UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM

CIÊNCIA DA COMPUTAÇÃO

Aline R. Alves

Uma Metodologia para Representação de Estruturas Anatômicas Utilizando Modelagem NURBS

Dissertação submetida à Universidade Federal de Santa Catarina
como parte dos requisitos para a obtenção do grau de
Mestre em Ciência da Computação

Orientador: Prof. Dr. Aldo von Wangenheim

Coorientador: Prof. Dr. Jáuber Cavalcante de Oliveira

Florianópolis, julho de 2002

Uma Metodologia para Representação de Estruturas Anatômicas Utilizando Modelagem NURBS

Aline R. Alves

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Fernando A. Ostuni Gauthier, Dr.
Coordenador CPGCC, UFSC

Banca Examinadora

Prof. Aldo von Wangenheim, Dr.
Orientador, INE, UFSC

Prof. Jauber Cavalcante de Oliveira, Dr.
Coorientador, MTM, UFSC

Prof. Pierre Galvagni Silveira, Dr.
CCS, UFSC

Para todos aqueles que sonham
e que acreditam em seus sonhos.

Agradecimentos

Meu agradecimento especial ao meu querido Edson.
Sem seu carinho e apoio, este trabalho não seria possível.

Obrigada ao meus familiares, pela compreensão nos muitos momentos de ausência.

Agradeço à Universidade Federal de Santa Catarina pela oportunidade de realizar uma pós-graduação em uma universidade pública e de qualidade, gratuitamente.

Agradeço também à companhia da voz de Zélia Duncan nos momentos mais difíceis.

Sumário

Lista de Ilustrações

Resumo

O projeto Cyclops visa o desenvolvimento de soluções de auxílio a diagnóstico médico, realizado pelo Departamento de Informática e de Estatística da Universidade Federal de Santa Catarina localizada em Florianópolis, SC, em conjunto com a Universidade de Kaiserslautern, localizada na cidade homônima na Alemanha. As ferramentas computacionais desenvolvidas no projeto utilizam técnicas de Inteligência Artificial e Visão Computacional no processamento de imagens digitais. É também utilizada Realidade Virtual na visualização de estruturas anátomo-patológicas em três dimensões.

Este trabalho apresenta o resultado das atividades realizadas, no contexto do projeto Cyclops, na área de visualização de imagens médicas. A pesquisa realizada teve como principal propósito, apresentar uma metodologia para representação de estruturas anatômicas em três dimensões, utilizando um modelo de representação baseado em curvas NURBS, proporcionando a visualização das imagens através de técnicas de Realidade Virtual.

Foi desenvolvido posteriormente, um protótipo, contemplando o método aqui apresentado, que tomando como entrada um conjunto de pontos representando um objeto 3D em modelo poligonal, gera a representação do objeto em modelo baseado em curvas (NURBS) e sua representação em linguagem de modelagem em Realidade Virtual (VRML).

Palavras-chave: Ajuste de Curvas, Ajuste de Superfícies, NURBS, Splines, VRML, Computação Gráfica, Realidade Virtual

Abstract

The Cyclops Project develops software tools to aid medical diagnosis though medical images analysis, developed at Universidade Federal de Santa Catarina and the University of Kaiserslautern, at Germany.

The tools developed in the context of the Cyclops Project use Artificial Inteligence and Computer Vision to process digital images. Virtual Reality techniques are also used in visualization of anathomical structures in three dimensional space.

This works presents the results of the activities done in the Cyclops Project context during the year of 2002. This research intended to propose a method to generate the represention of anatomical structures in the three dimensional space, using a model based on NURBS curves providing the visualization of the images though Virtual Reality techniques, in the Internet.

A prototype was developed following the method proposed here. The software uses as input a set of 3D points that represents an geometrical structure of an object, and generates a parametrization needed to represent such object using a NURBS surfaces based modeling and last, the image described using Virtual Reality Modeling Language (VRML).

Keywords: NURBS, Splines, Curve Fitting, Surface Fitting, VRML, Virtual Reality, Computer Graphics

1 Introdução e Objetivos

Uma das consequências do grau de desenvolvimento alcançado em ciência e tecnologia está no fato de que os tipos de problemas enfrentados nestas áreas são cada vez mais complexos e precisam de conhecimentos e ferramentas de projeto e cálculo cada vez mais potentes.

Nesse contexto, a Computação Gráfica tem sido utilizada nas mais diversas aplicações em muitas áreas de conhecimento, tais como arte, medicina, arquitetura, publicidade, etc. Seu poder de auxílio tem aumentado progressivamente na medida em que vão surgindo computadores com maior poder de cálculo, recursos gráficos e linguagens de programação adaptadas às necessidades atuais.

As imagens tridimensionais tem sido cada vez mais utilizadas nos sistemas devido ao rápido aumento da capacidade computacional.

A visualização de imagens médicas em três dimensões atualmente vem se tornando uma grande ferramenta para o auxílio ao trabalho dos profissionais da área médica. A Computação Gráfica utilizando os recursos fornecidos pela Realidade Virtual, tem proporcionado meios para a análise digital de imagens e manipulação das mesmas, possibilitando a percepção de aspectos não observados de outra maneira.

Considerando a melhoria significativa dos resultados obtidos pela computação aplicada à área médica, ainda pode-se perceber o consumo de recursos computacionais e complexidade utilizados na análise das imagens médicas. Os sistemas desenvolvidos necessitam de máquinas com considerável poder de processamento e muitas vezes, constituem aplicações específicas.

Assim, torna-se necessário um método que possibilite a visualização de diversas estruturas anatômicas de forma genérica, bem como a sua manipulação. Um sistema de baixo custo envolvendo tais métodos, que necessite pouco poder computacional e que possibilite o intercâmbio de imagens em diferentes máquinas independentemente da plataforma utilizada, constitui uma ferramenta bastante aplicável em diversas áreas da medicina.

O projeto Cyclops é um projeto de pesquisa que utiliza a análise automatizada de imagens para o desenvolvimento de soluções de auxílio a diagnóstico médico, realizado

no Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, em conjunto com a Universidade de Kaiserslautern, localizada na cidade homônima na Alemanha. Sob coordenação do Prof. Dr. Aldo von Wangenheim, o projeto desenvolve ferramentas computacionais objetivando o auxílio ao diagnóstico de problemas tais como câncer de mama, neurocisticercose, derrame cerebral, e também contempla pesquisas nas áreas de cirurgia endovascular (objetivando o auxílio ao desenvolvimento de próteses sob medida), telemedicina e desenvolvimento de atlas cerebral digital.

As ferramentas desenvolvidas no projeto realizam processamento de imagens radiológicas através de técnicas de Inteligência Artificial e Visão Computacional, e utilizam representação de imagens tridimensionais e bidimensionais para a visualização das estruturas anátomo-patológicas resultantes do processamento. Imagens tridimensionais, juntamente com aplicação de técnicas de Realidade Virtual, são utilizadas para visualização e manipulação de estruturas anátomo-patológicas no espaço 3D. A figura 1.1 representa o processamento de imagens médicas no projeto Cyclops sob o ponto de vista deste trabalho.

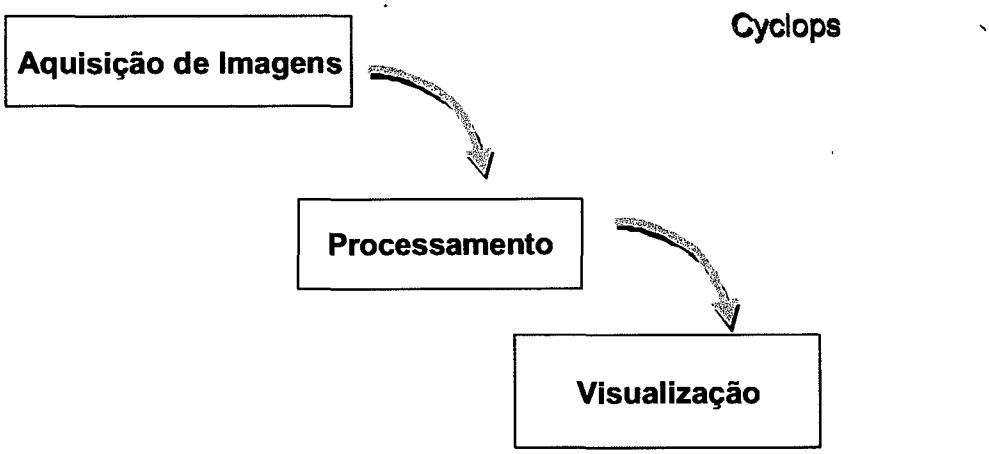


Figura 1.1: Processamento de imagens no Cyclops

As imagens 3D são geradas a partir de imagens bidimensionais obtidas através de exames de tomografia computadorizada ou ressonância magnética. As imagens bidimensionais são processadas pelas ferramentas do sistema Cyclops, que utilizam técnicas de segmentação para obter as formas geométricas significativas para o estudo,

por exemplo, a forma do crânio, descartando o tecido gorduroso. No entanto, o método de segmentação utilizado atualmente no projeto Cyclops, que visa encontrar o limiar de uma estrutura anatômica, utiliza um processo que gera uma forma poligonal aproximada da estrutura anatômica. Tem-se como resultado da segmentação um polígono que fornece informações sobre a forma do objeto através dos pontos representados pelos seus vértices e cada polígono é tomado como uma "fatia" da imagem tridimensional.

A figura 1.2 mostra imagens médicas que são processadas pelas ferramentas do Cyclops, obtendo-se uma forma poligonal aproximada. Estas imagens são obtidas em exames de tomografia computadorizada e representam estudo na área de cirurgia endovascular.

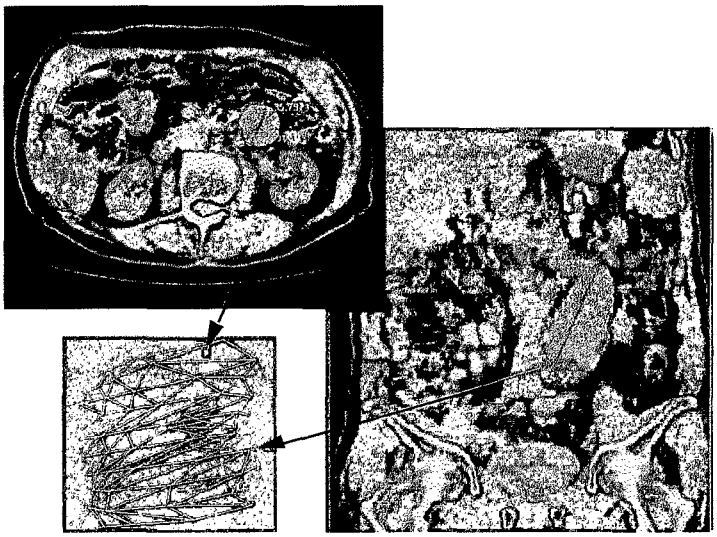


Figura 1.2: Obtenção das informações de estudo. Fonte: Projeto Cyclops

Os vértices de cada polígono são unidos aos vértices dos polígonos adjacentes formando uma malha poligonal representando uma estrutura tridimensional em modelo de fio de arame (*wireframe*). São utilizadas faces triangulares e assim, este processo é chamado de triangularização.

A figura 1.3 mostra um exemplo de uma estrutura em 3D representada em modelo de fio de arame com faces poligonais, representando a reconstrução tridimensional de um segmento de artéria.

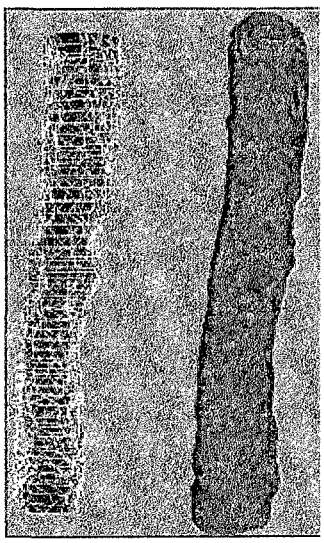


Figura 1.3: Segmento de artéria em modelo de fio de arame - *wireframe*

Cada vértice dos polígonos é um ponto tridimensional composto pelas coordenadas x,y,z. Estes pontos são utilizados como parâmetros para a representação da imagems em um espaço tridimensional.

O projeto Cyclops utiliza a linguagem VRML como linguagem de modelagem em Realidade Virtual para a visualização das estruturas anátomo-patológicas em três dimensões, através da *Internet*. Após a geração da malha poligonal, as imagens são escritas em formato VRML para posterior visualização em um programa navegador *Internet,* como *Microsoft Internet Explorer* ou *Netscape Navigator*.

A figura 1.4 ilustra as etapas do processo de visualização tradicional utilizado no projeto Cyclops.

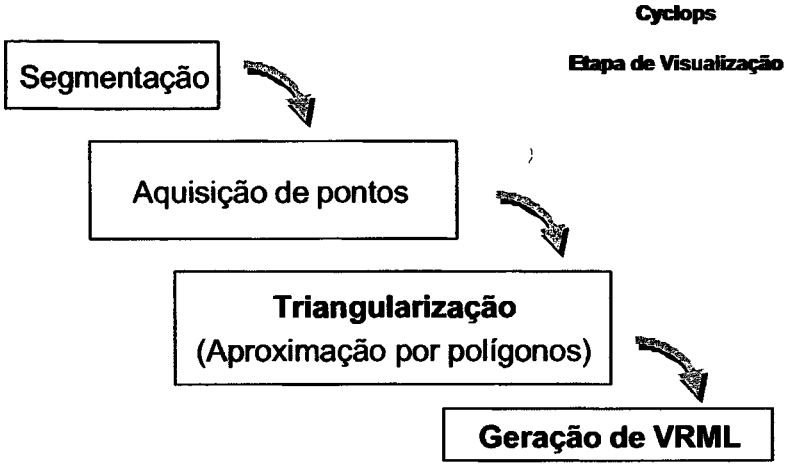


Figura 1.4: Etapas da Visualização no Cyclops

Primeiramente na etapa de visualização, as imagens bidimensionais (exames radiológicos) passam por um processo de segmentação onde são obtidas as formas relevantes ao estudo. Tendo-se as formas relevantes ao estudo, as imagens são processadas e são aproximadas por polígonos e posteriormente cada polígono é unido ao polígono adjacente através de seus vértices, obtendo-se assim, uma estrutura 3D formada por faces triangulares. Por fim, os pontos que representam a imagem no espaço tridimensional são utilizados para sua descrição em linguagem VRML. A utilização de VRML como formato para visualização possui várias vantagens:

- **O formato VRML é um padrão.** Vários sistemas sistemas CAD (*Computer Aided Design*) utilizam o formato VRML atualmente. O formato pode ser utilizado para intercâmbio entre vários sistemas.

- **VRML é um formato compacto.** Por ser um formato baseado no padrão de representação de caracteres UTF8, os arquivos possuem tamanho reduzido, em comparação aos formatos proprietários de imagens gerados por outras aplicações no mercado, tais como os formatos de imagens utilizados nas aplicações 3D Studio Max e Maya.

- **VRML é lido por programas navegadores.** As imagens podem ser visualizadas na

Internet através de um navegador utilizando um programa adicional (*plug-in*). Atualmente existem vários *plug-ins* para VRML. Aqui são utilizados os *plug-ins* desenvolvidos pela Parallel Graphics (Cortona) e Blaxxun.

- **É uma linguagem de modelagem**. É interpretada pelos programas de visualização e não necessita de um processo de compilação.

Assim, o Cyclops se beneficia das características do VRML citadas acima, utilizando este formato nas ferramentas do sistema.

No entanto, apesar do formato VRML padrão ser adequado para transmissão de imagens 3D através da redes de computadores de longa distância, o modelo de representação padrão utilizado ainda é baseado em polígonos. Um objeto 3D mais complexo necessita uma grande quantidade de pontos para caracterizar uma imagem de boa qualidade, o que faz com que a imagem se torne grande para transmissão em redes e custosa em termos de memória para a visualização e espaço para armazenamento. Assim, tais imagens ainda não seriam adequadas às aplicações do projeto, considerando a complexidade dos objetos 3D e a qualidade desejada da imagens.

1.1 Objetivos do Trabalho e Contribuição

1.1.1 Objetivos Gerais

A seguir são listados o principais objetivos gerais deste trabalho.

Obj.1. **Utilização de modelos geométricos 3D baseados em curvas.** Neste trabalho, propõe-se uma metodologia para geração de representação de objetos tridimensionais baseado em curvas utilizando coordenadas tridimensionais

Obj.2. **Desenvolvimento de uma ferramenta de baixo custo para geração de estruturas 3D baseadas em curvas, a partir de nuvem de pontos 3D.** Desenvolver uma

ferramenta computacional que, a partir de pontos tridimensionais que representam um objeto em estrutura poligonal, forneça uma estrutura 3D baseada na forma original dos pontos de entrada, porém utilizando um modelo geométrico caracterizado por curvas.

Obj.3. **Geração de imagens em Realidade Virtual.** As imagens 3D resultantes do modelo geométrico devem ser apresentadas em linguagem de modelagem em Realidade Virtual como resultado final do processamento, para a sua visualização.

1.1.2 Objetivos Específicos

O principal objetivo específico deste trabalho é propor um método transformação da representação de estruturas anatômicas tridimensionais baseada em polígonos para a representação tridimensional baseada no modelo de curvas NURBS, e posteriormente a geração de imagens de Realidade Virtual para sua visualização através de Internet, no escopo do projeto Cyclops.

Os dados resultantes do modelo NURBS serão então utilizados para a geração de imagens no formato VRML extendido pela empresa Blaxxun.

Para a geração das imagens em VRML baseadas no modelo NURBS será necessário fornecer os parâmetros para os objetos (nós) VRML de representação NURBS. Os parâmetros necessários para tal representação tratam-se dos pontos de controle, do vetor de nós, os pesos e o grau das curvas NURBS. Esses aspectos são explicados em mais detalhes no capítulo 3. A figura 1.5 contextualiza este trabalho considerando as etapas do processo de visualização de imagens no Cyclops, apresentado anteriormente.

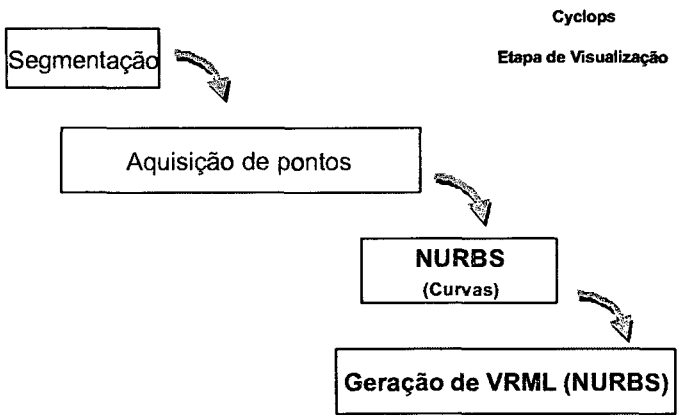


Figura 1.5: Objetivos do Trabalho

Faz-se necessário um processo de geração dos parâmetros NURBS a partir dos pontos que constituem o objeto. O problema constitui-se mais complexo devido ao fato de um conjunto de pontos de controle não correspondem aos mesmos pontos utilizados para representar o objeto no modelo poligonal e também o fato de se ter várias curvas que satisfazem o mesmo conjunto de pontos de controle. A princípio tem-se infinitas soluções para o problema. Assim, um método de ajuste de curvas faz-se necessário. O ajuste garante uma determinada precisão aos resultados obtidos na transformação de pontos 3D em parâmetros NURBS. A seguir são listados os principais objetivos específicos deste trabalho, de maneira sucinta.

Obj.1. **Utilização de modelos geométricos 3D baseados em curvas.**

Obj.1.1 A metodologia proposta deve utilizar o modelo geométrico baseado em curvas B-Spline Racionais Não-uniformes (NURBS - Non Uniform Rational B-Spline). As curvas NURBS são caracterizadas como curvas próprias para a representação de superfícies complexas para a geração de imagens de boa qualidade e exatidão matemática. O modelo baseado em curvas necessita de menos pontos para a representação geométrica pelo fato de utilizar características matemáticas que induzem a forma do objeto. Unindo-se as características do modelo geométrico baseado em curvas às características da representação visual baseada em Realidade Virtual, é possível desenvolver-se uma ferramenta que utilize poucos recursos computacionais, gerando imagens tridimensionais de boa qualidade e exatidão matemática atendendo às necessidades do sistema Cyclops. A figura 1.6 ilustra um objeto 3D formado por curvas.

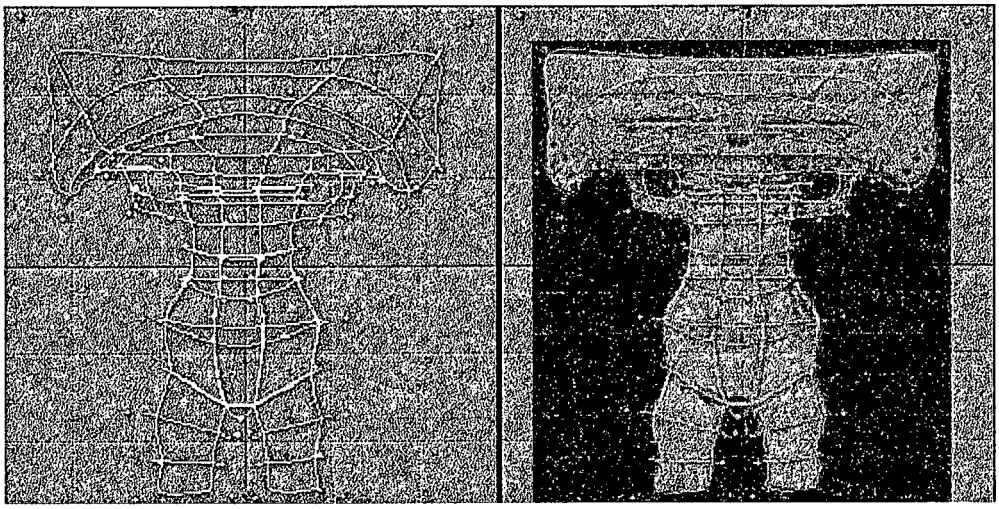


Figura 1.6: Objeto 3D formado por curvas NURBS

Fonte:http://www.geocities.com/SiliconValley/Lakes/2057/index.html

Obj.2. Desenvolvimento de uma ferramenta para geração de estruturas 3D baseada em curvas, a partir de nuvem de pontos 3D.

Obj.2.1 Desenvolver uma ferramenta que utiliza como dados de entrada pontos tridimensionais, gerados por uma ferramenta do sistema Cyclops, que representam anatomias tridimensionais baseadas no modelo geométrico poligonal. O protótipo deve fornecer como pré-resultado os parâmetros significativos (pontos de controle) para a construção do objeto tridimensional no modelo NURBS. Os pontos de entrada são provenientes da segmentação de imagens radiológicas utilizadas em exames de pacientes reais.

Obj.2.2 Para a geração das imagens no modelo baseado em curvas, é necessário fornecer os parâmetros do modelo NURBS. Para isto, é necessária a utilização de um método de ajuste de curvas que, por aproximação, permita que sejam encontrados tais parâmetros a partir de pontos de entrada.

Obj.3. Geração de imagens em Realidade Virtual.

Obj3.1 Utilizar a linguagem de modelagem em Realidade Virtual VRML para a visualização das imagens através de programa navegador, bem como a transmissão

através da *Internet*.

A figura 1.7 apresenta uma reconstrução 3D de uma estrutura anatômica em VRML. Pode-se perceber as faces planares que compõem a superfície do objeto. A percepção destas faces na visualização da imagem constitui falta de qualidade visual na imagem. O ideal é que se tenha uma imagem com uma superfície o mais suave possível.

O formato VRML padrão não fornece suporte ao modelo NURBS e assim, será utilizada uma extensão da linguagem proposta pela empresa Blaxxun. Esta extensão suporta modelagem baseada em curvas. As principais vantagens de se utilizar um modelo baseado em curvas é a melhoria da qualidade das imagens, em termos de suavização das superfícies e a redução do número de pontos para sua representação.

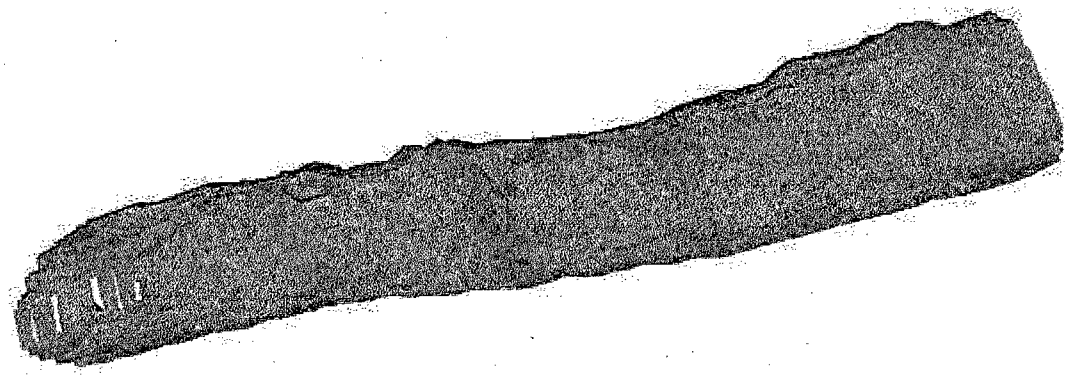


Figura 1.7: Reconstrução facetada de uma artéria
Fonte: Projeto Cyclops.

Deve-se ressaltar que atualmente o sistema Cyclops utiliza a representação geométrica baseada em polígonos. A transformação deste modelo em um modelo baseado em curvas, traria um ganho considerando a diminuição da quantidade de recursos de armazenamento para as imagens, devido à compactação das mesmas, e também a diminuição do *overhead* necessário para a transmissão destes arquivos através de redes de longa distância. Outra melhoria seria percebida pela qualidade das imagens pela suavização das superfícies e da aquisição de mais exatidão matemática na representação dos objetos 3D. É importante ressaltar que no modelo geométrico poligonal é necessário um grande número de pontos 3D para aumentar a qualidade das imagens. O modelo baseado em curvas utiliza apenas os pontos significativos para caracterizar a forma do

objeto.

1.2 Requisitos

Este trabalho deve ser desenvolvido obedecendo os seguintes requisitos:

Req. 1 - O formato das imagens deve ser compacto. O número de parâmetros de representação de um objeto tridimensional baseado em curvas (pontos de controle) deve ser menor que o número de parâmetros de representação do mesmo objeto no modelo poligonal (pontos 3D). A representação baseada em curvas utiliza menos pontos e os arquivos em formato VRML são pequenos.

Req. 2 - Qualidade da imagem. Os pontos de controle devem fornecer uma superfície mais similar possível à forma original do objeto e no entanto, deve ter boa qualidade visual.

Req. 3 - Exatidão matemática. A representação 3D deve ser invariante em relação a transformações geométricas, como translação e mudança de escala.

Req. 4 - Formato VRML. O formato das imagem deve ser compatível com VRML padrão que atualmente é utilizado no projeto Cyclops.

1.3 Estrutura do Trabalho

A estrutura deste trabalho está organizada como se segue. Inicialmente o capítulo 2 mostra a revisão bibliográfica realizada nesta pesquisa, de forma sucinta. O capítulo 3 fornece os conceitos básicos sobre a teoria da representação NURBS. Os capítulos 4 e 5 descrevem os principais conceitos relacionados com a representação em linguagem de modelagem de Realidade Virtual VRML e VRML utilizando representação NURBS, respectivamente.

O capítulo 6 apresenta a metodologia adotada neste trabalho, conceituando o método de ajuste de curvas, utilizado para gerar a parametrização necessária para a representação dos objetos tridimensionais utilizando um modelo baseado em curvas.

O capítulo 7 descreve os resultados obtidos na fase de testes, utilizando o protótipo desenvolvido. Por fim, os capítulos 8 e 9 constituem as conclusões deste trabalho e as referências bibliográficas, respectivamente.

Adicionalmente, o anexo 1 traz fragmentos de código Smalltalk correspondente à implementação e o anexo 2 traz código VRML relativos aos exemplos comentados nos capítulos seguintes.

2 Revisão Bibliográfica

Vários sistemas proprietários de geração de imagens tridimensionais em VRML foram encontrados no mercado. Como exemplo pode-se citar o Maya e o 3D Studio Max, que são os softwares bastante conhecidos, utilizados para modelagem 3D. Estes sistemas, apesar de utilizarem o formato VRML padrão (versão 2), como um dos formatos disponíveis para exportação de objetos tridimensionais, não fornecem meios para a integração direta ao sistema Cyclops. Estes sistemas, que são proprietários e requerem custo elevado na compra de licenças, são utilizados principalmente para modelagem (construção de objetos) e não foram encontrados registros da sua utilização para aquisição e transformação de nuvens de pontos em superfícies baseadas em curvas, mesmo possuindo recursos para construção de objetos baseados em curvas (NURBS - Non Uniform Rational B-Splines) (Lammers, 2002).

A maioria dos sistemas de geração de código VRML, proprietários ou de código aberto, utiliza o modelo de representação de objetos tridimensionais baseado em faces poligonais.

Os sistemas que utilizam modelagem baseada em curvas (NURBS) são restritos no que se refere à construção de superfícies NURBS utilizando uma nuvem de pontos 3D específica, sendo utilizados na maioria das vezes, para a criação de formas livres (modelagem de objetos). Estes sistemas são em geral, proprietários com código fonte fechado e não podem ser adaptados as necessidades do projeto e integrados ao sistema Cyclops, ou não permitem a construção de superfícies de um modo automatizado fornecendo como resultado do processamento da nuvem de pontos inicial, a parametrização necessária para a representação NURBS.

No entanto, mesmo sistemas de modelagem baseados em NURBS que geram imagens em formato VRML, utilizam o versão padrão da linguagem VRML, que por sua natureza geométrica, não suporta a representação dos objetos utilizando curvas (NURBS). Estes sistemas fazem uma aproximação do objeto 3D representado em NURBS para a representação baseada em polígonos utilizando o nó VRML correspondente a um conjunto de faces planares. Embora a linguagem VRML padrão suporte apenas a representação de objetos utilizando polígonos, há uma extensão da linguagem proposta

pela empresa Blaxxun, onde há suporte à representação baseada em curvas. Este trabalho propõe a utilização desta extensão de VRML. Apesar de existirem pesquisas na geração de superfícies utilizando um nuvem de pontos como parâmetro de entrada, não foram encontrados registros de implementações que contemplem numa mesma aplicação todas as características que atendam as necessidades do projeto Cyclops.

Assim, as aplicações encontradas no mercado, que foram analisados neste trabalho até o presente momento, não atenderiam todas as necessidades do Cyclops, já que o enfoque das aplicações é voltado pra modelagem e não a transformação de um conjunto arbitrário de pontos.

O sistema Cyclops, depois de processar as imagens (exames de tomografia computadorizada) através do processo de segmentação e o processo de triangularização (geração da malha poligonal 3D), fornece os pontos 3D que constituem a malha poligonal do objeto tridimensional. Esses pontos podem ser utilizados por uma aplicação para a geração de uma imagem tridimensional. Atualmente no projeto Cyclops, é utilizada a linguagem VRML padrão (versão 2) para a geração das imagens para visualização. O processo de geração da malha poligonal não fornece informações para a geração direta de uma representação baseada em curvas NURBS, pois esta, devido às suas características, requer uma parametrização diferente.

2.1 Trabalhos Relevantes em Geração de Superfícies

A seguir são citados alguns trabalhos importantes que propõem técnicas para geração de superfícies.

2.1.1 (Krishnamurthy, 1996) propõe uma técnica semi-automática baseada em mapas de deslocamento *(displacement maps)* calculados a partir de dados de entrada obtidos através de scanners 3D. Seu trabalho é mais voltado para a parte artística, onde a qualidade visual das imagens é mais importante que a fidelidade em relação aos dados. Krishnamurthy propõe um método de geração de superfícies NURBS a partir de um conjunto arbitrário de pontos 3D. Um objeto complexo é então dividido em várias partes caracterizando superfícies independentes compondo uma espécie de "colcha de retalhos".

No entanto, para evitar um problema de otimização, as superfícies originais (partes da colcha de retalhos) são alteradas para satisfazer os requisitos do ajuste através de um processo chamado relaxamento (*relaxation*). Este processo utiliza um cálculo de forças resultantes em um determinado ponto e altera a posição deste. É também utilizada a subdivisão da malha poligonal, como processo adicional no refinamento dos dados originais. A forma básica do objeto é capturada por um processo de ajuste, e os detalhes da superfície são fornecidos por técnicas de mapas de deslocamento. No caso, os detalhes da superfície são fornecidos por uma imagem (*bitmap*). Para maiores detalhes sobre mapas de deslocamento recomenda-se a leitura de (Foley, 1997).

A imagem 2.1 demonstra a técnica empregada, onde uma forma aproximada do objeto em conjunto com duas imagens utilizadas no mapa de deslocamento geram uma imagem resultado.

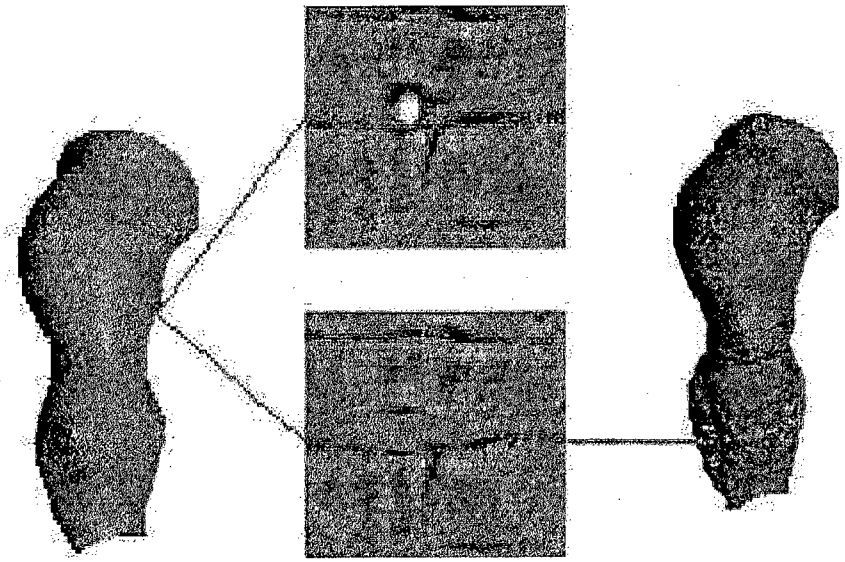


Figura 2.1: Técnica proposta por Krishnamurthy
Fonte: (Krishnamurthy, 1996)

A técnica proposta por Krishamurthy é bastante utilizada em animação. No entanto, perde-se a precisão em relação aos dados originais quando o objeto original é alterado. Quando se trata de um objeto caracterizado por uma imagem médica, a fidelidade em relação aos dados é necessária.

2.1.2 Em (Forsey, 1995) é proposta uma técnica para geração de B-Splines a partir

de um conjunto inicial de pontos de controle. Seu trabalho reforça as características de animação fornecidas pelas B-Splines. É proposta uma técnica de refinamento local das superfícies NURBS. Esta característica é interessante quando as superfícies são geradas e possivelmente podem sofrer alterações em sua topologia através da modificação de pontos, acréscimo ou decréscimo do número de pontos. Para a utilização de métodos de ajuste locais, são necessários parâmetros que caracterizem as curvaturas presentes na superfície.

A principal característica deste trabalho é o processo de refinamento de uma superfície NURBS, considerando dados de curvatura, para se conseguir a forma desejada. Esta técnica não pode ser empregada diretamente pois são desconhecidos os ponto de controle que compõem a superfície e não se tem dados sobre as curvaturas. São fornecidos pelo sistema Cyclops somente os pontos de uma malha poligonal.

2.1.3 (Zorin, 2000) propõe uma técnica para manipulação e edição de superfície onde os dados são obtidos através de scanners 3D. A técnica em questão utiliza a subdivisão dos polígonos que compõem a superfície, onde a região que possui mais irregularidade é subdividida até se obter uma qualidade de apresentação desejada, considerando certo nível de continuidade. Neste caso a malha poligonal inicial é alterada. Zorin utiliza teoria dos grafos para obter os parâmetros para a subdivisão. A figura 2.2 ilustra a técnica proposta por Zorin.

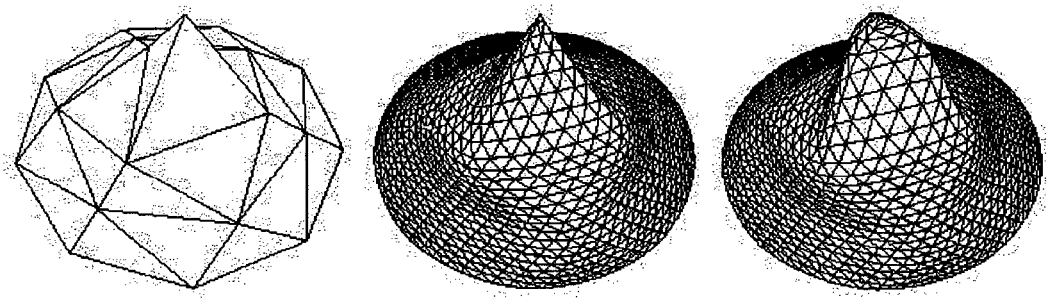


Figura 2.2: Técnica de subdivisão de polígonos proposta por Zorin

Neste caso é utilizada a subdivisão visando obter-se continuidade C1 (para mais detalhes, ver capítulo 3). Pode-se observar que o número de pontos que compõem o objetos aumentam quando o objeto se torna mais complexo. Neste caso necessita-se um

grande número de pontos para descrever um objeto, fazendo com que a imagem se torne maior, não satisfazendo um requisito deste trabalho.

2.1.4 (Segars, 1998) propõe a estrutura NURBS 4D onde os pontos de controle possuem além das coordenadas cartesianas, uma coordenada a mais, relativa ao tempo. É proposta uma superfície dinâmica que muda com o tempo, resultando em animação. A figura 2.3 ilusta a animação dos batimentos cardíacos. Este trabalho é aplicado na área médica, visando a simulação dos movimentos cardíacos. Neste caso, o objeto representado é um modelo genérico de uma estrutura anatômica. Aqui é utilizada a representação de estruturas anatômicas com modelagem NURBS e a extensão da linguagem VRML que suporta NURBS, para visualização.

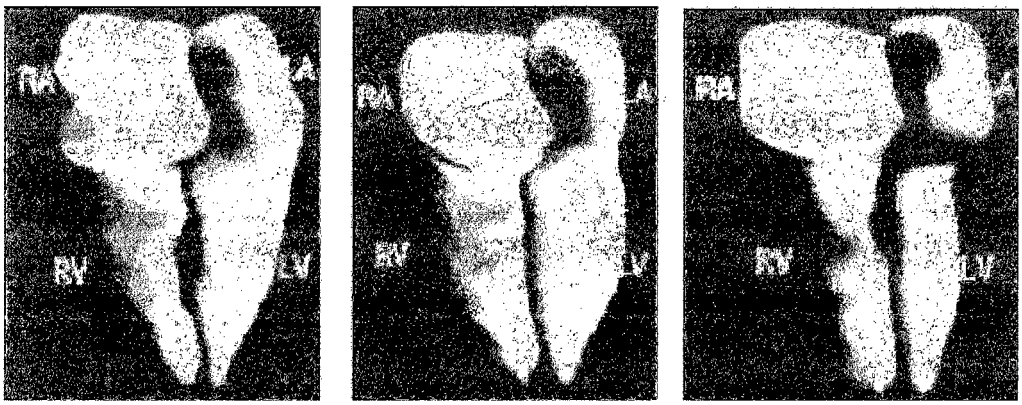


Figura 2.3: Spline 4D. Fonte: www.unc.edu

Os objetos anatômicos são gerados a partir de softwares de modelagem e os dados utilizados para a construção do objeto são fornecidos pelo usuário, sendo simplesmente uma representação de uma forma anatômica e não um objeto reconstruído a partir de dados de exames de pacientes reais.

Este trabalho propõe uma metodologia para a reconstrução de superfícies a partir de dados de exames reais, visando obter fidelidade em relação aos dados de entrada. Assim, a proposta de Segars, apesar de utilizar representação NURBS em imagens anatômicas representadas em VRML, não atenderia a todos os requisitos deste trabalho se fosse aplicada diretamente para a resolução do problema.

Foram descritos neste capítulo alguns trabalhos considerados relevantes durante o

desenvolvimento desta pesquisa. No entanto, os trabalhos apresentados aqui não satisfazem simultaneamente os requisitos deste trabalho, de forma que possam ser aplicados diretamente na resolução do problema.

3 Introdução à Representação NURBS

Este capítulo apresenta os conceitos básicos da representação NURBS, denominada Non Uniform Racional B-Spline. São também apresentadas aqui, as principais propriedades das curvas e superfícies NURBS. Para maiores detalhes sobre as fórmulas matemáticas mostradas neste capítulo, recomenda-se a leitura de (Tyller, 1997). Considerando os requisitos apresentados no capítulo 1, é utilizada neste trabalho a representação NURBS para permitir que os requisitos 2 e 3 sejam satisfeitos.

3.1 Formas Paramétricas e Implícitas

Na representação NURBS, um objeto tridimensional (3D) é composto conjuntos de superfícies e as superfícies são compostas de curvas NURBS. Uma curva NURBS, baseada em curvas B-Spline, é definida por um **grau** de sua função base (um polinômio), pelos **pontos de controle** que alteram a forma da curva, por um conjunto de **pesos** aplicado aos pontos de controle e pelo **vetor de nós**. Para se alterar a forma da curva pode-se utilizar a elevação ou diminuição do grau da curva, alterando a localização dos pontos de controle, modificando a multiplicidade e peso desdes, ou a alteração da multiplicidade dos nós (alterando assim a continuidade da curva). Esses parâmetros são apresentados a seguir.

A figura 3.1 ilustra um conjunto de curvas (em branco), tratadas separadamente.

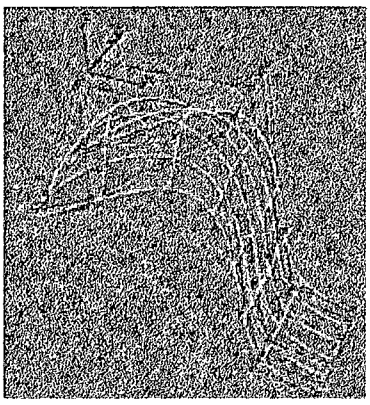


Figura 3.1: Curvas NURBS que compõem uma superfície. Fonte. www.blaxxun.com

Cada curva completa é formada por vários segmentos de curva. A figura 3.2 ilustra uma curva formada por 3 segmentos (C1, C2 e C3) definidos pelo parâmetro u, unidos por pontos de junção.

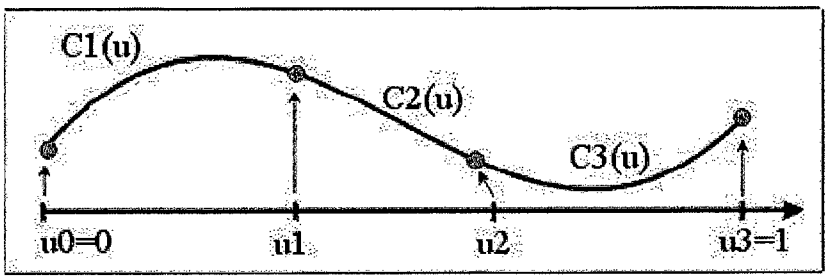


Figura 3.2: Curva formada por 3 segmentos.

Os dois métodos de representação mais utilizados para representar curvas e superfícies em modelagem geométrica são as **equações implícitas** e as **funções paramétricas** (Tyller, 1997).

As equações implícitas de uma curva definida sobre o plano xy, por exemplo, têm a forma f(x,y)=0. Esta equação descreve um relacionamento implícito entre as coordenadas x e y dos pontos pertencentes à curva. Para uma dada curva a equação é única em relação a uma constante. Um exemplo, é um círculo de raio unitário centrado na origem, especificado pela equação $f(x, y) = (x^2 + y^2) = 1$ (3.1)

Na forma paramétrica, cada coordenada de um ponto na curva é representado separadamente como uma função explícita de um parâmetro independente.

Assim, C(u) é uma função da variável independente denotada por u. Embora o intervalo que define u, denotado por [a,b] seja arbitrário, é comumente normalizado para [0, 1]. O primeiro quadrante do círculo mostrado na equação 3.1 é definido pelas funções paramétricas:

$$C(u) = (x(u), y(u)) \qquad (a \geq u \geq b) \qquad (3.2)$$

E considera-se:

$$x(u) = \cos(u) \qquad y(u) = sen(u) \qquad 0 \leq u \leq \pi/2 \qquad (3.3)$$

Sendo $t=tan(u/2)$, pode-se derivar a representação alternativa:

$$y(u) = \frac{2t}{1+t^2} \quad x(u) = \frac{1-t^2}{1+t^2} \qquad 0 \leq t \leq 1 \qquad (3.4)$$

Ou seja, $C(u) = (x(u),y(u)) = \left(\frac{2u}{1+u^2},\frac{1-u^2}{1+u^2}\right)$ (3.5)

Assim, pode-se perceber que a representação paramétrica de uma curva não é única. Para uma mesma curva, pode-se obter várias representações paramétricas. Tendo as funções x(u), y(u), e z(u) como arbitrárias, pode-se obter uma variedade de curvas diferentes, mesmo para uma mesma representação paramétrica, como por exemplo, a representação mostrada na equação 3.5. Entretanto, há restrições quando se implementa um sistema de modelagem geométrica. O ideal é se limitar a uma classe de funções utilizadas tal que:

- Sejam capazes de representar precisamente todas as curvas necessárias no sistema;
- Sejam facilmente, eficientemente e precisamente processadas por um computador, em particular;
- A computação dos pontos e derivadas na curva seja eficiente;
- O processamento numérico das funções seja relativamente estável a erros de ponto flutuante;
- As funções necessitem de poucos recursos de armazenamento.
- Sejam simples e matematicamente coerentes.

Assim, a representação NURBS utiliza funções B-Spline, por atender os requisitos citados acima.

3.2 Definição e Propriedades das Curvas B-Spline Racionais Não Uniformes (NURBS)

Uma classe de curvas e superfícies paramétricas é a curva B-Spline Racional Não Uniforme (*Non Uniform Rational B-Spline – NURBS*). A representação NURBS está sendo utilizada para fins computacionais sendo facilmente processada, por ser relativamente estável a erros de ponto flutuante, ter poucos requisitos de memória e pela habilidade de representar qualquer tipo de curva ou superfície. Uma curva NURBS de grau p é definida como:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad a \le u \le b \qquad (3.6)$$

Um curva é formada por segmentos caracterizados por funções N, B-Splines de grau p, definidas pelo parâmetro u pertencente a um intervalo [a, b]. Os coeficientes geométricos P_i são chamados de pontos de controle. Os (n+1) pontos de controle agem sobre a curva, alterando sua forma, formando o chamado polígono de controle. Pode-se perceber que os pontos de controle interpolam os extremos da curvas, mas os demais pontos não fazem parte da curva. A figura 3.3 demonstra o comportamento dos pontos de controle agindo sobre uma curva.

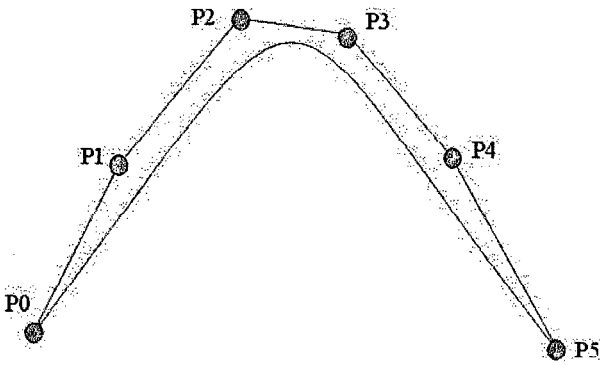


Figura 3.3: Pontos de controle

As funções $N_{i,p}(u)$ são definidas recursivamente como:

$$N_{i,0}(u) = \begin{cases} 1 & se\ u_i \le u \le u_{i+1} \\ 0 & \text{senão} \end{cases} \qquad (3.7)$$

$$N_{i,p}(u) = \left(\frac{u - u_i}{u_{i+p} - u_i}\right) N_{i,p-1}(u) + \left(\frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}}\right) N_{i+1,p-1}(u) \qquad (3.8)$$

A figura 3.4 ilustra a definição recursiva das curvas B-Spline.

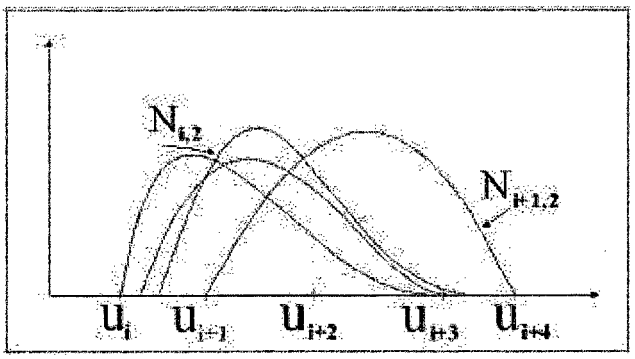


Figura 3.4: B-Splines

Assim, são definidas as seguintes propriedades:

1. $N_{i,0}(u)$ é uma função igual a zero, exceto no intervalo $u \in [u_i, u_{i+1})$;
2. Para p>0, $N_{i,0}(u)$ é uma combinação linear de duas funções base de grau (p-1);
3. O cálculo de um conjunto de funções base requer a especificação de um vetor de nós U e de um grau p;
4. $N_{i,p}(u)$ são polinômios definidos sobre a linha real e normalmente só o intervalo $[u_0,u_m]$ é de interesse;
5. O intervalo $[u_i, u_{i+1})$ é chamado iésimo gerador de nós (*knot span),* e define o intervalo onde a função é diferente de zero.

As funções base $N_{i,p}(u)$ são as funções base B-Spline definidas sobre o vetor não periôdico chamado vetor de nós (*knot vector*), com m+1 nós, ou seja, são os pontos de junção das funções que compõem uma curva. A figura 3.2 mostra uma curva formada por três segmentos (C2, C2 e c3) unidos por pontos de junção, que caracterizam o vetor de nós. Assume-se que a=0 e b=1:

$$U = \{a_0,\ldots,a_p,u_{p+1},\ldots,u_{m-p-1},b_0,\ldots,b_p\}$$

Os w_i são escalares chamados pesos, que agem sobre os pontos de controle. Quando os pesos variam, um ponto de controle pode agir com mais influência sobre a curva, modificando sua forma. O exemplo mostrado a seguir, na figura 3.5, ilustra melhor esta característica.

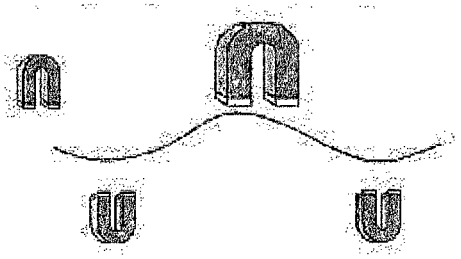


Figura 3.5: O efeito da mudança de peso do terceiro ponto de controle.

Uma curva consistindo apenas de um segmento é inadequada. Os problemas de se ter um único segmento variam da necessidade de se ter um alto grau para a curva, para se conseguir com precisão uma forma complexa, o que é ineficiente de processar e numericamente instável, à necessidade de geração interativa de curvas na qual um único segmento possui limitações a medida que se aumenta o número de pontos de controle.

Para resolver esse problema, uma curva formada por vários segmentos é utilizada. Uma curva formada por vários segmentos é construída a partir de várias funções B-Spline unidas por seus pontos de junção (nós) onde existe algum nível de continuidade entre os segmentos (não necessariamente a mesma continuidade entre cada segmento). Um exemplo deste tipo de curva é mostrado na figura 3.2.

A curva $C(u)$ é definida em $u \in [0, 1]$ e é composta por segmentos $C_i(u) \quad 0 \leq i \leq m$. Os segmentos são unidos nos pontos de junção $0 = u_0 < u_1 < u_2 < u_3 < u_4 = 1$ com algum nível de continuidade. Uma curva é dita ser C^k contínua se no ponto de junção u_i, se $C_i^{(j)}(u_i) = C_{i+1}^{(j)}(u_i)$ para todo $0 \leq j \leq k$ onde $C_i^{(j)}$ representa a j-ésima derivada de C_i.

A figura 3.6 exemplifica alguns níveis de continuidade possíveis entre dois segmentos de curva. No primeiro exemplo, não há continuidade entre os segmentos. No segundo exemplo os pontos de junção coincidem, mas as derivadas nos pontos são diferentes, caracterizando a continuidade C^o ou posicional. No terceiro exemplo, a primeira derivada nos dois pontos coincide e é chamada de continuidade C^1 (continuidade tangencial). No quarto exemplo a segunda derivada nos pontos de junção

dos segmentos é igual e existe a continuidade C^2 (continuidade de curvatura).

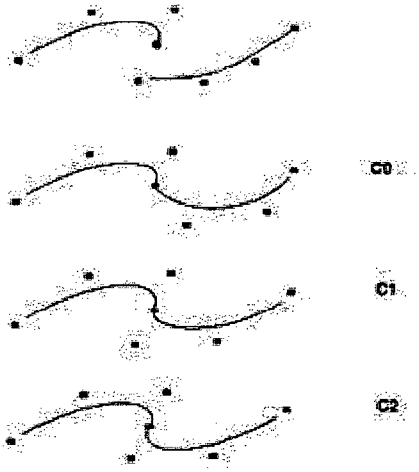


Figura 3.6: Exemplos de tipos de continuidade. Fonte: (Tyller, 1997)

A seguir são listadas as propriedades das curvas B-Spline. A partir destas propriedades são definidas as propriedades das curvas NURBS (B-Spline racional não uniforme). Seja C(u) uma curva B-Spline de grau p definida pela equação 3.9:

$$\sum_{i=0}^{n} N_{i,p}(u)P_i \qquad a \le u \le b \qquad (3.9)$$

Onde os P_i são os pontos de controle e as $N_{i,p}$ são as funções base B-Spline definidas sobre o vetor de nós U.

Pode-se observar as seguintes propriedades:

P1. Se n=p e U ={0,.....,0,1,.....,1}, então C(u) é uma curva Bézier.

P2. C(u) é um polinômio formado por várias curvas; o grau p, o número de pontos de controle n+1 e o número de nós, m+1 são relacionados de forma que: m=n+p+1

P3. Há interpolação nos pontos finais: $C(0) = P_0$ e $C(1) = P_n$;

P4. **Invariância Afim**: Uma transformação é aplicada na curva através dos pontos de controle. Seja r um ponto em $\mathfrak{E}^3$ (espaço Euclidiano tridimensional). Uma transformação afim, denotada por Φ mapeia $\mathfrak{E}^3$ em $\mathfrak{E}^3$ e tem a forma de $\Phi(r) = Ar + v$. Onde A é uma matriz 3x3 e v é um vetor.

P5. Casco convexo: A curva está contida em um casco convexo formado por seu polígono de controle;

P6. Esquema de modificação local: movendo-se P_i, a curva C(u) muda somente no intervalo u_i, u_{i+p+1}.

P7. O polígono de controle representa uma aproximação da curva; esta aproximação é melhorada através da inserção de nós ou elevação de grau. Como regra geral, quanto menor o grau, mais próxima a curva está do seu polígono de controle.

P8. Movendo-se ao longo da curva de u=0 até u=1, as funções $N_{i,p}(u)$ agem como pontos de mudança.

P9. Propriedade da variação mínima: nenhum plano (linhas no caso de 2D) tem mais intersecções com a curva que seu polígono de controle.

P10. C(u) é infinitamente diferenciável no interior do intervalo de nós e é pelo menos p-k vezes diferenciável em um nó com multiplicidade k.

P11. É possível utilizar múltiplos pontos de controle, alterando a continuidade da curva.

Foram citadas as principais propriedades das curvas Spline. A seguir serão descritas as principais características das curvas B-Spline racionais não uniformes (curvas NURBS). Lembrando que as curvas NURBS são definidas como:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad a \le u \le b \qquad (3.10)$$

Pode-se reescrever esta definição como sendo

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \qquad (3.11)$$

Sendo que C(u) pode ser escrita na forma

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) P_i \qquad (3.12)$$

As $R_{i,p}(u)$ são as funções base racionais definidas considerando $u \in [0, 1]$. São definidas as seguintes propriedades:

P1. Não negatividade: $R_{i,p} \geq 0$ para todo $i, p, u \in [0, 1]$;

P2. Partição de Unidade: $\sum_{i=0}^{n} R_{i,p}(u) = 1$ para todo $u \in [0, 1]$;

P3. $R_{0,p}(0) = R_{n,p}(1) = 1$;

P4. Para $p>0$, todo $R_{i,p}(u)$ alcança um máximo no intervalo $u \in [0, 1]$

P5. Suporte local: $R_{i,p}(u) = 0$ para $u \notin [u_i, u_{i+p+1})$;

P6. Todas as derivadas de $R_{i,p}(u)$ existem no interior do gerador de nós (*knot span*), onde é uma função racional com denominador não zero;

P7. Se todo $w_i = 1$, então $R_{i,p}(u) = N_{i,p}(u)$;

P8. $C(0) = P_0$ e $C(1) = P_n$;

P9. Invariância Afim: Uma transformação é aplicada na curva através dos pontos de controle;

P10. Casco convexo: A curva está contida em um casco convexo formado por seu polígono de controle;

P11. Propriedade da variação mínima: Nenhum plano (linhas no caso de 2D) tem mais intersecções com a curva, que seu polígono de controle;

P12. C(u) é infinitamente diferenciável no interior do intervalo de nós e é pelo menos p-k vezes diferenciável em um nó com multiplicidade k;

P13. Uma curva NURBS sem nós interiores é uma curva Bézier racional, desde que $N_{i,p}(u)$ reduz a $B_{i,p}(u)$, ou seja, uma curva B-Spline pode ser um curva Bézier.

P14. Aproximação local: Se o ponto de controle P_i é movido ou o peso w_i é modificado, somente a porção da curva no intervalo $[u_i, u_{i+p+1})$ é afetada.

3.3 Definição e Propriedades das Superfícies NURBS

Uma superfície representa um mapeamento de uma região R em R3. Assim, uma

superfície S pode ser definida como:

$$S(u, v) = (x(u, v), y(u, v), z(u, v)) \qquad (u, v) \in R \qquad (3.13)$$

A figura 3.7 ilustra esta definição.

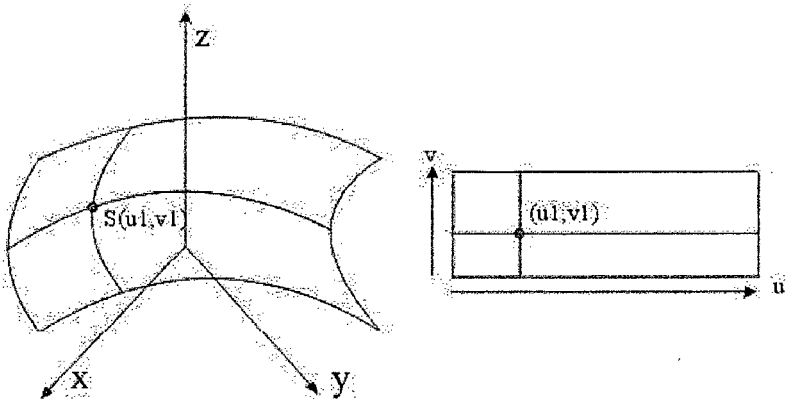


Figura 3.7: Superfície definida em u,v

Uma superfície NURBS de grau p, q nas direções u e v respectivamente, é uma função racional da forma:

$$S(u, v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad 0 \le u, v \le 1 \qquad (3.14)$$

Os $\{P_{i,j}\}$ formam uma malha de controle bidirecional (n+1)x(m+1), os $\{w_{i,j}\}$ são os pesos e as $\{N_{i,p}(u)\}$ e $\{N_{j,q}(v)\}$ são funções base B-Spline não racionais definidas sobre os vetores de nós:

$$U = \{a_0,\ldots,a_p,u_{p+1},\ldots,u_{r-p-1},b_0,\ldots,b_p\}$$

$$V = \{a_0,\ldots,a_q,u_{q+1},\ldots,u_{s-q-1},b_0,\ldots,b_q\}$$

Sendo que U tem r+1 nós e V tem s+1 nós definindo que r=n+p+1 e s=m+q+1.

Considerando a função base na forma:

$$R_{i,j}(u, v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad (3.15)$$

A superfície NURBS pode ser reescrita como

$$S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)P_{i,j} \qquad (3.16)$$

A figura 3.8 ilustra uma superfície NURBS caracterizada por seus pontos de controle.

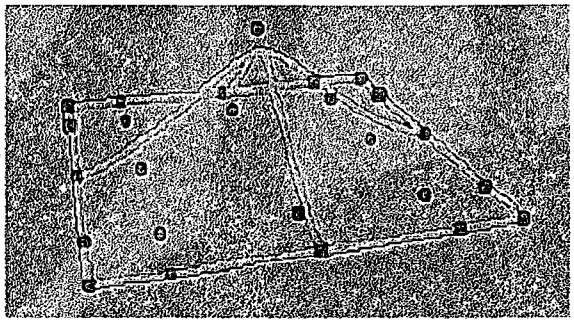


Figura 3.8: Superfície NURBS e seus pontos de controle.

As propriedades importantes das funções $R_{i,j}(u, v)$ são praticamente as mesmas definidas para as funções base $N_{i,p}(u)$ e $N_{j,q}(v)$. Estas propriedades são apresentadas a seguir, de forma sucinta.

P1. Não negatividade: $R_{i,j}(u,v) \geq 0$ para todo i, j, u, v;

P2. Partição de unidade: $\sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v) = 1$ para todo $(u,v) \in [0,1]\times[0,1]$;

P3. Suporte local: $R_{i,j}(u,v) = 0$ se (u, v) está fora do retângulo definido por $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$;

P4. Em um dado retângulo na forma $[u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1})$ no mínimo (p+1)(q+1) funções base são diferente de zero, em particular as $R_{i,j}(u,v)$ para $i_0 - p \leq i \leq i_0$ e $j_0 - q \leq j \leq j_0$ são diferentes de zero;

P5. $R_{0,0}(0,0) = R_{n,0}(1,0) = R_{0,m}(0,1) = R_{n,m}(1,1) = 1$;

P6. Superfícies B-Spline e Bézier não racionais e Bézier racionais são casos especiais de superfícies NURBS.

3.4 Fator de Tesselação (*Tesselation*)

Renderizar uma superfície é um processo de duas etapas. A primeira etapa consiste em calcular os pontos que constituem a malha e a seguir enviar estes dados para a API 3D. A primeira etapa, chamada *tesselation* consiste em transformar uma representação contínua em uma representação discreta. Normalmente as APIs 3D utilizam uma aproximação por triângulos. Os *plug-ins* utilizados para a visualização usam normalmente APIs como Open-GL e DirectX. Neste caso a superfície NURBS, que é uma representação matemática, é aproximada para um conjunto de faces planares, para a visualização. Pode-se utilizar um valor fixo para o número de subdivisões da superfície, chamado fator de tesselação *(tesselation).*

A imagem 3.9 ilustra uma superfície NURBS. Neste caso a imagem é caracterizada por faces triangulares, devido à renderização feita pelo software utilizado para visualização. Sendo que a superfície é definida seguindo os conceitos apresentados anteriormente.

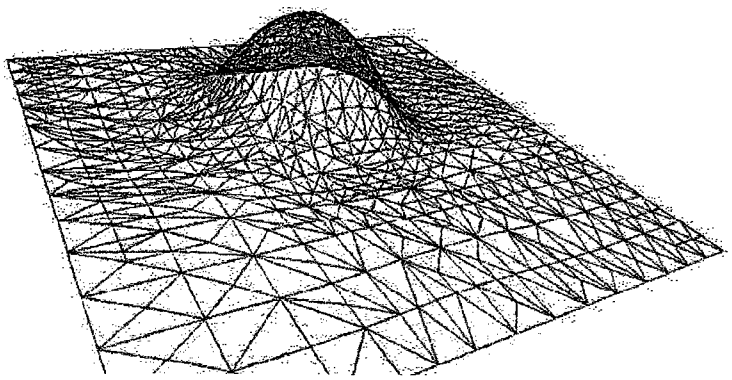


Figura 3.9: Superfície NURBS vista em modo *wireframe*

Foram apresentadas neste capítulo as principais características da representação NURBS. Pode-se perceber que são utilizados vários parâmetros para a representação de uma superfície ou curva e cada parâmetro, tal como grau, peso, ponto de controle ou nó, pode ser utilizado para alterar a forma ou as propriedades de uma curva ou superfície. Assim, uma curva ou superfície NURBS é definida por um conjunto de parâmetros que se relacionam para caracterizar a forma de um determinado objeto.

4 Introdução à Linguagem VRML

VRML é abreviação de Virtual Reality Modeling Language ou linguagem para modelagem em Realidade Virtual. É uma linguagem independente de plataforma que permite a criação de cenários em 3D, visualização de objetos por ângulos diferentes e interação com eles através de um programa navegador de *Internet*.

A linguagem foi concebida para descrever simulações interativas de múltiplos participantes, em mundos virtuais disponibilizados na Internet e ligados com WWW (*World Wide Web*), mas a primeira versão da linguagem não possibilitou muita interação do usuário com o mundo virtual.

Nas versões futuras foram acrescentadas características como animação, movimentos de corpos, som e interação entre múltiplos usuários em tempo real. A última versão é a 2.0, chamada *Moving Worlds VRML 2.0*.

Esta linguagem tem como objetivo dar suporte necessário para o desenvolvimento de mundos virtuais tridimensionais multi-usuário na *Internet*, sem precisar de redes de alta velocidade. O código de VRML é um conjunto de formato UFT-8 (*Unicode*) do Open Inventor, da Silicon Graphics, com características adicionais para navegação na WEB. Esta característica é semelhante às âncoras em HTML. Assim, pode-se criar âncoras em um ambiente virtual que levem a outros ambientes virtuais.

A linguagem, na sua versão 1.0, trabalha com geometria 3D, que permite a criação de objetos baseados em polígonos, possuindo alguns objetos pré-definidos como triângulo, esfera, cilindro, cubo e cone, suportando transformações como rotação, translação e mudança de escala. Permite também a aplicação de texturas, luz sombreamento, etc. Outra característica importante da linguagem é o Nível de Detalhamento (LOD, *Level of Detail)*, que permite o ajustamento da complexidade dos objetos, dependendo da distância em que eles se encontram do observador

4.1 Características de VRML

Tudo que se precisa para criar uma cena em VRML é um editor de textos. Uma vez

que os arquivos são salvos em formato UTF-8 com a extensão **.wrl**. A linguagem apenas descreve como os ambientes tridimensionais devem ser apresentados, pois os arquivos não são compilados.

A versão 1.0 é uma simplificação do *Open Inventor* da *Silicon Graphics*, com propriedades de materiais, transformações, visões de câmera, texturas mapeadas e iluminação. Há pouca possibilidade de interação, tendo como principal objetivo a criação de mundos virtuais estáticos e criação de âncoras para outros ambientes.

A versão 2.0, padrão ISO-IEC 14772-1:1997, permite que os objetos do mundo virtual possam se movimentar e responder a eventos, baseados nas iniciativas do usuário. Permite também a utilização de objetos multimídia, como sons e filmes. As principais características desta versão são: melhoria dos mundos estáticos, interação, animação, comportamento baseado em scripts e prototipação de novos objetos VRML.

A linguagem VRML é baseada no conceito de nós (*Nodes*), onde cada nó é um objeto com alguma funcionalidade. Geralmente o nome do nó identifica sua funcionalidade. Cada nó é formado por atributos que o caracterizam. Esses atributos podem assumir valores numéricos, caracteres, e URLs indicando arquivos externos ou não. As URLs podem indicar links para outros arquivos VRML, arquivos HTML, ou imagens. A figura 4.1 mostra a estrutura hierárquica de nós que compõem uma cena VRML.

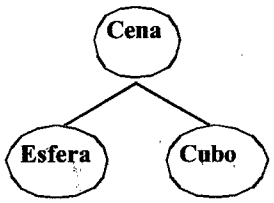


Figura 4.1: Estrutura hierárquica de nós que compõem uma cena VRML exemplo.

Os nós VRML constituem a cena a ser apresentada ao usuário através de um navegador e contém toda a informação a respeito da cena, como a forma dos objetos, sua cor, sua textura e localização no mundo virtual. A figura 4.2 mostra uma cena VRML vista através de um navegador.

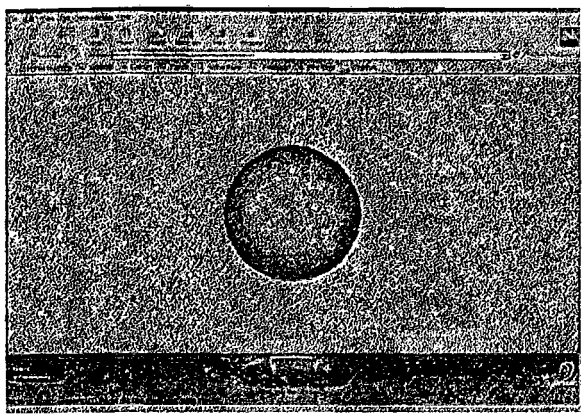


Figura 4.2: Uma esfera em três dimensões em VRML.

É necessário um programa adicional (*plug-in*) ao navegador para possibilitar a visualização de mundos virtuais VRML. Os mundos virtuais VRML são compostos basicamente de texto e podem ser gerados a partir de um editor gráfico ou um editor de textos.

O código gerado é interpretado pelo navegador através do *plug-in* e a cena é mostrada ao usuário. O usuário interage com os objetos do mundo virtual através de botões apresentados na tela, gerados pelo *plug-in*. A seguir, na listagem 4.1, é mostrado um código exemplo de uma cena básica escrita em linguagem VRML que representa uma esfera. O exemplo mostra a sintaxe do nó *Shape* e seus atributos (*appearance* e *geometry*). Os atributos dos nós podem ser caracterizados por valores ou por outros nós (no caso *Appearance* e *Sphere*). Os nomes de nós iniciam com letra maiúscula e os nomes de atributos iniciam com letra minúscula.

```
#VRML V2.0 utf8
Shape {
     appearance Appearance {
                material Material {
                              diffuseColor 1 0.5 0}
                geometry Sphere{radius 2}}
       }
```

Listagem 4.1: Sintaxe VRML.

Maiores detalhes sobre a sintaxe VRML versão 2, encontram-se na especificação da linguagem. A especificação (VRML, 1997) apresenta todos os objetos VRML em

detalhes e mostra também como é a geração de *scripts,* na linguagem VRMLScript, para a criação de objetos dinâmicos (animações).

O capítulo seguinte apresenta as principais características da representação VRML utilizando a representação baseada em curvas NURBS.

5 A Extensão NURBS paraVRML

É utilizada neste trabalho a extensão VRML que permite a utilização de modelagem NURBS, proposta pela empresa Blaxxun. Esta extensão não faz parte da especificação da linguagem VRML padrão, mas utiliza as mesmas características da linguagem citadas no capítulo 4. Este capítulo apresenta as principais características da extensão proposta pela Blaxxun.

A utilização do modelo NURBS em VRML traz as seguintes vantagens:

- Tamanho reduzido dos arquivos devido à representação compacta de NURBS;
- Formas com superfícies mais suaves;
- Autoria facilitada devido a vários softwares gráficos utilizarem NURBS;
- Melhor animação devido à mudança de poucos parâmetros causarem grande impacto na forma das curvas;
- Escalabilidade automática de apresentação (*Level of Detail-LOD*) dependendo da CPU e performance da placa de vídeo e complexidade da cena.

As curvas e superfícies NURBS tem sido utilizadas em desenho industrial de objetos como carros, barcos, aviões, etc, devido às suas características matemáticas serem adequadas para a modelagem de formas fluidas, com superfícies mais suaves. A grande utilização de NURBS em aplicação CAD/CAM/CAE, justifica-se pelas seguintes razões:

- Formas podem ser representadas utilizando NURBS sem perda de exatidão matemática.
- Métodos de geração e manipulação de formas são fornecidos por várias ferramentas de mercado, como 3DStudio Max e Maya.
- Os algoritmos são relativamente rápidos e estáveis.

A listagem 5.1 mostra a sintaxe do nó VRML que representa uma curva NURBS.

```
NurbsCurve {
  field           MFFloat  knot          []
  field           SFInt32  order         3
  exposedField    MFVec3f  controlPoint  []
  exposedField    MFFloat  weight        []
  exposedField    SFInt32  tessellation
}
```

Listagem 5.1: Nó NURBSCurve.

A curva NURBS é caracterizada pelos parâmetros ***knot***, ***order***, ***controlPoint***, ***weight*** e ***tesselation***. O parâmetro knot é caracterizado por um conjunto de valores numéricos reais (**MFFloat**), caracterizando o vetor de nós da curva. O parâmetro order indica a ordem da curva através de um valor inteiro (**SFInt32**). O parâmetro controlPoint é formato por um conjunto de pontos no espaço tridimensional (**MFVec3f**), sendo um vetor de pontos de controle. O parâmetro chamado ***weight*** é descrito através de um conjunto de valores numéricos reais (**MFFloat**), caracterizando os pesos dos pontos de controle, também um vetor.

Por fim, o fator *tesselation* é indicado através de um valor inteiro (**SFInt32**) e caracteriza a qualidade da renderização da imagem.

Para a geração de uma curva NURBS em VRML, é apenas necessário que se forneça os valores dos parâmetros do nó VRML. Esse código será então interpretado pelo navegador e a cena será gerada. Um exemplo de um objetos VRML baseado em NURBS é mostrado na figura 5.1.

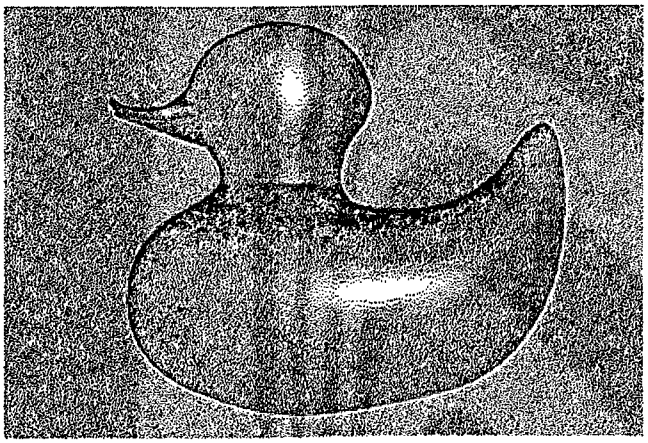


Figura 5.1: Gráfico VRML baseado em NURBS. Fonte: www.blaxxun.com

Um exemplo mais complexo de um objeto NURBS representado pela linguagem VRML, gerado pelo protótipo desenvolvido aqui, pode ser visto na listagem 5.2. É descrito pelo nó VRML uma superfície NURBS, com cada campo apresentando valores.

```
NurbsSurface {
          ccw FALSE
          solid FALSE
          uOrder  3
          vOrder 3
          uDimension 7
          vDimension 7
 uKnot [ 0.0 0.0 0.0 0.215231 0.360922 0.506614 0.652306 1.0 1.0 1.0
]
 vKnot [ 0.0 0.0 0.0 0.238419 0.404924 0.571429 0.737933 1.0 1.0 1.0
]

 controlPoint [
  8.0 0.0 1.0, 8.0 0.0 1.81979, 8.0 0.0 2.95569, 8.0 0.0 4.04471,
  8.0 0.0 4.95821, 8.0 0.0 6.76596, 8.0 0.0 8.0, 7.25783 0.0 1.0,
  7.25783 -0.0343931 1.81979, 7.25783 0.0864405 2.95569,
  7.25783 0.0868814 4.04471, 7.25783 -0.0134844 4.95821,
  7.25783 0.0119486 6.76596, 7.25783 0.0 8.0, 5.95745 0.0 1.0,
  5.95745 0.0824312 1.81979, 5.95745 -0.207175 2.95569,
  5.95745 -0.208232 4.04471, 5.95745 0.0323185 4.95821,
  5.95745 -0.0286376 6.76596, 5.95745 0.0 8.0, 5.04845 0.0 1.0,
  5.04845 -0.52601 1.81979, 5.04845 1.32203 2.95569 ,
  5.04845 1.32877 4.04471, 5.04845 -0.206231 4.95821,
  5.04845 0.182742 6.76596, 5.04845 0.0 8.0, 3.93581 0.0 1.0,
  3.93581 -0.522398 1.81979, 3.93581 1.31295 2.95569,
  3.93581 1.31965 4.04471, 3.93581 -0.204815 4.95821,
  3.93581 0.181487 6.76596, 3.93581 0.0 8.0, 2.94376 0.0
  1.0, 2.94376 0.230983 1.81979, 2.94376 -0.580532 2.95569,
  2.94376 -0.583493 4.04471, 2.94376 0.0905606 4.95821,
  2.94376 -0.0802462 6.76596, 2.94376 0.0 8.0, 1.0 0.0 1.0, 1.0 0.0
  1.81979,
  1.0 0.0 2.95569, 1.0 0.0 4.04471, 1.0 0.0 4.95821, 1.0 0.0 6.76596,
  1.0 0.0 8.0 ]
 weight [
  1.0 1.0 1.0 1.0 1.0 1.0 1.0
  1.0 1.0 1.0 1.0 1.0 1.0 1.0
  1.0 1.0 1.0 1.0 1.0 1.0 1.0
  1.0 1.0 1.0 1.0 1.0 1.0 1.0
  1.0 1.0 1.0 1.0 1.0 1.0 1.0
  1.0 1.0 1.0 1.0 1.0 1.0 1.0
  1.0 1.0 1.0 1.0 1.0 1.0 1.0 ]}
 }] }
```

Listagem 5.2: Sintaxe de um nó VRML exemplo

Este objeto VRML traz todos os campos de uma superfície NURBS (ccw, solid, uOrder, vOrder, uDimension, vDimension, controlPoint e weight) com valores, caracterizando a imagem 5.2.

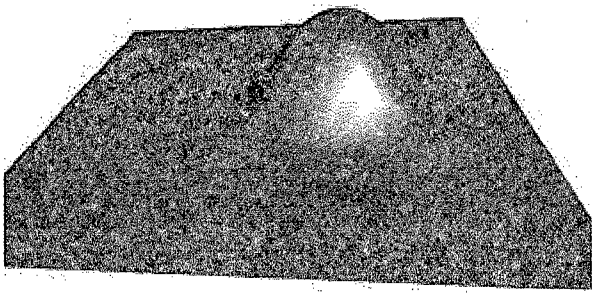


Figura 5.2. Imagem VRML do nó representado na listagem 5.2

Os campos do nó VRML significam:

ccw - *Counterclockwise*, se a superfície deve ser renderizada do sentido anti-horário ou não.

solid - Se o lado interno e externo são vísiveis simultaneamente ou não.

uOrder - Ordem das curvas que compõem a superfície na direção u.

vOrder - Ordem das curvas que compõem a superfície na direção v.

uDimension - Quantos pontos de controle forma uma curva em u.

vDimension - Quantos pontos de controle forma uma curva em v.

controlPoint - Um conjunto de pontos de controle sendo que cada valor corresponde a uma coordenada cartesiana. Cada três valores correspondem as coordenadas x,y e z de un ponto de controle.

weight - Um vetor de pesos definidos sobre os pontos de controle. O primeiro valor do vetor caracteriza o peso atribuído ao primeiro ponto de controle e assim, sucessivamente. Neste caso são definidos pesos iguais a 1 para todos os pontos de controle.

A empresa Blaxxun disponibiliza na sua *homepage,* em www.blaxxun.com, a documentação necessária descrevendo a extensão proposta.

6 Metodologia

O principal desafio deste trabalho constitui-se na transformação dos pontos que representam um objeto 3D baseado em polígonos em parâmetros para a representação do mesmo objeto em modelagem NURBS, seguindo os requisitos definidos no capítulo 1.

Para isto, foram definidas as etapas descritas a seguir:

- Divisão do objeto em conjuntos de superfícies;
- Cálculo dos parâmetros NURBS para cada superfície (Ajuste);
- Verificação da qualidade do resultado do ajuste;
- Geração de VRML com suporte à representação NURBS para visualização.

6.1 Divisão do Objeto em Conjuntos de Superfícies

Os dados fornecidos pelo sistema Cyclops, caracterizando um objeto 3D, formam um conjunto de várias superfícies. Assim os objetos mais complexos são divididos em pedaços mais simples, formando uma espécie de "colcha de retalhos" e o processo de ajuste é realizado para cada superfície separadamente. A figura 6.1 ilustra esta idéia.

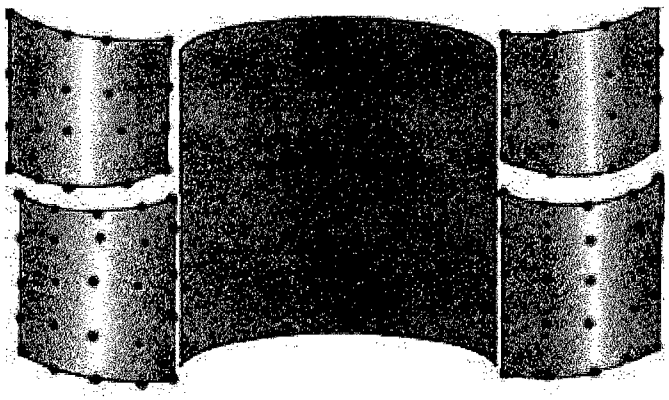


Figura 6.1: Divisão de objeto em conjunto de superfícies

Neste caso os pontos que caracterizam as superfícies mais simples são utilizados no processo de ajuste, para cada superfície independentemente.

A seguir é apresentada a técnica utilizada para o ajuste das superfícies que compõem a "colcha de retalhos" que caracterizam um objeto mais complexo.

6.2 Cálculo dos Parâmetros NURBS para Superfícies (Ajuste)

Nesta etapa foi utilizada a técnica de ajuste de superfícies citada por Wayne Tyller em (Tyller, 1997). Esta técnica de ajuste de superfícies utiliza sucessivas aplicações da técnica de ajuste de curvas, já que uma superfície é composta por diversas curvas, na representação NURBS. A técnica de ajuste de curvas utiliza o método dos Mínimos Quadrados. Através da aplicação do método padrão Mínimos Quadrados, deriva-se a seguinte relação:

$$(N^T N)P = R$$

Onde N é a matriz de funções base da representação NURBS, P são os pontos de controle desconhecidos e R é calculado utilizando-se os pontos tridimensionais de entrada. Neste caso somente os pontos de controle são desconhecidos e os demais parâmetros da representação NURBS, tais como vetor de nós, grau, devem ser fornecidos ou pré-calculados.

O problema constitui-se na aproximação dos pontos 3D que constituem a malha original do objeto em uma malha de pontos de controle e a geração dos vetores de nós para as direções u e v. O ajuste de curvas caracteriza-se como um problema de otimização. Para a aproximação dos pontos é utilizado o método dos mínimos quadrados, que possibilita a resolução de um sistema linear para encontrar os pontos de controle que definem a superfície NURBS, resultado da aproximação. A seguir é apresentada a técnica de ajuste.

6.2.1 Ajuste de Curvas

O ajuste de curvas refere-se à construção destas satisfazendo um conjunto de elementos geométricos, tais com pontos tridimensionais. O termo **ajuste**, refere-se ao fato de que as curvas se "ajustam" a um determinado conjunto arbitrário de pontos, aproximando a tendência destes. Na aproximação, há a captura da tendência dos pontos e a curva não passa pelos pontos precisamente, mas aproximadamente.

Nos casos onde os pontos são obtidos por dispositivos como por exemplo, scanners 3D, podendo conter ruído computacionalo, ou o número de pontos utilizados como dados de entrada é muito grande, é interessante utilizar aproximação. Assim, a curva captura a

forma do objeto de modo aproximado, através da tendência dos pontos, e pode-se descartar os pontos menos significativos. Na aproximação é desejado que se tenha definido um fator de erro máximo utilizado, para cálculo da aproximação com a precisão esperada. O fator de erro pode ser calculado, por exemplo, pela verificação da distância entre o ponto original e o ponto calculado sobre a curva.

A maioria dos algoritmos de ajustes de curvas podem ser categorizados como **locais** ou **globais.** Nos algoritmos de ajuste de curvas globais, um sistema de equações pode ser resolvido. Se os dados consistem de pontos, e se somente os pontos de controle são desconhecidos (grau, nós, e pesos são fornecidos), o sistema é linear e pode ser facilmente resolvido. Se os nós ou pesos são desconhecidos, tem-se um sistema não linear. Teoricamente, nos métodos globais, uma perturbação em um ponto da curva ou superfície, afeta toda a forma da curva ou superfície. Entretanto a magnitude da modificação dos pontos vizinhos diminui a medida que aumenta a distância entre os pontos.

Os métodos de ajuste locais são por natureza algoritmos geométricos e constróem a curva utilizando dados localmente, em cada iteração. Uma perturbação em um determinado ponto da curva ou superfície afeta somente a forma da curva ou superfície de forma local (pode-se fazer uma analogia às propriedades físicas da borracha). Tais algoritmos são normalmente menos custosos do que os algoritmos globais e podem lidar melhor com anomalias como linhas retas, quebras, etc. No entanto, as derivadas nos pontos de junção tem de ser fornecidas.

Os métodos de aproximação utilizam menos pontos para construir o objeto, em comparação com os métodos de interpolação, onde a curva passa por todos os pontos dados. A figura 6.2 ilustra o ajuste de curvas e mostra os dados originais unidos por linhas.

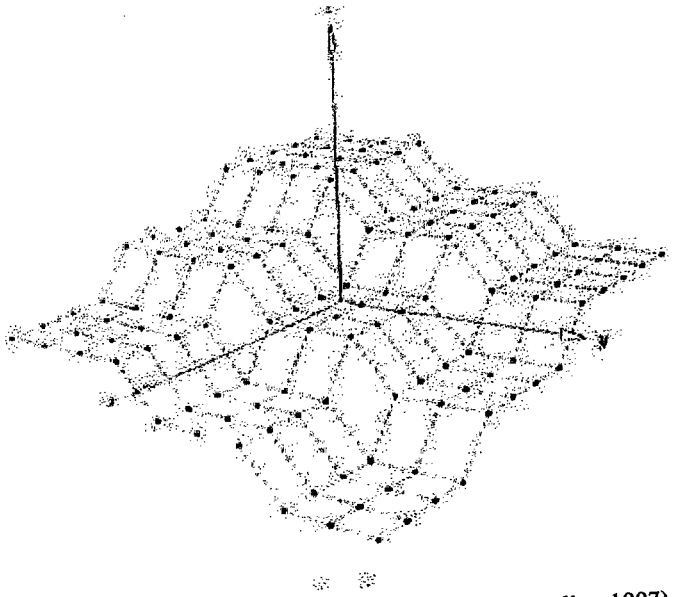


Figura 6.2: Dados utilizados para o ajuste. Fonte: (Tyller, 1997)

A figura 6.3 mostra a superfície ajustada fornecida como resultado do processo iterativo.

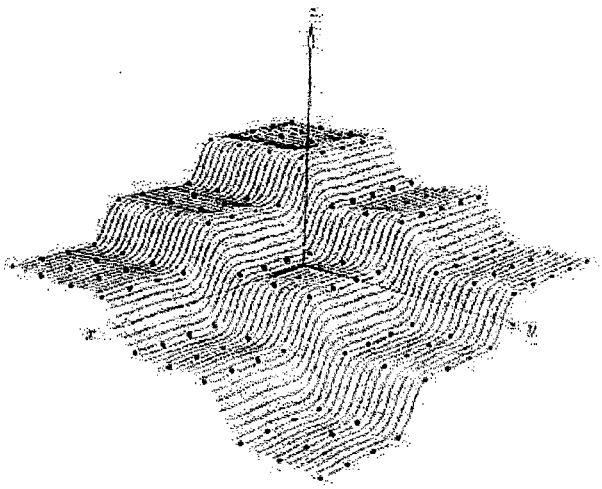


Figura 6.3: Superfície resultado, com grau (2,3) usando uma rede de controle (8x8)
Fonte: (Tyller, 1997)

Há várias pesquisas na área de ajuste de curvas. Muitas técnicas são baseadas em heurísticas, já que geralmente não há respostas consideradas únicas para o problema. O problema fundamental é que não existe uma única solução para o ajuste dos dados e várias

curvas diferentes e matematicamente corretas, que podem ser consideradas resultado do ajuste (Tyller, 1997).

A próxima seção apresenta o ajuste de curvas utilizando mínimos quadrados. Nesta técnica são desconhecidos somente os pontos de controle e os demais parâmetros para a representação NURBS são pré-calculados.

6.2.2 Aproximação de Curvas por Mínimos Quadrados

Esta técnica baseia-se fundamentalmente na resolução de um sistema linear para encontrar a malha de pontos de controle de uma curva que correspondem a um conjunto de pontos de entrada. Para isto, os parâmetros da representação NURBS tais como vetor de nós, grau das curvas e pesos são fornecidos em uma primeira etapa.

A princípio o número de pontos de controle é um parâmetro de entrada para a resolução do sistema. Embora seja possível resolver um sistema linear para o cálculo dos pontos de controle de uma curva, o problema constitui-se mais complexo para superfícies.

A metodologia adotada para ajuste de superfícies é caracterizado como múltiplos ajustes de curvas em ambas as direções u e v.

A seguir é apresentado o método de ajuste de curvas por mínimos quadrados. O ajuste de superfícies utiliza esta técnica considerando uma superfície como um conjunto de curvas.

Para se evitar um problema não-linear, assume-se que os pesos valem 1. Os valores dos parâmetros que determinam os valores das funções B-Spline e os nós são pré-calculados e então resolve-se um único sistema de equações lineares para os n+1 pontos de controle desconhecidos, sendo que o número de pontos de controle n é definido. A princípio, utiliza-se uma curva simples para ser ajustada aos pontos, assumindo que o grau da curva $p \geq 1, n \geq p$ e os (m+1) pontos de entrada $Q_0,...,Q_m$ (m>n) são fornecidos.

Assume-se que:

$$Q_o = C(0) \qquad e \qquad Q_m = C(1)$$

Ou seja, que a curva passa pelos pontos inicial e final.

Os pontos Q_k restantes são aproximados por mínimos quadrados tal que o somatório

dos erros é mínimo, em relação os n+1 pontos de controle P_i. Os parâmetros de ajuste $\{\bar{u}_k\}$ são pré-calculados.

$$\sum_{k=1}^{m-1} |Q_k - C(\bar{u}_k)|^2 = Min! \qquad (6.1)$$

A figura 6.4 ilustra a principal idéia do método de ajuste.

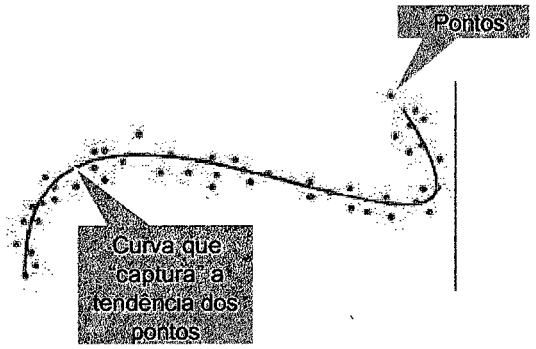


Figura 6.4: Captura da tendência dos pontos pela curva de ajuste

Considera-se os R como sendo o lado direito do sistema linear e calculados como:

$$R_k = Q_k - N_{0,p}(\bar{u}_k)Q_o - N_{n,p}(\bar{u}_k)Q_m \qquad k = 1, \ldots, m-1 \qquad (6.2)$$

Onde Qk é um ponto de entrada e N é uma função B-Spline, definida no parâmetro $\bar{u}_k$ e calculada recursivamente como apresentado no capítulo 3.

Aplica-se o método padrão de mínimos quadrados para encontrar os n-1 pontos de controle, $P_1,\ldots,P_{n-1}$, já que o primeiro ponto de controle e o último são iguais aos pontos dos dados de entrada (a curva interpola os extremos), resultando em $(N^T N)P = R$, (Tyller, 1997). Onde a matriz N (m-1)x(n-1), utilizada para a resolução do sistema é representada como:

$$N = \begin{bmatrix} N_{1,p}(\bar{u}_1) & \cdots & N_{n-1,p}(\bar{u}_1) \\ \cdots & \cdots & \cdots \\ N_{1,p}(\bar{u}_{m-1}) & \cdots & N_{n-1,p}(\bar{u}_{m-1}) \end{bmatrix} \qquad (6.3)$$

R é o vetor de n-1 pontos calculados utilizando os valores R_k calculados anteriormente:

$$R = \begin{bmatrix} N_{1,p}(\bar{u}_1)R_1 + \ldots + N_{1,p}(\bar{u}_{m-1})R_{m-1} \\ \ldots \\ N_{n-1,p}(\bar{u}_1)R_1 + \ldots + N_{n-1,p}(\bar{u}_{m-1})R_{m-1} \end{bmatrix} \qquad (6.4)$$

P é o vetor de pontos de controle desconhecidos:

$$P = \begin{bmatrix} P_1 \\ \ldots \\ P_{n-1} \end{bmatrix} \qquad (6.5)$$

Para cada coordenada de Q(x,y,z), é resolvido o sistema com o lado direito R correspondente a coordenada. Assim, calcula-se um vetor R para cada coordenada dos ponto de entrada. Como os pontos possuem 3 coordenadas cartesianas, são calculados 3 vetores R e o sistema é resolvido para cada coordenada separadamente. Para a resolução do sistema, o vetor de nós U=$\{u_0,...,u_r\}$ e os parâmetros $\{\bar{u}_k\}$ de ajuste, que definem as funções B-Spline, são calculados. Percebe-se que a matriz N pode apresentar muitos elementos, dependendo do número de pontos utilizados no ajuste, e a maioria dos seus coeficientes pode ser igual a zero.

6.2.3 Etapas do Ajuste de Superfícies

A metodologia adotada neste trabalho baseia-se na técnica de ajuste de superfícies utilizando Mínimos Quadrados utilizada por Wayne Tyller em (Tyller, 1997). Esta técnica é considerada simples e adequada para a maioria das aplicações.

A principal idéia utilizada é ajustar os pontos da malha correspondente às curvas em uma direção (u) e posteriormente é aplicado o ajuste novamente nos pontos resultantes, mas através da outra direção (v).

Considera-se os pontos de entrada $\{Q_{k,l}\}$, k= 0,...,r e l= 0,...,s sendo o conjunto de pontos a ser aproximados, com dimensão (r+1)x(s+1), por uma superfície de grau (p, q) com (n+1)x(m+1) pontos de controle. Os valores de r e s são conhecidos, pois definem

as dimensões da matriz de pontos de entrada Q.

O algoritmo adotado interpola os pontos extremos $Q_{0,0}$, $Q_{r,0}$, $Q_{0,s}$ e $Q_{r,s}$ precisamente e aproxima os demais pontos por Mínimos Quadrados. São ajustados s+1 linhas de pontos de Q na direção u, que são armazenados temporariamente, e utilizados para a segunda etapa do ajuste na outra direção produzindo uma superfície de (n+1)(m+1) pontos de controle.

Nesta etapa, cada linha da matriz Q é considerada uma curva e é então aplicado o ajuste de curvas.

A seguir são detalhados os passos do processo de ajuste de superfícies. São fornecidos os seguintes parâmetros: O conjunto Q de pontos que constituem a malha poligonal do objeto 3D. Esse conjunto é fornecido em forma matricial. Também é fornecido o número de pontos de controle que será gerado como resultado, ou seja, o processo de ajuste retornará uma superfície com as dimensões definidas, bem como os graus das curvas na direção u e na direção v.

6.2.3.1 Cálculo dos Parâmetros de Ajuste

Inicialmente é feito o cálculo dos parâmetros de ajuste relativos a direção u, chamado vetor $\{\bar{u}_k\}$ e os parâmetros de ajuste na direção v chamado vetor $\{\bar{v}_l\}$. Estes valores são calculados a partir dos pontos de entrada e definem os valores das funções N nas direções u e v.

Para o cálculo dos parâmetros é utilizado a técnica da média (Tyller, 1997), através de todos os $\{\bar{u}_k^l\}$, para l= 0,...m (m é o numero de pontos Q da curva). Os valores de $\bar{u}_k$ são calculados a partir das distâncias 3D dos pontos de entrada Q na direção u e os valores de $\bar{v}_l$ são calculados a partir das distâncias entre os pontos considerando a direção v (Tyller, 1997):

$$\bar{u}_k = \frac{1}{m+1} \sum_{l=0}^{m} \bar{u}_k^l \qquad k = 0,\ldots,m \qquad (6.6)$$

Tendo-se os vetores de parâmetros $\bar{u}_k$ e $\bar{v}_l$ pode-se calcular os vetores de nós para as direções u e v, utilizando este parâmetro.

6.2.3.2 Cálculo do Vetor de Nós

São necessários no total, n+p+2 nós, ou seja, o número de pontos de controle (n+1) somado à ordem da curva (p+1), segundo a propriedade P2 das curvas B-Splines apresentada no capítulo 3. É calculado o vetor $U=\{u_0,...,u_r\}$, sendo que existem n-p nós internos. O vetor segue a forma {0,0,0,... nós internos, 1,1,1...}. Sendo que existem p+1 nós externos valendo 0 e p+1 nós externos valendo 1.

Os nós externos podem ser calculados como:

$$u_0 = \ldots = u_p \qquad u_{m-p} = \ldots = u_m = 1 \qquad (6.7)$$

$$u_{p+j} = \frac{1}{p}\sum_{i=j}^{j+p-1} \bar{u}_i \qquad j = 1, \ldots, n-p \qquad (6.8)$$

Os nós internos podem ser calculados considerando m como o número de pontos Q, ou seja as dimensões de $\bar{u}_k$ para a direção u e $\bar{v}_l$ para a direção v, d como um número real e i o maior inteiro tal que $i \leq d$:

O valor d pode ser calculado como:

$$d = \frac{m+1}{n-p+1} \qquad (6.9)$$

E i pode ser calculado segundo a equação 6.10:

$$\begin{aligned} i &= int(jd) & \alpha &= jd - i \\ u_{p+j} &= (1-\alpha)\bar{u}_{i-1} + \alpha\bar{u}_i & j &= 1, \ldots, n-p \end{aligned} \qquad (6.10)$$

Assim, são calculados os vetores de nós utilizando os parâmetros das funções B-Spline definidos pela equação 6.6.

6.2.3.3 Cálculo da Matriz N

A matriz N é calculada segundo a equação:

$$N = \begin{bmatrix} N_{1,p}(\bar{u}_1) & \cdots & N_{n-1,p}(\bar{u}_1) \\ \cdots & \cdots & \cdots \\ N_{1,p}(\bar{u}_{m-1}) & \cdots & N_{n-1,p}(\bar{u}_{m-1}) \end{bmatrix} \qquad (6.11)$$

Sendo que cada coeficiente de N, ou seja, as funções $N_{i,p}(\bar{u})$, são definidas recursivamente como mostrado no capítulo 3. Posteriormente é calculada a matriz $N^T N$. São calculadas 2 matrizes N. Uma matriz considera a direção u (utilizando o vetor $\bar{u}_k$ definindo as funções N) e a outra considera a direção v (utilizando o vetor $\bar{v}_l$ definindo as funções N).

6.2.3.4 Cálculo do Vetor R (lado direito do sistema)

A seguinte etapa constitui-se no cálculo do vetor R, utilizado para a resolução do sistema linear. São calculados vetores R para cada coordenada dos pontos a serem aproximados. Assim tem-se os R considerando as coordenadas x, y e z. O cálculo dos R seguem as definições do ajuste de curvas, seção 6.2.

Diferentemente do ajuste de curvas, tem-se o ajuste da superfície na direção u e na direção v. Desse modo, faz-se necessário os vetores R considerando ambas a direções. São calculados assim, os seis vetores R considerando a combinação dos parâmetros u, v e as coordenadas x, y e z.

6.2.3.5 A Resolução do Sistema Linear

Para a resolução do sistema foi utilizado o método padrão de decomposição LU (Cheney, 1994). Como resultado, pôde-se encontrar os pontos de controle. É utilizada a matriz N^T*N, calculada na etapa enterior, os vetores R e são fornecidos como saída os

valores das coordenadas x, y e z dos pontos de controle.

6.2.4 Algoritmo para Aproximação de Superfícies

A seguir é apresentado o algoritmo utilizado para o ajuste de superfícies. O ajuste é feito seguindo um número fixo de pontos de controle fornecidos como entrada (n+1 pontos na direção u e m+1pontos na direção v).

São fornecidos adicionalmente os seguinte parâmetros de entrada:

1. O conjunto de pontos Q, representando a malha poligonal 3D, sendo esta apresentada em forma matricial, suas dimensões são denotadas por r+1 e s+1.
2. As dimensões em u e v relativas a matriz Q, denotadas por r+1 e s+1. Ou seja, quantos pontos formam a malha na direção u e v respectivamente.
3. Os graus a serem utilizados nas curvas NURBS para ambas as direções u e v, representados por p e q.
4. Como saída tem-se a matriz de pontos de controle e os vetores de nós nas direções u e v denotados por U e V.

A estratégia adotada aqui é caracterizada por ajustar s+1 linhas de dados de Q, ajuste na direção u, e utilizar o resultado desta etapa para o ajuste na direção v, resultando nos n+1, m+1 pontos de controle.

Algoritmo AjusteComNMFixos

/* entrada: r,s,Q,p,q,n,m , saida: U,V e P*/

Cálculo dos vetores de parâmetros $\bar{u}k$ e $\bar{v}l$

Cálculo de U e V

Cálculo de Nu e NuTNu

Para j=0 ate s /*ajuste na direção u*/

Temp[0][j] = Q0,l

Temp[n][j] = Qr,j

Cálculo de Rux, Ruy e Ruz

Decomposição LU para calcular os Temp[1][j]...Temp[n-1][j]

fim para

Cálculo de Nv e NvTNv

Para i=0 ate n/*ajuste na direção v*/

P[i][0]=Temp[i][0]

P[i][m]=Temp[i][s]

Cálculo de Rvx, Rvy e Rvz

Decomposição LU para calcular os P[i][1]...P[i][m-1]

fim para

6.3 Verificação da Qualidade do Resultado do Ajuste

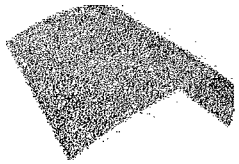
O método de Mínimos Quadrados é utilizado para a etapa de ajuste. Para a verificação da precisão do resultado em relação aos dados originais, é calculado um fator de erro E definido da seguinte forma:

$$\max_{\substack{0 \le k \le r \\ 0 \le l \le s}} \left| Q_{k,l} - S(\bar{u}_k, \bar{v}_l) \right| \qquad (6.12)$$

Sendo que $Q_{k,l}$ representa um ponto da matriz de entrada e $S(\bar{u}_k, \bar{v}_l)$ representa um ponto correspondente da superfície resultado. A correspondência é obtida a partir dos parâmetros $\bar{u}_k$ e $\bar{v}_l$, já que estes são calculados a partir dos pontos originais. O fator de erro é calculado como sendo a máxima distância entre dois pontos, sendo que um ponto pertence a superfície original e o outro ponto correspondente pertence a superfície resultante em NURBS.

Percorrendo-se toda a matriz de dados originais e calculando-se pontos correspondentes sobre a superfície NURBS, pode-se calcular a distância Euclidiana entre os pontos.

A figura 6.5 demonstra a fator de erro encontrado para duas superfícies geradas, sendo que foram utilizados parâmetros de entrada diferentes para cada uma. Foram aproximadas superfícies utilizando graus 1 e 2 para as direções u e v.

Grau 1  Grau 2 

Distâncias da superfície com grau 1:

0.0d 0.310255278349444d 1.256073966947d-15 2.8284271247462d 0.0d
0.233926239550994d 0.38856122206694d 0.233926239550994d 2.8380841223527d 0.233926239550994d
0.0d 0.310255278349444d 1.5383701491069d-15 2.8284271247462d 0.0d
2.0d 2.023921524601 1d 2.0d 3.4641016151378d 2.0d
0.0d 0.310255278349444d 1.2755491433176d-15 2.8284271247462d 0.0d

Distâncias da superfície com grau 2:

0.0d 0.236238581226254d 0.061024596159077d 2.8284271247462d 0.0d
0.180138357804044d 0.2970833135875d 0.190194188367354d 2.834157692852d 0.180138357804044d
0.056569081112575d 0.242917122076054d 0.083210950447041d 2.8289927643842d 0.056569081112575d
2.0d 2.013903837639 7d 2.0009307837445d 3.4641016151378d 2.0d
0.0d 0.236238581226254d 0.061024596159077d 2.8284271247462d 0.0d

Figura 6.5: Erros para as curvas obtidas utilizando grau 1 e 2 respectivamente

Existe uma infinidade de superfícies que podem satisfazer um determinado conjunto de dados de entrada. Pode-se utilizar este fator de erro, como sendo um fator decisório na escolha de um novo conjunto de dados de entrada e na repetição do processo de ajuste até que se obtenha um resultado satisfatório, ou na realização de um processo de ajuste local onde os pontos obtiveram maior distância em relação aos dados originais.

6.3.1 O Processo Iterativo

O processo de ajuste de curvas descrito anteriormente, utiliza um número mínimo de pontos de controle (fixo), escolhido pelo usuário, bem como o grau das curvas na direção u e v.

Após feito o ajuste, é calculado o fator de erro máximo entre os pontos de entrada e os pontos da superfície resultante. Este fator de erro é então utilizado para a realização de um processo iterativo. É fornecido como entrada um número mínimo de pontos de controle para a etapa inicial de ajuste e o processo é repetido, aumentando-se o número de pontos de controle em uma unidade, até que o número de pontos de controle seja igual ao

número de pontos de entrada. É requisito deste trabalho que a superfície resultante não tenha mais pontos que a superfície original. No processamento, é escolhida então a superfície que obteve o menor fator de erro.

Uma outra estratégia seria especificar um valor de erro máximo e realizar o processo iterativo até que o erro seja igual ou menor que o erro máximo especificado. No entanto, foi adotada neste trabalho a primeira estratégia onde é escolhida uma superfície com o menor erro encontrado, em todas as iterações, onde o número de iterações é fixo. Isto devido ao fato de que não se tem um garantia de que o sistema vá sempre convergir, considerando um conjunto qualquer de pontos de entrada, de modo que atinja o valor de erro especificado, dependendo do comportamento dos pontos fornecidos como entrada. Esta pode ser considerada uma limitação do método de ajuste utilizado neste trabalho. O método é bastante sensível em relação à escolha dos pontos que serão ajustados. Quando o objeto é muito complexo ou é caracterizado por objetos que contém furos, por exemplo o método pode não convergir.

6.3.2 O Cálculo da Matriz de Distâncias

Como processo adicional, é calculada uma matriz de distâncias, para a superfície que apresentou o menor fator de erro, ou seja, a superfície escolhida como sendo o melhor resultado. Os coeficientes da matriz de distâncias são definidos como sendo as distâncias Euclidianas entre os pontos da malha poligonal fornecida como entrada e os pontos correspondentes da superfície, calculados posteriormente.

Esta matriz de distâncias é util na verificação da correspondência e precisão do resultado do ajuste e serve para uma etapa posterior de ajuste local onde houve maior distância em relação à malha original.

Esta etapa de refinamento do resultado e possível ajuste local, não é realizada neste trabalho.

6.4 A Representação em VRML

Como apresentado no capítulo 4, VRML é uma linguagem de modelagem bastante

fácil de se utilizar, pois é uma linguagem para representação de objetos 3D em forma de texto.

Foi utilizada a extensão da linguagem VRML padrão proposta pela Blaxxun. O formato compatível com a representação NURBS pôde ser facilmente integrado, com a geração de um arquivo no formato *.wrl,* fornecendo-se os parâmetros calculados para as superfícies.

Obtendo-se uma superfície adequada para a representação do objeto, é então realizado o processo de geração da superfície em representação VRML. A partir desta etapa, a superfície pode ser visualizada em um programa navegador Web, dotado de um *plug-in* que suporte a extensão NURBS.

6.5 Discussão

Em relação a metodologia proposta aqui, são apresentadas a seguir, algumas considerações em relação aos requisitos deste trabalho.

Req. 1 - O formato das imagens deve ser compacto.

Em relação a este requisito, pode-se garantir que o número de pontos de controle seja menor que o número de pontos de entrada. Como pode ser visto nos resultados descritos no capítulo 7, a representação NURBS é compacta. Foram obtidos resultados com número de pontos menor que os pontos de entrada e no entanto, pode-se perceber grande qualidade visual. Se a representação dos objetos em NURBS fosse feita utilizando-se polígonos, utilizando a mesma qualidade, seriam necessário muitos pontos.

Req. 2 - Qualidade da imagem.

Pelo fato das imagens serem representadas por curvas e não por polígonos, foi possível obter uma qualidade visual na representação dos objetos. É interessante também comentar que a representação NURBS apresenta-se ideal para modelagem de objetos arredondados, como formas anatômicas. Se o objeto a ser representado em NURBS possui quebras, então a representação tende a arredondar as quebras.

Req. 3 - Exatidão matemática.

Na metodologia proposta é utilizada a representação de um objeto baseada em funções B-Spline, caracterizando uma representação contínua do objeto. Também é calculado um fator de erro (distância Euclidiana entre os pontos de entrada e os pontos da superfície resultante) para verificação da fidelidade em relação aos dados originais. Pelo fato de se estar utilizando uma técnica de aproximação, pode-se observar que as curvas utilizadas na aproximação não passam precisamente por todos os pontos de entrada. A aproximação tende a "capturar" a forma do objeto, como apresentado na figura 6.3. Pode-se verificar a correspondência do objeto gerado em representação NURBS em relação ao objeto original em modelo poligonal, através do cálculo das distâncias dos pontos correspondentes nas duas representações.

6.5.1 Limitações

Como apresentado anteriormente, um objeto complexo aqui é composto por um conjunto de superfícies mais simples. As superfícies são unidas seguindo a continuidade do tipo C^0, posicional, apresentada no capítulo 3. A princípio o método não garante o tipo de continuidade C^1 ou C^2 entre segmentos.

Outra limitação deste método é a falta de garantia de convergência considerando objetos complexos, ou seja, a sensibilidade em relação aos dados de entrada. Não é possível prever de antemão se o processo retornará uma superfície adequada, se os dados de entrada caracterizarem uma forma muito complexa. Recomenda-se então a subdivisão do objeto.

O método aqui proposto utiliza a manipulação da quantidade de pontos de controle no processo iterativo de ajuste. A princípio o grau das curvas nas direções u e v e os pesos são considerados fixos.

No próximo capítulo são apresentados resultados obtidos com dados experimentais utilizando o protótipo desenvolvido seguindo a metodologia descrita neste capítulo.

7 Avaliação Experimental

7.1 Introdução

Utilizando o método apresentado no capítulo 6, foram elaborados testes utilizando o protótipo desenvolvido. Este capítulo apresenta alguns resultados que puderam ser obtidos utilizando ajuste de superfícies e a posterior geração de representação em VRML para a visualização das imagens.

7.2 Implementação

Os estudos que substanciaram os capítulos anteriores, serviram como subsídio para a implementação de um protótipo inicial para avaliação da metodologia.

Um dos requisitos deste trabalho é a facilidade de integração ao sistema Cyclops. Dessa forma, o protótipo foi desenvolvido utilizando a linguagem orientada a objetos Smalltalk, por ser a linguagem utilizada no projeto. Foi implementada uma categoria de classes NURBS, que contém as classes necessárias para os cálculos, bem como a exportação para VRML.

Além disso, a linguagem Smalltalk é conhecida no projeto, podendo o protótipo ser atualizado por outros membros do grupo, desde que conheça a teoria utilizada para a implementação. Foi utilizado o ambiente Visualworks 3. A existência de versões de máquinas virtuais Smalltalk para diversas plataformas propicia a portabilidade do programa desenvolvido.

Foi utilizada a plataforma Windows para o desenvolvimento dos protótipo devido a utilização do *plug-in* Blaxxun para visualização VRML ser desenvolvido para esta plataforma, mas este fato não implica na perda de utilidade do código desenvolvido aqui devido a portabilidade do ambiente Smalltalk Visualworks para outras plataformas. A escolha do *plug-in* VRML que suporte NURBS é livre.

No anexo 1 seguem fragmentos de códigos Smalltalk caracterizando a

implementação dos métodos utilizados neste trabalho.

7.3 Conjuntos de dados

Foram utilizados dados experimentais de objetos geométricos simples e dados fornecidos pelo sistema Cyclops correspondendo a formas anatômicas, no caso, representações de artérias. Os dados, caracterizados por conjuntos de pontos tridimensionais, foram dispostos em forma matricial de tamanho até 12x12 pontos. O código a seguir apresenta um exemplo de dados utilizados como entrada do sistema para o cálculo do ajuste.

```
S 1
1 5
c 5
{
138 150 1500 154 155 1500 159 167 1500 163 175 1500 158 181 1500
129 158 1500 134 151 1500 139 149 1485.0 156 155 1485.0 163 161 1485.0
130 178 1485.0 125 173 1485.0 126 163 1485.0 131 153 1485.0 142 148 1470.0
145 186 1470.0 135 184 1470.0 130 178 1470.0 126 169 1470.0 126 161 1470.0
161 179 1455.0 153 184 1455.0 143 185 1455.0 134 180 1455.0 127 174 1455.0
}
```

Listagem 7.1: Conjunto de dados de entrada para o ajuste de superfícies

Neste caso, S representa um identificador para a superfície, já que o objeto pode ser composto por um conjunto de várias superfícies, formando uma "colcha de retalhos". Os parâmetros l e c indicam o número de linhas e colunas da matriz, respectivamente. Estes valores caracterizam os parâmetros r e s utilizados no processo de ajuste.

Os valores da matriz representam os valores das coordenadas dos pontos 3D. Por exemplo, o primeiro, segundo e terceiro números da primeira linha da matriz definem a coordenada x, y e z do primeiro ponto da primeira linha da matriz e assim, sucessivamente.

7.4 Resultados Experimentais

Devido ao fato de se desejar uma determinada precisão em relação à malha de pontos original, pode-se perceber que os melhores resultados fornecidos pelo método utilizaram um número de pontos de controle na ordem de r-1, para matrizes de até 12x12 pontos. Apesar do número de pontos não ser muito reduzido, perde-se precisão quando quando tenta-se aproximar um objeto que já é composto por poucos pontos por outro objeto com muito menos pontos.

A seguir são apresentados alguns resultados obtidos com o protótipo.

7.4.1 Resultado 1

Como modelo para a primeiro teste, foi elaborado um objeto VRML (versão padrão) de forma simples, utilizando uma estrutura poligonal chamada IndexedFaceSet. Os pontos deste objeto serviram como estrada para o processo de ajuste. A figura 7.1 mostra o modelo utilizado em modo de visualização estrutura *wireframe*, fornecido pelo *plug-in* VRML e visualização das faces que formam o objeto.

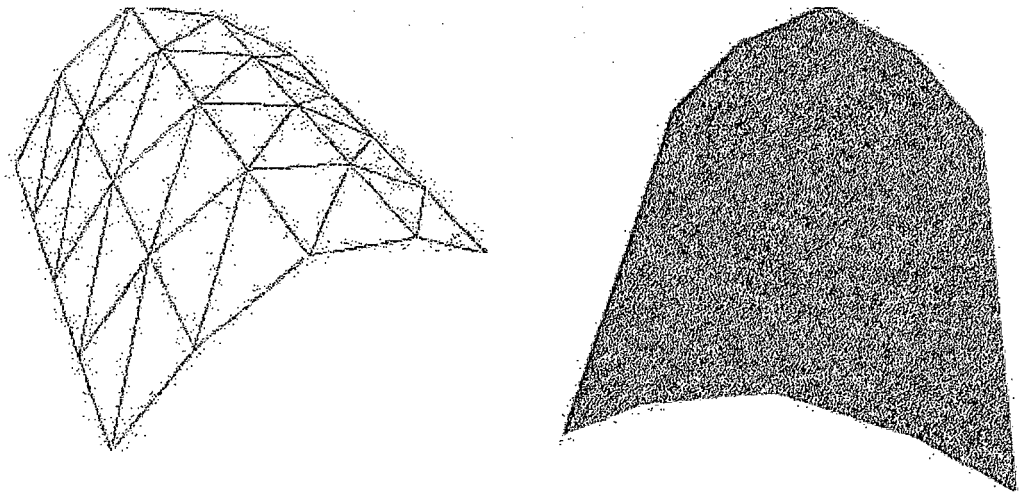


Figura 7.1: Modelo 1 em wireframe e visualização das faces

O modelo constitui-se de uma malha composta por 5x5 pontos tridimensionais (r=4, s=4) e os seguintes valores definem cada ponto da malha poligonal:

9 0 1, 9 2 3, 9 3 5, 9 2 7, 9 0 9
7 0 1 ,7 2 3, 7 3 5 ,7 2 7 ,7 0 9
5 0 1 ,5 2 3, 5 3 5, 5 2 7, 5 0 9
3 0 1 ,3 2 3, 3 3 5, 3 2 7, 3 0 9
1 0 1 ,1 2 3, 1 3 5, 1 2 7, 1 0 9

E como parâmetros de entrada para o processamentos foram fornecidos os seguintes dados:

1. Aproximação utilizando o mínimo de pontos da malha de pontos de controle como sendo n=2 e m=2 , ou seja uma matriz 3x3;

2. A ordem das curvas é 3 (grau 2) para as direções u e v;

3. Assume-se que os pesos valem 1.

Como resultado do ajuste obteve-se os seguintes vetores de nós para as direções u e v respectivamente:

uKnot [0.0 0.0 0.0 0.375 1.0 1.0 1.0]

vKnot [0.0 0.0 0.0 0.487025 1.0 1.0 1.0]

E por fim teve-se como resultado o seguinte conjunto definindo uma matriz 4x4 de pontos de controle:

(9.0 0.0 1.0) (9.0 1.83553 2.5) (9.0 4.04605 6.5) (9.0 0.0 9.0)
(7.77734 0.0 1.0) (7.77734 1.83553 2.5)(7.77734 4.04605 6.5) (7.77734 0.0 9.0)
(4.27777 0.0 1.0) (4.27777 1.83553 2.5) (4.27777 4.04605 6.5) (4.27777 0.0 9.0)
(1.0 0.0 1.0) (1.0 1.83553 2.5) (1.0 4.04605 6.5) (1.0 0.0 9.0)

Visualmente pode-se perceber uma grande diferença em temos de suavização da superfície. É apresentada a superfície em modo wireframe, devido a utilização de do fator de *tesselation* utilizado pela API 3D do *plug-in* Blaxxun (aproximação por polígonos para a renderização), no entanto a superfície é definida matematicamente. Em relação ao tamanho dos arquivos não obteve-se uma diferença significativa, já que a diferença entre

o número de pontos utilizado no modelo original e no resultado é muito pequena.

O tamanho dos arquivos VRML é 884 bytes para a representação NURBS e 1.148 bytes para a representação em polígonos IndexedFaceSet. A diferença de tamanho nos arquivos não é muito significativa, mas a medida em que há redução no número de pontos utilizada para representar a supefície em termos de curvas, há uma redução considerável no tamanho do arquivo. Se no modelo original, fosse desejada a mesma qualidade do objeto em NURBS, seriam necessários mais pontos e assim, seria visível a diferença de tamanho nos arquivos. Como exemplo, pode-se citar um objeto fornecido pela Blaxxun, em que os dois objetos possuem qualidade visual similar. O arquivo representando o objeto em NURBS tem tamanho de 11 Kbytes e o arquivo em polígonos tem o tamanho de 179 Kbytes.

A imagem 7.2 apresenta o resultado em visualização VRML na superfície NURBS gerada como resultado do processo de ajuste, utilizando um *plug-in* para visualização VRML que suporta o modelo baseado em curvas.

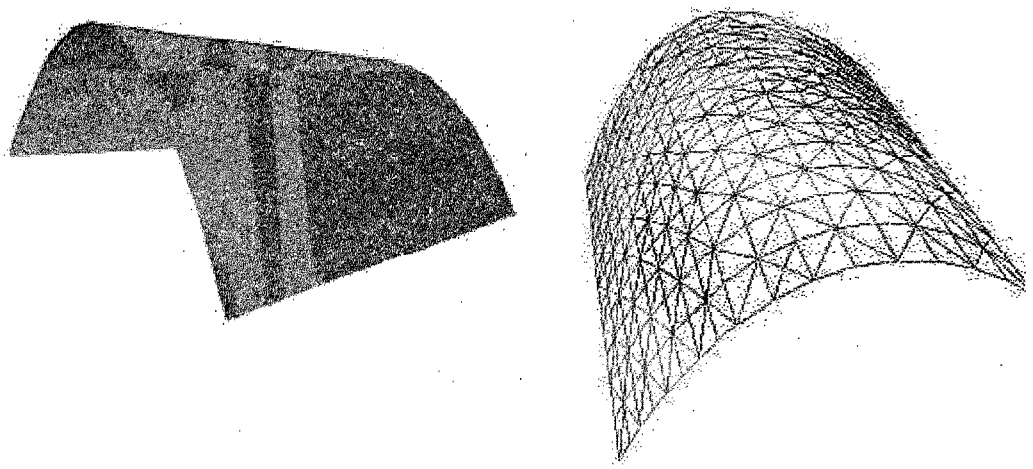


Figura 7.2: Resultado 1 em NURBS Surface, vista no VRML

No processo iterativo foi obtida a seguinte matriz de distâncias:

Distâncias da Matriz: 1.0

0.0d 0.23623858122625d 0.061024596159077d 2.8284271247462d 0.0d

0.18013835780404d 0.2970833135875d 0.19019418836735d 2.834157692852d 0.18013835780404d

0.056569081112d 0.24291712207605d 0.083210950447041d 2.8289927643842d 0.05656908d
2.0d 2.0139038376397d 2.0009307837445d 3.4641016151378d 2.0d 0.0d 0.23623858122625d
0.061024596159077d 2.8284271247462d 0.0d

Após o processo de ajuste, foi gerado um arquivo VRML representando a superfície NURBS. Este código é mostrado a seguir, na listagem 7.1:

```
#VRML V2.0 utf8
NavigationInfo {
type ["EXAMINE","ANY"] }
Transform {
children [Shape{
appearance Appearance {
            material Material {
diffuseColor 0.8 0.8 0.8
specularColor 1 1 1
emissiveColor 0 0 0
}
}
geometry NurbsSurface { # Superfície 1.0
ccw FALSE
solid FALSE
uOrder  3
vOrder 3
uDimension 4
vDimension 4
uKnot [ 0.0 0.0 0.0 0.375 1.0 1.0 1.0  ]
vKnot [ 0.0 0.0 0.0 0.487025 1.0 1.0 1.0  ]
controlPoint[
9.0 0.0 1.0 9.0 1.83553 2.5 9.0 4.04605 6.5 9.0 0.0 9.0
7.77734 0.0 1.0 7.77734 1.83553 2.5 7.77734 4.04605 6.5 7.77734 0.0
9.0
4.27777 0.0 1.0 4.27777 1.83553 2.5 4.27777 4.04605 6.5 4.27777 0.0
9.0
1.0 0.0 1.0 1.0 1.83553 2.5 1.0 4.04605 6.5 1.0 0.0 9.0
 ]
```

```
weight[
1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0
]
}
}] }
```

Listagem 7.1: Superfície NURBS em VRML

7.4.2 Resultado 2

Neste teste foi utilizado um conjunto de pontos cuja secção reta é representada pela figura 7.3, representando os pontos do plano xy.

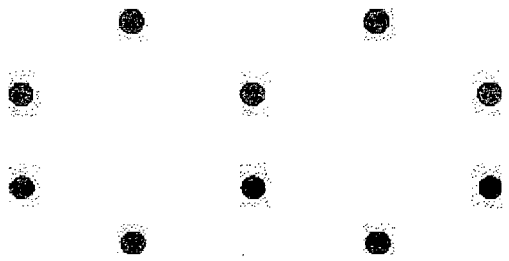


Figura 7.3: Disposição dos pontos no plano xy

Pode-se observar a suavização das quebras presentes na estrutura do objeto original, pela formação das curvas. O objeto final possui uma secção em forma de "oito".

A figura 7.4 e a figura 7.5 apresentam a visualização do objeto final no modo *wireframe* e visualização das faces, respectivamente.

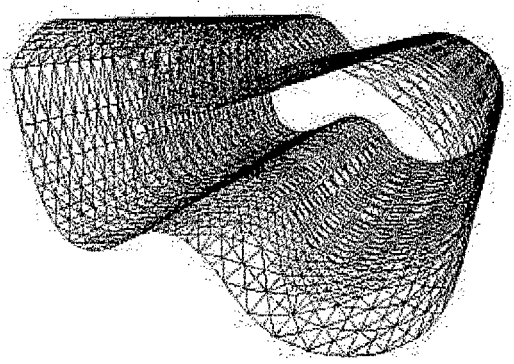


Figura 7.4: Objeto em modo *wireframe*.

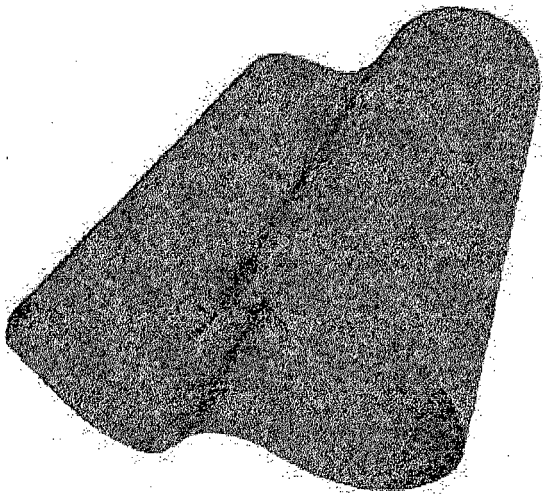


Figura 7.5 Visualização das faces

O anexo 2 traz o código VRML gerado, bem como a matriz de pontos de entrada e a matriz de distâncias.

7.4.3 Resultado 3

Como modelo para este teste, foram utilizados dados fornecido pelo systema Cyclops. O objeto corresponde a uma estrutura parcial de uma artéria, definida por uma matriz 10x10. A figura 7.6 mostra o modelo utilizado no formato poligonal representado pelo objeto VRML IndexedFaceSet, sendo caracterizado por um conjunto de faces triangulares. O arquivo VRML possui tamanho de 8.06 Kbytes.

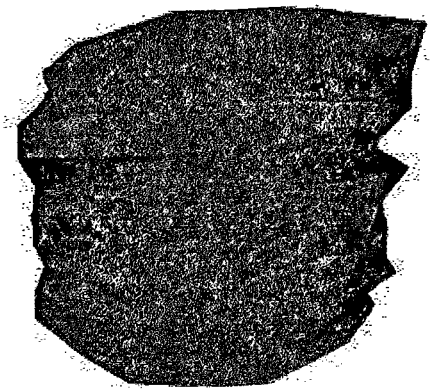


Figura 7.6: Segmento de artéria facetada.

A figura 7.7 apresenta o resultado do processo de ajuste do modelo apresentado na figura anterior. A malha resultante de pontos de controle caracterizam uma matriz 9x9.

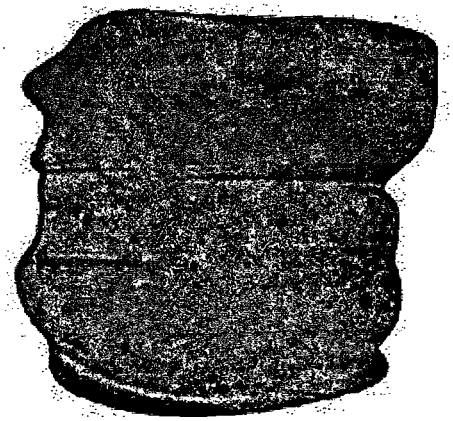


Figura 7.7: Estrutura em NURBS

Foi observada uma pequena redução no tamanho do arquivo VRML, para 3.02 Kbytes. Como matriz de distâncias obteve-se o seguinte resultado apresentado na figura 7.8. Este gráfico representa as regiões da matriz de distâncias classificadas por seus intervalos de valores apresentados na legenda ao lado. A maior parte da matriz apresentou valores entre 0 e 5. As linhas da matriz são numeradas no gráfico, de 1 a 10 e as colunas de S1 a S10.

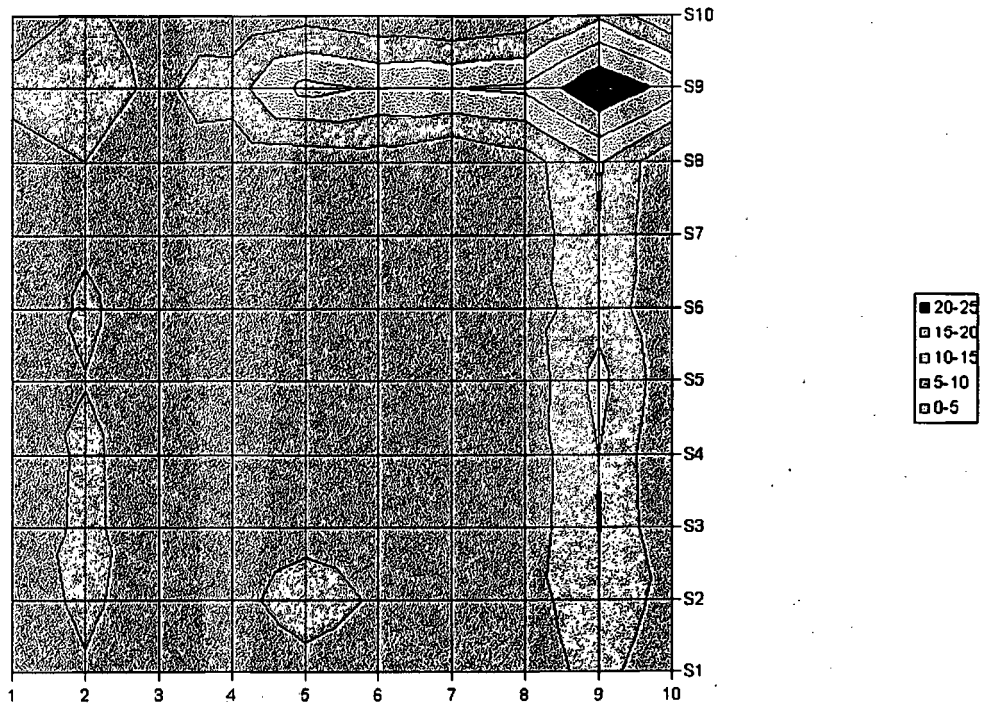


Figura 7.8: Matriz de Distâncias do Resultado 3

7.3.4 Resultado 4

Neste teste foram utilizados dados de um segmento de uma artéria representada pela figura 7.8 (completa). Os dados de entrada definem uma matriz 10x10 pontos. O tamanho total do segmento de artéria utilizado no ajuste possui tamanho 9 Kbytes, sendo que o arquivo VRML representando o segmento completo da artéria possui 133 Kbytes.

Figura 7.9: Artéria facetada

Foi utilizado o processo iterativo, com inicialmente um número mínimo de pontos de controle, com n=6 e m=6. Como resultado obteve-se a superfície representada na figura 7.10.

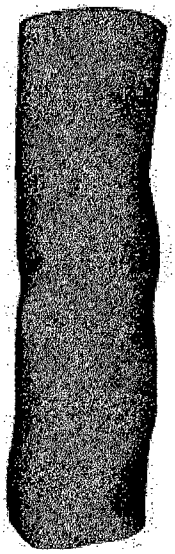


Figura 7.10: Resultado do ajuste de um segmento da artéria mostrada na figura 7.9.

Foram utilizadas curvas de grau 2 nas direções u e v e os pesos dos pontos de controle foram definidos como 1. O tamanho do arquivo VRML resultante possui 3.01 Kbytes. A seguir é mostrada na figura 7.11, a matriz de distâncias, de tamanho 10x10,

resultante do processo de ajuste. Este gráfico representa as regiões da matriz de distâncias classificadas por seus intervalos de valores apresentados na legenda ao lado. A maior parte da matriz apresentou valores entre 0 e 5. As linhas da matriz são numeradas no gráfico, de 1 a 10 e as colunas de S1 a S10.

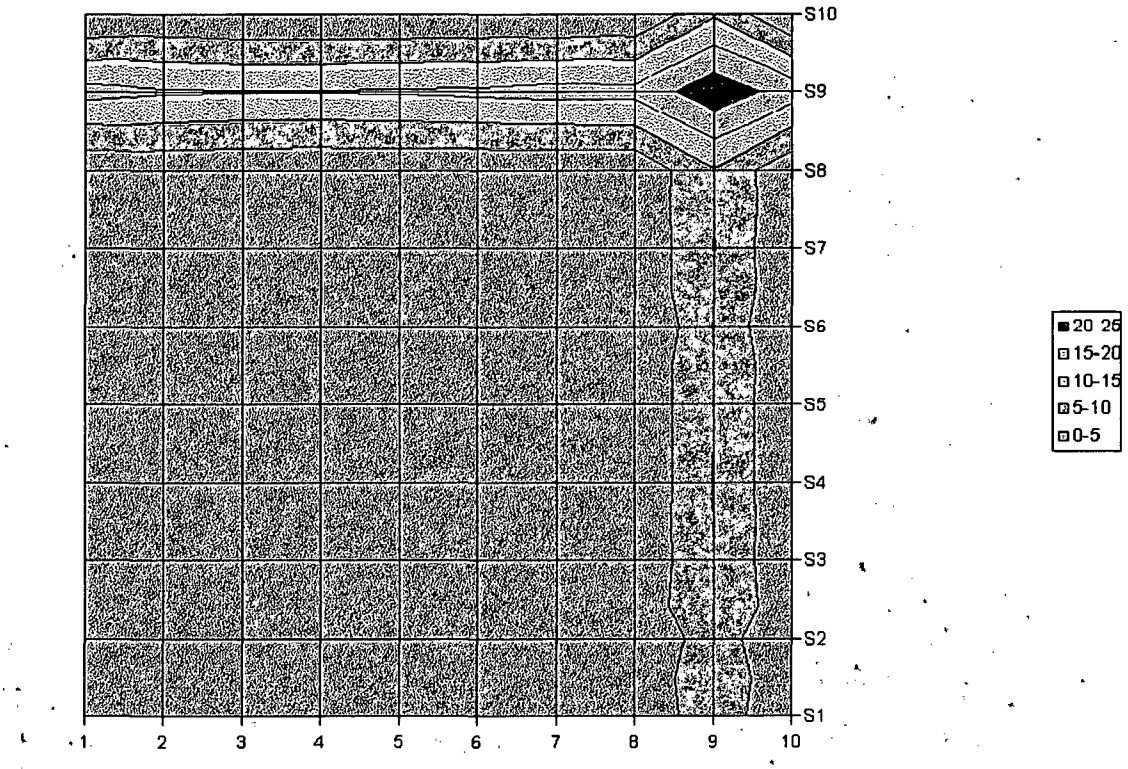


Figura 7.11: Matriz de Distâncias do Resultado 4.

Pode-se perceber neste exemplo, que o método obteve uma boa aproximação dos pontos e que seria necessário um ajuste local na região próxima a linha 9, coluna 9 da matriz de pontos de entrada.

8 Considerações Finais

Este capítulo descreve as principais conclusões tiradas no desenvolvimento desta pesquisa. São apresentadas as considerações em relação a metodologia utilizada e em relação a utilização do VRML. São apresentadas adicionalmente, sugestões de trabalhos futuros e a lista de publicações.

Em relação a utilização de NURBS neste trabalho, percebe-se que o modelo possui muitas vantagens como:

- Utilização de um mínimo de pontos na modelagem;
- Pode-se criar animações modificando poucos pontos;
- Atualmente a modelagem NURBS tem tido bastante aplicabilidade e diversos softwares e APIs tem utilizado modelagem baseada em curvas;
- Escalável à performance da CPU (LOD), pois pode-se utilizar um número fixo de subdivisões na renderização da superfície, ou um número adaptado à performance;
- Precisão geométrica, já que as superfícies são definidas em termos matemáticos;
- Qualidade das imagens quando se tem superfícies mais suaves;
- Arquivos pequenos (menores do que modelos usando polígonos), quando se utiliza o formato VRML para representação;
- Pode-se perceber também que a modelagem NURBS tem grande utilidade na representação de formas anatômicas devido ao fato de que as superfícies por sua natureza são mais arredondadas, caracterizando muitas formas que são encontradas na natureza.

No entanto, apesar das inúmeras vantagens, tem-se muito mais complexidade na representação baseada em curvas. É necessário conhecimento mais aprofundado sobre a modelagem para a construção de superfícies e conhecimentos avançados para a realização de ajuste, considerando um conjunto de dados específico. Muitas técnicas envolvendo modelagem NURBS estão também em fase de teste e não foi encontrado um padrão que possa ser adotado, considerando o escopo de aplicação deste trabalho.

A modelagem NURBS utiliza um conjunto de parâmetros para a caracterização das curvas e superfícies como apresentado no capítulo 2. Pode-se utilizar, além da localização

e da quantidade de pontos de controle, para a alteração da forma da superfície, a alteração dos pesos sobre os pontos de controle, a alteração da multiplicidade de pontos de controle e da multiplicidade do nós, e a alteração do grau das curvas, caracterizando uma fase adicional de processamento, chamada **refinamento**.

O trabalho aqui proposto é o processo de geração de modelagem NURBS, ou seja, os parâmetros para representação em curvas, a partir de uma modelagem poligonal. Esta fase caracteriza-se por encontrar um conjunto de parâmetros NURBS adequado, baseado em um número de pontos de controle e graus das curvas fornecidos pelo usuário, utilizando pesos valendo 1, para uma superfície específica e posteriormente a geração da representação em VRML para visualização através da Internet. Os parâmetros que são gerados como resultado do processo de ajuste são um conjunto de pontos de controle e os vetores de nós e assume-se que os pesos dos pontos de controle valem 1.

Desta forma a metodologia aqui proposta não pode garantir que sempre serão representadas facilmente através dos pontos de controle qualquer tipo de superfície, sem um processamento adicional de adaptação dos parâmetros NURBS. Além da importância e complexidade do processo de refinamento, necessário muitas vezes como etapa adicional ao ajuste, não foram tratatos neste trabalho dados relativos a superfícies de revolução, superfícies *Trimmed* (por exemplo, formas que possuem furos) e superfícies com quebras (quinas) ou bifurcações, devido ao fato de ser necessário etapas adicionais para a alteração da continuidade das curvas.

Os parâmetros NURBS gerados como resultado podem ser utilizados por qualquer API que suporte o modelagem NURBS. Foram utilizados aqui os *plug-ins* para VRML desenvolvidos pela Blaxxun e pela Parallel Graphics, devido ao requisito do trabalho de que as imagens fossem representadas em VRML. Não foram encontrados nesta pesquisa, outros *plug-ins* VRML que utilizassem o modelo NURBS e este trabalho considerou o Cortona, da Parallel Graphics como melhor opção entre os *plug-ins* utilizados, por apresentar uma interface amigável ao usuário e fornecer opções adicionais relativas e renderização do objeto e por apresentar melhor performance. No entanto, a escolha do *plug-in* é considerada livre, para o usuário final.

Como sugestão de atividades complementares para este trabalho são apresentadas algumas etapas de trabalhos futuros na seção a seguir.

8.1 Trabalhos Futuros

Uma etapa considerada importante ao processo de ajuste é a escolha dos pontos de uma superfície considerada como entrada do sistema. Foram utilizados de forma experimental malhas escolhidas aleatoriamente e considerou-se necessário um processamento adicional na fase de escolha das malhas a serem ajustadas, considerando critérios como por exemplo, similaridade no comportamento dos pontos ou curvatura, quando na adaptação do método de ajuste utilizado neste trabalho. Os resultados obtidos neste trabalho apresentaram continuidade C^0 e seria interessante um método de divisão da superfície a ser ajustada considerando as regiões onde se deseja obter continuidade C^0.

Adicionalmente, pode-se sugerir como atividade complementar a esta pesquisa, a utilização do processo de refinamento, aplicado à malha resultante do processo de ajuste, utilizando a matriz de distâncias, de modo a minimizar seus valores considerados altos. É necessário conhecimento aprofundado a respeito da relação dos parâmetros NURBS entre si e em relação à superfície.

8.2 Publicações

Simpósio Catarinense de Processamento Digital de Imagens. 1a.Ed. Florianópolis, 2002. ISBN: 85-902627-1-5. Uma Metodologia para Representação de Estruturas Anátomo-Patológicas Utilizando NURBS. pgs, 64-68. Aline da Rosa Alves e Prof. Dr. rer. nat. Aldo von Wangenheim.

9. Referências Bibliográficas

CHENEY, E. W.; KINCAID, D. K. **Numerical Mathematics and Computing**. USA: Brooks/Cole Publ. Co.,1994.

FARIN, Gerald E. **Curves and Surfaces for Computer-Aided Geometric Design**. USA: Academic Press, Inc., 1997.

FOLEY, James D.; VAN DAM, Andries; FEINER, Steven K.; HUGHES, John F. **Computer Graphics: Principles and Practice**. USA: Addison Wesley, 1997. 2ª Ed.

FORSEY, D. ;WONG, D. **Multiresolution Surface Reconstruction for Hierarchical B-Splines**. USA: University of British Columbia, 1995. Disponível em: <http://www.cs.ubc.ca/spider/forsey/Approx/App_1.html> Acesso em junho de 2002.

LAMMERS, Jim; GOODING, Lee. **Maya 4 Fundamentals**. USA: New Riders, 2002.

KRISHNAMURTHY, Venkat; LEVOY, Marc. **Fitting Smooth Surfaces to Dense Polygon Meshes**. In: SIGGRAPH (1996) Proceedings... p. 313-324.

SEGARS, W. **Realistic Spline-Based Dynamic Heart Phantom**. In: IEEE Medical Imaging Conference (1998).

TYLLER, W; PIEGL, Les. **The NURBS Book**. USA: Springer, 1997. 2ª Ed.

ZORIN, Denis; Levin, A.; BIERMANN, H. **Piecewise Smooth Subdivision Surfaces With Normal Control**, in SIGGRAPH (2000) *Proceedings*... p. 113-120.

9.1 Referências Consultadas mas Não Referenciadas

ALTMANN, Makus. About Nonuniform Rational B-Splines - NURBS
Disponível em: http://cs.wpi.edu/~matt/courses/cs563/talks/nurbs.html
Acesso em: Junho de 2002.

Documentação VRML. Disponível em:
http://www.parallelgraphics.com/products/cortona/extensions/nurbs/
Acesso em: Junho de 2002.

Documentação VRML. Especificação VRML 2. Disponível em: www.vrml.org
Acesso em: Junho de 2002.

Anexo 1

Fragmentos de Código Smalltalk

```
"*************************************************************************"
```

globalSurfErrorBound:r s:s Q:Q p: p q:q n:n m:m P: P

```
| uk vl U V lu NTNu Nu L  NTNv Nv NuT Rux  Ruy   Ruz Rvx Rvy Rvz Tempx Tempy
Tempz bx by bz  knotsIndex lNu vknotsIndex NvT lNv Px Py Pz cPointsx cPointsy
cPointsz span funs N0p Nnp Rkx Rky Rkz linQ coliNu somax somay somaz colx coly
colz Rvkx Rvky Rvkz oP aN aNv Surface distance dTemp |

"Calcula os pontos de controle a partir do conjunto Q de pontos.
=====inicializacao============="

     Rkx:=Array new:(r-1).
     Rky:=Array new:(r-1).
     Rkz:=Array new:(r-1).
     Rvkx:=Array new:(s-1).
     Rvky:=Array new:(s-1).
     Rvkz:=Array new:(s-1).
     Rux:= Array new:(n-1).
     Ruy:= Array new:(n-1).
     Ruz:= Array new:(n-1).
     coliNu:= NArray new.
     colx:= NArray new.
     coly:= NArray new.
     colz:= NArray new.
     lu:= NDecomposicaoLU new.
     uk:= NArray new: (s+1).
     vl:= NArray new: (r+1).
      Nu:= NMatriz new:(r-1).
     Nu n: (r-1).
     Nu m: (n-1).
     lNu:= NArray new:(n-1).
     0 to: (r-2) do:[:i|
        Nu at: i put: lNu copy.
     ].

     aNv := NMatriz new:(s+1).
```

```
 0 to: (s) do:[:i|
   aNv at: i put: ((NArray new:(m+1))copy).
].
 aNv n: (s+1).
aNv m: (m+1) .
aN := NMatriz new:(r+1).
 0 to: (r) do:[:i|
   aN at: i put: ((NArray new:(n+1))copy).
].
aN n: (r+1).
aN m: (n+1) .
Nv :=  NMatriz new:(s-1).
Nv n: (s-1).
Nv m: (m-1).
lNv:= NArray new:(m-1).
0 to: (s-2) do:[:i|
   Nv at: i put: lNv copy.
].
U:= NArray new.
V:= NArray new.
NTNu:=NMatriz new.
NTNv:= NMatriz new.
Rux:= NArray new:(n-1).
Ruy:= NArray new:(n-1).
Ruz:= NArray new:(n-1).
Rvx:= NArray new:(m-1).
Rvy:= NArray new:(m-1).
Rvz:= NArray new: (m-1).
bx:= NArray new.
by:= NArray new.
bz:= NArray new.

Tempx:= NMatriz new:(s+1).
0 to:( s )do:[:lin|
   Tempx at: lin put: ((NArray new: (n+1))copy).
].
Tempx n:(s+1).
Tempx m:(n+1).
Tempy:= NMatriz new:(s+1).
Tempy n:(s+1).
Tempy m:(n+1).
```

```
      0 to:( s )do:[:lin|
         Tempy at: lin put: ((NArray new: (n+1))copy).
      ].

      Tempz:= NMatriz new:(s+1).
      0 to:( s)do:[:lin|
         Tempz at: lin put: ((NArray new: (n+1))copy).
      ].
      Tempz n:(s+1).
      Tempz m:(n+1).
      Px:= NMatriz new: (n+1).
      0 to:( n )do:[:lin|
         Px at: lin put: ((NArray new: (m+1))copy).
      ].
      Py:= NMatriz new: (n+1).
      0 to:( n )do:[:lin|
         Py at: lin put: ((NArray new: (m+1))copy).
      ].
      Pz:= NMatriz new: (n+1).
      0 to:( n )do:[:lin|
         Pz at: lin put: ((NArray new: (m+1))copy).
      ].

      cPointsx:= NArray new.
      cPointsy:= NArray new.
      cPointsz:= NArray new.

"== fim inicializacao==="

      "Calculo dos parametros uk e vl"
       self surfParamsR:s S:r Q:Q Ub:uk Vb: vl.
      U:= self computeKnotsU:uk comN:n eM:r eP:p.
      V:= self  computeKnotsU:vl comN:m eM:s eP:p.
      knotsIndex:= (U size)-1.

"Para garantir convergencia"
      uk at: (r-1) put: 1.0d.
      vl at: (s-1) put: 1.0d.

"Calculo de Nu"
```

```
0 to: r do:[:i|
      0 to:(n) do:[:j|
         aN at: (i) at:(j) put:0.0
      ].
   ].
Nu n: (r-1).
Nu m: (n-1).
aN at:0 at:0 put: 1.0.
aN at: (r) at: (n) put: 1.0.
0 to: (r) do:[:i|
        span:= self findSpanN:(knotsIndex-p-1) eP:p eUk:(uk at:i) eU:U.
        funs:= self basisFunsSpan:span eUk:(uk at:i) eP:p eU:U.
         0 to: p do:[:j |
           aN at: (i) at:(span-p + j)  put: (funs at:j).
   ] .
].

1 to: (r-1) do:[:i|
     1 to: (n-1) do:[:j|
   Nu at:(i-I) at:(j-1) put: ((aN at:i at:j)copy) .
     ].
   ].

NuT:= self transposeNMatriz: Nu .
NTNu:= NuT*Nu.
L :=lu elimGauss: NTNu.

"Ajuste na direcao u"
0 to: s do:[:j |
Tempx at:j at:0put:(( Q at:j at:0 )x).
Tempx at: j at:n put: ((Q at: j at:r )x).
Tempy at:j at:0 put:(( Q at:j at:0 )y).
Tempy at: j at:n put: ((Q at: j at:r )y).
Tempz at:j at:0 put:(( Q at:j at:0 )z).
Tempz at: j at:n put: ((Q at: j at:r )z).

linQ:= Q retornaLinha: j.
"Calculo de Rk"
1 to: (r-1) do:[:i|
   N0p:=self oneBasisFunP:p comM:knotsIndex comI:0 comU:U comuk: (uk at:i).
   Nnp:=self oneBasisFunP:p comM:knotsIndex comI:n comU:U comuk: (uk at:i).
```

```
          Rkx at:i put: ((linQ at:i  )x) -(((linQ at:0 )x)*(N0p)) - (((linQ at:
r  )x)*(Nnp )).
          Rky at:i put: ((linQ at:i  )y) -(((linQ at:0 )y)*(N0p)) - (((linQ at:
r  )y)*(Nnp )).
          Rkz at:i put: ((linQ at:i  )z) -(((linQ at:0 )z)*(N0p)) - (((linQ at:
r  )z)*(Nnp )).
        ].
      "Calculo de R"
      1 to: (n -1) do:[:col|
        coliNu:= Nu retornaColuna: (col-1).
        somax:=0.
        somay:=0.
        somaz:=0.
        1 to: (r -1) do:[:j|
             somax:= somax+((Rkx at:j)*(coliNu at:(j-1))).
             somay:= somay+((Rky at:j)*(coliNu at:(j-1))).
             somaz:= somaz+((Rkz at:j)*(coliNu at:(j-1))).
        ].
      Rux at:(col-1) put: somax.
      Ruy at:(col-1) put: somay.
      Ruz at:(col-1) put: somaz.
      ].
      "Decomposicao Lu, para resolucao do sistema"
      bx:= lu substituicaoDireta: NTNu com: Rux eOrdenamento: L.
      by:= lu substituicaoDireta: NTNu com: Ruy eOrdenamento: L.
      bz:= lu substituicaoDireta: NTNu com: Ruz eOrdenamento: L.
      cPointsx:=    lu    substituicaoInversaMatriz:    NTNu    comVetorZ:    bx
comOrdenamento:L.
      cPointsy:=    lu    substituicaoInversaMatriz:    NTNu    comVetorZ:    by
comOrdenamento:L.
      cPointsz:=    lu    substituicaoInversaMatriz:    NTNu    comVetorZ:    bz
comOrdenamento:L.
      "Armazena resultado em matriz temporaria, Temp"
      1 to: n-1 do:[:k|
        Tempx at: j at: k put: (cPointsx at: (k-1) ).
        Tempy at: j at: k put: (cPointsy at: (k-1) ).
        Tempz at: j at: k put: (cPointsz at: (k-1) ).
       ].
].
"==================Ajuste na direcao v======================="
```

```
"Calculo de Nv"
vknotsIndex:= (V size)-1.
0 to: s do:[:i|
        0 to:(m) do:[:j|
          aNv at: (i) at:(j) put:0.0
        ].
     ].
0 to: (s) do:[:i|
     span:= self findSpanN:(vknotsIndex-q-1) eP:q eUk:(vl at:i) eU:V.
     funs:= self basisFunsSpan:span eUk:(vl at:i) eP:q eU:V.
     0 to: p do:[:j |
        aNv at: (i) at:(span-q + j)  put: (funs at:j).
     ] .
].

1 to: (s-1) do:[:i|
     1 to: (m-1) do:[:j|
        Nv at:(i-1) at:(j-1) put: ((aNv at:i at:j)copy) .
     ].
].
     NvT:= self transposeNMatriz: Nv .
     NTNv:= NvT*Nv.
     L :=lu elimGauss: NTNv.

"========Ajuste na direcao v=================================="
     0 to: n do:[:i |
        Px at:0 at:i put: (Tempx at: 0 at: i).
        Px at:m at:i put:(Tempx at:s at:i) .
        Py at:0 at:i put: (Tempy at: 0 at:i).
        Py at:m at:i put:(Tempy at:s at:i) .
        Pz at:0 at:i put: (Tempz at:0 at: i).
        Pz at:m at:i put:(Tempz at:s at:i) .
        colx:=  Tempx retornaColuna: i.
        coly:=  Tempy retornaColuna: i.
        colz:=  Tempz retornaColuna: i.

     1 to: (s-1) do:[:i|
        N0p:=self  oneBasisFunP:q  comM:vknotsIndex  comI:0  comU:V  comuk:  (vl
at:i).
        Nnp:=self  oneBasisFunP:q  comM:vknotsIndex  comI:m  comU:V  comuk:  (vl
at:i).
```

```
          Rvkx at:i put: (colx at:i ) -((colx at:0)*(N0p)) - ((colx at: s )*(Nnp
)).
          Rvky at:i put: (coly at:i  ) -((coly at:0 )*(N0p)) - ((coly at: s
)*(Nnp )).
          Rvkz at:i put: (colz at:i  ) -((colz at:0 )*(N0p)) - ((colz at: s
)*(Nnp )).
        ].

     1 to: (m -1) do:[:col|
        coliNu:= Nv retornaColuna: (col-1).
        somax:=0.
        somay:=0.
        somaz:=0.
        1 to: (s -1) do:[:j|
             somax:= somax+((Rvkx at:j)*(coliNu at:(j-1))).
             somay:= somay+((Rvky at:j)*(coliNu at:(j-1))).
             somaz:= somaz+((Rvkz at:j)*(coliNu at:(j-1))).
        ].
     Rvx at:(col-1) put: somax.
     Rvy at:(col-1) put: somay.
     Rvz at:(col-1) put: somaz.
     ].
        bx:= lu substituicaoDireta: NTNv com: Rvx eOrdenamento: L.
        by:= lu substituicaoDireta: NTNv com: Rvy eOrdenamento: L.
        bz:= lu substituicaoDireta: NTNv com: Rvz eOrdenamento: L.
        cPointsx:=  lu  substituicaoInversaMatriz:  NTNv  comVetorZ:  bx
comOrdenamento:L.
        cPointsy:=  lu  substituicaoInversaMatriz:  NTNv  comVetorZ:  by
comOrdenamento:L.
        cPointsz:=  lu  substituicaoInversaMatriz:  NTNv  comVetorZ:  bz
comOrdenamento:L.
        1 to: m-1 do:[:k|
          Px at: k at: i put: (cPointsx at: (k-1) ).
          Py at:k at: i put: (cPointsy at: (k-1) ).
          Pz at: k at: i put: (cPointsz at: (k-1) ).
         ].
     ].
     oP:= self montaPontosPx: Px Py: Py Pz:Pz N:n M:m .
     self knotVectorU: U .
     self knotVectorV:V.
     self controlPoint: oP.
```

```
      self orderU:p+1.
      self orderV:q+1.
      self cN:n.
      self cM:m.
"==========Error Bound================="
      Surface := NMatriz new:(r+1).
      0 to:( r )do:[:lin|
         Surface at: lin put: ((NArray new: (s+1))copy).
      ].
      0 to: s do:[:i|
         0 to: r do:[:j|
         Surface at:i at:j put: (self surfacePointN:n P:p U:U M:m q:q V:V Points:
oP umUk: (uk at:i) umVl: (vl at:j)).
         ].
      ].
      self surfacePoints: Surface.
      distance:= 0.
      0 to: s do:[:i|
         0 to: r do:[:j|
            dTemp:= self distance3DP1: (Q at: i at: j ) P2: (Surface at:i at: j).
            (((dTemp) abs) > distance) ifTrue:[
            distance:= (dTemp) abs.
         ].
      ].
].    self maxError: distance.
      ^distance.
"*************************************************************************"
```

surfParamsR:n S:m Q:umQ Ub:uk Vb: vl

```
"Calcula os parametros para a superficie Entrada: n linhas,m colunas de Q,
Saida: uk, vl "

|num k l total cds d temp |
"============Calculo de uk===================="
cds:= NArray new:n.
num:= m+1.
uk at: 0 put: 0.0d.
uk at: (n) put: 1.0d.

      k:=1.
```

```
      [k<(n)]whileTrue:[
         uk at: k put: 0.0d.
      k:= k+1.
         ].
      l:=0.
      [l<=(m)] whileTrue:[
         total:=0.0d.
         k:=1.
         [k<=n]whileTrue:[
           cds at:(k-1) put: (self distance3DP1: (umQ at:l at:k) P2:( umQ at:l
at: (k-1)))asDouble.
           total:= total+ (cds at:(k-1)).
           k:=k+1.
         ].
         (total=0)ifTrue:[ num:= num-1]
             ifFalse:[
               d:=0.0d.
               k:=1.
               [k<(n)]whileTrue:[
               d:= (d+ (cds at:(k-1)))asDouble.
               uk at:k put: (( uk at: k) + (d/total) )asDouble.
               k:=k+1.
               ].
           ].
      l:= l+1.
      ].
      "se num=0 da erro"
      (num=0.0)ifFalse:[
         k:=1.
         [k<(n)]whileTrue:[
           temp:= (uk at:k).
           uk at:k put:((temp/num)asDouble) .
         k:=k+1.]
      ].
"============Fim do Calculo de uk====================
==========Calculo de vl==================="
cds:= NArray new:m.
num:= n.
vl at: 0 put: 0.0d.
vl at: (m) put: 1.0d.
      l:=1.
```

```
      [l<(m)]whileTrue:[
         vl at: l put: 0.0d.
      l:= l+1.
         ].
      k:=0.
      [k<=(n)] whileTrue:[
         total:=0.0.
         1 to: (m) do:[:l|
            cds at:(l-1) put: (self distance3DP1: (umQ at:l at:k) P2:( umQ at:(l-
1) at: k))asDouble.
            total:= total+ (cds at:(l-1)).
         ].
         (total=0)ifTrue:[ num:= num-1]
               ifFalse:[
                d:=0.0d.
                l:=1.
                [l<(m)]whileTrue:[
                d:= (d+ (cds at:(l-1)))asDouble.
                vl at:l put: ((vl at: l) + (d/total) )asDouble.
                l:=l+1.
                ].
                ].
      k:=k+1.
      ].
      "se num=0 da erro"
      (num=0.0)ifFalse:[
         l:=1.
         [l<(m)]whileTrue:[
            vl at:l put:(( (vl at:l)/num)asDouble).
            "vl at:(m-1) put:1.0."
         l:=l+1.]
      ].
"*****************************************************************************
*"
```

computeKnotsU:uk comN:n eM:m eP:p

```
"m- num pontos
  n- num pontos de controle
U tem r+1 knots
r=n+p+1"
```

```
| d i alpha U j |

U:= NArray new:(n+p+2).

      j:= 0.
      [j<=p] whileTrue:[
         U at: j put: 0.
         j:= j+1.
      ].
      j:= 1.
      [ j<=(n-p)]whileTrue:[

         U at: (j+p) put: 0.
         i:=j.
         [ i<=(j+p -1)]whileTrue:[
            U at:(j+p) put: ((U at:(j+p)) + (uk at:i)).
            i:=i+1.
         ].
         U at: (j+p) put: (( U at:(j+p))/ (p asFloat)).
         j:=j+1.
      ].
      j:=n+p+1.
      [ j>=(n+1)] whileTrue:[
           U at:j  put: 1.
      j:=j-1.
      ].
^U

"*****************************************************************************
*"
```

findSpanN:umN eP:umP eUk:umuk eU:umVetorU

```
"determine the knot span index"

| low high mid |
(umuk=(umVetorU at:(umN+1))) ifTrue:[^umN].

low:=umP.
high:=umN+1.
mid:=((low+high)/2)asInteger.
```

```
[(umuk<(umVetorU at:mid)) | (umuk>= (umVetorU at:(mid+1)))]whileTrue:[

(umuk<(umVetorU at:mid))ifTrue:[
            high:=mid.]
            ifFalse:[low:=mid.].
      mid:=((low+high)/2)asInteger.
].
^mid.
"*****************************************************************************"
```

oneBasisFunP:p comM:m comI:i comU:U comuk: u

```
"entrada: p,m,U,i,u - saida: Nip"
"--------------casos especiais-----------"
| Nip j Nt k saved Uleft Uright temp |

Nt:= NArray new:(p+1).
      (((i=0) & u=(U at:0)) | (i=(m-p-1) & ( u=(U at:m)))) ifTrue:[
      Nip:= 1.0 .
      ^ Nip
].
"propriedade local"
      ((u<(U at:i) )| (u>= (U at:(i+p+1)))) ifTrue:[
      Nip := 0.0.
      ^Nip
].
      j:=0.
      [j<=p]whileTrue:[
         ((u>=(U  at:(i+j)))&(u<(U  at:(i+j+1))))ifTrue:[Nt  at:j  put:1.0  .]
ifFalse:[ Nt at:j put:0.0.].
      j:=j+1.
      ].

      k:=1.
      [k<=p]whileTrue:[
      ((Nt at:0) =0) ifTrue:[
            saved:=0.0.]
            ifFalse:[
            saved:= ((u - (U at:i))*(Nt at:0))/((U at:(i+k)) - (U at:i)).
            ].
      j:=0.
```

```
      [j<(p-k+1)]whileTrue:[
         Uleft:= U at:(i+j+1).
         Uright:= U at:(i+j+k+1).
         ((Nt at:(j+1))=0.0) ifTrue:[
              Nt at:j put: saved.
              saved:= 0.0.
         ]
           ifFalse:[
              temp:=( Nt at:(j+1))/ (Uright-Uleft).
              Nt at: j put:(saved+ ((Uright - u)*temp)).
              saved := (u - Uleft)*temp.
         ].
         j:=j+1.
      ].
      k:= k+1.
      Nip := Nt at:0.
].
^Nip
"*****************************************************************************"
```

elimGauss:umaMatriz

```
"Decomposicao da matriz e LU"
| ma L smax s rmax r j temp xmult i |
      ma:=umaMatriz.
n:= ma n.
L:= List new.
L changeSizeTo: n.
s:= List new.
s changeSizeTo: n.
 "-------------------------------------------------------------"
      i:=1.
        [i<=n] whileTrue:[
           L at:i put: i.
           smax:=0.0d.
           j:=1.
           [j<=n] whileTrue:[
                smax := ((smax abs) max: (( ma at:(i-1) at:(j-1)) abs)) asDouble.
                j:=j+1.
           ].
           s at: i put: (smax asDouble).
```

```
            i:=i+1.
      ].
       i:= i-1. " pra ficar = n"
"================================="
1 to: n-1 do:[:k|
      rmax:=0.0d.
      k to: n do:[:i |
            r := ( ((ma at: ((L at:i)-1) at:(k-1))/ (s at:(L at:i)))asDouble )abs.
            (r>rmax) ifTrue:[ rmax:=r. j:=i.]
      ].

      temp:= L at: j.
      L at:j put: (L at:k).
      L at:k put: temp.

      k+1 to: n do:[:i |
         xmult:=( (ma at:((L at:i)-1) at:(k-1))/(ma at:((L at:k)-1) at:(k-1)
))asDouble.
         "xmult:=( (ma at:((L at:i)-1) at:(j-1))/(ma at:((L at:k)-1) at:(k-1)
))asDouble."

         (k+1) to:n do:[:j |
           ma at:((L at:i)-1) at:(j-1) put: (((ma at:((L at:i)-1) at:(j-1)) -
(xmult* (ma at:((L at:k) -1)at:(j-1)) )))asDouble.
         ].
         ma at:((L at:i)-1) at:(k-1) put: xmult.
      ].
].
^L.
"*****************************************************************************
*"
```

substituicaoDireta: A com: vetorB eOrdenamento: L

```
"Algoritmo 2 (Substituicao Direta: grava solucao de L z = b sobre o vetor b)
Lê n, a, b, l [a é n x n, b: vetor de Ax=b; l é o vetor ordenamento do
algoritmo 1]
para k = 1 até n-1, faça
    para i = k+1 até n, faça
        b(l(i)) = b(l(i)) - b(l(k))*a(l(i),k)
    fim(para)
```

```
fim(para)
"

| n k |
n:= A n.
k:=1.
[k<=(n-1)] whileTrue: [ |i|
              i:= k+1.
              [i<=n] whileTrue:[ |m|
              m:= (vetorB at:( (L at:i)-1))  -  ( (vetorB at: ((L at:k)-1)) *  (
A atLinha: ((L at:i)-1) atColuna:(k-1))) .
              vetorB at: ((L at:i)-1) put: m  .
              i:= i+1.
              ].
              k := k+1.
].
^vetorB
"*************************************************************************
*"
```

substituicaoInversaMatriz: matrizU comVetorZ:vetorZ comOrdenamento: L

```
"Algoritmo 3 (Substituicao Inversa: grava solucao de U x = z sobre o
vetor x)
Lê n, a, z, l [a é n x n, z: vetor de U x=z; l é o vetor ordenamento do
algoritmo 1]
x(n) = z(l(n))/a(l(n),n)
para i = n-1 até 1, faça
    soma = z(l(i))
    para j = i+1 até n, faça
        soma = soma - x(j)*a(l(i),j)
    fim(para)
    x(i) = soma/a(l(i),i)
fim(para)"

| n x soma i |
     n:= matrizU n.
     x:= NArray new:n.
     soma:=0d.
     x at:(n-1) put: (( vetorZ at: ((L at: n) -1))/ (matrizU atLinha:((L at: n)-
1) atColuna: (n-1))).
```

```
    i:=n-1.
    [i<1] whileFalse:[
        soma:= vetorZ at:((L at:i)-1).
        (i+1) to: n do:[:j|
          soma:= soma - ((x at:(j-1)) * ( matrizU atLinha:((L at: i)-1)
atColuna:(j-1))).
        ].
        x at:(i -1)put: (soma/(matrizU atLinha:((L at:i)-1) atColuna: (i-1))).
        i:=i-1.
    ].
    ^x.
"*****************************************************************************
*"
```

GenerateNURBS

```
"Aplicação 1 : Teste com surfacePoint e calculo do erro. Realiza 'passos'
iteracoes e imprime a supeficie com o menos erro no arquivo vrml. Em cada iteracao
aumenta o numero de pontos de controle
Imprime um arquivo com os erros em todos os pontos"

| fname su matrizes aMat str  r s p q n m  aFName P e eMin suTemp passos
matrixOfErrors dTemp strErr |

    fname:= 'Entrada'.
    su:= NurbsSurface new.
    suTemp:= NurbsSurface new.
    matrizes:= OrderedCollection new.
    matrizes:= su loadMatriz: fname.
    "parametros necessarios para a geracao da superficie: graus e numero de
pontos de controle"
    p:= 2.
    q:= 2.
    n:= 6.
    m:=6.

     aFName:= 'saida'.
    strErr := ((aFName, '.erb') asFilename) writeStream.
    str := ((aFName, '.wrl') asFilename) writeStream.

     str nextPutAll: '#VRML V2.0 utf8
    NavigationInfo {
```

```
            type ["EXAMINE","ANY"] }
      Transform {
         children ['.
         1 to: (matrizes size) do:[:i|
            aMat:= matrizes at:i.
            s:= (aMat s).
            r:= (aMat r).
            "assume q o primeiro eh o minimo, ja faz o passo 1"
            passos:= r-n-1.
            eMin:= su globalSurfErrorBound:r s:s Q:aMat p: p q:q n:n m:m P: P.
            1 to: passos do:[:k|
               n:= n+1.
               m:= m+1.
               suTemp:= su copy.
               e:= suTemp globalSurfErrorBound:r s:s Q:aMat p: p q:q n:n m:m P: P.
               (e<eMin)ifTrue:[
                  eMin:=e.
                  su:= suTemp.
            ].
         ].
         str nextPutAll:'# Superficie: '.
         (i asFloat) printOn: str.
         str cr.
         su printOn: str.
"===========Matriz de distancias==================="
      strErr nextPutAll:'Erro da Matriz:  '.
      (i asFloat)  printOn: strErr.
      strErr cr.
      matrixOfErrors:= NMatriz new: (r+1).
      0 to: r do:[:k| matrixOfErrors at:k put:((NArray new:(s+1)) copy)].
      matrixOfErrors n: (r+1).
      matrixOfErrors m: (s+1).
      0 to: s do:[:i|
      0 to: r do:[:j|
         dTemp:= su distance3DP1: (aMat at: i at: j ) P2: ((su surfacePoints) at:i
at: j).
         matrixOfErrors at: i at:j put: dTemp.

         ].
      ].
```

```
        matrixOfErrors printOn: strErr.
        strErr cr.

"================================="
        ].
        str nextPutAll:'] }'.
        str close.
        strErr close.

^matrixOfErrors.
```

Anexo 2

Dados de Entrada e Arquivos VRML

Teste 2: Superfície em em forma de 8

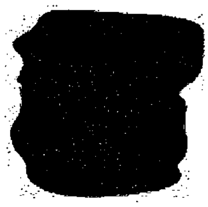
```
********************************* Dados de Entrada ******************************
S 1
l 11
c 11
{
1 4 0 3 5 0 5 4 0 7 5 0 9 4 0 9 2 0 7 1 0 5 2 0 3 1 0 1 2 0 1 4 0
1 4 1 3 5 1 5 4 1 7 5 1 9 4 1 9 2 1 7 1 1 5 2 1 3 1 1 1 2 1 1 4 1
1 4 2 3 5 2 5 4 2 7 5 2 9 4 2 9 2 2 7 1 2 5 2 2 3 1 2 1 2 2 1 4 2
1 4 3 3 5 3 5 4 3 7 5 3 9 4 3 9 2 3 7 1 3 5 2 3 3 1 3 1 2 3 1 4 3
1 4 4 3 5 4 5 4 4 7 5 4 9 4 4 9 2 4 7 1 4 5 2 4 3 1 4 1 2 4 1 4 4
1 4 5 3 5 5 5 4 5 7 5 5 9 4 5 9 2 5 7 1 5 5 2 5 3 1 5 1 2 5 1 4 5
1 4 6 3 5 6 5 4 6 7 5 6 9 4 6 9 2 6 7 1 6 5 2 6 3 1 6 1 2 6 1 4 6
1 4 7 3 5 7 5 4 7 7 5 7 9 4 7 9 2 7 7 1 7 5 2 7 3 1 7 1 2 7 1 4 7
1 4 8 3 5 8 5 4 8 7 5 8 9 4 8 9 2 8 7 1 8 5 2 8 3 1 8 1 2 8 1 4 8
1 4 9 3 5 9 5 4 9 7 5 9 9 4 9 9 2 9 7 1 9 5 2 9 3 1 9 1 2 9 1 4 9
1 4 10 3 5 10 5 4 10 7 5 10 9 4 10 9 2 10 7 1 10 5 2 10 3 1 10 1 2 10 1 4 10
}
 ************************** Arquivo VRML de Saída ***************************
#VRML V2.0 utf8
                NavigationInfo {
                type ["EXAMINE","ANY"] }
     Transform {
        children [Shape{
appearance Appearance {
            material Material {
                diffuseColor 0 0.2 1
                specularColor 1 1 1
                #emissiveColor 0.7 0.6 0.5
```

```
            }
          }

geometry NurbsSurface {
          ccw FALSE
          solid FALSE
          uOrder  3
vOrder 3
uDimension 10
vDimension 10
uKnot  [ 0.0  0.0  0.0  0.153235  0.255392  0.357549  0.454314  0.551078  0.653235
0.755392 1.0 1.0 1.0  ]
vKnot  [ 0.0 0.0 0.0 0.15 0.25 0.35 0.45 0.55 0.65 0.75 1.0 1.0 1.0  ]
controlPoint[ 1.0 4.0 0.0    2.49608 5.80215 0.0    5.00914 3.54499 0.0    6.94646
5.37934 0.0     9.24968 4.21064 0.0     9.25016 1.78913 0.0     6.94378 0.622549
0.0     5.02464 2.44394 0.0     1.43293 -0.492512 0.0     1.0 4.0 0.0
1.0 4.0 0.75     2.49608 5.80215 0.75     5.00914 3.54499 0.75     6.94646 5.37934
0.75     9.24968 4.21064 0.75     9.25016 1.78913 0.75     6.94378 0.622549 0.75
5.02464 2.44394 0.75     1.43293 -0.492512 0.75     1.0 4.0 0.75
1.0 4.0 2.0     2.49608 5.80215 2.0     5.00914 3.54499 2.0     6.94646 5.37934 2.0
9.24968 4.21064 2.0     9.25016 1.78913 2.0     6.94378 0.622549 2.0     5.02464
2.44394 2.0     1.43293 -0.492512 2.0     1.0 4.0 2.0
1.0 4.0 3.0     2.49608 5.80215 3.0     5.00914 3.54499 3.0     6.94646 5.37934 3.0
9.24968 4.21064 3.0     9.25016 1.78913 3.0     6.94378 0.622549 3.0     5.02464
2.44394 3.0     1.43293 -0.492512 3.0     1.0 4.0 3.0
1.0 4.0 4.0     2.49608 5.80215 4.0     5.00914 3.54499 4.0     6.94646 5.37934 4.0
9.24968 4.21064 4.0     9.25016 1.78913 4.0     6.94378 0.622549 4.0     5.02464
2.44394 4.0     1.43293 -0.492512 4.0     1.0 4.0 4.0
1.0 4.0 5.0     2.49608 5.80215 5.0     5.00914 3.54499 5.0     6.94646 5.37934 5.0
9.24968 4.21064 5.0     9.25016 1.78913 5.0     6.94378 0.622549 5.0     5.02464
2.44394 5.0     1.43293 -0.492512 5.0     1.0 4.0 5.0
1.0 4.0 6.0     2.49608 5.80215 6.0     5.00914 3.54499 6.0     6.94646 5.37934 6.0
9.24968 4.21064 6.0     9.25016 1.78913 6.0     6.94378 0.622549 6.0     5.02464
2.44394 6.0     1.43293 -0.492512 6.0     1.0 4.0 6.0
1.0 4.0 7.0     2.49608 5.80215 7.0     5.00914 3.54499 7.0     6.94646 5.37934 7.0
9.24968 4.21064 7.0     9.25016 1.78913 7.0     6.94378 0.622549 7.0     5.02464
2.44394 7.0     1.43293 -0.492512 7.0     1.0 4.0 7.0
1.0 4.0 8.75     2.49608 5.80215 8.75     5.00914 3.54499 8.75     6.94646 5.37934
8.75     9.24968 4.21064 8.75     9.25016 1.78913 8.75     6.94378 0.622549 8.75
5.02464 2.44394 8.75     1.43293 -0.492512 8.75     1.0 4.0 8.75
1.0 4.0 10.0     2.49608 5.80215 10.0     5.00914 3.54499 10.0     6.94646 5.37934
```

```
10.0     9.24968 4.21064 10.0     9.25016 1.78913 10.0     6.94378 0.622549 10.0
5.02464 2.44394 10.0     1.43293 -0.492512 10.0     1.0 4.0 10.0
 ]
weight[ 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
]}
}] }
******************************************************************************
```

Teste 3: Artéria



```
******************************** Dados de Entrada ****************************
S 1
l 10
c 10
{
138 150 1500 154 155 1500 155.25 158.0 1500.0 156.5 161.0 1500.0 159 167 1500
158 181 1500 150 185 1500 141 188 1500 135 180 1500 129 176 1500
139 149 1497.0 156 155 1497.0 157.75 158.75 1497.0 159.5 162.5 1497.0 163 170
1497.0 153 186 1497.0 145 188 1497.0 135 184 1497.0 130 178 1497.0 125 173 1497.0
142 148 1494.0 160 156 1494.0 160.25 161.0 1494.0 160.5 166.0 1494.0 161 176
1494.0 155 183 1494.0 145 186 1494.0 135 184 1494.0 130 178 1494.0 126 169 1494.0
147 147 1491.0 162 161 1491.0 161.75 165.5 1491.0 161.5 170.0 1491.0 161 179
1491.0 153 184 1491.0 143 185 1491.0 134 180 1491.0 127 174 1491.0 126 164 1491.0
141 147 1488.0 158 154 1488.0 158.75 158.75 1488.0 159.5 163.5 1488.0 161 173
1488.0 159 181 1488.0 149 184 1488.0 139 183 1488.0 129 181 1488.0 126 171 1488.0
142 145 1485.0 159 155 1485.0 159.5 159.75 1485.0 160.0 164.5 1485.0 161 174
1485.0 156 180 1485.0 147 184 1485.0 137 182 1485.0 128 177 1485.0 125 169 1485.0
141 147 1482.0 159 155 1482.0 159.75 157.5 1482.0 160.5 160.0 1482.0 162 165
```

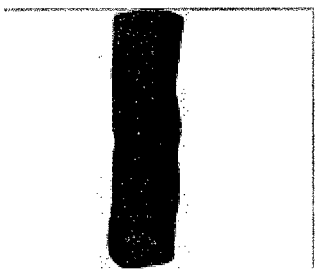
```
1482.0 161 175 1482.0 154 182 1482.0 144 184 1482.0 134 181 1482.0 127 174 1482.0
137 148 1479.0 155 153 1479.0 156.5 154.75 1479.0 158.0 156.5 1479.0 161 160
1479.0 162 170 1479.0 158 180 1479.0 148 184 1479.0 138 183 1479.0 129 177 1479.0
137 148 1476.0 155 152 1476.0 156.5 153.75 1476.0 158.0 155.5 1476.0 161 159
1476.0 162 169 1476.0 157 178 1476.0 150 183 1476.0 140 181 1476.0 130 180 1476.0
142 143 1473.0 158 152 1473.0 158.25 157.0 1473.0 158.5 162.0 1473.0 159 172
1473.0 153 181 1473.0 143 182 1473.0 133 179 1473.0 125 172 1473.0 124 163 1473.0
}
 ************************** Arquivo VRML de Saída ***************************
#VRML V2.0 utf8
               NavigationInfo {
               type ["EXAMINE","ANY"] }
     Transform {
        children [# Superficie: 1.0
Shape{
appearance Appearance {
            material Material {
               diffuseColor 1 0 0
               #specularColor 1 1 1
               #emissiveColor 0.7 0.6 0.5
               }
             }

geometry NurbsSurface {
          ccw FALSE
          solid FALSE
          uOrder  3
vOrder 3
uDimension 9
vDimension 9
uKnot [ 0.0 0.0 0.0 0.246915 0.292445 0.360742 0.470957 0.594631 0.71368 1.0 1.0
1.0  ]
vKnot [ 0.0 0.0 0.0 0.145327 0.238679 0.35955 0.465799 0.558401 0.666784 1.0 1.0
1.0  ]
controlPoint[ 138.0 150.0 1500.0     151.547 148.066 1500.0     155.238 158.008
1500.0     156.794 161.799 1500.0     159.636 166.79 1500.0     159.246 182.601
1500.0    149.969 184.763 1500.0    134.343 191.799 1500.0    129.0 176.0 1500.0
137.822 149.354 1498.1    148.227 147.81 1498.1    156.81 157.948 1498.1    159.971
161.908 1498.1     168.092 165.912 1498.1     149.074 192.106 1498.1     144.859
190.304 1498.1     125.411 177.486 1498.1     123.762 174.451 1498.1
141.711 147.976 1493.84     161.079 142.752 1493.84     160.435 160.651 1493.84
```

```
160.441 167.228 1493.84    160.803 178.983 1493.84    157.295 181.638 1493.84
145.37 186.296 1493.84    127.669 184.921 1493.84    126.333 169.131 1493.84
149.344 146.606 1490.47    164.567 154.09 1490.47    162.542 167.857 1490.47
161.916 173.086 1490.47    162.043 182.284 1490.47    151.245 185.159 1490.47
141.127 186.033 1490.47    126.209 172.895 1490.47    125.842 161.484 1490.47
139.879 148.008 1489.07    155.452 141.784 1489.07    158.123 157.544 1489.07
159.35 163.367 1489.07    161.296 172.989 1489.07    163.176 181.7 1489.07
150.368 184.124 1489.07    133.505 184.671 1489.07    126.528 172.826 1489.07
142.536 144.089 1484.28    158.994 144.034 1484.28    159.722 160.499 1484.28
160.164 166.613 1484.28    161.788 177.57 1484.28    154.665 179.631 1484.28
145.608 185.377 1484.28    127.742 179.043 1484.28    124.464 167.622 1484.28
141.623 147.241 1482.23    158.861 150.235 1482.23    160.43 157.607 1482.23
161.259 160.478 1482.23    162.651 164.317 1482.23    163.067 175.656 1482.23
154.694 182.647 1482.23    137.217 186.786 1482.23    126.966 174.34 1482.23
130.983 149.213 1475.3    143.198 147.981 1475.3    151.489 151.077 1475.3
154.752 152.274 1475.3    160.406 153.492 1475.3    162.618 161.379 1475.3
164.907 178.813 1475.3    144.982 188.22 1475.3    131.797 181.061 1475.3
142.0 143.0 1473.0    157.911 140.428 1473.0    158.233 157.028 1473.0    158.527
163.211 1473.0    159.733 173.512 1473.0    153.805 181.95 1473.0    142.912
182.417 1473.0    125.279 177.44 1473.0    124.0 163.0 1473.0
 ]
weight[ 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
]}
}] }
************************** Matriz de Distâncias *******************************
Erro da Matriz:  1.0
0.0d  4.5754380740269d  0.79143537718577d  0.56873124876963d  3.3324687378505d
1.2875801065647d  0.011711045241906d  1.7177524450362d  7.211102550928d  0.0d
2.469387914039d    5.9960012125965d    2.3310182041076d    3.4805800041286d
7.4059759086509d    4.3034039216768d    2.194293002068d    2.8262705687154d
8.670723349934d  2.9598950545098d
0.4719257849838d    6.5063306875808d    1.1411110927679d    1.4273345129318d
3.2548046780351d    1.7966424275886d    0.4278793103253d    2.0269213708562d
```

```
10.175232828901d  0.56830920178405d
0.32495734105312d     6.343292566601d     1.192139795782d     0.96358054026349d
2.6108281279739d     1.228557535784d     0.37262726575196d     2.0136805220096d
10.049403103499d  0.30953094016057d
1.5317634499012d     4.8391231174899d     1.0441988129066d     2.2323184233354d
3.6309074462515d     1.0916701057604d     1.2122526316933d     1.6412754675957d
11.39402371625d  1.4230387584458d
0.43848729485949d     6.0543312547692d     1.2403237635224d     0.82483928432852d
1.7663545391948d     1.5968651172601d     0.56408910264872d     2.3735872938324d
8.3611210783243d  0.4711680197383d
0.034221431531616d     4.0845490010734d     0.64379543793881d     0.4652356692296d
2.3377001322901d     1.2699466436087d     0.028955148855664d     1.9532099152665d
9.9077520216897d  0.018606942154236d
1.1219902118953d     4.9609379825753d     1.6515417662457d     0.84865216278145d
1.5238907346914d     2.4170529794974d     0.95679933613318d     2.7424383704585d
10.406777575014d  0.86337265955348d
7.6811457478687d     7.2255001761609d     4.0032090910994d     8.0414271313663d
16.506008836226d     14.227019583498d     14.871733689874d     15.828140596495d
24.269322199023d  18.275666882497d
2.3089905893801d-13     6.5360636722242d     1.2378496096442d     0.94168688342578d
3.0884394427554d     1.2553944046583d     0.019035770202596d     1.9613317204651d
9.0553851381374d  2.2958338407482d-13
*******************************************************************************
```

Teste 4: Artéria



```
******************************** Dados de Entrada ****************************
S 1
l 10
c 10
{
138 150 1500 154 155 1500 159 167 1500 163 175 1500 158 181 1500 150 185 1500
141 188 1500 135 180 1500 129 176 1500 126 168 1500
139 149 1485.0 156 155 1485.0 163 161 1485.0 163 170 1485.0 161 180 1485.0 153
186 1485.0 145 188 1485.0 135 184 1485.0 130 178 1485.0 125 173 1485.0
```

```
142 148 1470.0 152 149 1470.0 160 156 1470.0 163 166 1470.0 161 176 1470.0 155
183 1470.0 145 186 1470.0 135 184 1470.0 130 178 1470.0 126 169 1470.0
147 147 1455.0 156 152 1455.0 162 161 1455.0 163 171 1455.0 161 179 1455.0 153
184 1455.0 143 185 1455.0 134 180 1455.0 127 174 1455.0 126 164 1455.0
141 147 1440.0 158 154 1440.0 162 163 1440.0 161 173 1440.0 159 181 1440.0 149
184 1440.0 139 183 1440.0 129 181 1440.0 126 171 1440.0 126 161 1440.0
142 145 1425.0 151 149 1425.0 159 155 1425.0 162 164 1425.0 161 174 1425.0 156
180 1425.0 147 184 1425.0 137 182 1425.0 128 177 1425.0 125 169 1425.0
141 147 1410.0 151 150 1410.0 159 155 1410.0 162 165 1410.0 161 175 1410.0 154
182 1410.0 144 184 1410.0 134 181 1410.0 127 174 1410.0 125 164 1410.0
137 148 1395.0 147 148 1395.0 155 153 1395.0 161 160 1395.0 162 170 1395.0 158
180 1395.0 148 184 1395.0 138 183 1395.0 129 177 1395.0 125 168 1395.0
137 148 1380.0 147 148 1380.0 155 152 1380.0 161 159 1380.0 162 169 1380.0 157
178 1380.0 150 183 1380.0 140 181 1380.0 130 180 1380.0 124 172 1380.0
142 143 1365.0 150 145 1365.0 158 152 1365.0 161 162 1365.0 159 172 1365.0 153
181 1365.0 143 182 1365.0 133 179 1365.0 125 172 1365.0 124 163 1365.0
}
 ************************** Arquivo VRML de Saída **************************
#VRML V2.0 utf8
               NavigationInfo {
               type ["EXAMINE","ANY"] }
     Transform {
        children [Shape{
appearance Appearance {
            material Material {
               diffuseColor 1 0 0
               #specularColor 1 1 1
               #emissiveColor 0.7 0.6 0.5
               }
             }

geometry NurbsSurface {
          ccw FALSE
          solid FALSE
          uOrder  3
vOrder 3
uDimension 9
vDimension 9
uKnot [ 0.0 0.0 0.0 0.191257 0.301808 0.408623 0.515561 0.623873 0.735131 1.0
1.0 1.0  ]
vKnot [ 0.0 0.0 0.0 0.183243 0.305723 0.42801 0.555051 0.681129 0.802503 1.0 1.0
```

```
1.0  ]
controlPoint[ 138.0 150.0 1500.0     153.91 149.102 1500.0     158.626 167.621
1500.0     164.523 175.096 1500.0     158.145 181.392 1500.0     150.221 184.808
1500.0    140.306 189.87 1500.0    131.379 172.842 1500.0    126.0 168.0 1500.0
138.176 149.233 1488.77     157.471 157.703 1488.77     166.888 161.994 1488.77
162.409 171.076 1488.77     163.073 182.96 1488.77     151.653 188.296 1488.77
146.349 189.822 1488.77    126.51 180.65 1488.77    124.294 175.634 1488.77
141.341 148.04 1469.95     147.055 143.791 1469.95     159.387 153.678 1469.95
163.807 164.302 1469.95     161.055 175.199 1469.95     156.399 182.684 1469.95
144.779 186.308 1469.95    127.593 184.854 1469.95    126.233 168.931 1469.95
149.205 146.751 1455.11     153.753 148.315 1455.11     163.188 161.229 1455.11
163.672 171.688 1455.11     162.362 179.471 1455.11     153.824 184.591 1455.11
143.319 186.412 1455.11    129.094 174.216 1455.11    125.93 164.052 1455.11
139.25 147.476 1439.4     160.772 151.636 1439.4     163.106 164.433 1439.4
160.021 175.254 1439.4     159.761 183.529 1439.4     146.237 185.474 1439.4
136.545 181.844 1439.4    117.663 182.552 1439.4    126.195 158.754 1439.4
142.491 144.216 1425.75     145.566 145.517 1425.75     158.953 152.934 1425.75
162.832 161.406 1425.75     161.685 173.053 1425.75     158.867 178.566 1425.75
149.395 185.514 1425.75    132.595 180.835 1425.75    124.818 171.893 1425.75
141.591 147.189 1409.51     149.05 149.243 1409.51     160.71 154.451 1409.51
162.461 166.664 1409.51     161.664 176.938 1409.51     152.626 183.721 1409.51
142.026 184.425 1409.51    124.673 178.51 1409.51    125.011 161.806 1409.51
133.991 149.186 1392.78     141.52 144.287 1392.78     151.772 152.116 1392.78
161.319 155.584 1392.78     163.474 166.384 1392.78     162.817 179.351 1392.78
151.16 185.05 1392.78    135.07 186.018 1392.78    125.154 171.781 1392.78
142.0 143.0 1365.0    146.626 141.919 1365.0    158.651 151.422 1365.0    161.643
161.987 1365.0     159.422 172.046 1365.0     153.601 182.284 1365.0     142.756
182.336 1365.0     125.67 177.646 1365.0     124.0 163.0 1365.0
 ]
weight[ 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
]
      }
}] }
```

```
************************** Matriz de Distâncias ********************************
Erro da Matriz:  1.0
0.0d  1.0496902322951d  0.33663160775923d  0.050056092394255d  0.14269299006845d
0.31105623754454d 0.086363753567013d 0.55867262546819d 8.5440037453175d 0.0d
1.1742553356855d     1.8297002568759d     1.3815762958709d     1.2626690035871d
1.2604608561175d     1.1653331115941d     1.1673483358721d     1.317206873743d
7.2285461840612d  1.1711888084743d
0.40267395152239d     1.0086067839217d     0.53280002311471d     0.40092709632489d
0.44327153189583d     0.46553981363738d     0.39314344194684d     0.62193193885429d
9.969281078159d  0.40882460846972d
0.11180878506401d     0.82042348945122d     0.27464686115041d     0.11540388300902d
0.15177805932584d     0.36695772505991d     0.1415286345413d     0.50978239925818d
10.087403188864d  0.11729932209183d
0.25915275100476d     0.96896752633204d     0.38889929453498d     0.27043051685544d
0.28591621901354d     0.41115077267711d     0.25011783083776d     0.54301496768448d
9.9769229815432d  0.24687244506245d
0.46449709849411d     0.8450390454189d     0.49321502862362d     0.46534273478857d
0.46961078555338d     0.57346804031292d     0.47439229461607d     0.6669581766419d
8.5573266489157d  0.46393732735338d
0.080143112699338d     0.82955532879162d     0.2979017271028d     0.15165281460985d
0.22391256649349d     0.25857490940223d     0.083341811516724d     0.60455773481049d
10.109174221635d  0.11968667414257d
0.83860405625382d     1.2849862054823d     0.95802539494346d     0.89187486481186d
0.9383796607296d     0.82500577188226d     0.83789094805805d     1.1124265535293d
9.6594996816616d  0.85660443511725d
16.583123951777d     15.560292106031d     15.262797889131d     15.282534744054d
15.546153736446d     15.90847743399d     16.605452979675d     16.45711269067d
23.452078799117d  17.492855684536d
0.0d        0.80535963783735d        0.29134165912722d        0.074982963675189d
0.17616024765332d     0.30263582736087d     0.054302160163603d     0.51762257269146d
9.0553851381374d  0.0d
```