Mars rover segmentation

The goal for this project was to segment the mars rover from the martian terrain in four images provided to us. A variety of methods were used to achieve the best possible segmentation for each image. For each method on each image, hyperparameters had to be fine-tuned in order to get a good segmentation, and their descriptions are detailed below:

K Means clustering: This is an unsupervised machine learning algorithm that groups similar data points together to discover any underlying patterns. This is done by a process called clustering, wherein a cluster refers to a collection of data points that are grouped together because they share certain similarities. The K means clustering function takes an integer as input which refers to the number of cluster centers that will be chosen (the chosen data points will become the center of the clusters, and more data points will be added to the cluster around the center) and these points are chosen by averaging the data, each data point will then be assigned to the cluster that is spatially the closest to the data point in question.

Mean shift clustering: This is an unsupervised machine learning algorithm that groups data points together to discover any underlying patterns. The algorithm works by assigning data points to clusters iteratively by moving them toward where the highest density of data points is in the region. This region, where most points are concentrated, will become the cluster center. This process works until each data point is assigned to a cluster. Mean shift does not require the number of clusters to be inputted as a parameter; the algorithm determines the number of clusters based on the data.

Decoloration stretching: Works on datasets with high correlation, which leads to images with bland color. Decoloration stretching removes high correlation in these kinds of images so that the resulting image is richer in color.

Linear stretching: Narrows the dynamic range of the image based on the intensities that are inputted. Pixels with intensities higher than the higher intensity inputted will be assigned an intensity value of 255. Similarly, pixels with intensities lower than the lowest intensity inputted will be assigned an intensity value of 0.

Histogram equalization: Increases the contrast in images by spreading out the most frequent intensity (stretching the intensity range of the image).

Felzenswalb's algorithm: Produces an over-segmentation of a multichannel (i.e., RGB) image using a fast, minimum spanning tree based clustering on the image grid. It has three parameters that need to be input: The number of produced segments, as well as their size, can only be controlled indirectly through scale, which is a free parameter, and the higher the scale, the more

the number of clusters. Sigma is the diameter of a Gaussian kernel used for smoothing the image prior to segmentation, and min_size is the minimum component size.

SLIC superpixel segmentation: Simple Linear Iterative Clustering produces an over-segmentation of an image using superpixels which are larger groups of pixels. These superpixels are often compact and uniform. This algorithm does not need much computational power and is very fast. The parameters that were changed for SLIC over-segmentation were the compactness parameter which trades off color-similarity and proximity and determines how "square" the superpixels are. The n_segments parameter approximately creates n superpixels on the image.

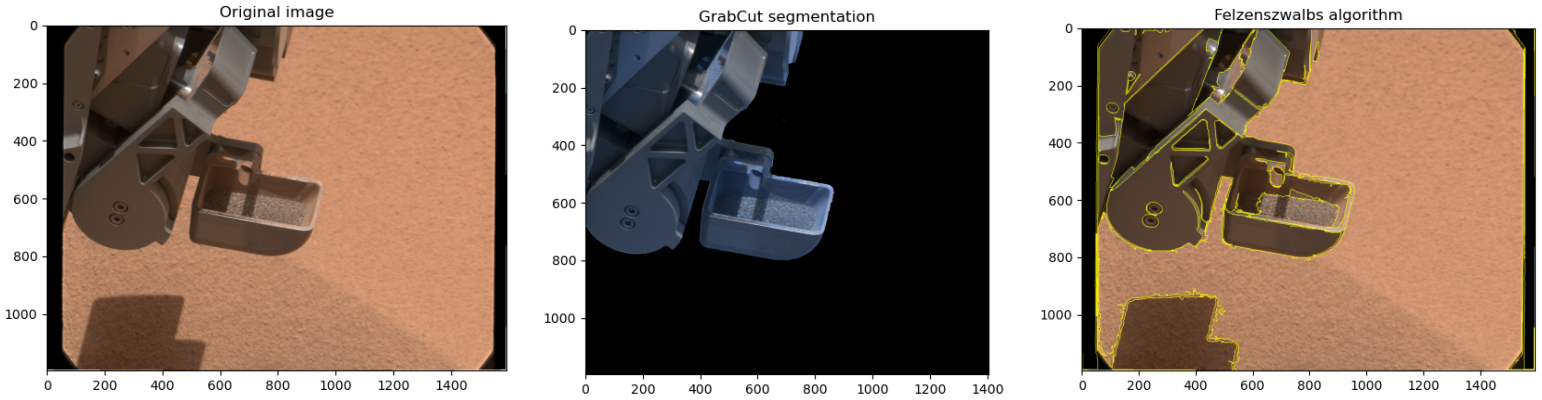
Region Adjacency Graphs (RAG): The regions obtained in the segmentation stage are represented as vertices, and relations between neighboring regions are represented as edges.

Hierarchical merging: Picks up the smallest weighing edge of a RAG and combines regions connected by it. The new region is adjacent to all previous neighbors of the two combined regions. The weights are updated accordingly, and it continues to do so till the minimum edge weight in the graph is more than the input value of the threshold parameter.

GrabCut segmentation: Takes an image as input from the user in addition to taking coordinates of a rectangle that corresponds to the part of the image that the user would like to segment, and the user can also tell the algorithm that the pixels inside this defined rectangle are guaranteed background, guaranteed foreground, probably background, or probably foreground. Instead of defining a rectangle, the user can also give a mask that approximates the segmentation. It then iteratively estimates the color distribution of the foreground and background through a gaussian mixture model and constructs a Markov random field over the labels of every pixel (background or foreground). Graph cut optimization is then performed to output the final segmented image.

Watershed segmentation: This is a technique used for segmentation that works on grayscale images. It works by finding an intensity level in the image that will act as a threshold, and then all the pixels that are above this threshold will be marked as foreground, and all the pixels that are below this threshold will be marked as background. Then the algorithm will identify the local minima of the image, these minima will be used as markers, and the algorithm will flood-fill the regions around the markers. This will create a segmentation of the image based on the markers, denoised image, and flooding. The algorithm can take an array of markers as input; if no markers are given, the algorithm will find the local minima itself.

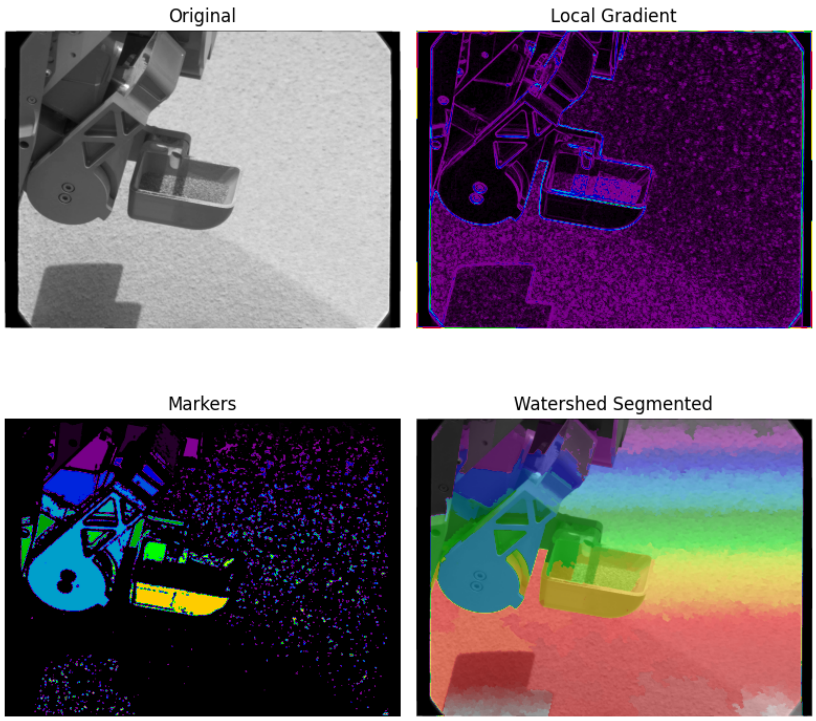
Segmenting image 1:



Discussion for best segmentation (not including GrabCut or watershed): The best segmentation was achieved using Felzenszwalb's algorithm and worked very well for the segmentation of the first image. The chosen parameters for felzenszwalb's algorithm that produce the perfectly segmented image are: ***scale = 500, sigma = 0.85, min_size = 600***; changing any of the three parameters to a higher or lower value either results in images that are starting to over segment or images that do not completely segment the rover from the martian terrain. Other techniques used for the segmentation of the first image produced results that were far inferior to the above-used method because other techniques either could not segment the full rover correctly or segmented points from the martian terrain as part of the rover. Moreover, the segmentation using Felzenszwalb's algorithm above clearly shows how in addition to the rover, its shadow is also being correctly segmented, which is another thing that made this technique superior to the others.

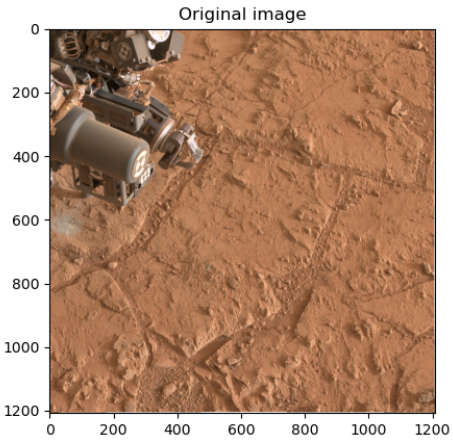
Discussion for GrabCut segmentation: For the first image, the GrabCut segmentation technique worked incredibly well, producing a very detailed image that is perfectly segmented. To produce this image, the following coordinates were used to define the rectangle: smaller x coordinate = 0, smaller y coordinate = 0, larger x coordinate = 1200, larger y coordinate = 900, meaning that all pixels between the given values were defined as part of the rectangle. I also indicated that the areas inside of this rectangle are mostly foreground pixels that I want to segment, which helped the algorithm to discard any pixel outside the defined rectangle as surely background and correctly segment the rover. This algorithm did a really good job in identifying the pixels that belong to the rover inside the defined rectangle because some pixels inside of the defined rectangle belong to the terrain, which the algorithm correctly identified as background and subsequently blackened them out. The only aspect that GrabCut was inferior in compared to Felzenszwalb's algorithm was that the shadow of the rover was not segmented and was discarded

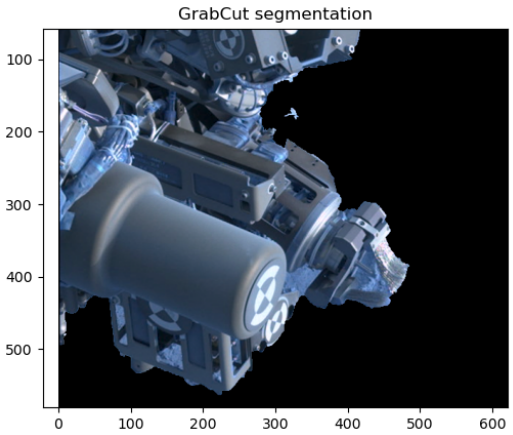
by being identified as part of the terrain, but GrabCut produced the best segmentation of the actual rover.

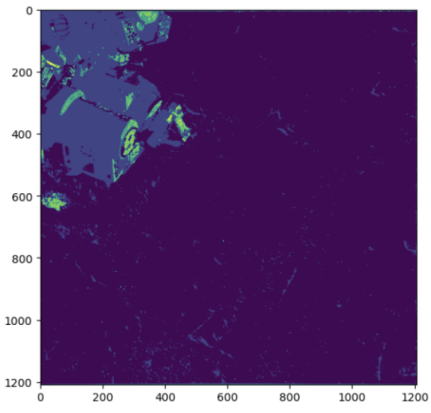


Discussion for watershed segmentation: First, the image is denoised using the ‘rank’ and ‘disk’ functions in the scikit-image ‘filters’ and ‘morphology’ sub-functions. disk(2) is used to denoise the image. For the markers, the threshold value was set as 7 and used disk(5) to get a more smooth image. Then the gradient is found using the ‘rank’ function. disk(2) is used to keep edges thin. I then input the markers and gradients in the watershed function and plot the labels. The markers are sufficient, as you can see, and the gradient perfectly marks the borders of the rover. This results in a fairly good segmentation of the rover from the background. It was noticed that the higher the threshold, the more information is captured by the markers.

Segmenting image 2:





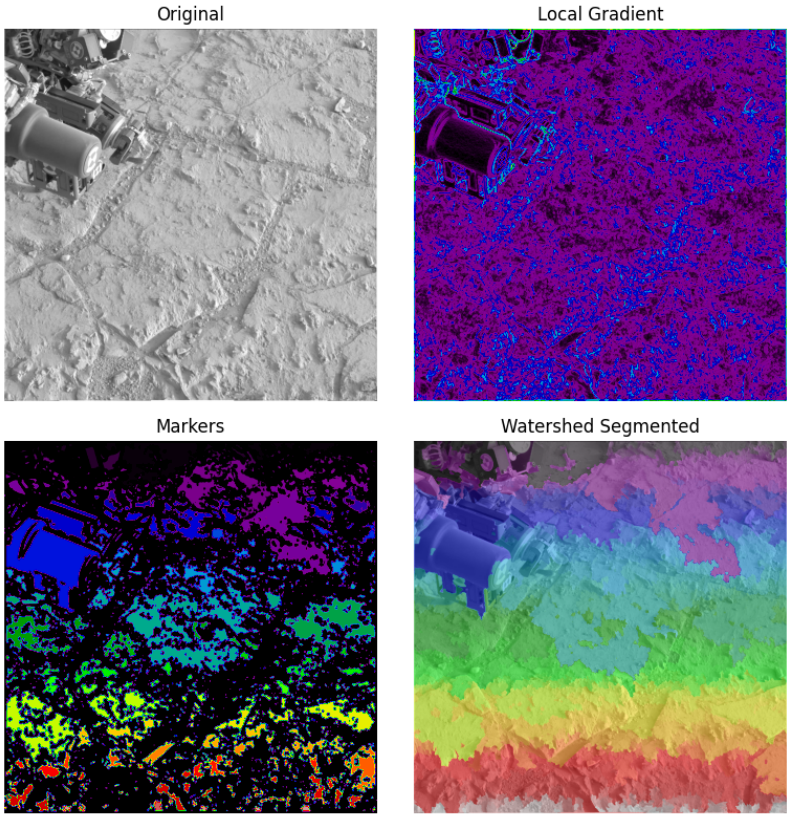


Resulting image after decoloration and mean shift

Discussion for best segmentation (not including GrabCut or watershed): For the resulting image with mean shift, no bandwidth was specified; instead, sklearn's estimate bandwidth function was used with 0.3 quantile, which along with decoloration stretching, was able to segment the rover the most out of all the other combinations were tried. I tried to use mean shift with multiple quantile values being input to the estimate bandwidth function, but the other quantile values either didn't segment the rover well enough or didn't show the details in the rover well enough. Using only mean shift without decoloration stretch also produced inferior results, which had a lot of pixels that are supposed to be a part of the rover being clustered with the terrain. As can be seen from the images, the combination of techniques were used resulted in images where the rover is distinctly segmented from the terrain. I conclude that for this image, decoloration stretch and mean shift produced the best segmentation out of all the other unsupervised techniques since, in the image that is a result of decoloration stretching and mean shift, you can see the rover having a good amount of details as well as being segmented along with its shadow with the least amount of pixels which are supposed to be a part of the rover being mis-clustered as parts of the terrain.

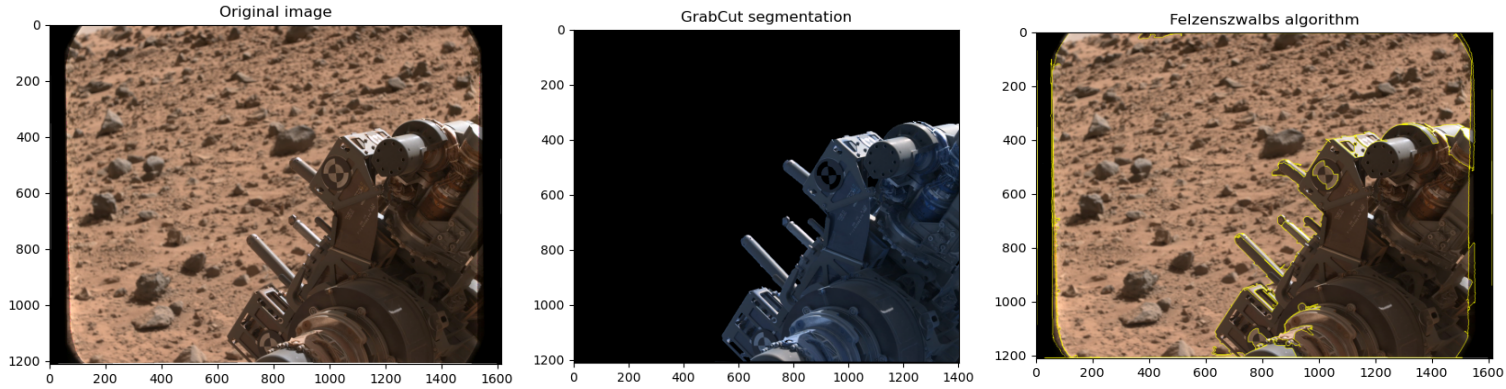
Discussion for GrabCut segmentation: For the third image, the GrabCut segmentation technique worked decently well, producing a detailed image that is not perfectly segmented. To produce this image, the following coordinates were used to define the rectangle: smaller x coordinate = 0, smaller y coordinate = 0, larger x coordinate = 500, larger y coordinate = 580, meaning that all pixels between the given values were defined as part of the rectangle. I also indicated that the areas inside of this rectangle are mostly foreground pixels that I want to

segment, which helped the algorithm to discard any pixel outside the defined rectangle as surely background. As can be seen from the picture above, there are some areas where pixels belonging to the terrain have been taken as foreground by the algorithm and thus are segmented as well. However, more iterations and a custom mask indicating the foreground pixels in a little more detail will likely fix this issue and allow us to get a well-segmented image of just the rover.



Discussion for watershed segmentation: First, the image is denoised using the ‘rank’ and ‘disk’ functions in the scikit-image ‘filters’ and ‘morphology’ sub-functions. disk(2) is used to denoise the image. For the markers, I set the threshold value as 30 and use disk(5) to get a more smooth image. I then find the gradient using the ‘rank’ function. disk(2) is used to keep edges thin. I then input the markers and gradients in the watershed function and plot the labels. Unfortunately, for this image, watershed segmentation did not work the best because of the very slight difference between the rover and the background. I tried other disk values for both the markers and the gradient. However, it did not make much of a difference. I tried different threshold values, but that still did not clearly segment the rover from the background.

Segmenting image 3:

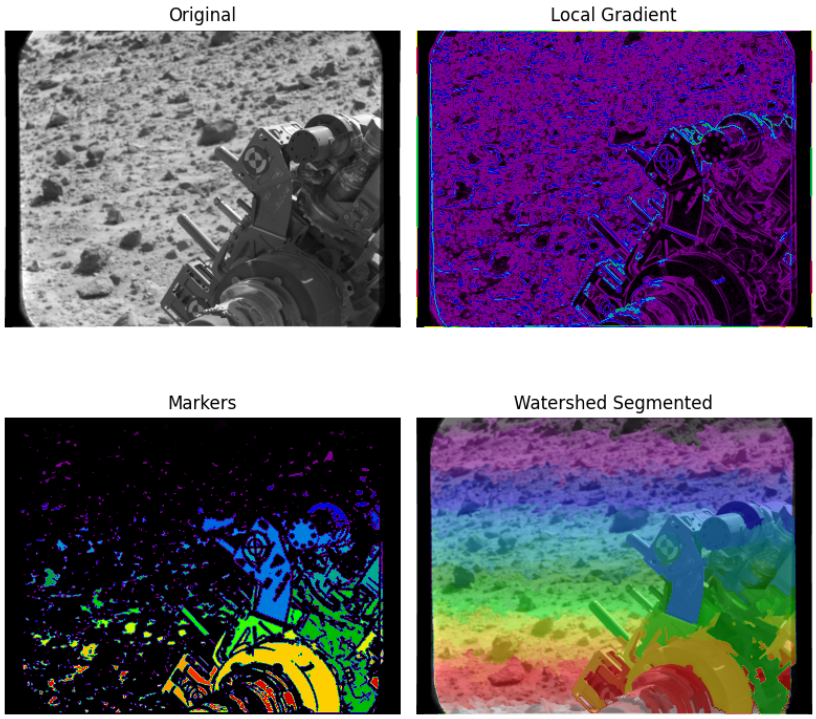


Discussion for best segmentation (not including GrabCut or watershed): The best segmentation was achieved using Felzenszwalb's algorithm and worked very well for the segmentation of the third image. The chosen parameters for felzenszwalb's algorithm that produce the perfectly segmented image are: ***scale = 1200, sigma = 0.58, min_size = 1200***; changing any of the three parameters to a higher or lower value either results in images that are starting to over-segment or images that do not completely segment the rover from the martian terrain. Other techniques used for the segmentation of the first image produced results that were far inferior to the above used method because other techniques either could not segment the full rover correctly or segmented points from the martian terrain as part of the rover. Although this technique segments a small rock near the topmost rod of the rover as being a part of the rover due to its color and placement, as you can see from the above-segmented image using Felzenszwalb's algorithm that the rover is perfectly segmented along its borders and there are no other parts on the terrain that are being segmented (except for the small rock as discussed above).

Discussion for GrabCut segmentation: For the third image, the GrabCut segmentation technique worked incredibly well, producing a very detailed image that is perfectly segmented. To produce this image, the following coordinates were used to define the rectangle: smaller x coordinate = 500, smaller y coordinate = 300, larger x coordinate = 1400, larger y coordinate = 1200, meaning that all pixels between the given values were defined as part of the rectangle. I also indicated that the areas inside of this rectangle are mostly foreground pixels that I want to segment, which helped the algorithm to discard any pixel outside the defined rectangle as surely background and correctly segment the rover. This algorithm did a really good job in identifying the pixels that belong to the rover inside the defined rectangle because some pixels inside of the defined rectangle belong to the terrain, which the algorithm correctly identified as background and subsequently blackened them out. For this image, GrabCut produced a much better segmentation than ANY other unsupervised technique since not even one pixel of the

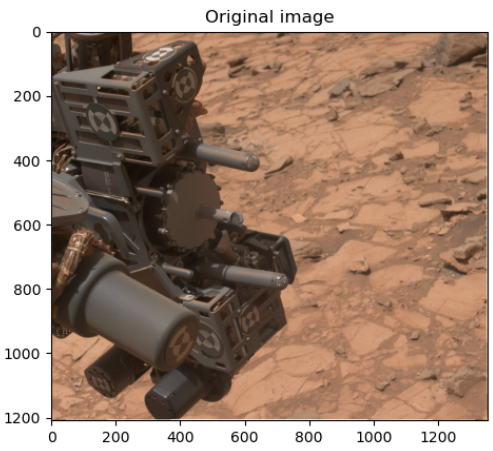
background was segmented while every pixel of the rover was segmented in addition to the details on the rover being perfectly preserved and clearly visible; no other technique was able to achieve this.

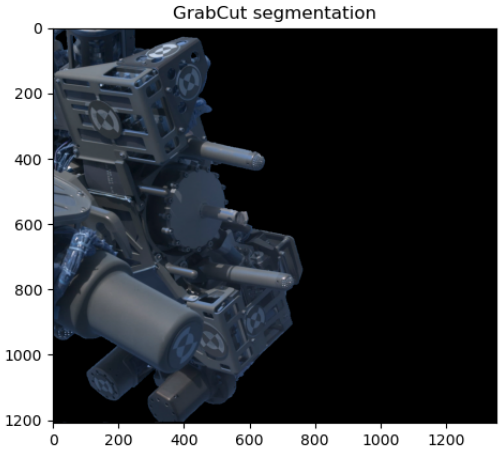
Discussion for watershed segmentation: First, the image is denoised using the ‘rank’ and ‘disk’

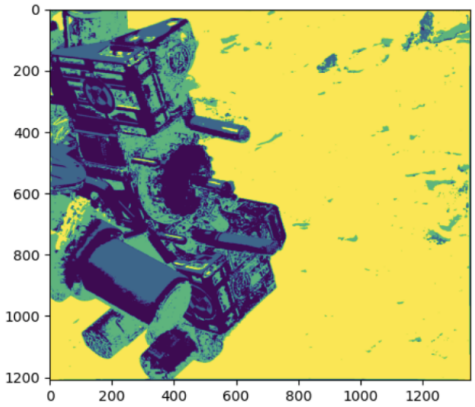


functions in the scikit-image ‘filters’ and ‘morphology’ sub-functions. disk(2) is used to denoise the image. For the markers, I set the threshold value as 13 and used disk(5) to get a more smooth image. I then find the gradient using the ‘rank’ function. disk(2) is used to keep edges thin. I then input the markers and gradients in the watershed function and plot the labels. This resulted in a very good segmentation of the rover from the background. I chose the threshold as 13 because the markers gave more information on the rover. The edges on the gradient also perfectly create a border about the rover. This is a good segmentation. However, not as good as some of the other methods used.

Segmenting image 4:





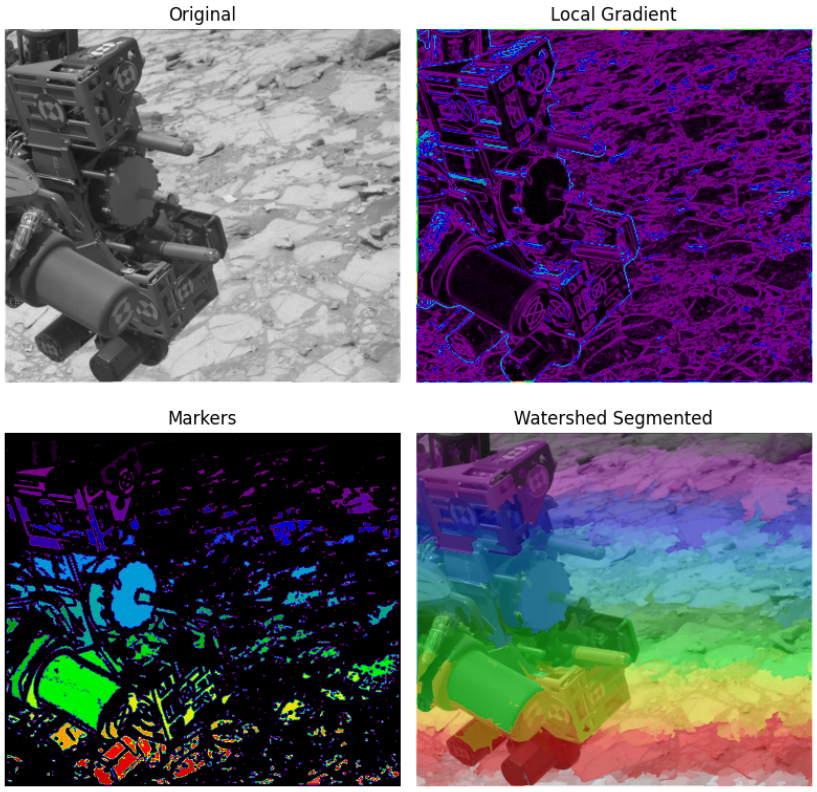


Resulting image after converting to HSV format, histogram Equalization, and k means

Discussion for best segmentation (not including GrabCut or watershed): For the resulting image with k means in HSV format, which I found to be the best result out of all the combinations, were tried along with k means since it distinctly segments the rover, 4 clusters were used, which is why you see the 4 distinct colors. The rover is fully segmented from the terrain, and only some pixels that belong to the terrain are clustered with some part of the rover (as can be seen by some blue pixels on the yellow when the entire terrain should have been yellow). Histogram equalization was needed to improve the contrast since the resulting image without it had much lower contrast. I could see that the image resulting from k means was superior to any other unsupervised technique since it showed a significant amount of detail, could fully segment the rover, and had fewer mis-clustered pixels. As seen from the image above, some terrain areas have the same color as the rover. However, this is because the algorithm clustered those pixels with the rover. I think this is the case because these pixels have very similar intensities (colors) as the rover, as well as noise being a contributing factor, and cannot be perfectly segmented.

Discussion for GrabCut segmentation: For the fourth image, the GrabCut segmentation technique worked incredibly well, producing a very detailed image that is perfectly segmented. To produce this image, the following coordinates were used to define the rectangle: smaller x coordinate = 0, smaller y coordinate = 0, larger x coordinate = 800, larger y coordinate = 1350, meaning that all pixels between the given values were defined as part of the rectangle. I also indicated that the areas inside of this rectangle are mostly foreground pixels that I want to segment, which helped the algorithm to discard any pixel outside the defined rectangle as surely

background and correctly segment the rover. This algorithm did a really good job in identifying the pixels that belong to the rover inside the defined rectangle because some pixels inside of the defined rectangle belong to the terrain, which the algorithm correctly identified as background and subsequently blackened them out. For this image, GrabCut produced a much better segmentation than ANY other unsupervised technique since not even one pixel of the background was segmented while every pixel of the rover was segmented in addition to the details on the rover being perfectly preserved and clearly visible; no other technique was able to achieve this.



Discussion for watershed segmentation: First, the image is denoised using the 'rank' and 'disk' functions in the scikit-image 'filters' and 'morphology' sub-functions. disk(2) is used to denoise the image. For the markers, I set the threshold value as 8 and used disk(5) to get a more smooth image. I then find the gradient using the 'rank' function. disk(2) is used to keep edges thin. I then input the markers and gradients in the watershed function and plot the labels. This also resulted in a good segmentation, except that some of the protruding elements, assuming they are antennas, have not been segmented as the rover. I tried changing the threshold, but the higher it was, the more information about the background was being captured, and any lower and some of the details on the rover were not being captured. I tried changing the disk for the markers and gradient, but they resulted in worse segmentations. This is a good segmentation. However, not as good as some of the other methods.

Overall Result: Through a combination of the above-mentioned techniques, I was successfully able to segment the rover from the martian terrain for all 4 images provided. GrabCut segmentation was clearly the method that segmented the rover in each image perfectly, and the resulting segmentation achieved through this algorithm had just the rover, with all the terrain being blackened out, making it much easier to see. A strength of this method was the fact that the background could be completely blackened, allowing for the segmented aspects of the image to be very visible and look crisp. While providing a really good segmentation, GrabCut was also able to preserve the details that were present on the rover; however, GrabCut could only segment the rover but not its shadow.