Rover Navigation and Visual Odometry: a New Framework for Exploration Activities

Enrica Zereik, Enrico Simetti, Alessandro Sperindé, Sandro Torelli,
Fabio Frassinelli, Davide Ducco and Giuseppe Casalino

Abstract— **Mobile robots are fundamental for future exploration plans, involving both the Moon and Mars, in order to accomplish different useful surface operations in a completely autonomous manner (no human supervision required). A very important issue (even if many other capabilities are desirable) for this kind of robots is surely the mobility skill: a rover able to traverse different terrains in a reliable enough manner represents a great improvement, above all within a scenario in which man presence is not yet planned or still implies big challenges to be faced. In order to endow the robotic system with a high degree of reliability and to be able to simply test different kind of algorithms within a short time, a software architecture developed at Graal Lab, in Genoa, is presented, together with a first module implementing visual odometry.**

I. Introduction

Software technology is traditionally characterized by proprietary platforms and programming languages for each manufacturer; this generates many problems as usually proprietary software is closed source and cannot be modified: this is very limiting for a robotic researcher. Moreover, software reuse becomes very hard: control algorithms developed for a platform from a specific manufacturer cannot be employed for another different platform and must be implemented again. Finally, learning different proprietary languages is a time consuming operation, compared to directly programming in well-known C/C++ language. Hence a reliable software architecture is strongly needed for basically two reasons: to neglect all problems linked to the specific used hardware and, above all, to give researchers the possibility to focus their work on the control algorithms only, while interacting with the low-level operative system through a set of high-level APIs, very simple to use. Furthermore, the presence of such an object-oriented framework allows researchers to develop, implement and test their real-time control algorithms, while being able to reuse already written software. They are also prevented from spending countless hours on debugging low-level code and facing hostile real-time operative system APIs: their research interest can thus be totally focused on control algorithm functioning and possible improvements.

Recently, a few proposal have been presented, as OROCOS and CLARAty; they both are interesting but have some drawbacks: OROCOS is too complex and requires much time to be properly learned, while CLARAty seems to be mostly oriented toward rover motion control, discarding other possible desired capabilities (for example a fine manipulation system through suitable robotic arms and hands). To this aim, the present framework was realized, in order to be able to modularly develop control algorithms concerning different issues (navigation, manipulation, ...) and to simply test these modules, focusing on their functioning. This is the case of a preliminary visual odometry module that is being implemented at Graal and whose presence can be notified to the software architecture itself through a simple registration process. After this module has been registered, it can exploit framework functionalities through its standard APIs.

II. Framework Architecture

The main objective behind the realization of such a framework consists in the development of a software architecture allowing robotic control engineers to focus only on the algorithmic part, thus relieving them of the underlying system details.

The framework is divided into different abstraction levels, whose interaction allows to obtain the following objectives:

- independence of control algorithm code from the underlying software platform;
- minimization of code lines not strictly related to control system algorithm;
- capability of coordination among remote frameworks;
- standard communication mechanism among control tasks with a minimum impact onto the algorithmic part.

The first objective is accomplished thanks to the *Kernel Abstraction Layer* (KAL), whose job is that of masking all calls to the underlying operative system; it allows developers to use services supplied by the operative system but, at the same time, to ignore their specific parameters and peculiar calls. Thanks to this abstraction layer, being at the very base of the framework, another very important objective can be fulfilled: since all specific calls of the operative system are concentrated at this level, the process of framework porting to another software platform is vastly simplified, as only the implementation of a new KAL is needed; the rest of the software is automatically compatible with the new system. The second objective has been met with the development of the so-called *WorkFrame* (WF), which takes care, through system tasks and resources, of the overall system functioning; it imposes a centralized control of resources and helps the developer to properly administer the system. Furthermore, the *Network* (NET) layer realizes the required communication mechanism, abstracting it from the underlying physical data channels (by exploiting the calls provided by the KAL level). Common matrix operations are provided by a C++ mathematical library (CMAT), developed together with the framework. Finally, the *Black Board System* (BBS)

allows each control task to make available to all other tasks (eventually even running on remote systems) data produced during its life cycle. The framework structure is summarized by Fig. 1. All the software has been developed using

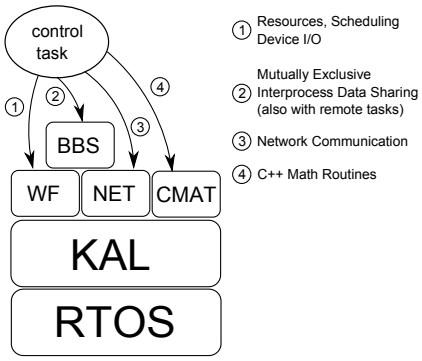

Fig. 1. Framework main components and duties

the object-oriented paradigm and C++ language. Thanks to these choices, the developed code is easy to read and ready to be used, since for every resource and functionality a dedicated class with its specific interface was created. Moreover, in order to emphasize the distinction among the different abstraction levels, each function is part of a proper *namespace*, in order to clearly highlight, within every portion of the code, the belonging of each method and object to the specific framework layer. Currently, the framework has been implemented on Linux/RTAI and, recently, the porting for QNX Neutrino has been completed as well. Work in progress consists in the implementation of a KAL for VxWorks and, above all, for RTEMS.

III. Visual Odometry Module

Motion estimation and environment three-dimensional structure understanding are essential issues in order to dynamically move and interact with surrounding spaces, people and objects; space robots need these capabilities in order to become more and more effective. In particular, for what concerns planetary robotics, today's rovers basically rely on IMU and mechanical odometry in order to detect the system position. Another technique to estimate robot motion is visual odometry; while wheel odometry can be deceived by non-smooth terrain, hard slopes, sandy soil and so on, a reliable vision system could make motion estimation independent of terrain conditions.

Starting from NASA missions, which demonstrated the potentialities of the use of such a visual technique for navigation on planetary surfaces, an algorithm for visual odometry has been developed and is currently being tested (and improved) onto a mobile platform developed at Graal Lab. Preliminary results showed a good behavior, both in terms of precision of the pose estimation and for what concerns time constraints; this visual odometry module, in fact, can provide an estimation of the robot motion within less than a second, usually from 600 to 800 ms, according to the captured images that are to be processed. Using relatively new feature extraction techniques, such as SURF, this proposed algorithm is able to track, in a quite robust manner, natural features present in the environment within a short time (also thanks to the elimination of image pyramids). The preliminary motion estimation is then refined with the aid of optimization techniques, such as the maximum likelihood procedure, in order to better fit feature motion noticed from the images.

The visual odometry module can be simply integrated into the framework as a task providing an estimation of the rover motion. Thanks to the framework architecture, as already said, this feedback estimation can be made available to all other control tasks, which can use it for their processing. Moreover, with this philosophy, as soon as the estimation is available, it can be used by itself (if vision is the most trusted sensor) or together with other available measures (for example coming from an IMU or classical wheel odometry) by simply reading data published by the BBS.

Preliminary tests were conducted and showed the potentialities carried by this kind of image processing; the robot was asked to accomplish closed paths and, while it was continuously moving, visual odometry module was required to estimate the accomplished motion. When the robot goes back again to the starting point, the algorithm returned approximately a null vector and the total estimated motion was correctly represented as a closed path, as pointed out by Fig. 2. This vision algorithm is thus being currently

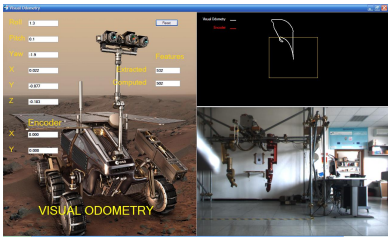

Fig. 2. Preliminary visual odometry interface after a closed path

tested and evaluated and possible improvements are being studied; above all, robustness of this technique seems to be a fundamental issue, in such a way to make vision able to recover cases in which other sensors fail.

The accomplishment of an effective and robust real-time (or at least close to it as much as possible) visual odometry technique would represent a very important step, as it would allow a significant improvement in autonomous navigation ability, in order for the rover to be able to efficiently perform many operations, without slowing down astronauts (e.g. during the execution of cooperative tasks).

Moreover, the accomplishment of independence from wheel odometry, without increasing required time for computation, allows robotic people, working in space applications, to extend the use of rovers to rough terrains. These are only few preliminary steps in order to endow future rovers with an effective dynamic vision system and with the capability to react to external stimuli, in such a way to increase robotic employment in space missions.