Pontifícia Universidade Católica
do Rio de Janeiro

Allan Werner Schöttler

Visualização de Fluxo em Reservatórios de Petróleo Usando LIC Volumétrico

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro
Setembro de 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Allan Werner Schöttler

Visualização de Fluxo em Reservatórios de Petróleo Usando LIC Volumétrico

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática – PUC-Rio

Prof. Marcelo Gattass
Departamento de Informática – PUC-Rio

Prof. Luiz Henrique de Figueiredo
IMPA

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do
Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 14 de Setembro de 2018

Allan Werner Schöttler

Graduado em Ciência da Computação pela Pontifícia Universidade Católica do Rio de Janeiro (2016). É aluno de Mestrado da PUC-Rio, membro do instituto TECGRAF, onde desenvolve pesquisas na área de visualização em tempo real.

Ficha Catalográfica

Schöttler, Allan Werner

Visualização de Fluxo em Reservatórios de Petróleo Usando LIC Volumétrico / Allan Werner Schöttler; orientador: Waldemar Celes Filho. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2018.

v., 45 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. convolução de integral de linha;. 3. fluxo 3D;. 4. visualização volumétrica;. 5. reservatório de petróleo;. I. Filho, Waldemar Celes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador, professor Waldemar Celes, por todo conhecimento passado ao longo do meu tempo na PUC-Rio e por todo o apoio e paciência do mundo nesta fase difícil que tem sido a pós-graduação.

À toda minha família, em especial aos meus pais, Moira e Ralph, e minha irmã, Sharon, pelo amor incondicional e por sempre acreditarem em mim.

A todos meus amigos do TECGRAF e da PUC-Rio, por toda a ajuda, educação e apoio que me deram nesses anos todos que fizeram parte da minha formação.

À Márcia e ao Bernardo do GERESIM por me ajudarem a gerar dados para as imagens desta dissertação, e ao Augusto Ícaro do v3o2 por me ajudar com LaTeX.

Aos meus amigos, os Douguistas, por sempre estarem lá por mim, pelas inacabáveis conversas acompanhadas de açaí e por todas as batalhas que já passamos juntos nas montanhas.

À todas as pessoas que não foram mencionadas aqui e que sabem o quão grato sou por conhecê-las.

E por último, gostaria de agradecer ao TECGRAF e à PUC-Rio pelo financiamento sem o qual este trabalho não seria possível.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Schöttler, Allan Werner; Filho, Waldemar Celes. **Visualização de Fluxo em Reservatórios de Petróleo Usando LIC Volumétrico**. Rio de Janeiro, 2018. 45p. Dissertação de Mestrado– Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Na indústria de petróleo, é imprescindível a visualização clara e desambigua de campos vetoriais resultantes de simulações numéricas de reservatórios de petróleo. Nesta dissertação, estudamos o uso da convolução de integral de linha (Line Integral Convolution – LIC) para gerar imagens de campos vetoriais 3D estacionários e aplicar o resultado em um visualizador volumétrico na GPU. Devido a densidade de informação presente na visualização volumétrica, estudamos os uso de texturas esparsas como entrada para o algoritmo de LIC e aplicamos funções de transferência para designar cor e opacidade a volumes de campos escalares, a fim de codificar informações visuais a voxels e aliviar o problema de oclusão. Além disso, tratamos o problema de codificação da direção de fluxo, inerente do LIC, usando uma extensão do algoritmo – Oriented LIC (OLIC). Por último, demonstramos um método de animação do volume a fim de ressaltar a direção do fluxo ainda mais. Comparamos então resultados do algoritmo LIC com o de OLIC.

Palavras-chave

convolução de integral de linha; fluxo 3D; visualização volumétrica; reservatório de petróleo;

Abstract

Schöttler, Allan Werner; Filho, Waldemar Celes (Advisor). **Visualizing Flow in Black-Oil Reservoirs Using Volumetric LIC**. Rio de Janeiro, 2018. 45p. Dissertação de Mestrado– Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In the oil industry, clear and unambiguous visualization of vector fields resulting from numerical simulations of black-oil reservoirs is essential. In this dissertation, we study the use of line integral convolution techniques (LIC) for imaging 3D steady vector fields and apply the results to a GPU-based volume rendering algorithm. Due to the density of information present in volume renderings of LIC images, we study the use of sparse textures as input to the LIC algorithm and apply transfer functions to assign color and opacity to scalar fields in order to encode visual information to voxels and alleviate the occlusion problem. Additionally, we address the problem of encoding flow orientation, inherent to LIC, using an extension of the algorithm – Oriented LIC (OLIC). Finally, we present a method for volume animation in order to enhance the flow orientation. We then compare results obtained with LIC and with OLIC.

Keywords

line integral convolution; 3D flow; volume rendering; black-oil reservoir;

Sumário

1 Introdução **12**
1.1 Motivação 12
1.2 Contribuições desta Dissertação 13
1.3 Estrutura da Dissertação 13

2 Trabalhos Relacionados **14**

3 Fundamentos **16**
3.1 Dimensionalidade 16
3.2 Domínio 16
3.3 Grades 17
3.4 Integração 18

4 Metodologia **20**
4.1 LIC 20
4.2 LIC Volumétrico 26
4.3 LIC Orientado 27

5 Implementação **31**

6 Resultados **34**

7 Conclusões e Trabalhos Futuros **43**

Referências bibliográficas **44**

Lista de figuras

Figura 2.1 Glifos de setas. 14
Figura 2.2 Geometria de linhas de corrente representando fluxo em um reservatório de petróleo. 15

Figura 4.1 Diferença entre integração em DDA e em LIC. *Fonte*: (Cabral *et al*, 1993) 21
Figura 4.2 Singularidades ressaltadas pela normalização fixa. *Fonte*: (Cabral *et al*, 1993) 24
Figura 4.3 Filtro periódico. *Acima*: instantes de tempo da função de Hann; *centro*: janela de Hann; *abaixo*: instantes de tempo função de Hann aplicada sobre a janela. *Fonte*: (Cabral *et al*, 1993) 25
Figura 4.4 Volume de ruído branco de entrada (*esquerda*) e volume gerado por LIC (*direita*). *Fonte*: (Interrante *et al*, 1997) 26
Figura 4.5 Imagens LIC (*abaixo*) geradas por diferentes texturas (*acima*). *Da esquerda para a direita*: ruído branco, esparsa regular e esparsa regular com perturbações. *Fonte*: (Wegenkittl *et al*, 1997) 27
Figura 4.6 Efeito de "auréolas". A imagem central mostra linhas de corrente sem o efeito, enquanto as imagens laterais revelam diferentes suposições das mesmas. *Fonte*: (Interrante *et al*, 1997) 28
Figura 4.7 OLIC em comparação a LIC. É possível ver a orientação no sentido anti-horário (*acima*) e horário (*abaixo*). *Fonte*: (Wegenkittl *et al*, 1997) 29
Figura 4.8 Filtro de rampa aplicado ao longo da linha de corrente integrada. O resultado é um série de círculos rasterizados (ou esferas) com luminosidades crescentes ao longo do campo vetorial. *Fonte*: (Wegenkittl *et al*, 1997) 30
Figura 4.9 Magnitude codificada no tamanho das linhas de corrente. *Fonte*: (Wegenkittl *et al*, 1997) 30

Figura 6.1 Solução da equação: $\overline{F}(x, y, z) = -y\hat{i} + x\hat{j} + z\hat{k}$ 34
Figura 6.2 Solução da equação: $\overline{F}(x, y, z) = y^2\hat{i} + x^2\hat{j} + z\hat{k}$. Função degrau em combinação com tabela de cor de magnitude. 35
Figura 6.3 Reservatório sem funções de transferência. 36
Figura 6.4 Região interna de reservatório vista com plano de corte. Singularidades ressaltadas em preto, posteriormente filtradas pela função degrau. 37
Figura 6.5 Visualização de reservatório com algoritmo LIC. 38
Figura 6.6 Visualização do modelo da Figura 6.5 com plano de corte. 39
Figura 6.7 Comparação de densidades de textura esparsa com a função de transferência 5-3 em modelo de reservatório. 40
Figura 6.8 Tamanhos variantes de linhas de corrente sem tabela de cor. Linhas brancas verticas representam poços. 41
Figura 6.9 Figura 6.8 com função de transferência de magnitudes. 42

Lista de algoritmos

Algoritmo 1	Algoritmo de compactação de quadros	32
Algoritmo 2	Algoritmo de descompactação de quadros	32

Lista de abreviaturas

DDA – *Digital differential analyzer*
LIC – *Line Integral Convolution*
OLIC – *Oriented Line Integral Convolution*
OSG – *Open Scene Graph*
GPU – *Graphics Processing Unit*

What we observe is not nature itself, but nature exposed to our method of questioning.

Werner Heisenberg, .

1
Introdução

O processo de extração de petróleo de reservatórios, que essencialmente são estruturas de rocha porosa, envolve a perfuração de poços que permitem a pressão subterrânea natural ou induzida (através do bombeamento de água por poços de injeção) trazer o petróleo à superfície. O planejamento para exploração de um reservatório consiste em avaliar conjuntos de poços de produção e de injeção, em uma simulação numérica, que maximizam a extração do petróleo. O modelo de reservatórios nestas simulações consiste em uma série de células discretas tridimensionais cada uma contendo um grande conjunto de dados, dos quais fluxos de petróleo, água e gás fazem parte. Estes dados devem então ser analisados por meio de alguma técnica de visualização.

1.1
Motivação

Uma visualização clara e desambigua de campos vetoriais 3D não é trivial. Linhas de corrente são tradicionalmente usadas na indústria de petróleo para revelar o comportamento do fluxo em um reservatório, mas as imagens resultantes mostram uma representação esparsa do fluxo e são geralmente difíceis de interpretar sem ambigüidade no espaço 3D. Isto é particularmente verdadeiro para modelos de reservatório em que o fluxo torna-se denso e complexo em regiões próximas de poços. Muitas soluções acabam dependendo de artifícios como planos de corte ou funções de transferência – no caso de visualização volumétrica – para melhor interpretação.

Nesta dissertação, revisitamos o uso da convolução de integral de linha (LIC), uma técnica baseada no uso de texturas para gerar imagens 3D de campos vetoriais. Além da forma tradicional de LIC que usa texturas de ruído branco, estudamos o uso de texturas esparsas para aliviar o problema da oclusão, que é uma das maiores desvantagens da visualização 3D. Tratamos também o problema da falta de informação da orientação do fluxo, presente no LIC, através do uso de filtros periódicos que simulam o efeito de movimento e de um esquema de degradê na direção do fluxo.

Uma vez gerada a textura de LIC, uma série de propriedades do campo

vetorial são extraídas para produzir funções de transferência que serão usadas no algoritmo de visualização volumétrica.

1.2 Contribuições desta Dissertação

Como contribuições desta dissertação, destacam-se:

1. Um *framework* para gerar imagens LIC 3D;

2. Visualização volumétrica densa de fluxo em reservatórios de petróleo.

1.3 Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma: no Capítulo 2 apresentamos alguns métodos de visualização de fluxo; no Capítulo 3 explicamos alguns fundamentos matemáticos necessários nestes métodos; no Capítulo 4 apresentamos a metodologia utilizada neste trabalho; no Capítulo 5 mostramos alguns detalhes de implementação; no Capítulo 6 apresentamos os resultados obtidos e por fim, no Capítulo 7, apresentamos algumas conclusões e propomos trabalhos futuros.

2
Trabalhos Relacionados

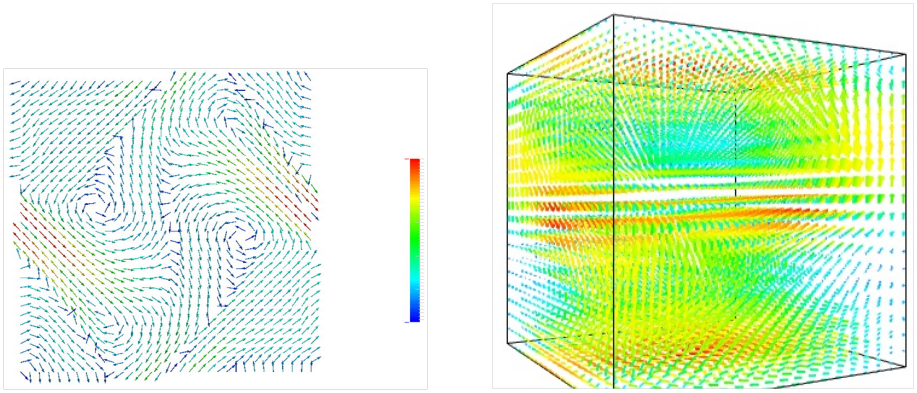
Existem diversas técnicas para visualização de campos vetoriais que dependem do tipo do dado e da aplicação pretendida. Este capítulo faz um resumo breve de algumas dessas técnicas.

(Laramee *et al*, 2004) propuseram uma classificação para problemas de visualização de campos vetoriais baseada nas necessidades de usuários.

Visualização direta de fluxo: são técnicas objetivas onde o resultado consiste em uma visão imediata do fluxo. Uma abordagem comum é desenhar glifos de flechas na direção do fluxo e codificar a magnitude das velocidades por cores (Figura 2.1). Este método é eficaz em cenários 2D, mas pode ficar muito convoluto e difícil de interpretar no caso 3D.

Visualização geométrica de fluxo: é um conjunto de técnicas onde geralmente o dado de fluxo é integrado e objetos geométricos são usados na visualização resultante. Linhas de corrente (Figura 2.2) são exemplos desta categoria; no entanto, esta técnica tem como desafio fazer uma boa seleção de sementes a fim de evitar perda de informações importantes do fluxo. A popularidade deste método na indústria de petróleo deve-se ao fato de regiões entorno de poços serem uma ótima heurística para o algoritmo de seleção de sementes.

Visualização baseada em características: métodos que procuram extrair, numa etapa anterior à de visualização, características específicas de regiões de inte-



2.1(a): 2D 2.1(b): 3D

Figura 2.1: Glifos de setas.

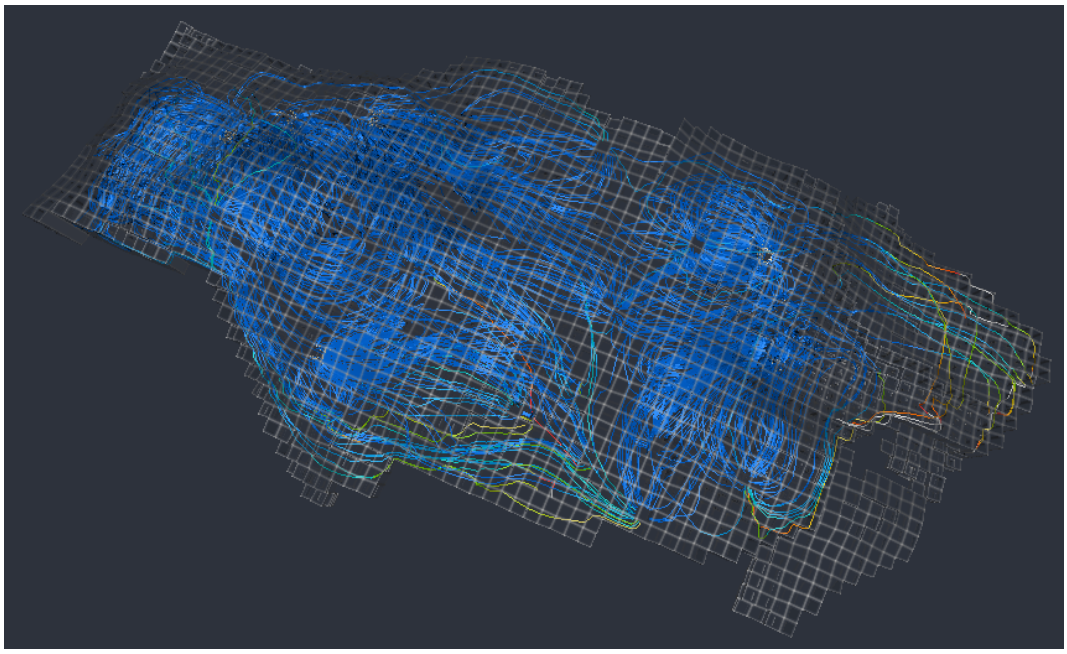


Figura 2.2: Geometria de linhas de corrente representando fluxo em um reservatório de petróleo.

resse, como fenômenos importantes (e.g. vorticidade e temperatura) ou informação topológica do fluxo. A visualização então é baseada nestas características de fluxo (ao invés do conjunto de dados como um todo), permitindo uma visualização compacta e eficiente até em conjuntos de dados muito grandes.

Visualização baseada em textura: métodos que computam uma textura que representa um campo de forma densa. A noção de direção do fluxo é dada através de pixels correlacionados ao longo do campo vetorial. Este efeito é geralmente alcançado através da filtragem de valores de textura de acordo com o comportamento local do fluxo. Exemplos dessa categoria de métodos são (1) técnicas de *spot noise*, introduzidas por (Wijk *et al*, 1991), onde a primitiva é uma "mancha"(e.g. um elipsóide) que é deformada para refletir características do campo vetorial, (2) técnicas de advecção de textura, onde a primitiva é um texel, grupo de texels ou coordenadas de textura, como (Max *et al*, 1992), e (3) técnicas de LIC, indroduzidas por (Cabral *et al*, 1993), onde as primitivas são texturas, geralmente de ruído branco, convoluídas por um filtro. Quanto à visualização densa de fluxo em reservatórios de petróleo, poucos trabalhos existem atualmente. (Toledo *et al*, 2011) aplicam texturas LIC 2D em superfícies que representam camadas de um reservatório.

3
Fundamentos

Este capítulo apresenta alguns fundamentos matemáticos necessários em diversas técnicas de visualização de fluxo.

3.1
Dimensionalidade

Métodos de visualização de fluxo diferem quanto à dimensionalidade do dado de fluxo. Técnicas geralmente úteis para dados 2D (e.g. flechas) muitas vezes carecem de vantagens similares em 3D. Além da dimensão espacial, a dimensão temporal é de grande importância. Velocidade incorpora a noção de tempo – fluxos são muitas vezes interpretados como dados diferenciais com relação ao tempo, i.e. quando integra-se o dados, obtêm-se trajetórias como linhas de corrente. Chamamos este tipo de campo vetorial de *estacionário*. Por outro lado, o dado de fluxo pode variar com o tempo; chamamos de *variante no tempo*.

A visualização deve cuidadosamente distinguir entre um e outro; em especial, se for necessário o uso de animações. É comum ver objetos geométricos como linhas de corrente sendo animadas para ressaltar a orientação do fluxo, e.g. através do movimento de cor em uma tabela de cores. Animações em métodos de textura também são muito usadas com o mesmo objetivo em mente.

Em muitas situações, dimensões adicionais, i.e. atributos como temperatura, pressão ou vorticidade, são fornecidas no dado. A visualização também podem levar em conta estes valores, e.g. usando cores ou extração de isosuperfícies.

3.2
Domínio

Simulações de fluxo são geralmente soluções discretas de sistemas de EDPs, como equações de Euler e Navier-Stokes. Uma característica inerente de dados de fluxo é que contêm informação derivativa (velocidade) em relação ao tempo que, por sua vez, é disposta sobre um domínio espacial n-dimensional $\Omega \subseteq \mathbb{R}^n$, e.g. $n = 3$ para representar fluxo 3D. Derivadas temporais v de posições n-dimensionais p no domínio Ω são vetores também nD:

$$v = \frac{dp}{dt}, p \in \Omega \subseteq \mathbb{R}^n, t \in \mathbb{R} \tag{3-1}$$

Uma formulação generalizada do fluxo v (possivelmente variante de tempo) é

$$v(p,t) : \Omega \times \Pi \rightarrow \mathbb{R}^n,$$

onde $p \in \Omega \subseteq \mathbb{R}^n$ representa a referência espacial do dado e $t \in \Pi \subseteq \mathbb{R}$ representa o sistema de tempo. Para fluxos estacionários, é dado o caso mais simples $v(p) : \Omega \rightarrow \mathbb{R}^n$, onde v não depende de t.

Em simulações *n*D, como de aplicações automotivas e petrolíferas, estes vetores v não são usualmente dados na forma analítica, mas sim requerem reconstrução da saída da simulação. Os métodos numéricos usados para simulação de fluxo (e.g. método de diferenças finitas) geram valores em grades com vetores v_i que representam de forma discreta a solução da simulação.

3.3 Grades

Na simulação de fluxo, os vetores v_i amostrados normalmente são dispostos por um domínio com uma certa grade. Tipos de grade variam desde grades cartesianas e retilíneas até grades curvilíneas e não-estruturadas complexas. Naturalmente, operações feitas em grades devem diferir cuidadosamente entre cada tipo, e.g. buscas de vizinhos podem ser triviais em grades retilíneas, enquanto grades híbridas tendem a ser complexas. Algumas operações fundamentais são interessantes citar.

Começando com *localização de pontos*, i.e. o problema de encontrar a célula de uma grade onde um dado ponto nD, expresso em uma matriz P de dimensão $1 \times n$, se encontra. Para grades regulares, este problema é trivial:

$$I = \lfloor P \oslash S \rfloor$$

onde S é uma matriz $1 \times n$ que representa um passo incremental da grade e I é a matriz onde cada componente de P e S são divididos respectivamente para representar os índices da célula à qual o ponto pertence. Para grades mais complexas, estruturas de dados especiais podem ser usadas para subdividir o domínio espacial e acelerar a busca, e.g. estruturas de árvores como *octrees*. Como muitas vezes a localização de pontos é feita iterativamente (como no caso da integração), muitos algoritmos eficientemente exploram coerência espacial durante a busca (Hege *et al*, 1998). Certos algoritmos estimam uma célula e a partir daí refinam a busca até a célula-alvo ser alcançada.

Além de localização de pontos na grade, a *reconstrução de fluxo*, ou interpolação, dentro de uma célula é desejável. Uma vez localizada a célula onde se encontra o ponto de consulta, a reconstrução leva em conta apenas os vetores amostrados nos vértices da célula. A abordagem mais comum é a reconstrução de primeira-ordem

através de interpolações lineares dentro da célula, e.g. o fluxo de uma célula 3D hexaédrica pode ser reconstruído por interpolações trilineares.

A localização de pontos e reconstrução de fluxo já permitem o uso de muitas técnicas de visualização. Mesmo assim, pode ser necessário o uso de *cálculo vetorial* para produzir visualizações mais sofisticadas. Normalmente, o primeiro passo é computar o gradiente $\nabla v|_p$ para pontos arbitrários do domínio, que por sua vez traz informações de propriedades locais do fluxo em um ponto, como convergência, divergência, rotação e cisalhamento. Dependendo da aplicação, o valor de vorticidade ($\omega = \nabla \times v$) pode ser interessante também. (Park *et al*, 2004) extraem diversas propriedades do campo vetorial para gerar funções de transferência que codificam estas informações em uma visualização volumétrica.

3.4 Integração

Relembrando que dados de fluxo muitas vezes são derivadas em relação ao tempo, a idéia de integrar o dado sobre o tempo é natural e nos dá uma noção intuitiva da evolução induzida pelo fluxo. Um exemplo de visualização é a advecção de partículas. A trajetória de uma partícula $p(t)$ – aqui representando fluxo variante – pode ser definida por

$$p(t) = p_0 + \int_0^t v(p(\tau), \tau) d\tau \tag{3-2}$$

onde p_0 representa a partícula no tempo $t = 0$. Note que as equações 3-1 e 3-2 são complementares.

É importante notar que equações integrais, como na equação 3-2, geralmente não podem ser resolvidas analiticamente e portanto requerem o uso de algum método de integração numérica. A técnica mais simples é o método de Euler de primeira-ordem para computar uma aproximação $p_E(t)$, onde uma única iteração da integração da curva é dada por

$$p_E(t + \Delta t) = p(t) + \Delta t \cdot v(p(t), t).$$

Δt usualmente é um incremento muito pequeno de tempo e $p(t)$ é a posição inicial de cada passo. Uma aproximação melhor, porém mais custosa, é o método Runge-Kutta de segunda-ordem, que utiliza a aproximação de Euler p_E para computar uma melhor aproximação $p_{RK2}(t)$:

$$p_{RK2}(t + \Delta t) = p(t) + \Delta t \cdot \frac{v(p(t), t) + v(p_E(t + \Delta t), t)}{2}.$$

Métodos de maior ordem, como o integrador Runge-Kutta de quarta-ordem, utilizam mais passos para alcançar aproximações melhores. Tamanhos de passo adaptativos também podem ser usados para computar passos menores em regiões de alta curvatura.

4
Metodologia

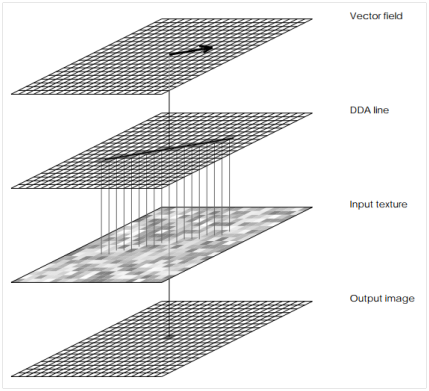
Nesta dissertação, estudamos o uso de técnicas LIC para visualização de campos vetoriais 3D originados de simulações numéricas de reservatórios de petróleo. Estes campos vetoriais são geralmente instáveis, mas muitas vezes é desejável que sua visualização seja a de um campo estacionário, i.e. um único instante de tempo do campo original para uma análise mais precisa.

No entanto, LIC volumétrico traz alguns problemas para a visualização que serão abordados neste capítulo. Apresentamos algumas soluções que permitem uma melhor visualização do dado.

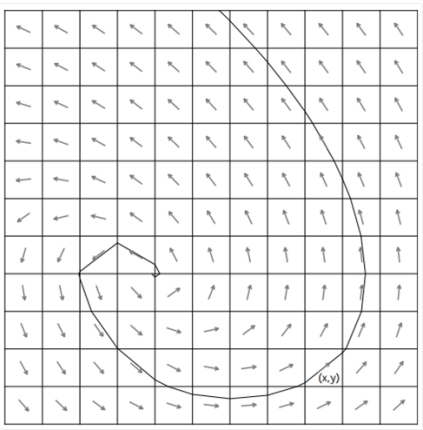
4.1
LIC

(Cabral *et al*, 1993) introduziram a convolução de integral de linha (LIC) para imagem de campos vetoriais bidimensionais. Desde então, uma grande coleção de pesquisas surgiu entorno deste método. O algoritmo de LIC é baseado em convolução DDA, uma técnica tradicional de rasterização de linhas. Em DDA, cada vetor do campo é usado para gerar um kernel tangente ao fluxo localmente e que se estende nas direções positiva e negativa do vetor por uma distância L fixa. Uma textura é então mapeada ao campo vetorial. Os pixels da textura de entrada que se encontram sob o kernel são somados e, então, normalizados pelo comprimento total $2L$ do kernel para gerar um único escalar de luminosidade, que é então posto em um pixel de uma imagem de saída. Este processo é elucidado na Figura 4.1(a). Embora esta técnica faça um trabalho eficaz de filtrar a textura subjacente, uma linha geralmente não é suficiente para aproximar o fluxo expresso localmente em um campo vetorial. Regiões de alta turbulência e vorticidade, que têm alta curvatura, tendem a gerar resultados confusos neste método.

O algoritmo de LIC procura tratar este comportamento local do fluxo gerando um kernel que é a linha de corrente local, ao invés de apenas a tangente. O algoritmo original recebe um campo vetorial em uma grade 2D cartesiana e uma textura de entrada, geralmente uma textura de ruído branco. Similar a convolução DDA, para cada pixel da textura de entrada, uma linha de corrente é computada nas direções positiva e negativa do campo vetorial, começando do centro do pixel. Em seguida,



4.1(a): Integração e convolução em DDA



4.1(b): Integração de Euler em LIC

Figura 4.1: Diferença entre integração em DDA e em LIC. *Fonte*: (Cabral *et al*, 1993)

pixels que se encontram sob as linhas de corrente são convoluídas usando um filtro para gerar uma imagem densa do campo. Esta convolução de pixels por uma linha de corrente σ – parametrizada por s – pode ser expressa matematicamente como o cálculo de intensidade I de um pixel em $x_0 = \sigma(s_0)$:

$$I(x_0) = k \otimes F = \int_{s_0-L}^{s_0+L} k(s - s_0)F(\sigma(s))ds \tag{4-1}$$

onde F é a textura de entrada, k é o filtro e s é um comprimento de arco usado como parâmetro para a linha de corrente.

4.1.1 Método Numérico

Como mencionado na Seção 3.4, a forma analítica na Equação 4-1 não é suficiente e, portanto, é necessário o uso de um método de integração numérica. Com isto em mente, a técnica original de LIC usa o método de Euler da seguinte forma:

$$p_0 = (x + 0.5, y + 0.5)$$

$$p_i = p_{i-1} + (\Delta s_{i-1} + \epsilon) \cdot \frac{V(\lfloor p_{i-1} \rfloor)}{\|V(\lfloor p_{i-1} \rfloor)\|} \tag{4-2}$$

$$s_e = \begin{cases} \infty, & \text{se } V \| e. \\ 0, & \text{se } \frac{\lfloor p_c \rfloor - p_c}{v_c} < 0. \\ \frac{\lfloor p_c \rfloor - p_c}{v_c}, & \text{caso contrário.} \end{cases} \text{ para } (e,c) \in \begin{cases} (topo, y) \\ (base, y) \\ (esquerda, x) \\ (direita, x) \end{cases} \tag{4-3}$$

$$\Delta s_i = min(s_{topo}, s_{base}, s_{esquerda}, s_{direita})$$

Δs_i é a distância paramétrica positiva ao longo da tangente local em p_i até a aresta da célula mais próxima. ϵ é um pequeno incremento que garante a entrada do ponto p_i na célula seguinte. Como no algoritmo de DDA, é importante manter a simetria, portanto a linha de corrente na direção negativa também é calculada. Variáveis com aspas simples representam as variáveis correspondentes na direção negativa. Note que $\Delta s'$ é sempre positivo.

$$p'_0 = p_0$$

$$p'_i = p'_{i-1} - (\Delta s'_{i-1} + \epsilon) \cdot \frac{V(\lfloor p'_{i-1} \rfloor)}{\|V(\lfloor p'_{i-1} \rfloor)\|} \tag{4-4}$$

O cálculo de Δs_i é sensível a arredondamentos. Δs_i deve produzir coordenadas contidas na $i+1$-ésima célula, tal que cada segmento de reta comece na i-ésima célula e termine na $i+1$-ésima. Um fator de arredondamento ϵ é somado a cada Δs_i para garantir a entrada na célula seguinte. A Figura 4.1(b) ilustra este processo de advecção.

Nesta representação, cada célula é unitária. Todas as quantias espaciais (e.g. Δs_i) são relativas a esta medida. Todavia, as células não precisam necessariamente ser quadradas ou até mesmo retangulares para esta aproximação funcionar.

O resultado final da integração da linha de corrente é uma coleção de segmentos de linha contínuos parametrizados por s. Estes segmentos são então integrados sobre a textura de entrada, que pode ser tratada como uma função F escalar contínua de (x, y), e convoluídos como no algoritmo de DDA, gerando um resultado que é uma melhor aproximação do fluxo. Para cada segmento de reta i, uma integral exata h_i do kernel de convolução $k(w)$ é computada e usada como peso de um pixel da textura de entrada:

$$h_i = \int_{s_i}^{s_i + \Delta s_i} k(w) dw \tag{4-5}$$

onde

$$s_0 = 0$$

$$s_i = s_{i-1} + \Delta s_{i-1}$$

O pixel de saída é dado pela equação

$$F'(x,y) = \frac{\sum_{i=0}^{L} F(\lfloor p_i \rfloor) h_i + \sum_{i=0}^{L'} F(\lfloor p'_i \rfloor) h'_i}{\sum_{i=0}^{L} h_i + \sum_{i=0}^{L'} h'_i} \tag{4-6}$$

onde $F(\lfloor p \rfloor)$ é o pixel de entrada correspondente ao vetor na posição $(\lfloor p_x \rfloor, \lfloor p_y \rfloor)$.

O numerador da equação 4-6 representa os pixels de entrada da textura F ponderados pela integral de cada segmento de reta. O denominador é o somatório dos pesos usado para normalizar o pixel de saída. O comprimento $2L$ de cada linha de corrente é dado em pixels e portanto diretamente afeta a performance do algoritmo. Se L for muito pequeno em relação ao tamanho da textura F, pode ocorrer filtragem insuficiente. Por outro lado, se L for muito grande, a tendência é que a textura de saída tenha valores muito parecidos para todo (x, y).

Singularidades podem ocorrer no campo vetorial em linhas de corrente adjacentes que apontam para uma mesma aresta de uma célula compartilhada. Isso resulta em valores de Δs_i iguais a zero. Esta situação pode facilmente ser detectada e o algoritmo terminado. Similarmente, o algoritmo pára caso o campo vetorial for para zero em qualquer ponto. Estes dois casos podem gerar linhas de corrente truncadas. No caso de uma linha de corrente que começa em um pixel onde o vetor é nulo, um valor arbitrário é retornado dependendo do resultado desejado para vetores nulos. O uso de uma normalização fixa no denominador da Equação 4-6 – i.e. uma constante arbitrária, mas que garante a normalização do pixel de saída – ressalta as sigularidades do campo vetorial na imagem final (ver Figura 4.2). Isto ocorre pois linhas de corrente truncadas tendem a ser menores quando próximas de singularidades, fazendo com que o numerador na Equação 4-6 e consequentemente a luminosidade de saída tendam a zero.

Finalmente, a imagem de saída é normalizada para manter as propriedades de contraste, já que o algoritmo de LIC reduz o contraste da imagem de entrada em função de L, i.e. se L tende ao infinito, a imagem toda é convoluída e normalizada tendendo à média da imagem de entrada. Uma repassada do algoritmo de LIC sobre a própria saída geralmente ajuda a ressaltar as linhas de corrente. Isto ocorre pois as linhas de corrente na segunda passada do algoritmo já vão estar mapeadas sobre pixels de cores similares.

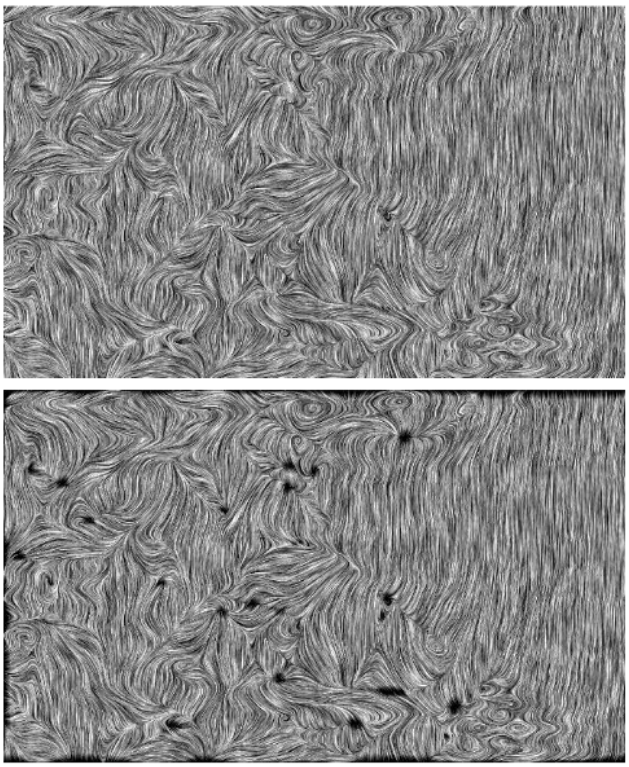


Figura 4.2: Singularidades ressaltadas pela normalização fixa. *Fonte*: (Cabral *et al*, 1993)

4.1.2 Animação

O algoritmo de LIC visualiza as tangentes locais do campo vetorial, mas não suas direções. (Freeman *et al*, 1991) propõem uma técnica que simula movimento em imagens estáticas pelo uso de kernels de convolução especiais. Esta técnica pode ser estendida para representar a direção do campo vetorial através da animação de imagens LIC sucessivas geradas pelo deslocamento de fase em filtros periódicos.

No entanto, o sucesso desta técnica depende do formato do filtro. Para imagens estáticas, um filtro constante geralmente atende. Para o efeito de animação, uma função periódica (e.g. função de Hann) é necessária. Esta função é chamada de *função de onda*. Como o algoritmo de LIC é definido por operações locais, é necessário também usar uma *função de janela*. Uma função *caixa* como janela normalmente vai gerar cortes repentinos no filtro que por sua vez resultam em artefatos na animação final. Idealmente, uma função Gaussiana resolve este problema. No entanto, devido a natureza imprópria da integral Gaussiana, seu uso é inviável. (Cabral *et al*, 1993) conseguem resultados efetivos usando a própria função de Hann – que se aproxima da Gaussiana – como janela (ver Figura 4.3). A forma geral desta função pode ser escrita como:

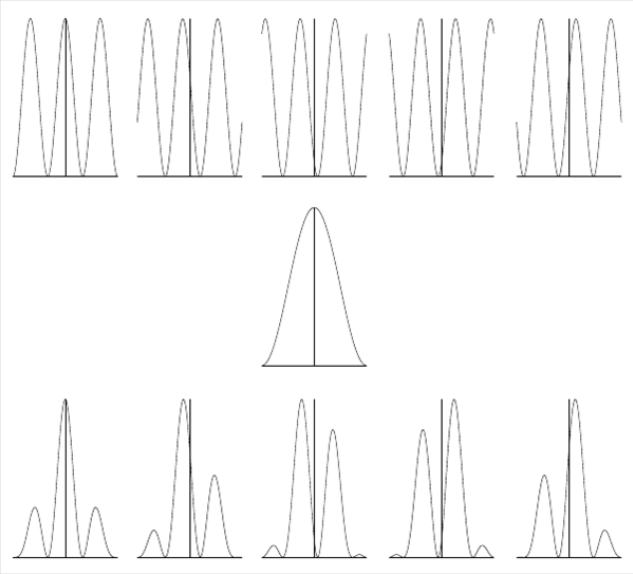


Figura 4.3: Filtro periódico. *Acima*: instantes de tempo da função de Hann; *centro*: janela de Hann; *abaixo*: instantes de tempo função de Hann aplicada sobre a janela. *Fonte*: (Cabral *et al*, 1993)

$$\begin{aligned} k(w) &= \frac{1+\cos(cw)}{2} \times \frac{1+\cos(dw+\beta)}{2} \\ &= \frac{1}{4}(1+\cos(cw)+\cos(dw+\beta)+\cos(cw)\cos(dw+\beta)) \end{aligned} \tag{4-7}$$

onde c e d representam constantes de dilatação da janela e do filtro respectivamente e β representa o deslocamento de fase do filtro em radianos. A integral de $k(w)$ de a a b é então dada por:

$$\int_a^b k(w)dw = \frac{1}{4}\begin{bmatrix} b-a+\frac{\sin(bc)-\sin(ac)}{c} \\ +\frac{\sin(bd+\beta)-\sin(ac)}{d} \\ +\frac{\sin(b(c-d)-\beta)-\sin(a(c-d)-\beta)}{2(c-d)} \\ +\frac{\sin(b(c+d)+\beta)-\sin(a(c+d)+\beta)}{2(c+d)} \end{bmatrix} \tag{4-8}$$

É importante notar que a frequência da função onda não pode superar o limite de Nyquist imposto pela amostragem da textura.

Para produzir a animação final, o algoritmo gera N quadros filtrados por pequenos incrementos de fase β na função onda, tal que $\beta = 2\pi/N$, e os cicla por um período arbitrário.

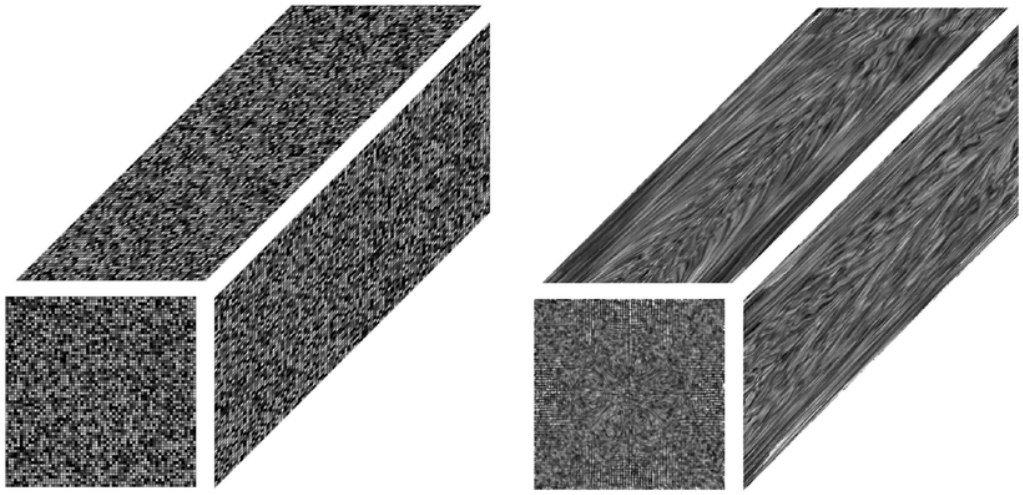


Figura 4.4: Volume de ruído branco de entrada (*esquerda*) e volume gerado por LIC (*direita*). *Fonte*: (Interrante *et al*, 1997)

4.2 LIC Volumétrico

O algoritmo de LIC é facilmente generalizável para dimensões maiores já que a linha de corrente sempre será parametrizada por um escalar s durante a convolução. As Equações 4-2, 4-3, 4-4 e 4-6 são trivialmente extensíveis para três dimensões. No caso tridimensional, arestas de células são trocadas por faces. Tanto o campo vetorial como a textura de entrada devem ser tridimensionas. A saída portanto tambem será uma imagem 3D, i.e. um campo escalar volumétrico. Este campo é então visualizado através de alguma técnica de visualização volumétrica ou pela aplicação de uma textura sobre superfícies, muitas vezes considerado na literatura como uma técnica 2.5D. A visualização volumétrica no entanto traz consigo certas dificuldades. A densidade do volume (ver Figura 4.4) normalmente não permite uma visualização clara do fluxo sem o uso de técnicas que desambiguem as informações contidas.

Primeiramente, o uso de *texturas esparsas* pode aliviar o problema de oclusão, por serem texturas de baixa frequência em comparação às texturas de ruído branco. Com texturas esparsas, as regiões nulas podem ser descartadas na visualização volumétrica deixando apenas as linhas de corrente visíveis e consequentemente mais nítidas. O problema então passa a ser o de criar sementes na textura de entrada que gerem resultados desejáveis para a aplicação em questão. Melhores resultados são obtidos quando as sementes são distribuídas por uma aproximação em disco de Poisson, ao invés de apenas distribuí-las de forma regular, aleatória ou com pequenas perturbações, como mostrado na Figura 4.5. Muitos trabalhos procuram posicionar sementes de tal forma que as linhas de corrente integradas nestes pontos não se sobreponham (Turk *et al*, 1996).

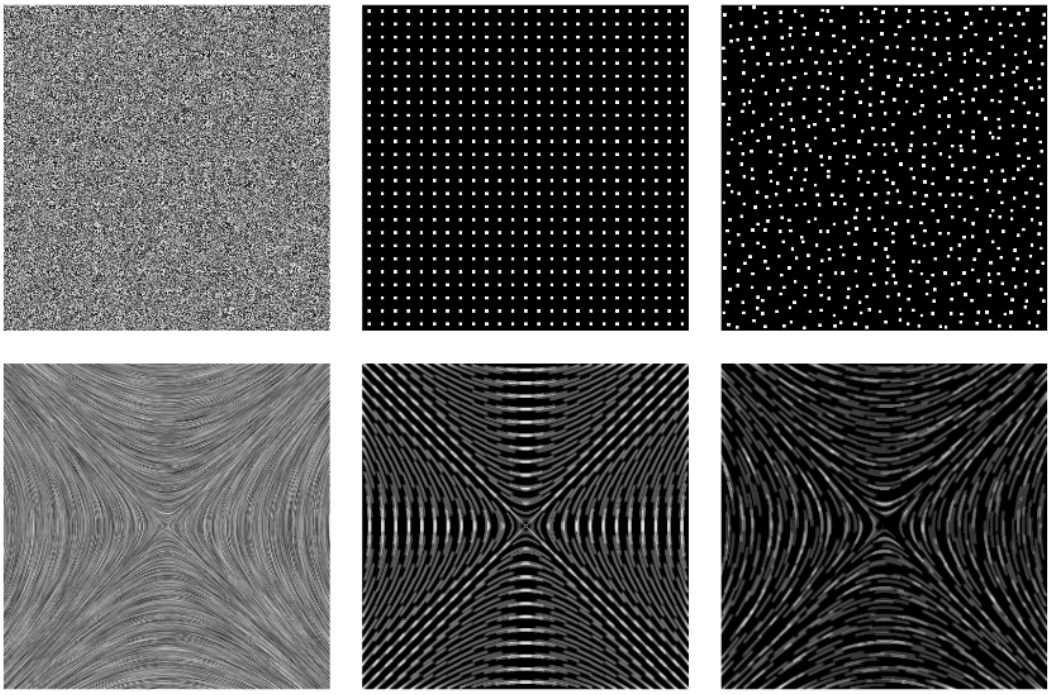


Figura 4.5: Imagens LIC (*abaixo*) geradas por diferentes texturas (*acima*). *Da esquerda para a direita*: ruído branco, esparsa regular e esparsa regular com perturbações. *Fonte*: (Wegenkittl *et al*, 1997)

(Interrante *et al*, 1997) propõem o uso de *auréolas* (Figura 4.6) que ajudam a ressaltar as discontinuidades de profundidade entre linhas de correntes, geralmente difíceis de distinguir devido à luminosidades similares das mesmas. Estas auréolas podem ser computadas durante o algoritmo de visualização volumétrica consultando uma versão dilatada da textura LIC esparsa para verificar a sobreposição de duas linhas de corrente em espaço de tela.

4.3 LIC Orientado

A técnica de LIC apresentada não consegue codificar informação de orientação sem o uso de animações. (Wegenkittl *et al*, 1997) apresentam a técnica de *Oriented LIC* – ou OLIC – para resolver este problema. Devido ao uso de texturas esparsas nesta abordagem – que consequentemente facilitam a distinção de linhas de corrente – um esquema simples de degradê sobre estas linhas produz um efeito de "rastro" em uma semente, exemplificado na Figura 4.7. Este efeito é facilmente reproduzível adaptando o kernel de convolução do algoritmo original de LIC por uma função *rampa*, como mostrado na Figura 4.8. Além de possibilitar a interpretação da direção do fluxo sem o uso de animações, a imagem final pode se tornar menos

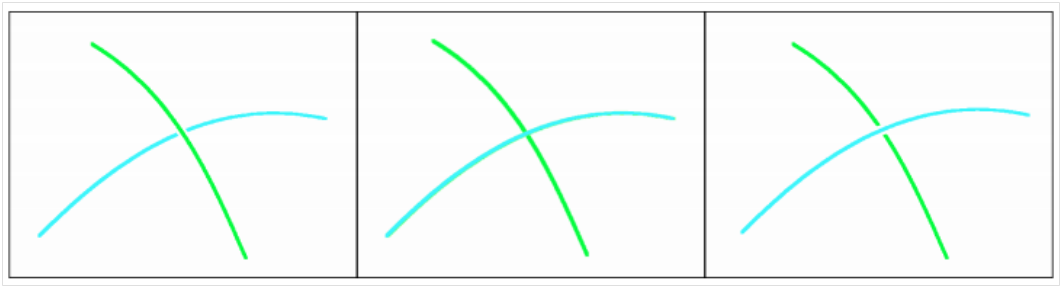


Figura 4.6: Efeito de "auréolas". A imagem central mostra linhas de corrente sem o efeito, enquanto as imagens laterais revelam diferentes suposições das mesmas. *Fonte*: (Interrante *et al*, 1997)

densa, e.g. se uma visualização volumétrica atribuir mais opacidade a elementos com lumininosidade mais intensa.

Posteriormente, (Wegenkittl *et al*, 1997) apresentam uma solução mais eficiente em tempo para o cálculo de OLIC, onde a premissa básica é que regiões nulas da textura esparsa não precisam ser pontos de partida da integração. Apenas pontos onde sementes foram selecionadas são integrados. Uma outra adaptação do método propõe aplicar uma máscara em cada elemento que foi raterizado durante a integração. Desta forma, há um controle maior sobre a densidade da imagem final, i.e. adaptando a espessura da máscara, podendo inclusive aplicá-la a um campo de distâncias a fim de passar este controle para a visualização volumétrica, por exemplo. Esta máscara pode representar qualquer formato, mas normalmente uma máscara de um círculo rasterizado (ou esfera, no caso 3D) produz resultados mais claros.

Além disso, (Wegenkittl *et al*, 1997) propõem codificar a magnitude da velocidade no tamanho de cada linha de corrente, como na Figura 4.9. Isto implica em integrar o dado em função de tempo ao invés do espaço paremétrico s como no algoritmo original. Esta abordagem acompanhada de tabelas de cores em função de magnitude traz informações ricas quanto à velocidade do campo.

Animação nesta abordagem ainda é possível através do mesmo método de deslocamento em fase do LIC original. A maior diferença entre este método e o de LIC é que devido às linhas de corrente de comprimentos diferentes, e.g. truncadas, o comprimento total de uma linha deve ser computado antes de aplicar o filtro deslocado em fase, o que permite mapear um ciclo de animação (2π) ao comprimento total de cada linha. (Wegenkittl *et al*, 1997) mostram um método de animação mais eficiente em tempo onde a idéia básica é ciclar cada elemento (e.g. pixel ou voxel) de uma linha de corrente por uma tabela de cor.

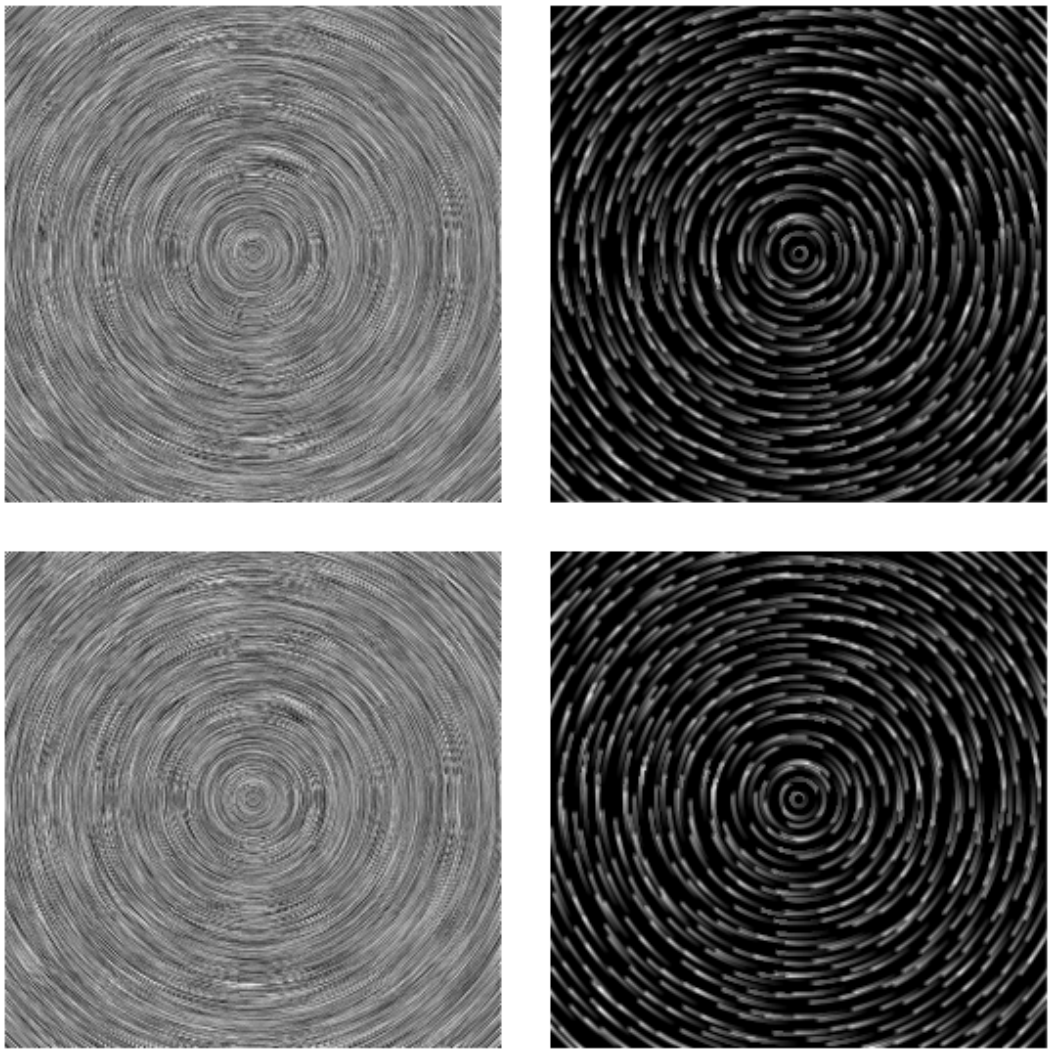


Figura 4.7: OLIC em comparação a LIC. É possível ver a orientação no sentido anti-horário (*acima*) e horário (*abaixo*). *Fonte*: (Wegenkittl *et al*, 1997)

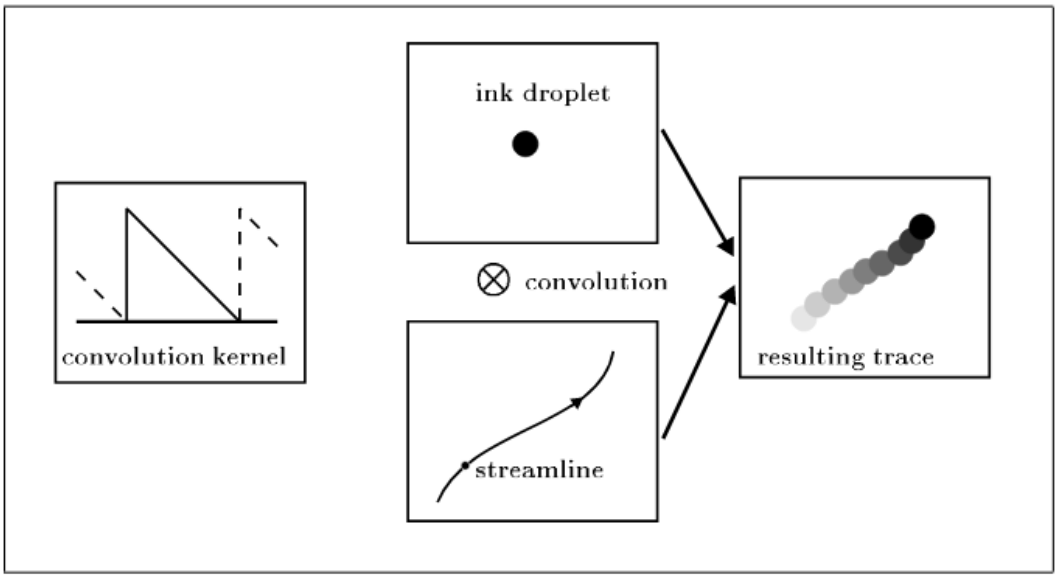


Figura 4.8: Filtro de rampa aplicado ao longo da linha de corrente integrada. O resultado é um série de círculos rasterizados (ou esferas) com luminosidades crescentes ao longo do campo vetorial. *Fonte*: (Wegenkittl *et al*, 1997)

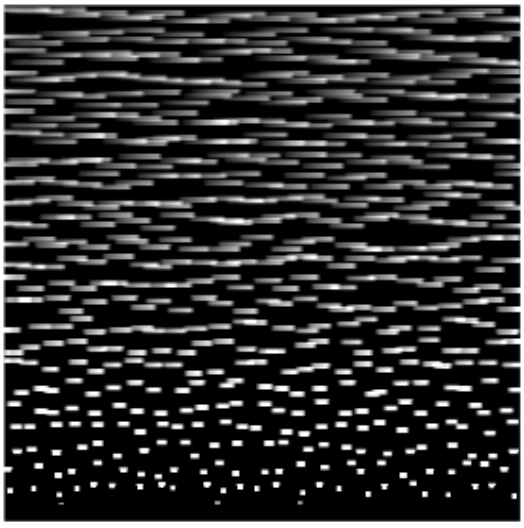


Figura 4.9: Magnitude codificada no tamanho das linhas de corrente. *Fonte*: (Wegenkittl *et al*, 1997)

5
Implementação

Nesta implementação experimental em C++11, usamos um ambiente OpenGL juntamente à OSG para controle de grafo de cena. O controle de janelas é feito em GTK. Usamos CUDA tanto para a geração de imagens LIC – devido a sua natureza paralelizável (Zöckler *et al*, 1997) – quanto para a visualização volumétrica.

Os dados de fluxo utilizados são estacionários e tridimensionais. As grades de amostragem são retangulares (não necessariamente cartesiana). Os vetores carregam informação de direção e magnitude podendo ser nulos, e.g. em regiões de falta de amostragem.

Implementamos tanto o algoritmo de LIC quanto o de OLIC para, no Capítulo 6, compararmos os resultados de cada técnica. No caso do OLIC, criamos uma textura esparsa de entrada onde sementes são inicialmente distribuídas de forma regular e posteriormente deslocadas por um pequeno fator aleatório r em uma direção aleatória θ.

Os volumes que representam tanto o campo vetorial V quanto a textura de entrada F são modelados por *arrays* de $float3$ e $uchar$, respectivamente, e uma função de acesso dadas as coordenadas (x, y, z). Estes são copiados na memória da GPU como texturas para o algoritmo de LIC. Metadados da grade como dimensão, posição em utm e tamanho do voxel também são necessários.

O algoritmo de integração numérica usado é o método de Euler. No algoritmo de LIC, uma *thread* de CUDA é disparada para cada voxel e cada *thread* é responsável pelo cálculo de uma linha de corrente (começando do centro do voxel) e por sua convolução, gerando um voxel de saída correspondente à posição do voxel de entrada. No caso do algoritmo de OLIC, onde as texturas passam a ser esparsas, apenas voxels de sementes selecionadas passam pelo processo de integração.

Uma pré-computação das linhas de corrente é inviável em GPUs atuais, devido ao custo – proporcional ao tamanho da textura de entrada e ao tamanho máximo L – de armazenar uma linha de corrente para cada voxel. Portanto, a convolução é feita em cada passo de integração, somando cada passo da convolução em variáveis que representam o numerador e denominador da Equação 4-6.

Para a animação do fluxo, N quadros, i.e. N volumes $uchar$ precisam ser computados. Isto implica que para cada voxel de entrada, N voxels de saída

são calculados. Isto é feito durante a convolução – sem que seja necessário o recálculo de integração – através de pequenos incrementos de fase no cálculo de h_i na Equação 4-5. Para evitar o envio de N volumes para a visualização volumétrica, o algoritmo retorna um volume $uint4$ que representa as imagens LIC $uchar$ compactadas. Em um $uint$ é possível compactar quatro voxels $uchar$. Logo, para um voxel $uint4$ são compactados um total de 16 voxels $uchar$, sendo então $N = 16$. O Algoritmo 1 mostra este processo de compactação. Durante a visualização volumétrica, a descompactação do volume $uint4$ (Algoritmo 2) é feita baseada em um valor de tempo que dá o índice de cada quadro.

Algoritmo 1: Algoritmo de compactação de quadros

Entrada: *Array* de voxels $uchar$ a serem compactados.

```
__host__ __device__
inline uint4 pack_uint4(uchar const* bytes) {
    uint4 p;
    for(size_t i(0); i < 16; ++i) {
        int coord = i / 4;
        int shift = (i % 4) * 8;

        if      (coord == 0) p.x |= uint(bytes[i]) << (24-shift);
        else if(coord == 1) p.y |= uint(bytes[i]) << (24-shift);
        else if(coord == 2) p.z |= uint(bytes[i]) << (24-shift);
        else                p.w |= uint(bytes[i]) << (24-shift);
    }
    return p;
}
```

Saída: Voxel $uint4$ compactado.

Algoritmo 2: Algoritmo de descompactação de quadros

Entrada: Voxel compactado p e indice i do quadro a ser descompactado.

```
__host__ __device__
inline uchar unpack_uint4(uint4 p, size_t i) {
    int coord = i / 4;
    int shift = (i % 4) * 8;

    if      (coord == 0) return (uchar)((p.x << shift) >> 24);
    else if(coord == 1) return (uchar)((p.y << shift) >> 24);
    else if(coord == 2) return (uchar)((p.z << shift) >> 24);
    else                return (uchar)((p.w << shift) >> 24);
}
```

Saída: Voxel $uchar$ descompactado.

Na visualização volumétrica implementamos um método comum de *ray casting*. Usamos algumas funções de transferência para definir cor e opacidade a certos voxels. Estas funções recebem um ponto p que é usado para extrair atributos, e.g. escalar da imagem LIC F ou velocidade do campo vetorial V em p, e retornam

uma cor RGBA. Uma função *degrau* que descarta voxels abaixo de um certo limite l pode ser definida por:

$$f_{step}(p,l) = (u,u,u,a) \text{ tq. } a = \begin{cases} 1, & \text{se } u >= l. \\ 0, & \text{caso contrário.} \end{cases} \text{ e } u = F(p) \qquad (5\text{-}1)$$

Esta função é particularmente usada em imagens LIC para inserir esparsidade que imagens OLIC já implementam.

Em seguida, definimos uma função de transferência para dar cor às magnitudes de velocidade do campo vetorial, onde C é uma tabela de cor arbitrária mapeada entre os valores mínimo e máximo de magnitude do campo vetorial:

$$f_{mag}(p) = C(\|V(p)\|) \qquad (5\text{-}2)$$

Por último, definimos uma função de transferência que mapeia a direção normalizada do campo vetorial para RGB. Esta função é particularmente interessante para ter uma visualização geral de regiões onde o fluxo não desvia muito da norma. Ela pode ser definida por:

$$f_{dir}(p) = \frac{V(p)}{\|V(p)\|} \qquad (5\text{-}3)$$

Para um efeito similar ao de "auréolas"de (Interrante *et al*, 1997), durante o traçado de raios, definimos o tempo de entrada e de saída de um raio pelo volume e então normalizamos o tempo do voxel selecionado por estes valores. A luminosidade do voxel selecionado é então multiplicada por este valor normalizado. O resultado é um sombreamento maior em voxels mais distantes da câmera o que cria uma noção de profundidade entre linhas de corrente.

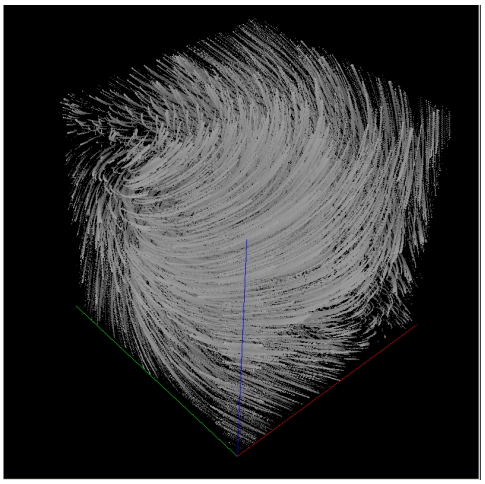
6
Resultados

Neste capítulo, apresentamos alguns resultados obtidos com os métodos descritos nesta dissertação.

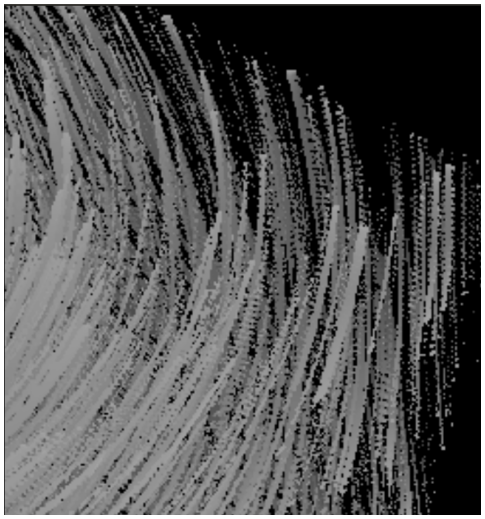
As Figuras 6.1 mostram a solução de uma equação diferencial onde a função de transferência de degrau (Equação 5-1) é aplicada para tornar a imagem menos densa. A Figura 6.1(b) mostra as profundidade entre linhas ressaltadas pelo uso de sombreamento de voxels mais distantes. A Figura 6.2 mostra outra solução de uma equação diferencial com uma tabela de cor mapeada às magnitudes dos vetores.

A Figura 6.3 mostra uma imagem 3D LIC de um modelo de reservatório em um visualizador volumétrico sem o uso de funções de transferência. Naturalmente, a densidade não permite visualização alguma de regiões internas. Com um plano de corte, visto na Figura 6.4, é possível ver o fluxo na direção das singularidades – ressaltadas pelos nódulos pretos através de normalização fixa – mas não é possível dizer a orientação.

Os resultados gerados para as equações diferenciais são extremamente nítidos quando comparados ao resultados no modelo de reservatório. Isto se deve à natureza complexa do fluxo em reservatórios. No entanto, ainda é possível detectar caracte-



6.1(a): Função degrau.



6.1(b): Descontinuidade de profundidade através de sombreamento.

Figura 6.1: Solução da equação: $\overline{F}(x, y, z) = -y\hat{i} + x\hat{j} + z\hat{k}$

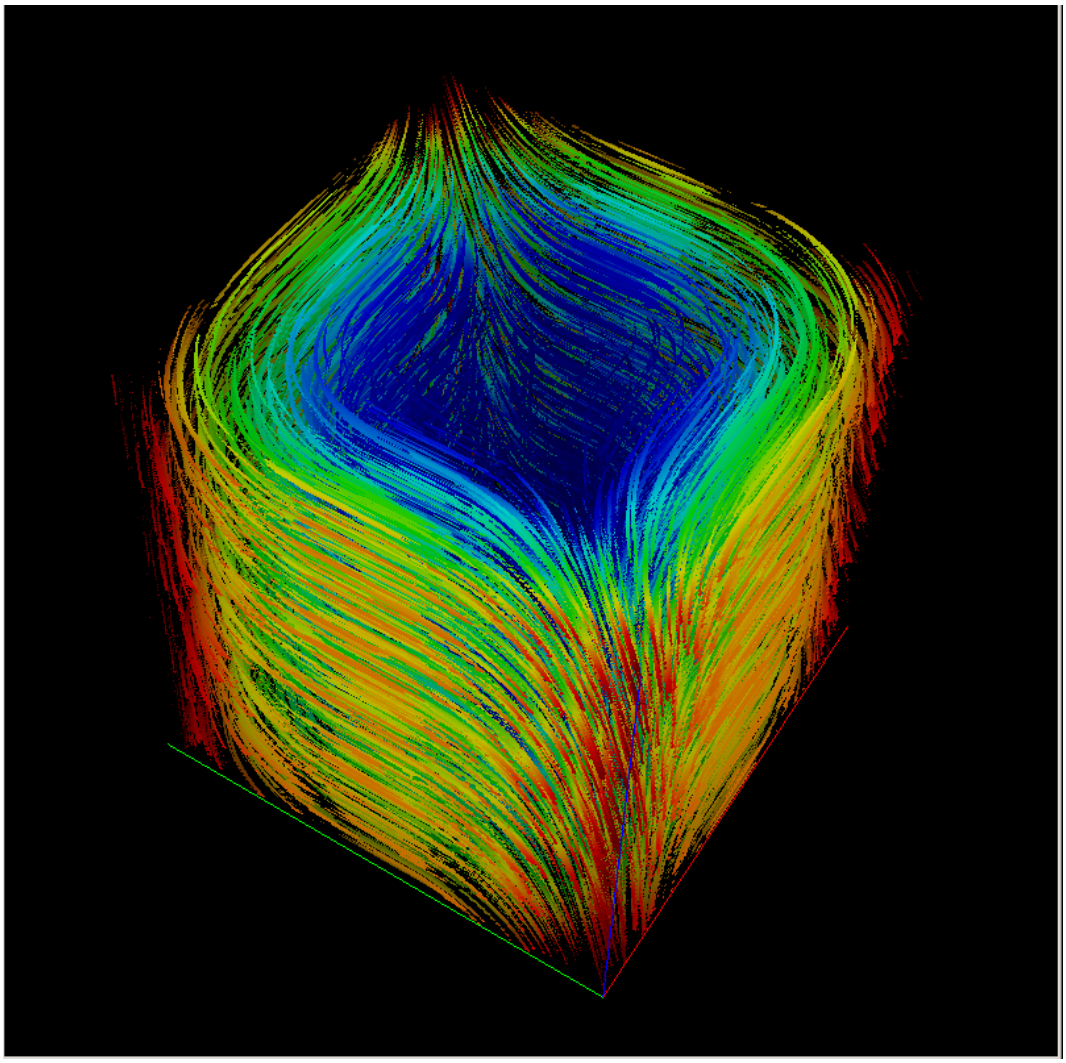


Figura 6.2: Solução da equação: $\overline{F}(x, y, z) = y^2\hat{i} + x^2\hat{j} + z\hat{k}$. Função degrau em combinação com tabela de cor de magnitude.

rísticas de interesse, como regiões de convergência e de alta velocidade do fluxo, que geralmente expressam a presença de poços.

Na Figura 6.5, mostramos uma imagem LIC do reservatório filtrada pela função degrau e com a função de transferência da Equação 5-3 aplicada. Esta função de transferência auxilia na análise geral do fluxo, mas não traz informações úteis sobre o dado. É possível notar, que mesmo com o uso de planos de corte (ver Figura 6.6), a densidade ainda não permite uma visualização clara de linhas de corrente individuais.

A Figura 6.7 mostra o resultado do algoritmo de OLIC. O uso de texturas esparsas aqui alivia bastante o problema de oclusão e ajuda a ressaltar cada linha de corrente. Além disso, é possível dizer a orientação do fluxo nestas imagens estáticas. Mostramos resultados com níveis diferentes de esparsidade. Ainda assim, regiões de baixa velocidade, que geralmente não são regiões de interesse em uma análise, apresentam uma certa complexidade, o que dificulta uma boa interpretação.

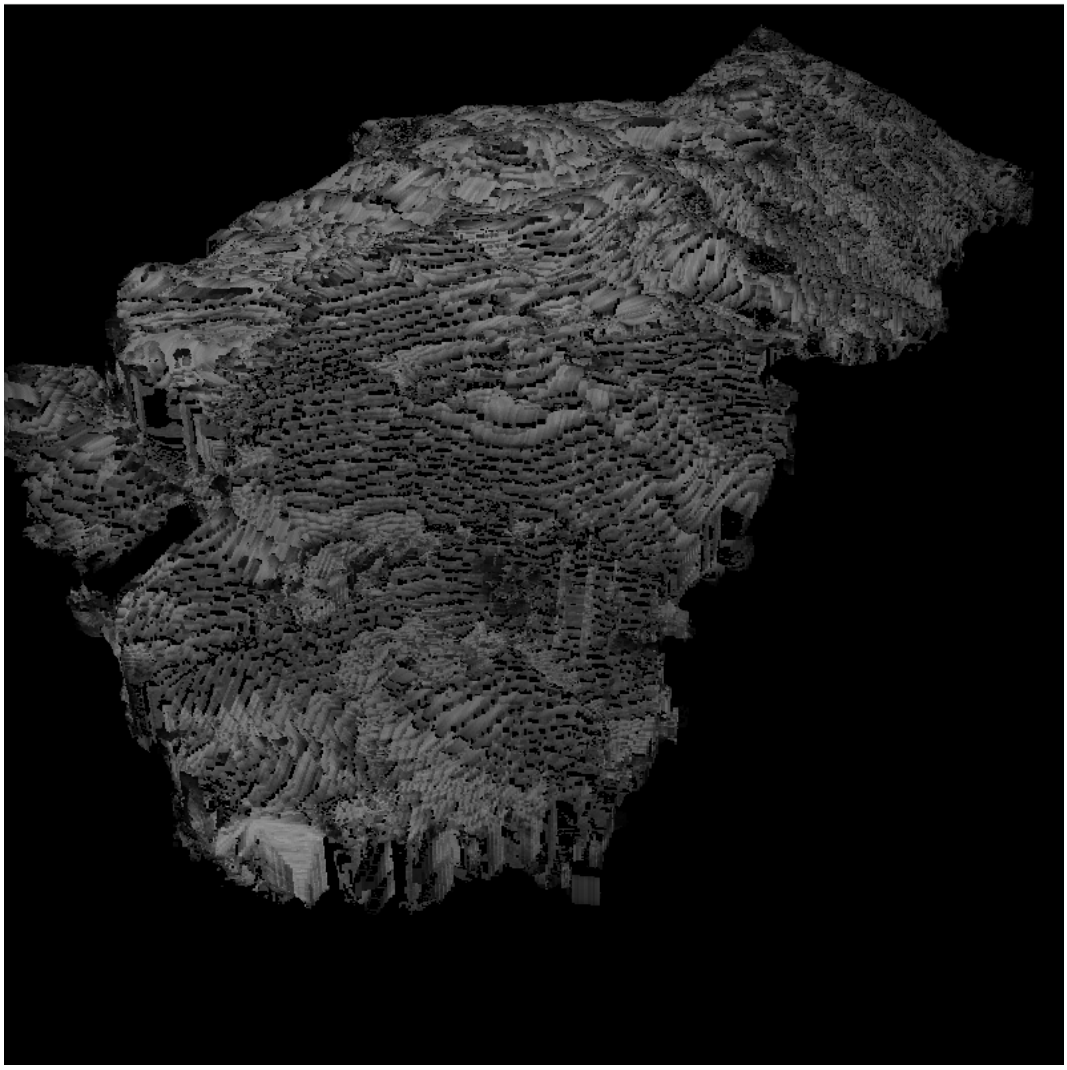


Figura 6.3: Reservatório sem funções de transferência.

As Figuras 6.8 e 6.9 mostram resultados com a mesma esparsidade de 6.7(b), porém com comprimentos de linhas de corrente variáveis. Isto reflete diretamente na densidade do volume em regiões de baixa velocidade, enquanto ressalta regiões de alta velocidade.

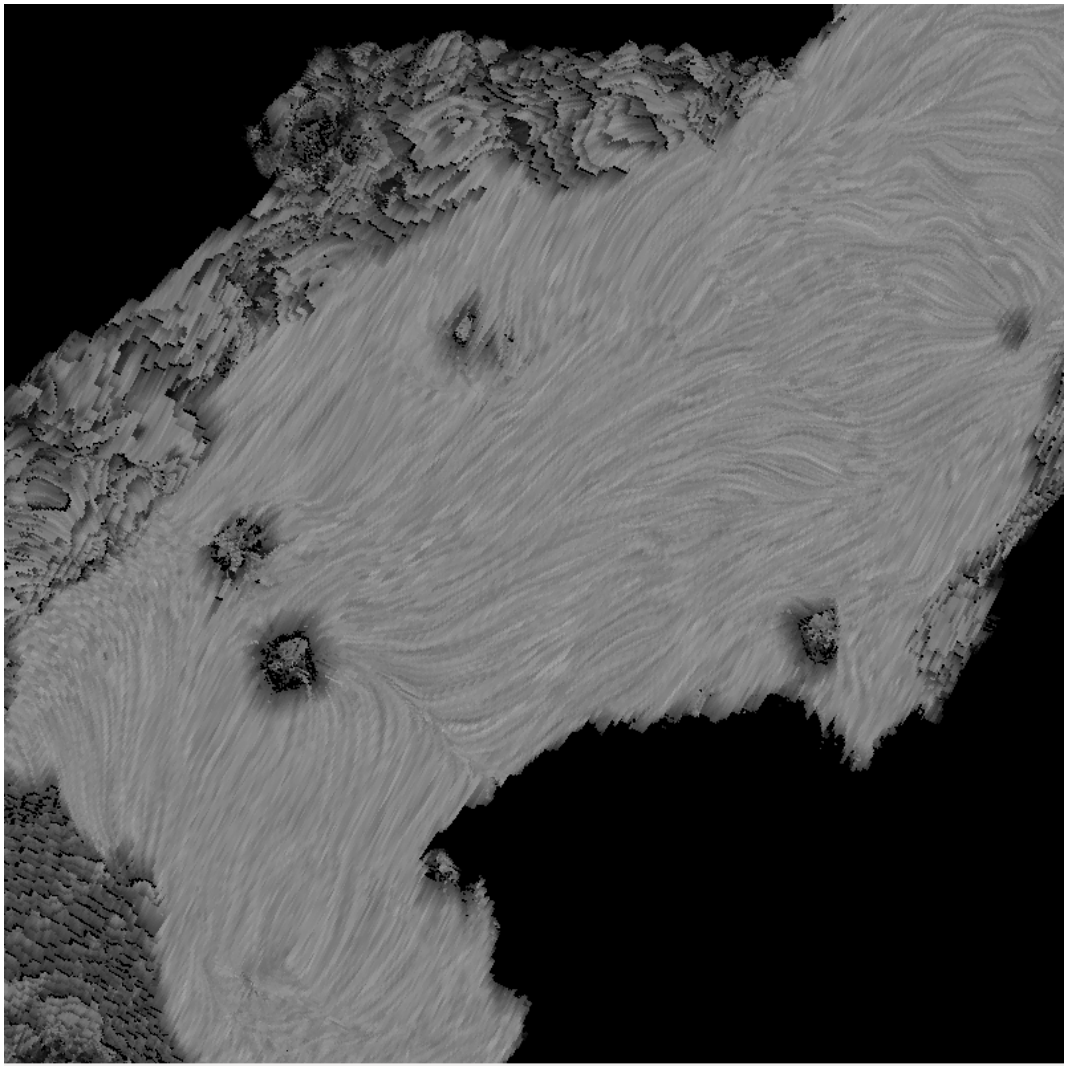


Figura 6.4: Região interna de reservatório vista com plano de corte. Singularidades ressaltadas em preto, posteriormente filtradas pela função degrau.

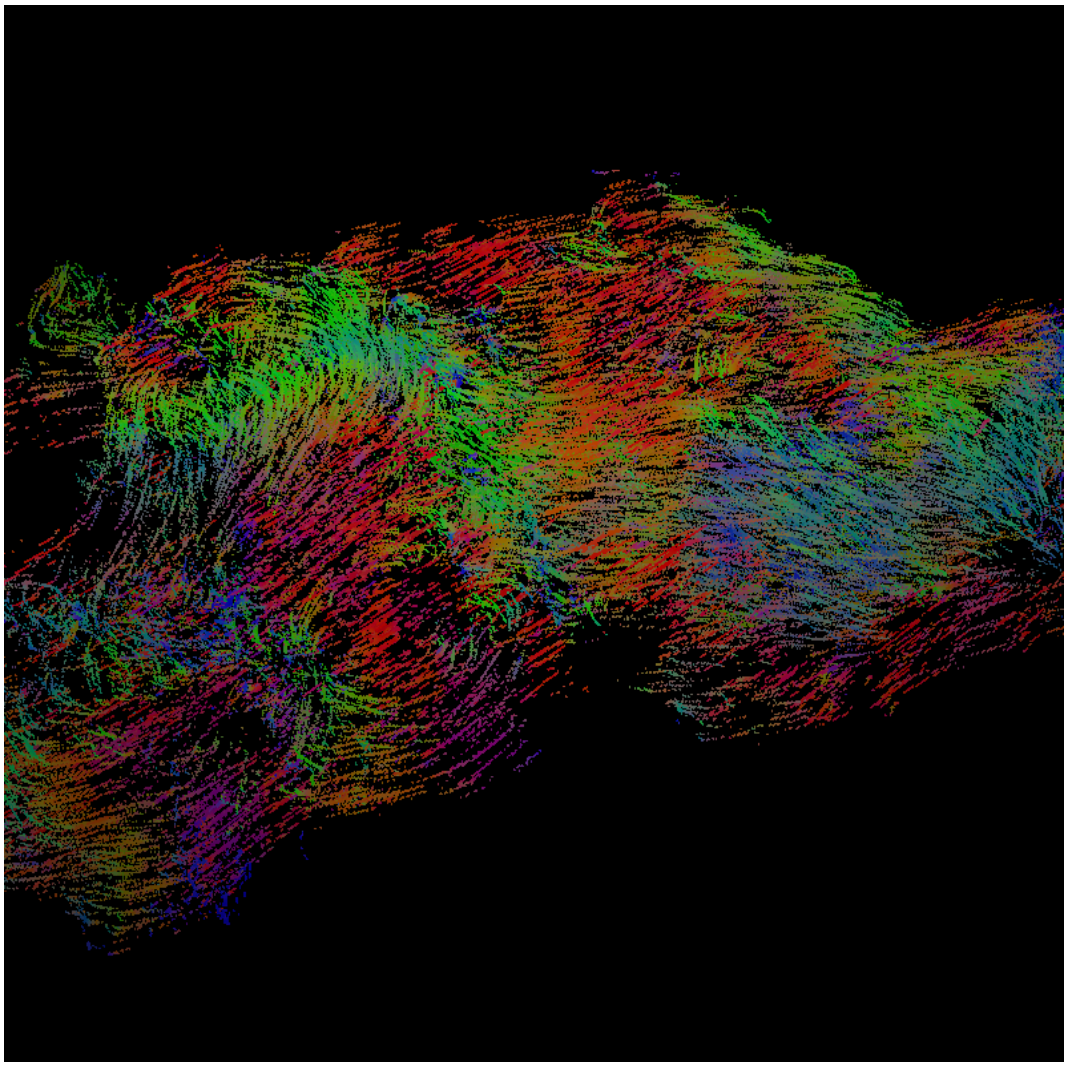


Figura 6.5: Visualização de reservatório com algoritmo LIC.

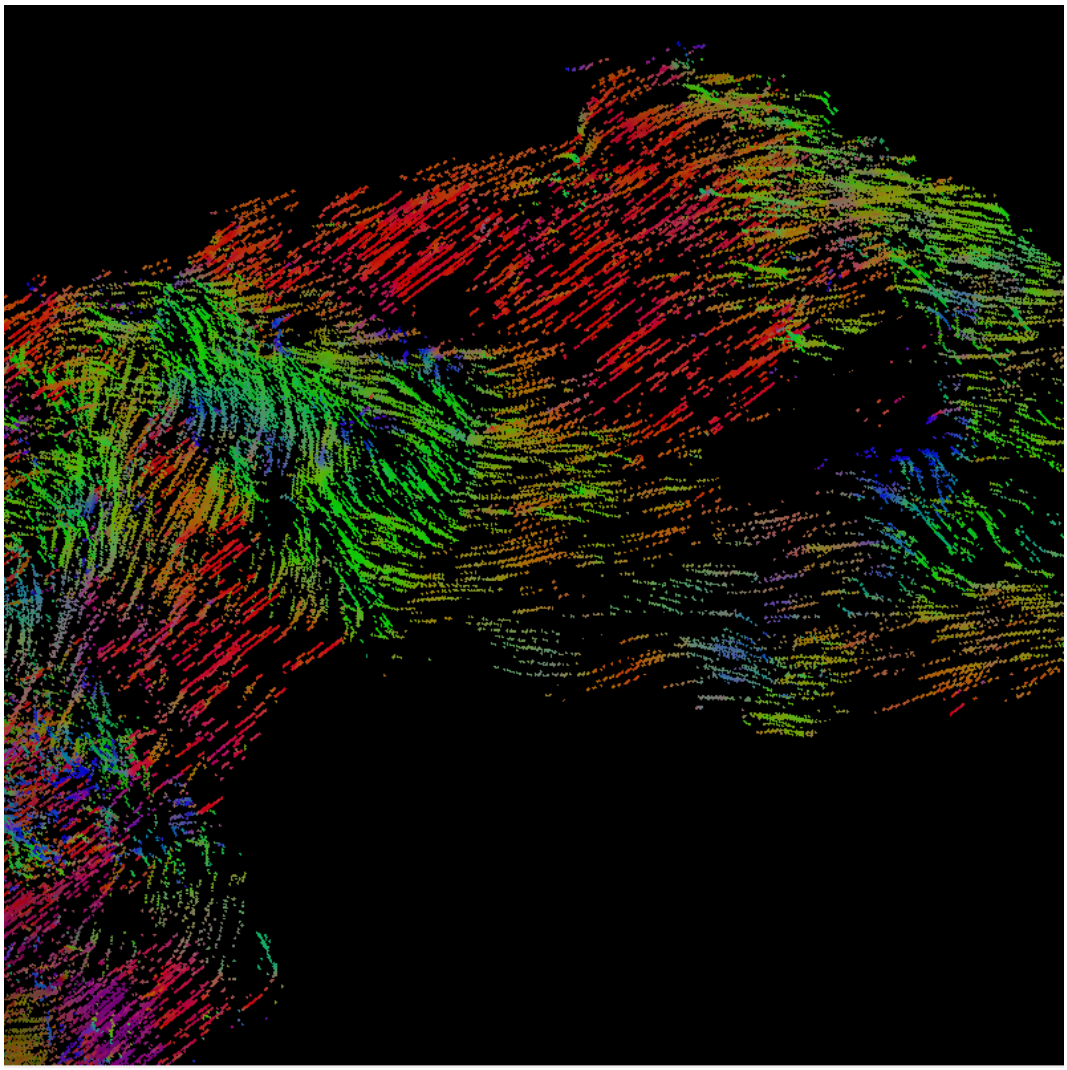
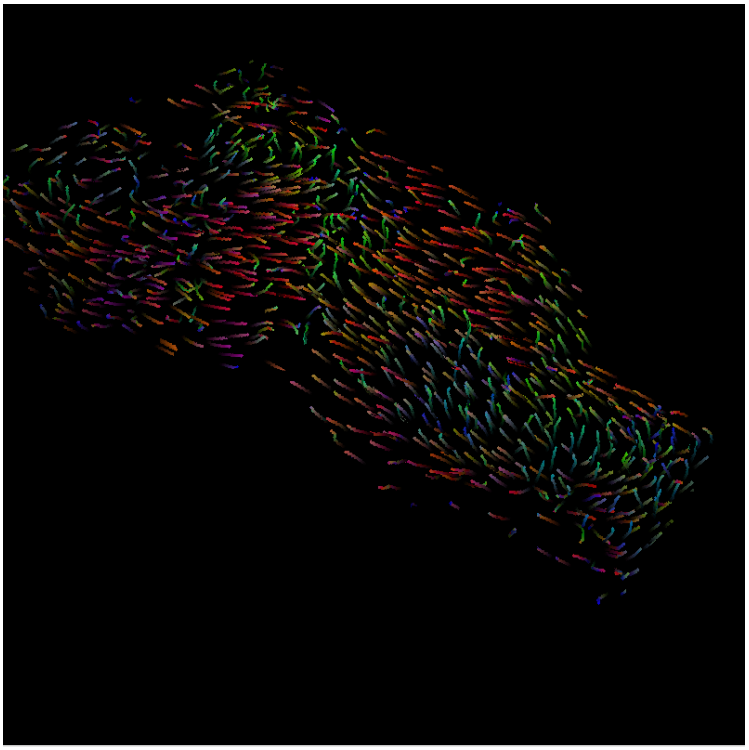
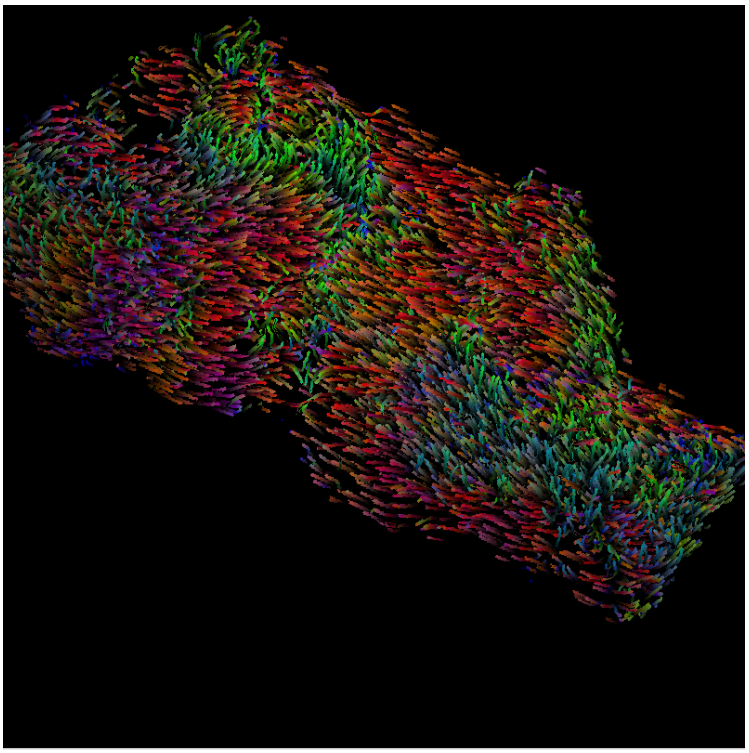


Figura 6.6: Visualização do modelo da Figura 6.5 com plano de corte.



6.7(a): OLIC com textura de entrada mais esparsa.



6.7(b): OLIC com textura de entrada menos esparsa.

Figura 6.7: Comparação de densidades de textura esparsa com a função de transferência 5-3 em modelo de reservatório.

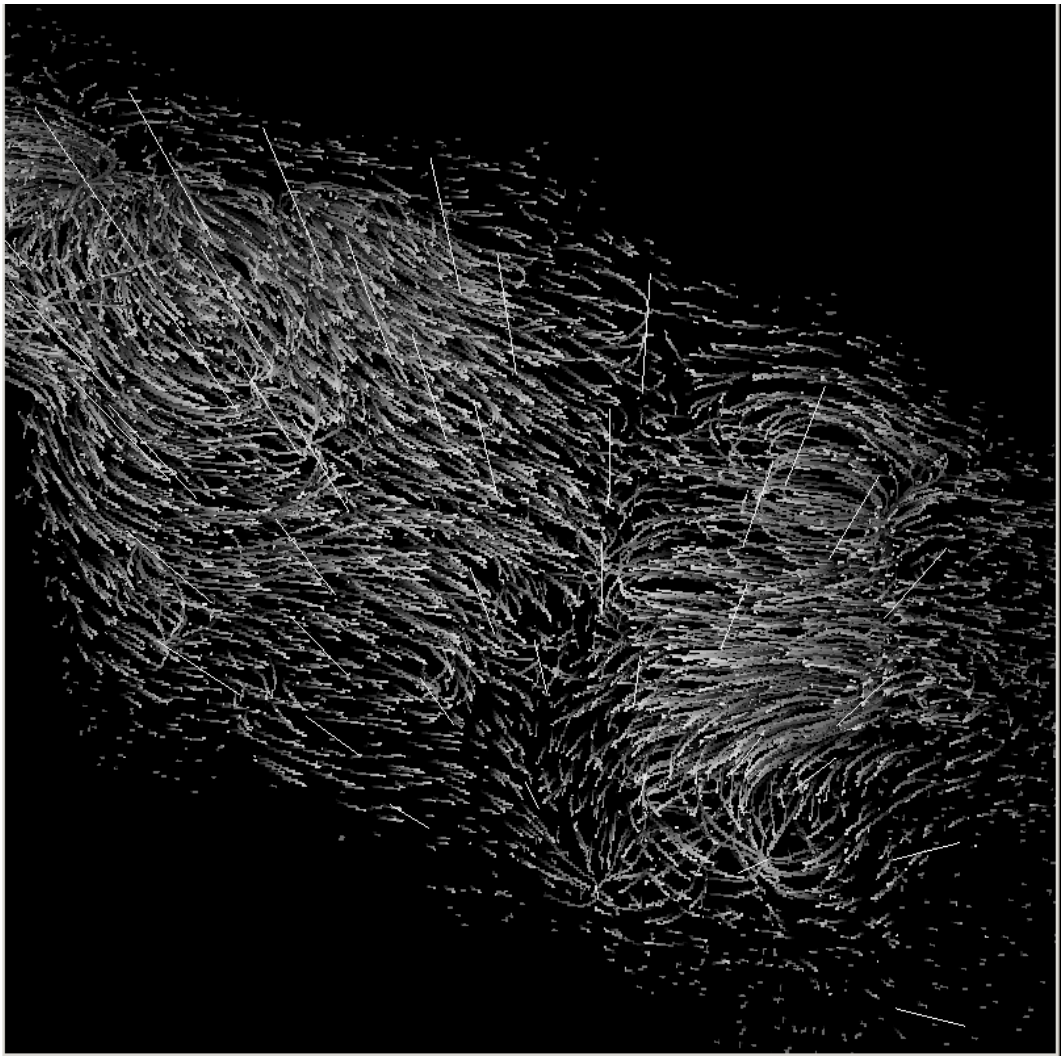


Figura 6.8: Tamanhos variantes de linhas de corrente sem tabela de cor. Linhas brancas verticas representam poços.

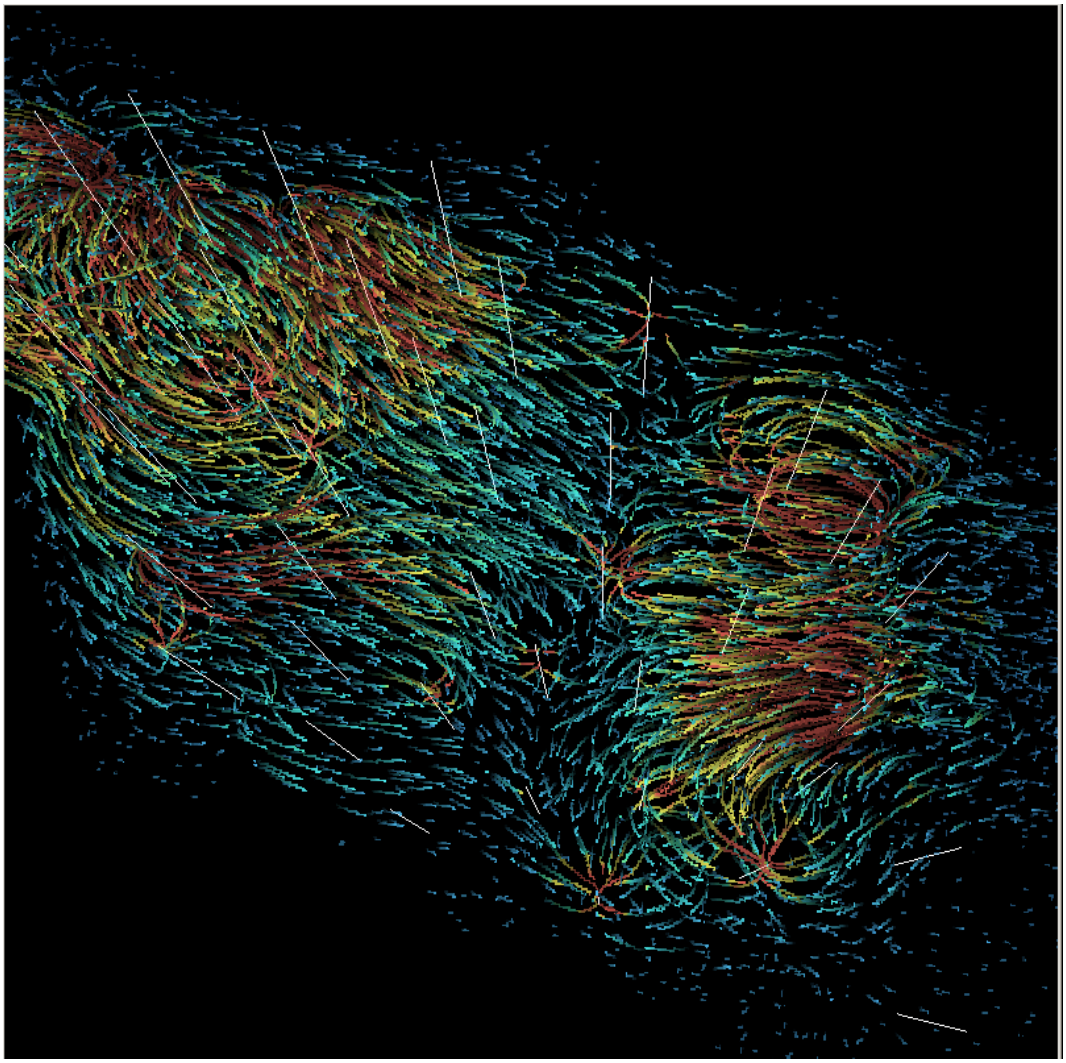


Figura 6.9: Figura 6.8 com função de transferência de magnitudes.

7
Conclusões e Trabalhos Futuros

Técnicas LIC são extremamente úteis para visualizar informações sobre o fluxo que outras técnicas de maior esparsidade simplesmente não revelam. Neste trabalho, implementamos esta técnica com dados reais de fluxo de reservatórios de petróleo. O alto custo computacional destes métodos é reduzido através de paralelismo na GPU. Artifícios como texturas esparsas e descontinuidades em profundade aliviam problemas decorrentes da densidade do volume. A orientação do fluxo é codificada através de kernels de convolução; seja através de animações com filtros periódicos ou através de gradientes de cor ao longo das linhas de corrente. Funções de transferência e comprimentos variáveis de linhas de corrente revelam a velocidade no campo vetorial e ajudam a deixar a imagem final mais clara.

Como possíveis trabalhos futuros, propomos um estudo maior com respeito ao posicionamento de sementes na textura esparsa, podendo trazer informações mais ricas do campo vetorial. Além disso, sugerimos o uso de funções de transferência mapeadas a atributos específicos de reservatórios, e.g. presença de óleo, água e gás e gradientes locais do fluxo.

Referências bibliográficas

[Cabral *et al*, 1993] CABRAL, B.; LEEDOM, L. C.. **Imaging vector fields using line integral convolution**. In: PROCEEDINGS OF THE 20TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '93, p. 263–270, New York, NY, USA, 1993. ACM. (document), 2, 4.1, 4.1, 4.2, 4.1.2, 4.3

[Freeman *et al*, 1991] FREEMAN, W. T.; ADELSON, E. H. ; HEEGER, D. J.. **Motion without movement**. SIGGRAPH Comput. Graph., 25(4):27–30, July 1991. 4.1.2

[Hege *et al*, 1998] HEGE, H.-C.; STALLING, D.. **Fast LIC with Piecewise Polynomial Filter Kernels**, p. 295–314. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. 3.3

[Interrante *et al*, 1997] INTERRANTE, V.; GROSCH, C.. **Strategies for effectively visualizing 3d flow with volume lic**. In: PROCEEDINGS OF THE 8TH CONFERENCE ON VISUALIZATION '97, VIS '97, p. 421–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press. (document), 4.4, 4.2, 4.6, 5

[Laramee *et al*, 2004] LARAMEE, R. S.; HAUSER, H.; DOLEISCH, H.; VROLIJK, B.; POST, F. H. ; WEISKOPF, D.. **The state of the art in flow visualization: Dense and texture-based techniques**. Computer Graphics Forum, 23(2):203–221, 2004. 2

[Max *et al*, 1992] MAX, N.; CRAWFIS, R. ; WILLIAMS, D.. **Visualizing wind velocities by advecting cloud textures**. In: PROCEEDINGS OF THE 3RD CONFERENCE ON VISUALIZATION '92, VIS '92, p. 179–184, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press. 2

[Park *et al*, 2004] PARK, S. W.; BUDGE, B.; LINSEN, L.; HAMANN, B. ; JOY, K. I.. **Multi-dimensional transfer functions for interactive 3d flow visualization**. In: 12TH PACIFIC CONFERENCE ON COMPUTER GRAPHICS AND APPLICATIONS, 2004. PG 2004. PROCEEDINGS., p. 177–185, Oct 2004. 3.3

[Toledo *et al*, 2011] TOLEDO, T.; CELES, W.. **Visualizing 3d flow of black-oil reservoir models on arbitrary surfaces using projected 2d line integral convolution**. In: 2011 24TH SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES, p. 133–140, Aug 2011. 2

[Turk *et al*, 1996] TURK, G.; BANKS, D.. **Image-guided streamline placement**. In: PROCEEDINGS OF THE 23RD ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '96, p. 453–460, New York, NY, USA, 1996. ACM. 4.2

[Wegenkittl *et al*, 1997] WEGENKITTL, R.; GROLLER, E. ; PURGATHOFER, W.. **Animating flow fields: rendering of oriented line integral convolution**. In: PROCEEDINGS. COMPUTER ANIMATION '97 (CAT. NO.97TB100120), p. 15–21, June 1997. (document), 4.5, 4.3, 4.3, 4.7

[Wegenkittl *et al*, 1997] WEGENKITTL, R.; GROLLER, E.. **Fast oriented line integral convolution for vector field visualization via the internet**. In: PROCEEDINGS. VISUALIZATION '97 (CAT. NO. 97CB36155), p. 309–316, Oct 1997. (document), 4.3, 4.3, 4.8, 4.9

[Wijk *et al*, 1991] VAN WIJK, J. J.. **Spot noise texture synthesis for data visualization**. SIGGRAPH Comput. Graph., 25(4):309–318, July 1991. 2

[Zöckler *et al*, 1997] ZÖCKLER, M.; STALLING, D. ; HEGE, H.-C.. **Parallel line integral convolution**. Parallel Comput., 23(7):975–989, July 1997. 5