DEFORMAÇÃO DE FORMAS 2D BASEADA EM RESTRIÇÕES POSICIONAIS INTUITIVAS E MANIPULAÇÃO DE CAMADAS

Tiago de Souza Mota

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Claudio Esperança
Antonio Alberto Fernandes de Oliveira

Rio de Janeiro
Junho de 2011

DEFORMAÇÃO DE FORMAS 2D BASEADA EM RESTRIÇÕES POSICIONAIS INTUITIVAS E MANIPULAÇÃO DE CAMADAS

Tiago de Souza Mota

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Claudio Esperança, Ph.D.

Prof. Antonio Alberto Fernandes de Oliveira, D.Sc.

Prof. Ricardo Guerra Marroquim, D.Sc.

Prof. Emilio Ashton Vital Brazil, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2011

Mota, Tiago de Souza

Deformação de formas 2D baseada em restrições posicionais intuitivas e manipulação de camadas/Tiago de Souza Mota. – Rio de Janeiro: UFRJ/COPPE, 2011.

XII, 45 p.: il.; 29, 7cm.

Orientadores: Claudio Esperança

Antonio Alberto Fernandes de Oliveira

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 43 – 45.

1. Deformação de formas. 2. Restrições posicionais. 3. Animação 2D. 4. Camadas. I. Esperança, Claudio *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha família, pelo amor e apoio constantes.

*Às minhas avós Nice e Zélia (*in memoriam*).*

Agradecimentos

Agradeço aos meus pais pelo amor incondicional, pelos ensinamentos de vida e pelos sacrifícios que fizeram para que eu chegasse até aqui. Agradeço às minhas irmãs pela amizade, amor e incentivo. Aos meus avós, tios e primos pela preocupação e pela torcida. Aos amigos baianos, juizforanos e cariocas pelo companheirismo e motivação. Aos companheiros do GRVa pela amizade e conhecimentos adquiridos. Aos meus orientadores e demais professores pela paciência pelos ensinamentos ao longo desses anos. Agradeço também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro. Agradeço por fim a Deus, na certeza de que, sem ele, nada disso seria possível.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DEFORMAÇÃO DE FORMAS 2D BASEADA EM RESTRIÇÕES POSICIONAIS INTUITIVAS E MANIPULAÇÃO DE CAMADAS

Tiago de Souza Mota

Junho/2011

Orientadores: Claudio Esperança
Antonio Alberto Fernandes de Oliveira

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta um método de interação com sistemas de deformação de objetos bidimensionais baseado na manipulação de restrições posicionais intuitivas e na criação e edição de camadas que representam partições específicas do objeto. As camadas são deformadas de forma independente, mas podem se relacionar por meio de regiões de interesse escolhidas pelo usuário. Através de ferramentas para deformação e edição das camadas, é possível realizar operações interessantes no objeto como alteração na ordem de exibição das partições, criação de deformações locais e aplicação de operações de edição de imagens. O método é mais recomendado para animação de personagens em forma de *cartoons* e é particularmente interessante para objetos que possuam algum tipo de sobreposição de regiões.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

2D SHAPE DEFORMATION BASED ON POSITIONAL CONSTRAINTS AND INTUITIVE MANIPULATION OF LAYERS

Tiago de Souza Mota

June/2011

Advisors: Claudio Esperança
Antonio Alberto Fernandes de Oliveira

Department: Systems Engineering and Computer Science

This work presents a method of interaction with systems of two-dimensional objects deformation based on the intuitive manipulation of positional constraints and the creation and editing of layers that represent specific partitions of the object. The layers are deformed in a independent way, but they can be related through regions of interest chosen by the user. Using tools for editing and for deformation of layers, it's possible to perform interesting operations on the object like changing the order of the partitions, creating local deformations and applying image editing operations. The method is best recommended for animation of characters in the form of cartoons and is particularly useful for objects that have some kind of overlapping regions.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O estudo da deformação de objetos bidimensionais tem por objetivo prover métodos eficientes para a manipulação dos mesmos, permitindo transformações geométricas como translação, rotação e alterações no comprimento, largura e curvatura dos objetos representados na forma de imagens 2D. Esses métodos se mostram muito importantes em diversas aplicações, como no enriquecimento de interfaces gráficas [10], na animação de personagens [11], na edição de imagens, e na manipulação de objetos em tempo real [12].

A utilização dos métodos de deformação de imagens se torna particularmente interessante quando é feita no contexto da animação de personagens, pois evita o trabalho de redesenhar manualmente o mesmo personagem em poses diferentes.

Na criação de um sistema de deformação de imagens 2D, deve ser considerada uma série de fatores que se tornam importantes para a geração de bons resultados:

- Desempenho: Na grande maioria das vezes se está interessado em um sistema interativo, que gere resultados em tempo real a partir de uma série de operações simples.
- Natureza do objeto a ser deformado: A maneira com que esses objetos se comportam no mundo real deve ser respeitada na escolha do método a ser utilizado para sua deformação. Em outras palavras, o resultado da deformação deve ser fisicamente plausível.
- Consistência dos resultados: A técnica deve permitir pouca margem para erros ou ruídos. Não é interessante, por exemplo, utilizar um sistema que não gere resultados visualmente agradáveis.
- Interface: Uma interface intuitiva e de fácil uso permite uma maior eficiência na manipulação do sistema por parte do usuário.

Grande parte dos métodos de deformação de formas bidimensionais têm sua origem no estudo de deformações de objetos tridimensionais. Apesar de nem sempre as propriedades que se deseja preservar serem a mesmas, os resultados esperados são bem similares.

Assim, é comum encontrarmos referências a trabalhos de manipulação de formas 3D nos trabalhos de manipulação de formas 2D.

Diversos paradigmas para a deformação de imagens já foram propostos, sendo que o mais popular é conhecido como deformação de forma livre (FFD, *free form deformation*) e pode ser visto no trabalho de MacCracken e Joy [1]. Os métodos que se baseiam nesse paradigma conseguem alcançar bons resultados através de subdivisões no domínio da imagem e obtêm seus resultados a partir da manipulação de pontos de controle. Cada subdomínio criado pode ser acompanhado de um ponto de controle, o qual o usuário move para criar a deformação desejada (ver figura 1.1). O problema dessa abordagem é que criar esses subdomínios é um trabalho tedioso e mover os pontos de controle para se criar a distorção desejada não é uma tarefa simples, uma vez que podem ser muitos os pontos criados. Além disso, os métodos de FFD não levam em consideração a forma natural na qual os objetos se movem no mundo real.

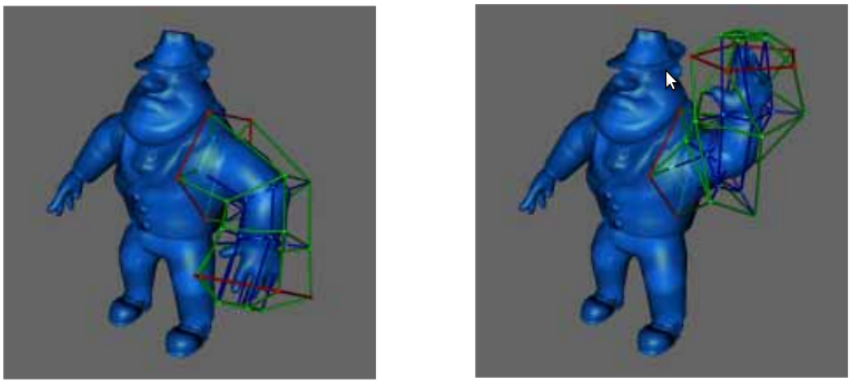

Figura 1.1: Exemplo de deformação de forma livre. Imagem retirada de [1].

Também são bastante conhecidos os métodos de deformação baseados na manipulação de esqueletos [2] (ver figura 1.2). Essa abordagem é mais intuitiva que a anterior e, como o nome sugere, é fundamental a criação de um esqueleto que represente de maneira fiel as características articuladas do objeto. Assim, a forma do objeto é alterada através da movimentação dos ossos e articulações do esqueleto criado. Estes métodos, no entanto, não alcançam resultados satisfatórios para objetos que não possuam ou que não possam ser representados por uma estrutura articulada. Além disso, definir um esqueleto não é uma tarefa trivial e definir os pesos para as articulações pode-se tornar bastante tedioso. Assim como os métodos de deformação de forma livre, os métodos baseados na manipulação de esqueletos são eficientes computacionalmente e fáceis de serem implementados, mas também não apresentam ferramentas convenientes de interação com o usuário [13].

Existem também os métodos de deformação que fazem o uso de simulações baseadas em física [3] (ver figura 1.3). Esses métodos também são conhecidos como métodos de otimização não lineares. Os métodos dessa abordagem apresentam bons resultados,

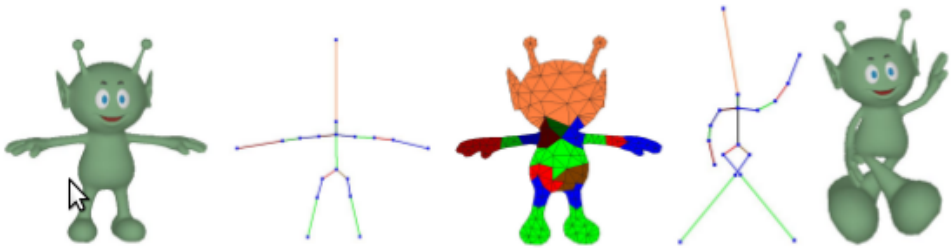

Figura 1.2: Exemplo de deformação baseada na manipulação de esqueletos. Imagem retirada de [2].

porém geralmente são demasiadamente lentos para uma abordagem interativa e apresentam um custo computacional muito alto.

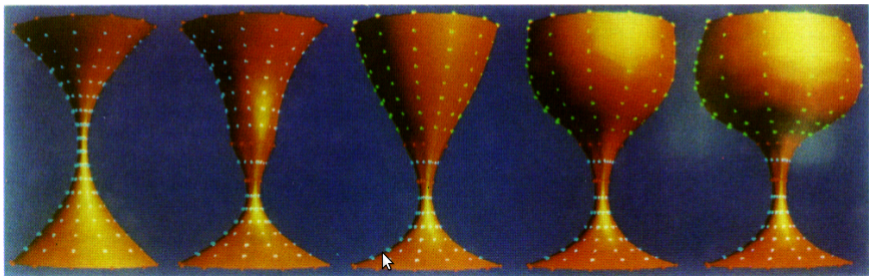

Figura 1.3: Exemplo de deformação empregando simulações baseadas em física. Imagem retirada de [3].

Este trabalho apresenta uma nova abordagem de interação com métodos de deformação de formas 2D baseadas em restrições posicionais intuitivas, introduzidos no trabalho de Igarashi et al. [4]. Estes métodos são similares aos de FFD e neles o usuário escolhe alguns pontos no objeto para funcionarem como restrições posicionais, e através da movimentação das mesmas, o sistema de deformação realiza uma série de operações geométricas para obtenção de uma nova forma para o objeto. Nosso método se vale dos conceitos de edição de camadas e tratamento de imagens para incrementar o paradigma de deformação mencionado, através de uma interface interativa e de fácil uso. Nossa contribuição para os trabalhos desta abordagem foi a utilização de uma estrutura de camadas que define uma representação diferente para o objeto e permite uma maior liberdade em sua deformação.

Capítulo 2

Trabalhos relacionados

Neste trabalho estamos interessados no estudo das abordagens de deformação de imagens baseadas no uso de restrições posicionais. Estes métodos conseguem atingir resultados fisicamente aceitáveis para as mais diversas formas através da movimentação de pontos de controle pré-definidos pelo usuário. Além de terem uma interface mais intuitiva que as demais abordagens, esses métodos são capazes de representar, de forma aceitável, objetos de natureza genérica que possuam, ou não, um esqueleto.

Apesar de terem representações lineares e algoritmos diferentes, os métodos mencionados a seguir apresentam uma série de características em comum:

- Manutenção da forma das regiões internas aos objetos;
- Particionamento do objeto;
- Dados de entrada são as posições dos pontos escolhidos como restrições posicionais após estes serem movidos;
- Taxas de desempenho interativas;
- Interfaces intuitivas;

Este trabalho pode ser visto como um acréscimo aos trabalhos de Igarashi et al. [4] e Weng et al. [5]. Esses trabalhos abordam principalmente a resolução dos sistemas numéricos que visam encontrar uma nova configuração para os objetos após a interação. Contribuimos para estes trabalhos possibilitando novas formas de interação com o usuário através de uma interface baseada na manipulação de camadas e em operações de edição capazes de transformar os segmentos da imagem.

Neste capítulo serão apresentados os trabalhos que seguem o paradigma mencionado que mais influenciaram o nosso trabalho. Também será feita uma descrição de alguns métodos importantes para compreensão desta dissertação.

2.1 Manipulação tão rígida quanto possível de formas

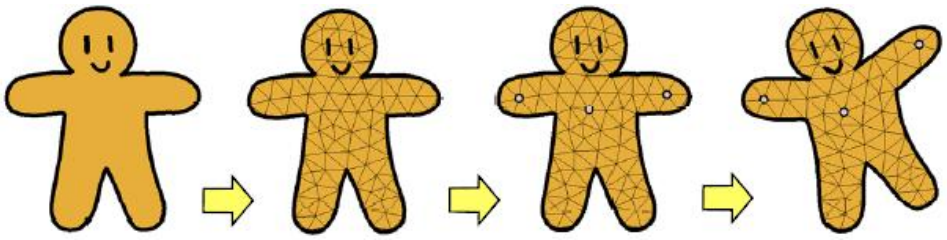

Figura 2.1: método desenvolvido por Igarashi. Imagem retirada de [4].

O primeiro trabalho de deformação de imagens baseados em restrições posicionais intuitivas foi desenvolvido por Igarashi et al. [4]. Através da criação de uma malha triangular 2D, ele permite deformar de maneira interativa a forma do objeto a partir da manipulação de alguns pontos de controle, que são escolhidos no início da interação. A partir da movimentação desses pontos de controle, o sistema computa a posição dos demais pontos da malha. A inovação nesse trabalho foi a preocupação com a manutenção da forma sob as restrições posicionais impostas.

O método emprega o conceito de rigidez interna à deformação de formas bidimensionais. No caso, se refere a uma resistência à mudança de forma, minimizada através de um processo de otimização linear que é feito em duas etapas. O algoritmo procura assim encontrar a configuração que minimiza a distorção em todos os triângulos da malha. Em cada uma das etapas se resolvem sistemas lineares que representam as alterações quanto à escala e orientação dos triângulos do objeto. As etapas correspondem à solução de dois sistemas lineares que representam as alterações nas propriedades de escala e rotação dos triângulos do objeto.

O problema de otimização é montado como um de minimizar o erro quadrático. Durante a interação, o problema já está estruturado. Dessa forma, é resolvido rapidamente.

A primeira tarefa a ser realizada é a obtenção de um polígono que represente a forma do objeto. Isso é feito, com a remoção manual do plano de fundo e com a aplicação do algoritmo *marching squares* (veja seção 2.3.1) para obtenção de silhueta (ou contorno). Tendo esta silhueta, um conjunto de pontos interiores é acrescido e com base neles e nos vértices do contorno uma triangulação com características de regularidade é obtida.

A partir desse momento, o objeto passará a ser representado por essa malha triangular. Muitos métodos podem ser usados para obtenção dessa malha, mas o de Markosian et al. [14] foi escolhido por conseguir alcançar melhores resultados. A obtida triangulação configura o objeto em sua forma inicial, e o usuário pode então escolher alguns de seus vértices como restrições posicionais.

O algoritmo tem como dados de entrada as coordenadas (x, y) dos vértices de controle já movimentados, e como dados de saída a posição dos demais vértices determinada em

função do movimento desses pontos de controle. Feita a especificação das restrições, o algoritmo procede em duas etapas.

2.1.1 Etapa 1

Nesta etapa é gerado um resultado intermediário minimizando uma métrica de erro que evita o cisalhamento e a expansão não uniforme da malha, porém permite rotação e escalamento uniforme. Ela é uma adaptação do caso 2D da edição do Laplaciano descrito em [15].

Considerando um triângulo qualquer (v_1, v_2, v_3), a função de erro é baseada na representação de um vértice qualquer (v_2) do triângulo num sistema de coordenadas (S) com origem em outro vértice (v_0) e tendo como base $(v_1 - v_0)$ e esse vetor rodado de $90°$. Reposicionando v_0 e v_1 o sistema é transformado em outro sistema (S'). Esta operação pode ser vista com clareza na figura 2.2.

O erro considerado (E) é a diferença entre as posições de v_2 na malha deformada (v_2') e $v_2^{desejado}$ é a posição desejada para v_2' depois da deformação, tendo em vista o deslocamento dos outros 2 vértices. Este cálculo é feito para todos os vértices da malha, considerando cada triângulo a que ele pertence:

$$v_2^{desejado} = v_0' + x_{01}\overrightarrow{v_0'v_1'} + y_{01}R_{90}\overrightarrow{v_0'v_1'}\,, \tag{2.1}$$

onde:

$$R_{90} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.2}$$

e v_0' e v_1' são posições transformadas de v_0 e de v_1 e x_{01} e y_{01} as coordenadas de v_2 no sistema inicial (S). Em cada triângulo temos então:

$$E_{\{v_0,v_1,v_2\}} = \sum_{i=1,2,3} \left\| v_i^{desejado} - v_i' \right\|^2 \tag{2.3}$$

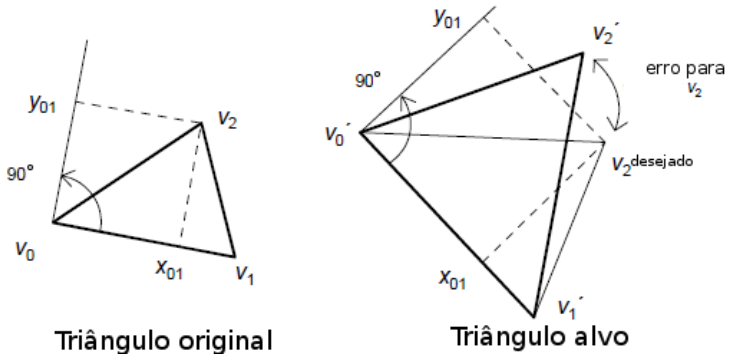

Figura 2.2: Métrica de erro da etapa 1. Imagem retirada de [4].

O erro para toda a malha é a soma de todos os erros dos triângulos. E como a métrica de erro é quadrática em v'. O erro da primeira etapa (E_1) pode ser expresso por:

$$E_{1\{v'\}} = v'^T G v', \tag{2.4}$$

onde G é uma matriz $|v|$ x $|v|$ que armazena as relações de proximidade entre os vértices da malha e v' é o vetor das posições dos vértices da malha após a deformação.

2.1.2 Etapa 2

Nesta etapa se determina para cada triângulo original(v_0, v_1, v_2) qual versão escalada e rodada dele melhor se adapta ao resultado obtido por ele na primeira etapa (v'_0, v'_1, v'_2). Essa versão $(v_0^{ajustado}, v_1^{ajustado}, v_2^{ajustado})$ é obtida minimizando para cada triângulo a função:

$$E_{f\left\{v_0^{ajustado}, v_1^{ajustado}, v_2^{ajustado}\right\}} = \sum_{i=1,2,3} \left\| v_i^{ajustado} - v'_i \right\|^2$$

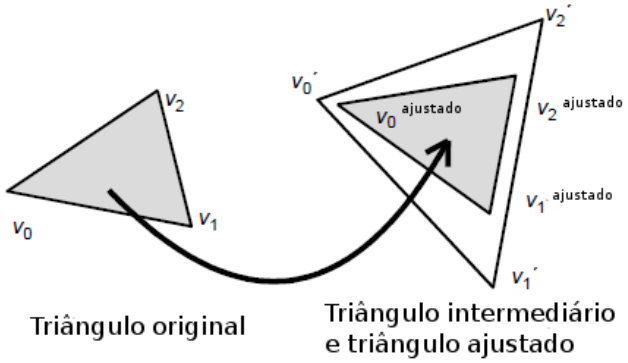

Figura 2.3: Ajuste de triângulos na etapa 2. Imagem retirada de [4].

Como a primeira etapa realiza um escalamento uniforme, o triângulo $(v_0^{ajustado}, v_1^{ajustado}, v_2^{ajustado})$ é congruente a (v_0, v_1, v_2) e podemos expressar $v_2^{ajustado}$ - $v_0^{ajustado}$ como combinação de linear de $\Delta_{v_1}^{ajustado} = v_1^{ajustado} - v_0^{ajustado}$ e $\Delta_{v_1}^{ajustado\perp}$ empregando os mesmos coeficientes (x_{01} e y_{01}) usados na primeira etapa exprimir $(v_2 - v_0)$ em função de $(v_1 - v_0)$ e $(v_1 - v_0)^{\perp}$. Podemos então escrever:

$$v_2^{ajustado} = v_o^{ajustado} + x_{01}\overline{v_0^{ajustado} v_1^{ajustado}} + y_{01} R_{90} \overline{v_0^{ajustado} v_1^{ajustado}}, \tag{2.5}$$

que explicita que temos apenas quatro variáveis livres por triângulo $(v_{0x}^{ajustado}, v_{0y}^{ajustado}, v_{1x}^{ajustado}, v_{1y}^{ajustado})$.

Pode-se, então, minimizar a função de erro igualando as suas derivadas parciais nas quatro variáveis livres a zero e ao final temos para cada vértice original um posicionamento ótimo considerando-se cada triângulo a que ele pertence.

É necessário reconciliar essas diferentes posições que correspondem ao mesmo vértice (ver figura 2.4). Para isso é definida uma função de erro quadrática a ser minimizada para encontrar as arestas de tamanho apropriado:

$$\sum_{(i,j)} \sum_{\Delta \ni \overline{v_i v_j}} \left\| \overline{v_i'' v_j''} - \overline{v_i^{\Delta} v_j^{\Delta}} \right\|^2, \tag{2.6}$$

onde v_R^{Δ} é a posição encontrada para $v_R^{ajustado}$ considerando-se o triângulo R a que ele pertence.

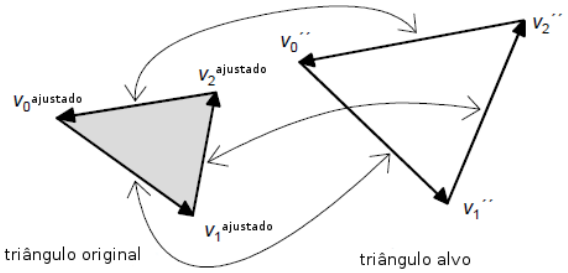

Figura 2.4: Ajuste de arestas na etapa 2. Imagem retirada de [4].

Após o ajuste de vértices, os vértices da malha assumem novas posições e o processo de deformação se encerra até a próxima interação introduzindo restrições posicionais. Este processo resulta numa deformação razoavelmente controlada e com distorção reduzida (veja exemplo na figura 2.5).

Figura 2.5: Exemplo da deformação proposta. Imagem retirada de [4].

2.2 Manipulação de formas 2D usando otimização de mínimos quadrados não lineares

Outro trabalho relevante utilizando restrições posicionais foi desenvolvido por Weng et al. [5]. Nele foi considerado que nem todos os componentes da função de energia que representam a deformação são lineares, e assim, ao invés de se tentar uma aproximação linear dos termos da função de energia a se minimizar, foi proposto o uso de componentes não lineares para a solução do sistema. Em vista disso, a função de energia (que não é quadrática) é solucionada utilizando métodos iterativos. O método tem como objetivo preservar duas propriedades geométricas importantes dos objetos bidimensionais:

Figura 2.6: Método desenvolvido por Weng. Imagem retirada de [5].

as coordenadas laplacianas da curva limitante do objeto e a área de porções locais dentro do objeto. Ao fim do procedimento iterativo, resultados fisicamente plausíveis são alcançados com taxa de desempenho interativa.

Além de possibilitar a preservação das propriedades locais do objeto, o trabalho de Weng et al. [5] introduz uma técnica para a preservação da área global do mesmo, o que é muito útil quando se deforma objetos de natureza genérica ou desprovidos de uma estrutura articulada.

Assim como no método proposto por Igarashi et al. [4], o objeto é representado por um polígono contendo os pontos de borda do objeto. Esse polígono é obtido através da remoção manual do plano de fundo e da aplicação de um algoritmo que realiza a obtenção de silhueta de forma automática, como o *marching squares* (ver seção 2.3.1). Após essa etapa, ocorre a inserção automática de pontos no interior do polígono e a geração de um grafo que conecta os vértices interiores e os da fronteira. Para a geração do grafo é usada uma técnica similar à da construção de grafos volumétricos proposta em [6] e a partir daí temos um grafo 2D (V, E), onde V é o conjunto de n vértices do grafo e E é o conjunto de arestas. O conjunto com todos os vértices do grafo (V) pode ser dividido em: vértices da polígono (V_p) e vértices do interior do grafo (V_g) (ver figura 2.7). Similarmente, o conjunto de arestas (E) pode ser dividido em dois: arestas do polígono (E_p) e arestas do interior (E_g).

A preservação das coordenadas laplacianas geralmente produz bons resultados em malhas 3D, porém em 2D sua aplicação de forma isolada é incapaz de manter o objeto sem distorções. Por este motivo, também é necessário um método para preservar as áreas de porções locais no interior do objeto. Para isso duas propriedades são importantes no grafo criado: a posição relativa de cada ponto interior em relação a seus vizinhos e o comprimento de cada aresta. A posição deformada de todos os vértices do grafo é obtida através da minimização de uma função de energia que consiste em quatro partes:

1. Coordenadas laplacianas da curva de fronteira.

2. Coordenadas da posição relativa.

3. Comprimento das arestas.

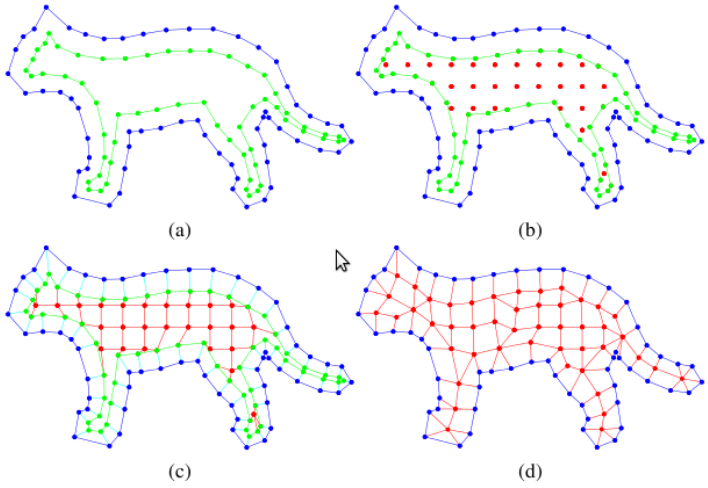

Figura 2.7: Objeto 2D e sua representação pelo grafo. Imagem retirada de [6].

4. Restrições posicionais.

2.2.1 Coordenadas laplacianas da curva de fronteira

O laplaciano da curva externa é definido para cada um de seus vértices v_p e é análoga para o caso de uma superfície. Assim, o laplaciano no ponto v_i é computado como a diferença entre ele mesmo e a média de seus vizinhos na curva:

$$L_p(v_i) = v_i - (v_{i-1} + v_{i+1})/2 \tag{2.7}$$

Assim para se preservar o laplaciano durante a deformação, é necessária a minimização da seguinte função:

$$\|L_p V_p - \delta(V_p)\|^2 , \tag{2.8}$$

onde:

- V_p é o vetor constituído pelos vértices de borda após a deformação;
- L_p é a matriz do sistema de igualdades (ver equação 2.7) para os vértices da borda deformada;
- $\delta(V_p)$ representa o vetor de laplacianos da borda da figura original.

2.2.2 Coordenadas da posição relativa

Para cada ponto v_i em V_g, se quer manter sua posição relativa com relação aos seus vizinhos. Para isso, é computado seu valor relativo para cada aresta do grafo adjacente a v_i

(ver figura 2.8):

$$w_{i,j} = \frac{tan(\alpha_j/2) + tan(\alpha_{j+1}/2)}{|v_i - v_j|}, \tag{2.9}$$

onde α_j é o ângulo formado pelo vetor $v_{j-1} - v_i$ e v_j–v_i.

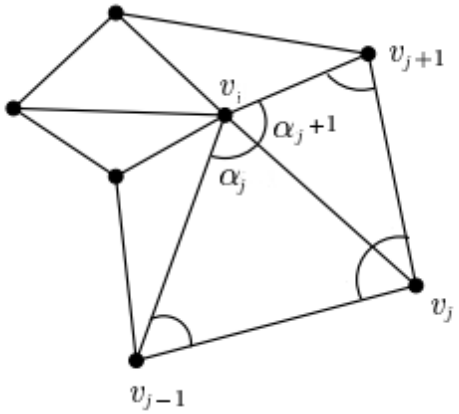

Figura 2.8: Exemplo de cálculo das coordenadas da posição relativa. Imagem alterada de [7].

Assim temos:

$$v_i - \sum_{(i,j)\in E} w_{i,j} * v_j = 0, \text{para } v_i \in V_g, \tag{2.10}$$

que em sua forma matricial é pode ser expressa por:

$$M_g V_g = 0, \tag{2.11}$$

onde M_g é a matriz contendo os valores das coordenadas da posição relativa nos pontos do interior do grafo.

Consegue-se preservar a posição relativa com relação aos vizinhos minimizando essa função de energia:

$$\|M_g V_g\|^2 \tag{2.12}$$

2.2.3 Comprimento das arestas

Como as coordenadas da posição relativa são invariantes à escala, faz-se necessária a preservação do comprimento das arestas. Isso é feito penalizando as alterações no tamanho de todas as arestas com a função:

$$\sum_{(i,j)\in E_g} \|(v_i - v_j) - e(v_i, v_j)\|^2, \tag{2.13}$$

onde:

$$e(v_i, v_j) = \|v_i - v_j\|,$$

e $(v_i - v_j)$ é o tamanho original da aresta antes da deformação.
A função de energia para este componente assume a forma:

$$\|HV - e(V))\|^2, \tag{2.14}$$

onde:

$$e(v) = \begin{pmatrix} e(v_{00}, v_{01}) \\ . \\ . \\ . \\ e(v_{n0}, v_{n1}) \end{pmatrix},$$

sendo v_{i0} o vértice inicial da aresta i, v_{i1} seu vértice final e $e(v)$ o valor da função para o componente que representa o com .

2.2.4 Restrições posicionais

Para alcançar uma deformação interativa, também é necessária a preservação dos pontos dados pelo usuário como restrições posicionais. Isso é feito através da criação de uma matriz (C) contendo a posição das restrições e da criação da função de energia que leva em consideração um vetor (U) com os pontos nas novas posições:

$$\|CV - U\|^2 \tag{2.15}$$

2.2.5 Resolução do sistema não linear

Considerando todos as quatro funções de energia, chega-se finalmente à função de energia do método, que é, em resumo, a soma de todas as quatro partes descritas:

$$\|LV - \delta(V)\|^2 + \|MV\|^2 + \|HV - e(V))\|^2 + \|CV - U\|^2 \tag{2.16}$$

Essa função pode ser reescrita como:

$$min_v \|AV - b(V)\|^2, \text{onde } A = \begin{pmatrix} L \\ M \\ H \\ C \end{pmatrix} e\ b(V) = \begin{pmatrix} \delta(V) \\ 0 \\ c(V) \\ U \end{pmatrix} \tag{2.17}$$

Esse sistema é resolvido através do método iterativo de Gauss-Newton na seguinte forma:

$$\min_{V^{k+1}} \left\| AV^{k+1} - b(V^k) \right\|^2, \tag{2.18}$$

onde V_k é o resultado da k-ésima iteração e V_{k+1} são as posições a se achar na iteração $k+1$. Como $b(V_k)$ já é conhecido na iteração atual, a equação pode ser resolvida através de um sistema de mínimos quadrados:

$$V^{k+1} = (A^T A)^{-1} A^T b(V^k) \tag{2.19}$$

Após a execução do método Gauss-Newton de vértices, os vértices da malha assumem novas posições e o processo de deformação se encerra até a próxima interação com as restrições posicionais. A figura 2.9 mostra um exemplo da interação com o sistema de deformação proposto por Weng et al. [5].

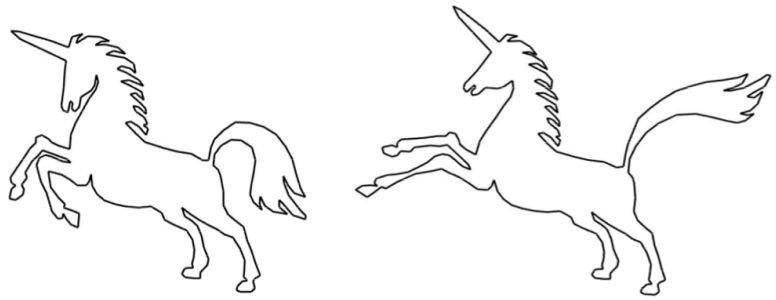

Figura 2.9: Exemplo da deformação proposta. Imagem retirada de [5].

2.3 Algoritmos auxiliares

Nesta seção serão apresentadas algumas rotinas importantes no processo de deformação sugerida por este trabalho. Elas serão apresentadas em uma ordem lógica de utilização no processo de criação de um sistema de deformação que utiliza o paradigma proposto.

2.3.1 Algoritmo Marching Squares

O algoritmo *marching squares* foi criado para a visualização de funções implícitas, porém é bastante utilizado para a realização de extração de contornos de objetos implícitos. Ele é a variante bidimensional para o conhecido algoritmo *marching cubes* [16].

Os dados de entrada para o algoritmo são os valores de um campo escalar, gerados a partir de uma função implícita que define o objeto do qual se deseja extrair o contorno. Os dados de saída dependem do contexto de aplicação do algoritmo, porém, na maioria

das vezes é um conjunto de linhas que representa uma aproximação para os pontos que envolvem o objeto. A função implícita, neste caso, define se um ponto do domínio no qual o objeto está inserido pertence ou não a ele.

A ideia básica do algoritmo é subdividir o espaço contendo o objeto implícito em células quadrangulares que armazenam em seus vértices os valores da função implícita. O objetivo é utilizar a estratégia de *divisão e conquista* para localizar uma aproximação válida para a silhueta (ou contorno) do objeto a partir da análise de cada célula independentemente. A análise é feita considerando que existem apenas dezesseis possíveis formas do contorno interceptar cada célula e uma tabela, que podemos chamar de *tabela de observação*, contendo essas formas, é criada para facilitar a identificação das células. O resultado do algoritmo é a soma dos resultados da análise de todas as células e os pontos exatos onde o contorno intercepta as células podem ser calculados através de interpolações lineares simples.

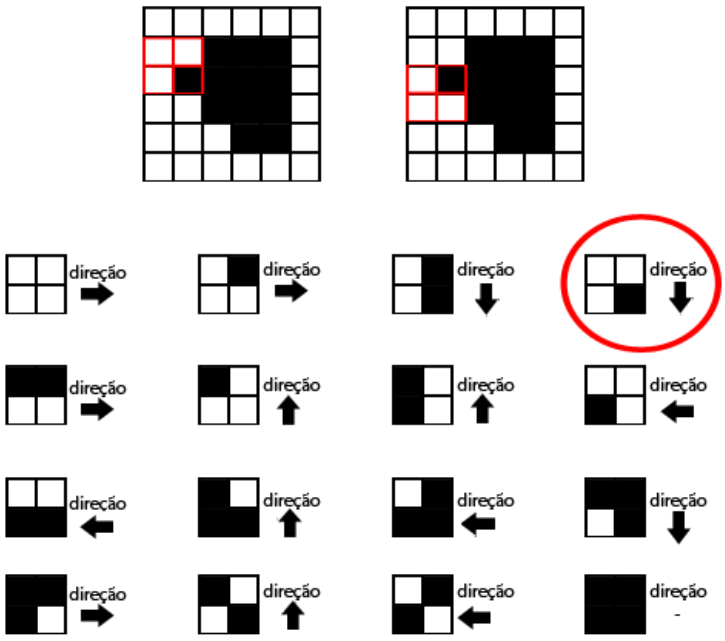

Figura 2.10: Variação do algoritmo marching squares para extração de contornos dos objetos contidos em imagens.

Quando o espaço no qual o objeto está inserido é discreto, como nos casos nos quais se deseja extrair os contornos de objetos contidos em imagens, o algoritmo recebe uma pequena alteração. Neste caso ele não necessariamente analisa todas as células do espaço da imagem. A ideia é percorrer as células até que se encontre a primeira célula contendo um valor de função implícita positivo (ou que indica que o local pertence ao objeto). A partir dessa momento, o algoritmo passa a percorrer o objeto analisando apenas células que contenham algum ponto de borda. Isso é feito definindo, a cada iteração, a direção da próxima célula a ser analisada de acordo com a *tabela de observação* (veja figura 2.10). O algoritmo se encerra quando alguma célula for analisada pela segunda vez. Assim,

o resultado é um polígono fechado contendo pontos que se distanciam entre seus dois vizinhos mais próximos em no máximo uma unidade do eixo x e/ou no eixo y.

2.3.2 Algoritmo Douglas-Peucker

O algoritmo Douglas-Peucker é um método heurístico de simplificação de curvas proposto no inicio da década de 1970. Sua autoria é creditada a vários autores (como Ramer et al. [17]), pois foi publicado em trabalhos de áreas diferentes em épocas similares, porém é comumente referenciado ao trabalho de Douglas e Peucker[18]. Devido a esta dubiedade na autoria, muitas vezes também encontramos referências ao método com o nome de *algoritmo Ramer–Douglas–Peucker* (RDP).

Este é o algoritmo mais utilizado para simplificação de polígonos e é classificado como o melhor método de simplificação que procura preservar a forma da linha original [19].

A ideia do algoritmo é bastante simples: dado um conjunto de pontos, o algoritmo traça uma segmento (r) de reta do primeiro ao último ponto do conjunto e descobre dentro do mesmo qual o ponto (p) mais distante deste segmento de reta. Se a distância de p até a r for maior do que um valor limite (chamado de valor de tolerância), a aproximação é aceita. Caso contrário o algoritmo divide o conjunto de pontos em dois (um contendo todos os pontos do inicio até p e outro contento todos os pontos de p até o último). Essa rotina é repetida até que a aproximação de todos os conjuntos criados seja aceita (ver figura 2.11). O valor de tolerância define quão boa será a aproximação para o conjunto de pontos.

2.3.3 Algoritmo de Análise e Síntese

O algoritmo da análise e síntese é utilizado neste trabalho como estratégia para o preenchimento de espaços vazios dentro do contexto de reconstrução de imagens. Ele gera uma nova configuração para a imagem e é classificado como algoritmo em pirâmide. Algoritmos dessa natureza são muito usados na área de processamento de imagens, e geralmente consistem em obter uma hierarquia de imagens com níveis e resoluções diferentes para realizar transformações e análises na imagem original.

Como o nome sugere, o algoritmo pode ser descrito em duas etapas:

- Análise: Gera uma pirâmide a partir da imagem original, fazendo com que assim, o menor nível seja o de maior resolução. Cada píxel na imagem correspondente ao nível é calculado como sendo a média ponderada dos 4 píxeis mais próximos a ele na imagem correspondente ao nível anterior. Como esses 4 píxeis são equidistantes do pixel do nível superior, o peso para cada um é igual. Caso algum desses 4 píxeis seja vazio, ele não será levado em consideração na média.

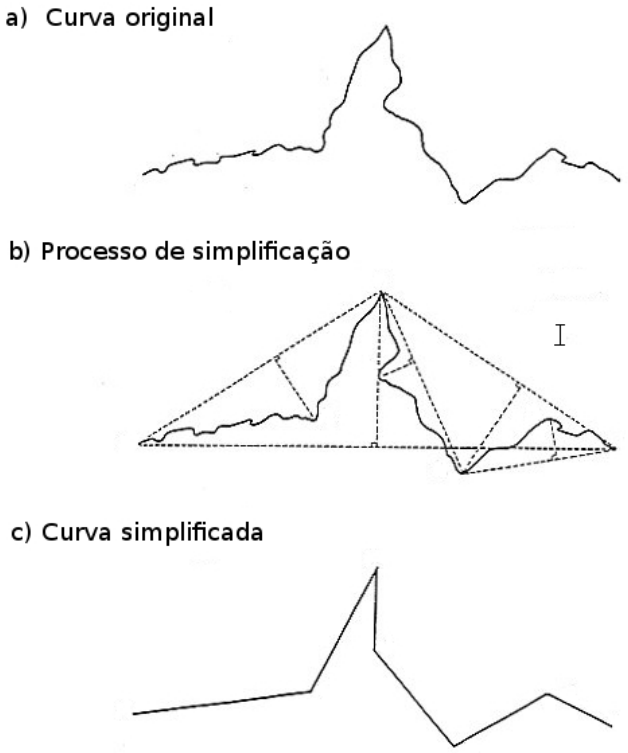

Figura 2.11: Exemplo do algoritmo Douglas-Peucker. Imagem retirada de [8].

- Síntese: Gera uma pirâmide a partir da imagem do último nível criado na etapa anterior (de cima para baixo). Esta é a etapa em que ocorre de fato a recuperação dos píxeis vazios. A imagem correspondente a um nível é gerada a partir da imagem correspondente ao mesmo nível no processo de análise, com exceção dos píxeis vazios. O valor a ser inserido nos píxeis vazios é calculado como sendo a média ponderada dos 4 píxeis mais próximos a ele na imagem correspondente ao nível superior na pirâmide. O pesos são relativos à distância entre píxeis (ver figura 2.12).

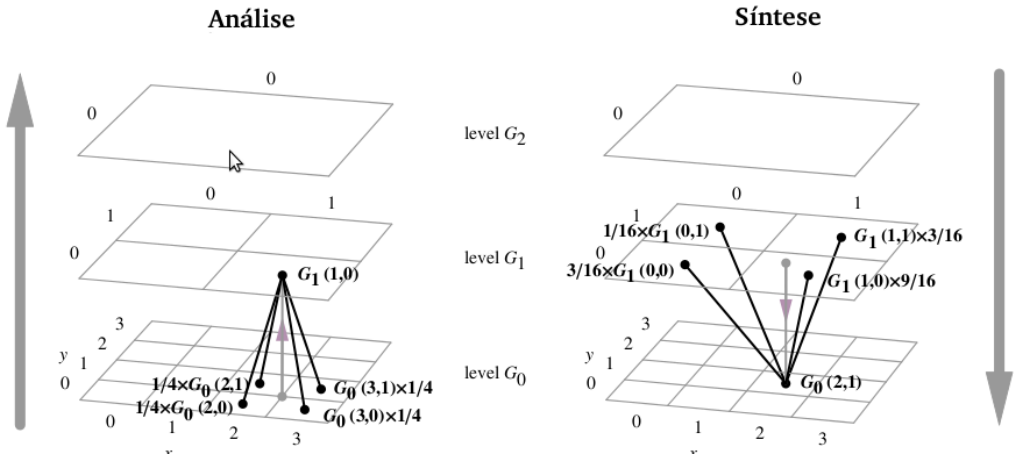

Figura 2.12: Estrutura básica do algoritmo de análise e síntese. Imagem retirada de [9].

Capítulo 3

Método Proposto

O método de interação desenvolvido neste trabalho utiliza as soluções numéricas dos trabalhos de Igarashi et al. [4] e Weng et al. [5] para criar um sistema de deformação baseado no uso de camadas e de ferramentas de tratamento de imagens. O objetivo do uso de camadas é permitir que as deformações propostas pelo usuário não alterem as características de todo o objeto, mas apenas das partições locais as quais o usuário tem a liberdade de selecionar.

Para se criar o sistema proposto, foi necessário passar por todas as etapas que são comuns aos métodos que utilizam restrições posicionais como base para a deformação. A primeira delas é a criação de um polígono que envolve o objeto preservando a sua silhueta. Para esta tarefa foi empregado o algoritmo *marching squares* (veja seção 2.3.1).

Após a obtenção do polígono formado pelos pontos de borda, a próxima etapa é a segmentação do mesmo para que o objeto bidimensional encontre uma estrutura apropriada para os métodos de deformação. Para esta etapa, diversos métodos são propostos e o resultado obtido é uma malha poligonal (triangular, quadrada ou poligonal convexa). O trabalho de Igarashi et al. [4] utiliza uma malha triangular gerada pelo método proposto por Markosian et al. [14], enquanto que o método de Weng et al. [5] obtém uma malha mista (contendo tanto triângulos quanto quadriláteros) utilizando um algoritmo proposto por Zhou et al. [6]. Após o teste de diversas formas de segmentação, optou-se pela utilização do método de triangulação descrito em [20], devido a seu desempenho e à detalhada documentação fornecida. A partir deste momento, o usuário deve escolher quais vértices da malha triangular gerada serão usados como restrições posicionais.

A segmentação é de fundamental importância para o desempenho do sistema de deformação, pois quanto maior o número de segmentos (triângulos neste caso), maior é o tamanho do sistema numérico a ser solucionado durante a movimentação de restrições (ver seções 3.2.6 e 3.2.7). Como o algoritmo de triangulação utiliza os pontos do contorno como vértices para a triangulação, é importante que o conjunto de pontos que representem o contorno não seja muito grande. Assim, é interessante a aplicação de um algoritmo de simplificação de polígonos para redução do número de pontos do polígono sem causar

grandes danos à forma do mesmo. Para este propósito foi utilizado o algoritmo Douglas-Peucker (veja seção 2.3.2), por este conseguir uma boa representação para o polígono de forma rápida (veja figura 3.1).

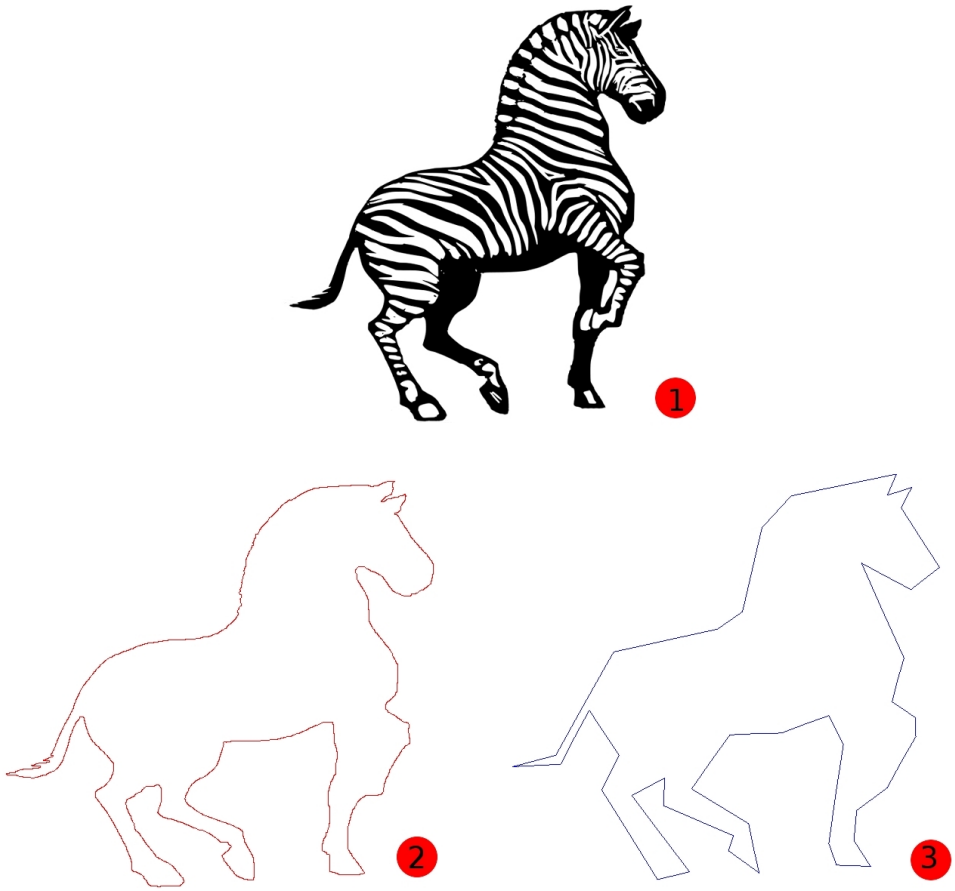

Figura 3.1: Exemplo da aplicação do algoritmo Douglas-Peucker sobre o polígono de borda: 1- Imagem para extração do contorno; 2- Polígono de contorno obtido através do algoritmo *marching squares*; 3- Simplificação do polígono obtida através do algoritmo Douglas-Peucker.

O problema da aplicação de um algoritmo para a simplificação de polígonos, neste caso, é que a nova aproximação para a borda provavelmente não deixará todos os pontos do objeto dentro do polígono, fazendo com que eles sejam desconsiderados no processo de deformação. Assim, uma alternativa seria deslocar os pontos de borda na direção de suas respectivas normais até que o objeto esteja totalmente (ou quase totalmente) inserido no polígono simplificado que o representa (veja figura 3.2). Um resultado similar pode ser alcançado através de um processo que realiza a dilatação na imagem antes da aplicação do algoritmo de simplificação de contorno.

A última etapa do processo é a utilização do método deformador. Para esta etapa, diversos métodos podem ser empregados, porém se optou pela utilização do método de Igarashi et al. [4], em razão dos já conhecidos resultados. A implementação do método proposto por Igarashi et al. [4] foi feita seguindo o roteiro descrito em [21], onde se sugere

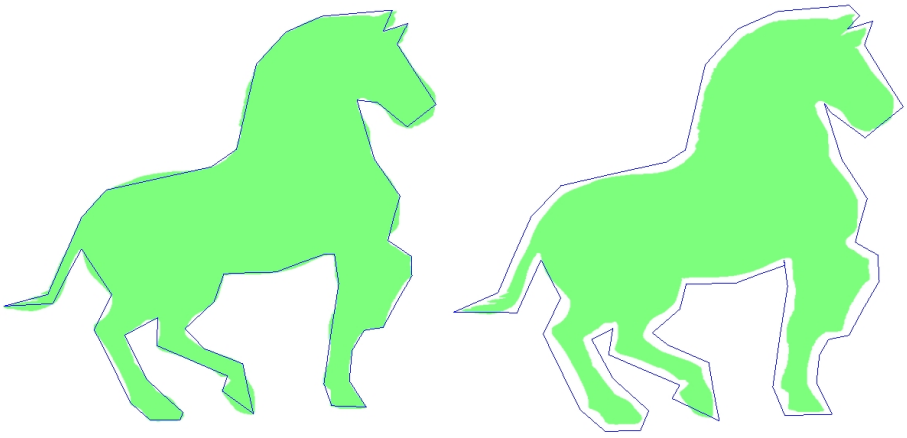

Figura 3.2: Exemplo de deslocamento dos pontos do polígono para que o mesmo englobe uma maior porção do objeto.

a obtenção das coordenadas da malha deformada através da minimização da distorção em cada aresta da malha, ou seja, a diferença entre o tamanho da aresta original e da aresta resultante.

Com esta etapa realizada, já é possível alcançar, de forma interativa, diversas novas configurações para o objeto através da simples movimentação de alguns dos vértices escolhidos como restrições posicionais. Desta forma, os métodos citados conseguem realizar o que se propõem a fazer, ou seja, uma deformação rápida e intuitiva dos objetos através de interações triviais.

3.0.1 Limitações dos métodos de deformação baseados no uso de restrições posicionais

Com a implementação dos métodos baseados no uso de restrições posicionais intuitivas, percebe-se no entanto, algumas limitações para os resultados a serem obtidos. Uma delas ocorre quando a movimentação das restrições resulta em sobreposição de segmentos do mesmo objeto. Isso pode fazer com que a escolha da porção do objeto a ser exibida sobre as outras seja inadequada e muitas vezes até aleatória. Os métodos de Weng et al. [5] e Igarashi et al. [4] utilizam, para este problema, uma ordem estática predefinida para exibição das subpartes do objeto, mas esta estratégia não funciona bem em todos os tipos de objeto, principalmente em áreas de muita sobreposição (ver figura 3.3).

Uma outra limitação para este paradigma de deformação é a falta de uma estrutura que controle as deformações locais no objeto. Muitas vezes se pretende a deformação de todo o objeto ao se mover os pontos de controle, porém algumas vezes o resultado esperado é uma deformação local ou uma deformação que transforme algumas partes mais que as outras (ver figura 3.4). Se nenhuma atenção for dada a este quesito, os métodos baseados na movimentação das restrições posicionais tenderiam a deformar mais as partes com menor

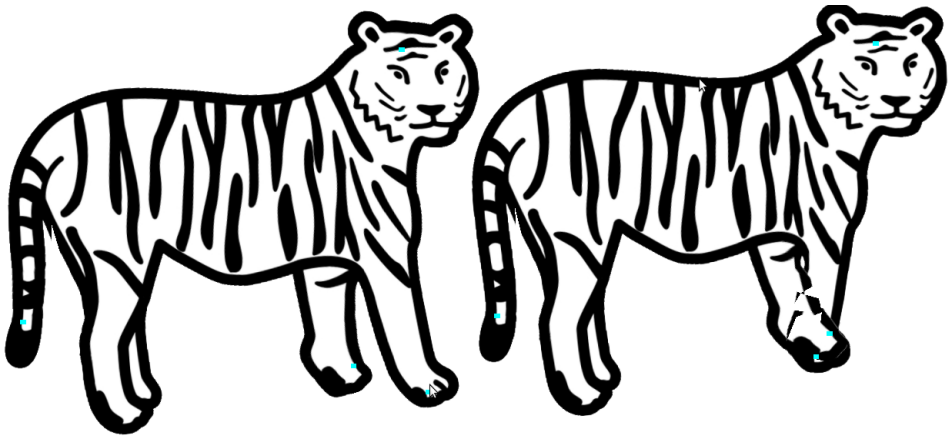

Figura 3.3: Exemplo de deformação na qual a sobreposição de regiões gera resultados inconsistentes.

concentração de pontos do objeto. Em outras palavras, o resultado das transformações em um único ponto é calculado com base em cálculos ponderados que envolvem a posição de seus pontos vizinhos, fazendo com que os pontos com menor número de vizinhos sejam mais suscetíveis às deformações. Assim, para um maior controle das deformações, uma atribuição cuidadosa dos pesos nas diversas partes do objeto seria necessária. Para isto, o trabalho de Igarashi et al. [4] propõe uma interface de pintura para atribuição manual dos pesos em cada vértice. Esta solução porém, torna-se tediosa quando se quer atribuir pesos diferentes a regiões específicas.

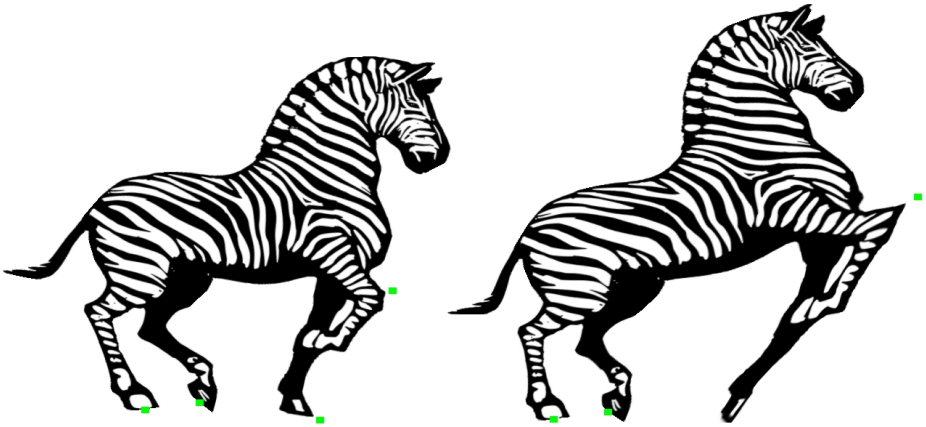

Figura 3.4: Exemplo de deformação que não leva em consideração deformações locais.

3.1 Estrutura de camadas

A solução usada neste trabalho para a diminuição destas limitações foi o emprego de uma estrutura baseada na manipulação de camadas, que representariam diferentes regiões não necessariamente disjuntas do objeto. A estrutura baseia-se na seleção de porções do objeto, fazendo com que as mesmas passem a compartilhar algumas características,

como a rigidez dos pontos, a ordem de exibição dos segmentos, influência dos pontos de controle, entre outras (ver figura 3.5).

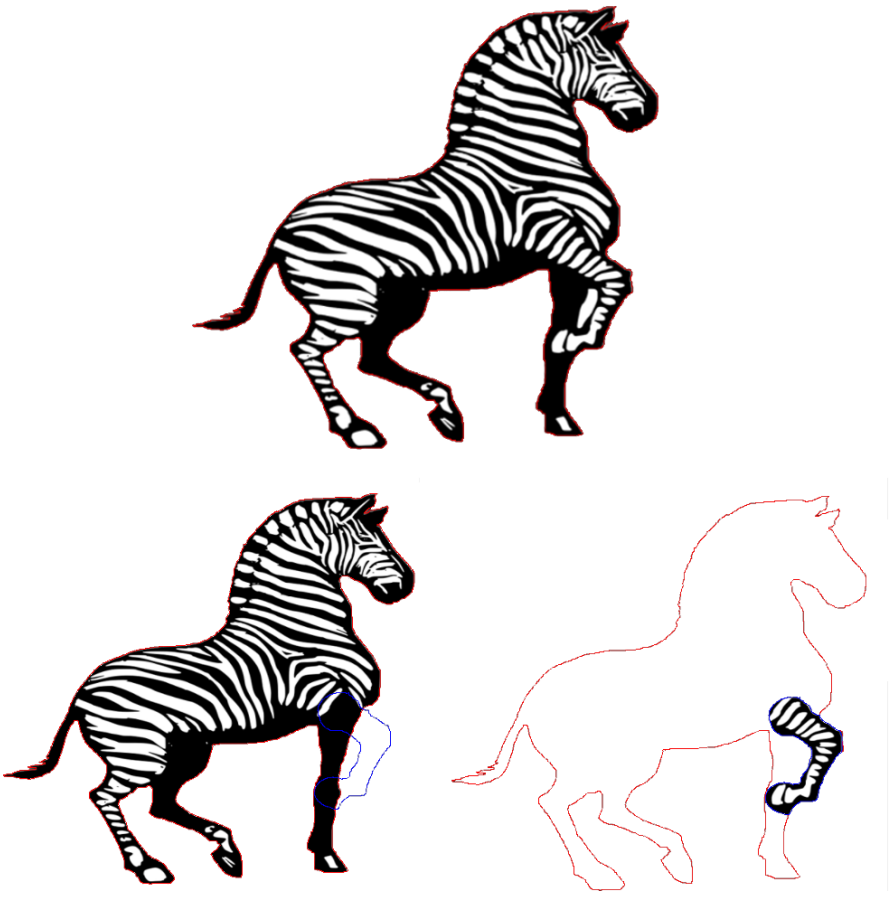

Figura 3.5: Exemplo da utilização de camadas.

A delimitação das partes da imagem que comporão uma dada camada pode ser feita com base em métricas tais como a existência de características em comum entre pontos, a existência de uma separação real no objeto representado ou quaisquer outras necessidades de diferenciação entre pontos.

As camadas são em grande parte independentes, podendo se interligar por meio de regiões definidas pelo usuário. Assim, cada camada passaria por todas as etapas descritas até o momento, desde a criação da silhueta até a aplicação do método deformador. Isso faz com que seja possível a manipulação de forma natural das partes do objeto que possuam alguma sobreposição. Além disso, essa abordagem permite a realização de deformações locais, uma vez que os pontos de controle podem afetar apenas camadas individuais.

Uma opção para a alteração das características do objeto seria a edição de imagens referentes às camadas através da interface para tratamento das mesmas. Para facilitar o processo, foi adicionado à estrutura de camadas um componente de máscara responsável por dar suporte às operações nas camadas, como por exemplo a consulta se um píxel (x, y) faz parte ou não do objeto.

Para o processo de criação das camadas, a interface permite a seleção das regiões através de ferramentas manuais e automáticas de seleção, além de operações básicas de tratamento de imagens.

O próximo passo após a criação das camadas é a definição da ordem de exibição das mesmas. Na criação das camadas, um índice para ordem de exibição é atribuído automaticamente (por ordem de criação) e esse índice pode ser alterado a qualquer momento durante a interação. Após esta definição é possível a criação de uma relação entre as camadas, para que as deformações não sejam apenas locais. Isto é feito antes da etapa de triangulação e por meio da definição de um ou mais segmentos de reta contendo pontos pertencentes à interseção das camadas as quais se quer relacionar.

A estrutura de camadas permite uma deformação mais apropriada para objetos que já possuam sobreposição em sua configuração original (objeto em estado inicial). Isso acontece devido ao fato da porção sobreposta poder ser associada a uma camada, o que torna sua movimentação mais independente. Quando uma camada é criada, cabe ao usuário escolher se a área de interseção entre a camada criada e as demais será excluída, clonada ou editada. Para o caso das regiões em comum serem excluídas, foi usado neste trabalho uma heurística de preenchimento de espaços vazios descrito em [9]. Nela, a porção da imagem perdida é recuperada através da execução de dois processos em pirâmide chamados de análise e síntese (ver seção 2.3.3 e figura 3.6).

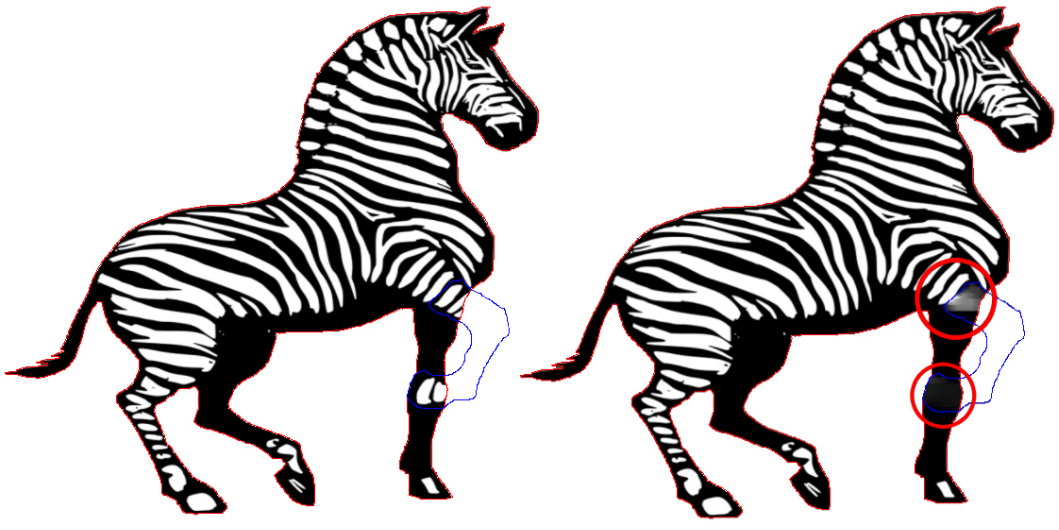

Figura 3.6: Exemplo da utilização do algoritmo da análise e síntese para o preenchimento de regiões de interseção entre camadas.

3.2 Interação

Para a utilização do sistema de deformação proposto, foi criado um roteiro de tarefas a serem realizadas sequencialmente para a obtenção do resultado final:

1. **Carregamento da imagem**
2. **Criação do objeto**
3. **Edição de camadas**
4. **Seleção de Pontos em comum**
5. **Triangulação e seleção de restrições**
6. **Registro de restrições**
7. **Manipulação de restrições**

Este roteiro é mostrado explicitamente como botões na interface da ferramenta, que comandam o modo de interação (ver painel (1) da figura 3.7).

Figura 3.7: Interface proposta: 1- Roteiro de tarefas; 2- Ferramentas de edição e manipulação de camadas; 3- Opções de Visualização; 4- Opções de Segmentação; 5- Preview e ordenação das camadas.

3.2.1 Carregamento da imagem

Nesta etapa o usuário escolhe a imagem que contém o objeto a ser deformado. Após a escolha da imagem, a primeira camada é criada e à ela é associada uma máscara monocromática para identificação de quais píxeis da imagem possuem seus valores acima de um limiar previamente definido (valores variam entre 0 e 255). Nesta etapa é possível a edição da imagem através de ferramentas de tratamento de imagens (como ferramentas de pintura, preenchimento, etc.) assim como a edição da máscara definindo quais píxeis poderiam ou não fazer parte do objeto (ver figura 3.8).

Figura 3.8: Primeira estimativa para a máscara monocromática.

3.2.2 Criação do objeto

Nesta etapa, a máscara criada é utilizada no algoritmo *marching squares* para fornecer os dados de entrada para a definição da silhueta do objeto. A primeira camada é então redefinida e a máscara passa a conter apenas os pontos contidos na borda do objeto (ver figura 3.9). Um algoritmo de rasterização baseado em varredura é aplicado ao polígono e a máscara passa a conter não apenas os pontos de borda, mas também todos os pontos internos ao objeto (ver figura 3.10). Esta máscara é usada para criação da imagem referente à primeira camada, de forma com que apenas os píxeis pertencentes ao objeto sejam copiados da imagem resultante da primeira etapa, deixando os demais transparentes.

3.2.3 Edição de camadas

Esta pode ser definida como a principal etapa na interação com o sistema de deformação criado. Nela, ocorre de fato a manipulação de camadas descrita no escopo deste trabalho (ver figuras 3.11, 3.12, e 3.13). A criação das camadas pode ser feita de forma manual (por meio do desenho de silhuetas ou pintura de regiões) ou automática (por meio de ferramentas de seleção de cores e de seleção automática de contornos por através de variantes do algoritmo *marching squares*). Cada camada é associada à uma imagem e com a utilização da máscara criada na etapa 2, píxeis não pertencentes ao objeto podem ser retirados após a criação de camadas de forma manual. Também é possível a remoção e alteração da ordem de exibição entre as camadas (veja painel (5) da interface apresentada na figura 3.7). Nesta etapa, ferramentas de tratamento de imagens podem ser usadas para

Figura 3.9: Criação do polígono.

Figura 3.10: Rasterização do polígono redefinindo a máscara monocromática.

alterar o aspecto visual das camadas. Todas as camadas passam pelo processo de obtenção de silhueta e de inserção automática de pontos interiores que serão úteis no processo de segmentação das camadas (ver seção 3.2.5) .

3.2.4 Seleção de Pontos em comum

O objetivo desta etapa é relacionar as camadas entre si. Para isso é preciso identificar ou criar regiões em comum entre as mesmas. A primeira tarefa a ser realizada é a escolha das

Figura 3.11: Delimitação da região para criação da camada.

Figura 3.12: Edição da região por meio de operações de tratamento de imagens.

camadas que se quer relacionar. Após a seleção, um segmento de reta é desenhado na área de interseção das camadas. Após a etapa de triangulação, este segmento será convertido numa aresta pertencente às camadas relacionadas (ver figura 3.14). Desta forma, os pontos desta aresta poderão ser tratados como restrições posicionais, tornando possível que a deformação de uma camada influencie na deformação das demais. Também é possível a desassociação entre as camadas, tornando sua movimentação e deformação das mesmas independentes.

Figura 3.13: Exemplo de exibição das camadas pertencentes a um objeto (selecionadas manualmente).

Figura 3.14: Exemplo de relacionamento entre duas camadas por meio de um segmento de reta em comum.

3.2.5 Triangulação e seleção de restrições

O objetivo desta etapa é realizar a segmentação descrita no capítulo anterior. Neste momento, as camadas passam pelo processo de segmentação, contendo como restrições iniciais as definidas na etapa anterior. Como cada camada passa por um processo de segmentação próprio, é preciso que o usuário escolha em cada camada quais serão os vértices da triangulação que serão tratados como restrições posicionais (ver figura 3.15). É possível também que o usuário escolha um ponto do objeto e o sistema encontre a camada mais próxima para adição da restrição. Caso este ponto não seja um vértice pertencente a nenhuma triangulação, pode-se então adicioná-lo como vértice à camada mais próxima, sendo que esta deverá ser retriangulada neste processo.

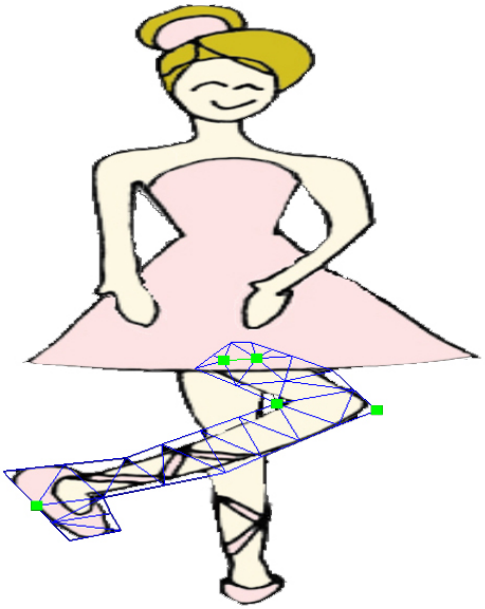

Figura 3.15: Exemplo de segmentação e seleção de restrições em uma camada.

3.2.6 Registro de restrições

Esta é a última etapa antes de ser possível a realização da deformação por meio da movimentação de restrições. Nela acontecem as pré-computações descritas no trabalho de Igarashi et al. [4]. O objetivo desta etapa é eliminar o maior número possível de cálculos a serem realizados durante a movimentação de restrições. A maioria destas pré-computações está relacionada ao cálculo de valores relacionados ao objeto em estado de repouso, como o tamanho das arestas, relação entre vértices vizinhos, razão entre triângulos, etc.

3.2.7 Manipulação de restrições

Nesta etapa cabe ao usuário apenas a movimentação dos pontos de controle para realizar as deformações desejadas (ver figura 3.16). A seleção das restrições para a movimentação pode ser feita de maneira manual (selecionando a camada e a restrição) ou de maneira automática. No último caso, o sistema retorna a restrição e a camada mais próxima a partir de um ponto selecionado pelo usuário.

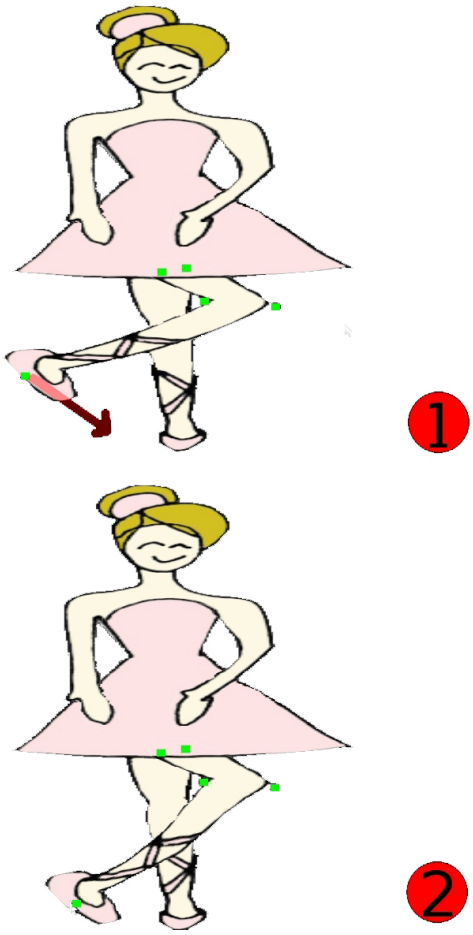

Figura 3.16: Manipulação de restrições: 1- Posição 1; 2- Posição 2.

Algumas estratégias também podem ser adotadas para realizar as alterações mais comuns ao objeto. Por exemplo, é possível movimentar todo o objeto adicionando um vetor de descolamento a todas as restrições posicionais das camadas. A alteração nas medidas do objeto também pode ser alcançada através do uso de uma função linear de deslocamento que ajuste a posição de todas as restrições ao ser escolhido um novo valor para a largura ou comprimento do objeto.

Capítulo 4

Implementação e resultados

A validação das ideias vistas nos capítulos anteriores se deu através do desenvolvimento de um software que implementa um sistema de deformação de imagens baseado no uso de restrições posicionais.

Neste capítulo serão apresentados alguns detalhes que foram importantes na criação deste software. Além disso, será feita uma discussão sobre os resultados obtidos pelo mesmo através de testes e exemplos.

4.1 Detalhes de implementação

A implementação deste projeto foi feita utilizando a linguagem de programação C++ juntamente com o paradigma de orientação a objetos. Para a criação da interface e para a manipulação de imagens foi utilizada como biblioteca auxiliar o QT. Para a realização do processo de triangulação e para utilização de uma estrutura de dados que represente a mesma, foi utilizada como biblioteca auxiliar o CGAL [20].

A parte mais crítica do projeto foi a resolução dos sistemas numéricos que encontram uma nova configuração para o objeto durante a interação. Como os sistemas de otimização vistos em [21] podem ser representadas através de matrizes esparsas, a biblioteca deveria dar suporte a este tipo de representação, para que as operações com estas matrizes não implicassem em um tempo de resposta considerável. Algumas bibliotecas foram testadas com este propósito (Taucs [22], Lapack [23] e Eigen [24]) , porém foi escolhida a Eigen [24], devido à sua eficiência e simplicidade de instalação.

Para a implementação do conceito de camadas foi criada uma estrutura de dados, que é responsável por armazenar os elementos relacionados às mesmas. Entre os elementos associados a uma camada estão a imagem referente à mesma, a máscara utilizada para operações sobre a camada, o polígono de borda original, o polígono de borda simplificado, a estrutura de dados que representa a triangulação e todos os elementos auxiliares que serão importantes para a realização da deformação, como os pontos de controle (em

repouso ou transladados), os pontos extremos, vértices em comum com outras camadas, etc.(veja figura 4.1).

A implementação foi possível através da criação de um modelo orientado a objetos contendo basicamente quatro módulos juntamente com a estrutura de dados que representa as camadas (veja figura 4.1):

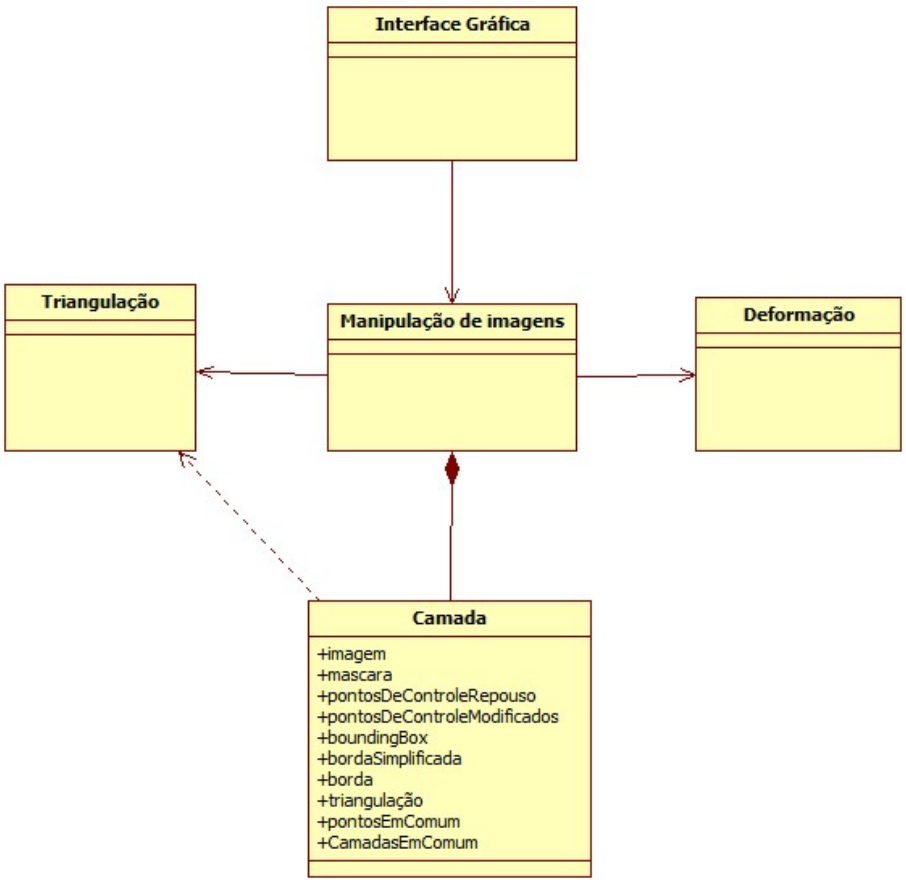

Figura 4.1: Diagrama de classes do sistema de deformação.

O módulo de manipulação de imagens é o principal componente deste sistema, cabendo a ele a responsabilidade por basicamente todas as operações de manipulação de imagens e objetos, como o carregamento da imagem, criação do objeto, criação e alteração das camadas, etc.

O módulo de triangulação é responsável por utilizar uma estrutura de dados adequada para a representação da mesma, além de realizar todas as operações relativas à segmentação.

O módulo de deformação é responsável por gerar uma nova configuração para o objeto através da utilização do método de deformação escolhido. Utilizamos o método de Igarashi et al. [4], mas existe o interesse da utilização de outros métodos (como o método não linear descrito em [5]) para a realização da mesma tarefa.

O módulo de Interface Gráfica é responsável por permitir a realização das etapas de interação descritas neste trabalho a partir da utilização de botões, janelas gráficas, etc.

4.2 Resultados e testes

O sistema foi testado em um computador com 2GB de memória RAM e um processador Intel Core™2 Quad Q8300 2.5GHz. O sistema operacional utilizado foi o Ubuntu 9.10 e o programa foi compilado com o gcc.

O sistema foi testado com a utilização de diversos desenhos e foi visto na prática que um dos parâmetros que podem fazer com que o sistema deixe de trabalhar com taxas interativas é a quantidade de triângulos existentes na malha do objeto (em todas as camadas). Quanto mais refinada a malha, mais lenta será a convergência do sistema. Percebemos que se consegue obter uma taxa de resposta interativa quando o número de triângulos é menor que 300 (ver tabela 4.1).

Tabela 4.1: Tabela de desempenho do algoritmo

Modelo	N° de camadas	N° total de vértices	Tempo de uma iteração (milissegundos)
a (figura 4.2)	1	231	130
b (figura 4.2)	2	254	135
c (figura 4.2)	2	451	2013

A estrutura de camadas permite um controle mais preciso sobre os objetos. Isso também se deve ao fato de ser possível se associar malhas triangulares com níveis de refinamento diferentes às camadas criadas (ver figura). Assim, as regiões mais finas, que são geralmente mais suscetíveis às deformações, podem realizar uma deformação mais rígida, o que não é comum em uma abordagem sem camadas (ver figura 4.3).

Após a seleção de camadas o usuário pode escolher como deseja editar as regiões onde ocorrem sobreposição de camadas. Como a edição manual pode se tornar um trabalho tedioso, o algoritmo de análise e síntese (veja 2.3.3) pode ser empregado para o preenchimento automático destas regiões. Este algoritmo, no entanto, funciona melhor para a recuperação de pequenas regiões espalhadas pela imagem, não obtendo bons resultados quando a região a se preencher for muito grande (ver figura 4.4).

Com o estudo das demais abordagens de deformação, percebemos que nenhum método é suficientemente genérico a ponto de realizar deformações que respeitem as características de movimentação de todos os tipos de objetos. Nosso método, por exemplo, é bastante interessante para a deformação de objetos de natureza genérica (ou que não possuam um padrão de movimentação definido), porém não é aconselhado para uma deformação fisicamente precisa de objetos .

O método de deformação escolhido neste trabalho [4] se mostrou capaz de deformar os objetos mais variados de forma satisfatória. Porém, uma das características de

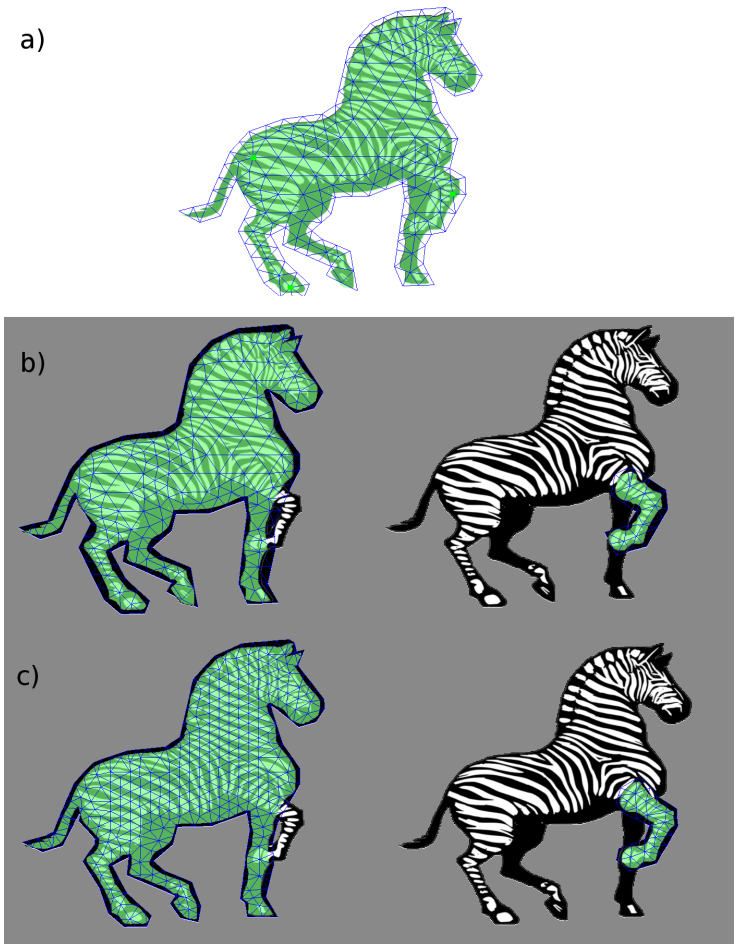

Figura 4.2: Exemplos usados no teste de desempenho do método descrito em [4].

deformação percebidas é que, devido à sua natureza linear, suas deformações tendem a manter a curvatura original dos objetos, como podemos ver na figura 4.5.

O relacionamento entre as camadas também é um fator a ser levado em consideração. Quando as camadas são relacionadas entre si (ver figura 4.6), o processo de deformação é mais prático e rápido, porém em alguns casos pode ser necessário um ajuste na posição das restrições em comum para alcançar uma melhor deformação para o objeto. Camadas desassociadas (ver figuras 4.7 e 4.8) tendem deixar o processo de deformação mais livre, porém pode deixa-lo mais tedioso.

O número de camadas é outro fator que pode contribuir para a complexidade da deformação, pois quanto maior o número de camadas, maior será o número de possíveis interações necessárias por parte do usuário. Foram feitos testes com até oito camadas, e um exemplo de deformação através da criação de quatro camadas é demonstrado na figura 4.9.

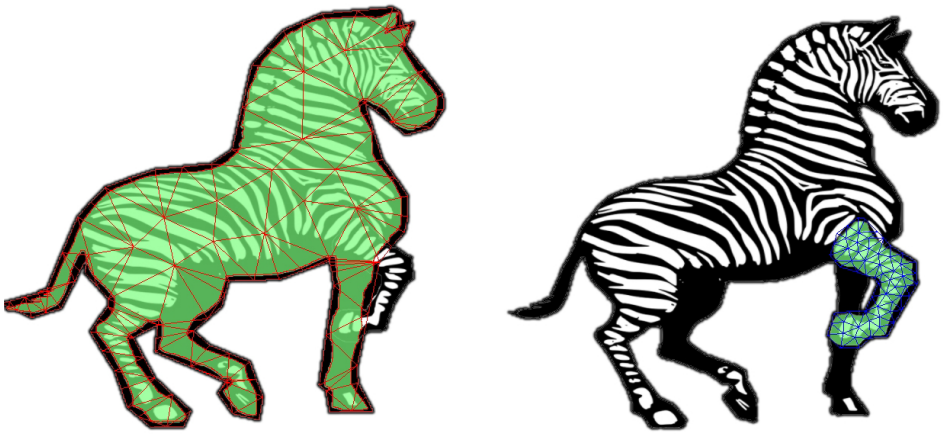

Figura 4.3: Exemplo de camadas com níveis de refinamento de malha diferentes.

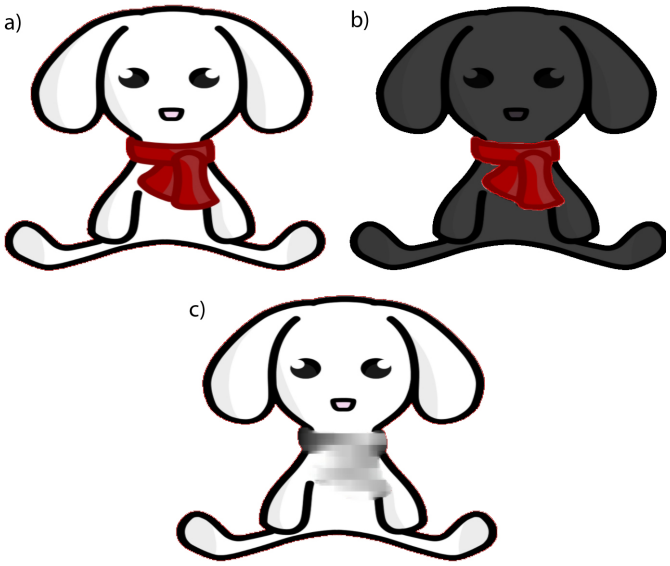

Figura 4.4: Exemplo do emprego do algoritmo de análise e síntese em regiões não muito pequenas.

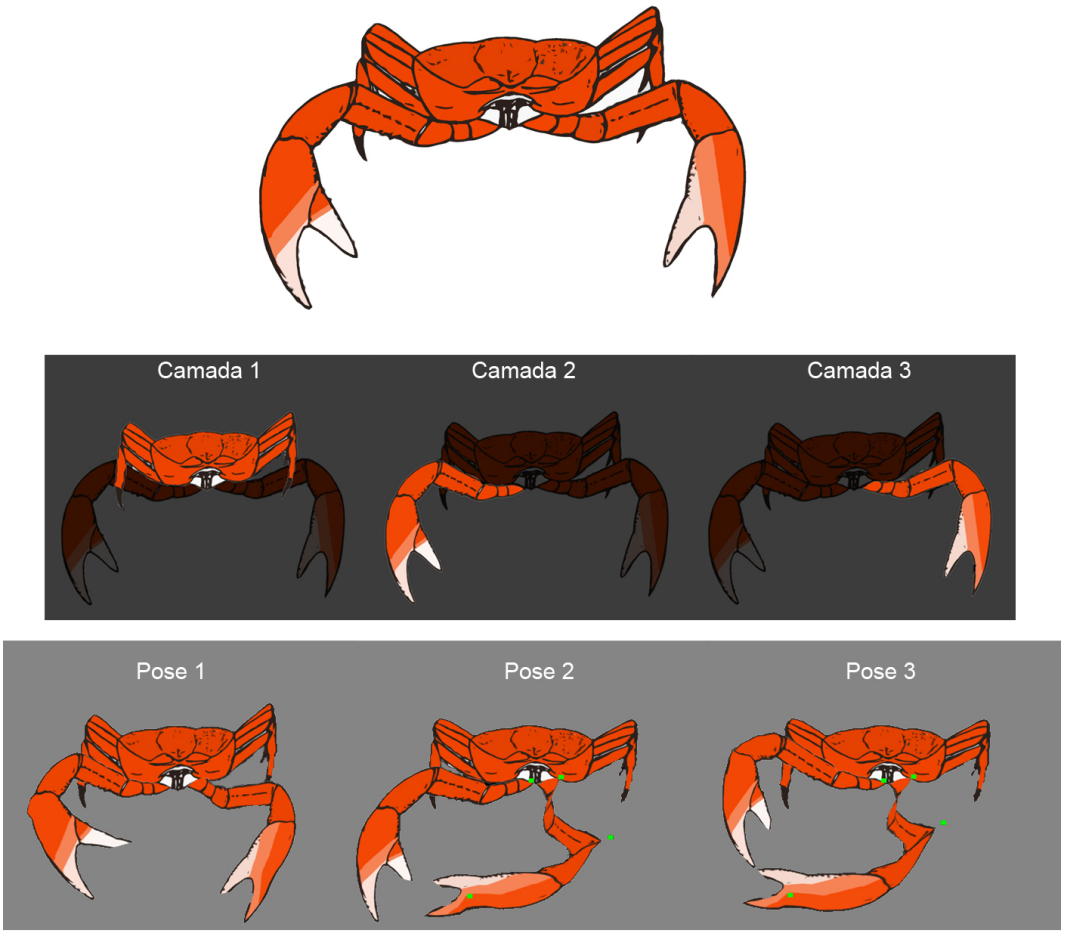

Figura 4.5: Exemplo de deformação não satisfatória utilizando três camadas.

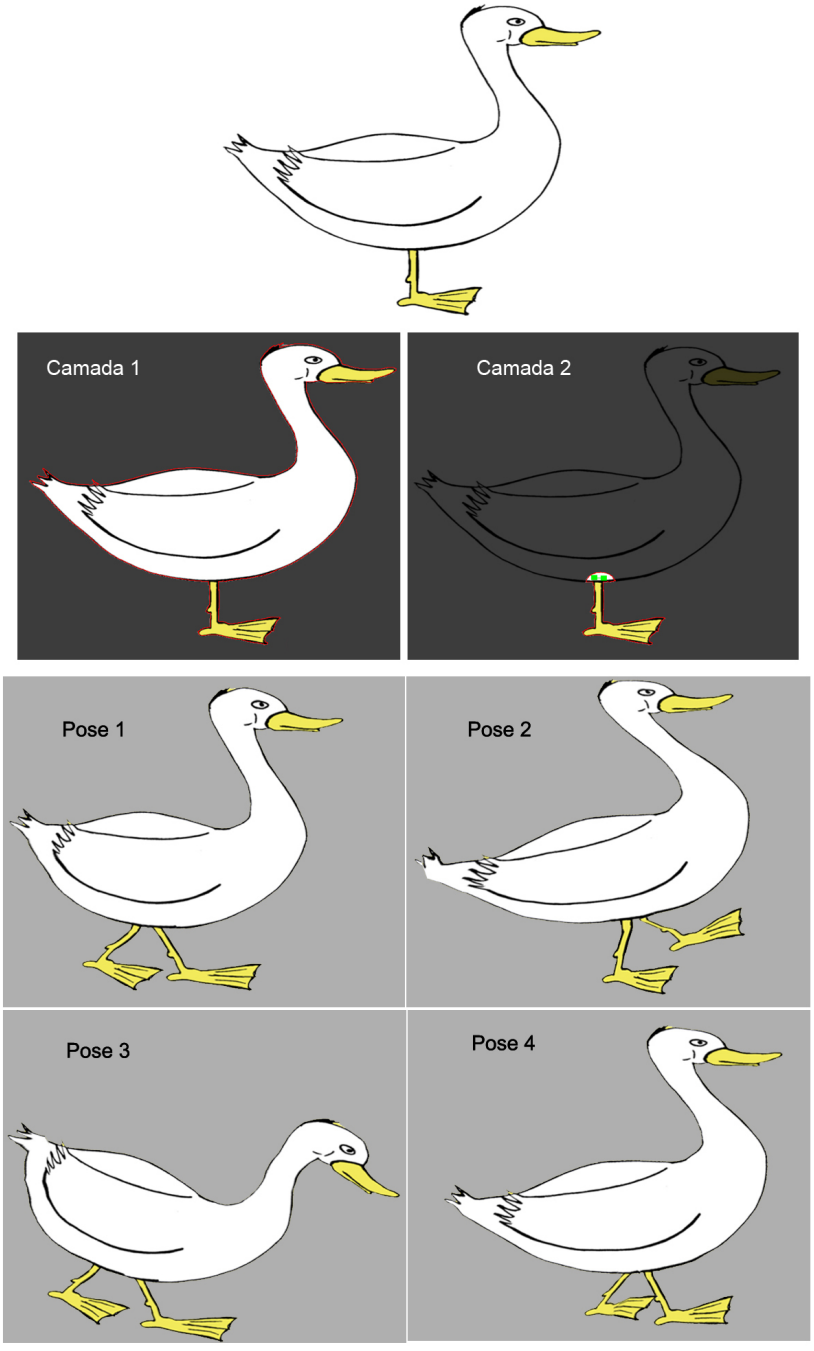

Figura 4.6: Exemplo de deformação utilizando duas camadas relacionadas.

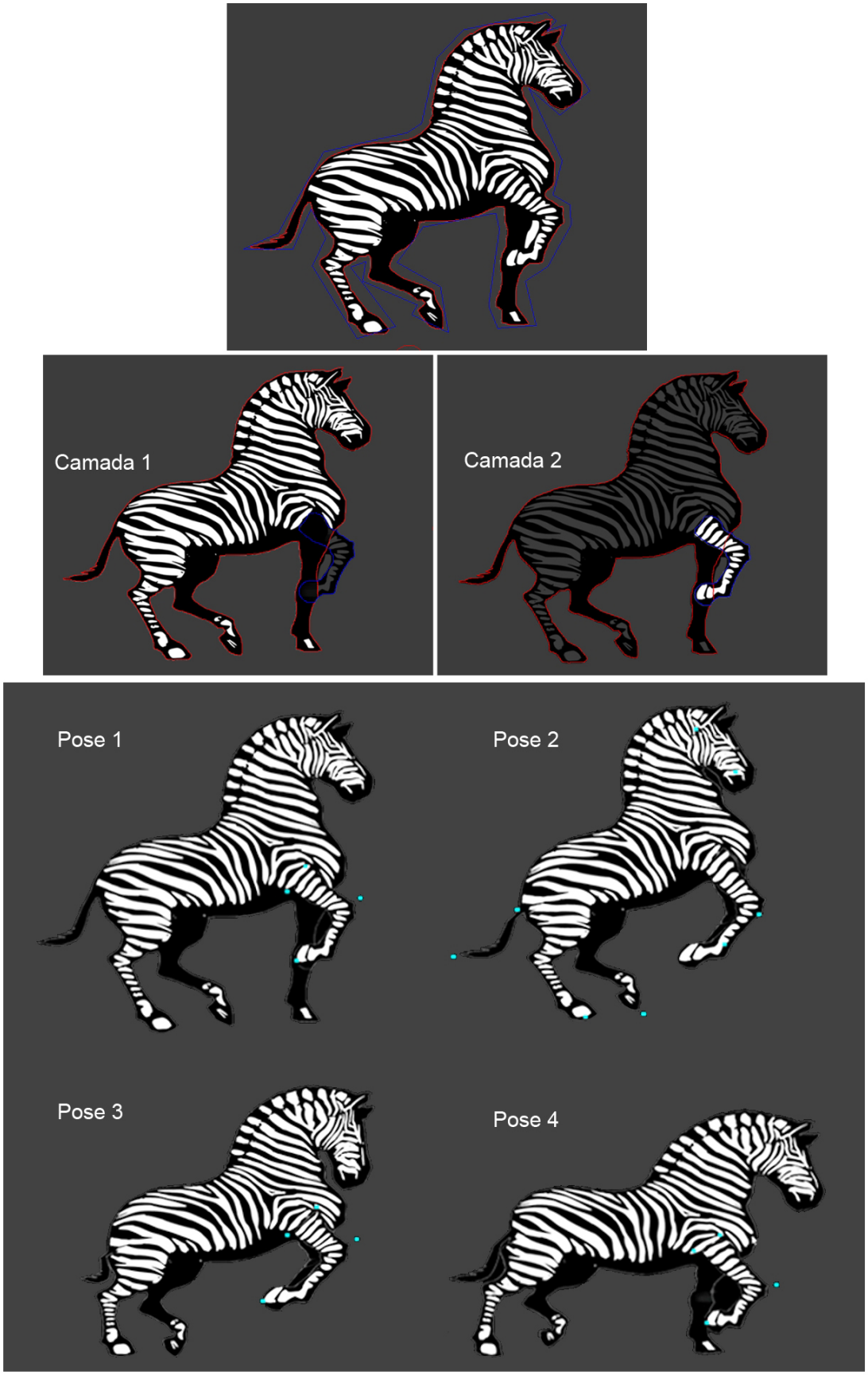

Figura 4.7: Exemplo de deformação utilizando duas camadas desassociadas.

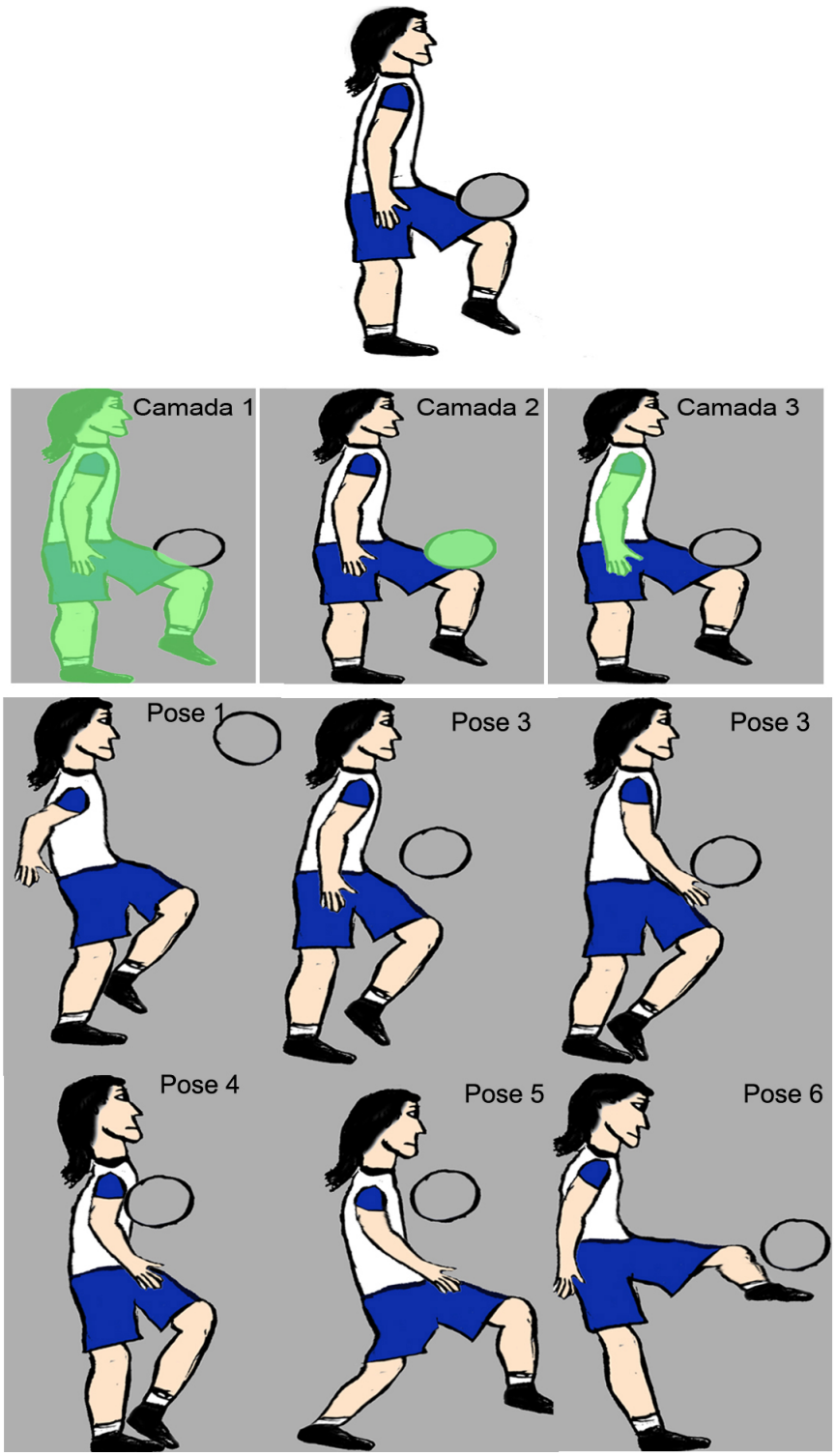

Figura 4.8: Exemplo de deformação utilizando três camadas desassociadas.

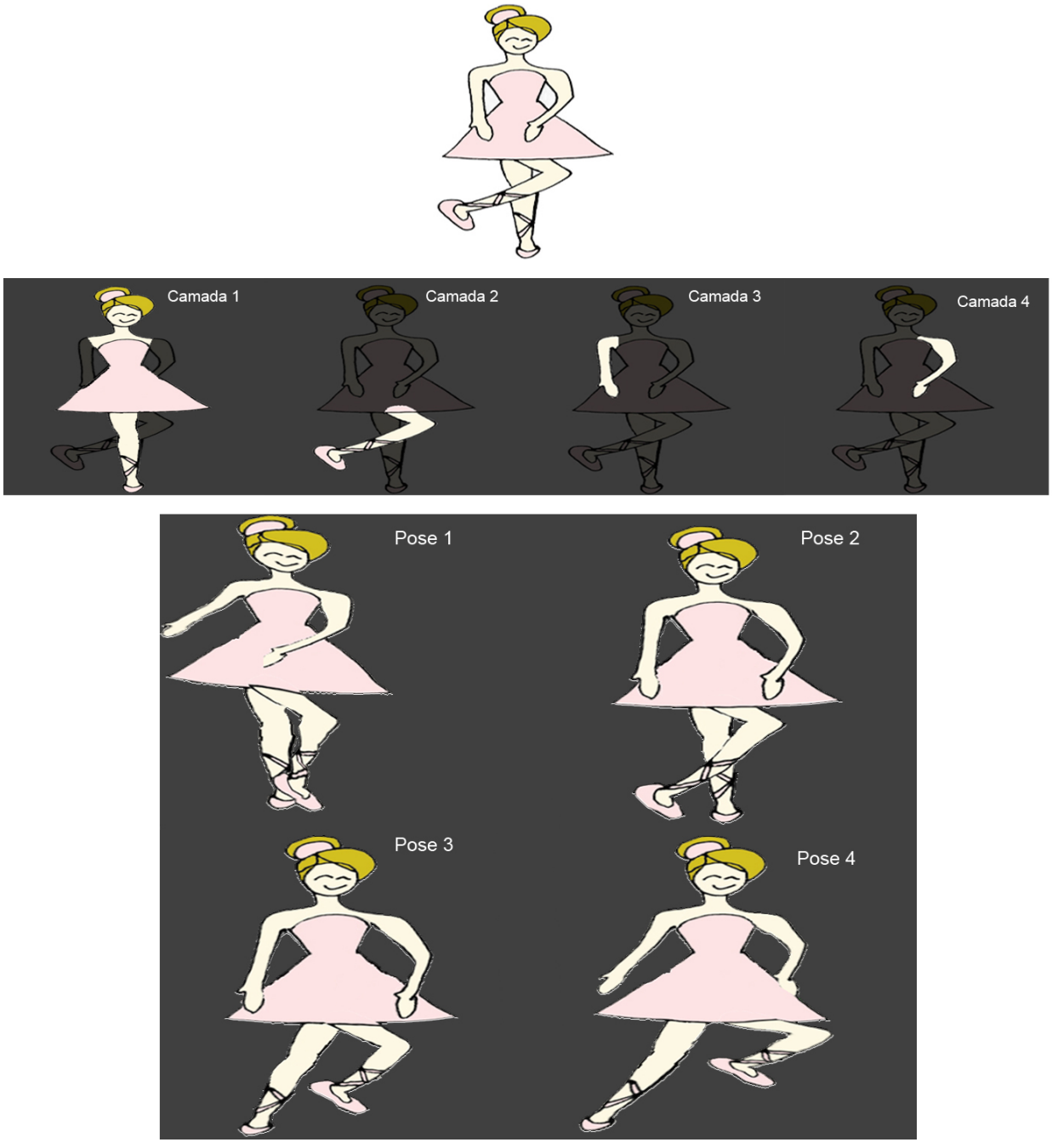

Figura 4.9: Exemplo de deformação com um número maior de camadas (quatro).

Capítulo 5

Conclusão

Neste trabalho foi feito um estudo dos métodos que se baseiam no paradigma de deformação de imagens a partir do uso de restrições posicionais. Nossa contribuição para os trabalhos desta abordagem foi a utilização de uma estrutura de camadas que define uma representação diferente para o objeto e permite uma maior liberdade em sua deformação.

As duas maiores influências para este trabalho são os métodos desenvolvidos por Igarashi et al. [4] e Weng et al. [5]. Do primeiro, as maiores influências foram os princípios de manipulação de imagens descritos e a implementação das duas etapas lineares apresentadas para a geração uma nova configuração ao objeto. Do segundo, a maior influência foi a preocupação com as áreas em sobreposição da imagem.

A preocupação com a interface é um ponto crítico do sistema de deformação, pois se deseja obter resultados rapidamente a partir de operações simples. Desta forma, é necessário se ter uma preocupação com o número de funcionalidades agregadas à interface, para que a mesma não apresente uma sobrecarga de itens, tornando as operações complexas e não intuitivas. Assim, este trabalho ainda pode ser melhorado com a utilização de um número menor de funcionalidades e um número maior de processos automatizados.

Percebemos que a utilização de camadas é uma opção viável para agregar mais possibilidades à deformação de imagens por meio de restrições posicionais intuitivas sem afetar consideravelmente o desempenho. Além disso, o usuário muitas vezes já é familiarizado com o uso de ferramentas de transformação de imagens, fazendo com que o tempo aprendizado da ferramenta de deformação não seja demorado.

Percebe-se também a possibilidade de expansão deste trabalho através da utilização de novos métodos de recuperação de imagens não só para o preenchimento de regiões em sobreposição, mas também para a substituição automática de cores de uma determinada camada.

Outro incremento ao trabalho desenvolvido seria a adição de novos métodos de deformação que respeitem o paradigma mencionado neste trabalho. Isso permitirá com que o usuário tenha um controle ainda maior sobre o tipo de deformação que se deseja obter.

Seria interessante, também, a utilização de uma estrutura de camadas em métodos baseados em outros paradigmas de deformação, como por exemplo, nos métodos de deformação que fazem o uso de simulações baseadas em física [3], deixando o trabalho com deformação de objetos com um número maior de funcionalidades.

Referências Bibliográficas

[1] MACCRACKEN, R., JOY, K. I. "Free-form deformations with lattices of arbitrary topology". In: *Proceedings of SIGGRAPH '96*, pp. 181–188, New York, NY, USA, 1996.

[2] YAN, H.-B., HU, S., MARTIN, R. R., et al. "Shape Deformation Using a Skeleton to Drive Simplex Transformations", *IEEE Transactions on Visualization and Computer Graphics*, v. 14, pp. 693–706, 2008.

[3] CELNIKER, G., GOSSARD, D. "Deformable curve and surface finite-elements for free-form shape design", *Proceedings of SIGGRAPH '91*, v. 25, pp. 257–266, 1991.

[4] IGARASHI, T., MOSCOVICH, T., HUGHES, J. F. "As-rigid-as-possible shape manipulation". In: *Proceedings of SIGGRAPH '05*, pp. 1134–1141, New York, NY, USA, 2005.

[5] WENG, Y., XU, W., WU, Y., et al. "2D shape deformation using nonlinear least squares optimization", *Vis. Comput.*, v. 22, n. 9, pp. 653–660, 2006.

[6] ZHOU, K., HUANG, J., SNYDER, J., et al. "Large mesh deformation using the volumetric graph Laplacian". In: *Proceedings of SIGGRAPH '05*, pp. 496–503, New York, NY, USA, 2005.

[7] FLOATER, M. S. "Mean value coordinates", *Computer Aided Geometric Design*, v. 20, pp. 2003, 2003.

[8] WHYATT, J. D., WADE, P. R. "The Douglas-Peucker Line Simplification Algorithm", *Society of University Cartographers Bulletin*, v. 22, pp. 17 – 25, 1988.

[9] STRENGERT, M., KRAUS, M., ERTL, T. "Abstract Pyramid Methods in GPU-Based Image Processing". , 2008.

[10] THOMAS, B. H., CALDER, P. "Animating direct manipulation interfaces". In: *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pp. 3–12, New York, NY, USA, 1995.

[11] SÝKORA, D., DINGLIANA, J., COLLINS, S. "As-rigid-as-possible image registration for hand-drawn cartoon animations". In: *NPAR '09: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pp. 25–33, New York, NY, USA, 2009.

[12] NGO, T., CUTRELL, D., DANA, J., et al. "Accessible animation and customizable graphics via simplicial configuration modeling". In: *Proceedings of SIGGRAPH '00*, pp. 403–410, New York, NY, USA, 2000.

[13] WANG, Y., XU, K., XIONG, Y., et al. "2D shape deformation based on rigid square matching", *Comput. Animat. Virtual Worlds*, v. 19, n. 3-4, pp. 411–420, 2008.

[14] MARKOSIAN, L., COHEN, J. M., CRULLI, T., et al. "Skin: A Constructive Approach to Modeling Free-form Shapes". In: *Proceedings of SIGGRAPH '99*, pp. 393–400, 1999.

[15] SORKINE, O., COHEN-OR, D., LIPMAN, Y., et al. "Laplacian surface editing". In: *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 175–184, New York, NY, USA, 2004.

[16] LORENSEN, W. E., CLINE, H. E. "Marching cubes: A high resolution 3D surface construction algorithm", *Proceedings of SIGGRAPH '87*, v. 21, pp. 163–169, 1987.

[17] RAMER, U. "An iterative procedure for the polygonal approximation of plane curves", *Computer Graphics and Image Processing*, v. 1, n. 3, pp. 244 – 256, 1972.

[18] DOUGLAS, D. H., PEUCKER, T. K. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", *Cartographica: The International Journal for Geographic Information and Geovisualization*, v. 10, n. 2, pp. 112–122, out. 1973.

[19] MCMASTER, R. B. "Automated line generalization", *Cartographica*, v. 24, pp. 74–111, 1987.

[20] HEMMER, M. *Polynomials, CGAL - Computational Geometry Algorithms Library, release 3.4*. CGAL, Campus E1 4, 66123 Saarbrücken, Germany, January 2009.

[21] IGARASHI, T., IGARASHI, Y. "Implementing As-Rigid-As-Possible Shape Manipulation and Surface Flattening", *journal of graphics, gpu, and game tools*, v. 14, n. 1, pp. 17–30, 2009.

[22] TOLEDO, S. “Taucs: A Library of Sparse Linear Solvers”, 2003.

[23] ANDERSON, E., BAI, Z., DONGARRA, J., et al. “LAPACK: a portable linear algebra library for high-performance computers”. In: *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, Supercomputing ’90, pp. 2–11, 1990.

[24] GUENNEBAUD, G., JACOB, B., OTHERS. “Eigen v3”. . http://eigen.tuxfamily.org [Last accessed: 10/07/2011].