Particle RRT for Path Planning in Very Rough Terrain

Nik A. Melchior, Jun-young Kwak, and Reid Simmons
{nmelchio,junyoung.kwak,reids}@cs.cmu.edu
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3890.

***Abstract*— The Particle-based Rapidly-exploring Random Tree (pRRT) algorithm is a new method for planetary rover path planning in very rough terrain. The Rapidly-exploring Random Tree algorithm is a planning technique that accounts for effects such as vehicle dynamics by incrementally building a tree of reachable states. pRRT extends the conventional RRT algorithm by explicitly considering uncertainty in sensing, modeling, and actuation by treating each addition to the tree as a stochastic process.**

The pRRT algorithm has been experimentally verified in simulation, and shown to produce plans that are significantly more robust than conventional RRT. Our recent work has investigated several vehicle models to improve the performance and accuracy of the pRRT algorithm in simulation. Based on these results, we have integrated the simulator with the iRobot ATRV-Jr hardware platform and tested and verified the pRRT algorithm using IPC communication.

I. Introduction

One of the most challenging aspects of mobile robot path planning is the planner's lack of complete knowledge. Imperfect sensors present limited and imprecise information about the world around a robot. Modelling techniques make simplifying assumptions about this sensor data in order to save memory or computational complexity. Even the actuation of the robot itself is not modelled with infinite precision. In many cases, the inaccuracies in sensing, modelling, and actuation are small enough to be safely ignored, or the robot can compensate during execution of the plan.

Often, though, ignoring this problem is not enough. Many robots lack the sensor capability or processing power to detect and correct unexpected results while executing planned actions. Additionally, robots such as planetary rovers cannot afford to take any risks with their safety. If a rover were to become damaged, stuck, or overturned on another planet, there is no opportunity to repair it. Thus, we recognize a need for path planners to account for uncertainty in sensing, modelling, and actuation. If the uncertainty of data and algorithmic parameters is explicitly considered when planning, it will be possible to create plans which are more robust to uncertainty. That is, the robot will be more likely to precisely follow the plan despite this uncertainty.

This work considers a planetary rover, similar to the Mars Exploration Rovers (MER), traversing rough terrain. We assume that the rover uses stereo vision to sense the shape of the terrain nearby, but the composition of the terrain cannot be precisely determined. This leads to uncertainty in the terrain friction, and thus the rover's behavior as it moves. For the sake of safety, the current MER navigation strategy is cautious to remain far from obstacles. Even if the rover were to behave in an unexpected manner, the likelihood of impacting an obstacle is still quite low. The disadvantage of this approach is that, in especially rough terrain, autonomous planning algorithms may be unable to find any path to move forward. In this case, the rover would cease to drive autonomously, and wait for explicit driving commands from operators on Earth. It is our hope that by modelling the uncertainty in rover operation, it might be possible for the rover to continue driving autonomously in more difficult situations without sacrificing safety.

This work has been funded by NASA under grant number NASA-03-OSS-01.

II. Related Work

Previous work in path planning has taken several approaches to planning with uncertainty. One of the most common simplifying approaches is to ensure proper operation in the worst case scenario. For example, if uncertainty is only considered in actuation, but not in sensing or modelling, Hait and Siméon [1] consider the range of possible rover poses and test for impact with the terrain. Related work by Esposito in the domain of plan validation [2] samples several possible values of the uncertain parameter from a given distribution and repeats planning for each value. In the more general case, approaches such as Iagnemma's [3] computes the cost metric of traversing a particular region based on the worst case estimate of uncertainty.

Another tactic for dealing with uncertainty is to construct an initial plan, perhaps avoiding areas with the greatest uncertainty, then replan as new sensor information becomes available. This strategy, emphasizing efficient structures for replanning as uncertainty changes, has been explored by Hsu [4] and Leven [5]. Similarly, Burns and Brock [6] present an active sensing technique which adjusts planned motions in order to collect additional information about uncertain areas, in order to aid replanning.

The approach most similar to our own was presented by Gonzalez and Stentz [7]. This work, again, considers only actuation uncertainty, which is modelled as a zero-mean symmetric Gaussian. However, they are able to compute resolution-optimal paths for a point robot using the grid-based A* planner [8].

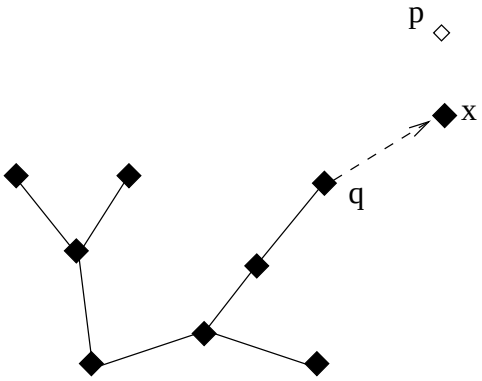

Fig. 1. An RRT extension

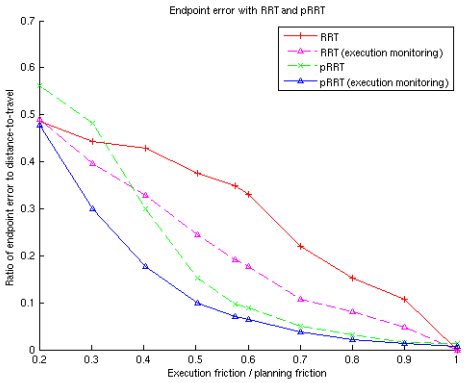

Fig. 2. Execution error in open and closed loop modes.

III. Algorithm

The Particle RRT (pRRT) algorithm is an extension to the Rapidly-exploring Random Tree (RRT) algorithm introduced by Lavalle and Kuffner [9]. RRT is a widely-used algorithm for motion planning in high-dimensional spaces with kino-dynamic constraints. The operation of RRT is conceptually simple. Each iteration of the algorithm, as depicted in figure 1, begins with a tree of states that the rover can reach. At the first step, this tree only consists of the initial state of the rover. A new state p is chosen stochastically from the state space, and the nearest node q in the tree is determined. An action is estimated to reach p from q, and the action is executed. The resulting state x is added to the tree. The planner may compute the forward simulation of the action with any level of fidelity appropriate to the task at hand. It is not necessary for the final state x to coincide with p, so it is often preferable to select the action using simple inverse kinematics, but simulate the result of the action using more accurate dynamic models.

The extension introduced by pRRT produces distributions of states, rather than single states, at each node of the tree. The distributions are nonparametric, and are derived from the uncertainty specified as input to the algorithm, and the forward simulation process itself. Specifically, we use a set of discrete particles to estimate the distribution at each node. For each extension added to the tree, several particles are computed by a Monte Carlo process. To compute a single particle, one particle from the node q is chosen as the start state, and a value is drawn from the prior distribution over the uncertain parameter (in our case, terrain friction). Simulations of the same action under different values for friction will result in different final states for the rover. After simulating several times in this manner, the resulting particles are clustered into one or more nodes, which are added to the tree with the same parent. The clustering procedure is described in detail in a previous publication [10], but its purpose is to separate qualitatively different particles into different nodes.

In general, building a planning tree while accounting for uncertainty should result in nodes whose variance grows with the depth in the tree. However, since the clustering step permits a single extension to be broken into more than one node, the variance is split as well. In addition, the probability of a single particle is associated with the prior likelihood of the sampled value of friction used to produce it. Since the prior over the uncertain parameter can have an arbitrary probability distribution function (PDF) and nodes may contain different numbers of particles, some nodes will contain more probability mass than others. Nodes which combine both low variance and high probability are good candidates for path planning, because we expect the rover to be able to reach them accurately despite the uncertainty. Consequently, a path of such nodes (from the root of the tree to a leaf) can be executed with greater accuracy by the rover. Thus, as the planning tree is built, there is a bias towards extending new leaves from such high-probability paths. This bias is implemented using a selection mechanism similar to that introduced by Urmson in [11].

The pRRT algorithm was tested in simulation and shown to produce paths which can be followed with reduced execution error compared to conventional RRT. Tests were conducted using several synthetic and realistic terrain models. RRT planned a path using a typical assumption of terrain friction, while pRRT assumed a uniform prior over a range of friction values. After planning a path with each algorithm, the path was executed open-loop several times, using a different constant value for friction each time. Endpoint error was measured as a percentage of the initial distance from start to goal.

The results are summarized in the figure 2, which shows the error in executing the planned path as a function of the terrain friction during execution. Although this algorithm was designed primarily to enhance the path planning capabilities of rovers which can obtain little or no sensor feedback during execution, this plot shows that closed-loop driving is improved as well. The plots labelled ‘execution monitoring’ were planned in an identical manner, but errors during execution were used to correct subsequent actions using the pure pursuit [12] strategy. Our tests show that even open-loop execution of a path planned with pRRT tends to result in less error than closed-loop driving using a path planned by RRT.

IV. The optimal vehicle model

In this part we try to make the optimal vehicle model to reduce execution errors over the planned path in the simulator. Currently, there are two different simulators - internal and external simulator. The external simulator might be NASA JPL ROAMS simulator [13] [14] or a real hardware platform. FIDO model (High fidelity model) is used in the external simulator, and this model is close to real robot platform. We make three different vehicle models - 4-wheeled basic, 6-wheeled basic and 6-wheeled rocker-bogie model - in the internal simulator to reduce the actual execution errors over the planned path. To test and make the optimal path, we use RRT and pRRT (particle RRT) algorithms. Evaluation factor might be the comparison results based on the each simulator's execution errors on the same conditions (including the same path). In the vehicle model, the control part is essential to operate the vehicle practically and get the optimal movement of the vehicle.

A. 4-wheeled Basic Model

We made 4-wheed basic model [15] to generate the plan and test it in the simulator (See figure 3).

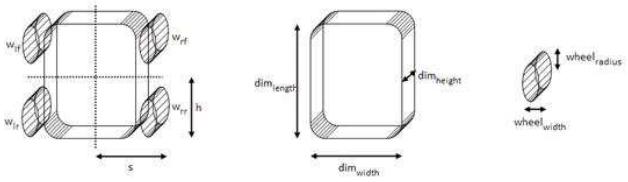

Fig. 3. 4-wheeled basic model

We used the following equations for the control part of the vehicle [15] [16] and values in table I.

$$
\begin{aligned}
v_{lf} &= \sqrt{\left(\frac{1}{\theta_{body}} - s\right)^2 + \frac{(2h)^2 \times v_{desired}}{\left|\frac{1}{\theta_{body}}\right|}} \\
v_{rf} &= \sqrt{\left(\frac{1}{\theta_{body}} + s\right)^2 + \frac{(2h)^2 \times v_{desired}}{\left|\frac{1}{\theta_{body}}\right|}} \\
v_{lr} &= \frac{\left|\frac{1}{\theta_{body}} - s\right|}{\left|\frac{1}{\theta_{body}}\right|} v_{desired}, v_{rr} = \frac{\left|\frac{1}{\theta_{body}} + s\right|}{\left|\frac{1}{\theta_{body}}\right|} v_{desired} \\
\theta_{lf} &= \tan^{-1}\left(\frac{2h}{\frac{1}{\theta_{body}} - s}\right), \theta_{rf} = \tan^{-1}\left(\frac{2h}{\frac{1}{\theta_{body}} + s}\right) \\
s &= \frac{dim_{width}}{2} + wheel_{width}, h = \frac{dim_{length}}{2}
\end{aligned}
$$

B. 6-wheeled Basic Model

We made 6-wheed basic model to generate the plan and test it in the simulator (See figure 4). We used the same equations and values (See Table I) as 4-wheeled case and additionally consider the following equations for the control part of the vehicle.

TABLE I
PROPERTY VALUES

Property	Value
Max velocity (v_{max})	30m/s
Max curvature ($\theta_{body,mas}$)	0.2
Max torque (τ_{max})	1600
Car width (dim_{width})	0.37m
Car height (dim_{height})	0.3014m
Car length (dim_{length})	0.57m
Wheel width ($wheel_{width}$)	0.1232m
Wheel radius ($wheel_{radius}$)	0.1m
Suspension upper limit	0.3m
Suspension lower limit	-0.3m
Suspension stiffness	800
Suspension damping	200
Suspension softness	0

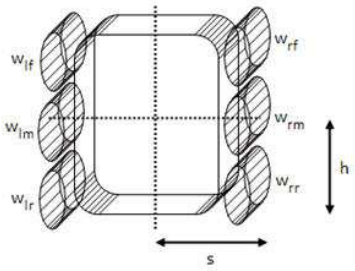

Fig. 4. 6-wheeled basic model

$$
\begin{aligned}
v_{lm} &= v_{lr} = \frac{1}{2}\frac{\left|\frac{1}{\theta_{body}} - s\right|}{\left|\frac{1}{\theta_{body}}\right|} v_{desired} \\
v_{rm} &= v_{rr} = \frac{1}{2}\frac{\left|\frac{1}{\theta_{body}} + s\right|}{\left|\frac{1}{\theta_{body}}\right|} v_{desired}
\end{aligned}
$$

C. 6-wheeled Rocker-bogie Model

For the last model, we devised the 6-wheeled rocker-bogie model [17] and tested that (See figure 5).

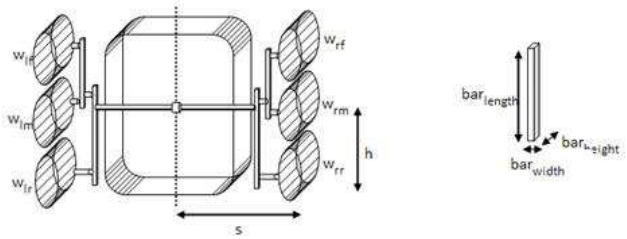

Fig. 5. 6-wheeled rocker-bogie model

We used the following equations for the control part of the vehicle based on 6-wheeled basic model, and values in Table I and II.

$$\theta_{lm} = \frac{1}{2}\tan^{-1}\left(\frac{2h}{\frac{1}{\theta_{body}} - s}\right), \theta_{rm} = \frac{1}{2}\tan^{-1}\left(\frac{2h}{\frac{1}{\theta_{body}} + s}\right)$$

TABLE II

CHANGED PROPERTY VALUES

Property	Value
Car height (dim_{height})	0.2009m
Car length (dim_{length})	0.456m
Wheel width ($wheel_{width}$)	0.1232m
Wheel radius ($wheel_{radius}$)	0.1m
Bar width (bar_{width})	0.01m
Bar height (bar_{height})	0.03m
Bar length (bar_{length})	Front: 0.2736m Rear: 0.4104m

D. Results

This project is motivated mainly by planetary rovers, such as The Mars Exploration Rover (MER), so we used Vortex simulator and ROAMS simulator as testbed. Vehicle models in both simulators are made based on real rover platform like Rocky8 and FIDO. In this experiment, the vehicle speed in the simulator is set to 5km/h between planned nodes. We tested the experiments with simulators and used MATLAB to analyze and plot the graph. We tested each vehicle model with the pRRT plan generated in the simulator. For testing, we set 4 different cases which have different goal positions, and iterated 20 times for each case to get the average value. In the Table III, 4WB means 4-wheeled basic model, 6WB means 6-wheeled basic model, and 6WRB means 6-wheeled rocker-bogie model. For the evaluation factors, we used raw endpoint error and scaled endpoint error for each case:

$$\text{Error}_{\text{scaled}} = \frac{\text{Error}_{\text{raw}}}{\text{distance}}$$

As the results are shown in Table III and figure 6 & 7, the scaled endpoint errors were reduced in the case which 6-wheeled rocker-bogie model was used in the internal simulator. This is because if we use more accurate vehicle model close to the real robot (the vehicle in the external simulator or real robot platforms) to generate the pRRT plan, we can get more reliable path for the testing platform. Of course, the results depend on the specific propert values in each model, and thus we need to find the best values for each vehicle part.

TABLE III

SCALED ENDPOINT ERRORS WITH PRRT PLAN

Scenario	Distance	4WB	6WB	6WRB
Case1	2.2597m	0.1793m	0.1338m	0.1250m
Case2	4.4463m	0.1193m	0.1197m	0.1014m
Case3	6.3354m	0.1493m	0.1501m	0.1299m
Case4	6.6298m	0.1542m	0.1849m	0.1300m
Average		0.1505m	0.1471m	0.1216m
Error scope (Max-Min)		0.0600m	0.0652m	0.0286m

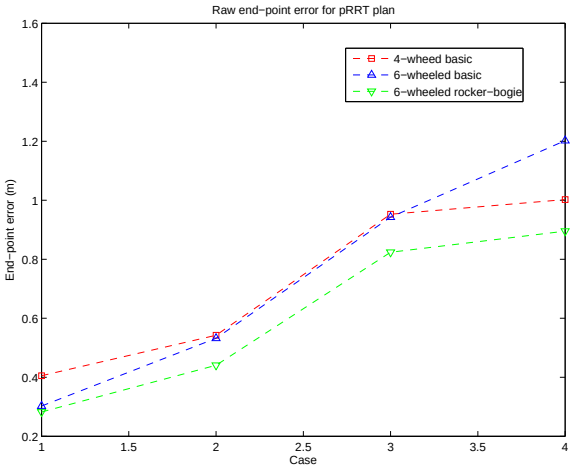

Fig. 6. Raw end-point error for pRRT plan

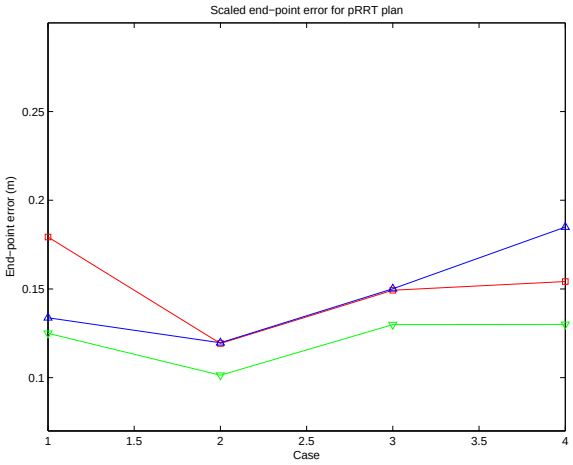

Fig. 7. Scaled end-point error for pRRT plan

V. INTEGRATION WITH ATRV-JR

For the last part, we demonstrate the result which we integrate pRRT plan with ATRV-Jr robot platform (See figure 8). We used the real robot size and 4-wheeled basic model to generate the pRRT plan. We have made three simple scenarios and generated the optimal plan for each case. Over the generated path, we actually executed a robot with commands and measured the end-point errors.

- To support a remote control, we used IPC v3.7.10
- For encapsulation, we used the wrapper API to communicate with ATRV-Jr
- "aj_locomotor" module was used to send drive command and get the actual status of the rob

A. Results

To verify the results, we demonstrate three simple scenarios. To get more reliable results, we have used the exact same property values of ATRV-Jr (See table IV).

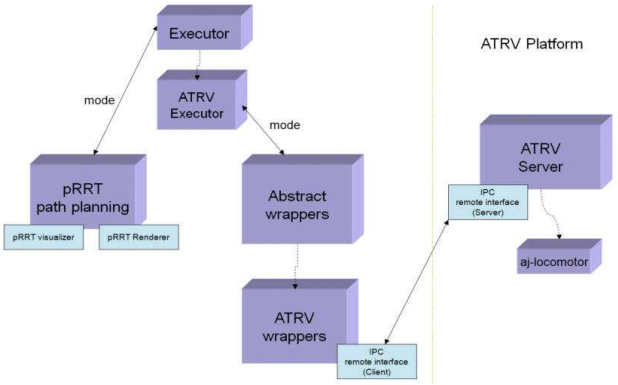

Fig. 8. Overview - integration with the ATRV-Jr

TABLE IV
ATRV-JR PROPERTY VALUES

	Value
Car width (dim_{width})	0.40m
Car height (dim_{height})	0.41m
Car length (dim_{length})	0.57m
Wheel width ($wheel_{width}$)	0.1232m
Wheel radius ($wheel_{radius}$)	0.1m

- Test 1: Test getting to the goal position along a planned straight line. You can check the details in the following table and figure 9 & 10.

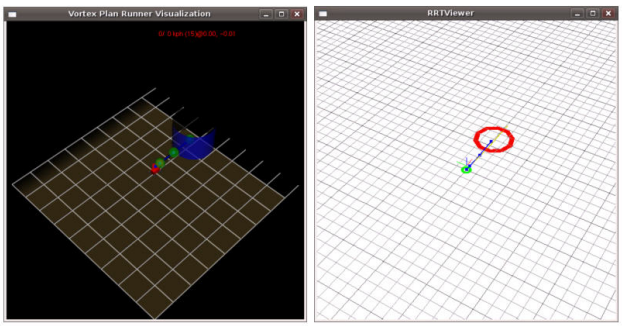

Fig. 9. pRRT plan for Test 1

Planned Path	Path build time	0.407684s
	Path length	4.59887m
Endpoint error	Planned position	x: 4.59809 y: 0.0780552 z: 0.0
	Current position	x: 4.60993 y: -0.0397487 z: 0.0
	Node diff.	0.118398m

- Test 2: Test getting to the goal position along a planned curved line. You can check the details in the following table and figure 11 & 12. When the robot moved along a curved path, it made a slightly bigger end-point error despite a similar travel distance. This is because when the robot tries to make a turn, it uses the difference between each side's linear velocity of wheels, and that causes more slips while it moves.

Fig. 10. Screenshot for Test 1

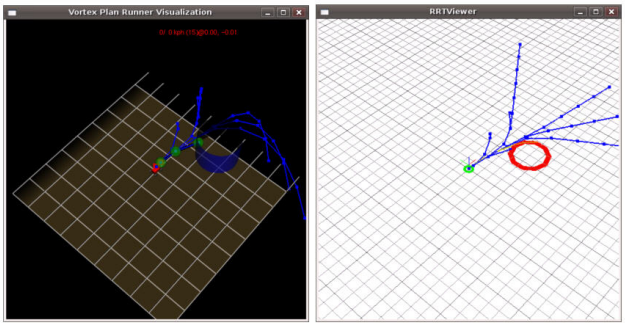

Fig. 11. pRRT plan for Test 2

Planned Path	Path build time	3.93594s
	Path length	4.54368m
Endpoint error	Planned position	x: 4.26976 y: -1.56048 z: 0.0
	Current position	x: 4.15812 y: -1.69545 z: 0.0
	Node diff.	0.175158m

Fig. 12. Screenshot for Test 2

- Test 3: Test getting to the goal position with obstacle avoidance. You can check the details in the following table and figure 13 & 14.

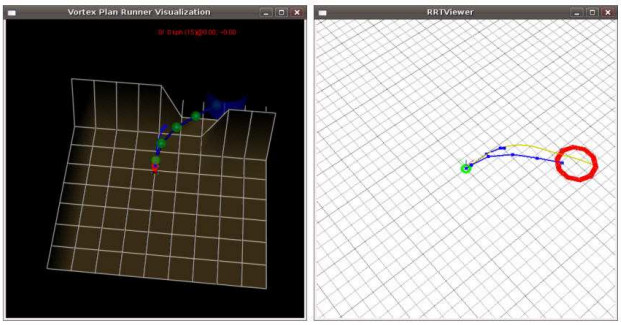

Fig. 13. pRRT plan for Test 3

Planned Path	Path build time	9.10327s
	Path length	8.74798m
Endpoint error	Planned position	x: 7.0689 y: -5.15592 z: 0.0
	Current position	x: 7.07212 y: -5.26122 z: 0.0
	Node diff.	0.105347m

Fig. 14. Screenshot for Test 3

VI. Conclusions

In this paper, we have explained pRRT algorithm, and showed the performance in the simulation and simple real cases. pRRT algorithm is a new and extended method for planetary rover path planning in very rough terrain. With this algorithm, we also have tried to find the optimal vehicle models to reduce the end-point errors. We have created three different vehicle models - 4-wheeled basic, 6-wheeled basic and 6-wheeled rocker-bogie model, and compared the performances among them. We could can make more reliable and optimal paths with pRRT algorithm than basic RRT. Furthermore, when we have used the created vehicle models, the end-point errors for both of the simulation and real cases were pretty small. Of course, because the error depends on the difficulty for the plan, it is a little difficult to tell the differences clearly between the different models with slightly changed values. However, based on the resuls, we can conclude the control design and coefficient values of the vehicle model are good and reasonable.

VII. Future Work

We have showed the results in the simulation cases and integrated with ATRV-Jr with simple scenarios. However, in real and more complicated situations, we have to consider more factors like the interaction between the soil and vehicle, slips and the coefficient of friction because we cannot assume the ideal conditions. Even though we use the same assumption and consider those factors, we cannot guarantee the exact same results in the real life. For the next step, we thus will verify pRRT algorithm on the uncertain terrain with JPL's FIDO Rover. Through more experiments, we can show the difference between the simulated results and real executions over more complicated scenarios, and verify the performance and reliability of this method in uncertain cases. In addition, we will apply learning algorithms to find the optimal property values including weight of body and wheel, and suspension, softness of each joint for the selected model.

References

[1] A. Hait and T. Siméon, "Motion planning on rough terrain for an articulated vehicle in presence of uncertainties," *IEEE/RSJ International Symposium on Intelligent Robots and Systems*, pp. 1126–1133, 1996.

[2] J. M. Esposito, J. Kim, and V. Kumar, "Adaptive RRTs for validating hybrid robotic control systems," *International Workshop on the Algorithmic Foundations of Robotics*, July 2004.

[3] K. Iagnemma, F. Genot, and S. Dubowsky, "Rapid physics-based rough-terrain rover planning with sensor and control uncertainty," in *Proceedings of IEEE International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 2286–2291.

[4] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2000.

[5] P. Leven and S. Hutchinson, "Toward real-time path planning in changing environments," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2000.

[6] B. Burns and O. Brock, "Sampling-based motion planning with sensing uncertainty," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 2007.

[7] J. P. Gonzalez and A. T. Stentz, "Planning with uncertainty in position: An optimal and efficient planner," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '05)*, August 2005, pp. 2435 – 2442.

[8] N. J. Nilsson, *Principles of artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1980.

[9] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[10] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 2007.

[11] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *IEEE/RSJ IROS 2003*, October 2003.

[12] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, January 1992.

[13] A. Jain, J. Balaram, J. Cameron, J. Guineau, C. Lim, M. Pomerantz, and G. Sohl, "Recent developments in the ROAMS planetary rover simulation environment," in *Proceedings of the 2004 IEEE Aerospace Conference*, Big Sky, Montana, March 2004.

[14] A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, and R. Steele, "Roams: Planetary surface rover simulation environment," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2003)*, Nara, Japan, May 2003.

[15] J. Y. Wong, *Theory of Ground Vehicles, 3rd Ed.* Wiley, 2001.

[16] K. Iagnemma, H. Shibly, A. Rzepniewski, and S. Dubowsky, "Planning and control algorithms for enhanced rough-terrain rover mobility," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2001.

[17] H. Hacot, S. Dubowsky, and P. Bidaud, “Modeling and analysis of a rocker-bogie planetary exploration rover,” in *Proceedings of the Twelvth CISM-IFToMM Symposium (RoManSy 98)*, Paris, France, July 1998.