Cooperative Navigation and Path Planning using a Jumping Robot and Ground Rover for Efficient Exploration

S. Paudel*, T. Kubota**

*Graduate School of Engineering, The University of Tokyo
e-mail: paudels@ac.jaxa.jp

**Institute of Space and Astronautical Sciences (JAXA)
e-mail: kubota@isas.jaxa.jp

Abstract

To increase the scientific return of planetary missions in future, cooperation among different types of exploration vehicles like orbiter, lander and rover will be required. Cooperative approach seems to be more effective than using a single rover as each rover adds a different perspective of the environment being explored. This paper presents ongoing works on cooperative Mars exploration using a conventional rover and a jumping robot. A jumping robot provides an additional aerial perspective much closer to the surface as compared to orbiter. Added to that, a jumping robot has greater mobility than a conventional one. The data obtained from these two perspectives are then used for localization, mapping and path planning. In this paper simulation results on visual odometry for jumping robot and cooperative path planning are presented.

1 Introduction

Space exploration has intrigued mankind since the development of the first satellite. It is often believed that other planets and bodies might have better clues regarding the origin of the universe than those currently available to us and these bodies at some time in the past might have supported lives. Among the planets and other bodies, Mars has caught significant interests as it is one of the colsest planets in the solar system, second only to the Venus, and has the most resembling of the environmental conditions. Numerous missions that included flyby missions, orbiters, landers and rovers have been launched with partial success since 1960. Historically orbiters and flyby missions have successfully collected large amount of data required before landers and rovers were launched. Among the landers, the Viking I and II missions are well known.These missions helped obtain high resolution images of martian surface, characterize the structure and composition of the atmosphere and surface and search for evidence of life. The rovers are mobile and thus have much needed advantage over landers[1][2][3]. However, because of the rough terrain including large sized rocks on the Martian surface, mobility is restricted. This limitation on the mobility on rough terrain can be addressed if we have an aerial rover. An aerial rover would explore a wider area in much lesser time. Autonomy on top of that would reduce the dependency on telemtry from the earth station. However, an aerial vehicle capable of sustainable flight in Mars environment is yet to be developed because of its very small atmospheric density (0.6% of earth atmospheric density). To generate enough lift in Martian atmosphere, aerial rovers should be very light, with large wing-span or propellers and fly at greater speed at a much higher altitude. Further, wider wings create complications to fit the rover inside the launching capsule during the deployment. In this regard, this paper proposes a cooperative approach for exploration comprising of a small jumping robot and a conventional ground rover. The jumping robot is capable of repititively jumping for some tens of meters and simultaneously collecting information about the environment. The visual information is used to determine the current position and orientation of the jumping robot and also passed to the ground rover for planning its path.

Cooperative approach seems to be more effective than using a single rover as each rover adds a different perspective for perception of the environment being explored. The jumping robot provides an additional aerial perspective much closer to the surface and hence essentially acts as an extra sensing platform or a scout robot. Stereo visual odometry in ground rover and monocular visual odometry in jumping robot forms the core of the navigation algorithms. Features in each image are merged to give a cloud of features which essentially represents the sparse point cloud map of the environment. As most of the features are associated with rough terrain or obstacles, features that are at untraversible region are regarded as obstacles and path planning is done accordingly. The remaining of this paper is structured as follows: Section 2 discusses the monocular visual odometry as applied to the jumping robot while the combined path planning using D^*Lite is presented in section 3. Simulation results for visual odometry and path planning are presented in section 4 before final conclusion.

2 Visual Odometry and *6-DOF* Pose Estimation

In visual odometry prominent features are tracked along the image sequence to determine the change in the pose of the camera and hence the robot. It is in fact the determination of the amount of rotation and translation undergone by the camera represented by 3×3 rotation matrix, $\mathbf{R}$ and 3×1 translation vector, $\mathbf{t}$ respectively between two consecutive image sequence. For simplicity it is assumed that the robot frame coincides with the camera frame, a valid assumption when the cameras are rigidly fixed on the robot. Stereo vision for the ground rover has been discussed in detail with novel $2-point$ algorithm in [8] while monocular vision assisted by an altimeter for the jumping robot is discussed below.

The set of image samples taken from the jumping robot camera is represented by $I_{0:k} = \{I_0, I_1, I_2 I_{k-1}, I_k\}$. Camera positions at two consecutive time instants $k-1$ and k are related by the rigid body transformation $T_{k,k-1}$ of the following form and as shown in Figure 1:

$$T_{k,k-1} = \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ \mathbf{0} & 1 \end{pmatrix} \tag{1}$$

Where $R_{k,k-1} \in SO(3)$ is the rotation matrix and $t_{k,k-1} \in R^{3\times1}$, the translation vector. x_k, x_{k-1} are the projection of point P in the image plane at time k and $k-1$ respectively. The set $T_{1:n} = \{T_{1,0}, T_{2,1}, ..T_{n,n-1}\}$ contains all subsequent motions. Finally, the set of camera poses $C_{1:n} = \{C_0, C_1, ..., C_n\}$ contains the transformations, T_k $(k = 1, ..., n)$ and therefore, $C_n = C_{n-1}T_n$, with C_0 being the camera initial pose [10]. The geometric constraint between two consecutive images I_{k-1} and I_k can be obtained from intrinsic projective geometry and called as epipolar constraint [9] and written as:

$$x_k^T E_k x_{k-1} = 0 \tag{2}$$

$$E_k \simeq \hat{t}_k R_k \tag{3}$$

Where the E matrix, a 3×3 matrix of rank 2 is called the essential matrix and $\hat{t}_k$ is skew symmetric matrix of translation vector, t. The equivalent sign indicated that it is accurate up to a scale factor since the Epipolar constraint is equally valid when multiplied by a scalar. Matrix E can be obtained by five point algorithm [11] or a simpler eight-point algorithm[12]. To make sure that the points are not among the outliers, a robust outlier rejection algorithm, Random Sample and consensus (RANSAC)[13] is used at each iteration in finding the correct E matrix. In general there are four different solution for R, t for one essential matrix; however, the correct R, t pair can be identified using triangulation. The pair which corresponds to image point lying behind both the image planes or conversely the pair that results in positive depth of the points in both image planes is the correct R, t pair.

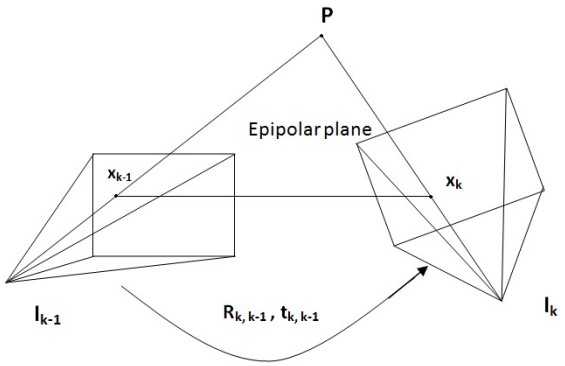


Figure 1. : Transformation between two consecutive image frames.

Since the determined translation vector is accurate only up to a scale factor, the next step is to determine the scale factor. This is where the altimeter is used. The scale factor is determined by dividing the actual altitude gained or lost in the corresponding image pair by the z-component of the translation vector in the world coordinate system. Once the scale factor is determined the translation is scaled and the pose of the robot is updated. The scale factor, λ is given by (4).

$$\lambda = \frac{z_k^w - z_{k-1}^w}{t_z^w} \tag{4}$$

$$t^w = R_{c,k-1}^w t_{c,k} \tag{5}$$

where, z_k^w and z_{k-1}^w are the altitude sensor output at time k and $k-1$ respectively (superscript w represents world coordinate system) while $R_{c,k-1}^w$ is the transformation matrix from camera frame at time $k-1$ to the world co-ordinates and $t_{c,k}$ is the incremental translation of camera from time instant $k-1$ to k and t^w is its representation in world coordinate. The slightly modified visual odometry algorithm for the jumping robot can be summarized as:

1. Detect features, f_{k-1}, in image I_{k-1}
2. Track the features f_{k-1} in image I_k and call it f_k
3. Evaluate the essential matrix E using $5-point$ or $8-point$ algorithm for image pair I_{k-1} and I_k
4. Use SVD of E to determine R matrix and $\hat{t}$ matrix
5. Obtain absolute scale factor from altitude measurements and scale the translation vector
6. Concatenate the transformation
7. Repeat from 1.

2.1 Reducing Error Propagation

In monocular visual odometry, the pose estimation error propagates over time as the final pose is derived form the earlier poses in a cumulative fashion. So it is necessary that the pose estimation error is minimized at each instant. Determination of the scale factor at each image frame is often prone to higher errors as the translation of the camera, equivalently the baseline, is not significantly high. To reduce each individual error so that the cumulative error is within the tolerable limit, it is often advantageous to track features for as long a possible, so that the motion estimation can be performed over an extended baseline and the location of tracked features in the image frame have significantly changed. Since the information about the rotation of the jumping robot is not known *apriori* and the rotation cannot be controlled during the jump, it is thus necessary to perform frequent estimation across relatively short intervals which makes rapidly builds up the estimation errors unavoidable. Apart from the errors accumulated overtime during the jump, errors also result while landing due to the landing impact. In such case, searching for the features from the keyframes just before landing might prove to be beneficial. However, this case is out of the scope of this paper.

2.2 Computational Consideration

The most computationally demanding step in the visual odometry is the outlier rejection algorithm, RANSAC[13]. The number of iterations, N, to achieve a given probability of success is given by equation (6).

$$N = \frac{log(1-p)}{log(1-(1-e)^s)} \tag{6}$$

where, p is the probability of success or the proportion of inliers, e is the proportion of outliers in original data and s is the number of initial sample points taken. Figure 2 shows the case for success probability, $p = 0.9$. It shows that the number of iteration increases exponentially with the increase in the number of initial sample points as well as the outlier ratio in the original data and hence the smaller the number of initial sample points the faster the iteration ends. This implies 5−point algorithm[11] performs faster than the 8−point algorithm[12].

3 Incremental Path Planning

Incremental path planning methods reuse the information from previous path planning instance in order to find new path when obstacles are detected. And the search is much faster as compared to solving for the new path from the scratch. At every instant the rover sensors are looking for information about the obstacles around the rover path so that a feasible path can be constructed avoiding collisions. While the mobility of the ground rover

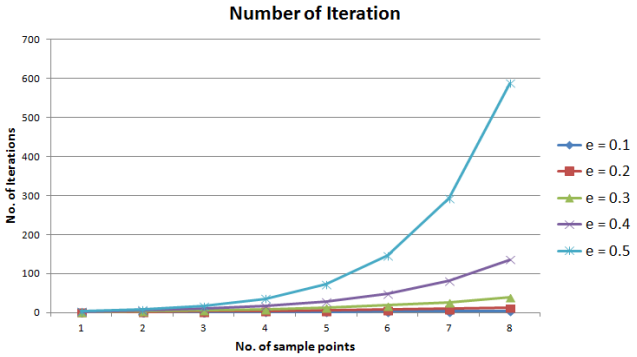


Figure 2. : Number of iterations.

is limited in obstacle-rich region, the jumping robot can jump over the obstacles. At every keyframes the jumping robot sends the feature locations to the ground rover which then incorporates this information to expand the feature map. As obstacle being rough in texture and size, features are clustered around the obstacle in the image. Taking account of current ground rover position, if the features are clustered above the height traversible by the rover, such feature cluster is identified as obstacle and assigned a high cost while the features that lie below the height traversible by the rover are discarded. Once the obstacle map is created from the feature data from both the robots, the ground rover uses D^*Lite algorithm to calculate the feasible path towards the goal. D^*Lite algorithm is built upon Lifelong Planning A^* algorithm, wherein only some grid cells in the map that are affected by the detection of new obstacles are updated and hence is much faster and applicable for real-time planning applications. The current implementation of D^*Lite is taken from [15].

4 Simulation Study

A simulation model of ground rover and jumping robot in *Gazebo* simulator is shown in Figure 3. The ground rover model is of *Cuatro* rover, a small sized four-wheeled rover testbed for planetary exploration developed at ISAS/JAXA[5]. It is provided with stereo cameras and IMU for perception while an optional GPS sensor is available to get the ground truth data for comparison with experimental data. Each wheel is independently driven and steered allowing the robot to make a perfect rotation about its center point. Simulation model is realized using multiple links connected by joints, actuators and sensor models to make it resemble as close as possible to the real rover.

The jumping robot has been inspired from Sand Flea robot [16] developed by Boston Dynamics which uses CO_2 piston engine. The robot sets itself to a launching angle before the CO_2 piston engine pushes the ground to lift the robot flying in the air. In the simulation however,

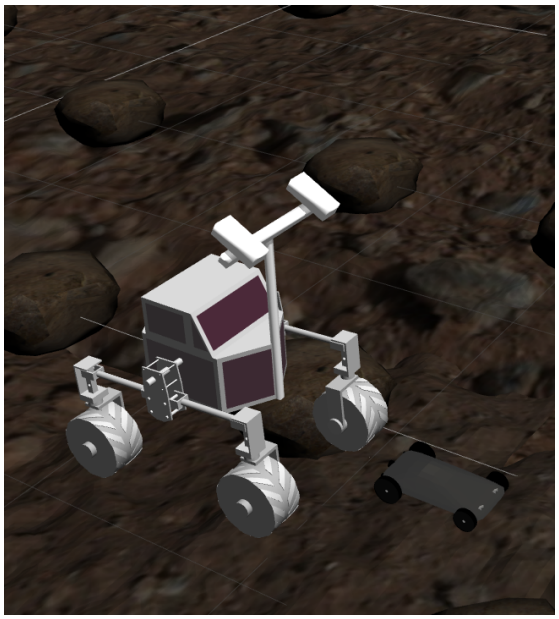


Figure 3. : A ground robot and jumping robot in *Gazebo* Simulator.

the robot is first set to a launching angle and an initial launching velocity is applied. The jumping robot has a minimal sensing system consisting of a monocular camera at its rear end facing backward and a pressure based altimeter as the atmospheric pressure variation with altitude and time of day has been well known [4][6][7].

The simulation is basically carried out in *Robot Operating System (ROS)* because of its strong framework for exchanging information across different nodes as *ROS* messages and visualization in done in open source *Gazebo* simulator. Simulation of D^*Lite algorithm is done independently once the obstacle locations are identified.

4.1 Simulation Environment

The terrain is realized from the height map of the terrain data. The terrain data having been taken from a greater height has smaller resolution. Therefore to obtain realistic terrain, a real image resembling Mars surface is used to wrap the terrain resulting from the height map. Similarly, rocks and boulders are realized and added as required. The terrain has been created in open-source software called *Blender*[18] and exported in *Collada* format for visual appearance and *stl* format for collision geometry. These two formats are understood by open-source simulators like *Gazebo* for visualization.

Figure 4 shows feature tracking over image sequences of the simulated terrain as taken by the jumping robot camera facing backward in *ROS*.

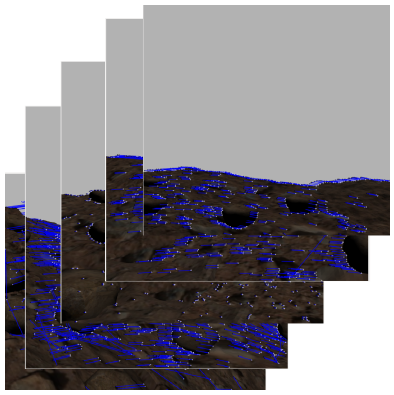


Figure 4. : Tracking features along image sequence

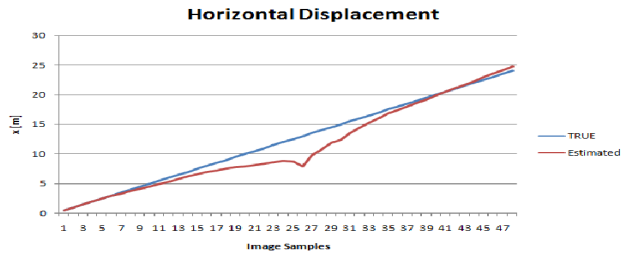
Table 1. : **Simulation Parameters for Jumping Robot**

Initial velocity	$10m/s$
Initial Launch angle	60^0
Accleration (Gravity)	$3.711m/s^2$
Angular velocity	$0.1rad/s$
Camera resolution	$320 \times 320 pixels$
Camera FOV	$45^0 \times 45^0$
Camera focal length	521.78
Frame Rate	$10fps$
Max. number of Features	1000
Altimeter noise	Gaussian ($\mu = 0; \sigma = 1m$)

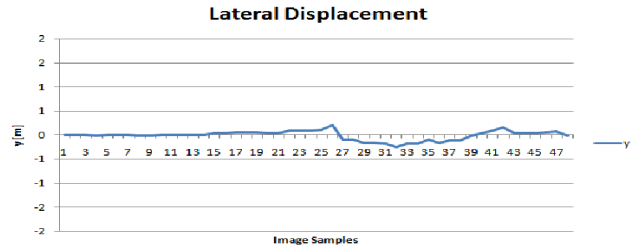
4.2 Pose Estimation of Jumping Robot

Pose estimation is carried out using appropriate functions in *OpenCV*. Table 1 shows the key simulation parameters for the jumping robot. For the simulation, the motion during the jump is assumed to be along positive $x-$direction and positive $z-$direction while the motion along $y-$direction is constrained. Similarly, rotation along $z-axis$ and $x-axis$ are also constrained. This lets us use the perfect jump on the $xz-$plane although the algorithm is able to estimate any arbitrary rotation or translation provided that enough features are available in each frame. Figure 5 shows the simulation result of pose estiomation of jumping robot. Red curve shows the true values while blue curve shows the estimated values. At around the maximum height, the error in $x-$position is the highest and is due to the fact that the detected features are far away and hence greater error. Similar is the case for $y-$position, which ideally should have been zero as the horozontal motion is constrained along the $x-$direction only. However, the simulation results shows that the position error is within the acceptable limit. Figure 5(d) and (e) show the simulation result of the orientation of the jumping robot. The pitch angle of the robot changes gradually as expected from the initial launch angle of 60^0

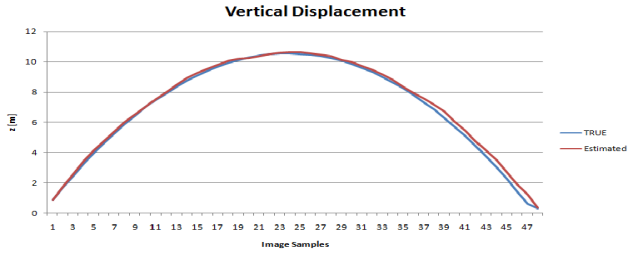
while the roll and yaw error are within acceptable limit.



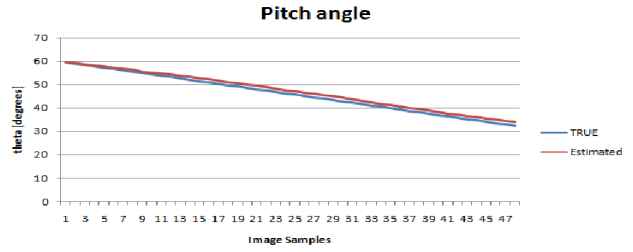
(a)



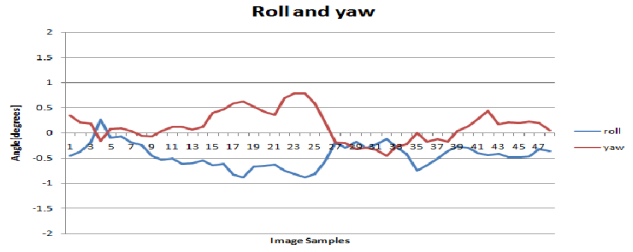
(b)



(c)



(d)



(e)

Figure 5. : Position and Orientation of jumping robot (a) Translation along $x-axis$, (b) Translation along $y-axis$ (error), (c) Translation along $(z-axis)$ (d) Change in pitch angle and (e) Roll and yaw error

4.3 Path Planning

At current time path planning is implemented independently. The terrain is divided into uniform square grid cells. Each grid cells is either an obstacle or is traversable which is not known *apriori*. A robot can move in any of its eight neighboring cells. A grid cell is designated as an obstacle if cluster of features are in and around it. Because of the limited range of the sensing system in the ground rover, it is unable to detect obstacle that lie beyond its region of perception. Apart from these, the ground rover also cannot get information about regions behind obstacles. In such cases the ground rover instructs the jumping rover to fetch information about the region of interest, usually that are beyond its perception region.

A typical path planning is presented in Figure 6. Figure 6 shows the path planning for two cases, first when the ground rover uses information from its own sensors and second when the information from jumping robot is incorporated in the path planning, overlaid on the same image. Here A and B indicate START and GOAL location. When the ground rover detects an obstacle, the grid cells corresponding to that area are denoted by red while the cells whose cost is increased because of the presence of obstacle are denoted by light green and when the jumping robot detects obstacle, corresponding cells are denoted by brown and the cells whose cost in increased are denoted by dark green. The lighter blue line (upper) shows the path generated when the ground rover is acting alone while dark blue line (lower) shows the path planning when the ground rover incorporates the information from the jumping robot. As can be seen from the figure that the lower blue line representing the resulting path when information from both the rovers are incorporated is feasible while the upper blue line seems infeasible. However, it does not rule out the possibility that the ground rover on its own is capable to reach the GOAL B.

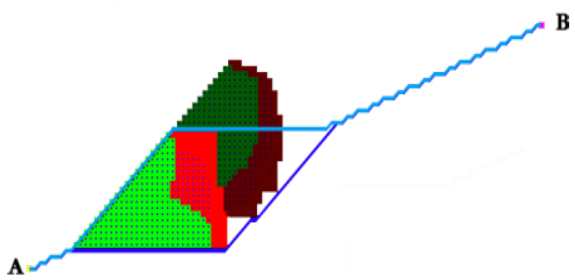


Figure 6. : Path planning between START A and GOAL B.

5 Conclusions

In this paper, a cooperative approach for planetary exploration is proposed using a conventional rover and a jumping robot. The jumping robot has the advantage of higher mobility as it can jump over higher obstacles as compared to ground rover. When the ground rover cannot get enough information, usually when it encounters obstacle that occlude its sensing system, it instructs the ground rover to jump in the direction and acquire information from an aerial perspective. Visual odometry assisted by altimeter for $6-DOF$ absolute pose estimation of the jumping robot was proposed and simulation results of pose estimation during a single hop in $xz-plane$ was presented. Immediate future works include the inclusion of local bundle adjustment routine in visual odometry, integration of path planning and visual odometry into one system and finding out ways to solve the visual odometry when the camera faces the sky. Finally the system will be tested in the real test bed.

References

[1] M. Bajracharya, M.W. Maimone, D. Helmick, "Autonomy for Mars rovers: Past, present, and future," Computer, vol. 41, Issue 12, Dec. 2008, pp. 44-50.

[2] B.K. Muirhead, "Mars rovers, past and future," IEEE Proceedings, Aerospace conference, Mar. 2004.

[3] J. Bell, "Mars Exploration: Roving the red planet," Nature 490:International weeklty journal of science, pp.34-35, Oct. 2012.

[4] S.L. Hess, J.A. Ryan, J.E. Tillman, R.M. Henry, C.B. Leovy, "The annual cycle of pressure on mars measured by Viking landers 1 and 2," Geophysical Research Letters, Vol. 7, Issue 3, March 1980, pp. 197-200.

[5] P.G. Jayasekara, G. Ishigami, T. Kubota, "Testing and validation of autonomous navigation for a planetary exploration rover using open source simulation Tools," in Proc. of the 11^{th} Intl. Symp. on Artificial Intelligence, Robotics and Automation in Space, Turin, Italy, Sept. 2012.

[6] http://mars.jpl.nasa.gov/MPF/science/atmospheric.html, Last accessed Feb. 24, 2014.

[7] http://www.nasa.gov/mission_pages/msl/multimedia/pia16080.html. Last accessed Feb. 24, 2014.

[8] K. Otsu,"Stereo visual odometry for field robots in natural terrain," Master Thesis, The University of Tokyo, 2013.

[9] R. Hartley, A. Zisserman,"Multiple view geometry in computer vision," pp. 219-243, 2003 http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook1/HZepipolar.pdf Last accessed Feb 24, 2014.

[10] D. Scaramuzza, "Visual odometry [Tutorial]," IEEE Robotics and Automation magazine, vol. 18, Issue 4, Dec, 2011.

[11] D. Nister, "An efficient solution to the five-point relative pose estimation'," IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 26, Issue 6, June 2004, pp 756-777.

[12] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," Nature, vol 26 Issue 6, pp 756-777, Jun 2004.

[13] M. Fischler, R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* vol. 24, Issue 6, June 1981, pp 381-395.

[14] S. Koenig, M. Likhachev, "D^* Lite," AAAI/IAAI 2002 http://idm-lab.org/bib/abstracts/papers/aaai02b.pdf Last accessed Oct. 22, 2013.

[15] https://code.google.com/p/dstarlite/, Last accessed Feb. 24, 2014.

[16] http://www.bostondynamics.com/robot_sandflea.html, Last accessed Oct. 22, 2013.

[17] http://hirise.lpl.arizona.edu/, Last accessed Oct. 22, 2013.

[18] www.blender.org, Last accessed Dec. 20, 2013.