Elliptic Edge Polygons for Observational Coverage Planning

Michael Trowbridge, Russell Knight, Elly Shao
Jet Propulsion Laboratory, California Institute of Technology
{michael.a.trowbridge,russell.l.knight,elly.j.shao}@jpl.caltech.edu

Abstract

This paper presents a representation of the polygonal footprint of an Earth-observing 2D framing sensor (i.e. instruments on Rosetta, Planet Labs SkySat) for observation coverage planning that preserves curvature of the footprint edge at little additional memory cost compared to previously published techniques.

Binary operations on field-of-view, ellipsoid intersection edges are introduced, allowing them to serve as edges in a polygon. A computational experiment examines the error produced by using this method versus existing methods of camera footprint representation. Edge approximation error is most significant when the field of view footprint is large compared to the body being observed (small body exploration, fly-bys, or other distant observer scenarios), and negligible when it is small (low altitude Earth observers with narrow fields of view). Great Circles polygons are degenerate elliptic edge polygons, admitting them to the polygon and edge operations in this paper.

Introduction

Planning a mapping campaign from space with a 2D framing instrument implies reasoning about which parts of the target body have been observed. Two approaches to this reasoning are the Footprint Placement for Mosaic Imaging optimization problem (Mitchell et al. 2018) and the Area Coverage Planning problem for 3-axis steerable, 2D framing sensors (Shao et al. 2018). As planning problems, both formalizations may be described as searches in a decision space for which observation footprint tuples $(\text{time}, \mathbf{r}_{\text{tgt}}, \theta)$[1] producing a set of instrument footprints $F = \{f_1, f_2, \ldots, f_n\}$ such that a target region of interest polygon P is contained in the union of all scheduled observation footprints. The goal state $g(n)$ is

$$g(n) = P \in \bigcup_{i=1}^{n} f_i \tag{1}$$

with a heuristic distance to the goal state $h(n)$ of

$$h(n) = \text{area}(P) - \text{area}\left(P - \bigcup_{i=1}^{n} f_i\right) \tag{2}$$

How the planner computes the footprint f_i is a lower level detail that can cause problems with evaluation of $g(n)$ and $h(n)$. (Shao et al. 2018) used 4-vertex great circles quadrilaterals to represent footprints f_i for computational efficiency. In the post-talk Q&A session, Valicka noted that these quadrilaterals did not resemble the curved footprints he had seen in his research, and rightly questioned the reasonableness of the quadrilateral approximation (which Shao et al. did not justify). As figure 1 shows, ignoring this curvature can cause the plan to achieve insufficient coverage.

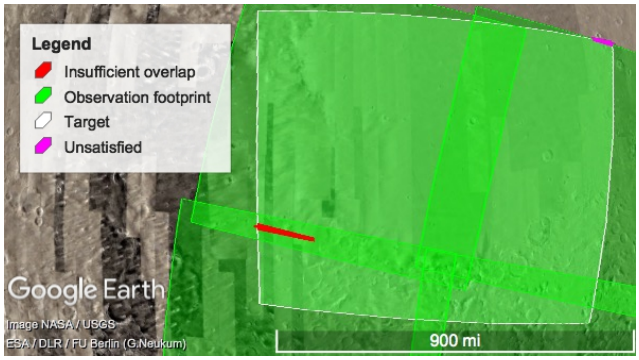

Figure 1: A mosaic mapping campaign has insufficient coverage because the distance to goal state heuristic ignored footprint curvature.

This paper is an excursion into the computational geometry that supports observational coverage planning with 2D framing instruments. First, we summarize existing ways to represent the edge of a footprint polygon f_i. An alternative data structure that preserves the incidence angle of the field of view edge is introduced. A computational experiment explores a practical concern of when it is appropriate to use a quadrilateral approximation of the footprint polygon for various algorithms of drawing the edge, and when the planner should preserve the footprint's curvature.

Contact author: Michael Trowbridge

[1](Mitchell et al. 2018) simplified the problem by fixing time and θ, focusing their approach on optimizing $\mathbf{r}_{\text{tgt}}$.

Background

Special Needs of Space-based Planners

To plan a mapping campaign of a planet, moon, or other small body, a planner should handle:

- Geometry on an arbitrary triaxial ellipsoid
- Polygons that contain the North or South pole
- Polygon edges that cross the antimeridian
- Edges defined by the shape of the instrument

While the planets of our solar system are modeled as spheres or spheroids, many of the smaller moons are modeled as non-spherical, triaxial ellipsoids (Archinal et al. 2011). Comet 67P Churyumov-Gerasimenko, which was later found to be very non-ellipsoidal (Jorda et al. 2016), was modelled as a triaxial ellipsoid with semi-axes 2.3, 1.9 and 1.7 km (Mysen 2004) before the Rosetta observation campaign. For small body exploration, an observational coverage planner shouldn't rely on spherical body assumptions.

The poles have been scientifically interesting areas of Mars (Bibring et al. 2004) and the Moon (Spudis et al. 2013). It's a safe assumption that future missions will study the poles of other celestial bodies. Studying the poles, or making any complete mapping campaign, requires handling observation polygons that cross the antimeridian ($-\pi = \pi$ longitude).

One framing instrument (camera) model is a rectangular pyramid of infinite height, with the detector at the peak. Each side of the camera's field of view is a plane. The intersection of that plane and the target triaxial ellipsoid is a straight line from the camera's perspective, but it is being projected obliquely onto a curved surface. This can be difficult to represent as two points and a connecting edge.

Existing Spherical Polygon Edge Types

In planar Euclidean geometry, the edge that connects two vertices of a polygon is a straight line. The closest analog in spherical geometry is the Great Circle: the intersection between a plane through the sphere's center and the surface of the sphere. Great-circles polygons can be treated much like polygons on a Euclidean plane, despite being on a non-Euclidean surface (the sphere). One implementation, the Computational Geometry Algorithm Library (CGAL), replaces 2D Euclidean lines with oriented planes to partition half-spaces and points with 3D vectors (Hachenberger and Kettner 2019).

Two other alternatives are rhumb and equirectangular lines[2], which are conceptually related. They are both curved paths across the sphere that are straight in a 2D map projection (Mercator for rhumb, plate carrée for equirectangular).

Rhumb lines are convenient for navigators because they follow a path of constant compass heading. Rhumb lines have the disadvantages of requiring special handling when the line is nearly a line of constant longitude (Bennett 1996) or following a logarithmic spiral into the poles (Alexander 2004).

[2] "Lat-Lon lines" in (Chamberlain and Duquette 2007).

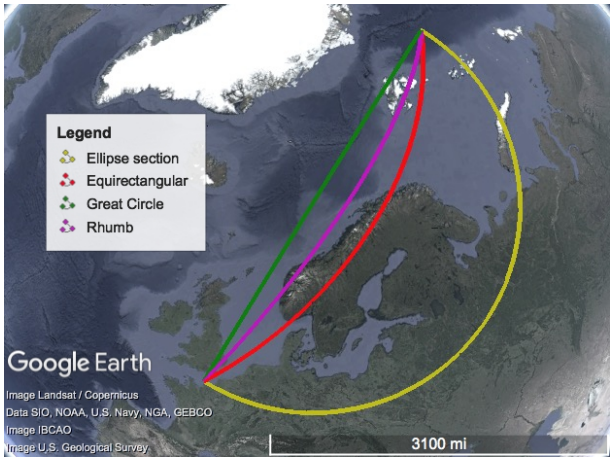

Figure 2: The Great Circle arc is straight when viewed from above a sphere

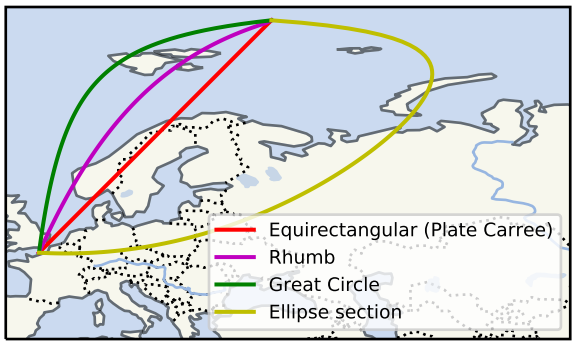

Figure 3: The equirectangular line is straight in a plate carrée projection

Equirectangular plate carrée lines are convenient for programmers because latitude and longitude map directly onto Cartesian coordinates (Wikipedia contributors 2019). This admits simpler or more powerful methods, such as the 2D planar packing algorithms that Mitchell et al. applied to the frame instrument footprint optimization problem (Mitchell et al. 2018). While this is a compelling trait, it adds projection distortion, which is worse at extreme latitudes. The poles, which are points in 3D, are horizontal lines in the 2D projection. Either the prime meridian or antimeridian ($0 = 2\pi$ or $-\pi{=}\pi$ radians longitude) requires special handling when a polygon crosses it.

None of these methods are appropriate for long distances. Figure 4, shows that they can all deviate significantly from the true shape (beige Elliptic).

The obvious workaround is to interpolate along the edge of the footprint. Instead of four corner points,

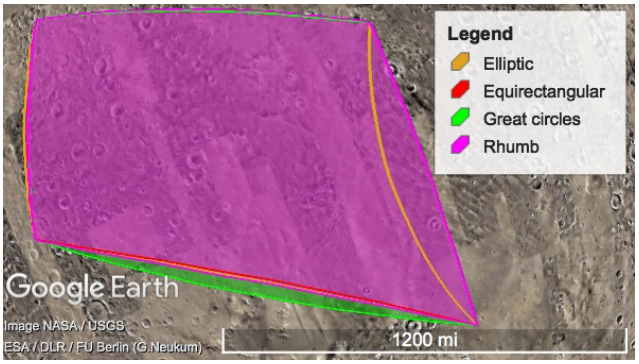

Figure 4: Large footprint with different edge types between 4 corners.

project look vectors at regular intervals along each edge of the observer's field of view. Consult a table[3] to choose a point spacing close enough to keep the approximation error within a reasonable bound. These extra points add space and computational complexity to subsequent polygon operations. There is a cheaper alternative, which we define in the next two sections.

Nomenclature

Unless otherwise specified ($_{/something}$), assume that the position vector is relative to the origin of the target body in target-fixed ($\mathcal{B}$-frame) coordinates.

ϵ	Working floating point precision of a math operation
$\hat{\mathbf{d}}$	The direction of some line
$\hat{\mathbf{n}}$	Surface normal of an ellipsoid-plane intersection
$\hat{\mathbf{r}}_{obs}$	Normalized (unit) vector of $\mathbf{r}_{obs}$
$\mathbf{b}$	An argument (query) point to one of the polygon/edge operations
$\mathbf{c}$	Position vector of a corner of the instrument's field of view
$\mathbf{c}_{1/obs}$	Corner 1's position relative to the observer
$\mathbf{d}_0$	Position vector of a known point on some line
$\mathbf{m}$	Position vector of an ellipse's center
$\mathbf{r}_{obs}$	Observer's position vector relative to the target body's origin
$\mathbf{x}$	A point on the surface of a target triaxial ellipsoid
$\mathcal{B}$	The target body-fixed coordinate frame
$\mathcal{C}$	The camera coordinate frame
$\mathcal{E}$	Edge containment check frame
$\mathcal{M}$	Ellipsoid-plane intersection coordinate frame
ϕ	A half-cone angle about some vector
θ	A clock angle within an ellipse
ε	Error
${}^{\mathcal{M}}\mathbf{r}$	Vector $\mathbf{r}$ expressed in $\mathcal{M}$-frame coordinates
${}^{\mathcal{M}}R_{\mathcal{B}}$	Coordinate transformation matrix from $\mathcal{B}$ to $\mathcal{M}$
A	x semi-axis of a 2D ellipse
B	y semi-axis of a 2D ellipse
e	A polygon edge
h_i	Point-to-plane distance of point i
P	A polygon on a triaxial ellipsoid
t	Some parametric scalar (general)

[3](Chamberlain and Duquette 2007) contains tables listing the maximum allowed edge length before error exceeds a critical bound.

Formulation

Handedness conventions

The operations in this paper take the approach that clockwise point ordering defines an enclosed region, with each arc's normal vector pointing inside. Reverse the order of the cross products if implementing in a framework that uses the opposite convention.

Intersection plane ellipse arcs as edges

The surface of a target triaxial ellipsoid is the set of points $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ such that

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{3}$$

The spacecraft's position vector relative to the target body is $\mathbf{r}_{obs}$, which is computed by an ephemeris library. If the sensor field of view is w radians wide and h radians high, the four corners of its field of view, in $\mathcal{C}$-frame (camera) coordinates and relative to the observer, are

$${}^{\mathcal{C}}\mathbf{c}_{i/obs} = \begin{bmatrix} 1 \\ \pm\tan(w/2) \\ \pm\tan(h/2) \end{bmatrix} \tag{4}$$

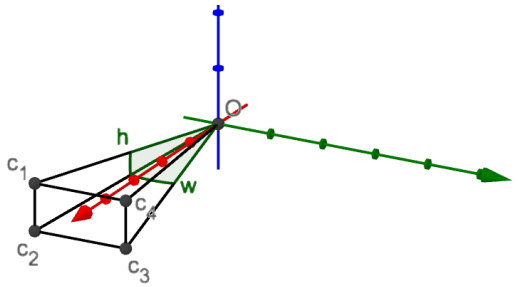

Figure 5: Camera field of view, $\mathcal{C}$-frame (red=x, green=y, blue=z) relative to the observer

The coordinate transformation from the $\mathcal{C}$ frame to the $\mathcal{B}$ frame is constructed by basis vectors as

$${}^{\mathcal{B}}R_{\mathcal{C}} = \begin{bmatrix} {}^{\mathcal{B}}\hat{\mathbf{u}} & {}^{\mathcal{B}}\hat{\mathbf{v}} & {}^{\mathcal{B}}\hat{\mathbf{w}} \end{bmatrix} \tag{5}$$

setting

$$\hat{\mathbf{u}} = \frac{\mathbf{r}_{tgt} - \mathbf{r}_{obs}}{|\mathbf{r}_{tgt} - \mathbf{r}_{obs}|} \tag{6}$$

choosing the secondary alignment axis to achieve a desired target footprint alignment and using a cross product to compute the third.

If the four corners of the footprint intersect the target body, the intersection point will satisfy both equation 3 and

$$\mathbf{c}_i = \mathbf{r}_{obs} + t\mathbf{c}_{i/obs} \tag{7}$$

for some value of scalar t. We use the CSPICE function surfpt_c() for this calculation (Acton 1996).

The observer's position and the surface intercepts of two consecutive imager field of view corner points $\mathbf{c}_1$, $\mathbf{c}_2$ form a plane that defines a field of view edge. This plane's surface normal $\hat{\mathbf{n}}$ may be computed as

$$\mathbf{c}_{1/obs} = \mathbf{c}_1 - \mathbf{r}_{obs} \tag{8}$$

$$\mathbf{c}_{2/obs} = \mathbf{c}_2 - \mathbf{r}_{obs} \tag{9}$$

$$\hat{\mathbf{n}} = \frac{\mathbf{c}_{1/obs} \times \mathbf{c}_{2/obs}}{|\mathbf{c}_{1/obs} \times \mathbf{c}_{2/obs}|} \tag{10}$$

The position vector $\mathbf{x} = [\ x \quad y \quad z\ ]^T$ of each point on the edge e between $\mathbf{c}_1$ and $\mathbf{c}_2$ satisfies the target's ellipsoid surface equation 3 and is entirely in the plane defined by $\mathbf{c}_1$ and $\hat{\mathbf{n}}$:

$$(\mathbf{x} - \mathbf{c}_1) \cdot \hat{\mathbf{n}} = 0 \tag{11}$$

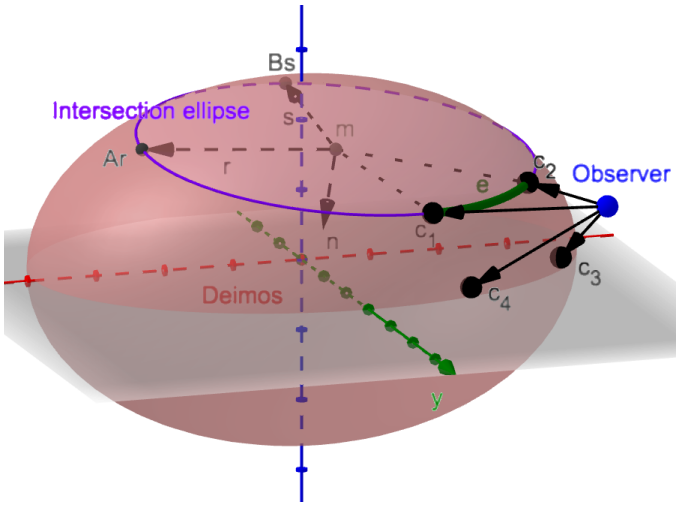

Figure 6: 3D view of the observer, target and FOV edge

The intersection of this plane and the ellipsoid is an ellipse with a solution of the form (Klein 2012)[4]

$$\mathbf{x} = \mathbf{m} + A\cos\theta\hat{\mathbf{r}} + B\sin\theta\hat{\mathbf{s}} \tag{12}$$

[4]Klein's paper has instructions for computing $\mathbf{m}, A, B, \hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$. Van Wal and Scheeres present an abridged version (Van wal and Scheeres 2016).

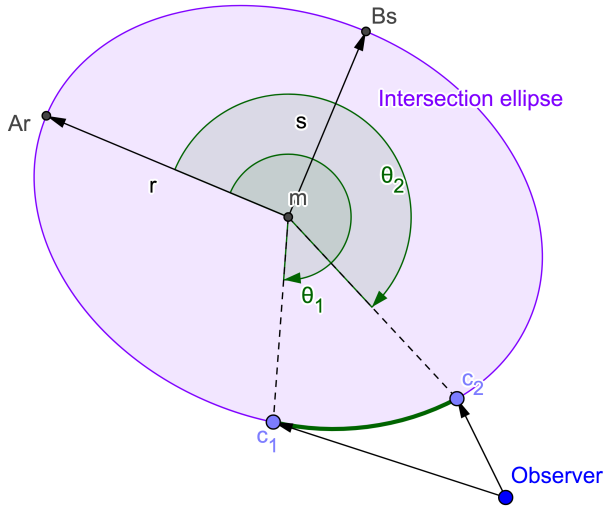

Figure 7: 2D view of the FOV intersection ellipse plane

The polygon edge e covers the domain $\theta_1 \leq \theta \leq \theta_2$, where θ_1 corresponds to $\mathbf{c}_1$ and θ_2 corresponds to $\mathbf{c}_2$. We can compute θ_1 and θ_2 by defining a new coordinate frame $\mathcal{M}$ centered at $\mathbf{m}$ and basis vectors $\{\ \hat{\mathbf{r}} \quad \hat{\mathbf{s}} \quad \hat{\mathbf{n}}\ \}$. The coordinate transformation from target body-fixed frame $\mathcal{B}$ to $\mathcal{M}$ is

$${}^{\mathcal{M}}R_{\mathcal{B}} = \begin{bmatrix} \hat{\mathbf{r}}^T \\ \hat{\mathbf{s}}^T \\ \hat{\mathbf{n}}^T \end{bmatrix} \tag{13}$$

To transform some arbitrary vector $\mathbf{b}$ to $\mathcal{M}$-frame coordinates relative to the ellipse center,

$${}^{\mathcal{M}}\mathbf{b}_{/m} = \left[{}^{\mathcal{M}}R_{\mathcal{B}}\right](\mathbf{b} - \mathbf{m}) \tag{14}$$

The arc edge e may be represented minimally as $e = (\mathbf{c}_1, \mathbf{c}_2, \hat{\mathbf{n}})$.

Operations on intersection ellipse arcs

This section defines some primitive operations on ellipse arc edges that are required by many polygon operation algorithms. They are listed constructively, building on the prior operations.

Operation 1 (Is point $\mathbf{b} \in e$?). *Is some point* $\mathbf{b} = [\ b_x \quad b_y \quad b_z\ ]^T$ *on the arc* $e = (\mathbf{m}, \hat{\mathbf{n}}, \mathbf{c}_1, \mathbf{c}_2)$*?*

If $\mathbf{b} \in e$, the relative position vector

$$\mathbf{b}_{/m} = \mathbf{b} - \mathbf{m} \tag{15}$$

must be within the intersection plane and in a sector bounded by lines $(\mathbf{m}, \mathbf{c}_1)$ and $(\mathbf{m}, \mathbf{c}_2)$[5]. The in-plane condition can be checked with the dot product

$$\left|\mathbf{b}_{/m} \cdot \hat{\mathbf{n}}\right| < \epsilon \tag{16}$$

[5]If $\mathbf{b}$ is actually on the surface (not inside or outside of the target ellipsoid), $\mathbf{b}$ must also satisfy the triaxial ellipsoid surface equation. This check is deliberately omitted so that the operation handles sub-surface and irregular terrain points.

The sector bounds may be checked by computing the dot product of $\mathbf{b}_{/m}$ with two line normal vectors: normal to $(\mathbf{m}, \mathbf{c}_1)$ and in the $\mathbf{c}_2$ direction ($\mathbf{c}_{1\perp}$) and its converse, normal to $(\mathbf{m}, \mathbf{c}_2)$ and in the $\mathbf{c}_1$ direction ($\mathbf{c}_{2\perp}$).

$$\mathbf{c}_{1/2} = \mathbf{c}_1 - \mathbf{c}_2 \tag{17}$$

$$\mathbf{c}_{1\perp} = \mathbf{c}_{1/2} - \mathbf{c}_{1/2} \cdot \mathbf{c}_1 \mathbf{c}_1 \tag{18}$$

$$\mathbf{c}_{2\perp} = -\mathbf{c}_{1/2} + \mathbf{c}_{1/2} \cdot \mathbf{c}_2 \mathbf{c}_2 \tag{19}$$

The vectors, $\hat{\mathbf{n}}, \mathbf{c}_{1\perp}$ and $\mathbf{c}_{2\perp}$ may be computed once and memoized as a non-orthogonal frame basis $\mathcal{E}$, where the coordinate transformation from $\mathcal{B}$ to $\mathcal{E}$ is

$${}^{\mathcal{E}}R_{\mathcal{B}} = \begin{bmatrix} \mathbf{c}_{1\perp} & \mathbf{c}_{2\perp} & \hat{\mathbf{n}} \end{bmatrix}^T \tag{20}$$

Transform $\mathbf{b}_{/m}$ to $\mathcal{E}$-frame coordinates:

$${}^{\mathcal{E}}\mathbf{b}_{/m} = \begin{bmatrix} b_{1/m} \\ b_{2/m} \\ b_{3/m} \end{bmatrix}_{\mathcal{E}} = \left[{}^{\mathcal{E}}R_{\mathcal{B}}\right] \left[{}^{\mathcal{B}}\mathbf{b}_{/m}\right] \tag{21}$$

Point $\mathbf{b} \in e$ iff, in $\mathcal{E}$-frame coordinates,

$$\left(b_{1/m} \geq 0\right) \wedge \left(b_{2/m} \geq 0\right) \wedge \left(\left|b_{3/m}\right| < \epsilon\right) \tag{22}$$

Operation 2 (Are arcs e_1, e_2 coplanar?)**.**

Defining each arc as

$$e_1 = (\mathbf{m}_1, \hat{\mathbf{n}}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \tag{23}$$

$$e_2 = (\mathbf{m}_2, \hat{\mathbf{n}}_2, \mathbf{c}_{21}, \mathbf{c}_{22}) \tag{24}$$

Arc e_1 is coplanar with e_2 if

$$|(\mathbf{m}_2 - \mathbf{m}_1) \cdot \hat{\mathbf{n}}_1| < \epsilon \tag{25}$$

and

$$|1 - |\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2|| < \epsilon \tag{26}$$

Operation 3 (Find intersection of arcs e_1, e_2)**.** *Where, if anywhere, does arc e_1 intersect arc e_2? Possible outputs: $\emptyset, \mathbf{p}_1$ or $(\mathbf{p}_1, \mathbf{p}_2)$*

If e_1 intersects e_2, conditions 3.1, 3.2 and 3.3 must be satisfied. Checking the conditions in that order reduces corner cases later.

Condition 3.1 (Planes are not parallel)**.** *Equation 26 must be false.*

If the two planes are not parallel, then the planes will intersect at a line of the form

$$\mathbf{d}_0 + t\hat{\mathbf{d}} \tag{27}$$

This line will be orthogonal to both plane normal vectors, so its direction $\hat{\mathbf{d}}$ can be computed from their cross product (McKinnon 2012):

$$\hat{\mathbf{d}} = \frac{\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2}{|\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2|} \tag{28}$$

There are an infinite number of correct $\mathbf{d}_0$, so choose a convenient form:

$$\mathbf{d}_0 = \mathbf{m}_1 + c\hat{\mathbf{r}}_1 \tag{29}$$

Because $\mathbf{d}_0$ must also be in e_2's plane,

$$(\mathbf{m}_1 + c\hat{\mathbf{r}}_1 - \mathbf{m}_2) \cdot \hat{\mathbf{n}}_2 = 0 \tag{30}$$

yielding

$$c = \frac{(\mathbf{m}_2 - \mathbf{m}_1) \cdot \hat{\mathbf{n}}_2}{\hat{\mathbf{r}}_1 \cdot \hat{\mathbf{n}}_2} \tag{31}$$

Equation 31 contains a singularity when $\hat{\mathbf{r}}_1 \cdot \hat{\mathbf{n}}_2 = 0$. If this occurs, substitute $\hat{\mathbf{s}}_1$ for $\hat{\mathbf{r}}_1$ in equations 29 to 31.

Condition 3.2 (Intersection line crosses ellipses)**.** *The line of planar intersection $\mathbf{d}_0 + t\hat{\mathbf{d}}$ must intersect both ellipses 1 and 2.*

This condition is easier to evaluate in the $\mathcal{M}$-frame of ellipse 1, relative to its center $\mathbf{m}_1$. Transform equation 27 as

$${}^{\mathcal{M}}\left(\mathbf{d}_0 + t\hat{\mathbf{d}}\right) = \left[{}^{\mathcal{M}}R_{\mathcal{B}}\right]\left(\mathbf{d}_0 + t\hat{\mathbf{d}} - \mathbf{m}_1\right)$$

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{cases} \begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} \delta_1 \\ \delta_2 \\ 0 \end{bmatrix} & \text{if } \hat{\mathbf{r}}_1 \cdot \hat{\mathbf{n}}_2 \neq 0 \\ \begin{bmatrix} 0 \\ c \\ 0 \end{bmatrix} + t \begin{bmatrix} \delta_1 \\ \delta_2 \\ 0 \end{bmatrix} & \text{otherwise} \end{cases}$$

The first case corresponds to equations 29 to 31 as written. The second case applies when $\hat{\mathbf{s}}_1$ must be used to avoid equation 31's singularity. Removing t, the first case reduces to

$$x = c + \frac{y}{\delta_2}\delta_1 \rightarrow y = \frac{\delta_2}{\delta_1}x - \frac{\delta_2}{\delta_1}c \tag{32}$$

or, if $\hat{\mathbf{s}}_1$ were used instead of $\hat{\mathbf{r}}_1$,

$$y = \frac{\delta_2}{\delta_1}x + c \tag{33}$$

Equations 32 and 33 have the form $y = mx + b$. If there is an intersection between this line and the ellipse, it will also satisfy the ellipse equation $\frac{x^2}{A^2} + \frac{y^2}{B^2} = 1$ for some real-valued x and y. Substituting for y,

$$\frac{x^2}{A^2} + \frac{(mx + b)^2}{B^2} = 1 \tag{34}$$

arranging in quadratic form,

$$\left(A^2m^2 + B^2\right)x^2 + 2A^2mbx + A^2b^2 - A^2B^2 = 0 \tag{35}$$

defining

$$\alpha = A^2m^2 + B^2 \tag{36}$$

$$\beta = 2A^2mb \tag{37}$$

$$\kappa = A^2b^2 - A^2B^2 \tag{38}$$

then applying the quadratic formula,

$$x = \frac{-\beta \pm \sqrt{\beta^2 - 4\alpha\kappa}}{2\alpha} \tag{39}$$

If $\beta^2 > 4\alpha\kappa$, then there are two real valued solutions (two intersections). If $\beta^2 = 4\alpha\kappa$, then there is a single intersection (a tangent point), and if $\beta^2 < 4\alpha\kappa$, then

the plane intersection line does not intersect the ellipse. Evaluate equation 39 to obtain x_i, then either equation 32 or 33 (as appropriate) to obtain y_i in $\mathcal{M}$ frame coordinates. Reverse the transformation in equation 14 to construct the intersection point position vectors in body-fixed $\mathcal{B}$-frame coordinates:

$$\mathbf{p}_i = \left[{}^{\mathcal{M}}R_{\mathcal{B}}\right]^T \begin{bmatrix} {}^{\mathcal{M}}x_i \\ {}^{\mathcal{M}}y_i \\ 0 \end{bmatrix} + \mathbf{m}_1 \tag{40}$$

Condition 3.3 (Ellipse-line intersections within the arc)**.** *One or two of the line-ellipse intersections must be within the each ellipse's angle bounds θ_{i1} and θ_{i2}.*

The points $\mathbf{p}_i$ are intersections of the two ellipses. We must now check that they fall within the bounds of the arcs, which are subsets of the ellipses. Invoke operation 1 for $\mathbf{p}_1$ and $\mathbf{p_2}$ against arcs e_1 and e_2. Point $\mathbf{p}_i$ is an arc intersection if

$$\mathbf{p}_i \in e_1 \wedge \mathbf{p}_i \in e_2 \tag{41}$$

Operation 4 (Split an arc)**.** *Given a point $\mathbf{b} \in e = (\hat{\mathbf{n}}, \mathbf{c}_1, \mathbf{c}_2), \mathbf{b} \neq \mathbf{c}_1 \wedge \mathbf{b} \neq \mathbf{c}_2$, subdivide e into arcs e_1 and e_2 at $\mathbf{b}$.*

Trivial:

$$e_1 = (\hat{\mathbf{n}}, \mathbf{c}_1, \mathbf{b}) \tag{42}$$

$$e_2 = (\hat{\mathbf{n}}, \mathbf{b}, \mathbf{c}_2) \tag{43}$$

Operation 5 (Furthest point from cone center)**.** *Find the point $\mathbf{x}_{max} \in e$ that is the furthest from the center of a unit vector $\hat{\mathbf{b}}$ drawn from the origin, where*

$$d = 1 - \cos\phi = 1 - \frac{\mathbf{x} \cdot \hat{\mathbf{b}}}{|\mathbf{x}|} \tag{44}$$

Equation 44 is continuous and differentiable because $\mathbf{x}(\theta)$ is a linear sum of constants and continuous, differentiable functions. Its derivative with respect to θ is

$$\frac{\partial d}{\partial \theta} = \frac{\partial \mathbf{x}}{\partial \theta} \cdot \left(\frac{\mathbf{x} \cdot \hat{\mathbf{b}}}{|\mathbf{x}|^3} - \frac{\hat{\mathbf{b}}}{|\mathbf{x}|} \right) \tag{45}$$

and

$$\frac{\partial \mathbf{x}}{\partial \theta} = -A \sin\theta \hat{\mathbf{r}} + B \cos\theta \hat{\mathbf{s}} \tag{46}$$

Use a second derivative concavity check to determine whether e is the one local maximum or one local minimum case. If $d(\theta)$ is concave up on (θ_1, θ_2), then the local max of d will be either θ_1 or θ_2. If $d(\theta)$ is concave down, gradient ascent or bisection may be used to locate the maximum.

After locating

$$\theta_{max} = \text{argmax } d, \theta \in (\theta_1, \theta_2) \tag{47}$$

the most extreme point on the arc $\mathbf{x}_{max}$ is obtained from evaluating equation 12 for $\theta = \theta_{max}$.

Operation 6 (Compute polygon bounding cone)**.** *Construct a bounding cone $(\hat{\mathbf{b}}, d_{max})$ enclosing polygon P, where $\hat{\mathbf{b}}$ is a unit vector drawn from the center of the triaxial ellipsoid and d corresponds to a half-cone angle ϕ about $\hat{\mathbf{b}}$.*

Start by using the optimal bounding cone algorithm for vectors in three dimensions (Barequet and Elber 2005), where each 3D vector to be bounded is a vertex of polygon P. Because the edges of P are not great circles arcs, we must expand the bounding cone to enclose the most extreme point on each arc:

$$d_{max} \leftarrow \max\left(d_{max}, \max\left(d(\theta_{max}(e_i)), e_i \in P\right)\right) \tag{48}$$

The expansion in equation 48 doesn't consider alternate pointings for $\hat{\mathbf{b}}$, causing $(\hat{\mathbf{b}}, d_{max})$ to lose its guarantee of optimality. The cone retains the trait of completely enclosing P, however, which is sufficient for our purposes.

Operation 7 (Point-in-convex-polygon check)**.** *Is some point $\mathbf{b}$ inside, outside, or on the boundary of convex polygon $P = \{(c_1, c_2, \ldots, c_n), (e_1, e_2, \ldots, e_n)\}$?*

Because our bounding planes are not required to pass through the ellipsoid's center, the volume that they enclose may intersect the ellipsoid on both sides. We use a bounding cone to to separate the polygon of interest P from its antipodal polygon P'.

Condition 7.1 (Bounding cone containment)**.** *If point $\mathbf{b}$ is contained by polygon P, $\mathbf{b}$ must also be contained by the bounding cone $(\hat{\mathbf{b}}, d_{max})$ enclosing P.*

Define $\mathbf{x}_b$ as the position vector corresponding to point $\mathbf{b}$. Compute the distance d_b between $\mathbf{x}_b$ and $\hat{\mathbf{b}}$ using equation 44. Point $\mathbf{b}$ is contained by the bounding cone iff $d_b \leq d_{max}$.

Condition 7.2 (Convex bounding plane containment)**.** *To be contained in P, point $\mathbf{b}$ must also be on the inside of each edge's intersection ellipse plane.*

Evaluate the signed point to plane distance h_i for each edge e_i:

$$h_i = (\mathbf{b} - \mathbf{c}_i) \cdot \hat{\mathbf{n}}_i \tag{49}$$

If any $h_i < -\epsilon$, $\mathbf{b}$ is outside. If all $h_i > \epsilon$, $\mathbf{b}$ is inside. Otherwise, point $\mathbf{b}$ is on the boundary of P.

Theorem 1 (Interoperability with Great Circles)**.** *Polygons may be composed of any mix of Great Circles and ellipsoid-plane intersection arcs.*

Lemma 1.1 (Great Circle degeneracy)**.** *Great Circle arcs are degenerate ellipsoid-plane intersection arcs.*

Set $\mathbf{m} = \vec{0}, \hat{\mathbf{r}} = \mathbf{c_1}$ and $\hat{\mathbf{n}} = (\mathbf{c}_2 \times \mathbf{c}_1) / |\mathbf{c}_2 \times \mathbf{c}_1|$. The center of the intersection ellipse will be at the origin (center of the target body). All intersections of a plane and a sphere are a circle, which makes this particular elliptic edge also a Great Circle.

Corollary 1.1. *Great Circle arcs may be converted to ellipsoid-plane intersection arcs.*

The Great Circle arc $(\mathbf{c}_1, \mathbf{c}_2)$ may be augmented with $\hat{\mathbf{n}} = (\mathbf{c}_2 \times \mathbf{c}_1) / |\mathbf{c}_2 \times \mathbf{c}_1|$ to completely define $e = (\mathbf{c}_1, \mathbf{c}_2, \hat{\mathbf{n}})$.

Methodology

We hypothesize that the choice of polygon line is important for large, oblique footprints of distant observers, but not important for small footprints of closer observers. A computational experiment will test this hypothesis for two mission cases.

The first mission case (near/narrow) is modeled on the Earth-observing Planet Labs (formerly Skybox, Terra Bella) SkySat-1 in a Low Earth Orbit (LEO). Its field of view is derived from nominal scene size at reference orbit altitude in the Planet Image Product Specifications (Planet Labs, Inc. 2018). The nadir altitude is based on a propagation of the SkySat-1 ephemeris from the STK Data Federate (Analytic Graphics, Inc. 2019).

The second case examines a smaller, more distant body, with a larger observer field of view. The Mars Reconnaissance Orbiter (MRO) aerobraking phase (Semenov and You 2006) is used with the 6° MRO context camera (Malin Space Systems, Inc. 2005) and a nadir footprint that is 17% of the Mars ellipsoid's smallest semiaxis. This is similar to the closest approach in the Rosetta OSIRIS mapping campaign (Jorda et al. 2016), where a 9 km altitude image with the 2.2° square field of view NAC has a footprint approximately 16% comet's smallest approximating ellipsoid[6] semiaxis.

Table 1: Observer footprint configurations

	Near/Narrow	**Far/Wide**
Spacecraft	SkySat-1	MRO
Body observed	Earth	Mars
Trajectory	LEO	Aerobraking
Nadir Altitude	578 km	8992 km
Field of view	$0.37° \times 0.15°$	$6° \times 6°$

Rhumb lines, equirectangular lines and great circles arcs as approximations of the true shape (an ellipsoid-plane intersection arc). Error of this approximation is computed as a fractional disagreement ε between the truth polygon P_t and the approximation (rhumb/equirectangular/great circle) polygon P_a:

$$\varepsilon = \frac{\text{area}\,(P_a \cup P_t) - \text{area}\,(P_a \cap P_t)}{\text{area}\,(P_t)} \qquad (50)$$

Large ε is a poor approximation and $\varepsilon = 0$ is a perfect approximation.

Polygon area is computed using the great circles polygon algorithm in (Chamberlain and Duquette 2007). All footprint polygons will be interpolated with 100 points per side according to the edge type under test and stored as great circles polygons. Polygon intersections and unions will use the Margalit and Knott algorithm (Margalit and Knott 1989), adapted for great circle arcs.

[6](Jorda et al. 2016) provides an approximating triaxial ellipsoid, but notes that 67B's structure is more properly modeled as bilobate.

Results

Impact of the footprint polygon line type

Figure 8 shows that for a low altitude observer with a small footprint, the approximation error of the footprint is small (less than 1%). In general, error increases as off-nadir angle increases.

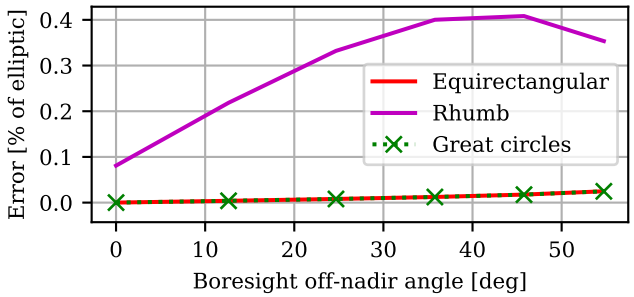

Figure 8: Error caused by non-elliptic edges for a low altitude observer with a small footprint (SkySat-1)

When the observer was further away and had a larger field of view, the error grew to almost 10% (figure 9).

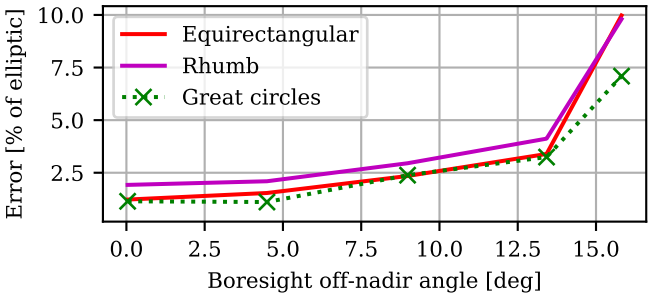

Figure 9: Error caused by non-elliptic edges for a larger footprint further from the target body (MRO/CTX)

Discussion

Does the type of edge line matter?

Sometimes. For commercial imagery operators in Low Earth Orbit (close relative to the target body's radius) with small fields of view, choice of edge type doesn't matter. All of the edge types in the Earth LEO mission case had less than 1% approximation error, even up to 55° off-nadir angle.

When the observer is far from the target or has a large field of view relative to the target, yes, the choice of line can matter. The edge type was more significant as off nadir angle increased. Even at 10% error, however,

a planner could compensate by margining the field of view size or using more interpolation points.

Round trip error to/from ellipse angle

While testing against Earth, we found that repeatedly transforming a point between position vector and ellipse-frame clock angle θ via equation 12 had round trip error ranging from 0.01 km (single trip, nadir) to 44 km (10 round trips, 35° off-nadir). The error generally increased with off-nadir angle and accumulated linearly with each invocation. We did not identify the root cause, but did find a workaround.

Earth, which is near spherical, is an edge case in Klein's algorithm where $D_1\hat{\mathbf{r}} \approx D_1\hat{\mathbf{s}}$. Klein recommends using $\omega = \pi/4$ in this case (Klein 2012). $\omega = \pi/4$ removed the compounding effect from the round trip error, but did not correct the first round trip error.

The best workaround we found was to first define the unit vector $\hat{\mathbf{b}}$ pointing in the direction of θ, expressed in $\mathcal{M}$-frame coordinates, as

$$^{\mathcal{M}}\hat{\mathbf{b}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \end{bmatrix}^T \tag{51}$$

then transform it to body-fixed $\mathcal{B}$-frame coordinates

$$^{\mathcal{B}}\hat{\mathbf{b}} = \begin{bmatrix} \hat{\mathbf{r}} & \hat{\mathbf{s}} & \hat{\mathbf{n}} \end{bmatrix} \left[^{\mathcal{M}}\hat{\mathbf{b}} \right] \tag{52}$$

Projected to the surface of the triaxial ellipsoid, the output point $\mathbf{b}$ can be written as

$$\mathbf{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix} = \mathbf{m} + t\hat{\mathbf{b}} \tag{53}$$

where only t is unknown and

$$\frac{b_x^2}{a^2} + \frac{b_y^2}{b^2} + \frac{b_z^2}{c^2} = 1 \tag{54}$$

Evaluating these equations with CSPICE surfpt_c() had 10^{-12} km round trip error (machine precision).

Generalizations and Applicability

No spherical approximations are used in the formulation of the elliptic edge polygon. Without modification, these polygons may be used for any triaxial ellipsoid where the field of view is small enough that it does not span more than 180° of the ellipsoid (defined by a cutting plane through the ellipsoid's origin).

Elliptic edge polygons treat the edge of the instrument footprint as a plane. Pushbroom sensor swath sides are solids of revolution formed by the sensor center line. Elliptic edge polygons are applicable to the ASPEN Eagle Eye scheduler (Knight, Donnellan, and Green 2013), but not the Compressed Large Activity Scheduler-Planner (CLASP) (Knight and Chien 2006) or AEOS strip selection problems (Lemaître et al. 2002).

Related work

Shao et al. showed that footprint size and skew change during an overflight, but used 4-corner Great Circles arc polygons for the footprints, losing the curvature of footprint edges (Shao et al. 2018). GeoPlace used a square camera footprint with more realistic curvature at the edges (Mitchell et al. 2018). The publicly released GeoPlace source code appears to use an oblique intersection of the square field of view and a sphere, treating the footprint as a 2D planar polygon thereafter. Edge curvature was preserved by adding intermediate vertices[7]. This paper presents a method of achieving footprint edge fidelity similar to (Mitchell et al. 2018), but at a lower space complexity and on a triaxial ellipsoid.

The Computational Algorithms Geometry Library (CGAL) implements Nef polygons on a sphere, where edges are segments of the intersection of a sphere and plane through the origin (Hachenberger and Kettner 2019). This is sufficient for great circles polygons, but has no explicit provision for triaxial ellipsoids or halfspaces from planes that do not intersect the origin.

Recommendations for future work

The equation for the area of an arbitrary polygon on a sphere is supported by a proof that approximates a polyhedron with a sphere. It is not clear that this proof also covers polygons on ellipsoids whose arc edges do not pass through the body's center. Handedness checks and winding number point-in-polygon algorithms should be similarly scrutinized.

This paper is missing a critical operation: point-in-concave-polygon containment checks, required for polygon intersection, union and subtraction (Margalit and Knott 1989). Our next step would have been to decompose the concave polygon into convex polygons using the method outlined in (Li, Wang, and Wu 2007).

Conclusion

By storing an observer's camera footprint polygon as an elliptic edge polygon, the planner can preserve the curvature of the footprint without adding intermediate points along the edge. The experiment showed that curvature is only a concern when the observer's field of view is large compared to the observed body, more so when the observer obliquely views the target body. The elliptic edge polygon algorithms in this paper can be used more broadly to handle edge cases in existing Great Circles polygon/arc algorithms.

Acknowledgements

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

Special thanks to Chris Valicka for prompting the closer look that led to this paper, and to Marco Ortega and Arthur Bohren for the discussions on geodesy and nonorthogonal transformations.

[7] See files Point.hpp, squareprojectfootprint_cubit.jou and squareprojectfootprint.txt from the GitHub repository https://github.com/cgvalic/GeoPlace

References

Acton, C. 1996. Ancillary Data Services of NASA's Navigation and Ancillary Information Facility. *Planetary and Space Science* 44(1):65–70.

Alexander, J. 2004. Loxodromes: A rhumb way to go. *Mathematics magazine* 77(5):349–356.

Analytic Graphics, Inc. 2019. Orbital ephemeris (agi stk data federate) for skysat-1 (39418), epoch 21 march 2019 17:42:20 utc. Electronic. Accessed 21 March 2019.

Archinal, B. A.; A'Hearn, M. F.; Bowell, E.; Conrad, A.; Consolmagno, G. J.; Courtin, R.; Fukushima, T.; Hestroffer, D.; Hilton, J. L.; Krasinsky, G. A.; et al. 2011. Report of the iau working group on cartographic coordinates and rotational elements: 2009. *Celestial Mechanics and Dynamical Astronomy* 109(2):101–135.

Barequet, G., and Elber, G. 2005. Optimal bounding cones of vectors in three dimensions. *Information Processing Letters* 93(2):83–89.

Bennett, G. 1996. Practical rhumb line calculations on the spheroid. *The Journal of Navigation* 49(1):112–119.

Bibring, J.-P.; Langevin, Y.; Poulet, F.; Gendrin, A.; Gondet, B.; Berthé, M.; Soufflot, A.; Drossart, P.; Combes, M.; Bellucci, G.; et al. 2004. Perennial water ice identified in the south polar cap of mars. *Nature* 428(6983):627.

Chamberlain, R. G., and Duquette, W. H. 2007. Some algorithms for polygons on a sphere. Technical report, Pasadena, CA: Jet Propulsion Laboratory.

Hachenberger, P., and Kettner, L. 2019. 2D boolean operations on nef polygons embedded on the sphere. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition.

Jorda, L.; Gaskell, R.; Capanna, C.; Hviid, S.; Lamy, P.; Ďurech, J.; Faury, G.; Groussin, O.; Gutiérrez, P.; Jackman, C.; et al. 2016. The global shape, density and rotation of Comet 67P/Churyumov-Gerasimenko from preperihelion Rosetta/OSIRIS observations. *Icarus* 277:257–278.

Klein, P. P. 2012. On the ellipsoid and plane intersection equation. *Applied Mathematics* 3(11):1634–1640.

Knight, R., and Chien, S. 2006. Producing large observation campaigns using compressed problem representations. In *International Workshop on Planning and Scheduling for Space (IWPSS-2006)*.

Knight, R.; Donnellan, A.; and Green, J. 2013. Mission design evaluation using automated planning for high resolution imaging of dynamic surface processes from the iss. In *International Workshop on Planning and Scheduling for Space (IWPSS 2013)*.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6(5):367–381.

Li, J.; Wang, W.; and Wu, E. 2007. Point-in-polygon tests by convex decomposition. *Computers & Graphics* 31(4):636 – 648.

Malin Space Systems, Inc. 2005. Mars Reconnaissance Orbiter (MRO) Context Camera (CTX) Instrument Description. Electronic (web). [Online - accessed 4 April 2019].

Margalit, A., and Knott, G. D. 1989. An algorithm for computing the union, intersection or difference of two polygons. *Computers & Graphics* 13(2):167–183.

McKinnon, J. 2012. Finding the line of intersection of two planes. Electronic (web).

Mitchell, S. A.; Valicka, C. G.; Rowe, S.; and Zou, S. X. 2018. Footprint placement for mosaic imaging by sampling and optimization. In *Int. Conf. on Automated Planning and Scheduling, ICAPS*, 8.

Mysen, E. 2004. Rotational dynamics of subsolar sublimating triaxial comets. *Planetary and Space Science* 52(10):897 – 907. Nonlinear Processes in Solar System Plasmas.

Planet Labs, Inc. 2018. Planet imagery product specifications. techreport, Planet Labs, Inc.

Semenov, B. V., and You, T.-H. 2006. Mars Reconnaissance Orbiter Aerobraking SPK file, MRO NAV Solution (mro_ab.bsp). Binary SPICE kernel.

Shao, E.; Byon, A.; Davies, C.; Davis, E.; Knight, R.; Lewellen, G.; Trowbridge, M.; and Chien, S. 2018. Area coverage planning with 3-axis steerable, 2d framing sensors. In *Scheduling and Planning Applications Workshop , International Conference on Automated Planning and Scheduling (ICAPS SPARK 2018)*.

Spudis, P.; Bussey, D.; Baloga, S.; Cahill, J.; Glaze, L.; Patterson, G.; Raney, R.; Thompson, T.; Thomson, B.; and Ustinov, E. 2013. Evidence for water ice on the moon: Results for anomalous polar craters from the lro mini-rf imaging radar. *Journal of Geophysical Research: Planets* 118(10):2016–2029.

Van wal, S., and Scheeres, D. J. 2016. The lift-off velocity on the surface of an arbitrary body. *Celestial Mechanics and Dynamical Astronomy* 125(1):1–31.

Wikipedia contributors. 2019. Equirectangular projection — Wikipedia, the free encyclopedia. [Online; accessed 4-April-2019].