid-range 3D Model Detection and Tracking
Figure 6: Lidar-based Tracking of a Target Activity Diagram
Figure 7: Activity diagram of edge-based pose tracking
Figure 8: Activity Diagram of 3D Reconstruction and Tracking
Figure 9: On-orbit servicing system consists of servicer and targets as the main components.
Figure 10: Sensor system for an orbit-servicing.
Figure 11: Overview of the Software system of the orbital reference implementation.
Figure 12: The overview of the main on-orbit servicing ground simulation.
Figure 13: Deployment overview of the InFuse with the DLR OOS-sim. The Links and Nodes is the DLR robotics middleware used here for inter-process communication.
Figure 14: The DFN of the model-based visual tracking.
Figure 15: Sequence diagram of the model-based visual tracker.
Figure 16: The OOS-sim facility.
Figure 17: A simple schematic view of DFNs and DFPCs.
Figure 18: Example of a possible implementations of a DFN
Figure 19: LIDAR-PG-SLAM

1 Introduction

1.1 Purpose

The purpose of this document is to provide a detailed design for the implementation of orbital reference scenarios. The detailed design includes the definition and specification of EGSE as well as the software detailed design. The software design is based on the CDFF preliminary design [RD7] and the technical trade-off analysis [RD5]. The document addresses two related reference implementations (i.e. integration and validation tracks), the first one at the consortium level, RI-INFUSE, the second one at the SRC Space Robotics level, RI-SRC.SR. The objective of RI-INFUSE 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. The objective of RI-SRC.SR is to demonstrate CDFF is ready to be integrated with OG1, OG4 and OG6.

1.2 Document Structure

In brief, the document is structured as follows:

Section 1: This introductory material.

Section 2: The reference scenarios and validation of the implementation

Section 3: The system modeling of the test bed to emulate a chaser and target satellite of OOS operations.

Section 4: The detailed architecture and design - of the data fusion processing compound (DFPC) and associated DFN

Section 5: Detailed description of EGSE that will be used for testing and validating Orbital reference implementation of CDFF.

Section 6: Conclusion

Section 7: Reference

Section 8: Appendix

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 Description of Action document

RD2 D2.2: System Requirements and Operational Concept

RD3 D2.3: Functional and Physical Architecture Specification

RD4 D3.2: System Requirements and Scenario descriptions

RD5 D4.1 Technical Trade-off Analysis

RD6 Facilitators Interface Control Document

RD7 D4. 2 Preliminary Design Document

RD8 D5.2 Planetary RI and associated EGSE Detailed Design

1.5 Acronyms

DF: Data Fusion

CDFF: Common Data Fusion Framework

API: Application Program Interface

OOS: On-Orbit Servicing

RCOS: Robot Control Operating System

DFN: Data Fusion Node

DFPC: Data Fusion Process Chain/Compound

DFNCI: Data Fusion Node Common Interface

DPM: Data Product Manager

HDL: Hardware Description Language

HLS: High-Level Synthesis

MW: Middleware

LOS: Line of Sight

Fps: Frames per second

OOS-sim: On Orbit Servicing simulator

OG: Operational Grant

IMU: Inertial Measurement Unit

OT: Orbital Track

PT: Planetary track

OBC: On Board Computer

DEM: Digital Elevation Model

FPGA: Field Programmable Gate Array

2 Reference Scenarios Implementation

Here we describe the detailed implementation, and associated demonstration and validation scenarios presented in [RID4 & RID5].

2.1 Introduction

A rendezvous mission for an on-orbit servicing has been defined irrespective of an orbit (LEO, MEO,GEO) in [RID4], hence the choice of an orbit is left to the end user of the CDFF. The performance of the reference implementation is influenced mainly by the intensity and direction of the sun with the respect to visual and TOF sensors. The reference implementation (RI) takes account of various conditions of space lighting by employing appropriate rendezvous sensors such as a LIDAR and a camera system. Moreover, the RI should address the space environment related to the target background which poses challenges during approach or proximity operation. The target background may be the following:

- *Deep space background (Fig.1),* where the sensor points away from the Earth and towards space objects. In this case, other celestial bodies such as stars may exist as a background (not shown here)
- *Earth and deep space (Fig.2), t*he sensor line of sight slightly drifted from nadir direction, enabling the Earth and deep space in sensors field of view. The sensor may also point completely toward the nadir direction

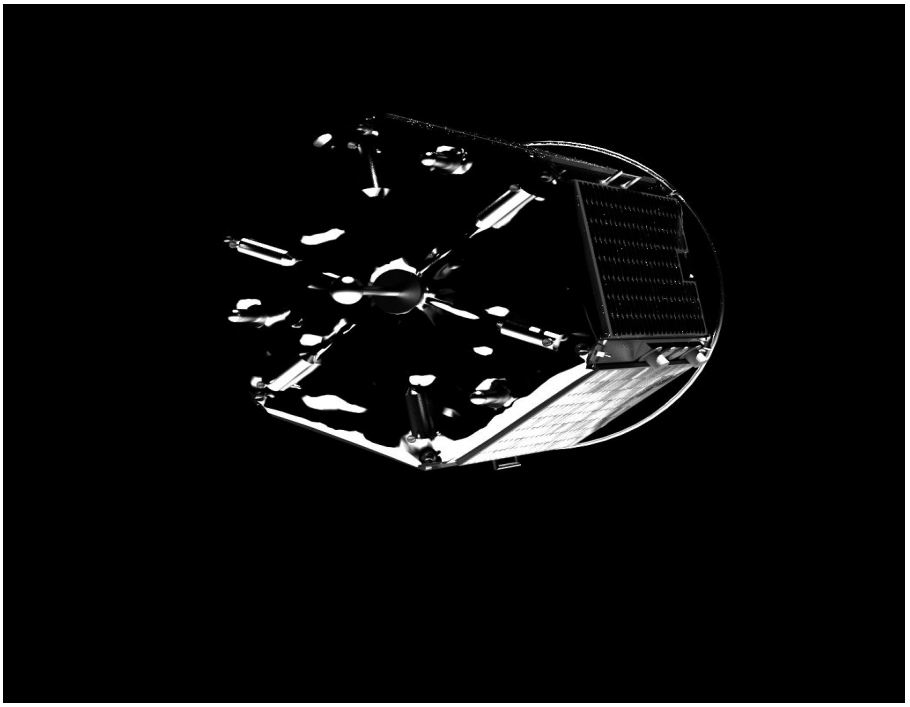

Figure 1: Deep Space background to a target

Consequently, the mission definition, preparation and execution should consider such sensor and target configuration with respect to space or Earth in order that the RI captures typical scenarios of a space mission.

Furthermore, it is possible to assume a certain orbit in space to support navigation to the target satellite so that the dynamics of the client spacecraft can be used to predict and filter the state estimate provided by CDFF. This choice of orbit will be left for the end user of the CDFF.

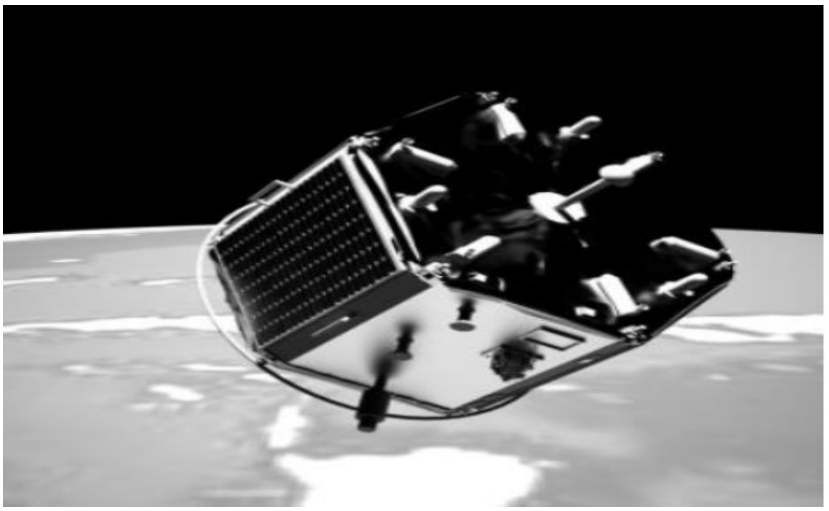

Figure 2: Earth and deep space background to the target

The RI of the orbital track consists in the relative localization, which include approaching a target spacecraft from far-range to docking/berthing of to/on the desired region of interest. In practice, the rendezvous of a target space object is bounded by range and time as illustrated in the following Figure (the numerical values are approximate and varies according to the requirements of a certain mission). The range of operations pose specific requirement in specification, design and implementation of sensor and software components. Moreover, the farther the target from the servicer is, the lesser the accuracy requirement.

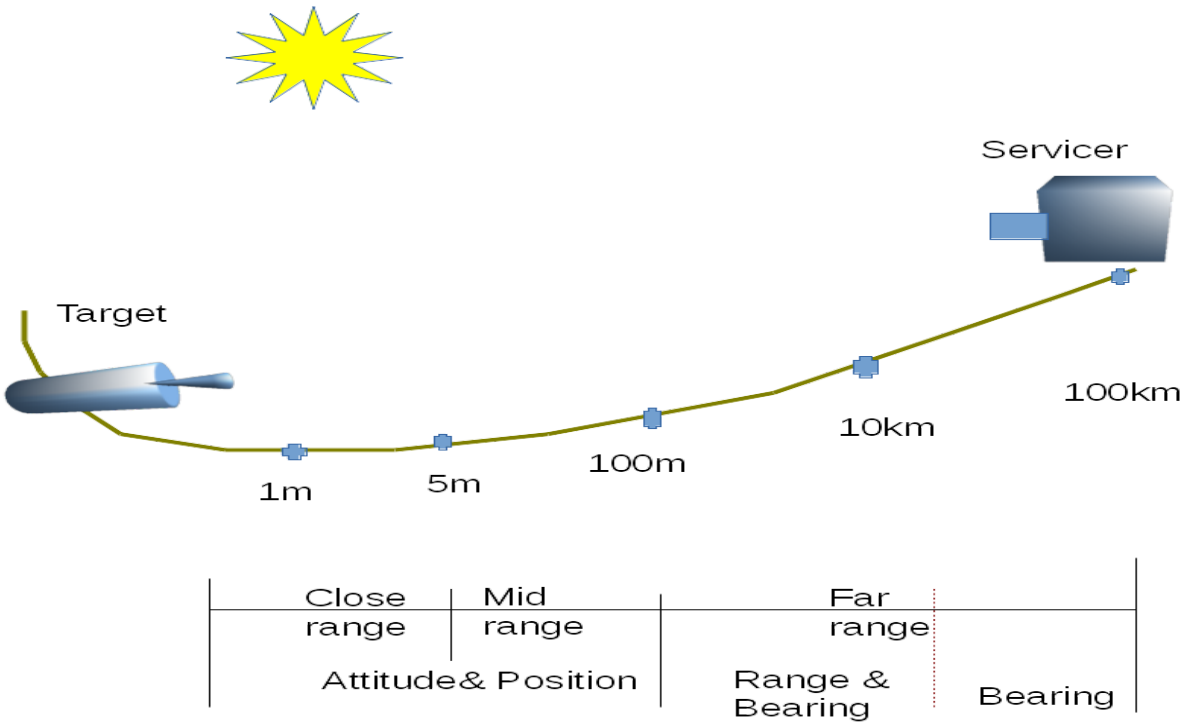

Figure 3: Illustration of a spacecraft (a servicer) rendezvous to a target from far-range to close-range and states that should be estimated.

The demonstration and validation of orbital scenario can be performed by

- Software simulation, for example rendering the target at various ranges
- Hardware and software simulation

The software simulation is relatively easy and flexible validation approach in order to reproduce the on-orbit scenarios, demonstrate and validate the CDFF framework as well as DFPCs. However, such validation approach is not sufficient to reproduce the real space environment and optical characteristics of the target spacecraft. Thus, the DFPCs demonstrated with a software simulation environment could not reliably capture the real world scenarios that could encounter in space missions.

More realistic validation approach is to simulate the space environment and the optical characteristics of the target satellite with a representative hardware, such as a sun simulator with a high power floodlight, full or scaled mock-up of a target satellite and deep space or Earth's albedo background. This is what we call an Electrical Ground Support Equipment (EGSE). However, the hardware simulation limits the range of operation. This range limitation can be partially overcome by using

- scaled mock-up of a satellite

- representative subranges for mid-range validation.

We foresee the latter approach to validate the DFPCs in the mid-range and close-range. Table 1 shows the rendezvous ranges often used for on-orbit servicing and formation flying mission, and a demonstration and validation distances in InFuse.

Range	Approximate distance	InFuse demonstration and validation distance	Observables
Far-range	10's km to 100's m	N/A	Bearing and range
Mid-range	100's m to 5 m	17 m to 2 m	Position and attitude
Close-range	5m to 1m	2 m to 0.5 m	Position and attitude

Table 1: Rendezvous and demonstration / validation ranges, and respective observables

The reference implementation will be demonstrated and validated with the data recorded with OG6 facilities listed in Table 2.

OG6 Facility	Validation range	Remark
DLR OOS-sim	Close-range	
GMV facility	Mid-range	recorded data and collaboration with OG4 sensors

Table 2: Demonstration and validation range and associated EGSE

In order to validate in the mid-range, the sensor data recorded at OG6-GMV facility should consist in at least

- sequence of time stamped point clouds from LIDAR and corresponding synchronized ground truth pose trajectory
- sequence of time stamped stereo images and corresponding synchronized ground truth pose

The ground truth for validation of algorithms in mid-range can be generated through calibrated senor, robot and target as follows:

- transformation of target frame to TCP frame of a robot carrying the target satellite

- transformation of Tcp frame of target robot to TCP frame of servicer robot
- transformation of TCP frame of a servicer robot to the sensor (camera, LIDAR) frame

Moreover, a wavefront or a CAD model of the target satellite, which will be post-processed to suite for a certain data fusion algorithm is required. Inter-sensor transformation is also required, for fusion of camera data with LIDAR.

2.2 Detection, Reconstruction and Tracking of a Target

An on-orbit servicing involves rendezvous operations from the location where the servicer spacecraft is put in-orbit to the resident space object (target spacecraft). The navigation function requires various maneuvers during approach. The fundamental operations that enable to achieve the spacecraft navigation to a target include mainly a *target detection* and *tracking.* In order to precisely track the target at a given range of distance, an accurate geometric model of the spacecraft is essential. In case of inaccurate geometry, it is necessary to reconstruct the satellite/spacecraft with an aid of navigation sensors. Particularly, at a very close-range where higher accuracy is required, the *reconstruction* of the desired region of interest (e.g grasping region) on the target spacecraft enhances the performance for an on-orbit servicing during visual servoing.

In this Section, we describe each localization function (DFPC), and provide a general overview of the DFPC components (DFN). Each scenario is implemented in one or several use cases. The Use cases are distinguished by the use of different types of sensors or by different main DFPC structures. Each DFPC structure can in turn be instantiated in various "flavors" where different combinations of functions with the same type of inputs and outputs can be used. These variations on a DFPC are described in section 4.1.

The following section will address reference implementations at two levels:

- *RI-INFUSE*: the consortium level, in which we demonstrate and evaluate the full capabilities of the CDFF.
- *RI-SRC.SR:* the SRC Space Robotics level, that demonstrates the CDFF is ready to be integrated with OG1, OG2, OG4 and OG6.

2.2.1 RI-INFUSE: Detection, Reconstruction and Tracking

The reference implementation in InFuse will be carried out mainly with the DLR OOS-sim. The objective is to carry out an offline validation of the reference implementation. Furthermore, an online demonstration can be conducted with selected DFPC. The offline validation consists in primarily data recording and data exploitation. The evaluation metric relies on an accurately calibrated test facility used to carry out the validation. The ground truth obtained from a robot joint encoders and a hand-eye calibration will be used to assess the performance of reference implementation.

The package diagram in Fig.5 shows an overview of foreseen software and hardware elements in order to carry out the reference implementation.

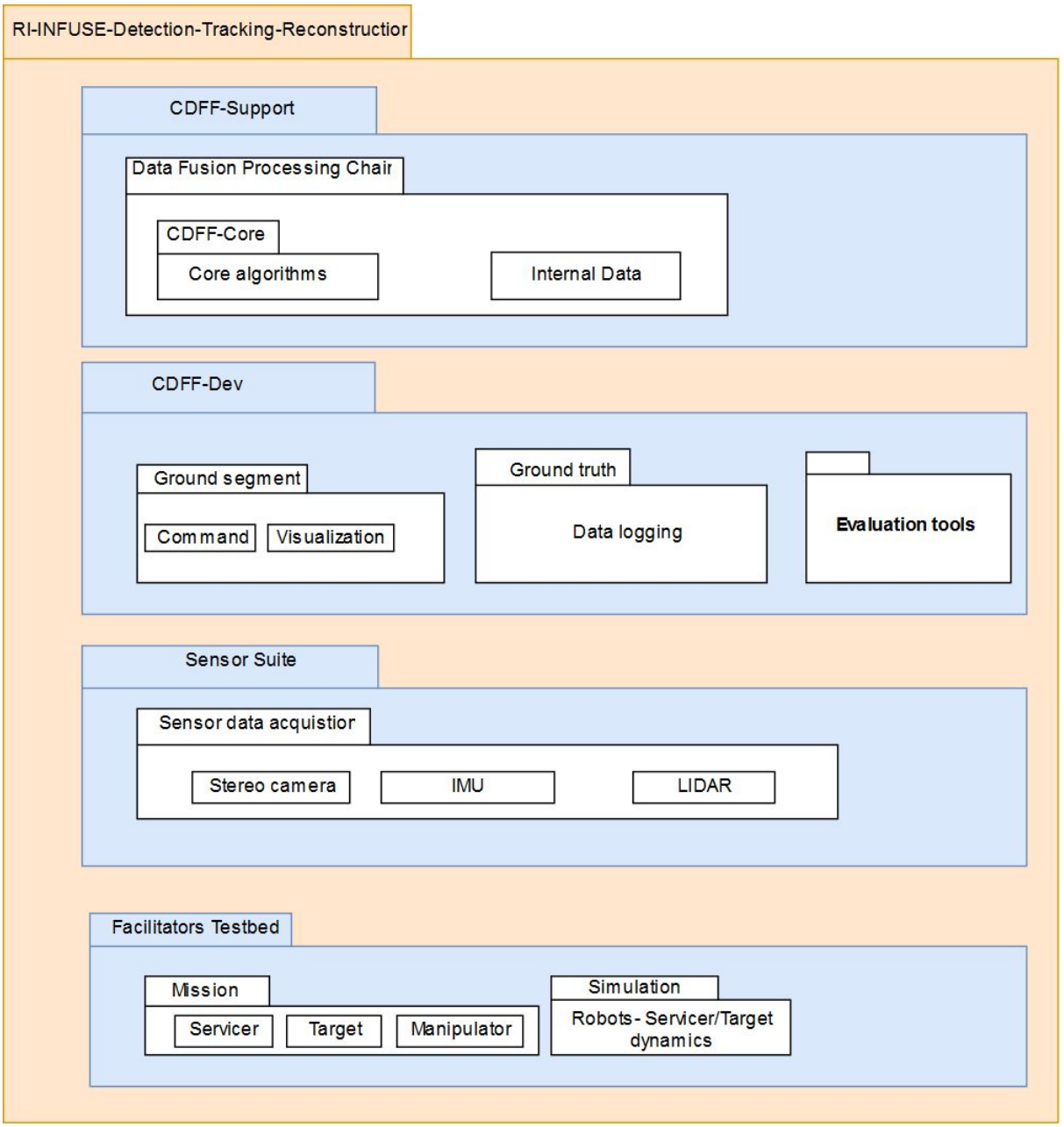

Figure 4: Package diagram of the demonstration and validation scenario in orbital track.

2.2.2 Use Case 1 : Far-range Target Detection and Tracking

Regarding the validation and demonstration approach for this use case, current discussions between InFuse and the PSA have brought the conclusion that, on the one hand, a full physical validation of this scenario is not currently possible with our internal hardware setups, and on the other hand, simulation-based validation would not be representative enough. Therefore, this use case will still be described, but its implementation will be put on

standby to increase the emphasis on mid-range operational scenarios which can actually be performed on the foreseen physical test beds.

Since, at far range, it is only realistic to accurately estimate the target bearing, we focus this use case on a simple 2D camera. The target, which we assume can possibly have its own translational and rotational motion, is first detected in the image by dense matching, then this bearing measurement is fed into a classic filtering function which uses a chaser and target motion model to ensure a continuous and robust tracking. Additionally, the tracking filter would benefit from having access to inertial measurements of the chaser, 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.

2.2.2.1 Description

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,
- Chaser navigation and locomotion control loops,
- Chaser (and target) position and orientation measurements for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Localisation accuracy evaluation.

2.2.2.2 Algorithm Performance

Since the long range tracking DFPC only operates in bearing to guide the chaser 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.

2.2.3 Use Case 2 : Mid-range 3D Model 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 combination of a 2D camera and a radar or lidar sensor 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

requires an initialisation step. In order to reduce operator intervention, we propose to perform initialisation with a target detection function. 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.

In the context of this implementation, with the considered EGSE, using a radar sensor may not be feasible, however radar measurements could easily be simulated from lidar measurements of ground truth data.

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 cameras, lidar/radar and inertial measurements,
- Chaser and target mockups navigation and control loops,
- Chaser and target pose measurements for ground truth determination,
- User interfaces for target selection, live monitoring,
- Data logging and replay functions,
- Relative localisation accuracy evaluation.

2.2.3.1 Activity Diagram

This use case presented here is described in the following activity diagram, from the point of view of the user. The diagram helps in highlighting the various agents involved in the use case, and the sequence of actions necessary to implement the scenario.

1. The user initialises the OOS-Sim and chooses the required trajectory from the provided set,
2. The user loads the model of the target, the sensor calibration files,
3. Start the rendezvous/tracking process to execute the pre-planned trajectory:
    a. Camera images, LIDAR/Radar and IMU are acquired at a predefined rate,
    b. The detection DFPC is run on the current RGB-D data,
    c. If detection was successful, the visual tracking DFPC is initialised, and starts to use the image stream as input to estimate a relative pose between the servicer and client cartesian frames,
    d. The user can monitor the status and the execution, visualise data, and receive estimated pose,
    e. Execution is stopped as soon as the client has reached its target pose, or if the trajectory has reached its final point.

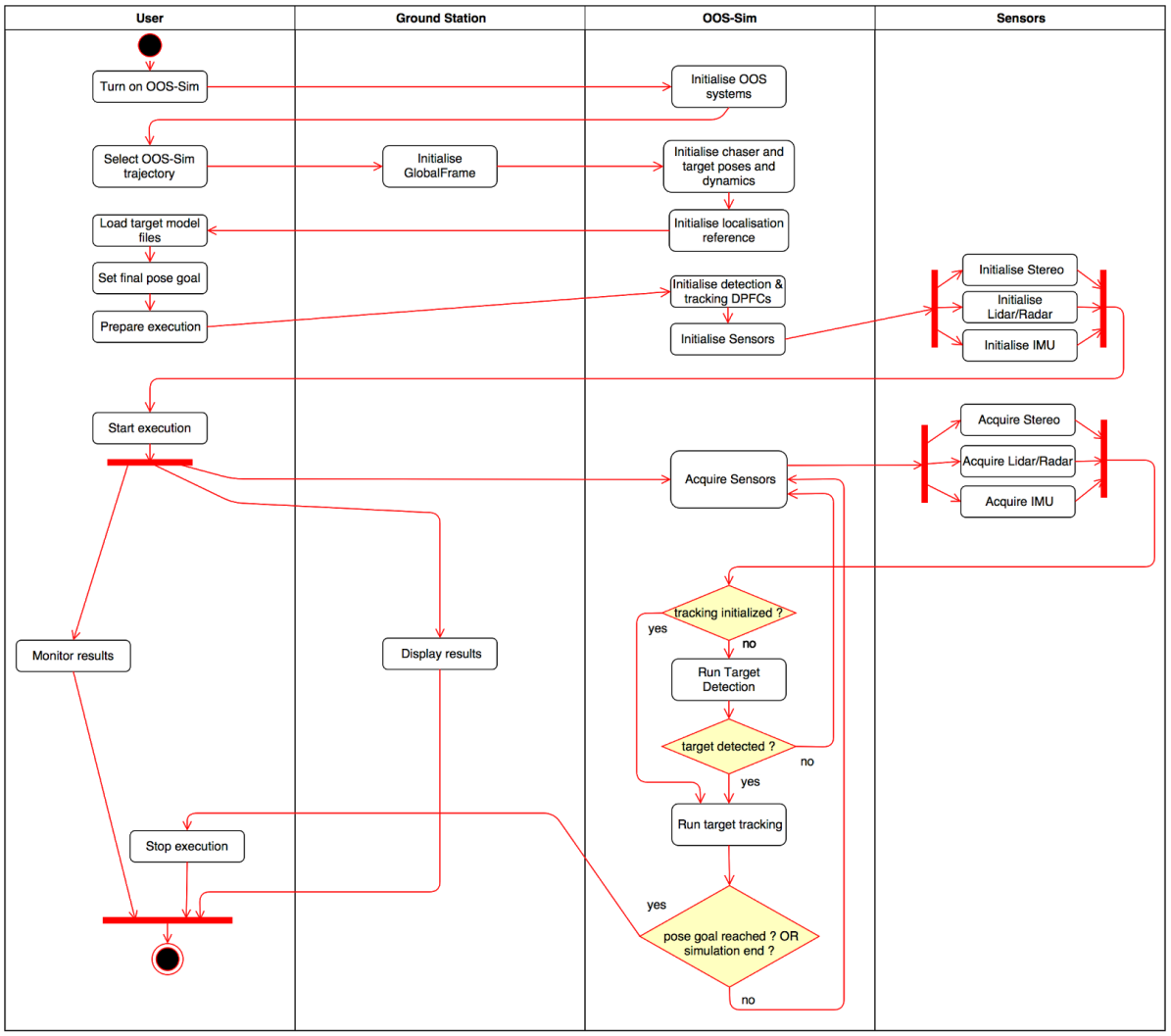

Figure 5: Use Case 2 Mid-range 3D Model Detection and Tracking

2.2.3.2 Algorithm Performance

The detection subfunction of 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.

Concerning the tracking subfunction, 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

[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.

2.2.4 Use Case 3: Mid to Close-range LIDAR-based Tracking of a Target

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,

- Chaser navigation and locomotion control loops.This refers to the simulated GNC (guidance, navigation and control) loops found on the chaser satellite, which control its dynamic behaviour in rendezvous maneuvers. In this case, these functions will be covered by the OOS-Sim equipment.
- Chaser and target pose 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.

2.2.4.1 Activity Diagram

This use case presented here is described in the following activity diagram, from the point of view of the user. The diagram helps in highlighting the various agents involved in the use case, and the sequence of actions necessary to implement the scenario.

1. The user initialises the OOS-Sim and chooses the required trajectory from the provided set,
2. The user loads the model of the target, the sensor calibration files,
3. Starts the rendezvous/tracking process to execute the pre-planned trajectory:
    a. LiDAR and IMU are acquired at a predefined rate,
    b. The point cloud tracking DFPC is initialised, and starts to uses the LiDAR stream as input to estimate a relative pose between the servicer and client cartesian frames,
    c. The user can monitor the status and the execution, visualise data, and receive estimated pose,
    d. Execution is stopped as soon as the client has reached its target pose, or if the trajectory has reached its final point.

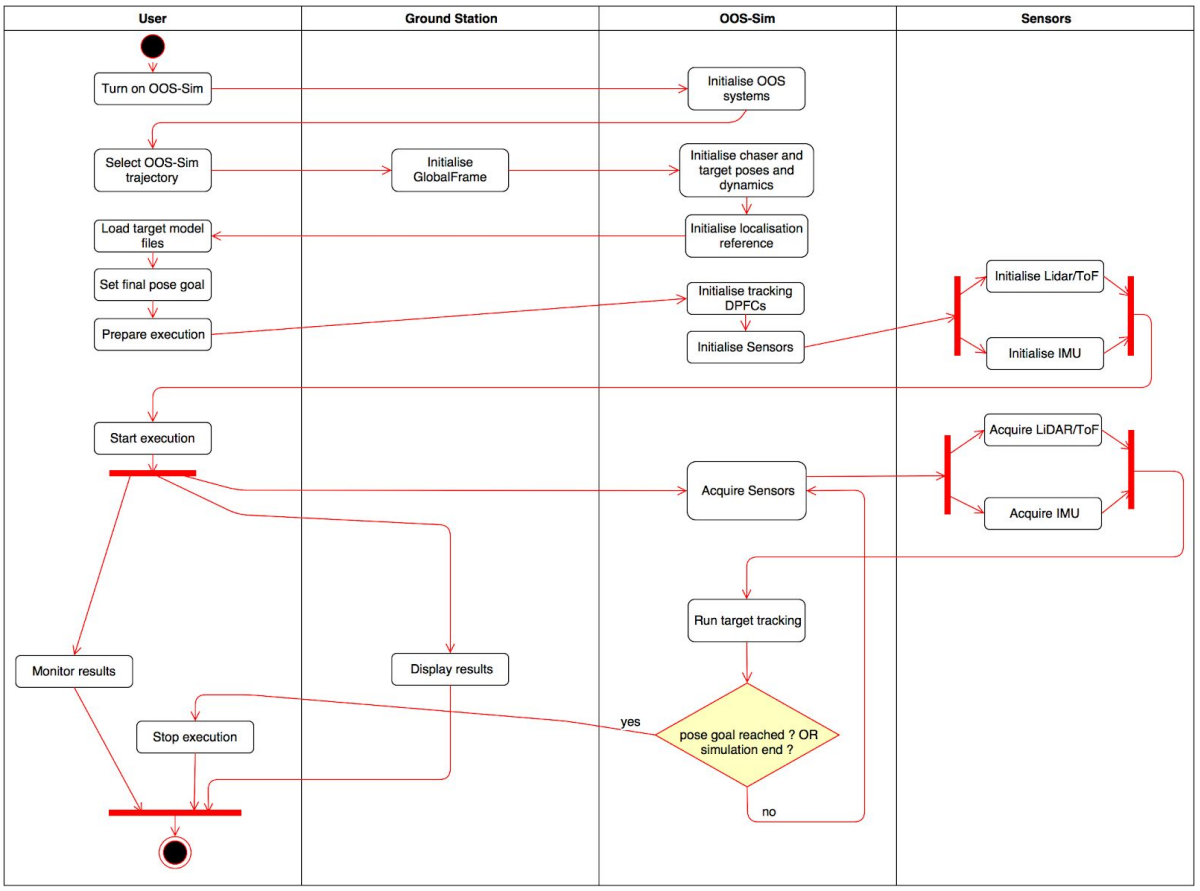

Figure 6: Lidar-based Tracking of a Target Activity Diagram

2.2.4.2 Algorithm Performance

Since this is a similar, yet simplified implementation of the Mid- and Close- Range, 3D reconstruction and object detection DFPC (Use Case 5) 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°.

2.2.4.3 Use of Point Cloud Data from Other Sources

Although LIDAR provides a means by which to obtain point clouds directly, other methods are possible - specifically the use of stereo vision and visual reconstruction (see section 2.2.6). InFuse is a modular system, and therefore allows point cloud generation from different sources to be used in the same algorithms. Therefore, it is possible for LIDAR point clouds to be used for 3D model-based target tracking as described in 2.2.6 by modification of the DFPCs used.

2.2.5 Use Case 4: Mid- and Close-range Visual Tracking of a Target and Estimation of Robot Relative State

In this use case, we focus on a camera-based pose estimation of a target satellite provided an ordered image sequence in respective timestamp. We assume a non-cooperative target, where neither vision-aiding markers nor GN&C on-board sensors on the target spacecraft are accessible for exploitation. Thus, the pose tracking relies on calibrated camera images and a geometric model of the target. It is assumed that the geometric model of the target can be obtained either from a spacecraft manufacturer's CAD model or reconstruction and modeling techniques.

The pose estimation provides the 3D position and orientation between two cartesian coordinate frames located on the servicer frame or TCP of a robotic manipulator, and on the target or a chosen grasping point on the target spacecraft. In order to perform the pose tracking, we employ image sequences, resulting from the relative motion of the servicer and client satellites using two cameras in stereo configuration. The algorithm will rather exploit each camera image independently and fuse the data to compute the pose. This two monocamera fusion has an advantage particularly in space application; in case one camera fails due to unexpected radiation, the tracking can proceed with the remaining monocular camera. Notice that two camera configuration may be used in case a higher accuracy is required at close range, otherwise a monocular camera can be used to reduce computational burden. The monocular camera with a priori knowledge of the geometric model of the target can be used to estimate the absolute position and orientation.

The visual tracking in 6 DOF relies on a local optimization, hence it requires an external initialization as well as re-initialization in case of a loss of tracking. The external initialization could be achieved by a global 3D detection method, e.g from a DFPC described in Use case 2. Furthermore, the visual tracking and detection DFPCs must have a synchronized sensor data in order to perform detection-tracking procedure robustly. In fact, there may exist certain delays because of the intensive computation of the global detector compared to the local tracker which is much faster. If the detector is based on sensor data other than the tracker camera, a relative pose of the sensors must be pre-determined through calibration procedure. The workflow of the visual tracker and the detector is illustrated in Fig.8. Notice that, the detector is inactive during tracking and is activated on demand such as when re-initialization is required. This sleep-mode is used to efficiently utilize resources (memory, processor and power).

Figure 8: The workflow of the integrated visual tracker and detector DFPCs.

We remark that the same visual tracking method is used both for mid- and close-range, with an appropriately selected camera and lense system according to range.

2.2.5.1 Description

Hereafter we describe the tracking procedure and processing flow.

Preparation (offline):

- 3D model pre-processing of the target geometry
- Calibration of stereo cameras and camera - TOF sensors (if any)
- A robot TCP- sensor calibration, aka hand-eye calibration

The tracking procedure follows

1. The user determines mode of tracking (mid-range or close-range), based on the appropriate distance to the target and respective camera-lense specification. If the mode of the operation is close-range, the user should define the desired grasping point of the client (goal point).
2. The user loads the model of the target, the calibration file and a pre-planned or reference trajectory.
3. Start the rendezvous/tracking process to execute the the pre-planned trajectory:
    a. Camera image is acquired at a predefined rate
    b. The visual tracker waits until the initialization takes place by the detection DFPC
    c. The visual tracking DFPC uses the image stream as input to estimate a relative pose between the servicer and client cartesian frames
    d. The user can monitor the status and the execution, and receive estimated pose
    e. The visual tracking is stopped and the process finishes as soon as the client is out of field of view of the servicer or at a predetermined distance to the target.

2.2.5.2 Activity Diagram

Here we describe the functional aspects of the visual tracking. It is basically a model-based edge tracking, following the classical contour matching method yet a state of the art technique. The activity diagram in Fig.9 shows a stereo camera based pose tracking procedure and interface to external DFPC.

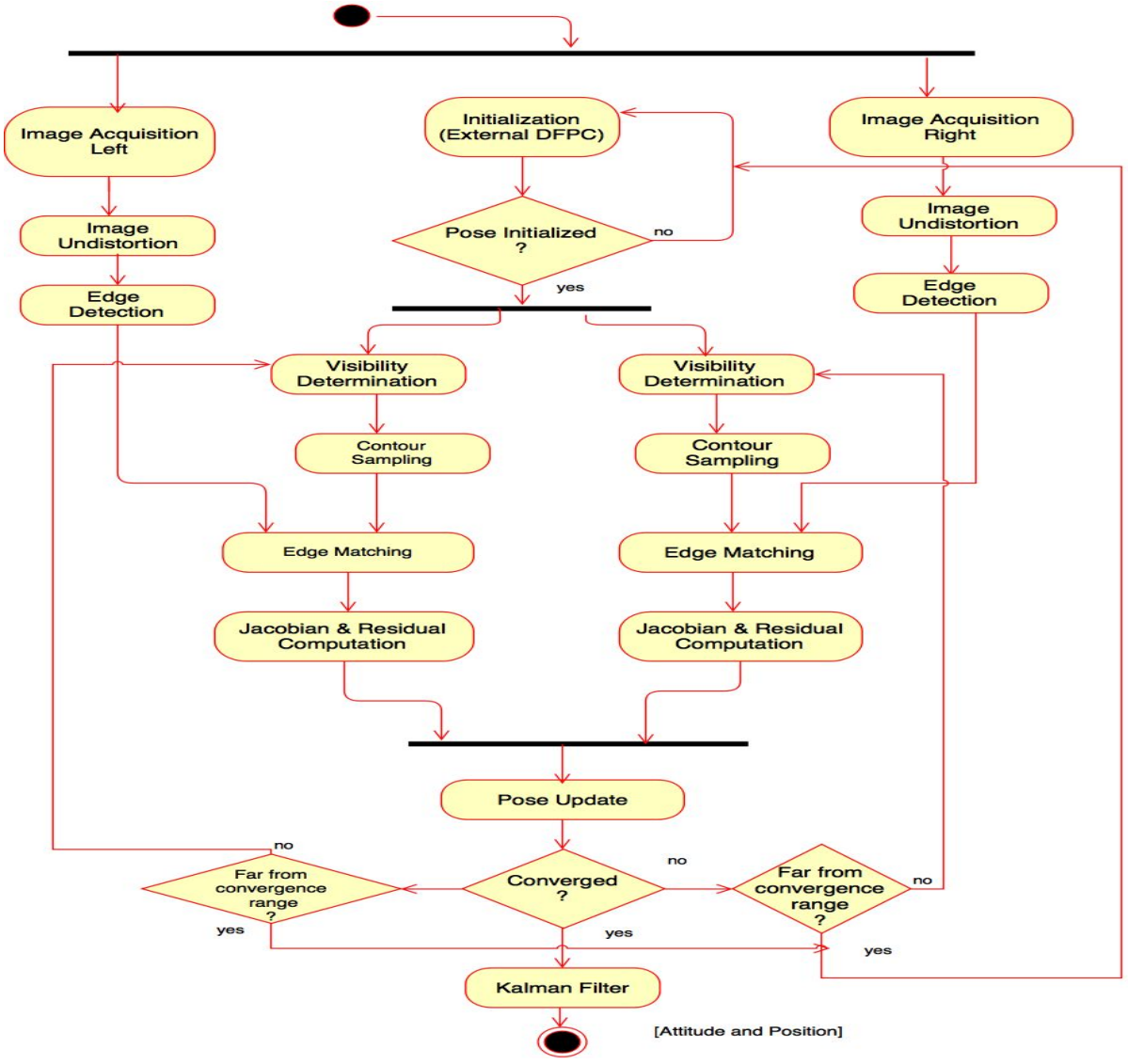

Figure 7: Activity diagram of edge-based pose tracking

2.2.5.3 Algorithm Performance

The performance of the algorithm is measured according to the accuracy with respect to range and robustness to space lighting. With regard to lighting, two boundary conditions are considered where the lighting is very poor hence the target is under illuminated, and the space lighting is highly directional to a reflective surface, leading to over illuminated target. In this lighting condition, the camera-based tracking algorithm is expected to provide inaccurate and unreliable pose estimate. The bottom line here is, that a vision-based

algorithm could be reliably used only in appropriate lighting condition with distinctive visible image features on the target. However, with support of target dynamics identification, a long-term prediction can be performed to improve the visual tracking performance in case of worst lighting condition. Identification of a target dynamics is currently not in the scope of InFuse.

Moreover, the performance of a visual tracking depends on a structure of the target, hence it may not be feasible to specify its performance boundaries without knowing the target. An accuracy of 30 mm at close- range and 5% error at mid-range is achieved for a typical satellite, with solar panel, adapter rings and launcher bracket interface

2.2.6 Use Case 5: Mid- and Close- Range, 3D Reconstruction and object detection

The localization approaches described above assume that the object geometric model exists as in the form of CAD model or by 3d reconstruction. Below, we describe the latter as a use case of the orbital implementation. The 3D Reconstruction and tracking DFPC family is useable within close range (0-2m) and mid range (2-17m) subject to appropriate sensor capabilities (sufficient separation of stereo cameras and illumination range on LIDAR/ToF devices). This use case is split into two DFPCs due to complexity of implementation. The 3D reconstruction DFPC performs environment reconstruction from camera images - in most cases stereo images as per the InFuse sensor suite, but with some degradation of performance monocular images could be used - and also active devices such as LIDAR and Time-of-Flight (ToF) cameras that produce point clouds directly. Different flavors of DFPC are designed to perform reconstruction from these different sources using common DFNs. The 3D tracking DFPC operates on the scene point clouds that are produced from the reconstruction process to identify instances of a pre-defined model within the scene.

2.2.6.1 Description

To allow a tracker spacecraft to identify and estimate the movement of a target spacecraft, four options are possible according to the sensors available: 3D reconstruction from a 3D Lidar or ToF camera, 3D Reconstruction from a stereo camera, 3D Reconstruction from a mono-camera, 3D reconstruction from mono camera and 3D Lidar and ToF camera. In the first case we obtain a point cloud directly from the sensors, in the second and third case, the point cloud is computed by detection of 2d features, matching and 3d triangulation of the correspondences. By projecting the keypoints into three dimensions, we build up a point cloud of the target, which can then be matched in shape to a point cloud model, and the pose of the model accurately obtained by three-dimensional keypoint correspondences.

The operation of 3D reconstruction proceeds as follows:

1. Parameters must be set for the sensors used, resolution, and sensor type (mono camera, stereo camera or TOF/LIDAR point cloud)
2. For the case of identifying a target shape, a point cloud model must be pre-stored or loaded into the system with known resolution and parameters

3. The sensor calibration must be available
    a. When using cameras, the camera parameters (frustum parameters, focal length etc.) must be known.
    b. When using a stereo camera, the camera baseline must be know in addition to the above parameters.
    c. For using ToF cameras or LIDAR, the scaling and range of data must be known
4. Images are read in from the 2D visual camera(s) or point clouds from ToF cameras or LIDAR.
5. *If the mono camera is used:*
    a. features are extracted from the images;
    b. Features from an image are matched with a suitable past image;
    c. The correspondences are used for the computation of a camera matrix and estimation of the camera transform between the two images;
    d. A 3d triangulation step allow to project the features correspondence in 3d space, and obtain a point cloud.
    e. The sequence of pose estimates from one image to a past image allows the computation of the current pose of the camera with respect to the initial pose.
6. *If the stereo camera is used:*
    a. A disparity map is computed and it is used for the construction of a 3d point cloud;
    b. Features are extracted from both left and right images;
    c. Features from the left image are matched with a suitable past left image;
    d. The correspondences are used for the computation of a camera matrix and estimation of the camera transform between the two images;
    e. The sequence of pose estimates from one image to a past image allows the computation of the current pose of the camera with respect to the initial pose.
7. *If a ToF Camera or 3D Lidar is used:*
    a. A point cloud is already available directly from the sensor;
    b. 3d features are extracted from the point clouds;
    c. Features from the points cloud are matched with a suitable past point cloud;
    d. The correspondences are used to compute a transform between the two point cloud;
    e. The sequence of transform estimates from one point cloud to a past point cloud allows the computation of the current pose of the sensor with respect to the initial pose.
8. If *we use mono camera and a ToF Camera or 3D Lidar* then*:*
    a. A point cloud is already available directly from the sensor;
    b. Features from an image are matched with a suitable past image;
    c. The correspondences are used for the computation of a camera matrix and estimation of the camera transform between the two images;
    d. The sequence of pose estimates from one image to a past image allows the computation of the current pose of the camera with respect to the initial pose.

9. The point cloud must be scaled, potentially filtered for outliers, and the resolution established.
10. The point cloud is merged with the reconstructed 3d environment in the position defined by the estimated sensor pose;
11. Features are extracted from the point cloud and the object model;
12. Features are matched and a transform from model to scene is estimated.

To perform tracking, the following process is used, and will be done in a separate DFPC due to the clear separation of processes once a point cloud is produced.

The operation of model-based tracking proceeds as follows:

13. The point cloud is culled if there is a significant resolution difference between the resolution of model and scene clouds
14. The scene is matched with the model using 3D descriptors to determine instances of the model within the scene.
15. The process is repeated with new sensor data. Successive matches will indicate the motion of the model within the scene

2.2.6.2 Activity Diagram

In Figure 8 we show a diagram of the complete 3D reconstruction and tracking process. On the left side of the picture, we are using a mono camera and a LIDAR sensor for 3D reconstruction. On the right side of the picture, an instance of a model within the point cloud can be tracked by identifying 3D correspondences. Again, the 3D reconstruction process (up until “Solve PnP for Poses” on left) is done in a separate DFPC from the 3D tracking process (following Scene Point Cloud” on right).

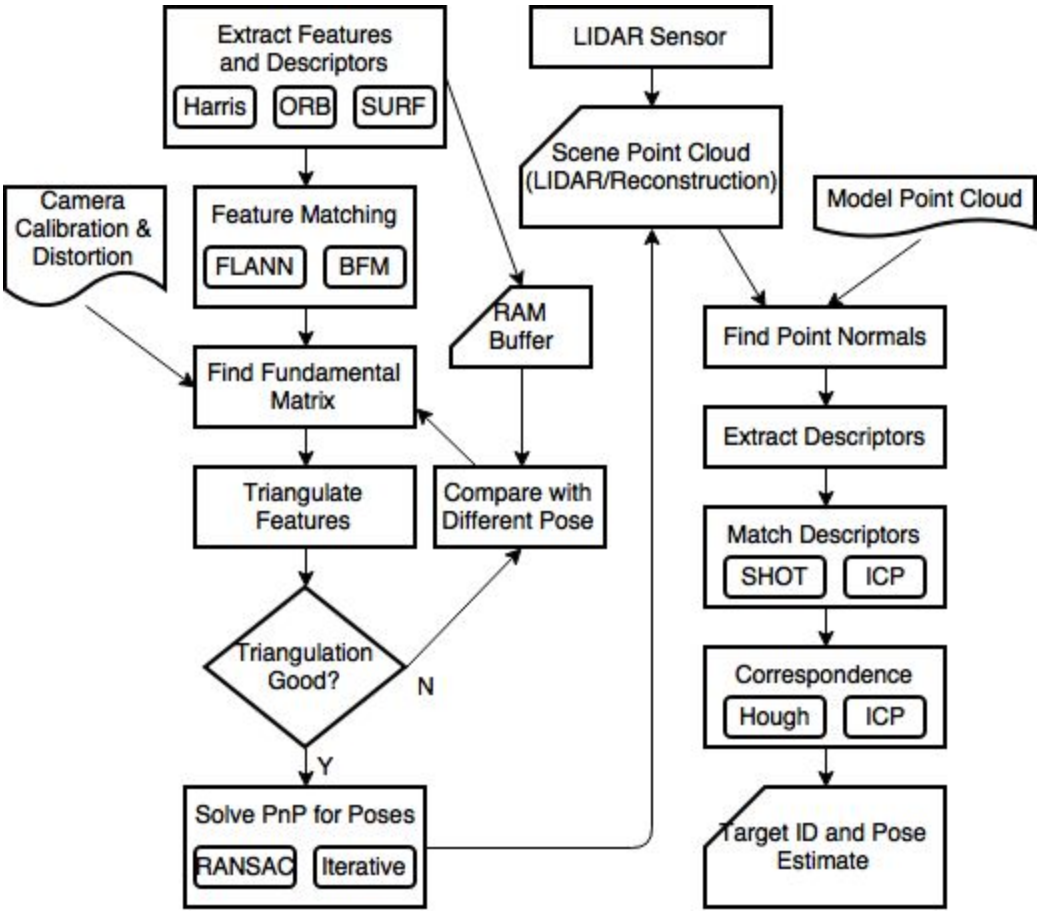

Figure 8: Activity Diagram of 3D Reconstruction and Tracking

2.2.6.3 Algorithm Details

Due to the complexity and the number of algorithms used in the 3D Reconstruction and 3D tracking DFPCs, a more detailed description of how the algorithms work is provided as follows.

2.2.6.3.1 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.

2.2.6.3.2 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

$$f_n(p)=\sum_{1\leq i\leq n} 2^{i-1}\tau(p;a_i,b_i). \quad (3)$$

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:

$$F=R_f\begin{pmatrix}a_1 \cdots a_n\\ b_1 \cdots b_n\end{pmatrix}. \quad (4)$$

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.

2.2.6.3.3 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.

2.2.6.3.4 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.

2.2.6.3.5 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 ˆx and ˆx' in the camera plane as:

$$\hat{a}_i'^{T}\mathbf{E}\hat{a}_i=0,\ i=1,\ldots,n. \quad (8)$$

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

2.2.6.3.6 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

$$\mathbf{W}=\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (9)$$

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.

2.2.6.3.7 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

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

(11)

and

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

(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 $x_2>0$. Triangulation produces a point cloud in local (camera) coordinates with points p_i.

2.2.6.3.8 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 known 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

$$\mathbf{x}=\sum_{i=1}^{m}\beta_i\mathbf{v}_i$$

(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 [Error! Reference source not found.] for practical solution, but the methods are complex and not summarized here.

2.2.6.3.9 Object Pose Estimation

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

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

(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^MS_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.

2.2.6.4 Algorithm Performance

In the case a single mono camera is used, the performance of this algorithm has been tested on a simulation of tracking a CubeSat. 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. The SHOT descriptor radius and cluster size parameters were varied to test the relationship of these variables to the resulting matches. Similar performance is expected from the stereo camera reconstruction as a similar set of DFNs is used. The reconstruction process is faster using LIDAR and ToF camera hardware as it does not require initial feature detection, matching, selection, and triangulation steps.

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

Table 6: Parameters for 3D Reconstruction and Tracking Tests

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 7 shows the timing results for each part of the 3D reconstruction process, and Table 8 shows the timing results (in seconds) for the 3D model-based identification and tracking process[1]. It can be seen from this that the majority of time is spent on keypoint production and FLANN search during the tracking process.

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

Table 7: Timing Results for CubeSat 3D reconstruction from image sequences

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

Table 8: Timing Results for CubeSat 3D model-based tracking

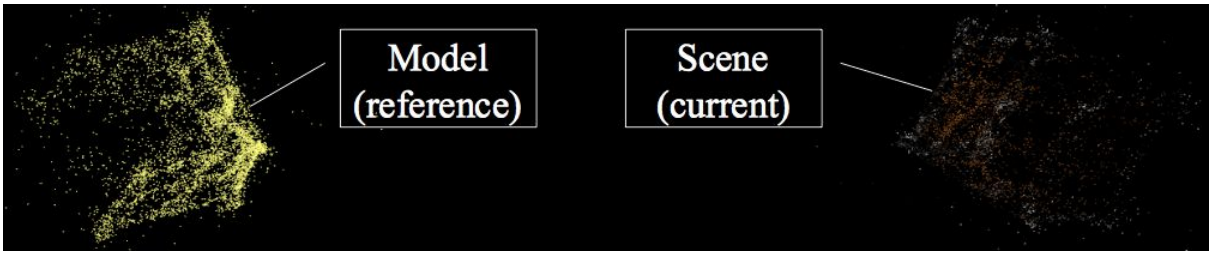

Figure 9: 1U CubeSat model (left) and reconstructed scene (right)

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 10. Figure 11 shows plots of the pose

[1] 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.

estimation accuracy in translation and Figure 12 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.

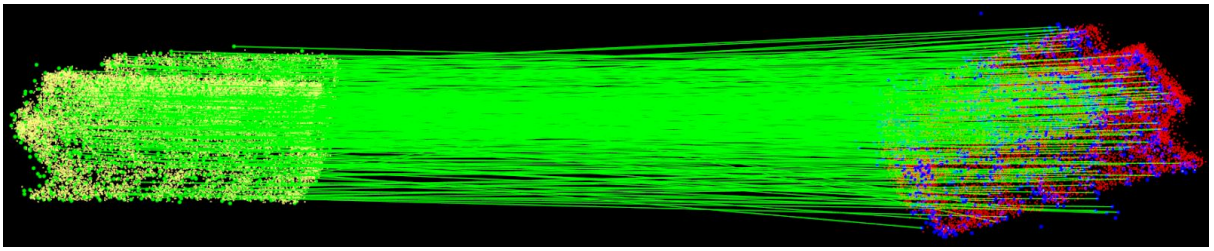

Figure 9: 3U CubeSat model (left) and match with scene (right)

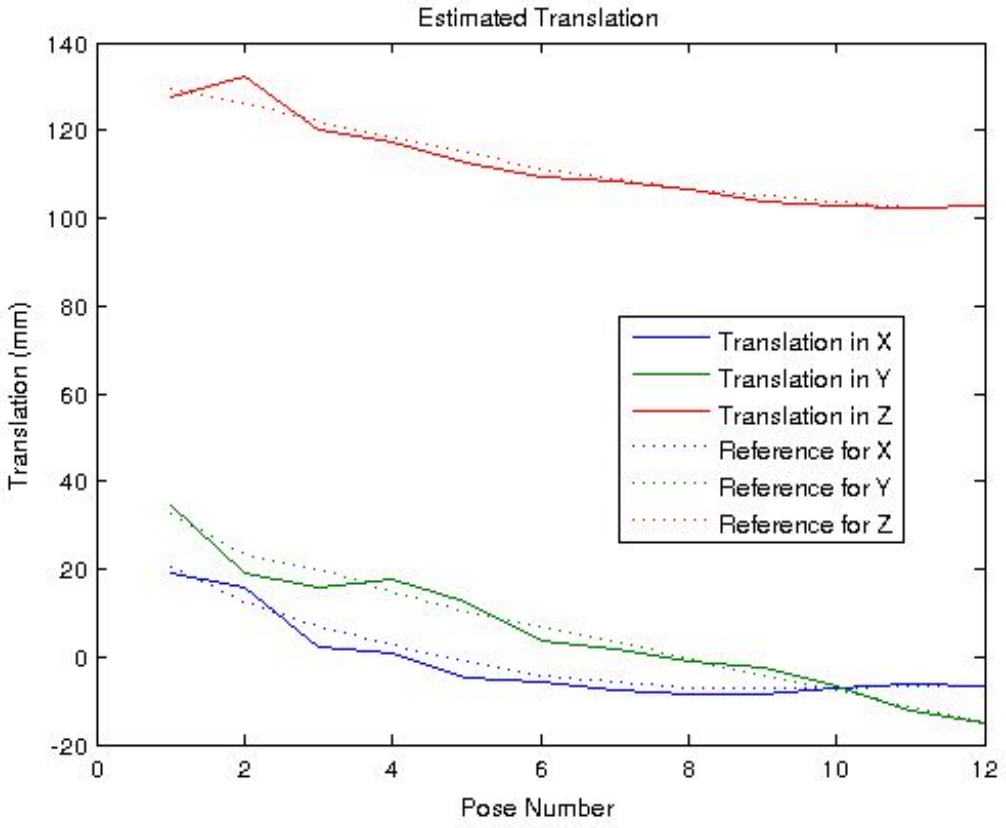

Figure 11: Tracking accuracy of 3U CubeSat model in translation

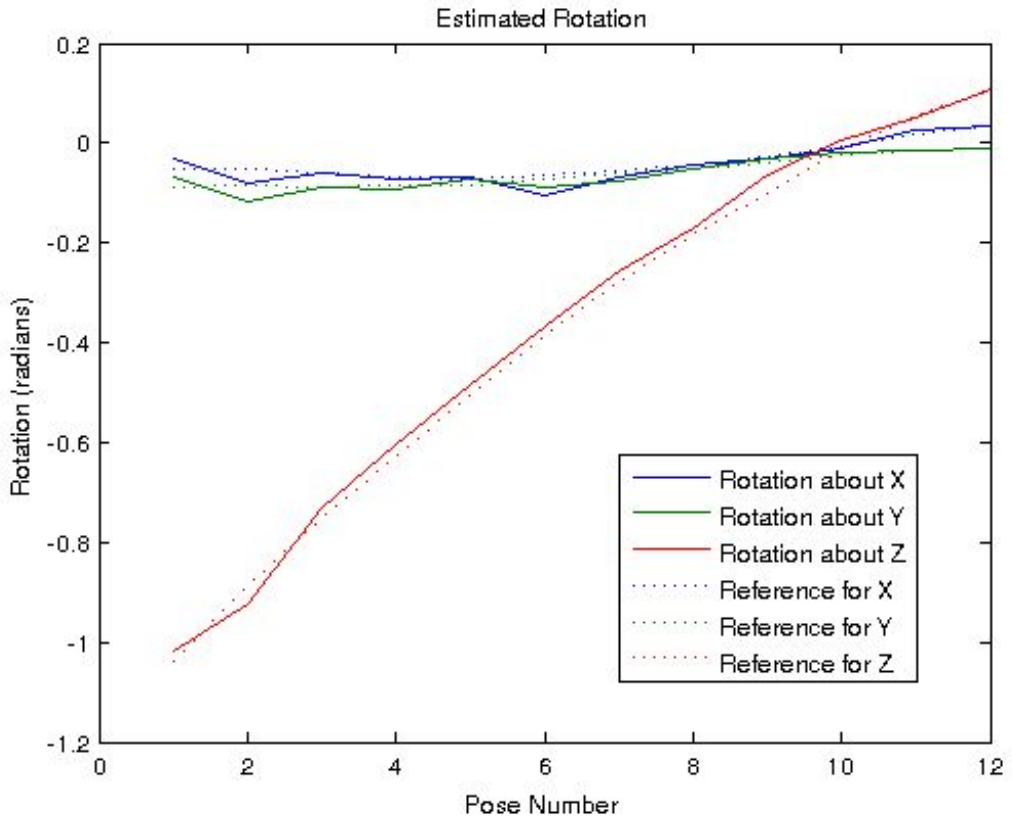

Figure 12: Tracking accuracy of 3U CubeSat model in rotation

This tracking accuracy is considered to be accurate within the close range (0-2m) scenario. Accuracy is expected to scale approximately linearly with distance, and can be expressed as a percentage of the distance to target R. For the close-range scenario presented here, positional accuracy is within 1% of R. For the mid-range scenario (2-17m), positional accuracy is expected to remain 1% of R.

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)

2.2.7 Use Case 6: 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 orbital 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, 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

2.2.8 Requirements

The system requirement [RD4] derives the requirements of the reference implementation in this document. The description of the demonstration and validation scenario of InFuse extends and solidifies this system requirement. Hence, we follow the requirement described in Appendix 7.1 to evaluate the implementation.

2.2.9 RI-SRC.SR-Detection, Reconstruction and Tracking

The RI-SRC.SR is a superset of the RI-INFUSE, hence the descriptions tailored to the RI-INFUSE will be applied for RI-SRC.SR. The further description specific to RI-SRC.SR will be reported once the interfaces to other OGs are matured.

3 System Modeling

This chapter describes the InFuse system architecture and its EGSE, Facilitators (OG6), ESROCOS (OG1) and ERGO (OG2) products.

The objective is to identify all components, interfaces and relationships of the system. The system is defined as a hardware and software subsystems which allow to implement scenarios described in chapter 2. Depending on the scenario that will be demonstrated, all parts of the overall system might not be required.

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 following a top down approach. As mentioned in previous chapters, the system architecture corresponding to RI-INFUSE and RI-SRC.SR scenarios are presented in dedicated sections.

3.1 Satellite and Robotic System

The actors in orbit servicing are servicer satellite, target or client satellite and robotic manipulator. The servicer and clients are usually reproduced with a respective mock-up satellite for ground validation and verification purposes. The motion of the satellites are simulated with industrial robots. We categorize the system as mission and simulation elements as in Table 3. The mission represents actual hardware systems deployed in space and the simulation elements define ground simulation which reproduces the behaviour of the actual space system. In general, the on-orbit servicer and client system are represented with a robotic system to reproduce the behaviour of the space systems. Here only kinematic aspects of on-orbit servicing is considered. In fact, the InFuse is concerned with perception and data processing aspects and will not address the simulation of satellite dynamics.

Satellite/Spacecraft	Ground Validation Hardware	Motion
Servicer	Servicer mock-up	Simulate with Industrial robot
Target	Target mock-up	Simulate with Industrial robot
Robotic manipulator	Robotic manipulator	Robotic manipulator

Table 3: On-orbit servicing simulation

Therefore, the satellite system in the context of on Ground validation in InFuse reduced to a robotic system. As defined in planetary track D5.2, the robotics system is mainly composed of robots, sensors, actuators, on-board computers, environments, ground stations, communication links and software.

- robot system: the robot in our case is an industrial manipulator having all actuators, sensors, controllers, on-board computers and software to reproduce servicer/target system that can be controlled in speed and direction,
- sensor system: it includes all exteroceptive and interoceptive sensors used for perception and sensing,
- actuator system: it include all actuators that are in the robot and required to provide necessary torque in order generate motion and interact with the world,
- computer system: it includes all the processing units,
- environment: it includes the representative space environment where the on-orbit servicing is performed.
- ground station system: it is the set of computers that allow the end-user to interact with the manipulators,
- communication link system : it consists of all communications links that required by ground stations, on-board computers, sensors, actuators, microcontrollers to communicate,
- embedded software system: it is all the necessary softwares to operate on-orbit servicing. It could be functionally decomposed as detection, tracking and reconstruction.

3.2 RI-INFUSE

This section describes the InFuse system architecture dedicated to the implementation of the RI-INFUSE scenarios.

3.2.1 System Components

The Servicer/Target system : is the robot System (shown in Fig. 11), consisting of

- servicer
- target

- sensor system
- environment
- platforms such as ground station and on-board system

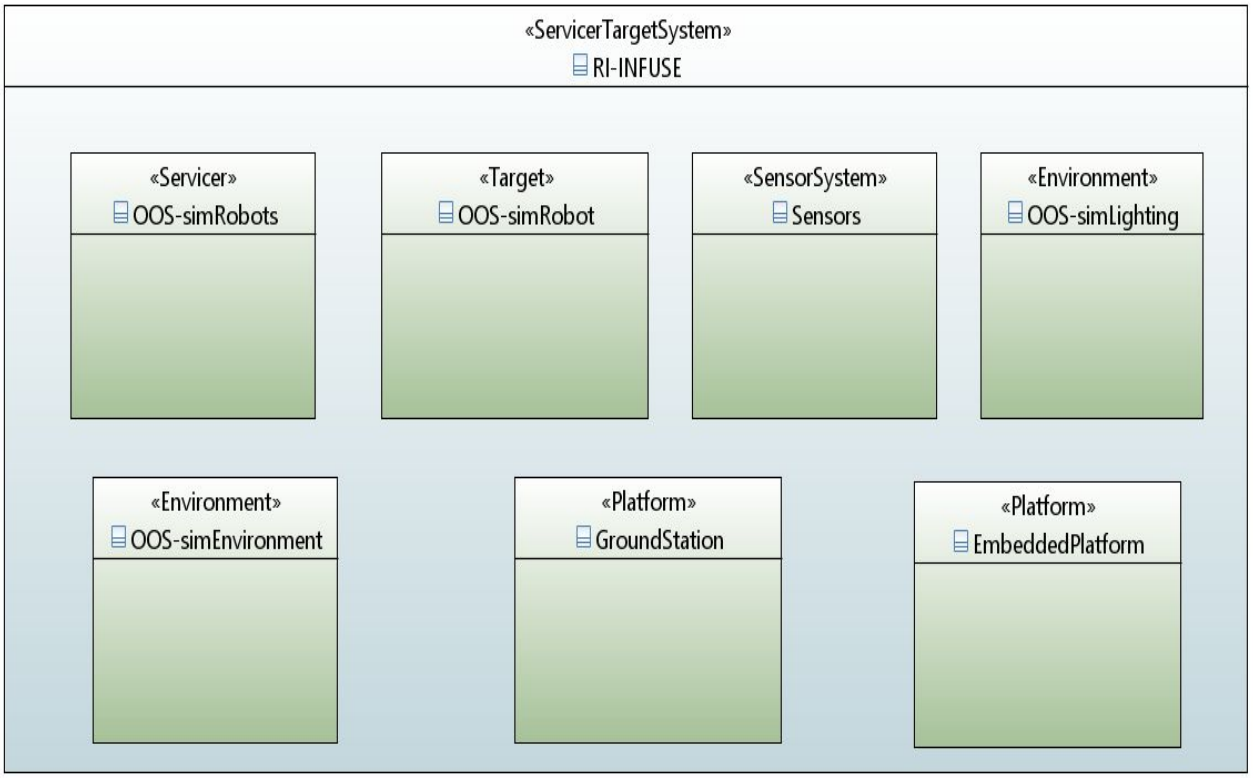

Figure 9: On-orbit servicing system consists of servicer and targets as the main components.

Sensor system: These are main sensors used for on-orbit servicing, including

- stereo cameras
- LIDAR
- IMU
- force/torque sensors (for grasping/docking)

Each sensor can be used independently or as a system of sensors (Fig. 12) for sensor fusion for an on-orbit servicing. Notice that one sensor is preferable to the other depending on the range of operation between the servicer and client satellites.

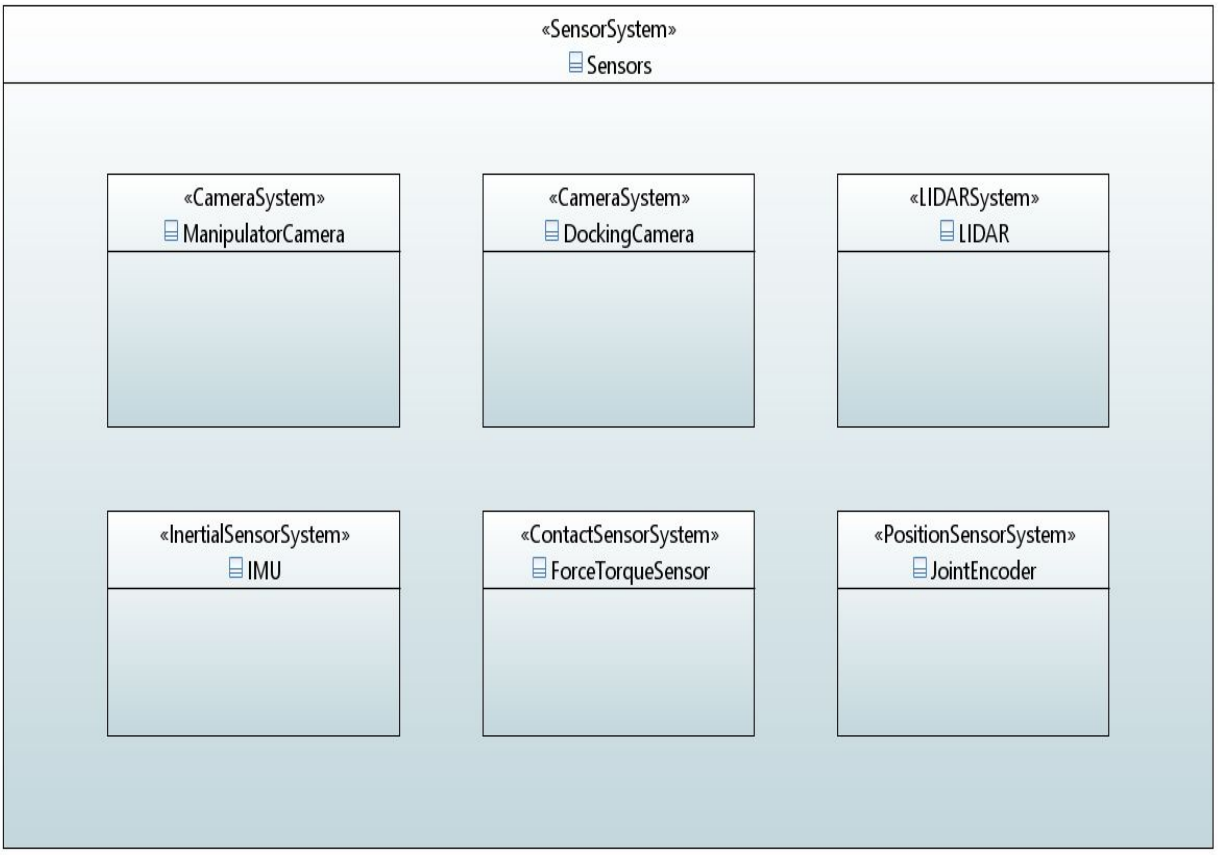

Figure 10: Sensor system for an orbit-servicing.

Software system: The software is a crucial component of the on-orbit servicing robotic system. The overview of the the software system of the orbital reference implementation is shown in Fig.13. The sensor system (left) provides raw data in order that the software system (middle) produces the required measurements such as position and attitude.

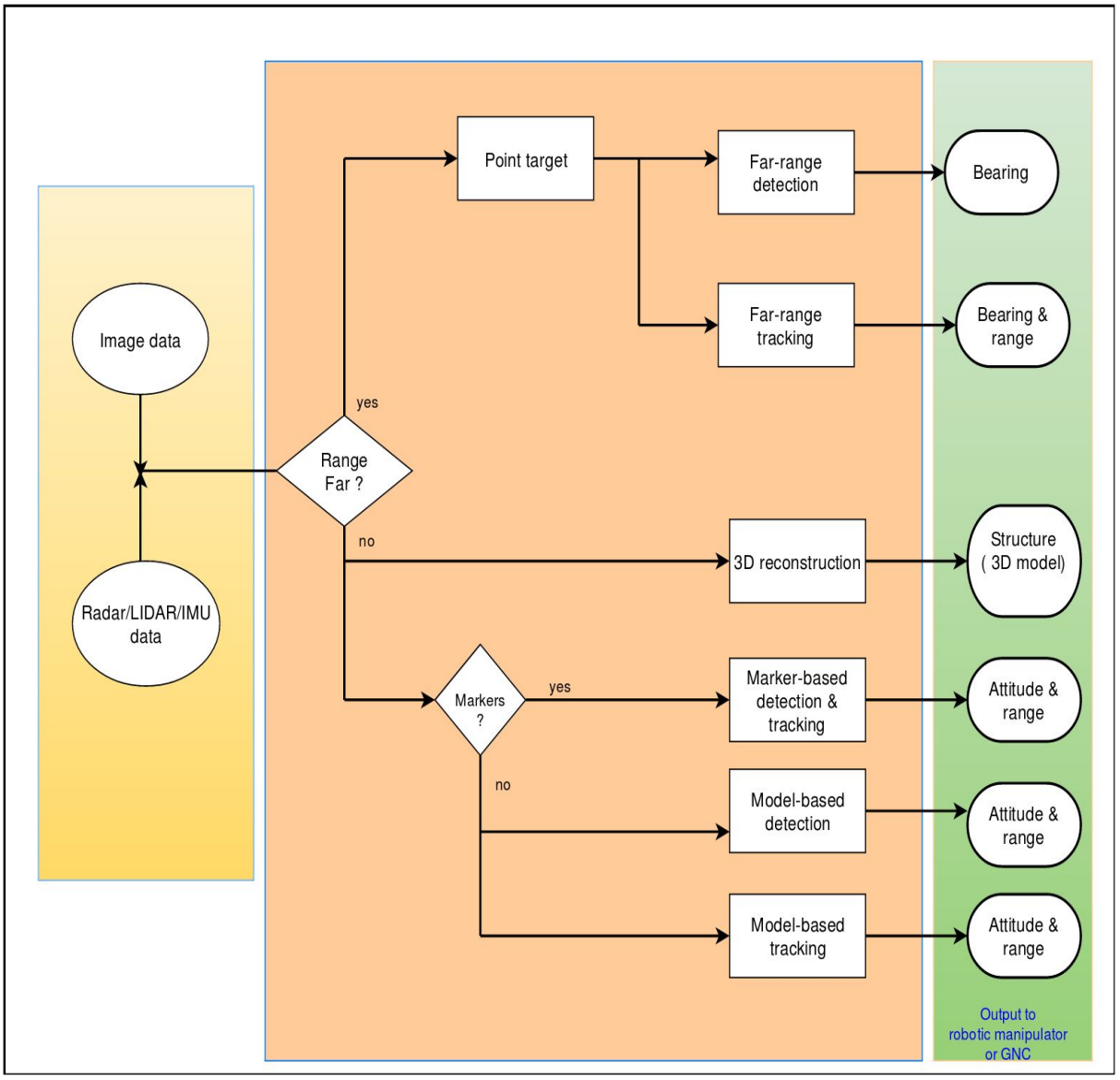

Figure 11: Overview of the Software system of the orbital reference implementation.

In the figure above, The sensor system (left) supplies raw data in order the software system (middle) produces data product (right) which can be communicated to an external user such as ERGO.

3.2.2 On-orbit Servicing Simulator

Here we provide an example ground simulation system for an on-orbit servicing. The DLR OOS-sim facility is mainly used for close-range operation and equipped with (Fig. 14)

- two industrial robots
- servicing robotic arm

- Manipulator camera
- Docking camera
- force/torque sensors

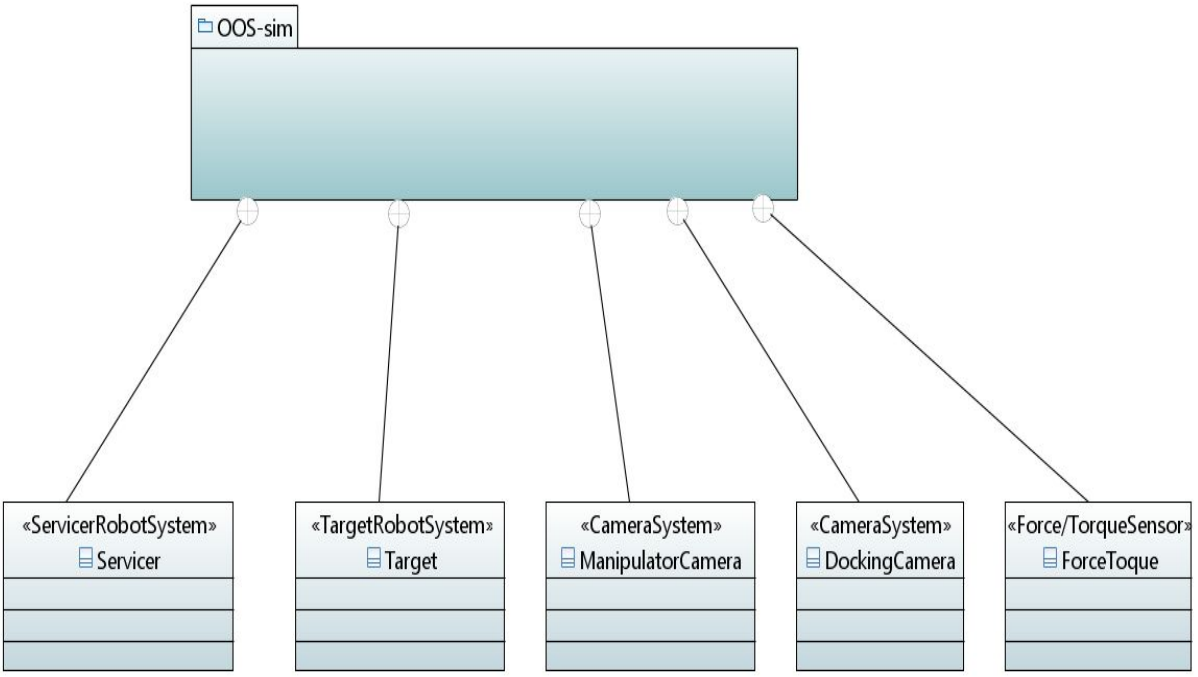

Figure 12: The overview of the main on-orbit servicing ground simulation.

Finally, we provide an example that shows a deployment of the CDFF on the DLR OOS-sim (see Fig. 15).

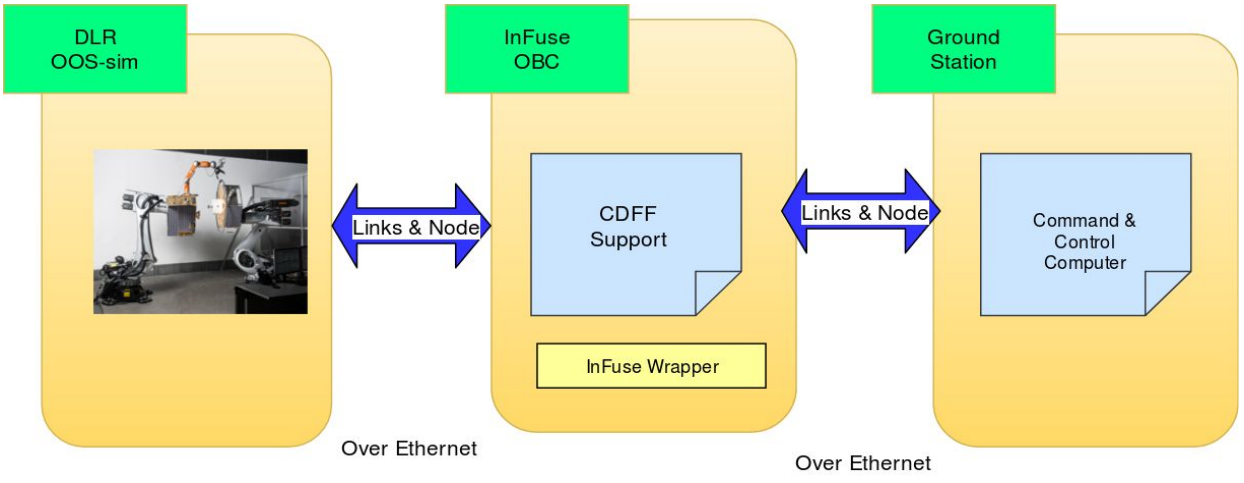

Figure 13: Deployment overview of the InFuse with the DLR OOS-sim. The Links and Nodes is the DLR robotics middleware used here for inter-process communication.

3.3 RI-SRC.SR

RI-SRC.SR is a subset of RI-INFUSE where less sensors will be available. Any major difference between the different system architecture will be included in this document. The detailed interface and workflow of OG3 between OG4 and OG2 is provided in Appendix 7.3

4 Detailed Architecture and Design

This chapter presents detailed description of DFPC and associated DFN corresponding to the use cases provided in chapter 2.

1.1 4.1 DFPC Architecture View

First, we focus on the architecture of the data fusion processing compound/chain (DFPC) and provide detail design of associated DFNs. Thus, at this level we aspire to identify and describe

- each DFN used in DFPC,
- DFN internal interfaces,
- internal and external interfaces of DFPcs .

The DFPC description follows the DFPC specification provided in Appendix 7.2. 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*: description of the shared data between the DFNs in the given DFPC, and the interfaces between this data and the various DFNs.
- *Control Description*: description of the control flow within a DFPC: the order in which DFNs are called, DPM access to shared data, synchronicity of time stamped data. The control flow will be achieved by the Orchestrator for implementation.

The specification, definition and the purpose of the DFPC specification are tabulated in Table 4.

DFPC description	purpose	specifies/defines
Data Flow description	Provides a layout and ordering of different DFNs	-inputs/outputs of DFPC -inputs/outputs of each DFN -shared data between DFNs
Data Product management	Manages the CDFF products such as pose, map, model and features	-shared data structures among DFNs -processing units that query and insert those data structures

Control description	Describes flow of processes within DFPC. Temporal execution of a DFPC is dictated by the orchestrator	-temporal course of the execution of a DFPC

Table 4: DFPC Specification: data flow and control description, and data product management.

The following template is used to describe the implementation detail of each DFPC:

- *List of DFNs used* **:** as described in section 4
- *Data structures:* data types of the input/output and shared Data structure, relation to the DPM (as a client, a provider).
- *DFPC Parameters :* the user of the DFPC will select parameters of algorithm. The default parameter will be provided. These parameters are set in a configuration file

The list of DFPCs and the partners responsible for developing them are provided in Table 5. The sections below present the description of each DFPC.

DFPC	**Partner**
Mid-and close-range detection	Magellium
LIDAR-based tracking	Magellium
Mid-and close-range visual tracking	DLR
3D reconstruction	USTRATH
Haptic Scanning	SPACEAPPS

Table 5: List of DFPCs and responsible partners

The following sections present the implementation details of the DFPC in the context of the **RI-INFUSE**. Since the CDFF, as well as ESROCOS, are not available and operational yet, we implement the DFPC using another RCOS and middleware.

Details of INFUSE-RI:

- Middleware, RCOS (GENOM, YARP, ROS),
- DFPC Controller : implementation detail (e.g. GENOM/YARP State Machine),
- Specific data structures,
- Deployment scheme.
- AHPC interfaces : sensor acquisition, display

4.1.1 DFPC : Far-range Object Tracking

This DFPC responds to the following reference implementation scenarios:

- 2.2.1 RI-INFUSE-Detection, Reconstruction and Tracking
    - 2.2.2 Use Case 1 : Far-range Target Detection and Tracking

It is used to provide a relatively simple bearing-only relative localisation of the target asset when its distance with regard to the chaser 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,
- Chaser attitude from AHRS,
- Radar range.

DFPC Outputs:

- Estimated target bearing and range.

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 chaser and target 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 chaser. 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.

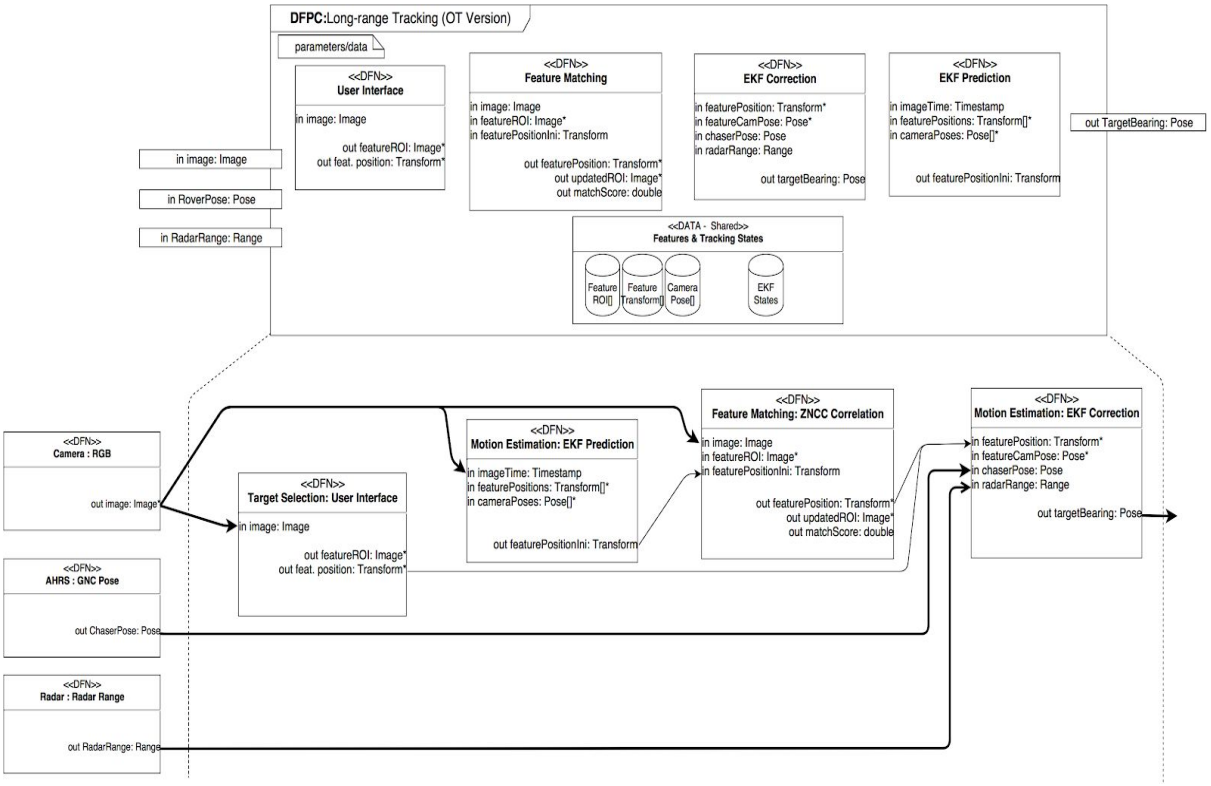

Figure 16: Long-range Tracking Data Flow Description.

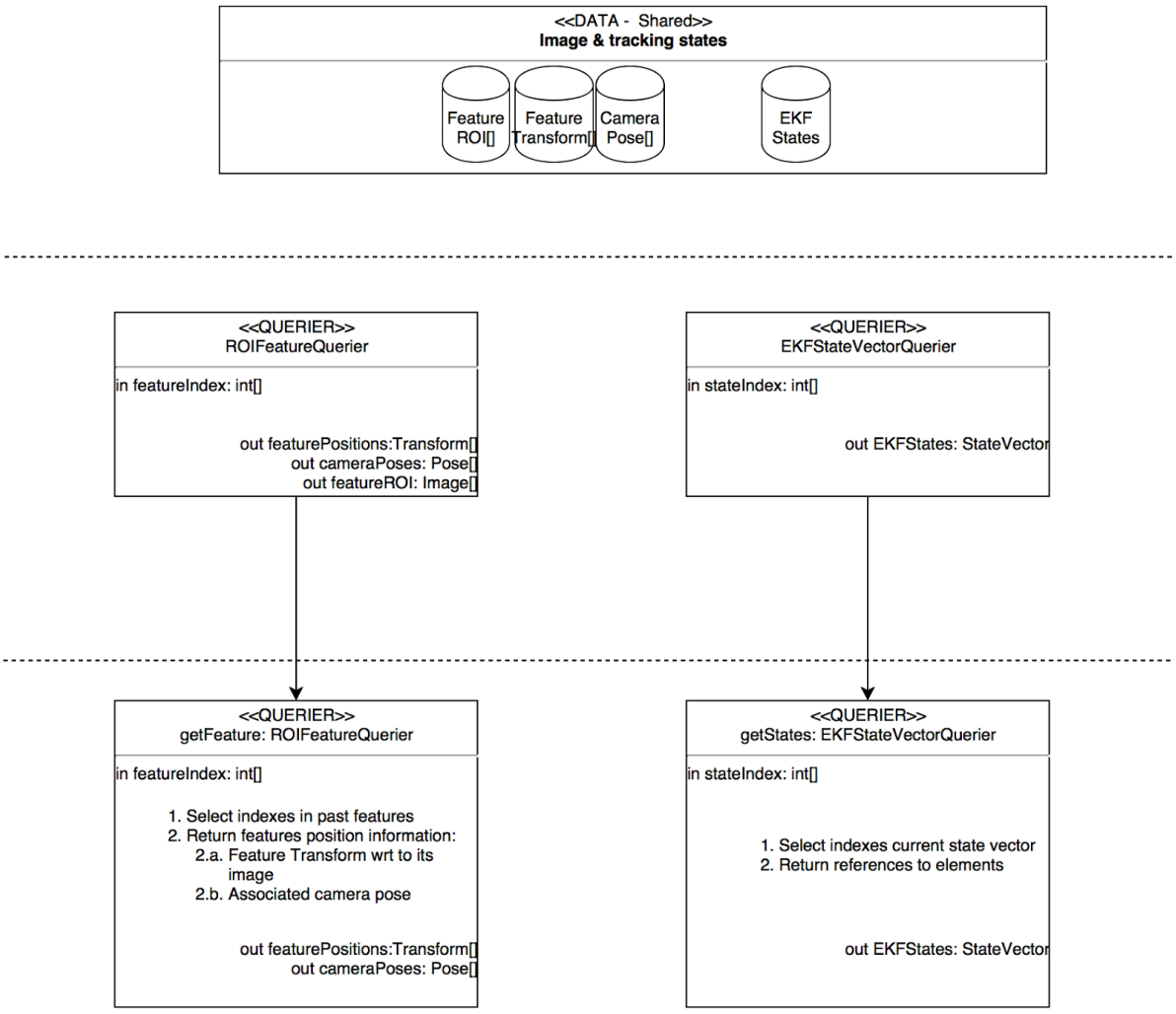

Figure 17: Long-range Tracking Data Product Management.

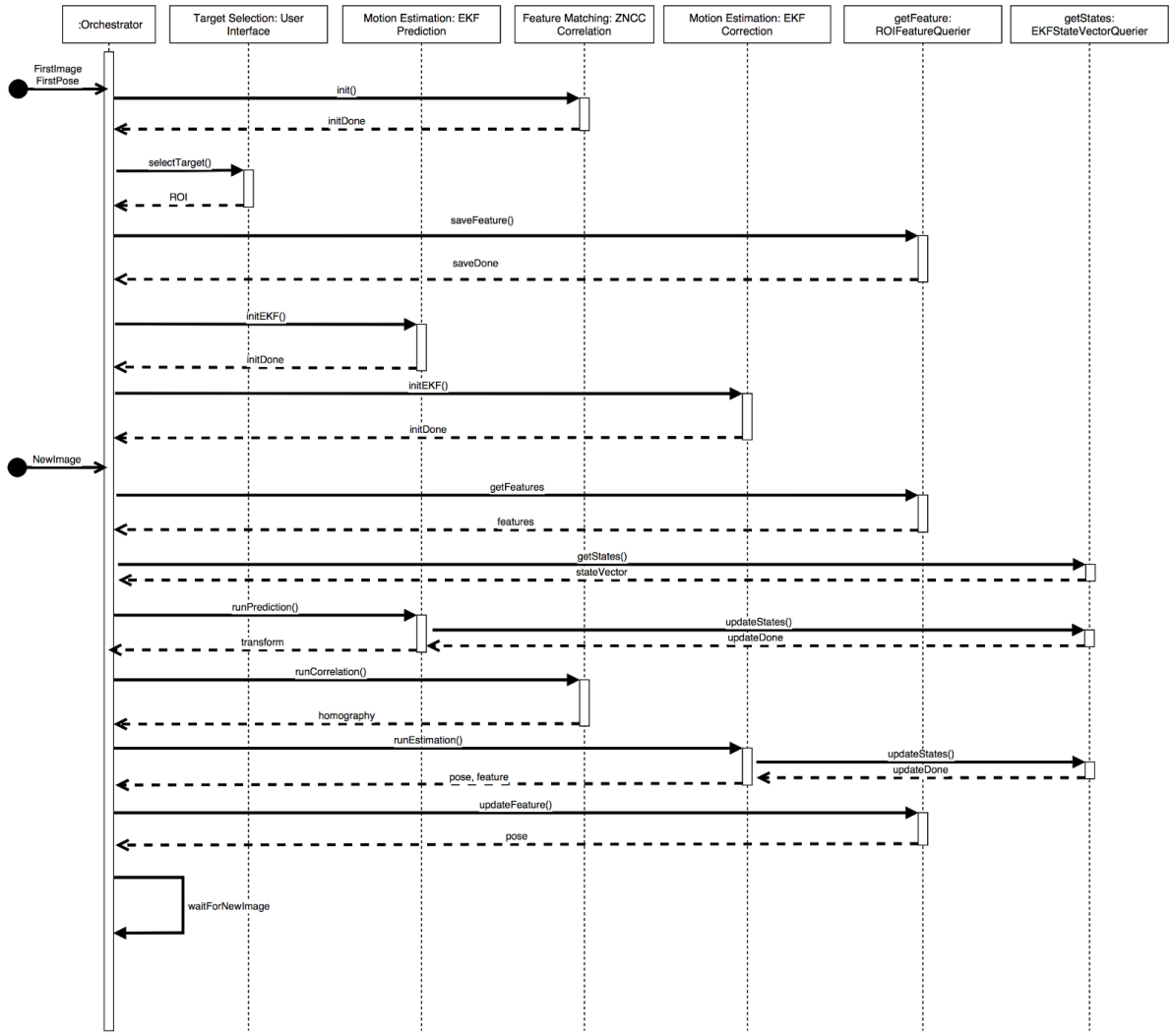

Figure 18: Long-range Tracking Control Description

4.1.2 DFPC : Mid- and Close-range Target Detection

This DFPC responds to the following reference implementation scenarios:

- 2.2 2 Detection, Reconstruction and Tracking of a target
  - 2.2.3 Use Case 2 : Mid-range 3D Model Detection and Tracking

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 chaser 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.

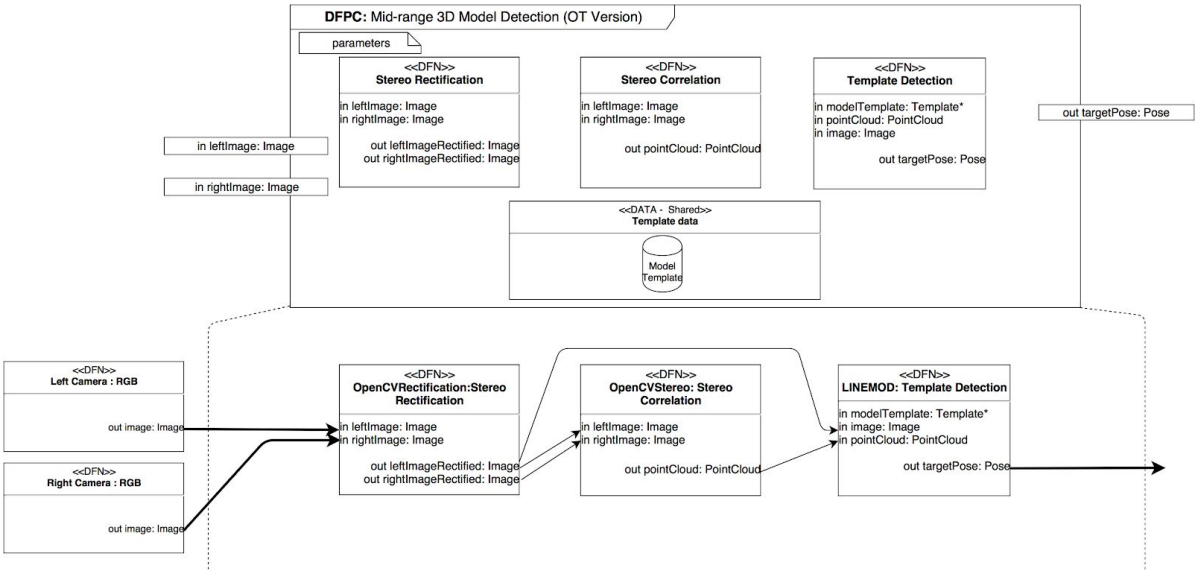

Figure 19: Mid-range 3D Model Detection Data Flow Description.

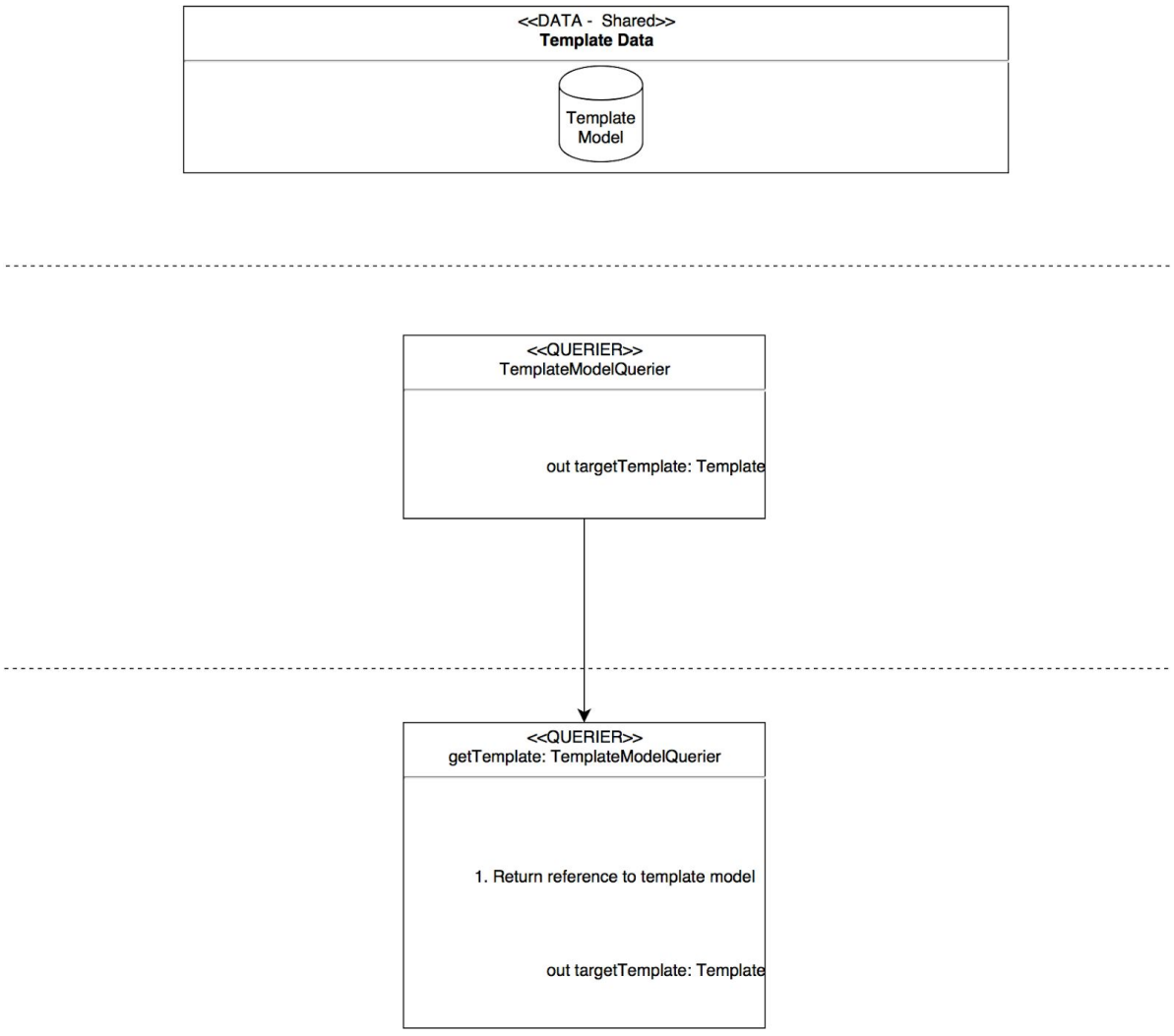

Figure 20: Mid-range 3D Model Detection Data Product Management

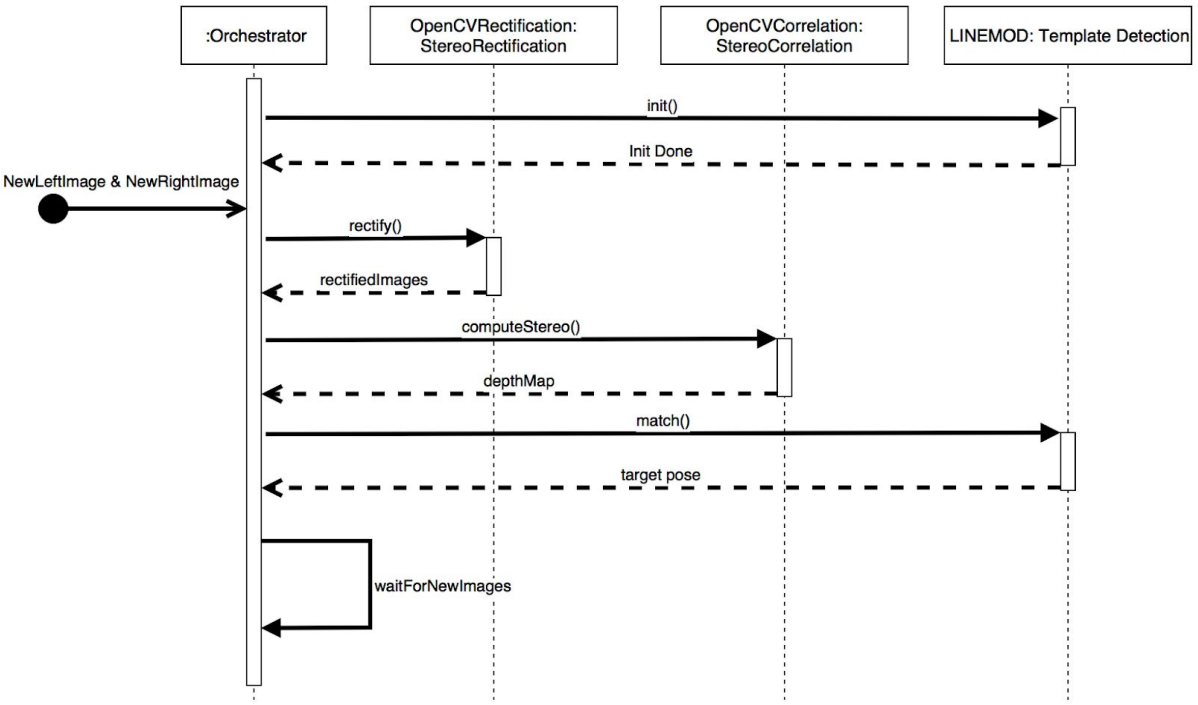

Figure 21: Mid-range 3D Model Detection Control Description

4.1.3 DFPC : Mid- and Close-range Target Tracking

This DFPC responds to the following reference implementation scenario:

- 2.2.1 RI-INFUSE-Detection, Reconstruction and Tracking
  - 2.2.3 Use Case 2 : Mid-range 3D Model Detection and Tracking

This DFPC is activated once the chaser 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): edges and corners, and KLT keypoints. The tracking function is thus adapted for textured or untextured objects, with visible edges or not. In this implementation, we attempt to augment the camera input with a radar range measurement, which will be used to add robustness to the tracking DFN.

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,
- Radar range.

DFPC Outputs:

- Estimated relative chaser pose with respect 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: A self-contained DFN which 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.

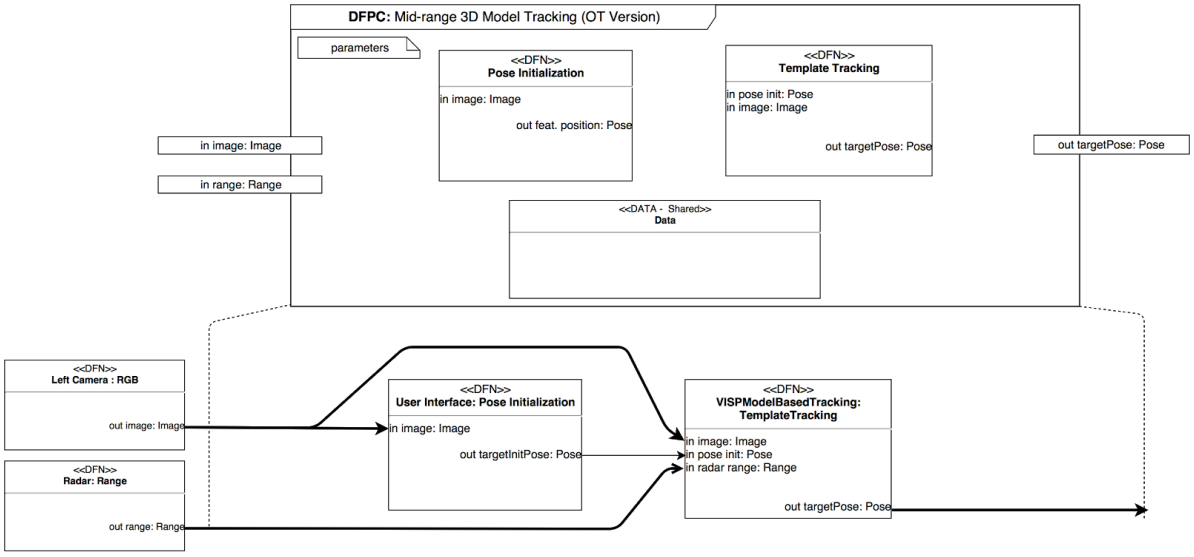

Figure 22: Mid-range 3D Model Tracking Data Flow Description

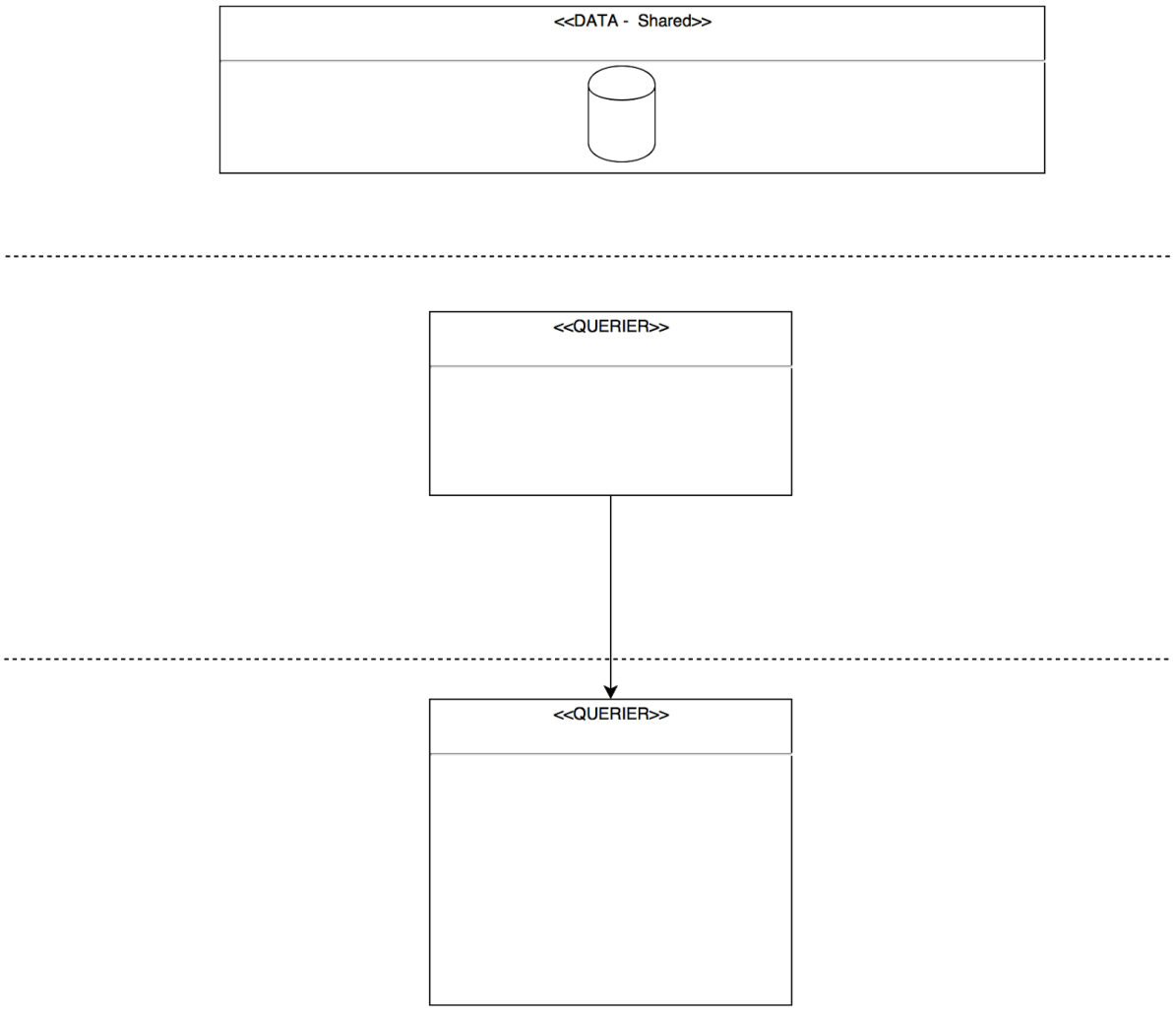

Figure 23: Mid-range 3D Model Tracking Data Product Management

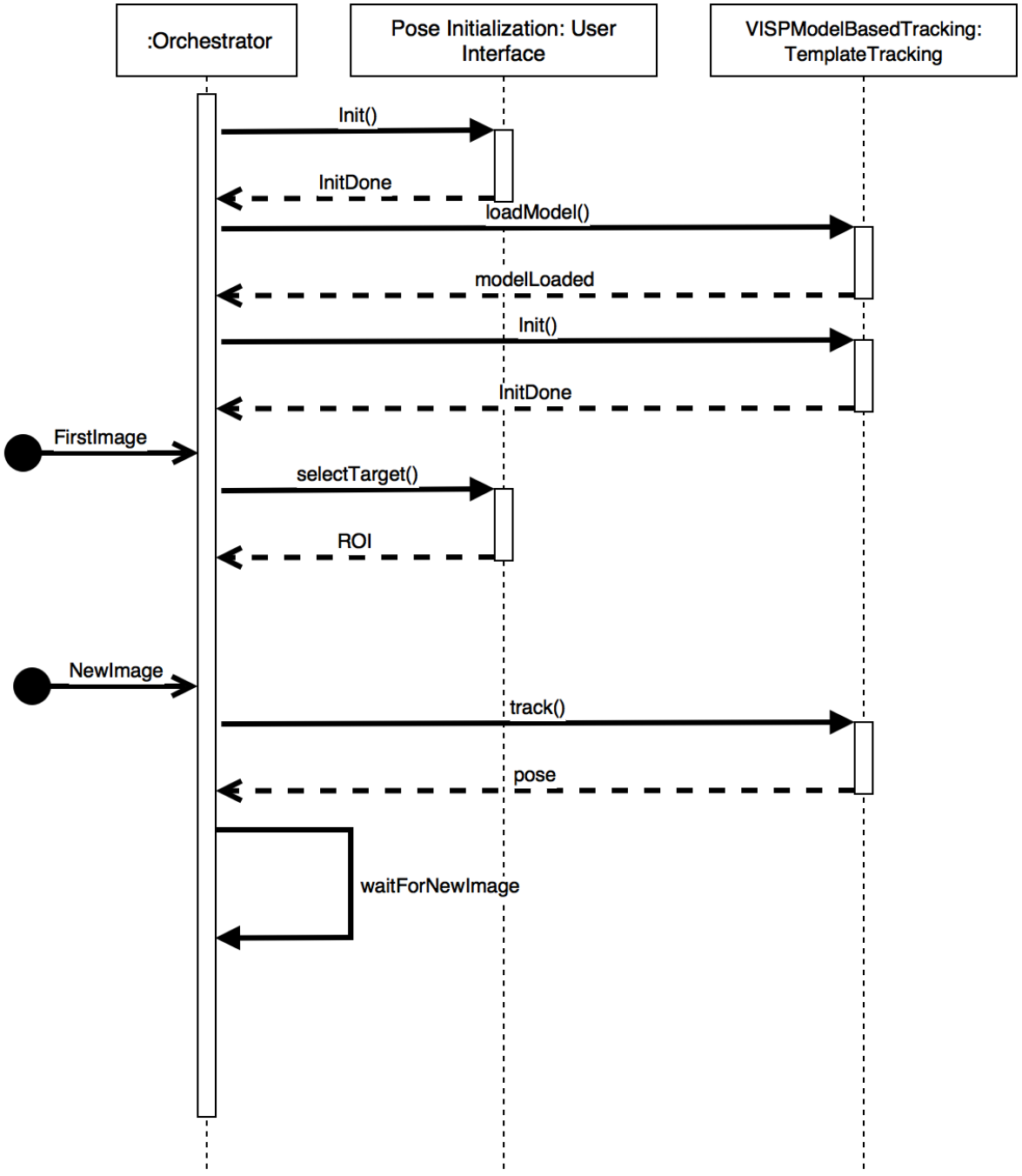

Figure 24: Mid-range 3D Model Tracking Control Description

4.1.4 DFPC: LIDAR-based Tracking of a Target

This DFPC responds to the following implementation scenarios:

- 2.2.1 RI-INFUSE-Detection, Reconstruction and Tracking
  - 2.2.5 Use Case 4: LIDAR-based tracking of a target

The general goal of this DFPC is to perform a robust tracking of a known target described by a point cloud, either obtained through LiDAR sensors, stereo cameras or ToF cameras. The method requires that a sufficiently dense point cloud model of the target is provided in advance by the user.

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 a dense ICP matching step between the input point cloud and the provided model. The target model is a high density point cloud created offline from a 3D model, or from prior 3D reconstruction. 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 chaser's AHRS sensor.

For a detailed description of the EKF DFN reused in this DFPC, refer to section 4.1.5.

DFPC Inputs:

- Chaser attitude from AHRS,
- Point cloud with associated metadata from LiDAR sensor or stereo camera or ToF camera. The point cloud density can vary, depending on the input sensor used.

DFPC Outputs:

- Estimated chaser 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 chaser and target motion models 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 chaser. 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 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.

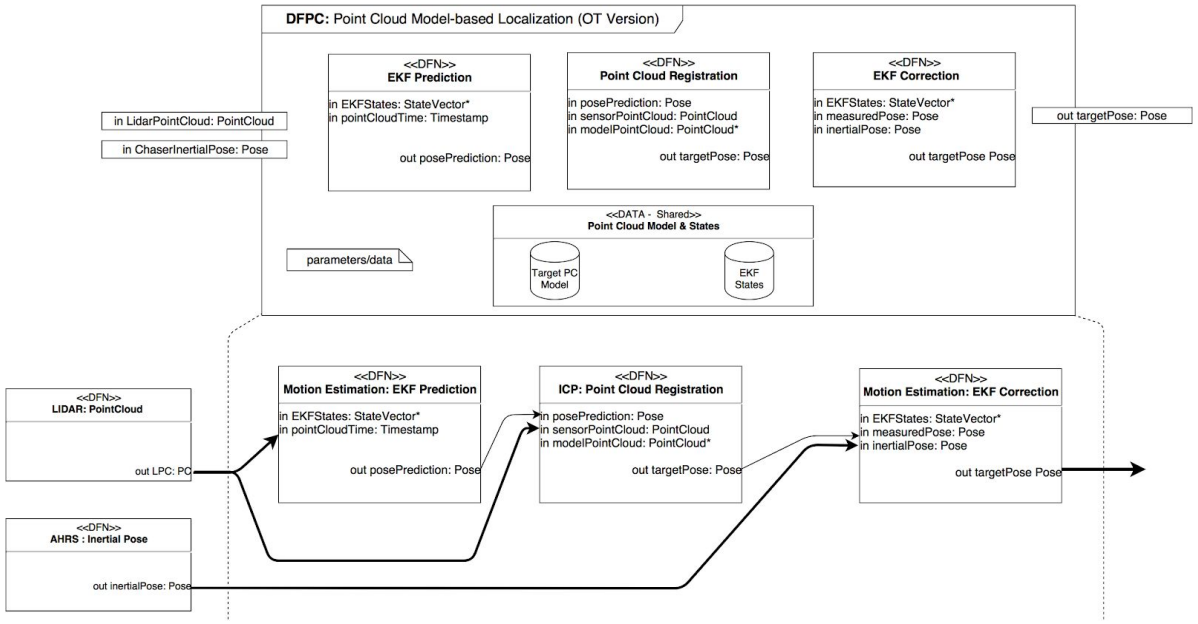

Figure 25: ICP Point Cloud Matching Data Flow Description

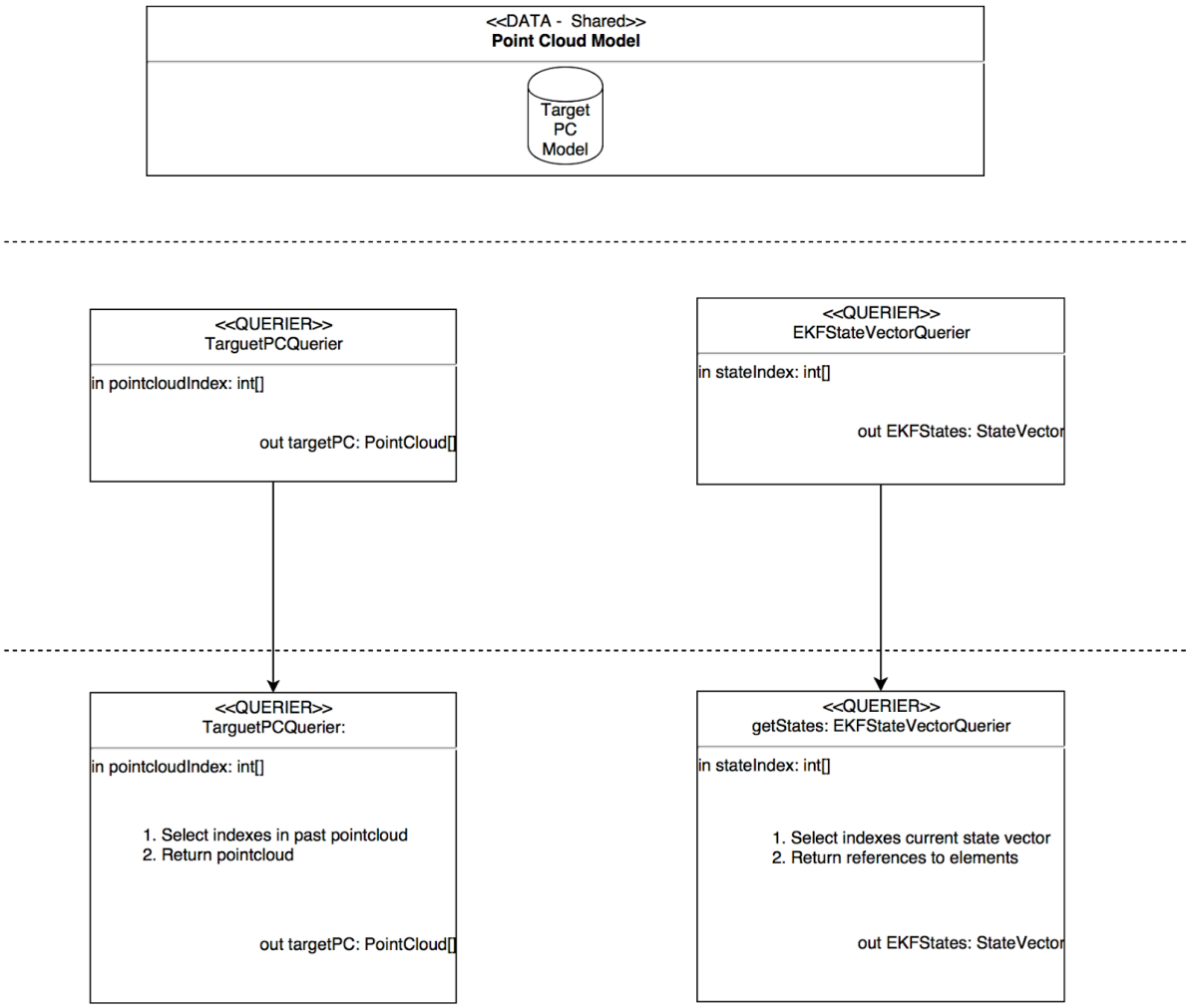

Figure 26: ICP Point Cloud Matching Data Product Management

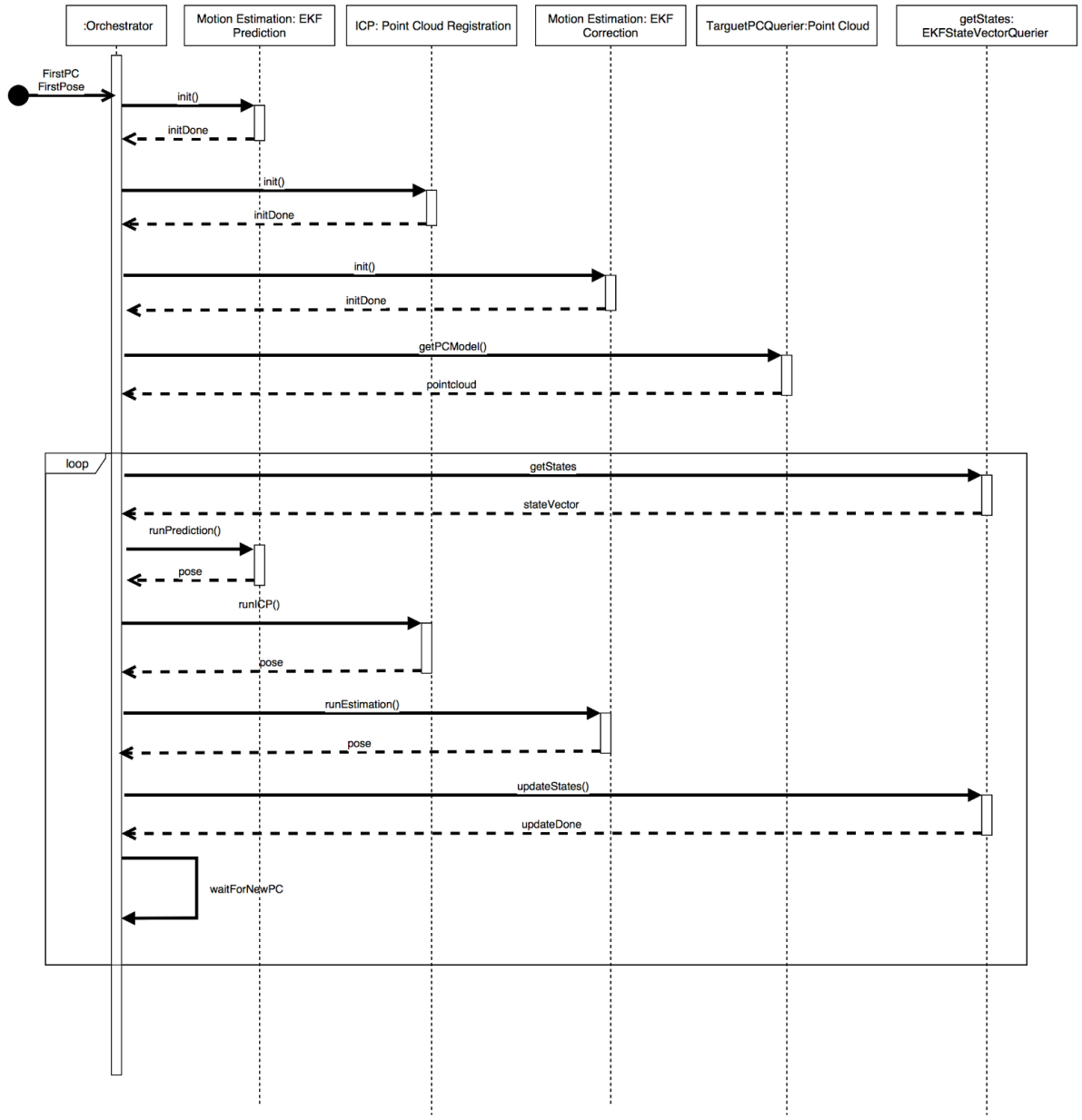

Figure 27: ICP Point Cloud Matching Control Description

4.1.5 DFPC : Mid- and Close-range Visual Tracking of a Target

This DFPC relates to the use case described in

- 2.2.1 RI-INFUSE-Detection, Reconstruction and Tracking
  - 2.2.6 Use Case 5: Visual Tracking of a Target and Estimation of Robot Relative State

DFPC Inputs :

- [Stream] Stereo images with associated metadata
- [Data Product] Valid initial guess of the target pose,
- [Parameter] Model file similar to wavefront format,
- [Parameter] Tracker parameters,
- [Parameter] Calibration parameters

DFPC Outputs:

- [Stream] Estimated pose
- [Stream] Estimated local velocity

The DFPC will be composed of the following DFNs (Fig. 28):

- Image undistortion
- Edge detection for an an edge based tracking,
- Kalman prediction for capturing frame-to-frame local motion
- Kalman correction for updating the prediction with measurement
- Visibility determination
- Contour sampling
- Matching model and image edges
- Pose estimation

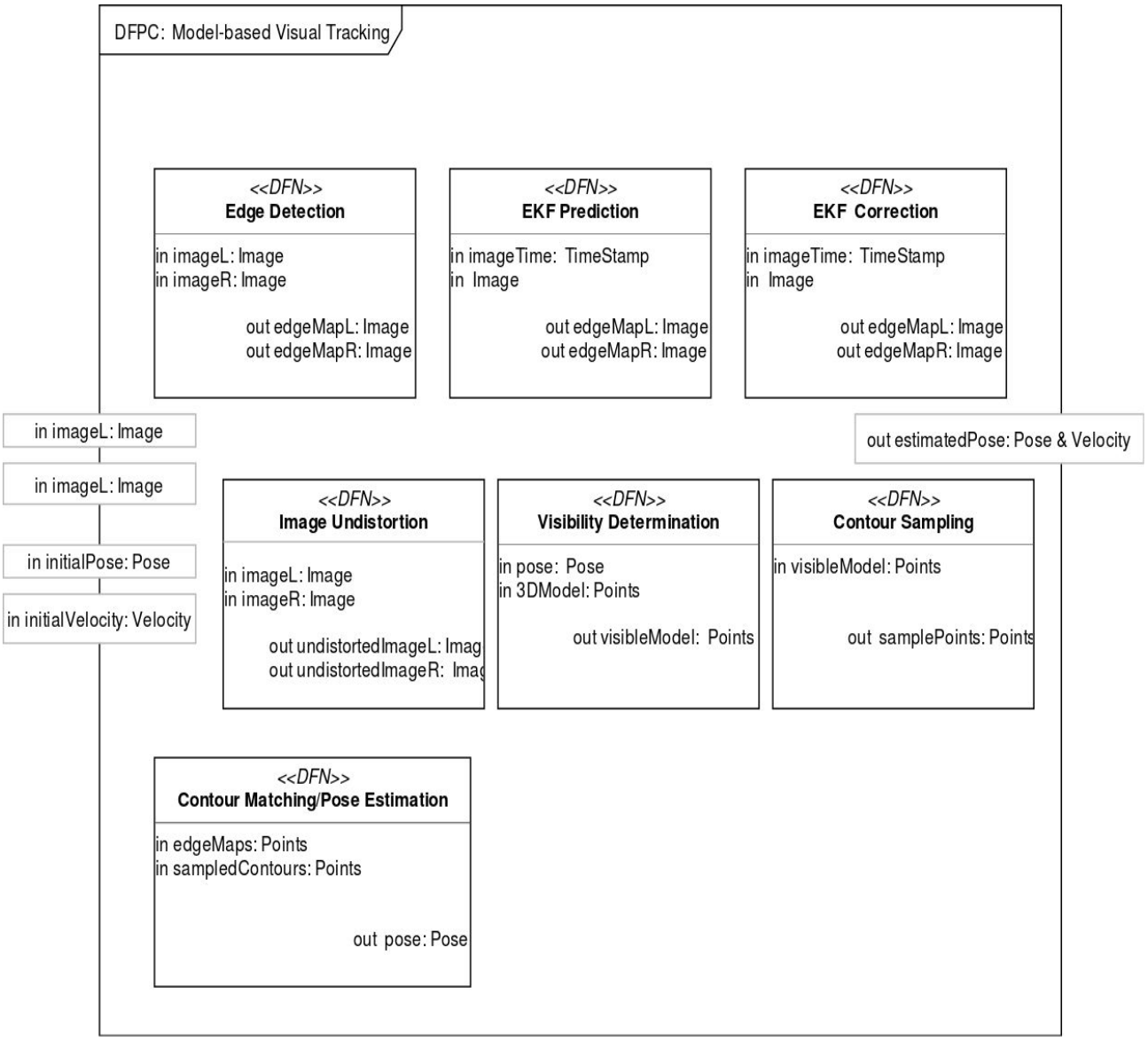

Figure 14: The DFN of the model-based visual tracking.

The following figure details the interaction of DFN components inside the DFPC. Notice that the objects in the sequence diagram: model, m_contour, m_kalman and mainTracker correspond to visibility determination, contour matching, EKF DFN respectively.

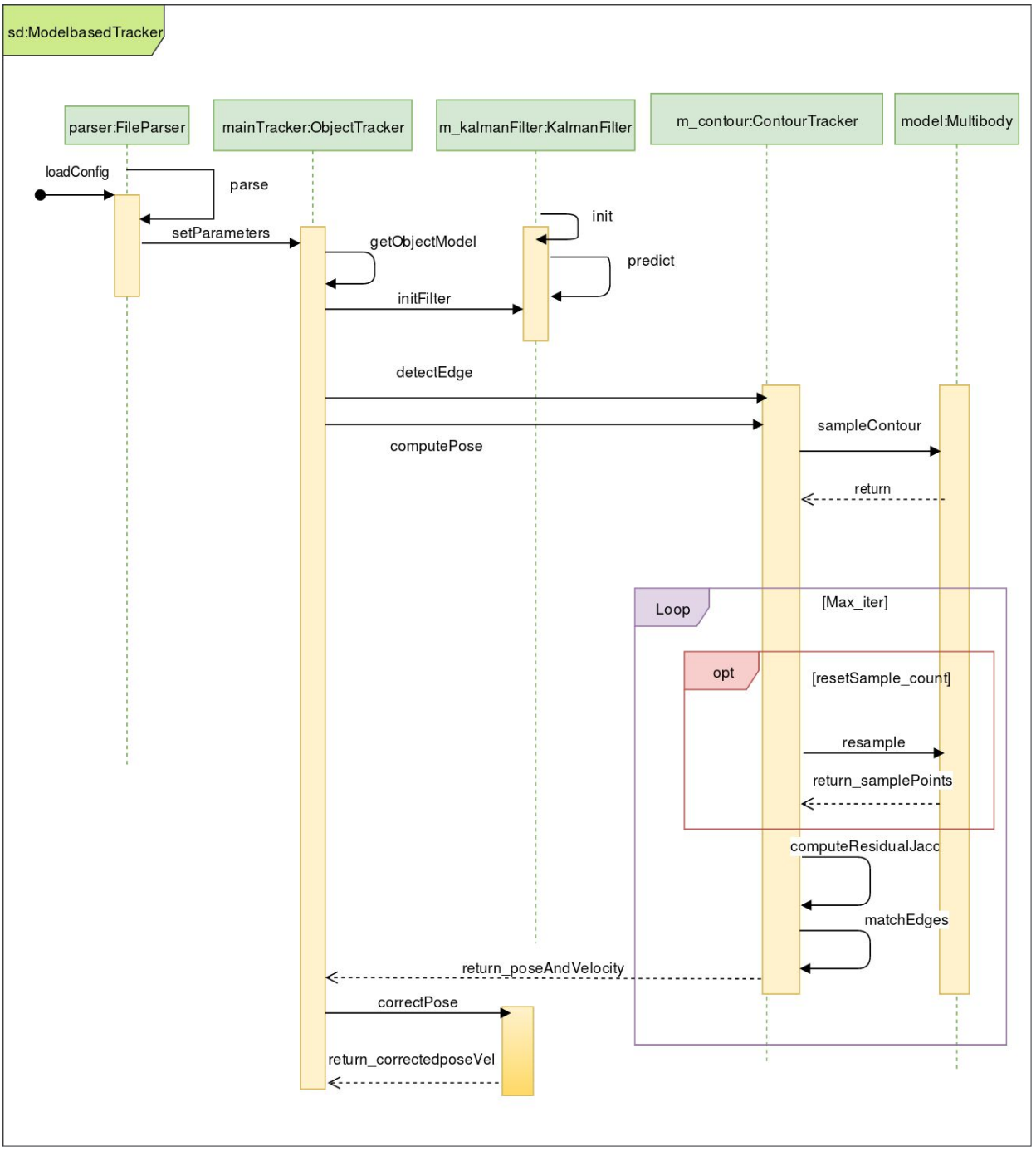

Figure 15: Sequence diagram of the model-based visual tracker.

The camera-based pose tracking is described by its DFNs (Table 6), DFPC parameters (Table 7), inputs and outputs. Moreover, the input and output data types are defined to enable interface to a Sensor Suite and data product consumer such as ERGO.

List of DFNs

Data Fusion Node (DFN)	function
Edge detection	extracts image edges
Kalman filtering	Predicts the state based on a motion model and updates with measurement
Image undistortion	Corrects image pixels distorted due to camera lens
Visibility determination	Computes the visible model features from the current camera view
Contour sampling	Samples points along the visible contours
Contour matching and pose estimation	Estimates pose by aligning image edges to sampled contours iteratively

Table 6: Data Fusion Nodes of the model-based visual tracking 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 image 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.

DFPC Parameters: There exist two types of parameters which should be specified before the DFPC is deployed: tracker and camera parameters. The tracker and camera parameters are related to the DFN and internal parameters, and camera properties respectively.

Parameter	Parameter type
Max number of sample points	Tracker parameter
Min distance between a polygon plane and the camera center [mm]	Tracker parameter

Subsampling factor while collecting visible pixels [1=take all, ….]	Tracker parameter
Minimum length of a valid 3D segment [mm]	Tracker parameter
Search distance [pixel integer]	Tracker parameter
Canny Thresholds	Tracker parameter
Angular threshold for matching edges [deg]	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
Minimum parameters wrt the last Visibility Line Determination (VLD) to call for new VLD [deg, mm]	Tracker parameter
Kalman filter process noise	Tracker parameter
Kalman filter measurement noise	Tracker parameter
Kalman filter initial covariance	Tracker parameter

Table 7: DFPC Parameters of the model-based visual tracking

Parameter	Parameter type
Number of cameras	Camera parameter
Resolution of camera [pixel]	Camera parameter
Minimum and Maximum depth [mm]	Camera parameter
Projection matrix	Camera parameter
Distortion parameters	Camera parameter
World-to-camera transformation	Camera parameter

Table 8: Camera parameters for the model-based visual tracking DFPC

Input and output data description: The data types are described according to ESROCOS ASN.1 in section 5.2 Datatypes. Here, we provide the data types specific to the DFPC, model-based visual tracking.

Input data	Data type description	Meta-data	Data type (ASN.1)
image	Gray images from one or two cameras	Frame-mode-t Frame-size-t	mode-grayscale width T-UInt8 height T-UInt8
time	Time [s] of image acquisition	Time	microseconds T-Int64, usecPerSec T-Int32

Table 9: Input data type description for visual tracking DFPC

Where T-UIntx are ASN.1 INTEGER defined in ESROCOS.

Output data	Description	Meta-data	Data type (ASN.1)
position	Position in x ,y and z-direction		vector3d
orientation	attitude		AngleAxisd
Velocity	Translational velocity in local coordinate frame x,y and z		vector3d
Angular velocity	Rotational velocity in local coordinate frame x,y and z		vector3d
status	Success/failure in pose estimation possibly with failure/error code		T-Int8

Table 10: Output data type description for visual tracking DFPC

Where vector3d:= SEQUENCE(SIZE(1..3)) OF REAL and

AngleAxisd:=SEQUENCE(SIZE(1..4)) OF REAL as defined in ESROCOS.

4.1.6 DFPC: 3D Reconstruction

The 3D Reconstruction DFPC is specifically for reconstruction of a scene point cloud based on motion of the target (structure-from-motion), which can supplement or replace a point cloud obtained by LIDAR.

DFPC Inputs :

- [Stream] Left and Right stereo images with associated metadata
- [Parameter] Camera calibration matrix
- [Parameter] Additional optional parameters

DFPC Outputs:

- [Stream] Ego-motion estimation from PnP for the camera/tracker
- [Stream] Scene point cloud as reconstructed over time from motion

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

- Feature detection: for identifying features in the scene
- Feature matching: for correlating features between images
    - Stereo correlation or separate for monocular use
- Fundamental matrix calculation: for finding the homography between images
- Triangulation of features: to locate features in 3D space
- Perspective-and-Point (PnP) solution: to find the ego-motion of the camera

Fig.30 details the interaction of DFN components inside the DFPC. The parameters that can be set for this DFPC include the following:

- camera calibration parameters
- maximum number of features
- feature scaling
- edge threshold
- patch size
- first feature level (in the case of ORB descriptors)
- number of feature levels (in the case of ORB descriptors)
- WTA K value (in the case of ORB descriptors)

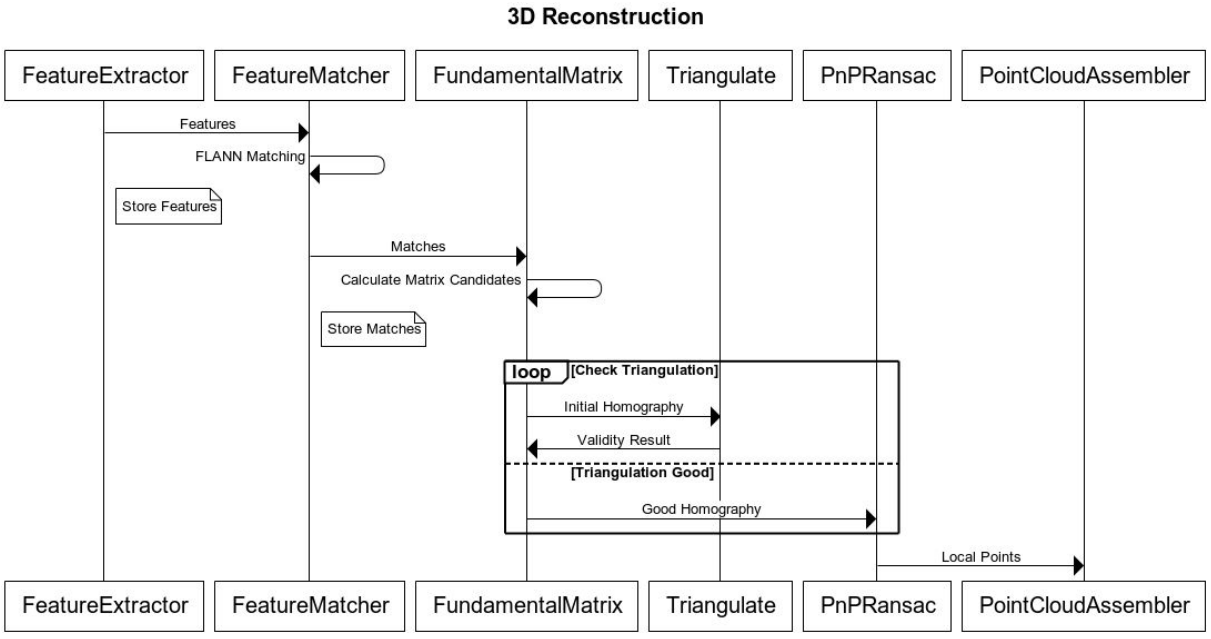

Figure 30: Sequence diagram of the 3D reconstruction DFPC.

4.1.7 DFPC: 3D Tracking

The 3D tracking DFPC is for locating an instance of a model point cloud within a scene point cloud obtained from the 3D reconstruction DFPC or from LIDAR. This DFPC is separate as it is not needed for target reconstruction, only for identification and tracking tasks that could be potentially done by other DFPCs.

DFPC Inputs :

- [Data Product] Scene point cloud from reconstruction
- [Data Product] Model point cloud file (PCD, PLY, A3D).
- [Data Product] Valid initial guess of the target pose,
- [Parameter] Additional optional parameters

DFPC Outputs:

- [Stream] Estimated pose in reference frame of target
- [Stream] Estimated orientation and matching of target with respect to model

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
  - Point normal + SHOT feature descriptors,
- Descriptor matching: to match descriptors between scene and model
  - FLANN descriptor matcher
- Correspondence grouping: to find correspondences between scene and model

- Hough voting or ICP
- Pose estimation calculation
  - Extended Kalman filter

Figure 31 shows the sequence diagram for tracking objects in a scene point cloud that has been reconstructed by visual or LIDAR means. The parameters that can be set for this DFPC include the following:

- Clustering algorithm to use
- Model uniform sampling radius
- Scene uniform sampling radius
- Reference frame radius
- 3D Descriptor radius
- Cluster size

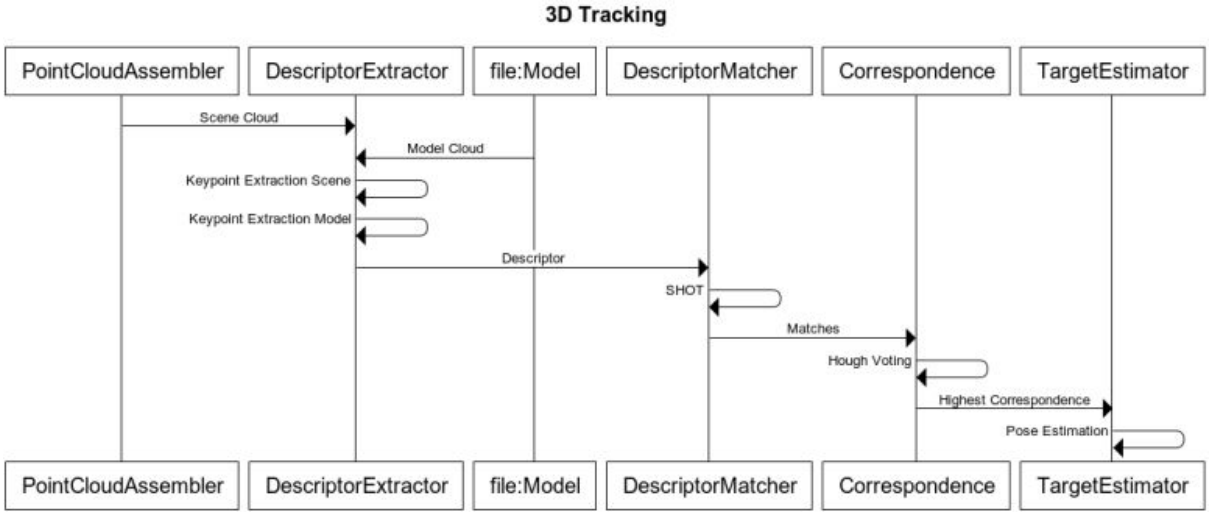

Figure 31: Sequence diagram of the model-based 3D point cloud tracker

4.1.8 DFPC : Haptic scanning

This DFPC responds to the following reference implementation scenarios

- Use Case 6: 3D reconstruction and mapping with Haptic

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 :

- End-effector position
- End-effector force measurements
- Estimated chaser pose relative to target.

DFPC Outputs:

- 3D point cloud with normal forces embedded describing

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.

4.1.8 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

RI-SRC.SR

INFUSE-SRC RI is a superset of INFUSE-RI, hence further details of interface using ESROCOS middleware with CDFF will be described during the development.

4.2 DFN

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.

4.2.1 DFN Template

This section of the document is adapted from the document [RD8] as the core DFN design is shared by the planetary and orbital track. This DFN template will be used as a guideline to design each components of DFPC.

4.2.1.1 DFN Description

We start by describing the common DFN elements.

DFN element	Remarks
Generic description	
Input(s) and Ouput(s) data	Data here refers to: • Actual data (e.g. Image16bit) • Metadata (e.g. CameraParameters, Timestamp) and are data structures necessary for the DFN to function properly.
Input Parameters	Can be provided by: • Configuration file (e.g. Threshold, Size of patch) • Output of another DFN (e.g. KF reinitialization)

Table 11: DFN template elements for interfacing

The DFN template elements related to implementation detail are listed in Table 12.

DFN element	Remark
Performance and cost estimation methods	A cost/performance estimation method which is common to all DFI of this DFN.
Diagnostic capacities	Includes: • Errors/warnings at runtime (e.g. unexpected data type, 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.
Unit test	This must be provided with the code, along with the dataset used for validation.

Table 12: DFN template elements at implementation level

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 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.

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.

An example YAML description file following the CDFF-Support specification is provided in Table 13.

File: LaserFilterDFN.yml

```
name: LaserFilter
input_ports:
   - name: scanSamples
     type: base::samples::LaseScan
     doc: samples of a laser scan
   - name: laser2BodyTf
     type: base::samples::RigidBodyState
     doc: laser to body transformation
output_ports:
   - name: filteredScans
     type: base::samples::LaserScan
     doc: filtered laser scans
```

Table 13: Template description file for setters and getters of LaserFilter DFN

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 the detail 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: 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

4.2.2.1.1 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	

4.2.2.2 DFN: Edge Detection

This DFN consists of the functions: Sobel filter, Scharr operator and Canny detector.

DFN Name	Edge detection
DFN element	**Remark**
Generic description	Extracts image edges using a given detector
Input(s) and Ouput(s) data	Input: image (type, cv::Mat) Output: edge map (type, cv::Mat)
Input Parameters	None
Performance and cost estimation methods	Detection time can be used to estimate the cost of any edge extractor. For some instances, performance evaluation methods may exist.

4.2.2.2.1 DFI: Canny Edge Detector

DFI Name	SIFT Feature Extractor
DFI element	**Remark**
Est. performance and cost	
Input Parameters	Upper and lower thresholds, aperture size
External library dependencies	OpenCV
Diagnostic capacities	TBD

Unit test	

4.2.2.3 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	

4.2.2.3.1 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	
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	TBD

4.2.2.4 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	Comparison with known feature list in image

4.2.2.4.1 DFI: ORB Feature Extractor

DFI Name	**ORB Feature Extractor**
DFI element	**Remark**
Est. performance and cost	Relatively fast compared to SIFT/SURF but slower for large image sizes
Input Parameters	Pyramid decimation ratio (scale factor) Edge threshold Number of points to produce for BRIEF Patch size used by BRIEF Number of pyramid levels (in case of ORB descriptor) number of feature levels (in the case of ORB descriptors) WTA K value (in the case of ORB descriptors)
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.5 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.
Unit test	Comparison with known matches in two images

4.2.2.5.1 DFI: FLANN Matcher

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

4.2.2.6 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

4.2.2.6.1 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

4.2.2.7 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: 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	Computation Time.
Unit test	Given a dataset check the triangulation of points

4.2.2.8.1 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	N/A

4.2.2.9 DFN: 2D-3D Motion Estimation

DFN Name	**2D-3D Motion Estimation**
DFN element	**Remarks**
Generic description	Estimates the self-motion of the camera using RANSAC given a set of triangulated points and a fundamental matrix.
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 (Pose estimation matrix)
Input Parameters	None
Performance and cost estimation methods	Complexity. Computation Time. Percentage of inliers.
Unit test	Given a dataset check the ego-motion estimate

4.2.2.9.1 DFI: PnP (Perspective from n-Points)

DFI Name	**PnP**
DFI element	**Remark**
Est. performance and cost	Dependent from the algorithm parameterisation, fast overall
Input Parameters	Solving method (e.g. EPNP, Iterative, P3P) Ransac parameters - Number of iterations - Reprojection error - Number of inliers - 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.

4.2.2.10 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	Computation Time
Unit test	N/A

4.2.2.10.1 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

4.2.2.11 DFN: 3D Keypoint Descriptor Extraction

DFN Name	3D Keypoint Descriptor Extraction
DFN element	**Remarks**
Generic description	This DFN determines keypoints, their normals, and extracts descriptors for these keypoints
Input(s) and Ouput(s) data	Input: Point cloud Output: Keypoint descriptors
Input Parameters	Model uniform sampling radius Scene uniform sampling radius Reference frame radius
Performance and cost estimation methods	Computation Time
Unit test	Given a point cloud check keypoints produce

4.2.2.11.1 DFI: SHOT 3D Keyoint Extractor

DFI Name	SHOT 3D Keyoint Extractor
DFI element	**Remark**
Est. performance and cost	Fast but dependent on size of descriptor radius
Input Parameters	3D Descriptor radius
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.12 DFN: Correspondence Grouping

DFN Name	Correspondence Grouping
DFN element	**Remarks**
Generic description	This DFN finds correspondences between two sets of point clouds
Input(s) and Ouput(s) data	Input: Keypoint descriptors for scene; Keypoint descriptors for model

	Output: Corresponding poses of model in scene
Input Parameters	Clustering algorithm to use; Cluster Size
Performance and cost estimation methods	Computation Time
Unit test	N/A

4.2.2.12.1 DFI: Hough Correspondence Grouping

DFI Name	**Hough Correspondence Grouping**
DFI element	**Remark**
Est. performance and cost	Highly computationally intensive depending on cluster size
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.13 DFN: Target Pose Estimation

DFN Name	**Target Pose Estimation**
DFN element	**Remarks**
Generic description	This DFN determines the most likely pose of the target given a set of potential poses and a movement filter
Input(s) and Ouput(s) data	Input: Potential poses Output: Estimated pose of target
Input Parameters	Sensor covariances for pose, model covariances for pose, previous inputs
Performance and cost estimation methods	Computation time
Unit test	N/A

4.2.2.13.1 DFI: Target Pose Estimator

DFI Name	**Target Pose Estimator**

DFI element	Remark
Est. performance and cost	Moderate loading
Input Parameters	None
External library dependencies	OpenCV
Diagnostic capacities	N/A

4.2.2.14 DFN: Haptic scanning

DFN Name	Target Pose Estimation
DFN element	**Remarks**
Generic description	This DFN gathers opportunistically any contact point and generates an associated mesh (point and normal)
Input(s) and Ouput(s) data	Input: Estimated pose of target Input: Current Manipulator joint state Input: Force sensors Output: Force mesh representing contact points
Input Parameters	Sensor covariances for object detection
Performance and cost estimation methods	Computation time
Unit test	Given a CAD, generate random collision paths, collision paths must match generated force mesh

4.3 Data Types

A common data type is defined to facilitate an internal and an external interface among DFNs, DFPCs and CDFF clients such as the autonomy (OG2) and the Sensor Suite (OG4). The ESROCOS data types will be used for the communication among OGs, while intermediate DFNs are free to use other data types.

The table below shows description of each data structure, with asn files name. ASN files name in bold exist in ESROCOS[2].

[2] https://github.com/ESROCOS/types-base/tree/master/asn

Data Name	Description	ASN file name
Image8UC1	Image with one channel of 8 bits values.	Image8UC1.asn
Image8UC3	Image with three channel of 8 bits values.	Image8UC3.asn
Image32FC1	Image with one channel of 32 bits values.	Image32FC1.asn
Timestamp	Time in microsecond.	**Time.asn**
3DPoint	3 points in space (x, y, z).	**Point.asn**
3DPointCloud	Array of 3DPoint.	PointCloud.asn
Pose	Position and orientation of an object.	**Pose.asn**
Transform	Define a pose transformation.	**TransformWithCovariance.asn**
CalibrationMatrix	Define the calibration matrix to a camera.	CalibrationMatrix.asn

5 Detailed Description of EGSE

This chapter describes the EGSE, which will be used to validate DFPCs presented in previous chapter.

5.1 Introduction

In chapter 4, we described the detailed architecture and design of the reference implementations, addressing the CDFF-core data fusion components (DFN) and processing that leads to data products (DFPC) of the CDFF. The Electrical Ground Support Equipment (EGSE) is used as a test platform to validate the implementation by providing an orbital simulation environment. The integrated components of the CDFF such as DFPC, DFN and related interfaces to orchestrator, Data product manager and middleware will be verified. The following section describes the hardware and software interfaces of the EGSE.

5.2 EGSE

The Electrical Ground Support Equipment (EGSE) provides a validation and test platform for the CDFF, particularly for the orbital DFPCs. For this purpose, the DLR OOS-sim facility shown in Fig. 32 will be used as our major EGSE. Moreover, datasets recorded by the OG4 Sensor suite at the GMV facility will be used for the validation of mid-range DFPCs. The main objective of the EGSE is the acquisition of a number of datasets from the facility in order to validate and test the DFPCs described in CDFF of the orbital track. Hereafter we

specify and describe the orbital EGSE design based on the document [RD5] provided by the OG6.

DLR OOS-sim facility: It consists of

- a servicer satellite mock-up
- a lightweight robotic manipulator
- a target satellite mock-up
- an environment and space lighting simulator

DLR OOS-sim Sensors:

- Stereo cameras for close-range approach
- IMU
- LIDAR System

5.2.1 Sensor Specifications

Camera: a pin-hole projective model

- Ethernet Prosilica GC1600H, a 2.0 Megapixel camera , 25 fps @ full resolution
- Sony ICX274 CCD sensor
- C-Mount lens focal length 6mm

IMU: provides orientation based on measurements of angular velocity and linear acceleration

- Xsense MTI, RS232 interface
- High update rate (120 Hz), inertial data processing at max 512 Hz
- 360º orientation referenced by gravity and Earth Magnetic Field
- Integrated 3D gyroscopes, accelerometers and magnetometers
- Angular resolution 0.05 deg
- Dynamic accuracy 2 deg RMS

LIDAR: point cloud of a target by measuring time of flight of a short pulse

- Velodyne VL-16
- 16 Channels
- 300,000 Points per Second
- 360° Horizontal FOV
- ± 15° Vertical FOV
- 100m Range

The OOS-Sim facility consists of a lightweight robotic manipulator mounted on the servicer satellite mock-up. A stereo camera system is mounted at the end effector and the potential LIDAR position is indicated on the servicer.

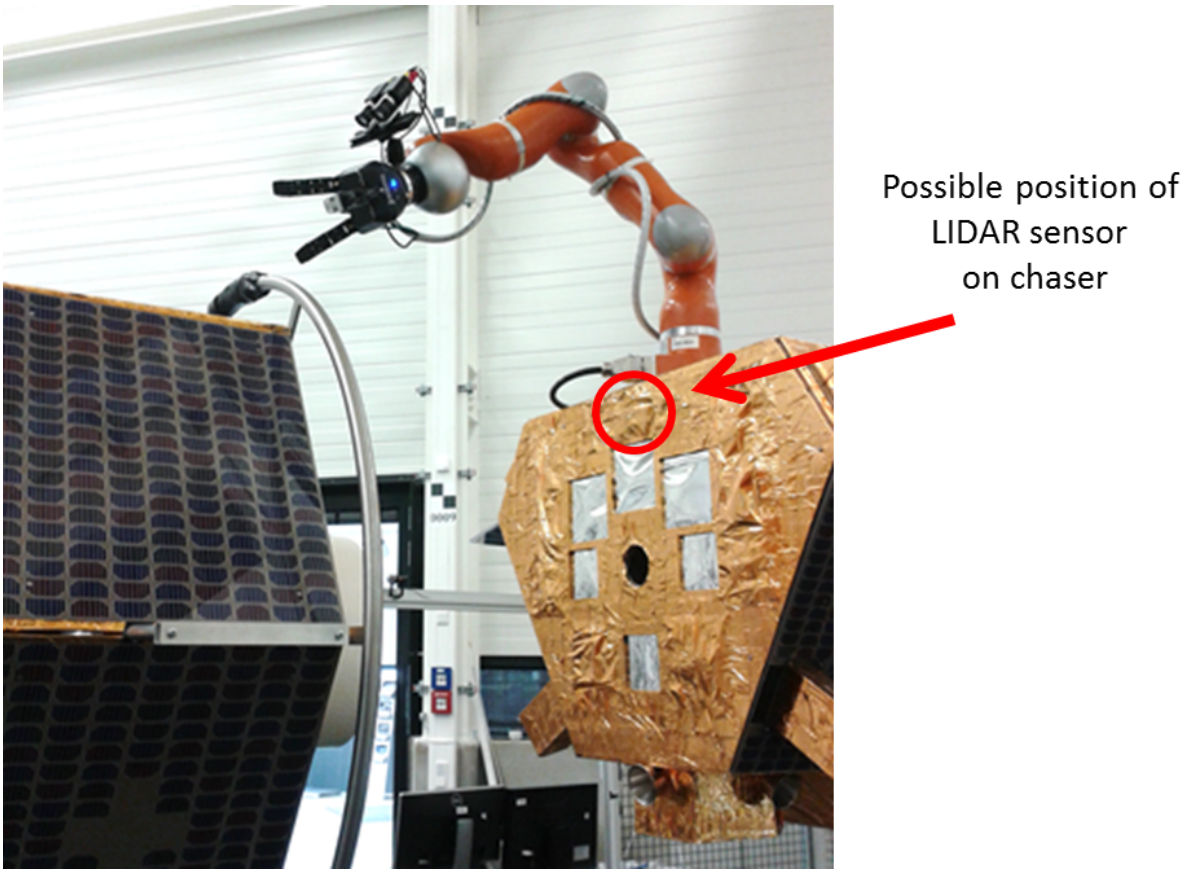

Figure 16: The OOS-sim facility.

5.2.2 Hardware and Software Interface

The hardware interface defines an electrical and a mechanical interface between the OG3 sensors and the DLR OOS-sim facility.

- The electrical and mechanical interface of the stereo camera system is already implemented.

- The mechanical interface of the OG3/InFuse LIDAR (TBC) will be designed to accommodate the sensor on the front surface of the servicer satellite mock-up. The electrical interface will be specified.

The software interface defines the data interface between the OSS-sim facility, and the InFuse CDFF and associated sensors. The facility uses an intel desktop linux computer for acquisition and processing of a sensor data. The interface of the stereo camera to the facility is already implemented.

5.2.3 Functional Interface

The functional interface is defined by the commanding trajectories and lighting condition. Commanding trajectories:

- OG3 users will be able to choose from the list of representative trajectories and repeat them at their will from the OBC which controls the facility.

Lighting conditions:

- OG6 provides possible lighting conditions to choose.

5.2.4 Operational Interface

The operational interface consists in commanding the servicer/manipulator from a set of pre-planned trajectories (trajectory library). The content of the trajectory library will be further discussed with OG6.

Trajectory library:

- a set of representative trajectories for the chaser satellite, the Light Weight Robot and the target satellite on the OOS-SIM facility will be made available for taking measurements with the chosen sensor(s).

Trajectory library viewing:

- The trajectories in the libraries will be viewable in a dedicated simulator.

6 Conclusion

In this document the detailed design of the orbital reference implementation is presented, in order to facilitate the software development. Firstly, the demonstration and validation scenarios are described by identifying various use cases for mid- and close-range rendezvous. The use cases are mainly concerned with a target object detection, reconstruction and tracking. They arise also from the fact that the associated method might exploit a 3D and/or vision senor. One one hand, a point cloud-based pose tracking exploits a LIDAR data or stereo camera to estimate relative motion between a servicer and a target satellite. On the other hand, an image-based approach to visual tracking relies on a monocular or stereo camera. The 3D reconstruction exploits stereo camera or LIDAR for modeling of the target satellite. An offline data processing recorded from an EGSE will be used to extensively study the performance of the algorithms. On the other hand, an online validation will be performed with a selected use cases to verify the integration and real time behaviour. Secondly, in order to identify and motivate demonstration and validation scenarios for the reference implementation, we addressed a general target satellite configuration with respect to rendezvous sensor, space environment and lighting in an on-orbit servicing scenarios.

The document provides the overall view of the system modeling, consisting of the sensor system, communication system, software system and actuator system to highlight the possible integration of the CDFF to other systems and the Electrical Ground Support Equipment (EGSE). The reference implementations are then described in detail, by defining the input, output and its function. Moreover, each data fusion processing compound (DFPC) is decomposed to its low-level function (DFN). The interaction of each DFN within a DFPC is illustrated with sequence diagram to show how the smallest blocks could build up the DFPC to provide data product required by the end user such as autonomy. Moreover, the decomposition of DFPC helps identify common functions shared among the DFPCs and internal behaviors. Finally, the consolidated detail design of DFPCs and DFNs are described in the last chapter of the document. Here, apart from a DFPC and DFN internal as well as external structures, a detailed interface to an external such as autonomy framework, Sensor Suite will be exposed through the orchestrator and data product manager. This chapter will be updated during the development phase.

7 Appendix

7.1 Requirements

ID	Category	Title	Description	Identified Use Case in D5.1	Requirement driven by SRC and specified in RD4
SR_DFPC_0100	DFPC	Mid-range tracking	Pose estimation accuracy w.r.t chaser - <1m - < 10 deg	Mid- and close-range visual tracking LIDAr-based tracking	SR_PerfR_A305
SR_DFPC_0101	DFPC	Close-range tracking	Pose estimation accuracy w.r.t chaser - <0.05 m - < 5 deg	Mid- and close-range visual tracking	SR_PerfR_A306
SR_DFPC_0102	DFPC	Mid-range tracking & Close-range tracking	Pose estimation update frequency >= 0.1Hz		SR_PerfR_A307
SR_DFPC_0103	DFPC	Close-range tracking	Estimation of robot relative state w.r.t		SR_PerfR_A206

			orbital object		
SR_DFPC_0104	DFPC	3D Reconstruction	Object 3D model reconstruction	3D Reconstruction	SR_PerfR_A204
SR_DFPC_0105	DFPC	Close-range detection and tracking	Object localization in robot reference frame	Mid- and close-range target detection	SR_PerfR_A202
SR_DFPC_0106	DFPC	Mid-range detection and tracking	Facilitate localization w.r.t structured and unstructured objects	Mid- and close-range target detection	SR_UserR_A102
SR_DFPC_0107	DFPC	3D Reconstruction	Support map building, reconstruction	3D Reconstruction	SR_UserR_A103

7.2 Technical Note on DFN and DFPC Specification

This section provides specification of DFN and DFPC, in order that a DFPC developer should follow this guideline as a template to realize the reference implementations.

7.2.1 Scope of the 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. Its consistency with D4.2 must be improved. It will be become the introduction of section 4 Detailed Architecture of DFPC in D5.2.

7.2.2 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.

7.2.2.1 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.

7.2.2.2 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.

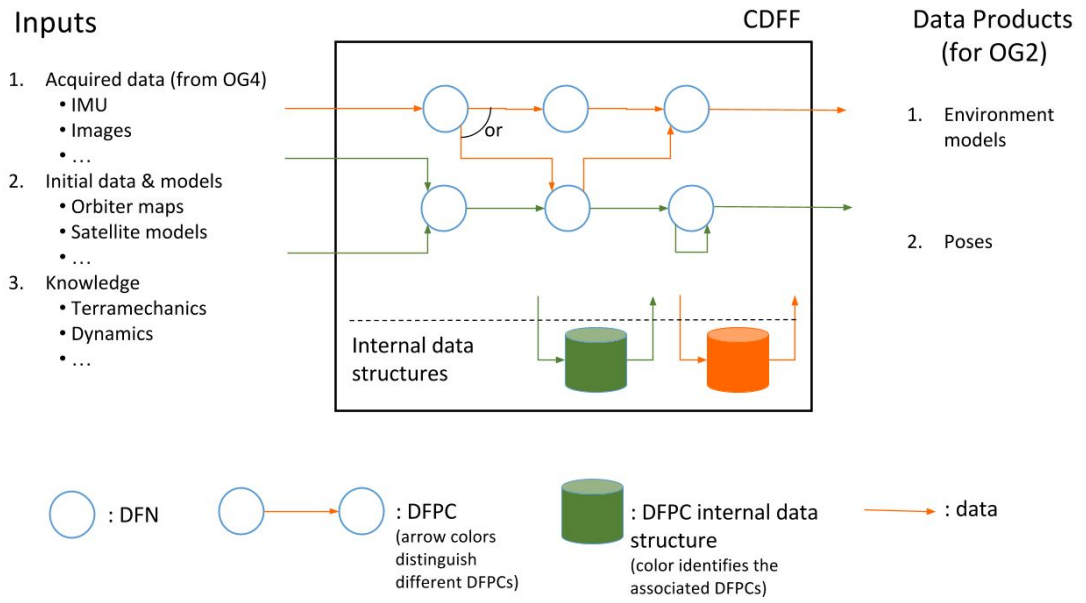

Figure 17: A simple schematic view of DFNs and DFPCs.

The figure 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)

7.2.3 DFN Template

7.2.3.1 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.

7.2.3.2 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.

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.)

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 - though not a good example actually, as the output data structure are slightly different). 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.)

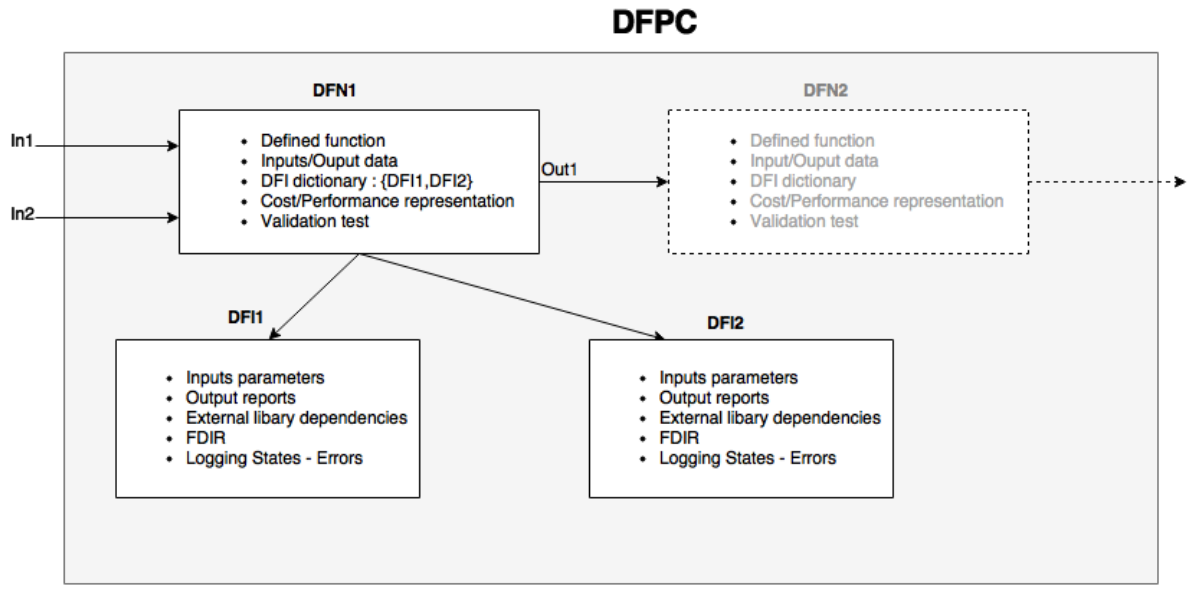

Figure 18: Example of a possible implementations of a DFN

7.2.4 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… The control flow will be achieved by the Orchestrator for implementation.

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:

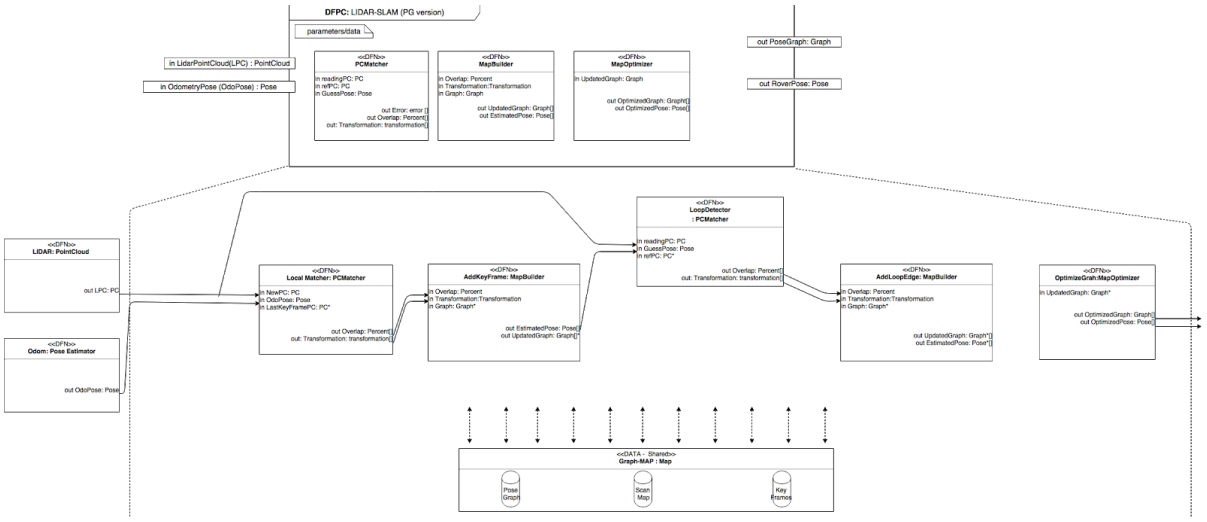

Figure 19: LIDAR-PG-SLAM

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 Detailed Interface among OGs

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,	sensor names, expected	DEM map or error state	Map produced

			update rate, resolution, area of coverage	accuracy values, Hz, pixel to cm coverage, in sq. mts		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 LocalTerrain Frame
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

getAbsolute Pose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	Produces the AbsolutePos e Pose of the BodyFrame in the AbsoluteFra me
getTargetRel ativePose	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
getTargetRel ativeVelocity	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
getModelOf Target	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values,	3D Model	This interface produces the 3D model of the target spacecraft.

				Hz		
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, ConfigurationID	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	
getStereoCa	DFPC -> OG4	Synch	NULL	N/A	Images or	

mImages					error state	
getToFPoint Cloud	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	
getLidarPoin tCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getLaserScan	DFPC -> OG4	Synch	NULL	N/A	Planar 2D PC or error state	
getRadarSca n	DFPC -> OG4	Synch	NULL	N/A	2D or 3D PC or error state	
getHRCamer aImage	DFPC -> OG4	Synch	NULL	N/A	Image or error state	
getTIRCamer aImage	DFPC -> OG4	Synch	NULL	N/A	Image or error state	
getForceTor que	DFPC -> OG4	Synch	NULL	N/A	Wrench data or error state	
getStructure dLightPointC loud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getStarTrack erOrientatio n	DFPC -> OG4	Synch	NULL	N/A	Orientation data or error state	