Ashton Gray

32000589

ECE 597IP Project 1

Mean Shift

Background:

Mean-Shift is an algorithm that calculates the density gradient of an image using all pixel feature vectors of an image. Each pixel feature vector is the information stored about a pixel. This can be RGB values, HSI values, or even can include the location of the pixel in the original image. Then with this information, the algorithm finds the local maximums to find the centers clusters. This is done by, for example, starting at a random point and using a kernel centered at that point to find the center of mass in that region. This is done recursively, which creates a trail towards the final local maximum. With this being done all around the density graph, multiple local maximums can be found, and therefore all points linked to a local maximum will be assigned into the same cluster.

Pros: Unlike K-means, this algorithm can be good for elongated density patterns, where you can see clusters of density, however they begin to merge together. Also, with mean shift, unlike K-means you do not need to give a specified K value for the number of clusters, instead the algorithm will automatically detect this.

Cons: However, there are still cons to this algorithm. One being, is if the size of the kernel being used is too small, holes can be created. This is because when the kernel is centered on a sample, and the size of the kernel is so small it does not contain other samples, and therefore the algorithm will think that the single sample is a local maximum and therefore its own cluster.

Though, if the kernel is too large, this can have the opposite effect. This would then take too large of an area and may combine multiple local maximums into a single local maximum, therefore combining clusters.

SciKit-Learn: Meanshift

The SciKit-Learn package in python contains the function MeanShift to perform this clustering. This function performs the meanshift algorithm mentioned above on an inputted 2D feature vector. The other parameters of the function are:

Bandwidth – the bandwidth of the kernel, which is how closely the density should match to the distribution. The higher the bandwidth, the less clusters that the algorithm will create, and the lower the bandwidth, the more clusters that will be found.

Min_bin_freq – To speed up the algorithm, I used a value of 1000, meaning only bins with 1000 points are used as seeds. This means that the starting points must have at least 1000 surrounding points. This is to also help against holes.

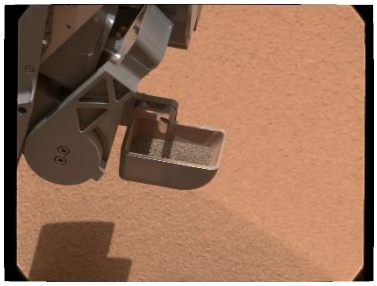
Bin_seeding – To also speed up the algorithm, this was set equal to True. This means That the starting points are not every point, instead are just a discretized version of all the points, where the size between points are correlated to the size of the bandwidth.

Varieties to Mean-Shift:

To try and get the best results of meanshift, we can try apply different transformations to the image. This will impact how effective the meanshift algorithm will be. Here, we can use some transformations such as blurring, HSI, and decorrelation stretching.

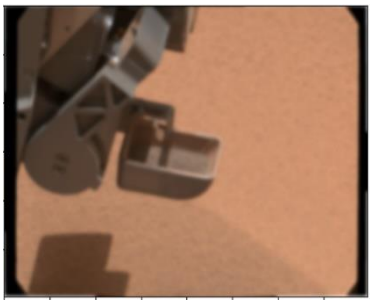
Image 1: Image 2: Image 3: Image 4:

Original images RGB:



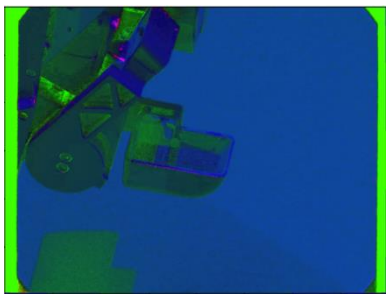
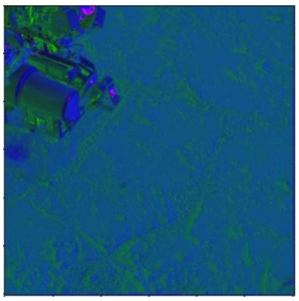
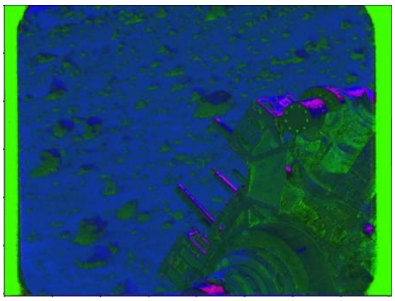
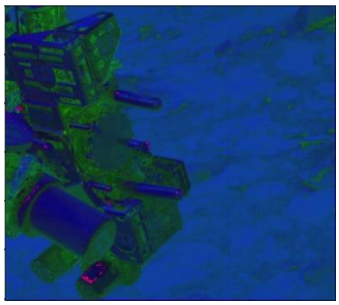

Here are the original images in the RGB color range, without any transformations done to them.

Blurred images:




Here are the original images in RGB, but instead a gaussian filter with a sigma of size 7 was used to blur the photo. As you can see this is useful because it ends up averaging surrounding pixels. This will be useful to the mean-shift algorithm, because the averaged pixels will be easier to differentiate if pixels are actually in a cluster or not.

Images in HSI:

These are the original images, but in in HSI (Hue, Saturation, and Intensity) instead of RGB (Red, Green, and Blue). These were calculated using the equations:

$$I = \frac{1}{3}(R + G + B)$$

$$S = 1 - \frac{3}{(R + G + B)}[\min(R, G, B)]$$

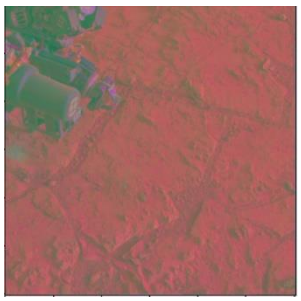
if $B \leq G$

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}}\right]$$

else, $H = 360 - H$

This will also be useful because as you can see visually, the difference in rover and surrounding mars soil is much higher. However, you can see in image 1, parts of the arm of the rover are still very similar to the surrounding soil, as well as for image 3, parts of the shadows of the rocks are very similar to the rover.

Decorrelation Stretching:

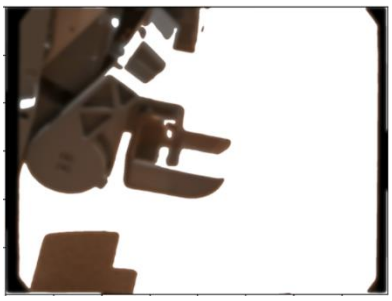
Here are the images after using decorrelation stretching on them. Decorrelation stretching is rotating and stretching the color inputted into an image, therefore decorrelating it. As you can see, this greatly helped differentiate the rover and surrounding soil, however in image 1, it did does not differentiate the rover from the shadow very well.

These are some of the different transformations we can do to the image before applying the mean-shift algorithm in order to help segmentation. In each of these examples, the feature vector for each pixel is still only 3 values. However, we can go even further by adding the pixels x and y location in the image to these vectors.

Here we can see the effect of adding pixel location in the feature vector, with image 1 being blurred, and bandwidth = 0.2.

Image Blurred/Bandwidth=0.15:

Cluster 1



Cluster 2

Cluster 3

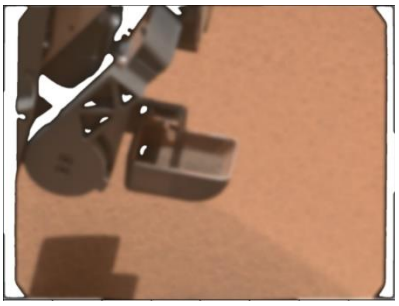
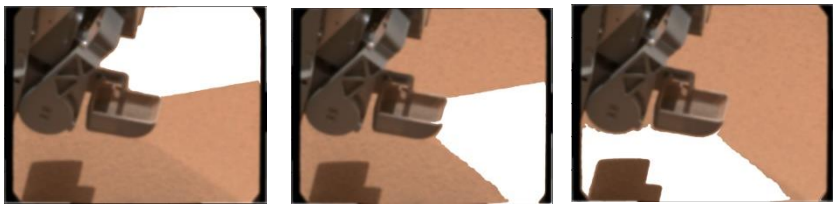
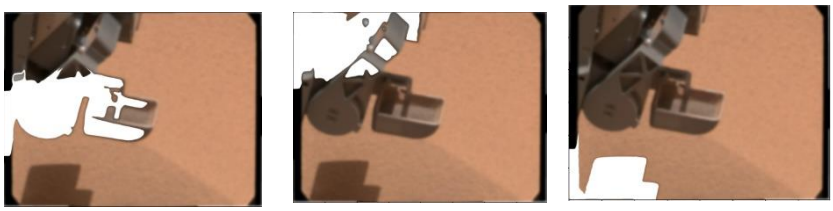


Image Blurred/Location Used/Bandwidth=0.15

Cluster 1 *Cluster 2* *Cluster 3*

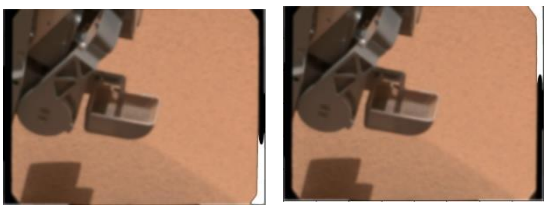


Cluster 4 *Cluster 5* *Cluster 6*



Cluster 7 *Cluster 8* *Cluster 9*

Cluster 10 *Cluster 11*



As you can see from above, using the location, while still using the same bandwidth, more segments are created, however, you can tell that the location was used because of how the clusters are only in the same areas. Such as being able to differentiate the rover from the shadow, and the inside of the scoop from the rest of the surrounding soil. An important thing to notice here, is even though there are many clusters, boundaries are not crossed. We can see that clusters 1, 2, and 3 make up the soil, clusters 4, 5, 7, and 8 make up the rover, and cluster 6 makes up the shadow. (9, 10, and 11 are just the outsides of the image)

This is where we can now use a combination of all the previously mentioned tools to find the best ways to segment the image. As you can tell with all of these options, as well as the ability to change the bandwidth, it makes it hard to find the exact parameters to separate the rover, shadow, and soil into three clusters.

Clustering:

Image 1

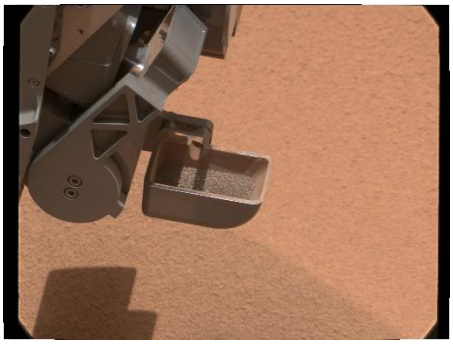
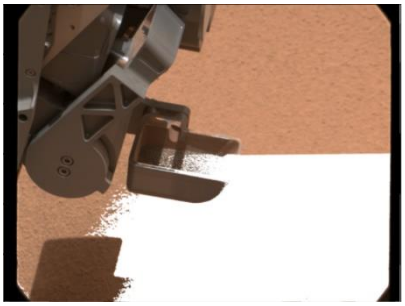


Image 1 was best separated using the example before, in most other combinations, issues were run into either with the shadow:

In this example, decorrelation and location was used, with a Bandwidth of 5. As you can see, the shadow is still put into the same cluster as parts of the rover.

Or with the scoop:



In this example, HSI and location was used, with a bandwidth of 0.3. As you can see, the inside of the scoop, that contains soil, was out into the same cluster as the surrounding soil.

In each case however, the more clusters gave a more accurate segmentation. Below is manually putting together the separate clusters shown above when blurring, location, and a bandwidth of 0.15 are used:

Clusters 1, 2, 3, and 6 Combined:

This forms the rover

Clusters 4, 5, 6, 7, 8 Combined:

This forms the terrain

Clusters 1, 2, 3, 4, 5, 7, 8 Combined:

This forms the shadow of the rover

Image 2

For Image 2, we can see from the HSI image shown earlier, that it would not be a good idea to segment using HSI. This is because the rover blends in a lot with the surrounding soil. Here is the segmentation if we did use HSI:

Cluster 1

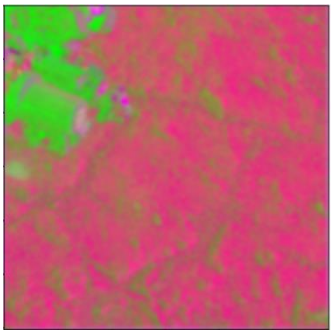
Cluster 2

HSI and location were used, with a bandwidth of 0.3

As you can see, the image was segmented into two parts across the diagonal of the image. This goes along with the HSI image where the rover and soil looked blended together. Therefore, when the mean is being calculated, the image will be split in two, with similar averages on both sides.

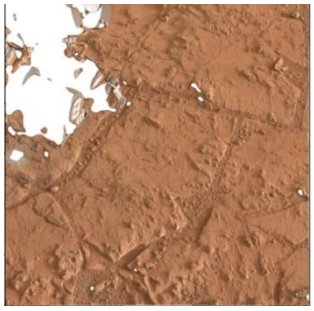
Looking at the decorrelation stretch image for image 2, we can see that the rover and soil look visually separate. Therefore, this would be the best method for clustering. There are also dark shadows in the soil after using the decorrelation stretch on the image, therefore blurring should also be used.

Here is image 2 after blurring and decorrelation stretch was used:



Here is image 2 segmented using blurring, decorrelation stretch, location, and bandwidth=40:

Cluster 1



Cluster 2

As you can see, this is not perfect, and the mean shift algorithm us unable to distinguish the part of the rover from the soil. This is a limitation of the mean shift algorithm, because if the colors of two objects are too similar, they will be put into the same cluster.

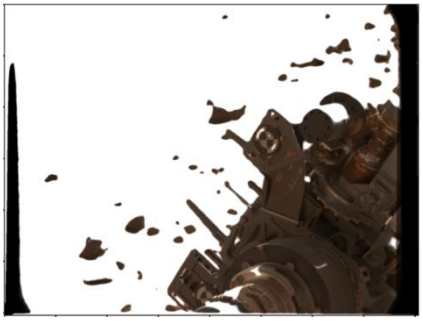
Image 3

Looking at image 3, we can tell that the decorrelation stretched image will prove to not be useful. This is because in this image, the rover is too similar to the surrounding soil and rocks. Therefore, we can try HSI and blurring.

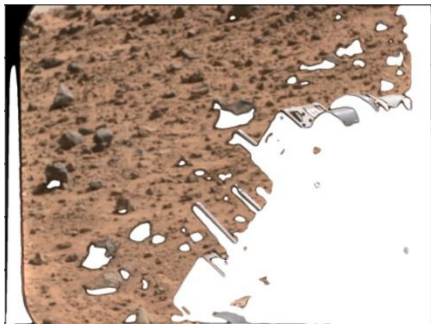
Using only HSI, we can see that the rocks and rover are both in the same cluster.

However, if we only use blurring in combination with location, and bandwidth=0.5 we find:

Cluster 1

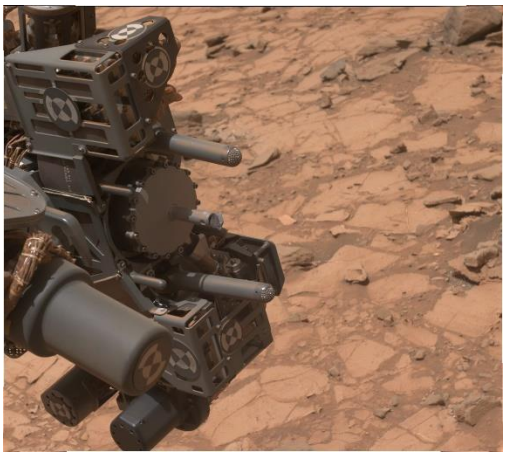


Cluster 2



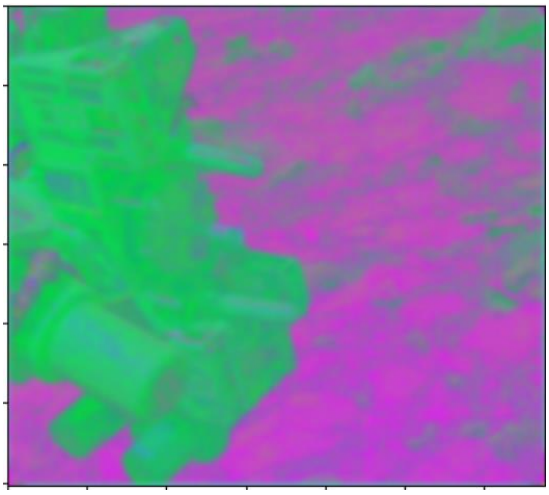
As you can see, the rover was segmented away from the surrounding terrain. However, only the rocks close to the rover were put into the same cluster as the rover. This is due to the location of pixels being used in the feature vector. This still is only a limited result. This is because the surrounding rocks are very similar to the rover, in both RGB and HSI, as well as when stretched.

Image 4



Looking at the transformed images above, we can see that both the blurred image and the decorrelated stretched image separate the rover from the terrain very well, and this will be useful for the algorithm.

Using decorrelation stretching, blurring, and location the image can be seen as:

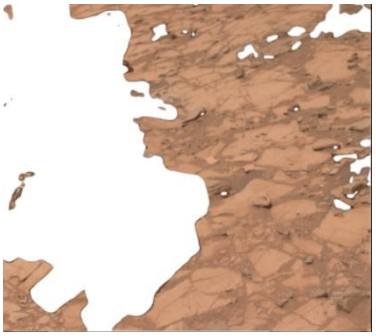


This image clearly separates the rover from the surrounding terrain.

Using the meanshift on this image, with a bandwidth=25, we get these two clusters:

Cluster 1

Cluster 2



These two clusters did remove the rover from the terrain; however, it is not perfect. To the right side of the image, the algorithm included rocks that look similar to the rover in the same cluster. This could be due to them looking similar in both RGB and even when decorrelated stretching was used.

Conclusion

Mean-shift is by no means perfect on its own. However, the image can be manipulated in order to help the algorithm. For example, it was found that when location was added into the feature vector, it had only helped the accuracy of the mean-shift. There was no situation when using the location, that the accuracy had gotten worse, therefore the location of each pixel was used in all feature vectors for this project. Along with this, the accuracy seemed to increase the more clusters that were found, and therefore the lower the bandwidth. The bandwidth could not go too low however, or else the issues of holes happened.

Another attribute of the algorithm was found, that the mean-shift is repelled by object edges. This is because of the recursiveness of finding local maxima. By going in the direction of local maxima, it is therefore going in the opposite direction of the norm of the gradient. The gradient is the directional change in intensity of color, and flows from the maxima in RGB (highest intensity values). Due to the fact

that the norm of the gradient flows towards edges, the mean-shift therefore flows away from edges and has anti-edge detection.

All of the functions shown have, for the most part, been able to separate the rover from surrounding terrain, and therefore mean-shift is a suitable segmentation technique.