ABSTRACT

This paper presents a novel autonomous planetary rover navigation system consisting of specialisation functions for localisation and mapping, visual rock detection, and path planning. We propose a common pipeline to share generic data products obtained from real-time sensory data. For solving the technical challenge of integrating such system elements, we use ROS (Robot Operating System) to implement a software design strategy for facilitating the system's development. Furthermore, we present results of the design and implementation of autonomous functions by using computer simulation with realistic and artificial data.

1. INTRODUCTION

In planetary exploration, an autonomous rover can provide support and extend human capabilities by travelling through rough terrain and exploring unknown environments. The actions taken under such circumstances are normally realised with a minimum of human assistance and a limited perception of the robot's surroundings. The rover normally performs these actions autonomously by interpreting noisy information from a variety of on-board instruments, as described in [1]. In this paper, we present the design and implementation steps to define a generic data pipeline for sensor integration. Additionally, we present a technique for integrating such a system with Robotics Operating System (ROS, [2]).

The proposed system consists of autonomous functions that are integrated into software modules for encapsulating interchangeable algorithms in a strategy pattern design (see [1] for details). In specific, we define algorithms for autonomous navigation that model the robot's environment from standardised data types. From the system perspective, the autonomous functions are implemented as software components similar to black boxes with expected input and output. We furthermore turn to the technical challenges of developing such a system using ROS nodes to define and execute asynchronous robot functions.

The system presented requires the integration of sensory data obtained from different sources, with different frequency and quality. The presented model integrates generic data products into a pipelined process. This is a model that evaluates the content of generic data products to provide the robot with a self-awareness state during path planning. The integration process then provides a map for autonomous and safe displacement in an unknown environment.

Section 2 of the presented paper outlines the challenges for autonomous planetary exploration systems and links to related work. Section 3 details the problem description and Section 4 presents a solution using the proposed methodology for the system architecture. In Section 5 the implementation is explained in greater detail including examples of experimental results obtained in laboratory conditions. Finally, Section 6 gives a conclusion and we suggest future work in Section 7.

2. RELATED WORK

To design autonomous rover navigation for planetary exploration, we make use of waypoint navigation that integrates different sources of sensory data to represent the robot's environment. This navigation is a procedure that produces robot mobility by defining sequences of destination spatial points (i.e. a route) as goals to accomplish. For example, for a planetary rover the mission is a route to a specific place in its surroundings where a subsequent in-situ scientific exploration and sample collection is required, as outlined in [4].

In such rover navigation systems, it is essential to take into account the safety to travel long distances. The rover system has traditionally been divided into autonomous functions to modularise task functionalities during the navigation process. Specifically, the functions are tasks for generating path planning, rock detection, localisation, mapping, and mission control for goal designation (see also [1]). These functions process raw sensor data in real-time decision-making for mapping terrain conditions (see [5], [6], [7]).

Most planetary exploration systems combine autonomous functions with a sporadic user input for control and navigation, as it is described in [8]. Limited user control is a highly desirable requirement but it also means that the rover needs to be provided with exhaustive methods for analysing terrain conditions to generate a safe route and motion plan [9]. In most cases of planetary exploration missions human intervention is limited and delayed.

*Corresponding author
In recent works [10], systems for long-range rover navigation use visual information to represent data structures. Current work (such as in [11]) focuses on the generation of prototypes for upcoming planetary exploration missions and makes use of experience from ground applications.

The work presented in this paper focuses on designing a system that interacts with independent functions. Modular real-time processes are described by standardising data products (as in [12]). The system describes the robot’s surroundings by modelling data from visual [13] and Lidar scans. The model also generates derived data products that are integrated into separate subsystems by using ROS open source framework [2]. As a result, this system offers a generic approach to incorporate drivers and other software components into a data processing pipeline.

3. PROBLEM DESCRIPTION
An autonomous planetary rover is a mobile robot with limited user control with a defined mission plan. It is therefore necessary to define the requirements of an autonomous navigation system that iteratively evaluates performance and efficiency. We propose a novel approach to standardise sequences of tasks that progressively integrate sensory data into a collection of stages for processing generic data products.

To represent the robot's surroundings we make use of maps conforming by sensory data processed from each of the autonomous functions. In specific, the system described in this work employs generic data types in a pipeline as a data fusion method for measuring system performance. Hence, the processed data products are also accessible for measuring quality and quantity of data sources as part of the system's performance.

4. SYSTEM ARCHITECTURE
In order to generate a development and testing platform for an autonomous planetary rover system, we propose an architecture that implements independent processes for localisation and mapping, rock detection and path planning. The aim of this work is to provide a service that converts raw sensor data into data products that conforms a map of the robot’s surroundings.

To integrate the autonomous robot functions, we...
implement a strategy pattern design as following the specifications defined in [1] to standardise the pipeline processing of sensor data. The pattern implementation makes use of a generic type’s strategy. In the proposed system a sequence of rules is established that is interfaced by the robot functions to generate derived data types. Additionally, a concrete strategy is defined as a method for encapsulating algorithms based on a changing context.

Each autonomous function implements independent specifications and requirements. To standardise such concrete strategies, we define input and output data products as a description of the system's state. The architecture is shown in Figure 1, which illustrates the implementation of a general policy and the concrete implementations for each autonomous function.

Subsequently, we define interfaces for data transitions that are asynchronously communicated by publishing existing data in particular data flows. This scheme follows the data architecture defined in ROS as an Open Source project to build a meta-operating system for mobile robotics (see [14] [2]). The data products are generated by interfaces of each transition step defined for autonomous functions that are implemented according to the general policy. The implementation of these interfaces standardises the data product transitions and facilitates the organisation of information in a pipelined process.

The system architecture is modelled by a reduced version of Data Fusion process model [6]. The processing of sensor data is done in three stages to generate a map for robot navigation. The process begins with the pre-processing (or enhancing) of raw sensor data and the sensor calibration for removing systematic and random noise. Afterwards, the enhanced data is redefined in its size and format according to each autonomous function requirements. The data product of this phase offers a transition from sensor input into key data elements that describe the robot's environment. The last integration phase is followed by a map generation that makes use of space projections in 2D/3D to generate distinctive maps for each of the applications. The final data product is a map that represents the rover's spatial surroundings with data obtained from rock detection and SLAM (Simultaneous Localisation and Mapping). These maps are constantly updated and represent a model of the robot's mobility.

5. IMPLEMENTATION

This section describes the technical challenges for implementing the proposed system architecture within robot autonomous functions. To implement each autonomous function as part of a sensor fusion model, we define levels that define data transition processing. These processing levels are implementations of concrete policies with a particular topic to publish their data products.

The implementation of the system's pattern is shown in Figure 2. This diagram explains a strategy pattern design with its different parts. The policy implementation is defined according to the robot's environment policy and its autonomous functions.

The pattern elements are:

- **Strategy**: Also known as policy, a strategy in this implementation is a class that contains a definition of data construction stages.
- **Concrete Strategy**: These are classes that implement specific methods for data control according to autonomous robot functionality.
- **Context**: In this use case, the class context contains a ROS node for its interaction and communication with messages/services within other strategies.

To implement the pattern with the autonomous functions and generic data products, the functions are modularised into steps. The pattern definition is realised according to the requirements of independently processed tasks. The processes produce a data product in each step according to a pipeline that models the robot's environment.

For generating the concrete implementation of the robot's functions, we use Plug-in interfaces from the ROS project². In specific, a ROS Plug-in is defined for

---

² The Pluginlib package provides tools for writing and dynamically load plugins programs using the ROS build infrastructure. More information in: http://www.ros.org/wiki/pluginlib
each concrete policy by a base class definition in which the general policy is implemented. The pattern context is placed in a ROS node that makes the publishing of available information possible through a pipelined process.

The system consists of independently executed ROS nodes that access generic data products as topics. The content of each topic is specified by each autonomous function. In the following subsections, the system description is presented for each of the robot’s functions with a description of their data transitions.

5.1. Localisation and Mapping

This module uses a SLAM (Simultaneous Localisation and Mapping) approach that provides a reactive calculation of the rover position and heading while building a sparse map of visual features using an information filter [17]. The module uses SURF features [17] as visual key points and translates them to 3D landmarks once they are filtered from noise and it is obtained values of their uncertainty. Figure 4 illustrates how the SURF feature points (in green) are extracted from the images producing 3D landmarks (in red) through matching and triangulation. Figure 3 presents an example output representing a sparse 3D map of the environment and the estimated rover trajectory.

5.2. Rock Detection

This robot function detects rocks within the rover's field of view by a supervised learning process using image data (see details in [13] and [12]). Rocks are detected by finding changes in intensity and size from input examples and learning a binary classification for the pixels.

A linear Support Vector Machine (SVM) classifier is used to locate rock features in image data by running a binary classification of pixels. A set of rock features is subsequently interconnected through a standard breadth-first search algorithm. This process offers a mask to the input image with rock candidates. Each individual group of candidate pixels from the rock masks is checked for correspondence to a rock. This procedure is realised through heuristic methods that operate on the shape and the texture of the candidate rock's image region. The generated data product is a list of rocks from a candidate rock mask. Finally, the centre point of each detected rock is transformed from image-view 2D R coordinates to 3D W world coordinate system. Since the depth information is not available from the 2D image, it is assumed that a set of observable stable 3D key points and their locations in the image is given (i.e. by invoking a Depth Estimation component of the Localisation and Mapping module).

The output of rock detection function is presented in Figure 5.

5.3. Path Planning

This module is a route generator to reach a pre-located goal in a simulated planetary environment. The route is
obtained by calculating the minimal cost between distances from available waypoints with a graph search algorithm. We use waypoint navigation to reach map positions in robot's known space. In specific, this function produces a route as a collection of waypoints that guide the rover to its goal by avoiding rocks and obstacles.

The algorithm's objective is to estimate the robot's control actions by defining a world state according to its wheel velocities, position and available map information. This information is asynchronously provided as output from SLAM robot functions. We make use of an agent to collect the information and find the actions to follow the route and produce robot mobility.

The method’s performance is measured by properties that characterise its efficiency by a) Priority buffer, b) Amount of visited cells during route generation, c) Endurance of the algorithm in total cycles for replanning and d) Route length by measuring the total steps as route sections to cover. The experiment is defined by incrementally discovered maps and rock-based cost maps for generating a search space comparison among the AStar (A*) [21], Dijkstra [22] and light Dstar (LD*) [23] methods.

The method’s functionality is graphically described in Error! Reference source not found.. On the one hand, the functionality of Dijkstra methods shows an initialisation that requires plenty of the robot's search space. The A* and light LD* methods maintain a lower proportion of use of resources for path planning. On the other hand, the incremental search characteristics of LD* method accumulates search space size over time. A performance comparison is presented in Figure 6.

As a result, we noticed the importance of initialising the search space with as many cells as possible and to make use of replanning with a reduced search space. In specific, we made use of A* for initial propagation and keeping the cost of the shortest path (G-value) from the explored cells based on a map. Experimental testing shows that by using Light D* with a replanning approach is generated a collection of cells in an observe-update scheme.

6. CONCLUSION

We present the integration of a multi-threaded and scalable system for autonomous rover navigation. The system is integrated by using generic data products in a strategy pattern design. It provides a framework that generates maps from independently processed functions of localisation and mapping, path planning and rock detection.

On the one hand, we build software components that implement ROS Plug-ins for defining concrete policies for the software components of each autonomous robot function. Therefore, we construct a system by using ROS components of Message-oriented middleware for encapsulating and communicating the robot functions and our algorithms.

On the other hand, the system's functions are also part of a pipelined process to standardise sensor data into data products. The system performance is shown in each robot function as a collection of variables that describe the state of the planetary rover.

For testing the robot architecture, we make use of a simulation for robot navigation and MER data for training and detecting rocks using image processing. We presented preliminary results for path planning techniques that compare exhaustive and replanning search. The system performance was shown by presenting the methods’ efficiency.

7. FUTURE WORK

The presented system is an integration of autonomous robot functions that is tested under simulated and laboratory conditions. To perform a full driven test, the system should provide performance measures for

---

3 NASA's Mars Exploration Rover Mission (MER).
Figure 7. Images that show the sequential calculation of search space for path planning algorithms. The first row shows the performance of the Dijkstra algorithm. In the second row Astar (A*) is illustrated (both included in ROS navigation stack), and in the third row shows the light DStar (LD*).

Figure 6. Path planning performance is described by the priority buffer, amount of visited cells, amount of cycles after each re-planning cycle.
accuracy and processing requirements for each of the autonomous robot functions. Additionally, the system integration is realised by following a pipelined process. This process contemplates the use of exteroceptive information only and it could complete its functionality by extending its performance with proprioceptive data.

8. REFERENCES