ABSTRACT (15 LINES OR 150 WORDS)

Data fusion algorithms provide a system with the capacity to combine the data from different sensors and metadata (e.g. timestamps, geometric models) into symbolic representations like maps or position estimations. Software that implements these algorithms needs to provide a solution to the challenge of awareness using multiple sensors faced by robots and autonomous systems as well as the means to access and store this data for future uses. The software community in space robotics lacks a common framework to make developing, reusing and comparing data fusion solutions easier. InFuse provides that framework, and not only a set of data fusion solutions based on state-of-the-art algorithms, but also performance metrics to qualify algorithms. InFuse is developed by six industrial and academic partners working in the space sector, under the supervision of several European space agencies (ASI, CDTI, CNES, DLR, ESA, UK Space). This paper describes the architecture and the methods of the robot perception and localization framework InFuse Operational Grant 3 (OG3), in order to provide the Space Robotics community with a Common Data Fusion Framework (CDFF).

1. INTRODUCTION

In the context of artificial intelligence applied to space robotics, common evaluation and deployment frameworks are crucial for efficiently developing reliable robotic solutions. Qualification of software for space is indeed highly demanding. Methods and tools that make it easier to develop software for space and evaluate it as early as possible are highly valuable. This is the motivation of the European Commission’s Horizon 2020 Strategic Research Cluster in Space Robotics, which comprises several projects to develop open, modular and reusable solutions in the domains of Robotic Control Operating Systems (RCOS (OG1) [1][2], Autonomy (OG2) [3], Perception and Localization (OG3) [4], and Sensing (OG4) [5].

Our CDFF framework has been designed to handle the challenges that developing and integrating sensor data fusion solutions pose in a space context: (1) it will be compliant with the requirements that space-grade software impose at interface level, and partially conforms to lifecycle and coding guidelines based on ECSS-E-ST-40C standards (European Cooperation for Space Standardization), (2) it will be easily deployable in ESROCS [1][2] that uses TASTE modeling [7], (3) it will be experimentally validated by our porting of selective Data Fusion Nodes (DFNs) and Processing Compounds (DFPCs) to RTEMs, and (4) it will be validated in a partially hardware-accelerated setup, with some data fusion processing taking place on a FPGA coprocessor (e.g. Xilinx Zynq SoC).

Although our framework is designed targeting space robotics, one of our design guidelines is also to facilitate its integration with RCOs more generally than ESROCS, in particular with ROCK and the widely-used ROS. Furthermore, it comes with tools that make it possible to evaluate perception and localization modules with framework-independent logged data and without using any RCOS. These two points make it RCOs-independent and therefore suitable for many terrestrial applications with minimal modification. Furthermore, the CDFF after its initial development will
be available for the general public as open source project. Thus, the potential users of such. Section 2 introduces the architecture design of the framework, here the main conceptual components as well as its correspondent software modules are presented. Section 3 presents the lists of Data Fusion Processing Compounds (DFPC) that will be available in InFuse along with an example for one of them. Section 4 describes the analog missions that will be executed in order to demonstrate the utility of the framework. Section 5 summarizes of the conclusions.

2. ARCHITECTURE OF THE CDFF

The Data Fusion Node (DFN) concept is in the core of the architecture. DFNs are the processing units for data fusion. DFNs are designed to be reusable and atomic and therefore might require for a data product construction the connection and coordination with other DFNs. Data Fusion Processing Compounds (DFPCs) are specialized structures of DFNs which are internally controlled and can include local storage of spatio-temporal data.

The whole framework activity is controlled by the Orchestrator with capability to activate or deactivate the different DFPCs available. The last main component of the architecture is the Central Data Product Manager which can store and access under request from the orchestrator data from the persistent memory. Fig. 1 presents briefly the architecture.

Figure 1. The Orchestrator manages the queries to the Central Data Product Manager, the activation of different Data Fusion Processing Compounds (DFPCs) and the operating modes of OG4 to satisfy the requests from OG2.

Our framework is made up of three component groups: (1) CDFF-Core comprises the DFNs, (2) CDFF-Support provides the tools to connect these into DFPCs, orchestrate their operation, and manage their data products, and (3) CDFF-Dev is an environment for developing and evaluating data fusion solutions, independently of the target robotic system and of its RCOS.

2.1. Data fusion libraries

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 two control interfaces: (1) configure() sets all the configuration parameters of the DFN while (2) process() calls library functions to compute the outputs of the DFN.

The interface of the DFN has been designed to be compatible with the RCOS ESROCOS. Nevertheless, in the cases developed DFNs will not be deployed as single ESROCOS modules.

A DFN Template to incorporate the knowledge regarding each DFN has been designed. The templates allow for an efficient management of all existing DFN. These templates are available on the deliverables corresponding to the Orbital and Planetary Track Test Plans [1, 2]. The DFN Template incorporates the following information: (1) Generic Description, (2) Input(s) and Output(s) data, (3) Input(s) Parameters, (4) Estimated performance and cost, (5) External library dependencies, (6) Diagnostic capacities and (7) Unit test.

2.2. Data fusion solutions

The CDFF Support consists of a set of components designed to run on the target system along with the DFNs. These components provide supporting tools to use multiple DFNs together and in a coordinated fashion.

Furthermore CDFF-Support provides the Data Products Manager (DPM) which stores a consistent representation of the environment, a history of acquired pre-processed sensor data, estimated poses, and a selection of the generated fused data products, to deliver them under request to OG2. The three main components
of CDFF Support are the DFPCs, the Orchestrator, and the DPM.

Each DFPC is characterized by its functionality, the datastreams that it receives and produces, the operations that can execute on demand, the DFNs it uses and how these are configured and setup. A DFPC description template to include the information regarding a DFPC. The fields of a DFPC template are: (1) Data Flow description, (2) Data Product Management, (3) Control description. An example is presented in Table 1.

<table>
<thead>
<tr>
<th>DFPC: Lidar Map-based Localisation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data flow description</strong></td>
</tr>
<tr>
<td>- Inputs: LIDAR point cloud, Pose estimate</td>
</tr>
<tr>
<td>- Outputs: Pose estimate</td>
</tr>
<tr>
<td>- DFNs: PCMatcher, PoseEstimator</td>
</tr>
<tr>
<td><strong>Data Product Management</strong></td>
</tr>
<tr>
<td>- Graph-Map: Pose, Graph, Scan Map, Key Frames</td>
</tr>
<tr>
<td><strong>Control Description</strong></td>
</tr>
<tr>
<td>1. getLocalMap</td>
</tr>
<tr>
<td>2. PCMatcher.doICP</td>
</tr>
<tr>
<td>3. PoseEstimator.estimatePose()</td>
</tr>
</tbody>
</table>

The orchestrator has the main task of receiving queries from the Autonomy Framework (OG2), activate and provide the fused data products to OG2. It acts as the central coordinator in the target system to control the activation states of DFPCs. The orchestrator has the following functions: (1) Interface between OG2-OG3, (2) Translate the perception and localization data into the format required by OG2, (3) Interface with OG4 Instrument Control Unit (ICU) to configure a limited set of operational modes, (4) Interface with the Data Product Management (DPM) tool and provide mechanisms for querying fused data products and (5) Activation and deactivation of DFPCs according to data product requests and operational modes of the sensors. Does not interfere within the DFPC decision making.

The role of the DPM is to handle the selection, structuring and storage of all the data processed or produced by the CDFF that may be re-used, either internally by OG3 processes or to satisfy OG2 requests. Additionally, it is the interface through which robots expose and retrieve the CDFF data products in multi-robot scenarios, and also the interface through which ground operators can access the CDFF data products. The DPM can be seen a robotics-dedicated Geographic Information System (GIS). With respect to the activated DFNs and DFPCs in the CDFF, the DPM processes the data insertion requests. Internally, it manages all the spatial related data by implementing insertion, deletion or update functions, aiming at satisfying future needs for data products and storage constraints.

### 2.3. Development toolkit

CDFF-Dev provides tools for testing and prototyping data fusion solutions independent of the target RCOS. That includes a tool to replay, visualize and analyze logs in Python, and a framework to develop and test new DFNs that use signal processing or machine learning algorithms, in particular, for data filtering and outlier detection. None of these tools are deployed on the target system. In addition, code generators for DFN and DFPC scaffolds and corresponding Python bindings are provided. They are tools for developers of data fusion solutions.

The first step to evaluate or implement a DFN or DFPC using CDFF-Dev, is to generate a DFN or DFPC description file from the DFN or DFPC template, described in Sections 2.1 and 2.2. From the description file, the code of the interface is autogenerated. As an
example, Figure 3 a) shows the DFN description file in YAML format and Figure 2 b) displays the artifacts that are created by the DFN code generator from this DFN description file. DFPC description files can include some additional information, for example, internal connections between DFNs but the DFPC code generator works exactly in the same way. The Python bindings for DFNs and DFPCs are generated as Cython files that are compiled to CPython extensions. Bindings for InFuse data types are already provided to make the prototyping of DFNs and DFPCs more convenient in Python. Bindings for other data types must be implemented manually but could be generated automatically from an abstract syntax tree (AST) of the C/C++ header of the type. A prototype for this has been developed.¹

```python
name: LaserFilter
implementations:
- NoiseFilter
- BoxFilter
input_ports:
- name: scanSamples
type: LaserScan
doc: samples of a laser scan
- name: laser2BodyTf
type: RigidBodyState
doc: laser frame to body frame
output_ports:
- name: filteredScans
type: LaserScan
doc: filtered laser scans
```

(a)

DFNs and DFPCs interfaces are kept as minimal as possible to ease integration to any target RCOS. The only dependencies are common base classes for DFNs or DFPCs and the types that are used as inputs and outputs. This also simplifies the integration in the log replay tool of CDFF-Dev.

Testing DFNs or DFPCs offline with log data is possible with the provided Python bindings. Logs are replayed with a data flow control module that emulates the communication layer of an RCOS and a log player that replays logged data chronologically. Two essential elements are needed for a user to be able to replay data logs from a desired RCOS: (1) a conversion from the data log format used by the RCOS to an intermediate format that is used by InFuse, and (2) a data-type conversion from the RCOS to InFuse data types.

MessagePack² is the intermediate log file format that can be handled by CDFF-Dev. An example of the intermediate log format is shown in Fig. 4. A converter from ROCK’s log format pocolog is already available, and a ROCK base-types to InFuse data types is under development. The converters will be stored in open

¹ https://github.com/AlexanderFabisch/cythonwrapper
² https://msgpack.org

Figure 3. (a) Example of a DFN description file based on YAML. (b) The DFN code generator creates an abstract C++ base class that defines the interface of the DFN, templates for concrete implementations and Python bindings for these implementations.
repositories to ease its reuse between developers. The intermediate log format can be loaded as InFuse types (C++) wrapped in Python. These will be given as input to the Python interface of DFNs or DFPCs.

To replay log files, the user would provide the path to the logged data. While replaying log data, an orchestrator can suggest which DFPCs should be activated or deactivated. It will analyze incoming log data and the output of each active DFPC. The orchestrator in the final deployed setting receives requests from the Autonomy Framework (see Fig 1, 2). When testing on CDFF-Dev the user, or a script, generates this requests. The Orchestrator then activates the DFPCs that produce the requested data products or triggers the operations that generate them. Additionally, the orchestrator can store the data in the Central Data Product Manager for later access.

The current state of the system is stored in an EnviRe graph [6] while replaying log data. EnviRe provides various tools to store and handle environment representations. This data structure can be displayed with the EnviRe visualizer. Objects for which a visualization has been implemented can be displayed with the EnviRe visualizer, for example, point clouds, laser scans, or poses.

For the development, analysis and comparison of new DFNs that use signal processing or machine learning algorithms, the framework pySPACE [7] is integrated in CDFF-dev. Log files can be annotated while replaying log files and converted to a format that can be used by pySPACE. pySPACE provides numerous algorithms for signal processing and machine learning. They can easily compared with respect to various performance metric. The comparison can be run in parallel on a cluster.

```python
{'/component.port':
    [{'# sample 1
        'sourceFrame': 'A',
        'targetFrame': 'B',
        'timestamp':
            {'microseconds':0},
        'pos': [0, 0, 0],
        'cov_position': [ ... ],
    ...
    },
    {'# sample 2 ... }, ...
},
'/component.port.meta':
    [  
        'timestamps': [ 1, 2, ... ],
        'type': 'RigidBodyState'
    ],
    ...
}
```

Figure 4. Example of the intermediate log format in Python syntax

2.4. Integration of data fusion solution in RCOsEs

As part of our will to address a large community of users and ease the technological transfer from R&D studies to space products, the CDFF offers a convenient way to integrate DFN/DFPC into major open source robotics middlewares namely ROS, ROCK, GenoM3 [MAL10] and YARP [PAI15]. The core of the proposed approach is to use ASN.1 to specify data structures and binary serialisation mechanisms implemented in ASN1SCC (An open source ASN.1 compiler for embedded systems [MAM11]) to exchange data between components and also across RCOS.

To insure the compatibility between RCOS and programming languages like C++, python, javascript, ..., a common and basic ROS message (asn1_bitstream.msg) has been defined to transport the serialised ASN.1 data structures. Almost all RCOS provide some compatibility with ROS messages and communication protocol and should be able to use the proposed message. The message is structured as follow:

```python
std_msgs/Header header
# Message type
string type
# Serialization method : 0 (UPER)
uint8 sermethod
# Serialised data
uint8[] buf
```

where the field buf contains serialised data.

The Unaligned Packed Encoding Rules (UPER) are used by default for an optimal compactness and encoding efficiency with low memory and CPU footprints. However, Basic Encoding Rules (BER) or (XML Encoding Rules) XER could be used to facilitate communications with different programming languages. For instance, we are using BER to communicate with web applications through YARP.

Figure 5. CDFF integration principle in an heterogeneous environment of RCOS. A simple ROS message is defined to transport serialized ASN.1 data structures.

3 http://envire.github.io
4 https://pyspace.github.io/pyspace
This has been applied to GenoM3 with ROS and YARP, and shown all the expected benefits. Moreover, the use of YARP allows to go even further towards an optimal integration, as it allows local communication; in that particular case ASN.1 data structures are exchanged without any serialisation. The key advantages of the proposed method are manifold: clear data type / interface management with ASN.1, smooth integration effort in RCOS with ASN.1, enable inter RCOS communications, favor the separation between the RCOS and algorithms (this is further true when GenoM3 is used).

3. DATA FUSION METHODS

# List of the DFPCs to be implemented
The InFuse framework will be demonstrated by the development of the following DFPCs:
3D Environment Reconstruction: images collected by a stereo or mono camera are projected in 3d coordinates and merged together by estimation of an appropriate transform;
Short and Medium Range Object Detection: 3d features are extracted from the environmental 3d map and a 3d model of the object, their matching identifies the object and its position;
Short and Medium Range Object Tracking: a Kalman filter improved the estimated pose computed by the object detection DFPC;

<table>
<thead>
<tr>
<th>Planetary Track DFPCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAAS Visual odometry</td>
</tr>
<tr>
<td>MAG/CNES Visual odometry</td>
</tr>
<tr>
<td>Absolute localization</td>
</tr>
<tr>
<td>DEM Building</td>
</tr>
<tr>
<td>Lidar SLAM</td>
</tr>
<tr>
<td>Lidar-based localisation</td>
</tr>
<tr>
<td>Visual SLAM</td>
</tr>
<tr>
<td>Visual Map-based Localisation</td>
</tr>
<tr>
<td>Long-range Tracking</td>
</tr>
<tr>
<td>Mid-range 3D Model Detection</td>
</tr>
<tr>
<td>Mid-range 3D Model Tracking</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Orbital Track DFPCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1 DFPC: Far-range Object Tracking 36</td>
</tr>
<tr>
<td>4.1.2 DFPC: Mid- and Close-range Target Detection 39</td>
</tr>
<tr>
<td>4.1.3 DFPC: Mid- and Close-range Target Tracking 42</td>
</tr>
<tr>
<td>4.1.4 DFPC: LIDAR-based Tracking of a Target 46</td>
</tr>
<tr>
<td>4.1.5 DFPC: Mid- and Close-range Visual Tracking of a Target 50</td>
</tr>
<tr>
<td>4.1.6 DFPC: 3D Reconstruction 55</td>
</tr>
<tr>
<td>4.1.7 DFPC: 3D Tracking</td>
</tr>
</tbody>
</table>

# Detailed example of a DFPC including the information that is contained in the description file: Connections, DFNs and how they connect, services that provides, services that can require...
# A sequence or activity diagram of one data fusion solution, including orchestrator and DPM

4. DEMONSTRATION SCENARIOS
The CDFF perception and localization solutions will be demonstrated in dedicated facilities as well as in space analog scenarios, designed to validate and verify them under realistic conditions. Two types of demonstration scenarios are planned: orbital and planetary.

### 4.1. Orbital Track

For the orbital scenarios, the integrated software will be deployed on specialized test platforms for each rendezvous/on-orbit servicing and planetary exploration. It consists in three main DFPCs: detection, tracking, and reconstruction at various ranges. These DFPCs rely on the data obtained mainly from a camera, a LIDAR, IMU, or combination of them. In order to validate the performance, we simulate motion trajectories of various type (linear, spinning, tumbling target motion) under different space lighting conditions (nominal, under-illuminated, ver-illuminated). These conditions depend on the direction of the sun with respect to the sensor main axis using on-ground simulation facility [6], shown in Fig. XX. The rendezvous and on-orbit servicing simulator consists of a mock-up including a servicer and a target satellite, mounted on a six degrees of freedom Kuka robots, and a sun simulator. A robotic arm is also mounted on the servicer satellite for on-orbit servicing tasks such as capturing the target and refueling. The arm incorporates stereo cameras that are used for the close-range DFPC. Fig.YY shows an exemplar camera image expected in space during on-orbit servicing.

**Fig.YY:** A camera image simulating on-orbit scenario with the Earth and deep space in background to the target.

**Fig. XX:** Ground test facility of DLR for close-range approach and visual servoing.

### 4.2. Planetary Track

The CDFF will provide state-of-the-art algorithms to implement necessary perception, localization and navigation functions for planetary exploration rovers. Four functional use cases will sustain the development of algorithms in the project: long traverse localization, long traverse navigation, rendez-vous and return to base. Each use case involves a limited set of key functions that will be evaluated.

The planetary scenario focuses on localization and mapping within planetary environments. The sensors used include stereo vision, LIDAR, and inertial measurement. Navigation over long distances is enabled with DFPCs for localization and for production of a Digital Elevation Map over long distances (~1km), guidance to a defined objective point on the map and rendezvous with a target there, and return-to-base functionality once the above objectives are met. In addition, DFPCs provide the capability to build 3D point cloud environment models incrementally through structure-from-motion and SLAM methods.

Demonstrations of planetary scenarios are planned at CNES and DLR facilities with accurate ground truth of the terrain and the robot, as well as on the desert of Morocco. These will involve 5 different rovers from three different institutions, CNRS/LAAS, DLR and DFKI.
5. CONCLUSION

Write conclusion

Add appendices if any. Appendices appear before the acknowledgment.

Acknowledgement. The acknowledgement to sponsor should be put here in footnotesize font (9pt).

REFERENCES


