Lab 2: Image corrections

Masking and bad pixels (30 points possible)

A PDF of the report for this lab must be uploaded to Gradescope by 10 am on Thursday, February 3 2022 (check Brightspace for updates on due date). See Lab 1 instructions for submission details. There are two key documents for this lab - this document (lab02-instructions.pdf) gives instructions for the lab, but make sure to also fill out the lab report (lab02-report.doc). You should open and edit this file, and write your answers in all the indicated areas.

Setup

(1) Make sure to create a ***working directory*** for the lab called lab02 in your home directory (you may want to put this in an EAPS577 folder).
(2) Download all of the files for the lab from this folder into your working directory (*it's best to unzip the file in your downloads folder and then copy to your working directory*): https://www.dropbox.com/sh/owvccixozhgxqov/AADYyao4OEg4oht0wsfVwMsxa?dl=0
(3) Next, we want to set up your IDL "path". Go to Window-> Preferences-> IDL -> Paths. Click Insert, and then select your working directory.
(4) Anytime you are working in IDL, you should start a journal file, as discussed in the tutorial. You should create a new journal file every time you restart IDL.

Key Concepts in this Lab:

- Images are just 2D arrays of values – learning to create and manipulate them
- The all important "where" function and selecting portions of images
- Creating and using masks to modify images
- Identifying bad pixels and removing them

1. Histograms

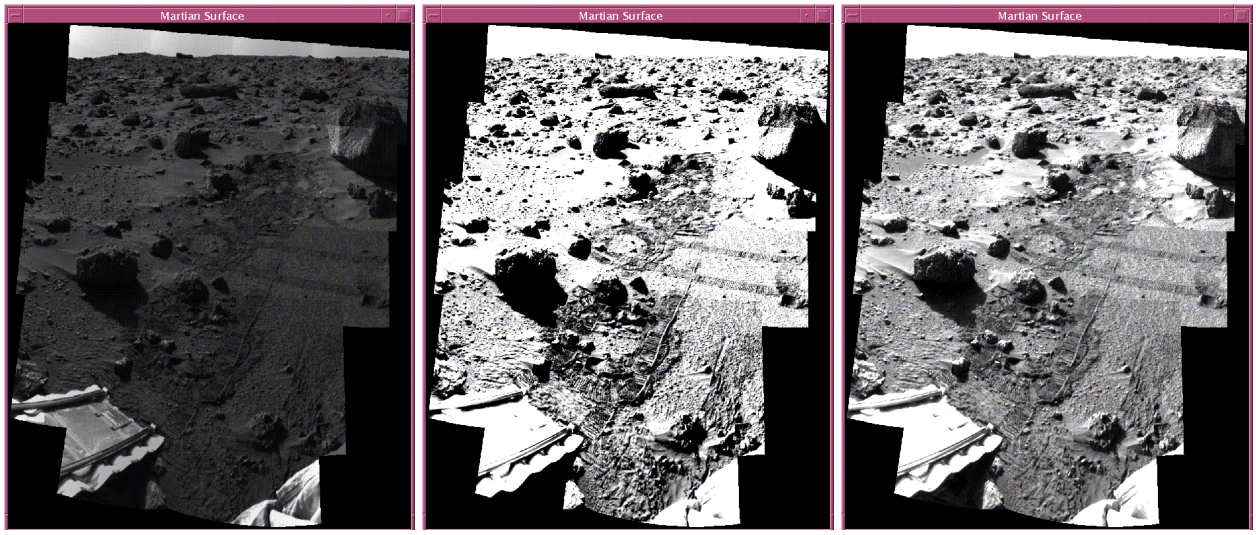

Fig. 1 *– Ares Valles as viewed by Mars Pathfinder (av_img), with tracks from the Sojourner rover visible in the foreground. (left) Standard linear stretch. (middle) An example of a harsh stretch. (right) An example of an image scaled using a histogram.*

There is an image mosaic in the lab02 directory under the filename **imp.sav** that was taken by the Imager for Mars Pathfinder (IMP) on the Pathfinder lander in an old flood channel, Ares Vallis, on the martian surface (see Fig 2). Restore the data and check out the array:

```
IDL> restore,'imp.sav'
IDL> help, av_img
```

The output from the last command tells you that the *av_img* array is a 612 x 768 floating point array. You can use the **max** and **min** functions to determine the range of the data values.

```
IDL> print, min(av_img), max(av_img)
```

Now look at the image in atv2, with the default linear stretch. *(Note: Make sure your IDL path includes the directory with atv2 in it – the easiest way to do this is by making your class directory part of your path, with folders for both labs 1 and 2 in there)*

```
IDL> atv2, av_img
```

The hardware related to the Mars Pathfinder lander and Sojourner rover in the lower area of the image is neat, but if we are interested in the Martian surface, this image appears too dark. As above, we can try stretching the range of data values so that we can see the Martian surface more clearly. How much should we narrow the range? Let's make a guess and try changing the min to 0.0 and the max to 0.15.

It is brighter, but if we want to be more precise, we don't have to guess! Atv2 provides a way to interactively see the values of the image. The box below the min and max entry fields displays the x- position, y-position, and pixel value at the cursor as we move the cursor on the image. You'll notice that the Sojourner rover tracks tend to be ~0.03, so try a "harsh stretch" of 0.03-0.05. The result is shown in Figure 1 (middle). Notice how the track marks are accentuated.

Often even better is to make a histogram of the original image because this tells you the distribution of pixel values. We can do this using the plothist function in IDL. Back at the IDL command line, enter:

```
IDL> window,0
IDL> plothist,av_img,bin=0.001
```

You can change the binsize to change the "resolution" if you like. Notice that this shows a large spike at zero corresponding to the edge of the image, you can cut this out using the yrange or yr option – for example:

```
IDL> plothist,av_img,bin=0.001,yr=[0,1e4]
```

The easiest way to save this plot is by taking a screenshot. To do this, I recommend the Snipping Tool in Windows to grab an image of just the plot window. See this tutorial for opening and using the tool: https://support.microsoft.com/en-us/help/13776/windows-10-use-snipping-tool-to-capture-screenshots

Save the screenshot of the histogram in your lab02 working directory as "imp_histogram.png" and paste or insert this image into your lab report (Question #1a in the Lab Report).

Using your histogram, see if you can determine the range of values that corresponds to the black background, the shadows, and the surface. (Question #1b in the Lab Report).

Using the range that you think makes sense for the shadows and surface, pick a new range for your image to emphasize the variability on the ground surface. It should look like Figure 1 (right). Save your image in ATV2 as "imp_stretched.tif" in your lab02 folder and paste or insert this image into your lab report. (Question #1c in the Lab Report).

The last portion of the image that may not be obvious from this histogram is the sky, so let's plot just that part of the image. Because images are just arrays, we can select out chunks of them by indexing the arrays:

- av_img[x1:x2,y1:y2] selects out a box within av_img that includes columns x1 to x2 and rows y1 to y2
- av_img[x1:x2,*] uses the "wildcard" * to indicate all rows, so this selects a narrow vertical column from the image between columns x1 and x2

So for example, **`atv2,av_img[0:100,0:100]`** will just display the lower left portion of the image. *Note: If you forget which direction (rows vs. columns) the first and second entry in the array notation indicates (I often do, and different languages flip it around), you can open up an image in ATV2 and watch the numbers in the cursor location box change as you move your mouse over the image.*

Make a new histogram that just includes data from a cropped portion of the image that is mostly sky. You can use ATV2 to help figure out the limits on the crop. Take a screenshot of your new histogram, save it as "imp_sky_histogram.png", and paste it into your lab report (Question #1d in the Lab Report).

Based on your histogram, what is the approximate range of values in the sky? (Question #1e in the Lab Report).

2. Masking: Part I

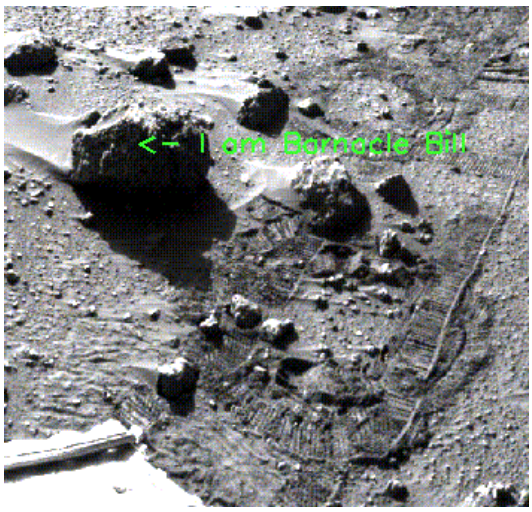

Fig. 2 *– Barnacle Bill! (Did he live under a martian sea? It's possible!)*

Masking refers to the "blacking out" of different parts of the image to focus on specific areas or parts of the image that meet certain requirements. Back in the day this would have been literally done on a physical image with black masking tape (a handy tool for propaganda – the original Photoshop!). Suppose we are interested in only a portion of the image - say, the rock named Barnacle Bill, shown in Figure 2. In this section, you'll figure out: what lies in the deepest darkest parts of his shadow?

To look at just Barnacle Bill, one approach is to mask out everything in the image that isn't near Barnacle Bill. Create a mask that is the same size as the image:

```
IDL> mask = fltarr(612, 768)
```

This will initially be an empty array with every pixel equal to zero.

Now, use ATV2 and your cursor to find the x and y limits of the area right around Barnacle Bill. Set the portion of the mask inside of these coordinates to 1:

```
IDL> mask[90:220, 290:440] = 1.
```

Check to see what the mask looks like – it should look like a window:

```
IDL> atv2, mask
```

Now, what will happen if we multiply this mask by the original image? Try this in atv2:

```
IDL> atv2, av_img*mask
```

Save your masked image as "imp_masked.tif" and paste it into your lab report. (Question #2a in the Lab Report).

Plot a histogram of the masked image (make sure to zoom in on the data on the y-axis), and use it to find the limits that you should use to illuminate the shadowed region behind Barnacle Bill. Save a screenshot of your histogram as **`'imp_bb_histogram.png'.`** and put it in your lab report (Question #2b in the Lab Report). What limits will you use to illuminate the shadowed region and why? (Question #2c in the Lab Report).

Make the stretched image with these values in atv2, and save the stretched image showing the illuminated shadow as **`'imp_bb.tif'`**. (Question #2d in the Lab Report). What did you find in Barnacle Bill's shadow? (Question #2e in the Lab Report).

3. Masking: Part II

Masks can also be used to remove parts of an image based on their characteristics. For example, let's say that you would like to do a study of the opaqueness of the sky as a function of distance along the horizon. This would be easiest if you could extract the sky from the image. Alternatively, you can just mask out everything in the image that isn't sky.

Above you cropped the image to show mostly sky, but now we'll use a mask to truly remove all pixels showing the ground.

To do this, we'll need to use the all important **`where`** function in IDL. To practice using where, use it along with **`n_elements`** to figure out how many valid pixels (i.e., not black

background) are in the image (Question #3a in the Lab Report).

```
IDL> print, n_elements(where(av_img gt 0.))
```

If we wanted to create a mask that only selected the valid portions of the image, we can follow a similar approach to the above "window" mask. However, instead of selecting a specific x/y range to mask out, we can use the where function to select only valid pixels:

```
IDL> mask = fltarr(612, 768)
IDL> mask[where(av_img gt 0.)]=1.
IDL> atv2, mask
```

This should look like a white out version of the image.

Back to the sky. You should know what the range of values is for the sky from Question #2, so use this knowledge to create a mask that only selects the sky in the image. How did you isolate the sky? Note that you will have to remove the other bright objects at the bottom of the scene to make a true sky mask, how did you do this? (Questions #3b-c in the Lab Report)

Using your mask, determine how many pixels represent the sky in this image. What percentage of the image is the sky? (Questions #3d-e in the Lab Report)

Display your masked image in atv2, and use your range of values for the sky to create a stretch that emphasizes the variability in the sky. Save your stretched and masked image as a tif file in atv2, with the filename `'imp_sky.tif'`, and paste or insert this image into your lab report. (Question #3f in the Lab Report)

4. Bad Pixel Correction

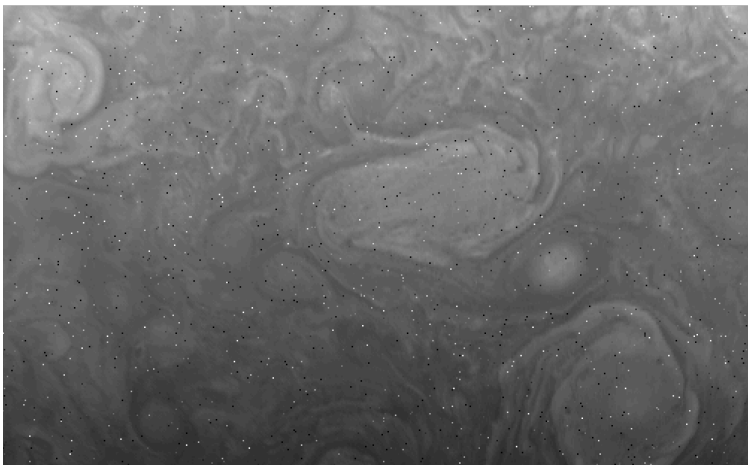

Figure 3: *Gross noise added to the red channel of a Junocam image of the northern circumpolar cyclones on Jupiter, save the cyclones!!*

The other data file in the lab02 folder is "juno_red.sav". This is an image taken by the Juno spacecraft of Jupiter, and I've added some gross noise to it, as shown in Figure 3. Our goal in this section is to fix this lovely image by correcting the bad pixels.

The picture on the right shows an example of a "bad" pixel. This is usually defined as a value that is some number of standard deviations above or below its neighbors. The juno_red image has one really bad pixel with a value great than 10 (most pixels in the raw image are between 0-1). You should be able to find it using atv.

65	48	71
60	3500	68
55	73	65

This "hot" pixel might either be an overly-responsive pixel (potentially fixable if we understand the behavior), or it might be a random event like a cosmic ray (not fixable). For now, let's assume that it's not fixable, so let's replace it with a more reasonable value. A common strategy is to replace the bad pixel with the median of its neighbors.

IDL has a built-in median function, where it calculates the local median within a given neighborhood around each point in the image. The smallest neighborhood for the median filter has a width of 3, one pixel on either side and above and below of the middle. Try applying it to juno_red:

```
IDL> juno_med=median(juno_red, 3)
```

How does the median image look different from the original? Try changing the width of the median filter (note that this has to be an odd number), how does that affect the image? (Question #4a)

We want to replace the really bad pixel above with a better value, which we can take directly from the median image. The median image is a reasonable guess for what that pixel should be. We can use the where command to find the location of the bad pixel in the original image:

```
IDL> bad_pixel=where(juno_red gt 10)
```

If you print out this variable (**`print,bad_pixel`**), you can see that it's just one location (the one dimensional index of that location in the array). So now we can replace the pixel at that location in the original image with the value at that same location in the median image:

```
IDL> juno_red[bad_pixel]=juno_med[bad_pixel]
```

If you look at juno_red now, the one really bad pixel should look like it belongs.

The next step is to use the same process to replace the less obviously bad pixels with their value in the median image. But how do we determine what constitutes a bad pixel? There are two key steps:

(1) We need a reference for what a good pixel should be. For this, we can use the median image.
(2) We need to determine what "bad" means. All of the pixels in the original image will be at least a little different from the median image, so how far away from the equivalent pixel in the median image is "bad"?

So the first thing we need to do is to compare the image to the median image. To do this, create an array that is the difference between the median image and the real image. Since we don't care about the sign of this difference, make it the absolute value (abs function). Take a look at this difference image in atv2.

This difference image is telling us about how "bad" each pixel is, and now we need to decide at what values we will replace those pixels. Are there statistical measures that you can use to determine what values are way outside of normal (e.g., *standard deviation*)? Think about this and then write down *in words* your definition of a bad pixel. Will this definition work for most images, or is it unique to this image? (Question #4b)

Now use that definition to create an array that stores the locations of the bad pixels:

```
IDL> bad_pixels=where(definition)
```

How many "bad" pixels did you find in each of the six raw images? For reference, I replaced about 1% of the pixels with noise. How does your number of bad pixels compare to this? (Question #4c)

Finally, create a new array that is a copy of the original image that will have the correction applied, and replace the bad pixels in that array:

```
IDL> juno_nobad=juno_red
IDL> juno_nobad[bad_pixels]=juno_med[bad_pixels]
```

Take a look at your new corrected image in ATV2, how did you do? Did you remove all of the obvious bright *and* dark bad pixels? Compare to the original image - do you see any evidence that you "over-correct" in some way (replaced apparently good pixels)? (Question #4d) Zoom in on the middle of the corrected image in ATV, save this view as "juno_red_no_bad_pixels.tif", and paste the image into your lab report (this should look something like Figure 3, but hopefully without the noise). (Question #4e)

5. Turn it into a program!

To complete this section, you'll now take these commands and turn them into a generic *function* called **replace_bad_pixels.pro** that you could run on any (single band) image, just like we did for the true/false color images in Lab 1. This will be a function rather than a program because it will return the corrected image into a new variable. You can start by editing the "replace_bad_pixels_template.pro" file in the lab directory.

This program should follow the steps you took above (but feel free to modify!), which were:

(1) calculate the median image from the raw image
(2) calculate a difference image
(3) determine what the cutoff for "bad" is based on the difference image
(4) use "where" to create an array with the locations of the bad pixels
(5) print to the screen how many bad pixels were found
(6) create a new image that is initially a copy of the old image (e.g., **img_nobad**)
(7) replace the bad pixels in this image with values from the median image
(8) return the corrected image

When you have working code, please copy and paste it into your lab report. (Question #5a), then save and paste the resulting corrected image into your report as well (Question #5b).