Captura de Movimento Usando Algoritmo Lucas-Kanade e Rastreio de Olhos

Matheus Ferraz Petrovich Pereira
Email : matheus.petrovich@gmail.com

15 Junho 2016

1 Introdução

Captura de movimentos (motion capture, ou mocap) consiste na gravação de movimentos do mundo real no mundo virtual. O conceito de mocap teve início centenas de anos antes de Cristo, quando Aristóteles estudou e documentou os comportamentos e padrões de movimentos humanos e animais. Quase dois milênios depois, Da Vinci, Galileu, Newton e outros pensadores continuaram essa análise e documentação buscando entender e descrever a dinamica dos movimentos. [6] Com o passar do tempo e o desenvolvimento da tecnologia, os registros passaram de desenhos e fotos para filmagem usando iluminação infravermelha, marcadores reflexivos e sensores inerciais[5, 6]. O objetivo da captura de movimento é trazer para a animação uma base com o máximo de realismo rapidamente, criando uma espécie de rascunho sobre qual o animador pode construir, ou ainda com leves retoques ser aplicada diretamente no produto final. Quanto mais distante de humano o personagem, menos o observador o questiona, mas quando se aproxima de humano, o olho do observador se torna mais crítico, e assim o suporte de um sistema de captura permite que sejam transmitidos ao modelo virtual todas as nuances dos movimemtnos do rosto do ator, tornando o personagem mais críve[2]. Este artigo propõe o desenvolvimento de um aparato para captura de movimentos faciais, bem como o desenvolvimento de um software para o processamento das imagens. O dispositivo pode ser utilizado não somente para a indústria de entretenimento como filmes e video-games, mas também para o estudo e monitoramento de microexpressões faciais em diversas aplicações[1, 10].

2 Referencial Teórico

2.1 Fluxo Óptico

Fluxo óptico é uma importante ferramenta para estimar o movimento de certos pontos em uma sequência de imagens. Indicando a movimentação de padrões de intensidade na imagem, essa técnica permite obter informações sobre o arranjo de objetos na imagem e a taxa de mudança desses arranjos. [3] Para um melhor desempenho devem ser buscados na imagem pontos ótimos de rastreio. Esses são pontos que possuam fortes gradientes tanto na direção vertical quanto na direção horizontal [7]. O algoritmo de fluxo óptico utilizado neste projeto foi o de Lucas-Kanade, proposto em [8]

2.2 Algoritmo de Lucas-Kanade

O algoritmo de Lukcas-Kanade foi proposto em 1981, pode ser utilizado para identificar o movimento de um ponto ótimo de rastreio em uma pequena janela ao redor do ponto de inicial. Para isso o algoritmo requer três condições: constância de brilho, pequenos movimentos e coerência espacial (os pontos observados devem ter movimento similar aqueles em seu entorno) [7]. O algoritmo funciona baseado na premissa de que o padrão de brilho na imagem é quase constante, assim temos as equações $\frac{dI(x,y,t)}{dt} = 0$ ou $I_x u + I_y v + I_t = 0$ onde I_x, I_y e I_t são as derivadas da imagem nas respectivas dimensões e u e v são os componentes do fluxo óptico no instante t. Sendo esse um sistema subdeterminado, não pode ser resolvido. Lucas e Kanade consideraram então, que em uma região de NxN pixels o fluxo ótico é constante, tendo que $I_x iu + I_y iv + I_t i = 0$ para $i = 1, ..., NxN$. Podendo agora estimar os mínimos quadrados do fluxo óptico nessa região, dada pela equação $(uv) = (A^T A)^{-1} A^T b$ [9, 4].

3 Aparato de Captura

O aparato de cûaptura é composto por um capacete em fibra de vidro e uma câmera presa ao fim de um braço de alumínio. O capacete foi confeccionado a partir de um moldelo da cabeça do usuário, obtido em um processo de moldagem com ataduras de gesso. Sobre o modelo foram laminadas camadas de resina de poliester pigmentada reforçadas com manta de fibra de vidro. Cortes foram feitos na parte traseira do capacete para permitir flexibilidade e uma tira de velcro foi afixada para permitir um ajuste de modo que o capacete ficasse justo na cabeça do usuario. No interior do capacete foram colados pedaços de feltro para melhorar o conforto e ajudar a deixar o capacete mais fixo, para que a câmera estivesse sempre estática em relação ao rosto do usuário.

Dois parafusos foram usados para afixar o braço de alumínio ao capacete. A barra de alumínio foi medida e moldada de forma a permitir que a câmera tivesse uma visão

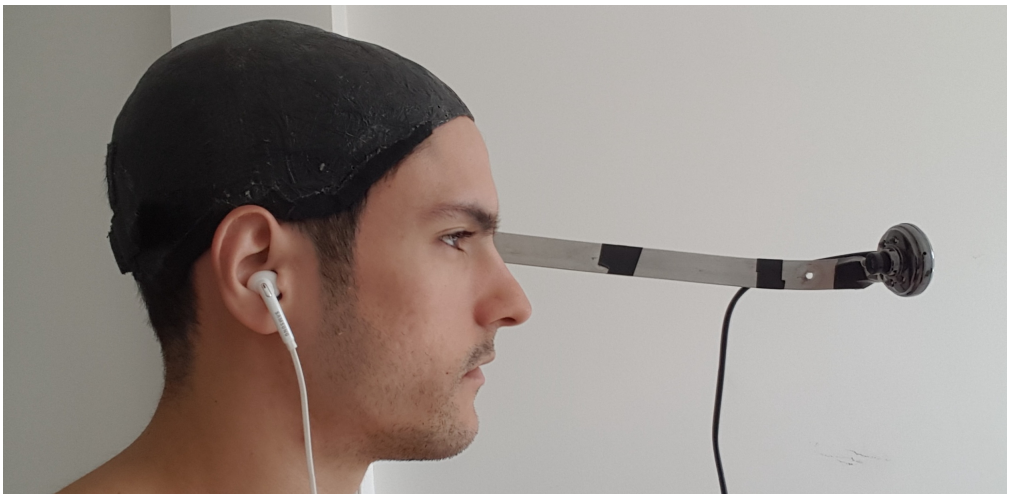

Figure 1: Aparato de captura completo

completa da face do usuário. Na ponta do braço de alumínio foi presa com massa de epoxi uma câmera USB que fará a captura da imagem para o processamento.

Figure 2: Capacete em resina de poliester reforçada com fibra de vidro

4 Implementação em Software

O programa foi implementado em c++ utilizando o QTCreator para desenvolver a interface gráfica e OpenCV para a manipulação das imagens. Ao iniciar o programa o usuário está na tela vista na imagem, aqui ele escolhe a câmera que deseja usar (0 para câmera

embutida, 1 para webcam do capacete) e clica em iniciar. Se a câmera estiver disponível a janela se adequará para comportar o tamanho da imagem e o vídeo capturado será exibido na tela. Como a webcam está posicionada de lado é necessário rotacionar a imagem, isso é atingido por uma operação de transposição e um espelhamento em seguida.

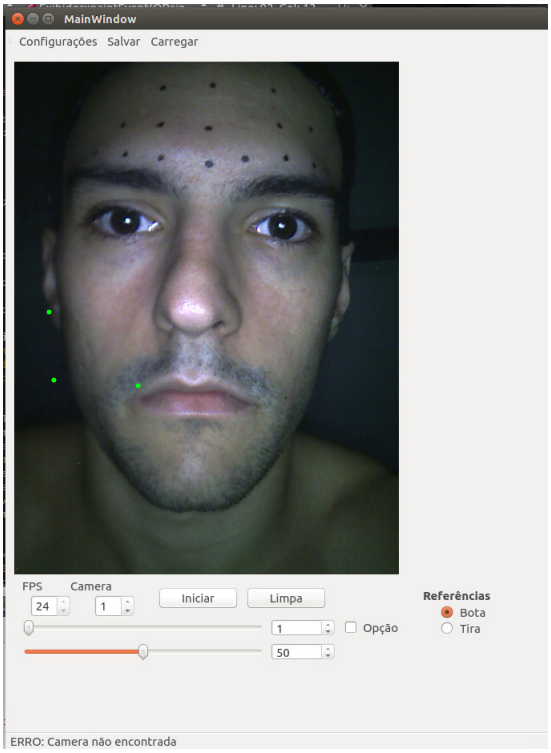

Figure 3: Tela inicial com vídeo

Embora a imagem exibida na figura 3 seja colorida, o processamento se dá apenas sobre uma das componentes de cor. A imagem é dividida em 3 planos e o plano a ser utilizado (vermelho, verde ou azul) pode ser escolhido em tempo real.

Como mencionado anteriormente, o algoritmo de Lucas-Kanade seleciona pontos ótimos para rastreio, a função para o cálculo desses pontos é executada uma vez na inicialização da captura, e em seguida em cada novo quadro. Esses pontos são então exibidos pelo programa em verde, e o rastreamento se dá pela diferença na posição do ponto no frame atual e no anterior. Para criar pontos mais fortes e que possam ser rastreados apesar das variações na iluminação da face causadas pela movimentação do ator, usualmente são colocadas marcações em sua face, dos principais pontos de movimento. Esses marcadores podem ser vistos na figura 4.

O algoritmo é rodado sobre a imagem e os pontos ótimos são identificados, pode-se observar na figura 5 como todos os marcadores têm um ponto ótimo sobre eles.

Clicando uma vez na imagem pode-se inserir pontos de referencia, que são exibidos

Figure 4: Marcações na testa do usuário

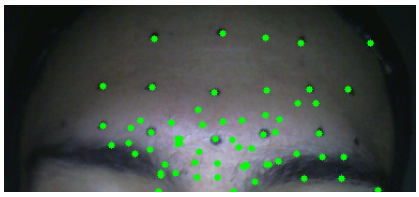

Figure 5: Pontos ótimos para rastreio

em azul. Esses pontos podem ser salvos e carregados em arquivos externos caso haja a necessidade de se trabalhar com um mesmo usuário repetidamente e estejam sendo usados muitos pontos de referência. Iteramos pela lista de pontos bons para rastreio, e o primeiro que estiver a 3 pixels ou menos da referência será seu ”pai” e o ponto azul começará a seguí-lo. O deslocamento começa a ser calculado pela diferênça entre a posição atual do pai e a no frame anterior. Esse deslocamento é a saída final do programa, esses valores podem ser exportados por meio de um socket ou da geração de um arquivo de mocap a ser importado por softwares de animação.

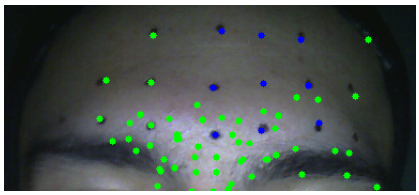

Figure 6: Referências rastreando pontos ótimos

Para iniciar o rastreio dos olhos, o usuário deve clicar duas vezes sobre a pupila de cada olho, aparecerão retângulos sobre os olhos indicando a região processada para o rastreio, como pode ser visto na figura 7.

Como o aparato mantém a câmera estática em relação ao rosto, a posição dos retângulos não precisa ser corrigida no decorrer da captura. Para rastrear o movimento das pupilas o algoritmo de Lucas-Kanade não é eficaz, uma vez que a cada piscada todas as referências são perdidas, assim uma outra abordagem é necessária. As imagens dos olhos são segmentadas com pode ser visto na figura 8. Analisando o resultado da imagem segmentada foi desenvolvido o método de ratreio aqui utilizado. O rastreio se dá em duas fazes, primeira-

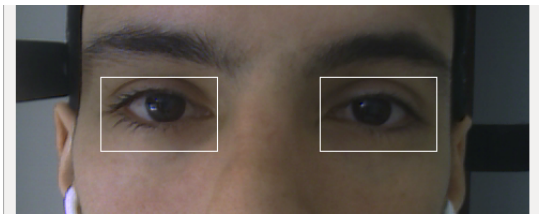

Figure 7: Regiões de rastreio dos olhos

mente a posição horizontal e então a vertical. A pupila está sempre no lado central das imagens (ultimas colunas do olho esquerdo e primeiras do direito) iteramos inicialmente a partir desses pontos, percorrendo as colunas no loop mais externo e as linhas no loop mais interno. Assim que a primeira sequência ininterrupta de dez pontos pretos for encontrada, ali começa a pupila horizontalmente, sabendo que o raio dela permanece constante em toda a captura utilizamos o raio da pupila como offset para determinar a posição horizontal do seu centro. Utilizamos o valor de dez pixels para evitar que ruídos da segmentação dessem um resultado falso-positivo. Em seguida seguimos para a identificação do centro vertical da pupila. Como agora sabemos qual o centro horizontal, iteramos nessa coluna iniciando do zero (do topo da imagem) e assim que a primeira sequência de dez pixels for encontrada, alí será o início da pupila verticalmente, novamente aplicamos um offset para localizar o centro vertical. O ponto encontrado pode ser visto na figura 8 como um círculo preto dentro de um branco.

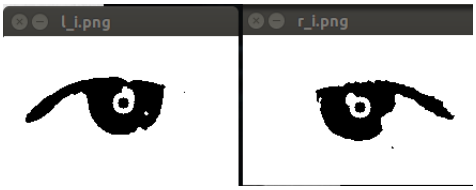

Figure 8: Regiões de rastreio dos olhos segmentadas

Caso seja necessário ajustar o tamanho da região dos olhos, pode-se clicar em configurações e fazer as alterações desejadas lá. Nessa janela também é possível alterar independentemente o threshold para limiarização dos olhos, caso haja necessidade devido a questões com a iluminação. O deslocamento do olho é dado por uma estrutura que dentre outras informações acompanha a posição x, y das pupilas e permite o cálculo da diferença.

5 Conclusão

Através de um processamento simples e com pouco poder computacional pôde-se atingir o proposto para o rastreamento dos pontos de referência e dos olhos. O objetivo

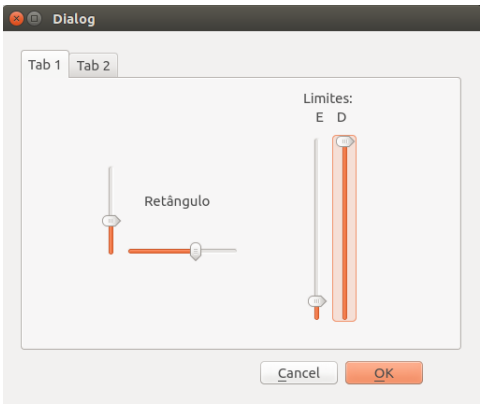

Figure 9: Janela de configuração

do trabalho não pode ser alcançado por completo dado que o resultado final não pôde ser aplicado a uma animação. Para trabalhos futuros se pretende exportar do programa um arquivo de mocap que possa ser importado por programas de animação (exemplo o .bvh pro programa de animação Blender), tornar o sistema totalmente autocontido, embarcando um microcomputador ultraportátil Raspberry PI e fazendo a gravação e o processamento online.

References

[1] Thabo Beeler Fabio Zund Derek Nowrouzezahrai Ilya Baran Olga Sorkine Hanspeter Pfister Robert W. Sumner Bernd Bickel Markus Gross Amit H. Bermano, Derek Bradley. Facial performance enhancement using dynamic shape space analysis. 2013.

[2] Karl F. MacDorman e Z. A. Dwi Pramono Chin-Chang Ho. Human emotion and the uncanny valley: A glm, mds, and isomap analysis of robot video ratings. 2008.

[3] Vantuil José de O. Neto e David M. Gomes. Comparação de métodos para localização de fluxo óptico em sequências de imagens. 2011.

[4] Thiago Vinícius D. Ferraz e Gustavo F. Rodrigues. Rastreamento labial utilizando fluxo Óptico para reconhecimento de fala em imagens de vídeo. 2015.

[5] Organic Motion Inc. What is motion capture? http://www.organicmotion.com/motion-capture/. Accessed: 2016-06-15.

[6] Xsens North America Inc. The fascination for motion capture. https://www.xsens.com/fascination-motion-capture/. Accessed: 2016-06-08.

[7] Agostinho B. Junior. Processamento digital de imagens - fluxo Óptico. http://agostinhobritojr.github.io/cursos/pdi/fluxo.pdf. Accessed: 2016-06-15.

[8] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, page 674–679, 1981.

[9] Hansjorg A. Schneebeli e Eliete M. de O. Caldeira Mário S. Filho. Cálculo do fluxo Óptico em tempo real e sua utilização na navegação de robôs móveis. 2001.

[10] Yaser Sheikh e Iain Matthews Natasha Kholgade. Content retargeting using parameter-parallel facial layers. *Eurographics Symposium on Computer Animation*, pages 1 – 9, 2011.