# Usage

To clone this project use the following command

git clone <https://github.com/jeccec51/PNoise_Filter.git>

The OpenCV headers and dlls are provided along with this project. If you want to change the OpenCV version, you may need to build OpenCV and then change the linker settings to accommodate OpenCV libs and dlls.

This project is an introductory example, to demonstrate how OpenCV libraries can be employed to remove periodic noise pattern from images. This project is developed in windows platform, with visual studio 2019 community addition. This is developed as a shared DLL.

To need to include PeriodicNoiseFilter.h in your main project, and then you can use the class BandReject\_Noise\_Filter I your code.

This class has two options to provide the input image.

1. Image path can be provided, the constructor will read the image and store in its member variables after pre-processing
2. Can provide the floating-point image buffer, along with its metadata like height and width. Constructor will load the data, and store in its member variables after proper usage.
3. The Filter\_Periodic\_Noise() function will perform the noise filtering and store the output in its member variables
4. The GetOutPutImage() function will return the filtered image. You can use this in your project either to process further or to displa/store.

# Theory

The periodic noise is an undesirable repetitive pattern present in images. One such image is given below.

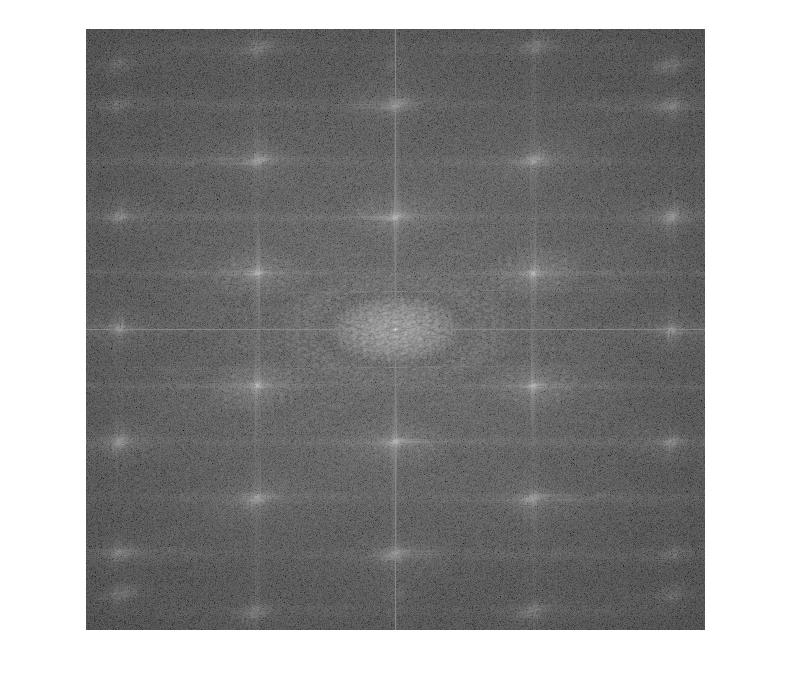


This is a moon landing picture corrupted with periodic noise.

It can be observed that the spectrum of such images contains mainly 2 parts.

1. The periodically repeating peaks corresponding to periodic Noise
2. Low frequency information contains image details

The core idea here is to preserve the low frequency content of the image, while filtering out the high frequency part.



In this project the central low frequency area will be preserved, while the other responses will be filtered out.

Compute2DSpectrum() function computes the magnitude spectrum of the image. The spectrum will be stored in m\_matMAgnitudeSpectrum. It scomputes the PSD as well for better visualization.

LowPassFilter() function adaptively generates an Lpf mask to filter out high frequency components. It first thresholds the magnitude spectrum using OTSU thresholding. Later, a circular mask is adaptively identified, by sequentially increasing its radius form 1 to cut off radius. After he cut off radius it can be observed that the histogram of the circular portion will be drastically changing. This is the cut-off point, and can be easily identified by plotting the underlying histogram for every increased radius. A sample low frequency mask development is shown in figure.

Iteration 1 Iteration 2 Iteration 3

In the next step, the mask will be applied on to the magnitude spectrum. The resulting spectrum will be used to calculate the inverse FFT and then store the results.