Automated 3D Visualisation of 2D Geo-spatial Data

Jack Purkiss

Background

- Geospatial data is a widely used form of data with many applications
- It is assumed that 60-80% of the data available can be interpreted as geodata or spatial data
- Contour lines are a widely used representation of elevation and geospatial information

Contour lines

- Contour lines are continuous lines with a given elevation
- The contour interval is the space between two lines which shows the difference in elevation
- The digital visualisation of elevation is known as a Digital Elevation Model (DEM)

3D visualisation

- The 3D visualisation of contour lines helps provide a more intuitive and immersive representation of the terrain
- It helps to identify trends and patterns
  - Slope, shape, relief
  - Relationship between lines

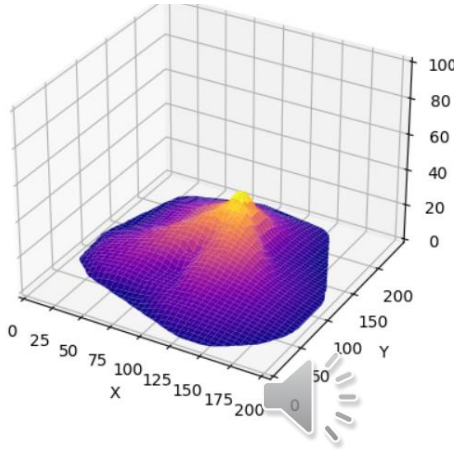


Image-based contours

- GIS models often use contour lines with elevation data assigned directly
  - Allows for highly scalable projects using pre-existing elevation data
- To most people, contour lines are seen in images
- Extracting contour lines from images provides more accessible applications

Example application – Outdoor recreational activities

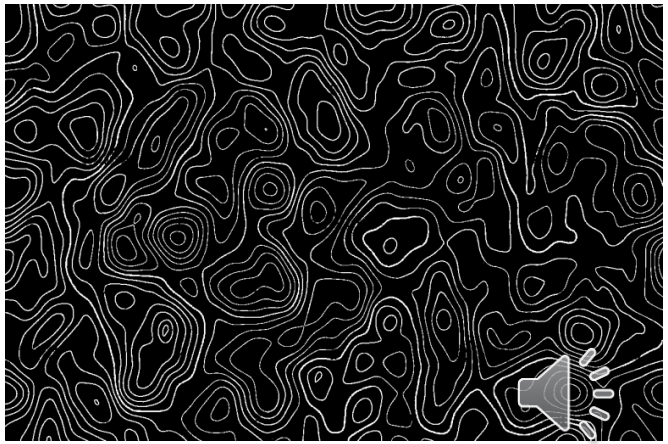
- Having an interactive graphical visualisation of contours can be useful for multiple practical applications
- Outdoor recreational activities such as skiing, hiking or mountaineering
- Can better understand the topography of the terrain
- Image-based extraction would provide more flexibility and accessibility for these areas

Aims and Objectives

- To successfully extract the contour lines from the image
- To generate elevations for the lines
- To create an accurate surface plot a user can interact with
- To automate this whole process

Contour Extraction - Segmentation

- The first step in extracting the contours is segmentation
- A threshold is used for every pixel $g(x,y) = \begin{cases} 1 & if\ f(x,y) > T \\ 0 & otherwise \end{cases}$
- Threshold methods
  - Global, Otsu's, Adaptive



Segmentation – Global Threshold

- Threshold value is provided by the user
- Applies the formula for every pixel in the image
- Not automatic

$$g(x,y) = \begin{cases} 1 & if\, f(x,y) > T \\ 0 & otherwise \end{cases}$$

Segmentation – Otsu's threshold

- Automatic thresholding technique
- Calculates optimal threshold based on the histogram of the image
- Minimises intra-class variance
- Maximise between class variance
- Uses equation:

$$\omega_1(t) = \sum_{i=1}^{t} p_i, \quad \omega_2(t) = \sum_{i=t+1}^{L} p_i$$

$$\mu_1(t) = \frac{\sum_{i=1}^{t} i p_i}{\omega_1(t)}, \quad \mu_2(t) = \frac{\sum_{i=t+1}^{L} i p_i}{\omega_2(t)}$$

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

$$\sigma_1^2(t) = \frac{\sum_{i=1}^{t} (i - \mu_1(t))^2 p_i}{\omega_1(t)}, \quad \sigma_2^2(t) = \frac{\sum_{i=t+1}^{L} (i - \mu_2(t))^2 p_i}{\omega_2(t)}$$

Segmentation – Adaptive threshold

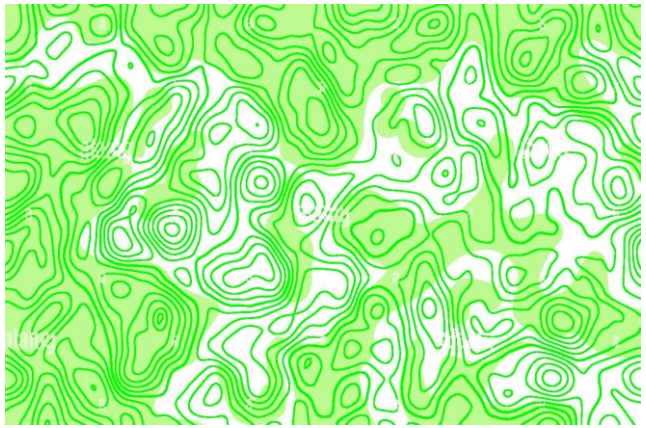
- Automatic thresholding technique
- Threshold used varies across the image based on local pixel intensity variation
- Not best when the image does not vary
- Useful on images with uneven lighting or contrast

$$\text{dst}(x, y) = \begin{cases} \text{maxval} & \text{if } \text{src}(x, y) > \text{T}(x, y) \\ 0 & \text{otherwise} \end{cases}$$

Contour detection

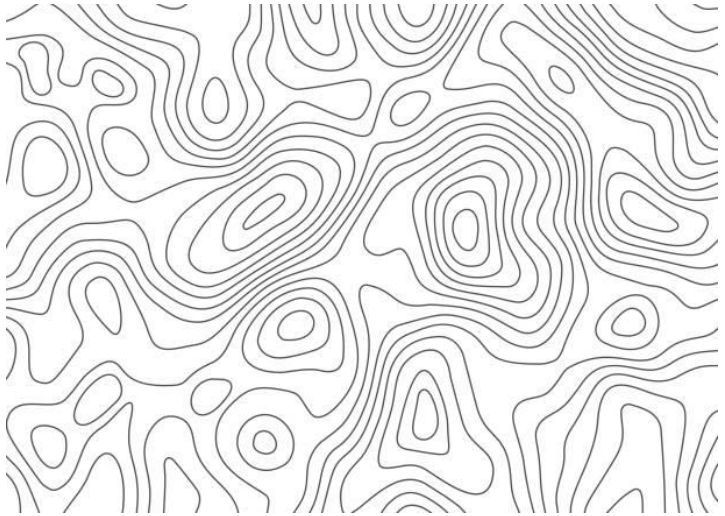
- After segmenting the image to show just the contour lines we have to extract this data
- Use a border-following algorithm to find the contours from the image
  - OpenCV findContours

- Iterate through so there aren’t duplicate lines for borders



DEM Visualisation – Elevation calculation

- We have to find the elevations from the extracted contours
- In theory an iterative process as the elevation increases with internal lines
- Varying peaks and hills make this more challenging



Clustering

- The hills can be grouped by clustering so their elevations can be calculated independently
- Find if a cluster is contained by another to determine its starting elevation

K-Means Clustering

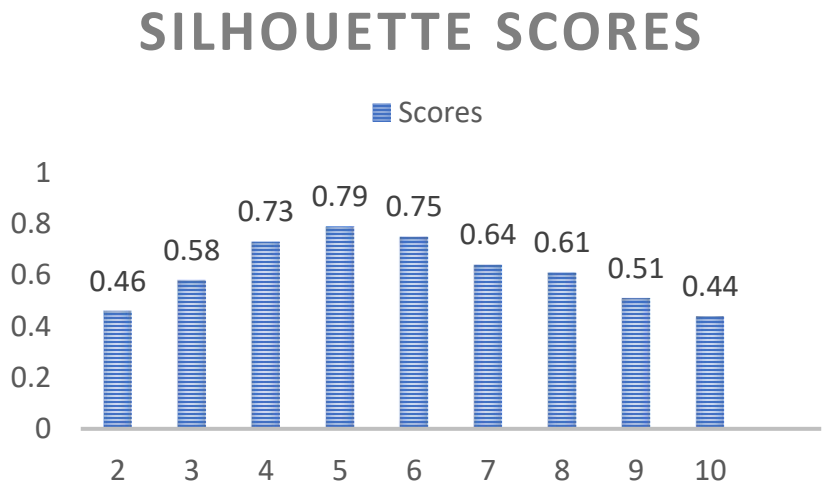
- Groups data points into clusters based on similarity
- Use each contour as a data point and extract its features
  - Area, perimeter, aspect ratio, and centroid coordinates
- Requires a given k number of clusters
  - Can use the silhouette score to find a suggested number

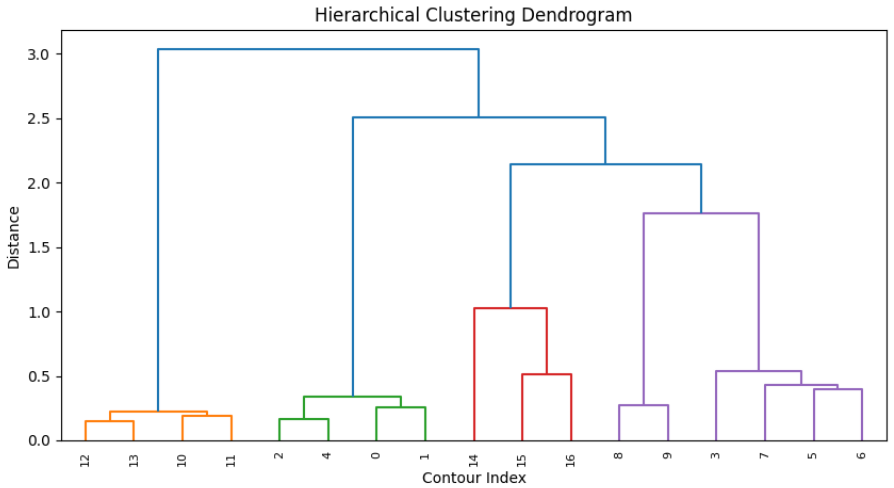
Hierarchical Clustering

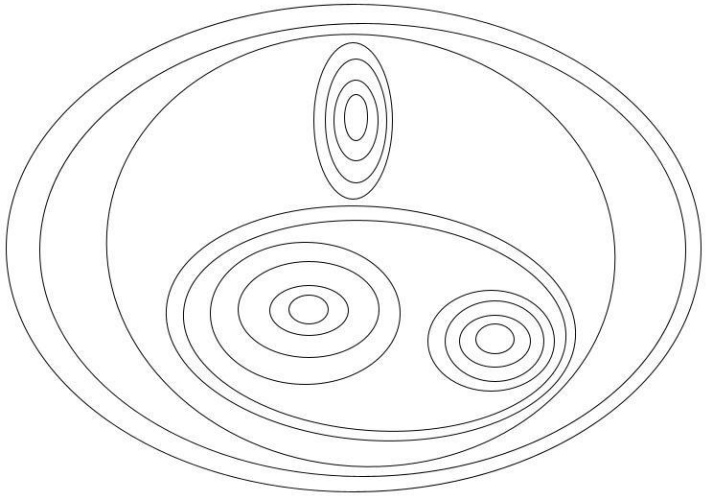
- Automatically determines the cluster number
- Generates a dendrogram and finds a suitable point where the clusters split
- Uses a linkage method to measure the inter-cluster distance
  - Single, Ward, Complete, Average

Clustering example

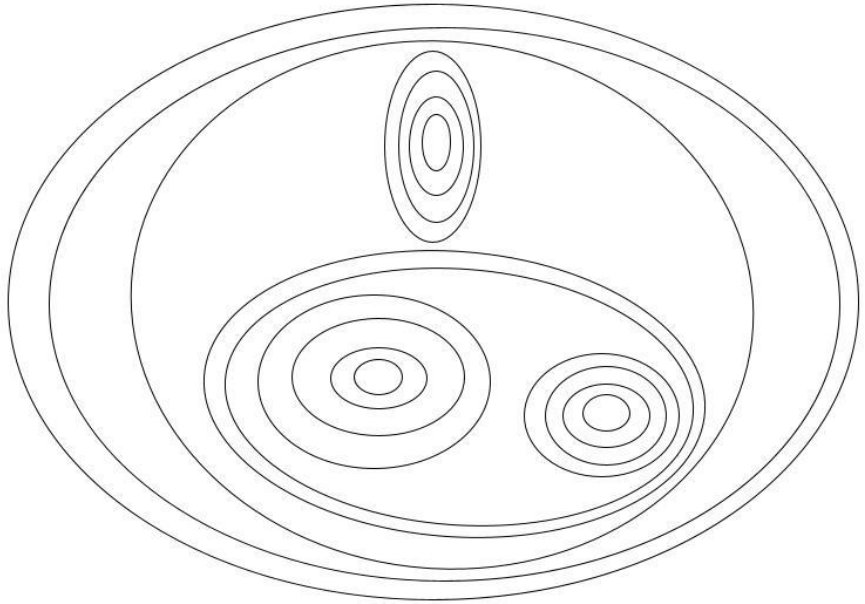
N Clusters	Silhouette Coefficient
2	0.46
3	0.58
4	0.73
5	0.79
6	0.75
7	0.64
8	0.61
9	0.51
10	0.44

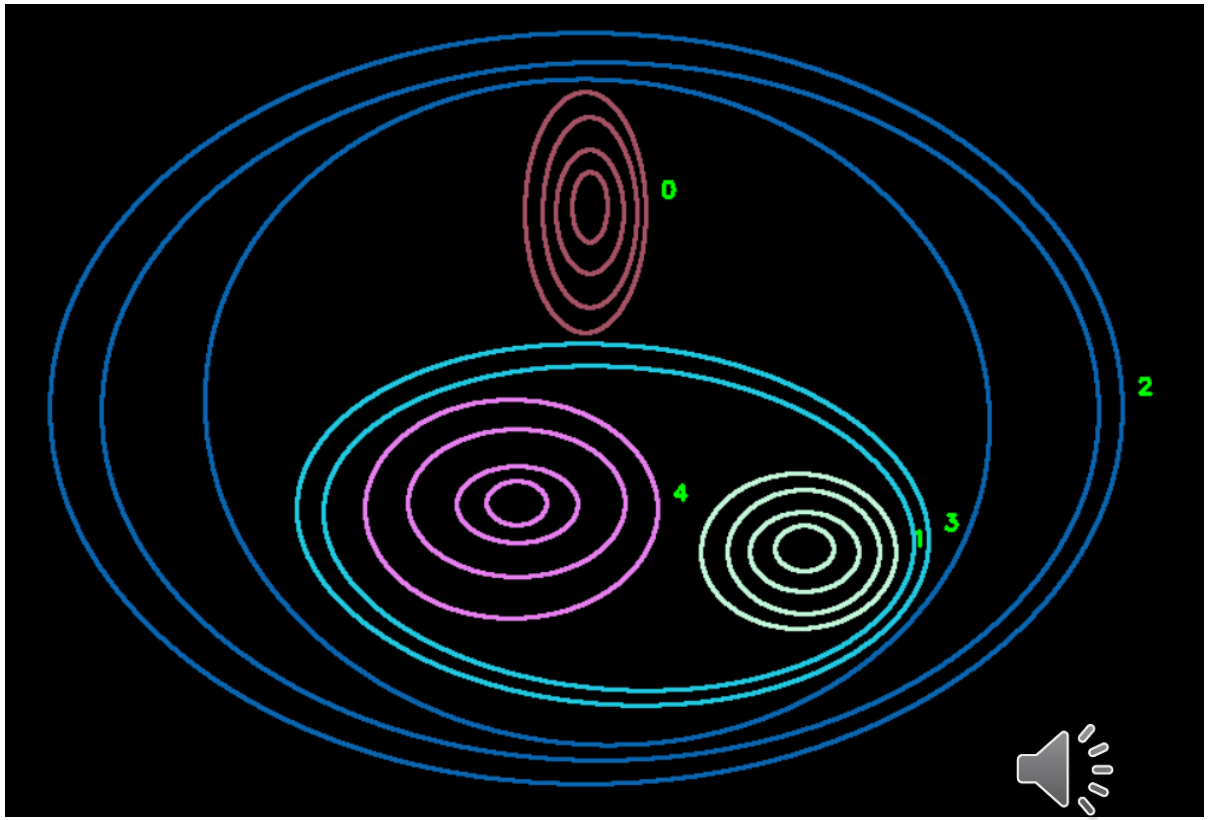






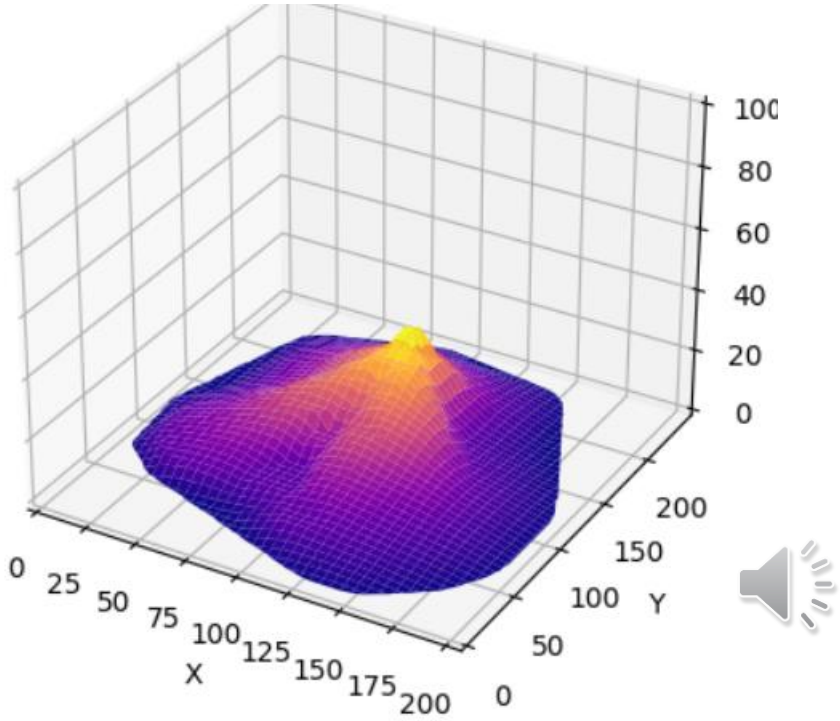
Clustering Example





Surface plot

- Once the elevation data is found from the clustering, it can be plotted
- We create a mesh grid of interpolated height values between the contours
  - scipy.interpolate griddata
- Regular grid and Triangulated surfaces



Results

- Here we are looking at the testing process to determine the best methods to use
- These will be explored in the demo video

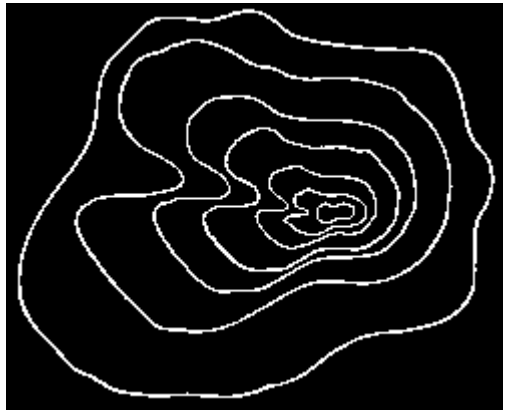
Threshold Results

- To compare threshold methods, we are using the following test cases:
  - Simple undisturbed black and white contour lines
  - Coloured maps with additional information
  - Simple maps with large colour changes
  - Coloured maps with faint contour lines
  - Detailed maps with more complex lines
  - Photographed contour maps with different lighting conditions

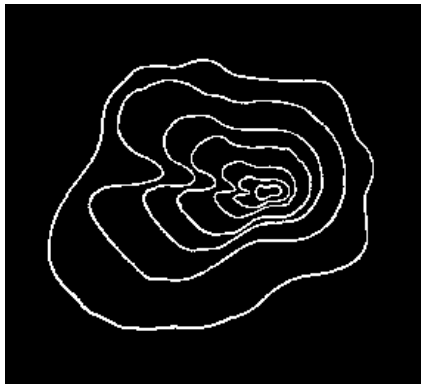
Simple contours

- Otsu’s and Global perform well
- Adaptive doesn’t

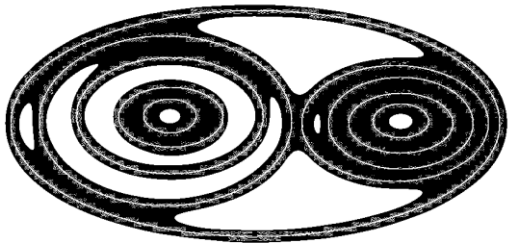
Adaptive



Global

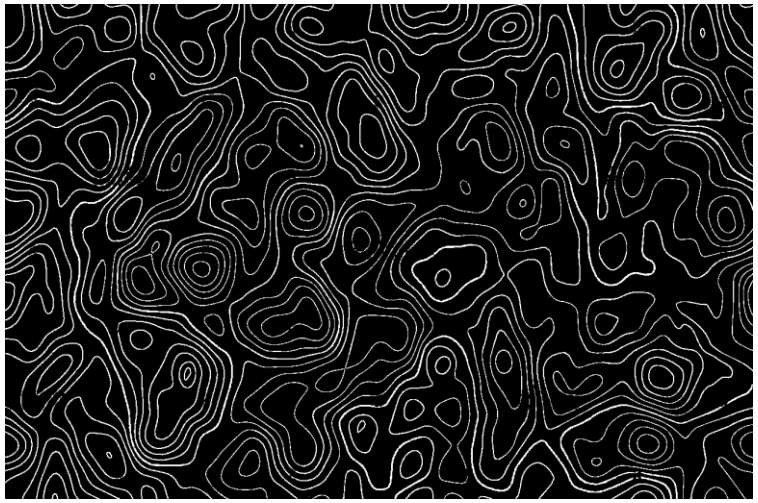


Otsu’s

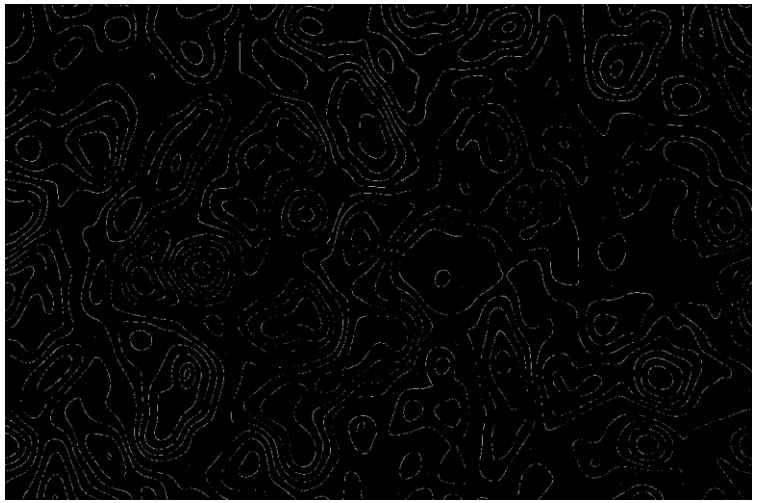


Colour maps with additional info

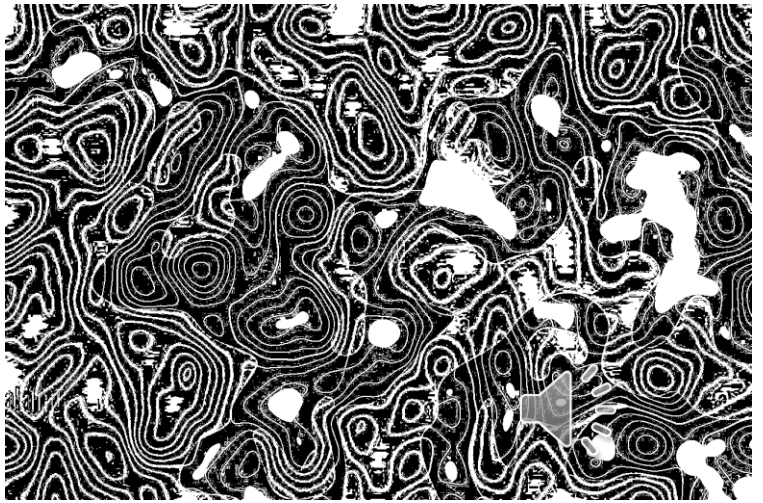
Otsu



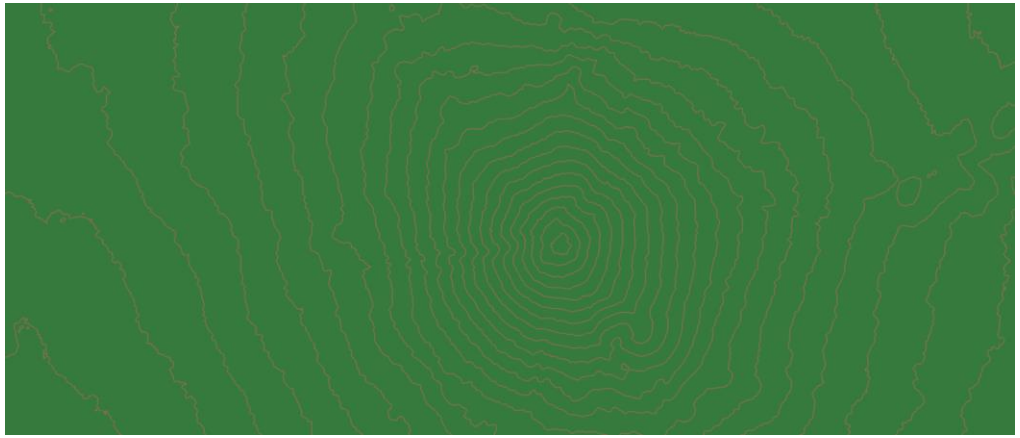
Global



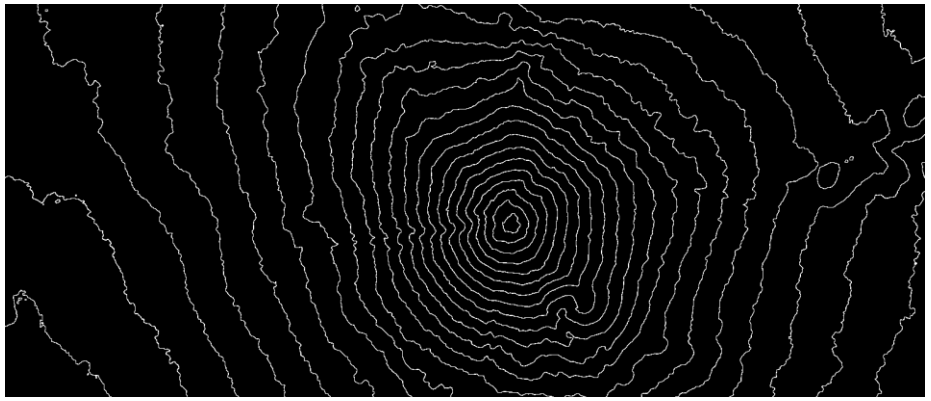
Adaptive



Simple maps with large colour changes

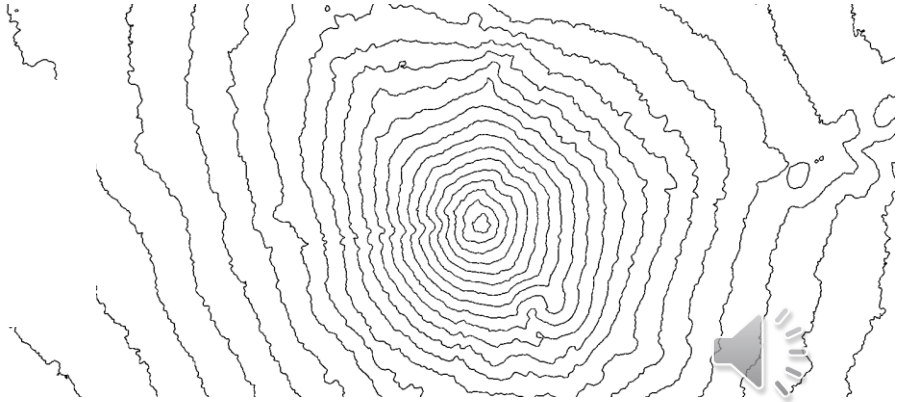


Otsu’s

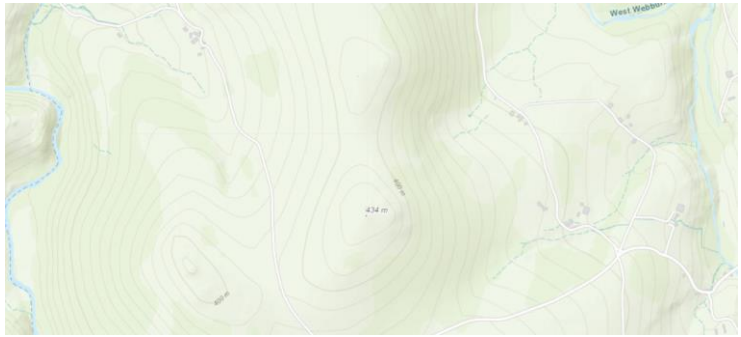


Global

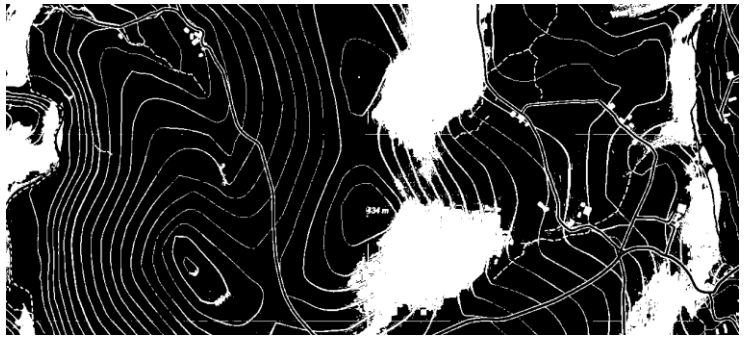
Adaptive



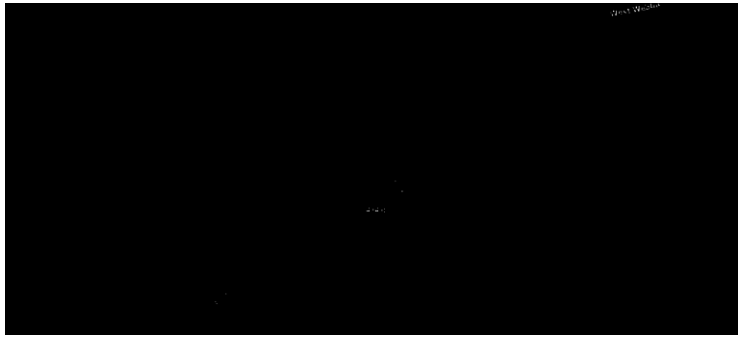
Coloured maps with faint contour lines



Otsu’s

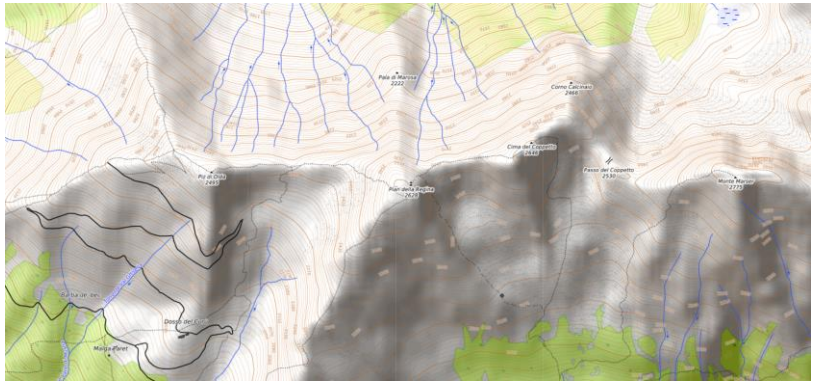


Global

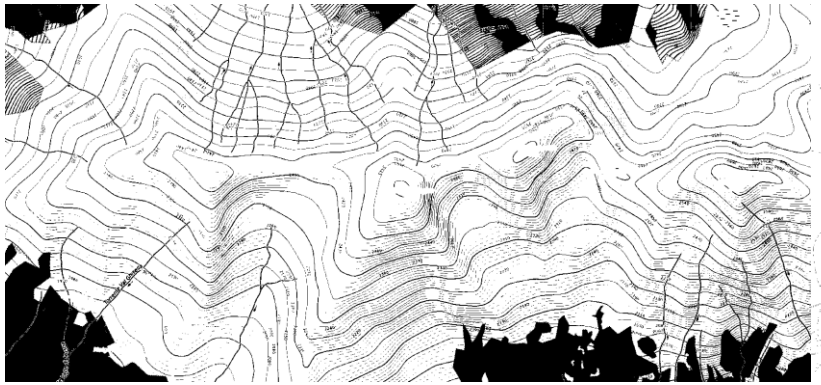


Adaptive

Detailed maps with more complex lines

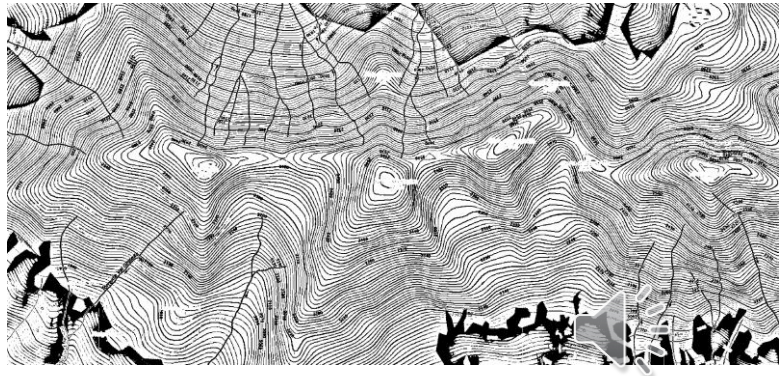


Otsu’s

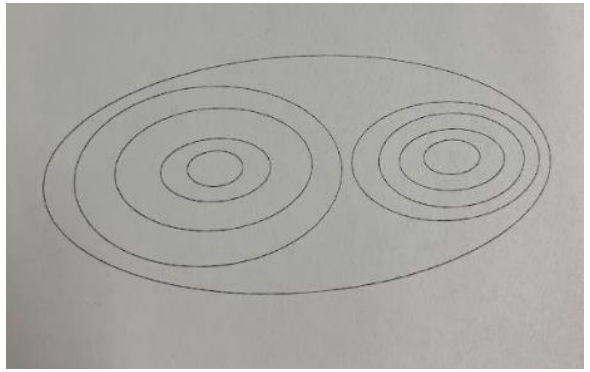


Global

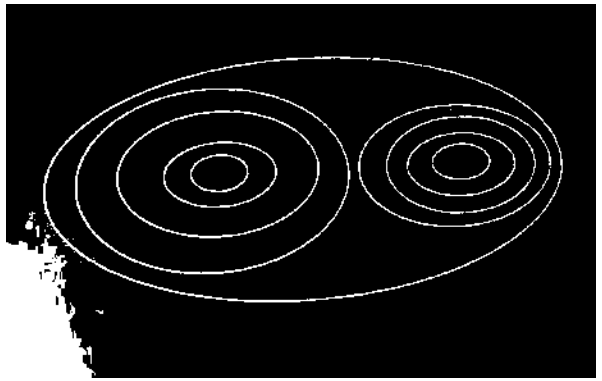
Adaptive



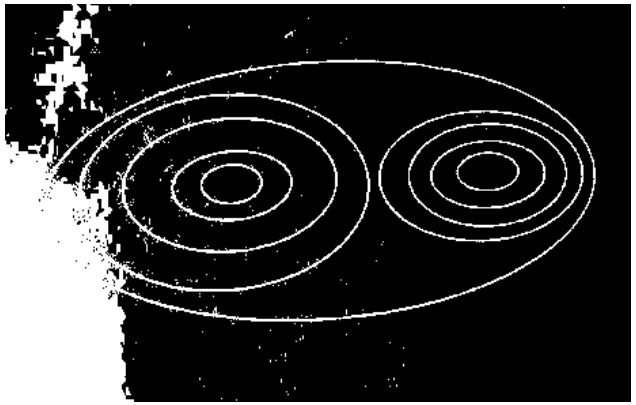
Photographed contour maps with different lighting conditions



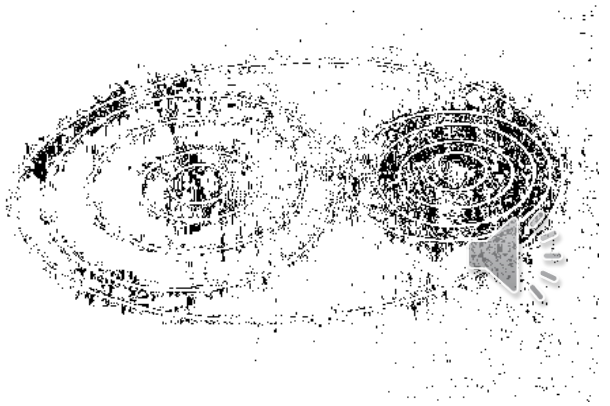
Otsu’s



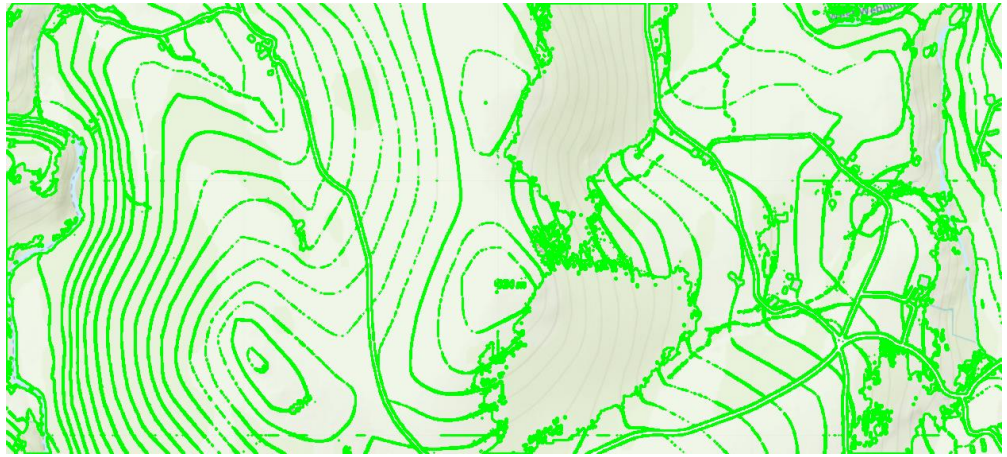
Global

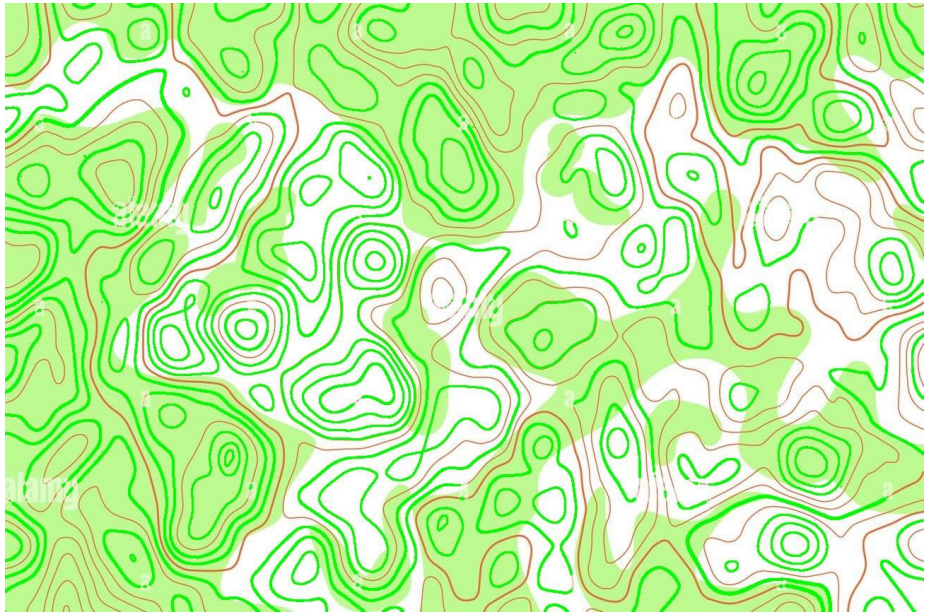


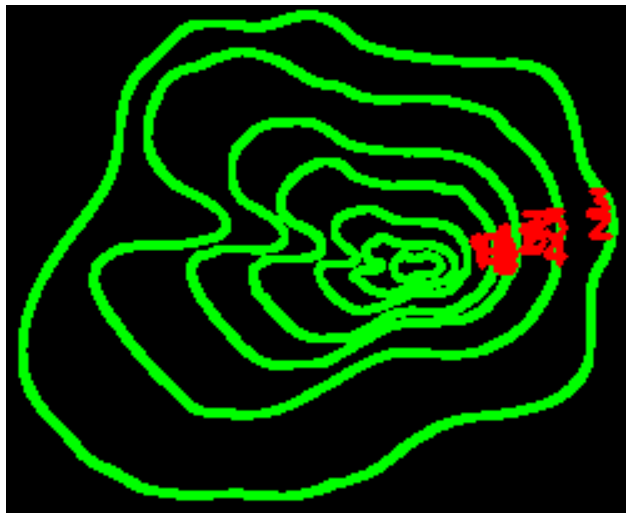
Adaptive



Contour detection results

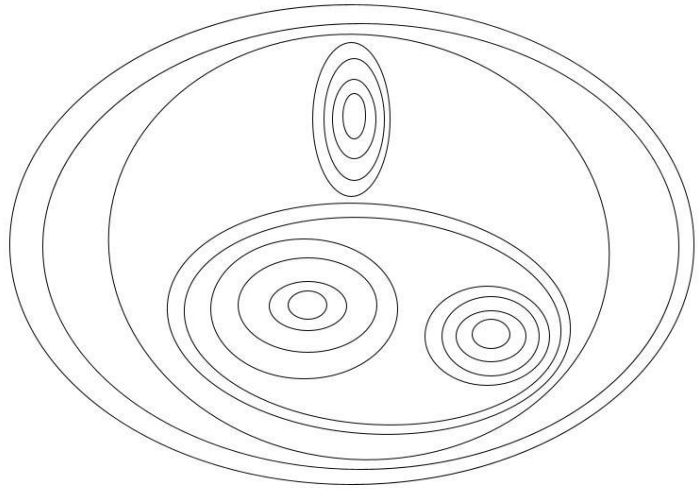


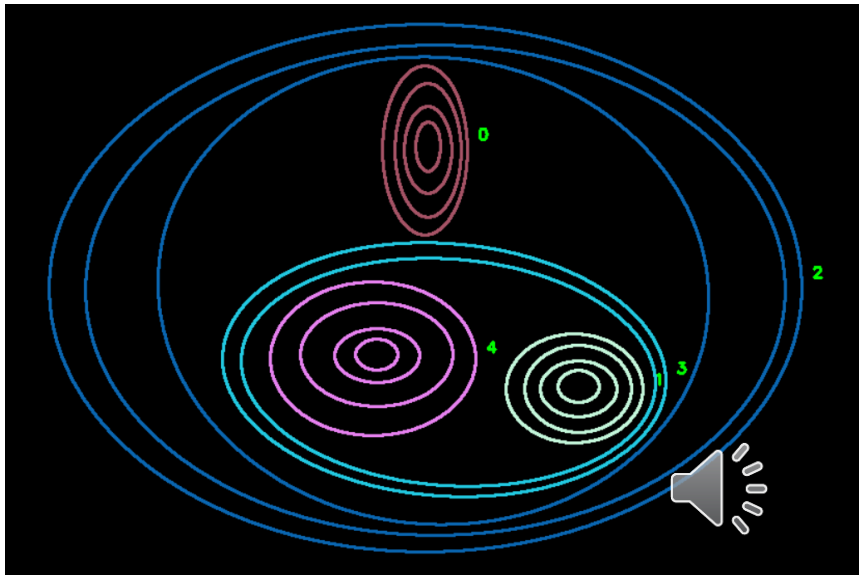




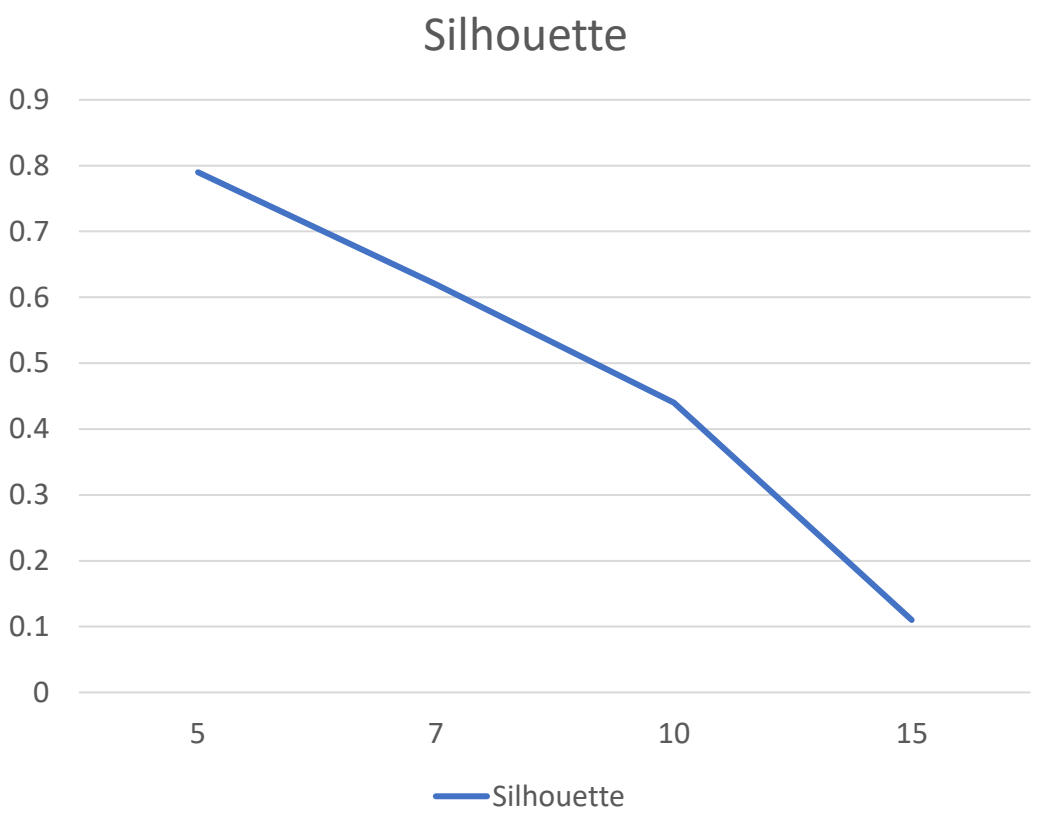
Clustering Results

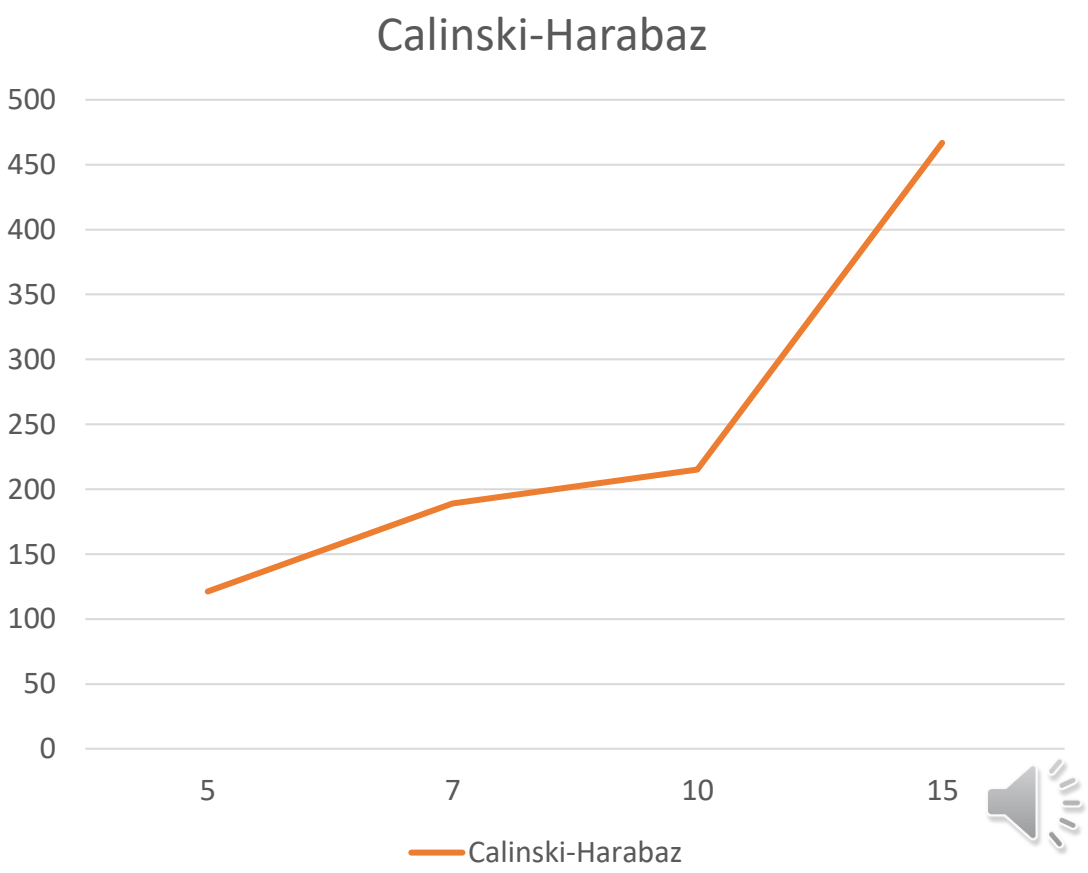
N Clusters	Method	Silhouette	Calinski-Harabasz
4	Ward	0.73	49.12
5	Single	0.79	121.14
3	Complete	0.58	18.54
3	Average	0.58	18.54
5	K-Means (auto)	0.79	121.14
7	K-Means k = 7	0.62	189.08
10	K-Means k = 10	0.44	215.96
15	K-Means k = 15	0.11	466.83



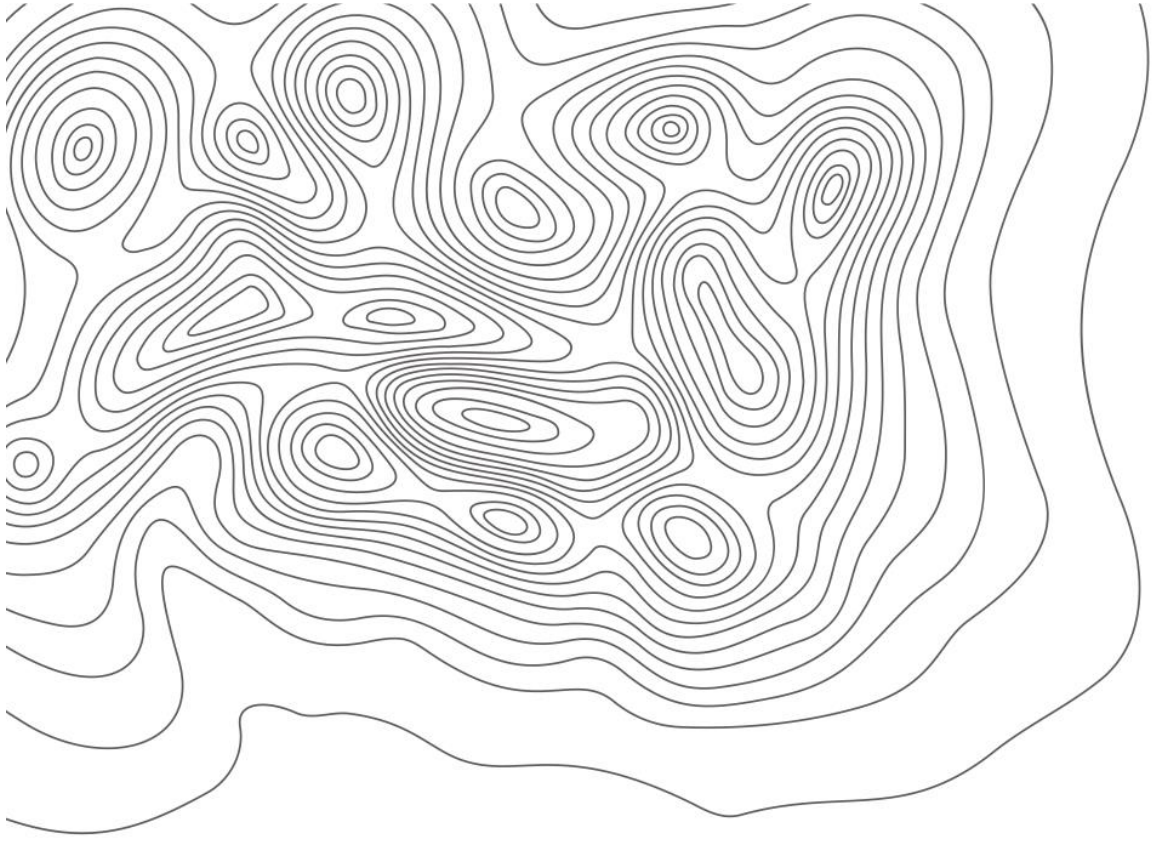


Scores compared

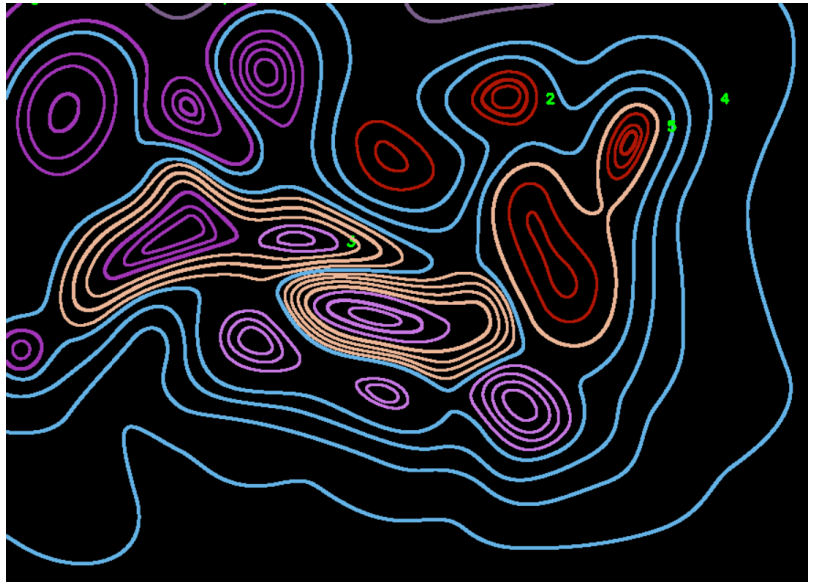




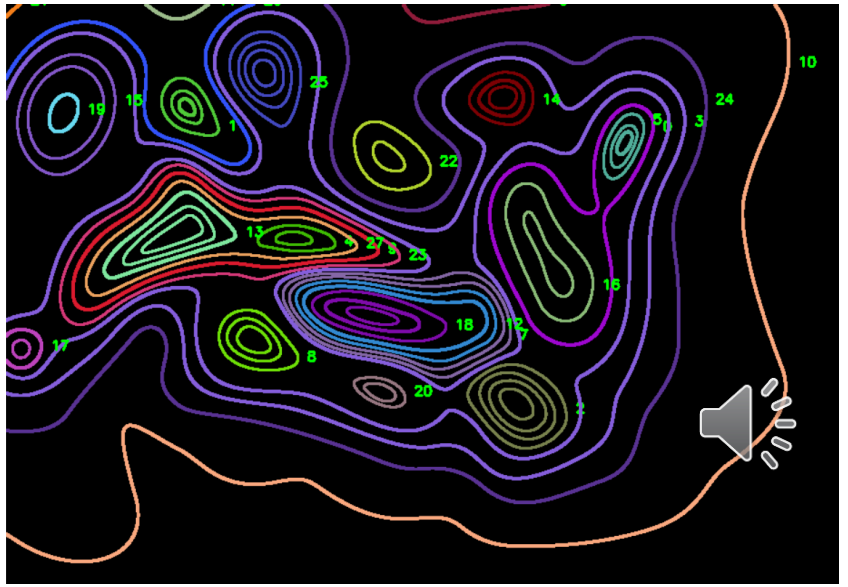
Clustering results



k-=6

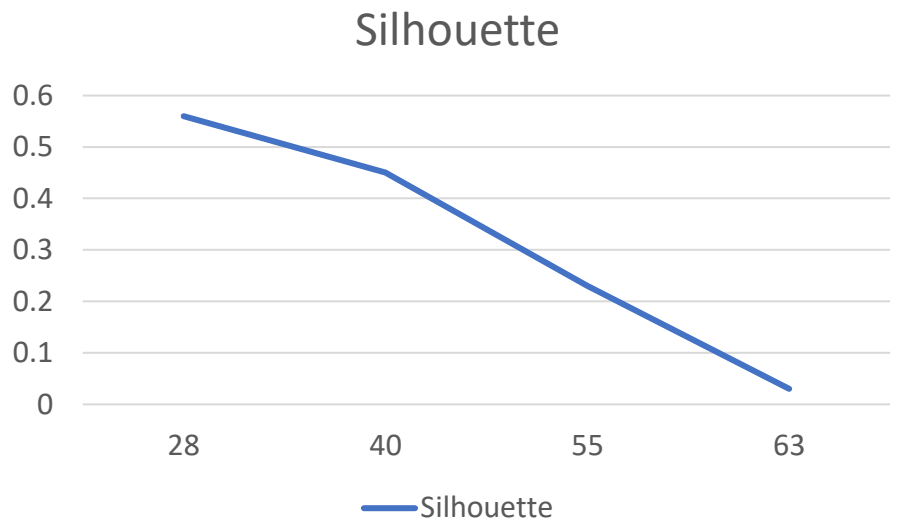


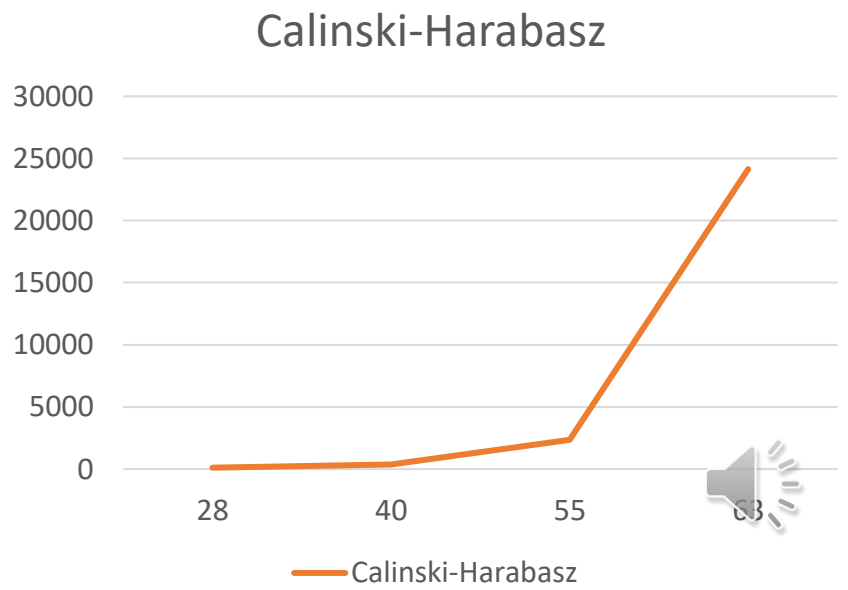
k-=28



Scores compared

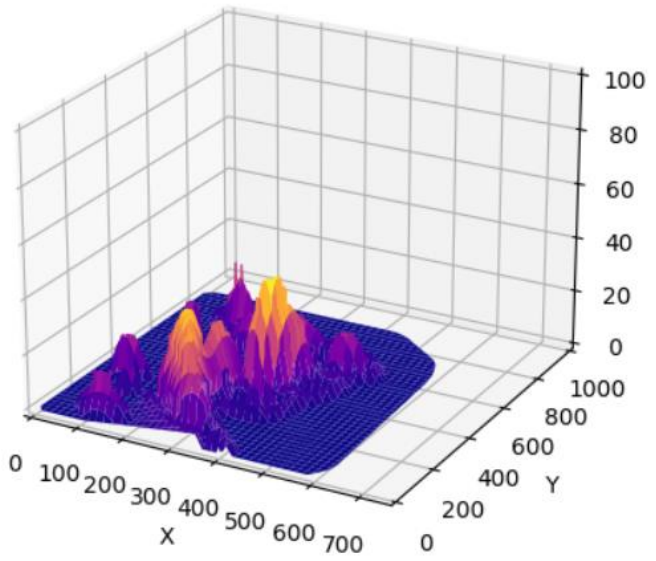
N Clusters	Method	Silhouette	Calinski-Harabasz
6	Ward	0.39	34.26
2	Single	0.555	7.92
2	Complete	0.46	9.01
3	Average	0.45	14.21
28	K-Means (auto)	0.56	127.23
40	K-Means k = 40	0.45	379.54
55	K-Means k = 55	0.23	2362.86
63	K-Means k = 63	0.03	24131.68



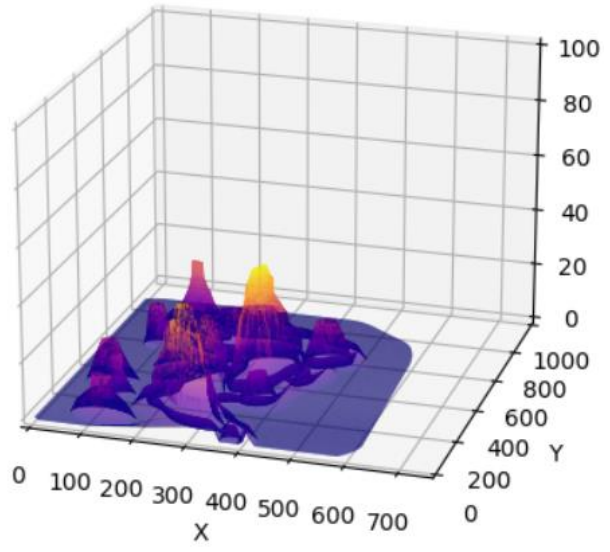


Surface Plot comparison

- The regular grid provides a less detailed plot
- Triangulated plot is more detailed but computationally expensive
  - Runs slowly



Regular Grid Plot



Triangulated Plot

Discussion

- Everything works on simple examples
- Issues start to appear as we have more complex inputs
  - Line breaks
  - Shading disrupting threshold
- Not very automated

Future work

- Better automated threshold
  - Detect the type of image it is and adapt
- Improved contour detection
  - One contour per line
  - Re constructing broken lines
- Different elevation calculation
  - Move from clusters to find an efficient approach
- Alternative plotting options
  - OpenGL
  - Export options

Conclusion

- A program that successfully extracts contour lines from an image
- The elevations are calculated through clustering
- Interpolate the elevations to be plotted onto a surface
- Needs work to be more automatic

Link to code and video

- Project code and presentation