Presented at the 7th International Conference on Automation Technology, Sept. 12-14, 2003, Chia-Yi, Taiwan

Design and Motion Simulation of the Autonomous Exploration Vehicle

Jyh-Cheng YU*

Department of Mechanical and Automation Engineering
National Kaohsiung First University of Science and Technology

Johnson W.C. LIAO, Ming-Yang LI, and Shian-Hung LI

Dep. of Mechanical Engineering
National Taiwan University of Science and Technology

ABSTRACT

This paper describes a novel design of exploration vehicle, ACV, and the scheme of motion control and path planning using the virtual prototyping system. ACV possesses the high local mobility and obstacle crossing ability due to the design of four independent drives and the active rock arms. An integrated design environment is established using ADAMS and Matlab/Simulink to simulate the vehicle kinematics, the interaction between the environment and the vehicle, the motion planning algorithm, the dynamic behavior, and the control strategy. A prototype path planner, the Modified Tangentbug Algorithm (MTA), is described and implemented to the navigation of the virtual vehicle. MTA generates a feasible path to avoid blocking obstacles, and to stride over passable obstacles in a known environment. The generated path is then decomposed into a sequence of motion commands written in ADAMS macros to control the vehicle. At last, a prototype ACV is described, and the interface structure between the simulation platform and the prototype is presented.

Keywords: Terrain vehicle, Virtual prototype, ADAMS, Path planning

1. Inroduction

In some cases, such as hazard environments and extraterrestrial exploration, where a manned task is prohibited, an autonomous unmanned rover that can adapt to the various severe terrain is an alternative for the investigation purpose. There are many studies done about various candidates regarding the exploration rover. Related configuration designs for the wheeled roving vehicles are the Jet Propulsion Laboratory's CARD (Computer-Aided Remote Driving) rover [9], WAAV (Wheeled Actively Articulated Vehicle) [4][8], and the Rock 7 [6]. The JPL CARD design is a six-wheeled rover that is composed of three modules and two passive articulations. It is a direct derivative of a design for a lunar rover [10]. WAAV adopts a similar configuration with six independently actuated driving wheels and two active articulations with three degrees of freedom. The actively coordinated wheeled vehicles [7] have shown to possess considerable advantages in providing these capabilities as compared with a passive vehicle system. With the aid of the active articulation design, obstacle-crossing ability has been greatly enhanced. WAAV can negotiate uneven terrain including vertical steps and horizontal gaps.

This study will introduce a novel design of exploration vehicle, Adaptive Configuration Vehicle (ACV), and present the virtual prototype procedure to integrate the kinematics analysis, the path planning, the motion control, and the simulation of terrain negotiation, using the dynamic analysis system, ADAMS. A physical prototype of ACV is thus built to verify the design.

2. Adaptive Configuration Vehicle

ACV [3] possesses the similar obstacle crossing capability as WAAV but using a simpler four-wheeled design. ACV is designed to travel on the artificial environments where the capabilities of avoiding blocking obstacles and crossing regular obstacles, such as steps and gaps are required. We assume that the working environment of the vehicle is known, and neglect the vision module at the current study.

ACV is composed of a body module and four independently actuating wheels attached to two active rock arms as shown in Figure 1. The batteries and the control module are stored on the body module. There is another passive wheel beneath the body module for balance purpose. There are eight actuators in the vehicle. In addition to four driving wheels, two motors control the angle of the front wheel arms for steering purpose. The front wheel arm design enables ACV to spin on the middle

* Corresponding Author: jcyu@ccms.nkfust.edu.tw

of the rear wheel pair. The other two motors lift the rock arm when striding over obstacles. ACV can adapt the vehicle configuration to uneven terrains and cross some obstacles, such as vertical steps and horizontal gaps

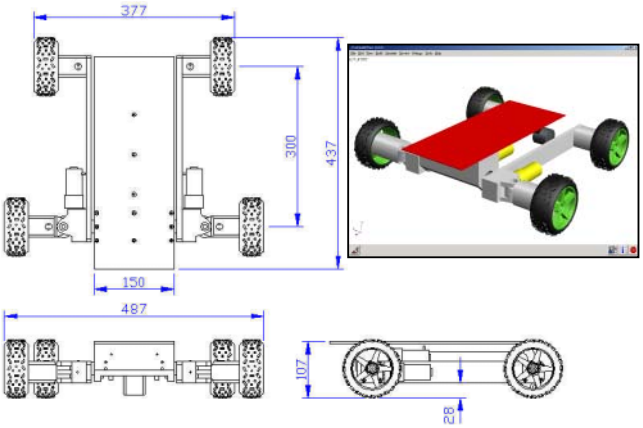


Figure 1: The structure design of ACV

3. Motion Simulation

This study applies ADAMS as a virtual design platform to integrate the kinematics simulation, the interaction with environment, the control strategy, and the path planning module. ADAMS is based on multi-body theories that are suitable to analyze complex motions [12]. MATLAB/Simulink [13] is used to design the motion control kernel of ACV. These tools assist efficiently in evaluating and analyzing our concept before the manufacturing of the prototype.

The 3D model of ACV is built in ProE and imported into ADAMS by the Mech/Pro interface module. The Mech/Pro Interface can transfer not only the complete geometries, but also the definitions of mass properties, joints, and motions. The environment of the vehicle is constructed using ADAMS as well. We then define the relationships between the tires and the environment using the TIRE STATEMENT. Besides, we build several road types to investigate the negotiation capability of ACV when traveling on different terrains.

3.1. Motion Commands

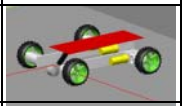
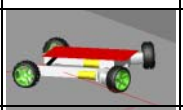
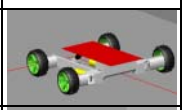
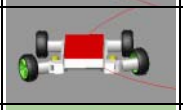
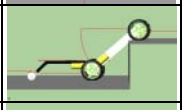
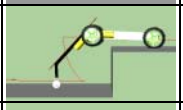
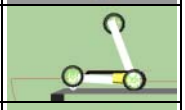
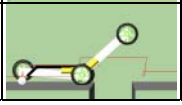
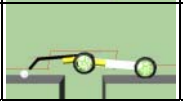
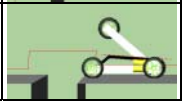
The motion control of ACV is predefined into eight motion commands written as the macros of ADAMS. The summary of the motion commands is listed in Table 1. Each command is associated with the control parameters listed in Table 1, and a corresponding time sequence of the actuators, such as Figure 2. FWM and BWM are motion commands for straight move. SPIN pivots ACV for a given angle on the middle of the rear wheel pair. RTN and LTN are commands for turning ACV round a given center. There are three more motion commands for crossing the obstacles with the help of two active rock arms. USTEP and DSTEP will command ACV to surmount and climb down a negotiable high step. CRG simulates the motion to go across a wide gap (Table 2). To facilitate the execution of the motion macros, we design a customized GUI interface. This GUI interface serves as the integrated control panel of ACV for the path planning, the actuator control, and the motion simulation.

By using the motion simulation of ADAMS, we can determine the feasibility of the control logic. When the results show any instability at specific conditions, compensators can be added into the system to improve the vehicle stability. For the design of compensators, we define the inputs and outputs of the control system by ADAMS/Controls interface linked with MATLAB/Simulink to integrate the mechanism and control system.

Table 1: Motion commands of ACV

Motion commands	Motion Discription	Motion Parameters
FWM	Straight forward moving	Speed, Distance
BWM	Straight backward moving	Speed, Distance
USTP	Climb up a step	Height
DSTP	Climb down a step	Height
CRG	Stride over the gap	Width
RTN	Right turn by arc	Angular displacement, Center
LTN	Left turn by arc	Angular displacement, Center
SPIN	Spinning on the middle of the rear wheel pair	Direction, Spin angle

Table 2: The illustrations of the motion commands of ACV

Motion commands	1st snapshot	2nd snapshot	3rd snapshot
SPIN			
LTN			
USTP			
CRG			

3.2. Simulation of Control Module

MATLAB/Simulink receives the information of plant transferred from ADAMS/Controls, and implements the design of controller. For DC servomotors, we can control angular speed and position. The PID controller is used to control the output error. The control parameters of the corrected movement are then passed to ADAMS.

The ACV has a autonomous path planning capability. Given a destination in a known environment, the path planning module can generate a feasible path and converts the path into a series of motion commands as shown in Figure 3. The motion commands then pass to the control kernel to implement the position control. Detailed strategy of path planning will be discussed in the next section.

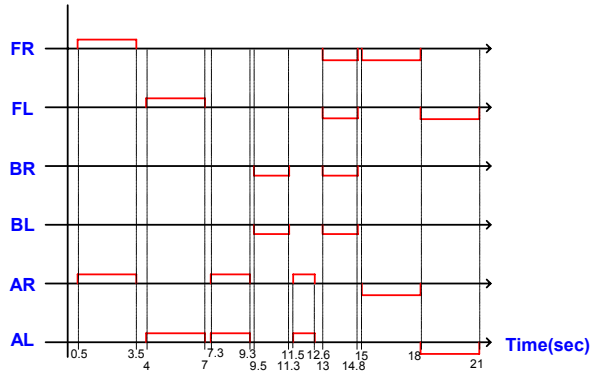


Figure 2: The time-sequence diagram of the motors for climbing up a step (USTP)

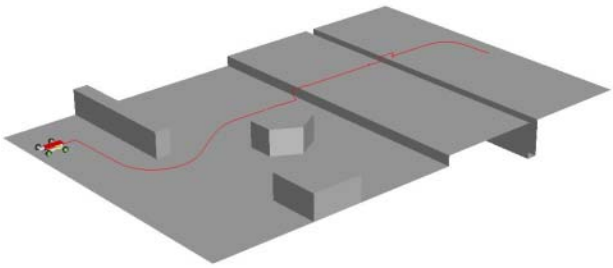


Figure 3: ACV travels in the virtual environment

4. Strategies of Path Planning

4.1. Relevant Algorithms

To make unmanned missions possible, the exploration vehicle should be able to generate the moving path to a given destination. Relevant work can be divided into three major categories: "classical " path planners, heuristic planners, and sensor-based motion planners. "Classical" planners assume full knowledge of the environment. Heuristic planners are based on a set of "behaviors", and can be applied to unknown environment[11]. However, heuristic planners do not guarantee to reach the goal. Sensor-based motion planners rely solely on the rover's sensor and yet guarantee completeness.

Kamon *et al* [5] proposed the Tangentbug algorithm that is a sensor-based motion planner. The Tangentbug algorithm uses range data from an omnidirectional camera, Local Tangent Graph (LTG), to choose the local optimal direction while moving towards the target. If a obstacle is in the path to target, a tangent line from the current robot position to the obstacle will be the first move following by a boundary following until the target is not blocked by the obstacle. LTG iterates until the target is reached.

The Wedgebug algorithm [14] improves the shortcomings of Tangentbug. It uses a stereo pair of cameras mounted on a pan-able mast. Instead of an omnidirectional view at every step, the planner scans only specific areas to avoid unnecessary sensor scans and rover motion. The rover is modeled as a point robot in a 2D binary environment. Obstacle boundaries block sensing as well as motion. The planner consists of two modes: motion-to-goal and boundary-following. Its extended version, called the Roverbug Algorithm, has been implemented on the JPL's Rocky7 prototype microrover [11]. Roverbug calculates the obstacles' silhouettes to relax the assumption of a point robot rover. The basic operation of Roverbug Algorithm includes calculating obstacles' convex hulls, looking in the appropriate direction, and increasingly building and executing each subpath until the goal is reached.

Cunha et al. [1] and Yamamoto et al. [2] are two classical path planners that use cell decomposition technologies to get the feasible and the optimal trajectories toward the goal. The environment is assumed known and obstacles are transformed to circumscribed polygons. Feasible trajectories are consisted of lines, arcs, and splines. Traveling time is used as the objective to determine the best path.

4.2. Modified Tangentbug Algorithm (MTA)

The above algorithms only consider the strategy to avoid obstacles, but neglect the situations where passing through certain obstacles is possible if the rover has the obstacle-crossing ability like ACV. Besides, path planners should include controlling cost in the selection of the optimal path. The control of rovers to follow a path consisted of splines and irregular contours will be very difficult. The theoretical optimal path may not be suitable for practical purpose. Plus, slippery between wheels and loose terrains will make complex paths unrealistic. A simple path consisted of lines and arcs will be much easier to realize.

This study modifies the Roverbug Algorithm to address the path planning needs of ACV. The environment is assumed known although MTA can be easily adapted to the range data obtained from the rover sensors. A morphing technique is first applied to transform the environment to a 2D map. The path planner iterates between two motion modes, the To-Goal Mode and the Obstacle Mode, until the target is reached. The schematic flowchart of MTA is shown in Figure 4.

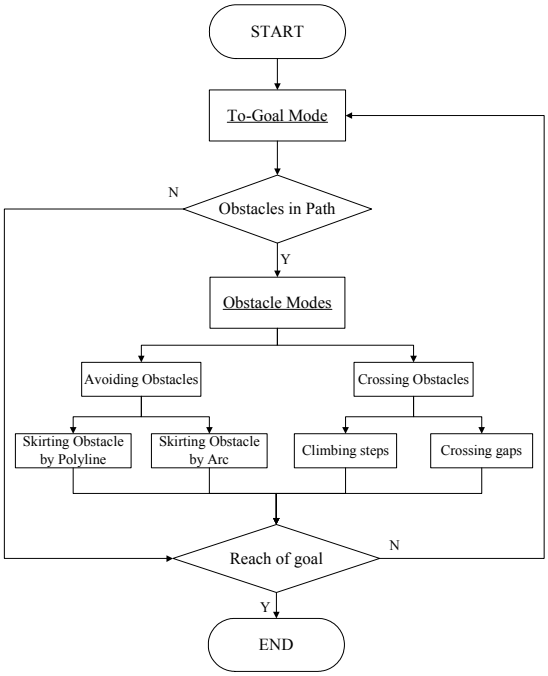
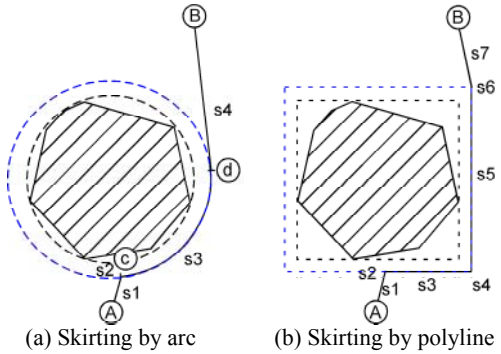


Figure 4: Flowchart of Path planning



(a) Skirting by arc (b) Skirting by polyline

Figure 5: Two different paths for the two morphing geometry of the obstacle

Morphing of Obstacle

The obstacles are first enlarged to include the required clearance between the pivot of ACV and the obstacles. Blocking obstacles are simplified as circumscribed polygons and circles, and steps and gaps are simplified as polyline. The environment is thus represented as a two dimensional map for path planning. There are various shapes for the obstacles in the real environment, so the morphing of obstacles is very helpful to simplify the complex circumstances. Two basic shapes of obstacle, circles and polygons, are considered in the morphing process. These two shapes fit the basic motion commands of ACV: the straight moving, spinning, and the arc turning. Detailed morphing procedure is awaited future investigation. For the example in Figure 5, the shape of the obstacle can be morphed into a circle or a rectangle to reduce the number of turns for ACV. A shorter path doesn't guarantee a shorter controlling time. The final choice should consider the actual performance of motors and controllers.

Basic Planning Mechanism of Motion Modes

The path planner has two basic motion modes: To-Goal, and Obstacle Modes. There are two sub-modes of the Obstacle Modes. One is to avoid the obstacles by skirting the morphing boundary of the obstacle. Depending on the type of obstacle, the planner will use arc or polyline motion to skirt the obstacle. The other sub-mode is to stride over the obstacles if they can be crossed, such as steps and gaps. In such a case, ACV will approach these types of obstacle vertically to reduce the crossing distance.

The path planner will use the two dimensional map from the morphing process to design the path. ACV starts from a direct move toward the goal until reaching the boundary of a morphing obstacle, such as the S1 in Figure 5. The planner then decides if the obstacle can be stridden over. If avoiding mode is adopted, ACV will skirt the obstacle using appropriate motion commands until this obstacle is not in its path to goal. For instance, in the Figure 5(a), ACV uses LTN command, S3, to skirt the circle circumscribed the obstacle until point "d" where the target becomes "visible". There is a SPIN command, S2, before LTN to reorient ACV to the tangent direction of the circle at point "c". The generated path can be easily decomposed into a sequence of motion commands that are used directly to control the vehicle.

Selection of Path

The vehicle usually could skirt the obstacle in either direction. A rule of thumb is to pick the direction that ACV could sooner clear the obstacle and the goal becomes visible. Figure 6 shows another. Point B is the goal to reach. If the rover takes the right route, the first obstacle can be cleared at step S4. On the other side, if the left route is taken, the obstacle cannot be cleared until step S6', which takes more turns and appears to be longer. Similar situation happens at the second obstacle. It's obvious that the right route is preferred.

Simplification of Path

The feasible path generated from MTA could be simplified to provide a better path. The simplification process tries to combine consecutive turning points to form a more direct path. The planner will search consecutive turning points, and merges unnecessary turning points if the new sub-path is clear of any obstacle. For the example shown in Figure 7,, the path L1 is the primitive path based on MTA. However, the point b in route $L1$, could be skipped to merge sub-paths $\overline{ab}$ and $\overline{bc}$ into a shorter sub-path $\overline{ac}$. Similar case appears in sub-paths $\overline{cd}$, $\overline{de}$,

and $\overline{ef}$. The simplified route *L2* appears to be a more direct path. The result will be the same as the Tangentbug algorithm.

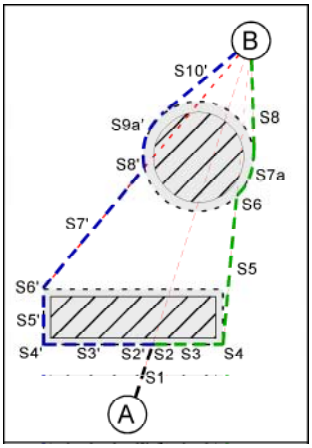


Figure 6: The direction determination for the Obstacle Mode

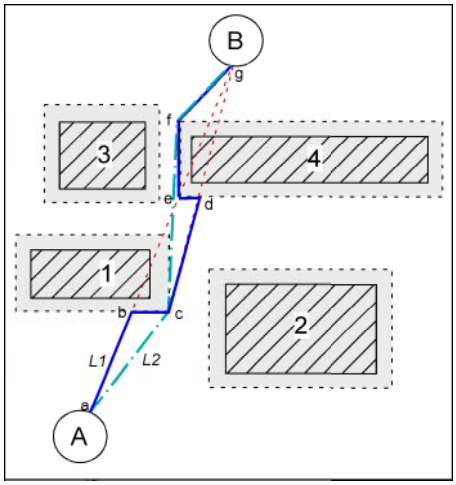


Figure 7: The different paths by using MTA

Path Planning Example

Figure 8 shows the example of ACV moving on a typical terrain. According the Modified Tangentbug Algorithm, we can generate a feasible path that is composed of S1'~S9. These subpaths are associated with the predefined ACV motion commands. The coordinates of each turning pints will become the controlling parameters of the motion commands. The motion command list of this example is shown in Table 3. ACV usually needs to reorient itself before take the next motion command. For instance, in subpath S1', ACV will make a spin to orient the vehicle following by a forward motion. The corresponding commands are "SPIN" and "FWM". Subpath S4 represents that ACV will skirt the round obstacle by a arc motion. The motion commands are "SPIN" and "LTN". Before striding over the vertical step in s6, ACV needs to reorient itself to the normal direction of the step.

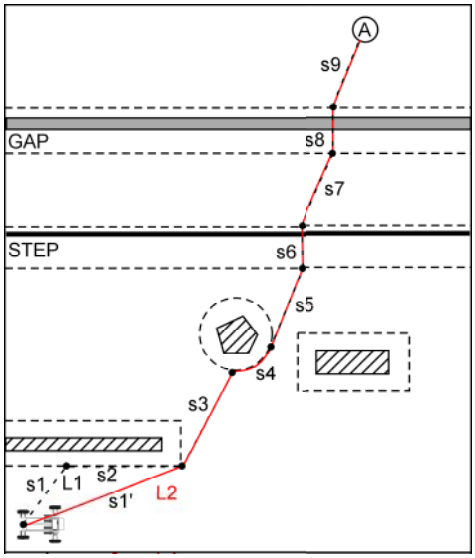


Figure 8: Path planning of ACV moving

Table 3: Motion command list of ACV

Sequence of Motion Mode	Commands
S1'	SPIN, FWM
S3	SPIN, FWM
S4	SPIN, LTN
S5	FWM
S6	SPIN, USTP
S7	SPIN, FWM
S8	SPIN, COG
S9	SPIN, FWM

5. Integration of Software and Hardware Systems

A prototype ACV is built to verify the design concept and the control strategy. The controlling hardware is composed of one 8051 chip, wireless control module, and eight DC servomotors. Four encoders are installed on the motors of steering mechanism and the wheel-arm lifting mechanisms for feedback functions.

We develop a GUI module to integrate the control module, motion simulation, and the hardware system.. The functions of the interface are explained as follows, and the schematic relationship among each module is shown in Figure 9.

a. Build the 3D CAD model in Pro/Engineer.
b. Define analysis parameters and relations to the virtual prototype by Mechanism/Pro and ADAMS
c. Analyze functionalities of ACV in ADAMS.
d. Use Matlab/Simulink to simulate the control circuit and link with the ADAMS/Controls to study the motion dynamics.

e. Simulate the motion sequences generated from the path planner.
f. Output the motion commands to the control card to drive the hardware.
g. Get the feedback information from the encoders of ACV, and provide correction motion to follow the theoretical path.

If the simulation results confirm the feasibility of the motion planning, the operation moves to the hardware control system. To improve the performance, we use MATLAB/RTW (Real Time Workshop) to generate C code of the motor control circuit. The C code compiled by MATLAB/Real Time Windows Target module will control the hardware circuit.

6. Summary and Conclusions

This study introduced a novel design of exploration vehicle, and the design procedure of the motion control using simulation techniques. To apply the high obstacle crossing ability of ACV, a prototype path planner, the Modified Tangentbug Algorithm, is described and implemented to the virtual vehicle. An integrated design environment is established using ADAMS and Matlab/Simulink to simulate the vehicle kinematics, the interaction between the environment and the vehicle, the motion planning algorithm, and the control strategy. ACV can successfully move to the target using the planner MTA in the predefined virtual environment. The result from the virtual prototyping is directly applied to the control of the hardware. In the future, we will continue to improve the path planning algorithm, and install GPS (Global Positioning System) for vehicle positioning. A feedback control scheme will also be investigated since slipping is inevitable in the real world. Finally, a CCD vision system will be put on the vehicle for environment recognition.

7. Acknowledgement

This study is partially supported by the National Science Council in Taiwan under Contract No. NSC 91-2213-E-008 –020.

8. References

[1] S. R. Cunha, A. C. de Matos, F. L. Pereira, (1993) "An Automatic Path Planning System for Autonomous Robotic Vehicles", *Proc. 1993 IEEE Int. Conf. On Robotics & Automation*, pp.1442-1447.
[2] M. Yamamotom M. Iwamura, A.Mohri, (1999) "Quasi-Time – Optimal Motion Planning of Mobile Platforms in the Presence of Obstacles", *Proc. 1999 IEEE Int. Conf. On Robotics & Automation*, Detroit, MI., USA, pp. 2958 – 2963.
[3] Chih-Chiang Lee (2000) *Kinematics simulation and design of Adaptive Configuration Vehicle for 0ff-road applications*, Master Thesis, National Taiwan University of Science and Technology, Taipei, Taiwan.
[4] J. Yu and K. J. Waldron, (1991) "Design of Wheeled Actively Articulated Vehicle", *Proc. Applied Mechanisms and Robots Conference*, Cincinnati, OH., USA, Nov. 5, 1991
[5] I. Kamon, E. Rimon, and E. Rivlin, (1996) "A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots," *Proc. 1996 IEEE Conf. Robotics Automat.*
[6] R. Volpe, J. Balaram, T. Ohm, and R. Ivlev, (1996) "The Rocky 7 Mars Rover Prototype", *Proc. of IEEE/RSJ Conf. Intelligent Robots and Sys.*.
[7] K. Waldron （1995）,"Terrain Adaptive Vehicles", *Journal of Mechanical Design*, Vol. 117B, June, pp. 107-112.
[8] V. Kumar and K.J. Waldron (1989), "Actively coordinated vehicle system" *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 111, n2, pp. 223-231.
[9] G. Klein, K.J. Waldron, and B. Cooper, (1986) "Current Status of Mission/System Design for a Mars Rover", *Unmanned Systems*, Volume 5, No. 1, pp 28~39, Summer 1986.
[10] M.G. Bekker, (1969) *Introduction to terrain vehicle system*, University of Michigan Press.
[11] S.L. Laubach, J.W. Burdick, and L.H. Matthies（1998）, "An Autonomous Path Planner Implemented on the Rocky 7 Prototype Microrover", *Proc. IEEE International Conference on robotics & Automation*, May, pp. 292-297
[12] *ADAMS R12 user manuals*, Mechanical Dynamics Inc., 2001
[13] *Matlab/Simulink user guide*, Mathworks Inc., 2001
[14] S.L. Laubach and J.W. Burdick（1999）, "An Autonomous Sensor-Based Path-Planner for Planetary Microrovers", *Proc. IEEE International Conference on Robotics & Automation*.

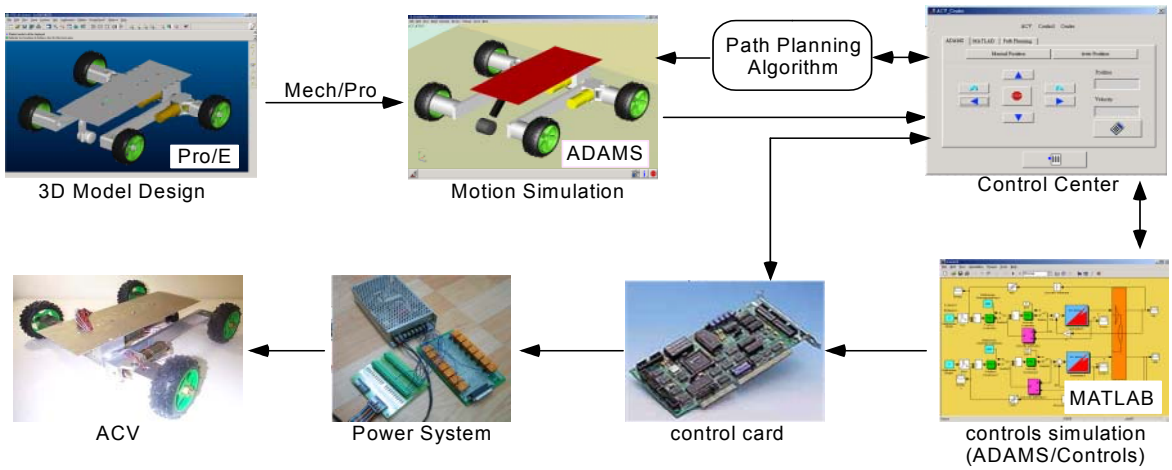


Figure 9: Integration chart of the software and hardware systems