Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Rastreamento e Modelagem de um Objeto Rigido num Video

Anderson Mayrink da Cunha
Luiz Velho (orientador)

Technical Report TR-2004-03 Relatório Técnico

Setembro - 2004 - September

Rastreamento e Modelagem de um Objeto Rígido num Vídeo

Anderson Mayrink da Cunha Luiz Velho

Setembro de 2004

Resumo

Um modo relativamente simples para se estimar a forma e movimento de um objeto filmado em um vídeo é usar o algoritmo de Lucas Kanade para rastreamento e o de Tomasi-Kanade para modelagem (obtenção da forma e posição) de um objeto rígido. Nesse texto descrevemos a teoria e implementação desses métodos, assim como alguns aprimoramentos.

1 Introdução

O problema de estimar o movimento e forma de um objeto rígido em um vídeo tem muitas aplicações em computação gráfica e visão computacional e é alvo de muito interesse nos últimos anos.

O modo que escolhemos para abordar esse problema é composto basicamente de dois passos:

1. Rastreamento: O usuário inicialmente seleciona pontos do objeto na primeira imagem do vídeo. Esses pontos são rastreados ao longo do vídeo obtendo uma matriz de registro com a posicão desses pontos em todas as imagens do vídeo.

2. Modelagem: A partir da matriz de registro obtida no rastreamento, obtemos a forma e o movimento do objeto.

Um método conhecido para o rastreamento é o de Lucas-Kanade [1], descrito na seção 2.

O método que utilizamos para a modelagem é o de Tomasi-Kanade [2], descrito na seção 3.

O rastreamento dos pontos de um objeto rígido pelo algoritmo de Lucas Kanade não usa a importante restrição de que o objeto é rígido. Para atacar esse problema seguimos o método sugerido pela Irani [3], descrito na seção 4.

Podemos também usar métodos probabilisticos no rastreamento [4], [5], como descrito na seção 5.

Na seção 6 apresentamos o método de posto 1 sugerido por P. Aguiar e J. Moura [6] e na seção 7 descrevemos a implementação em Matlab.

2 Rastreamento pelo método de Lucas Kanade

Vamos descrever de modo intuitivo o método de Lucas Kanade [1], que é um método iterativo de primeira ordem (tipo método de Newton). Assumimos que a iluminação é constante e a superfície é Lambertiana.

Começamos analisando o caso de casamento de duas funções $1D$. Sejam duas funções $f(x)$ e $f_1(x)$, onde $f_1(x)$ é uma cópia ruidosa de $f(x)$ transladada no eixo x. Isto é: $f_1(x) = f(x-k) + \mathcal{N}(x)$, onde $\mathcal{N}$ é um ruído.

Seja x um ponto marcado em f. Desejamos fazer uma estimativa para o valor de $x_1 = x + \Delta x$ tal que $f_1(x_1)$ tenha as mesmas características locais de $f(x)$.

Uma possível estimativa do passo pelo método iterativo de Newton para o deslocamento Δx é:

$$f(x) - f_1(x) = f'(x).\Delta x$$

No caso $2D$ temos duas imagens em tons de cinza I_0 e I_1.

$$I_0, I_1 : D \rightarrow [1, \cdots, L]$$

onde $D = [1, \cdots, nl] \times [1, \cdots, nc]$ (nl e nc são respectivamente os números de linhas e de colunas da imagem) e L é o número de níveis de cinza da imagem.

Um ponto p é selecionado em I_0. Queremos rastrear esse ponto p e descobrir qual a sua posição na imagem I_1. O deslocamento bidimensional da posição do ponto p entre as imagens é chamado de fluxo ótico, denotado por F.

$\nabla I_0(p)$, o valor do gradiente no ponto $p = \begin{pmatrix} x \\ y \end{pmatrix}$ na imagem I_0, é calculado com a discretização:

$$\nabla I_0(p) = \nabla I_0 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} (I_0(x+1, y) - I_0(x-1, y))/2 \\ (I_0(x, y+1) - I_0(x, y-1))/2 \end{bmatrix}$$

Uma possível extensão do método de Newton para $2D$ é considerar a seguinte estimativa para o fluxo:

$$I_0(p) - I_1(p) = \nabla I_0(p)^T . F$$

A equação acima é indefinida pois temos uma equação e duas incógnitas (o fluxo F é um vetor de 2 linhas $\times$ 1 coluna). $\nabla I_0(p)^T$, de dimensão (1×2), é a transposta do vetor gradiente.

Multiplicando ambos os lados da equação por $\nabla I_0(p)$ e calculando os valores de itensidade da imagem e do gradiente não só no ponto p, mas em uma janela ao redor desse ponto (a janela é um filtro passa-baixa, suavizando a imagem). Com isso temos a seguinte equação:

$$Y_{2\times1} = X_{2\times2} . F_{2\times1}$$

onde F é o fluxo ótico e Y, a variação temporal, é dada por:

$$Y = \sum_{q \in Jan(p)} \left(I_0(q) - I_1(q)\right) . \nabla I_0(q)$$

onde o somatório é calculado na janela retangular $Jan(p)$ centrada no ponto p. A variação espacial X é dada por:

$$X = \sum_{q \in Jan(p)} \nabla I_0(q) . \nabla I_0(q)^T$$

A equação $Y_{2\times1} = X_{2\times2} . F_{2\times1}$ pode ser resolvida se a matriz $X_{2\times2}$ não for mal-condicionada. Se $X_{2\times2}$ é bem-condicionada temos:

$$F = X^+ . Y$$

onde X^+ é a pseudo-inversa de X.

Se desejamos rastrear N pontos temos, para cada ponto i, as equações $Y_{i(2\times1)} = X_{i(2\times2)} . F_{i(2\times1)}$. Essas equações podem ser concatenadas matricialmente na seguinte forma:

$$Y_{2.N\times1} = X_{2.N\times2.N} . F_{2.N\times1}$$

onde Y e F são respectivamente a concatenação vertical dos Y_i e F_i ($Y_{i(2\times1)}$ é a variação temporal do ponto i). X é a concatenação diagonal dos $X_{i(2\times2)}$. Isto é:

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix} \quad e \quad F = \begin{bmatrix} F_1 \\ \vdots \\ F_N \end{bmatrix} \quad e \quad X = \begin{bmatrix} X_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & X_N \end{bmatrix}$$

Em um vídeo com vários frames, a partir dos pontos iniciais no primeiro frame, devemos executar o processo acima para achar os pontos rastreados no segundo frame. Com os pontos obtidos no segundo frame, executamos o mesmo processo para achar os pontos rastreados no terceiro frame. Esse processo é repetido até obtermos os pontos rastreados em todos os frames do vídeo.

A matriz construída com a posição de todos os pontos em todos os frames do vídeo é chamada de matriz de registro e é o dado de entrada para o modelamento.

O algoritmo de Lucas Kanade, como descrito aqui, não funciona muito bem para o nosso caso de interesse, em que rastreamos vários pontos selecionados em uma face. Há a tendência de em poucos frames, os pontos se "perderem", se distanciando da posição esperada. Na seção 4 descrevemos um aprimoramento desse método feito pela Irani [3], em que ela usa no rastreamento a idéia de Tomasi-Kanade da redução do posto da matriz de registro, como explicado na próxima seção.

3 Modelagem pelo método de Tomasi Kanade

O método da Fatorização de Tomasi Kanade [2], tem como entrada a matriz de registro dos pontos rastreados ao longo do vídeo e a saída é a forma e movimento do objeto rígido rastreado. Assumimos que o modelo de câmera é ortográfico.

Seja $p_{f,n} = \begin{pmatrix} x_{f,n} \\ y_{f,n} \end{pmatrix}$ a posição do ponto n no frame f do vídeo. A matriz $\tilde{P}_{2.F \times N}$ registra a posição dos pontos $n = 1 \dots N$ nos frames $f = 1 \dots F$.

$$\tilde{P} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,N} \\ y_{1,1} & y_{1,2} & \dots & y_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{F,1} & x_{F,2} & \dots & x_{F,N} \\ y_{F,1} & y_{F,2} & \dots & y_{F,N} \end{bmatrix}$$

A matriz $\tilde{P}_{2f-1:2f,1:N}$ ocupa a posição das linhas $2f-1$ e $2f$ da matriz $\tilde{P}$ e registra as posições dos N pontos no frame f.

A forma do objeto rígido, também chamado de modelo ou shape, é representado pela matriz $S_{3 \times N}$, que registra a posição tridimensional de cada ponto:

$$S = \begin{bmatrix} s_{x,1} & s_{x,2} & \dots & s_{x,N} \\ s_{y,1} & s_{y,2} & \dots & s_{y,N} \\ s_{z,1} & s_{z,2} & \dots & s_{z,N} \end{bmatrix}$$

De acordo com o modelo de câmera ortográfico, temos que:

$$\widetilde{P}_{2f-1:2f,1:N(2 \times N)} = R_{f(2 \times 3)}.S_{(3 \times N)} + T_{(2 \times 1)}.1_{(1 \times N)}$$

onde $R_{f(2\times3)} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \end{bmatrix}$ é a matriz de rotação do frame f e contém somente as duas primeiras linhas de uma matriz ortogonal 3×3. $T_{f(2\times1)}$ é a translação no frame f e $1_{(1\times N)}$ é um vetor linha de 1's.

Chamando de r_1 e r_2 a primeira e a segunda linha de $R_{f(2\times3)}$ temos:

$$\begin{cases} r_1 r_1^T = 1 \\ r_2 r_2^T = 1 \\ r_1 r_2^T = 0 \end{cases}$$

As restrições de ortogonalidade de $R_{f(2\times3)}$ podem ser resumidas em:

$$R_f.R_f^T = I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Concatenando verticalmente as equações em cada frame, temos:

$$\widetilde{P}_{2.F \times N} = M_{2.F \times 3}.S_{3 \times N} + T_{(2.F \times 1)}.1_{(1 \times N)}$$

onde $M_{(2.F\times3)} = \begin{bmatrix} R_1 \\ \vdots \\ R_F \end{bmatrix}$ e $T_{(2.F\times1)} = \begin{bmatrix} T_1 \\ \vdots \\ T_F \end{bmatrix}$.

Denotando $M_{i(1\times3)}$ a i-ésima linha de M, isto é, $M_i = M_{i,1:3}$. As restrições de ortogonalidade da matriz M são, para todo $f = 1 \dots F$:

$$\begin{cases} M_{2f-1}.M_{2f-1}^T = 1 \\ M_{2f}.M_{2f}^T = 1 \\ M_{2f-1}.M_{2f}^T = 0 \end{cases} \quad (1)$$

Podemos estimar T_f, a translação no frame f, considerando que ela é o centróide dos N pontos nesse frame, isto é:

$$T_{f(2\times1)} = \frac{1}{N} \sum_{n=1}^{N} \begin{pmatrix} x_{f,n} \\ y_{f,n} \end{pmatrix}$$

Com isso $P_{2.F\times N} = \widetilde{P}_{2.F\times N} - T_{(2.F\times 1)}.1_{(1\times N)}$ é somente a rotação do shape. A soma dos elementos de cada linha i de P ($P_{i,1:N}$) é zero. Daí temos que:

$$P_{2.F\times N} = M_{2.F\times 3}.S_{3\times N} \tag{2}$$

Escrevendo S da forma $S = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix}$, onde $s_{x(1\times N)} = \begin{bmatrix} s_{x,1} & s_{x,2} & \dots & s_{x,N} \end{bmatrix}$ contém as coordenadas x dos N pontos do shape. $s_{y(1\times N)}$ e $s_{z(1\times N)}$ são as coordenadas y e z do shape, respectivamente. Com isso temos que:

$$P_{2f-1:2f,1:N(2\times N)} = R_f.S = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \end{bmatrix} . \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} = \begin{bmatrix} r_{1,1}.s_x + r_{1,2}.s_y + r_{1,3}.s_z \\ r_{2,1}.s_x + r_{2,2}.s_y + r_{2,3}.s_z \end{bmatrix}$$

Note que $P_{2f-1:2f,1:N}$ é combinação linear dos vetores s_x, s_y e s_z. O mesmo vale para qualquer frame f. Daí concluímos que a matriz P, que é a concatenação vertical dos $P_{2f-1:2f,1:N}$, também é combinação linear desses três vetores e portanto tem posto 3.

A matriz $P_{2.F\times N}$ em geral tem posto maior que 3, no entanto isso pode ser considerado ruído ou deficiência do modelo usado e deve ser descartado.

A decomposição por valores singulares reduzida de posto n (thin SVD) [7] de uma matriz A tem a propriedade de ser a melhor aproximação de posto n da matriz A na norma de Freobenius. No Matlab obtemos $P3$, a melhor aproximação de posto 3 da matriz P, executando os seguintes comandos:

$> [U, Sng, V] = svds(P, 3);$

$> P3 = U * Sng * V';$

Obtida a matriz $P3$, que é intuitivamente a matriz P limpa de ruído, temos de fatorar essa matriz nas matrizes do movimento (M) e da forma (S). É necessário obter S e M satisfazendo as restrições de ortogonalidade da eq. 1 de forma que $P3 = U.Sng.V^T = M.S$.

A estratégia para essa fatoração é:

1. Obter matrizes $\widetilde{M}$ e $\widetilde{S}$ que satisfaçam a equação $P3 = U.Sng.V^T = \widetilde{M}.\widetilde{S}$ (mas não necessariamente satisfaçam às restrições de ortogonalidade da matriz de movimento M, eq. 1). Tomamos os seguintes valores de $\widetilde{M}$ e $\widetilde{S}$:
$$\begin{cases} \widetilde{M} = U.\sqrt{S} \\ \widetilde{S} = \sqrt{S}.V^T \end{cases}$$

2. Para fatorar $P3 = M.S$ com M satisfazendo a equação 1 é preciso obter uma matriz inversível $G_{3\times 3}$ de forma que na equação $P3 = M.S = \widetilde{M}.\widetilde{S} = \widetilde{M}GG^{-1}\widetilde{S}$ tenhamos que $M = \widetilde{M}G$ satisfaça à equação 1. Obtida essa matriz G, a matriz do movimento (M) e a matriz do modelo (S) são facilmente calculadas:
$$\begin{cases} M = \tilde{M}G = U.\sqrt{S}G \\ S = G^{-1}\widetilde{S} = G^{-1}\sqrt{S}.V^T \end{cases} \tag{3}$$
O processo de se determinar essa matriz G é chamado de normalização e pode ser resolvido por métodos não lineares, como sugerido em [2] ou pelo método linear descrito a seguir.

3.1 Normalização

Devemos calcular G de forma que $M = \widetilde{M}.G = U.\sqrt{S}G$ satisfaça as restrições de ortogonalidade da eq. 1, Podemos escrever a eq. 1 em função das linhas de $\widetilde{M}$. Por exemplo, a restrição $M_{2f}.M_{2f}^T = 1$ pode ser escrita da forma:

$$1 = M_{2f}.M_{2f}^T = \tilde{M}_{2f}G\left(\tilde{M}_{2f}G\right)^T = \tilde{M}_{2f}^T GG^T\tilde{M}_{2f}^T = \tilde{M}_{2f}^T H\tilde{M}_{2f}^T \tag{4}$$

onde $H = GG^T$ é uma matriz simétrica:

$$H = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

Aplicando o mesmo raciocínio da equação 4, temos as restrições de ortogonalidade expressas em função das linhas de $\tilde{M}$. Para todo $i = 1 \dots F$:

$$\begin{cases} \tilde{M}_{2.i-1}.H.\tilde{M}_{2.i-1}^T = 1 \\ \tilde{M}_{2.i}.H.\tilde{M}_{2.i}^T = 1 \\ \tilde{M}_{2.i-1}.H.\tilde{M}_{2.i}^T = 0 \end{cases} \tag{5}$$

Como as equações 5 dizem respeito ao produto interno entre linhas de $\tilde{M}$, inicialmente vamos analisar um caso particular.

Seja $r_{(1\times3)} = \begin{bmatrix} x & y & z \end{bmatrix}$ e $s_{(1\times3)} = \begin{bmatrix} u & v & w \end{bmatrix}$. Vamos calcular uma matriz G tal que $(rG).(sG)^T = rGG^Ts^T = rHs^T = 1$.

$$rHs^T = \begin{bmatrix} x & y & z \end{bmatrix} . \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} . \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} . \begin{bmatrix} a.u + b.v + c.w \\ b.u + d.v + e.w \\ c.u + e.v + f.w \end{bmatrix} =$$

$$= [a.(x.u) + b.(x.v + y.u) + c.(x.w + z.u) + d.(y.v) + e.(y.w + z.v) + f.(z.w)] = 1$$

Sejam

$$F(r,s)_{1\times6} = \begin{bmatrix} x.u & x.v + y.u & x.w + z.u & y.v & y.w + z.v & z.w \end{bmatrix}$$
$$h_{6\times1} = \begin{bmatrix} a; & b; & c; & d; & e; & f \end{bmatrix}$$

Os coeficientes da matriz H podem ser calculados resolvendo o seguinte sistema linear:

$$F(r,s).h = 1$$

Aplicando o mesmo raciocínio acima para as $3.F$ equações em (5), podemos calcular os coeficientes da matriz H resolvendo o seguinte sistema linear (com a pseudo-inversa):

$$\begin{bmatrix} F(\tilde{M}_1, \tilde{M}_1) \\ F(\tilde{M}_2, \tilde{M}_2) \\ F(\tilde{M}_1, \tilde{M}_2) \\ \vdots \\ F(\tilde{M}_{2F-1}, \tilde{M}_{2F-1}) \\ F(\tilde{M}_{2F}, \tilde{M}_{2F}) \\ F(\tilde{M}_{2F-1}, \tilde{M}_{2F}) \end{bmatrix} .h = \begin{bmatrix} F(\tilde{M}_1, \tilde{M}_1) \\ F(\tilde{M}_2, \tilde{M}_2) \\ F(\tilde{M}_1, \tilde{M}_2) \\ \vdots \\ F(\tilde{M}_{2F-1}, \tilde{M}_{2F-1}) \\ F(\tilde{M}_{2F}, \tilde{M}_{2F}) \\ F(\tilde{M}_{2F-1}, \tilde{M}_{2F}) \end{bmatrix} . \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Obtidos os coeficientes de $H = GG^T$, podemos descobrir a matriz de transformação G com a decomposição em autovalores e autovetores da matriz H. No Matlab executamos os comandos:

$> [W, A] = eig(H);$

$> G = W * sqrtm(A);$

Nem sempre a matriz G calculada acima possui inversa. Nesse caso há um erro na normalização e devemos procurar métodos alternativos para o cálculo de M e S.

Se G possui inversa, obtemos M e S com a equação 3.

4 Rastreamento com Restrições de Subespaço

O método de Lucas Kanade para rastreamento não funciona bem no nosso caso de interesse em que selecionamos vários pontos no primeiro frame e desejamos rastrear o movimento desses pontos ao longo do vídeo usando uma janela pequena (com dimensões típicas de 3×3, 5×5 ou 7×7). Irani [3] sugeriu aplicar restrições de sub-espaço a cada iteração do Lucas Kanade.

As restrições de sub-espaço aplicadas são as de Tomasi-Kanade, que já foram discutidas na seção anterior. A cada iteração devemos obrigar que a matriz de registro tenha posto 3.

O algoritmo de Lucas Kanade calcula a posição dos pontos no frame f depois de calcular essas posições no frame $f-1$. No entanto, para a aplicação desse método é necessário que a cada passo do Lucas Kanade tenhamos a matriz de registro P, que contém a posição de todos os pontos em todos os frames.

Diante disso, vamos calcular a posição dos pontos no frame f a partir da posição dos pontos no frame 1. Como a iteração de Lucas Kanade será calculada entre frames distantes entre si, é essencial que tenhamos o cálculo multi-escala do fluxo ótico. Esse algoritmo pode ser escrito da seguinte forma.

4.1 Algoritmo de Rastreamento

Para cada nível da pirâmide (começando com baixa resolução):
Calcular vídeo na resolução.
Calcular a matriz X, só no primeiro frame do vídeo.
Para cada iteração:
Calcular para todo f, as matrizes Y_f da variação temporal entre o frame f e o frame 1.
Calcular para todo f, o fluxo ótico $F = X^{+}Y$
Calcular a matriz de registro P.
Tomar a melhor aproximação de posto 3 da matriz de registro $P : (svds(P, 3))$.

5 Fatorização com Incerteza

Os métodos da seção 3 e 4 usados para fatorar a matriz de registro P nas matrizes de movimento e forma ($P = M.S$) usam decomposição por valores singulares para achar a melhor aproximação de posto 3 da matriz P. Esse tipo de solução só é correta se os erros de posição da matriz P são não correlacionados e têm distribuição uniforme, o que raramente acontece num caso real.

Seguimos aqui o artigo de Brand [5], onde assumimos que na equação do fluxo ótico de Lucas Kanade, $Y = X.F$, temos que a variância do fluxo F é $\sum_F = X^{-1}$ e que a variância de Y, a variação temporal, é dada por $\sum_Y = X.\sum_F.X^T = X$.

Diante disso, é necessário uma transformação de coordenadas para que esse problema com norma elíptica (distância de Mahalanobis) se transforme em um problema com norma esférica (distância de Frobenius), onde a solução por decomposição por valores singulares pode ser aplicada. Maiores detalhes podem ser obtidos em [5].

6 Método de Posto 1

No método de Tomasi-Kanade, depois de obter a translação e a melhor aproximação de posto 3 da matriz P, devemos fazer a fatoração $P = M.S$. Para isso usávamos somente as restrições de ortogonalidade da matriz M e não foi feito nenhuma restrição sobre S.

Aguiar e Moura [6] sugerem que as duas primeiras linhas de $S_{3\times N}$, que chamaremos de $S_{0(2\times N)}$ e que contém as coordenadas x e y de S sejam as coordenadas dos pontos selecionados no primeiro frame. Com isso, somente a terceira linha do shape S (a coordenada z) é variável.

Podemos escrever a equação $P = M.S$ da forma:

$$P_{2.F\times N} = M_{2.F\times 3}.S_{3\times N} = \begin{bmatrix} M_{0(2.F\times 2)} & m_{2.F\times 1} \end{bmatrix} . \begin{bmatrix} S_{0(2\times N)} \\ z_{1\times N} \end{bmatrix} \tag{6}$$

onde $S_{0(2\times N)}$ são as coordenadas dos pontos selecionados no primeiro frame. $M_{0(2.F\times 2)}$ é a matriz formada com as duas primeiras colunas de M. $m_{2.F\times 1}$ é a matriz formada com a terceira coluna de M. A última linha de S, a coordenada z, pode ser decomposta da forma:

$$z_{1\times N} = a_{1\times N} + b_{1\times 2}.S_{0(2\times N)} \tag{7}$$

onde $a_{1\times N}.S^T_{0(N\times 2)} = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

Usando a decomposição de z da equação 7 na equação 6 obtemos:

$$P_{2.F\times N} = \begin{bmatrix} M_{0(2.F\times 2)} & m_{2.F\times 1} \end{bmatrix} . \begin{bmatrix} S_{0(2\times N)} \\ z_{1\times N} \end{bmatrix} = M_0.S_0 + m.b.S_0 + m.a$$

Multiplicando ambos os lados da equação acima por S_0^T e como $a.S_0^T = \begin{bmatrix} 0 & 0 \end{bmatrix}$, temos que:

$$PS_0^T = (M_0 + m.b).S_0.S_0^T$$

Logo temos a seguinte estimativa para $M_0 + m.b$:

$$M_0 + m.b = PS_0^T \left(S_0.S_0^T\right)^{-1} = PS_0^{+} \tag{8}$$

onde $S_0^+ = S_0^T \left(S_0.S_0^T\right)^{-1}$ é a pseudo-inversa de S_0.

Esse tipo de cálculo tem uma interpretação em termos de projeção no espaço $ran\,(S_0)$, como descrito em [6] ou [7].

Tomando a estimativa da equação 8 temos que $P1 = P - P.S_0^+ S_0 = P(I - S_0^+ S_0)$ deve ter posto 1 e devemos minimizar a equação:

$$\min ||P(I - S_0^+ S_0) - ma|| \tag{9}$$

A melhor aproximação de posto 1 com decomposição por valores singulares da equação 9 é:

$$P1 = svds(P(I - S_0^+ S_0), 1) = u_{2.F\times 1}\sigma_{1\times 1}v_{1\times N}^T$$

onde σ é o maior valor singular de $P1$.

Dessa aproximação de posto 1 e da equação 9 temos as seguintes estimativas para m e a :

$$\begin{cases} m = \alpha u \\ a = \frac{\sigma}{\alpha} v^T \end{cases} \tag{10}$$

onde $\alpha > 0$ é um fator de normalização.

De posse das estimativas das equações 8 e 10, devemos ainda obrigar que a matriz M satisfaça as restrições de ortogonalidade da eq. 1. A normalização do método é parecida com a descrita na seção 3.1. Maiores detalhes podem ser obtidos em [6].

7 Implementação

Os algoritmos descritos nas seções anteriores foram implementados em MATLAB 6.5 (release 13) e testados no Windows. A implementação foi dividida em três arquivos: (1) trm.m, que contém os parâmetros, a simples interface e as rotinas de entrada e saída. (2) tracker.m, que implementa os algoritmos descritos nas seções 2, 4 e 5. (3) modeler.m, que implementa os algoritmos das seções 3 e 6.

Para ler e gravar vídeos (em formato mov e compressão Foto JPEG) usamos as rotinas MakeQTMovie.m e ReadQTMovie.m implementadas por Malcolm Slaney [8]. Uma pequena alteração foi feita para compatibilidade com o MATLAB 6.5. As funções MakeQTMovie.m e ReadQTMovie.m modificadas estão incluídas no arquivo.

Para executar o programa digitamos *trm* na linha de comando do MATLAB. Será mostrado a primeira imagem do vídeo de entrada. Vamos selecionar pontos do objeto rígido na imagem com o botão esquerdo do mouse. Para terminar a seleção dos pontos basta clicar o botão direito do mouse. Com isso, serão chamadas as rotinas tracker.m e modeler.m e os resultados serão apresentados na tela e gravados, dependendo dos parâmetros.

Todos os parâmetros estão definidos nas primeiras linhas do arquivo trm.m. O nome do arquivo de entrada é definido pelo parâmetro VIDEOIN. O vídeo de entrada deve ser em tons de cinza e de formato mov com compressão Foto JPEG. O tipo de técnica para modelagem é dado por MODEL. O método de rastreamento é dado por TRACK. Os parâmetros do rastreamento são: (1) PYRAMID, o número de escalas para o cálculo do rastreamento. Se PYRAMID=0, o número de escalas é calculado automaticamente de acordo com o tamanho da imagem. Como no método de Newton, a iteração de Lucas Kanade deve ser repetida até que tenhamos convergência, mas isso nem sempre acontece. Por isso, na implementação optamos por repetir o processo um número de vezes definido pelo usuário. (2) NITER define o número de iterações de rastreamento na escala 1. (3) NITERC define o número de iterações de rastreamento para escalas maiores que 1. (4) WINDOWSIZE é o tamanho da janela.

Depois do cálculo do modelo, podemos obter pontos com valores grandes da coordenada z. No nosso caso de interesse, em que rastreamos uma face, isso indica algum erro no rastreamento desses pontos. Para remover pontos do modelo com coordenada z maior que REMOP vezes o desvio padrão da coordenada z do shape, usamos o parâmetro REMOP. Se REMOP>0 removemos esses pontos do modelo assim como da matriz de registro e depois modelamos novamente os pontos da nova matriz de registro sem as colunas referentes aos pontos excluídos. Se REMOP=0 não removemos nenhum ponto.

A cada execução do programa podem ser criados os arquivos: (1) o vídeo de saída, (2) a matriz de registro com parâmetros do vídeo e (3) a matriz de movimento com as rotações em cada frame. O nome desses arquivos é definido, respectivamente, por: VIDEOOUT, PARAMFILE e ROTFILE. Se gravamos ou não esses arquivos é definido, respectivamente, por: SAVVIDOUT, SAVPARFILE e SAVROTFILE.

Reproduzimos aqui a parte do arquivo trm.m que contém os parâmetros:

```
% Basic Parameters
VIDEOIN = '74.mov';       % input video name
MODEL= 1;       % =0, Do not model.
                % =1, Rank 1 method.
                % =2, Tomasi-Kanade.
TRACK= 5;       % =0, Do not track.
                % =1, original Lucas Kanade tracking method
                % =2, Irani method without subspaces constraints
                % =3, Irani method with Tomasi-Kanade constraints
                % =4, Irani method with Tomasi-Kanade constraints and warp
                % =5, Irani method with Rank 1 constraints
% tracker parameters
PYRAMID= 0;       % int >= 0. Analysis with PYRAMID scales:
                  % =0, automatic. =1, only original video.
NITER= 3;         % int >=1. Number of iterations to calculate track with scale=1.
NITERC= 1;        % int >=1. Number of iterations to calculate track with scale>1.
WINDOW_SIZE= 5;       % odd number >=3. square window size. usual options: 3, 5, 7.
INTERPOL=0;       % logical. use interpolation to calc movie sub-pixels values.
                  % active only in original size.
% Remove outliers, input and output file names, file save options and other parameters.
REMOP=0;       % real=>0. Remove outliers. Remove shape points with z greater
               % than REMOP*std. if REMOP==0 then we do not remove outliers.
VIDEOOUT = 'track.mov';       % output video name
PARAMFILE= 'point.tra';       % file with parameters, initial points and tracked points.
ROTFILE = 'rotfile.rot';      % text file with rotations.
SAVVIDOUT= 2;       % =0, do not view and do not save output video.
                    % =1, only view movie.
                    % =2, view and save output video.
SAVPARFILE=2       % =0, do not save the file PARAMFILE.
                   % =1, only save parameters and the initial points.
                   % =2, save par, init pt. and registered matrix (tracked points).
SAVROTFILE=1;      % logical. save the file ROTFILE.
VERBOSE=1;         % logical. display bad points and info.
MVS= 1;            % real >0. multiply movie size.
```

Referências

[1] B. D. Lucas and T. Kanade, *"An iterative Image Registration Technique with an Application to Stereo Vision."* In IUW, pages 121-130, 1981.

[2] C. Tomasi and T. Kanade, *"Shape and motion from image streams under orthography: A factorization method."* ICVJ, 9:137-154, 1992.

[3] M. Irani, *"Multi-frame optical flow estimation using subspace constraints."* In ICCV, 1999.

[4] M. Irani and P. Anandan, *"Factorization with uncertainly."* In ECCV, 2000.

[5] M. Brand, *"Morphable 3D models from video."* 2001.

[6] P. Aguiar and J. Moura. *"Rank 1 Weighted factorization for 3D structure recovery: Algorithms and performance analysis."* IEEE-TPAMI, v.25, n.9, sept 2003.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[8] Malcolm Slaney. MakeQTMovie.m e ReadQTMovie.m. http://www.slaney.org/malcolm/pubs.html