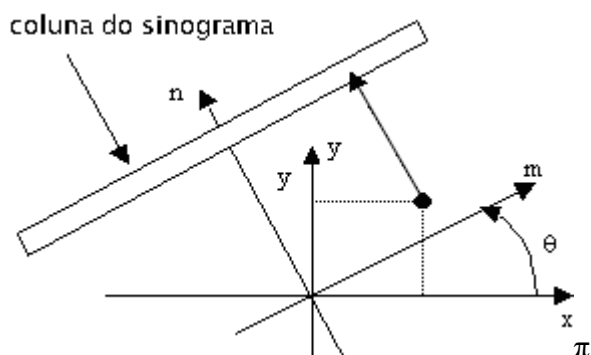


Transformada de Radon e retro-projeção

1. Transformada de Radon

A Transformada de Radon (ou de projeção) ao ângulo θ corresponde às integrais de linha de uma imagem $f(x,y)$ perpendicular à direção θ . Na prática, o ângulo é amostrado uniformemente entre $[-\pi/2 \ \pi/2]$. A projeção ao ângulo θ_k é armazenado em uma coluna do sinograma $p(k, m)$.



A coordenada (x,y) é obtido girando a grade de coordenadas (m,n) por um ângulo de θ em torno do centro da imagem quadrada (c,c) . É dada pela relação:

$$\begin{pmatrix} x - c \\ y - c \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} m - c \\ n - c \end{pmatrix}$$

O método `transformRadon()` no arquivo `Radon.java` implementa a Transformada de Radon de uma imagem quadrada. O método requer uma interpolação 2D que é realizada pelo método `getInterpolatedPixel2D()` (interpolação linear).

Notas de implementação:

- Para encurtar tempo de computação, a imagem é copiada primeiro em uma “array” 2D que usa o método:

```
double[][] array = input.getArrayPixels();  
double v = array[i][j]; // acesso ao pixel (i,j)
```
- O método padrão `Math.floor()` de Java é muito lento; assim nós usaremos um método `floor()` disponível que é executado mais rapidamente.
- A varredura da imagem (índice m, n) é restrita a um disco de diâmetro igual ao tamanho da imagem.

Entenda o código e execute nas imagens de teste (`point.tif`, `lines.tif`, `circles.tif` e `phantom.tif`) especificando números diferentes de projeções. Escolhendo as opções apropriadas na caixa de diálogo, podem-se executar as três operações independentemente: (1) Transformada de Radon, (2) filtragem e (3) retro-projeção. Não se esqueça de selecionar a imagem de entrada correta.

2. Transformada de Retro-projeção.

O de imagem resultante $b(x,y)$ é obtido através da retro-projeção do sinograma $p(k, m)$:

$$b(x, y) = \frac{\pi}{nbAngulos} \sum_k p(k, m(x, y, \theta_k))$$

com m expresso como uma função de x , y e θ_k .

Note que a retro-projeção é essencialmente o fluxograma transposto do transformRadon(). Codifique o método inverseRadon() que computa o inverso do transformada de Radon de um sinograma no arquivo Radon.java.

O método requer uma interpolação 1D (unidimensional) que você deverá que implementar no método getInterpolatedPixel1D() (interpolação linear).

Para fazer um método computacional mais eficiente, calcule uma “array” 2D (double $b[] []$) e ponha esta “array” em uma ImageAccess ao fim do processo.

```
double b[][] = new double[size][size];  
.  
.  
.  
ImageAccess ReconstudedImage = new ImageAccess(b);
```

As imagens de teste são: point.tif, lines.tif, circles.tif e phantom.tif.

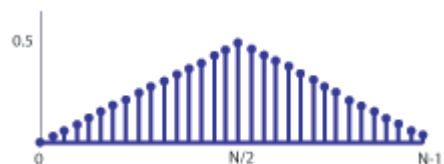
3. Sinograma filtrado

Propomos implementar três filtros: (1) Ram-Lak e (2) Co-seno no domínio de Fourier, e (3) Laplaciano no domínio espacial.

3.1 Ram-Lak filtro

O Ram-Lak filtro está definido por sua resposta de frequência $H(w) = |w|$. O método applyRamLakFilter() rotaciona pelos ângulos (k) e filtra cada coluna do sinograma através de multiplicação com a função de transferência no domínio de Fourier.

Codifique o método generateRamLakFilter() isso devolve uma ordem que corresponde ao diagrama seguinte.



Nota de implementação:

Os métodos fft.transform(real, imaginário),
fft.inverse(real, imaginário) processam os dados
localmente (por referência); isto significa que as
saídas são postas nos mesmos arranjos (array) das

entradas.

3.2 filtro de co-seno

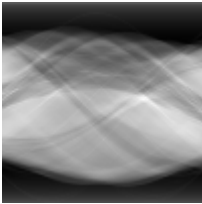
Codifique o filtro de co-seno de um modo semelhante usando o método `applyCosineFilter()`. A resposta em frequência do filtro de co-seno é:

$$H(w) = |w| \cdot \cos(\pi \cdot |w|)$$

3.3 filtro Laplaciano

Codifique uma versão digital do filtro Laplaciano no método `applyLaplacianFilter()`. Aqui também, o filtro Laplaciano que está definido pela máscara $[1, -2, 1]$, é aplicado separadamente a cada coluna do sinograma (condições limites de espelho aplicadas).

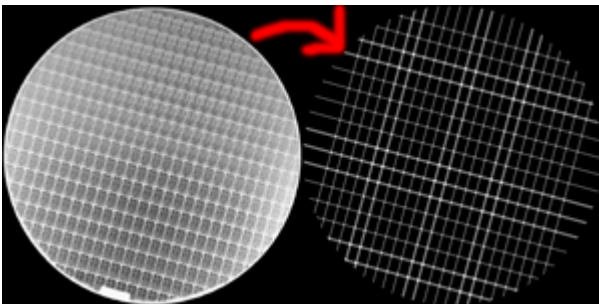
4. Reconstrução de um sinograma



Escolha um filtro apropriado para reconstruir este sinograma dado no arquivo `ctslicesino.tif`

Salve a imagem (8 bits) e insira no `relatorio.doc`.

5. Localização de linhas



Use a Transformada de Radon para localizar as linhas em uma imagem `wafer.tif`.

Sugestão: O sinograma pode ser “thresholded” usando o comando `ImageJ` (Imagem->Adjust->Threshold).

Salve a imagem (8 bits) e insira no `relatorio.doc`.