Universidade de Coimbra
Faculdade de Ciências e Tecnologias

Departamento de Engenharia Electrotécnica e de Computadores

Interpretação de Gestos Usando Visão por Computador para Interacção Homem-Máquina

Pedro **Miraldo**

8, Setembro 2008

Mestrado em Engenharia Electrotécnica e de Computadores
Júris:

Doutor Hélder Araújo (Orientador)
Doutor Paulo Peixoto
Doutor Rui Cortesão

Resumo

Com o desenvolvimento tecnológico na área da informática, surge a necessidade do aparecimento de novos modos de interacção homem-máquina. Este trabalho apresenta um conjunto de métodos que proporcionam a um computador o reconhecimento de vários gestos humanos em simultâneo. O método proposto agrupa um conjunto de técnicas que possibilitam a detecção de pele e seguimento na imagem, reconhecimento do gesto a partir do movimento e da postura.

A detecção de pele é feita em cor, recorrendo ao *teorema de Bayes* associado à aproximação da dinâmica da cor por uma gaussiana. Quanto ao seguimento, foi usada uma técnica baseada na distribuição espacial do objecto a ser seguido, admitindo que essa distribuição pode ser aproximada por uma elipse. Ainda para o seguimento, foi implementado um preditor (*filtro de Kalman*) para *data association* com o método probabilístico. O reconhecimento de movimentos é feito utilizando *Hidden Markov Models* com base em velocidades. Por fim, no reconhecimento de posturas de mãos, foi utilizado um método em tempo real que faz a segmentação dos dedos da mão.

O projecto foi implementado em *C/C++* utiliza uma *webcam USB* com uma resolução de 320×240 num processador *centrino* $2.13\,[GHz]$ com $1\,[GB]$ *de RAM*. Foi conseguida uma taxa de reconhecimento média de 82% no reconhecimento de movimentos e 92.5% no reconhecimento de posturas.

Abstract

This work presents a set of methods that provide a computer recognition of various human gestures simultaneously. The proposed method includes a set of techniques that enable skin color detection, tracking and gesture recognition based in movement and pose recognition.

The skin detection is done based in color, using the *Bayes theorem*, associated with the aproximation of dynamics color by a Gaussian distribution. Regarding the tracking, was used a technique based on the spatial distribution of the object being followed, assuming that the distribution can be approximated by an ellipse. Also with respect to tracking , was implemented a predictor (Kalman filter) to data association with the probabilistic method. The movement recognition is done using Hidden Markov Models based on velocity. Finally, for hand pose recognition, was implemented a method in real time that provide fingers segmentation.

The project was implemented in *C/C++*, using a USB camera with a resolution of 320×240 in a centrino processor $2.13\,[GHz]$ with $1\,[GB]$ of RAM. It was achieved an average recognition rate of 82% in movements recognition and 92.5% in poses recognition.

Reconhecimento

Neste último trabalho desenvolvido, é a altura ideal para agradecer a um conjunto de pessoas que não só contribuiram para a boa conclusão deste trabalho, como para a realização de todo o curso.

O primeiro agradecimento é para o professor Helder Araújo pela orientação ao longo de todo o projecto. Ao Rui Caseiro pelo tempo dispendido na correcção do meu relatório e a todas as pessoas do laboratório de visão pela companhia ao longo do projecto.

Para finalizar um agradecimento a todos os meu amigos e colegas e, sobre tudo, aos meus pais e à Andreia que sempre me apoiaram em tudo o que foi preciso.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Conteúdo

Capítulo 1

Introdução

O trabalho desenvolvido e descrito neste relatório, procura demonstrar as potencialidades da visão por computador em ambientes de cooperação homem-máquina. O reconhecimento de gestos e movimentos humanos é essencial neste tipo de ambientes. Neste trabalho foi implementado um conjunto de algoritmos capazes de classificar movimentos e posturas de mãos humanas usando visão por computador.

Este capítulo faz uma introdução ao projecto, estado de arte, objectivos, montagem e recursos utilizados.

1.1 Estado de Arte

O reconhecimento de gestos é uma área cada vez mais investigada, mesmo considerando a elevada quantidade de algoritmos já implementados. Esta área pode trazer mais comodidade às pessoas, como por exemplo, facilitar interfaces com o computador a pessoas portadoras de deficiências motoras.

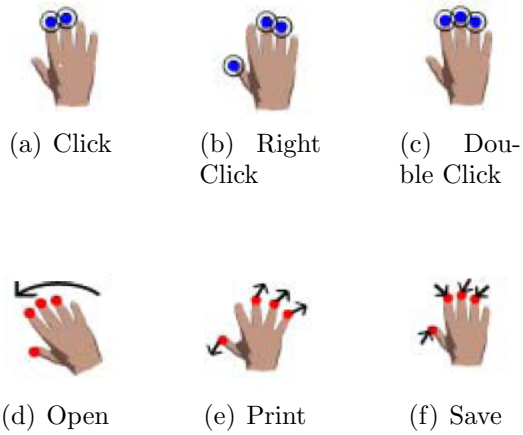
Existe um vasto leque de algoritmos para reconhecimento de gestos. Um método bastante interessante é o proposto por Athitsos em [4]. Um algoritmo capaz de estimar a pose $3D$ com base numa imagem. Athitsos considera a mão como um objecto articulado com 16 links (palma da mão e 15 correspondentes aos dedos) com um total de 20 graus de liberdade. A pose numa imagem é estimada a partir da comparação com poses bem definidas e guardadas numa base de dados. A

forma da mão representada na imagem depende dos parâmetros da câmara, o que implica que se tenha um conhecimento destes a fim de realizar uma classificação da postura.

Uma das áreas em que a interacção homem computador é essencial, é nos jogos de computador ou consola. Os jogadores interagem com os jogos através de *joysticks*, comandos, *trackballs* ou mesmo luvas eléctricas. Freeman em [7] propõe uma interacção natural com a consola através gestos de mãos ou do próprio corpo. O sistema proposto visa ser barato, relativamente a outras soluções, bem como ser robusto e rápido com algoritmos simples. Para além disso, procura introduzir novas e agradáveis experiências aos utilizadores. Freeman desenvolveu um módulo de retina artificial (AR), o qual combina um detector com um microprocessador com função de processamento de imagem. Foram concebidos dois algoritmos para o módulo (AR) para as aplicações em causa. Um deles baseado em momentos de imagem que fornece informações de posição e orientação e outro baseado em histogramas com o objectivo de classificação de poses. Estes algoritmos respondem ao corpo ou mão do utilizador em 10 $[ms]$.

João Carreira em [5] propõe uma alternativa aos tradicionais dispositivos de input, como o rato e o teclado. Os sistemas baseados em visão por computador capazes de reconhecer gestos são uma tecnologia promissora. O sistema proposto é capaz de controlar o computador através de gestos naturais sem auxílio de dispositivos externos. Carreira descreve uma interface baseada em visão por computador capaz de detectar e seguir mãos de múltiplos utilizadores e reconhecer movimentos e posturas. A interface proposta recorre à cadeia *Hidden Markov Model* para o reconhecimento de movimentos e silhuetas de contorno para o reconhecimento de posturas. João Carreira apresenta ainda uma aplicação que permite aos utilizadores tocar instrumentos musicais através do movimentos de mãos.

Com o objectivo de criar novas interfaces homem-computador cada vez mais investigadores tentam encontrar diferentes métodos nesta área. No conjunto de métodos de interacção, o reconhecimento contínuo de gestos e reconhecimento da fala podem ser considerados, se não os mais importantes, dos mais importantes. Em [13, 9] Sharma e Kettebekov, respectivamente, propõem dois métodos para inter-



(a) Click (b) Right Click (c) Double Click

(d) Open (e) Print (f) Save

Figura 1.1: Exemplos de gestos.

face homem-computador, baseados no reconhecimento de gestos e reconhecimento da fala, para interacção de um pivô de televisão, na apresentação do boletim meteorológico. Para o reconhecimento da fala bem como o reconhecimento dos gestos utilizam a tradicional cadeia *Hidden Markov Model*.

Os algoritmos de reconhecimento de gestos de mãos, baseados em tecnologias de *multi-touch*, muitas vezes são criados com o objectivo de trabalhar com toda a mão. Sendo assim, estes algoritmos não estão acessíveis a pessoas com deficiências físicas. Em [14] Yuan criou um conjunto de gestos, do tipo comando, para utilizadores com limitações físicas. Trajectórias e ângulos são extraídos de gestos e passados para uma rede neuronal para o reconhecimento. O conjunto de gestos criados é mostrado na figura 1.1.

Normalmente a investigação na área da interacção homem-máquina foca assuntos como gestos de mãos e reconhecimento da fala. No entanto, o reconhecimento de gestos de todo o corpo é muitas vezes necessário para uma boa interacção homem-máquina. Em [10] Lee apresenta um novo método para o reconhecimento de gestos de todo o corpo implementado num robô móvel. À semelhança de Athitsos em [4], Lee considera o corpo humano como um corpo rígido articulado, dividido em 17 segmentos com um máximo de 40 graus de liberdade como podemos ver na figura 1.2. O reconhecimento do gestos é feito através de uma *HMM*. O robô utilizado no projecto foi o *T-Rot*, um robô de serviço pessoal destinado a ajudar pessoas

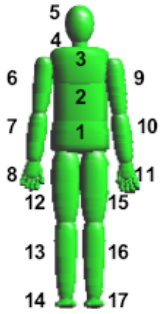


Figura 1.2: Modelo $3D$ do corpo proposto por Lee.

idosas.

1.2 Objectivos

Pode subdividir-se o objectivo de reconhecimento de gestos em tarefas. A primeira, e talvez a mais importante, é realizar a detecção e segmentação de padrões. A detecção dee padrões para a sgmentação deve ser feito em cor.

Após a realização da detecção de pixeis alvo, a associação em regiões e a segmentação da imagem, é necessário fazer a associação temporal das regiões na imagem.

O reconhecimento do movimento de regiões de uma imagem é a tarefa que se segue, ficando para último a elaboração do reconhecimento de poses de mãos.

1.3 Montagem e Recursos Utilizados

Para a elaboração do trabalho foi utilizado como *hardware*, uma *webcam USB* vulgar, ligada a um computador portátil *Centrino* $2.13\,[GHz]$, com $1024\,[MB]$ de *RAM*. Até à parte do seguimento, a câmara foi colocada no monitor apontada para uma pessoa. Quando foi feito o reconhecimento de movimentos e mais tarde o reconhecimento de poses ou posturas, foi utilizado um suporte para fixar a câmara, para assim poder dirigir o seu foco para a mesa. A figura 1.3 mostra o local de

Figura 1.3: Figura do posto de trabalho.

desenvolvimento do projecto.

A nível de *software*, recorreu-se à linguagem de programação *C/C++* com o compilador *GCC* em *GNU Linux*. Foram utilizadas bibliotecas como OpenCV [6], desenvolvida pela Intel, Lti-Lib [1], desenvolvida pela Universidade de Aachen e, ainda, *GTK+2* para criar um ambiente gráfico. Foi ainda utilizado o *Matlab* a fim de realizar alguns testes.

1.4 Estrutura do Trabalho Desenvolvido

Esta secção é dedicada à descrição de todo o relatório e do código do software.

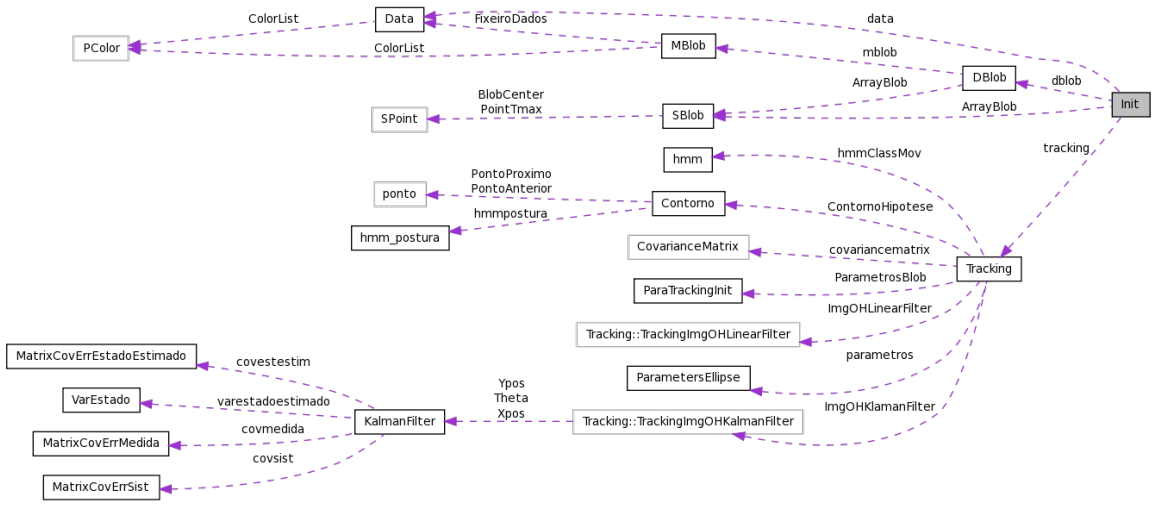


Figura 1.4: Esquema de ligação de classes do projecto *C/C++*.

1.4.1 Estrutura do Relatório

Este relatório é composto por 5 capítulos. O primeiro faz uma pequena introdução do trabalho elaborado. O segundo capítulo faz uma descrição mais completa do problema da detecção de pixeis cor de pele e seguimento. O terceiro capítulo explica como foi implementado o reconhecimento de movimentos seguindo-se o capítulo relativo ao reconhecimento de poses. Por fim, é feita a conclusão com um sumário do trabalho realizado e trabalho futuro.

1.4.2 Estrutura do Código

Como foi dito na secção 1.3 todo o trabalho foi implementado em *C++*. Para uma melhor organização do trabalho foram criadas várias bibliotecas de classes com os vários objectivos propostos no trabalho. A figura 1.4 mostra a ligação das classes (*C/C++*) utilizadas na elaboração do projecto.

O software desenvolvido inclui um pacote de documentação gerado por *Doxygen*. Este pacote é constituido por diagramas de classes bem como algumas descrições das respectivas funções.

Capítulo 2

Detecção de Padrões e Seguimento

Como já foi dito anteriormente, o ponto de partida deste projecto é realizar a "detecção" das regiões que queremos seguir. Este capítulo foca a parte da detecção e segmentação das regiões de pixeis correspondentes aos objectos em causa. Depois de feita essa diferenciação, é implementado a associação temporal dessas regiões na imagem.

2.1 Introdução

Um dos problemas fundamentais dos sistemas de visão por computador é o seguimento de *features*[1] numa sequência de imagens. Um exemplo pode ser o seguimento de partes do corpo humano que contêm pele (como caras e mãos) com o objectivo de seguir a realização de um gesto.

O corpo humano pode ser visto como uma estrutura complexa, não rígida, com muitos graus de mobilidade. Assim, o seguimento de parte dessa estrutura, extraída de uma imagem, torna-se bastante complexa e dispendiosa, dependendo da actividade dos vários graus de mobilidade.

Os métodos mais utilizados para a detecção de objectos são a detecção a partir de

[1] Objectos ou simplesmente pontos de interesse numa frame

Figura 2.1: Exemplo de segmentação de uma mão.

cor, texturas e movimento bem como qualquer fusão destes três métodos. A cor oferece muitas vantagens sobre métodos geométricos, sendo que a mais relevante é o facto de ser substancialmente menos dispendioso a nível de processamento que os métodos geométricos. Em [2] Argyros propõe um método de detecção em cor, a partir do *teorema de Bayes*, e um método de seguimento clássico em tempo real.

A pergunta que se coloca é como segmentar pixeis que representam cor de pele do resto da imagem. Um exemplo dessa tarefa é mostrado na figura 2.1.

2.1.1 Modelação da Cor e Detecção

A primeira decisão a ser tomada é definir o espaço de cor que se vai usar para a segmentação em cor. Vários espaços estão à disposição, entre eles: RGB (mais comum), RGB *normalizado*, HSV, YCC (ou $YCrCb$), YUV, etc. Os espaços de cor que contêm crominância e luminosidade (YCC), são preferíveis aos que apenas contêm crominâncias (RGB). Quando utilizamos espaços de cor apenas com crominâncias, perde-se alguma robustez na detecção de uma cor, quando há variações de intensidade de luz.

Após a selecção do espaço de cor, uma maneira simples de definir, se um dado pixel é ou não de cor de pele, é dar pesos à gama de cores e definir um limiar. Os pixeis com peso superior a esse limiar são considerados pixeis cor de pele. Esses pesos são normalmente atribuídos empiricamente, isto é, observando a distribuição de cor de pele num dado número de imagens de treino. Outra possível opção é admitir

que a probabilidade de cor da pele segue uma dada distribuição, que pode ser extraída a partir de imagens de treino. No caso de aproximações não paramétricas, a distribuição pode ser treinada através de um histograma de probabilidades.

2.1.2 Associação Temporal

Depois de ter modelado as regiões de cor de pele, segue-se o problema da identificação das várias regiões detectadas em sequências de imagens, problema do seguimento. A resolução tradicional deste problema é a aplicação do *filtro de Kalman*. Se as observações e a dinâmica do sistema podem ser representadas por uma distribuição gaussiana, o *filtro de Kalman* pode ser uma solução para o problema. No entanto, em muitos casos, a dinâmica da cor não pode ser aproximada por uma gaussiana, o que faz com que não seja possível recorrer à utilização do *filtro de Kalman*.

2.1.3 Método Adoptado

A pergunta que se coloca agora é como fazer a detecção de objectos (cor pele) e como se vai associa-los ao longo do tempo.

Para fazer a detecção de objectos recorreu-se à detecção em cor. A primeira coisa a fazer é escolher um espaço de cor associado ao problema (como podemos ver na secção 2.1.1). Neste caso, com o objectivo de tentar minimizar os danos devidos a variações na luz, foi escolhido o espaço de cores *YCC*. Este espaço de cor contém uma componente de luminosidade para duas de cor.

Para detecção de pixeis potencialmente pertencentes à pele, foi utilizado o *teorema de Bayes*. Porém, para a aplicação deste teorema, é necessário um treino, para a determinação dos seus parâmetros. Foi também implementada uma adaptação online da classificação da cor através da aproximação da dinâmica da cor por uma gaussiana. Se houver uma quantidade de pixeis considerados cor de pele aglomerados, então pode-se criar ai uma região que se pode afirmar ser de cor de pele, segmentando-a do resto dos pixeis.

Para o seguimento foi utilizado um método estatístico que se baseia em matrizes

de covariâncias da distribuição dos pixeis em cada região considerada cor de pele. A partir das matrizes de covariâncias são geradas elipses de hipóteses de posições. Se se considerar que se tem uma recolha de imagens a uma taxa aceitável, é de esperar que na frame seguinte à formação da elipse, a maioria dos pixeis ainda esteja no seu interior. Com isto, consegue-se associar regiões em função do tempo. No entanto, este algoritmo é algo sensível, nomeadamente no que se refere ao cruzamento de mãos. Para resolver esse problema, foi implementado um *filtro de Kalman* com o objectivo de fazer a predição da posição das mãos.

2.2 Descrição do Método Adoptado

Partindo o problema em partes, fica-se perante quatro tarefas a cumprir para realizar o seguimento:

- Conversão de *RGB* para *YCC*;
- Treino offline e mecanismo de segmentação utilizando o *teorema de Bayes*;
- Adaptação em tempo real utilizando a representação da dinâmica da cor por uma Gaussiana;
- Seguimento utilizando um método de seguimento em tempo real, associado a um *filtro de Kalman*.

Os objectos adoptados para o projecto foram partes do corpo humano (cor pele) como mãos e caras.

2.2.1 Conversão de RGB para YCC

O espaço de cor mais comum é o *RGB*. A câmara utilizada (Logitech Quickcam pro 9000) fornece imagem nesse mesmo espaço de cores, pelo que foi necessário realizar a conversão de cor para *YCC*. Este espaço de cor é um composto de canais de luminosidade e cor, mais precisamente, por 3 canais:

Y - Canal que representa a luminosidade;

C - Canais de cor.

Como a cor verde pode ser obtida a partir da subtracção da luminosidade por a componente de vermelho e azul, conseguiu-se representar as componentes de RGB através de duas componentes de cor subtraídas à luminosidade. A dita luminosidade, cor vermelha e cor azul são normalmente designados como Y, Cr e Cb respectivamente.

Como a maior parte dos espaços de cor disponíveis, o $YCrCb$ pode ser obtido directamente de RGB (o espaço de cor mais comum) a partir de uma função linear:

$$\begin{aligned} v1 &= Ar_1R + Ag_1G + Ab_1B \\ v2 &= Ar_2R + Ag_2G + Ab_2B \\ v3 &= Ar_3R + Ag_3G + Ab_3B \end{aligned} \tag{2.1}$$

em que: Ar, Ag e Ab são constantes.
Para o caso dos espaços de cor que vamos utilizar:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cr &= 0.701R - 0.587G - 0.114B \\ Cb &= -0.299R - 0.587G + 0.886B \end{aligned} \tag{2.2}$$

Matricialmente:

$$YCrCb = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.701 & -0.587 & -0.114 \\ -0.299 & -0.587 & 0.886 \end{bmatrix} RGB \tag{2.3}$$

Reparar, em (2.2), que a maior contribuição para Cr é a da componente vermelha de RGB assim como a maior contribuição para Cb é a componente azul.

2.2.2 Segmentação da Imagem em Cor

Depois de ter sido escolhido o espaço de cor (YCC), a tarefa que se segue é detectar regiões definidas com cor de pele. A detecção de regiões cor de pele envolve quatro etapas:

- Estimação da probabilidade de um pixel ser cor de pele;

- Definição de um limiar para o qual um pixel é considerado ser pele a partir da probabilidade estimada;
- Um conjunto de pixeis ligados, pode ser considerado como sendo uma região cor de pele;
- A informação relativa ao conjunto de pixeis identificados como cor de pele gera uma adaptação online para a classificação futura desses mesmos pixeis.

Para a estimação da probabilidade de um pixel ser cor de pele, recorreu-se ao *teorema de Bayes*. Enquanto, para a adaptação online obtida a partir do histórico dos pixeis cor de pele, é feita a aproximação da dinâmica da cor de pele por uma gaussiana.

Detecção de Regiões Cor de Pele

Assumindo que se tem um conjunto de imagens de treino, no formato YCC, que contêm um conjunto de regiões cor de pele definidas manualmente. Essas imagens vão servir de treino para a estimação de pixeis cor de pele. As componentes que vão ser utilizadas para treino vão ser, unicamente, as duas componentes de cor CC. A componente Y não foi utilizada por duas razões. A primeira razão, como é óbvio, é devido a Y ser a componente de luminosidade. Como não se quer que o sistema seja sensível à luminosidade então não se utiliza esta componente. A segunda razão é a de que se utilizarmos um espaço de representação de cor $2D$ (CC) em relação a um $3D$, a dimensão do problema é reduzida, assim como o tempo de processamento de todo o sistema.

Assumindo que os pontos da imagem $I(x, y)$ têm uma cor $c = c(x, y)$, o treino baseia-se em:

- Calcular a probabilidade de uma imagem conter cor de pele para todos os pixeis, $P(s)$;
- Calcular a probabilidade de ocorrer cada cor c nas imagens de treino, $P(c)$;
- Calcular a probabilidade de cada cor c em regiões cor de pele, $P(c|s)$.

Seguidamente ao treino, a probabilidade de um dado pixel ser cor de pele, sabendo que tem uma cor c pode ser representada pela aplicação do *teorema de Bayes*:

$$P(s|c) = \frac{P(c|s)\,P(s)}{P(c)} \tag{2.4}$$

Assim, a probabilidade de cada pixel na imagem ser cor de pele pode ser determinada e todos os pontos com probabilidade $P(s|c) > T_{max}$ são considerados como sendo pixeis cor de pele (T_{max} é um limite superior para o qual todos os pixeis são considerados imediatamente pixeis cor de pele). Um outro limite, T_{min}, é um limite inferior, o qual considera que um pixel é cor de pele se satisfizer duas condições. A primeira é $P(s|c) > T_{min}$. A segunda condição é ser vizinho de um pixel considerado cor de pele. Este processo é feito recursivamente. T_{max} é então considerado como o limite potencializador da existência de regiões cor de pele.

As regiões com um número mínimo de pixeis cor de pele são etiquetadas, por exemplo utilizando os valores em que $P(s|c) > T_{max}$, e é aplicada uma filtragem no número de pixeis de cada região com o objectivo de evitar que pequenos aglomerados de pixeis sejam considerados como sendo regiões de pele. Todas as regiões que restam da filtragem são consideradas como regiões cor de pele.

Treino Offline

O treino é um procedimento offline que vai fornecer a probabilidade de um pixel ser cor de pele (2.4). O treino baseia-se no cálculo da probabilidade $P(s)$, probabilidade de haver pixeis cor de pele nas imagens de treino. $P(c)$, probabilidades de haver a cor c nas imagens de treino (nota que as cores são representadas em $8[bit]$ ou seja, 256 cores por componente, o que dá 256×256 cores possíveis para as duas componentes). E por fim, calcular $P(c|s)$ para as mesmas cores de $P(c)$.

Para o cálculo dessas probabilidades é necessário assinalar todos os pontos que são cor de pele nas imagens de treino, o que é uma tarefa trabalhosa. Para tornar mais fácil a leitura dos pixeis que são cor de pele, foi assinalada a fronteira das regiões consideradas como pele nas imagens de treino (figura 2.2).

Após a aquisição das probabilidades necessárias para todas as cores, basta analisar

Figura 2.2: Exemplo das imagens de treino.

as cores dos pixeis da nova imagem e determinar se são cor pele ou não.

Adaptação Online

Para a determinação de pixeis cor de pele foi ainda implementada a adaptação online da cor, utilizando a aproximação da dinâmica da cor por uma gaussiana. A distribuição gaussiana univariável é caracterizada pelos parâmetros μ e σ e tem a seguinte forma

$$f_g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.5}$$

Muitas vezes expresso por:

$$X \sim N\left(\mu, \sigma^2\right) \tag{2.6}$$

Em que μ e σ representam a média e o desvio padrão respectivamente. No entanto, no caso em estudo (utilizando o espaço de cor YCC) estamos a utilizar dois canais de cor. Para isso, recorre-se a uma distribuição gaussiana multivariável.

$$p(x) = p\left(x_{c_r}, x_{c_b}\right) = \frac{1}{\sqrt{2\pi\left|\Sigma\right|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{2.7}$$

Aqui $\mu = \begin{bmatrix} \mu_{c_r} & \mu_{c_b} \end{bmatrix}$ é um vector de médias e Σ é a matriz de covariâncias:

$$\Sigma = \begin{bmatrix} \sigma_{c_r c_r} & \sigma_{c_r c_b} \\ \sigma_{c_b c_r} & \sigma_{c_b c_b} \end{bmatrix} \tag{2.8}$$

Para os cálculos de μ e Σ vamos utilizar n frames de uma janela ao longo do tempo. Notar que o aumento de n pode provocar a divergência da detecção de pixeis de pele para toda a imagem.

A atribuição da identificação de um pixel como sendo cor de pele é então feita a partir de um limiar l tal que: $p\left(x_{c_r}, x_{c_b}\right) > l$.

2.2.3 Seguimento de Objectos em Função do Tempo

Assumindo que no instante t, M regiões, consideradas como cor de pele, foram detectadas segundo a secção 2.2.2. Cada região b_j, $1 \leq j \leq M$, corresponde a um conjunto de pixeis cor de pele. No entanto, de notar que a relação região cor de pele e objectos (por exemplo, mãos e caras) não é um para um. Por exemplo, mãos cruzadas correspondem a dois objectos cor de pele, no qual aparece apenas uma região cor de pele segundo a secção 2.2.2. Neste projecto foi assumido que um objecto pode corresponder a um blob[2] ou parte do blob. Ao contrário, um blob pode corresponder a um ou mais objectos.

Considerando também que a distribuição espacial dos pixeis pode ser aproximada por uma elipse. Esta consideração é válida para objectos como palmas de mãos e caras. Assumimos com isto que existem N objectos (regiões) cor de pele presentes no instante de tempo t e o_i, $1 \leq i \leq N$, ser o conjunto de pixeis que representam o objecto i. Por fim, sendo $h_i = h_i\left(c_{x_i}, c_{y_i}, \alpha_i, \beta_i, \theta_i\right)$ o modelo da elipse do objecto i com $\left(c_{x_i}, c_{y_i}\right)$ o centróide, α_i e β_i o maior e menor eixo respectivamente e θ_i a orientação da elipse no plano da imagem. Para terminar, foram utilizadas as letras maiúsculas $B = \cup_{j=1}^{M} b_j$, $O = \cup_{j=1}^{N}$ e $H = \cup_{j=i}^{N}$ para definir a união dos pixeis cor de pele, união dos pixeis relativos aos objectos e união de elipses respectivamente.

[2]Blob (do inglês *Binary Large Object*) significa um grande pedaço de dados que são armazenados em um banco de dados. Neste caso, o blob corresponde à região composta por pixeis de pele.

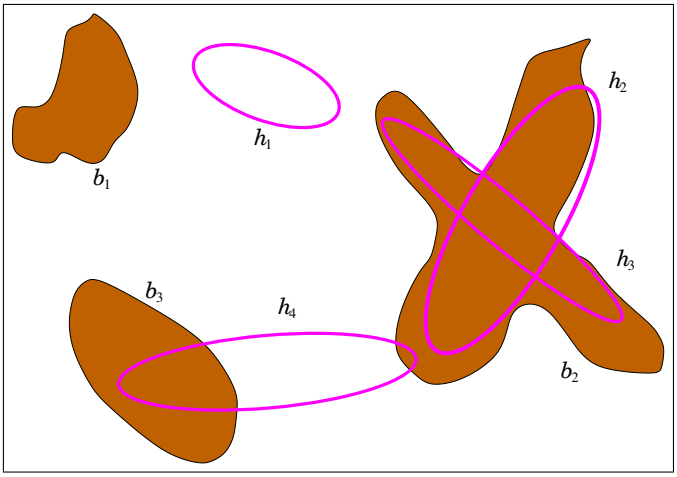


Figura 2.3: Vários casos de relações entre blobs cor de pele e hipóteses de objectos. Elipses a magenta (h_i) e blobs cor de pele a castanho (b_i).

O problema do seguimento, torna-se então no problema da determinação da relação entre o modelo dos objectos (h_i) e das observações (b_j) em função do tempo. A figura 2.3 exemplifica o problema. Neste exemplo, no instante t, existem três blobs (b_1, b_2 e b_3) enquanto existem quatro hipóteses de objectos (h_1, h_2, h_3 e h_4) definidas na imagem tirada anteriormente.

De seguida é apresentado um algoritmo com capacidade de resolver o problema da associação. O algoritmo proposto necessita de endereçar três subproblemas:

- Geração de uma hipótese de objecto, isto é, um objecto aparece no campo de vista da câmara pela primeira vez;

- Seguimento de um hipótese de objecto na presença de múltiplos potenciais objectos ao longo do tempo, isto é, objectos previamente detectados movem-se arbitrariamente ao longo do campo de vista da câmara;

- Remoção de uma hipótese de objecto, isto é, quando um objecto, previamente detectado, desaparece do campo de vista da câmara.

Geração de uma hipótese de objecto

Definindo a distância $D(p,h)$ como sendo a distância de um ponto $p=(x,y)$ a uma elipse $h(c_x,c_y,\alpha,\beta,\theta)$ como

$$D(p,h)=\sqrt{\overrightarrow{v}\,\overrightarrow{v}} \tag{2.9}$$

com

$$\overrightarrow{v}=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}\frac{x-x_c}{\alpha}\\ \frac{y-y_c}{\beta}\end{bmatrix} \tag{2.10}$$

A partir da definição de (2.9) o valor de $D(p,h)$ pode tomar três gamas de valores, menor que um, maior que um e igual a um, dependente de onde se situa o ponto p relativamente à elipse h. Se p se situa dentro da elipse h, $D(p,h)$ toma um valor menor que um, se se situa no exterior, toma um valor maior que um e se está na linha pertencente à elipse toma o valor de um.

Considerando um modelo de elipse h e um ponto p pertencente a um blob b. No caso de $D(p,h)<1.0$, conclui-se que esse ponto p e o blob b suportam a existência de uma hipótese de objecto h e que a hipótese de objecto h prediz o blob b. Considerar agora um blob b tal que:

$$\forall p\in b, \min_{h\in H} D(p,h)>1.0 \tag{2.11}$$

A equação (2.11) descreve um blob sem intersecção com qualquer hipótese de objectos. Na figura 2.3, o blob b_1 exemplifica um destes casos. Isto significa que nenhuma nenhuma hipótese de objecto anterior suporta a existência do blob. Para cada um desses casos, uma nova hipótese de objecto deve ser gerada. Os parâmetros da hipótese de objecto gerada podem ser extraídos directamente a partir da estatística da distribuição dos pontos pertencentes ao blob. O centro da elipse da hipótese de objecto é obtida directamente, a partir do centróide do blob e o resto dos parâmetros podem ser obtidos a partir da matriz de covariâncias da distribuição bivariante dos pixeis pertencentes ao blob. Mais especificamente, pode ser mostrado que se matriz de covariâncias Σ da distribuição dos pixeis no

blob é

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} \tag{2.12}$$

então, podemos definir a elipse com os parâmetros:

$$\alpha = \sqrt{\lambda_1} \quad \beta = \sqrt{\lambda_2} \quad \theta = \tan^{-1} \frac{\sigma_{xy}}{\lambda_1 - \sigma_{yy}} \tag{2.13}$$

com:

$$\begin{aligned} \lambda_1 &= \frac{\sigma_{xx}+\sigma_{yy}+\Lambda}{2} \\ \lambda_2 &= \frac{\sigma_{xx}+\sigma_{yy}-\Lambda}{2} \\ \Lambda &= \sqrt{(\sigma_{xx}-\sigma_{yy})^2 - 4\sigma_{xy}^2} \end{aligned} \tag{2.14}$$

A cada instante t, todos os blobs (detectados segundo 2.2.2) devem ser testados segundo o critério (2.11). Para todos os blobs que satisfaçam a equação (2.11), uma hipótese de objecto nova é gerada. Os parâmetros da nova elipse são gerados segundo (2.13).

Seguimento de uma hipótese de objecto

Após se ter analisado e detectado todos os casos de novas hipóteses, os restantes blobs devem suportar a existência de hipóteses passadas. A principal tarefa do algoritmo de seguimento resume-se a associar os blobs a hipóteses. Subdividiu-se esta tarefa em duas regras:

Regra 1 Se um pixel cor de pele do blob está localizado dentro da elipse de alguma hipótese de objecto (isto é, suporta a existência de uma hipótese) então, este pixel é considerado como pertencente a essa hipótese.

Regra 2 Se um pixel cor de pele está fora de todas as elipses correspondentes a hipóteses de objectos, então esse é associado à elipse de objecto de hipótese mais próxima segundo (2.9)

Formalmente, o conjunto O de pixeis cor de pele que estão associados e hipóteses de objectos h é dado por $O = R_1 \cup R_2$ com $R_1 = \{p \in B \mid D(p,h) < 1.0\}$ e $R_2 = \{p \in B \mid D(p,h) = \min_{k \in H} \{D(p,k)\}\}$.

No exemplo da figura 2.3 duas hipóteses de objectos (h_2 e h_3) competem pelo blob cor de pele b_2. Segundo a **Regra** 1, todos os pixeis cor de pele dentro da elipse h_2 são associados à hipótese de objecto 2 o mesmo acontecendo para os pixeis dentro da elipse h_3. Notar que os pixeis que estão dentro das duas elipses vão ser associados às duas hipóteses h_2 e h_3. Segundo a **Regra** 2, os pixeis do blob b_2 que não se situam dentro de nenhuma elipse, vão ser associados à elipse de hipótese mais próxima determinado por (2.9).

Mais um caso de analise é quando uma hipótese é suportada por mais do que um blob. (hipótese h_4 na figura 2.3). Esta situação pode acontecer, por exemplo, quando dois objectos aparecem ligados, no instante de tempo em que estes entram em cena, e mais tarde se separam. Para resolver a situação de uma hipótese suportar mais que um blob foi adoptada a seguinte estratégia. Se existe apenas um blob b que é suportado por h e , no mesmo instante, não é suportado por mais nenhuma hipótese, então h é associado a b. Se não for o caso, h é associado ao blob que contém o maior número de pixeis cor de pele. Voltando ao exemplo da figura 2.3, a hipótese h_4 contém os blobs b_2 e b_3. Baseado no que foi descrito, h_4 vai ficar associado ao blob b_3.

Depois de se ter associado os pixeis cor de pele às hipóteses de seguimento, os parâmetros das elipses de hipóteses de objecto (h_i) devem ser recalculados baseado na distribuição estatística dos pixeis O_i que foram associados a eles.

Remoção de uma hipótese de objecto

Uma hipótese de objecto deve ser removida quando um objecto move-se para fora do campo de visão da câmara, ou quando o objecto é obstruído por outro objecto (não cor de pele) em cena. Assim, uma hipótese de objecto h_i deve ser removida quando:

$$\forall p \in B, D\left(p, h\right) > 1.0 \tag{2.15}$$

A equação (2.15) descreve hipóteses não associadas a nenhum pixel cor de pele. Mais uma vez, na figura 2.3, a hipótese h_1 é um desses casos. Na prática, deve ser permitido a uma hipótese de objecto "sobreviver" um certo intervalo de tempo mesmo que não satisfaça (2.15). Neste caso, foi usado meio segundo.

Predição

Nos processos de geração, seguimento e remoção de hipóteses de objecto que foram descritos em cima, a associação dos dados no instante (t) é baseada apenas na hipótese de objectos definidos no instante de tempo anterior. Existe então um problema de ligação entre a dinâmica do movimento e a amostragem das imagens.

Para resolver este problema foi utilizado o *filtro de Kalman* [3]. Este filtro pode ser dividido em duas partes:

Predição Como o próprio nome indica, a partir dos dados obtidos no passado prevê o estado actual.

Correcção Com base nos dados medidos e na predição feita, faz uma filtragem.

A descrição matemática das duas partes é dada pelo algoritmo 1, em que:

Algoritmo 1 Filtro de Kalman

Predição:

$$\begin{aligned} \hat{x}^{-}(k) &= \Phi \hat{x}(k-1) + \Gamma u(k-1) \\ P^{-}(k) &= \Phi P(k-1) \Phi^{T} + Q(k-1) \end{aligned} \tag{2.16}$$

Correcção:

$$\begin{aligned} K(k) &= P^{-}(k) C^{T} \left[C P^{-}(k) C^{T} + R(k) \right]^{-1} \\ \hat{x}(k) &= \hat{x}^{-}(k) + K(k) \left[y(k) - C \hat{x}^{-}(k) \right] \\ P(k) &= \left[I - K(k) C \right] P^{-}(k) \end{aligned} \tag{2.17}$$

$x(k)$ **-** Representa vector de estado;

$P(k)$ **-** Matriz de covariâncias do erro do estado;

Φ**,** Γ **e** C **-** Representam a dinâmica do sistema a seguir;

$K(k)$ **-** Matriz de ganhos do *filtro de Kalman*;

$Q(k)$ **e** $R(k)$ **-** Representam a matriz de covariâncias do ruído do processo e a matriz de covariâncias do ruído da medida respectivamente.

Existem então dois grandes problemas para a aplicação do filtro. Qual é a dinâmica do sistema que se quer seguir (Φ, Γ e C) e quais são as matrizes de ruído do processo e da medida ($Q(k)$ e $R(k)$).

A dinâmica adoptada foi a dinâmica de um sistema uniformemente acelerado descrito por:

$$\begin{aligned} p &= p_0 + v_0 t + \tfrac{1}{2} a t^2 \\ v &= v_0 + at \end{aligned} \tag{2.18}$$

Matricialmente:

$$\begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2} t^2 \\ t \end{bmatrix} a \tag{2.19}$$

Considerando a aceleração constante (2.20) fica:

$$\begin{bmatrix} p \\ v \\ a \end{bmatrix} = \begin{bmatrix} 1 & t & \frac{1}{2} t^2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ v_0 \\ a \end{bmatrix} \tag{2.20}$$

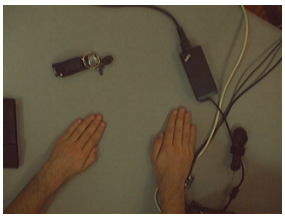
Em (2.20) tiramos facilmente que: $x(k) = \begin{bmatrix} p & v & a \end{bmatrix}^T$, a matriz de transição de estados $\phi = \begin{bmatrix} 1 & t & \frac{1}{2} t^2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix}$ e a matriz $\Gamma = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.
O objectivo deste filtro é fazer a predição de um objecto numa imagem. Da imagem podemos extrair directamente a posição (na imagem) dos objectos, portanto,

$$y(k) = p(k) \equiv y(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x(k) \equiv y(k) = Cx(k) \tag{2.21}$$

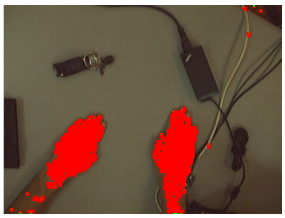
ou seja,

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{2.22}$$

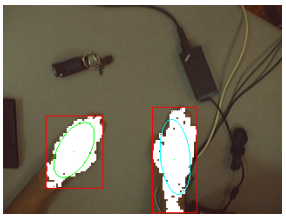
A saída do *filtro de kalman* vai fornecer posição do centro da hipotese (x_c, y_c).



(a) Frame capturada.

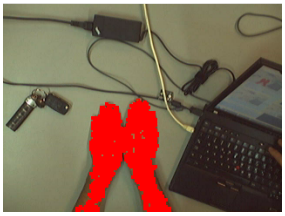


(b) Pixeis cor de pele na imagem.

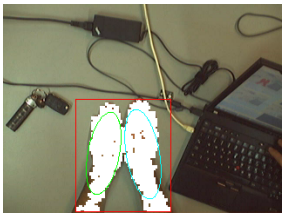


(c) As regiões consideradas como pele estão assinaladas dentro de rectângulos vermelhos. As hipóteses de objecto estão assinaladas pelas elipses.

Figura 2.4: Exemplo de detecção de regiões cor pele e seguimento.



(a) Pixeis cor de pele detectados.



(b) Hipoteses de objectos detectados.

(c) Exemplo de imagem segmentada.

Figura 2.5: Exemplo de detecção de regiões cor pele e seguimento com duas mãos juntas.

2.3 Resultados

Na figura 2.4 pode ver-se o resultado da aplicação da detecção do reconhecimento de padrões implementado bem como das elipses de seguimento.

Na figura 2.4(b) estão representados os pixeis cor de pele encontrados numa frame. De notar que se pode encontrar na figura duas cores, vermelho e verde. A cor vermelha mostra os pixeis encontrados ao aplicar o limite T_{min} recursivamente. Enquanto a verde é quando é encontrado um pixel com limite T_{max} explicado na secção 2.2.2.

Na figura 2.4(c) são mostradas as regiões consideradas como objectos cor pele

e também as elipses de hipótese usadas no seguimento de cada região, secções 2.2.2 e 2.2.3 respectivamente. As regiões estão assinaladas dentro de rectângulos vermelhos, enquanto as hipótese são as elipses, cores diferentes para diferentes hipóteses de objectos.

Na figura 2.5 é mostrado o caso de haver duas mãos juntas. Ou seja, segundo a secção 2.2.2 apenas é assinalado um blob (como podemos ver na figura 2.5(b), só existe um rectângulo a vermelho). No entanto, segundo o algoritmo de seguimento, são encontradas duas hipóteses de objecto (como é mostrado pelas duas elipses de cor diferente na figura 2.5(b)). A figura 2.5(c) mostra a segmentação da imagem.

Capítulo 3

Reconhecimento de Movimentos

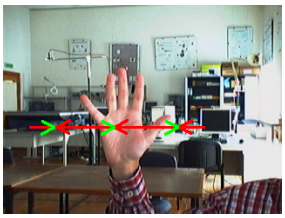
Após a realização da detecção de objectos como mãos e caras, a primeira tarefa no reconhecimento de gestos foi o reconhecimento de movimentos.

3.1 Introdução

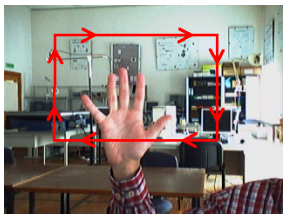
Este capítulo é então dedicado ao reconhecimento de movimentos. O objectivo é fazer o reconhecimento baseado em movimentos previamente treinados. No capítulo 2 foi explicado como foi feita a detecção de cor de pele, respectiva segmentação e a associação de regiões cor de pele ao longo do tempo.

O reconhecimento de um dado movimento depende da posição de um dado objecto em função do tempo. Ou seja, depende do seu vector velocidade $v = (v_x, v_y)$. A velocidade de um objecto na imagem pode ser obtida a partir da diferenciação da sua posição em relação ao tempo $\left(v = \frac{\nabla p}{\nabla t}\right)$. O facto de considerarmos velocidades e não posições, faz com que o reconhecimento do gesto seja independente da zona da imagem onde é detectado.

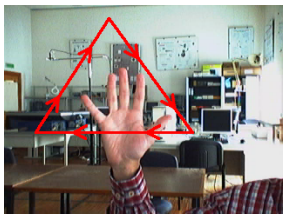
Neste trabalho foram reconhecidos 5 movimentos: movimento *ondular* (3.1(a)), *quadrado* (3.1(b)), *triângulo* (3.1(c)), *circunferência* (3.1(d)) e movimento em L (3.1(e)). Para o cálculo de v foi utilizada a diferenciação da posição do centro da elipse de hipótese de objecto (h_i), $c = (c_x, c_y)$ definida em 2.2.3.



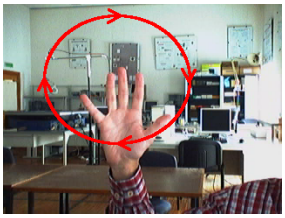
(a) Movimento ondular.



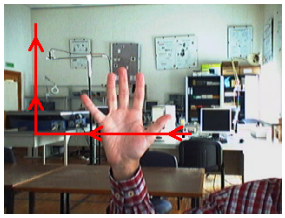
(b) Movimento quadrado.



(c) Movimento triangular.



(d) Movimento circular.



(e) Movimento L.

Figura 3.1: Movimentos com treino *offline* para o reconhecimento de movimentos de objectos.

Para fazer a classificação e treino recorreu-se à cadeia *Hidden Markov Model (HMM)*. Para a criação das sequências de treino foi gerado um ambiente gráfico em GTK que possibilita ao utilizador criar as sequências de treino.

3.2 Hidden Markov Model

Considere-se um sistema que pode ser descrito como estando em qualquer momento, em um, de um conjunto de N estados distintos $(s_1, s_2, \ldots, s_N)$ como ilustra a figura 3.2. A cada amostra de tempo o sistema sofre uma transição de estado de acordo com as probabilidades associadas às transições $(a_{i,j})$.

No modelo *Markov* regular é considerado que a cada estado corresponde um evento observável. Esta abordagem tem desvantagens uma vez que impõe demasiadas restrições para a aplicação em alguns problemas interessantes. Tendo em conta estes factos, o conceito de modelo de *Markov* é estendido de forma a considerar que as observações sejam funções probabilísticas do estado. O modelo resultante (*Hid-*

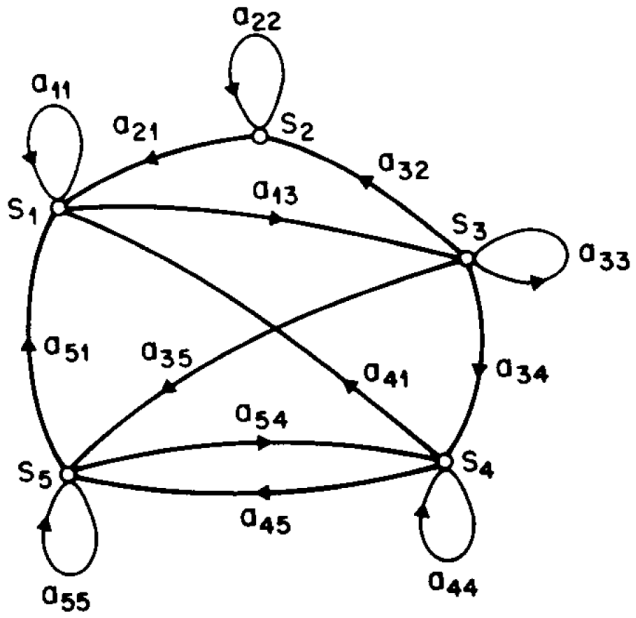


Figura 3.2: Exemplo de cadeia de *Markov* com 5 estados.

den Markov Model) é um processo estocástico duplo no qual um deles é escondido (*hidden*).

O problema de reconhecimento de movimentos resume-se então a duas tarefas:

Classificação - Atribuir a uma dada sequência de observações uma classificação;

Treino - Com base num dado número de sequências agrupadas em função do tipo de movimento, gerar os parâmetros para a classificação.

Contudo, é ainda necessário gerar as sequências para a fase de treino. Para isso, foi criado um programa em *GTK* que nos possibilita, com o rato, gerar sequências e agrupá-las em função dos movimentos.

O *Hidden Markov Model*, é um modelo estatístico utilizado exaustivamente em áreas como reconhecimento de fala e escrita com bons resultados. No entanto, como podemos ver [5, 10, 13, 9], é também muito aplicado em reconhecimento de gestos.

Um processo *Markov* é um processo probabilístico que actua sobre um número

finito de estados, { q_1, q_2, ... ,q_N}. As transições entre estados são feitas através de probabilidades denominadas de *probabilidades de transição*. Em cada estado pode ser determinado um caminho dependendo da distribuição de probabilidades associada às probabilidades de transições.

3.2.1 Definição de HMM

Para definir uma cadeia *HMM* são necessários os seguintes elementos:

- O número de estados do modelo, N;
- O número de observações , M;
- O conjunto de transições de estado $\Lambda = \{a_{i,j}\}$ também designado de A

$$a_{i,j} = p\left\{q_{t+1} = j | q_t = i\right\}, \quad 1 \leq i, j \leq N \tag{3.1}$$

em que q_t representa o estado actual. As probabilidades de transição de estado devem satisfazer as condições estocásticas normais

$$a_{i,j} > 0, \quad 1 \leq i, j \leq N \tag{3.2}$$

e

$$\sum_{j=1}^{N} a_{i,j} = 1, \quad 1 \leq i \leq N \tag{3.3}$$

- A probabilidade de uma observação (k) pertencer a um estado, $B = \{b_j\,(k)\}$

$$b_j\,(k) = p\left\{o_t = v_k | q_t = j\right\}, \quad 1 \leq j \leq N, 1 \leq k \leq M \tag{3.4}$$

em que v_k representa a k^{th} observação e o_t o actual vector de observação. Mais uma vez, os parâmetros estocásticos devem ser garantidos:

$$b_j\,(k) > 0, \quad 1 \leq j \leq N \quad 1 \leq k \leq M \tag{3.5}$$

e

$$\sum_{k=1}^{M} b_j\,(k) = 1, \quad 1 \leq j \leq N \tag{3.6}$$

- O estado inicial, $\pi = \{\pi_i\}$ em que:

$$\pi_i = p\{q_1 = i\}, \quad 1 \leq i \leq N \tag{3.7}$$

Concluindo, podemos usar a notação compacta

$$\lambda = (\Lambda, B, \pi) \tag{3.8}$$

para representar uma *HMM*.

3.2.2 Pressupostos da teoria de HMMs

Por uma questão de simplificação matemática e eficiência computacional, são impostos os seguintes pressupostos na teoria dos *HMM's*

Pressuposto Markov Como foi dito na definição de *HMM's*, as probabilidades de transição de estado são dadas por:

$$a_{i,j} = p\{q_{t+1} = j | q_t = i\} \tag{3.9}$$

Por outras palavras, assume-se que o próximo estado depende unicamente do estado actual. Esta relação define o pressuposto de Markov para *HMM* de primeira ordem. Contudo, o estado seguinte (futuro) pode depender de estados passados e pode ser obtido um modelo designado *HMM* de ordem k^{th} definindo as probabilidades de transição como

$$a_{i_1,i_2,...,i_k,j} = p\{q_{t+1} = j | q_t = i_1, q_{t-1} = i_2, ..., q_{t-k+1} = i_k\}, \quad 1 \leq i_1, i_2, ..., i_k, j \leq N \tag{3.10}$$

Notar que o aumento da ordem de HMM incrementa o nível de complexidade do modelo. Por isso, o mais comum é serem utilizadas *HMM* de primeira ordem.

Pressuposto de Estacionaridade Neste item é assumido que a probabilidade de transição de estado é independente do tempo em que ocorre a transição.

Matemáticamente,

$$p\{q_{t_1+1} = j | q_{t_1} = i\} = p\{q_{t_2+1} = j | q_{t_2} = i\} \tag{3.11}$$

para qualquer t_1 e t_2.

Pressuposto de Independência da Saída Neste item é dito que a observação actual é estatisticamente independente da observação anterior,

$$O = o_1, o_2, ..., o_T \tag{3.12}$$

então, de acordo com o assumido, para uma *HMM* λ

$$p\{O | q_1, q_2, ..., q_T, \lambda\} = \prod_{t=1}^{T} p\{O_t, q_t, \lambda\} \tag{3.13}$$

3.2.3 Três Problemas Básicos das HMM's

Depois de se ter definido e caracterizado uma *HMM* podemos definir três problemas de interesse:

Problema da Avaliação Dada uma *HMM* λ e uma sequência de observações $O = o_1, o_2, ..., o_T$, qual é a probabilidade de as observações serem geradas pelo modelo λ, $p\{O, \lambda\}$;

Problema da Descodificação Dada uma *HMM* λ e uma sequência de observações $O = o_1, o_2, ..., o_T$, qual a sequência de estados com maior probabilidade de ter produzido a sequência de observações;

Problema da Aprendizagem Dada uma *HMM* λ e uma sequência de observações $O = o_1, o_2, ..., o_T$, como devemos ajustar os parâmetros do modelo $\{\Lambda, B, \pi\}$ de forma a maximizar $p\{O|\lambda\}$.

Para este trabalho, destes três problemas, os que realmente interessam são o problema da descodificação e o da aprendizagem.

3.2.4 Problema da Descodificação

O problema da descodificação é o mais importante dos dois problemas. A descodificação é nos útil na classificação. Dado um conjunto de observações, $O = o_1, o_2, ..., o_T$, deve-se escolher, de um conjunto de modelos $\{\lambda_1, \lambda_2, ..., \lambda_R\}$, qual é o modelo que mais se ajusta ao conjunto observado O.

Segundo Rabiner [11] existem muitas maneiras de resolver este problema. A dificuldade está na definição de "sequência de estados óptima", isto é, existem vários critérios para julgar a optimabilidade de uma sequência de estados. Por exemplo, um possível critério de optimabilidade é escolher o estado q_t que é individualmente mais semelhante ao observado. Apesar deste método maximizar o número esperado de estados correctos (escolhendo os estados mais semelhantes para cada instante de tempo), podem existir alguns problemas na determinação de uma sequência de estados. Como por exemplo, quando a *HMM* tem uma probabilidade de transição igual a 0 ($a_{i,j} = 0$ para algum i e j), a sequência "óptima" de estados pode nem ser sequer uma sequência válida.

Ainda segundo Rabiner, uma possível solução para resolver o problema da descodificação é modificar o critério de optimabilidade da seguinte forma. Procura-se resolver o problema da sequência de estados, que maximiza o número esperado de pares de estados correctos (q_{t-1}, q_t) ou sequência de tripletes correctos (q_{t-2}, q_{t-1}, q_t), etc. Contudo, apesar de este critério ser razoável para algumas aplicações, o critério mais utilizado é o de encontrar a melhor sequência de estados única, isto é, que maximiza $P(Q, O|\lambda)$. A técnica mais comum para a determinação desta sequência de estados é designada de algoritmo de *Viterbi*.

Segundo o algoritmo de *Viterbi*, para encontrar a melhor sequência de estados $(Q = \{q_1, q_2, ..., q_N\})$ dado uma sequência de observação $(O = \{o_1, o_2, .., o_T\})$ é necessário definir,

$$\delta_t(i) = \max_{q_1, q_2, ..., q_{t-1}} p\{q_1, q_2, ..., q_{t-1}, q_t = i, o_1, o_2, ..., o_{t-1}|\lambda\} \tag{3.14}$$

sendo que $\delta_t(i)$ é a probabilidade mais elevada através de um único caminho, no instante t quando o estado actual é i. Consequentemente, temos que

$$\delta_{t+1}(j) = \left[\max_j \delta_t(i)\, a_{i,j}\right] .b_j(o_{t+1}) \tag{3.15}$$

Para ser devolvida a sequência de estados é preciso seguir o valor que maximiza (3.15), para cada instante de tempo t e j. Isto é conseguido via a utilização da tabela $\psi_t(j)$. Para completar o procedimento para encontrar a melhor sequência de estados é definido o algoritmo de *Viterbi* (algoritmo 2)

Algoritmo 2 Algoritmo de Viterbi

Inicialização

$$\delta_1(i) = \pi_i b_i(o_1) \tag{3.16}$$

$$\psi_1(i) = 0 \tag{3.17}$$

Recursividade

$$\delta_t(j) = \max_{1\le i\le N}\left[\delta_{t-1}(i)a_{i,j}\right] b_j(o_t), \quad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \tag{3.18}$$

$$\psi_t(j) = \operatorname{argmax}_{1\le i\le N}\left[\delta_{t-1}(i)\, a_{i,j}\right], \quad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \tag{3.19}$$

Terminação

$$p^* = \max_{1\le i\le N}\left[\delta_T(i)\right] \tag{3.20}$$

$$q_T^* = \operatorname*{argmax}_{1\le i\le N}\left[\delta_T(i)\right] \tag{3.21}$$

Caminho (sequência de estados) até ao instante t

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right), \quad t = T-1, T-2, ..., 1 \tag{3.22}$$

3.2.5 Problema da Aprendizagem

O problema da aprendizagem é o problema mais complicado dos três problemas das *HMM's*, tendo como objectivo o de ajustar os parâmetros do modelo (A, B, π). Não

existe solução analítica para a resolução do problema de maneira a maximizar a probabilidade da sequência observada dado o modelo (λ). De facto, dada qualquer sequência finita de observações como treino, não existe nenhum método óptimo de estimar os parâmetros do modelo. No entanto, podemos escolher $\lambda = (A, B, \pi)$ tal que $P(O|\lambda)$ é localmente maximizado utilizando métodos iterativos como *Baum-Welch* (ajusta os parametros de modo a maximizar $p(O|\lambda)$) ou utilizando técnicas com base em gradientes. Nesta secção é descrito o algoritmo *K-Means* [8], mais um método iterativo que visa resolver o problema da aprendizagem. O grande objectivo do algoritmo *K-Means* é o de ajustar os parâmetros do modelo (λ) de modo a maximizar a estimação *Likelihood*.

Assim, considerando uma cadeia de Markov de primeira ordem, como tem sido descrito em cima, o algoritmo *K-Means* leva-nos de λ^k para λ^{k+1} tal que

$$p\left(O, I_k^*|\lambda^k\right) \leq p\left(O, I_{k+1}^*|\lambda^{k+1}\right) \tag{3.23}$$

com I_k^* a sequência de estados óptima dada a observação O e λ^k determinado segundo a resolução do problema da descodificação (descrito na secção 3.2.4). Este critério de optimização (3.23) é designado de *maximum state optimized likelihood criterion*. A função $p(O, I^*|\lambda) = \max_I p(O, I|\lambda)$ é designada de *state optimized likelihood function*. Neste método, com o objectivo de treinar o modelo, são necessárias um certo número de observações de treino. Seja ω o número de sequências de treino disponíveis. Cada sequência de observações $O = o_1, o_2, ..., o_T$ contem T observações. É assumido que cada observação o_i é um vector de dimensão $D\,(\geq 1)$. O algoritmo 3 mostra a resolução do problema.

3.3 Descrição do Método Adoptado

Depois de explicada a parte teórica de *HMM's*, nesta secção são descritos alguns assuntos importantes na implementação do detector de movimentos.

A classificação resume-se ao segundo problema das *HMM* que visa resolver o problema da descodificação. O algoritmo de *Viterbi* fornece-nos uma sequência óptima I^* para uma dada sequência de observações. O modelo mais semelhante à sequência

Algoritmo 3 Segmental K-Means

[1 :] Escolher arbitrariamente N elementos observáveis (vectores de dimensão D). Cada um destes elementos vai iniciar a formação de um *cluster* (futuros estados). Associar cada um dos ωT vectores de observações a um dos N *clusters*, a partir da distância euclidiana mínima ($\min_i \|o_t - c_i\|$).

[2 :] Calcular as probabilidades iniciais e as probabilidades de transição:

while $0 \leq i \leq N$ **do**

$$\hat{\pi}_i = \frac{N\acute{u}mero\ de\ ocorrências\ de\ \{o_1 \in i\}}{N\acute{u}mero\ total\ de\ ocorrências\ de\ o_1\ (isto\ é\ \omega)} \tag{3.24}$$

end while

while $1 \leq i \leq N$ & $1 \leq j \leq N$ **do**

$$\hat{a}_{i,j} = \frac{N\acute{u}mero\ de\ ocorrências\ de\ \{o_t \in i\ \&\ O_{t+1} \in j\}\ para\ todo\ o\ t}{N\acute{u}mero\ de\ ocorrências\ de\ O_t \in i\ para\ todo\ o\ t} \tag{3.25}$$

end while

[3 :] Calcular o vector de médias e a matriz de covariâncias para cada estado:

while $0 \leq i \leq N$ **do**

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{o_t \in i} o_t, \quad \hat{\Sigma}_i = \frac{1}{N_i} \sum_{o_t \in i} (o_t - \hat{\mu}_i)^T (o_t - \hat{\mu}_i) \tag{3.26}$$

end while

[4 :] Calcular a distribuição de probabilidade para cada vector de treino para cada estado (assumindo uma distribuição gaussiana):

while $0 \leq i \leq N$ **do**

$$\hat{b}_i (o_t) = \frac{1}{(2\pi)^{D/2} \left|\hat{\Sigma}_i\right|^{1/2}} e^{-\frac{1}{2}(o_t - \hat{\mu}_i)\hat{\Sigma}_i^{-1}(o_t - \hat{\mu}_i)^T} \tag{3.27}$$

end while

[5 :] Encontrar a sequência de estados óptima $I^* = \{i_1^*, i_2^*, ..., i_T^*\}$ (dada pela solução do problema da descodificação) para cada sequência de treino, usando o modelo $\hat{\lambda}_i = \left(\hat{A}_i, \hat{B}_i, \hat{\pi}_i\right)$ calculados nos passos 2, 3 e 4. Um vector de observação é reatribuído a um estado se a sua atribuição realizada originalmente é diferente do estado óptimo estimado correspondente. Concluíndo, iremos atribuir o_t (por exemplo da $k - ésima$ sequência de treino) ao *cluster* (estado) q_t se i_t^* (correspondente à $k - ésima$ sequência de treino) for igual ao estado q_t. Este processo é executado para todas as sequências de treino.

[6:] Se algum vector tiver sido reatribuido a um novo estado no passo 5, usar a nova atribuição, e repetir os passos 2 a 6.

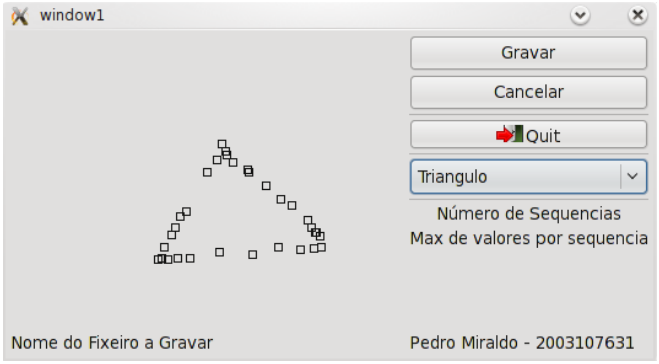


Figura 3.3: Programa criado com o objectivo de gerar as sequências.

I^* é então devolvido.

Já a tarefa relativa ao treino não é tão simples. sendo necessário criar as sequências de treino.

3.3.1 Treino

Esta secção visa resolver o problema do treino utilizando a resolução do terceiro problema das *HMM* (problema da aprendizagem, algoritmo *K-Means*). Para a execução desse algoritmo é necessário gerar sequências de treino para cada um dos movimentos mostrados nas figuras 3.1. Para a criação dessas sequências foi implementado um programa (figura 3.3) que contém um painel do tamanho das imagens adquiridas (320×240 pixeis) onde é possível ao utilizador assinalar posições com o rato. Com isto, é possível gerar trajectórias a partir das posições definidas com o rato. E com as trajectórias é possível extrair a informação da velocidade ao longo da trajectória (vectores que vão ser utilizados no treino). Este trabalho é bastante monótono uma vez que se tem de repetir o processo várias vezes para obter um número de sequências aceitável para o reconhecimento. Foram criadas, para cada movimento, entre sete a quinze sequências de treino e foram estimados os parâmetros dos modelos (λ_i) usando o algoritmo *K-Means*

3.4 Resultados

A elaboração de testes ao reconhecimento de movimentos implica ter de executar o movimento um certo número de vezes e verificar quantas vezes esse movimento foi reconhecido pelo programa. Este trabalho é árduo uma vez que são movimentos que por si demoram algum tempo a realizar, o que faz com que elaborá-los muitas vezes despenda muito tempo.

Foram efectuados vinte movimentos de cada gesto, aleatoriamente, em diferentes velocidades e diferentes comprimentos. O resultado final é apresentado na tabela 3.1.

Movimentos	**Taxa de reconhecimento**
Ondular	85%
Quadrado	95%
Triângulo	100%
Circunferência	70%
L	60%
Média	82%

Tabela 3.1: Tabela com a taxa de reconhecimento dos movimentos.

Capítulo 4

Reconhecimento de Posturas

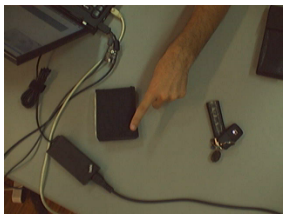
Após ter sido feito o reconhecimento de movimentos, outro aspecto não menos importante para o reconhecimento de um gesto é o reconhecimento de uma pose (postura). Este capítulo mostra o que foi feito para o reconhecimento de posturas de uma mão.

4.1 Introdução

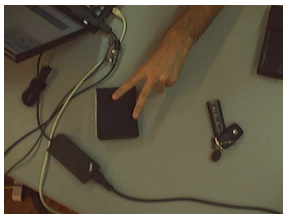
Para o reconhecimento de posturas da mão foi implementado um algoritmo baseado na segmentação dos dedos da mão como foi mostrado por Schwarz em [12].

Existe um vasto leque de algoritmos para realizar o reconhecimento de posturas baseado em visão por computador. Assim como há algoritmos capazes de estimar poses $3D$ como [4], existem também algoritmos para estimar poses $2D$ como [5]. Neste trabalho, por uma questão de simplicidade, optou-se por um algoritmo $2D$ que se baseia na segmentação dos dedos da mão.

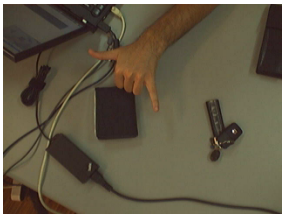
Foi feito o reconhecimento de quatro posturas: *One* 4.1(a), *Two* 4.1(b), *Hang loose* 4.1(c) e *Open* 4.1(d), devidamente treinadas.



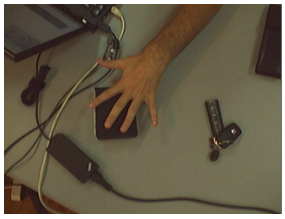
(a) *One.*



(b) *Two.*



(c) *Hang loose.*



(d) *Open.*

Figura 4.1: Posturas escolhidas para a elaboração do reconhecimento.

4.2 Algoritmo de Reconhecimento de Postura da Mão

Este algoritmo tem como ponto de partida uma imagem da mão segmentada e é baseado em erosões e dilatações dessas regiões segmentadas, com o objectivo de remover a palma da mão. Por fim, é feita a classificação da postura da mão com base na distribuição dos dedos.

4.2.1 Segmentação da Mão

A segmentação da mão é diferente da segmentação dos dedos. No primeiro caso, o problema resume-se a separar a mão de todo o fundo da imagem (capítulo 2), como podemos ver na figura 2.1. Enquanto a segmentação dos dedos refere-se à separação dos dedos da palma da mão. Para a segmentação dos dedos é necessária uma boa segmentação da mão.

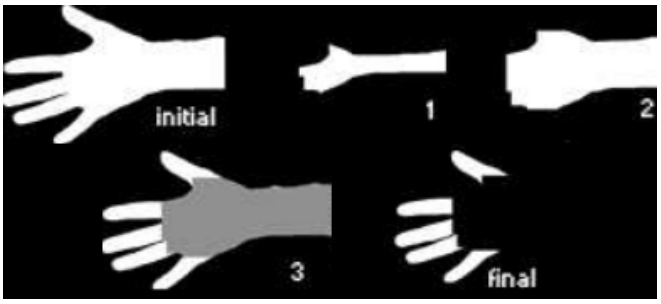


Figura 4.2: "initial" imagem inicial segmentada, (1) depois da erosão, (2) depois da dilatação, (3) comparação da iamgem inicial com a imagem dilatada, "*final*" imagem com os dedos segmentados.

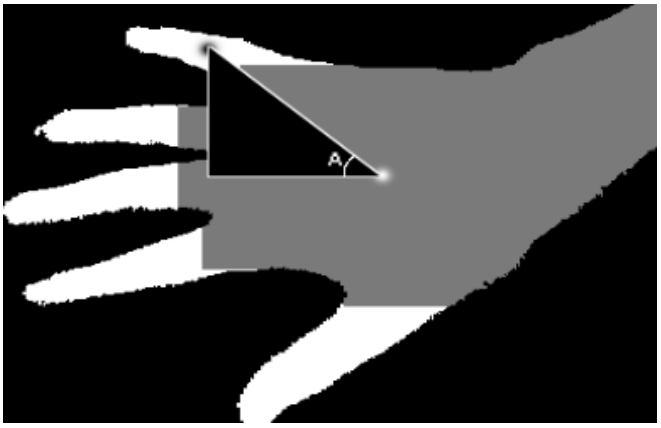


Figura 4.3: Exemplo de ângulos de dedos numa mão.

4.2.2 Segmentação dos Dedos

A segmentação dos dedos consiste na sua separação do resto da imagem. Como já foi dito em cima, essa segmentação é baseada na erosão e dilatação da mão (segmentada do resto da imagem). Tal como se pode verificar na figura 4.2, a subtracção da imagem inicial (*initial*) pela imagem 2, resulta numa imagem unicamente formada pelos dedos da mão (*final*). O algoritmo 4 mostra como é feita a segmentação dos dedos. No final do algoritmo, os dedos são etiquetados.

Algoritmo 4 Segmentação de Dedos

[1 :] É aplicada a erosão à imagem da mão segmentada. A erosão foi feita num raio de 10 pixeis. O raio de erosão ideal a usar pode variar de caso para caso, consoante a resolução da imagem, o tamanho da mão e a distância desta à câmara. No caso concreto desde trabalho a erosão com 10 pixeis foi a que mostrou melhores resultados. O resultado deste passo deve ser a mão, sem dedos, apenas com a parte central da palma da mão.

[2 :] A imagem erodida é agora dilatada através de um raio igual ao determinado no passo anterior. O objectivo é restaurar a palma da mão ao seu tamanho original.

[3 :] É comparada a imagem dilatada com a imagem original. É o momento ideal para obter a erosão e dilatação perfeita. As duas imagens apenas devem diferenciar nos dedos.

[4 :] A nova imagem com os dedos segmentados é dada subtraindo a palma da mão (imagem 2) pela imagem original (*initial*). Com o objectivo de evitar ligações na parte inferior dos dedos, foi ainda feito uma pequena erosão (2 pixel de raio) na imagem *final*.

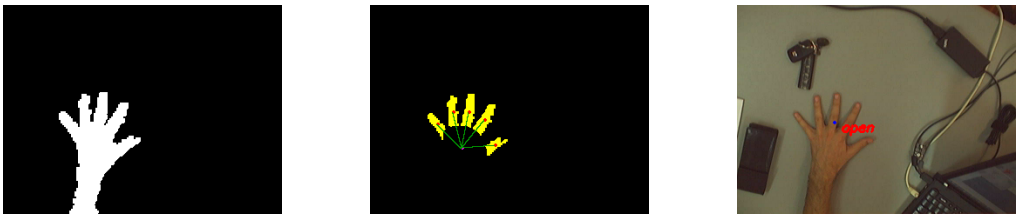
4.2.3 Reconhecimento

Depois de executado o algoritmo de segmentação de dedos resta-nos realizar a classificação da postura em função da distribuição dos dedos.

Neste trabalho optou-se pela classificação através da distribuição angular dos dedos relativamente à orientação da mão. Para isso precisamos da orientação da mão que nos é dada a partir da secção 2. A figura 4.3 mostra um exemplo.

Depois de feita a determinação dos ângulos dos dedos a classificação é o último passo. Para a classificação foi utilizado o método *HMM* já utilizado neste trabalho. Os ângulos são ordenados de ordem crescente para ser feita a classificação.

Os ângulos da esquerda e da direita são inversos. A solução encontrada para a classificação foi efectuar a classificação para os dois casos. Das duas melhores hipóteses, verificar as probabilidades geradas. A que mostrar maior probabilidade é a hipótese correcta (pose).



(a) Mão segmentada. (b) Dedos segmentados. (c) Resultado depois da classificação.

Figura 4.4: Posturas escolhidas para a elaboração do reconhecimento.

As sequências de treino são obtidas a partir de várias observações dos ângulos *offline*.

4.3 Resultados

Na figura 4.4 são mostradas algumas imagens obtidas com o programa criado.

A elaboração de testes ao reconhecimento de posturas é mais fácil que o reconhecimento de movimentos. Como o reconhecimento de posturas é feito em tempo real, é possível retirar resultados mais rapidamente.

Movimentos	**Taxa de reconhecimento**
One	96%
Two	90%
Hang Loose	87%
Open	97%
Média	92.5%

Tabela 4.1: Tabela com a taxa de reconhecimento posturas.

Foram efectuados 100 gestos de posturas aleatoriamente com orientações de mãos diferentes. O resultado final é apresentado na tabela 4.1.

Capítulo 5

Conclusões

Neste capítulo final são descritas as principais conclusões retiradas da realização deste trabalho. E são sugeridos eventuais trabalhos futuros.

5.1 Conclusões

No fim do trabalho realizado pode concluir-se que o recurso à cor, para a segmentação do objecto, obtém bons resultados. Contudo, apesar de ter sido usado um modelo de cor (YCC) que nos possibilita a remoção da componente de luminosidade, o sistema continua ligeiramente sensível às variações de luz. O sistema comprovou a rapidez da segmentação em cor, porém, quando entram em cena objectos com dimensões elevadas (semelhentes à resolução da imagem), o sistema torna-se ligeiramente lento (maior área, mais pixeis para analisar, aumenta exponencialmente o tempo de processamento).

Pode-se tambem concluir que o seguimento de vários objectos simultaneamente tempo baseado na matriz de covariâncias da distribuição dos pixeis obtém bons resultados. Possibilita a junção de objectos sem que a hipótese associada a um deles desapareça. Todavia, não é suficiente para efectuar o cruzamento desses objectos. O *filtro de Kalman* resolve esse problema. Porém, os parâmetros estocásticos do filtro devem ser bem definidos para cada aplicação.

O reconhecimento de movimentos, baseado em velocidade, considerando que os

movimentos podiam seguir o modelo *HMM* mostrou ser robusto. Para um bom reconhecimento é necessário um bom treino. A técnica adoptada para a criação de sequências de treino (utilizando um pequeno programa *GTK* criado) mostrou bons resultados na parte da classificação e simplificou a tarefa, árdua, da criação destas. A escolha de *HMM* deve-se ao facto de reconhecerem com sofisticação padrões que se estendem no tempo, enquanto tanto as redes neuronais como as máquinas de vectores de suporte (*SVM*) apresentam uma eficácia muito elevada em padrões que não os desse tipo.

Conclui-se ainda que o reconhecimento de posturas baseado na segmentação dos dedos foi conseguido com êxito. De notar que este método depende de uma boa segmentação da mão, o que mais uma vez prova o bom resultado da segmentação em cor. A utilização de *HMM* mostrou bons resultados apesar de não ser o método mais apropriado para a aplicação. Poderiam ter sido utilizadas redes neuronais ou máquinas de vectores de suporte.

Para finalizar, tentando dar resposta aos objectivos propostos no capítulo 1, verificou-se que foi possível a implementação de uma interface em tempo real para o reconhecimento de gestos. O programa consegue processar a cerca de 10 $[fps]$ num *centrino* 2.14 $[GHz]$ que prova a possibilidade de interfaces baseadas na interpretação de gestos humanos utilizando visão por computador para interacção homem-máquina.

5.2 Trabalho Futuro

Olhando para o trabalho realizado, um dos tópicos que poderia ser alvo de trabalho futuro é a tarefa da detecção de pixeis cor de pele por duas razões. Por um lado, todo o resto do projecto acenta nesta tarefa, ou seja, quanto melhor estiver a detecção de pixeis chave, melhor vai funcionar a segmentação, seguimento e reconhecimento de gestos. Por outro lado porque o funcionamento do sistema ainda é algo sensível a variações de luz.

Impunham-se alguns melhoramentos no que toca ao reconhecimento de posturas. Poderia ser implementado uma adaptação *online* do raio de erosão e dilatação. De modo a tornar o algoritmo de segmentação de dedos (algoritmo 4) independente

da resolução da imagem, do tamanho da mão e da distância desta à câmara.

Outra possibilidade interessante seria fazer o reconhecimento da postura $3D$ de uma mão, a partir de uma imagem.

Gestos como, "dizer adeus" e "chamar alguém" são muitas vezes constituidos por movimentos e posturas. Por isso, seria interessante introduzir a capacidade de reconhecimento de gestos com fusão de movimentos e posturas.

Bibliografia

[1] P. Alvarado, P. Doerfler, and U. Canzler. Lti image processing library developer's guide. 2003.

[2] A. A. Argyros and M. L. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camara. *European Conference on Computer Vision*, pages 368–380.

[3] K. Astrom and B. Wittenmark. *Computer-controlled systems: theory and design*. 2002.

[4] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. *IEEE CVPR*, 2:432–439, 2003.

[5] J. Carreira and P. Peixoto. Unraveling natural interfaces for co-located groupware: Lessons learned in an experiment with music. *Journal of Multimédia*, 1:18–24, 2006.

[6] I. Corporation. Opencv reference manual. 2001.

[7] W. T. Freeman, K. Tanaka, J. Ohta, and Kyuma. Computer vision for computer games. *Automatic Face and Gesture Recognition*, 14:100–105, 1996.

[8] B. Juang and L. Rabiner. The segmentation k-means algorithm for estimating parameters of hidden markov models. *IEEE ASSP Magazine*, 38(9):1639–1641, 1990.

[9] S. Kettebekov, M. Yeasin, and R. Sharma. Improving continuous gesture recognition with spoken prosody. *Computer Vision and Pattern Recognition*, 1:565–570, 2003.

[10] S.-W. Lee. Automatic gesture recognition for intelligent human-robot interaction. *Automatic Face and Gesture Recognition*, pages 645–650, 2006.

[11] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[12] C. Schwarz and N. V. Lobo. Segment-based hand pose estimation. *IEEE Computer Society*, pages 42–49, 2005.

[13] R. Sharma, J. Cai, S. Chakravarthy, I. Poddar, and Y. Sethi. Exploiting speech/gesture co-occurrence for improving continuousgesture recognition in weather narration. *Automatic Face and Gesture Recognition*, pages 422–427, 2000.

[14] Y. Yuan, Y. Liu, and K. Barner. Tactile gesture recognition for people with disabilities. *Acoustics, Speech, and Signal Processing*, 5:461–464, 2005.