Transformée de Radon

Application à l'imagerie des structures internes

RHZIF Ali

N° SCEI 24624

Session 2025

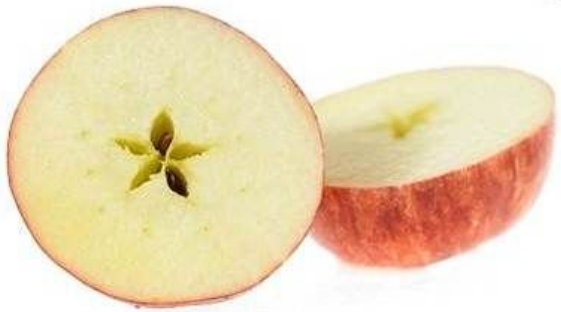
Sommaire

1. Introduction
2. Idée de base : Projections
3. Transformée de Radon
4. Reconstruction analytique d'images
   - Rétro-projection
   - Rétro-projection filtrée
5. Expérience et Reconstruction avec Python et Tomviz

Introduction

Tomographie

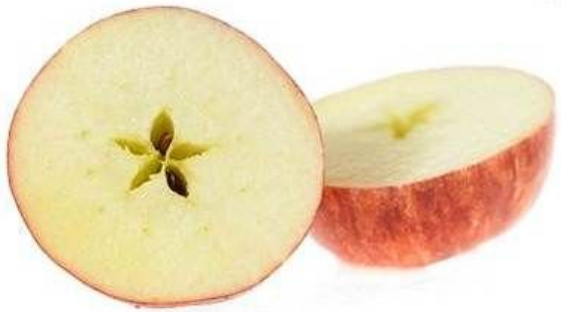
Imagerie en coupe: Le mot tomographie est dérivé des mots grecs tome (coupe) et graphein (écrire).



Introduction

Tomographie

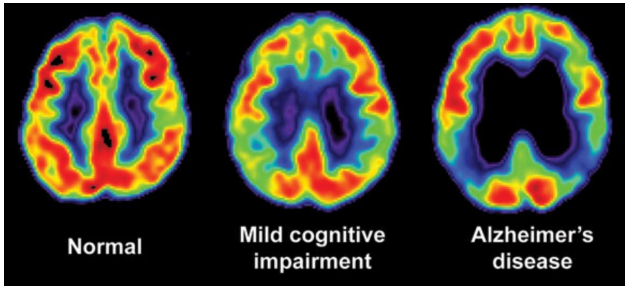
Imagerie en coupe: Le mot tomographie est dérivé des mots grecs tome (coupe) et graphein (écrire).



- Le découpage en tranches n'est pas une bonne idée à réaliser sur des humains.

Introduction

(a) PET-CT



(b) Images cérébrales par TEP

(c) CT: 2D $\longrightarrow$ 3D

Problématique

Comment peut-on reconstruire une image en trois dimensions d'une structure à partir de ses projections bidimensionnelles ?

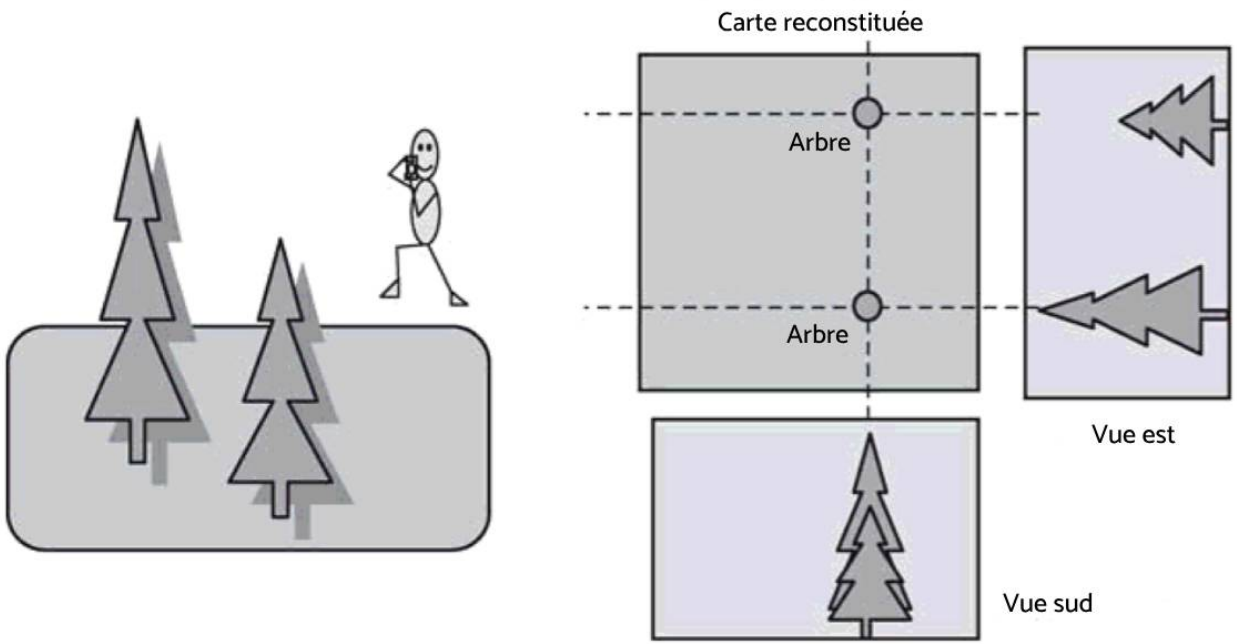
Sommaire

1. Introduction
2. **Idée de base : Projections**
3. Transformée de Radon
4. Reconstruction analytique d'images
   - Rétro-projection
   - Rétro-projection filtrée
5. Expérience et Reconstruction avec Python et Tomviz

Idée de base : Projections

Exemple : Photographie

Deux arbres dans un parc, comment peut-on créer une carte du parc à partir de deux photos prises à l'est et au sud ?

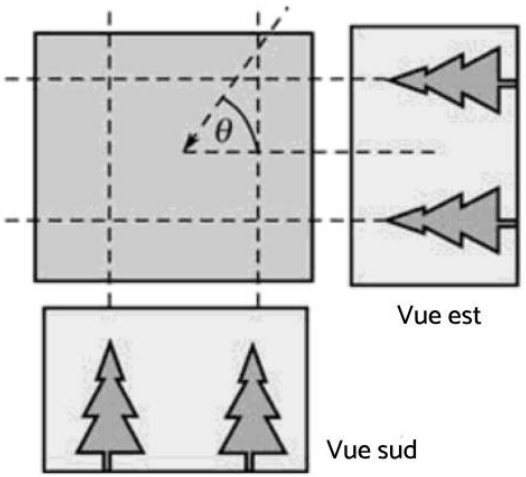


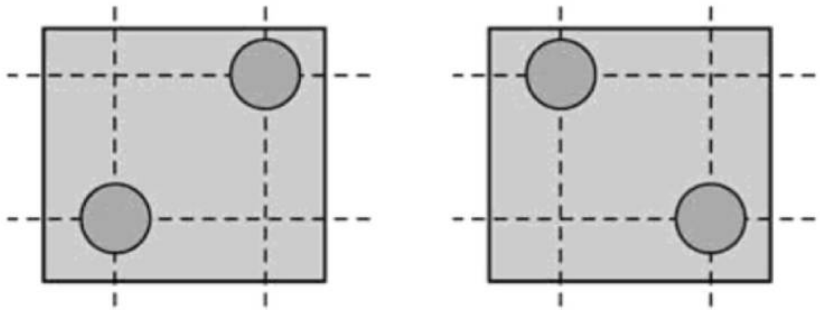
- Une photo est une projection d'un objet sur un plan

Idée de base : Projections

Exemple : Une autre photographie

Autre configuration : Si nous voyons deux arbres distincts sur les deux vues, pouvons-nous reconstruire de manière unique la carte des arbres ?



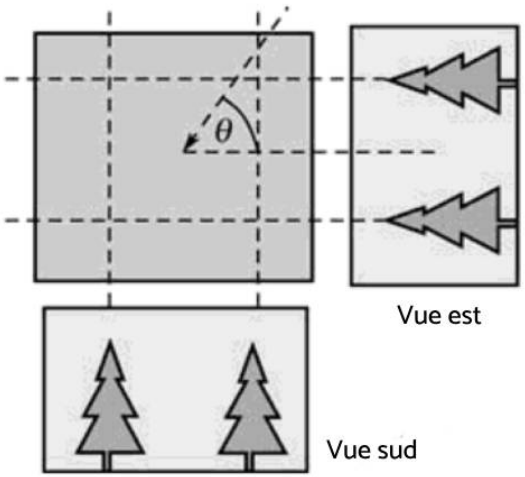


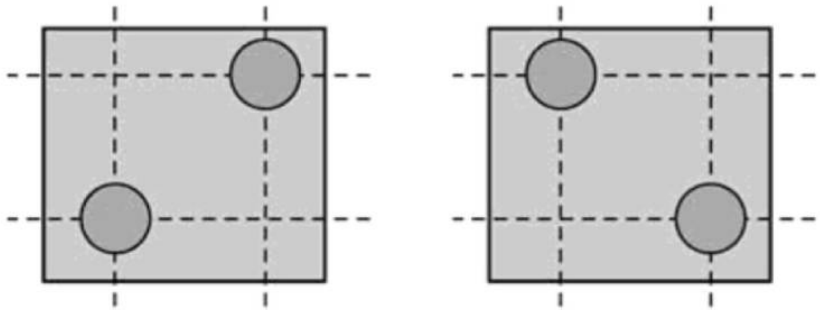
Il existe deux solutions

Idée de base : Projections

Exemple : Une autre photographie

Autre configuration : Si nous voyons deux arbres distincts sur les deux vues, pouvons-nous reconstruire de manière unique la carte des arbres ?





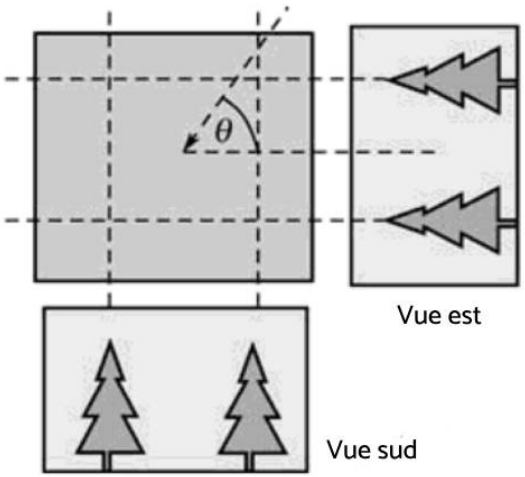
Il existe deux solutions

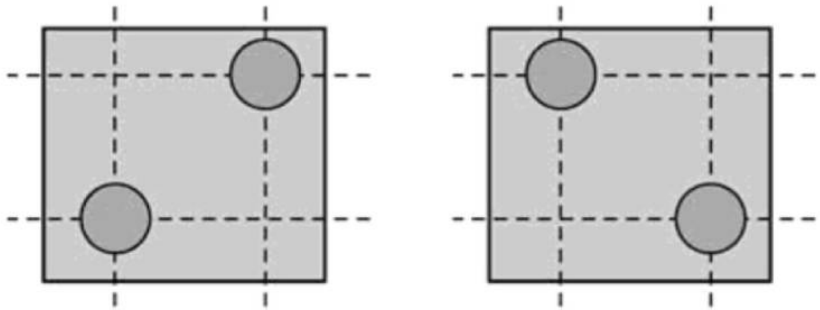
- Ici, nous ne pouvons pas reconstruire la position et la hauteur des deux arbres.

Idée de base : Projections

Exemple : Une autre photographie

Autre configuration : Si nous voyons deux arbres distincts sur les deux vues, pouvons-nous reconstruire de manière unique la carte des arbres ?





Il existe deux solutions

- Ici, nous ne pouvons pas reconstruire la position et la hauteur des deux arbres.
- nous pouvons résoudre l'ambiguïté avec une autre photo à 45°

Sommaire

1. Introduction
2. Idée de base : Projections
3. **Transformée de Radon**
4. Reconstruction analytique d'images
   - Rétro-projection
   - Rétro-projection filtrée
5. Expérience et Reconstruction avec Python et Tomviz

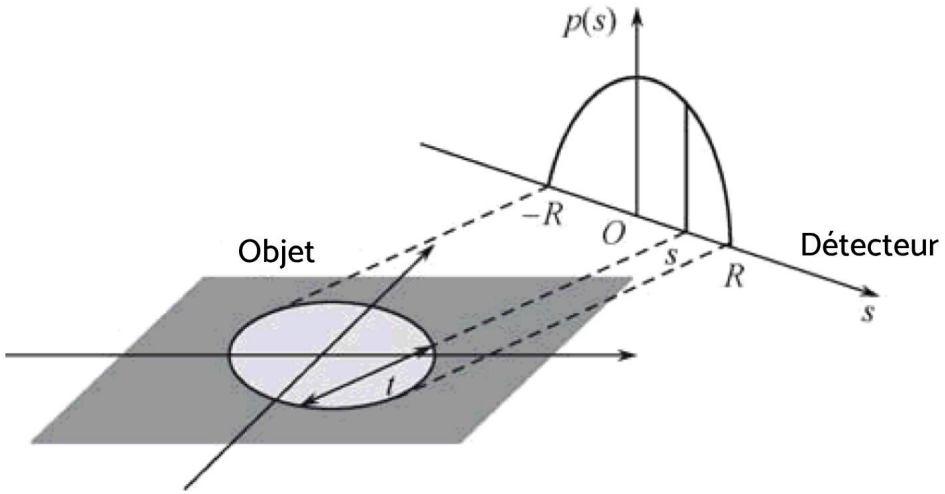
Transformée de Radon

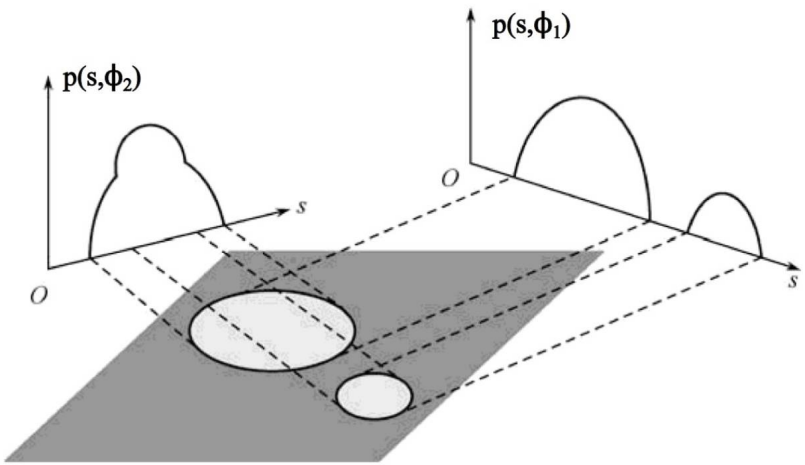
Définition

Si f est une fonction continue sur $\mathbb{R}^2$, on appelle transformée de Radon de f la fonction $\hat{f}$ définie, là où c'est possible, par:

$$p(s,\phi) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\cdot\delta(x\cos\phi + y\sin\phi - s)\,dx\,dy$$

- Fonction delta : L'intégrale est nulle partout sauf sur la droite $L(s,\phi)$

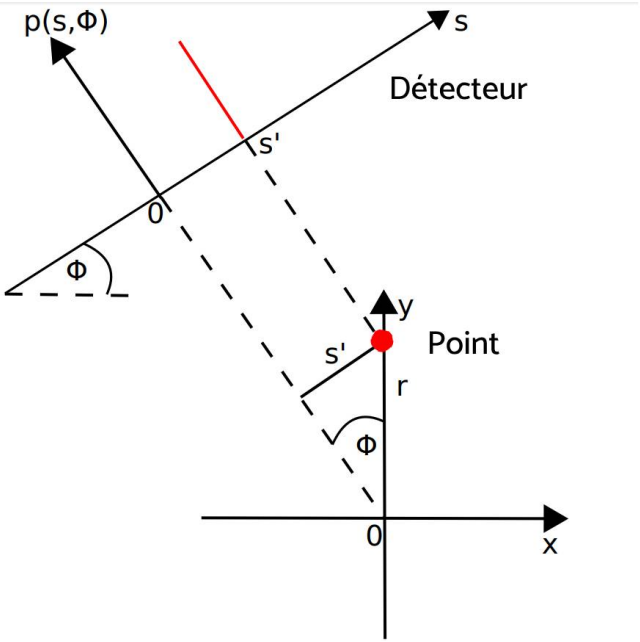


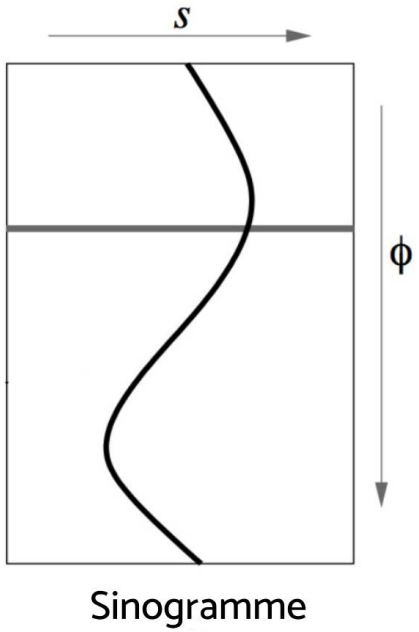


Transformée de Radon

Exemple : Point sur l'axe des y

la coordonnée s du pic sur le détecteur 1D est $s = rsin\phi$.
La projection $p(s, \phi)$ dans le système de coordonnées $s - \phi$ est une fonction sinusoïdale.





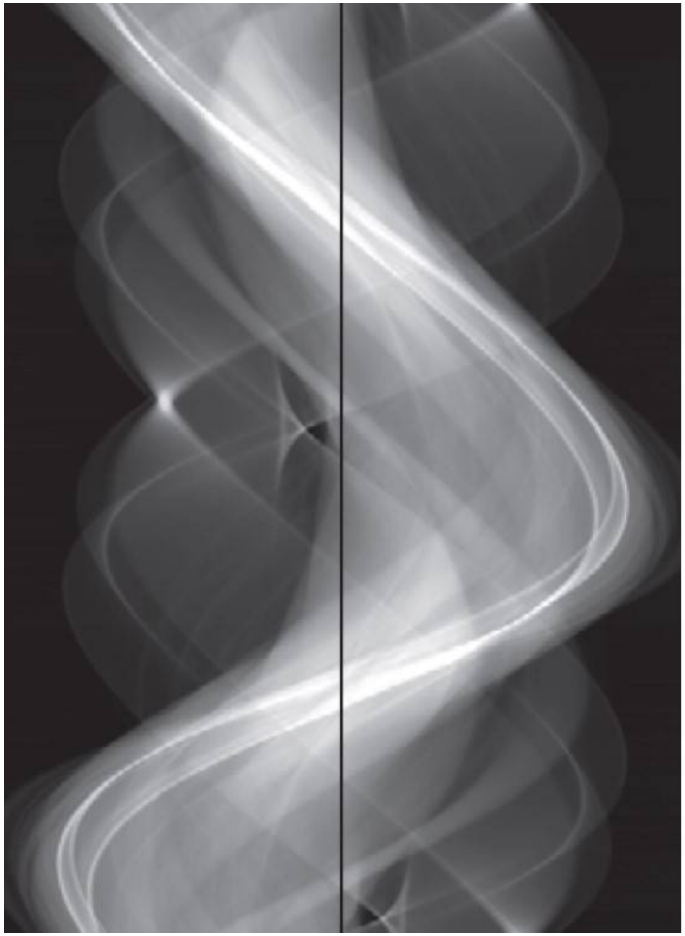
Sinogramme

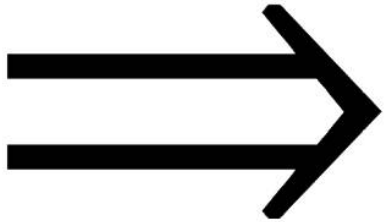
Un sinogramme est une représentation des projections sur le plan $s - \phi$.

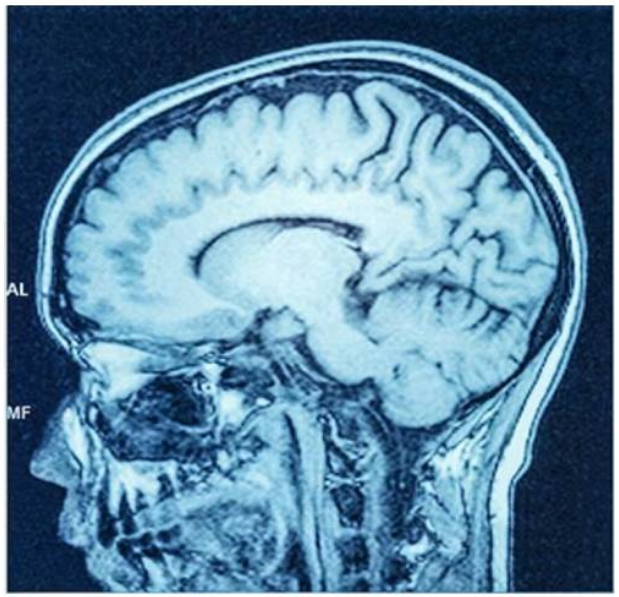
Transformée de Radon

Tomographie

Nous voulons trouver l'image $f(x, y)$ à partir des projections mesurées $p(s, \phi)$.





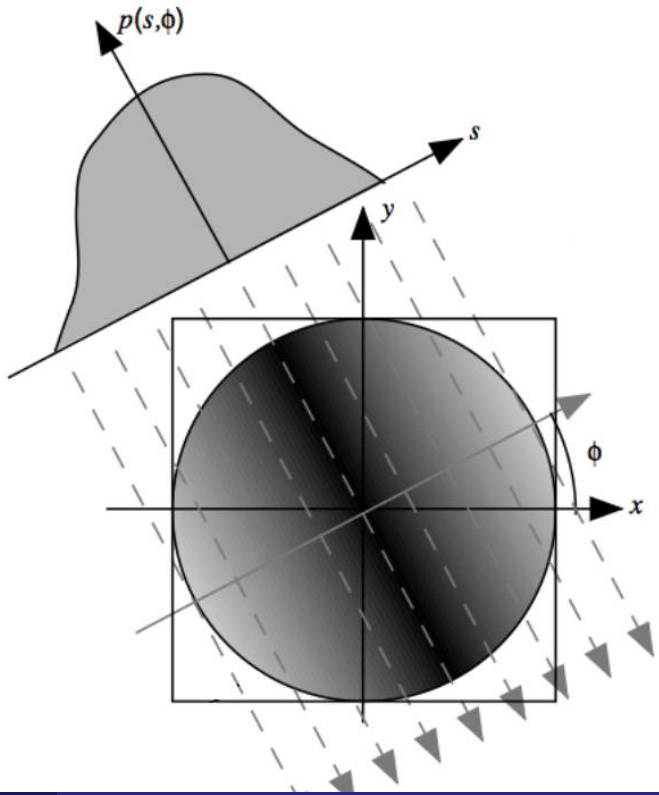


Sommaire

1. Introduction
2. Idée de base : Projections
3. Transformée de Radon
4. **Reconstruction analytique d'images**
   - Rétro-projection
   - Rétro-projection filtrée
5. Expérience et Reconstruction avec Python et Tomviz

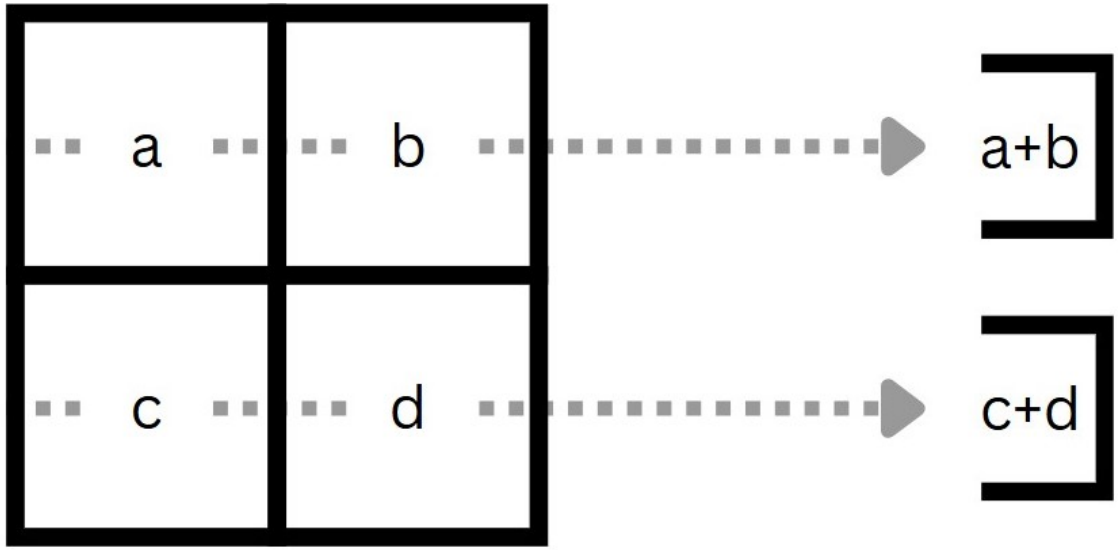
Procédure de rétroprojection

- Nous souhaitons replacer les valeurs de $p(s, \phi)$ dans la position de la ligne de projection appropriée.
- Mais la connaissance de l'origine des valeurs a été perdue lors de l'étape de projection
- Le mieux que nous puissions faire est d'attribuer une valeur constante à tous les éléments le long de la ligne.



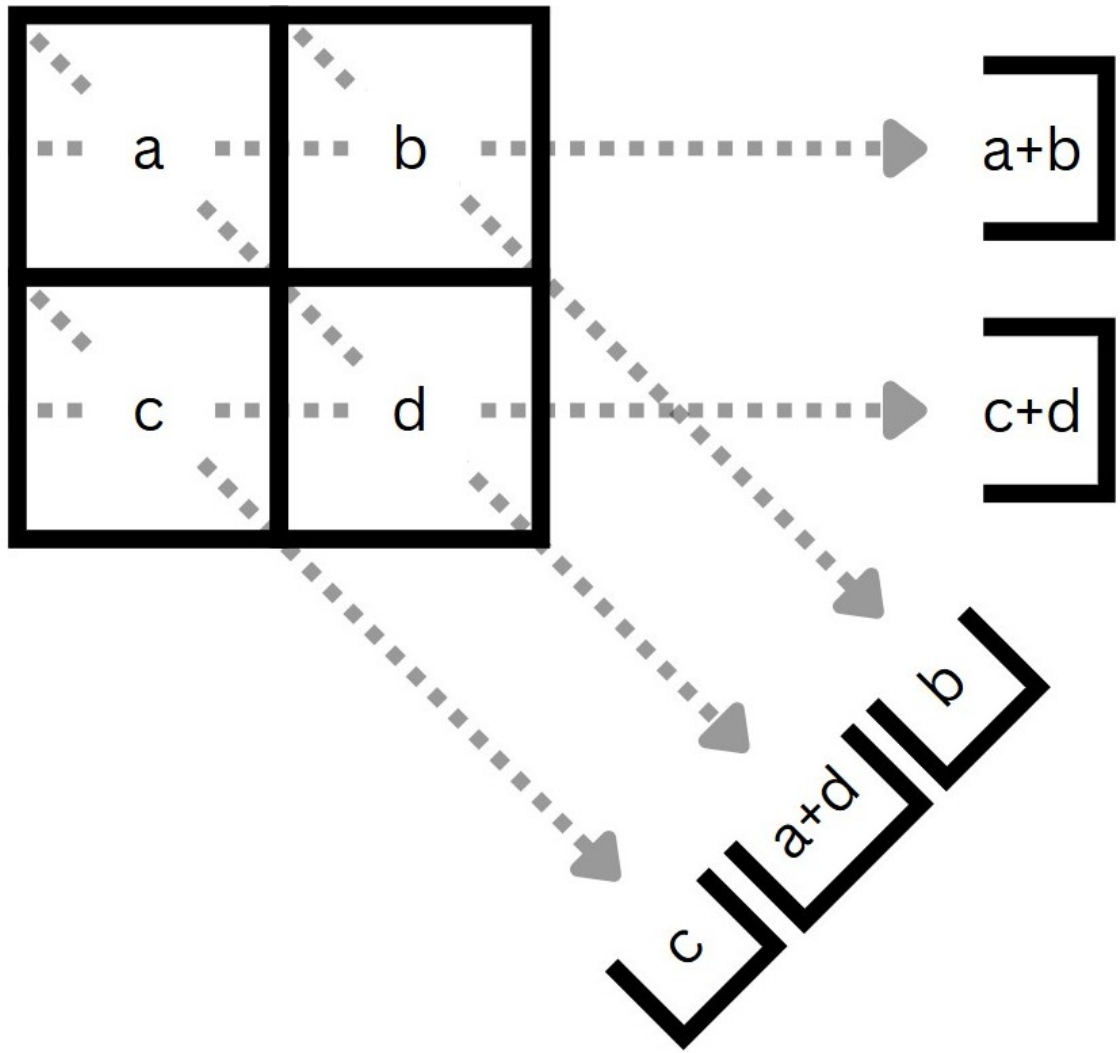
Exemple de rétroprojection

- $1^{ère}$ projection



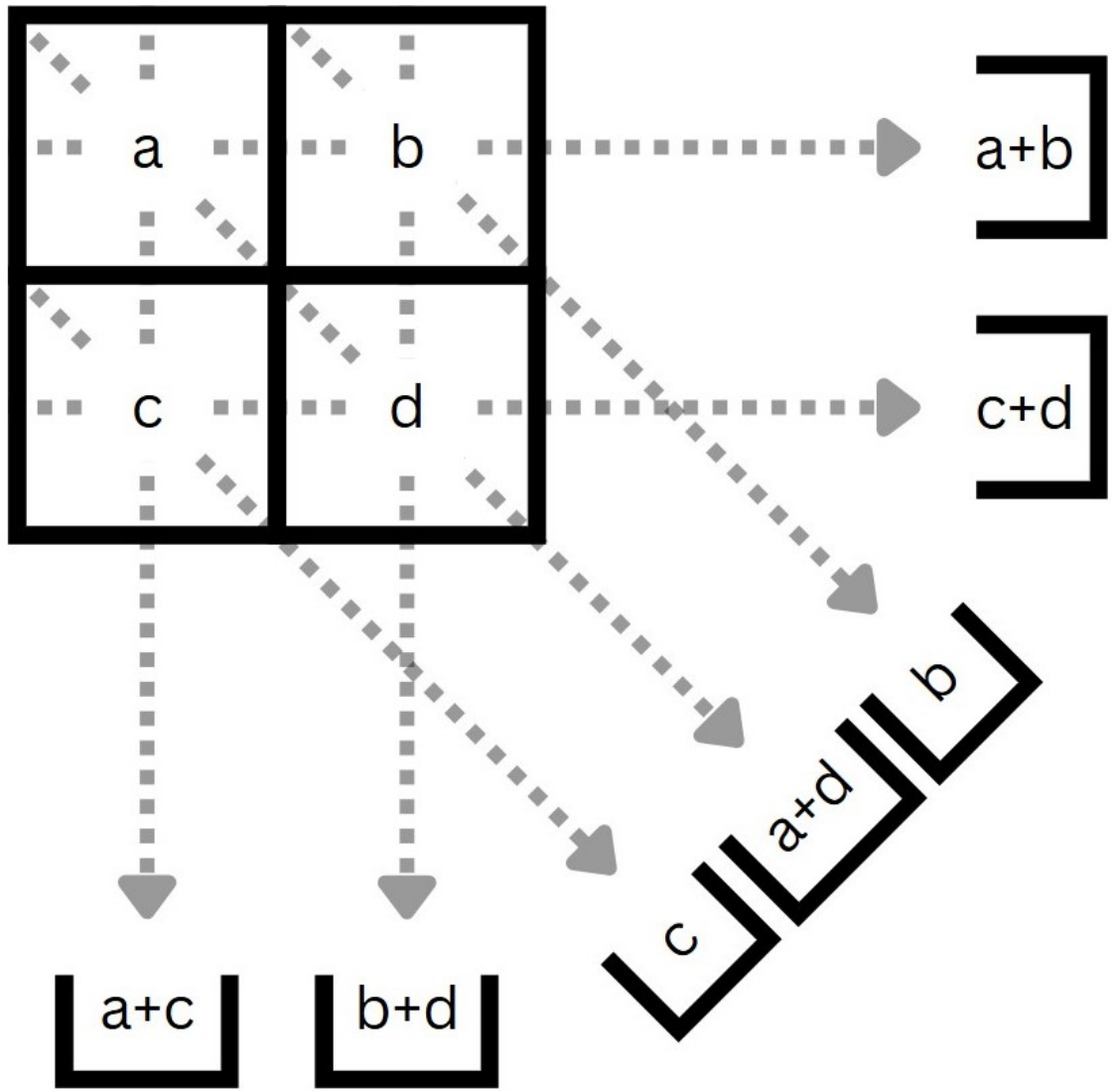
Exemple de rétroprojection

- $2^{ème}$ projection



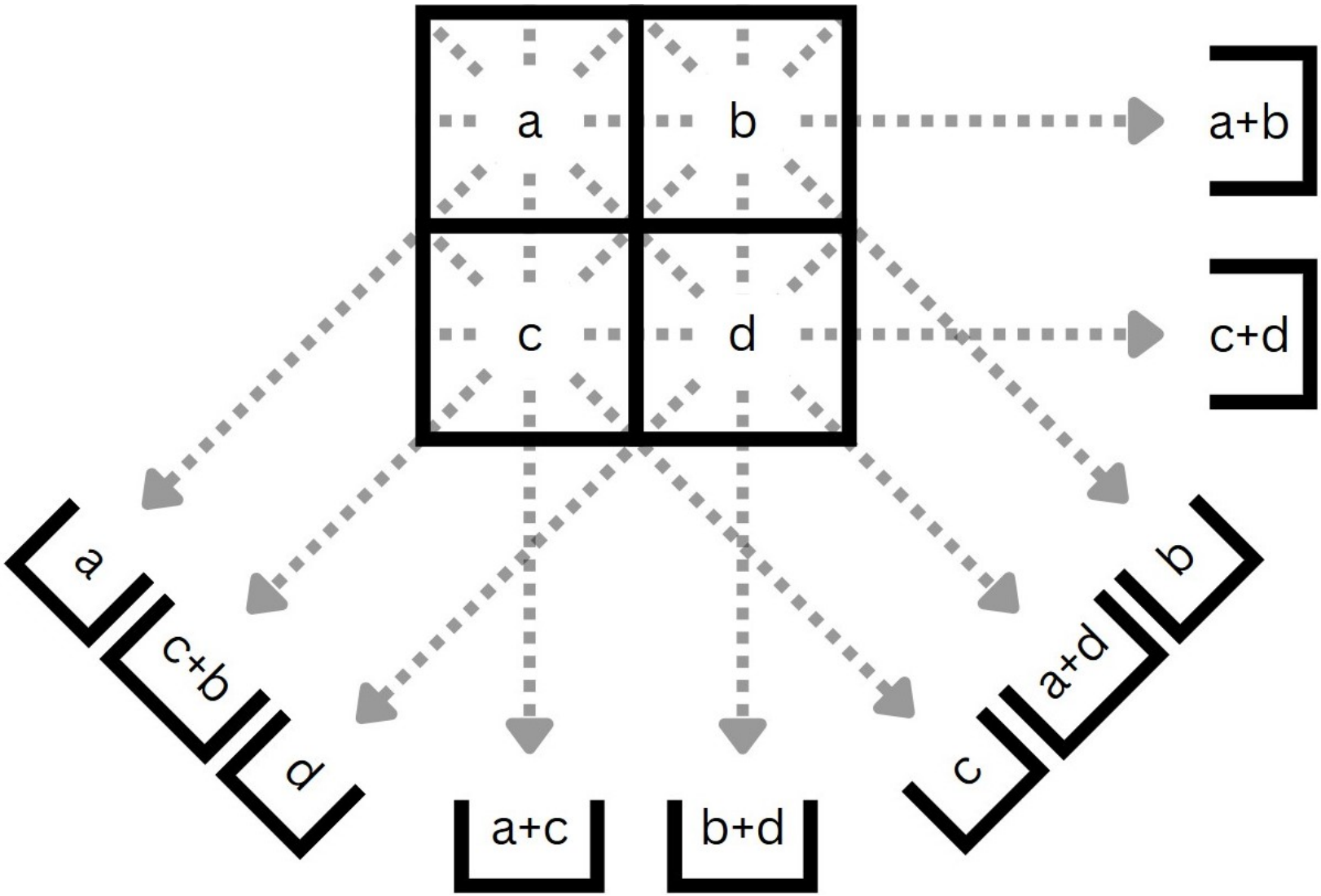
Exemple de rétroprojection

- $3^{ème}$ projection



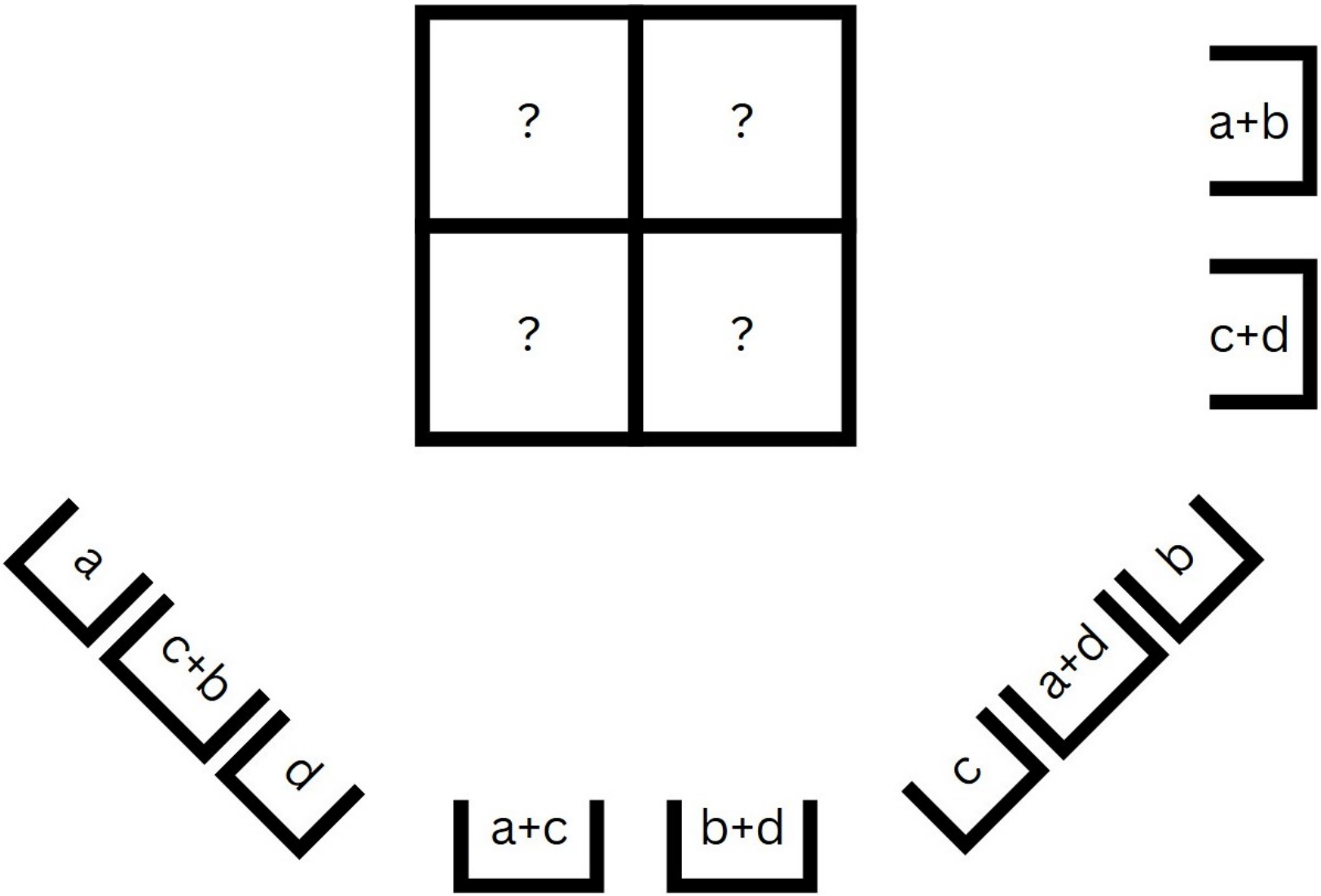
Exemple de rétroprojection

- $4^{ème}$ projection



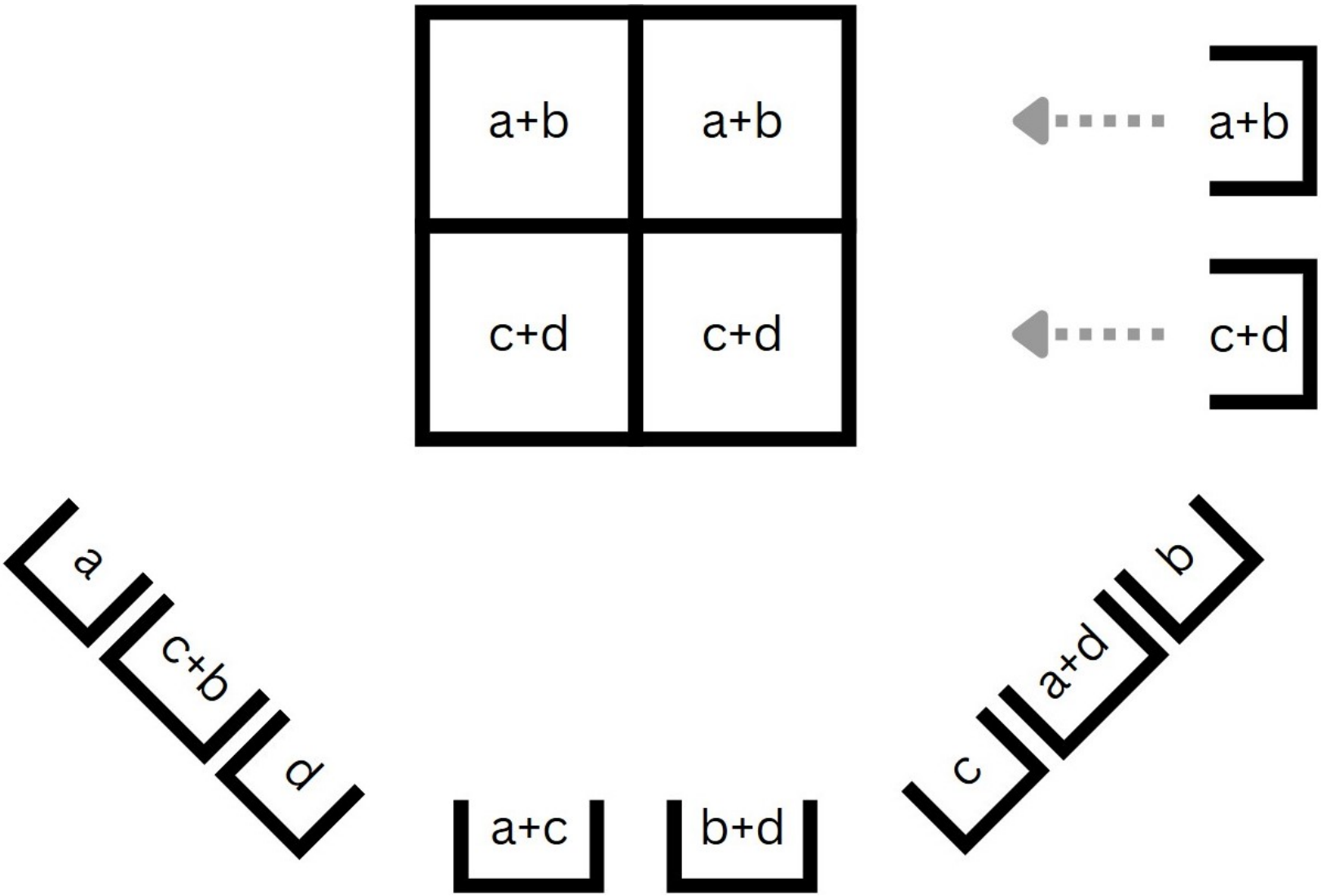
Exemple de rétroprojection

- Étape de rétroprojection



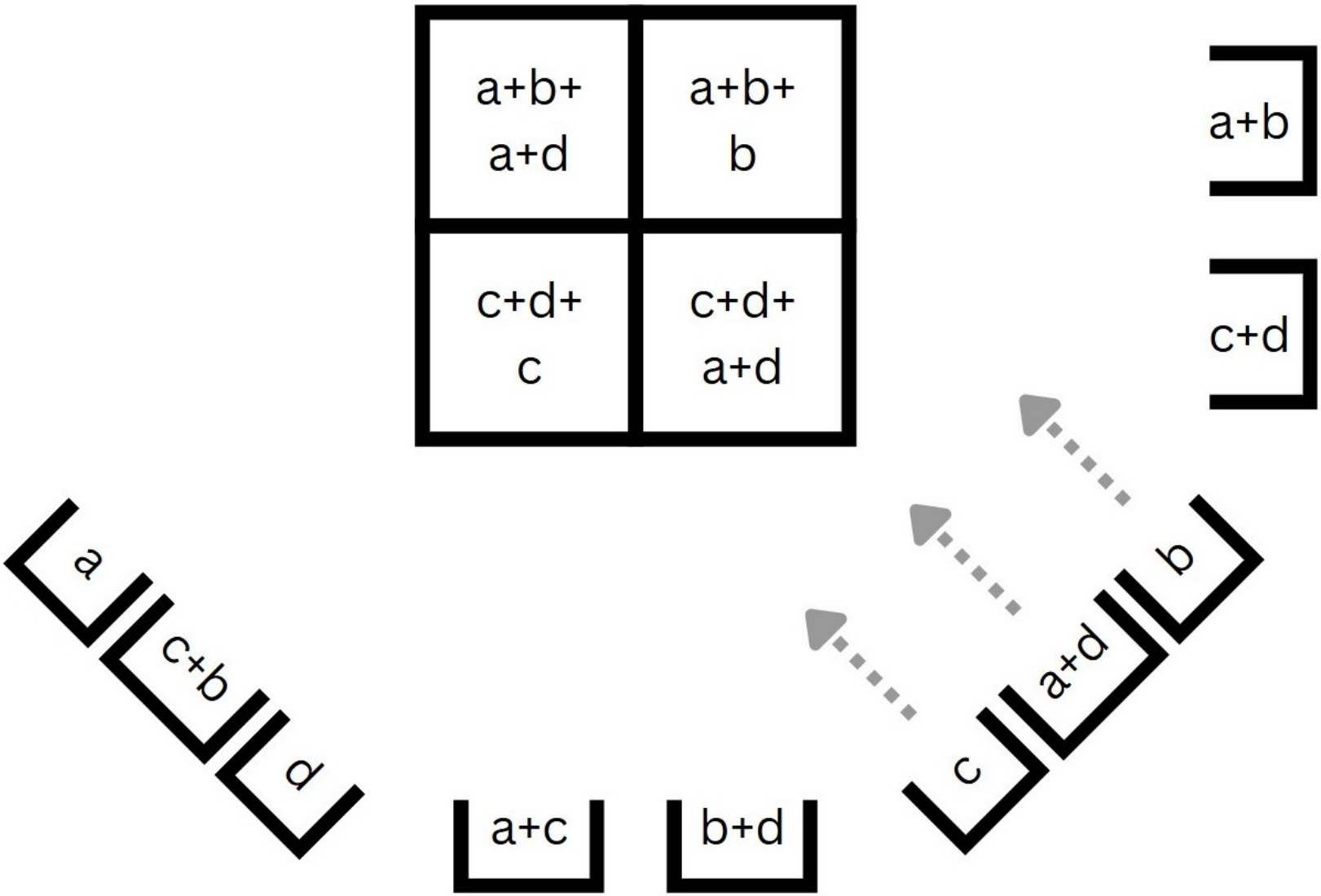
Exemple de rétroprojection

- $1^{ère}$ rétroprojection



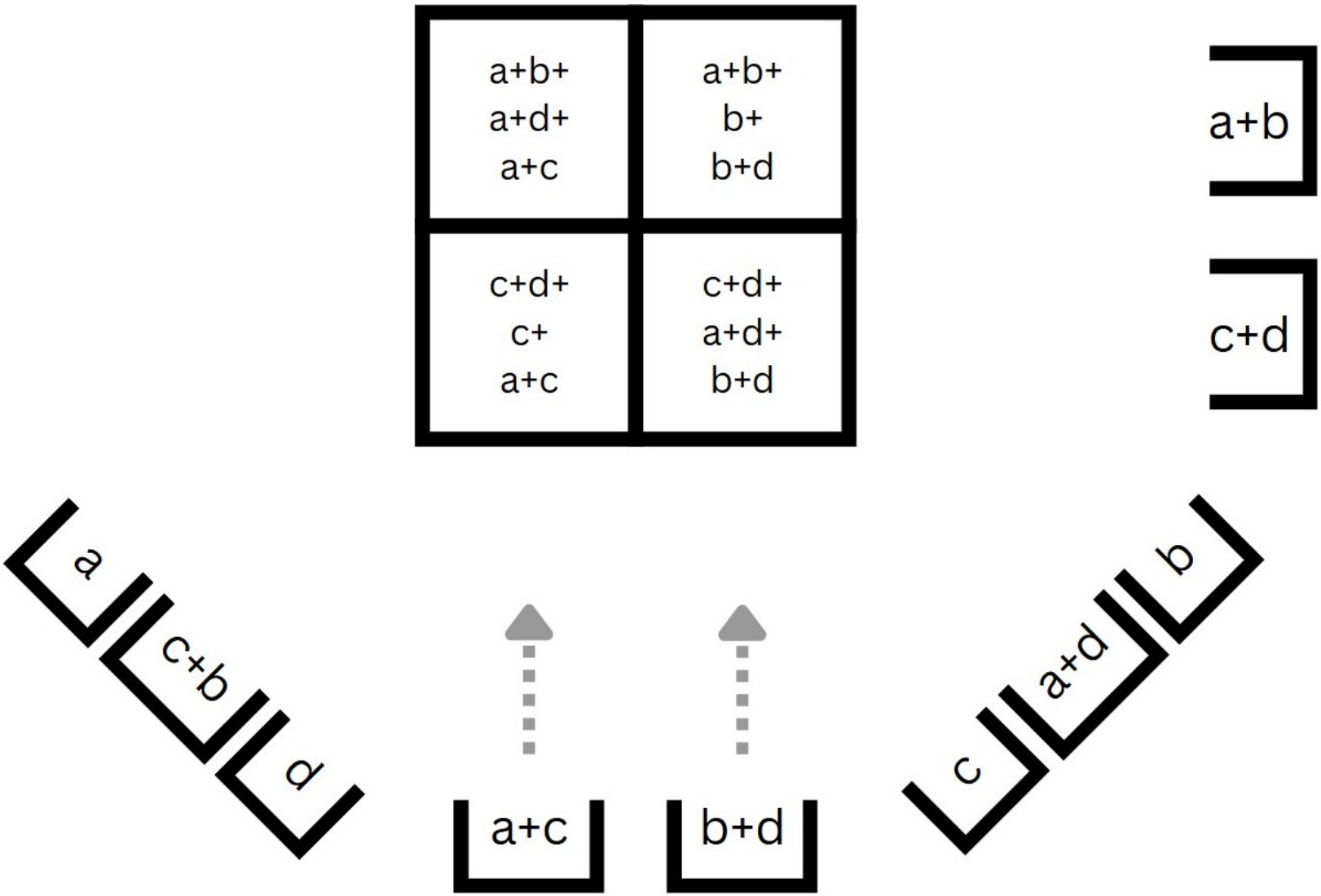
Exemple de rétroprojection

- $2^{ème}$ rétroprojection



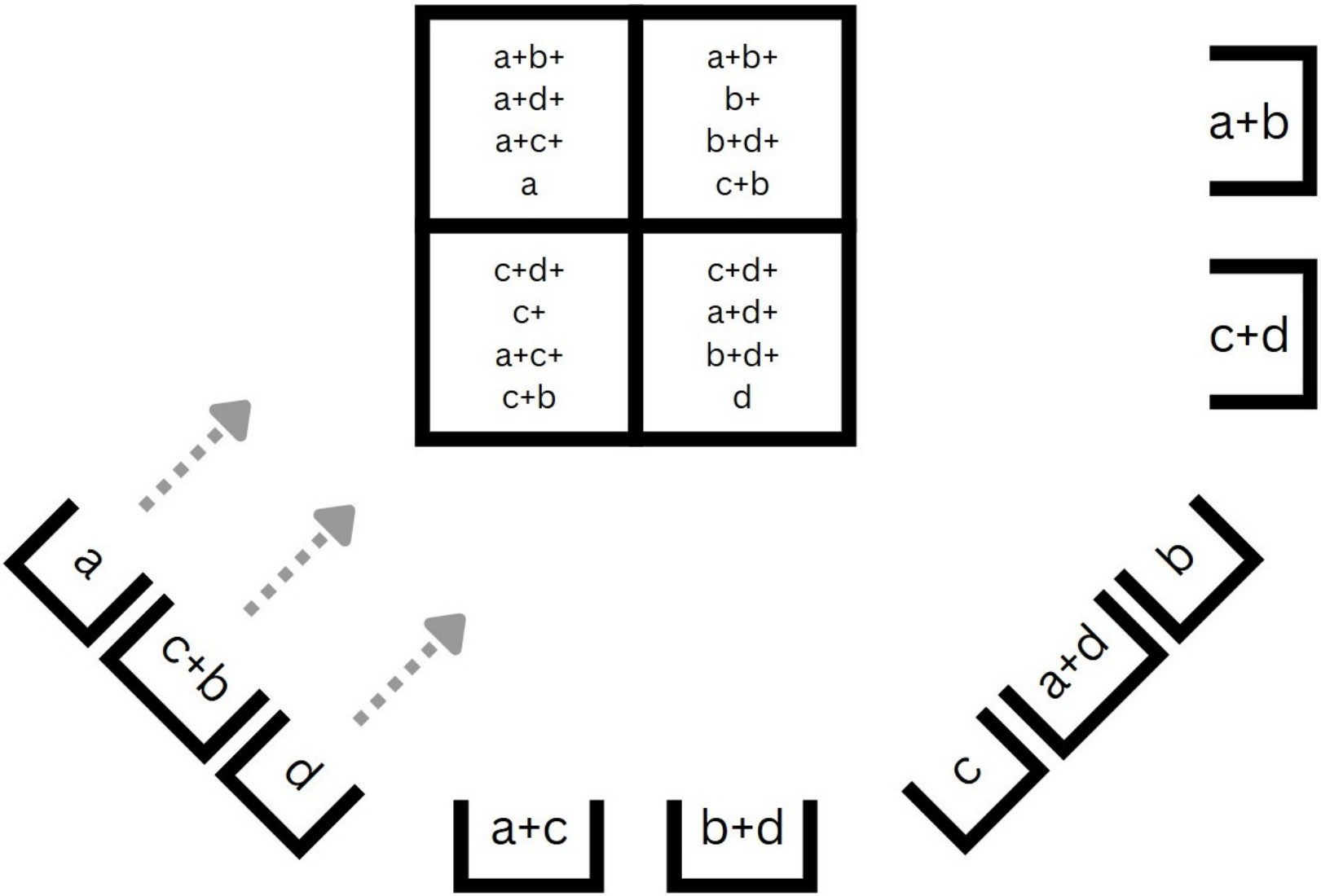
Exemple de rétroprojection

- $3^{ème}$ rétroprojection



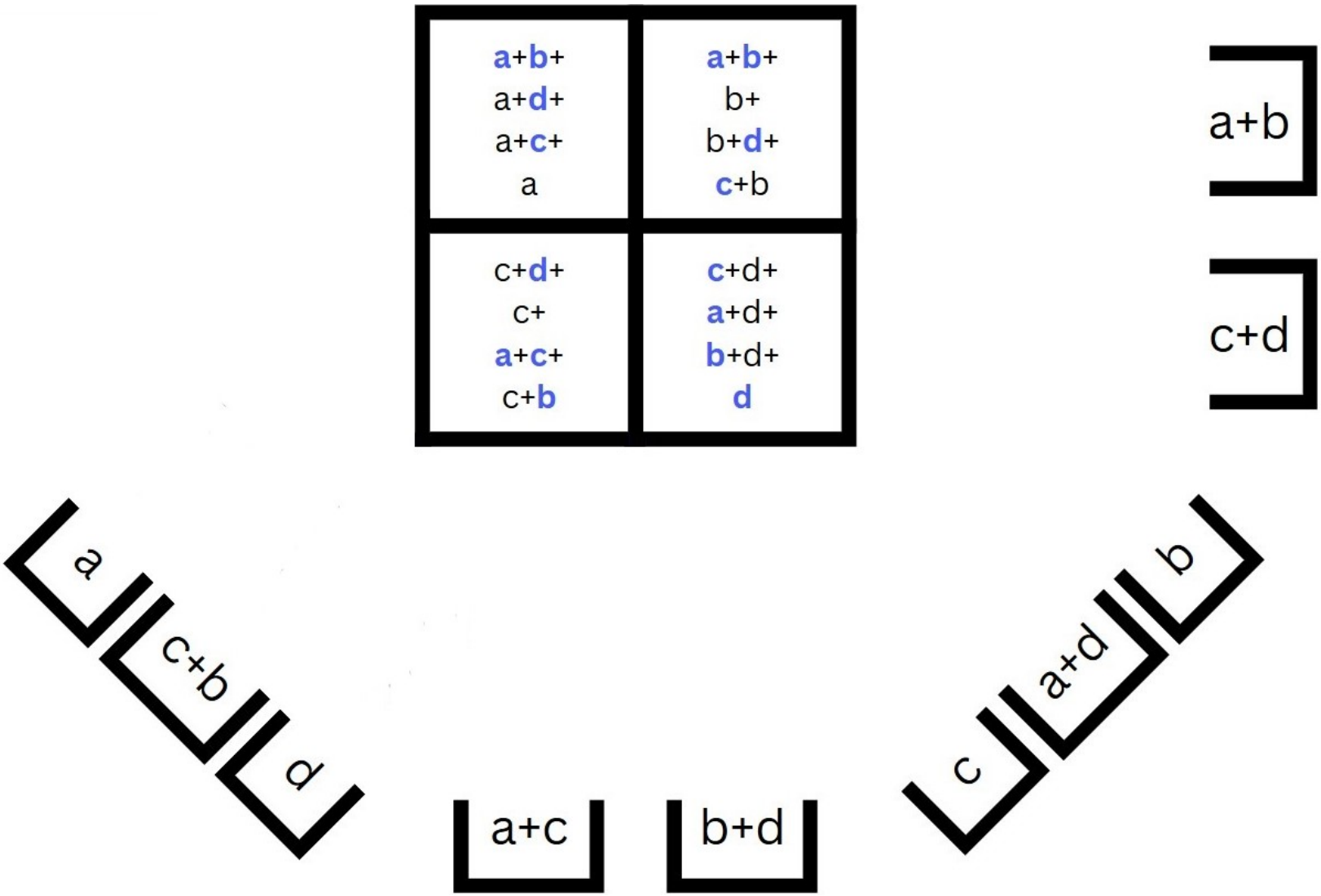
Exemple de rétroprojection

- $4^{ème}$ rétroprojection



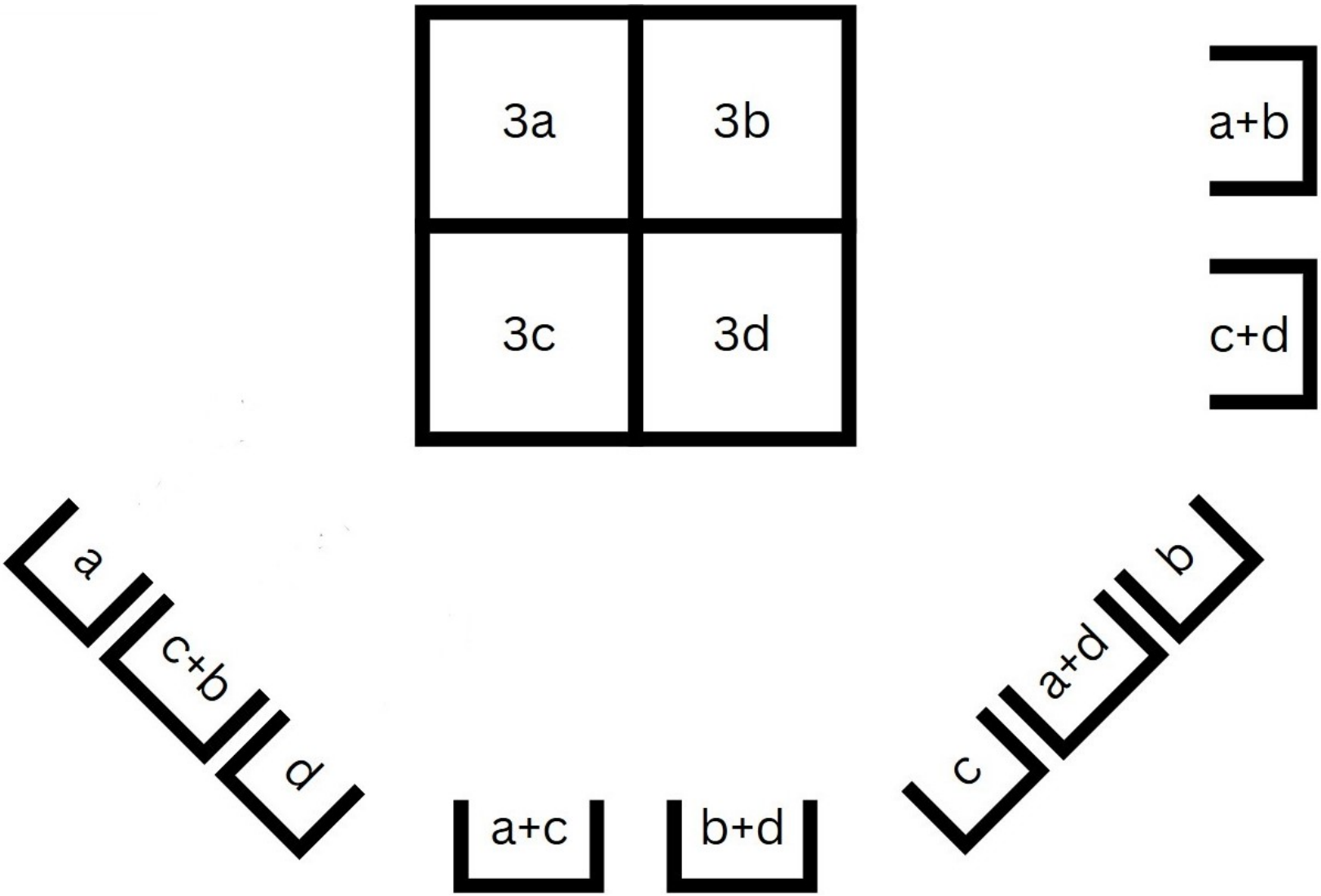
Exemple de rétroprojection

- Soustraire la somme de projection de chaque entrée



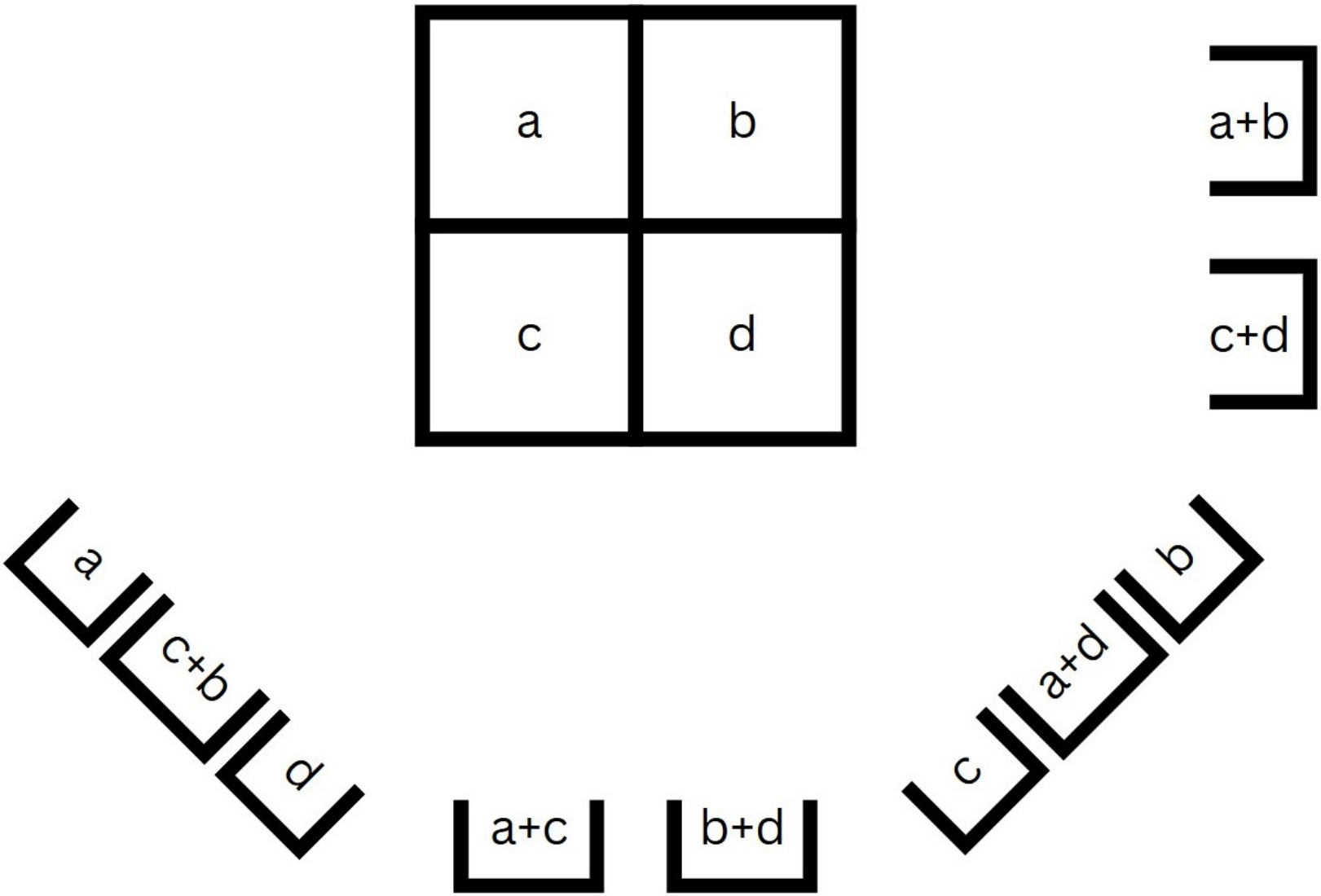
Exemple de rétroprojection

- Soustraire la somme de projection de chaque entrée



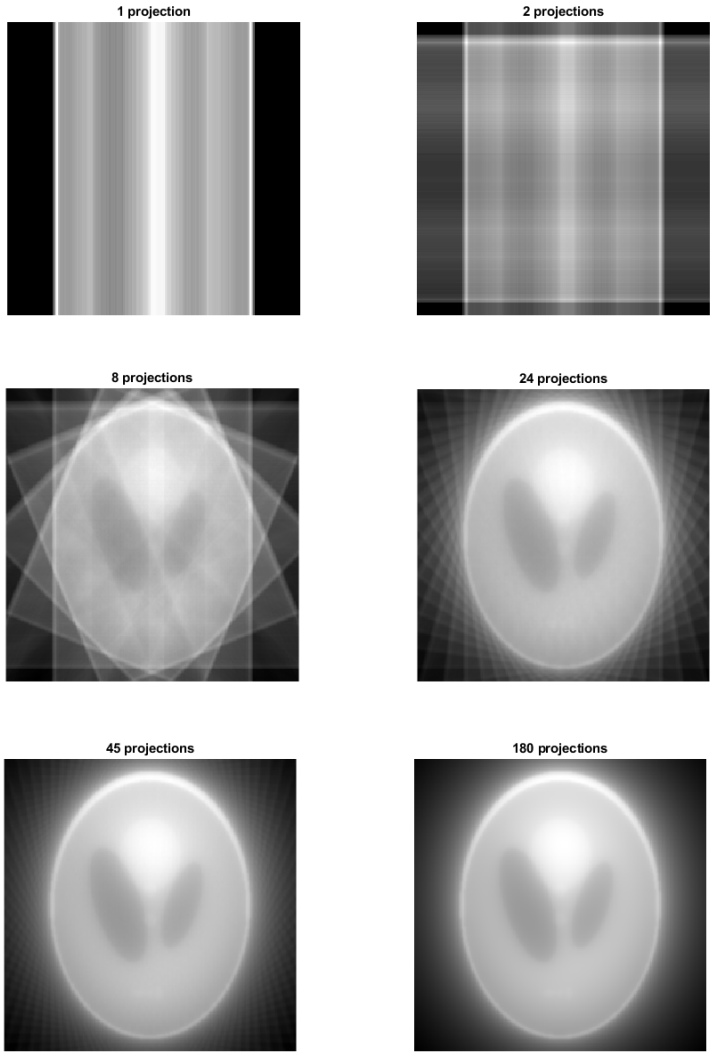
Exemple de rétroprojection

- Diviser par le nombre de projections -1



Rétroprojection : Flou de l'image

(a) Fantôme Original Shepp-Logan



(b) Image rétro-projetée pour les projections (1, 2, 8, 24, 45, 180)

Rétroprojection

La transformée de Radon d'une distribution $f(x, y)$ est donnée par

$$p(s, \phi) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \cdot \delta(x \cos \phi + y \sin \phi - s)\, dx\, dy$$

Rétroprojection

La transformée de Radon d'une distribution $f(x, y)$ est donnée par

$$p(s, \phi) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \cdot \delta(x \cos \phi + y \sin \phi - s)\, dx\, dy$$

Image rétro-projetée :

$$b(x, y) = \int_{0}^{\pi} p(s, \phi)\big|_{s = x \cos \phi + y \sin \phi}\, d\phi$$

- Intégrer sur 180°, l'autre moitié ne donne pas d'information supplémentaire

Rétroprojection

La transformée de Radon d'une distribution $f(x, y)$ est donnée par

$$p(s, \phi) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \cdot \delta(x \cos \phi + y \sin \phi - s)\, dx\, dy$$

Image rétro-projetée :

$$b(x, y) = \int_{0}^{\pi} p(s, \phi)\big|_{s=x \cos \phi + y \sin \phi}\, d\phi$$

- Intégrer sur 180°, l'autre moitié ne donne pas d'information supplémentaire

L'image reconstruite est floue :

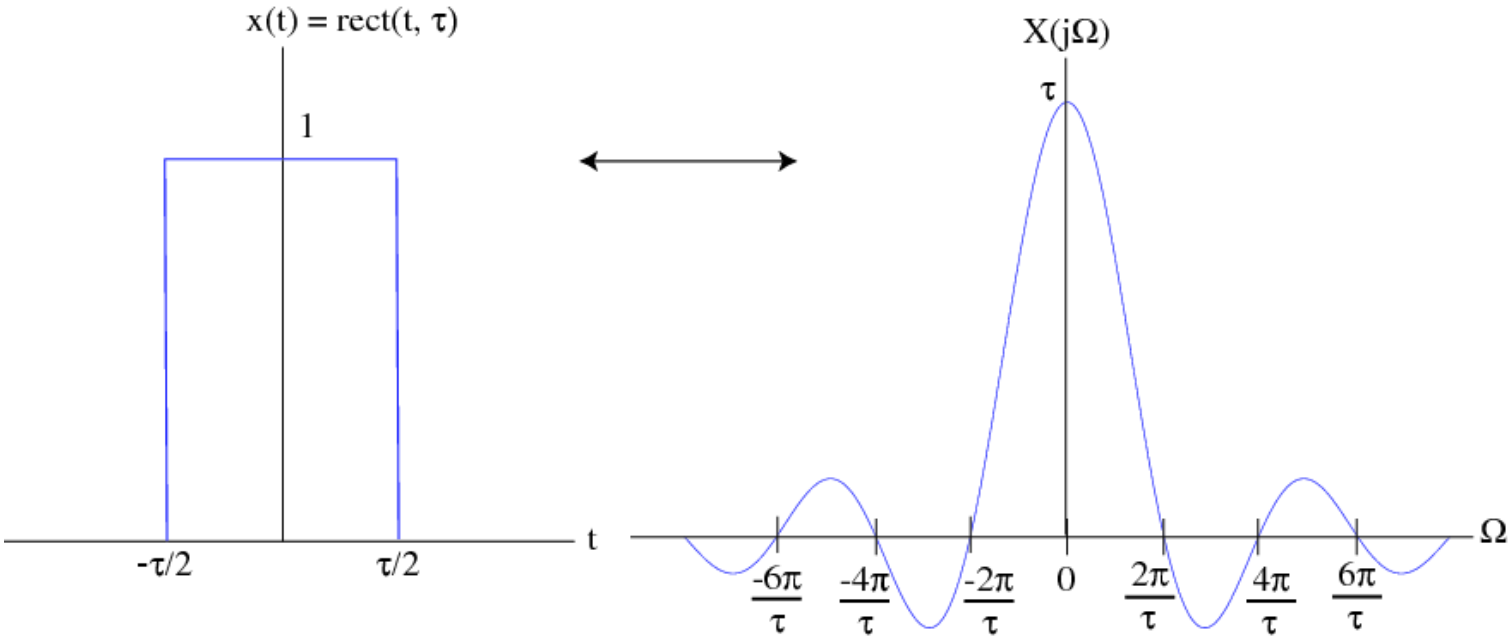
$$b(x, y) = f(x, y) * \frac{1}{\sqrt{x^2 + y^2}}$$

Transformée de Fourier

Définition

Si f une fonction de $L^1(\mathbb{R})$. On appelle **transformée de Fourier** de f, qu'on note $\mathcal{F}(f)$, la fonction définie sur $\mathbb{R}$ par :

$$\mathcal{F}\{f(s)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(s) e^{-i\omega s} ds$$



Transformée de Fourier

Théorème de la Convolution

La transformée de Fourier d'une convolution est le produit ponctuel des transformées de Fourier :

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\} \cdot \mathcal{F}\{h\}$$

Transformée de Fourier

Théorème de la Convolution

La transformée de Fourier d'une convolution est le produit ponctuel des transformées de Fourier :

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\} \cdot \mathcal{F}\{h\}$$

Application à la Rétroprojection

En appliquant ce théorème à notre image floue $b(x, y)$:

B(u,v) = F{b(x,y)}

Transformée de Fourier

Théorème de la Convolution

La transformée de Fourier d'une convolution est le produit ponctuel des transformées de Fourier :

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\} \cdot \mathcal{F}\{h\}$$

Application à la Rétroprojection

En appliquant ce théorème à notre image floue $b(x, y)$:

$$\mathrm{B(u,v)} = \mathrm{F\{b(x,y)\}} = \mathrm{F}\left\{ f(x,y) * \frac{1}{r} \right\}$$

Transformée de Fourier

Théorème de la Convolution

La transformée de Fourier d'une convolution est le produit ponctuel des transformées de Fourier :

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\} \cdot \mathcal{F}\{h\}$$

Application à la Rétroprojection

En appliquant ce théorème à notre image floue $b(x, y)$:

$$\mathrm{B(u,v)} = \mathrm{F\{b(x,y)\}} = \mathrm{F}\left\{ f(x,y) * \frac{1}{r} \right\} = F(u,v) \cdot \mathcal{F}\left\{ \frac{1}{r} \right\}$$

Transformée de Fourier

Théorème de la Convolution

La transformée de Fourier d'une convolution est le produit ponctuel des transformées de Fourier :

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\} \cdot \mathcal{F}\{h\}$$

Application à la Rétroprojection

En appliquant ce théorème à notre image floue $b(x, y)$:

$$\mathrm{B(u,v)} = \mathrm{F\{b(x,y)\}} = \mathrm{F}\left\{ f(x,y) * \frac{1}{r} \right\} = F(u,v) \cdot \mathcal{F}\left\{ \frac{1}{r} \right\} = F(u,v) \cdot \frac{1}{\rho}$$

où $\rho = \sqrt{u^2 + v^2}$ est la fréquence radiale.

Transformée de Fourier

Théorème de la Convolution

La transformée de Fourier d'une convolution est le produit ponctuel des transformées de Fourier :

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\} \cdot \mathcal{F}\{h\}$$

Application à la Rétroprojection

En appliquant ce théorème à notre image floue $b(x, y)$:

$$\mathrm{B(u,v)} = \mathrm{F\{b(x,y)\}} = \mathrm{F}\left\{f(x,y) * \frac{1}{r}\right\} = F(u,v) \cdot \mathcal{F}\left\{\frac{1}{r}\right\} = F(u,v) \cdot \frac{1}{\rho}$$

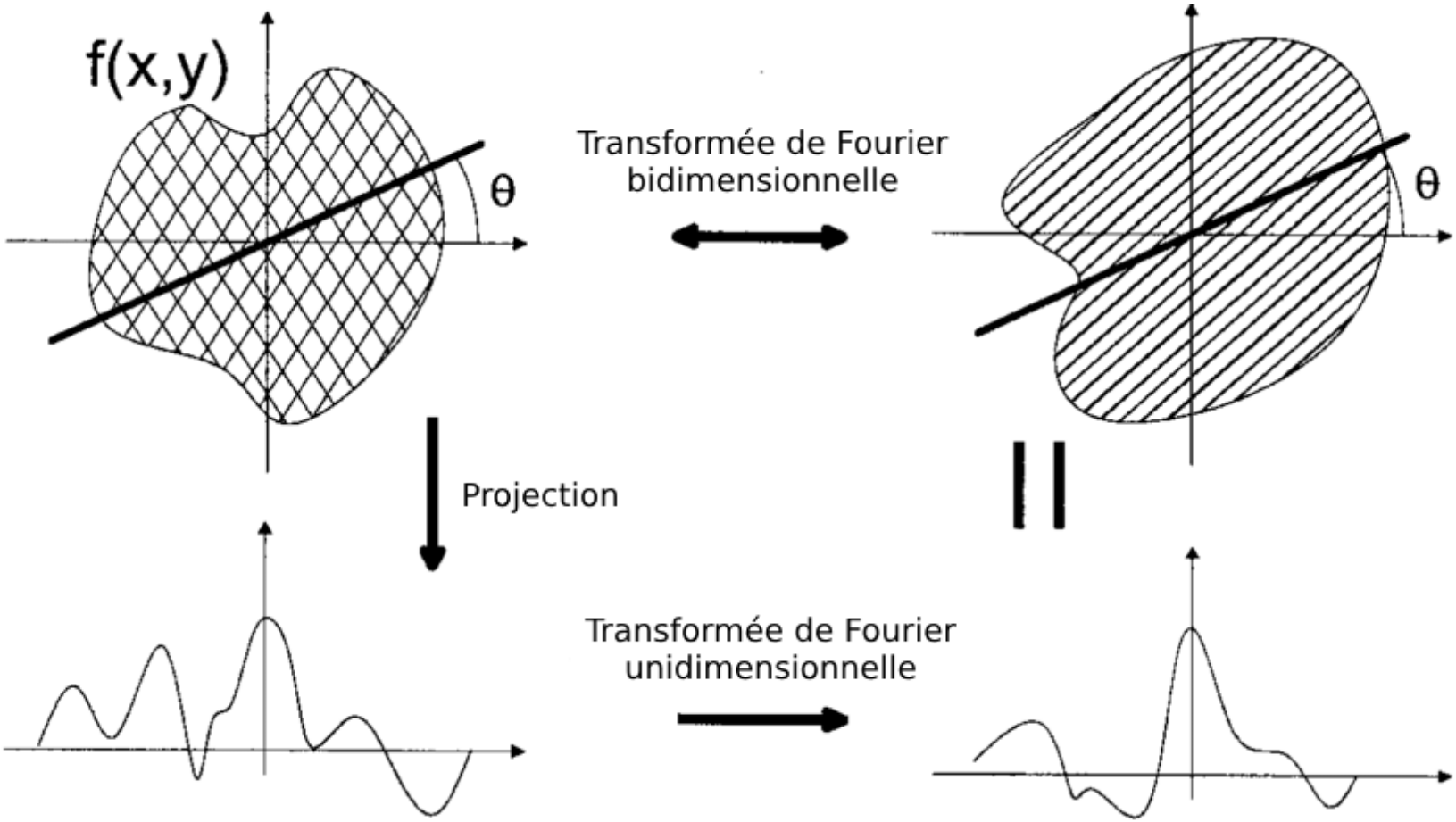
où $\rho = \sqrt{u^2 + v^2}$ est la fréquence radiale.

- Le flou est donc une suppression des hautes fréquences.

Le Théorème de la Tranche Centrale

Théorème

$$\mathcal{F}_2\{f(x,y)\}|_{\phi=\phi'} = \mathcal{F}_1\{p(s,\phi')\}$$



L'Algorithme de Rétroprojection Filtrée

$$f(x,y) = \int_0^{\pi} d\phi \Big[\int_{-\infty}^{\infty} d\omega |\omega| P(\omega) e^{2\pi i \omega s} \Big]_{s = x\cos\phi + y\sin\phi}$$

L'Algorithme de Rétroprojection Filtrée

$$f(x,y) = \int_0^{\pi} d\phi \Big[\int_{-\infty}^{\infty} d\omega |\omega| P(\omega) e^{2\pi i \omega s} \Big]_{s = x\cos\phi + y\sin\phi}$$

- Transformée de Fourier de la projection $p(s,\phi)$

L'Algorithme de Rétroprojection Filtrée

$$f(x,y) = \int_0^{\pi} d\phi \Big[\int_{-\infty}^{\infty} d\omega |\omega| P(\omega) e^{2\pi i \omega s} \Big]_{s = x\cos\phi + y\sin\phi}$$

- Transformée de Fourier de la projection $p(s,\phi)$
- Multiplier par le filtre de fréquence $|\omega|$

L'Algorithme de Rétroprojection Filtrée

$$f(x,y) = \int_0^{\pi} d\phi \Big[\int_{-\infty}^{\infty} d\omega |\omega| P(\omega) e^{2\pi i \omega s} \Big]_{s = x\cos\phi + y\sin\phi}$$

- Transformée de Fourier de la projection $p(s,\phi)$
- Multiplier par le filtre de fréquence $|\omega|$
- Transformée de Fourier inverse de ce produit

L'Algorithme de Rétroprojection Filtrée

$$f(x,y) = \int_0^{\pi} d\phi \left[\int_{-\infty}^{\infty} d\omega |\omega| P(\omega) e^{2\pi i \omega s} \right]_{s=x\cos\phi + y\sin\phi}$$

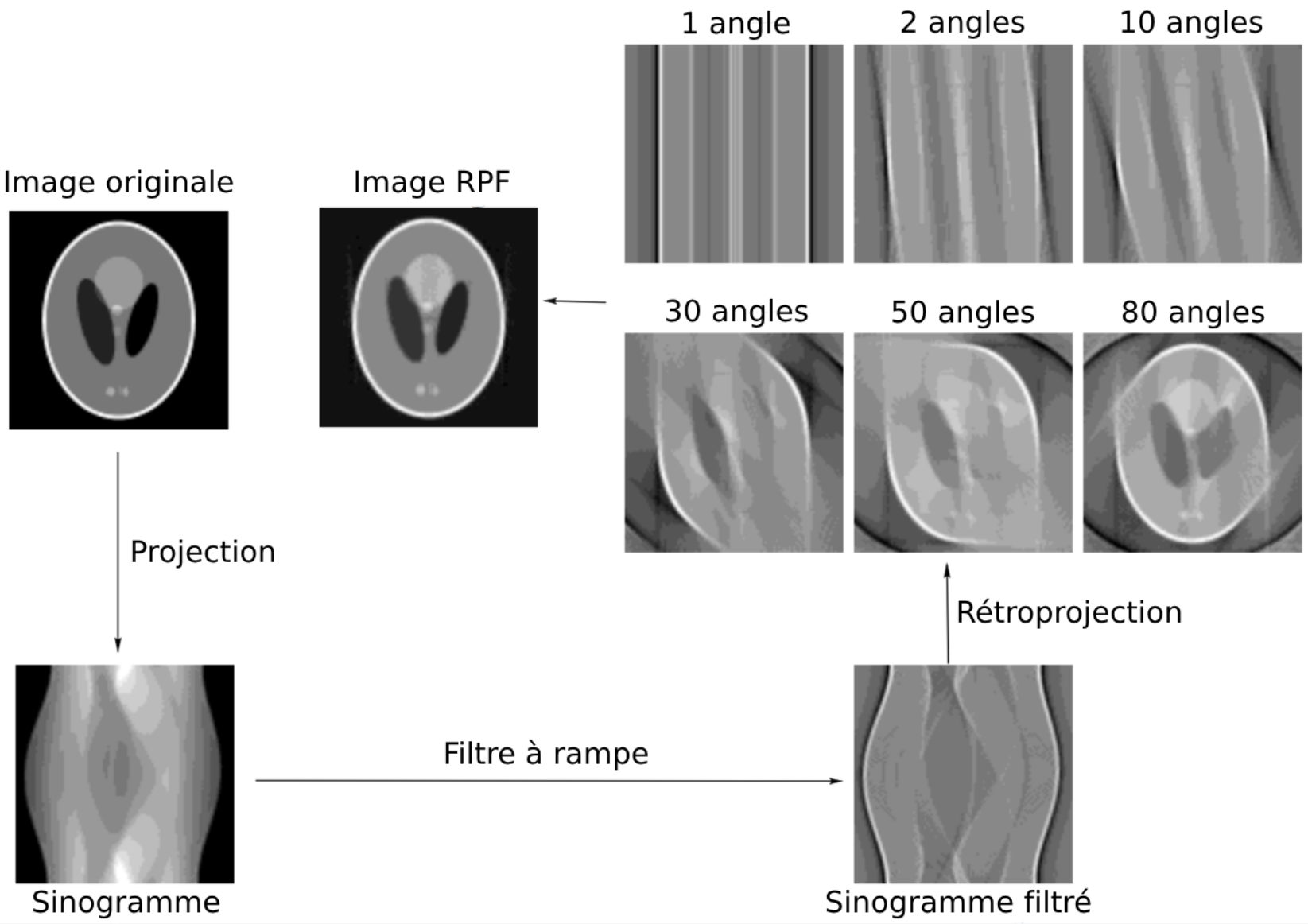
- Transformée de Fourier de la projection $p(s,\phi)$
- Multiplier par le filtre de fréquence $|\omega|$
- Transformée de Fourier inverse de ce produit
- Cette projection filtrée est rétroprojetée

L'Algorithme de Rétroprojection Filtrée

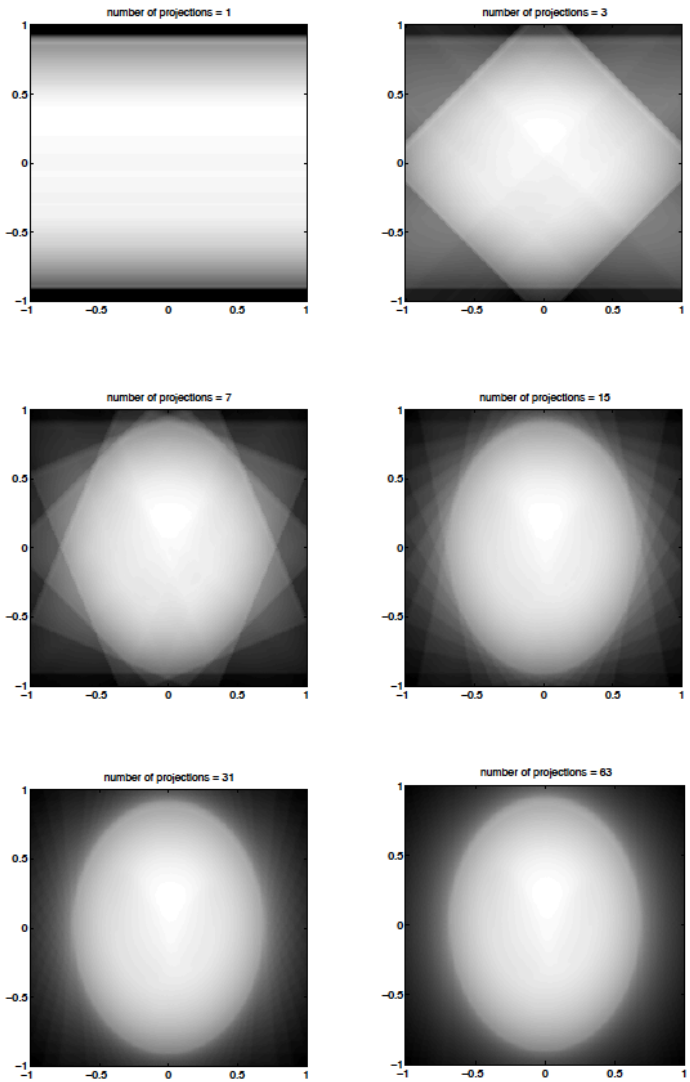
$$f(x,y) = \int_0^{\pi} d\phi \Big[\int_{-\infty}^{\infty} d\omega |\omega| P(\omega) e^{2\pi i \omega s} \Big]_{s = x\cos\phi + y\sin\phi}$$

- Transformée de Fourier de la projection $p(s,\phi)$
- Multiplier par le filtre de fréquence $|\omega|$
- Transformée de Fourier inverse de ce produit
- Cette projection filtrée est rétroprojetée
- Faire ensuite la somme de toutes les projections filtrées

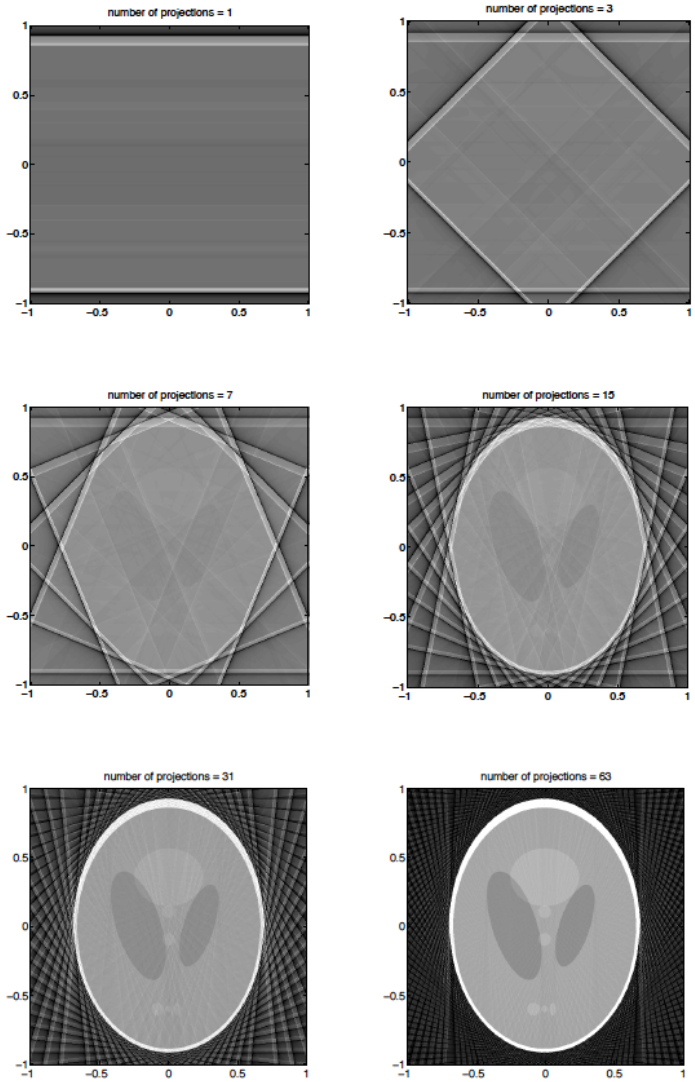
Rétroprojection Filtrée



RP vs. RPF



(a) Rétroprojection

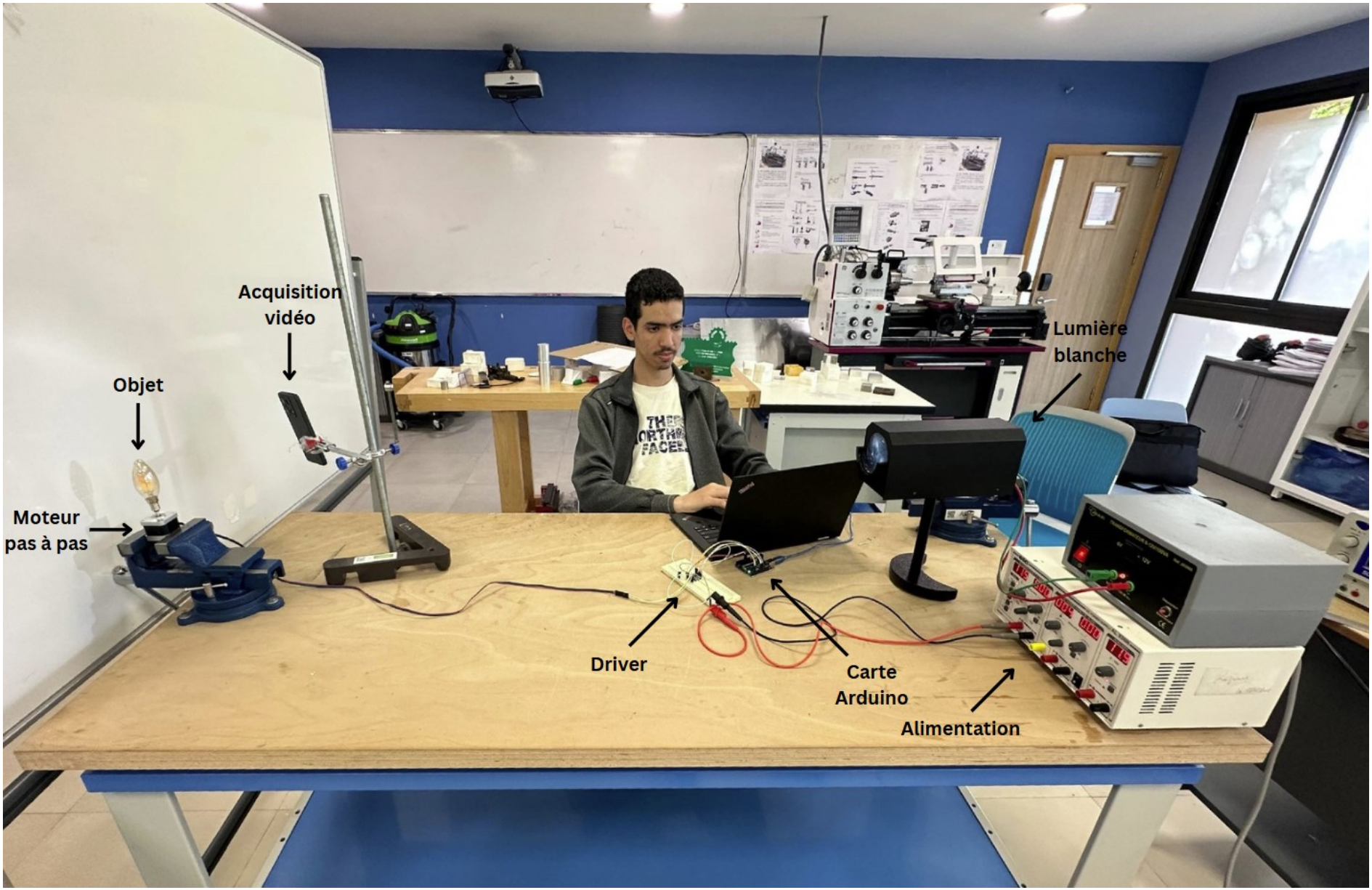


(b) Rétroprojection Filtrée

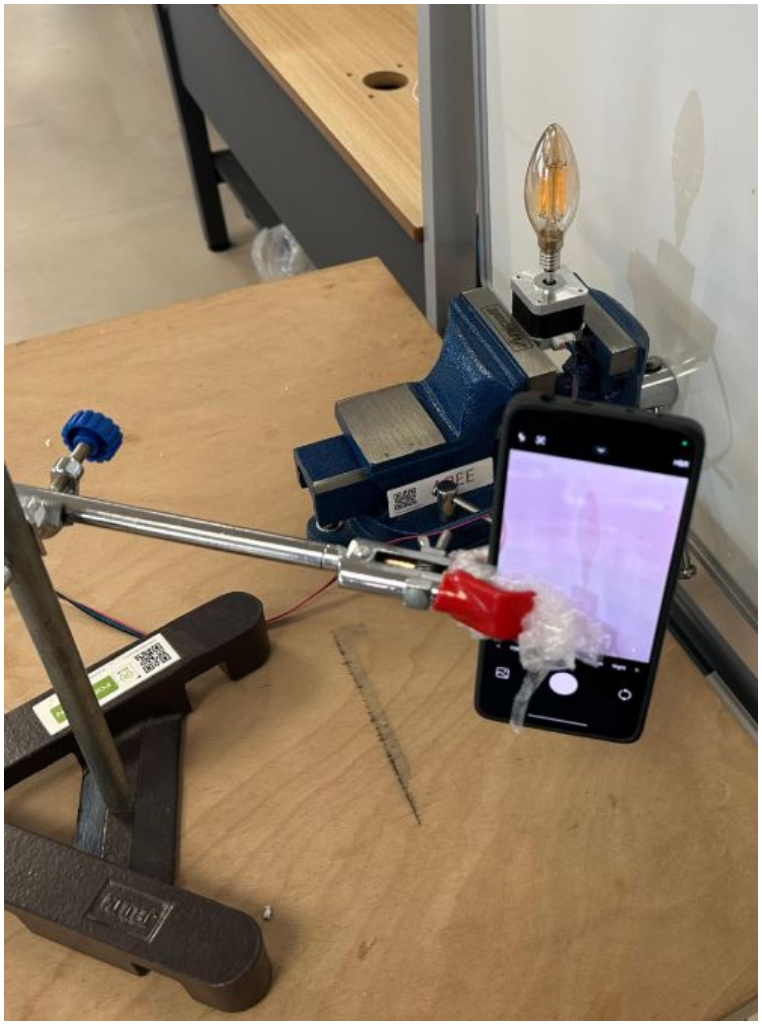
Sommaire

1. Introduction
2. Idée de base : Projections
3. Transformée de Radon
4. Reconstruction analytique d'images
   - Rétro-projection
   - Rétro-projection filtrée
5. **Expérience et Reconstruction avec Python et Tomviz**

Matériel et montage expérimental



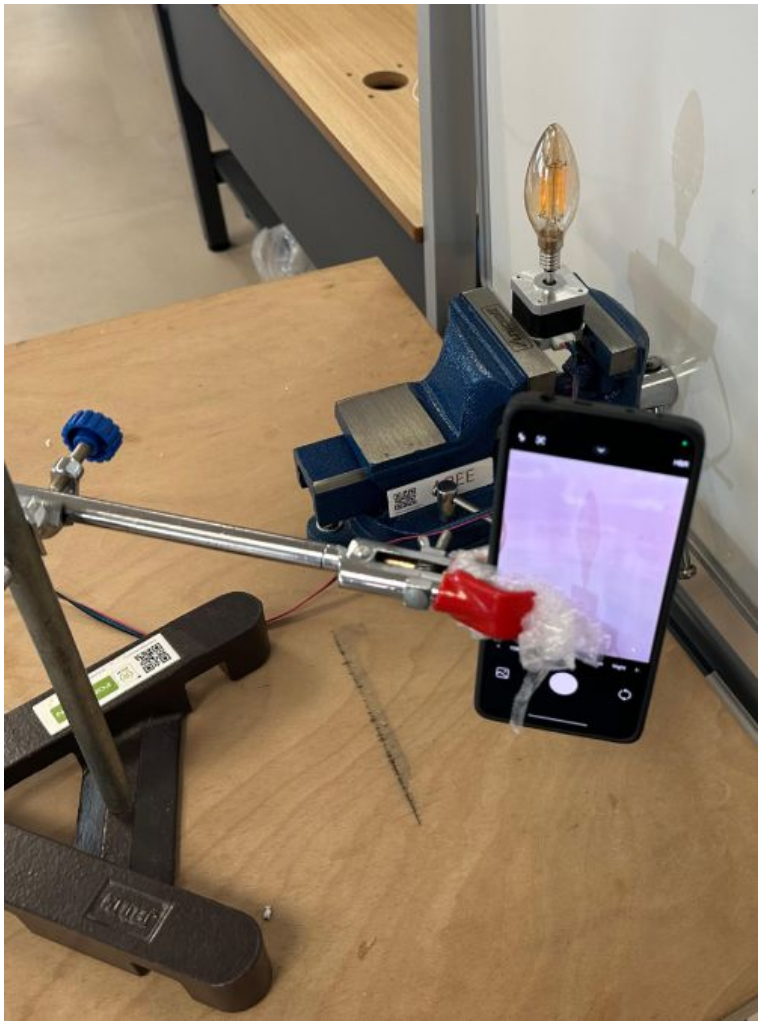
Matériel et montage expérimental



Pour l'acquisition de projections :

- Le moteur pas à pas tourne à 10 tr/min

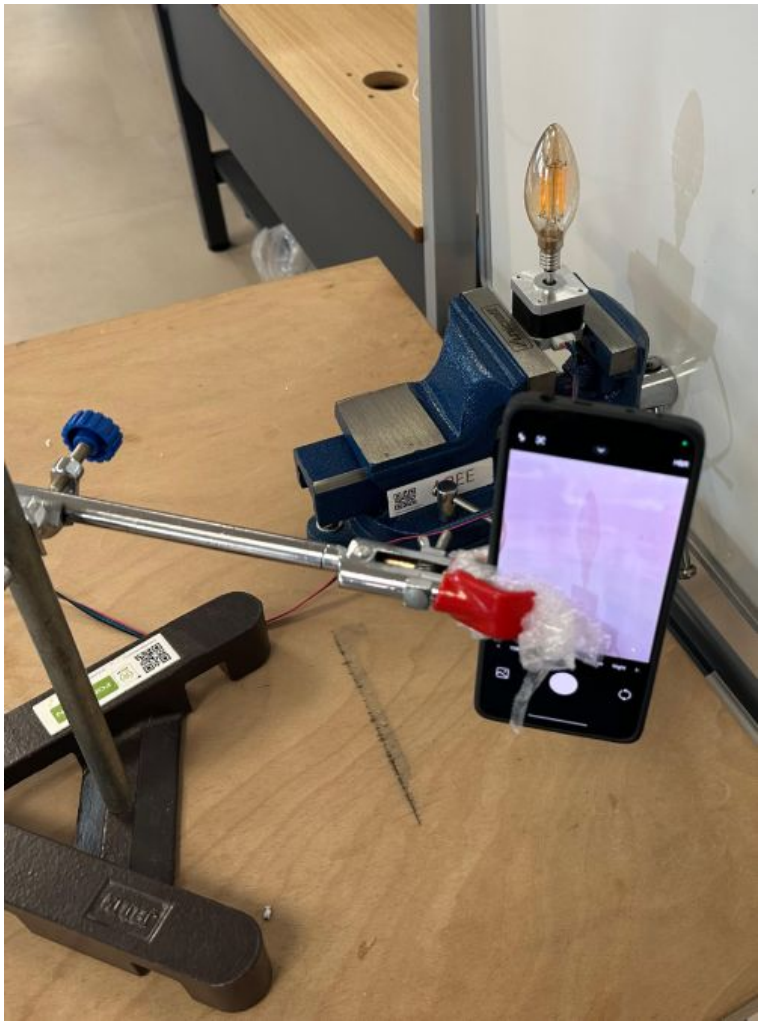
Matériel et montage expérimental



Pour l'acquisition de projections :

- Le moteur pas à pas tourne à 10 tr/min
- Le téléphone capture une vidéo à 120 fps

Matériel et montage expérimental

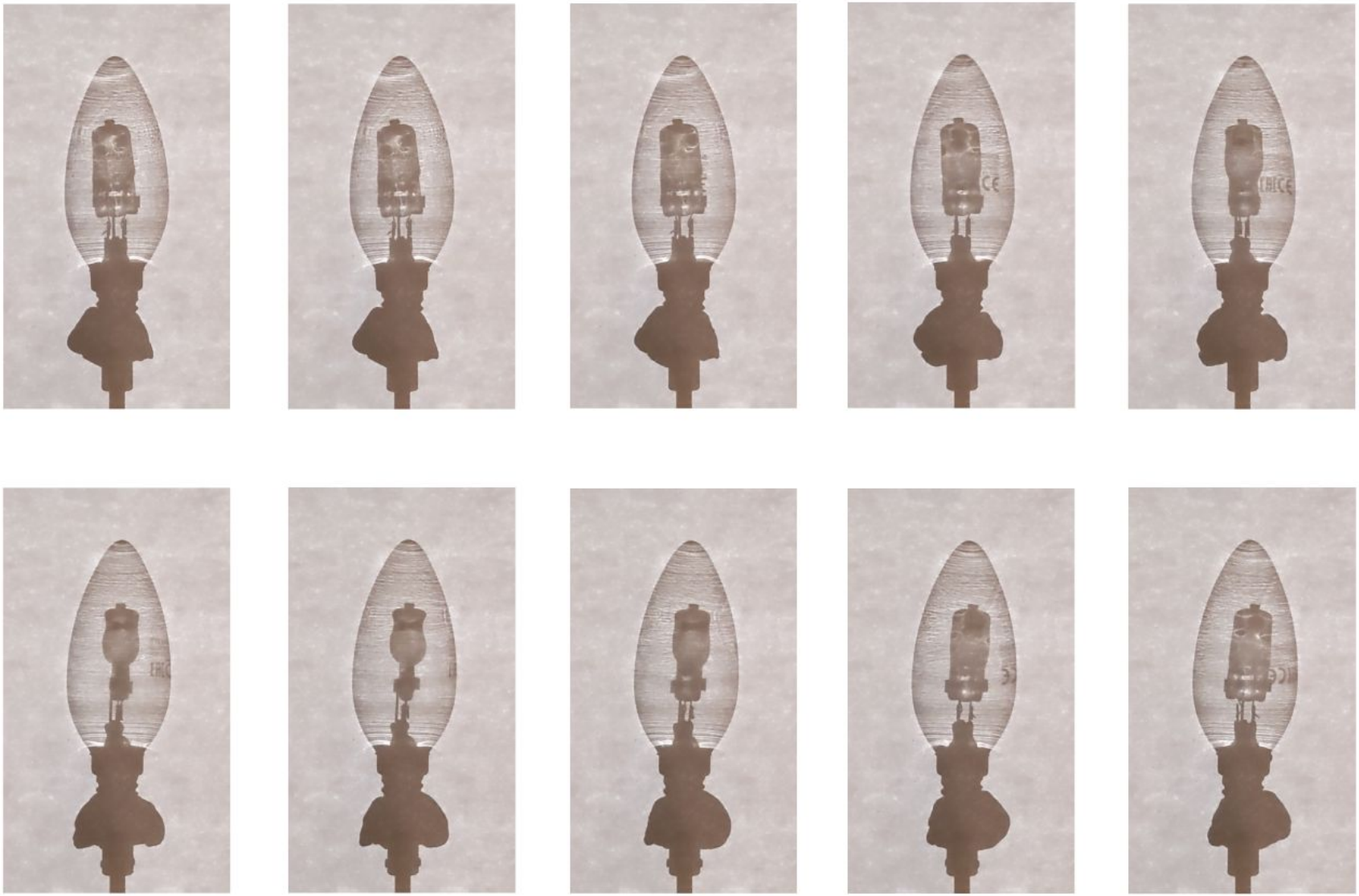


Pour l'acquisition de projections :

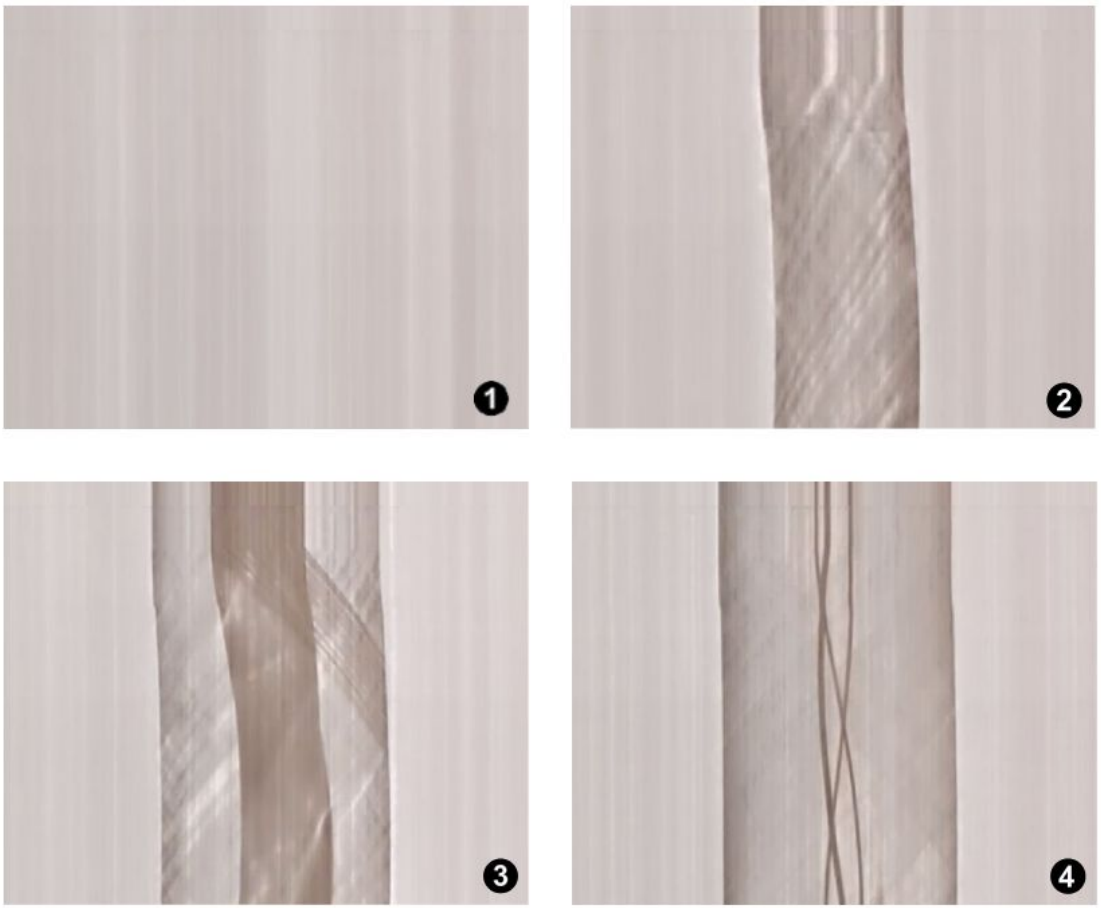
- Le moteur pas à pas tourne à 10 tr/min
- Le téléphone capture une vidéo à 120 fps
- Nous obtenons $\frac{120}{\frac{10}{60}} = 720$ images/tour

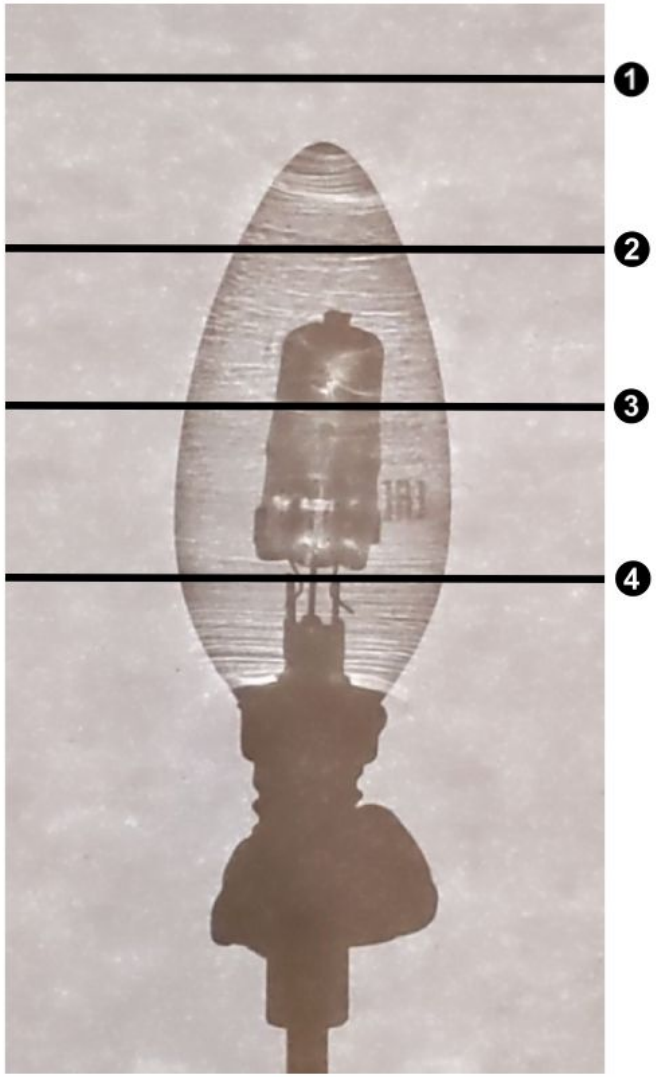
dans l'intervalle $[0, 180°]$ nous obtenons 360 images.

Acquisition de projections

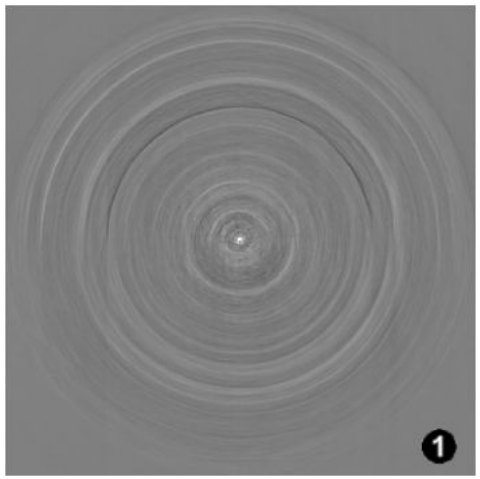


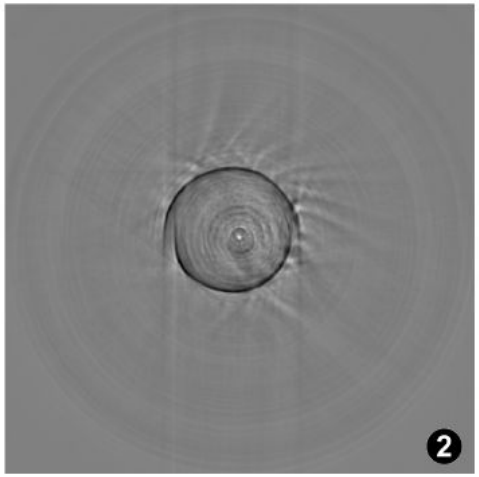
Génération de sinogrammes

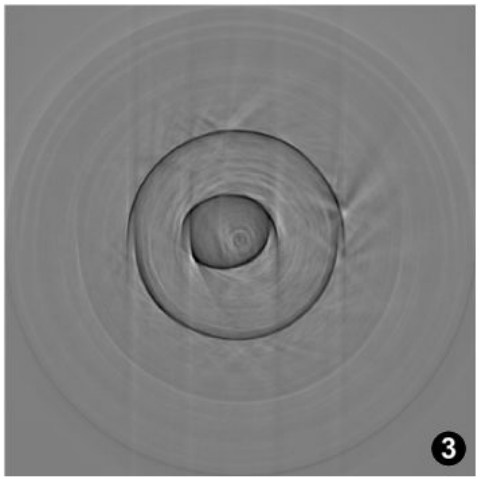


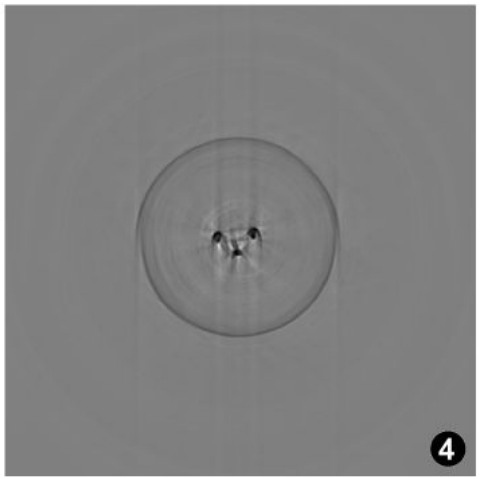


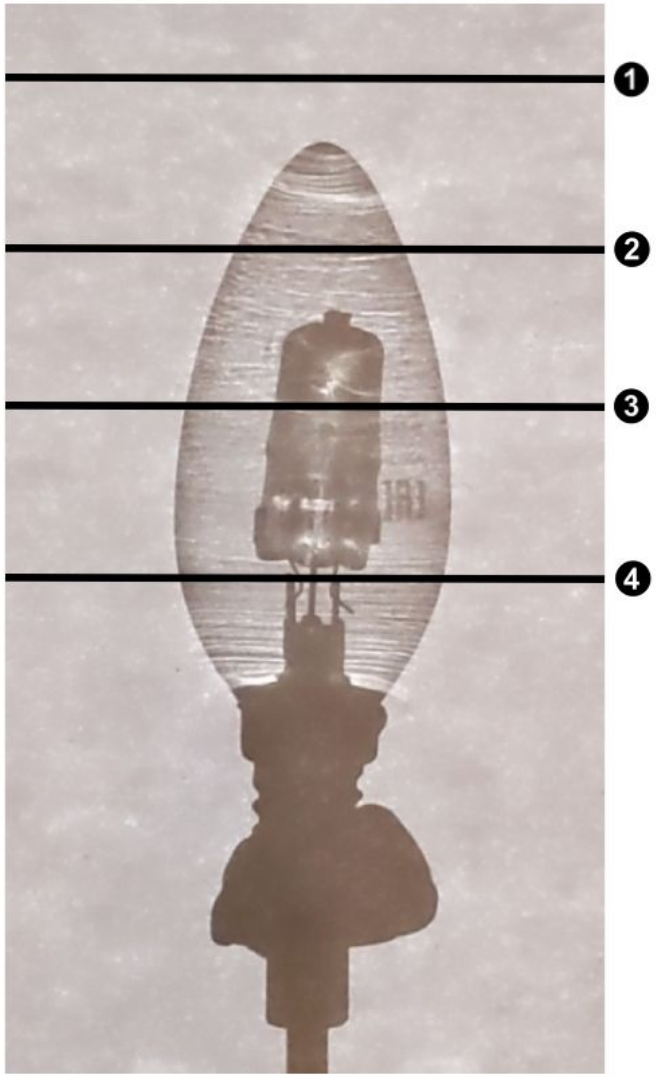
Rétroprojection filtrée des sinogrammes



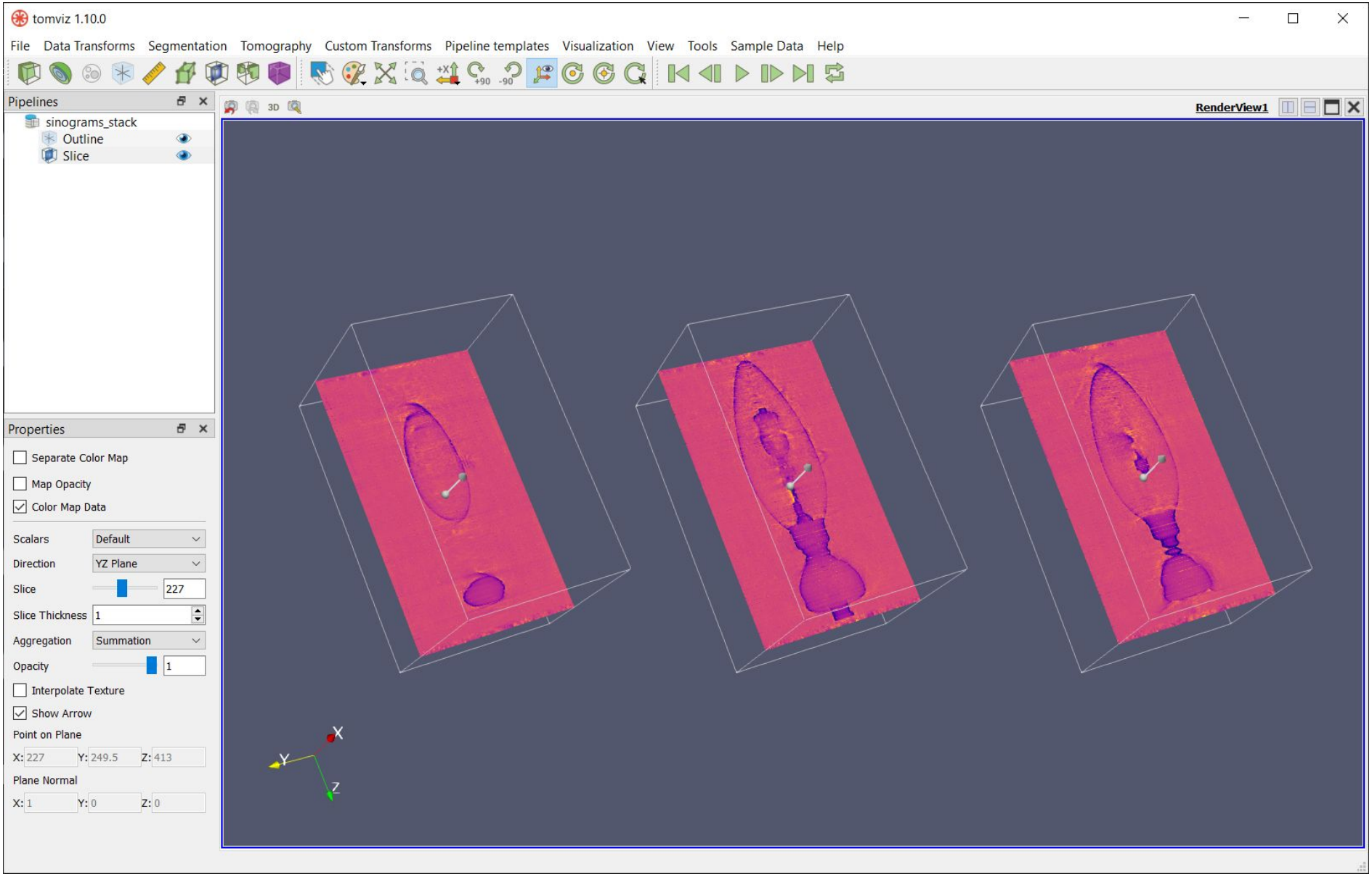




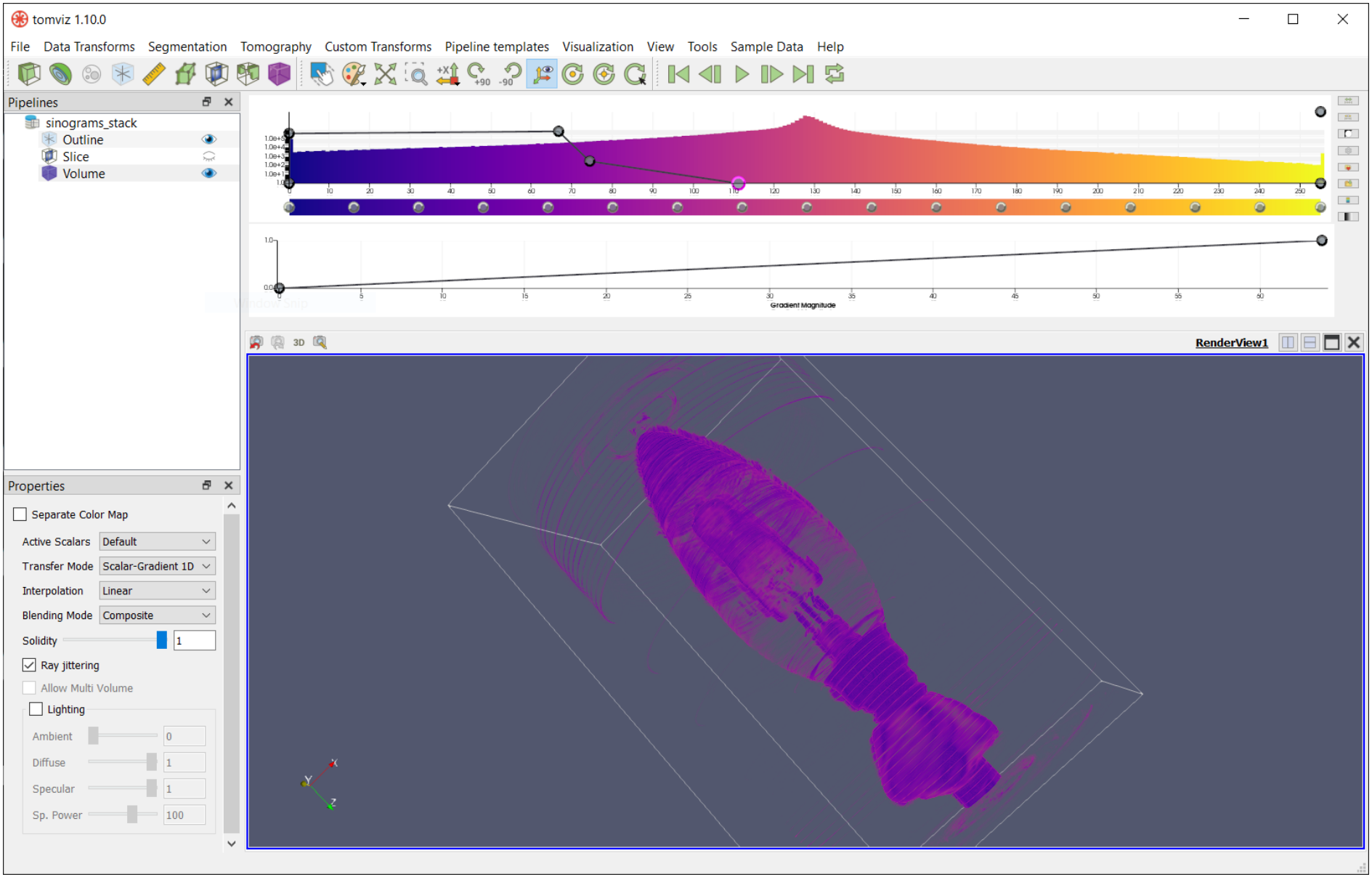




Coupes suivant d'autres directions



Reconstruction complète en 3D



MERCI POUR VOTRE ATTENTION

Annexe 1 : extract_frames.py

```python
import cv2
import os


def extract_frames(video_path, output_folder):
    """
    Extrait toutes les images d'une vidéo et les enregistre sous forme de fichiers images individuels.

    Args :
        video_path (str) : Le chemin complet du fichier vidéo d'entrée.
        output_folder (str) : Le chemin vers le dossier où les images seront sauvegardées.
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    video_capture = cv2.VideoCapture(video_path)
    frame_count = 0
    while True:
        success, frame = video_capture.read()

        if not success:
            break

        frame_filename = os.path.join(output_folder, f"frame_{frame_count}.jpg")
        cv2.imwrite(frame_filename, frame)
        frame_count += 1

    video_capture.release()
    print(f"Extraction réussie de {frame_count} images vers {output_folder}")


if __name__ == '__main__':
    video_file = "bulb.mp4"
    output_directory = "frames"

    extract_frames(video_file, output_directory)
```

Annexe 2 : fix_perspective.py

```python
import os
import cv2
import numpy as np


def fix_perspective(i, image, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Corners: Top-left, Top-right, Bottom-right, Bottom-left
    corners = np.float32([[69, 24], [351, 28], [347, 527], [63, 524]])
    (tl, tr, br, bl) = corners

    # Calculer la largeur de la nouvelle image
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    # Calculer la hauteur de la nouvelle image
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    output_points = np.array([[0, 0], [maxWidth - 1, 0], [maxWidth - 1, maxHeight - 1], [0, maxHeight - 1]], dtype="float32")

    perspective_matrix = cv2.getPerspectiveTransform(corners, output_points)
    warped_image = cv2.warpPerspective(image, perspective_matrix, (maxWidth, maxHeight))

    output_path = os.path.join(output_folder, f"warped_{i}.jpg")
    cv2.imwrite(output_path, warped_image)


if __name__ == '__main__':
    input_directory = "frames"
    for i, filename in enumerate(os.listdir(input_directory)):
        file_path = os.path.join(input_directory, filename)
        image = cv2.imread(file_path)

        fix_perspective(i, image, "warped")
```

Annexe 3 : generate_sinogram.py

```python
import os
import cv2
import numpy as np


def generate_sinogram(input_directory, output_folder):
    height = cv2.imread(f"{input_directory}\\0.jpg").shape[0]

    for j in range(0, height):
        line_y = j
        lines = []

        for i, filename in enumerate(os.listdir(input_directory)):
            file_path = os.path.join(input_directory, filename)
            img = cv2.imread(file_path)

            height = img.shape[0]
            y = min(line_y, height - 1)
            line = img[y:y + 1, :, :]
            lines.append(line)

        stacked = np.vstack(lines)
        output_path = os.path.join(output_folder, f"sinogram_{j}.jpg")
        cv2.imwrite(output_path, stacked)


if __name__ == '__main__':
    generate_sinogram("warped", "sinograms")

```

Annexe 4 : backproject.py

```python
import numpy as np
import imutils
from skimage.transform import rotate
import scipy.fftpack as fft
import matplotlib.pyplot as plt
import os


def ramp_filter(ffts):
    ramp = np.floor(np.arange(0.5, ffts.shape[1] // 2 + 0.1, 0.5))
    return ffts * ramp


def back_project(operator):
    laminogram = np.zeros((operator.shape[1], operator.shape[1]))
    dTheta = 180.0 / operator.shape[0]
    for i in range(operator.shape[0]):
        temp = np.tile(operator[i], (operator.shape[1], 1))
        temp = rotate(temp, dTheta * i)
        laminogram += temp
    return laminogram


def normalize_image(image, target_value):
    min_val = np.min(image)
    max_val = np.max(image)

    if max_val - min_val > 0:
        image = (image - min_val) / (max_val - min_val)

    region = image[0:10, 0:10]
    region_avg = np.mean(region)
    bias = target_value - region_avg
    adjusted_image = image + bias

    return np.clip(adjusted_image, 0, 1)
```

Annexe 4 : backproject.py

```python
def fbp(input_directory, output_folder):
    for i, filename in enumerate(os.listdir(input_directory)):
        file_path = os.path.join(input_directory, filename)
        sinogram = imutils.imread(file_path)

        frequency_domain_sinogram = fft.rfft(sinogram, axis=1)
        filtered_frequency_domain_sinogram = ramp_filter(frequency_domain_sinogram)
        filtered_spatial_domain_sinogram = fft.irfft(filtered_frequency_domain_sinogram, axis=1)
        reconstructed_image = back_project(filtered_spatial_domain_sinogram)

        normalized_image = normalize_image(reconstructed_image, target_value=0.9)

        output_path = os.path.join(output_folder, f"slice_{i}.jpg")
        plt.imsave(output_path, normalized_image, cmap='gray')


if __name__ == '__main__':
    fbp("sinograms", "backprojections")
```

Annexe 5 : Preuve de Rétroprojection

$$
\begin{aligned}
b(x,y) &= \int_0^{\pi} p(s,\phi)\big|_{s=x\cos\phi+y\sin\phi}\, d\phi \\
&= \int_0^{\pi} \left[\iint_{\mathbb{R}^2} f(x',y')\delta(x'\cos\phi + y'\sin\phi - s)\, dx'\, dy' \right]_{s=x\cos\phi+y\sin\phi} d \\
&= \int_0^{\pi} \iint_{\mathbb{R}^2} f(x',y')\delta((x'-x)\cos\phi + (y'-y)\sin\phi)\, dx'\, dy'\, d\phi \\
&= \iint_{\mathbb{R}^2} f(x',y') \underbrace{\left[\int_0^{\pi} \delta((x'-x)\cos\phi + (y'-y)\sin\phi)\, d\phi \right]}_{\text{Appelons cette intégrale } I} dx'\, dy'
\end{aligned}
$$

Annexe 5 : Évaluer l'intégrale I

La fonction delta possède une propriété pertinente $g(\phi)$:

$$\int \delta(g(\phi))\, d\phi = \sum_i \frac{1}{|g'(\phi_i)|} \quad \text{où } g(\phi_i) = 0$$

Annexe 5 : Évaluer l'intégrale I

La fonction delta possède une propriété pertinente $g(\phi)$:

$$\int \delta(g(\phi))\, d\phi = \sum_i \frac{1}{|g'(\phi_i)|} \quad \text{où } g(\phi_i) = 0$$

1. Définir $g(\phi)$ et trouver sa racine ϕ_0

Posons $u = x' - x$ and $v = y' - y$:

$$\begin{aligned} g(\phi) = 0 &\Leftrightarrow u\cos\phi = -v\sin\phi \\ &\Leftrightarrow \tan(\phi_0) = -\frac{u}{v} \end{aligned}$$

Cette équation possède exactement une racine, ϕ_0, dans notre intervalle d'intégration $\phi \in [0, \pi]$.

Annexe 5 : Évaluer l'intégrale I (cont.)

2. Trouver la dérivée $|g'(\phi_0)|$

$$g'(\phi) = -u \sin \phi + v \cos \phi$$

Annexe 5 : Évaluer l'intégrale I (cont.)

2. Trouver la dérivée $|g'(\phi_0)|$

$$g'(\phi) = -u\sin\phi + v\cos\phi$$

Puisque $\tan(\phi_0) = -u/v$,la géométrie du triangle rectangle correspondant donne :

$$|\cos\phi_0| = \frac{|v|}{\sqrt{u^2+v^2}} \quad \text{et} \quad |\sin\phi_0| = \frac{|u|}{\sqrt{u^2+v^2}}$$

Annexe 5 : Évaluer l'intégrale I (cont.)

2. Trouver la dérivée $|g'(\phi_0)|$

$$g'(\phi) = -u\sin\phi + v\cos\phi$$

Puisque $\tan(\phi_0) = -u/v$,la géométrie du triangle rectangle correspondant donne :

$$|\cos\phi_0| = \frac{|v|}{\sqrt{u^2+v^2}} \quad \text{et} \quad |\sin\phi_0| = \frac{|u|}{\sqrt{u^2+v^2}}$$

Donc :

$$\begin{aligned} |g'(\phi_0)| &= |-u\sin\phi_0 + v\cos\phi_0| \\ &= \frac{u^2}{\sqrt{u^2+v^2}} + \frac{v^2}{\sqrt{u^2+v^2}} = \frac{u^2+v^2}{\sqrt{u^2+v^2}} = \sqrt{u^2+v^2} \end{aligned}$$

Annexe 5 : Évaluer l'intégrale I (cont.)

2. Trouver la dérivée $|g'(\phi_0)|$

$$g'(\phi) = -u\sin\phi + v\cos\phi$$

Puisque $\tan(\phi_0) = -u/v$,la géométrie du triangle rectangle correspondant donne :

$$|\cos\phi_0| = \frac{|v|}{\sqrt{u^2+v^2}} \quad \text{et} \quad |\sin\phi_0| = \frac{|u|}{\sqrt{u^2+v^2}}$$

Donc :

$$\begin{aligned} |g'(\phi_0)| &= |-u\sin\phi_0 + v\cos\phi_0| \\ &= \frac{u^2}{\sqrt{u^2+v^2}} + \frac{v^2}{\sqrt{u^2+v^2}} = \frac{u^2+v^2}{\sqrt{u^2+v^2}} = \sqrt{u^2+v^2} \end{aligned}$$

L'intégrale $I = \frac{1}{|g'(\phi_0)|} = \frac{1}{\sqrt{u^2+v^2}} = \frac{1}{\sqrt{(x'-x)^2+(y'-y)^2}}$

Annexe 5 : Identification de la convolution

$$b(x,y) = \iint_{\mathbb{R}^2} f(x',y') \underbrace{\left[\int_0^{\pi} \delta((x'-x)\cos\phi + (y'-y)\sin\phi)\, d\phi\right]}_{\text{Appelons cette intégrale } I} dx'\, dy'$$

$$= \iint_{\mathbb{R}^2} f(x',y') \cdot \frac{1}{\sqrt{(x'-x)^2+(y'-y)^2}}\, dx'\, dy'$$

$$= \iint_{\mathbb{R}^2} f(x',y') \cdot \frac{1}{\sqrt{(x-x')^2+(y-y')^2}}\, dx'\, dy'$$

C'est précisément la définition d'une convolution 2D.

Résultat final

$$b(x,y) = f(x,y) * \frac{1}{\sqrt{x^2+y^2}}$$