Comparative Analysis of Image Processing Techniques to Identify Mines

ENGR 3499: Image Processing
Jiayuan Liu, Emma Mascillaro, Miranda Pietraski, Gia-Uyen Tran
12/13/2024

Introduction

The expansion of the human population and the development of technology have had great impacts on the planet. With growing threats of climate change and resource depletion to the environment, it is critical to monitor changes underway. The widespread use of satellite photography makes it more possible now than ever to track and study large-scale changes like city development, glacier melting, deforestation, and many other phenomena.

Image segmentation is a valuable tool in remote sensing for its ability to streamline the process of identifying features in an image and even find things the human eye might miss. With this project, we were interested in exploring methods of image segmentation by using them to study satellite images of mines.

Problem Statement

We set out to explore the differences in the segmentation methods explained above by using them on photos featuring a variety of different mines from around the world. We decided to use mines because we noticed a wide range of physical features could indicate the presence of a mine while exploring the satellite data. We hypothesize that different clustering methods would be better suited to identifying different mines depending on their features.

Our main questions to answer were:

- What image processing methods are best able to identify mines from satellite pictures?
- What are the pros and cons of different clustering methods?
- Are there methods that work better for different types of mines?

Literature Review

After preliminary research on segmentation techniques, we focused on two major types of image segmentation: clustering based methods and edge based methods. Our goal for clustering was to group image pixels to identify mines. These methods would also allow us to easily convert pixels to approximate land areas. In contrast, our goal for edge based

methods was to identify road infrastructure that supported the mines. We also explored strategies to use edges to identify mines by enclosing them in polygons.

Clustering Methods

In image processing, clustering is a method of segmentation involving assigning individual pixels to clusters based on some shared criterion. Figure 1 illustrates the intuition of a variety of methods using different toy datasets. We explored several of the below methods to understand their effectiveness in segmenting satellite images.

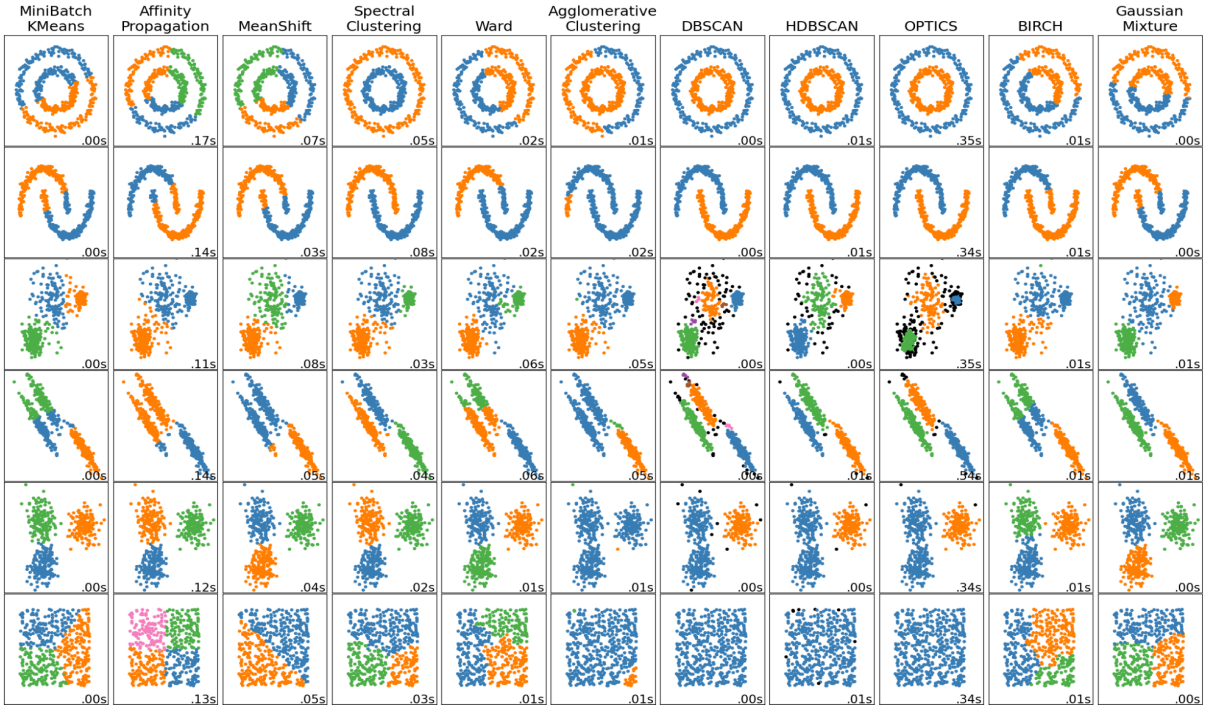


Figure 1: Application of various clustering methods on different toy datasets from the [scikit website].

Agglomerative Clustering

Agglomerative clustering is a bottom-up hierarchical clustering method. The algorithm searches the data points and determines the Euclidean distance between each pair of points. It finds the pair with the smallest distance and links them. These linked points are considered a single point. Next, the algorithm repeats the process of linking the closest points. Eventually, all points will be linked into one big cluster. The final step is to choose a number of clusters or distance to act as a threshold.

The results of this process are often visually represented as a dendrogram, with the vertical axis indicating linkage distance. The greater the linkage distance, the lower the number of identified clusters.

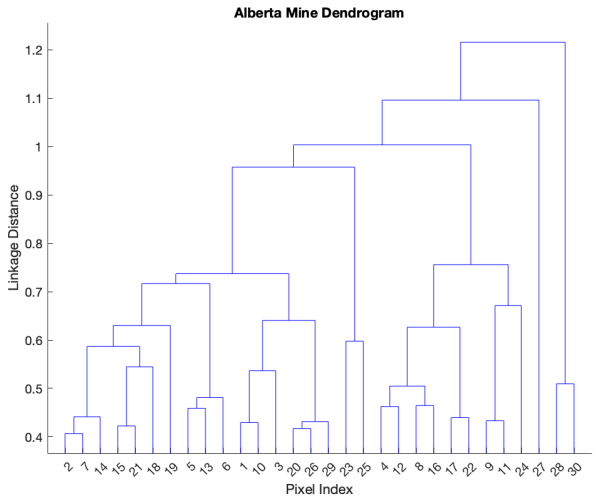


Figure 2: A dendrogram generated from the linkage function in MatLab with the 2020 Alberta mine as input data.

While this method is simple and easy to understand, its major weaknesses are a time complexity of $O(n^3)$ and that it requires $\Omega(n^2)$ memory. As such, it is not suitable for high resolution data. However, we still chose to implement agglomerative clustering to examine whether it would be effective with downsampled data.

K-Means

One of the first methods investigated was K-Means. It is a method of locating a specified number of means within a dataset by iteratively minimizing the average distance in Euclidean space between each point and its assigned cluster mean. This can be represented mathematically by

$$J(c,u) = \sum_{i=1}^{k} \sum_{j=1}^{n} \left|\left| x_i - u_{(c)j} \right|\right|^2$$

where J represents the sum of the squared distances between points x and their corresponding means $u_{(c)}$. The iterative step of K-Means,

$$u_{(c)i} = \frac{1}{|k_i|} \sum_{j \epsilon k}^{k} x_j$$

is the coordinate descent of J(c,u), an algorithm designed to find the minimum of a function.

To actually find clusters, the K-Means method performs the following steps:

1. Randomly assign each of a k amount of means to points within the Euclidean space of the dataset
2. Based on the current means, for each point, determine which mean is the smallest distance away and assign the point to that cluster
3. Once all points are assigned, recalculate the means to be the actual means of their cluster of points
4. Repeat steps 2 and 3 until the means do not change

The K-Means method has the advantage of being relatively easy to understand and perform. As shown in figure 1, it is most effective in segmenting similarly sized, circular clusters. It has the disadvantages of needing a predefined number of clusters to find and being less suited for clusters with complex shapes.

Gaussian Mixture Model

Gaussian Mixture is similar to K-Means, with the notable difference of a probabilistic approach rather than a distance one. This method tries to model the dataset as a mixture of Gaussian distributions, then, for each point, calculates the probability that it belongs to each of the Gaussians. The points are assigned a cluster based on their highest probability.

The Gaussians are defined by the parameters: mean (center), covariance (width), and probability. These are shown graphically in Figure 3. The probability density is defined as a function of the densities of the k distributions:

$$p(X) = \sum_{k=1}^{K} \pi_k G\left(X|\mu_k, \Sigma_k\right)$$

where πk is the mixing coefficient for the k-th distribution.

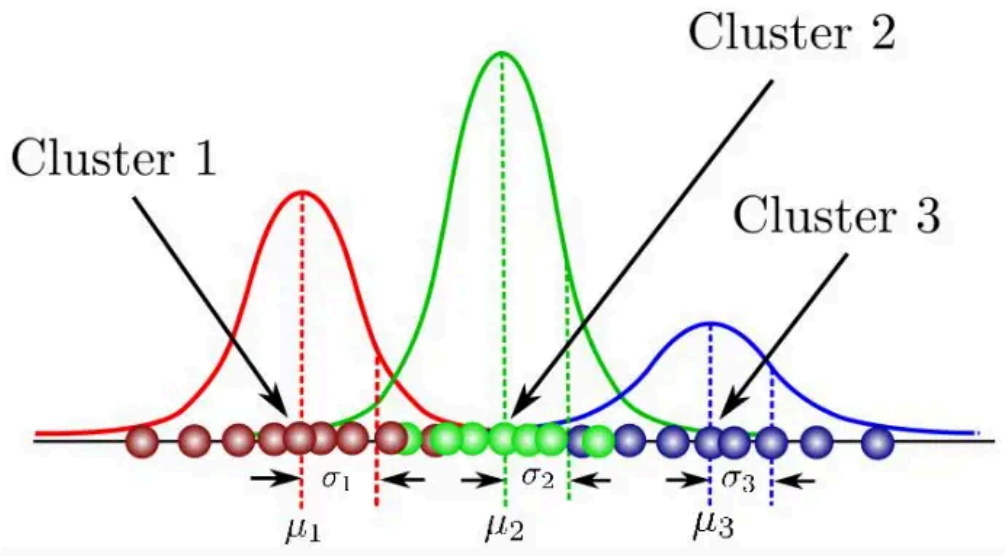


Figure 3: Representation of data as a mixture of Gaussians with labeled parameters.

The steps of the gaussian mixture model are as follows:

1. Set k, the number of clusters to form
2. Initialize mean, covariance, and weight parameters for each cluster
3. Implement the Expectation-Maximization (EM) Algorithm
    a. Expectation Step: Calculate the probability of each point belonging to each cluster with the current parameters and evaluate the likelihood function
    b. Maximization Step: Update the means, covariances, and weights to maximize the expected likelihood found in the expectation step
4. Repeat step 3 until the model converges

DBSCAN

Another clustering method used is density-based spatial clustering of applications with noise, or DBSCAN. DBSCAN places specific emphasis on searching for continuity when identifying clusters. It is based on the idea that, for a point within a cluster, there must be some number of surrounding points also a part of that cluster. To quantify this, we use the terminology of **core points**, **border points**, and **outliers**.

The parameters involved in DBSCAN are **epsilon** and **minimum points**. Epsilon represents the radius around a point in which to look for other points in a cluster. Two points within epsilon distance from each other are considered neighbors. The minimum points refer to the minimum neighbors a point needs to be considered a **core point**.

A core point is a point with more than the minimum points needed within its neighborhood. A border point does not have the number of neighbors needed to be considered core but is in the neighborhood of at least one core point. An outlier is a point

without the required number of neighbors and not within the neighborhood of a core point. An example of these is shown in figure 4.

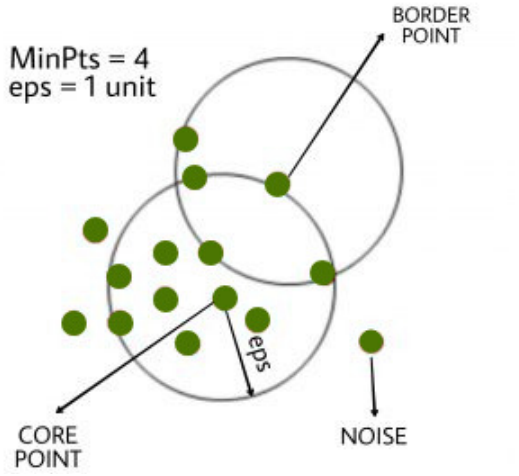


Figure 4: Example diagram of DBSCAN clustering and categorization of points

DBSCAN performs the following steps to define clusters:

1. Visit each point and determine if it has the neighbors required to be considered a core point
2. Assign a new core point a new cluster
3. Find density-connected core and border points and assign them the same cluster
4. Repeat 2 and 3 until all core points are visited
5. Any remaining non-core and non-border points are marked as noise or outliers

Edge/Road Detection

Active Contour

To better understand how to fit curved roads, we explored active contour, a technique used to find an object's boundary in a 2D image. The concept of active contour is analogous to a stretchy rubber band being placed around the target object. Based on the gradient map of the initial image, the contour is iteratively stretched to fit the shape of the object's boundary.

The contour is constructed as a list of points. Around each contour point, a local window is defined with a customized size, as shown in Figure 5. Through the iterations, the contour points are shifted in the direction towards the "peaks" of the gradient map, which is ideally the location of the target object boundary.

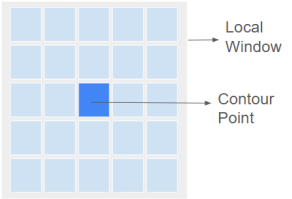


Figure 5: Depiction of a 5x5 local window used in active contour

To implement this iterative shifting, three components of the "stretching force" are defined to guide the movement of the contour points during each iteration:

a. **E_Image:**
This is the reversed sum of the squared magnitudes of the contour points when projected onto the gradient map. The maximum absolute value of E_image indicates that the contour points lie on the "peaks" of the gradient map. While the goal is to eventually move the contour points to these "peaks," E_image is defined as the negative sum and is minimized when the contour converges with the actual boundary.

b. **E_Eelastic:**
This term represents how much the contour behaves like a contracting rubber band. It is defined as the rate of change in the curve between two neighboring contour points. Mathematically, it is calculated as the first derivative of the discretized contour v:

$$E_{elastic}(v_i) = \left|\left|\frac{dv}{ds}\right|\right|^2 \approx \left|\left|v_{i+1} - v_i\right|\right|^2 = \left(x_{i+1} - x_i\right)^2 + \left(y_{i+1} - y_i\right)^2$$

c. **E_smooth**
This term indicates how smooth and free from twists and turns the contour is. It is defined as the curvature of the contour and is mathematically calculated as the second derivative of the discretized contour v:

$$E_{smooth}(v_i) = \left|\left|\frac{d^2v}{ds^2}\right|\right|^2 \approx \left|\left|\left(v_{i+1} - v_i\right) - \left(v_i - v_{i-1}\right)\right|\right|^2$$

$$= \left(x_{i+1} - 2x_i + x_{i-1}\right)^2 + \left(y_{i+1} - 2y_i + y_{i-1}\right)^2$$

Together, these components form the total "energy" of the active contour:

$$E_{tot} = E_{image} + \alpha E_{elastic} + \beta E_{smooth}$$

To approach the actual target boundary, E_tot must be minimized at each step of contour movement.

Implementation

Dataset

We selected mines to maximize image diversity. While Canada Oil Mines were the most representative of normal mines, we used this project as an opportunity to test the efficacy of these algorithms across different mines. We searched for the largest mines in the world. From there, we selected the most unique ones.

We manually created a dataset with images from Google Earth. We took standardized screenshots of all available historical data, which typically included one image per year from 1984 to 2020. In total, our data includes 172 images of 5 mines.

Name	Location	Resources	Start Year	End Year	Rationale
Canada's Oil Sands	Alberta, CA	Oil	1984	2022	A "typical" mine: a green background with beige clusters for mining activity
Mir Mine	Mirny, Russia	Diamond	1999	2020	Similarly colored town directly adjacent to mine
Grasberg Mine	Central Papua, Indonesia	Copper, Gold	1984	2020	Rocky terrain with subtle mine borders
Tagebau Garzweiler	Jüchen, Germany	Lignite	1984	2020	Large patches of farmland with similar colors to the mine
Chuquicamata	Calama, Chile	Copper	1984	2020	Lack of color distinction

Table 1: Information for each mine included in our dataset.

In this report, we will focus on images of these 5 mines taken in 2020, show in Figure 6.

Unknown
Alberta, Canada

Tagebau Garzweiler
Jüchen, Germany

Grasberg Mine
Central Papua, Indonesia

Mir Mine
Mirny, Russia

Chuquicamata
Calama, Chile

Figure 6: Satellite photos of 5 mines taken in 2020

Mine Identification

To identify mines, we tested five different clustering algorithms: agglomerative clustering, k-means, Gaussian mixture model, and density-based spatial clustering of applications with noise (DBSCAN). K-means was implemented from scratch, but the remaining methods were implemented using MATLAB functions.

Each image was considered to be a “dataset” with each pixel as a data point. We generally included 5 dimensions: x position, y position, hue, value, and saturation. For hue and saturation, we increased the contrast by multiplying the respective channels by 2. We then used MatLab functions to implement these algorithms on the 2020 image of each of the five mines.

Edge and Contour Detection

Hough Transform

The initial approach to edge detection involved implementing the Hough Transform alongside other built-in MATLAB functions. The Hough Transform is a widely used feature extraction technique in image processing that identifies straight lines from an image's edges. It does so by mapping the identified lines into a 2D parameter space, with the axes representing the two parameters defining the straight line.

In our implementation, we pre-processed the test images to improve edge detection. This included enhancing image contrast using the `imadjust()` function, removing small background areas with the `bwareafilt()` function, and detecting edges using the `edge()` function.

Active Contour

1. **Converting the image into a gradient map**
   The gradient map serves as the "force" field that determines how the contour stretches to resemble the object's actual boundary. To ensure that the gradient map works effectively even when the initial contour is not close to the object, a Gaussian filter is applied to blur the gradient map, distributing the "forces" created by sharp edges.

   Figure 7 below shows the original image and its gradient map, both before and after the Gaussian filter is applied. Here, the gradient map has magnitudes equal to the square of the initial values.

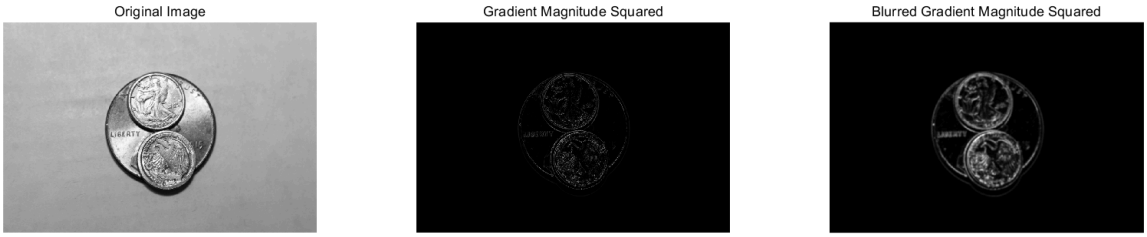


Figure 7: Ideal test image (a) and its gradient map before (b) and after (c) the Gaussian filter is applied

2. **Plotting the Initial Contour**
   The contour is constructed as a list of points on the image, with their xx and yy coordinate values forming a matrix. During the boundary-searching process, these discrete points are iteratively moved toward the object's actual boundary using the greedy algorithm.

   In terms of location, the initial contour must be plotted around the target object without being too far from the object's boundary, as shown in Figure 8. This ensures that the gradient map effectively influences the movement of the contour points and that the contour points are not "out of reach" of the "peak" forces on the gradient map.

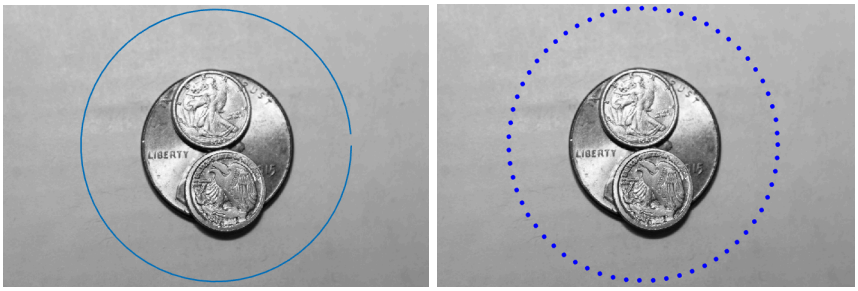
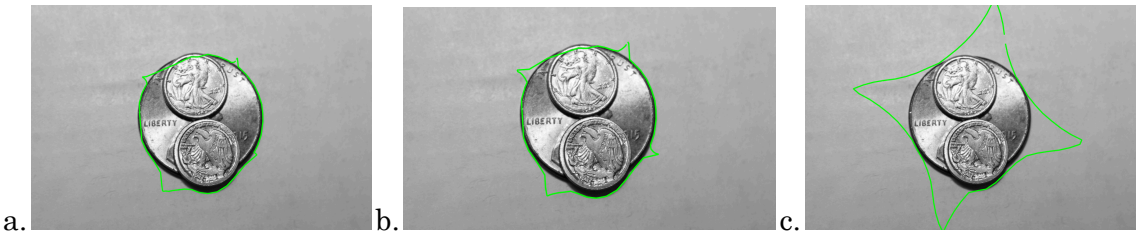


Figure 8: Initial contour as an array of points applied to the image

3. Fine-tuning parameters

Alpha (α) and **Beta (β)** are parameters that must be fine-tuned for each case to optimize the performance of the active contour technique. The graphs in Figure 9 below illustrate how these parameters affect the behavior of the contour.



a.

Alpha (weight of E_elastic)	0.95
Beta (weight of E_smooth)	0.05

b.

Alpha (weight of E_elastic)	0.4
Beta (weight of E_smooth)	0.05

c.

Alpha (weight of E_elastic)	0.95
Beta (weight of E_smooth)	0.95

Figure 9: Effects of α and β parameters with test image with alpha and beta corresponding to the values in the table

The contour in both images result from 100 iterations. The one on the left has less sharp corners, aligning to the theory that larger alpha resulting in the contour to perform more like a rubber band. The image on the right has a high E_soomth weight.

4. **Implementing the Iterative Process**

   For the local window around each contour point, the window size is set to 2 to balance computational expense and refinement. This results in a local window with dimensions of 5×5.

   The smaller the window, the more refined the steps of the contour points during the iterations. However, this also requires more computational power.

   In my implementation, Two nested for loops are used to move each contour point during every iteration. Using the formula above, E_tot is calculated for every pixel within the local window at each contour point. The pixel with the minimum E_tot value becomes the new contour point in the next iteration, representing the "shift" of the contour point toward the target object boundary.

Results

Agglomerative Clustering

We used the `linkage()` and `cluster()` functions in MatLab's statistics and machine learning. The MatLab implementation starts with the `linkage()` function, which returns a matrix that encodes a tree containing hierarchical clusters. We used the 'average' method for the distance between points. It is an unweighted average distance, which is defined by the equation

$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri} x_{sj})$$

Where r is a cluster, n_r is the number of objects in cluster r; s is the closest cluster, n_s is the number of objects in cluster r; x_{ri} is the i-th object in cluster r; and x_{sj} is the j-th object in cluster s.

From here, we used the `cluster()` function to generate a specified number of clusters. The function takes the hierarchical cluster matrix (the output of the `linkage()` function) and a cluster number. We swept several cluster numbers to determine which one most effectively clustered each image.

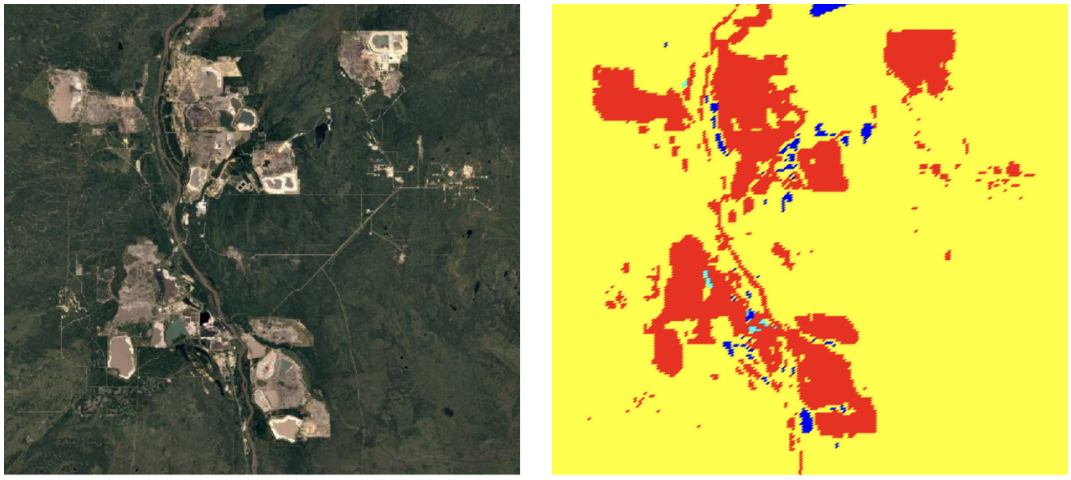


Figure 10: Clustering of the Canadian mine with 4 clusters. The mine was able to accurately identify all of the major areas of the mine and some of the smaller areas. However, it could not identify any of the road networks. The algorithm also began to identify bodies of water (dark blue), likely because the uniformity in color creates a strong cluster.

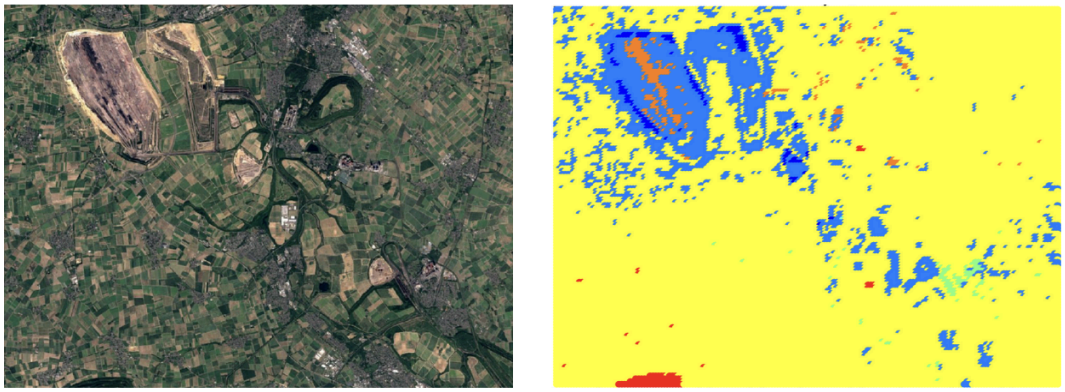


Figure 11: Clustering of the Garzweiler mine with 7 clusters. While the clustering identified the largest mine, it could not filter out the noise from patches of similarly colored farmland.

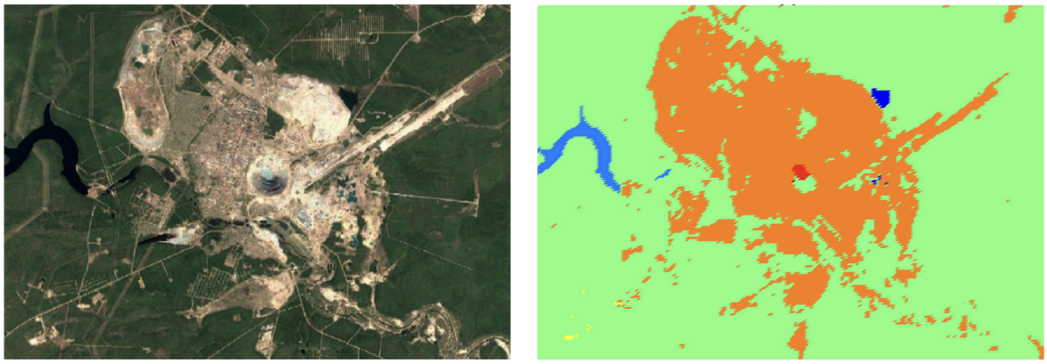


Figure 12: Mirny Mine with 8 clusters. While the algorithm was able to identify the mine, it could not distinguish it from the surrounding town. Furthermore, it consistently clusters the bodies of water because of high contrast and uniform color.

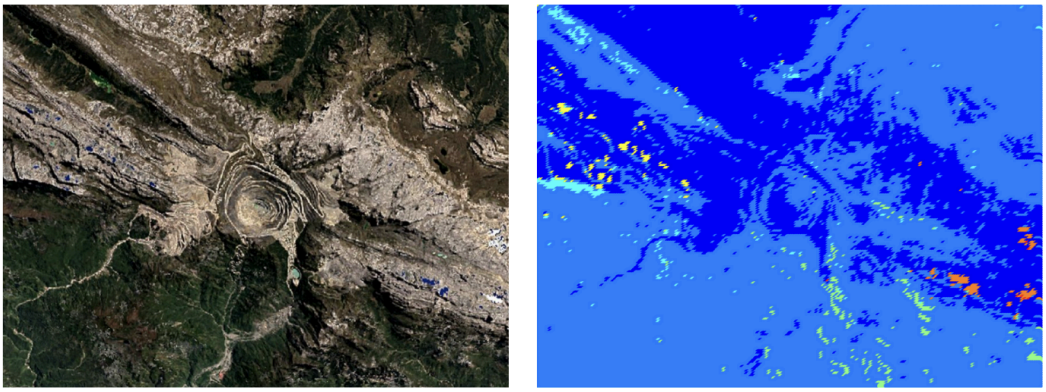


Figure 13: Grasberg mine with 7 clusters. The algorithm performed poorly on this mine. Because of noise from geographic features, there is not much contrast between the mountains and the mine itself.

In general, this method created clusters that contained the mines. However, it had difficulty with noise, low color contrast, and processing speed. We had to limit our image dimensions to a factor of 0.15 (just 2.3% of the original image size) for the `linkage()` function to run without exceeding the maximum array size. Therefore, this algorithm works in simple cases but other methods proved to be more effective. They identify mines more precisely with fewer computational resources.

K-Means

The results of our custom K-Means function with the selected mine images are shown below. For each image, we set the algorithm to find 3 clusters in the HSV Euclidean space. Results are shown in Figure 14, Figure 15, and Figure 16.

a. b.

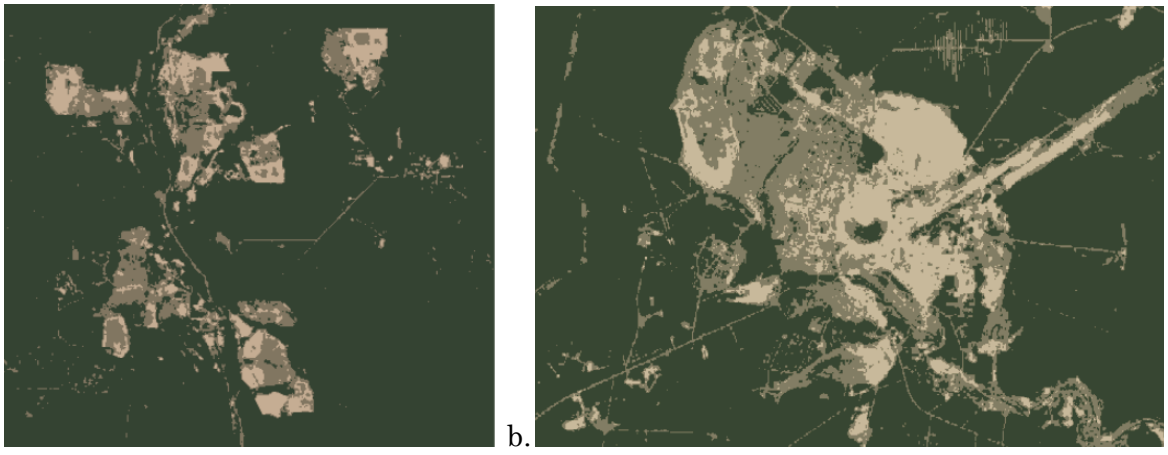


Figure 14: (a) K-Means segmentation of the Alberta mine. This method was very successful in separating the background in one cluster and the mine and roads into the other two clusters due to the clear separation of, in particular, the hue and value. (b) K-Means segmentation of the Mirny mine. This method was successful in separating the mine, city, and roads from the other terrain. It

was also partly successful in separating the mine (lighter tan) and city (darker tan) area into two different clusters, though with some inaccuracy.

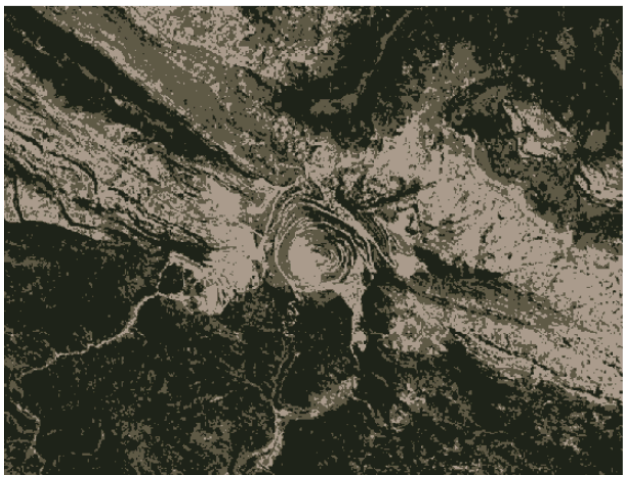
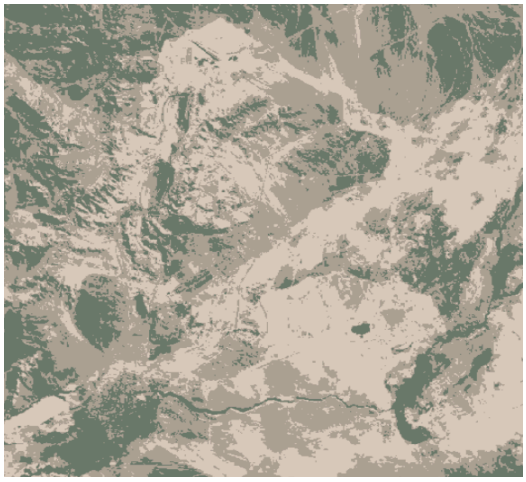
a.  b. 

Figure 15: (a) K-Means segmentation of the Grasberg mine. This method was not successful in isolating the mine, because there was little color variation between it and the surrounding rock. (b) K-Means segmentation of the Chuquicamata mine. This method was not successful in isolating the mine, because there was little color variation between it and the surrounding rock.

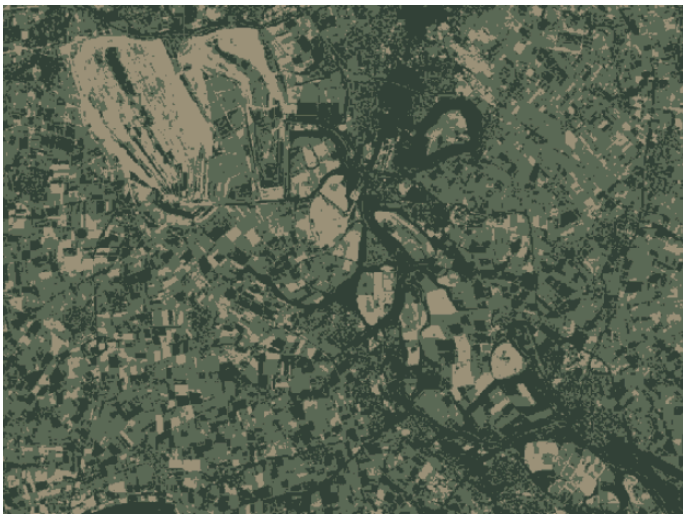


Figure 16: K-Means segmentation of the Garzweiler mine. This method was not successful in isolating the mine, because much of the surrounding farmland was a similar color to the mine.

Overall, K-Means was most useful in detecting mines with consistent coloration that was not present in the background terrain. Low contrast and/or noisy imagery made this method less effective. It was also not helpful in differentiating any features not based on color, like roads and textures.

Gaussian Mixture

The results of the Gaussian Mixture function with the selected mine images is shown below. For each image, we set the algorithm to find 2 clusters in the HSV Euclidean space. Results are shown in Figure 17, Figure 18, and Figure 19.

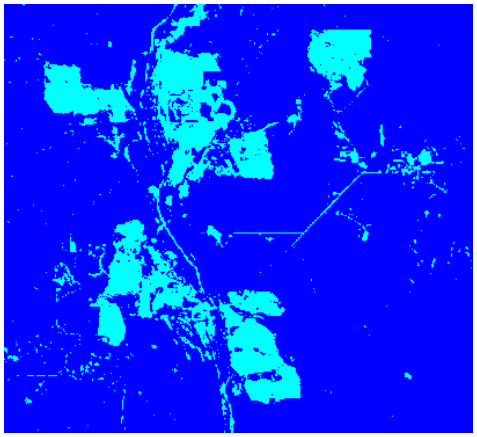
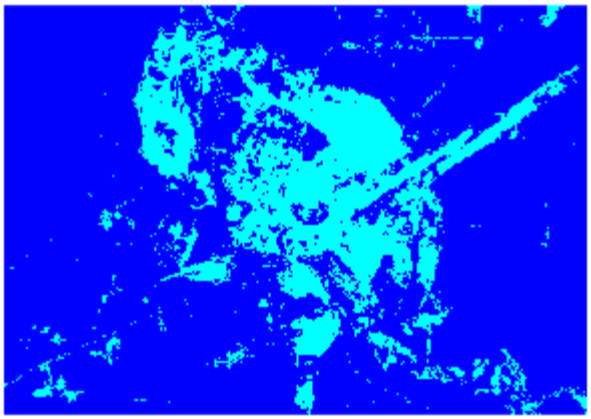
a.  b. 

Figure 17: (a) Gaussian mixture segmentation of the Alberta mine. This method was very successful in separating the background in one cluster and the mine and parts of the roads into one other cluster. (b) Gaussian mixture segmentation of the Mirny mine. This method was also relatively successful in separating the mine, city, and roads from the other terrain.

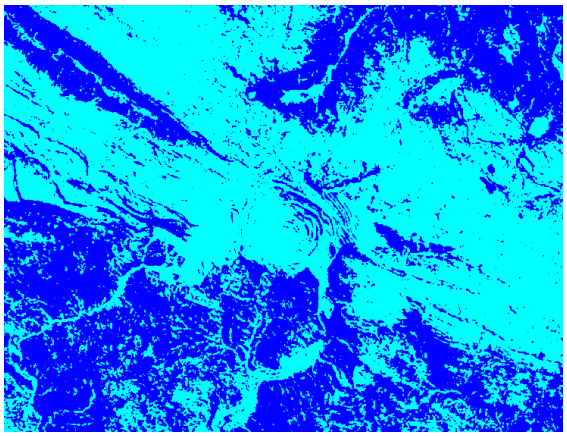
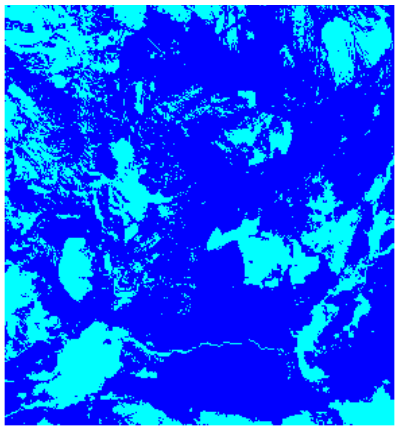
a.  b. 

Figure 18: Gaussian Mixture segmentation of the Grasberg mine. This method was not successful in isolating the mine.(b) Gaussian Mixture segmentation of the Chuquicamata mine. This method was not successful in isolating the mine, because there was little color variation between it and the surrounding rock.

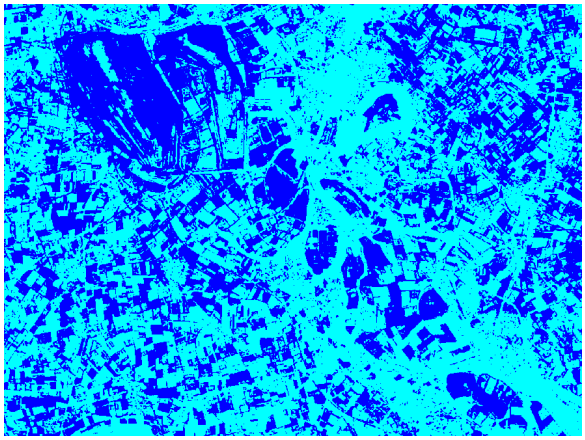


Figure 19: Gaussian Mixture segmentation of the Garzweiler mine. This method was partially successful in segmenting the mine but included much of the similarly colored farmland and large areas in the same cluster.

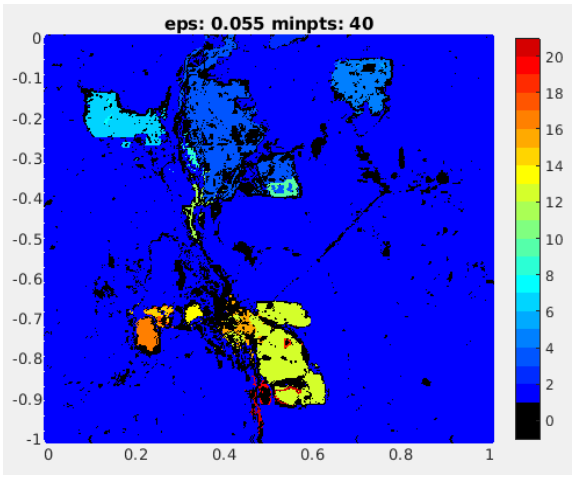
Overall, Gaussian Mixture segmentation of the mines functioned in a very similar manner as the K-Means segmentation, doing well on images with distinct coloration differences in the HSV space between the mines and the backgrounds, and not working as well on images with low contrast and significant background noise. Due to its nature in this implementation of making predictions solely on pixel HSV values and not pixel location, it was really good at picking up roads which most other models struggled to do, at the cost of also being highly noise-sensitive.

For the future, to try to improve the segmentation accuracy of this model, a couple things we have considered trying are as follows:

1. Refine Initialization
    a. The Gaussian Mixture Model that we used for this project implemented a random initialization of means and covariances which it then improved on in the expectation and maximization steps. However, initializing the Gaussian Mixture Model with means determined from using the K-Means algorithm or means and covariance matrices from Agglomerative Clustering could improve results.
2. Apply Spatial Regularization
    a. The Gaussian Mixture Model that we used for this project operates solely on pixel HSV values, ignoring spatial awareness. To encourage clusters to group in spatially continuous regions, we could include spatial coordinates in our feature matrix. We predict that this could be particularly helpful in images that don't contain as much color variation or have backgrounds that contain noise, such as in the Garzweiler mine, that the model currently doesn't do a good job of filtering out.

DBSCAN

The results of MATLAB's DBSCAN function with the selected mine images is shown below. For each image, we set the algorithm to find clusters in the HSV and x and y position Euclidean space. Results are shown in Figure 20, Figure 21, and Figure 22.

a. 

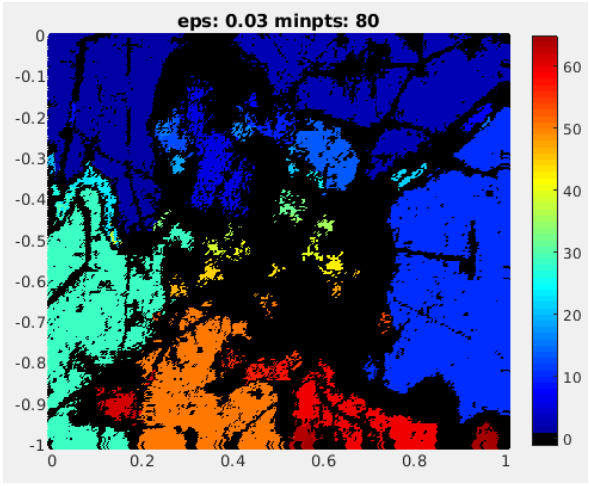
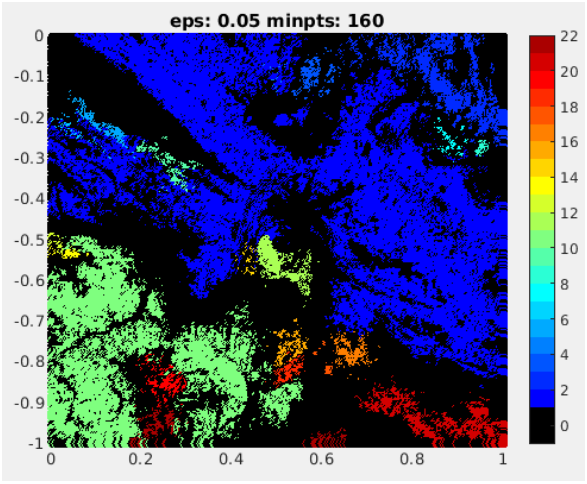
b. 

Figure 20: (a) DBSCAN segmentation of the Alberta mine with epsilon=0.055 and minimum points=40. This method was very successful in identifying the area of the mine and was additionally capable of separating out the different sections. (b) DBSCAN segmentation of the Mirny mine with epsilon=0.03 and minimum points=80. This method was successful in separating the urbanized areas from their surroundings. The most useful method in separating the mine from the city was by tuning the parameters to treat the higher frequency changes in the mine terrain as outliers, shown in black. This is an unconventional use of DBSCAN, as the outlier category is the only one responsive to frequency. This use case suggests that it would be worth exploring clustering by texture for this example.

a. 

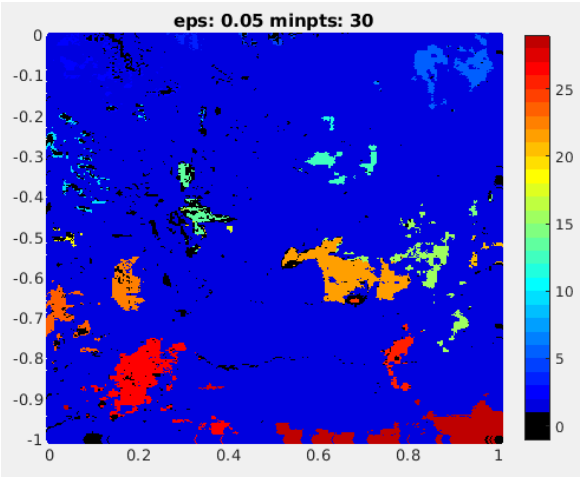
b. 

Figure 21: (a) DBSCAN segmentation of the Grasberg mine with epsilon=0.05 and minimum points=160. This method did mark a group separate from others in the center of the photo where the mine is located, though there were other, larger groups segmented by variation in the rocky

surroundings, and it would be difficult to call out the mine, specifically. This is another instance of taking advantage of the nature of the outlier group, indicating a texture-based clustering may be a better choice. (b) DBSCAN segmentation of the Chuquicamata mine with epsilon=0.05 and minimum points=30. This method was successful in segmenting the mine (shown in light orange) from the background, though it did not mark the entire area and there were several other groups unrelated to the mine identified due to irregularities in the rock.

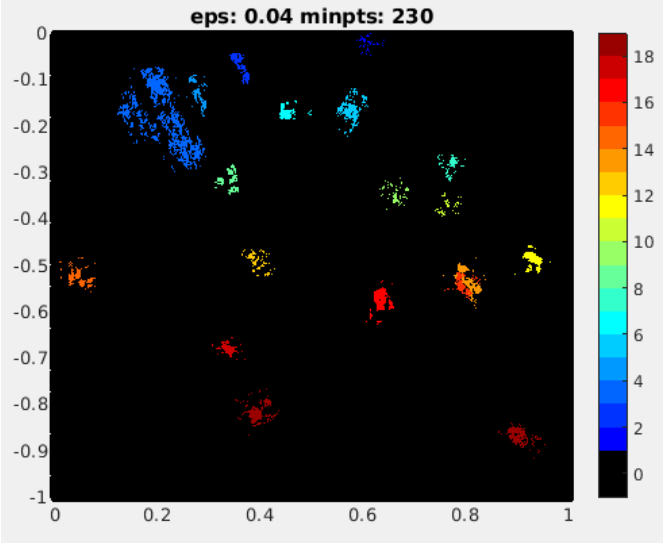


Figure 22: DBSCAN segmentation of the Garzweiler mine with epsilon=0.04 and minimum points=230. This method successfully created a large cluster of points at the main location of the mine, however there were several other places highlighted. Additionally, the algorithm did not segment the entire area of the mine, so it would be difficult to use area estimation here.

Hough Transform

Applications of the Hough transform are limited for several reasons: a) Most roads are not straight lines, which poses a challenge for the Hough Transform, as it is primarily designed to detect straight lines by fitting data points using two parameters. b) Vegetation and buildings often border roads, making it difficult to distinguish the road's contour from the surrounding environment. c) Smaller roads, when converted into edge images, sometimes appear discontinuous or fragmented, therefore they could potentially be removed during the pre-processing stage.

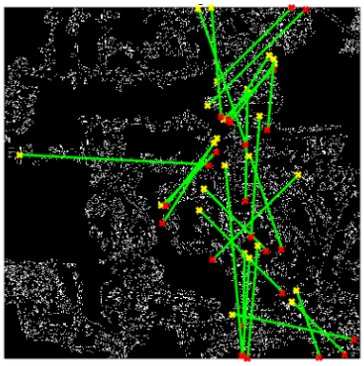
a.  b. 

Figure 23: Testing results of the Hough Transform method on the Massachusstes road satellite image are shown: (a) the original image, and (b) the detected roads marked in the edge image. As observed, only the two main roads are detected, with many false discontinuous marker lines.

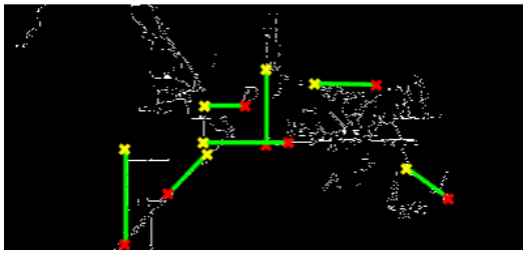
a.  b. 

Figure 24: Testing results of the Hough Transform method on the Alberta mine satellite image are shown: (a) the original image, and (b) the detected roads marked in the edge image. As observed, the main straight roads are partially detected in certain areas.

Active Contour

To ensure the implemented code works, I first tested it in an ideal scenario where the image contains a single clear object with high contrast against the background. This confirmed that the code performed as expected, successfully approaching the target boundary after 100 iterations.

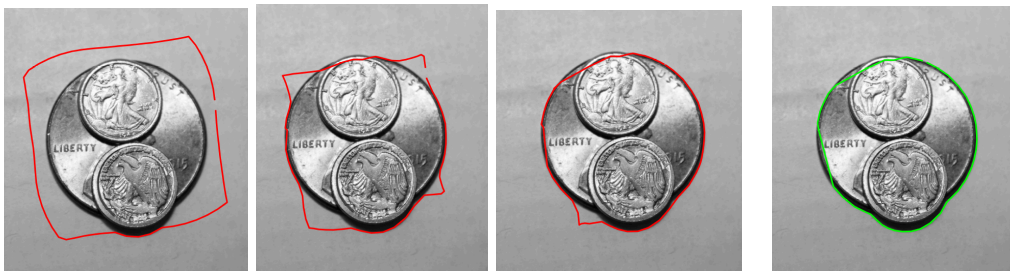


Figure 25: Results of 100 iterations on the test image

Next, I tested the algorithm using several images of mining sites extracted from real-world GPS databases. During this process, numerous issues were identified, both in the technical aspects of the algorithm and the practical usability of the script.

Figure 26: Results of 100 iterations on the Garzweiler mine

Testing the "Garzweiler" mine image from 1990 shows that the algorithm is ineffective at fitting the mine area when distractions are present around the mine site. The final image, after 100 iterations, yields a result no better than that at the 60th iteration.

Another test was conducted on a mine site in Alberta, Canada, in 1984.

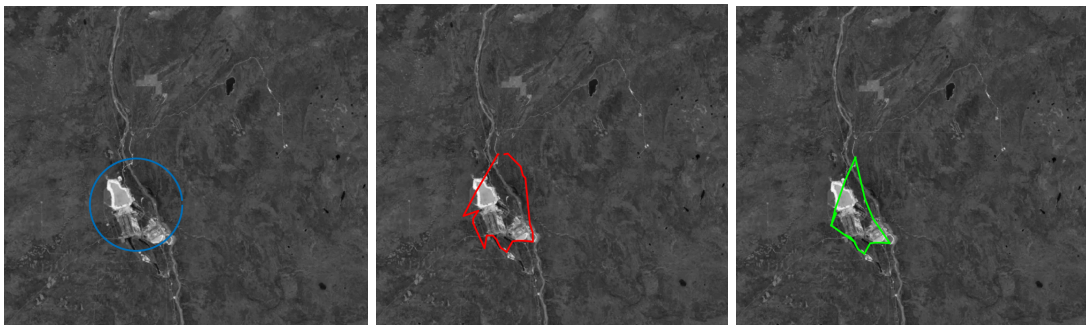


Figure 27: Results of 100 iterations on the Alberta mine

Similarly, the algorithm fails to draw the boundary around the mining site, despite the visually clear color contrast. To understand why the final boundary has a sharp edge at the top, the contour is plotted over the gradient map.

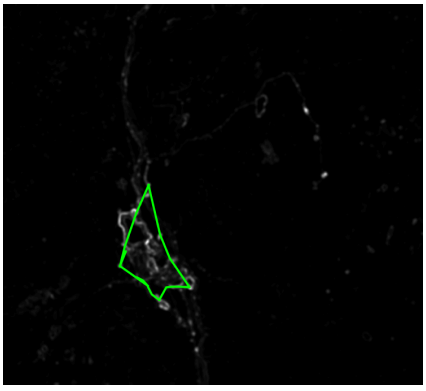


Figure 28: Contour plotted on Alberta mine gradient map

This reveals that the contour is likely influenced by the gradient of the road, resulting in a "compromised" location between the road gradient and the mining site gradient.

The effect of the contour being influenced by the road near the target object is also evident in the example below. After 100 iterations, the initial contour circle fits the overall shape of the roads around the mining site rather than the mining area itself.

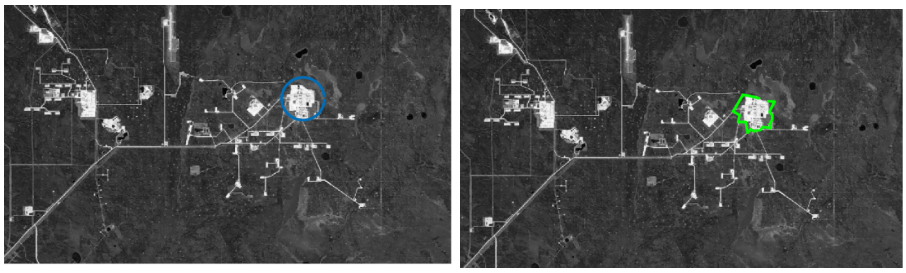


Figure 29: Contour of roads near the mining area

Limitations

In addition to the evident inaccuracy in matching the target boundary, another limitation of this method is its lack of generalization. The parameters that control smoothness and elasticity, as well as the initial contour circle, must all be manually configured for each case. While this approach may be useful if the user knows the approximate location of the object they are trying to contour, it becomes challenging when the contour is arbitrarily placed on the image. The method is also highly sensitive to noise and easily distracted by gradients outside the target area.

Conclusion

This project compared different methods of non-ML image segmentation methods. We got to explore clustering methods in depth and discovered their relative advantages and disadvantages. The ways in which some methods did and didn't work also revealed insights into possible future expansions into segmentation of satellite images. For example, the outlier category of the DBSCAN method revealed the influence of texture in addition to HSV and position in some images.

If we were to redo this project, we might be more deliberate with our choices of methods and explore preprocessing steps before segmentation. It also would have been nice to start with a clearer scope and definition of the project as a whole.

In the future, it would be interesting to explore other methods of edge detection. We are also interested in the less technical aspects of the project, such as linking our analysis to the historical demand for natural resources.

Code

The code for this project lives in this repo. The algorithms run in these files:

- K-Means: k_means_all.mlx
- Gaussian Mixture: final_gaussian_mixture_model.mlx
- DBSCAN: dbscan_all.m
- Agglomerative clustering: agglom.mlx

All images are in fa24-imageprocessing/final/images/.

References

Eyes on Nature: How Satellite Imagery Is Transforming Conservation Science. (2017). *Eyes on Nature: How Satellite Imagery Is Transforming Conservation Science*. Yale E360. https://e360.yale.edu/features/eyes-on-nature-how-satellite-imagery-is-transforming-conservation-science

Kumar, M. (n.d.). *Gaussian Mixture Model in Image Processing Explained*. CronJ. Monu Kumar

Mittal, H., Pandey, A. C., Saraswat, M., Kumar, S., Pal, R., & Modwel, G. (2021). A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-021-10594-9

Scikit-Learn Developers. (2024). *Comparing different clustering algorithms on toy datasets*. Scikit Learn. https://scikit-learn.org/1.5/auto_examples/cluster/plot_cluster_comparison.html

Yang, M., Mei, H., & Huang, D. (2017). AN EFFECTIVE DETECTION OF SATELLITE IMAGES VIA K-MEANS CLUSTERING ON HADOOP SYSTEM. *International Journal of Innovative Computing, Information and Control ICIC International, 2017*, *13*(3), 1037–1046. http://www.ijicic.org/ijicic-130323.pdf

Zhang, Z., Liu, Q., & Wang, Y. (2018). Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, *15*(5), 749–753. https://doi.org/10.1109/LGRS.2018.2802944

List of Figures

and roads from the other terrain. It was also partly successful in separating the mine (lighter tan) and city (darker tan) area into two different clusters, though with some inaccuracy.

Figure 15: (a) K-Means segmentation of the Grasberg mine. This method was not successful in isolating the mine, because there was little color variation between it and the surrounding rock. (b) K-Means segmentation of the Chuquicamata mine. This method was not successful in isolating the mine, because there was little color variation between it and the surrounding rock.

Figure 16: K-Means segmentation of the Garzweiler mine. This method was not successful in isolating the mine, because much of the surrounding farmland was a similar color to the mine.

Figure 17: (a) Gaussian mixture segmentation of the Alberta mine. This method was very successful in separating the background in one cluster and the mine and parts of the roads into one other cluster. (b) Gaussian mixture segmentation of the Mirny mine. This method was also relatively successful in separating the mine, city, and roads from the other terrain.

Figure 18: Gaussian Mixture segmentation of the Grasberg mine. This method was not successful in isolating the mine.(b) Gaussian Mixture segmentation of the Chuquicamata mine. This method was not successful in isolating the mine, because there was little color variation between it and the surrounding rock.

Figure 19: Gaussian Mixture segmentation of the Garzweiler mine. This method was partially successful in segmenting the mine but included much of the similarly colored farmland and large areas in the same cluster.

Figure 20: (a) DBSCAN segmentation of the Alberta mine with epsilon=0.055 and minimum points=40. This method was very successful in identifying the area of the mine and was additionally capable of separating out the different sections. (b) DBSCAN segmentation of the Mirny mine with epsilon=0.03 and minimum points=80. This method was successful in separating the urbanized areas from their surroundings. The most useful method in separating the mine from the city was by tuning the parameters to treat the higher frequency changes in the mine terrain as outliers, shown in black. This is an unconventional use of DBSCAN, as the outlier category is the only one responsive to frequency. This use case suggests that it would be worth exploring clustering by texture for this example.

Figure 21: (a) DBSCAN segmentation of the Grasberg mine with epsilon=0.05 and minimum points=160. This method did mark a group separate from others in the center of the photo where the mine is located, though there were other, larger groups segmented by variation in the rocky surroundings, and it would be difficult to call out the mine, specifically. This is another instance of taking advantage of the nature of the outlier group, indicating a texture-based clustering may be a better choice. (b) DBSCAN segmentation of

the Chuquicamata mine with epsilon=0.05 and minimum points=30. This method was successful in segmenting the mine (shown in light orange) from the background, though it did not mark the entire area and there were several other groups unrelated to the mine identified due to irregularities in the rock.

Figure 22: DBSCAN segmentation of the Garzweiler mine with epsilon=0.04 and minimum points=230. This method successfully created a large cluster of points at the main location of the mine, however there were several other places highlighted. Additionally, the algorithm did not segment the entire area of the mine, so it would be difficult to use area estimation here.

Figure 23: Testing results of the Hough Transform method on the Massachusstes road satellite image are shown: (a) the original image, and (b) the detected roads marked in the edge image. As observed, only the two main roads are detected, with many false discontinuous marker lines.

Figure 24: Testing results of the Hough Transform method on the Alberta mine satellite image are shown: (a) the original image, and (b) the detected roads marked in the edge image. As observed, the main straight roads are partially detected in certain areas.

Figure 25: Results of 100 iterations on the test image

Figure 26: Results of 100 iterations on the Garzweiler mine

Figure 27: Results of 100 iterations on the Alberta mine

Figure 28: Contour plotted on Alberta mine gradient map

Figure 29: Contour of roads near the mining area