Towards Autonomous Long Range Navigation

Erick Dupuis, Pierre Allard, Joseph Bakambu, Tom Lamarche, Wen-Hong Zhu

Canadian Space Agency, Space Technologies
6767 route de l’Aéroport, St-Hubert (Qc), J3Y 8Y9, Canada
Email:firstname.lastname@space.gc.ca

INTRODUCTION

The recent success of the Mars Exploration Rovers “Spirit” and “Opportunity” has demonstrated the important benefits that mobility adds to landed planetary exploration missions. The recent announcement by NASA to increase its activities in planetary exploration (via the Moon and Mars) and the ESA Aurora program will certainly result in an increase in the number of robotic vehicles roaming on the surface of other planets. The current state-of-the-art in control of planetary rovers requires intensive human involvement throughout the planning portion of the operations [3]. Unless the terrain is relatively easy to navigate, rovers are typically limited to traverses on the order of a few tens of meters. Recently, the Mars Exploration Rovers “Spirit” and “Opportunity” have managed to conduct traverses on the order of 100 meters per day.

Although the terrains in which these traverses were accomplished were relatively free from obstacles, this is already quite an achievement. However, to increase the science return and minimise operations costs, future planetary missions will undoubtedly require the ability to traverse even longer distances autonomously. One of the key technologies that will be required to succeed towards the ambitious objectives that are being set internationally will be to streamline the operations of future space missions.

To address this requirement, many laboratories are currently pursuing autonomous navigation of rovers for planetary exploration. Several teams approach the problem of long-range navigation through a succession of short-range traverses. Typically, the rover performs traversability analysis of the terrain in the immediate vicinity of the rover, picks a local path that is obstacle-free and moves the rover towards the target destination (or way point) and executes this trajectory [7][12]. In contrast to this approach, a more behaviourist implementation has been successfully demonstrated on a rover in a desert environment [8]. Despite the fact that only simple navigation behaviours were used, the robot successfully performed traverses up to 1.3 km in natural settings with way points spaced up to 200m apart while using a very limited sensor suite for environment sensing. In addition, some work has been done on instrument placement to reduce the level of human involvement necessary to position a scientific instrument on a target area of interest [10]. In contrast with the previous approaches some research is conducted to increase autonomy by performing basic paradigm shifts. Examples of such research are the inclusion of on-board planning and re-planning capability [3] or localisation schemes targeted specifically at long-range navigation [4].

The Canadian Space Agency (CSA) has been conducting research in ground control and in autonomous robotics for several years already. One of the target applications is planetary exploration using mobile platforms. The emphasis of our research program is on reactive on-board autonomy software and long-range rover navigation. This paper describes recent activities of the CSA in this area. Particular emphasis is put on terrain scanning and modelling, path planning in natural settings and rover guidance. Experimental results from the summer 2004 test campaign are presented.

TERRAIN MODELLING

Range imaging is a reliable and simple way to extract accurate three-dimensional data of objects and environments. With readily available range sensors, it is understandably becoming a very popular technology for 3D modelling. Range sensing is, in our case, the data source to our terrain modelling algorithms, the first step of the long-range navigation scheme. The sensor used in our laboratory is an ILRIS-3D LIdar Detection And Ranging (LiDAR) sensor commercially available from Optech Inc. This scanner uses TOF (Time-Of-Flight) principle to measure depth data on two axes. That is, a single scan provides a complete “image” of the scanned area, not just a line. Although it is not optimised for ranges under ~10 meters, its ability to gather data ranging from half a meter to more than a kilometre away makes it well adapted to long-range considerations. Specifications in a nutshell are: eye-safe IR laser, data sample rate of 2000
points/second, ±20° field-of-view on both axes, modelled output accuracy in the 5mm range and a maximum angular resolution of 26x10⁻⁶ radians (2.6mm spacing @ 100m).

Figure 1 - Scan of the Mars yard (CSA building in background)  
Figure 2 - Scan of the Mars yard cliff

Data returned by the ILRIS-3D is a point cloud expressed as a list of three-dimensional coordinates in the scanner's reference frame. For navigation in a planetary exploration scenario, analysis was conducted to establish the most appropriate data structure to map unknown and unstructured environments.

Meshing

An irregular triangle mesh structure was chosen because it inherently supports variable resolution, unlike the classical digital elevation map (DEM). This allows modelling precise details of uneven areas while simplifying flat areas to just a few triangles, therefore minimising the overall memory requirements. Variable resolution could also be accomplished by the quad-tree representation, a cousin of the DEM. Unfortunately, quad-trees introduce a loss of precision because the acquired data points get approximated by square areas. Adding to that, both DEM and quad-trees are 2.5D representations. Therefore, they do not support concave geological structures like overhangs and caverns, which pose no problem to irregular triangular meshes.

Once a static LIDAR scan is taken, we generate the initial high-resolution mesh from the data points. Triangulating a set of general three-dimensional points presents many difficulties like, for example, determining neighbour points that define a common surface on the real object. To avoid such problem, we take advantage of a property inherent to range sensing devices, namely 2.5D data. As a matter of fact, even though we tend to say these sensors provide 3D points, a single scan will in reality always be a set of 2.5D data. Even though you may get Cartesian data (x, y, z) from the range sensor, it is converted from the original measurements made in a spherical coordinates system (θ, φ, r) defining respectively azimuth, elevation and radius (distance). Data of one scan in this coordinate system is 2.5D, with only one radius (r) value corresponding to any angular position set (θ, φ). That means that neighbours in the θ-φ plane are necessarily neighbours in reality (discontinuity may exist, but will be handled later by the shadow removing algorithm). From there, we compute the Delaunay triangulation of the points projected in the 2D θ-φ plane. This is done by temporarily generating emulated "r" values for every point according to (1), which produces a paraboloid of revolution.

\[
r = \theta^2 + \phi^2 \quad (1)
\]

The resulting emulated surface is fed into a 3D convex hull algorithm [6]. The lower part of the returned hull is the Delaunay triangulation of the data points projected in the θ-φ plane. Projecting this 2D triangulation back on the original coordinates gives us our initial mesh. Let's note here that once projected on the original data, the triangulation may not hold its Delaunay properties anymore (min-max and empty-circle criterions). Also, this procedure can be directly used on DEM, using (x, y) coordinates instead of (θ, φ) and emulating "z" (z axis being along gravity). This in fact shows to be a simpler case to deal with.
We now have a mesh that does not present any holes, even though the real surface "seen" by the LIDAR usually has some discontinuities due to the sensor's low angle of incidence (Figure 4). These shadow regions exist whenever there is an object in front of another. Triangles covering these shadow regions must be identified and removed from the mesh because they do not model an existing surface.

Figure 3 - ILRIS-3D reference frame

Figure 4 - "Shadows"

Shadow filtering

Two consecutive algorithms remove the "shadow" triangles. The first one, doing most of the job, is based on the coefficient of variance (CV) of the triangles' vertices distance ($r$). For every triangle we have in the mesh, we compute the average distance ($r_{mean}$) and standard deviation ($s$) of its vertices. Coefficient of variance then provides a normalized representation of the distance variability among these three vertices (2). That is, triangles having their three vertices near one another will have a low CV while elongated triangles (presumably shadow triangles) will have higher CV.

$$CV = \frac{s}{r_{mean}} \times 100$$

(2)

Level of CV threshold to apply will depend on the scanning resolution. For example, empirical testing showed that a threshold of 8% resulted in a reasonable filtering of the shadows for scan steps around 0.26° while 4% was adequate for the higher resolution 0.16° steps (Figure 5: down-sampled data of Figure 1 without the background buildings). Note here that this filtering also has the effect of removing the unwanted triangles generated by outlier points, if any. In some cases, there will also be unwanted large triangles linking points located at the extremity of the scan. This makes sense from the convex hull algorithm point of view, but is not representative of the real scanned environment. These triangles are simply treated by the second algorithm, which eliminates any triangle that has a perimeter larger than a specified threshold.

Decimating

Finally, in order to reduce memory requirements, the mesh is simplified. The preliminary implementation currently used is a simplified version of the decimation algorithm presented in [11]. Mainly, we do not deal with what [11] refers to as "complex triangles" because our triangulation algorithm does not create any. Plus, instead of using the presented plane splitting technique for re-triangulating the holes, we simply "slide" triangles references from the eliminated vertex to the closest vertex among its neighbours. Triangles that were squeezed to flat lines by the operation are removed. Evaluating the decimation criteria on one point out of two, alternating on every consecutive pass seems to preserve a relatively good shape ratio among the triangles. Figure 6 shows some results. Note that even though the edge preservation criterion is not implemented yet, the mesh boundaries are still relatively faithful to the original mesh.

Figure 5 - Triangulation of a 49468 points scan, 1279 shadow triangles removed, 97623 triangles left
Figure 6 - Decimated triangulation, 23775 triangles left

Stitching

Upcoming activities will include the challenge of stitching multiple meshes in order to build more complete maps from successive LIDAR scans. The Java implementation of all the terrain modelling algorithms presented so far was made along a data structure designed to accommodate this.

PATH PLANNING

In the context of long-range navigation, the path planners used on the CSA’s Mobile Robotics Testbed concentrate on finding a global solution to travel between two points in natural settings while optimizing some cost function. The emphasis is on global path planning rather than local path planning and obstacle avoidance. The basic assumption is that a priori knowledge of the environment is available at a coarse resolution from orbital imagery/altimetry and is refined using local range sensing of the environment. The composite environment model (coarse with refined portions) is then used to plan a path that will be generally safe and that will be updated periodically as new environment data is available.

Previous CSA work used DEM from which a separate traversability map was created based on local slope. The traversability map was itself represented in a quad-tree structure on which a graph search algorithm was applied to find a safe path [5]. While this approach worked, it required a separate structure for the terrain data and traversability map, which forced the update of the traversability map and the quad-tree structure when the DEM was modified.

The use of irregular triangular mesh to represent terrain data allows us to integrate the terrain representation with the path planning easily. To do so, a undirected weighted graph representing the triangles connectivity is created where the triangles are the vertices of the graph and a triangle connectivity to its neighbours are represented as edges. The JGraphT Java Library [9], available freely on the Web, as been used to implement the graph structure and functions.

The edge weight or cost is defined by providing a function that yields a cost based on the edge's vertex. This cost function can use distance between the vertex, slope of the edge, slope of the triangles, mean altitude, or a combination of these to yield the cost associated with moving from one triangle to another. The cost function is associated to the edges at the graph creation, but the actual cost computed only on request.

Once the graph is constructed, a path between the current rover location and a destination can be planned. The process involves four steps:

1) Finding the triangles where the current location and destination lie;
2) Applying the Dijkstra's shortest path search algorithm (provided in JGraphT) to find a safe path;
3) Creating a list of waypoints based on the path found;
4) Generating a simplified trajectory from the list of waypoints.

Applying the Dijkstra's shortest path algorithm on the triangle connectivity graph from the current location to the destination triangles produces a list of edges along the path. This list is used to create a list of the triangles to be traversed. Finding the center of each of the triangle making the path yields a list of waypoints.
The trajectory defined by the waypoints list has often a "saw tooth" look, which makes it difficult to follow for the robot guidance. In order to alleviate the problem, the waypoint list is processed in order to remove unnecessary waypoints while maintaining the resulting trajectory on safe ground. Figure 7a and Figure 7b show the effect of the trajectory simplification.

![Figure 7a - Trajectory generated using waypoint list](image)
![Figure 7b - Trajectory after simplification](image)

Various cost functions have been tested with the path planner. For example, Figure 8 shows the result for planning a path from location (15.0, 5.0) to (80.0, 100.0) using a cost function that takes into account distance and slope. Figure 9 shows a path obtained between the same two points using a modified version of the first cost function that associate a high cost to low altitude terrain (i.e. is tends to have the rover travel on high grounds). Such a cost function could be used to have a rover take the "scenic route" in order to get better ground coverage with a camera.

![Figure 8 - Trajectory minimizing slope and distance traveled](image)
![Figure 9 - Trajectory minimizing slope and distance traveled while maximizing altitude along the path](image)

ROVER GUIDANCE

The usage of a scanning lidar for terrain sensing results in a concept of operation slightly different from the more common schemes using stereo pairs. Indeed the scanning lidar used on the CSA’s Mobile Robotics Test-bed typically takes on the order of one or two minutes to perform a terrain scan but it has a sensing range of over a kilometre. As a result, the terrain is not imaged continuously. It is rather imaged using snapshots taken at discrete intervals. Obviously, since the lidar is located near ground level, the effective range of the measurements is typically not on the order of kilometres but of a few tens of meters. Consequently, the rover has the ability to plan path segments on the order of 20 to 30 meters and does not rely on environment sensing while moving along these path segments. It has, therefore, been necessary to develop guidance software that can keep the robot precisely on the planned trajectory. The proposed rover guidance has two mains parts: 3D odometry and autonomous motion controller.
3D Odometry

The first step to ensure that the robot does not deviate from the planned trajectory is to provide accurate knowledge of its position. This task can be accomplished by fusing odometry, inertial, and absolute heading data. In this system, the robot odometry is combined with a solid-state IMU (inertial measurement unit) sensor to provide inertial navigation with 3D odometry. An absolute heading sensor, a digital compass TCM2 from PNI Corporation provides absolute orientation in Yaw (in Mars exploration, this sensor could easily be replaced by a sun sensor).

The angular velocities measured by the IMU are integrated to form the orientation in $SO(3)$ using the quaternion formulation. Once the orientation is obtained, the 3D odometry can be easily obtained by incorporating the robot odometry based on the wheel movements. However, data drifting in 3D odometry is fundamental. Correction or recalibration is needed regularly. In the system, the gravitational vector is extracted from the three-axis acceleration signals provided by the solid-state IMU. The gravitational vector is used to correct the pitch and roll generated by the 3D odometry. Since the gravitational acceleration vector is very noise, particularly when the robot rolls over small rocks, Kalman filter based on quaternion in $SO(3)$ is used. Finally, the yaw correction is performed by the absolute heading sensor, which is activated every time the robot stops because the compass data is not reliable when the robot motors are running due to the electromagnetic interference.

Table 1 lists three different IMU with their cost, random walk, and the resulting orientation drifting. The first row corresponds to the IMU used in this system. The second and third rows correspond to two IMU suggested by Durrant-Whyte [13]. The angular velocity random walk is a key indicator that represents the original performance of the IMU. In general, it is proportional to the standard deviation of the angular velocity measurement noise. A big random walk is always associated with a low price and a high orientation drifting for a given integration algorithm. In view of the first two rows, the random walk in the first row is about 4.5 times higher than that in the second row. But the orientation drifting in the first row is only twice as much as the second row. This indicates the effectiveness of the quaternion based integrator developed at CSA.

<table>
<thead>
<tr>
<th>IMU</th>
<th>Price (US$)</th>
<th>Random Walk (deg/hr$^{1/2}$)</th>
<th>Orientation drifting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMU300 (Crossbow)</td>
<td>3K</td>
<td>&lt; 2.25</td>
<td>3 deg in 10 minutes, 6 deg in 15 minutes</td>
</tr>
<tr>
<td>ISIS-IMU (Inertial Science)</td>
<td>10K</td>
<td>&lt;0.5</td>
<td>3 deg in 15 minutes</td>
</tr>
<tr>
<td>DMARS-I (Inertial Science)</td>
<td>30K</td>
<td>&lt;0.02</td>
<td>0.5 deg in 15 minutes</td>
</tr>
</tbody>
</table>

Experimental results of the 3D odometry with the gravitational vector based pitch and roll correction are illustrated in Figure 10. The dashed line represents the planned trajectory that covers an 8m by 8m region. The solid line represents the actual robot positions in 3D. In the far edge when x is around 10m. The vertical difference between the solid line and the dashed line is due to the fact that the commanded trajectory does not take into account a rise in the physical terrain. After completion of the closed trajectory, the total travelled distance is approximately 32m. The position drifting in z only amounts to 3.1cm. This error is indicative of the portion of the error due to drift of the IMU. In contrast, the position drifting by using the robot odometry alone (based on wheel movements) amounts to 9.3m for the same trajectory, resulting from the significant orientation error caused by wheel slippage.
Autonomous Motion Controller

The developed motion control is based on a discontinuous state feedback control law initially proposed by Astolfi [2]. Experimental results in an outdoor 3D environment (see Figure 1) show the robustness and the stability of the developed path following approach. The Figure 13 illustrates an 8m by 8m square-shaped reference path following result. A part on the path was on a slope and during the autonomous motion execution, artificial perturbations were induced twice as show in Figure 14. This figure shows that the rover can robustly, quickly and smoothly recover the path. During our tests with and without perturbations, physical error at the end of the motion was always negligible (about few centimeters in position and few degree in orientation) and it is due essentially to the wheels slippage and the gyroscope drift. For example, the errors of the test in Figure 13 and Figure 14 were on the order of 15 centimeters in position and 3.3 degree in orientation. The physical error in position was measured by putting marks on the ground, while an onboard compass (in rest state) provided the orientation error. The same trajectory was executed without perturbation and the result is shown in Figure 10. Others trajectory such as eight-shape (Figure 15, Figure 16) and closed-spiral-shaped (Figure 17, Figure 18) have been also executed. All those results illustrate the precision and the performance of the proposed autonomous motion controller and the 3D odometry based localization.

Figure 10 - Experimental results from 8m x 8m outdoors traverse

Figure 13 - Square-shaped path following with artificially induced perturbations: in 3D
Figure 14 - Square-shaped path following with artificially induced perturbations: 2D projection

Figure 15 - Eight-shaped trajectory execution

Figure 16 - Eight-shaped trajectory 2D projection

Figure 17 – Closed spiral-shaped trajectory execution

Figure 18 - Spiral-shaped trajectory 2D projection
CONCLUSION

This article has presented and discussed the advantages of map building via triangulation and path planning through an irregular triangulated mesh, and the algorithms used for rover guidance in outdoor terrain. Experimental results demonstrate that the terrain-modelling scheme can be used to model natural terrains efficiently and is directly usable for path planning using a variety of cost functions. The robustness and stability of the rover guidance in rough 3D terrain is demonstrated. Closed trajectories of up to 50 meters have been executed successfully in natural terrain even in the presence of external disturbances. Position errors on the order of less than 1% of the total distance travelled have been observed in many cases.

Future work will focus on Simultaneous Localisation and Mapping, increased autonomy and longer-range navigation. Traverses on the order of 100 metres and more will require the ability to stitch maps together and to perform map-based localisation.

REFERENCES