PMR2560 – Visão Computacional
Detecção de linhas

Prof. Eduardo L. L. Cabral

Objetivos

- Processamento de imagens:
  - Detecção de linhas;
  - Transformada de Hough;
  - Detecção de cantos.

Detecção de linhas

- Linhas são características importantes pois definem os contornos de objetos em uma imagem.
- Exemplos:
  - Faixa no chão;
  - Linha de parede;
  - Abertura de portas;
  - Escadas.
- Algoritmos para extrair linhas iniciam com a detecção de bordas na imagem.

Detecção de linhas

- **Como detectar linhas?**

- Dada uma imagem de bordas ⇒ deseja-se agrupar grupos de pixels de bordas para identificar as linhas presentes na imagem.
- Uma forma de realizar essa operação é correspondência com padrões na imagem de bordas:
  - Exige a geração de padrões de linhas e a sua convolução com a imagem;
  - Exige um grande número de padrões para localizar todos os tipos de linhas com precisão ⇒ computacionalmente muito intensivo.
- Outra forma é realizar uma transformação onde os pixels de bordas "votam" por linhas:
  - Esse método reduz o problema a um processo de limiarização;
  - **Transformada de Hough.**

Transformada de Hough

- Idéia básica da Transformada de Hough:

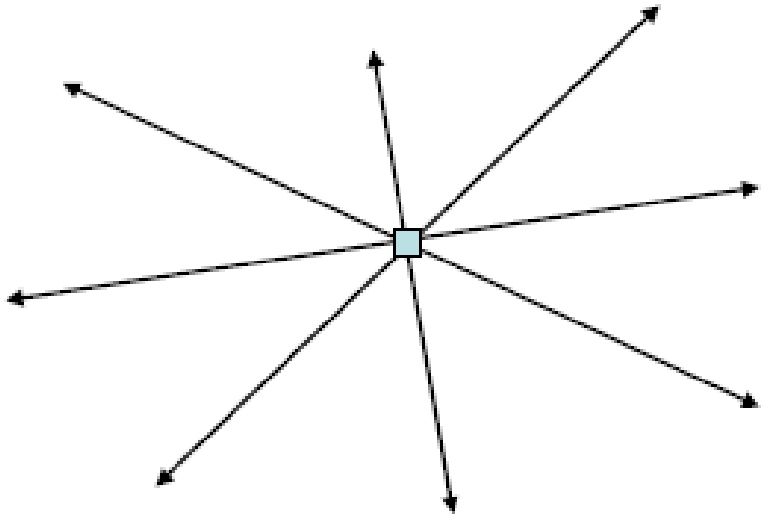
1. Uma linha i na imagem pode ser parametrizada por duas variáveis:

$$y = a_i x + b_i$$

2. Cada pixel de borda (x, y) pode corresponder a uma família de linhas $L(x, y) = \{l_1,\ldots,l_n\}$;
3. Cada pixel de broda (x, y) "vota" para uma linha l_i da família $L(x, y)$;
4. Cada pixel de borda que forma uma linha gera um voto para uma determinada linha i;
5. Linhas que de fato estão presentes na imagem receberão mais votos do que as linhas que não existem.

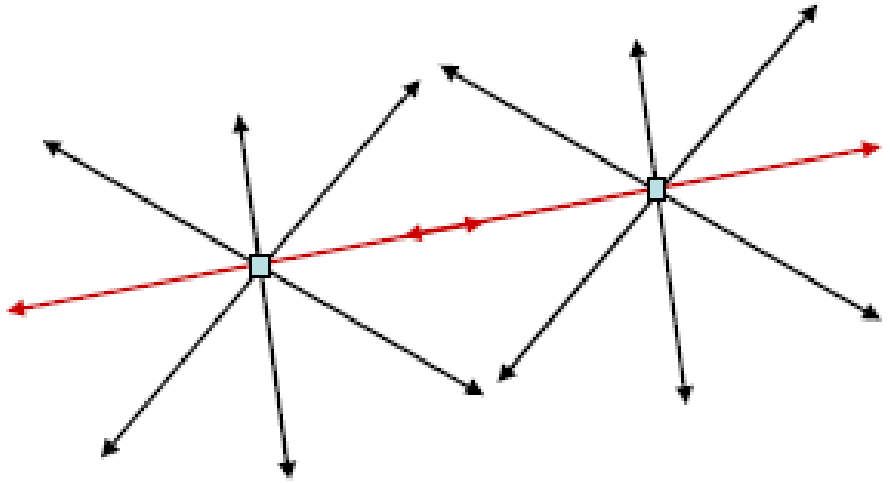
Transformada de Hough

- Cada pixel de borda (x, y) corresponde a uma família de linhas $L(x, y) = \{l_1,\ldots,l_n\}$.
- Cada pixel de borda (x, y) "vota" para uma linha l_i de $L(x, y)$.



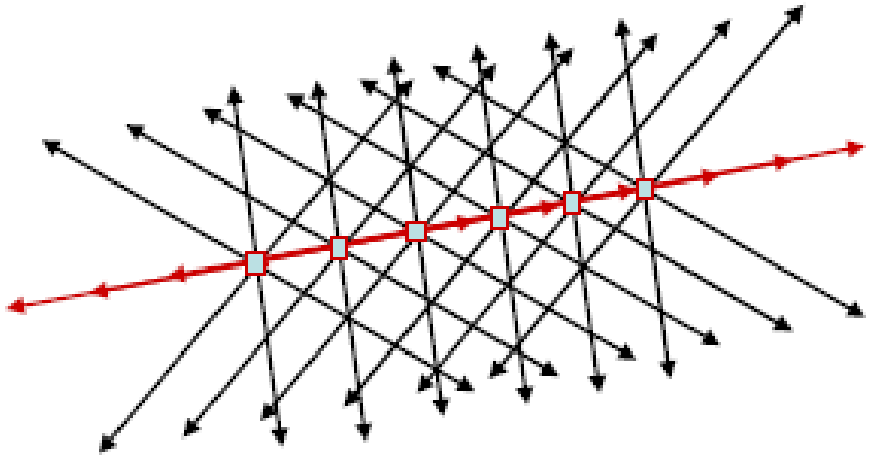
Transformada de Hough

- Cada pixel de borda que forma uma linha gera um voto para essa linha entre muitas outras linhas.



Transformada de Hough

- Linhas que de fato estão presentes na imagem recebem mais votos do que as linhas que não existem.



Transformada de Hough

- O problema de detecção de linhas se transforma em um problema de detecção de picos $\Rightarrow$ é necessário somente detectar as linhas com mais votos.
- Problema $\Rightarrow$ representação de uma linha:

$$y = a_i x + b_i,$$

onde: $a_i \in [-\infty, +\infty]$

$b_i \in [-\infty, +\infty]$

Como representar a_i e b_i?

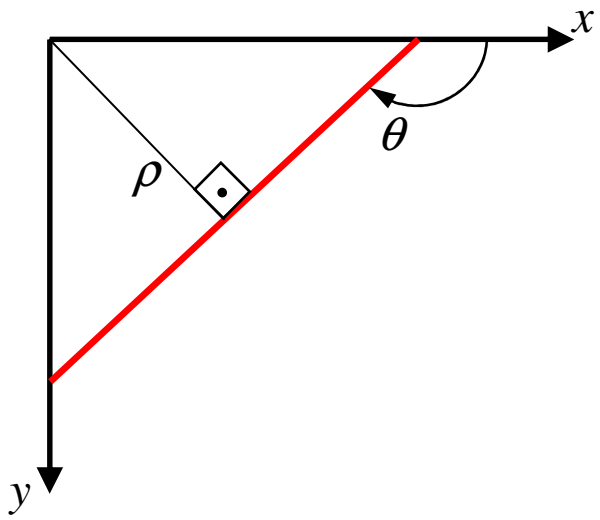
- Solução $\Rightarrow$ usar representação polar da reta.

Transformada de Hough

- No lugar da representação cartesiana da reta pode-se usar a representação polar de uma reta:

$$\rho = -x\cos\theta + y\sin\theta$$

onde ρ é a distância normal de um ponto à reta (em geral a origem do sistema de coordenadas) e θ é o ângulo de inclinação da reta com a horizontal.



Transformada de Hough

- Esses parâmetros são limitados por:

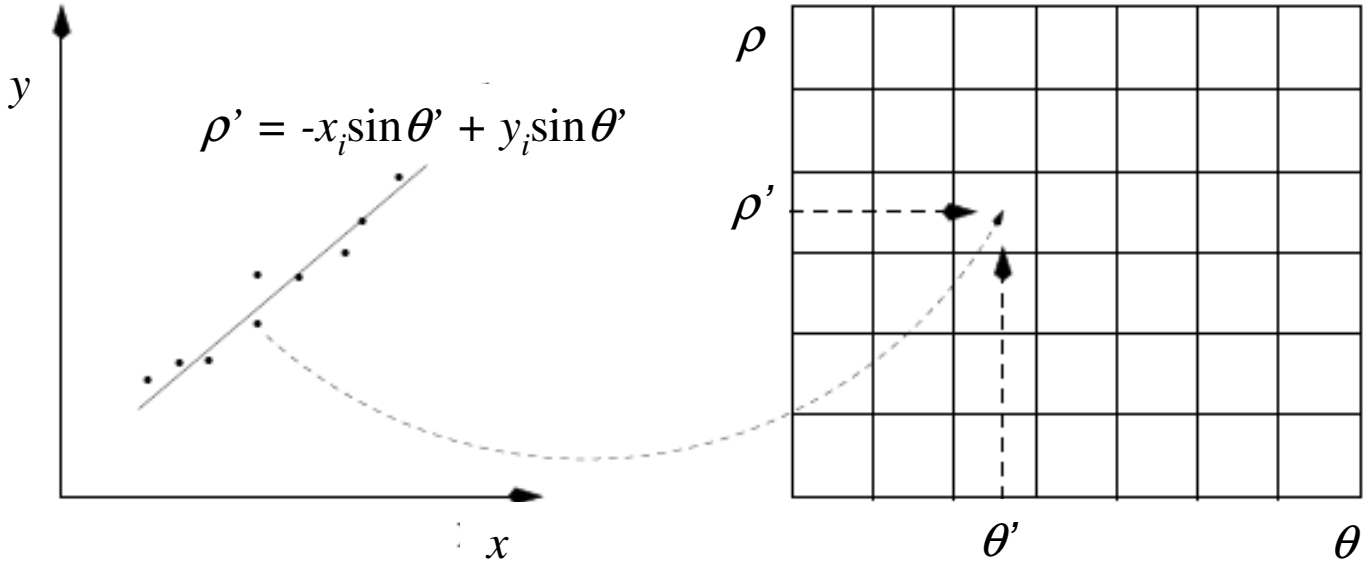
$$\rho \in \left[0, \sqrt{L_x^2 + L_y^2}\right]$$

$$\theta \in \left[0, 180^o\right]$$

que podem ser discretizados com a resolução desejada.

- Os parâmetros ρ e θ podem ser estimados mais precisamente usando-se uma discretização “fina”.
- Quanto mais fina for a discretização $\Rightarrow$ maior as exigências de espaço de memória e tempo de processamento.
- Para uma maior tolerância a ruídos uma discretização grosseira é melhor.

Transformada de Hough



- Quanto mais grosseira a discretização dos parâmetros ⇒ maior a probabilidade de cada ponto "votar" na mesma célula ρ' e θ'.

Transformada de Hough

- O Algoritmo:

1. Dada uma imagem de bordas **B**(i,j);
2. Definir uma resolução $\Delta\theta$ e $\Delta\rho$ para θ e ρ respectivamente;
3. Construir uma matriz **A**(m,n) para acumular os votos das linhas, onde:

$$m = \frac{1}{\Delta\rho}\rho_{\max} \qquad n = \frac{180^{o}}{\Delta\theta}$$

4. Para cada pixel de borda (i, j) fazer:

> *Para* $\theta = \Delta\theta:\Delta\theta:180^{o}$;
>
> $\rho = -i*\cos(\theta) + j*\sin(\theta)$;
>
> *Se* $0 < \rho < \rho_{max}$, *então:*
>
> *Arredondar para o inteiro mais próximo;*
>
> **A**(θ,ρ) = **A**(θ,ρ) + 1;
>
> *Fim.*

5. Limiarizar a matriz **A**(i,j) para achar as linhas principais.

Transformada de Hough

- Exemplo:

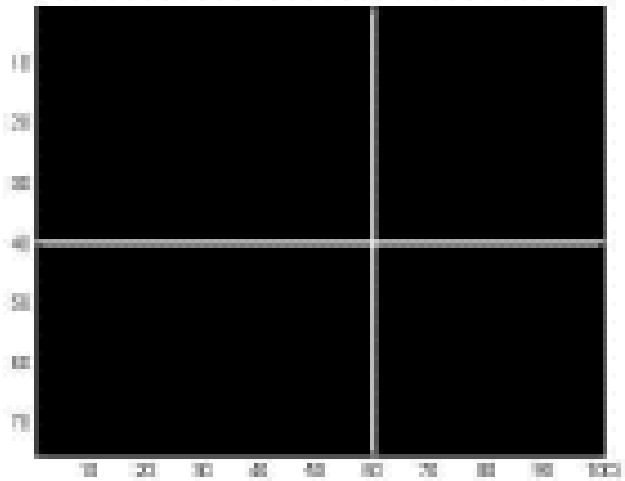
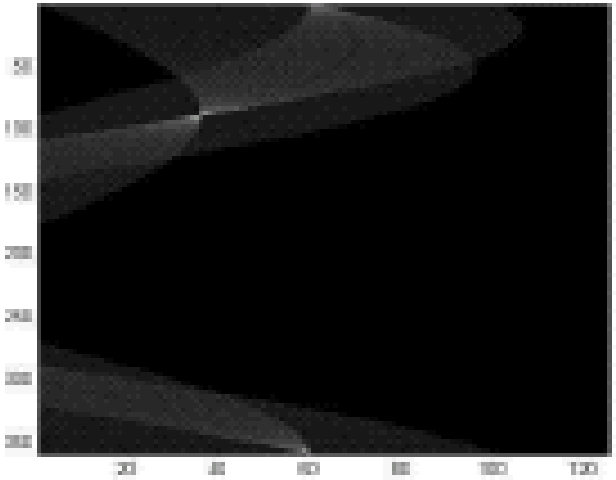
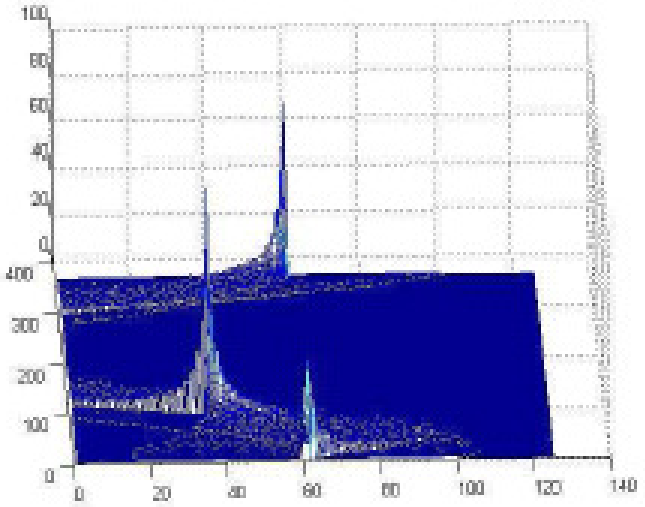


Imagem de bodas
$\mathbf{B}(i,j)$



Matriz de votação
$\mathbf{A}(\rho,\theta)$



Transformada de Hough

- Exemplo:

Imagem original

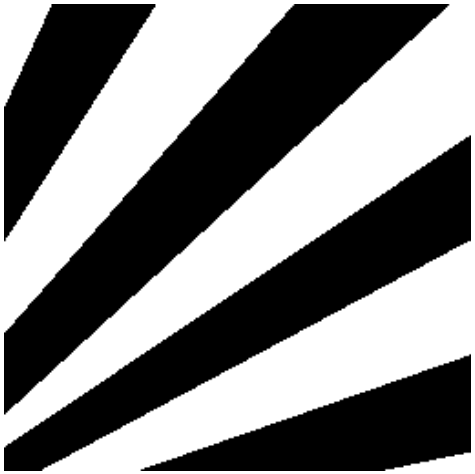
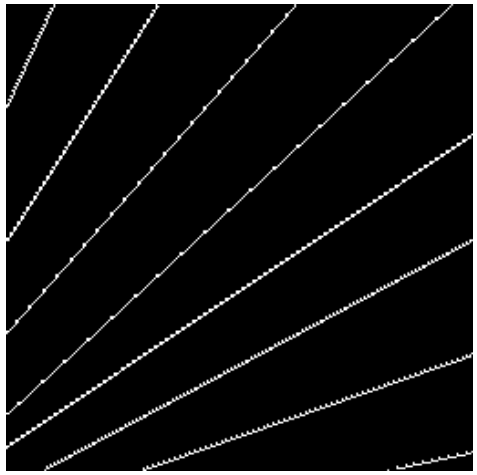
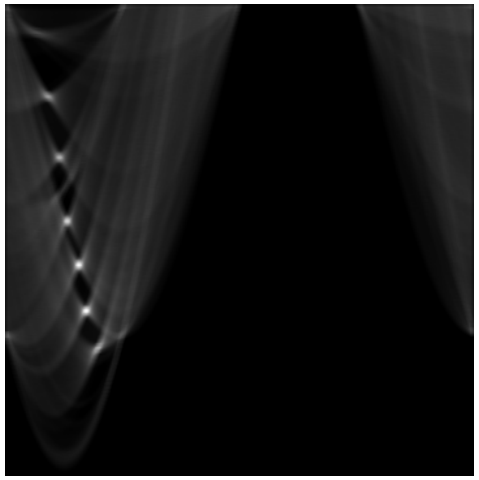


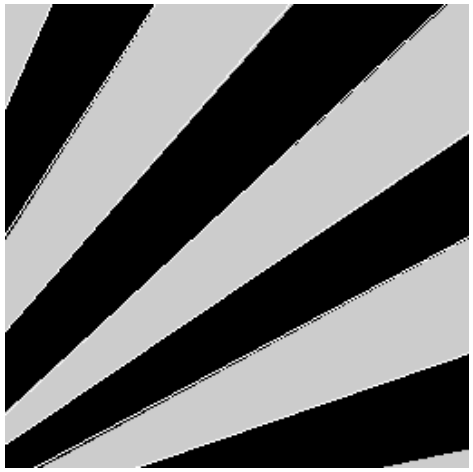
Imagem de bordas



Matriz de votação



Linhas detectadas



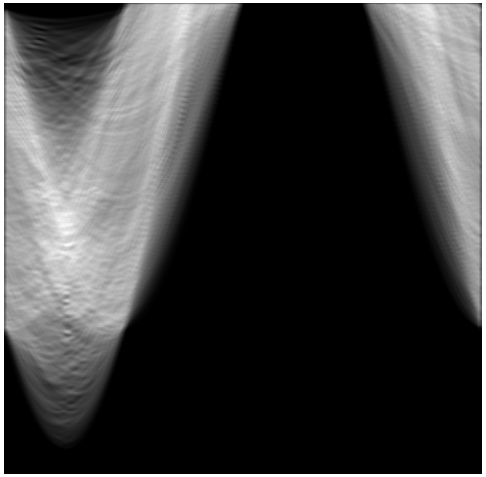
Transformada de Hough

- Exemplo:

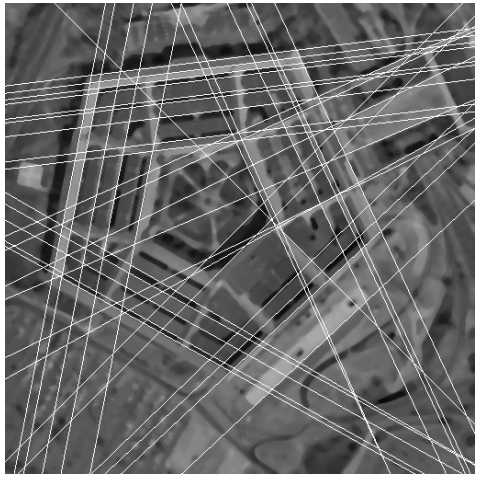
Imagem original

Imagem de bordas

Matriz de votação



Linhas detectadas



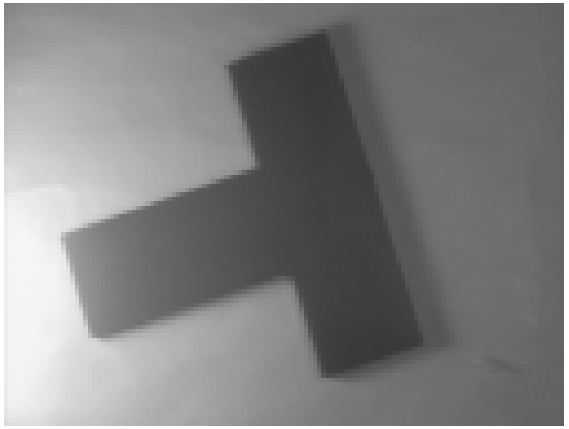
Transformada de Hough

- Observações:
  - Pelo fato de ser um processo de votação, a Transformada de Hough (TH) é robusta a ruídos e pode detectar linhas quebradas, desconectadas ou parcialmente oclusas.
  - A TH também pode ser utilizada para detectar linhas mais complexas, tais como, círculos, elipses etc.
  - A complexidade do algoritmo aumenta exponencialmente com o tipo de curva a ser detectada $\Rightarrow$ por isso a TH é mais utilizada para detectar curvas simples como as retas.

Exercícios

1. Calcule a Transformada de Hough dos objetos abaixo.

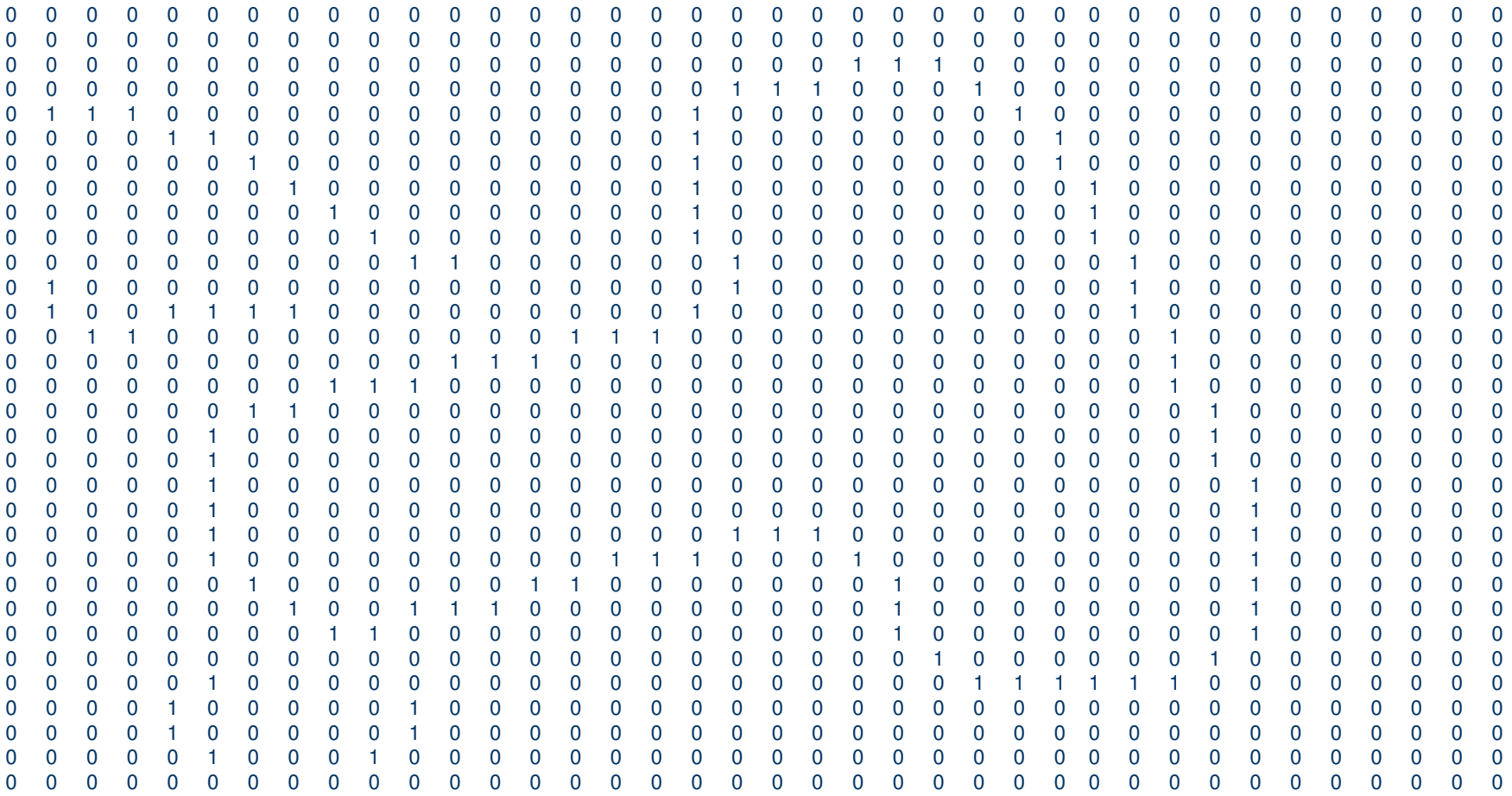
2. Detecte as linhas presentes na imagem abaixo dada pela matriz de bordas da próxima página.



Exercícios

80	80	63	80	80	80	63	80	80	80	63	80	80	80	63	80	63	46	63	46	63	46	63	46	63	46	46	46	63	46	46	46	46
97	80	97	97	97	80	97	97	97	63	80	80	97	63	80	46	80	63	80	46	80	46	80	46	46	29	46	46	63	46	46	46	46
97	80	80	97	97	97	97	97	97	80	80	80	80	80	80	80	63	80	63	80	63	46	12	0	12	29	29	46	46	46	46	46	46
97	97	97	97	114	97	97	97	114	97	97	80	97	80	97	63	80	80	80	29	29	0	0	0	29	29	46	46	63	46	46	46	46
114	97	97	97	114	97	97	97	114	97	97	80	80	97	63	80	63	29	0	0	12	0	0	0	12	29	12	29	63	46	46	46	63
131	114	131	97	131	114	114	97	114	97	97	97	97	80	97	80	80	0	29	0	12	0	0	0	0	12	29	46	63	46	46	46	63
114	131	114	131	114	131	114	97	114	97	97	97	97	80	80	80	80	29	12	0	12	0	12	0	0	12	12	29	46	46	63	46	46
131	131	131	114	131	114	131	97	131	97	97	97	97	97	97	80	97	46	29	0	29	0	12	0	12	0	46	29	46	46	63	46	63
131	131	114	131	131	131	114	131	114	97	97	97	97	97	97	80	80	80	12	0	12	0	0	0	12	0	12	29	29	46	46	46	63
148	131	148	131	148	114	131	97	131	97	114	97	114	97	97	97	97	80	29	0	29	0	12	0	12	0	29	12	46	46	63	46	63
148	148	131	131	131	131	114	131	114	131	114	97	114	97	97	97	97	80	46	0	12	0	12	0	0	0	12	29	29	46	63	46	46
165	148	148	148	148	131	148	114	131	114	131	97	131	97	114	97	97	97	80	0	29	0	29	0	12	0	12	12	29	46	80	46	80
165	148	148	148	148	148	131	131	131	131	114	131	114	97	97	97	114	97	63	29	12	0	12	0	12	0	0	29	29	46	46	46	63
182	148	165	148	165	148	148	148	148	114	131	97	131	114	114	63	46	12	29	0	29	12	12	0	29	0	12	0	46	12	46	46	63
165	182	165	182	165	148	165	148	148	131	131	131	97	46	29	29	12	29	12	12	12	12	12	0	12	0	12	0	12	29	46	46	46
189	182	182	165	182	148	182	148	148	97	80	29	46	29	46	12	29	29	29	12	29	0	29	0	29	0	29	0	29	29	46	46	80
182	182	165	182	165	148	131	80	63	46	46	29	29	29	29	29	29	29	12	29	12	0	12	0	12	0	12	0	12	29	29	46	63
189	182	182	165	148	80	80	46	80	46	46	46	46	29	46	29	46	12	29	0	29	12	29	0	12	0	29	0	29	12	29	46	63
182	182	182	182	148	80	80	80	63	46	46	46	46	29	29	29	12	29	12	12	12	12	12	0	12	0	12	0	12	29	29	29	46
189	182	189	182	182	80	97	63	80	46	63	46	46	29	46	29	29	29	29	12	29	0	29	0	29	0	29	0	29	12	46	29	46
182	189	182	182	182	97	63	80	63	46	46	46	29	46	29	29	29	29	12	29	12	0	12	0	12	0	0	0	12	0	12	29	29
189	182	189	189	189	114	80	63	80	46	63	46	46	29	46	29	29	12	46	46	80	63	29	0	29	0	12	0	12	0	29	29	46
182	182	182	182	182	148	80	80	63	46	46	46	46	29	29	46	63	80	80	80	80	80	12	0	12	0	12	0	12	0	12	29	29
189	182	189	182	189	165	97	63	80	46	46	29	63	63	97	97	114	97	97	80	97	80	46	0	29	0	29	0	12	0	29	12	29
182	189	182	182	182	182	97	80	63	46	63	97	114	97	114	97	114	97	97	80	80	80	63	0	12	0	0	0	12	0	0	29	29
189	182	189	182	189	165	148	97	131	114	131	114	131	114	131	97	114	97	97	97	97	80	80	0	12	0	0	0	12	0	0	0	46
182	182	182	182	165	148	165	148	148	131	131	131	114	131	114	97	97	97	80	97	80	80	80	29	0	0	0	0	0	0	12	29	46
189	182	189	165	182	148	148	148	148	131	148	114	131	97	131	97	97	97	97	80	97	80	97	46	0	0	29	12	46	46	46	46	63
182	182	165	182	165	148	148	148	131	148	114	131	114	97	114	97	97	97	97	80	80	80	63	80	63	46	46	46	63	46	46	46	63
189	182	182	148	182	148	148	148	148	131	131	97	131	97	131	97	97	97	97	97	97	63	80	63	80	63	80	46	63	46	63	46	63
165	182	165	182	165	148	148	148	148	131	114	97	114	97	114	97	97	97	80	80	80	80	63	80	63	63	63	46	63	46	63	46	46
182	148	182	148	165	148	148	131	131	131	131	97	131	97	114	97	97	97	97	80	80	63	80	63	80	46	80	46	80	46	63	46	63

Exercícios



Exercícios