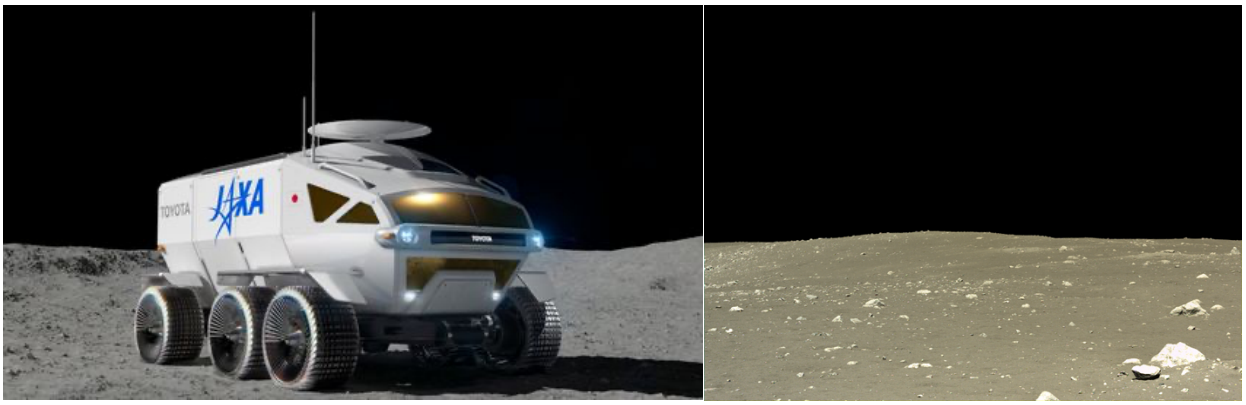
# In-class Programming Assignment #1 – Image Processing

**Overview:**
In this lab, we will process a series of images using algorithms such as thresholding and connected components to perform obstacle detection for a planetary rover. To complete the assignment you must submit via Canvas a screenshot / image of the final result you obtained after running your code.




**Scenario**

As part of the Artemis program, NASA is planning to send its first mobile robot to the Moon in late 2023 in search of ice and other resources on and below the lunar surface. Data from the Volatiles Investigating Polar Exploration Rover, or VIPER, would help the agency map resources at the lunar South Pole that could one day be harvested for long-term human exploration at the Moon.

As a robotics engineer, you are assigned the task of developing code that can help the lunar rover navigate around the surface of the moon while avoiding obstacles in the form of lunar rocks. Given an input image of the lunar surface captured by the on-board camera, your code will need to compute (i) the image pixels belonging to rocks and (ii) the centroid position (in image coordinates) of each individual large rock.

**Setup**
You will need the following Python libraries installed in your environment:

- Numpy https://numpy.org/
- Matplotlib https://matplotlib.org/stable/users/installing.html
- OpenCV https://pypi.org/project/opencv-python/

## 1. Thresholding

The first step is to assign each pixel in the input image as either *rock* or *not-rock*. We can make use of the property that rocks appear at a higher intensity in the image. Thresholding refers to the

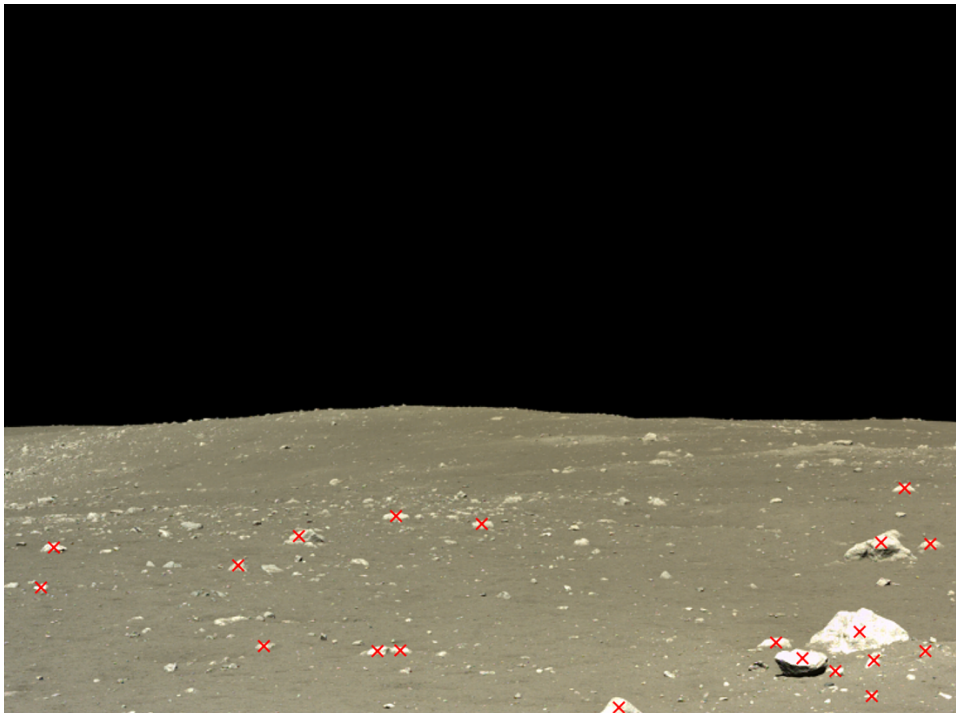
process of creating a binary image by applying a threshold on the pixel intensity such that pixels that fall above the threshold are assigned a value of 1 whereas pixels that fall below the threshold are assigned a value of 0. Refer to this link for more information about thresholding https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html

## 2. Connected Components

The next step is to group the rock pixels together to obtain individual rocks. The connected components algorithm can be used to find neighboring pixels and connect them together to form individual components. The connected components algorithm implemented in OpenCV will output a label image, where each pixel is given an integer label ranging from 0 to N (N is the number of components). Refer to this link for more information about connected components https://www.pyimagesearch.com/2021/02/22/opencv-connected-component-labeling-and-analysis/

### 3. Centroid Computation

Finally, we need to compute the centroid position (in image coordinates) of each individual large rock. By iterating through each component (obtained in the previous step), we can determine the image coordinates of the pixels that belong in that component and compute the centroid position by taking the mean over the image coordinates. To overcome noise and erroneous detections in the data, we would also need to sort the components by size (number of pixels) and only use components that are large enough (i.e. belonging to large rocks).



```
Centroid 1 x = 719.8928571428571 y = 390.57142857142856
Centroid 2 x = 317.0147058823529 y = 412.80882352941177
Centroid 3 x = 240.0 y = 428.6951219512195
Centroid 4 x = 701.5972222222222 y = 433.63194444444446
Centroid 5 x = 45.815384615384616 y = 437.3076923076923
Centroid 6 x = 684.6148648648649 y = 504.5251842751843
Centroid 7 x = 618.2049180327868 y = 513.016393442623
Centroid 8 x = 302.60869565217394 y = 519.6086956521739
Centroid 9 x = 320.42857142857144 y = 518.8214285714286
Centroid 10 x = 639.0253807106599 y = 525.0558375634517
Centroid 11 x = 695.561797752809 y = 526.7303370786517
Centroid 12 x = 493.94594594594594 y = 564.5405405405405
```