1 Laboratory 4: The Radon Transform, the filtered backprojection

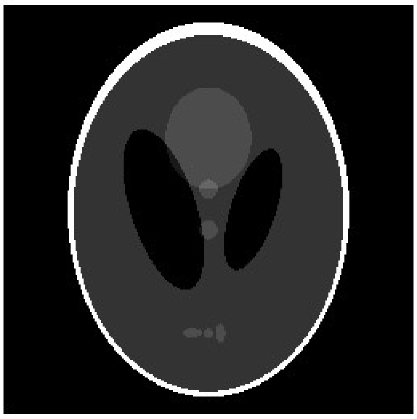
Recall the definition of the Radon transform: The Radon transform of a given function on $\mathbb{R}^2$ is a function defined on the set of all lines of $\mathbb{R}^2$. Every line is parametrized by a normal vector to the line, $\theta \in \mathbb{T}$, and its (signed) distance from the origin $s \in \mathbb{R}$, so that it can be written as

$$\theta_s := \{x \in \mathbb{R}^2 \ : \ x \cdot \theta = s\}$$

Let $f : \mathbb{R}^2 \to \mathbb{R}$, then the Radon transform of f is the function $\mathcal{R}f$ defined as follows:

$$\mathcal{R}f(\theta, s) = \mathcal{R}_\theta f(s) := \int_{\theta_s} f(x)dx.$$

Matlab allows us to compute the Radon transform directly by using the function `radon(F, theta)`. An example is shown on Figures 1. Observe that the inverse radon transform can also been computed by considering `iradon(G, theta)`. In fact, different angles can be considered when computing the inverse Radon Transform, providing different results, see Figures 2b.


(a) Phantom image.

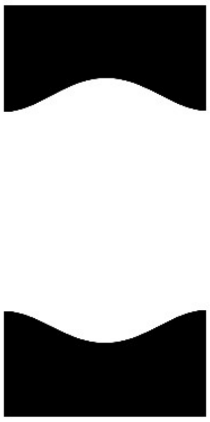
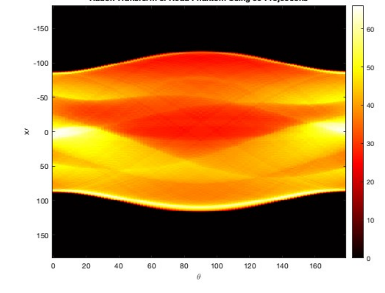

(b) Radon transform of the phantom image.

Figure 1: Representation of the Logan-Shepp phantom test image and its radon transform using command `imshow(F)`.



(a) Radon Transform of Head Phantom Using 90 Projections.

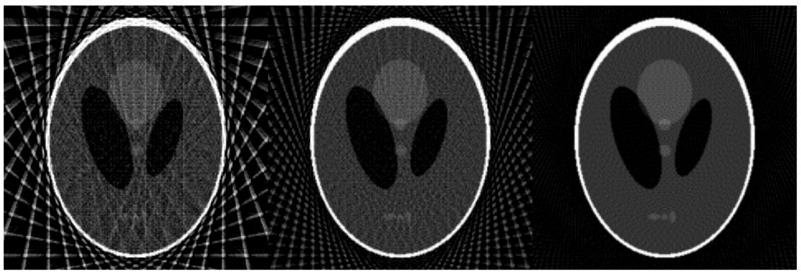

(b) Radon inverse transform of the phantom image.

Figure 2: Representation of the Radon transform and inverse transform of Logan-Shepp phantom test image for different angle ranges.

Exercise 1. *Implement the function* ***iradon***. *The program should take as an input a matrix with columns corresponding to the Radon transform along the angles given in the vector that you pass as second parameter. The output of the program should be a matrix that corresponds to an approximation to the original image.*

The explicit algorithm we need to implement is described as follows:

Step 1 For every direction $\theta_j, j = 1, \ldots, p$ take the discrete convolution

$$h_{j,k} = \Delta s \sum_{l=-q}^{q} v_\Omega(s_k - s_l) g_{j,l} \quad (k = -q, \ldots, q)$$

Here observe that Δs is the grid used verifying $\Delta s \leq 1/(2\Omega)$, and that $g_{j,l} = g(\theta_j, s_l) = \mathcal{R}f(\theta_j, s_l)$, where $\theta_j = \frac{\pi(j-1)}{q}$. Moreover, $\hat{v}_\Omega(\sigma) = \frac{1}{2}|\sigma|^{-1}\hat{\phi}\left(\frac{|\sigma|}{\Omega}\right)$, where $\hat{v}$ represents the 1-dimensional Fourier transform and $\hat{\phi}$ is given by a filter close to 1 when $|\sigma| \leq 1$ and negligent when $|\sigma| > 1$.

Step 2 For each x compute the discrete backprojection using the linear interpolation of the values obtained in Step 1:

$$f_A(x) = \frac{2\pi}{p} \sum_{j=0}^{p-1} (1-\eta) h_{j,k} + \eta h_{j,k+1},$$

where $k = k(j,x) = \lfloor \frac{x\theta_j}{\Delta s} \rfloor$, $\eta = \eta(j,x) = \frac{x\theta_j}{\Delta s} - \lfloor \frac{x\theta_j}{\Delta s} \rfloor$, and $\lfloor a \rfloor$ denotes the integer part of a.

Let us consider the function that takes as parameters `inverse_radon(Rph, q, thetas, step_grid, filter)`, this is,

- `Rph` is the projection data we seek to get the inverse from, $\mathcal{R}f$.
- `q` is the number of different distances we take from the grid. In the code, it may be the maximum values of `rho`.
- `thetas` may be the angle vector determining the range of angle considered, for example, drawing the angles θ_j from the $[0, \pi]$ interval.
- `filter` may determine which filter we use on the computation of the inverse radon transform, $\hat{\phi}$. Initially, it will be the Ram-Lak filter, $\hat{\phi}(\sigma) = \chi_{[0,1]}(\sigma)$, and therefore $v_\Omega(s) = \Omega^2 u(2\pi\Omega s)$, where $u(s) = sinc(s) - \frac{1}{2}\left(sinc(\frac{s}{2})\right)^2$. The implementation of this filter, in a way to use it as stated on *Step 1* can be given as follows

```
function h = ramLakFilter(p, T)
    f = -floor(p/2):T:floor(p/2); % Frequency axis
    omega = 2 * pi * f;
    u = sinc(omega) - 0.5 * (sinc(omega/2)).^2;
    phi_hat = zeros(size(f));
    phi_hat(abs(f) <= p/2) = 1;
    v_Omega = phi_hat .* (omega.^2) .* u;
    h = v_Omega;
end

```

However, a modification for this Ram-Lak filter is also implemented, by setting `filter=RamLak_KS` which provides more accurate results. Moreover, given that we are studying the Logan-Shepp

Head Phantom image, it may make sense to use the Shepp-Logan filter, described as follows: $\hat{\phi}(\sigma) = sinc(\frac{\sigma\pi}{2})\chi_{[0,1]}$ and

$$v_\Omega(s) = \frac{2\Omega^2}{\pi}u(2\pi\Omega s), \quad \text{where } u(s) = \begin{cases} \frac{\pi/2 - ssin\ s}{(\pi/2)^2 - s^2}, & \text{if } s \neq \pm\pi/2, \\ 1/\pi, & \text{if } s \neq \pm\pi/2. \end{cases}$$

On the code provided by the file `LAB4_EX1_complet.m` there are several functionalities, chosen by determining the value of the parameter `example`, as follows:

- For value 1: MATLAB `iradon` function is applied for different angle ranges.
- For value 2: MATLAB `iradon` function is applied for different filters
- For value 3: CUSTOM `inverse_radon` applied function for different filters and ranges
- 4: Alternative CUSTOM `iradon_custom` function with the linear custom interpolation

Experiments 1 and 2 were simply examples in order to see how the MATLAB function behaves. For experiments 3 and 4, different settings are considered. For the first case study, the `Shepp-Logan` filter does not provide satisfactory results, however, the modified `Ram-Lak` computes an inverse randon function quite satisfactory. Different results are obtained when the initial parameters are changed. For the case 4, all three filters used perform great, providing an image close to the one provided by the original MATLAB `iradon` function. Some results obtained are shown on Figures 3.

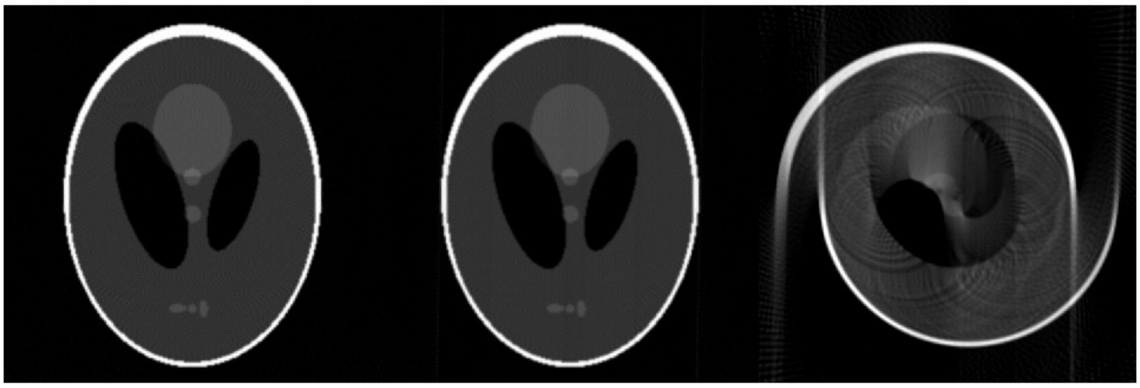


Figure 3: Inverse radon transformation for different filters (Ram-Lak, Shepp-Logan, and Cosine filter, respectively) and for different angle ranges (`0:179`, `0:170`, `0:2:178`, respectively).

References

1. A. C. Kak, Malcolm Slaney, "Principles of Computerized Tomographic Imaging", IEEE Press 1988