4
Algoritmos do sistema *multitouch*

4.1
Calibração da câmera

A calibração da filmadora foi feita para tornar possível o seu uso como sensor e a conseqüente determinação da posição de pontos capturados em cada imagem.

O procedimento de calibração envolveu três etapas para a estimativa dos parâmetros para o funcionamento da filmadora, ou seja: (a) Recuperação das coordenadas dos pixels de interesse presentes na imagem; (b) As coordenadas do ponto correspondente no sistema de coordenadas da tela; e (c) Utilização de uma heurística de minimização para determinar os parâmetros da câmera utilizando as informações relativas aos itens “a” e “b”. Esses pontos serão discutidos a seguir.

4.1.1
Projeção perspectiva

A Figura 4.1 mostra um modelo simplificado de uma câmera fotográfica baseada no modelo *pinhole*. O plano onde a imagem é formada, representada pelos eixos x_{im}, y_{im}, é espelhado em relação ao plano que contém o *pinhole* O_c correspondente ao centro óptico, que fica a uma distância focal f do plano de formação da imagem. Dessa forma, a imagem formada não é invertida e ela fica localizada entre os pontos do mundo P e o centro óptico O_c.

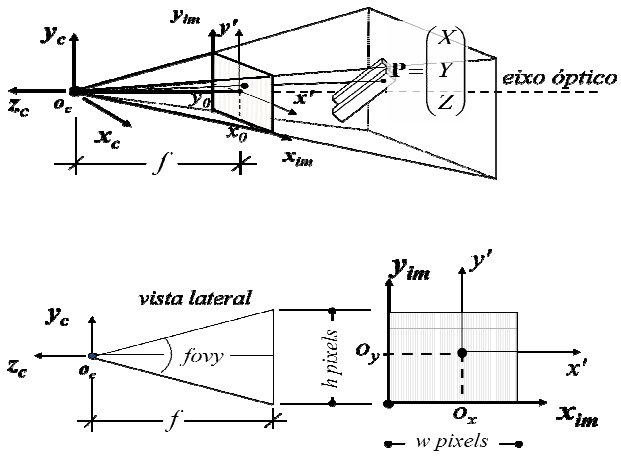

Figura 4.1 – Modelo de uma câmera fotográfica utilizada na modelagem das equações da projeção perspectiva.

A relação, em coordenadas homogêneas, de um ponto no sistema do mundo M (X,Y,Z,1) para sua coordenada (u,v), projetada na imagem através das matrizes A e [R t], é dada pela Equação 4.1.

Na notação utilizada na Equação 4.1, a matriz A armazena os 4 parâmetros intrínsecos da câmera que fornecem suas características ópticas e geométricas, sendo elas: a dimensão dos pixels (escalonamentos vertical f_y e horizontal f_x) e a posição do ponto principal (o_x, o_y) formado pela interseção do eixo óptico com o plano da imagem. E a matriz [R t] armazena os 6 parâmetros extrínsecos, que fornecem informações de posição e orientação da câmera em relação ao sistema de coordenadas do objeto capturado. Este sistema consiste em uma transformação rígida do modelo formado por três parâmetros de rotação, um para cada coluna da matriz R, e três parâmetros de translação T_x, T_y, T_z.

$$x = \frac{u}{s} \qquad y = \frac{v}{s}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -f_x & 0 & o_x \\ 0 & -f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (4.1)$$

$$s\tilde{\mathbf{m}} = \mathbf{A}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\tilde{\mathbf{M}}$$

4.1.2
Calibração de câmera como minimização

A calibração da câmera foi embasada nas informações de um conjunto de pontos p_i presentes na imagem capturada nos experimentos e seus correspondentes P_i, na cena. O processo de minimização da calibração da câmera foi calculado pela Equação 4.2 [32].

$$\min_{A,R,t} \sum \left(pi - \mathcal{F}_{A,R,t}(P_i) \right)^2 \qquad (4.2)$$

pontos da cena

projeção (função não linear)

pontos da imagem

Alguns métodos para encontrar as soluções das matrizes A e [R t] têm sido propostos na literatura científica [11,2]. Neste trabalho, uma matriz de homografia [14] foi aplicado para a determinação dos parâmetros pertinentes envolvidos nos experimentos.

4.1.3
Homografia

As medidas apresentadas neste trabalho foram feitas no caso particular em que a cena é planar, ou seja, todos os pontos de interesse capturados pela câmera estão contidos em um plano formado pela placa de acrílico onde a imagem é projetada. Dessa forma, os pontos P no sistema de coordenadas do mundo são simplificados e calculados pela Equação 4.3:

$$P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{4.3}$$

Como o valor de Z é fixo e igual a zero, a equação que descreve a transformação dos pontos no sistema de coordenadas do mundo para o plano de projeção da câmera pode ser simplificada, eliminando-se, dessa forma, a terceira coluna da matriz R:

$$x = \frac{u}{s} \qquad y = \frac{v}{s}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -f_x & 0 & o_x \\ 0 & -f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & T_x \\ r_{21} & r_{22} & T_y \\ r_{31} & r_{32} & T_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{4.4}$$

$$s\tilde{\mathbf{m}} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\tilde{\mathbf{M}}$$

Efetuando o produto A[R t], obtém-se uma matriz genérica [h]:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.5}$$

O que resulta na Equação 4.6, que descreve a projeção de cada ponto com coordenada u,v no plano da imagem para a coordenada x,y no plano formado pelo acrílico:

$$u = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad v = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{4.6}$$

A relação entre a posição de um ponto em um cenário planar e sua posição na imagem é dada por uma matriz h_{3x3} denominada homografia. A homografia é constante e engloba tanto os parâmetros intrínsecos como os extrínsecos da câmera. A calibração da câmera nesse tipo de situação depende unicamente do cálculo da homografia H.

4.1.4
Cálculo da homografia

Cada correspondência entre as coordenadas de um ponto no sistema do mundo e a sua projeção no sistema de imagem da câmera é calculada pela Equação 4.7:

$$\begin{cases} u_i x_i . h_{31} + u_i y_i . h_{32} + u_i = x_i . h_{11} + y_i . h_{12} + h_{13} \\ v_{ix_i} . h_{31} + v_{iy_i} . h_{32} + v_i = x_i . h_{21} + y_i . h_{22} + h_{23} \end{cases} \quad (4.7)$$

Para um conjunto de n pontos cujas coordenadas são conhecidas, um conjunto de 2n equações é definido e representado matricialmente pela Equação 4.8 [2]:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1 x_1 & -u_1 y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1 x_1 & -v_1 y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -u_2 x_2 & -u_2 y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -v_2 x_2 & -v_2 y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -u_n x_n & -u_n y_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -v_n x_n & -v_n y_n \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \\ \\ u_n \\ v_n \end{pmatrix} \Leftrightarrow Ax = b \quad (4.8)$$

A matriz da homografia multiplicada por um fator de escala também constitui uma solução do sistema da Equação 4.8. Uma solução para que a homografia tenha um valor determinado consiste na fixação de h_{33}=1 [28].

O sistema matricial representado pela Equação 4.8 tem 2n equações e oito incógnitas. Isso não permite o cálculo da homografia com um número de pontos de correspondência igual ou menor do que três. Com quatro pontos, os elementos da matriz H têm solução única determinada pela Equação 4.9:

$$x = A^{-1} b \quad (4.9)$$

Com um número maior de correspondências, o sistema matricial pode ser resolvido de forma a minimizar o erro quadrático médio, através da pseudo-inversa dada pela Equação 4.10 [32]:

$$x = (A^T A)^{-1} A^T b \tag{4.10}$$

Erros decorrentes das medidas e das coordenadas dos pontos filmados, na projeção de imagens capturadas pela filmadora, introduzem ruídos no sistema calculado pela Equação 4.10. De acordo com Medeiros [28], os resultados da minimização utilizada para encontrar a matriz h são melhorados quando um número maior de pontos é utilizado para a calibração da câmera.

4.2
Rastreamento dos toques

Um programa computacional, utilizando técnicas de visão computacional e de comunicação entre processos, foi desenvolvido com as seguintes finalidades:

- Interpretar as imagens capturadas pela filmadora;
- Separar e calcular as coordenadas de cada toque ou pressão feita na tela do sistema *multitouch*;
- Registrar o movimento de toques feitos em um ponto na superfície da tela, e
- Transmitir essa informação para outros programas ativos no computador que irão aplicá-los para fornecer algum tipo de interatividade com o usuário.

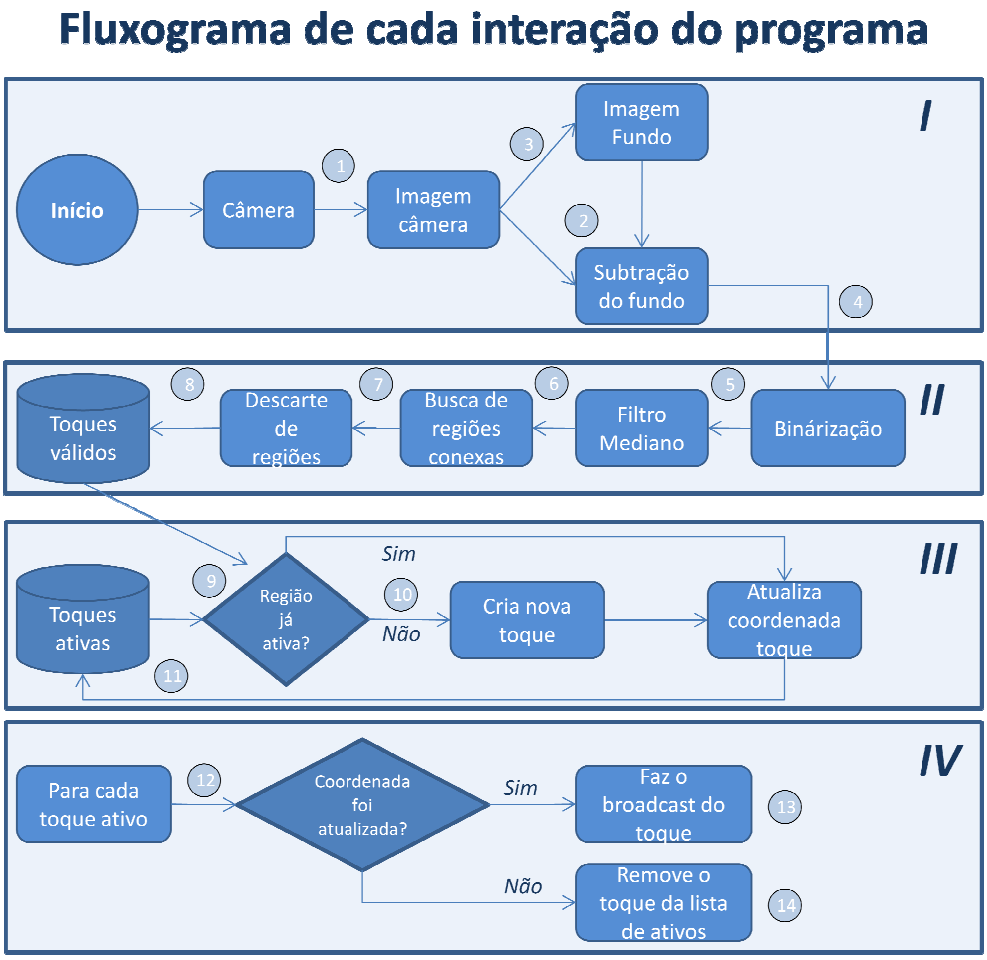

Figura 4.2 – Processo de interação nas quatro etapas de funcionamento do programa de reconhecimento.

O funcionamento desse programa é constituído de quatro módulos. As etapas (I, II, III e IV) de funcionamento do programa, mostradas na Figura 4.2, compreendem: subtração da luminosidade de fundo, identificação de regiões conexas, rastro (*tracking)* de regiões, e envio (*broadcast)* de regiões. O funcionamento de cada uma das etapas será descrito nas próximas seções.

4.2.1
Subtração da luminosidade de fundo

Uma fração da luz emitida pelos LEDs, que opera na região do infravermelho, é difundida pela camada de parafina e pelo material utilizado para a projeção da imagem, que estão sobrepostos ao acrílico e que são os constituintes da tela do sistema *multitouch*. Tal radiação de fundo associada ao componente de infravermelho

oriundo da luz ambiente faz com que a imagem capturada pela câmera tenha um comportamento de fundo que tende a se manter constante no tempo. Para auxiliar o processo de segmentação de imagem, um algoritmo foi utilizado para subtrair a luminosidade de fundo. O processo para a eliminação da luminosidade e do ruído de fundo, que é a primeira parte do programa de reconhecimento, é ilustrada na Figura 4.3.

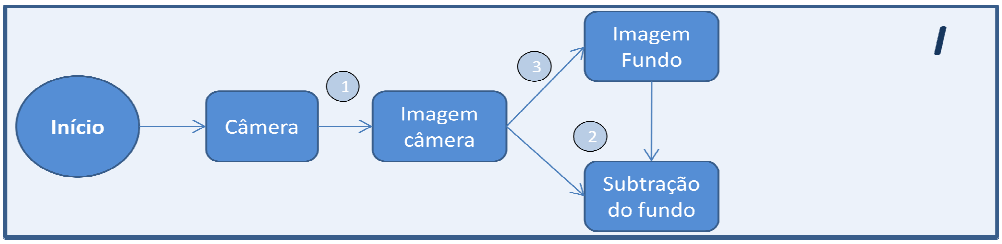

Figura 4.3 – Bloco do fluxograma relativo à parte de subtração da radiação de fundo. Primeira etapa do programa de reconhecimento.

4.2.1.1 Luminosidade para tons de cinza

Na primeira etapa do programa, a subtração da luminosidade de fundo foi feita mediante a conversão da imagem capturada pela câmera, formada por 720x480 pixels contendo 8 bits para cada canal de cor (R,G,B), em uma imagem em tons de cinza. Isso é feito calculando-se a média das intensidades de luz de cada componente de cor de cada um dos pixels R_i G_i, e B_i da imagem capturada pela câmera, ou seja:

$$C_i = (R_i + G_i + B_i)/(3 * 255) \qquad (4.11)$$

C_i é o pixel da imagem em tons de cinza que será armazenada em um vetor de pontos flutuantes de 32 bits.

4.2.1.2 Cálculo da imagem sem ruído de fundo

A segunda fase do programa utiliza um modelo pré-computado com o fundo atual para calcular a imagem com fundo subtraída. Para tal, cada pixel C_i da imagem em tons de cinza é subtraído o pixel F_i correspondente ao modelo de fundo, também

formado por um vetor de pontos flutuantes de 32 bits, fornecendo a intensidade do pixel Lum:

$$Lum = C_i - F_i \quad (4.12)$$

O valor de Lum é limitado ao intervalo de 0 a 1 antes de ser armazenado no pixel S_i da imagem com fundo subtraído, formada por um vetor de pontos flutuantes de 32 bits.

$$S_i = Min(1, Max(Lum, 0) \quad (4.13)$$

4.2.1.3 Atualização do modelo de fundo

Após a subtração do fundo da imagem capturada pela câmera, a imagem com o modelo de fundo é atualizada pixel a pixel para ser utilizada na próxima interação do programa:

$$F_i = F_i(1-\propto) + C_i \propto \quad (4.14)$$

F_i corresponde aos pixels da imagem de fundo e C_i aos pixels da imagem da câmera em tons de cinza. A constante α representa o fator de contribuição da imagem atual para o modelo de fundo. Qualquer valor entre 0 e 1 pode ser atribuído a essa constante. Com valor igual a 0 a imagem não será incorporada ao modelo de fundo; com valor 1, o modelo de fundo será substituído pela última imagem.

4.3 Identificação de regiões conexas

Obtida a imagem em tons de cinza fornecida pelo algoritmo de subtração da imagem de fundo, a próxima etapa do programa faz uso de técnicas de visão computacional para identificar agrupamentos de pixels presentes na imagem.

Nesta etapa do programa é calculado o número de pixels em cada região encontrada na imagem, assim como a coordenada média de todos os pixels de cada região. Caso o número de pixels existentes em uma determinada região esteja fora de uma faixa pré-estabelecida, a região é descartada. No final do processamento, uma lista é gerada contendo, para cada região encontrada, a coordenada média dos pixels

que formam a região e o número de pixels da região. Um fluxograma da segunda etapa do programa é mostrado na Figura 4.4.

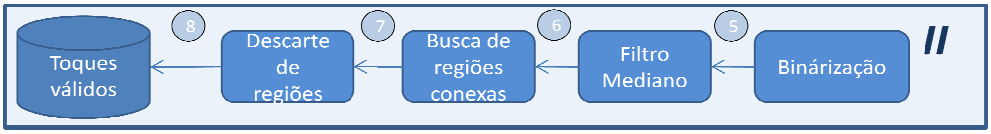

Figura 4.4 – Bloco do fluxograma relativo à parte do algoritmo de identificação de regiões do programa.

4.3.1
Binarização da imagem

No processo de binarização de cada pixel, S_i, da imagem processada na etapa de subtração da luminosidade de fundo, o pixel é comparado a um valor de referência constante, β. Se o valor do pixel analisado for maior do que o valor de β, o valor S_i é unitário. Se o valor de Si for menor ou igual ao valor de referência β, S_i assume o valor zero.

4.3.2
Filtro da mediana

A binarização da imagem introduz ruídos de alta freqüência nos pixels da imagem binarizada. Isso gera regiões na imagem com um conjunto de pixels esparsos. Para evitar que essas regiões sejam identificadas pela próxima etapa de identificação de regiões, usa-se um filtro conhecido na computação gráfica como filtro da mediana [10].

O filtro da mediana é um filtro de convolução que é aplicado em cada pixel da imagem. A Figura 4.5 ilustra esse processo. Para cada pixel da imagem, os 8 pixels vizinhos, assim como o valor do pixel central, são consultados e armazenados em uma lista de tamanho 9. A seguir, essa lista é ordenada e o valor central da lista é utilizado como o novo valor do pixel.

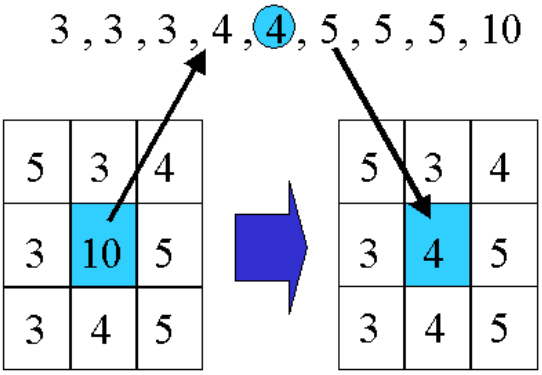

Figura 4.5 – Ilustração do resultado da aplicação do filtro da mediana em um pixel de uma imagem.

Como a imagem utilizada é binária, para evitar longos períodos de tempo computacional no processo de ordenação, o valor da mediana é determinado contando-se o número de pixels com valor igual a 0 existentes dentro da janela de busca. Se esse número for maior ou igual a 5, o valor 0 é atribuído como resultado da mediana, caso contrário o pixel resultante terá valor igual a 1.

4.3.3
Busca de regiões conexas

Após a binarização da imagem, a próxima etapa consiste em encontrar regiões conexas com valor 1. Isso é feito com uma busca recursiva para identificar as regiões conexas.

Inicialmente uma tabela com o mesmo tamanho da imagem (largura igual a 720 e altura igual a 480) é inicializada para indicar que nenhum pixel foi “visitado”. A seguir a imagem é varrida, pixel a pixel, procurando por algum pixel ainda não visitado que tenha valor na imagem binária igual a 1. Quando um pixel é encontrado, ele é marcado como visitado na tabela e suas coordenadas são acumuladas em uma nova região. Uma busca recursiva à procura de pixels conexos é inicializada novamente a partir das coordenadas do pixel atual.

No processo de recursão, utilizando-se uma sub-rotina do programa chamada de “Busca de Regiões Conexas”, é feita uma busca nos 8 vizinhos do pixel de origem. Essa sub-rotina verifica se o pixel não foi marcado como visitado e se ele assume um valor na imagem igual a 1. Caso positivo, suas coordenadas são acumuladas na região atual e outra chamada recursiva é feita. Essa sub-rotina escrita em linguagem C++ está detalhada no Anexo A.

Como o tamanho médio das regiões de interesse na imagem é conhecido *a priori*, pode-se limitar a profundidade da busca recursiva, melhorando o desempenho do programa e evitando o estouro da pilha de execução do mesmo.

4.3.4
Descarte de regiões

Mesmo após a passagem da imagem binária pelo filtro da mediana, ela ainda possui regiões conexas formadas por um número pequeno de pixels que ainda formam ruídos na imagem. Isso significa que podem existir regiões na imagem com um número pequeno de pixels que devem ser descartados pelo programa, assim como regiões formadas por um número grande de pixels que podem ser geradas por variações bruscas na iluminação de fundo (por exemplo, um flash fotográfico).

O descarte dessas regiões, determinado através da comparação da quantidade de pixels no processo de varredura, é feito pela sub-rotina computacional “Descarte de Regiões com Número de Pixels Inválidos”, apresentada no Anexo B.

A análise da quantidade de pixels feita por essa sub-rotina verifica se o número de pixels encontrados na busca recursiva está dentro de um intervalo pré-definido; neste caso as coordenadas são normalizadas e o contador do número de regiões encontradas no processo de busca é incrementado, validando assim a região. Caso o número de pixels encontrados esteja fora do valor pré-definido, o contador do número de regiões não é incrementado. Dessa forma, tais regiões não são computadas e os valores armazenados nelas serão sobrescritos pela próxima região encontrada.

Ao término desta etapa, pode-se verificar se o número de regiões encontradas é maior do que um limite pré-estabelecido; neste caso a imagem pode ser descartada. Isto serviria para limitar em 40 regiões uma superfície projetada para no máximo quatro usuários simultâneos.

4.4
Tracking de regiões

Para cada imagem capturada pela filmadora um conjunto de regiões diferentes é detectado. Portanto, o programa verifica antes de criar uma nova região se as coordenadas da região encontrada são suficientemente próximas de alguma região já identificada em uma imagem anterior, indicando que a região apenas sofreu um

deslocamento. Um fluxograma desse processo de rastreamento (*tracking*) é apresentado na Figura 4.6.

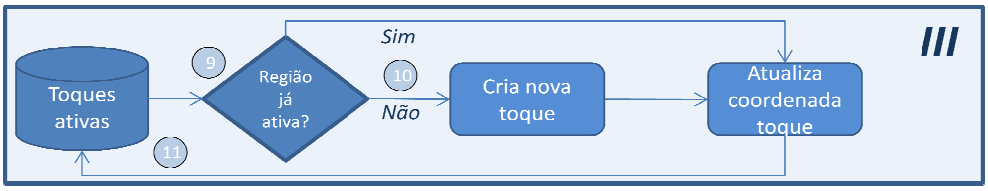

Figura 4.6 – Bloco do fluxograma relativo à parte de *tracking* de regiões do programa.

Dois métodos foram testados para identificar a correspondência entre regiões já ativas no sistema e as identificadas na imagem capturada pela filmadora.

No primeiro método, as coordenadas R (x,y) de cada região foram comparadas às últimas coordenadas P_{n-1} de cada uma das regiões ativas. A distância D entre essas regiões foi calculada pelos passos apresentados na Figura 4.7:

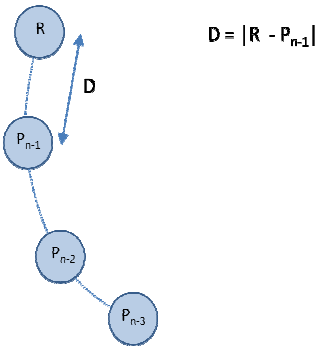

Figura 4.7 – Cálculo da distância D entre uma região nova R e a última coordenada P_{n-1} de uma das regiões ativas.

No segundo método, para cada região ativa foi feita uma estimativa da posição no quadro atual P_{EST}, que é calculado extrapolando linearmente a posição da região nas últimas duas imagens registradas pela filmadora. Esse valor estimado é utilizado para calcular a distância D_{EST} entre as coordenadas da região por comparação entre o valor de R (x,y) e o de cada uma das regiões ativas, como pode ser visto na Figura 4.8.

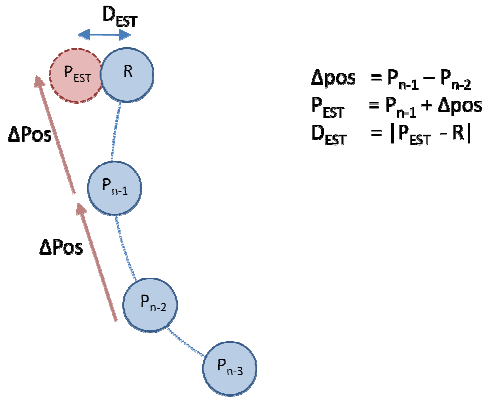

Figura 4.8 – Cálculo da distância D_{EST} entre uma região nova R e a estimativa P_{EST} da próxima coordenada de uma das regiões ativas.

Em ambos os métodos, se a distância calculada D_{EST} for menor do que um valor pré-estabelecido, as coordenadas R da região na imagem atual são consideradas um deslocamento da região com a qual ela foi comparada. Se isso ocorre, as coordenadas da região analisada são atualizadas com o valor de R. Caso contrário, uma nova região é então demarcada pelo programa.

4.5
Broadcast de regiões

O programa de reconhecimento identifica e calcula as coordenadas dos pontos capturados pela câmera. Essas informações são utilizadas por outros programas para possibilitar a interação da máquina com o usuário. Um fluxograma desse processo é apresentado na Figura 4.9.

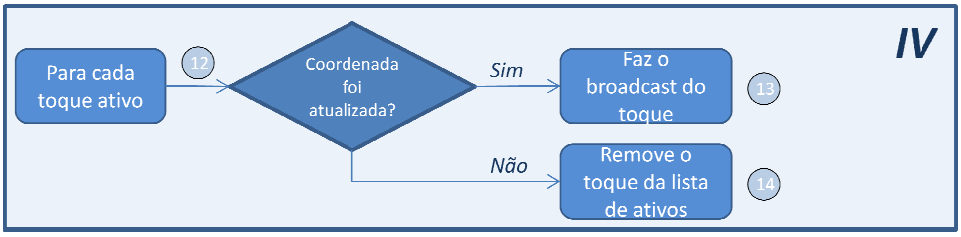

Figura 4.9 – Bloco do fluxograma relativo ao *broadcast* das regiões encontradas.

As bibliotecas Touchlib [13] e ReactiVision [17], para desenvolvimento de aplicativos *multitouch* distribuídos pela licença "*general public license*", utilizam o protocolo Open Sound Control. Esse protocolo faz a comunicação entre os programas, através de *sockets* utilizando pacotes UDP.

Neste trabalho, uma nova abordagem foi utilizada. A troca de mensagens entre o programa de reconhecimento e os aplicativos *multitouch* foi feita utilizando a API de mensagens do sistema operacional Windows.

Essa abordagem é idêntica àquela utilizada pelo sistema operacional para enviar para um aplicativo uma mensagem quando uma tecla do teclado é pressionada, o mouse é movimentado ou qualquer outro tipo de evento do qual um aplicativo deve ser notificado pelo sistema operacional.

Dois tipos de mensagens foram empregados para a comunicação entre o programa de reconhecimento e os aplicativos desenvolvidos para utilizar os recursos *multitouch*. A primeira mensagem, chamada “MSG_FTIR_HIT”, informa aos programas que fazem uso da informação *multitouch* a coordenada e o identificador único de uma região na tela. A segunda mensagem, chamada “MSG_FTIR_HIT_FIM”, informa que uma determinada região especificada pelo seu identificador único foi desativada ou descartada. Para assegurar o processo de envio e recebimento dessas mensagens entre os aplicativos, as mensagens são inicialmente registradas no sistema através das funções:

```
UINT MSG_FTIR_HIT      = RegisterWindowMessage("MSG_FTIR_HIT");

UINT MSG_FTIR_HIT_FIM = RegisterWindowMessage("MSG_FTIR_HIT_FIM");
```

A função “RegisterWindowMessage” recebe uma palavra-chave como parâmetro e retorna um código com um identificador único em todo o sistema operacional para uma mensagem com esse texto. Esta etapa é feita uma única vez, quando o programa de reconhecimento ou aplicativos desenvolvidos para *multitouch* são inicializados.

A cada interação do programa e para cada região identificada pelo programa de reconhecimento, uma mensagem do tipo MSG_FTIR_HIT é enviada utilizando o método PostMessage:

```
BOOL PostMessage(
    HWND hwnd,

    UINT Msg,

    WPARAM wParam,
```

```
    LPARAM lParam};
```

O parâmetro HWND é utilizado para fornecer um *handler* para o elemento de interface que irá receber a mensagem. O Msg recebe um identificador com o código da mensagem a ser enviada. Já os parâmetros wPARAM e lPARAM são inteiros, de 32 bits, sem sinal, que podem opcionalmente armazenar parâmetros adicionais da mensagem [27].

No programa de reconhecimento, o método denominado "enviaMensagemCoordenada" é utilizado para codificar e enviar as coordenadas de cada região ativa e identificador único da região:

```
void enviaMensagemCoordenada(int x, int y, int codHit)

{

  unsigned int coord = x + (y << 16);

  PostMessage (HWND_BROADCAST, MSG_FTIR_HIT, coord, codHit);

}
```

O parâmetro wPARAM é utilizado para armazenar as coordenadas da tela onde a região que está sendo transmitida se localiza. Para que isso ocorra, as coordenadas (x,y) foram reduzidas de 32 bits para 16 bits e codificadas de forma que os primeiros 16 bits da variável wPARAM armazene a coordenada x e os últimos 16 bits, a coordenada y da região na tela. O identificador da região foi codificado utilizando os 32 bits do lPARAM.

O parâmetro HWND_BROADCAST foi utilizado para assegurar que a mensagem MSG_FTIR_HIT seja inserida na lista de mensagens de todas as janelas existentes no sistema operacional. Os aplicativos que não são programados para tratar essa mensagem irão descartá-la automaticamente.

Se no processo de interação computado no ciclo de *tracking* de regiões uma das regiões não for atualizada com uma nova coordenada, uma mensagem de término MSG_FTIR_HIT_FIM é enviada para informar aos demais aplicativos que a região foi extinta, ou seja:

```
Void enviaMensagemTermino (int codHit)

{

  PostMessage (HWND_BROADCAST, MSG_FTIR_HIT_FIM, codHit, 0);

}
```

Nessa mensagem, o identificador de região é codificado na variável wPARAM e a variável lPARAM, que não será utilizada, é preenchida com “0”. O parâmetro HWND_BROADCAST é novamente acionado a fim de que a mensagem seja enviada para todas as janelas do sistema operacional.

4.6
Programas *multitouch*

Nos programas utilizados nos sistemas *multitouch,* para o aplicativo receber e analisar as mensagens enviadas pelo programa de reconhecimento, os códigos das mensagens utilizadas para troca de informações entre os aplicativos são criados pelo próprio sistema operacional. Esse processo é feito utilizando-se o método RegisterWindowMessage, usando como chave as mesmas palavras-chaves utilizadas pelo programa de reconhecimento: MSG_FTIR_HIT e MSG_FTIR_HIT_FIM.

```
UINT MSG_FTIR_HIT      = RegisterWindowMessage("MSG_FTIR_HIT");

UINT MSG_FTIR_HIT_FIM = RegisterWindowMessage("MSG_FTIR_HIT_FIM");
```

Dessa forma, o sistema operacional assegura que o código retornado por uma chamada seja idêntico ao código utilizado pelo programa de reconhecimento para o envio de mensagens.

Para que os programas *multitouch* sejam capazes de tratar as mensagens enviadas pelo programa de reconhecimento, eles devem interceptá-las através de sua função responsável pelo recebimento das mensagens do sistema operacional.

No padrão de interface de janelas utilizado pelo sistema operacional Windows, uma janela define a função WindowProc [27]. Nessa função, todas as mensagens enviadas pelo sistema operacional para o programa são processadas.

```
LRESULT CALLBACK WindowProc(
    HWND hwnd,

    UINT uMsg,

    WPARAM wParam,

    LPARAM lParam );
```

Para cada mensagem recebida por esse programa, a função WindowProc é chamada. O parâmetro uMsg recebido por tal função informa o identificador da mensagem, enquanto os parâmetros wPARAM e lPARAM armazenam as informações extras e opcionais da mensagem.

Mediante uma comparação do valor de uMsg com os valores de MSG_FTIR_HIT e MSG_FTIR_HIT_FIM, identifica-se se a mensagem foi enviada pelo programa de reconhecimento. A parte do código computacional intitulada “Callback de mensagens”, responsável por essa análise, é apresentada no Anexo C.

Na sub-rotina mostrada no Anexo C, foi utilizada uma estrutura de dados para armazenar a seqüência de coordenadas dos pontos/informações registrados pela tela.

No caso de um programa *multitouch* simples, as coordenadas (x,y) recebidas em cada mensagem podem ser utilizadas diretamente para gerar algum tipo de resposta para o usuário. A forma como essas informações serão utilizadas depende puramente da natureza do programa *multitouch* desenvolvido.

Um exemplo simples seria um programa que simule um piano. Para cada toque feito na tela, basta verificar se alguma tecla foi pressionada e, nesse caso, o som relativo à tecla é tocado. Para aplicativos mais complexos que dependem do rastreamento de uma seqüência de coordenadas de um toque sobre a tela, deverão armazenar e gerenciar os toques ativos, e suas atualizações.