VISÃO ROBÓTICA:

- **Captura e Representação de Imagens monocromáticas e coloridas.**
- **Pré-processamento.**
- **Segmentação.**
- **Descritores de Imagem.**
- **Casamento de Padrões**
- **Calibração de Câmera.**

Captura e representação de imagens monocromáticas

- Imagem monocromática = $i(x,y)$ = função contínua de intensidade de luz.
- Câmara com $m \times n$ elementos sensores de largura Δx e altura Δy produz imagem amostrada com $m \times n$ *pixels*.
- Cada elemento sensor captura a intensidade média:

$$I_a(k,j) = \left[\int_0^{\Delta x} \int_0^{\Delta y} (i[(k-1)\Delta x + x, (j-1)\Delta y + y])dx.dy \right] / (\Delta x. \Delta y) \geq 0$$

- A intensidade de cada *pixel* é quantizada em 2^b níveis de cinza com uma precisão de b bits, produzindo a imagem digital $I(k,j)$.

Imagem Digital

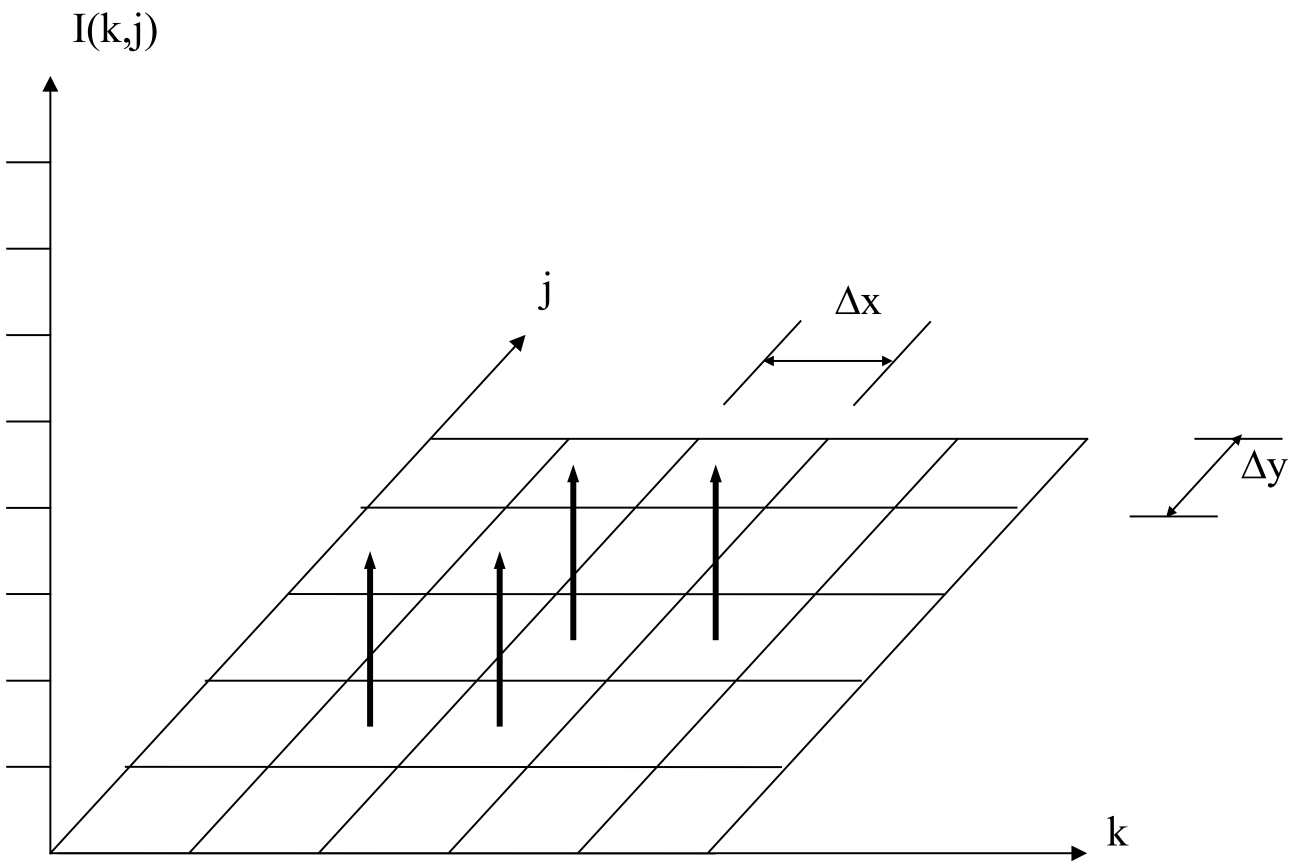


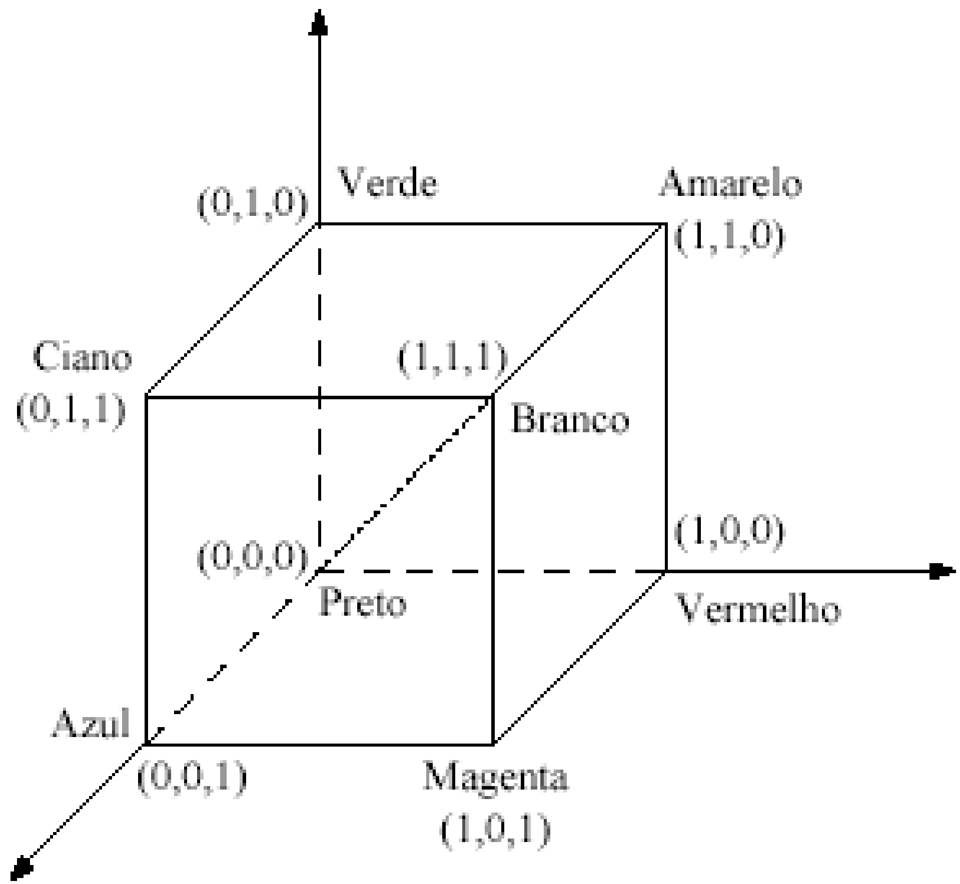
Imagem Colorida

- *Pixel* = informação de intensidade + informação de cor.
- São necessários três valores (1 de intensidade + 2 de cor).
- Modelos de cor = subespaços tridimensionais.

Modelo RGB

- Baseado na teoria do tri-estímulo: o olho humano percebe cor através da ativação de três pigmentos visuais nos cones da retina, (Vermelho, Verde, Azul).
- Sistema de coordenadas cartesianas.
- Cor = combinação ponderada das três cores primárias: Red (vermelho), Green (verde) e Blue (azul).

Subespaço de cores – Cubo RGB



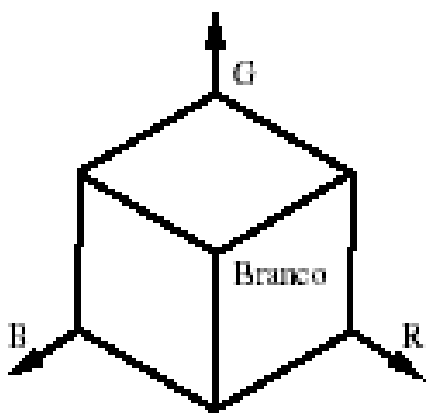
Modelo HSL

- Baseado em parâmetros intuitivos de cor.
- HSL = (Hue, Saturation, Lightness):

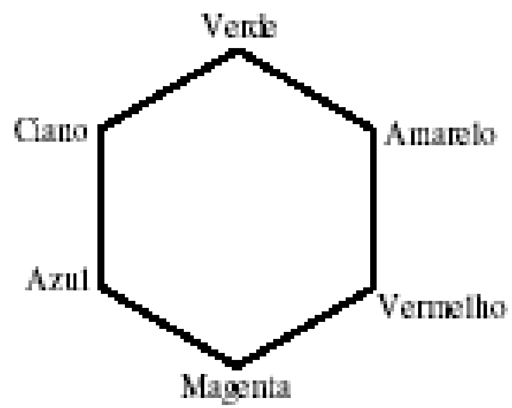
  – H = Matiz da cor.
  – S = Saturação da cor (mistura com branco).
  – L = Luminância (Brilho)

Modelo HSL

- Derivado do cubo RGB.
- Subespaço = Pirâmide hexagonal dupla, (sistema de coordenadas cilíndricas).



Modelo RGB
(a)

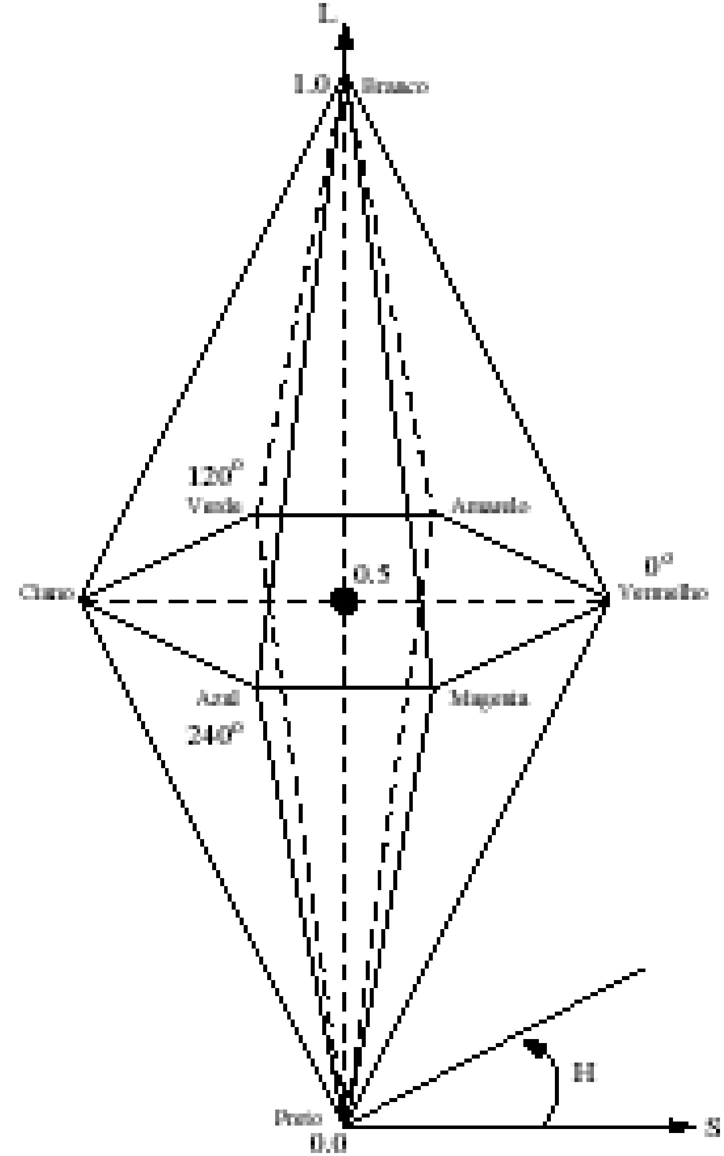


Hexágono de cores
(b)

Modelo HSL

- Matiz de Cor H: especifica cor como um ângulo em torno do eixo vertical da pirâmide.
- Saturação S: medida ao longo do eixo radial, especifica a pureza relativa (diluição com a cor branca).
- Luminância L: medida ao longo do eixo da pirâmide.

Subespaço de cores – Pirâmide HSL



Pré-processamento – Detecção de Eixos e Bordas

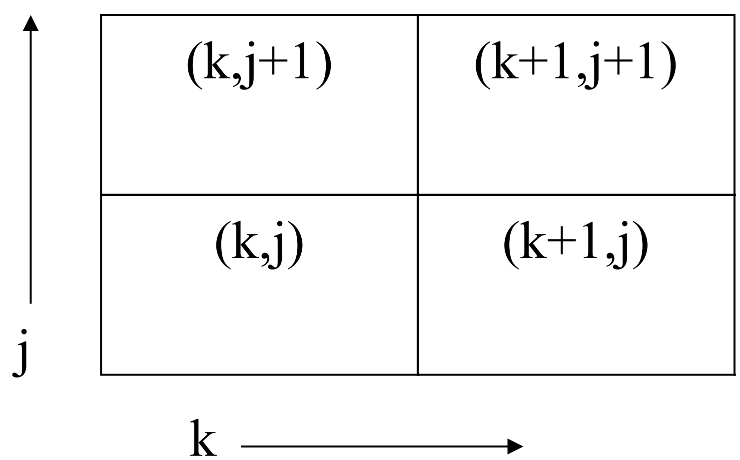
- i(x,y) varia continuamente sobre a face de um objeto.
- Nos limites (bordas ou eixos) i(x,y) sofre descontinuidade.

$\forall \nabla i(x,y) = [\partial i(x,y)/\partial x, \partial i(x,y)/\partial y] \rightarrow \infty$

- Sua magnitude pode ser utilizada para detectar bordas e eixos de objetos.

Aproximação discreta de $\nabla i(x,y)$

- Para imagem digitam $I(k,j)$: aproximação discreta $\nabla I(k,j)$ por diferenças finitas.



Aproximação discreta de $\nabla i(x,y)$

- $\nabla I_x(k,j) = ([I(k+1,j) - I(k,j)] + [I(k+1,j+1) - I(k,j+1)])/2\Delta x$

- $\nabla I_y(k,j) = ([I(k,j+1) - I(k,j)] + [I(k+1,j+1) - I(k+1,j)])/2\Delta y$

$\Rightarrow \quad \|\nabla I(k,j)\| = [\nabla I_x^2(k,j) + \nabla I_y^2(k,j)]^{1/2}$

- Se $\Delta x = \Delta y$:

- Magnitude: $\|\nabla I(k,j)\| = ([I(k+1,j+1) - I(k,j)]^2 + [I(k,j+1) - I(k+1,j)]^2)/2(\Delta x)^2$

- Direção: $\phi(k,j) = \mathrm{atan2}[\nabla I_x(k,j), -\nabla I_y(k,j)]$

Algoritmo para detecção de eixos

- Um eixo ou uma borda pode ser detectado no *pixel* (k,j) se:

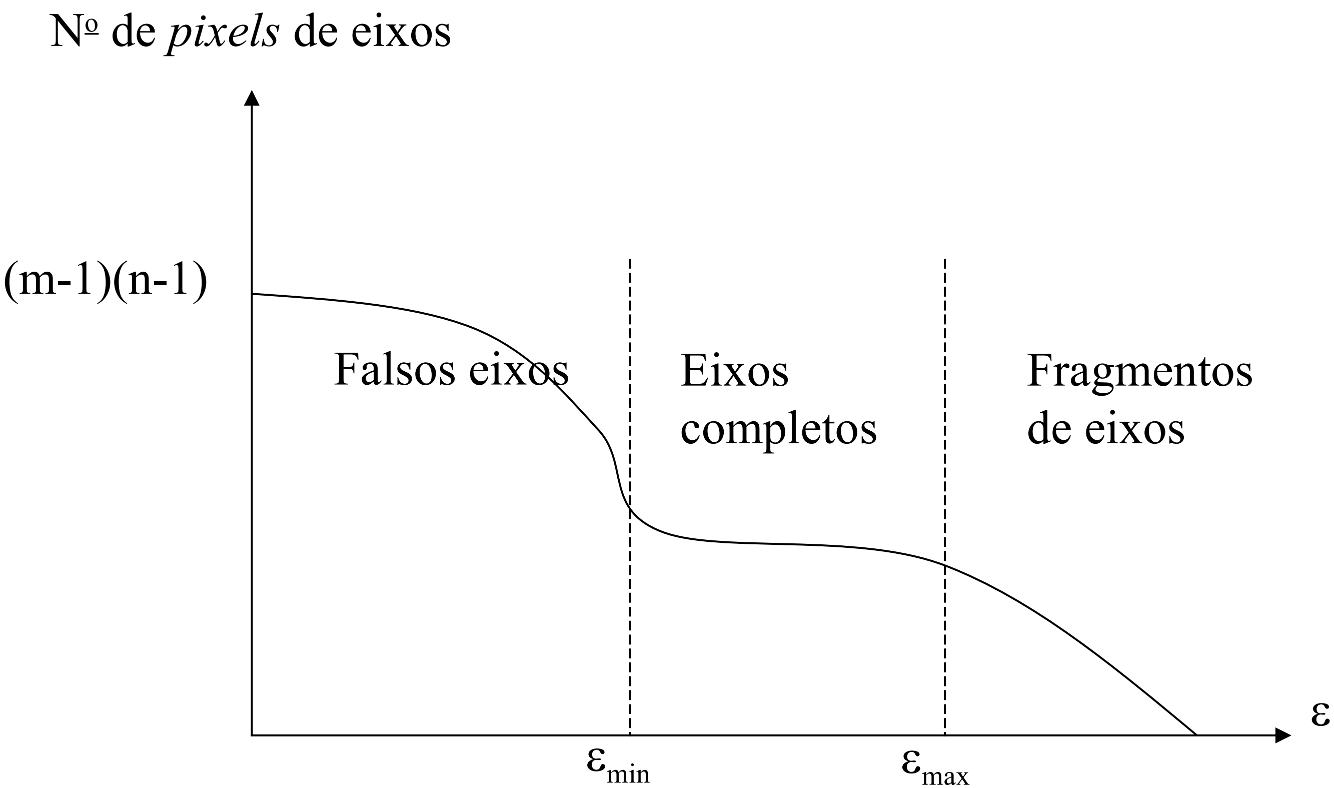
$$\|\nabla I(k,j)\| \geq \varepsilon \geq 0$$

$\forall\ \varepsilon$ = limiar de detecção de eixo.

Algoritmo para detecção de eixos

1. Inicializar $k = 1$, $j = 1$, $\varepsilon > 0$.
2. Computar $\|\nabla I(k,j)\|$.
3. Se $\|\nabla I(k,j)\| > \varepsilon$ fazer $L(k,j) = 1$, caso contrário $L(k,j) = 0$.
4. Faça $j = j+1$. Se $j < n$ ir para o passo 2.
5. Fazer $L(k,n) = 0$.
6. Faça $j = 1$, $k = k+1$. Se $k < m$ ir para o passo 2.
7. Para j variando de 1 a n, fazer $L(m,j) = 0$.

Especificação do limiar ε



Pré-Processamento - Filtragem

- A captura, quantização e digitalização de uma imagem introduz ruídos de origem variada.
- Resultado – Imagem ruidosa:
  - Bordas borradas.
  - Apêndices e entradas nas bordas.
  - Presença de pixels espúrios (pequenas áreas no fundo da imagem e buracos nas faces de objetos).
- Operadores morfológicos podem filtrar a imagem usando técnicas locais, suavizando contornos e eliminando parte do ruído.

Pré-Processamento - Filtragem

- Função de *pixel* sobre imagem binária mxn I(k,j):

$$P(k,j) = [\Sigma_u \Sigma_v(I(k+u,j+v)] - I(k,j)$$

- onde $u,v = -1, 0, 1$, $1 < k < m$, $1 < j < n$..
- Retorna o Nº de *pixels* = 1 vizinhos ao *pixel* (k,j). $\Rightarrow 0 \leq p(k,j) \leq 8$.
- Definida só para *pixels* interiores de I(k,j).

Operador de Erosão (*Shrink*)

shrink(i) I(k,j) = I(k,j) AND 1(i – 1 – [8 – p(k,j)])

onde $0 \leq i \leq 8$,   1(.) = degrau unitário.

- Torna *pixel* (k,j) = 0 se pelo menos i vizinhos = 0, caso contrário, mantém o seu valor.

Operador de Erosão (*Shrink*)

- Shrink é monotônico, № de *pixels* = 1 na imagem resultante ≤ № de *pixels* =1 da imagem original.
- Aplicado iterativamente converge em um número finito de iterações.
- Usualmente, i > 4.
- Shrink(8) remove todos os *pixels* isolados do fundo da imagem em uma única iteração.
- Shrink(7) remove do fundo da imagem todas as regiões de até 2 *pixels*, bem como apêndices verticais ou horizontais com um *pixel* de largura e de comprimento arbitrário.

Operador de Dilatação (*Swell*)

Swell(i) I(k,j) = I(k,j) OR 1(p(k,j)-i)

onde $0 \leq i \leq 8$,   1(.) = degrau unitário.

- Torna *pixel* (k,j) = 1 se pelo menos i vizinhos = 1, caso contrário, mantém o seu valor.
- Dual do operador Shrink.
- Usado para remover pequenos buracos.

Abertura e Fechamento

- Erosão e dilatação podem ser combinados para produzir operadores mais complexos.

- Operador de Abertura: erosão seguida de dilatação. Elimina pequenos ruídos e abre lacunas em regiões de fraca conexão.

- Operador de Fechamento: dilatação seguida de erosão. Restaura conexões fracas entre objetos.

Segmentação

- Processo pelo qual itens em uma imagem são separados do fundo e uns dos outros.
- Particionar a imagem em regiões conexas e "homogêneas"
- A cada região deve ser atribuído um rótulo único.

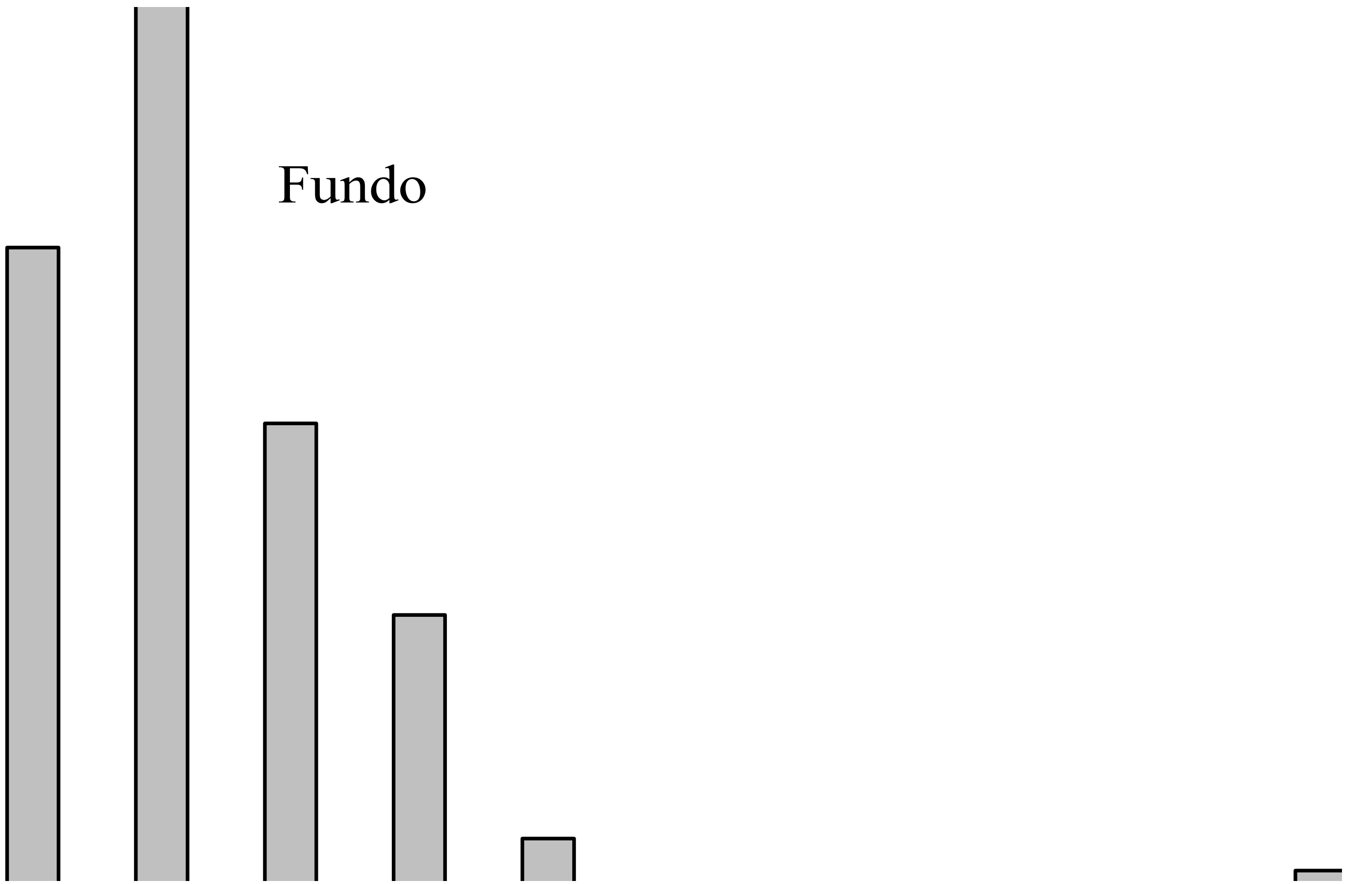
Limiarização - Histograma

- A Limiarização transforma uma imagem I(k,j) monocromática em uma imagem binária $I_B(k,j)$.
  - Pixel de fundo = 0.
  - Pixel de região de objeto = 1.

Limiarização - Histograma

- Considere uma imagem quantizada com precisão de b bits.
- Nível de cinza i varia de 0 a $2^b - 1$
- h(i) = Nº de pixels com nível de cinza i presentes na imagem.
- Histograma = gráfico $h(i) \times i$.

Limiarização - Histograma



Limiarização - Histograma

- Dado um limiar de nível de cinza L, limiarizar:

  Faça k variar em $1 \leq k \leq m$,
  Faça j variar em $1 \leq j \leq n$

  Se $I(k,j) < L$, $I_B(k,j) = 0$,
  caso contrário $I_B(k,j) = 1$.

Rotulagem de Regiões

- Imagem binária I(k,j) $\Rightarrow$ imagem em que a cada região conexa foi atribuído um rótulo diferente.
- Implementação: algoritmo de <u>crescimento de região</u>.

Rotulagem de Regiões

1. Inicializar $k = 1$, $j = 1$.
2. Se $I(k,j) = 1$, fazer $I(k,j) = 255$. // Seleciona regiões que // não são fundo.
3. Fazer $k = k+1$. Se $k \leq m$, ir para o passo 2.
4. Fazer $k = 1$, $j = j+1$. Se $j \leq n$, ir para o passo 2.
5. Inicializar $k = 1$, $j = 1$, $i = 1$. // i é o rótulo
6. Se $I(k,j) = 255$. // Achou semente

- a. Fazer $i = i+1$. // Atualiza rótulo.
- b. Aplicar algoritmo de crescimento de região a partir da semente (k,j).

7. Fazer $k = k+1$. Se $k \leq m$, ir para o passo 6.
8. Fazer $k = 1$, $j = j+1$. Se $j \leq n$, ir para o passo 6.

Crescimento de Regiões

1. Fazer I(k,j) = i, push(k,j), push (0,0). // Rotula e empilha semente,
   // coloca marca na pilha.
2. Se j < n AND I(k,j+1) = 255, // Checa *pixel* acima.
   - a. Fazer I(k,j+1) = i. // Rotula *pixel* acima.
   - b. Push (k,j+1). // Empilha *pixel* acima.
3. Se k > 1 AND I(k-1,j) = 255, // Checa *pixel* esquerdo.
   - a. Fazer I(k-1,j) = i. // Rotula *pixel* esquerdo.
   - b. Push (k-1,j). // Empilha *pixel* esquerdo.
4. Se j > 1 AND I(k,j-1) = 255, // Checa *pixel* abaixo.
   - a. Fazer I(k,j-1) = i. // Rotula *pixel* abaixo.
   - b. Push (k,j-1). // Empilha *pixel* abaixo.
5. Se k < m AND I(k+1,j) = 255, // Checa *pixel* direito.
   - a. Fazer I(k+1,j) = i. // Rotula *pixel* direito.
   - b. Push (k+1,j). // Empilha *pixel* direito.
6. Pop (k,j). Se (k,j) ≠ (0,0) ir para Passo 2. // Desempilha o *pixel* mais recente.
7. Pop (k,j). Retornar. // Restaura endereço da semente.

Rotulagem de Regiões

Observações:

- *Pixel* (k,j) $\in$ região $R_i \Rightarrow I(k,j) = i$.
- *Pixel* 4-conectado: pelo menos um dos seus quatro *pixels* vizinhos (acima, abaixo, à direita ou à esquerda) possui o mesmo valor.
- Algoritmo de Crescimento rotula regiões 4-conectadas. Linhas diagonais da largura de um *pixel* são rotuladas como uma série de pequenas regiões.
- *Pixel* é 8-conectado: pelo menos um dos seus oito *pixels* vizinhos possui o mesmo valor.
- 4-conectividade é um critério mais forte do que 8-conectividade. Todo região 4-conectada também é 8-conectada.
- Algoritmo de rotulagem modificado para utilizar diferentes critérios:
  - 4-conectividade para regiões correspondentes a objetos.
  - 8-conectividade para regiões referentes ao fundo da imagem.

Descritores de Imagem

- Descritores = atributos de um objeto obtidos a partir da sua imagem.
- Objetos poligonais ou poliédricos podem ser descritos pelos seus vértices.
- Objetos de contornos mais complexos precisam de outros descritores.

Descritores de Linha

- Medidas obtidas a partir dos *pixels* que constituem o perímetro limite de um objeto na imagem.
- Poucos *pixels* no contorno implicam baixa robustez em relação ao ruído.
- Contorno de um objeto geralmente incorpora bastante informação relevante sobre o mesmo.

Descritores de Linha

Chain Codes

- Método Direto: lista ordenada das coordenadas dos pontos do perímetro – Pouco eficiente.

- Alternativa mais eficiente: Chain Codes.

Chain Codes

- $a \in \mathbf{R}^n$ = Chain Codes, Códigos de Freeman ou Códigos de Cadeia de uma curva C(a).
- Utiliza mudanças incrementais entre *pixels*.
- Mais eficiente: 3 bits por ponto.
- Invariante a translações.
- Convenção: sentido anti-horário, partindo do *pixel* extremo direito de C(a).

Chain Codes

0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	0
0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0

$a = [3, 4, 5, 7, 0, 0, 2]$

Chain Codes

- Dados a curva C(a) ($a \in \mathbf{R}^n$) e um fragmento visível de C(a) C(b) ($b \in \mathbf{R}^m$) com $m < n$. Índice de Similaridade:

$$\rho_j(a,b) = (\Sigma_{k=1,m} \cos[(a_{k+j} - b_k)\pi/4])/m$$

onde $0 \leq j \leq n\text{-}m$. $-1 \leq \rho_j(a,b) \leq +1$

Chain Codes

- Para verificar se C(b) é uma parte de C(a), calcular $\rho_j(a,b)$ na faixa $0 \leq j \leq n\text{-}m$.
- O valor máximo de $\rho_j(a,b)$ corresponde à máxima similaridade.
- O *pixel* j é o ponto inicial de C(b) em C(a).

Descritores de Área

- Descritores de Área: Medidas obtidas a partir dos *pixels* enclausurados pelo perímetro limite de uma região conexa na imagem.
- Mais robustos do que descritores de linhas.
- Uma região R em uma imagem I(k,j) é <u>conexa</u> se para qualquer par de *pixels* pertencentes a R existe um caminho em R conectando o par (de acordo com alguma regra de conectividade, por exemplo, 4-vizinhança, 8-vizinhança).

Momentos

- Momento de ordem k+j ($k \geq 0$, $j \geq 0$) de uma região conexa R:

$$m_{kj} = \sum_{(x,y)\in R} x^k . y^j$$

- Momentos são atributos que caracterizam tamanho, forma e orientação de um objeto.

Momentos de Baixa Ordem

- Caracterizam área e posição do centróide:

$A = m_{00}$

$x_c = m_{10}/m_{00}$

$y_c = m_{01}/m_{00}$

Momentos Centrais

- Calculados em relação ao centróide (x_c,y_c).

$$\mu_{kj} = \sum_{(x,y)\in R} (x-x_c)^k.(y-y_c)^j$$

- Invariantes a translações.

∀ μ_{02} e μ_{20} são os momentos de inércia de R em relação x e y através do centróide

∀ μ_{11} é o produto de inércia de R.

Momentos Centrais Normalizados

- Momentos centrais normalizados em relação à área:

$$\nu_{kj} = \mu_{kj}/\mu_{00}^{(k+j+2)/2}$$

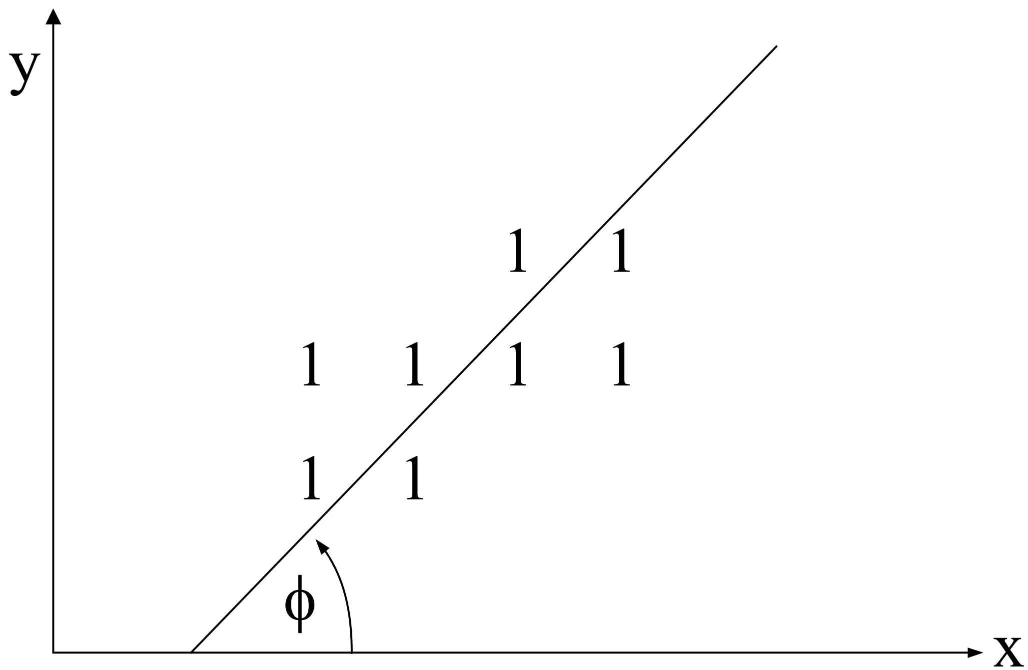
- Invariantes a mudanças de escala.

Ângulo Principal

- Ângulo de orientação do eixo passando pelo centróide de R que minimiza o seu momento de inércia:

$\forall\ \phi = [\mathrm{atan2}(2.\mu_{11}, \mu_{20} - \mu_{02})]/2$

Ângulo Principal



Ângulo Principal

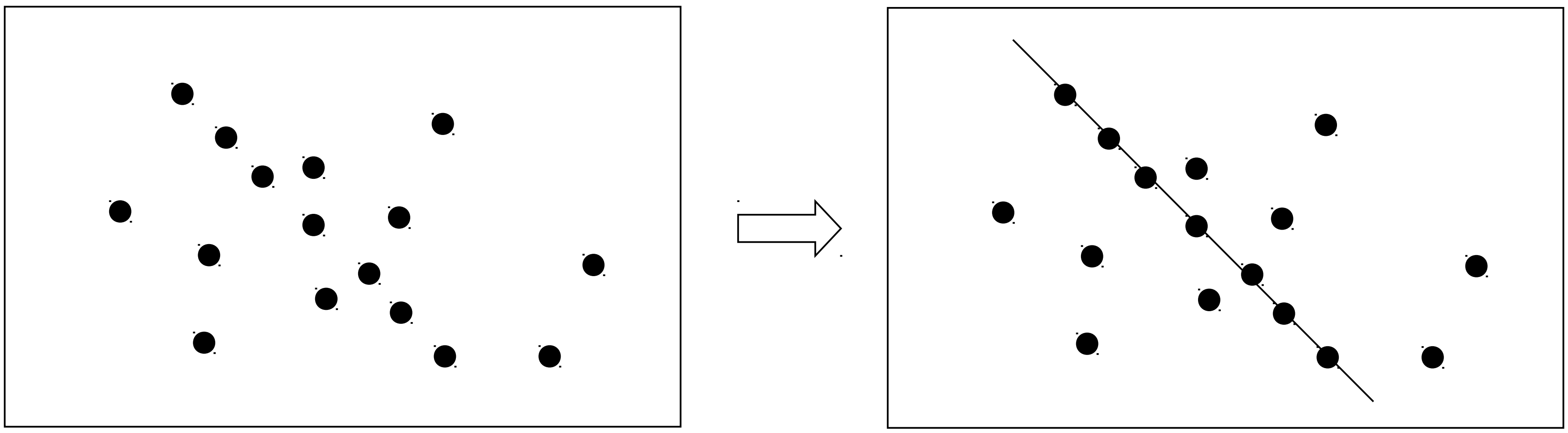
- O Ângulo Principal pode ser utilizado para obter medidas de R invariantes à rotação:
  - Transladar R para a origem (através de um vetor $(-x_c, -y_c)$ e girar R um ângulo $-\phi$.
  - A região resultante possui centróide na origem e ângulo principal igual a zero.
  - Os momentos normalizados são invariantes a translação, rotação e mudança de escala.

Transformada de Hough

- Técnica para detectar formas geométricas em imagens.
- Aplica-se a formas geométricas planas representadas por curvas paramétricas, (Ex: retas, círculos e elipses, etc.).

Transformada de Hough para Retas

- Determina a equação da reta que se ajusta ao maior número de pontos alinhados na imagem.



Transformada de Hough para Retas

- Realiza mapeamento entre o espaço cartesiano da imagem e o espaço de parâmetros da reta.
- Pontos da reta se concentram no espaço de parâmetros de acordo com as características que os unem no espaço cartesiano.
- Pontos colineares na imagem se acumulam em um ponto no espaço de parâmetros, o qual corresponde aos parâmetros da reta que passa pelos pontos alinhados na imagem.

Transformada de Hough para Retas

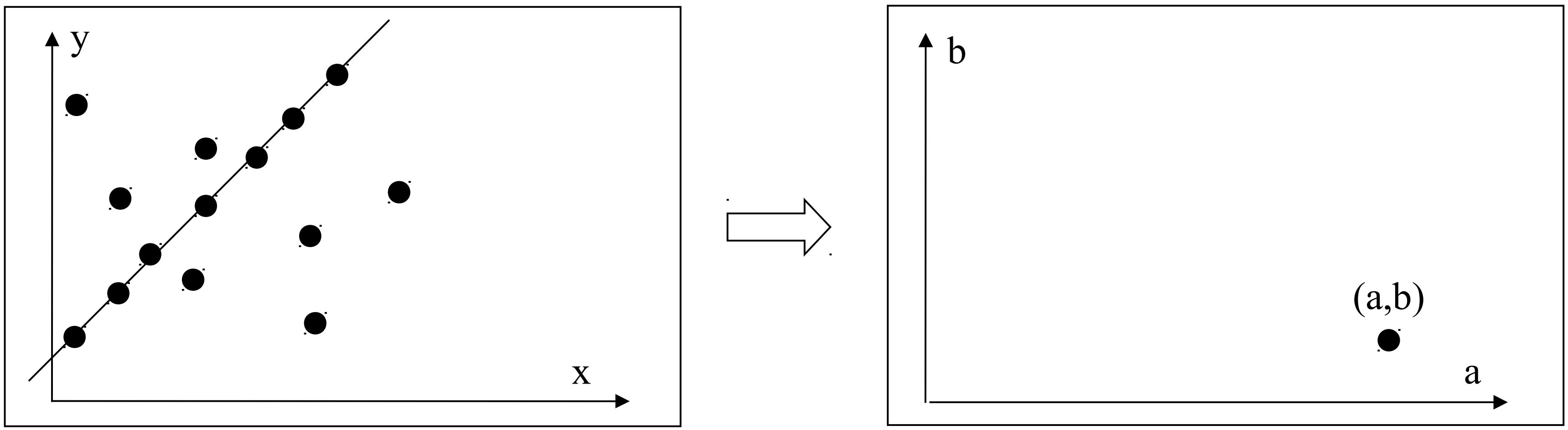
- Uma reta na imagem pode ser parametrizada por:

$$y = a.x + b,$$

- a = inclinação da reta.
- b = ponto em que a reta corta o eixo y.

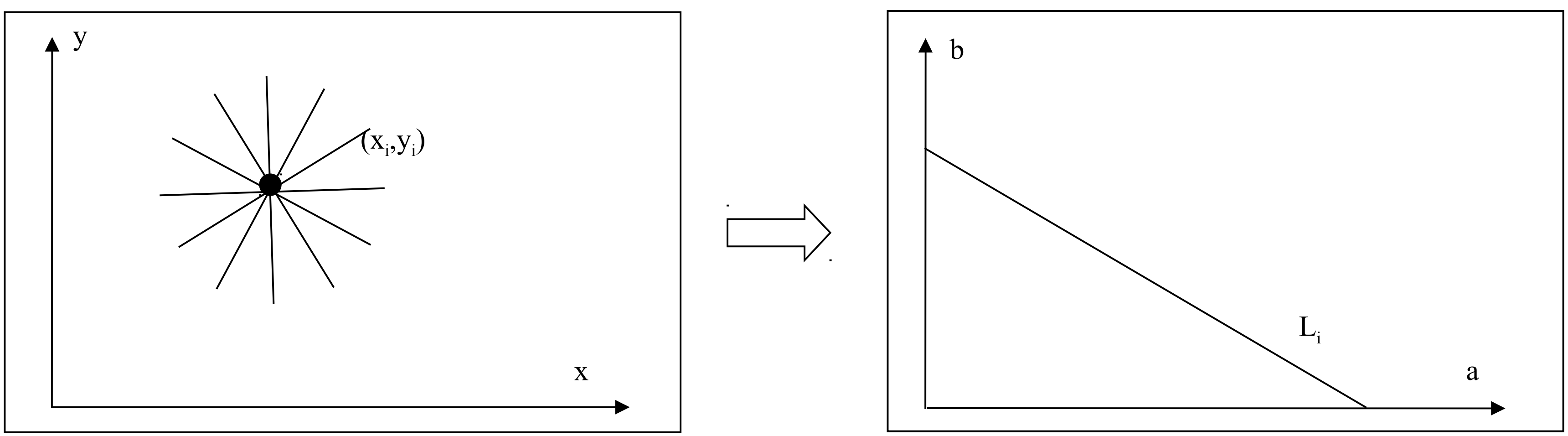
Transformada de Hough para Retas

- Pontos que pertencem à mesma reta no plano XY correspondem a um único ponto (a,b) no plano do espaço de parâmetros AB.



Transformada de Hough para Retas

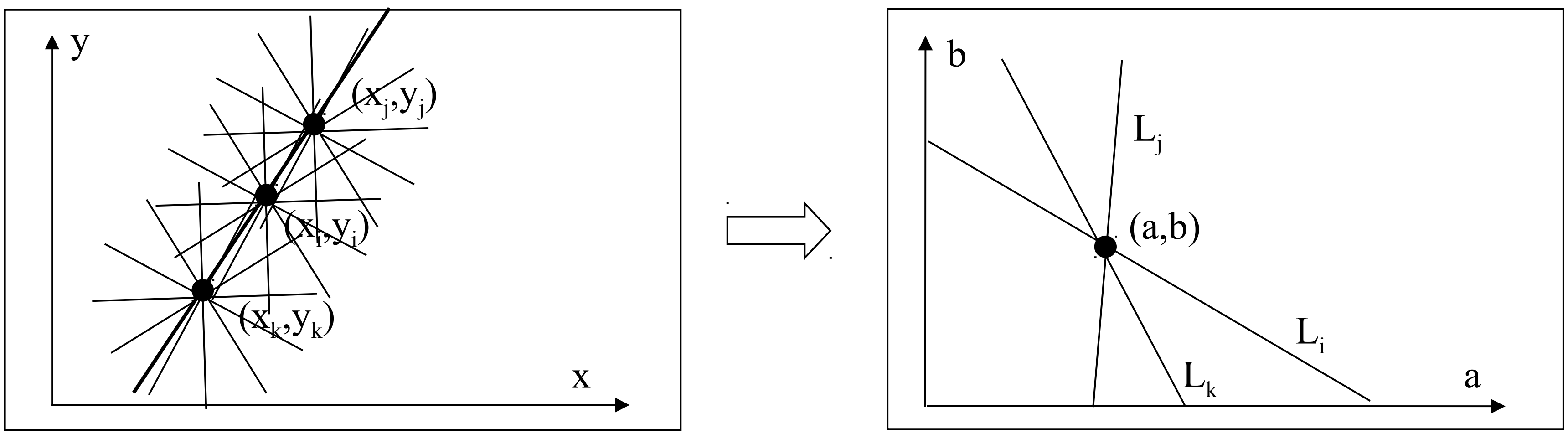
- Para cada ponto (x_i,y_i) da imagem, calcula-se os parâmetros de todas as possíveis retas que passam por ele.
- No plano AB, correspondem à reta L_i: $b = x.a – y$.



Transformada de Hough para Retas

- Repetindo o procedimento para cada ponto na imagem, obtêm-se retas correspondentes no plano de parâmetros.
- O ponto de interseção dessas retas, corresponde ao ponto (a,b) que representa os parâmetros de uma reta que passa por pontos colineares na imagem.
- Para vários alinhamentos na imagem , várias interseções no espaço de parâmetros.
- O ponto que acumula mais interseções de retas no plano AB, corresponde à reta com mais pontos na imagem.

Transformada de Hough para Retas

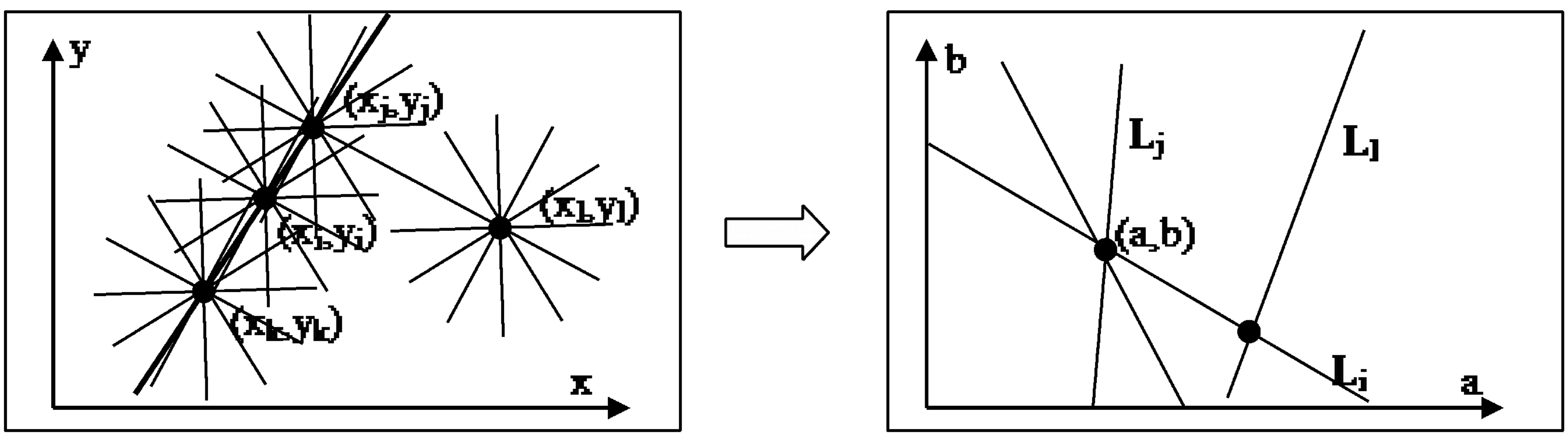


Transformada de Hough para Retas

- Para vários alinhamentos na imagem , várias interseções no espaço de parâmetros.
- O ponto que acumula mais interseções de retas no plano AB, corresponde à reta com mais pontos na imagem.

Transformada de Hough para Retas

- Para vários alinhamentos na imagem, teremos várias interseções de retas no espaço de parâmetros.
- O ponto que acumula mais interseções de retas no plano AB, corresponde à reta com mais pontos na imagem.



Representação Normal da Reta

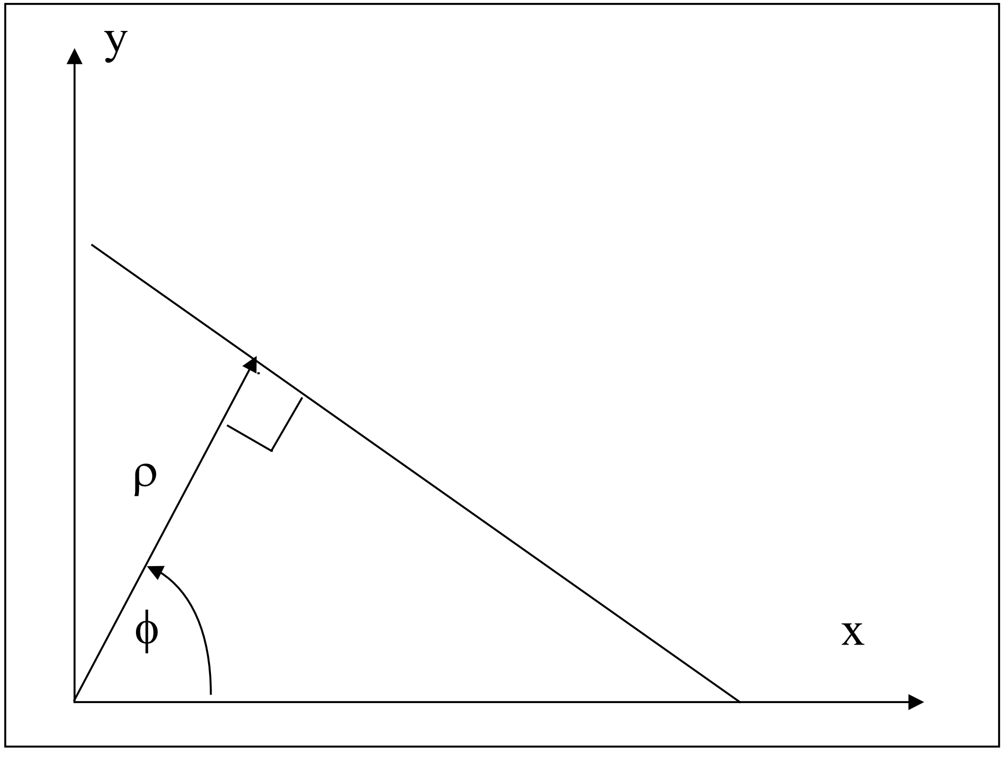
- Representação paramétrica $y = a.x + b$ não é muito adequada para a Transformada de Hough.
- Parâmetros a e b podem assumir valores de $-\infty$ a $+\infty$, tornando o espaço de busca inviável na prática.
- Alternativa $\rightarrow$ representação normal da reta:

$$\rho = x.\cos(\phi) + y.\text{sen}(\phi)$$

- Dado o vetor que passa pela origem e é normal à reta, ρ é o módulo (distância da reta à origem), ϕ é o ângulo entre o vetor e o eixo x.

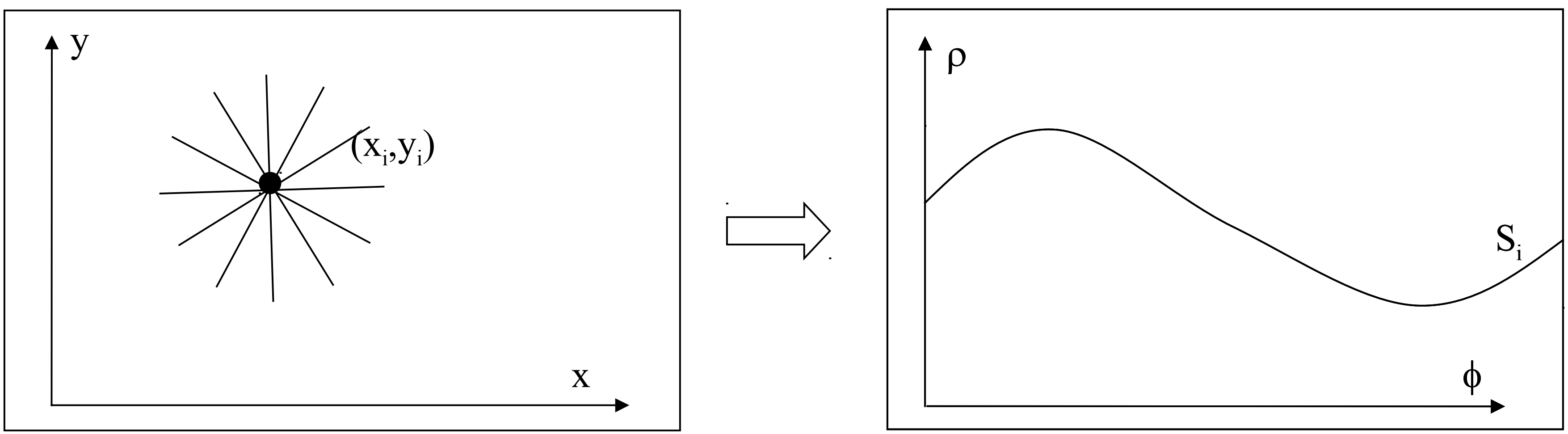
Representação Normal da Reta

- Como ($0^o \leq \phi \leq 180^o$) e ρ é limitado pelo tamanho da imagem, o espaço de busca de parâmetros é limitado.



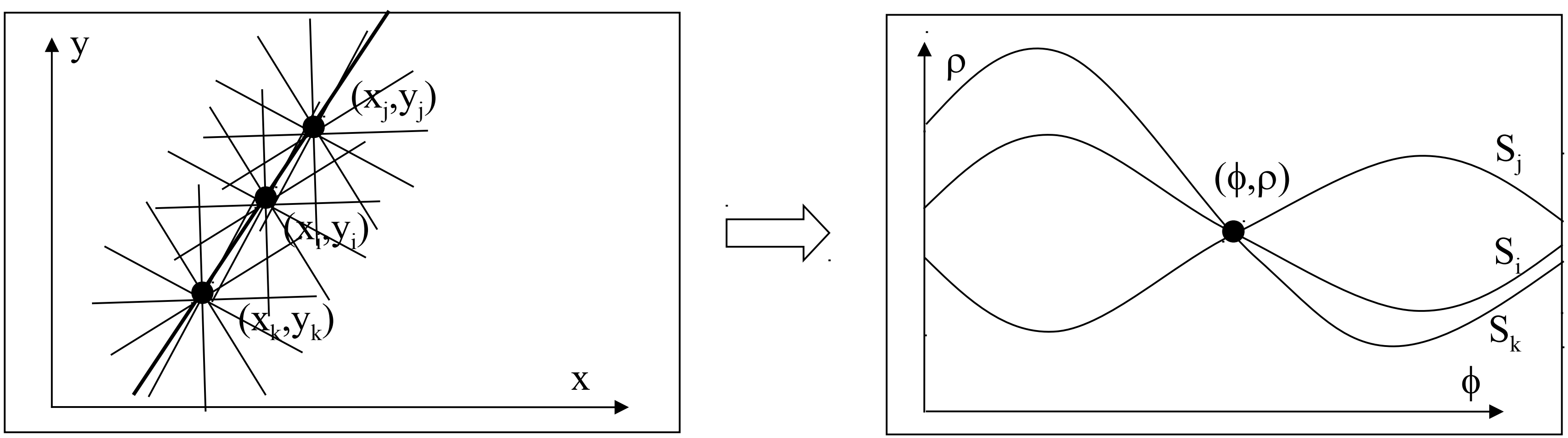
Representação Normal da Reta

- Possíveis retas que passam por um ponto (x_i,y_i) na imagem são representadas por uma senoide S_i no espaço de parâmetros $\phi\rho$.



Representação Normal da Reta

- Pontos colineares na imagem são representados por senoides que se intersectam em um ponto (ϕ,ρ) no espaço de parâmetros.
- Valores de ϕ e ρ são os parâmetros procurados da reta que passa pelos pontos na imagem.



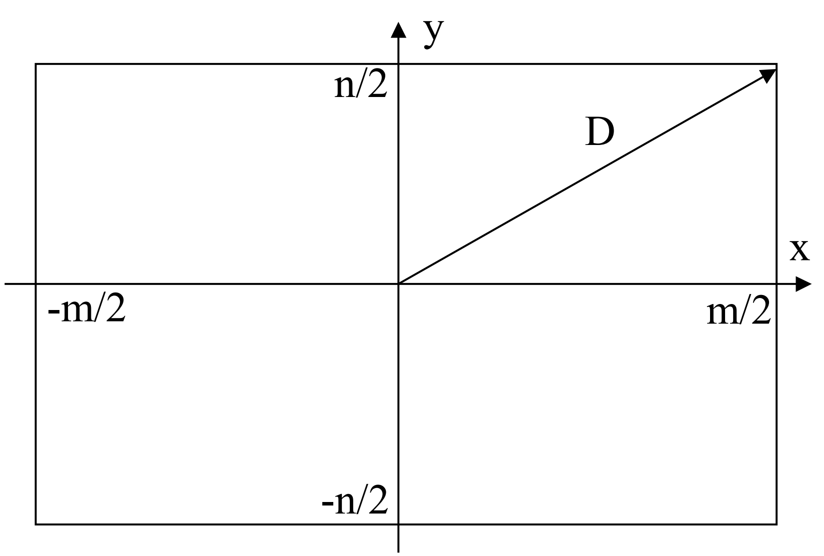
Aspectos de Implementação da Transformada de Hough

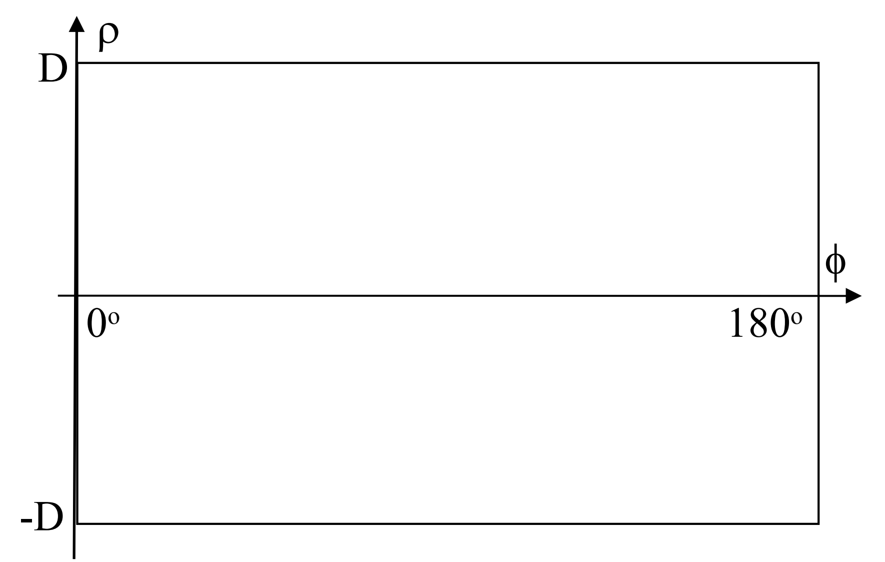
- Para implementar a Transformada de Hough, tanto a imagem, como o espaço de parâmetros devem ser discretizados.
- Eixos X, Y e ϕ, ρ devem ser quantizados adequadamente.
- Uma imagem digital é naturalmente discreta, representada por mxn pixels.
- Se a origem do referencial da imagem estiver no seu centro, então: $x \in [-m/2, m/2]$ e $y \in [-n/2 , n/2]$.

Aspectos de Implementação da Transformada de Hough

- Dado D = comprimento da maior diagonal da imagem:

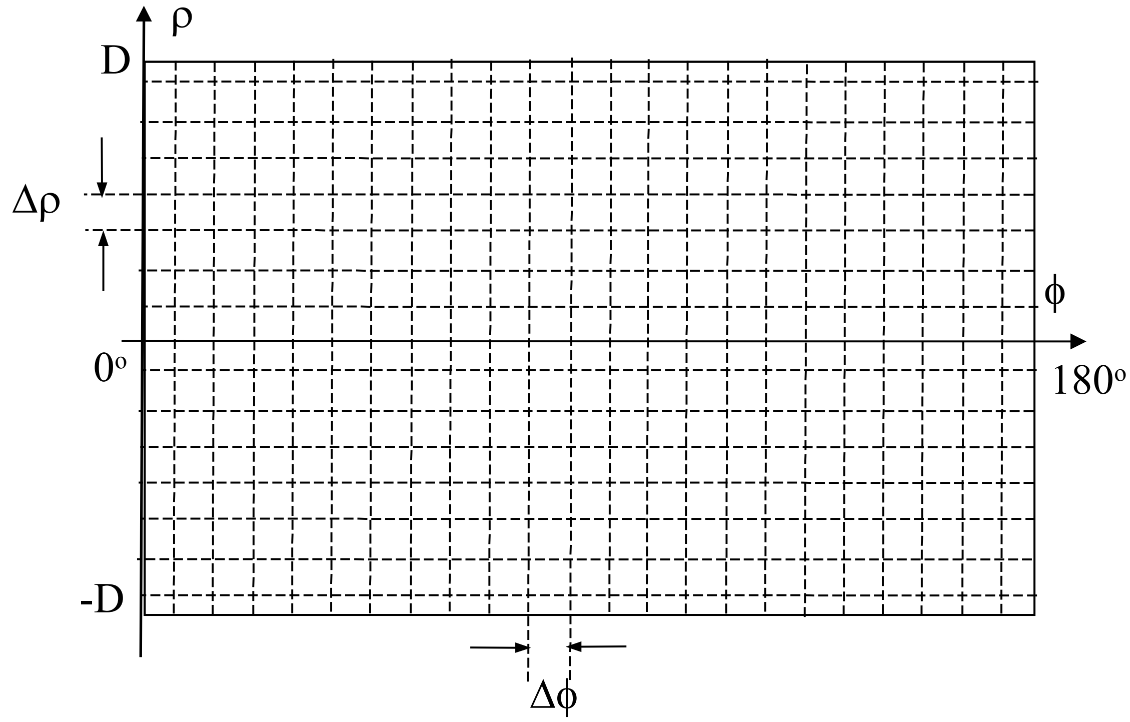
$\phi \in [0^o, 180^o]$ $\rho \in [-D , D]$





Aspectos de Implementação da Transformada de Hough

- O espaço de parâmetros é discretizado numa Matriz Acumuladora, quantizando os eixos ϕ, ρ em intervalos discretos $\Delta\phi$, $\Delta\rho$.



Aspectos de Implementação da Transformada de Hough

A escolha dos intervalos de quantização $\Delta\phi$ e $\Delta\rho$ deve ser feita cuidadosamente:

- Valores muito pequenos podem resultar em um tempo de processamento grande.

- Valores grandes diminuem a precisão (pontos não estritamente alinhados podem ser considerados como pertencentes à mesma reta).

Aspectos de Implementação da Transformada de Hough

- Para executar a Transformada de Hough em uma imagem digital, inicialmente a mesma é pré-processada e limiarizada, de forma a transformá-la em um bitmap.

- A matriz acumuladora é inicializada com todos os seus elementos nulos.

Aspectos de Implementação da Transformada de Hough

- Quando a transformada é aplicada a um ponto (x,y) da imagem, para ϕ dentro do intervalo de busca, obtém-se o valor quantizado de ρ, de acordo com a senoide:

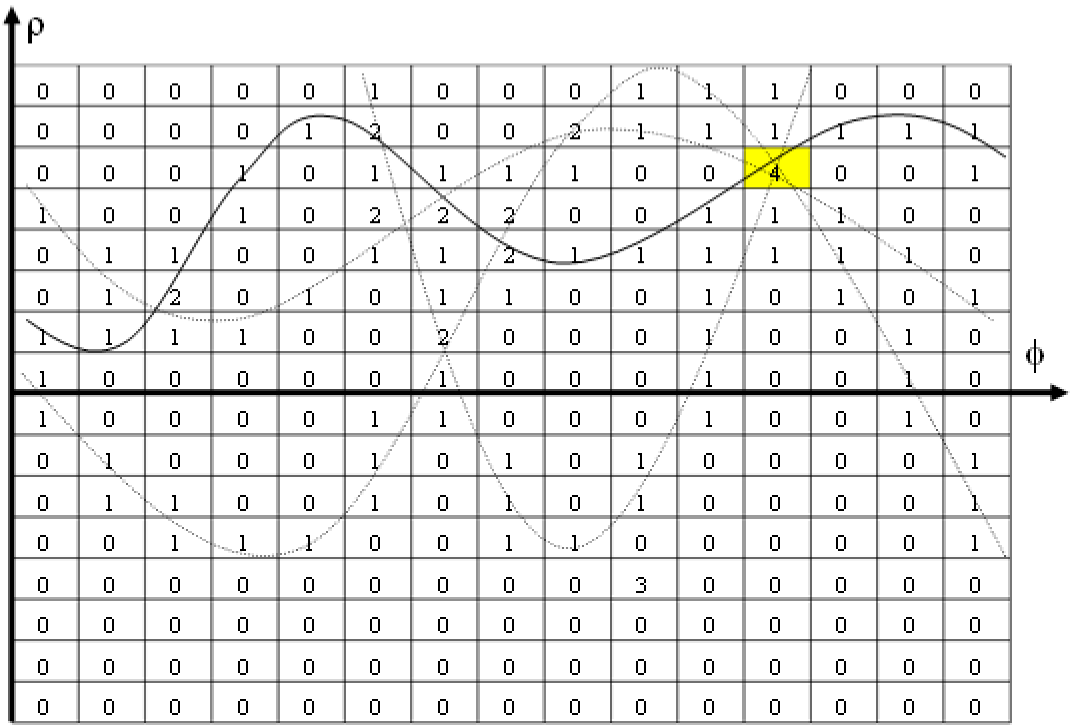
  $$\rho = x.\cos(\phi) + y.\mathrm{sen}(\phi)$$

- Desta forma, todos os elementos correspondentes (ϕ,ρ) da matriz acumuladora são incrementados em uma unidade.

Aspectos de Implementação da Transformada de Hough

- O processo é repetido para cada ponto da imagem,
- Ao final, cada elemento da matriz acumuladora terá um valor correspondente ao número de senoides que passam pelo ponto (ϕ,ρ) correspondente no espaço de parâmetros.
- O elemento da matriz com valor máximo corresponde aos parâmetros da reta que passa pelo maior número de pontos alinhados na imagem.
- Para achar os parâmetros dessa reta é necessário varrer a matriz acumuladora para procurar o máximo valor acumulado.

Aspectos de Implementação da Transformada de Hough



Casamento de Padrões

- Tarefa: reconhecer se um objeto presente em uma imagem $I(k,j)$ mxn é membro ou não de uma classe de objetos conhecidos.
- Solução simples: Casamento de Padrões.
- Padrão (máscara): imagem $T(k,j)$ representativa de um objeto, onde $1 \le k \le m_o$ e $1 \le j \le n_o$.
- Para um conjunto de N classes diferentes de objetos construir Biblioteca de padrões:

$$BIB = \{T_i(k,j): 1 \le k \le m_o \le m,\ 1 \le j \le n_o \le n,\ 1 \le i \le N\}$$

Casamento de Padrões

- Procedimento: varrer a imagem com o padrão e medir o grau de casamento.

$\Rightarrow$ translação (x,y) de $T_i(k,j)$ sobre $I(k,j)$ = superpor o padrão sobre $I(k+x,j+y)$.

- Para não ultrapassar os limites da imagem, $1 \leq x \leq m-m_o$, $1 \leq y \leq n-n_o$.

Casamento de Padrões

- Exemplo de Índice de Desempenho:

$$\rho_i(x,y) = \sum_{k=1}^{m_o} \sum_{j=1}^{n_o} |I(k+x,j+y)-T_i(k,j)|$$

$\forall\ \rho_i(x,y) \geq 0.$

- Casamento perfeito: $\rho_i(x,y) = 0$.
- Na prática, devido a ruídos, temos casamento se: $\rho_i(x,y) \leq \varepsilon$ (Limiar).

Algoritmo - Casamento de Padrões

- 1. Inicializar $i = 1$, $x = 0$, $y = 0$, $\varepsilon > 0$, Achado = VERDADEIRO.
- 2. Calcular $\rho_i(x,y)$.
- 3. Se $\rho_i(x,y) \leq \varepsilon$, Retornar (x,y), Parar.
- 4. Fazer $x = x+1$. Se $x \leq m-m_o$, Ir para o Passo 2.
- 5. Fazer $x = 0$, $y = y+1$. Se $y \leq n-n_o$, Ir para o Passo 2.
- 6. Fazer $x = 0$, $y = 0$, $i = i+1$. Se $i \leq N$, Ir para o Passo 2.
- 7. Fazer Achado = FALSO.

Casamento de Padrões

- Problema: $\rho_i(x,y)$ sensível variações de iluminação.
- Um bom casamento só ocorrerá se padrão e objeto possuírem as mesmas intensidades médias:

$$||T_i|| = \left[\sum_{k=1}^{m_o} \sum_{j=1}^{n_o} T_i^2(k,j)\right]^{1/2}$$

$$||I_{x,y}|| = \left[\sum_{k=1}^{m_o} \sum_{j=1}^{n_o} I^2(k+x,j+y)\right]^{1/2}$$

Casamento de Padrões

- Solução: normalizar o índice de desempenho.
- Correlação Cruzada Normalizada:

$$\rho_i(x,y) = [\sum_{k=1}^{m_o} \sum_{j=1}^{n_o} I(k+x,j+y).T_i(k,j)] / \|T_i\|.\|I_{x,y}\|$$

- Quando ocorre o casamento, o índice é maximizado.

Calibração de Câmera

- Para relacionar atributos obtidos de imagens de objetos, (em *pixels*), com as dimensões reais (em metros) desses objetos, é necessário obter transformações apropriadas que dependem dos parâmetros da câmera.
- Calibração: determinação destes parâmetros.
- Captura de um padrão com dimensões e localização conhecida.
- Reconhecimento e determinação da localização do padrão na imagem.
- Montagem e solução de sistema de equações relacionando pontos conhecidos no mundo real com seus correspondentes na imagem.

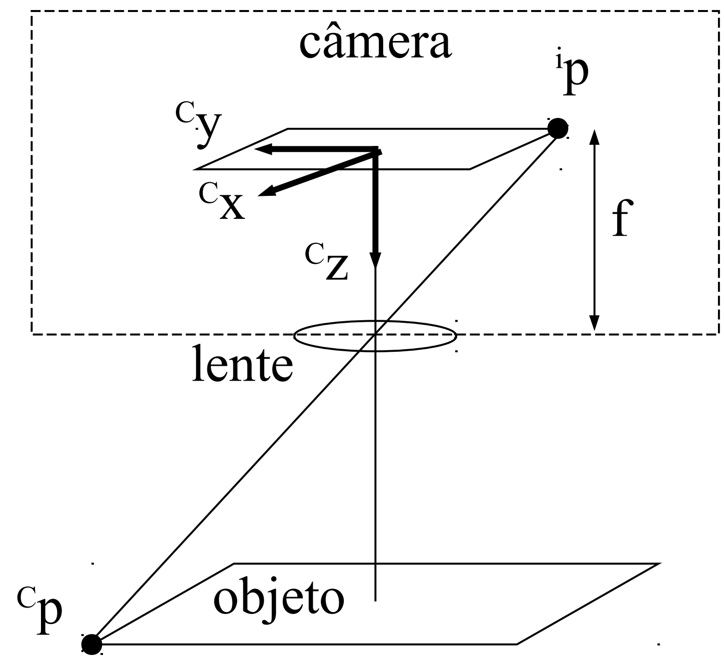
Transformação de Perspectiva

- Mapeamento entre ponto em referencial de Câmera ${}^{C}p$ e ponto ${}^{i}p$ no plano da imagem:

$-{}^{i}p_x/f = {}^{C}p_x/({}^{C}p_z - f)$

$-{}^{i}p_y/f = {}^{C}p_y/({}^{C}p_z - f)$

${}^{i}p_z = 0$



Transformação de Perspectiva

- Transformação de ${}^{c}p$ para ${}^{i}p$ = mapeamento não linear de $\mathbf{R}^3$ para $\mathbf{R}^3$ (envolve divisão por ${}^{c}p_z$).
- Não pode ser representada por uma matriz de transformação 3x3.
- É possível representar a transformação usando um operador matricial em um espaço de maior dimensão.
- Solução: representar os pontos em coordenadas homogêneas.

Transformação de Perspectiva

- Coordenadas homogêneas de um vetor p:

  $p_h = [e.p_x, \quad e.p_y, \; e.p_z, \; e]^T \qquad e \neq 0$

  $p_x = p_{hx}/e p_y = p_{hy}/e \quad p_z = p_{hz}/e$

Transformação de Perspectiva

$$^{i}p_h = {}^{i}T_C . {}^{C}p_h$$

$$^{i}T_C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$

Perspectiva Inversa

- Aplicação prática: obter a posição de objetos em relação à câmera a partir de imagens dos mesmos.
- Calcular a transformação de perspectiva inversa, ${}^{C}T_{i}$.
- Problema: a transformação de perspectiva destrói informação no mapeamento 3D para 2D.
- ${}^{i}T_{C}$ é singular.
- Problema pode ser contornado se informação de profundidade for disponível a priori.

Perspectiva Inversa

- Assumindo que ${}^{C}p_z$ é conhecida a priori.
- Ex: objeto sobre esteira, câmera a altura conhecida.
- Artifício: acrescentar informação de profundidade a ${}^{i}p$, construir ponto de imagem aumentado ${}^{ia}p$:

$${}^{ia}p = {}^{i}p - [{}^{C}p_z/({}^{C}p_z - f)]I^3 \qquad\qquad I^3 = [0\ 0\ 1\ 0]^T$$

- Equivale a elevar a imagem através da translação:

$${}^{ia}T_i({}^{C}p_z) = \mathrm{Trans}([{}^{C}p_z/(f - {}^{C}p_z)]I^3)$$

Perspectiva Inversa

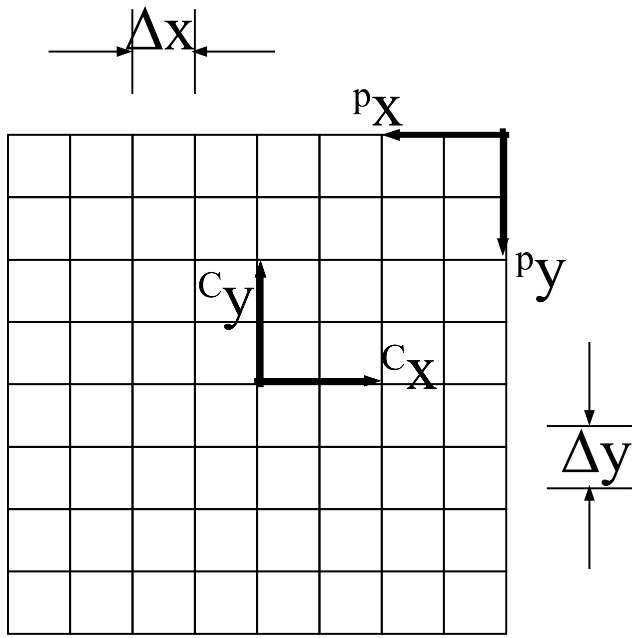
- Transformação de Perspectiva Inversa:

$$ {}^{C}p = {}^{C}T_{i} . {}^{i}p = {}^{C}T_{ia} . {}^{ia}T_{i} . {}^{i}p $$

$$ {}^{C}T_{i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f & f.{}^{C}p_{z}/(f-{}^{C}p_{z}) \\ 0 & 0 & 1 & f/(f-{}^{C}p_{z}) \end{bmatrix} $$

Coordenadas de *Pixel*

- É preciso mapear *pixels* para metros.
- Considere que a câmera possui um matriz de mxn elementos sensores, (*pixel* com largura Δx e altura Δy).



$$^{i}T_{p} = \begin{bmatrix} -1 & 0 & 0 & m.\Delta x/2 \\ 0 & -1 & 0 & n.\Delta y/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Calibração de Câmera

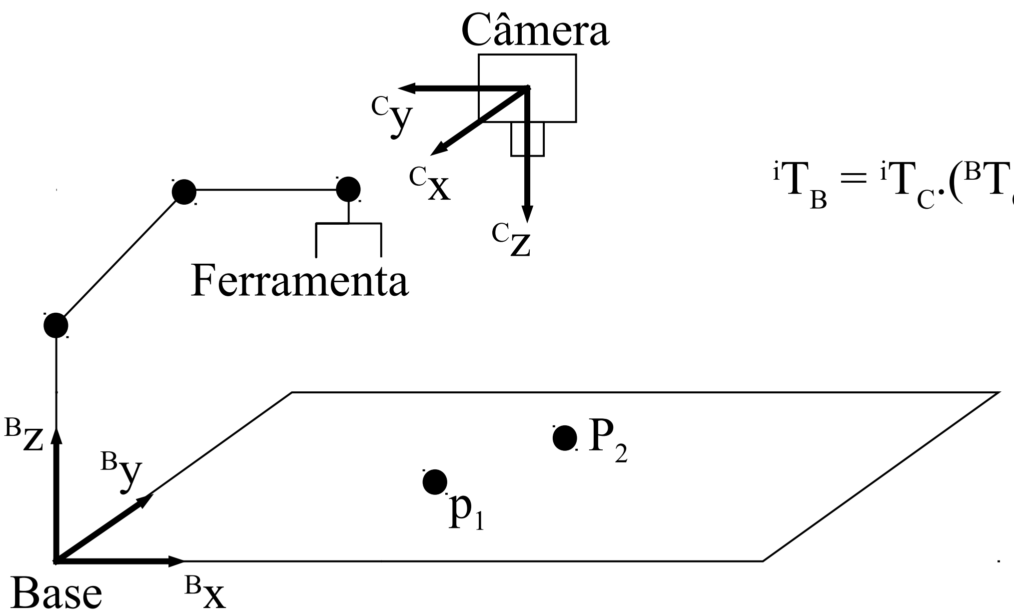
- Dadas as coordenadas de um objeto em *pixels*, as suas coordenadas em relação à base do manipulador podem ser obtidas por uma composição de transformações:

$$ {}^{B}p = {}^{B}T_{C} . {}^{C}T_{i} . {}^{i}T_{p} . {}^{p}p $$

- Os parâmetros destas transformações devem ser conhecidos a priori, obtidos por um processo de calibração.

Calibração de Câmera

- Estudo de caso: orientação da câmera conhecida, incógnita = posição (x_0, y_0, z_0) da câmera.



$$
{}^iT_B = {}^iT_C.({}^BT_C)^{-1} = \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & (f-z_0)/f \end{bmatrix}
$$

Calibração de Câmera

- Dados dois pontos conhecidos, ${}^{B}p_1$ e ${}^{B}p_2$, a imagem dos mesmos, ${}^{i}p_1$ e ${}^{i}p_2$ pode ser capturada, de tal forma que:

  ${}^{i}p_1 = {}^{i}T_B . {}^{B}p_1$ ${}^{i}p_2 = {}^{i}T_B . {}^{B}p_2$

- De onde obtemos quatro equações em função de (x_0, y_0, z_0):

$$f(y_0 - {}^{B}p_{1y}) = {}^{i}p_{1x}(f - z_0)$$
$$f(x_0 - {}^{B}p_{1x}) = {}^{i}p_{1y}(f - z_0)$$
$$f(y_0 - {}^{B}p_{2y}) = {}^{i}p_{2x}(f - z_0)$$
$$f(x_0 - {}^{B}p_{2x}) = {}^{i}p_{2y}(f - z_0)$$

Calibração de Câmera

- Solução: z_0 obtido subtraindo a 3ª da 1ª Eq. x_0 e y_0 obtidos da 1ª e da 2ª função de z_0:

$$z_0 = f[1 + ({}^Bp_{1y} - {}^Bp_{2y})/({}^ip_{1x} - {}^ip_{2x})]$$
$$y_0 = {}^Bp_{1y} + (f - z_0){}^ip_{1x}/f$$
$$x_0 = {}^Bp_{1x} + (f - z_0){}^ip_{1y}/f$$

- Solução válida para ${}^ip_{1x} \neq {}^ip_{2x}$. Quando ${}^ip_{1x} = {}^ip_{2x}$, solução alternativa derivada subtraindo a 4ª da 2ª Eq. para obter z_0:

$$z_0 = f[1 + ({}^Bp_{1x} - {}^Bp_{2x})/({}^ip_{1y} - {}^ip_{2y})]$$

- Desde que ${}^ip_1 \neq {}^ip_2$, sempre uma das duas soluções é factível.