Pontifícia Universidade Católica
do Rio de Janeiro

Renata Thomaz Lins do Nascimento

Visualização por Imagens Auto-animadas de Campos Vetoriais Baseada na sua Topologia

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Matemática Aplicada do Departamento de Matemática da PUC–Rio

Orientador: Prof. Thomas Lewiner

Rio de Janeiro
Março de 2011

Pontifícia Universidade Católica
do Rio de Janeiro

Renata Thomaz Lins do Nascimento

Visualização por Imagens Auto-animadas de Campos Vetoriais Baseada na sua Topologia

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Matemática Aplicada do Departamento de Matemática do Centro Técnico Científico da PUC–Rio. Aprovada pela comissão examinadora abaixo assinada.

Prof. Thomas Lewiner
Orientador
Departamento de Matemática — PUC–Rio

Prof. Luiz Velho
Instituto Nacional de Matemática Pura e Aplicada - IMPA

Prof. Esteban Clua
Instituto de Computação – UFF

Prof. Sinésio Pesco
Departamento de Matemática – PUC–Rio

Prof. Hélio Lopes
Departamento de Matemática – PUC–Rio

Prof. José Eugenio Leal
Coordenador do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 24 de Março de 2011

Renata Thomaz Lins do Nascimento

Graduou–se Bacharel em Ciência da Computação na Universidade Federal de Alagoas – UFAL.

Ficha Catalográfica

Nascimento, Renata

Visualização por Imagens Auto-animadas de Campos Vetoriais Baseada na sua Topologia / Renata Thomaz Lins do Nascimento; orientador: Thomas Lewiner. — Rio de Janeiro : PUC–Rio, Departamento de Matemática, 2011.

v., 61 f: il. ; 29,7 cm

1. Dissertação (Mestrado em Matemática Aplicada) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática.

Inclui referências bibliográficas.

1. Matemática – Tese. 2. Campo vetorial. 3. Visualização. 4. Métodos topológicos. 5. Imagem auto-animada. 6. Edição baseada na topologia. 7. Computação Gráfica.
I. Lewiner, Thomas. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. III. Título.

CDD: 510

Agradecimentos

Agradeço a Deus por ter me dado força e coragem de seguir adiante.

Aos meus pais Urânia e Thomaz e à minha irmã Bárbara por estarem sempre ao meu lado e por me ajudarem a realizar meus sonhos.

Ao meu tio Antônio Célio por todo apoio e incentivo durante minha caminhada.

Ao meu orientador Thomas Lewiner por toda paciência e dedicação durante o desenvolvimento do trabalho e principalmente pela amizade que construímos.

Aos amigos Cabral, Douglas, Allan, Thales, Aninha, Clarissa, Sabrina, Lis, Maria por toda ajuda e companhia nos últimos anos, e em especial ao João Paixão, pelas horas e horas de discussão e companheirismo.

Aos funcionários do departamento de Matemática por toda ajuda.

Ao CNPq, CAPES, FAPERJ e à PUC-Rio, pelos auxílios sem os quais esse trabalho não poderia ter sido concretizado.

Resumo

Nascimento, Renata; Lewiner, Thomas. **Visualização por Imagens Auto-animadas de Campos Vetoriais Baseada na sua Topologia**. Rio de Janeiro, 2011. 61p. Dissertação de Mestrado — Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

A visualização de campos vetoriais é uma componente essencial de numerosas aplicações, em particular na Visualização Científica. Porém, produzir representações de um fluxo nem sempre é uma tarefa simples, principalmente em se tratando de dados medidos, pois estes se apresentam corrompidos por ruídos.

Esse trabalho apresenta uma técnica de visualização baseada em imagens auto-animadas, que expressa o movimento do fluxo à base de ilusões ópticas. A utilização de informações topológicas é proposta tanto como forma de melhorar o desempenho das técnicas existentes como na remoção de ruído, onde o conhecimento do usuário sobre o dado se torna peça fundamental no processo.

Palavras–chave

Campo vetorial Visualização Métodos topológicos Imagem auto-animada Edição baseada na topologia Computação Gráfica

Abstract

Nascimento, Renata; Lewiner, Thomas (Adviser). **Topology-aware Vector Field Visualization by Self-animating Images**. Rio de Janeiro, 2011. 61p. MSc. Thesis — Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Vector field visualization is an essential component of various applications, particularly in Scientific Visualization. However generating useful flow representation is not a simple task, especially when dealing with measured data which is corrupted by noise.

This work presents a self-animating image visualization technique which conveys the flow movement based on optical illusions. The field's topological information is used to improve the performance of existing techniques and remove noise, where the user's knowledge of data is fundamental.

Keywords

Vector field; Visualization; Topological methods; Self-animated images; Topology-aware editing; Computer Graphics.

Sumário

1 Introdução **11**
1.1 Motivação e Objetivos 12
1.2 Trabalhos Relacionados 13

2 Conceitos Básicos **18**
2.1 Campo Vetorial e Fluxo 18
2.2 Grafo Topológico 22
2.3 Campo Discreto 23

3 Técnicas Utilizadas **24**
3.1 Detecção de Singularidades 24
3.2 Espaço de Escala 28
3.3 Geração das Linhas de Fluxo 29
3.4 Imagens Auto-Animadas e Otimização 31

4 Técnicas Desenvolvidas **35**
4.1 Edição de Campos Vetoriais - Remoção de Ruidos 35
4.2 Segmentação do Campo Vetorial 38
4.3 Imagens Auto-Animadas e Grafo Topológico 41

5 Resultados **45**

6 Considerações Finais e Trabalhos Futuros **57**

Referências Bibliográficas **59**

Lista de figuras

1.1 *Rotating snakes* 11
1.2 Linhas de fluxo igualmente espaçadas 14
1.3 Passos intermediários na geração das linhas 14
1.4 Visualização utilizando LIC 15
1.5 Esquema topológico de um fluxo em volta de um cilindro 15
1.6 Visualização de campos através de PAR 16
1.7 Visualização de campos através de PAR - método computacional 16
1.8 Filtragem de campos vetoriais 17
1.9 Espaço de escala 17

2.1 Singularidade - sela 20
2.2 Singularidade - poço 20
2.3 Singularidade - fonte 21
2.4 Órbita fechada 22
2.5 Grafo topolólgico 23
2.6 Grade regular 23

3.1 Interpolação bilinear 24
3.2 Curva com índice -2 em volta do ponto p 25
3.3 Curva no sentido anti-horário 26
3.4 Curva no sentido horário 26
3.5 Regiões fracas 27
3.6 Espaço de escala em imagens 29
3.7 Espaço de escala em campos vetoriais 29
3.8 Esquema de criação das linhas de fluxo 30
3.9 Linhas de fluxo 31
3.10 Otimização da ilusão de Fraser-Wilcox 31
3.11 Otimização do fragmento 34

4.1 Campo vetorial artificial 35
4.2 Campo vetorial artificial na escala s = 10 36
4.3 Interface de mudanças topológicas 37
4.4 Interpolação e reconstrução 38
4.5 Segmentação do campo sem tratamento de bordo 39
4.6 Exemplos de segmentação do campo sem tratamento de bordo 39
4.7 Esquema de segmentação do campo sem tratamento de bordo 40
4.8 Esquema tratamento de bordo - primeira parte 40
4.9 Segmentação a partir das separatrizes 41
4.10 Segmentação com a primeira parte de tratamento de bordo 41
4.11 Segmentação final 42
4.12 Posicionamento dos fragmentos desconsiderando a segmentação 42
4.13 Posicionamento dos fragmentos considerando a segmentação 43
4.14 Otimização considerando a segmentação do campo 44

5.1 Edição: Resultado 1 - Experimento em campo vetorial analítico 46

5.2 Edição: Resultado 2 - Campo vetorial medido por um PIV 47
5.3 Visualização: Resultado 1 - Campos estáticos 48
5.4 Visualização: Resultado 2 - Campo com duas singularidades 49
5.5 Visualização: Resultado 3 - Campo com quatro singularidades 50
5.6 Visualização: Resultado 4 - Campo com quatro singularidades 51
5.7 Visualização: Resultado 5 - Campo com seis singularidades 52
5.8 Grafos topológicos dos resultados 53
5.9 Edição e visualização: Resultado 6 - Dado analítico 54
5.10 Edição e visualização Resultado 7 - PIV 55

Dificuldades são como as montanhas: só se aplainam quando avançamos sobre elas.

Émile Zola.

1
Introdução

Uma ilusão de ótica ou ilusão visual é caracterizada por uma imagem que é percebida de maneira diferente da imagem real. Essas imagens enganam o sistema visual podendo adicionar à imagem percebida objetos que não estão na imagem real ou até inferir uma idéia de movimento. Por exemplo, em *Rotating snakes*, Figura 1.1, podemos observar uma ilusão de movimento criada por Kitaoka (12), onde os círculos aparentam estar se movendo espontaneamente embora se trate de uma imagem estática.

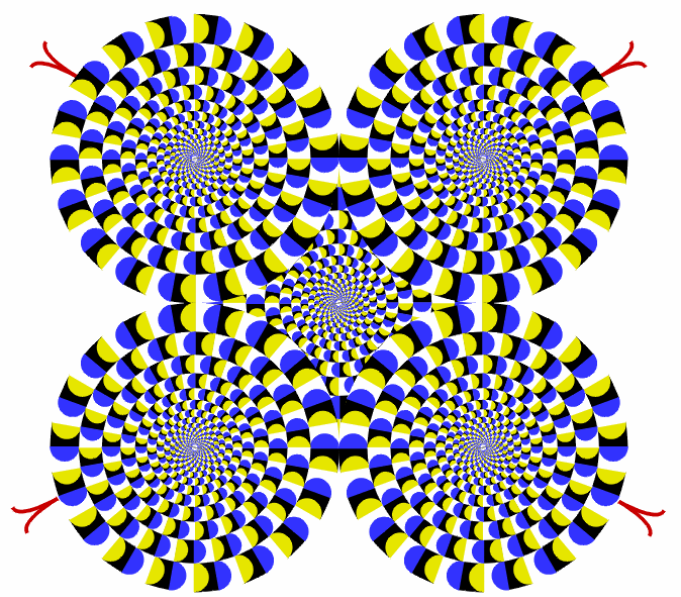


Figura 1.1: *Rotating snakes* (Figura extraída do artigo original (12)).

Em seus estudos, Kitaoaka (14) mostrou que a combinação de cores que mais intensificam a ilusão de movimento é azul-amarelo ou vermelho-verde. Além disso, observou que 5% das pessoas não podem perceber esse tipo de ilusão. Baseado neste trabalho, Chi *et al.*(6), introduziram a conversão automática de uma imagem estática numa que possuísse a ilusão de movimento, criando um campo de vetores intermediário.

A visualização de campos vetoriais é um assunto que tem recebido atenção por suas numerosas aplicações e sua dificuldade. De fato, visualizar de

forma nítida a direção e magnitude de um fluxo, por exemplo, nem sempre é uma tarefa simples, principalmente quando tratamos de fluxos turbulentos.

Analisar um campo vetorial consiste em entender o comportamento de suas linhas de fluxo. Encontrar regiões características como *poços*, *fontes* e *selas*, pode ser um passo interessante para a visualização pois são essas regiões que determinam a dinâmica do campo vetorial.

De fato, a visualização de campos vetoriais por imagens auto-animadas, embora atrativa por expressar a movimentação do fluxo dentro do campo vetorial, é um método extremamente custoso pois as etapas geração e otimização do posicionamento dos fragmentos, se tratam de processos força bruta. O objetivo deste trabalho consiste em utilizar análises locais do campo de forma a otimizar independentemente cada região, garantindo a coerência global da imagem auto-animada.

1.1 Motivação e Objetivos

Visando acelerar o processo de visualização de campos vetoriais por imagens auto-animadas, propomos uma etapa de pré-processamento que se utiliza da topologia do campo vetorial para segmentá-lo.

A segmentação particiona o campo em componentes conexas formadas por um par de pontos singulares, podendo ser selas, poços, fontes ou singularidades de bordo, e por linhas de fluxo que apresentam comportamento semelhante – por comportamento semelhante denotamos linhas de fluxo que têm duas singularidades em comum (Seção 4.2).

Em posse da segmentação, é possível realizar análises de forma local no campo vetorial já que dispomos da possibilidade de focar em regiões específicas. Propomos a utilização desta como forma de acelerar o processo de geração das imagens auto-animadas (Seção 4.3), onde a principal contribuição no processo ocorre na etapa de *Otimização do Posicionamento dos Fragmentos* do trabalho proposto por Chi *et al.* (6).

Propomos também um método para a detecção de pontos com potencial para ser um ponto singular (Seção 3.1.3) o qual denominamos *Regiões Fracas*.

Além disso, propomos uma técnica semi-automática para a remoção de ruído em campos vetoriais (Seção 4.1), onde o usuário controla as mudanças topológicas resultantes dos processos tradicionais de filtragem de campos vetoriais.

Esta técnica é aplicável, principalmente, em dados reais os quais estão sujeitos a ruído em diversas escalas devido ao processo de medição.

1.2 Trabalhos Relacionados

Linhas de fluxo: A eficiência de técnicas de visualização de campos vetoriais por meio de *linhas de fluxo* é muito sensível ao posicionamento das sementes que irão originá-las. Um trabalho pioneiro nessa área foi proposto por Turk e Banks (33), onde a idéia principal consiste em minimizar uma energia para gerar posicionamento de sementes com qualidade. O algoritmo consiste em gerar um conjunto de pequenas linhas de fluxo - *streamlets* - e aplicar uma série de operações como união, remoção, aumento ou diminuição de comprimento, de forma a diminuir a energia. A energia é calculada fazendo a diferença entre uma imagem cinza uniforme e uma imagem criada do atual posicionamento com um filtro passa-baixa.

Posteriormente, Jobard e Lefer (10) propuseram um algoritmo para criar linhas de fluxo igualmente espaçadas em apenas uma passada (ver Figura 1.2 à esquerda). Esse método consiste em posicionar sementes na vizinhança de linhas de fluxo já posicionadas. Enquanto uma linha de fluxo está sendo integrada, sementes são posicionadas de ambos os lados da linha, de forma que essas semente têm prioridade para a construção da próxima linha de fluxo. Embora esse algoritmo proporcione um bom equilíbrio entre eficiência e tempo de computação, no resultado final aparecem espaços em branco além de linhas curtas ou quebradas. Mebarki *et al.* (17) tentou obter linhas de fluxo longas e igualmente espaçadas (ver Figura 1.2 à direita). O algoritmo consiste em posicionar uma linha de fluxo por vez, onde a semente que irá originá-la, está no ponto mais longe possível de todas as linhas posicionadas posteriormente. Para calcular essa semente é feita uma triangulação de Delaunay do domínio, sendo a semente o centro do maior círculo circunscrito da triangulação (ver Figura 1.3).

Visualização por imagens: A visualização de fluxos é frequentemente obtida pelo método de convolução da integral da linha (LIC) proposto por Cabral e Leedom (5) (ver Figura 1.4). O LIC é uma técnica popular para visualização densa de campos vetoriais. A metodologia para gerar o LIC consiste em pegar um campo vetorial num grid cartesiano e uma textura de ruído branco do mesmo tamanho. A textura de ruído é localmente filtrada ao longo das linhas de fluxo que são definidas pelo campo vetorial, resultando numa imagem que é uma representação densa do campo. Embora essa técnica mostre detalhes interessantes do campo, o custo para computá-la é alto pois necessita de um grande número de linhas de fluxo por *pixel*.

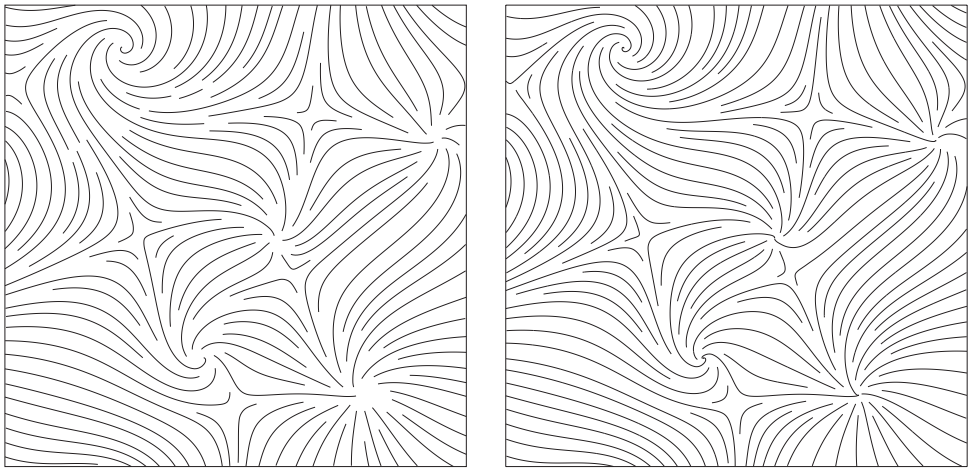


Figura 1.2: Linhas de fluxo igualmente espaçadas. Esquerda: técnica desenvolvida por Jobard e Lefer (Figura extraída do artigo original (10)); Direita: técnica desenvolvida por Mebarki *et al.* (Figura extraída do artigo original (17)).

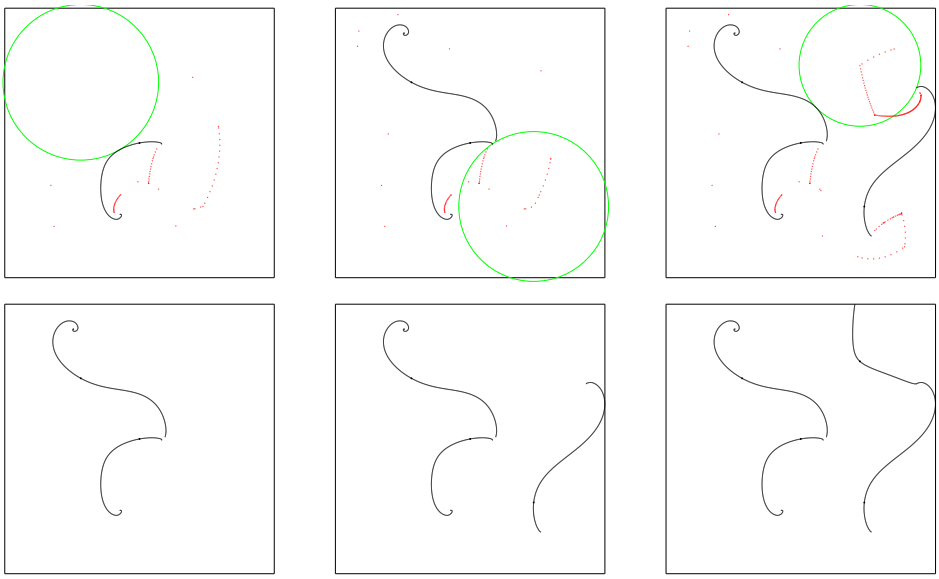


Figura 1.3: Passos intermediários na geração das linhas. Em cada etapa é escolhida uma semente que consiste no centro do maior círculo de Delaunay (Figura extraída do artigo original (17)).

Visualização com topologia: Outra forma de ser feita a análise de campos vetoriais planares é por meio de métodos topológicos (7, 8). Neste caso, é obtido um grafo que representa as relações topológicas entre linhas de fluxo vizinhas (ver Figura 1.5). Embora essa metodologia não seja muito antiga na área de Visualização Científica, tem suas origens em um trabalho de Poincaré no final do século XIX. O objetivo consiste em particionar o domínio por meio de um grafo em sub-regiões que apresentem comportamento semelhante. Para a construção do grafo topológico é feita a extração dos pontos críticos

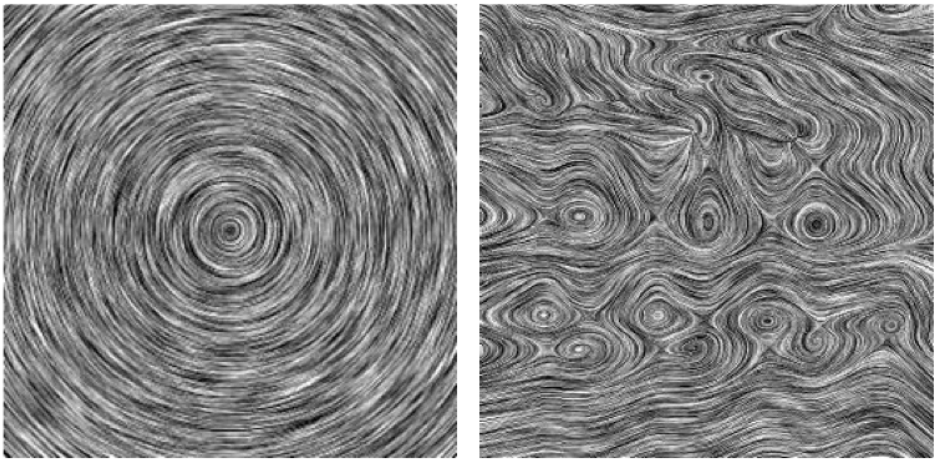


Figura 1.4: Visualização por LIC (Figura extraída do artigo original (5)).

e a integração das linhas de fluxo que conectam tais pontos. Essas linhas são conhecidas como *separatrizes*. Em Tricoche (31) e Tricoche *et al.* (32) são apresentados métodos para a simplificação da topologia de campos vetoriais planares turbulentos mantendo a consistência estrutural com o dado original.

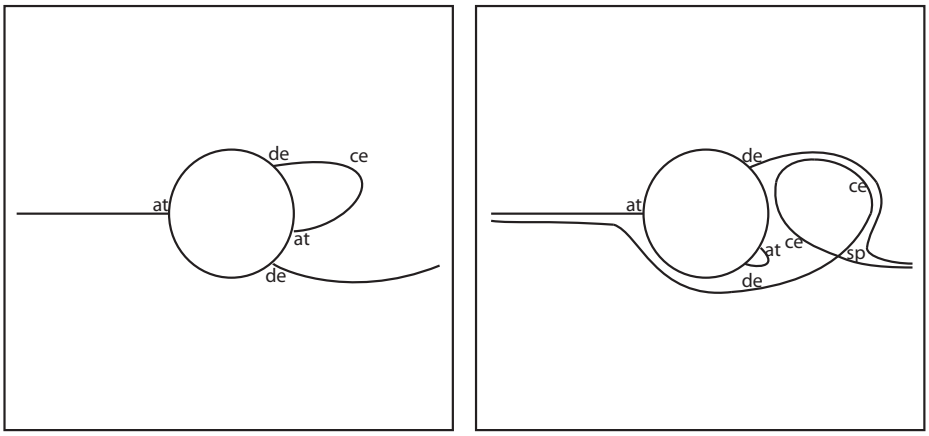


Figura 1.5: Esquema topológico de um fluxo em volta de um cilindro (Figura extraída do artigo original (8)).

Outra estratégia ainda, se utiliza do conhecimento do usuário sobre o campo vetorial, permitindo que de forma interativa, o usuário decida que singularidade topológica manter ou suavizar. Essa abordagem já foi proposta no campo de reconstrução de superfícies (11, 23) e foi estendida durante a preparação deste trabalho (18).

Mais recentemente, uma forma não tão usual para a visualização de campos consiste na produção de imagens auto-animadas. Esse modelo um tanto criativo, usa uma área de percepção humana onde uma ilusão de movimento é criada para uma imagem estática.

Wei (34), propôs um método completamente automático para visualizar campos vetoriais utilizando um padrão assimétrico repetido de cores (PAR). Para gerar essa visualização, círculos são dispostos ao longo do campo seguindo por exemplo, uma distribuição uniforme (ver Figura 1.6).

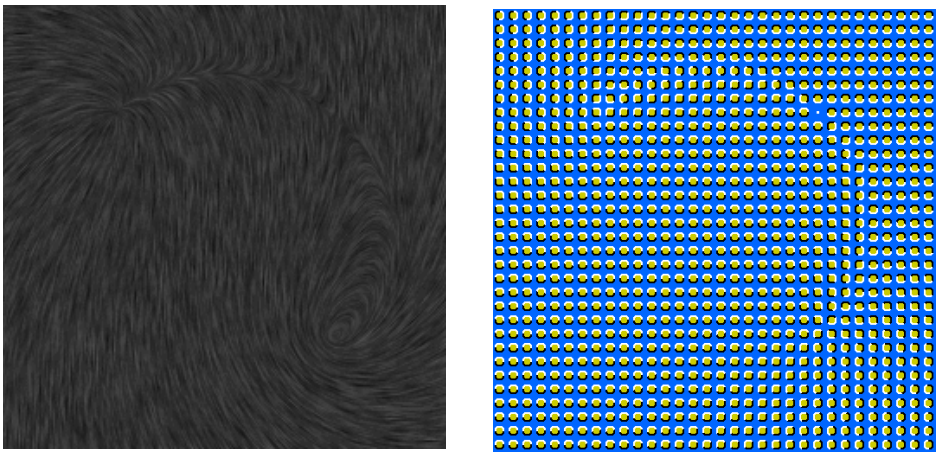


Figura 1.6: Visualização proposta por Wei. Esquerda: campo de entrada visualizado com LIC; Direita: visualização de Wei (Figura extraída do artigo original (34)).

Chi *et al.*(6) desenvolveram uma abordagem computacional para maximizar o efeito da ilusão de movimento (ver Figura 1.7).

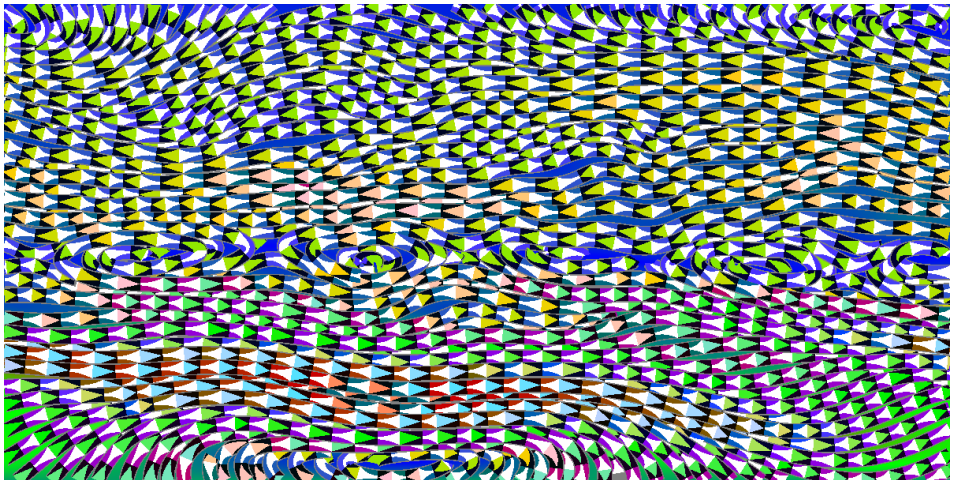


Figura 1.7: Visualização proposta por Chi *et al.* (Figura extraída do artigo original (6)).

Para a geração da ilusão, dado um campo vetorial como entrada, primeiramente são extraídas as linhas de fluxo. Posteriormente é feita a disposição de um PAR ao longo das linhas de fluxo gerando a ilusão de movimento em um campo vetorial estático. O objetivo deste trabalho consiste em melhorar essa geração usando a análise topológica do campo.

Pré-processamentos: Já na área de filtragem de campos vetoriais em grades regulares, parte dos trabalhos são especificamente dedicados no tratamento de imagens (20). Em particular, filtros para imagens coloridas focam na redução de ruído impulsivo (16, 25, 30). Mais recentemente, Westenberg e Erlt (35) propuseram um algoritmo para filtragem de campos vetoriais 2D que suprime ruído aditivo limitando os coeficientes do vetor de *wavelets*. Um conjunto de filtros para remoção de ruídos foi visto como uma generalização de *random walk*: em imagens (29), malhas (27, 28) e campos vetoriais (19) (ver Figura 1.8).

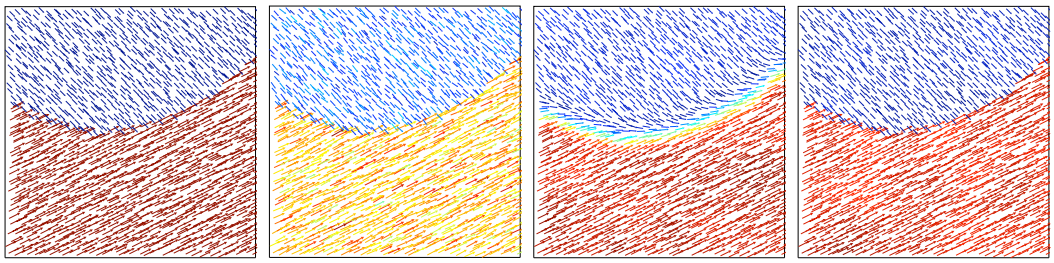


Figura 1.8: Esquerda: campo vetorial simples descontínuo; Meio à esquerda: pertubado com ruído Gaussiano; Meio à direita: filtro Gaussiano; Direita: *random walk* (Figura extraída do artigo original (19)).

As técnicas envolvendo espaços de escala se tornaram populares em Visão Computacional pela sua capacidade de representação de dado em multi-escala (ver Figura 1.9). Em particular, Bauer e Peikert (3) usam espaço de escala para rastrear vórtices em simulações 2D de dinâmica dos fluidos. Klein e Ertl (15) propuseram uma estratégia para rastrear singularidades em múltiplas escalas de forma a avaliar a importância dos pontos críticos na análise e interpretação do campo vetorial. O pré-processamento proposto no presente trabalho se baseia nesta linha de trabalho, mas aproveitando o conhecimento do usuário para escolher a escala localmente.

Figura 1.9: Espaço de escala.

2
Conceitos Básicos

Neste capítulo são apresentados alguns conceitos importantes e necessários para o desenvolvimento do trabalho. São apresentadas as definições de *campo vetorial*, *fluxo* e *linhas de fluxo*. Tratamos também de *pontos singulares* destacando como classificá-los. Apresentamos como os dados se comportam num *campo vetorial discreto* além de abordar um pouco da representação de topologia do campo vetorial por meio do seu *grafo topológico*.

2.1
Campo Vetorial e Fluxo

Os campos vetoriais foram originalmente usados em física para descrever movimentos num domínio, como por exemplo, a velocidade e direção de um fluido em um dado espaço. Nesta dissertação, nos restringiremos ao caso de campos vetoriais bidimensionais.

Definição 2.1 *Um* ***campo vetorial*** $\mathbf{v}$ *em um domínio planar* $D \in \mathbb{R}^2$ *é uma função que associa a cada ponto* $(x, y) \in D$ *a um vetor bidimensional*

$$\mathbf{v}(x, y) = (v^x(x, y), v^y(x, y)).$$

Assumindo que v^x e v^y são funções diferenciáveis nas duas variáveis, a matriz Jacobiana de $\mathbf{v}$ em um ponto $\boldsymbol{p} = (x_0, y_0)$ é dada por:

$$J_{\mathbf{v}}(x_0, y_0) = \begin{bmatrix} \frac{\partial v^x}{\partial x}(x_0, y_0) & \frac{\partial v^x}{\partial y}(x_0, y_0) \\ \frac{\partial v^y}{\partial x}(x_0, y_0) & \frac{\partial v^y}{\partial y}(x_0, y_0) \end{bmatrix}, \tag{2-1}$$

onde essa matriz expressa a aproximação linear local do campo.

Definição 2.2 *O campo vetorial* $\mathbf{v}$ *gera um* ***fluxo*** $\phi : U \subset \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$*, satisfazendo* $\dfrac{\partial}{\partial t}\phi(\boldsymbol{p}, t)\,|_{t=\tau} = \mathbf{v}(\phi(\boldsymbol{p}, \tau))$*. A partir desta função suave* ϕ*, definimos* $\phi_t : U_t \times \mathbb{R}^2 \to \mathbb{R}^2$ *por* $\phi_t(\boldsymbol{p}) = \phi(\boldsymbol{p}, t)$*.*

Propriedade 2.1.1 *Com a definição acima, ϕ_t satisfaz as propriedades:*

1. $\phi_0 = id$,
2. $\phi_t \circ \phi_s = \phi_{t+s}$, *em particular,* $\phi_t \circ \phi_{-t} = id$

Definição 2.3 *As **linhas de fluxo** de um campo vetorial* $\mathbf{v}$ *são as trajetórias seguidas por uma partícula cujo campo de velocidade é um campo vetorial dado. Os vetores do campo vetorial são tangentes as suas linhas de fluxo.*
Assim, a linha de fluxo $sl(x_0, y_0)$*, passando por* (x_0, y_0)*, é unicamente definida por:*

$$sl(x_0, y_0) = \bigcup_{t_0=0}^{\infty} \phi_t(x_0, y_0), \tag{2-2}$$

se $\mathbf{v}(x_0, y_0) \neq (0, 0)$.

Existem outras formas para a Definição 2.3, onde esta é dada por $sl(x_0, y_0) = \bigcup_{t_0=-\infty}^{\infty} \phi_t(x_0, y_0)$. Essa abordagem, porém, não será utilizada neste trabalho.

2.1.1 Pontos Singulares

Um ponto $(x_0, y_0) \in D$ é dito *singular* ou *crítico* em $\mathbf{v}$ se $\mathbf{v}(x_0, y_0) = (0, 0)$. Em Andronov *et al.* (1) e em Hirsch *et al.* (9) pode ser encontrada uma classificação completa para pontos singulares.
Observe que $\frac{\partial \phi}{\partial t}(x_0, y_0, t) = \mathbf{v}(x_0, y_0) = (0, 0)$. Portanto, $sl(x_0, y_0) = (x_0, y_0)$.

Através dos autovalores da matriz Jacobiana $J_{\mathbf{v}}$, ou dos vetores singulares caso $J_{\mathbf{v}}$ seja singular, podemos classificar os pontos singulares do campo vetorial como nos casos mostrados abaixo:

- **Caso 1:** Se J tem autovalores com parte real não nula e com sinais opostos, então a singularidade é chamada **ponto de sela** (ver Figura 2.1).
- **Caso 2:** Se J tem ambos autovalores com parte real estritamente negativa, então a singularidade é chamada **poço ou sorvedouro** (ver Figura 2.2).
  - **Caso 2.1:** Se J é diagonalizável e seus autovalores são diferentes, a singularidade é chamada **poço de nó** (ver Figura 2.2(a)). Caso os autovalores sejam iguais, então a singularidade é chamada de **poço focal** (ver Figura 2.2(b)).

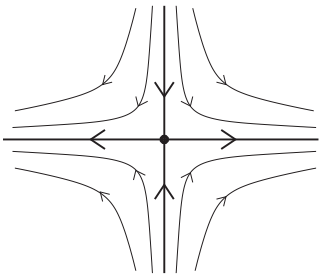
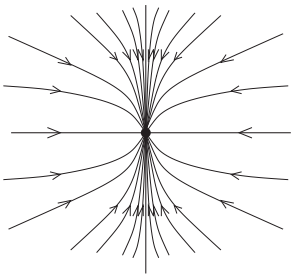
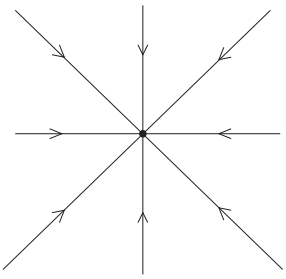


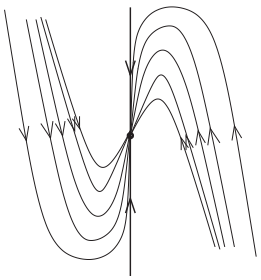
Figura 2.1: Sela



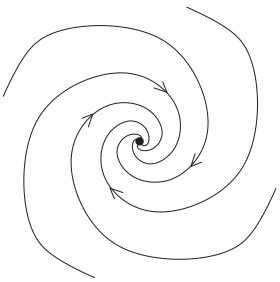
2.2(a): Poço de nó



2.2(b): Poço focal



2.2(c): Poço de nó impróprio



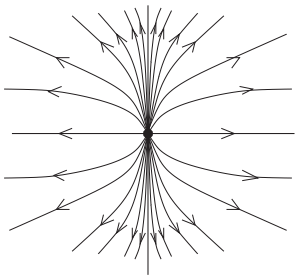
2.2(d): Poço espiral

Figura 2.2: Poço

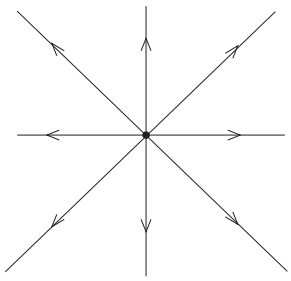
- **Caso 2.2:** Se J não é diagonalizável mas um autovalor tem parte real negativa, então a singularidade é chamada **poço de nó impróprio** (ver Figura 2.2(c)).
- **Caso 2.3:** Se J tem dois autovalores complexos conjugados com parte real negativa, então a singularidade é chamada **poço espiral** (ver Figura 2.2(d)).

– **Caso 3:** Se J tem ambos autovalores com parte real estritamente positiva, então a singularidade é chamada **fonte** (ver Figura 2.3).

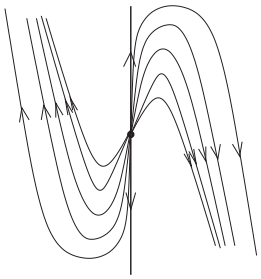
- **Caso 3.1:** Se J é diagonalizável e seus autovalores são diferentes, a singularidade é chamada **fonte de nó** (ver Figura 2.3(a)). Caso os autovalores sejam iguais, então a singularidade é chamada de **fonte focal** (ver Figura 2.3(b)).
- **Caso 3.2:** Se J não é diagonalizável mas um autovalor tem parte real é positiva, então a singularidade é chamada **fonte de nó impróprio** (ver Figura 2.3(c)).
- **Caso 3.3:** Se J tem dois autovalores complexos conjugados com parte real positiva, então a singularidade é chamada **fonte espiral** (ver Figura 2.3(d)).



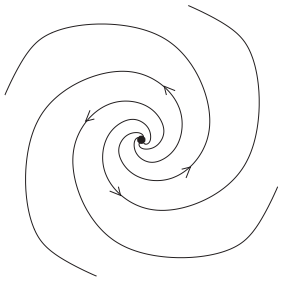
2.3(a): Fonte de nó



2.3(b): Fonte focal



2.3(c): Fonte de nó impróprio



2.3(d): Fonte espiral

Figura 2.3: Fonte

- **Caso 4:** Se J tem pelo menos um dos autovalores com a parte real nula, então a singularidade é de ordem superior (por exemplo, ver Figura 2.4).

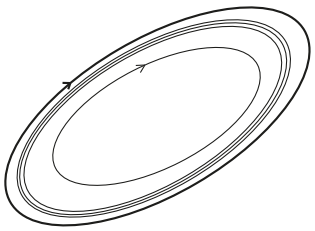


Figura 2.4: Órbita fechada

2.2 Grafo Topológico

Um fator interessante para auxiliar na compreensão de um campo vetorial é observar sua topologia por meio da construção do *grafo topológico*.

Definição 2.4 ***Separatrizes*** *são linhas de fluxo que dividem localmente o domínio de um campo vetorial em sub-domínios nos quais, dentro deles, todas as linhas de fluxo partem na vizinhança de uma única singularidade e convergem para uma mesma singularidade.*

Definição 2.5 *O* ***grafo topológico*** *de um campo vetorial planar* $\mathbf{v}$ *é constituído de todos os seus* ***pontos críticos*** *e suas* ***separatrizes****.*

Para o posicionamento das separatrizes em um campo sem singularidade de alta ordem, os pontos de sela são fundamentais, pois é a partir desses pontos e dos seus autovetores, que as separatrizes serão calculadas.

Uma questão interessante, e que requer atenção, ocorre quando estamos tratando de domínios com bordo. Nesses casos, as linhas de fluxo têm um comportamento que não depende exclusivamente das sigularidades, ele também depende das restrições impostas pelo bordo. Nesse caso, o bordo pode atuar localmente como um poço (as linhas de fluxo convergem para o bordo); como uma fonte (as linhas de fluxo partem do bordo) ou ainda, como uma sela (o bordo separa as linhas de fluxo e forma dois grupos). Em Scheuermann *et al.* (22) pode ser encontrada uma apresentação desse tópico do ponto de vista da visualização da topologia local de um campo vetorial. Um exemplo de um grafo topológico pode ser visto na Figura 2.5.

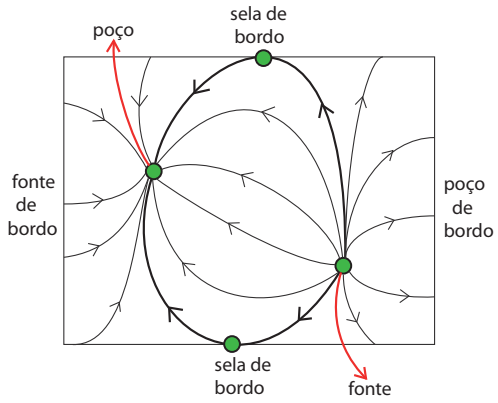


Figura 2.5: Grafo topolólgico

2.3 Campo Discreto

Em dados obtidos por medição, o campo vetorial $\mathbf{v}$ não é dado em forma de uma função diferenciável; esses dados se apresentam de forma discretizada.

Suporemos então que os valores de $\mathbf{v}$ estão dados nos pontos (x_i, y_i) de uma grade regular, como na Figura 2.6, de dimensão MxN. Usaremos a seguinte notação:

$$\mathbf{v}_{i,j} = (v^x_{i,j}, v^y_{i,j}) = \mathbf{v}(x_i, y_j),$$

para $i = 1, ..., M$ e $j = 1, ..., N$.

Quando for necessário obter os valores fora dos vértices da grade, interpolaremos os valores dos pontos amostrados. Uma interpolação simples, dentro de uma célula $[0, 1]^2$ da grade, consiste em uma interpolação bilinear como mostrado abaixo:

$$\begin{aligned} \mathbf{b}_{i,j} &: [0,1]^2 \to \mathbb{R}^2 \ , \\ \mathbf{b}_{i,j}(x,y) &= \mathbf{v}_{i,j} \cdot (1-x)(1-y) + \mathbf{v}_{i+1,j} \cdot x(1-y) \\ &\quad + \mathbf{v}_{i,j+1} \cdot (1-x)y + \mathbf{v}_{i+1,j+1} \cdot xy \quad . \end{aligned} \tag{2-3}$$

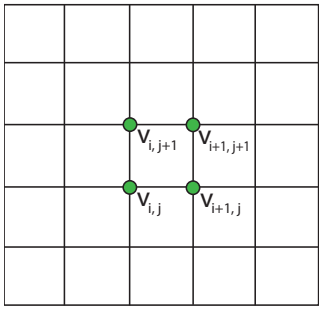


Figura 2.6: Grade regular

3
Técnicas Utilizadas

Neste capítulo são apresentadas as técnicas existentes utilizadas no desenvolvimento do trabalho. Abordamos alguns métodos para *detecção de singularidades* e a forma como aplicá-los no campo discreto. Apresentamos também como gerar o *espaço de escala*, as *linha de fluxo* e discutimos como gerar a visualização do campo vetorial com as *imagens auto-animadas*.

3.1
Detecção de Singularidades

Nós utilizamos duas abordagens clássicas para a detecção de singularidades em grades regulares bidimensionais. A primeira consiste em detectar onde a interpolação bilinear do campo se anula. A segunda consiste em computar o *winding numbers* da mesma interpolação bilinear. Propomos também um método para a detecção de *regiões fracas* do campo vetorial, que são as regiões onde a interpolação bilinear está *próxima* de zero. Para controlar essa *proximidade*, nós permitimos ao usuário impor a margem de precisão desejada.

3.1.1
Singularidades da Interpolação Bilinear

Para encontrarmos as singularidades no caso da interpolação bilinear, precisamos determinar se o vetor zero é atingido dentro das células quando é feita a interpolação (ver Figura 3.1).

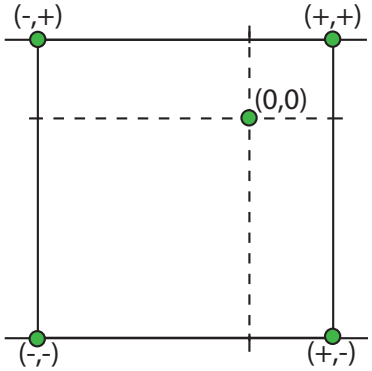


Figura 3.1: Interpolação bilinear.

Ou seja, queremos resolver o sistema de equações quadrático $\mathbf{b}_{i,j} = (0,0)$, onde $\mathbf{b}_{i,j}$ é definido como na equação (2-3). Isso pode ser resolvido explicitamente encontrando as raízes do polinômio em y:

$$\begin{aligned} & (\quad -v_{01}^x\, v_{00}^y + v_{01}^x\, v_{10}^y + v_{11}^x\, v_{00}^y - v_{11}^x\, v_{10}^y + \\ & \quad +v_{00}^x\, v_{01}^y - v_{00}^x\, v_{11}^y - v_{10}^x\, v_{01}^y + v_{10}^x\, v_{11}^y \quad) \cdot y^2 \\ + \; & (\quad 2\, v_{01}^x\, v_{00}^y - 2\, v_{00}^x\, v_{01}^y - v_{11}^x\, v_{00}^y - \\ & \quad -v_{01}^x\, v_{10}^y + v_{10}^x\, v_{01}^y + v_{00}^x\, v_{11}^y \quad) \cdot y \\ + \; & \quad v_{00}^x\, v_{01}^y - v_{01}^x\, v_{00}^y \quad ; \end{aligned}$$

Buscando simplificar a notação, estamos particularizando para uma célula com vértices $(i,j) = (0,0)$, $(i+1,j) = (1,0)$, $(i,j+1) = (0,1)$ e $(i+1,j+1) = (1,1)$.

Para obter o valor da coordenada x do ponto singular, podemos usar a seguinte expressão:

$$x = \frac{(v_{00}^y - v_{01}^y) \cdot y - v_{00}^y}{(v_{00}^y - v_{10}^y - v_{01}^y + v_{11}^y) \cdot y - v_{00}^y + v_{10}^y} \; .$$

Eventualmente, esse sistema pode se degenerar em um polinômio de grau mais baixo, ou seja, o sistema pode ter duas, uma ou nenhuma solução. As soluções precisam ser testadas para garantir que elas estejam dentro do quadrilátero.

3.1.2 Winding Numbers

O *winding number* ou *índice* de uma curva plana fechada e parametrizada Γ em torno de um ponto fora da curva é um número inteiro que representa o número de voltas dadas pela curva em torno do ponto p (ver Figura 3.2).

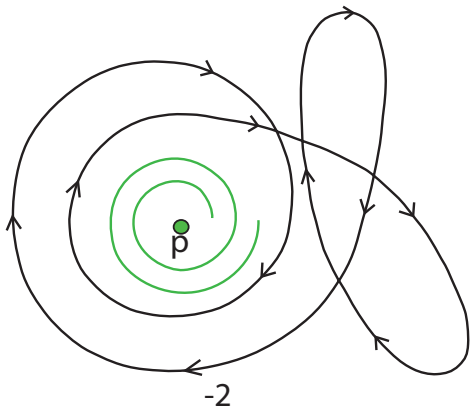


Figura 3.2: Curva com índice -2 em volta do ponto p.

O número de voltas depende da orientação da curva. As curvas em sentido anti-horário têm valor positivo (ver Figura 3.3) e as curvas em sentido horário têm valor negativo (ver Figura 3.4). Assim, se a curva em torno de p é percorrida descrevendo uma volta em sentido horário e uma volta em sentido anti-horário, o índice será 0.

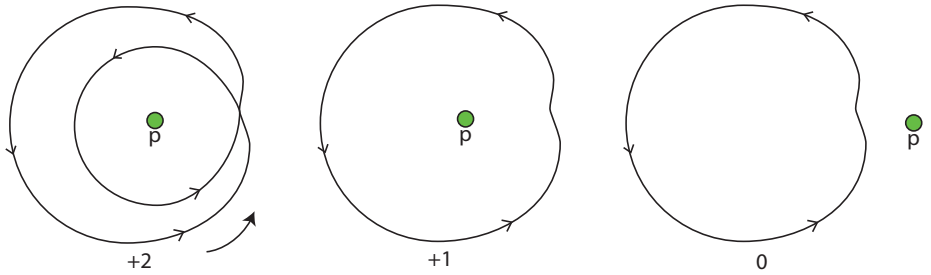


Figura 3.3: Curva no sentido anti-horário.

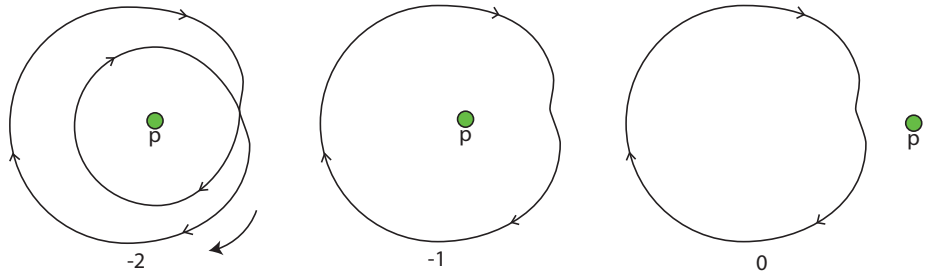


Figura 3.4: Curva no sentido horário.

O índice de um campo vetorial $\mathbf{v}$ numa região delimitada pela curva fechada $\Gamma = \{\gamma(t), t \in [a,b]\}$ é o índice da curva $\Gamma_{\mathbf{v}} = \{\gamma(t) + \varepsilon\mathbf{v}(\gamma(t)), t \in [a,b]\}$, para ϵ pequeno, em volta de um ponto p no interior da região. Ele pode também ser calculado a partir da componente angular $\theta(p)$ do campo vetorial $\theta(p) = arctan\left(\dfrac{\mathbf{v}^y(p)}{\mathbf{v}^x(p)}\right)$, por:

$$w_\Gamma(\mathbf{v}) = \frac{1}{2\pi}\oint_{\Gamma_{\mathbf{v}}} \theta d\Gamma_{\mathbf{v}} \tag{3-1}$$

Esse índice é zero se a região dentro de Γ não contém nenhum ponto crítico. Se em Γ existe um ponto de sela, então $w_\Gamma(\mathbf{v}) = -1$ e se contém um poço ou uma fonte, então $w_\Gamma(\mathbf{v}) = +1$.

Na grade regular, calculamos o índice para cada célula usando a curva Γ como o quadrado que contorna a célula. Utilizando uma interpolação linear nas arestas, podemos calcular explicitamente a contribuição da aresta $(x_0, y_0) \rightarrow (x_1, y_0)$ na integral (3-1) da seguinte forma:

$$
\begin{aligned}
w_{00\to 10} = \arctan & \left(\frac{{v_{00}^x}^2 - v_{00}^x\, v_{10}^x - v_{00}^y\, v_{10}^y + {v_{00}^y}^2}{v_{10}^y\, v_{00}^x - v_{00}^y\, v_{10}^x} \right) \\
- \arctan & \left(\frac{v_{00}^x\, v_{10}^x - {v_{10}^x}^2 + v_{00}^y\, v_{10}^y - {v_{10}^y}^2}{v_{10}^y\, v_{00}^x - v_{00}^y\, v_{10}^x} \right).
\end{aligned}
\tag{3-2}
$$

Repetimos esta conta para as outras arestas de forma a obter $w = w_{00\to 10} + w_{10\to 20} + ...$

3.1.3 Regiões Fracas

Agora proporemos um método para detectar regiões fracas. Por regiões fracas denominamos as regiões onde a interpolação bilinear está *próxima* de zero. Assim, ao invés de procurarmos os pontos onde o campo vetorial se anula, iremos procurar pontos dentro de um intervalo, controlado por um parâmetro ε, para encontrarmos pontos que podem ser singularidades com uma pequena pertubação (ver Figura 3.5). Procuraremos então:

$$(i, j) \text{ tal que } \min\|\mathbf{b}_{i,j}\| \leq \varepsilon,$$

onde ε é uma margem de tolerância especificada pelo usuário.

Esse problema se resume a encontrar as raízes de um sistema polinomial de terceiro grau em duas variáveis, reduzindo a um sistema de quinto grau em uma variável. Nesta dissertação, utilizamos métodos numéricos para obter a solução.

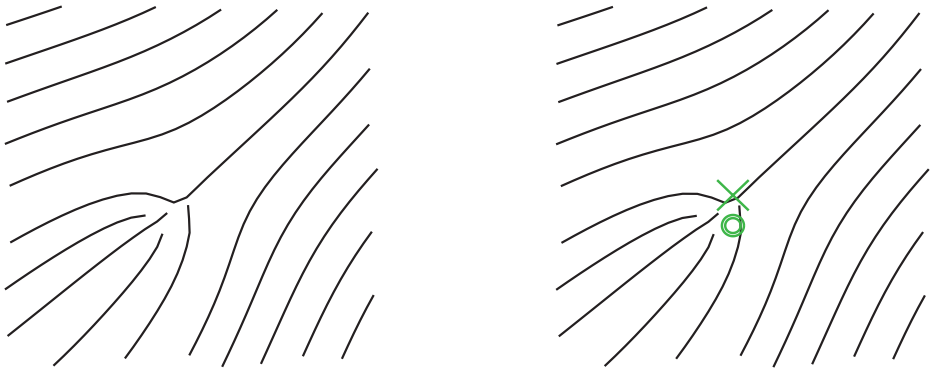


Figura 3.5: Detecção de singularidades. Esquerda: bilinear. Direita: regiões fracas.

3.1.4
Classificação das Singularidades

Como dito na seção 2.1.1, podemos classificar as singularidades através da matriz Jacobiana aplicada àquele ponto. Explicitamente, a matriz Jacobiana da interpolação bilinear de $\mathbf{b}_{0,0}$ é dada por:

$$\begin{bmatrix} v_{11}^x\, y - v_{00}^x\, \bar{y} + v_{10}^x\, \bar{y} - v_{01}^x\, y \;;\; v_{11}^x\, x - v_{00}^x\, \bar{x} - v_{10}^x\, x + v_{01}^x\, \bar{x} \\ v_{11}^y\, y - v_{00}^y\, \bar{y} + v_{10}^y\, \bar{y} - v_{01}^y\, y \;;\; v_{11}^y\, x - v_{00}^y\, \bar{x} - v_{10}^y\, x + v_{01}^y\, \bar{x} \end{bmatrix},$$

onde $\bar{x} = 1 - x$ e $\bar{y} = 1 - y$.

Os autovalores são diretamente calculados utilizando o traço e o determinante da matriz. Lembrando que, de forma geral,

- Se a parte real de ambos autovalores é estritamente negativa, então o ponto singular é um *poço*.
- Se a parte real de ambos autovalores é estritamente positiva, então o ponto singular é uma *fonte*.
- Se os autovalores têm partes reais não nula e têm sinais opostos, então o ponto singular é uma *sela*.
- Se a parte real de um dos autovalores é nula, então a singularidade é de ordem superior.

3.2
Espaço de Escala

Espaço de escala é um conjunto formado por varias versões de um mesmo objeto em escalas diferentes desenvolvido pelas comunidades de Visão Computacional, Processamento de Imagem e Processamento de Sinais.

Em imagens, o espaço de escala é uma família de imagens suavizadas, com a mesma dimensão da imagem original e parametrizadas pela quantidade de suavização aplicada (ver Figura 3.6).

A representação de campos vetoriais por um espaço de escala nada mais é que uma coleção de versões filtradas progressivamente do campo (ver Figura 3.7). Cada versão está associada a um parâmetro de escala s crescente. Denotamos por $\bar{\mathbf{v}}(s, x, y)$ o vetor de valores do campo na escala s e no ponto (x, y).

O exemplo mais comum de espaço de escala num campo vetorial contínuo é o espaço de escala Gaussiano. Ele é obtido por uma convolução do campo com um núcleo Gaussiano de variância crescente:

Figura 3.6: Espaço de escala em imagens.

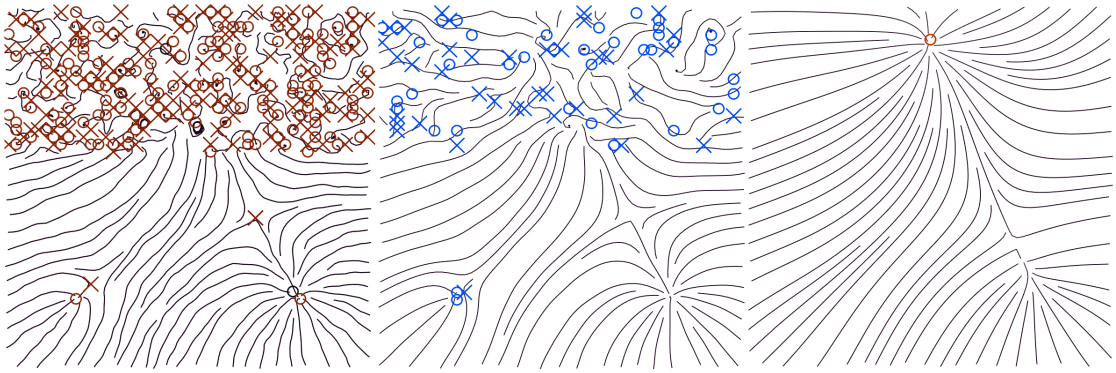


Figura 3.7: Espaço de escala em campos vetoriais.

$$\bar{\mathbf{v}}(s,x,y) = \mathbf{v}(x,y) * G_s(x,y), \text{ com } G_\sigma(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma}\right).$$

Em dados discretizados, a abordagem da convolução se encaixa em técnicas como o *random walks* (26), que asseguram boas propriedades para máscaras de convolução local. O número de convoluções aplicadas, será então o parâmetro de escala que usaremos.

3.3 Geração das Linhas de Fluxo

Para a geração das linhas de fluxo, utilizamos o algoritmo proposto por Jobard e Lefer (10) para a criação de linhas de fluxo igualmente espaçadas com densidade arbitrária com uma pequena modificação na escolha da primeira semente.

O algoritmo pode ser descrito da seguinte forma: primeiramente escolhemos uma semente, de forma aleatória, no campo vetorial de entrada e construímos a primeira linha de fluxo. Queremos que as linhas de fluxo estejam igualmente espaçadas, para isso, impomos uma distância d_{sep} que é a distância mínima entre duas linhas de fluxo vizinhas.

Durante a construção da linha de fluxo, dispomos pontos com distância

d_{sep} de ambos os lados da linha de fluxo e esses pontos são colocados em uma pilha como candidatos à nova semente.

As próximas linhas de fluxo são construídas da seguinte forma: tomamos o primeiro ponto da pilha. Se ele for um ponto válido, ou seja, se a distância de separação é maior que d_{sep}, então ele é a nova semente. A nova linha de fluxo é então integrada para trás e para frente de forma independente.

Durante a construção, o novo ponto é considerado válido, se e somente se, a distância para as outras linhas de fluxo é menor que d_{sep} e se estiver dentro do domínio. Se não for, a integração pára nesse sentido.

O algoritmo continua até que haja a saturação do domínio, ou seja, até que a pilha esteja vazia e não possa ser adicionada nenhuma nova semente.

A Figura 3.8, mostra todas as linhas de fluxo que derivaram da primeira linha de fluxo do campo vetorial.

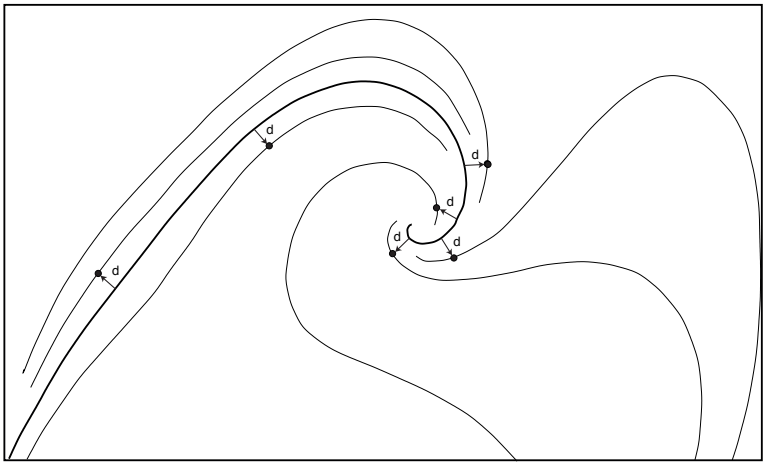


Figura 3.8: Linhas de fluxo derivadas da primeira linha de fluxo criada no campo (mais escura) (Figura extraída do artigo original (10)).

Para a integração da linha de fluxo, podemos usar vários tipos de integradores. Neste trabalho, utilizamos o método de *Runge-Kutta* de segunda ordem.

A escolha das primeiras sementes, porém, não é dada de forma aleatória. Escolhemos as sementes de forma que as primeiras linhas de fluxo a serem construídas partam das singularidades do campo vetorial. Como a singularidade é um ponto fixo, geramos sementes próximas da singularidade nas direções dos autovetores de J.

Apenas quando esgotada a possibilidade de construção partindo de singularidades é que faremos a construção das demais linhas de fluxo. Vale observar que, neste ponto, ainda não estamos considerando as singularidades de bordo. Na Figura 3.9 podem ser observadas as linhas de fluxo geradas com essa metodologia.

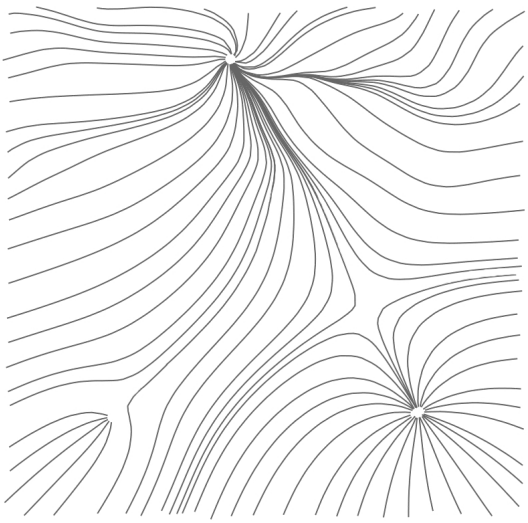


Figura 3.9: Linhas de fluxo.

3.4 Imagens Auto-Animadas e Otimização

Nessa seção descreveremos parte do trabalho proposto por Chi *et al.* (6), que foi o principal motivador para o desenvolvimento do nosso trabalho.

Este trabalho é baseado na classificação da otimização da ilusão de Fraser-Wilcox proposta por Kitaoka (13) como mostrado na Figura 3.10.

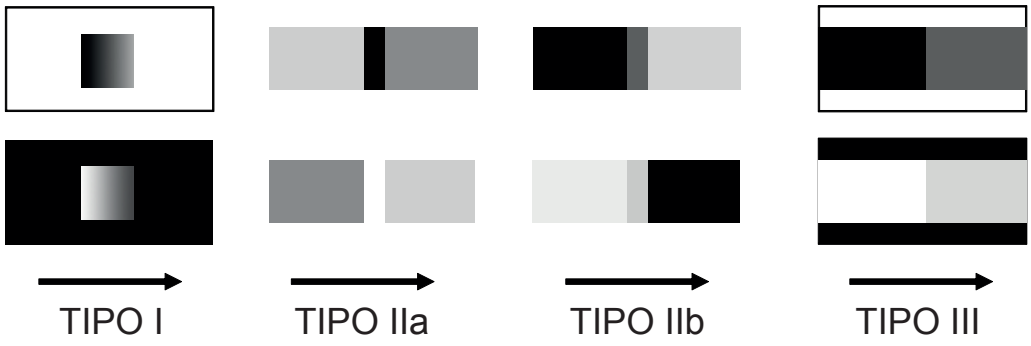


Figura 3.10: Tipos da otimização da ilusão de Fraser-Wilcox. As setas indicam a direção do movimento percebido (Figura extraída do artigo original (6)).

Será focado apenas o Tipo IIa. A combinação das quatro intensidades desse tipo está na ordem: P-CE-B-CC (preto, cinza escuro, branco, cinza claro). Cada padrão de quatro intensidades é também chamado de padrão assimétrico repetido (PAR) (Backus *et al.* (2)). O padrão de cor usado será: preto-azul-branco-amarelo.

Em seu trabalho, Chi *et al.* (6) propõem uma nova abordagem computacional para gerar a ilusão de Fraser-Wilcox usando o posicionamento de PAR. Dado um campo vetorial, os PARs são posicionados ao longo do fluxo

para gerar linhas de fluxo com movimento consistente com o campo vetorial. Além disso, é feita uma otimização do posicionamento dos PARs de forma a aumentar o efeito da ilusão.

3.4.1 Posicionamento das Linhas de Fluxo

Uma idéia simples para a construção das linhas de fluxo consiste em escolher uma semente aleatória no campo, e a partir desta, integrar a linha de fluxo na qual os PARs serão dispostos. Porém, nem sempre esse processo é favorável à geração da ilusão pois podem ser criadas linhas curtas e irregulares. Em Chi *et al.* (6), foi adotado o método proposto por Mebarki *et al.* (17) para gerar linhas de fluxo longas e igualmente espaçadas o que resulta em uma melhor qualidade de ilusão de movimento, após a disposição dos PARs.

Feito isso, cada linha é então parametrizada de forma a aplicar a textura de PAR. As linhas de fluxo são divididas em segmentos de mesmo tamanho, chamado *SegLen*. O tamanho de cada cor no PAR segue a sequência 1:2:1:2, de forma a satisfazer os requerimentos do TipoII da otimização da ilusão de Fraser-Wilcox (13). Previamente, é criado um vetor *corPar* = {*preto, azul, azul, branco, amarelo, amarelo*}. Cada linha de fluxo é colorida da seguinte forma:

$$Cor = corPAR[(i \bmod SegLen) \times (6/SegLen)]$$

onde i é a distância para o ponto inicial da linha de fluxo.

3.4.2 Otimização do Posicionamento dos Fragmentos

O padrão em volta do segmento PAR também afeta o efeito da ilusão. A Figura 3.11 mostra a importância do posicionamento do fragmento, pois, se feito de forma arbitrária em linhas de fluxo adjacentes (ver Figura 3.11(a)), podem ser gerados blocos que prejudicam a ilusão do movimento, contrastando com a Figura 3.11(d), onde o posicionamento dos fragmentos foi feito de forma mais adequada.

Para melhorar o posicionamento dos PARs, Chi *et al.* (6) desenvolveram o seguinte problema de otimização que tentamos melhorar neste trabalho. Um segmento de PAR pode ser dividido em fragmentos claro e escuro, usando tons de cinza, por exemplo. Seja X o conjunto dos fragmentos de todos os PARs, $N(x)$ o conjunto dos fragmentos vizinhos de x em X. $L(z)$ é a intensidade do fragmento z.

$$E_{fragmento} = \sum_{x \in X} \sum_{y \in N(x)} (L(x) - L(y))^2. \quad (3\text{-}3)$$

O objetivo da otimização é maximizar $E_{fragmento}$, isto é, a diferença de intensidade entre fragmentos de PARs vizinhos entre linhas de fluxo vizinhas. Em Chi *et al.* (6) é proposto um método força bruta baseado em imagem para medir a diferença de posicionamento entre duas linhas de fluxo vizinhas de forma a maximizar a equação (3-3).

Para isso, uma imagem guia é primeiramente renderizada. A linha superior da Figura 3.11 mostra como a imagem guia deriva do padrão inicial. Como simplificação, tomaremos duas linhas de fluxo e sua imagem guia como exemplo, Figuras 3.11(b) e 3.11(c), respectivamente. Primeiro, todos os segmentos começam com o mesmo tamanho. Em seguida, a parte clara do segmento de PAR é colorida de vermelho-médio e a parte escura é colorida de verde-médio. Assim, cada segmento de PAR contém um par de fragmento vermelho-médio e verde-médio. A largura da linha de fluxo pode ser determinado de forma que os PARs de linhas vizinhas toquem ou sobreponham uns aos outros. As linhas vermelha-verde são então desenhadas uma a uma sendo as regiões de interseção misturadas. Elas são misturadas de forma que as regiões de mesma cor que se sobrepõem obtenham uma cor mais intensa. Assim, regiões com vermelho ou verde claro, indicam posicionamento de baixa qualidade já que a intensidade nas linhas de fluxo vizinhas eram parecidas, enquanto as regiões na cor amarela, indicam grande diferença de intensidade, e portanto, melhor posicionamento. Então, quanto mais pixels amarelos forem obtidos, mais otimizado está o posicionamento.

Com a imagem guia, resolver o problema de otimização da equação (3-3) se resume a maximizar o número de *pixels* amarelos. Os fragmentos são posicionados nas linhas de fluxo, seguindo alternadamente a ordem escuro–para–claro. Durante a otimização, o posicionamento é ajustado em dois parâmetros: a cor inicial do fragmento - escuro ou claro e o tamanho de cada fragmento. Para evitar que os fragmentos sejam muito pequenos ou muito grandes, eles são restringidos ao intervalo $[0.4(SegLen), 0.65(SegLen)]$. O tamanho do fragmento é inicializado em $0.5(SegLen)$. Randomicamente, uma linha de fluxo e sua vizinha são selecionadas e o objetivo é diferenciar as cores entre dois fragmentos vizinhos dessas linhas de fluxo. Como mostrado na Figura 3.11(e), o fragmento que inicia a linha de fluxo selecionada é ajustado para ser um fragmento claro (branco e amarelo, por exemplo), de acordo com a linha vizinha referenciada, que tem fragmento inicial escuro (preto e azul). O tamanho desse fragmento também é atualizado de forma a diferenciar a cor com seu vizinho.

Esses ajustes são aplicados a todas as outras linhas de fluxo de forma a aumentar a quantidade de regiões amarelas na imagem guia 3.11(f). As Figuras 3.11(d) e 3.11(e) mostram os resultados depois da otimização.

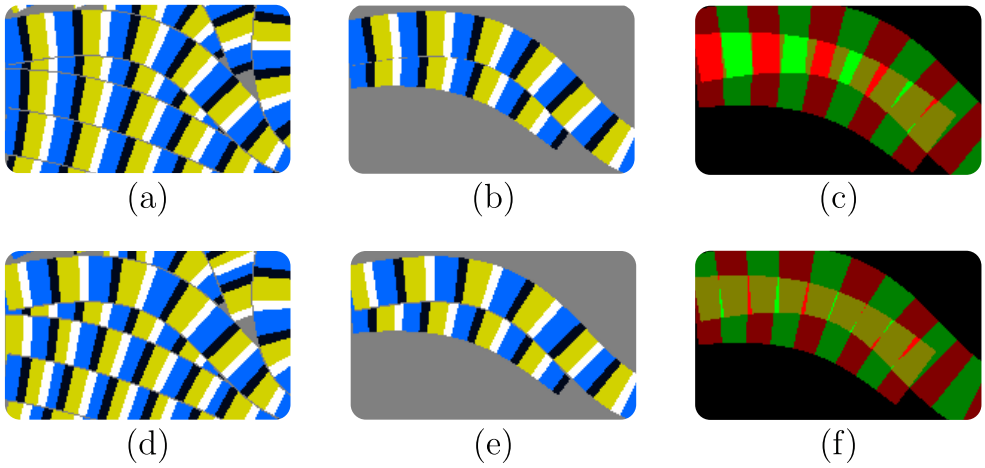


Figura 3.11: Otimização do fragmento. Linha superior: antes da otimização. Linha inferior: depois da otimização (Figura extraída do artigo original (6)).

Por fim, as linhas são separadas por uma borda cinza, pois sem essa borda, PARs vizinhos podem acidentalmente se misturar mudando o padrão.

4
Técnicas Desenvolvidas

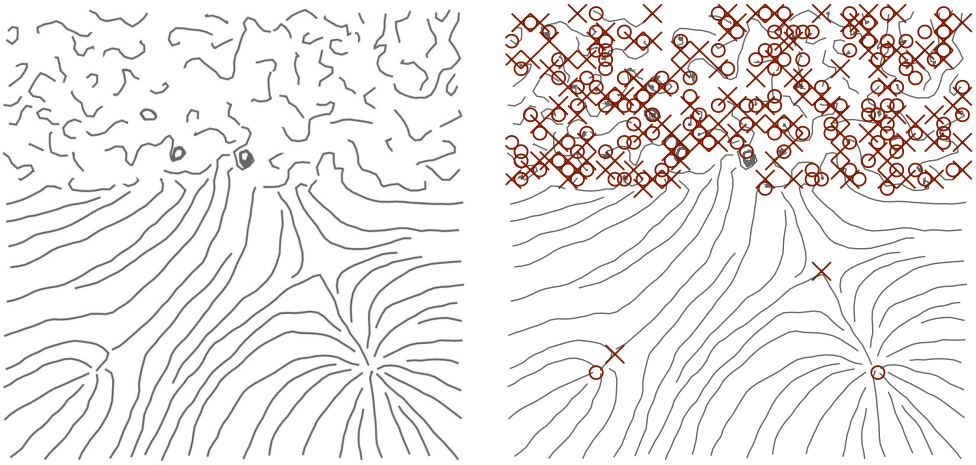
Neste capítulo descrevemos as técnicas desenvolvidas neste trabalho. Apresentamos uma inteface para a *edição* de campos vetoriais (18) e como construir a *segmentação* deste baseado em sua topologia. Além disso, mostraremos como essas etapas auxiliam na visualização por imagens auto-animadas.

4.1
Edição de Campos Vetoriais - Remoção de Ruidos

Vários fatores podem interferir na visualização dos campos vetoriais. Dados reais estão sujeitos a variações causadas por ruídos devido principalmente às imprecisões provenientes das medições.

Apresentamos uma metodologia que consiste em permitir ao usuário selecionar localmente uma escala de ruído para remoção, definindo um parâmetro de escala $s(x, y)$ em cada ponto (18).

A Figura 4.1 ilustra a técnica que estamos propondo. Construímos um campo vetorial artificial e adicionamos ruído não gaussiano em escalas diferentes, sendo em maior escala na parte superior e menor escala na parte inferior. As singularidades da parte inferior do campo devem ser mantidas, já as da parte superior, com exceção de um poço, devem ser removidas.



4.1(a): Campo vetorial artificial

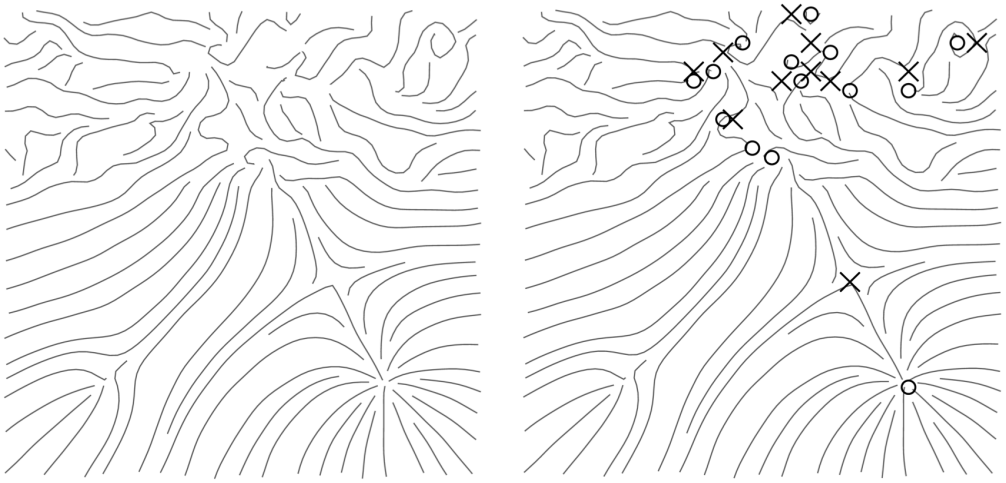
4.1(b): Detecção de singularidades

Figura 4.1: Campo vetorial artificial representado por linhas de fluxo.

A primeira etapa, consiste na geração do espaço de escala a partir do campo original, onde o usuário escolhe uma escala central s_0 (ver Figura 4.2). Para evitar o custo de definir o parâmetro de escala $s(x, y)$ em todos os pontos amostrados no campo, apresentamos para o usuário as singularidades que aparecem e/ou desaparecem em escalas vizinhas de s_0 de acordo com um parâmetro definido pelo usuário. Este parâmetro pode ser a diferença entre escalas ou a quantidade de mudanças topológicas. Neste trabalho realizamos a filtragem por *random walk* com núcleo Gaussiano G_α e com o núcleo anisotrópico $A_{\sigma,\tau}$ (3, 19):

$$A_{\sigma,\tau}(x, y, \mathbf{v}) = \exp\left(-\frac{x^2 + y^2}{2\sigma}\right) \exp\left(-\frac{||\mathbf{v}||^2}{2\tau}\right)$$

Ambos núcleos representam naturalmente o dado na forma hierárquica do espaço de escala.



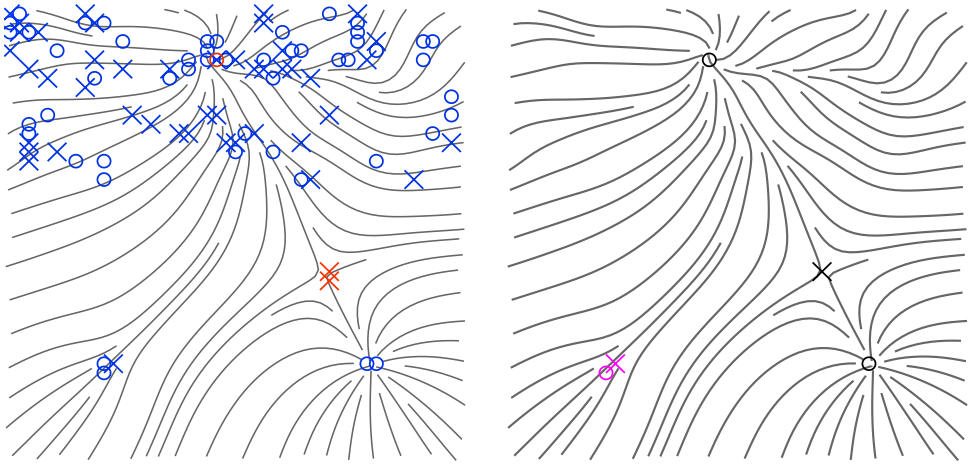
4.2(a): Escala s = 10

4.2(b): Detecção de singularidades

Figura 4.2: Campo vetorial artificial na escala s = 10 do espaço de escala Gaussiano.

Em posse do espaço de escala, o usuário pode decidir quais mudanças topológicas são relevantes e desejadas para o campo em questão (ver Figura 4.3). Quando uma mudança topológica em um ponto singular (x_0, y_0) é selecionada, definimos $s(x_0, y_0)$ para ser a escala mais próxima de s_0 que reverte a mudança. Em outras palavras, utilizamos o espaço de escala para permitir que o usuário decida que escala utilizar *localmente* na reconstrução do campo.

A última etapa, consiste na reconstrução do campo vetorial que nada mais é que uma versão suave do campo vetorial resultante da inteporlação das escalas do espaço de escala (ver Figura 4.4(a)). As singularidades selecionadas pelo usuário fornecem amostras da função de escala $s(x, y)$ no domínio. Para



4.3(a): Espaço de escala

4.3(b): Mudanças à reverter

Figura 4.3: A interface mostra para o usuário as mudanças topológicas em escalas próximas, aqui da 5 à 15 4.3(a) onde o usuário seleciona que mudanças ele deseja reverter (em roxo) 4.3(b).

reconstruir campo vetorial, precisamos interpolar essas amostras. Denotando por $\bar{\mathbf{v}}_{i,j}(s)$ as amostras do campo vetorial na escala s, definimos o campo vetorial reconstruído $\tilde{\mathbf{v}}$ no ponto da grade (x_i, y_j) por:

$$\tilde{\mathbf{v}}_{i,j} = \bar{\mathbf{v}}_{i,j}(s(x_i, y_j)) \ .$$

Qualquer método de interpolação pode ser utilizado, porém mudanças bruscas no parâmetro de escala podem comprometer a qualidade do resultado. Nesse trabalho, utilizamos dois métodos para a interpolação: funções de base radial (RBF) com base Gaussiana, e o método de interpolação de Shepard com núcleo Gaussiano (4). Ambos apresentaram resultado satisfatório (ver Figura 4.4(b)).

A interpolação com RBF de $s(x, y)$ das escalas provenientes das singularidades selecionadas pelo usuário s_k em (x_k, y_k) é obtida por uma minimização por mínimos quadrados nos coeficientes α_k de

$$\min_{\{\alpha_k\}} \sum_k \|s_{rbf}(x_k, y_k) - s_k\|^2 \ , \quad onde \tag{4-1}$$

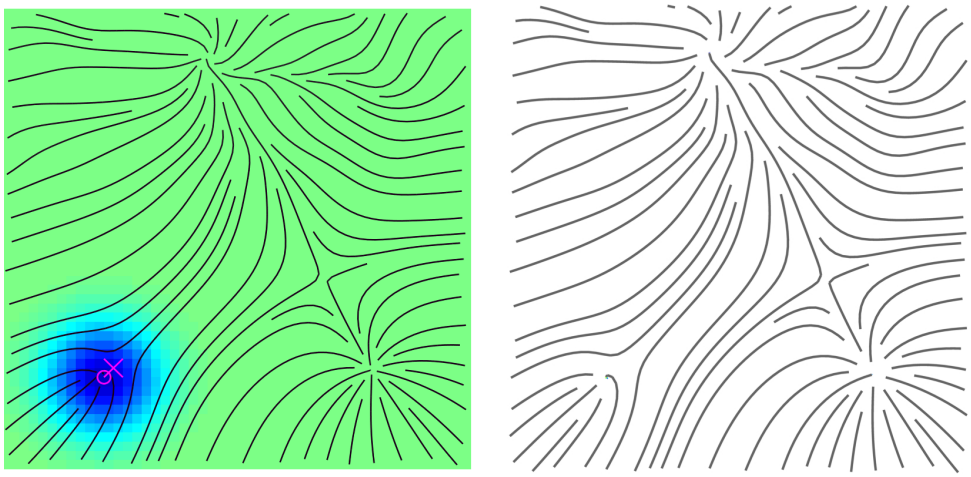
$$s_{rbf}(x, y) = \sum_k \alpha_k G_\sigma \left(x - x_k, y - y_k\right) \ . \tag{4-2}$$

O método de Shepard com núcleos (4) modifica a interpolação original

de Shepard (24) substituindo a distância Euclidiana por um núcleo:

$$s_{ks}(x,y) = \frac{1}{\sum_k G_\sigma(x-x_k, y-y_k)} \cdot \sum_k G_\sigma(x-x_k, y-y_k) \cdot s_k \ .$$

Uma propriedade importante nesse método é que a imagem fica limitada por $[min_k s_k, max_k s_k]$.



4.4(a): Função interpoladora 4.4(b): Campo vetorial reconstruído

Figura 4.4: Interpolação das escalas indicadas pelo usuário por uma função suave 4.4(a) que define a reconstrução do campo vetorial 4.4(b).

4.2 Segmentação do Campo Vetorial

Após a edição do campo vetorial, podemos fazer a segmentação do mesmo. A segmentação consiste em separarmos o domínio em grupos, de forma que linhas de fluxo com comportamento semelhante fiquem em um único grupo. Dizemos que as linhas de fluxo s_1 e s_2 têm o *comportamento semelhante* se s_1 e s_2 se originam em uma mesma singularidade e convergem para uma mesma singularidade.

Assim, após a detecção das singularidades e a construção das linhas de fluxo, resta detectar como o domínio pode ser particionado. Embora tenhamos adotado um método para a geração de linhas de fluxo igualmente espaçadas, nesta etapa, as estendemos até alcançarem o bordo ou até alcançarem uma singularidade. Para cada linha de fluxo então, associamos uma singularidade ao começo da linha e outra ao fim, podendo estas serem bordo ou não. Com isso, geramos um particionamento inicial.

Num primeiro tempo, poderíamos tratar o bordo inteiro como uma única singularidade. Na Figura 4.5, por exemplo, a segmentação ocorreu de maneira

natural, resultando em três grupos distintos: o primeiro formado pelo bordo, pela singularidade 1 e pelas linhas de fluxo que os conectam (cor azul); o segundo formado pelas singularidades 1 e 2 e pelas linhas de fluxo que as conectam (cor laranja); e um terceiro formado pela singularidade 2 e pelo bordo e pelas linhas de fluxo que os conectam (cor roxa).

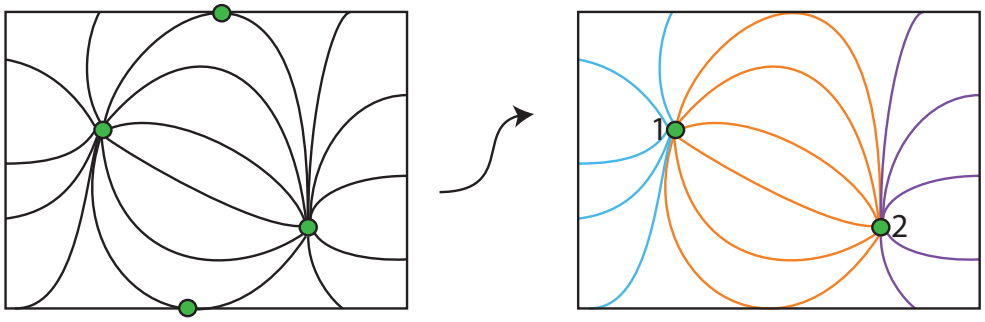
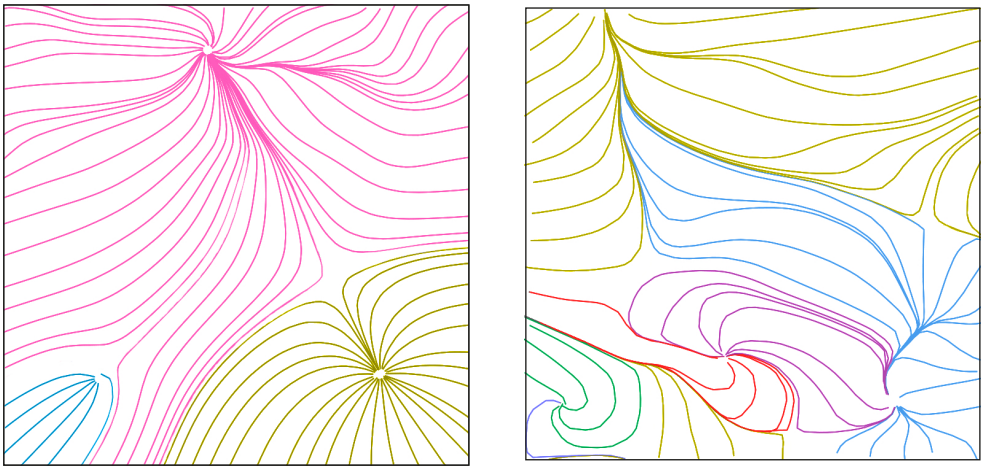


Figura 4.5: Segmentação do campo vetorial tratando o bordo como uma única singularidade.

Mas, nem sempre essa segmentação ocorre de forma tão natural. Seguindo a metodologia de tratar o bordo como uma única singularidade, não conseguimos segmentar, por exemplo, os campos da Figura 4.6 de maneira adequada, em particular, a separatriz saindo no canto abaixo a esquerda da Figura 4.6(a) não foi representada corretamente.



4.6(a): Campo segmentado em três grupos

4.6(b): Campo segmentado em seis grupos

Figura 4.6: Exemplos de campos segmentados tratando o bordo como uma única singularidade.

Em testes realizados neste trabalho, constatamos que nem sempre podemos utilizar o critério das separatrizes como solução. Utilizando a técnica de gerar linhas de fluxo igualmente espaçadas, a semente que irá gerar a separa-

triz nem sempre é válida, ou seja, a validade da semente ficará condicionada à densidade das linhas de fluxo no campo.

Precisamos então realizar um tratamento no bordo de forma que a segmentação ocorra de maneira adequada. O tratamento é realizado em duas etapas.

Como nossos dados estão armazenados em uma grade regular, podemos associar a cada célula de bordo, as linhas de fluxo incidentes e seu respectivo grupo. Tomemos a Figura 4.7 como exemplo para explicarmos como é feita a primeira parte do tratamento de bordo: o bordo desse campo ficará de acordo com a Figura 4.7(centro):

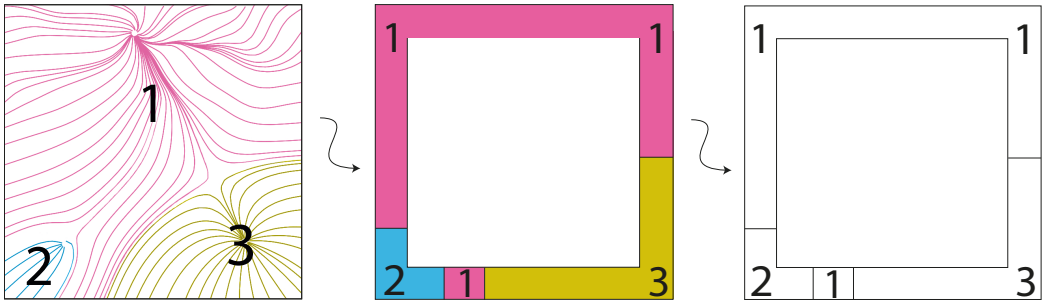


Figura 4.7: Esquema de campos segmentados tratando o bordo como uma única singularidade.

O que precisamos fazer na primeira parte é detectar quando algum grupo se divide em diferentes componentes conexas. Caso isso aconteça, particionamos o bordo criando uma nova singularidade por componente conexa (ver Figura 4.8).

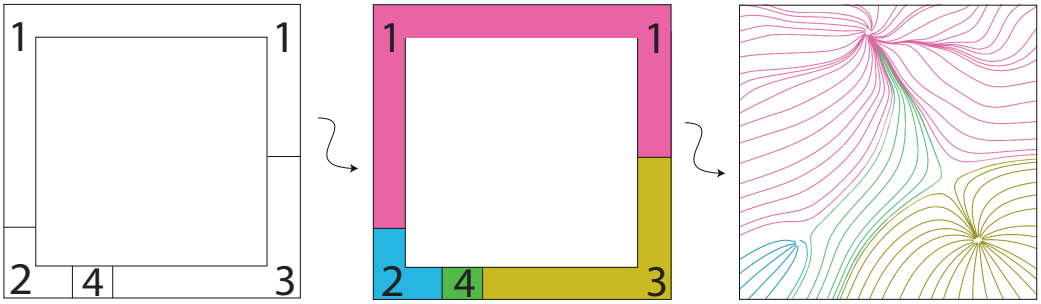


Figura 4.8: Esquema da primeira parte do tratamento de bordo.

A segunda parte do tratamento de bordo ocorre para o caso dos grupos onde as linhas de fluxo se iniciam e terminam no bordo, como ocorre na Figura 4.6(b). Aqui temos um caso onde podemos generalizar o tratamento de bordo.

Na Figura 4.9 observamos como seria a segmentação adequada realizada pelas separatrizes e na Figura 4.10, observamos a segmentação utilizando a primeira parte do tratamento de bordo.

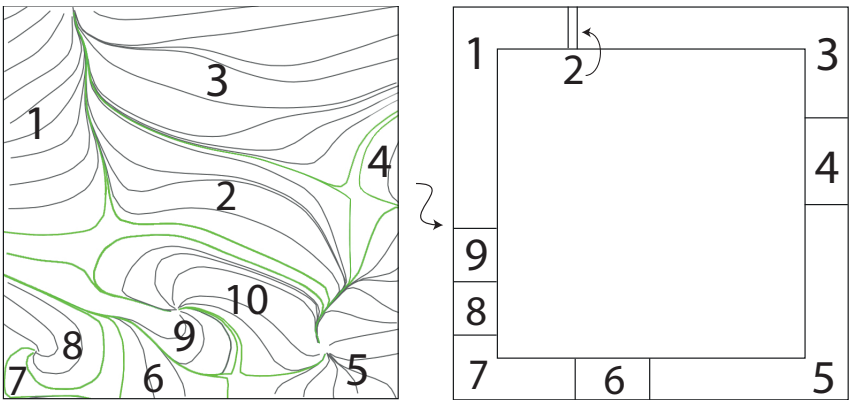


Figura 4.9: Segmentação a partir das separatrizes

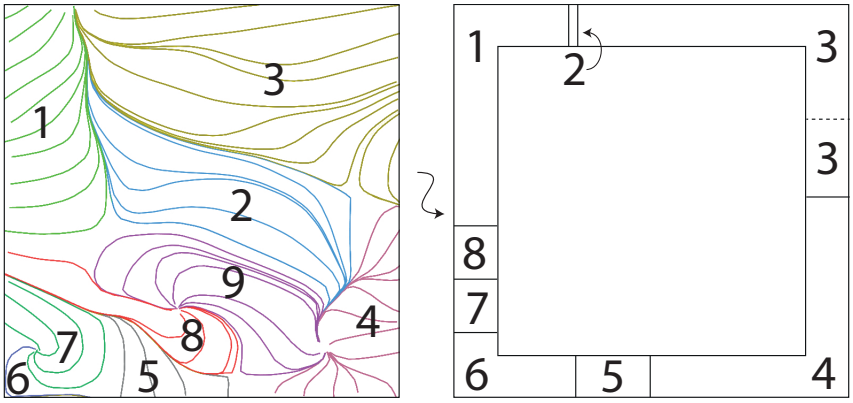


Figura 4.10: Segmentação com a primeira parte de tratamento de bordo

Como pode ser observado na Figura 4.9, existem grupos adjacentes, cujas linhas de fluxo se inicam e terminam no bordo, mas que deveriam ser interpretados como grupos distintos - grupo 3 e 4, o que não ocorre na Figura 4.10 - grupo 3.

Para isso, podemos recuperar a informação de quais linhas de fluxo estão associadas às células de bordo. Como as linhas não se intersectam, basta descobrir quais linhas de fluxo delimitam os grupos percorrerendo as células de bordo e selecionar as linhas de fluxo adequadas.

Na Figura 4.11, podemos observar a segmentação final do campo.

4.3 Imagens Auto-Animadas e Grafo Topológico

Embora a qualidade dos resultados obtidos por Chi *el al.* (6) seja alta, o custo para atingí-los também é relativamente grande, pois usa uma otimização força bruta com semente aleatória. Além dos dois parâmetros que são otimizados no posicionamento (cor inicial e tamanho do segmento), uma questão interessante é que a velocidade da convergência também depende da escolha das linhas de fluxo.

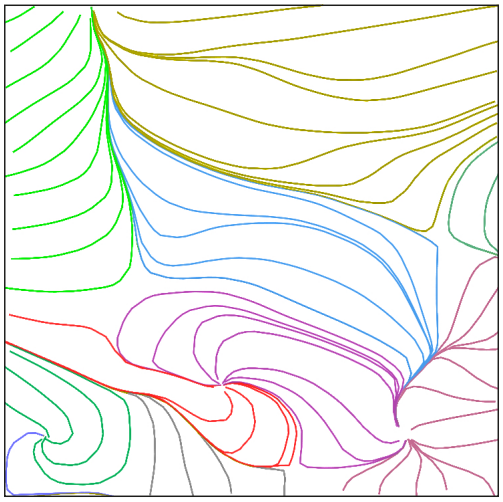
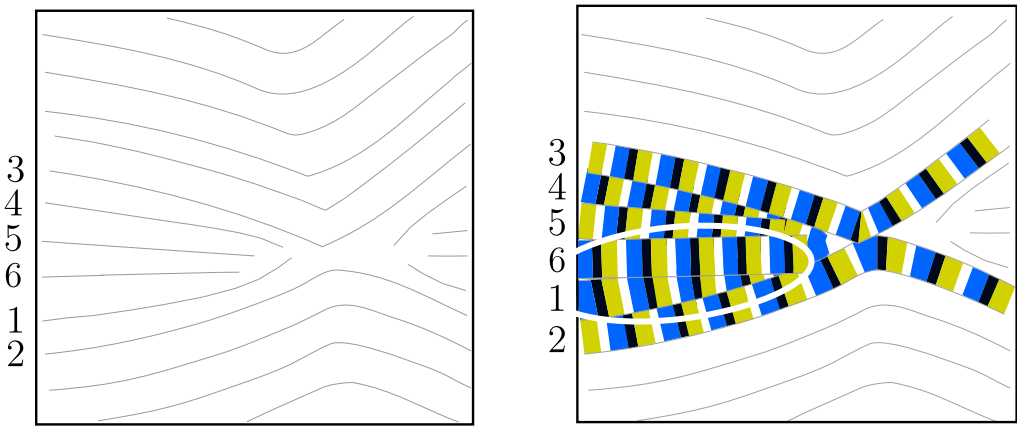


Figura 4.11: Segmentação final.



4.12(a): Linhas de fluxo

4.12(b): Posicionamento dos fragmentos

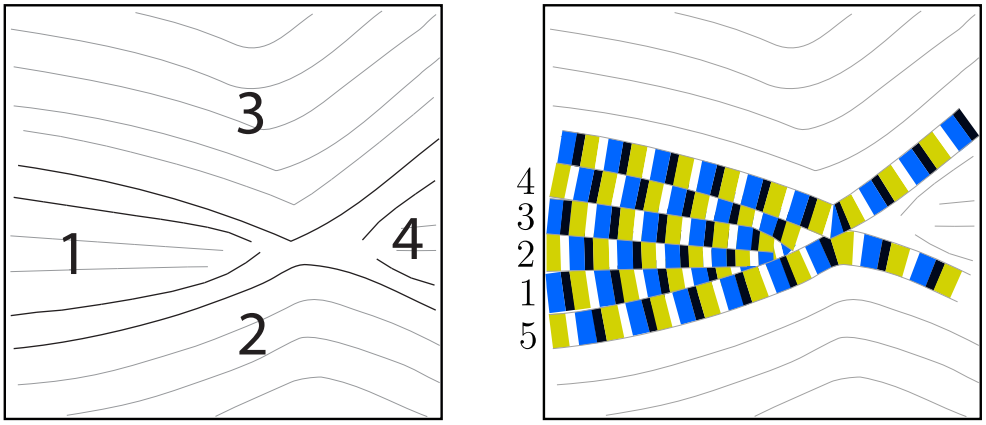
Figura 4.12: Posicionamento dos fragmentos desconsiderando a segmentação do campo.

Tomemos a Figura 4.12(a) como exemplo. Seja a linha de fluxo 1 a escolha inicial. O próximo passo, seguindo a otimização descrita na seção 3.4.2, é a otimizarmos em relação à sua vizinha. Suponha que a linha 2 seja a vizinha escolhida. Feito isso, faremos a otimização da linha 2 com a sua vizinha, linha 3, e da linha 3 com a 4, e assim seguiremos o processo. Quando desejarmos otimizar a linha 6 com a 1, o processo pode se complicar um pouco. Como pode ser observado na Figura 4.12(b), um padrão indesejado pode ser criado, o que prejudica o efeito da ilusão. Assim, será necessário recomeçar a otimização considerando novos parâmetros.

Um dos motivos desse problema ocorrer é que as linhas de fluxo escolhidas como vizinhas têm comportamento relativamente diferente, o que não evidencia qual seria o melhor parâmetro para a otimização.

Então, a otimização entre linhas com comportamento semelhante se da de maneira mais simples e robusta, permitindo uma estratégia adaptativa ao invés de força bruta. Uma forma de fazermos isso é justamente por meio da segmentação do campo descrita na seção 4.2, que particiona o campo vetorial em grupos de linhas de fluxo com comportamento semelhante.

Façamos então o processo de otimização no mesmo campo da Figura 4.12, mas agora utilizando a segmentação do campo 4.13.



4.13(a): Campo segmentado

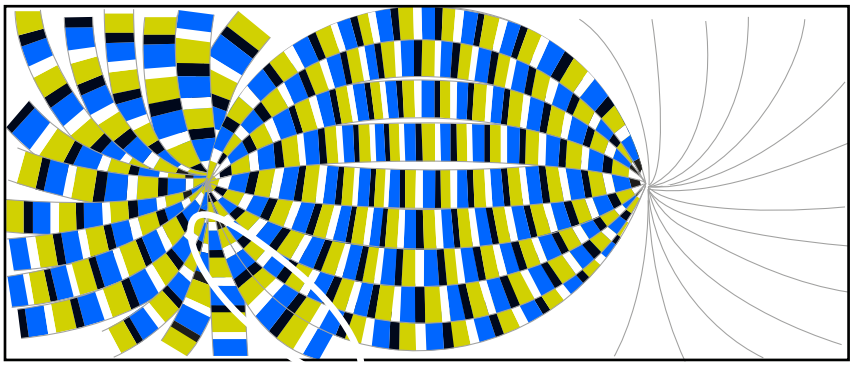
4.13(b): Posicionamento dos fragmentos utilizando a segmentação

Figura 4.13: Posicionamento dos fragmentos considerando a segmentação do campo.

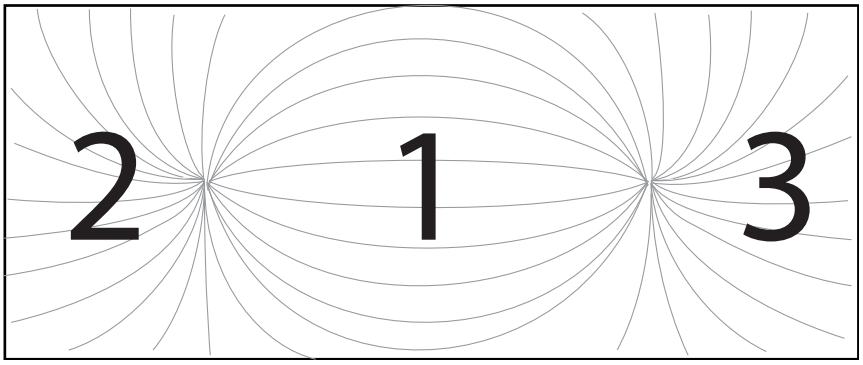
Como as linhas dentro do grupo se comportam de maneira semelhante, o primeiro parâmetro que modificaremos é a cor inicial das linhas, o que, em boa parte dos casos, já mostra um resultado satisfatório. Nas transições de grupos, otimizamos as linhas vizinhas entre grupo e continuamos a otimização agora no novo grupo.

Em alguns casos, a otimização por meio dos grupos pode não ocorrer de forma tão imediata, sendo necessário repetir o processo. A vantagem em utilizar os grupos, é que, diferentemente do que acontece quando é feita a otimização de forma global, não é necessário repetir o processo para todas as linhas do campo, será preciso apenas repetir o processo para o grupo corrente, deixando os demais intactos.

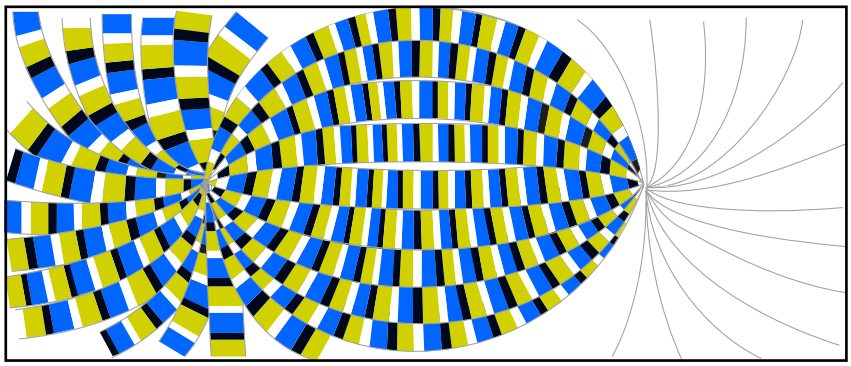
Um exemplo é mostrado na Figura 4.14. Nesta figura, primeiramente foi feita a otimização no grupo 1. Caso a transição entre o grupo 1 e 2 não seja razoável (Figura 4.14(a)), basta repetirmos o processo para o grupo 2 sem ser necessário modificarmos o grupo 1(ver Figura 4.14(c)). Caso a segmentação do campo vetorial não tenha sido considerada, não teremos como discernir quais linhas não precisarão ser modificadas, sendo necessário, portanto, repetir o processo para todas as linhas, aumentando o custo do processo.



4.14(a): Problema de posicionamento na transição dos grupos



4.14(b): Segmentação do campo



4.14(c): Reposicionamento dos fragmentos do grupo 2

Figura 4.14: Otimização considerando a segmentação do campo.

5
Resultados

Neste capítulo são apresentados alguns resultados obtidos tanto na edição como na visualização de campos vetoriais. São feitas comparações entre as técnicas para geração da visualização do campo por meio de imagens auto-animadas utilizando a segmentação proposta - $OCGT$ e sem a utilização - $OSGT$. Para geramos os resultados de edição, utilizamos um Mac Pro com processador Intel 2 x 3.2GHz Quad-Core com 18GB de memória. Já para a visualização por imagens auto-animadas, utilizamos um MacBook Pro com processador Intel Core 2 Duo de 2.4 GHz com 4GB de memória.

Edição: Apresentaremos resultados experimentais em dado sintético e em dado proveniente de medição. Como estamos tratando de campos vetoriais relativamente pequenos (Dado analítico: 2500 pontos; Dado de medição: 15624) armazenados em uma grade regular, a interface responde em tempo real às iterações do usuário.

Primeiramente nós validamos nossa abordagem em um campo vetorial sintético, corrompido por um ruído artificial não Gaussiano (ver Figura 5.1). Podemos remover o ruído do campo vetorial de forma adaptativa recuperando as singularidades originais. Utilizamos o espaço de escala Gaussiano com a interpolação de Shepard com núcleo - SN. Observe que variando o raio do núcleo, podemos manter uma porção maior de dados da escala selecionada.

Testamos nosso método em um dado real adquirido por um *Particle Image Velocimetry* (21) - PIV (ver Figura 5.2). Esse experimento é medido a partir de um jato de água, onde a água parte do lado esquerdo da imagem e bate contra a parede do lado direito. A parte à esquerda da imagem apresenta bastante ruído pois a quantidade de água é menor. Já a parte direita apresenta turbulência. Do lado direito então, existem singularidades importantes que desejamos manter, porém estas desaparecem antes das singularidades causadas pelo ruído. No campo reconstruído, as singularidades desejadas são recuperadas.

Os tempos para processamento de ambos os dados podem ser vistos na Tabela 5.1.

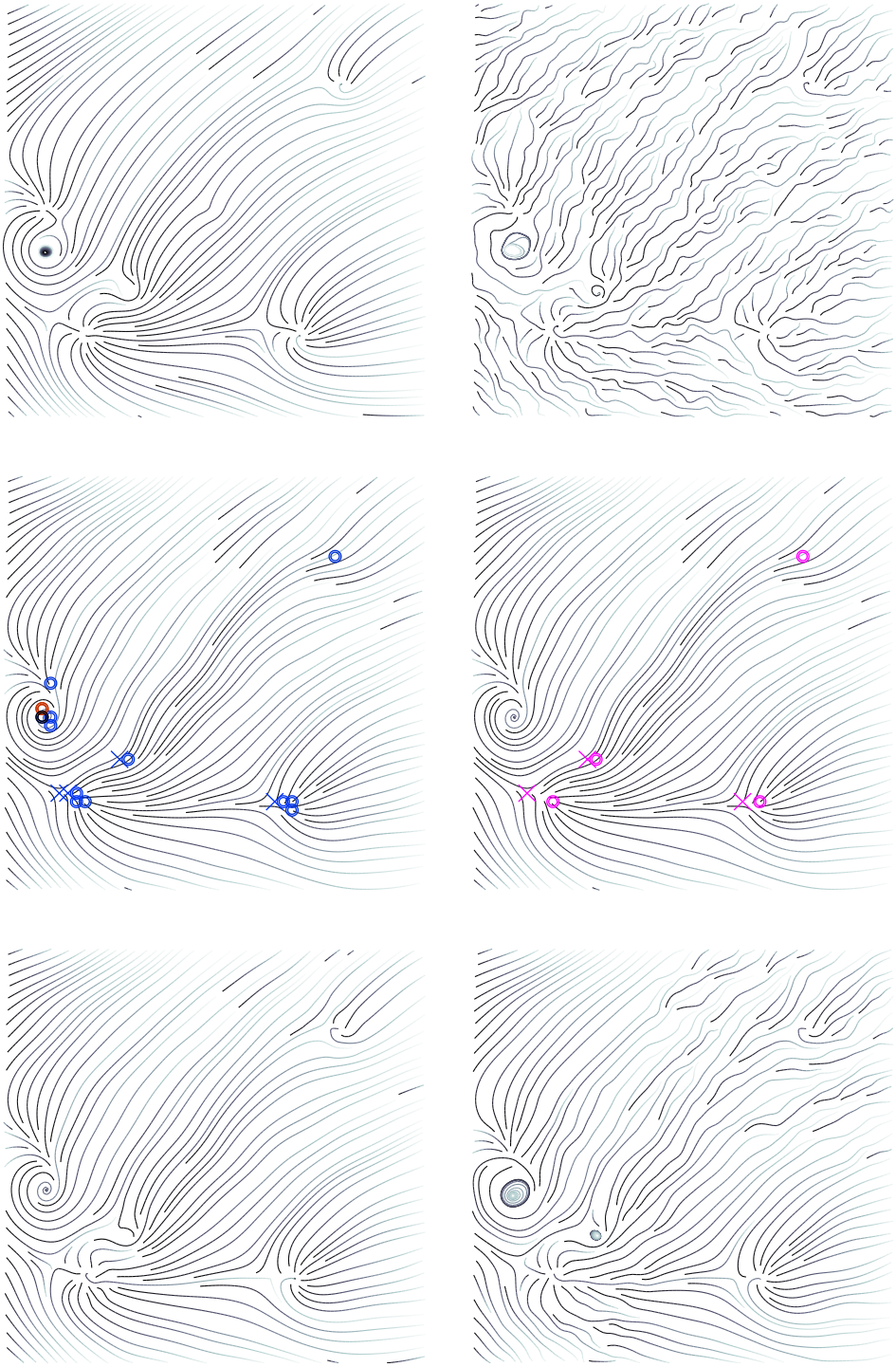


Figura 5.1: Experimento em campo vetorial sintético (topo à esquerda) artificalmente corrompido por ruído não gaussiano (topo à direita). O usuário pode escolher entre singularidades que desapareceram antes da escala s_0 (em azul) ou singularidades que podem ser suavizadas fora da escala $s > s_0$ (em vermelho, no meio à esquerda). A partir da seleção do usuário, reconstruímos o campo vetorial mantendo as singularidades da escala selecionadas em um raio pequeno (em baixo à esquerda) ou grande (em baixo à direita).

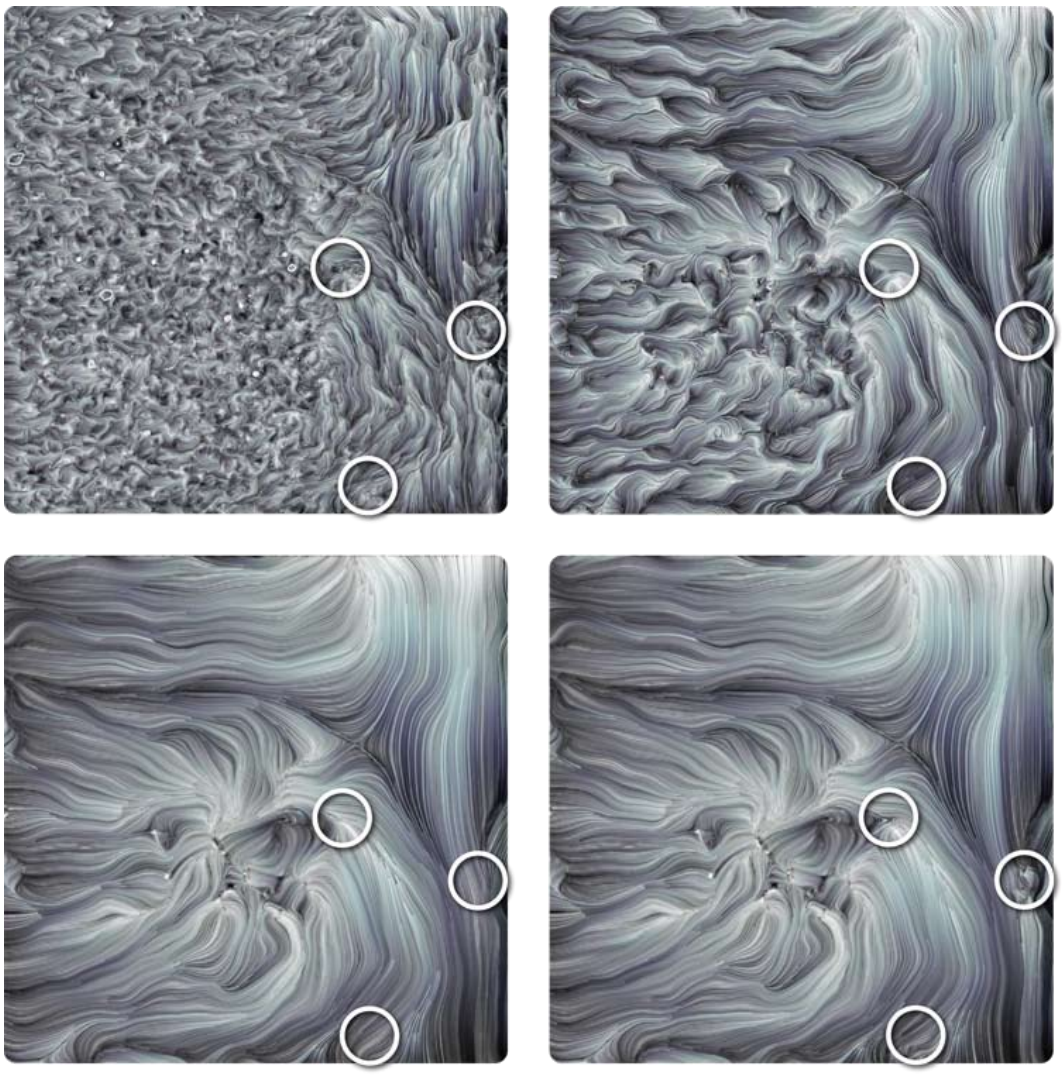
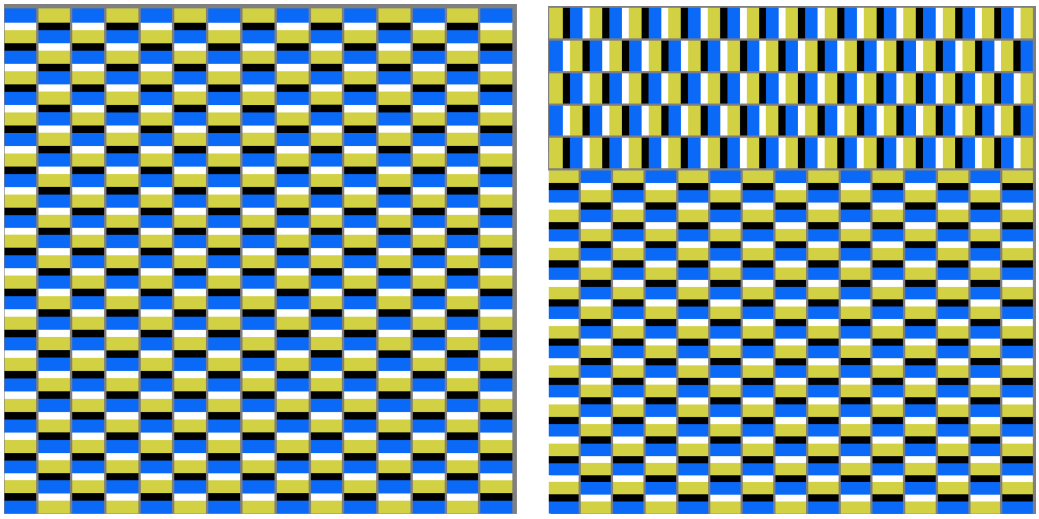


Figura 5.2: Campo vetorial medido por um PIV (topo à esquerda) em um experimento com jato de água: o espaço de escala se encontra no passo s = 30 ate s = 100 (topo à direita e em baixo à esquerda, respectivamente). No passo s=30, as singularidades do lado direito foram mantidas mas ainda há uma grande quantidade de ruído. Já no passo s=100, há pouco ruído mas as singularidades desapareceram. Selecionando as singularidades do lado direito da imagem, obtemos um melhor comportamento do fluido (em baixo à direita).

Dado	**Fig**	**Tam**	**Filtro**		**Singularidade**		**Escala**	**Reconstrução**		
			tipo	*(ms)*	*tipo*	*(ms)*	**selec**	*tipo*	*resol*	*aval*
Analítico	5.1	2500	G_σ	18.9	w_Γ	98.0	7.3	SN		0.1
PIV	5.2	15624	G_σ	135.0	$\mathbf{b}=0$	947.6	65.6	RBF	0.1	7.6

Tabela 5.1: Tempos, em milisegundos, para cada passo da edição.

Visualização por imagens auto-animadas: Para a comparação dos resultados, utilizamos o seguinte critério: geramos a visualização de um mesmo campo em duas versões, uma utilizando a segmentação do campo e uma sem a segmentação. Nas duas versões, utilizamos a otimização descrita em 3.4.2. Afim de obtermos uma comparação entre as duas versões, fixamos um fator *pontos não otimizados - PNO* - para mensurar a quantidade de *pixels* não amarelos durante a etapa de otimização. Quando o mesmo *PNO* é obtido em ambas as imagens, paramos a otimização.



5.3(a): Campo estático 5.3(b): Composição de campos estáticos

Figura 5.3: Resultado 1 - Campos estáticos.

Neste primeiro resultado, a visualização do campo em 5.3(a) foi gerada em 0.89 *segs* em ambas as técnicas, pois neste caso, não foi necessária a geração do grafo topológico, já que o campo não apresenta singularidades. Em 5.3(b), observamos um resultado semelhante, já que também não foram detectadas singularidades em todo o campo. Neste exemplo, o resultado foi obtido em 0.93*segs*. O único *overhead* de nossa proposta nesses casos é a detecção de singularidades, que demorou 0.0002 *segs*.

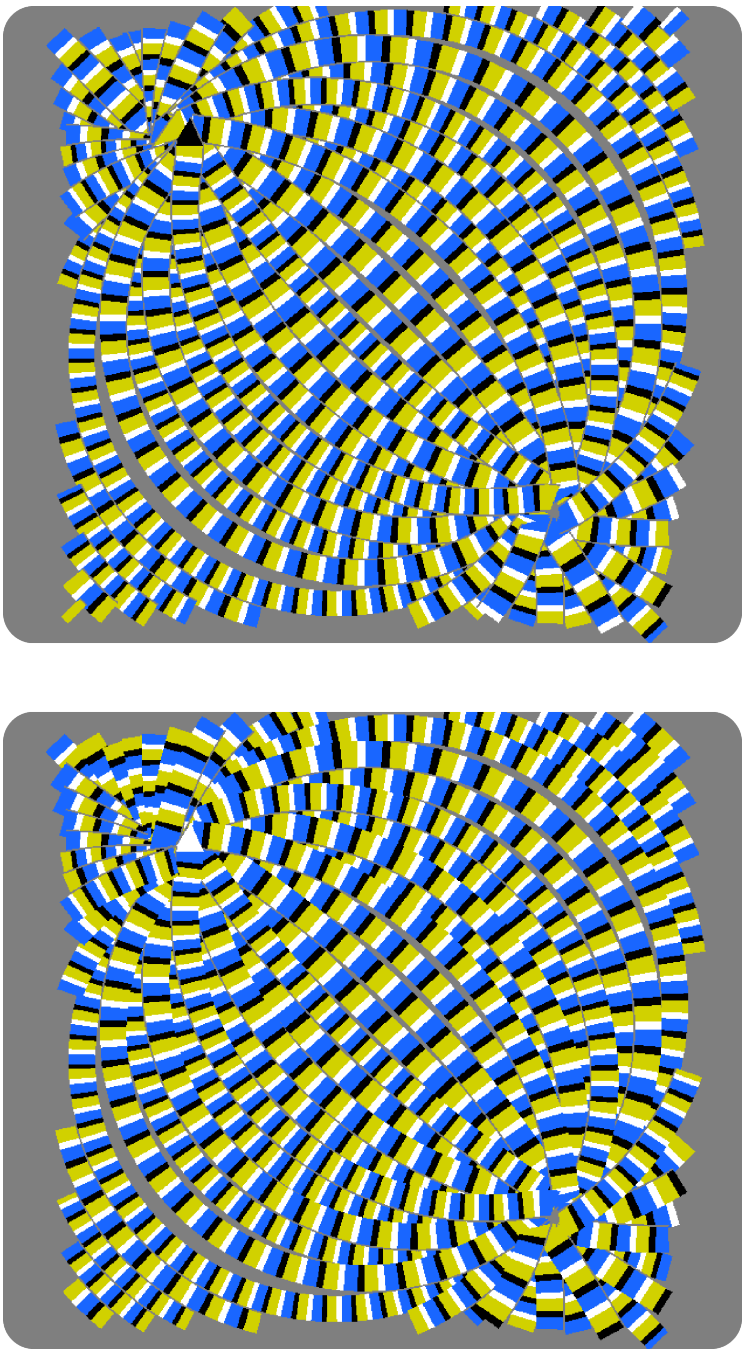


Figura 5.4: Resultado 2 - Campo com duas singularidades. Visualização gerada utilizando a segmentação do campo (topo). Visualização sem utilizar a segmentação do campo (em baixo).

Neste segundo resultado, o tempo utilizado para a geração do grafo topológico foi de 0.0002 *segs* e para a geração da otimização foi de 1.4203 *segs*, totalizando 1.4205 *segs*. No campo onde não foi feita a segmentação, o tempo gasto na otimização foi de 3.0696 *segs*. Assim, a versão utilizando o grafo topológico foi gerada gastando 46.3% do tempo da versão sem o grafo.

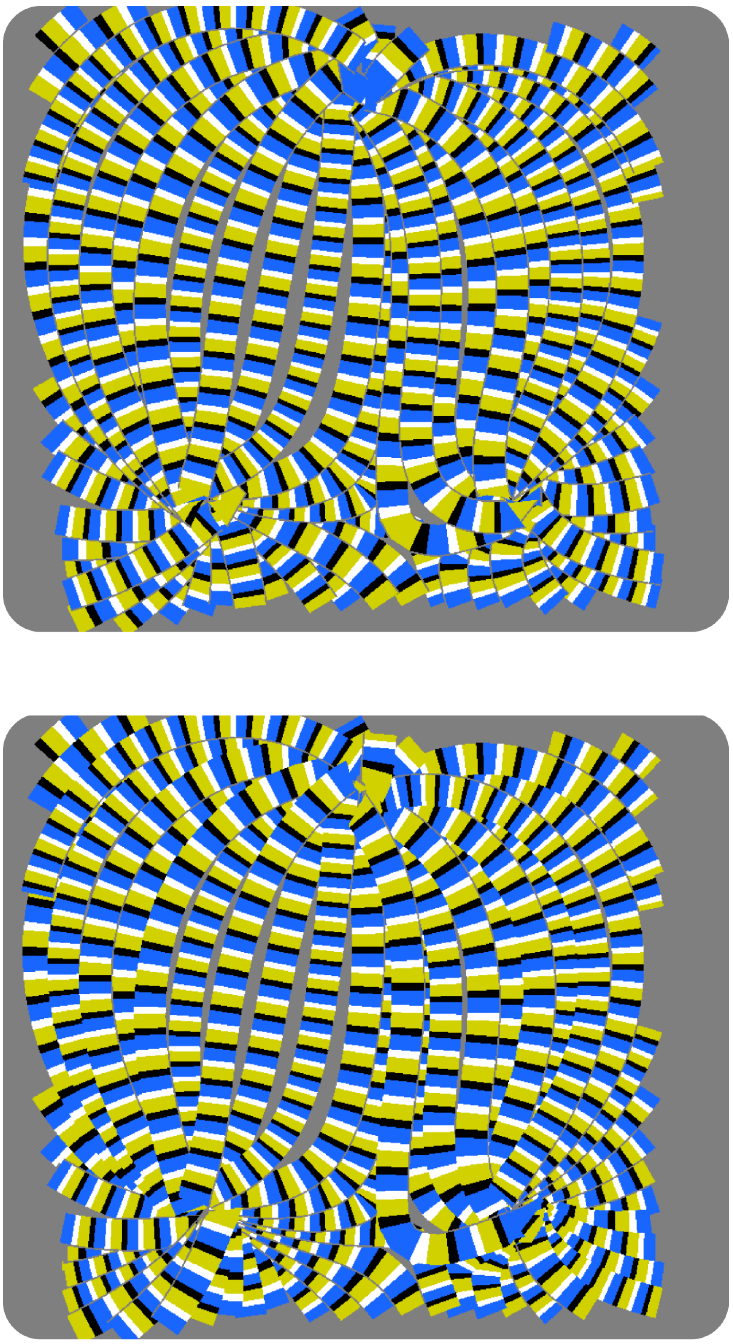


Figura 5.5: Resultado 3 - Campo com quatro singularidades. Visualização gerada utilizando a segmentação do campo (topo). Visualização sem utilizar a segmentação do campo (em baixo).

No resultado da Figura 5.5, observamos que o tempo utilizado para a geração do grafo topológico foi de 0.0004 *segs* e para a geração da otimização foi de 1.3022 *segs*, totalizando 1.3026 *segs*. Já no campo onde não foi feita a segmentação, o tempo gasto na realização da otimização foi de 3.0011 *segs*. Desta forma, a versão utilizando o grafo topológico foi gerada gastando 43.4% do tempo da versão sem o grafo.

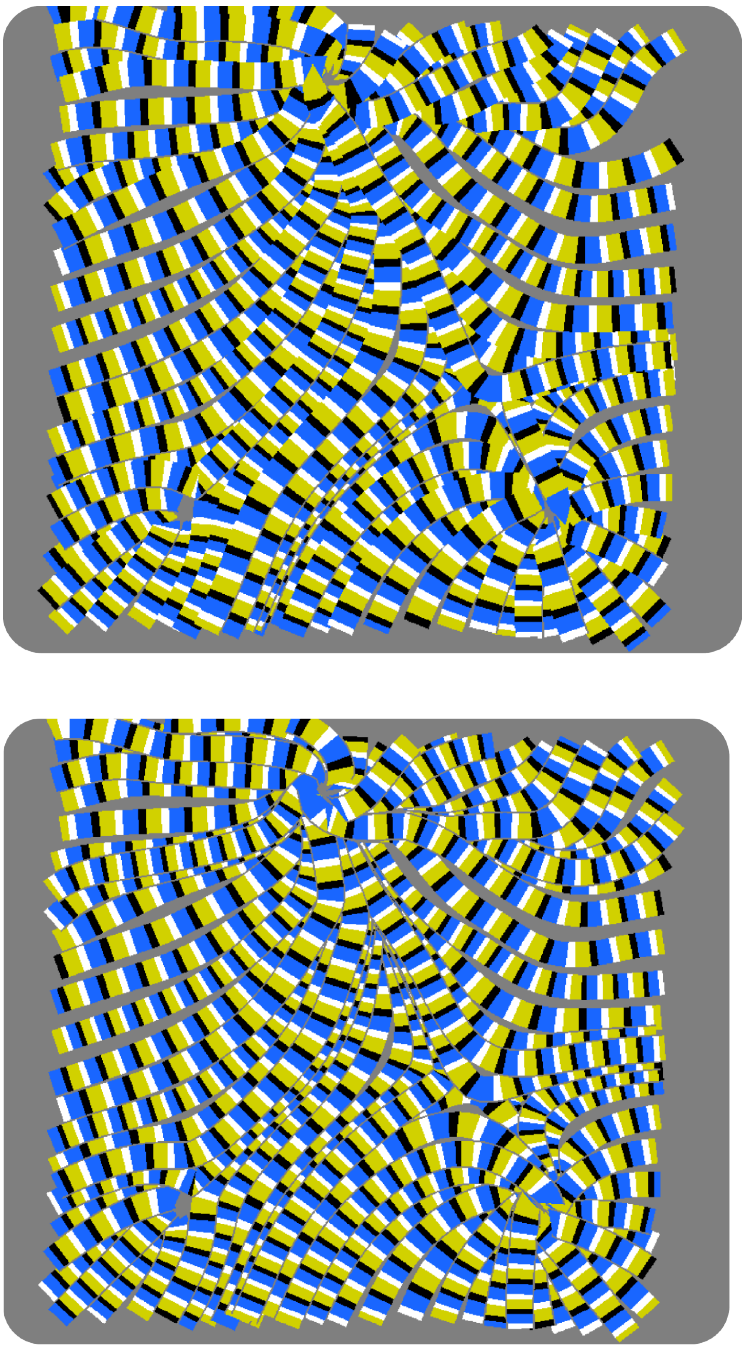


Figura 5.6: Resultado 4 - Campo com quatro singularidades. Visualização gerada utilizando a segmentação do campo (topo). Visualização sem utilizar a segmentação do campo (em baixo).

O tempo utilizado para a geração do grafo topológico do resultado da Figura 5.6 foi de 0.0004 *segs* e para a geração da otimização foi de 1.5708 *segs*, totalizando 1.5712 *segs*. Já o tempo gasto na etapa de otimização do campo onde não foi feita a segmentação foi de 3.7206*segs*. Observamos então que a versão utilizando o grafo topológico gastou 42.2% do tempo da versão sem o grafo.

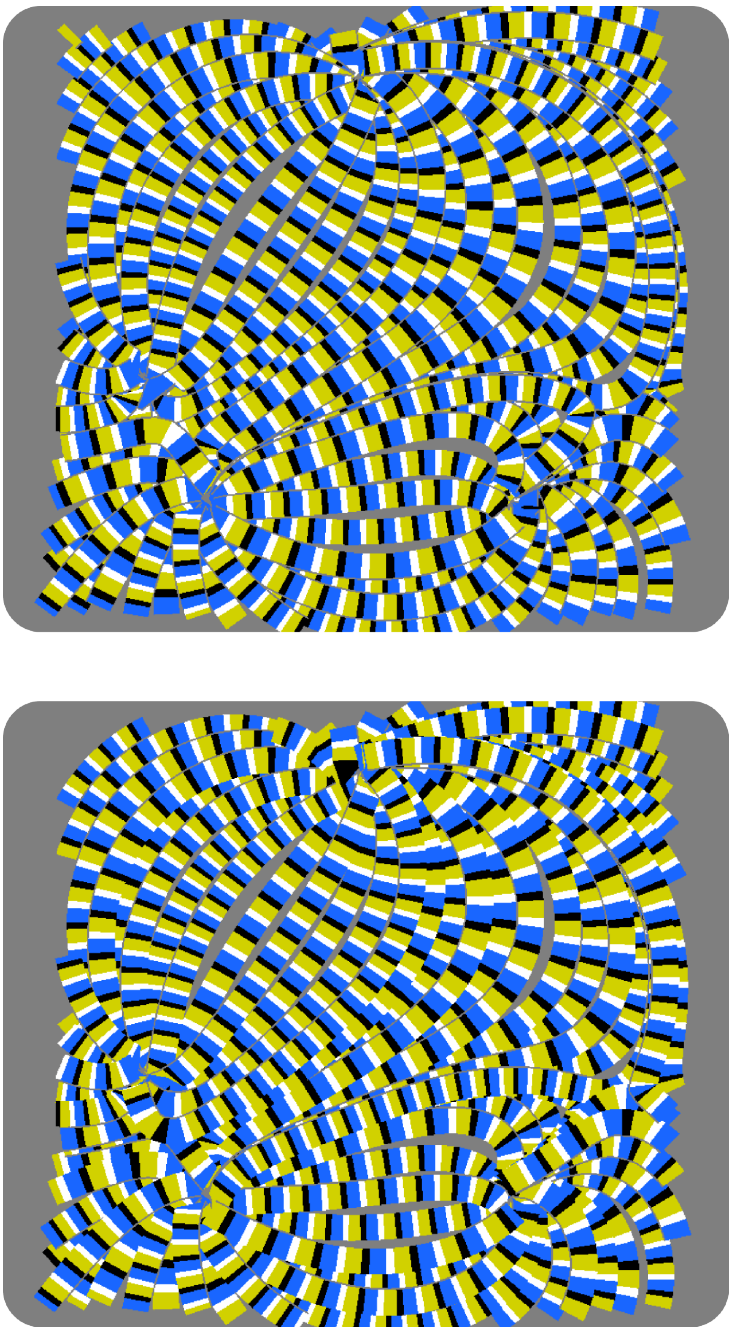


Figura 5.7: Resultado 5 - Campo com seis singularidades. Visualização gerada utilizando a segmentação do campo (topo). Visualização sem utilizar a segmentação do campo (em baixo).

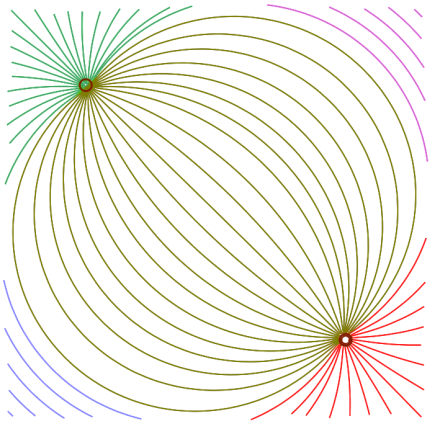
Neste resultado o tempo utilizado para a geração do grafo topológico foi de 0.0003 *segs* e para a geração da otimização foi de 1.5459 *segs*, totalizando 1.5462 *segs*. No campo onde não foi feita a segmentação, o tempo gasto na otimização foi de 3.7717 *segs*. Sendo assim, a versão utilizando o grafo topológico foi gerada gastando 40.9% do tempo da versão sem o grafo.

Abaixo observamos uma tabela sintetizando os tempos dos resultados apresentados.

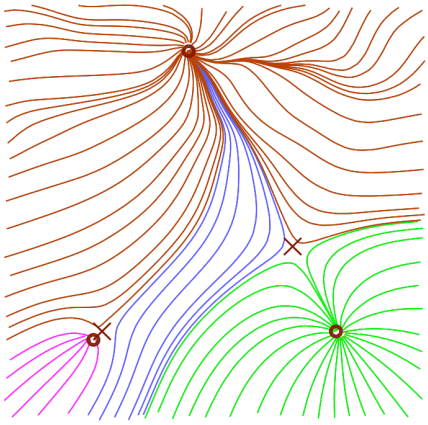
Dado	**Fig**	**Grafo**	**OCGT**	**Tempo total**	**OSGT**	**Ganho**
Resutado 1	5.3	0	(a)0.89 (b)0.93	(a)0.89 (b)0.93	(a)0.89 (b)0.93	0.0 %
Resutado 2	5.4	0.0002	1.4203	1.4205	3.0696	53.7%
Resutado 3	5.5	0.0004	1.3022	1.3026	3.0011	56.6%
Resutado 4	5.6	0.0004	1.5708	1.5712	3.7206	57.8%
Resutado 5	5.7	0.0003	1.5459	1.5462	3.7717	59.1%

Tabela 5.2: Tempos, em segundos, dos resultados apresentados no processo de visualização.

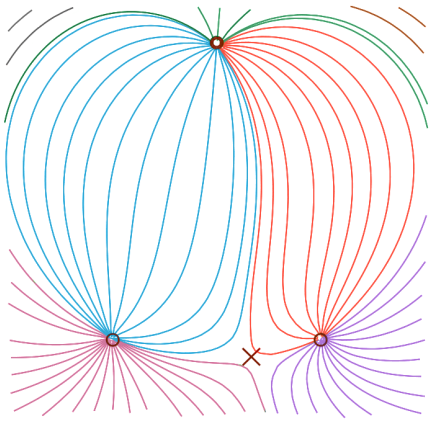
Logo a seguir, estão dispostos os grafos topológicos dos campos acima.



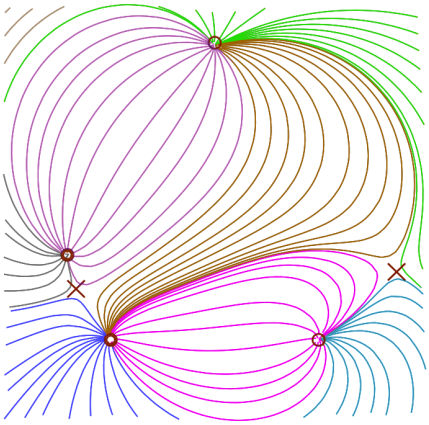
5.8(a): Grafo do Resultado 2



5.8(b): Grafo do Resultado 3



5.8(c): Grafo do Resultado 4



5.8(d): Grafo do Resultado 5

Figura 5.8: Grafos topológicos.

Fizemos também experimentos em dados antes de ser realizada a edição do campo vetorial:

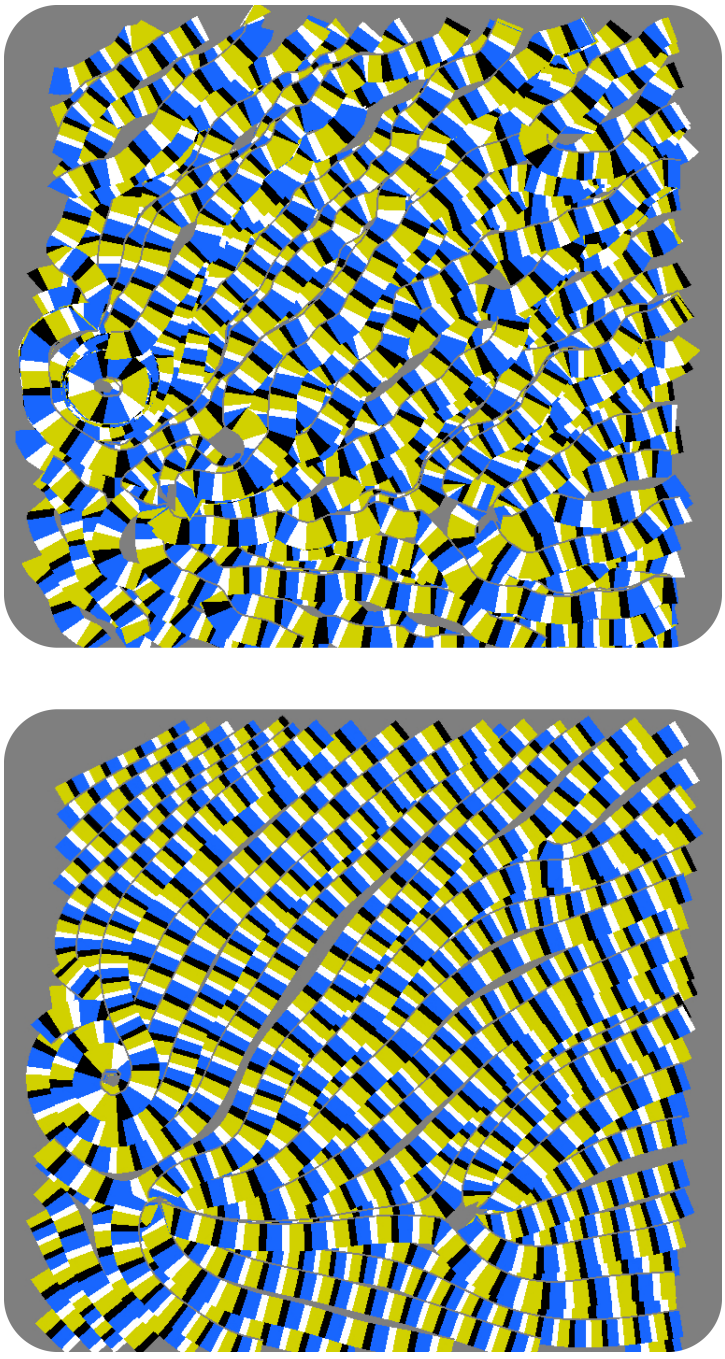


Figura 5.9: Resultado 6 - Dado analítico

No resultado 6 (Figura 5.9), observamos o resultado da visualização do campo vetorial por imagens auto-animadas sem utilizar a edição do campo (topo) e utilizando a edição (em baixo). Para gerar a visualização do dado sem a edição foram utilizados 4.8029 *segs*, enquanto na visualização do dado editado, foram utilizados 2.6185 *segs*. A versão editada foi então gerada gastando 54.5% do tempo da versão não editada.

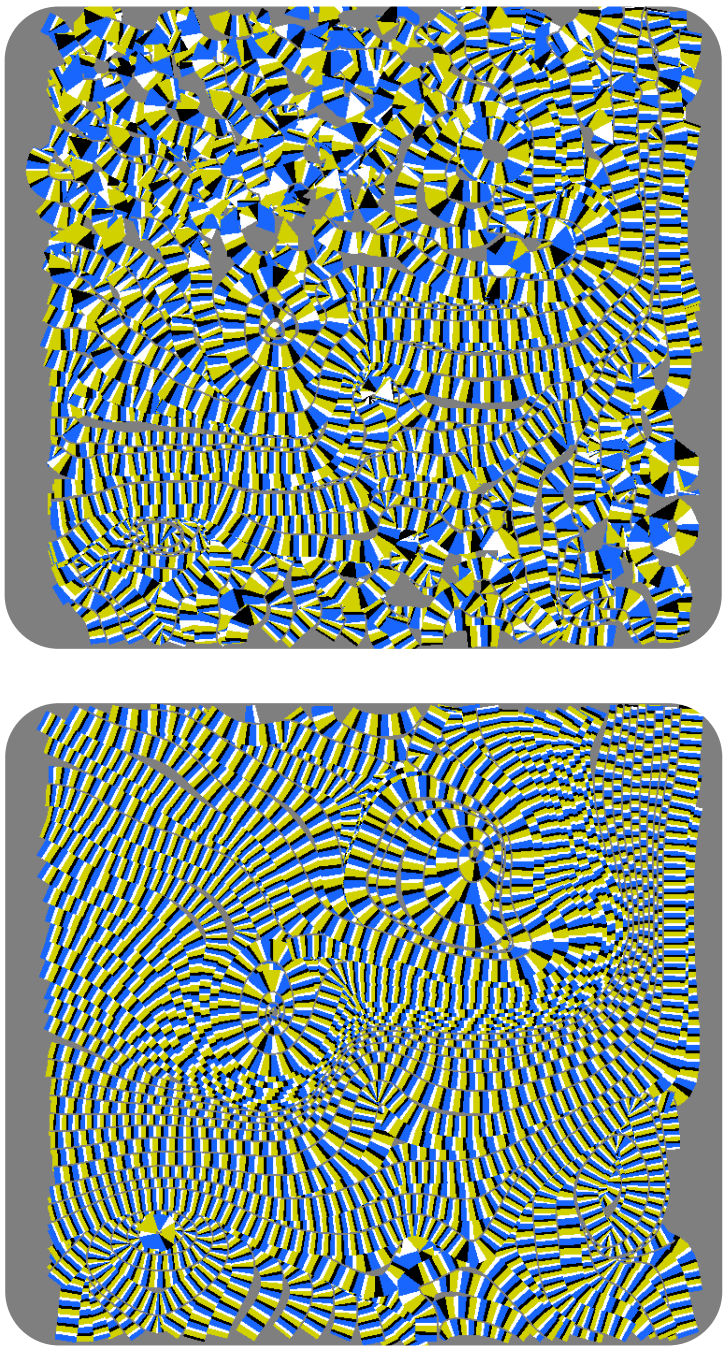


Figura 5.10: Resultado 7 - PIV

No PIV, resultado 7 (Figura 5.10), observamos a visualização do campo vetorial por imagens auto-animadas sem utilizar a edição do campo (topo) e utilizando a edição (em baixo). No dado não editado, foram gastos 24.5488 *segs*, enquanto, na visualização do dado editado, foram gastos 9.2604 *segs*. A versão editada foi então gerada gastando 37.7% do tempo da versão não editada.

Abaixo, obsevamos uma tabela sintetizando o tempo do processo de visualização por imagens auto-animadas. Dispomos os tempos utilizados tanto para o dado sem edição como para o dado editado.

Dado	**Fig**	**Dado sem edição**	**Dado com edição**	**Ganho**
Resutado 6	5.9	4.8029	2.6185	55.5%
Resutado 7	5.10	24.5488	9.2604	62.3%

Tabela 5.3: Tempos, em segundos, dos resultados apresentados no processo de edição seguido da visualização.

6
Considerações Finais e Trabalhos Futuros

Este trabalho tem como foco auxiliar nos processos de análise e visualização de campos vetoriais. Para esse fim, desenvolvemos uma metodologia para a edição e para a segmentação de campos vetoriais, ambos baseados na topologia. O processo de segmentação se faz de grande utilidade principalmente no caso de visualização através de imagens auto-animadas.

No processo de edição, desenvolvemos uma metodologia semi-automática para a remoção de ruídos de campos vetoriais onde o principal parâmetro é o conhecimento do usuário sobre o dado. Na interface desenvolvida, disponibilizamos para o usuário um conjunto de mudanças topológicas visando permitir ao usuário guiar o processo de suavização do dado. Essa abordagem surgiu da necessidade da remoção do ruído de forma adaptativa, pois, principalmente em dados reais, o ruído se apresenta em escalas diferentes, podendo até estar em escala maior que o próprio dado.

O método proposto suporta diferentes técnicas de detecção de singularidades, de geração de espaço de escala e de interpolação. Na detecção de singularidades, foram utilizados dois métodos clássicos, o da interpolação bilinear e o *winding number*, e foi proposto um terceiro método para detectar regiões com potencial para ser ponto singular - regiões fracas. Para a geração do espaço de escala, utilizamos a filtragem por *random walk* com núcleo Gaussiano e com o núcleo anisotópico. Já para a reconstrução do campo, utilizamos funções de base radial (RBF) com base Gaussiana e o método de interpolação de Sherpad com núcleo Gaussiano (4).

Já no processo de visualização, embora este já fosse realizado por Chi *et al.*(6) de forma automática, o alto grau de qualidade dos resultados está condicionado ao alto custo para atingí-los. Buscando acelerar o processo de geração deste tipo de imagens associadas a campos vetoriais, este trabalho se utilizou das informações topológicas do campo de forma a segmentá-lo em componentes conexas nas quais as linhas de fluxo se comportam de forma semelhante em seu interior.

Para a realização da segmentação, foram primeiramente detectadas as singularidades do campo vetorial. Após a detecção as singularidades, foram

calculadas as linhas de fluxo dentro do campo vetorial. Posteriormente, foi realizada uma primeira segmentação do campo, onde neste caso, todo o bordo foi tratado como uma única singularidade. Um segundo processamento foi realizado de forma a reparar os casos onde o bordo precisava ser tratado como mais de uma singularidade. Ao fim desse processo, obtemos o grafo topológico do campo vetorial. A eficiência desse processo permite não adicionar custos significantes comparado com a geração da imagem auto-animada.

Em posse do grafo, partimos para a geração das imagens auto-animadas. Para esse fim, utilizamos a mesma otimização proposta por Chi *et al.* (6), porém, nos utilizamos da topologia do campo vetorial, dada pela segmentação. Com essa informação adicional, conseguimos evitar que regiões onde a otimização já estivesse no padrão desejado fossem modificadas, pois era nesta etapa onde o consumo de tempo mais se elevava.

Como trabalhos futuros, pretendemos estender tanto o processo de edição, como o processo de visualização por imagens auto-animadas para maiores dimensões.

Para o processo de visualização, também buscamos estudar a possibilidade de paralelização na otimização de cada região, já que é possível encontrar independência entre algumas delas.

Referências Bibliográficas

[1] ANDRONOV, A.; LEONTOVICH, E.; GORDON, I. ; MAIER, A. **Qualitative Theory of Second-Order Dynamic System**. Israel Program for Scientific Translations, 1973.

[2] BACKUS, B.; ORUC, I. **Illusory motion from change over time in the response to contrast and luminance**. Journal of Vision, 5:1055–1069, 2005.

[3] BAUER, D.; PEIKERT, R. **Vortex tracking in scale-space**, 2002.

[4] BECKER, J. **Método de interpolação de shepard baseado em núcleo**. Dissertação de Mestrado, Departamento de Matemática, PUC-Rio, 2009. Orientada por Hélio Lopes.

[5] CABRAL, B.; LEEDOM, L. **Imaging vector fields using line integral convolution**. In SIGGRAPH, p. 263–270, 1993.

[6] CHI, M.; LEE, T.; QU, Y. ; WONG, T. **Self-animating images: Illusory motion using repeated asymmetric patterns**. In SIGGRAPH, 2008. Artigo 62.

[7] GLOBUS, A.; LEVIT, C. ; LASINSKI, T. **A tool for visualizating the topology of three-dimensional vector fields**. In Visualization, IEEE, p. 33–40, 1991.

[8] HELMAN, J.; HESSELINK, L. **Visualizing vector field topology in fluid flows**. IEEE Computer Graphics and Applications, p. 36–46, 1991.

[9] HIRSCH, M.; SMALE, S. **Differential Equations, Dynamical Systems and Linear Algebra**. Academic Press, 1974.

[10] JOBARD, B.; LEFER, W. **Creating evenly-spaced streamlines of arbitrary density**. In Eurographics Workshop on Visualization in Scientific Computing, p. 45–55, 1997.

[11] JU, T.; ZHOU, Q.-Y. ; HU, S.-M. **Editing the topology of 3d models by sketching**. Transactions on Graphics (Proceedings of Siggraph), 26(3):42, 2007.

[12] KITAOKA, A. **Rotating snakes**. http://www.psy.ritsumei.ac.jp/ akitaoka/rotsnakee.html., 2003.

[13] KITAOKA, A. **Anomalous motion illusion and stereopsis**. Journal of Three Dimensional Images, Japan, 20:9–14, 2006.

[14] KITAOKA, A. **The effect of color on the optimized fraser-wilcox illusion**. 9th L'OREAL Art and Science of Color Prize, Tokyo, p. 1–16, 2006.

[15] KLEIN, T.; ERTL, T. **Scale-Space tracking of critical points in 3D vector fields**. In: TOPOLOGY-BASED METHODS IN VISUALIZATION, 2005.

[16] LUKAC, R.; PLATANIOTIS, K. N.; SMOLKA, B. ; VENETSANOPOULOS, A. N. **Generalized selection weighted vector filters**. Journal of Applied Signal Processing, p. 1870–1885, 2004.

[17] MEBARKI, A.; ALLIEZ, P. ; DEVILLERS, O. **Farthest point seeding for placement of streamline**. IEEE Transaction on Visualization and Computer Graphics, 10:479–486, 2005.

[18] NASCIMENTO, R.; PAIXÃO, J.; LOPES, H. ; LEWINER, T. **Topopogy-aware vector field denoising**. In: SIBGRAPI, 2010.

[19] PAIXÃO, J.; LAGE, M.; PETRONETTO, F.; BORDIGNON, A.; PESCO, S.; TAVARES, G.; LEWINER, T. ; LOPES, H. **Random walks for vector field denoising**. In: SIBGRAPI. IEEE, 2009.

[20] PLATANIOTIS, K. N.; VENETSANOPOULOS, A. N. **Color image processing and applications**. Springer, 2000.

[21] RAFFEL, M.; WILLERT, C. ; KOMPENHANS, J. **Particle Image Velocimetry: A Practical Guide**. Springer, 2002.

[22] SCHEUERMANN, G.; HAMANN, B.; KENNETH, J. ; KOLLMANN, W. **Visualizing local vector field topology**. Journal of Eletronic Imaging, 9(4), 2000.

[23] SHARF, A.; LEWINER, T.; SHKLARSKI, G.; TOLEDO, S. ; COHEN-OR, D. **Interactive topology-aware surface reconstruction**. Transactions on Graphics (Proceedings of Siggraph), 26(3):43.1–43.9, 2007.

[24] SHEPARD, D. **A two-dimensional interpolation function for irregularly-spaced data**. In: ACM NATIONAL CONFERENCE, p. 517–524, 1968.

[25] SMOLKA, B.; LUKAC, R.; CHYDZINSKI, A.; PLATANIOTIS, K. N. ; WOJCIECHOWSKI, W. **Fast adaptive similarity based impulsive noise reduction filter**. Real-Time Imaging, 9(4):261–276, 2003.

[26] SMOLKA, B.; WOJCIECHOWSKI, K. W. **Random walk approach to image enhancement**. Signal Processing, 81(3):465–482, 2001.

[27] SUN, X.; ROSIN, P. L.; MARTIN, R. R. ; LANGBEIN, F. C. **Random walks for mesh denoising**. In: SOLID AND PHYSICAL MODELING, p. 11–22. ACM, 2007.

[28] SUN, X.; ROSIN, P. L.; MARTIN, R. R. ; LANGBEIN, F. C. **Random walks for feature-preserving mesh denoising**. Computer Aided Geometric Design, 25(7):437–456, 2008.

[29] SZCZEPANSKI, M.; SMOLKA, B.; PLATANIOTIS, K. N. ; VENETSANOPOULOS, A. N. **On the geodesic paths approach to color image filtering**. Signal Processing, 83(6):1309–1342, 2003.

[30] TOMASI, C.; MANDUCHI, R. **Bilateral filtering for gray and color images**. In: INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 839–846, 1998.

[31] TRICOCHE, X. **Vector and Tensor Field Topology Simplification, Tracking, and Visualization**. Tese de Doutorado, Schriftenreihe Fachbereich Informatik, University of Kaiserslautem, http://www.cs.purdue.edu/homes/xmt/papers/phd.pdf, 2002.

[32] TRICOCHE, X.; SCHEUERMANN, G. ; HAGEN, H. **Continuous topology simplification of planar vector fields**. In Visualization, IEEE, p. 159–166, 2001.

[33] TURK, G.; BANKS, D. **Image-guided streamline placement**. In SIGGRAPH, p. 453–460, 1996.

[34] WEI, L.-Y. **Visualizing flow fields by perceptual motion**. Tech. Rep. MSR-TR, Microsoft Research, 82, 2006.

[35] WESTENBERG, M. A.; ERTL, T. **Denoising 2D vector fields by vector wavelet thresholding**. Journal of WSCG, 13(1-3):33–40, 2005.