Technological Feasibility Analysis

11/02/21

GeoSTAC

Sponsor:
United States Geological Survey (USGS)
Astrogeology Science Center

Mentor: Melissa Rose

Team Members:
Jacob Cain
Zachary Kaufman
Gavin Nelson
Amy Stamile

Table of Contents

1 Introduction

To date, scientists have explored about 4% of the visible universe (Dunbar). Scientific knowledge of other planets is informed by data and images that astronomers have obtained from satellites and robotic planetary missions. As space exploration increases among both federal agencies and private civilians there is a need for community access to accurate up-to-date maps and data. The next big target for exploration is Mars because of the proximity to Earth inside of our solar system. Mars could be riddled with potential resources that Earth currently lacks. In order to plan this next exploration, it is very important that scientists use the data and images that our satellites and non-human space missions have gathered. Using these resources, scientists can create a map of a planet like Mars to plan for future space exploration missions. There are many tools that allow scientists to analyze and create maps from the information gathered; however, these tools are not well-developed and require the individual user to download terabytes of data on their own devices. The data that is downloaded most of the time is not analysis ready for the science community.

1.1 The Problem:

The United States Geological Survey (USGS) Astrogeology Science Center provides critical resource support for several NASA satellites and robotic planetary missions. USGS also provides support with the data processing software scientists, students and the general public. The clients have implemented a new standard for storing and accessing geological information and an image of a given location on a planet. This data is stored using the spatial temporal asset catalog (STAC). STAC provides a range of geospatial data sets, such as images, geographic data, and date and time information. This information can be easily indexed giving users access to this analysis ready data. The issue with implementing a new standard for storing and accessing data is that there are no interfaces that currently implement STAC for the planetary science community. Without interfaces that implement STAC, users must download large amounts of data and process this data on their own devices. STAC will contain assets that are typically images in a Cloud Optimized GeoTIFF format, but can also be entire packs of related images called catalogs. By having a Cloud Optimized storage system it will stop the need for having to process and display the TIFF assets on their devices instead of on a friendly web interface. USGS has all these STAC assets with no way of providing them to scientists, students and the general public.

In 2019, USGS assigned an NAU capstone team to develop a virtual planetary mapping viewer. The team that developed this new planetary viewer is called CartoCosmo. The CartoCosmo team developed an enhanced planetary viewer using

Leaflet. The CartoCosmo viewer is great for visualizing large planetary image mosaics, but has no support for visualizing these individual STAC assets or STAC catalogs. Thus, the USGS team has tasked Team GeoSTAC with upgrading and adding new features to this viewer to add the ability to visualize individual STAC assets locations on a map of Mars and load the actual selected STAC images into the CartoCosmos viewer.

1.2 Our Solution:

The team will be implementing new features to the CartoCosmo viewer using JavaScript and Leaflet plugins containing the required mapping and rendering capabilities to support the new STAC standard. The team will ingest the USGS STAC catalog made of JSON to gather image footprints - an image's true location on Mars - in geoJSON format. The team will then implement a new feature in the existing Leaflet, an open-source JavaScript Library for developing interactive maps, to show these footprints using simple polygons on the existing maps. Users will be able to select a single polygon or multiple polygons using a selector tool to load the Cloud Optimized GeoTIFF assets. This new Leaflet viewer will be connected to USGS STAC catalog with the ability to provide a convenient way for a user to functionally search for images that contain or match specific input data. In the section below, the technological challenges will identify the challenges that need to be addressed when implementing the solution for this project.

2 Technological Challenges

The first technical challenge of this project is accessing the STAC catalog data using a web-based solution using JavaScript or a similar technology. This may involve loading only a narrow subset of the available data. Tiles outside of the user's view or zoom range will not be loaded until necessary, as the complete data is too large to be streamed all at once on a web page. The formats that the client uses are optimised to be partially loaded this way, but it could still be a challenge working with these formats in a web-based environment.

After mapping and data loading, selection is the next concern. Users will need selection tools to specify and locate relevant data. They should have the ability to pan and zoom around the map and select an area of interest. Ideally, users will be able to outline an area of focus by clicking points and drawing a polygon. Text search to filter through the data would be helpful: users could specify date collected or data source.

Finally, there may be multiple data results for the same area, so a layering solution will be needed to differentiate between close or overlapping images.

In summary, here is a list of the project requirements:

- Web-Based
  - HTML/CSS/JavaScript
  - Leaflet
- Loading and Displaying Map Tiles
- Loading Data formats
  - GeoJSON
  - GeoTIFF
  - STAC
- Displaying data locations/footprints
- Tile Navigation
- Ability for arbitrary maps (not only Earth)
- Text Searching
- Area Selection
- Displaying Data
  - Differentiation between data with close or overlapping locations
  - Filtering

3 Technology Analysis

The project has restrictive guidelines in terms of what software to utilize. These guidelines include expanding off an existing Leaflet modification from the previous CartoCosmos capstone project. Therefore, there is a limited range of technology to choose from. That said, various plugins that have been designed for Leaflet. Thanks to Leaflet's diverse user base, many solutions are already available. Some requirements for the project are rendering GeoTIFF images in Leaflet, overlaying GeoJSON polygons on existing planetary mosaic images, searching with text, and making selections. This section explores these issues in detail and the team's analysis of what plugins met the requirements for this project.

3.1 GeoTIFF Rendering in Leaflet

GeoTIFF is a file extension that contains geographic metadata describing the spatial location of an individual pixel. There will be references to these GeoTIFFs inside of the GeoJSONs that are obtained from the STAC catalog. Currently the existing Leaflet that CartoCosmos developed does not support GeoTIFFs. A GeoTIFF viewer will need to be implemented inside of the existing Leaflet map that CartoCosmos has already developed. The team's goal is once a polygon inside of our Leaflet map is selected, the corresponding GeoTIFF asset will load into a viewer. This viewer will be able to render multiple GeoTIFFs inside of it providing the ability to choose an individual or multiple footprints. The viewer should allow the user to interact with the GeoTIFF assets by zooming and moving around on the image. The existing GUI that is being updated uses Leaflet, an open-source JavaScript Library for developing interactive maps. In this section will go into specifics of what existing Leaflet and JavaScript plugins/libraries that can be utilized to meet the needs of rendering a GeoTIFF asset inside of Leaflet. The metrics used to decide the plugin for rendering the GeoTIFF assets are:

- Features
- Scalability
- Documentation
- Maintainability

These metrics will judge the different Leaflet and JavaScript plugins/libraries for rendering GeoTIFFs inside of Leaflet. The first metric determines whether the approach has an easy-to-use viewer. The viewer cannot be difficult to use or understand because it needs to be for people of all backgrounds. The second metric determines if multiple GeoTIFF assets can be loaded inside of a single viewer. The third metric will provide guidelines on how to use and implement the plugin into the existing Leaflet map that CartoCosmo developed. The last metric is to ensure that the plugin is being maintained and updated regularly.

3.1.1 Candidates:

The two potential candidates evaluated to successfully render GeoTIFF assets are Leaflet GeoTIFF-2 plugin and GeoRaster Layer for Leaflet. Leaflet GeoTiff-2 is a Leaflet plugin for displaying GeoTIFF asset data. Assets can be drawn as colored rasters or with direction arrows. These layers can be clipped using a polygon. GeoRaster Layer for Leaflet is also a Leaflet plugin. It is capable of loading and displaying large scale GeoTIFFS greater than hundred megabytes and supports custom rendering such as colors, directional arrows, and context drawing. In the subsections each solution specifics will be looked at if they meet the metrics that the team has decided upon.

Leaflet GeoTIFF-2 Plugin:

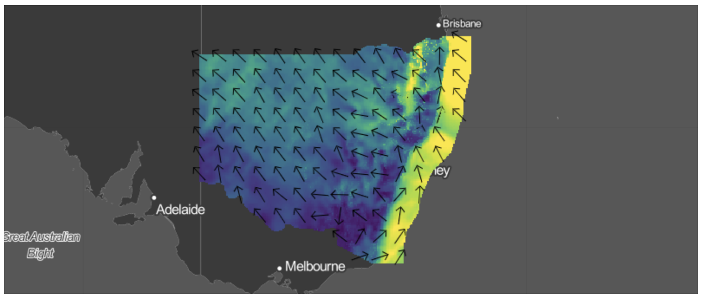


Figure 1.1: Leaflet GeoTIFF-2 Plugin Example

The first solution involves using Leaflet GeoTIFF-2, an official LeafletJS plugin for displaying GeoTIFF raster data originally developed by Stuart Matthews, but now is under CSIRO's open source software license agreement. This plugin was created because there was not an easy way of viewing GeoTIFF raster data inside of web browsers. Browsers support all different kinds of image extension, but the tiff format is not among that list. By creating a Leaflet plugin, anyone can look at the rendered GeoTIFF raster data inside of any browser given that Leaflet is part of the JavaScript library. This plugin supports all Leaflet versions after Leaflet 0.7.7.This plugin uses geotiff.js to render the GeoTIFF raster data from the tif asset and then displays the data. The data can be drawn as a colored raster or with direction arrows.

This plugin has the ability to load and render multiple GeoTIFF assets inside of our already existing Leaflet map. There would be two possible ways of loading the GeoTIFF asset data into a given map, the first would be creating a new Leaflet when clicking on a drawn polygon. The other option would be just loading all of the GeoTIFF assets that are gathered from the STAC catalog on top of CartoCosmo map as new polygons. By not using a new GUI application it will have all the same functionality that base Leaflet already has such as zooming and moving around a given image. The data can be displayed very simply using either a custom color scheme as seen above or using arrows to display the raster data to the user. In the example above, both the custom color scheme and arrows are displayed allowing the user to further understand the raster information.

In terms of implementability and maintainability, the given plugin can easily be installed using npm install giving the team access to all the written classes that will be needed for rendering and displaying a given asset. This plugin is open source on both GitHub and npmjs.com. This has been contributed to recently in the last month, making it version 1.0.0 on both GitHub and npmjs.com. This plugin gets new updates continuously with any one being able to contribute being an open source project. After looking into this Leaflet plugin the team found:

- Features: Custom colored or with Arrows of raster data
- Scalability: Yes
- Documentation: One example, GitHub Guide
- Maintainability: Updated to Version 1.0.0 on 9/30/2021

GeoRaster layer for Leaflet Plugin:

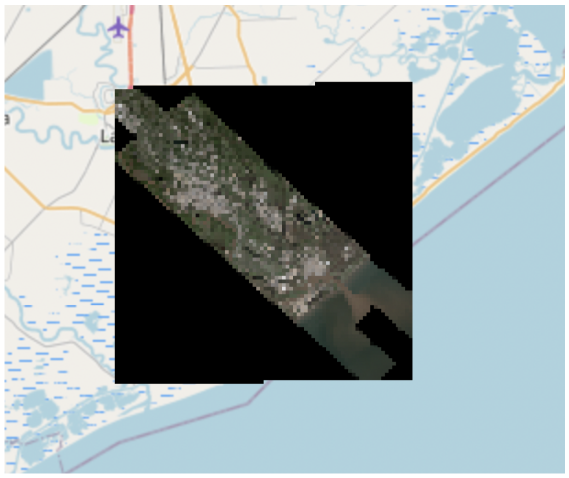


Figure 1.2: GeoRaster Layer for Leaflet Plugin Example

GeoRaster layer for Leaflet is a Leaflet plugin that allows visualization of GeoTIFFs on a web map. This plugin is created by Daniel J. Dufour of GeoSurge. GeoSurge is a geospatial technology company with a focus on data compression, natural language processing, remote sensing and visualizations. He works on quite a view open source projects for the geospatial field.

GeoRaster layer for Leaflet is a newer open-source plugin for leaflet that has been maintained on GitHub. The main reason that the GeoRaster layer for Leaflet was created was to support all projections. GeoRaster was also created to make rendering GeoRaster data load faster inside of Leaflet. It uses a unique technique for rendering called neighbor interpolation. This means it is rendering pixels inside of the Leaflet map by assuming the values of the pixels right next to the current pixel that is being rendered.

GeoRaster is able to load multiple GeoTIFFs, especially at a larger scale. GeoRaster allows for Server-Free GeoTIFF visualization by not relying on WebGL. The GeoTIFF is able to load over top of the CartoCosmo map like seen above or load a different Leaflet for all the different GeoTIFFs that are selected. GeoRaster would be very easy to use allowing for custom rendering including custom colors, directional arrows and context drawings. GeoTIFF has the ability to either give the initial GeoTIFF raster data or make it more comprehensible to users using these custom renderings. This is a nice addition to GUI allowing the users to decide what kind of information they would like to see.

In terms of implementability and maintainability, the given plugin can easily be installed using npm install giving open access to all the written classes needed to use for rendering and displaying a given GeoTIFF asset. This plugin also has a very thorough guide on GitHub with examples to fully walk you through on how to implement the GeoRaster layer for Leaflet plugin. This plugin is open source on both GitHub and npmjs.com. This has been contributed to recently in the last month since looking at it, making it version 3.4.0 on both GitHub and npmjs.com. It has also had an open talk about the plugin on 9/30/2021 talking about the new features that have been installed. This plugin seems to get new updates continuously with people continuously working on this open source project. Overall the GeoRaster specifics that were found are:

- Features: Original TIFF image, custom colored or with arrows of raster data, fast rendering, not relying on a server
- Scalability: Yes
- Documentation: 5 Examples, Step by Step guide for use
- Maintainability: Updated to Version 3.4.0 on 9/30/2021

3.1.2 Summary

Plugin	Features	Scalability	Documentation	Maintainability	Total
Leaflet GeoTIFF-2	3	5	2	5	15/20
GeoRaster Layer for Leaflet	5	5	5	5	20/20

Table 1: Analysis Results for Rendering GeoTIFFs in Leaflet

In order to compare the two different plugins for rendering GeoTIFFs inside of Leaflet, they were ranked on a 1-5 scale. On this scale 1 would be the worst and 5 would be the best. As seen above in Table 1, both the plugins have close ranking scores. They both have the ability to render multiple GeoTIFFs inside of one given Leaflet. They have both been maintained and updated recently and seem to have continued support. Ultimately, the GeoRaster layer for Leaflet is a better choice because it slightly edges out the competition of Leaflet GeoTIFF-2 plugin. GeoRaster has more features allowing to make the same customizability that GeoTIFF-2 plugin allows with displaying the raster data from the GeoTIFF image with different colors or arrows. GeoRaster takes it one step further by allowing the original GeoTIFF image to be displayed inside of the Leaflet as a layer that can be hidden away or brought forward if selected. Also, as seen above GeoRaster has a more comprehensive guide when it comes to implementing the plugin into the existing CartoCosmo Leaflet. To further test these solutions, GeoSTAC will be creating multiple test Leaflets with known good maps of places on Earth. These demos will be given to the clients to make sure they are meeting the expectations of how the GUI should handle rendering of the GeoTIFFs.

3.2 Search Images Functionality

With innumerable images and data sets available in the STAC database, the goal is to provide a convenient way for users to search for images that contain or match specific input data. To do this, the team will be using various search driven plug-ins that are supported natively by Leaflet. By searching the input data and matching those parameters to metadata from the STAC items, the goal is to display the corresponding image(s) to the user via the Leaflet engine.

The desired characteristics with image search functionality is to be able to display various locations of interest. Visualizing these locations will be based on the metadata of the corresponding STAC item of the image. A key characteristic of this functionality is allowing the user to input specific data about a location or feature they would like to view. Taking in this data would then allow parsing the STAC items to find correlations between the input data and metadata thus displaying the correlating image(s) to the user.

3.2.1 Candidates:

The plugins that are candidates for the search functionality in Leaflet are: Leaflet GeoJSON Autocomplete, Leaflet Underneath, Leaflet Search and Leaflet Fuse Search. Each section provides a detailed analysis of each plugin in terms of what searching capabilities each provides and describes how they match with specifications needed for this product.

Leaflet GeoJSON Autocomplete

Leaflet GeoJSON Autocomplete provides a structured autocomplete service for searching GeoJSON's. With this plugin, the search engine for searching for specific metadata within a STAC item will provide autocomplete suggestions for the user requiring them to only know part of the data they would like to view. This is useful as a user may only know part's and pieces of information they would like to search and this is where Leaflet GeoJSON will assist the user in suggesting similar/matching search results.

Leaflet Underneath

Leaflet Underneath provides a functionally structured search. This plugin can be structured on the backend providing a seamless and automated search upon a user request. It can also be structured for a GUI experience, allowing the user to click through various options to display specific footprints that match the search results from the metadata.

Leaflet Search and Leaflet Fuse Search

Alternatively to the plugins listed above, Leaflet Search can be used in conjunction with Leaflet Fuse search. Using Leaflet Search and Leaflet Fuse Search would provide another path to implement image search functionality.

- Leaflet Search
  - Leaflet Search is a npm package that provides control for marking specific points on a map. This plugin functionally searches for custom properties within the metadata of a layer group or GeoJSON, then displays a point for the location of those custom properties.

- Leaflet Fuse Search
  - Leaflet Fuse provides a GUI panel for users to visually mark locations on a map that contain set features defined by their respective GeoJSON. Leaflet Fuse will allow for display points of interest that are predefined within the metadata of the STAC items.

With the combination of using Leaflet Search and Leaflet Fuse Search to develop image search functionality, this will provide the user with individual STAC items that match the given criteria of the image search While Leaflet Search is good for providing custom metadata property searches, it lacks in providing a easy interface for user queries. This is where Leaflet Fuse Search comes in to provide a driver for the user interaction with custom metadata search requests.

Ideally, by using a combination of these plugins into the product, this will allow users to search for specific data in the STAC catalog. This will then pull the relevant individual STAC items that contain matching metadata with the input request from the user. With the correlating STAC items that pertain to the user's search, the Leaflet map will display the image that matches the requested search.

3.2.2 Summary:

After reviewing these plugins further, it was clear that two of these plugins aligned with the project's needs more than the others. The built in functionality of Leaflet GeoJson Autocomplete provides support for matching part or all of a user's input with custom properties contained in a STAC item's metadata, which then the information can then pass to Leaflet Underneath. Once the information has been passed to Leaflet Underneath this will allow for a full structured search of the input data across every STAC item contained in any given STAC catalog. Leaflet Underneath provides two unique ways of searching for this data:

1. Being fully autonomous to the user and relying on the back end logic to find the relevant footprints and the other

2. Being a GUI interface that allows the user to be more engaged with the selection of the various data matching footprints.

With these two plugins working together the team will have the structure needed to implement image search functionality into the product.

When testing the functionality of each of these plugins both offered functional support to implement the functionality of image searching based on metadata into the product. The Leaflet GeoJSON plugin was not only able to autocomplete the partial search query, it also gave various suggestions for possible searches based on the provided input. Overall, the Leaflet GeoJSON plugin meets the expectations set for searching through metadata to match a partial or full search query.

The testing of the Leaflet Underneath plugin showed that this plugin is capable of searching though the images on a map and providing correlating data to the user once a footprint is interacted with. With the Leaflet GeoJSON plugin doing all of the leg work for structuring these searches, Leaflet Underneath will then be able to take this data in and display requested data to the user represented on any given footprints.

3.3 GeoJSON Tile Layer in Leaflet

GeoJSON is the data format the team will be utilizing from the STAC catalog, specifically a STAC Item. The geometries within a GeoJSON's feature object provide image polygon coordinates. A Tile Layer will be needed to plot these polygons over an existing image layer. For example, the Leaflet map CartoCosmos, has mosaic image layers for different target bodies in the solar system. The goal is to then add an additional GeoJSON tile layer over the existing mosaic layer that provides the image polygons pertaining to a particular planetary mission of that target body. This section will go into the specifics of what existing Leaflet plugins can utilize to meet the needs to overlay these STAC item image polygons.

The metrics the team will use to determine which tile layer plugin best fits the team's needs are:

- Ease of Use
- Maintainability
- Visual Appeal
- User Interactivity

These metrics will allow the team to quantitatively determine how to resolve the GeoJSON tile layer issues the most effectively. The team is looking for ease of use in which the user interface is easy to use and understand. This can be determined by simple tool options and minimal buttons. The team is also looking for maintainability. It is ideal to find a plugin that is regularly maintained or open sourced so that future maintainers of CartoCosmos can keep the Leaflet application relevant. The team also would like to ensure that the tile layer has a visual appeal. Since this project is front-end focused, visual aspects play an important role in the use of the CartoCosmos application. Therefore, the goal is to have a tile layer that has a good visual representation of polygons and text. Finally the team is looking for an interactive component. This includes the ability to have adaptive polygon lines that change size when zooming into the tile layer. The team also would like to have the ability for polygons to be highlighted when hovering to indicate the ability to select and view information.

3.3.1 Candidates

The team has narrowed down three possible plugin candidates for solving the GeoJSON tile layer issue. There were a few plugins that could not be considered due to their limitations of relying on an older version of Leaflet. Other plugins did not meet the needs due to the incompatibility with the GeoJSON format.

The following solution subsections will focus on the specifics of what each plugin provides in terms of usability, maintenance, visual appeal, and user interactivity.

Leaflet.VectorGrid

Leaflet.VectorGrid is an official Leaflet plugin that is provided on the Leaflet Library website. This plugin was originally created by Iván Sánchez who has created various official Leaflet plugins. The Leaflet.VectorGrid plugin was created due to Leaflet version 1.0.0 update making vectors incompatible with current implementations of load vector tiles. This plugin displays gridded vector data using GeoJSON data. The benefit of using vector tiling is that the vectors are adaptable to zooming. This is because vectors rescale as the user zooms closer to the vector lines. This is beneficial for user visibility and allows for a smoother zooming experience for the user. The rendering of vector tiling is also faster than typical raster tiling.

As you can see in Figure 3.1, this plugin has the feature of highlighting the polygons when hovering over the tile. It also has the option of displaying a bubble of quick information for the user. The coloring of the lines provides a visual appeal of allowing the user to easily differentiate what polygons they are selecting.

In terms of maintainability, the plugin is open source but the Github site for this plugin has not seen any contributors commit in over a year with various pending pull requests. This is indicating that the maintainer has not been completely active with this repository. With that being said, there are loose copyright restrictions and the maintainer states that this plugin can be used in any way.

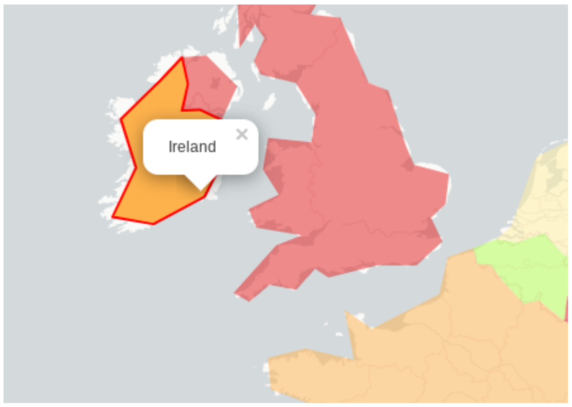


Figure 3.1: VectorGrid Layer for Leaflet Plugin Example

Leaflet-tilelayer-geojson

This plugin is also an official Leaflet plugin that is provided on the Leaflet Library website. This plugin is maintained by Glen Robertson on a Github repository. This plugin renders GeoJSON on the L.GeoJSON layer in Leaflet. Unlike the Leaflet.VectorGrid rendering vector tiles, this plugin renders raster tiles. This means that the loading time is slower and not as adaptable to user zooming. The positive side to raster tiles is they are more readily maintained and have better support in Javascript libraries.

In terms of interactivity, this plugin includes the options of hovering, clickability, and the ability to add colors to the polygons. There is also the pop up feature in which key descriptions are displayed to the user when clicking a particular polygon.

This plugin has not been updated since 2016. Alongside this, there are few pending issues and no pending pull requests on it's Github. This could be an indicator that the plugin is stable and has limited bugs. This could also be that there are limitations for improvement. This plugin is currently in use in over a dozen Github repositories and is open to contributors. Therefore, there is promise of a reliable and functioning plugin.

Leaflet.GeoJSON.Encoded

This plugin is also an official Leaflet plugin that is provided on the Leaflet Library website. This plugin is maintained by geobricks on a Github repository. This plugin is an extension to the L.GeoJSON layer by utilizing Google's polyline encoding algorithm. This means that the data transfer is optimized using this algorithm. Like the

Leaflet-tilelayer-geojson plugin, this plugin renders raster tiles. This means there is a slower and non-adaptable zooming capabilities.

The maintainability for this plugin is limited. The last update to this plugin was in 2015. There are no known users of this plugin based on Github. This could be an indicator that this plugin is stable with no need for modifications. This could also be an indicator that this plugin is limited in being able to improve. The licensing indicates that this plugin can be used with minimal restrictions.

The plugin is limited on features that include interactivity or visuals. As seen in Figure 3.2, the visualization of the polygons are standard and do include a description bubble when selected. This could be beneficial if the team decides to use standard styling for coloring polygons or utilizing hovering . The standard methods for these styling options are available within the Leaflet API therefore this option could be ideal for limiting the functionality that is relying on a plugin.

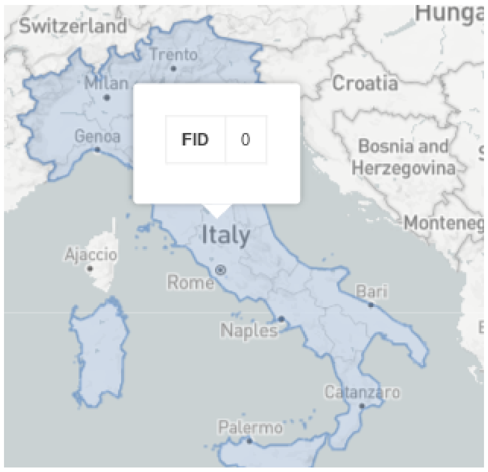


Figure 3.2: GeoJSON.Encoded Layer for Leaflet Plugin Example

3.3.2 Summary

Plugin	Ease of Use	Visual Appeal	User Interactivity	Maintainability	Total
Leaflet.VectorGrid	5	5	5	4	19/20
Leaflet-tilelayer-geojson	3	3	3	2	11/20
Leaflet.GeoJSON.Encoded	1	1	1	2	5/20

Table 2: Analysis Results for GeoJSON Tile Layers in Leaflet

The analysis in Table 2 provides a synopsis of the three potential GeoJSON tile layer plugins and how they rank in terms of ease of use, visual appeal, user interactivity, and maintainability. The Leaflet.VectorGrid plugin was the only plugin to meet all of the desired criteria. Vector polygons provide the zooming adaptiveness and rendering speed that was not possible in the other two raster plugin implementations. This plugin also is being maintained more frequently with a much more prominent Github community. Therefore the team has chosen the Leaflet.VectorGrid plugin as the choice for solving the GeoJSON tiling layer issue within Leaflet.

3.4 Selection Tools in Leaflet

Users of the application will need to select an area to examine data within. Since the client's major requirement is the use of Leaflet, this section will examine Leaflet plugins that provide selection tools. There are four potential selection shapes that can be utilized to let the user graphically select an area including: a point, a rectangle, a polygon, and a lasso.

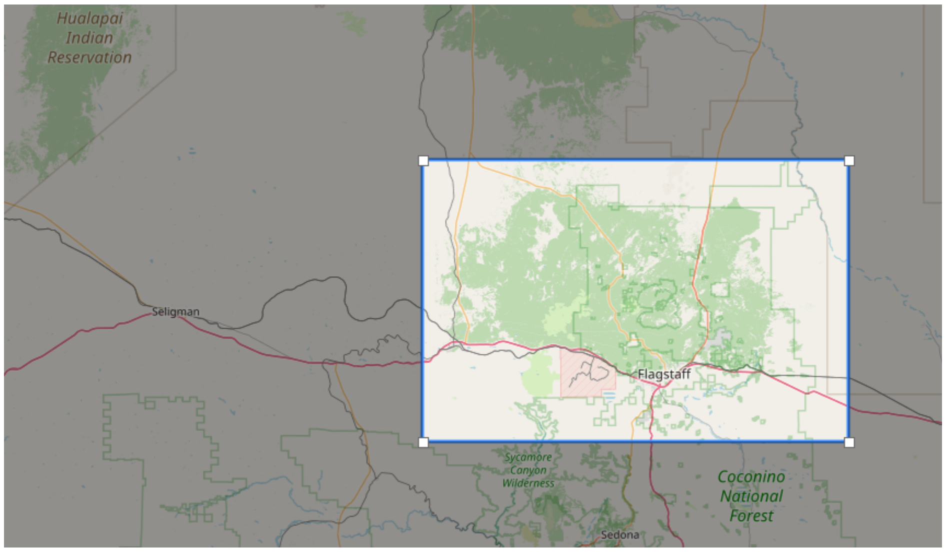
3.4.1 Candidates

Leaflet-locationfilter

The Leaflet-location filter plugin utilizes a rectangular selection shape for searching within a Leaflet map. It has the unique feature of being able to select the whole visible screen with one click. The plugin has not been updated since 2014, which could be an indicator that this plugin is not regularly maintained.

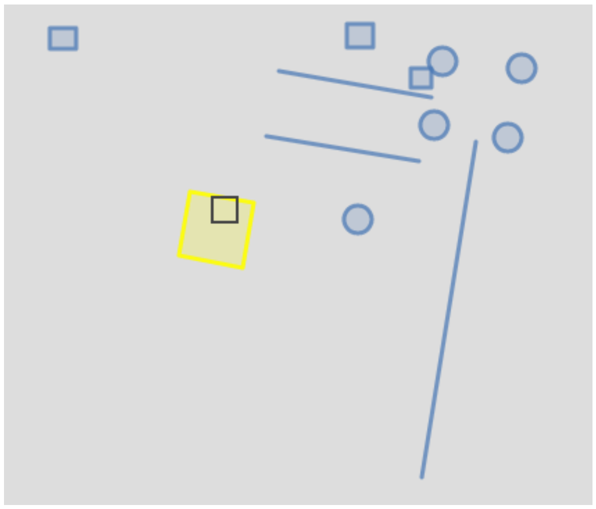
Leaflet-Shades

The Leaflet-Shades plugin utilizes a simple rectangular selection as well. It doesn't include many extra features, unlike some of the other plugins examined. It is, however, reasonably up to date in terms of maintainability.



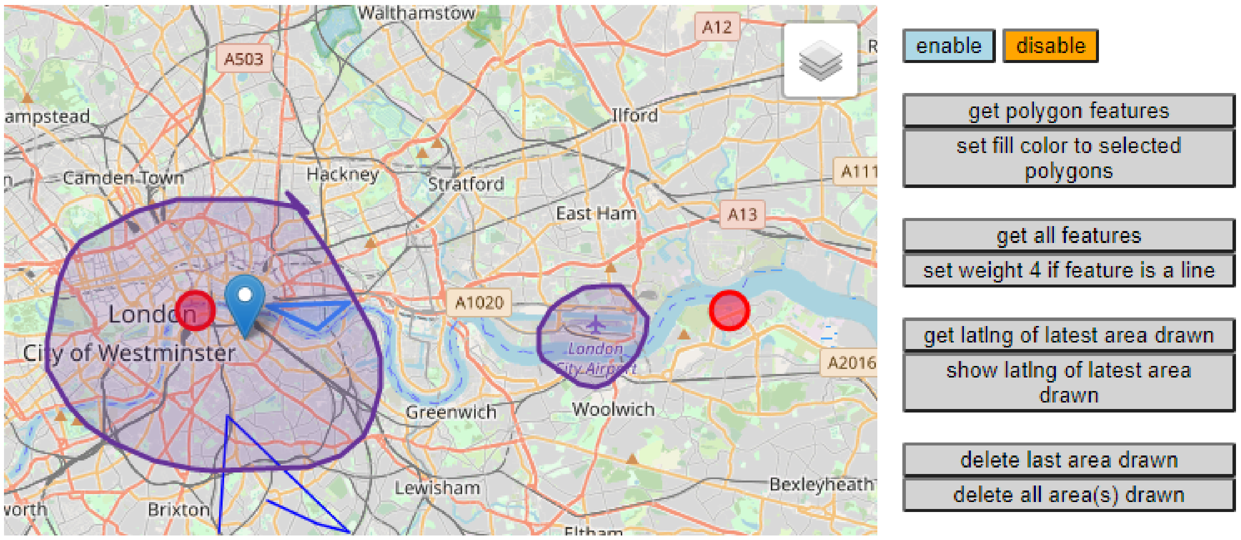
Leaflet.FeatureSelect

The Leaflet.FeatureSelect plugin uses a single selection point in the middle of the screen for searching within Leaflet. While this is inflexible in some ways, it has the advantage of being easy to use on mobile screens. This plugin also has built-in GeoJSON compatibility.



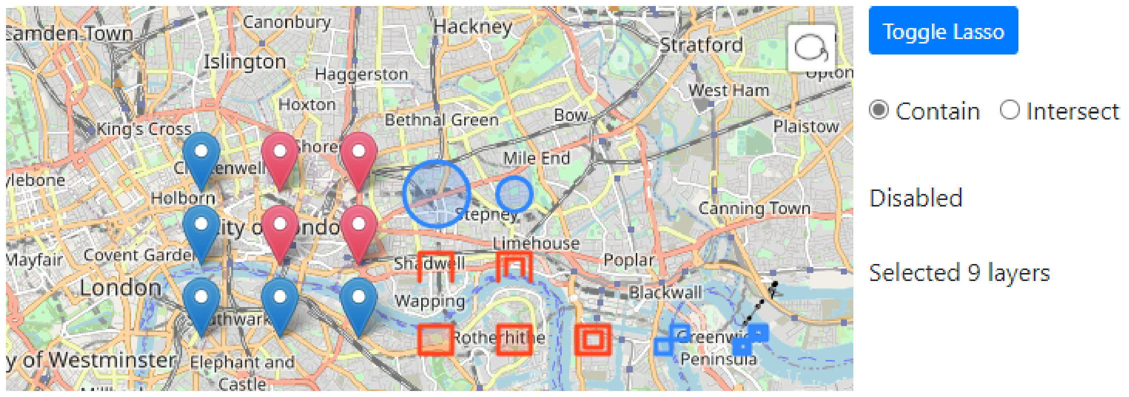
Leaflet.SelectAreaFeature

The Leaflet.SelectAreaFeature plugin uses a free polygon selection shape for searching within Leaflet. One of the advantages of this plugin is the ability to draw multiple polygons on the Leaflet map.



Leaflet-lasso

The Leaflet-lasso plugin utilizes a free polygon selection shape similar to the Leaflet.SelectAreaFeature. It can be set to select all shapes it interacts with, or the selection can be narrowed to only the shapes completely contained within the outline.



3.3.2 Summary

Plugin	Shape flexibility	Visual Appeal	Mobile Friendly	Selection Support	Ease of Use	**Total Score**
Leaflet-locationfilter	3	2	1	3	3	12/20
Leaflet-Shades	2	3	1	2	2	10/20
Leaflet.FeatureSelect	1	1	5	4	2	13/20
Leaflet.SelectAreaFeature	5	5	1	2	3	16/20
leaflet-lasso	4	3	1	4	5	17/20

Table 3: Analysis Results for Selection Tools in Leaflet

It would be ideal to be able to draw a lasso for selection, since different users would likely be interested in particular areas that don't necessarily conform to a rectangular shape. A point selection would not allow the user to specify a broad area of the map, but may be useful if the user wants to select one specific footprint.

The leaflet-lasso plugin is closest to the original design plan. With leaflet lasso, the user can draw an outline around the area of interest, and then see all the data features in that area. A helpful feature of leaflet-lasso is that it can be set to select only fully-contained objects, or any objects the lasso intersects. The only drawback of this plugin is that it does not have built-in support for GeoJSON.

A good second choice would be the Leaflet.FeatureSelect plugin. Using a point for selection was not originally considered, but has its advantages of being mobile-friendly as well as providing GeoJSON support.

4 Technology Integration

With the constraints of this project, the overall success for development will be judged by how the team implements each technological solution, as well as how these solutions link together. The requirements of this project include, using Javascript with Leaflet for translating individual STAC items into an interactive map that is capable of displaying individual images and image catalogs. With these requirements, there are specific key problems that the team must overcome. These problems must be solved in the development phase of this project in order to be successful. These problems include:

- Ability to visually display single images (or catalogs)
- Ability to display corresponding metadata to any given image
- Integrating a new standard of using STAC catalogs to display imagery
- Ability to select individual polygons that are represented on STAC catalogs
- Ability to load polygons independently for viewing

With these problems being so closely related on a functionality level, there will be a need to develop support for these features with the plugins and tools discussed previously in this document. Once these features are developed and integrated with the previous project CartoCosmos, the end product will be a planetary map powered by Leaflet with all of the functionality of CartoCosmos with the new capabilities of:

- Visually displaying single image catalog items
- Visually displaying GeoTIFF imagery from STAC items
- Individual polygon selection of STAC items represented on the map
- Ability to render individually selected polygons for viewing

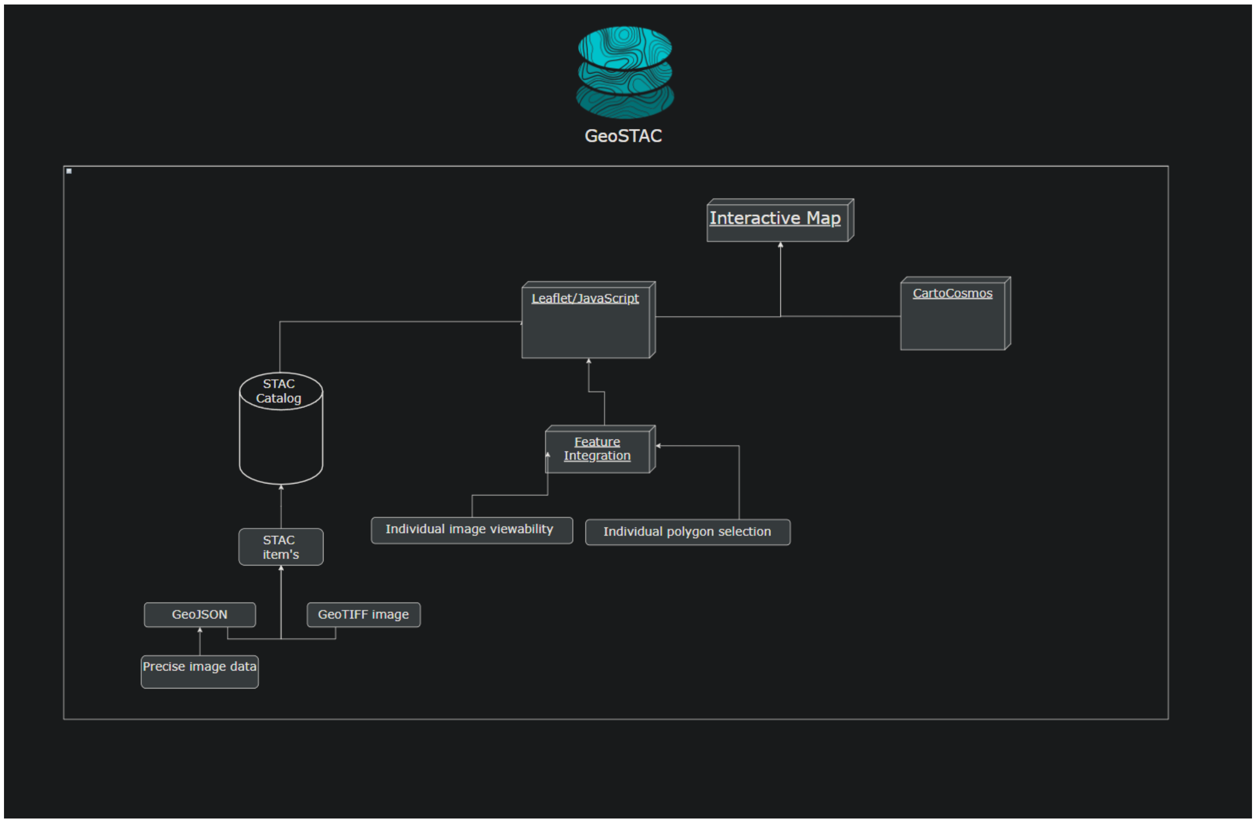


Diagram 1: GeoSTAC Interactive Map System

GeoSTAC's Interactive Map system will be composed with four main components: Leaflet, JavaScript, a STAC catalog, and CartoCosmos. The STAC catalog will contain all of the physical data needed to provide a scientific accurate interactive map. This data is grouped together in individual STAC items that are mainly constructed with a GeoJSON, and GeoTIFF image file. This STAC catalog will then be passed to Leaflet for processing. With the new feature integration working alongside the existing CartoCosmos project , specialized feature functionality will be supported allowing users to view and select individual polygon items represented by their respective GeoJSON bonds.

5 Conclusion

In conclusion, these proposed features to the CartoCosmos Leaflet map viewer will provide the planetary science community with access to visualize the STAC catalog of analysis ready planetary images. These features will ease the burden of processing, downloading, and storing planetary images alongside providing a user-friendly interface for quick access of data for scientific research. Throughout this document, the team has thoroughly researched the best tools that meet the needs of the client. The summary of solutions for each technical challenge is outlined in Table 4. This table shows each of the technical challenges for this project and the associated solution. These challenges include:

- Rendering a GeoTIFF image within CartoCosmos
- The ability to search for specific images
- The ability to overlay a tile layer for displaying footprint information
- The ability to select various footprints to provide visual data to the user

The team is confident that the selected tools will improve the existing CartoCosmos Leaflet interface and satisfy the clients overall vision for this project.

Technical Challenge	Solution
Rendering GeoTIFF in Leaflet	GeoRaster Layer for Leaflet
Search Images Functionality	Leaflet GeoJSON Autocomplete, Leaflet Underneath
GeoJSON Tile Layer to Leaflet	Leaflet.VectorGrid
Selection Tools in Leaflet	Leaflet-lasso

Table 4: Solutions Summary

6 References

CSIRO Oceans and Atmosphere - Coastal Informatics Team. (2017). *GitHub - onaci/leaflet-geotiff-2: Leaflet plugin for displaying geoTIFF raster data.* GitHub. https://github.com/onaci/leaflet-geotiff-2

Dunbar, B. (2009). *NASA - How Big is Our Universe?* NASA. https://www.nasa.gov/audience/foreducators/5-8/features/F_How_Big_is_Our_Universe.html

Cudini, S. (2012). *GitHub - stefanocudini/leaflet-search: Search stuff in a Leaflet map.* GitHub. https://github.com/stefanocudini/leaflet-search

GeoTIFF. (2017). *GitHub - GeoTIFF/georaster-layer-for-leaflet: Display GeoTIFFs and soon other types of raster on your Leaflet Map.* GitHub. https://github.com/GeoTIFF/georaster-layer-for-leaflet

Kong, M. (2017). *GitHub - mkong0216/leaflet-shades: Leaflet plugin for creating gray overlay in unselected areas and transparent overlay for selected area.* GitHub. https://github.com/mkong0216/leaflet-shades

Leaflet. (2016). GitHub - Leaflet/Leaflet.VectorGrid: Display gridded vector data (sliced GeoJSON or protobuf vector tiles) in Leaflet 1.0.0. GitHub. https://github.com/Leaflet/Leaflet.VectorGrid

Liedman, P. (2016). *GitHub - perliedman/leaflet-underneath: Find interesting features is in your map using Mapbox Vector Tiles data.* GitHub. https://github.com/perliedman/leaflet-underneath

OpenPlans. (2013). *GitHub - openplans/Leaflet.FeatureSelect: Leaflet plugin for precise feature selection.* GitHub. https://github.com/openplans/Leaflet.FeatureSelect

Özkaya, Y. (2015). *GitHub - utahemre/Leaflet.GeoJSONAutocomplete: Leaflet Search Bar For Remote Searching with GeoJSON Services.* GitHub. https://github.com/utahemre/Leaflet.GeoJSONAutocomplete

Pibia, S. (2017). *GitHub - sandropibia/Leaflet.SelectAreaFeature: Plugin that selects feature(s) by drawing an area on the map.* GitHub. https://github.com/sandropibia/Leaflet.SelectAreaFeature

Plugins - Leaflet - a JavaScript library for interactive maps. (n.d.). LeafletJs. https://leafletjs.com/plugins.html

Riche, A. (2014). *GitHub - naomap/leaflet-fusesearch: A plugin for Leaflet to search features in a GeoJSON layer using Fuse.js.* GitHub. https://github.com/naomap/leaflet-fusesearch

Robertson, G. (2012). GitHub - glenrobertson/leaflet-tilelayer-geojson: Leaflet TileLayer for GeoJSON tiles. GitHub. https://github.com/glenrobertson/leaflet-tilelayer-geojson

ŽáK, J. (2018). *GitHub - zakjan/leaflet-lasso: Lasso selection plugin for Leaflet*. GitHub. https://github.com/zakjan/leaflet-lasso