7. INTRODUÇÃO ÀVISÃO ROBÓTICA

Neste capítulo fazemos uma pequena introdução às técnicas de processamento de imagem e à sua aplicação em visão robótica.

7.1. Captura e representação de imagens

Imagem Monocromática:

Uma imagem bidimensional monocromática no plano xy pode ser representada por uma função contínua i(x,y) que especifica a intensidade de luz refletida. A imagem contínua é discretizada ao ser capturada por uma câmera que possui uma matriz de mxn elementos sensores de largura Δx e altura Δy. Isto resulta em uma imagem amostrada com resolução espacial de mxn *pixels*. A superfície do elemento sensor na linha k e na coluna j captura a intensidade luminosa média:

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

Onde a função não negativa $I_a(k,j)$ assume valores sobre uma faixa contínua de intensidade luminosa. A intensidade de cada *pixel* é quantizada com uma precisão de b bits, resultando em 2^b possíveis valores de intensidade (níveis de cinza).

Desta forma, a imagem digital I(k,j) é uma imagem analógica i(x,y) amostrada com resolução espacial mxn e quantizada em intensidade com uma precisão de 2^b níveis de cinza.

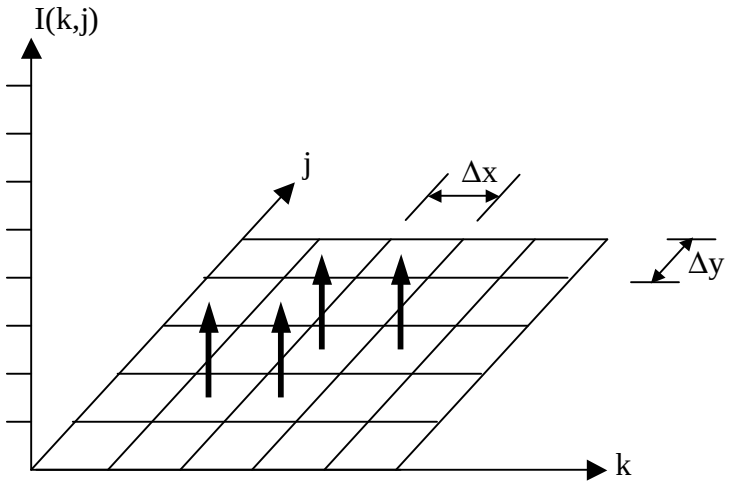

Fig. 7.1 – Imagem 5x5 com 8 níveis de cinza.

Imagem Colorida:

Uma imagem colorida inclui, além da informação de luminosidade de cada *pixel*, a informação da cor do mesmo. Para representar estas informações são necessários três valores. Desta forma, modelos de representação de cor se baseiam em subespaços de cor tridimensionais.

Modelo RGB:

Modelo baseado na teoria do tri-estímulo da visão, o olho humano percebe a cor através da ativação de três pigmentos visuais nos cones da retina, os quais têm picos de sensibilidade nos comprimentos de onda por volta de 630nm (vermelho), 530nm (verde) e 450nm (azul). A cor de uma fonte luminosa é percebida como a combinação das intensidades dessas cores primárias.
O modelo RGB baseia-se em um sistema de coordenadas cartesianas, onde cada cor aparece em seus componentes espectrais primários Red (vermelho), Green (verde) e Blue (azul). A soma das três cores primárias forma a cor branca, e a ausência das três corresponde à cor preta. verde e azul. O modelo RGB pode ser representado, com valores normalizados no intervalo [0; 1], formando um subespaço cúbico de um sistema de coordenadas tridimensionais. A cor preta corresponde à origem do sistema de coordenadas. A cor branca corresponde ao vértice (1; 1; 1) do cubo de cores. A escala de cinza varia do preto ao branco, ao longo da diagonal entre esses dois vértices. As cores primárias (vermelho, verde e azul) são representadas pelos vértices do cubo sobre os eixos do sistema de coordenadas. Os vértices remanescentes representam as cores complementares (amarelo, ciano e magenta). Cada cor primária somada ao seu par complementar (vermelho/ciano, verde/ magenta, azul/amarelo) resulta na cor branca.

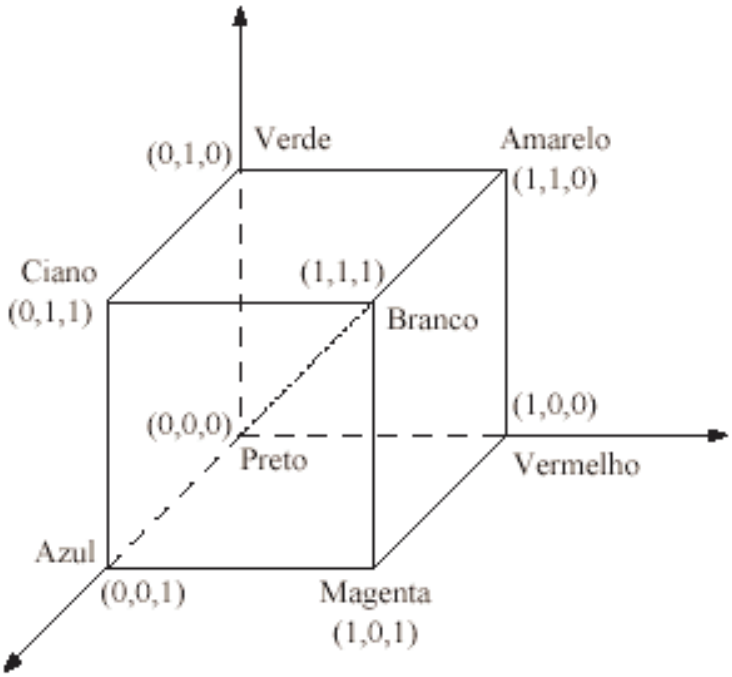

Fig. 7.2 – Modelo RGB.

Modelo HSL:

Derivado do cubo de cores do modelo RGB, O modelo HSL (Hue, Saturation, Lightness) baseia-se em parâmetros intuitivos de cor e é representado por uma pirâmide hexagonal dupla, a qual constitui um subconjunto de um sistema de coordenadas cilíndricas. Os seus parâmetros são especificados como:

- Matiz de Cor H: especifica um ângulo em torno do eixo vertical da pirâmide, variando de 0^o (vermelho) a 360^o. Os vértices do hexágono, separados por ângulos de 60^o, correspondem às cores primárias e suas cores complementares correspondentes. Cada vértice de cor complementar situa-se a 180^o do vértice da respectiva cor primária. O parâmetro H possui valor indefinido para a escala de cinza, que vai do preto ao branco, ao longo do eixo da pirâmide.
- Saturação S: medida ao longo do eixo radial, especifica a pureza relativa (diluição com a cor branca) da cor. Varia de 0 a 1. Para cores puras, S = 1. À medida em que S diminui, a pureza da cor também diminui. A escala de cinza corresponde a S = 0.
- Luminância L: medida ao longo do eixo da pirâmide, possui valor 0 para o preto e 1 para o branco. Especifica a quantidade de luz na cor. Para L = 0:5 e S = 1 temos o hexágono que corresponde às cores puras.

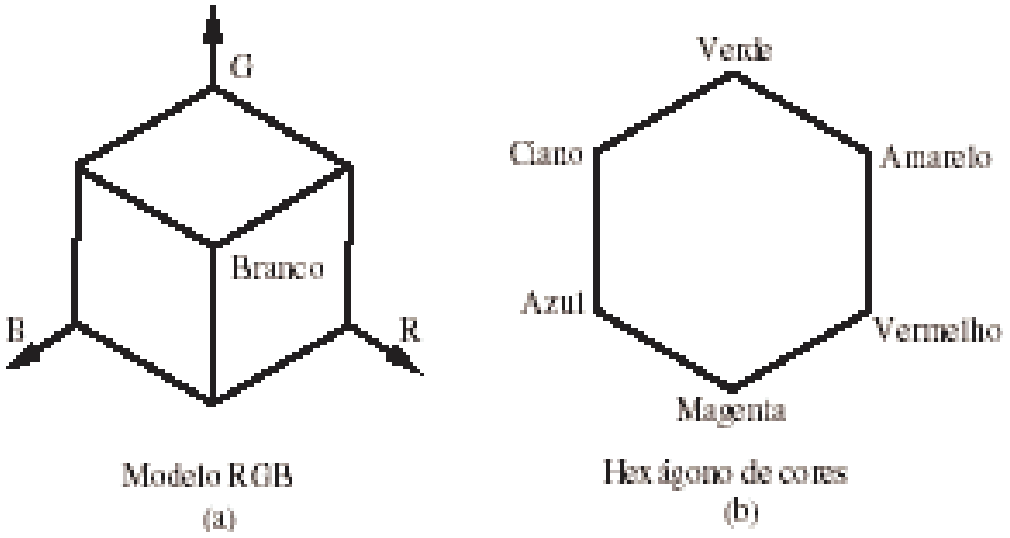

Fig. 7.3 – Derivação do Modelo HSL a partir do Modelo RGB.
a) Cubo RGB visto a partir da sua diagonal. b) Hexágono de cores puras.

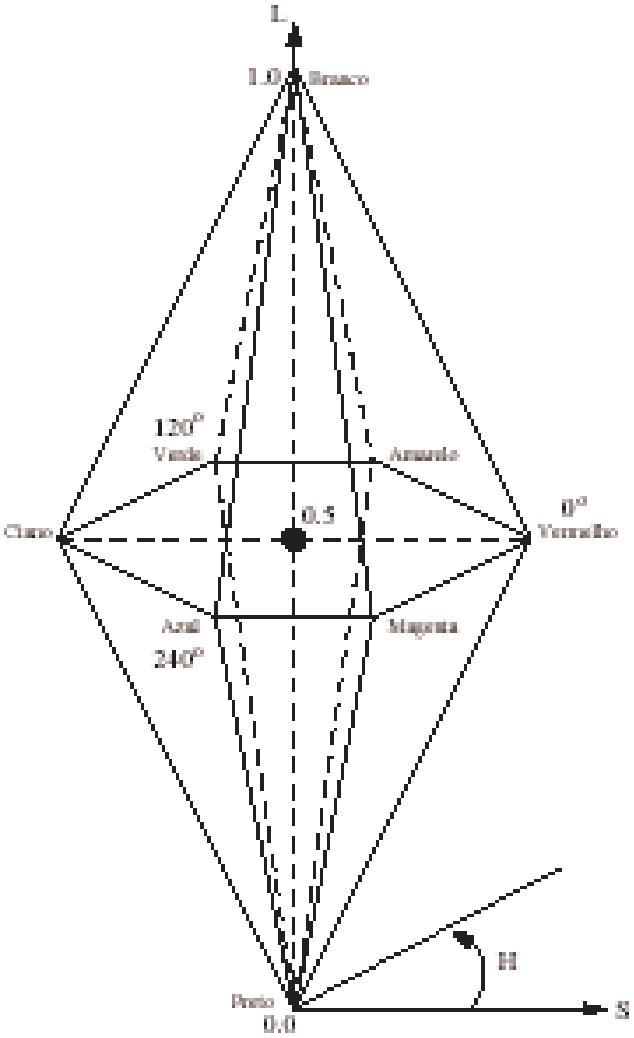

Fig. 7.4 – O Modelo de cor HSL.

7.2. Pré-processamento

Detecção de eixos e bordas:

A intensidade de luz refletida geralmente varia de forma contínua sobre a superfície da face de um objeto. Na passagem de uma face para outra ou na passagem entre um objeto e outro a intensidade luminosa geralmente possui uma descontinuidade. Assim, nessas régiões, o gradiente da intensidade luminosa $\nabla i(x,y) = [\partial i(x,y)/\partial x, \partial i(x,y)/\partial y]$ tende a infinito e o seu valor absoluto pode ser utilizado para detectar bordas e eixos de objetos. Para imagens digitais I(k,j) devemos utilizar uma aproximação discreta do gradiente.

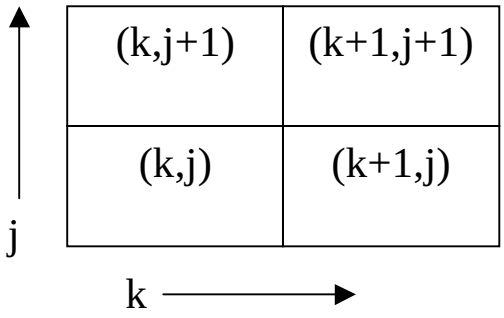

Fig. 7.5 – *Pixels* usados para aproximar o gradiente de intensidade luminosa.

Definindo a aproximação discreta do gradiente como $\nabla I(k,j) = [\nabla I_x(k,j), \nabla I_y(k,j)]$, as componentes $\nabla I_x(k,j)$ e $\nabla I_y(k,j)$ são obtidas pela média das primeiras diferenças ao longo das linhas j e j+1 e das colunas k e k+1 da imagem respectivamente:

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

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

onde Δx e Δy são os espaçamentos entre *pixels* ao longo dos eixos x e y respectivamente. A magnitude do gradiente de intensidade é:

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

No caso em que Δx e Δy são iguais, a magnitude do gradiente pode ser simplificada para:

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

A direção do vetor gradiente pode ser aproximada pelo ângulo:

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

Um eixo ou uma borda de um objeto pode ser detectado no *pixel* (k,j) se a magnitude do gradiente for maior do que um certo valor positivo de limiar de eixo ε:

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

Algoritmo para detecção de Eixos e Bordas em imagem monocromática m×n:

1. Inicializar k = 1, j = 1, ε > 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.

A imagem binária m×n L(k,j) é a saída do algoritmo, onde *pixels* iguais a 1 representam os eixos presentes na imagem. Observe que apenas as m-1 primeiras linhas e as n-1 primeiras colunas são varridas pelo algoritmo, visto que o cálculo da magnitude do gradiente requer o uso de mais três *pixels* vizinhos para cada *pixel*.

A especificação do limiar ε é fundamental para o bom funcionamento do algoritmo. Se ε for muito grande, apenas fragmentos de eixos serão obtidos. Se ε for muito pequeno, eixos falsos poderão ser detectados.

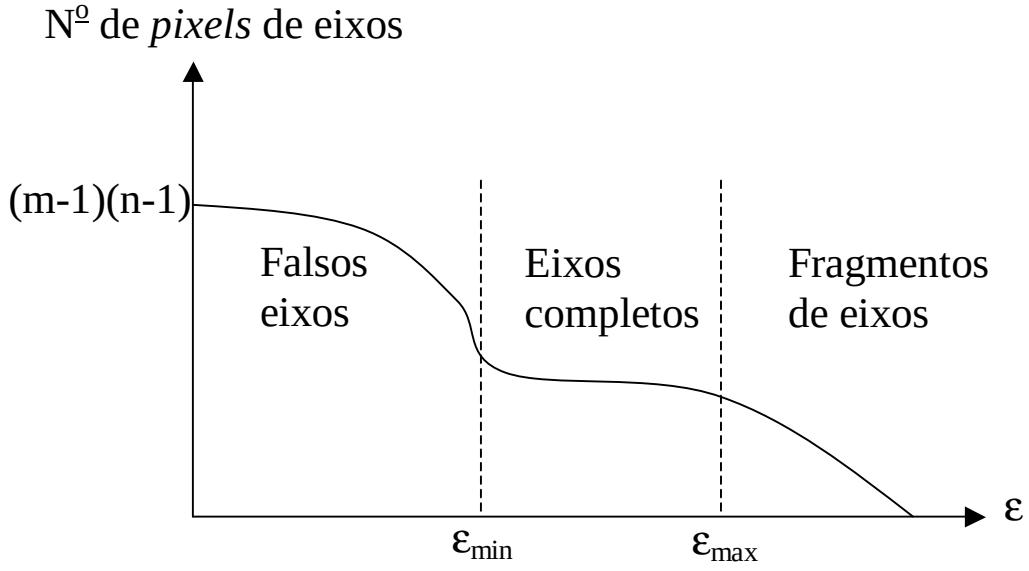

Fig. 7.6 – Influência do limiar ε na detecção de eixos.

Filtragem

Imagens geralmente são contaminadas por ruídos de origem variada. Sombras e variação de iluminação podem fazer com que regiões de borda apareçam borradas e com pequenos apêndices e entradas espúrios. Dependendo dos limiares utilizados, regiões pequenas podem aparecer no fundo da imagem ou buracos pequenos podem ser criados na superfície dos objetos em cena. Operadores iterativos derivados da morfologia matemática podem ser utilizados para filtrar imagens usando técnicas locais, de modo a minimizar este tipo de problemas, suavizando contornos e eliminando parte do ruído. Para formular operadores morfológicos, define-se a seguinte função de *pixel* sobre uma imagem binária mxn I(k,j):

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

Com u,v = -1, 0, 1, 1 < k < m, 1 < j < n.. Verifica-se que esta função retorna o número de *pixels* iguais a 1 vizinhos ao *pixel* (k,j). Assim, $0 \leq p(k,j) \leq 8$. A função de *pixel* só está definida para *pixels* interiores de uma imagem, já que os *pixels* da borda da imagem não possuem um conjunto completo de vizinhos.

Operador de Erosão (*Shrink*):

O operador de erosão shrink(i) I(k,j) torna igual a zero o *pixel* (k,j) se pelo menos i vizinhos possuem valor zero, caso contrário mantém o seu valor:

$$\text{shrink(i) } I(k,j) = I(k,j) \text{ AND } 1(i - 1 - [8 - p(k,j)])$$

onde $0 \leq i \leq 8$ e 1(.) é a função degrau unitário. O operador shrink é monotônico, uma vez que o número de pixels iguais a 1 na imagem resultante é sempre menor ou igual do que o número de pixels iguais a 1 na imagem original. Aplicado iterativamente a uma imagem converge em um número finito de iterações. Usualmente, i > 4. O operador shrink(8) remove todos os *pixels* isolados do fundo da imagem em uma única iteração. O operador iterativo shrink(7) remove do fundo da imagem todas as regiões de até dois *pixels,* assim como apêndices verticais ou horizontais com um *pixel* de largura e de comprimento arbitrário.

Operador de Dilatação (*Swell*):

O operador de dilatação swell(i) I(k,j) é dual do operador shrink, tornando igual a um o *pixel* (k,j) se pelo menos i vizinhos possuem valor um, caso contrário mantém o seu valor:

$$\text{Swell(i) } I(k,j) = I(k,j) \text{ OR } 1(p(k,j)-i)$$

onde $0 \leq i \leq 8$ e 1(.) é a função degrau unitário. O operador swell é utilizado para eliminar pequenos buracos na superfície dos objetos em cena. O operador swell opera sobre pequenos buracos na imagem tal como o operador shrink remove pequenas regiões no fundo da imagem.

Operadores de Abertura e Fechamento

Os operadores de erosão e dilatação podem ser aplicados de forma combinada de modo a produzir operadores mais complexos.

Operador de Abertura:

O operador de abertura é obtido pela aplicação de uma erosão seguida de uma dilatação da imagem. A erosão elimina pequenos ruídos e abre lacunas em regiões de fraca conexão entre objetos. A dilatação restaura o tamanho original dos objetos.

Operador de Fechamento:

Inverso da abertura, o operador de fechamento é obtido pela aplicação de uma dilatação seguida de uma erosão da imagem. Restaura conexões fracas entre objetos da imagem.

7.3. Segmentação

A segmentação é o processo pelo qual itens em uma imagem são separados do fundo e uns dos outros. Envolve particionar a imagem em regiões conexas e homogêneas de acordo com algum critério ou atributo. O conjunto de *pixels* conexos e que compartilham um mesmo atributo é denominado Região. A cada região deve ser atribuído um rótulo único.

Limiarização:

Para imagens em escala de cinza onde os objetos em cena contrastam bem com o fundo da imagem, a limiarização é o método mais simples de segmentação. A limiarização transforma a imagem monocromática original em uma imagem binária, onde o fundo da mesma contém *pixels* iguais a zero e as regiões correspondentes aos objetos segmentados são compostas de *pixels* iguais a 1.

Considere uma imagem em escala de cinza, onde a intensidade luminosa dos *pixels* foi quantizada em b bits. Isto corresponde a 2^b níveis de cinza diferentes, variando na faixa de 0 a $(2^b - 1)$. Um método comum de limiarização deste tipo de imagem é baseado no Histograma da Imagem, o qual é o gráfico de h(i) versus i, onde h(i) denota o número de *pixels* com nível de cinza i presentes na imagem.

Por exemplo, dada uma imagem com objetos bem claros sobre um fundo escuro, o seu histograma apresentará dois picos, um centrado na tonalidade de cinza do fundo e outro centrado na tonalidade de cinza dos objetos. O vale entre os mesmos é povoado por um pequeno número de *pixels* devido a ruídos, sombras ou não uniformidade nas tonalidades do fundo e dos objetos. A área total do histograma é igual ao número de *pixels* na imagem.

A escolha apropriada de um limiar de nível de cinza L no meio do vale que separa os dois picos permite classificar e rotular os *pixels* em duas categorias: *pixels* de fundo (0) ou *pixels* de objetos (1), resultando em uma imagem binária $I_B(k,j)$ segmentada.

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 $I_B(k,j) = 1$.

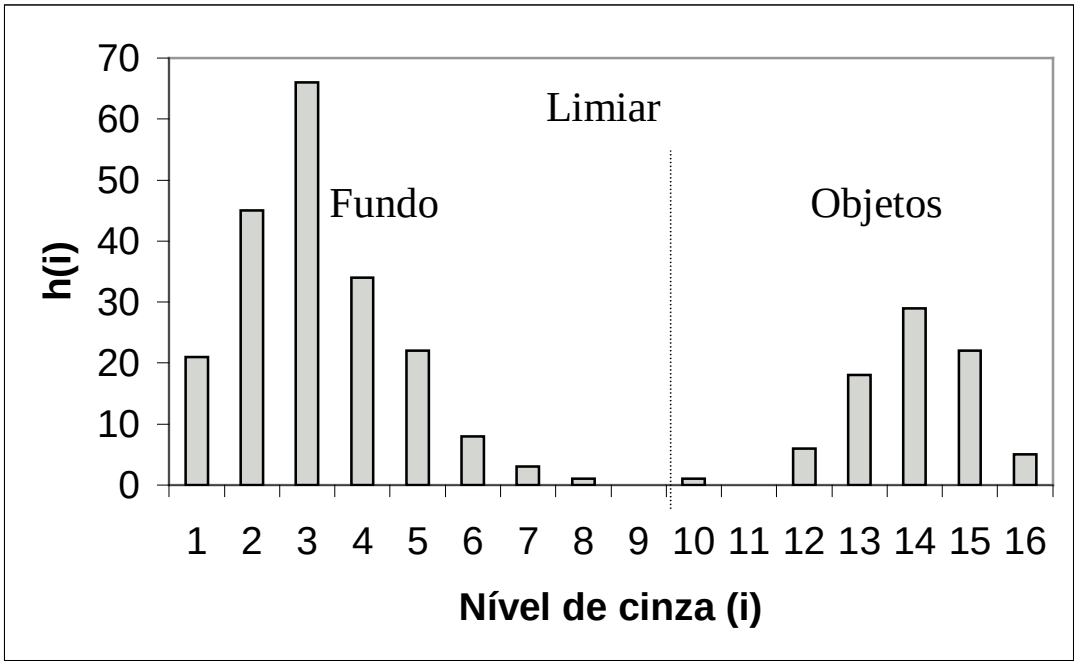

Fig. 7.7 – Limiarização de imagem com 16 níveis de cinza.

Rotulagem de Regiões:

Considere uma imagem binária I(k,j) segmentada em área de fundo e regiões conexas correspondentes a diversos objetos em cena. O processo de rotulagem consiste em produzir, a partir de I(k,j) uma imagem em que a cada região conexa foi atribuído um rótulo diferente. Uma maneira simples de implementar este processo é através de um algoritmo denominado crescimento de região, o qual, a partir de um *pixel* pertencente a uma dada região segmentada, "pinta" todos os *pixels* conexos ao mesmo. De acordo com este método, varre-se a imagem binária até encontrar um *pixel* pertencente a uma região conexa correspondente a um objeto em cena. Este *pixel* é utilizado como semente para crescimento dessa nova região conexa. O procedimento continua até que todos os *pixels* da imagem original forem rotulados.

Algoritmo de Rotulagem de Região

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 ≤ m, ir para o passo 2.
4. Fazer k = 1, j = j+1. Se j ≤ 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 ≤ m, ir para o passo 6.
8. Fazer k = 1, j = j+1. Se j ≤ n, ir para o passo 6.

Algoritmo de Crescimento de Região

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.

Observações:

- Para checar se o *pixel* (k,j) pertence à região R_i basta verificar se I(k,j) = i.
- Um *pixel* é 4-conectado a seus vizinhos se e somente se pelo menos um dos seus quatro *pixels* vizinhos (acima, abaixo, à direita ou à esquerda) possui o mesmo valor. As regiões rotuladas pelo algoritmo de Crescimento de Região são 4-conectadas. Linhas diagonais da largura de um *pixel* será rotulada como uma série de pequenas regiões.
- Um *pixel* é 8-conectado a seus vizinhos se e somente se 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.
- O algoritmo de rotulagem pode ser modificado para utilizar diferentes critérios: 4-conectividade para regiões correspondentes a objetos e 8-conectividade para regiões referentes ao fundo da imagem.

7.4. Descritores de Imagem

Para reconhecer objetos rotulados em imagens segmentadas torna-se necessário descrevê-los por meio de atributos que possam ser obtidos a partir da sua imagem. Objetos poligonais ou poliédricos podem ser descritos facilmente pela geometria dos seus vértices. Objetos de contornos mais complexos, com bordas curvas, precisam de outro tipo de descritores.

Descritores de Linha:

Medidas obtidas a partir dos *pixels* que constituem o perímetro limite de um objeto na imagem são denominadas Descritores de Linha. A maneira mais direta de representar uma linha é através de uma lista ordenada das coordenadas dos pontos da mesma. Este método, porém, não é muito eficiente. Uma representação mais compacta é o Código de Freeman (*Chain Code*, Código de Cadeia). Neste esquema usa-se a codificação de *pixels* vizinhos mostrada na figura abaixo.

3	2	1
4	2	0
5	6	7

Fig. 7.8 – Códigos de Freeman para *pixels* vizinhos.

Uma curva C(a) com n *pixels* de comprimento é representada como uma seqüência de códigos de Freeman a $\in \mathbf{R}^n$. A seqüência de códigos representa as mudanças incrementais de direção ao percorrer a curva *pixel* a *pixel*. Convencionaremos que a seqüência será gerada no sentido anti-horário a partir do *pixel* mais à direita (se acontecer empate entre dois ou mais *pixels* tomamos dentre estes o *pixel* mais acima). Esta representação é mais eficiente, uma vez que utiliza apenas três bits por ponto. Esta representação também é invariante a translações.

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]

Fig. 7.9 – Curva representada por Códigos de Freeman.

Considere uma curva C(a) representada pelos códigos de Freeman a $\in \mathbf{R}^n$. Dado um fragmento visível da curva C(b) representado pela seqüência de Códigos de Freeman b $\in \mathbf{R}^m$, com m < n, o seguinte índice de similaridade permite verificar se C(b) representa uma porção de C(a):

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

onde $0 \leq j \leq n\text{-}m$. Note que $-1 \leq \rho_j(a,b) \leq +1$. Como os ângulos utilizados são múltiplos de $\pi/4$, $\rho_j(a,b)$ pode ser calculado de forma eficiente usando valores tabelados, sem a necessidade de calcular os cosenos em tempo real. Avaliar o índice de similaridade na faixa $0 \leq j \leq n\text{-}m$ corresponde a deslizar o fragmento C(b) ao longo da curva C(a). A maior similaridade corresponde ao valor máximo de $\rho_j(a,b)$. Neste caso, j representa o ponto inicial de C(b) em C(a).

Descritores de Área:

Medidas obtidas a partir dos *pixels* enclausurados pelo perímetro limite de um objeto na imagem são denominadas Descritores de Área. Estes descritores tendem a ser mais robustos do que descritores de linhas, uma vez que mudanças grandes no limite de um objeto resultam geralmente em mudanças pequenas na área do mesmo.

Uma região R em uma imagem I(k,j) é dita conexa se para qualquer par de *pixels* pertencentes a esta região existe um caminho em R conectando o par (de acordo com alguma regra de conectividade, por exemplo, 4-vizinhança, 8-vizinhança).

Momentos:

Para uma dada região conexa R em uma imagem I(k,j) definem-se os Momentos de R de ordem k+j ($k \geq 0$, $j \geq 0$), $\{m_{kj}\}$, que caracterizam a sua forma:

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

Momentos de Baixa Ordem:

Os momentos de ordem mais baixa representam a área A e a posição do centróide (x_c,y_c) da região R:

$$A = m_{00} \qquad x_c = m_{10}/m_{00} \qquad y_c = m_{01}/m_{00}$$

Momentos Centrais:

Os momentos centrais μ_{kj} são calculados em relação ao centróide (x_c,y_c) da região, sendo assim invariantes a translações:

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

Os momentos centrais de segunda ordem têm significado físico claro: μ_{02} e μ_{20} são os momentos de inércia de R em relação aos eixos x e y através do centróide e μ_{11} é o produto de inércia de R.

Momentos Centrais Normalizados:

Normalizando os momentos centrais em relação à área de R, obtêm-se os momentos centrais normalizados, os quais são invariantes a mudanças de escala:

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

Ângulo Principal:

O ângulo principal de uma região R é uma medida da sua orientação e pode ser obtido a partir dos seus momentos de segunda ordem:

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

O ângulo principal pode ser utilizado para obter medidas invariantes à rotação de R. Fisicamente representa o ângulo de orientação do eixo passando pelo centróide de R que minimiza o seu momento de inércia.

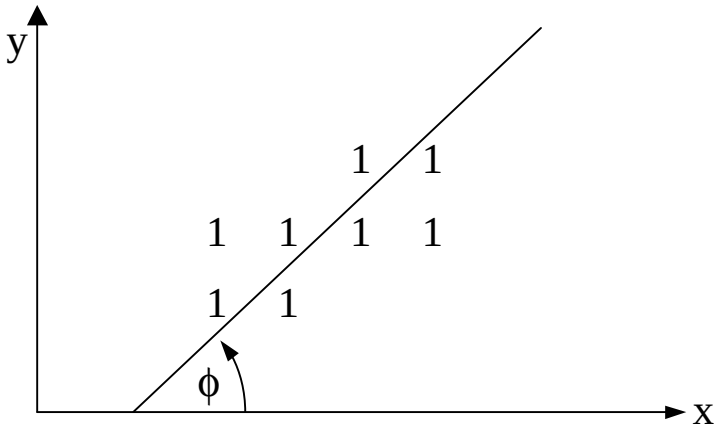

Fig. 7.10 – Ângulo Principal.

Dada uma região R com centróide (x_c,y_c) e ângulo principal ϕ, podemos transladar R para a origem (através de um vetor $(-x_c,-y_c)$ e girar R um ângulo $-\phi$. A região resultante possuirá centróide na origem e ângulo principal igual a zero. Os momentos normalizados desta região serão invariantes a translação, rotação e mudança de escala.

7.5. Casamento de Padrões

Uma das tarefas de um sistema de visão consiste em reconhecer se um objeto presente em uma imagem I(k,j) mxn é membro ou não de uma classe de objetos conhecidos. Uma maneira de implementar esta tarefa é por meio de Casamento de Padrões. Um padrão, ou máscara, é representado por uma imagem T(k,j) representativa de um objeto, onde $1 \leq k \leq m_o$ e $1 \leq j \leq n_o$. Para um conjunto de N classes diferentes de objetos devemos construir uma biblioteca BIB de padrões:

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

Para determinar se uma imagem I(k,j) contém um objeto pertencente à classe i, devemos procurar na mesma um casamento com o padrão $T_i(k,j)$, tipicamente menor do I(k,j). Para fazer isto, realiza-se uma varredura da imagem com o padrão e medir o grau de casamento de acordo com alguma métrica. A varredura é implementada realizando sistematicamente uma translação (x,y) do padrão $T_i(k,j)$ sobre a imagem I(k,j), o que corresponde a superpor o padrão sobre I(k+x,j+y). Para evitar ultrapassar os limites da imagem, $1 \leq x \leq m-m_o$, $1 \leq y \leq n-n_o$. Uma possível medida de casamento pode ser o índice de desempenho:

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

onde $1 \leq i \leq N$. Este índice é não negativo, sendo nulo se I(k,j) contém o padrão $T_i(k,j)$ transladado ao ponto (x,y). Neste caso, temos um casamento perfeito e (x,y) é a posição do padrão na imagem. Na prática, devido a ruídos presentes na imagem, dificilmente conseguiremos um casamento perfeito ($\rho_i(x,y) = 0$). Assim, considera-se que ocorreu casamento se $\rho_i(x,y) \leq \varepsilon$, onde ε é um limiar positivo que precisa ser cuidadosamente sintonizado.

Algoritmo de 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.

O índice de desempenho acima apresenta o problema de ser muito sensível a mudanças de iluminação. Mudanças na luminosidade da imagem pode fazer com que o padrão não seja reconhecido. Um bom casamento só ocorrerá se o padrão e o objeto correspondente na imagem possuírem as mesmas intensidades médias, definidas como:

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

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

Este problema pode ser contornado normalizando o índice de desempenho. A correlação Cruzada Normalizada é um índice de desempenho alternativo com esta característica:

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

Neste caso, quando ocorre o casamento, o índice é maximizado.

Detecção de Quinas:

Para uma imagem binária, a técnica de casamento de padrões pode ser utilizada para detectar quinas. Neste caso, varre-se a imagem com um conjunto de oito padrões de pontos de quina 3x3, os quais representam todos os possíveis pontos de quina em objetos com pelo menos dois *pixels* de largura, com a quina localizada no *pixel* central da máscara. Cada padrão é gerado por rotações múltiplas de 45^o do primeiro padrão.

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

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

Fig. 7.11 – Máscaras para detecção de padrões de pontos de quina.

O procedimento de busca pode ser consideravelmente acelerado se os (m-2)(n-2) *pixels* interiores da imagem forem varridos para buscar *pixels* candidatos a ponto de quina (com valor igual a 1) e, somente neste caso, for calculada a correlação cruzada normalizada com a máscara centralizada no *pixel* candidato.

7.6. Calibração de Câmera

Para relacionar atributos obtidos de imagens de objetos, normalmente expressas em unidades baseadas em *pixels*, com as dimensões reais (em metros) desses objetos, é necessário obter transformações apropriadas que dependem dos parâmetros da câmera que capturou a imagem. Calibração de câmera é o processo de determinar estes parâmetros.

Transformação de Perspectiva:

Considere uma câmera capturando uma imagem de um objeto situado em frente da mesma. A origem do sistema de coordenadas fixo na câmera C é localizada no plano da imagem e o seu eixo ${}^{C}z$ é alinhado com o eixo ótico da lente da câmera. A distância f entre a lente e o plano da imagem é denominada distância focal efetiva. Dado um ponto p de um objeto visualizado pela câmera, ${}^{C}p$ são as coordenadas de p em referencial de câmera. Da mesma forma, ${}^{i}p$ são as coordenadas da imagem de p também expressas em referencial de câmera.

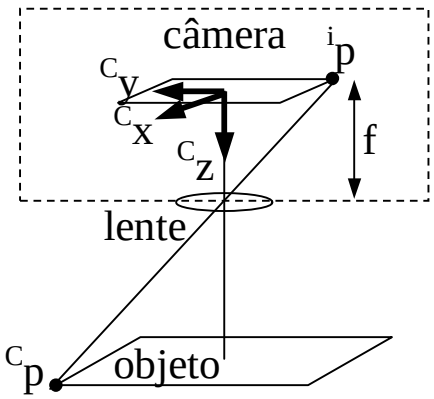

Fig. 7.12 – Geometria de formação de imagem em câmera.

As coordenadas de ${}^{i}p$ em termos de ${}^{C}p$ podem ser obtidas por simples relações geométricas:

$$-{}^{i}p_x/f = {}^{C}p_x/({}^{C}p_z - f) \qquad -{}^{i}p_y/f = {}^{C}p_y/({}^{C}p_z - f) \qquad {}^{i}p_z = 0$$

Assim:

$${}^{i}p = [(-f.{}^{C}p_x/({}^{C}p_z - f)), \quad (-f.{}^{C}p_x/({}^{C}p_z - f)), \quad 0]^T$$

A transformação de ${}^{C}p$ para ${}^{i}p$ é um mapeamento não linear de $\mathbf{R^3}$ para $\mathbf{R^3}$ (envolve divisão por ${}^{C}p_z$). Este mapeamento é denominado Transformação de Perspectiva. Devido a ser não linear, a transformação de perspectiva não pode ser representada por uma matriz de transformação 3x3. Entretanto, é possível representar a transformação usando um operador matricial em um espaço de maior dimensão. Isto pode ser realizado representando os pontos em coordenadas homogêneas:

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

onde e é um fator de escala não nulo, de tal forma que:

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

Desta forma, a transformação de perspectiva pode ser representada pelo seguinte operador matricial:

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

tal que: ${}^{i}p_h = {}^{i}T_C.{}^{C}p_h$

Daqui em diante omitiremos o índice h para denotar coordenadas homogêneas, ficando subentendido dentro do contexto se trata-se de vetores em $\mathbf{R^3}$ ou em $\mathbf{R^4}$.

Transformação de Perspectiva Inversa:

Em aplicações práticas desejamos obter a posição de objetos em relação à câmera a partir de imagens dos mesmos. Ou seja precisamos calcular a transformação de perspectiva inversa, ${}^{C}T_i$. O problema é que a transformação de perspectiva é uma matriz singular, ou seja, não pode ser invertida diretamente. Isto é esperado, pois retrata o fato desta transformação projetar informação de um objeto 3D em uma imagem 2D, portanto perdendo informação neste processo. Para poder mapear pontos da imagem novamente para o espaço tridimensional é necessário recuperar de alguma forma a informação sobre a profundidade da cena. Assumindo que a coordenada ${}^{C}p_z$ é conhecida a priori, ou que a mesma pode ser obtida através de medição direta, então teremos a informação de profundidade disponível. Este é o caso, por exemplo, se o objeto está sobre a superfície de uma esteira sendo focalizado por uma câmera situada a uma altura conhecida da mesma. Com esta informação adicional é possível realizar o mapeamento inverso, construindo um ponto de imagem aumentado ${}^{ia}p$ a partir do ponto de imagem ${}^{i}p$, acrescentando a este informação de profundidade:

$${}^{ia}p = {}^{i}p - [{}^{C}p_z/({}^{C}p_z - f)]I^3$$

onde I^3 representa a terceira coluna da matriz identidade 4x4. Este artifício pode ser entendido como elevar a imagem do seu plano usando a informação de profundidade conhecida. Isto pode ser representado por uma transformação homogênea de translação ao longo do vetor $[{}^{C}p_z/(f-{}^{C}p_z)]I^3$:

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

O vetor de imagem aumentada pode ser transformado para coordenadas de câmera. Assim, agora é possível transformar o vetor de imagem para o vetor de imagem aumentado e depois para coordenadas de câmera através de transformações homogêneas, o que resulta na seguinte representação para a transformada de perspectiva inversa:

$$ {}^{C}T_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f & f.{}^{C}p_z/(f-{}^{C}p_z) \\ 0 & 0 & 1 & f/(f-{}^{C}p_z) \end{bmatrix} $$

Pode-se verificar facilmente que ${}^{C}p = {}^{C}T_i.{}^{i}p$.

Coordenadas de *Pixel*:

Considere que a câmera possui um matriz de mxn elementos sensores, cada *pixel* com largura Δx e altura Δy. Considere que a localização do referencial fixo na câmera tem a sua origem no centro da matriz de sensores e rotacionado 180^o em torno do eixo z em relação ao sistema de coordenadas em *pixel*.

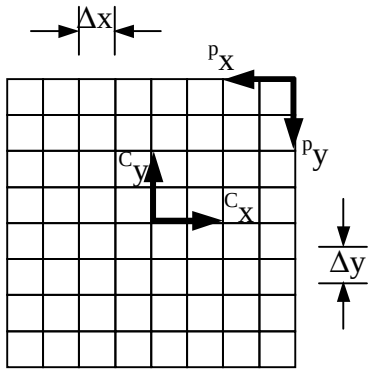

Fig. 7.13 – Sistemas de referência em uma matriz de mxn = 8x8 elementos sensores.

Então, a transformação que mapeia pontos em coordenadas de *pixel* ${}^{p}p$ para pontos em coordenadas de imagem ${}^{i}p$ é:

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

Assim, dado um objeto, se quisermos converter as coordenadas em *pixels* da imagem do mesmo para um referencial {B} na base de um manipulador, devemos realizar a seguinte composição de transformações homogêneas:

${}^{B}p = {}^{B}T_C.{}^{C}T_i.{}^{i}T_p.{}^{p}p$

Calibração de Câmera:

Considere o problema simples de calibração de câmera, onde a orientação da mesma supõe-se conhecida e deseja-se determinar a sua posição (x_0, y_0, z_0) em relação ao referencial de base do manipulador. A câmera aponta para baixo e os objetos estão

sobre o plano de trabalho. Considere que a localização da câmera em relação à base pode ser expressa pela seguinte transformação homogênea:

$$ {}^{B}T_{C} = \begin{bmatrix} 0 & -1 & 0 & x_0 \\ -1 & 0 & 0 & y_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

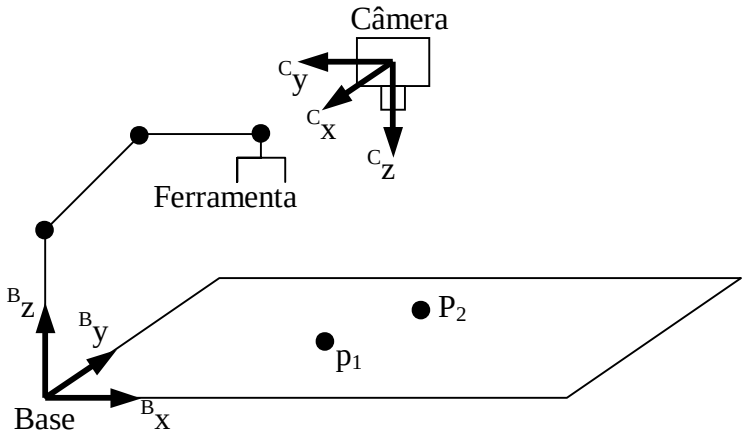

Fig. 7.14 – Pontos para Calibração de Câmera.

Dados dois pontos conhecidos em referencial de base, ${}^{B}p_1$ e ${}^{B}p_2$, a imagem dos mesmos, ${}^{i}p_1$ e ${}^{i}p_2$ pode ser capturada através da câmera, de tal forma que:

$${}^{i}p_1 = {}^{i}T_{B}.{}^{B}p_1 \qquad {}^{i}p_2 = {}^{i}T_{B}.{}^{B}p_2$$

onde,

$$ {}^{i}T_{B} = {}^{i}T_{C}.({}^{B}T_{C})^{-1} = \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & (f-z_0)/f \end{bmatrix} $$

Das equações acima obtemos quatro equações em função de (x_0,y_0,z_0):

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

Estas quatro equações podem ser resolvidas para (x_0,y_0,z_0), obtendo z_0 a subtraindo a terceira da primeira equação e depois resolvendo a primeira e a segunda equações em função de z_0:

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

Esta solução é válida para ${}^{i}p_{1x} \neq {}^{i}p_{2x}$. Quando ${}^{i}p_{1x} = {}^{i}p_{2x}$, uma solução alternativa pode ser derivada subtraindo a quarta da segunda equação para obter z_0:

$$z_0 = f[1 + ({}^{B}p_{1x} - {}^{B}p_{2x})/({}^{i}p_{1y} - {}^{i}p_{2y})]$$

Desde que os pontos sejam diferentes, sempre pelo menos uma das duas soluções será factível.