Image Analysis Coursework

Vishal Jain

June 19, 2024

Contents

1 Question 1 - Image Segmentation **2**
1.1 Part A: Lung CT Image Segmentation . . . 2
1.1.1 Segmentation Algorithm: . . . 2
1.1.2 Results . . . 3
1.1.3 Discussion . . . 3
1.2 Part B: Noisy Flowers Image Segmentation . . . 3
1.2.1 Segmentation Algorithm: . . . 4
1.2.2 Results . . . 5
1.2.3 Discussion . . . 7
1.3 Part C: Coin Image Segmentation . . . 8
1.3.1 Segmentation Algorithm . . . 9
1.3.2 Results . . . 11
1.3.3 Discussion . . . 11

2 Question 2 - Inverse Problems and Multi-resolution analysis **11**
2.1 Part A - Solving Over Determined Inverse Problems: Line Fitting . . . 11
2.1.1 Results . . . 12
2.1.2 Discussion . . . 13
2.2 Part B - Compressed Sensing Reconstruction and Underdetermined Inverse Problems . . . 13
2.2.1 Problem Setup . . . 14
2.2.2 Results . . . 15
2.3 Part C - Compressed Reconstruction and Multi-resolution Analysis . . . 17
2.3.1 Discrete Wavelet Transform of the Image . . . 17
2.3.2 Image Reconstruction with Selective Coefficient Retention . . . 19

3 Question 3 - Solving inverse problems **20**
3.1 Part A - Convergence Rate of Gradient Descent . . . 20
3.1.1 Proving f is Convex . . . 21
3.1.2 Identifying the Lipschitz Constant of the Gradient of f . . . 21
3.1.3 Finding the Optimal Solution . . . 21
3.1.4 Theoretical Convergence Rate . . . 22
3.1.5 Practical Convergence Rate . . . 22
3.1.6 Strong Convexity of the Function . . . 22
3.2 Part B - Data Driven and Model Driven Reconstruction . . . 23
3.2.1 Model-Driven Approach . . . 23
3.2.2 Data-Driven Approach . . . 23
3.2.3 Model Architecture . . . 24
3.3 Training . . . 24
3.4 Results . . . 24

1 Question 1 - Image Segmentation

1.1 Part A: Lung CT Image Segmentation

For the segmentation of lung CT image, shown in Figure 1, the primary objective is to accurately delineate lung tissues, including nodules, from surrounding anatomical structures. The inherent high contrast between the lung fields and adjacent tissues suggests the utility of a threshold-based segmentation method.

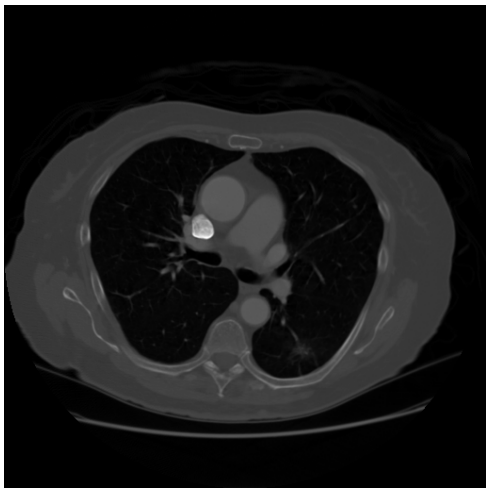
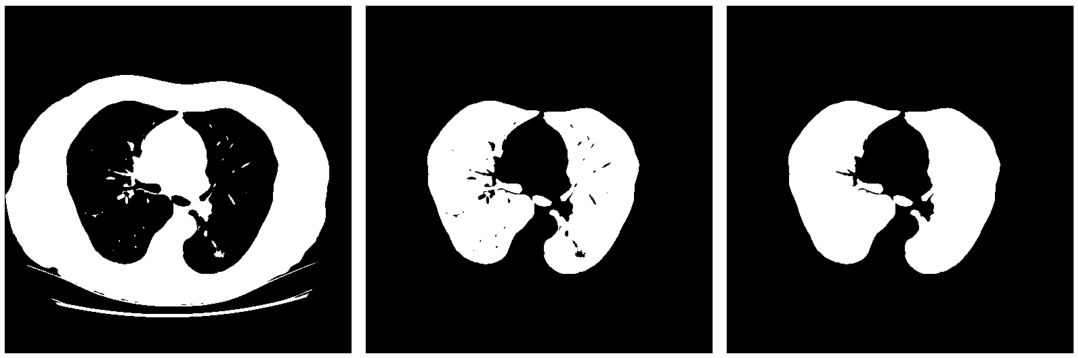


Figure 1: Lung CT image.

1.1.1 Segmentation Algorithm:

- **Otsu Thresholding:** The image is binarised via Otsu's threshold, which finds the threshold that maximises the inter-class variance.
- **Connected Component Analysis:** Connected component analysis with 8-connectivity is used to select the left and right lungs.
- **Binary Hole Filling:** To recover the nodules and tissue within the lung regions, binary hole filling is performed.



(a) (b) (c)

Figure 2: Segmentation Algorithm: (a) Binarisation using Otsu's threshold, (b) Connected component analysis, (c) 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`.

1.1.2 Results

The final segmentation of the lung CT image is shown in Figure 3 The segmentation clearly captures the relevant lung regions and all the nodules and tissues within.

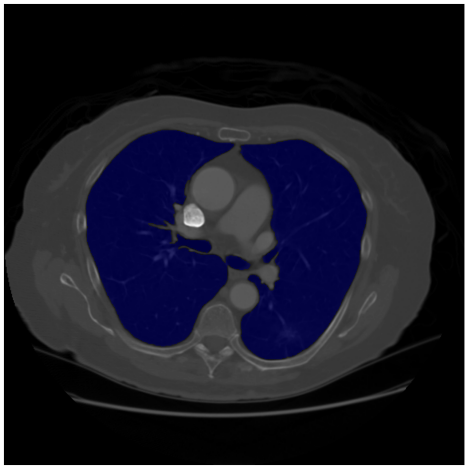
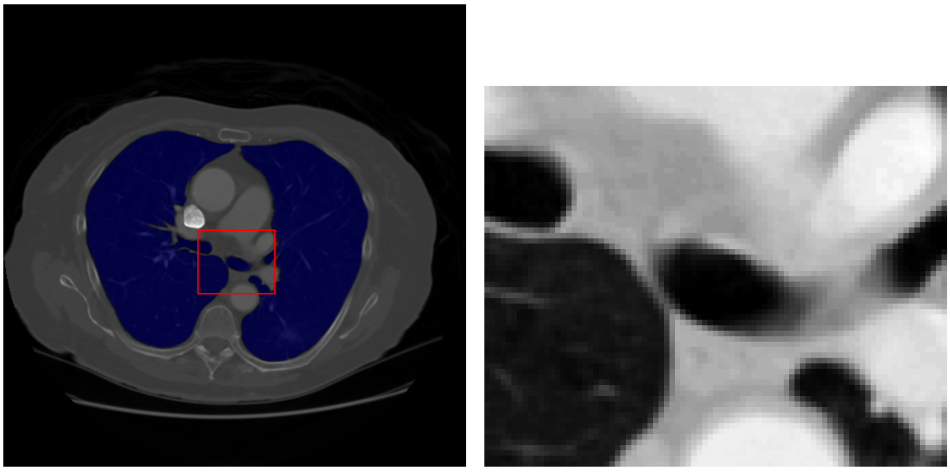


Figure 3: Final segmentation of the lung CT image.

1.1.3 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.



(a) (b)

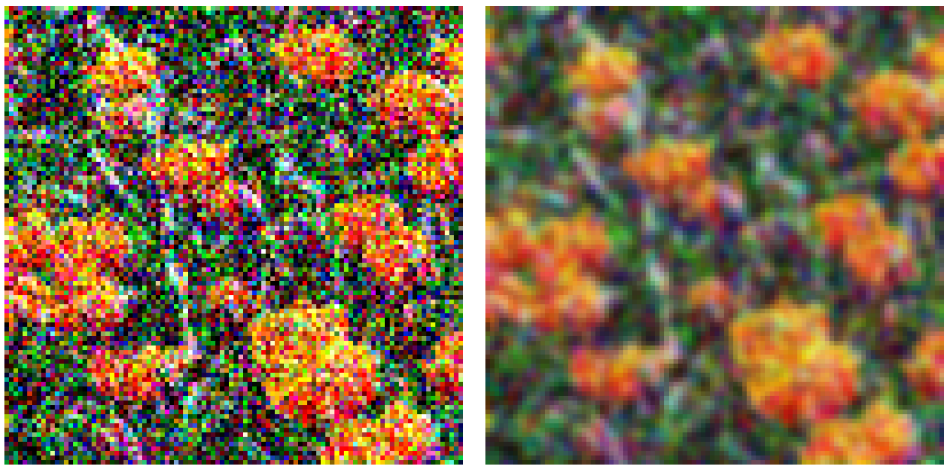
Figure 4: Ambiguous region in the segmentation: (a) Segmentation with bounding box around the ambiguous region, (b) Zoomed in view of the ambiguous region.

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.

The noise in the image was classified to be gaussian noise due to the success of applying gaussian filters seperately on each colour channel, 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.

1.2.1 Segmentation Algorithm:

- **Bilateral Gaussian Filtering:** Initial denoising is achieved using a bilateral filter, chosen for its ability to reduce noise while preserving edges by considering both color similarity and spatial proximity. This step maintains the integrity of flower boundaries against the noise.
- **Conversion to LAB Color Space:** The image is converted from RGB to LAB color space. Euclidean distances in LAB more closely resemble perceptual differences and thus

makes the clustering more effective.

- **K-Means Clustering:** K means is applied with five clusters to match the number of distinct color groups observed.
- **Post-Clustering Processing:** Additional steps include selecting the cluster closest to purple's LAB value and refining the segmentation mask through morphological operations and removing small connected components.

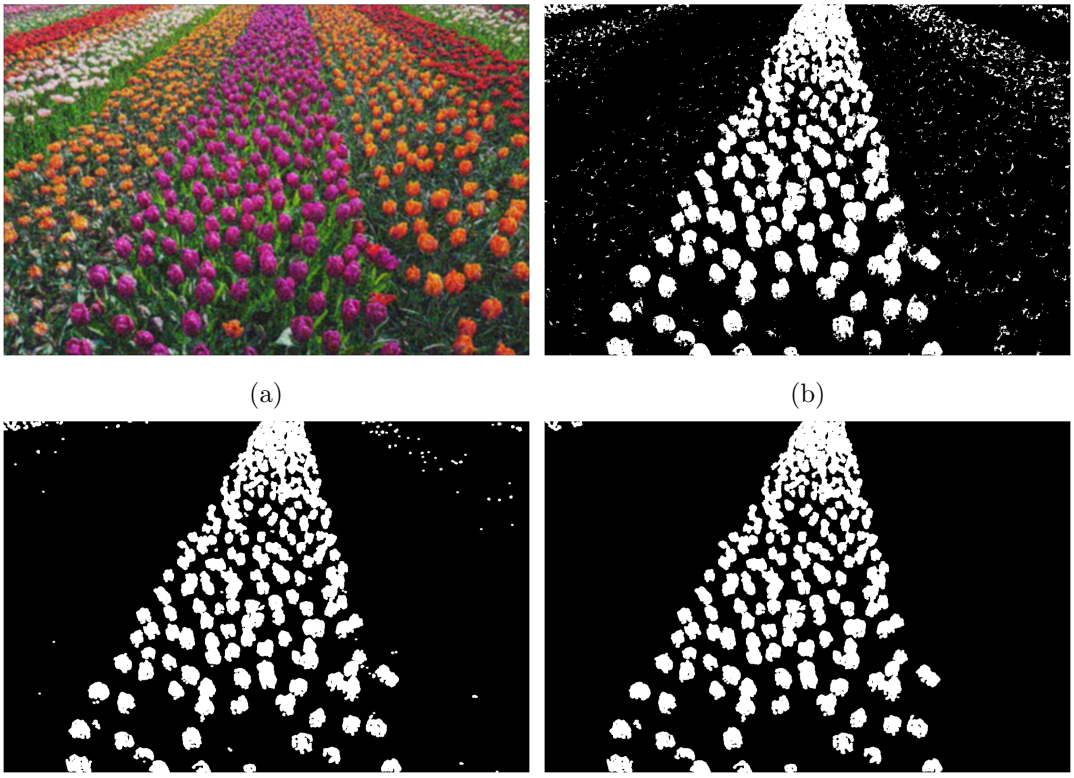
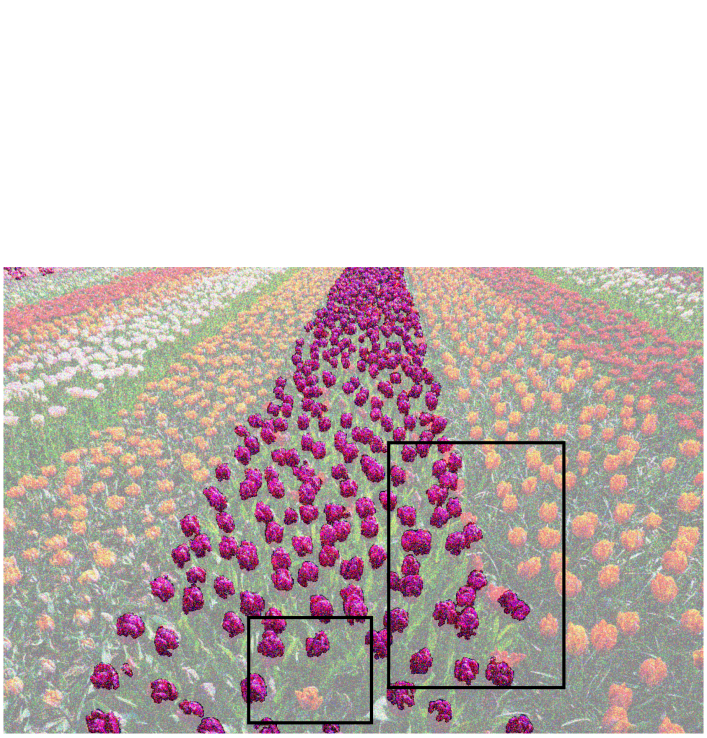


Figure 7: Segmentation algorithm: (a) Bilateral filtering (b) K-means clustering to segment purple pixels. (c) Segmentation mask post morphological opening (d) Final segmentation after removing small connected components

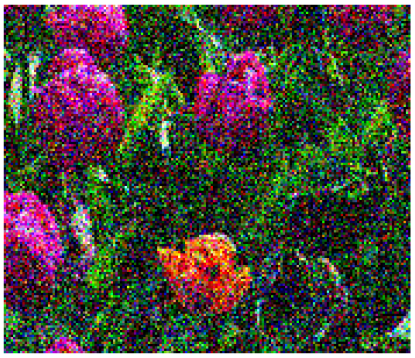
1.2.2 Results

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)



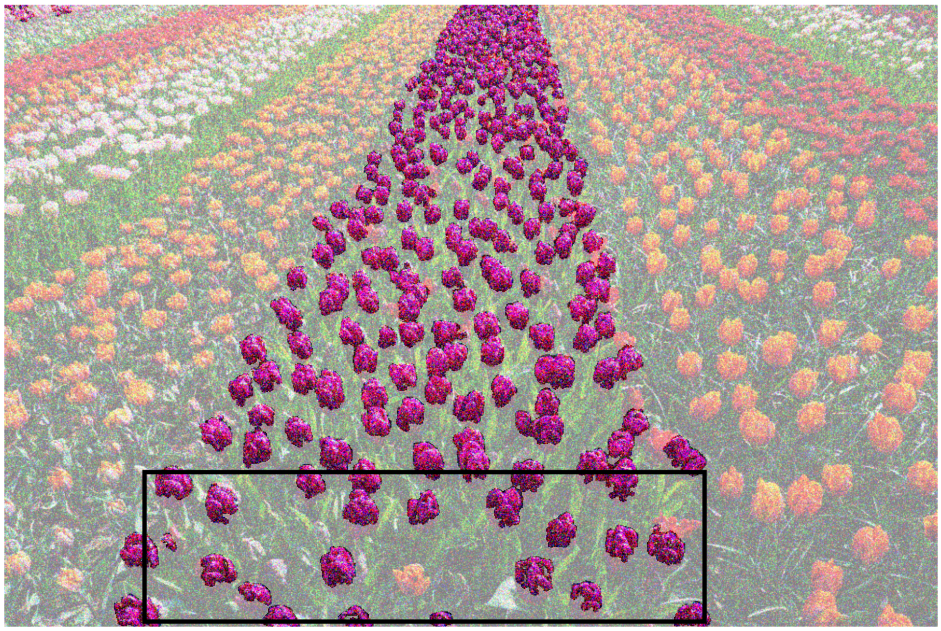
(b)

(c)

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

1.2.3 Discussion

The primary challenge in segmenting the foreground purple flower heads arises from shadows that darken the lower half, complicating color-based segmentation. This issue is illustrated in Figure 10.



(a)

(b)

Figure 10: Limitations in segmentation: (a) Full view of the image with poorly segmented areas highlighted in a bounding box (b) Flowers with strong shadows

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. The image has been degraded with vertical line type artefacts, however can be corrected by filtering. There is generally good contrast between the coins and the background suggesting that edge detection can be used to segment the coins.

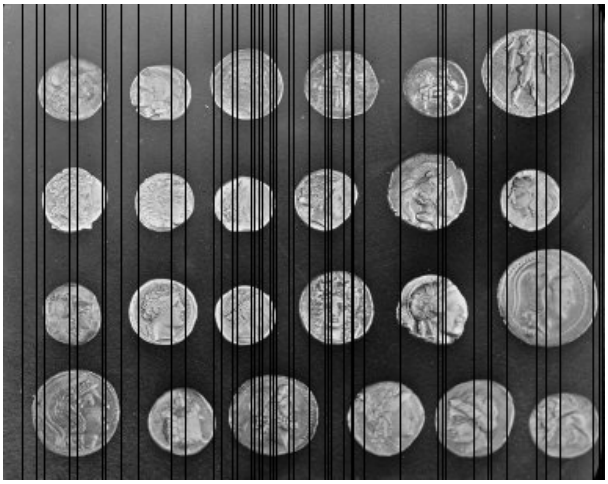


Figure 11: Coin image.

1.3.1 Segmentation Algorithm

- **Row-wise Median Filtering:** A row-wise median filter is applied to the image to remove vertical line artefacts.
- **Edge Detection with Canny:** Canny edge detection is used to get a binary edge mask. 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.
- **Dilation of Edges:** Next, a dilation operation is performed to close small gaps in the edge outlines of the coins.
- **Removal of the Largest Component:** The resulting image is then labelled using 4-connectivity. 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 back to enhance the definitions of the coins' borders.
- **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 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.
- **Sorting and Assigning Grid Positions:** The centroids of the coin masks are then sorted and used to assign each coin a grid position.
- **Selecting Specific Coins:** The coins whose grid position matches the required positions are selected.

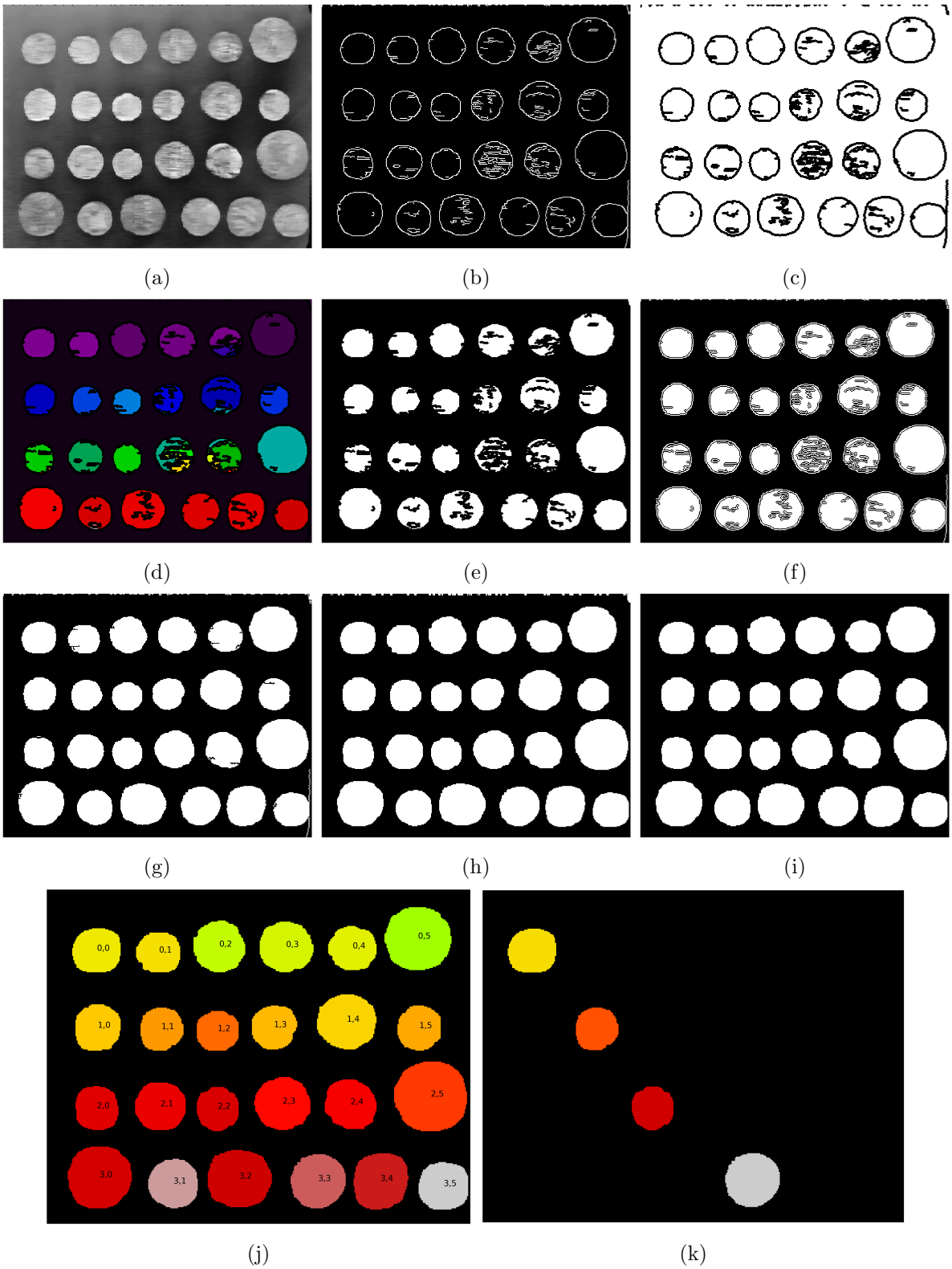


Figure 12: Segmentation Algorithm: (a) Initial row wise median filtering (b) Canny edge detection (c) Dilated edges (d) Identification of connected components (e) Binary Mask after background removal (f) Mask integrated with edges, (g) Binary hole filling, (h) Post row-wise and column-wise median filtered mask (i) Final segmentation after eliminating small connected components, (j) Identified coins with their grid positions, (k) Final Selected coins.

1.3.2 Results

The final segmentation of the coin image is shown in Figure 13. The segmentation clearly captures the required coins in the specified grid positions.

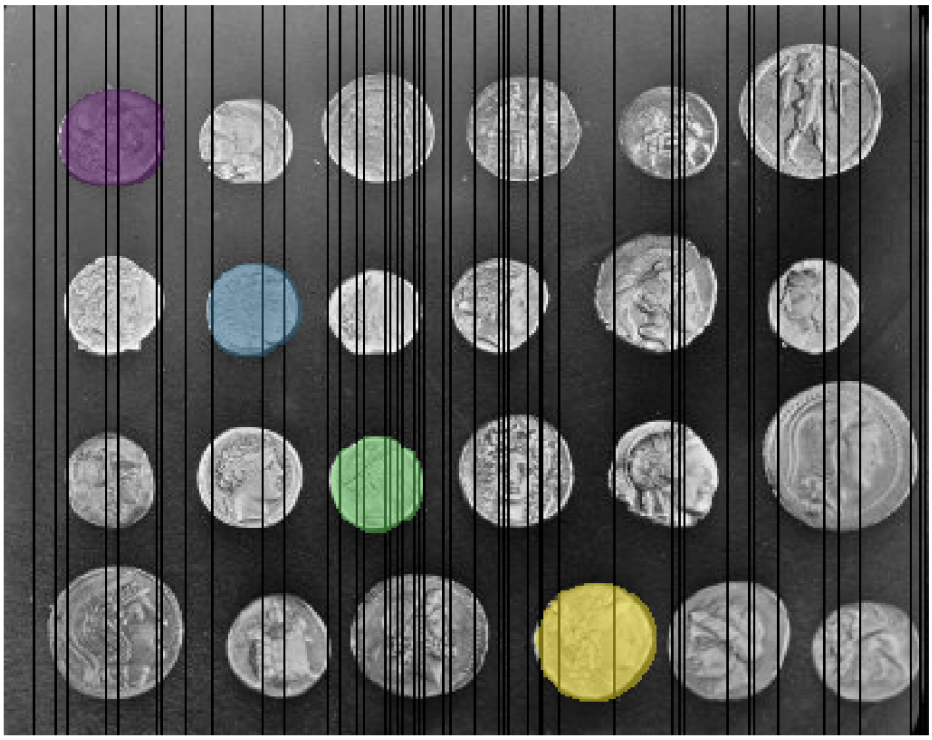


Figure 13: Final segmentation of the coin image.

1.3.3 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.

2 Question 2 - Inverse Problems and Multi-resolution analysis

2.1 Part A - Solving Over Determined Inverse Problems: Line Fitting

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 mathematical terms can be framed as:

$$y = Ax + b.$$

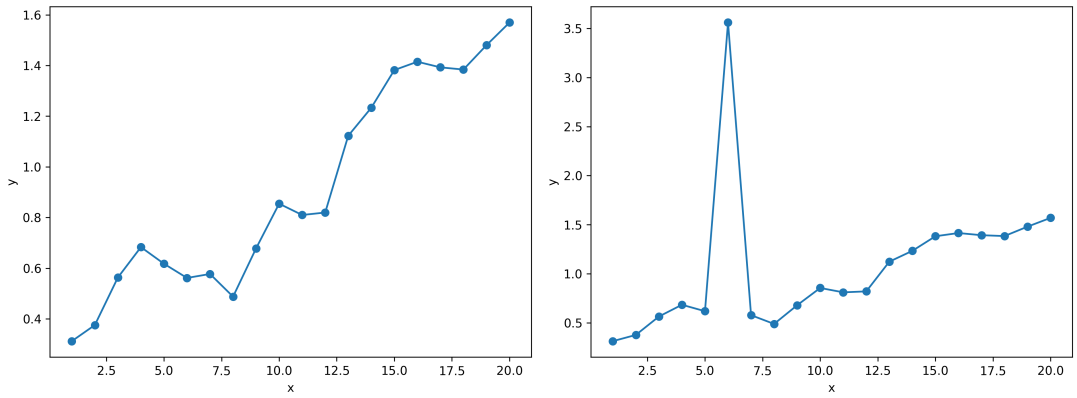
Here:

- 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 14, 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.

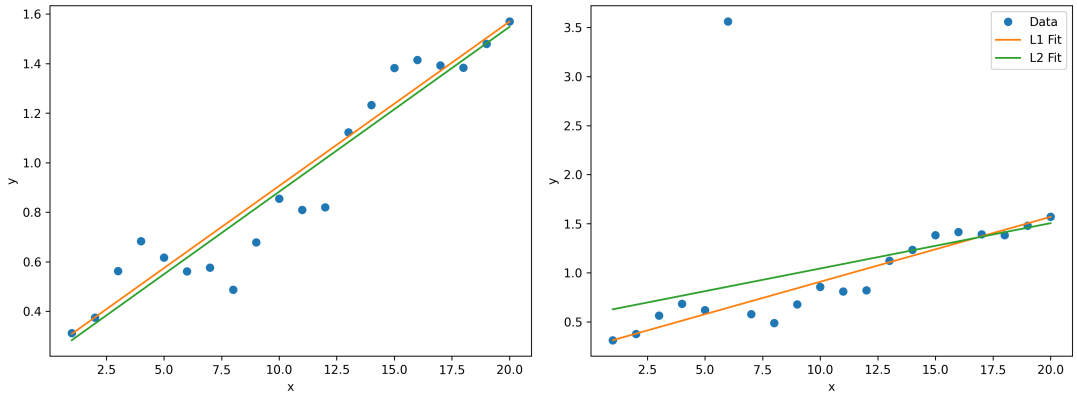


(a) (b)

Figure 14: 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.

2.1.1 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 15 and the estimated line parameters are summarised in Table 1.



(a) (b)

Figure 15: 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.

2.1.2 Discussion

In the absence of outliers, both L_2 and L_1 norms provide comparable results for line fitting. 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 and Underdetermined Inverse Problems

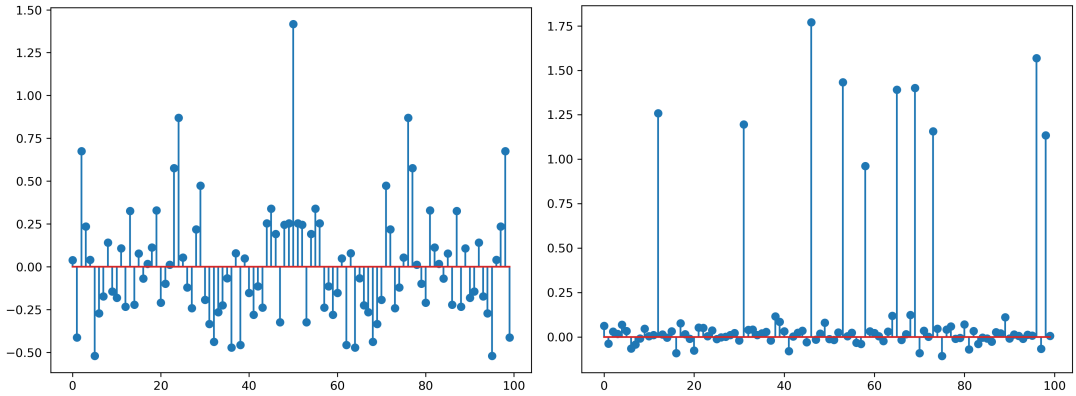
This section addresses signal reconstruction under compressed sensing, which operates on the principle that many signals are sparse in an appropriate basis. Specifically, a signal $x \in \mathbb{R}^n$ is K-sparse if $x = \Psi s$ where Ψ is a basis matrix and s has only K non-zero entries, with $K \ll n$. Compressed sensing allows for signal reconstruction from fewer measurements than the Nyquist rate by utilising a non-uniform or random measurement matrix C, and solving the underdetermined equation:

$$\arg\min_s \|s\|_1 \quad \text{subject to} \quad y = C\Psi s.$$

The L_1 norm minimisation promotes sparsity in s, aiding in accurate reconstruction.

2.2.1 Problem Setup

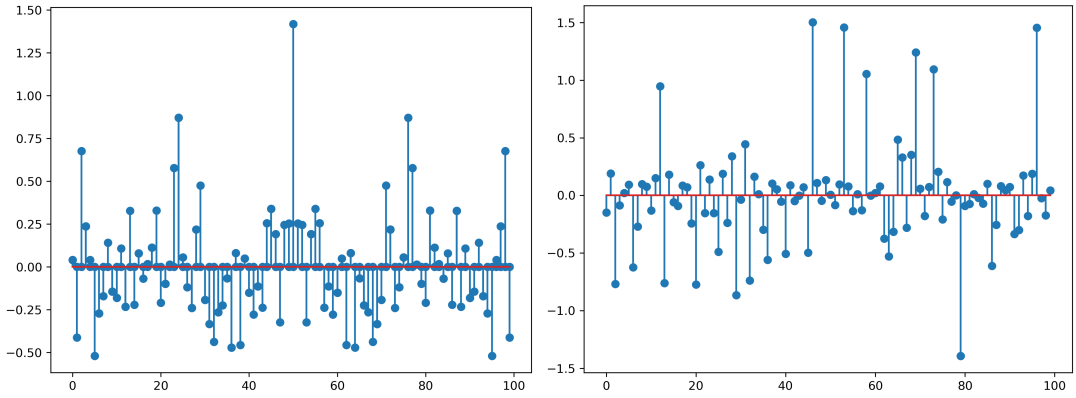
A practical demonstration involves a 1D sparse signal s with 100 components and 10 non-zero coefficients, transformed into the Fourier domain via the discrete fast Fourier transform (DFFT) to yield x. Zero mean Gaussian noise (standard deviation 0.05) is added to simulate realistic conditions. The sparse signal and its Fourier transform are illustrated in Figure 16, where Ψ represents the DFFT matrix.



(a) Frequency domain (b) Time Domain

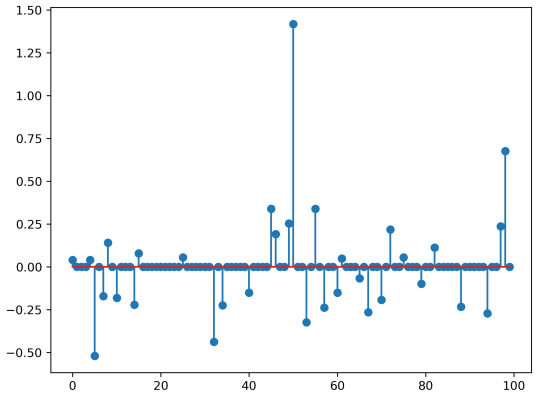
Figure 16: Generated signal expressed in sparse and non sparse domains: (a) Non sparse signal x in the fourier domain, (b) sparse signal s in the time domain.

These signals can be seen in Figure 16. The signal in the non sparse measurement domain, x, is then sub sampled under uniform and random strategies to obtain two sets of 32 measurements y. These sub sampled signals are shown in Figure 17 in both the time and frequency domains. Note that a correction factor is applied to the IDFFT of y to account for the sub sampling reducing the energy of the signal.

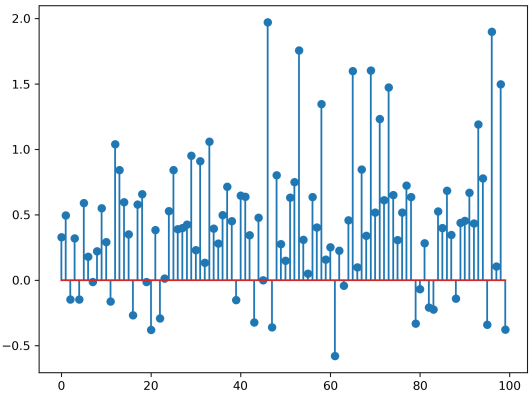


(a) Uniform Sampling - Frequency Domain

(b) Uniform Sampling - Time Domain



(c) Random Sampling - Frequency Domain

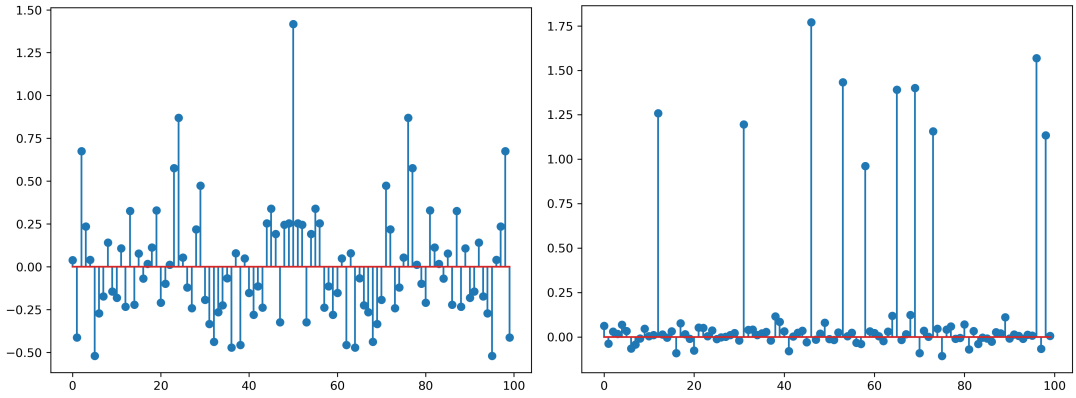


(d) Random Sampling - Time Domain

Figure 17: Comparison of uniformly and randomly subsampled y in both the non-sparse and sparse domains. Top row: Uniform sampling in (a) Fourier domain and (b) time domain. Bottom row: Random sampling in (c) Fourier domain and (d) time domain.

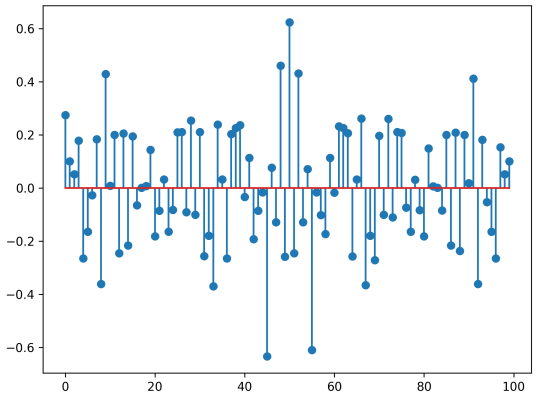
2.2.2 Results

Using iterative soft thresholding and enforcing consistency in the measurement domain, the original sparse signal s is reconstructed from the sub sampled measurements y. The reconstructed signals under uniform and random sampling strategies are shown in Figure 18. The reconstructed signals in the frequency domain are shown in Figure 18.

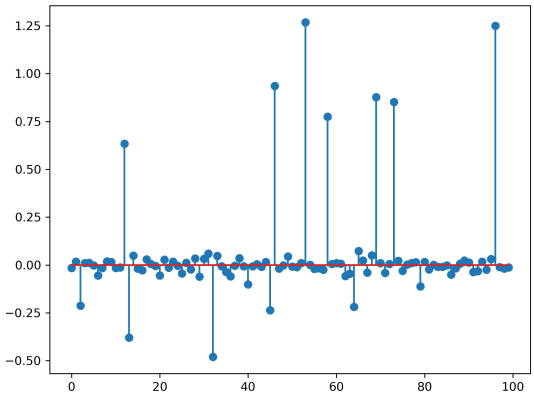


(a) Original - Frequency

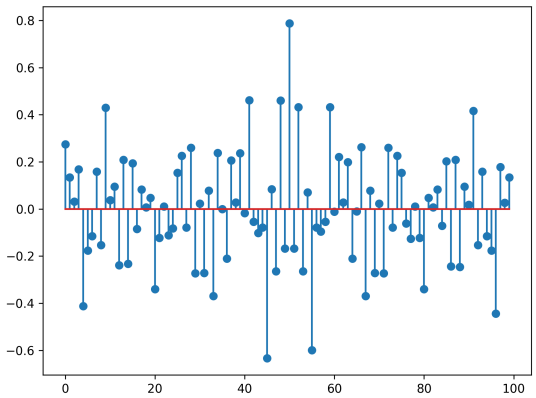
(b) Original - Time



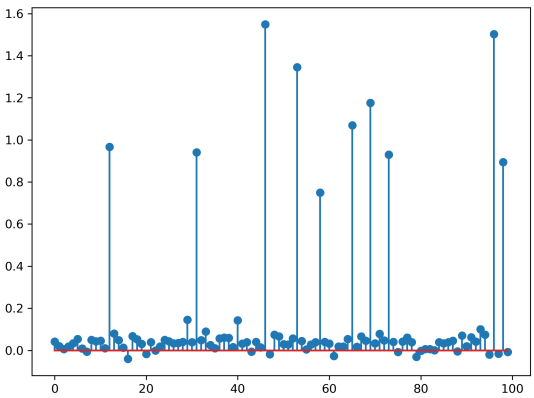
(c) Reconstructed (Uniform) - Frequency



(d) Reconstructed (Uniform) - Time



(e) Reconstructed (Random) - Frequency



(f) Reconstructed (Random) - Time

Figure 18: Comparison of the original and reconstructed signals under uniform and random sampling strategies in both the non-sparse and sparse domains.

According to compressed sensing theory, faithful signal reconstruction requires measurements on the order of $O(K \log(N/K))$, which evaluates to 10 in this instance. Thus, 32 samples should suffice. However, uniform sampling with 32 measurements likely falls below the Nyquist rate, predicting its failure due to non-adherence to the Nyquist criterion and the incoherence condition. In contrast, random sampling likely meets the incoherence condition, enabling more accurate signal reconstruction.

2.3 Part C - Compressed Reconstruction and Multi-resolution Analysis

This question focuses on the reconstruction of the image shown in Fig 19, using a varying subset of its wavelet coefficients. The motivation is to demonstrate how natural images can have sparse representations in the another domain.

Figure 19: River side image.

2.3.1 Discrete Wavelet Transform of the Image

The image is first cropped to remove white space and then processed using the discrete wavelet transform (DWT) with Daubechies wavelets across three levels of frequency decomposition. The wavelet coefficients are then binary thresholded, retaining as '1' only those whose magnitude falls within the top 15%, setting all others to '0'. These thresholded coefficients are displayed in Figure 20, organised hierarchically: the upper-left quadrant contains the low-frequency components, while the other quadrants show higher frequency details in horizontal, vertical, and diagonal orientations. This display underscores the concentration of energy in the low-frequency components and the sparsity of activations in higher frequencies, illustrating the inherent sparsity of natural images in the wavelet domain.

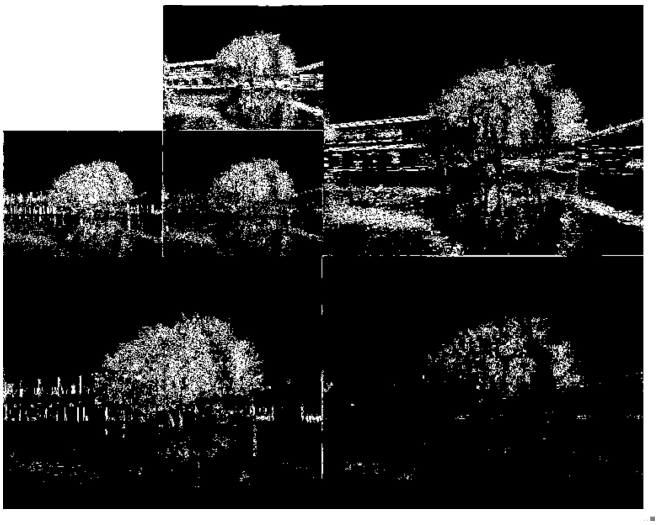
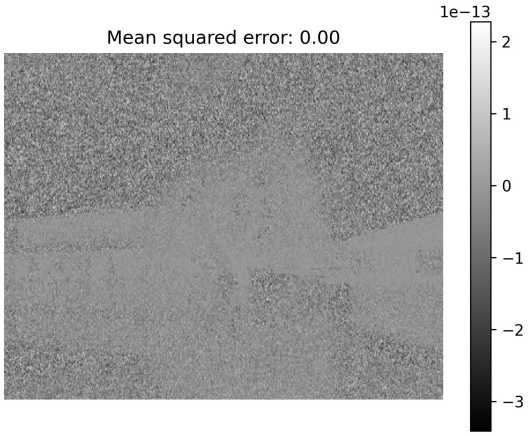


Figure 20: Thresholded Daubechies wavelet transform of the river side image, showing the top 15% coefficients for 3 levels of decomposition.

The image can be reconstructed using the inverse DWT. The original, reconstruction and difference images are shown in Fig 21, the negligible mean squared error between the reconstruction and original image illustrates how near perfect reconstruction can be achieved from the discrete set of wavelet coefficients.

(a) Original Image

(b) Reconstructed Image



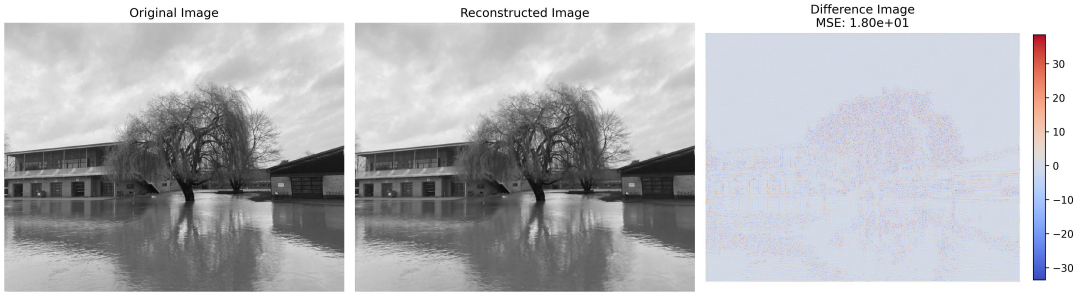
(c) Difference Image

Figure 21: Comparison of the original and reconstructed river side images, along with the difference image.

2.3.2 Image Reconstruction with Selective Coefficient Retention

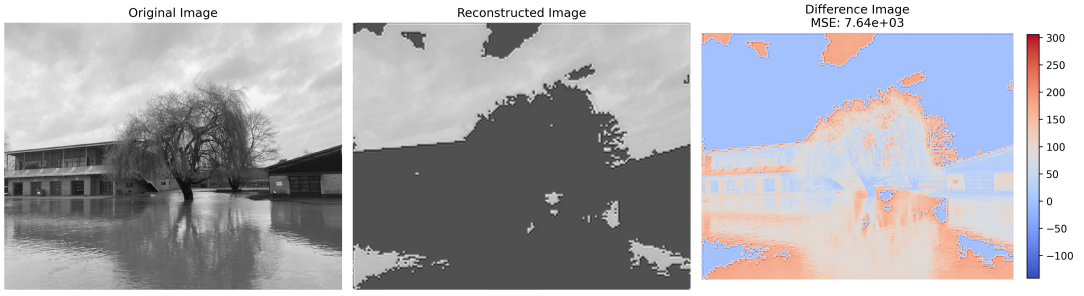
The reconstructed images, displayed in Figure 22, result from retaining varying percentages of the largest wavelet coefficients—20%, 10%, 5%, and 2.5%. A discrete wavelet transform with two levels of decomposition is employed to make the effect of the coefficient retention more pronounced. It is apparent that the quality of reconstruction diminishes as fewer coefficients are retained, particularly affecting areas of the image rich in high-frequency details. Remarkably, even with only 10% of the coefficients preserved, the image reconstruction remains a highly accurate depiction of the original, underscoring the inherent sparsity of natural images in the wavelet domain.

(a) Reconstruction with top 20% of coefficients retained



(b) Reconstruction with top 10% of coefficients retained

(c) Reconstruction with top 5% of coefficients retained



(d) Reconstruction with top 2.5% of coefficients retained

Figure 22: Reconstruction of the river side image using varying levels of coefficient retention.

3 Question 3 - Solving inverse problems

3.1 Part A - Convergence Rate of Gradient Descent

In this exercise, the function $f : \mathbb{R}^2 \to \mathbb{R}$ is considered, defined as $f(\mathbf{x}) = \frac{1}{2}x_1^2 + x_2^2$. The task is to minimise the function using gradient descent, starting from the initial point $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The goal is to theoretically determine the number of iterations K required to achieve a specified accuracy, defined as $f(x) - f(x_*)$ where x_* is the optimal solution, less than 0.01. This prediction is then compared to computational results.

3.1.1 Proving f is Convex

First the function convexity is verified by considering the Hessian matrix of f:

$$H_f(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

This matrix is constant and has positive eigenvalues (1 and 2) for all $\mathbf{x} \in \mathbb{R}^2$, indicating that f is convex over the entire domain.

3.1.2 Identifying the Lipschitz Constant of the Gradient of f

The Lipschitz constant of the gradient of f is determined by considering the function f with the following gradient:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} x_1 \\ 2x_2 \end{bmatrix}.$$

The Lipschitz constant L is derived through the application of the definition of the Lipschitz continuity property of the gradient of f:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\|_2 \leq L\|\mathbf{x} - \mathbf{x}'\|_2,$$

where $\mathbf{x}$ and $\mathbf{x}'$ represent any two points in the domain of f.

The norm of the gradient difference is calculated as follows:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\|_2 = \left\| \begin{bmatrix} x_1 - x_1' \\ 2x_2 - 2x_2' \end{bmatrix} \right\|_2 = \sqrt{(x_1 - x_1')^2 + 4(x_2 - x_2')^2}.$$

$$\sqrt{(x_1 - x_1')^2 + 4(x_2 - x_2')^2} \leq \sqrt{4} \cdot \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2} = 2 \cdot \|\mathbf{x} - \mathbf{x}'\|_2.$$

To confirm rigorously that any constant $L < 2$ does not satisfy the Lipschitz continuity property, specific vectors $\mathbf{x}$ and $\mathbf{x}'$ are considered such that the inequality is challenged. Specifically, if $\mathbf{x} = \begin{bmatrix} 0 \\ x_2 \end{bmatrix}$ and $\mathbf{x}' = \begin{bmatrix} 0 \\ x_2' \end{bmatrix}$ are selected, where x_2 and x_2' are any real numbers, the following is observed:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\|_2 = 2|x_2 - x_2'| = 2\|\mathbf{x} - \mathbf{x}'\|_2.$$

$$2\|\mathbf{x} - \mathbf{x}'\|_2 > L\|\mathbf{x} - \mathbf{x}'\|_2 \quad \forall L < 2.$$

Therefore, it is shown that for $L < 2$, the inequality does not hold.

3.1.3 Finding the Optimal Solution

The next step is to identify the optimal solution $\mathbf{x}^*$ of the function f by solving the gradient equation. $\nabla f(\mathbf{x}) = \mathbf{0}$. The gradient of f is given by:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} x_1 \\ 2x_2 \end{bmatrix}.$$

The single solution to this system of equations are $x_1 = 0$ and $x_2 = 0$, thus there is a single optimal solution at the origin $\mathbf{x}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

3.1.4 Theoretical Convergence Rate

Now that f is confirmed to be convex and the Lipschitz constant of the gradient is determined, the following bound for a convex and L-smooth can be used:

$$f(\mathbf{x}_K) - f(\mathbf{x}_*) \leq \frac{L\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2K},$$

Thus, the above inequality can be rearranged to give a bound on the number of iterations K to achieve the desired accuracy.

$$K \leq \frac{L\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2(f(\mathbf{x}_K) - f(\mathbf{x}^*))}$$

Substituting the values of $L = 2$, $\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 = 2$ and $f(\mathbf{x}_K) - f(\mathbf{x}^*) = 0.01$ into the equation, the theoretical number of iterations required to achieve the specified accuracy is calculated as $K \leq 200$.

3.1.5 Practical Convergence Rate

To practically determine the number of iterations required to achieve the specified accuracy, the gradient descent algorithm is implemented in Python with a learning rate of $\alpha = \frac{1}{L} = \frac{1}{2}$. The algorithm is run starting at $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and iterated until the difference between the function at the current point and the optimal solution is less than 0.01. The number of iterations required to achieve this accuracy is found to be 3, which while consistent with the theoretical prediction, is two orders of magnitude lower.

3.1.6 Strong Convexity of the Function

The primary reason for the observed convergence rate being significantly lower than the theoretical bound is that the strong convexity of the function has not considered. A function f is strongly convex with parameter μ if the following holds:

$$\nabla^2 f(\mathbf{x}) \succeq \mu \mathbf{I} \quad \forall \mathbf{x} \in \mathbb{R}^2,$$

where $\nabla^2 f(\mathbf{x})$ is the Hessian matrix of f and $\mathbf{I}$ is the identity matrix. The Hessian matrix of f is given by:

$$H_f(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

Which is equivalent to stating that $H_f(\mathbf{x}) - \mu \mathbf{I}$ must be positive semi-definite for all $\mathbf{x} \in \mathbb{R}^2$. The smallest value of μ that satisfies this condition is 1.

Now that f is confirmed to be strongly convex with $\mu = 1$, the following bound for a strongly convex and L-smooth function can be used:

$$\|\mathbf{x}_K - \mathbf{x}^*\|_2^2 \leq \left(1 - \frac{\mu}{L}\right)^K \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2$$

This can be seen to give a convergence rate of $\mathcal{O}\left(\frac{1}{\epsilon}\right)$, where ϵ is the desired accuracy. In this case the convergence rate order evaluates to approximately 5 which is more consistent with the practical results.

3.2 Part B - Data Driven and Model Driven Reconstruction

This exercise compares the performance of data-driven and model-driven approaches for solving the image reconstruction inverse problem. Here the goal is to minimise the objective function:

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\nabla \mathbf{x}\|_1 ,$$

where in this context $\mathbf{A}$ is the CT forward operator, $\mathbf{y}$ is the observed noisy sinogram, $\mathbf{x}$ is the true image to be reconstructed, and λ is the regularisation parameter. The particular choice of regularisation is known as the total variation (TV) regularisation, which promotes sparsity in the gradient of the image. Relevant code for this question can be found in the `src/q3b.py` or `src/coursework_LGD_filled_vj.ipynb`.

3.2.1 Model-Driven Approach

Such a problem would typically be solved using proximal gradient descent however the proximal operator for the total variation regularisation is not straight forward to calculate. However, the proximal operator can be computed under a variable splitting scheme, so the problem is tackled using the alternating direction method of multipliers (ADMM) algorithm. Here a new quantity $\mathbf{z} = \nabla \mathbf{x}$ is introduced and a penalty is added to the objective function to enforce the constraint $\mathbf{z} = \nabla \mathbf{x}$. The ADMM algorithm is then used to solve the problem iteratively and can be summarised as follows:

Algorithm 1 Alternating Direction Method of Multipliers (ADMM)

initialise: $\mathbf{x}_0, \mathbf{z}_0, \mathbf{u}_0, \rho$
define: $L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{z}\|_1 + \mathbf{u}^T(\nabla \mathbf{x} - \mathbf{z}) + \frac{\rho}{2} \|\nabla \mathbf{x} - \mathbf{z}\|_2^2$ ▷ Define the augmented Lagrangian
while not converged **do**
    $\mathbf{x}_{k+1} \leftarrow \min_x L_\rho(\mathbf{x}, \mathbf{z}_k, \mathbf{u}_k)$ ▷ Update x by minimizing the augmented Lagrangian
    $\mathbf{z}_{k+1} \leftarrow \min_z L_\rho(\mathbf{x}_{k+1}, \mathbf{z}, \mathbf{u}_k)$ ▷ Update z via proximal operator of $\|\cdot\|_1$
    $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + \rho(\nabla \mathbf{x}_{k+1} - \mathbf{z}_{k+1})$ ▷ Update the dual variable
    $k \leftarrow k + 1$
end while
return $\mathbf{x}_k, \mathbf{z}_k, \mathbf{u}_k$

3.2.2 Data-Driven Approach

The solution for $\mathbf{x}$ obtained using the ADMM algorithm is compared to the solution obtained using the data-driven approach of learned gradient descent. The learned gradient descent approach involves replacing the proximal operator at each iteration with a neural network that takes as input the arguments of the proximal operator and returns the output. The algorithm can be summarised as follows:

Algorithm 2 Learned Gradient Descent

initialize: $\mathbf{x}_0$, N
define: $f(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ ▷ Define the data fidelity term
for $k = 1$ to N **do**
    $\mathbf{g}_k \leftarrow \mathbf{A}^T(\mathbf{A}\mathbf{x}_k - \mathbf{y})$ ▷ Gradient computation
    $\mathbf{dx} \leftarrow h_k(\mathbf{x}_k, \mathbf{g}_k; \theta_k)$ ▷ Apply learned proximal mapping
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \tau_k \mathbf{dx}$ ▷ Update the solution
end for
return $\mathbf{x}_N$

The initial estimate $\mathbf{x}_0$ is set to be the filtered back projection (FBP) reconstruction of the sinogram $\mathbf{y}$. This provides a reasonable initial estimate for the image reconstruction problem.

3.2.3 Model Architecture

The architecture of the unrolling neural network h_* is a convolutional neural network that processes concatenated tensors of the current iterate x and the gradient u. It comprises three convolutional layers, each with a kernel size of 3×3 and a stride of 1. The first layer transforms 2 input channels to 32 output channels, followed by a Parametric Rectified Linear Unit (PReLU) activation. The second layer continues with 32 channels, also followed by PReLU. The final layer reduces the channel count from 32 to 1. The output of the network is delta dx, used to update the current estimate x_i at each iteration of the learned gradient descent algorithm.

3.3 Training

The neural network is trained in a supervised manner using the Adam optimiser with a learning rate of 10^{-4}, across 2000 epochs. The learned gradient descent algorithm is executed for 5 iterations, employing mean square error as the loss function, calculated between the final estimate x_5 and the true image x^*.

3.4 Results

The results of the model-driven and data-driven approaches are compared in Figure 23. The data-driven approach is observed to produce a reconstruction with a higher level of detail, confirmed by the higher peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM) compared to the model-driven approach and simple FBP reconstruction.

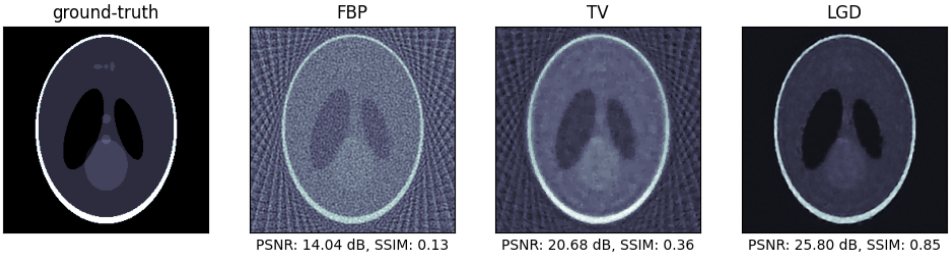


Figure 23: Comparison of the FBP, model-driven, and data-driven reconstructions of the image.

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.