1 Question 1 - Image Segmentation

This question details the segmentation algorithms used to segment the required sections of the lung ct image, the noisy flowers and the coin images.

1.1 Part A: Lung CT Image Segmentation

This section details the segmentation algorithm of the Lung CT image shown in Figure 1. The region to be segmented is the lung area, including all the nodules and tissue within. The image is taken from the LIDC-IDRI dataset [**?**].

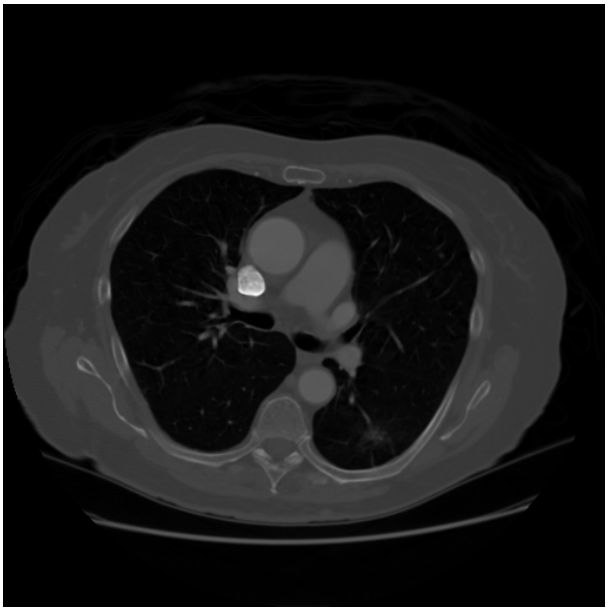


Figure 1: Lung CT image from the LIDC-IDRI dataset.

1.1.1 Image Characteristics

The image shows a high contrast between the lungs and surrounding non lung tissue. This observation motivates a threshold based approach to segment the lung area. The nodules and tissue in the intra lung regions are of a much higher intensity than the lung tissue, as such it is appropriate supplement the thresholding step with additional morphological operations.

1.1.2 Assumptions

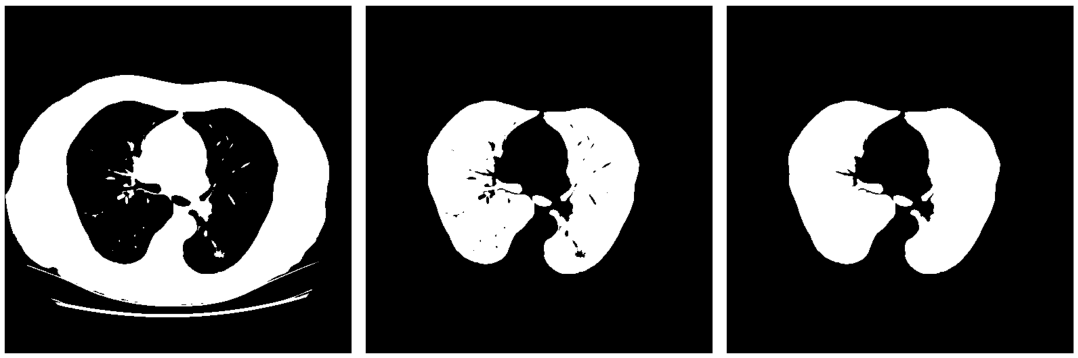
The algorithm assumes that after binarising the image via thresholding, the background makes up the largest connected component, with the left and right lungs being the next two largest. The algorithm also assumes that the nodules and tissue within the lung regions are fully enclosed and do not touch the border of the lung regions.

1.1.3 Segmentation Algorithm

- **Binarise Image via Otsu Thresholding:** The image is thresholded using Otsu's threshold. This is a global thresholding method that maximises the inter-class variance of foreground and background.
- **Connected Component Analysis:** The binary image is inverted to make the lung and background regions the foreground. The connected components of the inverted image are labelled using 8-connectivity. The second and third largest connected components are

selected to recover the left and right lung regions. The connected components are selected based on their area.

- **Binary Hole Filling** To recover the nodules and tissue within the lung regions, binary hole filling is performed. The algorithm fills holes in a binary image by inverting the image, labelling connected regions, and converting regions not connected to the border from background to foreground.



(a) (b) (c)

Figure 2: Key steps in the segmentation of the lung CT image: (a) Binarisation using Otsu's threshold, (b) Mask inversion and selection of lung regions via connected component analysis, (c) Recovery of internal structures through binary hole filling.

Note, this segmentation algorithm was also implemented from scratch. Relevant code can be found in `src/q1a_from_scratch.py` and `src/from_scratch_seg_funcs.py`.

1.1.4 Results

The final segmentation of the lung CT image is shown in Figure 1 The segmentation clearly captures the relevant lung regions and all the nodules and tissues within.

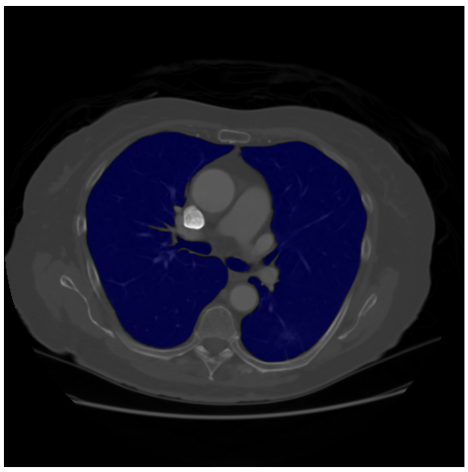
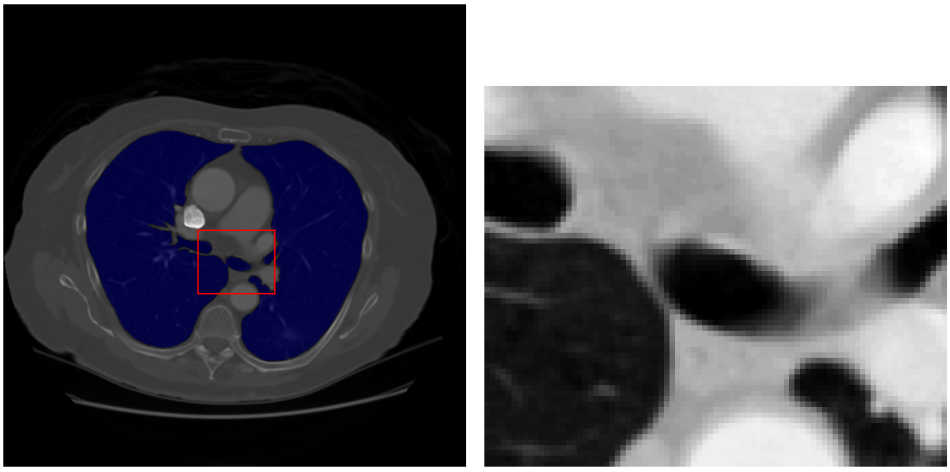


Figure 3: Final segmentation of the lung CT image.

1.1.5 Discussion

Figure 4 shows a potential ambiguous region in the segmentation. This region was segmented due to having a similar intensity as the lung tissue and being connected to lung tissue post Otsu

thresholding. However, as this region could not be clearly identified as non lung tissue without additional context, it was kept in the final segmentation. This also highlights the limitations of thresholding based segmentation methods in identifying regions that are not clearly defined by intensity.



(a) (b)

Figure 4: Ambiguous region in the segmentation: (a) Segmentation with bounding box around the ambiguous region, (b) Zoomed in view of the ambiguous region.

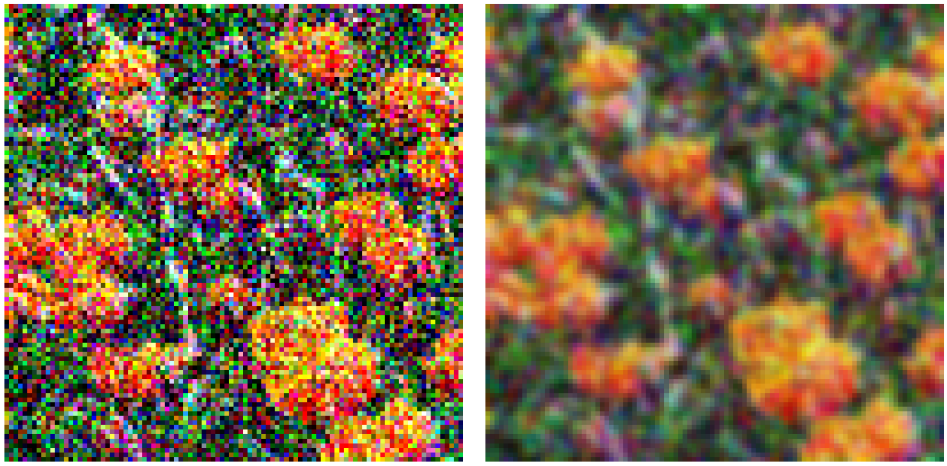
1.2 Part B: Noisy Flowers Image Segmentation

This section details the segmentation algorithm of the noisy flowers image shown in Figure 5. The region to be segmented are all the purple flower heads in the image.

Figure 5: Noisy Flowers image.

1.2.1 Image Characteristics

The image features a type of noise that is typical to gaussian degradations. This was confirmed by applying a gaussian filter to each of the colour channels seperately in the image and observing that nearby pixel colours were more. This effect is shown in Figure 6.



(a) (b)

Figure 6: Effect of Gaussian filtering on the colour channels of the noisy flowers image: (a) Cropped region of the noisy flowers image, (b) Cropped region after applying a Gaussian filter with standard deviation 1.

The purple flowers in 5, except for the ones clearly in the foreground, are all roughly of the same shade of purple. There are two batches of purple flowers in the image, one large group down the middle and another smaller group along the upper left corner of the image. The purple flowers in the foreground have shadows on them making the bottom half of the flower darker than the top half. Due to the general uniformity in colours of the flower after denoising, a clustering type algorithm is appropriate for this image.

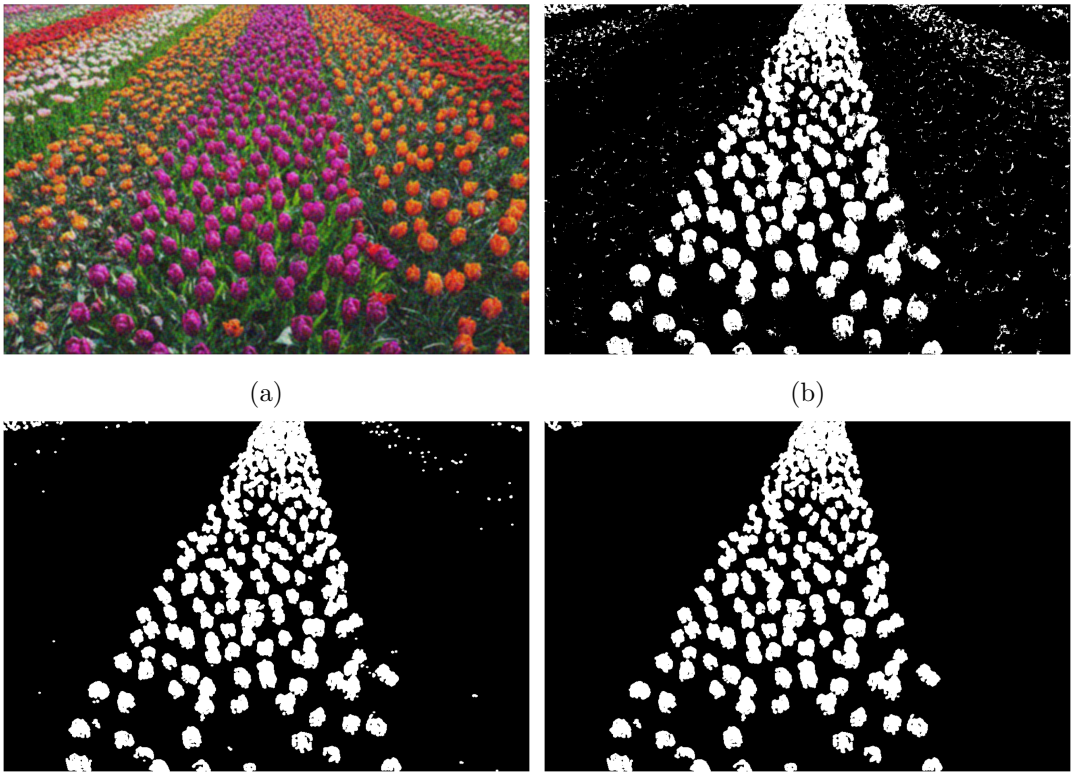
1.2.2 Assumptions

The main assumption made by this algorithm is in choosing the number of clusters. The number of clusters is chosen to be 5, which is chosen based on the number of distinct colours in the image. The green grass and stems, the orange flower heads, the purple flower heads, the white flower heads and the red flower heads.

1.2.3 Segmentation Algorithm

- **Bilateral Gaussian Filtering:** First the image was denoised using a bilateral filter [2]. This filter was chosen for its ability to denoise while considering both spatial proximity and colour similarity, ideal for preserving flower head edges. A spatial standard deviation of 15 was chosen to allow for smoothing across the entire flower head regions, while a colour standard deviation of 1 ensures edge sharpness by only blending similarly coloured pixels. This approach effectively reduces noise without blurring distinct colour boundaries.
- **Conversion from RGB to LAB colour Space:** The image is converted from RGB to LAB colour space, which better approximates human vision by separating colour and luminance. This space enhances clustering accuracy because Euclidean distances more closely reflect perceptual differences, making it ideal for grouping similar colours like purple flower heads.

- **K-Means Clustering:** K-means clustering is applied to the LAB image. The number of clusters is chosen to be 5, based on the number of distinct colours in the image. The image is then segmented by assigning each pixel to the cluster with the closest centroid.
- **Select Purple Pixels** The cluster centre closest to the the LAB value of purple is chosen to be the desired cluster to represent the foreground. The image is binarised by setting all pixels not in this cluster to 0 and all pixels in this cluster to 1.
- **Denoise Segmentation via Morphological Opening** To remove small-scale noise from the mask while retaining the shape and size of the main flower heads, a morphological opening operation is applied.
- **Remove Small Connected Components:** Finally to remove any left over noise, all connected components with an area less than 100 pixels are removed. The previous morphological opening step ensures the required thresholding does not remove any of the main flower heads.



(a) (b)

(c) (d)

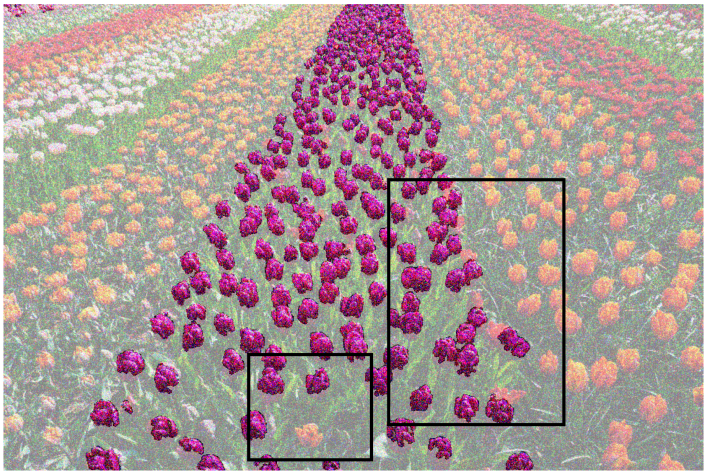
Figure 7: Segmentation process for the noisy flowers image: (a) Image after denoising with a bilateral filtering, showing reduced colour noise while preserving edge details. (b) Result of applying K-means clustering to segment purple pixels. (c) Segmentation mask post morphological opening, illustrating reduced noise in the segmentation. (d) Final segmentation after removing small connected components, highlighting the refined segmentation of flower regions.

1.2.4 Results

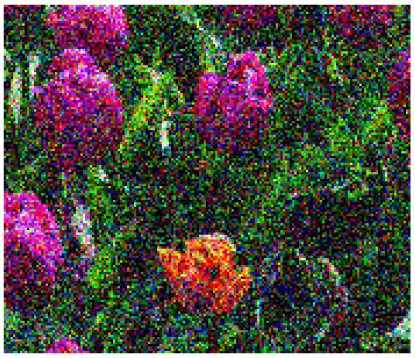
The final segmentation of the noisy flowers image is shown in Figure 8. The segmentation clearly captures most of the purple flower heads in the main central bunch. It also captures some in the smaller bunch in the upper left corner. It is able to distinguish between the colours

effectively, with some of the orange and red flowers in the purple group being excluded from the segmentation quite effectively as seen in figure 9.

Figure 8: Final segmentation of the noisy flowers image.



(a)



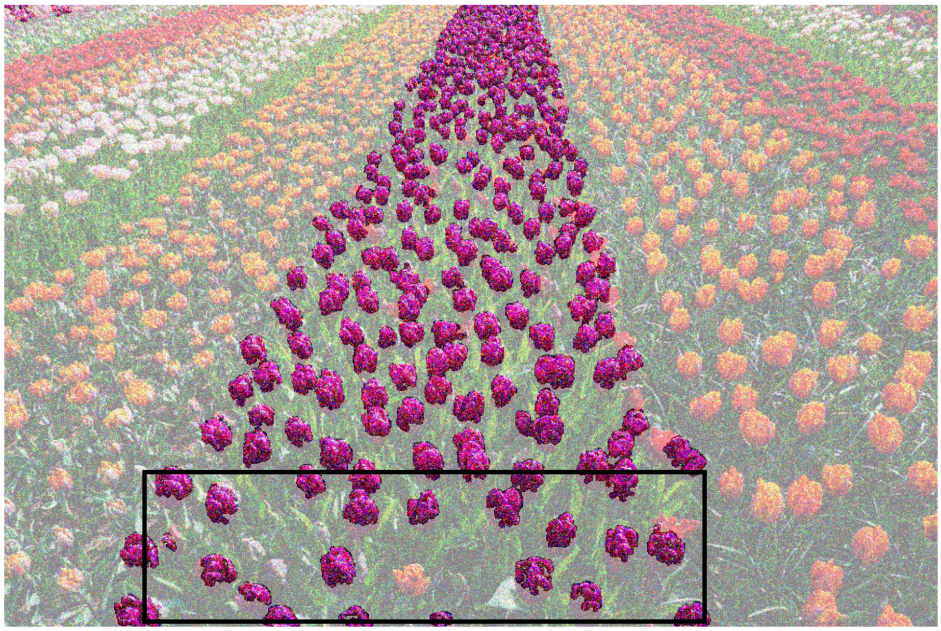
(b)

(c)

Figure 9: Zoomed in view of the segmentation: (a) Segmentation with bounding boxs around regions of interest where the segmentation has performed well, (b) Zoomed in view of the segmentation showing the exclusion of orange flower from the purple group, (c) Zoomed in view of the segmentation showing the exclusion of orange and red flowers that are occluded by flowers from the purple group.

1.2.5 Discussion

The main challenge in this segmentation is the segmentation of the purple flower heads that are in the foreground (closest to the camera). These flower heads have shadows on them, making the bottom half of the flower darker than the top half, making them difficult to fully segment with the current approach. An example such case is shown in figure 10. This highlights the limitations of a segmentation algorithm which segments purely on colour. A potential improvement to this algorithm would be to consider spatial shape based information to better segment these flower heads or make use of more sophisticated pre-processing to remove the shadows.



(a)

(b)

Figure 10: Limitations in segmentation of noisy flowers image: (a) Full view of the image with poorly segmented areas highlighted in a bounding box, demonstrating where the algorithm fails to accurately segment due to shadows. (b) Close-up view of a poorly segmented area, showing flower heads which feature strong shadows, impacting segmentation accuracy.

1.3 Part C: Coin Image Segmentation

This section details the segmentation algorithm of the coin image shown in Figure 11. The region to be segmented are the first coin in the first row, the second coin in the second row, the third coin in the third row and the fourth coin in the fourth row.

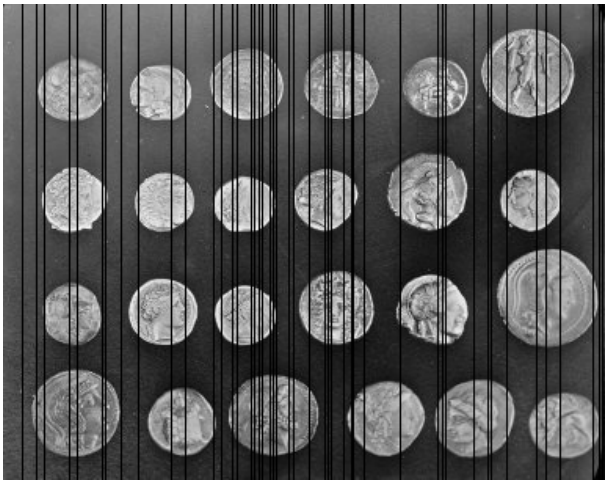


Figure 11: Coin image.

1.3.1 Image Characteristics

The image has been degraded with vertical line type artefacts. These artefacts have likely been synthetically generated and can be corrected by filtering. There is generally good contrast between the coins and the background, however fig 12, applying an otsu threshold reveals that the coins in the first row are not clearly separated from the background. Edge detection to segment the coins is likely to perform better in this case.

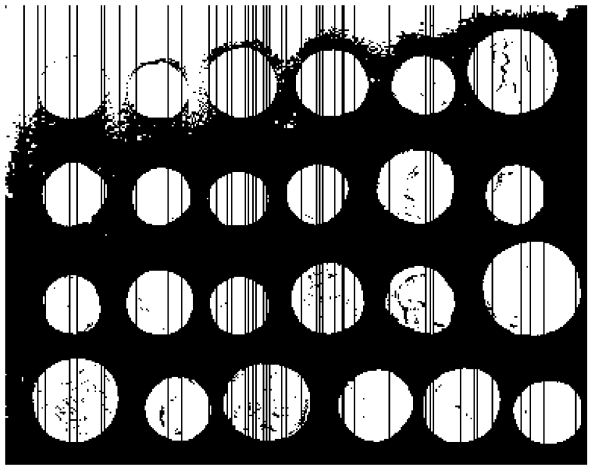


Figure 12: Effect of Otsu thresholding on the coin image.

1.3.2 Assumptions

To select the required coins for the task, the algorithm assumes that the coins are arranged in 4 rows with 6 columns, without making prior assumptions on what coin belongs to which position in the grid. The algorithm also assumes that the coins are not touching each other. Lastly, it assumes that the background is the largest connected component in the image.

1.3.3 Segmentation Algorithm

- **Row-wise Median Filtering:** Initially, a row-wise median filter is applied to the image to remove vertical line artefacts. This step is crucial to recover high quality edges for subsequent edge detection.

- **Edge Detection with Canny:** Canny edge detection is a multi-stage algorithm that uses Gaussian smoothing to reduce noise, then detects edges by identifying areas with high gradient magnitudes. It employs a double threshold to classify edges into strong, weak, and non-edges: strong edges exceed the high threshold and are confirmed as true edges; weak edges between the two thresholds are only confirmed if connected to strong edges; and areas below the low threshold are dismissed as non-edges. This method ensures robust edge continuity and noise suppression.

- **Dilation of Edges:** After detecting edges, a dilation operation is performed using an elliptical structuring element. This step helps to close small gaps in the detected edges, which aids in forming continuous outlines for the coins.

- **Removal of the Largest Component:** The resulting image is then inverted and connected components are labelled. The largest connected component is assumed to be the background and is removed.

- **Integration of Edges and Mask:** Edges detected from the Canny step are added to the mask of coins to enhance the definitions of the coins' borders and 'close' the coins so that the mask is more faithful to the actual coin shapes.

- **Binary Hole Filling:** A binary hole filling operation is then performed to fill any unsegmented areas within each coin.

- **Median Filtering on Mask:** Finally, to remove spurious regions of the mask which arose from edges that did not belong to any coins, both row-wise and column-wise median filtering is applied to the mask. This step removes regions that are not part of the coins. This step ensures that the next step of removing small components need not require too large a threshold, risking the removal of actual coins.

- **Removal of Small Connected Components:** Finally, any small connected components with an area less than 100 pixels are removed from the mask. This step ensures that only the coins are retained in the final mask.

- **Sorting and Assigning Grid Positions:** The centroids of the coin masks are then sorted based on their vertical and horizontal coordinates to assign each coin a grid position.

- **Selecting Specific Coins:** The coins whose grid position matches the required positions are selected.

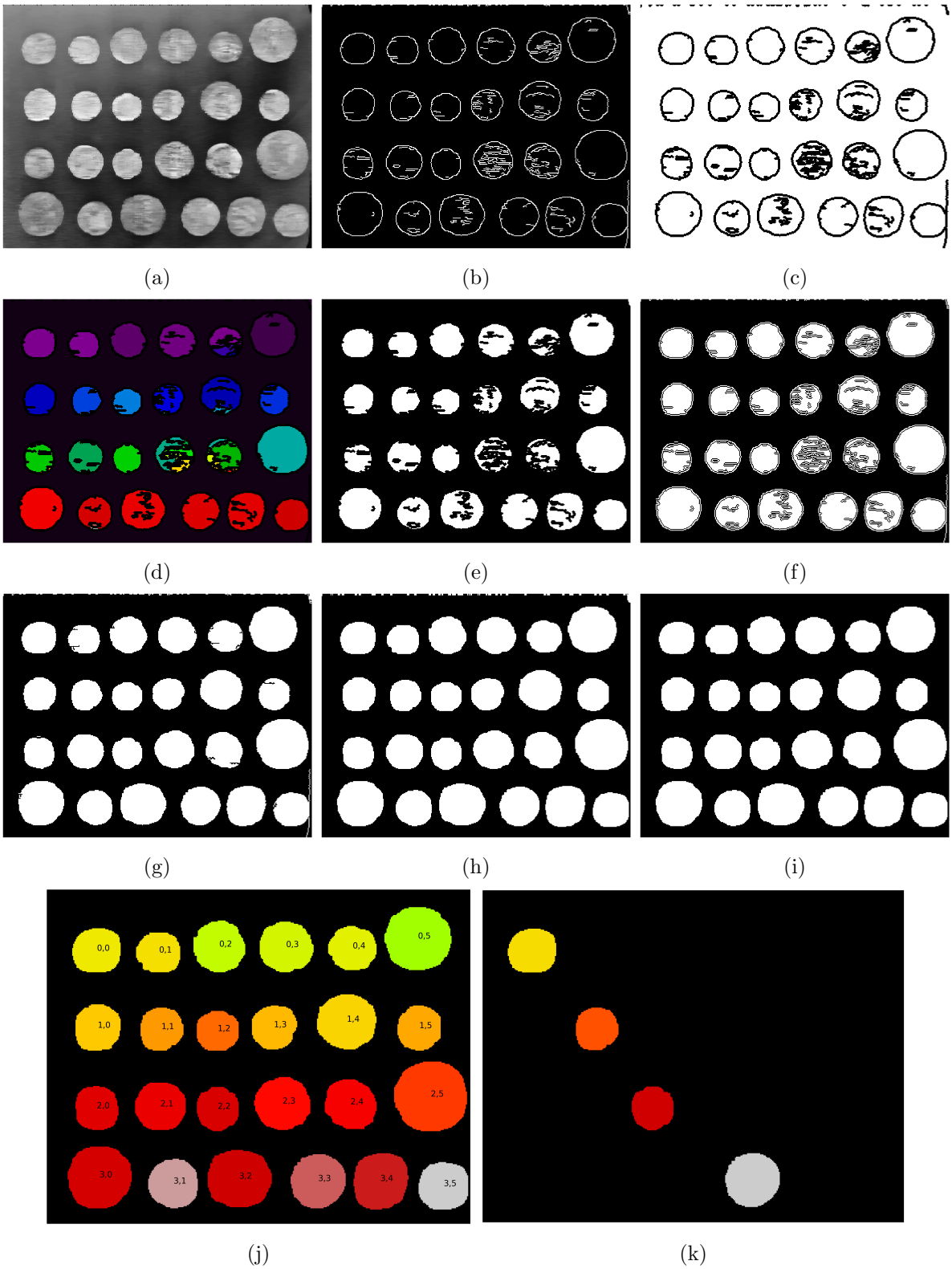


Figure 13: Detailed steps of the coin segmentation process including: (a) Initial median filtering to remove line artefacts, (b) Canny edge detection to outline the coins, (c) Dilated edges to enhance edge continuity, (d) Identification of connected components to identify background, (e) Mask after background removal, (f) Mask integrated with edges to refine the coin outlines, (g) Binary hole filling to complete the interior of the coins, (h) Median filtered mask to remove spurious edges, (i) Final segmentation after eliminating small connected components to isolate individual coins, (j) Identified coins with their grid positions, (k) Selected coins based on specified positions.

1.3.4 Results

The final segmentation of the coin image is shown in Figure 14. The segmentation clearly captures the required coins in the specified grid positions.

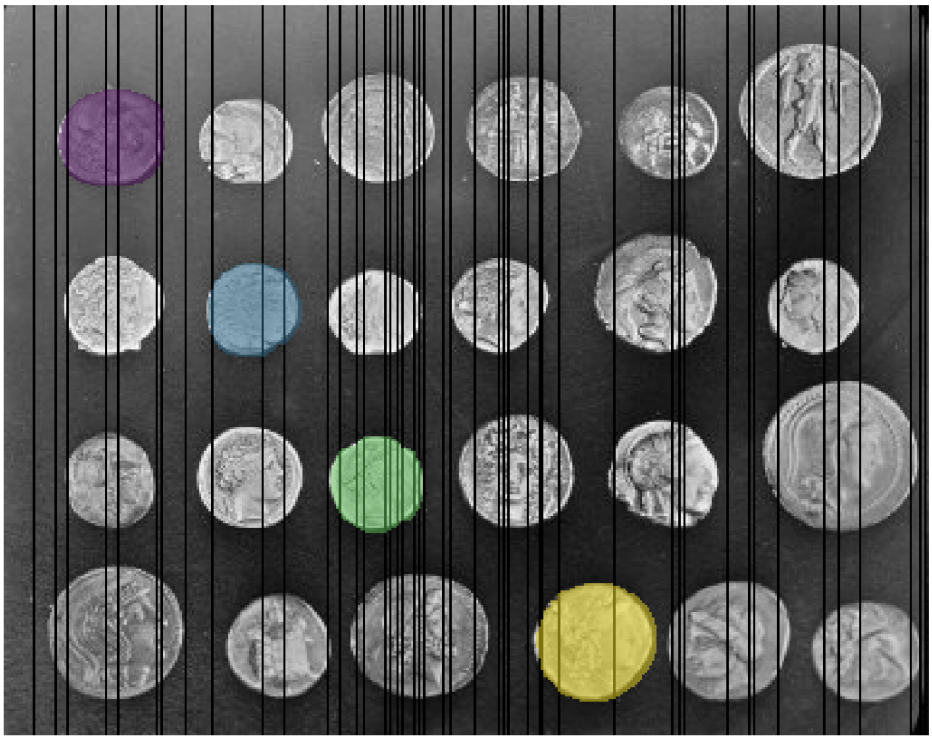


Figure 14: Final segmentation of the coin image.

1.3.5 Discussion

The chosen method works almost perfectly, the only issue is that the median filtering and edge dilation steps can result in the final mask being slightly unfaithful to the actual coin shapes around the edge. However, with more experimentation the current approach could be improved to better capture the coin shapes.

2 Question 2 - Inverse Problems and Multi-resolution analysis

2.1 Part A - Inverse Problems

2.1.1 Motivation

In many scientific and engineering contexts, problems can be modelled as a linear system of equations, expressed as

$$y = Ax + b,$$

where:

- y is the vector of measurements,
- x represents the unknown vector of the underlying signal or model parameters.
- A is the system matrix that defines how measurements y relate to the signal / parameter vector x.

- b is the (typically unknown) vector representing noise or measurement errors.

The challenge lies in recovering the unknown vector x from the measurements y. However, such systems are rarely well-posed; it is not possible to invert the matrix A and obtain a unique solution. It is more typically for these systems to fall into the categories of being either over-determined, with more measurements than unknowns, or under-determined, with more unknowns than measurements.

In such cases, finding a solution involves minimising an objective function that quantifies the error between the observed measurements and those predicted by the model. This process is complemented by incorporating regularisation terms, which are required to find a unique solution in the under determined regime.

2.1.2 Problem Set Up

In this scenario, the goal is to fit a line described by the equation $y = ax + c$ to a dataset consisting of noisy measurements. This is a typical example of a linear inverse problem where we aim to determine the parameters of a line, which in matrix terms can be framed as:

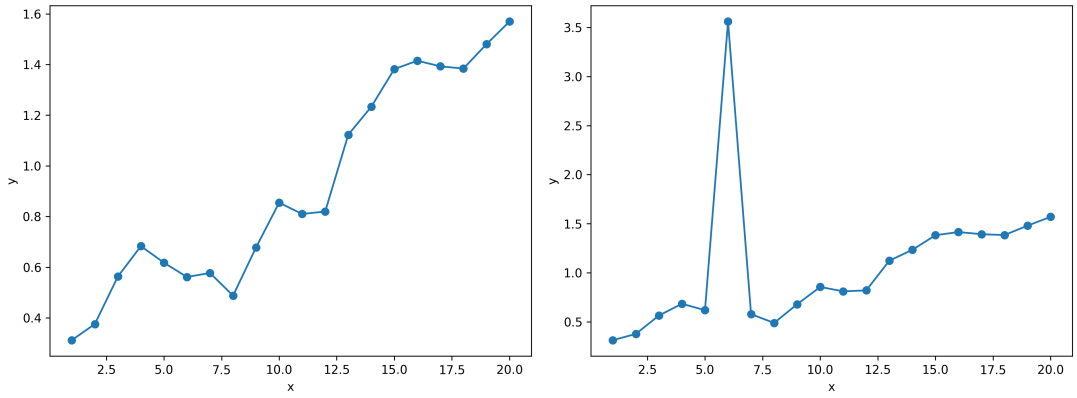
$$y = Ax + b.$$

Here:

- y is the vector of observed y-coordinates from the dataset, representing the outputs or dependent variables.
- A is the system matrix that incorporates the independent variable data. For each measurement pair (x_i, y_i), the matrix A contains a row $[x_i, 1]$ to account for the slope and the intercept of the line.
- x is the vector of parameters to be estimated, specifically $[a, c]$, where a is the slope and c is the intercept of the line.
- b represents the vector of measurement errors.

This linear regression problem is tackled for two datasets describing noisy line measurements. Both datasets are depicted in Figure 15, illustrating the points used for line fitting. It is worth noting that the second dataset is identical to the first for all points except for one which is modified to describe an outlier point.

This system is over-determined for the given datasets because there are more measurements (20 pairs of x, y values) than there are parameters to estimate (just two, a and c). Although over-determined systems may be well posed and have a unique solution and when all measurements are consistent with the true signal. However, in the presence of noise, the system will not have any exact solutions, where all data points lie perfectly on the same line. Instead, the objective is to find the best estimates for the parameters $x = [a, c]$ that minimise the difference between the predicted line Ax and the observed data y, via the objective function. In this problem, the L_2 and L_1 norms of the error vector, $\epsilon = y - Ax$, are explored as the objective functions.

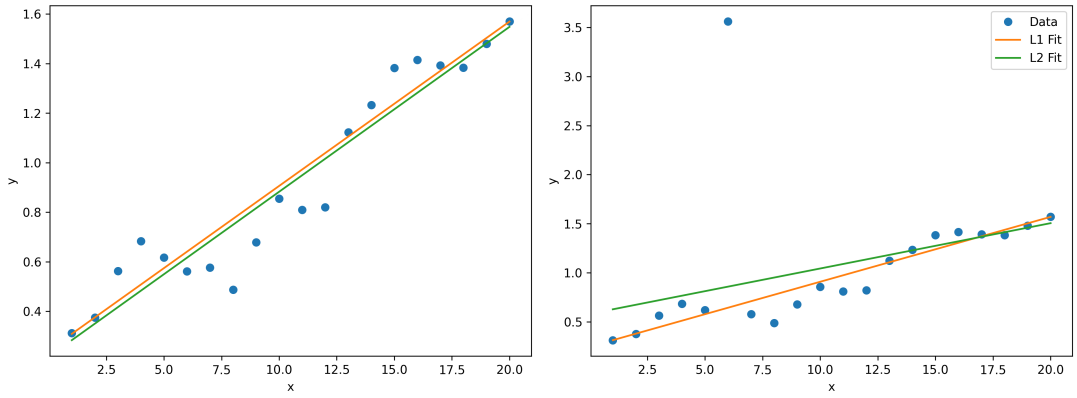


(a) (b)

Figure 15: Two datasets describing noisy measurements of points along a line: (a) Dataset 1 - line data with noisy observations (b) Dataset 2 - line data with noisy observations and a single outlier.

2.1.3 Results

To minimise the objective function for the given datasets, the `minimize` function from the `scipy.optimize` library is employed. This function uses the SLSQP optimisation algorithm to find the minimum of the objective function [1]. The results of the line fitting for the two datasets are shown in Figure 16 and the estimated line parameters are summarised in Table 1.



(a) (b)

Figure 16: Results of line fitting using the SLSQP optimization algorithm to minimise the L_2 and L_1 error norms for the two datasets, which are shown in green and orange respectively: (a) Dataset 1, (b) Dataset 2 with the outlier.

Dataset	**Objective Function**	**Slope (a)**	**Intercept (c)**
1	L_2 Norm	0.0665	0.2172
	L_1 Norm	0.0663	0.2429
2	L_2 Norm	0.0462	0.5804
	L_1 Norm	0.0662	0.2458

Table 1: Summary of the estimated line parameters for the two datasets using the L_2 and L_1 error norms.

2.1.4 Discussion

In the absence of outliers, both L_2 and L_1 norms provide comparable results for line fitting, indicating a robust estimation of parameters under normal noise conditions. However, when outliers are present, the L_1 norm offers a more accurate estimation because it minimises the absolute errors, which reduces the impact of extreme values. Conversely, the L_2 norm, which minimises the sum of squared errors, gives disproportionately high weight to outliers. This sensitivity causes the L_2 norm to be more influenced by the outlier, skewing the line fit significantly away from the majority of the data points, as observed in the results for the second dataset.

2.2 Part B - Compressed Sensing Reconstruction

2.3 Part C - Multi-resolution Analysis

References

[1] D. Kraft. A software package for sequential quadratic programming. Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center – Institute for Flight Mechanics, Koln, Germany, 1988.

[2] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998.