MOONTREK TELESCOPE

Status Update
CSULA - Fall Semester 2020
Sponsor: JPL

TEAM MEMBERS AND ROLES

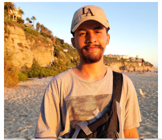
Nicolas Ojeda
Team Lead
Backend(django) , Computer Graphics(Threejs)

Alex Lamb
Communications Lead

Albert Ramirez
Lead Backend Developer

Dakota Townsend
Documentation Lead

Jacob Frausto
QA/Testing Lead, Circle Detection (Python)



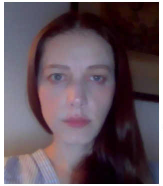
Kevin Aguilera
Development (Graphics Model)

Gerard Rosario
Front End Graphics Lead

Pavit Chawla
User Interface Lead

Elvira Sakalenka
Computer Vision Lead

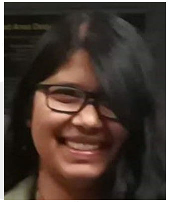
Faculty Advisor



Weronika Cwir

JPL SPONSORS

Shan Malhotra



Natalie Gallegos

Jet Propulsion Laboratory
California Institute of Technology

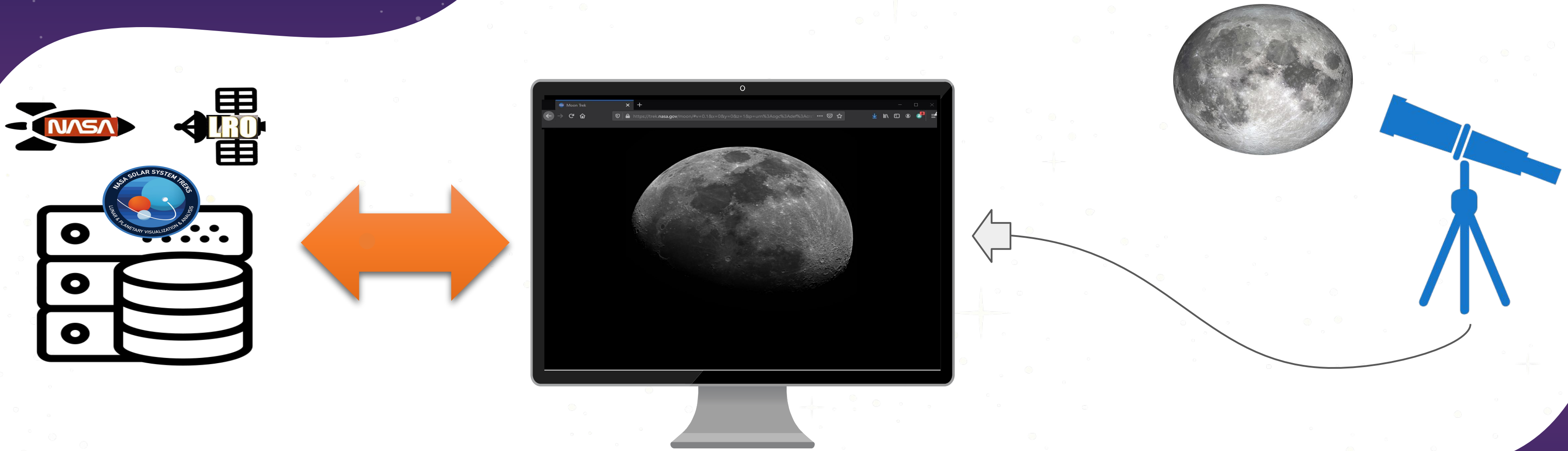
Agenda

1. Overview
2. JPL's Moon Trek Portal Overview
3. General approach
4. Circle Detection
5. Graphics Model
6. Back end
7. User Interface
8. Future Plans
9. Conclusion

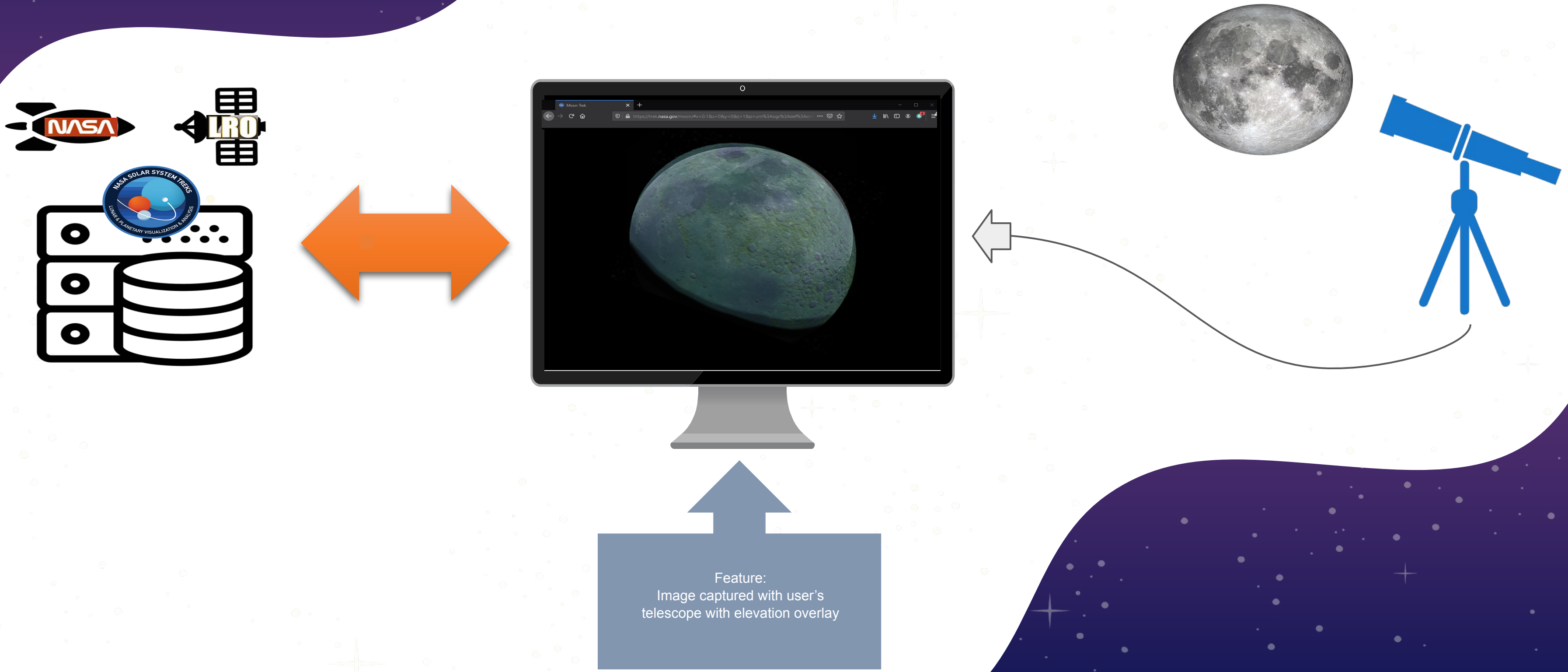
About the Project

- Telescope and JPLs Moon Trek portal interface.
- Interface: a web application that routes images of the moon from a telescope.
- Features: Important annotations such as;
  - landing sites
  - local temperatures,
  - chemical makeups of the soil -- such as iron, etc.

Moon Trek Telescope | About



Moon Trek Telescope | About

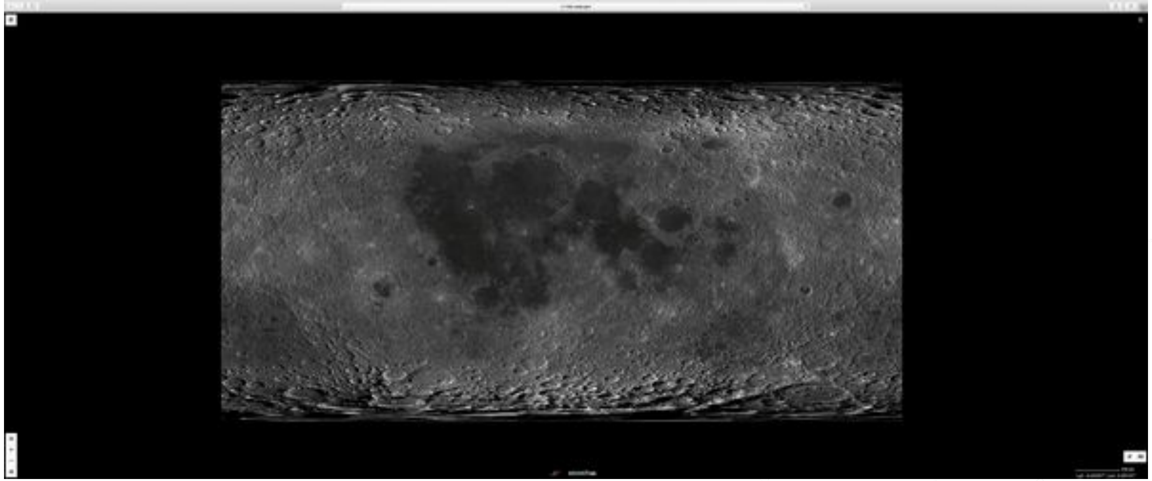


JPL'S MOON TREK PORTAL

- JPL's mapping and modeling portal of the Moon.
- Showcases data collected by NASA at various locations.
- High-resolution data sets covering most of the Moon.
    - Imagery
    - Layers : spectrometry, radiometry, gravity fields, radar, slope, roughness, mineralogy, etc.

Moon Trek Portal

Apollo 16 - Landing Site Layer Example



https://trek.nasa.gov/moon/index.html

Moon Trek Portal

Rock Abundance Layer Example

https://trek.nasa.gov/moon/index.html

General Approach

Image registration of source image to a reference image that correctly correlates with the Moon Trek Portal

Moon Trek Telescope | Approach

Image Registration

Real-time automatic registration between a view of the moon routed from a telescope(Source Image) and a map of the moon containing the corresponding coordinates of moon in JPL's Moon Trek Portal (Reference Image)

LRO WAC Mosaic emulates the Trek Portal Moon map

Location can be referenced using the pixel/degree .

Reference Image

Threejs - 3D model (snapshot of moon at given time & given location)

Goal is to exactly map the source view in the reference image .

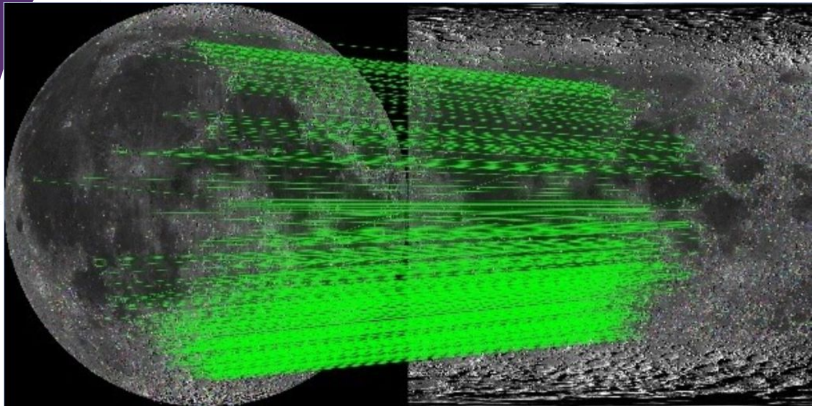
Source Image

OpenCv - Circle Detection

'time' and 'location' data

Moon Trek Telescope | Approach

Work is focused on removing Geometric distortions



All images processed by applications will not be the same scale

By performing morphological operations, on the source image prior to registration, we can generate a downsampled image that is the same scale as reference

This will better allow us to obtain the longitude latitude coordinate for each pixel in our downsampled image

Moon Trek Telescope | Approach

By extracting time and location data from an image, we can know when and where a picture was taken

We can then calculate the nearest point on the moon from the coordinates and time of geotagged picture.

We can also plot the moon in rectangular space with respect to earth.

By generating the correct moonphase, we can better register our image with the reference image.

Request

```
http://54.157.167.17:5000/nearest-point/earth/moon/-118.173225/34.195109/2020-10-07T01:10:45
```

Response

```
{
  "observer": "earth",
  "target": "moon",
  "altitude_km": 1737.4,
  "longitude": -4.551454259598997,
  "latitude": 2.151806905975941
}
```

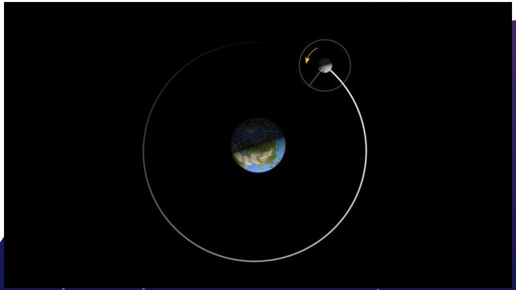
API calls*

Request

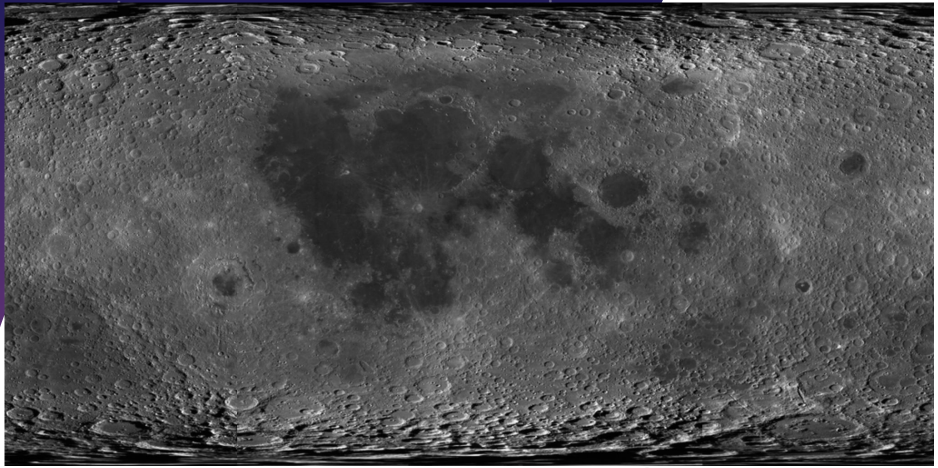
```
http://54.157.167.17:5000/lat-to-rect/moon/earth/0/0/2019-10-07T01:10:45
```

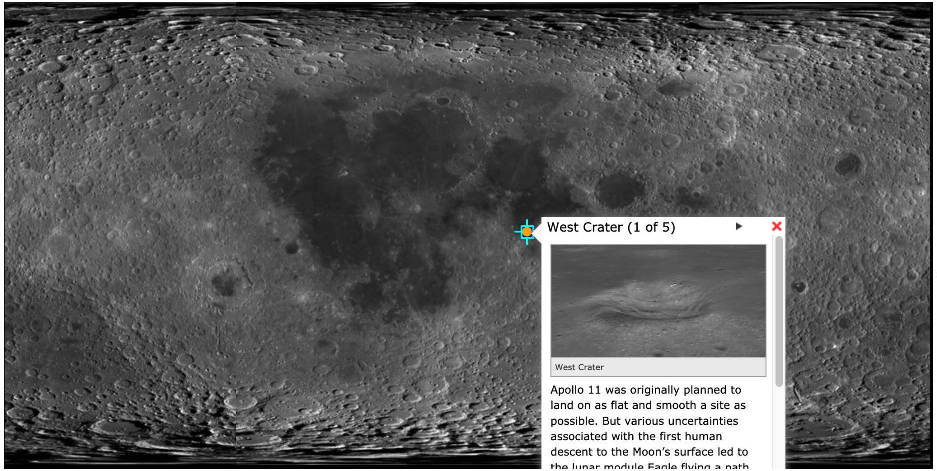
Response

```
{
    "origin": "EARTH",
    "units": "km",
    "positions": {
        "moon": {
            "x": -12282.268170093246,
            "y": -368872.8597873927,
            "z": -147150.61159076623
        }
    }
}
```



Moon Trek Telescope | Approach





- Determine what is moon in source image
- Perform morphological transformations on source image
- Register source image with reference image

Moon as an equirectangular projection and with applied data layer.

Circle Detection | Technical Challenges

To identify the moon in our images

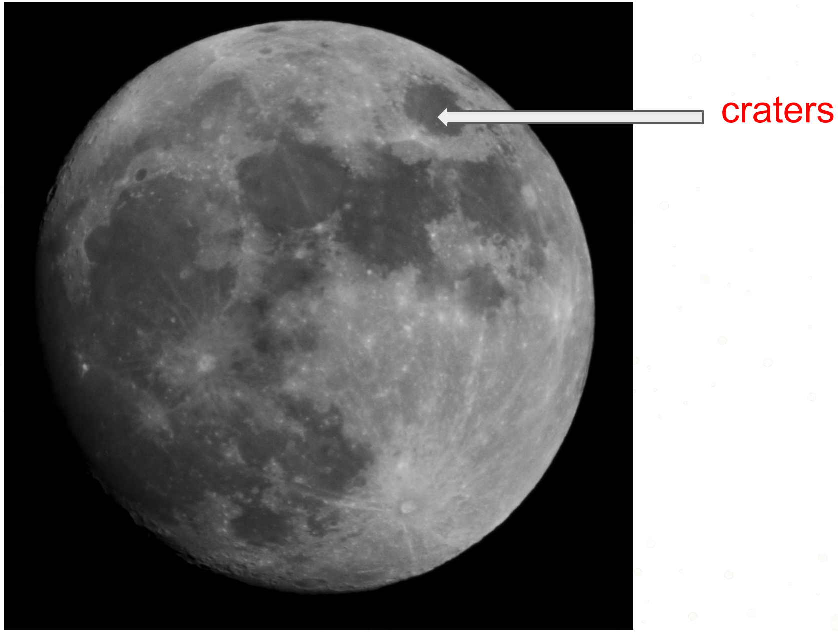
1

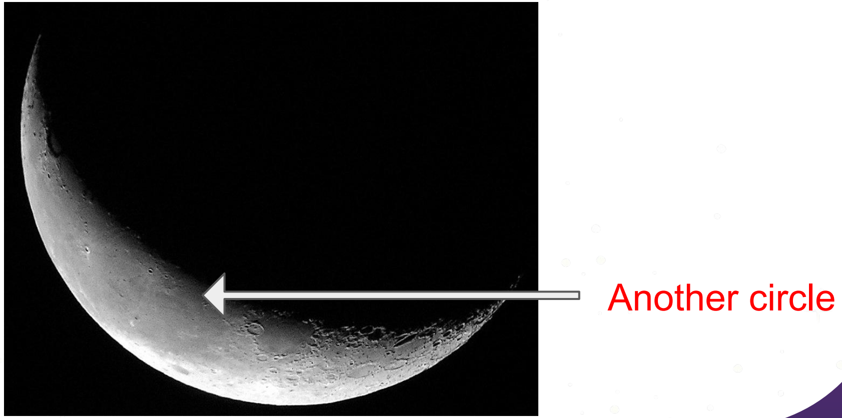
2

3

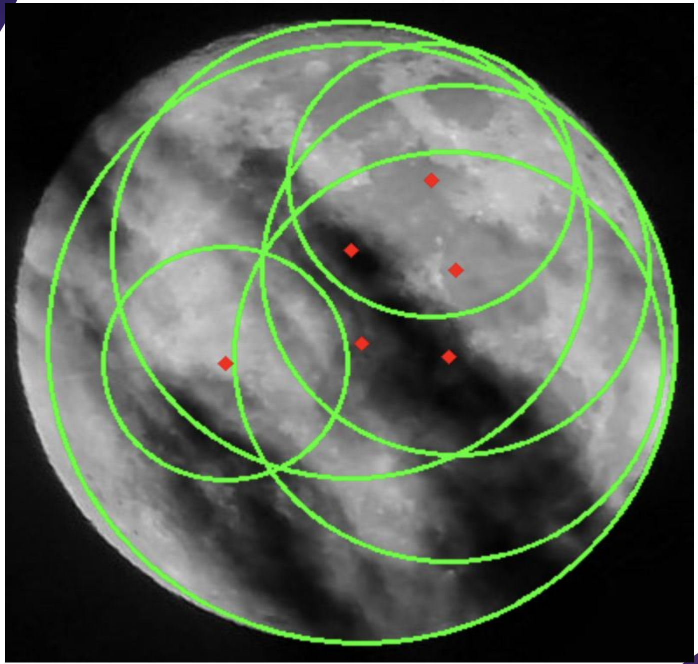
OpenCV Hough Circle Transform

The function we used is : **cv2.HoughCircles()**
Which determines circles within our images





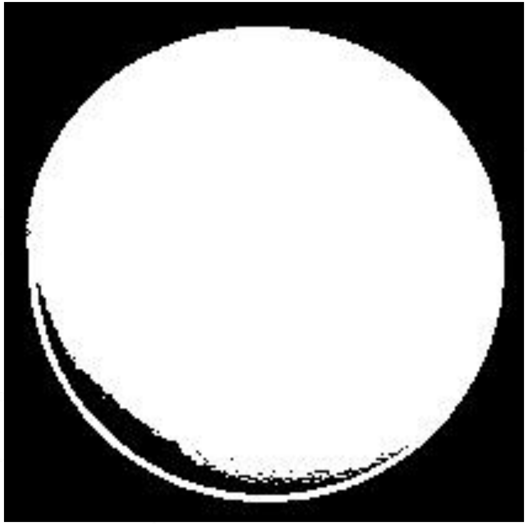
Circle Detection | Example



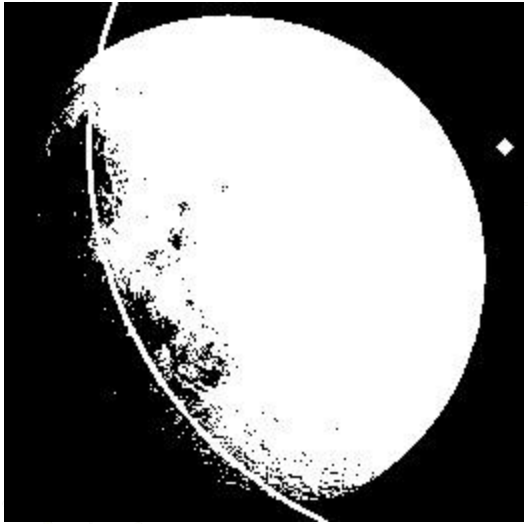
OpenCV Image Thresholding

The function we used is:

```
cv2.threshold(img,50,255,cv2.THRESH_BINARY)
```



1



2

OpenCV, Morphological Transformations (Closing)

It is useful in closing small holes inside the foreground objects, or small black points on the object.

`cv2.morphologyEx()`

OpenCV, Morphological Transformations (Dilation)

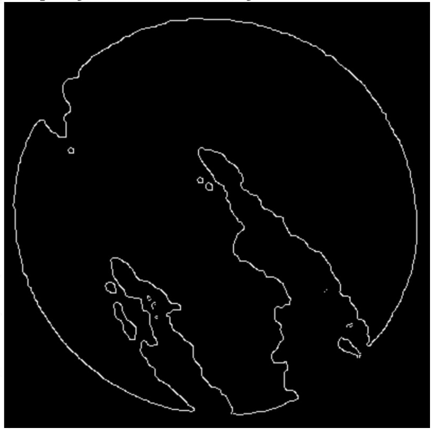
The object area increases, also useful in joining broken parts of an object.

`cv2.dilate()`

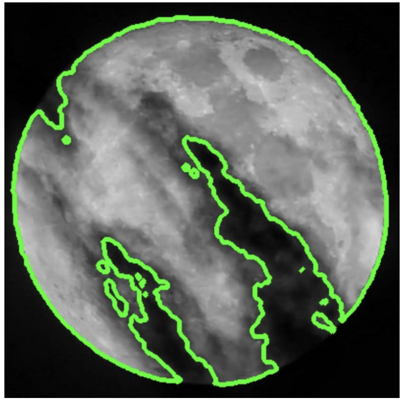
OpenCV Find and Draw Contours

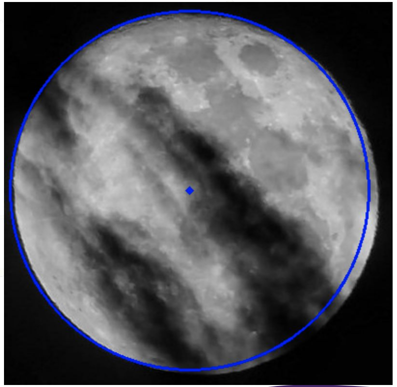
The function we used is:
`cv2.findContours()`

Canny Edges After Contouring



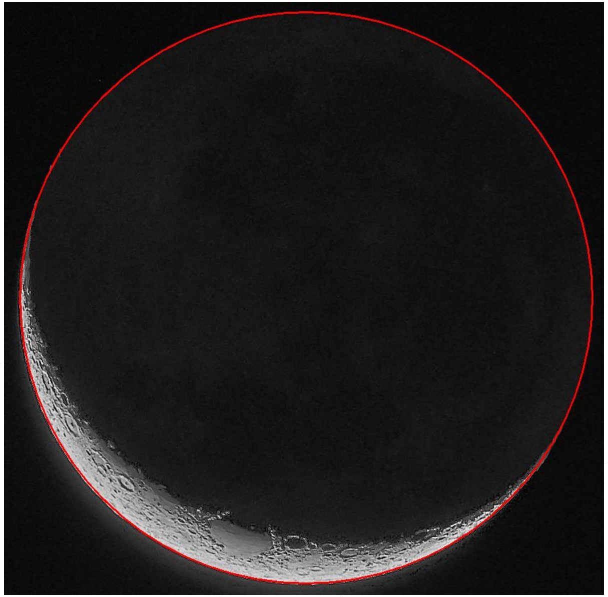
Number of Contours found = 17
Contours





Progress on Circle Detection

- OpenCV HoughCircles() technique gave the best results
- Out of 108 photos, only 6 photos did not give desired results
- Red line is the result of HoughCircles()



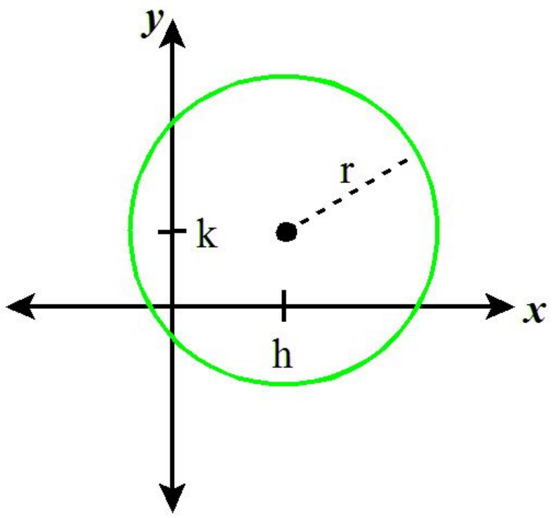
OpenCV HoughCircles() Method

- Image is first read without editing
- **Gray-scaling** converts an image into a image that a computer can manipulate

HoughCircles() Functionality

- Circle defined by three parameters: ($\mathbf{x}_{\mathbf{center}}$, $\mathbf{y}_{\mathbf{center}}$, $\mathbf{r}$)

- First stage finds possible circle centers
- Second stage finds the best radius for each detected circle centers.



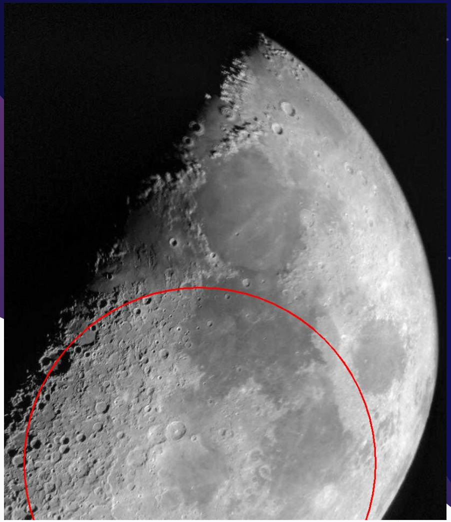
$$(x - h)^2 + (y - k)^2 = r^2$$

OpenCV HoughCircles() Method

HoughCircles() Input

- **image:** image for circle detection
- **detection method:** HOUGH_GRADIENT - method for circle detection
- **dp :** inverse ratio of resolution
- **min_dist:** minimum distance between detected centers
- **param_1:** Upper threshold for the internal canny edge detector
- **param_2:** Threshold for center detection
- **min_radius:** (unused) Minimum radius to be detected
- **max_radius:** (unused) Maximum radius to be detected

```
cv2.HoughCircles(gray_scale, cv2.HOUGH_GRADIENT, 1, 100, param1=420, param2=10)
```



Errors of Circle Detection

Progress of Circle Detection

3D Model

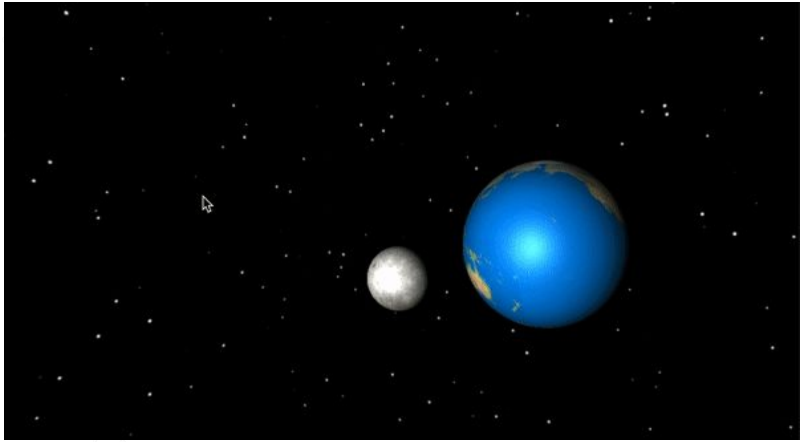
Our tool for image registration.

3D Model Overview / Purpose

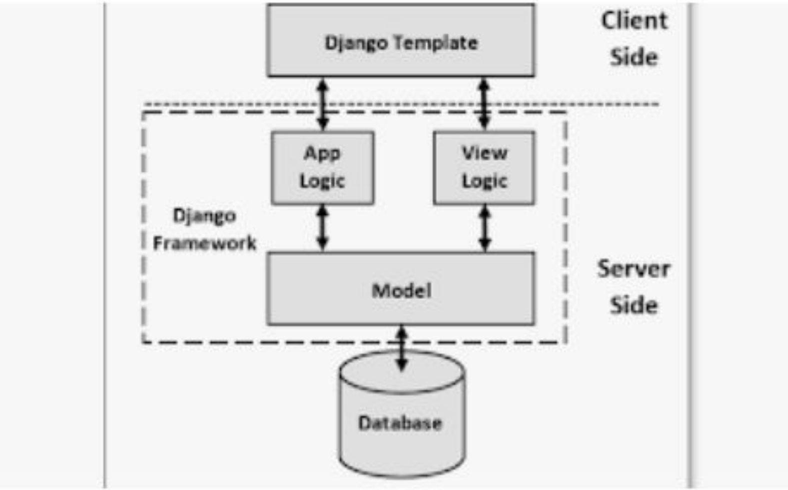
- View of the Moon from the perspective of a point on the surface of the Earth.
- Necessary intermediate step for image registration.
- 3D model is a representation of the LRO WAC Mosaic map from JPL’s Moon Trek portal.

3D Model Progress

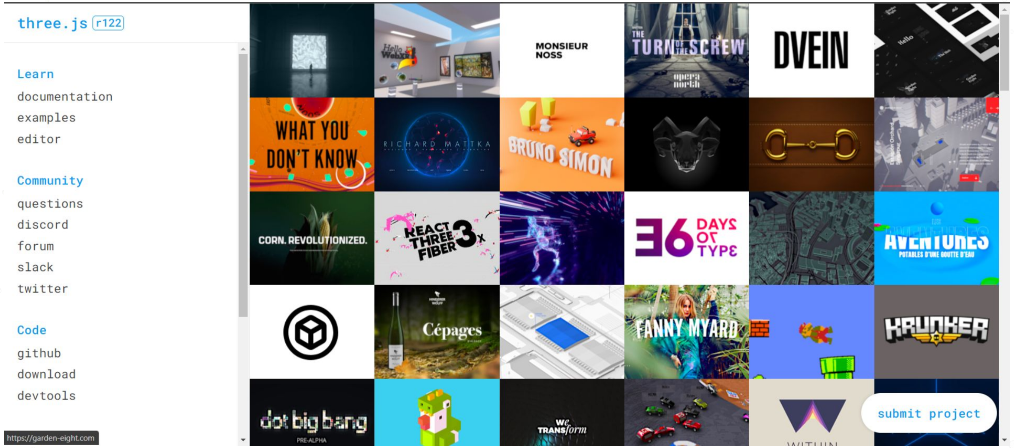
- Currently a view of 3 spheres that are the Earth, Moon and Sun.
- Lighting of the Moon dependent on the Sun.
- Working towards appropriate scaling of each sphere.



3D Model Technical Challenges

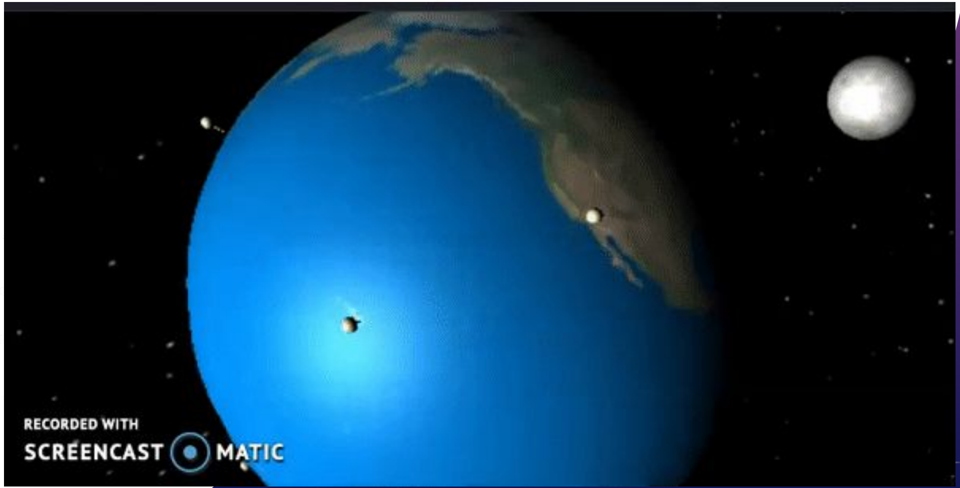


3D Model Technical Challenges



three.js

3D Model Technical Challenges



BACK-END

Pillow: EXIF Data Extraction

```python
def extractCoordinates(img_file):
    image = Image.open(img_file)
    exif = {}
    latitude = {}
    longitude = {}
    coordinates = {}
    img_exif = image.getexif()
    if img_exif:
        for tag, value in image._getexif().items():
            if tag in TAGS:
                exif[TAGS[tag]] = value

        if 'GPSInfo' not in exif:
            print('Your file does not have GPSInfo. Please upload a photo with the appropriate metadata.')
            # Instead of exiting out, can work to ask user for different file name instead

        if 'GPSInfo' in exif:
            latitude = str(
                float((exif['GPSInfo'][2][0]) + ((exif['GPSInfo'][2][1]) / 60) + ((exif['GPSInfo'][2][2]) / 3600)))

            longitude = str(
                float((exif['GPSInfo'][4][0]) + ((exif['GPSInfo'][4][1]) / 60) + ((exif['GPSInfo'][4][2]) / 3600)))

            coordinates = (latitude + exif['GPSInfo'][1] + ", " + longitude + exif['GPSInfo'][3])

    return(coordinates)
```

- Python Imaging Library (PIL, aka Pillow) carries the bulk of our EXIF data manipulation.
- Pillow's "Image" and "ExifTags" modules allow us to access an image's EXIF data.
- EXIF data contains information vital to MoonTrek's operation:
    - Time
    - Longitude
    - Latitude

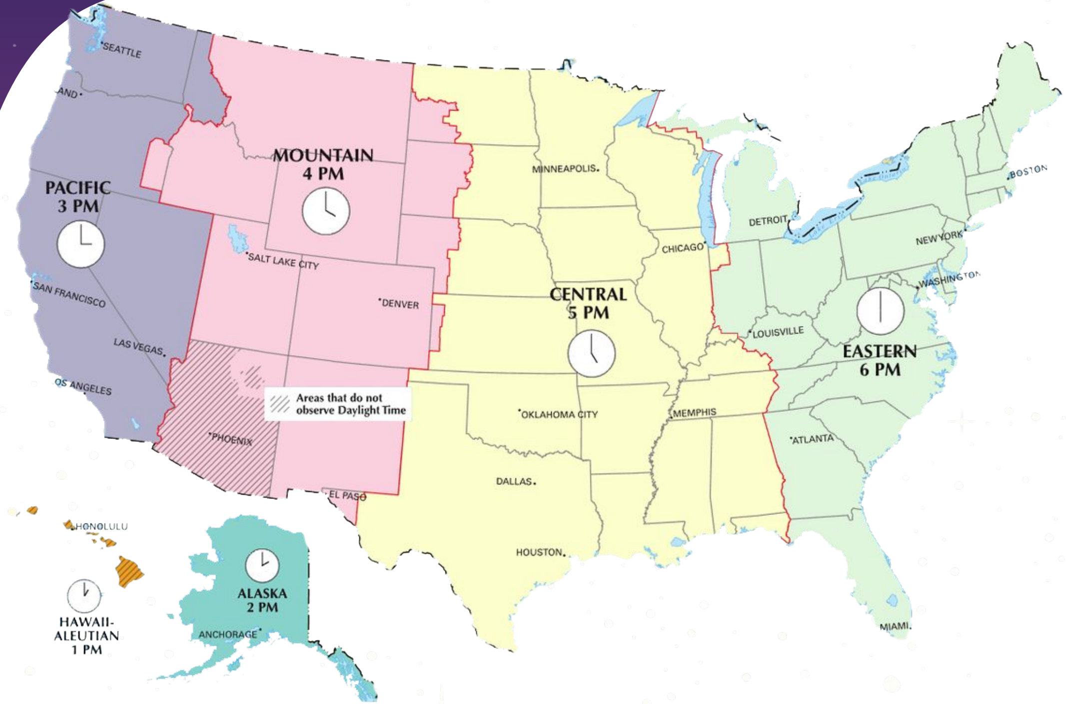
GeoPy

Coordinate Locator

```
Alberts-MBP:ExifExtract Albert$ python3 addressFinder.py
Address for file: PracticeImage.JPG:
1337, La Brea Avenue, Mid-Wilshire, Los Angeles, Los Angeles County
, California, 90019, United States of America
Alberts-MBP:ExifExtract Albert$
```

- GeoPy Library is responsible for taking Pillow’s coordinates and turning them into a location.
- Our application takes a latitude and longitude float, parses them together as a string labeled “coordinates”, and makes use of GeoPy’s “reverse()” function.
- Reversing a pair of coordinates, we can find an image’s address.

Back-end | Technical Challenges

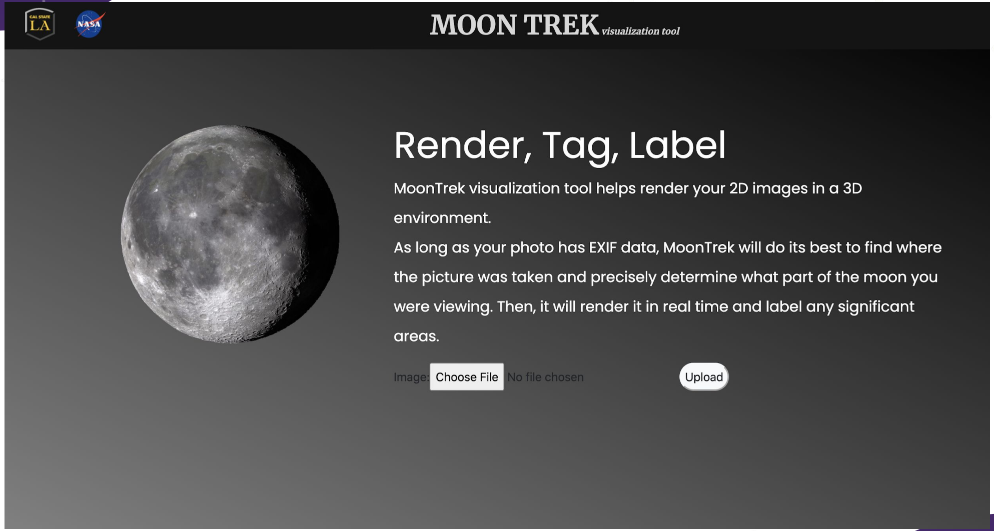


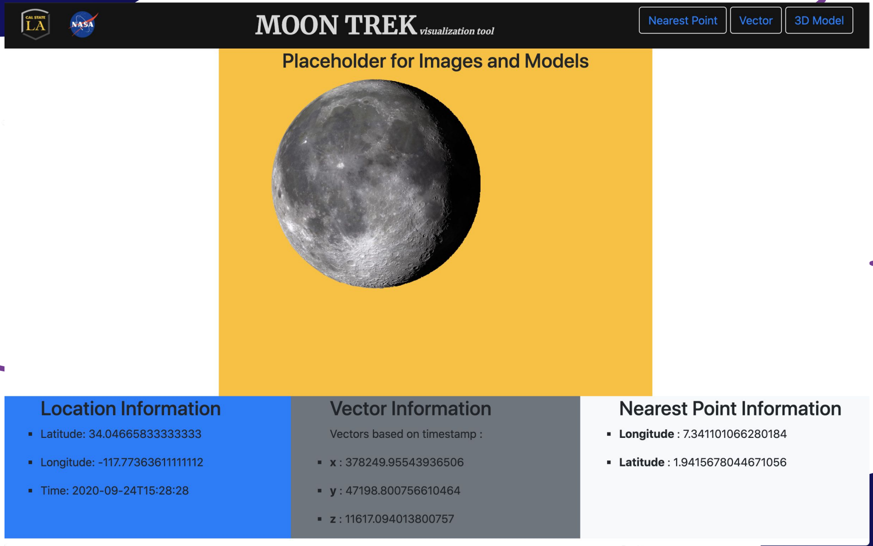
- Getting Geopy to work for all users
- Finding the time that accurately reflects the location of the photo
- Possible Solution: Converting EXIF data's time tags into UTC time
- This approach is difficult because EXIF data only carries time, no information for timezone (Pacific, Mountain, etc..)

UI & User Experience

- Crowdsourcing
- Location and time of the images

Front-End Development



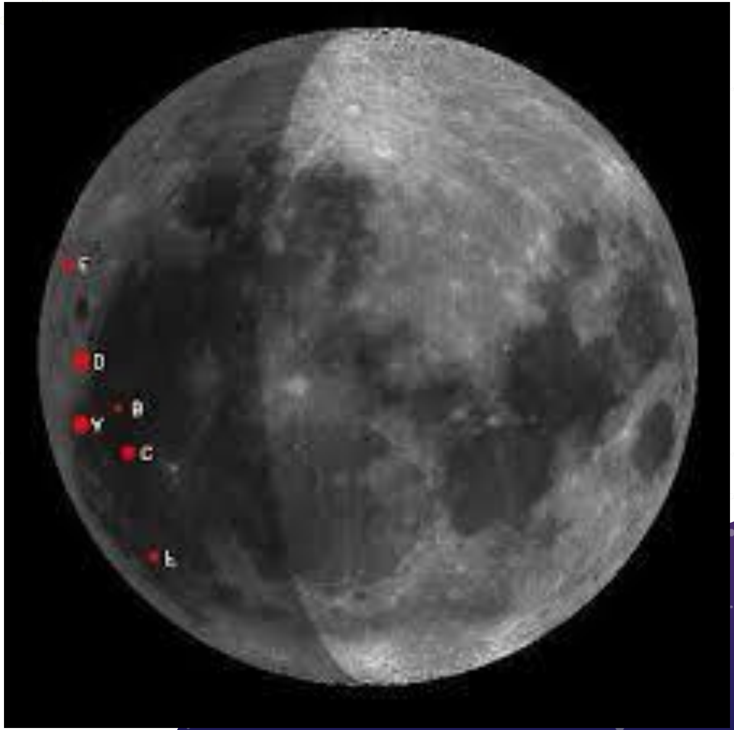

MOON TREK visualization tool
Nearest Point
Vector
3D Model
Placeholder for Images and Models
Location Information
Latitude: 34.04665833333333
Longitude: -117.77363611111112
Time: 2020-09-24T15:28:28
Vector Information
Vectors based on timestamp :
x : 378249.95543936506
y : 47198.800756610464
z : 11617.094013800757
Nearest Point Information
Longitude : 7.341101066280184
Latitude : 1.9415678044671056

Challenges

- COVID-19
- Integrate support for telescopic images
- Add security to the website

Future Plans

- Continue to hone and develop our skill with Django, OpenCV, and ThreeJS
- Produce context aware reference images
- Produce downsampled map/globe - matching with user provided image



Future Plans

- Improving 3D models
- Create context aware image models
- Generate correct view of model given time and geolocation

- Research and analyze other applications
- Design user friendly, efficient, and intuitive interface
- Connect everything together with the telescope!

THANK YOU