(Preprint) AAS 20-504

AUTONOMOUS SELECTION OF SPACECRAFT LANDING LOCATION ON HAZARDOUS SMALL BODIES

Joshua D. Nelson* **and Hanspeter Schaub†**

In recent years, there has been a great deal of research and development pertaining to the autonomous landing of spacecraft on small bodies, such as asteroids. The capabilities to identify and avoid large rocks and other hazards on the surface of small bodies has seen significant improvement, however most modern techniques search for a location on the surface that contains no hazards within a scaled circular/elliptical footprint. A challenge with this approach is that such acceptable landing locations may be few and far between, or may not even exist at all, on asteroids with highly hazardous terrain. This paper proposes the use of a geometrically conforming footprint to significantly widen possible landing regions. A technique is formulated as the foundation for an autonomous landing location selection algorithm that utilizes such a footprint. This technique offers coarse and fine variations for determining a landing location, both with their own pros and cons. An algorithm that utilizes this technique is constructed, and preliminary test results are presented. These results highlight the differences between coarse and fine variations, and show the current state of the technique to have a 94.3% success rate. Finally, improvements and the future development of this technique are discussed.

INTRODUCTION

Since the turn of the 21st Century, scientific interest in landing spacecraft on small celestial bodies has been on the rise. To further understand the origin and makeup of the Solar System, spacecraft are being sent out to these bodies to perform in situ analysis or, more recently, sample extraction and return. Several missions, such as *Rosetta*, *Hayabusa2*, and *OSIRIS-REx*, have been mounted where the Entry, Descent, and Landing (EDL) phase is critical to mission success. Missions such as these have shown that comets and asteroids contain a large amount of hazardous terrain, such as large rocks and steep slopes, often in the most scientifically interesting regions.[1] Due to the long ground communication delay that is experienced so far from Earth, EDL operations have a limited real-time human input and thus require a certain level of autonomy. The process of locating a safe place to land, known as Hazard Detection and Avoidance (HDA), becomes very challenging due to the abundance of hazardous terrain, especially if the spacecraft must do so autonomously during descent.

Many studies and developments of HDA focus primarily on hazard detection, which for a long time has been the deciding factor for successful landings. There are methods that attempt to match

*Graduate Research Assistant, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO, 80309 USA, Nelson.Joshua@colorado.edu

†Glenn L. Murphy Chair of Engineering, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, 415 AERO, Colorado Center for Astrodynamics Research, Boulder, CO 80309.

identified features on a 2D image to database of known hazards, such as Yu and Cui's affine invariant matching algorithm.[2] Other methods involve the construction of a Digital Elevation Model (DEM) with either LIDAR or stereo-vision techniques.[3] Recently proposed techniques utilize artificial neural networks in combination with vision based sensors to identify hazards.[4] While the further development of hazard detection techniques is of great importance, many of these approaches simply choose an area that is free of hazards as the landing site. Few have explored landing area selection criteria beyond the lack of hazards. Wei et al. provide a method for avoiding regions that are closed environments, such as craters with a flat interior that may otherwise be selected as a safe landing location.[5] Further, Cui et al. propose a safety index method in which hazardous terrain is classified with varying levels of safety, and fuel consumption and touchdown performance are factored into an optimization problem for landing site selection.[6] However, these methods all contain the same constraint on their outcome: they search for landing locations that fit a scaled circular/elliptical footprint.

The use of an elliptical footprint presents a major issue in that such acceptable landing locations may be few and far between, or may not even exist at all, on asteroids with highly hazardous terrain. Historically, these elliptical footprints were necessary to reduce online computation time and to account for uncertainties in surface geometry and landing pose. However, recent innovations in both the hardware and software involved with hazard detection have greatly reduced these uncertainties.[4,7–9] Alongside the increasing capabilities of space-flight ready CPUs,[10] these modern hazard detection techniques allow the possibility of using geometrically conforming footprints, as seen in Figure 1, for landing among hazardous terrain.

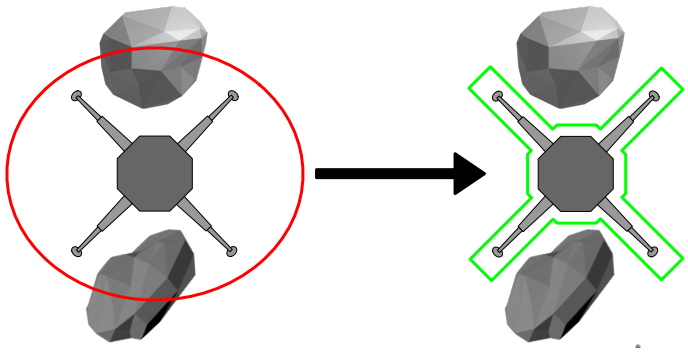


Figure 1. An elliptical footprint (left) compared to a geometrically conforming footprint (right).

This paper investigates a novel landing technique that utilizes a geometrically conforming footprint for landing site selection. This technique is to be used in conjunction with a hazard detection technique that provides a DEM in which representative 3D models of hazards can be extracted. The following section produces a method to find the closest landing pose to a desired location, such that the lander is not intersecting any hazards. Two variants of this method are discussed, and are labeled as a coarse search and fine search respectively. Next, a numerical algorithm utilizing a dimensionally reduced version of this method is developed and then test results are presented. Lastly, conclusions on the potential of this concept and the future steps in its development are expounded.

PROBLEM FORMULATION

Let there be two sets of input convex polyhedra: a set representing the spacecraft lander, and a set representing the surface in some large box containing the desired landing location, denoted as point $\boldsymbol{L}$. Assume both sets are rigid bodies such that all polyhedra within a set maintain a constant pose relative to each other. The reference frame* $\mathcal{F} : \{\boldsymbol{F}, \hat{\boldsymbol{f}}_1, \hat{\boldsymbol{f}}_2, \hat{\boldsymbol{f}}_3\}$ is created for the lander, such that the frame origin $\boldsymbol{F}$ is located at the geometric centroid of the feet and is co-planar with the bottom faces of the feet. The third basis vector of frame $\mathcal{F}$ will be defined to be orthogonal to the feet plane and pointed toward the lander (such that the entire lander geometry is in the $+\hat{\boldsymbol{f}}_3$ direction), while the other two basis vectors will be defined such that $\mathcal{F}$ is right-handed, as seen in Figure 2.

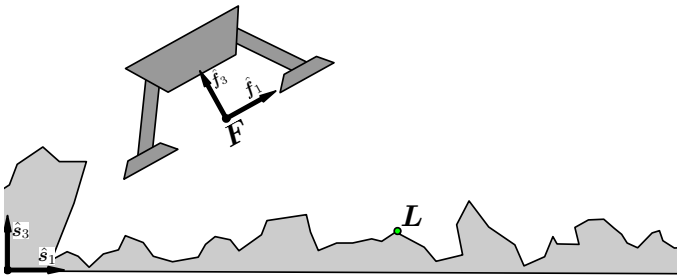


Figure 2. Frame definitions for the lander and surface.

The reference frame $\mathcal{S} : \{\boldsymbol{S}, \hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \hat{\boldsymbol{s}}_3\}$ is defined such that the frame origin $\boldsymbol{S}$ is located at a corner of the plane that exists directly underneath the surface. The basis vectors are defined such that entire surface geometry exists in the first octant, assuming the curvature of the asteroid is locally negligible. The third basis vector of frame $\mathcal{S}$ is chosen to be antiparallel to the local gravity direction, with the other two basis vectors defined such that $\mathcal{S}$ is right-handed.

Let this problem be *initially* cast as the Quadratic Program (QP):

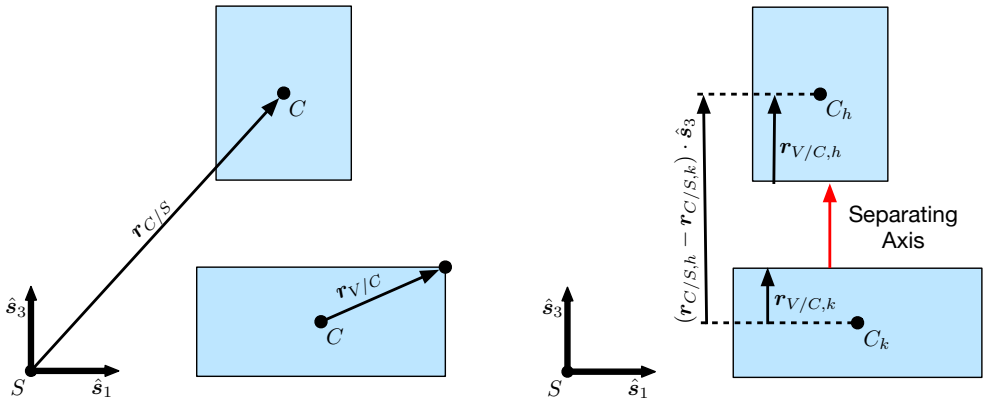
$$\text{Minimize} \qquad V(\boldsymbol{r}_{L/F}) = \frac{1}{2}\boldsymbol{r}_{F/S}^T Q \boldsymbol{r}_{F/S} + \boldsymbol{c}^T \boldsymbol{r}_{F/S}, \tag{1}$$

where $Q \succeq 0$ is the hessian, and $\boldsymbol{c} = -Q\boldsymbol{r}_L$ is the gradient. This QP is further expanded by constraints preventing the intersection of the surface and lander polyhedra. From the definitions of the $\mathcal{S}$ and $\mathcal{F}$ frames, it is assumed that the $\hat{\boldsymbol{f}}_3$ and $\hat{\boldsymbol{s}}_3$ directions should be kept close to parallel, such that the lander does not topple off its feet due to gravity.

The Separating Axis Theorem & Approximate Convex Decomposition

The safety of a selected landing location materializes as the assurance that surface hazards do not penetrate the lander hull. This assurance is quantified by the non-intersection of the lander and surface polyhedra. The foundation of the non-intersection constraints comes from the *Separating Axis Theorem*[11] (SAT); which says that any two convex polyhedra do not overlap if, and only if, there exists at least one spatial axis where the projections of those polyhedra do not overlap. Let $\boldsymbol{r}_{C/S}$ be a vector from the $\mathcal{S}$ frame origin to the centroid $\boldsymbol{C}$ of a polyhedron, and let $\boldsymbol{r}_{V/C}$ be a vector from $\boldsymbol{C}$ to any exterior point on the polyhedron, as seen in Figure 3(a). Then for any two non-intersecting polyhedra h and k, there exists an axis where the projection of $\boldsymbol{r}_{C/S,h} - \boldsymbol{r}_{C/S,k}$ is

*See table at the end for notation



(a) Vector Definitions

(b) Depiction of Separating Axis

Figure 3. A two dimensional example of the Separating Axis Theorem.

greater in magnitude than $\boldsymbol{r}_{V/C,k} + \boldsymbol{r}_{V/C,h}$, where $\boldsymbol{r}_{V/C,k}$ is the same direction as the projection, and $\boldsymbol{r}_{V/C,h}$ is the opposite direction. This can be seen in Figure 3(b) where the separating axis is shown to be along $\hat{\boldsymbol{s}}_3$.

While the SAT provides a mathematically reliable method to determine if two polyhedra intersect, note that it is defined specifically for *convex* polyhedra. This presents a natural issue, as surface hazards and spacecraft landers are seldom convex in their geometry. This issue can be avoided by constructing a convex hull around any non-convex polyhedra. However, this idea has substantial drawbacks; a convex hull over a highly non-convex polyhedra would result in a large amount of empty space removed from consideration when fitting two polyhedra together. Thus, a more refined application of convex hulls can be used in form of a technique known as *Approximate Convex Decomposition*[12] (ACD). The general idea behind ACD is to subdivide highly non-convex polyhedra into multiple polyhedra, that are less non-convex than their parent, before applying a convex hull to each new polyhedra. The number of subdivided polyhedra is determined by the maximum allowable level of non-convexity, which is parameterized by the user.

For the landing technique being proposed in this paper, the spacecraft lander and surface hazards are decomposed into convex polyhedra. All convex polyhedra created from the ACD are divided into two sets; the surface set contains all polyhedra representing the surface hazards, and the lander set contains all polyhedra representing the spacecraft lander. As for what specific ACD algorithm to use; many algorithms have been developed around the ACD technique, some of which who focus on computational speed. Investigation and selection of an ACD algorithm is out of the scope of this paper, and the following sections assume the decomposition step has already been completed.

Constraint Formulation

The existence of a separating axis between a surface polyhedron k and a lander polyhedron h can be easily represented with the inequality

$$(\boldsymbol{r}_{C/S,h})_\beta - (\boldsymbol{r}_{C/S,k})_\beta \geq (\boldsymbol{r}_{V/C,k})_\beta + (\boldsymbol{r}_{V/C,h})_\beta, \tag{2}$$

where the subscript β represents some direction in $\mathcal{S}$ and $(\cdot)_\beta$ is the scalar projection of a vector on β. Testing for a separating axis in only one direction is insufficient, so this inequality must be

applied over an encompassing set of directions (such a set is defined in a following subsection). When testing over a set of directions, the SAT says that the above inequality only needs to be satisfied in one direction. In fact, this inequality will fail in some other directions, rendering this problem infeasible. Therefore, Equation (2) must be adjusted in the following way:[13]

$$(\boldsymbol{r}_{C/S,h})_\beta - (\boldsymbol{r}_{C/S,k})_\beta \geq (\boldsymbol{r}_{V/C,k})_\beta + (\boldsymbol{r}_{V/C,h})_\beta - D_i(1-\epsilon_{hk,\beta}), \tag{3}$$

$$\sum_\beta \epsilon_{hk,\beta} \geq 1, \tag{4}$$

where D_i is a scalar big-M coefficient and $\epsilon_{hk,\beta} \in \{0,1\}$ are activation decision variables. With a large enough value of D_i and with $\epsilon = 0$, the inequality in Equation (3) becomes always true within the scope of the problem. Thus, for a value of $\epsilon_{hk,\beta} = 1$, the separating axis criteria becomes active along the β direction. As stated previously, the separating axis must exist in at least one direction for the two polyhedra to not intersect, which is enforced with the inequality in Eq. (4). The value of D_i may be chosen to be infinitely large, however its minimum effective value is the length of the longest dimension of the surface area containing known hazards.

These constraints are applied between every convex polyhedron in the lander set h and surface set k (they are not applied between two polyhedra contained in the same set). The centroid positions of lander set are known and constant in the $\mathcal{F}$ frame, however these constraints are evaluated in the $\mathcal{S}$ frame. Thus Equation 3 is modified by

$${}^{\mathcal{S}}\boldsymbol{r}_{C/S,h} = {}^{\mathcal{S}}\boldsymbol{r}_{F/S} + [SF]{}^{\mathcal{F}}\boldsymbol{r}_{C/F,h} \tag{5}$$

where $\boldsymbol{r}_{C/F,h}$ is constant in the $\mathcal{F}$ frame and $[SF]$ is a Direction Cosine Matrix (DCM) that rotates a vector description from $\mathcal{F}$ to $\mathcal{S}$. This alteration now adds the position of $\boldsymbol{F}$ relative to $\boldsymbol{S}$ and the attitude of $\mathcal{F}$ relative to $\mathcal{S}$ as decision variables to this problem.

The attitude representation chosen for this problem are the Modified Rodriguez Parameters[14] (MRP). In order to maintain these constraints as linear, the DCM representation of the MRP must be linearized about some reference frame $\mathcal{R}$. The DCM form of the MRP linearizes to

$$[C(\boldsymbol{\sigma}_{F/R})] = \begin{bmatrix} 1 & 4\sigma_3 & -4\sigma_2 \\ -4\sigma_3 & 1 & 4\sigma_1 \\ 4\sigma_2 & -4\sigma_1 & 1 \end{bmatrix} \quad \text{where} \quad \boldsymbol{\sigma}_{F/R} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix}. \tag{6}$$

In order to avoid more than a 5% error in the DCM introduced by linearization, rotations defined by $\boldsymbol{\sigma}_{F/R}$ cannot exceed 20 degrees, or $\frac{\pi}{36}$ radians, about each axis. Therefore, this problem does not consider the surface frame $\mathcal{S}$ as the frame these MRPs are linearized about. To ensure that the $\hat{\boldsymbol{f}}_3$ and $\hat{\boldsymbol{s}}_3$ directions are kept close to parallel, the angles between the $\hat{\boldsymbol{f}}_1$ and $\hat{\boldsymbol{s}}_1$ directions and the $\hat{\boldsymbol{f}}_2$ and $\hat{\boldsymbol{s}}_2$ directions are constrained to be less than 20 degrees. Therefore, let frame $\mathcal{R}$ be some intermediate frame that relates to $\mathcal{S}$ by a rotation of some reference angle θ_r about the $\hat{\boldsymbol{s}}_3$ axis, denoted by the DCM $[SR]$. To maintain linearity in these constraints, the angle θ_r is held constant when solving for a solution. Therefore, this problem is to be solved several times over iterations of θ_r, and the solution with the lowest cost is chosen as the final solution.

To enforce the discussed constraints on attitude, the following bounds are added to the problem:

$$-\frac{\pi}{36} \leq \sigma_1 \leq \frac{\pi}{36} \qquad -\frac{\pi}{36} \leq \sigma_2 \leq \frac{\pi}{36} \qquad -\frac{\pi}{36} \leq \sigma_3 \leq \frac{\pi}{36}. \tag{7}$$

Note that these bound introduce a double inequality to the problem constraints. In fact, upper and lower bounds on ${}^{\mathcal{S}}\boldsymbol{r}_{F/S}$ must also be introduced to contain the problem within the known region of surface hazards. For consistency in the constraints, and since many effective QP solvers operate on double inequality constraints, an upper bound is added to Equations (3) and (4). First, isolating the decision variables in Equation (3) to one side of the inequality and expressing the vectors in their known frame leads to

$$({}^{\mathcal{S}}\boldsymbol{r}_{C/S,k} + {}^{\mathcal{S}}\boldsymbol{r}_{V/C,k})_\beta - D_i \leq \left({}^{\mathcal{S}}\boldsymbol{r}_{F/S} + [SR][C(\boldsymbol{\sigma}_{F/R})]^T({}^{\mathcal{F}}\boldsymbol{r}_{C/F,h} - {}^{\mathcal{F}}\boldsymbol{r}_{V/C,h})\right)_\beta - D_i\epsilon_{hk,\beta}, \tag{8}$$

$$1 \leq \sum_\beta \epsilon_{hk,\beta}. \tag{9}$$

Next, another big-M value, called D_{sr}, is introduced to Equation (8) as an upper bound. This new value can be the same as D_i; however, if the surface area being considered for landing does not fully contain all known hazards, then D_{sr} may be the length of longest dimension of the surface area being considered. The upper bound for Equation (9) is the number of search directions for a separating axis, labeled β_{max}. Therefore, all constraints defined thus far are

$${}^{\mathcal{S}}\boldsymbol{r}_{F/S,\text{min}} \leq {}^{\mathcal{S}}\boldsymbol{r}_{F/S} \leq {}^{\mathcal{S}}\boldsymbol{r}_{F/S,\text{max}}, \tag{10}$$

$$\boldsymbol{\sigma}_{F/R,\text{min}} \leq \boldsymbol{\sigma}_{F/R} \leq \boldsymbol{\sigma}_{F/R,\text{max}}, \tag{11}$$

$$({}^{\mathcal{S}}\boldsymbol{r}_{C/S,k} + {}^{\mathcal{S}}\boldsymbol{r}_{V/C,k})_\beta - D_i \leq \left({}^{\mathcal{S}}\boldsymbol{r}_{F/S} + [SR][C(\boldsymbol{\sigma}_{F/R})]^T({}^{\mathcal{F}}\boldsymbol{r}_{C/F,h} - {}^{\mathcal{F}}\boldsymbol{r}_{V/C,h})\right)_\beta - D_i\epsilon_{hk,\beta} \leq D_{sr}, \tag{12}$$

$$1 \leq \sum_\beta \epsilon_{hk,\beta} \leq \beta_{\text{max}}. \tag{13}$$

Note that Equation (12) exists for every direction being used to test for a separating axis, with each having a unique decision variable $\epsilon_{hk,\beta}$. For each pairing of surface and lander polyhedra, let the lower bound of Equations (12) and (13) form the vector $\boldsymbol{l}_{hk}$ and the upper bound form the vector $\boldsymbol{u}_{hk}$. Let there be decision variable vectors $\boldsymbol{x} = [{}^{\mathcal{S}}\boldsymbol{r}_{F/S}^T \quad \boldsymbol{\sigma}_{F/R}^T]^T$ and $\boldsymbol{\epsilon}_{hk} = [\epsilon_{hk,1} \quad ... \quad \epsilon_{hk,n}]^T$ for q directions in β. Then Equations (12) and (13) become

$$\boldsymbol{l}_{hk} \leq \begin{bmatrix}\Sigma_{hk} & D_{hk}\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ \boldsymbol{\epsilon}_{hk}\end{bmatrix} \leq \boldsymbol{u}_{hk}, \tag{14}$$

where $\boldsymbol{l}_{hk}, \boldsymbol{u}_{hk} \in \mathbb{R}^{q+1}$, $\Sigma_{hk} \in \mathbb{R}^{(q+1)\times 6}$, and $D_{hk} \in \mathbb{R}^{(q+1)\times q}$. Let the upper and lower bounds of Equations (10) and (11) be $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ respectively. For an example of how these constraints evolve, let there be two lander polyhedra $h \in \{1, 2\}$ and two surface polyhedra $k \in \{1, 2\}$. Then the inequality constraints would be

$$\begin{bmatrix}\boldsymbol{x}_l \\ \boldsymbol{l}_{11} \\ \boldsymbol{l}_{12} \\ \boldsymbol{l}_{21} \\ \boldsymbol{l}_{22}\end{bmatrix} \leq \begin{bmatrix}I_{6\times6} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \Sigma_{11} & D_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \Sigma_{12} & \mathbf{0} & D_{12} & \mathbf{0} & \mathbf{0} \\ \Sigma_{21} & \mathbf{0} & \mathbf{0} & D_{21} & \mathbf{0} \\ \Sigma_{22} & \mathbf{0} & \mathbf{0} & \mathbf{0} & D_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ \boldsymbol{\epsilon}_{11} \\ \boldsymbol{\epsilon}_{12} \\ \boldsymbol{\epsilon}_{21} \\ \boldsymbol{\epsilon}_{22}\end{bmatrix} \leq \begin{bmatrix}\boldsymbol{x}_u \\ \boldsymbol{u}_{11} \\ \boldsymbol{u}_{12} \\ \boldsymbol{u}_{21} \\ \boldsymbol{u}_{22}\end{bmatrix}. \tag{15}$$

Recall that every variable in $\boldsymbol{\epsilon}_{hk}$ is a binary decision variable and thus, with the constraints defined, this problem becomes cast as a Mixed Integer Quadratic Program (MIQP) in the form:

$$\text{Minimize} \qquad V(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + \boldsymbol{c}^T \boldsymbol{z}, \tag{16}$$

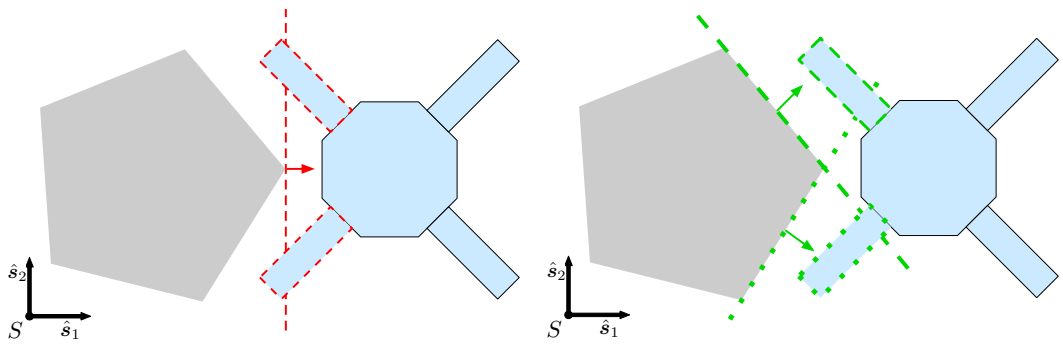
$$\text{Subject To} \qquad \boldsymbol{l} \leq A\boldsymbol{z} \leq \boldsymbol{u}, \tag{17}$$

$$z_i \in \{0, 1\}, \tag{18}$$

where $\boldsymbol{z} \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and $i = 7, ..., n$. Let K and H be the number of polyhedra in the surface set and lander set respectively, and K_q be the sum of the amount of search directions for each surface polyhedron. Then the dimensions of this problem are $n = 6 + HK_q$ and $m = 6 + 2HK_q + HK$.

Search Directions

The set of directions to search for a separating axis must be encompassing; i.e. for two polyhedra sufficiently far from each other anywhere in three-space, there must exist at least one valid direction in the set. An obvious choice for the set are the six basis directions of the $\mathcal{S}$ frame. However, while such a set satisfies the encompassing requirement, it may not provide an ideal result when the two polyhedra are very close to one other. For example, if a separating axis is only checked along the basis directions of $\mathcal{S}$, then the case shown in Figure 4(a) would fail, even though the lander is not intersecting the rock. Think of the separating axis as creating a plane defined by the direction of the axis, and the most extreme point in the axis direction on the polyhedron. Therefore, using only the $\mathcal{S}$ frame basis vectors as separating axes creates a frame-oriented bounding parallelepiped. Luckily, a separating axis can be defined in any direction in three-space, and thus the set of search directions can be chosen freely. Consider a set of search directions defined by the normal vectors that define the planes of one of the polyhedra, which is also an encompassing set. Since only one separating axis needs to exist to prove separation, then this new selection of possible axes allows the previously failed case to pass, as seen in Figure 4(b).



(a) The dashed line represents the plane created by the separating axis in the $\hat{s}_1$ direction

(b) The shape with the dashed/dotted border only passes the separating axis test with the dashed/dotted line

Figure 4. Comparing the effectiveness of the two search methods.

There are pros and cons to using either of the described search direction sets. The set containing the basis directions of $\mathcal{S}$ is consistent throughout the problem space, and is thus simpler to preprocess. The negative aspect of this set is potential restrictions it places on two polyhedra that are

closely placed together. The set containing the normal directions of a polyhedron's faces involves a greater amount of pre-processing, as each pair of polyhedra being compared requires a specific set of normals. Because the surface hazard polyhedra remain constant in the $\mathcal{S}$ frame, they are used to define these search direction sets. Thus, each polyhedron in the surface set has a unique corresponding search direction set that must be defined in pre-processing. However, these types of search direction sets provide a more conforming fit between lander and surface polyhedra, which is ideal in the fine placement of the final landing position. Therefore, the search direction set containing the basis directions of $\mathcal{S}$ is referred to as the *coarse* searching set, and the sets that contain the face normal directions of the surface polyhedra is referred to as the *fine* searching set.

ALGORITHM TO SOLVE FOR THE LANDING LOCATION

The program described in Equations (16-18) is known to be $\mathcal{NP}$-hard[15] due to the presence of integer decision variables. This raises tractability concerns in regards to solving this problem on board a spacecraft. Fortunately, recent studies[16,17] show tractable results for solving MIQPs on embedded systems. These techniques use the Branch & Bound (B&B) approach to solve MIQPs by sub-dividing the problem into a tree of relaxed QP problems which can be searched to find the optimal solution. The performance of this approach depends heavily on the method used to search the tree. The following proposes an algorithm for solving this problem with a B&B approach.

Using the B&B approach allows the binary decision variables in this problem to be abstracted away, and replaced by continuous variables whose value is dictated by the tree of QP problems. Therefore, Equations (16-18) are restructured to

$$\text{Minimize} \qquad V(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + \boldsymbol{c}^T \boldsymbol{z}, \tag{19}$$

$$\text{Subject To} \qquad \begin{bmatrix} \boldsymbol{l} \\ \bar{\boldsymbol{l}} \end{bmatrix} \leq \begin{bmatrix} A \\ \bar{A} \end{bmatrix} \boldsymbol{z} \leq \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}, \tag{20}$$

where $\bar{\boldsymbol{l}}$, $\bar{A}$, and $\bar{\boldsymbol{u}}$ are the continuous representation of Equation (18). For every binary decision variable in the original problem, there is a row in $\bar{A}$ with a value of 1 at the column corresponding to that variable. Initially, the upper and lower bounds are $\bar{\boldsymbol{l}} = -\tau\mathbf{1}$ and $\bar{\boldsymbol{u}} = \tau\mathbf{1}$, where τ is a small feasibility tolerance. Each successive branch in the tree alters a single set of $\boldsymbol{\epsilon}_{hk}$, choosing one $\epsilon_{hk,\beta}$ to have its upper and lower bounds changed to $\bar{l} = 1-\tau$ and $\bar{u} = 1+\tau$. In context to the problem, this process begins by deactivating all of the separating axis constraints defined by Equation (12), and then reactivating one for each combination of surface and lander polyhedra as the tree is explored. Note that a final solution is only accepted if it is found at the maximum depth of the tree, then the constraints defined by Equation (13) become unnecessary, and thus can be removed from the problem. This reduces the dimensionality of this problem such that $\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n$, $\bar{\boldsymbol{l}}, \bar{\boldsymbol{u}} \in \mathbb{R}^m$, $Q, A \in \mathbb{R}^{n\times n}$, and $\bar{A} \in \mathbb{R}^{m\times n}$ with $n = 6 + HK_q$ and $m = HK_q$.

Note that the theory in the previous section does not provide a means of ensuring that the lander's final position has it resting on a surface. As it stands, the formulated problem could result with the lander floating somewhere above the surface. The development of such a condition will be touched upon in future work, and the scope of this paper is to show the functionality of the problem as it stands. To accomplish this, the dimensionality of this problem places a restriction on the $\hat{\boldsymbol{s}}_3$ direction. Only two discrete levels in this direction are considered: the lower level of the surface, where point F is restricted to, and the top of the surface hazards. This allows the problem to be fully tested in the $\hat{\boldsymbol{s}}_1$ and $\hat{\boldsymbol{s}}_2$ directions.

As stated previously, this algorithm assumes that the surface hazards and lander have already been identified and decomposed into sets of convex polyhedra $\{P\}$. Each polyhedron P in a set contains a set of vertices $\{V\}$ and face normal unit vectors $\{N\}$ that describes the polyhedron. Let there be two sets of polyhedra denoted with the subscripts k and h, which contain the polyhedra for the surface and lander respectively. Within a polyhedron, every vertex $\boldsymbol{r}_V$ and face normal $\hat{\boldsymbol{n}}$ is described in the reference frame respective to the polyhedron's set. Furthermore, this algorithm is presented using the fine searching set and may be adjusted to use the coarse searching set by replacing all $\{N\}$ with the set of positive and negative basis vectors of the $\mathcal{S}$ frame.

Algorithm 1: Pre-Processing The Geometry

Input: The surface and lander sets of polyhedra $\{P_k\}$ & $\{P_h\}$, where each polyhedron contains a set of vertices $\{V\}$ and face normal unit vectors $\{N\}$ in their respective frames.

1 **foreach** $P \in \{P_k\} \cup \{P_h\}$ **do**
2 $\quad \boldsymbol{r}_C = \mathbf{0}$; $j = 0$; $\{\rho\} = \emptyset$
3 $\quad$ **foreach** $\boldsymbol{r}_V \in \{V\}$ **do**
4 $\quad\quad \boldsymbol{r}_C = \boldsymbol{r}_C + \boldsymbol{r}_V$
5 $\quad\quad j = j + 1$
6 $\quad \boldsymbol{r}_C = \dfrac{\boldsymbol{r}_C}{j}$
7 $\quad$ **if** $P \in \{P_k\}$ **then**
8 $\quad\quad$ **foreach** $\hat{\boldsymbol{n}} \in \{N\}$ **do**
9 $\quad\quad\quad \{\rho\} \leftarrow \max\{\hat{\boldsymbol{n}} \cdot (\boldsymbol{r}_V - \boldsymbol{r}_C),\ \forall \boldsymbol{r}_V \in \{V\}\}$
10 $\quad$ **else if** $P \in \{P_h\}$ **then**
11 $\quad\quad$ **foreach** $i \in \{1, 2, 3\}$ **do**
12 $\quad\quad\quad \{\rho\} \leftarrow \max\{\hat{\boldsymbol{f}}_i \cdot (\boldsymbol{r}_V - \boldsymbol{r}_C),\ \forall \boldsymbol{r}_V \in \{V\}\}$
13 $\quad P \leftarrow (\boldsymbol{r}_C, \{\rho\})$

Before the elements in Equation (17) can be assembled, the values of $\boldsymbol{r}_{V/C}$ and $\boldsymbol{r}_C$ (this notation is truncated from $\boldsymbol{r}_{C/S}$ and $\boldsymbol{r}_{C/F}$ for surface and lander polyhedra respectively) must be found for each polyhedron, as shown in Algorithm 1. The value of $\boldsymbol{r}_C$ is simply the vertex-weighted centroid of each polyhedron. As previously discussed, $\boldsymbol{r}_{V/C}$ must be the vector from the centroid to the edge of the polyhedron in the search direction. This value is easier found on a polyhedron from the surface set, since the search directions are defined in the $\mathcal{S}$ frame. As seen on line 9, the largest projection of the vertices relative to the centroid are found for each search direction vector. These scalar values are stored in a set $\{\rho\}$ that aligns with the search direction set. Due to the variability in attitude between the lander and the surface, $\boldsymbol{r}_{V/C}$ for a lander polyhedron cannot be predetermined using the search directions. Instead, the set $\{\rho\}$ is filled with the largest projections in the basis directions of the $\mathcal{F}$ frame.

With the geometry pre-processed, the vectors and matrices in Equations (19) and (20) are constructed in the manner depicted in Algorithm 2. Lines 1-4 performs initialization, followed by the hessian and gradient value assignment. Since this problem is only concerned with minimizing the distance of $\boldsymbol{r}_{F/S}$ to some objective position $\boldsymbol{r}^*_{F/S}$ in the $\hat{\boldsymbol{s}}_1$ and $\hat{\boldsymbol{s}}_2$ directions, the hessian should only be non-zero (and positive) at the elements $Q_{1,1}$ and $Q_{2,2}$. Line 6 implements the upper and

lower bounds for $\boldsymbol{r}_{F/S}$ and $\boldsymbol{\sigma}_{F/R}$, with $r_{F/S,3}$ having both an upper and lower bound of zero. The solver is warm started with the vector $\boldsymbol{z}_0$, which contains the objective position. Next, each combination of surface and lander polyhedra and their corresponding search directions are looped through to build the constraints defined by Equation (12). In the matrix A, every ith row is the start of a new polyhedra combination and β iterates through that combination's search directions. To improve the readability of lines 16-18, the notation of $\boldsymbol{r}_\alpha$, $\boldsymbol{r}_{\sigma_1}$, $\boldsymbol{r}_{\sigma_2}$, and $\boldsymbol{r}_{\sigma_3}$ is defined for vectors rotated from the $\mathcal{F}$ frame to the $\mathcal{S}$ frame, such that

$$\begin{aligned}[SR][C(\boldsymbol{\sigma}_{F/R})]^{T\mathcal{F}}\boldsymbol{r} = &{}^{\mathcal{S}}\begin{bmatrix} r_1\mathrm{c}\theta_r + r_2\mathrm{s}\theta_r \\ -r_1\mathrm{s}\theta_r + r_2\mathrm{c}\theta_r \\ r_3 \end{bmatrix} + {}^{\mathcal{S}}\begin{bmatrix} -4r_3\mathrm{s}\theta_r \\ -4r_3\mathrm{c}\theta_r \\ 4r_2 \end{bmatrix}\sigma_1 \\ &+ {}^{\mathcal{S}}\begin{bmatrix} 4r_3\mathrm{c}\theta_r \\ -4r_3\mathrm{s}\theta_r \\ -4r_1 \end{bmatrix}\sigma_2 + {}^{\mathcal{S}}\begin{bmatrix} 4r_1\mathrm{s}\theta_r - 4r_2\mathrm{c}\theta_r \\ 4r_1\mathrm{c}\theta_r + 4r_2\mathrm{s}\theta_r \\ 0 \end{bmatrix}\sigma_3 \\ = &{}^{\mathcal{S}}\boldsymbol{r}_\alpha + {}^{\mathcal{S}}\boldsymbol{r}_{\sigma_1}\sigma_1 + {}^{\mathcal{S}}\boldsymbol{r}_{\sigma_2}\sigma_2 + {}^{\mathcal{S}}\boldsymbol{r}_{\sigma_3}\sigma_3. \end{aligned} \tag{21}$$

Using this notation, the coefficients in A for $\boldsymbol{\sigma}_{F/R}$ are defined as

$$A_{i+\beta,4} = \hat{\boldsymbol{n}} \cdot \left(\boldsymbol{r}_{C/F,h,\sigma_1} - \boldsymbol{r}_{V/C,h,\sigma_1}\right), \tag{22}$$

$$A_{i+\beta,5} = \hat{\boldsymbol{n}} \cdot \left(\boldsymbol{r}_{C/F,h,\sigma_2} - \boldsymbol{r}_{V/C,h,\sigma_2}\right), \tag{23}$$

$$A_{i+\beta,6} = \hat{\boldsymbol{n}} \cdot \left(\boldsymbol{r}_{C/F,h,\sigma_3} - \boldsymbol{r}_{V/C,h,\sigma_3}\right). \tag{24}$$

This part of the algorithm also includes the construction of the B&B searching tree. For the overall structure of the tree; every level of depth represents a unique combination of surface and lander polyhedra, with the choice of $\epsilon_{hk,\beta}$ to activate existing laterally. Starting with the lateral structure of the tree, a set $\{\gamma\}$ is created for each combination of polyhedra. This set contains groupings of values β and γ for each separating axis search direction, where β is the position of the search direction in the block in $\bar{\boldsymbol{l}}$ & $\bar{\boldsymbol{u}}$ that represents this polyhedra combination. The value of γ, seen in line 13, is the signed distance of the lander polyhedron centroid, if the lander was at the objective position, from the edge of the lander polyhedron in the search direction, assuming that $\boldsymbol{\sigma}_{F/R} = \boldsymbol{0}$. The set $\{\gamma\}$ is then sorted in descending order of the value of γ in line 20. This step acts as a rough heuristic for the order in which the search direction constraint should be explored. Next, for the depth-wise structure of the tree, a value Γ is determined for each combination of polyhedra in line 11. This value is simply the distance from the surface polyhedron centroid to the objective landing position, and is added to the set $\{\Gamma\}$ along with the starting position of the block in $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$ that represents this polyhedra combination and the set $\{\gamma\}$. Then in line 21, the set $\{\Gamma\}$ is sorted in descending order of the value of Γ. The method for ordering the tree laterally is more likely to be accurate the further the surface polyhedron in question is from the final landing position. Therefore, assuming that a feasible landing position exists anywhere near the objective position, starting the tree at the surface polyhedron furthest away from the objective position leads to the shortest number of iterations until a feasible solution is found.

The branches of relaxed QP problems can be individually solved using any QP solver that operates on problems in the structure of Equations (19) and (20) with $Q \succeq 0$ and can be warm started. The primary motivating factor in choosing a solver for this problem is how well it performs with limited computing power. The solver chosen for the development of this paper is the OSQP[18] solver, which

Algorithm 2: Solver Part 1

Input: The surface and lander sets of polyhedra $\{P_k\}$ & $\{P_h\}$, $\boldsymbol{r}_{F/S,1}$ & $\boldsymbol{r}_{F/S,2}$ hessian weights x & y, objective position $\boldsymbol{r}^*_{F/S}$, position lower and upper bounds $\boldsymbol{r}_{F/S,\text{min}}$ & $\boldsymbol{r}_{F/S,\text{max}}$, big-M values D_i & D_{sr}, reference angle θ_r, tolerance τ, and max solver iterations Λ.

1 $H = \text{size}\left(\{P_h\}\right)$; $K_q = 0$
2 **foreach** $P \in \{P_k\}$ **do** $K_q = K_q + \text{size}\left(\{\rho\}\right)$
3 $n = 6 + HK_q$; $m = HK_q$
4 $Q = \mathbf{0}_{n\times n}$; $\boldsymbol{c} = \mathbf{0}_n$; $A = \mathbf{0}_{n\times n}$; $\bar{A} = \mathbf{0}_{m\times n}$; $\boldsymbol{l} = \mathbf{0}_n$; $\boldsymbol{u} = \mathbf{0}_n$; $\bar{\boldsymbol{l}} = -\tau\mathbf{1}_m$; $\bar{\boldsymbol{u}} = \tau\mathbf{1}_m$
5 $Q_{1,1}, Q_{2,2} = x, y$; $c_1 = -Q_{1,1} * r^*_{F/S,1}$; $c_2 = -Q_{2,2} * r^*_{F/S,2}$
6 $A_{1-6,1-6} = I_{6\times6}$; $\boldsymbol{l}_{1-3} = \boldsymbol{r}_{F/S,\text{min}}$; $\boldsymbol{u}_{1-3} = \boldsymbol{r}_{F/S,\text{max}}$; $\boldsymbol{l}_{4-6} = -\frac{\pi}{36}\mathbf{1}$; $\boldsymbol{u}_{4-6} = \frac{\pi}{36}\mathbf{1}$
7 $\boldsymbol{z}_0 = \mathbf{0}$; $z_{0,1} = r^*_{F/S,1}$; $z_{0,2} = r^*_{F/S,2}$
8 $i, j = 7$; $t = 1$; $\{\Gamma\} = \emptyset$
9 **foreach** $P_h \in \{P_h\}$ **do**
10 **foreach** $P_k \in \{P_k\}$ **do**
11 $\Gamma = ||\boldsymbol{r}^*_{F/S} - \boldsymbol{r}_{C/S,k}||$; $\{\gamma\} = \emptyset$; $\beta = 1$
12 **foreach** $(\hat{\boldsymbol{n}}, \rho) \in (\{N_k\}, \{\rho_k\})$ **do**
13 $\gamma = \hat{\boldsymbol{n}} \cdot \left(\boldsymbol{r}^*_{F/S} + [SR(\theta_r)]\boldsymbol{r}_{C/F,h} - \left(\boldsymbol{r}_{C/S,k} + \rho\hat{\boldsymbol{n}}\right)\right)$; $\{\gamma\} \leftarrow (\beta, \gamma)$
14 $\boldsymbol{r}_{V/C,h} = [\rho_{h,1}, \rho_{h,2}, \rho_{h,3}]^T$
15 $A_{i+\beta,1} = \hat{n}_1$; $A_{i+\beta,2} = \hat{n}_2$; $A_{i+\beta,3} = \hat{n}_3$
16 Using Equations (22-24)
17 $A_{i+\beta,j} = -D_i$; $\bar{A}_{t,j} = 1$; $u_{i+\beta} = D_{sr}$
18 $l_{i+\beta} = \hat{\boldsymbol{n}} \cdot \left(\boldsymbol{r}_{C/S,k} - \boldsymbol{r}_{C/F,h,\alpha} + \boldsymbol{r}_{V/C,h,\alpha}\right) + \rho - D_i$
19 $j = j + 1$; $t = t + 1$; $\beta = \beta + 1$
20 sort $\left(\{\gamma\}\right)$; $\{\Gamma\} \leftarrow (j - 6, \Gamma, \{\gamma\})$; $i = i + \beta$
21 sort $\left(\{\Gamma\}\right)$
22 Execute Algorithm 3 with Q, A, $\bar{A}$, $\boldsymbol{c}$, $\boldsymbol{l}$, $\boldsymbol{u}$, $\bar{\boldsymbol{l}}$, $\bar{\boldsymbol{u}}$, $\boldsymbol{z}_0$, $\{\Gamma\}$, τ, and Λ

meets all the requirements and is shown to be robust and efficient enough to operate on low-power embedded systems.[17]

Algorithm 3 depicts how the QP solver operates on the searching tree map to obtain a feasible solution to the problem. The lowest feasible cost value V and its associated solution $\boldsymbol{z}^*$ are initialized with ∞ and $\boldsymbol{z}_0$ respectively. After the QP solver is initialized, the set $\{T\}$ is created to contain all active branches of relaxed QPs (the current deepest branch and all previous branches that links it back to the start of the tree). Every $T \in \{T\}$ contains the solution of the previous branch $\boldsymbol{z}$ (initialized with $\boldsymbol{z}_0$), the current modification of the vectors $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$, and the depth μ and lateral ν position of the current branch. The relaxed QP of the most recently added branch is solved in lines 6-8, using the previous solution for warm starting. If the solution of the relaxed QP is both feasible and less than the lowest feasible cost value, the first lateral position of the next depth level in $\{\Gamma\}$ is accessed to create a new branch to be added to $\{T\}$ in lines 13-15. Otherwise, the current branch is removed from $\{T\}$, and is replaced with the next lateral position of its depth level in lines 22-25. If there are no more lateral positions in the current depth level, the current branch is not replaced and

the previous branch in $\{T\}$ is cycled through.

When the final depth level of the tree obtains an accepted solution, the values of V and $\boldsymbol{z}^*$ are updated in line 11 and then the tree searched for a better solution. This process continues until either the entire tree is searched, or some maximum number of iterations Λ is reached. At that point, the current values of V and $\boldsymbol{z}^*$ are returned as the solution of the MIQP.

Algorithm 3: Solver Part 2

Input: The problem matrices Q, A, $\bar{A}$ and vectors $\boldsymbol{c}$, $\boldsymbol{l}$, $\boldsymbol{u}$, $\bar{\boldsymbol{l}}$, $\bar{\boldsymbol{u}}$, initial solution $\boldsymbol{z}_0$, searching tree map $\{\Gamma\}$, tolerance τ, and max solver iterations Λ.

1 Initialize the QP solver with Q, $\boldsymbol{c}$, $A^* = \begin{bmatrix} A \\ \bar{A} \end{bmatrix}$, $\boldsymbol{l}^* = \begin{bmatrix} \boldsymbol{l} \\ \bar{\boldsymbol{l}} \end{bmatrix}$, $\boldsymbol{u}^* = \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}$
2 $\boldsymbol{z}, \boldsymbol{z}^* = \boldsymbol{z}_0$; $V = \infty$; $\mu = 1$; $\nu = 1$; $\lambda = 0$; $\{T\} = \emptyset$; $\{T\} \leftarrow (\boldsymbol{z}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \mu, \nu)$
3 **while** $\{T\} \neq \emptyset$ **do**
4 **if** $\lambda > \Lambda$ **then return** $V, \boldsymbol{z}^*$
5 $\lambda = \lambda + 1$
6 $(\boldsymbol{z}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \mu, \nu) \leftarrow$ last element of $\{T\}$
7 Update QP solver with $\boldsymbol{l}^* = \begin{bmatrix} \boldsymbol{l} \\ \bar{\boldsymbol{l}} \end{bmatrix}$, $\boldsymbol{u}^* = \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}$, and warm start $\boldsymbol{z}$
8 Run QP solver, set $\boldsymbol{z} =$ solution
9 **if** QP problem is feasible and $\frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + \boldsymbol{c}^T \boldsymbol{z} < V$ **then**
10 **if** $\mu =$ size $(\bar{\boldsymbol{l}})$ **then**
11 $V = \frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + \boldsymbol{c}^T \boldsymbol{z}$; $\boldsymbol{z}^* = \boldsymbol{z}$
12 **goto** line 17
13 $(j, \Gamma, \{\gamma\}) \leftarrow \mu$th element of $\{\Gamma\}$; $(\beta, \gamma) \leftarrow$ 1st element of $\{\gamma\}$
14 $\bar{l}_{j+\beta} = 1 - \tau$; $\bar{u}_{j+\beta} = 1 + \tau$; $\mu = \mu + 1$; $\nu = 1$
15 $\{T\} \leftarrow (\boldsymbol{z}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, V, \mu, \nu)$
16 **else**
17 $(\boldsymbol{z}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, V, \mu, \nu) \leftarrow$ last element of $\{T\}$; Remove last element of $\{T\}$
18 $(j, \Gamma, \{\gamma\}) \leftarrow \mu$th element of $\{\Gamma\}$
19 **if** $\nu =$ size $(\{\gamma\})$ **then**
20 **if** $\{T\} \neq \emptyset$ **then goto** line 17 **else return** $V, \boldsymbol{z}^*$
21 **else**
22 $(\beta, \gamma) \leftarrow \nu$th element of $\{\gamma\}$; $\bar{l}_{j+\beta} = -\tau$; $\bar{u}_{j+\beta} = \tau$
23 $\nu = \nu + 1$
24 $(\beta, \gamma) \leftarrow \nu$th element of $\{\gamma\}$; $\bar{l}_{j+\beta} = 1 - \tau$; $\bar{u}_{j+\beta} = 1 + \tau$
25 $\{T\} \leftarrow (\boldsymbol{z}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \mu, \nu)$

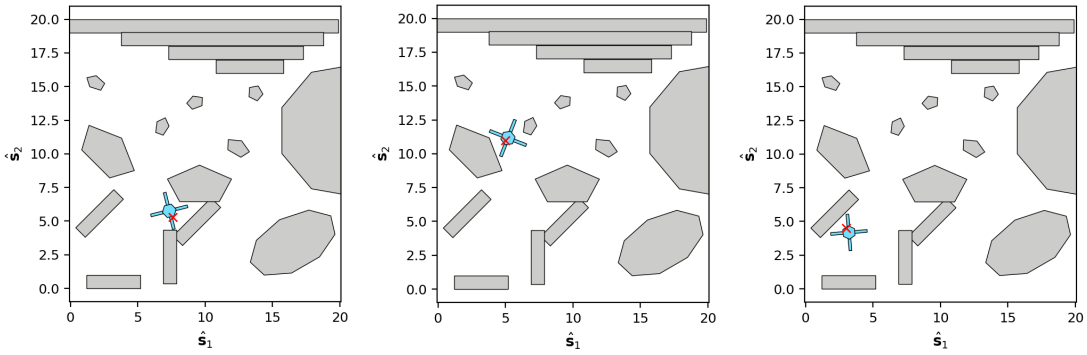
Note that the returned solution is not guaranteed to be optimal. The feasible region of this MIQP is non-convex, and will settle into a valley based on the construction of the searching tree. While brute force searching through the entire tree will find the optimal solution (if one exists), restrictions on computation time, controlled with Λ, may cause the algorithm to exit with a sub-optimal solution.

NUMERICAL RESULTS

The following results comes from two variations of the algorithm described in the previous section; one using the fine searching set, and the other using the coarse searching set. The algorithm has been implemented in C++ using the Eigen linear algebra library.[19] A Python 3.7 script is used to call and time the algorithm with the appropriate inputs, and to process the resulting data. Note that the algorithm implementation has not yet been optimized for solve times; the following solve times are presented for comparison between the fine and coarse searching methods. A MacBook Pro 2.8 GHz Intel Core i7 with 16 GB of RAM was used to collect this data.

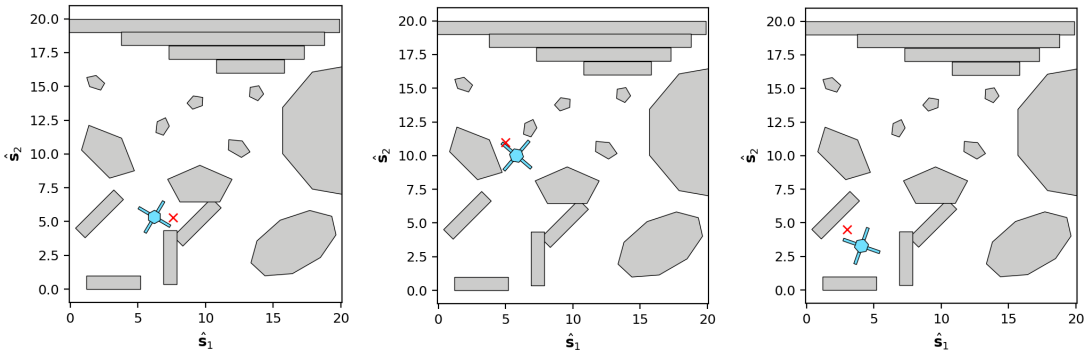
Convex polyhedra were used to hand craft 3D models for both the lander and the surface, and the dimensions of these models were left unitless for simplicity. The searchable surface area is constrained to 20×20 units in the $\hat{\boldsymbol{s}}_1$ and $\hat{\boldsymbol{s}}_2$ directions and contains 17 polyhedra, one of which extends beyond the 20 unit bounding box. The lander is comprised of 5 polyhedra that fit within a 2.5×2.5 unit box. The values chosen for D_{sr} and τ are 20 and 1.11×10^{-10}, respectively. Since one surface polyhedron exists partly outside the searchable area, the value of D_i is set to 30. The maximum number of iterations Λ is set dynamically to be one third the number of ϵ variables. Finally, the Python script, given an objective landing location, is set to run the solver 18 times over 20° intervals of θ_r for both searching sets.

Trials using this Python script were initially run with hand-picked objective landing locations, such to stress test the solver algorithm. Three notable trials are presented graphically in Figures 5 & 6 with some more analytic results in Table 1. The solve time marks the total duration of all 18 runs for each searching set, and the objective offset is the final distance from the center of the lander (point $\boldsymbol{F}$) to the objective landing location. The first notable result from this data is that the fine searching set solution is closer to the objective in all three trials. In particular, Trial 1 has a 178.8% improvement in the fine objective offset over the coarse. Next, Figure 6(c) clearly shows one of the potential downfalls of the coarse searching set, in which it effectively draws a boundary box around the surface polyhedron closest to the objective. This behavior of the coarse searching set is discussed previously in the Search Directions subsection. The last result to note from these trials is the solve time difference between the two searching sets. Both Trials 1 & 3 see the coarse searching set solve significantly faster than the fine. However, the fine search time in Trial 2 significantly out performs that of the coarse. This shows that the potential drawback of the fine searching set, there being more possible search directions to iterate over, does not solely influence the solve time.



(a) Trial 1 Fine Search (b) Trial 2 Fine Search (c) Trial 3 Fine Search

Figure 5. Solution to three objective locations (marked with an X) using the fine searching set.



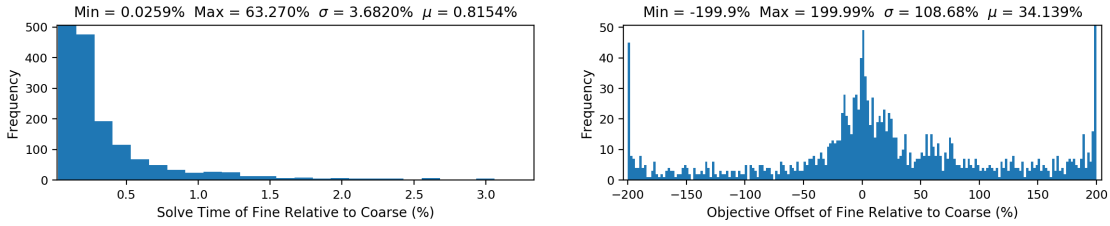
(a) Trial 1 Coarse Search (b) Trial 2 Coarse Search (c) Trial 3 Coarse Search

Figure 6. Solution to three objective locations (marked with an X) using the fine searching set.

Table 1. Numerical results of the three hand-picked trials.

	Trial 1 Fine	Trial 1 Coarse	Trial 2 Fine	Trial 2 Coarse	Trial 3 Fine	Trial 3 Coarse
Solve Time (s)	4.904723	3.916956	4.258778	5.862934	4.145936	2.723373
Objective Offset	0.059947	1.070691	0.218215	0.518254	0.136274	0.172912

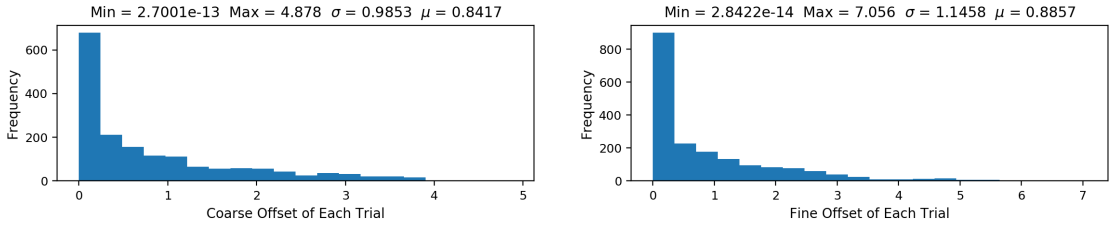
Several other hand-picked trials were run that are not presented here. Some notable results from these trials are that the coarse searching set occasionally returns a solution that is closer to the objective than the fine searching set, and that the solver has the possibility of failing to return a solution with the given Λ. These can occur when the objective landing location is within one of the surface polyhedra. Given unlimited time, the solver will always return a solution, if one exists, however the limit on solve time is crucial to HDA operations. To further quantify the performance of this solver with the given setting, a set of 2000 objective landing locations within the bounds of the surface area were randomly generated from a uniform distribution. This set was run through the Python script for the solver to produce solve time and objective offset data. Figure 7 provides a comparison of this data by plotting the signed percent difference of the fine searching set values relative to the coarse searching set values. Note that trials where one or both of the searching sets failed to provide a solution are not displayed here.



(a) Solve Time Performance (b) Objective Offset Performance

Figure 7. Percent increase of the fine searching set performance values over the coarse set.

This data provides some interesting insights; particularly in the difference of scale between the two performance measures. Starting with Figure 7(a) which indicates that, when both searching sets return a solution, the solve time using the coarse searching set is always less than that of the fine. The hand-picked trials clearly contradict this statement, however those previous results appear to be the exception, rather than the the norm. Also note that the increased solve time for the fine searching set is relatively small on average, with a mean of $\mu = 0.8154\%$, and is well clustered below a 5% increase. On the other hand, the objective offset data seen in Figure 7(b) varies widely, with a deviation of $\sigma = 108.68\%$ off the mean of $\mu = 34.139\%$. This behavior is unexpected since the theory suggests that the fine searching set is more flexible than the coarse, and thus should not perform significantly worse in finding a solution close to the objective. Further investigation is required to ascertain the cause of this behavior, however, this data still indicates that the fine searching set finds a better solution than the coarse on average. Furthermore, this data is put into perspective when looking at the actual objective offset values for both sets, seen in Figure 8. It is clear that solutions from both sets are, on average, placed relatively close to the objective landing location. Finally, of the 2000 random trials, 112 of them see both searching sets fail to find a solution, 173 see only the coarse set fail to find a solution, and two of them see only the fine set fail to find a solution. Therefore, using only the coarse searching set shows a 85.9% success rate, while using the fine set shows a 94.3% success rate.



(a) Coarse Searching Set

(b) Fine Searching Set

Figure 8. The objective offset values in each trial for both searching sets.

CONCLUSION & FUTURE WORK

In this paper, a new technique is presented as the foundation for a new Hazard Detection and Avoidance algorithm. The core idea of this technique is to decompose 3D models of surface hazards and the spacecraft lander into sets of convex polyhedra, and create an optimization problem with procedurally generated constraints that prevent the polyhedra from intersecting via the Separating Axis Theorem. This problem is cast into a Mixed Integer Quadratic Program that minimizes the distance between a feasible landing location and a desired landing location. Two possible variations of this MIQP are discussed; one which uses the basis directions of the surface frame as possible separating axes, and one that uses the face normal vectors of the surface polyhedra, named the coarse and fine searching sets, respectively. An algorithm that utilizes a Branch & Bound framework to solve these MIQPs is formulated, and has been tested in a simplified 2D scenario. While the tests show only a 94.3% success rate, this factor of the algorithm has a lot of room for improvement, specifically in how the B&B tree is searched. Once refined, the contents of this paper show promise to be further developed into a technique that can effectively determine a landing pose for a spacecraft with a geometrically conforming footprint.

The future development of this technique will involve the extraction of the required information from existing hazard identification techniques, and the utilization of trajectory knowledge and the surface environment when finding a feasible landing pose. Efficient convex decomposition techniques will be investigated and adapted for use on-board a spacecraft. Further constraints for the presented MIQP will be developed to ensure a stable solution is found with respect to local gravity. Such a constraint will enable this technique to be used with all six degrees of freedom, by preventing the solution from floating contactless above the surface proper. The use case of both the coarse and fine searching sets will be further investigated along with warm starting from previous solutions, such that this technique can be run several times during descent to provide refined guidance as the lander approaches the surface. Finally, the robustness and efficiency of this technique must be improved to the standards required for use in future small body missions.

NOTATION

$\boldsymbol{r}_{A/B}$	a vector from point $\boldsymbol{B}$ to point $\boldsymbol{A}$.
$\mathcal{N} : \{\boldsymbol{O}, \hat{\boldsymbol{n}}_1, \hat{\boldsymbol{n}}_2, \hat{\boldsymbol{n}}_3\}$	a frame definition with basis vectors $\hat{\boldsymbol{n}}$ and origin $\boldsymbol{O}$.
${}^{\mathcal{N}}\boldsymbol{r}$	a vector with components expressed in frame $\mathcal{N}$.
$\boldsymbol{\sigma}_{B/N}$	a set of Modified Rodriguez Parameters that represent the rotation from the $\mathcal{N}$ frame to the $\mathcal{B}$ frame.
$\mathrm{s}\theta$ & $\mathrm{c}\theta$	$\sin\theta$ & $\cos\theta$, respectively.

REFERENCES

[1] N. Serrano, "A Bayesian Framework for Landing Site Selection during Autonomous Spacecraft Descent," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, IEEE, Oct. 2006, pp. 5112–5117, 10.1109/IROS.2006.282603.

[2] M. Yu and H. Cui, "Robust hazard matching approach for visual navigation application in planetary landing," *Aerospace Science and Technology*, Vol. 47, Dec. 2015, pp. 378–387, 10.1016/j.ast.2015.09.028.

[3] S. Woicke and E. Mooij, "A stereo-vision hazard-detection algorithm to increase planetary lander autonomy," *Acta Astronautica*, Vol. 122, May 2016, pp. 42–62, 10.1016/j.actaastro.2016.01.018.

[4] P. Lunghi, M. Ciarambino, and M. Lavagna, "A multilayer perceptron hazard detector for vision-based autonomous planetary landing," *Advances in Space Research*, Vol. 58, July 2016, pp. 131–144, 10.1016/j.asr.2016.04.012.

[5] R. Wei, J. Jiang, X. Ruan, and J. Li, "Landing Area Selection Based on Closed Environment Avoidance from a Single Image During Optical Coarse Hazard Detection," *Earth, Moon, and Planets*, Vol. 121, July 2018, pp. 73–104, 10.1007/s11038-018-9516-2.

[6] P. Cui, D. Ge, and A. Gao, "Optimal landing site selection based on safety index during planetary descent," *Acta Astronautica*, Vol. 132, Mar. 2017, pp. 326–336, 10.1016/j.actaastro.2016.10.040.

[7] M. Perenzoni, D. Perenzoni, and D. Stoppa, "A 64 $\times$ 64-Pixels Digital Silicon Photomultiplier Direct TOF Sensor With 100-MPhotons/s/pixel Background Rejection and Imaging/Altimeter Mode With 0.14% Precision Up To 6 km for Spacecraft Navigation and Landing," *IEEE Journal of Solid-State Circuits*, Vol. 52, Jan. 2017, pp. 151–160, 10.1109/JSSC.2016.2623635.

[8] J. Seabrook, M. Daly, O. Barnouin, C. Johnson, A. Nair, E. Bierhaus, W. Boynton, R. Espiritu, R. Gaskell, E. Palmer, L. Nguyen, M. Nolan, and D. Lauretta, "Global shape modeling using the OSIRIS-REx scanning Laser Altimeter," *Planetary and Space Science*, Vol. 177, Nov. 2019, p. 104688, 10.1016/j.pss.2019.07.003.

[9] Y. Yan, D. Qi, C. Li, M. Yu, and T. Chen, "A Holistic Vision-based Hazard Detection Framework for Asteroid Landings," *IFAC-PapersOnLine*, Vol. 49, No. 17, 2016, pp. 218–223, 10.1016/j.ifacol.2016.09.038.

[10] S. Nakabeppu, Y. Ide, M. Takahashi, Y. Tsukahara, H. Suzuki, H. Shishido, and N. Yamasaki, "Space Responsive Multithreaded Processor (SRMTP) for Spacecraft Control," *2020 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, Kokubunji, Japan, IEEE, Apr. 2020, pp. 1–3, 10.1109/COOLCHIPS49199.2020.9097637.

[11] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004.

[12] M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien, "Fast approximate convex decomposition using relative concavity," *Computer-Aided Design*, Vol. 45, Feb. 2013, pp. 494–504, 10.1016/j.cad.2012.10.032.

[13] G. Fasano, *Solving Non-standard Packing Problems by Global Optimization and Heuristics*. SpringerBriefs in Optimization, Cham: Springer International Publishing, 2014, 10.1007/978-3-319-05005-8.

[14] H. Schaub and J. L. Junkins, *Analytical mechanics of space systems*. AIAA education series, Reston, VA: American Institute of Aeronautics and Astronautics, Inc, fourth edition ed., 2018.

[15] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*. Wiley-Interscience series in discrete mathematics and optimization, New York: Wiley, 1988.

[16] A. Bemporad and V. V. Naik, "A Numerically Robust Mixed-Integer Quadratic Programming Solver for Embedded Hybrid Model Predictive Control," *IFAC-PapersOnLine*, Vol. 51, No. 20, 2018, pp. 412–417, 10.1016/j.ifacol.2018.11.068.

[17] B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd, "Embedded Mixed-Integer Quadratic optimization Using the OSQP Solver," *2018 European Control Conference (ECC)*, Limassol, IEEE, June 2018, pp. 1536–1541, 10.23919/ECC.2018.8550136.

[18] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, Feb. 2020, 10.1007/s12532-020-00179-2.

[19] G. Guennebaud, B. Jacob, and others, *Eigen v3*. 2010.