UNIVERSIDADE FEDERAL DO CEARÁ

DEPARTAMENTO DE COMPUTAÇÃO

MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

CRAb - COMPUTAÇÃO GRÁFICA, REALIDADE VIRTUAL E ANIMAÇÃO

TIAGO GUIMARÃES SOMBRA

GERAÇÃO ADAPTATIVA DE MALHAS DE SUPERFÍCIES PARAMÉTRICAS EM PARALELO COM CONTROLE DE CURVATURA

FORTALEZA

2016

TIAGO GUIMARÃES SOMBRA

GERAÇÃO ADAPTATIVA DE MALHAS DE SUPERFÍCIES PARAMÉTRICAS EM PARALELO COM CONTROLE DE CURVATURA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito parcial para a obtenção do Título de Mestre em Ciência da Computação. Área de concentração: Computação Gráfica.

Orientador: Prof. Dr. Joaquim Bento Cavalcante Neto

Coorientador: Prof. Dr. Creto Augusto Vidal

FORTALEZA

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

S676g Sombra, Tiago Guimarães.
Geração adaptativa de malhas de superfícies paramétricas em paralelo com controle de curvatura. / Tiago Guimarães Sombra. – 2016.
70 f. : il.; color.

Dissertação (Mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2016.
Área de Concentração: Computação Gráfica.
Orientação: Prof. Dr. Joaquim Bento Cavalcante Neto.
Coorientação: Prof. Dr. Creto Augusto Vidal.

1. Computação de alto desempenho. 2. Partições. 3. Algoritmos. I. Título.

CDD 005

TIAGO GUIMARÃES SOMBRA

Geração Adaptativa de Malhas de Superfícies Paramétricas em Paralelo com Controle de Curvatura

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito para a obtenção do grau de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Joaquim Bento Cavalcante Neto
(Presidente)
Universidade Federal do Ceará – UFC

Prof. Dr. Creto Augusto Vidal
(Coorientador)
Universidade Federal do Ceará – UFC

Prof. Dr. Evandro Parente Junior
Universidade Federal do Ceará – UFC

Fortaleza, 28 de março de 2016.

Este trabalho eu dedico a minha família,
principalmente a minha esposa
que, durante essa jornada, nunca deixou de me apoiar.

AGRADECIMENTOS

Agradeço em primeiro lugar a minha esposa Jannayna e ao meu filho João, pelos momentos compartilhados entre números e fraldas, que por muitas vezes souberam lidar com o meu distanciamento devido ao meu foco na pesquisa, de forma paciente e sincera.

A minha família, pelo apoio e incentivo incondicional para realização deste trabalho.

Agradeço aos meus professores orientadores, Bento, Creto e Emanuele, pelos ensinamentos, por terem acreditado no meu trabalho e por terem guiado minha jornada acadêmica. Ao membro da Banca Evandro Parente pelas indispensáveis contribuições que aprimoraram e engrandeceram esta pesquisa.

A Cynthia Maria pela sua atenção e contribuições nas correções deste trabalho. Aos colegas de laboratório (Lab 3 e Lab 4), que prefiro não citar nomes para não esquecer nenhum, pelos momentos de ensinamento, descontrações e ajuda durante esta jornada.

Agradeço ao programa de Mestrado e Doutorado em Ciência da Computação (MDCC) da Universidade Federal do Ceará (UFC), ao grupo de Pesquisa em Computação Gráfica, Realidade Virtual e Animação (CRAb), pela oportunidade dada. Aos funcionários pertencentes a essa instituição, que sempre se mostraram disponíveis para ajudar no que fosse preciso.

Finalmente agradeço, ao Centro Nacional de Processamento de Alto Desempenho, instalado na Universidade Federal do Ceará (CENAPAD-UFC), pela disponibilização do ambiente para a realização dos experimentos deste trabalho. À Coodernação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro para realização desta pesquisa.

"O estudo é a essência da sabedoria."
(Salomão)

RESUMO

Este trabalho descreve uma técnica para gerar malhas de superfícies paramétricas utilizando computação paralela, com processadores de memória compartilhada. A entrada para o algoritmo é um conjunto de *patches* paramétricos que modela a superfície de um determinado objeto. Uma estrutura de partição espacial é proposta para decompor o domínio em tantos subdomínios quantos forem os processos no sistema paralelo. Cada subdomínio é formado por um conjunto de *patches* e a divisão de sua carga é guiada seguindo uma estimativa de carga. Esta decomposição tenta equilibrar a quantidade de trabalho em todos os subdomínios. A quantidade de trabalho, conhecida como carga, de qualquer gerador de malha é geralmente dada em função do tamanho da saída do algoritmo, ou seja, do tamanho da malha gerada. Assim, faz-se necessária uma técnica para estimar previamente o tamanho dessa malha, que é a carga total do domínio. Este trabalho utiliza-se de um cálculo de curvatura analítica média para cada *patch*, que por sua vez, é dado de entrada para estimar esta carga e a decomposição é feita a partir dessa curvatura analítica média. Uma vez decomposto o domínio, cada processo gera a malha em seu subdomínio ou conjunto de *patches* pela técnica de *quadtree* para regiões internas, avanço de fronteira para regiões de fronteira e por fim é aplicado um melhoramento na malha gerada. Esta técnica apresentou bons resultados de *speed-up*, mantendo a qualidade da malha comparável à qualidade da malha gerada de forma sequencial.

Palavras-chave: Geração de malhas de superfícies em paralelo. Computação de alto desempenho. Decomposição de domínios.

ABSTRACT

This work describes a technique for generating parametric surfaces meshes using parallel computing, with distributed memory processors. The input for the algorithm is a set of parametric patches that model the surface of a given object. A structure for spatial partitioning is proposed to decompose the domain in as many subdomains as processes in the parallel system. Each subdomain consists of a set of patches and the division of its load is guided following an estimate. This decomposition attempts to balance the amount of work in all the subdomains. The amount of work, known as load, of any mesh generator is usually given as a function of its output size, i.e., the size of the generated mesh. Therefore, a technique to estimate the size of this mesh, the total load of the domain, is needed beforehand. This work makes use of an analytical average curvature calculated for each patch, which in turn is input data to estimate this load and the decomposition is made from this analytical mean curvature. Once the domain is decomposed, each process generates the mesh on that subdomain or set of patches by a quadtree technique for inner regions, advancing front technique for border regions and is finally applied an improvement to mesh generated. This technique presented good speed-up results, keeping the quality of the mesh comparable to the quality of the serially generated mesh.

Keywords: Parallel surface mesh generation. High performance computing. Domain decomposition.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

3D	Espaço tridimensional
CAD	*Computer aided design*
CG	Computação gráfica
CPU	*Central processing unit*
EF	Elementos finitos
FEM	*Finite element methods*
GIS	*Geographical information systems*
HPC	*High performance computing*
MEF	Método dos elementos finitos
SIG	Sistemas de informações geográficas

SUMÁRIO

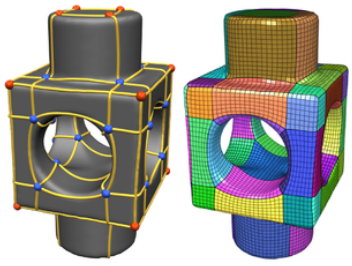
1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Malhas estão presentes nas mais diversas áreas da Ciência, como por exemplo Engenharia, Computação e Geologia, sendo numerosas as aplicações que as utilizam. Elas podem ser empregadas para representar geometrias e informações de modelos e superfícies nas mais variadas áreas (Hjelle e Dæhlen (2006)). Podem ser utilizadas em: **Computação Gráfica (CG) e Visualização**, na modelagem de objetos, avatares e ambientes tridimensionais para aplicações de realidade virtal (Figura 1.1a); em **Sistemas de informações geográficas** (SIG, ou GIS, do inglês *geographical information systems*), representando terrenos e relações entre objetos geográficos (Figura 1.1b); em **Projeto assistidos por computadores** (CAD, *computer aided design*), para auxiliar a construção e modelagem de objetos industriais (Figura 1.1c); na **Engenharia**, ajudando na análise de fenômenos físicos através de métodos numéricos, especialmente o Método dos Elementos Finitos (MEF, *finite element methods*, FEM) (Figura 1.1d).

Figura 1.1 – Áreas em que malhas são utilizadas.

(a) CG

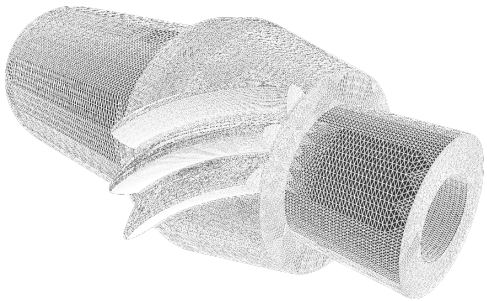


Fonte: Campen *et al.* (2012).

(b) SIG

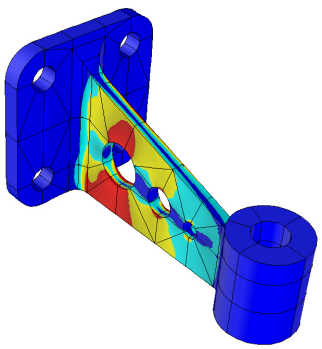
Fonte: <http://goo.gl/8i2qFH>.

(c) CAD



Fonte: Freitas *et al.* (2016).

(d) MEF



Fonte: <http://goo.gl/g8nG48>.

Os elementos que constituem uma malha geralmente são triângulos (tetraedros) ou quadriláteros (hexaedros); esta escolha facilita, por exemplo, a aplicação de processos como renderização e análise pelo MEF.

Além disso, um aspecto importante quando o assunto é malha, trata-se de sua geração, que pode ser classificada de duas formas: geração manual, que em muitos casos é um processo demorado e exige uma certa capacidade artística de quem estiver modelando, e geração automática, que é um processo que demanda uso do computador para gerá-la.

A pesquisa sobre geração automática de malhas vem crescendo expressivamente nas últimas décadas, inicialmente desenvolvendo-se, em sua grande maioria, algoritmos com base na triangulação de Delaunay e Avanço de Fronteira (Owen (1998)), bem como em outras técnicas que geram malhas de *quadtree/octree*, por varredura, por extrusão, por bisseção de arestas, por aplicação de padrões, de Voronoi, dentre outras.

Existem dois aspectos importantes a serem observados e destacados na geração automática de malhas:

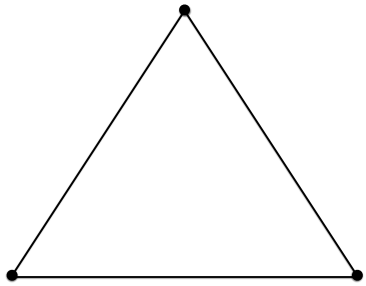
- **O tempo para geração da malha**, que está diretamente ligado ao tamanho da discretização utilizada e ao tamanho da malha, ou seja, quanto maior a discretização e a malha, maior será o tempo de execução.
- **A qualidade dos elementos que formam a malha gerada**, que será influenciada pela quantidade, distribuição e forma dos elementos gerados, que é importante para trabalhos que lidam com análise de elementos finitos e para os que usam resultados visuais de modelos renderizados.

Desenvolver programas, técnicas e algoritmos que utilizem todo o poder computacional já é uma realidade, e para geração de malhas não poderia ser diferente. Gerar uma malha em um menor tempo possível explorando todo o poder computacional disponível é um grande objetivo a ser alcançado. Para atingir este objetivo com uma boa escalabilidade, ou seja, para ter um aumento no desempenho quando mais recursos computacionais forem disponíveis, mantendo os resultados obtidos pelo algoritmo sequencial, é preciso explorar o paralelismo de uma maneira inteligente e ordenada, sendo o uso desses recursos computacionais é um desafio a ser almejado pelas técnicas desenvolvidas na atualidade.

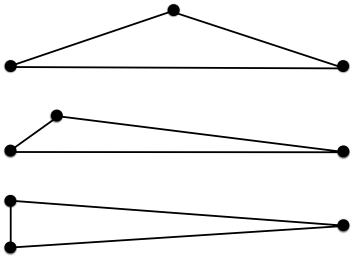
Na geração de malha para aplicações científicas é preciso também garantir que os elementos gerados sejam de boa qualidade, por exemplo, em MEF os elementos de uma malha são considerados bons se forem próximos ou igual a triângulos equiláteros (Figura 1.2a), caso contrário, ele pode ser ruim (Figura 1.2b). Esta qualidade dos triângulos também são importante para renderização de elementos em CG, CAD, bem como em várias outras aplicações.

Figura 1.2 – Exemplos de triângulos.

(a) Triângulo bom.



(b) Triângulos ruins.

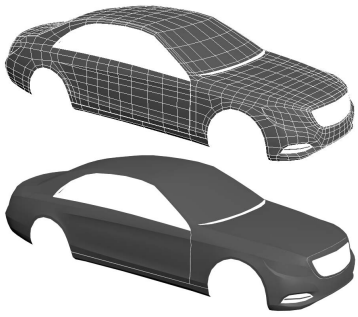


Fonte: Elaborada pelo autor.

Uma malha pode ser de vários tipos, como por exemplo: malhas de superfícies, que representam os limites de um determinado modelo (Figura 1.3a) e malhas volumétricas, que por sua vez, representam todo o volume do objeto (Figura 1.3b).

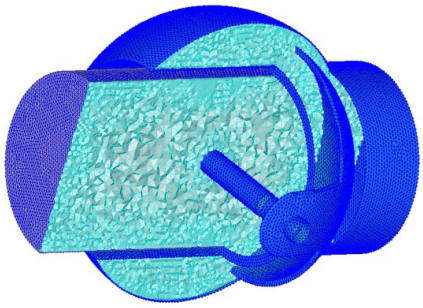
Figura 1.3 – Exemplos de malhas.

(a) Superfície.



Fonte: <http://goo.gl/X8wgHQ>.

(b) Volumétrica.



Fonte: <http://goo.gl/UpZyDd>.

Com as malhas geradas automaticamente, tornou-se possível simular fenômenos físicos com melhor precisão. Por isso a Computação de Alto Desempenho (HPC do inglês, *High-Performance Computing*) torna o processo de simulação mais rápido, já que várias unidades de processamento são utilizadas para uma única instância de um determinado problema.

Existem também simulações mais complexas, nas quais a cada passo é necessário gerar uma nova malha, como por exemplo, em simulações de problemas de propagação de trincas (Cozin (2008)), onde o bom desempenho na geração da malha está ligado diretamente ao poder computacional disponível. Portanto, o desenvolvimento de técnicas paralelas para geração de malhas com melhora no desempenho e diminuição no tempo de processamento é um objetivo a ser alcançado pelas técnicas, já que a grande maioria dos computadores, na atualidade, possuem mais de uma unidade de processamento disponível.

1.2 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo principal propor uma técnica de geração em paralelo de malhas de superfície paramétricas com controle de curvatura. Esta técnica paralela é baseada na técnica sequencial proposta por Siqueira *et al.* (2014), desenvolvida pelo mesmo grupo de pesquisa deste trabalho, e deve satisfazer os seguintes requisitos:

1. ser capaz de refinar e/ou desrefinar regiões de uma malha;
2. garantir uma boa transição entre regiões refinadas e não-refinadas;
3. garantir a compatibilidade entre regiões;
4. considerar a contribuição das medidas dos erros locais, a fim de garantir uma boa qualidade geral para a malha;
5. estimar com precisão o número de elementos gerados em cada subdomínio;
6. apresentar um bom balanceamento de carga, ou seja, distribuir uniformemente os subdomínios entre as unidades de processamento; e
7. apresentar um boa redução no tempo de execução e consequentemente um *speed-up* de boa qualidade.

Os quatro primeiros requisitos são oriundos da técnica sequencial e, portanto, a técnica paralela também deve satisfazê-los. O primeiro requisito mostra a capacidade de adaptabilidade da técnica, considerando que o grau de curvatura em cada região guiará o processo de geração da malha, onde regiões com maiores curvaturas serão mais refinadas e regiões com curvaturas baixas serão menos refinadas e/ou desrefinadas.

O segundo requisito está ligado à boa transição entre as malhas geradas nas regiões com variações de curvatura, ou seja, a adequação entre essas regiões tem que ser garantida pela técnica.

Com relação ao terceiro requisito, o mesmo está fortemente ligado às curvas que delimitam cada *patch*[1] que forma uma região; essas curvas garantem que *patches* vizinhos sejam compartilhados, garantido assim uma boa compatibilidade entre as regiões. Além disso, a boa transição entre regiões está fortemente ligada a este requisito, isso porque o processo adaptativo segue uma hierarquia, na qual as curvas são adaptadas antes do domínio.

[1] *patch* é o nome usual para partes de superfícies paramétricas, normalmente correspondendo ao domínio paramétrico completo. Seu conjunto é conhecido como *patches*.

O quarto requisito aborda a qualidade da malha gerada onde sua medição durante o processo de geração, em cada *patch*, se dará através do erro entre as curvaturas analíticas e discretas.

Os três últimos requisitos estão relacionados somente à técnica paralela e a intenção é que a implementação funcione em diversas plataformas paralelas, desde computadores pessoais com várias unidades de processamento aos computadores de alto desempenho, como do tipo *cluster*. Apesar da maioria dos *clusters* serem equipados com memória compartilhadas e distribuída, este trabalho utilizou somente a arquitetura de memória distribuída.

Com relação ao quinto requisito, este trata da capacidade da técnica em estimar a quantidade de elementos que será gerada no final do processo adaptativo, utilizando-se apenas das informações de entrada da estratégia, ou seja, informações geométricas dos *patches* paramétricos para gerar a estimativa. Por consequência desta estimativa, que por sinal é executada no início e apenas uma única vez durante o processo, o balanceamento de carga entre as unidades de processamento durante a execução da técnica também é garantido, satisfazendo, assim, o sexto requisito.

O último requisito está ligado ao quinto e sexto requisitos, uma consequência natural que, se bons resultados forem obtidos nestes dois requisitos, o tempo de execução e *speed-up* também serão bons para fins práticos.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em cinco capítulos. O segundo capítulo mostra os trabalhos relacionados ao tema, relacionando com pesquisas atuais na área de geração de malhas de superfície. No terceiro capítulo é apresentada a técnica proposta deste trabalho e o quarto capítulo mostra os resultados obtidos com a sua implementação. Por último, no quinto capítulo são apresentadas as conclusões acerca desta pesquisa e abordados alguns trabalhos futuros sobre a temática tratada.

2 TRABALHOS RELACIONADOS

2.1 INTRODUÇÃO

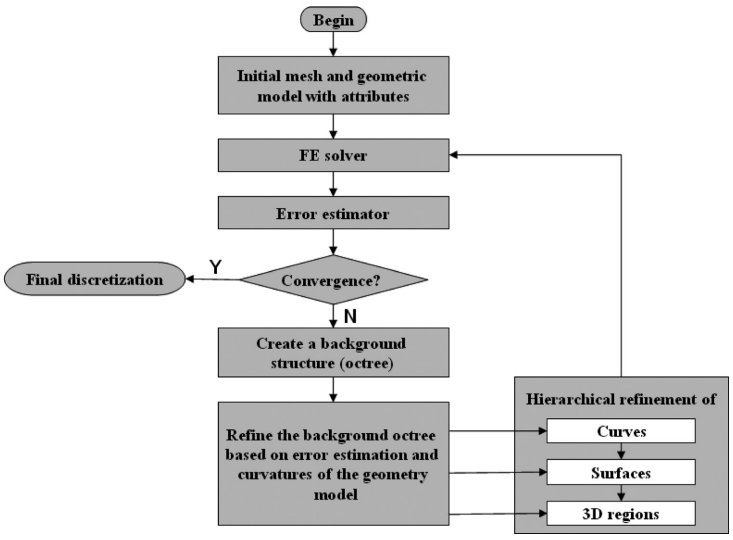
Este capítulo apresenta trabalhos relacionados ao tema abordado nessa pesquisa. Inicialmente serão abordados trabalhos sobre geração de malhas de superfície de forma sequencial (Seção 2.2), e logo em seguida, as pesquisas que tratam de geração de malha de superfície em paralelo (Seção 2.3). Como o foco desse trabalho é a geração de malha em paralelo, é dada mais atenção a trabalhos relacionados a paralelismo, sendo relatados somente os trabalhos sequenciais que são relacionados fortemente ao tema dessa pesquisa.

2.2 GERAÇÃO SEQUENCIAL

O trabalho de Miranda *et al.* (2013) descreve uma metodologia para geração de malha 3D (espaço tridimensional) de EF (elementos finitos) de uma forma adaptativa, utilizando-se de modelagem geométrica com multirregiões e superfícies paramétricas.

Resumidamente, o processo adaptativo envolve três etapas: a primeira, a análise de EF com a estimativa de erro usando curvas, superfícies e volume; a segunda, a construção de uma estrutura de *background* global, que é composta por uma composição espacial recursiva representada por uma *octree*, para armazenar os novos tamanhos dos elementos finitos e, por último, o refinamento, que prossegue numa adaptação independente e hierárquica do modelo de EF. Na Figura 2.1 é mostrado um quadro resumo do processo adaptativo.

Figura 2.1 – Processo de refinamento adaptativo de Miranda *et al.* (2013).



Fonte: Miranda *et al.* (2013).

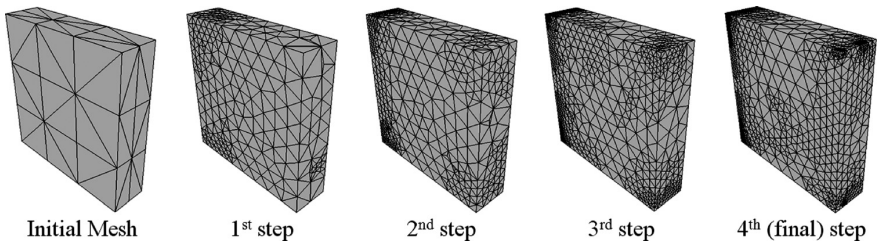
A proposta mostrou-se capaz de refinar e desrefinar regiões de alto e baixo gradientes de resposta, respectivamente. Ela usa apenas uma *octree* de *background* para todas as regiões do modelo, permitindo, assim, uma transição suave entre as regiões e elementos que formam o modelo, utilizando uma estrutura de refinamento hierárquica, onde as curvas, as superfícies e os volumes são refinados, respectivamente. Já as informações de curvatura das curvas e superfícies complementam o processo de refinamento adaptativo.

Os resultados obtidos pela estratégia demonstraram que ela se mostrou mais eficaz do que trabalhos consultados na literatura, convergindo mais rápido com um erro relativo menor. Esta eficácia só foi possível devido ao gerador de malha 3D ter uma característica de liberdade para criar novos elementos com base nos tamanhos desejados, resultando assim, em uma malha gerada com poucos passos no processo adaptativo.

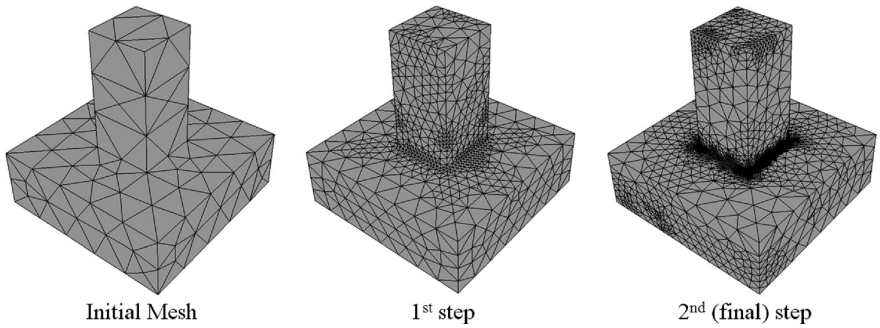
Além disso, os autores deixam claro que os exemplos práticos utilizados no trabalho foram pequenos (Figura 2.2), e que foram gerados em menos de um minuto, mas uma possível solução proposta no trabalho para modelos maiores é decompor o domínio em subdomínios e aplicar o gerador para cada um. Uma outra possível solução seria usar um gerador de malhas 3D em paralelo. Esta última opção só reforça a importância de se utilizar todo o poder computacional de métodos paralelos.

Figura 2.2 – Exemplos práticos da estratégia de refinamento.

(a) Exemplo 1.



(b) Exemplo 2.



Fonte: Miranda *et al.* (2013).

Li *et al.* (2013) abordam a medida de convergência para a parametrização baseada

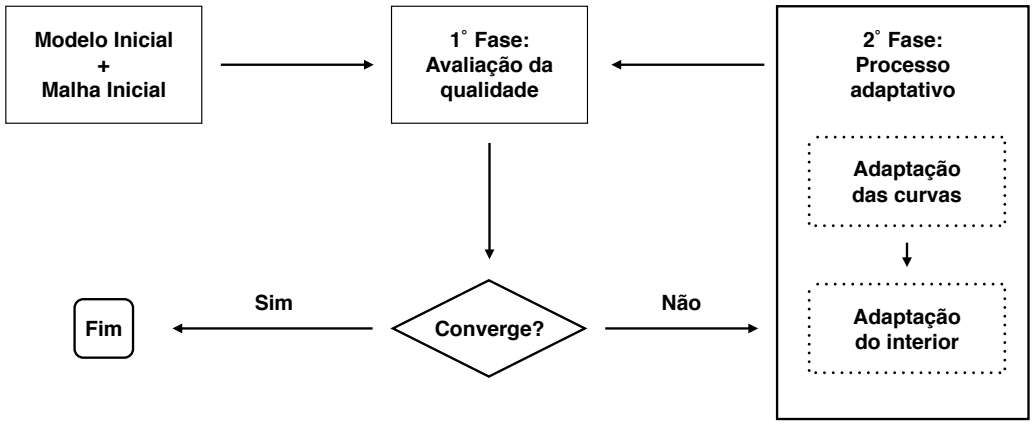
no algoritmo de Delaunay. Usando as métricas da superfície, a proposta é triangular todo seu domínio usando Delaunay, e, por consequência, gerar uma malha de qualidade. Utilizando-se de estimativas explícitas para a distância de Housdorff, o desvio normal e as medidas de curvatura entre a superfície e a malha, a medida da curvatura Gaussiana é inerente à métrica de Riemann.

Após gerar a malha, um dos critérios utilizados para mediar a qualidade da malha gerada é a curvatura Gaussiana e, em um exemplo prático, é exposto um comparativo entre a curvatura média e a Gaussiana da malha. O resultado foi bastante promissor, demonstrando que a curvatura Gaussiana se comportou melhor e obteve excelente resultado, apesar da curvatura média também ter obtido um resultado bom.

A principal contribuição deixada pelo trabalho de Li *et al.* (2013) para a presente pesquisa é o fato de que a convergência pelo uso da curvatura Gaussiana pode ser obtida, e a validação deste tipo de curvatura é um ponto positivo para a continuidade de sua utilização.

Siqueira *et al.* (2014) propõem uma estratégia hierárquica de geração de malhas adaptativas de superfícies paramétricas baseada em uma estrutura de árvore binária, onde o erro entre as curvaturas analítica e discreta, calculado para a superfície da malha, guia o processo adaptativo de geração da malha, que utiliza a técnica de avanço de fronteira em conjunto com a de Delaunay. A Figura 2.3 demonstra como se dá o esquema de refinamento do processo.

Figura 2.3 – Resumo do processo adaptativo de Siqueira *et al.* (2014).



Fonte: Siqueira *et al.* (2014).

O processo consiste, basicamente, de duas fases, por iteração: a primeira é inicializada através de dados do modelo geométrico: os *patches*, as curvas e a malha da iteração

corrente. Caso seja a primeira iteração, a malha inicial já está predefinida e nas demais a malha é calculada na iteração anterior. A qualidade da malha é mensurada com base nos cálculos dos erros entre as curvaturas analíticas e discretas. A próxima fase é baseada na malha da iteração corrente, na qual uma nova malha é gerada, pela adição de mais elementos em algumas regiões, podendo ou não acontecer remoção de elementos em outras. O processo iterativo se encerra quando um critério, predefinido, de qualidade global, é atingido. Como esta estratégia é a base do presente trabalho, ela será abordada com mais detalhes no próximo capítulo, o de técnica proposta.

2.3 GERAÇÃO EM PARALELO

Estudos sobre geração de malhas volumétricas em paralelo podem ser encontrados na literatura em abundância. Em contrapartida, quando se trata de geração de malhas de superfície em paralelo há uma escassez de trabalhos. Por este motivo, em determinados trabalhos que tratam de geração de malhas de superfície e volumétricas em uma mesma pesquisa, o foco será dado na geração das malhas de superfície.

Dentre os trabalhos pesquisados destacam-se o de Kohout *et al.* (2005) no qual são apresentadas técnicas de Delaunay com implementação prática, usando memória compartilhada. A estratégia usa um grafo direcionado acíclico para auxiliar na busca por triângulos, influenciados pela inserção de um novo ponto. Três métodos foram utilizados, e a principal diferença entre eles é a forma de acesso ao grafo, que por sua vez, é compartilhado entre as unidades de processamento. Na criação de um novo ponto, os métodos utilizam busca ou inserção concorrentemente. No primeiro método, a inserção é feita apenas por um único processador e os outros processadores fazem a busca no grafo. No segundo, a inserção é feita por qualquer um dos processadores disponíveis com acesso exclusivo a todo o grafo e a busca é feita simultaneamente por todos os processadores. No terceiro, a inserção e busca são feitas de forma simultânea, a inserção é feita de forma exclusiva apenas para alguns nós do grafo, com o auxílio de uma técnica de detecção de *deadlock*, em que os processadores envolvidos decidem qual deles executará a inserção em detrimento dos outros, que, consequentemente, faz com que os outros processadores retrocedam no procedimento. O processo total poderá ser custoso devido à detecção de *deadlock*, que poderá causar situações em que as *threads* tenham que desfazer várias vezes o trabalho já feito, derrubando assim, o desempenho do método.

O trabalho de Tremel *et al.* (2005) aborda um esquema que considera gerar a malha de cada superfície como uma tarefa, e a geração de malhas dessas tarefas entre todas as superfícies são distribuídas em diferentes processos e realizada em paralelo. Este esquema é essencialmente não escalável e a sua eficiência tem uma forte dependência com a natureza geométrica do modelo. Pode-se destacar a seguinte relação: se o número de superfícies S

é muito maior do que o número de processos P, e os tempos de processamento de cada superfície são equivalentes, com estas características a estratégia poderá atingir uma alta eficiência; caso contrário, a eficiência poderá ser bastante baixa.

Na técnica apresentada por Ito *et al.* (2007) é proposto um ambiente de implementação para gerar malhas em larga escala. A malha é dividida em subdomínios por um algoritmo particionador, e logo em seguida, de forma paralela, a malha de superfície de cada subdomínio é refinada usando o método de Delaunay no espaço bidimensional, sendo finalmente a malha gerada usando avanço de fronteira. A técnica paralela conseguiu gerar malhas com reduções nos tempos de execução mantendo uma boa qualidade. Vale ressaltar a importância da qualidade da malha de superfície para gerar uma malha de boa qualidade, onde o bom desempenho do algoritmo proposto está fortemente ligado à qualidade da malha de superfície, ou seja, se a malha de superfície gerada tiver uma boa qualidade haverá um ganho no desempenho do algoritmo, caso contrário, afetará de forma negativa todo o processo de geração, no que diz respeito ao tempo e à qualidade. Portanto, a malha de superfície é fundamental para a qualidade de todo processo de geração.

No trabalho de Yılmaz *et al.* (2010) são propostos três métodos para geração de malhas para domínios com geometrias simples. Os três métodos utilizam-se inicialmente da geração de uma malha grosseira de maneira sequencial para uma determinada geometria. A diferença entre os três métodos é a forma de decompor a malha, onde o primeiro método é alinhado aos eixos, no segundo a distribuição dos elementos é feita usando um particionador de grafos, e, por último, assim como no segundo, utiliza-se outro particionador de malha.

Figura 2.4 – Terceira etapa de geração de malha de superfície do primeiro método de Yılmaz *et al.* (2010).

Fonte: Adaptado de Yılmaz *et al.* (2010).

A geração de malhas de superfície da proposta é feita de forma sequencial; logo

após a decomposição do domínio, uma malha de superfície refinada é gerada seguindo essa decomposição. As malhas geradas apresentaram problemas de qualidade, provavelmente, devido à presença de tetraedros ruins na malha grosseira e, consequentemente, os mesmos, quando gerados na nova malha refinada, têm qualidades equivalentes aos tetraedros da malha grosseira. Além disso, outro ponto a ser destacado é a escalabilidade, que não se mostrou boa devido às etapas sequenciais pertencentes à estratégia.

O trabalho de Zhao *et al.* (2015) apresenta uma proposta de melhora, em termos de escalabilidade e eficiência, em relação ao trabalho de Tremel *et al.* (2005). No algoritmo melhorado, uma superfície com grande carga é dividida em vários subdomínios menores, e nestes são geradas as malhas de superfície individualmente. O número total de subdomínios é determinado em tempo de execução e escalável no número de processadores; desta forma, o número de tarefas não depende da carga de superfícies, e sim do controle exercido pelo usuário ou pela estratégia. Desta maneira, os autores conseguiram equilibrar as cargas entre as tarefas com subdomínios balanceados. Este trabalho apresenta, além da geração de malhas de superfície, a geração de malhas volumétricas, porém conforme dito anteriormente, não foram abordados aspectos relacionados com esta etapa.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados alguns trabalhos relevantes relacionados ao tema abordado nesta pequisa. Os trabalhos apresentados na Seção 2.2 abordam técnicas sequenciais de avanço de fronteira, Delaunay. A Seção 2.3 mostra trabalhos em que o paralelismo é usado para geração de malhas de superfície, através de memória compartilhada ou distribuída.

3 TÉCNICA PROPOSTA

3.1 INTRODUÇÃO

O objetivo principal deste trabalho é desenvolver uma técnica de geração de malhas de superfície em paralelo para computadores com memória distribuída, que funcione em modelos formados por suas descrições geométricas (*patches*), gerando uma malha de boa qualidade e apresentando um bom balanceamento de carga, bem como um bom *speed-up*.

A técnica proposta neste trabalho é baseada em uma técnica de geração de malhas de superfície sequencial existente, desenvolvida pelos mesmos autores desse trabalho. Essa técnica sequencial foi inicialmente proposta por Souza (2004), utilizando-se da técnica de *quadtree* e depois melhorada por Siqueira *et al.* (2014), adicionando a técnica de avanço de fronteira.

Além disso, ela deve satisfazer os quatro requisitos, já mencionados nos itens 1, 2, 3 e 4 da Seção 1.2, que motivaram o desenvolvimento da técnica sequencial e os três últimos critérios, também mencionados na Seção 1.2, que, por sua vez, estão relacionados à técnica paralela. Portanto, a técnica desenvolvida respeita os critérios da técnica sequencial com algumas adaptações e melhorias para facilitar a implantação do paralelismo, com o intuito de atingir bons resultados.

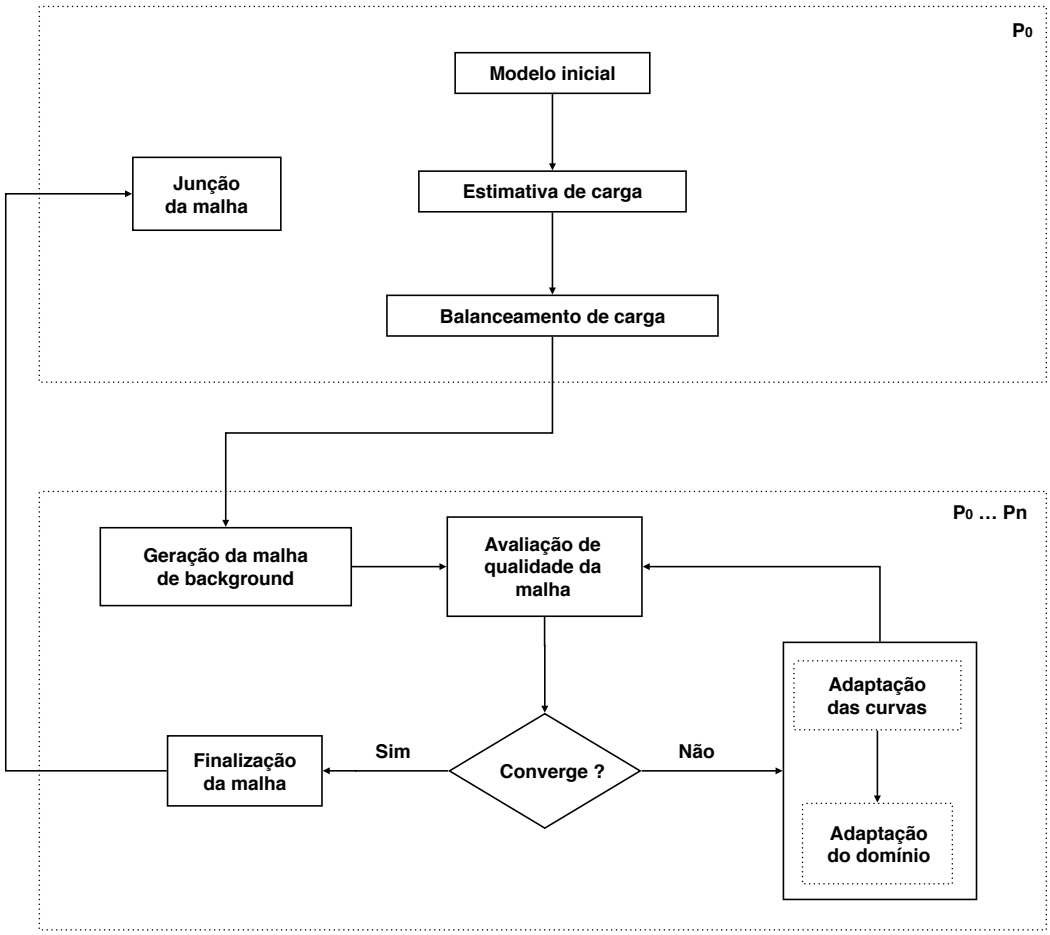
A técnica de geração de malhas de superfície de forma paralela é exposta neste capítulo seguindo a divisão em dois grupos: o primeiro mostra as tarefas que são executadas de forma sequencial e o segundo contempla as que são executadas de forma paralela.

A estimativa de carga, o balanceamento de carga e a junção da malha são realizados de forma sequencial; já as tarefas de geração da malha de background, avaliação de qualidade da malha, adaptação das curvas e do domínio e a finalização da malha são efetivadas de forma paralela. A Figura 3.1 mostra a sequência da estratégia proposta, bem como uma visão geral das tarefas.

Analisando a Figura 3.1, observa-se o grupo que representa as tarefas sequenciais, circunscrito por P_0 e o que representa as tarefas paralelas, circunscrito por $P_0 ... P_n$. A ligação entre esses grupos acontece após a etapa sequencial de balanceamento de carga, e, somente após esta etapa, a técnica de geração de malhas de superfície é inicializada e permanece em um ciclo iterativo entre a medição da qualidade da malha, adaptação das curvas e do domínio em cada unidade de processamento. Caso haja convergência, a técnica é finalizada na sua respectiva unidade de processamento, que envia a malha gerada ao processo mestre, o qual é responsável pela junção das malhas geradas pelos vários

processos.

Figura 3.1 – Visão geral da técnica proposta.



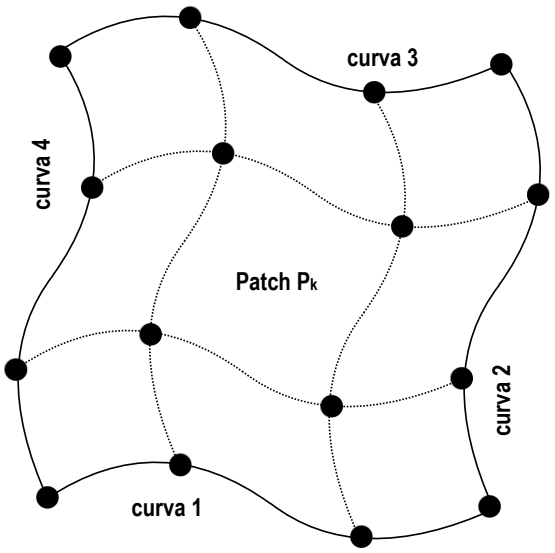
Fonte: Elaborada pelo autor.

3.2 MODELO INICIAL

O modelo de entrada da estratégia é composto por um conjunto de *patches* paramétricos, tais como *patches* de Coons, como por exemplo Bezier e Hermite, com suas devidas descrições geométricas e curvas delimitantes, formando assim, sua geometria. Pode-se verificar na Figura 3.2 um exemplo de *patch* paramétrico, P_k, com suas informações geométricas e curvas delimitantes, que é utilizado como modelo didático para compreensão da técnica dentro deste capítulo.

Após a entrada das informações de curvas e descrições geométricas do modelo, a técnica segue com a estimativa de carga.

Figura 3.2 – *Patch* paramétrico P_k.



Fonte: Elaborada pelo autor.

3.3 ESTIMATIVA DE CARGA

Em Computação de Alto Desempenho, o valor que determina a quantidade de computação a ser realizada em um processo é chamado de carga. Neste trabalho, como cada processo é responsável por um subdomínio, e cada subdomínio é formado por um conjunto de *patches*, a computação desses subdomínios deve refletir no tempo total de geração da malha de superfície. Entretanto, prever o tempo é um problema bastante difícil, isso porque exitem fatores externos à implementação, como por exemplo, tempos de acesso à memória *cache* e RAM, tempo de comunicação entre processos e outros fatores. Portanto, o método usado nesta pesquisa é baseado no cálculo da complexidade do algoritmo.

A complexidade dos algoritmos de geração de malhas que utilizam o método de avanço de fronteira é dada em função da quantidade de elementos gerados. A estimativa de carga é utilizada devido à dificuldade encontrada para se calcular a carga em função da entrada. Portanto, o presente trabalho utiliza uma estimativa de carga proporcional à quantidade de elementos gerados, ou seja, proporcional à saída do algoritmo.

O cálculo da estimativa de carga para cada *patch* é obtido de forma individual, isto é, cada *patch* é calculado de forma isolada e não sofre interferência dos *patches* vizinhos, usando a seguinte estratégia:

1. inicialmente é calculada a área do *patch* P_k (Subseção 3.3.1);
2. logo em seguida, é obtida a curvatura Gaussiana analítica média G_{am} de P_k (Subseção 3.3.2);

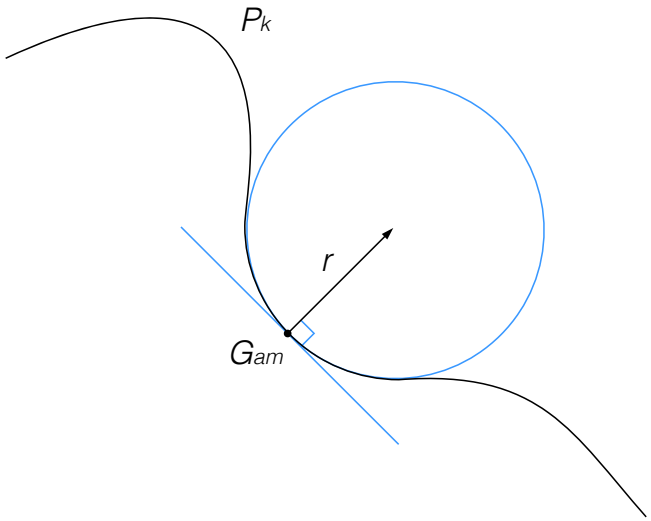
3. posteriormente, é calculada a relação entre a curvatura Gaussiana média e o raio de uma circunferência (Figura 3.3), como mostra a Equação 3.1:

$$G_{am} = \frac{1}{r}, \tag{3.1}$$

onde r é o valor do raio. Isso resulta em um valor que representa o lado L_k de um triângulo médio T_k;

4. por último, utilizando-se L_k, calcula-se a área T_k, e a relação entre a área de P_k e T_k resulta na estimativa de carga E_{ci} para o *patch* P_k.

Figura 3.3 – Círculo osculador no *patch* P_k (corte longitudinal).

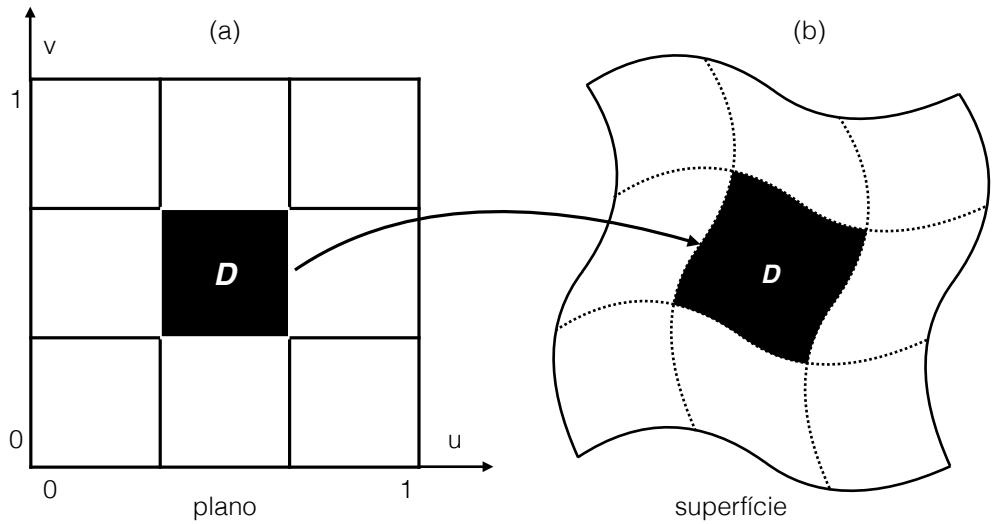


Fonte: Elaborada pelo autor.

3.3.1 Cálculo da área de um *patch*

Este trabalho segue a formulação apresentada por Nykamp (2016) para o cálculo da área aproximada de uma superfície paramétrica utilizando soma de Riemann. A área é calculada através de uma aproximação da integração do mapeamento de regiões de um domínio paramétrico para uma superfície, representada pela função $\Phi : \mathbf{R}^2 \rightarrow \mathbf{R}^3$. A Figura 3.4 mostra uma região D com seu respectivo mapeamento, onde a) representa o plano com a região D em destaque e b) o mapeamento de D para uma superfície.

Figura 3.4 – Mapeamento da região D do domínio paramétrico para uma superfície.



Fonte: Elaborada pelo autor.

O próximo passo para calcular a área da superfície P_k é estimar a área de cada um dos retângulos curvilíneos. Para cada retângulo D, temos $\Delta u \times \Delta v$ e a aproximação dos vetores desse retângulo, representado por $\frac{\partial \Phi}{\partial u}\Delta u$ e $\frac{\partial \Phi}{\partial v}\Delta v$, onde $\Phi(u, v)$ é um ponto no espaço tridimensional, e então a área da região D é obtida através do produto vetorial entre esses dois vetores, como podemos observar na Equação 3.2:

$$\Delta A = \left\| \frac{\partial \Phi}{\partial u} \times \frac{\partial \Phi}{\partial v} \right\| \Delta u \Delta v. \tag{3.2}$$

Desta forma a área total da superfície é uma aproximação de uma soma de Riemann de tais termos. Se Δu e Δv forem próximos de zero, a área total da superfície é a integração dupla da Equação 3.2, resultando na Equação 3.3:

$$A = \iint_D \left\| \frac{\partial \Phi}{\partial u}(u, v) \times \frac{\partial \Phi}{\partial v}(u, v) \right\| dudv, \tag{3.3}$$

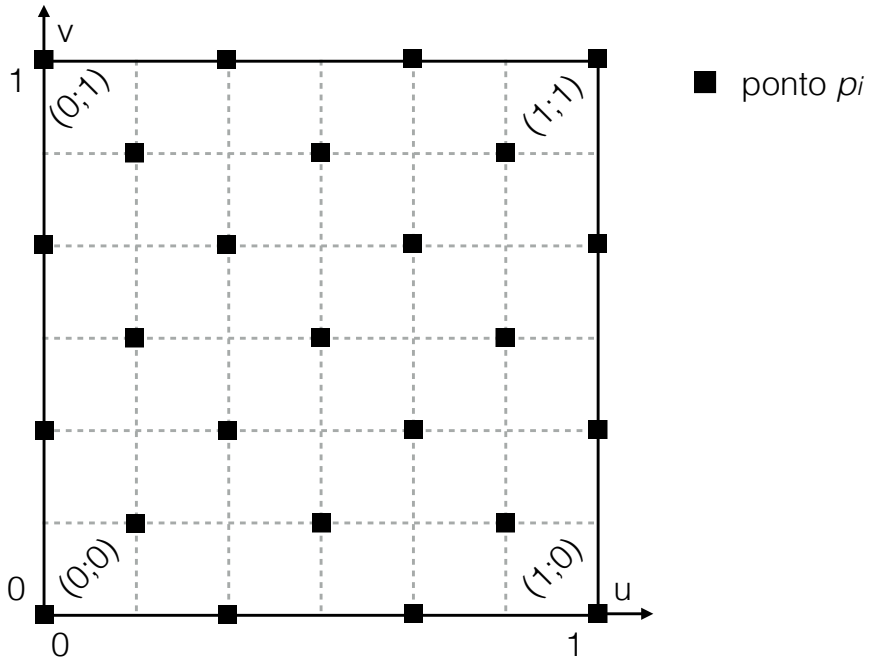
onde A é área aproximada da superfície paramétrica.

Com a área de *patch* P_k calculada, o próximo passo é obter a curvatura Gaussiana analítica média de P_k.

3.3.2 Cálculo da curvatura gaussiana analítica média

A curvatura Gaussiana analítica média é obtida através da seleção de pontos dentro do espaço paramétrico no *patch* P_k. A Figura 3.5, mostra a disposição dos pontos.

Figura 3.5 – Pontos selecionados para o cálculo da curvatura Gaussiana analítica média no *patch* P_k.



Fonte: Elaborada pelo autor.

Após a seleção dos pontos, calcula-se a curvatura Gaussiana analítica G_a de cada ponto p_i (este calculo é abordado de forma mais detalhada na Subseção 3.6.1) e na sequência a média entre estas curvaturas é obtida, como mostra a Equação 3.4:

$$G_{am} = \frac{\sum_{i=1}^{n} G_a(i)}{n}, \tag{3.4}$$

em que G_{am} é a soma das curvaturas Gaussiana analíticas G_a de cada ponto p_i, dividida pela quantidade de pontos n.

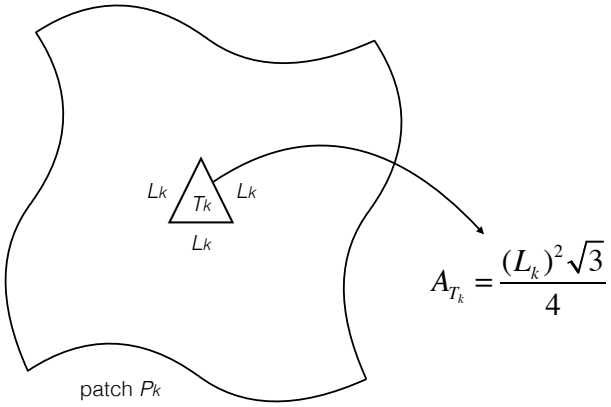
3.3.3 Quantidade de elementos estimados

Com as informações de G_{am}, a estratégia estipula L_k, usando a relação entre uma curvatura e o raio de uma circunferência, como mostra a Equação 3.5:

$$L_k = \frac{1}{G_{am}}\theta, \tag{3.5}$$

onde θ é um ângulo pequeno e L_k é utilizado para a obtenção da área do triângulo equilátero T_k (a Figura 3.6 mostra a área do triângulo T_k).

Figura 3.6 – Área do triângulo T_k no *patch* P_k.



Fonte: Elaborada pelo autor.

Para encontrarmos a quantidade de elementos estimados E_{ci} para o *patch* P_k, é só dividir a área A de P_k pela área de T_k, como mostra a Equação 3.6:

$$E_{ci} = \frac{A}{A_{T_k}}, \tag{3.6}$$

gerando assim uma estimativa na superfície de P_k da quantidade de elementos. Caso o *patch* seja plano, E_{ci} é sempre igual a 2, não sendo necessário o cálculo de A e A_{T_k}, já que G_{am} é igual a zero.

Apesar da técnica proposta neste trabalho apresentar característica adaptativa, o foco da estratégia é estimar a quantidade de elementos e obter o número de elementos proporcionais à quantidade que será gerada em cada *patch*. Considerando que os *patches* do modelo de entrada são bem comportados, ou seja, não há uma alta variação entre os tamanhos e as curvaturas dos mesmos, e que esta estimativa guia a próxima etapa de balanceamento de carga através da distribuição entre as unidades de processamento, de acordo com os valores de carga estimado, o número exato de elementos não é o foco desta estimativa, sendo o principal objetivo atingir um bom balanceamento de carga.

3.4 BALANCEAMENTO DE CARGA

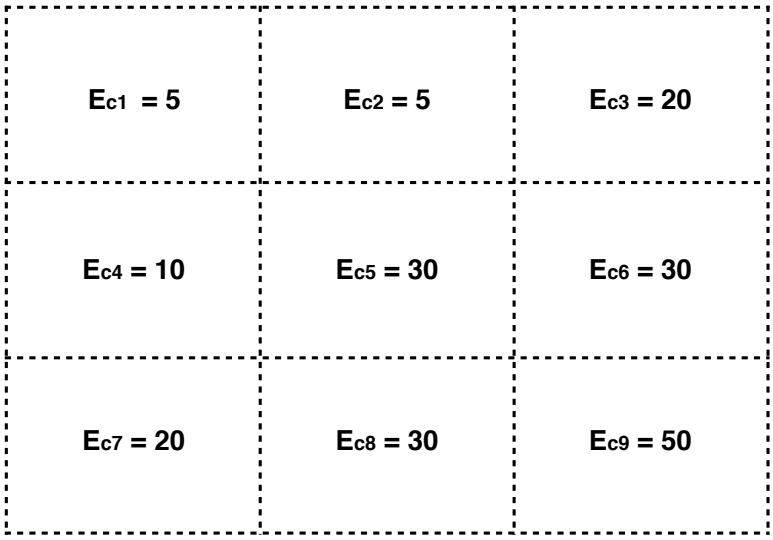
O balanceamento de carga visa a distribuição da carga de trabalho, de acordo com a disponibilidade de unidades de processamento e recursos de cada máquina no sistema computacional. O foco desta distribuição é maximizar a utilização de recursos e possibilitar um melhor desempenho da técnica aqui proposta.

Utilizando-se de uma decomposição de dados de entrada com característica de grossa granularidade (devido ao menor custo de sincronização e à quantidade de processamento em cada CPU ser maior do que a comunicação entre os processos), o balanceamento

desta técnica proposta se dá através da informação de carga calculada pela estimativa de carga. Utilizando esta carga como métrica, os *patches* são distribuídos entre os processos disponíveis para o processamento de maneira ordenada e balanceada, resultando, no final do balanceamento, em uma quantidade de carga equivalente entre os processos. Portanto, o fator que define a quantidade de *patches* em cada processo é a carga acumulada nele em comparação com os outros processos.

Para demonstrar melhor o balanceamento será utilizado um modelo didático M_k formado por nove *patches* com seus devidos valores estimados de carga (E_{ci}). A Figura 3.7 mostra M_k, em que E_{ci} representa os valores estimados de carga para cada um dos *patches*.

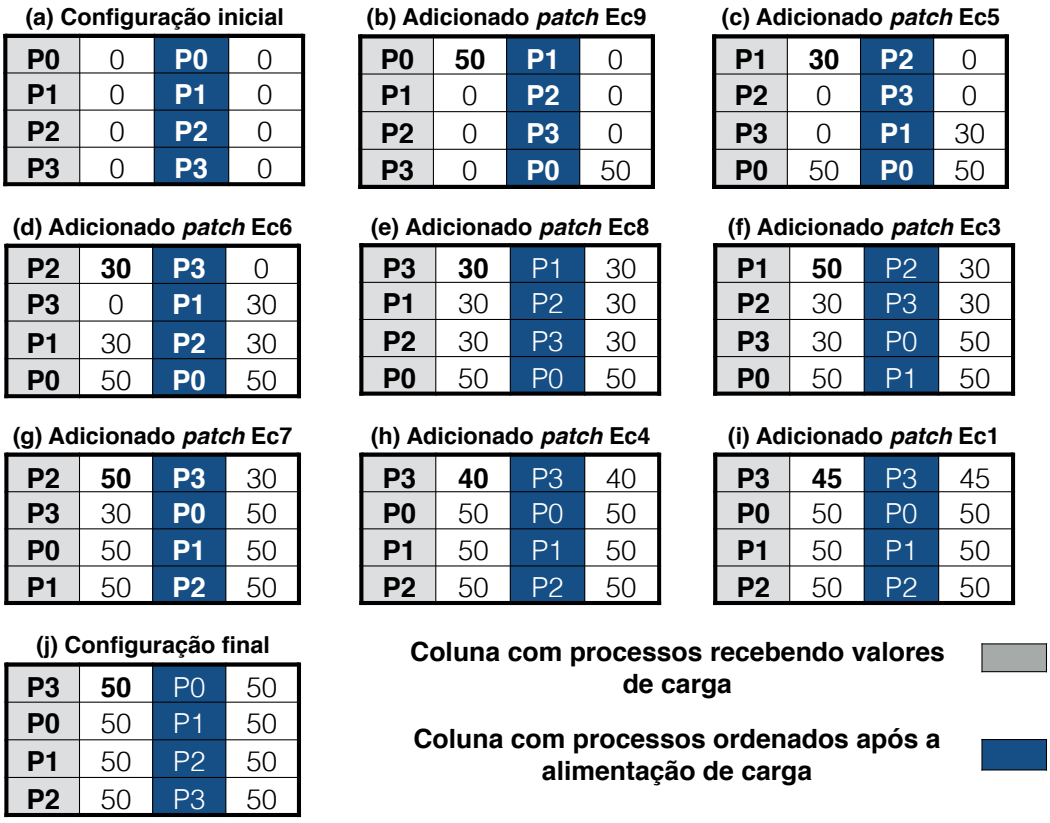
Figura 3.7 – Modelo didático M_k.



Fonte: Elaborada pelo autor.

O balanceamento de carga tem início através da ordenação, de forma decrescente, da lista formada pelos valores de carga estimada dos *patches*. Na sequência os valores de carga são alocados em cada processo seguindo dois passos: o primeiro refere-se ao processo que recebe o valor da carga; o segundo está relacionado a ordenação dos processos através dos valores de suas cargas. Um fator importante é que essas duas fases sempre são executadas em conjunto e seguindo esta sequência. A Figura 3.8 exibe um conjunto de tabelas que mostram as ações para o balanceamento do modelo M_k.

Figura 3.8 – Balanceamento de carga para o modelo M_k.



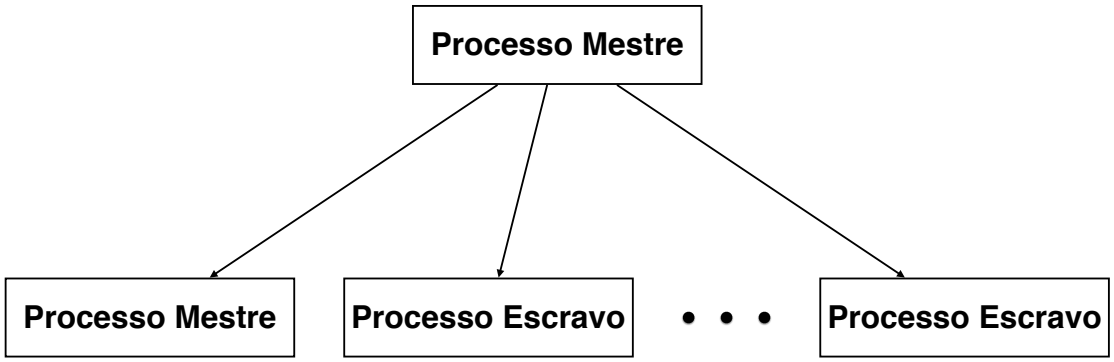
Fonte: Elaborada pelo autor.

Analisando as tabelas da Figura 3.8, observa-se que a primeira coluna está ordenada de cima para baixo levando-se em conta a prioridade para receber os valores de carga, ou seja, o processo que estiver na primeira linha recebe a carga e esta carga é lançada na primeira linha da segunda coluna. A terceira e quarta colunas estão relacionadas à fase de ordenação, onde o processo que tiver a menor carga é alocado com a prioridade para receber a próxima carga. Desta forma tem-se em a) a alocação dos processos com valores 0 e obviamente a ordenação permanece intacta, em b) o processo $P0$ recebe o primeiro valor da lista e em seguida os processos são ordenados e assim sucessivamente, até que, por fim, em j) todos os processos estão com suas cargas balanceadas.

Após o balanceamento, as tarefas são distribuídas entre os processos como pode-se ver na Figura 3.9, em que todos os processos participam das etapas que estão por vir, inclusive o processo mestre, que faz a distribuição.

Vale ressaltar que as etapas até aqui explicadas são executadas de maneira sequencial pelo processo mestre, tanto a estimativa como o balanceamento de carga, e só após estas duas etapas, é que a técnica proposta é executado de forma concorrente.

Figura 3.9 – Distribuição de carga após o balanceamento.

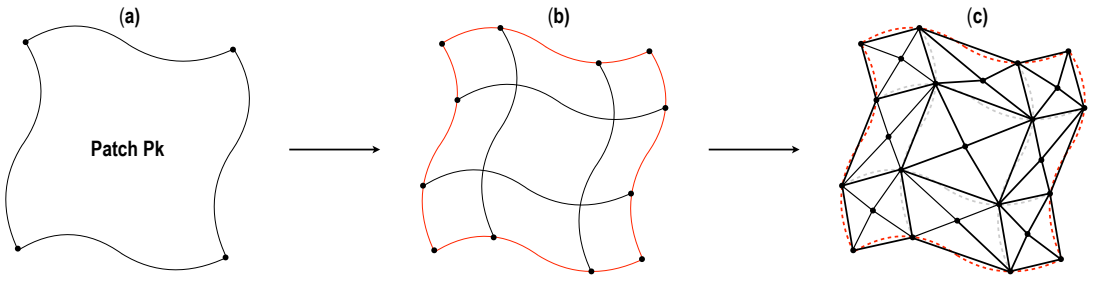


Fonte: Elaborada pelo autor.

3.5 GERAÇÃO DA MALHA DE *BACKGROUND*

A malha de *background* do modelo é formada pelo conjunto de malhas de *background* de cada *patch*. Usando o *patch* P_k como exemplo sua malha de *background* é M_k. A Figura 3.10 mostra os passos da geração de M_k: em a) vê-se o *patch* P_k como suas curvas delimitantes, logo em seguida P_k é dividido em nove regiões através da discretização das curvas delimitantes, como se vê em b). Por último, cada região, auxiliada pela criação de um ponto no centroide de cada uma é divida em quatro sub-regiões com mostrada em c), formando assim uma malha de *background* inicial. Apesar da técnica aqui proposta gerar a malha de *background*, a mesma pode ser dada como entrada pelo usuário (Siqueira *et al.* (2014)).

Figura 3.10 – Malha de *background* para o *patch* o P_k.



Fonte: Adaptado de Siqueira *et al.* (2014).

3.6 AVALIAÇÃO DA QUALIDADE DA MALHA

Durante a etapa adaptativa em cada processo, a qualidade da malha gerada é avaliada levando-se em conta um critério de estimativa de erro, que calcula em cada nó da malha a diferença entre as curvaturas analítica e discreta da superfície. Essas curvaturas são detalhadas nas Subseções 3.6.1 e 3.6.2, respectivamente.

A relação entre esses dois tipos de curvaturas em cada nó da malha é chamado de erro local η_l, e a média do conjunto de erros locais é tratada como erro global em um processo η_{gp}. Se o erro global numa unidade de processamento atingir uma precisão desejada, o processo adaptativo é interrompido e finaliza para aquele processo, ou seja, sua convergência é atingida.

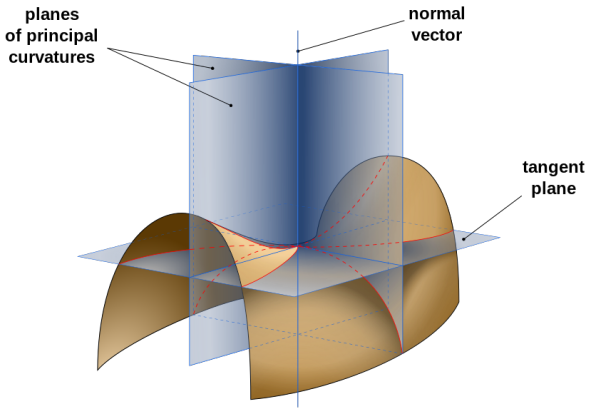
Portanto, o erro local é que indica se uma determinada região da malha precisa ser refinada, desrefinada ou se já atingiu a precisão desejada. Um outro ponto importante a ser tratado é que esta etapa se repete durante todo o procedimento iterativo, em cada processo, sempre tomando como entrada a malha gerada no passo anterior. Assim sendo, todos os valores de adaptação têm como base essa malha e só após o término da avaliação no passo corrente é que a malha gerada no passo anterior é descartada.

3.6.1 Cálculo da curvatura analítica

A curvatura analítica, em linha gerais, é obtida através da média de duas curvas imaginárias que se formam sobre a superfície, levando-se em conta um nó, em que tais curvas são as que possuem maior e menor curvatura sobre a superfície. Segundo Rogers e Adams (1990), dado um nó n qualquer sobre uma superfície paramétrica, se através deste nó traça-se um plano normal em relação à superfície, onde a interseção do plano com a superfície define uma curva, obtemos a curvatura Gaussiana analítica.

Em outras palavras, a curvatura analítica é formada pelo plano de curvatura mínima K_{mim} e máxima K_{max}, que se dá traçando uma normal sobre o nó n e rotacionando o plano anteriormente citado sobre n para encontrar as duas únicas direções principais, as quais representam a curvatura máxima e mínima, como pode-se ver na Figura 3.11.

Figura 3.11 – Curvatura Gaussiana de uma superfície bi-paramétrica.



Fonte: Wikipedia (2015).

Uma formulação matemática para o *patch* se faz necessária para o cálculo. As duas curvaturas principais são utilizadas para calcular a curvatura média H e a curvatura Gaussiana K, como pode-se observar nas equações 3.7 e 3.8:

$$H = \frac{K_{min} + K_{max}}{2}, \tag{3.7}$$

$$K = K_{min}.K_{max}, \tag{3.8}$$

onde a curvatura analítica média é obtida através da média entre as curvaturas máxima e mínima e a Gaussiana por um produto entre a mínima e a máxima.

Segundo Dill (1981), as equações que representam as curvaturas analíticas média e Gaussiana para as superfícies bi-paramétricas são as seguintes:

$$H = \frac{A|\Phi_v|^2 - 2B\Phi_u.\Phi_v + C|\Phi_u|^2}{2|\Phi_u \times \Phi_v|^2}, \tag{3.9}$$

$$K = \frac{AC - B^2}{|\Phi_u \times \Phi_v|^4}, \tag{3.10}$$

na qual os valores A, B e C são:

$$\begin{cases} A = [\Phi_u \times \Phi_v].\Phi_{uu} \\ B = [\Phi_u \times \Phi_v].\Phi_{uv} \\ C = [\Phi_u \times \Phi_v].\Phi_{vv}, \end{cases} \tag{3.11}$$

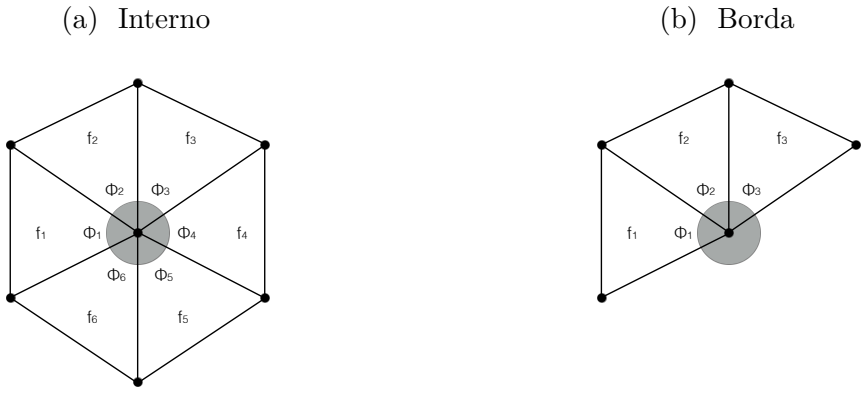
em que as subscrições são notações para derivadas parciais (por exemplo: $\Phi_u = \partial\Phi/\partial u$, $\Phi_v = \partial\Phi/\partial v$) e para os demais segue a mesma lógica de derivação.

3.6.2 Cálculo da curvatura discreta

A curvatura discreta é calculada usando informações contidas na malha, sendo sua avaliação composta da seguinte forma: dado um nó n da malha, as curvaturas discretas

Gaussiana e média são calculadas usando informações dos triângulos adjacentes ao nó n. Kim *et al.* (1999) apresentam operadores de curvatura, com base no esquema Gauss-Bonnet para o cálculo da curvatura discreta Gaussiana. Um nó em uma malha pode ser interno ou de borda, como mostra a Figura 3.12.

Figura 3.12 – Nó em uma curvatura discreta Gaussiana.



Fonte: Adaptado de Siqueira *et al.* (2014).

Quando o nó for interno, a curvatura discreta Gaussiana K é representada pela Equação 3.12:

$$K = \frac{2\pi - \sum_{i=1}^{n} \Phi_i}{\frac{1}{3}A_f}, \tag{3.12}$$

e quando o nó for de borda, a curvatura discreta Gaussiana K é dado pela Equação 3.13:

$$K = \frac{\pi - \sum_{i=1}^{n} \Phi_i}{\frac{1}{3}A_f}, \tag{3.13}$$

na qual Φ_i é o ângulo do nó, e a soma das áreas das faces é representada por A_f.

Vale ressaltar que em determinadas regiões onde a curvatura discreta Gaussiana se aproxima de zero há um indicativo de que a região em questão possui características planares ou próximas a estas. Quando isso acontecer, o operador de curvatura discreta Gaussiana será substituído pelo de curvatura discreta média, representado pela Equação 3.14:

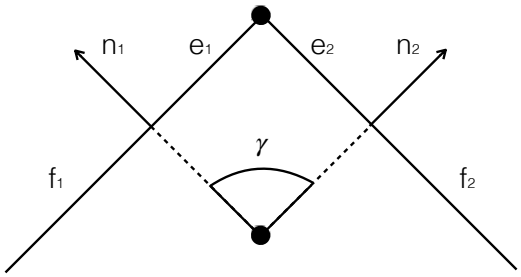
$$K = \frac{\sum m(e_i)}{\frac{1}{3}A_f}, \tag{3.14}$$

na qual e_i é a representação de uma aresta ligada ao nó, e $m(e_i)$ é uma função que retorna o ângulo γ entre os dois vetores normais das faces adjacentes em e_i, conforme a Figura 3.13.

O ângulo formado pelas arestas $m(e_i)$ pode formar três valores de ângulos possíveis para γ:

$$m(e_i) = \begin{cases} \gamma, & \text{se } e_i \text{ for convexo} \\ 0, & \text{se } e_i \text{ for plano} \\ -\gamma, & \text{se } e_i \text{ for côncavo.} \end{cases} \tag{3.15}$$

Figura 3.13 – Ângulo γ para um nó de uma curvatura discreta.



Fonte: Adaptado de Siqueira *et al.* (2014).

3.6.3 Estimativa de erro

A estimativa de erro é a etapa que guia e faz a medição de convergência da técnica proposta. Todo o processo adaptativo é uma rediscretização das curvas e dos *patches* em cada processo ou unidade de processamento. Conforme dito anteriormente, essa rediscretização é executada com base na estimativa de erro entre as curvaturas discreta e analítica em cada nó da malha num determinado processo.

Ela está divida em duas subetapas: a primeira trata da estimativa de erro local η_l, ou seja, a que está relacionada a cada nó da malha, e a segunda é a estimativa de erro global η_{gp}, que por sua vez, representa a média aritmética de todas as estimativas de erros locais numa determinada unidade de processamento.

3.6.3.1 Estimativa de erro local

O cálculo da estimativa de erro local é baseado na curvatura discreta relacionada com a curvatura analítica, que são representadas por K_d e K_a respectivamente, avaliadas em cada nó da malha em um determinado processo. Se o resultado do cálculo apresentar uma discordância de valores entre as curvaturas, faz-se necessário um refinamento ou desrefinamento local, usando como base um parâmetro de tamanho h que representa a diferença entre as duas curvaturas.

Alguns dos possíveis cenários relacionados ao erro local entre K_d e K_a estão ilustrados na Tabela 3.1, que podem ser:

Tabela 3.1 – Possíveis cenários da estimativa de erro local.

Curvaturas K_a e K_d	**Analítica**	**Tamanho**	**Procedimento**
$K_a \approx K_d \quad (K_a/K_d) \to 1$	$k_a \to 0$	$h_{novo} = h_{velho}.f$	desrefinar
	$k_a \not\to 0$	$h_{novo} = h_{velho}$	parar
$K_a \gg K_d$	$k_a \to 0$	$h_{novo} = h_{velho}/f$	refinar
	$k_a \not\to 0$	$h_{novo} = h_{velho}/f$	refinar
$K_a \ll K_d$	$k_a \to 0$	$h_{novo} = h_{velho}/f$	refinar
	$k_a \not\to 0$	$h_{novo} = h_{velho}/f$	refinar

Fonte: Adaptado de Siqueira *et al.* (2014).

- $K_a \approx K_d$, na qual os dois valores são aproximadamente iguais:
  - $K_a \to 0$, onde a curvatura analítica é próxima a zero. Necessita de um desrefinamento da malha ($h_{novo} = h_{velho}.f,\ f > 1$).
  - $K_a \not\to 0$, na qual K_a está disposta adequadamente na malha e nenhum tipo de ação faz-se necessária.
- $K_a \gg K_d$, na qual a curvatura analítica é consideravelmente maior do que a curvatura discreta:
  - $K_a \to 0$ e $K_a \not\to 0$,K_a próximo e distante de zero respectivamente. Nestes casos a malha não é suficientemente refinada para capturar a planaridade da superfície e deve ser refinada ($h_{novo} = h_{velho}/f$).
- $K_a \ll K_d$, em que a curvatura analítica é consideravelmente menor do que a curvatura discreta. O procedimento a ser adotado é análogo ao do item anterior.

O fator f usado na rediscretização é importante para o processo, pois a taxa de convergência é inversamente proporcional a ele. Nesse trabalho f é um fator empírico, que pode ser devidamente calibrado. A estimativa de erro é calculada para todos os pontos da malha em cada processo, e esta etapa é realizada de forma iterativa até o processo atingir uma convergência, a qual é determinada pela estimativa de erro global.

3.6.3.2 Estimativa de erro global

A estimativa de erro global e as rediscretizações no processo são baseadas nas estimativas de erros locais. Esta estimativa global é fundamental para guiar o processo adaptativo, no qual pretende-se atingir uma taxa de convergência previamente estipulada. Quando a qualidade da malha atingir essa taxa de convergência, a técnica proposta termina na unidade de processamento corrente. Uma medida de estimativa de erro global η_{gp} é

usada. Para obter essa medida, o seguinte cálculo é efetuado como mostra a Equação 3.16:

$$\eta_{gp} = \frac{\sum_{j=1}^{N_v} \eta_j}{N_v}, \tag{3.16}$$

onde N_v é o número de nós de toda a malha do processo corrente e η_j é o valor absoluto da diferença relativa entre as curvaturas analítica e discreta em um determinado nó, o qual é calculado seguindo a Equação 3.17:

$$\eta_j = \frac{|(K_a - K_d)|}{|K_a|}. \tag{3.17}$$

Quando $\eta_{gp} < \varepsilon$, onde ε é uma precisão desejada, isso indica que a estimativa de erro global no processo atingiu a convergência estipulada através do processo iterativo e a qualidade em questão chegou ao patamar desejado.

3.7 ADAPTAÇÃO DAS CURVAS

Esta etapa de adaptação consiste em rediscretizar as curvas que delimitam cada *patch* de forma independente. Isso faz com que a compatibilidade entre *patches* adjacentes seja garantida e haja uma regularização das fronteiras com as características geométricas próximas a essas curvas.

A adaptação de curvas tem como base o trabalho de Baehmann *et al.* (1987), no qual o procedimento para rediscretização de curvas delimitantes é proposto, utilizando uma árvore binária para armazenar e organizar as informações de discretização da curva, e além desta árvore binária, o critério de refinamento é guiado pelas curvaturas analítica e discreta. Portanto, a discretização de uma determinada curva C_i pode ser obtida através do cálculo das curvaturas analítica e discreta num dado nó da aproximação poligonal desta curva.

A discretização de uma curva é feita em três fases: na Subseção 3.7.1 é apresentada a inicialização da árvore binária; na Subseção 3.7.2, a rediscretização da curva C_i e, por último, na Subseção 3.7.3, a atualização de C_i baseada na árvore binária. O Algoritmo 3.1 detalha os passos para a discretização da curva. Essa discretização é baseada no trabalho de (Siqueira *et al.* (2014)) com pequenas adaptações e com a diferença que é executado em diferentes processos.

Algoritmo 3.1: Construção da árvore binária.

Entrada:

Inicialize a raiz da árvore binária com nó vazio.

As coordenadas mínima e máxima do nó raiz recebem $(0, 1)$.

Calcule o comprimento da curva, L_{curva}.

início

  para *cada segmento da lista de vértices iniciais,* S_k, $K \leftarrow 1$ **até** n_{seg} **faça**

    Calcule o tamanho do segmento no espaço 3D, C_{seg_k};

    Calcule K_a em C_{seg_k};

    Calcule K_d com a média das K_{ds} dos vértices extremos de S_k

    $K_d = \frac{1}{2}(^{k-1}K_d +^{k} K_d)$;

    Calcule h_{novo} de acordo com a Tabela 3.1;

    Calcule $h_{par} = \frac{h_{novo}}{L_{curva}}$, $h_{par} \in [0, 1]$;

    Determine o parâmetro correspondente a C_{seg_k}, u_k;

    Determine em que nó da árvore está u_k;

    enquanto *tamanho do segmento* $> h_{par}$ **faça**

      Subdivida o nó em dois filhos;

      Incremente a profundidade da árvore;

      Determine em que nó da árvore está u_k;

    fim

  fim

fim

Fonte: Adaptado de Siqueira *et al.* (2014).

3.7.1 Construção da árvore binária

Nesta fase, a árvore binária é inicializada para guiar a discretização da curva C_i que possui característica paramétrica, ou seja, as coordenadas que as delimitam estão entre o espaço paramétrico $[0, 1]$. Como é o inicio da fase, a árvore binária possui altura com valor zero.

3.7.2 Rediscretização da árvore binária

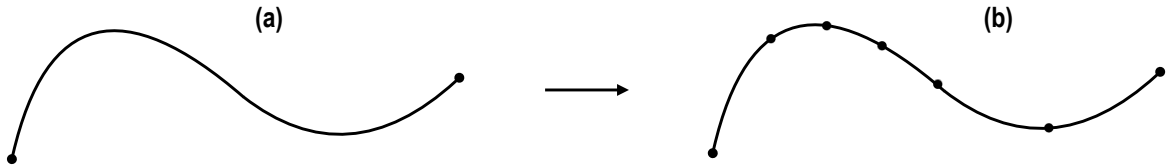
A discretização da curva C_i está fortemente ligada à geração da malha M_k em cada passo numa determinada unidade de processamento. Isso porque a etapa de adaptação de C_i armazena as informações do conjunto de nós do passo anterior até que uma nova malha seja gerada. Quando a rediscretização da curva C_i começa percorrendo os nós

da discretização do passo anterior, tem-se um segmento de curva formado a cada dois nós consecutivos e as folhas da árvore binária são responsáveis pelo armazenamento das informações de valores mínimo e máximo de cada segmento de C_i.

Toda esta fase ocorre no espaço paramétrico e, após a rediscretização percorrer todos os segmentos de C_i, tem-se uma nova árvore binária com informações do novo refinamento. Desta forma, a nova discretização da curva C_i tem como novos nós os segmentos de cada folha da árvore binária.

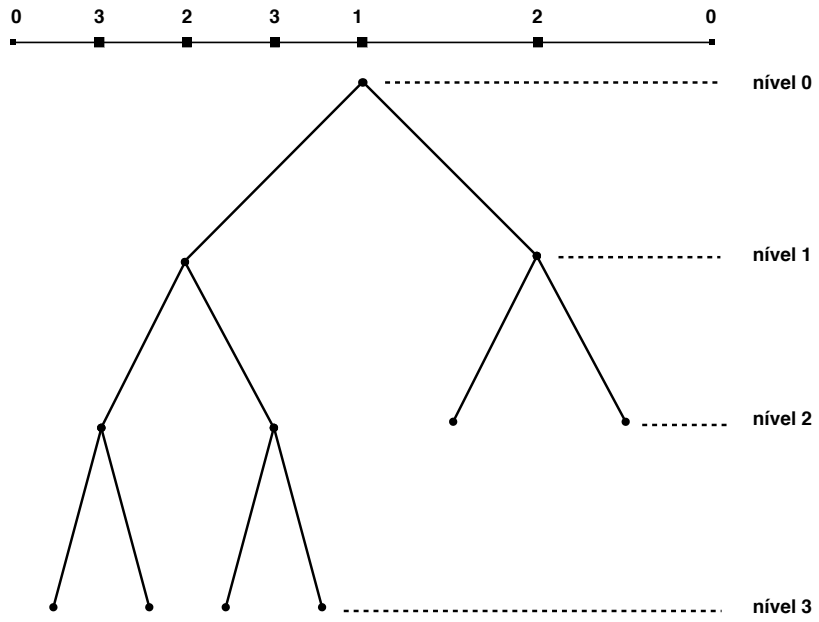
Na Figura 3.14 é mostrada a curva C_i com seu devido refinamento, em que a) mostra a curva sem refinamento e b) mostra essa curva devidamente refinada. A árvore binária correspondente a este refinamento pode ser vista na Figura 3.15 com seus níveis de refinamento e folhas.

Figura 3.14 – Refinamento da curva C_i.



Fonte: Adaptado de Siqueira *et al.* (2014).

Figura 3.15 – Árvore binária da curva C_i.



Fonte: Adaptado de Siqueira *et al.* (2009).

3.7.3 Atualização da curva com base na árvore binária

A atualização da curva C_i ocorre quando a árvore binária é redefinida e a nova discretização é incluída na curva. O Algoritmo 3.2 demonstra este processo. Logo depois, o conjunto de nós antigos é liberado e os novos nós representam a nova discretização de C_i de forma ordenada.

Algoritmo 3.2: Discretização da curva.

Entrada:

Discretização da curva.

início

  para *cada folha da árvore binária* **faça**

    Obtenha a coordenada paramétrica máxima;

    Calcule as coordenadas 3D correspondente;

    Inclua essas coordenadas na estrutura da curva;

  fim

fim

Fonte: Adaptado de Siqueira *et al.* (2009).

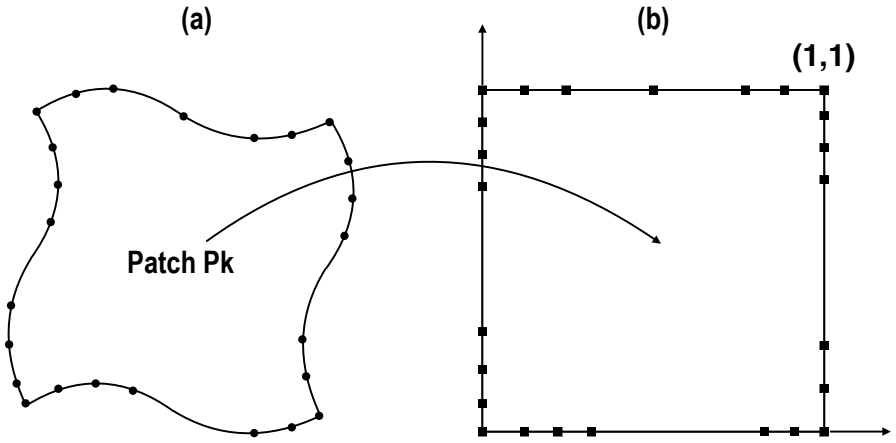
Desta forma, a etapa de adaptação das curvas dos *patches* numa unidade de processamento é sempre executada antes da adaptação do domínio, a fim de garantir compatibilidade entre *patches* adjacentes. Além disso, a malha gerada em diferentes unidades de processamento tem garantia de compatibilidade nas curvas compartilhadas.

3.8 ADAPTAÇÃO DO DOMÍNIO

A adaptação do domínio é iniciada nas partes internas do *patch*, com a geração da *quadtree* no espaço paramétrico. Esta característica da *quadtree* depende de um mapeamento do conjunto de nós do espaço tridimensional para o espaço bidimensional. Posteriormente, os nós gerados são remapeados para o espaço 3D. A Figura 3.16 ilustra esse mapeamento das curvas discretizadas que delimitam o *patch* P_k, onde a) representa P_k no espaço tridimensional e b) sua representação no espaço bidimensional.

A malha é então gerada na parte interna do domínio, através de elementos que são gerados nos nós internos da *quadtree*. Logo em seguida, a malha é gerada na zona de transição, entre os nós internos da *quadtree* e a fronteira do *patch*, através de uma combinação da técnica de avanço de fronteira com um critério de Delaunay e, por último, uma suavização Laplaciana é realizada com intuito de melhorar a malha gerada. Vale ressaltar que a técnica de avanço de fronteira é adaptada para considerar informações da *quadtree* gerada.

Figura 3.16 – Discretização da borda de um *patch*.



Fonte: Adaptado de Siqueira *et al.* (2009).

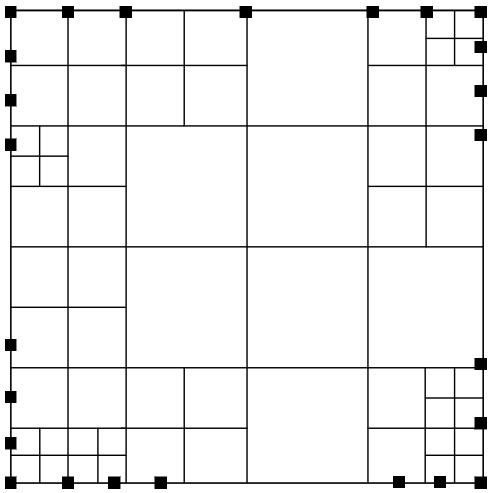
Portanto, a adaptação do domínio está divida em três fases: a primeira está relacionada à geração da malha no interior do *patch* utilizando a técnica de *quadtree*, ou seja, a malha é gerada em regiões que não fazem fronteira com as curvas delimitantes; a segunda consiste em gerar a malha nas regiões delimitantes (zona de transição) do *patch*, usando a técnica de avanço de fronteira e, finalmente, uma suavização da malha é gerada para cada *patch*.

3.8.1 Geração da malha no interior do *patch*

Utilizando o *patch* didático P_k para descrever a geração da malha no interior do domínio, usa-se a técnica de *quadtree* como geradora da malha desta etapa. A *quadtree* garante uma boa densidade entre as células geradas no interior e na fronteira como também uma boa transição entre regiões com graus de refinamento diferentes.

A geração da malha desta etapa inicia-se com a construção da células de *quadtree* no espaço paramétrico, como pode-se observar na Figura 3.17, e a descrição de sua construção pode ser vista com detalhes no Algoritmo 3.3.

Figura 3.17 – *Quadtree* inicial do *patch* P_k.



Fonte: Adaptado de Siqueira *et al.* (2009).

Algoritmo 3.3: Construção da *quadtree* iniciando com a discretização da borda.

Entrada:

Inicialize a árvore com nó raiz vazio.

início

  para *cada curva de borda,* C_i, $i \leftarrow 1$ **até** 4 **faça**

    Calcule o comprimento de arco de C_i no espaço 3D, L_i;

    para *cada segmento,* S_j, $j \leftarrow 1$ **até** n_{seg} **faça**

      Calcule o comprimento de arco de S_j no espaço 3D, L_{seg_j};

      Calcule o centro de S_j no espaço 3D, C_{seg_j};

      Encontre as coordenadas paramétricas de C_{seg_j}, (u, v);

      Encontre em que célula (u, v) está localizada;

      enquanto *tamanho da célula* $> (\frac{L_{seg_j}}{L_i})$ **faça**

        Subdivide a célula em quatro filhos;

        Determine em que célula (u, v) está localizado;

      fim

    fim

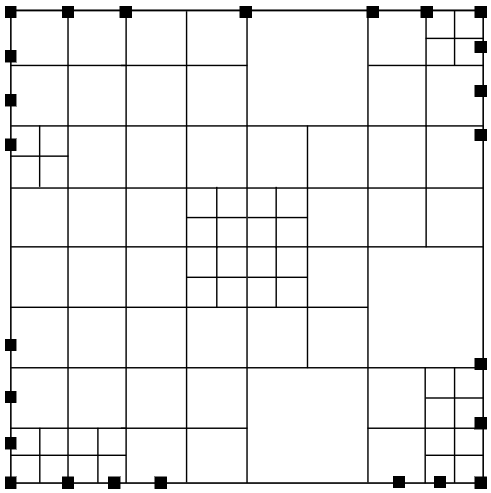
  fim

fim

Fonte: Adaptado de Siqueira *et al.* (2014).

Usando como base a discretização das curvas delimitantes de P_k (Figura 3.16), as células de *quadtree* é ajustada de acordo com as características geométricas da malha e os erros entre as curvaturas em cada um dos nós, como pode-se ver na Figura 3.18.

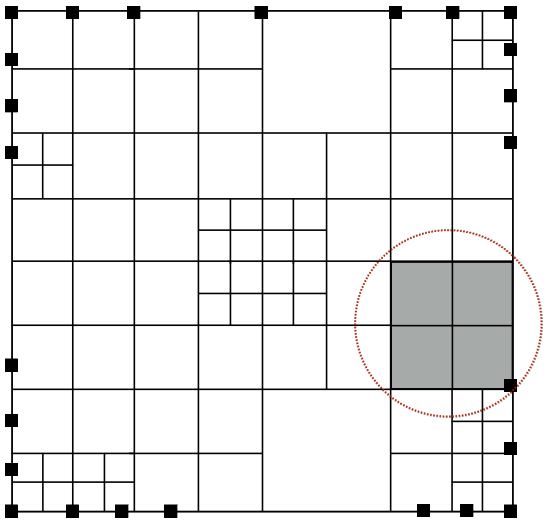
Figura 3.18 – *Quadtree* ajustada do *patch* P_k.



Fonte: Adaptado de Siqueira *et al.* (2009).

Na sequência, as células de *quadtree* é transformada em restrita para garantir que diferentes regiões da *quadtree* não tenham diferença entre os graus de refinamento maior que uma unidade, como podemos visualizar na Figura 3.19, onde a região demarcada mostra a melhoria exercida pela *quadtree* restrita. Nesta região, a alteração é aplicada para garantir uma boa transição entre as células adjacentes.

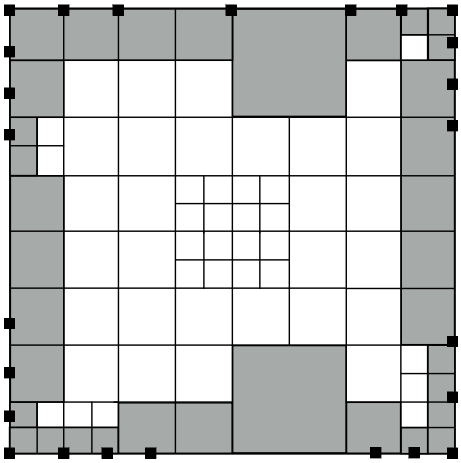
Figura 3.19 – *Quadtree* restrita do *patch* P_k.



Fonte: Adaptado de Siqueira *et al.* (2009).

Após a criação da *quadtree* do *patch* P_k, o próximo passo é identificar e classificar as células que pertencem à zona de transição e ao interior de P_k, conforme observa-se na Figura 3.20, em que as regiões sombreadas são as que pertencem ao tipo de transição e as demais pertencem ao interior do domínio.

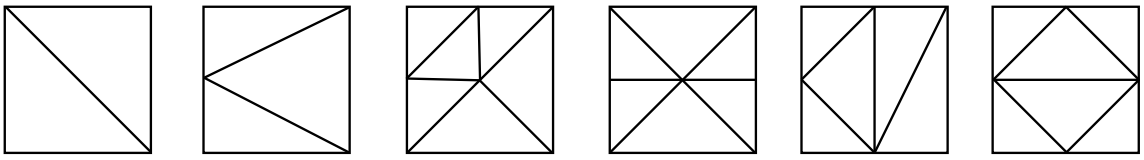
Figura 3.20 – Divisão da *Quadtree* do *patch* P_k em dois tipos: transição e interior.



Fonte: Adaptado de Siqueira *et al.* (2009).

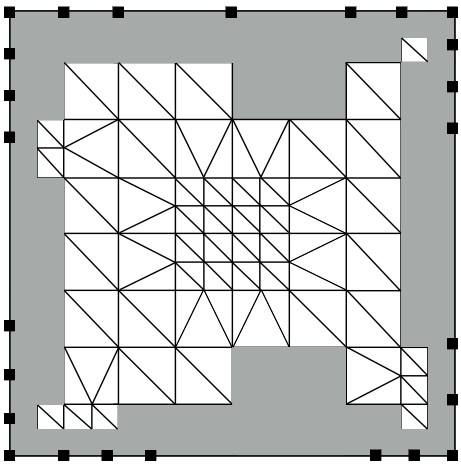
Na sequência, a malha no interior de P_k é gerada seguindo a proposta de Baehmann *et al.* (1987) através de padrões (Figura 3.21). Deste modo, após o procedimento de geração da malha no interior do *patch* P_k utilizando padrões, tem-se como resultado desta geração regiões do *patch* P_k do tipo interior com suas respectivas malhas no espaço bidimensional, conforme a Figura 3.22.

Figura 3.21 – Exemplo de padrões para geração da malha do interior.



Fonte: Adaptado de Siqueira *et al.* (2009).

Figura 3.22 – Malha gerada por padrões no interior de P_k.



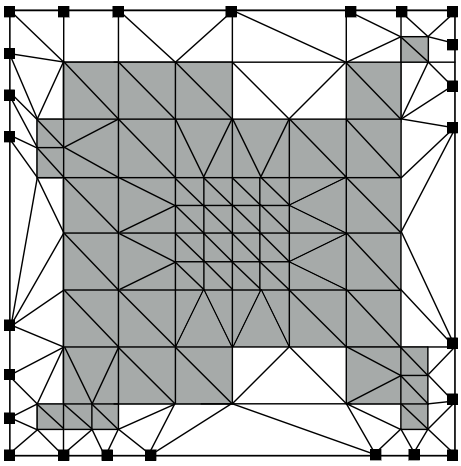
Fonte: Adaptado de Siqueira *et al.* (2009).

Após a conclusão desta etapa nas regiões de interior, o próximo passo é gerar malhas nas regiões de transição para a conclusão da etapa de geração da malha em P_k.

3.8.2 Geração da malha na zona de transição

A zona de transição é triangulada através da técnica de avanço de fronteira combinada com um critério de Delaunay. Esta etapa está descrita no Algoritmo 3.4 e a malha final gerada para o *patch* P_k pode ser vista na Figura 3.23, onde são expostas as células geradas por avanço de fronteira na zona de transição (células brancas) e as do tipo interior (células cinzas) por *quadtree* no *patch* P_k.

Figura 3.23 – Malha Final gerada para o *patch* P_k.



Fonte: Adaptado de Siqueira *et al.* (2009).

Algoritmo 3.4: Triangulação por avanço de fronteira.

Entrada:

A partir de todos os segmentos do contorno fornecidos, construir uma lista de arestas ativas.

início

  enquanto *a lista de arestas ativas não esteja fazia* **faça**

    Escolher uma aresta (base para um novo triangulo);

    para *cada aresta escolhida* **faça**

      Construir uma lista de células interiores adjacentes às células que contêm os vértices inicial e final dessa aresta base;

    fim

    se *não houver células* **então**

      A lista de células será formada pelas oito adjacências das células que contêm os vértices inicial e final da aresta base;

    fim

    Dentro da lista de células, buscar o vértice que forme o triângulo com o maior ângulo oposto à aresta base;

    Após adicionar um triângulo, a lista de arestas ativas deve ser atualizada;

  fim

fim

Fonte: Adaptado de Siqueira *et al.* (2014).

3.8.3 Suavização da malha gerada

Finalmente, após a conclusão do processo de geração da malha, uma suavização é aplicada. A suavização opera da seguinte forma: calcula-se novas coordenadas, no espaço paramétrico, para todos os nós da malha, excluindo aqueles que se encontram na fronteira.

A técnica utilizada é uma formulação geral do operador Laplaciano, que usa um fator que relaciona as medidas do espaço real e do espaço paramétrico, em que a média das coordenadas dos nós adjacentes ao nó que está sendo modificado é calculada, desta forma, evitando e corrigindo possíveis distorções, como mostrado na Equação 3.18:

$$X_0^{n+1} = X_0^n + \Phi \frac{\sum_{i=1}^{m} \omega_{i0}(X_i^n + X_0^n)}{\sum_{i=1}^{m} \omega_{i0}}, \tag{3.18}$$

na qual X_0 é o vetor do vértice 0, em que incidem os nós i (X_i), m é o número de vértices incidentes e ω_{i0} é um fator que relaciona a distância real e paramétrica entre os vértices i

e 0, e Φ é um fator de relaxação que deve estar entre 0 e 1. É importante destacar que este processo de suavização pode se repetir várias vezes, e logo após, os nós são mapeados do espaço paramétrico para o espaço tridimensional.

3.9 FINALIZAÇÃO E JUNÇÃO DA MALHA

Com o término da geração e suavização da malha em cada unidade de processamento, que, por sua vez, finaliza a malha gerada e envia para o processo mestre, este faz a junção, ou seja, o processo mestre trata todos os identificadores dos nós e elementos para que não haja possíveis duplicações de identificação, destas malhas incluindo a que foi gerada por ele mesmo. Assim, a técnica de geração de malhas adaptativas de superfície em paralelo é finalizada.

3.10 CONSIDERAÇÕES FINAIS

Neste capítulo, a técnica proposta foi explicada de forma detalhada, expondo e exemplificando como funcionam as etapas de geração de malhas em paralelo.

A técnica aqui proposta gera malhas triangulares e sua qualidade está ligada à relação entre as curvaturas discreta e analítica em cada nó da malha gerada durante todos os passos do procedimento de geração.

Um dos fatores que permitiu a divisão do trabalho em processos foi a independência entre os *patches*, resultando assim em uma baixa comunicação entre as unidades de processamento, fazendo com que as perdas com comunicação fossem irrelevantes no resultado final da geração.

4 EXEMPLO E RESULTADOS

4.1 INTRODUÇÃO

Neste capítulo são apresentados os resultados obtidos com a implementação da técnica paralela. Estes resultados englobam tanto aspectos de geometria computacional, como a qualidade da malha, quanto aspectos de computação de alto desempenho, como o tempo de execução e o *speed-up* da implementação.

O programa foi desenvolvido em C++ (The C++ Standards Committee, 2016), utilizando a biblioteca de passagem de mensagens MPI (*Message Passing Interface*)(MPI Forum, 2016) para o paralelismo com memória distribuída. Para a visualização das malhas geradas, foram utilizadas as bibliotecas OpenGL (com suas bibliotecas glu, glut e glew) para renderização e wxWidgets para interfaces. O compilador utilizado foi o g++ 4.8 com suporte para C++11 e a versão do MPI foi a MVAPICH2 1.9a2.

O gerador de malhas de superfície sequencial, desenvolvido pelos mesmos autores em um trabalho anterior (Siqueira *et al.* (2014)), foi adaptado para poder executar em paralelo, bem como otimizado em alguns pontos específicos para uma melhor performance.

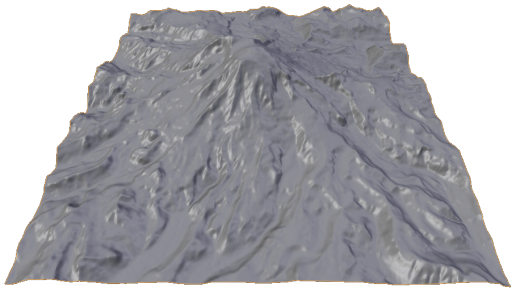
O *cluster* utilizado foi o do Centro Nacional de Processamento de Alto Desempenho (CENAPAD), instalado na Universidade Federal do Ceará (UFC). O *cluster* está equipado com 48 nós, em que cada nó possui dois processadores *Intel*® *Westmere*® X5650 EP e 24 GB de RAM, totalizando 576 núcleos de processamento e 1152 GB de RAM. Os testes foram feitos com até 8 processadores, utilizando o máximo de processadores por nó, não excedendo a memória.

4.2 MODELOS

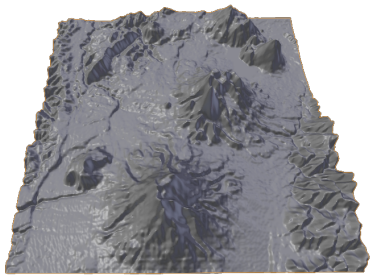
Os modelos considerados neste capítulo englobam geometrias de aplicação prática. A Figura 4.1 mostra modelos de terrenos montanhosos, com as mais variadas características geométricas. A escolha dos modelos está relacionada à quantidade de *patches* e às suas altas variações geométricas, assim como ao tamanho da malha gerada, para validar a técnica proposta.

Figura 4.1 – Modelos.

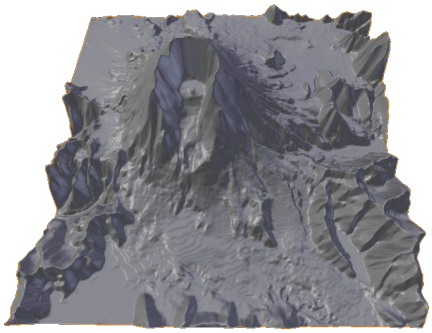
(a) Mont Rainier.



(b) Mont Ruapehu.



(c) Mont Sants Helens.



Fonte: Elaborada pelo autor.

A geração da malha de superfície está baseada nos seguintes critérios: o primeiro refere-se à quantidade de *patches*, que deve ser grande o suficiente para compensar o uso do paralelismo, e, por consequência, a malha gerada também possui esta característica de grandeza; o segundo é que o gerador de malhas de superfície sequencial deve conseguir gerar a malha do tamanho desejado, para que seja viável o cálculo do *speed-up*, isto é, a quantidade de memória utilizada na geração da malha sequencial não pode ser maior que a disponível em um nó do *cluster*; o terceiro e último critério está relacionado à utilização do número máximo de processadores em um nó do *cluster*, que visa tornar eficiente a utilização dos nós, devida à alta concorrência para sua utilização. Além disso, os modelos foram testados com 2, 4, 6 e 8 processos.

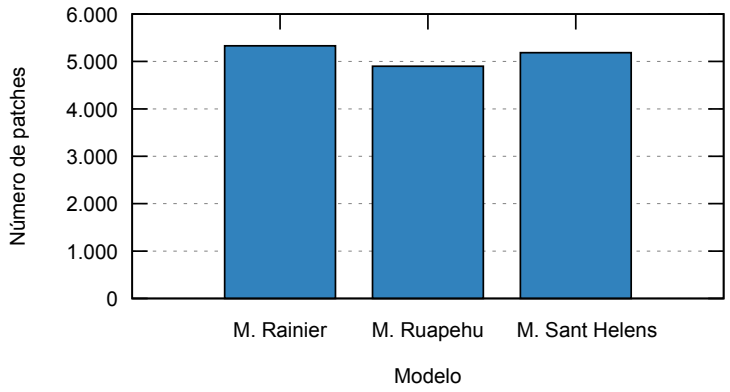
Vale ressaltar, a importância de eliminar amostras atípicas (*outliers*) de tempo de execução e, para isso, todas as configurações possíveis para um modelo foram executadas 5 vezes. Portanto, os resultados utilizados para cada modelo foram obtidos através da média das execuções feitas.

O gráfico da Figura 4.2 mostra a quantidade de *patches* dos modelos práticos. O modelo Mont Ruapehu possui 4.900 *patches*, enquanto que os outros dois modelos possuem uma quantidade de *patches* equivalentes: o Mont Rainier com 5.329 e o Mont Sant Helens com 5.184. Apesar desta equivalência, suas geometrias são distintas e, por consequência,

as malhas geradas também são.

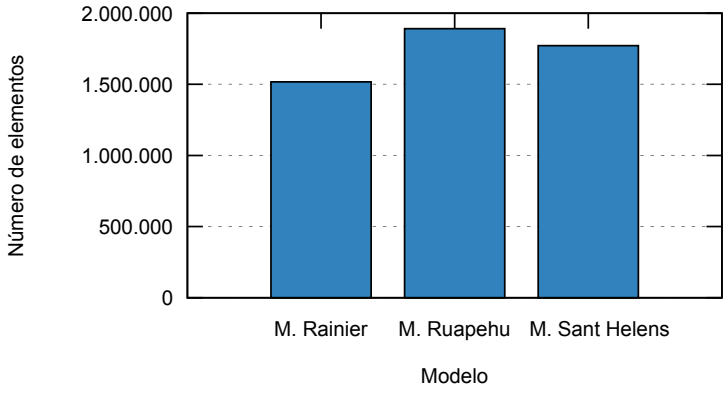
A quantidade de elementos gerados pela técnica sequencial pode ser vista no gráfico da Figura 4.3. Analisando os modelos, o Mont Rainier gerou uma malha com aproximadamente $1,5$ milhões de elementos, o Mont Raupehu $1,8$ milhões, e por último, o Mont Sant Helens gerou $1,7$ milhões de elementos.

Figura 4.2 – Número de *patches*.



Fonte: Elaborada pelo autor.

Figura 4.3 – Número de elementos gerados sequencialmente.



Fonte: Elaborada pelo autor.

4.3 ESTIMATIVA E BALANCEAMENTO DE CARGA

A estimativa de carga é calculada apenas uma única vez durante o processo de geração da malha, de forma sequencial, conforme visto na Seção 3.3. A divisão da carga

para cada processo é baseada na soma de todas as estimativas calculadas para cada *patch* que pertence a ele, como podemos ver na Equação 4.1:

$$E_{cpr} = \sum_{i=1}^{pt} E_{ci}, \tag{4.1}$$

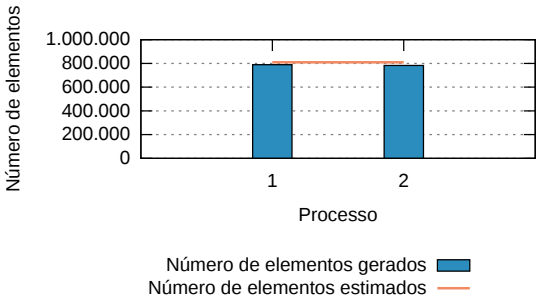
na qual E_{cpr} representa a estimava de carga para cada processo, pt é o número de *patches* em cada processo e E_{ci} é o valor estimado da carga para cada *patch* alocado no processo.

Vale reforçar que a estimativa proposta não tem a intenção de calcular a quantidade exata de elementos gerados, mas sim estimar de forma proporcional a quantidade total. Com isso, o principal objetivo desta estimativa é obter uma carga balanceada entre os processos, ou seja, a quantidade total de elementos gerados deve estar balanceada nos processos.

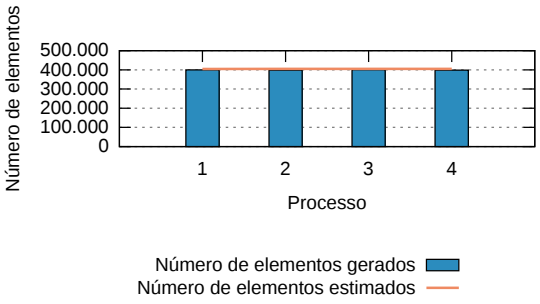
O balanceamento de carga considerado neste trabalho leva em conta o número de elementos gerados em cada processo (Seção 3.4). Por esse motivo, a estimativa e o balanceamento de carga serão analisados juntos. Os gráficos das Figuras 4.4, 4.5 e 4.6 mostram a estimativa e o balanceamento para os três modelos, executados com 2, 4, 6 e 8 processos.

Figura 4.4 – Estimativa e balanceamento do modelo Mont Rainier.

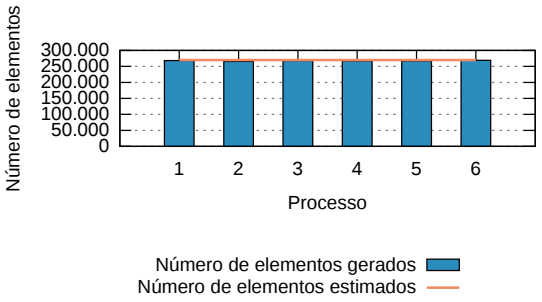
(a) 2 processos.



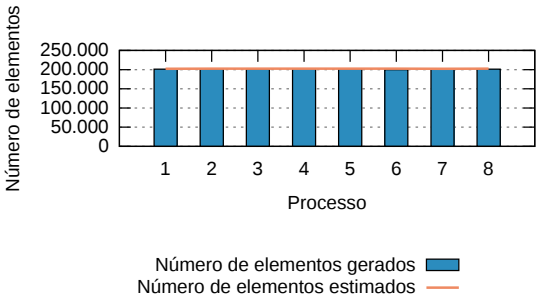
(b) 4 processos.



(c) 6 processos.



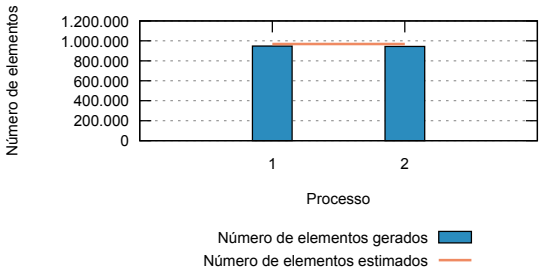
(d) 8 processos.



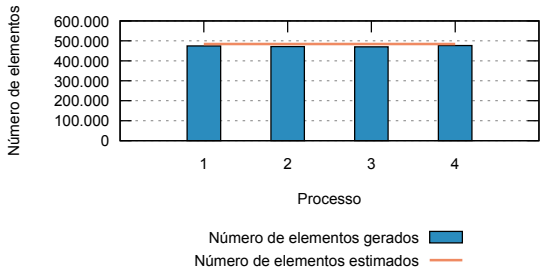
Fonte: Elaborada pelo autor.

Figura 4.5 – Estimativa e balanceamento do modelo Mont Ruapehu.

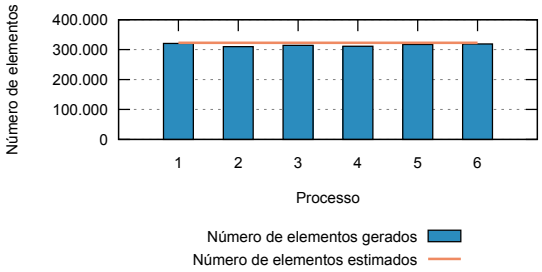
(a) 2 processos.



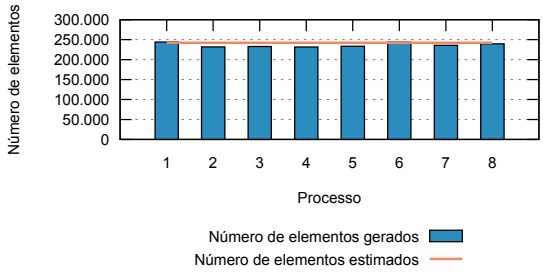
(b) 4 processos.



(c) 6 processos.



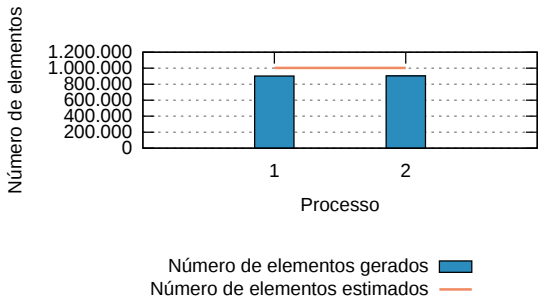
(d) 8 processos.



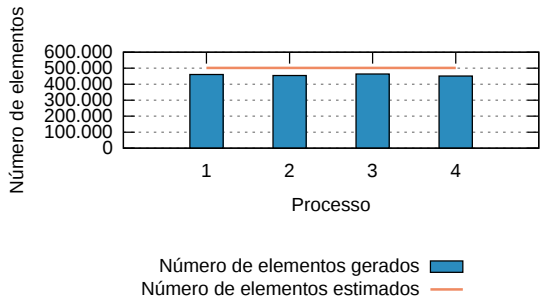
Fonte: Elaborada pelo autor.

Figura 4.6 – Estimativa e balanceamento do modelo Mont Sant Helens.

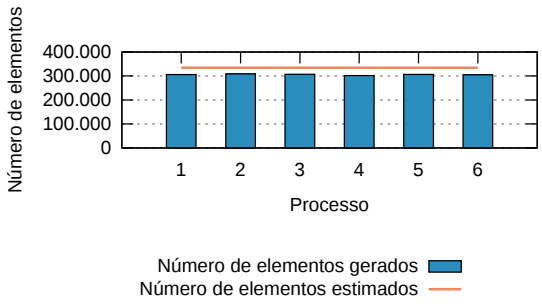
(a) 2 processos.



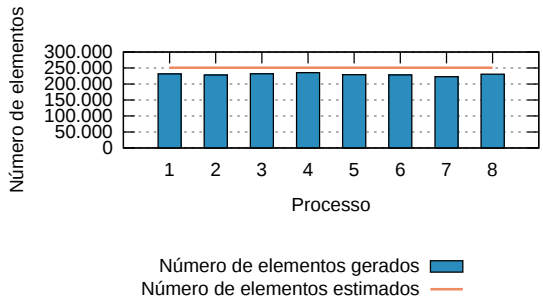
(b) 4 processos.



(c) 6 processos.



(d) 8 processos.



Fonte: Elaborada pelo autor.

Analisando os resultados obtidos, observa-se que a estimativa manteve-se acurada para os dois primeiros modelos, enquanto que para o terceiro foi um pouco acima da quantidade de elementos gerados. Entretanto, o balanceamento de carga mostrou uma boa distribuição para todos os modelos, atingindo, portanto, o objetivo principal da estimativa e resultando no balanceamento equilibrado desejado.

Os valores de carga estimadas para cada modelo com 2, 4, 6 e 8 processos podem ser vistos na Figura 4.7. As tabelas da Figura 4.7 mostram um equilíbrio entre os valores da carga estimada para cada modelo.

Figura 4.7 – Valores de carga estimada para cada processo.

(a) Mont Rainier.

Número de elementos estimados em cada processo	Número de processos			
	2	4	6	8
P1	809.401	404.615	269.904	202.233
P2	809.604	404.732	269.843	202.467
P3	-	404.866	269.850	202.352
P4	-	404.793	269.834	202.495
P5	-	-	269.817	202.386
P6	-	-	269.915	202.361
P7	-	-	-	202.322
P8	-	-	-	202.389

(b) Mont Ruapehu.

Número de elementos estimados em cada processo	Número de processos			
	2	4	6	8
P1	968.797	484.399	322.933	242.199
P2	968.021	484.034	320.125	241.983
P3	-	484.871	322.007	242.348
P4	-	483.901	321.971	241.908
P5	-	-	322.325	243.431
P6	-	-	321.170	242.564
P7	-	-	-	243.124
P8	-	-	-	241.789

(c) Mont Sant Helens.

Número de elementos estimados em cada processo	Número de processos			
	2	4	6	8
P1	1.003.760	501.918	334.615	251.120
P2	1.004.010	502.069	334.766	251.078
P3	-	322.007	334.614	250.860
P4	-	501.968	334.524	250.908
P5	-	-	334.586	250.893
P6	-	-	334.662	250.889
P7	-	-	-	251.109
P8	-	-	-	250.911

Fonte: Elaborada pelo autor.

4.4 TEMPO DE EXECUÇÃO E *SPEED-UP*

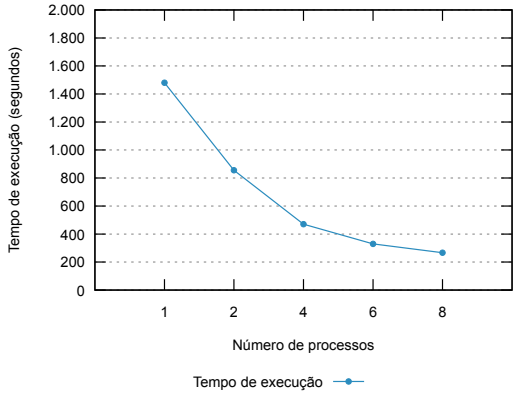
Os gráficos da Figura 4.8 mostram o tempo de execução dos modelos. O tempo de execução sequencial nos três modelos não tem diferenças significativas, embora o modelo onde a execução mais demorou foi o Mont Ruapehu, que obviamente foi o que gerou a maior malha. O modelo Mont Rainier foi o que executou em menor tempo, consequentemente gerando a menor malha.

Pelos gráficos de *speed-up*, da Figura 4.9, é possível observar um padrão nos três modelos, onde o *speed-up* foi sempre crescente, ou seja, sempre que o número de processos aumentou, o tempo de execução diminuiu, até mesmo no pior caso.

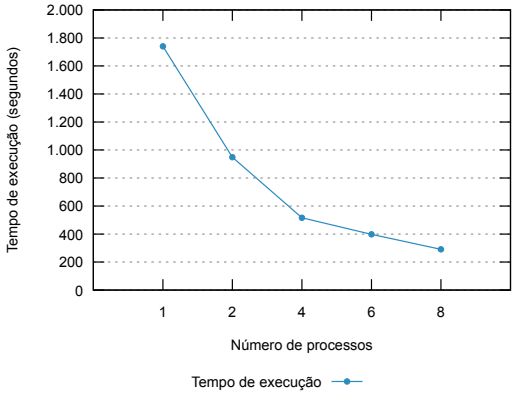
Uma implementação paralela ideal teria um *speed-up* linear, isto é, n processos a tornam n vezes mais rápida. Na prática, isso é difícil de acontecer, devido às inevitáveis porções sequenciais presentes no algoritmo paralelo, além do tempo de comunicação entre os processos, comunicação essa que não existe no algoritmo sequencial.

Figura 4.8 – Tempo de execução dos modelos.

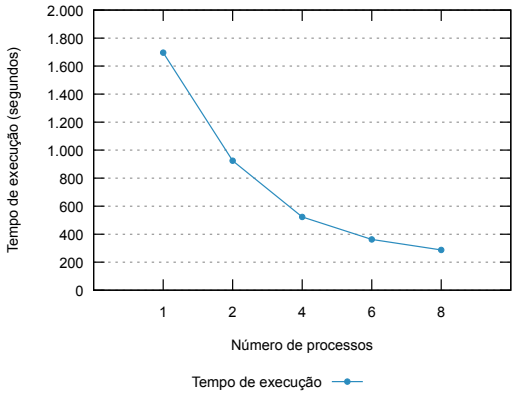
(a) Mont Rainier.



(b) Mont Ruapehu.



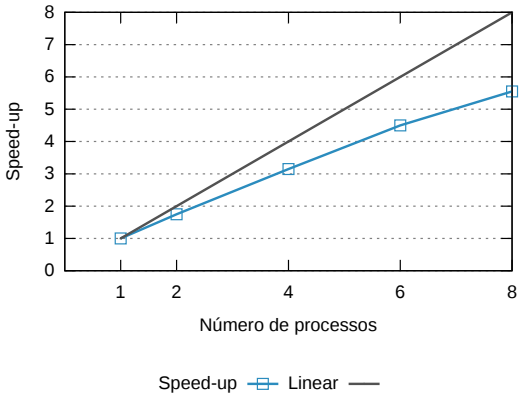
(c) Mont Sant Helens.



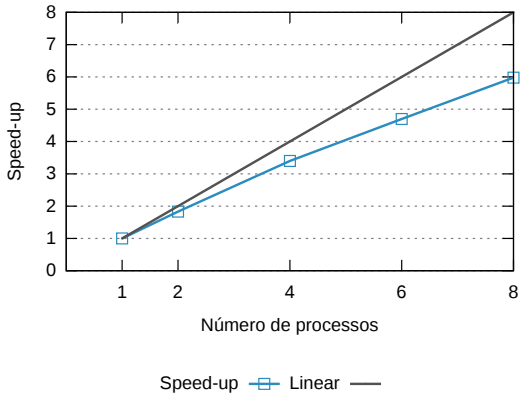
Fonte: Elaborada pelo autor.

Figura 4.9 – *Speed-up* dos modelos.

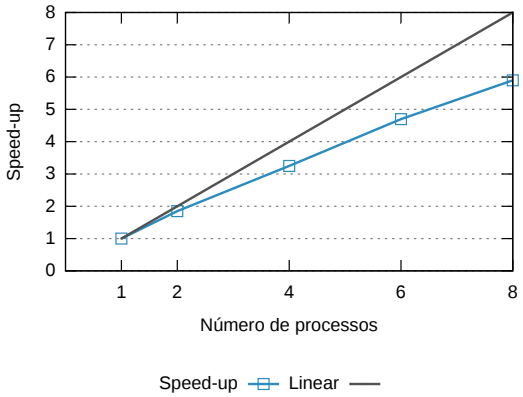
(a) Mont Rainier.



(b) Mont Ruapehu.



(c) Mont Sant Helens.



Fonte: Elaborada pelo autor.

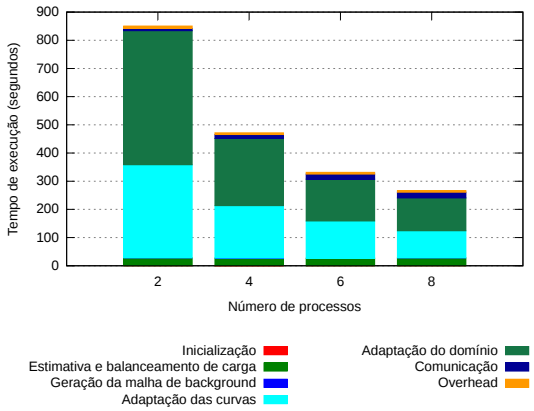
Avaliando os tempos de execução para os três modelos, executados com 8 processos, temos o seguinte resultado: no modelo Mont Rainier a redução foi de aproximadamente 82%; para os outros dois a redução obtida foi de aproximadamente 84%. Portanto, os tempos de execução foram reduzidos significativamente para os três modelos, satisfazendo assim, um dos requisitos proposto pelo trabalho. Além disso, bons resultados de *speed-up* foram obtidos para os três modelos, alcançando, desta forma, uma boa escalabilidade, que também é um requisito proposto.

4.4.1 Detalhamento do tempo de execução

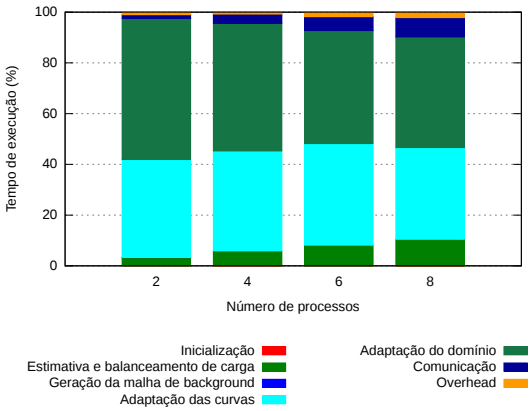
As duas colunas da Figura 4.10 espõem o detalhamento do tempo de execução do processo mestre, em termos absolutos e relativos, respectivamente, para os modelos apresentados neste capítulo.

Figura 4.10 – Detalhamento do tempo de execução absoluto (coluna da esquerda) e em porcentagem (coluna da direita).

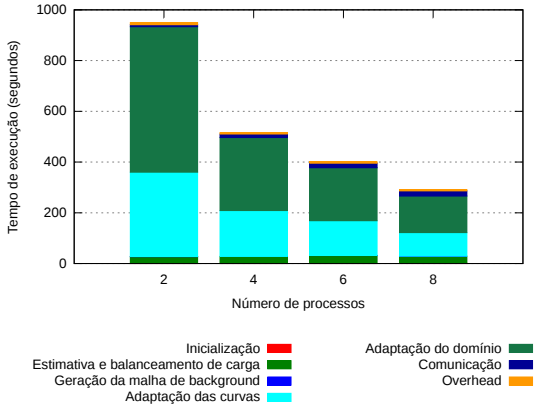
(a) Tempo absoluto do Mont Rainier.



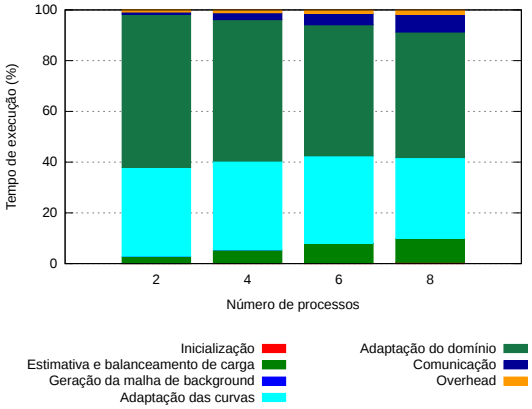
(b) Tempo em porcentagem do Mont Rainier.



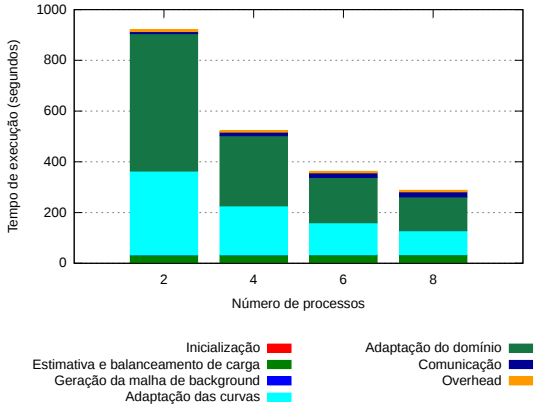
(c) Tempo absoluto do Mont Ruapehu.



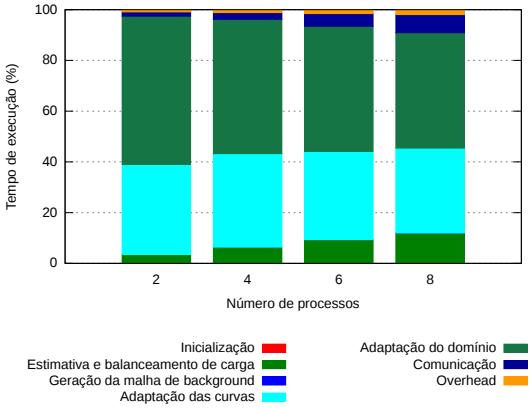
(d) Tempo em porcentagem do Mont Ruapehu.



(e) Tempo absoluto do Mont Sant Helens.



(f) Tempo em porcentagem do Mont Sant Helens.



Fonte: Elaborada pelo autor.

Os gráficos mostram que a inicialização e geração da malha de *background* tiveram porções de tempo muito pequenas e, por esse motivo, não apareceram nos gráficos dos três modelos. A estimativa e o balanceamento, por sua vez, podem ser fatores limitantes para o *speed-up*, já que são executados de forma sequencial e, por isso, a paralelização destas porções do algoritmo trariam melhorias para a estratégia. Em relação às fases de adaptação das curvas e domínio, são as que mais demandam tempo, como esperado, devido ao alto grau de computação para estas fases.

4.5 QUALIDADE DA MALHA

A qualidade da malha de superfície gerada por este trabalho é avaliada de duas formas: a primeira avalia o modelo levando-se em conta o erro entre as curvaturas analítica e discreta do modelo (Subseção 4.5.1) e a segunda avalia a qualidade dos elementos gerados através de uma métrica que utiliza-se da relação entre o círculo circunscrito e inscrito de cada elemento do modelo (Subseção 4.5.2).

O principal propósito desta técnica proposta não é, necessariamente, gerar uma malha de superfície de boa qualidade, mas sim gerar uma malha com qualidade próxima à malha gerada de maneira sequencial. Obviamente se a malha sequencial for de boa qualidade, a malha paralela também deve ser.

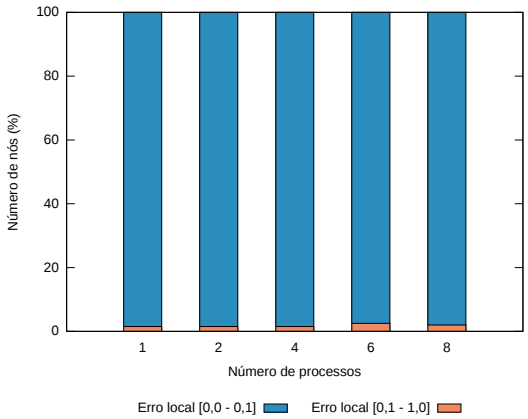
4.5.1 Qualidade do modelo

A primeira avaliação de qualidade da malha de superfície gerada por este trabalho se dá através do erro entre a curvatura discreta e a curvatura analítica, que é calculado para cada nó pertencente à malha. A média global entre todos esses erros de curvaturas resultam na medida de qualidade da malha, como já foi visto na Seção 3.6. Portanto, uma análise de forma percentual será mostrada nos gráficos da Figura 4.11, com a finalidade de expor a qualidade da malha gerada, tanto sequencialmente, como em paralelo.

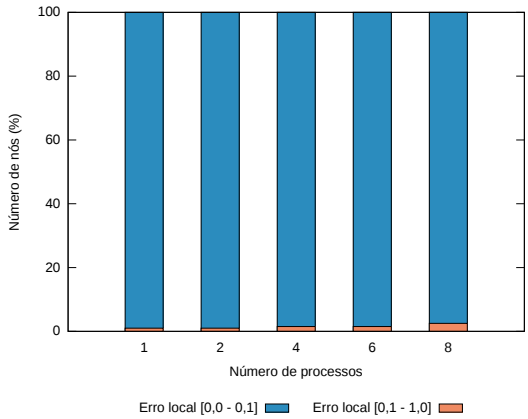
Analisando os gráficos da Figura 4.11, tem-se para os três modelos executados em paralelo malhas que mantiveram praticamente a qualidade da sequencial, onde os nós que estão entre $[0,0-0,1]$ são de melhor qualidade, ou seja, como o valor da qualidade de cada nó é formado pela diferença entre os erros analítico e discreto, quanto menor a diferença entre os dois, melhor a qualidade do nó. Nos modelos, a grande maioria dos nós está com erro entre $[0,0-0,1]$, em média 99% e os demais estão com erro entre $[0,1-1,0]$. Deste modo, as malhas geradas em paralelo, além de manter uma diferença mínima para a malha sequencial, também obtiveram uma excelente qualidade, confirmando assim uma boa convergência.

Figura 4.11 – Qualidade da malha gerada.

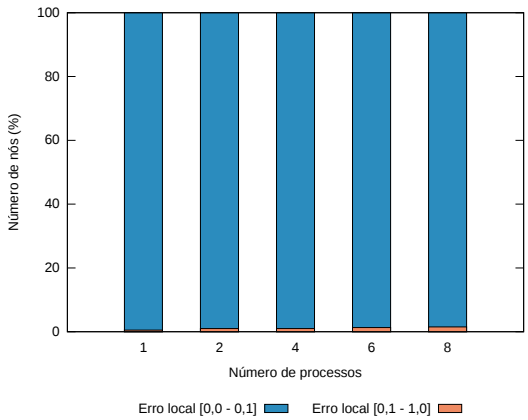
(a) Mont Rainier.



(b) Mont Ruapehu.



(c) Mont Sant Helens.



Fonte: Elaborada pelo autor.

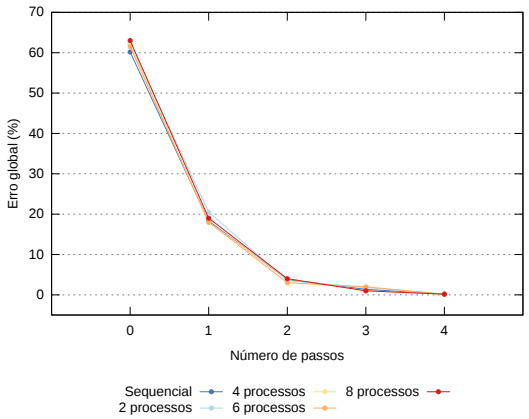
Os gráficos da Figura 4.12 mostram a convergência do erro global para os três modelos, executados com $1, 2, 4, 6$ e 8 processos. Tem-se no eixo X (horizontal) o número de passos da estratégia, em que o primeiro passo (passo 0) corresponde ao erro inicial, que, por sua vez é formado pela cálculo do erro na malha de *background* e os demais erros são formados pelo mesmo cálculo, mas desta vez, na malha correspondente a cada passo da estratégia. No eixo Y (vertical), tem-se o erro global em percentual para todos os passos.

Para o modelo Mont Rainier, o erro global cai de $60,1\%$ para $0,15\%$, para o Mont Ruapehu a convergência caiu de $45,7\%$ para $0,26\%$, e por último, o erro global do Mont Sant Helens passou de $55,5\%$ para $0,12\%$. Esses resultados foram atingidos em 4 passos para os três modelos, mas esse número pode sofrer variações, isso porque a convergência da técnica é guiada pela taxa de erro global e não pelo número de passos. Portanto, com a configuração mostrada acima os modelos atingem uma boa taxa de convergência.

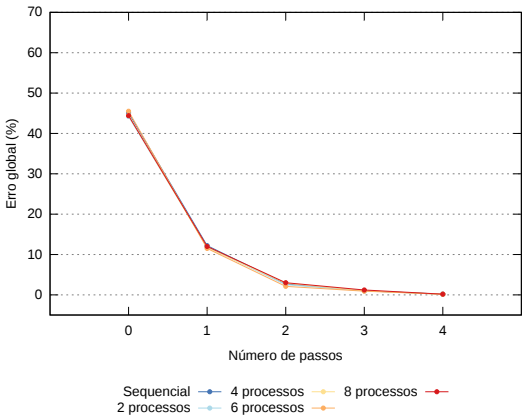
Os resultados obtidos para os três modelos mostram que a estratégia conseguiu manter a convergência obtida pela estratégia sequencial, ou seja, a técnica paralela obteve resultados bastante próximos dos obtidos pela técnica sequencial, atingindo assim, através

Figura 4.12 – Convergência do erro global.

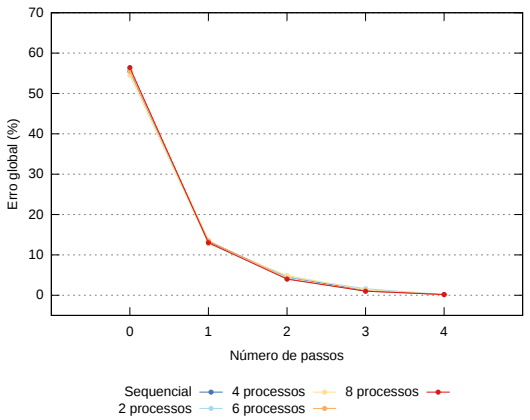
(a) Mont Rainier.



(b) Mont Ruapehu.



(c) Mont Sant Helens.

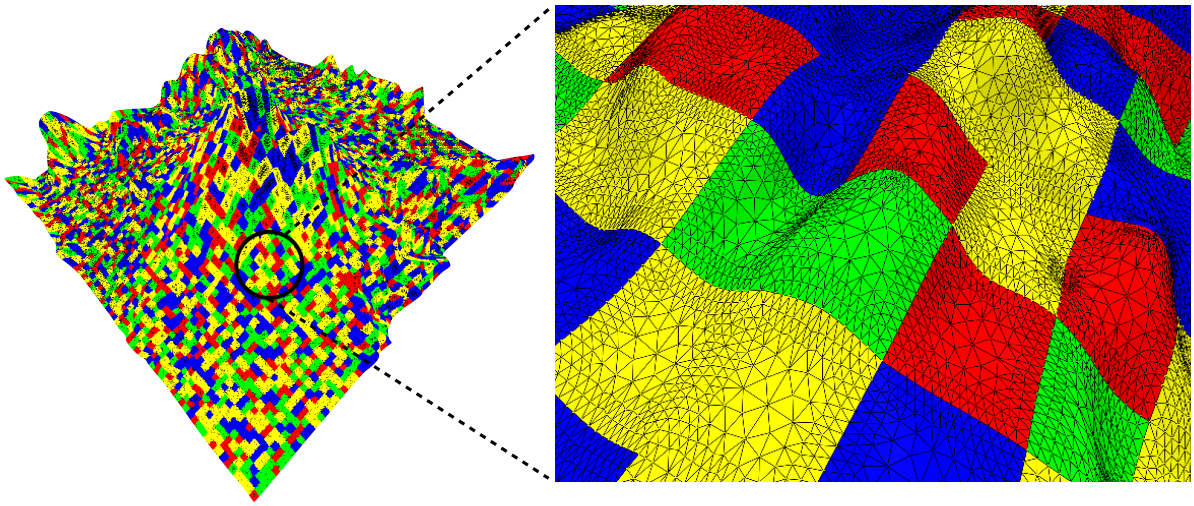


Fonte: Elaborada pelo autor.

da medição de qualidade entre os erros, uma malha discreta fiel à malha analítica.

A Figura 4.13 apresenta a ampliação de uma região do modelo Mont Sant Helens. A imagem do lado esquerdo representa o modelo executado com 4 processos, em que cada cor na superfície representa um processo. Na imagem do lado direito é dado destaque a uma pequena região, mostrando a malha gerada e a transição entre os *patches* e regiões do modelo. Portanto, a malha gerada mostrou boa transição entre regiões com diferentes graus de refinamento.

Figura 4.13 – Detalhe de uma região no modelo Mont Sant Helens.



Fonte: Elaborada pelo autor.

4.5.2 Qualidade dos elementos

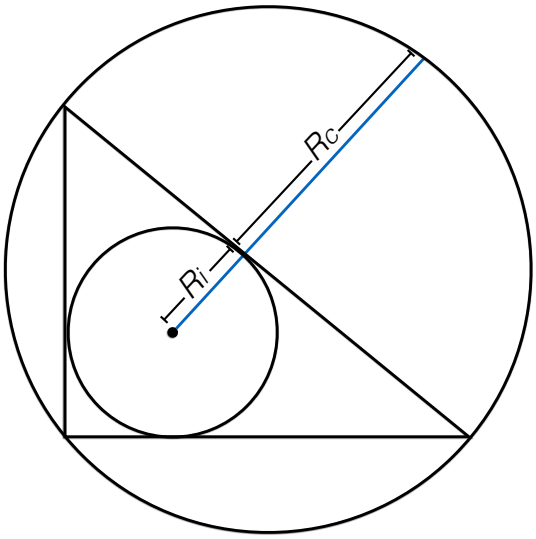
A segunda avaliação de qualidade da malha de superfície é através da métrica entre o círculo circunscrito e o inscrito em cada elemento triangular do modelo (Figura 4.14), definido pela Equação 4.2:

$$\alpha = \frac{2R_i}{R_c}, \tag{4.2}$$

onde R_i e R_c são os raios dos círculos inscritos e circunscritos, respectivamente (Teixeira (2014)). Esta métrica α tem valor $1,0$ para um triângulo equilátero. Quanto pior a qualidade do elemento, mais próximo de $0,0$ é o valor de α. Pode-se dizer que os elementos com $\alpha \leq 0,1$ são de péssima qualidade e que os elementos com $\alpha \geq 0,6$ são de boa qualidade.

Os gráficos da Figura 4.15 mostram a qualidade dos elementos gerados para cada modelo. Em a) cada barra indica a quantidade de elementos em cada intervalo de qualidade, para cada malha gerada, variando-se o número de processos. Os demais modelos são considerados de forma análoga.

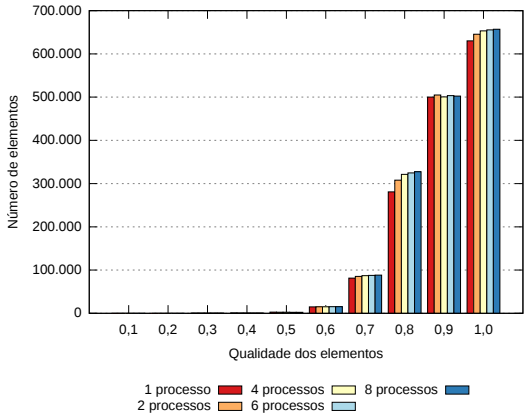
Figura 4.14 – Exemplo de um círculo circunscrito e inscrito em um triângulo.



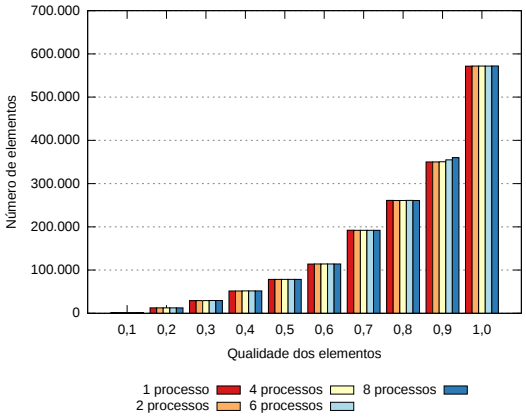
Fonte: Elaborada pelo autor.

Figura 4.15 – Qualidade dos elementos gerados.

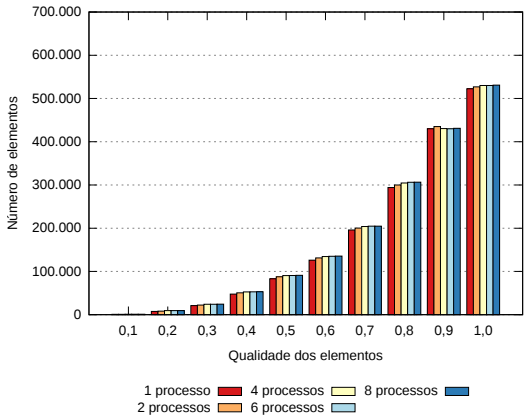
(a) Mont Rainier.



(b) Mont Ruapehu.



(c) Mont Sant Helens.



Fonte: Elaborada pelo autor.

Analisando os gráficos pode-se ver que as malhas geradas em paralelo, para cada um dos modelos, seguem a mesma tendência de qualidade da malha gerada sequencialmente. Como esta malha foi de boa qualidade, ou seja, a grande maioria dos elementos tem qualidade $0,6$ ou mais, as malhas geradas em paralelo também têm boa qualidade. Para os três modelos práticos foram obtidos bons resultados e a porcentagem de elementos que ficaram com qualidade igual ou superior a $0,6$ no modelo Mont Rainier foi de $96,1\%$, no Mont Ruapehu foi aproximadamente de $87,5\%$ e no Mont Sant Helens foi próximo de $91,7\%$. Portanto, para todos os modelos tem-se um alto número de elementos que podem ser considerados de boa qualidade.

4.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados alcançados pela execução da técnica paralela para três modelos. A estimativa de carga proposta atingiu seu principal objetivo, que consiste em um bom balanceamento de carga. Os resultados obtidos pela estimativa e balanceamento de carga foram fundamentais para alcançar um bom tempo de execução e *speed-up*.

As malhas geradas em paralelo mantiveram a qualidade da malha gerada pela técnica sequencial e, como a técnica sequencial produziu uma malha de excelente qualidade, obviamente a técnica paralela também conseguiu.

O próximo capítulo apresenta as conclusões referentes a este trabalho, destacando a suas principais contribuições, identificando em que pontos podem ser feitas melhorias e descrevendo como essas melhorias podem ser feitas.

5 CONCLUSÃO

5.1 PRINCIPAIS CONTRIBUIÇÕES

Este trabalho apresentou uma estratégia hierárquica de geração de malhas de superfícies paramétricas em paralelo com controle de curvatura para computadores com memória distribuída. A estratégia proposta mostrou-se capaz de refinar e desrefinar regiões da malha, garantindo uma boa transição entre essas regiões. Ela assegura, também, a compatibilidade entre regiões adjacentes, uma vez que a rediscretização das curvas acontece de forma independente do domínio.

A presente estratégia funciona para qualquer tipo de superfície, desde que seja paramétrica. Ela se utiliza da contribuição das medidas dos erros locais para garantir uma boa qualidade global. Os modelos mostrados neste trabalho demonstraram que a estratégia adaptativa em paralelo converge para uma malha de boa qualidade mesmo que a malha inicial seja extremamente não-refinada. Além da geração de malhas para modelos de terrenos, a estratégia pode ser usada para qualquer outro modelo que possa ser construído por superfícies paramétricas como modelos CAD, esculturas e imagem médicas.

As características do método paralelo proposto são também importantes para aplicações de computação gráfica e realidade virtual, visto que conseguiram uma boa distribuição dos elementos na malha, garantindo assim, uma boa renderização, devido à maior concentração de elementos em regiões com maiores curvaturas e redução na densidade em regiões com curvaturas menores. Isso pode ser difícil de alcançar utilizando os modeladores geométricos disponíveis, podendo ser crucial para aplicações em tempo real. A técnica também pode ser utilizada para problemas onde os níveis de detalhes (*level of details*) são frequentemente aplicados. Essa boa distribuição dos elementos da malha é também fundamental em problemas de engenharia, em especial os que usam métodos numéricos, como o Método dos Elementos Finitos, onde a qualidade dos elementos é crucial.

Finalmente, os ganhos obtidos com a paralelização sobre a técnica sequencial proposta por Siqueira *et al.* (2014) foram mostrado através de modelos práticos com a redução nos tempos de execução e nos bons números de *speed-up*. Mesmo a estratégia possuindo porções sequenciais no algoritmo paralelo, que por ventura podem ser paralelizadas, ela mostrou-se robusta e eficaz.

5.2 TRABALHOS FUTUROS

Alguns pontos podem ser melhorados neste trabalho: o primeiro está relacionado à estratégia, a qual poderia ser adaptada para ser aplicada em modelos *non-manifolds*, tornando assim mais ampla a sua aplicação. A estratégia paralela poderia também utilizar outros operadores de curvaturas, tornando-a mais abrangente.

O segundo refere-se ao fato que, durante a execução da técnica proposta, várias vezes são executados mapeamentos do espaço paramétrico para o espaço tridimensional e desse para o paramétrico, dos nós da malha, podendo causar distorções nos elementos gerados. Portanto, a aplicação de uma técnica adaptativa que não utiliza este mapeamento poderia ser uma opção interessante.

A terceira melhoria está relacionada à estimativa e balanceamento de carga, onde suas execuções são feitas de forma sequencial, quando poderiam ser executados de forma paralela, com a finalidade de obter melhores ganhos, no que diz respeito a tempo de execução e *speed-up*.

Outros dois pontos a serem explorados são: a utilização de memória compartilhada em conjunto com a distribuída, porque existem partes no algoritmo que podem executadas usando memória compartilhada; a utilização somente de memória compartilhada para malhas menores, por conta da inexistência de comunicação entre processos, pois acredita-se que o uso de memória compartilhada pura pode se comportar melhor que a memória distribuída nesses casos. Entretanto, podem surgir gargalos, como por exemplo, o possível aumento na intensidade de alocação de memória.

No que diz respeito a modelagem de superfícies paramétricas, existe certa escassez de modeladores para tal finalidade. Deste modo, para modelar superfícies mais complexas com grande variedade de curvaturas, faz-se necessário aprofundamento na modelagem de superfícies com essas características, com o propósito de obter um modelador compatível com superfícies paramétricas que respeite suas características de continuidade geométrica e paramétrica.

Por último, testes podem-se ser realizados com o aumento no de processadores, na qual uma análise mais detalhada do algoritmo pode ser feita, afim de, aferir sua eficiência e desempenho.

REFERÊNCIAS

BAEHMANN, P. L.; WITTCHEN, S. L.; SHEPHARD, M. S.; GRICE, K. R.; YERRY, M. A. Robust, geometrically based, automatic two-dimensional mesh generation. **International Journal for Numerical Methods in Engineering**, Wiley Online Library, v. 24, n. 6, p. 1043–1078, 1987. Citado 2 vezes nas páginas 42 e 49.

CAMPEN, M.; BOMMES, D.; KOBBELT, L. Dual loops meshing: quality quad layouts on manifolds. **ACM Transactions on Graphics (TOG)**, ACM, v. 31, n. 4, p. 110, 2012. Citado na página 16.

COZIN, C. **Um estudo da computação de alto desempenho aplicada à simulação do escoamento líquido-gás em golfadas em tubulações**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, 2008. Citado na página 18.

DILL, J. C. An application of color graphics to the display of surface curvature. In: ACM. **ACM SIGGRAPH Computer Graphics**. [S.l.], 1981. v. 15, n. 3, p. 153–161. Citado na página 38.

FREITAS, M. O.; WAWRZYNEK, P. A.; CAVALCANTE-NETO, J. B.; VIDAL, C. A.; CARTER, B. J.; MARTHA, L. F.; INGRAFFEA, A. R. Parallel generation of meshes with cracks using binary spacial decomposition. **Engineering with Computers**, p. 1–20, mar 2016. Citado na página 16.

HJELLE, Ø.; DÆHLEN, M. **Triangulations and applications**. [S.l.]: Springer, 2006. Citado na página 16.

ITO, Y.; SHIH, A. M.; ERUKALA, A. K.; SONI, B. K.; CHERNIKOV, A.; CHRISOCHOIDES, N. P.; NAKAHASHI, K. Parallel unstructured mesh generation by an advancing front method. **Mathematics and Computers in Simulation**, Elsevier, v. 75, n. 5, p. 200–209, 2007. Citado na página 25.

KIM, S.; JEONG, W.; KIM, C. Lod generation with discrete curvature error metric. In: **proceedings of 2nd Korea Israel Bi-National Conference on Geometrical Modeling and Computer Graphics in the WWW Era**. [S.l.: s.n.], 1999. p. 97–104. Citado na página 39.

KOHOUT, J.; KOLINGEROVÁ, I.; ŽÁRA, J. Parallel delaunay triangulation in E2 and E3 for computers with shared memory. **Parallel Computing**, Elsevier, v. 31, n. 5, p. 491–522, 2005. Citado na página 24.

LI, H.; ZENG, W.; MORVAN, J.; CHEN, L.; GU, X. Surface meshing with curvature convergence. IEEE, 2013. Citado 2 vezes nas páginas 22 e 23.

MIRANDA, A. C.; LIRA, W. W.; CAVALCANTE-NETO, J. B.; SOUSA, R. A.; MARTHA, L. F. A three-dimensional adaptive mesh generation approach using geometric modeling with multi-regions and parametric surfaces. **Journal of Computing and Information Science in Engineering**, American Society of Mechanical Engineers, v. 13, n. 2, p. 021002, 2013. Citado 3 vezes nas páginas 9, 21 e 22.

MPI Forum. **The Message Passing Interface (MPI) Standard**. 2016. Disponível em: <http://www.mcs.anl.gov/research/projects/mpi>. Citado na página 53.

NYKAMP, D. Q. **Surface area of parametrized surfaces**. 2016. Acessado em 04/02/2016. Disponível em: <http://mathinsight.org/parametrized_surface_area>. Citado na página 30.

OWEN, S. J. A survey of unstructured mesh generation technology. In: **Proceedings 7th International Meshing Roundtable**. Dearborn, Michigan: [s.n.], 1998. p. 239–267. Citado na página 17.

ROGERS, D. F.; ADAMS, A. J. **Mathematical elements for computer graphics**. [S.l.: s.n.], 1990. Citado na página 37.

SIQUEIRA, D. M. de; FREITAS, M. O.; CAVALCANTE-NETO, J. B.; VIDAL, C. A.; SILVA, R. J. da. An adaptive parametric surface mesh generation method guided by curvatures. In: **Proceedings of the 22nd International Meshing Roundtable**. [S.l.]: Springer, 2014. p. 425–443. Citado 15 vezes nas páginas 9, 19, 23, 27, 36, 39, 40, 41, 42, 43, 44, 47, 51, 53 e 68.

SIQUEIRA, D. M. de; VIDAL, C. A.; CAVALCANTE-NETO, J. B.; SILVA, R. J. Uma técnica de geração de malhas adaptativas para aplicações de realidade virtual. In: **Proceedings of the 11th Symposium on Virtual and Augmented Reality**. [S.l.: s.n.], 2009. Citado 7 vezes nas páginas 44, 45, 46, 47, 48, 49 e 50.

SOUZA, D. **Geração adaptativa de malha baseada em erro de curvatura**. Dissertação (Mestrado) — Departamento de Computação. Universidade Federal do Ceará. Brasil, 2004. Citado na página 27.

TEIXEIRA, D. N. **Uma Técnica de Decomposição a Priori para Geração Paralela de Malhas Bidimensionais**. Dissertação (Mestrado) — Universidade Federal do Ceará - UFC, mar 2014. Citado na página 65.

The C++ Standards Committee. **JTC1/SC22/WG21 - The C++ Standards Committee - ISOCPP**. 2016. Disponível em: <http://www.open-std.org/jtc1/sc22/wg21>. Citado na página 53.

TREMEL, U.; DEISTER, F.; HASSAN, O.; WEATHERILL, N. P. Parallel generation of unstructured surface grids. **Engineering with Computers**, Springer, v. 21, n. 1, p. 36–46, 2005. Citado 2 vezes nas páginas 24 e 26.

WIKIPEDIA. **Gaussian curvature**. 2015. Disponível em: <https://en.wikipedia.org/wiki/Gaussian_curvature>. Acesso em: 27 Julho. 2015. Citado na página 38.

YILMAZ, Y.; ÖZTURAN, C.; TOSUN, O.; ÖZER, A. H.; SONER, S. Parallel mesh generation, migration and partitioning for the elmer application. p. 1–12, 2010. Citado 2 vezes nas páginas 9 e 25.

ZHAO, D.; CHEN, J.; ZHENG, Y.; HUANG, Z.; ZHENG, J. Fine-grained parallel algorithm for unstructured surface mesh generation. **Computers & Structures**, Elsevier, v. 154, p. 177–191, 2015. Citado na página 26.