Sistema atencional para detecção e rastreamento de faces

Leonardo Pinto da Silva Panta Leão [1]
Tunai Porto Marques [1]
Leonardo Nogueira Matos [1]

Resumo: O sistema visual humano realiza rapidamente tomadas de decisões complexas devido, em parte, ao sistema atencional, que posiciona alvos de maior relevância no centro do campo visual, região com maior concentração de células fotorreceptoras. O sistema atencional envolve elementos sensoriais, cognitivos e também mecânicos, pois a musculatura do olho e da cabeça precisa ser acionada para produzir movimentos. Neste trabalho apresentamos a proposta de um sistema de detecção e rastreamento de faces que, tal como no sistema biológico, produz um movimento coordenado com o propósito de posicionar a imagem do alvo detectado no centro do campo visual da câmera. O sistema desenvolvido possui partes distintas, uma responsável pelo reconhecimento de padrões em vídeo e outra pelo controle da parte mecânica, implementadas como processos que se comunicam através de *sockets*.

Palavras-chave: Detecção de objetos. Rastreamento de face. Motor de passo.

Abstract: *The human visual system quickly performs complex decisions due, in part, to attentional system, which positions the most relevant targets in the center of the visual field, region with greatest concentration of photoreceptor cells. The attentional system involves sensory, cognitive and also mechanical elements, because the eye and head muscles must be activated to produce movement. In this paper we present the proposal of a face detector system that, as well as the biological system, produces a coordinated movement with the purpose of positioning the target image in the center of camera's visual field. The developed system has distinct parts, one responsible for video pattern recognition and other for controlling the mechanical part, implemented as processes that communicate with each other by sockets.*

Keywords: *Object detection. Face tracking. Stepper motor.*

1 Introdução

Algoritmos mais sofisticados de visão computacional vêm sendo usados em aplicações progressivamente complexas e com diversas finalidades, tais como detecção de movimento, rastreamento de objetos, reconhecimento de face, dentre outras. Uma dessas aplicações é a detecção e rastreamento de seres humanos em vídeo, o que permite, por exemplo, tornar um sistema de vigilância [5] capaz de realizar o movimento da câmera para acompanhar indivíduos de maneira automática.

A tarefa de detectar seres humanos em imagens é bastante estudada na literatura. Existem algoritmos que detectam faces humanas de forma bastante eficiente [14], apesar da imposição de algumas limitações, decorrentes, por exemplo, da aquisição de imagens de perfil ou parcialmente ocluídas. Por outro lado, a tarefa de rastrear um objeto previamente identificado é um pouco menos complexa e tem um grau maior de eficiência computacional. As abordagens mais comuns são baseadas em filtro de Kalman [9], para rastreamento baseado em movimento, ou histograma backprojection [2], para rastreamento baseado na cor.

Este trabalho descreve um sistema de visão de tempo real que detecta a face de um indivíduo usando o algoritmo de Viola e Jones [14], realiza o rastreamento com base no algoritmo de Bradski [2] e controla o acionamento de motores que movimentam a câmera com o propósito de manter a imagem do indivíduo sempre no centro do vídeo.

[1]Departamento de Computação, Universidade Federal de Sergipe (UFS), São Cristóvão (SE) - Brasil
{{leonardopspl, tunaip}@gmail.com, lnmatos@ufs.br}

doi: 10.5335/rbca.2011.006

2 Sistema de atenção visual

O sistema de visão desenvolvido neste trabalho, se comparado ao que os seres humanos fazem, imita o sistema atencional, que procura manter sempre o foco da visão em um objeto de interesse, que neste caso é uma face. Para isso é realizada, inicialmente, uma etapa de segmentação, isto é, de distinção do objeto de interesse, ou *foreground*, dos demais artefatos da imagem, *background*. Na etapa seguinte o objeto é continuamente acompanhado através do algoritmo de rastreamento. Eventualmente, durante essa etapa o campo de visão deve ser movimentado a fim de manter o objeto visível e centralizado.

Neste trabalho usamos um detector de faces que utiliza máscaras convolucionais inspiradas nas funções de Haar [14]. A implementação dessa técnica está disponível na biblioteca de visão computacional OpenCV [3], que possui licença livre. Essa técnica é bastante eficaz e eficiente para detecção de faces humanas, e por isso tem sido utilizada para segmentação online de faces em vídeo.

Para a etapa de rastreamento utilizamos um algoritmo que se baseia na cor do ser-humano filmado. Com base na posição da face no quadro vídeo anterior e na informação de cor, o algoritmo de rastreamento obtém rapidamente a posição da face no quadro atual. Com isso é possível passar para uma plataforma de hardware instruções de controle que irão movimentar um motor acoplado à câmera. As duas técnicas citadas e a plataforma controladora da câmera serão explicadas nas próximas seções deste artigo.

3 Detecção da face

O método de segmentação proposto por [14] e discutido em [10] é uma técnica de detecção baseada na aparência do objeto. Métodos desse tipo aprendem características a partir de conjuntos de imagens de treinamento que capturam a variedade da classe do objeto. Como sugerido em [15], esse método pode ser usado para detectar faces humanas. Nesse método é feita uma etapa inicial de treinamento, onde ocorre a seleção de dois conjuntos de imagens, um positivo e outro negativo. O conjunto de imagens positivas contém recortes onde está contido apenas o objeto a ser detectado; já o conjunto de imagens negativas são imagens que não contêm o objeto, geralmente paisagens onde o objeto pode ser encontrado. Em seguida, é feito um conjunto de exemplos onde as imagens positivas são combinadas com as negativas para formar imagens de provável aparição do objeto. O processo de extração de características é baseado em máscaras convolucionais inspiradas em funções de Haar [8] (Figura 1). Essas características são calculadas pela convolução na imagem do objeto baseadas numa decisão binária a partir de um *threshold*.

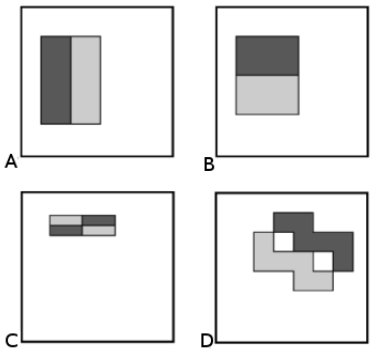

Figura 1. Exemplos de características retangulares [14]

Na Figura 1 duas características retangulares estão representadas em (A) e (B), as quais capturam transições de nível de cinza verticais e horizontais. Em (C) e (D) são mostradas máscaras que capturam características diagonais.

Na etapa de treinamento o método utiliza o algoritmo AdaBoost [7] para construir o classificador. O objetivo desse algoritmo é construir um classificador eficiente a partir de uma série de classificadores fracos baseados em

decisões binárias a partir da convolução dessas máscaras na imagem do objeto. O uso de características reduz a variedade de dados na classe e aumenta a variedade de dados fora da classe em comparação a dados de entrada crus. Além disso, as características geralmente codificam conhecimento sobre o domínio, o que é difícil aprender a partir de um conjunto cru e finito de dados. Em uma imagem de 64 por 64 pixels temos um conjunto de mais de 120 mil características possíveis.

Para o cômputo rápido dessas características é utilizado uma imagem, chamada imagem integral, descrita em detalhes no trabalho de [10]. Essa imagem é construída utilizando o princípio da programação dinâmica, onde cada pixel $(\imath, \jmath)$ guarda a soma dos valores de todos os pixels do canto superior esquerdo (a origem) até o pixel corrente, isto é, o pixel $(\imath, \jmath)$. Usando a imagem integral, é possível realizar o cálculo de uma característica em tempo constante.

O AdaBoost seleciona as características que melhor classificam os objetos e chama um classificador fraco repetidamente numa série de turnos. Para cada chamada, uma distribuição de pesos é atualizada para indicar a importância de alguns exemplos no conjunto de dados para a classificação. A cada turno, os pesos de cada exemplo incorretamente classificado é incrementado, de maneira que o novo classificador tenha um maior foco nesses exemplos. Assim, após selecionar um classificador ótimo, baseado nessas características e nessa distribuição de pesos, os exemplos que o classificador classifica incorretamente têm seus pesos aumentados e os exemplos classificados corretamente têm seus pesos diminuídos. Consequentemente, quando o algoritmo testa uma nova distribuição de pesos, irá selecionar um classificador que melhor identifica esses exemplos que o classificador anterior errava. Cada classificador forte obtido a partir desses conjuntos de características é posto numa estrutura de cascata de rejeição na ordem do menos complexo (menos características usadas) para o mais complexo (mais características usadas).

Na etapa de detecção utilizamos a arquitetura de cascata de rejeição, como é mostrado na Figura 2. A entrada é passada pelo primeiro classificador, que decide entre verdadeiro ou falso (objeto encontrado ou não encontrado). Uma determinação de falso interrompe computação posterior e faz com que o detector retorne falso. Uma determinação verdadeira passa a entrada para o próximo classificador na cascata. Se todos os classificadores votarem em verdadeiro, a entrada é classificada como um exemplo verdadeiro. Dessa maneira, podemos economizar vários ciclos computacionais, já que, se um dado de entrada é rejeitado logo no início, toda a computação posterior nos próximos nós é evitada.

A detecção ocorre numa janela deslizante desde o canto superior esquerdo da imagem até o canto inferior direito. A cada término de deslizamento na imagem, a janela de detecção é aumentada 20% do seu tamanho inicial, até que a janela enquadre ou exceda o tamanho da imagem. Caso mais de uma face seja encontrada praticamente na mesma região, já que poderá ser encontrada tanto na janela de detecção menor quanto na próxima aumentada, essas detecções são mescladas e consideradas uma só.

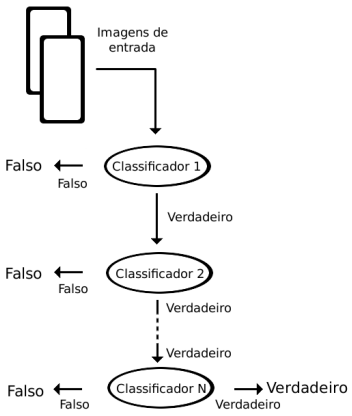

Figura 2. Modelo em cascata do algortimo de Viola e Jones [14]

4 Rastreamento rápido da face

O algoritmo de rastreamento rápido da face utilizado é o CamShift, que se baseia na cor do objeto para distingui-lo do fundo. Este algoritmo foi proposto por [2] e discutido em [1]. A implementação empregada nos experimentos admite como região de interesse aquela obtida pelo uso do método de detecção executado na etapa anterior.

Para a tarefa de rastreamento o Camshift utiliza um histograma unidimensional consistindo em canais quantizados do espaço de cores HSV, que é um modelo de cores baseado em três canais: a matiz (*hue*), que representa a cor, a saturação (*saturation*), que representa o quão concentrada a cor se encontra, e o valor (*value*), que representa o brilho da cor. A Figura 3 ilustra, resumidamente, o espaço de cores do HSV. Esse algoritmo foi desenvolvido tendo como objetivo, além do rastreamento de objetos de interesse, o consumo do menor número possível de ciclos de CPU. Então, apenas um canal (a matiz) é considerado no modelo de cor.

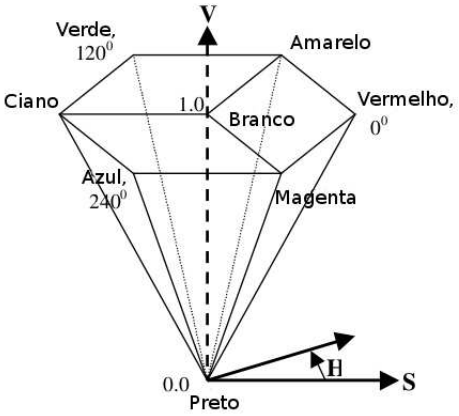

Figura 3. Espaço de cores do HSV

Camshift significa Continuously Adaptive Mean Shift (Mean Shift continuamente adaptativo). Trata-se, portanto, de uma extensão do Mean Shift [4], que será explicado mais adiante. Segundo [2] o Camshift pode ser resumido nos seguintes passos:

1. Selecione a região de interesse da imagem de distribuição de probabilidade como sendo a imagem inteira.
2. Selecione a localização inicial da janela de busca do Mean Shift. A localização dessa região será a distribuição alvo a ser rastreada (nesta etapa usamos o detector descrito anteriormente).
3. Calcule a distribuição de probabilidade de cor centrada na janela de busca do Mean Shift.
4. Itere o algoritmo do Mean Shift para encontrar o centróide (centro de massa, ponto que representa a média da distribuição de cor do alvo) da imagem de probabilidade. Armazene o momento zero (área da distribuição) e localização do centróide.
5. Para o próximo frame, centre a janela de busca na localização encontrada no passo 4 e selecione o tamanho da janela como sendo um função do momento zero. Vá para o passo 3.

A fim de construir a imagem de probabilidade que é utilizada no algoritmo do CamShift usamos um método que associa um valor de pixel correspondente à probabilidade que o dado pixel pertença ao alvo a ser rastreado. No Camshift, um método conhecido como Histograma Back-Projection é utilizado. Para gerar essa imagem de distribuição de probabilidade computamos um histograma inicial da região de interesse inicial da imagem filtrada no passo 1 do algoritmo do CamShift.

O histograma é quantizado em bins, o que reduz a complexidade espacial e computacional e permite que valores de cor similares sejam agrupados juntos. Os bins desse histograma são escalados entre a intensidade mínima e máxima da imagem de probabilidade. Dessa maneira, quando fazemos uma projeção do histograma alvo com quaisquer frames consecutivos, geramos uma imagem onde o valor de cada pixel caracteriza a probabilidade de que aquele pixel na imagem original pertença ao histograma que foi utilizado. Dado m-bins de entrada do histograma, definimos n localizações de pixels na imagem $\{x_i\}_{i=1...n}$ e o histograma $\{q_u\}_{u=1...m}$. Definimos também uma função *c :* $\Re^2 \longrightarrow \{1...m\}$, que associa ao pixel na localização x_i^* o índice do bin do histograma $c : (x_i^*)$, como apresentado na Eq (1). Cada bin do histograma recebe a quantidade de pixels com aquela respectiva intensidade de cor.

$$q_u = \sum_{i=1}^{n} \delta[c(x_i^*) - u] \tag{1}$$

Após a construção do histograma, escalamos os bins do histograma para um intervalo discreto para a construção da imagem de distribuição de probabilidade 2D usando a Eq (2).

$$\left\{ q_u = min\left(\frac{255}{max(q)} q_u, 255\right)\right\}_{u=1...m} \tag{2}$$

O resultado é uma imagem onde os *pixels* mais prováveis de pertencer ao objeto rastreado tenham intensidades visíveis na imagem de probabilidade 2D.

O Mean Shift, descrito em detalhe em [4] e [6], é uma técnica robusta e não paramétrica que escala o gradiente de uma distribuição de probabilidade a fim de encontrar a média (pico) da distribuição. Assim, o CamShift é baseado numa adaptação do Mean Shift, que, dada uma imagem de densidade de probabilidade, encontra a média da distribuição iterando na direção do crescimento máximo da densidade de probabilidade. Mais precisamente, a parte do Mean Shift dentro do algoritmo do CamShift é o cálculo do centro de massa do objeto, ou seja, o ponto do objeto que representa a média da intensidade de cor de todo esse objeto. Essa localização da média, o centroide, é encontrada na janela de busca da imagem de probabilidade discreta computada no passo 3 do algoritmo utilizando momentos [2]. A área da janela de busca é uma função do momento zero calculado a cada iteração. Dado que $I(x, y)$ seja a intensidade da imagem de probabilidade discreta na localização (x, y) da janela de busca:

1. Compute o momento zero
$$M_{00} = \sum_x \sum_y I(x, y)$$

2. Encontre o primeiro momento para x e y
$$M_{10} = \sum_x \sum_y xI(x, y)$$
$$M_{01} = \sum_x \sum_y yI(x, y)$$

3. Compute a janela de busca da média
$$x_c = \frac{M_{10}}{M_{00}}; y_c = \frac{M_{01}}{M_{00}}$$

Os valores de (x_c, y_c) são continuamente recomputados até que não haja deslocamento significativo em sua posição. Podemos usar como critério de convergência para término da etapa iterativa do Mean Shift um deslocamento mínimo de um pixel na direção horizontal ou vertical. Segundo [2], o número máximo de iterações do Mean Shift é, geralmente, de 10 a 20 iterações.

Além disso, o algoritmo deve terminar no caso onde M_{00} é zero, que corresponde a uma janela consistindo totalmente de intensidade zero (preto).

A Figura 4 ilustra um diagrama de blocos que resume o algoritmo do CamShift.

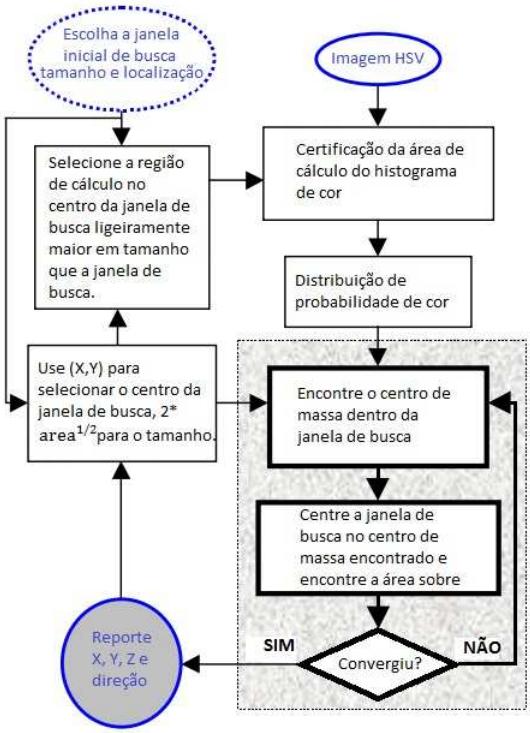

Figura 4. Diagrama de blocos do algoritmo CamShift (Adaptado de [2])

5 Plataforma do motor de passo

Com a aplicação descrita, fomos capazes de incrementar a habilidade de rastreamento do objeto de interesse simulando os movimentos feitos pela musculatura dos olhos e da cabeça, mantendo-o sempre no centro da área de captura. Esse dispositivo, construído usando um motor elétrico, é acoplado à câmera e faz movimentos no eixo horizontal, conferindo liberdade à câmera.

Para construí-lo foi utilizado um motor passo, onde é acoplada a câmera (Figura 6). Escolhemos este tipo de motor elétrico pois, além de possuir precisão elevada e torque considerável, é bastante compacto. Para controlar seu movimento basta gerenciar bits que são enviados a ele, como será detalhado adiante. Para isso, primeiro foi projetado um software que envia, via USB, uma sequência de bits para um circuito que converte a informação serial em paralela [11] (Figura 5).

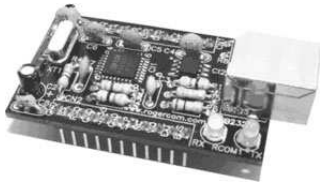

Figura 5. USB232-RCOM1

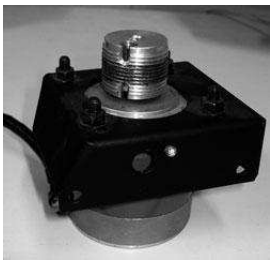

Figura 6. Motor de passo utilizado

A sequência de bits enviada para o circuito conversor contém a indicação de quais bobinas do motor de passo [13] serão energizadas. A energização das bobinas faz com que o rotor do motor de passo gire, criando um torque que é responsável pelo movimento (Figura 7).

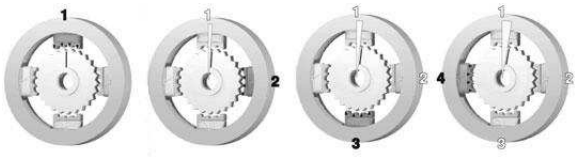

Figura 7. Excitação dos solenoides pela corrente elétrica

A sequência serial de bits que sofreu a conversão é recebida por uma matriz de contatos (*protoboard*), onde é gerenciada e segue para o motor de passo. Entretanto, antes de essa informação chegar ao motor, um hardware específico responsável pelo controle de correntes maiores do que a fornecida pela protoboard atua para alimentar o motor de passo. Este hardware é um array de transistores Darlington. Nos experimentos usamos o circuito integrado ULN 2003, que controla correntes de até $500mA$.

O programa que detecta, rastreia e dá as coordenadas do objeto é independente do programa que faz o controle do motor de passo. Esses programas são partes de uma aplicação cliente-servidor, onde o detector de faces atua como cliente, e o controlador do motor, como servidor. A comunicação entre esses dois processos é feita por *raw socket* [12] em um sistema GNU/Linux. Dessa forma, os dois tornam-se capazes de operar em conjunto. A Figura 8 mostra, resumidamente, o esquema de comunicação entre o programa cliente (o que realiza a detecção e rastreamento do indivíduo) e o programa servidor. A arquitetura cliente-servidor utilizada será detalhada a seguir.

5.1 Aplicação cliente-servidor

Como há duas aplicações envolvidas neste trabalho, houve a necessidade de criar uma solução capaz de gerenciá-las em paralelo. Desenvolvemos um sistema de cliente-servidor, onde o programa de reconhecimento atua como cliente e o programa de movimentação do motor atua como servidor, assim, é possível subordinar os movimentos do motor aos comandos (coordenadas) fornecidos pelo programa de rastreamento. Tais comandos são enviados seguindo um protocolo de comunicação, que é detalhado na Seção 5.2. Esta solução é baseada no uso de sockets.

Socket (ou soquete, em português) é um termo usado para designar uma estrutura física, bem como uma abstração de baixo nível responsável pela comunicação entre computadores [12]. No escopo deste trabalho, o soquete é usado para designar a interface que oferece um elo bidirecional de comunicação entre dois processos independentes. Com ele é possível escrever e ler bytes como um fluxo de dados e, então, estabelecer e gerenciar conexões entre máquinas [12].

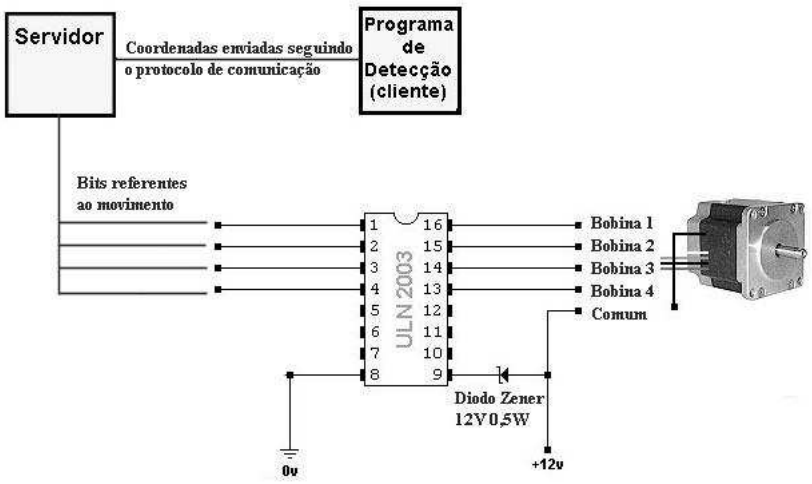

Figura 8. Esquema completo do funcionamento da plataforma que movimenta a câmera

5.2 Protocolo de comunicação

O programa de rastreamento é capaz de determinar qual a distância do objeto de interesse do centro do vídeo. Esses dados são enviados bit a bit para o programa servidor, de forma que cada bit equivale a uma coordenada que indica em que sentido o motor deve girar para manter o objeto de interesse no foco da área de captura. A comunicação entre os processos segue o protocolo descrito na Tabela 1.

Tabela 1. Protocolo de comunicação entre as aplicações

Bit recebido	Significado	Sentido do movimento do motor
-1	O objeto está à esquerda do centro da área de captura.	Anti-horário.
0	O objeto está no centro da área de captura.	Não há movimento.
1	O objeto está à direita do centro da área de captura.	Horário.

6 Experimentos

Nos testes do sistema proposto foram utilizados os seguintes itens:

- Webcam com resolução de 1,2 megapixels.
- Motor de passo com precisão de 1,8 graus por passo e tensão de 6V.
- Protoboard de 2420 furos.
- Placa conversora serial-paralelo.
- Desktop de 2Ghz e 1GB de memória RAM usando o sistema operacional Linux, distribuição Ubuntu 9.10.

A Figura 9 mostra a matriz de contatos onde foi montado o circuito controlador do motor.

Para a efetivação do trabalho proposto, posicionamos a face da pessoa no campo de visão da imagem adquirida pela câmera. Após a detecção da face, o algoritmo de rastreamento pela cor (CamShift) é ativado e passa a enviar para o motor a informação do movimento correto da câmera, para que ela continue a ter o indivíduo no centro do vídeo (Figura 10).

Fizemos diversos testes onde o objeto de interesse percorre uma área de $180°$, sendo acompanhado pela câmera, sempre tentando mantê-lo no centro da área de captura. Com isso, foi possível determinar a velocidade

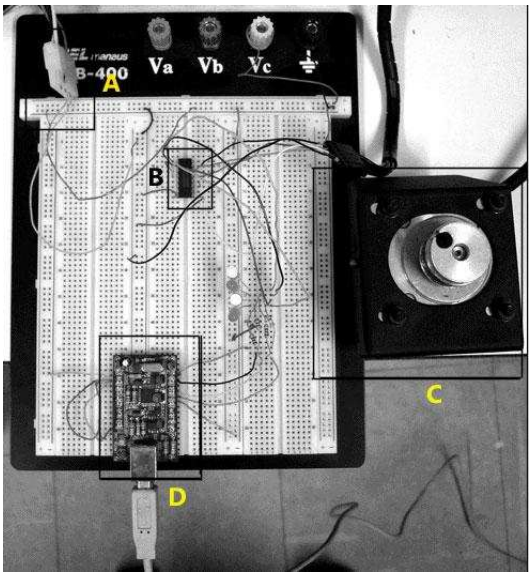

Figura 9. Circuito de controle do motor de passo: (A) Fonte externa de tensão, (B) CI ULN 2003, (C) Motor de passo, (D) Placa conversora serial-paralelo

média de varredura do sistema. Para isso, o tempo gasto pelo sistema para realizar a varredura de $180°$ foi medido repetidas vezes. Utilizamos 20 amostras, 10 com sentido horário e 10 com sentido anti-horário, como é apresentado na Tabela 2.

Tabela 2. Tempo gasto pelo sistema para varrer uma área de $180°$

Amostra	Sentido horário	Sentido anti-horário
1	$43,7s$	$35,5s$
2	$36,4s$	$35,6s$
3	$41,0s$	$33,9s$
4	$35,7s$	$31,6s$
5	$41,2s$	$32,7s$
6	$45,2s$	$37,0s$
7	$33,0s$	$37,5s$
8	$38,7s$	$36,6s$
9	$41,4s$	$35,2s$
10	$32,7s$	$35,0s$
Incerteza da Medida	$0,05s$	$0,05s$
Desvio Padrão	$4,32$	$1,87$
Média com Incertezas	$38,9 \pm 4,37$	$35,06 \pm 1,92$

Constatamos experimentalmente que a velocidade de varredura é de aproximadamente $4,9°$ por segundo. Nos experimentos também constatamos algumas limitações do sistema. O algoritmo de rastreamento pode perder a região de interesse, caso haja uma oclusão prolongada do indivíduo, ou se o indivíduo estiver próximo, na imagem, de algum objeto de cor semelhante à sua pele, por exemplo, se algum outro indivíduo tomar frente no vídeo.

7 Conclusões

Este trabalho apresentou um sistema de visão computacional que realiza o enquadramento da face de um indivíduo no centro do campo visual da câmera, simulando dessa forma o sistema atencional humano, que procura

Figura 10. Em (A) temos a etapa de detecção da face do indivíduo. Em (B) são mostrados o sistema de detecção e rastreamento e a câmera de vídeo

projetar o objeto de atenção no centro do campo visual. Utilizamos uma abordagem dividida em duas etapas, ambas baseadas em algoritmos implementados na biblioteca OpenCV. A construção do sistema de controle e acionamento do motor foi baseada em uma arquitetura cliente-servidor em que os processos se comunicam utilizando sockets, já que nesse caso não é necessário usar a pilha de protocolos TCP/IP, tendo em vista que cliente e servidor são executados na mesma máquina. A arquitetura cliente-servidor, que utiliza módulos coesos e fracamente acoplados, melhora a escalabilidade do sistema, pois favorece o desenvolvimento de extensões e melhorias, desde que preservado o protocolo de comunicação entre os módulos. Como trabalhos futuros, pretendemos investigar formas de realizar o reconhecimento e rastreamento de outras partes do corpo e propor uma solução para detecção do ser-humano por inteiro.

Referências

[1] ALLEN, G.; XU, R.; JIN, J. *Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces*. In: NSW 2006, 2006.

[2] BRADSKI, G. *Computer Vision Face Tracking For Use in a Perceptual User Interface*. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 1998.

[3] BRADSKI, G.; KAEHLER, A. *Learning OpenCV Computer Vision with the OpenCV Library*. OReilly Media, 2008.

[4] CHENG, Y. Mean Shift, Mode Seeking, and Clustering. *IEEE Transactions on Pattern Alnalysis and Machine Intelligence*, v. 17, n. 8, 1995.

[5] COLLINS, R.; LIPTON, A.; KANADE, T. A System for Video Surveillance and Monitoring. In: American Nuclear Society Eighth International Topical. *Proceedings*. Meeting on Robotics and Remote Systems, 1999.

[6] COMANICIU, D. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions On Pattern Alnalysis And Machine Intelligence*, v. 24, n. 5, 2002.

[7] FREUND, Y.; SCHAPIRE, R. A short introduction to boosting. *Journal Of Japanese Society For Artificial Intelligence*, 1999.

[8] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing.* 2nd Edition. Prentice Hall, 2002.

[9] JONG-YUN, K.; TAE-YONG, K. Soccer Ball Tracking Using Dynamic Kalman Filter with Velocity Control. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS, IMAGING AND VISUALIZATION, VI. *Proceedings*. 2009.

[10] LIENHART, R.; MAYDT, J. *An Extended Set of Haar-like Features for Rapid Object Detection*. In: ICIP, 2002.

[11] MESSISAS, R. **Controle de Motor de Passo através de Porta Paralela**. Disponível em: <http://www.rogercom.com/pparalela/IntroMotorPasso.htm>. Acesso em: 04 jun. 2010.

[12] NEIL, M.; STONE, R. *Beginnig Linux Programing*. Wrox Press, 1996.

[13] PATSKO, L. **Tutorial Controle de Motor de Passo**. Disponível em: <http://www.pdfescape.com/open/?116203>. Acesso em: 04 jun. 2010.

[14] VIOLA, P.; JONES, M. Rapid Object Detection using a Boosted Cascade of Simple Features. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, VII. *Proceedings*. 2001.

[15] VIOLA, P.; JONES, M.; SNOW, D. Detecting Pedestrians Using Patterns of Motion and Appearance. In: IEEE International Conference On Computer Vision, IX. *Proceedings*. 2003.