

Move-In: Uma API para Interação com Webcam

Jeferson José de Miranda*

Marcelo da Silva Hounsell

Alexandre Gonçalves Silva

LARVA – Laboratório de Realidade Virtual Aplicada, DCC – Departamento de Ciência da Computação, UDESC – Universidade do Estado de Santa Catarina, Joinville, SC, Brasil

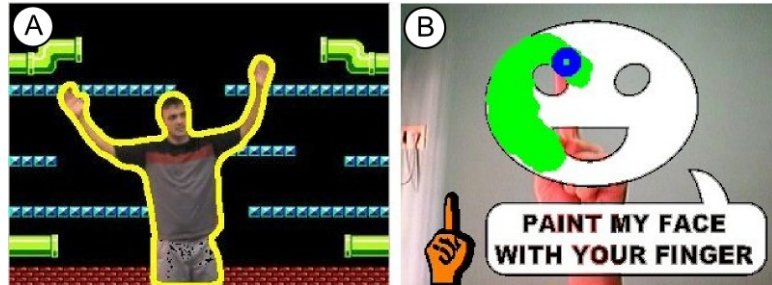


Figura 1: Aplicações da API Move-In.

Resumo

Este artigo descreve a implementação de uma API, denominada Move-In, de código aberto criada para facilitar o desenvolvimento de aplicações onde a comunicação com o *software* se dá através do movimento do corpo do usuário capturado com uma *webcam*, sem o uso de marcadores. As funcionalidades resultantes tornam possível a identificação da presença e do movimento de um usuário em frente a uma *webcam*; a interação com objetos virtuais e; a identificação de certas posturas do corpo e da mão humana. Testes mostraram que a API pode ser usada para prototipação de aplicações com captura de vídeo.

Palavras-chave: processamento de imagens, visão computacional, captura de vídeo, captura de movimento.

Abstract

This paper describes the implementation of an open source API, named Move-In, created to ease the development of applications where the communication with the software is given by the users' body motion captured with a webcam without markers. The resulting functionalities make it possible to identify user's presence and movements in front of the webcam; the interaction with virtual objects and; the identification of some human hand or body postures. Tests have shown that the API can be used for prototyping video captured applications.

Keywords: image processing, computer vision, video capture, motion capture.

Authors' contact:

*jeferson.miranda@gmail.com
{marcelo,alexandre}@joinville.udesc.br

1. Introdução

Uma forma de facilitar a importação de recursos de processamento de imagens e visão por computador para ter uma mais fácil integração a jogos, sem que o projetista tenha que se tornar um especialista, está se tornando cada vez mais necessária. Neste contexto, este trabalho apresenta a API Move-In que tem o objetivo de facilitar a aquisição da imagem de entrada, a segmentação da área de interesse e a extração das características. A análise das características é de responsabilidade do projetista do *software* que fará interação com a *webcam*.

A API Move-In implementa quatro técnicas de identificação de movimento ou de presença e uma forma de gerenciar a região da imagem de entrada afetada por essas técnicas. Dentro do contexto deste trabalho, a presença deve ser considerada como a detecção do usuário (parado) em frente à câmera. Já o movimento se refere à detecção de mudanças de posição entre uma sequência de imagens.

A imagem de entrada da API são os quadros capturados pela câmera. Pode-se definir N regiões de interesse (*Region Of Interest* – ROI) para partes específicas da imagem, como um botão virtual no canto superior esquerdo ou um objeto virtual que se movimenta pelo centro da tela. Na API Move-In as ROIs são espaços retangulares não orientados. Isso facilita a implementação, otimiza a execução e resulta em dados equivalentes à implementação de regiões com formato arbitrário. Fica a cargo do programador definir o conjunto de ROIs que será associado ao objeto virtual.

2. Implementação

O conjunto de funcionalidades, que constitui o resultado deste trabalho, foi implementada sobre a API

OpenCV [Intel 2006], escolhida pela sua eficiência, número de algoritmos de processamento de imagens e visão computacional implementados e pela possibilidade de utilização com *software* comercial.

A API Move-In é um conjunto de classes codificadas em C++ que foi construída utilizando como base as funções estruturadas da versão 1.0 da API OpenCV. A API Move-In está organizada em quatro conjuntos de classes, ou módulos do sistema: **Classes auxiliares**: constituem abstrações para o paradigma orientado a objetos de funcionalidades do OpenCV referentes à criação de imagens e conexão com *webcams*; **Estruturas de dados**: agrupa as classes criadas para armazenar os dados referentes às características extraídas das imagens; **Gerenciamento de regiões de interesse (ROI)**: gerencia as regiões retangulares que limitam o cálculo de segmentação da área de interesse a uma região específica da imagem capturada pela câmera; **Classes do núcleo do sistema**: agrupa as classes responsáveis pela segmentação da área de interesse e extração das características (objetivo deste trabalho).

Existe uma classe, a classe Screen, que é a única que não faz parte de um dos módulos do sistema pelo fato de que ela centraliza o acesso às principais funcionalidades. Isso atende ao padrão de projeto (*design pattern*) *facade* [Gamma et al. 1995].

Uma das características que deve ser observada na hora da escolha da webcam para uso com a técnica de presença na API Move-In é a possibilidade de desabilitar o ganho automático dos níveis de branco. Isso porque o resultado da diferença entre quadros é efetuado a partir das variações de intensidade dos valores RGB, e a mudança brusca na configuração do cenário resulta em grandes diferenças de intensidade de iluminação entre o quadro que representa o cenário de fundo e o último quadro capturado.

2.1 Formas de Segmentação

A API Move-In utiliza duas formas de segmentação da área de interesse, a saber: (1) diferença entre quadros de animação e (2) identificação da cor da pele. A partir da diferença entre quadros pode-se obter dados de presença ou movimento. A Figura 2 expõe as formas de segmentação do tipo 1.

A Figura 2 mostra, respectivamente, o quadro inicial capturado com evento de teclado e contendo somente o cenário de fundo (Figura 2.A), o quadro N-1 (Figura 2.B), o quadro N (Figura 2.C), a diferença entre N e N-1 (Figura 2.D), a diferença entre o quadro inicial e o quadro N no espaço de cor RGB (Figura 2.E), e essa mesma diferença no espaço de cor RGB normalizado (Figura 2.F). A primeira segmentação (Figura 2.D) identifica movimento e as duas últimas (figuras 2.E e 2.F), presença.

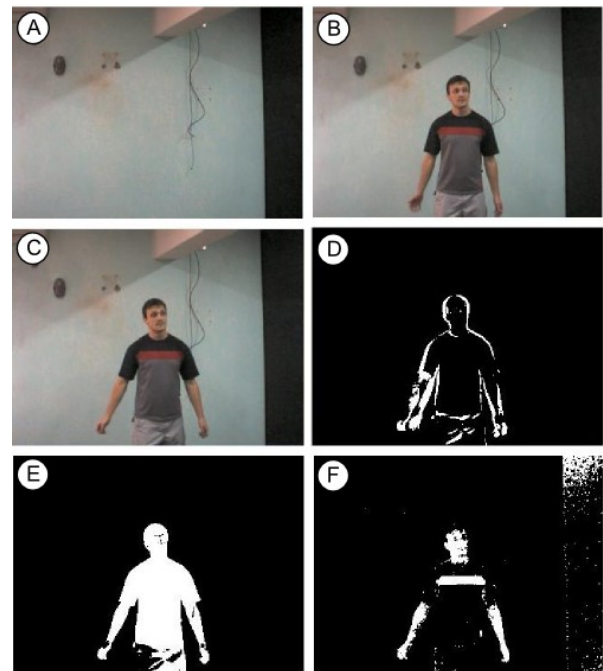


Figura 2: Formas de segmentação da área de interesse.

A segmentação obtida a partir da diferença entre quadros depende do ajuste de um limiar ou fator de corte. As intensidades de nível de cinza resultantes da imagem subtraída variam de 0 a 255. Na binarização dessa imagem, valores menores ou iguais que o limiar são tratados como branco (255) e valores acima do limiar são tratados como preto (0).

Para as segmentações do tipo presença, que objetivam identificar toda a silhueta do usuário parado observou-se, a partir de experiências realizadas de forma empírica, que os limiares que devem ser utilizados para o espaço de cor RGB, sob pena de produzir muitos falsos positivos ou falsos negativos, variam entre 20 e 50, e para o espaço de cor RGB normalizado variam de 5 a 20. Isso porque a normalização das cores diminui as variações bruscas de intensidade entre os *pixels*. Um limiar abaixo de 20 para o espaço de cor RGB tende a ser extremamente sensível à mudanças de iluminação durante a sequência de imagens.

A Figura 2 mostra que a segmentação por diferença de quadros do tipo presença dificilmente produz um resultado ótimo, tanto para o espaço de cor RGB quanto para RGB normalizado. No entanto, esses resultados podem ser combinados, e muitos falsos positivos podem ser eliminados.

A partir deste experimento pode-se concluir que a segmentação por diferença de quadros do tipo presença é extremamente dependente dos valores de limiar e do pós-processamento efetuado a partir dos resultados da segmentação. Já o mesmo não ocorre para a diferença entre quadros do tipo movimento, que geralmente não depende de ótimos resultados para as aplicações as

quais se destina, e pode ser resolvido no espaço de cor RGB com limiares que variam entre 40 e 60.

A Figura 3 mostra o segundo tipo de segmentação obtido pela identificação da cor da pele. A figura mostra a imagem original (Figura 3.A) seguida da segmentação nos espaços de cor RGB (Figura 3.B) e RGB normalizado (Figura 3.C).

Na API Move-In, a cor da pele é segmentada a partir de uma equação paramétrica no espaço de cor RGB definida no trabalho de Gasparini e Schettini [2006]. Esta equação foi obtida de forma empírica e avalia os valores dos componentes RGB, classificando como cor da pele aqueles que se encontram dentro de um determinado intervalo.



Figura 3: Segmentação da cor da pele.

A imagem binária resultante da segmentação da cor da pele produziu muitos falsos positivos e demonstra que o algoritmo não é ótimo. Por esse motivo, esse tipo de segmentação deve ser utilizado em casos onde não há elementos no cenário, além da própria pele do usuário, que façam parte do intervalo de valores definido como pele pela equação paramétrica (como no exemplo da Figura 6).

2.2 Gerenciamento de Regiões de Interesse

O não uso de toda a imagem para realizar uma das formas de segmentação é freqüente. A utilização de regiões de interesse (ROIs) ocorre principalmente na ativação de botões virtuais e movimentação de objetos virtuais. Para tanto, a API Move-In conta a classe ROIGroup, onde um conjunto de regiões de interesse pode ser criado para uso conjunto com as técnicas de segmentação.

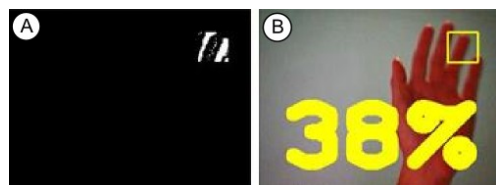


Figura 4: Demonstração de ativação de botão virtual.

A Figura 4 demonstra como uma ROI pode ser usada em conjunto com a segmentação do tipo movimento para ativação de botão virtual. Para a ROI localizada no canto superior direito da Figura 4.B, a diferença entre os dois últimos quadros é realizada Figura 4.A. Após isso, calcula-se a porcentagem de ativação, ou proporção de *pixels* brancos dentro da ROI (38% na Figura 4.B). Dessa forma, um botão pode

ser programado para ser ativado após uma sequência de quadros cuja porcentagem de ativação seja superior a 25%, por exemplo.

Se um conjunto de ROIs for utilizado em volta de um objeto virtual, conforme Figura 5, pode-se permitir que o usuário o “empurre” ao longo da tela. Isto é, movimentos detectados na ROI 0 deslocam o objeto, e, conseqüentemente, a ROIGroup, para a direita.



Figura 5: Organização de ROIs para movimentação de objeto virtual.

Movimentos detectados nas ROIs 1, 2 e 3, respectivamente, deslocam o objeto virtual para baixo, direita e esquerda. Experimentos realizados concluíram que 4 ROIs, conforme demonstrado na Figura 5, são suficientes para esta função. Complementarmente, pode-se compor movimentos caso sejam detectados duas ROIs ao mesmo tempo, gerando movimentos inclinados, por exemplo.

2.3 Extração de Características

As características que podem ser extraídas a partir de uma imagem segmentada são: **Área ativa**: proporção dos pixels brancos dentro de uma ROI; **Centróide**: ponto médio aproximado da região afetada; **Contorno**: sequência de pontos referente ao contorno dos *blobs*; **Fecho convexo**: o fecho convexo referente ao contorno; **Pontos extremos**: o fecho convexo acrescido dos defeitos de convexidade, que consiste de um ponto côncavo que maximiza a distância de uma região côncava entre dois pontos convexos.

Exemplo de uso da característica “área ativa” aparece na Figura 4 (botão virtual). O “contorno” pode ser utilizado para destacar os limites entre o usuário real e o cenário virtual (figura 1.A). O uso dos “pontos extremos” pode ser conferida na Figura 6.

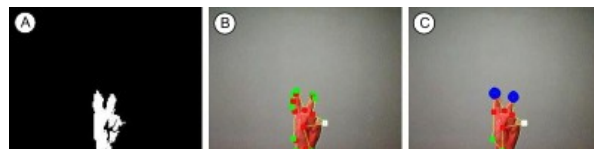


Figura 6: Análise da característica “pontos extremos”.

Na Figura 6.A, os pontos extremos foram calculados a partir da imagem segmentada com identificação da cor da pele. O pontos convexos (em verde na Figura 6.B) foram eliminados e substituídos pela média entre esses pontos quando estes se encontravam muito próximos. Então, as seguintes

heurísticas (análise de características) foram aplicadas para determinar quais dedos estavam erguidos:

1. O ponto i , sendo i sua posição na lista de pontos, deve ser convexo;
2. Os pontos $i-1$ e $i+1$ devem ser côncavos;
3. O ponto i deve se encontrar em uma posição (altura) superior à dos pontos $i-1$ e $i+1$;

O resultado aparece em azul na Figura 6.C. Com a ponta do dedo, a aplicação de pintura com o dedo da figura 1.B foi criada.

2.4 Fluxo Óptico

A utilização da classe Flow da API Move-In está ilustrada na Figura 7. Primeiramente, o conjunto de pontos a serem rastreados é determinado (Figura 7.A). Após isso, os vetores de movimentação referentes a esses pontos são calculados (Figura 7.B). Nas figuras 7.B e 13.C os vetores foram aumentados em um fator de 10X para facilitar a visualização.

A Figura 7.C mostra o vetor resultante calculado a partir dos vetores ilustrados na Figura 7.B. A soma dos vetores dá a direção do vetor resultante, e sua magnitude é calculada pela média aritmética das magnitudes dos conjuntos totais de vetores.

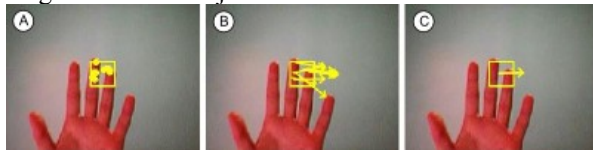


Figura 7: Pontos rastreados, vetores de movimentação e vetor resultante do fluxo óptico.

É o projetista da aplicação que define a quantidade de pontos a serem rastreados. O vetor resultante é utilizado opcionalmente, mas geralmente é indispensável, já que é ele quem determina a direção e velocidade do movimento.

O cálculo dos pontos a rastrear é efetuado automaticamente por uma função da API OpenCV. Alternativamente, pode-se definir quais pontos devem ser rastreados. O resultado é o rastreamento de um objeto real ao invés da leitura de movimento sobre um objeto virtual. Esta alternativa é recomendada somente para objetos deslocados nas duas dimensões correspondentes à largura e altura da imagem de entrada, sob pena de apresentar resultado extremamente impreciso.

As aplicações do fluxo óptico variam como forma alternativa à utilização da segmentação do tipo movimento. O fluxo óptico tem a capacidade de efetuar a movimentação de um objeto virtual (Figura 5) com a utilização de uma única ROI.

O cálculo do fluxo óptico é muito sensível a quaisquer mudanças de iluminação em uma sequência de imagens. Dessa forma, seu uso é recomendado em

conjunto com a detecção de movimento com diferença entre quadros.

3. Conclusão

Este trabalho apresentou os detalhes do projeto, implementação e funcionamento da API Move-In, que visa facilitar o desenvolvimento de aplicações interativas, onde o usuário usa o próprio corpo, identificado por uma *webcam*, durante a comunicação com a aplicação.

A API Move-In criou uma camada de abstração sobre a API OpenCV que, além de agrupar um conjunto de funcionalidades relacionadas à identificação de presença e movimento do usuário, ainda provê formas de gerenciamento dessas funcionalidades em regiões específicas definidas pelo projetista da aplicação. A API Move-In está disponível no *website* <http://code.google.com/p/move-in/> e maiores informações sobre esta pesquisa podem ser obtidas em <http://www.joinville.udesc.br/larva>.

Neste trabalho ficou evidente a relação de dependência entre a detecção de movimentos rápidos e a qualidade da *webcam* em uso somada à capacidade de processamento dos computadores. Jogos que simulam esportes como tênis de mesa ou voleibol, dificilmente serão aplicados com sucesso com o *hardware* disponível na maioria das residências da atualidade, a não ser que o projetista limite a velocidade de movimentação da bola para estimular o jogador a não utilizar movimentos muito bruscos.

Foi observado que o trabalho com técnicas de identificação de movimento é recomendado para uso doméstico quando em comparação às técnicas de identificação de presença devido ao maior grau de robustez em relação a mudanças de iluminação. Por outro lado, identificou-se que a identificação de poses ou posturas do corpo e das mãos não é possível somente com técnicas de movimento.

Referências

- GAMMA E., HELM R., JOHNSON R., VLISSIDES J., 1995. Design Patterns: Elements of Reusable Software. Addison-Wesley.
- GASPARINI, F., SCHETTINI, R., 2006. "Skin Segmentation Using Multiple Thresholding". In: *Proceedings of SPIE - The International Society for Optical Engineering*, 15-19 January 2006 San Jose.
- INTEL., 2006. *Open Source Computer Vision Library* [online]. Disponível em: <http://www.intel.com/technology/computing/opencv/> [Acessado em 13 de agosto 2008].
- SONY COMPUTER ENTERTAINMENT INC., 2003. *Sony Eye Toy* [online]. Disponível em: <http://www.eyetoy.com/index.asp> [Acessado em 13 de agosto de 2008].