Proceedings of the 28th Conference
on Decision and Control
Tampa, Florida • December 1989

A Perception System for a Planetary Explorer

M. Hebert, E. Krotkov, T. Kanade [1]
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

To perform planetary exploration without human supervision, a complete autonomous robot must be able to model its environment and to locate itself while exploring its surroundings. For that purpose, we propose a modular perception system for an autonomous explorer. The perception system maintains a consistent internal representation of the observed terrain from multiple sensor views. The representation can be accessed from other modules through queries. The perception system is intended to be used by the Ambler, a six-legged vehicle being built at CMU. A partial implementation of the system using a range scanner is presented as well as experimental results on a testbed that includes the sensor, one computer controlled leg, and obstacles on a sandy surface.

1 Introduction

The unmanned exploration of planets, such as Mars, requires a high level of autonomy due to the communication delays between a robot and the Earth-based station. This impacts all the components of the system: planning, sensing, and mechanism [6]. In particular, such a level of autonomy can be achieved only if the robot has a perception system that can reliably build and maintain models of the environment. We propose a perception system that is designed for application to autonomous planetary exploration. The perception system is a major part of the development of a complete system that includes planning and mechanism design. The target vehicle is the Ambler, a six-legged walking machine being developed at CMU (Figure 1, [1]).

The perception system can be viewed as an intelligent memory that can be interrogated by external modules (e.g., path planning modules) while maintaining an internal representation of the world built from sensors as the vehicle navigates. The paper addresses the choice of the basic representation maintained by the system in Section 2 and the architecture of the perception system in Section 3. Although the architecture is designed to handle a variety of sensors, we have focused on the use of a laser range finder, since the first requirement for safe navigation of the robot is reliably modeling the geometry of the surrounding terrain. Section 4 describes the algorithms developed for the construction of terrain models from range images. Finally, Section 5 describes the experiments that were conducted to evaluate the perception system.

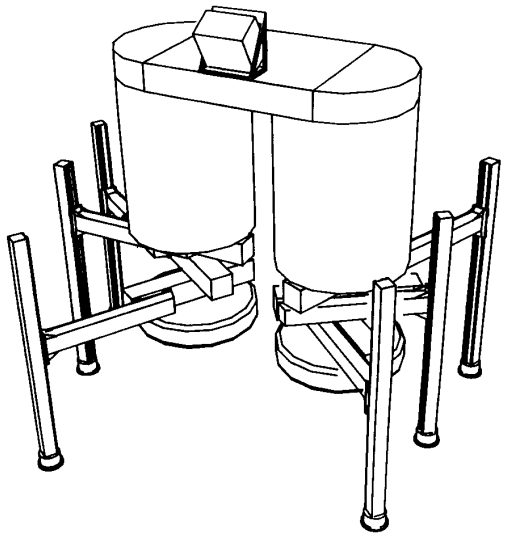

Figure 1: The Ambler

2 Terrain Representation

The basic internal representation used by the perception system is a grid, the local terrain map, each cell of which contains attributes of the terrain. A cell must contain at least the elevation of the terrain and the uncertainty on the elevation due to sensor noise. The uncertainty is modeled as a Gaussian distribution, whose standard deviation σ is stored in the terrain map. Other attributes may include the slope, the surface texture, etc. In addition, attributes that are stored in a cell may be of non-geometric nature, such as the color of the terrain. Several resolutions of the grid may be maintained simultaneously.

On top of the base grid, higher level information can be represented in the form of labeled features in the grid, such as topographic features (hills, ravines, etc.), regions of homogeneous terrain type, objects of interest that have been extracted (boulders, rocks, etc.).

Other terrain representations are possible. The surface could be represented directly by 3-D patches that either are approximations of the measured surface or are built directly upon the set of data points. In both cases, however, retrieving a region of interest from the map becomes a complex operation. Another possibility

[1]This research was sponsored by NASA under Contract NAGW 1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the US Government.

is to represent only a higher-level description of the terrain, such as a segmentation of the surface. This is not appropriate in our case since some planning tasks need information at the lowest level (elevation map). One example for the Ambler is to estimate how stable a foot placement on the terrain would be, in that case a surface description at a resolution that is well below the size of the foot is needed. By contrast with the alternative representations, the terrain map representation as an elevation map is simple to manipulate, can include high-level information as well as high resolution elevation data, and can be accessed by external modules in a simple way by giving the boundary of the region of interest in the map.

3 Architecture

The perception system is divided into six logical modules (Fig. 2). The system communicates with external modules using messages that are routed through a central message handler [4]. The perception system is controlled by a front end—the Local Terrain Map Manager (LTMM)—that receives the messages. Once a message requesting data is received, the LTMM checks whether it is available in the current internal terrain map. If not, then the LTMM instructs the Imaging Sensor Manager (ISM) to take a new image from the relevant sensors, and the terrain map from the new image is merged in the current terrain map. The internal representation is a terrain map built with respect to a fixed reference frame, the *global* frame $\mathcal{G}$. All the operations in the overall Ambler system are expressed with respect to $\mathcal{G}$. A separate module provides the vehicle pose in $\mathcal{G}$. Since the terrain map is of interest only in a region around the vehicle, useless parts of the terrain map are discarded by another module, the Scroller.

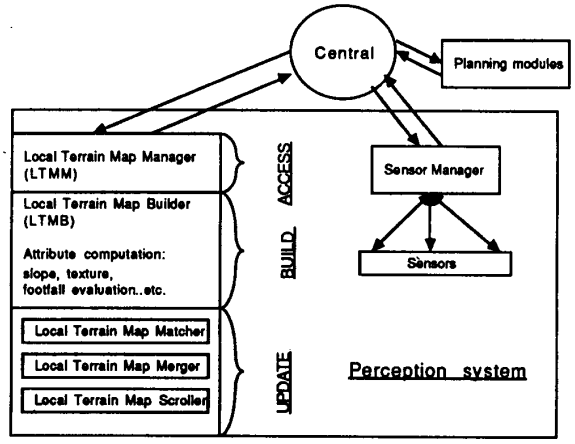

Figure 2: Architecture of the perception system

3.1 Accessing the Perception System

External modules communicate with the perception system by exchanging messages (Fig. 2). Two types of messages are used: *queries* that are request for data, and *replies* that are used for sending data in response to a query. The queries and replies are routed and synchronized by a central module [4]. The simplest example of a query from a planning module would be a request for an elevation map in a given area for clearance checking. The main advantage of this mode of access is that external modules do not need to know the internal workings of the perception system such as the sensor used or the format of the internal representation.

The LTMM is the front end to the perception system, and is responsible for processing queries and for activating the proper submodules. When a query is received, the manager first checks if the area of interest has been already processed at the requested resolution, if that is the case the requested information is extracted from the existing terrain map, otherwise the manager requests a new image from the ISM that is processed and merged with the current terrain map.

To be processed all queries must contain three pieces of information: a polygon that is the boundary of the region of interest; a resolution that indicates at what level of detail the requested calculations must be carried out on the terrain map;[2] and the type of information requested (elevation, uncertainty, slopes, etc.). Because all queries are expressed in $\mathcal{G}$, external modules do not need to know the pose of the sensors. The transformation between a sensor and the vehicle's base frame is stored internally by the perception system, while the current vehicle pose with respect to $\mathcal{G}$ is requested each time a query is received.

3.2 Acquiring Sensor Data

Instead of hardcoding the sensor interface into the LTMM, sensor data is obtained through the same query mechanism. Whenever an image is requested, the requesting module sends a query to the ISM that includes the type of sensor and the type of data desired. The ISM is responsible for activating the requested sensor. The ISM can be viewed as a virtual sensor that hides the details of the sensors' interfaces from the perception system, thus allowing for a more flexible way of changing sensor specifications. Because all queries are expressed in $\mathcal{G}$, the ISM is also responsible for requesting the position of the vehicle with respect to $\mathcal{G}$ from a module that keeps track of the position of the robot either by dead reckoning or by using a navigation system. The other transformation that is needed in order to use the sensor data is the transformation between sensor frame and vehicle frame; this transformation is pre-computed by a calibration procedure and stored by the ISM at initialization time. The composition of those two transformations, that is the transformation between sensor frame and $\mathcal{G}$, is returned to the perception system along with the sensor data (Fig. 3).

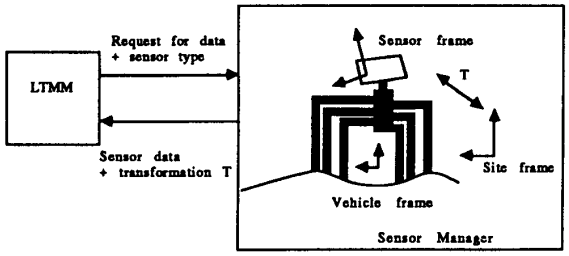

Figure 3: The Imaging Sensor Manager

[2]For example, a resolution of several tens of centimeters is sufficient for checking that the path of the body is clear. Analyzing the stability of one foot of the Ambler requires a resolution of a few centimeters.

3.3 Building the Terrain Map

Once a query is received, a terrain map must be built within the boundary of the region of interest. This is the role of the Builder (Fig. 2) which constructs a terrain map given sensor data taken at one position of the vehicle. The terrain map is computed at the requested resolution and includes the elevation and the uncertainty at each point. In addition to those two attributes, the Builder also implements the algorithms for computing other local attributes such as slope or surface texture as well as non-geometric attributes such as color or terrain type depending on the available sensors. In addition to computing the local attributes, the Builder also identifies the portions of the map that are outside of the fields of view of the sensors, and those that are occluded by parts of the terrain.

The Builder is optimized in several ways: It maintains maps at different resolutions so that it is not necessary to always compute the map at finest resolution. Also, if a map at the desired resolution does not exist, the Builder will create one, thus allowing for arbitrary resolutions. The Builder minimizes the amount of computation by remembering both the regions of the world that have already been computed and by storing the past images so that if a query falls within the field of view of an existing image, it is not necessary to acquire and process a new image.

An implementation of a Builder that uses range images is described in Section 4.

3.4 Updating the Terrain Map

Since the Builder constructs a terrain map from a single image, new sensor data has to be acquired each time a new query is received. This is sufficient as a first approximation, however the perception system should be able to handle terrain maps built from sensor data acquired at different positions. There are two motivations for handling multiple frames. It is obviously more efficient to remember terrain maps built from previous frames rather than recomputing everything at each step. A more compelling motivation is that merging multiple frames may be the only way to provide the requested data. Such a situation occurs when parts of the vehicle, usually a leg, lie within the field of view of the sensor and therefore occludes a part of the terrain map, in that case it may not be possible to extract the region of interest from the current position. A second case in which multiple frames are needed is when data that is outside of the current field of view of the sensor is needed. In the case of the Ambler, this is actually the standard situation since, in the normal walking mode, the leg further behind the body is moved to the front of the body which requires data *behind* the body so that the path of the leg can be checked for clearance. For these reasons, the perception system must include the capability to merge terrain maps from successive frames into a single terrain map.

The responsibility for the management of multiple terrain maps is shared by two modules, the Matcher and the Merger (Fig. 2). The Matcher estimates the displacement between a new terrain map and the current internal terrain map. The displacement is in general a 3-D transformation. It is estimated by matching features extracted from the maps, or by using a correlation technique that compares the two maps directly. Section 4 briefly describes an implementation of the latter in the case of terrain maps built from range images. An initial estimate of the displacement is always available either from dead reckoning or from a navigation system. Once the displacement is computed, the Merger is responsible for merging the new map into the current map. Actually, only the part of the map that is within the requested region of interest is actually merged for efficiency reasons. The maps are merged by combining the elevation values at each location of the map using the uncertainty values to obtain the maximum likelihood estimate of the elevation. The Merger must also update the occluded areas of the current map.

It is important to logically separate the matching and merging operations. First of all, the matching operation may or may not be necessary depending on the accuracy of the positioning system. If a navigation system provides displacement estimates that are well below the resolution of the grids, the estimates will not be improved by terrain matching. Second, if raw sensor data is stored along with past terrain maps, a new query requires only merging portions of the terrain maps since the displacements have already been computed at the time the images were acquired. Finally, separating the two modules allows for experimenting with different matching algorithms, presumably the most difficult part of the system, while retaining the same structure for the rest of the system.

Since the terrain map must grow as the vehicle moves and as new sensor data is acquired, a third module, the Scroller, is responsible for discarding the part of the map that is too far from the vehicle to be useful. This can be viewed as sliding a window centered on the current position of the vehicle; only data within this window is retained. The Scroller is motivated both by the need to prevent the size of the terrain map from expanding during the course of a long mission, with the risk of memory overflow, and by the fact that only the most recent terrain maps can be used with confidence due to the accumulation of errors in the displacement estimates between maps.

4 Elevation Maps from Range Data

The perception system is designed to use multiple sources of data. Because geometric information is most important for local navigation, we consider in this section the case of data from an active range scanner.

We use the Erim laser scanner, which delivers 64×256 range images by measuring the phase difference between a laser beam and its reflection from a point in the scene [7]. The scanner measures the range ρ in a spherical coordinate system in which ϕ and θ are the vertical and horizontal scanning angles, corresponding to row and column positions in the image.

Prior to operation, the position of the sensor with respect to the vehicle's coordinate frame must be computed. This is done by a calibration procedure that computes the position by observing markings on the leg using the range scanner at different known positions of the leg. A least-squares estimation algorithm estimates the transformation between the coordinate system of the scanner and the coordinate system of the vehicle. This transformation is compounded with the transformation between vehicle and global frames by the ISM each time a new image is acquired.

The easiest way to convert the range images to elevations maps is to convert each pixel (ρ, θ, ϕ) to a point in space (x, y, z), which is straightforward knowing the geometry of the sensor and the transformation between sensor and global frames. This approach has some severe drawbacks, however, such as the need for interpolation, the dependency on a particular coordinate system, and the fact that it is not possible to limit the computation to a region of the terrain map because we do not know apriori where this region is in the image. Instead, we use the locus method described in [3]. This approach has many advantages including the explicit detection of range shadows, the representation of uncertainty, the independence of the algorithm with respect to a reference frame,

a straightforward extension to the case of multiple frames of data. Furthermore, a major feature of the locus method is its ability to limit the computation of the terrain map to any region in space, thus facilitating the computation of the maps within the boundaries of the queries of Section 3.1.

The terrain map building algorithms were evaluated on range images taken by the Erim scanner. The test images were taken in a construction site that exhibits the type of rugged terrain that we are interested in. Fig. 4 shows a map built from one range image using the locus algorithm. The resolution is 10 cm over a 10×10 m square.

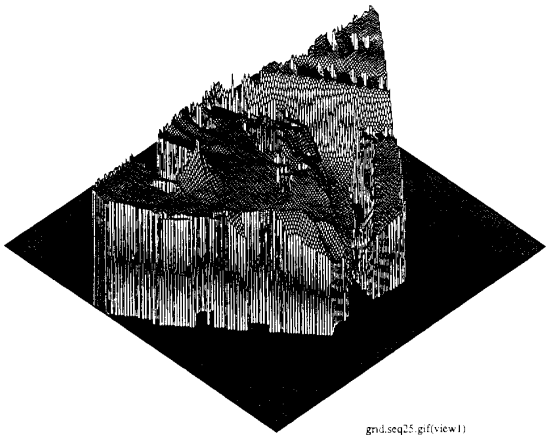

Figure 4: Elevation map built from one range image

An extension of the locus algorithm allows for matching and merging terrain maps built from images taken from different positions. The matching algorithm computes the best 3-D transformation between maps, while the merging algorithm computes optimal combination of the elevation from the two maps given this transformation. The map building from multiple frames was tested on sequences of Erim images as well as on synthesized images. Fig. 5 shows the terrain map obtained by merging data from four successive range images. The resulting terrain map is about thirty meters long. In this example the images were collected along a general path including a sharp turn (about 30°). The matching between consecutive terrain maps was performed by first matching features to obtain a first estimate of the transformation [3], and by using the estimate as a starting point for the minimization of the difference between the two terrain maps that is used as the final transformation for the merging. Experiments on synthesized images for which the transformation between images is known show that the error on the resulting transformation can be as small as the resolution of the grid. The error in elevation is of the order of a few centimeters, increasing with the uncertainty as the points are further away from the sensor.

These experiments show that the algorithms developed for range data provide the type of terrain maps required for rugged, unstructured environments including variable resolution, arbitrary reference frame, explicit uncertainty representation, and representation of occluded areas.

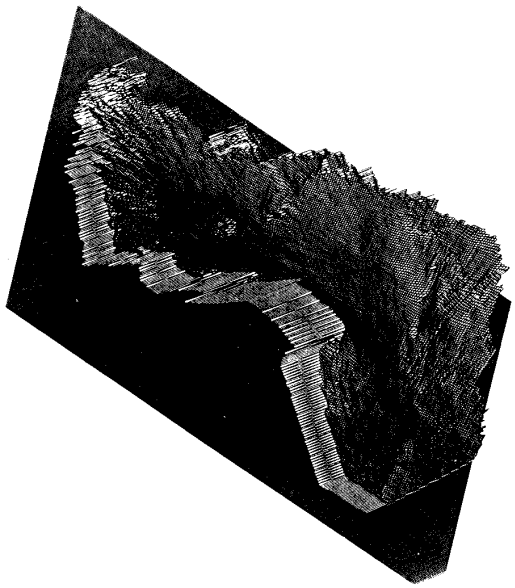

Figure 5: Elevation map from 125 range images

5 Experimentation

A first version of the perception system of Fig. 2 is implemented. This version includes the LTMM, LTM Builder, and ISM. This implementation includes the algorithms of Section 4 and uses the Erim scanner. This implementation of the system is used to validate the interface on single range images. The system builds terrain maps with currently two attributes: uncertainty and footfall evaluation. The latter is a measure of how good a footfall each location in the map would be, based on the local shape of the terrain. The algorithms used for the footfall evaluation are described in [2].

Three types of queries are currently recognized:

- Elevation map: This is a request for an elevation map within a given region (polygon) with a given resolution.
- Elevation and uncertainty map: This is basically the same query except that the uncertainty at each point of the terrain map is returned as well.
- Footfall evaluation: This is a request for the best position of the foot within a region. Currently this request is processed by computing the stability of a circular foot at each point of the terrain map by using only the geometry of the terrain [2].

Fig. 6 shows the result of processing a footfall evaluation query. The lower left view displays an overhead view of the site with the region of interested displayed as a shaded polygon. The three other views are the map computed from a range image. The lower right map is a map of the footfall evaluation in which the highest values correspond to the best footfall locations. The dimensions on the lower left diagram are in meters. The resolution of this query is 10 cm.

A testbed was built in order to test the fully integrated planning/perception/mechanism system. The testbed (Fig. 7) includes a single leg, the range finder mounted on top of the "body" of the vehicle, and a $25m^2$ sandbox that simulates the terrain in which

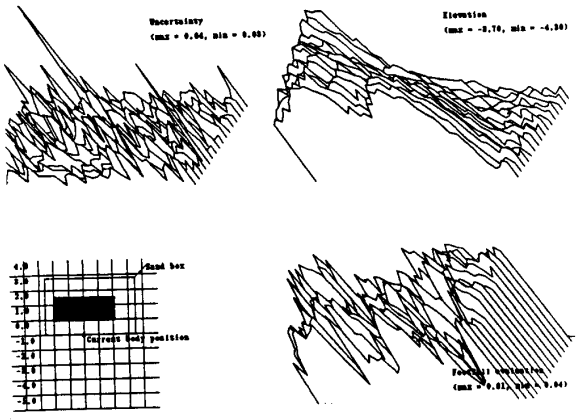

Figure 6: Processing a footfall query

the rover will navigate. The testbed leg was built from an earlier design, as a result it is slightly different from the legs in Figure 1. The main difference is that the testbed leg uses rotational joints while the design of Figure 1 uses prismatic joints. A real-time controller drives the leg to specified locations in Cartesian or joint space, allowing for constraints on the velocity of the leg and the forces applied to the foot. In addition, the body can be translated along two parallel rails by controlling the two horizontal joints of the leg while keeping the foot on the ground, thus simulating the motion of the body in the actual rover. The testbed is equipped with a linear position sensor and two clinometers that together give an estimate of the position and orientation of the rover with respect to the global frame.

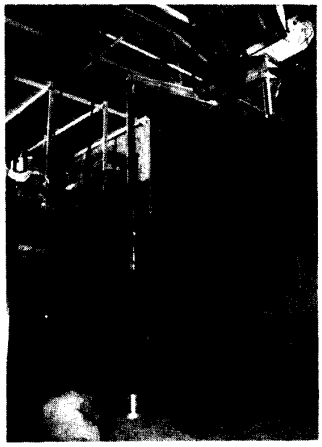

Figure 7: The single leg testbed

The most complicated task that is used in order to test the perception system as part of the complete single leg testbed is the so-called "move-body" task in which, given a desired length of travel and a desired step length, the leg takes a series of steps, pulling the body forward after each step. The locations of the footfalls as well as the trajectory of the leg are computed using the terrain maps from the perception system.

For each step, the sequence of operations is as follows.

1. A region in which the foot may be placed to achieve the next step is computed by the gait planner module.
2. The gait planning module queries the perception system for the best footfall position within this region. This query activates the whole cycle of taking an image, computing the terrain map, computing the footfall evaluation attribute, and replying.
3. Given the footfall position, the leg recovery planning module computes a path for the leg and sends a region around this path to the perception system requesting a map.
4. Perception answers the query by sending back a map within the specified region, including the uncertainty attribute.
5. The planning module uses the map to compute the locations of intermediate points along the path of the leg. The leg is moved to the goal location and the foot is lowered onto the terrain using position control. The uncertainty is used as a safety margin both for the travel of the leg and for the actual footfall. In the latter case the foot is lowered to a position that is 2σ above the nominal value reported in the terrain map, and then lowered using force control until it contacts the soil.

Repeated experiments with the "move-body" scenario with different terrain shapes and different initial and goal configurations of the leg have shown conclusively that the first version of the perception system performs reliably and allow the system to safely walk around obstacles.

Several lessons were learned during these experiments. Good calibration between the sensor and the leg is essential for computing reliable elevation value in the vehicle's reference frame. It is important to use information already extracted when possible, if an image is taken whenever a query is received we then run the risk to have the leg in the field of view of the sensor occluding the region of interest. The solution to this problem is to include in the perception system the algorithms that extract the relevant information from the existing terrain map before acquiring new data. Finally, it is clear from those experiments that more development is needed as far as the computation of attributes is concerned. The only attributes are currently the footfall evaluation and the uncertainty.

6 Discussion

We have presented a perception system for an autonomous vehicle designed for planetary exploration. The perception system uses terrain maps as the basic internal representation that is accessed by external modules. Parts of the system have been demonstrated using algorithms for building terrain maps from range images. The current version of the perception system has been included in a complete single leg testbed.

6.1 Improvements

Several improvements are needed in the current system. First, calibration is of critical importance for the successful operation of the overall system. We therefore need to improve the calibration procedure to the point at which the errors due to miscalibration are minimal compared to the other sources of errors. This involves in

particular a more detailed analysis of the geometry of the mechanism, a more accurate model of the sensor, and more reliable algorithms for the detection of calibration targets.

The quality and accuracy of the terrain maps may also be improved. In particular, the uncertainty model may reflect the actual environment by using a more detailed model of the sensor measurements. Another improvement is the extensive use of map merging to produce more accurate maps by combining many measurements at each point in the map.

The last improvement is in the area of exception handling. Currently, the perception system cannot recover gracefully from errors such as corrupted sensor data, bad transformation from corrupted position readings, or bad message handling. In order to have a robust system we need to design a mechanism to detect and recover from these conditions.

6.2 Extensions

Further work is required to demonstrate a perception system that can handle the tasks of a complete autonomous system. Other sensors must be used in conjunction with the laser range finder in order to compute non-geometric types of information such as the type of the terrain in a region. This is important both for sampling tasks, which require the identification of specific types of terrain, and for the evaluation of footfall selection since the soil compliance depends on the type of terrain. The best candidates are color cameras and thermal cameras. We are working on integrating those sensors into the perception system.

Other sensors that should be added to the perception system include sensors for short-range perception, such as proximity sensors. Those sensors would be used in the final phase of the footfall to provide better control of the foot contact with the terrain. Currently, the foot is lowered to a nominal value given by the elevation map, after which point it is slowly lowered until a given force reaction is observed. This is a potentially dangerous approach if the map is inaccurate at that point, or if the terrain has changed between the time the map was built and the time the foot is moved. A proximity sensor would guarantee that the foot does not attempt to penetrate the ground.

The perception system uses only local information from its sensors. A possible extension would be the addition of more global information such as a large-scale map from an orbiter. The main issue is then to establish the relationship between the low-resolution global map and the high-resolution local maps. This is essentially a matching capability that can greatly enhance the performances of the rover. For instance, the rover could register itself with respect to large-scale terrain features from the global map.

Finally, we must complete the inclusion of the matching of multiple frames in the system. Map matching will give the rover a "self-localization" capability, that is the ability to register itself with respect to its environment without relying entirely on special-purpose position sensors (clinometers, dead reckoning, INS). It has been our experience that those sources of position information cannot be relied upon at all time because they do not necessarily give an accurate description of the position and orientation of the sensor at the time an image is taken. Furthermore, they have to be carefully calibrated with respect to the perception sensors which add another level of complexity to the already difficult calibration problem. The solution will be to use the output of the position sensors as an initial estimate for the map matching process which will provide the accurate position estimate actually used.

6.3 Remaining Issues and Lessons Learned

In the course of developing this system we have encountered the usual fundamental issues in the design of autonomous systems [5], and had to make choices to overcome those problems. Two issues were of special interest: the mode of synchronization between the perception system and the other modules, and the limitations due to message-passing between modules.

The architecture is currently entirely query-driven in that the terrain maps are computed only in response to a specific query from another module. This may not be the best strategy in a system that includes many other computation-intensive modules. In that case, the perception system would be idle most of the time. A different strategy would be for the perception system to keep computing the terrain map around the vehicle even if no query has been received. That way, the perception would take advantage of the idle time to perform some additional computations. The main issue is for the perception system to be able to predict the regions of the environment that will be "useful" to compute for the future queries. This also requires a careful analysis of the synchronization between modules so that this self-driven approach does not accidentally slow down the other modules.

Our experience with this system has been that the communication bandwidth using conventional network technology is not a limitation. In this application, shipping images and maps between the different modules of the perception system and the other modules does not affect the performance of the overall system significantly. There are still some synchronization issues to be addressed, however. The most important one is to guarantee that the position of the vehicle is correctly read at the time that an image is taken (Section 3.2), which is not possible if there is too much of a delay between the ISM and the module that sends the vehicle position. One solution is to bypass the central message handler completely for some of the low-level operations such taking an image so that the communications are performed by direct memory transfer with minimal delay.

References

[1] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. Ambler: An Autonomous Rover for Planetary Exploration. *IEEE Computer*, 18–26, June 1989.

[2] C. Caillas, M. Hebert, E. Krotkov, I. S. Kweon, and T. Kanade. Methods for Identifying Footfall Positions for a Legged Robot. In *Proc. IEEE International Workshop on Intelligent Robots and Systems*, Tsukuba, Japan, September 1989.

[3] M. Hebert, T. Kanade, and I. Kweon. *3-D Vision Techniques for Autonomous Vehicles*. Technical Report CMU-RI-TR-88-12, The Robotics Institute, Carnegie Mellon University, 1988.

[4] R. Simmons and T. Mitchell. A Task Control Architecture for Mobile Robots. In *Proc. AAAI Spring Symposium*, Stanford, California, March 1989.

[5] T. Stentz and C. Thorpe. Against Complex Architecture. In *Proc. AAAI Spring Symposium*, Stanford, California, March 1989.

[6] B. Wilcox. Session 3: Planetary Rovers. In *Proc. SPIE Vol. 1007: Mobile Robots III*, 1988.

[7] D. Zuk, F. Pont, R. Franklin, and V. Larrowe. *A System for Autonomous Land Navigation*. Technical Report IR-85-540, Environmental Research Institute of Michigan, Ann Arbor, Michigan, 1985.