Dissertação para obtenção do grau de Mestre em Matemática pelo
INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

RECONSTRUÇÃO DE MOVIMENTO USANDO ANÁLISE DE MOMENTOS

por
LOURENA KÁRIN DE MEDEIROS ROCHA

Orientador: Paulo Cezar Pinto Carvalho
Co-orientador: Luiz Velho

Rio de Janeiro, 07 de Outubro de 2004

Agradecimentos

À Deus, por ter posto em meu caminho tantas pessoas maravilhosas entre família, namorado, amigos, colegas de trabalho e orientadores, pois sem qualquer um deles não teria conseguido chegar até aqui.

Resumo

O objetivo desta dissertação é apresentar um método para reconstrução de movimento. Movimentos contêm muita informação sobre as relações espaço-temporais entre objetos em imagens, sendo sua reconstrução de grande importância em várias áreas de pesquisa.

Nossa abordagem é baseada em momentos de imagem e utiliza a estrutura de dados de árvores de partição binárias (*bsp*). Construímos *bsp's* que armazenam elipses usadas para aproximar a forma do objeto em cada quadro. A estrutura hierárquica, bem como as elipses, são computadas a partir de propriedades invariantes de momentos. Em seguida, essas estruturas elipsoidais são usadas para acompanhar o objeto quadro a quadro. O movimento é representado pelas transformações geométricas entre os conjuntos de elipses e é armazenado também na estrutura hierárquica. A reconstrução do movimento é feita interpolando essas transformações.

Propomos também uma aplicação para o método desenvolvido. A técnica é utilizada para produzir formas intermediárias de objetos em uma seqüência de imagens.

Palavras-Chave: Reconstrução de movimento, estrutura elipsoidal, invariantes de momentos, coerência hierárquica, combinação linear de transformações.

Abstract

The goal of this dissertation is to present a method for motion reconstruction. Motion carries a lot of information about spatio-temporal relationships between image objects. Its reconstruction plays a very important role in many research areas.

The approach is based on image moments and the bsptree data structure. In each frame we construct a bsptree to store a set of ellipses that approximates the object's shape. The hierarchical structure, as well as the set of ellipses, are computed using invariant properties of image moments. These ellipsoidal structures are matched, which allows to track the object. Motion is represented by geometric transformations between the sets of ellipses and stored in the hierarchical structure. Motion reconstruction is done by interpolating these transformations.

Following, we propose an application for our method. The approach is used to produce intermediate object's shape in an image sequence.

Keywords: Motion reconstruction, ellipsoidal structure, moment invariants, hierarchical coherence, linear combination of transformations.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

De forma simplificada, movimento é o fenômeno que envolve uma mudança na posição ou localização de algo. No caso de vídeo, o movimento contém informação sobre a relação espaço-temporal entre os objetos nas imagens. As imagens em uma seqüência de vídeo nada mais são do que uma representação discreta da projeção do movimento contínuo feito pelos objetos. A reconstrução de movimento, nesse contexto, significa a aproximação do movimento original contido no vídeo.

A reconstrução de movimento é de grande importância em muitas áreas de pesquisa tais como processamento e compressão de vídeo e animação, entre outras. Trabalhos recentes sobre reconstrução de movimento de um objeto a partir de uma seqüencia de imagens lidam com movimento humano e movimento de objetos 3D em geral [5, 21, 24].

Neste trabalho, propomos um método de reconstrução do movimento projetado no plano da imagem. O nosso objetivo é descobrir o movimento que aconteceu entre dois quadros da seqüência, interpolá-lo e assim obter uma aproximação para o movimento original. Assumimos que as imagens da seqüência já passaram por um processo de segmentação contendo apenas a forma de um único objeto, e que são binárias. A forma não pode mudar de topologia durante a seqüência, e também não podem ocorrer oclusões. O movimento da forma deve ser suave para que o método funcione apropriadamente e garanta os melhores resultados.

Como as imagens contêm apenas uma forma, a primeira providência a ser tomada é descobrir uma maneira de representá-la em cada quadro. Usamos *momentos de imagens* para realizar essa tarefa. Construímos uma árvore hierárquica *bsp* a partir das informações de momentos, onde cada nó armazena

uma *elipse de inércia*, associada à porção do objeto representada nessa partição. Essa elipse, também fornecida pelos momentos, tem a propriedade de ser a elipse que melhor se ajusta à forma do objeto [19, 14]. Desse modo, representamos a forma em cada quadro através de um conjunto de elipses.

Um passo fundamental na análise de movimento de um objeto é a tarefa de acompanhá-lo através da seqüência. Tal habilidade é conhecida como *tracking*, ou *acompanhamento*. Nos últimos anos, muitos trabalhos foram desenvolvidos com esse intuito. Os mais comuns são baseados em *features* [29], fluxo ótico [4, 3], contornos ativos [13, 28] e modelos de formas [22, 18]. No entanto, esses métodos possuem inconvenientes. Enquanto uns precisam de inicialização manual, outros não lidam com objetos não-rígidos e alguns são muito caros.

Na técnica proposta, o *tracking* se dá com a correspondência das estruturas hierárquicas (*bsp's*), funcionando bem para objetos rígidos e não-rígidos, sem precisar de inicialização manual ou necessitar de grande acumulo de memória.

Com essa correspondência estabelecida, o movimento é representado pelas transformações geométricas entre os conjuntos de elipses de quadros consecutivos. Propomos a interpolação dessas transformações, para obter uma aproximação do movimento original que ocorreu entre duas imagens. Usamos a teoria de combinação linear de transformações descrita por Alexa [1], para realizar tais interpolações.

O método proposto pode ter aplicações em várias áreas de Computação Gráfica, como por exemplo:

Visão Computacional

- Em aplicações onde a forma e algumas de suas propriedades – por exemplo, deformação – forem importantes para simular visão pré-atentiva (busca visual por características básicas, como detecção de regularidades e similaridades).

Processamento de Imagem

- Filtragem temporal em processamento de vídeos;
- Remoção de redundância temporal em compressão de vídeos: em uma situação ideal, somente a primeira imagem e o movimento subseqüente precisariam ser transmitidos.

Animação

- Suavização de animação, interpolando os quadros principais (*keyframes*) de uma seqüência de imagens.
- Transposição do movimento de um objeto para outro.

Deve-se notar que em algumas dessas aplicações, como a filtragem, compressão e até a transposição, a reconstrução do movimento pode ser feita sem a interpolação das transformações.

Neste trabalho, aplicamos o algoritmo na reconstrução de formas de objeto entre imagens consecutivas.

Esta dissertação está organizada em seis capítulos, incluindo essa introdução:

- *Capítulo 2:* Trata da representação de formas usando momentos de imagem. Os momentos são introduzidos, destacando-se as propriedades de nosso interesse, como a elipse de inércia, por exemplo.
- *Capítulo 3:* Expõe resumidamente a teoria de combinação linear de transformações desenvolvida por Alexa [1], que nos permite interpolar as transformações.
- *Capítulo 4:* Descreve o método proposto, destacando as etapas principais: construção da estrutura hierárquica, *tracking* e interpolação das transformações.
- *Capítulo 5:* Discute a aplicação que implementamos para o algoritmo: a reconstrução de formas intermediárias em uma seqüência de imagens.
- *Capítulo 6:* Conclui a dissertação e dá direções de trabalhos futuros.

Capítulo 2

Representação de Formas Usando Momentos de Imagens

A representação e a descrição de formas são procedimentos importantes em muitas áreas de visão computacional e reconhecimento de padrões. Correspondência de contornos para reconstrução 3D de imagens médicas, reconhecimento de caracteres e muitas outras tarefas visuais podem ser feitas usando reconhecimento de formas. Em geral, há duas abordagens para representação de formas: por contornos e por regiões. Aproximação poligonal, *chain code*, primitivos geométricos, curvas paramétricas, descritores de Fourier e a transformada de Hough são descritores de formas representadas por contornos. Na representação por regiões, decomposição morfológica e momentos são alguns exemplos de descritores. Esse último tem como interessante característica poder ser usado para descrever formas mesmo se a região é representada por sua fronteira.

Em nosso trabalho optamos por usar momentos para descrever formas por serem de fácil implementação e pelo seu poder de descrição. A Seção 2.1 introduz momentos no contexto de análise de imagens e destaca especialmente os momentos geométricos, por terem sido os utilizados em nosso método. Na Seção 2.2 são descritas algumas definições e propriedades deste tipo em particular e principal destas propriedades, a elipse de inércia, é descrita na Seção 2.3.

2.1 Momentos

Momentos são ferramentas estatísticas [20] amplamente utilizadas em análise de imagens, tendo sido introduzidas nesse contexto por Hu em [11] no início dos anos 60.

Sua importância nessa área se deve ao fato de que um conjunto de momentos calculados a partir de uma imagem digital geralmente representa características globais da forma da imagem além de fornecer informação sobre diferentes tipos de características geométricas da imagem.

Naquele trabalho, Hu usou momentos geométricos para gerar um conjunto de invariantes que foram usados para reconhecimento automático de caracteres. Desde então, diversas aplicações de momentos foram desenvolvidas e hoje podemos encontrá-los na literatura sendo usados em identificação de aeronaves [27], reconhecimento de caracteres [15, 8], como descritores de formas [17, 23, 6] e em reconstrução e registro de imagens [16]. Algoritmos preocupados com o desempenho do cálculo de momentos também têm sido descritos [12, 7, 9].

Além dos momentos geométricos, que são os mais tradicionais, existem momentos de Legendre, momentos de Zernike, momentos rotacionais e momentos complexos. Estudos sobre as propriedades desses momentos, incluindo questões fundamentais como seu uso em representação de imagens, sensibilidade a ruído, redundância de informação e análise de erro, podem ser encontrados em [33, 16, 19].

As propriedades de momentos geométricos de imagem têm analogias em estatística e mecânica. Em estatística, os momentos de ordem 0, 1 e 2 de uma função densidade de probabilidade representam a probabilidade total, a esperança e a variância, respectivamente. Em mecânica, esses momentos de uma distribuição espacial de massa fornecem a massa total, a posição do centro de massa, e os valores de inércia, respectivamente. Considerando uma imagem como uma distribuição de intensidade bi-dimensional, os momentos geométricos dos valores dos *pixels* com respeito às suas localizações espaciais na imagem podem fornecer informações similares, como a área total da imagem, as coordenadas do centróide da imagem, e a orientação. Essa características podem ser usadas para construir vetores que são invariantes com respeito a translações, rotações e escalas da imagem. Enquanto momentos de ordens 0 a 3 são usados para representar características da imagem num nível mais grosseiro, momentos de ordem mais altas contém mais detalhes sobre a imagem e são freqüentemente mais sensíveis a ruídos.

2.2 Definições e Propriedades

Seja $f : \Re^2 \rightarrow C$ uma função contínua com valores em um espaço de cor $C \subset \Re^n$. Tal função é dita uma *imagem bi-dimensional*, ou simplesmente, *imagem*[1].

Os *momentos geométricos* de ordem $p + q$ de f são definidos como [20, 19]:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \tag{2.1}$$

com p, q = 0, 1, 2 ...

Para $f : U \subset \Re^2 \rightarrow C$ contínua por partes e limitada em U, valem as propriedades abaixo (ver [20]):

Existência: os momentos de todas as ordens existem e são finitos.

Unicidade: a seqüência de momentos m_{pq} é unicamente determinada por f, e vice-versa.

Neste trabalho, o espaço de cor C está quantizado com 1 bit, e f é dita uma *imagem binária*. Assim, como assumimos a existência de apenas um objeto na imagem, f é a função característica desse objeto. As propriedades citadas acima de *existência* e *unicidade*, asseguram neste caso, que a forma do objeto é completamente determinada pelos seus momentos.

Uma imagem $f : U \subset \Re^2 \rightarrow C$ é dita *imagem digital* se U é discretizado, isto é, U é um reticulado do plano, por exemplo $U = \{0, 1, \ldots, m\} \times \{0, 1, \ldots, n\}$. Para uma imagem digital, os momentos geométricos (m_{pq}) são definidos como:

$$m_{pq} = \sum_{x,y \in U} x^p y^q f(x, y) \tag{2.2}$$

onde o somatório se estende sobre todos os elementos em U.

Se a função f é definida como $f : U \subset \Re^2 \rightarrow \{0, 1\}$, com U discretizado, então f é chamada de *imagem digital binária*. De agora em diante falaremos em *imagem de silhueta I*, onde $I = \{(x, y); f(x, y) = 1\}$, para nos referir à imagem digital binária. Para tal imagem I as funções de momento (m_{pq}) são dadas por:

$$m_{pq} = \sum_{I} x^p y^q \tag{2.3}$$

[1] As afirmações feitas neste capítulo são válidas para imagens em geral $f : \Re^n \rightarrow C$.

onde o somatório se estende sobre todos os elementos em I, i.e., todos os $pixels$ "pretos" da imagem.

Por definição, o momento de ordem zero (m_{00}) representa a intensidade total de uma imagem. Para uma imagem de silhueta, esse termo fornece a área da região da imagem e, aqui especificamente, a área da forma do objeto.

As funções de primeira ordem m_{10} e m_{01} fornecem os momentos de f sobre os eixos y e x da imagem, respectivamente. O *centróide* (x_c, y_c) de uma imagem é dado por:

$$x_c = \frac{m_{10}}{m_{00}}; \qquad y_c = \frac{m_{01}}{m_{00}}. \tag{2.4}$$

Para um imagem de silhueta, o ponto (x_c, y_c) representa o centro geométrico da região da imagem e, no nosso caso, do objeto. Muitas vezes é conveniente calcular os momentos com a origem do sistema de referência transladado para o centróide da imagem. Essa transformação torna o cálculo dos momentos independente da posição do sistema de referência da imagem. Os momentos calculados com respeito ao centróide de uma imagem são chamados de *momentos centrais*, e para imagens de silhueta são dados por [2]:

$$\mu_{pq} = \sum_{x,y \in U} (x - x_c)^p (y - y_c)^q, \quad p, q = 0, 1, 2, 3... \tag{2.5}$$

Da definição de momentos centrais, temos que

$$\mu_{00} = m_{00}; \qquad \mu_{10} = \mu_{01} = 0. \tag{2.6}$$

Os momentos de centrais de segunda ordem, μ_{20}, μ_{02} e μ_{11} são conhecidos como os *momentos de inércia* da imagem.

2.3 Elipse de Inércia

Estatisticamente, os momentos centrais de segunda ordem são uma medida da variância da distribuição de intensidade da imagem com relação a origem do sistema de referência. Os momentos centrais μ_{20}, μ_{02} são as variâncias com relação a média (centróide) e a covariância é dada por μ_{11}.

[2] Observe que apesar de estarmos discutindo objetos binários, é possível generalizar todas as relações seguintes e resultados para objetos em tons de cinza.

Eles também podem ser pensados como os *momentos de inércia* da imagem com relação à um conjunto de eixos de referência paralelos aos eixos de coordenadas da imagem, e passando através do centróide. Os *eixos principais de inércia* da imagem são definidos como o conjunto de duas linhas que se cruzam ortogonalmente no centróide da imagem e representam as direções com covariância $zero$. Os momentos de inércia (μ_{20}, μ_{02}) da imagem com relação a esse sistema de referência são então chamados os *momentos principais de inércia* da imagem.

Se μ_{20}, μ_{02}, μ_{11} são os momentos centrais de segunda ordem de uma imagem em seu sistema de referência e Σ é a matriz de covariância da distribuição de intensidade da imagem f, então Σ é dada por

$$\Sigma = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}. \tag{2.7}$$

Seus autovalores I_1 e I_2 são

$$\begin{aligned} I_1 &= \frac{(\mu_{20}+\mu_{02}) + [(\mu_{20}-\mu_{02})^2 + 4\mu_{11}^2]^{\frac{1}{2}}}{2}, \\ I_2 &= \frac{(\mu_{20}+\mu_{02}) - [(\mu_{20}-\mu_{02})^2 + 4\mu_{11}^2]^{\frac{1}{2}}}{2}, \end{aligned} \tag{2.8}$$

e correspondem aos valores dos momentos principais de inércia da imagem [19].

Note que se $\mu_{11} = 0$, então $I_1 = \mu_{20}$ e $I_2 = \mu_{02}$. O ângulo θ que o maior eixo principal de inércia faz com o eixo x é dado pela seguinte equação:

$$\mu_{11}\tan^2\theta + (\mu_{20}-\mu_{02})\tan\theta - \mu_{11} = 0. \tag{2.9}$$

Assim,

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right). \tag{2.10}$$

As equações 2.8 e 2.10 podem ser usadas para definir uma *elipse de inércia*. Ela é interpretada como a elipse que melhor se ajusta à imagem [14] (Fig. 2.1). No nosso caso, é a melhor elipse que se ajusta à forma do objeto. Ela está centrada em (x_c, y_c); seus eixos são as linhas passando através do centróide para os quais os momentos de segunda ordem sobre tais linhas são máximo e mínimo, respectivamente, e suas direções correspondem aos autovetores da matriz Σ de covariância do objeto.

Os comprimentos w e l do semi-eixo maior e semi-eixo menor da elipse de inércia são definidos pelos autovalores associados aos autovetores dessa matriz:

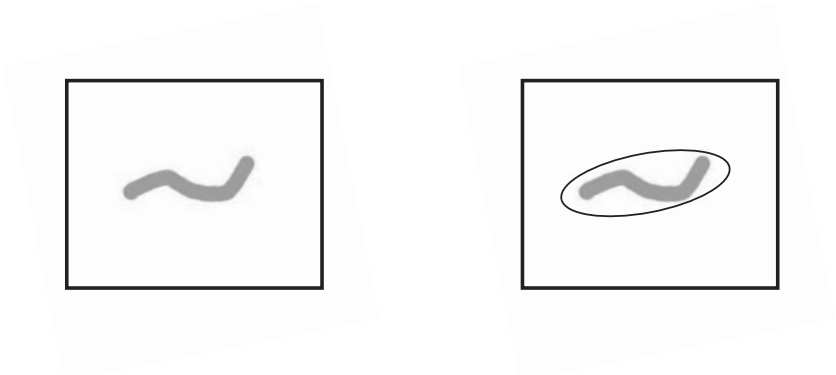
$$w = 2\sqrt{\frac{I_1}{\mu_{00}}}; \qquad l = 2\sqrt{\frac{I_2}{\mu_{00}}}. \tag{2.11}$$

A razão entre os eixos descreve a noção de alongamento do objeto. É possível atribuir um valor k de intensidade aos pontos da elipse (e $zero$ aos de fora), de modo a preservar o valor do momento de ordem zero:

$$k = \frac{\mu_{00}}{\pi ab}. \tag{2.12}$$

Portanto, uma elipse de inércia pode representar as características fundamentais da forma, e também fornecer a posição bi-dimensional e orientação. Os parâmetros w, l e θ obtidos como em 2.10 e 2.11 são chamados de *descritores elípticos* da imagem [19].

Em [14] é feito um estudo e experimentos sobre o conceito de elipse que melhor se ajusta ao objeto e em [2] é dada uma aplicação deste conceito a objetos tridimensionais.



(a) (b)

Figura 2.1: Aproximação baseada em momentos de imagens: (a) imagem fonte, (b) elipse de inércia

Capítulo 3

Combinação Linear de Transformações

Neste capítulo estudaremos a ferramenta matemática que será usada para reconstruir o movimento da forma de um objeto em uma seqüência de imagens: *a interpolação de transformações geométricas*.

Faremos uma breve exposição da teoria apresentada por Alexa em [1], onde podem ser encontrados mais detalhes.

Fazendo uso de argumentos geométricos simples, apresentamos uma definição de *múltiplo escalar* e *adição comutativa* de transformações baseada em sua representação de matrizes, se estas não possuírem auto-valores reais negativos. Juntas, essas operações permitem a combinação linear de transformações, o que possibilita usar transformações arbitrárias numa estrutura similar a uma base de espaço vetorial.

Na Seção 3.1 introduziremos o problema de interpolação de transformações usando motivação geométrica. Na Seção 3.2 definiremos a multiplicação de transformações por escalar, e a adição comutativa será definida na Seção 3.3. A Seção 3.4 mostra a relação das definições com a álgebra de matrizes, com o intuito de torná-las computáveis.

3.1 Introdução e Motivação

Quando falamos sobre “manipular” (ou “trabalhar com”) imagens, estamos nos referindo a dois conjuntos diferentes de processos que interferem de alguma maneira nas imagens. Um conjunto lida com a mudança na aparência de objetos

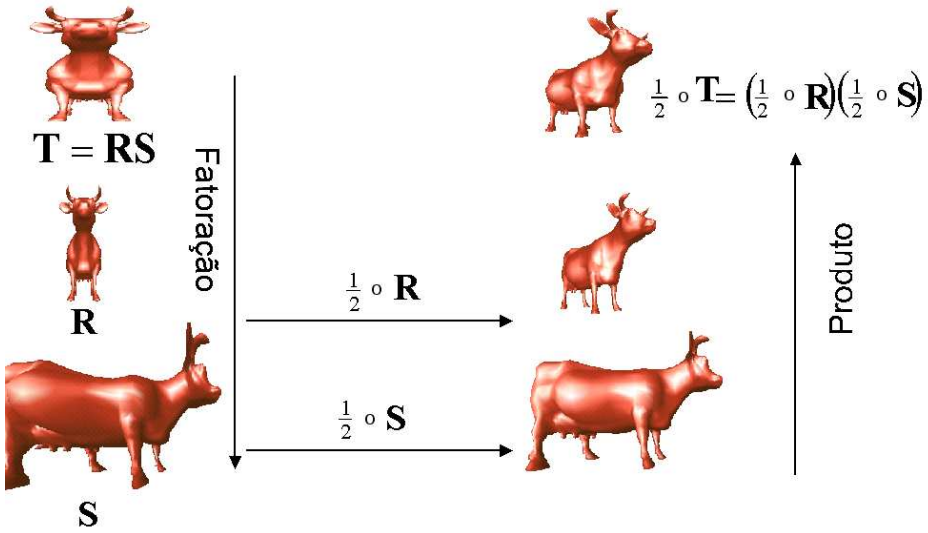


Figura 3.1: Método de Shoemake e Duff para fatoração de transformações.

através da modificação de sua cor, textura, ou "comportamento" (como por exemplo, sua transparência). O outro conjunto é composto pelos processos geométricos. Aplicados a um objeto particular, esses processos resultam, por exemplo, no movimento do objeto na tela, ou no aumento ou diminuição do seu tamanho.

Podemos realizar as transformações de objetos de dois modos: movendo o objeto ou movendo o resto do mundo. Se deixamos o sistema de coordenadas fixo e manipulamos as coordenadas do objeto, estamos "movendo o objeto". Se deixamos o objeto fixo e manipulamos a origem do sistema de coordenadas, estamos "movendo o resto do mundo". Na maioria dos sistemas de computação gráfica, as coordenadas do objeto são manipuladas diretamente, usando *transformações geométricas* [1].

Transformações são tipicamente representadas como matrizes reais quadradas e aplicadas multiplicando a matriz por um vetor de coordenadas. Coordenadas homogêneas ajudam a representar transformações aditivas (translações) e transformações multiplicativas (rotação, escala, cisalhamento)

[1] Ao longo do trabalho muitas vezes falaremos apenas em *transformações* para nos referir às transformações geométricas.

como multiplicações de matrizes. Essa representação é bastante vantajosa quando muitas transformações precisam ser compostas: como o produto de matrizes é associativo, as matrizes das transformações são multiplicadas e a concatenação de transformações é representada por uma única matriz.

Para a representação de movimento muitas vezes é necessário interpolar de uma transformação dada para outra. Um dos primeiros métodos para produzir essas interpolações foi introduzido por Shoemake [30, 31] e por Shoemake e Duff [32]. A idéia deles é escrever a matriz de uma transformação T como produto de uma matriz de rotação R e outra de *stretch* S ($T = RS$), onde R é representada usando quatérnios (Fig. 3.1). Como os quatérnios e matrizes de *stretch* podem ser interpolados, obtém-se a interpolação de T. O principal problema desse método é que ele não é comutativo, visto que essa não é uma propriedade do produto de matrizes (Fig. 3.2). Essa é uma restrição forte, pois espera-se que a "metade" de uma transformação T aplicada duas vezes resulte em T (Fig. 3.3).

Esta é a principal motivação para as definições de múltiplo escalar e adição (comutativa) de transformações.

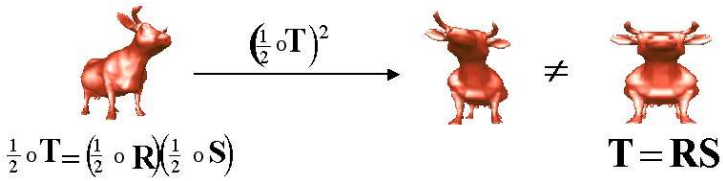


Figura 3.2: Problema do método de fatoração: falta de comutatividade.

3.2 Múltiplo Escalar de Transformações

Dada uma transformação T e um escalar s, queremos definir o múltiplo escalar $s \odot T$. Intuitivamente, espera-se que a operação $\odot$ tenha a seguinte propriedade: no caso particular $s = \frac{1}{2}$, isto é, "metade" de T, queremos que a transformação resultante aplicada duas vezes leve à transformação original T, i.e.,

$$\left(\frac{1}{2} \odot T\right) \circ \left(\frac{1}{2} \odot T\right) = T. \tag{3.1}$$

O mesmo é esperado para um terço de uma transformação, um quarto, e assim por diante.

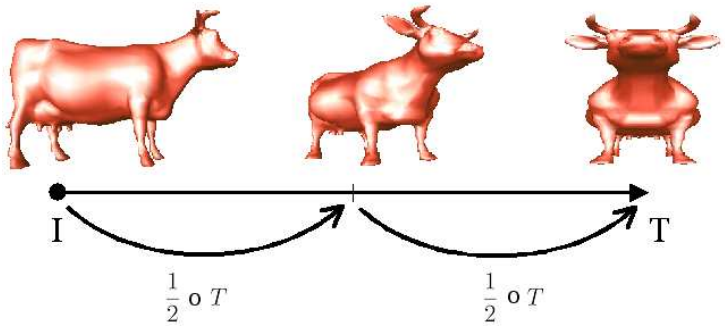


Figura 3.3: Intuitivamente, espera-se que a "metade" de uma transformação T aplicada duas vezes resulte em T.

Analisando essa propriedade em transformações como translação (Fig. 3.4), rotação (Fig. 3.5) e escala (Fig. 3.6), temos que para valores inteiros positivos de s, o candidato para $s \odot T$ é T^s; e o mesmo vale para a representação matricial da transformação T.

Definição:

Dados $s \in \Re$ e uma transformação T cuja matriz M não possui auto-valores reais não-negativos, definimos

$$s \odot T = T^s. \tag{3.2}$$

Isto significa que $s \odot T$ é a transformação representada pela matriz M^s.

A restrição a matrizes com auto-valores reais não negativos é para garantir que T^s seja real para todo s real (ver Alexa [1]).

3.3 Adição de Transformações

Estamos interessados nesta seção em definir uma operação comutativa $\oplus$, que aja sobre a composição de diferentes transformações. Ela será chamada de *adição (comutativa) de transformações*.

Dadas duas matrizes quadradas reais A e B de mesma dimensão, sabemos que geralmente AB e BA são diferentes (Fig 3.7); contudo, elas são iguais se A e B comutam e nesse caso, o produto usual de matrizes é exatamente a operação que queremos. Então, se modificarmos esse produto de forma que seja sempre

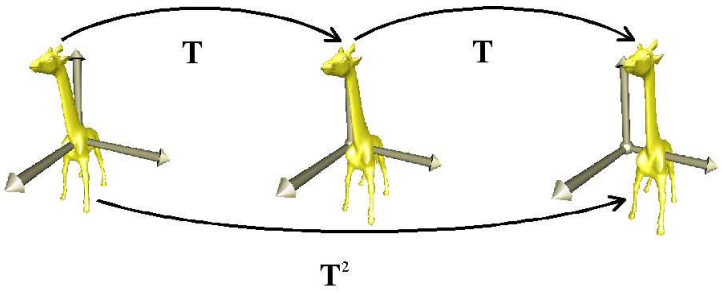


Figura 3.4: Translação

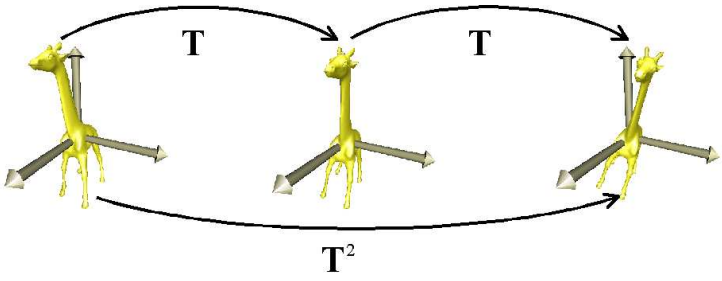


Figura 3.5: Rotação

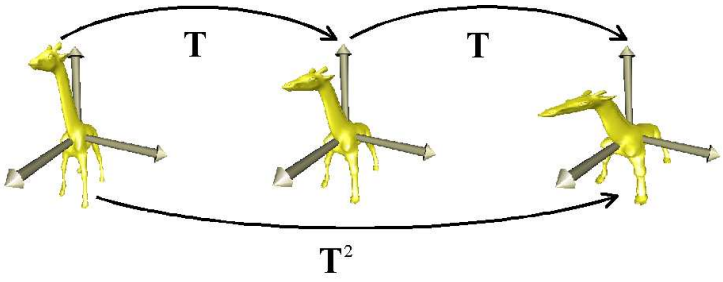


Figura 3.6: Escala

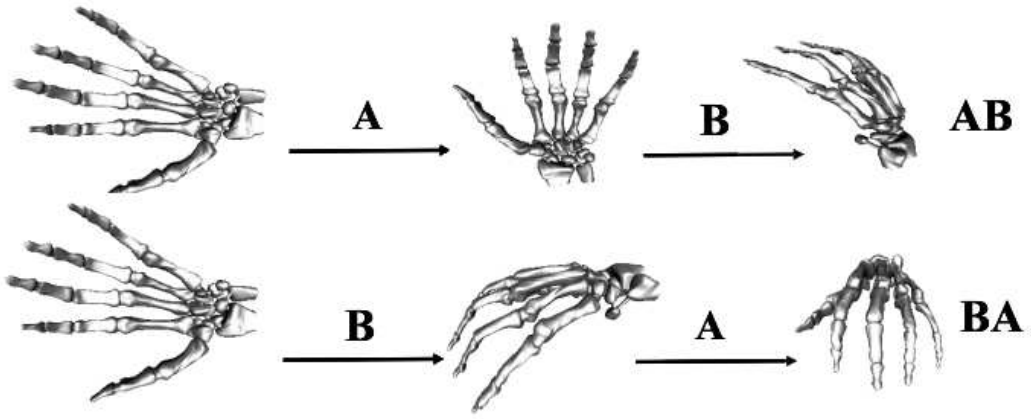


Figura 3.7: Produto de matrizes não é uma operação comutativa.

comutativo, obteremos a definição desejada de *adição*. A idéia é dividir A e B em "partes bem pequenas" e multiplicá-las alternadamente. Partes pequenas de A e B são geradas usando multiplicação escalar com um número racional pequeno, digamos $\frac{1}{n}$:

$$\frac{1}{n} \odot A = A^{\frac{1}{n}} \qquad e \qquad \frac{1}{n} \odot B = B^{\frac{1}{n}}.$$

Informalmente falando, supomos que $\left(A^{1/n}B^{1/n}\right)^n$ seja mais próximo de $\left(B^{1/n}A^{1/n}\right)^n$ do que AB é de BA. A Figura 3.8 é uma visualização da idéia explicitada aqui.

Observe que a maior parte dos dois produtos é igual:

$$\left(A^{1/n}B^{1/n}\right)^n = \underbrace{A^{1/n}\left(B^{1/n}A^{1/n}B^{1/n}A^{1/n}B^{1/n}\right)}_{n\ vezes}$$

$$\left(B^{1/n}A^{1/n}\right)^n = \underbrace{\left(B^{1/n}A^{1/n}B^{1/n}A^{1/n}B^{1/n}\right)A^{1/n}}_{n\ vezes}$$

e a diferença entre eles é representada por $A^{1/n} = \frac{1}{n} \odot A$ ou respectivamente, por $B^{1/n} = \frac{1}{n} \odot B$. Da definição de $\odot$, temos que $0 \odot A = I$, e portanto a diferença

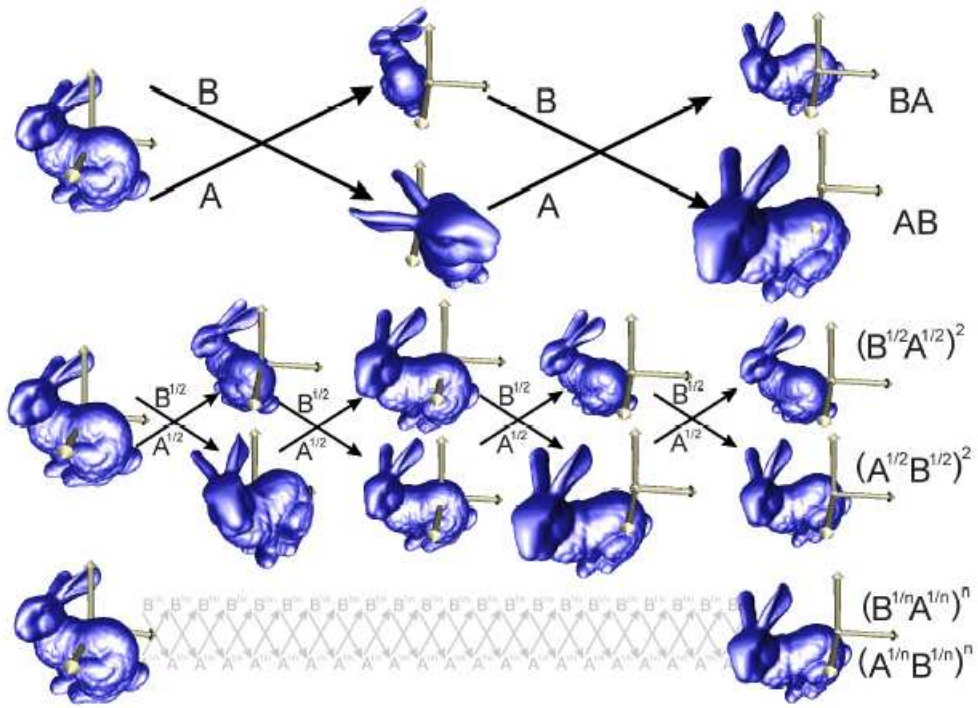


Figura 3.8: A adição de transformações. Dadas duas transformações, A e B, aplicar uma depois da outra (i.e, multiplicar as matrizes) geralmente fornece resultados diferentes dependendo da ordem das operações. Aplicando as n-ésimas partes das duas transformações diminui a diferença com relação à ordem em que são aplicadas. O limite $n \to \infty$ pode ser entendido como a aplicação de ambas transformações ao mesmo tempo. Esta é a definição geométrica intuitiva de uma adição comutativa para transformações baseada em matrizes.

$A^{1/n}$ ou $B^{1/n}$ deve desaparecer quando $\frac{1}{n} \to 0$, isto é, quando $n \to \infty$. Como conseqüência temos a seguinte definição:

Definição:

Sejam T_A e T_B duas transformações representadas, respectivamente, pelas matrizes reais quadradas A e B de mesma dimensão. A *adição* das transformações T_A e T_B é definida em termos de suas matrizes como:

$$A \oplus B = \lim_{n\to\infty} \left(A^{1/n} B^{1/n}\right)^n. \tag{3.3}$$

De uma maneira informal, $A \oplus B$ é a aplicação de A e B ao mesmo tempo. Como pode ser visto em Alexa [1], o limite acima existe quando as raízes reais das matrizes existem e o comportamento geométrico de $\oplus$ é similar ao produto usual de matrizes.

3.4 Computação

As operações $\odot$ e $\oplus$ agem sobre transformações, mas não foi por acaso que foram definidas em termos das matrizes que as representam. Trabalhar com matrizes é mais fácil e computacionalmente eficiente. Os operadores de adição e multiplicação escalar podem ser expressos e computados usando exponencial e logaritmo de matrizes. Um estudo detalhado sobre a álgebra de matrizes, incluindo as afirmações que serão feitas aqui foi feito por Horn e Johnson em [10].

A exponencial de uma matriz A é definida por:

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k, \tag{3.4}$$

o que define o logaritmo de uma matriz como a sua função inversa:

$$e^X = A \iff X = \log A. \tag{3.5}$$

Se a transformação não contém reflexão, o logaritmo de sua matriz de representação existe (para mais detalhes, ver Alexa [1]).

Fazendo uso dessa representação algébrica de matrizes, temos que os múltiplos escalares podem ser expressos como:

$$r \odot A = A^r = e^{\log(A^r)} = e^{r \log A} \tag{3.6}$$

e a adição $\oplus$ é dada por:

$$A \oplus B = e^{\log A + logB}. \tag{3.7}$$

Usando as equações (3.6) e (3.7) uma combinação linear de um número arbitrário de transformações T_i com pesos w_i, é computada como:

$$\bigoplus_i w_i \odot T_i = e^{\sum_i w_i \cdot \log T_i}.$$

Para implementar a teoria descrita aqui, algoritmos para a computação de raiz, logaritmo e exponencial de uma matriz são necessários. Usamos em nosso método os descritos em Alexa [1].

Capítulo 4

O Método Proposto

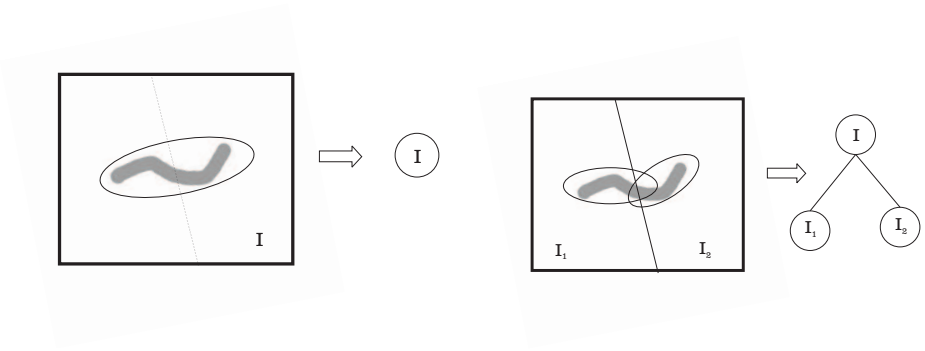
O nosso paradigma de reconstrução divide o algoritmo de reconstrução em três partes principais. Na primeira, construímos uma *bsp* hierárquica a partir dos momentos da imagem para todos os quadros (Seção 4.1). A estrutura armazena um conjunto de elipses de inércia, que representa a forma do objeto no quadro correspondente. Na segunda parte, a cada elipse do quadro atual é feita a correspondência a uma elipse no quadro consecutivo (Seção 4.2). Assim, podemos acompanhar o objeto, bem como uma parte específica, através da seqüência de imagens. Esses dois estágios foram descritos em [26]. No último estágio do algoritmo, computamos transformações geométricas entre as elipses correspondentes, e assim obtemos a representação do movimento, e a sua reconstrução é feita interpolando tais transformações (Seção 4.3). Esse último passo está descrito em [25].

4.1 Construção da Estrutura Hierárquica

Uma árvore *bsp* é construída para cada quadro da seqüência de imagens. Nela, cada nó representa uma elipse de inércia que aproxima uma parte do objeto. O nível k da árvore (0 para a raiz) tem 2^k nós, correspondendo a um conjunto de elipses que representa o objeto naquele nível. Assumimos que a *bsp* tem o mesmo número de níveis para todos os quadros da seqüência de imagens.

Cada nó contém os seguintes parâmetros:

- μ_{00} - momento central de ordem zero;
- μ_{11}, μ_{02}, μ_{20} - momentos centrais de segunda ordem;



(a) (b)

Figura 4.1: Construção da *bsp*: (a) nível 0, (b) nível 1

- c = (x_c, y_c) - centro da elipse de inércia (centróide do objeto);
- w, l - comprimentos dos semi-eixos maior e menor da elipse de inércia, respectivamente;
- θ - ângulo de orientação do maior eixo da elipse;
- d - direção do menor eixo da elipse e da linha de partição da *bsp*;
- M - matriz que representa a transformação geométrica entre esta elipse e sua correspondente no quadro anterior. (No primeiro quadro, a matriz identidade é armazenada.)

Os parâmetros acima são calculados de acordo com as equações (2.10) e (2.11), com exceção da direção d, cuja definição é dada por:

$$d = (\cos(\theta + \frac{\pi}{2}), \sin(\theta + \frac{\pi}{2})),$$

com θ como em (2.10).

Dado um quadro i da seqüência e o número k de níveis da árvore, a rotina que constrói a *bsp* consiste dos seguintes passos:

1 - Calcule μ_{00}, μ_{11}, μ_{02}, μ_{20}, c, w, l, θ, d e obtenha a elipse centrada em c e eixo menor com direção d. Ponha esta informação no nó raiz da árvore. (Figura 4.1(a))

2 - Subdivida a imagem na direção d e acrescente dois nós filhos à raiz, cada um

contendo as informações correspondentes a uma sub-imagem. (Fig. 4.1(b))

3 - Para cada nó filho, aplique recursivamente o algoritmo (passos 1 e 2) até o nível k.

Após aplicar essa rotina para cada imagem da seqüência, obtemos uma nova seqüência de quadros onde a estrutura do objeto é representada pelo conjunto de elipses de inércia. (Fig. 4.2)

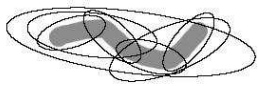


Figura 4.2: Objeto aproximado por 2^k elipses em cada nível. Estrutura com 3 níveis (0, 1 e 2).

A fim de demonstrar os resultados do algoritmo, damos exemplos de três formas com diferentes topologias, ambas mostradas na Figura 4.3. A primeira é uma forma com topologia de "S". O segundo exemplo mostra um objeto um pouco mais complicado com forma de "X" e a terceira possui a topologia de um "O". Note que o método captura a forma do objeto muito bem em todos os casos.

4.2 *Tracking*

Combinar as estruturas hierárquicas das *bsp*'s quadro a quadro significa que dada uma parte específica do objeto, devemos identificar em cada *bsp* o semi-plano (com respeito a linha de partição) que a contém, e verificar se eles são o mesmo em cada quadro. Sem esta verificação é possível criar correspondências erradas.

Assim, considere bsp_j a árvore do quadro j e d_{ij} a linha de partição da *bsp* para cada nó i de bsp_j.

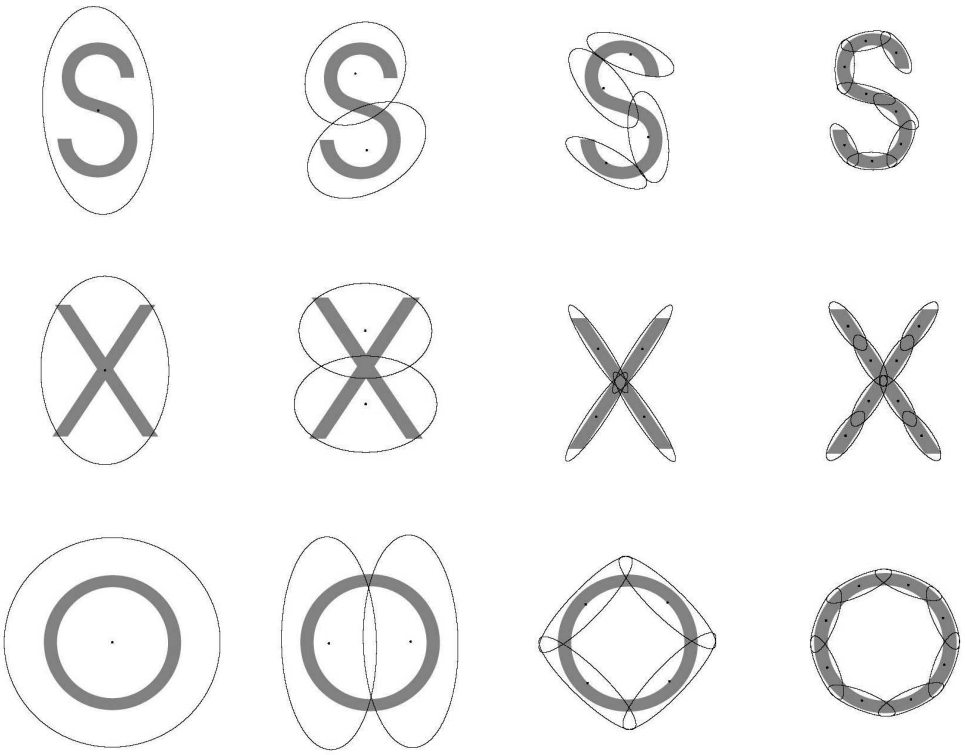


Figura 4.3: Objetos aproximados por elipses nos níveis 0, 1, 2 e 3, respectivamente.

Para garantir a correspondência entre as elipses de quadros consecutivos, é necessário manter a consistência de suas direções d_{ij}, de acordo com o ângulo de orientação θ que foi calculado para cada elipse.

Se $\langle d_{ij}, d_{i(j+1)} \rangle \leq 0$, o valor correto para θ é $\theta = (\pi + \theta)$. Sem esse teste de coerência, podemos obter correspondências erradas. Assim, $bsp_{(j+1)}$ depende de bsp_j e a rotina anterior se torna:

1 - Construa bsp_0 para o quadro 0 como descrito na seção anterior, e considere d_{00} como a linha de partição da *bsp* contida no nó raiz.

Para construir bsp_1 para o quadro 1 aplique os seguintes passos:

2 - Calcule μ_{00}, μ_{11}, μ_{02}, μ_{20}, c, w, l, θ, d_{01} e obtenha a elipse de inércia

centrada em c e eixo menor com direção d_{01}. Armazene esta informação no nó raiz de bsp_1.

3 - Verifique se a direção d_{01} é compatível com a direção d_{00},i.e., se $\langle d_{00}, d_{01} \rangle > 0$. Caso contrário, faça $\theta = (\pi + \theta)$ e calcule d_{01} novamente.

4 - Calcule a transformação entre a elipse armazenada no nó raiz de bsp_0 e a contida na raiz bsp_1. Armazene a matriz M que representa esta transformação geométrica.

5 - Acrescente dois nós filhos ao nó raiz de bsp_1, e para cada um, aplique a rotina recursivamente (passos 2, 3 e 4) até o nível k.

Para cada par de quadros consecutivos aplique o algoritmo acima.

Ao final, as *bsp's* são tais que se uma parte específica do objeto estiver contida no primeiro nó filho à esquerda da raiz da *bsp* do quadro inicial, então ela estará contida no mesmo nó em todas as outras árvores.

As figuras 4.4 e 4.5 ilustram o algoritmo de *tracking*. Na Figura 4.4, são mostradas todas as elipses do nível 3 da *bsp* que aproximam o objeto. Observe que as elipses capturam o movimento em toda parte do objeto. Na Figura 4.5 mostramos somente uma elipse do nível 2 de decomposição, a fim de demonstrar que o algoritmo é capaz de manter para cada quadro uma correspondência apropriada entre os conjuntos de elipses.

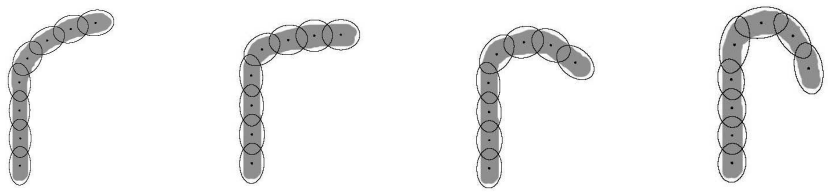


Figura 4.4: *Tracking* do objeto no nível 3, mostrando todas as elipses deste nível

4.3 Interpolação de Transformações

O movimento do objeto é representado hierarquicamente nas árvores *bsp's* que foram construídas e combinadas quadro a quadro. Para reconstruir

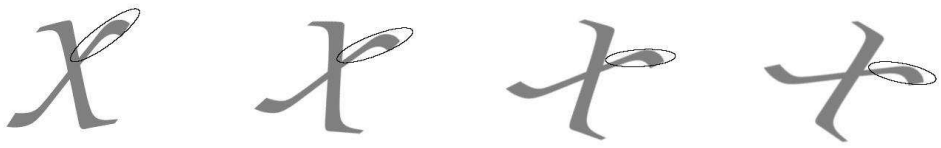


Figura 4.5: *Tracking* do objeto no nível 2 (somente uma elipse é mostrada)

esse movimento, interpolamos as transformações geométricas contidas nessas estruturas usando o método descrito no capítulo 3 e calculamos novas transformações contendo os parâmetros interpolados.

Dadas bsp_j e bsp_{j+1} as *bsp's* de dois quadros consecutivos na seqüência de imagens, e um número real $0 < p < 1$, representando a imagem I_p entre I_j(posição 0) e I_{j+1}(posição 1), o pseudo-código da rotina que constrói uma nova árvore entre as árvores consecutivas é dado pelo Algoritmo 1.

Algorithm 1 BSP = InterpoleBSP(bsp_j, bsp_{j+1}, p)

```
if bsp_j != NULL e bsp_{j+1} != NULL then
    BSP ⇐ NovoNó()
    InterpoleNó(BSP, bsp_j, bsp_{j+1}, p)
    BSP→filho1 = InterpoleBSP(bsp_j→filho1, bsp_{j+1}→filho1, p)
    BSP→filho2 = InterpoleBSP(bsp_j→filho2, bsp_{j+1}→filho2, p)
    return BSP
else
    return NULL
end if
```

A rotina `InterpoleNó` interpola os parâmetros armazenados em dois nós correspondentes, e é dada pelo Algoritmo 2.

Para obter todos os benefícios dessa interpolação, é necessário que o movimento esteja bem representado pelos conjuntos de elipses. Esse é o motivo pelo qual assumimos que o movimento é suave, não permitindo assim mudanças bruscas de topologia.

Dessa forma, podemos obter de maneira robusta aproximações para o movimento original. Quanto mais transformações obtivermos interpolando duas outras consecutivas, melhor será a reconstrução do movimento original.

Algorithm 2 InterpoleNó(BSP, bsp_j, bsp_{j+1}, p)

BSP$\rightarrow$M = ($\text{bsp}_{j+1}\rightarrow$M)^p
BSP$\rightarrow$c = (BSP$\rightarrow$M) $\cdot$ ($\text{bsp}_j\rightarrow$c)
A, B $\Leftarrow$ extremidades do eixo maior da elipse em bsp_j
a, b $\Leftarrow$ extremidades do eixo menor da elipse em bsp_j
A' = (BSP$\rightarrow$M) $\cdot$ A
B' = (BSP$\rightarrow$M) $\cdot$ B
a' = (BSP$\rightarrow$M) $\cdot$ a
b' = (BSP$\rightarrow$M) $\cdot$ b
BSP$\rightarrow$w = $\parallel B' - A' \parallel /2.0$
BSP$\rightarrow$l = $\parallel b' - a' \parallel /2.0$
BSP$\rightarrow\theta$ = ângulo($\overrightarrow{A'B'}$, eixo-x)

Capítulo 5

Aplicação: Reconstrução de Formas Intermediárias

Assumindo que as imagens da seqüência são binárias, que elas contêm apenas um objeto e que o movimento é suave, como em nosso algoritmo, estamos interessados em reconstruir a forma contida na imagem I^* entre duas outras imagens consecutivas I_i e I_{i+1} da seqüência original.

O primeiro passo é construir uma *bsp* para cada imagem e fazer a correspondência entre elas. Para isso, precisamos escolher o melhor número k de elipses de inércia que representarão a forma do objeto. Lembre que todas as *bsp's* tem a mesma quantidade de níveis, e o nível k contém 2^k elipses. Depois, as transformações geométricas entre I_i e I_{i+1} são calculadas e armazenadas na árvore do quadro I_{i+1}.

Suponha que a imagem I^* contenha a forma que se movimentou apenas a metade do movimento estimado entre I_i e I_{i+1}. Isso significa que queremos aplicar à cena em I_i a metade da transformação que leva I_i em I_{i+1}. Neste caso, usamos $p = 0.5$ no algoritmo de interpolação (Fig. 3.3).

Para cada nó folha da árvore interpolada identificamos sua sub-imagem correspondente em I_i. Para cada localização de *pixel* na sub-imagem, a inversa da transformação armazenada no nó folha correspondente é aplicado, obtendo a localização de um *pixel* em I_i. O *pixel* na nova imagem recebe a mesma cor que o *pixel* da imagem original. Os resultados fracionários, na atual implementação, são apenas truncados, gerando o problema de *aliasing* na imagem.

Para reconstruir movimento de formas rígidas, apenas o nó raiz da *bsp* (nível $k = 0$) é necessário, visto que o movimento em cada parte da forma é o mesmo. Isso significa que ela é representada por somente uma elipse. A

(a)

(b)

Figura 5.1: (a) A imagem em tom de cinza foi reconstruída usando $p = 0.5$. (b) Elipses representando a forma em cada quadro.

Figura 5.1 mostra dois quadros consecutivos e um intermediário, onde a forma foi interpolada com $p = 0.5$. As elipses de representação em cada quadro também são mostradas. Além de rotação e translação, este exemplo contém uma escala suave. A Figura 5.2 mostra interpolações de três diferentes formas com $p = 0.25$, 0.50 e 0.75. Os movimentos dos objetos são compostos por translação e/ou rotação e/ou escala suave. Mais exemplos e algumas animações podem ser vistas em *http://w3.impa.br/ lourena/sib04/*.

Uma limitação dessa abordagem para a interpolação de formas não-rígidas é a formação de descontinuidades (Fig. 5.3). Isso acontece quando usamos informações armazenadas em um nível k da *bsp*, com $k > 0$, no algoritmo de reconstrução. A causa desse efeito é que, até o presente momento, o algoritmo lida com o movimento em cada sub-imagem individualmente.

Mesmo assim, usar um nível $k > 0$ fornece uma melhor reconstrução para a forma (Fig. 5.4). Embora descontinuidade apareça, ela diminui quando k aumenta (Fig. 5.5).

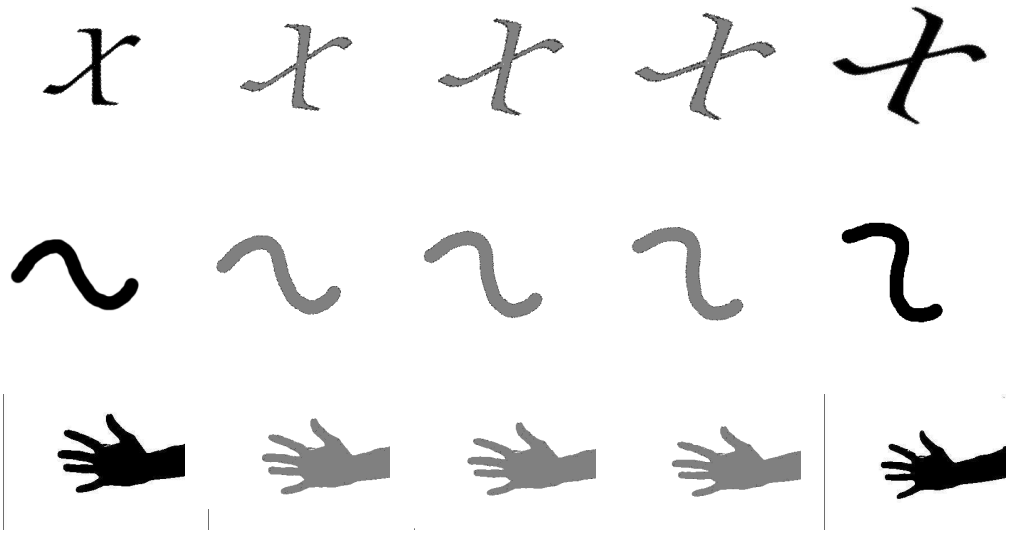


Figura 5.2: Reconstrução de formas intermediárias. As imagens em tons de cinza são interpolações dos quadros originais em negrito, com $p = 0.25$, 0.50 e 0.75, respectivamente. Na primeira linha o movimento contém rotação, translação e uma escala suave. Na segunda, o movimento é uma rotação e na última o movimento contém translação com uma suave rotação.

Figura 5.3: Descontinuidade da imagem. A imagem em tom de cinza foi interpolada usando $p = 0.5$ e $k = 1$.

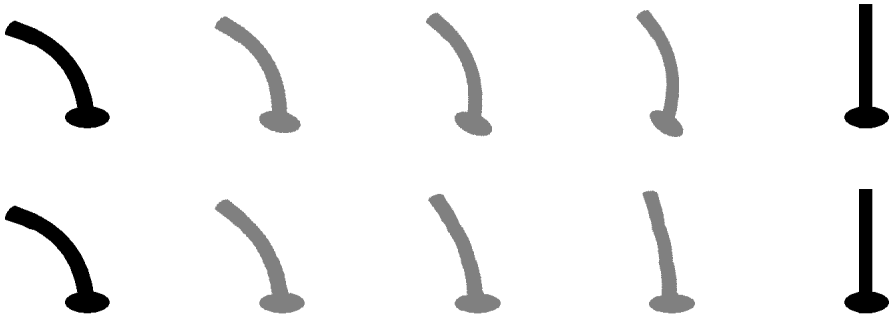


Figura 5.4: Embora para movimento de formas não-rígidas as imagens interpoladas sejam descontínuas quando $k > 0$, a reconstrução para esses valores é melhor do que para $k = 0$. Primeira linha: $k = 0$. Segunda linha: $k = 2$. As imagens foram interpoladas com $p = 0.25$, 0.50 e 0.75.

(a) (b)

Figura 5.5: A descontinuidade da imagem diminui quando k aumenta. (a) $k = 1$. (b) $k = 2$.

Capítulo 6

Conclusão

Esta dissertação apresentou um método para reconstrução do movimento projetado nas imagens de uma seqüência de vídeo. Mais do que isso, a técnica apresentada foi capaz de obter a estrutura da forma do objeto em movimento, e acompanhá-la na seqüência. Momentos foram utilizados para fazer a representação das formas. Para isso, foi construída uma estrutura hierárquica *bsp* em cada imagem, aproximando a forma por um conjunto de elipses de inércia em cada nível. *Tracking* foi feito explorando a coerência estrutural e temporal da representação. Transformações geométricas entre as estruturas elipsoidais foram computadas e a reconstrução do movimento foi obtida interpolando-as.

Usamos o algoritmo para reconstruir formas de objetos que possuíam movimento rígido e não-rígido. Para o primeiro caso, o método funcionou muito bem, tendo sido necessário usar apenas o primeiro nível das estruturas. Já para o movimento não-rígido, foi feita uma análise mais detalhada, e embora o método não tenha obtido resultados tão bons quanto no caso rígido, damos indicações que melhorias podem ser feitas.

Alguns problemas que não tratamos neste trabalho são oclusão, *tracking* de múltiplos objetos e descontinuidades de movimento. Trabalhos futuros vão nessas direções. Pretendemos melhorar a reconstrução de formas que possuem movimento não-rígido. Outro interesse é aplicar o algoritmo para acompanhar objetos reais em vídeo.

Uma possível aplicação consiste em transferir o movimento capturado de um determinado objeto para outro. Isto poderia ser usado para produzir animação de personagens baseado no comportamento de uma animação existente. Também gostaríamos de estender nosso método para o espaço tridimensional. A idéia seria usar elipsóides ao invés de elipses, obtendo a representação 3D do objeto.

Bibliografia

[1] M. Alexa. Linear combination of transformations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387. ACM Press, 2002.

[2] F. Banegas, M. Jaeger, D. Michelucci, and M. Roelens. The ellipsoidal skeleton in medical applications. *Sixth Solid Modeling*, June 4-8 2001.

[3] J. Barron, D. Fleet, S. Beauchemin, and T. Burkit. Performance of optical flow techniques. *CVPR*, pages 236–242, 1992.

[4] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. Nara, Japan, April 1998.

[5] J. R. D. E. Difranco, T. J. Cham. Reconstruction of 3d figure motion from 2d correspondences. In *IEEE Int. Conference Computer Vision and Pattern Recognition*, Kauai, Hawaii, November 2001.

[6] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York,, 1973.

[7] J. Flusser. Fast calculation of geometric moments of binary images. In M. Gengler, editor, *Pattern Recognition and Medical Computer Vision*, pages 265–274. Illmitz, 1998.

[8] J. Flusser and T. Suk. Affine momemt invariants: A new tool for character recognition. *Pattern Recognition Letters*, 15:433–436, 1994.

[9] J. Flusser and T. Suk. On the calculation of image moments. Technical Report 1946, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, January 1999.

[10] R. A. Horn and C. A. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[11] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions and Information Theory*, IT-8:179–187, Feb 1962.

[12] X. Jiang and H. Bunke. Simple and fast computation of moments. *Pattern Recognition*, 24(8):801–806, 1991.

[13] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *First International Conference on Computer Vision*, 1:321–331, 1987.

[14] K. Lau, W. Siu, and N. Law. Best-fit ellipse concept from second order moments. In *IEEE International Conference on Information Technology and Applications*, pages 167–75, Bathurst, Australia, November 2002.

[15] S. Liao and Q. Lu. A study of moment functions and its use in chinese character recognition. In *ICDAR97*, page Poste, 1997.

[16] S. X. Liao and M. Pawlak. On image analysis by moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(3):254–266, March 1996.

[17] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[18] D. Metaxas. Shape representation and nonrigid motion tracking using deformable superquadrics. *Geometric Methods in Computer Vision*, pages 12–20, July 1 991.

[19] R. Mukundan and K. R. Ramakrishnan. *Moment Functions in Image Analysis: Theory and Applications*. World Scientific World Scientific Publishing Co Pte Ltd., Singapore, September 1998.

[20] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 3rd edition, 1991.

[21] M. J. Park, M. G. Choi, and S. Y. Shin. Human motion reconstruction from inter-frame feature correspondences of a single video stream using a motion library. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 113–120. ACM Press, 2002.

[22] A. Pentland. Modal descriptions for recognition and tracking. *IAPR Workshop on Machine Vision Applications*, pages 435–444, december 1992.

[23] M. Peura and J. Iivarinen. Efficiency of simple shape descriptors. World Scientific, 1997.

[24] Z. Popov and A. Witkin. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM Press/Addison-Wesley Publishing Co., 1999.

[25] L. Rocha, P. C. Carvalho, and L. Velho. Motion reconstruction using moments analysis. In *Proceedings of SIBGRAPI / SIACG*. IEEE Press, 2004.

[26] L. Rocha, L. Velho, and P. C. Carvalho. Image moments-based structuring and tracking of objects. In *Proceedings of SIBGRAPI 2002 - XV Brazilian Symposium on Computer Graphics and Image Processing*. SBC - Sociedade Brasileira de Computacao, IEEE Press, October 2002.

[27] K. J. B. S. A. Dudani and R. B. McGhee. Aircraft identification by moment invariants. *IEEE Trans. Comput. C - 26*, pages 39–46, 1977.

[28] S. Sclaroff and J. Isidoro. Active blobs. Technical Report 1997-008, 5 1997.

[29] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, Jun 1994.

[30] K. Shoemake. Animating rotation with quaternion curves. In *"Proceedings of" SIGGRAPH 1985*, volume 19, pages 245–254. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 22–26 1985.

[31] K. Shoemake. *Graphics Gems II*, chapter Quaternions and 4x4 matrices, pages 351–354. Academic Press, Boston, 1991.

[32] K. Shoemake and T. Duff. Matrix animation and polar decomposition. In *Graphics Interface '92*, pages 258–264, 1992.

[33] C. H. Teh and R. T. Chin. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):496–513, July 1988.