EnviRe - Environment Representation for Long-term Autonomy

Javier Hidalgo-Carrió, Sascha Arnold, Arne Böckmann, Anna Born, Raúl Domínguez,
Daniel Hennes, Christoph Hertzberg, Janosch Machowinski, Jakob Schwendner, Yong-Ho Yoo and Frank Kirchner[1]

Abstract— **Data representation is a key element for robots to navigate and perform autonomous tasks in unstructured environments. This is due to autonomous tasks which call for an environment model suitable for systems that might operate during long periods of time. This work presents *EnviRe*, an environment representation model that facilitates long-term tasks for autonomous systems in real-world scenarios. The EnviRe model is a strongly connected directed graph which allows sensor data acquisition, data processing, reasoning and operations among different data formats in order to accomplish the navigation and planning demands.**

I. INTRODUCTION

Data representation should be compact, natural, and maintainable as well as rich enough to express the knowledge needed to solve the problem at hand. The underlying model should fit the domain being represented. It should enable efficient computation and recover data from robot past experiences, while being flexible enough to represent various tasks. Data representation could be also identified not only to provide sufficient means to perform autonomous tasks or to produce data, but also as the capability to correctly relate the result at different levels during the reasoning process (i.e. spatial, temporal, semantic).

In robotics, data fusion has the role to link sensor data with semantic, spatial and/or temporal relations. Planning includes relational hypotheses among such allowing some reasoning to operate over the environment. Data fusion and planning holds for autonomous systems where robots revolve around a set of object models and perceptual structures. Traditionally, these two tasks are separated from the model representation by only sharing a minimal set of information. Furthermore, a robot internal simulation concept enables robot execution and re-planning of tasks while doing operations. It can be used to guarantee safer and adaptive autonomy by providing a better testbed of the system in the current environment. The internal simulation can provide the means to adapt models and policies by adjusting the simulated (i.e. expected) behavior with the perceived one.

Long-term autonomous systems would also need to store and retrieve environment models. Databases are usually doing the job. However, the accessing time could not meet the perceptual and computational demands when working in dynamic environments. The database can be moved towards a graph oriented concept, where nodes represent entries in the database, i.e., data, and edges represent relations, i.e., how data is connected. The graph warranties real-time execution, search algorithms and efficient access in memory while a subset of the environment can be saved in databases.

Our aim with environment representation is to provide the means towards homogenization between knowledge representations from AI and perceptual models from robotics together with reasoning algorithms, i.e., data fusion, planning and navigation.

All authors are affiliated with the DFKI - Robotics Innovation Center, Robert-Hooke-Str. 1, 28359 Bremen, Germany
{name}.{surname}@dfki.de
[1]Department of Mathematics and Informatics, University of Bremen, 28359 Bremen, Germany

II. RELATED WORK

Concerning perception several libraries are available in order to perceive the environment. OpenCV [1], Point Cloud Library (PCL) [2] and octomap [3] are very well known examples in the robotic community. Those libraries have their strength of doing what they are designed for (e.g. in the case of PCL accurately manipulating and processing pointclouds and its derivative products). However, they lack of a model representation of the world. Envire is designed for filling this gap leaving the details of manipulating specific products to a dedicated library.

In the planning field, Open Motion Planning Library (OMPL) [4] is an open-source motion planning library that primarily implements randomized motion planners. MoveIt! [5] integrates directly with OMPL which has no concept of a robot model. MoveIt! extends OMPL for a ready to use representation of a robot model with maps (e.g. octomap). However, the *Move Group* interface lacks for an internal generic model representation of the world and is highly couple with a specific framework (e.g. ROS). We believe that an environment representation requires special attention to model in an extensible and generic manner at the same time that being domain specific.

Environment representations have been thoroughly studied in previous research. An architecture composed with databases for object was first introduced in [6]. The work describes the terrain representation for Autonomous Land Vehicles (ALV) in order to perform multiple inference processes. Though the work was done at the time when no vision systems could interpret general natural scenes rather than restrictive visual landmarks, the described modelling capabilities and a general object schema still apply to autonomous robots. The proposed architecture consisted of a Short Term Memory (STM) and Long Term Memory (LTM) datasets to store the necessary information.

Fayek *et al* [7] describes a method to acquire and exploit domain-knowledge using hypegraphs. His work applies to outdoor terrain navigation and mission planning. The results accommodate incremental model building and fusing information of different types which are typical required for real-world planning. Similarly, a predecessor of EnviRe was first implemented for localization and mapping. EnviRe was first model as a tree structure containing objects and operators. Global localization was accomplished by reasoning feet position information with the world representation (i.e. Embodied Localization and Mapping [8], [9]). The system was capable of performing autonomous exploration without a priori knowledge of the surroundings [10].

III. ENVIRONMENT REPRESENTATION

EnviRe essentially provides a strongly connected directed graph with spatial-temporal relations among nodes. Each node is able to store environmental objects such as sensory data (e.g. point clouds, RGB and depth images), local maps (e.g. elevation maps, octomaps, multi-level-surface maps, geometric maps), semantic information (e.g. features, segmented objects) which are either represented directly using third-party libraries (e.g. PCL, OpenCV) via lightweight wrappers or using dedicated EnviRe types. A main motivation is to ease the interaction of different algorithms by working on shared data structures which are required for various autonomous tasks, especially when information from multiple sensors is to be merged. This is also important when robots perform complex mission scenarios where different subsystems intervene (i.e. perception, planning, internal simulation, telemetry, ground control operation). Sharing of data structures happens with near-zero overhead, assuming that the interfaces of the involved algorithms are sufficiently compatible. EnviRe can be serialized for logging/replaying and communication between different hosts, and they can be visualized using the visualization library Vizkit3D[2].

[2]Visualization Kit 3D depends on Qt and OpenSceneGraph, [Online] Available: https://github.com/rock-core/gui-vizkit3d

EnviRe is designed to represent data for long-term autonomous systems. EnviRe is a model of the world defined by several different objects and their relations to each others (see Fig. 1). EnviRe provides (1) multiple representation models and sensor data types (2) a common structure for the data fusion: reducing the integration efforts of different techniques and easing the comparison from various approaches. Some aspects of EnviRe are:

1) The environment model is a strongly connected directed graph with nodes and edges.
2) Objects are data products stored in the nodes. Directly generated by any sensor or data fusion algorithm (e.g. maps, point clouds, robot poses, vision features, descriptors, meshes, physical objects, semantic).
3) Relations are spatial-temporal associations among data products in the graph edges.
4) Operators are algorithms than might change and manipulate the EnviRe graph. Two types of operators are currently considered:
    a) Object operators, algorithms that create, modify, delete data products (i.e. converting a point cloud into an octomap).
    b) Relation operators, algorithms that manipulate the graph structure itself (i.e. optimization algorithms backends).

Fig. 2 shows a simple EnviRe graph initialization. The EnviRe graph implements the STM dataset described in [6] and combines with the incremental model building similar to [7] in a strongly connected digraph. The implementation of hyperedges and level of abstraction is still an open discussion. The LTM dataset is adapted and implemented in EnviRe as a spatial database using Geographic Network Models (GNM) and PostGIS (see section III-A). Fig. 3 depicts a diagram showing the graph as a central structure which data and graph topology might be exploited by different subsystems. Some of those subsystem and their interaction with EnviRe are described in the following.

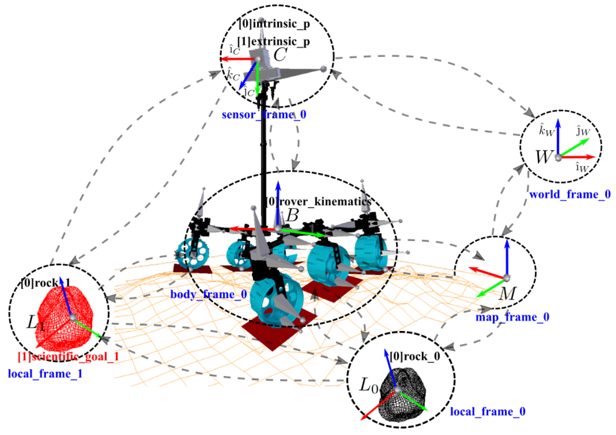

Fig. 1: Schematic representation of EnviRe into a planetary exploration scenario. Perceived objects, maps, semantic and relationships are stored using a common environment model.

```
#include <envire_core/all>

/** Create graph **/
envire::core::EnvireGraph g;

/** Frame Node **/
envire::core::FrameId frame = "Jupiter";
g.addFrame(frame);

/** Insert std::string item to node **/
const std::string text("Good news everyone!");
envire::core::Item<std::string>::Ptr item(new envire::core::Item<std::string>(
        text));
g.addItemToFrame(frame, item);

/** Remove std::string item from node **/
using Iterator = envire::core::EnvireGraph::ItemIterator<envire::core::Item<std::
        string>>;
Iterator it = g.getItem<envire::core::Item<std::string>>(frame, 0);
g.removeItemFromFrame(frame, it);

/** Get all available std::string items from frame **/
Iterator begin, end;
boost::tie(begin, end) = g.getItems<envire::core::Item<std::string>>(frame);
/** It is empty: begin == end **/
```

Fig. 2: Basic example of creating an EnviRe graph model and inserting/removing std::string items.

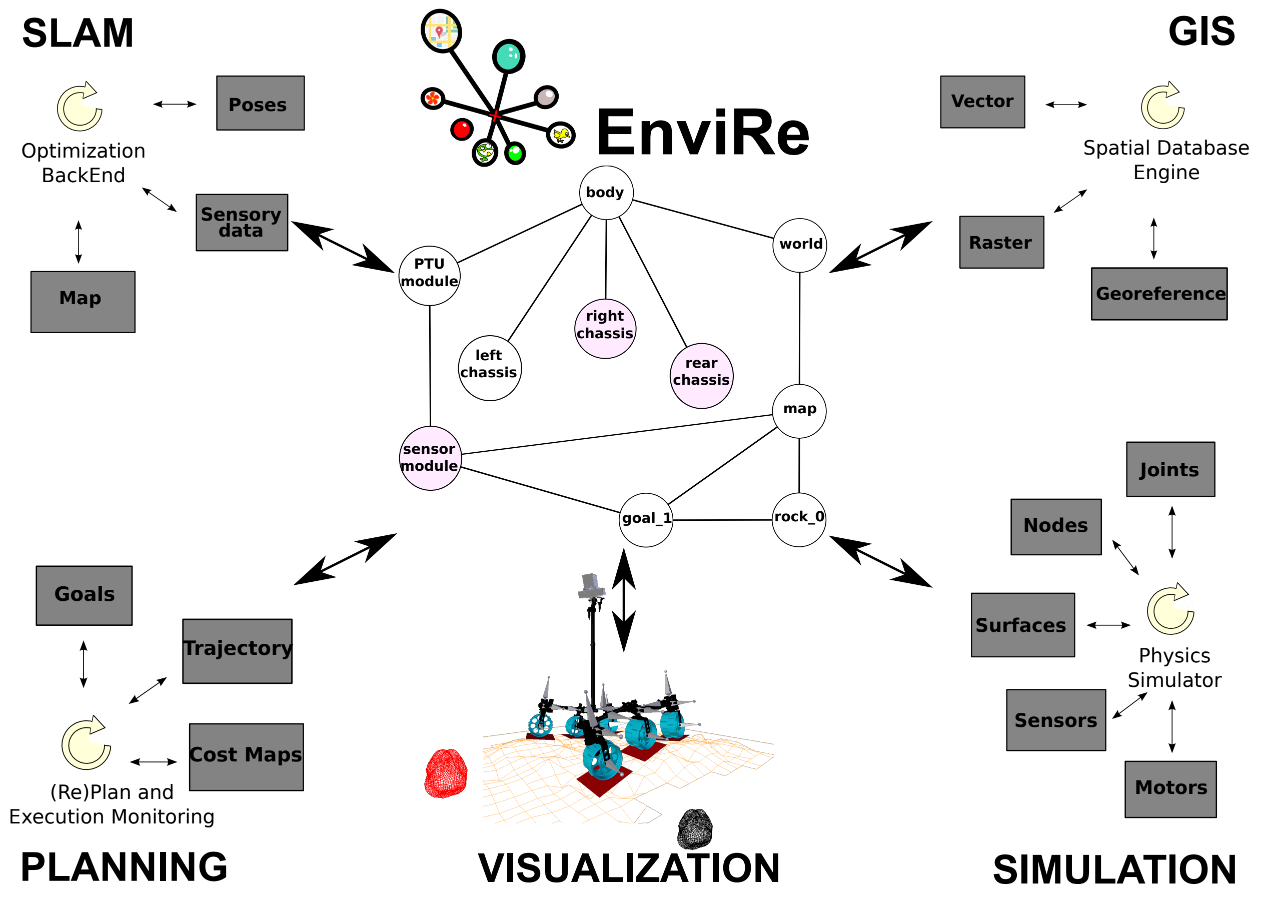

Fig. 3: Envire is used as a common environment representation towards long-term autonomous systems performing SLAM, planning and internal simulation. The concept facilitates the integration among tasks and easily enable human-in-the-loop intervention.

```
#pragma once

/** Envire **/
#include <envire_core/items/Item.hpp>
#include <envire_core/plugin/Plugin.hpp>

/** PCL **/
#include <pcl/PCLPointCloud2.h>

namespace envire { namespace pcl
{
    class PointCloud : public envire::core::Item< ::pcl::PCLPointCloud2 >
    {
        ENVIRE_PLUGIN_HEADER(envire::pcl::PointCloud)
    };
}}
```

Fig. 4: Integration of PCL PointCloud2 type into the Item node of an EnviRe graph. The Item class gives timestamp, frame name and a unique identifier to the point cloud element. The plugin mechanism is based on the class_loader library[6].

A. *EnviRe with Localization and Mapping*

Localization and mapping requires the representation of robot's surrounding, normally in form of maps (see Fig. 6). EnviRe offers an interface to integrate maps into the environment. Maps can be either implemented by using the offered map library or integrated from an already existing one via a wrapper interface. Envire map library adapts the IEEE Standard [11] for map data to 3D environments. The library allows to represent metric as well as geometric and topological maps. Besides the maps, it contains a collection of operators to merge two local maps, to project the data into a map or to convert one map type into another. Besides the Envire map library, common maps, such as multi-level-surface, octomap and traversability maps, are already integrated in EnviRe. Fig. 4 and 5 shows the code for integrating PCL and octomap datatypes respectively.

By providing the spatial and temporal relationship EnviRe is able to support a SLAM front-end to perform data association and loop closing. It can be used to determine closest and potentially overlapping features for revisited objects.

The EnviRe graph can be serialized and deserialized using boost serialization as back-end. This allows to store and retrieve the state of the world at a giving time. Loading an existing state allows the mapping and localization to continue at a certain point or to start with a-priori map. It allows to share and synchronize environment information among processes within the same and different machines. Robots could map together an environment or send part of it to an external observer. The information can be stored

[6]Class Loader library [Online] Available: http://wiki.ros.org/class_loader

```
#pragma once

/** Envire **/
#include <envire_core/items/Item.hpp>
#include <envire_core/plugin/Plugin.hpp>

/** Octomap **/
#include <octomap/AbstractOcTree.h>

/** Boost */
#include <boost/shared_ptr.hpp>

namespace envire { namespace octomap
{
    typedef boost::shared_ptr< ::octomap::AbstractOcTree > AbstractOcTreePtr;

    class OcTree : public envire::core::Item< AbstractOcTreePtr >
    {
        ENVIRE_PLUGIN_HEADER( envire::octomap::OcTree )
    };
}}
```

Fig. 5: Integration of octomap OcTree type into the Item node of an EnviRe graph

in a spatial database using the Geographical Information Systems (GIS) extension. A GNM from GDAL is chosen in order to transfer the graph structure into a network topology. The information is accessible using GIS-browsers or directly query information. A conversion between raster and grid maps is available as well as features to vector formats. Dedicated converters need to be written per each type using the GDAL [3] library (see Fig. 3).

B. *EnviRe with Planning*

A suitable environment representation is also key to long-term navigation. An autonomous robot relies on an accurate representation of its environment to plan and replan its trajectory to ensure safe and efficient navigation. EnviRe allows to capture various properties of the environment which can be compiled in traversability- or costmaps. A traversability map provides a binary or continuous measure that informs the planner of non-negotiable terrain or hazards and in the later case allows to encode a preference for traveling over relatively flat terrain. Traversability criteria strongly depends on the deployed platform and may be automatically derived to some degree from the robot model. EnviRe

[3]Geospatial Data Abstraction Library, [Online] Available: http://www.gdal.org

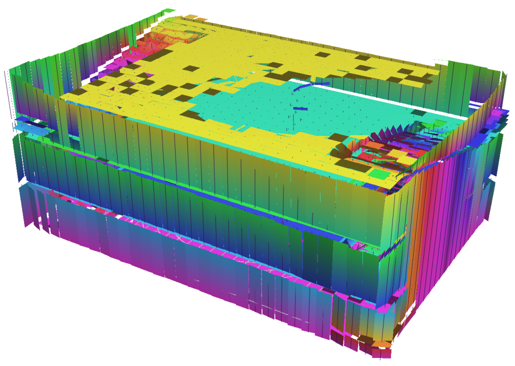

Fig. 6: Vizkit3D visualization of a multi-level-surface map representation of a parking garage.

```
/** EnviRe **/
#include <envire_core/all>
#include <envire_urdf/GraphLoader.hpp>

/** URDF **/
#include <urdf_parser/urdf_parser.h>
#include <urdf_model/model.h>

envire::core::EnvireGraph graph;
boost::shared_ptr<urdf::ModelInterface> robot = urdf::parseURDFFile("model.urdf")
      ;
envire::urdf::GraphLoader graphLoader(graph);
graphLoader.loadLinks(robot->_links);
graphLoader.loadJoints(robot->_joints);
robot->clean();
```

Fig. 7: Example of populating the Envire graph with a URDF robot model.

supports costmaps with multiple layers to allow for optimal planning under a compound cost function, e.g., energy consumption, traveled distance, and trajectory duration. During autonomous exploration, a costmap can be used to capture potential information gain at a so far unexplored frontier. Furthermore, a homogenized environment representation, across mapping & localization, visualization and planning, enables straightforward integration of meaningful human-in-the-loop interfaces. These allow the operator to interact with the planning process by manipulating the environment representation or robot configuration.

C. *EnviRe with Simulation*

EnviRe supports the process to automatically build simulation models in the real-time physics engine (e.g. Open Dynamics Engine [12]). It works by populating the environment with simulation objects. This feature is useful for headless simulation (i.e. on-board internal simulation) because it enables to simulate robot interactions with the environment based on models generated by robot's interaction with the real world.

In order to reproduce an environment, a physical simulator requires a representation of it (see Fig. 3). This representation can be based on EnviRe. The graph structure of EnviRe is in general suitable for a simulation engine as these also concern on spatial-temporal object relations. Furthermore, when EnviRe is also used for the real-world representation the generation of simulated objects becomes straightforward. The simulation-to-reality gap is potentially reduced when the simulations are generated based on real measurements in contrast to off-line designed representations. Internal simulation based approaches can tackle different tasks [13][14]. Specifically our research will initially focus on simulation-assisted navigation in planetary scenarios (e.g. validation, soil interactions).

The unification of robot models enhances the cooperation among modules. Differences between the simulated and the real robot behavior would be reduced when the same robot model shares a common representation. Currently, in EnviRe the robot model is initiated from the static definition given in the Unified Robot Description Format (URDF) [4] (see Fig. 7) and is extended with additional information in Supplementable, Mostly Universal Robot Format (SMURF) [5].

A complete robotics simulation includes not only the physics computations describing how the real world should behave, but also the full execution of the software components which normally perform perception and planning in the real world. On one hand, on-board simulation might require high computational demands. On the other hand, it provides the most realistic approximation to the real world. One of the ideas to reduce the computational costs is to simulate with different fidelities depending on regions of interest. EnviRe's maps spatial resolution is exploited to this purpose. For instance, in navigation the simulation would ideally be of higher fidelity at robot's vicinity than in farther areas.

IV. CONCLUSION

We have designed EnviRe to support our philosophy of modular, tools-based environment representation models. Currently EnviRe support spatial-temporal representation and inference to support a robot to navigate through the environment. We started the development with an open-ended design which can be extended and built upon by others to create autonomous long-term systems for a variety of research and industrial activities. Future work relates to connect EnviRe with existing robotics frameworks (e.g. Rock, ROS) with interests towards model-driven software development of robots.

ACKNOWLEDGMENT

The Entern project is funded by the Space Agency of the German Aerospace Center with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament, grant no. 50RA1407.

REFERENCES

[1] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, 2nd ed. O'Reilly Media, Inc., 2013.

[2] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com.

[4] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.

[5] S. C. Ioan A. Sucan. Moveit! [Online]. Available: http://moveit.ros.org

[6] D. T. Lawton, T. Levitt, C. McConnell, and P. Nelson, "Environmental modeling and recognition for an autonomous land vehicle," in *Workshop on Space Telerobotics*, 1987, pp. 313 –336.

[7] R. Fayek and A. Wong, "Using hypergraph knowledge representation for natural terrain robot navigation and path planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 1996, pp. 3625–3630.

[8] J. Schwendner, S. Joyeux, and F. Kirchner, "Using Embodied Data for Localization and Mapping," *Journal of Field Robotics*, pp. n/a–n/a, nov 2013.

[4]Unified Robot Description Format (URDF), [Online] Available: http://wiki.ros.org/urdf

[5]Supplementable Mostly Universal Robot Format (SMURF), [Online] Available: https://github.com/rock-simulation/smurf_parser

[9] J. Schwendner, "Embodied Localisation and Mapping," Ph.D. dissertation, Universitaet Bremen, 2013.

[10] S. Joyeux, J. Schwendner, F. Kirchner, A. Babu, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig, "Intelligent Mobility - Autonomous Outdoor Robotics at the DFKI," *KI - Künstliche Intelligenz*, feb 2011.

[11] IEEE Robotics and Automation Society, "1873-2015 IEEE Standard for Robot Map Data Representation for Navigation," pp. 1–54, 2015.

[12] R. Smith. (2016) Open dynamics engine. [Online]. Available: www.ode.org

[13] J. Roß mann, E. Guiffo Kaigom, L. Atorf, M. Rast, G. Grinshpun, and C. Schlette, "Mental Models for Intelligent Systems: eRobotics Enables New Approaches to Simulation-Based AI," *KI - Künstliche Intelligenz*, vol. 28, no. 2, pp. 101–110, Mar. 2014.

[14] S. Rockel, S. Konecny, S. Stock, J. Hertzberg, F. Pecora, and J. Zhang, "Integrating physics-based prediction with semantic plan execution monitoring," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 2883–2888.