COPPE/UFRJ

MORPHING EMPREGANDO MODELOS DE DIFUSÃO DE FUMAÇA

Marcelo Caniato Renhe

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: Antônio Alberto Fernandes de Oliveira
Cláudio Esperança

Rio de Janeiro
Junho de 2010

MORPHING EMPREGANDO MODELOS DE DIFUSÃO DE FUMAÇA

Marcelo Caniato Renhe

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. Antônio Alberto Fernandes de Oliveira, D.Sc.

Prof. Cláudio Esperança, D.Sc.

Prof. Ricardo Marroquim, D.Sc.

Prof. Marcelo Bernardes Vieira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2010

Renhe, Marcelo Caniato

Morphing empregando modelos de difusão de fumaça/Marcelo Caniato Renhe. – Rio de Janeiro: UFRJ/COPPE, 2010.

XI, 64 p.: il.; 29, 7cm.

Orientadores: Antônio Alberto Fernandes de Oliveira
Cláudio Esperança

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 62 – 64.

1. Animação física. 2. Métodos baseados em grade. 3. Dinâmica de fluidos. 4. Simulação de fumaça. I. Oliveira, Antônio Alberto Fernandes de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Agradeço a todos aqueles que contribuíram de uma maneira ou de outra para que eu pudesse chegar até aqui. Agradeço à minha família pela educação e pelos princípios ensinados. Aos meus professores que, cada um à sua maneira, buscou da melhor forma contribuir para a minha formação, seja nos ensinos fundamental e médio, na graduação ou no mestrado. Deixo aqui registrado também minha eterna gratidão a todos os meus amigos que me apoiaram em todos os momentos nas minhas decisões: os antigos e os novos amigos feitos no Rio de Janeiro. Não citarei nomes pois são muitos e cada um deles sabe a importância que tem. Sou grato também pela oportunidade de ter partilhado da companhia de todos os companheiros de república, que assim como eu passaram pela experiência de ter que se adaptar a uma nova cidade, até então desconhecida para nós. Por fim, mas não menos importante, agradeço a Deus por tudo que me foi proporcionado nestes últimos anos, tendo a certeza de que Ele continua e sempre continuará me dando forças e inspiração para tomar as decisões corretas e saber melhor aproveitar as oportunidades que a vida nos apresenta.

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

MORPHING EMPREGANDO MODELOS DE DIFUSÃO DE FUMAÇA

Marcelo Caniato Renhe

Junho/2010

Orientadores: Antônio Alberto Fernandes de Oliveira
Cláudio Esperança

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma técnica para implementação de *morphing* de objetos utilizando modelos matemáticos empregados na animação de fluidos, especificamente a fumaça. É utilizada uma abordagem euleriana para a representação computacional do fluido, discretizando e calculando as suas propriedades através de métodos bem estabelecidos de difusão e advecção. Partindo dos métodos de simulação propostos por Jos Stam em [1], apresentamos aqui uma variação do trabalho de Fattal e Lischinski [2], utilizando a transformada da distância para o cálculo da força de direcionamento do processo de *morphing*.

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

MORPHING USING SMOKE DIFFUSION MODELS

Marcelo Caniato Renhe

June/2010

Advisors: Antônio Alberto Fernandes de Oliveira
Cláudio Esperança

Department: Systems Engineering and Computer Science

This work presents a technique for implementation of object morphing using mathematical models employed in fluid animation, specifically smoke. We use an Eulerian approach for the computational representation of the fluid, discretizing and calculating its properties through well established methods of diffusion and advection. Starting from the simulation methods proposed by Jos Stam in [1], we present here a variation of the work by Fattal and Lischinski [2], using the distance transform for the calculation of the morphing driving force.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma das áreas da computação gráfica que tem recebido grande atenção nos últimos anos é a animação baseada em física e, mais recentemente, a animação de fluidos de maneira específica. Simulações de processos físicos já vinham sendo pesquisadas há muitos anos, porém, até esta década, os parâmetros eram muito restritos e as simulações relativamente simples, devido à ausência de tecnologia de *hardware* capaz de suprir o poder de processamento necessário por esse tipo de aplicação, principalmente no caso dos fluidos, que normalmente envolve a solução de sistemas lineares, equações diferenciais e mecanismos de otimização de alto custo computacional. Nos últimos anos, um grande número de trabalhos envolvendo simulação de fluidos tem sido publicado, obtendo, muitas vezes executando em tempo real, animações de alta qualidade, tanto em termos de realismo físico quanto de realismo visual. Boa parte dos trabalhos se apóiam em técnicas e metodologias que vêm sendo estudadas já há muito tempo pela Dinâmica dos Fluidos Computacional, que é uma área da computação destinada a estudar formas de representar e simular computacionalmente a física do movimento dos fluidos.

Um outro ponto bastante interessante dentro da computação gráfica e do processamento de imagens é a técnica conhecida como *morphing*. Esta técnica consiste em realizar uma transformação entre duas imagens previamente selecionadas, de maneira suave e natural, mesclando características das duas imagens durante o processo de transformação. Trata-se de uma técnica comumente utilizada em efeitos especiais no cinema e em animações.

Visando integrar estes dois tópicos citados acima, a animação de fluidos e o *morphing*, o objetivo principal deste trabalho é estudar metodologias para realizar o *morphing* de imagens através de um processo de simulação física da dinâmica de uma fumaça. A idéia central é, partindo de uma imagem inicial de entrada, realizar uma transformação desta imagem em uma outra imagem final, de maneira que a transição se assemelhe ao movimento de um fluido. Ambas as imagens são representadas por concentrações de densidade presentes nas posições correspondentes às dos seus pixels.

Essa concentração de densidade é, então, submetida aos processos de simulação de fluidos, sendo posta em movimento por forças específicas que possuem o intuito de guiar toda a massa do fluido até a configuração final, correspondente aos pixels da imagem-alvo. Uma definição mais detalhada e clara do problema é apresentada no capítulo 4.

Mais especificamente, este trabalho se propõe a apresentar soluções para questões que normalmente são problemáticas quando se emprega uma metodologia de difusão para aplicações de *morphing*, muito pelo fato de o resultado da simulação de fluidos ser de difícil previsibilidade. Uma série de parâmetros estão envolvidos no decorrer do processo, muitos dos quais não temos um controle explícito, dificultando tarefas mais específicas, como a necessidade de se manter uma uniformidade na evolução do fluido, geração de buracos e determinadas formas em imagens com várias componentes conexas, entre outros problemas que serão apresentados e discutidos no decorrer do capítulo 4. Um ponto importante que o trabalho trata é a utilização de uma função distância para o cálculo da força que direciona a massa de fumaça até o seu alvo, ao contrário de trabalhos anteriores, que utilizavam outras abordagens. Em especial, Fattal e Lischinski em [2] utilizam uma função gaussiana para o cálculo da força de direcionamento. Essa abordagem, porém, se mostra ineficiente para os casos em que o objeto inicial está espacialmente afastado do objeto alvo. Esse problema é significativamente reduzido com a utilização da função distância, satisfazendo uma maior gama de configurações de entrada para o *morphing*. Essas questões serão analisadas com maior profundidade no capítulo 4.

Este trabalho visa obter uma animação realista o suficiente para caracterizar o movimento da fumaça como um movimento coerente com a dinâmica de fluidos, porém não com a precisão necessária de uma simulação. Por isso o uso do termo animação para descrever o processo como um todo. Embora utilizemos um simulador baseado nas equações que regem a dinâmica dos fluidos, uma série de termos não-físicos são introduzidos para gerar o *morphing*.

1.1 Trabalhos relacionados

Dentre os principais trabalhos publicados na área de simulação de fluidos, podemos destacar o de Foster e Metaxas [3], que foi uma das primeiras publicações com bons resultados nessa área. Contudo, o primeiro a obter uma simulação de fluidos incondicionalmente estável foi Jos Stam, com a publicação de seu *Stable Fluids* [1]. A partir daí, uma série de trabalhos surgiram, baseados no trabalho de Stam, utilizando o arcabouço delineado nele e aprimorando com novas técnicas para aumentar o realismo, melhorar o desempenho e incorporar situações cada vez mais complexas. Alguns exemplos de trabalhos incluem [2, 4–6].

Treuille e McNamara foram os primeiros a realizar a animação de fumaça, imprimindo formas à massa de fumaça [6]. Eles permitem controlar a movimentação da fumaça através de um conjunto de imagens de referência determinadas pelo usuário do *software*. A partir destes pontos de referência, um processo de otimização é empregado para garantir que a evolução da fumaça atinja o seu objetivo da maneira mais próxima possível.

Contudo, o trabalho publicado por Fattal e Lischinski [2], em especial, foi a principal publicação utilizada no desenvolvimento desta dissertação. Neste trabalho, os autores realizam a transformação de imagens através de uma função gaussiana baseada na concentração de densidades no local onde está a fumaça e no ponto onde deveria estar o alvo. Dessa forma, eles se utilizam do valor desta função para determinar a direção e a magnitude da força que guia a fumaça até o seu alvo. O uso dessa função gaussiana apresenta algumas desvantagens, conforme será discutido no capítulo 4.

1.2 Organização do trabalho

Além desta introdução, o restante do trabalho está organizado da seguinte forma:

- Capítulo 2: este capítulo trata dos conceitos básicos envolvendo a física dos fluidos, tanto as denominações dos diferentes tipos de fluidos quanto as equações responsáveis pelo seu movimento.

- Capítulo 3: partindo dos conceitos e equações estabelecidos no capítulo anterior, serão discutidas as principais técnicas e abordagens adotadas pela Dinâmica de Fluidos Computacional para a simulação computacional da física dos fluidos. O mecanismo de advecção, a representação dos dados e a discretização das equações serão discutidos neste capítulo.

- Capítulo 4: neste capítulo serão analisadas as contribuições para a solução da situação-problema, buscando argumentar as razões pelas quais cada técnica foi empregada na implementação do trabalho. Uma série de dificuldades inerentes à concretização da transformação por métodos de difusão são abordadas, visando apresentar soluções específicas para cada um dos obstáculos encontrados no desenvolvimento do trabalho.

- Capítulo 5: os detalhes de implementação são aqui apresentados, bem como os resultados obtidos com o *software* desenvolvido.

- Capítulo 6: por fim, este capítulo apresenta as conclusões tiradas do desenvolvimento do trabalho e alguns possíveis pontos relacionados que podem ser pesquisados no futuro de maneira a aprimorar o trabalho realizado.

Capítulo 2

Dinâmica de fluidos

A dinâmica de fluidos, ou hidrodinâmica, é uma subárea da mecânica dos fluidos que visa estudar o movimento destes quando sujeitos à ação de forças externas quando estas não se anulam, ou seja, quando não há equilíbrio. Neste capítulo, serão apresentados os conceitos básicos e a terminologia utilizada no estudo da física dos fluidos, bem como a formulação matemática que descreve o seu comportamento.

2.1 Mecânica dos fluidos

A parte da física que estuda o movimento dos corpos e as forças que neles atuam é chamada de Mecânica. De acordo com o estado físico dos corpos estudados, a Mecânica pode ainda ser dividida em alguns ramos distintos, sendo um deles a Mecânica dos fluidos, que fornece as bases físicas para o entendimento deste trabalho.

Primeiramente, é necessário conceituar **fluido**. De acordo com a definição de Ferziger e Perić [7], fluidos são substâncias que possuem uma estrutura molecular organizada de tal forma que não gera resistência a forças externas de cisalhamento, que são forças que agem tangencialmente ao fluido. Esta propriedade dos fluidos independe da magnitude da força tangencial aplicada, ou seja, eles não apresentarão resistência alguma, mesmo que esta força seja muito pequena.

Sendo assim, a Mecânica dos fluidos é o ramo da Mecânica responsável por estudar o comportamento dos fluidos em repouso e em movimento, bem como o efeito das forças aplicadas neles. Líquidos e gases são exemplos de corpos estudados por esta área. Mais particularmente os gases são de maior interesse para o desenvolvimento deste trabalho.

A Mecânica dos fluidos se divide em estática e dinâmica. A estática dos fluidos, também chamada de hidrostática, estuda os fluidos em equilíbrio e é importante no estudo de forças que agem em corpos submersos, como a pressão e o empuxo. A dinâmica dos fluidos, ou hidrodinâmica, se concentra no estudo do escoamento dos

fluidos, que ocorre quando estes estão sob a ação de forças externas. A dinâmica será a área de maior interesse para os propósitos desta dissertação.

2.2 Propriedades dos fluidos

Como explicado na seção anterior, os fluidos não oferecem resistência a forças de cisalhamento. Logo, eles se deformam com facilidade e possuem a capacidade de fluir, ou escoar. O escoamento permite que o fluido adquira a forma física do seu recipiente. Os gases, principalmente, fluem com extrema facilidade, buscando ocupar todo o volume disponível no recipiente. Dessa forma, os gases não possuem superfície livre, que é uma superfície criada pelos líquidos que não corresponde a alguma delimitação do recipiente.

Para simplificar o estudo dos fluidos, podemos considerá-los como substâncias contínuas. A Hipótese do Contínuo permite abstrair o fato de que um fluido é composto de um conjunto de moléculas, o que tornaria desnecessariamente complexo o estudo do escoamento. Dessa forma, todas as grandezas são definidas em um elemento representativo de volume (REV), no qual cada propriedade do fluido é representada através de um valor médio. Logo, assumimos que todas as propriedades possuem um valor definido para cada ponto do espaço, implicando na possibilidade de considerá-las como funções contínuas do espaço e do tempo[8].

Algumas das grandezas mais relevantes no estudo dos fluidos são a densidade, a velocidade e a viscosidade. Nas subseções a seguir, falaremos de cada uma delas em detalhes.

2.2.1 Densidade

Quando lidamos com corpos sólidos, frequentemente necessitamos de saber a massa do corpo observado. Em um fluido, entretanto, a massa está dispersa pelo volume do recipiente que o contém. Dessa forma, recorremos ao conceito de **densidade**, que pode ser definida como a massa presente em uma unidade de volume.

Lembrando do conceito de REV, introduzido na seção anterior, teremos um valor da densidade para cada elemento de volume definido. Pela Hipótese do Contínuo, a densidade possui um valor definido para cada ponto do espaço, sendo o REV a menor porção do espaço considerada. Logo, podemos definir um valor da densidade para cada ponto em função de suas coordenadas espaciais. Uma vez que a densidade em um determinado ponto também pode variar com o tempo, a densidade pode ser representada como um campo escalar dado por:

$$\rho = \rho(x, y, z, t) \tag{2.1}$$

2.2.2 Velocidade

Assim como a densidade, a velocidade pode ser definida como um campo de velocidades, partindo do princípio de que o fluido é contínuo. Fisicamente, a velocidade é definida como uma variação no deslocamento em um determinado espaço de tempo. É uma propriedade que só faz sentido ser analisada quando o fluido está em movimento. Se definirmos uma partícula do fluido como sendo um REV, a velocidade em um ponto qualquer do espaço será a velocidade instantânea da partícula que passa por este ponto em um dado espaço de tempo. Sendo assim, para qualquer ponto do fluido, teremos uma dada velocidade que pode variar com o tempo. Logo, o campo de velocidades será dado por:

$$\vec{u} = \vec{u}(x, y, z, t) \tag{2.2}$$

Diferentemente da densidade, a velocidade é um campo vetorial, pois ela possui magnitude e direção. No decorrer deste trabalho, iremos representar as componentes da velocidade nos eixos x, y e z por u, v e w, respectivamente.

2.2.3 Viscosidade

A viscosidade pode ser entendida como a resistência que o fluido impõe à deformação. Quanto maior a viscosidade, maior será a resistência à tensão de cisalhamento aplicada sobre o fluido, que é a responsável pela sua deformação. Fluidos como a água e o ar, por exemplo, possuem baixa viscosidade, enquanto que outros, como o óleo, possuem uma viscosidade maior, o que os torna mais difíceis de serem derramados.

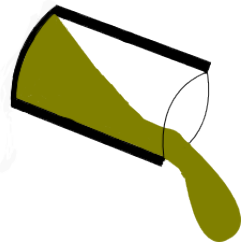

Figura 2.1: Exemplo de escoamento de um fluido com uma viscosidade alta.

2.3 Tipos de fluido

Baseado na relação entre a tensão de cisalhamento aplicada e a taxa de deformação obtida, um fluido pode ser classificado como newtoniano ou não-newtoniano.

Fluidos newtonianos são aqueles cuja taxa de deformação é diretamente proporcional à tensão aplicada. Neste caso, a viscosidade, também chamada de **viscosidade**

absoluta ou **dinâmica**, é constante, desde que não haja outros fatores envolvidos, como alterações na temperatura, por exemplo. Esses fluidos são assim chamados por obedecerem à lei de viscosidade de Newton.

Fluidos não-newtonianos, por sua vez, são aqueles cuja taxa de deformação não é diretamente proporcional à tensão. Neste caso, a viscosidade não é mais constante, passando a ser chamada de **viscosidade aparente**. Ela varia com a tensão de cisalhamento e pode ser dependente ou não do tempo. Esses fluidos podem ainda ser classificados em subcategorias baseadas nas alterações da viscosidade aparente. Essas subcategorias podem ser encontradas em [8].

2.4 Tipos de escoamento

De forma a simplificar o estudo dos fluidos, uma série de categorias para o escoamento são usadas para classificar os fluidos com relação a várias características, dentre elas a viscosidade e a compressibilidade. Nesta seção, serão detalhados os principais tipos de escoamento.

2.4.1 Permanentes e não-permanentes

Um escoamento é dito permanente quando todas as propriedades do fluido são constantes no tempo, podendo, contudo, variar espacialmente. Seja uma propriedade qualquer do fluido $\eta = f(x, y, z, t)$. Matematicamente, esse tipo de escoamento pode ser representado pela seguinte equação:

$$\frac{\partial \eta}{\partial t} = 0 \tag{2.3}$$

Por outro lado, se pelo menos uma das propriedades do fluido varia no decorrer do tempo, então o escoamento é chamado de não-permanente.

2.4.2 Viscosos e invíscidos

A viscosidade tem um papel muito importante no escoamento de fluidos. Quando analisamos uma situação-problema, temos condições de saber o quão relevante é o papel da viscosidade no escoamento estudado através do cálculo do número de Reynolds. Este número é dado pela seguinte expressão:

$$Re = \frac{\rho V L}{\mu} \tag{2.4}$$

Na equação acima, ρ representa a densidade do fluido e μ sua viscosidade. V e L são a velocidade do escoamento e a sua largura, respectivamente. Quando o número

de Reynolds é muito grande, podemos dizer que o efeito gerado pela viscosidade é desprezível. Se, ao contrário, o número for um valor muito pequeno, a viscosidade terá uma contribuição significativa na análise do problema.

Um escoamento é chamado de **invíscido** quando a viscosidade do fluido é nula. Na prática, esse tipo de fluido não existe, mas pode ser muito útil na análise de problemas da mecânica dos fluidos em que a viscosidade pode ser desprezada sem prejuízo dos resultados obtidos. Em muitos casos, o efeito da viscosidade é importante apenas próximo às paredes que limitam o escoamento, de forma que podemos considerá-lo como invíscido na maior parte da região [7].

2.4.3 Laminares e turbulentos

O escoamento laminar é aquele em que as partículas do fluido se movem de maneira organizada, através de camadas suaves (ou lâminas). Um exemplo desse tipo de escoamento é quando abrimos muito pouco uma torneira. O escoamento será bem lento, e a água sairá da torneira suavemente.

Se abrirmos a torneira até o fim, a água sairá com uma taxa maior de escoamento, e o fluxo será muito mais caótico. Esse tipo de escoamento é chamado turbulento, pois as partículas se misturam umas com as outras rapidamente devido às flutuações no campo tridimensional de velocidades [8].

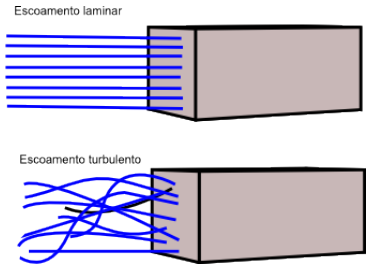

Figura 2.2: Escoamento laminar, acima, e turbulento, abaixo.

O número de Reynolds também pode ser usado para determinar se um escoamento é laminar ou turbulento. Normalmente, se o número for menor ou igual a 2300, o escoamento será considerado laminar. Caso contrário, ele será turbulento.

Na maioria dos casos, a presença de turbulência no escoamento é indesejável, porém muito difícil de ser impedida. O problema da turbulência é que ela gera mais resistência ao movimento do fluido, além de aumentar a tensão de cisalhamento.

2.4.4 Compressíveis e incompressíveis

A compressibilidade é uma das características mais importantes dos fluidos. Ela possui diversas aplicações nos mais variados projetos de engenharia. Um escoamento

pode ser considerado compressível ou incompressível, de acordo com sua variação de densidade.

Um escoamento é considerado incompressível se as variações na densidade do fluido podem ser desprezadas. O escoamento será compressível caso as variações de densidade não possam ser consideradas desprezíveis. Normalmente, líquidos podem ser considerados incompressíveis, enquanto a maioria dos escoamentos de gases são compressíveis.

Líquidos submetidos a altas pressões podem sofrer os efeitos da compressibilidade de forma mais expressiva, sendo, desta forma, considerados compressíveis. Paralelamente, gases nos quais ocorre muito pouca condução de calor podem ser considerados incompressíveis, desde que a velocidade do escoamento seja pequena comparada com a velocidade do som. A razão entre as duas velocidades é chamada de número de Mach. Se o número de Mach for menor que 0.3, o escoamento pode ser tratado como incompressível.

O número de Mach tem ainda utilidade para classificar os escoamentos compressíveis em três subcategorias. Para $Ma < 1$, o escoamento é dito subsônico; caso contrário ele é considerado supersônico. A partir de $Ma > 5$, o escoamento passa a ser chamado de hipersônico. Essas diferenças são importantes pois afetam a natureza matemática da situação sendo analisada [7].

2.4.5 Internos e externos

Quando um fluido escoa dentro de um duto ou uma tubulação, por exemplo, dizemos que o escoamento é interno. Essa classificação se aplica a qualquer caso em que o fluxo seja totalmente limitado por superfícies sólidas. Se o escoamento não é limitado e o fluido corre sobre a superfície de corpos imersos nele, o escoamento é chamado de externo.

Fluidos internos e externos podem ser, simultaneamente, laminares ou turbulentes, e compressíveis ou incompressíveis.

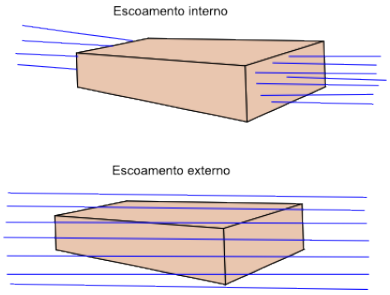

Figura 2.3: Escoamento interno, acima, e externo, abaixo.

2.5 Princípios de conservação

As leis de conservação são de importância fundamental para o estudo da dinâmica dos fluidos. A formulação matemática dos princípios de conservação é feita através de uma massa de controle, ou MC. Porém, como estamos lidando com fluidos, temos que definir um volume de controle VC, uma vez que é difícil selecionar uma porção definida de massa em um fluido.

O **volume de controle** é um dos dois tipos de análise usados nos problemas de mecânica dos fluidos. Ele consiste em selecionar um volume qualquer no espaço através do qual haja escoamento de fluido. Este volume é delimitado por uma superfície de controle, que pode ser estática ou móvel. O outro método de análise utilizado é o **sistema**, que consiste em uma porção finita de fluido, fixa, limitada por um bordo fixo ou móvel. Neste caso, não ocorre escoamento no sistema.

Basicamente, vamos discutir aqui a conservação da massa e a conservação do momento linear. A conservação da massa diz que não existe criação nem perda de massa durante qualquer escoamento. Sendo assim, a equação da conservação da massa pode ser escrita da seguinte forma:

$$\frac{dm}{dt} = 0 \tag{2.5}$$

Já a conservação do momento linear diz que só pode haver variação no momento caso haja uma força agindo sobre o fluido. A equação, neste caso, é a própria Segunda Lei de Newton.

$$\frac{d(m\vec{u})}{dt} = \sum \vec{f} \tag{2.6}$$

O problema das equações acima é que elas dependem da quantidade de matéria presente no volume de controle, uma vez que estamos utilizando a massa nos cálculos, que, como dito anteriormente, é difícil de ser monitorada. Utilizando a abordagem do volume de controle, é possível chegar a uma equação geral para as leis de conservação que não dependa da quantidade de matéria considerada no problema.

Seja ϕ uma propriedade qualquer, a equação de conservação para uma propriedade genérica ficará da seguinte forma:

$$\frac{d}{dt}\int_{\Omega_{MC}} \rho\phi d\Omega = \frac{d}{dt}\int_{\Omega_{VC}} \rho\phi d\Omega + \int_{S_{VC}} \rho\phi(\vec{u} - \vec{u_b}) \cdot \hat{n} dS \tag{2.7}$$

Esta equação é também conhecida como **equação do volume de controle** ou **teorema do transporte de Reynolds**. Ω_{MC} representa a porção do volume de controle no qual a MC está contida, Ω_{VC} é o próprio volume de controle e S_{VC} é a superfície que envolve o volume. Com relação às variáveis, ρ é a densidade, $\vec{u}$ e $\vec{u_b}$ são, respectivamente, a velocidade do fluido e a velocidade com que se move a

superfície do volume, e $\hat{n}$ é um vetor normal à superfície. Para a conservação da densidade, ϕ será igual a 1. Já para a conservação do momento, ϕ será igual a $\vec{u}$. Mais detalhes sobre essa equação e sua derivação podem ser encontrados em [7] e [8].

2.6 Equações de Navier-Stokes

As equações de Navier-Stokes são comumente usadas para descrever o movimento dos fluidos. Diversos trabalhos dentro da Dinâmica dos Fluidos Computacional e da computação gráfica utilizam estas equações como base. Devido à extensão do processo de derivação destas equações e de forma a manter a objetividade deste trabalho, vamos nos abster de demonstrar como estas equações são obtidas. A derivação completa pode ser encontrada em [8].

A primeira equação é a equação de conservação da massa. A partir da equação do volume de controle apresentada na seção anterior, podemos chegar à seguinte equação, também chamada de **equação da continuidade**, conforme demonstrado em [7]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \vec{u} = 0 \tag{2.8}$$

No caso de escoamentos incompressíveis, essa equação pode ser simplificada, uma vez que a densidade é constante com relação ao tempo e possui o mesmo valor em todos os pontos do espaço. Logo, a equação da continuidade se reduz a:

$$\nabla \cdot \vec{u} = 0 \tag{2.9}$$

O princípio de conservação do momento linear gera ainda mais três equações para o conjunto das equações de Navier-Stokes, cada uma correspondente a uma componente da velocidade. São elas:

$$\rho \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \right) = \rho F_x - \frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \tag{2.10}$$

$$\rho \left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \right) = \rho F_y - \frac{\partial p}{\partial y} + \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) \tag{2.11}$$

$$\rho \left(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \right) = \rho F_z - \frac{\partial p}{\partial z} + \mu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) \tag{2.12}$$

Os termos F_x, F_y e F_z representam as componentes da força resultante do somatório de todas as forças externas que agem sobre o fluido. Essas equações, porém,

podem ser unificadas e apresentadas numa forma vetorial que é mais sintética e legível. Esta equação é apresentada em [1] e é exibida abaixo:

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{F} \tag{2.13}$$

Nesta equação, ν é a viscosidade cinemática, que é definida como a razão entre a viscosidade absoluta μ e a densidade ρ. O primeiro termo no lado direito da equação corresponde ao processo chamado de **advecção**, que consiste no transporte de substâncias pelo fluido ao longo do escoamento. Na verdade, não apenas substâncias são transportadas, mas também qualquer propriedade do fluido, como valores de densidade, temperatura e a própria velocidade.

Se observarmos bem este termo da advecção e o passarmos para o lado esquerdo da igualdade, obtemos uma formulação que corresponde à definição de **derivada material**, que descreve a variação no tempo de uma grandeza específica ao longo de um campo de velocidades. Repare que isto é exatamente a definição de advecção. A derivada material pode ser matematicamente definida da seguinte forma, conforme é mostrado em [9]:

$$\frac{D}{dt} = \frac{\partial}{\partial t} + \vec{w} \cdot \nabla \tag{2.14}$$

Podemos então utilizar o operador definido acima para calcular a derivada material de qualquer campo escalar ou vetorial ao longo do campo $\vec{w}$. No caso da advecção, $\vec{w} = \vec{u}$. Podemos então definir equações para a advecção de propriedades escalares do fluido, como a densidade, por exemplo, e propriedades vetoriais, como a advecção do próprio campo de velocidades. As equações correspondentes são apresentadas a seguir.

$$\frac{D\rho}{dt} = \frac{\partial \rho}{\partial t} + \vec{u} \cdot \nabla \rho \tag{2.15}$$

$$\frac{D\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \tag{2.16}$$

A notação com parênteses do termo advectivo apresentada na equação 2.13 é apenas uma simplificação da versão acima. Podemos representar o termo de duas formas: como $\vec{u} \cdot (\nabla \vec{u})$, que envolve um tensor diádico, ou como $(\vec{u} \cdot \nabla)\vec{u}$, que não envolve um tensor diádico. O segundo caso é muito mais simples, o que faz com que essa notação seja normalmente preferida em detrimento da outra [9].

Os outros termos, da esquerda para a direita, correspondem à pressão, à viscosidade e às forças externas que agem no fluido. Aplicando esta equação a escoamentos incompressíveis e invíscidos, eliminamos o termo da viscosidade e obtemos uma

versão simplificada da equação, conhecida como equação de Euler.

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \nabla p + \vec{F} \tag{2.17}$$

Embora as equações de Navier-Stokes descrevam completamente o movimento dos fluidos, a forma como elas foram apresentadas aqui assume um espaço contínuo. No capítulo seguinte, iremos apresentar as versões discretizadas dessas equações, de forma que uma solução para elas possa ser implementada via *software*.

Capítulo 3

Simulação de fluidos

No capítulo anterior, foram apresentados os conceitos básicos que regem o movimento dos fluidos. O estudo dos fluidos e das equações associadas são de grande importância para diversas áreas, incluindo engenharia ambiental, engenharia civil, aeronáutica, entre outras. Contudo, a dinâmica dos fluidos normalmente é muito complexa. Realizar experimentos precisos e bem controlados se torna uma tarefa muito difícil e custosa. Além disso, as equações que descrevem a dinâmica dos fluidos são normalmente equações diferenciais parciais de difícil solução analítica.[7]

Neste contexto, e graças à rápida evolução dos computadores, surgiu uma área de pesquisa conhecida como **Dinâmica dos Fluidos Computacional**. O objetivo desta área é utilizar o poder de processamento dos computadores para automatizar os cálculos dos mais diversos tipos de movimento associados aos fluidos, desde os mais simples até os mais complexos, bem como simular uma série de experimentos que, se fossem realizados na vida real, se tornariam impraticáveis.

Para atingir os objetivos desejados, os *softwares* usados na solução de problemas da dinâmica de fluidos necessitam recorrer à análise numérica, discretizando as equações e aplicando uma série de métodos iterativos, de forma a obter resultados que sejam os melhores possíveis para as simulações, buscando sempre minimizar os erros e aumentar a precisão das soluções.

Neste capítulo, portanto, serão discutidos as discretizações utilizadas, abordagens para a representação computacional dos fluidos, os problemas numéricos comumente encontrados e como são simulados os processos de advecção, difusão e a ação de forças externas nos fluidos. Além disso, será discutido o método de projeção apresentado em [1] utilizado no processo de solução das equações de Navier-Stokes.

3.1 Representação computacional

Ao modelar um problema computacionalmente, é necessário *a priori* definir como os dados serão representados internamente. Basicamente, duas formas de representação

são comumente utilizadas, baseadas em duas diferentes abordagens aplicadas no estudo do movimento dos fluidos. Cada uma tem suas vantagens e desvantagens. Essas duas abordagens serão discutidas a seguir, juntamente com suas respectivas implementações computacionais.

3.1.1 Abordagem euleriana

A abordagem euleriana é a mais usada no campo da Física, principalmente nos casos em que se utiliza a análise pelo volume de controle, explicada na seção 2.5. Esta abordagem visa observar o escoamento em uma determinada região fixa do espaço, por onde passa o fluido. Dessa forma, as propriedades do escoamento são analisadas em função do tempo e do espaço, ou seja, observa-se apenas uma mesma região do espaço, ao invés de seguir um determinado número de partículas ou uma porção contínua de fluido.

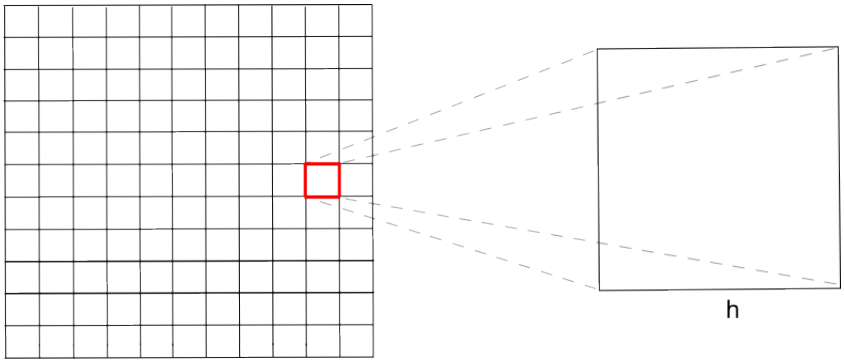

Figura 3.1: Representação computacional na abordagem euleriana

Quando essa abordagem é utilizada, o espaço é discretizado através de uma grade, que pode ser bidimensional ou tridimensional, dependendo do tipo de simulação que se intenciona obter. Cada célula da grade armazena os valores relativos às grandezas do fluido naquele ponto do espaço representado por ela. Dessa forma, a velocidade e a densidade em cada célula é que dirão se existe ou não fluido naquele local. A atenção é totalmente voltada para o estado corrente da célula, e não do fluido em si, o que condiz com a proposta de Euler.

Em geral, assume-se que cada propriedade está localizada no centro de cada célula. Em alguns casos, entretanto, a velocidade é separada em suas componentes, as quais são representadas nas faces (ou arestas, em 2D) de cada célula. Esse tipo de representação da velocidade ajuda a evitar um problema comum na simulação computacional de fluidos conhecido como **instabilidade xadrez**. [2]

Esta instabilidade gera um efeito visual em que valores de densidades muito diferentes surgem em células vizinhas, dando a aparência de um tabuleiro de xadrez.

Suponha que a distância espacial entre duas células da grade seja igual a h. O cálculo das derivadas de primeira ordem via diferenças finitas utiliza uma distância de $2h$ nos casos em que a velocidade está localizada no centro ou nos vértices das células. A separação das componentes da velocidade permite utilizar uma distância igual a h nos cálculos, evitando o efeito indesejado. [10]

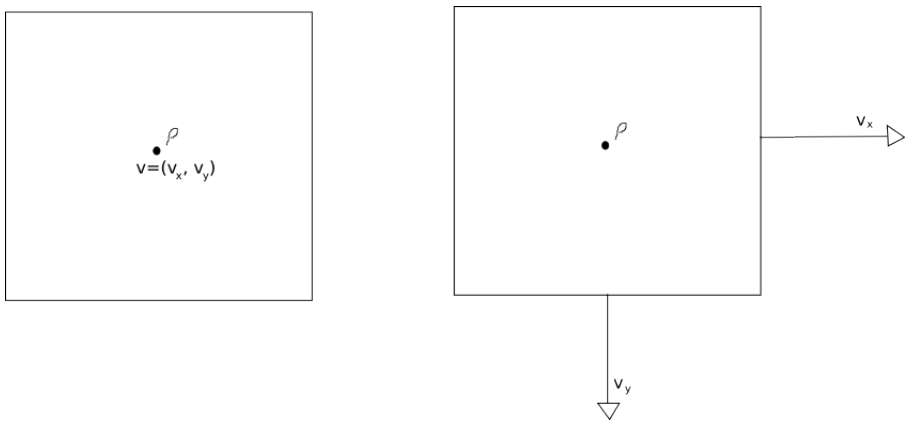

Figura 3.2: À esquerda, as velocidades são discretizadas no centro das células. À direita, velocidades localizadas nas arestas das células

3.1.2 Abordagem lagrangiana

Um outro tipo de abordagem muito utilizada nos problemas resolvidos computacionalmente é a lagrangiana. Dentro da Física, esta abordagem não é muito prática, pois ela consiste em observar as partículas em movimento, ao invés de observar uma região fixa do espaço. Cada propriedade está conectada à sua partícula correspondente. Isso é complicado para os experimentos tradicionais da Física, uma vez que só é possível seguir as partículas quando há um número reduzido e gerenciável delas.

Contudo, com o auxílio do computador, torna-se muito mais fácil gerar e controlar um número muito grande de partículas. Por este motivo, o uso da abordagem de Lagrange tem sido muito comum nos trabalhos que se apóiam na Dinâmica dos Fluidos Computacional. Ao invés de discretizar o espaço em uma grade, as partículas são representadas individualmente, cada uma contendo suas respectivas propriedades, como velocidade, densidade e viscosidade.

Um método que tem sido muito utilizado é o SPH, do inglês *Smoothed Particle Hydrodynamics*. Trata-se de um método de interpolação que foi desenvolvido para a simulação de fenômenos astrofísicos. Ele interpola as propriedades das partículas vizinhas e é muito útil para problemas envolvendo deformação. O trabalho de [11] oferece uma boa introdução ao assunto. Outros trabalhos que utilizam o SPH são [12] e [13].

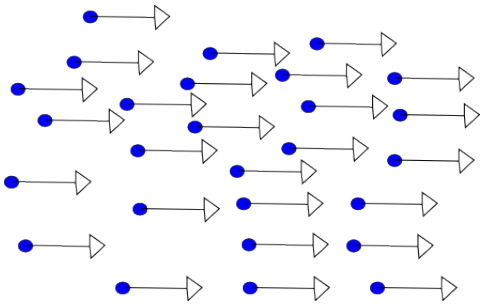

Figura 3.3: Representação computacional na abordagem lagrangiana

3.2 Solução das equações de Navier-Stokes

Na seção 2.6, foram apresentadas as equações da continuidade e do momento, que juntas formam as equações de Navier-Stokes. Para fins de recapitulação, as equações são apresentadas novamente abaixo. Como trataremos com fluidos incompressíveis e invíscidos neste trabalho, iremos apresentar a versão simplificada das equações, também conhecidas como equações de Euler, conforme dito anteriormente.

$$\nabla \cdot \vec{u} = 0 \tag{3.1}$$

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \nabla p + \vec{F} \tag{3.2}$$

Para resolver estas equações, é necessário utilizar-se de métodos numéricos precisos e estáveis para solução de equações diferenciais parciais. Um método bastante estável e simples é o chamado **método dos passos fracionados** ou **divisão de operadores**. Este método é aplicado nos trabalhos de [1], [4] e [2].

O método dos passos fracionados consiste em dividir a equação em partes menores, facilitando, desta forma, a solução da mesma. Através desta divisão, torna-se fácil aplicar o método mais apropriado para a melhor solução de cada parte separadamente.

Vamos supor um operador $\mathcal{L}$ aplicado à variável u na seguinte equação diferencial parcial:

$$\frac{\partial u}{\partial t} = \mathcal{L}u \tag{3.3}$$

Suponhamos que $\mathcal{L}$ possa ser escrito como uma soma linear de um número qualquer de parcelas, das quais cada uma incrementa o valor de u, mesmo que $\mathcal{L}$ não seja linear. Se, para cada uma dessas parcelas, possuirmos um método de diferenciação que permita atualizar a variável, como se essa parcela fosse o único termo no lado direito da equação, nós podemos resolver a equação inteira incrementalmente, diferenciando suas parcelas uma após a outra através de uma sequência de

atualizações.[14]

$$u^{n+(1/m)} = \mathcal{U}_1(u^n, \Delta t)$$
$$u^{n+(2/m)} = \mathcal{U}_2(u^{n+(1/m)}, \Delta t)$$
$$\cdots$$
$$u^{n+1} = \mathcal{U}_m(u^{n+[(m-1)/m]}, \Delta t)$$

Logo, obtemos u^{n+1} a partir de u_n através de sucessivos passos, sempre utilizando o resultado do passo imediatamente anterior no cálculo do passo seguinte. Nas equações acima, m é o total de parcelas em que a equação principal foi dividida e $\mathcal{U}_n$ é o método de diferenciação aplicado à parcela n.

Retomando a equação 3.2, podemos nos aproveitar do conceito de divisão de operadores para resolvê-la de forma mais simplificada. Vamos utilizar a mesma sequência de operações utilizada em [1].

O primeiro passo é calcular o termo $\vec{F}$, que compreende todas as forças externas que agem sobre o fluido. Forças como o vento, o empuxo ou até mesmo forças artificiais com algum objetivo específico, como a força de confinamento de vorticidade utilizada em [4], podem ser adicionadas a este termo. Essa força de confinamento não é uma força real, estudada pela Física. Ela é parte de uma técnica apresentada por [15] para reintroduzir no sistema os detalhes do movimento turbulento dos fluidos perdidos devido à dissipação numérica. No capítulo seguinte, introduziremos uma outra força artificial que terá papel fundamental no desenvolvimento da proposta deste trabalho.

O segundo passo é o mecanismo conhecido como **advecção** ou **convecção**, brevemente explicada na seção 2.6. O termo da equação que descreve este processo é a expressão $-(\vec{u} \cdot \nabla)\vec{u}$. Ela descreve a propagação de qualquer perturbação no fluido. Trata-se da aplicação da advecção do fluido sobre si mesmo, como bem descreve [1]. Os métodos utilizados para simular a advecção serão apresentados na seção seguinte.

Após a advecção, o próximo passo é difundir o fluido. A difusão é mais importante para fluidos viscosos, o que não é o caso deste trabalho, mas ela pode ter um efeito importante para a criação de um movimento mais natural para o fluido, especialmente a fumaça, como veremos nos próximos capítulos. O termo responsável pela difusão é $\nu\nabla^2\vec{u}$, e não está presente na equação de Euler, que trata de fluidos invíscidos. O termo de difusão pode, contudo, ser incluído na equação caso haja necessidade, o que nos leva de volta à equação de Navier-Stokes.

Por fim, o passo final consiste em uma operação conhecida como **projeção**. Este passo tem como principal objetivo garantir que o campo de velocidades tenha divergência nula, atendendo, dessa forma, ao requisito de manter a incompressibilidade do fluido. A projeção envolve o cálculo da pressão, que é o termo restante

da equação de Euler ($-\nabla p$), e a solução de uma equação de Poisson, como será demonstrado mais adiante na seção 3.5.

Ao tentar resolver numericamente essas equações, nos deparamos com uma série de operadores matemáticos de natureza contínua, como é o caso dos divergentes, os gradientes e os laplacianos. Como o computador não é capaz de operar com dados contínuos, precisamos recorrer a uma série de discretizações que nos permitam calcular o resultado de expressões envolvendo estes operadores, de forma a solucionar as equações aqui apresentadas.

Vamos tomar como exemplo um problema em duas dimensões. Assumindo a utilização de uma grade bidimensional, como descrita na seção 3.1.2, podemos definir os operadores necessários utilizando o método das diferenças finitas. Este método consiste em aproximar as derivadas através de diferenças finitas para poder resolver numericamente equações diferenciais. Basicamente, podemos aproximar a derivada através da seguinte equação:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \tag{3.4}$$

Para aplicar o método, ignoramos o termo de erro e definimos o valor de h. Suponhamos uma grade que discretize uniformemente o espaço em N^2 células, e que o espaçamento entre as células seja uniforme. Para o cálculo da derivada explicitada na equação acima, necessitamos de obter o valor da função em dois pontos distintos: x e $x + h$. Como as grandezas de interesse para os propósitos da simulação estão todas localizadas no centro das células, podemos assumir x e $x+h$ como sendo duas células quaisquer, o que nos permite obter o valor de f imediatamente. Se utilizarmos duas células adjacentes, h passa a ser exatamente o espaçamento entre elas, que é um dado previamente conhecido. Dessa forma, podemos utilizar a definição de derivada da equação 3.4 para calcular os operadores discretos que serão necessários para os cálculos desenvolvidos nesse trabalho. Como a grade possui duas dimensões, precisamos definir as derivadas parciais com relação aos eixos x e y, aqui representados, respectivamente, pelos índices i e j. A discretização dessas derivadas, como demonstrado em [4], fica da seguinte forma:

$$F_x = \frac{f_{i+1,j} - f_{i,j}}{h} \tag{3.5}$$

$$F_y = \frac{f_{i,j+1} - f_{i,j}}{h} \tag{3.6}$$

Seja uma grandeza vetorial $\vec{q} = q_x\vec{i} + q_y\vec{j}$. Precisamos então definir expressões para o divergente, o gradiente e o laplaciano desta grandeza. O divergente tem papel fundamental no cálculo da velocidade durante o processo de advecção, conforme será

visto na próxima seção. Podemos defini-lo como:

$$\nabla \cdot \vec{q} = \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} \tag{3.7}$$

Recorrendo às discretizações das derivadas parciais apresentadas acima, podemos definir uma discretização válida para a equação do divergente, substituindo os dois termos no lado direito da equação pelos seus equivalentes discretos. A equação, então, fica da seguinte forma:

$$(\nabla \cdot \vec{q})_{i,j} = \frac{q_{x_{i+1,j}} - q_{x_{i,j}} + q_{y_{i,j+1}} - q_{y_{i,j}}}{h} \tag{3.8}$$

O gradiente é um vetor que aponta na direção de maior crescimento de um campo escalar qualquer. No nosso caso, ele é importante no passo de projeção, no qual necessitamos computar o gradiente da pressão. Vamos assumir uma função escalar $S(x, y)$. O gradiente é então matematicamente definido como:

$$\nabla S = \left(\frac{\partial S}{\partial x}, \frac{\partial S}{\partial y} \right) \tag{3.9}$$

De forma similar ao que foi feito com o divergente, basta que recorramos às discretizações para derivadas parciais para definirmos uma versão discreta do vetor gradiente. Obtemos, então:

$$(\nabla S)_{i,j} = (\frac{S_{i+1,j} - S_{i-1,j}}{h}, \frac{S_{i,j+1} - S_{i,j-1}}{h}) \tag{3.10}$$

Por fim, temos o laplaciano, que é também utilizado na projeção e no processo de convergência da fumaça para a imagem-alvo, como será visto no próximo capítulo. Além disso, possui um papel fundamental nos processos de difusão. Ele é definido como sendo o divergente do gradiente. Decorre daí que podemos facilmente obter a expressão discreta do laplaciano a partir das equações obtidas acima para o divergente e o gradiente.

3.3 O mecanismo de advecção

O ponto fundamental no processo da simulação computacional de fluidos é a advecção. É ela que é responsável pelo transporte das substâncias e das propriedades do fluido ao longo do mesmo. Se tomarmos como exemplo um rio, as substâncias e impurezas presentes na água são transportadas de um ponto a outro do curso do rio pelo movimento da água. Isso é o que chamamos advecção. Propriedades como o calor, por exemplo, também são transportadas de maneira similar. Portanto, qualquer

transporte em um fluido devido ao seu escoamento pode ser chamado de advecção.

A advecção está presente na equação de Navier-Stokes através do termo $-(\vec{u}\cdot\nabla)\vec{u}$. Como visto anteriormente, $\vec{u}$ é o campo vetorial de velocidades que determina o movimento do fluido. Este termo, especificamente, define a advecção do fluido sobre ele mesmo. Em outras palavras, é o transporte da velocidade de acordo com a própria velocidade do escoamento. Por esse mesmo motivo, utilizamos este termo para atualizar a velocidade no fluido a cada instante de tempo.

Para realizar a advecção de outras grandezas, basta que substituamos o segundo $\vec{u}$ no termo destacado. No nosso caso específico, que se trata da simulação de fumaça, interessa-nos atualizar a cada passo a densidade, já que ela é usada para permitir a visualização de uma maior ou menor concentração de fumaça nos diferentes pontos do espaço. Portanto, o termo advectivo para a densidade se torna $-(\vec{u}\cdot\nabla)\rho$.

A questão que surge a seguir é como resolver esta equação. A advecção é o segundo passo no processo de atualização da velocidade apresentado no início da seção. Uma solução possível é aplicar o método de diferenças finitas e obter imediatamente o novo valor da velocidade. Esta abordagem é usada por [3], mas possui algumas restrições para que não se afete a estabilidade. Por exemplo, o espaço de tempo entre cada iteração não pode ser maior que um determinado limite, caso contrário a simulação pode se tornar instável e estourar. Um método estável, contudo, foi apresentado por [1] e acabou se tornando a base para praticamente todos os trabalhos subsequentes na área de simulação de fluidos.

Este método estável para a simulação da advecção é baseado em uma técnica de resolução de equações diferenciais parciais conhecida como método das características. Ele consiste em obter a velocidade do fluido em um instante anterior de tempo e usar o valor obtido para atualizar a velocidade em um determinado ponto do espaço. O detalhamento matemático do método pode ser encontrado em [1].

O esquema utilizado é conhecido como semi-lagrangiano. Ele recebe essa denominação pois, embora a abordagem euleriana seja utilizada, o esquema semi-lagrangiano trata o fluido como sendo composto por partículas, exatamente como ocorre na abordagem lagrangiana. Suponhamos que queremos computar a nova velocidade no ponto x. Imaginamos, então, uma partícula nesse ponto e calculamos qual era a posição desta partícula em um instante de tempo anterior, reconstituindo a trajetória percorrida por ela através do campo de velocidades. Ao identificar a posição inicial da partícula, interpolamos com os valores das velocidades nas células adjacentes e adotamos o valor resultante da interpolação como sendo a nova velocidade do fluido no ponto x.

A principal vantagem deste método em relação ao método utilizado no trabalho de [3] é que este não possui restrição nenhuma com relação ao intervalo de tempo entre as iterações. Por maior que seja o intervalo, a simulação nunca irá explodir.

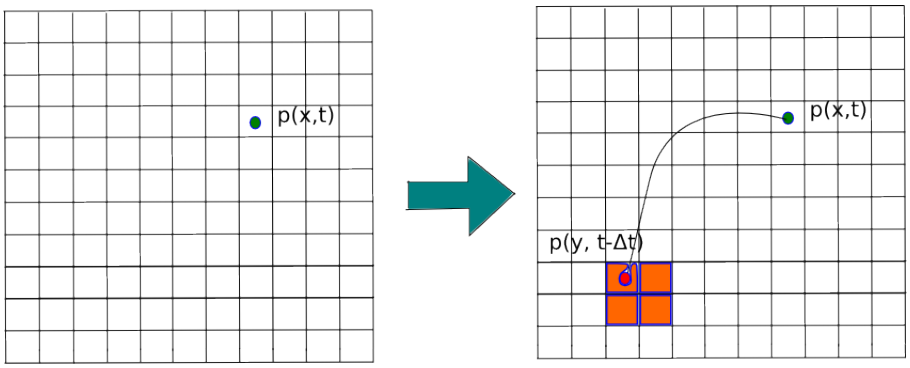

Figura 3.4: Como funciona a advecção semi-lagrangiana

Isso ocorre pois novos valores para a velocidade não são introduzidos no sistema pelo mecanismo de advecção. Como destaca [1], o valor máximo que o novo campo de velocidades pode ter jamais ultrapassa o maior valor do campo de velocidades no instante anterior. Dessa forma, não há o risco de que a advecção provoque um crescimento exagerado das velocidades a ponto de gerar instabilidades na simulação. Além disso, o esquema semi-lagrangiano é extremamente simples e pode ser implementado rapidamente, gerando resultados satisfatórios.

3.4 Difusão

Grosseiramente, existem duas formas de movimentação de substâncias dentro de um fluido. Uma delas é a advecção, explicada na seção anterior, que ocorre apenas quando o fluido está em movimento. A outra forma é a chamada **difusão**, que independe do escoamento do fluido. Mesmo que não haja movimento presente no fluido em um determinado instante, poderá haver um processo de difusão ocorrendo neste momento.

O mecanismo de difusão é essencialmente aleatório. Sempre que há uma diferença de concentração para uma determinada substância entre partes distintas de um fluido, um movimento aleatório surgirá no fluido no intuito de uniformizar a distribuição dessa substância. Dessa forma, se uma porção de água possui uma grande concentração de sal, por exemplo, a tendência é que essa concentração seja difundida pelo fluido, das regiões de maior concentração para as que possuem menor quantidade de sal. Se tomarmos como exemplo a fumaça, que é o objeto de estudo deste trabalho, as partículas em suspensão que compõem a fumaça sofrerão esse processo de difusão através do fluxo do ar, que é o fluido que serve de meio para sua propagação. As regiões com menor densidade de partículas aos poucos serão ocupadas por fumaça proveniente das regiões de maior concentração. Assim como no caso da advecção, não precisa ser necessariamente uma substância a ser difundida.

Qualquer grandeza física identificável no fluido, como a temperatura, por exemplo, pode sofrer difusão.

Por ser um processo que ocorre independentemente da presença de movimento ou de forças externas atuando sobre o fluido, a difusão é essencialmente lenta, não podendo ser utilizada, portanto, como o principal mecanismo para a simulação do fluido. Ela é apenas uma contribuição a mais no sentido de conferir um movimento mais natural ao fluido.

Para resolver o termo de difusão apresentado na seção 3.2, adotamos a abordagem utilizada por [1]. Segundo [1], a forma como [3] resolve a equação de difusão pode gerar instabilidades em alguns casos. Dessa forma, utiliza-se um método implícito, ao invés de discretizar e resolver explicitamente a equação. Obtemos, então, um sistema linear esparso, que tem como solução o novo campo de velocidades. Esse sistema pode ser resolvido utilizando métodos *multigrid* ou através da implementação das rotinas da biblioteca FISHPAK [16], como fez Jos Stam em [1], obtendo bons resultados. Métodos *multigrid* para solução de sistemas lineares esparsos podem ser encontrados em [10].

Para efeitos de completude, a equação apresentada por [1] que gera o sistema esparso é a seguinte:

$$(\vec{I} - \nu \Delta t \nabla^2)\vec{u}'(x) = \vec{u}(x) \tag{3.11}$$

onde $\vec{I}$ é o operador de identidade, $\vec{u}'(x)$ é o novo campo de velocidades e $\vec{u}(x)$ é o campo de velocidades resultante do processo anterior executado na iteração. O coeficiente ν representa a viscosidade do fluido.

3.5 Projeção

O passo final em cada iteração da simulação de fluidos é conhecido como projeção. Este passo tem o importante objetivo de garantir que o campo de velocidades possua divergência nula. É essa condição que garante que a simulação esteja em conformidade com o princípio de conservação da massa, que é representado aqui pela equação da continuidade, já apresentada na seção 2.6.

Primeiramente, devemos calcular a pressão a partir da seguinte equação de Poisson, utilizando o campo de velocidades obtido na advecção, que ainda não representa o campo de velocidades final. [4]

$$\nabla^2 p = \frac{1}{\Delta t} \nabla \cdot \vec{u}' \tag{3.12}$$

A decomposição de Helmholtz-Hodge diz que qualquer campo vetorial pode ser decomposto como sendo a soma de um campo vetorial com divergência nula e o gradiente de um campo escalar qualquer. Se assumirmos $\vec{u}$ como sendo nosso campo

de velocidades desejado, com divergente igual a 0, e a pressão como o campo escalar, podemos decompor o campo de velocidades atual (aqui representado por $\vec{u}'$) em $\vec{u}$ e ∇p. Desta forma, podemos obter $\vec{u}$ apenas reorganizando a equação:

$$\vec{u} = \vec{u}' - \nabla p \tag{3.13}$$

O campo de velocidades resultante é então o campo final desejado. A projeção pode ser demonstrada matematicamente através de um operador $\mathbf{P}$, que projeta um campo vetorial $\vec{w}$ qualquer em sua parte com divergência nula $\vec{u} = \mathbf{P}\vec{w}$, como demonstrado em [1]. Esse operador pode ser usado para definir uma equação única para a atualização da velocidade, que compreende as duas equações de Navier-Stokes apresentadas no capítulo anterior.

Capítulo 4

Morphing a partir da simulação de fumaça

Nos capítulos anteriores, apresentamos os conceitos fundamentais da dinâmica de fluidos e as principais metodologias e abordagens empregadas na área computacional para a simulação de líquidos e gases. Como explicitado anteriormente, o foco do trabalho é na simulação de fumaça. Embora a fumaça em si não seja um tipo de fluido, podemos considerá-la como tal, uma vez que trata-se de partículas em suspensão carregadas por um gás qualquer, como o ar, que por sua vez é um fluido. Mais ainda, podemos considerar que o conjunto de densidades pertence a um determinado gás visível ao olho humano, como é o caso do cloro no estado gasoso.

O objetivo deste capítulo é definir o problema que se pretende trabalhar e discutir soluções possíveis para contornar as dificuldades específicas advindas de diferentes configurações de dados de entrada.

4.1 Introdução ao problema

A idéia principal do trabalho, como brevemente citado no capítulo 1, é apresentar um processo de *morphing* entre duas imagens utilizando mecanismos de difusão de fumaça. O grande problema desses mecanismos é que eles são bastante imprevisíveis e de difícil controle. Sendo assim, torna-se necessário empregar uma série de ajustes ao procedimento padrão de simulação de maneira a contornar os problemas que surgem no caminho.

Basicamente, utilizamos como entrada para a simulação dois arquivos de imagem, cada um contendo uma forma qualquer. Por exemplo, no caso mais simples, podemos imaginar o primeiro arquivo contendo um círculo e o segundo arquivo contendo um quadrado, como ilustra a figura 4.1. Estes são apenas alguns exemplos simples, mas podemos usar formas mais complexas, como mostraremos nos testes realizados no

próximo capítulo.

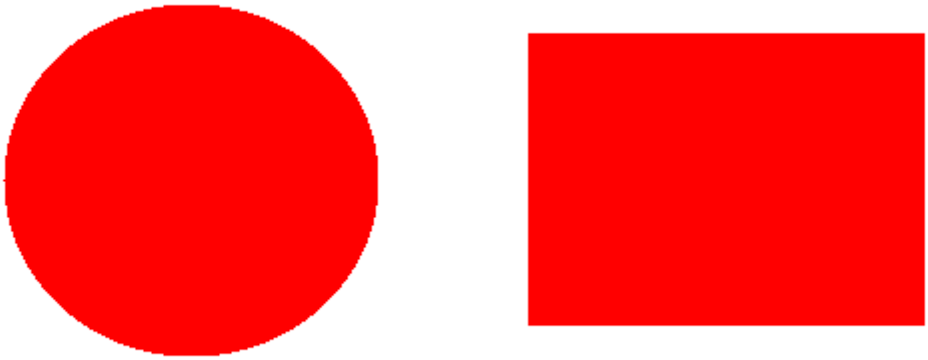

Figura 4.1: Exemplos de arquivos utilizados como entrada para o *morphing*

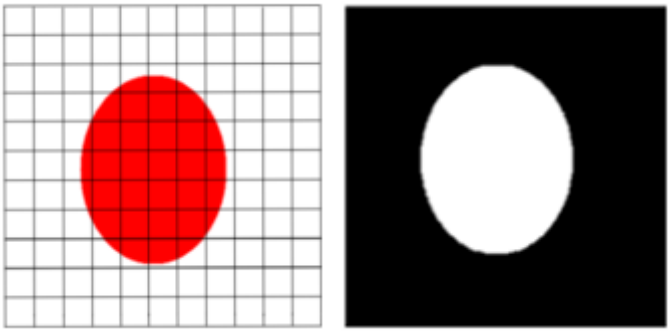

Figura 4.2: Representação da imagem de entrada na grade bidimensional à esquerda, gerando a configuração de densidades na direita

Os pixels que definem a forma contida na imagem são utilizados para representar a presença de densidade naquele ponto. Sendo assim, obtemos uma configuração inicial para a simulação em que as células da grade espacialmente correspondentes às posições dos pixels citados estão ocupadas por uma densidade de fumaça inicial, como ilustra a figura 4.2. Essa densidade inicial é um valor máximo possível para densidade, que no nosso caso varia de 0 (total ausência de densidade) a 1 (densidade em sua quantidade máxima). A partir dessa configuração inicial, queremos gerar a configuração correspondente à outra imagem de entrada através da difusão das densidades geradas para a imagem inicial. A configuração de densidades para esta outra imagem de entrada será o que chamaremos de **alvo**, ou seja, é a configuração

que queremos obter ao final do *morphing*.

Processados os dados de entrada, iremos então aplicar o processo de simulação de fluidos propriamente dito, conforme descrito no capítulo anterior. Como força propulsora da dinâmica da fumaça, utilizaremos uma força artificial criada especialmente para dar o direcionamento correto à fumaça de maneira que a imagem final possa ser atingida. Reiteramos que apenas esta força não é suficiente para alcançarmos o objetivo desejado, em virtude da dificuldade de se controlar o fluido. Por isso, empregamos uma série de ajustes, que serão discutidos ao longo do capítulo, que permitem gerar uma simulação que leve a um *morphing* mais preciso.

Quando a imagem está final está completamente formada, o processo de *morphing* se dá por encerrado, obtendo, dessa forma, a saída final da simulação. O nosso objetivo é obter um resultado visualmente satisfatório, ou seja, a imagem final deve estar plenamente formada, visivelmente nítida para o observador. Além disso, a evolução da fumaça deve parecer o mais natural possível, buscando se assemelhar a um processo físico, evitando transparecer características artificiais que podem resultar da utilização de parâmetros que não sejam baseados na física dos fluidos.

Uma série de problemas podem surgir, contudo, no meio do processo. Imagens simples, como as das figuras apresentadas, são facilmente transformadas, mas quando temos formas mais complexas, com reentrâncias, buracos ou várias componentes conexas, a própria característica imprevisível da dinâmica de fluidos dificulta atingir o estado final do *morphing*. As densidades levadas pelos processos de advecção e difusão muitas vezes atingem pontos do alvo que não deveriam ser ocupados, e estabelecer um critério para que esses processos não ultrapassem o limite desejado não é uma tarefa trivial. Tendo em vista estas questões, iremos discutir, nas seções seguintes, quais foram as abordagens adotadas para se obter os resultados desejados e resolver os problemas encontrados.

4.2 *Morphing* usando modelos de fumaça

As transformações obtidas usando modelos de fumaça indicados na literatura refletem o fato de serem, preponderantemente, calcadas no processo de simular de forma realística a dinâmica de um volume de fumaça, sem que se aborde, ao menos de forma explícita, as questões essenciais do problema de *morphing*. Por simplicidade, vamos resumir aqui essas questões em buscar os cinco objetivos básicos descritos a seguir e analisar o que pode ser alcançável pelas transformações usando modelos de fumaça. Obviamente, alguns desses objetivos só podem ser atingidos em parte devido a se usar um meio gasoso que não preserva forma nem volume. Mesmo assim, vamos enfocar neste capítulo o que se pode fazer, em cada caso, que propriedades

podem ser preservadas e o que se tem de relevar. Os cinco objetivos básicos são os seguintes:

I Em qualquer momento da transformação, o objeto deve conter uma mescla de elementos de sua versão original e da indicada como alvo.

II As características do objeto em si, assim como a influência de cada um dos extremos da transformação no aspecto do objeto, devem evoluir de forma contínua e suave.

III A evolução de cada característica do objeto deve ocorrer de forma bem distribuída durante o processo de transformação a menos que se deva atender a um *script* pré definido que proponha uma sub-divisão do processo onde em cada etapa apenas um conjunto restrito de características é alterado. Por exemplo, a transformação de uma girafa numa zebra, em que primeiro as manchas da girafa devem ficar escuras, depois se transformar em listras, até que, finalmente, o pescoço seja encolhido e ajustes finais produzam uma zebra.

IV A evolução deve ser monótona no sentido de que as características do alvo devem aparecer de forma cada vez mais definida, enquanto as da versão original vão se dissipando.

V O processo deve ser capaz de reconhecer que a situação final foi atingida, ao menos do ponto de vista do observador, e encerrar a transformação.

A seguir, comentamos a busca por cada um desses objetivos dentro do contexto de *morphing* de fumaça.

Empregando um modelo difuso como o de fumaça, é claro que a condição I tem de ser relaxada, devido à volatilidade do meio empregado. Neste contexto, a diluição da forma inicial e o subsequente surgimento da forma do alvo podem ser perfeitamente aceitáveis.

De fato, a pretensão de se manter uma mescla de características dos extremos da transformação durante todo o seu curso vai exigir, no mínimo, um considerável grau de interseção entre essas figuras. Se elas forem disjuntas, o processo de difusão empregado para que a figura inicial se torne sensível à presença do alvo usualmente faz com que a parte do alvo mais próxima da figura inicial tenha influência preponderante, e essa porção do alvo pode não traduzir adequadamente, em nenhum sentido, o alvo como um todo. O exemplo, pretensamente jocoso, que traduz essa situação é o do processo de se transformar uma formiga que está no chão num homem incluir um estágio em que ela se torna um pé simplesmente por que essa é a parte do alvo mais próxima da posição inicial da formiga.

Apesar de não poder se cogitar em atingir o objetivo expresso pela condição I, podemos ao menos tentar reduzir a influência da posição relativa dos objetos extremos no processo de transformação. O melhor que podemos esperar é que o objeto inicial preserve características de sua forma, embora sua densidade se difunda ao redor dele enquanto se mantiver afastado do alvo. Isso certamente é mais apropriado do que promover sua mescla com uma parte do alvo.

A condição expressa em II é atingida naturalmente para a forma e posição do objeto que se está transformando. Já a velocidade com que se processa a transformação requer atenção. Sabemos que na simulação da dinâmica de um fluido a velocidade evolui continuamente até que um limite do recipiente seja atingido, quando ela pode apresentar uma súbita mudança de direção. Claramente, a presença dessa borda, que separa os meios fluido e não-fluido, só é sentida quando o fluido a atinge ou, possivelmente, dependendo da metodologia, alguns poucos estágios antes, não sendo cogitada anteriormente. No caso em questão, atingir o alvo funciona como chegar a uma borda num processo de simulação de fluidos. Neste momento, mudam os objetivos: enquanto se está fora do alvo o objetivo maior é chegar a ele. Já dentro do alvo, o processo deve privilegiar a uniformização da densidade em todos os seus pontos. Se essa uniformização se der a uma velocidade diferente daquela com que o alvo é atingido no processo de transformação, essa descontinuidade é, usualmente, perceptível pelo observador em particular porque provoca retenção de densidade junto ao contorno do alvo.

Com respeito à condição III, há, certamente, uma ordenação - expressa pelo fato de certas características serem transformadas antes de outras - da qual não se pode fugir. Assim, reentrâncias estreitas ou orifícios no alvo só serão escavados depois que é atingido um conjunto que descreveríamos, de forma aproximada, como um "fechamento morfólogico do alvo usando como elemento estruturante a união de umas poucas células". Outro exemplo, já descrito anteriormente, se refere à precedência de se resolver a diferença posicional antes da diferença de forma.

Há, entretanto, exemplos dessas ordenações que podem ser evitados. Um deles diz respeito exatamente a se começar o processo de construção do alvo a partir de sua porção que primeiro é atingida pelo objeto em transformação. Para isso deve-se continuar a movê-lo até que ele se ajuste melhor à forma do alvo. Relembrando o exemplo dado anteriormente, primeiro aumentamos a formiga até que tenha a altura do homem antes de efetivamente fazermos a transformação.

Em relação à condição IV, a situação a ser evitada no caso do *morphing* usando volumes de fumaça é deixar que a forma do objeto original se dilua completamente antes de características do alvo despontarem. Pela nossa experiência, a maneira mais prática de conseguir isso é manter elementos da forma inicial enquanto não há uma sobreposição considerável do objeto que se está transformando com o alvo. É claro

que um processo de difusão ou, mais diretamente, o emprego de uma transformada da distância podem fazer com que a forma do alvo seja percebida mesmo longe de seus limites. Mas, de novo, fora do alvo, é a porção mais próxima que prepondera, e ela não representa o alvo.

Finalmente, a condição V, que não tem sentido num *morphing* expresso pela interpolação de parâmetros, passa a ser objeto de atenção pelo fato de, efetivamente, efetuarmos um processo de simulação de fluidos cujas regras de parada traduzem, essencialmente, ou uma situação de convergência ou que se atingiu um limite de tempo para a simulação. Ocorre que sob o ponto de vista do observador o alvo já está plenamente construído muito antes dessa convergência, quando ainda há densidades e velocidades não nulas fora do alvo. Se o processo não é detido, as oscilações em torno da configuração de convergência causam um *flickering* que não é desejável. Estabelecer uma regra que detenha o processo quando um determinado percentual alto do alvo tiver sido preenchido pode não ser suficiente para deter essas oscilações em função do processo implementado ser discreto. Se o percentual for muito alto, o salto entre uma iteração e outra pode fazer com que ele não seja atingido e ainda assim haja oscilação. Se baixamos esse percentual, imperfeições podem aparecer, especialmente no contorno do alvo, demandando um tratamento de finalização. Conseguir uma solução de compromisso em relação a essas duas possibilidades - ou resolver essa questão de uma outra forma - é a questão a ser resolvida no que se refere à finalização do processo.

4.3 Forças de campo

Na formulação dada em [2] para se obter a força de campo em cada pixel, tanto a densidade do alvo como a do próprio objeto em transformação são submetidas a um processo de difusão pela aplicação reiterada de um filtro gaussiano. A força de direcionamento em cada pixel é, então, obtida tomando-se o gradiente da densidade do alvo difundida ponderado inversamente pela relação entre ela e a densidade, também difundida, do objeto que se transforma. De maneira a tornar mais clara a explicação, apresentamos abaixo a fórmula utilizada no trabalho citado.

$$\vec{F}(\rho, \rho^*) = \tilde{\rho}\frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*} \tag{4.1}$$

Na equação acima, ρ e ρ^* são as densidades no objeto em transformação e no alvo, respectivamente. Quando ρ aparece com um $\sim$ em cima, nos referimos à densidade após a convolução com o filtro gaussiano. Detalhes de como esta equação foi derivada

podem ser encontrados em [2]. A função gaussiana é definida da seguinte forma:

$$g(\vec{x}) = e^{-\vec{x}^T\vec{x}/\sigma^2} \tag{4.2}$$

Essa formulação, entretanto, apresenta inconvenientes, que serão analisados a seguir.

- O gradiente da gaussiana é máximo a uma distância igual a $1/\sqrt{2}$ vezes o desvio padrão, decaindo fortemente a partir daí. Assim, caso a variância escolhida não seja apropriada à distância entre o objeto inicial e o alvo, o procedimento custará a afetar de forma sensível o objeto inicial. Por exemplo, suponhamos que os objetos em questão, um quadrado e um círculo, estejam separados por uma determinada distância, como mostra a figura 4.3. O tempo que o quadrado levará para começar a sentir os efeitos da força de direcionamento de forma mais significativa será relativamente grande, em função dos baixos valores do gradiente, provocando um estado de repouso perceptível na simulação, até que o resultado de várias iterações de aplicação da força gerem uma velocidade suficientemente grande para que o observador possa ver a massa de densidades do quadrado se movimentando de fato. Para evitar esse efeito indesejável - pois o expectador não vê, efetivamente, nada acontecer por um dado tempo - ou os objetos a serem transformados têm que ser colocados a uma distância devida, preferencialmente próximos um do outro, ou essa inércia inicial é eliminada via edição de vídeo. Um dos objetivos deste trabalho, entretanto, é evitar ao máximo essa última alternativa, e por causa disso optamos por empregar outra formulação para a força de direcionamento.

- Podemos, então, aumentar a variância do filtro gaussiano para fazer com que a presença do alvo seja melhor percebida em casos como o da figura 4.3. Mas ao fazer isso perdemos localidade, o que significa desprezar detalhes. O efeito dessa perda no processo é a maior dificuldade de aproximação em trechos do contorno do alvo onde componentes de alta frequência são relevantes. A utilização reiterada de filtros do tipo "Xu and Prince" [17], mais apropriados a tentar resolver esse compromisso, pode minorar esse aspecto sem, no entanto, reduzi-lo substancialmente. Além disso, da equação 4.1 se pode depreender que um pixel do objeto inicial mais longe que outro será atraído para o alvo com mais força, provocando o acúmulo de densidade na porção do objeto sendo transformado que está mais próxima do alvo. O resultado disso, embora em diversos exemplos não chegue a comprometer a qualidade da transformação, nem sempre é desejável.

Tentando evitar os inconvenientes acima, optamos por fazer a força de direci-

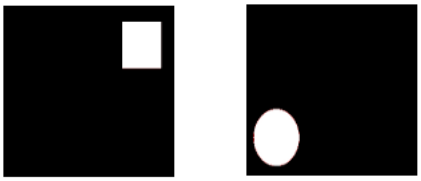

Figura 4.3: Exemplo de objetos separados espacialmente utilizados como entrada para o *morphing*.

onamento proporcional ao gradiente de uma função distância ao alvo. Utilizamos uma formulação matemática similar à empregada por Fattal e Lischinski em [2]. Definimos a força da seguinte forma:

$$\vec{F} = \rho \frac{\nabla \tau}{\tau} \tag{4.3}$$

onde τ é um parâmetro baseado no valor resultante da aplicação da função distância $T(x, y)$ para um determinado ponto (x, y). O parâmetro τ é definido como:

$$\tau = 1 - \frac{T(x, y)}{\max T(x, y)} \tag{4.4}$$

Figura 4.4: Visualização do resultado da aplicação da função distância $T(x, y)$ ao alvo da figura 4.3

Na figura 4.4, podemos observar uma representação gráfica da aplicação da função distância ao círculo que utilizamos como alvo na figura 4.3 mais acima. As regiões mais claras da imagem representam um maior valor para a função. Repare que a partir de um certo limite, a função atinge seu valor máximo. Em contra-

partida, podemos observar que no alvo, como era de se esperar, a função distância possua valor nulo.

Tendo em vista que a norma do gradiente de τ é igual a 1 em todo o ponto da função, o procedimento de *morphing* para objetos espacialmente distantes se inicia imediatamente, pois a força de direcionamento resultante terá uma magnitude suficientemente grande para gerar movimento perceptível no objeto inicial. Implementar adequadamente essa opção, entretanto, requer algumas considerações.

1. Se pensarmos que o gradiente da distância ao alvo é nulo dentro dele isso fará com que o objeto a ser transformado no alvo se aproxime dele com velocidade e se detenha ao chegar ao entorno dele. O *morphing* perde então continuidade, se tornando perceptível a existência de uma fase de aproximação e outra de uniformização da densidade, uma vez que os objetos estejam próximos. Podemos ver isso com clareza na figura 4.5, que mostra a massa de fumaça proveniente do quadrado frear ao chegar na borda do círculo, dando início à fase de uniformização da densidade sem que toda a massa tenha penetrado de fato o alvo. Esse efeito é mais acentuado quando a discretização no tempo é feita de forma mais refinada porque, nesse caso, o processo de visualização fica mais lento e assim, para o observador, mais tempo o objeto em transformação demora a "entrar no alvo". Considerar distâncias negativas dentro do alvo tem o efeito desagradável de concentrar densidade na borda, e por isso essa alternativa foi descartada.

2. Assuma que o objeto alvo é um polígono convexo e seja v um dos seus vértices. Em todos os pontos do cone com vértice em v e delimitado pelas normais às arestas adjacentes a v, o gradiente da função distância aponta para esse vértice. Isso favorece que uma quantidade maior de densidade entre no alvo por um de seus vértices do que por um ponto de fronteira no interior de uma aresta.

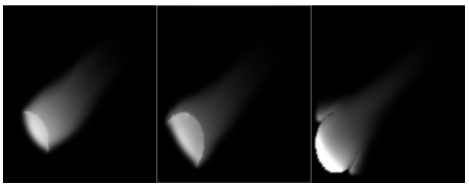

Figura 4.5: Problema de continuidade no *morphing* para o exemplo da figura 4.3

Para evitar a característica não desejável expressa em (1), optou-se por estender para dentro do alvo a velocidade determinada externamente a ele pela força de campo. Isso é feito aplicando-se um esquema do tipo *fast-marching* [18] à velocidade obtida na fronteira do alvo. A esse esquema se juntou um mecanismo de amortização

para que, à medida que a distância à borda, dentro do alvo, aumente, a velocidade seja reduzida de forma suave.

Chamamos atenção para o fato de que isso não é suficiente para impedir que porções do objeto em transformação entrem por um lado do alvo e venham a sair por outro - afinal há outras forças envolvidas que podem determinar isso. Consideramos, entretanto, que esse "*overshoot*" não prejudica a qualidade do processo de *morphing*.

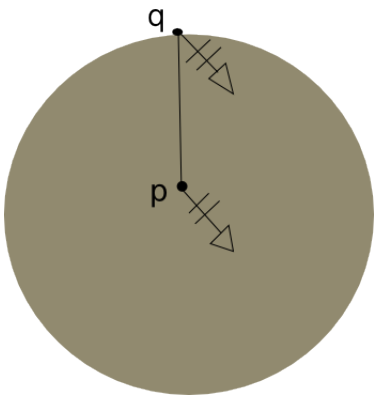

Figura 4.6: Mecanismo de *fast-marching*, em que velocidades da borda são replicadas para o interior do alvo

Para evitar a concentração de densidade em vértices do alvo quando seu contorno não for suave, podemos redefinir a transformada da distância. Obtemos, inicialmente, sucessivas dilatações do alvo usando um elemento estruturante do tamanho de uma célula, ou seja, como a célula é quadrada, utilizamos um pequeno quadrado de lado h (que é a dimensão da célula) como elemento estruturante do processo de dilatação. Imaginemos o conjunto de células da borda do alvo, como mostrado na figura 4.7.

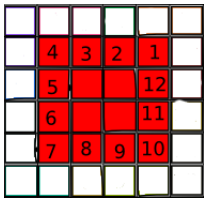

Figura 4.7: Células que compõem um alvo quadrado. As células do alvo estão em vermelho, sendo que as da borda se encontram numeradas.

Associamos, então, as células atingidas em um nível desse processo de dilatação, representado pelas células brancas na figura, com uma célula vizinha que está no nível anterior, que nesse caso é a própria borda. Só que essas associações devem ser feitas de forma que na árvore gerada por elas o número de descendentes das

células dentro de um mesmo nível seja o mais uniforme que se puder conseguir. O descendente de uma célula em um determinado nível é justamente uma célula em um nível posterior atingida pelo processo de dilatação da célula no nível anterior. Observando a figura 4.8, cada célula além da borda na figura é descendente da célula da borda cujo rótulo numérico seja idêntico ao seu próprio rótulo. Logo, células rotuladas como 1 serão descendentes da célula 1 da borda na árvore gerada. De maneira similar, células rotuladas como 2 serão descendentes da célula 2 da borda, e assim por diante.

Figura 4.8: Sequência de dilatações para cada célula do primeiro nível após a borda

Como resultado desse processo, o número de descendentes das células da borda (ou fronteira) do alvo fica bem distribuido. Substituimos então a transformada da distância pela função TD', que a cada ponto exterior ao alvo associa o ponto da fronteira do qual ele é descendente na árvore definida pelas associações descritas acima. Em lugar do gradiente da transformada da distância em um pixel p, empregamos então o unitário do vetor $\overline{pTD'(p)}$.

Figura 4.9: Resultado da distribuição de densidades para o primeiro nível de dilatação.

Essa estratégia tem uma complexidade maior quando a fronteira tem arestas com uma orientação genérica em relação à grade. Em particular, se o alvo tem uma simetria aproximada, seria suficiente envolvê-lo num círculo e considerar o gradiente da distância a esse círculo para os pontos que estão fora dele e o gradiente da distância ao alvo apenas para os que estão dentro.

A seguir, exibimos um procedimento para repartir melhor o conjunto de descendentes dentre os pontos da borda num caso genérico. Vale ressaltar que o procedimento a seguir não gera o caso ideal ilustrado na figura 4.10, a qual demonstra uma distribuição inteiramente uniforme para cada célula da borda. O algoritmo

abaixo tenta aproximar o máximo possível aquela configuração. Com a aplicação do mesmo, nem todas as células da borda obtém o mesmo número de descendentes, porém consegue-se uma clara suavização das densidades nos vértices do alvo. Obter a configuração ideal, contudo, é uma tarefa um tanto mais complexa, e o resultado expresso pelo procedimento a seguir já é suficiente para solucionar o problema em questão.

Inicialização:

Considere um processo de dilatação, iniciado a partir da borda do alvo e que a cada passo acrescenta os pixels externos ao alvo ainda não atingidos mas que possuem um 8-vizinho atingido. Inicialize um contador desses passos fazendo $i = 0$ e faça X_0 ser o conjunto de pixels da borda do alvo. Para cada um desses pixels (p), faça $R(p) = p$ e $D(p) = 1$. Além disso, atribua a todos esses pixels um rótulo para indicar que eles já foram atingidos. O significado de $R(.)$ e $D(.)$ no decorrer da execução do algoritmo é o seguinte:

i Para um pixel genérico - q - que já tenha sido atingido, $R(q)$ é a raiz da arborescência à qual pertence q na floresta orientada gerada pelo processo.

ii Definido apenas para um pixel - p - da borda, $D(p)$ é o número de nós da arborescência enraizada em p, cujos pixels correspondentes já foram atingidos.

Procedimento Iterativo:

```
Faça:
{
    i = i+1;
    Faça X_i+1 ser o conjunto de vizinhos de X_i externos
    ao alvo e ainda não rotulados como atingidos.

    Se X_i+1 for vazio, PARE. Caso contrário, percorra cada
    componente conexa de X_i+1 no sentido anti-horário e para
    cada pixel r atingido execute:
    {
        i) Determine entre os 8-vizinhos de r em X_i aquele
        para o qual D(R(.)) é minimo. Seja V_r esse vizinho.
        Resolva empates favorecendo as transições segundo as
        direções coordenadas em relação às transições na diagonal.
        ii) Faça, então:
            {
                R(r) = R(V_r);
                D(R(V_r)) = D(R(V_r)) +1;
```

```
            }
    }
    Atribua a r o rótulo atingido.
    Repita "Procedimento Iterativo".
}
```

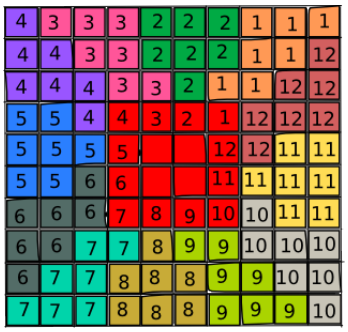

Figura 4.10: Grade 10x10 mostrando o resultado ideal de distribuição de densidades através do processo de dilatação em 3 níveis além da borda

Como tratamos o conjunto X_{i+1} de forma que cada uma de suas componentes conexas seja percorrida de forma contínua e privilegiamos as transições diretas em relação às diagonais, pode-se garantir que, ao representar cada ramo (n_1, n_2) da floresta gerada pelo segmento entre os centros dos pixels associados a n_1 e n_2, não se tenha interseções entre arestas de arborescências distintas. Além disso, se o alvo for um polígono convexo discretizado, as árvores geradas alcançarão as bordas da imagem. Concavidades, é claro, podem fazer com que algumas delas se extingam a partir de um dado nível, sem chegar à borda. No procedimento de *morphing*, em lugar do unitário do gradiente da função de distância, usamos o unitário do segmento $(R(r), r)$ para definir a direção da força de campo no pixel r.

4.4 Evolução da densidade média

A diferença de dimensões entre o objeto inicial e o alvo, assumidos serem compostos de células da mesma densidade, faz com que a densidade total no início e no fim do *morphing* sejam diferentes. Em vista disso, para dar suavidade ao processo de transformação precisamos interpolá-la ao longo do processo, que nesse caso deixa de ser conservativo.

Uma vez obtida por essa interpolação qual deve ser a densidade total - digamos $d(t)$ - de um dado estágio t da transformação, aplicamos um fator de correção uniforme a todas as células de forma que a densidade total obtida pelo processo de simulação se torne, efetivamente, $d(t)$. A questão é quais pesos usar nesse processo

de interpolação. Assumindo que uma interpolação linear entre as densidades totais inicial e final é suficiente, optamos pela seguinte formulação para obter os pesos:

- Considerando a configuração corrente, calculamos a relação entre a soma das densidades das células que pertencem ao alvo e a densidade total do estágio atual. Seja R_t essa relação.
- Fazemos o mesmo em relação ao estágio inicial, obtendo a relação R_0.
- O peso atribuido à densidade total final é então computado, fazendo:

$$p_0(t) = \frac{\max(R_t - R_0, 0)}{1 - R_0} \tag{4.5}$$

4.5 Uniformizando a densidade em alvos não-convexos

Dentro do alvo o objetivo é essencialmente uniformizar a densidade dos pixels. Dentro de uma componente conexa do alvo, a uniformização da densidade pode ser provida pela aplicação de um operador laplaciano. A velocidade com que essa uniformização se processa pode ser aumentada pela aplicação reiterada desse operador numa mesma iteração. Entretanto, se o número de aplicações desse operador cresce, pode-se perder a continuidade do processo, porque o intervalo entre duas visualizações sucessivas cresce, e a diferença entre elas fica mais notória. Para tornar mais ágil o processo, uma possibilidade seria utilizar um mecanismo multiescala - o que significaria aplicar o operador num nível de resolução mais baixa. Essa perda de resolução pode ter o efeito indesejável de criar estruturas de contorno ajustadas à orientação da malha. Em resumo, a aplicação repetida do laplaciano deve ser aplicada modicamente, restringindo-se a um número pequeno.

Porém, o que a mera aplicação de um operador laplaciano não consegue garantir é a uniformização da densidade em todo o alvo quando ele tem mais de uma componente conexa. Normalmente, em cada componente há convergência para um valor diferente, e o processo de normalização empregado não consegue resolver isso, resultando que ao final do processo se tenha um valor maior que 1 para algumas componentes e menor que 1 para outras. De fato, resolver essa questão iria requerer a aplicação de um procedimento específico, eventualmente considerando a topologia do alvo. A dificuldade é que um procedimento desse tipo não deve ser aplicado como se fosse um pós-processamento ao *morphing*, tornando distinguivel a existência de etapas da transformação.

Uma solução mais direta para essa questão se revelou ser a simples substituição do operador Laplaciano para aquele que atribui a um pixel o valor máximo dos vizi-

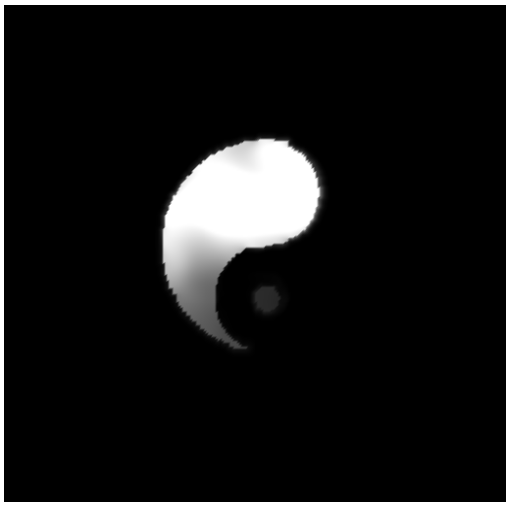

Figura 4.11: Problema de uniformização da densidade. Repare que o círculo menor exterior do Yin-Yang praticamente não aparece.

nhos (`Max_Viz`). Em nossos testes essa solução funcionou bem, essencialmente pelo simples fato de que na maior parte dos casos o processo de advecção leva a densidade 1 do objeto inicial - ou valores altos, próximos de 1 - a qualquer componente do alvo. Se isso acontecer, esse valor será propagado para toda a componente pela aplicação do operador `Max_Viz`, o que determina que os valores para onde converge a densidade em todas as componentes do alvo sejam todos mais próximos entre si e próximos de 1.

4.6 Desvinculando a densidade do brilho da célula

No sentido de prover determinados efeitos ao processo de transformação, é interessante desvincular a densidade que está sendo transformada durante o processo de *morphing* do brilho atribuído a uma célula na visualização da metamorfose. Por exemplo, podemos pretender que o alvo se apresente envolto em uma névoa, como se ele se mantivesse fumegante durante a parte final da transformação. De forma análoga, pode-se querer apresentar o objeto em transformação como se ele estivesse mais disperso do que realmente se encontra. Em ambos os casos, podemos utilizar o fato de que ao longo do processo há um conjunto de células cuja densidade é tão reduzida que, mantendo o brilho de uma célula proporcional à sua densidade, esse brilho nem será perceptível. Se entretanto obtivermos o brilho por uma função que para densidades baixas, até um dado limite L, promova uma amplificação bem maior do que desse limite em diante, então podemos torná-las perceptíveis. Idealmente elas devem aparecer com um brilho fraco para caracterizar a presença de uma névoa que acompanha o objeto em transformação ou um rastro que ele deixa.

Uma função F que se aplique a essa finalidade deve possuir um conjunto de propriedades simples tais como:

I $F(0) = 0$

II F deve ser crescente.

III F deve ser múltipla da identidade ($F(x) = kx$) fora de um intervalo da forma $[0, L]$ para que se mantenha a proporcionalidade entre brilho e densidade fora desse intervalo.

IV A derivada de F (F') deve ser contínua no intervalo $[0, L]$.

Para se fazer cumprir essas especificações, o mais complicado é prover a continuidade de F e F' no limite do intervalo (L). Uma opção que satizfaz aos requisitos I, III e IV é dada por:

$$F(x) = \begin{cases} kx + \lambda \cdot (L - x)^2 \cdot \left[1 - \left(\frac{(L-x)^2}{L^2}\right)\right] & 0 \leq x \leq L \\ kx & L \leq x \leq 1 \end{cases} \tag{4.6}$$

Fazendo $\lambda = (3\sqrt{6})/(4L)$, obtemos que $F'(x) > 0, \forall x \in [0, L]$, tornando F crescente nesse intervalo. Observamos que polinômios de grau menor ou igual a 3 não conseguem satisfazer simultaneamente às seguintes restrições: $F(0) = 0$; $F'(L) = k$; $F(L) = kL$ para algum k pré-definido; F crescente em $[0, L]$. O valor de λ especificado acima é o maior possível para o qual $F'(x^*) \geq 0$, onde $x^* = (1 - \sqrt{6}/6)L$ é o único zero de F'' em $[0, L]$, onde ocorre o mínimo de F' nesse intervalo. O gráfico ilustrativo da função é exibido a seguir.

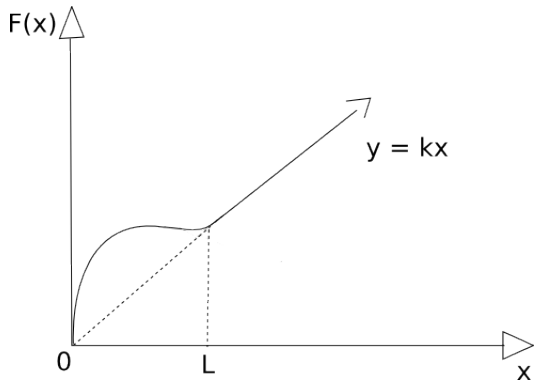

Figura 4.12: Gráfico da função que define o brilho da célula

4.7 Escavando reentrâncias e produzindo buracos no alvo

Procedimentos de difusão tais como o empregado por Fattal e Lischinski [2] para definir o campo no exterior do alvo eliminam frequências mais altas e, portanto, tornam mais difícil reproduzir reentrâncias estreitas que o alvo eventualmente venha a ter. Fazer surgir um buraco no alvo depois dele ter sido ocupado - isto é, ter células com densidade perceptível - no processo de transição também não pode ser obtido simplesmente pela advecção da densidade, porque, nesse caso, como ilustra a figura 4.13, as células do buraco podem trocar de densidade umas com as outras sem que haja uma redução na densidade total do buraco. As setas na figura mostram as direções tomadas pelas densidades devido à advecção. A densidade na base da seta é buscada pelo processo de advecção semi-lagrangiano e repassada para a posição na ponta da seta. Isso se repete por toda a reentrância, impedindo a abertura da mesma.

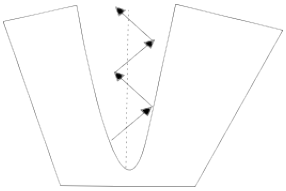

Figura 4.13: Troca de densidades dentro de uma reentrância de um objeto em formação devido à advecção.

Mecanismos para tornar o processo de advecção capaz de lidar com ambos os casos provêm de metodologias empregadas para fazer a evolução de *level-sets*, onde a situação em que a "trajetória" percorrida no processo de advecção corta o eixo medial (esqueleto) do *level-set* zero. Nessa situação, se o *backtracking* realizado na advecção a partir de um ponto x produz o seguinte ponto:

$$y = x + \Delta t \cdot \nabla F(x) \tag{4.7}$$

então $\nabla F(y)$ tem direção completamente diferente da de $\nabla F(x)$ e, portanto, x não poderia ser atingido a partir de y no pequeno espaço de tempo Δt em que o deslocamento da "partícula" é bem aproximado pela expressão 4.7 acima. Essa situação pode ser identificada pelo fato de $\nabla F(x)$ e $\nabla F(y)$ fazerem um ângulo grande (Condição I) e provendo alternativas para realizar a advecção, cuja escolha - voltando ao nosso contexto de *morphing* - está vinculada ao propósito que se esteja objetivando.

Assim, se escolhermos realizar advecção, substituindo $\nabla F(x)$ por $(1/2)(\nabla F(x)+$

$\nabla F(y)$) estamos favorecendo o processo de escavação de uma reentrância no contorno do alvo, porque na parte aberta (boca, entrada) dessa reentrância, ∇F tem uma componente que aponta na direção (do eixo) da reentrância, e a componente transversal a ela tem uma orientação que depende do lado do eixo medial do alvo em que o ponto se encontra. Assim, se somarmos os gradientes em dois pontos, situados em lados distintos do eixo medial, a componente transversal tende a se atenuar e prevalece a que aponta na direção da reentrância. Isso significa que a advecção irá buscar para a boca da reentrância densidades que se encontram fora da mesma, para as quais o processo não tem dificuldade de reduzir o valor, fazendo com que a reentrância seja penetrada um pouco mais.

Essa alternativa, entretanto, não seria suficiente para abrir buracos no alvo que estejam cobertos. Nesse caso, não se tem de onde buscar as densidades mais baixas, pois o buraco não faz fronteira com o exterior do alvo. Para dar início ao processo de abertura do buraco, o recurso, então, é apagar os pontos (X) para os quais a Condição I se verifique. Considerando que a resolução das células é consideravelmente mais fina que a do buraco, isso dará origem a um processo de expansão que parte do centro para as bordas do buraco. Uma vez que o orifício criado tenha uma dimensão maior do que o máximo deslocamento por iteração, podemos voltar ao processo de advecção padrão, pois a partir daí a advecção terá onde buscar densidades de valores baixos, que será justamente nos pontos do buraco onde ocorreu a eliminação dos valores que deu início ao processo de abertura. Em relação a essa alternativa, a questão fundamental é quando dispará-la, dado que não se pode cogitar em identificar que uma dada célula pertence a um "buraco no alvo".

Suponha que queremos transformar um objeto O_1 em outro O_2 que é similar a ele, diferindo apenas pela existência de um buraco circular C tal como está mostrado na figura 4.14.

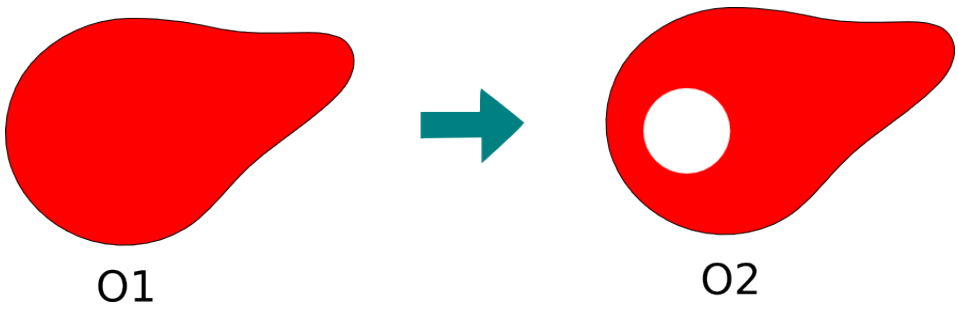

Figura 4.14: Transformação de um objeto fechado em um alvo com buraco

Se a advecção da densidade for feita na forma padrão então o ponto p receberá no próximo estágio a densidade do ponto $q = p - v\Delta t$ (vide figura 4.15). Como para todos os pontos de C teremos uma situação análoga, C continuará cheio.

Entretanto, se realizarmos essa transferência de densidade apenas se $\langle v(p), v(q) \rangle > 0$, então um buraco circular concêntrico a C será aberto na próxima iteração. Criado esse buraco, nas iterações subsequentes ele se ampliará, acabando por ocupar todo o círculo C.

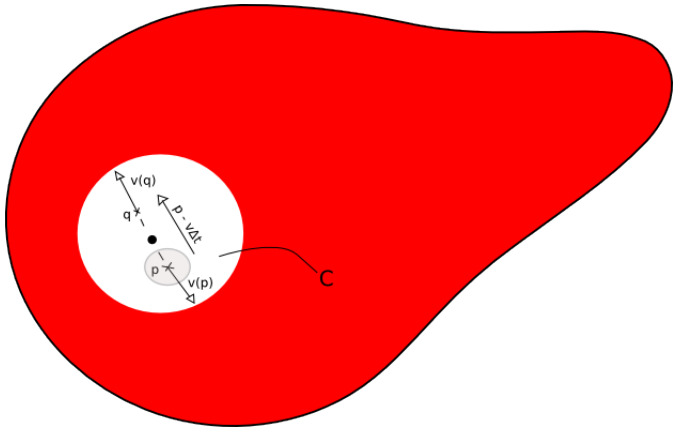

Figura 4.15: Transferência de densidades no buraco do alvo

4.8 Eliminação do *flickering*

Em especial nas situações em que pretendemos produzir figuras finais estáveis - alternativamente, podemos querer obter "figuras fumegantes" onde a densidade do alvo oscila indefinidamente - é não desejável a situação em que a convergência para o alvo se realize exceto por pequenas oscilações que só vão desaparecer muito tempo depois. Enquanto não desaparecem, elas produzem um efeito de *flickering*, que demonstra que o processo ainda não se encerrou embora nada de significativo aconteça em canto nenhum da imagem no que diz respeito à transformação. Se bem que haja outras alternativas, e que procedimentos *anti-aliasing* reduzam a intensidade do *flickering*, particularmente nas bordas, adotamos para tratar essa questão o procedimento que nos pareceu mais simples. Não haveria *flickering* se ao final do processo tivéssemos um intervalo de tempo entre as iterações muito reduzido, de forma que as oscilações não fossem perceptíveis de uma iteração para a outra. É claro que um intervalo muito pequeno não poderia vigorar durante o processo de transformação, levando à conclusão óbvia que ele teria de ser reduzido durante o processo, à medida que nos aproximamos da situação final.

Para medir a proximidade da situação final, poderíamos usar a fração da densidade total do alvo que já está obtida na situação corrente (sumTargetCurrent/sumTargetFinal). O intervalo vigente poderia ser uma combinação convexa ponderada por essa fração e seu complemento a um de um intervalo inicial grande e um final pequeno. Levando em conta que a diferença entre as

densidades totais inicial e final nos obriga a usar um fator de correção, adotamos determinar o intervalo da iteração corrente segundo a seguinte formulação:

$$\Delta t = \Delta t_0(1 - k^2) + 0.005k^2 \tag{4.8}$$

onde k é o valor máximo entre o último k computado e o fator de correção multiplicado pela razão entre a densidade total corrente e a densidade total final.

4.9 Procedimentos *anti-aliasing*

Eventuais problemas de *aliasing* podem surgir ao longo da borda do objeto em transformação. Isso se manifesta especialmente nos extremos do processo, quando há um contraste mais definido nas bordas. No meio do processo, a névoa que normalmente envolve o objeto atenua esse efeito. Além disso, a própria forma dos objetos dados e a resolução empregada podem propiciar o aparecimento do *aliasing*.

O tratamento que demos a essa questão foi a aplicação de filtros aplicados a vizinhanças 4x4 nos moldes indicados por Catmull [19] e Lanczos [20]. No primeiro, o novo valor do centro da vizinhança é substituído pela altura desse ponto em uma superfície bicúbica. No segundo, o polinômio cúbico em cada direção coordenada é substituido por uma função da forma:

$$L(x) = \begin{cases} \frac{a\ \mathrm{sen}(\Pi x)\mathrm{sen}(\Pi x/a)}{\Pi^2 x^2} & -a < x < a, x \neq 0 \\ 1 & x = 0 \\ 0 & \text{caso contrário} \end{cases} \tag{4.9}$$

onde a é um inteiro pequeno. Para que o *kernel* do filtro seja 4x4, a deve ser feito igual a 2. O método de Lanczos é reputado como sendo o que apresenta o melhor compromisso considerando-se conjuntamente os objetivos de redução do *aliasing*, manutenção do contraste e a não-existência de oscilações. Esclarecemos que a aplicação desse filtro pode ser aplicada à imagem inteira ou apenas aos contornos, quando as dimensões da imagem forem grandes e precisarmos economizar tempo para não tornar o processo de transformação desagradavelmente lento.

Capítulo 5

Implementação e resultados

No intuito de validar as idéias apresentadas nos capítulos anteriores, foi desenvolvido um *software* que implementa um simulador de fumaça e realiza o *morphing* da imagem inicial para a imagem final, gerando uma animação visualmente aceitável. Neste capítulo, serão apresentadas as principais decisões em nível de projeto do *software*, como a representação dos dados e a estrutura do programa, e como foram implementadas as técnicas discutidas até aqui.

5.1 Introdução

O desenvolvimento do programa mencionado foi parte essencial do trabalho e permitiu verificar a relevância dos métodos empregados para a solução do problema proposto. Podemos identificar duas etapas bem definidas no processo de implementação realizado. A primeira consiste da implementação de um simulador de fluidos genérico, que teve por finalidade gerar uma animação fisicamente plausível, utilizando como base as equações discretizadas de Navier-Stokes apresentadas no capítulo 3. Os processos de advecção, difusão e projeção são todos realizados nesta etapa. O simulador não tem, no entanto, o propósito de ser rigorosamente realista, já que o objetivo final do trabalho é o *morphing*, que é um processo de animação não realista.

Para a implementação desta etapa, foi utilizado como base o código-fonte do próprio Jos Stam, referente ao trabalho apresentado em [5]. O código original é bem simples e eficiente. Foi desenvolvido em C, utilizando OpenGL para os gráficos e a biblioteca GLUT para realizar a interface com o usuário. O programa de Stam usa um esquema básico de renderização para gerar uma animação de fumaça controlada pelo mouse. O usuário pode, com o botão direito do mouse, aumentar a densidade de fumaça em um determinado ponto do espaço delimitado pela janela do aplicativo. Com o botão esquerdo, ele pode aplicar forças à fumaça, fornecendo a direção da força através do movimento do mouse. A aplicação das forças e da densidade faz

com que se inicie o processo de simulação, que a partir daí fica a cargo da advecção e da difusão, que mantêm o fluido em movimento, até que a fumaça se dissipe completamente.

Para facilitar os nossos propósitos, reestruturamos o código de Stam, de forma a ficar mais legível e extensível. O código, que era puramente escrito em C, foi convertido para C++, utilizando o paradigma de orientação a objetos. Foram também realizados alguns testes com Python no início do desenvolvimento, na tentativa de se utilizar a biblioteca `numpy`. Essa biblioteca implementa uma série de operações sobre estruturas matemáticas como vetor e matriz. Contudo, como o Python é uma linguagem interpretada e a adaptação do OpenGL disponível para a linguagem não é muito eficiente, os resultados acabaram se mostrando bastante insatisfatórios, e esta opção foi consequentemente descartada.

A segunda etapa da implementação é constituída pelo *morphing* propriamente dito. Nesta fase do desenvolvimento, o simulador de Stam adaptado foi estendido para incorporar o código que realiza a transformação de imagens. O programa usa como entrada duas imagens, uma para ser a imagem de origem e outra representando a imagem-alvo que a fumaça deve atingir. Para utilizar essas imagens, recorremos ao módulo de manipulação de imagens do Qt, que pôde ser facilmente incorporado ao código e serviu aos nossos objetivos. A implementação do programa como um todo, tanto da primeira etapa quanto da segunda, será analisada em maiores detalhes nas seções seguintes deste capítulo.

5.2 Estruturas de dados

Na seção 3.1, foram apresentadas duas abordagens que podem ser adotadas na representação computacional dos dados do fluido. Decidimos por adotar a abordagem euleriana, uma vez que os principais trabalhos que usamos como base para este ([1], [4] e [2]) também adotaram esta opção.

Como o objetivo do trabalho é o *morphing* de imagens, utilizamos uma grade bidimensional para implementar uma solução ao problema da transformação. Foi criada uma classe `Grid2D` que representa a grade propriamente dita. Ela armazena informações como a resolução da grade, o seu tipo e as células que a constituem. O tipo da grade foi definido para podermos fazer uma diferenciação mais clara entre os estados intermediários e os estados inicial e final. Os tipos de grade são:

- `SOURCE_GRID`: representa o estado inicial usado como entrada para a simulação. Contém a distribuição de densidade relativa à imagem de origem do *morphing*.
- `SWAP_GRID`: representa qualquer estágio da simulação entre os estados inicial e final.

- `TARGET_GRID`: representa o estado final. Contém a distribuição de densidade relativa à imagem-alvo da transformação.

As células da grade são representadas através de um vetor de objetos da classe `GridCell`. Os objetos dessa classe armazenam os valores das grandezas físicas relevantes ao fluido. Além disso, são computadas e armazenadas as relações de vizinhança entre todas as células, de modo a facilitar os cálculos durante a execução do simulador. A velocidade foi representada no centro de cada célula, contudo, o *software* foi desenvolvido de forma a ser facilmente adaptado para a implementação das velocidades nas arestas das células, que é a alternativa adotada por [4]. Sabemos que há o risco de instabilidades relacionadas com a representação centralizada das velocidades, conforme explicado na seção 3.1.1, porém, como utilizamos o código de [5] como base e não tivemos problemas relacionados com este tipo de representação, não vimos necessidade de fazer uso da alternativa mais estável, que não chegou a ser implementada.

5.3 Visão geral do *software*

A implementação do programa foi realizada usando o paradigma de orientação a objetos. Basicamente, existem dois módulos centrais no *software*: o módulo responsável pela simulação, representado pela classe `Simulation`, e o módulo que realiza a visualização dos resultados da simulação, representado pela classe `Rendering`. Ambos os módulos acessam as estruturas de dados da forma como é mostrada no diagrama da figura 5.1.

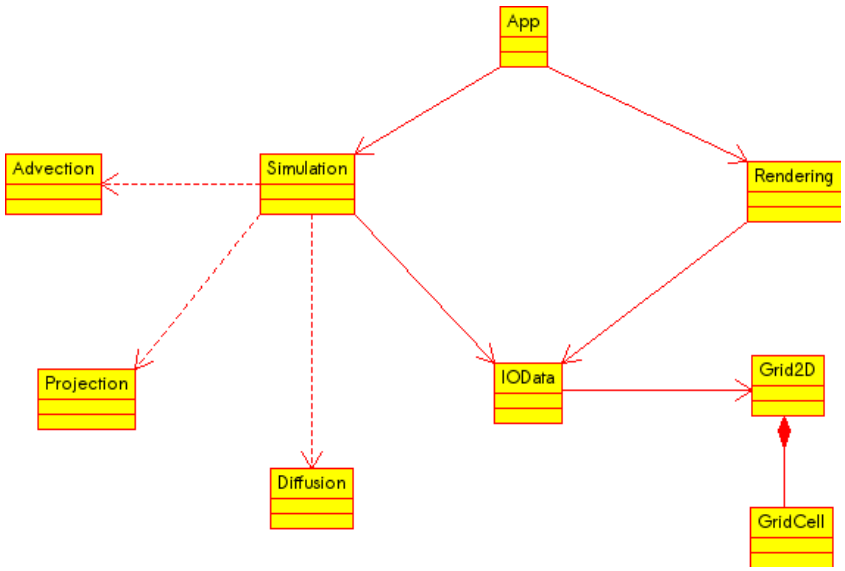

Figura 5.1: Diagrama de classes do *software*

Para realizar a interface com o usuário, foi criada a classe `App`, que encapsula as funções de *callback* da GLUT. A classe implementa o padrão *Singleton*, garantindo que apenas uma instância da classe seja criada durante a execução do programa. Através da janela de visualização gerada pelo GLUT, o programa permite ao usuário visualizar a evolução da fumaça através de uma representação em tons de cinza da distribuição das densidades, bem como a possibilidade de visualizar as alterações no campo de velocidades, caso haja necessidade. É possível ainda alternar entre diferentes métodos de renderização para comparar o resultado final.

Os dados de entrada para a simulação são fornecidos através de um arquivo de texto, que é lido e interpretado durante a inicialização do *software*. Dados como o intervalo de tempo utilizado entre cada iteração, resolução da grade, módulo da força de direcionamento, entre outros, podem ser fornecidos através deste arquivo de entrada. O *software* então se encarrega de utilizar os dados fornecidos para criar as estruturas de dados de acordo. Após este processo inicial de configuração, inicia-se a simulação propriamente dita. O processo é repetido continuamente até que o usuário pare a execução do programa ou provoque uma pausa temporária na simulação.

A saída do programa é gerada também a cada iteração, após o processo de simulação. A distribuição de fumaça é continuamente renderizada no seu estado corrente, passo após passo, gerando uma animação do movimento da mesma. A fluidez da animação será influenciada pela velocidade de execução do simulador. Visando os casos de simulações que demandam mais poder de processamento, foi implementada também a possibilidade de realizar uma renderização *offline* do procedimento. Os dados de saída da simulação são gerados e armazenados em um arquivo binário, que pode ser usado mais tarde como entrada para o visualizador.

5.4 Simulador de fumaça

Durante o processo de inicialização do programa, quatro objetos da classe `Grid2D` são criados: um para representar a imagem inicial do *morphing*, um para a imagem final e os outros dois para os estados intermediários entre as duas imagens, que são os estados que compõem a animação em si. Estas duas grades intermediárias estarão no centro das discussões desta seção, enquanto as outras duas serão consideradas na seção seguinte.

As duas grades utilizadas durante o processo de simulação são do tipo `SWAP_GRID`, explicado na seção 5.2. Essas duas grades, por serem do mesmo tipo, possuem exatamente a mesma funcionalidade e intercalam, a cada iteração do simulador, as funções de entrada e saída da simulação. Por exemplo, na primeira iteração, os dados armazenados na grade 1 serão utilizados como dados de entrada para o primeiro passo da simulação. Após a execução deste passo, o resultado será armazenado na

grade 2. Antes que a próxima iteração comece, as grades trocam de posição. Aquela que antes servia como saída agora se torna a entrada do simulador, já que ela possui os dados atualizados necessários para o próximo passo.

A necessidade de utilizar duas grades se dá pelo fato de que muitos dos cálculos realizados para cada célula envolvem as células vizinhas. Suponhamos uma célula C, que possui como vizinhas as células A, B, D e E, como mostra a figura 5.2. Imagine que as células A e B já tenham tido seus novos valores calculados. Se tivéssemos atualizado esses valores na própria grade 1, teríamos que calcular as grandezas referentes a C usando os valores já atualizados de A e B, junto com valores desatualizados de D e E. Ou seja, estaríamos misturando dados de duas iterações diferentes, o que não geraria o resultado esperado. Dessa forma, armazenamos os novos valores de A e B na grade 2, assim como os valores de C, D e E e, somente ao final do processo, permutamos as grades.

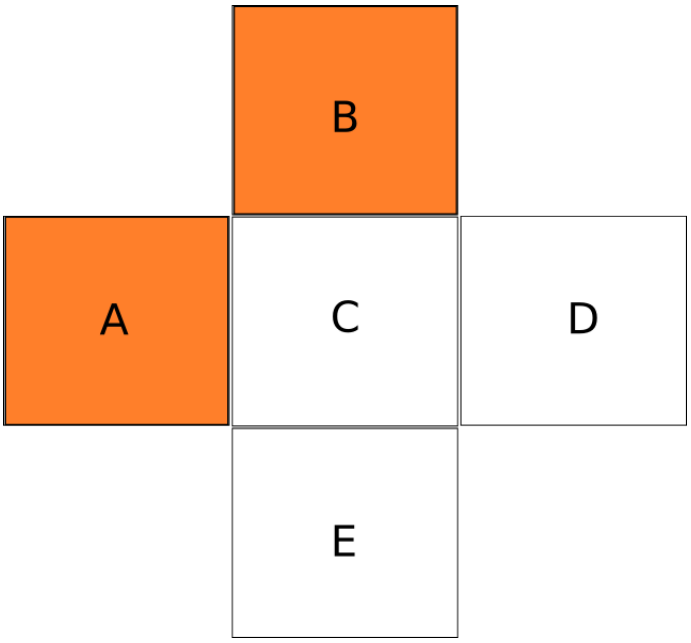

Figura 5.2: Estado de 5 células vizinhas em uma dada iteração do simulador. As células destacadas já tiveram suas grandezas calculadas, enquanto as outras ainda não sofreram alteração.

Em cada iteração, realizamos a sequência de passos descrita na seção 3.2. Em primeiro lugar, aplicamos as forças externas ao fluido. No nosso caso, a única força externa agindo sobre a fumaça é a força de direcionamento definida com base na Transformada da Distância, que é a responsável por guiar o processo de *morphing*. Após a aplicação da força, realizamos a advecção semi-lagrangiana da velocidade e da densidade, conforme explicado na seção 3.3. O código para a advecção da velocidade pode ser visto na listagem abaixo.

Listagem 1: Advecção da velocidade [5]

```
double x = i - dt0*cell->getVelocity()[0];
double y = j - dt0*cell->getVelocity()[1];

if(x < 0.5) x = 0.5;
if(x > gridSize+0.5) x = gridSize + 0.5;

int i0 = static_cast<int>(x);
int i1 = i0 + 1;

if(y < 0.5) y = 0.5;
if(y > gridSize+0.5) y = gridSize + 0.5;

int j0 = static_cast<int>(y);
int j1 = j0 + 1;

double s1 = x - i0;
double s0 = 1 - s1;
double t1 = y - j0;
double t0 = 1 - t1;

Vector2D v = (neighbor1->getVelocity()*t0 +
              neighbor2->getVelocity()*t1) * s0 +
             (neighbor3->getVelocity()*t0 +
              neighbor4->getVelocity()*t1) * s1;
```

Após a advecção, resta apenas difundir o fluido e aplicar a operação de projeção. A difusão é inteiramente opcional, sendo seu principal objetivo conferir maior fluidez e naturalidade ao movimento da fumaça. Já a projeção precisa ser executada sempre após a advecção, pelos motivos explicitados na seção 3.5.

5.5 *Morphing*

As outras duas grades criadas durante a inicialização do programa são essenciais no processo de *morphing*. Uma delas, do tipo `SOURCE_GRID`, é utilizada para armazenar os dados da imagem de origem. É esta grade que serve como fonte de dados para a grade de permuta, usada como entrada para a primeira iteração da simulação. A outra grade, do tipo `TARGET_GRID`, possui os dados relativos à imagem-alvo. Nenhuma dessas duas grades sofrem alterações durante o processo de animação da fumaça.

Os dados das imagens, tanto para a grade de origem quanto para a grade-alvo, são obtidos através de uma conversão de pixels para valores de densidade. As duas

imagens são fornecidas à aplicação via linha de comando e são acessadas através do módulo de manipulação de imagens do Qt. As imagens são previamente formatadas para atender aos requisitos do *software*. No nosso caso, definimos uma cor específica para ser considerada na imagem. Qualquer outra cor fora do valor RGB especificado não é assumida como sendo parte integrante do alvo. Escolhendo o vermelho puro (R=255, G=0, B=0), por exemplo, nós selecionamos arquivos de imagem nesta cor e, ao acessá-los através da aplicação, todos os pontos vermelhos nessa imagem são representados pelo valor 1.0 na grade, que significa a presença de fumaça. O valor 1.0 é o maior valor possível para a densidade em uma célula. Além disso, é feita uma escala para que haja uma correspondência entre as posições dos pixels e as células da grade.

Logo após a interpretação dos arquivos de entrada das imagens, é realizado o cálculo da Transformada da Distância. Os valores da transformada para cada célula são armazenados na grade, não sendo necessário repetir estes cálculos durante o processo de simulação. O valor da transformada é usado para calcular a força de direcionamento do *morphing*, como explicado anteriormente.

Ao final de cada iteração, é realizada uma correção nos níveis de densidade, conforme explicado na seção 4.4. O trecho de código que realiza essa correção pode ser visto na listagem a seguir.

Listagem 2: Correção da densidade

```
double initial_relation = sum_final / sum_initial;
double present_relation = sum_final / sum_present;
double coeff = (present_relation-initial_relation)
             / (1.0-initial_relation);
if(coeff < 0.0) coeff = 0.0;
if(coeff > 1.0) coeff = 1.0;

double correction_factor = (coeff * sum_final
                         + (1.0-coeff) * sum_initial)
                         / (sum_present);

for(int i = 1; i < (N+1); i++)
{
    for(int j = 1; j < (N+1); j++)
    {
        GridCell *outCell = getCell(i, j);
        double d = outCell->getDensity();
        outCell->setDensity(d * correction_factor);
    }
```

```
}
```

5.6 Renderização

A visualização da animação gerada foi realizada via OpenGL. A princípio, utilizamos um esquema simples de renderização aproveitado do código de [5], que consiste em desenhar objetos OpenGL do tipo `QUAD`, usando como vértices a célula atual e mais três vizinhos. A animação resultante é bastante satisfatória, porém, à medida que o alvo se torna mais claro, a velocidade da fumaça diminui e a imagem final começa a se estabilizar, gerando um efeito de serrilhado indesejado.

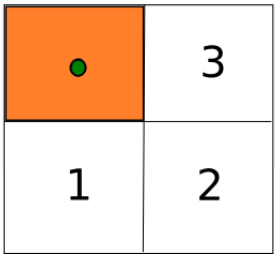

Figura 5.3: Células usadas no esquema simplificado de renderização utilizado

Como indicado previamente na seção 4.9, utilizamos procedimentos *anti-aliasing* no intuito de melhorar o resultado final. Empregamos dois métodos distintos. Um deles consiste na simples utilização do dual da grade para a formação dos quadrados. A outra solução foi a implementação do filtro de Lanczos. Ambas as alternativas resultaram em uma imagem final mais suavizada. As listagens com os trechos de código dos dois métodos podem ser vistas abaixo.

Listagem 3: Renderização Dual

```
double dmm = grd->getCell(i-1, j-1)->getDensity();
double dm0 = grd->getCell(i-1, j)->getDensity();
double dm1 = grd->getCell(i-1, j+1)->getDensity();

double d0m = grd->getCell(i, j-1)->getDensity();
double d00 = grd->getCell(i, j)->getDensity();
double d01 = grd->getCell(i, j+1)->getDensity();

double d1m = grd->getCell(i+1, j-1)->getDensity();
double d10 = grd->getCell(i+1, j)->getDensity();
double d11 = grd->getCell(i+1, j+1)->getDensity();
```

```
double a = (d00 + d10 + d1m + d0m) * 0.25;
double b = (d00 + d01 + d10 + d11) * 0.25;
double c = (d00 + d01 + dm1 + dm0) * 0.25;
double d = (d00 + dm0 + dmm + d0m) * 0.25;

glColor3f(d, d, d);
glVertex2f(x, y);
glColor3f(c, c, c);
glVertex2f(x, y+h);
glColor3f(b, b, b);
glVertex2f(x+h, y+h);
glColor3f(a, a, a);
glVertex2f(x+h, y);
```

Listagem 4: Filtro de Lanczos

```
double Rendering::lanczosKernel(int i, int j)
{
    const double PI = 3.14159265;
    double a = 2.0;
    double x = cell[i][j]->getDensity();

    double num_x = a * std::sin(PI*(i-0.5)) * sin(PI*(i-0.5)/a);
    double den_x = PI*PI * (i-0.5)*(i-0.5);
    double num_y = a * std::sin(PI*(j-0.5)) * sin(PI*(j-0.5)/a);
    double den_y = PI*PI * (j-0.5)*(j-0.5);

    double L = (num_x*num_y) / (den_x*den_y);
    return L;
}

void Rendering::lanczosResampling()
{
    (...)

    for(int i = 1; i < N; i++)
    {
        for(int j = 1; j < N; j++)
        {
```

```
            double dSum = 0, kSum = 0;
            for(int k = -1; k <= 2; k++)
            {
                for(int l = -1; l <=2; l++)
                {
                    double m = lanczosKernel(k, l);
                    dSum += cell[i+k][j+l]->getDensity() * m;
                    kSum += m;
                }
            }
            if(kSum <= 0.0) kSum = 1.0;
            array[i][j] = dSum / kSum;
        }
    }

    for(int i = 0; i < N+2; i++)
    {
        for(int j = 0; j < N+2; j++)
        {
            double d00 = array[i][j];
            double d01 = array[i][j+1];
            double d10 = array[i+1][j];
            double d11 = array[i+1][j+1];

            glColor3f(d00, d00, d00);
            glVertex2f(x, y);
            glColor3f(d01, d01, d01);
            glVertex2f(x, y+h);
            glColor3f(d11, d11, d11);
            glVertex2f(x+h, y+h);
            glColor3f(d10, d10, d10);
            glVertex2f(x+h, y);
        }
    }
}
```

5.7 Testes e resultados

Uma série de testes foram realizados com o *software* buscando validar os métodos e abordagens apresentados nos capítulos anteriores. Os resultados obtidos foram capturados em um computador com 2GB de memória RAM e um processador AMD Turion 64 X2 1.9GHz. O sistema operacional utilizado foi o Ubuntu 9.04 e o programa foi compilado com o gcc.

Para comparar o tempo de execução para diferentes níveis de resolução da grade, utilizamos o exemplo da transformação da letra grega Ψ na letra grega Ω. Testamos o *software* com grades de resolução 64, 128 e 254, com e sem a aplicação da difusão. Nota-se que, com a difusão fazendo parte do processo de simulação, o tempo de execução para cada quadro aumenta em torno de 3.7 vezes. Esse fator pode, no entanto, ser reduzido através da alteração do número de iterações realizadas no método de difusão. Na execução que gerou os dados do gráfico, dentro de cada passo do simulador 20 iterações da difusão foram realizadas para cada célula da grade. Quanto maior o número de iterações, mais suavizado se torna o processo. Contudo, em testes onde o número de iterações de difusão era pequeno, os resultados obtidos podem ser considerados tão satisfatórios quanto os conseguidos quando esse número de iterações era maior, além de exibirem uma nítida vantagem em termos de performance.

Além disso, pode-se observar que com a difusão, embora o processo de transformação como um todo pareça mais natural, o estado final - isto é, o instante em que o movimento cessa - demora mais para ser atingido, muitas vezes sobrando restos de fumaça nas proximidades das bordas do objeto-alvo, como pode ser observado na figura 5.4.

Podemos observar também nas tabelas contendo os dados de execução do programa que as simulações com difusão tendem a convergir em menos iterações do que as que não aplicam o processo difusivo, atingindo uma situação em que o alvo, em termos visuais, está perfeitamente representado. Observamos que isso não contradiz o que foi dito no parágrafo anterior acerca da demora de convergência utilizando a difusão, pois nos referíamos ao estado **final** do processo, ou seja, ao ponto da simulação no qual o movimento da fumaça cessa. Os dados de execução mostram que os estágios **intermediários** evoluem de maneira mais rápida, tornando mais rápida a convergência para uma configuração que, para o observador, é equivalente ao alvo. Entretanto, a convergência do mapeamento das densidades para a configuração referente ao alvo pode até não ocorrer completamente, pelo menos dentro dos limites de tempo estabelecidos. Essa convergência, ao contrário, ocorre sem problemas nas simulações nas quais apenas a advecção e a projeção são aplicadas, em que o estado final é mais bem definido. Trata-se mais de uma questão de condição de parada,

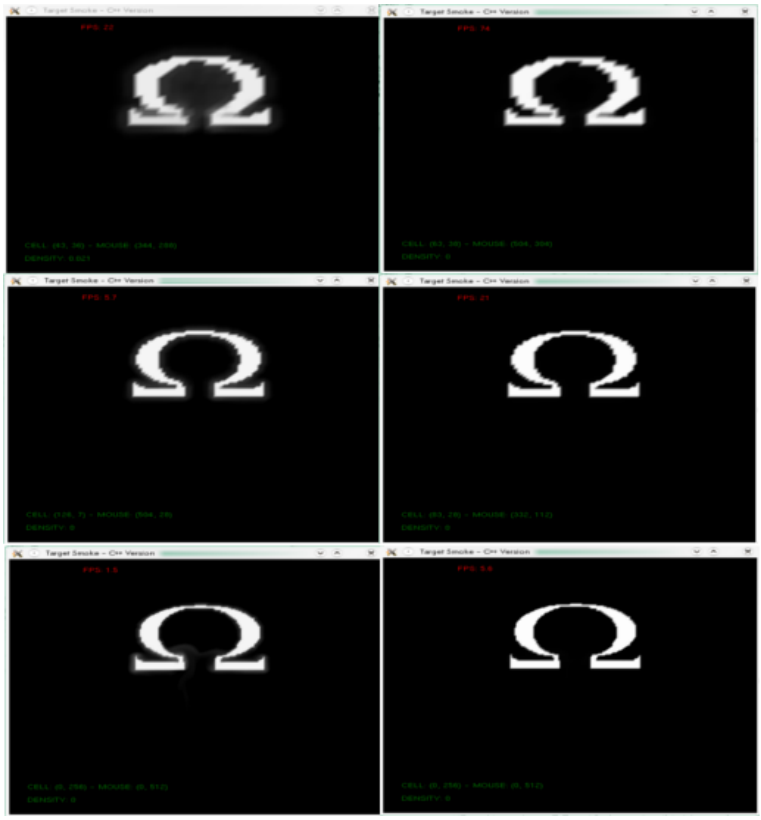

Figura 5.4: Estado final da transformação da figura 5.10 com grades de resolução 64, 128 e 256. As imagens à esquerda sofreram difusão, enquanto as da direita não.

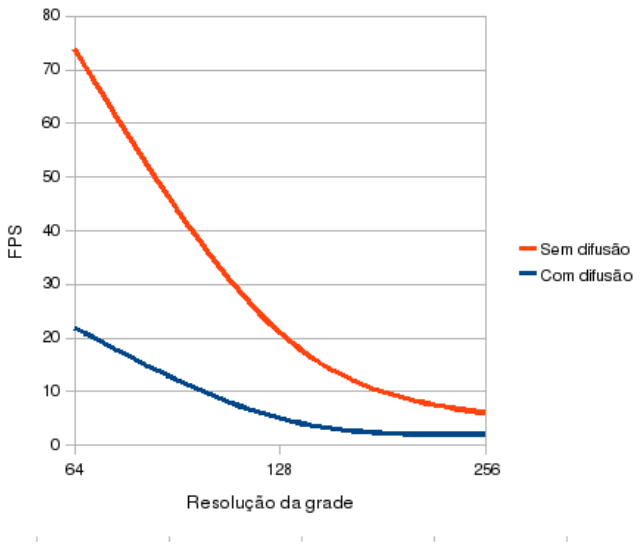

Figura 5.5: Gráfico que mostra a relação entre a taxa de quadros por segundo e a resolução da grade.

uma vez que visualmente ambos os contextos de simulação conseguem atingir seu objetivo.

Tabela 5.1: Dados de execução sem difusão

Exemplo	Intervalo de tempo	Número de iterações
Psi-Omega	0.2	118
	0.05	232
YinYang	0.2	109
	0.05	161
LCG	0.2	237
	0.05	209
QuadCircle	0.2	178
	0.05	341

Tabela 5.2: Dados de execução com difusão

Exemplo	Intervalo de tempo	Número de iterações
Psi-Omega	0.2	96
	0.05	167
YinYang	0.2	89
	0.05	146
LCG	0.2	117
	0.05	133
QuadCircle	0.2	192
	0.05	233

Além do exemplo das letras gregas, utilizado por [2], utilizamos uma série de outras imagens para realizar o *morphing*. Os primeiros testes foram realizados com formas simples. Inicialmente, realizamos a transformação de um círculo em um quadrado, em duas situações diferentes. Em uma delas, o círculo e o quadrado compartilham o mesmo espaço, enquanto que, em um outro cenário, há um quadrado em um canto da tela que se dilui para formar um círculo no canto oposto. Em ambos os casos obtivemos uma transição suave com um aspecto interessante, especialmente no caso das imagens separadas espacialmente, em que pode-se observar, no meio do processo, a mescla entre as duas formas. A massa de fumaça, à medida que se aproxima do alvo (o círculo), começa a adquirir formas circulares, assemelhando-se a um cometa. As figuras 5.6 e 5.7 mostram a evolução da fumaça nesses dois exemplos.

Como explicado na seção 4.7, um dos desafios foi escavar buracos presentes em determinados alvos, depois que parte da densidade da fumaça tivesse adentrado o local onde deveria se formar o buraco. Para testar a solução implementada, realizamos o *morphing* do símbolo do Yin Yang, transformando-o no seu dual. O resultado esperado foi obtido, como pode-se conferir na figura 5.8.

Outra experiência realizada foi o *morphing* de uma imagem em uma palavra. Uma característica específica de uma palavra é que ela é formada por um conjunto de letras, que podem ser interpretadas como partes disjuntas de uma mesma imagem.

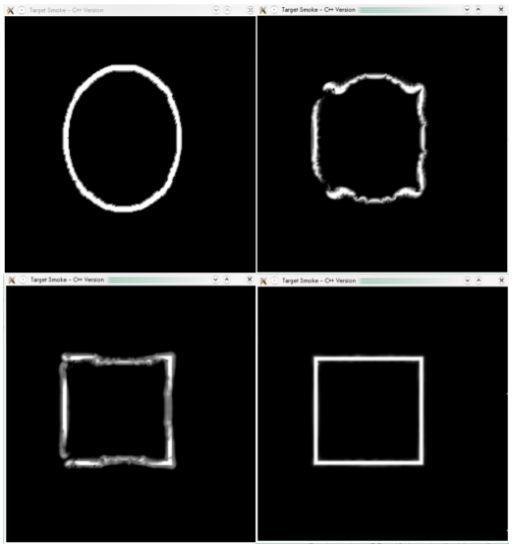

Figura 5.6: Círculo se transformando em um quadrado no centro.

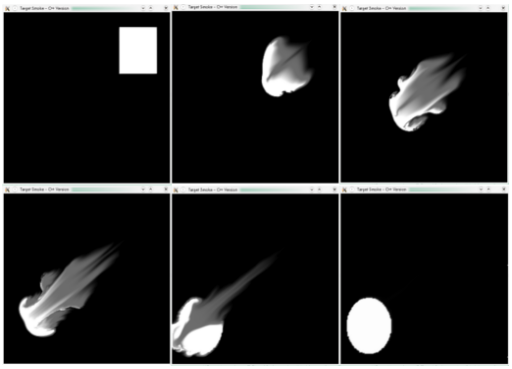

Figura 5.7: Quadrado saindo do canto superior direito e gerando um círculo no canto inferior esquerdo.

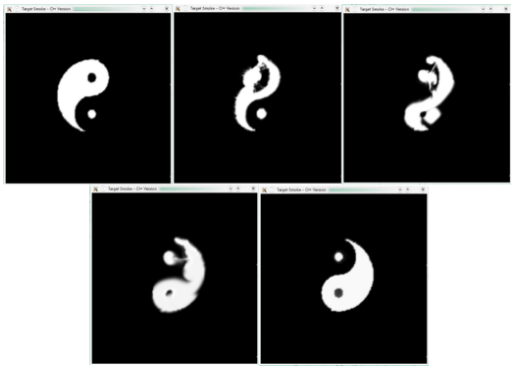

Figura 5.8: Transformação do símbolo Yin Yang.

Cada letra é como uma subimagem que precisa ser gerada para a formação da imagem total. Utilizamos novamente a letra grega Ω, porém dessa vez como a imagem de origem, para formar a sigla LCG, correspondente ao laboratório onde foi desenvolvido o trabalho. Os resultados podem ser vistos na figura 5.9.

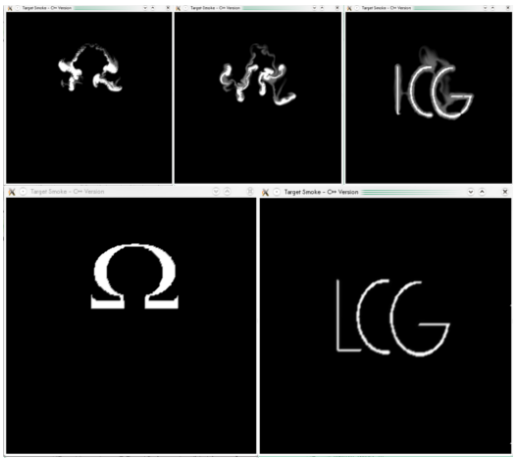

Figura 5.9: Geração da sigla LCG. As imagens superiores mostram os estados intermediários da transformação.

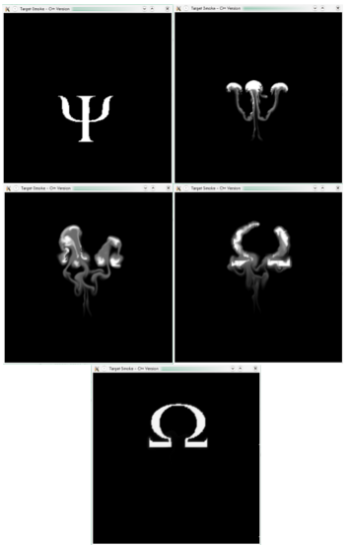

Figura 5.10: O *morphing* do Ψ para o Ω.

Capítulo 6

Conclusões

Este trabalho apresentou um modelo para o *morphing* entre duas imagens a partir de um processo de animação de fumaça. Os principais trabalhos utilizados no desenvolvimento desta dissertação foram [1], [4] e [2], sendo os dois primeiros utilizados na implementação das equações que regem a dinâmica dos fluidos e das estruturas de dados que dão suporte à simulação, enquanto o terceiro trabalho foi utilizado como base para o modelo de transformação de imagens proposto, visando - dentro do que é possível num contexto em que os objetos são constituídos por material difuso - melhorar o modelo original afim de ajustá-lo melhor às premissas básicas requeridas de um processo de *morphing*. As mudanças efetuadas começam por se empregar um gradiente aproximado de uma Transformada da Distância como componente principal da força responsável por garantir que a densidade de fumaça, inicialmente concentrada na figura inicial, receba o direcionamento adequado de forma a gerar a figura-alvo. Além dessa modificação foram introduzidos uma série de procedimentos que enfocam problemáticas específicas como a manutenção da velocidade da transformação, homogenizar a densidade do resultado final em componentes conexas distintas do alvo, respeitar o nível de detalhamento do alvo e a presença de buracos nele, obter uma solução final estável ao menos visualmente ou evitar o *aliasing* de contornos no fim do processo. Essas modificações foram descritas no capítulo 4.

Além disso, ao longo do trabalho, foram estudados e explicados os métodos envolvidos na simulação física de fluidos, assim como conceitos básicos necessários para a devida compreensão do assunto. As técnicas empregadas na literatura foram implementadas em *software* para dar validade à proposta do trabalho.

As modificações realizadas no mecanismo de transformação das figuras com relação a trabalhos existentes na área renderam resultados satisfatórios, como pôde ser observado no capítulo 5. Obtivemos, com a utilização da Transformada da Distância, um parâmetro constante para o cálculo da força de direcionamento, calculado apenas uma vez no início do processo. A transição entre as imagens, por sua vez, se realizou de forma suave, criando de fato a aparência de uma massa de fumaça

que se difunde a partir de uma imagem inicial e converge formando a imagem final. Diversos testes foram realizados, com diferentes tipos de imagens, para garantir que o método implementado atenda a variadas situações.

Possíveis trabalhos futuros em continuação ao que foi desenvolvido aqui incluem, certamente, a extensão da metodologia para a transformação de objetos 3D e obtenção de *morphings* de figuras coloridas. Com respeito a esse último caso, deve-se considerar que, transformando separadamente cada canal de cor, o processo de difusão pode, com grande probabilidade, determinar que, numa dada instância da transformação, algumas células tenham sido atingidas num canal e não atingidas no outro. Em consequência, as figuras intermediárias da transformação passam a apresentar em algumas partes cores decorrelacionadas com as das figuras inicial e final. Deve-se, portanto, empregar um único processo de difusão em que, por exemplo, a densidade seja associada ao brilho das imagens. Ainda assim evitar mesclas de cores cujo efeito visual não seja satisfatório é um problema que ainda requer investigação e, principalmente, muitos testes.

Além disso, há todo um conjunto de aspectos que ainda requerem aperfeiçoamento. Por exemplo, em alguns casos, os estados intermediários produziram uma massa amorfa, sendo que o ideal é que seja possível identificar neles características tanto do estado inicial quanto do final. Com relação à conclusão do processo, as baixas concentrações de densidade em regiões próximas ao alvo podem demorar muito a convergir para a região de interesse. Isso pode, em certas situações, impedir a simulação de atingir seu estado final. Por estado final devemos entender a completa migração da densidade da fumaça, uma vez que, embora a imagem-alvo sempre se forme, pequenas porções do fluido podem se manter fora do alvo, principalmente nos casos da difusão. Mesmo sendo quase imperceptíveis na maioria das vezes, o desejado é que essas porções restantes completem seu trajeto até o destino final.

Uma outra possibilidade ainda seria explorar outras técnicas para conferir à imagem-alvo já formada e aos estágios finais da transformação, como efeito especial, um aspecto mais fumegante. Em alguns casos, pode parecer que a imagem não é composta por uma massa de fumaça, em aplicações em que esse é, exatamente, o objetivo de se ter optado por essa metodologia para fazer o *morphing*. Uma possível saída seria introduzir um ruído no sistema que mantenha, de forma sutil, a fumaça em movimento.

Por fim, poderíamos utilizar outros métodos de renderização, visando suavizar ainda mais o efeito de *aliasing* citado na seção 4.9 e produzir, dessa forma, um resultado final mais interessante e realista.

Referências Bibliográficas

[1] STAM, J. "Stable fluids". In: *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN: 0-201-48560-5. doi: http://doi.acm.org/10.1145/311535.311548.

[2] FATTAL, R., LISCHINSKI, D. "Target-driven smoke animation". In: *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pp. 441–448, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1186562.1015743.

[3] FOSTER, N., METAXAS, D. "Modeling the motion of a hot, turbulent gas". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 181–188. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997.

[4] FEDKIW, R., STAM, J., JENSEN, H. W. "Visual simulation of smoke". In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 15–22, New York, NY, USA, 2001. ACM. ISBN: 1-58113-374-X. doi: http://doi.acm.org/10.1145/383259.383260.

[5] STAM, J. "Real-time fluid dynamics for games". In: *Proceedings of the Game Developer Conference*, v. 18. Citeseer, 2003.

[6] TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., et al. "Keyframe control of smoke simulations". In: *ACM SIGGRAPH 2003 Papers*, p. 723. ACM, 2003.

[7] FERZIGER, J. H., PERIĆ, M. *Computational methods for fluid dynamics*. Springer, 2002. ISBN: 3-540-42074-6.

[8] FOX, R. W., MCDONALD, A. T., PRITCHARD, P. J. *Introduction to fluid mechanics*. John Wiley & Sons, 2004. ISBN: 0-471-20231-2.

[9] EMANUEL, G. *Analytical Fluid Dynamics*. CRC Press LLC, 2001. ISBN: 0-8493-9114-8.

[10] TROTTENBERG, U., OOSTERLEE, C. W., SCHÜLLER, A. *Multigrid*. Academic Press, 2001. ISBN: 0-127-01070-X.

[11] DA SILVA NETO, A. A., RODRIGUES, P. S. S., GIRALDI, G. A., et al. "Animação computacional de fluidos via smoothed particle hydrodynamics". In: *SIBGRAPI '05: Digital Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, 2005.

[12] MÜLLER, M., CHARYPAR, D., GROSS, M. "Particle-based fluid simulation for interactive applications". In: *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN: 1-58113-659-5.

[13] FEDKIW, R., LOSASSO, F., TALTON, J., et al. "Two-Way Coupled SPH and Particle Level Set Fluid Simulation", *IEEE Transactions on Visualization and Computer Graphics*, v. 14, n. 4, pp. 797–804, 2008. ISSN: 1077-2626. doi: http://dx.doi.org/10.1109/TVCG.2008.37.

[14] PRESS, W., FLANNERY, B., TEUKOLSKY, S., et al. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 10 1992. ISBN: 0521431085.

[15] STEINHOFF, J., UNDERHILL, D. "Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings", *Physics of Fluids*, v. 6, n. 8, pp. 2738–2744, 1994.

[16] SWARZTRAUBER, P., SWEET, R. "Efficient FORTRAN subprograms for the solution of elliptic partial differential equations", *NCAR Technical Note-TN/IA-109*, 1975.

[17] XU, C., PRINCE, J. "Generalized gradient vector flow external forces for active contours", *Signal Processing*, v. 71, n. 2, pp. 131–139, 1998.

[18] SETHIAN, J. "Fast marching methods", *SIAM review*, v. 41, n. 2, pp. 199–235, 1999.

[19] PARENT, R. *Computer animation: algorithms and techniques*. The Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann, 2002. ISBN: 1558605797.

[20] DUCHON, C. "Lanczos filtering in one and two dimensions", *Journal of Applied Meteorology*, v. 18, n. 8, pp. 1016–1022, 1979.

[21] BATCHELOR, G. K. *An introduction to fluid dynamics*. Cambridge University Press, 2000. ISBN: 0-521-66396-2.