Risk-aware Planetary Rover Operation: Autonomous Terrain Classification and Path Planning

Masahiro Ono, Thoams J. Fuchs, Amanda Steffy, Mark Maimone, and Jeng Yen
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109
{Masahiro.Ono, Thomas.Fuchs, Amanda.C.Steffy, Mark.W.Maimone, Jeng.Yen}@jpl.nasa.gov

***Abstract*—Identifying and avoiding terrain hazards (e.g., soft soil and pointy embedded rocks) are crucial for the safety of planetary rovers. This paper presents a newly developed ground-based Mars rover operation tool that mitigates risks from terrain by automatically identifying hazards on the terrain, evaluating their risks, and suggesting operators safe paths options that avoids potential risks while achieving specified goals. The tool will bring benefits to rover operations by reducing operation cost, by reducing cognitive load of rover operators, by preventing human errors, and most importantly, by significantly reducing the risk of the loss of rovers.**

The risk-aware rover operation tool is built upon two technologies. The first technology is a machine learning-based terrain classification that is capable of identifying potential hazards, such as pointy rocks and soft terrains, from images. The second technology is a risk-aware path planner based on rapidly-exploring random graph (RRG) and the A* search algorithms, which is capable of avoiding hazards identified by the terrain classifier with explicitly considering wheel placement. We demonstrate the integrated capability of the proposed risk-aware rover operation tool by using the images taken by the Curiosity rover.

TABLE OF CONTENTS

1 INTRODUCTION 1
2 OVERVIEW OF MARS ROVER OPERATION 2
3 TERRAIN TYPES 3
4 TERRAIN CLASSIFICATION 3
5 PATH PLANNER 5
6 CONCLUSION 7
ACKNOWLEDGMENTS 9
REFERENCES 9
BIOGRAPHY 9

1. INTRODUCTION

The greatest single source of risk for Mars rovers is terrain. For example, the Spirit rover ended its mission because it got stuck in a soft terrain as in the left picture in Figure 1. For another example, the Mars Science Laboratory (MSL) rover Curiosity has experienced unexpectedly high damage rate of wheels, particularly on Sols 450-515. The right picture in Figure 1 shows a puncture on a wheel. The MSL Wheel Wear Tiger Team identified that the period of highest damage accrual occurred when the rover was driving over angular, embedded rocks; it also found that rocks on hard terrain are more likely to cause damages on the wheels. Such terrain hazards can only be identified visually; existing on-board geometric hazard detection method cannot tell terrain type and embeddedness of a rock. Currently, risks are managed by a labor-intensive process in which rover operators carefully examine the terrain and plan a path to avoid any potential hazards. This poses a challenge, particularly for the operation of an MSL-class rover, because, on one hand it must be very risk-averse in order not to lose the precious rover, while on the other hand it already requires a significant amount of labor due to its complexity.

978-1-4799-5380-6/15/$31.00 ©2015 IEEE.

Figure 1. **Left:** MER rover Spirit's wheel embedded in soft soil. **Right:** a puncture on a wheel of MSL rover Curiosity. As shown in this examples, major risks to planetary rovers come from terrain.

In order to overcome this challenge, we develop a ground-based software tool, namely the Risk-aware Mars Rover Operation Tool, which collaboratively works with rover planners during operation to help detecting and avoiding risks more efficiently and reliably. To achieve this objective, we identify that the Tool must have following two capabilities:

- Vision-based terrain classification capability to reliably identify terrain types as well as characteristics of rocks, such as pointiness and embeddeness
- Risk-aware path planning capability to suggest safe paths in consideration of terrain types, slopes, and positive and negative obstacles.

We note that these capabilities can also be used for the future enhancement of on-board autonomous navigation software in order to enable Mars rovers to traverse more difficult terrain types. This paper reports the successful development of the two capabilities as well as the prototype of the Tool.

Figure 2 gives an overview of the Risk-aware Mars Rover Operation Tool. The inputs to the Tool are raw camera images and a digital elevation map (DEM). On the images, the terrain classifier visually identifies terrain types and characteristics (e.g., sand, bedrock, embedded pointy rock, etc.). From the DEM, positive and negative obstacles are geometrically identified. It is also used to generate a slope map, which is used as a part of the cost function in the path planning. Finally, the terrain types, the obstacles, and the cost map are

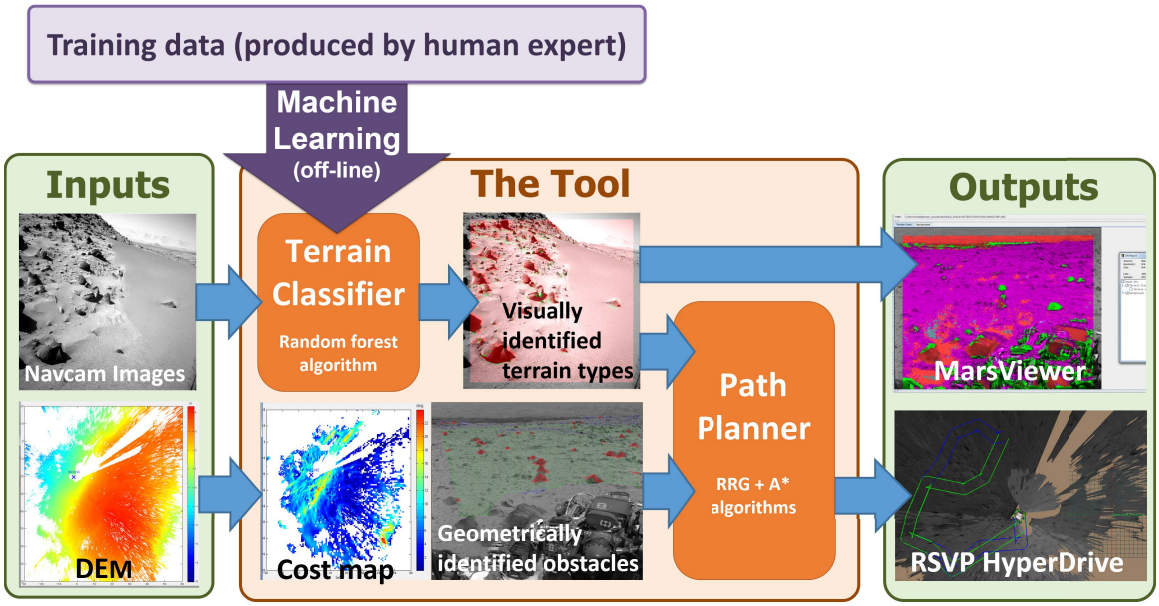


Figure 2. Overview of the Risk-aware Mars Rover Operation Tool

used by the path planner, which interacts with rover planners to make suggestions of safe paths. The outputs from the Tool are terrain classes and suggested paths. The outputs are in a format that can be imported by MarsViewer and RSVP HyperDrive [1], the software that are currently used for MSL and MER operations.

We build the terrain classifier using a machine learning approach in order to capture human expert's knowledge and incrementally improve performance as a mission accumulates data. First, human experts generate a training data set, which consists of images with labels specifying terrain types. We then run the random forest algorithm [2], [3] on the training set to automatically build a classifier. A trained classifier takes an image as an input, and performs pixel-by-pixel identification of terrain types on the images.

Our path planner features the capability of explicitly considering wheel placement. When the existing on-board path planner on the Mars rovers computes a path, it dilates obstacles by the radius of the envelop of the rover to avoid collision [4]. This approach is not applicable to our problem because the terrain classifier often finds densely populated small rocks. By dilating all of them, nearly the whole surface may become untraversable, as in Figure 3-(a). This issue is address by planning a path with wheel placement, as in Figure 3-(b). Our path planner is built upon the rapidly-exploring random graph (RRG) algorithm [5], an extension to the celebrated RRT algorithm. Unlike the regular RRG, when adding an edge to the graph, we check that the wheel tracks are obstacle free. After constructing a graph, the A* algorithm is used to compute a path to achieve user-specified waypoints and goals. The A* algorithm is a search algorithm that is commonly used for graph-based algorithm. When the state space is finite, with an admissible heuristic (e.g., line distance) the algorithm is guaranteed to find the best path if a feasible path exists.

The rest of the paper is organized as follows: In Section 2, we review the Mars rover operation process in order to identify the challenges for operation. Then, in Section 3, we identify terrain types that need to be classified in order to operate Mars

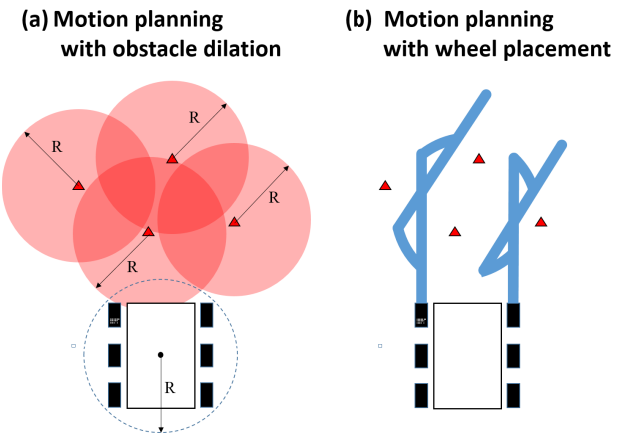


Figure 3. The terrain classifier often finds densely populated small hazardous rocks, as in the right two figures in Figure 7. (a) In such a situation, existing motion planner that dilates the obstacles does not work since it leaves no traversable space. (b) Therefore, the proposed tool performs motion planning with wheel placement.

rovers safely. In Sections 4 and 5, we describe our technical approach to build the terrain classifier and the path planner, respectively. These sections also present the demonstration of the capabilities using real MSL data. Finally, Section 6 concludes the paper.

2. Overview of Mars Rover Operation

The Risk-aware Mars Rover Operation Tool is designed to support actual Mars rover operations. This section outlines the drive planning aspect of MSL operations.

MSL is operated in a Marsian-daily (sol) cycle. On a sol on which rover makes a traverse, a drive sol, rover planners (RPs) first carefully review the data acquired at the end of the previous sol's drive. In particular, the rover captures stereo

pairs of panorama images using its cameras. These images are used in two ways. First, the RPs collaborate with Surface Properties Scientists (SPSs) to visually identify hazards in the images. Second, the images are used to produce a digital elevation map (DEM), a 3D reconstruction of the terrain, through stereo image processing. The DEM is used by RPs to identify obstacles.

RPs then plan a traverse path. A path consists of a sequence of *arcs* and turns. An arc can be either a straight line or a curve. Often a path is required to go through multiple waypoints to perform scientific observations or to ensure safety. Figure 4 shows an example of a planned path projected on a DEM.

There are various safety requirements that must be respected when planning a path. The rover must avoid driving over any potential hazards to its wheels, particularly angular embedded rocks. The rover has the ability to drive over large obstacles, but in the interest of rover safety, rocks larger than 30cm are avoided, and rocks larger than 10-15cm are usually avoided as well. To protect the wheels, the amount of turning during a drive is minimized and the rover is driven either forwards or backwards to keep the number of turns low. The rover planners also consider the slope of the terrain and keep within the distance limits of the current navigation imagery.

In order to meet all the safety requirements, rover operation is inevitably a labor-intensive process; in order to plan a path for a sol, which is often 30 m - 70 m long, it typically takes three people working for 8-10 hours. The new Risk-aware Rover Operation Tool will bring benefits to Mars rover missions by reducing operation cost, by reducing cognitive load of rover operators, by preventing human errors, and most importantly, by significantly reducing the risk of the loss of rovers.

3. TERRAIN TYPES

As shown in Figure 5, we identified the following five terrain classes that need to be distinguished in order to operate a rover safely: sand, bedrock, loose rock, embedded pointy rock and embedded round rock.

These five categories encompass a majority of the terrain seen during the rover drives. The embedded pointy rock was the primary terrain during the period of highest damage accrual on Curiositys wheels (Sol 450-515). This type of rock cannot move as the rover drives over it, applying a point load high enough to puncture the skin sections of the wheels. The embedded round rock does not apply as high of a point load, but can stress the grousers and cause crack propagation. The loose rock, especially when sitting amongst sand, can be pushed into the sand or out of the drive path by the wheels when the rover drives over the rock, making it a less hazardous terrain. When sitting on bedrock, the loose rock cannot be pushed into the ground, which can induce wheel damage, depending on the size and geometry of the rock. The hardness of the bedrock can also induce stress concentration cracking at the chevrons of the grousers. Sand has been considered the most benign of the terrain types for Curiosity, and testing proved that even the pressure of the sand is not great enough to cause crack growth. On the other hand, for MER-class rovers, sand can be the greatest danger, as is evident from the fact that Spirit ended its mission because it was immobilized by sand.

When planning a path for a Mars rover, one or multiple of these terrain classes are carefully assessed to determine the drive path. Therefore, the capability to classify these five types of terrain is essential to achieve our project goal.

4. TERRAIN CLASSIFICATION

Overview

The role of the terrain classifier is to take an image as input, and classify every pixel in the image into one of the five categories defined above. Figure 7 shows sample outputs from the classifier. In this project we use MSL NAVCAM images; however, by retraining the classifier, the same system can be applied to learn models for any form of image type, including the ones from HAZCAM and MASTCAM, as well as HiRISE imagery.

Training Data

Recall that our approach for training a reliable terrain classifier is to learn from human experts using a machine learning algorithm (which is described in detail in the next subsection). The training data is a set of NAVCAM images from MSL which were labeled manually using a web-based annotation tool tailored for Martian imagery. A domain expert used the labeling tool to label pixels with one of five terrain categories defined in the previous subsection. In total a set of 66 images were labeled by Co-I Steffy, who is a member of the MSL Operations team as well as the MSL Wheel Wear Tiger Team. In the Tiger Team, she has collaborated on a surface terrain classification scheme for wheel hazards, the development of a terrain-simulated wheel test track, and has been involved in drive planning and MSL rover operation. Figure 6 shows a subset of the training data.

Technical Approach

We formulated the task of terrain understanding as semantic segmentation problem. To this end we train a random forest classifier [2], [3] based on manually labeled training data which then is used to predict the class label of every pixel in an image and as such differentiates between various terrain types in an image.

Feature Descriptors— The gray-scale intensity of a single pixel in a NAVCAM image is not enough to classify it into one of the five classes of interest. To overcome this limitation each pixel is not only described by its own intensity but also by features of its local surrounding. This provides texture information and allows the system to capture statistical properties of the local context. Specifically we generate a set of channels derived from the original NAVCAM images as basis for feature extraction. Besides gray intensity we use image gradients and range information, which was derived from the NAVCAM stereo pair and is available as image product in NASA's planetary data system (PDS). From these channels a variety of features can be extracted for describing a pixel of interest: channel intensity, intensity at an x and y offset from the pixel [6] and averages of rectangles at random positions in the local context of the point interest. The classifier is then able to learn thresholds on the evaluated responses of these features or on the difference of features. In addition we implemented a strategy from computational pathology [7] which allows to generate Boolean responses by computing only the relation between features [8]. This results in very fast training of diverse trees which can improve the power of an ensemble. The classifier described in the next section is able to choose meaningful combinations of channels, features and evaluation methods to predict the class of pixels.

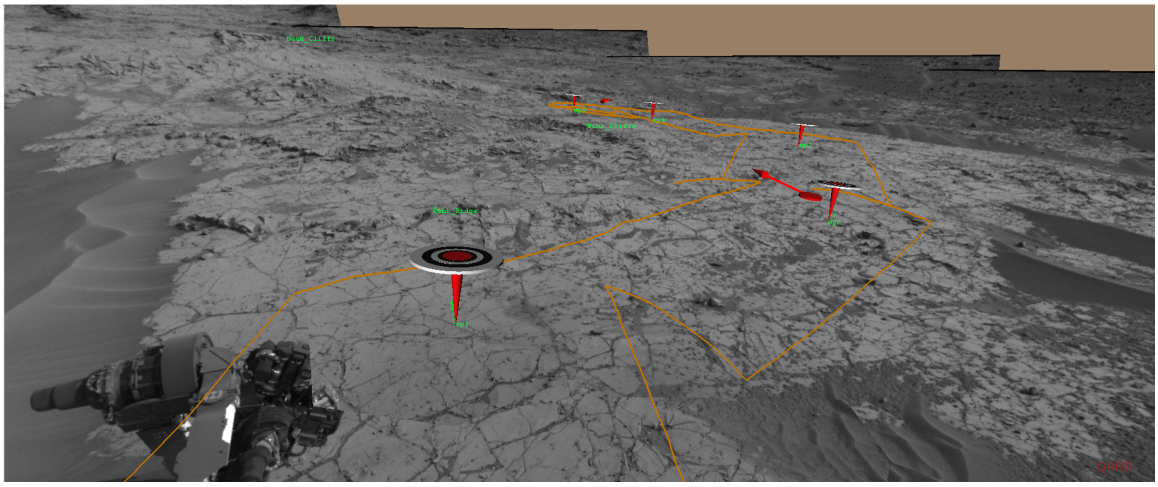


Figure 4. Example of a path planned for Curiosity on Sol 780.

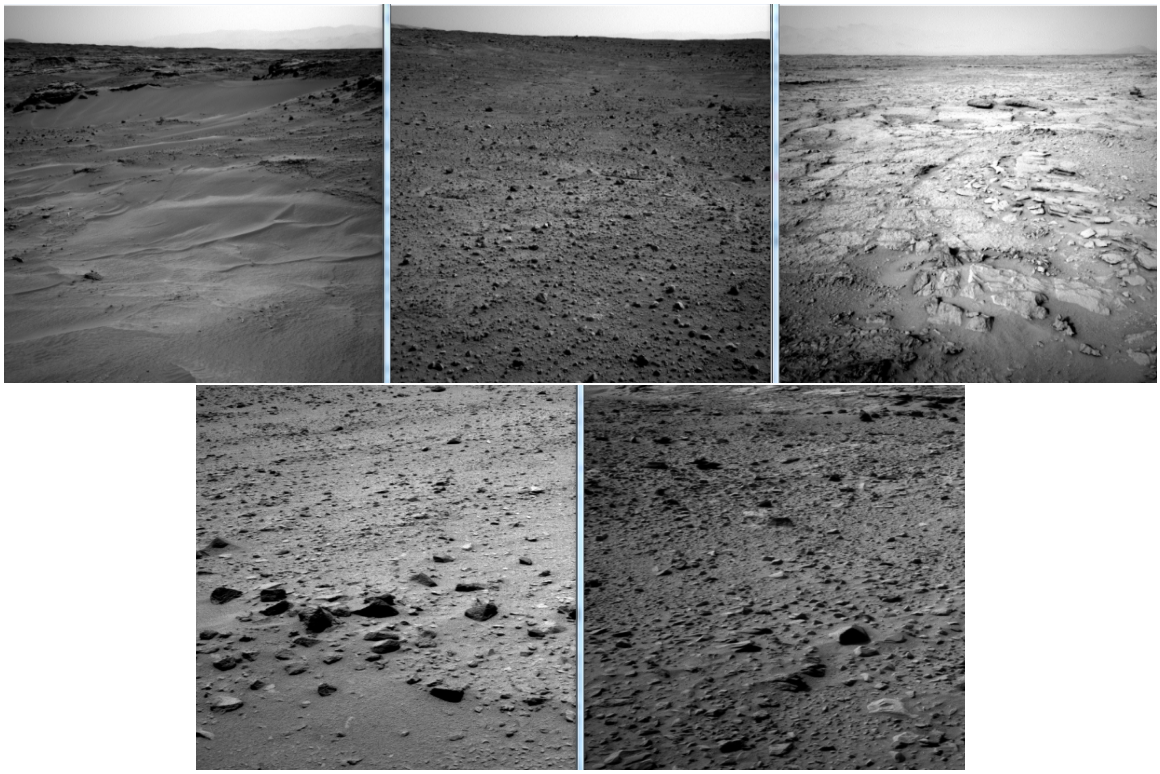


Figure 5. The Five Terrain Types. Top: Sand, Loose Rock, Bedrock. Bottom: Angular Embedded Rock, Round Embedded Rock.

Model Training—Random forests have a number of properties which make them a suitable choice for this task: (i) They can model non-linear interactions between features and hence are able to construct complex models necessary for highly accurate terrain classification. (ii) Random forests implicitly perform feature selection and thus can deal with a large number of statistical features while being robust against noisy or non-informative variables. (iii) The ensemble structure favors parallel training of the decision trees in a distributed manner which allows handling of large amounts of training data in a reasonable time frame. (iv) Random forests can not only be learned but also tested in parallel which results in fast execution speed. Besides graphical processing unit (GPU) implementations [9], field-programmable gate array (FPGA) implementations are already available for space exploration [10] and hence make random forests an ideal choice for the robotic exploration of Mars.

Specifically we train a random forest model comprising 50 binary decision trees. Each tree is learned from a bootstrap [11] of the training data. At every split node the best feature out of 500 randomly sampled features is chosen by maximized the information gain [**?**] over all possible thresholds on a feature. The trees are grown until a predefined depth of 9 and at each leaf node the class histogram of samples reaching this node is stored in the model. During test time, the final prediction is achieved by taking the average of the approximated posteriors, represented as the class histograms

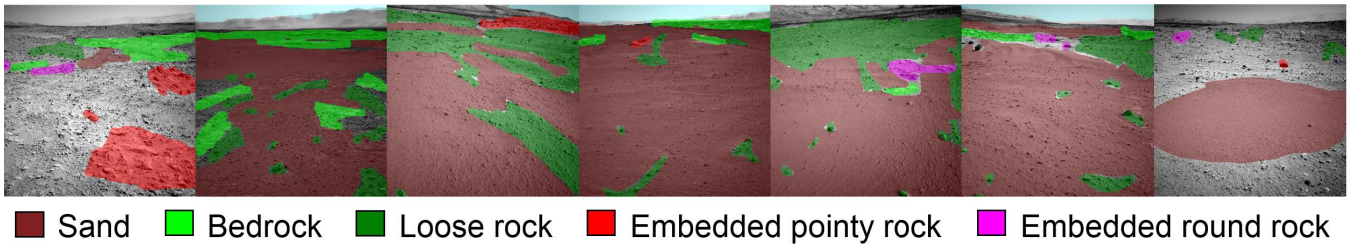


Figure 6. Examples of manual annotations in the training data generated by a human domain expert. In total, 66 images were labeled and used to train the terrain classifier.

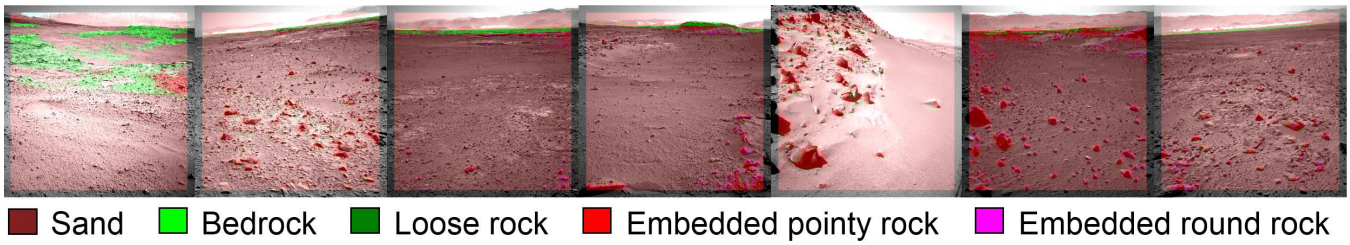


Figure 7. Example predictions of the classification system. Depicted are the classification of frames from the training set. These results show, that the feature set is expressive enough to accurately describe the difference between terrain classes.

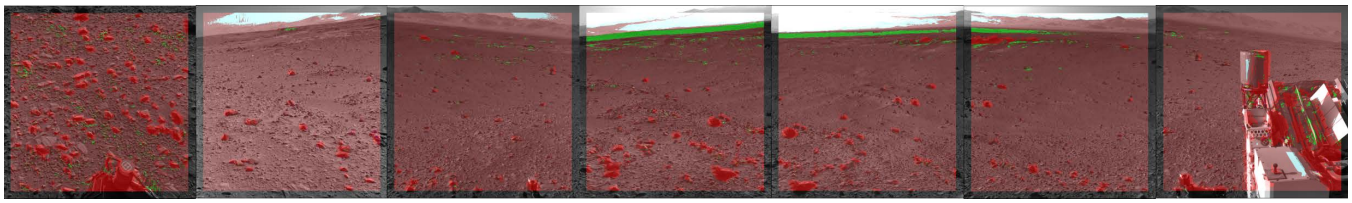


Figure 8. Example predictions of the classification system on panorama images on Sol 504. These images are not included in the training set.

in the tree leafs, over all trees in the ensemble. The maximum in this histogram represents the final class label which is assigned to a pixel. When a new image is classified every pixel in the image is classified by all trees of the ensemble. Finally, the inverse of the entropy of the ensemble posterior histogram can be interpreted as the confidence of the classifier in its overall prediction. This confidence can then be used by the path planner to stratify the classification and use only predictions in which the classifier has high confidence. Alternatively, class specific confidence estimates can be used to handle high risk and low risk classes differently.

Performance Evaluation

The proposed framework is evaluated by conducting cross-validation experiments on publicly available data from the Mars Science Laboratory (MSL) from NASA's Planetary Data System (PDS). 66 NAVCAM frames were chosen over a period of 700 Sols. A domain expert labeled all images using a dedicated version of JPL's Weblabeling tool. We conducted 10-fold cross validation by splitting the dataset into 10 subsets. Every subset is held out once for testing while the classifier is trained on the other 9 subsets. Repeating this procedure 10 times results in a set of predictions which can be used to estimate the generalization error of the system. Since we are dealing with a multiclass classification problem we report per class error rates and confusion matrices shown in Figure 9 to visualize the error in-between classes.

5. PATH PLANNER

Requirements

We first identify the capabilities required for the path planner.

1. **Risk-aware path planning:** The resulting path must avoid not only obstacles but also types of terrains types that are potentially risky. It must achieve user-specified goals in full consideration of terrain types, obstacles, wheel placement, and other risk factors such as slope.
2. **Multi-objective optimization:** Typically there are multiple criteria over which RPs optimize a path, such as path length, maximum tilt over the path, or number of arcs. What constitutes the "best" path is highly dependent on situations. A solution optimized with an arbitrarily defined objective function function often fails to meet user's need. Rather, a plausible approach is to generate multiple paths that are Pareto optimal and let the user choose one.
3. **Interactive planning:** Planning for a Mars rover is typically an iterative process where a path is evaluated and modified multiple times. Therefore, to maximize the utility of the Tool, it must allow intuitive interaction between users and algorithm.

Technical Approach

In order to meet the first requirement, the path planner first generates a traversability map by incorporating the outputs of terrain classifier with DEM. It then constructs a graph by

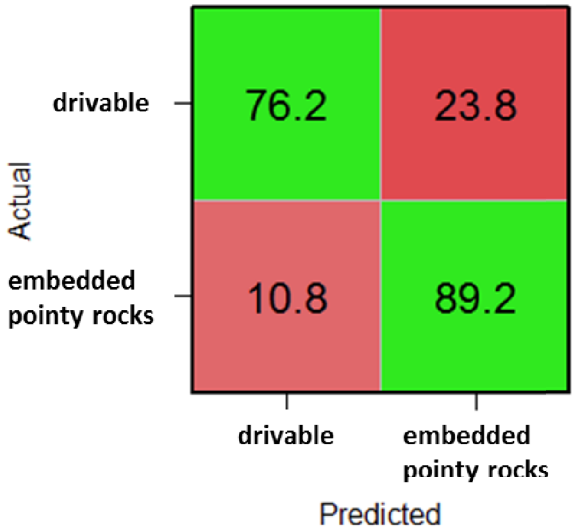


Figure 9. Confusion matrix showing the classification performance on the hold-out test set. The accuracy on drivable terrain comprising sand, bedrock and loose rock is 76.2%. Embedded pointy rocks which pose danger for MSL is detected with an accuracy of 89.2%.

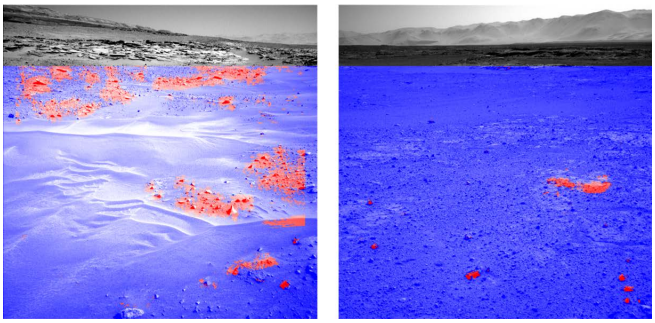


Figure 10. Prediction on the hold-out test set. Blue represents the drivable classes sand, loose rock and bedrock and red represents the dangerous class of embedded pointy rocks.

using the rapidly-exploring random graph (RRG) algorithm [5]. After the user specifies a goal and waypoints, it runs the A* search algorithm on the graph. To meet the second requirement, the search is performed multiple times with different objective functions. As a result, the Tool outputs multiple suggested paths. Finally, to meet the third requirement, the Tool provides a graphical user interface (GUI) to allow collaborative planning between users and the path planning algorithm. The following subsections explain these three technical aspects in detail.

Traversability Map

A traversability map is a grid of cells, each of which takes a Boolean value indicating whether it is traversable. It is constructed from two sources: the output from the terrain classifier and a DEM.

Recall that the terrain classifier attaches a label to each pixel of input images. These labels need to be projected on a

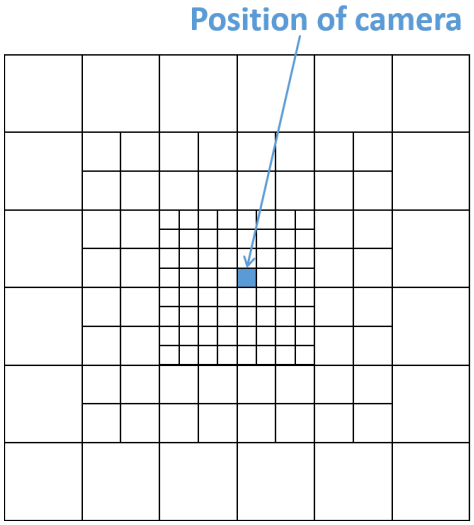


Figure 11. A variable-size grid used for traversability map. The grid size increases as moving away from the position of the stereo camera used to produce a DEM. The grid has a nested structure in which a n-by-n grid is embedded in a n-by-n grid that is twice in size.

2D Cartesian space. Such a projection is performed when a DEM is created out of stereo images. Therefore we run the classifier on the same images used for stereo processing, and apply the same projection to map the classifier's output to the DEM. The user chooses which terrain classes to avoid, and the cells of the map in the specified terrain classes are labeled as untraversable.

We also use DEM to geometrically identify obstacles. A challenge is that, as a nature of stereo processing, the level of error quickly grows as moving away from the position of camera. This error appears as a high-frequency noise. Our solution is to use a variable-size grid. In the traversability map, the grid size is small near the rover's position, and it becomes greater as going away from the rover, as shown in Figure 11. A greater grid size helps to reject the high-frequency noises by averaging them out.

In particular, in our application, the grid size is 20 cm within a square centered at the rover with 30 meter-long edges. (Thus $n = 150$ in Figure 11.) The grid size becomes 40 cm outside of this square and within a greater square with 60 meter-long edges. In this manner, the grid size is doubled at a doubling interval.

Then, for each cell in this grid, we compute the difference in height between the cell and the best fitting plane to the m-by-m cells around the cell. (We use $m = 5$.) If the difference is greater than a pre-specified positive threshold, we identify the the cell is a positive obstacle; if the difference is less than a negative threshold, we identify it a negative obstacle. Both positive and negative obstacles are labeled as untraversable on the map.

Path Planning with Wheel Placement

The Mars rovers already have an on-board path planning capability, which represents a rover by a point while obstacles are expanded by the radius of the envelop of the rover to avoid collision [4], as in Figure 3-(a). A challenge is that our terrain classifier often finds densely populated small

hazardous rocks, as in the right two figures in Figure 7. In such a situation, the existing path planner often fails to find a feasible path because nearly the whole surface becomes untraversable, as in Figure 3-(a). Therefore, we developed a new path planning capability that explicitly considers wheel placement, instead of dilating obstacles, as in Figure 3-(b).

The path planner performs planning in two steps, one of which is off-line while the other is online. The first step is to pre-compute graphs by using the rapidly-exploring random graph (RRG) algorithm [5], a graph extension of the celebrated rapidly-exploring random tree (RRT) algorithm. A graph is a representation of safe transitions in a map, given a set of acceptable terrain classes. Hence, we construct graphs for all combinations of terrain classes. RRG is an incremental algorithm, where in each iteration a randomly generated node is added to a graph and obstacle-free arcs are added to the node from its neighboring nodes. Figure 12 shows the RRG algorithm extending a graph throughout a state space. Although it is a randomized algorithm, the probability that the shortest path between any given two locations are included in the graph converges to one as the number of nodes goes infinity. In practice, it is known that the convergence is very quick even for a high-dimensional space. RRT, RRG and their extensions are often used for practical applications such as a ground vehicle for the DARPA Grand Challenge [12] and micro aerial vehicles [13].

RRG adds a node to the graph only if there is an obstacle-free arc from a nearby node. In a regular implementation, obstacle freeness of an arc is checked by representing the rover as a point in a space with dilated obstacles. In our implementation, as we discussed previously, instead of dilating obstacles, we explicitly consider wheel placement. More specifically, as shown in Figure 3-(b), for each arc, we consider two parallel trajectories representing the left and right wheel tracks, and check if both of them do not intersect with untraversable cells.

The second step of path planning is A* search, which is performed on-line during the interaction with a user through the GUI. A* search is a standard graph search algorithm that is very frequently used for path planning. In our path planner, after user makes choices about which terrain classes to be avoided, the path planner loads the corresponding graph generated in the previous step. Then, once a waypoint is picked, the search algorithm runs on the graph to find the path from the previous waypoint to the new waypoint that minimizes a given cost function. The search is repeated for each of the cost functions to produce multiple suggestions.

The algorithm does not require any assumption on the cost function. For example, it can be a function of terrain types, path length, number of turns, slope, or any combination of them. Figure 13 shows optimal paths for two cost functions: path length (shown in green) and a weighted sum of path length and number of turns (shown in blue).

User Interface

The path planner works collaboratively with a rover planner through the GUI shown in Figure 13. The followings are the typical workflow to find the optimal path.

1. The user specifies which terrain classes must be avoided by checking the boxes on the top left
2. The user specifies waypoints by clicking on the map
3. The path planner generates multiple path options that achieve the waypoints while respecting constraints. The property of each path (e.g., path length, number of arcs, and maximum tilt) is shown on the right.
4. The user checks the validity of the paths by flipping through different views, such as terrain classes, DEM, slope map, traversability map, and NAVCAM image overlay.
5. If the result is unsatisfactory, the user clears the result and start over.
6. If a satisfactory path is obtained, the user export the result to RSVP HyperDrive by clicking the button on the bottom right.

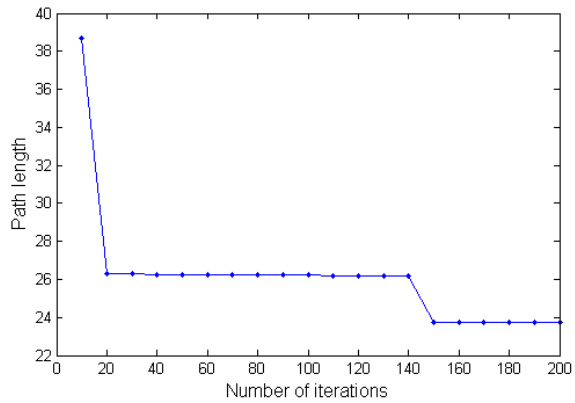


Figure 14. Convergence of RRG. This graph shows the length of the path shown in pink in Figure 12 as a function of the number of RRG iterations. As the graph grows, shorter paths become available. It is guaranteed that, as the number of iterations goes to infinity, the probability that the shortest path between two points is included in the graph converges to one.

Performance evaluation

Figure 12 visualizes the process of RRG exploring a map. The graph is initialized with a single node at the initial position of the rover and quickly grows to cover the feasible state space. As can be seen in Figure 14, the best path in the graph it quickly converges to the optimal solution. In our path planner, we use graphs generated by 1000 iterations of RRG. Computation time of a graph with 1000 nodes are typically 2-3 minutes. Remind that the graphs are pre-computed, hence computation time is not critical. We also note that the computation of RRG can be easily parallelized, although we have not implemented parallel computing capability yet. The average computation time of the A* search to find the optimal path between two randomly chosen points was 1.41 seconds, with the standard deviation 0.73 seconds. This is quick enough to allow collaborative interaction with users. The algorithms are prototyped in Matlab. Computation time is evaluated by a machine with Intel Xenon CPU clocked at 3.10 GHz with 16GB of RAM.

6. Conclusion

We presented the Risk-aware Mars Rover Operation Tool, which is capable of 1) autonomously identifying hazardous terrain and 2) suggesting safe paths that avoid the hazards. The hazard detection capability was built upon a vision-based terrain classifier using a machine learning algorithm, Random Forest. The classifier was trained by a training data set consisting of on-board images labeled by a human expert. The trained classifier classifies each pixel of an image into five terrain types: sand, loose rock, bedrock, embedded pointy rock, and embedded round rock. The path planning

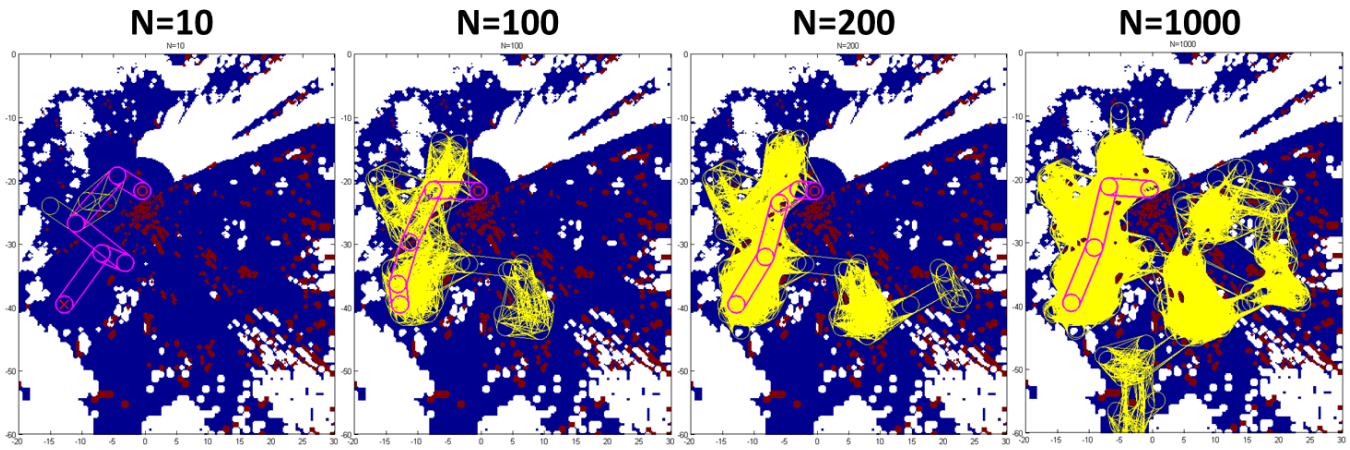


Figure 12. Visualization of the RRG algorithm exploring feasible paths on a Mars terrain. Shown in yellow are all the obstacle-free paths in the graph, while shown in pink is the shortest path from the initial position of the rover to a given goal.

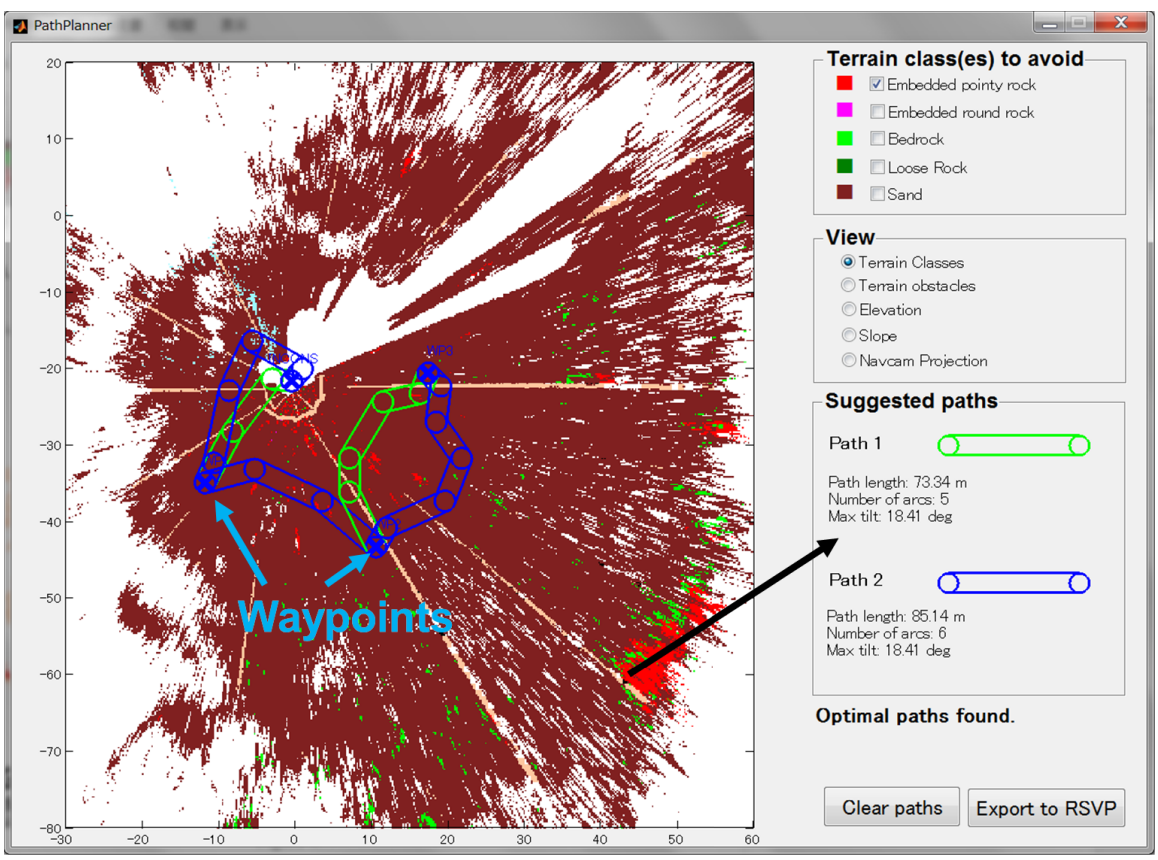


Figure 13. Graphical Interface of the path planner. A user specifies which terrain class(es) to avoid, as well as waypoints and goals by clicking on the map. Then the path planner makes multiple path suggestions. The result can be exported to RSVP. The figure shows optimal paths for two cost functions: path length (shown in green) and a weighted sum of path length and number of turns (shown in blue).

capability was built on the rapidly-exploring random graph (RRG) algorithm with an extension to explicitly consider wheel placement. The resulting graph is used to generate multiple path suggestions by running the A* algorithm multiple times with different objective functions. The developed capabilities were demonstrated using the real data from the MSL rover Curiosity.

ACKNOWLEDGMENTS

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. We thank Nicholas Toole, Paolo Bellutta, Fred Calef, Matt Heverly, Mihail Petkov, and Bob Deen for their help and advice.

REFERENCES

[1] J. R. Wright, F. Hartman, S. Maxwell, B. Cooper, and J. Yen, "Updates to the rover driving tools for curiosity," in *Proceedings of the 8th International Conference on System of Systems Engineering*, May 2013, pp. 147–152.

[2] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, pp. 1545–1588, 1997.

[3] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] J. J. Biesiadecki and M. Maimone, "The mars exploration rover surface mobility flight software: Driving ambition," in *2006 IEEE Aerospace Conference Proceedings*, 2006.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics*, vol. 30, no. 7, 2011.

[6] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 1297–1304.

[7] T. J. Fuchs and J. M. Buhmann, "Computational pathology: Challenges and promises for tissue analysis," *Journal of Computerized Medical Imaging and Graphics*, vol. 35, no. 7, pp. 515–530, April 2011.

[8] T. J. Fuchs, P. J. Wild, H. Moch, and J. M. Buhmann, "Computational pathology analysis of tissue microarrays predicts survival of renal clear cell carcinoma patients," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2008*, ser. Lecture Notes in Computer Science, D. Metaxas, L. Axel, G. Fichtinger, and G. Szekely, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, vol. 5242, pp. 1–8.

[9] T. Sharp, "Implementing decision trees and forests on a GPU," *Computer VisionECCV 2008*, pp. 595–608, 2008.

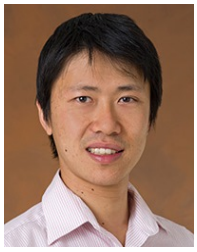
[10] D. Thompson, A. Allwood, D. Bekker, N. Cabrol, T. Fuchs, and K. Wagstaff, "Texturecam: Autonomous image analysis for astrobiology survey," in *Lunar and Planetary Institute Science Conference Abstracts*, vol. 43, 2012, p. 1659.

[11] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, Jan. 1979, mathematical Reviews number (MathSciNet): MR515681; Zentralblatt MATH identifier: 0406.62024.

[12] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 1681–1686.

[13] M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "Path planning for motion dependent state estimation on micro aerial vehicles," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 3926–3932.

BIOGRAPHY



Masahiro Ono *is a Research Technologist in the Robotic Controls and Estimation Group. He is particularly interested in risk-sensitive planning/control that enables unmanned probes to reliably operate in highly uncertain environments. His technical expertise includes optimization, path planning, robust and optimal control, state estimation, and automated planning and scheduling. Before joining JPL in 2013, he was an assistant professor at Keio University. He earned Ph.D. and S.M. degrees in Aeronautics and Astronautics as well as an S.M. degree in Technology and Policy from MIT, and a B.S. degree in Aeronautics and Astronautics from the University of Tokyo.*

Thomas J. Fuchs *is a Research Technologist in JPL's computer vision group. His research focuses on the development of new ensemble methods and Bayesian sampling techniques for large scale machine learning. His applications range from space exploration and robotics to computer vision and cancer research. Thomas is PI and CoI for several big data related research efforts at JPL. He received his Ph.D. from ETH Zurich for his work in the Machine Learning and his MSc degree in Technical Mathematics from the Technical University Graz.*

Amanda Steffy *is a Mission Operations Engineer in the Flight Systems Engineering Group. Amanda extensively assessed the MSL wheel damage as a member MSL Wheel Wear Tiger Team, conducted a terrain analysis tailored toward the wheel wear issue, and tested wheel life on the Mars-weight mobility vehicle. She currently works on the MSL Engineering Operations Team as a Mobility Chair and Systems Engineer assessing the health and performance of MSL. She earned her B.S. degree in Biomedical Engineering from Cornell University, and is currently working toward her M.S. in Aerospace Engineering at UCLA.*

Mark Maimone *is a Navigation and Machine Vision researcher at JPL. Mark designed and developed the autonomous vision and navigation software that lets the MER and MSL Mars Rovers drive themselves safely, and wrote ground software that automated the analysis of Mobility and arm operations on MER. Mark is now a Rover Driver for Curiosity, and he continues to develop and enhance the onboard autonomous vision and navigation software for the rovers. Mark earned his Ph.D. in Computer Science at Carnegie Mellon University in 1996, and completed a postdoc there in 1997 as Navigation and Software Lead for the 1997 Atacama Desert Trek. At JPL since 1997, Mark has also worked on the Long Range Science Rover, Planetary Dexterous Manipulator, and Pioneer Vision System for Chornobyl Inspection projects, delivering 3D vision systems for autonomous robotic operations and mapping.*

Jeng Yen *is the Supervisor of the Robot Operations Group at JPL. Joined JPL at 1998 as a member of technical staff, he sucessfully developed rover models and simulation methods used for MER surface operation and Rover Analysis Modeling and Simulation (ROAMS) software. Started at 1987, He developed a class of numerical integration algorithm for the real-time driving simulator in the University of Iowa, later implemnted the numerical integration to support the National Advanced Driving Simulator (NADS) at the university's research park. After graduating from the Applied Math. program, he joined Comupter Adied Design Software Inc. (CADSI), where he implemented and developed new computational methods for the vehicle system design and simulation. At 1993, he joined Army High Performance Computing Research Center (AHPCRC) at the University of Minnesota at Twin City. His research is focused on the computational methods for highly oscillatory dynamic systems in the next four years at the center. In the meantime, he also served as a reserach faculty at the Automotive Research Center of University of Michigan. From 1996-98, he led the project to create a simulation of Ford automotive assembling line. This project was awarded by Engineering Animation Inc. (EAI) at Ames Iowa. At the same time, he also led another project for modeling and simulation of Army's track vehciles at United Defense, LP in Minneapolis, Minnesota.*