INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

Deformação e Metamorfose de Imagens usando Simulação de Fluidos

Dalia Melissa Bonilla Correa

Orientador: Luiz Carlos Pacheco Rodrigues Velho

Rio de Janeiro, Junho de 2011.

Agradecimentos

Meus agradecimentos ao meu orientador, que sempre confiou em mim. Cheio de entusiasmo me incentivou muito desde o início, e não desespero pelo meu ritmo lento. Professor muito obrigada, não só pelo exemplo acadêmico, mas pelo exemplo de vida.

Também Agradeço:

A meus primeiros orientadores na época de mestrado, André Nachbin e Felipe Linares, que guiaram-me com suas idéias, conceitos e conselhos. Eles me mostraram uma visão clara das coisas. Muito obrigada.

Agradeço aos professores que estavam presentes na minha defesa de tese, Luiz Henrique de Figueiredo, Diego Nehab, Paulo Cezar Carvalho, André Nachbin, Luis Gustavo Nonato, Hélio Lopes e meu orientador Luiz Velho. Agradeço todas as sugestões e correções feitas nesta tese.

Agradeço a minha família, minha mãe Nunila, meu pai Mariano e meu irmão Mario, que me apoiaram incondicionalmente todo esse tempo na distância. Meus primos e tios que sinto muita falta.

Gerardo e Alicia, que sem dúvida me apoiaram e ajudaram muito. Devo muito a vocês, até com juros. Muito obrigada.

Meu principal apoio, ‟mi gordo Juan″.

Para os amigos que fiz aqui, certamente amigos pelo resto de minha vida.

Minha turma de mestrado.

Ao pessoal do Visgraf. Em especial Ives, Emilio, Anderson, Maria e Leo Carvalho. (Falta gente eu sei, desculpem, a lista é grande e tal vez posso deixar alguém por fora.)

Ao IMPA, que me recebeu com muito carinho com um sorriso de bom dia. Com certeza vai ser uma bonita lembrança.

Ao CNPq e CAPES pelo apoio destes anos.

A Brasil.

Abstract

In this thesis we present a technique of image deformation using dynamic of fluids. We call this technique fluid warping. This technique was present in first time by Jos Stam in Stable Fluids [38], as one aplication of fluid simulation. The main advantage of this approach is the effect is obtained in the deformation of image, like a liquid image.

The key idea is to think of the image domain as a two-dimensional incompressible fluid, and to use the Navier- Stokes equations to model the fluid. We deform the image through a vector field generated by equations.

The challenge now is to control the deformation. For this purpose, we present three techniques for controlling the fluid warping: The first technique uses a constant viscosity and forces as parameters control. The second technique againg uses viscosity as a control tool, but in this case it is variable in space and defined through of a image. Finally, the third technique, more complex than the previous ones, need three processes.

- Mark deformation goal by keyframes,
- employ the adjoint method to calculate derivatives
- and use the derivatives in a optimization process to find the necessary forces to achieve the goal of deformation.

Resumo

Nesta tese apresentamos uma técnica de deformação de imagens usando dinâmica dos fluidos. Técnica que chamaremos de fluid warping. A principal vantagem desta é o efeito obtido na deformação.

No fluid warping, consideramos um fluido 2D (homogêneo e incompressível) sobre o domínio da imagem e deformamos a imagem a través do campo de velocidades do fluido. As velocidades são modeladas pelas equações de Navier-Stokes.

Para chegar até uma deformação desejada, devemos controlar a deformação. Para esse fim, são apresentadas três técnicas para controlar o fluid warping. A primeira usa a viscosidade constante e forças como parâmetros de controle. A segunda usa de novo a viscosidade como ferramenta de controle, mas neste caso é variável no espaço e definida a través de uma imagem. Finalmente a terceira técnica, mais complexa que as anteriores, precisa de três processos. Marcar o objetivo de deformação por keyframes, por meio do método adjunto calcular derivadas e utilizar as derivadas num processo de otimização para achar as forças necessárias para atingir o objetivo de deformação. (Veremos a eficacia do método adjunto sobre outras técnicas para calcular derivadas).

Para todas as técnicas de controle temos que, um pequeno conjunto de parâmetros fornecem mecanismos poderosos e intuitivos para controlar a deformação de imagens.

Sumário

1 Introdução

A dinâmica dos fluidos se tornou um tópico de interesse nas pesquisas de computação gráfica nos últimos anos e foram obtidos surpreendentes resultados. A simulação de fluidos ficou mais realista e atingiu uma forma especifica controlando seu movimento. O warping de imagens faz parte importante nos fundamentos da computação gráfica, é usado para remover ou produzir distorções na imagem, com objetivos artísticos e para efeitos especiais na indústria do entretenimento. Até hoje não existe nenhuma técnica com estudos profundos baseados em física. Nesta tese propomos uma técnica de warping de imagens usando dinâmica dos fluidos.

1.1 Breve Descrição da Técnica

O warping de imagens usando dinâmica dos fluidos tem como ideia pensar no domínio da função imagem como um fluido 2D e usar as equações de Navier-Stokes, que descrevem o movimento do fluido, para modifica-la aplicando forças sobre o domínio. O processo transporta as coordenadas da parametrização da imagem através do campo vetorial gerado pelas equações transformando a imagem. O warping é controlado pelas propriedades físicas tais como viscosidade associada aos valores da imagem ou de outras imagens e forças aplicadas sobre o domínio da imagem.

1.2 Motivação e Vantagens

A principal vantagem deste enfoque é o efeito obtido. Considerar o domínio da imagem como um fluido 2D produz o efeito na deformação da imagem de um processo físico, tornando interessante o warping. A técnica mostra bons resultados e apresenta o grande potencial de transformações. Este trabalho apresenta técnicas de controle especializadas de fluidos. Umas de controles específicos e outras de controle local usando propriedades físicas obtendo bons resultados para efeitos desejados.

1.3 Contribuições

Propomos um novo enfoque para warping de imagens usando dinâmica dos fluidos, atingindo objetivos específicos com diferentes técnicas de controle. As contribuições desta tese são:

1. A caracterização do domínio de uma imagem como um fluido para gerar uma deformação contínua, e controle desta.
2. Um método de controle de warping usando fluidos, e propriedades físicas associadas à imagem.
3. Usamos um método numérico de otimização, e a técnica de controle de warping utiliza, pela primeira vez em deformação de imagens, o método adjunto.
4. O controle baseado em transportar partículas através do fluido até uma posição específica. Usando para isto uma função objetivo que trabalha de forma natural e eficiente.
5. Uma aplicação das técnicas de controle ao morphing de imagens.

As duas primeiras contribuições estão reportadas em (Bonilla,Velho, Nachbin e Nonato, 2009) [10].

1.4 Organização da Tese

A tese está organizada da seguinte forma.O capítulo 1 introduz a ideia da técnica de warping e mostra a importância e vantagens do trabalho.
O capítulo 2 faz uma revisão do histórico dos trabalhos em computação gráfica: warping, fluidos, processamento de imagens e o método adjunto.
O capítulo 3 faz uma introdução teórica aos tópicos e técnicas que são necessários para desenvolver as técnicas de controle de warping de imagem usando fluidos.
O capítulo 4 mostra as técnicas desenvolvidas para controlar a deformação de imagens utilizando os resultados do capítulo anterior. Este capítulo está dividido em três partes: a primeira descreve cada técnica de forma geral, a segunda mostra os parâmetros de entrada das técnicas e finalmente a terceira é uma descrição teórica detalhada do controle de warping.
O capítulo 5 apresenta e descreve os resultados obtidos através de cada uma das técnicas.
O capítulo 6 finaliza o trabalho com um resumo das principais ideias da tese e os pontos que discutem os trabalhos futuros desta pesquisa.

2 Trabalhos Relacionados

2.1 Warping

Para o warping de imagens há muitos métodos. Estes podem ser classificados como: baseados em parâmetros, baseados em características, de forma livre e híbridos. Métodos baseados em parâmetros são técnicas de warping controladas por uma família de transformações tais como escalar, torcer e dobrar. Este tipo de técnica foi introduzida em computação gráfica por Alan Barr em 1984 [6]. Métodos baseados em características abrangem toda uma classe de técnicas de deformação, que vão do tipo de características geométricas até funções de reconstrução. Neste método a correspondência entre as características entre os objetos de fonte e os de destino deve ser dada pelo usuário. As funções de reconstrução típicas incluem a interpolação de dados dispersos, base radial etc. [4]. Os métodos baseados em forma livre usam especificação por sistemas de coordenadas. Para isso, empregam curvas de forma livre (B-splines, Bézier etc.) para definir as curvas coordenadas [9]. Estas técnicas foram introduzidas por Smith em 1987 [36], desenvolvidas por Smithe em 1990 [37] e descritas por Wolberg em 1990 [48].

2.2 Fluidos e Controle de Fluidos

Na simulação de fluidos, Foster e Metaxas modelaram água em 1996 [19] e gás em 1997 [20], usando as equações de Navier-Stokes. Produzindo uma simulação de fluido convincente em grades relativamente grosas, eles requereram entretanto um pequeno passo do tempo para que suas simulações permaneceram estáveis, resultando então num tempo de cálculo grande. O problema foi

usar o esquemas de diferenças finitas explícitas para resolver numericamente as equações, tornando-se instáveis para grandes passos de tempo e para pequenos passos de tempo o cálculo era lento [38]. Assim em 1999 Jos Stam introduz o algoritmo Stable Fluids [38] onde trata as limitações anteriores com uma combinação do método semi-Lagrangiano para resolver a advecção [14], um passo de projeção para assegurar a incompressibilidade [13] e uma solução implícita para a viscosidade. O resultado é um método rápido e incondicionalmente estável para simular a velocidade de um fluido, embora com muita dissipação numérica. Em 2001, Fedkiw [17] estende esta aproximação com confinamento de vorticidade contrapondo assim a dissipação numérica, para simulação de fumaça. No mesmo ano Foster e Fedkiw [18] e logo em 2002 Enright [15], trataram a perda de massa dada pelo método semi-Lagrangiano no passo da advecção usando uma combinação de partículas com o método de curvas de nível. Outra vertente de simulação de fluidos surgiu, trabalhos fundados puramente sobre partículas, baseados no método (SPH), do inglês Smoothed Particle Hydrodynamics. Alguns exemplos destes trabalhos são: Müller 2003 [32]; Zhu e Bridson 2005 [50]; Keiser 2005 [27]; Adams 2007 [3]. Outras pesquisas foram focalizadas em simulações de fluidos visco-plásticos (Bargteil 2007 [5]) e viscoelásticos (Wojtan e Turk em 2008 [46]), interação de fluidos com corpos rígidos (Carlson em 2004 [11]; Kwatra em 2010 [28]) e controle da simulação de fluido para atingir comportamentos ou formas especificas (Treuille 2003 [44]; McNamara 2004 [30]; e Fattal em 2004 [16]). Em 2003, Treuille [44] propôs o primeiro sistema para controlar simulação de fluidos usando keyframe e otimização não linear, embora o cálculo do gradiente da função objetivo usado era ineficiente ou muito lento. Em 2004, McNamara [30] introduz a técnica do Método Adjunto onde o cálculo do gradiente é muito eficiente e melhora a velocidade da otimização. Em 2006, o trabalho de Chris Wojtan [45] utiliza de novo o Método Adjunto, mas esta vez para simulações de sistemas de partículas, novamente com interessantes resultados. Na literatura de computação gráfica existem então só dois trabalhos que usam o Método Adjunto, [30, 45] e agora o nosso.

2.3 Método Adjunto

A teoria sobre o Método Adjunto é baseada nos artigos de Giles e Pierce de 2000 [22], Giering e Kaminski de 1998 [21] e Stam de 2004 [40]. Embora Giles e Pierce [22] não falem do Método Adjunto especificamente, mostram um enfoque adjunto contínuo e discreto e discutem vantagens e desvantagens de cada enfoque. Para o caso do enfoque adjunto contínuo as vantagens são a fácil interpretação física das variáveis adjuntas e um programa adjunto simples que requer menos memória. Para o caso do enfoque adjunto discreto, o cálculo do gradiente da função objetivo discreta é exato e a criação do programa é direta. Em qualquer caso o programa adjunto é criado depois de serem formuladas as equações adjuntas. Para Gering e Kaminski [21] o programa adjunto é criado do programa original diretamente sem formular as equações adjuntas e o cálculo do valor do gradiente é rápido e de baixo custo computacional. Nesse artigo se definem vários conceitos em forma clara: O modo inverso, Método Adjunto, código adjunto, entre outros. Também mostra as 'receitas' para a criação do código adjunto. Agora Stam em [40] mostra um exemplo do código adjunto e uma visão de álgebra linear do Método Adjunto.

2.4 Dinâmica dos Fluidos e Processamento de Imagens

O primeiro trabalho que usou dinâmica dos fluidos em processamento de imagens foi o trabalho de Bertalmio em 2001 [8] que introduziu a ideia para fazer inpainting na reconstrução de uma imagem, embora sua ideia tenha sido usar as equações de Navier-Stokes. Ele pensou na intensidade da imagem como uma função corrente e o laplaciano da intensidade da imagem como a vorticidade do fluido transportado através do campo vetorial criado pela função corrente dentro da região de inpainting. Em 2009, Bonilla et al. [10] apresentam a técnica *fluid warping*, a qual é simples e usa o campo vetorial criado pelas equações de Navier-Stokes diretamente, não usa vorticidade nem a função corrente, e transporta as coordenadas da imagem através de esse campo. O warping é controlado por viscosidade e forças associadas a características da

imagem ou de outras imagens auxiliares; embora esse controle proporcione interessantes resultados, o controle não é preciso. Apresentamos então no presente trabalho um controle melhorado da técnica *fluid warping* usando o Método Adjunto.

3 Introdução Teórica

3.1 Warping

Uma imagem é uma aplicação $f : U \to C$, onde $U \subset \mathbb{R}^2$, C é um espaço vetorial qualquer que, em geral, contém o espaço de cor como subespaço. A função f é chamada de *função imagem*. Quando C é um espaço de cor de dimensão 1, dizemos que a imagem é *monocromática* [24].

O processo que deforma a figura dos objetos em uma imagem é chamado de *warping*. O uso da deformação desempenha um papel importante em muitas aplicações, desde a correção de distorções em imagens em dados médicos, na criação de efeitos especiais da indústria do entretenimento.

Dada uma imagem, $f : U \subset \mathbb{R}^2 \to C$, o mapeamento entre o espaço de origem (u,v) e o espaço de destino (x,y) é chamado de *filtro de warping*. Esse mapeamento, $W(f) = g$, age sobre a imagem de entrada $f(u,v)$, dando origem a uma imagem de saída $g(x,y)$ e pode ser considerado como uma deformação do domínio da imagem. O filtro de warping é definido por um homeomorfismo[1] $h : U \subset \mathbb{R}^2 \to V \subset \mathbb{R}^2$. A imagem filtrada é obtida por $g = f \circ h^{-1} : V \to C$ ver a figura 3.1.

3.1.1 Morphing

Uma transformação de uma imagem que combina simultaneamente transformação do domínio (warping) com transformação de amplitude (cor) é chamada de *morphing*. Os *filtros de amplitude* permitem fazer alteração na cor de uma

[1] A aplicação h é um homeomorfismo, isto é, uma aplicação bijetiva e contínua com inversa h^{-1} contínua.

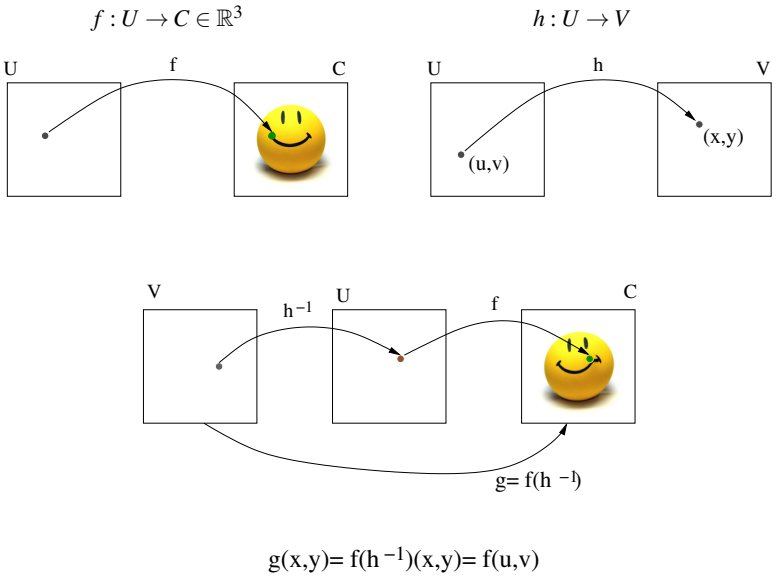


Figura 3.1: Filtro de Warping.

imagem. As transformações de morphing combinam mudanças de forma (filtros de warping) com mudanças na intensidade de cor (filtros de amplitude) epodemos obter efeitos de transição entre imagens. No diagrama abaixo combinamos as transformações de warping e de cor.

$$\begin{array}{ccc} U & \xrightarrow{f} & C \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle T} \\ U & \xrightarrow[g]{} & C \end{array}$$

A transformação de morphing M aplicada à imagem f resulta em uma imagem g, definida por

$$g = M(f) = T \circ f \circ h^{-1},$$

ou seja, a transformação h faz uma mudança de coordenadas do domínio (*warping*), e a transformação T de cor.

3.2 Simulação

Supomos que queremos simular numericamente e computacionalmente um fenômeno físico. O desenvolvimento de um programa de simulação numérica é

usualmente feito em 3 passos:

1. As equações diferenciais são formuladas. Elas constituem o *modelo matemático.*

2. Um esquema de discretização é escolhido e as equações diferencias discretas são construídas. Elas descrevem o *modelo da simulação.*

3. O ultimo passo é implementar um algoritmo que resolva as equações discretas numa linguagem de programação. A sequencia de instruções de computador ou *código* são o *programa da simulação.*

O estado q do sistema físico em algum tempo t consiste de um conjunto de dados do sistema como por exemplo: posições de partículas x e velocidades v, e outros. Assim temos que o estado q pode ser representado por

$$q(t) = (x(t), v(t)) = \{x_i(t), v_i(t) : i = 1, ..., N_p\}$$

onde $x_i(t)$, $v_i(t) \in \mathbb{R}^2$ e N_p é o numero de partículas.

A simulação computacional define o sistema em alguma região finita do espaço. No nosso caso necessitamos uma representação finita para o fluido. Dividimos então o espaço em células quadradas idênticas, o fluido é modelado sobre uma malha quadrada como é mostrado na figura 3.2 e é definida além uma borda de células ao redor do domínio para simplificar os cálculos na fronteira. O sistema é baseado no algoritmo Stable Fluids apresentado por [Stam 1999[38].] Um estado na simulação de fluidos, que transporta partículas, consiste de uma malha x de posições de partículas e uma malha v de vetores velocidades que definimos sobre o centro de cada célula do domínio. Assim a simulação é calculada a partir de um estado inicial q_0 e é avaliada aplicando a função S (que modela as equações de Navier-Stokes que descrevem o fluido). Logo, para um passo de tempo Δt, o estado da simulação é avaliado recursivamente por

$$q_{k+1} = S(q_k)$$

onde $q_k = (x(k\Delta t), v(k\Delta t))$.

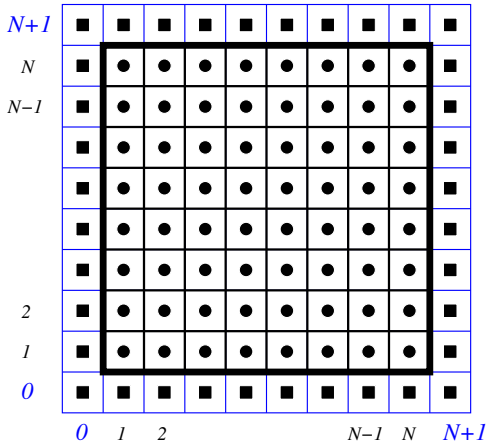


Figura 3.2: Domínio do fluido. Possui $(N+2)\times(N+2)$ células. A posição das partículas e a velocidade é definida sobre o centro de cada célula.

3.3 Fluidos

Um fluido é uma substância que se deforma continuamente quando é submetida a uma tensão de cisalhamento, não importando o quão pequena possa ser essa tensão. Uma tensão de cisalhamento é a componente da força tangente a uma superfície [41].

Os fluidos compartilham a propriedade de não resistir a deformação e apresentam a capacidade de fluir (também descrita como a habilidade de tomar a forma de seus recipientes). Estas propriedade são tipicamente em decorrência da sua incapacidade de suportar uma tensão de cisalhamento.[2] Os fluidos são divididos em líquidos e gases.

3.4 Modelagem Matemática

Nós já temos uma definição física de fluido e uma intuição do comportamento do mesmo. Agora, o próximo passo é definir alguns conceitos, a fim de modelar o movimento fluido. Tentamos não entrar em detalhes teóricos, apenas o suficiente para deduzir as equações clássicas de Navier-Stokes com viscosidade constante e as equações de Navier-Stokes com viscosidade variável no espaço. Estas equações são o coração das simulações deste trabalho, como veremos mais tarde. Para mais detalhes teóricos, recomendamos o leitor a ver

as seguintes referências bibliográficas [33], [13], [31].

Seja D uma região cheia de fluido com $D \subset \mathbb{R}^2$.(Como sugestão)

- Fixemos $\bar{x} = (x, y)$ um ponto de D e consideremos a partícula de fluido movendo-se a través de $\bar{x}$ no tempo t. Ver figura 3.3.
- Seja $u(\bar{x}, t) = (u_1(\bar{x}), u_2(\bar{x}))$ a velocidade da partícula do fluido movendo-se a través de $\bar{x}$ no tempo t.
- Para t fixo, u é o campo vetorial em D, tangente na trajetória da partícula.
- A função u é denominada o *campo de velocidade do fluido*.
- Para cada tempo t assumimos que o fluido tem bem definida a densidade de massa $\rho(\bar{x}, t)$. Assim se W é uma sub-região de D, a massa do fluido em W no instante t é dada por

$$m(W, t) = \int_W \rho(\bar{x}, t) dV$$

  onde dV é o elemento de volume no plano ou no espaço.
- O fluido é chamado de *incompressível* quando

$$\nabla \cdot u = 0.$$

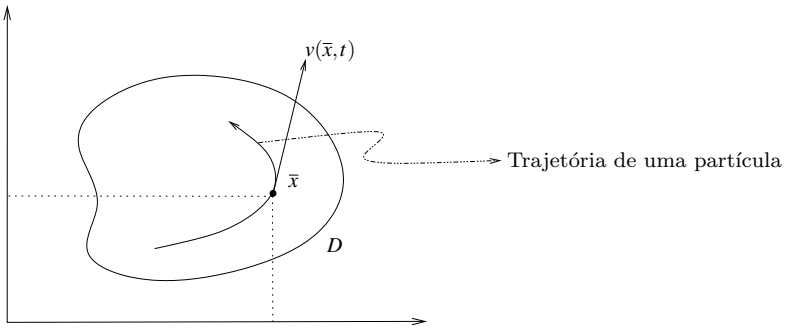


Figura 3.3: Trajetória

De agora em diante usaremos as seguintes notações. O operador Laplaciano é denotado por $\Delta = \nabla \cdot \nabla$, "$\cdot$" é o produto escalar entre vetores, $\nabla =$

$(\partial/\partial x, \partial/\partial y)$ é o vetor gradiente no espaço, ∂_t é derivada parcial no tempo $\partial/\partial t$, e $\nabla\cdot$ é o divergente.

O Momento (linear). O momento (linear) de uma porção de fluido que ocupe, no instante t, a região Ω_t é dado pela integral

$$\int_{\Omega_t} \rho(\bar{x},t)u(\bar{x},t)dX.$$

Conservação do Momento. Pela segunda lei de Newton temos que, a derivada em relação ao tempo do momento é igual a soma das forças externas que autuam no fluido mais as forças internas exercidas sobre Ω_t pelo resto do fluido.

$$\frac{d}{dt}\int_{\Omega_t} \rho(\bar{x},t)u(\bar{x},t)dX = \text{forças externas} + \text{forças internas}.$$

Forças externas. A função $f(x,t)$ denota as forças externas. A força total atuando na porção de fluido no tempo t na região Ω_t é

$$\int_{\Omega_t} \rho(\bar{x},t)f(\bar{x},t)dX.$$

Forças internas. Supomos as forças internas como forças de *contato* ou *tensões* e que podemos definir um campo de tensões $\tau(\bar{x},t,n)$. O campo τ dá a força de contato por unidade de área atuando tangencialmente numa superfície no ponto $\bar{x}$, no instante t. Mais precisamente, a força exercida pelo resto do fluido na porção de fluido em Ω_t é dada por

$$\int_{\partial\Omega_t} \tau(\bar{x},t,n)dA,$$

onde n denota o vetor unitário normal a $\partial\Omega_t$ apontando para fora. Um teorema de Cauchy garante que, se o fluido satisfaz a segunda lei de Newton, então τ depender linearmente de n. Mais precisamente, existe uma função matricial $S(\bar{x},t)$ tal que

$$\tau(\bar{x},t,n) = S(\bar{x},t)\cdot n.[13]$$

Portanto temos que

$$\frac{\partial}{\partial t}\int_{\Omega_t}\rho(\bar{x},t)u(\bar{x},t)dX = \int_{\Omega_t}\rho(\bar{x},t)f(\bar{x},t)dX + \int_{\partial\Omega_t}S(\bar{x},t)\cdot ndA.$$

Pelo Teorema do Transporte[2] e pelo Teorema da Divergência[3] temos que

$$\int_{\Omega_t}\rho(\bar{x},t)\frac{Du}{Dt}(\bar{x},t)dX = \int_{\Omega_t}\rho(\bar{x},t)f(\bar{x},t)dX + \int_{\Omega_t}\mathrm{div}S(\bar{x},t)dX. \tag{3.1}$$

Acima $\frac{D}{Dt} = \frac{d}{dt} + u\cdot\nabla$ é o operador *derivada material,* que para qualquer função $f(\bar{x},t)$ é definido como $\frac{Df}{Dt} = \frac{df}{dt} + u\cdot\nabla f$. Em particular para a velocidade u temos que $\frac{Du}{Dt} = \frac{du}{dt} + u\cdot\nabla u$. Agora se consideramos cada componente da equação

$$\int_{\Omega_t}\nabla\cdot S(\bar{x},t)dX$$

isto é, a componente i-ésima é dada por

$$\left(\int_{\Omega_t}\nabla\cdot S(\bar{x},t)dX\right)_i = \left(\int_{\partial\Omega_t}S(\bar{x},t)\cdot ndX\right)_i = \int_{\partial\Omega_t}S_i\cdot ndA = \int_{\Omega_t}\nabla\cdot S_i(\bar{x},t)dX \tag{3.2}$$

onde $S_i = (S_{i1}, S_{i2}, S_{i3})$. Esta decomposição em componentes será usada nos seguintes cálculos.

3.4.1 Fluidos Viscosos, Equações de Navier Stokes

A *viscosidade* é a propriedade associada a resistência que o fluido oferece à tensão de cisalhamento. Esta também descreve a resistência interna ao movimento de um fluido e deve ser pensada como a medida do atrito do fluido. São exemplos de fluidos viscosos o mel e o petróleo. Observamos além que quanto maior a viscosidade, menor será a velocidade em que o fluido se movimenta. Aqui notaremos a *viscosidade* do fluido com a letra grega μ.

Agora, para um fluido viscoso seguindo as referências [33] [13] e [31] vemos

[2] Teorema do Transporte. $\frac{\partial}{\partial t}\int_{\Omega_t}\rho f dX = \int_{\Omega_t}\rho\frac{Df}{Dt}dX$ para qualquer função f. Para mais detalhes ver [13].

[3] Teorema da Divergência. $\int_{S=\partial U}F\cdot ndS = \int_U\nabla\cdot FdV$ onde o vector n é o vetor normal á superfície S apontando para fora do exterior do volume V e F é um campo vetorial de classe C^1.

que a forma da matriz S está definida por

$$S = -pI - \frac{2}{3}\mu(\nabla \cdot u)I + 2\mu(\nabla u + (\nabla u)^t) \tag{3.3}$$

onde p é a pressão do fluido,[4] I é a matriz identidade e

$$\nabla u = \begin{pmatrix} \partial_x u_1 & \partial_y u_1 \\ \partial_x u_2 & \partial_y u_2 \end{pmatrix},$$

e $(\nabla u)^t$ é sua transposta.

Viscosidade Constante. Vamos supor agora que a viscosidade é *constante* no espaço, logo voltando na equação 3.2 e usando a definição 3.3 temos que

$$\begin{aligned}
\left(\int_{\Omega_t} \nabla \cdot S(\bar{x},t)dX\right)_i &= \int_{\Omega_t} \nabla \cdot (S_i)dX = \int_{\Omega_t} \sum_{j=1}^{3} \frac{\partial(S_i)}{\partial x_j} dX = \\
&= \int_{\Omega_t} \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(-p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}(\nabla \cdot u)\right)\right) dX \\
&= \int_{\Omega_t} \left[-\frac{\partial(p)}{\partial x_i} + \sum_{j=1}^{3} \mu\left(\frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{\partial^2 u_j}{\partial x_j \partial x_i} - \frac{2}{3}\frac{\partial}{\partial x_i}(\nabla \cdot u)\right)\right] dX \\
&= \int_{\Omega_t} \left[-\frac{\partial(p)}{\partial x_i} + \mu\Delta u_i + \mu\frac{\partial(\nabla \cdot u)}{\partial x_i} - \mu\frac{2}{3}\frac{\partial(\nabla \cdot u)}{\partial x_i}\right] dX.
\end{aligned}$$

Uma boa aproximação dos líquidos é supor que fluido é incompressível, isto é, quando $\nabla \cdot u = 0$. Portanto para um fluido incompressível temos que

$$\left(\int_{\Omega_t} \nabla \cdot S(\bar{x},t)dX\right)_i = \int_{\Omega_t} \left[-\frac{\partial(p)}{\partial x_i} + \mu\Delta u_i\right] dX$$

ou vetorialmente

$$\int_{\Omega_t} \nabla \cdot S(\bar{x},t)dX = \int_{\Omega_t} [-\nabla p + \mu\Delta u]\, dX.$$

Substituindo o resultado anterior na equação 3.1 obtemos as equações clássicas de Navier-Stokes para viscosidade constante, isto é,

$$\begin{cases} \partial_t u = -\nabla p - u \cdot \nabla u + \mu \triangle u + f \\ \nabla \cdot u = 0 \end{cases} \tag{3.4}$$

[4] A pressão p é a magnitude das forças atuando sobre a superfície do fluido na direção paralela à normal n.

onde $\rho = 1$, isto é, fluido de densidade constante no tempo e no espaço.

Viscosidade Variável no espaço. Supomos agora que viscosidade é variável no espaço.

$$\begin{aligned}
\left(\int_{\Omega_t} \nabla \cdot S(\bar{x},t)dX\right)_i &= \int_{\Omega_t} \nabla \cdot (S_i)dX = \int_{\Omega_t} \sum_{j=1}^{3} \frac{\partial (S_i)}{\partial x_j} dX = \\
&= \int_{\Omega_t} \sum_{j=1}^{3} \frac{\partial}{\partial x_j} \left(-p\delta_{ij} + \mu(\bar{x}) \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}(\nabla \cdot u) \right) \right) dX \\
&= \int_{\Omega_t} \left[-\frac{\partial (p)}{\partial x_i} + \sum_{j=1}^{3} \frac{\partial \mu(\bar{x})}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] dX + \\
&\quad \int_{\Omega_t} \left[\sum_{j=1}^{3} \mu(\bar{x}) \left(\frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{\partial^2 u_j}{\partial x_j \partial x_i} - \frac{2}{3} \frac{\partial}{\partial x_i} (\nabla \cdot u) \right) \right] dX \\
&= \int_{\Omega_t} \left[-\frac{\partial (p)}{\partial x_i} + \nabla \mu(\bar{x}) \cdot \nabla u_i + \nabla \mu(\bar{x}) \cdot \frac{\partial u}{\partial x_i} \right] dX + \\
&\quad \int_{\Omega_t} \left[\mu(\bar{x}) \Delta u_i + \mu(\bar{x}) \frac{\partial (\nabla \cdot u)}{\partial x_i} - \mu(\bar{x}) \frac{2}{3} \frac{\partial (\nabla \cdot u)}{\partial x_i} \right] dX.
\end{aligned}$$

Portanto para um fluido incompressível temos que

$$\left(\int_{\Omega_t} \nabla \cdot S(\bar{x},t)dX\right)_i = \int_{\Omega_t} \left[-\frac{\partial (p)}{\partial x_i} + \nabla \mu(\bar{x}) \cdot \nabla u_i + \nabla \mu(\bar{x}) \cdot \frac{\partial u}{\partial x_i} + \mu(\bar{x}) \Delta u_i \right] dX$$

ou vetorialmente

$$\int_{\Omega_t} \nabla \cdot S(\bar{x},t)dX = \int_{\Omega_t} \left[-\nabla p + \nabla \mu(\bar{x})(\nabla u + (\nabla u)^t) + \mu(\bar{x}) \Delta u \right] dX.$$

Por conseguinte substituindo o resultado anterior nas equações 3.1 obtemos as equações de Navier-Stokes para viscosidade variável no espaço

$$\begin{cases} \dfrac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \nabla \mu(\bar{x})(\nabla u + (\nabla u)^t) + \mu(\bar{x}) \Delta u + f, \\ \nabla \cdot u = 0. \end{cases} \tag{3.5}$$

Aqui, novamente assumimos densidade do fluido constante ao longo do tempo e o espaço, ou seja, consideramos $\rho = 1$. Fazemos esta suposição matemática para simplificar a equação ao mesmo tempo que os cálculos numéricos da equação. Agora, na revisão da literatura encontramos casos de modelos físicos de fluido de viscosidade variável no tempo e esses modelos assumem densidade

constante[12], enquanto outros modelos para fluidos com viscosidade variável com a temperatura, consideram a densidade como uma função da temperatura [25].

3.4.2 Teorema de Decomposição de Helmholtz-Hodge

Este teorema nos diz que qualquer campo de vetores pode ser decomposto como a soma única de um campo de vetores com divergente igual a zero e um campo de vetores igual ao gradiente de um campo escalar. Veremos que o teorema de decomposição é usado para escrever as equações de Navier-Stokes sem o termo da pressão, isto sera útil dentro do esquema da simulação do fluido.

Teorema 1. *Seja $D \subset \mathbb{R}^n$ uma região no espaço com contorno suave ∂D. E seja $w : \overline{D} \to \mathbb{R}^n$ um campo de vetores de classe $C^1(D, \mathbb{R}^n)$. Então w pode ser decomposto unicamente da forma:*

$$w = u + \nabla\phi$$

tal que $u : \overline{D} \to \mathbb{R}^n$ e $\nabla\phi : \overline{D} \to \mathbb{R}^n$ são de classe $C^1(D, \mathbb{R}^n)$ onde $\nabla \cdot u = 0$ e $u \cdot n = 0$ em ∂D (u é tangente a ∂D) e $\phi : \overline{D} \to \mathbb{R}$ é classe $C^2(D, \mathbb{R})$.

Demonstração. (Ver Chorin [13] pagina 37).

Existência.

Vamos supor que já temos uma decomposição de w, isto é,

$$w = u + \nabla\phi$$

onde $\nabla \cdot u = 0$. Aplicando o divergente a cada lado da igualdade anterior temos que

$$(*) \begin{cases} \nabla \cdot w = \Delta\phi \\ w \cdot n = \nabla\phi \cdot n \end{cases}$$

note que $\nabla \cdot \nabla\phi = \Delta\phi$. As equações $(*)$ dão uma idea para provar existência da

decomposição. Dado w, definimos ϕ como a solução do problema de Neumann

$$(**)\begin{cases}\Delta\phi = \nabla\cdot w \text{ em } D\\ \frac{\partial\phi}{\partial n} = w\cdot n \text{ em } \partial D.\end{cases}$$

O problema de Neumann $(**)$ tem solução única (ver [13]) a menos uma constante se, somente se,

$$\int_D \nabla\cdot w dV = \int_{\partial D} w\cdot n dS,$$

o que é verdade pelo teorema da divergência. Portanto temos que existe a solução única ϕ de $(**)$ a menos uma constante. Agora, definimos u por:

$$u = w - \nabla\phi.$$

E vemos que o campo de vetores u tem as propriedades desejadas:

$$\begin{cases}\nabla\cdot u = \nabla\cdot w - \nabla\phi = 0 \text{ em } D\\ u\cdot n = w\cdot n - \frac{\partial\phi}{\partial n} = 0 \text{ em } \partial D.\end{cases}$$

Assim provamos a existência.

Unicidade Primeiro vejamos que

$$\int_D u\cdot\nabla\phi dV = 0. \tag{3.6}$$

Usemos a identidade $\nabla\cdot(\phi u) = (\nabla\cdot u)\phi + u\cdot\nabla\phi$, junto com o teorema da divergência temos que

$$\int_D (u\cdot\nabla\phi)dV = \int_D (\nabla\cdot(u\phi))dV = \int_{\partial D}(\phi u\cdot n)dS = 0,$$

porque $u\cdot n = 0$ em ∂D e $\nabla\cdot u = 0$.

Suponhamos agora que $w = u_1 + \nabla\phi_1 = u_2 + \nabla\phi_2$. Então

$$0 = u_1 - u_2 + \nabla(\phi_1 - \phi_2).$$

Logo o produto interno de $u_1 - u_2 + \nabla(\phi_1 - \phi_2)$ com $(u_1 - u_2)$ é integrado sobre

D temos que

$$\begin{aligned} 0 &= \int_D (u_1 - u_2 + \nabla(\phi_1 - \phi_2)) \cdot (u_1 - u_2) dV \\ &= \int_D \|u_1 - u_2\|^2 + \nabla(\phi_1 - \phi_2) \cdot (u_1 - u_2) dV \\ &= \int_D \|u_1 - u_2\|^2 dV, \end{aligned}$$

pela equação 3.6. Logo $u_1 = u_2$, e portanto $\nabla\phi_1 = \nabla\phi_2$, o que é o mesmo que $\phi_1 = \phi_2 + constante$. □

O teorema 1 permite-nos fazer a seguinte definição.

Definição 1. *Dada uma região $D \subset \mathbb{R}^n$ e $w : \overline{D} \to \mathbb{R}^n$ um campo de vetores o* ***Operador de Projeção*** $\mathbb{P}$ *projeta w em sua parte livre de divergente, isto é, $\mathbb{P}(w) = u$ onde $u : \overline{D} \to \mathbb{R}^n$ é um campo de vetores tal que $\nabla \cdot u = 0$.*

O *Operador de Projeção* possui as seguintes propriedades:

- $\mathbb{P}$ é linear.
- Se $u : \overline{D} \to \mathbb{R}^n$ é tal que $\nabla \cdot u = 0$, então $\mathbb{P}(u) = u$.
- Se $\phi : \overline{D} \to \mathbb{R}$ é de classe $C^2(D, \mathbb{R}^n)$, então $\mathbb{P}(\nabla\phi) = 0$.

Usamos o *operador de projeção* para reescrever as equações de Navier-Stokes 3.4 e 3.5, mas sem o termo da pressão. Temos então que as equações de Navier-Stokes para viscosidade constante agora são dadas por

$$\begin{cases} \partial_t u = \mathbb{P}(-u \cdot \nabla u + \mu \triangle u + f), \\ \nabla \cdot u = 0, \\ u|_{\partial D} = 0. \qquad \text{(condição de no escorregamento)} \end{cases} \tag{3.7}$$

E as equações de Navier-Stokes para viscosidade variável estão dadas por

$$\begin{cases} \dfrac{\partial u}{\partial t} = \mathbb{P}(-u \cdot \nabla u + \nabla\mu(\bar{x})(\nabla u + (\nabla u)^t) + \mu(\bar{x})\Delta u + f), \\ \nabla \cdot u = 0, \\ u|_{\partial D} = 0. \qquad \text{(condição de no escorregamento)} \end{cases} \tag{3.8}$$

3.4.3 Modelo Numérico e Método Computacional

Para resolver numericamente as anteriores equações de Navier-Stokes (3.7) e 3.8 e simular fluidos, nós adotaremos o esquema do algoritmo *Stable Fluids* [38]. Este algoritmo que tem uma estrutura simples é o algoritmo mais usado em computação gráfica para simular fluidos, já que é incondicionalmente estável para qualquer passo do tempo.
O Stable Fluids segue uma visão euleriana do fluido, discretiza os operadores diferenciais (Laplaciano, divergente, gradiente, etc) por diferencias finitas e usa o método *operator splitting* [43] para resolver as equações. O método *operator splitting* divide as equações em quatro etapas e em cada etapa é resolvido um termo das equações. Por cada passo do tempo $\triangle t$ o algoritmo resolve as quatro etapas, começando do campo de velocidades $w_0 = u(\bar{x},t)$ que é passado a cada etapa sequencialmente. As etapas são

$$w_0 \xrightarrow[\text{forças externas}]{\text{adição de}} w_1 \xrightarrow{\text{advecção}} w_2 \xrightarrow{\text{difusão}} w_3 \xrightarrow{\text{projeção}} w_4$$

O nomes das etapas dependem dos termos da equação. E seguindo essa ideia, vamos dividir as equações 3.7 e 3.8 cada uma em quatro etapas, para ambas equações as etapas de advecção, projeção e adição de forças são as mesmas. A etapa da difusão é diferente devido ao termo da viscosidade que é a principal diferencias entre as duas equações

Na primeira etapa de adição de forças externas, o campo de forças externas vezes o passo de tempo se soma ao campo anterior de velocidades, $w_1 = w_0 + \triangle t f(\bar{x},t)$.

A segunda etapa, advecção, calculamos como o fluido se transporta sobre seu próprio campo de velocidades.

$$\partial_t w_2 = -(w_1 \cdot \nabla) w_2$$

e a equação é resolvida usando a técnica de semi-Lagrangiano [14]

$$w_2(\bar{x}) = w_1(\bar{x} - \triangle t w_1(\bar{x})).$$

A ideia básica atrás da advecção é, ao invés de transportar a partícula através

do campo de velocidades no tempo, mover esta para trás no tempo através do campo e calcular a nova velocidade por interpolação.

A terceira etapa resolve o termo da viscosidade dada pela equação

$$\partial_t w_3 = \mu \Delta w_3$$

usa um esquema implícito de diferenças finitas para achar a solução da equação de difusão

$$(I - \Delta t \mu \Delta) w_3 = w_2$$

onde I é o operador identidade. Uma forma de resolver o sistema de equações dada pela equação acima é achar w_3 em

$$A w_3 = w_2$$

usando o método de Jacobi.

Finalmente a quarta etapa projeta o campo de velocidades sobre seu campo incompressível (de divergente zero). Para isso deve ser resolvida a equação de Poisson $\Delta q = \nabla \cdot w_3$ e portanto teríamos que $w_4 = w_3 - \nabla q$ de novo usamos aqui diferenças finitas e Jacobi para resolver a equação de Poisson.

Estendemos agora o método de Stable Fluids para o caso de fluido com viscosidade variável no espaço (3.8). Usaremos as mesmas etapas, exceto no termo da difusão que será modificado. No esquema o termo para resolver a difusão é

$$\partial_t w_3 = \mu \Delta w_3$$

para o caso da viscosidade variável, devemos achar a solução de

$$\partial_t w_3 = \nabla \mu(\bar{x})(\nabla w_3 + \nabla w_3{}^{\top}) + \mu(\bar{x}) \Delta w_3.$$

Se $w_3 = (w^1, w^2)$ escreveremos de novo a equação acima nesta forma

$$\begin{cases} w_t^1 = 2\mu_x w_x^1 + \mu_y (w_y^1 + w_x^2) + \mu \Delta w^1 \\ w_t^2 = 2\mu_y w_y^2 + \mu_x (w_y^1 + w_x^2) + \mu \Delta w^2 \end{cases}$$

onde os subíndices t, x e y denotam as derivadas parciais ∂_t, ∂_x e ∂_y. Dis-

cretizamos usando diferenças finitas implícitas no tempo e usamos o esquema *central space* para w_3 e μ no espaço (ver apêndice). Os resultados são estáveis como queríamos.

O algoritmo Stable Fluids também inclui um método para calcular o movimento de um material φ imerso no fluido. A equação de evolução deste material é

$$\partial_t \varphi = -(v \cdot \nabla)\varphi + \kappa \triangle \varphi + S \tag{3.9}$$

onde κ é o coeficiente de difusão, e S é uma fonte de material. O material é transportado pelo campo de velocidades do fluido, e usamos a técnica do semi-Lagrangiano para resolver essa parte.

3.5 Controle Eficiente com Método Adjunto

A partir da ideia de controlar fluidos de forma precisa, pontual e rápida (ver seção 4.5.3), começamos a estudar a técnica do método adjunto.

3.5.1 O Método Adjunto

O Método Adjunto foi o nome da técnica introduzida em [30] para calcular a derivada da função objetivo (função real definida no capítulo 4 na seção 4.3.3). Onde, a derivada desta função, foi usada dentro de um método de otimização numérico para achar os parâmetros necessários para controlar a simulação de fluidos.

A técnica do Método Adjunto proposta em [30] esta baseada no tópico ¨dualidade e variáveis adjuntas¨ de [22] porém, aqui adotaremos a definição de Greining [21] para o método adjunto. No artigo [22] o cálculo do gradiente da função objetivo é obtido a partir das equações diferenciais e em [21] temos que e calculado a partir do código.

Na literatura o termo ¨método adjunto¨ está usualmente relacionado com o conceito das equações diferenciais parciais (EDP's) adjuntas (discretas ou continuas) [22].

Observemos que o cálculo direto do gradiente é muito custoso. Suponhamos que temos a função objetivo $f : \mathbb{R}^m \to \mathbb{R}$, então o vetor gradiente $\nabla_x f(x_0)$ pode ser aproximado por diferenças finitas, o qual necessita de $m+1$ cálculos de f.

Uma forma de contornar o problema é proposto por Giles e Pierce em [22]. O processo começa com a discretização de uma EDP não linear, logo as equações são linearizadas e finalmente são usadas as transpostas das matrizes. As vantagens do trabalho, segundo os autores, são um cálculo exato do gradiente e a criação do código é direta [22]. Mas essa técnica partir das EDP's possui dois problemas, segundo Greining [21], um código grande e pesado além que as condições de fronteira devem ser tratadas separadamente.

Outro enfoque é tratar o problema direto do código, isto é, desenvolver a partir do código numérico o algoritmo para calcular o gradiente. O que e conhecido como derivação automática. Observamos de novo que para nós o método adjunto é a técnica ou regras para calcular o gradiente de uma função escalar (função objetivo) a partir do código numérico.

Diferenciação Automática Em matemática e álgebra computacional, *diferenciação automática*, ou DA, também conhecida como *diferenciação algorítmica*, é um conjunto de técnicas para avaliar numericamente a derivada de uma função especificada por um programa de computação. DA explora o fato que todo programa, não importa quão complicado seja, executa uma sequencia de operações ariméticas elementais (adição, subtração, multiplicação, divisão, etc.) e funções elementais (exp, log, sen, cos, etc.). Aplicando a regra da cadeia varias vezes sobre estas operações, temos que as derivadas de qualquer grau podem ser calculadas automaticamente com precisão. Temos que a *diferenciação automática* não é *Diferenciação simbólica*, ou *Diferenciação numérica* (método de diferencias finitas).

Estes métodos clássicos tem os seguintes problemas: a *diferenciação simbólica* trabalha de forma lenta e é difícil de converter em uma expressão simples, enquanto a *diferenciação numérica* pode introduzir erros nos processos de discretização e cancelação. Estes métodos tem problemas na hora de cal-

cular derivadas de alto grau, onde a complexidade e os erros se incrementam. Finalmente, estes métodos são lentos para calcular derivadas parciais de uma função que depende de varias variáveis, o que é necessário em algoritmos de otimização baseados em gradiente. A *diferenciação automática* resolve estes problemas [1].

Embora, a implementação da ferramenta DA e difícil, foi feita nesta tese pelo descrito no parágrafo anterior. Para isto, foram estudadas as regras descritas em [21] e logo implementadas para criar o código adjunto. O código adjunto (como sera visto nas seguentes seções) e criado a partir do código original (nosso caso o código da simulação de fluidos), e este código adjunto calcula o gradiente da função objetivo cada vez que for necessário, como veremos no capítulo 4 na seção 4.5.3.

3.5.2 Derivada de Algoritmos

Um algoritmo numérico pode ser visto como uma função de $\mathbb{R}^n$ até $\mathbb{R}^m$ e cada passo do algoritmo como uma função (supondo todas as hipóteses necessárias para considerar isto possível). Portanto o algoritmo é uma composição de funções, onde cada função representa uma instrução no código. Suponhamos que cada função é diferenciável. Representemos o algoritmo pela função H, logo a derivada do algoritmo pode ser calculado pela regra da cadeia.

Seja

$$\begin{aligned} H : \mathbb{R}^n &\longrightarrow \mathbb{R}^m \\ x &\longmapsto y \end{aligned}$$

a função que representa o algoritmo numérico. Temos que o algoritmo numérico pode ser decomposto em k passos ($k \in \mathbb{N}$), e cada passo pode ser representado explicitamente como

$$\begin{aligned} h^l : \mathbb{R}^{n_{l-1}} &\longrightarrow \mathbb{R}^{n_l} \qquad\qquad (l = 1, ..., k). \\ z^{l-1} &\longmapsto z^l \end{aligned}$$

A função H é portanto a composição

$$H = h^k \circ \cdots \circ h^1 := \bigodot_{l=1}^{k} h^l.$$

Um exemplo de isto pode ser visto na seção 3.5.5, onde temos um modelo matemático dado pelas equações 3.20 e 3.21 e o código (ver figura 3.8) gerado por destas equações. Já que cada passo é diferenciável derivamos H. Seja $J_H(x_0)$ a matriz Jacobiana de H em x_0 logo

$$J_H(x_0) = \left.\frac{\partial h^k}{\partial z^{k-1}}\right|_{z^{k-1}=\bigodot_{i=1}^{k-1} h^i(x_0)} \cdots\cdots \left.\frac{\partial h^1}{\partial z^0}\right|_{z^0=x_0}$$

onde $z^l = \bigodot_{i=1}^{l} h^i(x), \quad z_0^l = \bigodot_{i=1}^{l} h^i(x_0) \quad \text{e} \quad (1 \leq l \leq k).$

Então temos que a matriz Jacobiana é um múltiplo produto das matrizes $\frac{\partial h^l}{\partial z^{l-1}}$. Portanto podemos calcular este produto de duas formas. A primeira, calcula o produto na mesma ordem como é avaliada a composição. Chamamos essa forma de *modo direto.* A segunda, avalia o produto na ordem contraria ao modo direto. Chamamos esta estratégia de *modo inverso.*

Modo Direto

A ordem em que algoritmo avalia a composição é a seguinte: o algoritmo primeiro calcula $h^1(x_0)$ e logo $h^2(h^1(x_0))$ e depois $h^3(h^2(h^1(x_0)))$ e assim por diante.
Operar usando o modo direto é então avaliar primeiro a matriz $\frac{\partial h^1}{\partial z^0}$, logo o produto $\frac{\partial h^2}{\partial z^1}\frac{\partial h^1}{\partial z^0}$, depois o produto $\frac{\partial h^3}{\partial z^2}\frac{\partial h^2}{\partial z^1}\frac{\partial h^1}{\partial z^0}$ e assim por diante como vemos na figura 3.4.

Modo Inverso

Em contraste, o *modo inverso* começa com à avaliação de $\frac{\partial h^k}{\partial z^{k-1}}$ e logo do produto $\frac{\partial h^k}{\partial z^{k-1}} \cdot \frac{\partial h^{k-1}}{\partial z^{k-2}}$. Neste ultimo caso, os resultados intermediários tem n colunas, e o ultimo tem m linhas.

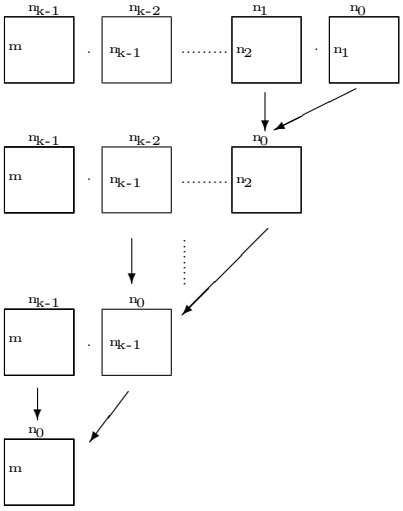


Figura 3.4: Modo Direto

Assim para $m > n$ o modo direto precisa de poucos cálculos numéricos, agora se $m < n$ o modo inverso precisa de menos cálculos e nesse caso é o melhor modo de calcular o matriz Jacobiana J_H. Ver figura 3.5.

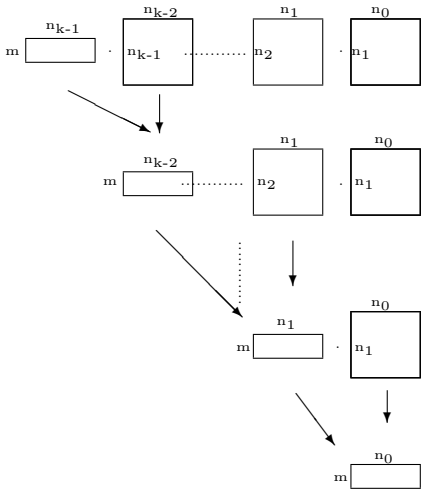


Figura 3.5: Modo inverso

3.5.3 Derivada de um Campo Escalar

Seja $H : \mathbb{R}^n \to \mathbb{R}^m$ a função definida por um algoritmo numérico. Se H é um campo escalar, o modo inverso é a melhor opção no caso de querer calcular o gradiente de H. Operar em modo inverso quando $m = 1$ é chamado de *método*

adjunto, e o algoritmo para calcular o gradiente é chamado de *modelo adjunto* e seu código é chamado de *código adjunto*.

A decomposição de H é

$$H = \bigodot_{l=1}^{k} h^l \quad \text{onde} \quad \begin{cases} h^l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l} & \text{para} \quad (l = 1, ..., k-1) \\ h^k : \mathbb{R}^{n_{k-1}} \to \mathbb{R} & \text{para} \quad l = k \end{cases} \tag{3.10}$$

Usaremos o seguinte fato que é válido para qualquer função real $f : \mathbb{R}^{n_{k-1}} \to \mathbb{R}$ diferenciável. Decompondo f por série de Taylor temos que

$$f(x) = f(x_0) + \langle \nabla_x f(x_0), x - x_0 \rangle + o(|x - x_0|)$$

Podemos aproximar e reescrever a equação como:

$$\delta f = \langle \nabla_x f(x_0), \delta x \rangle \tag{3.11}$$

onde $\delta f = f(x) - f(x_0)$ e $\delta x = x - x_0$ e são chamadas de *variações*.

Definimos $z^l = \bigodot_{i=1}^{l} h^i(x)$ por (3.10) para $(1 \leq l \leq k)$ logo a variação δz^l depende da variação das variáveis de controle δx e pode ser escrita assim

$$\delta z^l = \left. \frac{\partial \left(\bigodot_{i=1}^{l} h^i(x) \right)}{\partial x} \right|_{x=x_0} \delta x \qquad \text{onde } \delta z^0 := \delta x \tag{3.12}$$

Agora também temos que

$$\delta z^l = \left. \frac{\partial h^l(z^{l-1})}{\partial z^{l-1}} \right|_{z^{l-1}=z_0^{l-1}} \delta z^{l-1} \tag{3.13}$$

O **adjunto** do resultado z^l é definido como o gradiente de H com relação a z^l

$$\delta^* z^l := \nabla_{z^l} \bigodot_{i=l+1}^{k} h^i(z^l) \Bigg|_{z^l=z_0^l} \text{ para } 0 \leq l \leq k-2 \tag{3.14}$$

Pela equação (3.11) temos para o gradiente de H a seguinte expressão

$$\delta H = \langle \delta^* z^l \, , \, \delta z^l \rangle \tag{3.15}$$

Agora, isso é válido para todo l, logo

$$\begin{aligned}
\langle \delta^* z^{l-1} \, , \, \delta z^{l-1} \rangle &= \langle \delta^* z^l \, , \, \delta z^l \rangle \text{ por (3.13) segue} \\
&= \left\langle \delta^* z^l \, , \, \left(\frac{\partial h^l(z^{l-1})}{\partial z^{l-1}} \right) \Big|_{z^{l-1}=z_0^{l-1}} \delta z^{l-1} \right\rangle \\
&= \left\langle \left(\frac{\partial h^l(z^{l-1})}{\partial z^{l-1}} \right)^* \Big|_{z^{l-1}=z_0^{l-1}} \delta^* z^l \, , \, \delta z^{l-1} \right\rangle
\end{aligned}$$

Já que vale para todo δz^{l-1}, temos a seguinte identidade que será a regra geral para construção do código adjunto.

$$\delta^* z^{l-1} = \left(\frac{\partial h^l(z^{l-1})}{\partial z^{l-1}} \right)^* \Big|_{z^{l-1}=z_0^{l-1}} \delta^* z^l \tag{3.16}$$

Definimos aqui para $l = k-1$

$$\delta^* z^{k-1} = \left(\frac{\partial h^k(z^{k-1})}{\partial z^{k-1}} \right)^* = \left(\nabla_{z^{k-1}} h^k(z^{k-1}) \right)^\top \tag{3.17}$$

Escrevendo uma sequencia de passos temos então

$$\begin{aligned}
\delta^* z^{k-2} &= \left(\frac{\partial h^{k-1}(z^{k-2})}{\partial z^{k-2}} \right)^* \delta^* z^{k-1} \\
&\vdots \\
\delta^* z^1 &= \left(\frac{\partial h^2(z^1)}{\partial z^1} \right)^* \delta^* z^2 \\
\delta^* z^0 &= \left(\frac{\partial h^1(x)}{\partial x} \right)^* \delta^* z^1
\end{aligned}$$

Mas pela definição do adjunto do resultado $\delta^* z^l$ (3.14) temos que $\delta^* z^0 = \nabla_x H$ o gradiente de H com relação as variáveis de controle, e avaliado então no ultimo

passo.

$$\delta^* z^0 = \left(\frac{\partial h^1(x)}{\partial x} \right)^* \delta^* z^1 = \nabla_x H \tag{3.18}$$

3.5.4 Código Adjunto

O código que representa o modelo adjunto é chamado de código adjunto e mostraremos a implementação da regra (3.16) na construção deste. Para isso usaremos alguns conceitos básicos.

- **Variáveis adjuntas** Os resultados intermediários z_i^l denotam os valores das variáveis do código. Nós calculamos os adjuntos $\delta^* z_i^l$ destes valores. Então definimos variáveis adjuntas para cada um desses valores.

- **Variáveis ativas** dependem das variáveis de controle e tem influência na função objetivo. Só as variáveis caracterizadas por números reais podem ser ativas. Para variáveis não ativas não são construídas instruções adjuntas.

- **Localização** A posição das instruções adjuntas no código adjunto é determinado pela ordem das instruções no código isso se o código adjunto é uma implementação do modo inverso. Então as instruções adjuntas estão na ordem inversa das instruções do código.

- **Escrita** É recomendado seguir uma convenção para criar os nomes das variáveis adjuntas tal que sejam fáceis de lembrar e de localizar. Considerando isto aqui, teremos que o nome de uma variável adjunta é o nome da variável original precedido ou acompanhado de uma letra ou mais letras ou um símbolo.
  Exemplo: O nome da variável adjunta de `x` é `ax` ou `adx` ou `x'` ou `x*`. Aqui usaremos a notação `ax` para notar a variável adjunta de `x`. Outros autores podem seguir qualquer das notações anteriores.

Construção de Código Adjunto

Um programa é composto por um conjunto de instruções. Mostraremos como traduzir cada instrução em sua correspondente adjunta. Notaremos o adjunto de uma instrução I por $A(I)$. Assim se uma parte do código composto das instruções

$$I_1, I_2, \ldots, I_n$$

tem associado o programa adjunto

$$A(I_n), A(I_{n-1}), \ldots, A(I_1)$$

o programa adjunto inverte a ordem das instruções.

Mostraremos como construir o adjunto para algumas instruções usando os conceitos básicos e a regra (3.16).
Só de variáveis ativas criamos instruções adjuntas. Não todas as variáveis adjuntas intervém em uma instrução. Um subconjunto delas está presente em cada instrução.

Consideremos a instrução

$$x = y^2$$

Podemos considerar a instrução como uma função atuando sobre as variáveis ativas x e y

$$(x,y) = f(x,y) = (y^2, y).$$

Para construir a instrução adjunta achamos o Jacobiano da função f como em (3.13) e obtemos

$$\begin{pmatrix} 0 & 2y \\ 0 & 1 \end{pmatrix}$$

logo vetor variação é

$$\begin{pmatrix} \delta x \\ \delta y \end{pmatrix}^{l} = \begin{pmatrix} 0 & 2y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}^{l-1}$$

onde o $l-1$ e l denotão os valores das variáveis antes e depois da instrução. Agora o operador adjunto é a transposta da matriz atuando sobre as variáveis adjuntas, ver (3.16).

$$\begin{pmatrix} \delta^* x \\ \delta^* y \end{pmatrix}^{l-1} = \begin{pmatrix} 0 & 0 \\ 2y & 1 \end{pmatrix} \begin{pmatrix} \delta^* x \\ \delta^* y \end{pmatrix}^{l}$$

Daí temos

$$\begin{aligned} (\delta^* x)^{l-1} &= 0 \\ (\delta^* y)^{l-1} &= 2y(\delta^* x)^l + (\delta^* y)^l \end{aligned}$$

Usando o princípio básico de escrita temos então que o código é

Código	Código Adjunto
`x = y*y`	`ay = 2y*ax+ay`
	`ax = 0`

Do anterior podemos inferir a seguinte regra. Dada uma instrução

$$z = f(x, y, ..., z) \tag{3.19}$$

Achamos o Jacobiano da função $h(x, y, ..., z) = (x, y, ..., f)$ atuando sobre as variáveis ativas $(x, y, ..., z)$, e escrevendo a variação das variáveis temos

$$\begin{pmatrix} \delta x \\ \vdots \\ \delta y \\ \delta z \end{pmatrix}^{l} = \begin{pmatrix} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \\ f_x(x, y, \cdots, z) & \cdots & f_y(x, y, \cdots, z) & f_z(x, y, \cdots, z) \end{pmatrix} \begin{pmatrix} \delta x \\ \vdots \\ \delta y \\ \delta z \end{pmatrix}^{l-1}$$

Agora com a transposta da matriz temos

$$\begin{pmatrix} \delta^* x \\ \vdots \\ \delta^* y \\ \delta^* z \end{pmatrix}^{l-1} = \begin{pmatrix} 1 & \cdots & 0 & f_x(x, y, \cdots, z) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & f_y(x, y, \cdots, z) \\ 0 & \cdots & 0 & f_z(x, y, \cdots, z) \end{pmatrix} \begin{pmatrix} \delta^* x \\ \vdots \\ \delta^* y \\ \delta^* z \end{pmatrix}^{l}$$

Logo para criar o código adjunto para qualquer instrução da forma (3.19) temos a seguinte regra geral

Código	Código Adjunto

`z = f(x,y,...,z)`	`ax = ax +` $f_x(x,y,...,z)$`az`
	`ay = ay +` $f_y(x,y,...,z)$`az`
	$\vdots$
	`az =` $f_z(x,y,...,z)$`az`

Figura 3.6: Regra geral para criar o código adjunto de uma instrução da forma $z = f(x,y,...,z)$.

Para construção do código adjunto de laços `(for, while)` ou tomadas de decisão (`if-else`), cada instrução é substituída por sua instrução adjunta e para o caso do `for` temos que troca a ordem do laço

$$A\begin{pmatrix}\texttt{for i= } 0,\cdots,n \texttt{ do}\\ I_i \\ \texttt{end for}\end{pmatrix} = \begin{array}{c}\texttt{for i= } n,\cdots,0 \texttt{ do}\\ A(I_i) \\ \texttt{end for}\end{array}$$

$$A\begin{pmatrix}\texttt{if B then}\\ I_1 \\ \texttt{else}\\ I_2 \\ \texttt{end if}\end{pmatrix} = \begin{array}{c}\texttt{if B then}\\ A(I_1) \\ \texttt{else}\\ A(I_2) \\ \texttt{end if}\end{array}$$

Figura 3.7: Regra geral para criar o código adjunto para laços e tomada de decisões.

3.5.5 Exemplo

Para ilustrar todo o anterior consideremos o seguinte exemplo. Suponhamos que de algum processo físico temos as seguintes equações

$$x = y + z + u^2; \tag{3.20}$$
$$y = zy + xv;$$

onde queremos minimizar a função objetivo

$$f = x^2 + y^2 + u^2 + v^2; \tag{3.21}$$

em relação às variáveis de controle u e v usando um método de otimização. Para minimizar a função objetivo precisamos do gradiente da função . O código das equações e suas derivadas está dado pela figura 3.8. Estas derivadas serão chamadas de derivadas diretas.

```
float x, y, z, dx[2], dy[2], dz[2]; // variables
float u, v, du[2], dv[2];           // controls
float f, df[2];                     // cost variable

dx[0]=dx[1]=dy[0]=dy[1]=dz[0]=dz[1]=df[0]=df[1]=0;
du[0]=1; du[1]=0; dv[0]=0; dv[1]=1;

x = y + z + u*u;    // statement #1
  dx[0] = dy[0] + dz[0] + 2*u*du[0];
  dx[1] = dy[1] + dz[1] + 2*u*du[1];

y = z*y + x*v;      // statement #2
  dy[0] = z*dy[0] + y*dz[0] + v*dx[0] + x*dv[0];
  dy[1] = z*dy[1] + y*dz[1] + v*dx[1] + x*dv[1];

f = x*x + y*y + u*u + v*v; // cost function
  df[0] = 2*x*dx[0] + 2*y*dy[0] + 2*u*du[0] + 2*v*dv[0];
  df[1] = 2*x*dx[1] + 2*y*dy[1] + 2*u*du[1] + 2*v*dv[1]
```

Figura 3.8: Código das equações (3.20) e (3.21) e derivadas. Derivação direta.

Desvantagem da derivação direta A derivação direta é muito cara quando tem muitos controles, para um milhão de controles teríamos que calcular, então um milhão de derivadas para cada variável e por cada instrução no código. Usando o método adjunto, só é necessário o cálculo de uma variável adjunta por cada variável, em cada instrução. Para finalmente calcular o gradiente da função objetivo. Ver figura 3.9.

```
#define NC 1000000

float x, y, z, dx[NC], dy[NC], dz[NC]; // variables
float u[NC], du[NC];                   // controls
float f, df[NC];                       // cost variable

x = y + z + u[0]*u[0];   // statement #1
for ( int i=0 ; i<NC ; i++ ) {
   dx[i] = dy[i] + dz[i] + ( i==0 ? 2*u[0]*du[0] : 0 );
}
```

Figura 3.9: Cálculo de derivadas em relação a um milhão de controles.

Construção do Código Adjunto

Mostraremos a seguir como construir o código adjunto a partir do código original. Agora nosso primeiro passo é armazenar o gradiente de cada variável na matriz:

$$X^t = \begin{bmatrix} dx[0] & dy[0] & dz[0] & du[0] & dv[0] \\ dx[1] & dy[1] & dz[1] & du[1] & dv[1] \end{bmatrix}$$

Logo da `instrução #1` do código na figura 3.8 temos

```
x = y + z + u*u;    // statement #1
dx[0] = dy[0] + dz[0] + 2*u*du[0];
dx[1] = dy[1] + dz[1] + 2*u*du[1];
```

Podemos expressar isso como o produto de matrizes

$$\begin{bmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 2*u & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{bmatrix}$$

$$X_1 = A_1 X_0$$

Mas em termos de variação, a derivada da `instrução #1` na figura 3.8 também pode ser escrita na forma

$$\begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta u \\ \delta v \end{bmatrix}^l = \begin{bmatrix} 0 & 1 & 1 & 2*u & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta u \\ \delta v \end{bmatrix}^{l-1}$$

Definimos agora o vetor de variáveis adjuntas

$$q = \begin{bmatrix} ax & ay & az & au & av \end{bmatrix}$$

Aplicando então a transposta (adjunta) da matriz acima nas variáveis adjuntas seguindo (3.16) obtemos o seguinte produto

$$\begin{bmatrix} ax \\ ay \\ az \\ au \\ av \end{bmatrix}^{l-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 2*u & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ax \\ ay \\ az \\ au \\ av \end{bmatrix}^l$$

$$q_0 = A_1^\top q_1$$

Agora fazendo o produto acima conduz ao código adjunto da `instrução #1` na figura 3.8 esta dado por 3.10.

Aqui para ilustrar é escrito a matriz da derivada da instrução e sua transposta para derivar o código adjunto, na pratica é melhor seguir a regra acima 3.6 para escrever os códigos adjuntos.

```
au= au + 2*u*ax;
az= az + ax;
ay= ay + ax;
ax= 0;
```

Figura 3.10: Código adjunto da instrução $x = y + z + u * u$; no código 3.8

Seguindo então essas regras para cada instrução obtemos finalmente o código adjunto. Comparamos o código adjunto com o código original, ver figura 3.11.

```
float x, y, z, ax, ay, az;
float u, v, au, av;
float f, af, df[2];

af = 1;
ax = ay = az = au = av = 0;

// adjoint of cost function
ax += 2*x*af; ay += 2*y*af;
au += 2*u*af; av += 2*v*af;

// adjoint of statement #2
az += y*ay; ax += u*ay;
av += x*ay; ay = z*ay;

// adjoint of statement #1
ay += ax; az += ax;
au += 2*u*ax; ax = 0;

// compute gradient
df[0] = au;
df[1] = av;
```

```
float x, y, z;
float u, v;
float f;

// statement #1
x = y + z + u*u;

// statement #2
y = z*y + x*v;

// cost function
f = x*x + y*y + u*u + v*v;
```

Figura 3.11: Código adjunto e código original

Compare o código acima e código abaixo, e observe a diferença na definição da variável adjunta `af` [40]. O código abaixo, inicializa as variáveis adjuntas com os valores do gradiente de `f` e isto concorda com a equação (3.17).Estas duas definições são equivalentes, o usuário escolhe como trabalhar.

x = y + z + u*u;

```
float x, y, z, ax, ay, az;
float u, v, au, av;
float f, df[2];

ax = 2x; ay = 2y;  az = 0 ;
au = 2u; av = 2v;

// adjoint of statement #2
az += y*ay; ax += u*ay;
av += x*ay; ay = z*ay;

// adjoint of statement #1
ay += ax; az += ax;
au += 2*u*ax; ax = 0;

// compute gradient
df[0] = au;
df[1] = av;
```

```
float x, y, z;
float u, v;
float f;

// statement #1
x = y + z + u*u;

// statement #2
y = z*y + x*v;

// cost function
f = x*x + y*y + u*u + v*v;
```

Figura 3.12: Código adjunto.

4 Warping e Controle

A simulação de fluidos tem um grande potencial para produzir deformações complexas em imagens com a aparência de um líquido, como já vimos até agora. No entanto, a fim de ser útil, temos que ser capazes de controlar a simulação de tal forma de atingir a transformação desejada.

4.1 Fins do Warping

As finalidades do warping tratados aqui na tese são:

1. Animação da imagem.
2. Deformações específicas da imagem, por exemplo: redução ou aumento de um objeto ou região da imagem, ver figura 4.1.
3. Morphing ou metamorfose entre imagens.

Para tais fins apresentaremos três técnicas de deformação na seção 4.3.

(a) Aumento. **(b)** Imagem Original. **(c)** Redução

Figura 4.1: Deformação do chapéu de Van Gogh.

4.2 Simulação Direta, Específica e Interativa

Esta parte do capitulo está dividida em quatro partes. Definimos alguns casos de simulações usadas no trabalho, descrevemos as técnicas de controle do *warping* e depois fazemos uma descrição detalhada da interface que conecta o usuário e a simulação, isto é, os parâmetros de entrada para cada técnica de controle e finalmente os detalhes teóricos.

Lembrando que o fluido está definido sobre o domínio da função imagem, e as forças $f(x,t)$ estão definidas sobre esse domínio também. Temos que as forças são variáveis no espaço e no tempo. As forças estão presentes em todas as técnicas de controle, as vezes são parâmetros aplicados pelo usuário durante a simulação de fluidos, em outras já está determinado o tempo e a força que deve ser aplicada na simulação. Dependendo do anterior as simulações são classificadas, pela forma como são aplicadas as forças, e tempo total da simulação, assim:

Simulação Direta: A simulação que ocorre com as forças já determinadas, agindo em instantes de tempo específicos sem adição de forças aleatórias.

Simulação Direta Específica: Nesta simulação está determinado o objetivo de deformação da imagem e o tempo final da simulação T. Esta simulação é uma *simulação direta* e atinge o objetivo de deformação no final da simulação, isto é, no tempo $t = T$.

Simulação Interativa: Dado um objetivo de deformação da imagem o usuário aplica forças arbitrárias variáveis no tempo durante a simulação até conseguir o efeito desejado de deformação.

4.3 Técnicas de Controle

Apresentamos aqui três técnicas desenvolvidas para o controle da deformação de imagens usando dinâmica dos fluidos. Lembramos primeiro que dada

uma imagem queremos deformá-la, temos que um fluido que está definido sobre o mesmo domínio da imagem.

4.3.1 Técnica de Viscosidade e Forças (VF)

Nesta técnica podemos manipular as condições iniciais da simulação tais como viscosidade e forças para controlar a deformação da imagem.

A simulação de fluidos é altamente caótica [44], isto é, pequenas mudanças nos parâmetros da simulação podem produzir resultados drasticamente diferentes. Por exemplo para uma viscosidade muito baixa, o resultado ao aplicar qualquer força é imprevisível. Ver figura 4.2a.

Já para uma viscosidade um pouco maior, o fluido oferece maior resistência à força aplicada. Assim quanto maior a viscosidade menor será a velocidade em que o fluido se movimenta. Então um parâmetro de controle da simulação é a viscosidade. É aqui onde a viscosidade adquire alta importância e sentido, e se converte numa poderosa ferramenta de controle da simulação. Ver figura 4.2.

(a) visc=0.001. **(b)** visc=0.002. **(c)** visc=0.005. **(d)** visc=0.010.

Figura 4.2: Para todos os casos de viscosidade a força usada é a mesma.

Agora para uma viscosidade pequena, uma força muito pequena pode apresentar uma mínima deformação e para uma força maior o resultado é aleatório. Assim a força com sua magnitude e sua direção é também um parâmetro de controle e dependendo do resultado desejado de deformação, temos que:

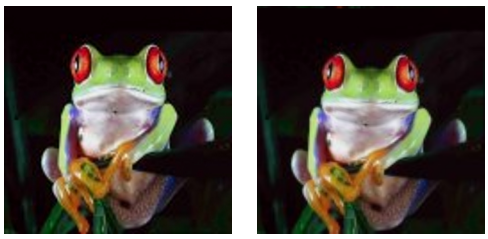
1. Para uma animação as, forças são aplicadas sistematicamente em tempos determinados, criando o movimento. Portanto é uma simulação direta,

onde o final da simulação ou animação é determinado pelo usuário. Ver figura 4.3.

(a) t = 1. **(b)** t= 3. **(c)** t = 5. **(d)** t = 8.

Figura 4.3: Animação da cabeça da Mona Lisa.

2. Para uma deformação, o usuário aplica as forças de forma interativa, adicionando forças através do tempo até conseguir o resultado desejado. Ver figura 4.4.



(a) Imagem Original. **(b)** Deformação.

Figura 4.4: O objetivo é abaixar a cabeça do sapo. Aqui temos uma viscosidade média e uma força vertical de magnitude baixa, com direção para baixo. Simulação deste exemplo é interativa.

4.3.2 Técnica de Viscosidade Variável (VV)

Esta técnica usa de novo a viscosidade como parâmetro de controle, mas aqui a viscosidade é variável no espaço, ao contrário da técnica VF. Aqui buscamos uma forma de definir os valores da viscosidade em cada ponto do domínio para usar os valores na deformação da imagem. Um caminho natural para definir a viscosidade é usar imagens auxiliares.

Nesta técnica a viscosidade é definida sobre regiões, assim:

- Se especifica menor viscosidade na região onde desejamos maior movimento.
- Se especifica maior viscosidade, no lugar que se deseja menor movimento ou deformação.

Por exemplo, consideremos a imagem do Spock (figura 4.5a). Deformaremos a imagem de tal forma que baixamos a sobrancelha um pouco.

(a) Imagem Original. Spock. **(b)** Região de pouca viscosidade. **(c)** Detalhe da região. **(d)** Região com alta viscosidade.

Figura 4.5: Definição da viscosidade.

Para isso, definimos com menor valor de viscosidade uma pequena região embaixo da sobrancelha (figura 4.5b). No resto da imagem definimos uma viscosidade alta procurando ter baixa influiência do fluido ou idealmente não ter influiência nos pontos desta região (figura 4.5).

Agora as forças $\mathbf{f}(\mathbf{x},\mathbf{t})$ também são variáveis no domínio da imagem e portanto também podem ser especificadas por imagens auxiliares.
A simulação aqui é interativa, ou seja, o usuário aplica a força até chegar ao resultado desejado.

4.3.3 Técnica de Keyframe e Partículas (KP)

Esta técnica de controle considera o domínio da imagem como um fluido homogêneo, incompressível, de *viscosidade constante*. Permite ao usuário controlar a deformação usando pontos. Especifica-se a posição de dois pontos na imagem: a posição de um ponto de saída (o ponto na imagem que queremos transladar a um outro lugar para modificar a imagem) e um ponto de chegada (ponto que deve atingir a deformação). O *keyframe* é o conjunto de pontos

de chegada. A coordenada no ponto de saída é transportado até o ponto de chegada pelo fluido. A simulação é controlada então por keyframes no tempo final.

Para conseguir que o fluido transporte o ponto de saída até o keyframe, é utilizado um processo iterativo continuo de otimização quase-Newton. Tal processo acha as forças apropriadas para ser aplicadas sobre o campo de velocidades através da simulação.

Queremos que a simulação se aproxime no máximo possível do keyframe. Para isso, minimizamos uma função real φ. A função real φ, chamada de função objetivo é definida como a distância entre a simulação e o keyframe. (Observamos que a função objetivo depende de toda a simulação).

Para minimizar a função objetivo e achar as forças apropriadas para atingir o keyframe devemos calcular o gradiente da função φ e portanto derivar a simulação. Para calcular as derivadas através da simulação usamos o Método Adjunto, eficiente e rápido é a ferramenta mais completa do trabalho.

4.4 Especificação de Parâmetros

Para cada técnica anteriormente mencionada existe um conjunto de parâmetros que o usuário pode definir como entrada. Eles permitem manipular a técnica e com isso a controlar a simulação.

4.4.1 Parâmetros da Técnica VF

Nesta técnica a interface entre o usuário e o controle da simulação são os parâmetros da viscosidade e a força.

Viscosidade:

Temos que se maior é a viscosidade então maior é o controle. Mas a viscosidade não pode ser muito alta, para que exista movimento e as deformações sejam significativas (ver figura 4.6). A viscosidade é uma poderosa ferramenta

de controle e pode dar o efeito de gelatina ou de um material elástico na simulação, mas é o usuário que deve decidir o valor dela dependendo do resultado que deseje.

(a) Pouca viscosidade. **(b)** Viscosidade muito alta. **(c)** Imagem Original.

Figura 4.6: Viscosidade parâmetro de controle da simulação de fluidos. Nas simulações 4.6a e 4.6b foi aplicada a mesma força.

Forças:

Há vários tipos de forças que podem ser usadas no controle da animação ou deformação de imagens.

Forças numa Direção São forças compostas por um vetor ω escalado por uma função de queda Gaussiana. Seja ω é a direção e c o centro da Gaussiana, definimos uma componente da força sobre a malha como

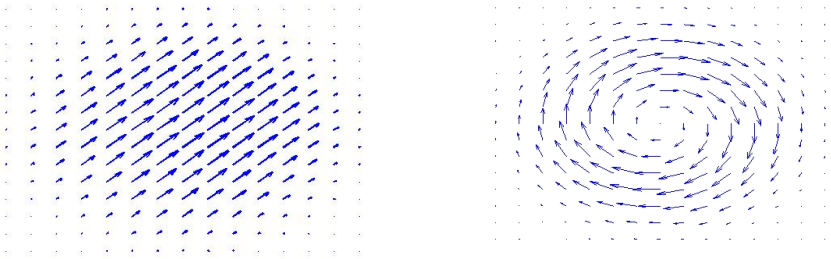
$$(f_\omega)_{ij} = G_{ij}\omega$$

onde $G_{ij} = e^{-a|\Delta c|^2}$, $\Delta c = [i,j]^T - c$ e a determina a amplitude da Gaussiana. Ver figura 4.7a.

Forças de Vorticidade Para estas forças usamos de novo a função de queda Gaussiana e um parâmetro r que controla a quantidade de força rotacional que deve ser aplicada, temos que uma componente da força sobre a malha é dada por

$$(f_v)_{ij} = rG_{ij}R\Delta c, \quad \text{onde } R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

é a matriz de rotação de angulo $\pi/2$ e G_{ij} e Δc estão definidas antes. Figura 4.7b.



(a) Forças numa Direção. **(b)** Forças de Vorticidade.

Figura 4.7: As forças estão discretizadas sobre a mesma grade da velocidade.

Força Osciladora Para criar animação na imagem usamos uma força osciladora que atua ao longo da simulação, movimentando continuamente a imagem. Usamos a função osciladora dada por

$$f_{osc}(t) = A \ \mathrm{sen}(tk)$$

com amplitude A e frequência $k = 1/T$ onde T é o período de oscilação. A força osciladora está definida como um campo vetorial de forças vezes a função osciladora.

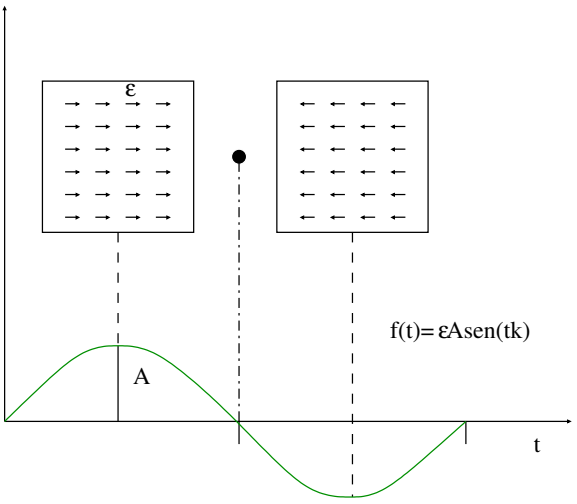
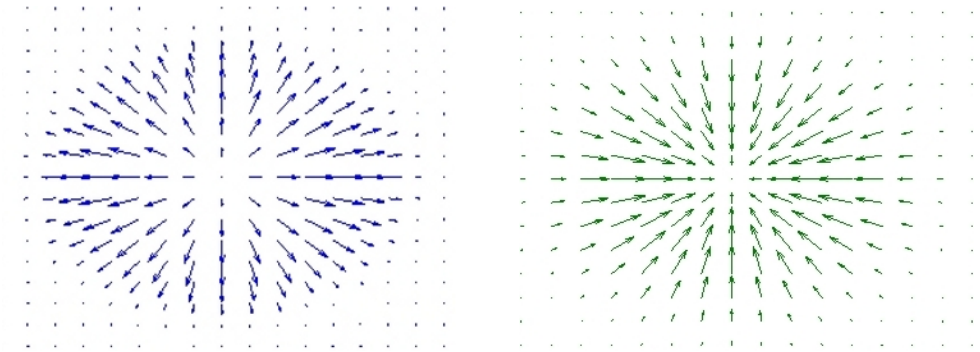


Figura 4.8: Força Osciladora.

Forças de Expansão e Redução Estas forças expandem ou reduzem a imagem respectivamente e estão dadas por



(a) $(f_\omega)_{ij} = G_{ij}\Delta c$. **(b)** $(f_\omega)_{ij} = G_{ij}(-\Delta c)$.

Figura 4.9: As forças estão discretizadas sobre a mesma grade da velocidade. Onde 4.9a é a *força de expansão* e 4.9b é a *força de redução*.

Outras forças: Definidas por um vetor, o usuário especifica a direção, pode ser um vetor vertical, horizontal ou qualquer outra direção e magnitude. As forças como já vimos podem ser classificadas segundo o tempo:

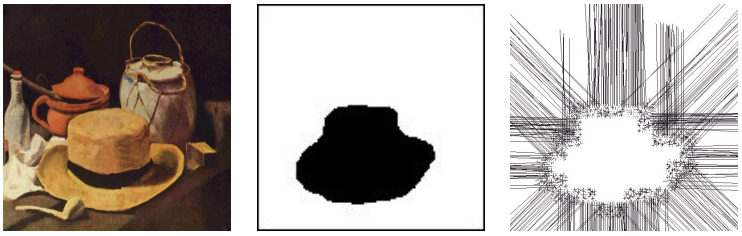
- Instantâneas.
- Constantes em certo intervalo de tempo.
- Arbitrárias, variáveis no tempo.

4.4.2 Parâmetros da Técnica de Viscosidade Variável

Nesta técnica, da mesma forma que na técnica VF, os parâmetros de controle são a viscosidade e a força. Já que a viscosidade é variável as equações de Navier-Stokes mudam para equações com mais termos. A definição dos valores da viscosidade é mais complexa e contudo é intuitiva. A força e a viscosidade são associadas a imagens auxiliares e faz da definição dos parâmetros uma tarefa natural para o usuário.

Definição de Parâmetros por Imagens. O usuário dá como parâmetros de entrada três imagens:

1. A imagem para deformar.
2. Uma imagem que especifica a viscosidade.
3. Uma imagem que especifica as forças.



(a) Imagem Original. **(b)** Viscosidade **(c)** Forças

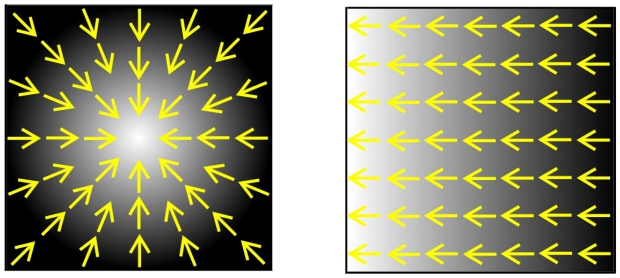
Figura 4.10: Parâmetros.

Viscosidade

Definimos a viscosidade por meio de uma função imagem auxiliar. Os valores da viscosidade são calculados da normalização dos valores da imagem em [0,1] vezes um fator global de escalamento. Então para uma imagem em tons de cinza, podemos definir como a região mais viscosa a parte clara da imagem e a região menos viscosa a parte escura da imagem.

Forças

Para definir o campo vetorial das forças calculamos o gradiente da intensidade da função imagem auxiliar I. O gradiente do campo escalar I é um campo vetorial que indica em cada ponto do campo escalar a direção do aumento máximo de I. Logo o campo vetorial esta dirigido da intensidade baixa para a intensidade alta, isto é, para uma imagem em tons de cinza, o campo vetorial vai da parte escura para a parte clara da imagem.



(a) (b)

Figura 4.11: Nas duas imagens acima (a) e (b), o campo escalar é indicado pela escala de cinzas, onde branco representa o maior valor, e seu gradiente correspondente é representado por setas amarelas.

Surpreendentemente, apenas este pequeno conjunto de parâmetros já fornecem mecanismos poderosos e intuitivos para o controlar a deformação de imagens.

4.4.3 Parâmetros da Técnica KP

As ferramentas que o usuário possui para manipular esta técnica são pontos, diferente das técnicas anteriores que manipulavam propriedades físicas da simulação de fluidos.

Dada uma imagem, o usuário possui um controle pontual da simulação, e localiza o ponto que quer modificar da imagem e logo o ponto até onde ele quer que chegue no final da simulação, o conjunto de pontos de chegada é o keyframe. Assim o usuário pode localizar todos os pontos que deseje sobre a imagem, em conjuntos de dois, um ponto de partida e um de chegada.

4.5 Teoria das Técnicas

Identificamos três técnicas básicas de controle da deformação de imagens usando fluidos. Consideremos cada uma e vejamos sua estrutura de funcionamento.

Lembremos que a ideia é pensar na imagem como um fluido 2D homogêneo é incompressível. Depois usar as equações de Navier-Stokes para criar um campo vetorial e modificar a imagem, aplicando forças sobre o domínio. O

processo transporta as coordenadas da parametrização da imagem através do campo vetorial gerado pelas equações.

4.5.1 Técnica de Viscosidade e Forças (VF):

Nesta técnica o warping é controlado pelas propriedades físicas, tais como viscosidade e forças aplicadas sobre o domínio da imagem. O estado do sistema físico em algum tempo t é descrito por um conjunto das coordenadas da imagem x e as velocidades do fluido v, definido sobre uma malha quadrada de $(N+2)\times(N+2)$ células

$$q = (x, v).$$

Para um tempo qualquer t a cada passo de tempo Δt atualizamos os valores das coordenadas e as velocidades a través das equações de Navier-Stokes e assim obtemos o estado do sistema no tempo $t+\Delta t$. O esquema para calcular as equações é descrito na seção 3.4.3 e está composto de 4 etapas: adição de forças externas, advecção, difusão e projeção. A estrutura da simulação que calcula os valores das velocidades e as coordenadas, para um passo do tempo Δt é a seguinte.

Estrutura:

1. Calcular forças.
   Nesta parte as forças são calculadas
   Exemplo para o caso de forças em uma direção:
   for $i, j = 1 to\ N$ *do*
   $(f_\omega)_{ij} = G_{ij}\omega$

2. Para um passo de tempo Δt temos que calcular as equações de Navier-Stokes (4.1), o processo para calcular as equações esta dado em 4 etapas.

$$u_t = P(-u\cdot\nabla u + \mu\Delta u + f) \tag{4.1}$$

$$x_t = -u\cdot\nabla x + \mu\Delta x \tag{4.2}$$

onde $x = (x, y)$ e $u = (u, v)$

(a) Adição das forças ao campo de velocidades

$for\ i,j = 1,..,N\ do$

$u^1_{ij} = u^0_{ij} + fx$

$v^1_{ij} = v^0_{ij} + fy$

(b) Cálculo da difusão

$for\ i,j = 1,..,N\ do$

$u^2_{ij} - u^1_{ij} = \Delta t \mu \Delta u^2_{ij}$

$v^2_{ij} - v^1_{ij} = \Delta t \mu \Delta v^2_{ij}$

(c) Advecção

$u^3 = -[(u^2, v^2) \cdot \nabla] u^3$

$v^3 = -[(u^2, v^2) \cdot \nabla] v^3$

(d) Projeção

Achar a solução de $\Delta q = \nabla \cdot (u^3, v^3)$ e logo

$(u^4, v^4) = (u^3, v^3) - \nabla q$

3. Para o passo de tempo Δt atualizamos a posição das coordenadas de textura, para isso resolvemos a equação (4.2) em 2 etapas.

(a) Cálculo da difusão das coordenadas de textura

$for\ i,j = 1,..,N\ do$

$x^1_{ij} - x^0_{ij} = \Delta t \mu \Delta x^1_{ij}$

$y^1_{ij} - y^0_{ij} = \Delta t \mu \Delta y^1_{ij}$

(b) Advecção das coordenadas através do campo de velocidades

$x^2 = -[(u^4, v^4) \cdot \nabla] x^2$

$y^2 = -[(u^4, v^4) \cdot \nabla] y^2$

4. Finalmente atualiza o tempo

$t = t + \Delta t$.

Observações Nesta técnica existe controle da simulação dado pela viscosidade. Sem viscosidade a simulação perde estabilidade, isso dá sentido ao seu uso.

O modelo é limitado porque usa uma simulação de fluidos. Por outro lado, ganhamos plausibilidade pela natureza física.

O limite é visível em animação, depois de certo tempo de estar rodando a simulação a imagem começa a deformar-se. Ver figura 5.2b. Ainda utilizando uma viscosidade alta temos este fenômeno. O anterior é devido a que as equações são dissipativas e existem erros numéricos de truncamento por aproximação das mesmas equações.

Os resultados de animação usando esta técnica mostram um efeito de aparência *natural* ou *orgânica*, uma animação *real* de imagens e isso é pelo uso de dinâmica dos fluidos.

4.5.2 Técnica de Viscosidade Variável (VV):

Esta técnica é mais precisa que a técnica VF para alguns casos de deformação e usa o mesmo esquema de VF : adição de forças, difusão, projeção e advecção. Mas o cálculo da difusão é diferente, além da forma de definir a viscosidade e a força.

A viscosidade aqui é um campo escalar representado por valores sobre a malha regular $N \times N$. Dado que a viscosidade varia no espaço, temos que o termo de difusão esta dado pelas equações:

$$\partial_t w = \nabla \mu(\bar{x})(\nabla w + \nabla w^\top) + \mu(\bar{x}) \triangle w \tag{4.3}$$

onde

$$\nabla w + \nabla w^\top = \begin{pmatrix} 2\partial_x w^1 & \partial_x w^2 + \partial_y w^1 \\ \partial_y w^1 + \partial_x w^2 & 2\partial_y w^2 \end{pmatrix}.$$

Na discretização do modelo as equações são substituídas por esquemas de diferença finita implícitas sobre a malha. Os esquemas implícitos fazem que o modelo continue estável, como no caso da viscosidade constante.

A viscosidade é definida usando uma imagem auxiliar. Para cada ponto x_i da malha existe um ponto y_i sobre o domínio da imagem, tal que a intensidade da imagem escalada a valores de $[0,1]$ vezes um fator global é igual a viscosidade no ponto x_i.

A imagem auxiliar que define a viscosidade é criada com relação à imagem

que queremos deformar e o elemento ou região que queremos deformar. A magnitude e direção da força é regulada pela mesma viscosidade.

4.5.3 Técnica de controle Keyframe de Partículas (KP):

Até agora, nós adotamos apenas a especificação de forças e as variações na viscosidade para controlar o warping da imagem. Previamente com uma viscosidade estabelecida e um campo de forças específico. A partir deste momento, usaremos só especificação de keyframes para controlar a simulaçãode fluidos.

A técnica começa quando o usuário especifica um conjunto P de pontos sobre o domínio da imagem que quer modificar. Para cada ponto em P especifica também uma posição ou ponto de chegada que deve tratar de alcançar num tempo T. O keyframe é o conjunto K de pontos de chegada e chamamos T, tempo final da simulação.

A simulação de fluidos de n passos inicia com um estado q_0 no tempo $t = 0$ e repetidamente aplica a função S que modela as equações de Navier-Stokes. Em cada passo de tempo Δt definimos

$$q_{k+1} = S(q_k),$$

onde $q_k = (x(k\Delta t), v(k\Delta t))$.

E deste modo calculamos o conjunto de estados $q_1, .., q_n$, temos uma sequencia de estados que define uma simulação $\mathscr{L}$. Escrevemos para cada estado q_0, a simulação $\mathscr{L}(q_0) = (q_1, .., q_n)$.

Nas técnicas anteriores a força é especificada pelo usuário, isto é, as forças são dadas explicitamente. Aqui em vez disso, devemos achar o conjunto de forças que nos levam até o objetivo. As forças são armazenadas num vetor u.

Dado qualquer valor de u, este define um único conjunto de forças aplicadas ao campo de velocidades e portanto uma única simulação $\mathscr{L}(u, q_0)$. O estado da simulação em nosso caso são as posições dos pontos de saída, ou seja o conjunto P. Queremos medir o quão perto está P do keyframe K no tempo T,

para isso definimos a função objetivo.

Função Objetivo. Queremos que o estado da simulação de fluidos alcance o keyframe tão perto quanto seja possível. Portanto definimos uma função escalar $\varphi(\mathscr{L}(u,q_0))$ que depende de toda a simulação, e mede a diferença entre o keyframe e o correspondente estado da simulação. Chamamos esta função de função objetivo.

Esta função definida para o caso de controle de simulação de fumaça por (Treuille em 2003 [44]) desenvolvida logo por (McNamara em 2004 [30]) para o caso de fumaça e água. Neste trabalho consideramos uma função objetivo da forma:

$$\varphi(u) = \varphi(\mathscr{L}(u,q_0)) = \frac{1}{2}\sum_{t\in C_p}\left(\| (q_t - K_t) \|^2\right) \tag{4.4}$$

onde C_p é o conjunto de passos de tempo com keyframe.

Esta nova versão da função objetivo é diferente das anteriores. Primeiro não possui o termo φ_s de "suavidade ", termo que nos trabalhos anteriores mede a quantidade de força u que adicionada durante a simulação. E adicionalmente trabalha avaliando a diferença do estado e o keyframe, sem pre-processamentos. Fato que no trabalho de Treuille e de McNamara não é muito robusto.

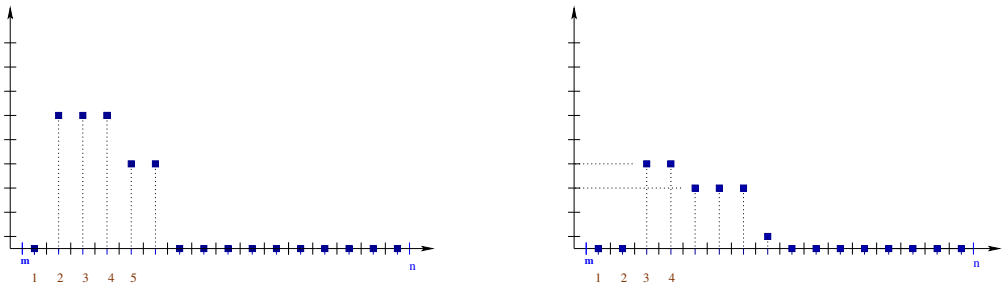
Função Objetivo e a diferença $\| (q_t - K_t) \|^2$ em *Keyframe Control of Smoke Simulations* [44] e *Fluid Control Using the Adjoint Method* [30].

Temos que a função objetivo mede o tanto a simulação emparelha com o keyframe. Isto é, φ mede o *erro* entre o keyframe e o estado correspondente da simulação. Supondo que o estado q_t no tempo t deve concordar com o keyframe K_t. A métrica natural está dada por $\| (q_t - K_t) \|^2$. Mas esta definição não funciona na hora de achar o mínimo da função objetivo.

Vejamos porque não funciona com um exemplo simplificado de uma simulação de fluidos sobre um domínio 1D.

Consideremos um fluido um intervalo $[m,n]$. Dividimos esse domínio em N partes iguais. Para representar finitamente o domínio pegamos o centro de

cada subintervalo. E transportamos sobre o fluido um material qualquer que e denotaremos por ρ. Este material ρ é representado por um campo escalar. Ver figura 4.12.



(a) ρ em $t1$. **(b)** ρ em $t2$.

Figura 4.12: Campo escalar ρ .

Definimos um keyframe K também como um campo escalar, para ser atingido pelo estado da simulação ρ no tempo t_k. Figura 4.13.

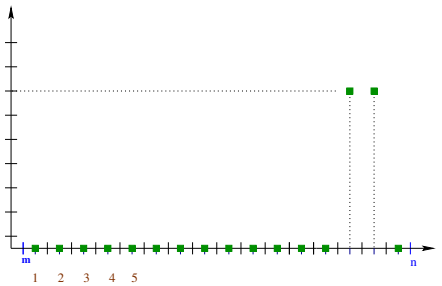
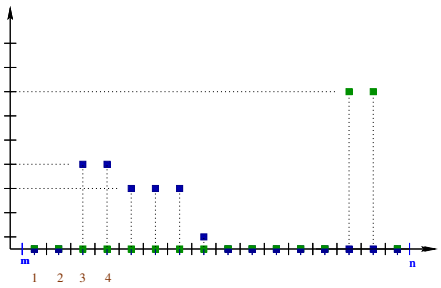


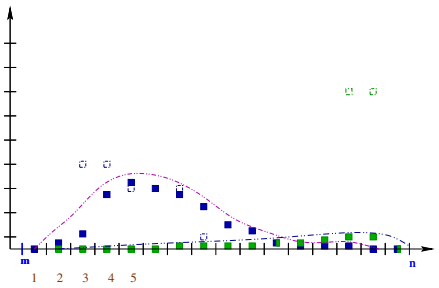
Figura 4.13: Campo escalar do Keyframe.

Imagine que no tempo t_k os valores diferentes de zero de ρ_{t_k} e K não se cruzam. Temos que uma pequena perturbação no estado não muda o valor do erro. Figura 4.14a. Isto é problemático porque desejamos que a direção do gradiente da função objetivo nos leve até a solução. Para solucionar este problema, suavizamos o estado e keyframe antes de avaliar a função. Figura 4.14b.

Todo o anterior mostra como é definida a função objetivo nos trabalhos de Treuille e McNamara e a relação com a diferença $\| (q_t - K_t) \|^2$. Nesta tese a função objetivo 4.5.3 é basicamente essa diferença, não fazemos um processamento porque φ mede a distancia entre as posições de pontos, e não compara campo escalares. Portanto φ é muito mais simples é intuitiva.



(a) Se os valores diferentes de zero do keyframe e o estado não se cruzam, então $\| (q_t - K_t) \|^2$ não muda com pequenas perturbações.



(b) A solução para o problema 4.14a é suavizar ou borrar o keyframe e o estado, e logo fazer a diferença e a metrica final é $\| B(q_t - K_t) \|^2$

Figura 4.14

Processo de Otimização e Método Adjunto.

Para esta técnica de controle de simulação de fluidos, já definimos o keyframe e os pontos de saída e a função objetivo φ. O seguinte passo é achar o mínimo da função φ. Portanto devemos achar as forças que aplicadas ao campo vetorial de velocidades da simulação, levem os pontos de saída até o keyframe a traves do fluido.

Para minimizar φ usamos uma técnica de otimização de memoria limitada quase-Newton (ver Zhu 1997[49]) a qual só precisa avaliar a função e seu gradiente.

Já que precisamos de calcular o gradiente da função objetivo e esta depende de toda a simulação então devemos derivar cada passo da simulação, isto é, derivar a adição de forças, a projeção, a advecção e a difusão, para achar o gradiente. Quem faz esse trabalho todo é o método adjunto que calcula de forma eficiente e rápida o gradiente da função objetivo.

Derivada da Simulação. O processo de derivação é o seguinte, os estados $q_1, .., q_n$ são calculados e cada estado é armazenado. O método adjunto cria a partir do código da simulação um código adjunto tal que para cada instrução no código da simulação existe uma instrução *adjunta* no código adjunto.

Mas as instruções no código adjunto estão na ordem inversa das instruções

do código da simulação. Criadas pelas regras já vistas nas seções 3.5.4 e 3.5.4.

Podemos notar cada instrução adjunta colocando um a antes do nome da instrução do código da simulação.
Assim em resumo, do código da simulação temos que as instruções são:

```
adição_de_forças_externas
advecção
difusão
projeção
```

Então o código adjunto é

```
a_projeção
a_difusão
a_advecção
a_adição_de_forças_externas
```

Os estados armazenados na simulação são usados no código adjunto para calcular os estados adjuntos $aq_n, .., aq_1$. Estos estados são usados finalmente para obter o gradiente da função objetivo.

Estrutura da Técnica.
Para qualquer vetor de forças u. Consideremos a estrutura, que reúne os processos: de simulação, armazenamento dos estados e cálculo dos estados adjuntos. Para qualquer u temos que a estrutura que avalia φ e $\dfrac{d\varphi}{du}$ é a seguinte:

Avaliar φ e $\frac{d\varphi}{du}$

- Os passos da simulação de fluidos são:
  Para $t = 1$ até N

  1. f $\leftarrow$ cálculo de forças(t,u)
  2. v $\leftarrow$ adição de forças(v,f)

3. v ← difusão(v)
4. v ← advecção(v,v)
5. v ← projeção(v)
6. x ← advecção(v,x)
7. armazena q_t

- Já calculada toda a simulação e armazenados os estados devemos avaliar φ
- O método adjunto calcula os estados adjuntos
  Para $t = N$ até 1

  1. av ← a_advecção(av,ax)
  2. av ← a_projeção(av)
  3. av ← a_advecção(av,av)
  4. av ← a_difusão(av)
  5. av ← a_adição de forças(av,af)

- Finalmente achamos o gradiente de φ,
  logo devemos avaliar $\dfrac{d\varphi}{du}$.

Em vista disto, o processo de otimização tem a seguinte estrutura.

Otimização.

Para $\varepsilon \geq 0$ escolhido pelo usuário

1. Atribuímos valores, logo $u = \dfrac{d\varphi}{du} = 0$
2. Seguimos aqui o processo anterior de avaliar φ e $\dfrac{d\varphi}{du}$
3. • Enquanto $\varphi > \varepsilon$ então
     – u ← método quase-Newton(φ,$\dfrac{d\varphi}{du}$)

– $(\varphi, \frac{d\varphi}{du}) \leftarrow$ avaliar(u)

Todo o anterior resume a técnica de controle KP.

5 Resultados

Nesta seção damos alguns exemplos e resultados obtidos com a ferramente de fluid warping e as técnicas de controle desenvolvidas.

5.1 Técnica VF

O primeiro conjunto de exemplos são resultados obtidos pela técnica VF e ilustra como os parâmetros de forças e viscosidade controlam a deformação.

A figura 5.1 apresenta uma animação de um sapo que move a parte do gargalo verticalmente, dando um efeito natural similar a um sapo real. Para este exemplo usamos uma viscosidade constante, uma força osciladora com frequência alta e a magnitude da força vertical é baixa, dando o efeito sutil de um movimento vertical.
A figura 5.1 exibe seis tempos diferentes da animação em ordem crescente e um gráfico dos parâmetros usados.

As seguintes figuras em 5.2 exibem os casos onde o limite do modelo é quebrado e os resultados são deformações não desejadas da imagem. Na figura 5.2a vemos o caso de viscosidade muito baixa, na ação da força, a imagem se deforma sem voltar até o estado original. A distorção da figura 5.2b ocorre quando a simulação roda por um período muito longo de tempo. Da mesma forma existe distorção nos casos de frequência baixa 5.2c e amplitude alta 5.2e.

Outra animação é mostrada na figura 5.3. Aqui uma risada de Oliver Hardy na personagem do ¨Gordo¨. A animação é criada a partir da imagem do sorriso, usando uma viscosidade média, localizamos os centros de duas forças verticais em cada bochecha.

Uma animação diferente da personagem do Gordo é apresentada na figura 5.4. Aqui, o movimento se centra na boca, um campo na direção diagonal move de lado a lado a boca.

Outro exemplo de animação diferente aos anteriores é mostrado na figura 5.5. O quadro de Starry Night de Van Gogh é animado localizando uma serie de forças de vórtice em cada estrela e na lua.

5.2 Técnica VV

O segundo grupo de exemplos ilustra a técnica de viscosidade variável.

A figura 5.6 exibe a deformação do quadro de Van Gogh. A figura 5.6c mostra as forças aplicadas e a figura 5.6d é a imagem que define a viscosidade. As forças provêm de uma segmentação do chapéu e a viscosidade de uma quantização dos valores da imagem. Note que as forças atuam para expandir o chapéu mas ao redor a viscosidade é variável logo o chapéu se deforma de maneira não uniforme.

Uma outra deformação do chapéu com a mesma força inicial, mas com outra definição de viscosidade e esta dada por 5.7.

O seguinte exemplo é uma tentativa de avaliar a técnica fluid warping com uma técnica já estabelecida de warping. Para este fim, fazemos uma comparação com um exemplo do artigo de Arad [4]. Nesse trabalho, os autores descrevem um técnica baseada em funções de base radial.

A figura 5.8 é um exemplo em [4] para levantar o canto da boca da menina. Para conseguir esse efeito, construímos uma função viscosidade, mostrada na figura 5.8b, que impõe uma restrição na área de deformação. Nesta função, a região fora da deformação desejada é branca e portanto mais viscosa (ou seja, opondo grande resistência ao movimento do fluido). A área de deformação é escura e menos viscosa (ou seja, oferece pouca resistência ao movimento de fluidos). As forças utilizadas no processo são dadas pelo campo gradiente da imagem mostrada na figura 5.8c. Elas exercem uma força ascendente no local da boca, produzindo o efeito desejado. Os resultados nas figuras 5.8d e 5.8e

demonstram que somos capazes a aproximar-se a essa técnica. Um resultado semelhante é alcançado no exemplo da figura 5.9.

5.3 Técnica KP

O terceiro conjunto de resultados são obtidos pela técnica de controle por keyframe.

A figura 5.10 ilustra a técnica. A Mona Lisa é deformada, localizando três pontos sobre a boca e um keyframe é definido para ser atingido pelos pontos. A viscosidade é constante e uma malha 5×5 de forças é estabelecida sobre o domínio da imagem. A figura 5.10b exibe o resultado.

5.3.1 Morphing

Expomos agora uma aplicação da técnica de controle usando keyframes. A figura 5.11 mostra um morphing entre a imagem da Mona Lisa e o sapo. Para este morphing, localizamos os pontos de saída sobre a Mona Lisa, i.e., pontos sobre os olhos, nariz, boca e cabeça. O keyframe é definido sobre o sapo. Pontos correspondentes a os pontos de saída ,i.e., sobre os olhos, nariz, boca e cabeça.
O método de otimização roda, e os pontos de saída são levados até os pontos de keyframe. Cada imagem deformada da Mona Lisa é guardada.
Agora os pontos do keyframe são os pontos de saída, e o novo keyframe são os antigos pontos de saída. De novo os processo de otimização é rodado, e agora o sapo se deforma até a Mona Lisa. Cada deformação é guardada, e posteriormente realizamos uma interpolação das imagens para obter o efeito de transição das mesmas.

5.4 Técnicas Híbridas

O uso particular de uma técnica tem bons resultados. Agora associar duas técnicas é possível. Ao associar duas técnicas o controle é mais estável,

aumentando a possíveis de deformações. Esta seção mostra os resultados de combinações das técnicas.

5.4.1 Viscosidade Variável e Viscosidade e Forças

Para os resultados que mostraremos em esta parte, usamos viscosidade variável para animar a cara da Mona Lisa e um força de vórtice com uma função osciladora. A ideia é mover só a cabeça da Mona Lisa e manter o fundo fixo. Por isso foi necessário o uso da viscosidade variável. Resultados na figura 5.12.

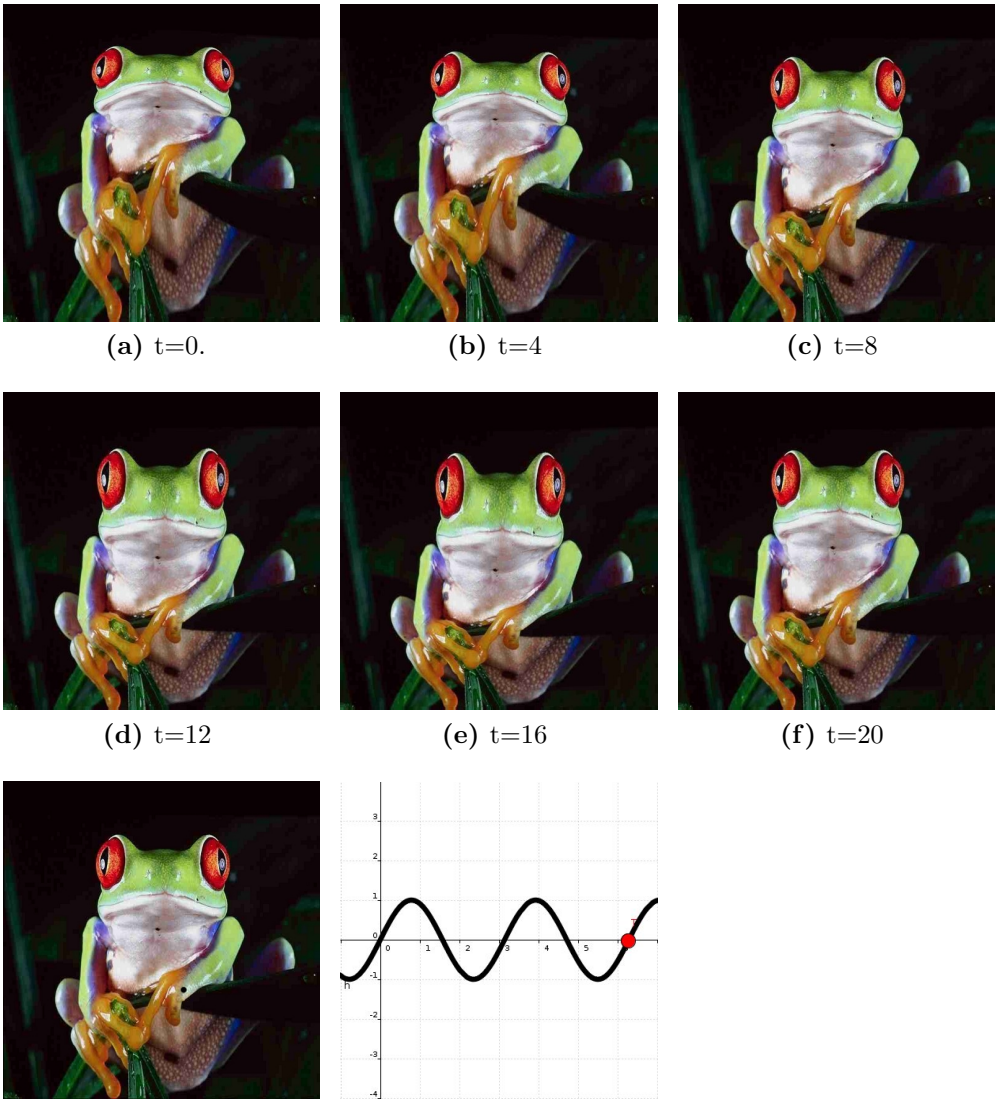
5.4.2 Viscosidade Variável e Keyframe e Partículas

Para esta técnica usamos de novo a viscosidade variável junto com o controle com keyframe. O objetivo é pegar a imagem de cara feliz e deformar a boca para uma cara triste.

Para isso localizamos pontos sobre a boca da figura e especificamos o keyframe. A viscosidade é especificada do seguinte jeito. A região da boca é menos viscosa porque nesta região onde vai ter maior deformação. Fora da cara a viscosidade a máxima possível porque queremos manter a forma redonda da cara.

A figura 5.13 os pontos de saída e keyframe, a viscosidade e o resultado da deformação. Na figura 5.14 ilustra o efeito da viscosidade sobre a simulação. Para cada caso os pontos de saída e keyframe são os mesmos, variando só a viscosidade.

5.5 Resultados da Técnica VF



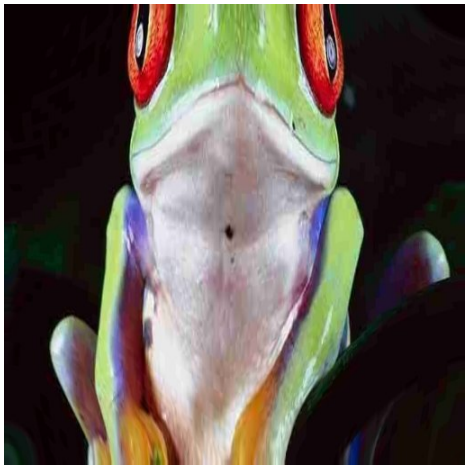
(a) t=0. **(b)** t=4 **(c)** t=8

(d) t=12 **(e)** t=16 **(f)** t=20

(g) Imagem original. **(h)** Força Osciladora

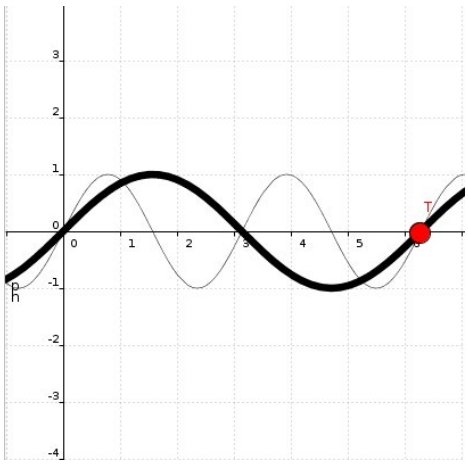
Figura 5.1: Animação Sapo. A partir da imagem do sapo 5.1g criamos uma animação, usando uma força osciladora 5.1h aplicada sobre o fluido, definido no domínio da imagem. As imagens 5.1a até 5.1f mostram diferentes tempos da simulação. O movimento é vertical e localizado no gargalo do sapo. Ver `http://w3.impa.br/~dalia/videos.html`

(a) Distorção por viscosidade baixa.



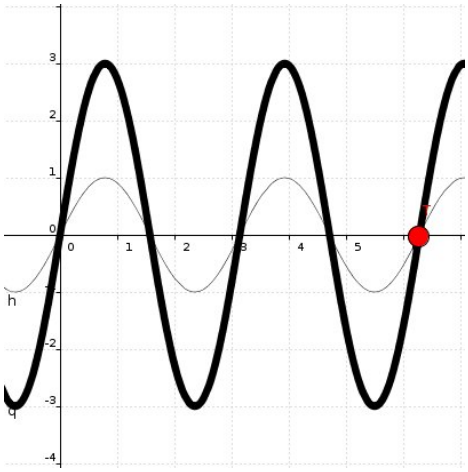
(b) A simulação roda por um período muito longo de tempo.

(c) Frequência baixa.



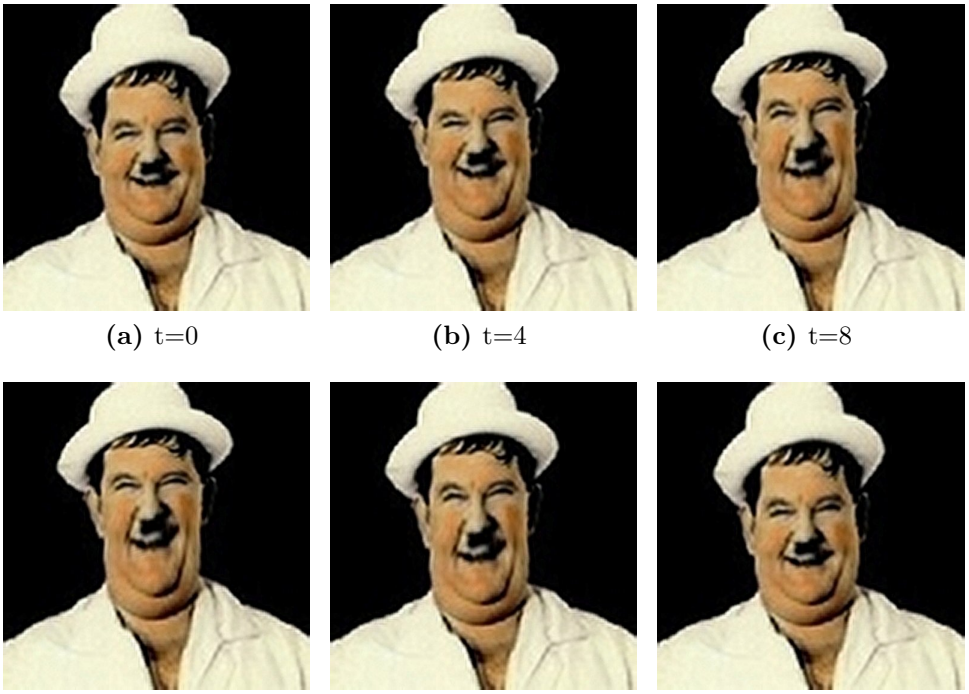
(d) Força osciladora de frequência baixa comparada com a frequência da animação 5.1h.

(e) Muita amplitude



(f) Força osciladora com amplitude maior que à amplitude da animação 5.1h.

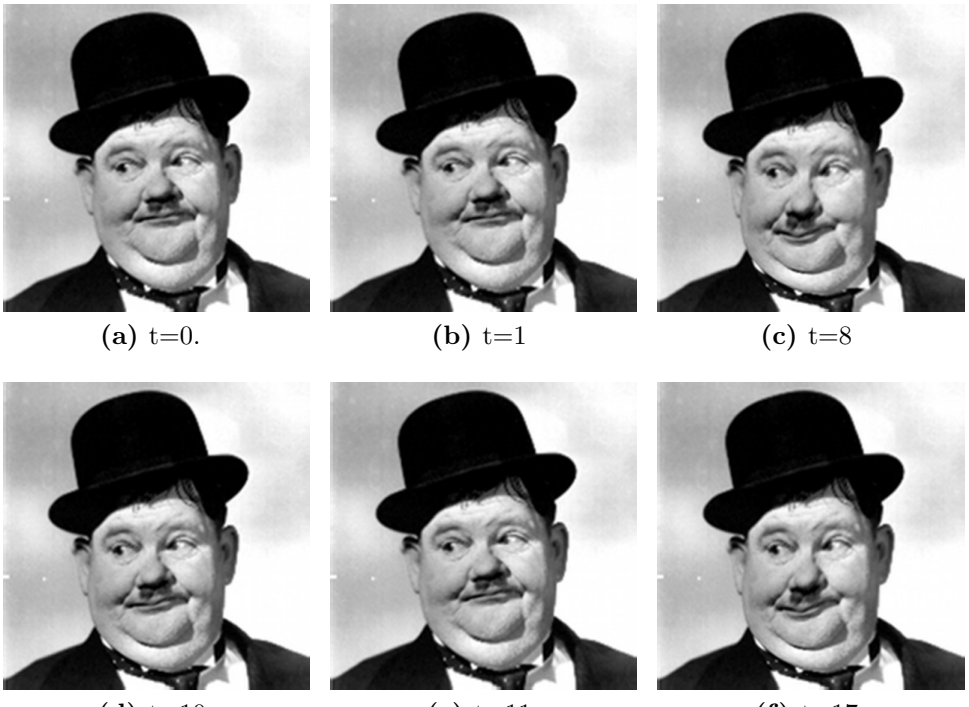
Figura 5.2: Limite do Modelo. Problemas de distorção da imagem.



(a) t=0 **(b)** t=4 **(c)** t=8

(d) t=12 **(e)** t=16 **(f)** t=20

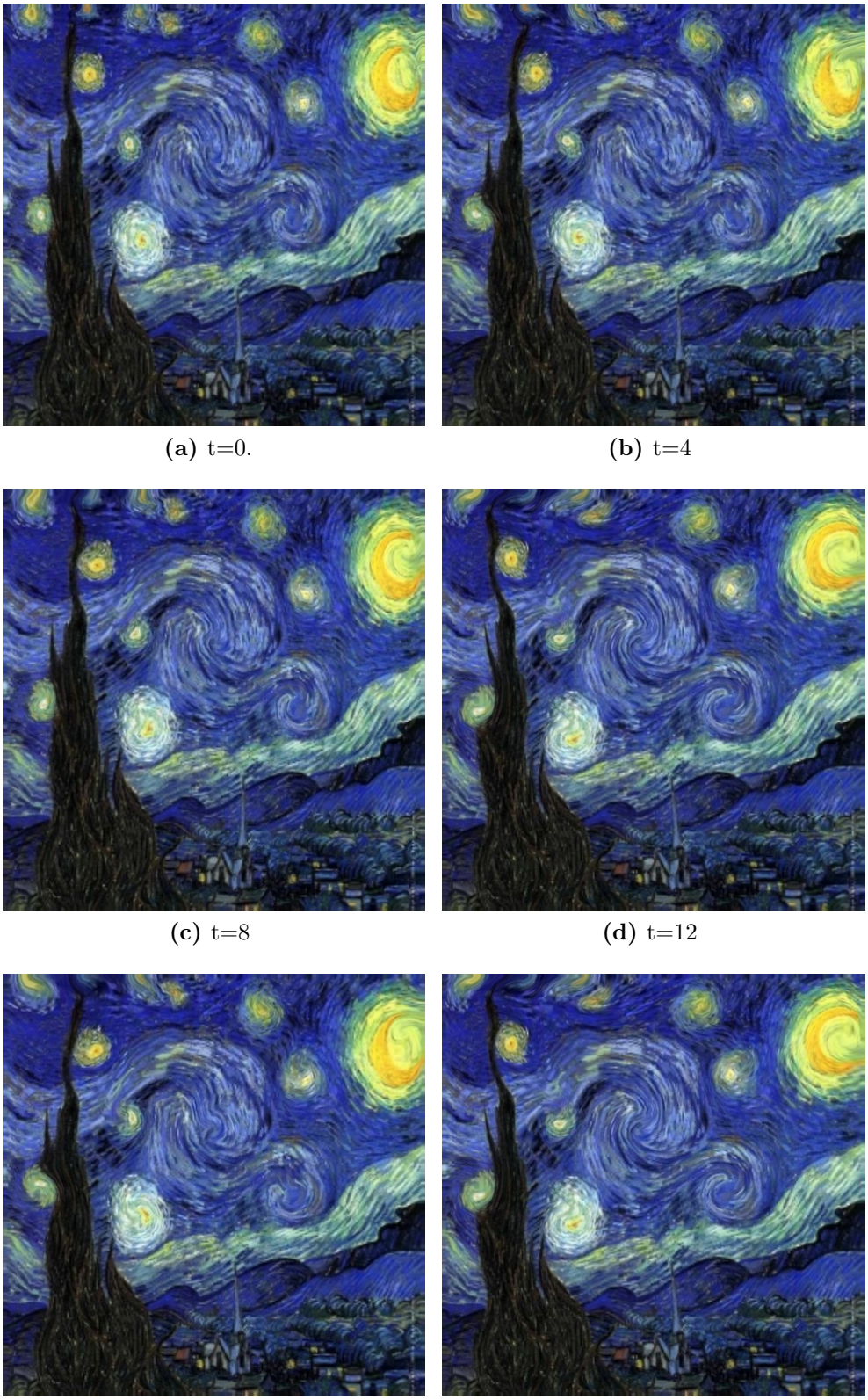
Figura 5.3: Animação do sorriso do Gordo. Sobre o domínio da imagem original aplicamos duas forças verticais osciladoras, centradas em cada bochecha. As imagens mostram diferentes tempos da animação. Ver `http://w3.impa.br/~dalia/videos.html`



(a) t=0. **(b)** t=1 **(c)** t=8

(d) t=10 **(e)** t=11 **(f)** t=17

Figura 5.4: Aqui outra animação de uma imagem de Oliver Hardy. Movemos a boca e o nariz com uma força osciladora diagonal centrada sobre a boca.Ver `http://w3.impa.br/~dalia/videos.html`



(a) t=0. **(b)** t=4

(c) t=8 **(d)** t=12

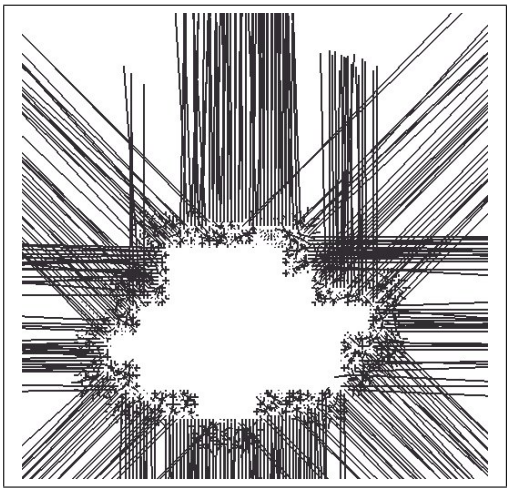
(e) t=16 **(f)** t=20

Figura 5.5: Animação do quadro Starry Night de Van Gogh. Aqui sobre cada estrela é localizada o centro de uma força de vorticidade osciladora, e sobre a Lua também temos uma força de vorticidade de magnitude baixa. Ver `http://w3.impa.br/~dalia/videos.html`

5.6 Resultados da Técnica VV

(a) Imagem Original.

(b) Imagem no tempo t_3

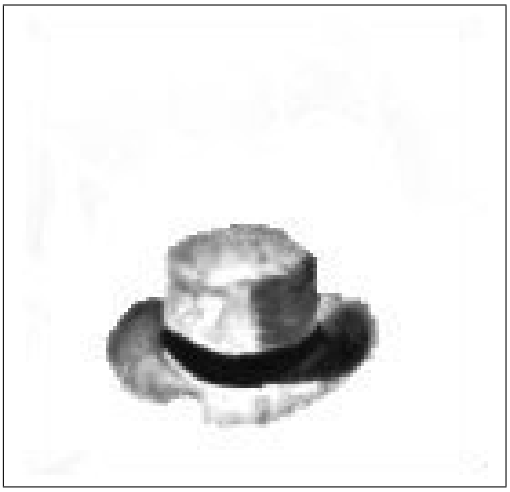


(c) Campo gradiente.

(d) Quantização para a viscosidade

Figura 5.6: Derretendo o chapéu de Van Gogh.

(a) Imagem no tempo t_{10}

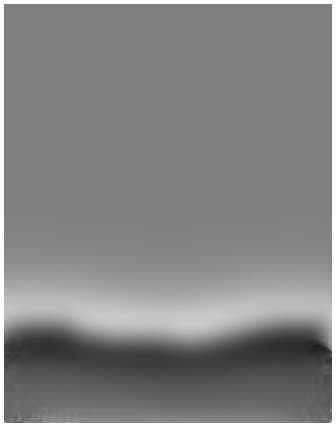


(b) Quantização para a viscosidade

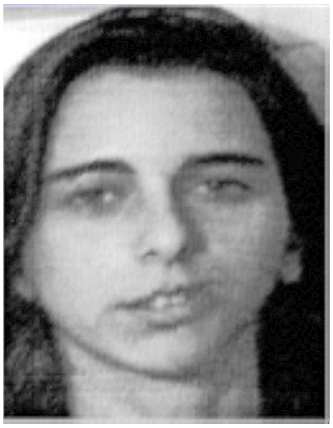
Figura 5.7: Expanção chapéu de Van Gogh.

(a) Imagem Original.

(b) Viscosidade



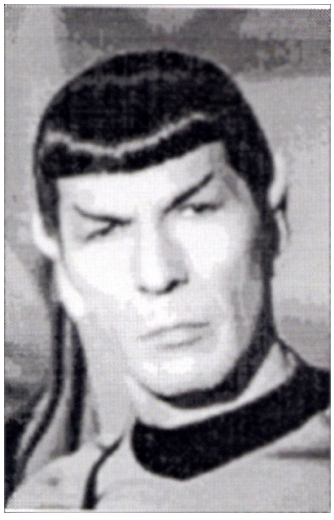
(c) Arestas do campo de forças.



(d) Fluid Warping

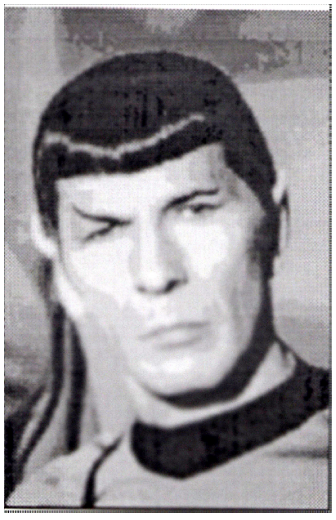
(e) Resultado de Arad [4]

Figura 5.8: Comparação com o exemplo de Arad. [4]



(a) Imagem Original.

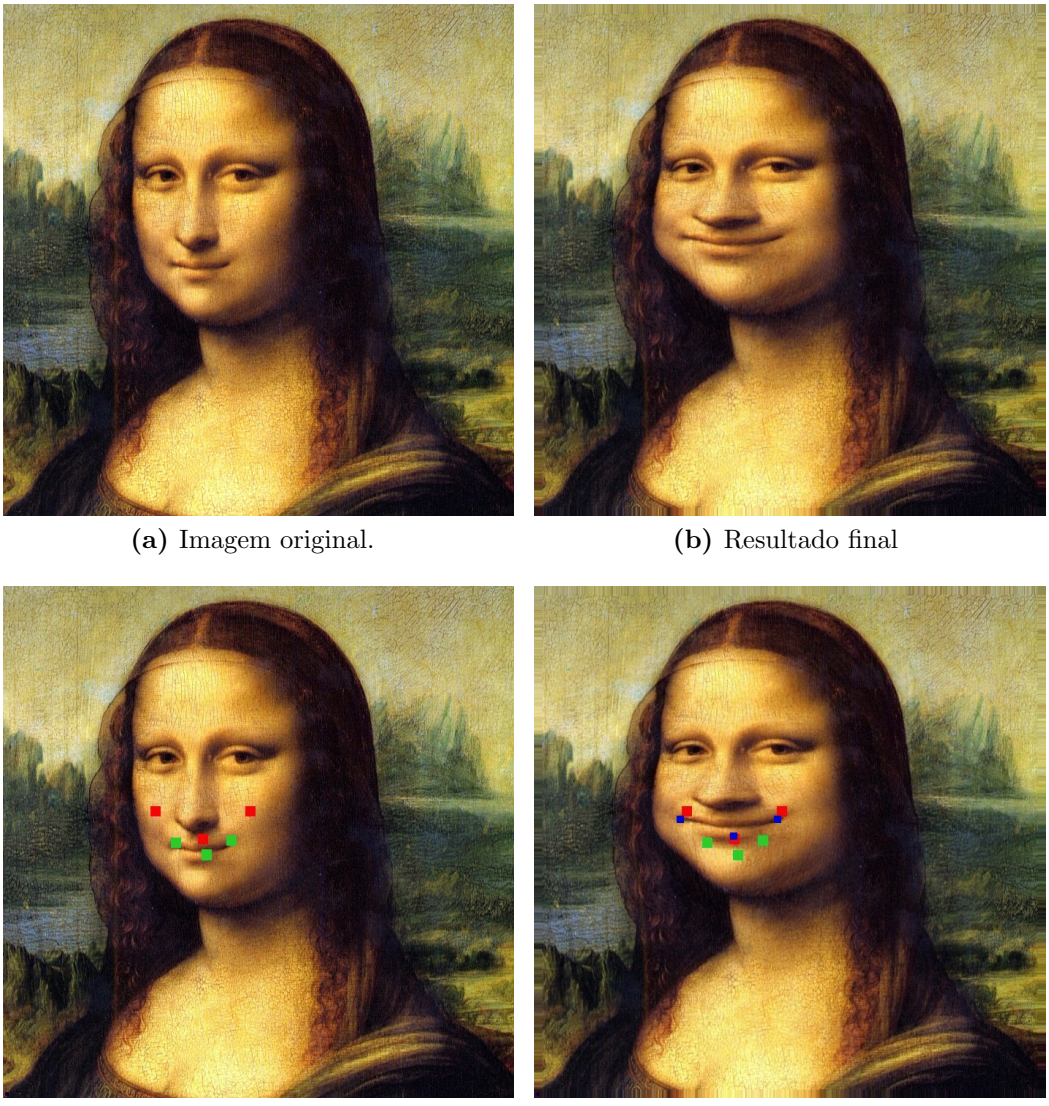
(b) Fluid Warping



(c) Resultado de Arad [4]

Figura 5.9: Comparação com o exemplo de Arad. [4] Note uma pequena diferença nos olhos.

5.7 Resultados da Técnica KP



(a) Imagem original.

(b) Resultado final

(c) Os ponotos verdes são os pontos de saída, e os pontos vermelhos são os pontos de chegada.

(d) Os pontos azueis são o resultado final.

Figura 5.10: Deformação da Mona Lisa por keyframes.

5.7.1 Morphing

Figura 5.11: Sequencia de morphing. Ver `http://w3.impa.br/~dalia/videos.html`

5.8 Resultados Técnicas Híbridas

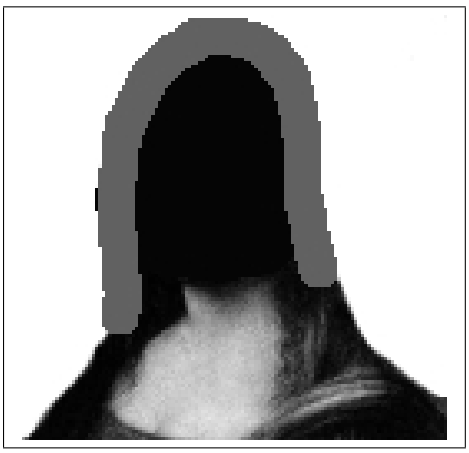
5.8.1 Viscosidade Variável e Forças

(a) t=0.

(b) t=4

(c) t=8

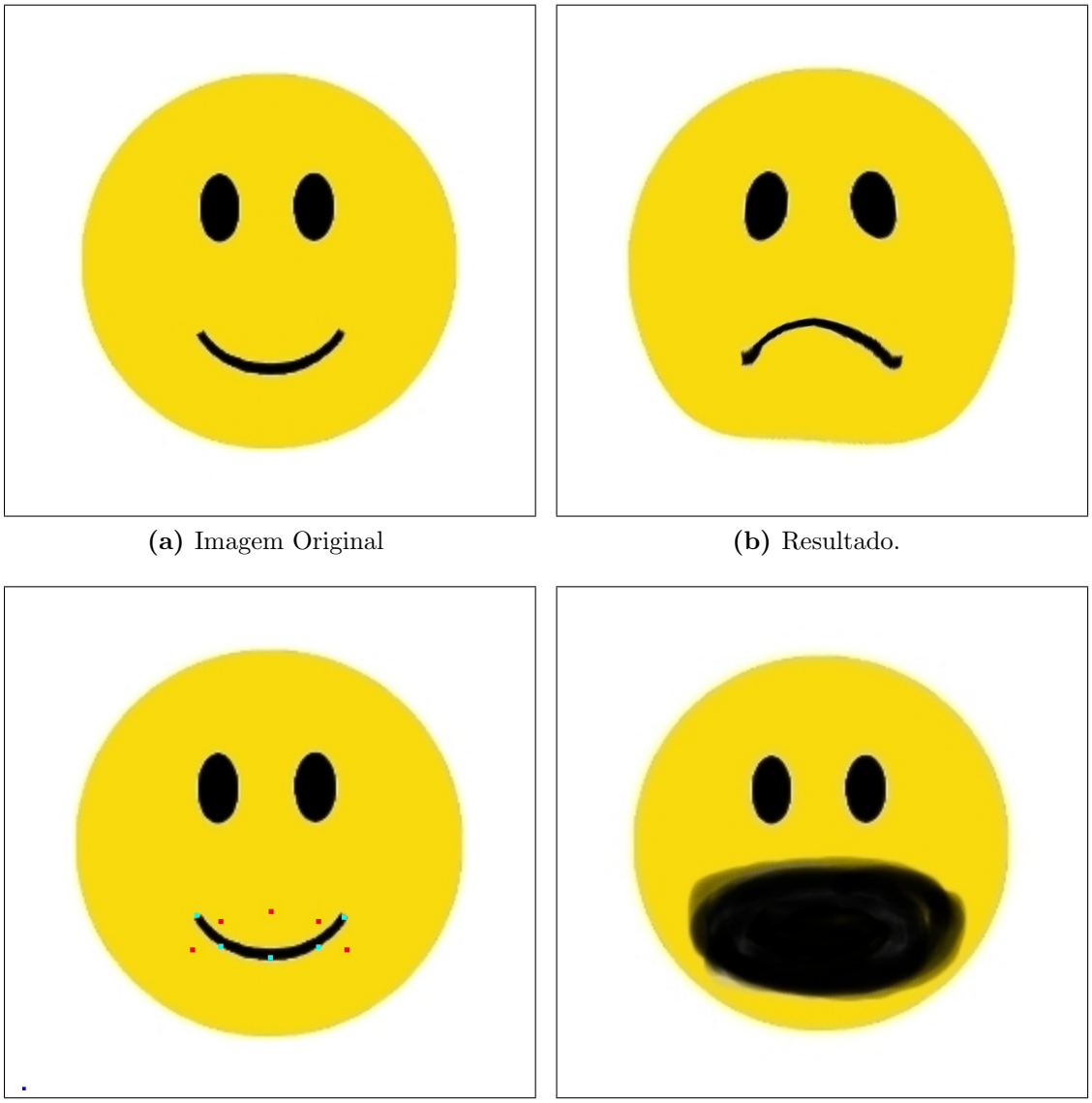
(d) t=12



(e) Viscosidade.

Figura 5.12: Animação da cabeça da Mona Lisa. No fundo de 5.12e a viscosidade é alta, já dentro o valor é menor. Na fronteira um valor intermediário para que a transição entre os dois valores, e portanto o efeito na animação seja menos brusco. Ver `http://w3.impa.br/~dalia/videos.html`.

5.8.2 Viscosidade Variável e Keyframes



(a) Imagem Original

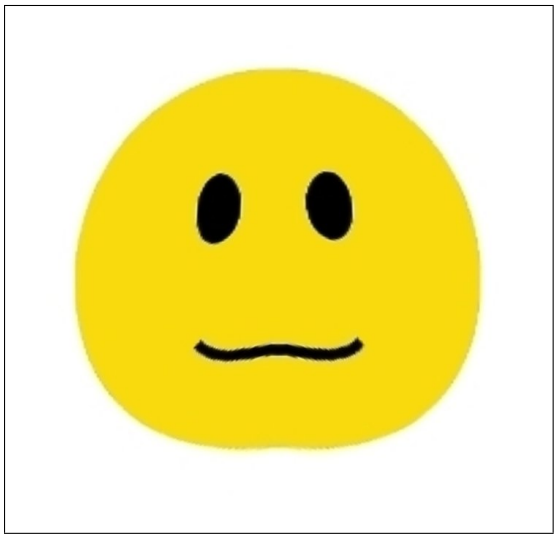
(b) Resultado.

(c) Pontos de saída e keyframe.

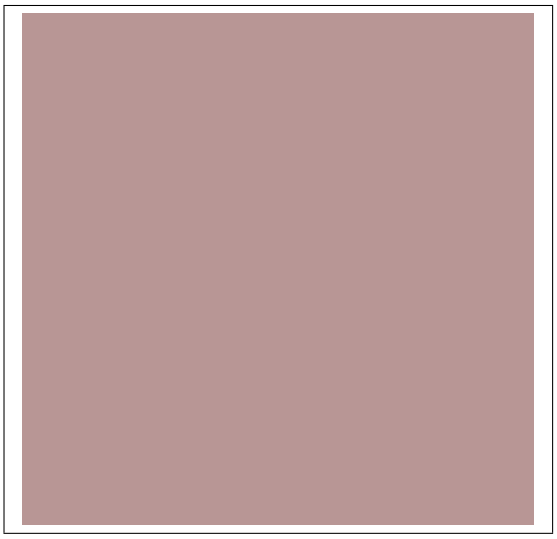
(d) Viscosidade.

Figura 5.13: Técnica que combina viscosidade variável e keyframes.

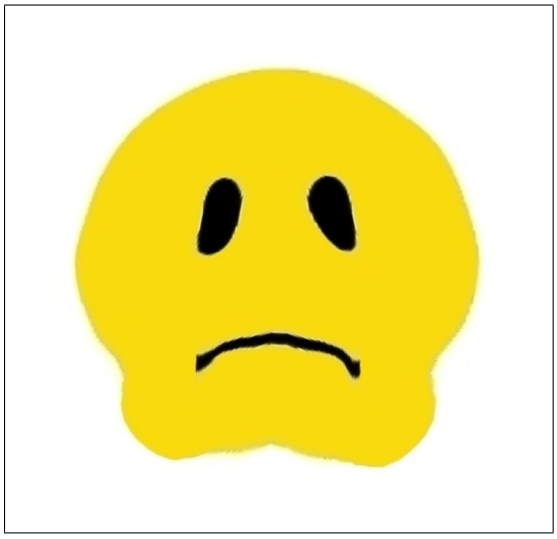
Viscosidades Diferentes na Técnica Viscosidade Variável e Keyframes



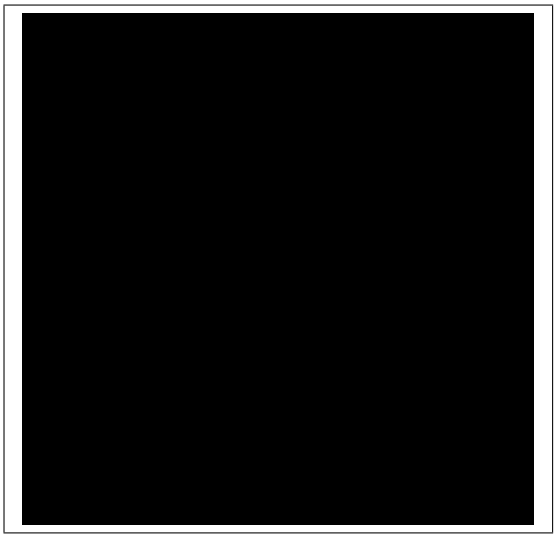
(a) A forma permanece, mas a boca não atinge toda a deformação.



(b) Viscosidade alta em todo o espaço.



(c) A boca se deforma totalmente mas cara perde a forma redonda.



(d) Viscosidade baixa em todo o espaço.

Figura 5.14: Técnica que combina viscosidade variável e keyframes. Outros resultados com diferentes viscosidades.

6 Conclusões

Nesta tese nós mostramos o conceito do domínio de imagem como um fluido 2D homogêneo e incompressível e a técnica *fluid warping* para deformar a imagem. Depois vimos que aplicar o modelo de fluidos sobre os valores da imagem não funciona porque o resultado é perda da imagem. E ao invés disso a técnica fluid warping transporta as coordenadas da parametrização da imagem através do campo vetorial gerado pelas equações de Navier-Stokes, aplicando forças sobre o domínio, sem a perda da imagem. Além disso, para o controle do warping foram desenvolvidas três técnicas de controle apresentadas no capítulo 4.

A primeira técnica VF (viscosidade e forças) mostrou a viscosidade como ferramenta fundamental de controle e dotamos a técnica com uma variedade de forças completas como parâmetros. Os resultados obtidos foram bons, e é possível animar imagens com esta técnica, com efeitos reais, mas para deformações locais, o controle não é preciso.

A segunda técnica VV (viscosidade variável) continua a usar a viscosidade no controle, mas agora é mais completo. As equações de Navier-Stokes são reformuladas, e a definição de viscosidade e forças é feita através de imagens auxiliares. Tornando a definição mas intuitiva para o usuário, e os resultados são mais precisos para deformações sobre regiões. Mas o controle ainda não é muito específico para casos pontuais.

Com a ideia de ter controle mais apurado a terceira técnica KP (keyframe

de partículas) faz uso da técnica de keyframe, para controlar o fluido e emprega o método adjunto para conseguir rapidamente o controle ótimo. O método adjunto já utilizado em outras áreas como é descrito no capítulo 2 e 3, é usado pela primeira vez para o processamento de imagens. Este método é detalhado teoricamente no capítulo 3. Finalmente esta terceira técnica é muito mais precisa que as anteriores, além de ser rápida graças ao método adjunto. O usuário define a deformação desejada através de pontos que localiza sobre a imagem. O resultado desta técnica foi o a sequencia da animação de morphing mostrada no capítulo 5, e que demonstra a precisão do controle.

Como resultado das técnicas anteriores surgiram técnicas hibridas onde a viscosidade variável é usada sobre a técnica VV para animar imagens, e sobre KP para obter um controle ainda mais ótimo.

As possíveis linhas de pesquisa no futuro, continuar criando exemplos para os casos das técnicas hibridas, mas no caso de viscosidade variável e keyframe de partículas. Assim como uma extensão de uma deformação de imagem com efeito 3D, usando para isso extensão das equações de Navier-Stokes para o caso 3D. Assim como achar soluções em casos onde o modelo do fluido limita a deformação.

APÊNDICE A -- Discretização de equações no caso de viscosidade variável

Neste apêndice fornecemos a discretização da fase de difusão da simulação, dados pelas equações abaixo:

$$\begin{cases} u_t = 2\mu_x u_x + \mu_y(u_y + v_x) + \mu\Delta u \\ v_t = 2\mu_y v_y + \mu_x(u_y + v_x) + \mu\Delta v. \end{cases}$$

Agora passamos a discretizar as equações por esquemas de diferencias finitas implícitas. Supomos que Δt é o passo de tempo. Para uma malha $N \times N$ cada quadrado tem de comprimento $h = 1/N$ de lado. Temos então que

$$\begin{aligned} \frac{u_{m,l}^{n+1} - u_{m,l}^{n}}{\Delta t} = \\ 2\mu_x \left[\frac{u_{m+1,l}^{n+1} - u_{m-1,l}^{n+1}}{2h} \right] + \\ \mu_y \left[\frac{u_{m,l+1}^{n+1} - u_{m,l-1}^{n+1}}{2h} + \frac{v_{m+1,l}^{n+1} - v_{m-1,l}^{n+1}}{2h} \right] + \\ \mu \left[\frac{u_{m-1,l}^{n+1} + u_{m+1,l}^{n+1} + u_{m,l-1}^{n+1} + u_{m,l+1}^{n+1} - 4u_{m+1,l}^{n+1} - 4u_{m,l}^{n+1}}{h^2} \right] \end{aligned}$$

E então temos

$$u_{m,l}^{n+1} + \frac{4\Delta t\mu}{h^2}u_{m,l}^{n+1} = 2\Delta t\mu_x\left[\frac{u_{m+1,l}^{n+1} - u_{m-1,l}^{n+1}}{2h}\right] + \Delta t\mu_y\left[\frac{u_{m,l+1}^{n+1} - u_{m,l-1}^{n+1}}{2h} + \frac{v_{m+1,l}^{n+1} - v_{m-1,l}^{n+1}}{2h}\right] + \Delta t\mu\left[\frac{u_{m-1,l}^{n+1} + u_{m+1,l}^{n+1} + u_{m,l-1}^{n+1} + u_{m,l+1}^{n+1} - 4u_{m+1,l}^{n+1}}{h^2}\right] + u_{m,l}^{n}$$

$$u_{m,l}^{n+1} = \frac{1}{1+4\mu\Delta t}\left\{u_{m,l}^{n} + \frac{\Delta t\mu}{h^2}\left(u_{m+1,l}^{n+1} + u_{m-1,l}^{n+1} + u_{m,l+1}^{n+1} + u_{m,l-1}^{n+1}\right)\right.$$

$$\left.\frac{\mu_x\Delta t}{n}\left[u_{m+1,l}^{n+1} - u_{m-1,l}^{n+1}\right] + \frac{\Delta t\mu_y}{2h}\left[u_{m,l+1}^{n+1} - u_{m,l-1}^{n+1} + v_{m+1,l}^{n+1} - v_{m-1,l}^{n+1}\right]\right\}$$

E finalmente da mesma forma obtemos a discretização de v_t

$$v_{m,l}^{n+1} = \frac{1}{1+4\mu\Delta t}\left\{v_{m,l}^{n} + \frac{\Delta t\mu}{h^2}\left(v_{m+1,l}^{n+1} + v_{m-1,l}^{n+1} + v_{m,l+1}^{n+1} + v_{m,l-1}^{n+1}\right)\right.$$

$$\left.\frac{\mu_y\Delta t}{n}\left[v_{m,l+1}^{n+1} - v_{m,l-1}^{n+1}\right] + \frac{\Delta t\mu_x}{2h}\left[u_{m,l+1}^{n+1} - u_{m,l-1}^{n+1} + v_{m+1,l}^{n+1} - v_{m-1,l}^{n+1}\right]\right\}.$$

Referências Bibliográficas

[1] Automatic differentiation. `http://en.wikipedia.org/wiki/Automatic_differentiation`, 2011. [Online; accessed 22-December-2011].

[2] Fluid. `http://en.wikipedia.org/wiki/Fluid`, 2011. [Online; accessed 8-December-2011].

[3] Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. Adaptively sampled particle fluids. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 48.

[4] Arad, N., and Reisfeld, D. Image warping using few anchor points and radial functions. *Computer Graphics Forum 14*, 1 (1995), 35–46.

[5] Bargteil, A. W., Wojtan, C., Hodgins, J. K., and Turk, G. A finite element method for animating large viscoplastic flow. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 16.

[6] Barr, A. H. Global and local deformations of solid primitives. *In Proceedings of SIGGRAPH* (1984).

[7] Beier, T., and Neely, S. Feature-based image metamorphosis. *SIGGRAPH Comput. 2*, 26 (July 1992), 35–42.

[8] Bertalmio, M., Bertozzi, A., and Sapiro, G. Navier-stokes fluid dynamics and image and video inpainting. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)* (December 2001), 9–14.

[9] Birkholz, H., and Jackèl, D. Image warping with feature curves. *In Proceedings of SIGGRAPH* (2003), 199–202.

[10] Bonilla, D., Velho, L., Nachbin, A., and Nonato, L. Fluid warping. In *Proceedings of the IV Iberoamerican Symposium in Computer Graphics* (June 2009), O. Rodríguez, F. Serón, R. Joan-Arinyo, and E. C. J. Madeiras, J. Rodríguez, Eds., Sociedad Venezolana de Computación Gráfica, DJ Editores, C.A.

[11] Carlson, M., Mucha, P. J., and Turk, G. Rigid fluid: animating the interplay between rigid bodies and fluid. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 377–384.

[12] Chen, Q., Wang, M., Pan, N., and Guo, Z.-Y. Optimization principle for variable viscosity fluid flow and its application to heavy oil flow drag reduction. *Energy & Fuels 23*, 9 (2009), 4470–4478.

[13] Chorin, A., and Marsden, J. E. *A mathematical Introduction to Fluid Mechanics*. Springer-Verlag, 1993.

[14] Courant, R., Isaacson, E., and Rees, M. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on Pure and Applied Mathematics 5* (1953), 243–255.

[15] Enright, D., Marschner, S., and Fedkiw, R. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (2002), 736–744.

[16] Fattal, R., and Lischinski, D. Target-driven smoke animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 441–448.

[17] Fedkiw, R., Stam, J., and Jensen, H. W. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 15–22.

[18] Foster, N., and Fedkiw, R. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 23–30.

[19] Foster, N., and Metaxas, D. Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[20] Foster, N., and Metaxas, D. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 181–188.

[21] Giering, R., and Kaminski, T. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software 24*, 4 (1998), 437–474.

[22] Giles, M. B., and Pierce, N. A. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion 65* (2000), 393–415.

[23] Gomes, J., Darsa, L., Costa, B., and Velho, L. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann Publ., 1999.

[24] Gomes, J., and Velho, L. *Image Processing for Computer Graphics*. Springer Verlag, 1997.

[25] Hassanien, I. A. The effect of variable viscosity on flow and heat transfer on a continuous stretching surface. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 77*, 11 (1997), 876–880.

[26] Heckbert, P. S. *Fundamentals of Texture Mapping and Image Warping*. Master's Thesis, University of California, Berkeley, 1989.

[27] Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutre, P., and Gross, M. A unified lagrangian approach to solid-fluid animation. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics 0* (2005), 125–148.

[28] Kwatra, N., Wojtan, C., Carlson, M., Essa, I., Mucha, P. J., and Turk, G. Fluid simulation with articulated bodies. *IEEE Transactions on Visualization and Computer Graphics (TVCG 2010)* (2010).

[29] Lee, S.-Y., Wolberg, G., Kyung-YongChwa, and Shin, S. Y. Image metamorphosis with scattered feature constraints. *IEEE Transactions on Visualization and Computer Graphics 2*, 4 (December 1996), 337–354.

[30] McNamara, A., Treuille, A., Popović, Z., and Stam, J. Fluid control using the adjoint method. *ACM Trans. Graph. 23*, 3 (2004), 449–456.

[31] Melo, S. T., and Neto, F. M. *Mecânica dos Fluidos e a Equações Diferenciais*. 18o Colóquio Brasileiro de Matemática. IMPA, July 1991.

[32] Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[33] Nachbin, A. *Notas do Curso: Dinâmica dos Fluidos*. 2006.

[34] Okazaki, N. libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). `http://www.chokkan.org/software/liblbfgs/`, 2009. [Online; accessed 22-January-2010].

[35] S, S., McPhail.T, and J, W. Image deformation using moving least squares. *ACM Transactions on Graphics (TOG) 25*, 3 (July 2006).

[36] Smith, A. R. Planar 2-pass texture mapping and warping. *In Proceedings of SIGGRAPH* (1987).

[37] Smithe, D. B. A two-pass mesh warping algorithm for object transformation and image interpolation. *Technical memo, Industrial Light and Magic* (1990).

[38] Stam, J. Stable fluids. *SIGGRAPH 99 Conference Proceedings, Annual Conference Series* (August 1999), 121–128.

[39] Stam, J. Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG), Proceedings of SIGGRAPH* (July 2003), 724–731.

[40] Stam, J. Simulation and control of physical phenomena in computer graphics. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 171–173.

[41] Streeter, V., Wylie, E., and Bedford, K. *Mecánica de fluídos*. McGraw-Hill, 2000.

[42] Strikwerda, J. C. *Finite Difference Schemes and Partial Differential Equations*. CRC Press, 1999.

[43] Témam, R. Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (i). *Archive for Rational Mechanics and Analysis 32* (1969), 135–153. 10.1007/BF00247678.

[44] Treuille, A., McNamara, A., Popović, Z., and Stam, J. Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 3 (2003), 716–723.

[45] Wojtan, C., Mucha, P. J., and Turk, G. Keyframe control of complex particle systems using the adjoint method. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 15–23.

[46] Wojtan, C., and Turk, G. Fast viscoelastic behavior with thin features. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–8.

[47] Wolberg, G. Skeleton based image warping. *The Visual Computer 5*, 1-2 (January 1989), 95–108.

[48] Wolberg, G. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1990.

[49] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw. 23*, 4 (1997), 550–560.

[50] Zhu, Y., and Bridson, R. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 965–972.