OPTIMIZATION OF LIDAR BASED LANDING HAZARD DETECTION[1]

Andrew E. Johnson[1], Geoffrey Vaughan[1], Po-Ting Chen[1] and Robert Bocchino[1], [1]Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91205 [aej@jpl.nasa.gov]

Abstract. *Landing hazard detection (HD) identifies objects that pose risk to the lander during touchdown like steep slopes that could result in tip-over or protuberances that could damage the lander on contact. Finding safe places to land can be computationally expensive when the data being processed is high resolution and the lander can touch down in any orientation. The presentation details optimizations to an existing hazard detection algorithm that make it amenable to real-time processing on a flight processor. Some of the improvements are generic and can be applied to other space imaging applications.*

Introduction. Hazard Detection (HD) is a landing function that uses data collected on board to identify safe landing sites in real time as the vehicle descends. After detection, the vehicle is diverted to the autonomously selected landing site. The ideal sensor for HD is an imaging lidar that can quickly generate a high resolution elevation map over an area many times the size of the lander. HD is most applicable when the proposed landing site has many small hazards that are unavoidable through landing ellipse placement or when the maps are so coarse that landing hazards cannot be identified from orbit.

Related Work. The Chinese Space Agency has successfully employed HD during lunar and Mars landing albeit by hovering the spacecraft and using a fairly slow scanning lidar[1][2]. NASA is developing HD in technology programs including ALHAT/Morpheus, which successful demonstrated HD on a vertical take-off and landing rocket [3], SPLICE [4] and Europa Lander [5]. The Dragonfly mission plans to use a lidar for HD [6], and missions to Europa, Enceladus and the moon could also benefit.

HD can be computationally complex and time consuming. This presentation will describe algorithmic optimizations applied to the ALHAT HD algorithm [7] that have resulting in a 2x improvement in run-time without hardware acceleration. It will then describe HD performance for two test cases that indicate HD performance is maintained even after significant optimizations.

Hazard Detection Algorithm. HD algorithms take as input a Digital Elevation Map (DEM) and parameters that describe the lander mechanical configuration, lidar performance and GNC targeting performance. The ALHAT HD algorithm was an advance over past algorithms in two ways: it placed the lander down on the terrain to measure lander configuration specific slopes and protuberances under all orientations while also using probabilistic reasoning to deal with elevation errors in the DEM and then touchdown landing location. These improvements came at the cost of significantly more processing time. ALHAT mitigated this issue by running HD on eight cores of a multi-processor compute element.

Recently, multiple optimizations were identified that led to a recoding of the ALHAT HD algorithm and a 2x improvement in run-time. Figure 1 shows the refactored data flow. First holes are identified in the DEM at its original resolution and these holes are filled using an iterative algorithm that relies on the grassfire transform. The filled DEM is converted into two reduced resolution maps that define the lander leg touchdown pad heights and the maximum elevation in a local region. The slope of the lander in every orientation is computed from the pad map and then the maximum elevation map is used to compute the probability that a pixel under the lander could cause roughness hazards. The slope, roughness and holes hazards are then combined to a define an overall safety probability with is convolved with a Gaussian smoothing kernel to model touchdown uncertainties. Finally peaks in this safety map are identified as possible safe landing sites.

Algorithm Optimizations. Described below are the specific improvements that lead to decrease in run-time.

Choose loop order. The original algorithm placed the lander at each pixel in the DEM and then checked each orientation of the lander. Rotating the lander and placing it on the terrain requires multiple floating point calculations. Changing the loop order to put orientation in the outer loop allows a large number of calculations to be done once per orientation instead of repeated at every DEM pixel.

Avoid Transcendentals. Transcendental functions (e.g., log, exp, cos, erf, etc) are calculated on the fly in the compiled code and therefore can be computationally expensive. In the original code, the inner most loop of the HD algorithm converts a distance into a probability using by calling the erf function. It turns out there is fixed number of inputs to this function, so the erf values could be precomputed and stored in a lookup table.

Pre-condition the data. DEMs from lidar data can have holes due to missed detections, inadequate sampling or occlusions that cast shadows when viewing the ground from off nadir. The location of holes is not known a-priori, and they should be treated as hazards. In the original code, holes were checked for at all levels of processing which added processing and branching in the code. In the optimized code, the holes are identified,

[1] This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

filled in and then a DEM with no holes is processed. After the main hazard detection, the holes are marked as hazards so that they can be avoided during site selection.

Minimize if statements. The compiler has trouble optimizing code that has branches so if statements should be reduced to the minimum. Through careful coding it was possible to eliminate if statements related to array bounds checking, and plane fit calculations

Carefully reduce resolution. The typical DEM is N=100s of pixels across and the lander can be M=10s of pixels across. The number of orientation steps is driven by the width of the lander in pixels. Put this all together and the HD algorithm is $O(N^2M^3)$. If the number of pixels in the DEM and therefore under that lander can be carefully reduced then there will be a pentic improvement in run-time. The trick is to not introduce false positives (selecting a site that is not safe). This is accomplished by constructing two maps of reduced size from the original DEM: a map of lander foot pad heights for slope computations and a map of the maximum elevation in local region for roughness computations. The maximum elevation in a region is guaranteed to be the worst roughness under the lander, so this approach is conservative. The size of these maps is controlled by an HD Step Size parameter than can be used to tune HD performance vs run-time.

Results. The HD run-times on an A53 ARM processor vs HD Step Size are shown in Table 1. Obviously the 10cm full resolution of the DEM takes prohibitive computation, but the lower resolution versions are quite reasonable.

Figure 2 compares the final safety maps and safe sites selected for a synthetic lunar terrain (Apollo 12) and a controlled test terrain (Test) that has increasing slope from left to right and hemispherical rock hazards increasing in diameter from top to bottom. The Apollo 12 results show consistent site selection as resolution (and run-time) decreases; the top three safest sites are consistently selected in the middle left location in the map. The test terrain always selects sites in the upper left corner which is where the slopes are lowest and the hazards the smallest. Both sets of results indicate optimizations are not degrading performance but more detailed analysis required.

Conclusion. Optimizations to an existing HD algorithm have been described that improve run-time by 2x. The recoding of the ALHAT algorithm has made it amenable to further optimization through auto-vectorization and implementation on multiple processor cores.

Table 1: HD run-times.

HD Step Size	Run-time on A53 ARM Processor
10cm	11600.0s
30cm	55.6s
50cm	4.84s
70cm	1.14s
90cm	0.462s
110cm	0.210s

References.

[1] Xiuqiang Jiang, Shuang Li, Ting Tao, "Innovative hazard detection and avoidance strategy for autonomous safe planetary landing," Acta Astronautica, **126**, 2016. https://doi.org/10.1016/j.actaastro.2016.02.028

[2] Xiangyu Huang et al., "The Tianwen-1 Guidance, Navigation, and Control for Mars Entry, Descent, and Landing", *Space: Science & Technology*, 2021. https://doi.org/10.34133/2021/9846185

[3] N. Trawny et. al., "Flight testing a Real-Time Hazard Detection System for Safe Lunar Landing on the Rocket-Powered Morpheus Vehicle," *Proc. AIAA SciTech Conference,* 2015. https://doi.org/10.2514/6.2015-0326

[4] R. Sostaric et al., "The SPLICE Project: Safe and Precise Landing Technology Development and Testing," *AIAA Scitech Forum,* 2021. https://doi.org/10.2514/6.2021-0256

[5] N. Trawny et al., "The Intelligent Landing System for Safe and Precise Landing on Europa," *Proc. AAS Guidance Navigation and Control Conference (AAS-17-038),* February 2017.

[6] R. Lorenz, et al., "Dragonfly: A rotorcraft lander concept for scientific exploration at Titan," *Johns Hopkins APL Technical Digest*, **34**. 374-387.

[7] T. Ivanov, A. Huertas, and J. Carson, "Probabilistic Hazard Detection for Autonomous Safe Landing," *Proc. AIAA GN&C Conf.*, 2013. https://doi.org/10.2514/6.2013-5019

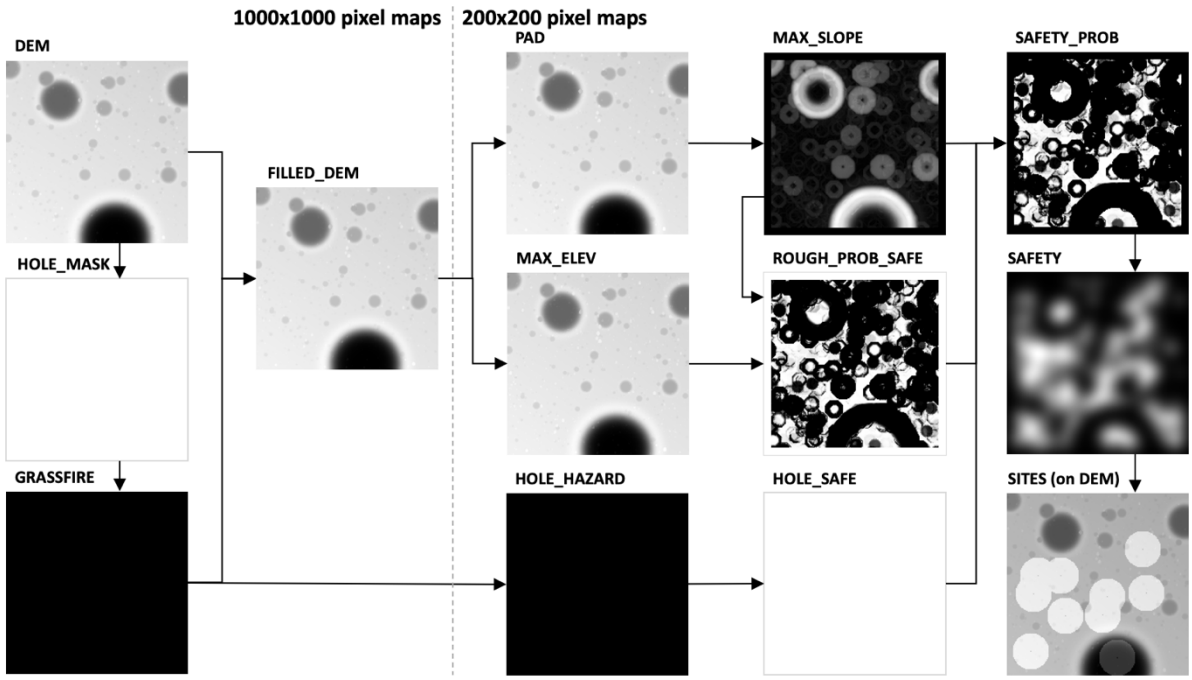

Figure 1. Hazard Detection algorithm data flow.

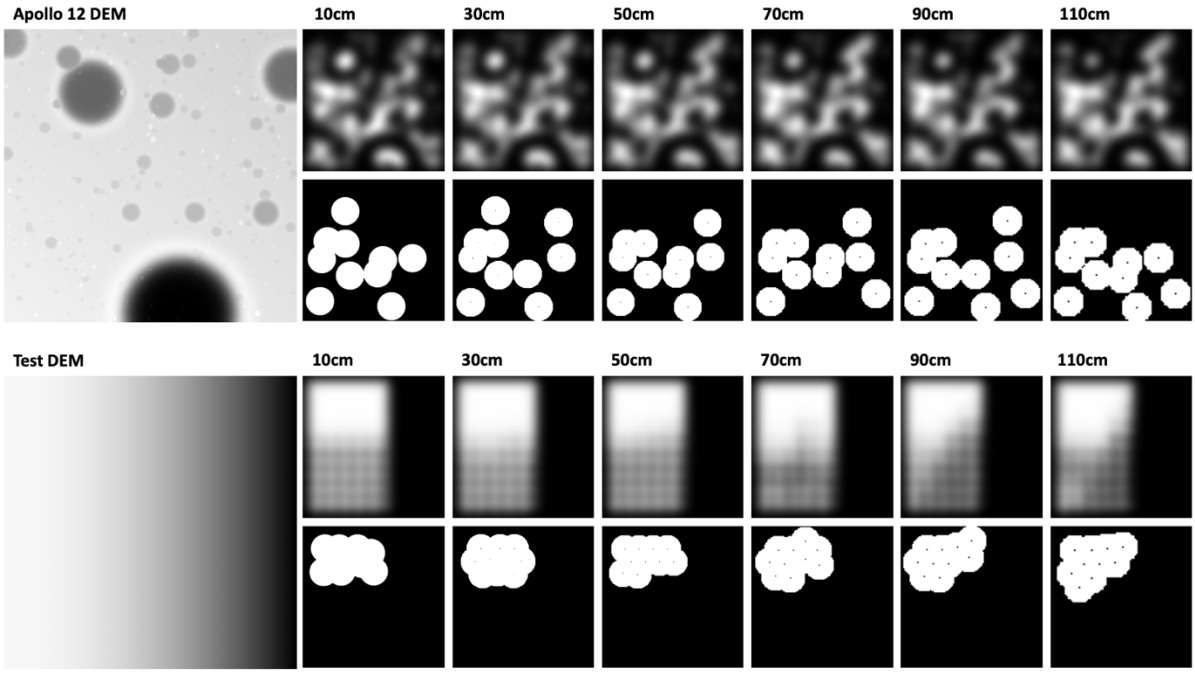

Figure 2. Two examples of safe sites vs. HD step size.