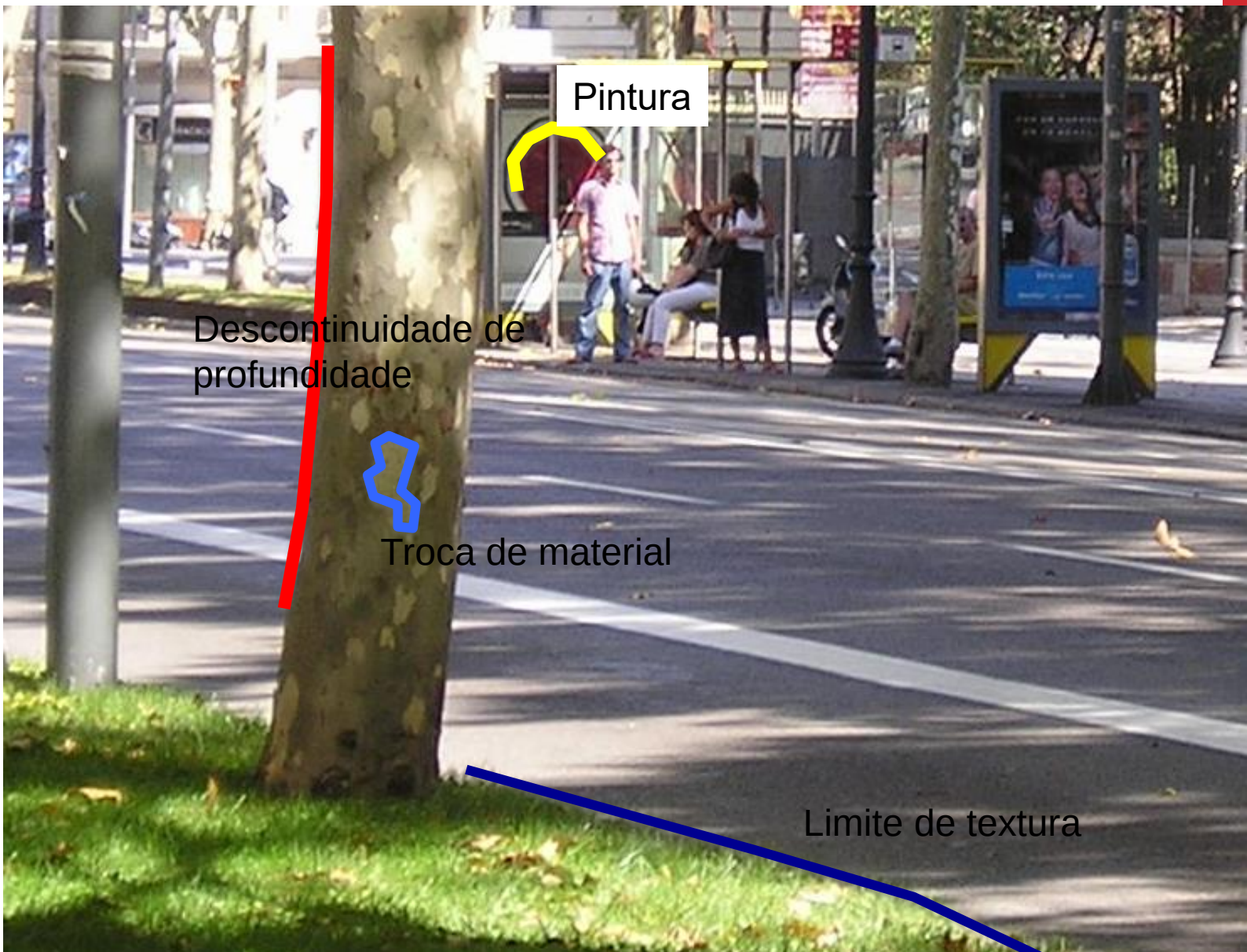
DETECÇÃO DE BORDAS

PROF. VALMIR MACÁRIO FILHO

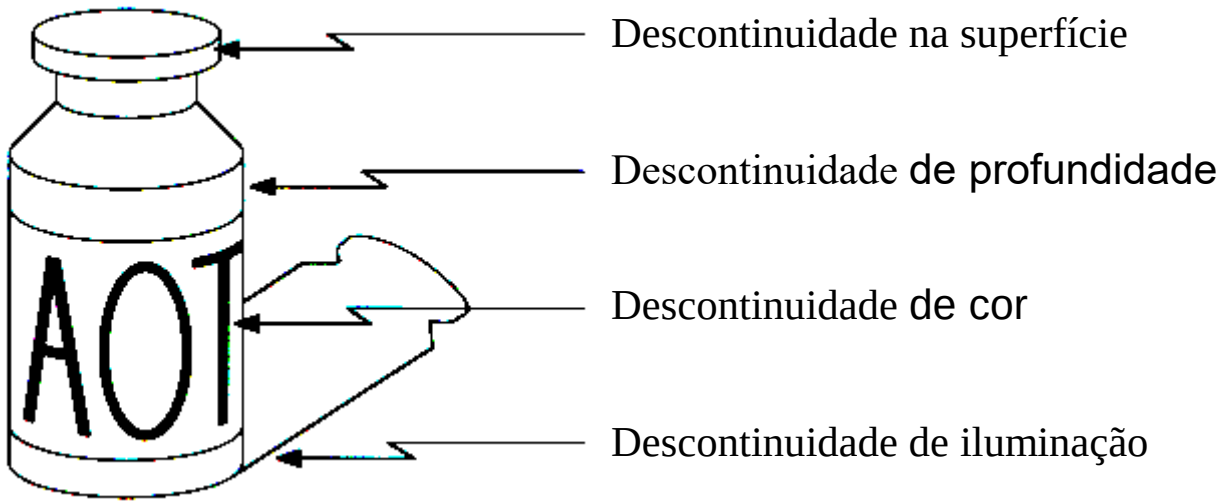
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

O QUE É UMA BORDA?

O QUE É UMA BORDA?



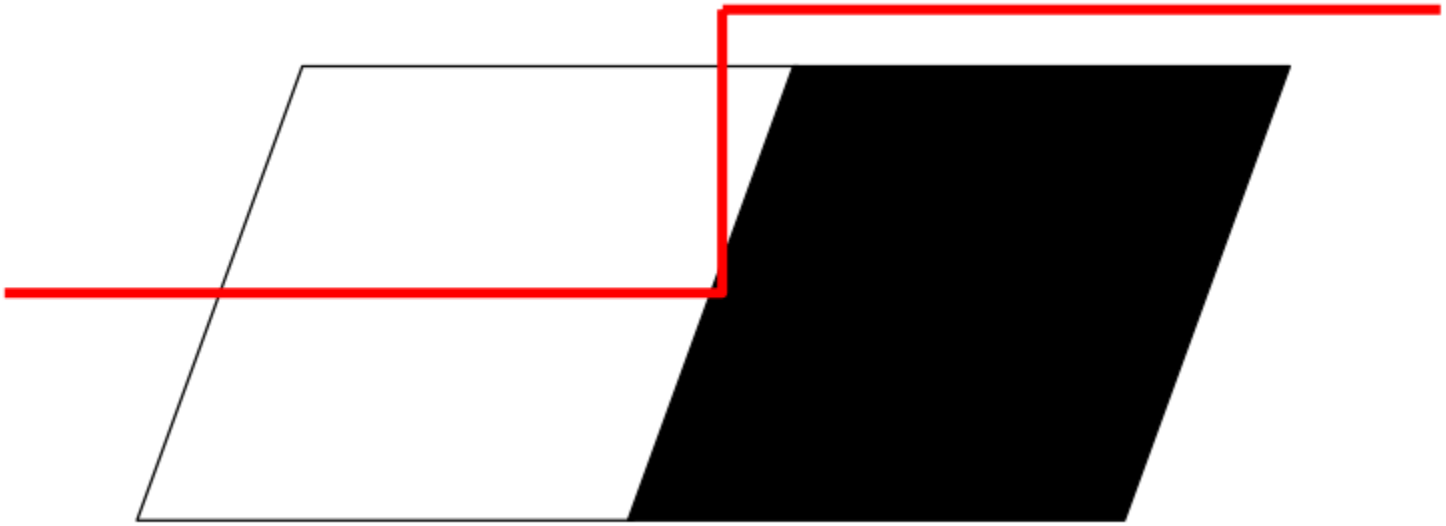
ORIGEM DAS BORDAS



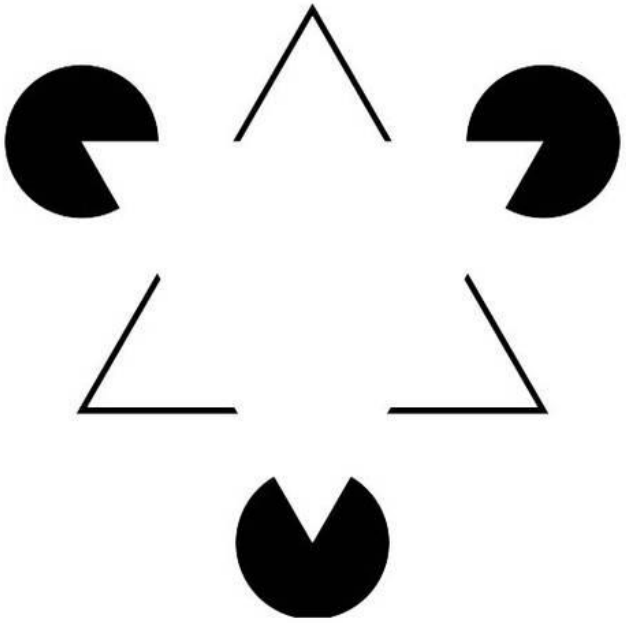
Bordas são geradas por uma infinidade de causas

ORIGEM DAS BORDAS

- As bordas ocorrem junto com mudanças



CONTORNOS ILUSÓRIOS



DETECÇÃO DE DESCONTINUIDADES

- **Detecção de pontos**

-1	-1	-1
-1	8	-1
-1	-1	-1

DETECÇÃO DE DESCONTINUIDADES

- **Detecção de pontos**

DETECÇÃO DE DESCONTINUIDADES

- **Detecção de pontos**

DETECÇÃO DE DESCONTINUIDADES

- **Detecção de linhas**

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

+45°

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45°

Intensifica regiões horizontais em uma imagem

DETECÇÃO DE DESCONTINUIDADES

- **Detecção de Bordas**
  - Abordagem mais comum para detecção de descontinuidades
  - Uma borda é o limite entre duas regiões com propriedades relativamente distintas de nível de cinza

DETECÇÃO DE BORDAS

- **Derivada de Primeira Ordem / Métodos de Gradiente**
  - Operadores de Roberts
  - Operadores de Sobel
  - Operadores de Prewitt
- **Derivada de Segunda Ordem**
  - Laplaciano
  - Laplaciano de Gaussiana (LoG)
- **Detecção de Bordas Ótima**
  - Canny Edge Detection
- **Transformada de Hough**

LAPLACIANO DA GAUSSIANA (LOG)

- **Para combater a sensibilidade de ruído do filtro Laplaciano, o kernel padrão Laplaciano é comumente combinada com o kernel da gaussiana para produzir um método de filtragem robusto**

- **Estes dois núcleos podem ser aplicados sequencialmente na imagem como duas operações separadas de convolução – primeiro a suavização o kernel gaussiano e depois com o Laplaciano**

- **No entanto, como convolução é associativa, podemos combinar os kernels de convolução do operador de suavização Gaussiana com o operador Laplaciano para produzir um único núcleo: o filtro Laplaciano da Gaussiana (LoG)**

LAPLACIANO DA GAUSSIANA (LOG)

- **A resposta do filtro será zero em áreas de intensidade uniforme da imagem, em contraponto, será diferente de zero em uma área de transição.**
- **Em uma vantagem dada, o operador irá retornar uma resposta positiva sobre o lado mais escuro e negativo no lado mais claro**

Gradient Magnitude

Laplacian
(original kernel)

Laplacian
(rotated+added kernel)

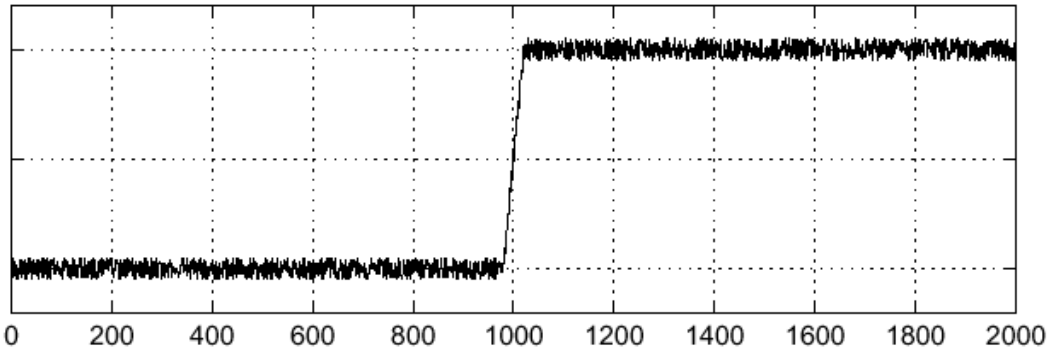
LAPLACIANO DA GAUSSIANA (LOG)

- **Passos**
  - Suavizar a imagem usando filtro Gaussiano
  - Intensificar as bordas usando o operador Laplaciano
  - Passagens pelo Zero denotam a localização de bordas
  - Uso de interpolação linear para determinar a localização dos pixels na borda

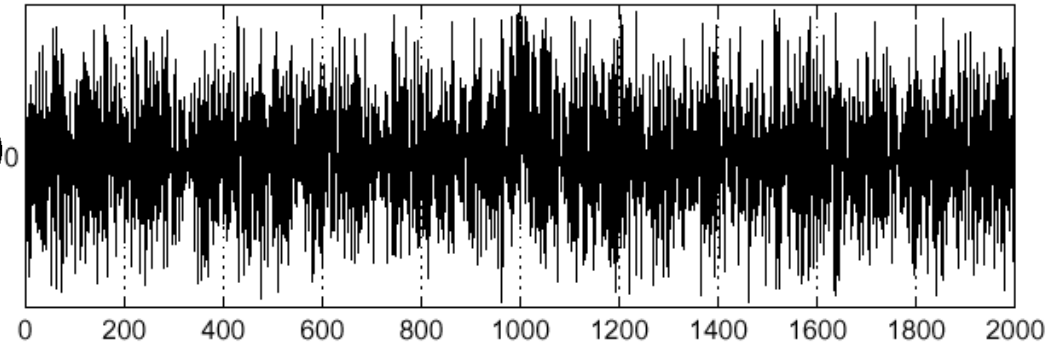
EFEITOS DO RUÍDO

- **Considere uma única linha ou coluna da imagem**
  - Plotando a intensidade como uma função da posisão:

$f(x)$

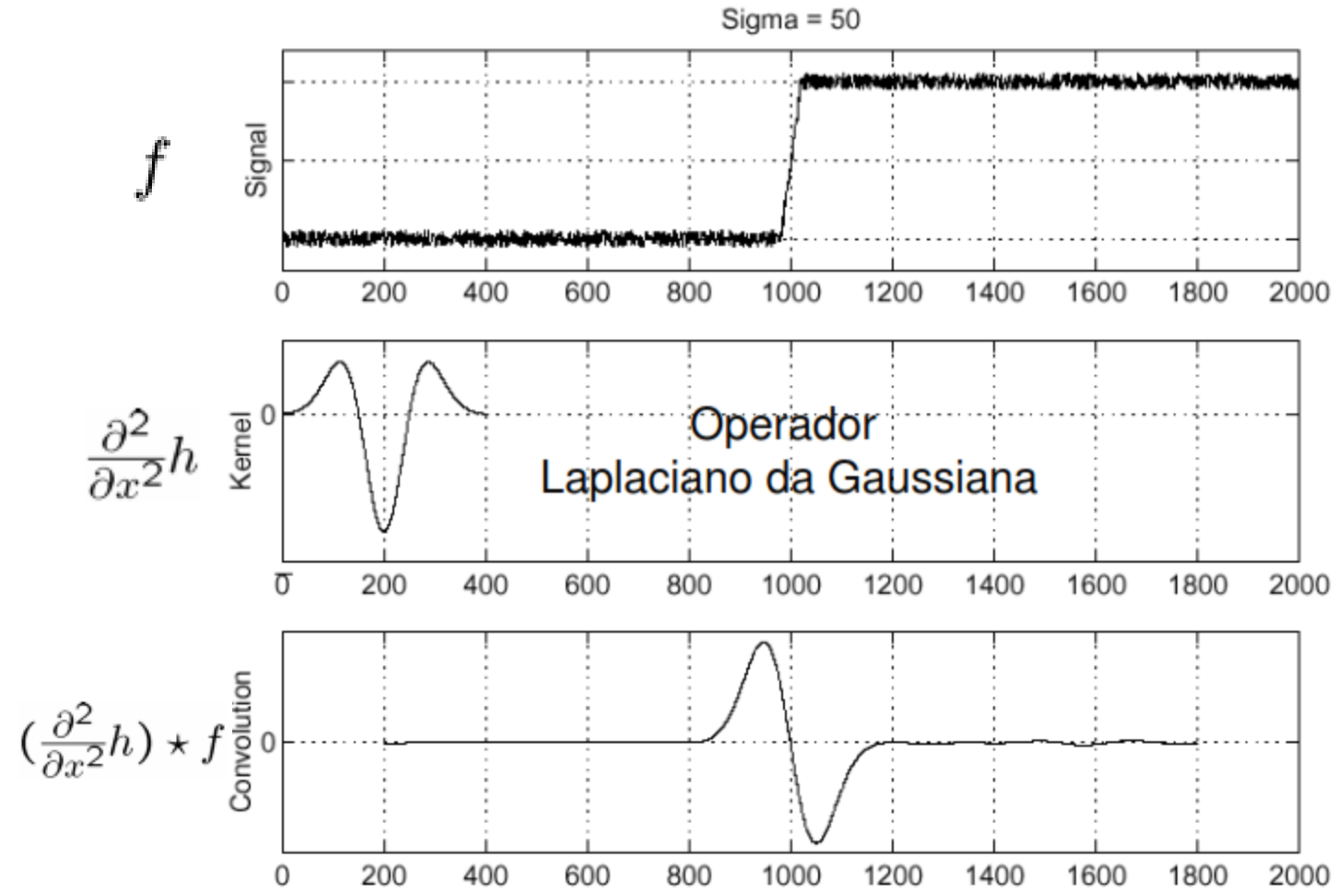


$\frac{d}{dx} f(x)$



Onde está a borda?

LAPLACIANO DA GAUSSIANA (LOG)



LAPLACIANO DA GAUSSIANA (LOG)

- **O resultado desse filtro é fortemente influenciado pelo tamanho da gaussiana utilizada para a fase de nivelamento deste operador**

- **A medida que a suavização é aumentada, então cada vez menosos contornos serão serão encontrados, e aqueles que permanecerem corresponderão às características de escala cada vez maior na imagem**

EFEITO DE σ (VALOR DO KERNEL DA GAUSSIANA)

original

Gaussiana com $\sigma = 1$

Gaussiana com $\sigma = 2$

A escolha de σ depende do comportamento desejado

- σ grande = detecta características espessas
- σ pequeno = detecta características finas

DETECÇÃO DE BORDA ÓTIMO: CANNY

- **John Canny, 1986**
- **Um bom detector deve se preocupar com:**
  - Taxa de erro: O detector deve responder apenas a bordas, encontrando todas
  - Localização: A distância entre os pixels de borda encontrados e as bordas reais deve ser a menor possível
  - Resposta: O detector não deve identificar múltiplos pixels de borda onde só existir uma borda

DETECÇÃO DE BORDA
ÓTIMO: CANNY

Algoritmo:

1. **Suavização da imagem**
   - Objetivo: Eliminar ruído
   - Operadores de gradiente são sensíveis à ruído e esta etapa preliminar é tomada para reduzir o ruído de imagem
   - Quanto maior o largura do kernel, mais de suavização (ou seja, redução de ruído) é alcançada. No entanto, kernels maiores resultam num erro maior na localização das bordas

DETECÇÃO DE BORDA
ÓTIMO: CANNY

2. **Busca por gradientes**
   - Intensifica regiões com maiores derivadas
   - Isto é conseguido tomando o gradiente da imagem com o Operadores Sobel nos sentidos horizontal e vertical e, em seguida, adicionar o magnitude desses componentes como uma medida da "força borda”

$$E(x,y) = |G_x(x,y)| + |G_y(x,y)|$$

DETECÇÃO DE BORDA ÓTIMO: CANNY

3. **Calcular a direção das bordas**

$$\theta = \tan^{-1} \frac{G_y(x, y)}{G_x(x, y)}$$

4. **Aproxima as direções para 0, 45, 90 ou 135 graus:**
   - Uma vez que a direção da borda é conhecida, podemos aproximá-la numa direção que pode ser rastreada em uma imagem digital
   - Considerando-se um pixel qualquer, a direção de uma aresta através deste pixel pode assumir um dos quatro valores possíveis: 0, 90, 45 e 135

DETECÇÃO DE BORDA ÓTIMO: CANNY

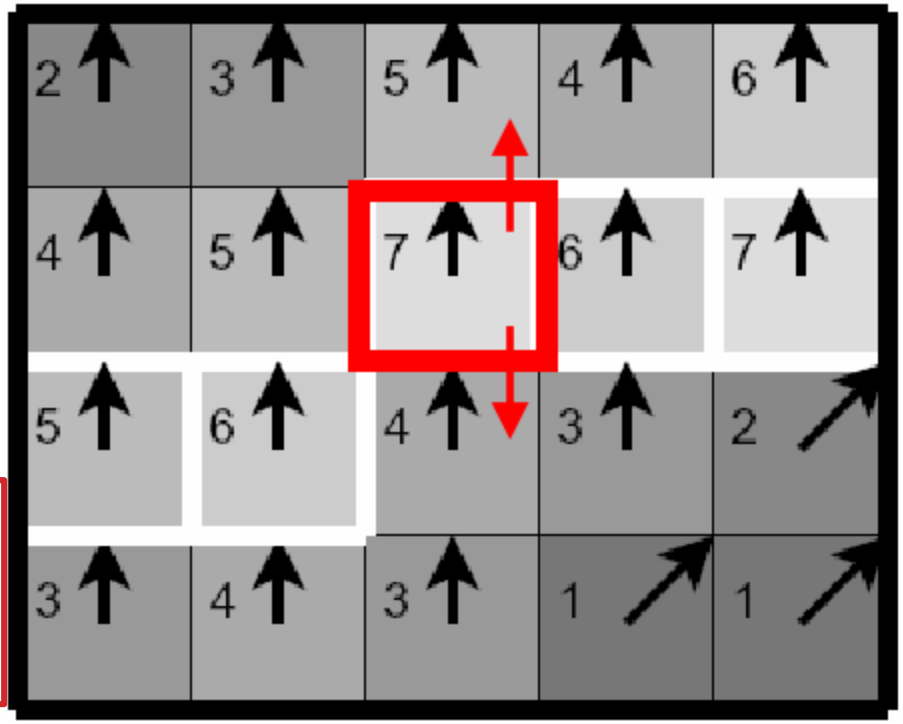
5. ***Non-maximum Suppression***
   - Apenas máximos locais são considerados bordas
   - Funciona como um desenho ao longo da borda, na direção da borda, e suprimindo qualquer valor de pixel (ou seja, defini-se igual a zero) que não seja considerado um borda
   - A comparação é feita na direção do ângulo:
     - Se a direção for de 0°, compara o elemento anterior e posterior da mesma linha. Se o valor do pixel for menor em um dos dois casos, torna-se zero (é suprimido). Faz algo semelhante nas outras direções.
   - O resultado será uma linha fina na imagem de saída

DETECÇÃO DE BORDA
ÓTIMO: CANNY

- **Non-Maximum Suppression:**
  - Exemplo: pixels apontando para o norte são comparados com pixels abaixo e acima

Os que são preservados são os que estão nos quadrados brancos, pois são os maiores (nessa direção); os outros são removidos.

Exemplo: compara o 7 com os valores acima e abaixo; como ele é maior permanece.



DETECÇÃO DE BORDA
ÓTIMO: CANNY

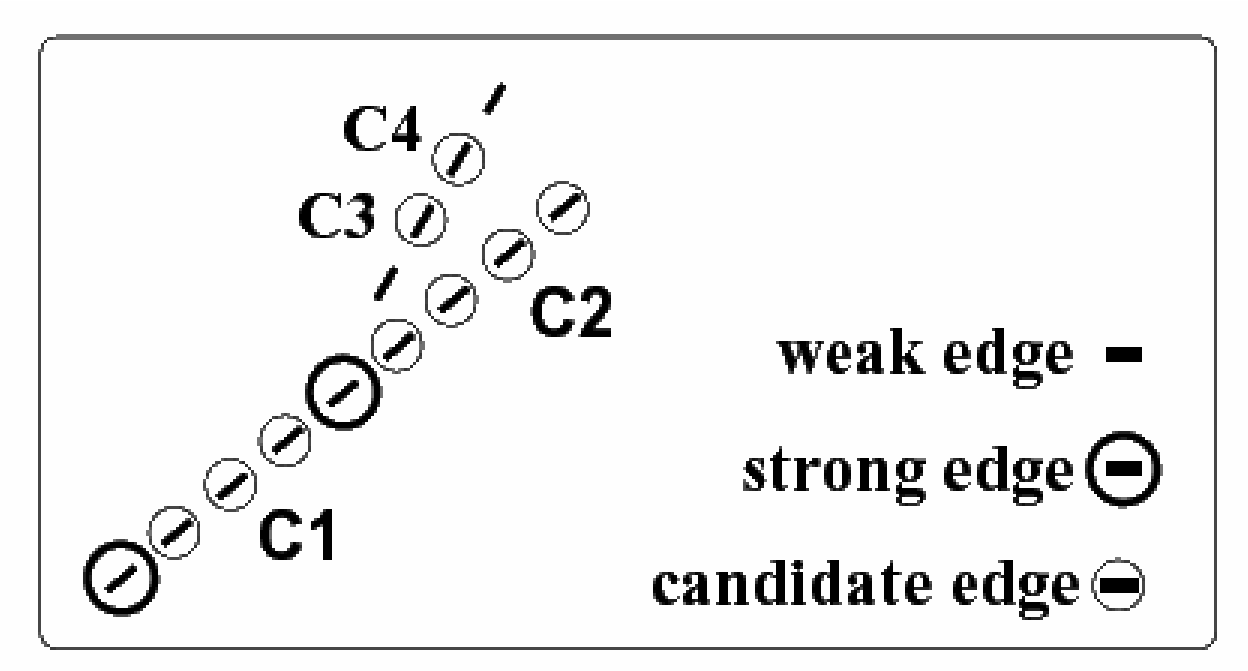
6. **Limiarização por histerese + Edge Tracking**
   - Bordas finais são determinadas, suprimindo as bordas que não estão conectadas a bordas “fortes”
   - O passo final é percorrer os pixels restantes que não foram suprimidos e limiarizar a imagem para identificar os pixels de borda

DETECÇÃO DE BORDA ÓTIMO: CANNY

- **Hysteresis thresholding**
  - Dois limiares: um baixo (T_1) e um alto (T_2)
  - Se $|E(x,y)| < T_1$, o pixel é rejeitado e não é um pixel de borda
  - Se $|E(x,y)| > T_2$, o pixel é aceito como pixel de borda
  - Se $T_1 < |E(x,y)| < T_2$, o pixel é dito candidato:
    - Se um pixel é um candidato, segue o caminho em ambas as direções na direção da aresta enquanto o pixel for menor que T_1
    - Se um pixel candidato inicial está conectado a um pixel forte, $|E(x,y)| > T_2$, ele não é suprimido; do contrário, é

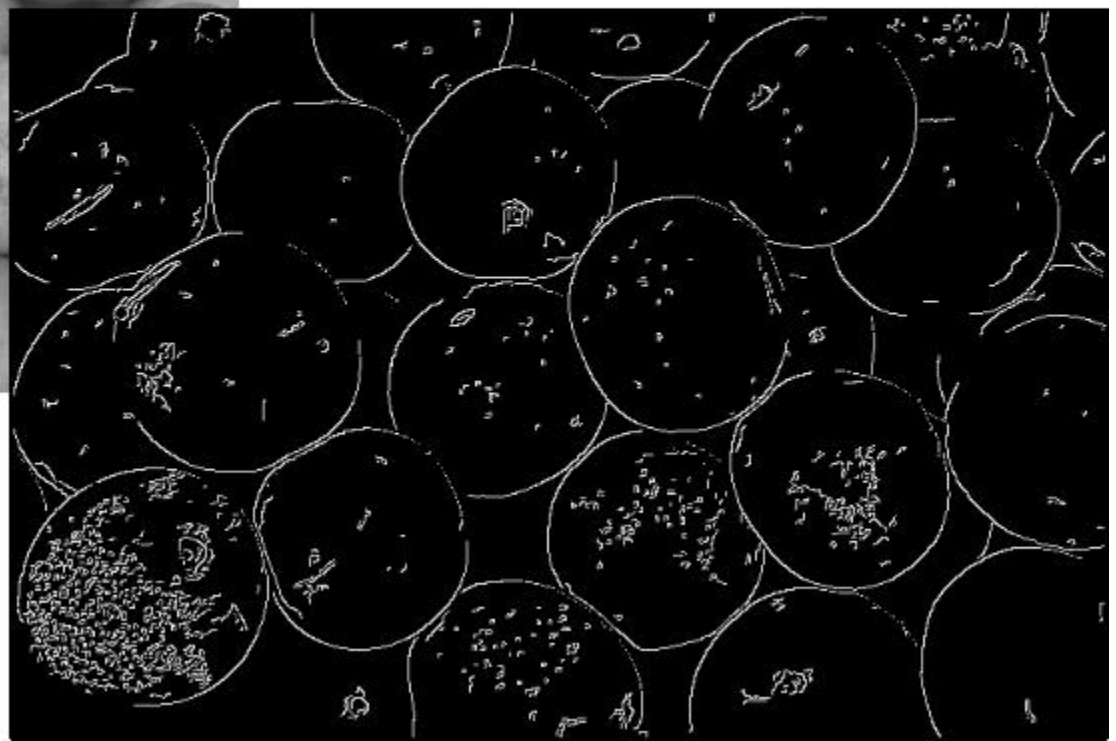
DETECÇÃO DE BORDA ÓTIMO: CANNY

- **Hysteresis thresholding**

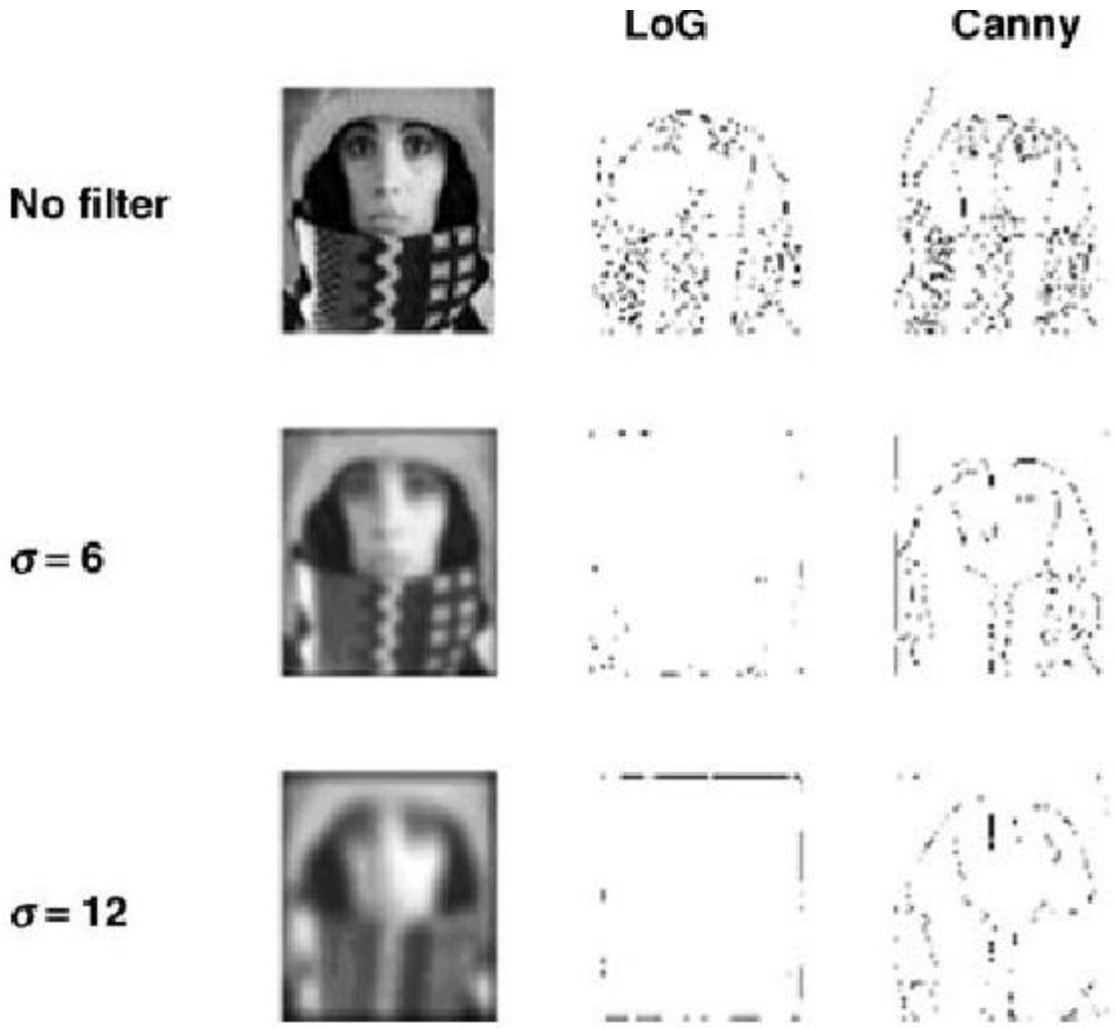


As arestas candidatas C1 e C2 são preservadas na saída, enquanto as arestas C3 e C4 são suprimidas.

DETECÇÃO DE BORDA
ÓTIMO: CANNY



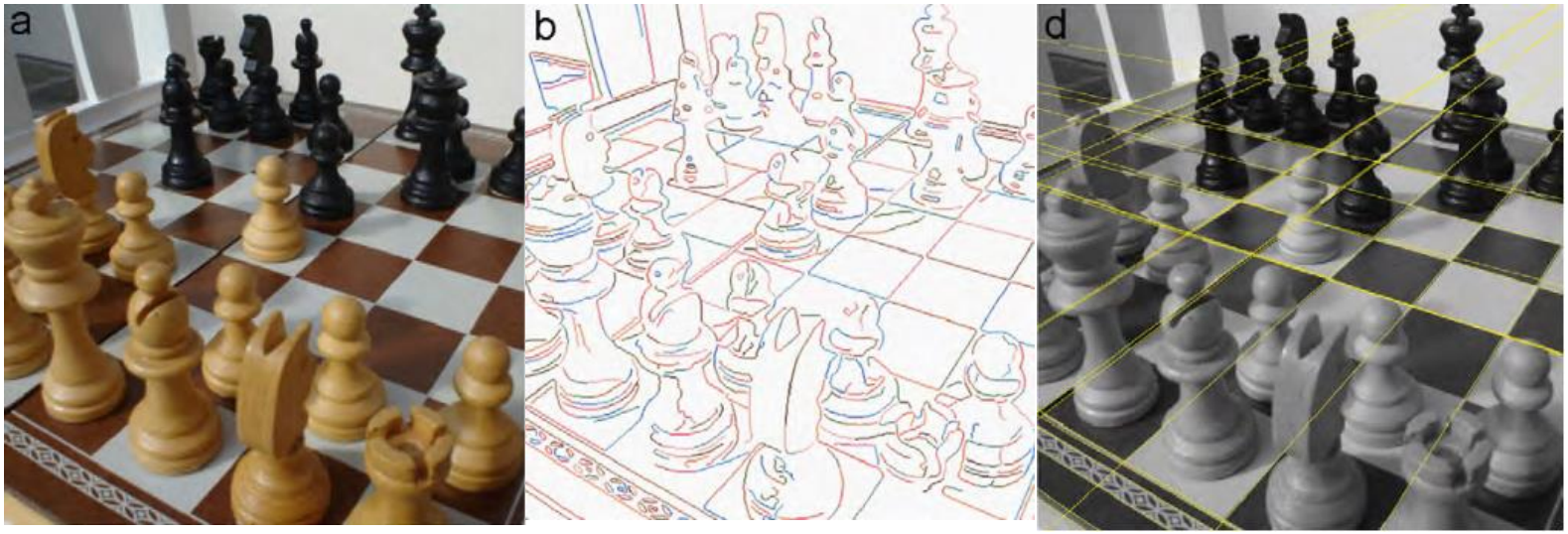
DETECÇÃO DE BORDA ÓTIMO: CANNY



TRANSFORMADA DE HOUGH

DETECÇÃO DE CURVA E LINHAS

- **Transformada de Hough: método elegante para o reconhecimento de objeto**
  - As bordas não necessitam estar ligadas
  - O objeto completo não precisa ser visível
  - Bordas VOTAM para o modelo possível



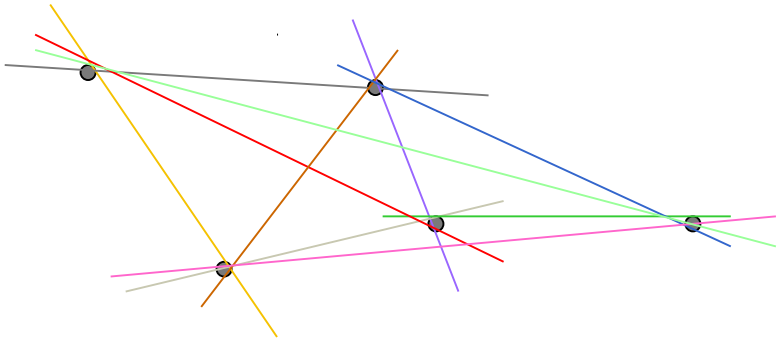
Detecta linhas oclusas parcialmente

TRANSFORMADA DE HOUGH

- **Processamento global para a detecção de linhas retas numa imagem**
- **Nenhum conhecimento é necessário a respeito da posição das linhas**
- **Método robusto que pode ser generalizado a outras formas geométricas**

Suponha que para uma imagem de *n* pontos queiramos encontrar subconjuntos destes pontos que sejam colineares :

Ideia 1: Encontrar todas as linhas determinadas por cada par de pontos e, então, encontrar todos os subconjuntos de pontos constituindo uma linha em particular.



10 retas

Complexidade: $n(n-1)/2$, i.e., $O(n^2)$ para se encontrar todas as linhas

e

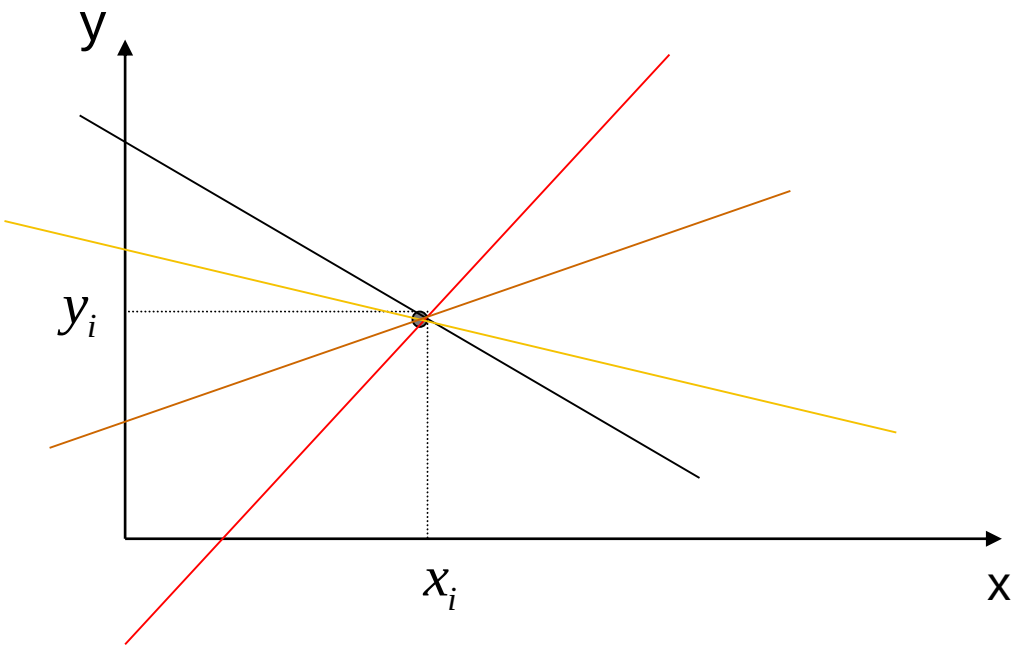
$n[n(n-1)/2]$, i.e., $O(n^3)$ para comparação de cada ponto com todas as linhas

Idéia 2: Transformada de Hough (1962)

- Considere um ponto (x_i, y_i) da imagem e a equação geral da reta:

$$y_i = ax_i + b$$

- Pelo ponto (x_i, y_i) passam infinitas retas (no plano contínuo) com valores de *a* e *b* variáveis.

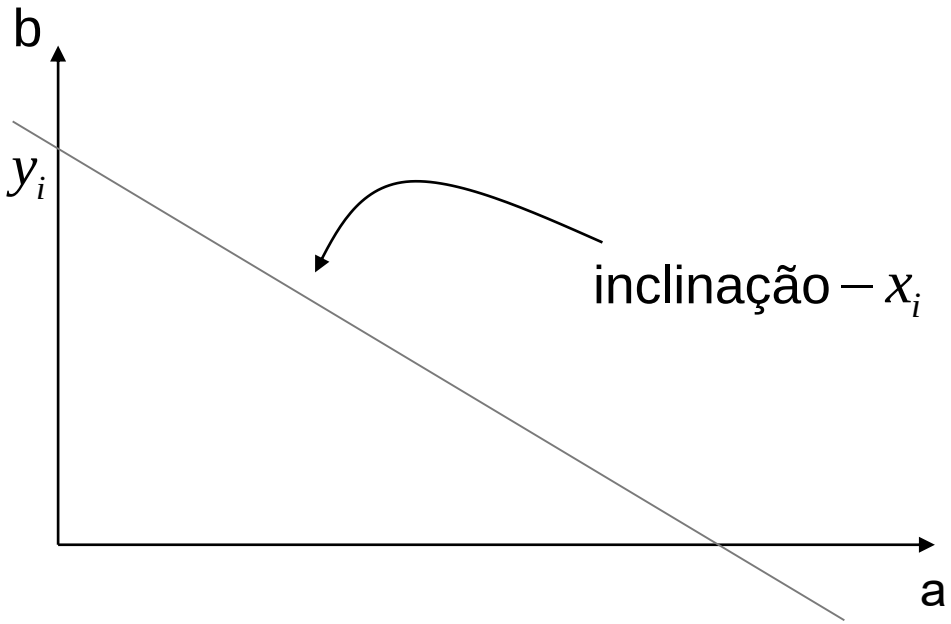


Todas estas retas obedecem à equação $y_i = ax_i + b$, com *a* e *b* **variáveis**.

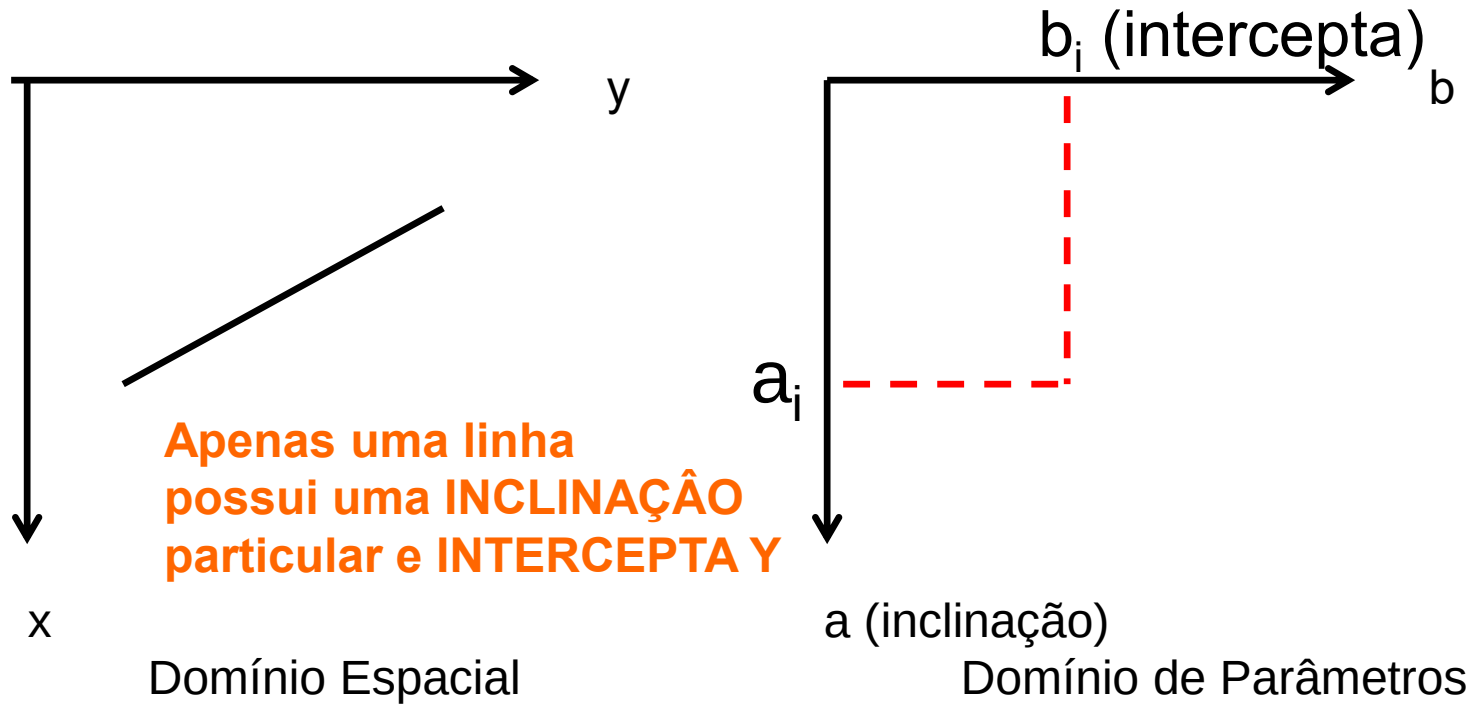
Assim, escrevendo a equação da reta na forma:

$$b = -x_i a + y_i \ ,$$

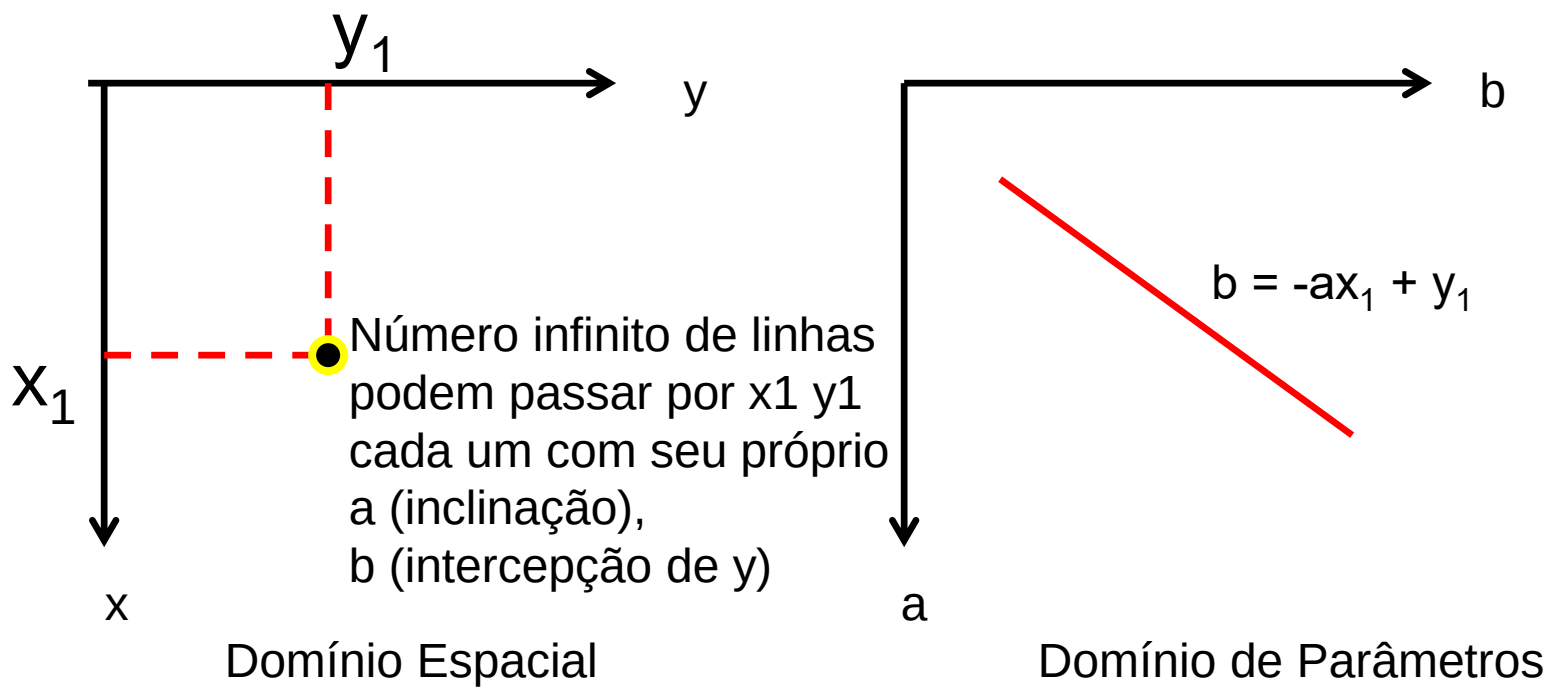
e considerando o plano *ab* (espaço de parâmetros), definimos uma reta de inclinação x_i e ponto de intersecção y_i



A LINHA É UM PONTO NO ESPAÇO DE PARÂMETROS

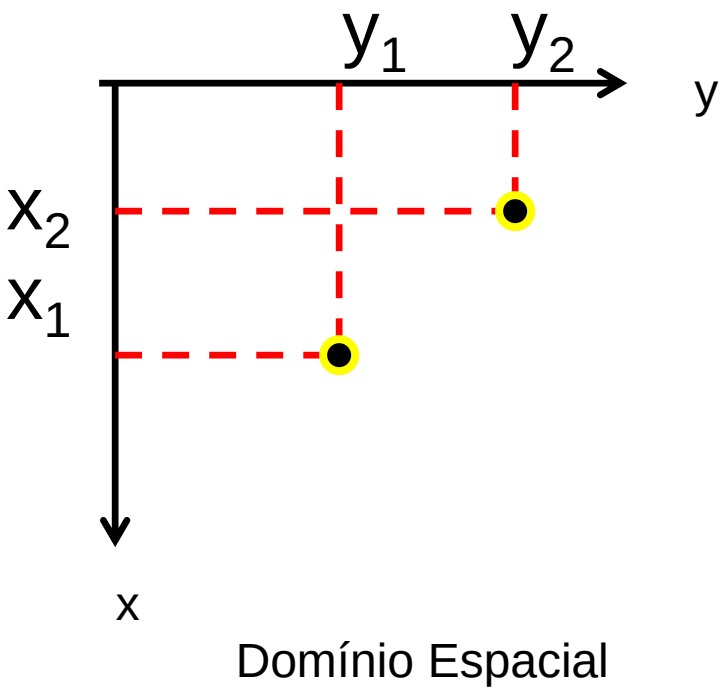


UM PONTO NO DOMÍNIO ESPACIAL É UMA LINHA NO ESPAÇO DE PARÂMETROS



(x_i , y_i) é constante, a e b são as variáveis

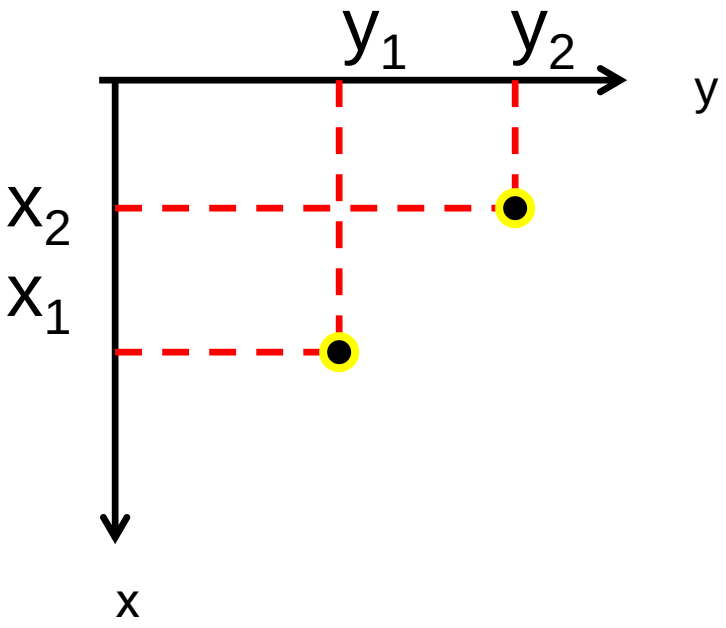
Apenas uma linha pode
Passar pelos dois pontos



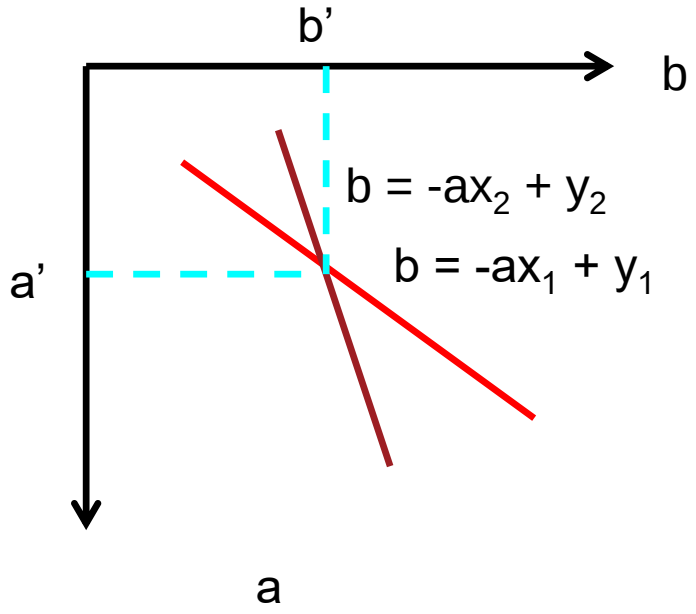
Domínio Espacial

OBJETIVO:
Dado dois pontos no domínio espacial usar a transformada de Hough para encontrar a equação da reta que passa por ambos os pontos

DOIS PONTOS NO DOMÍNIO ESPACIAL SÃO DUAS LINHAS NO ESPAÇO DE PARÂMETROS



Domínio Espacial



Domínio de Parâmetros

(x_i , y_i) é constante, a e b são as variáveis

a' e b' são os parâmetros da linha reta que passa através do dois pontos no domínio espacial

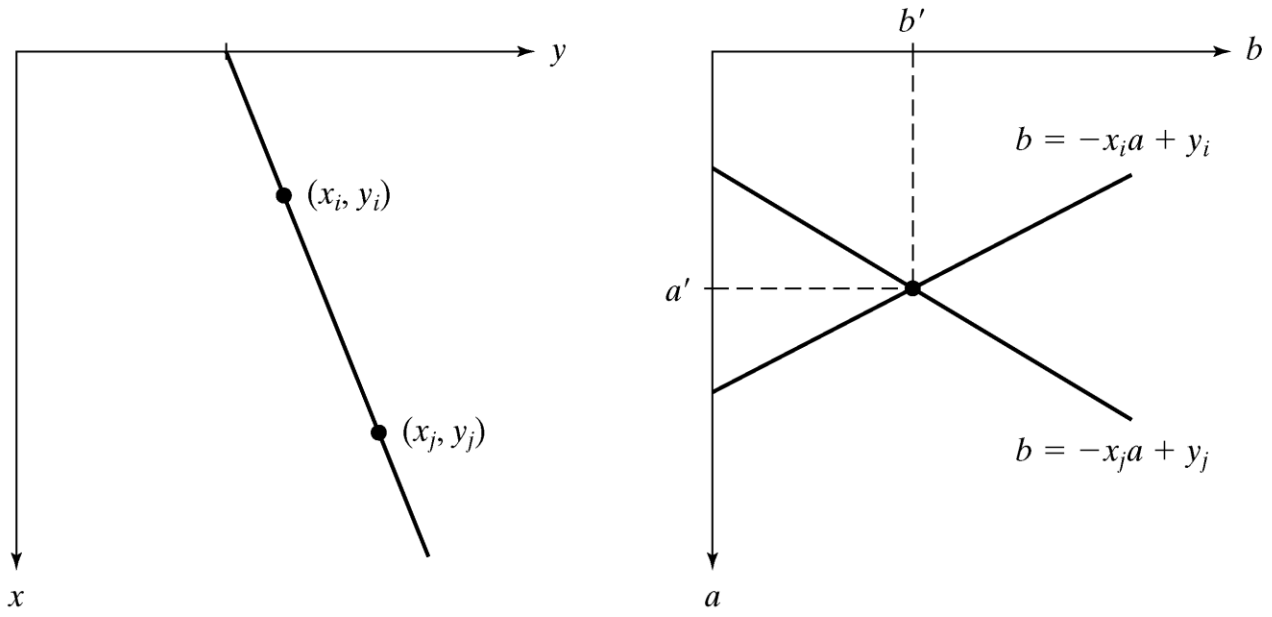
MAPEANDO O DOMÍNIO DE ESPAÇO NO DOMÍNIO DE PARÂMETROS

- Considere um ponto particular (x_i,y_i) no plano x,y
- Um número infinito de linhas pode passar por este ponto
- Todas satisfazem a equação de interceção da inclinação:

  $y_i = a\,x_i + b$ para algum a,b
- Resolver para b: $b = -a\,x_i + y_i$
- O espaço de parâmetros, o plano a-b, possui uma única linha para (x_i,y_i)

ESPAÇO DE PARÂMETROS

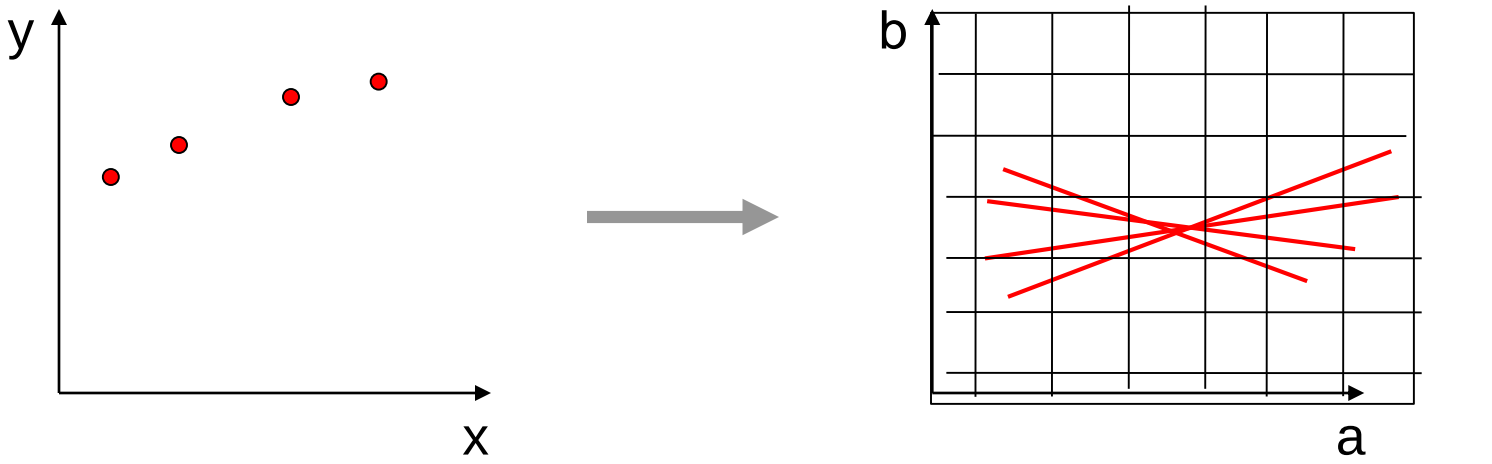
- Considere um segundo ponto (x_j,y_j) no plano x,y
- Também possui uma linha associada no espaço de parâmetros
- Esta linha intecepta a linha associada a (x_i,y_i) em (a', b')
- a' é a inclinação e
- b' é a intersecção da linha contendo ambos (x_i,y_i) e (x_j,y_j) no plano x-y
- Todos os pontos nesta linha possuem linhas no espaço de parâmetros que se cruzam em (a', b')



a b

FIGURE 10.8
(a) *xy*-plane.
(b) Parameter space.

LOCALIZANDO LINHAS EM UMA IMAGEM: ALGORITMO DE HOUGH



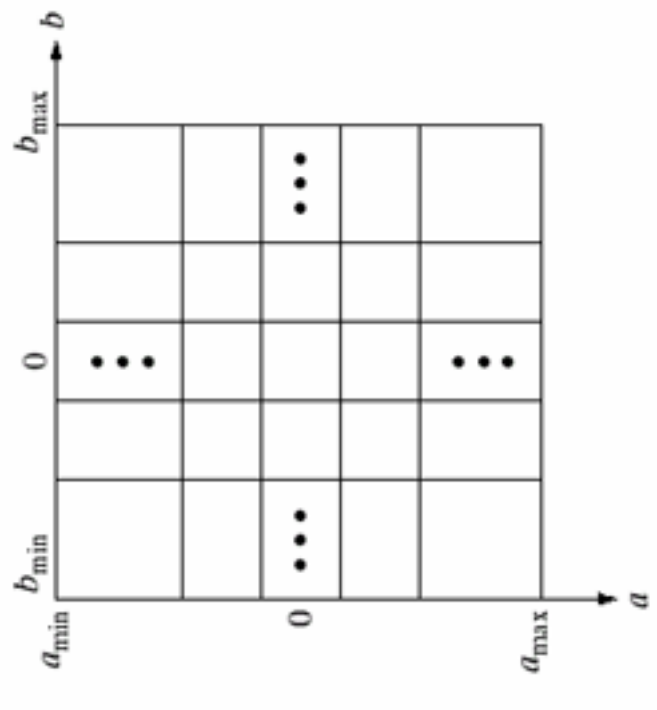
Espaço Imagem Espaço de Parâmetros

Como podemos usar isso para encontrar os parâmetros mais prováveis (a, b) para a linha mais proeminente no espaço imagem?

- Deixe que cada ponto na borda da imagem do espaço ***vote*** para um conjunto de parâmetros possíveis no espaço de Hough
- Acumule *votos* em uma estrutura; parâmetros com a maioria dos votos indicam uma linha no espaço imagem.

Implementação:

- Subdivide-se o espaço de parâmetros em **células acumuladoras**



(a_{min}, a_{max}) e (b_{min}, b_{max}) são os valores mínimos e máximos permitidos para a inclinação e intersecção das retas, respectivamente. Cada célula (i,j), com acumulador A(i,j), guarda o número de ocorrências de a_i, b_j .

LOCALIZANDO LINHAS EM UMA IMAGEM: ALGORITMO DE HOUGH

Algoritmo:

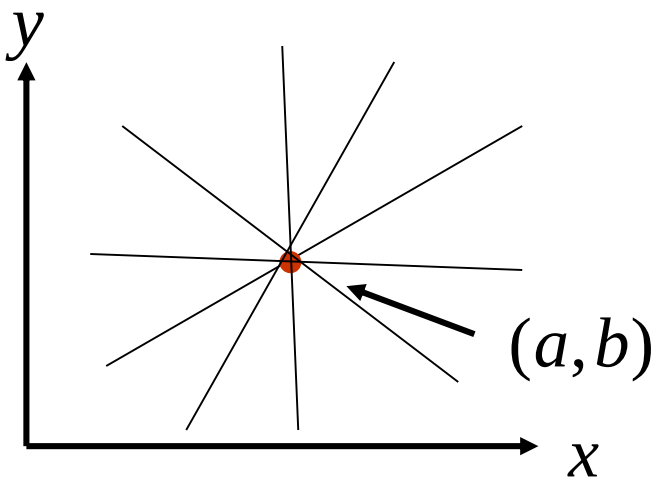
- Discretize o espaço de Parâmetros (a,b)
- Crie um array acumulador $A(a,b)$
- Faça $A(a,b)=0 \quad \forall a,b$
- Para cada ponto (x_i, y_i) no plano xy, considera-se o parâmetro a igual aos valores possíveis de a (na subdivisão do espaço ab) e calcula-se b na equação:

$$b = -x_i a + y_i$$

- Se um valor de a_i resulta em b_j, então

$$A(a,b) = A(a,b) + 1$$

- Encontre o máximo local em $A(a,b)$

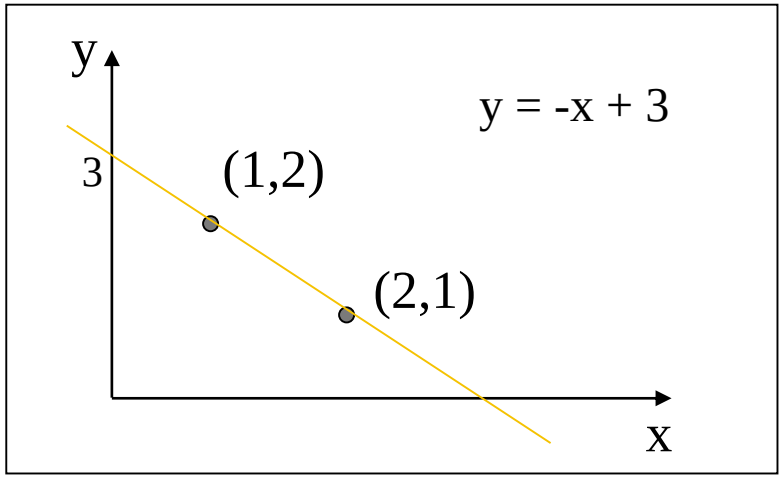


Espaço de Parâmetros

$A(a,b)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Exemplo



$$b = -x_k a + y_k$$

$(x_k, y_k) = (1,2)$

a = -2 → b = 4

a = -1 → b = 3

a = 0 → b = 2

a = 1 → b = 1

a = 2 → b = 0

$(x_k, y_k) = (2,1)$

a = -2 → b = 5

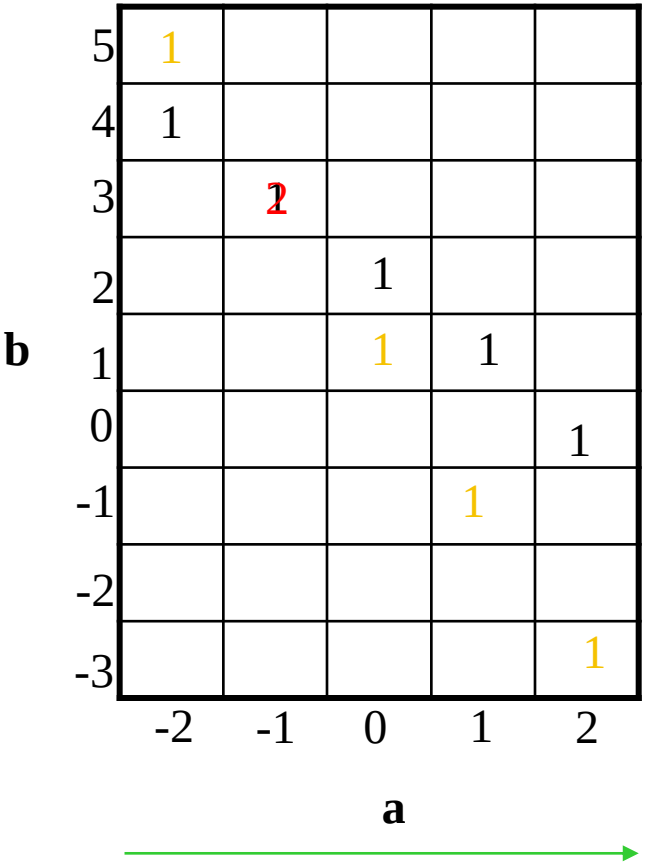
a = -1 → b = 3

a = 0 → b = 1

a = 1 → b = -1

a = 2 → b = -3

espaço de parâmetros



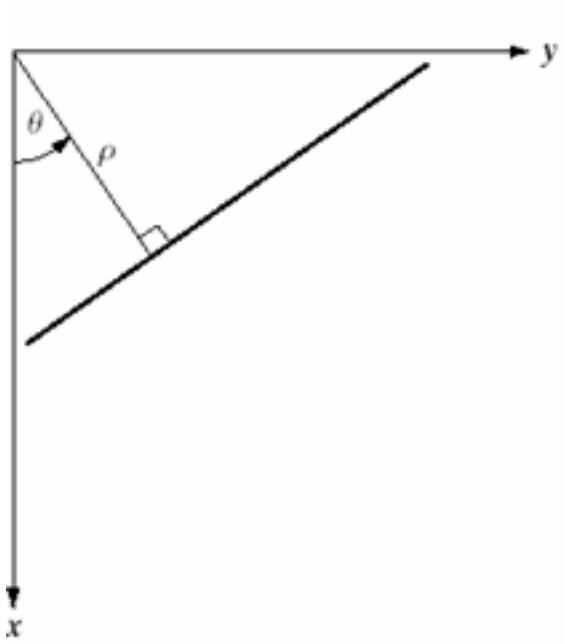
Problema:

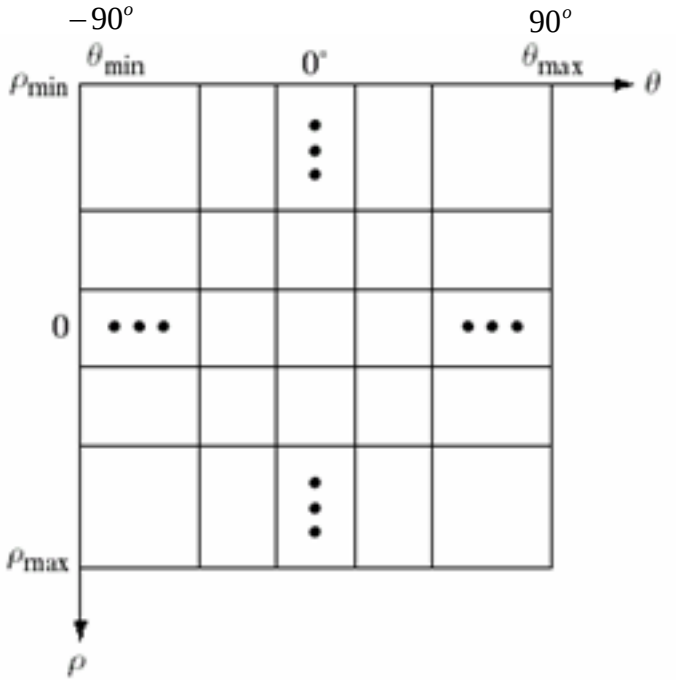
Os valores de *a* e *b* tendem para infinito à medida que as retas se tornam verticais.

Alternativa: Considerar a representação normal da reta (em coordenadas polares):

$$x\cos\theta + ysen\theta = \rho$$

ρ é a distância perpendicular da reta à origem do plano *xy e* θ é o ângulo desta reta perpendicular, em relação ao eixo x.





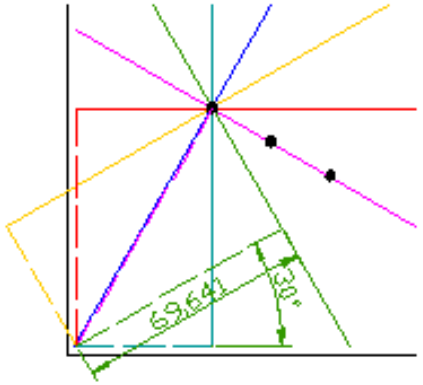
HOUGH – QUESTÕES DE IMPLEMENTAÇÃO

- Usando coordenadas polares para equação da reta
  - X cos ϴ + u sen ϴ = ρ
  - Reta horizontal: ϴ = 0 e ρ = interseção positiva de x
  - Reta Vertical:
    - ϴ = 90 e ρ = interseção positiva y
    - ϴ = -90 e ρ = interseção negativa y
  - Cada curva senoidal no plano de parâmetros representa a família de retas que passam por um ponto (x_k,y_k) no plano xy
  - O ponto de interseção (ρ, ϴ) corresponde à reta que passa por (x_i,y_i) e (x_j,y_j)

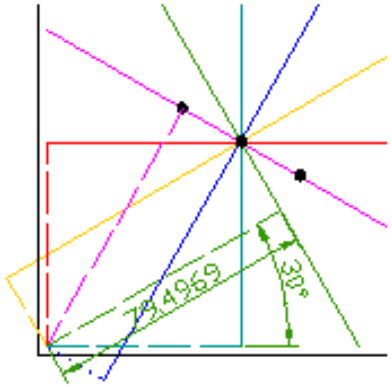
HOUGH – QUESTÕES DE IMPLEMENTAÇÃO

- O “espaço de Hough” é uma matriz do tipo
  - (ρ_{min}, ρ_{max}) e $(\theta_{min}, \theta_{max})$
  - $-90 < \theta < +90$
  - $-D < \rho < +D$, onde D é a distância máxima entre os cantos opostos de uma imagem
  - O eixo θ varia entre +- 90
  - O eixo ρ varia entre +- sqrt(2) max(image.h,image.w)
  - Os valores da matriz são iniciadas com zero
    - Para cada valor (x_i,y_i) em xy, deixamos que θ seja igual a cada valor da subdivisão permitido no eixo θ e calculamos ρ pela equação polar... Arredondamos o valor de ρ e fazemos A[p,q] = A[p,q] + 1.

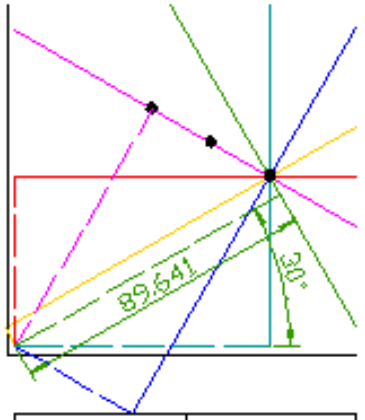
Transformada Hough



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



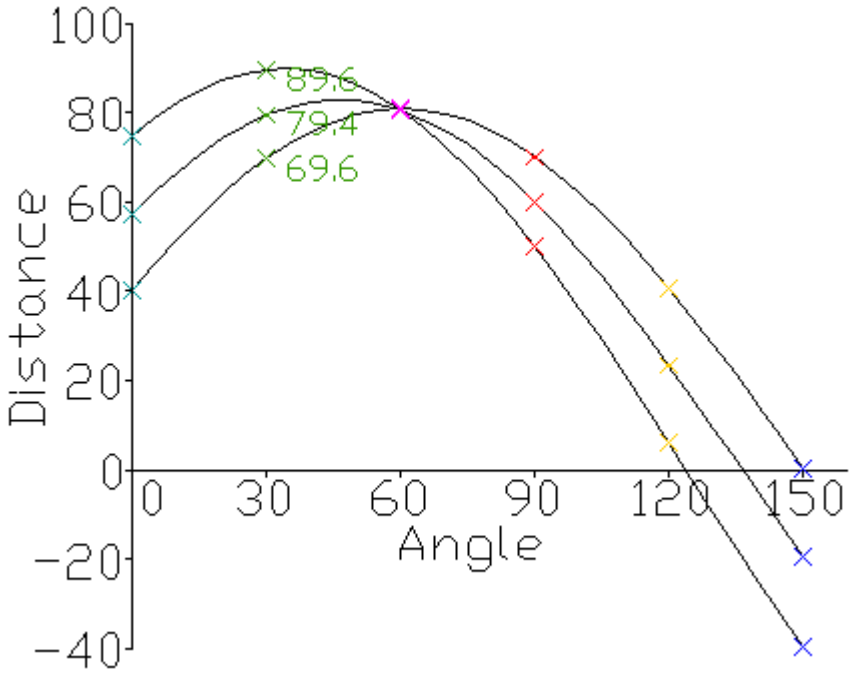
Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5



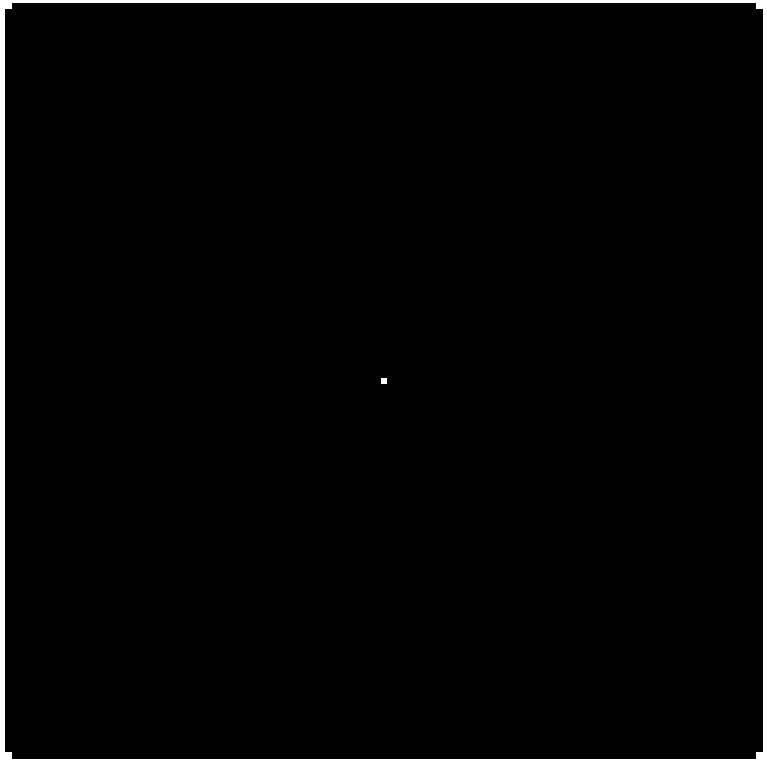
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

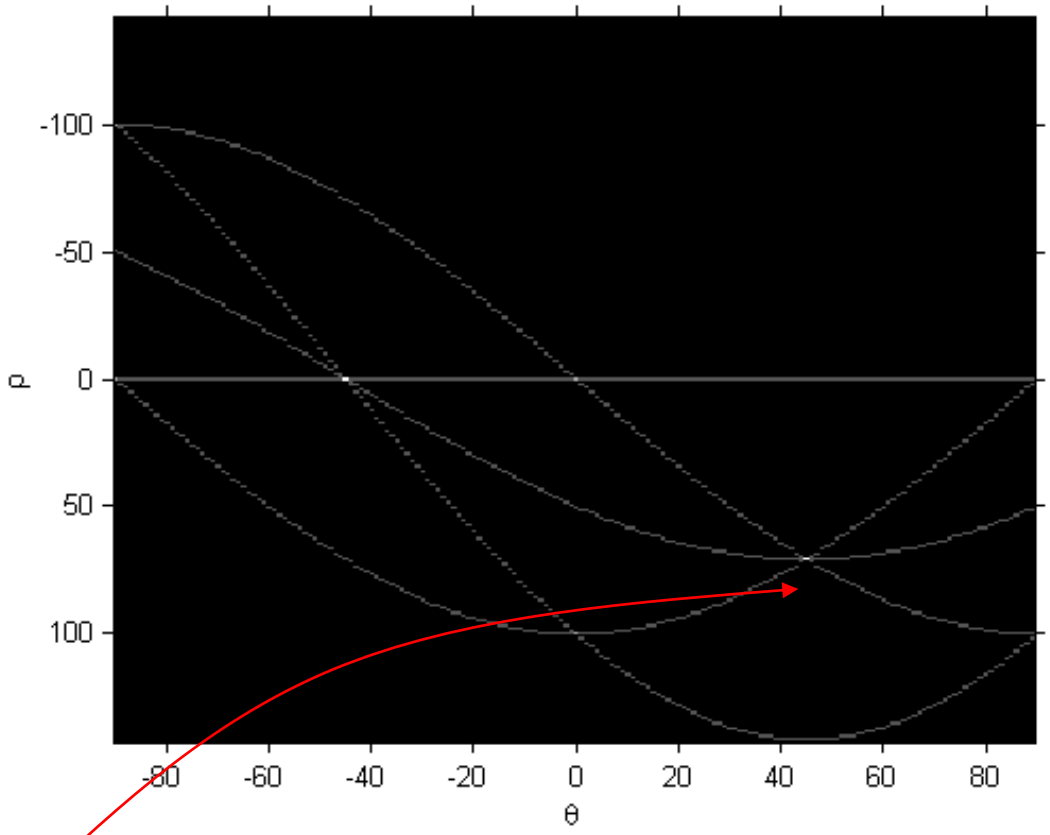
•Para cada ponto, retas são traçadas para certos ângulos (linha cheia)
•A linha pontilhada é perpendicular a cada reta e serve para computar o ângulo e o raio (coordenadas polares).
•Faça isso para os três pontos

Transformada Hough



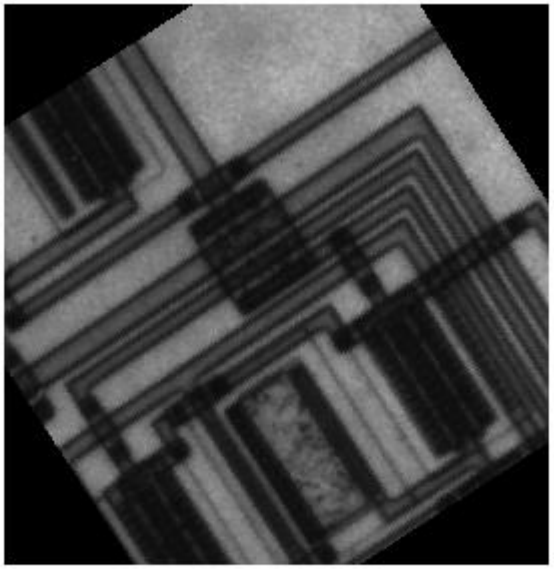
•Este é o espaço de Hough para os três pontos !!!
•Note que o maior acumulador é aquele para a linha rosa e tem valor 3



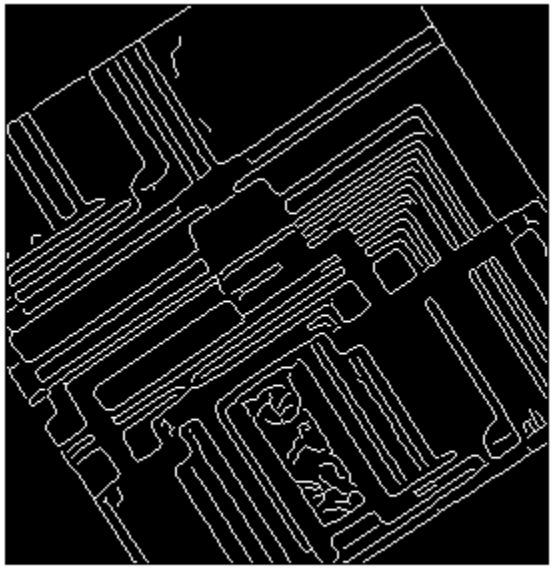


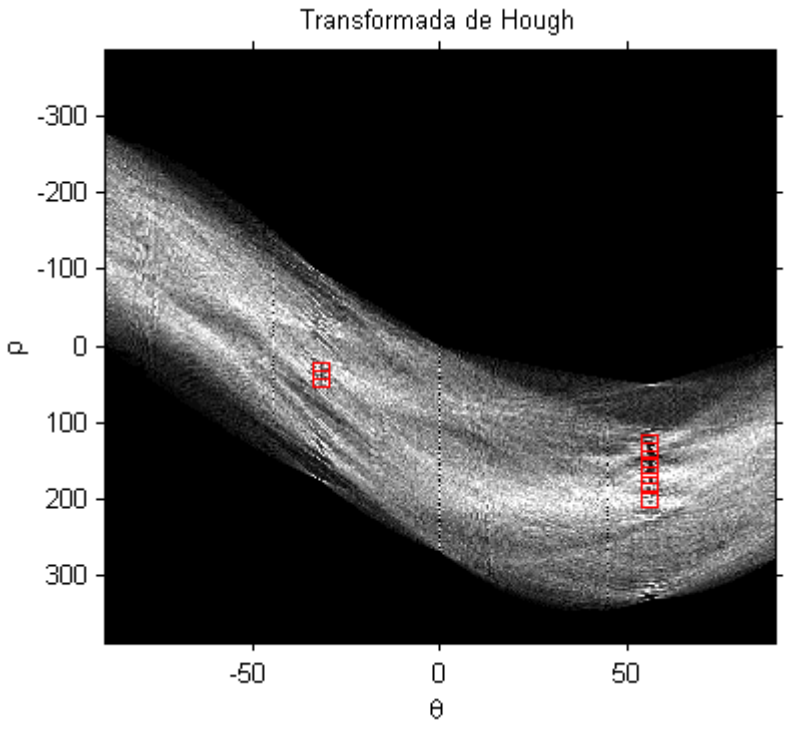
45^o

Original



Contornos

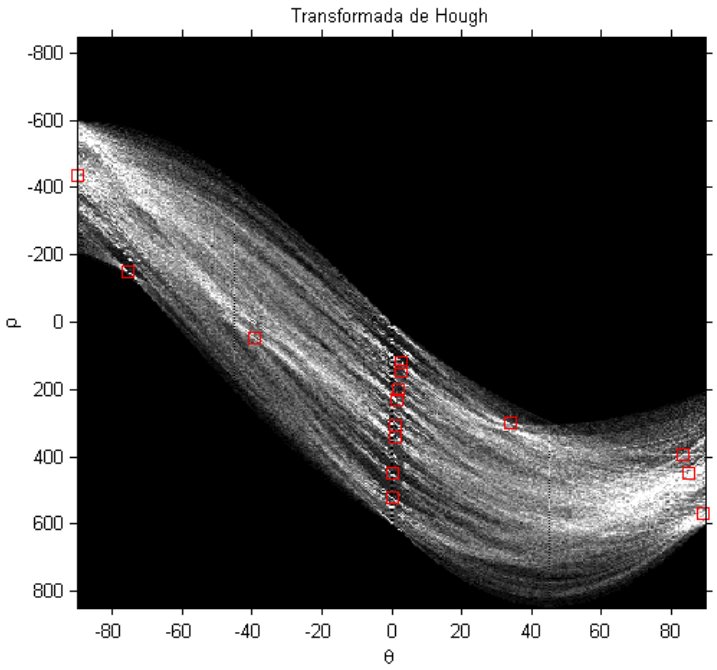




Exemplo 1

Original

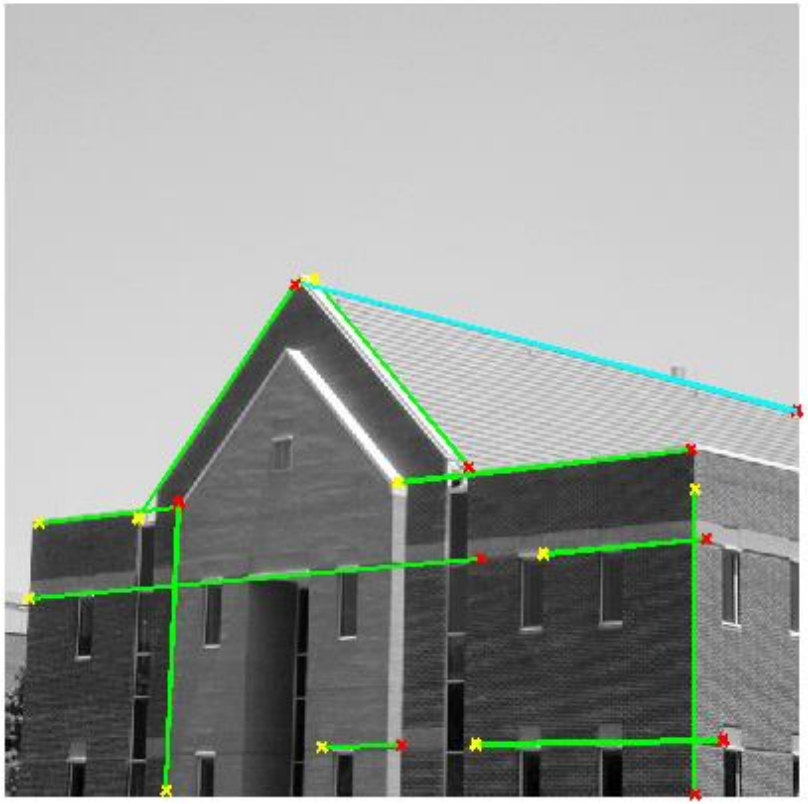
Contornos



Exemplo 2

Contornos

Detecção de linhas



PARÂMETROS PARA CURVAS

Forma	Parâmetros	Equações
Linha	ρ, θ	$x\cos\theta+y\sin\theta=\rho$
Círculo	x_0, y_0, ρ	$(x-x_o)^2+(y-y_0)^2=r^2$
Parábola	x_0, y_0, ρ, θ	$(y-y_0)^2=4\rho(x-x_o)$
Elipse	x_0, y_0, a, b, θ	$(x-x_o)^2/a^2+(y-y_0)^2/b^2=1$

CONCLUSÕES

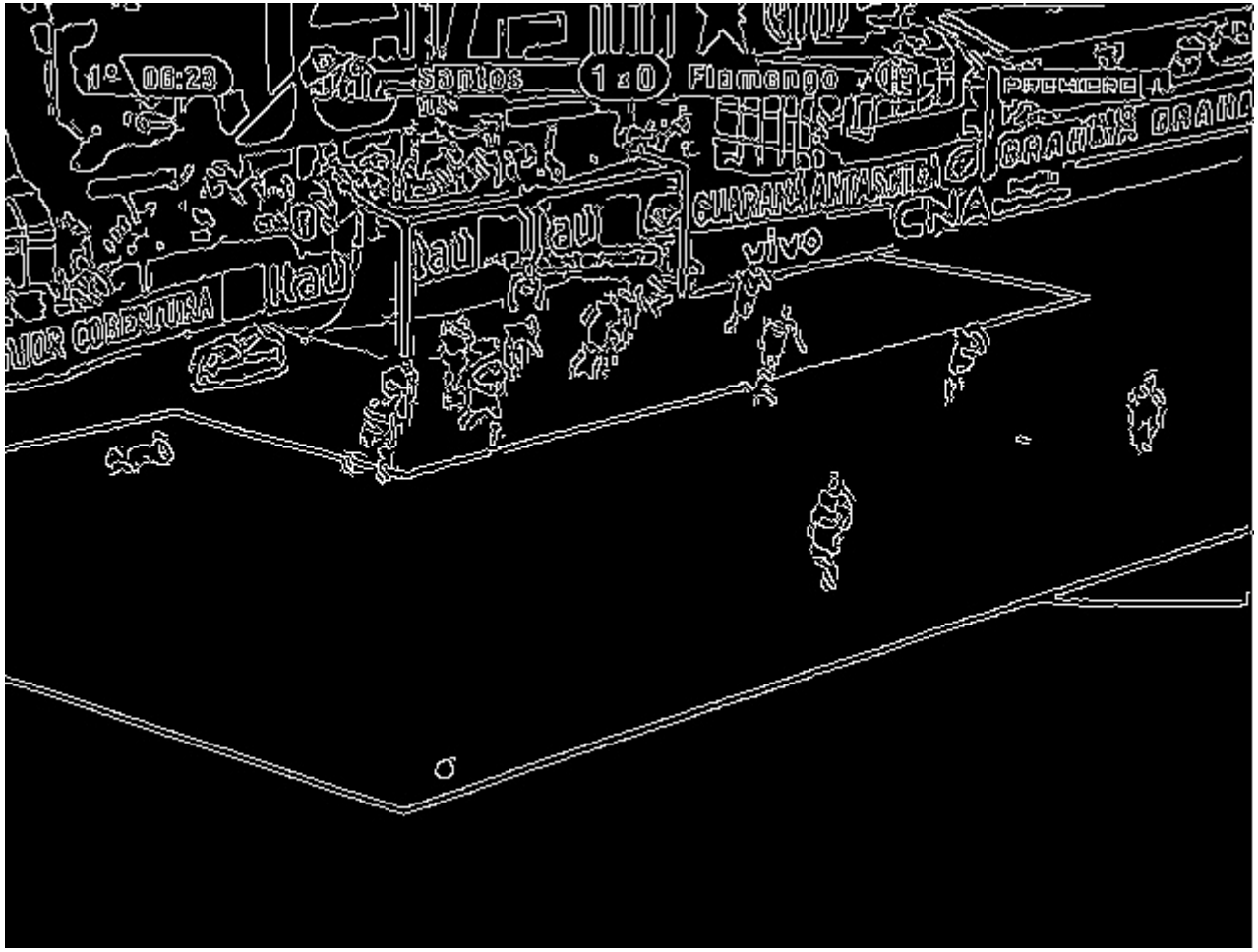
- **Pontos Positivos:**
  - Robusto a outliers: cada ponto vota em separado
  - Bastante eficiente (muito mais rápido do que tentar todos os conjuntos de parâmetros)
  - Fornece vários bons ajustes

- **Pontos Negativos:**
  - Alguma sensibilidade ao ruído
  - Tamanho do acumulador define tolerância ao ruído, precisão e velocidade/memória
    - Pode ser difícil encontrar o melhor tamanho
  - Não é adequado para mais do que alguns parâmetros
    - Tamanho da grade cresce exponencialmente

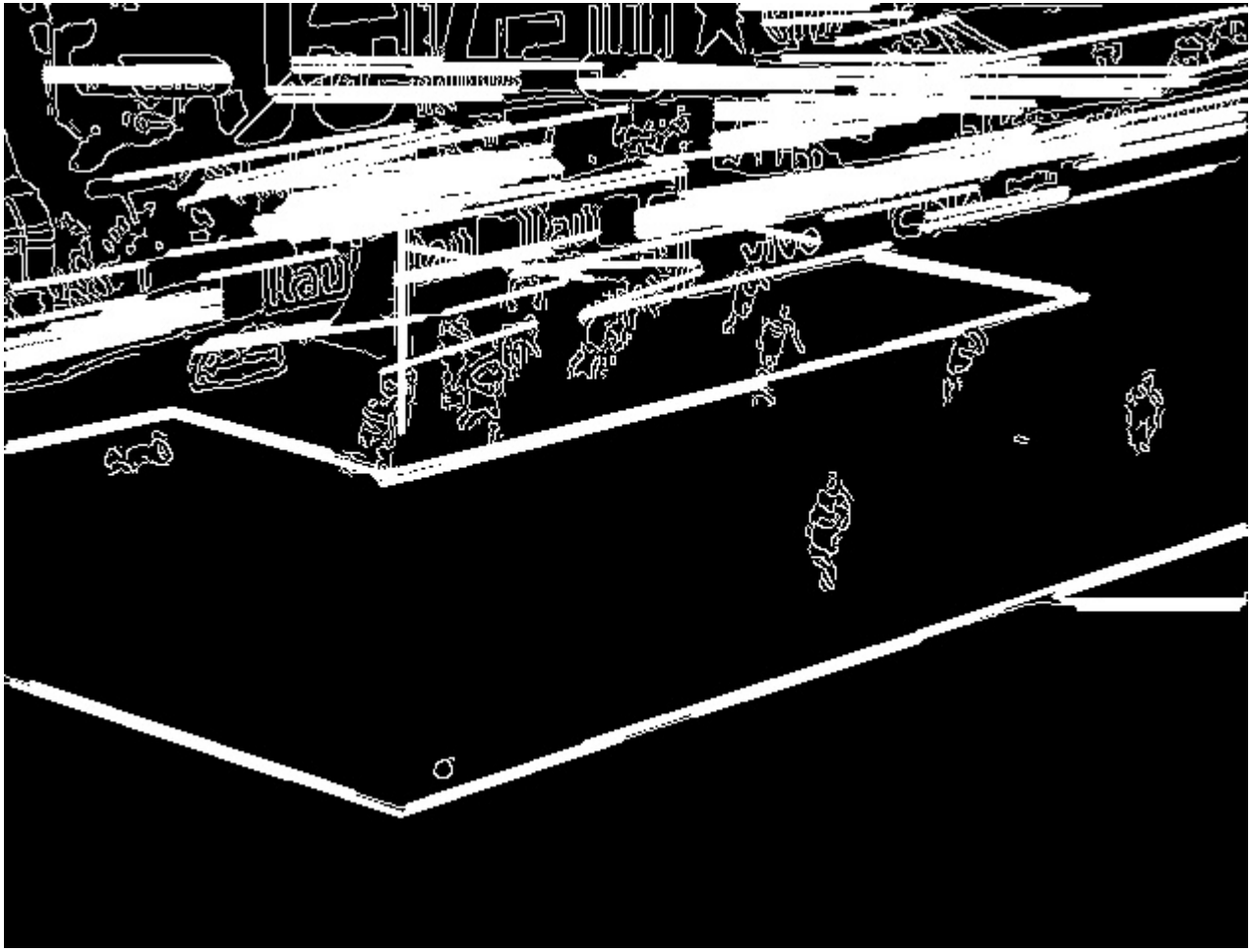
- **Aplicações:**
  - Encontrar linhas (também círculos, elipses, etc.)
  - Reconhecimento de objetos (parâmetros são transformação afim)
  - Reconhecimento da categoria do objeto (parâmetros são posição/escala)

APLICAÇÃO DA TRANSFORMADA DE HOUGH

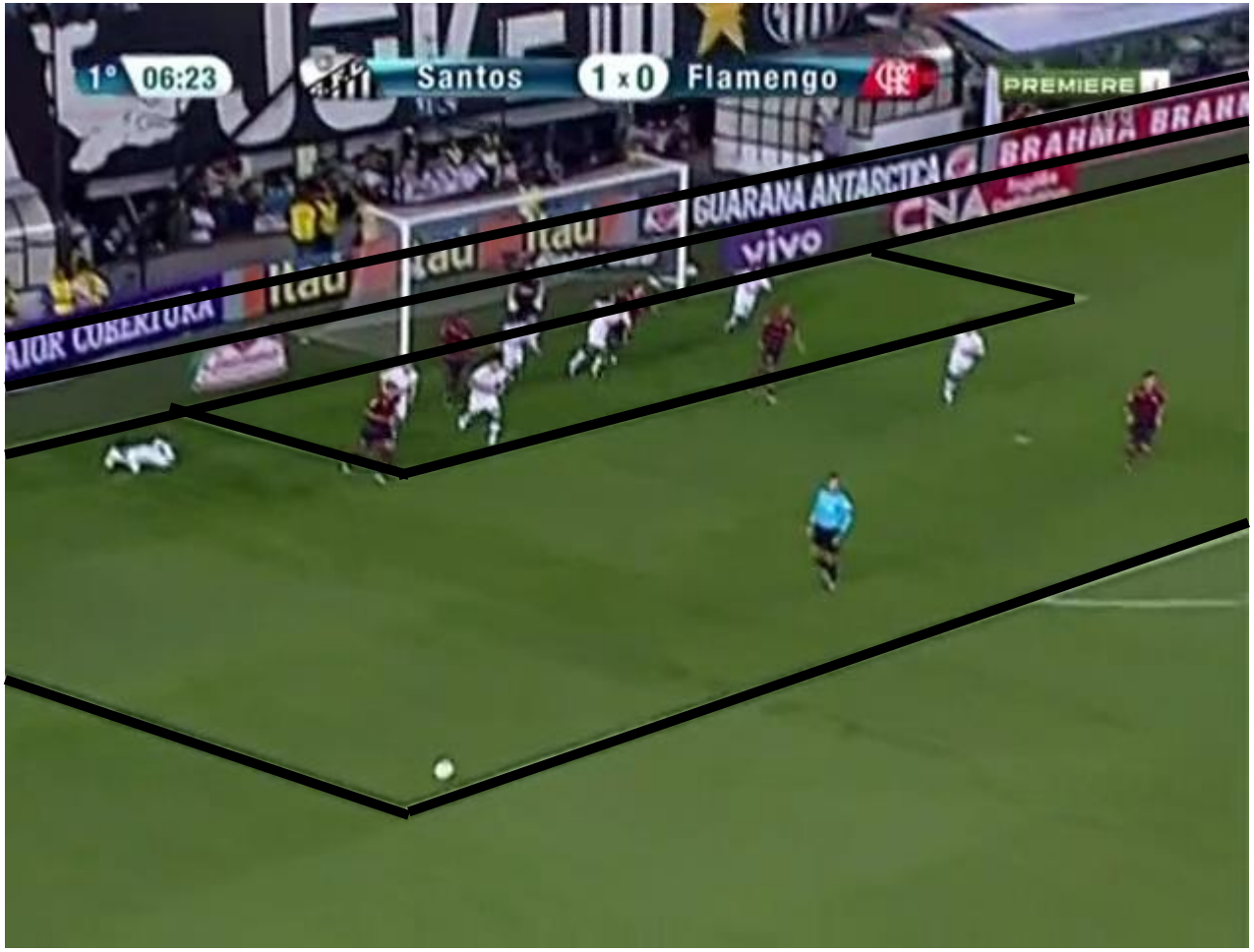
APLICAÇÃO DA TRANSFORMADA DE HOUGH



APLICAÇÃO DA TRANSFORMADA DE HOUGH

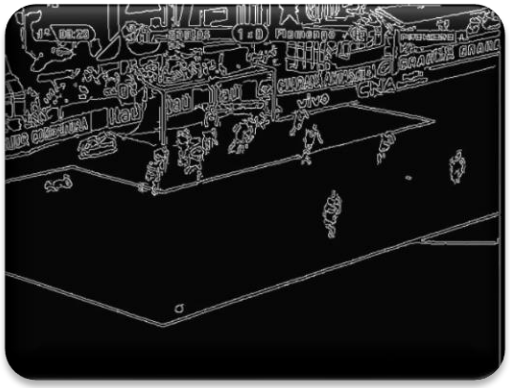


APLICAÇÃO DA TRANSFORMADA DE HOUGH

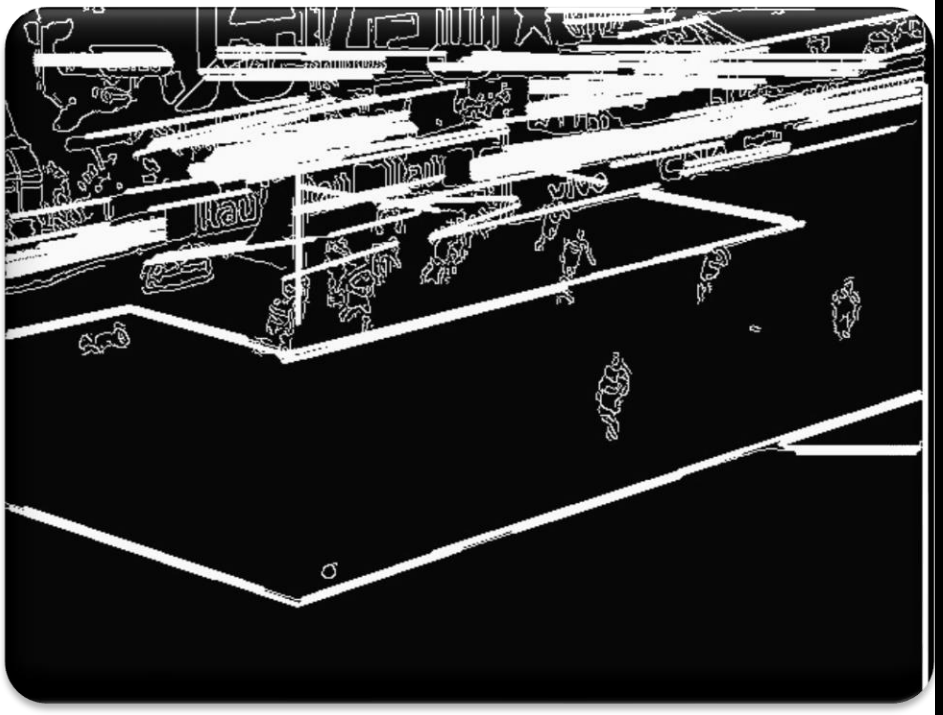


METODOLOGIA

Detecção de Bordas

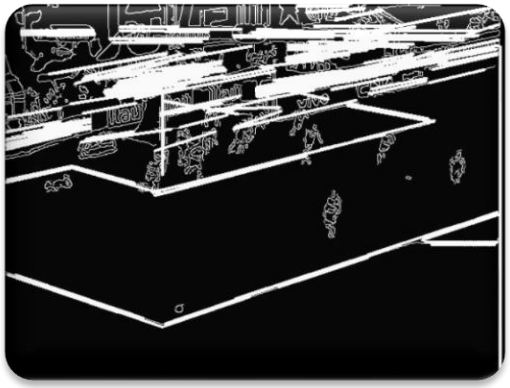


Detecção de Linhas



METODOLOGIA

Detecção de Linhas



Definição das Regiões de Interesse (ROI)

METODOLOGIA

Definição da ROI

Matching

Aplicação da Transformada de Hough

REFERÊNCIAS

- C. Solomon e T. Breckon, Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab, John Wiley & Sons, 2011

- W. Burger e M. J. Burge, Principles of Digital Image Processing: Core Algorithms, Springer, 2009

- R. Gonzalez e R. Woods, Processamento de Imagens Digitais, Edgard Blücher, 3 Ed., 2010

- Slides do Prof. Roger S. Gaborski, Rochester Institute of Technology