Path Planning for Planetary Exploration

Ioannis Rekleitis
School of Computer Science,
McGill University
yiannis@cim.mcgill.ca

Jean-Luc Bedwani, Erick Dupuis, and Pierre Allard
Canadian Space Agency,
Space Technologies
FirstName.LastName@space.gc.ca

Abstract

In this paper we present the work done at the Canadian Space Agency on the problem of planetary exploration. One of the main goals is the over-the-horizon navigation of a mobile robot on a Mars like environment. A key component is the ability to plan a path using maps of different resolutions and also to refine/replan when more data becomes available. Our algorithms on path planning and path segmentation are presented together with results from two years of experiments in realistic conditions.

1. Introduction

Planetary exploration is one of the biggest challenges in robotics research, and at the same time one of the more well known successes. The journey of the Mars Exploration Rovers (MERs) Spirit and Opportunity [16] has open the door for future missions to Mars and the Moon. Current missions have shown capabilities for semi-autonomous navigation where the rovers are capable of planning local paths and avoiding obstacles. Future missions such as the "Mars Science Laboratory" (MSL) [24] and ESA's ExoMars [23] will be required to traverse much longer distances. An important component of such missions is the capability to plan trajectories far beyond the sensing horizon of the rover.

At the Canadian Space Agency (CSA), an integrated solution has been developed to address the problem of Over-the-Horizon autonomous navigation. The sensing modality of choice is an active vision LIDAR system; the environmental representation is based on the Irregular Triangular Mesh (*ITM*); and graph-based path planning algorithms have been developed that utilize the *ITM* structure. In this paper the focus is on path-planning for planetary exploration.

A modified Pioneer P2-AT robot has been used as our mobility platform. In addition to the standard wheel encoders and compass a six-axis Inertial Measurement Unit (IMU) has been added. The measurements from the wheel encoders, the compass and the IMU are fused together to provide an accurate estimate of the robots pose [2]. The past year a new LIDAR sensor was developed at CSA consisting of a SICK laser range finder mounted vertically on top of a pan-unit. The sensor scans vertical slices of 180° field of view and is capable of rotating by 360° thus recording the surroundings of the rover. Figure 1 shows CSA's mobility platform together with the LIDAR sensor during an experiment. All experiments were performed at the Mars Emulation Terrain located at the CSA's facilities. The terrain is 60 m by 30 m and is constructed to represent a variety of topographical features encountered on Mars.

Figure 1. The modified P2-AT robot with the LIDAR sensor during an experiment.

In the next section we discuss related work. Section 3 presents a brief overview of our approach to over the horizon autonomous navigation. The next two sections discuss in detail the path planning and path segmentation methodologies developed at CSA. Experimental results are presented in Section 6, and the paper finishes with conclusions and future work.

2 Related Work

The work on planetary exploration can be divided according to the sensing modality used and also according to the environment representation used. Both vision [17, 8, 13] and LIDAR [10, 3] technologies have been proposed, each one having different advantages and disadvantages. Early work on planetary exploration using LIDAR [10, 3], though promising, was not compatible with the weight constraints of space missions. The Mars Exploration Rovers are currently performing long traverses using vision [9]. Vision although lightweight, requires more computing power, has limited range and accuracy. Currently, LIDAR based systems[1] have been successfully used in space mission and thus are space qualified. The major advantage of LIDAR systems is their superior resolution and range.

The problem of autonomous long range navigation is also very important in terrestrial settings. The DARPA grand challenge in 2005 resulted in several vehicles travelling more than 200 Km over desert terrain [18]. The majority of the contestants used a combination of multiple LIDAR, vision, and RADAR sensors. Similar work involved traverses on the

[1] http://www.neptec.com
http://www.optech.ca/
http://sm.mdacorporation.com/

order to 30 Km in the Atacama desert [25] using vision. See also [6] for a discussion of the many challenges and more related work.

Several types of exploration methods have been considered for autonomous planetary exploration. In the area of path-planning, a bug-like algorithm was proposed for micro-rovers [14], while potential-field like methods were also suggested [22]. The autonomy aspects of planetary exploration have been discussed ranging from trajectory planning in simulation [27] to varying behaviours based on the vision sensor data collected [12] and for ensuring visual coverage [7]. Finally the problem of human to rover interactions has also been studied [15].

Currently, the most advanced exploration robots that have been deployed for planetary exploration are the Mars Exploration Rovers (MER) "Spirit" and "Opportunity". These rovers have successfully demonstrated, on Mars, concepts such as visual odometry and autonomous path selection from a terrain model acquired from sensor data [4]. The main sensor suite used for terrain assessment for the MER has been passive stereo vision [26]. The models obtained through stereo imagery are used for both automatic terrain assessment and visual odometry.

In the case of automatic terrain assessment, the cloud of 3D points is used to evaluate the traversability of the terrain immediately in front of the rover, defined as a regular grid of square patches. In the case of visual odometry, the model is used to identify and track features of the terrain to mitigate the effect of slip [11].

It is worth noting that when the vehicle has enough clearance, then the error introduced by a vision system is acceptable. Our mobility platform, a modified Pioneer P2AT, has very low tolerance for obstacles and thus a laser range sensor (LIDAR) is used as the main sensing modality. A LIDAR sensor is capable of providing range data to build terrain models with 1-2 cm accuracy. Such an accuracy would be difficult to attain with most stereo vision systems over the full range of measurement. Such accuracy is also very important for the scientific return of the mission. In addition, LIDAR sensors, return accurate geometric information in three dimensions in the form of a 3D point-cloud without requiring additional processing. Finally, since they do not rely on ambient lighting, we do not have to address the problems arising from adverse lighting conditions.

We are taking advantage of the LIDAR sensor attributes to build a terrain model that not only preserves the geometry of the terrain, but it is also readily usable for path planning and navigation. The majority of previous work uses some form of grid representations, often in the form of an elevation map. Our approach utilizes an *ITM* to represent the environment. By allowing for variable resolution we are capable to preserve the surface details in the topographically interesting regions, obstacles, while achieving a very sparse representation

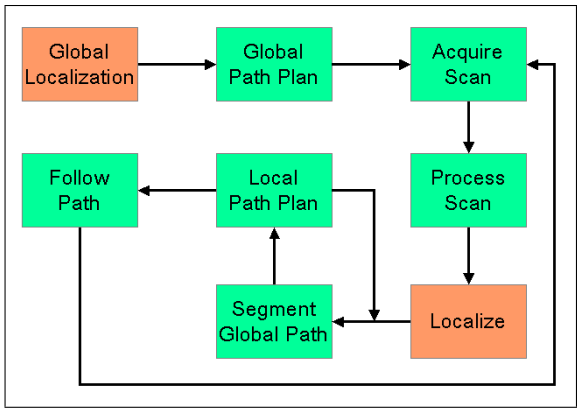


Figure 2. The main components of Autonomous Over-the-Horizon Navigation

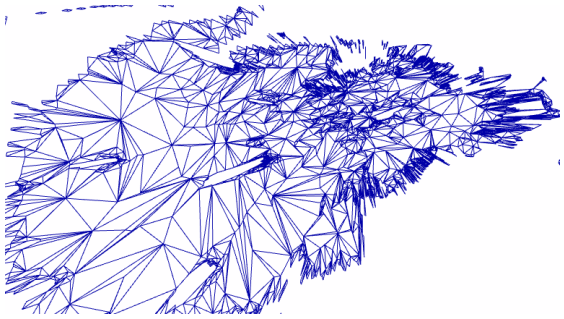


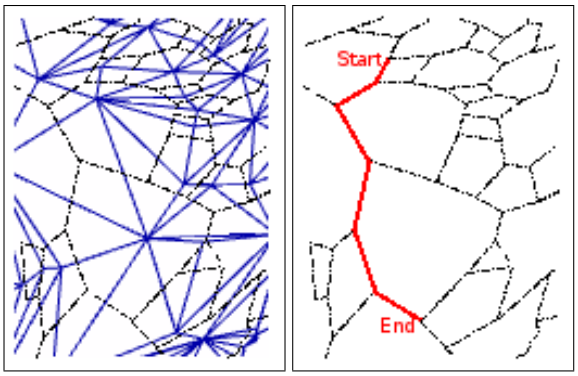
Figure 3. A local scan represented as an Irregular Triangular Mesh (*ITM*)

over the flat areas, allowing for very efficient path planning.

3 Operational Scenario

The goal of our work is to navigate autonomously from the current position to an operator-specified location which lies beyond the sensing horizon of the rover. We operate under the assumption that a global map is available from satellite imagery, previous missions, or from data collected during descent [19]. Figure 2 presents a flowchart of the complete exploration process. At top level, the rover uses the global map to plan a path from its current position to an operator-specified location; the rover collects the first local scan using its LIDAR sensor, then the global path is segmented successively using the locally collected scans; each time an optimal trajectory is planned through the representation of the local scan. Finally, the rover uses the local path to navigate to the next way-point. At the current state, the pose estimation from the Inertial Measurement Unit (IMU) and the odometer, combined with the trajectory length in the order of ten meters, allows to safely navigate in open loop without relocalizing between successive scans.

Central to our approach is the representation used for modelling the surrounding terrain. The LIDAR sensor returns a set of points in 3D. We used a Delauney triangulation in polar coordinates which results into an *ITM* representing the sensed surface. The mesh is further decimated



(a) (b)

Figure 4. (a) A section of the *ITM* and the underlying (dual) graph. (b) A path through the dual graph

by combining several coplanar triangles into a single triangle. The implementation of the terrain modelling and decimation, using a triangular mesh, is done using the Visualisation Toolkit [1] libraries. Figure 3 shows a decimated *ITM*, note the large triangles at the flat areas and how the triangle density increases where there are obstacles. A more detailed discussion of the *ITM* terrain representation is outside the scope of this paper; please refer to [20, 21] for more details.

4 Path-Planning

One of the advantages of the *ITM* representation is that it is amenable to path planning. Indeed, the triangles in the mesh form individual cells. While traversing the terrain, the robot moves from one cell to another by crossing their common edge. The *ITM* representation can therefore easily be transformed into a graph structure where the cells are the graph nodes and the common edges between cells are the edges between the nodes of the graph; see Figure 4a. The path-planning problem is then formulated as a graph search problem; see Figure 4b.

The results described in this paper were obtained using Dijkstra's graph search algorithm [5] from the *jgrapht* java library with a variety of cost functions taking into account distance travelled, terrain slope, and terrain roughness. One of the main advantages of graph search techniques is that they do not get stuck in local minima: if a feasible path exists between any two locations, graph search algorithms will find it. In addition, given any cost function, Dijkstra's algorithm always returns the lowest cost solution between any two locations.

4.1 Cost Functions

One of the most important factors affecting the performance of Dijkstra's graph search algorithm is the cost function that is used to guide the search. The cost function is used at every step of the search to evaluate the cost Q of travelling from cell i to cell j.

4.1.1 Cell Count Cost Function

The simplest cost function that can be used in the graph search algorithm is the cell count cost function. This function associates a unit cost to every edge in the graph: $Q = 1$. The output of a graph search using the cell count cost function is a path that minimizes the number of cells in the path. This cost function is easy to compute and leads to a very quick graph search. In the context of *ITM* models, it is meaningless since the size of cells varies widely. However, it can be used to perform a quick check of connectivity between sets of cells in the *ITM* model.

4.1.2 Distance Cost Function

Another simple cost function is the distance cost function. The cost of every edge in the graph is the Euclidean distance between the centre points of cells i and j. It is computed as follows: $Q = \|\mathbf{x}_j - \mathbf{x}_i\|$.

The output of the graph search using this cost function minimizes an approximation of the geodesic distance when travelling from cell centre to cell centre between any two locations. The distance cost function is useful to generate energy-efficient paths on an *ITM* terrain model but it does not take into account the terrain traversing limitations of the robot.

4.1.3 Distance-and-Slope Cost Function

A more realistic approach is to use a cost function that takes into account upward slope, downward slope and cross slope in addition to the distance between cells. This cost function is computed as follows:

$$Q = \|\mathbf{x}_j - \mathbf{x}_i\| \; \alpha \; \beta \tag{1}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the geometric centres of cells i and j respectively. The parameters α and β are penalty multipliers that take into account the slope of the terrain. They are obtained by computing the slope along the path and the slope cross-wise to the path in the following manner. First, the unit normal of cell j is computed.

$$\mathbf{n}_j = \frac{\mathbf{p}_j^1 \times \mathbf{p}_j^2}{\|\mathbf{p}_j^1\| \|\mathbf{p}_j^2\|} \tag{2}$$

where $\mathbf{p}_j^1$ and $\mathbf{p}_j^2$ are the coordinates of any two vertices of cell j. This calculation is valid based on the assumption that all vertices are co-planar, which is always true for us since the cells are triangular. The unit normal is assumed to be pointing up. The cross-track and along-track vectors are computed as:

$$\mathbf{c} = \mathbf{n}_j \times (\mathbf{x}_j - \mathbf{x}_i) \tag{3}$$

and

$$\mathbf{a} = \mathbf{c} \times \mathbf{n}_j \tag{4}$$

The cross-track slope angle and the along-track slope angles are then computed as:

$$\phi = |\text{atan2}(\mathbf{c}_z, \sqrt{\mathbf{c}_x^2 + \mathbf{c}_y^2})| \quad (5)$$

and

$$\theta = \text{atan2}(\mathbf{a}_z, \sqrt{\mathbf{a}_x^2 + \mathbf{a}_y^2}) \quad (6)$$

The values of ϕ and θ are then used to compute the slope penalty parameters in Eq. (1) as follows:

$$\alpha = \begin{cases} k_a \frac{\theta}{\theta_{max}} & \text{if } 0 < \theta \leq \theta_{max} \\ \infty & \text{if } \theta < \theta_{min} \text{ or } \theta > \theta_{max} \end{cases} \quad (7)$$

$$\beta = \begin{cases} 1 & \text{if } \phi \leq \phi_{max} \\ \infty & \text{if } \phi > \phi_{max} \end{cases} \quad (8)$$

Unlike the distance cost function, the distance-and-slope cost function takes into account the terrain-traversal limitations of the robot. It generates paths that do not exceed the slope capabilities of the robot in all directions (along-track and cross-track). However, one of the major limitations of this cost function is that it does not take into account the physical size of the rover. When used on a mesh that contains cells that are smaller than the physical size of the robot, it can generate paths that travel through areas that are too narrow for the rover to cross.

4.1.4 Rover Footprint Cost Function

The rover footprint cost function was developed to overcome the limitations of the distance-and-slope cost function. In fact, it is computed in a manner very similar to the distance-and-slope cost function except that the calculations are based on the normal of the set of all cells touching the footprint of the rover defined as a flat disk of a given radius r. The rover footprint cost function is formulated as:

$$Q = \|\mathbf{x}_j - \mathbf{x}_i\| \; \alpha \; \beta \; \gamma \; e^{\frac{\|\mathbf{x}_j - \mathbf{x}_I\|}{A_i + A_j}} \quad (9)$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the geometric centres, and A_i and A_j are the areas of cells i and j respectively. The parameters α and β are again penalty multipliers to take into account the slope of the terrain. The parameter γ is a penalty multiplier taking into account the roughness of the terrain. Parameters α, β and γ are computed taking into account the footprint of the robot. The exponential term is used to encourage the path to cross wide cells instead of long thin cells. The usefulness of the exponential term will be explained in the section on trajectory generation later in this document.

The footprint of the robot is defined as $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, the set of all cells with at least one vertex within distance r from $\mathbf{x}_j$. The average normal of the terrain within the footprint is defined as:

$$\bar{\mathbf{n}} = \frac{\sum_{k=1}^{m} A_k \mathbf{n}_k}{\sum_{k=1}^{m} A_k} \quad (10)$$

where A_k and $\mathbf{n}_k$ are the area and the upwards-pointing unit normal of cell k. The cross-track vector and along-track vector are then computed as:

$$\bar{\mathbf{c}} = \bar{\mathbf{n}} \times (\mathbf{x}_j - \mathbf{x}_i) \quad (11)$$

and

$$\bar{\mathbf{a}} = \bar{\mathbf{c}} \times \bar{\mathbf{n}} \quad (12)$$

The cross-track slope angle and the along track slope angles are then computed as:

$$\phi = |\text{atan2}(\bar{\mathbf{c}}_z, \sqrt{\bar{\mathbf{c}}_x^2 + \bar{\mathbf{c}}_y^2})| \quad (13)$$

and

$$\theta = \text{atan2}(\bar{\mathbf{a}}_z, \sqrt{\bar{\mathbf{a}}_x^2 + \bar{\mathbf{a}}_y^2}) \quad (14)$$

The values of ϕ and θ are then used to compute the slope penalty parameters α and β using equations (7) and (8).

The roughness penalty factor γ is computed by evaluating the distance between every vertex of the *ITM* contained in the rover footprint and the plane defined by the average normal $\bar{\mathbf{n}}$ and $\mathbf{x}_j$, the centre point of cell j. The maximum distance between the vertices in the footprint and the plane is computed as:

$$\delta = \max(|\bar{n} \cdot (\mathbf{p}_k - \mathbf{x}_j)|) \; \forall k \; / \; \|\mathbf{p}_k - \mathbf{x}_j\| < r \quad (15)$$

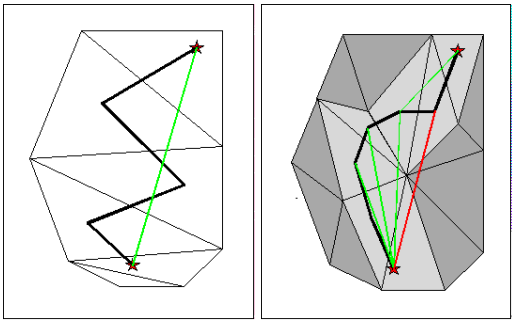
where $\mathbf{p}_k$ is any vertex of cell k. The roughness penalty factor γ is then computed from the maximum deviation from the average plane. Experimentation using realistic terrain models acquired using the 3D sensor in the CSA's Mars emulation terrain has shown that it is sufficient to compute γ in the following manner:

$$\gamma = \begin{cases} 1 & \text{if } \delta \leq \delta_{max} \\ \infty & \text{if } \delta > \delta_{max} \end{cases} \quad (16)$$

The rover footprint cost function defined in Eq. (9) coupled with Dijkstra's graph search algorithm is guaranteed to find a path that respects the robot's hill-climbing limitations in all directions and rough terrain traversal capabilities. Taking into account the physical dimensions of the robot. Every cell selected by the planner is guaranteed to be safe for the robot to cross. Furthermore, it finds the path that minimizes the total Euclidean distance between the centre points of all cells in the path.

The main drawback of the rover footprint cost function is that it is very costly to compute: every cost function evaluation requires the search engine to find all neighbours within a given radius of every cell being evaluated. The average normal, along-track vector and cross-track vectors as well as the surface roughness of the footprint need to be evaluated.

One of the issues encountered during our field-testing is due to the fact that the environment sensor has a 360° field-of-view. Dijkstra's graph search algorithm grows the search space from the start location irrespective of the target destination. The planner ends up spending much precious time searching in the direction away from the destination. Path planning on realistic terrain models using the rover footprint cost function described in Eq. (9) can lead to planning times on the order of 30 to 60 minutes, which is unacceptable.



(a) (b)

Figure 5. (a) Example of path planned through an *ITM*. (b) Example of trajectory simplification on an *ITM*

4.1.5 A* Cost Function

The A* cost function was implemented to reduce the search time of the rover footprint cost function. This cost function turns Dijkstra's undirected graph search algorithm into the A* directed graph search. The A* cost function is computed exactly in the same manner as any of the cost functions described above except that a heuristic term is added to it to guide the search.

$$Q_{A^*} = Q + Q_h \tag{17}$$

The heuristic term is computed as follows:

$$Q_h = \|\mathbf{x}_j - \mathbf{x}_F\| \tag{18}$$

where $\mathbf{x}_F$ is the final destination of the path being sought.

The version of the A* cost function that was implemented for the experiments conducted during the 2007 field-testing season was based on the rover footprint cost function. However, the heuristic cost function can be added to any of the cost functions described earlier to accelerate the search by guiding it towards the final destination. The resulting path may not minimize the cost function over all possible paths but, in most cases, an acceptable solution will be found much faster.

4.2 Path simplification

One important observation is that the output of the graph search algorithm is a series of cell identifiers. When traversed in the given order, the cells will lead the robot from start to destination along a path that is deemed safe and optimal according to the given cost function. The robot's guidance and motion control algorithms, however, require a trajectory composed of a series of points in 3D space. The easiest way to convert cell ID's to 3D points is to use the geometric centres of the cells as trajectory points. The trajectory is then the list of the centre points of all cells in the list generated by the graph search algorithm. This results in an unacceptable trajectory that zigzags unnecessarily between cell centres; see Figure 5a.

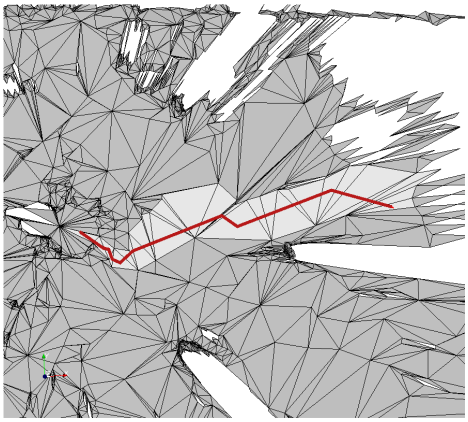


Figure 6. Results of path planner on typical Irregular Triangular Mesh

It is therefore necessary to smooth out the resulting trajectory by removing superfluous via-points in the trajectory. The trajectory simplification algorithm first defines a safety corridor as the set of all cells in the path generated by the graph search algorithm. Each of these cells has been identified by the planner as a safe area on which the robot can tread. The trajectory generation algorithm then assigns a via-point to the geometric centre of every cell in the path. The simplification algorithm removes intermediate points in the trajectory and verifies whether the straight-line segment joining the two points on either side of the removed via-point stays on the safety corridor. This procedure is applied iteratively starting from the initial location of the robot. Points are removed as long as the safety corridor constraint is not violated. At this point, the algorithm is re-started from the location of the via-point that could not be removed and stops when reaching the final destination. Figure 5b shows an example of the trajectory simplification algorithm. The light grey cells are the safety corridor, the thick black line is the raw path joining the geometric centres of all cells in the path. The green lines show the successive steps in the simplification of the trajectory by the elimination of superfluous via-points. The red line is a case of a simplification that would lead to the violation of the safety corridor constraint. In this context, the usage of *ITM* introduces additional challenges. First, on flat terrain, the cells are relatively large as can be observed by comparing Figure 5a and Figure 5b. Therefore, although large cells are preferable for safety reasons, a cost function taking only distance travelled into account would unduly penalize traversal through large cells because the raw path zigzags between cell centres.

On the other hand, on rough terrain, the cells are much smaller and the resulting safety corridor can be very narrow, hence more difficult to navigate. This fact prompted the introduction of the exponential term in Eq. 9 to favour the crossing of wide cells. This results in a safety corridor that is wider and hence in paths that can be simplified to a larger

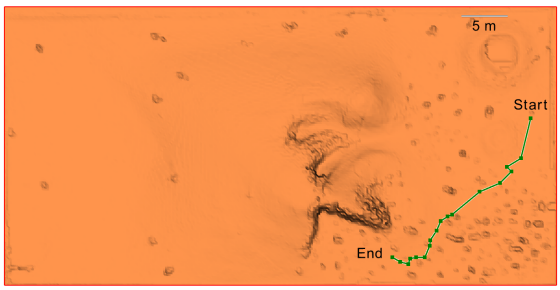


Figure 7. Global path planned using the low resolution map of the Mars-like terrain at CSA

extent than narrow paths.

In addition, the trajectory simplification algorithm, by design, simplifies the trajectory until it skims the boundaries of the safety corridor: the resulting trajectory can therefore skim obstacles. If the width of the robot is not considered, the planned trajectory will result in a collision between the robot and the environment.

Figure 6 shows a path that was planned in a typical terrain scan obtained using a LIDAR range scanner in the CSA's Mars emulation terrain. The scan was acquired from the start point located at the left end of the planned trajectory (the red multi-segmented line). The figure clearly shows that the trajectory remains within the bounds of a safety corridor without going through the centre points of every cell in the path.

5 Path-Segmentation

The path planning approach described above can be used in a variety of circumstances. The fast distance-and-slope cost function is used for global path planning over the low resolution global map; see Figure 7. The generated path while avoiding large slopes and big obstacles leads over smaller obstacles, not discernible at that resolution. This global path provides a general guideline for travelling from the starting position to the operator selected destination, beyond the robots sensing horizon. The global path is then divided in smaller segments using a series of via-points. These via-points act as intermediate destinations, each of them inside sensing distance from the previous one. The more accurate rover-footprint cost function in conjunction with the A* cost function is used to plan a path between via-points, using the high-resolution map constructed using a local scan obtained from the LIDAR.

The first step in segmenting the global path path is to consider what is the appropriate effective range of the local scan[2]. The range of our sensor is between ten to fifteen meters, further than that the angular resolution is too coarse for safe operations. When the sensed terrain is flat and sparsely populated with obstacles then a range of ten meters is used. When a large number of obstacles surrounds the rover, the sensed scan is full of shadows, areas with no data located behind obstacles, in such case, the effective range is usually set to four meters; see Figure 8a for such a terrain where the four meter range is indicated by a yellow circle. The intersection point between the global path and a sphere centred at the current position of the rover with a radius of four meters is calculated; see Figure 8b. The intersection point is used as a starting point in search of a suitable triangle in the area, at most one meter from the starting point; see blue circle in Figure 8b. All triangles that have at least one corner less than a meter away from the intersection point are considered and the bigger one that has an acceptable slope is selected; highlighted in dark red in Figure 8b. When the destination way-point is selected, the A* and the rover-footprint cost function are used to plan a local path between the current pose and the destination way-point; Figure 8c.

[2]When the local scan spans 360°, as with the current sensor, only range is considered. When the scan has a limited field of view, see [21], then the global path has to be segmented against the field of view of the sensor.

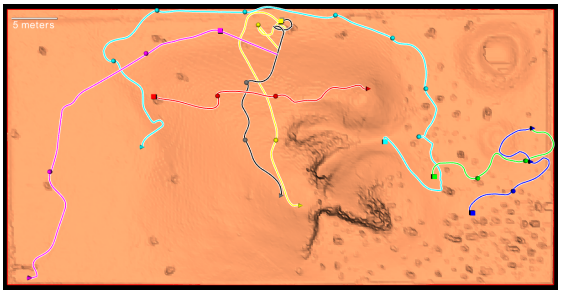
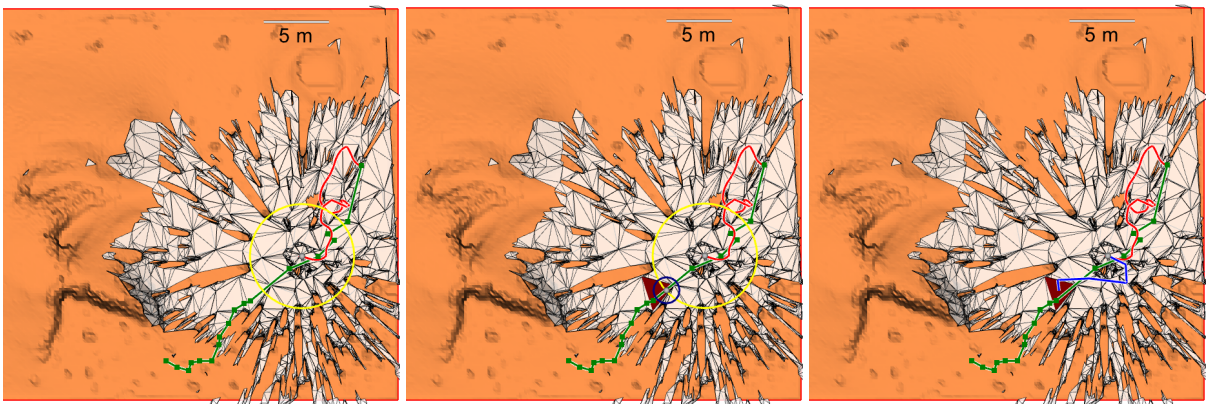


Figure 9. The Mars Emulation Terrain, with all the autonomous over-the-horizon navigation trajectories marked

6 Experimental Results

Over the last year a large number of experiments were conducted at the Mars emulation terrain at CSA. Figure 9 presents an overview of the different autonomous exploration trajectories traversed. As can be seen, the paths span most areas of the terrain, traversing through areas of varying slope and obstacle density.

Figure 10 presents a sequence of snapshots illustrating an autonomous navigation experiment. The first scan is taken and the rover traverses through the smoother area to its left instead of moving straight ahead; see Figure 10a. It was observed that in front of the rover there were many small rocks making the terrain, though traversible, to have high slope variance. As such, the cost function stirred the rover over flatter terrain. During the second scan, the rover traversed through two rocks, moving towards the top first; see Figure 10b. The third scan takes the rover to the target; see Figure 10c. Figure 10d shows a final scan taken at the target for future work on localization. Figure 10e contains the global path (in green) the recorded trajectory (in red) and the pure odometry estimate (in black). It is worth noting that the pure odometry drifts very early on inside the very rocky terrain at the right side.



(a) (b) (c)

Figure 8. (a) The global path (green), the local scan, the trajectory up to this point (red), the current position, and the four meter range the local path planner would use (yellow circle). (b) The intersection point between the global path and the four meter range; the local area from where the destination triangle (red) is selected (blue circle). (c) The local path planned (blue)

7 Conclusions

In this paper we presented in detail the path planning methodology adopted by the Canadian Space Agency. Path planning is a central component to any planetary exploration scheme. The environment representation selected (*ITM*) enables us to abstract the environment in a graph structure in which path planning is equivalent to graph-search. Different cost functions allow for optimizing the path with different objectives. It is worth noting that the topographical information on the terrain is fully encoded in the graph structure via the information in the supporting triangles. The numerous successful experiments conducted over the last year justify our choices.

Different mobility platforms are currently being evaluated by CSA for future missions. The autonomous exploration capabilities presented here would be adapted to fit the future choices. Furthermore, the use of the *ITM* representation is currently considered as a choice for terrestrial and underwater applications at McGill University.

The advantages of active vision for space applications as demonstrated by the successful use of Neptec's[3] Laser Camera System (LCS) sensor on mission (STS-105) and in every other space shuttle mission since then, are numerous. Moreover, LIDAR sensors from OPTECH[4] and MDA[5] have been successfully used in XSS11 rendezvous mission and are part of the NASA Mars Phoenix mission. LIDAR data are raw 3D points which are available without the cost of image processing, thus eliminating delays and possible errors. In planetary exploration in particular, such a sensor provides accurate data independent of the lighting conditions and allows for multiple uses, such as, accurate mapping of a variety of geological formations, use as a science instrument when combined with techniques such as LASER breakdown spectroscopy, and hazard avoidance during entry, descent, and landing. We are confident that the technologies presented in this paper are going to contribute greatly towards the success of future missions.

[3]http://www.neptec.com

[4]http://www.optech.ca

[5]http://sm.mdacorporation.com

References

[1] Kitware inc. visualization toolkits. http://www.vtk.org, Website (accessed: Sep. 2005)., 2005.

[2] P. Allard, J. N. Bakambu, T. Lamarche, I. Rekleitis, and E. Dupuis. Towards autonomous long range navigation. In *8th Int. Symposium on Artificial Intelligence, Robotics & Automation in Space (i-SAIRAS)*, Munich, Germany, Sep. 2005.

[3] J. Bares et al. Ambler: An autonomous rover for planetary exploration. *IEEE Computer*, 22(6):18–26, June 1989.

[4] J. Biesiadecki, C. Leger, and M. Maimone. Tradeoffs between directed and autonomous on the mars exploration rovers. In *Int. Symposium of Robotics Research*, San Francisco, 2005.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[6] A. Kelly et al. Toward reliable off-road autonomous vehicles operating in challenging environments. *The Int. Journal of Robotics Research*, 25(5-6):449–483, June 2006.

[7] D. M. Gaines, T. Estlin, and C. Chouinard. Spatial coverage planning and optimization for a planetary exploration rover. In *5th Int. Workshop on Planning & Scheduling for Space*, 2003.

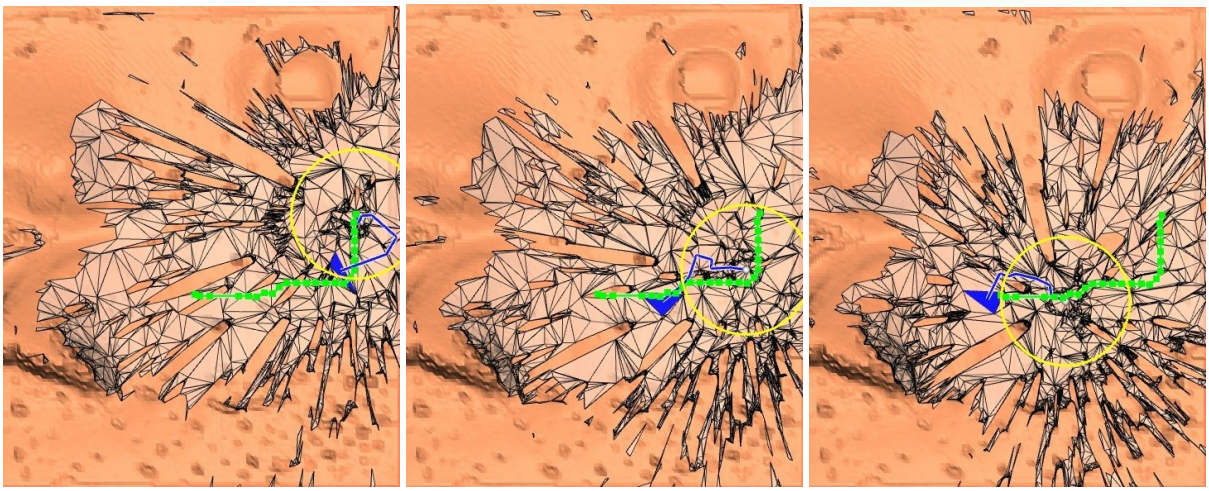
[8] G. Giralt and L. Boissier. The french planetary rover vap: Concept and current developments. In *lEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, v. 2, pp. 1391–1398, 1992.

[9] S. Goldberg, M. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *IEEE Aerospace conference proceedings*, v. 5, pp. 2025–2036, Big Sky, Montana, USA, Mar. 2002.

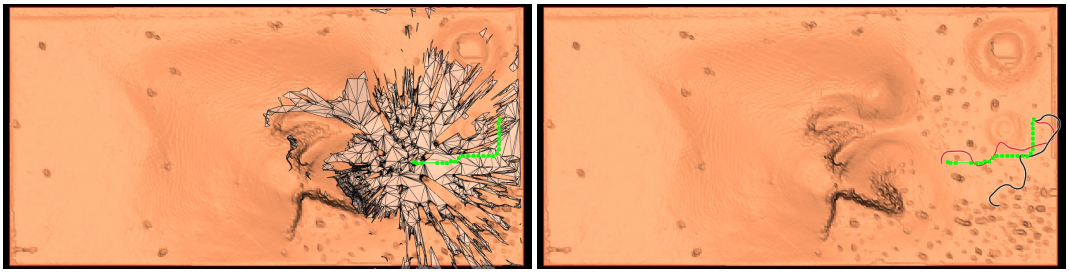
[10] M. Hebert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, v. 2, pp. 997–1002, May 1989.

[11] A. M. Howard and E. W. Tunstel, editors. *Intelligence for Space Robotics*, chapter MER Surface Navigation and Mobility. TSI Press, 2006.

[12] T. Kubota, R. Ejiri, Y. Kunii, and I. Nakatani. Autonomous behavior planning scheme for exploration rover. In *Second IEEE Int. Conf. on Space Mission Challenges for Information Technology (SMC-IT)*, p. 7, 17-20 July 2006.



(a) (b) (c)



(d) (e)

Figure 10. (a)-(d) The Mars emulation terrain model, the global path (green), a local scan (black), range for the local path planning (yellow circle), and the local path planned (blue). Mesh. (e) The Mars emulation terrain, the global path (in green), the recorded trajectory (in red), and the raw odometry (in black)

[13] Y. Kunii, S. Tsuji, and M. Watari. Accuracy improvement of shadow range finder: Srf for 3d surface measurement. In *Procs. off IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, v. 3, pp. 3041 – 3046, 27-31 Oct. 2003.

[14] S. Laubach and J. Burdick. An autonomous sensor-based path-planner for planetary microrovers. In *IEEE Int. Conf. on Robotics & Automation*, v. 1, pp. 347 – 354, May 1999.

[15] C. Lee et al. Software architecture of sensor data distribution in planetary exploration. *IEEE Aerospace Conf.*, p. 9, 2006.

[16] M. Maimone, J. Biesiadecki, E. Tunstel, Y. Cheng, and C. Leger. *Intelligence for Space Robotics*, chapter Surface Navigation and Mobility Intelligence on the Mars Exploration Rovers, pp. 45–69. TSI press, 2006.

[17] L. Matthies and S. Shafer. Error modeling in stereo navigation. *IEEE J. of Robotics & Automation*, (3):239 – 250, 1987.

[18] M. Montemerlo, S. Thrun, H. Dahlkamp, D. Stavens, and S. Strohband. Winning the darpa grand challenge with an ai robot. In *Proc. of the AAAI National Conf. on Artificial Intelligence*, Boston, MA, 2006.

[19] A. Mourikis, N. Trawny, S. Roumeliotis, A. Johnson, and L. Matthies. Vision-aided inertial navigation for precise planetary landing: Analysis and experiments. In *Robotics: Science & Systems Conf. (RSS)*, Atlanta, GA, June 2007.

[20] I. Rekleitis, J.-L. Bedwani, and E. Dupuis. Over-the-horizon, autonomous navigation for planetary exploration. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, pp. 2248–2255, San Diego, CA, USA, Oct. 2007.

[21] I. Rekleitis, J.-L. Bedwani, S. Gemme, and E. Dupuis. Terrain modelling for planetary exploration. In *Computer and Robot Vision (CRV)*, pp. 243–249, Montreal, Quebec, Canada, May 2007.

[22] M. Slack. Navigation templates: mediating qualitative guidance and quantitative control in mobile robots. *IEEE Trans. on Systems, Man & Cybernetics*, 23(2):452 – 466, 1993.

[23] J. Vago. Overview of exomars mission preparation. In *8th ESA Workshop on Advanced Space Technologies for Robotics & Automation*, Noordwijk, The Netherlands, Nov. 2004.

[24] R. Volpe. Rover functional autonomy development for the mars mobile science laboratory. In *IEEE Aerospace Conf.*, Big Sky, MT, USA, 2006.

[25] D. Wettergreen et al. Second experiments in the robotic investigation of life in the atacama desert of chile. In *8th Int. Symposium on Artificial Intelligence, Robotics & Automation in Space*, Sep. 2005.

[26] J. Wright et al. Terrain modelling for in-situ activity planning and rehearsal for the mars exploration rovers. *IEEE Int. Conf. on Systems, Man & Cybernetics*, v. 2, pp. 1372 – 1377, 2005.

[27] L. Yenilmez and H. Temeltas. Autonomous navigation for planetary exploration by a mobile robot. In *Int. Conf. on Recent Advances in Space Technologies (RAST)*, pp. 397– 402, 20-22 Nov. 2003.