Ashton Gray

ECE 597IP

32000589

Felzenszwalb

Background:

Felzenszwalb is an efficient graph-based image segmentation method. Graph-based segmentation is when each pixel in an image is treated as a node of a graph, where the edges connect the pixels based off of their similarity, or affinity, to each other. The graph is then cut, and those nodes that are left connected are each segmented part of the image. Felzenszwalb's algorithm does this by taking the graph and divides it into distinct regions. A predicate determines if there is a segmentation boundary between these regions by checking if the difference between two regions is greater than the internal difference of each region. This is controlled by a constant, where the larger the constant, the larger space needed between regions for a segmentation boundary, therefore fewer and larger segments. Similarly, the smaller the constant, the less space needed between regions for a segmentation boundary, therefore more and smaller segments. These segments are then refined, so that it is not too fine (too many segments), or too course (too little segments), by creating the graph one edge at a time according by its weight, creating a tree. Meanwhile this tree is checked so that the weight of the tree does not go over a threshold. This creates an efficient tree of each segmentation.

Sci-Kit Image: felzenszwalb

Sci-Kit Image implements the felzenszwalb algorithm into python. The parameters taken in are the image, a scale (which is similar to the constant mentioned above), a sigma (which is the size of a Gaussian kernel that blurs the image before performing segmentation), and a minimum size (which is the smallest size that a segment can be).

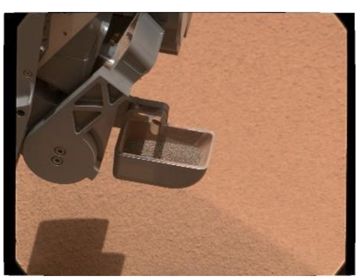


Image 1

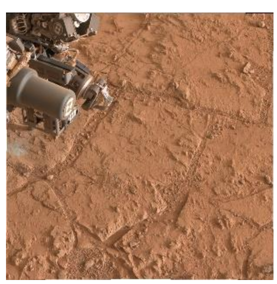


Image 2

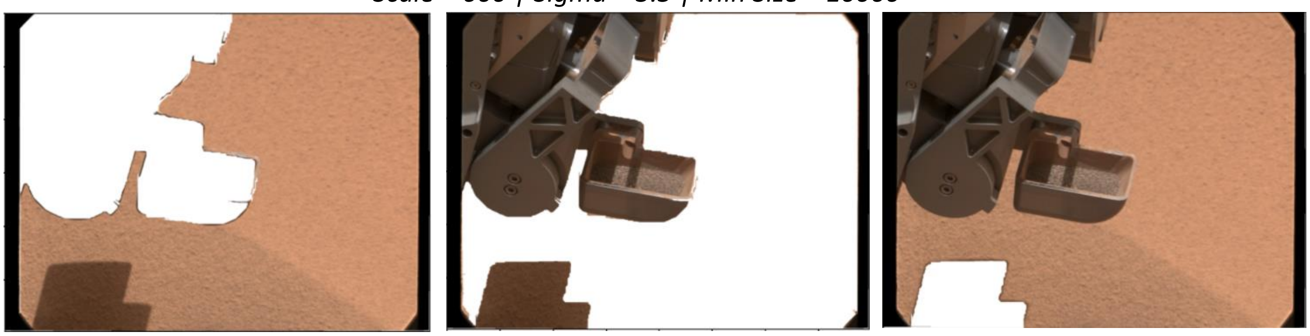
Image 3

Image 4

Above are the original images that segmentation will be performed on

Image 1

Image 1 After Felzenszwalb Performed
Scale = 600 | Sigma = 3.5 | Min Size = 10000

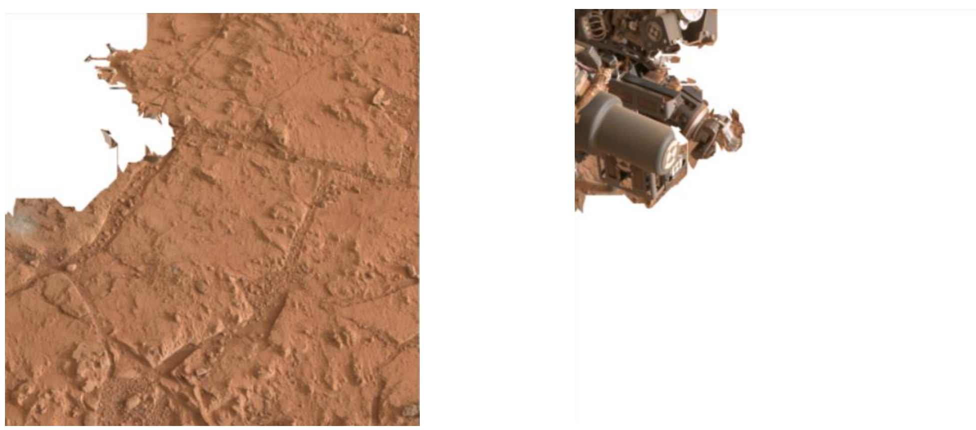


Original Image 1, Segmented

With these values, the algorithm was able to separate the rover, soil, and rover's shadow from one another. As you can see, a high scale value was used in order to not over segment the rover and soil. A lower min size was also used to make the shadow its own segment and not included with the rover. You can also notice that the edges are not perfect, this is because of the Gaussian blur that was used for pre-processing.

Image 2

Image 2 After Felzenszwalb Performed
Scale = 80 | Sigma = 20 | Min Size = 17000

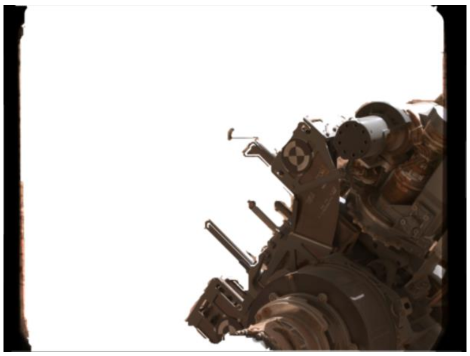


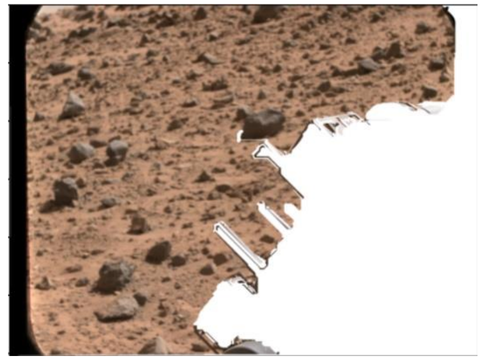
Original Image 2, Segmented

Due to the uneven terrain, a heavy blur was necessary to smooth the picture, and therefore separate the rover from soil. Otherwise, parts of the soil would be separated into its own segment. However, this had some negative effects. Looking at the rover, parts of the edges are the soil, and this is due to the blur blending soil and rover.

Image 3

Image 3 After Felzenszwalb Performed
Scale = 300 | Sigma = 8 | Min Size = 40000



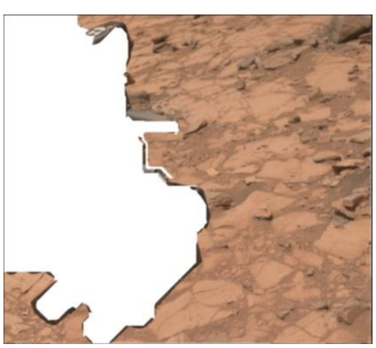


Original Image 3, Segmented

Similarly, to Image 2, a large blur was used to prevent all of the rocks in the soil from segmenting into their own clusters. Along with this, a large min size was used to prevent the rocks from being in their own clusters, and allowed for the tops of the rover to join the same cluster as the rover itself. As you can see, this was not perfect either, parts of the soil were included with the rover, and this could be due to the blurring of the image, where parts of the rover were blended closer to the soil than the rest of the rover.

Image 4

Image 4 After Felzenszwalb Performed
Scale = 230 | Sigma = 15 | Min Size = 30000



Original Image 4, Segmented

This image, also had to be heavily blurred due to the rocks in the surrounding soil. This had a similar effect where the edges around the rover included a small area of soil. However, you can see that the bottom left of the image, was segmented into its own cluster. This is a limitation of the algorithm, where if I increased either the scale to create less clusters, or increased the minimum size so that it would not allow that corner to be a cluster, the segment would just be included as part of the rover. This could be because location has an impact on the algorithm, and therefore was included into the rover.

Over-segmented Felzenszwalb

Just using the felzenszwalb algorithm has its limitations. A major limitation was due to the blur of the image. In order to get the most accurate segmentation, the image needed to be heavily blurred to separate rover from soil. Instead, we can over-segment using felzenszwalb, then use other clustering methods on top of it by using a region adjacency graph (RAG). A RAG represents each segment in an image as a node in a graph, with the edges connecting adjacent segments. The weight of these edges can be calculated by things such as difference in average color. This is what will be used. We can then merge nodes in this graph so that there are no edges with a weight left under some threshold.

Hierarchical Merge:

One method we can use on the RAG is hierarchical merging. This method merges similar clusters based off of the RAG, with the weights of edges being the average colors, creating a "hierarchy", and using a threshold to cut edges on the different segments of said hierarchy.

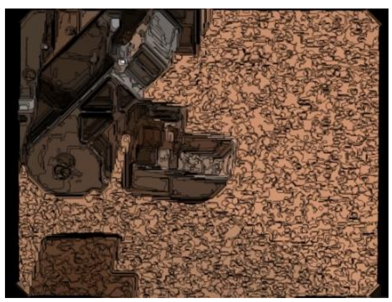


Image 1 After Felzenszwalb Over-Segmenting
Scale = 2.5 | Sigma = 5 | Min Size = 100

Image 1 After Hierarchical Merge
Threshold = 0.5

If we use too many segments for the Felzenszwalb, as you can see, the hierarchical merge has a hard time merging correct parts. For example, you can see that parts of the rover were segmented into their own clusters. To fix this, we will use a higher min size for the felzenszwalb, where it is still over-segmenting, but each segment will be larger. This brings up limitations of the algorithm, where boundaries of the image are slightly crossed, however better segmentations are found.

Image 1:

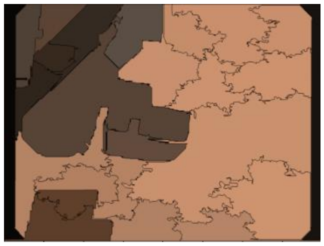
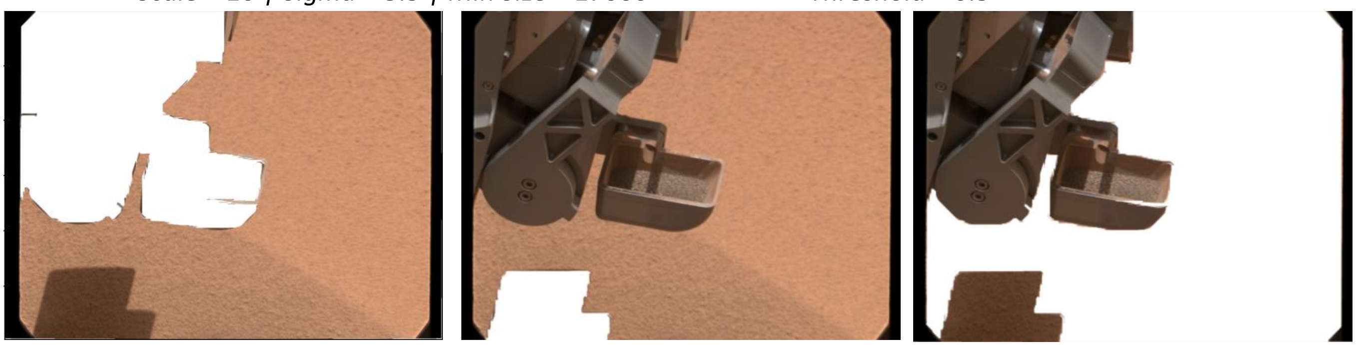


Image 1 After Felzenszwalb Over-Segmenting
Scale = 10 | Sigma = 3.5 | Min Size = 17000

Image 1 After Hierarchical Merge
Threshold = 0.3



Original Image 1, Segmented

Here you can see that the edges are slightly better defined than when only using Felzenszwalb. This is due to the less blur and the over-segmenting crossing over less boundaries. A higher min size was also used in order to include the inside of the scoop as part of the rover, otherwise the Felzenszwalb would include the soil in the scoop as part of the rest of the soil.

Image 2:

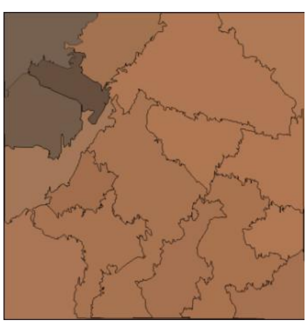
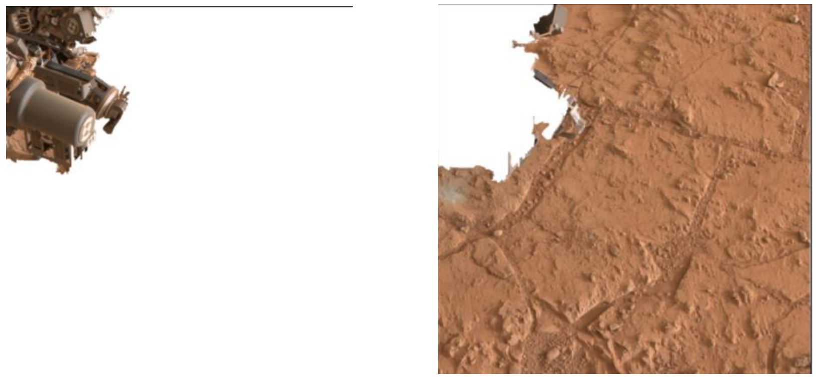


Image 2 After Felzenszwalb Over-Segmenting
Scale = 10 | Sigma = 3.5 | Min Size = 17000

Image 2 After Hierarchical Merge
Threshold = 0.1



Original Image 2, Segmented

This was also a limitation with over-segmenting. If the segments were too small, the rover would not be clustered to itself, but for example here, boundaries of the top right of the rover were crossed. Therefore, it was not a clear segment.

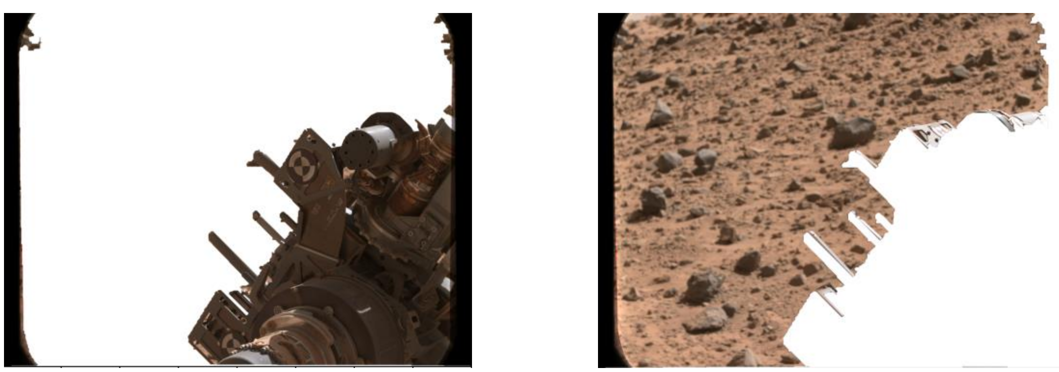
Image 3:

Image 3 After Felzenszwalb Over-Segmenting

Scale = 2.5 | Sigma = 2 | Min Size = 15000

Image 3 After Hierarchical Merge

Threshold = 0.5



Original Image 3, Segmented

As you can see, a pro with over-segmenting, you can see that the image did not need to be heavily blurred to worry about preventing the rocks from clustering. However, a limitation of this algorithm is that parts of the rover boundaries are being crossed. As mentioned earlier this was necessary however, to make the rover cluster into one segment.

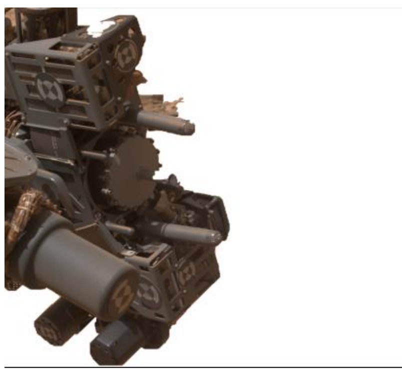
Image 4:

Image 4 After Felzenszwalb Over-Segmenting

Scale = 2.5 | Sigma = 2 | Min Size = 15000

Image 4 After Hierarchical Merge

Threshold = 0.5



Original Image 4, Segmented

An important thing to notice with this image when compared to just Felzenszwalb, is that the bottom left corner was not segmented into its own cluster. Instead, it was included with the rest of the soil. This is because of how the heirarchical merging works, and due to the affinity of soil, they were cut into the same segment. However, again, the same issue of crossing boundaries shows.

<u>N-Cuts Merge:</u>

Normalized Cuts (N-Cuts) is another method that can be used on the RAG. This method, computes the cost of a cut in relation to the total weights of edges in the graph. This can be an advantage because it is unbiased, and tries to create an even size equality between segments. However, this can also be a downfall of the algorithm, if the size of one object is a much different size than the others. The inputs for N-Cuts are the image, the RAG, a threshold for weights to combine clusters, and a num cuts, which is the minimum number of cuts used to find the most efficient cut, due to being a recursive algorithm.

To compare to hierarchical merging, the same Felzenszwalb inputs will be used for each image as shown earlier.

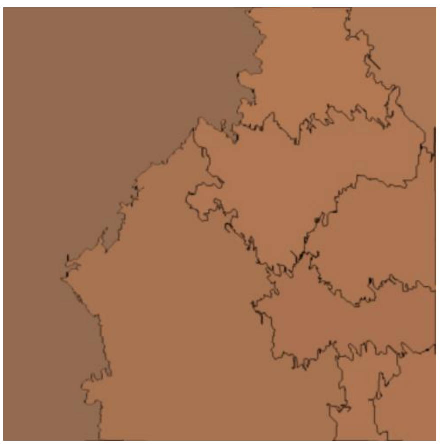
Image 1:

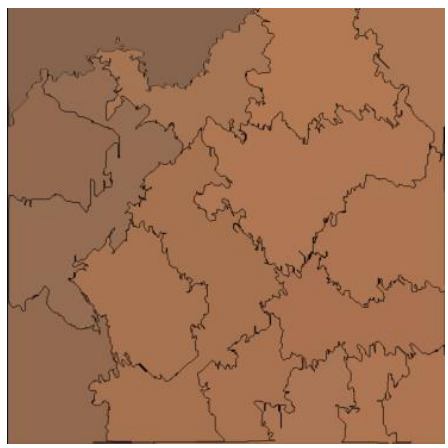
Threshold = 0.2 *Threshold = 0.8*

Image 1 After N-Cuts Merging

Both of these images were the N-Cuts algorithm performed on the over-segmented Felzenszwalb algorithm with scale = 10, sigma = 3.5, and min size = 17000. As you can see the result was much worse than hierarchical merging. The low threshold, showed 4 segments, however the rover is not distinguishable in the image. Even with a high threshold, and a high number of segments left, parts of the rover were still merged into soil. This is a big limitation of the algorithm. Due to not averaging surrounding values for the weights of edges, the wrong segments are clustered together.

Image 2:





Threshold = 0.1 *Threshold = 0.8*

Image 2 After N-Cuts Merging

Both of these images were the N-Cuts algorithm performed on the over-segmented Felzenszwalb algorithm with scale = 2.5, sigma = 2, and min size = 15000. Similar to the previous image, less segments overlap the rover and soil, as well as with higher segments, the rover and soil are still combined into the same segments.

Image 3:

Threshold = 0.2 *Threshold = 0.8*

Image 3 After N-Cuts Merging

Both of these images were the N-Cuts algorithm performed on the over-segmented Felzenszwalb algorithm with scale = 2.5, sigma = 2, and min size = 15000. Again, the same issue of overlapping segments for both large and small clusters.

Image 4:

Threshold = 0.2 *Threshold = 0.8*

Image 4 After N-Cuts Merging

Both of these images were the N-Cuts algorithm performed on the over-segmented Felzenszwalb algorithm with scale = 2.5, sigma = 2, and min size = 15000.

For each of these, the N-Cuts algorithm performed much worse than the hierarchical merging. This is due to how the N-Cuts algorithm works as mentioned earlier. The segments are trying to be created of similar sizes, therefore in order to do that, parts of the soil must be merged with the rover. Along with this, not having a bias of color means that the segments of rover and soil are more likely to be merged together.

MeanShift:

For each of the RAG methods mentioned above, there was an issue of when over-segmenting boundaries had to be crossed, and each segment had to be large. Otherwise, when a merging method was used, small parts of the rover or soil would be in its own cluster. However, in the previous project, we mentioned the effectiveness of the mean-shift algorithm. One pro of using mean-shift is that location of a pixel was found to only help the algorithm, therefore we can use location to help make the clustering better. Another perk of this algorithm was being able to have edge detection, and therefore very sharp edges. Mentioned earlier, this was a downfall for the Felzenszwalb algorithm. Therefore, instead of using a RAG merging algorithm, we can try to over-segment the image so we don't cross boundaries using Felzenszwalb, and perform the mean-shift on top.

Image 1:

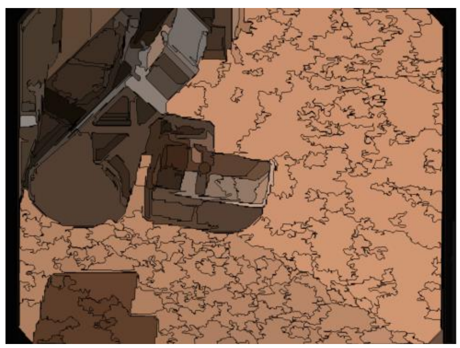
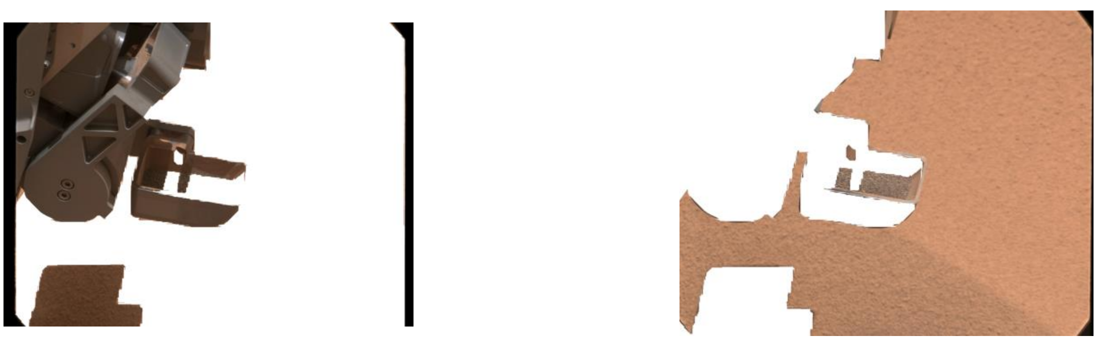


Image 1 After Felzenszwalb High Over-Segmenting
Scale = 10 | Sigma = 3 | Min Size = 1000



Original Image 1 After Mean-shift was Performed on Felzenszwalb Over-Segmenting
Bandwidth = 0.5

As you can see, the over-segmenting did not cross any object boundaries. Therefore, when the mean-shift algorithm was used, the edges of the rover and soil were very clear. However, when the algorithm was unable to distinguish the inside of the scoop from the soil, and placed them into the same cluster. This is a limitation of mean-shift, where there the pixels in the scoop go towards to local maximum which is the rest of the soil. This could be fixed, which was showed in project 1, by blurring and using a bandwidth that would over-segment again, and create more clusters, but completely segment the rover, shadow, and soil.

Image 2

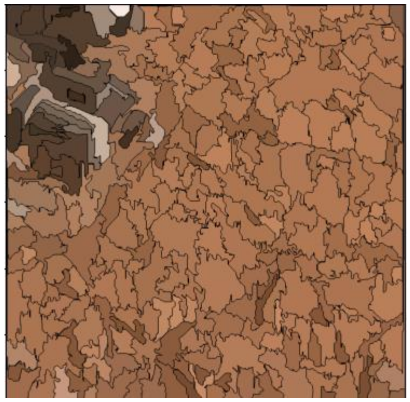
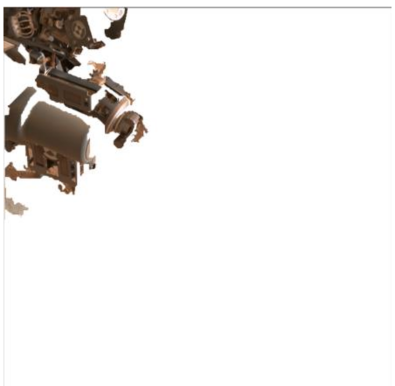
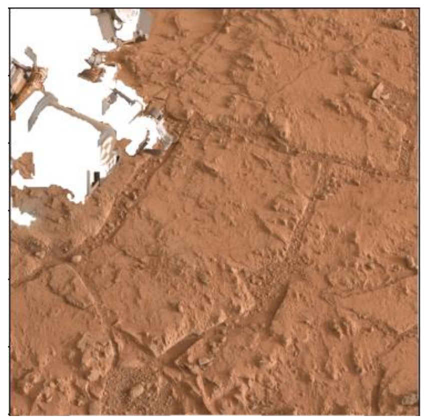


Image 2 After Felzenszwalb High Over-Segmenting
Scale = 2.5 | Sigma = 3 | Min Size = 1000





Original Image 2 After Mean-shift was Performed on Felzenszwalb Over-Segmenting and Decorrelation Stretching
Bandwidth = 0.45

From project 1, we found that decorrelation stretching was needed in order to get a good result from the mean-shift in this image, and therefore the same was done. The results are not as good as just hierarchical merging, and this is because of another limitation of the mean-shift algorithm. When the colors of two objects are too similar, the algorithm has a hard time being able to differentiate the two from each-other, and therefore cluster them into the same segment.

Image 3:

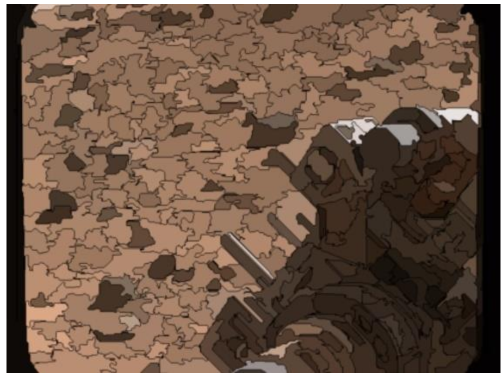
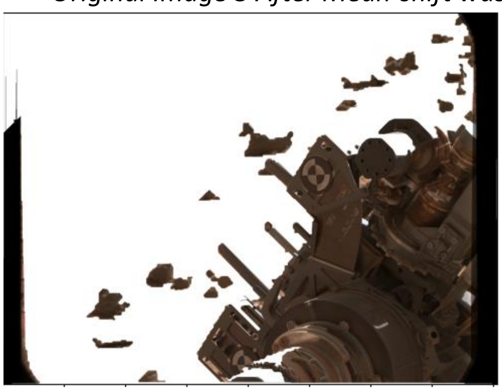
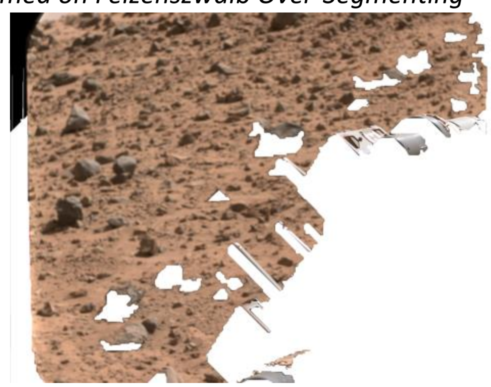


Image 3 After Felzenszwalb High Over-Segmenting
Scale = 2.5 | Sigma = 3 | Min Size = 1000

Original Image 3 After Mean-shift was Performed on Felzenszwalb Over-Segmenting



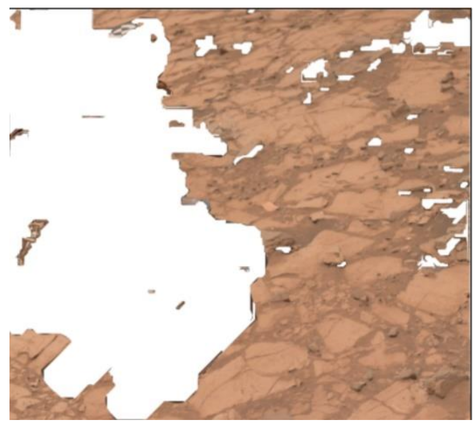


Bandwidth = 0.5

As you can see, the cluster for rover, included the surrounding rocks, and the soil included parts of the silver linings of the rover. Again, this is a limitation of the mean-shift algorithm, where if colors are too closely related, the algorithm, will cluster the objects together. However, this was better than hierarchical merging, due to the sharp edges of the rover to soil.

Image 4:

Image 4 after Felzenszwalb Over-Segmenting
Scale = 2.5 | Sigma = 5 | Min Size = 100



Original Image 4 After Mean-shift was Performed on Felzenszwalb Over-Segmenting and Decorrelation Stretching

From project 1, it was found that decorrelation stretching created the best results when in combination of mean-shift for this image, and therefore was used. A low min size was used for this image due to the high number of edges of the rover, and therefore in order to not cross boundaries, a lower min size was needed. Similarly, to the other images, the edges of the rover are very clear. However, parts of the soil from the right side of the image were included in the same cluster as the rover. This could be due to the similarity in color as the rover, which mean-shift is not very effective against.

Conclusion

In conclusion, the Felzenszwalb algorithm is very effective. On its own, the algorithm was able to segment each of the images with generally high accuracy. The only downside were the blurred edges due to needing to heavily blur most images in order to remove detail and get higher accuracy. However, a slight remedy to this, Felzenszwalb over-clustering, and a RAG were used for hierarchical merging of clusters. Hierarchical merging was able to effectively cluster an over-segmented image into rover and soil. The issue of blurring was still present, but lessened. The other method of merging, N-cuts, was tested as well. This method was much worse than hierarchical merging, and this was found to be due to not having a bias towards color, and only using distance and creating equal sized objects. Lastly, in order to get hierarchical merging to work well, parts of the over-segmentation crossed boundaries and therefore detail was lost. Instead, mean-shift was tested on the Felzenszwalb high over-segmenting. This method effectively removed the issue of blur between soil and rover, however due to limitations of the mean-shift algorithm explained in project 1, parts of the soil of similar color to the rover were included in the same clusters.