title

October 29, 2024

Abstract

1 Introduction

2 Methods For Segmentation

2.1 Finite Differences

The method of finite differences can be used to find the gradient at each point in the image. This is a simpler calculation than computing the derivative at each point.

The central difference equations for two variables, x and y are used for the calculation of image gradients [2].

$$\frac{\partial f}{\partial x}(x,y) = \frac{f(x_{i+1},y_i) - f(x_{i-1},y_i)}{2h} + O(h^2) \quad (1)$$

$$\frac{\partial f}{\partial y}(x,y) = \frac{f(x_i,y_{i+1}) - f(x_i,y_{i-1})}{2h} + O(h^2) \quad (2)$$

The pseudo-Laplace operator can be given by the discretisation of $|\nabla^2 f| = |\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}|$. This can be approximated using the central difference (2nd derivative), similar to equations (1) and (2).

$$\frac{\partial^2 f}{\partial x^2}(x,y) = \frac{f(x+h,y) - 2f(x,y) + f(x-h,y)}{h^2} + O(h^2) \quad (3)$$

$$\frac{\partial^2 f}{\partial y^2}(x,y) = \frac{f(x,y+k) - 2f(x,y) + f(x,y-k)}{k^2} + O(k^2) \quad (4)$$

Where h and k are the sizes of the steps in between values.

By defining A as the image matrix (entries indicating to each pixel's greyscale value) and using the results from (3) and (4):

$$|\nabla^2 f| = |A_{i+1,j} - A_{i-1,j} - 4A_{i,j} + A_{i,j+1} + A_{i,j-1}| \quad (5)$$

This can be re-written in matrix form to give the Laplace filter:

$$? = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The Laplace filter is an example of a Kernel that can be applied to an image for edge detection.

2.2 Kernels/Convolutions

2.3 The Canny Algorithm

2.3.1 Summary

The Canny Algorithm is an image edge detection algorithm that takes in a greyscale image and returns a new image just defined by its strongest edges.

The steps for the Canny Algorithm are:

- Initially the image is smoothed using a Gaussian convolution filter to reduce unwanted noise.
- The first derivative of each pixel in the image is computed to identify areas with the highest spatial derivatives. Edges create ridges in the gradient magnitude image that is created.
- The algorithm begins tracking along the top of the 'ridges', setting all other pixel values to zero (if they aren't also on a ridge). This creates a very thin, well defined line that runs along each edge. This process is known as non-maximal suppression.
- The tracking process is controlled by two threshold values: T_1 and T_2 (where $T_1 > T_2$). This hysteresis is performed to ensure that noisy edges are not made into separate smaller edge pieces.

One aspect of the Canny Algorithm that could be improved on would be automating the setting of the threshold values (T_l and T_h).

2.3.2 Median-Based threshold assignment

The simplest form of setting the threshold values based on the image can be done by computing the median. The median greyscale value m_{med} is calculated and combined with the standard deviation (σ) to obtain the bounds.

$$T_h = (1+\sigma)m_{med} \quad (6)$$

$$T_l = (1-\sigma)m_{med} \quad (7)$$

These values (T_l and T_h) are substituted into the algorithm to better detect the edges in the selected

image.

2.3.3 'Newtonian' Canny Algorithm

Newton's gravitational laws can also be applied to image segmentation to help automate the selection of threshold values. Every pixel can be represented as a point with mass equal to its greyscale value and then used in Newton's equations [1].

The total gravitational Intensity of a pixel ($E_{i,j}$), based on its neighbours represents the image gradient.

$$E_{i,j} = \sum_{i=1}^{N} \frac{G\ m_i}{||\vec{r_i}||^2} \cdot \frac{\vec{r_i}}{||\vec{r_i}||} \tag{8}$$

Where N is the number of neighbours, G is a constant, m_i and $\vec{r_i}$ are the greyscale value and position vector to the neighbour i from the current pixel.

The gradients at each pixel can be visualised in a 2D vector field:

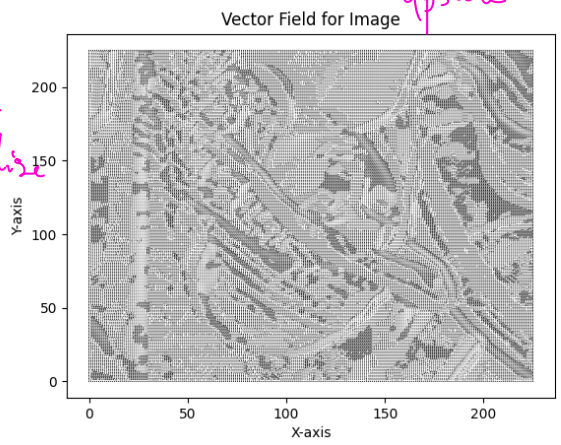


Figure 1: Vector field visualising the magnitude and direction of the image gradient at each point on an image (*Lena* with low resolution 225x225 pixels).

The Threshold values for the Canny Algorithm can be obtained by using relations with the average gradient (E_{avg}), standard deviation (σ) and a constant (k) [1].

$$E_{avg} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |E_{i,j}| \tag{9}$$

$$\sigma = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |E_{i,j} - E_{avg}|^2 \tag{10}$$

$$T_h = E_{avg} + k\sigma \tag{11}$$

$$T_l = \frac{T_h}{2} \tag{12}$$

k is a constant that can be obtained through experiment. The value of k should be inversely related to σ (larger σ means lower k).

2.4 Region based segmentation

Region-based segmentation groups pixels based on shared properties, such as colour composition. This approach becomes especially effective with images that contain a rich variety of properties, making colour images particularly suitable.

In this process:

The image is transformed into an RGB colour space to capture full colour information. A starting pixel is chosen, serving as the initial "seed" for a region. The algorithm examines adjacent pixels, calculating the Euclidean distance between their RGB values and that of the seed pixel. If the distance falls within a defined threshold, the neighbouring pixel is considered similar enough to join the region. This method continues to evaluate and add pixels iteratively until the region is fully defined. The result is a segmented image that groups pixels based on colour similarity, creating cohesive regions that align closely with the natural structure of the image

2.5 K-Means Algorithm

K-Means Clustering (KMC) is an unsupervised learning algorithm that partitions data into K clusters based on feature similarity. In image segmentation, KMC groups pixels by colour, aiding in applications such as image recognition and medical imaging.

The K-Means algorithm consists of three main steps:

1. Initialization: Randomly select K initial centroids from the dataset. These centroids represent the centre of each cluster. 2. Assignment: Each data point x_i is assigned to the nearest centroid c_j based on the Euclidean distance:

$$\text{label}(x_i) = \arg\min_j \|x_i - c_j\|^2$$

This step creates clusters by assigning pixels to the nearest centroid. 3. Update: The centroids

are recalculated as the mean of all points assigned to each cluster:

$$c_j = \frac{1}{N_j} \sum_{x_i \in C_j} x_i$$

where N_j is the number of points in cluster j. The algorithm iterates through the assignment and update steps until the centroids stabilise or convergence is achieved.

To evaluate clustering performance, the Within-Cluster Sum of Squares (WCSS) is calculated:

$$\text{WCSS} = \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - c_j\|^2$$

This metric quantifies the compactness of the clusters, with lower WCSS values indicating more tightly grouped clusters. By minimising WCSS, the algorithm enhances the quality of the clustering.

The Elbow Method is a heuristic for determining the optimal number of clusters K. It involves plotting WCSS against varying K values. As K increases, WCSS generally decreases. The optimal K is found at the "elbow" point, where the rate of decrease sharply changes, indicating that additional clusters yield diminishing returns in WCSS reduction.

An alternative approach for determining the optimal K is the Silhouette Method. This method evaluates how similar an object is to its own cluster compared to other clusters. The silhouette score for a point i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average distance between point i and all other points in its cluster, and $b(i)$ is the average distance from point i to points in the nearest cluster. A higher average silhouette score indicates better-defined clusters.

Optimisation focuses on selecting K using both the Elbow Method and the Silhouette Method to improve segmentation quality and computational efficiency.

2.6 Watershed

3 Noise and Smoothing Methods

3.1 Introduction to Noise

he main challenge with image segmentation is achieving accurate segmentation in the presence of noise. Often, images are not perfectly clear; they may contain various types of noise, be pixelated, or even corrupted, which can significantly impact the effectiveness of segmentation algorithms. Testing how different models perform under noisy conditions is essential to understanding their robustness.

3.2 salt and pepper method

One of the methods that can be used to replicate real world noise, such as that caused by sensor errors or transmission issues, is the salt and pepper noise.

Salt and Pepper noise turns randomly selected individual pixels to either white or black, hence the name. The proportion of pixels selected can be decided.

3.3 SG Filter

3.4 Gaussian Filter

The Gaussian filter is one of the most frequently used tools in image processing for noise smoothing and feature extraction. It is based on the Gaussian function in mathematics, which describes the normal distribution in statistics. The Gaussian function is defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where the variables are defined as follows:

- $G(x, y)$: the value of the Gaussian function at point (x, y)
- σ: the standard deviation, controlling the spread (blurriness) of the filter.
- x and y: the coordinates in the 2D plane

Computing the Gaussian filter involves the method of Fourier transform, where convolution in the spatial domain is equivalent to multiplication in the frequency domain, thus allowing Gaussian filtering to be more computationally efficient, especially for large images. Its computation can be summarised as following:

1) Given an image $I(x, y)$, take its 2D Fourier transform:

$$\hat{I}(u, v) = \mathcal{F}\{I(x, y)\}$$

where $\hat{I}(u, v)$ is the Fourier transform of the image in the frequency domain, and u and v are the frequency variables.

In the frequency domain, the Fourier transform of a Gaussian function is also a Gaussian function:

$$\hat{G}(u,v) = e^{-\frac{2\pi^2\sigma^2(u^2+v^2)}{2}}$$

Now, multiply the Fourier-transformed image by the Fourier-transformed Gaussian filter:

$$\hat{I}_{\text{filtered}}(u,v) = \hat{I}(u,v) \cdot \hat{G}(u,v)$$

This step replaces the convolution in the spatial domain with multiplication in the frequency domain, which is more computationally efficient for large images.

Finally, apply the inverse Fourier transform to bring the filtered image back into the spatial domain:

$$I_{\text{filtered}}(x,y) = \mathcal{F}^{-1}\{\hat{I}_{\text{filtered}}(u,v)\}$$

This gives us the Gaussian-filtered image in the spatial domain.

3.4.1 Laplacian of Gaussian

The Laplacian of Gaussian (LoG) is a technique used in image processing for edge detection. It is a combination of two operations:

1. Gaussian Smoothing: This step smooths the image, reducing noise and fine details, by applying a Gaussian filter.
2. Laplacian Operation: The Laplacian operator highlights areas of rapid intensity change, effectively identifying edges. The Laplacian of an image $I(x,y)$ is given by:

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

The Laplacian of Gaussian combines these two operations by first applying the Gaussian filter to smooth the image and then applying the Laplacian to detect edges. However, to simplify computation, the Laplacian of Gaussian filter is often defined as a single operation, which can be directly convolved with the image. In the spatial domain, the LoG filter can be derived by taking the Laplacian of the Gaussian function.

Similarly to the Gaussian, the Laplacian can also be calculated using Fourier Transform:

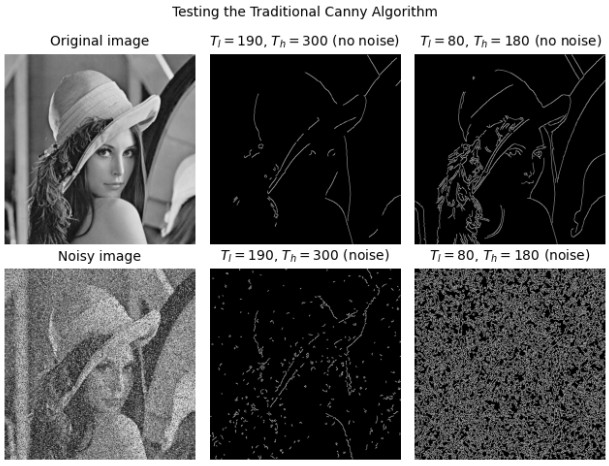


Figure 2: Figure displays the result for the traditional Canny algorithm with two different threshold values. The second row highlights how the algorithm reacts to salt and pepper noise.

3.5 Fourier Transforms

4 Results

4.1 Canny

4.1.1 Traditional Canny Algorithm

4.1.2 Canny with Median-Based Threshold Assignment

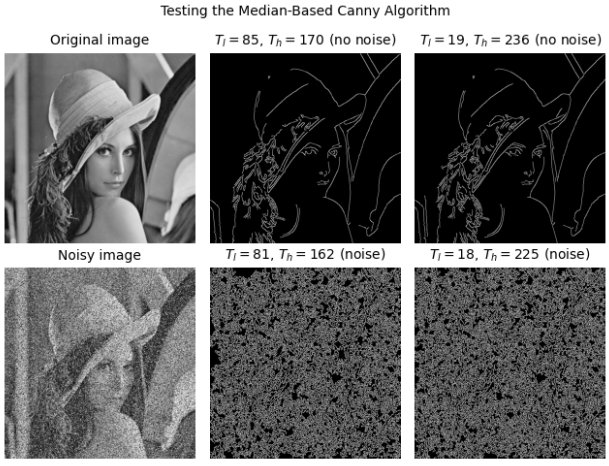


Figure 3: Figure showing how the Canny algorithm with threshold values set relating to the median, reacts to images with noise.Parameter values used: $\sigma = 0.33$ (middle column), $\sigma = 0.85$ (right column)

//

As seen in Figure 3, this variant of the Canny algorithm is very poor at dealing with noisy images. The original image is unrecognisable once the algorithm has been applied as most of the noise is picked up as edges. The traditional algorithm performed much better with thresholds selected by the user (shown in Figure 2) This means the median-based method used to select the threshold values for Canny struggled to find the best thresholds. This could be because the median of the image would remain similar when salt and pepper noise is added. The greyscale values of the salt and pepper are 0 and 255 meaning they pad the ends of the image greyscale distribution. This hardly changes the middle value (median).

4.1.3 'Newtonian' Canny Algorithm

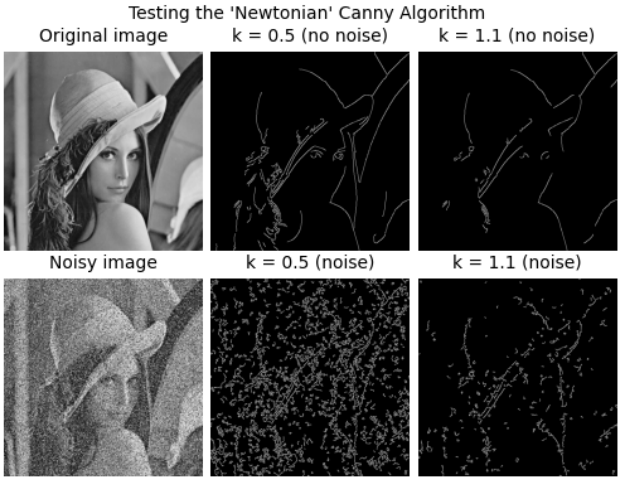


Figure 4: Figure displaying the differences between the Newtonian Canny algorithms when noise is applied to *Lena* image

By observing Figure 4, the Improved Canny algorithm identifies fewer edges than the traditional Canny algorithm but this means that it includes fewer unwanted details.

4.2 K-means

K-means clustering was applied to segment images, categorising pixels into distinct clusters based on colour similarity. The primary objective was to simplify image structure and highlight key regions. K-means is advantageous for image segmentation due to its balance of simplicity and computational efficiency, essential when handling high-resolution images. By clustering each pixel, the algorithm generates a segmented image that retains primary colour features and textures from the original.

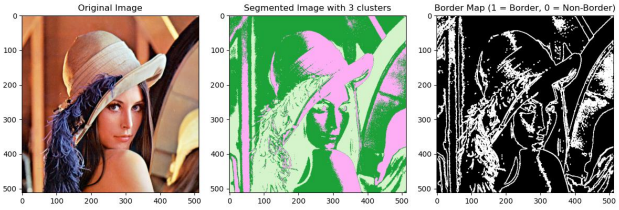


Figure 5: Figure displaying the K means segmentation of Lenna for $K = 3$, also displaying the borders as a binary value, which will be combined with the Canny Method.

Advantages of K-Means Clustering

- **Computational Efficiency:** K-means is computationally efficient, making it suitable for processing large, high-dimensional data such as images.
- **Simplicity of Implementation:** The algorithm is relatively straightforward, which allows for adaptable, quick segmentation results suitable for a variety of images and clustering tasks.
- **Control Over Detail Level:** K-means allows the user to define the number of clusters K, which was useful in allowing the segmentation of various images with different colours and complexities.

Limitations

- **Choice of K:** Selecting an optimal number of clusters is a challenge in K-means. Few clusters may result in oversimplification, while many may lead to over-segmentation, affecting image interpretability.
- **Sensitivity to Initialisation:** The algorithm is sensitive to the initial placement of centroids, which may lead to inconsistent results across different runs. Improved stability may require multiple initialisations.
- **Assumption of Cluster Shape:** K-means assumes clusters are spherical in the feature space, which is often unrealistic for image data, where clusters can have irregular boundaries.

- **Border Map Complexity:** Although the creation of a border map effectively identifies edges between clusters, it introduces additional computational overhead. Iterating through each pixel's neighbours can significantly slow down the algorithm, particularly for high-resolution images.

To determine the optimal number of clusters,we implemented both the elbow method and silhouette method, to achieve an optimum K value for each image. These approaches provide quantitative insights into cluster compactness and separation, enhancing segmentation quality. The figure below shows the plot of the elbow method with 10 iterations, with the aim of identifying the elbow point, or as our code identified, the point where the maximal change in curvature is observed.

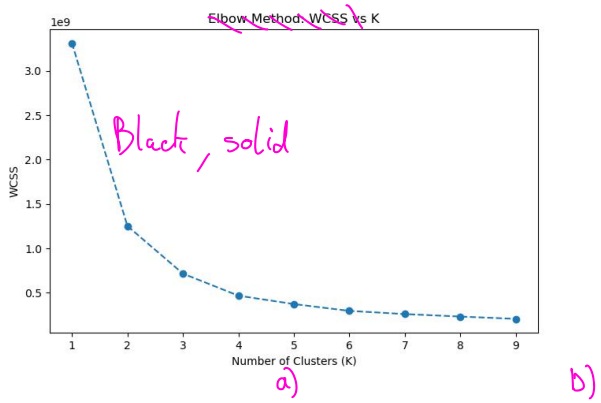


Figure 6: Figure displaying the plot of WCSS vs K values over 10 iterations

Elbow Method The Elbow method was employed to identify an optimal K based on the within-cluster sum of squares (WCSS). By plotting WCSS against K, the 'elbow' point signifies a diminishing rate of decrease in WCSS, indicating a reasonable K where further cluster addition yields minimal improvement in intra-cluster similarity. This method helps balance segmentation quality with computational efficiency, ensuring that the chosen K provides sufficient detail without over-segmentation.

Silhouette Method (Not Implemented) The Silhouette method evaluates pixel similarity within a cluster relative to other clusters. Calculating the average silhouette score allows for insight into the separation and cohesiveness of clusters. Higher scores indicate better-defined clusters with minimal overlap. Implementing silhouette analysis could improve segmentation by providing a secondary check on the optimal K, enhancing robustness in segmenting complex images.

reduce

4.3 Combining Methods

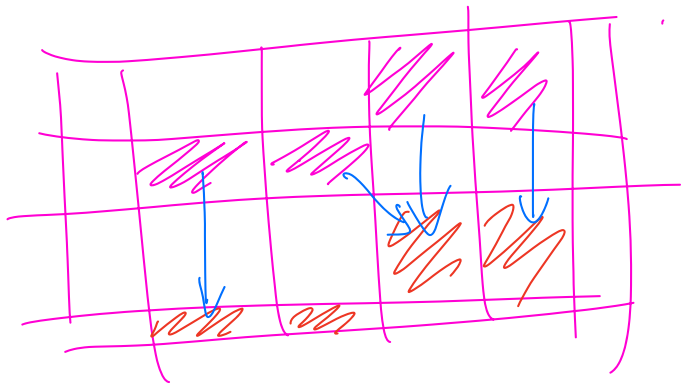
5 Discussion

6 Conclusion

References

[1] Weibin Rong et al. "An improved Canny edge detection algorithm". In: *2014 IEEE International Conference on Mechatronics and Automation*. 2014, pp. 577–582. DOI: 10.1109/ICMA.2014.6885761.

[2] Juan Triana and Luis Ferro. *Finite difference methods in image processing*. URL: https://revistas.unitru.edu.pe/index.php/SSMM/article/view/3903.


Method 1
Method 2