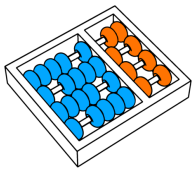
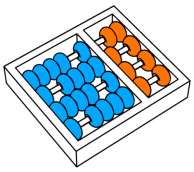


Dalton Ieda Fazanaro

"Metamorfose Planar via Métodos *Level Set* e *Particle Level Set* para a Reconstrução de Superfícies Tridimensionais"

"*Planar Metamorphosis by Level Set and Particle Level Set Methods for Tridimensional Surface Reconstruction*"

CAMPINAS
2013



Universidade Estadual de Campinas
Instituto de Computação

University of Campinas
Institute of Computing

Dalton Ieda Fazanaro

"Metamorfose Planar via Métodos *Level Set* e *Particle Level Set* para a Reconstrução de Superfícies Tridimensionais"

Orientador:
Supervisor: **Prof. Dr. Hélio Pedrini**

"Planar Metamorphosis by Level Set and Particle Level Set Methods for Tridimensional Surface Reconstruction"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

MSc Dissertation presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a Master degree in Computer Science.

Este exemplar corresponde à versão final da Dissertação defendida por Dalton Ieda Fazanaro, sob orientação do Prof. Dr. Hélio Pedrini.

This volume corresponds to the final version of the Dissertation defended by Dalton Ieda Fazanaro, under the supervision of Prof. Dr. Hélio Pedrini.

Assinatura do Orientador / *Supervisor's signature*

CAMPINAS
2013

FICHA CATALOGRÁFICA ELABORADA POR
ANA REGINA MACHADO - CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Fazanaro, Dalton Ieda, 1982-
F297m Metamorfose planar via métodos Level Set e Particle Level Set para a reconstrução de superfícies tridimensionais / Dalton Ieda Fazanaro. – Campinas, SP : [s.n.], 2013.

Orientador: Hélio Pedrini.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

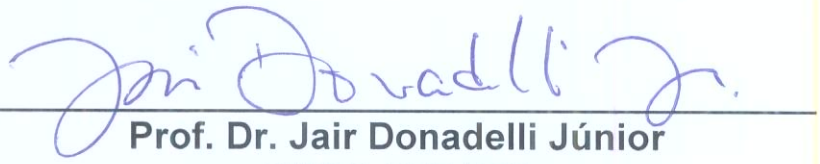
1. Processamento de imagens. 2. Computação gráfica. 3. Geometria computacional. I. Pedrini, Hélio,1963-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

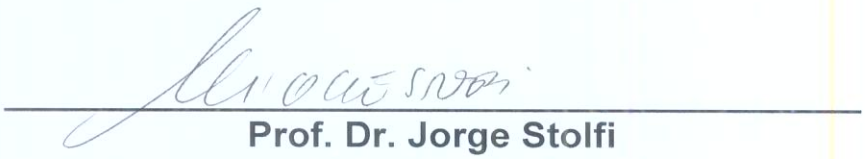
Título em inglês: Planar metamorphosis by Level Set and Particle Level Set methods for tridimensional surface reconstruction
Palavras-chave em inglês:
Image processing
Computer graphics
Computational geometry
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Hélio Pedrini [Orientador]
Jair Donadelli Júnior
Jorge Stolfi
Data de defesa: 15-01-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

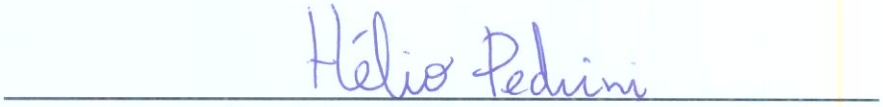
Dissertação Defendida e Aprovada em 15 de Janeiro de 2013, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Jair Donadelli Júnior
CMCC / UFABC



Prof. Dr. Jorge Stolfi
IC / UNICAMP



Prof. Dr. Hélio Pedrini
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Metamorfose Planar via Métodos *Level Set* e *Particle Level Set* para a Reconstrução de Superfícies Tridimensionais

Dalton Ieda Fazanaro

15 de Janeiro de 2013

Banca Examinadora:

- Prof. Dr. Hélio Pedrini (Orientador)
- Prof. Jair Donadelli Jr.
  Centro de Matemática, Computação e Cognição – Universidade Federal do ABC
- Prof. Dr. Jorge Stolfi
  Instituto de Computação – Universidade Estadual de Campinas

Resumo

Inicialmente centralizadas na solução de problemas científicos em Dinâmica dos Fluidos, as interfaces evolutivas, com o advento da modelagem mais eficiente e robusta provida pelo método *Level Set*, expandiram os seus limites originais de aplicabilidade, proporcionando uma nova frente de pesquisa para os campos dos mais diversos, com destaque à Ciência da Computação. Especificamente à área de Processamento de Imagens, os trabalhos até então apresentados, relacionando o *Level Set* à reconstrução de superfícies tridimensionais, concentram-se na reconstrução a partir de uma nuvem de dados dispersos no espaço; a abordagem baseada em fatias planas paralelas e transversais ao objeto a ser reconstruído evidencia-se ainda incipiente. Esse cenário fomenta, portanto, uma análise da viabilidade do *Level Set* para a reconstrução de superfícies tridimensionais. Fundamentando-se nessa constatação, a dissertação propõe-se a oferecer uma metodologia que agregue, simultaneamente, as ideias comprovadamente eficientes já publicadas sobre a aproximação em questão e as propostas para contornar as limitações inerentes ao método ainda não satisfatoriamente tratadas, em particular a suavização excessiva de características finas dos contornos em evolução sob o *Level Set*. Relativamente a esse ponto, o emprego da variante *Particle Level Set* é sugerido como uma possível solução, por sua intrínseca capacidade comprovada para a conservação de massa ou volume de fronteira dinâmicas traduzir-se, presumivelmente, em um controle ao problema destacado. Ao final, conjuntos de dados sintéticos e reais são utilizados para avaliar a metodologia de reconstrução de superfícies tridimensionais apresentada qualitativamente.

Abstract

Evolving interfaces were initially focused on solutions to scientific problems in Fluid Dynamics. With the advent of the more efficient and robust modeling provided by Level Set method, their original boundaries of applicability were extended, offering a new front of research to the more diverse fields, especially to Computer Science. Specifically to Image Processing area, the works published until then, relating Level Set to tridimensional surface reconstruction, centred themselves on reconstruction from a data cloud dispersed in space; the approach based on parallel planar slices transversal to the object to be reconstructed is still incipient. Therefore, this scenario foments a feasibility analysis of Level Set to the reconstruction of tridimensional surfaces. Basing on this fact, this dissertation proposes to offer a methodology that simultaneously integrates the proved efficient ideas already published about such approximation and the proposals to process the inherent limitations of the method not satisfactorily treated yet, in particular the excessive smoothing of fine characteristics of contours evolving under Level Set. In relation to this, the application of the variant Particle Level Set is suggested as a possible solution, for its intrinsic proved capability to preserve mass or volume of dynamic fronts manifests itself, presumably, into a control of the stressed problem. At the end, synthetic and real data sets are used to evaluate the presented tridimensional surface reconstruction methodology qualitatively.

Agradecimentos

Ao meu orientador Hélio Pedrini,
pela confiança desde o início.

Aos meus pais Antonio Carlos e Maria Elisa Fazanaro,
pela confiança desde sempre.

Aos amigos Paulo Henrique Junqueira Amorim e Thiago Franco de Moraes,
pelo auxílio essencial no momento crucial.

Sumário

Resumo vii

Abstract viii

Agradecimentos ix

1 Introdução 1
- 1.1 Contextualização . . . 1
- 1.2 Caracterização do Problema . . . 3
- 1.3 Objetivos e Contribuições . . . 5
- 1.4 Organização do Texto . . . 5

2 Conceitos Teóricos 6
- 2.1 *Level Set* . . . 6
  - 2.1.1 Fundamentação teórica . . . 7
  - 2.1.2 Funções distância sinalizada . . . 10
  - 2.1.3 Metamorfose planar via *Level Set* . . . 12
- 2.2 *Particle Level Set* . . . 13

3 Metodologia e Implementação 15
- 3.1 Etapa 1: Dados de Entrada . . . 16
- 3.2 Etapa 2: Leitura das Fatias . . . 16
- 3.3 Etapa 3: Geração das Fatias Intermediárias . . . 17
  - 3.3.1 Passo 3.1: redimensionamento das fatias . . . 17
  - 3.3.2 Passo 3.2: centralização dos contornos . . . 19
  - 3.3.3 Passo 3.3: determinação das funções distância sinalizada . . . 21
  - 3.3.4 Passo 3.4: determinação do passo temporal . . . 23
  - 3.3.5 Passo 3.5: metamorfose via *Level Set* . . . 25
  - 3.3.6 Passo 3.6: armazenamento em memória da fatia . . . 34
  - 3.3.7 Passo 3.7: teste de convergência . . . 37

3.3.8 Passo 3.8: arquivamento das alturas intermediárias 38
3.3.9 Passo 3.9: descentralização dos contornos 39
3.3.10 Passo 3.10: redimensionamento das fatias 41
3.3.11 Passo 3.11: empilhamento das fatias 41
3.3.12 Passo 3.12: armazenamento em arquivo das fatias 42
3.3.13 *Particle Level Set* 42
3.3.14 Saídas geradas 52
3.4 Etapa 4: Reconstrução 54
3.5 Procedimentos Otimizacionais 54

4 Resultados Experimentais **55**
4.1 Testes com Dados Sintéticos 55
4.2 Testes com Dados Reais 65

5 Conclusões e Trabalhos Futuros **73**

A Coordenadas de uma Imagem **76**

Bibliografia **77**

Lista de Tabelas

4.1 Configurações do computador. 55
4.2 Métricas dos conjuntos de fatias sintéticas. 56
4.3 Estatísticas dos testes – Pirâmide. 56
4.4 Estatísticas dos testes – Castiçal. 61
4.5 Estatísticas dos testes – Vaso. 61
4.6 Métricas dos conjuntos de fatias reais. 65
4.7 Estatísticas dos testes – Fragmento. 66
4.8 Estatísticas dos testes – Clavícula. 66
4.9 Estatísticas dos testes – Ilíaco. 66

Lista de Figuras

1.1 Expansão de uma interface retangular via *Level Set*. 4

2.1 Fronteira dinâmica no plano. 8
2.2 Evolução da interface com F de sentido positivo. 8
2.3 Evolução da interface com F de sentido arbitrário. 10

3.1 Fluxograma metodológico geral. 15
3.2 Espaço coordenado cartesiano. 16
3.3 Fluxograma metodológico da etapa de geração das fatias intermediárias. . . 18
3.4 Fatia com configuração de contorno não permitida. 19
3.5 Centralização dos contornos. 20
3.6 Dinâmica da metamorfose. 21
3.7 Detecção dos pixels internos aos contornos em uma fatia. 22
3.8 Metamorfose via *Level Set* entre as fatias origem e destino (I). 26
3.9 Metamorfose via *Level Set* entre as fatias origem e destino (II). 27
3.10 Metamorfose via *Level Set* entre as fatias origem e destino (III). 28
3.11 Metamorfose via *Level Set* entre as fatias origem e destino (IV). 29
3.12 Metamorfose via *Level Set* entre as fatias origem e destino (V). 30
3.13 Metamorfose via *Level Set* entre as fatias origem e destino (VI). 31
3.14 Metamorfose via *Level Set* entre as fatias origem e destino (VII). 32
3.15 Superfície reconstruída entre duas fatias. 39
3.16 Casos de implementação do passo de descentralização dos contornos. . . 40
3.17 Deslocamento do centroide. 41
3.18 Vértices de uma célula da grade. 43
3.19 Vetor de estruturas `particulas`. 46
3.20 Sedimentação das partículas. 47

4.1 Fatias originais – Pirâmide. 57
4.2 Evolução via *Level Set* – Pirâmide. 58
4.3 Evolução via *Particle Level Set* – Pirâmide. 59
4.4 Superfícies reconstruídas – Pirâmide. 60

4.5 Fatias originais – Castiçal. 62
4.6 Fatias originais – Vaso. 62
4.7 Superfícies reconstruídas – Castiçal. 63
4.8 Superfícies reconstruídas – Vaso. 64
4.9 Amostra das fatias originais – Fragmento. 67
4.10 Superfícies reconstruídas – Fragmento. 68
4.11 Amostra das fatias originais – Clavícula. 69
4.12 Fatias empilhadas – Clavícula. 70
4.13 Amostra das fatias originais – Ilíaco. 71
4.14 Fatias empilhadas – Ilíaco. 72

A.1 Convenção dos eixos coordenados para a representação de imagens. . . . 76

Lista de Algoritmos

3.1 Cálculo da hamiltoniana numérica $\hat{H}^n(y, x)$ via esquema de Roe-Fix 35
3.2 Sedimentação das partículas 48
3.3 Correção do contorno via *Particle Level Set* 53

Capítulo 1

Introdução

A partir de um panorama sobre as descobertas e os avanços científicos envolvendo o método *Level Set* nas últimas décadas, o presente capítulo estabelece e caracteriza o problema a ser investigado, descreve os objetivos propostos e as contribuições previstas do trabalho e apresenta, ao fim, a organização do texto.

1.1 Contextualização

Modelar a evolução de fronteiras dinâmicas sempre apresentou-se como uma tarefa desafiadora. O problema pode variar de simples interfaces evoluindo uniformemente (como um círculo se expandindo indefinidamente) a curvas aleatórias mais complexas, nas quais a intensidade da velocidade de deslocamento da curva depende do tempo, da posição da fronteira ou de um ponto em particular a esta pertencente.

Dentre as primeiras propostas para solucionar essa questão, os métodos de monitoramento de superfície discretizavam as equações geométricas lagrangianas modeladoras do movimento da interface dinâmica, depositando sobre esta partículas marcadoras, cujas posições, em qualquer instante da evolução, eram utilizadas para reconstruir a fronteira por meio de interpolação numérica [62]. Entretanto, tais métodos baseados em aproximações lagrangianas para o problema revelaram-se numericamente instáveis e incapazes de tratar corretamente mudanças na topologia da fronteira [28, 63].

As técnicas de volume de fluido apresentaram uma nova abordagem ao estudo, ao incorporarem a ótica euleriana de equações diferenciais parciais à modelagem das fronteiras dinâmicas, sendo o método SLIC o exemplo pioneiro [46]. Segundo esse trabalho, o plano era coberto por uma grade regular fixa e, à cada célula dessa grade, associava-se a fração da área da célula contida no interior da interface. Mediante equações diferenciais parciais auxiliares, a cada instante da evolução, os valores dessas frações eram atualizados e em-

pregados na aproximação da posição da fronteira no plano, sendo possível a extensão do método para fronteiras no espaço.

Apesar de lidarem com alterações topológicas de forma eficaz (capacidade esta decorrente da sua natureza euleriana), as técnicas de volume de fluido mostraram-se imprecisas, não preservando características finas da interface e com resultados satisfatórios altamente dependentes de uma grade com um elevado (e muitas vezes impraticável) número de células, entre outras desvantagens detectadas [63].

Até o final da década de 1980, vários outros estudos acerca de interfaces evolutivas ainda foram prospostos, como a utilização de grades dinâmicas e adaptativas [27, 28]. Contudo, notou-se a necessidade de um esquema mais geral e abrangente e que superasse as desvantagens observadas, tanto numéricas quanto computacionais, das técnicas até então propostas.

O método *Level Set*, proposto em meados da década de 1990 por Sethian [63], revelou-se uma ferramenta capaz de modelar robusta e eficientemente a evolução de fronteiras dinâmicas. Assim como as técnicas de volume de fluido, o método substitui a perspectiva geométrica lagrangiana pela euleriana de equações diferenciais parciais, com a primazia de empregar as aproximações de Hamilton-Jacobi [50] na resolução numérica das equações temporais modeladoras da dinâmica da interface evolutiva em estudo, o que permite a obtenção de resultados com uma acurácia significantemente aperfeiçoada.

Além desse aprimoramento na precisão, a formulação matemática do método *Level Set* possibilita o controle de mudanças topológicas da fronteira, mesmo as mais sutis, e a extensão para problemas de dimensões maiores de uma maneira mais natural [63]. Propriedades geométricas intrínsecas à interface, como a curvatura, são, inclusive, determinadas diretamente por fórmulas simples, deduzidas a partir dessa formulação.

Tamanha evolução na qualidade da modelagem das fronteiras dinâmicas propiciou ao método obter, rapidamente, sucesso em áreas além das convencionais. Em seu próprio trabalho pioneiro sobre *Level Set*, Sethian apresenta algumas aplicações do método em campos dos mais distintos, tais como Geometria Computacional, Análise Sísmica, Combustão e Ciência dos Materiais [63].

Ao longo dos últimos anos, o método *Level Set* permaneceu como uma ferramenta versátil e amplamente estudada pela comunidade científica, sendo empregado, isoladamente ou em conjunto com outras técnicas, para a solução de problemas nas áreas de Glaciologia [37, 57], Geologia [26, 29], Meteorologia [11, 16], Química [12, 13, 14], Dinâmica dos Fluidos [8, 52, 66], Eletromagnetismo [54], Pirologia [40, 59], Otimização Topológica [7, 10, 67, 71], entre diversas outras.

Sobressaindo-se, a Ciência da Computação pode ser considerada um dos campos de maior contribuição do *Level Set*, com inúmeros trabalhos descrevendo aplicações do método em várias frentes de pesquisa, com destaque às áreas de Processamento de Imagens

e de Visão Computacional. Detecção de contornos de peças defeituosas em processos industriais [41], redução de *aliasing* em superfícies binárias [70], rastreamento de formas em vídeos [35, 64] e de regiões em superfícies metamorfoseantes [4], metamorfose entre objetos tridimensionais [5, 6], simulações realísticas de líquidos [33] e reconstrução estéreo de superfícies a partir de múltiplas imagens [17, 23] são alguns exemplos dessa pluralidade.

Em particular, os cientistas da computação vêm explorando eficientemente as potencialidades do *Level Set* na segmentação de imagens. Lie et al. [36] aplicam a variante binária do método ao modelo de Mumford-Shah [44] para a segmentação de imagens digitais, enquanto De Santis e Iacoviello [61] propõem uma versão discretizada do *Level Set* em uma formulação adaptada desse mesmo modelo; Al-Qunaieer et al. [2] empregam *wavelets* no aprimoramento da velocidade de convergência da segmentação via *Level Set*; Machado et al. [39] utilizam as derivadas topológicas em uma etapa de pré-processamento das imagens a serem segmentadas pelo método; Riklin-Raviv et al. [60] associam o *Level Set* ao conceito de segmentação mútua em diferentes imagens de um mesmo objeto, para um resultado mais preciso; Fahmi e Farag [22] apresentam um algoritmo baseado no método *Level Set* que permite a segmentação de uma imagem em mais de duas regiões distintas; e Yeo et al. [73] expõem uma técnica de segmentação, fundamentada no método, mais robusta diante de imagens com ruído e menos dependente das condições iniciais.

Além desses, vários outros estudos com contribuições do método *Level Set* à segmentação de imagens foram, e continuam sendo, publicados [3, 34, 68, 74, 77], comprovando a eficiente adaptabilidade do método às mais diferentes abordagens desse assunto.

1.2 Caracterização do Problema

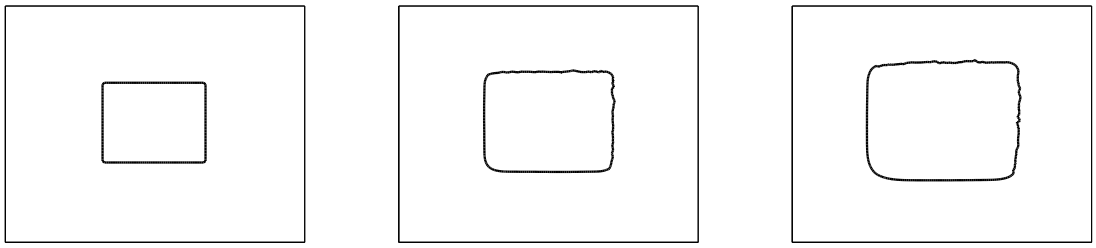
Como exposto, a aplicação do método *Level Set* como uma técnica, central ou auxiliar, na segmentação de imagens permanece em voga. Por outro lado, o emprego na reconstrução de superfícies tridimensionais tem sido relativamente pouco explorado, por parte dos cientistas, até então.

Majoritariamente, os trabalhos de pesquisa relativos a esse tema concentram-se na reconstrução de formas a partir de um conjunto de dados (pontos, curvas ou regiões) dispersos no espaço, utilizando-se o *Level Set* na deformação de uma superfície inicial até a otimalidade, com base na minimização de um funcional energético associado à superfície desenhada pelos dados dispersos [53, 72, 75, 76].

A reconstrução da superfície de um objeto tridimensional a partir de fatias planas paralelas a ele transversais constitui uma tradicional abordagem à reconstrução de superfícies, com uma relevante coleção de técnicas conhecidas para um correto tratamento do problema [55]. Entretanto, para casos em que a amostragem das fatias revela-se muito

espaçada (alto espaçamento vertical entre fatias adjacentes), não uniforme (espaçamento vertical entre fatias adjacentes variável) ou mesmo pouco fina (baixo número de fatias amostradas), o resultado da triangulação composta por essas técnicas pode retornar um objeto com uma superfície pouco suave ou, inclusive, divergente do teoricamente esperado.

Nilsson et al. [45] propõem uma solução para essa questão, empregando o método *Level Set* para gerar fatias intermediárias entre cada par de fatias originais consecutivas, por meio de metamorfose planar, a fim de preencher os espaços nos quais a amostragem de dados mostrou-se falha. Apesar de comprovadamente eficiente, o estudo não aborda, explicitamente, os casos em que nas fatias há contornos com cantos, extensões pontiagudas ou regiões de curvatura acentuada, características geométricas de uma fronteira que são suavizadas em excesso quando evoluída sob a ação do *Level Set*.



(a) Interface original; (b) n = 16; (c) n = 34;

(d) n = 61; (e) n = 125; (f) n = 249.

Figura 1.1: Expansão de uma interface retangular via *Level Set*.

A Figura 1.1 ilustra essa eventualidade para uma interface originalmente retangular, cuja forma, após a n-ésima iteração do método, é exposta em cada uma das subfiguras (b)∼(f). Dependendo da área científica à qual se direciona a reconstrução, o detrimento de características finas das fronteiras contidas nas fatias – e, consequentemente, da superfície reconstruída – firma-se como um fato relevante, pois acarreta perda de informações.

1.3 Objetivos e Contribuições

Originalmente concebido para a animação fotorrealística de líquidos [21], o método *Particle Level Set* [18, 19, 20], uma variante do original *Level Set*, apresentou-se como uma técnica robusta para a conservação de massa ou volume de interfaces em evolução dinâmica [69]. Com base nessa constatação, cogitou-se a possibilidade de a conservação impressa pelo *Particle Level Set* traduzir-se em um procedimento capaz de contornar o problema de suavização excessiva inerente ao *Level Set*, de modo a preservar as características finas da fronteira em evolução.

A proposta desta dissertação reside, portanto, na implementação do método *Particle Level Set* para a metamorfose planar entre fronteiras distintas, objetivando a geração de fatias intermediárias a complementar o conjunto original de fatias planas paralelas e transversais ao objeto a ser reconstruído, cuja amostragem apresente, porventura, alguma das singularidades citadas na Seção 1.2. O total de fatias, entre originais e intermediárias geradas, é, então, empregado na reconstrução desse objeto e a qualidade da superfície resultante, comparada à retornada pela execução de igual metodologia por meio do método *Level Set* exclusivamente.

Embasada na análise dos resultados experimentais obtidos, uma discussão sobre a viabilidade do método *Particle Level Set* para fins de reconstrução de superfícies tridimensionais, assim como a disponibilização completa do código computacional elaborado para a implementação prática, compõem a contribuição final deste trabalho.

1.4 Organização do Texto

O Capítulo 2 revisa bibliograficamente a teoria matemática na qual se fundamentam os métodos *Level Set* e *Particle Level Set* e explana a aplicação do primeiro como base da metamorfose planar. Na sequência, o Capítulo 3 detalha, etapa por etapa, a metodologia empregada para a reconstrução tridimensional, cujos resultados práticos obtidos são exibidos e analisados no Capítulo 4. O Capítulo 5 apresenta as conclusões acerca do processo de reconstrução proposto e os trabalhos futuros passíveis de serem realizados para sanar as eventuais limitações encontradas. Finaliza-se a presente dissertação com a descrição, no Apêndice A, da convenção adotada para as coordenadas das imagens.

Capítulo 2

Conceitos Teóricos

O embasamento teórico do processo de reconstrução de superfícies tridimensionais a ser exposto sustenta-se sobre o método *Level Set*. A partir de um conjunto de fatias planas paralelas e transversais ao objeto a ser reconstruído, emprega-se o método para a geração, por metamorfose planar, de fatias intermediárias entre cada par de fatias consecutivas desse conjunto, obtendo-se, assim, uma amostragem mais fina dos dados do problema e, consequentemente, um objeto recuperado com uma superfície mais suave ao final do processo, se comparada às resultantes de metodologias tradicionais de reconstrução. O método alternativo *Particle Level Set*, derivado do original *Level Set*, executa, além da rotina supracitada, uma etapa adicional de correção dos dados numéricos gerados ao longo do processo computacional de reconstrução.

O presente capítulo revisa a teoria e os conceitos utilizados para a elaboração do processo prático de reconstrução de superfícies tridimensionais. A Seção 2.1 detalha a fundamentação teórica do método *Level Set*, incluindo a descrição da metamorfose planar nele baseada, finalizando-se o capítulo com a apresentação conceitual da variante *Particle Level Set*, na Seção 2.2.

2.1 *Level Set*

Fronteiras (interfaces) dinâmicas (evolutivas) são estruturas geométricas que mudam de posição ou de forma em resposta a ações externas. É possível, por exemplo, abstrair o rastejar de uma cobra no chão e nele visualizar a evolução de uma fronteira dinâmica (a saber, o contorno do corpo da cobra) sobre uma região planar. De maneira análoga, o comportamento das chamas de fogo queimando em um incêndio florestal também pode ser visto sob a perspectiva de interfaces dinâmicas, permitindo o acompanhamento e um posterior controle da propagação das chamas, a partir de uma modelagem matematica-

mente correta das fronteiras evolutivas do problema.

Vários outros exemplos reais, do cotidiano, igualmente podem ser estudados (e suas consequências, quando houver, previstas) segundo uma abstração em fronteiras dinâmicas, tais como a deformação de uma bola rebatida por uma raquete de tênis, a quebra de uma onda na superfície do oceano ou o movimento de nuvens no céu. A aplicabilidade das interfaces dinâmicas é vasta.

Nas últimas décadas, várias técnicas foram propostas para o monitoramento de interfaces (Capítulo 1), a maioria baseada na ótica geométrica lagrangiana, pela qual a posição de uma fronteira Γ no plano, em um dado tempo t, é representada por uma curva paramétrica, isto é

$$\Gamma(t) = \{(x(t,s), y(t,s)) \in \mathbb{R}^2 \mid s \in I\} \tag{2.1}$$

sendo I um domínio unidimensional com topologia adequada, como, por exemplo, o intervalo $[0,1]$ para curvas abertas ou o círculo unitário para curvas fechadas.

Entretanto, essas técnicas manifestam algumas fragilidades, intrínsecas a esse prisma lagrangiano. Uma das mais relevantes é a incapacidade de lidarem matematicamente com mudanças na topologia da interface, ou seja, quando, por exemplo, durante a sua evolução, uma única curva se divide em duas ou mais curvas distintas, em um determinado momento.

O método *Level Set*, introduzido à comunidade científica no início dos anos 1990, conseguiu sanar essa fragilidade topológica (e outras mais) da perspectiva lagrangiana, ao substituí-la pela euleriana de equações diferenciais parciais. Com isso, afirmou-se como uma das técnicas numéricas mais robustas para a modelagem eficiente de interfaces evolutivas, sendo empregada em uma diversa gama de problemas – dos mais simples presentes no dia-a-dia aos mais complexos de áreas como a médica e a meteorológica.

2.1.1 Fundamentação teórica

A partir de um ponto de vista mais abstrato da definição apresentada no início desta seção, pode-se representar uma fronteira dinâmica por uma curva fechada que divide o plano em duas regiões distintas, uma interna e outra externa à curva, e se move a uma velocidade F geralmente variável, conforme o ponto no plano e o tempo da evolução (Figura 2.1).

O desafio matemático e computacional presente no estudo de interfaces dinâmicas reside no problema de rastrear o movimento da fronteira, à medida que ela evolui. O modelo da velocidade F é um dado muitas vezes desconhecido no início do problema e a sua modelagem, um exercício tão ou mais difícil do que uma própria esquematização precisa da evolução da fronteira. Mesmo conhecendo-se as formulações da velocidade F e da direção do movimento e compreendendo-se o comportamento da fronteira sob a ação daquelas,

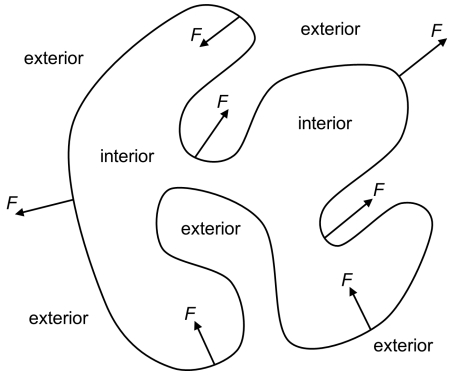


Figura 2.1: Fronteira dinâmica no plano.

predizer a posição e a forma da interface em um tempo t específico pode representar uma tarefa com um grau de dificuldade muito elevado, dependendo da aplicação prática. Frente a essa conjuntura, o objetivo, logo, é formular um esquema de evolução da interface no qual o modelo da velocidade F e a posição inicial da fronteira são dados.

Assume-se, neste primeiro momento, que o sentido da velocidade F seja positivo sobre toda a curva da fronteira, isto é, a fronteira está em constante expansão. Desse modo, a interface cruza cada ponto (x, y) do plano, no máximo, uma única vez. Pode-se, portanto, modelar a posição da fronteira em expansão pela função $T(x, y)$, que retorna o tempo t em que a fronteira cruza o ponto (x, y) do plano, definição segundo a qual, na Figura 2.2, tem-se $T(x_0, y_0) = 0$ e $T(x_1, y_1) = 1$.

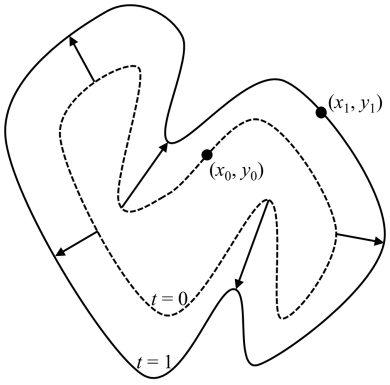


Figura 2.2: Evolução da interface com F de sentido positivo.

A derivação da expressão de $T(x, y)$ parte da relação da física cinemática para a velocidade v de um objeto

$$v = \frac{\Delta S}{\Delta t} \tag{2.2}$$

em que ΔS é o espaço percorrido pelo objeto no intervalo de tempo Δt. Sendo a velocidade F normal à fronteira um parâmetro conhecido, então, sob variações infinitesimais

$$F = \frac{dx}{dT} \Rightarrow F \cdot \frac{dT}{dx} = 1 \tag{2.3}$$

para o caso unidimensional.

Para dimensões maiores do problema (em particular, para a função bidimensional $T(x, y)$ em questão), sabe-se que o gradiente $\vec{\nabla} T(x, y)$ de $T(x, y)$ é ortogonal às curvas de nível da função e o seu módulo, inversamente proporcional à velocidade F, de modo que

$$|\vec{\nabla} T| \cdot F = 1, \quad T = 0 \text{ em } \Gamma_0 \tag{2.4}$$

sendo Γ_0 a posição inicial da interface.

A partir da Equação 2.4, obtém-se, assim, um problema de valor de contorno (PVC) como uma primeira formulação para o problema de evolução de uma fronteira dinâmica sob a ótica euleriana

$$\begin{gathered} |\vec{\nabla} T| \cdot F = 1 \\ \text{Fronteira} = \Gamma(t) = \{(x, y) \in \mathbb{R}^2 \mid T(x, y) = t\} \\ F \text{ de sentido positivo, } t > 0 \end{gathered} \tag{2.5}$$

No entanto, a restrição ao sentido de F presente nessa formulação é limitante, não permitindo a modelagem matematicamente correta da evolução de interfaces com velocidade F de sentido arbitrário, isto é, nem estritamente positivo, nem estritamente negativo. Em outras palavras, a formulação PVC não é capaz de lidar com fronteiras que podem tanto se expandir quanto se contrair, dependendo do tempo t do problema e do ponto (x, y) do plano o sentido desse movimento.

A Figura 2.3 ilustra esse contexto. O ponto (x_0, y_0) do plano é visitado pela interface, ao longo da evolução desta, em dois instantes distintos, em $t = 0$ e $t = 2$ (podendo, ainda, ser revisitado em instantes posteriores). Logo, $T(x, y)$ não mais representa uma função (no sentido matemático da palavra), pois um mesmo ponto do domínio $\mathbb{R}^2$ do problema possui duas imagens distintas e a perspectiva PVC, portanto, não mais se aplica.

Essa limitação é contornada formulando-se uma função tridimensional ϕ, de modo que, à medida que a superfície representada por ϕ evolui no espaço, a evolução das curvas de nível 0 de ϕ delinea, implicitamente, a propagação da fronteira Γ no plano. Equivalentemente, para qualquer instante t da evolução, a interface $\Gamma(t)$ é dada pela curva de nível 0 da função ϕ de maneira implícita

$$\Gamma(t) = \{(x, y) \in \mathbb{R}^2 \mid \phi(x, y, t) = 0\}, \forall t > 0 \tag{2.6}$$

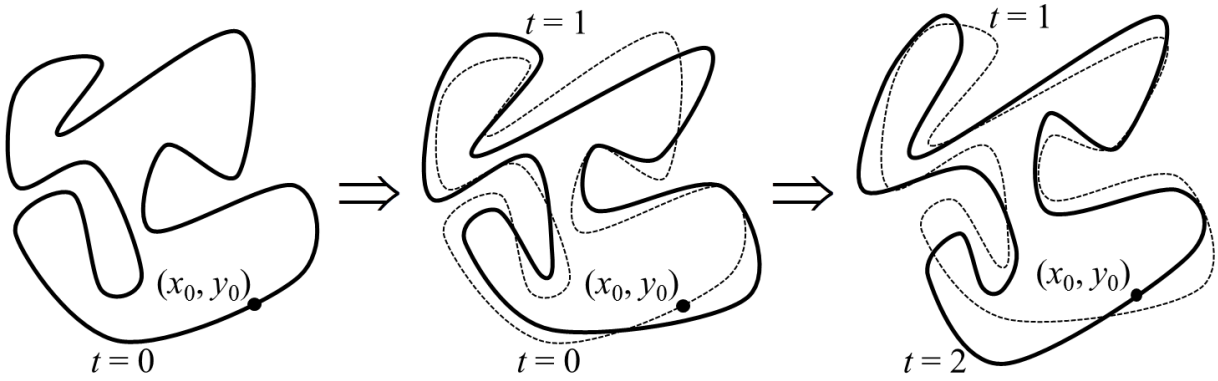


Figura 2.3: Evolução da interface com F de sentido arbitrário.

Aplicando a regra da cadeia e outras propriedades do cálculo vetorial [25] à Equação 2.6, obtém-se a Equação 2.7, conhecida como Equação *Level Set*

$$\phi_t + F \cdot |\vec{\nabla}\phi| = 0 \tag{2.7}$$

sendo ϕ_t a derivada parcial de ϕ com relação ao tempo t e $|\vec{\nabla}\phi|$ o módulo do gradiente de ϕ, e, enfim, o problema de valor inicial (PVI) que modela, sob a ótica euleriana, a propagação de fronteiras com velocidade F de sentido arbitrário

$$\begin{gathered} \phi_t + F \cdot |\vec{\nabla}\phi| = 0 \\ \text{Fronteira} = \Gamma(t) = \{(x,y) \in \mathbb{R}^2 \,|\, \phi(x,y,t) = 0\} \\ F \text{ de sentido arbitrário, } t > 0 \end{gathered} \tag{2.8}$$

Comparadas a outras formulações matemáticas para o problema de interfaces evolutivas, as perspectivas PVC e PVI apresentam duas principais vantagens: são diretamente adaptáveis para problemas dimensionalmente maiores – ou seja, (hiper)superfícies se propagando em três ou mais dimensões – e lidam naturalmente com mudanças topológicas da fronteira.

2.1.2 Funções distância sinalizada

Conforme visto, um problema de evolução de fronteiras dinâmicas, formulado sob a ótica euleriana do método *Level Set*, pressupõe que o modelo da velocidade F e a posição inicial $\Gamma(0)$ da interface são informações de entrada conhecidas. Essa afirmação se estende à função tridimensional a desempenhar o papel de ϕ durante a evolução, sendo necessárias, portanto, a modelagem e a determinação da função ϕ durante o estágio de pré-processamento dos dados do problema.

A função ϕ, por implicitamente representar a interface pela curva de nível 0 (Equação 2.6), possui a propriedade de determinar em qual região da interface um ponto (x, y)

qualquer do plano se localiza, simplesmente avaliando-se o valor de ϕ para esse ponto: se, para qualquer tempo t da evolução, $\phi(x, y, t)$ for positivo, o ponto situa-se na região externa à fronteira; para $\phi(x, y, t)$ negativo, o ponto encontra-se no interior da fronteira; e, por fim, caso $\phi(x, y, t)$ seja nulo, (x, y) é um ponto do plano pertencente à fronteira. Matematicamente, enuncia-se esse critério de classificação como

$$\begin{cases} \phi(x, y, t) > 0, & \text{se } (x, y) \notin \Omega \cup \Gamma \\ \phi(x, y, t) = 0, & \text{se } (x, y) \in \Gamma \\ \phi(x, y, t) < 0, & \text{se } (x, y) \in \Omega \end{cases} \tag{2.9}$$

sendo Γ a representação matemática da interface e Ω, da sua região interna.

Qualquer função tridimensional que satisfaça tais condições é candidata a exercer o papel da função ϕ. No entanto, a determinação de ϕ demanda uma maior atenção aos detalhes, uma vez que uma função que apresente gradientes excessivamente acentuados (ou, em um outro extremo, excessivamente planos), demasiadas oscilações ou propriedades que se alteram drástica e rapidamente, de acordo com variações no tempo e no espaço, são capazes de afetar a acurácia e a estabilidade dos resultados da evolução. Em outras palavras, a satisfação da Equação 2.9 é uma condição necessária, porém, não suficiente para uma dada função tridimensional ser estabelecida como a função ϕ do problema.

Exemplificando como uma escolha imprudente para ϕ influi na qualidade dos resultados, seja um ponto (x, y) do plano não pertencente à grade discretizada da função ϕ, em um determinado tempo t. Nesse caso, $\phi(x, y, t)$ é obtido via interpolação numérica dos valores de ϕ sobre os vértices da grade. Entretanto, como todo método numérico, a interpolação retorna estimativas com probabilidade não nula de manifestarem erros. Consequentemente, pontos pertencentes ao exterior da fronteira são passíveis de serem erroneamente classificados como pontos internos, e vice-versa, e a posição exata da fronteira em si pode, em decorrência disso, ser perturbada, divergindo do resultado esperado.

Além disso, os erros ocorrentes nas avaliações da função ϕ sobre os pontos do plano são propagados no cálculo das características e propriedades geométricas que dependem pontualmente dos valores de ϕ, tais como as derivadas parciais, o gradiente $\vec{\nabla}\phi$ e o vetor unitário normal $\vec{N}$, este definido como

$$\vec{N} = \frac{\vec{\nabla}\phi}{|\vec{\nabla}\phi|} \tag{2.10}$$

sendo as quantidades citadas incorretamente determinadas, por conseguinte, dentro desse contexto.

Para uma função ϕ suficientemente suave e com uma taxa de amostragem regularmente uniforme sobre toda a grade, os resultados numéricos do problema permanecem

estáveis, mesmo na (eventual) presença de erros nos dados interpolados, e a incorretude nas saídas das avaliações supracitadas é satisfatoriamente contornada, como requerido. Caso contrário, esses erros podem assumir magnitudes consideravelmente altas, tornando a resolução do problema impraticável.

Desde o início dos estudos acerca do método *Level Set*, as funções distância sinalizada (tradução livre do termo original *signed distance functions*) mostraram-se a escolha ideal para a representação implícita de interfaces por meio da função tridimensional ϕ [49, 63]. Enunciada segundo a lei

$$\begin{cases} \phi(x,y,t) = d_\Gamma(x,y), & \text{se } (x,y) \notin \Omega \cup \Gamma \\ \phi(x,y,t) = 0, & \text{se } (x,y) \in \Gamma \\ \phi(x,y,t) = -d_\Gamma(x,y), & \text{se } (x,y) \in \Omega \end{cases} \tag{2.11}$$

em que $d_\Gamma(x,y)$ retorna a distância euclidiana de um ponto (x,y) qualquer do plano ao ponto $P \in \Gamma$ mais próximo àquele, as funções distância sinalizada experimentam, em geral por toda a extensão da grade, o mencionado elevado nível de suavidade exigido para uma evolução consistente e apurada das fronteiras dinâmicas. Adicionalmente, a diferenciação numérica sobre a interface implicitamente representada por uma função distância sinalizada revela-se altamente precisa [49], possuindo a função a propriedade suplementar de o módulo do gradiente ser unitário, isto é

$$|\vec{\nabla}\phi(x,y,t)| = 1 \tag{2.12}$$

para qualquer ponto (x,y) do plano (exceto, possivelmente, nas regiões côncavas suavizadas da interface), o que simplifica o cálculo de diversas características geométricas possivelmente utilizadas durante a resolução numérica de um problema de evolução de interfaces evolutivas – o vetor unitário normal da Equação 2.10, por exemplo, passa a ser determinado simplesmente calculando-se $\vec{N} = \vec{\nabla}\phi$.

2.1.3 Metamorfose planar via *Level Set*

As fatias planas paralelas do processo de reconstrução consistem em seções transversais da superfície a ser reconstruída. Em alguns casos, é possível que a amostragem das fatias seja muito espaçada, não uniforme ou mesmo pouco fina (Capítulo 1). Como consequência, o objeto reconstruído pode apresentar uma superfície com continuidade C^0, ou seja, com pouca suavidade.

Para apurar a amostragem dos dados do problema e, com isso, obter-se uma superfície reconstruída mais suave, o processo de reconstrução proposto emprega a metamorfose planar entre cada par de fatias originais consecutivas, de maneira que, ao final desse procedimento, compõe-se um conjunto total de fatias, entre originais e intermediárias, capaz de gerar o objeto requerido com uma superfície mais acentuadamente suave.

A metamorfose planar consiste em um procedimento no qual um contorno no plano se deforma de modo contínuo e suave, objetivando maximizar a sua similaridade com relação a um outro contorno. No contexto do processo de reconstrução, equivale-se afirmar que o contorno presente em uma das fatias originais do problema deforma-se continuamente, até assumir, por completo, a forma do contorno na fatia imediatamente adjacente. Para cada par de fatias metamorfoseadas, a fatia contendo o contorno sob deformação e a sua fatia adjacente são denominadas, respectivamente, fatia origem e fatia destino – o procedimento se mantém para os casos em que há dois ou mais contornos distintos nas fatias.

Executar, ou ainda, automatizar a metamorfose planar é o papel do método *Level Set* dentro do processo de reconstrução a ser proposto. A partir das respectivas funções distância sinalizada ϕ_{origem} e $\phi_{destino}$ das fatias origem e destino (ou seja, deste ponto em diante da metamorfose, as fatias origem e destino originais são representadas, implicitamente, pelas curvas de nível 0 de ϕ_{origem} e $\phi_{destino}$ para $t = 0$), resolve-se, iterativamente, o PVI [45]

$$\begin{cases} \phi_t^{n+1} + (\phi^n - \phi_{destino}^n) \cdot |\vec{\nabla}\phi^n| = 0 \\ \phi^0 = \phi_{origem}^0 \end{cases} \tag{2.13}$$

obtido adaptando-se a Equação 2.8 às particularidades do procedimento de metamorfose planar, sendo $\phi^n = \phi_{origem}^n = \phi_{origem}(x, y, n)$ a função distância sinalizada definida para cada ponto (x, y) da grade e que representa, igualmente de forma implícita, a fatia intermediária gerada ao final da n-ésima iteração da metamorfose da fatia origem e $\phi_{destino}^n = \phi_{destino}(x, y, n)$ constante durante toda a evolução, isto é, $\phi_{destino}^n = \phi_{destino}^0$, $\forall n \geq 0$.

Segundo o PVI, portanto, a fatia origem é deformada conforme a perspectiva euleriana do *Level Set*, sob a ação de uma velocidade $F = \phi^n - \phi_{destino}^n = \phi^n - \phi_{destino}^0$ definida para cada ponto da grade e quantificadora da proximidade entre a n-ésima fatia intermediária e a fatia destino, de modo que as regiões de ϕ^n geometricamente mais distantes de $\phi_{destino}^0$ se movam mais rapidamente em direção a esta, se comparadas às regiões mais próximas. Ainda, essa expressão para F assegura o deslocamento de cada ponto do contorno na n-ésima fatia intermediária na respectiva direção normal e com uma velocidade proporcional ao valor de $\phi_{destino}^0$ naquele ponto.

Resolve-se a Equação 2.13 para cada par de fatias originais. O conjunto dessas fatias originais e das intermediárias geradas promove uma amostragem mais fina dos dados do problema, como objetivado, e, a partir desse conjunto, a superfície requerida é, então, reconstruída.

2.2 *Particle Level Set*

Apesar da versatilidade na resolução de diversos problemas científico-computacionais, o método *Level Set* apresenta uma considerável suscetibilidade à presença de dissipa-

ção numérica em seus resultados [43, 69], quando em ambientes nas áreas de Computação Gráfica e de Processamento de Imagens, decorrente, majoritariamente, do emprego de grades de baixa resolução, necessárias para uma evolução computacionalmente viável. Consequentemente, características geométricas da fronteira, como cantos, extensões pontiagudas e regiões de curvatura acentuada, são suavidas em demasia, à medida que o método evolui.

As soluções de início propostas para a minimização da dissipação numérica observada focavam na acurácia do método *Level Set*, isto é, objetivavam aumentar a precisão dos métodos numéricos envolvidos na evolução do PVI [43]. Entretanto, esse aumento na precisão ocorria, em contrapartida, sob um elevado custo no tempo computacional e na utilização de memória.

O método híbrido *Particle Level Set* introduz partículas pontuais lagrangianas à perspectiva euleriana do método original *Level Set* para rastrear a posição da interface ao longo da resolução do problema, utilizando-as para a correção de eventuais erros causados por dissipação numérica. Originalmente desenvolvido para a renderização fotorrealística de água e outros líquidos em movimento em animações computacionais [21], a variante *Particle Level Set* mostrou-se igualmente robusta para a conservação de massa ou volume de fronteiras em evolução [69], presumindo-se, a partir dessa competência constatada, uma eventual capacidade do *Particle Level Set* para o controle da suavização excessiva intrínseca ao *Level Set*.

Em uma rotina do método *Particle Level Set*, as partículas lagrangianas são aleatoriamente sedimentadas, em uma banda ao redor da interface, e passivamente evoluídas, conforme os valores interpolados da velocidade F sobre os vértices da grade mais próximos a cada partícula. A eventual localização, na região externa da fronteira, de partículas inicialmente depositadas na região interna (e vice-versa), após o passo citado de evolução das partículas, sinaliza a ocorrência de erros na posição da interface, decorrentes de dissipação numérica. Tais partículas, ditas escapadas, são, então, empregadas na reconstrução das regiões da fronteira afetadas por esses erros numéricos, assegurando, assim, a preservação das características finas da interface.

Capítulo 3

Metodologia e Implementação

O processo proposto para reconstrução de uma superfície tridimensional a partir de fatias planas paralelas, utilizando o método (*Particle*) *Level Set*, inicia-se com a leitura dos dados de entrada, a qual armazena, a cada iteração, as informações necessárias das duas fatias a desempenharem os papéis de fatia origem e fatia destino, durante a geração das intermediárias entre elas. Completado esse procedimento, todas as fatias originais e intermediárias são empilhadas, reconstruindo-se, por fim, a superfície tridimensional desejada.

A Figura 3.1 traduz a metodologia supracitada em um fluxograma de alto nível, no qual as caixas destacadas representam as saídas do processo. Cada etapa da metodologia, a ser mais amplamente detalhada nas seções subsequentes, foi implementada em ambiente MATLAB (versão R2011b) [65], à exceção da etapa final de reconstrução, escrita em linguagem Python [58].

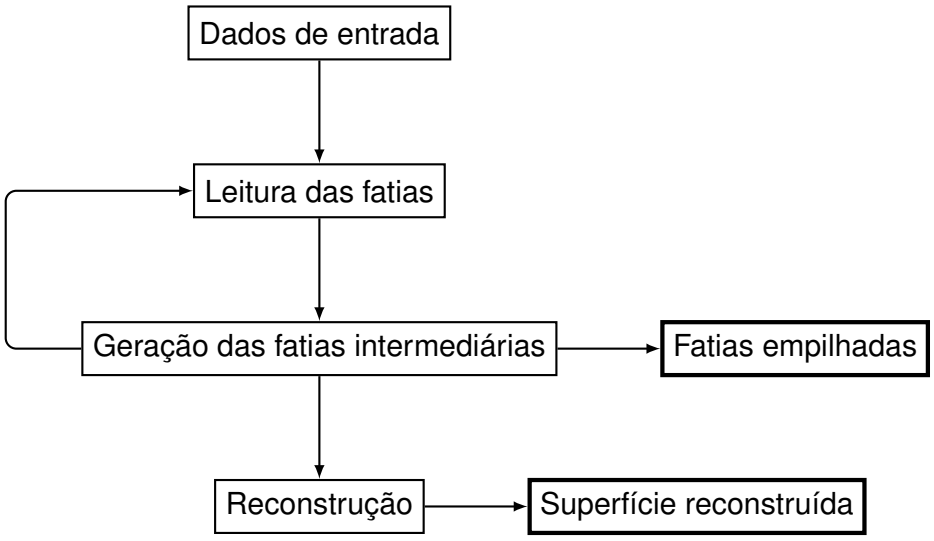


Figura 3.1: Fluxograma metodológico geral.

3.1 Etapa 1: Dados de Entrada

Três dados compõem a entrada do processo de reconstrução. O primeiro informa o diretório no qual estão armazenados os arquivos de extensão *bitmap* monocromático contendo as fatias originais transversais ao objeto a ser reconstruído, de modo que elementos matriciais de valor 0 (branco) representam pixels do fundo da imagem e de valor 1 (preto), pixels pertencentes aos contornos, de bordas finas 8-conexas, presentes nas fatias. Para fins de clareza e otimização de código, os nomes dos arquivos são enumerados em ordem crescente, a partir da base até o topo do objeto – o arquivo da n-ésima fatia original recebe, logo, o nome `n.bmp`.

As alturas das fatias originais constituem o segundo dado de entrada e são referenciadas ao plano $z = 0$ do espaço coordenado cartesiano (Figura 3.2): a distância ortogonal de cada fatia ao plano $z = 0$ a ela paralelo define a sua altura, sendo que, por padrão, a primeira fatia `1.bmp` (isto é, a base do objeto) coincide com o plano $z = 0$ e, portanto, possui altura igual a 0.

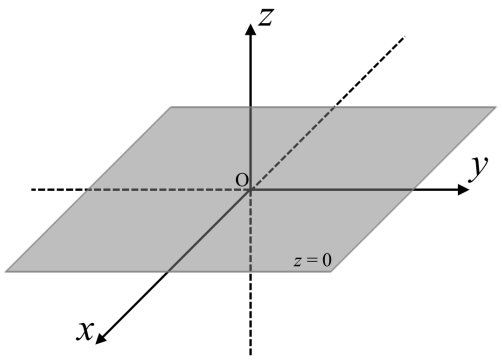


Figura 3.2: Espaço coordenado cartesiano.

O terceiro e último dado de entrada do processo informa o critério de parada da evolução do método (*Particle*) *Level Set*, a ser mais bem discutido na Subseção 3.3.7.

O mapeamento entre os contornos presentes em cada par de fatias originais consecutivas, de modo a informar ao processo a correspondência correta dentre todas as possíveis, evidencia-se como um admissível dado de entrada adicional, não empregado pela metodologia apresentada, porém de eventual utilidade, conforme as necessidades do usuário.

3.2 Etapa 2: Leitura das Fatias

A leitura dos arquivos armazenadores das fatias originais é realizada iterativamente ao longo do processo de reconstrução. Assim, para a i-ésima iteração, leem-se as fa-

tias `i.bmp` e `i+1.bmp` do conjunto, sendo a primeira a desempenhar o papel de fatia origem e a segunda, o de fatia destino, no procedimento de metamorfose para a geração das fatias intermediárias entre elas.

Ler os arquivos das fatias iterativamente garante um melhor aproveitamento da memória do sistema, uma vez que se demandam, durante toda a etapa de geração das fatias intermediárias, apenas duas matrizes para o armazenamento – uma para a fatia origem e outra para a fatia destino. Caso a totalidade dos k arquivos fosse processada logo no início do processo de reconstrução, seria necessária uma quantidade de matrizes da ordem de $O(k)$, o que poderia se tornar algo impraticável, dependendo do valor de k e das dimensões de cada fatia.

3.3 Etapa 3: Geração das Fatias Intermediárias

O núcleo computacional de todo o processo de reconstrução concentra-se nesta etapa, na qual são geradas as fatias intermediárias entre cada par de fatias originais consecutivas. Tal importância exige, logo, um minuciamento da etapa, para uma melhor compreensão do processo de reconstrução como um todo.

A Figura 3.3 apresenta o fluxograma da metodologia empregada na geração das fatias intermediárias. Os caminhos em linha cheia representam o fluxo original da etapa; as linhas tracejadas ilustram os caminhos adicionados aos originais, na ocasião de a geração empregar a variante *Particle Level Set* para a correção numérica das fatias geradas. Nas subseções seguintes, cada um dos passos dessa metodologia é explanado.

3.3.1 Passo 3.1: redimensionamento das fatias

Neste passo inicial, realiza-se um pré-processamento das fatias origem e destino, redimensionando ambas a dimensões quadradas, caso a altura e a largura das fatias não sejam iguais.

Para tanto, as matrizes que armazenam os dados das fatias são expandidas na direção da dimensão de menor magnitude. Exemplificando, para fatias de dimensões altura×largura não uniformes 20×30, o passo em questão executa um redimensionamento dessas fatias na direção da altura, adicionando linhas de zeros na parte inferior da matriz até esta atingir a dimensão quadrada 30×30. Para o caso inverso de a magnitude da largura ser menor do que a da altura (por exemplo, 50×40), colunas de zeros são adicionadas à direita da matriz.

Adicionar linhas ou colunas de zeros às matrizes das fatias não corrompe quaisquer informações nelas armazenadas, pois os contornos presentes nas fatias são sempre curvas fechadas (condição de entrada do método *Level Set*) integralmente contidas nas fatias,

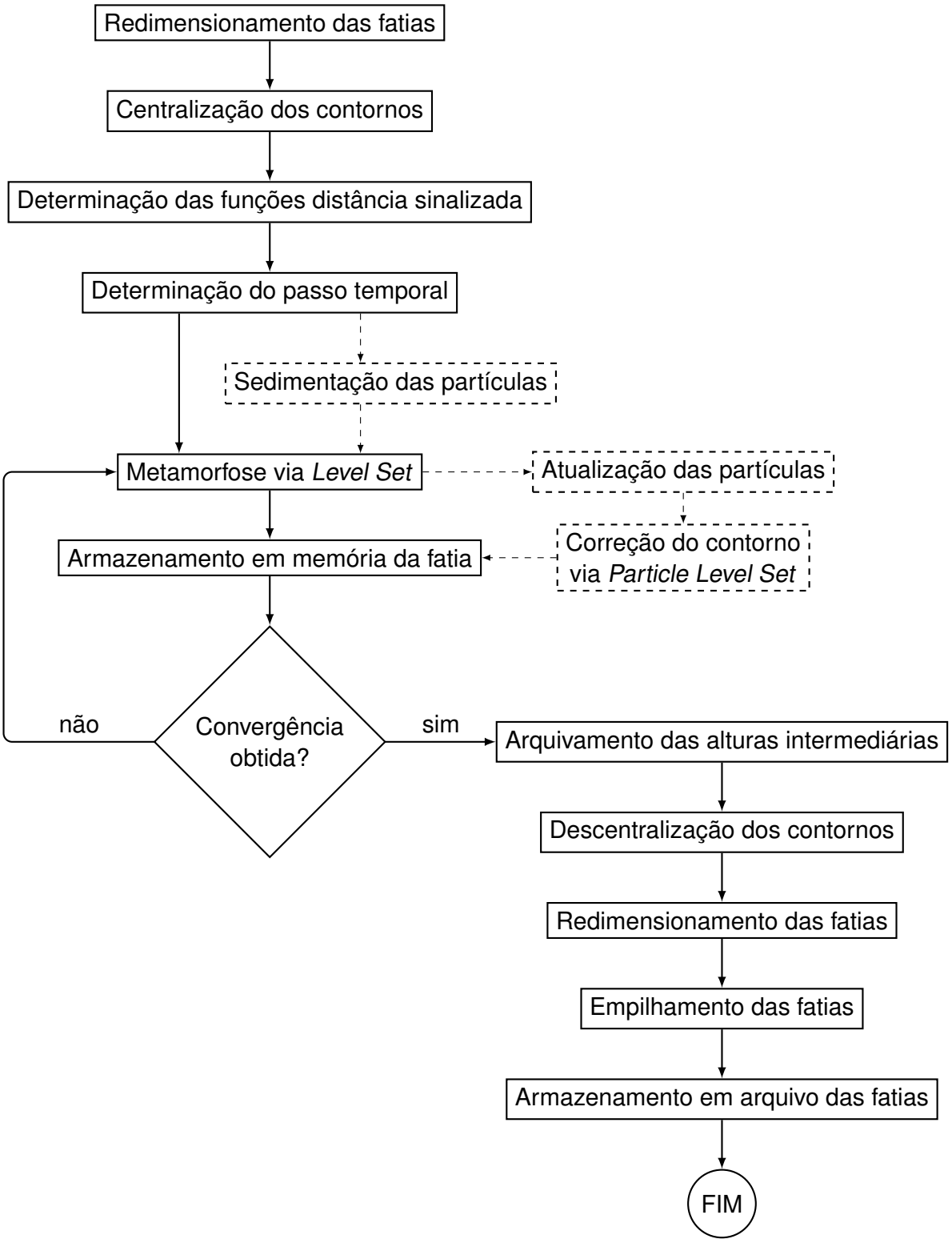


Figura 3.3: Fluxograma metodológico da etapa de geração das fatias intermediárias.

não havendo a possibilidade de configurações como a retratada na Figura 3.4 (em que as áreas cheias na cor preta representam os interiores dos contornos); nestas, a adição de linhas/colunas de zeros, certamente, afetaria a lógica das informações sobre os contornos nas matrizes contidas.

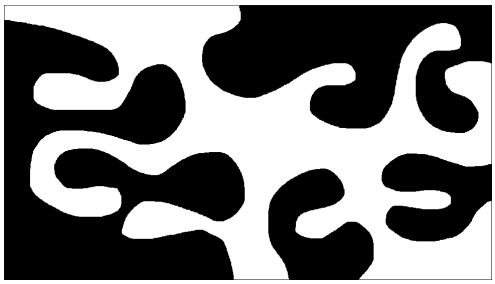


Figura 3.4: Fatia com configuração de contorno não permitida.

Este passo faz-se necessário para viabilizar o próximo de centralização dos contornos; caso o redimensionamento não fosse implementado, regiões dos contornos, eventualmente, ultrapassariam as dimensões da fatia após a translação executada pela centralização.

3.3.2 Passo 3.2: centralização dos contornos

Prosseguindo o estágio de pré-processamento, neste passo, os contornos presentes nas fatias origem e destino são centralizados geometricamente em suas respectivas fatias, a fim de haver a sobreposição dos seus centroides.

Exemplificando, na Figura 3.5, a primeira linha ilustra a centralização de um contorno em uma fatia origem e a segunda linha, a de um contorno em uma fatia destino (o centro geométrico da fatia é representado pelo símbolo × e os centroides dos contornos, pelo círculo cheio). Observa-se que, ao final do procedimento, os centroides de ambos os contornos estão sobrepostos, isto é, possuem as mesmas coordenadas matriciais.

A completude do passo dá-se com a execução de um bloco de comandos relativamente simples, para cada fatia. Inicialmente, calcula-se o elemento central da matriz da fatia e, na sequência, determinam-se o fecho convexo do contorno nela presente e o centroide desse fecho, por meio, respectivamente, das funções `bwconvhull` e `regionprops` pertencentes à biblioteca do MATLAB. A diferença entre as coordenadas do elemento central da fatia e as do centroide do contorno, nesta ordem, retorna, por fim, a distância a ser transladada pelos pixels do contorno para a conclusão do passo, o sinal do deslocamento obtido respeitando o sentido convencionado para os eixos coordenados, conforme a Figura A.1.

Na ocasião de a fatia possuir mais de um contorno distinto, o fecho convexo é aplicado a todos os contornos de uma forma global, e não a cada um em particular, pois, neste

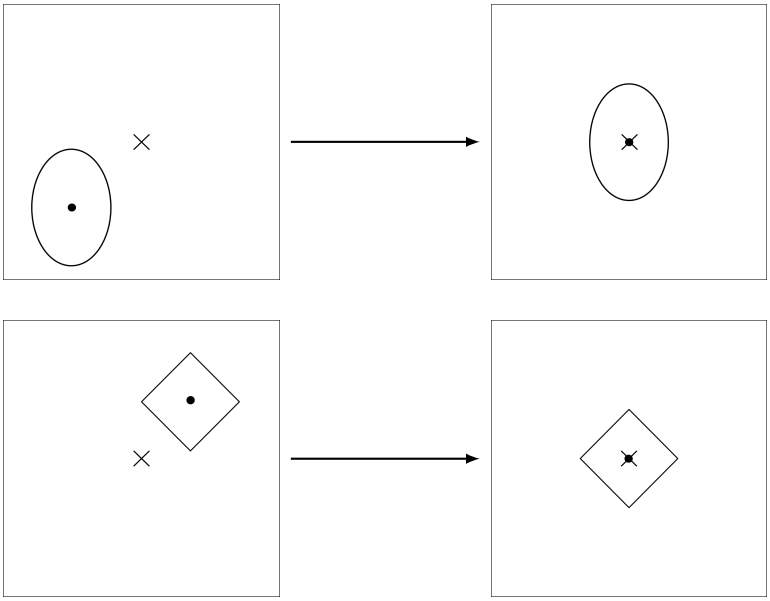


Figura 3.5: Centralização dos contornos.

caso, o centro de massa do conjunto de contornos como um todo constitui o centroide de interesse para o procedimento supracitado; o restante deste mantém-se inalterado – deste ponto em diante do texto, toda a descrição metodológica a ser apresentada considera fatias contendo um contorno unicamente; a extensão a casos com múltiplos contornos é direta e indistinta, sendo destacada a distinção quando existente.

Ao final da execução do passo, há a possibilidade de o contorno não estar precisamente centralizado na fatia, em termos geométricos. Tal fato é uma consequência de as fatias, por serem imagens binárias, estarem representadas em coordenadas inteiras de pixels (ou matriciais), e não em coordenadas reais cartesianas, e ocorre quando, no mínimo, uma das dimensões da fatia ou do retângulo envolvente do contorno for par ou quando ao menos uma das dimensões do retângulo envolvente do contorno possuir paridade diferente da dimensão relativa da fatia.

Entretanto, essa aparente adversidade não prejudica a corretude do presente passo ou do processo de reconstrução em seu todo, uma vez que o objetivo primordial da centralização dos contornos é fazer com que os centroides dos fechos convexos dos contornos presentes nas fatias origem e destino se sobreponham, o que é assegurado pela condição de entrada de que ambas as fatias possuem as mesmas dimensões e, portanto, o mesmo elemento central. Ademais, para dimensões maiores das fatias, essa discrepância torna-se, praticamente, imperceptível e, de certa forma, irrelevante.

A centralização dos contornos garante não somente uma completa automatização da metamorfose entre as fatias [6], mas, principalmente, a evolução, pelo método *Level Set*,

apenas das regiões do contorno da fatia origem que não estão perfeitamente sobrepostas a quaisquer regiões do contorno da fatia destino. A Figura 3.6 retrata essa evolução: os pontos do contorno da fatia origem (curva cheia) que se encontram no interior do contorno da fatia destino (curva tracejada) se expandem, na direção normal; os pontos do contorno da fatia origem no exterior do contorno da fatia destino se contraem, também na direção normal; e os pontos de intersecção entre os dois contornos permanecem imóveis. Essa distinção fundamental entre interior e exterior somente é possível com os contornos centralizados.

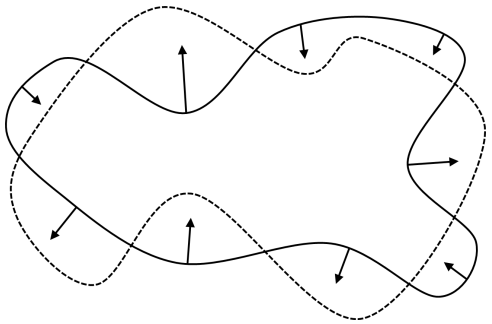


Figura 3.6: Dinâmica da metamorfose.

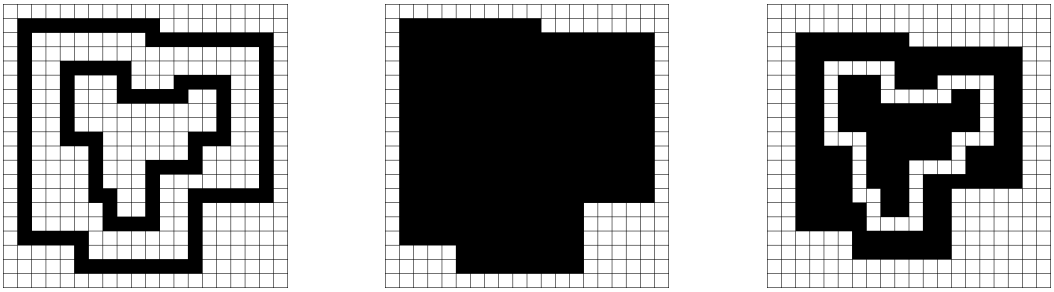
3.3.3 Passo 3.3: determinação das funções distância sinalizada

Pré-processadas as fatias origem e destino, inicia-se a execução do método *Level Set* com a determinação das funções distância sinalizada a desempenharem os papéis das funções ϕ_{origem} e $\phi_{destino}$ ao longo de toda a etapa de geração das fatias intermediárias.

Como explanado no Capítulo 2, as funções distância sinalizada são funções implícitas em geral suaves por toda a extensão da grade, sendo a suavidade uma condição desejável para que a resolução de equações diferenciais parciais (como a Equação *Level Set* 2.7), por métodos numéricos, ocorra com uma boa precisão, o que justifica a escolha.

A computação da função distância sinalizada de uma fatia inicia-se com o cálculo, para cada pixel da fatia, da distância euclidiana entre este pixel e o pixel não nulo a ele mais próximo. Essa transformação é obtida diretamente pela função `bwdist` pertencente à biblioteca do MATLAB e retornada em uma matriz de mesmas dimensões da fatia.

Para determinar o sinal das distâncias calculadas, segundo a convenção apresentada pela Equação 2.11, uma rotina mais complexa é necessária. Primeiramente, chama-se a função `imfill` (com o parâmetro de entrada adicional `'holes'`), também pertencente à biblioteca do MATLAB, a qual preenche a região interior do contorno presente na fatia. Do retorno dessa função, subtrai-se a fatia original, operação esta que gera uma imagem contendo apenas os pixels internos ao contorno. A Figura 3.7 ilustra esse procedimento.



(a) `I1` (fatia original); (b) `I2 = imfill(I1,'holes');` (c) `I2-I1`.

Figura 3.7: Detecção dos pixels internos aos contornos em uma fatia.

Bastaria, na sequência, para cada um dos pixels assinalados na imagem retornada, atribuir o sinal negativo ao elemento correspondente na matriz de retorno da função `bwdist`, uma vez que esses pixels representariam a região interna ao contorno presente na fatia. Porém, quando há contornos dentro de contornos, essa atribuição direta apresentaria falhas.

A fatia original da Figura 3.7(a) exemplifica casos desse tipo, pois possui dois contornos, sendo o menor completamente interno ao maior. Sob ótica diferente, o contorno maior representa a borda externa de um objeto e o contorno menor, a borda interna, de modo que, globalmente, os contornos formam um objeto com um buraco em seu interior. Assim sendo, se, para cada um dos pixels assinalados na Figura 3.7(c), fosse atribuído o sinal negativo ao elemento correspondente na matriz de retorno da função `bwdist` indistintamente, a região interna ao contorno menor seria considerada, erroneamente, como pertencente ao interior do objeto, e não ao fundo da imagem, conforme o correto.

Para contornar essa questão, percorre-se, da esquerda para a direita e de cima para baixo, a imagem retornada pelo procedimento de preenchimento explanado (Figura 3.7(c), no caso do exemplo exposto). Para cada pixel assinalado (ou, em outras palavras, para cada pixel de valor 1 encontrado), obtém-se a máscara 3×3 formada com o pixel analisado na última posição (isto é, na posição (3,3) da máscara) e verifica-se se algum dos elementos dessa máscara possui correspondente, na matriz de retorno da função `bwdist`, com distância negativa. Essa verificação procura determinar se o pixel em análise está localizado nas adjacências de regiões com distâncias negativas.

Em caso negativo (ou seja, se todos os correspondentes aos elementos da máscara apresentarem distâncias positivas ou nulas), testa-se, adicionalmente, se os vizinhos acima e à esquerda do pixel analisado pertencem a algum contorno (ou, equivalentemente, se possuem distância nula). Se ambas as comparações forem satisfeitas pelo pixel, atribui-se o sinal negativo à sua distância correspondente na matriz de retorno da função `bwdist`, uma vez que conclui-se que o pixel em análise encontra-se adjacente a algum contorno,

o qual o separa da região externa a esse contorno, isto é, de uma região com distâncias positivas.

Essas três condições avaliadas – ausência de elementos, dentro da máscara 3×3, com distâncias negativas e vizinhos acima e à direita com distâncias nulas – precisam ser satisfeitas simultaneamente. Na hipótese de alguma dessas condições ser falsa, averigua-se, alternativamente, se os vizinhos acima ou à esquerda do pixel analisado possuem, então, distâncias correspondentes negativas. A satisfação de ao menos uma dessas duas últimas condições indica que o pixel em análise é adjacente a um pixel interno ao contorno e, portanto, localiza-se na região interna do contorno igualmente, o que implica a atribuição do sinal negativo à sua distância correspondente.

A sequência de verificações apresentada assegura a corretude do procedimento de atribuição do sinal negativo como um todo, e não somente para os casos supracitados em que há contornos dentro de contornos, independentemente da quantidade daqueles, e a sua execução na sequência da chamada da função `bwdist`, conclui, enfim, a determinação das funções distância sinalizada das fatias origem e destino.

3.3.4 Passo 3.4: determinação do passo temporal

Uma equação do tipo Hamilton-Jacobi possui forma geral

$$\phi_t + H(\vec{\nabla}\phi) = 0 \tag{3.1}$$

ou, equivalentemente

$$\phi_t + H(\phi_x, \phi_y) = 0 \tag{3.2}$$

representação alternativa esta que explicita a dependência entre a função hamiltoniana H e as derivadas primeiras da função ϕ tão somente.

Numericamente, bem como utilizando o método de Euler progressivo para a discretização do tempo t, a equação Hamilton-Jacobi pode ser reescrita como

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \hat{H}^n(\phi_x^-, \phi_x^+; \phi_y^-, \phi_y^+) = 0 \tag{3.3}$$

em que $\hat{H}^n(\phi_x^-, \phi_x^+; \phi_y^-, \phi_y^+)$ representa a aproximação numérica da hamiltoniana $H(\phi_x, \phi_y)$ após a n-ésima iteração e ϕ_x^- e ϕ_x^+ (ϕ_y^- e ϕ_y^+) são as aproximações por diferenças finitas regressivas e progressivas, respectivamente, da derivada de ϕ na direção x (direção y) – os métodos Hamilton-Jacobi ENO [51] e Hamilton-Jacobi WENO [31] apresentam-se como alternativas mais precisas para o cálculo discreto das derivadas parciais, porém, a um custo computacional consideravelmente alto, fato este que fundamenta a opção pelo método das diferenças finitas.

Para a determinação do passo temporal Δt da Equação 3.3, emprega-se a condição de Courant-Friedrichs-Lewy [15]. A condição CFL, como é comumente conhecida, garante que a propagação dos dados numéricos ocorra, exemplificando no caso unidimensional com espaçamento Δx, sob uma velocidade $\Delta x/\Delta t$ igual ou maior ao valor absoluto da velocidade F de propagação dos dados físicos [49], o que, por sua vez, assegura a estabilidade e a convergência da resolução numérica da Equação 3.3 sob diferenças finitas.

No caso unidimensional, enuncia-se a condição CFL segundo a relação

$$\Delta t < \frac{\Delta x}{\max\{\,|F|\,\}} \;\Rightarrow\; \Delta t \cdot \max\left\{\frac{|F|}{\Delta x}\right\} < 1 \tag{3.4}$$

na qual $\max\{\,|F|\,\}$ é o máximo do valor absoluto da velocidade F definida sobre todo o eixo do domínio x, sendo fácil e diretamente estendida para

$$\Delta t \cdot \max\left\{\frac{|F_x|}{\Delta x} + \frac{|F_y|}{\Delta y}\right\} < 1 \tag{3.5}$$

no caso bidimensional de interesse, em que $|F_x|$ e $|F_y|$ são os valores absolutos da velocidade F nas direções x e y, respectivamente. Nota-se que a condição CFL determina não um valor específico para o passo temporal Δt, mas sim um intervalo válido de valores que garantem a estabilidade e a convergência da resolução numérica requeridas.

À Equação Hamilton-Jacobi 3.3 em específico, a condição CFL é dada por

$$\Delta t \cdot \max\left\{\frac{|H_1|}{\Delta x} + \frac{|H_2|}{\Delta y}\right\} < 1 \tag{3.6}$$

em que H_1 e H_2 são as derivadas parciais da hamiltoniana H com relação a ϕ_x e ϕ_y, respectivamente [49]. Uma vez que a Equação *Level Set* 2.7 encaixa-se no padrão das equações Hamilton-Jacobi, com $H(\vec{\nabla}\phi) = F \cdot |\vec{\nabla}\phi|$, estabelece-se, por fim, a relação

$$\Delta t \cdot \max\left\{\frac{|F| \cdot |\phi_x|}{|\vec{\nabla}\phi| \cdot \Delta x} + \frac{|F| \cdot |\phi_y|}{|\vec{\nabla}\phi| \cdot \Delta y}\right\} < 1 \tag{3.7}$$

como a condição CFL desejada.

A determinação computacional, conforme a Relação 3.7, do passo temporal Δt da Equação *Level Set* 2.7 discretizada é obtida percorrendo-se a matriz $\phi^0 = \phi^0_{origem}$ na qual armazenou-se, no passo anterior, a função distância sinalizada relativa à fatia origem – como a metamorfose inicia-se a partir da fatia origem em direção à fatia destino, o papel da função ϕ na Equação *Level Set* 2.7 é desempenhado pela função distância sinalizada ϕ_{origem}, sendo os valores desta, portanto, iterativamente atualizados durante a evolução, justificando a matriz de varredura.

Por toda a varredura, para cada elemento matricial de ϕ^0_{origem}, calculam-se as parcelas do parâmetro da função $\max$, em que $F = \phi^0 - \phi^0_{destino} = \phi^0_{origem} - \phi^0_{destino}$ é averiguado para

o pixel avaliado (Equação 2.13) e Δx e Δy são os espaçamentos respectivos da grade das fatias nas direções x e y. Havendo dois possíveis valores para cada uma das variáveis ϕ_x e ϕ_y (ϕ_x^-, ϕ_x^+ e ϕ_y^-, ϕ_y^+, respectivamente), o parâmetro da função $\max$ pode assumir, logo, quatro diferentes valores para cada elemento de ϕ^0_{origem}; o maior dentre os quatro é selecionado como o máximo local, e o maior dentre todos os máximos locais, como o máximo global ($\max_{global}$) da matriz.

O máximo global encontrado define o intervalo válido

$$\Delta t < \frac{1}{\max_{global}} \tag{3.8}$$

de valores para Δt, sendo que o valor específico $3/(4 \cdot \max_{global})$ mostrou-se, empiricamente, uma boa escolha para o passo temporal.

3.3.5 Passo 3.5: metamorfose via *Level Set*

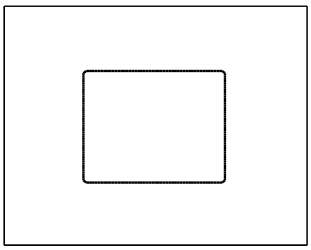
Os passos anteriores pré-processam os dados do problema, para deles extrair as informações necessárias para o procedimento de metamorfose entre as fatias origem e destino, a partir do qual ocorre a geração, propriamente dita, das fatias intermediárias requeridas para a completude do processo de reconstrução como um todo.

Durante a metamorfose, o contorno presente na fatia origem, sob a ação de uma velocidade normal, deforma-se iterativamente, a fim de assumir, ao final da evolução, a forma do contorno presente na fatia destino. As Figuras de 3.8 a 3.14 ilustram sete exemplos de metamorfose entre fatias pelo método *Level Set*, completados em 82, 327, 203, 297, 228, 91 e 120 iterações, respectivamente: em todos os sete casos, a subfigura (a) apresenta a fatia origem, a subfigura (b), a fatia destino, e cada uma das subsequentes (c)~(h), a fatia intermediária obtida na n-ésima iteração especificada.

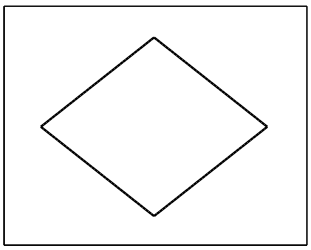
O procedimento de metamorfose modelado pelo *Level Set* é descrito, matematicamente, pela Equação 2.13, uma equação Hamilton-Jacobi, com $H(\vec{\nabla}\phi) = (\phi^n - \phi^n_{destino}) \cdot |\vec{\nabla}\phi^n|$. Para a resolução numérica de uma equação do tipo Hamilton-Jacobi, é necessário determinar os métodos a serem utilizados para a discretização do tempo t e da função hamiltoniana H. Há, para ambas, várias opções na literatura que oferecem bons resultados. Relativamente à discretização temporal, talvez o mais conhecido seja o método de Euler progressivo

$$\phi_t^{n+1} = \frac{\phi^{n+1} - \phi^n}{\Delta t} \tag{3.9}$$

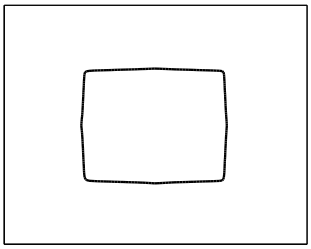
apresentado na seção anterior como o esquema selecionado, por sua simplicidade de implementação. Entretanto, outros métodos poderiam ser igualmente utilizados, como o TVD de Runge-Kutta [24].



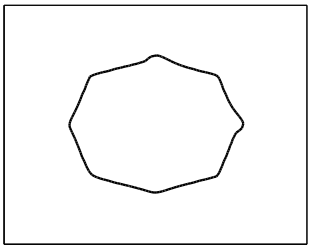
(a) Fatia origem;



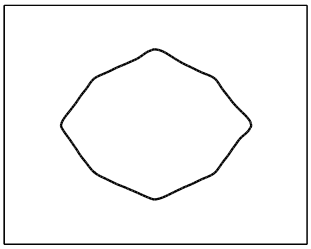
(b) Fatia destino;



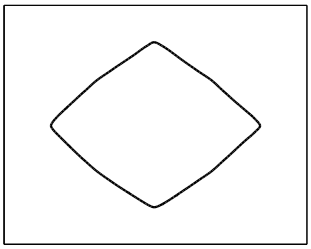
(c) n = 1;



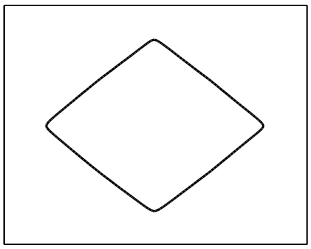
(d) n = 10;



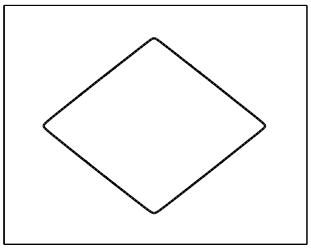
(e) n = 20;



(f) n = 40;

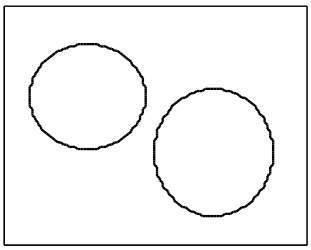


(g) n = 60;

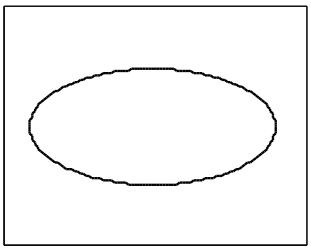


(h) n = 82.

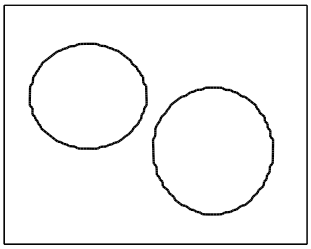
Figura 3.8: Metamorfose via *Level Set* entre as fatias origem e destino (I).



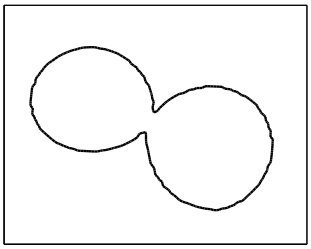
(a) Fatia origem;



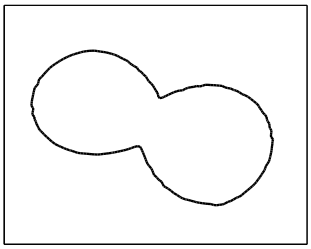
(b) Fatia destino;



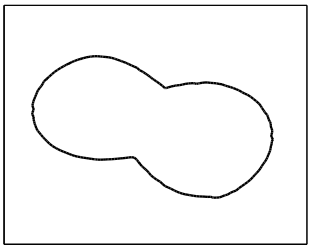
(c) n = 1;



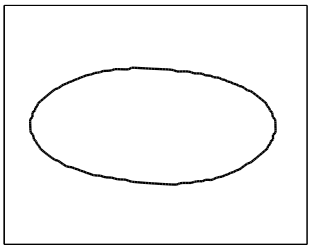
(d) n = 7;



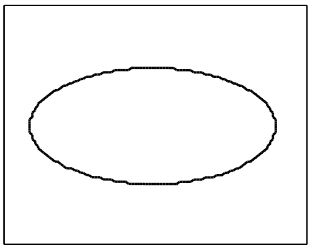
(e) n = 14;



(f) n = 28;

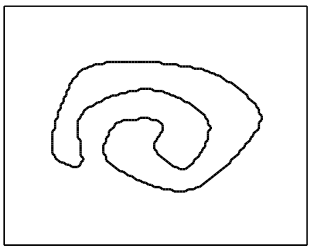


(g) n = 150;

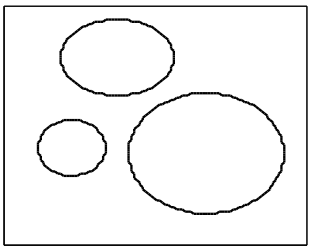


(h) n = 327.

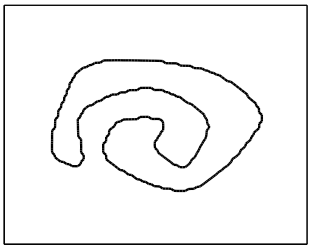
Figura 3.9: Metamorfose via *Level Set* entre as fatias origem e destino (II).



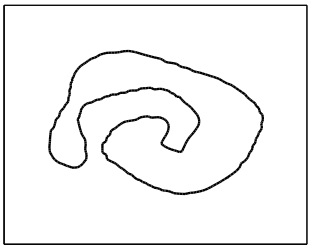
(a) Fatia origem;



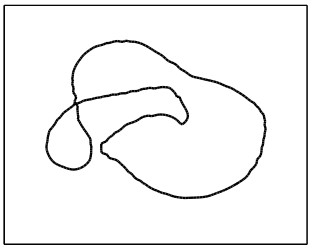
(b) Fatia destino;



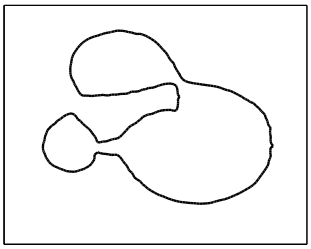
(c) n = 1;



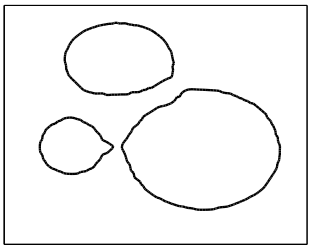
(d) n = 10;



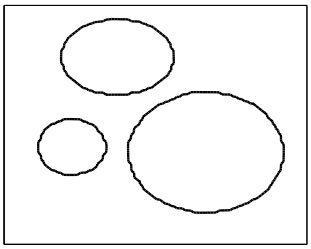
(e) n = 25;



(f) n = 50;

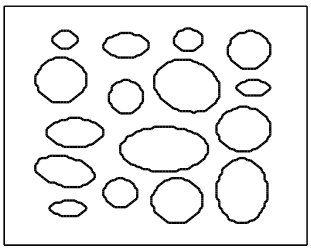


(g) n = 100;

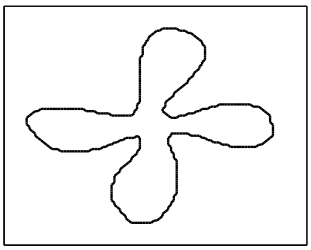


(h) n = 203.

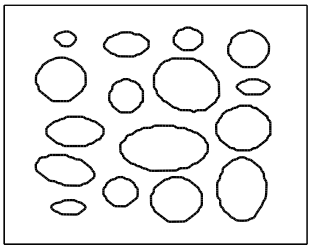
Figura 3.10: Metamorfose via *Level Set* entre as fatias origem e destino (III).



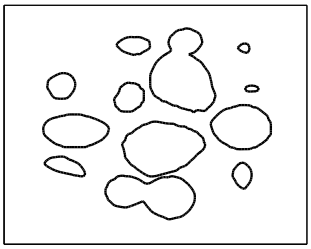
(a) Fatia origem;



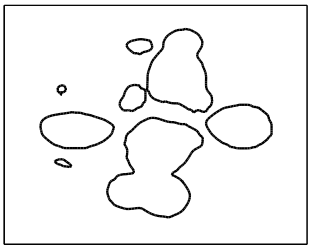
(b) Fatia destino;



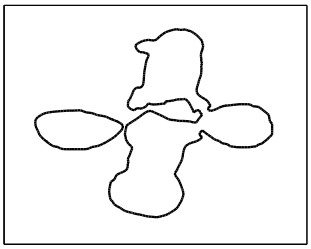
(c) n = 1;



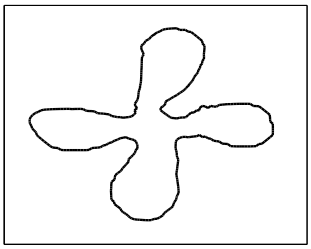
(d) n = 19;



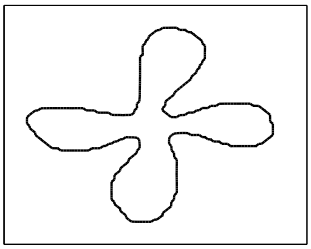
(e) n = 38;



(f) n = 75;

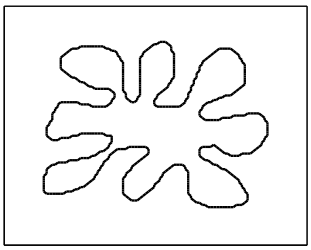


(g) n = 149;

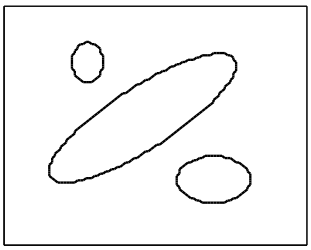


(h) n = 297.

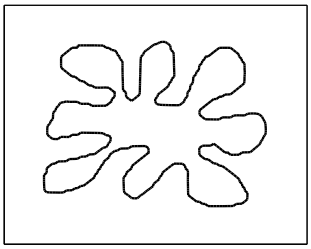
Figura 3.11: Metamorfose via *Level Set* entre as fatias origem e destino (IV).



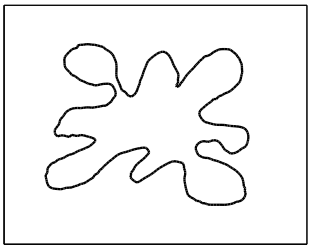
(a) Fatia origem;



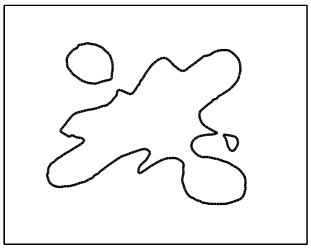
(b) Fatia destino;



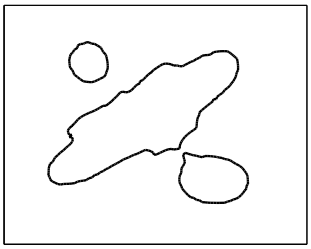
(c) n = 1;



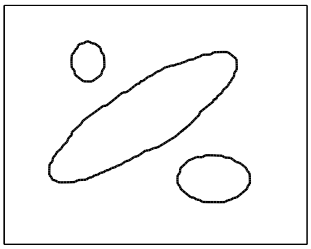
(d) n = 15;



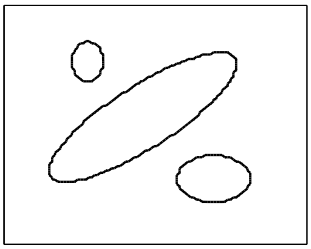
(e) n = 29;



(f) n = 57;

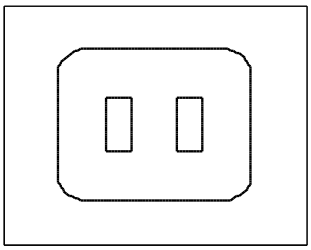


(g) n = 114;

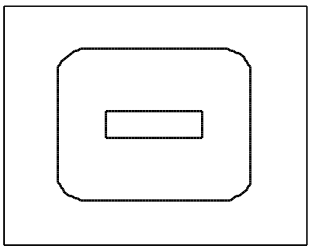


(h) n = 228.

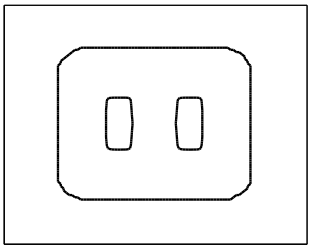
Figura 3.12: Metamorfose via *Level Set* entre as fatias origem e destino (V).



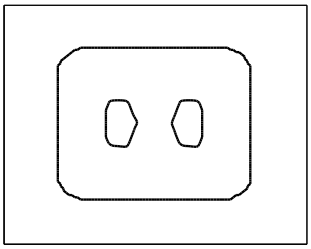
(a) Fatia origem;



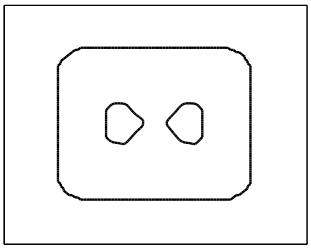
(b) Fatia destino;



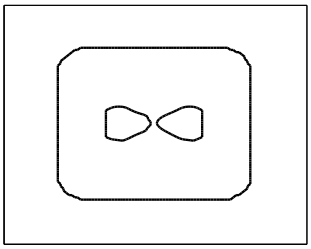
(c) n = 1;



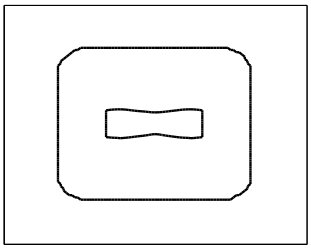
(d) n = 5;



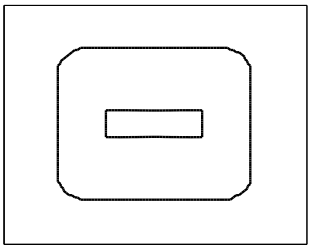
(e) n = 11;



(f) n = 22;

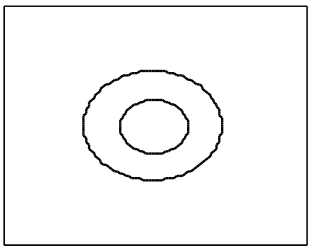


(g) n = 45;

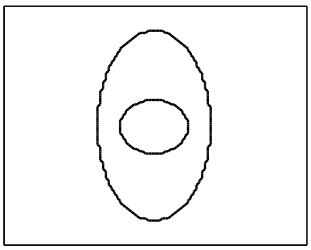


(h) n = 91.

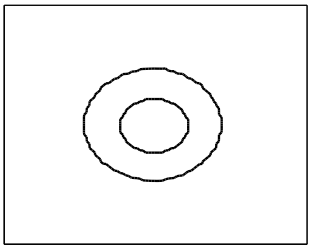
Figura 3.13: Metamorfose via *Level Set* entre as fatias origem e destino (VI).



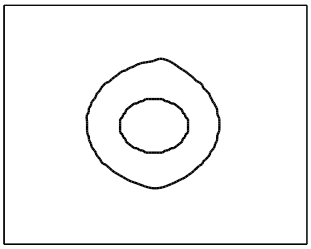
(a) Fatia origem;



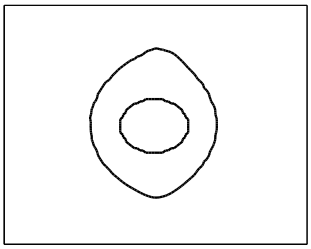
(b) Fatia destino;



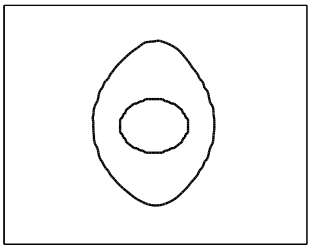
(c) n = 1;



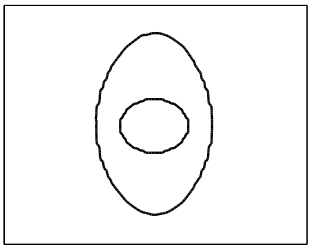
(d) n = 7;



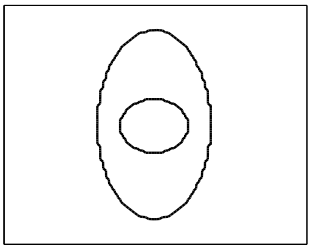
(e) n = 17;



(f) n = 30;



(g) n = 60;



(h) n = 120.

Figura 3.14: Metamorfose via *Level Set* entre as fatias origem e destino (VII).

Com relação à discretização da hamiltoniana H, são de notório conhecimento os esquemas de Lax-Friedrichs, de Roe-Fix e de Godunov [49]. Devido à eficiência computacional apresentada, o segundo método foi escolhido, o qual, por sua vez, baseia-se no primeiro.

De acordo com o esquema de Roe-Fix, a hamiltoniana numérica $\hat{H}^n$ da Equação 3.3, para cada ponto (y, x) (coordenadas invertidas, conforme explanação no Apêndice A) da grade da fatia na n-ésima iteração, é calculada segundo a expressão

$$\hat{H}^n(\phi_x^-, \phi_x^+; \phi_y^-, \phi_y^+) = H(\phi_x^*, \phi_y^*) - \alpha^x \left(\frac{\phi_x^+ - \phi_x^-}{2} \right) - \alpha^y \left(\frac{\phi_y^+ - \phi_y^-}{2} \right) \tag{3.10}$$

em que $H(\phi_x^*, \phi_y^*)$ é o valor da hamiltoniana $H(\phi_x, \phi_y)$ (Equação 3.2), para $\phi_x = \phi_x^* \in \{\phi_x^-, \phi_x^+\}$ e $\phi_y = \phi_y^* \in \{\phi_y^-, \phi_y^+\}$, e

$$\alpha^x = \max |H_1(\phi_x, \phi_y)| \tag{3.11}$$

$$\alpha^y = \max |H_2(\phi_x, \phi_y)| \tag{3.12}$$

são os coeficientes controladores da dissipação numérica, para $H_1(\phi_x, \phi_y)$ e $H_2(\phi_x, \phi_y)$ as derivadas parciais de $H(\phi_x, \phi_y)$ com relação a ϕ_x e ϕ_y, respectivamente, e o máximo averiguado sobre $\phi_x \in \{\phi_x^-, \phi_x^+\}$ e $\phi_y \in \{\phi_y^-, \phi_y^+\}$. Todas as derivadas parciais presentes nas Equações 3.10, 3.11 e 3.12 (assim como nas verificações a seguir descritas) são avaliadas no ponto da grade (y, x) em que se calcula a hamiltoniana numérica.

Para determinar os valores dos coeficientes α e, ainda, o método das diferenças finitas a ser empregado para o cálculo das derivadas parciais ϕ_x^* e ϕ_y^*, definem-se, inicialmente, os conjuntos $I^x = \{\phi_x^-, \phi_x^+\}$ e $I^y = \{\phi_y^-, \phi_y^+\}$. Em seguida, calculam-se as derivadas parciais H_1 e H_2 para ambos $\phi_x \in I^x$ e ambos $\phi_y \in I^y$ (ou seja, há, ao todo, quatro possibilidades de pares ordenados (ϕ_x, ϕ_y) para H_1 e H_2). Se tanto H_1 quanto H_2 mantiverem o mesmo sinal nas quatro respectivas avaliações possíveis (sem, necessariamente, o sinal de H_1 ser igual ao de H_2), faz-se $\alpha^x = \alpha^y = 0$; $\phi_x^* = \phi_x^-$, caso $H_1 > 0$; $\phi_x^* = \phi_x^+$, caso $H_1 < 0$; $\phi_y^* = \phi_y^-$, para $H_2 > 0$; e $\phi_y^* = \phi_y^+$, para $H_2 < 0$.

Na eventualidade de H_1 e/ou H_2 apresentar(em) avaliações com sinais distintos, os conjuntos I^x e I^y são, então, redefinidos: I^x passa a armazenar as derivadas parciais ϕ_x calculadas, tanto progressiva quanto regressivamente, para cada um dos pontos da grade no intervalo de $(y, x-3)$ a $(y, x+3)$, e I^y, as derivadas parciais regressivas e progressivas de ϕ_y no intervalo de $(y-3, x)$ a $(y+3, x)$. Para fins de distinção, a essas redefinições de I^x e I^y são atribuídas as notações I^x_{novo} e I^y_{novo} e às suas versões anteriores originais, as notações $I^x_{original}$ e $I^y_{original}$, isto é, $I^x_{original} = I^x$ e $I^y_{original} = I^y$.

Com base nesses novos conjuntos I^x_{novo} e I^y_{novo}, uma segunda bateria de testes é realizada, porém, mantendo raciocínio similar. Se H_1 conservar o sinal para todos os possíveis pares (ϕ_x, ϕ_y), com $\phi_x \in I^x_{original}$ e $\phi_y \in I^y_{novo}$, estabelecem-se $\alpha^x = 0$; $\phi_x^* = \phi_x^-$, caso $H_1 > 0$; $\phi_x^* = \phi_x^+$, caso $H_1 < 0$; e $\phi_y^* = (\phi_y^- + \phi_y^+)/2$. Particularmente a α^y, o coeficiente é determinado

conforme a definição dada pela Equação 3.12, com o argumento da função $\max$ calculado para todo $\phi_x \in I^x_{novo}$ e todo $\phi_y \in I^y_{original}$.

Não satisfazendo H_1 a conservação de sinal, testa-se H_2 de maneira análoga: caso H_2 mantenha o sinal para todos os pares (ϕ_x, ϕ_y) possíveis, com $\phi_x \in I^x_{novo}$ e $\phi_y \in I^y_{original}$, faz-se $\alpha^y = 0$; $\phi^*_y = \phi^-_y$, se $H_2 > 0$; $\phi^*_y = \phi^+_y$, se $H_2 < 0$; e $\phi^*_x = (\phi^-_x + \phi^+_x)/2$. Similarmente, calcula-se α^x pela Equação 3.11, sendo a função $\max$ avaliada para todo $\phi_x \in I^x_{original}$ e todo $\phi_y \in I^y_{novo}$.

Em última instância, se H_1 e H_2 ambos não conservarem o sinal em suas relativas avaliações para todas as respectivas possibilidades de pares (ϕ_x, ϕ_y), definem-se, por fim, $\phi^*_x = (\phi^-_x + \phi^+_x)/2$; $\phi^*_y = (\phi^-_y + \phi^+_y)/2$; α^x segundo a Equação 3.11, com $\phi_x \in I^x_{original}$ e $\phi_y \in I^y_{novo}$; e α^y segundo a Equação 3.12, com $\phi_x \in I^x_{novo}$ e $\phi_y \in I^y_{original}$.

A tradução computacional do esquema de Roe-Fix apresentado encontra-se no Algoritmo 3.1 e aplica-se a quaisquer modelos de equações Hamilton-Jacobi. Como parâmetros de entrada, o pseudocódigo solicita a matriz representativa da função ϕ (Equação 3.3) e as coordenadas (y, x) do ponto no qual será calculada a hamiltoniana numérica $\hat{H}^n(y, x)$, valor este retornado pelo algoritmo como saída única.

Para a evolução da Equação 2.13 de interesse, a entrada do Algoritmo 3.1 adota características próprias. A matriz ϕ, como explanado, armazena a função distância sinalizada da fatia sendo metamorfoseada na iteração corrente. Equivalentemente, o Algoritmo 3.1, para a execução da n-ésima iteração, recebe, como entrada, a matriz ϕ tal que $\phi = \phi^{n-1}_{origem}$, para ϕ^{n-1}_{origem} a matriz função distância sinalizada obtida da forma assumida pela fatia origem após a $(n-1)$-ésima iteração da metamorfose; executado o Algoritmo 3.1 para cada ponto da matriz ϕ^{n-1}_{origem}, tem-se, ao final da n-ésima iteração, a matriz ϕ^n_{origem}, portanto.

Ainda, como $H(\vec{\nabla}\phi) = F \cdot |\vec{\nabla}\phi^{n-1}| = (\phi^{n-1}_{origem} - \phi^{n-1}_{destino}) \cdot |\vec{\nabla}\phi^{n-1}_{origem}|$ para a $(n-1)$-ésima iteração da Equação 2.13, demanda-se o valor da velocidade normal $F = \phi^{n-1}_{origem} - \phi^{n-1}_{destino}$ para o ponto (y, x) em que se determina a hamiltoniana numérica como um terceiro parâmetro de entrada, uma vez que emprega-se essa quantidade no cálculo de H_1, H_2 e $H(\phi^*_x, \phi^*_y)$ no interior do algoritmo.

À parte das entradas particulares, o corpo do Algoritmo 3.1 permanece inalterado para a iteração da Equação 2.13.

3.3.6 Passo 3.6: armazenamento em memória da fatia

Ao término de cada iteração n da metamorfose, armazena-se, em mémória, a n-ésima fatia intermediária gerada, ou seja, a matriz bidimensional ϕ^n_{origem} contendo os valores da função distância sinalizada àquela relativa é acondicionada em uma matriz tridimensional, para utilização durante o estágio de pós-processamento das fatias.

Esse armazenamento revela-se necessário, ao invés do pós-processamento imediato

Algoritmo 3.1: Cálculo da hamiltoniana numérica $\hat{H}^n(y, x)$ via esquema de Roe-Fix

Entrada: Matriz ϕ e coordenadas (y, x)
Saída: Valor da hamiltoniana numérica $\hat{H}^n(y, x)$

1: Calcular $\phi_x^- = \phi_x^-(y, x)$ e $\phi_x^+ = \phi_x^+(y, x)$
2: $I^x_{original} \leftarrow \{\phi_x^-, \phi_x^+\}$
3: Calcular $\phi_y^- = \phi_y^-(y, x)$ e $\phi_y^+ = \phi_y^+(y, x)$
4: $I^y_{original} \leftarrow \{\phi_y^-, \phi_y^+\}$
5: $M_1 \leftarrow \varnothing$, $M_2 \leftarrow \varnothing$
6: **for all** $\phi_x \in I^x_{original}$ **do**
7: **for all** $\phi_y \in I^y_{original}$ **do**
8: Calcular $H_1(\phi_x, \phi_y) = \dfrac{\partial H}{\partial(\phi_x)}$
9: $M_1 \leftarrow M_1 \cup \{H_1(\phi_x, \phi_y)\}$
10: Calcular $H_2(\phi_x, \phi_y) = \dfrac{\partial H}{\partial(\phi_y)}$
11: $M_2 \leftarrow M_2 \cup \{H_2(\phi_x, \phi_y)\}$
12: **end for**
13: **end for**
14: **if** os sinais dos elementos de M_1 são todos iguais **and** os sinais dos elementos de M_2 são todos iguais **then**
15: $\alpha^x \leftarrow 0$, $\alpha^y \leftarrow 0$
16: **if** todos os elementos de M_1 são positivos **then**
17: $\phi_x^* \leftarrow \phi_x^-$
18: **else**
19: $\phi_x^* \leftarrow \phi_x^+$
20: **end if**
21: **if** todos os elementos de M_2 são positivos **then**
22: $\phi_y^* \leftarrow \phi_y^-$
23: **else**
24: $\phi_y^* \leftarrow \phi_y^+$
25: **end if**
26: **else**
27: $I^x_{novo} \leftarrow \varnothing$
28: **for** $i = x - 3$ até $i = x + 3$ **do**
29: Calcular $\phi_x^-(y, i)$

Algoritmo 3.1: Cálculo da hamiltoniana numérica $\hat{H}^n(y, x)$ via esquema de Roe-Fix

Calcular $\phi_x^+(y, i)$
$I^x_{novo} \leftarrow I^x_{novo} \cup \{\phi_x^-(y, i), \phi_x^+(y, i)\}$
end for
$I^y_{novo} \leftarrow \varnothing$
for $j = y - 3$ até $j = y + 3$ **do**
Calcular $\phi_y^-(j, x)$
Calcular $\phi_y^+(j, x)$
$I^y_{novo} \leftarrow I^y_{novo} \cup \{\phi_y^-(j, x), \phi_y^+(j, x)\}$
end for
$M_1 \leftarrow \varnothing$
for all $\phi_x \in I^x_{original}$ **do**
for all $\phi_y \in I^y_{novo}$ **do**
Calcular $H_1(\phi_x, \phi_y) = \dfrac{\partial H}{\partial(\phi_x)}$
$M_1 \leftarrow M_1 \cup \{H_1(\phi_x, \phi_y)\}$
end for
end for
$M_2 \leftarrow \varnothing$
for all $\phi_x \in I^x_{novo}$ **do**
for all $\phi_y \in I^y_{original}$ **do**
Calcular $H_2(\phi_x, \phi_y) = \dfrac{\partial H}{\partial(\phi_y)}$
$M_2 \leftarrow M_2 \cup \{H_2(\phi_x, \phi_y)\}$
end for
end for
if os sinais dos elementos de M_1 são todos iguais **and** os sinais dos elementos de M_2 não são todos iguais **then**
if todos os elementos de M_1 são positivos **then**
$\phi_x^* \leftarrow \phi_x^-$
else
$\phi_x^* \leftarrow \phi_x^+$
end if
$\phi_y^* \leftarrow \dfrac{\phi_y^- + \phi_y^+}{2}$

Algoritmo 3.1: Cálculo da hamiltoniana numérica $\hat{H}^n(y, x)$ via esquema de Roe-Fix

$\alpha^x \leftarrow 0$

$\alpha^y \leftarrow \max |H_2(\phi_x, \phi_y)|$, $\forall \phi_x \in I^x_{novo}$, $\forall \phi_y \in I^y_{original}$

else if os sinais dos elementos de M_1 não são todos iguais **and** os sinais dos elementos de M_2 são todos iguais **then**

$\phi^*_x \leftarrow \dfrac{\phi^-_x + \phi^+_x}{2}$

if todos os elementos de M_2 são positivos **then**

$\phi^*_y \leftarrow \phi^-_y$

else

$\phi^*_y \leftarrow \phi^+_y$

end if

$\alpha^x \leftarrow \max |H_1(\phi_x, \phi_y)|$, $\forall \phi_x \in I^x_{original}$, $\forall \phi_y \in I^y_{novo}$

$\alpha^y \leftarrow 0$

else

$\phi^*_x \leftarrow \dfrac{\phi^-_x + \phi^+_x}{2}$

$\phi^*_y \leftarrow \dfrac{\phi^-_y + \phi^+_y}{2}$

$\alpha^x \leftarrow \max |H_1(\phi_x, \phi_y)|$, $\forall \phi_x \in I^x_{original}$, $\forall \phi_y \in I^y_{novo}$

$\alpha^y \leftarrow \max |H_2(\phi_x, \phi_y)|$, $\forall \phi_x \in I^x_{novo}$, $\forall \phi_y \in I^y_{original}$

end if

end if

return $H(\phi^*_x, \phi^*_y) - \alpha^x \left(\dfrac{\phi^+_x - \phi^-_x}{2} \right) - \alpha^y \left(\dfrac{\phi^+_y - \phi^-_y}{2} \right)$

à geração, pois o total de fatias intermediárias geradas, dado este imprescindível para que, entre outros procedimentos ulteriores, as fatias intermediárias sejam empilhadas nas alturas corretas, é uma informação somente obtida *a posteriori* à completude da etapa de metaformose.

3.3.7 Passo 3.7: teste de convergência

Sejam $\phi^0_{destino}$, a matriz função distância sinalizada relativa à fatia destino, ϕ^n_{origem}, a matriz função distância sinalizada obtida da forma assumida pela fatia origem ao final da n-ésima iteração da metamorfose, e ϕ^{n-1}_{origem}, a matriz obtida após a $(n-1)$-ésima iteração. O critério de parada (CP) empregado para a resolução iterativa do procedimento de meta-

morfose é determinado pela expressão

$$\text{CP} = |\,\texttt{norm}(\phi^{n}_{origem} - \phi^{0}_{destino}) - \texttt{norm}(\phi^{n-1}_{origem} - \phi^{0}_{destino})\,| \tag{3.13}$$

sendo `norm` a função pertencente à biblioteca do MATLAB que calcula a norma-2 (ou norma euclidiana) entre matrizes.

O valor específico do parâmetro de convergência é dependente das características das fatias em processamento, principalmente da geometria dos contornos nelas presentes. Resultados práticos, no entanto, confirmaram 0.25 como um parâmetro aplicável à maioria dos casos e que retorna as saídas (isto é, as fatias intermediárias) com a precisão numérica esperada – em outras palavras, a fatia origem prossegue metamorfoseando-se em direção à fatia destino enquanto CP > 0.25.

Em um primeiro momento, a Equação 3.13 mostra-se desnecessariamente complicada, se comparada a alternativas mais simples e diretas para testes de convergência numéricos, como, por exemplo

$$\text{CP}_{\text{alternativo}} = |\,\texttt{norm}(\phi^{n}_{origem} - \phi^{n-1}_{origem})\,| \tag{3.14}$$

Entretanto, verificações realizadas comprovaram que, para um mesmo caso-teste, a metamorfose atinge a convergência mais rapidamente utilizando-se a Equação 3.13 como critério de parada, ante a Equação 3.14.

3.3.8 Passo 3.8: arquivamento das alturas intermediárias

Atingida a convergência estipulada pelo critério de parada, cessa-se a metamorfose entre as fatias origem e destino e aufere-se, logo, o total k de fatias intermediárias geradas. A partir desse dado, as respectivas alturas de cada fatia são determinadas, dividindo-se o intervalo compreendido entre as alturas das fatias origem e destino em $k+1$ subintervalos, de modo que o extremo direito do n-ésimo subintervalo retorne a altura da n-ésima fatia intermediária gerada.

Computacionalmente, a operação descrita é obtida pela execução da função `linspace` pertencente à biblioteca do MATLAB. Sendo `altura_origem` a variável na qual está armazenada a altura da fatia origem e `altura_destino`, a da fatia destino, a sintaxe

```
h = linspace(altura_origem,altura_destino,k+2)
```

retorna um vetor `h` com `k+2` elementos linearmente espaçados, em que o primeiro e o último elementos contêm as alturas respectivas das fatias origem e destino e cada um dos n elementos interiores, a altura da $(n-1)$-ésima fatia intermediária gerada, para $2 \leq n \leq k+1$.

Os elementos desse vetor, à exceção do último, são copiados, sequencialmente, para um arquivo texto, único para todo o processo de reconstrução. Estabelecido de outra forma,

as alturas da fatia origem e das intermediárias geradas, à conclusão da metamorfose implementada pela i-ésima iteração da leitura das fatias do conjunto original de entrada, são transcritas, no interior do arquivo texto, na sequência das alturas copiadas pela $(i-1)$-ésima iteração.

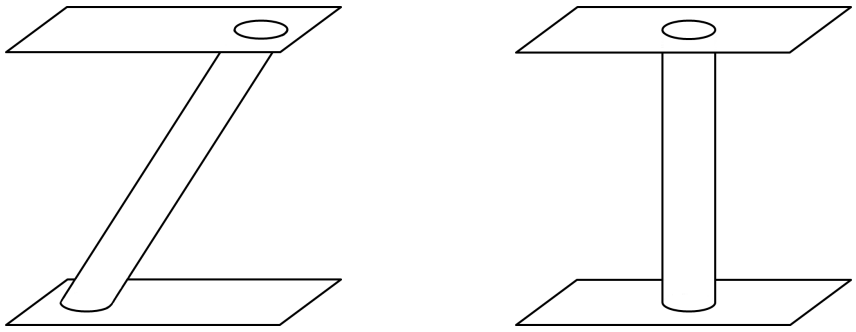
Somente transcreve-se a altura da fatia destino (equivalentemente, o elemento $k+2$ do vetor) para o arquivo texto quando a leitura das fatias originais encontra-se em sua última iteração, para evitar redundância de dados no arquivo, uma vez que, ao longo de todo o processo de reconstrução, a fatia destino da iteração i da leitura converte-se na fatia origem da iteração $i+1$.

O arquivo texto com as alturas de cada fatia, entre originais e intermediárias, constitui um primeiro parâmetro de entrada para a etapa de reconstrução (Seção 3.4), o que justifica o procedimento explanado.

3.3.9 Passo 3.9: descentralização dos contornos

Na Subseção 3.3.2, explanou-se a necessidade teórica e computacional da centralização dos contornos para a viabilidade da metamorfose via *Level Set*. Entretanto, a manutenção destes, após a metamorfose, em posições diferentes das originais ou estimadas afeta a corretude da superfície a ser reconstruída, dado que o processo de reconstrução embasa-se na localização relativa dos contornos em suas respectivas fatias.

A Figura 3.15 ilustra a diferença entre a superfície corretamente obtida a partir de fatias cujos contornos foram transladados às posições originais ou estipuladas e aquela em que não submeteram-se os contornos à transformação inversa de descentralização, justificando visualmente a imprescindibilidade do passo.



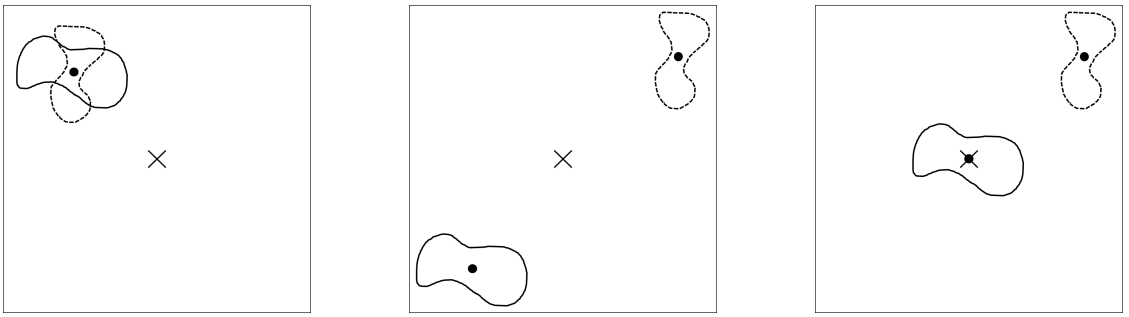
(a) Contornos descentralizados; (b) Contornos centralizados.

Figura 3.15: Superfície reconstruída entre duas fatias.

Porquanto os dados de entrada do problema prontamente armazenam as fatias origem e destino com seus respectivos contornos descentralizados, aplica-se a descentralização

exclusivamente nas fatias intermediárias geradas pela metamorfose via *Level Set*, e somente se ambos os contornos presentes nas fatias origem e destino não forem originalmente centralizados, isto é, se os centroides de ambos, em suas configurações iniciais de entrada, não se sobrepuserem ao centro da matriz da fatia. Nessa eventualidade, o passo de centralização não haverá sido executado e, consequentemente, o de descentralização torna-se desnecessário.

Há, logo, três circunstâncias nas quais a descentralização dos contornos é implementada (Figura 3.16): quando os centroides dos contornos (círculo cheio) pertencentes às fatias origem (curva cheia) e destino (curva tracejada) localizarem-se em um ponto distinto do centro geométrico da matriz da fatia (símbolo ×), superpostos (Figura 3.16(a)) ou não (Figura 3.16(b)), ou caso apenas um dos contornos esteja centralizado na fatia (Figura 3.16(c)), sendo a posição dos contornos sempre relativa às fatias em suas representações originais de entrada.



(a) Centroides superpostos; (b) Centroides não superpostos; (b) Contorno único centralizado.

Figura 3.16: Casos de implementação do passo de descentralização dos contornos.

Para situações como a da Figura 3.16(a), os centroides dos contornos pertencentes às fatias intermediárias geradas igualmente se sobrepõem aos dos presentes nas fatias origem e destino. Como consequência, a distância por estes transladada no passo de centralização define o deslocamento a ser percorrido por aqueles, no sentido inverso.

Em ocasiões semelhantes às das Figuras 3.16(b) e (c), para determinar a distância a ser percorrida durante a descentralização, é preciso estimar, primeiramente, os centroides dos contornos em cada fatia intermediária.

Seja, assim, o exemplo ilustrativo da Figura 3.17, sem perda de generalidade. O contorno origem deve evoluir não somente a fim de assumir a forma do contorno destino, mas inclusive deslocar-se, durante a metamorfose via *Level Set*, de sua posição inicial no canto inferior esquerdo da fatia à posição na qual localiza-se o contorno destino, no canto superior direito. A seta que une ambos os centroides demarca, portanto, o percurso a ser percorrido pelo centroide origem ao longo de sua evolução; em outras palavras, os cen-

troides dos contornos presentes nas fatias intermediárias geradas posicionam-se sobre o segmento de reta representado pela seta, respeitando a relação direta entre o sentido desta e a ordem de geração das fatias.

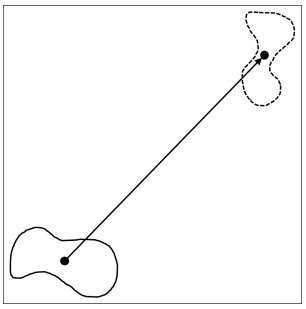


Figura 3.17: Deslocamento do centroide.

A determinação das coordenadas dos centroides intermedários dá-se por meio da execução de duas chamadas da função `linspace` pertencente à biblioteca do MATLAB, segundo sintaxe semelhante à apresentada na Subseção 3.3.8, as quais criam dois vetores distintos, um com as coordenadas x dos centroides intermediários linearmente espaçadas entre as coordenadas x dos centroides origem e destino, outro, de forma análoga, com as coordenadas y.

O deslocamento, nas direções x e y da fatia, a ser empregado pelo passo de descentralização ao n-ésimo contorno intermediário é, então, calculado, subtraindo-se as coordenadas determinadas para o centroide deste – a saber, os elementos de ordem $n + 1$ dos vetores gerados pela função `linspace` – das coordenadas do centro da matriz da fatia.

3.3.10 Passo 3.10: redimensionamento das fatias

A translação dos contornos à região delimitada pelas dimensões iniciais da fatia, pelo passo anterior, ocasiona o esvaziamento da área compreendida pelas linhas/colunas adicionadas às matrizes das fatias durante o redimensionamento inicial (Subseção 3.3.1). Não possuindo mais informações, essas linhas/colunas podem ser excluídas, de modo que as fatias são, logo, redimensionadas ao tamanho original de entrada.

3.3.11 Passo 3.11: empilhamento das fatias

Segundo o convencionado, a n-ésima fatia intermediária gerada é dada, implicitamente, pela curva de nível 0 da superfície ϕ^n_{origem}. Desse modo, para que, em ambiente MATLAB, a fatia seja empilhada na altura correta, determinada esta pelo elemento $n + 1$ do vetor `h` construído na Subseção 3.3.8, desloca-se ϕ^n_{origem} verticalmente, no sentido positivo

do eixo y, em `h(n+1)` unidades, somando-se, a cada elemento da matriz ϕ^n_{origem}, a quantidade `h(n+1)`. A fatia é, então, corretamente empilhada no espaço coordenado cartesiano, executando-se a linha de comando

```
contour3(phi,[h(n+1) h(n+1)])
```

sendo `contour3` a função pertencente à biblioteca do MATLAB que plota tridimensionalmente as curvas de nível de uma superfície e `phi`, a matriz computacional que armazena ϕ^n_{origem}. Esse procedimento evita que todas as fatias geradas sejam acumuladas sobre o plano $z = 0$.

A cada iteração *i* da leitura das fatias, empilham-se as fatias origem, cuja altura obtém-se diretamente dos dados de entrada, e as intermediárias, conforme o esquematizado. O empilhamento da fatia destino somente é realizado na última iteração da leitura, para evitar redundância (Subseção 3.3.8).

3.3.12 Passo 3.12: armazenamento em arquivo das fatias

Concluído o pós-processamento, as fatias intermediárias geradas são armazenadas em arquivo *bitmap* monocromático, a mesma extensão das fatias originais, compondo o conjunto total de fatias originais e intermediárias um segundo parâmetro de entrada para a etapa de reconstrução (Seção 3.4).

3.3.13 *Particle Level Set*

Todos os passos até então apresentados fundam a etapa de geração das fatias intermediárias, para o processo de reconstrução segundo o método *Level Set*. O método alternativo *Particle Level Set* baseia-se nessa mesma sequência de passos, diferenciando-se do original pela adição de três novos passos, conforme o fluxograma da Figura 3.3: sedimentação das partículas, atualização das partículas e correção do contorno via *Particle Level Set*.

Sedimentação das partículas

A sedimentação das partículas (Algoritmo 3.2), o primeiro passo particular ao *Particle Level Set*, é executada entre a determinação do passo temporal Δt (Subseção 3.3.4) e a metamorfose entre as fatias origem e destino (Subseção 3.3.5) e ocorre sobre a matriz função distância sinalizada $\phi^0 = \phi^0_{origem}$ relativa à fatia origem.

Primeiramente, percorre-se a matriz ϕ^0_{origem} e, para cada vértice (y, x) da grade (ou, equivalentemente, para cada elemento (y, x) da matriz), testa-se se algum dos quatro vértices da célula à qual pertence o elemento em análise – a saber, além de (y, x),

os vértices $(y, x+1)$, $(y+1, x+1)$ e $(y+1, x)$ (Figura 3.18) – encontra-se a uma distância máxima de três células do contorno. Matematicamente, equivale-se verificar se $|\phi^0_{origem}(j,i)| < 3 \cdot \max\{\Delta x, \Delta y\}$ para algum (j,i), com $j \in \{y, y+1\}$ e $i \in \{x, x+1\}$ e sendo Δx e Δy os espaçamentos da grade nas direções x e y, respectivamente.

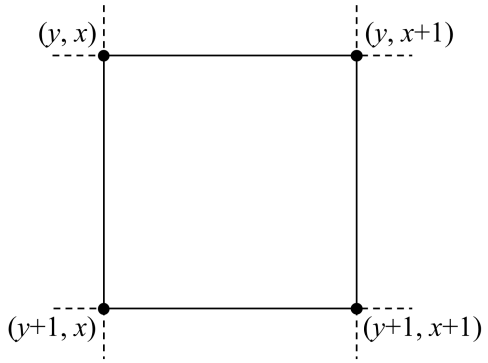


Figura 3.18: Vértices de uma célula da grade.

Em caso positivo, 16 partículas são, ao todo, depositadas no interior da célula em questão, sendo 8 positivas e 8 negativas [19]. Definem-se as coordenadas cartesianas de cada uma das partículas aleatoriamente, segundo as linhas de comando

```
X(i) + (X(i+1) - X(i))*rand(1,1)
Y(j) + (Y(j+1) - Y(j))*rand(1,1)
```

nas quais `X` e `Y` são os vetores que armazenam as coordenadas da grade nas direções x e y, respectivamente, e `rand`, o gerador de números aleatórios pertencente à biblioteca do MATLAB. A sintaxe `a + (b-a)*rand(1,1)` assegura a pertinência do número aleatório gerado ao intervalo `[a,b]`; logo, as linhas de comando apresentadas garantem a deposição das partículas dentro dos limites geométricos da célula.

Igualmente, o sinal (positivo ou negativo) das partículas é determinado de forma aleatória, utilizando uma combinação das funções `rand` e `sign` – esta, outra função pertencente à biblioteca do MATLAB, cuja saída indica o sinal do número passado como argumento. Assim, a execução

```
sign(-1 + 2*rand(1,1))
```

retorna o sinal de um número aleatório pertencente ao intervalo $[-1, 1]$. Sendo este intervalo simétrico e a função `rand` operante sobre uma distribuição uniforme, a probabilidade de se obter um sinal positivo é, do ponto de vista teórico, igual a de se obter um sinal negativo, o que assegura, dessa forma, que a determinação dos sinais das 16 partículas dentro de uma célula ocorra sob a ordem de $O(1)$.

Os sinais estabelecem em qual região do contorno presente na fatia origem, interna ou externa, as partículas, efetivamente, hão de situar-se, segundo o padrão da Equação 2.9.

Ou seja, partículas negativas devem localizar-se na região interna ao contorno, na qual $\phi^0_{origem} < 0$, e partículas positivas, na região externa, em que $\phi^0_{origem} > 0$. Dado que os sinais das partículas são estabelecidos aleatoriamente, partículas tanto positivas quanto negativas encontram-se, logo, em ambas as regiões interna e externa ao contorno, uma vez finalizada a deposição. Para adequar as posições das partículas conforme o padronizado, executa-se, na sequência, um procedimento de atração para a região apropriada da fatia, portanto.

Sejam, assim, as constantes v_{min} e v_{max} tais que

$$v_{min} = 0.1 \cdot \min\{\Delta x, \Delta y\} < v_{max} = 3 \cdot \max\{\Delta x, \Delta y\} \tag{3.15}$$

Para cada partícula positiva, seleciona-se um valor aleatório ϕ_{alvo}, com $\phi_{alvo} \in [v_{min}, v_{max}]$; analogamente, para cada partícula negativa, $\phi_{alvo} \in [-v_{max}, -v_{min}]$ é determinado aleatoriamente. Similarmente ao empregado durante o procedimento de deposição das partículas, geram-se os valores aleatórios ϕ_{alvo} pela execução da função `rand`.

As quantidades ϕ_{alvo} obtidas são, então, utilizadas na equação de atração

$$\vec{p}_{novo} = \vec{p}_{original} + \lambda \cdot [\phi_{alvo} - \phi^0_{origem}(\vec{p}_{original})] \cdot \vec{N}(\vec{p}_{original}) \tag{3.16}$$

sendo $\vec{p}_{novo} = (y_{novo}, x_{novo})$ as novas coordenadas da partícula; $\vec{p}_{original} = (y_{original}, x_{original})$ as coordenadas originais; λ um parâmetro de controle, inicialmente igual a 1; $\phi^0_{origem}(\vec{p}_{original})$ o valor interpolado de ϕ^0_{origem} para as coordenadas originais da partícula; e $\vec{N}(\vec{p}_{original})$ o vetor unitário normal a ϕ^0_{origem} no ponto $\vec{p}_{original}$, com

$$\vec{N}(y, x) = \frac{\vec{\nabla}\phi^0_{origem}(y, x)}{|\vec{\nabla}\phi^0_{origem}(y, x)|} \tag{3.17}$$

Resolve-se a Equação 3.16 iterativamente, para cada uma das partículas depositadas sobre a fatia. No início de cada iteração, determina-se o valor $\phi^0_{origem}(\vec{p}_{original})$, interpolado a partir da matriz ϕ^0_{origem}, por meio da função `interp2` pertencente à biblioteca do MATLAB. Do mesmo modo, interpolam-se as derivadas parciais (previamente calculadas sobre toda a grade da fatia), em ambas as direções x e y, para as coordenadas originais da partícula, cujos valores retornados são empregados na determinação do vetor unitário normal $\vec{N}(\vec{p}_{original})$, dado que $\vec{\nabla}\phi = (\phi_x, \phi_y)$ e $|\vec{\nabla}\phi| = \sqrt{\phi_x^2 + \phi_y^2}$. Nos testes realizados, foram utilizadas as aproximações por diferenças centrais para as derivadas, sendo possível o emprego das aproximações por diferenças progressivas ou regressivas de modo igual, sem prejuízo dos resultados práticos.

Na eventualidade de $\vec{N}(\vec{p}_{original})$, calculado a partir dos valores interpolados das derivadas parciais, ser nulo ou indeterminado (o que ocorre quando $\vec{\nabla}\phi = \vec{0} \Leftrightarrow |\vec{\nabla}\phi| = 0$), um vetor unitário arbitrário é, então, atribuído à posição da partícula em análise [49], como,

por exemplo, o vetor

$$\vec{N}_{\text{arbitrário}} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right) \tag{3.18}$$

De posse de todas as variáveis presentes na Equação 3.16, estabelece-se, assim, a nova posição $\vec{p} = \vec{p}_{novo}$ da partícula. Entretanto, se alguma das novas coordenadas não estiver dentro do domínio computacional da fatia (isto é, se $x_{novo} \notin [x_{min}, x_{max}]$ ou $y_{novo} \notin [y_{min}, y_{max}]$, para $[x_{min}, x_{max}]$ e $[y_{min}, y_{max}]$ os vetores nas respectivas direções x e y sobre os quais está definida a matriz da fatia), o parâmetro λ é dividido por dois e a Equação 3.16, repetida para este novo λ, mantendo-se, porém, as coordenadas originais $\vec{p}_{original}$ e os valores determinados para ϕ_{alvo}, $\phi^0_{origem}(\vec{p}_{original})$ e $\vec{N}(\vec{p}_{original})$.

Para as coordenadas $\vec{p}_{novo}$ retornadas pela reexecução da Equação 3.16, interpola-se o valor $\phi^0_{origem}(\vec{p}_{novo})$ e verifica-se se $\phi^0_{origem}(\vec{p}_{novo}) \in [v_{min}, v_{max}]$ ou $\phi^0_{origem}(\vec{p}_{novo}) \in [-v_{max}, -v_{min}]$, de acordo com o sinal da partícula, positiva ou negativa, respectivamente. Em caso positivo, estabelece-se $\vec{p} = \vec{p}_{novo}$ como a nova posição da partícula, independentemente de $\phi^0_{origem}(\vec{p}_{novo})$ ser, ou não, exatamente igual a ϕ_{alvo}; em caso negativo, a Equação 3.16, com o parâmetro λ dividido por dois novamente e as outras variáveis igualmente mantidas, é executada pela terceira vez, a iteração corrente é encerrada e as coordenadas $\vec{p}_{novo}$ obtidas passam a desempenhar o papel de $\vec{p}_{original}$ na iteração seguinte, reatribuindo-se, inclusive, o valor 1 ao parâmetro λ logo no início desta.

Limitou-se em 10 o número de iterações para o procedimento de atração de cada partícula. Se, ao final da décima iteração, a partícula não se encontra na região apropriada da fatia, descarta-se, por fim, esta partícula. Uma vez que as partículas possuem significado tão somente lógico, e não físico, a eventual eliminação de partículas incorretamente localizadas não afeta a corretude do processo de reconstrução e a qualidade dos resultados negativamente, sob quaisquer aspectos.

Depositadas e atraídas as partículas, o último procedimento relativo ao passo de sedimentação determina o raio r_p de cada uma delas, segundo a convenção

$$r_p = \begin{cases} r_{max}, & \text{se } s_p \cdot \phi^0_{origem}(\vec{p}) > r_{max} \\ s_p \cdot \phi^0_{origem}(\vec{p}), & \text{se } r_{min} \leq s_p \cdot \phi^0_{origem}(\vec{p}) \leq r_{max} \\ r_{min}, & \text{se } s_p \cdot \phi^0_{origem}(\vec{p}) < r_{min} \end{cases} \tag{3.19}$$

na qual $r_{min} = 0.1 \cdot \min\{\Delta x, \Delta y\}$, $r_{max} = 0.5 \cdot \min\{\Delta x, \Delta y\}$, s_p indica o sinal da partícula (+1 para partículas positivas e -1 para negativas) e $\phi^0_{origem}(\vec{p})$, o valor interpolado de ϕ^0_{origem} para a posição atraída $\vec{p}$ da partícula. Essa padronização assegura a tangência entre a borda da partícula e o contorno presente na fatia sempre que possível, possibilitando, ainda, uma amostragem multiescala do contorno a partir do conjunto de partículas [20].

Todas as informações acerca de cada partícula gerada são armazenadas em um vetor de estruturas, cada elemento contendo um vetor bidimensional do tipo `float` para as co-

ordenadas x e y da partícula, além de duas variáveis, também do tipo `float`, sendo uma para o sinal da partícula e a outra, para o raio. A Figura 3.19 esquematiza a construção descrita.

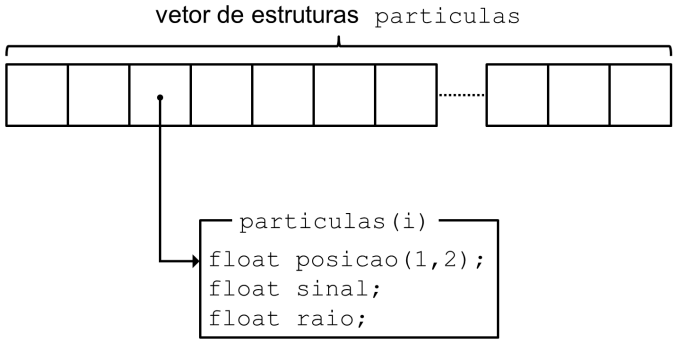


Figura 3.19: Vetor de estruturas `particulas`.

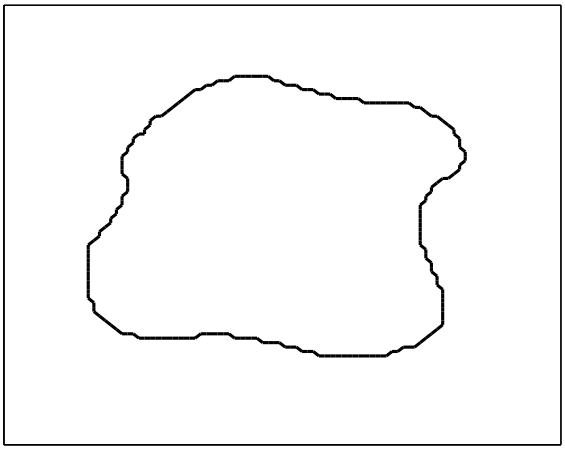
Ilustrando a dinâmica geométrica do presente passo, a Figura 3.20(a) apresenta a fatia original, sobre a qual serão depositadas as partículas. Na sequência, a Figura 3.20(b) contempla a disposição das partículas imediatamente após a deposição destas, sendo visível, pela lente de aumento central, a não uniformidade da localização das partículas (positivas em azul, negativas em verde) com relação às regiões interna e externa do contorno. Com a Equação de atração 3.16, as partículas são, enfim, atraídas para a região correta da fatia, de acordo com os sinais estabelecidos para cada uma, conforme visualizado na Figura 3.20(c) – como todo método numérico, o procedimento de atração das partículas não é integralmente preciso e, ao final da execução, uma quantidade relativamente pequena das partículas pode, eventualmente, localizar-se em regiões incorretas.

Atualização das partículas

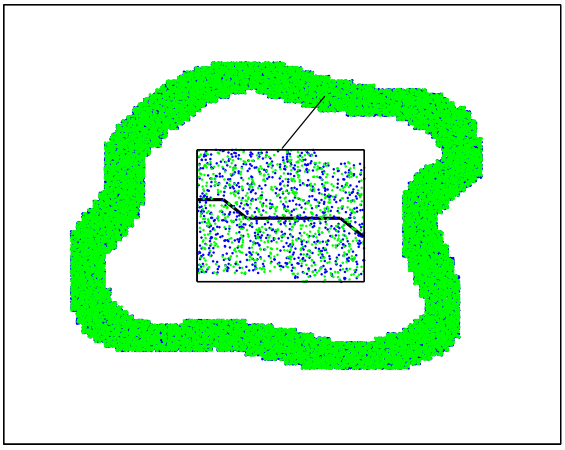
O passo seguinte implementado pelo método *Particle Level Set* desenvolve-se dentro do contexto da metamorfose entre as fatias origem e destino, posteriormente à resolução numérica da Equação 2.13 da iteração corrente. Em outras palavras, a atualização das partículas, na n-ésima iteração da metamorfose, ocorre imediatamente após a obtenção da matriz função distância sinalizada ϕ^n_{origem}.

A atualização das partículas objetiva adequar a posição de cada uma delas à nova configuração da fatia origem, descrita pela matriz ϕ^n_{origem}. Cada uma das partículas, logo, é evoluída de acordo com a equação diferencial parcial

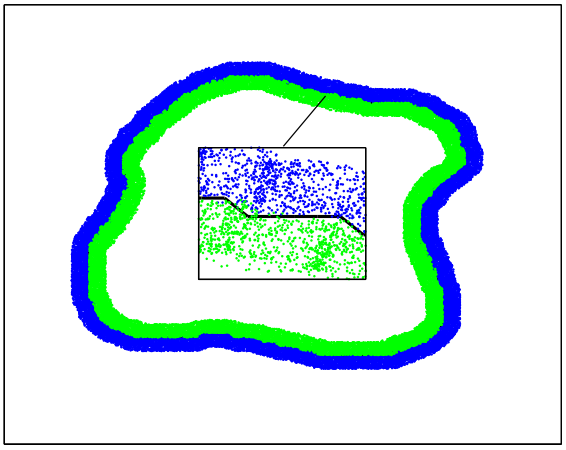
$$\vec{p}_t = \frac{d\vec{p}}{dt} = F(\vec{p}) \cdot \vec{N}(\vec{p}) \tag{3.20}$$



(a) Fatia original;



(b) Disposição das partículas após a deposição;



(c) Disposição das partículas após a atração.

Figura 3.20: Sedimentação das partículas.

Algoritmo 3.2: Sedimentação das partículas

Entrada: Matriz ϕ^0_{origem}, espaçamentos Δx e Δy
Saída: Vetor de estruturas `particulas`

1: Inicializar o vetor de estruturas `particulas` (Figura 3.19)
2: **for all** célula da grade de ϕ^0_{origem} **do**
3: Determinar os vértices $\{\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4\}$ da célula
4: **if** $\exists\, \vec{v} \in \{\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4\}$ tal que $|\phi^0_{origem}(\vec{v})| < 3 \cdot \max\{\Delta x, \Delta y\}$ **then**
5: $positivas \leftarrow 8$, $negativas \leftarrow 8$
6: **while** $positivas > 0$ **or** $negativas > 0$ **do**
7: Gerar, aleatoriamente, as coordenadas $\vec{p}_{original}$ e o sinal s_p da nova partícula e armazenar esses dados nos campos correspondentes do elemento do vetor de estruturas `particulas` referente à partícula criada
8: **if** (s_p é positivo **and** $positivas \leq 0$) **or** (s_p é negativo **and** $negativas \leq 0$) **then**
9: Inverter s_p
10: **end if**
11: **if** s_p é positivo **then**
12: $positivas \leftarrow positivas - 1$
13: **else**
14: $negativas \leftarrow negativas - 1$
15: **end if**
16: **end while**
17: **end if**
18: **end for**
19: $v_{min} \leftarrow 0.1 \cdot \min\{\Delta x, \Delta y\}$, $v_{max} \leftarrow 3 \cdot \max\{\Delta x, \Delta y\}$
20: **for all** partícula $\in$ `particulas` **do**
21: **if** s_p é positivo **then**
22: Selecionar, aleatoriamente, um valor $\phi_{alvo} \in [v_{min}, v_{max}]$
23: **else**
24: Selecionar, aleatoriamente, um valor $\phi_{alvo} \in [-v_{max}, -v_{min}]$
25: **end if**
26: $iteracao \leftarrow 1$
27: $posicao \leftarrow \vec{p}_{original}$
28: **while** $iteracao \leq 10$ **do**
29: Calcular $\phi_{interp} = \phi^0_{origem}(posicao)$ e $\vec{\nabla}\phi^0_{origem}(posicao)$, o valor interpolado e o gradiente de ϕ^0_{origem}, respectivamente, para as coordenadas $posicao$

Algoritmo 3.2: Sedimentação das partículas

if $\vec{\nabla}\phi^0_{origem}(posicao) \neq \vec{0}$ **then**
  $\vec{N} \leftarrow \dfrac{\vec{\nabla}\phi^0_{origem}(posicao)}{|\vec{\nabla}\phi^0_{origem}(posicao)|}$
else
  $\vec{N} \leftarrow \left(\dfrac{1}{\sqrt{2}}, \dfrac{1}{\sqrt{2}}\right)$
end if
$\vec{p}_{novo} \leftarrow posicao + (\phi_{alvo} - \phi_{interp}) \cdot \vec{N}$
if alguma das coordenadas de $\vec{p}_{novo}$ não estiver dentro do domínio computacional de ϕ^0_{origem} **then**
  $\vec{p}_{novo} \leftarrow posicao + 0.5 \cdot (\phi_{alvo} - \phi_{interp}) \cdot \vec{N}$
  Calcular $\phi_{aux} = \phi^0_{origem}(\vec{p}_{novo})$, o valor interpolado de ϕ^0_{origem} para as coordenadas $\vec{p}_{novo}$
  if (s_p é positivo **and** $\phi_{aux} \notin [v_{min}, v_{max}]$) **or** ($s_p$ é negativo **and** $\phi_{aux} \notin [-v_{max}, -v_{min}]$) **then**
    $posicao \leftarrow posicao + 0.25 \cdot (\phi_{alvo} - \phi_{interp}) \cdot \vec{N}$
    $iteracao \leftarrow iteracao + 1$
  else
    $posicao \leftarrow \vec{p}_{novo}$
    break
  end if
else
  $posicao \leftarrow \vec{p}_{novo}$
  break
end if
end while
if $iteracao > 10$ **then**
  Descartar a partícula
else
  Armazenar $posicao$ no campo `posicao` do elemento do vetor de estruturas `particulas` referente à partícula analisada
end if
end for
$r_{min} \leftarrow 0.1 \cdot \min\{\Delta x, \Delta y\}$, $r_{max} \leftarrow 0.5 \cdot \min\{\Delta x, \Delta y\}$
for all partícula $\in$ `particulas` **do**

Algoritmo 3.2: Sedimentação das partículas

59: Calcular $\phi_{interp} = \phi^0_{origem}(\vec{p})$, o valor interpolado de ϕ^0_{origem} para as coordenadas $\vec{p}$ da partícula
60: **if** $s_p \cdot \phi_{interp} > r_{max}$ **then**
61: $r_p \leftarrow r_{max}$
62: **else if** $s_p \cdot \phi_{interp} \geq r_{min}$ **then**
63: $r_p \leftarrow s_p \cdot \phi_{interp}$
64: **else**
65: $r_p \leftarrow r_{min}$
66: **end if**
67: Armazenar r_p no campo `raio` do elemento do vetor de estruturas `particulas` referente à partícula analisada
68: **end for**
69: **return** vetor de estruturas `particulas`

numericamente reescrita como

$$\frac{\vec{p}^{\,n} - \vec{p}^{\,n-1}}{\Delta t} = F(\vec{p}^{\,n-1}) \cdot \vec{N}(\vec{p}^{\,n-1}) \tag{3.21}$$

na qual $\vec{p}^{\,n}$ são as coordenadas da partícula após a evolução da metamorfose (equivalentemente, ao final da n-ésima iteração) a serem determinadas; $\vec{p}^{\,n-1}$, as coordenadas antes da evolução (no início da n-ésima iteração); Δt, o passo temporal calculado na Subseção 3.3.4; $F(\vec{p}^{\,n-1})$, o valor da velocidade normal F interpolado para as coordenadas $\vec{p}^{\,n-1}$; e $\vec{N}(\vec{p}^{\,n-1})$, o vetor unitário normal a ϕ^{n-1}_{origem} no ponto $\vec{p}^{\,n-1}$.

Assim como no passo de sedimentação anterior, faz-se necessária, para a determinação do vetor $\vec{N}(\vec{p}^{\,n-1})$, a interpolação das derivadas parciais ϕ_x e ϕ_y para a posição $\vec{p}^{\,n-1}$, estabelecendo-se o vetor dado pela Equação 3.18 como o vetor unitário normal arbitrário em $\vec{p}^{\,n-1}$, caso $\vec{N}(\vec{p}^{\,n-1})$ seja nulo ou indeterminado, da mesma forma. A inserção de $\vec{N}(\vec{p}^{\,n-1})$ na Equação 3.21 é essencial para garantir que a velocidade F se propague, efetivamente, na direção normal a ϕ^{n-1}_{origem}.

Correção do contorno via *Particle Level Set*

Imediatamente após a atualização das partículas, implementa-se a correção do contorno via *Particle Level Set* (Algoritmo 3.3), por meio da identificação das partículas ditas escapadas, isto é, as quais, após a atualização de suas posições, localizam-se em regiões incorretas da fatia intermediária gerada (a saber, partículas positivas em regiões

com $\phi^n_{origem} < 0$ e negativas em regiões com $\phi^n_{origem} > 0$) e a uma distância do contorno estritamente maior do que a magnitude dos respectivos raios. Tais condições são testadas a partir do valor interpolado de ϕ^n_{origem} para as coordenadas $\vec{p}^{\,n}$ de cada uma das partículas sedimentadas e atualizadas.

Para cada partícula escapada, define-se a função *Level Set* local $\phi_p(\vec{v})$

$$\phi_p(\vec{v}) = s_p \cdot (r_p - |\vec{v} - \vec{p}^{\,n}|) \tag{3.22}$$

em que s_p é o sinal, r_p é o raio e $\vec{p}^{\,n}$ são as coordenadas da partícula escapada, conforme estabelecido; e $\vec{v}$ são as coordenadas de um vértice qualquer da grade da matriz ϕ^n_{origem}. A Equação 3.22, calculada para cada um dos quatro vértices $\vec{v}$ da célula à qual pertence a partícula escapada, fornece os dados para a verificação de erros na matriz ϕ^n_{origem}: qualquer variação que haja entre $\phi_p(\vec{v})$ e o valor $\phi^n_{origem}(\vec{v})$ correspondente indica a possível existência de erros nos resultados numéricos presentes na matriz ϕ^n_{origem} [18].

Porém, o contorno resultante do procedimento de correção empregando-se a Equação 3.22 pode, eventualmente, apresentar quebra de suavidade em (regiões específicas de) sua forma, o que, por consequência, prejudica a determinação numérica de propriedades e ferramentas geométricas intrínsecas, como, por exemplo, os vetores unitários normais $\vec{N}$ [69]. Para contornar essa questão, adapta-se a lei da função *Level Set* local $\phi_p(\vec{v})$, sendo essa nova formulação enunciada como

$$\phi_p(\vec{v}) = \begin{cases} s_p \cdot r_p - |\vec{v} - \vec{p}^{\,n}|, & \text{se } \phi^n_{origem}(\vec{v}) \leq 0 \\ s_p \cdot r_p + |\vec{v} - \vec{p}^{\,n}|, & \text{se } \phi^n_{origem}(\vec{v}) > 0 \end{cases} \tag{3.23}$$

A detecção e a consequente correção dos erros numéricos, com base na Equação 3.23, iniciam-se com a declaração das matrizes auxiliares ϕ^- e ϕ^+, atribuindo a ambas a matriz ϕ^n_{origem} como valor inicial. O vetor de estruturas que armazena as informações de todas as partículas sedimentadas e atualizadas é, então, percorrido e, para cada partícula considerada escapada, determinam-se as coordenadas de cada um dos quatro vértices da célula à qual pertence a partícula analisada.

Para tanto, sejam `ceil` a função pertencente à biblioteca do MATLAB que retorna o teto de seu argumento; `particulas` o nome atribuído ao vetor de estruturas contendo os dados sobre as partículas e `posicao` ao vetor bidimensional no qual armazenam-se as coordenadas cartesianas de cada partícula (Figura 3.19); e `X` e `Y` os vetores que reúnem as coordenadas da grade nas direções x e y e com espaçamento `dx` e `dy`, respectivamente. Os comandos

```
ceil((particulas(i).posicao(1) - X(1))/dx)
ceil((particulas(i).posicao(2) - Y(1))/dy)
```

estabelecem, logo, as respectivas coordenadas x e y do vértice superior esquerdo da célula, a partir do qual são determinados os três vértices restantes (Figura 3.18).

Na ocasião de a partícula escapada ser positiva, calculam-se, pela Equação 3.23, os valores da função *Level Set* local $\phi_p(\vec{v})$ para cada um dos quatro vértices $\vec{v}$ da célula e o máximo local entre $\phi_p(\vec{v})$ e $\phi^+(\vec{v})$ é, então, atribuído como o novo valor de $\phi^+(\vec{v})$. Equivalentemente, em termos matemáticos

$$\phi^+(\vec{v}) = \max\{\phi^+(\vec{v}), \phi_p(\vec{v})\} \tag{3.24}$$

Para partículas escapadas negativas, emprega-se o mínimo local entre $\phi_p(\vec{v})$ e $\phi^-(\vec{v})$

$$\phi^-(\vec{v}) = \min\{\phi^-(\vec{v}), \phi_p(\vec{v})\} \tag{3.25}$$

com $\phi_p(\vec{v})$ igualmente dado pela Equação 3.23.

Ao término da varredura, corrige-se, por fim, a matriz ϕ^n_{origem}, selecionando-se, para cada elemento (y, x) da matriz, o valor de menor magnitude entre $\phi^-(y, x)$ e $\phi^+(y, x)$, ou seja

$$\phi^n_{origem}(y, x) = \begin{cases} \phi^+(y, x), & \text{se } |\phi^+(y, x)| \leq |\phi^-(y, x)| \\ \phi^-(y, x), & \text{se } |\phi^+(y, x)| > |\phi^-(y, x)| \end{cases} \tag{3.26}$$

Corrigida a matriz ϕ^n_{origem} via *Particle Level Set*, prossegue-se a etapa de geração das fatias intermediárias, com a execução do passo de armazenamento em memória da fatia (Subseção 3.3.6), de acordo com o fluxograma da Figura 3.3.

Definição do roteiro de implementação

Tradicionalmente, o roteiro de implementação do método *Particle Level Set* executa, além dos passos descritos, os adicionais de ajustamento e ressedimentação das partículas [18]: enquanto o primeiro redefine o raio de cada partícula após a correção do contorno, o segundo, fundamentado na nova geometria da interface, redistribui as partículas, adicionando novas ou excluindo as numericamente deterioradas, quando necessário.

Erros, contudo, são suscetíveis de serem inseridos nos dados do problema com a inclusão desses passos complementares [43]. Como todo procedimento numérico, há uma probabilidade não desprezível de o contorno não estar acuradamente localizado após cada iteração da metamorfose e o ajustamento e a ressedimentação das partículas pode, consequentemente, nelas introduzir essa inacurácia, deflagrando, assim, um fluxo propagativo de erros. Por essas razões, optou-se pelo roteiro sem a adição desses passos para a implementação do método *Particle Level Set*.

3.3.14 Saídas geradas

À conclusão da etapa de geração das fatias intermediárias, três saídas terão sido geradas. A figura MATLAB com a totalidade das fatias, originais e intermediárias, empilhadas

Algoritmo 3.3: Correção do contorno via *Particle Level Set*

Entrada: Matriz ϕ^n_{origem} e vetor de estruturas `particulas` (Figura 3.19)
Saída: Matriz ϕ^n_{origem} corrigida via *Particle Level Set*

$\phi^- \leftarrow \phi^n_{origem}$, $\phi^+ \leftarrow \phi^n_{origem}$
for all partícula $\in$ `particulas` **do**
  if o sinal de $\phi^n_{origem}(\vec{p}^{\,n})$, o valor interpolado de ϕ^n_{origem} para as coordenadas $\vec{p}^{\,n}$ da partícula, é diferente do sinal s_p da partícula **and** $|\phi^n_{origem}(\vec{p}^{\,n})|$ é maior do que o raio r_p **then**
    Determinar os vértices $\{\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4\}$ da célula à qual pertence a partícula
    if s_p é positivo **then**
      for all $\vec{v} \in \{\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4\}$ **do**
        if $\phi^n_{origem}(\vec{v}) \leq 0$ **then**
          $\phi_p(\vec{v}) \leftarrow s_p \cdot (r_p - |\vec{v} - \vec{p}^{\,n}|)$
        else
          $\phi_p(\vec{v}) \leftarrow s_p \cdot (r_p + |\vec{v} - \vec{p}^{\,n}|)$
        end if
        $\phi^+(\vec{v}) \leftarrow \max\{\phi^+(\vec{v}), \phi_p(\vec{v})\}$
      end for
    else
      for all $\vec{v} \in \{\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4\}$ **do**
        if $\phi^n_{origem}(\vec{v}) \leq 0$ **then**
          $\phi_p(\vec{v}) \leftarrow s_p \cdot (r_p + |\vec{v} - \vec{p}^{\,n}|)$
        else
          $\phi_p(\vec{v}) \leftarrow s_p \cdot (r_p - |\vec{v} - \vec{p}^{\,n}|)$
        end if
        $\phi^-(\vec{v}) \leftarrow \min\{\phi^-(\vec{v}), \phi_p(\vec{v})\}$
      end for
    end if
  end if
end for
for all elemento (y, x) de ϕ^n_{origem} **do**
  if $|\phi^+(y, x)| \leq |\phi^-(y, x)|$ **then**
    $\phi^n_{origem}(y, x) = \phi^+(y, x)$
  else
    $\phi^n_{origem}(y, x) = \phi^-(y, x)$
  end if
end for

em suas alturas respectivas no espaço coordenado cartesiano (Subseção 3.3.11) compõe uma saída global do processo de reconstrução; o arquivo texto contendo as alturas de cada uma das fatias (Subseção 3.3.8) e o conjunto das fatias intermediárias geradas armazenadas em arquivo *bitmap* monocromático (Subseção 3.3.12) constituem a saída local do processo, isto é, os dados de entrada da etapa seguinte de reconstrução.

3.4 Etapa 4: Reconstrução

O *script* utilizado para a implementação do procedimento de reconstrução – modelado em linguagem Python por Paulo Henrique Junqueira Amorim e Thiago Franco de Moraes, ambos pesquisadores do Centro de Tecnologia da Informação Renato Archer [9] – emprega a técnica *Marching Cubes* [38] para a construção de superfícies tridimensionais, por meio da importação do módulo de código aberto VTK [32].

A partir da entrada composta pelo conjunto de fatias originais e intermediárias geradas e pelo arquivo texto com as alturas respectivas, o *script* retorna a superfície requerida reconstruída, saída esta que sinaliza a conclusão do processo global de reconstrução.

3.5 Procedimentos Otimizacionais

A natureza euleriana dos métodos *Level Set* e *Particle Level Set* exige uma elevada quantidade de cálculos para a resolução numérica das equações diferenciais parciais àqueles intrínsecos. Por consequência, a implementação computacional de ambos os métodos é altamente custosa – custo este relevantemente majorado, em particular ao segundo, em decorrência das operações envolvendo as partículas a ele inerentes, geralmente presentes na ordem dos milhares. Para aperfeiçoar o desempenho de ambos os métodos, procedimentos otimizacionais foram, logo, introduzidos ao código do processo de reconstrução.

O MATLAB disponibiliza um grupo de ferramentas que permite a computação em paralelo, em computadores com processadores *multicore*. Dentre aquelas, empregaram-se a função `matlabpool`, necessária para a ativação da funcionalidade, e o comando `parfor`, o qual permite o processamento em paralelo das iterações de *for-loops*, distribuindo-as pelo conjunto de núcleos do processador. Em testes realizados, observou-se, com a adaptação otimizacional, um aprimoramento no tempo de execução da etapa de metamorfose de, aproximadamente, 25% para o método *Level Set* e 35% para o *Particle Level Set*.

Capítulo 4

Resultados Experimentais

Para a verificação prática da metodologia descrita, seis conjuntos de fatias foram utilizados. Cada um destes originou dois testes, o primeiro aplicando o método *Level Set* e o segundo, a variante *Particle Level Set*, totalizando doze execuções experimentais. A Tabela 4.1 detalha as configurações do computador empregado para a realização dos testes.

Sistema operacional	Windows 7 Home Premium 64 bits [42]
Processador	Intel Core i7 1.73GHz [30]
Memória RAM	6GB
Placa gráfica	NVIDIA GeForce GT 335M [47]

Tabela 4.1: Configurações do computador.

4.1 Testes com Dados Sintéticos

A fim de analisar o comportamento da metodologia ante fatias amostradas de algum modo falho, os testes com dados sintéticos, cujas métricas são pormenorizadas na Tabela 4.2, foram desenhados simulando as singularidades salientadas na Seção 1.2.

Sete fatias com contornos retangulares de dimensões decrescentes (Figura 4.1) compuseram o teste Pirâmide. Como ilustrado anteriormente na Figura 1.1, a evolução pelo método *Level Set* de fronteiras dinâmicas com cantos ou extensões pontiagudas suaviza em demasia tais características, justificando a aplicação alternativa da metamorfose via *Particle Level Set*, cujo propósito reside na contenção, ou ao menos no controle, dessa suavização, de modo a preservar atributos e aspectos finos da interface durante a sua evolução. A configuração do teste Pirâmide alvejou, logo, analisar a capacidade real da variante *Particle Level Set* para a satisfação desse objetivo proposto.

Métricas	**Conjuntos de fatias**		
	Pirâmide	Castiçal	Vaso
Número de fatias	7	4	3
Dimensões (altura×largura) das fatias (pixels)	70×100	70×100	125×125
Distância vertical entre as fatias	uniforme: 40	50-50-100	uniforme: 75
Número de pixels por fatia			
Mínimo	44	306	340
Máximo	278	648	424
Média no conjunto	134.29	544.50	368.00

Tabela 4.2: Métricas dos conjuntos de fatias sintéticas.

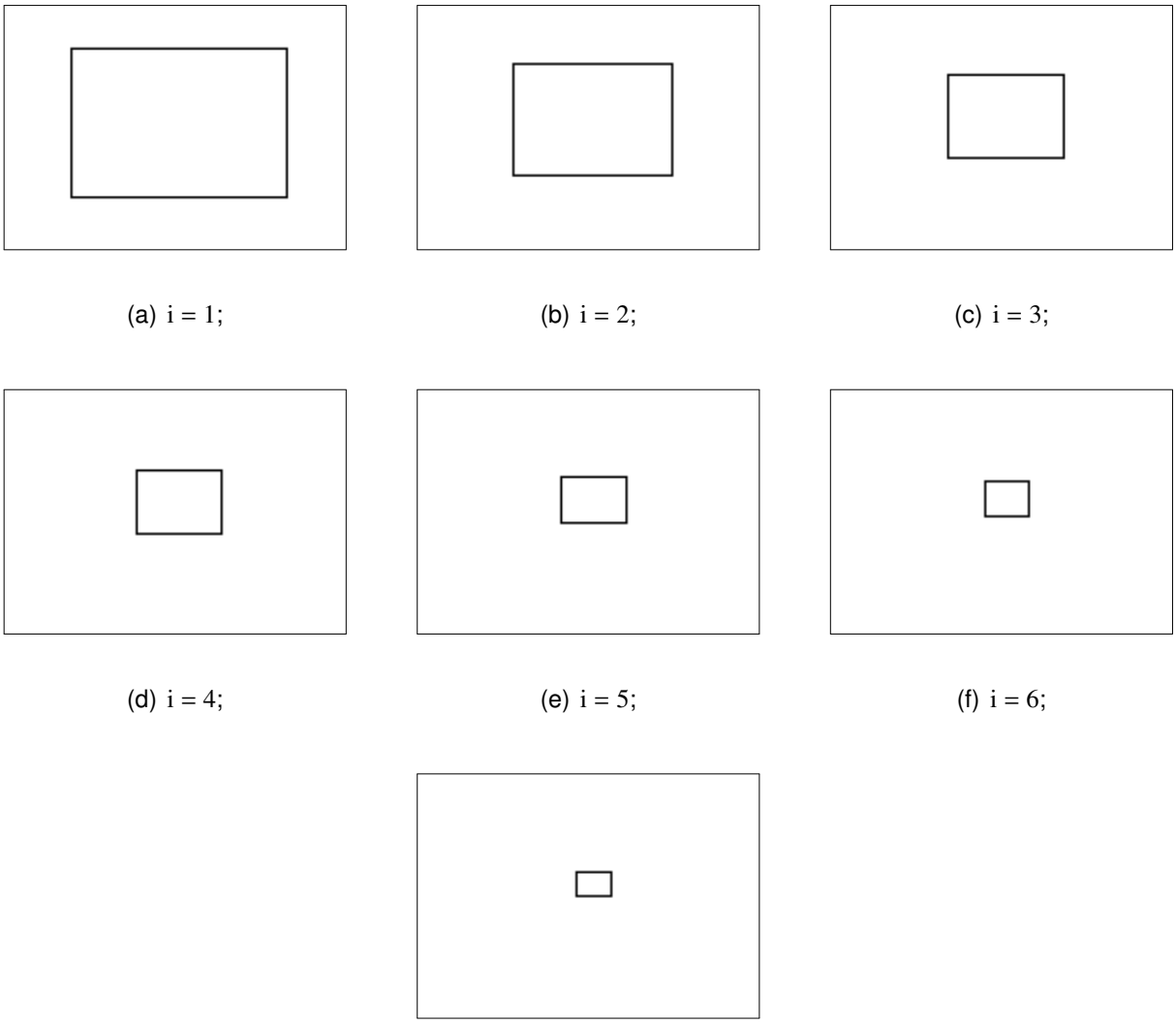
As Figuras 4.2 e 4.3 exibem amostras das fatias intermediárias provenientes da metamorfose entre a primeira e a segunda fatias do caso Pirâmide, mediante execução respectiva do *Level Set* e do *Particle Level Set*. Um exame minucioso do exposto, com ênfase na comparação entre as subfiguras (a) e (h) correspondentes, revela a sutileza do *Particle Level Set* na preservação dos cantos do contorno retangular sob deformação. Entretanto, esse controle na suavização deu-se por um procedimento de metamorfose relativamente custoso, conforme estatísticas mostradas na Tabela 4.3, para cada método.

Estatísticas	**Método**	
	Level Set	*Particle Level Set*
Total de fatias intermediárias geradas	223	250
Tempo médio (s) de geração de cada fatia intermediária	1.60	20.68

Tabela 4.3: Estatísticas dos testes – Pirâmide.

Restauradas as superfícies, as sutis diferenças contempladas entre as fatias oriundas do *Level Set* e do *Particle Level Set* tornaram-se praticamente imperceptíveis, tridimensionalmente. Uma avaliação comparativa das Figuras 4.4(b) e 4.4(c) ilustra tal constatação.

O teste Castiçal objetivou investigar a adaptabilidade da metodologia proposta para o tratamento de fatias com mudanças topológicas acentuadas. Estudando as fatias originais do teste, presentes na Figura 4.5, pressupõe-se que os dois contornos circulares iniciais, sob a ação da metamorfose, deformam-se continuamente até a união completa em um único, seguindo este o caminho inverso na sequência, quebrando-se em dois e regressando à topologia de início, a qual, por fim, transmuda-se até a divisão final em três contornos circulares. A observação analítica das Figuras 4.7(b) e 4.7(c), representativas das

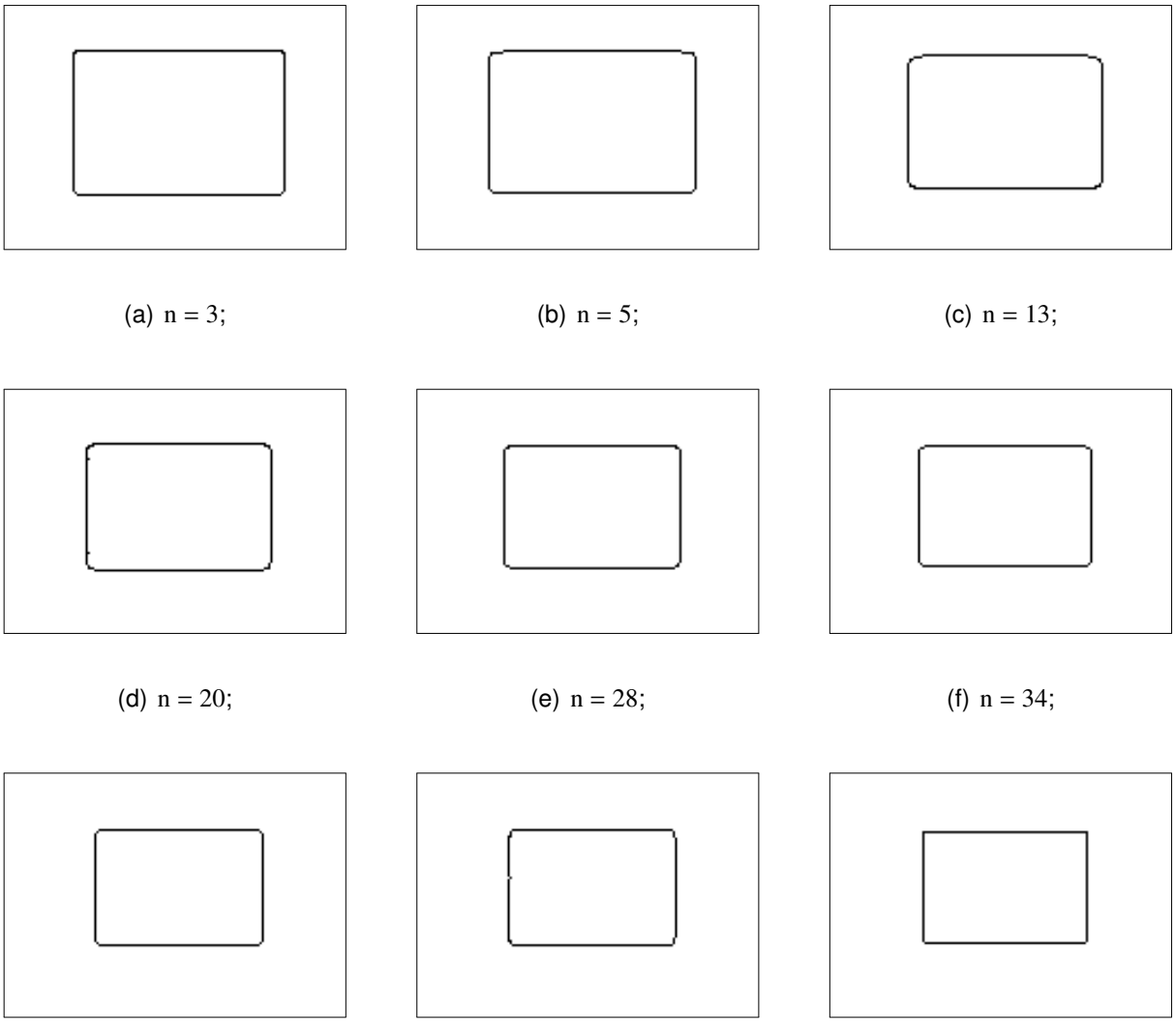


(a) i = 1; (b) i = 2; (c) i = 3;

(d) i = 4; (e) i = 5; (f) i = 6;

(g) i = 7.

Figura 4.1: Fatias originais – Pirâmide.

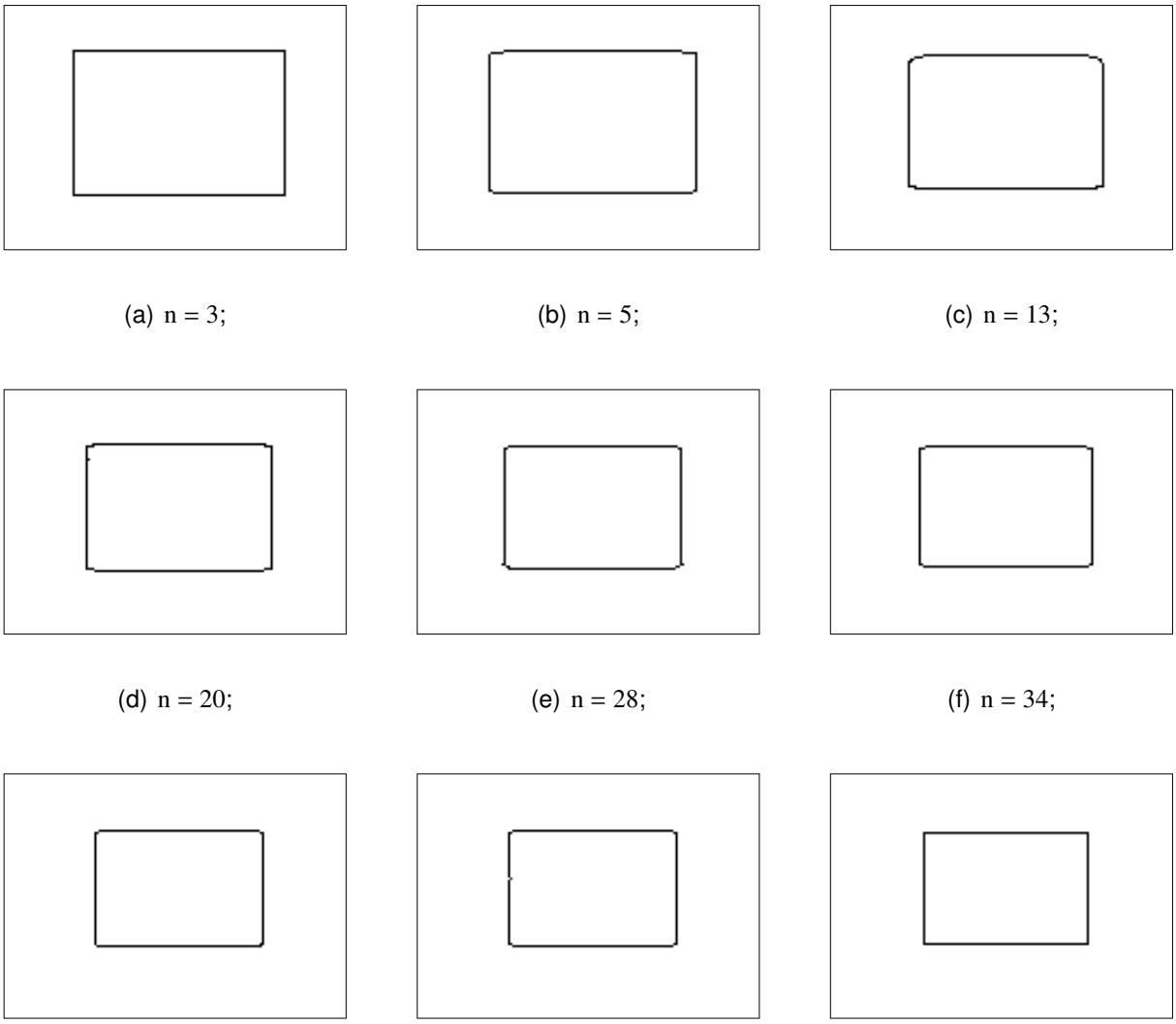


(a) n = 3; (b) n = 5; (c) n = 13;

(d) n = 20; (e) n = 28; (f) n = 34;

(g) n = 47; (h) n = 50; (i) n = 65.

Figura 4.2: Evolução via *Level Set* – Pirâmide.

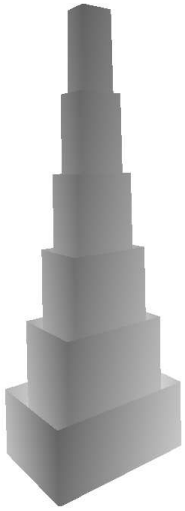


(a) n = 3; (b) n = 5; (c) n = 13;

(d) n = 20; (e) n = 28; (f) n = 34;

(g) n = 47; (h) n = 50; (i) n = 65.

Figura 4.3: Evolução via *Particle Level Set* – Pirâmide.



(a) Somente fatias originais;

(b) Metamorfose via *Level Set*;

(c) Metamorfose via *Particle Level Set*.

Figura 4.4: Superfícies reconstruídas – Pirâmide.

superfícies reconstruídas para o caso Castiçal, estabelece a asserção dessa conjectura.

A similaridade entre as reconstruções retornadas pelos métodos *Level Set* e *Particle Level Set* igualmente é reafirmada. Porém, uma análise mais criteriosa das respectivas Figuras 4.7(b) e 4.7(c) constata a capacidade da variante em controlar a velocidade da metamorfose, o que possibilitou o desenho da base do objeto com uma maior verossimilhança, sobressaindo-se, consequentemente, a segunda superfície perante a primeira. Pela Tabela 4.4, contudo, nota-se o elevado custo do *Particle Level Set* comparativamente ao *Level Set*, reiterando o atestado anteriormente pelo teste Pirâmide.

Estatísticas	**Método**	
	Level Set	*Particle Level Set*
Total de fatias intermediárias geradas	751	504
Tempo médio (s) de geração de cada fatia intermediária	1.20	23.27

Tabela 4.4: Estatísticas dos testes – Castiçal.

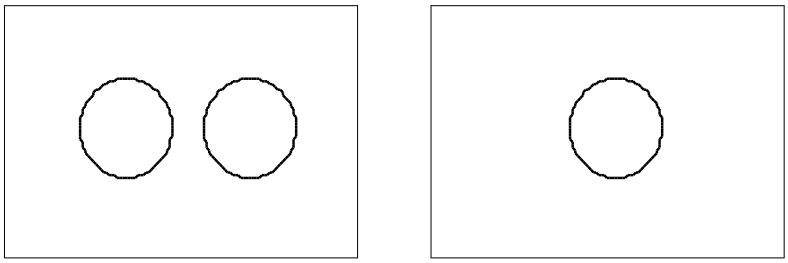
Avaliar problemas de reconstrução em que há fatias com contornos internos a outros foi o intuito do teste Vaso, o terceiro e último caso com dados sintéticos analisado, cujas fatias originais são exibidas na Figura 4.6. Na evolução entre a primeira e a segunda fatias, enquanto o círculo externo metamorfosea-se a fim de assumir a forma quadrangular, o quadrado interno transmuta-se no formato circular, revertendo-se ambos às correspondentes topologias originais durante a metamorfose entre a segunda e a terceira fatias.

Examinando a região central das Figuras 4.8(b) e 4.8(c), verifica-se, nesta, uma maior suavidade no ritmo de deformação entre os contornos externos das duas últimas fatias. Logo, as apreciações feitas no caso anterior, acerca do relativo controle da velocidade da metamorfose via *Particle Level Set* observado, são, da mesma forma, sustentadas pelo teste Vaso, assim como o alto custo computacional impresso pelo método (Tabela 4.5).

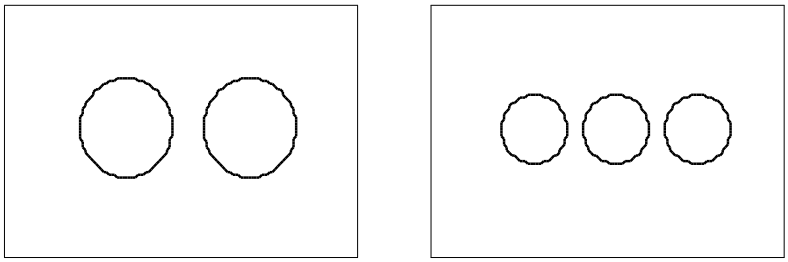
Estatísticas	**Método**	
	Level Set	*Particle Level Set*
Total de fatias intermediárias geradas	352	236
Tempo médio (s) de geração de cada fatia intermediária	1.71	51.11

Tabela 4.5: Estatísticas dos testes – Vaso.

Além das averiguações prévias, a bateria de testes com dados sintéticos pôde, com sucesso, comprovar a habilidade de ambos os métodos para a reconstrução de superfícies

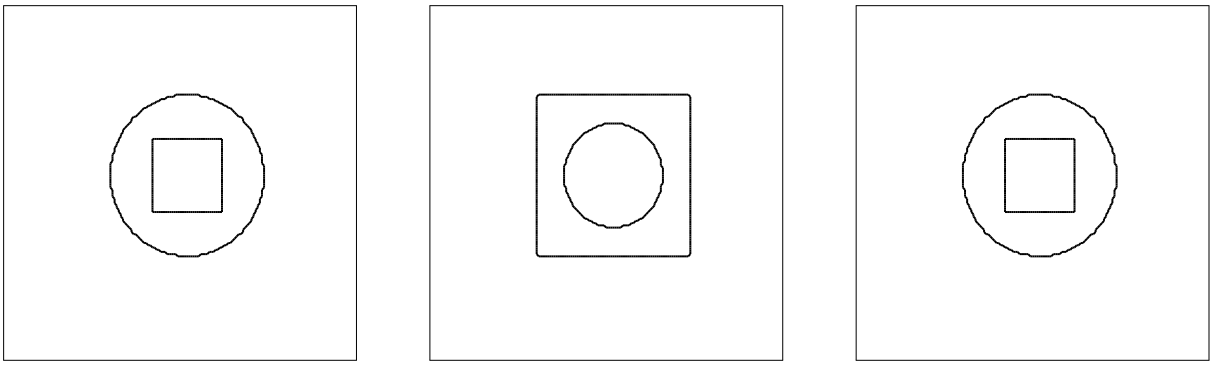


(a) i = 1; (b) i = 2;



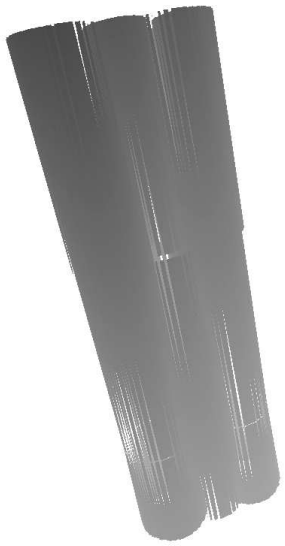
(c) i = 3; (d) i = 4.

Figura 4.5: Fatias originais – Castiçal.

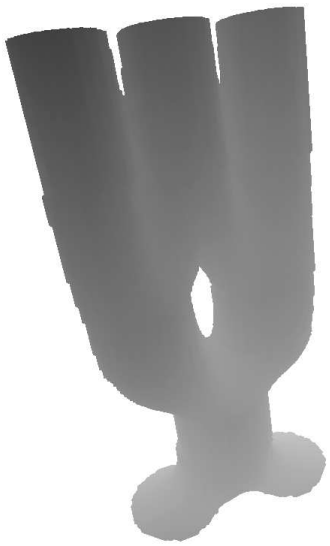


(a) i = 1; (b) i = 2; (c) i = 3.

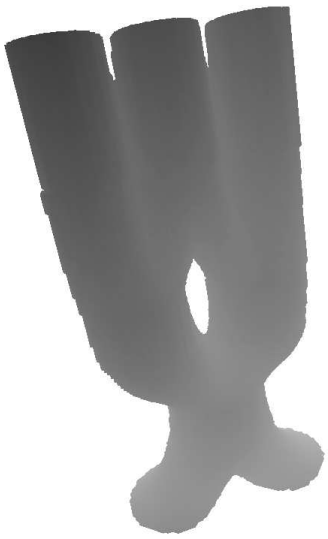
Figura 4.6: Fatias originais – Vaso.



(a) Somente fatias originais;

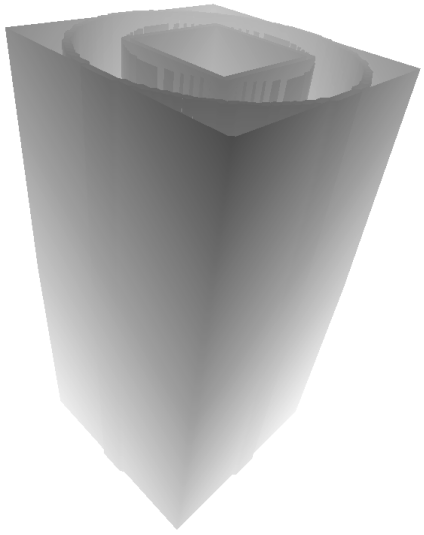


(b) Metamorfose via *Level Set*;

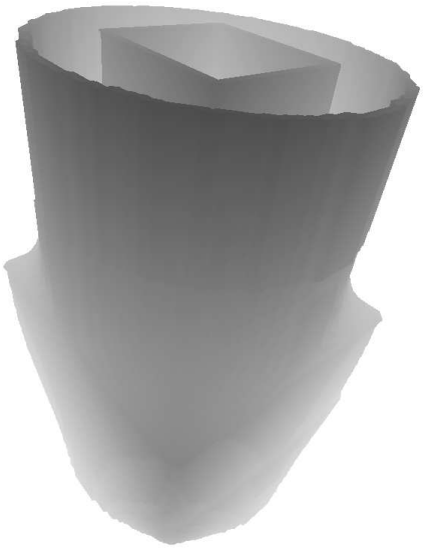


(c) Metamorfose via *Particle Level Set*.

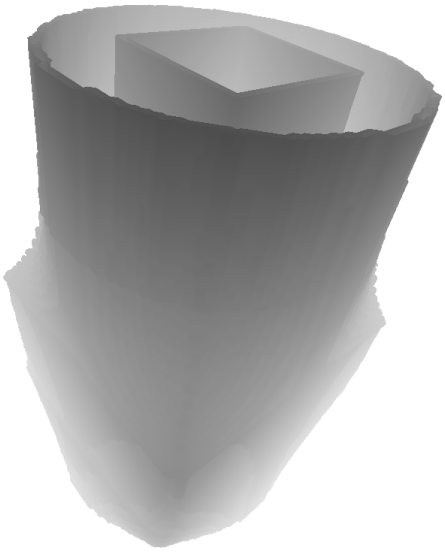
Figura 4.7: Superfícies reconstruídas – Castiçal.



(a) Somente fatias originais;



(b) Metamorfose via *Level Set*;



(c) Metamorfose via *Particle Level Set*.

Figura 4.8: Superfícies reconstruídas – Vaso.

a partir de fatias planas paralelas, quando a amostragem destas evidencia-se defectiva de alguma forma. Comparando, em cada uma das Figuras 4.4, 4.7 e 4.8, os objetos reconstruídos a partir das fatias originais tão somente com os retornados pela metodologia proposta, atesta-se a evolução da qualidade da superfície resultante entre aqueles e estes, com destaque aos resultados dos dois últimos testes, nos quais foi possível recuperar, com o emprego da metodologia, a forma apropriada de uma superfície originalmente amorfa e sem quaisquer indícios de fidelidade ao objeto expectado.

4.2 Testes com Dados Reais

A execução dos testes com dados sintéticos legitimou a competência do processo de reconstrução relatado e discutido em contornar singularidades específicas que, porventura, se revelem presentes na amostragem das fatias originais do problema. O conjunto de testes com dados reais teve como alvo verificar se essa aptidão comprovada se estendia para casos concretos, os quais, geralmente, possuem um numeroso conjunto de fatias originais dimensionalmente maiores e com contornos topologicamente mais complexos, a partir das quais se recupera um objeto tridimensional real, como, por exemplo, estruturas vegetais ou animais, em especial partes do corpo humano.

Para os testes com dados reais trabalhados, foram selecionados três conjuntos de fatias distintos, cujas superfícies reconstruídas representaram ossos do sistema esquelético humano. O primeiro caso reconstituiu um fragmento da mandíbula, enquanto os outros dois retornaram representações completas da clavícula e do ilíaco. As métricas de cada um dos testes são detalhadas na Tabela 4.6.

	Conjuntos de fatias		
Métricas	Fragmento	Clavícula	Ilíaco
Número de fatias	26	141	253
Dimensões (altura×largura) das fatias (pixels)	150×100	110×110	144×128
Distância vertical entre as fatias	uniforme: 1	uniforme: 1	uniforme: 0.5
Número de pixels por fatia			
Mínimo	363	21	111
Máximo	822	168	819
Média no conjunto	721.50	115.52	567.96

Tabela 4.6: Métricas dos conjuntos de fatias reais.

Relativamente ao caso de teste Fragmento, a Figura 4.9 expõe amostras das fatias do

conjunto original e a Figura 4.10, as superfícies reconstruídas. Uma apreciação destas ratifica a hipótese sugerida sobre a presumida competência de ambos os métodos *Level Set* e *Particle Level Set* na produção das informações faltantes para o preenchimento das regiões vazias e, consequentemente, para uma reconstrução mais completa e precisa dos objetos, ante a implementada sobre as fatias originais unicamente.

De modo a ilustrar com maior eloquência a capacidade afirmada de a metodologia proposta gerar os dados complementares necessários, para os testes Clavícula e Ilíaco, cujas respectivas amostras das fatias originais encontram-se nas Figuras 4.11 e 4.13, as Figuras 4.12 e 4.14 exibem as correspondentes representações ósseas reconstruídas por meio do empilhamento do conjunto completo de fatias, entre originais e intermediárias geradas, ao invés da renderização das superfícies.

Por fim, as Tabelas 4.7, 4.8 e 4.9 apresentam as estatísticas quantitativa e qualitativa do procedimento de geração das fatias intermediárias minuciadas para os três testes Fragmento, Clavícula e Ilíaco, respectivamente.

	Método	
Estatísticas	*Level Set*	*Particle Level Set*
Total de fatias intermediárias geradas	444	448
Tempo médio (s) de geração de cada fatia intermediária	2.22	43.48

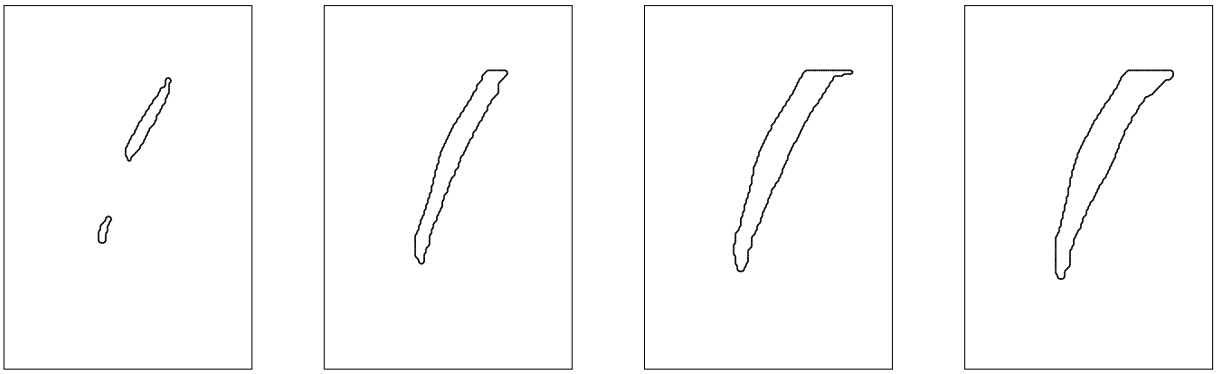
Tabela 4.7: Estatísticas dos testes – Fragmento.

	Método	
Estatísticas	*Level Set*	*Particle Level Set*
Total de fatias intermediárias geradas	1544	1578
Tempo médio (s) de geração de cada fatia intermediária	1.56	20.48

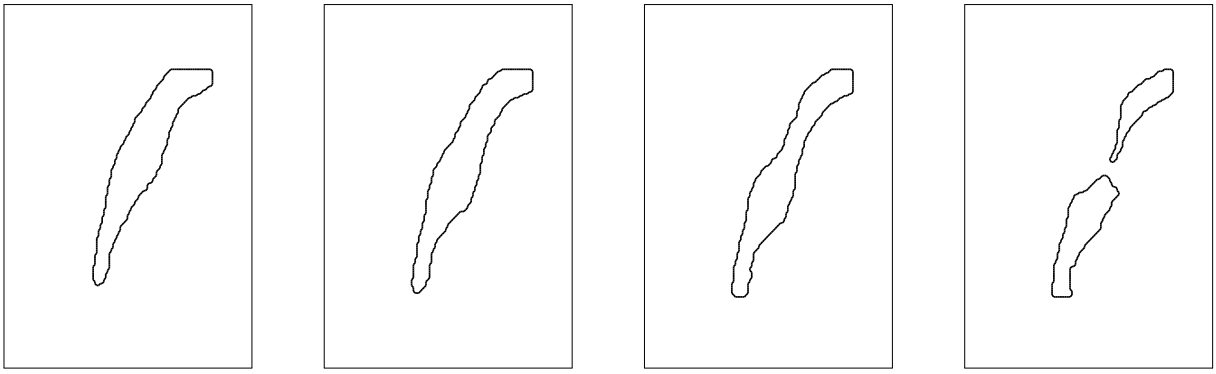
Tabela 4.8: Estatísticas dos testes – Clavícula.

	Método	
Estatísticas	*Level Set*	*Particle Level Set*
Total de fatias intermediárias geradas	2448	2460
Tempo médio (s) de geração de cada fatia intermediária	2.35	33.84

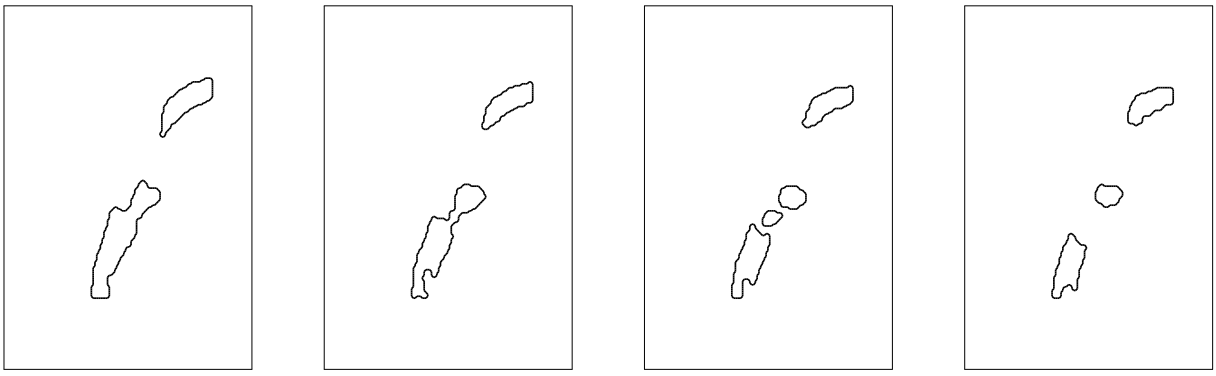
Tabela 4.9: Estatísticas dos testes – Ilíaco.



(a) i = 1; (b) i = 3; (c) i = 5; (d) i = 7;

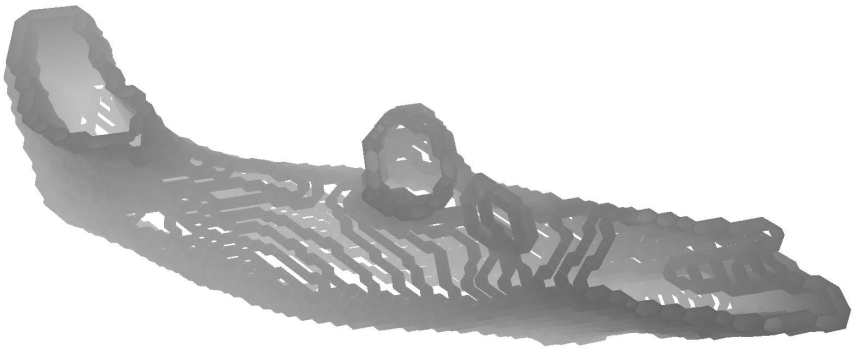


(e) i = 9; (f) i = 12; (g) i = 15; (h) i = 18;

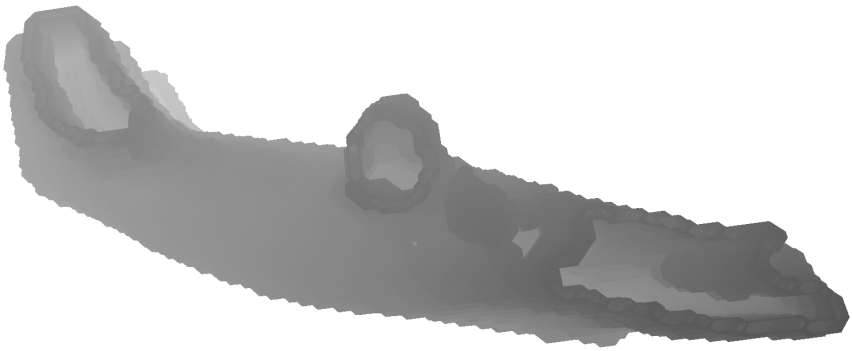


(i) i = 20; (j) i = 22; (k) i = 24; (l) i = 26.

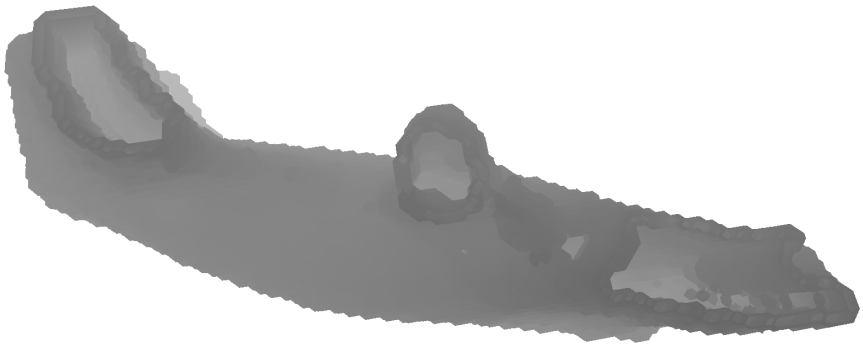
Figura 4.9: Amostra das fatias originais – Fragmento.



(a) Somente fatias originais;

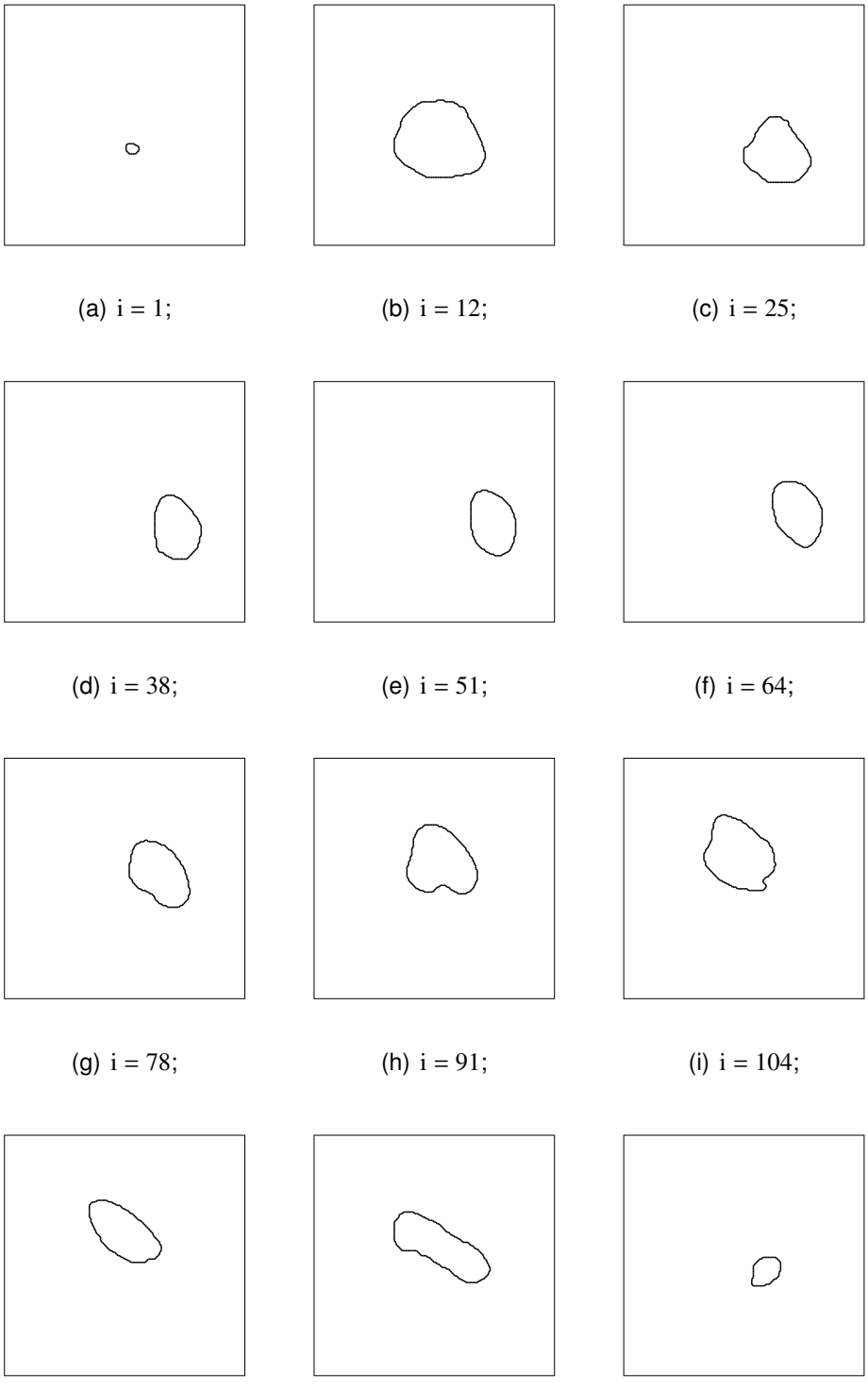


(b) Metamorfose via *Level Set*;



(c) Metamorfose via *Particle Level Set*.

Figura 4.10: Superfícies reconstruídas – Fragmento.



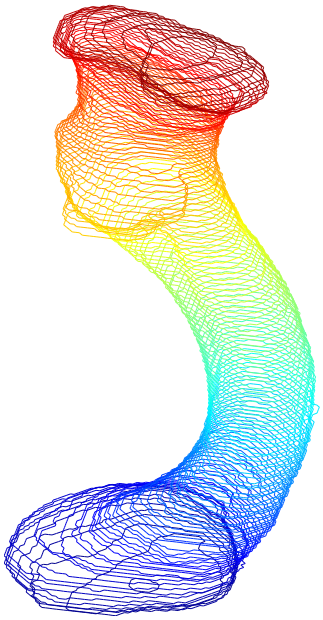
(a) i = 1; (b) i = 12; (c) i = 25;

(d) i = 38; (e) i = 51; (f) i = 64;

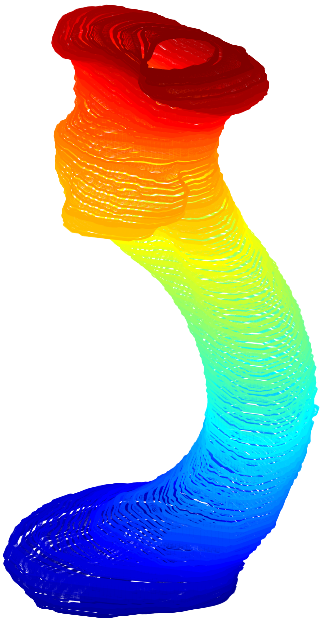
(g) i = 78; (h) i = 91; (i) i = 104;

(j) i = 117; (k) i = 130; (l) i = 141.

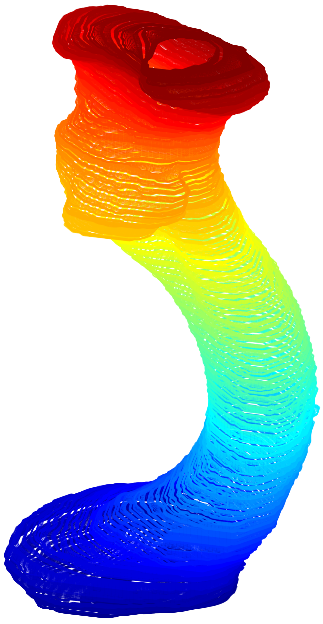
Figura 4.11: Amostra das fatias originais – Clavícula.



(a) Somente fatias originais;

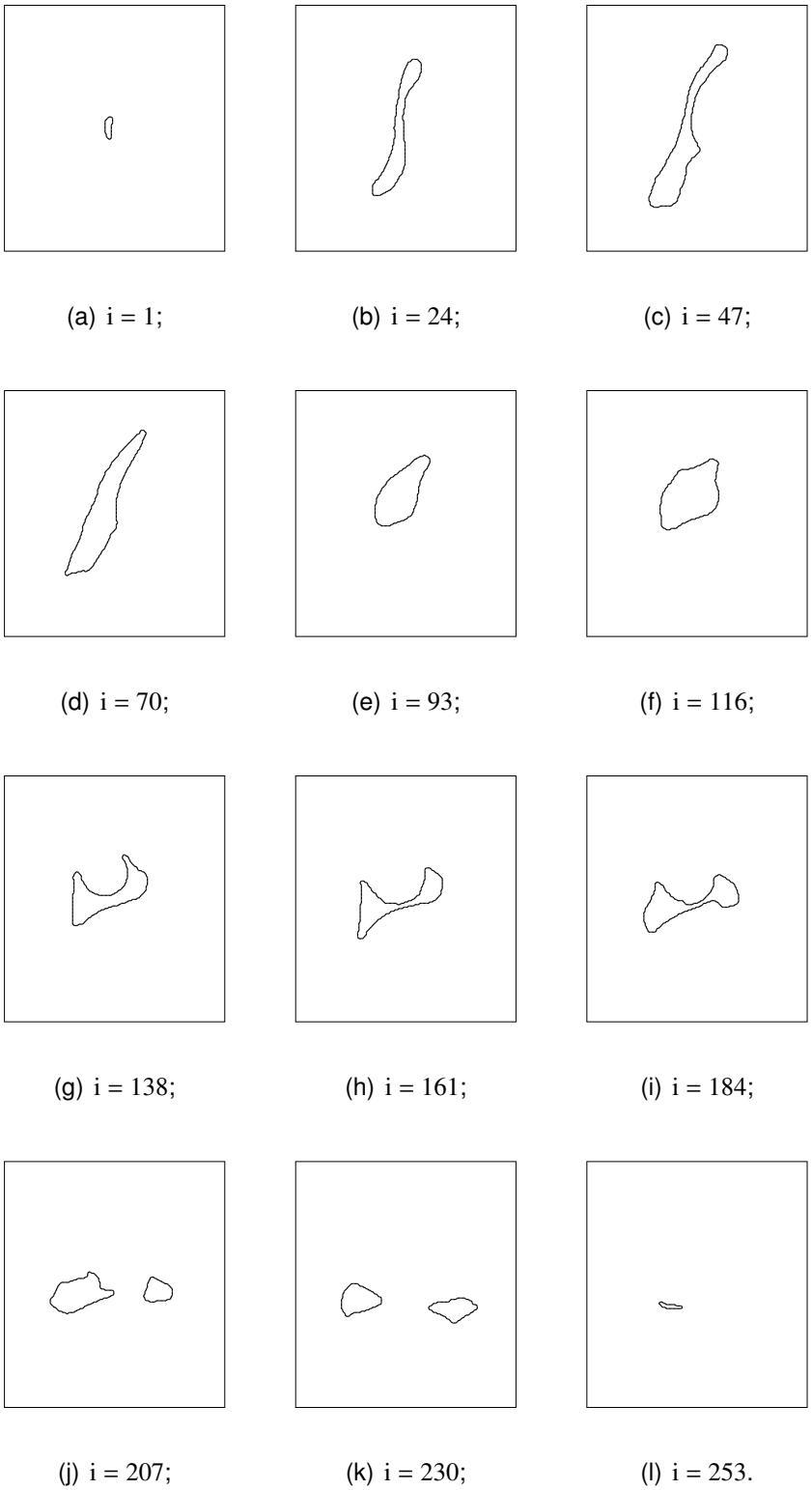


(b) Metamorfose via *Level Set*;



(c) Metamorfose via *Particle Level Set*.

Figura 4.12: Fatias empilhadas – Clavícula.



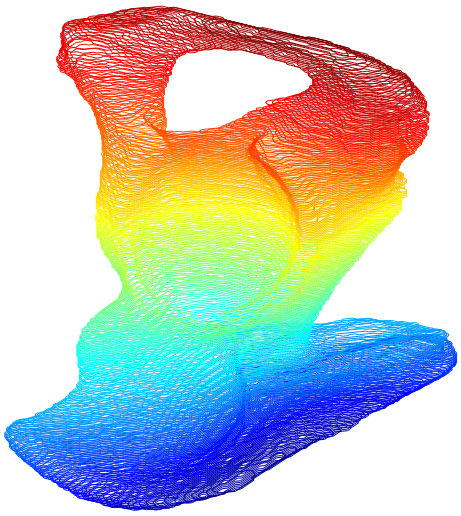
(a) i = 1; (b) i = 24; (c) i = 47;

(d) i = 70; (e) i = 93; (f) i = 116;

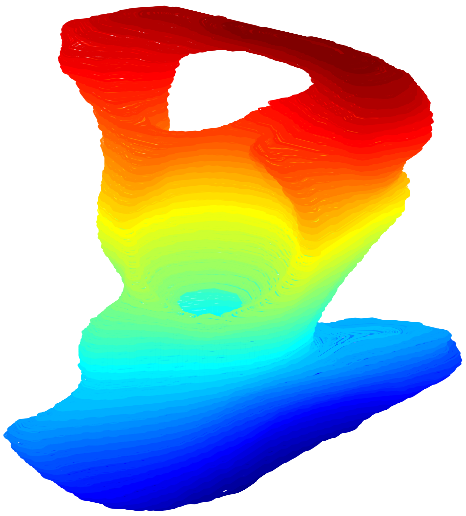
(g) i = 138; (h) i = 161; (i) i = 184;

(j) i = 207; (k) i = 230; (l) i = 253.

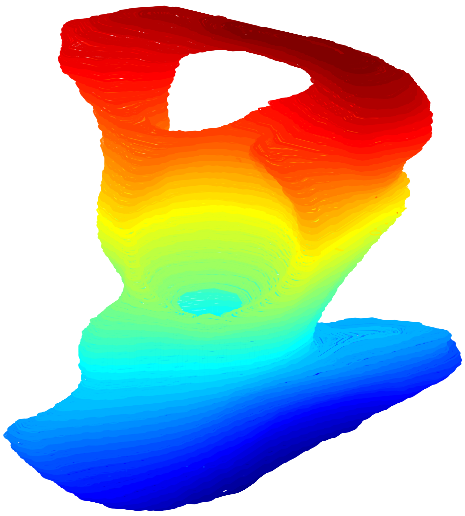
Figura 4.13: Amostra das fatias originais – Ilíaco.



(a) Somente fatias originais;



(b) Metamorfose via *Level Set*;



(c) Metamorfose via *Particle Level Set*.

Figura 4.14: Fatias empilhadas – Ilíaco.

Capítulo 5

Conclusões e Trabalhos Futuros

Ao longo das últimas décadas, o *Level Set* revelou-se um dos mais versáteis métodos para a resolução de problemas científico-computacionais, com relevantes benefícios ao campo da Ciência da Computação e às áreas de Processamento de Imagens e de Visão Computacional. Porém, uma carência de estudos acerca de um eventual emprego do método a questões envolvendo reconstruções de superfícies tridimensionais, mais especificamente aquelas originadas a partir de fatias planas paralelas, foi notada. Vislumbrou-se nessa lacuna, assim, uma oportunidade de pesquisa sobre a aplicação do método *Level Set* ao problema.

Os trabalhos até então publicados, relacionando o *Level Set* à reconstrução de superfícies a partir de fatias planas, utilizavam o método para a metamorfose planar entre cada par de fatias consecutivas do conjunto de fatias originais de entrada. Esse procedimento mostrava-se eficiente para a geração dos dados necessários faltantes para a completude do processo de reconstrução, em casos nos quais a amostragem das fatias manifestava-se falha sob determinado aspecto. Esses mesmos estudos, porém, não discutiam, de forma explícita, a questão da suavização excessiva dos contornos presentes nas fatias em evolução sob a ação do *Level Set*, cuja ocorrência na resolução do problema poderia acarretar perda de informações.

Diante dessa conjuntura e com a ciência da capacidade teórica de a variante *Particle Level Set* contornar essa adversidade, propôs-se um estudo a respeito da aplicação da metamorfose planar, por meio de ambos os métodos *Level Set* e *Particle Level Set*, à reconstrução de superfícies tridimensionais a partir de fatias planas paralelas, de modo que uma análise qualitativa sobre a viabilidade do método *Particle Level Set* para esse objetivo fosse discutida.

Para tanto, uma narrativa teórica dos métodos foi inicialmente exposta, sendo explanados os pontos essenciais da modelagem matemática do *Level Set* e a adaptação da metamorfose planar para o contexto numérico do método. Na sequência, minuciou-se

a metodologia empregada para a execução computacional do processo de reconstrução, com relevância destacada à geração das fatias intermediárias – considerada o núcleo do processo como um todo, por ser nessa etapa em que a metamorfose é efetivamente aplicada – e aos passos inerentes ao *Particle Level Set*. Para a verificação da corretude do código computacional escrito embasado na metodologia detalhada, testes, então, foram caracterizados e realizados e os resultados e estatísticas experimentais consequentes, apresentados.

Com base nesses resultados experimentais, pôde-se reforçar a habilidade, previamente divulgada em estudo anterior [45], do método *Level Set* para a reconstrução de superfícies a partir de fatias planas, habilidade essa igualmente compartilhada pelo método alternativo *Particle Level Set*. Nesse âmbito, é possível expandir o potencial do *Particle Level Set* por meio de aperfeiçoamentos teóricos e computacionais na metodologia descrita, em trabalhos futuros.

Conforme ilustrado na Figura 4.4, ambos os métodos imprimiram à metamorfose uma velocidade de deformação não uniforme, alta nas iterações iniciais e gradativamente pequena à medida em que se aproximou o procedimento da convergência estipulada – em outras palavras, os contornos foram intensamente deformados no início da metamorfose via *(Particle) Level Set* e praticamente inalterados ao final, o que originou uma pilha de fatias intermediárias com um notável grau de semelhança entre si, à conclusão da metamorfose, e, por consequência, uma superfície reconstruída possivelmente divergente da esperada teoricamente.

Como uma solução para essa questão, Nilsson et al. [45] propõem um esquema que, em sua essência, processa a totalidade dos pares de fatias originais de maneira concomitante, isto é, as metamorfoses inerentes a todos os pares são executadas ao mesmo tempo. Adicionalmente, o esquema, baseado no cálculo da distância que cada região do contorno metamorfoseante percorre até a convergência e na determinação do correspondente tempo de chegada, reformula a velocidade F em uma função contínua e uniforme, de modo que todas as regiões do contorno, sob a ação dessa velocidade remodelada, convirjam simultaneamente.

No tocante à regulação da suavização, imagens ilustrando a evolução da metamorfose via *Particle Level Set* (Figura 4.3) comprovaram a capacidade de a variante controlar, mesmo que sutilmente, a suavização excessiva inerente ao *Level Set*. Porém, para a maioria dos objetos retornados durante os testes, esse apuro nos detalhes tornou-se visualmente imperceptível no contexto tridimensional; para outros, no entanto, o emprego do *Particle Level Set* possibilitou refinamentos relevantes na superfície reconstruída, como observado nos casos Castiçal (Figura 4.7) e Vaso (Figura 4.8).

Para potencializar os resultados retornados pelo *Particle Level Set* nesse contexto, aprimoramentos na resolução das fatias intermediárias geradas e da superfície reconstruída

são, da mesma forma, praticáveis. Para tanto, pode-se ajustar a quantidade de partículas sedimentadas, por célula da grade, de acordo com o nível de resolução almejado, sendo essas duas variáveis diretamente proporcionais. Contudo, ponderação mostra-se necessária na definição da primeira, pois um aumento desmesurado em seu número afeta negativa e profundamente a duração do processo. É preponderante, logo, um equilíbrio entre os valores de ambas as variáveis para a viabilidade qualitativa e computacional da reconstrução via *Particle Level Set*.

A duração do processo utilizando o *Particle Level Set* evidenciou-se outra questão relevante. Com base na análise dos resultados, a reconstrução implementada por meio da variante apresenta um elevado tempo de execução, grandemente dependente das características das fatias originais, como, por exemplo, quantidade de pixels e topologia dos contornos nelas presentes.

Uma possibilidade para a aceleração do processo reside no método *Narrow Band Level Set* [1, 63], uma adaptação mais eficiente do original, em termos de tempo de execução, pelo qual apenas os pontos da grade pertencentes ao interior de uma banda estreita de largura k, posicionada ao redor da interface dinâmica, são atualizados. Para fronteiras em duas dimensões, como os contornos presentes nas fatias planas originais, o total de cálculos necessários para uma iteração da evolução *Level Set* decai de $O(N^2)$ (no caso em que atualizam-se todos os pontos da matriz $N \times N$ da fatia) para $O(kN)$.

O grupo de ferramentas de computação em paralelo disponibilizado pelo MATLAB oferece recursos adicionais além da função `parfor` citada (Seção 3.5), como a capacidade de expansão do paralelismo para uma rede de computadores, inclusive com o uso da computação em nuvem, ou o aproveitamento da potência da unidade de processamento gráfico (GPU) em computadores habilitados com plataformas NVIDIA CUDA [48], sem a necessidade da construção de arquiteturas ou códigos baixo-nível envolvendo GPU, entre outras técnicas. A aplicação desses recursos, isolados ou em conjunto, configuram uma segunda abordagem para o aumento do desempenho do processo de reconstrução proposto.

Fundamentando-se, logo, nos resultados apresentados no Capítulo 4 e na presente discussão analítica, conclui-se sobre a viabilidade tanto do método *Level Set* quanto da variante *Particle Level Set* para a reconstrução de superfícies a partir de fatias planas paralelas: enquanto o primeiro destaca-se eficiente para casos generalizados, recomenda-se o emprego do segundo a situações em que a preservação de características finas dos contornos e da superfície a ser reconstruída é imprescindível e impera sobre o tempo de execução do processo, para, por exemplo, a avaliação mais precisa de informações quantitativas do objeto em estudo, tais como a área lateral e o volume.

Apêndice A

Coordenadas de uma Imagem

Por convenção, a origem de uma imagem encontra-se em seu canto superior esquerdo, de modo que o eixo x representa a largura da imagem e o eixo y, a altura (Figura A.1) [56]. Consequentemente, uma vez armazenada em uma matriz, o eixo x da imagem passa a equivaler às colunas da matriz e o eixo y, às linhas. Exemplificando, o pixel destacado na Figura A.1 possui coordenadas cartesianas $(x, y) = (9, 14)$ e coordenadas matriciais $(i, j) = (14, 9)$. Como toda a resolução numérica do processo de reconstrução apresentado baseia-se em operações sobre matrizes, optou-se por padronizar ambas as representações pixelares, efetuando-se uma inversão na representação cartesiana, de $(x, y) = (9, 14)$ para $(y, x) = (14, 9)$, com o intuito de evitar erros de interpretação e falta de clareza durante a leitura do texto.

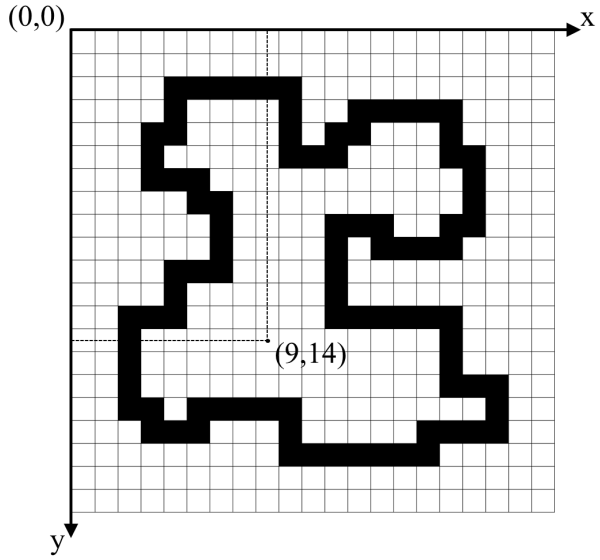


Figura A.1: Convenção dos eixos coordenados para a representação de imagens.

Bibliografia

[1] D. Adalsteinsson and J. A. Sethian. A Fast Level Set Method for Propagating Interfaces. *Journal of Computational Physics*, 118:269–277, 1995.

[2] F. Al-Qunaieer, H. Tizhoosh, and S. Rahnamayan. Multi-Resolution Level Set Image Segmentation Using Wavelets. In *Proceedings of the 18th IEEE International Conference on Image Processing*, pages 269–272, 2011.

[3] M. Aslan, E. Mostafa, H. Abdelmunim, A. Shalaby, A. Farag, and B. Arnold. A Novel Probabilistic Simultaneous Segmentation and Registration Using Level Set. In *Proceedings of the 18th IEEE International Conference on Image Processing*, pages 2161–2164, 2011.

[4] M. Bertalmio, G. Sapiro, and G. Randall. Region Tracking on Surfaces Deforming via Level-Sets Methods. In *Proceedings of the 2nd International Conference on Scale-Space Theories in Computer Vision*, pages 330–338, 1999.

[5] D. Breen, S. Mauch, R. Whitaker, and J. Mao. 3D Metamorphosis Between Different Types of Geometric Models. *Computer Graphics Forum*, 20:36–48, 2001.

[6] D. Breen and R. Whitaker. A Level-Set Approach for the Metamorphosis of Solid Models. *IEEE Transactions on Visualization and Computer Graphics*, 7:173–192, 2001.

[7] M. Burger, B. Hackl, and W. Ring. Incorporating Topological Derivatives into Level Set Methods. *Journal of Computational Physics*, 194:344–362, 2004.

[8] H. Ceniceros, A. Roma, A. Silveira-Neto, and M. Villar. A Robust, Fully Adaptive Hybrid Level-Set/Front-Tracking Method for Two-Phase Flows with an Accurate Surface Tension Computation. *Communications in Computational Physics*, 8:51–94, 2010.

[9] Centro de Tecnologia da Informação Renato Archer. Centro de Tecnologia da Informação Renato Archer. `http://www.cti.gov.br/`, 2012.

[10] V. Challis. A Discrete Level-Set Topology Optimization Code Written in Matlab. *Structural and Multidisciplinary Optimization*, 41:453–464, 2010.

[11] S. K. Cheedela, B. Stevens, H. Schmidt, and J. P. Mellado. Sensitivity of the ECHAM6 Single Column Model to Vertical Resolution and Implementation of the Level Set Method. In *EGU General Assembly Conference Abstracts*, page 13511. 2010.

[12] L. T. Cheng, J. Dzubiella, J. A. McCammon, and B. Li. Application of the Level-Set Method to the Implicit Solvation of Nonpolar Molecules. *The Journal of Chemical Physics*, 127:084503+, 2007.

[13] L. T. Cheng, B. Li, and Z. Wang. Level-Set Minimization of Potential Controlled Hadwiger Valuations for Molecular Solvation. *Journal of Computational Physics*, 229:8497–8510, 2010.

[14] L. T. Cheng, Y. Xie, J. Dzubiella, J. A. McCammon, J. Che, and B. Li. Coupling the Level-Set Method with Molecular Mechanics for Variational Implicit Solvation of Nonpolar Molecules. *Journal of Chemical Theory and Computation*, 5:257–266, 2009.

[15] R. Courant, K. Friedrichs, and H. Lewy. On the Partial Difference Equations of Mathematical Physics. *Journal of Research and Development*, 11:215–234, 1967.

[16] E. Dietze, B. Stevens, J. P. Mellado, N. Peters, and H. Schmidt. Level Set Methods for Meteorological Multi Scale Problems. In *EGU General Assembly Conference Abstracts*, page 10083. 2010.

[17] M. El-Melegy, N. Al-Ashwal, and A. Farag. Model-Based Multiview Stereo via Level Sets with Statistical Shape Prior. In *Proceedings of the 18th IEEE International Conference on Image Processing*, pages 1013–1016, 2011.

[18] D. Enright. *Use of the Particle Level Set Method for Enhanced Resolution of Free Surface Flows*. PhD thesis, Stanford University, Stanford, California, EUA, 2002.

[19] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*, 183:83–116, 2002.

[20] D. Enright, F. Losasso, and R. Fedkiw. A Fast and Accurate Semi-Lagrangian Particle Level Set Method. *Computers and Structures*, 83:479–490, 2005.

[21] D. Enright, S. Marschner, and R. Fedkiw. Animation and Rendering of Complex Water Surfaces. *ACM Transactions on Graphics*, 21:736–744, 2002.

[22] R. Fahmi and A. Farag. A Fast Level Set Algorithm for Shape-Based Segmentation with Multiple Selective Priors. In *Proceedings of the 15th IEEE International Conference on Image Processing*, pages 1073–1076, 2008.

[23] O. Faugeras and R. Keriven. Complete Dense Stereovision Using Level Set Methods. In *Proceedings of the 5th European Conference on Computer Vision*, pages 379–393, 1998.

[24] S. Gottlieb and C. W. Shu. Total Variation Diminishing Runge-Kutta Schemes. *Mathematics of Computation*, 67:73–85, 1998.

[25] H. L. Guidorizzi. *Um Curso de Cálculo – Volume 2*. LTC, 2001.

[26] B. Hillebrand, T. Geenen, W. Spakman, and A. P. van den Berg. Using the Level Set Method in Slab Detachment Modeling. In *EGU General Assembly Conference Abstracts*, page 3525. 2012.

[27] J. Hyman. Adaptive Moving Mesh Methods for Partial Differential Equations. *Advances in Reactor Computations*, pages 24–43, 1983.

[28] J. Hyman. Numerical Methods for Tracking Interfaces. *Physica D: Nonlinear Phenomena*, 12:396–407, 1984.

[29] M. Iglesias and D. McLaughlin. Level-Set Techniques for Facies Identification in Reservoir Modeling. *IOP Publishing Inverse Problems*, 27:035008, 2011.

[30] Intel Corporation. Processador Intel Core i7. `http://www.intel.com.br/`, 2012.

[31] G. S. Jiang and D. Peng. Weighted ENO Schemes for Hamilton-Jacobi Equations. *SIAM Journal on Scientific Computing*, 21:2126–2143, 1997.

[32] Kitware, Inc. VTK – The Visualization Toolkit. `http://www.vtk.org/`, 2012.

[33] Y. Kwak, C. C. J. Kuo, and A. Nakano. Hybrid Lattice-Boltzmann/Level-Set Method for Liquid Simulation and Visualization. *International Journal of Computational Science*, 3:579–592, 2009.

[34] C. Li, C. Xu, C. Gui, and M. Fox. Distance Regularized Level Set Evolution and Its Application to Image Segmentation. *IEEE Transactions on Image Processing*, 19:3243–3254, 2010.

[35] W. Li, X. Zhang, J. Gao, W. Hu, H. Ling, and X. Zhou. Discriminative Level Set for Contour Tracking. In *Proceedings of the 20th International Conference on Pattern Recognition*, pages 1735–1738, 2010.

[36] J. Lie, M. Lysaker, and X. C. Tai. A Binary Level Set Model and Some Applications to Mumford-Shah Image Segmentation. *IEEE Transactions on Image Processing*, 15:1171–1181, 2006.

[37] D. S. Lindsey and T. K. Dupont. Using the Level Set Method to Track Ice Sheet Boundaries. In *AGU Fall Meeting Abstracts*, page C454. 2009.

[38] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 163–169, 1987.

[39] D. A. Machado, G. A. Giraldi, and A. A. Novotny. Segmentation Approach Based on Topological Derivative and Level Set. In *Proceedings of the 17th International Conference on Systems, Signals and Image Processing*, pages 85–88, 2010.

[40] V. Mallet, D. Keyes, and F. Fendell. Modeling Wildland Fire Propagation with Level Set Methods. *Computers and Mathematics with Applications*, 57:1089–1101, 2009.

[41] J. Marot, Y. Caulier, A. Kuleschow, K. Spinnler, and S. Bourennane. Contour Detection for Industrial Image Processing by Means of Level Set Methods. In *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 655–663, 2008.

[42] Microsoft. Windows 7 – Microsoft Windows. `http://windows.microsoft.com/pt-BR/windows7/products/home`, 2012.

[43] E. Mokberi and P. Faloutsos. A Particle Level Set Library. Technical report, UCLA Computer Graphics & Vision Laboratory, 1999.

[44] D. Mumford and J. Shah. Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems. *Communications on Pure and Applied Mathematics*, 42:577–685, 1989.

[45] O. Nilsson, D. Breen, and K. Museth. Surface Reconstruction via Contour Metamorphosis: An Eulerian Approach with Lagrangian Particle Tracking. In *Proceedings of the 16th IEEE Visualization*, pages 407–414, 2005.

[46] W. F. Noh and P. Woodward. SLIC (Simple Line Interface Calculation). In *Proceedings of the 5th International Conference on Numerical Methods in Fluid Dynamics*, pages 330–340, 1976.

[47] NVIDIA Corporation. GeForce GT 335M. `http://www.nvidia.com.br/object/product_geforce_gt_335m_br.html`, 2012.

[48] NVIDIA Corporation. Plataforma de Computação Paralela CUDA. `http://www.nvidia.com.br/object/cuda_home_new_br.html`, 2012.

[49] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[50] S. Osher and J. A. Sethian. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[51] S. Osher and C. W. Shu. High-Order Essentially Nonoscillatory Schemes for Hamilton-Jacobi Equations. *SIAM Journal on Numerical Analysis*, 28:907–922, 1991.

[52] G. Pacquaut, J. Bruchon, N. Moulin, and S. Drapier. Combining a Level-Set Method and a Mixed Stabilized P1/P1 Formulation for Coupling Stokes-Darcy Flows. *International Journal for Numerical Methods in Fluids*, 69:459–480, 2012.

[53] C. Park, C. Min, and M. Kang. Surface Reconstruction from Scattered Point Data on Octree. *Journal of the Korean Society for Industrial and Applied Mathematics*, 16:31–49, 2012.

[54] W. K. Park and D. Lesselier. Reconstruction of Thin Electromagnetic Inclusions by a Level-Set Method. *IOP Publishing Inverse Problems*, 25:085010, 2009.

[55] H. Pedrini. Reconstrução 3D a partir de Seções Transversais de Objetos. Master's thesis, Universidade Estadual de Campinas, Campinas, São Paulo, Brasil, 1994.

[56] H. Pedrini and W. R. Schwartz. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*. Thomson Learning, 2007.

[57] A. Pralong and M. Funk. A Level-Set Method for Modeling the Evolution of Glacier Geometry. *Journal of Glaciology*, 50:485–491, 2004.

[58] Python Software Foundation. Python Programming Language. `http://www.python.org/`, 2012.

[59] R. Rehm and R. McDermott. Fire-Front Propagation Using the Level Set Method. Technical report, National Institute of Standards and Technology, 2009.

[60] T. Riklin-Raviv, N. Sochen, and N. Kiryati. Mutual Segmentation with Level Sets. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, page 177, 2006.

[61] A. D. Santis and D. Iacoviello. A Discrete Level Set Approach to Image Segmentation. *Signal, Image and Video Processing*, 1:303–320, 2007.

[62] J. A. Sethian. Curvature and the Evolution of Fronts. *Communications in Mathematical Physics*, 101:487–499, 1985.

[63] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1996.

[64] X. Sun, H. Yao, and S. Zhang. Contour Tracking via On-Line Discriminative Appearance Modeling Based Level Sets. In *Proceedings of the 18th IEEE International Conference on Image Processing*, pages 2317–2320, 2011.

[65] The MathWorks, Inc. MATLAB. `http://www.mathworks.com/products/matlab/`, 2012.

[66] L. Ville, L. Silva, and T. Coupez. Convected Level Set Method for the Numerical Simulation of Fluid Buckling. *International Journal for Numerical Methods in Fluids*, 66:324–344, 2011.

[67] S. Y. Wang, K. M. Lim, B. C. Khoo, and M. Y. Wang. An Extended Level Set Method for Shape and Topology Optimization. *Journal of Computational Physics*, 221:395–421, 2007.

[68] Y. Wang, L. Wang, S. Xiang, and C. Pan. Level Set Evolution with Locally Linear Classification for Image Segmentation. In *Proceedings of the 18th IEEE International Conference on Image Processing*, pages 3361–3364, 2011.

[69] Z. Wang, J. Yang, and F. Stern. An Improved Particle Correction Procedure for the Particle Level Set Method. *Journal of Computational Physics*, 228:5819–5837, 2009.

[70] R. Whitaker. Reducing Aliasing Artifacts in Iso-Surfaces of Binary Volumes. In *Proceedings of the 2000 IEEE Symposium on Volume Visualization*, pages 23–32, 2000.

[71] T. Yamada, K. Izui, S. Nishiwaki, and A. Takezawa. A Topology Optimization Method Based on the Level Set Method Incorporating a Fictitious Interface Energy. *Computer Methods in Applied Mechanics and Engineering*, 199:2876–2891, 2010.

[72] J. Ye, I. Yanovsky, B. Dong, R. Gandlin, A. Brandt, and S. Osher. Multigrid Narrow Band Surface Reconstruction via Level Set Functions. Technical report, UCLA Computational and Applied Mathematics, 2009.

[73] S. Y. Yeo, X. Xie, I. Sazonov, and P. Nithiarasu. Level Set Segmentation with Robust Image Gradient Energy and Statistical Shape Prior. In *Proceedings of the 18th IEEE International Conference on Image Processing*, pages 3397–3400, 2011.

[74] W. Yu, H. K. Lee, S. Hariharan, W. Bu, and S. Ahmed. Level Set Segmentation of Cellular Images Based on Topological Dependence. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, pages 540–551, 2008.

[75] H. K. Zhao, S. Osher, and R. Fedkiw. Fast Surface Reconstruction Using the Level Set Method. In *Proceedings of the 2001 IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 194–201, 2001.

[76] H. K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and Nonparametric Shape Reconstruction from Unorganized Data Using a Variational Level Set Method. *Computer Vision and Image Understanding*, 80:295–314, 2000.

[77] J. Zhou, L. Yao, J. Yang, and C. Li. Moment Based Level Set Method for Image Segmentation. In *Proceedings of the 15th IEEE International Conference on Image Processing*, pages 1069–1072, 2008.