Image Analysis - Coursework

William Knottenbelt, wdk24

Word Count: 2998

1 Image Segmentation

Chest CT

We segment the lungs in chest CT scan (Figure 1) using only the `numpy` package. The pixel intensity in chest CT scans is lower in regions filled with air (lungs and background) than regions of denser tissue, thus we can segment via binary thresholding. We use Otsu thresholding [1] to find the optimal binary threshold which minimizes intra-class intensity variance. This yields the binary image in Figure 2(a). Since lungs contain nodules/tissue, binary thresholding leaves foreground objects in the lung area which need to be removed. We identify all foreground objects with the flood fill algorithm [2], which spreads out from a starting pixel labelling neighboring foreground pixels. We remove those containing < 500 pixels (result in Figure 2(b)). Lungs often contain vessels that are connected to the body, which remain in the foreground. To address this, we perform contour smoothing via binary opening [3] with a circular structuring element. The structuring element first is centered on every pixel, changing the pixel's value to 0 if the element overlaps with any 0's ("erosion"). This is then repeated, except setting pixels to 1's when the structuring element overlaps with any 1's ("dilation"). This gives us the smoothed image in Figure 2(c).

Results We identify the lungs by finding the two largest background regions (with flood fill) that are not connected to the image border. The final segmentation is shown in Figure 3. The mask appears near-perfect, containing all lung nodules/tissue and aligning perfectly with the lung borders. However, there is a central object between the lungs which may be the patient's trachea, but was included in the segmentation.

Tulips

Next, we attempt to segment the purple tulips in Figure 4. Since we are segmenting based on colour, we can cluster the pixels in an appropriate colour space and identify the cluster corresponding to the purple colour to construct the segmentation.

Pre-processing The image is very noisy (Gaussian noise or similar), hence pixels within the same region may occupy different parts of colour space and will not be clustered together. To address this, we use bilateral filtering [4] for its ability to denoise images while preserving edges. We define an 8x8 neighborhood window around each pixel, and assign its new value to be the weighted sum of the neighborhood pixels. The weights are given by the product of two Gaussians centred on the central pixel (one decaying with distance and another decaying with intensity difference). This process is applied to each channel separately, giving the denoised image in Figure 5(a).

KMeans To cluster the pixels based only on colour information, we transform the image to the Lab colour space then remove the Lightness channel, leaving just colour expressions. We cluster the

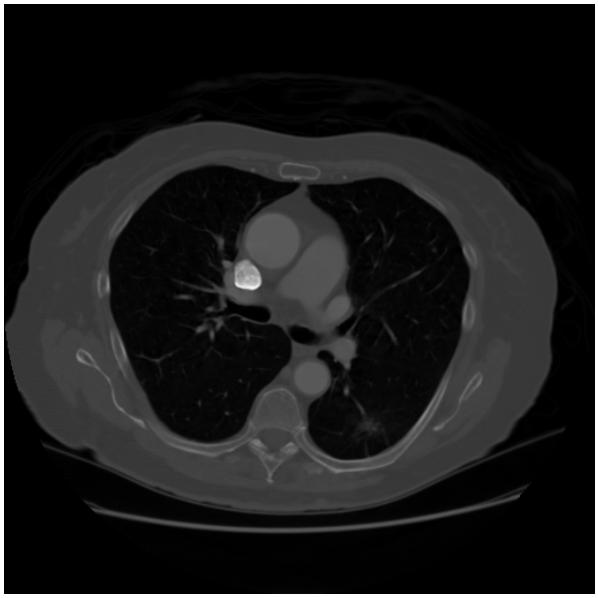

Figure 1: CT image of chest.

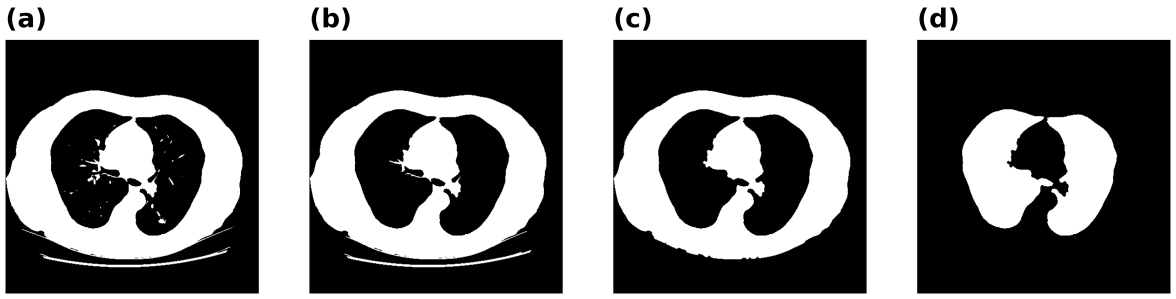

Figure 2: Steps in the segmentation of chest CT: (a) Otsu's threshold, (b) Removing small objects, (c) Binary opening, (d) Final segmentation mask.

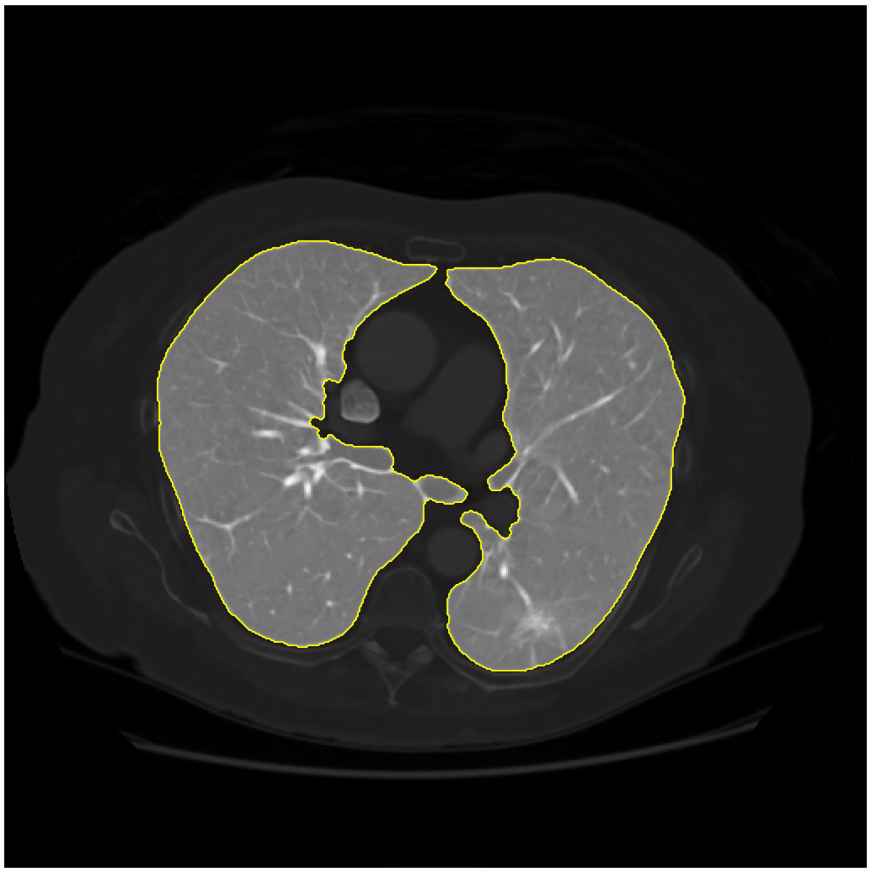

Figure 3: Segmentation of lungs in chest CT image.

Figure 4: Noisy image of tulips

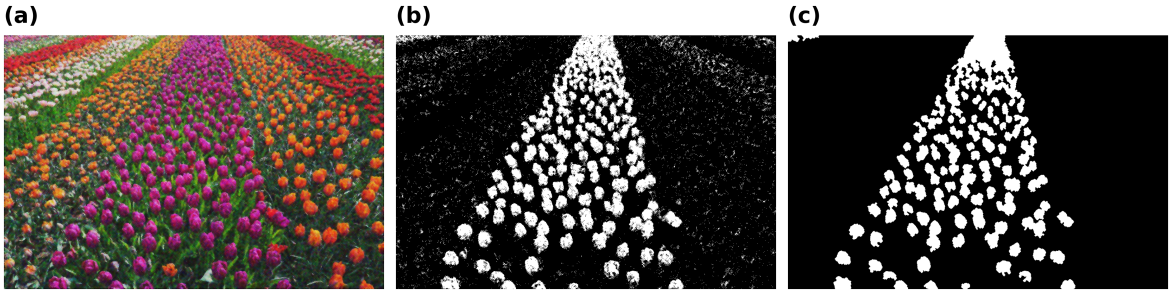

Figure 5: Steps in K-Means segmentation of noisy tulips image: (a) Bilateral filter denoising, (b) K-Means clustering in Lab space, (c) Morphological post-processing

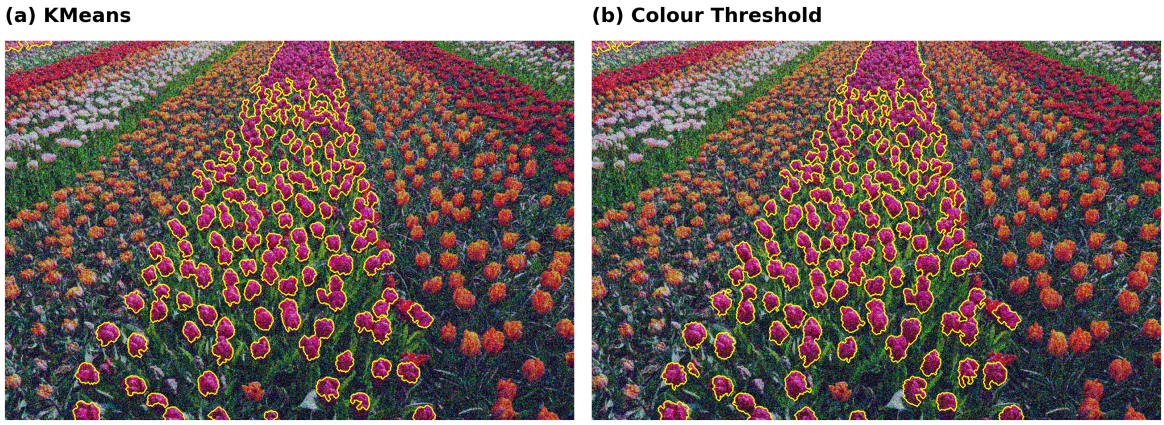

Figure 6: Segmentation results using (a) K-Means clustering and (b) Colour thresholding.

pixels using K-Means [5] by randomly initializing 5 centroids, assigning each pixel to the nearest centroid, and iteratively updating the centroids to the mean of the closest pixels until convergence. We find which centroid is closest to the known purple Lab (without lightness): $a = 34.45$, $b = -4.07$ according to `https://www.e-paint.co.uk/lab-hlc-rgb-lrv-values.asp?cRange=BS+5252&cRef=02+D+45&cDescription=Purple`. The pixels within this cluster form Figure 5(b).

Post-processing Naturally, some pixels will be mislabelled by the clustering. We mitigate this using `skimage`'s `remove_small_objects` and `remove_small_holes`, and use binary opening for contour smoothing. The result is shown in Figure 6(a).

Colour thresholding Another reasonable segmentation method is to transform the image to the HSV colour space and identify the pixels with Hue in the range corresponding to the colour purple. To serve as a comparison, we implement this method with Hue range [0.7, 0.96] and the same pre- and post- processing steps as before (result in Figure 6(b)).

Discussion We see in Figure 6 that both K-Means and Colour-thresholding achieve similarly strong segmentations. For some flowers, K-Means left out the darker shades of purple where Colour-thresholding segments the whole flower. Thresholding based on Hue alone inherently ignores all lightness ("value")

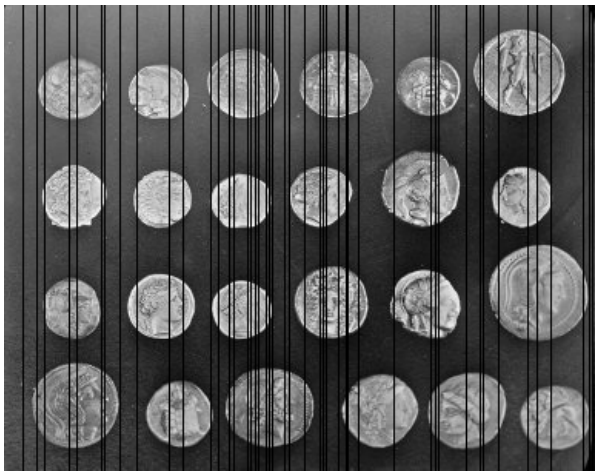

Figure 7: Damanged picture of Coins

and intensity ("saturation") information, capturing all shades of purple. However, this approach requires us to choose an appropriate Hue range, and is very sensitive to this choice. K-Means has no such requirement, but does require us to choose an appropriate number of clusters (though this can be automated using the Elbow method [6]). Overall, we recommend using colour thresholding if manually choosing a Hue range is feasible.

Coins

Finally, we attempt to segment specific coins from the image in Figure 7 (first coin in first row, second in second row, third in third and fourth in fourth). We assume prior knowledge that the grid of coins is 4x6. The colour of some coins is similar to certain regions of the background, but the coins have visually distinct edges, thus we segment via edge detection.

Pre-processing The image is damaged by vertical aberrations so we perform median filtering for its ability to remove aberrations while preserving edges. We define a 4x4 window around each pixel, and replace the pixel's value with the median of the windowed pixels, yielding the image in Figure 8(a).

Segmentation We use the canny edge detector [7] since it is powerful edge detection method and is robust to noise. This smooths the image with a Gaussian filter, calculates the image intensity gradient, removes pixels with non-maximum gradient magnitude and finds the edges by hysteresis thresholding (result in Figure 8(b)). We then fill in the coins using `scipy`'s `binary_fill_holes`, then remove left-over edges with `skimage`'s `remove_small_objects`, leaving just the segmented coins in Figure 8(c). We identify the positions of each coin using the centroids of the foreground objects in the segmentation mask. Knowing that the grid is 4x6, we can label the row and column indices of each coin based on their centroids. We retain only the coins of interest, yielding the successful segmentation in Figure 9.

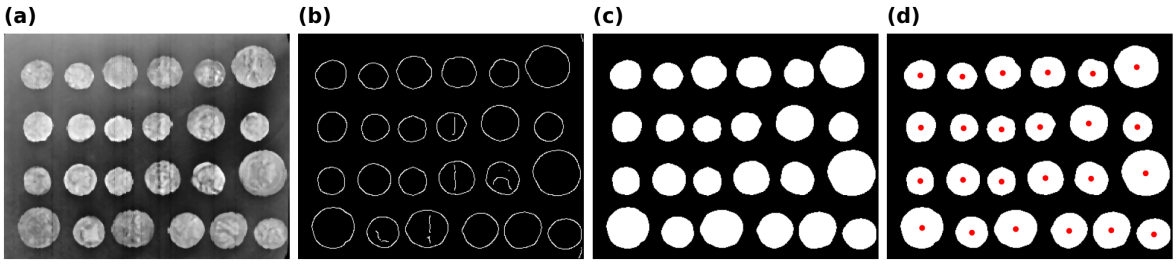

Figure 8: Steps of coins segmentation: (a) Median filtering, (b) Canny edge detection, (c) Filling in holdes and (d) Finding centroids

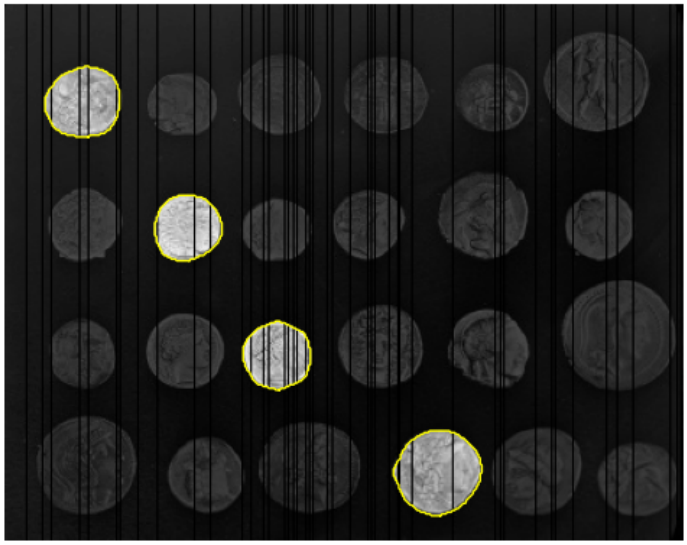

Figure 9: Segmentation of coins image.

2 Inverse problems and sparsity

2.1 Line fitting

A linear inverse problem can be represented as

$$\mathbf{y} = A\mathbf{u}, \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^m$ is the observed output, $A \in \mathbb{R}^{m \times n}$ is the known system matrix and $\mathbf{u} \in \mathbb{R}^n$ is the unknown input. The problem is well-posed if a solution: exists, is unique and is stable (varies continuously with y). If $m > n$ then the system is over-determined, and the existence condition can fail when y contains noise.

Fitting a straight line to a dataset $\{(x_i, y_i)\}_{i=1}^m$ is an example of a linear inverse problem with

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

This is an over-determined system, and it is ill-posed if no straight line passes perfectly through all data points. We can deal with this by employing regularization with the ℓ_1 or ℓ_2 norms such that

$$\hat{a}, \hat{b} = \min_{a,b} \|a\mathbf{x} + b - \mathbf{y}\|_{\ell_n}, \tag{2}$$

for $n \in \{1, 2\}$. For ℓ_2, we have the closed form solution:

$$\begin{aligned} a &= \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i\right)^2} \\ b &= \frac{\sum_{i=1}^n y_i - a \sum_{i=1}^n x_i}{n} \end{aligned} \tag{3}$$

For ℓ_1 we can find solutions using linear programming where we minimize $\sum_{i=1}^m v_i$ subject to $v_i \geq y_i - ax_i - b$ and $v_i \geq -y_i + ax_i + b$.

Case study

We fit straight lines using the ℓ_1 and ℓ_2 norms to a noisy dataset and a dataset containing a significant outlier. For ℓ_1, we use the `CVXPY` package with the `CLARABEL` solver. The results are shown in Figure 10.

For the noisy dataset, both fits are extremely similar but the ℓ_2 fit appears marginally better. This is because the ℓ_2 norm is less sensitive to small noisy fluctuations in the data, and thus captures the underlying relationship better when there are no outliers.

For the outlier dataset, the ℓ_2-fitted line is significantly skewed towards the outlier and does not capture the underlying trend as well as the ℓ_1 norm. This demonstrates that the ℓ_2 norm is more sensitive to outliers, making the ℓ_1 norm a better choice in this case.

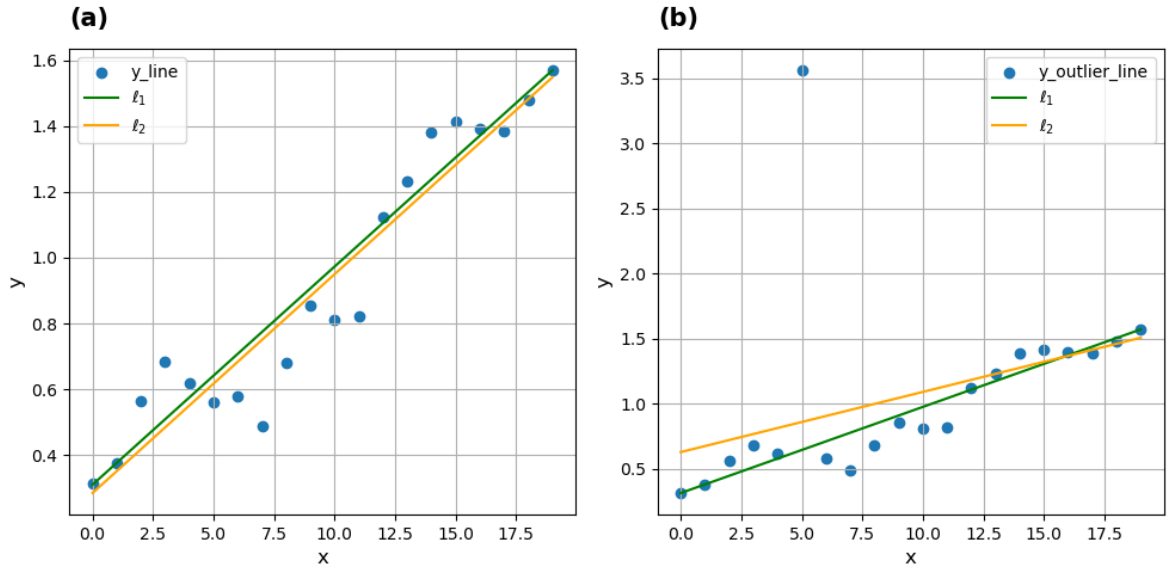

Figure 10: Results of using the ℓ_1 and ℓ_2 norms to fit straight lines to (a) noisy data without outliers and (b) data with outliers.

Algorithm 1 Projection-Over-Complex-Sets Reconstruction

Inputs: Measurement y, Threshold λ.
Initialize $\hat{X}_0 \leftarrow y$
for $i = 1$ to 100 **do**
    Compute inverse Fourier transform $F^{-1}\hat{X}_{i-1}$
    Soft threshold $\hat{x}_i \leftarrow \text{soft}(F^{-1}\hat{X}_{i-1}, \lambda)$
    Compute Fourier transform $\hat{X}_i \leftarrow F\hat{x}_i$
    Enforce consistency with measurement:

$$\hat{X}_i[j] \leftarrow \begin{cases} \hat{X}_i[j] & \text{if } y[j] = 0 \\ y[j] & \text{otherwise} \end{cases}$$

end for

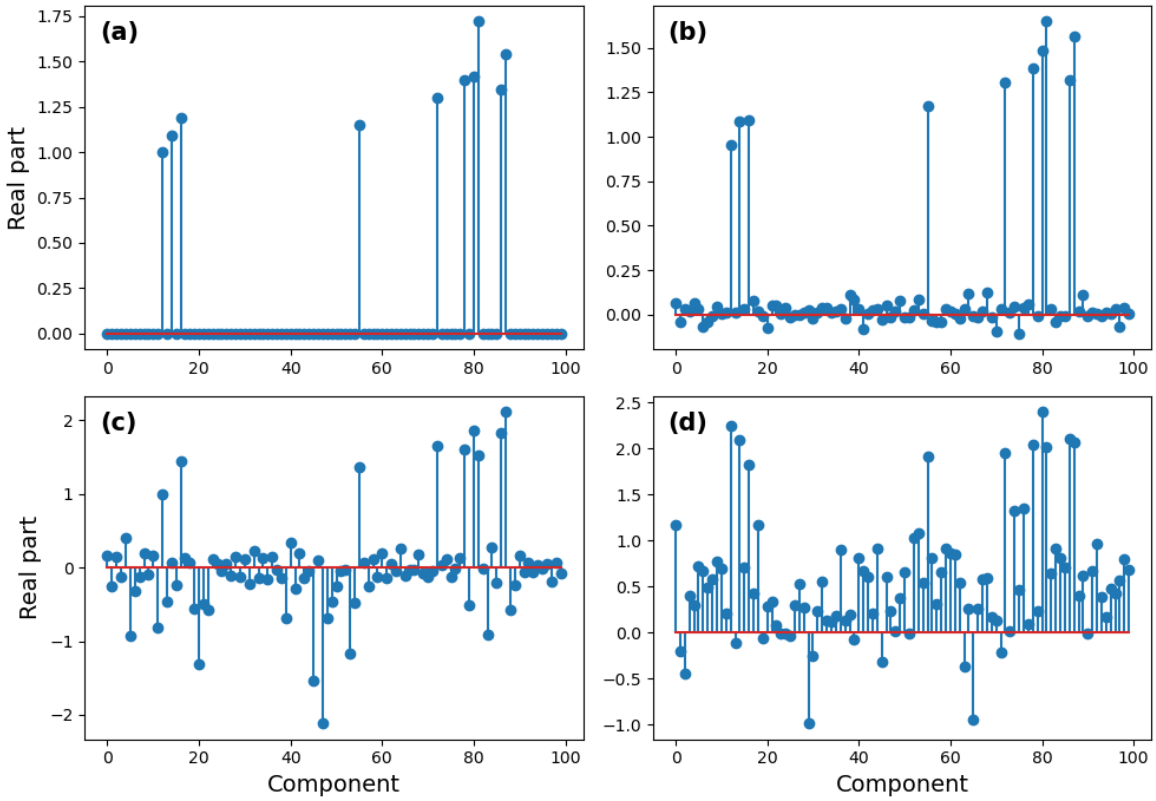

Figure 11: Signals in the time domain: (a) True signal, (b) Noisy signal, (c) 4-fold aliased signal from uniform under-sampling and (d) 4-fold noisy signal from random under-sampling

2.2 Compressed sensing

In this section we conduct a simulation study to illustrate the importance of the measurement strategy in compressed sensing. The Nyquist-Shannon theorem [8] states that the sample rate of a signal must be at least twice the signals bandwidth to avoid aliasing, which results in artifacts in the reconstructed signal. However, if a signal is sparse in some basis that is incoherent with the measurement basis, then compressed sensing allows it to be recovered from measurements below the Nyquist-Shannon sample rate.

We generated a 1D sparse signal $s \in \mathbb{R}^{100}$ with 10 non-zero coefficients (Figure 11(a)) and added Gaussian noise with standard deviation 0.05 (Figure 11(b)).

Denoising of a sparse signal $z \in \mathbb{C}^N$ can be done via ℓ_1 regularization:

$$\hat{x} = \underset{x}{\operatorname{argmin}} \frac{1}{2}\|\hat{x} - z\|_2^2 + \lambda\|\hat{x}\|_1.$$

This has a closed-form solution known as the soft-threshold:

$$\hat{x}_i = \text{soft}(z_i, \lambda) = \begin{cases} \left(1 - \frac{\lambda}{|z_i|}\right) z_i & \text{if } |z_i| > \lambda \\ 0 & \text{if } |z_i| \leq \lambda \end{cases} \tag{4}$$

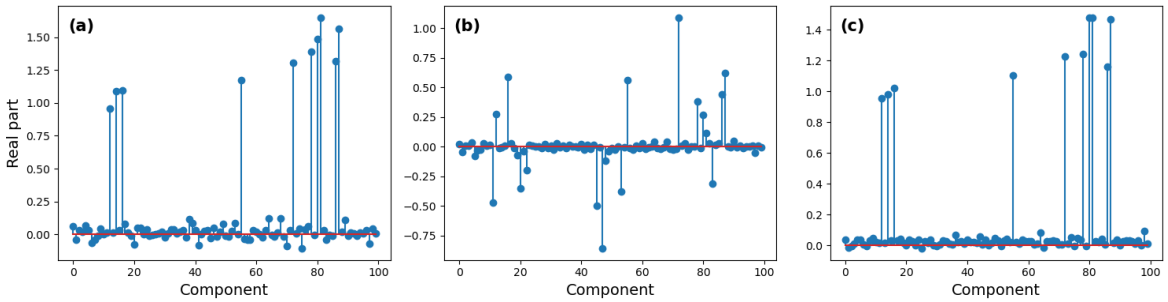

Figure 12: Signals in the time domain: (a) Original noisy signal, (b) POCS reconstruction from uniform under-sampling and (c) POCS reconstruction from random under-sampling.

We simulate two measurement strategies by undersampling the signal in the Fourier domain: $y = F_\Omega s$, where F_Ω represents the composite action of Fourier transform then undersampling. This is an under-determined system, since there are infinitely many signals which are consistent with y. We exploit the fact that s is sparse to reconstruct the signal by solving:

$$\hat{x} = \underset{\hat{x}}{\operatorname{argmin}} \frac{1}{2}||F_\Omega \hat{x} - y||_2^2 + \lambda||\hat{x}||_1$$

There is no closed-form solution for this problem, hence we solve it using algorithm 2.2 [9].

Uniform undersampling Taking 32 uniform (equidistant) samples gives us the 4-fold aliased signal shown in Figure 11(b) (factor of 4 compensates for the undersampling). We see there are significant aliasing artefacts that have similar magnitudes to the true signal components.

Random undersampling Taking 32 random samples gives us the signal in Figure **??**(b), which is extremely noisy but has no significant aliasing artefacts. The compressed-sensing measurement bound [10] tells us that if the measurements are sampled randomly then k-sparse vectors $s \in \mathbb{C}^N$ can be recovered if the number of measurements m is of the order $\mathcal{O}\left(k \ln(\frac{N}{k})\right)$. In this case $N = 100, k = 10 \to k \ln(\frac{N}{k}) \approx 23$, so 32 measurements should be sufficient.

Reconstruction We perform the reconstructions using 100 steps of algorithm 2.2 with $\lambda = 0.05$. The reconstructed signals are shown in Figure 12. For the uniformly undersampling signal, the significant aliasing artefacts were not removed, yielding a poor reconstruction. By contrast, the randomly undersampled signal was successfully denoised, yielding an accurate reconstruction of the true signal. We found a mean squared error of 0.1115 for the uniform reconstruction and 0.0029 for the random reconstruction. This disparity happens because the uniform sampling pattern overlaps with periodic information in the measurement basis, thereby destroying important information and causing aliasing artefacts that are difficult to distinguish from the true signal. Random undersampling does not suffer from this problem and thus captures more information about the true signal, leading to an accurate reconstruction.

2.3 Wavelets

In this section we consider image compression via discrete wavelet transforms (DWT).

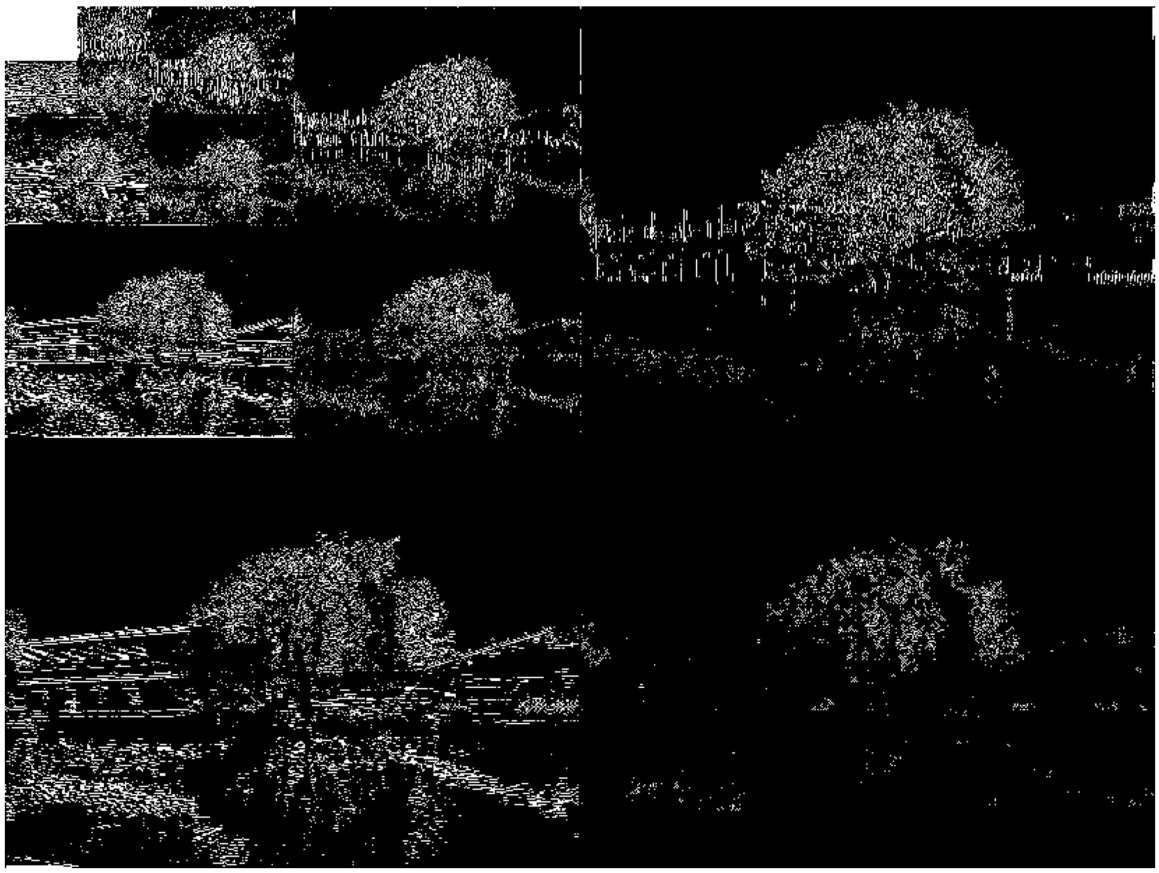

Figure 13: Hierarchical binary visualization of the largest 15% of wavelet coefficients of the river side image.

Daubechies transform

We compute the Daubechies wavelet transform of the image in Figure 14(a), decomposing up to the fourth level. Figure 13 shows a hierarchical visualization of the wavelet coefficients, where the lower-frequency components are shown closer to the top-left. The largest 15% are highlighted for clarity. We see that the significant coefficients are concentrated in the low-frequency levels, since these represent coarse-grained structure of the image which is most important. The significant high-frequency coefficients correspond to fine-grained textures of the tree and surrounding buildings.

We plot the reconstruction in Figure 14, which appears perfect and has a peak-signal-to-noise-ratio (PSNR) of 308 dB and structural similarity (SSIM) of 1, indicating an almost-perfect reconstruction. This illustrates the power of the DWT to represent an image with a finite number of coefficients. In the difference image we can vaguely identify the elements of the image, which tells us that the reconstruction error is non-zero.

A signal is considered "weakly sparse" if its sorted coefficients magnitudes, $|\alpha_i|$, decays as

$$|\alpha_i| \leq Ci^{-1/s},$$

Figure 14: Visualization of the (a) original river side image, (b) reconstruction from the (uncompressed) wavelet transform and (c) magnitude of the difference between them.

for some constant C and compressibility exponent s. In Figure 15 we plot the sorted (a) coefficient magnitudes and (b) log coefficient magnitudes, which is non-linear and thus the coefficients do not follow a power-law decay. Despite this, we clearly see a rapid decay of the coefficient magnitudes, hence the representation is sparse and likely compressible.

Compression

To compress the image, we retain just the largest 15% of coefficients. The reconstruction is shown in Figure 16 and appears near-perfect. The absolute difference image shows that the reconstruction is worst in regions containing many fine details (eg. tree leaves) since the high-resolution detail coefficients have lower magnitudes and were removed.

To investigate other levels of compression, we retained 20%, 10%, 5% and 2.5% of the wavelet coefficients, and plotted the reconstructions in Figure 17. All reconstructions appear almost indistinguishable from the original image except for the 2.5% case, where we observe blurring of tree leaves due to loss of detail coefficients. As we would expect, the PSNR and SSIM both decay as we reduce the proportion of retained coefficients. Generally, having PSNR > 30 dB and SSIM > 0.9 is considered a 'good' reconstruction, which is the case for all compression levels above 2.5% retention.

Discussion These results demonstrate the power of the DWT for image compression. We can retain just 5% of the coefficients and still recover most of the fine details. At the highest compression levels, many of the fine-grained details of the image are lost but the overall structure is maintained. This is the beauty of multi-resolution decomposition: if we do not care about high-resolution details then we can achieve extreme levels of compression and still recover a coarse approximation of the image.

3 Optimization

3.1 Gradient descent

In this section we consider the problem of minimizing the function $f : \mathbb{R}^2 \to \mathbb{R}$ defined as

$$f(x) = \frac{[x]_1^2}{2} + [x]_2^2 \tag{5}$$

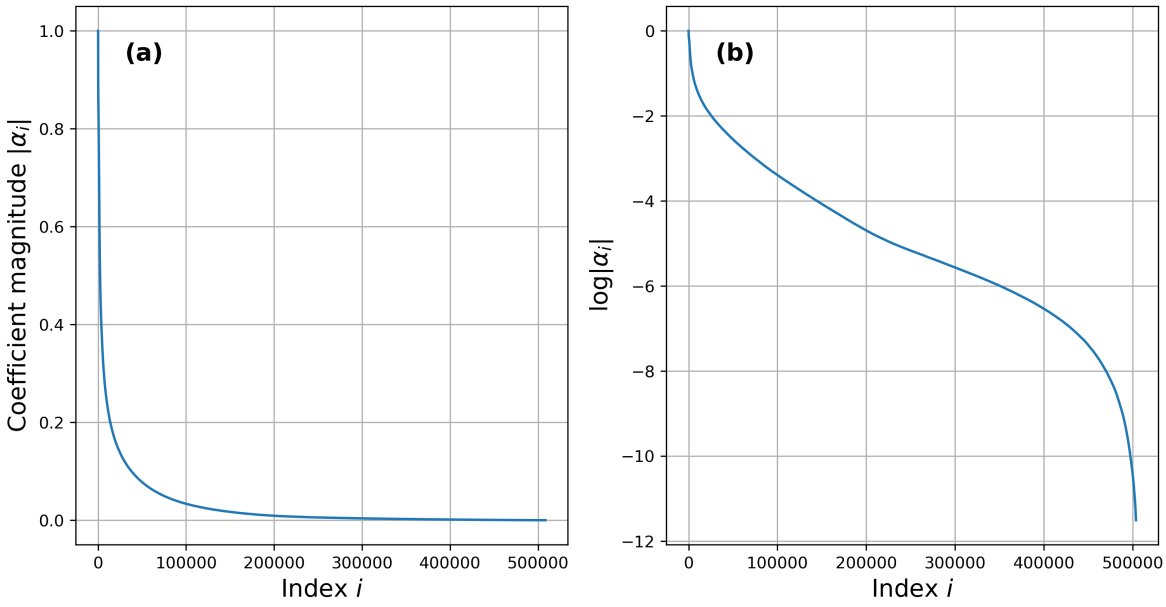

Figure 15: Plot of sorted (a) wavelet coefficient magnitudes and (b) natural logarithm of wavelet coefficient magnitudes, for the river side image.

Figure 16: Visualization of the (a) original river side image, (b) reconstruction from 15% of the wavelet coefficients and (c) magnitude of the difference between them.

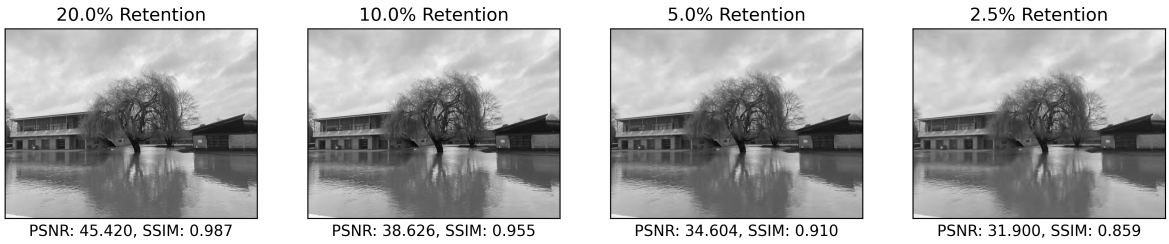

Figure 17: Reconstructions (top row) and absolute difference images (bottom row) from retaining the largest 20%, 10%, 5% and 2.5% of the wavelet coefficients, from the river side image.

Algorithm 2 Gradient Descent

1: **Inputs:** Learning rates α_k, starting point x_0, number of steps K.
2: **for** $k = 0$ to $K - 1$ **do**
3: Compute gradient $\nabla f(x_k)$
4: Update $x_{k+1} \leftarrow x_k - \alpha_k \nabla f(x_k)$
5: **end for**
6: **Output:** x_K

3.1.1 Theoretical guarantees

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its Hessian matrix $[H_f(x)]_{ij} = \frac{\partial^2 f}{\partial [x]_i \partial [x]_j}$ is positive definite, which is satisfied if all eigenvalues are ≥ 1 [11].

Our function (equation 5) has gradient $\nabla f(x) = ([x]_1, 2[x]_2)^T$ and Hessian matrix

$$H_f = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}. \tag{6}$$

This has eigenvalues $1, 2 > 0$, hence the Hessian is positive definite and the function is convex.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth if $||\nabla f(x_a) - \nabla f(x_b)||_2 \leq L||x_a - x_b||_2 \, \forall \, x_a, x_b \in \mathbb{R}^n$, where $\nabla f(x) = (\frac{\partial f}{\partial [x]_1}, \frac{\partial f}{\partial [x]_2}, ..., \frac{\partial f}{\partial [x]_n})^T$ [12].

For equation 5, we have

$$||x_a - x_b||_2 = \sqrt{([x_a]_1 - [x_b]_1)^2 + ([x_a]_2 - [x_b]_2)^2}$$

and

$$\begin{aligned} &||\nabla f(x_a) - \nabla f(x_b)||_2 \\ &= \sqrt{([x_a]_1 - [x_b]_1)^2 + (2[x_a]_2 - 2[x_b]_2)^2} = \sqrt{([x_a]_1 - [x_b]_1)^2 + 4([x_a]_2 - [x_b]_2)^2} \\ &\leq \sqrt{4([x_a]_1 - [x_b]_1)^2 + 4([x_a]_2 - [x_b]_2)^2} = 2\sqrt{([x_a]_1 - [x_b]_1)^2 + ([x_a]_2 - [x_b]_2)^2}). \end{aligned}$$

Hence $||\nabla f(x_a) - \nabla f(x_b)||_2 \leq 2||x_a - x_b||_2$ and f is L-smooth with $L = 2$.

Minimizing a differentiable, convex, L-smooth function via gradient descent (algorithm 3.1) with learning rate $\alpha_k = \frac{1}{L}$ has the error bound [13]:

$$f(x_K) - f(x^*) \leq \frac{L||x_0 - x^*||_2^2}{2K}, \tag{7}$$

where x_0 is the starting point, x^* is the global minimum of f and x_K is the position after K iterations.

By setting $\nabla f(x) = (0, 0)^T$, we trivially find $x^* = (0, 0)^T$. Hence, for gradient descent with a starting point $x_0 = (1, 1)^T$ and learning rate $\alpha_k = \frac{1}{2}$, we see that

$$f(x_K) - f(x^*) \leq \frac{2}{K}. \tag{8}$$

Hence, the number of iterations needed to guarantee accuracy $f(x_K) - f(x^*) \leq 0.01$ is $K = 200$.

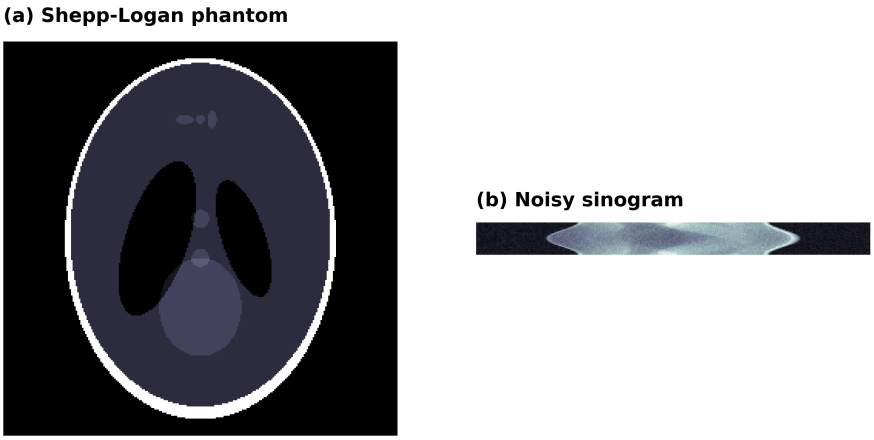

Figure 18: (a) Shepp-Logan phantom and its (b) noisy sinogram.

3.1.2 Observation

Empirically, we find that only 3 iterations were needed to achieve an accuracy of 0.01. This is two orders of magnitude below the upper-bound calculated in part 1. This disparity is related to the convexity of the function.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex if its Hessian satisfies

$$H_f(x) \geq \mu \mathbb{I}, \tag{9}$$

where $\mathbb{I}$ is the identity. The Hessian in equation 6 satisfies this condition with $\mu = 1$ therefore our function satisfies the stricter condition of 1-strong convexity. This leads to faster convergence of gradient descent, and is responsible for our empirically-observed convergence.

In fact, if we use gradient descent (algorithm 3.1) to minimize a μ-strongly convex, L-smooth function, with a learning rate of $\alpha_k = \frac{1}{L}$, then the following inequality holds [13]:

$$||x_K - x^*||_2^2 \leq \left(1 - \frac{\mu}{L}\right)^K ||x_0 - x^*||_2^2. \tag{10}$$

In our case $x_0 = (1, 1)^T$, $x^* = (0, 0)^T$, $\mu = 1$ and $L = 2$, hence $[x]_1^2 + [x]_2^2 \leq 2^{1-K}$. Since $f(x_K) - f(x^*) = \frac{[x_K]_1^2}{2} + [x_K]_2^2$ and $\frac{[x_K]_1^2}{2} + [x_K]_2^2 \leq [x_K]_1^2 + [x_K]_2^2$, we see that $f(x_K) - f(x^*) \leq 2^{1-K}$. This tells us that $K = 8$ iterations guarantees an accuracy of below 0.01, which is the same order of magnitude as our empirical observation and explains the fast convergence.

3.2 CT reconstruction

In this section we consider CT image reconstruction, which is an ill-posed linear inverse problem, $y = Ax + \epsilon$, where y is the observed sinogram, A is the forward Radon transform discretized into a matrix, x is the (unknown) ground truth image and ϵ is random noise.

CT reconstruction is often done via filtered back-projection (FBP), which is an approximate inverse to the Radon transform. However, the quality of FBP reconstructions rapidly declines with the amount of noise present in the sinogram. Variational regularization addresses this by incorporating prior information about the solution through a regularizer. We can formulate the problem as

$$\hat{x} = \underset{x}{\operatorname{argmin}}\, f(x) + g(x), \tag{11}$$

where $f(x) = \|Ax - y\|_2^2$ is the 'data fidelity' and $g(x)$ is the 'regularizer'. Their sum is the 'variational energy'. Under proximal gradient descent (PGD) [14], it is optimized by iteratively performing the updates

$$x_k = \text{prox}_g^\tau(x_k - \tau \nabla f(x_k)) \tag{12}$$

where prox_g^τ is the proximal operator, defined by

$$\text{prox}_g^\tau(u) = \underset{x}{\operatorname{argmin}} \frac{1}{2}\|x - u\|_2^2 + \tau g(x). \tag{13}$$

We investigate two methods to solve CT reconstruction from a noisy sinogram of the Shepp-Logan Phantom (Figure 18).

TV regularization The first method is to use total-variation (TV) regularization $g(\nabla x) = \lambda\|\nabla x\|_1$. The TV-regularizer is such that $\text{prox}_g^\tau(x_k - \tau\nabla f(x_k))$ is not readily available. We reformulate the problem as

$$\min_{x,z} f(x) + g(z), \tag{14}$$

subject to $\nabla x - z = 0$, and optimize it with 200 steps of the Alternating Direction Method of Multipliers (ADMM) [15]. Each step is given [14]:

$$\begin{aligned} x_{k+1} &= \text{prox}_f^\tau(x_k - \sigma^{-1}\tau\nabla^*(\nabla x_k - z_k + u_k)), \\ z_{k+1} &= \text{prox}_g^\sigma(\nabla x_{k+1} + u_k), \\ u_{k+1} &= u_k + \nabla x_{k+1} - z_{k+1}, \end{aligned} \tag{15}$$

where we use step-sizes $\sigma = 2$ and $\tau \approx 0.005$, and initialize $x_0, z_0, u_0 = 0$.

Learned gradient descent The second method is learned gradient descent (LGD) [16], which unrolls proximal Gradient descent by replacing the proximal operator with a neural network that takes x_k and $\nabla f(x_k)$ as input. In our case we use 5 unrolled iterations and fidelity $f(x) = \|Ax - y\|_2$, which has gradient $\nabla f(x_k) = 2A^T(Ax_k - y)$. A forward pass is then given by

$$x_{k+1} = x_k + \tau_k h_{\theta_k}(x_k, A^T(Ax_k - y)), \text{ for } k = 0, 1, 2, 3, 4. \tag{16}$$

The step-sizes τ_k are learned and h_{θ_k} is a 3-layer CNN with learnable parameters θ_k and `PReLU` activations. We initialise x_0 as the filtered back-projection of y. Given the ground truth image x_{GT}, the network is trained with MSE loss

$$\min_{\{\theta_k\}} \frac{1}{N}\|x_5(\{\theta_k\}, y) - x_{GT}\|_2^2, \tag{17}$$

where N is the number of pixels. We train the network on the Shepp-Logan phantom for 2000 epochs, using the Adam optimizer with a learning rate of 0.0001.

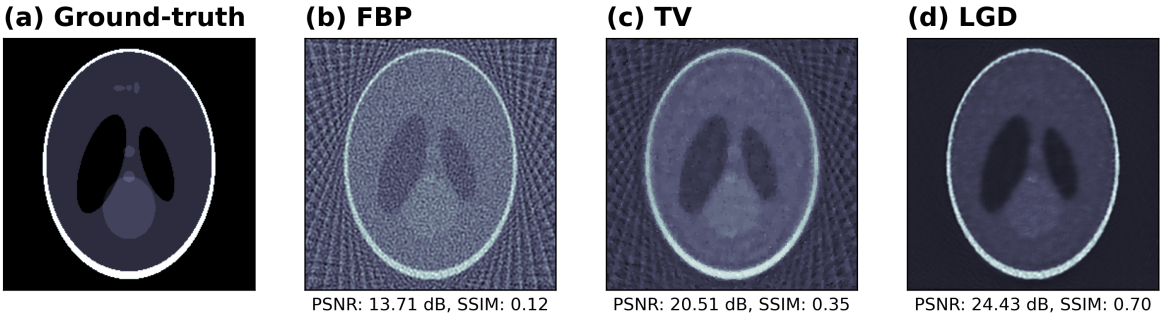

Figure 19: Reconstructions of Shepp-Logan phantom from a noisy sinogram by (a) Filtered back-projection, (b) TV-regularization and (c) Learned gradient descent. In each case the peak-signal-to-noise ratio (PSNR) and structural similarity (SSIM) is provided.

3.2.1 Results

We plot the reconstructions of the Shepp-Logan Phantom in Figure 19. All reconstructions are visibly imperfect, with varying degrees of noise, blurring and artifacts.

FBP achieves the worst results, where the reconstruction exhibits significant noise and artifacts, and achieves a peak-signal-to-noise ratio (PSNR) of 13.98 dB and structural similarity (SSIM) of 0.13. This is because FBP is highly sensitive to noise, which seriously limits its real-world applications. However, it is by far the most computationally efficient approach, making useful for situations where immediate reconstructions are necessary.

The TV-regularization (ADMM) achieves intermediate performance, where the reconstruction has similar but less severe artefacts than the FBP reconstruction, and we get PSNR = 20.73 dB and SSIM = 0.36. This is because TV-regularization encourages sparsity in the image gradient with the ℓ_1 norm, which reduces noise while preserving edges and fine details.

LGD achieves the best results with just light blurring compared to the ground truth and a PSNR of 25.64 dB and SSIM of 0.73. This high performance can be attributed to LGD learning appropriate regularization from the training data. The universal approximation property of neural networks allows them to flexibly capture complex prior information from the training data, which significantly improves the reconstruction but may be impossible to define mathematically as a model-based regularizer.

However, the performance of LGD is reliant on the availability and quality of training data. In this case, we used the same image for training and testing, thus we do not have an unbiased evaluation of the model. The good performance could be a result of overfitting to the Shepp-Logan Phantom, and might not generalise to other sinograms. Distribution shift can be a serious problem in real-world settings, causing lower performance at deployment compared to training. Furthermore, LGD is by far the most computationally intensive method. Although, after training the model, LGD inference (0.226s) was quicker than ADMM (11s), but still slower than FBP (0.006s). Finally, LGD is highly dependent on the choice of network architecture, number of layers and optimizer, while FBP requires no user input and the TV-scheme only requires the choice of regularization weight λ and ADMM step sizes.

Overall, we recommend using LGD if the computational resources and quality training data are available.

References

[1] Nobuyuki Otsu. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1):62–66, 1979.

[2] Alvy Ray Smith. Tint fill. In Proceedings of the 6th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '79, page 276–283, New York, NY, USA, 1979. Association for Computing Machinery.

[3] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Pearson India Education Services, 3 edition, 2017.

[4] 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.

[5] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[6] Robert L. Thorndike. Who belongs in the family? Psychometrika, 18(4):267–276, Dec 1953.

[7] John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8:679 – 698, 12 1986.

[8] C. E. Shannon. Communication in the presence of noise. Proceedings of the IRE, 37(1):10–21, 1949.

[9] Heinz H. Bauschke and Jonathan M. Borwein. On projection algorithms for solving convex feasibility problems. SIAM Review, 38(3):367–426, 1996.

[10] Geethu Joseph and Pramod K. Varshney. Measurement bounds for compressed sensing in sensor networks with missing data. IEEE Transactions on Signal Processing, 69:905–916, 2021.

[11] Dimitri Bertsekas. Convex Analysis and Optimization. Athena Scientific, 2003.

[12] Houshang H. Sohrab. Basic Real Analysis, volume 231. Birkhäuser, 2003.

[13] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, 2004.

[14] N. Parikh and S. Boyd. Proximal algorithms. Foundations and Trends in Optimization, 1(3):123–231, 2014.

[15] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn., 3(1):1–122, 2011.

[16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In Advances in Neural Information Processing Systems, volume 29, pages 3981–3989, 2016.

A Use of auto-generation tools

Github copilot was used to assist with basic programming tasks such as plotting figures and writing docstrings.