CSMA Junior - Session TP

Reconstruction tomographique

John Eric Dufour

Goal of the session:

- Build a (very) simple backprojection function
- Implement the FBP method in 2D to reconstruct a simple image
- Implement an iterative method to reconstruct a simple image
- Visualize the effects of classical experimental issues (noise, dead pixels, ...)
- Compare the two methods and their behaviour.

• Exercice 1: Backprojection

Given the projections of an object on a detector (as in X-Ray Tomography), the reconstructed image problem consists in backpropagating the signal in the appropriate spot in the volume/image.

Objective: Implement a function that compute the reconstructed image from a series of projections.

In order to build a very simple backprojector, the following steps must be performed :

1. Take a projection from our collection
2. Spread the values along the projection direction onto the reconstructed image/volume

Sinograms will be created from the projection of the Shepp-Logan phantom (see function **Shepp-Logan**)

1) Implement the function **backp** which compute the reconstructed image from a sinogram and a list of projection angles.

2) Observe the differences between the phantom and your reconstructed image.

Remark

An example of a working function is given in the file **utilities_CSMA.py**

• Exercice 2: Filtered Backprojection

Objective: Implement a function that compute the filtered backprojection reconstruction method

In order to perform FBP, the following steps must be followed :

1. Take the projections from our collection
2. Filter it by performing a multiplication in Fourier space
3. Backproject the results

Sinograms will be created from the projection of the Shepp-Logan phantom (see function **Shepp-Logan**)

1) One of the classical filters used in FBP is the ramp filter. Filter your sinograms by a ramp function in the frequency domain.

2) Implement an FBP reconstruction function **FBP()** performing a filtering and then a backprojection. Compare your resuts with the unfiltered version.

Remark

An example of a working function is given in the file **utilities_CSMA.py**
Several other filters are implemented in the skimage.transform library (in the function **_get_fourier_filter()**)

• Exercice 3: Noise and other artifacts

Objective: Visualize the influence of noise on the FBP reconstruction

Sinograms will be created from the projection of the Shepp-Logan phantom (see function **Shepp-Logan**)

1) Implement a function which adds noise (white) to the sinogram. Observe the influence of such artifacts on the reconstruction.

2) The detector used in your tomograph is now damaged. As a result, a pixel is always black and the sinogram shows a black line corresponding to this dead pixel. Observe the influence of such artifact on the reconstruction.

3) The experimental setup forces you to use a limited number of angles. Perform the reconstruction using a low (≈ 10) number of projections.

4) The experimental setup forces you to use a limited range of angles. Perform the reconstruction using a small range of projections.

• Exercice 4: ART reconstruction

Objective: Implement a function that compute an iterative reconstruction

The implement method is an iterative technique in which the update is computed as:

$$x^{k+1} = x^k + \lambda * A^T \frac{(p - Ax^k)}{A^T A}$$

The following steps must be followed :

1. Initialize the image x^0
2. For each iteration k:
    - Compute the projection of the image Ax^k
    - Compute the residual $p - Ax^k$
    - Divide by $A^T A$

Sinograms will be created from the projection of the Shepp-Logan phantom (see function **Shepp-Logan**)

1) Implement this function as **Recon_irt()**. Perform a recontruction on the phantom. Compare to the FBP results.

2) Perform the reconstruction on the phantom with artifacts (noise and dead pixels). Compare the results.

3) Perform the reconstruction on the phantom projections using few projections and or a limiited range of angles. Compare the results.

Remark

Going further: these functions are adapted to a python implementation, but they are far from the most effective. The projection or backprojection functions (A and A^T) can be implemented in many different ways, especially on new architectures (CUDA, RTX, ...)