IMPLEMENTATION OF A WEB-BASED VIEWSHED TOOL. T. Soliman[1] and F. Calef III[1], [1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. (Tariq.K.Soliman@jpl.nasa.gov)

Introduction: A viewshed is the visible area seen from a location based on an observer's height and viewable area. Viewsheds are particularly useful for rover missions because they can aid tactical and strategic planning by answering questions like: "What can we see from this rover position?", "Where do we need to drive to see this geological feature?" and "What geologic features am I seeing in this image?" Viewsheds are often found in geographic information systems (GIS) where they are commonly rendered two-dimensionally as a region on a raster map in a binary representation to signify visible or not-visible. We implemented a viewshed algorithm based on an existing open source tool and adapted it to a web-based environment. This new Viewshed Tool has been added to the Multi-Mission GIS (MMGIS) web application and is in use by the Mars2020 Perseverance rover science and engineering team.

Algorithm: The client-side JavaScript method the new Viewshed Tool uses is based on GDAL's gdal_viewshed function [1] and on the algorithms in [2]. The decision to implement the same viewshedding algorithm ourselves was driven by the need for custom functionalities, higher user interactivity, potential performance issues, translations to non-Earth radii, and because MMGIS already implements a tiled elevation data scheme for its 3D globe view.

GIS viewsheds are generated with raster data from a digital elevation model (DEM), a two-dimensional array of elevation values that's georeferenced to a surface. The viewshed problem can be thought of as having a one-dimensional list of values from observer to destination and asking whether the first value in the list (the observer) can "see", without obstruction, the last value in the list, keeping record of the visibility, then repeating for every other possible list of values. This operation's compute-time grows with distance. The viewshed algorithm by [2] avoids this growth and achieves constant calculation time between any source and destination by utilizing an "auxiliary grid" which enables the use of previous visibility computations to inform upcoming ones. The auxiliary grid keeps track of the actual or minimum elevation a coordinate becomes visible. These values are then used to compute the actual visibility of a further DEM point by generating a "reference plane" between the observer's point and two nearby auxiliary grid values (figure 1 in [2]). It then determines whether the DEM point has a position and elevation that falls above or below the plane (and then updates the auxiliary grid again accordingly). If it falls above, it's visible in the final viewshed, otherwise it is not.

We ran into issues with incorrect viewsheds using the provided formulas in [2] and fell back on gdal_viewshed's implementation to get it functioning successfully in the final MMGIS Viewshed Tool.

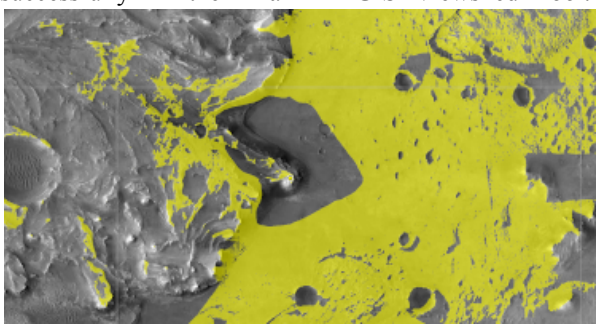

Figure 1: Example viewshed (yellow) inside MMGIS. Observer location (yellow dot) is set 2 m above the surface — approximately equivalent to the Perseverance rover mast camera's height.

Modifications: Our goal was to reshape and expand upon existing viewshed capabilities in [1] and build it out into a robust, performant, and interactive client-side tool.

The first adaption was into our tiled raster data model. Tiling is a standard optimization to serve raster data to data-rich web applications. The Viewshed Tool works with our 3D Globe's tiles which are RGBA elevation band-encoded seam-overlapped pngs in the TMS tile structure. Before each viewshed is generated, the tool queries all visible tiles at the user's current extent and zoom level and assembles them into a single grid for the base algorithm. In the cases where the observer point is not in the user's current view extent, the tool intelligently queries all the tiles between the current extent and the observer, only grabbing the tiles that will influence the final viewshed. All tiles are cached using a custom client side method that saves time between viewshed generations and in some cases delivers real-time, click-and-drag viewshed generation. Tapping into the tiled data ecosystem also enabled us to manipulate zoom levels and provide viewsheds at varying resolutions. If a user has a slow connection, they could still generate a rough viewshed, whereas a power-user can still view it at the highest resolution (i.e. deeper zoom level). By incorporating tiled data support into our viewshed algorithm, we can boost performance, reuse data, and take a resolution-based approach.

In addition, our implementation includes support to query viewshed regions within custom azimuth and elevation ranges simulating a camera field of view (FOV). We also can set a custom planetary radius to account for curvature in our viewsheds. The MMGIS

two-dimensional map is built with the open-source mapping library Leaflet, so we render the viewsheds using the leaflet.tilelayer.gl plugin.

User Interface: The tool's user interface offers an extensive suite of options (Figure 2):

- observation location
- observer height
- viewshed color, opacity, inversion
- pre-configured camera FOVs
- custom FOV and pointing in Az/El
- deep linking
- ASCII raster format export
- selectable elevation model datasets
- viewshed cloning to facilitate replicating

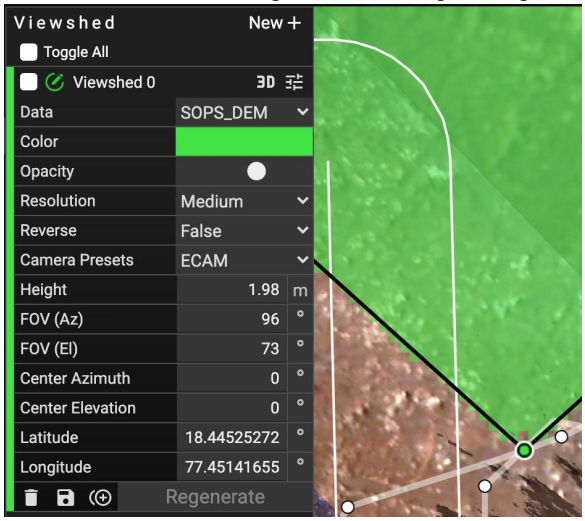

Figure 2: Viewshed options inside MMGIS.

Users can drag the observation point around the map and watch the viewshed update in real time in low-resolution mode. They can create and view multiple viewsheds on the map (Figure 3), as well as experience their viewshed in first-person in the Globe view.

Limitations: Our approach to viewshed generation has some known issues. For one, the RGBA seam-overlapped elevation tiles used as input data are a custom format which makes it a bit harder for others to adapt the tool to more generic needs. The seam fitting between tiles, while perfect for our 3D view, occasionally creates artifacts at some tile boundaries at low resolutions. However, it should be possible to replace this custom data structure with a different terrain tile format. The Viewshed Tool is dependent upon MMGIS as a plugin and it cannot stand alone. Its logic and code could however be more readily understood and adapted to fit in an individual JavaScript library. A minor constraint of the original algorithm itself is that it operates under the method of calculating the viewshed radially out from the observer. This means it is not suited for single line-of-sight sub-operations. If required, an alternative function would be needed. Lastly, the Viewshed Tool's camera preset model does not account for observer tilt (common when viewed from a rover) or camera sensor/lens shape (it's always defined as a rectangle at the size of the azimuth and elevation FOV).

Validation: We created viewsheds at known locations with a Mars Gale Crater elevation model with ESRI ArcGIS's Viewshed ArcTool to validate our Viewshed Tool's output. While there's no doubt the algorithms are different, the viewsheds we qualitatively observed at different scales appeared to coincide. We were also able to show that curvature was taken into account in our calculations by how far the viewshed extended. In addition, we compared Mars2020 rover camera images (e.g. Mastcam-Z) with their pointing values to recreated viewsheds with the new Viewshed Tool and could elucidate the oblique view of distant geographic features with what we observed in the 2D map view.

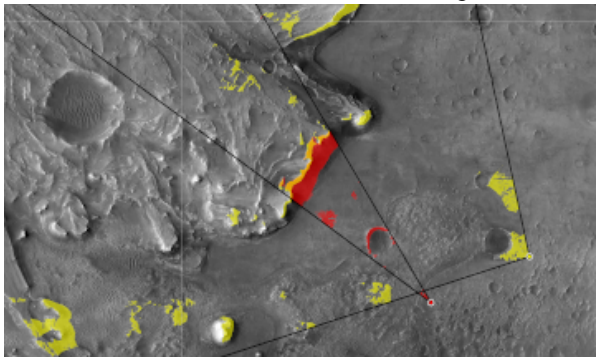

Figure 3: Multiple viewsheds from different observation points and FOVs.

Conclusion: Our new tool deployed in the web-based MMGIS mapping program allows viewshed creation on any spherical planetary body with an accompanying DEM. We've demonstrated its accuracy by comparison to commercial off the shelf (COTS) GIS viewshed tools. The Viewshed Tool is now deployed on the Mars2020 rover mission and in use by the science and engineering teams to evaluate feature locations in image mosaics and planning imaging views for future drive locations and observations.

Future Improvements: New improvements will be driven by user requests, though some additional features could include target height with evaluation (i.e. can I see this location, yes or no?") and incorporating other DEM tile dataset formats.

Acknowledgments: We'd like to acknowledge the Mars2020 rover mission and NASA Advanced Multi-Mission Operations System (AMMOS) for tool funding and support.

References:

[1] https://gdal.org/programs/gdal_viewshed.html , [2] Wang et al. 2000, PE & RS, Vol. 66, No. 1, pp. 87-90.