DESIGN OF A GRAPHICAL USER INTERFACE FOR ROVER CONTROL

Łukasz Kostrzewa, Patrycja Cieplicka, Marcin Gajewski, Michał Hałoń
Students' Space Association Warsaw University of Technology
l.kostrzewa12@gmail.com

ABSTRACT

Control of the robots on Mars is a challenging task. To facilitate the interaction between humans and robots, a well-designed Graphical User Interface (GUI) is needed. Mars rovers operate largely autonomously, mostly due to high delays in communication between the rover and the ground station. However when first humans land on Mars, rovers can be controlled by the astronauts, which may extend the functionality of the robots. In both cases a ground station application is needed to monitor rover's state, present data from sensors and allow operator to send control commands.
This paper discusses the design of the rover's GUI applications and technologies used for the robot interface design. It focuses on data visualization and the adjustment of the control application to various robot's configurations and use cases.
The paper also presents the ground station software used in Mars rover Sirius, which has been designed and constructed by the members of Students' Space Association at Warsaw University of Technology, Poland. The software was used during the University Rover Challenge 2019 and the European Rover Challenge 2018.

Keywords: control software, rover, RSVP, GUI

1 INTRODUCTION

Robot control software needs to be fast and robust. The functionality is often varied, including network programming, data processing and visualization, processing of input from control devices e.g. joysticks, robot configuration and video stream management. User experience is also important to facilitate the work of the operator. Engineers at JPL/Caltech have been developing control software for Mars rovers for years. They managed to create advanced and robust software, that has made Mars exploration possible. Mock-up rovers are becoming popular among engineering students. There are dozens of student teams around the world that construct their own rovers and participate in competitions like University Rover Challenge or European Rover Challenge. One of such teams is SKA Robotics from Warsaw University of Technology. Control software developed by SKA Robotics is presented in part 4.

2 ROVER SEQUENCING AND VISUALIZATION PROGRAM

The Rover Sequencing and Visualization Program (RSVP) is a program developed by Jet Propulsion Laboratory. It was built upon Rover Control Workstation software for the Pathfinder mission[3]. It was originally developed for Mars Exploration Rover mission. After some modifications and enhancements it has been also used in Curiosity rover mission. RSVP is a collec-

tion of applications that communicate using the Parallel Virtual Machine API [1]. The most important ones are Rover Sequence Editor (RoSE), that manages commands and creates interface for the user to create new one, and HyperDrive which generates 3D visualization of the rover's surroundings and simulates rover's motion. Other applications include: Logger – records messages passing through the system; Image Viewer – displays and processes images from the rover; Sequence Flow Browser – displays commands execution timeline [3].

2.1 Rover Sequence Editor

RoSE is a text-oriented command editor developed in Java. It allows the operator to enter commands and edit existing command sequences. The commands are based on XML. It is a markup language which is easily readable by both humans and machine. RoSE is highly data-driven as definitions of all the commands are stored in the configuration file. Each day operator can generate around 1000 commands [2]. To facilitate generation of standard and repeatable sequences RoSE supports macros. They are parametrized templates which contain validated sequences of commands.

2.2 HyperDrive

HyperDrive is an advanced tool used for 3D visualization of the rover's surroundings and simulating rover's motion. It is developed in C++ as performance in this tool is critical. The 3D model is generated based on stereo images, which provide information about the depth in the picture. The regions without data are in beige. HyperDrive interface is presented in fig. 1. This tool allows the operator to define control commands. They can select points in 3D and tell the rover to drive there or approach the point with the robotic arm. HyperDrive is also used to simulate motion commands as a part of command sequence validation procedure. HyperDrive is also capable to simulate images. The operator can see how the images that will be taken will probably look like. That simulations also include shadows that are computed based on a relative position of the sun at the given moment [2].

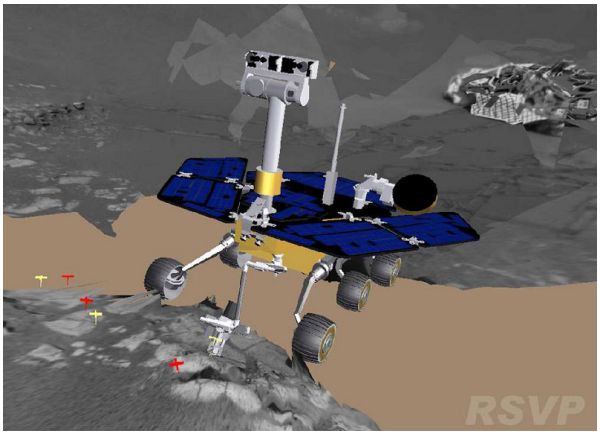


Figure 1: Rover and its surrounding visualization in HyperDrive program. Courtesy NASA/JPL-Caltech.

2.3 RSVP Workflow

The rovers operate in daily mission planning cycle. Each day the data are received from the rover and processed. Then, new commands are generated and send to the rover. Firstly, the operator can analyze the data from various sensors, which is important in detecting anomalous conditions. The operator can view the images taken by the on-board cameras. To understand better the surroundings of the rover, 3D terrain model is generated. When the operator has good situational awareness, they can generate control commands for the next Martian day. The command sequences are validated on a few levels. The syntax and arguments' range are checked. Motion commands are simulated in Hyper-

Drive. Some complex or hazardous tasks can be also simulated on an actual rover in a JPL testbed [2]. After the validation, command sequences are sent to the rover.

3 DESIGN OF A GRAPHICAL USER INTERFACE FOR ROBOT CONTROL

To design a robust and user-friendly control software, a well-designed architecture of the program is needed.

3.1 Separation of concerns

Separation of concerns approach breaks the complexity of a problem into easier to solve, but coupled subproblems. A big and complex software program should be separated into modules. Each module should encapsulate some information and functionality. As the modules create the whole, their interfaces need to be well-defined. In RSVP tool functionalities are separated into a few programs with standardized way of communication using PVM [1]. Such approach facilitates the development and maintenance of the whole system and allows developers to choose the best technology for each component. RoSE is written in Java, while HyperDrive, which requires greater processing speed, in C++ [3].

3.2 Data separation

Data separation approach tells that as much data as possible should be stored in configuration files. It greatly facilitates any modifications in the data. A good solution are markup languages like XML or JSON as they are easily readable for both humans and machines. Rover Sequence Editor stores definitions of commands in an XML configuration file [1]. It simplifies modification of commands definition and adjustment of the control software to a different rover. RSVP, after some upgrades, has been used in the Curiosity rover mission [5].

3.3 Design patterns

Design patterns are reusable solutions to software development issues. They facilitate software system design and maintenance. In RSVP such design patterns as *Model-View-Controller (MVC)*, *Factory* or *State* were used [4]. MVC pattern is commonly used for GUI development. It consists of three components: Model – responsible for program's logic, it can manage or store data; View – presents data to the user; Controller – coordinates data flow between the model and the view and allows the user to make changes in the model.

4 CONTROL SOFTWARE FOR ROVER SIRIUS

Sirius is a mock-up rover designed by the robotics division of Students' Space Association from Warsaw University of Technology, Poland. The rover is designed to participate in international rover competitions i.e. University Rover Challenge and European Rover Challenge series. During the competitions, the rover is teleoperated and is beyond the line of sight of the operator. Thus, the operator needs a GUI program to send control commands and analyze video stream and data from sensors and on-board devices. There are a few basic requirements regarding the control program. It needs to be easily configurable as the rover needs to operate in different task in different configurations. The interface needs to be user-friendly and intuitive because there is no much time for operator training. The program also needs to be cross-platform, work well on both Windows and Linux operating systems. To meet that requirements C++ with Qt framework are used. Input from joystick, which

is used to control the rover, is processed with SFML. The program was named *Main Console 2*.

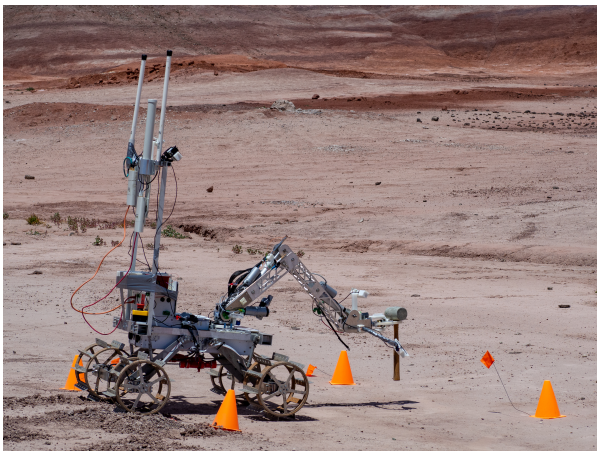


Figure 2: Rover Sirius during University Rover Challenge 2019. Author: Grzegorz Jasina.

4.1 Program architecture

The program is divided into 5 modules. The *Main Module* creates the application's main window. It manages other modules, settings and configuration files. *Rover* manages rover state, stores data from sensors and sends control commands to the rover. *Input* processes input from Joystick and Keyboard and transforms it to control commands. *GUI* creates the interface. *SensorsWidgets* is a set of classes used for data processing and visualization. The program also uses *Protocol* – a library for communication with the rover. The data flow between the modules is presented in fig. 3.

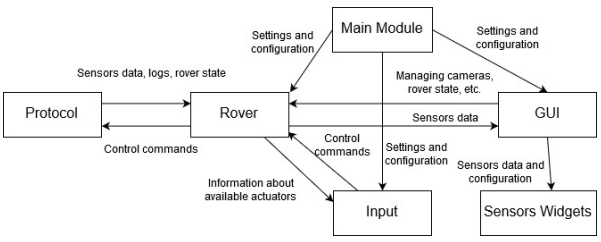
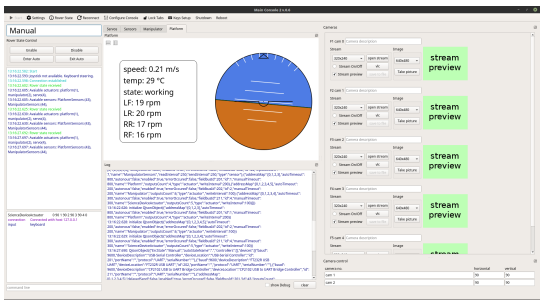


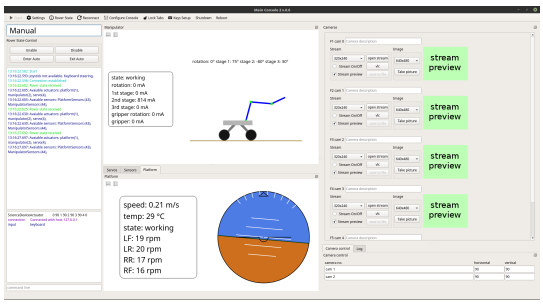
Figure 3: Data flow between the modules in Main Console 2

4.2 Graphical User Interface

The rover operates in different configurations. The user interface also needs to be easily configurable. It is achieved with tabbed interface – similar to the one in web browsers. It allows user to easily rearrange the layout by dragging and dropping the tabs. The operator can also hide the tabs that are not currently needed. The layout of the tabs is remembered when the user restarts the program.



(a)



(b)

Figure 4: User interface in different configurations.

4.3 Data management

To manage sensors data and their presentation Model-View-Controller design pattern was implemented. Data is stored in a model in the *Rover* module. The view presents the data to the user. In *Main Console 2* it is presented in a table and using visualization widgets described in 4.4. The controller is used to manage data flow from the model to the view. Most visualization widgets have their own controllers.

4.4 Data visualization widgets

During the competitions the time is very limited, so it is important for the operator to analyze data from sensors quickly. As raw data are hard to interpret, the set of visualization widgets was developed as a *SensorsWidgets* library. The widgets include, among others, attitude indicator, manipulator model or windows with data from motor boards.

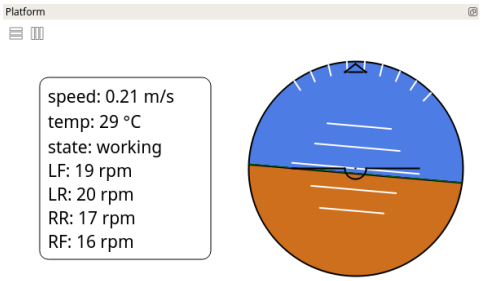


Figure 5: Widgets for sensors data visualization.

4.5 Configuration file

In software development it is important to separate data from the source code. Such approach makes program much more easy to modify. In *Main Console 2* information about on-board devices are stored in configuration file, in JSON format. Thanks to that approach when changing e.g. a sensor to a new one with different parameters, there is no need to recompile the control software and release a new version. Only the configuration file has to be modified.

5 ACKNOWLEDGEMENT

Sirius rover's control software is developed in Students' Space Association, Faculty of Power and Aeronautical Engineering, Warsaw University of Technology, Poland.
Rover Sequencing and Visualization Program is developed by the engineers at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. Great thanks for sharing images.

5 References

[1] Hartman, F., Cooper, B., Leger, C., Maxwell, S., Wright, J., Yen, J. (2005) *Data Visualization for Effective Rover Sequencing*

[2] Wright, J., Hartman, F., Cooper, B., Maxwell, S., Yen, J., Morrison, J. (2004) *Driving on the Surface of Mars with the Rover Sequencing and Visualization Programg*

[3] Maxwell, S., Cooper, B., Hartman, F., Wright, J., Yen, J. (2004) *The Design and Architecture of the Rover Sequencing and Visualization Program (RSVP)*

[4] Cooper, B., (2002) *Rover Sequencing and Visualization Program (RSVP) for the Mars Exploration Rover Program (MER)*, 2002 JPL IT Seminar

[5] MSL project at JPL/Caltech `https://www-robotics.jpl.nasa.gov/projects/MSL.cfm`