AN EFFICIENT ALGORITHM FOR CALCULATING THE INTERSECTION OF TWO OR MORE CONVEX CONES.

C. M. O'Shea[1], F. S. Turner[1], J. E. Turner[1], E.P. Turtle[1], and G. W. Patterson[1]. [1] Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Rd, Laurel, MD 20723 (Colleen.OShea@jhuapl.edu)

Introduction: Planning for, evaluating, and visualizing constraints of instruments on spacecraft involves many complicated but tractable mathematical problems. Solving for occultations, calculating illumination angle constraints, and drawing the field of view of an instrument on a body all come with their own complications. Most issues arise when the field of view or constraint cone moves off the limb of the body that the instrument is observing. Thankfully, many of these calculations can be simplified to the same problem: the intersection of two or more convex cones.

Here we describe an efficient algorithm that can be used to find the intersection of two or more convex cones in free space. This algorithm will be useful in many instances, including all those described in the above introduction. We will present the method of applying this algorithm to drawing the field of view of the Europa Imaging System (EIS) narrow angle camera (NAC) on Europa's surface.

The Problem: Consider solving for the occultation of a planet by its moon, from the point of view of a spacecraft instrument. We can set this problem up by creating a cone that is the instrument's field of view, and another representing the spacecraft's view of the moon. We want to know if these cones intersect. The task of drawing an instrument's field of view on a body is a similar one. We create two cones in the same fashion, but we want to know if and *how* these cones intersect.

It turns out there are actually several cases to consider, taking into account whether the cones in question intersect, and if so, to what extent. Figure 1 shows how this intersection can vary depending on spacecraft position and distance from the body. Specifically, it portrays the different ways a rectangular field of view can intersect an ellipsoid. It can fully intersect as shown in (a), or not at all as in (b). Furthermore, it can be an intersection as shown in (c), (d), and (e), where the instrument is partially on-limb and partially off-limb, or is far enough away that the entire body is within the field of view.

To handle each case individually is inefficient and ambiguous because we have to first consider each point along the edge of the field of view just to identify which case we are dealing with. It is insufficient to check, for instance, just the four corners of the field of view, since in case (d) this would falsely lead us to believe the field of view does not intersect the body at all. Thus, it is necessary to identify if, and at what point, the field of view crosses the limb of the body. Once we identify which case we are dealing with, there are additional challenges and inefficiencies to solving each case. However, it can actually be simplified to just one case if we change the calculation from three dimensions to two, and take advantage of an existing Java Application Programming Interface (API).

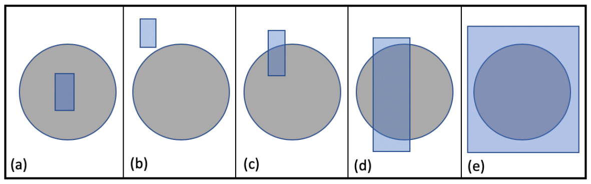

Figure 1 various cases for the intersection of a rectangular field of view (FOV) with an ellipsoid.

The Algorithm: Instead of thinking of each point around a cone individually, and whether it will intersect with the surface of the body or not, we consider the 2-D shape that is the intersection of the field of view with the elliptical limb of the body. This gives us the area of interest on the body. Any part of the cone that does not intersect the body is no longer part of the equation, and we have narrowed the problem to one case in which the entire field of view is contained on the body (shown in Figure 1 (a)).

A Case Study: In this presentation we will take a look at a specific example for which this algorithm was used. That is, a tool that is being developed to visualize and plan fly-bys for Europa Clipper which draws the accessible area of the EIS NAC on the surface of Europa under certain constraints. The NAC has a rectangular field of view, and a two-axis gimbal that enables a 60 degree cone that it can operate within. The NAC will also only operate within certain incidence angle, emission angle, and phase angle constraints [1]. Applying these constraints, the visualization of NAC potential imaging is no longer just a 60 degree cone intersected with the Europa surface, but is now a more complicated shape that is the intersection of all of the described constraints. Additionally, to get a better sense of what is being imaged at any time during a fly-by, we provide the ability to show the instantaneous field of view of the camera over time based on a NAC gimbal schedule. At any time in the gimbal schedule, it is possible that the field of view may come partially (Figure 1 (c), (d), (e)) or fully (Figure 1 (b)) off the limb of Europa, so we are faced with the challenges described above. In the following sections, we describe how we implemented this algorithm to get past these hurdles.

We start with two cones, each with its vertex at Clipper. One cone is the rectangular field of view, and one cone is the view of Europa from Clipper. By taking a plane perpendicular to the normal of each cone, we can obtain two-dimensional shapes to work with. In this example, the first two-dimensional shape is the ellipse that is the limb of Europa as seen from Clipper. The second shape is the field of view generated with a unitized boresight vector. That is, the rectangular field of view located one meter from the instrument. We set up a simple cylindrical projection, centered on the spacecraft, and project both the limb and the field of view into this projection. Now we have an ellipse and a rectangle, represented as Java Area objects in the same projection. The Area API has a function that will, given two areas, return the intersection of those areas. One call to this function gives us exactly what we want: that part of the NAC field of view which intersects the body of Europa. Once we have this area, all that is left to do is invert the projection back to three-dimensional space and "onto" the body of Europa.

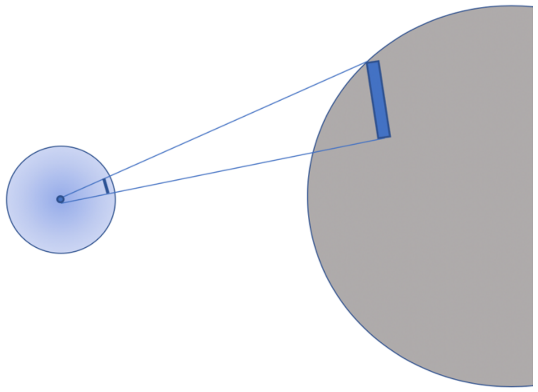

Figure 2 Finding where the computed field of view intersects the body of Europa

Drawing this area on Europa's Surface: Again, we make use of the Area API, which provides an iterator that will return all path segments that make up the resulting field of view. Iterating over these paths, we take each point and trace a ray from Clipper in the direction of that point, finding where on the body the ray intersects. Since we have already intersected with the limb, each ray is guaranteed to intersect with the body. This step is illustrated in Figure 2. The resulting list of points are on the surface of Europa and can then be combined into an Area to be rendered on a GUI or used for other calculations.

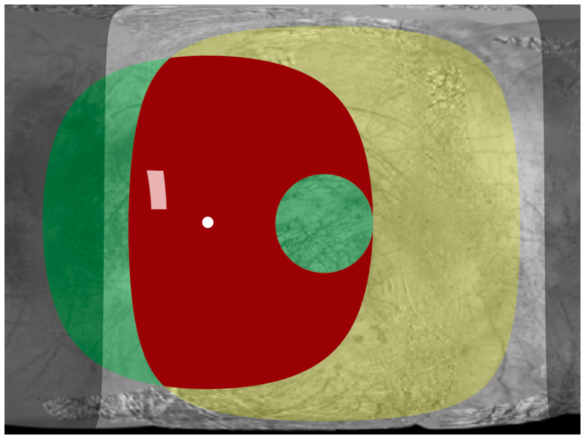

Figure 3 Emission (green) and Incidence (yellow) constraints intersected with 60 degree field of regard to show accessible area (red). Instantaneous field of view (white) shown inside potential imaging area. The white dot represents the nadir vector.

Conclusion and Future Applications: Despite the complications involved in this calculation, we were successfully able to compute and render the instantaneous field of view of the EIS NAC on Europa's surface (see Figure 3). The calculation is efficient enough that it is responsive to a time slider on the graphical user interface (GUI) and the field of view can be recomputed instantaneously when the time is updated. Additionally, the red coverage area is actually the intersection of three cones: the emission angle and incidence angle constraint cones, and the 60 degree field of regard cone. All of these shapes are calculated and rendered with no noticeable lag or delay, demonstrating the efficiency of the algorithm for greater than two cones. It is trivial to extend the algorithm to work with any number of cones.

As noted in the introduction, there are several other instances for which this algorithm will prove itself useful. Although we present the algorithm in terms of rendering a field of view on a three-dimensional body, it can also be used in free space to solve occultation problems. One such problem is to verify that the star-trackers on the Lunar Reconnaissance Orbiter (LRO) are not occulted by any object when slewing the spacecraft for a Mini-RF observation. It will also be useful in the processing of Mini-RF data for polar collects when the boresight vector comes off the limb of the body. The algorithm can be and will be generalized to find the intersection of any number of convex cones effectively and efficiently.

References: [1] Turtle E. P. et al (2019) *LPSC 50,* Abstract #3065. [2] Toussaint G. T. (1985) The Visual Computer, 1, 118-123