Pontifícia Universidade Católica
do Rio de Janeiro

João Marcos Silva da Costa

Propriedades de Curvas Silhuetas Discretas em Malhas Quadrangulares Planares

Dissertação de Mestrado

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

Orientador: Prof. Sinésio Pesco

Rio de Janeiro
Setembro de 2018

Pontifícia Universidade Católica
do Rio de Janeiro

João Marcos Silva da Costa

Propriedades de Curvas Silhuetas Discretas em Malhas Quadrangulares Planares

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

Prof. Sinésio Pesco
Orientador
Departamento de Matemática – PUC-Rio

Prof. Marcos Craizer
Departamento de Matemática – PUC-Rio

Prof. Luiz Henrique de Figueiredo
Instituto de Matemática Pura e Aplicada – IMPA

Prof. Ricardo Guerra Marroquim
Engenharia de Sistemas e Computação – UFRJ

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

Rio de Janeiro, 24 de Setembro de 2018

João Marcos Silva da Costa

Formou-se em licenciatura Matemática pela Universidade Federal Fluminense em 2016.

Ficha Catalográfica

Silva da Costa, João Marcos

Propriedades de Curvas Silhuetas Discretas em Malhas Quadrangulares Planares / João Marcos Silva da Costa; orientador: Sinésio Pesco. – 2018.

v., 70 f: il. ; 30 cm

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

Inclui bibliografia

1. Matemática – Teses. 2. Modelagem Matemática, Computação Gráfica – Teses. 3. Geometria Discreta;. 4. Silhuetas;. 5. Malhas quadrangulares planares..
I. Pesco, Sinésio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. III. Título.

CDD: 510

Agradecimentos

Antes de tudo agradeço aos meus pais, Albertino e Gislane, por todo o apoio durante estes dois anos.

Sou grato, também, ao professor Sinesio Pesco, pela orientação durante todo o período do mestrado.

Agradeço a Tamires, Pablo, Aline e Edson que me auxiliaram no período final de escrita dessa dissertação.

Em especial, agradeço aos meus amigos Danilo, Natasha, Patrícia, Tayná e Diego por não me deixarem desanimar nos momentos difíceis.

Meus sinceros agradecimentos à todos os funcionários e professores do Departamento de Matemática da PUC-RIO pela boa convivência.

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

Resumo

Silva da Costa, João Marcos; Pesco, Sinésio. **Propriedades de Curvas Silhuetas Discretas em Malhas Quadrangulares Planares**. Rio de Janeiro, 2018. 70p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

No presente trabalho apresentamos um estudo de curvas silhuetas discretas sobre alguns tipos particulares de malhas, com o objetivo de avaliar propriedades dessas curvas. Nosso objeto de estudo são malhas quadrangulares, ou seja, onde todas as faces sejam quadriláteros e também sejam planares. Em particular dois tipos de malhas são discutidas: circular e cônica. Essas malhas são particularmente interessantes em arquitetura para modelagem de estrutura de vidros. A geração das malhas é feita aplicando-se um processo de otimização e em seguida, sobre essas malhas, definimos curvas discretas como candidatas a silhuetas e buscamos medidas de qualidade para essas curvas.

Palavras-chave

Geometria Discreta; Silhuetas; Malhas quadrangulares planares.

Abstract

Silva da Costa, João Marcos; Pesco, Sinésio (Advisor). **Properties of Discrete Silhouette Curves on Planar Quad Meshes**. Rio de Janeiro, 2018. 70p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

In this work we study discrete silhouette curves on Planar Quad meshes (PQ meshes), with the objective of evaluate some properties of these curves. PQ meshes correspond to planar quadrilaterals meshes, and our interest is focused particularly on two kinds of meshes: Conical and Circular. They are interesting in architecture for design with glass structures. An optimization process is applied for the mesh generation and we follow defining discrete curves on the meshes to obtain silhouette and to measure their quality.

Keywords

Discrete Geometry; Silhouette; PQ Meshes.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

1.1
Motivação

Curvas de contorno ou silhuetas são muito conhecidas pelo seu aspecto artístico. Uma das formas como o olho humano percebe o ambiente é através das silhuetas, pois estas curvas delimitam a região visível do objeto e a que não é visível, causando uma sensação de profundidade. Algumas aplicações podem ser encontradas em problemas de renderização de sombras, detecção de colisões e na reconstrução de objetos.

Para reconstrução de objetos são utilizadas silhuetas projetadas sobre um plano, que são obtidas utilizando um ponto de observação sobre um círculo. À medida que movimentamos esse ponto sobre o circulo é gerada uma nova silhueta [1]. Assim, partindo de uma curva sobre uma superfície, estabeleceremos uma forma de reconhecer se a curva é uma silhueta, para estas curvas encontraremos o ponto de observação utilizando a geometria diferencial discreta [2].

Malhas planares são muito utilizadas pela arquitetura, principalmente, para modelar estruturas com formas livres. Durante o processo de modelagem pode ser necessário expressar características do material utilizado para construir o objeto físico. Um exemplo são estruturas de vidro, que para serem modeladas precisam que as malhas sejam planares. Estes tipos de malhas agregam propriedades importantes do modelo geométrico representado. Neste trabalho, utilizaremos dois tipos de malhas com estas características: as cônicas e circulares [3].

1.2
Descrição do problema

Neste trabalho estamos interessados em estudar as curvas silhuetas em malhas quadrangulares planares. Inicialmente, apresentaremos uma forma de obter estas malhas, dado que diferente de malhas de triângulos, as quadrangulares podem não ser planares. Um dos focos deste trabalho é descrever com detalhes um método de otimização que gere este tipo de malha, em particular, as cônicas e circulares. Estabeleceremos medidas locais e globais de qualidade para avaliar a qualidade e proximidade da malha gerada a um caso ótimo.

Em seguida, sobre malhas quadrangulares planares descreveremos o processo de geração de uma curva silhueta discreta e utilizaremos uma abordagem da geometria afim [4] para estas curvas. Com a abordagem da geometria discreta [2] e da geometria afim descreveremos uma forma de reconhecer se uma curva sobre a malha é uma silhueta. No caso positivo, encontraremos um ponto observador para a curva, caso contrário propomos uma medida que possa definir o quanto a curva esta distante.

1.3 Organização da dissertação

Este trabalho está dividido em quatro capítulos. O capítulo 2 apresenta alguns conceitos preliminares, com o objetivo de apresentar a estrutura de dados e um modelo de subdivisão da malha.

No capítulo 3, abordamos as malhas cônicas e circulares que são utilizadas na modelagem de formas livres. Apresentaremos algumas de suas propriedades e uma forma de obter estes dois tipos de malhas por meio de uma otimização. Para estas malhas descreveremos duas medidas que permitem avaliar a qualidade da malha e exibiremos os resultados obtidos na geração das mesmas.

No capítulo 4, descreveremos a silhueta discreta e seu processo de obtenção. Por meio de uma relação entre dois campos de vetores, definiremos uma medida de qualidade para a curva e exibiremos uma forma de encontrar um ponto de observação. Por fim avaliaremos as curvas sobre as malhas geradas. No capítulo 5, concluímos o trabalho e apresentamos algumas sugestões de trabalhos futuro.

2
Preliminares

2.1
Handle-Edge

Para representar os objetos e trabalhar com as estruturas geométricas utilizamos a estrutura de dados Handle-Edge [5], que é uma estrutura baseada na half-edge [6] com operadores definidos sobre o bordo da superfície. De forma simplificada, as operações no bordos são: unir, separar e criar arestas de bordo. Na Handle-Edge, podemos representar superfícies combinatórias com bordo, sem bordo, orientáveis e não orientáveis. Na Figura 2.1, temos as *half-edges* he_1 e he_2 que compartilham uma mesma aresta.

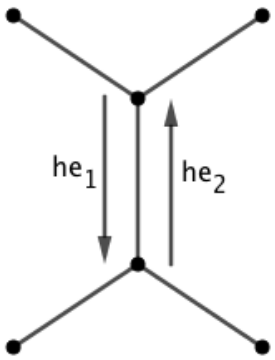


Figura 2.1: Half-Edge

A seguir (Figura 2.2), temos como a Handle-Edge está estruturada em níveis (*nós*); cada um desses *nós* representa uma lista duplamente encadeada.

- *Surface*: Lista de superfícies.
- *Face*: Lista de faces.
- *Half-Edge*: Um ciclo na face que estabelece uma orientação.
- *Edge*: Lista de arestas.
- *Vertex*: Lista de vértices.

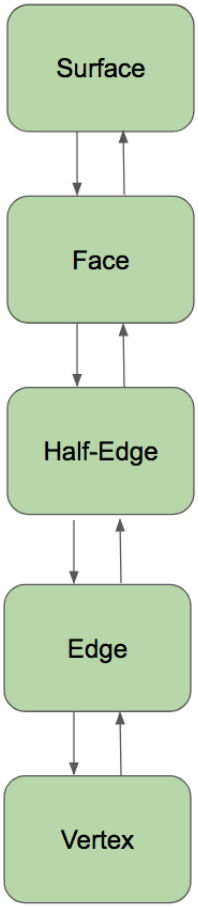
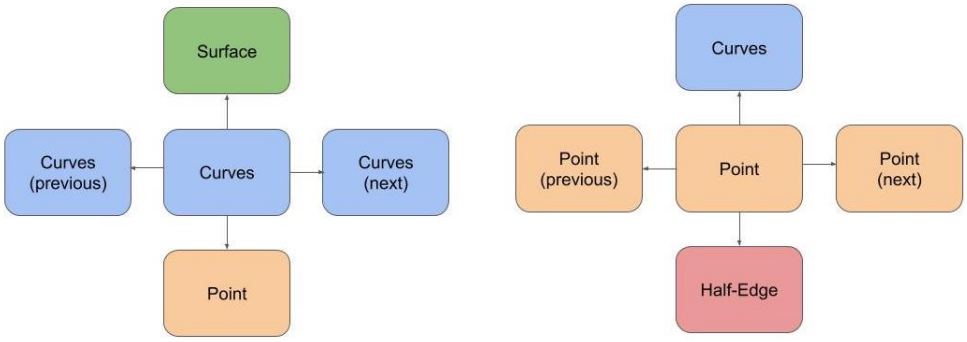


Figura 2.2: Nós da Handle-Edge

Neste trabalho acrescentamos duas novas listas, *Curves* e *Points*. Por meio destas novas listas é possível descrever uma orientação para a curva.

Na Figura 2.3, ilustramos como as novas listas estão relacionadas com a estrutura estabelecida.



2.3(a): Nó Curves 2.3(b): Nó point

Figura 2.3: Novos nós Handle-Edge

2.1.1 Catmull–Clark

A subdivisão Catmull–Clark [7] é um processo de subdivisão recursivo para malhas que gera novos vértices utilizando pontos face, pontos médios e pontos arestas . Este processo geralmente é realizado para suavizar e renderizar cenários em filmes de animação.

- Ponto face (*Pf*) é gerado pela média dos vértices da face.
- Ponto médio (Mp_i) é gerado pela média dos vértices de uma aresta.
- Ponto aresta (*Pa*) é gerado pela média entre os vértices da aresta escolhida e os *Pf* das faces que contém estas arestas (em vermelho na Figura 2.4)

$$Pa = \frac{Pf_1 + Pf_2 + v_1 + v_2}{4}$$

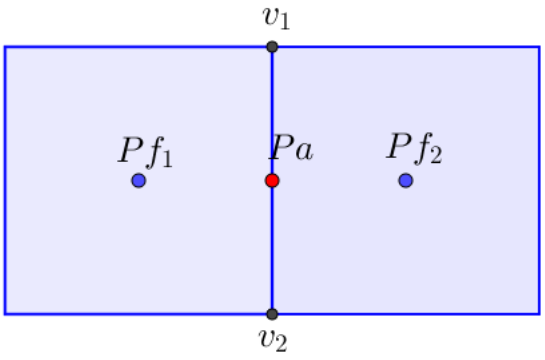


Figura 2.4: Ponto aresta e Ponto face

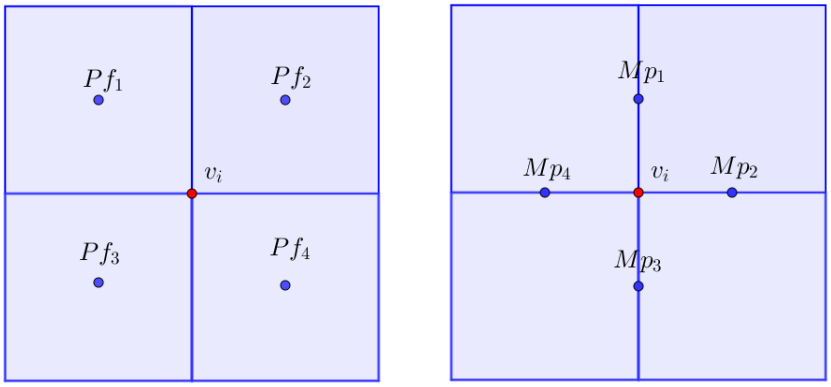
Para alterar os vértices da malha utilizaremos médias entre os *Pa* e *Pf*. Chamaremos de F a média dos *Pf* das faces que contém o vértice que será alterado (Figura 2.5 à esquerda).

$$F = \frac{Pf_1 + Pf_2 + Pf_3 + Pf_4}{4}.$$

Chamaremos R a média dos pontos médios Mp_i (Figura 2.5 à direita) das arestas que contém o vértice.

$$R = \frac{Mp_1 + Mp_2 + Mp_3 + Mp_4}{4}.$$

Com estes novos valores poderemos alterar um vértice da malha original para uma nova coordenada que chamaremos de v_{new}. O novo vértice é obtido a partir dos



2.5(a): Média dos pontos face F 2.5(b): Média dos pontos médios R

Figura 2.5: F e R

valores de R, F e do vértice que iremos alterar.

$$v_{new} = \frac{F + 2R + (n-3)v_i}{n},$$

onde n é a valência do vértice que será alterado. Com os vértices em novas posições, a subdivisão é feita em gerando uma nova face com os ponto v_{new}, Pa_1, Pa_2 e Pf_i.

3
Malhas Planares

Uma malha poligonal é uma representação discreta de uma superfície contínua. Essa representação é feita utilizando um conjunto de listas contendo faces, arestas e vértices. Existem diversos tipos de malhas, sendo mais utilizadas aquelas com a predominância de algum polígono como triângulos, quadriláteros ou outros polígonos convexos.

Neste capítulo definiremos dois tipos de malhas que serão utilizadas; as Cônicas e Circulares. Este capítulo está dividido em quatro seções. Na primeira seção definiremos os dois tipos de malhas que serão utilizados neste trabalho e algumas de suas propriedades. Na segunda seção descreveremos o processo de otimização utilizado para gerar estes tipos de malhas planares. Na terceira seção descreveremos os detalhes de implementação na otimização de malhas utilizando o software MATLAB. Por fim, analisaremos os desvios das malhas cônicas e circulares utilizando uma função que avalia a qualidade da malha gerada.

3.1
Malhas cônicas e circulares

Na arquitetura e design industrial é muito comum a utilização de matérias pouco flexíveis, como o vidro e madeiras compensadas. Na modelagem de formas livres a utilização destes materiais impõem restrições como a planaridade e faces quadrangulares. Para estas aplicações buscamos malhas que representem essas propriedades.

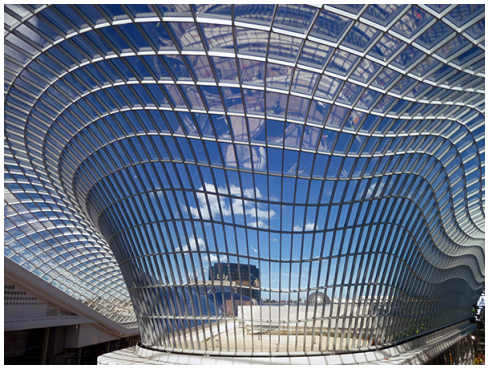
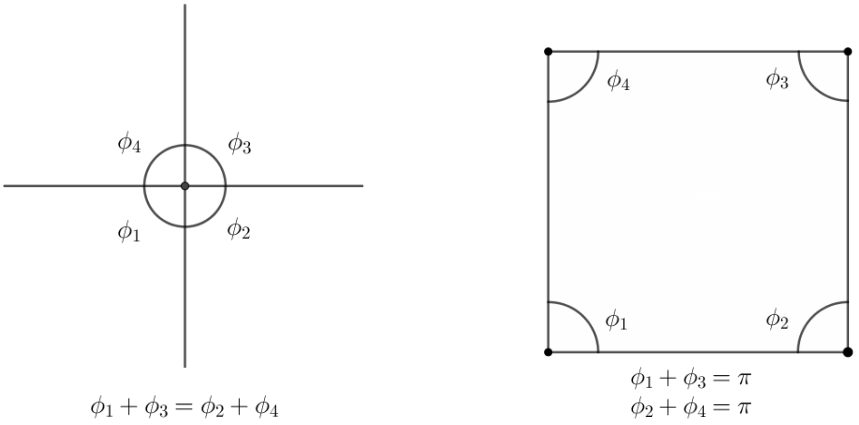


Figura 3.1: Estrutura de vidro [8]

Na Figura 3.1 podemos ver um exemplo de uma estrutura em vidro formada por faces planares. Dois exemplos de malhas quadrangulares planares presentes na modelagem destas estruturas são as malhas Cônicas e Circulares [3]. Estas malhas estão definidas pelas seguintes propriedades geométricas (Figura 3.2)

- Malhas Cônicas: a soma dos ângulos opostos na estrela de cada vértice deve ser igual.
- Malhas Circulares: a soma dos ângulos opostos em todas as faces é π



3.2(a): Malha Cônica 3.2(b): Malha Circular

Figura 3.2: Propriedades

Sobre estas malhas podemos destacar algumas características geométricas [9] como:

- Malhas Circulares: Todas as faces são inscritas em uma circunferência.
- Malhas Cônicas: em cada vértice interior da malha há um cone tangente às faces que se encontram nesse vértice.
- No processo de refinamento destas malhas, é possível subdividir preservando suas características utilizando o método de Catmull–Clark [7].
- Estas malhas possuem a propriedade de *offset*.

Definição 3.1 *O offset de uma malha M é uma nova malha M_d gerada pelo deslocamento de cada vértice da malha M a uma distância constante d na direção do vetor normal.*

$$M_d = M + dN_v,$$

onde N_v é o vetor normal no vértice.

A propriedade Offset não está presente em todo tipo de malha quadrangular, ocorrendo apenas em alguns casos particulares de malhas planares.

3.2 Geração das malhas

Seguindo o trabalho de Liu et al. [3], a geração de malhas Circulares ou Cônicas é feita aplicando um processo de otimização sobre uma malha de quadriláteros. Neste trabalho utilizamos a função objetivo estabelecida por Liu, porém a implementação é feita utilizando um vetor para representar a malha.

No processo de otimização, a função objetivo expressa cada uma das características que desejamos incluir na nova malha, como planaridade, proximidades da malha original, suavidade, além da característica que definem o tipo malhas. Assim a função objetivo utilizada é formada por quatro parcelas:

- Distorções na planaridade das faces (f_{det}).
- Distorções da propriedade Cônicas ou Circular (f_{angle}).
- Distância entre os vértices resultantes e os vértices da malha inicial (f_{close}).
- Fator de suavização (f_{fair}).

Utilizaremos a função para a otimização, que tem como parâmetro de entrada os vértices da malha M.

$$f_{obj}(M) := a_1 f_{close} + a_2 f_{fair} + f_{det} + f_{angle}.$$

Os valores a_1e a_2 são constantes que definirão como o processo deve priorizar a suavização e sua proximidade com a malha original. A seguir, descreveremos com mais detalhes cada uma das quatro componentes da função objetivo.

3.2.1 Planaridade

Para definirmos uma medida da planaridade para cada face da malha calculamos o volume definido por três vetores, com por exemplo os vetores e_1, e_2 e e_3 da Figura 3.3.

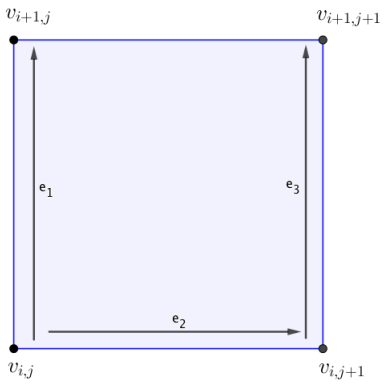


Figura 3.3: Vetores

O volume gerado por eles é determinado pelo valor absoluto obtido do determinante. Para que a face seja planar o volume deve ser nulo, ou seja, um dos vetores deve ser combinação linear dos outros dois. A função f_{det} utilizará em cada face todas as combinações desses três vetores. Esses volumes foram calculados nas faces utilizando os vetores:

$$e_{i,j} = (v_{i,j+1} - v_{i,j})/||v_{i,j+1} - v_{i,j}||$$
$$f_{i,j} = (v_{i+1,j} - v_{i,j})/||v_{i,j+1} - v_{i,j}||$$

Estes vetores definem duas direções na face $e_{i,j}$ na direção das colunas e $f_{i,j}$ na direção das linhas como ilustrado na Figura 3.4.

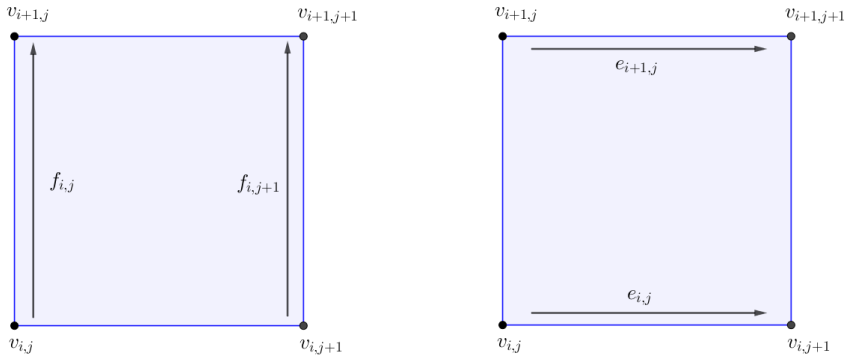


Figura 3.4: Vetores $e_{i,j}$ e $f_{i,j}$

Em cada uma das faces da malha são calculados quatro volumes que mostram o quanto esta face distancia-se de ser planar. Abaixo utilizamos os vetores $e_{i,j}$ e $f_{i,j}$ para descrever de forma mais clara cada um dos volumes.

$$c^1_{det,i,j} := det[e_{i,j} \quad e_{i+1,j} \quad f_{i,j}] \quad c^2_{det,i,j} := det[e_{i,j} \quad e_{i+1,j} \quad f_{i,j+1}]$$
$$c^3_{det,i,j} := det[e_{i,j} \quad f_{i,j} \quad f_{i,j+1}] \quad c^4_{det,i,j} := det[e_{i+1,j} \quad f_{i,j} \quad f_{i,j+1}]$$

Esses valores estão em cada uma das faces. Para que isso possa ser expresso de maneira global na malha utilizaremos a função

$$f_{det} = \sum_{i,j} (c^1_{det,i,j})^2 + (c^2_{det,i,j})^2 + (c^3_{det,i,j})^2 + (c^4_{det,i,j})^2.$$

Nesta função utilizamos o quadrado dos determinantes, pois algum dos vetores podem não estar na ordem de acordo com o sentido anti-horário permitindo assim que o valor do determinante seja negativo. Esta escolha torna a função sempre positiva e seu valor de mínimo busca reduzir as distorções na planaridade.

3.2.2
Propriedade cônica ou circular

Para que a malha resultante possa ter as propriedades que desejamos, utilizamos uma função que define o quanto a propriedade afasta-se do valor desejado. Para especificar o tipo de malha gerada, utilizamos a propriedade definida na seção 2.1. Como as malhas são definidas de maneiras diferentes utilizaremos duas funções que relacionam os ângulos na face ou no vértice.

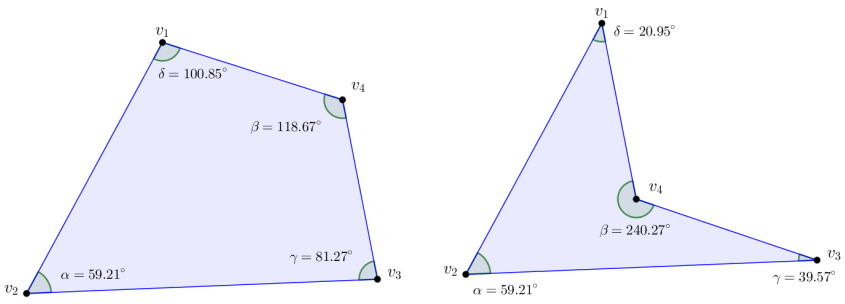
1. Malha Cônica: A propriedade é expressa sobre cada vértice da malha.

$$f_{angle-cônica} := \sum((\phi_1 + \phi_3) - (\phi_2 + \phi_4))^2$$

2. Malha Circular: A relação de ângulos está definida em cada face.

$$f_{angle-circular} := \sum(\phi_1 + \phi_3 - \pi)^2 + (\phi_2 + \phi_4 - \pi)^2.$$

Esta relação entre os ângulos, na malha circular, é fundamental para preservar a convexidade dos quadriláteros, pois desta forma estamos restringindo os vértices durante o processo. Podemos perceber melhor como a malha preserva a convexidade quando utilizamos a função referente à malha Circular. Nesta função a soma dos ângulos opostos devem estar próximos de π. Se um dos termos se distanciar da propriedade a mudança será ampliada pelos dois termos, tornando as faces convexas.



3.5(a): $f_{angle} = 27.38$ 3.5(b): $f_{angle} = 54639.58$

Figura 3.5: f_{angle} na face

Na Figura 3.5 temos os ângulos e os valores obtidos na função, os resultados obtidos estão em radianos.

3.2.3
Proximidade

Uma das características importantes é garantir que, após o processo de otimização, a malha resultante preserve a geometria da malha original. Assim, definiremos uma nova função que estabelece a proximidade entre as malhas resultante e inicial.

Uma maneira de preservar a forma é não permitir que os vértices possam sofrer grandes mudanças em suas coordenadas. Neste caso utilizamos o quadrado da distância entre os novos vértices e os vértices da malha inicial.

$$f_{close} := \sum_{i,j} ||(v_{i,j} - y_{i,j})||^2$$

Esta função mede a distância entre o vértice original e sua nova posição.

3.2.4
Suavidade

Embora a função f_{close} não permita que os vértices sofram grandes mudanças em suas coordenadas, ainda pode ocorrer da malha perder a suavidade. Na Figura 3.6 podemos ver um exemplo onde não incluímos a condição de suavidade. Para

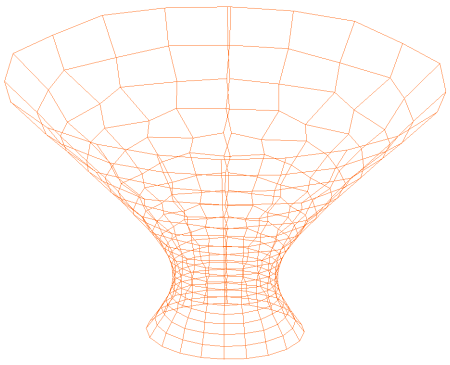


Figura 3.6: Otimização sem função f_{fair}

controlar a suavidade utilizaremos a seguinte função:

$$f_{fair} := \sum_{i,j} (v_{i+1,j} - 2v_{i,j} + v_{i-1,j})^2 + (v_{i,j+1} - 2v_{i,j} + v_{i,j-1})^2.$$

Esta função utiliza as relações de vizinhança da malha para suavizar o resultado. Esta função representa um Laplaciano discreto sobre os vértices da malha, visto que cada um dos termos representa uma segunda derivada discreta.

3.3 Implementação

Agora que estabelecemos as funções que compõem a função objetivo, descreveremos alguns detalhes de como foi feita a implementação da função objetivo. Para este trabalho utilizamos o solver de otimização do software MATLAB. Esta seção é dividida em quatro partes onde descrevemos como cada uma das funções foi implementada e como são obtidos os dados da malha.

3.3.1 Entrada dos vértices da malha

Para inserir as informações da malha no MATLAB, optamos por gerar arquivos auxiliares externos para leitura no MATLAB. Iniciamos com um arquivo que contém todos os vértices da malha. Este é um arquivo com apenas três colunas que representam cada uma das coordenadas do vértice. Como a estrutura de dados utilizada contém uma lista de vértices basta percorrê-la e escrever o arquivo da malha.

Algoritmo 1: Vértices da malha

```
1 Vértice V;
2 Pegue o primeiro Vértice da lista;
3 enquanto v ! = vazio faça
4 |   Imprima as coordenadas do vértice
5 fim
```

Resultado: Arquivo malha

Ao entrar com o arquivo de dados pela função *dlmread*, o MATLAB transforma o arquivo da malha em uma matriz V com três colunas e N linhas que representa o número de vértices presentes na malha. Nosso método de otimização organizará os dados de entrada em um vetor utilizando a função *reshape* (esta função transforma a matriz no vetor), conforme ilustrado na Figura 3.7.

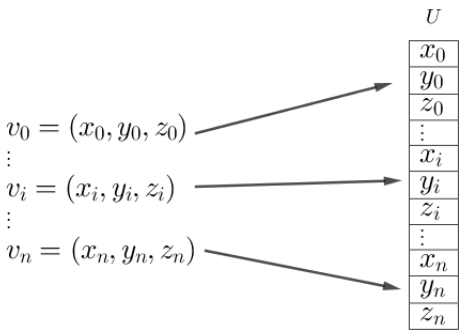


Figura 3.7: Vetor U

Assim um vértice v_n na malha corresponde

$$v_n = (U[3n], U[3n+1], U[3n+2]]),$$

no vetor U e nossa função objetivo

$$f_{obj} : \mathbb{R}^{3N} \to \mathbb{R},$$

é tal que N representa a quantidade de vértices da malha.

3.3.2 Função planaridade

A planaridade é avaliada pela função f_{det}, nela utilizamos os vetores construídos a partir dos vértices da face. Agora é necessário trabalhar de alguma forma estas relações nas faces da malha com o vetor U de maneira eficiente.

Iniciamos marcando cada vértice na malha com seu número correspondente na lista do Algoritmo 1, assim é possível identificar cada vértice da malha com seu correspondente no vetor U.

Como cada um dos vértices recebeu um número de ordem, podemos andar na lista de faces e escrever um arquivo com os índices dos vértices que formam a face. Neste arquivo cada linha representa uma face com quatro colunas que indicam os índices dos vértices da face orientados.

Algoritmo 2: Vértices da face

```
1  Face F;
2  he1=Half-Edge;
3  Pegue a primeira Face da lista;
4  enquanto F ! = vazio faça
5  |  he1=F → fedg; primeira half-Edge da face;
6  |  repita
7  |  |  imprima o índice do vértice;
8  |  |  he1=he1 → nxt;
9  |  até he1 != F → fedge;
10 |  F = F → nxt;
11 fim
```

Resultado: Arquivo de indices dos vértices de cada face

O MATLAB converte este arquivo de dados em uma matriz $F_{M\times 4}$, onde M é o número de faces da malha. Na Figura 3.8 podemos ver uma representação geométrica de como o algoritmo funciona em cada face e um exemplo de como o

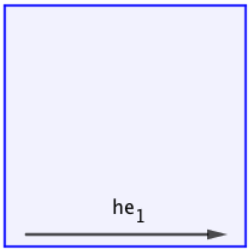
software transforma o arquivo .txt em matriz.

$$\begin{matrix} 10 & 15 & 20 & 17 \\ 23 & 28 & 16 & 32 \\ 31 & 58 & 36 & 82 \end{matrix}$$

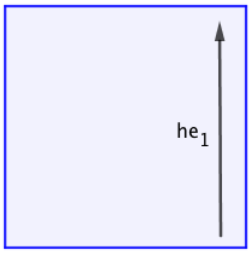
3.8(a): Face.txt

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ 10 & 15 & 20 & 17 \\ 23 & 28 & 16 & 32 \\ 31 & 58 & 36 & 82 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

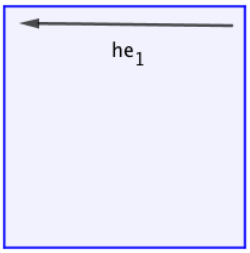
3.8(b): Matriz da face



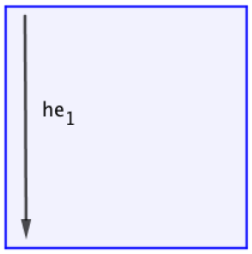
3.8(c): Passo 1



3.8(d): Passo 2



3.8(e): Passo 3



3.8(f): Passo 4

Figura 3.8: Algoritmo vértice da face

Como podemos ver o ciclo de semi-arestas permite pegar os vértices de forma ordenada, isso facilita a geração dos vetores para o cálculo dos determinantes. Abaixo temos como foi feita a implementação da função f_{det} utilizando as informações obtidas pelo arquivo de faces. Com o vetor U dos Vértices e a matriz F dos vértices da face definidos, a implementação da função f_{det} é representada no algoritmo 3. Para o código desta função no MATLAB veja o apêndice A1.

Algoritmo 3: Função f_{det}

```
m=número de linhas em F;
i=0;
fd=0;
enquanto i ≤ m faça
    j=1;
    enquanto j ≤ 4 faça
        v_j=[ U[ 3 F[ i ][ j ] ] : U[ 3 F[ i ][ j ] +1 ] : U[ 3 F[ i ][ j ] +2 ] ];
    fim
    construa os vetores da face;
    Calcule os determinantes c^1_{det,i,j}, c^2_{det,i,j}, c^3_{det,i,j} e c^4_{det,i,j};
    a=(c^1_{det,i,j})^2 + (c^2_{det,i,j})^2 + (c^3_{det,i,j})^2 + (c^4_{det,i,j})^2;
    f_det = f_det + a;
fim
```

Resultado: Valor de f_{det}

3.3.3
Proximidade das malhas

Para cada iteração da função objetivo teremos novos vértices e para que possamos manter a forma da malha temos que reconhecer os vértices em cada iteração. Como havíamos construído anteriormente o vetor da malha, temos que estes pontos estão associados sempre pelos mesmos índices. Para calcular a distância entre a malha original e a nova malha iremos utilizar a matriz com os vértices V. Para o código desta função no MATLAB veja o apêndice A3.

Algoritmo 4: Função f_{close}

```
m=número de linhas do arquivo da malha;
i=0;
fc=0;
enquanto i ≤ m faça
    v_new=[U[ 3 i ] : U[ 3 i +1 ] : U[ 3 i +2 ]];
    v_ini=[V[ i ][ 1 ] : V[ i ][ 2 ] : V[ i ][ 3 ] ]
    a=||v_new − v_ini||^2
    f_close = f_close + a;
fim
```

Resultado: Valor de f_{close}

3.3.4
Função de suavidade

Como vimos esta função está relacionada à suavização da malha. Para descrevermos esta função precisamos conhecer todos os vértices que compartilham uma aresta em comum com um "Vértice central", em vermelho na Figura 3.9 .

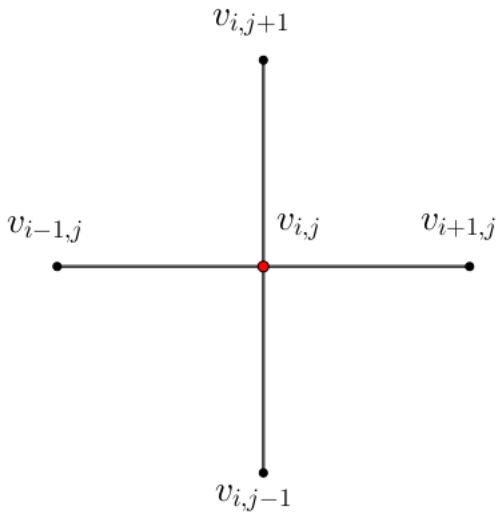


Figura 3.9: Vértice central

Para estabelecer essa relação de adjacência construímos um arquivo de dados similar ao arquivo de vértices da face. Nele cada linha representa a relação de adjacência do vértice através dos índices atribuído a cada um deles.

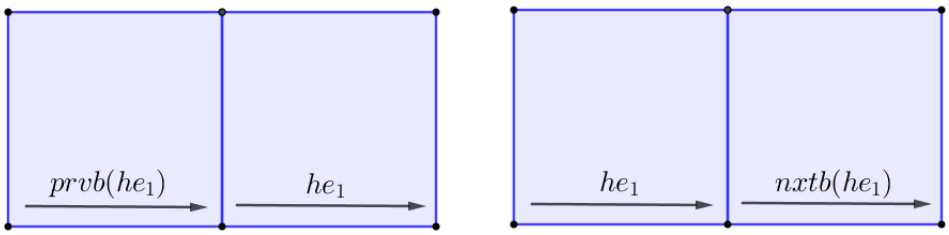
Para obter esses dados na estrutura utilizaremos as semi-arestas para percorrer a vizinhança do vértice. Esse processo deve considerar a existência de vértices de bordo, pois nestes vértices teremos arestas que não tem uma aresta oposta. Para que estes vértices sejam identificados é preciso conhecer sua valência, que indica o número de vértices adjacentes.

Como nossa malha é uma malha de quadriláteros temos que os vértices interiores tem valência 4, desta forma basta andar sobre o bordo e identificar apenas estes vértices que podem ter o valor da valência igual a 2 ou 3. A seguir descreveremos como foi feito para atribuir estes valores aos vértices.

Algoritmo 5: Valência do vértice

```
1  Vértice v;
2  Half-Edge he1;
3  enquanto v != vazio faça
4      se v → ebege == vazio então
5          v → valence = 4;
6      senão
7          he1=v → sbedge;
8          se h1 → prv != prvb(he1) então
9              v → valence = 3;
10         senão
11             v → valence = 2;
12         fim
13     fim
14     v = v → nxt;
15 fim
```

Na estrutura Handle-Edge, em cada vértice existe um ponteiro para uma possível semi-aresta de bordo (*ebedge*) sendo que, este ponteiro é vazio para vértices interiores. Abaixo temos os operadores que permitem andar apenas no bordo(Figura 3.10).



3.10(a): semi-aresta de bordo anterior 3.10(b): Próxima semi-aresta de bordo

Figura 3.10: Operadores de bordo

A informação de bordo permite que possamos indicar no processo de suavização quando estamos tratando de vértice de bordo. Nestes casos poderemos indicar com zero as ausências de vértices na geração do arquivo de adjacências.

```
1    2  3    4
10   15 25   0
20   9  0    0
```

Figura 3.11: Exemplo arquivo de dados

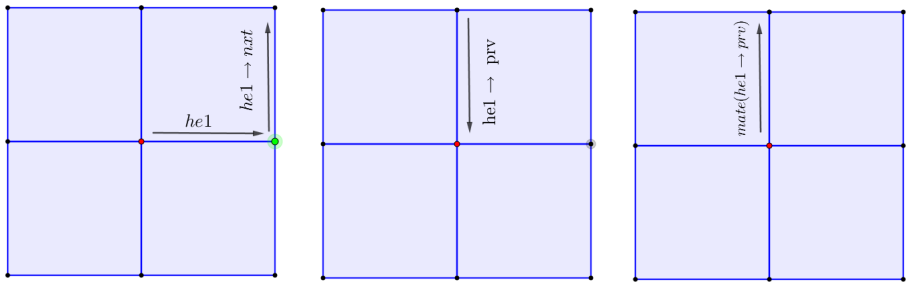
Na Figura 3.11 temos um exemplo do arquivo gerado, onde na primeira linha temos um vértice de valência quatro com todas as adjacências, na segunda linha um vértice de valência três e na terceira linha um vértice com valência dois. Nos dois últimos marcamos com zero as adjacências que não existem.

Definido a valência do vértice podemos percorrer seu entorno utilizando as semi-arestas e o operador *mate* que permite acessar a semi-aresta oposta em uma aresta.

Algoritmo 6: Adjacência do vértice

```
1  Vértice V;
2  he1=v→ sbedge;
3  Pegue o primeiro vértice da lista;
4  enquanto V != vazio faça
5  |  he1=v→ sbedge;
6  |  imprima a marca do vértice de he1;
7  |  enquanto he1 != v→ sbedge e he1 != vazio faça
8  |  |  imprima a marca do vértice de he1 → nxt;
9  |  |  he1 = mate(he1 → prv);
10 |  fim
11 |  se V → valence = 3 então
12 |  |  imprimi 0 na quarta posição.
13 |  senão
14 |  |  se V → valence = 2 então
15 |  |  |  imprime 0 na terceira e quarta posição.
16 |  |  fim
17 |  fim
18 |  V = V → nxt;
19 fim
   Resultado: Arquivo Adjacência do vértice
```

Abaixo uma representação de um passo do algoritmo que identifica um vértice adjacentes (em verde) ao vértice vermelho e segue ao próximo vértice adjacente (Figura 3.12).



3.12(a): Passo 1 3.12(b): Passo 2 3.12(c): Passo 3

Figura 3.12: Algoritmo de adjacência

Perceba que é necessário acessar a semi-aresta *nxt*, pois cada semi-aresta está relacionada com um vértice: he_1 esta relacionado ao vértice vermelho e o seu $he_1 \rightarrow nxt$ com o vértice em verde. Este arquivo de dados é lido no MATLAB e gerado uma matriz $S_{N\times 4}$, em que cada linha representa um "vértice central" e as colunas são os índices dos vértices adjacentes.

Agora com todas as informações armazenadas na matriz S, descreveremos a implementação da função f_{fair}. Para esta função ser aplicada em malhas com bordo utilizaremos o vetor nulo para indicar as relações que não existem nos vértices de bordo. Para o código desta função no MATLAB veja o apêndice A2.

Algoritmo 7: Função f_{fair}

```
1  m = Número de vértices da malha;
2  i=0;
3  f_fair=0;
4  enquanto i ≤ m faça
5      a_1=[ U[ 3i + 1 ] , U[ 3i + 2 ] , U[ 3i + 3 ] ];
6      a_2=[ U[3 S[ i ] [ 1 ] + 1] , U[3 S[ i ] [ 1 ] + 2] , U[3 S[ i ] [ 1 ] + 3] ];
7      a_3=[ U[3 S[ i ] [ 2 ] + 1] , U[3 S[ i ] [ 2 ] + 2] , U[3 S[ i ] [ 2 ] + 3] ];
8      se S[ i ] [ 3 ] = 0 então
9          a_4 = [ 0 , 0 , 0]
10     senão
11         a_4=[ U[3 S[ i ] [ 3 ] + 1 ] , U[3 S[ i ] [ 3 ] + 2 ] , U[3 S[ i ] [ 3 ]
               + 3 ] ];
12     fim
13     se S[ i ] [ 4 ] = 0 então
14         a_5 = [ 0 , 0 , 0]
15     senão
16         a_5=[ U[3 S[ i ] [ 4 ] + 1] , U[3 S[ i ] [ 4 ] + 2 ] , U[3 S[ i ] [ 4 ]
               + 3 ] ];
17     fim
18     c=||a_2 − 2a_1 + a_4||^2 + ||a_3 − 2a_1 + a_5||^2;
19     f_fair = f_fair + c;
20 fim
   Resultado: Valor f_fair
```

3.3.5
Propriedades cônica e circular

Para descrever esta última função não é necessário gerar novos arquivos, pois todas as relações de adjacência dos vértices foram descritas nos arquivos

anteriores. Na malha circular a propriedade está na face e utilizaremos o arquivo de vértices de uma face em sua implementação. Na malha Cônica utilizaremos o arquivo com a adjacencias do vértice para calcular os ângulos no entorno do vértice. Descreveremos agora a função para a malha circular. Para o código desta função no MATLAB veja o apêndice A4.

Algoritmo 8: Função f_{angle} malha Circular

```
m = Número de Faces da malha;
i=1;
f_angle=0;
enquanto i ≤ m faça
    a_0=[U[ 3 F[ i , 1 ] ], U[3 F[ i , 1 ] + 1 ],U[3 F[ i , 1 ] + 2]];
    a_1=[U[ 3 F[ i , 2 ] ], U[3 F[ i , 2 ] + 1 ],U[3 F[ i , 2 ] + 2]];
    a_2=[U[ 3 F[ i , 3 ] ], U[3 F[ i , 3 ] + 1 ],U[3 F[ i , 3 ] + 2]];
    a_3=[U[ 3 F[ i , 4 ] ], U[3 F[ i , 4 ] + 1 ],U[3 F[ i , 4 ] + 2]];
    j=0;
    enquanto j < 4 faça
        e_j=a_{j+1} − aj/(||a_{j+1} − aj||);
    fim
    e_3=(a_0 − a1)/(||a_0 − a1||);
    j=1;
    enquanto j < 4 faça
        g_j=arccos(⟨e_j − e_{j−1}⟩)
    fim
    g_0=arccos(⟨e_0 − e_3⟩);
    f_angle=f_angle +(g_0 + g_2 − π)^2 + (g_1 + g_3 − π)^2;
fim
Resultado: Valor f_angle
```

Abaixo descreveremos a implementação da função f_{angle} para a malha Cônica, nela teremos o cuidado de descrever o que é feito quando um ângulo não pode ser construído por não haver todas as adjacências. Para o código desta função no MATLAB veja o apêndice A5.

Algoritmo 9: Função f_{angle} malha Cônica

```
m = Número de vértices da malha;
i=0;
f_angle=0;
enquanto i ≤ m faça
    a_0=[ U[ 3i +1 ] , U[3i + 2 ] , U[ 3i + 3 ] ];
    a_1=[ U[3 S[ i ] [ 1 ] ] , U[3 S[ i ] [ 1 ] + 1] , U[3 S[ i ] [ 1 ] + 2] ];
    a_2=[ U[3 S[ i ] [ 2 ] ] , U[3 S[ i ] [ 2 ] + 1] , U[3 S[ i ] [ 2 ] + 2] ];
    e_0=(a_1 − a_0)/||(a_1 − a_0)||;
    e_1=(a_2 − a_0)/||(a_2 − a_0)||;
    g_0=arccos(⟨e_0, e_1⟩);
    se S[ i ] [ 3 ]!=0 então
        a_3=[ U[3 S[ i ] [ 3 ] ] , U[3 S[ i ] [ 3 ] + 1 ] , U[3 S[ i ] [ 3 ] + 2
         ] ];
        e_2=(a_3 − a_0)/||(a_3 − a_0)||;
        g_1=arccos(⟨e_1, e_2⟩);
    senão
        g1=0;
    fim
    se S[ i ] [ 4 ]!=0 então
        a_4=[ U[3 S[ i ] [ 4 ] ] , U[3 S[ i ] [ 4 ] + 1 ] , U[3 S[ i ] [ 4 ] + 2
         ] ];
        e_3=(a_4 − a_0)/||(a_4 − a_0)||;
        g_2=arccos(⟨e_3, e_2⟩);
        g_3=arccos(⟨e_0, e_3⟩);
    senão
        g2=0;
        g3=0;
    fim
    f_angle = f_angle + ((g0 + g2) − (g1 + g3))^2;
fim
```

Resultado: Valor f_{angle}

Os valores destas funções da malha Circular e Cônica podem ser obtidos de forma mais eficiente se forem executados junto com as funções f_{fair} e f_{det} respectivamente.

3.4 Método de otimização

Com uma função objetivo definida podemos agora utilizar o Solver do MATLAB para efetuar a otimização. Utilizamos a função *fuminunc* com as seguintes opções:

- Método utilizado Quase-Newton BFGS [10].
- Número máximo de iterações 10000.
- Número máximo de avaliações da função objetivo 100000.

Abaixo temos como utilizamos o otimizador do MATLAB para gerar as novas malhas.

```
tic
clear all
V = dlmread('vertexn.dat');
F = dlmread('facen.dat');
[l,c]=size(V);
options = optimoptions(@fminunc,'Algorithm','quasi-newton'
                       ,'MaxFunEvals',100000,'MaxIter',10000)
U = transpose(V); % Starting guess
[x,fval,exitflag,output] = fminunc(@objfun,U(:),options)
dlmwrite('circular.dat',transpose(reshape(x,3,l)),'delimiter');
toc
```

3.5 Resultados

Para uma avaliação dos resultados obtidos nas novas malhas utilizaremos uma medida local que verifica o desvio em cada vértice ou face e uma medida global na malha que indica o quanto a malha esta próxima de satisfazer as propriedades. Com o desvio definido poderemos verificar se as malhas planares com maior complexidade tem regiões com comportamento Cônico ou Circular e dizer se globalmente estão próximas desses dois tipos de malhas.

Para medida local (D_{local}), definimos:

- Malha Cônica: utilizaremos propriedade Cônica em cada vértice.

$$D_{cônico}(M) := (|\phi^1 + \phi^3| - |\phi^2 + \phi^4|)^2$$

- Malha Circular: utilizaremos propriedade Cônica em cada face.

$$D_{circular}(M) := (|\phi^1 + \phi^3 - \pi| + |\phi^2 + \phi^4 - \pi|)^2$$

Estas medidas locais são armazenadas nas faces ou nos vértices de acordo com o tipo de malha.

Para a medida global (d_{global}), definimos:

- Malha Cônica: Utilizaremos um somatório feito sobre os vértices da malha e os ângulos formados na estrela de cada vértice.

$$d_{cônico}(M) := \sqrt{\frac{\sum(|\phi^1 + \phi^3| - |\phi^2 + \phi^4|)^2}{V}},$$

onde V é o número de vértices da malha.

- Malha Circular: Nesta malhas a propriedade está expressa em cada face, desta forma teremos um somatório sobre o número de faces, avaliando os ângulos formados em seu interior.

$$d_{circular}(M) := \sqrt{\frac{\sum(|\phi^1 + \phi^3 - \pi| + |\phi^2 + \phi^4 - \pi|)^2}{F}},$$

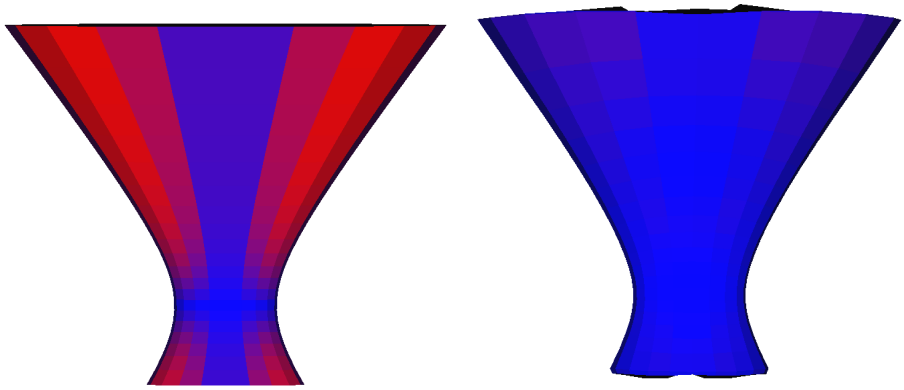
F é o número total de faces na malha.

Com estas funções podemos avaliar qualquer malha e definir se esta próxima ou não dos tipos de malhas que estamos trabalhando. Para a avaliação local, utilizamos uma variação de cores sobre a malha onde o azul representa o mínimo e vermelho o máximo desvio nas face ou vértice.

A seguir mostraremos os resultados obtidos no processo de otimização.

3.5.1 Exemplo 1 - hiperbolóide

No primeiro exemplo aplicaremos a otimização em um hiperbolóide com 400 faces com $a_1 = 1$ e $a_2 = 0.1$.



3.13(a): Antes 3.13(b): Depois

Figura 3.13: hiperbolóide - otimização Circular

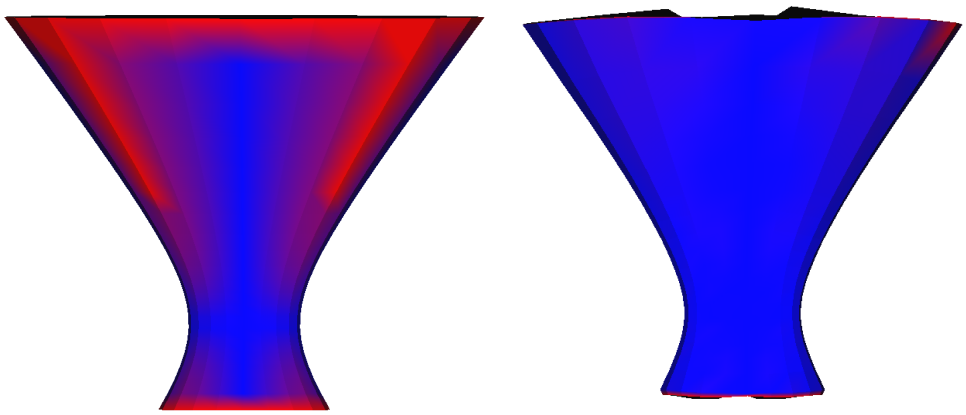
Na Figura 3.13 à esquerda temos o hiperbolóide antes de ser otimizado e à direita o resultado da otimização para uma malha Circular. A escala de cores representa o intervalo do desvio local $D_{circular}$, desta forma podemos vér a melhora da propriedade depois da otimização.

A tabela 3.1 informa o valor da parcela f_{close} e f_{det} (planaridade) após a otimização.

Iterações	f_{close}	f_{det}	Tempo
35	78.7147	3.1860×10^{-6}	30 min

Tabela 3.1: Dados de otimização - hiperbolóide Circular

Para gerar a malha cônica do hiperbolóide (Figura 3.14) utilizamos as constantes $a_1 = 0.8$ e $a_2 = 0.1$. Suas cores variam de acordo com a função $D_{cônico}$.



3.14(a): Antes 3.14(b): Depois

Figura 3.14: hiperbolóide - otimização Cônica

Na Tabela 3.2 temos os dados da otimização.

Iterações	f_{close}	f_{det}	Tempo
22	75.1990	3.1861×10^{-6}	28 min

Tabela 3.2: Dados de otimização - hiperbolóide Cônico

Aplicamos o mesmo processo para gerar uma malha Cônica e obtivemos o resultado visto na Figura 3.14. Podemos notar que as mudanças ocorreram na parte interna da malha (vértices que não estão no bordo), no bordo a propriedade Cônica não é satisfeita por não ter quatro ângulos. Na tabela 3.3 podemos ver os valores de desvios d_{global} obtidos nas malhas originais e otimizadas.

Otimização	Cônica	Circular
Antes	0.50854	0.57243
Depois	0.182264	0.229792

Tabela 3.3: Desvio total hiperbolóide

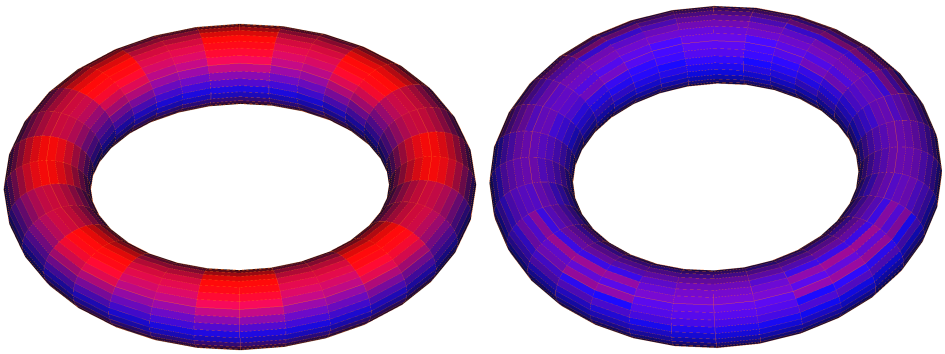
A Tabela 3.4 exibe o valor máximo de D_{local} antes e depois da otimização para os dois tipos de malhas.

Otimização	Cônico	Circular
Antes	1.78203	1.07801
Depois	1.09817	0.54615

Tabela 3.4: Desvio máximo hiperbolóide

3.5.2 Exemplo 2 - toro

Aplicamos o processo de otimização em uma malha sem bordo, para este caso utilizamos o toro (1024 faces) e os valores das contantes a_1 e a_2 são os mesmo do exemplo 1 (Figura 3.15). As cores utilizadas foram escolhidas através da função $D_{circular}$.



3.15(a): Antes 3.15(b): Depois

Figura 3.15: Toro - otimização Circular

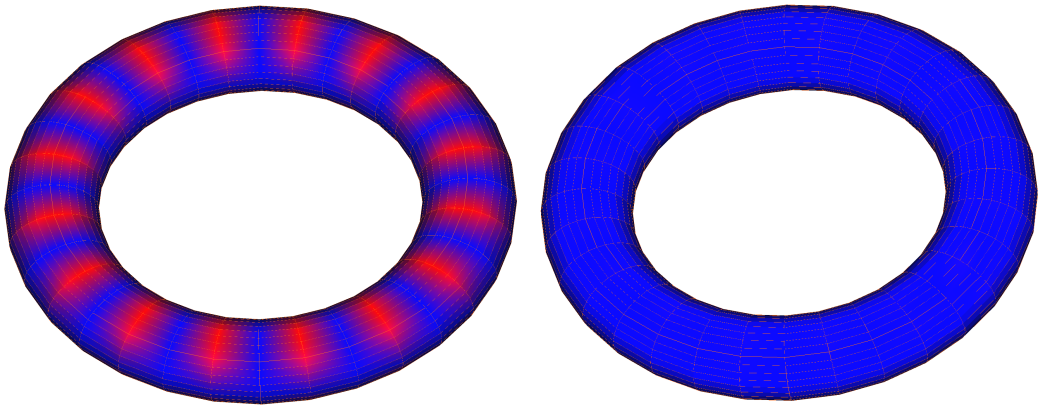
Na Tabela 3.5 abaixo temos os resultados da otimização do toro, sua planaridade e proximidade com a malha original.

Iterações	f_{close}	f_{det}	Tempo
28	0.022	0.008416	1 h 6 min

Tabela 3.5: Dados de otimização - Toro Circular

Na Figura 3.15 podemos perceber que as regiões em vermelho foram bem reduzidas, mostrando assim que a malha resultante teve uma grande melhora.

Para a avaliação Cônica (Figura 3.16) ocorre uma situação semelhante as regiões vermelhas são reduzidas mostrando que os desvios nesta nova malha são mínimos. As cores utilizadas foram escaladas utilizando o máximo desvio $D_{cônica}$.



3.16(a): Antes 3.16(b): Depois

Figura 3.16: Toro - otimização Cônica

Na Tabela 3.6 exibimos os dados da otimização, indicando seu tempo de processamento, planaridade e proximidade com a malha inicial.

Iterações	f_{close}	f_{det}	Tempo
30	0.0291	0.009865	51 min

Tabela 3.6: Dados de otimização - Toro Cônica

Na tabela 3.7 podemos comparar as medidas d_{global} antes e depois de aplicarmos a otimização.

Otimização	**Cônica**	**Circular**
Antes	0.00591328	0.00811455
Depois	0.00166444	0.00326713

Tabela 3.7: Desvios total Toro

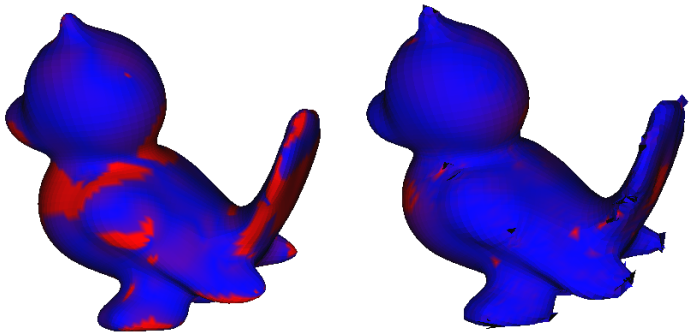
Além da medida d_{global} podemos analisar os valores de máximo da função D_{local} (Tabela 3.8) nos vértices e nas faces e assim comparar os resultados antes e depois.

Otimização	**Cônica**	**Circular**
Antes	0.0121264	0.0129496
Depois	0.00470322	0.00662084

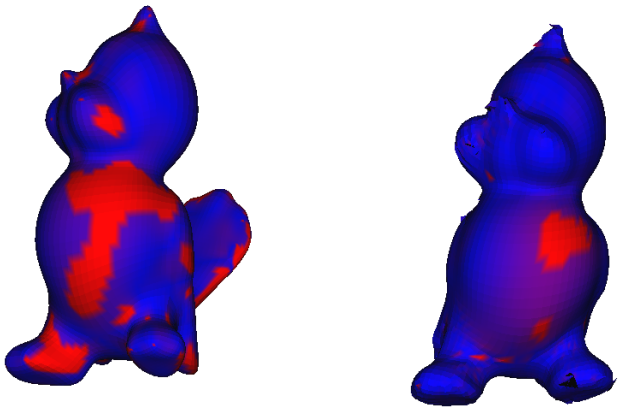
Tabela 3.8: Desvios máximo - Toro

3.5.3 Exemplo 3 - Tweety

Em malhas com um grande número de faces e vértices o tempo de processamento aumenta consideravelmente quando comparado aos exemplos anteriores. Na Figura 3.17 temos o tweety com 6000 faces, no antes e depois da otimização Cônica.



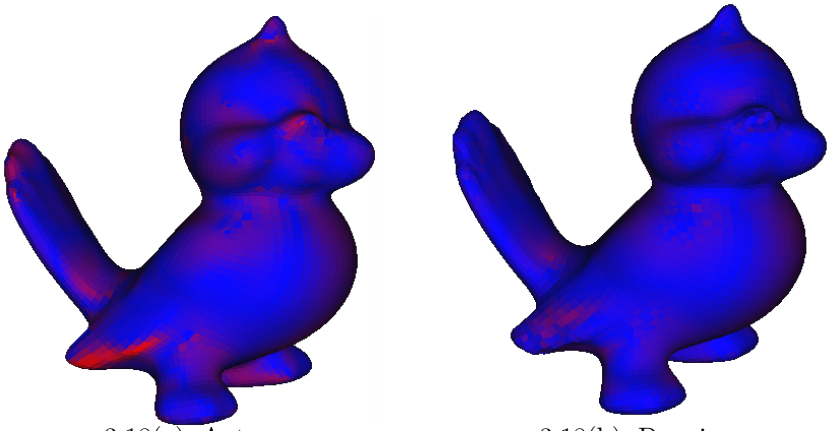
3.17(a): Antes 3.17(b): Depois



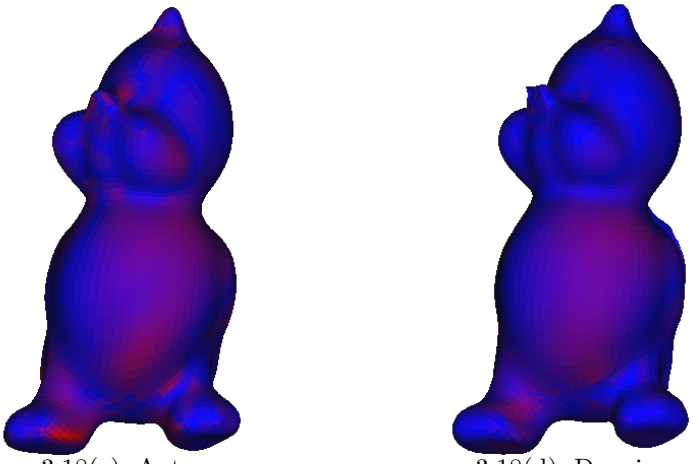
3.17(c): Antes 3.17(d): Depois

Figura 3.17: Otimização cônica Tweety

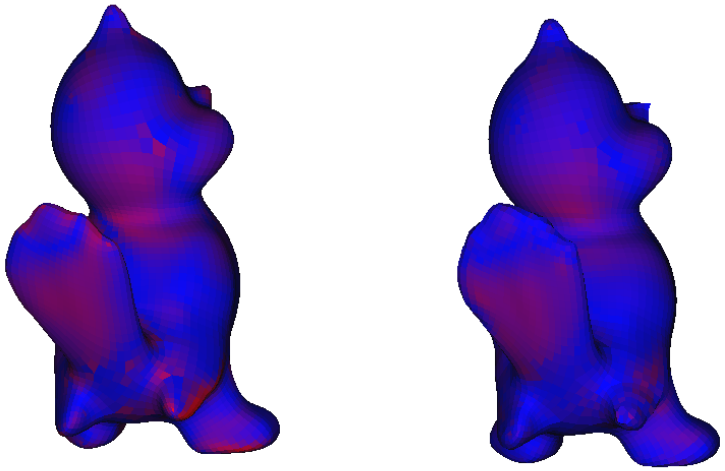
No caso cônica, utilizamos o $D_{cônico}$ para definir a variação das cores. Na Figura 3.18 temos o tweety no antes e depois da otimização circular.



3.18(a): Antes 3.18(b): Depois



3.18(c): Antes 3.18(d): Depois



3.18(e): Antes 3.18(f): Depois

Figura 3.18: Otimização circular Tweety

No caso cônica, utilizamos o $D_{cônico}$ para definir a variação das cores. Podemos notar que em nas regiões com maior desvio da propriedade foi onde obtivemos os melhores resultados em algumas partes os desvios sumiram (Figura

3.18). Estes resultados podem ser vistos através das medidas d_{global} e do máximo de D_{local} feita (Tabela 3.9)

Otimização	**Cônica**	**Circular**
Antes	0.78567	0.777733
Depois	0.336381	0.486021
Desvio máximo antes	3.2222	2.76394
Desvio máximo depois	2.22689	2.76394

Tabela 3.9: Resultados - tweety

Devido à complexidade da malha e seu número de faces ser grande a otimização do Tweety durou entorno de 13 horas. A demora para esta malha ocorre tanto pelo número de faces e vértices que não satisfazem as propriedades locais. Pela tabela 3.9 podemos notar que o desvio máximo é alto com relação aos outros dois exemplos. As constantes utilizadas para gerar a malha Cônica e Circular do Tweety foram as mesmas contantes do toro e hiperbolóide. De maneira geral, em todos os exemplos obtivemos boas aproximações das malhas para os tipos cônicas e circulares.

4
Silhuetas

Para estudar algumas propriedades das curvas silhuetas, vamos apresentar alguns conceitos da geometria afim que serão importantes para o estudo do caso discreto. Um dos principais tópico que abordaremos nas próximas seções é e definição do campo de Darboux paralelo [4] para uma curva contida em uma superfície. Pelo campo de Darboux estabeleceremos as principais propriedades que caracterizam a curva silhueta. Iniciamos com as definições no caso contínuo para em seguida definir o caso discreto. O principal objetivo deste capítulo é utilizar a geometria discreta para responder se uma curva sobre uma malha é silhueta, em caso negativo, medir sua distância de uma silhueta.

Em seguida apresentaremos resultados de algumas medidas de qualidade propostas para estas curvas em malhas geradas no capítulo 3. As definições utilizadas são baseadas no trabalho de [4] e [11].

4.1
Curva silhueta

A curva silhueta de uma superfície suave S em $\mathbb{R}^3$ é definida como o conjunto de pontos p de S que satisfazem a seguinte relação $\langle N_p, k\rangle = 0$, onde k representa o vetor de observação e N_p o vetor normal da superfície no ponto. O vetor de observação k é definido por $k = p - O$, onde O é o ponto observador e p o ponto da superfície (Figura 4.1).

A curva silhueta tem a propriedade de delimitar o que está dentro e fora do campo de visão.

4.1.1
Campo de Darboux paralelo

A geometria afim é o estudo das propriedades de curvas e superfícies que são invariantes por transformações afins. As transformações afins formam uma classe mais ampla de transformações geométricas que não utilizam uma origem fixa para defini-las. A partir delas é possível descrever as transformações geométricas utilizadas em processos de visualização e animação. Nosso objetivo é estudar curvas silhuetas no contexto da Geometria afim. Nesta seção utilizaremos alguns conceitos de geometria diferencial [11].

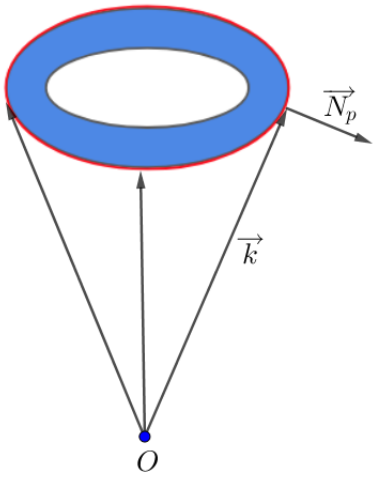
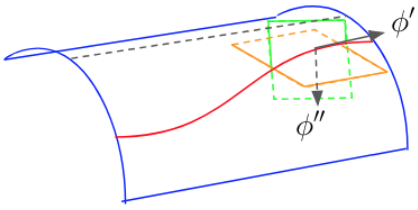


Figura 4.1: Curva silhueta

Definição 4.1 *Diremos que uma curva suave $\phi(t)$está parametrizada pelo comprimento de arco afim se $[\phi'(t), \phi''(t), \phi'''(t)] = 1$, onde $[\cdot, \cdot, \cdot]$ representa o determinante dos três vetores ([12] e [13]).*

Definição 4.2 *Diremos que uma curva suave $\phi(t)$ contida em uma superfície S é não-degenerada se seu plano osculante não coincide com o plano tangente a S em qualquer ponto.*



4.2(a): plano osculador em verde

Figura 4.2: Plano tangente(em laranja) e Plano osculante (em verde).

Na Figura 4.2 temos em laranja o plano tangente a superfície e em verde o plano osculador (gerado por ϕ' e ϕ'').

Definição 4.3 (Campo de Darboux paralelo) *Seja uma curva ϕ não-degenerada, contida em uma superfície S e $\xi(t)$ um campo de vetores ao longo de $\phi(t)$, tangentes a S e transversal a $\phi(t)$. Então existe um campo $\xi(t)$, único a menos de uma constante multiplicativa, tal que $\xi'(t)$ é tangente a $\phi(t)$ para todo $t \in I$. Este campo é denominado campo de Darboux paralelo de $\phi \subset S$ (Figura 4.3).*

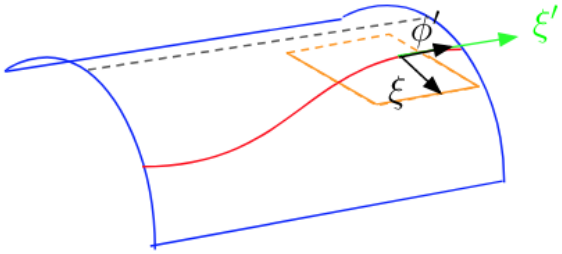


Figura 4.3: Campo de Darboux paralelo

Pela definição, a derivada do campo de Darboux paralelo esta na direção do vetor tangente $\phi'(t)$ e portanto $\xi'(t) = -\sigma(t)\phi'(t)$, onde $\sigma(t)$ é uma função escalar.

Definição 4.4 *Para uma curva ϕ suave contida em uma superfície S, nós diremos que uma parametrização $\phi(t)$ é adaptável a S se*

$$[\phi'(t), \phi''(t), \xi(t)] = 1,$$

onde $[\cdot, \cdot, \cdot]$ denota o determinante entre três vetores e $\xi(t)$ é o campo de Darboux paralelo da curva $\phi(t)$ contida em S.

Este determinante está relacionado com o volume gerado por estes vetores (Figura 4.4).

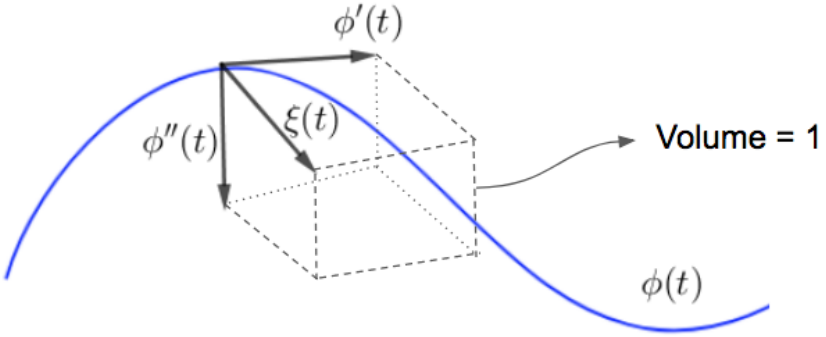


Figura 4.4: Parametrização adaptada

Definição 4.5 *Diremos que uma curva suave está parametrizada pelo comprimento de arco centro-afim se $[\phi(t) - O, \phi'(t), \phi''(t)] = 1$.*

Para que uma curva contida numa superfície seja uma silhueta é necessário satisfazer algumas condições estabelecidas em [14], descrita no Teorema abaixo.

Teorema 4.6 *A curva não-degenerada $\phi \subset S$ é uma curva silhueta com respeito a algum ponto O, se, e somente se σ é constante.*

Na definição do campo de Darboux paralelo se tomarmos $\sigma(t) = -1$ e $\xi(t) = \phi(t) - O$ teremos que a curva $\phi(t)$ é uma curva silhueta e seu ponto de observação encontra-se no ponto O.

Definição 4.7 *A superfície $X(t,u) = \phi(t)+u\xi(t)$, onde $\phi(t)$ é uma curva contida na superfície S e $\xi(t)$ é o campo de Darboux associado a esta curva, é chamada de envelope dos planos tangentes sobre a curva $\phi(t)$. Esta superfície também é chamada de tangente osculante.*

A superfície tangente osculante pode gerar um cone, como ilustrado na Figura 4.5.

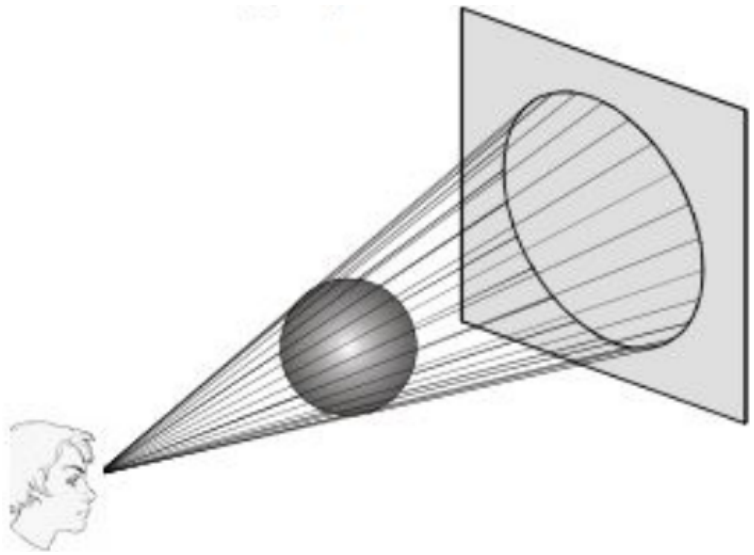


Figura 4.5: Cone de visualização [15]

Neste trabalho estamos interessados quando a superfície tangente osculante é um cone, pois neste caso a curva $\phi(t)$ é uma silhueta e podemos determinar o ponto O. Este cone é chamado cone de observação da curva.

4.2 Silhueta discreta

Nosso objetivo é estudar propriedades da curva silhueta discreta nas malhas quadrangulares do capítulo 3. Inicialmente vamos apresentar uma definição de silhueta discreta usando alguns conceitos da seção anterior e em seguida vamos buscar uma medida de qualidade destas silhuetas.

4.2.1
Geometria discreta

A geometria diferencial discreta [2] tem como foco de seus estudos objetos discretos análogos aos contínuos. Um dos objetivos é descrever os objetos discretos de forma que não sejam meramente aproximações do contínuo.

Nesta seção descreveremos algumas definições de geometria diferencial discreta [16], com o foco em definir Campo de Darboux paralelo discreto em malhas planares [4] e curvas silhuetas.

Definição 4.8 *Seja* $f : [1, ..., N] \to \mathbb{R}^3$. *Definimos a derivada utilizando diferenças finitas como* $f'(i) = f(i+1) - f(i)$.

Utilizado esta definição para a derivada, podemos definir os vetores tangentes a curva ϕ e o campo de Darboux paralelo na forma discreta. O vetor tangente é obtido pela derivada discreta de ϕ', ou seja: $\phi'(i) = \phi(i+1) - \phi(i)$.

Seja S uma malha de quadriláteros planares e $\phi(i)$ uma curva discreta que intersecta arestas opostas de uma face. O vetor $\xi(i)$ é definido na direção da aresta de S que contém $\phi(i)$, sobre estas condições definiremos o campo de Darboux paralelo discreto (Figura 4.6).

Definição 4.9 *Um campo de vetores* $\xi : [1, ..., N] \to \mathbb{R}^3$ *é chamado de campo de Darboux paralelo se* $\xi'(i)$ *é paralelo ao vetor tangente da curva* $\phi(i)$, *ou seja, satisfaz a seguinte equação* $\xi'(i) = -\sigma(i)\phi'(i)$, *para alguma função escalar* $\sigma(i)$.

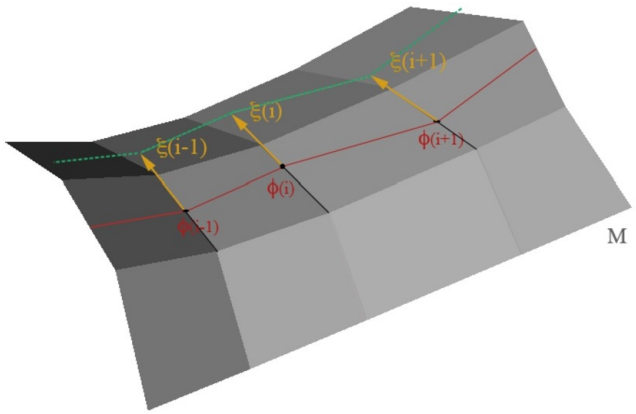


Figura 4.6: Campo de Darboux paralelo [4]

4.2.2
Avaliação da silhueta

Com os elementos discretos anteriores, podemos caracterizar a silhueta discreta para uma malha quadrangular:

Teorema 4.10 *A curva discreta $\phi(i) \subset M$ é uma curva silhueta com respeito a algum ponto O, se, e somente se $\sigma(i)$ é constante.*

Definição 4.11 *O poliedro $X(i,u) = \phi(i) + u\xi(i)$, $u \in \mathbb{R}$, onde $\phi(i)$ é uma curva contida na Malha M e $\xi(i)$ é o campo de Darboux associado a esta curva, é chamado de poliedro osculante desenvolvível.*

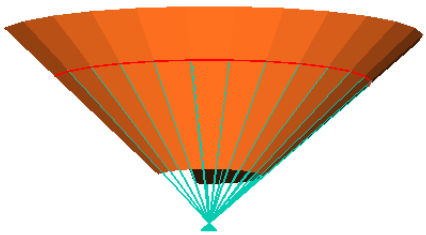


Figura 4.7: Poliedro tangente osculante discreta

Na Figura 4.7 temos o exemplo de uma curva no cone, podemos notar que o poliedro osculante desenvolvível é um cone, ou seja, a curva em vermelho é uma silhueta da malha e seu ponto observador encontra-se no vértice do cone.

Para avaliar uma curva silhueta C utilizaremos a relação estabelecida entre o campo de Darboux e o vetor tangente estabelecida pela função $\sigma(i)$. Para que uma curva seja chamada de silhueta pela abordagem descrita, a função $\sigma(i)$ deve ser constante. Para medir esta condição utilizaremos um somatório dos valores da derivada da função discreta σ:

$$H(C) = \sum_{1}^{N-1} |\sigma'(i)| = \sum_{1}^{N-1} |\sigma(i+1) - \sigma(i)|,$$

onde C representa a curva que será avaliada.

Se o valor dessa soma for nulo teremos uma silhueta e encontraremos um ponto de observação para esta curva. Nos casos em que a curva $\phi \subset M$ não é silhueta, buscaremos uma medida que indique sua proximidade com uma curva silhueta.

Se a curva não é uma silhueta não teremos um ponto observador. A Figura 4.8 ilustra esta situação. Neste caso podemos determinar os vários pontos $O(i)$ e

construir um polígono com estes pontos, para visualizar a curva formada. Este polígono será denominado polígono observador.

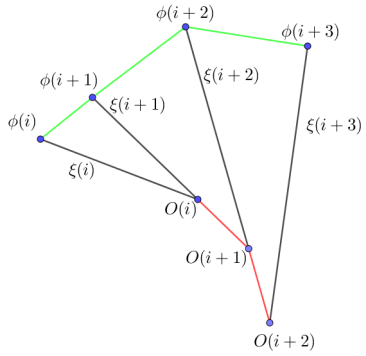


Figura 4.8: Polígono observador.

Neste tipo de curva utilizaremos uma avaliação de proximidade da silhueta utilizando a soma das distância entre os pontos $O(i)$ obtidos.

$$G(C) = \sum_{1}^{N-1} |O(i+1) - O(i)|,$$

Na próxima seção apresentaremos os resultados obtidos principalmente nas malhas cônicas e circulares. O objetivo das medidas é responder a questão proposta no início do capítulo para uma curva qualquer. Neste trabalho utilizaremos uma curva construída pelo pela variação de sinal da norma com o vetor observador e uma curva qualquer contida em uma faixa.

4.3 Geração das curvas silhuetas discretas

As silhuetas discretas são geradas utilizando um observador previamente estabelecido e o cálculo do produto interno entre as normais estimadas nos vértices e o vetor observador k (ver Figura 4.9).

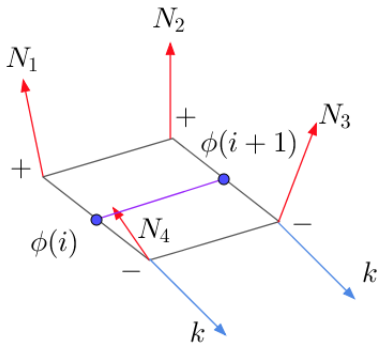


Figura 4.9: Silhueta.

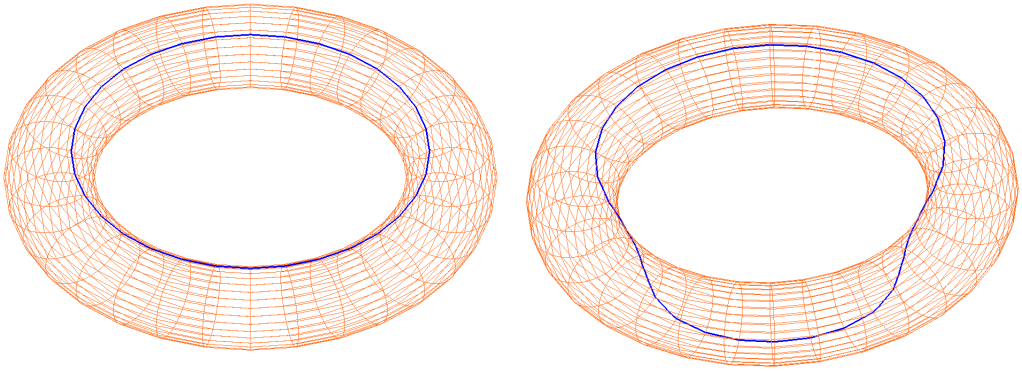
Em vermelho os vetores normais estimados em cada vértice, em azul os vetores de observação construídos em cada vértice da malha e em roxo temos a curva silhueta discreta. Note que a curva está sendo interpolada quando ocorre variação do sinal.

Um dos fatores importantes na geração da silhueta é a orientação, sem este fator não é possível estabelecer os campos de vetores. Para construir uma curva orientada precisamos primeiro verificar se a malha tem ou não bordo. Se tiver bordo a busca inicia nas arestas de bordo, caso contrário iniciamos a busca na lista de faces até encontrar a primeira face que contenha um parte da silhueta. Encontrada esta aresta seguimos a curva nas faces pela qual ela passa. Para esclarecer melhor como ocorre esse processo descreveremos de forma mais detalhada a sequência de passos utilizadas para a obtenção da curva silhueta.

- Passo 1: Andar inicialmente sobre as faces de bordo buscando a variação no produto interno. Caso encontre um mudança de sinal segue para o próximo passo, caso contrário busca pelas faces interiores até encontrar.
- Passo 2: Iniciamos a construção do nó curva, no qual os pontos estarão associados.
- Passo 3: Interpolar um ponto sobre a aresta onde a variação de sinal ocorre construir um nó Point, associá-lo a curva criada e marcar a face como visitada.
- Passo 4: Utilizar a Half-Edge da aresta para andar sobre as faces por onde a curva passa.
- Passo 5: Repetir os Passos 3 e 4 até que a curva chegue ao bordo ou retorne a aresta inicial.
- Passo 6: Voltamos a buscar por novas curvas nas faces que não foram visitadas até chegar ao fim da lista.

Na Figura 4.10 podemos ver duas curvas silhuetas geradas sobre o toro, à esquerda temos uma silhueta plana e à direita a curva que não está contida em nenhum plano. Estas curvas foram gerada seguindo a sequência de passos descrita, utilizando pontos de observação distintos.

Para atualizar os pontos devemos alterar as faces de forma a ponto consecutivos estarem numa mesma face.



4.10(a): Silhueta planar 4.10(b): Silhueta não planar

Figura 4.10: Silhuetas - toro

4.4 Resultados

4.4.1 Avaliação da curva silhueta

Para encontramos o ponto observador para curvas discretas contidas em uma faixa da malha utilizaremos o método de mínimos quadrados para encontrar as interseções entre retas geradas pelo vetores de Darboux paralelo. O Algoritmo 10 descreve este processo.

Algoritmo 10: Ponto de observação

Entrada: Curva C

```
1  Ponto p; ;
2  p= C→ point;
3  enquanto p diferente de vazio faça
4      q=p→nextp;
5      Armazene as direções de Darboux paralelo de p e q nas coluna da
         matriz A ;
6      b = p − q;
7      resolve o sistema A^t Ax = A^t b ;
8      armazene a solução;
9      imprime o ponto encontrado;
10     p=p→nextp;
11 fim
```

Resultado: Ponto de observação

Este algoritmo permite encontrar uma aproximação para o ponto de observação. Neste caso estamos restringindo as curvas silhuetas a um conjunto de curvas

contidas em uma faixa. Para outras curvas, pode ocorrer que o sistema não tenha solução ou para cada ponto ter interseções em pontos diferentes.

Na Figura 4.11, temos um exemplo de uma malha em que as interseções do campo de Darboux paralelo ocorrem em mais de um ponto. Nestes casos, teremos um polígono observador para a curva.

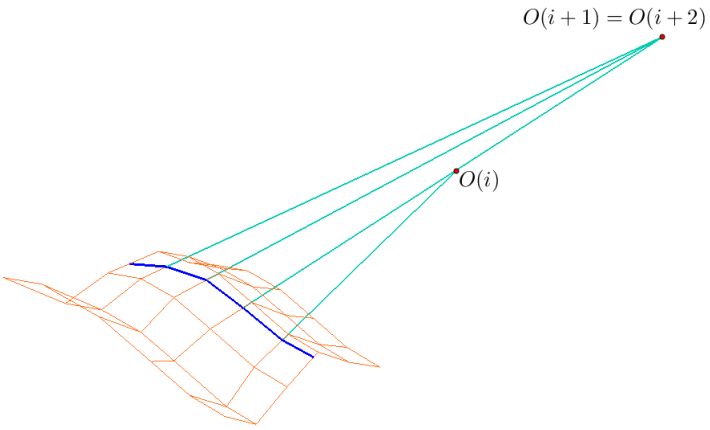
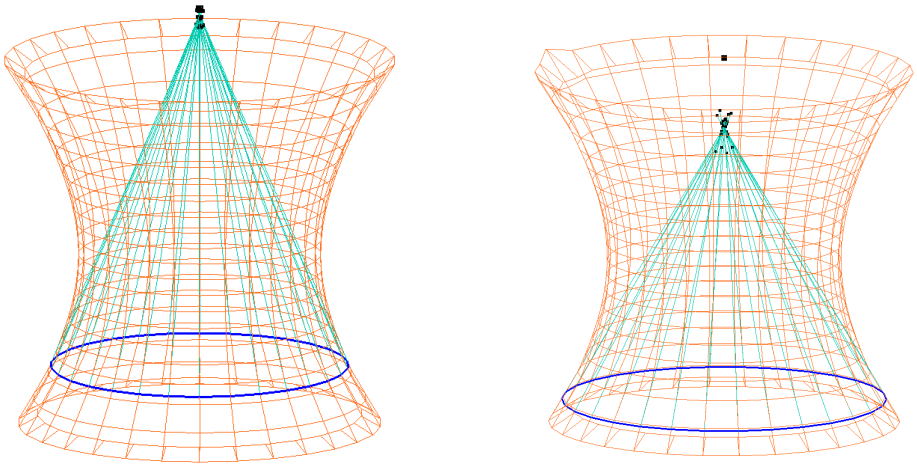


Figura 4.11: Exemplo - polígono observador

Agora que estabelecemos uma forma de encontrar um ponto observador (ou um polígono observador) e faremos algumas comparações entre as malhas cônicas, circulares e malhas. A comparação será feita com curvas que são aproximações de silhuetas(contida em uma faixa da malha), descritas na seção 4.3, avaliaremos o valor da função $\sigma(i)$ e o comprimento do polígono gerado.



4.12(a): Malha circular

4.12(b): Malha cônica

Figura 4.12: Interseções Campo de Darboux paralelo (em verde)

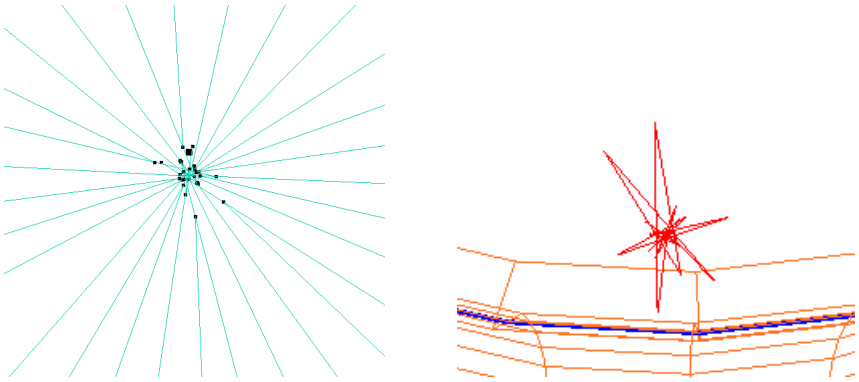
Como a malha é um objeto discreto é esperado que as interseções ocorram em mais de um ponto, porém estes devem estar próximos. Veja que em ambos

os casos temos vários pontos de interseção, estes formam um polígono observador e seu comprimento $G(C)$ fornecerá uma medida de qualidade da curva silhueta. Na tabela 4.1 temos as aproximações para o ponto de observação e a medida do comprimento $G(C)$ de cada polígono observador formado. Para esta curva o ponto observador está no ponto $(0, 0, 2)$

	Cônica	Circular
Valor de $G(C)$	0.157817	0.11038
Ponto observador	$(0, 0, 1.304)$	$(0, 0, 2.25)$

Tabela 4.1: Ponto de observação hiperbolóide.

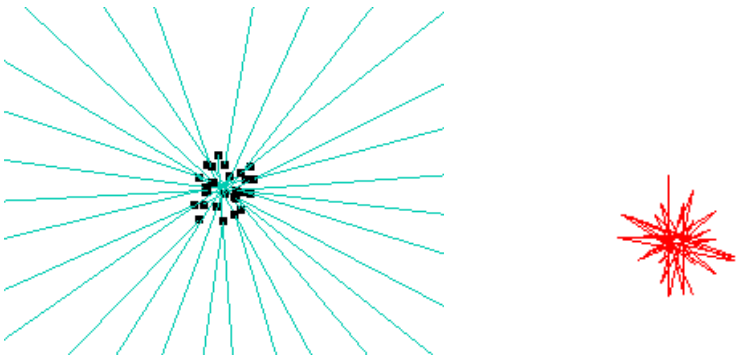
Podemos notar que na malha cicular o a medida de $G(C)$ é menor e também a aproximação utilizada está próxima do ponto real.



4.13(a): Interseções 4.13(b): Polígono

Figura 4.13: Polígono malha cônica - hiperbolóide

Na Figura 4.13 podemos ver o polígono observador na malha cônica.



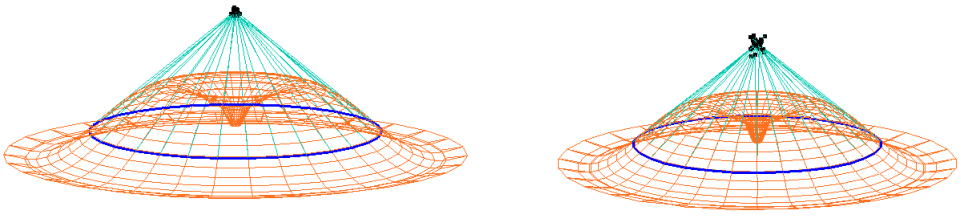
4.14(a): Interseções 4.14(b): Polígono

Figura 4.14: Polígono malha circular - hiperbolóide

Na Figura 4.14 temos os ponto de interseção e o polígono observador formado no caso da malha circular.

O resultado encontra-se melhor na malha circular, pois os pontos do polígono encontram-se mais próximos. No caso planar não é possível ver o polígono pois os pontos encontram-se muito próximos, isto pode ser percebido na avaliação $G(C)$ da curva.

Na Figura 4.15, uma malha gerada utilizando a parametrização $X(u, v) = (u\cos(v), u\sin(v), \sin(u) + 2)$ e a curva foi gerada com o ponto observador localizado em $(0, 0, 5)$



4.15(a): Malha circular 4.15(b): Malha cônica

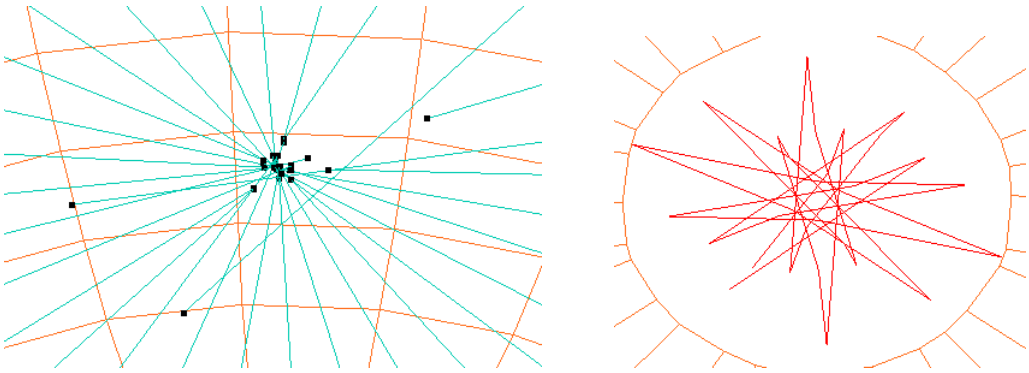
Figura 4.15: Interseções campo de Darboux paralelo - Exemplo 2

Na tabela 4.2, temos o resultados das medidas obtidas sobre a malha.

	Cônica	Circular
Valor de $G(C)$	0.19332	0.117436
Ponto observador	$(0.001, -0.003, 4.92)$	$(-0.002, 0.001, 4.9)$

Tabela 4.2: Ponto de observação - Exemplo 2.

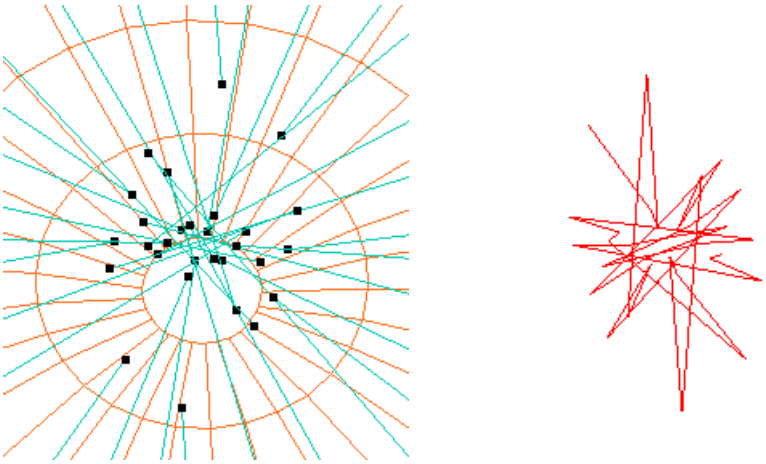
Na Figura 4.16, temos os pontos e o polígono gerado pelas interseções na malha circular.



4.16(a): Malha circular 4.16(b): Polígono

Figura 4.16: Polígono observador na malha circular - Figura 4.15

Na Figura 4.17, podemos ver as interseções na malha cônica e o seu polígono.

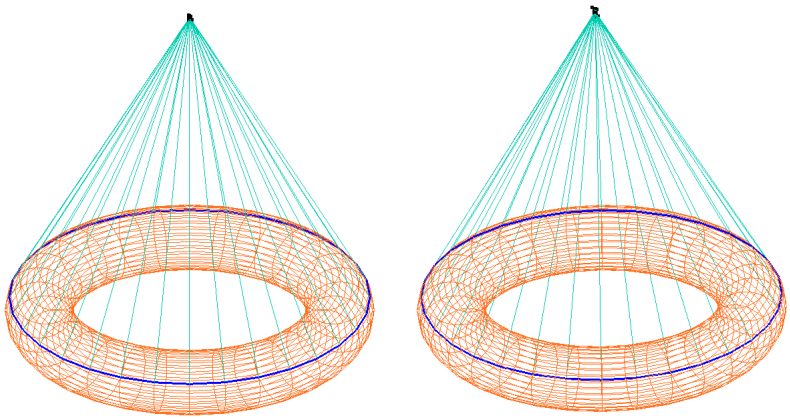


4.17(a): Malha cônica 4.17(b): Polígono

Figura 4.17: Polígono observador na malha cônica - Figura 4.15

No caso desta malha, podemos perceber que sua representante circular aproxima melhor o ponto de observação, pois nela o polígono tem comprimento menor em relação a malha cônica.

Na Figura 4.18, veremos as interseções do campo de Darboux paralelo no toro, a curva foi gerada utilizando o ponto observador $(0, 47, 0)$.



4.18(a): Malha circular 4.18(b): Malha cônica

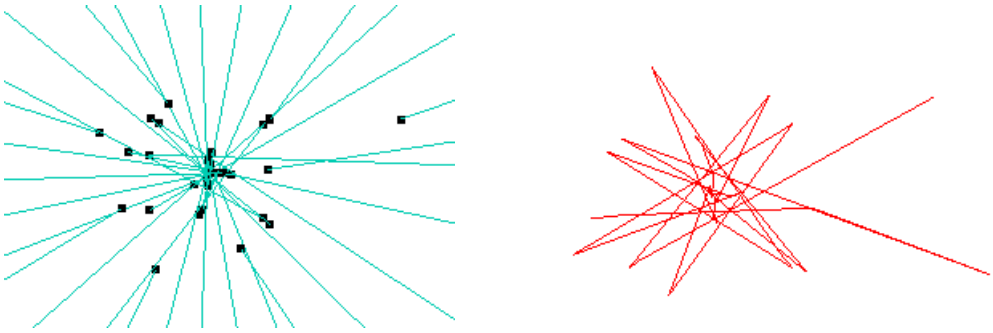
Figura 4.18: Interseções campo de Darboux paralelo - toro

Na tabela 4.3 temos os resultados da medida feitas e os pontos de observação obtidos.

	Cônica	Circular
Valor de $G(C)$	0.50369	0.292963
Ponto observador	$(0.007, 47.1, 0.002)$	$(0.0002, 47.08, 0.002)$

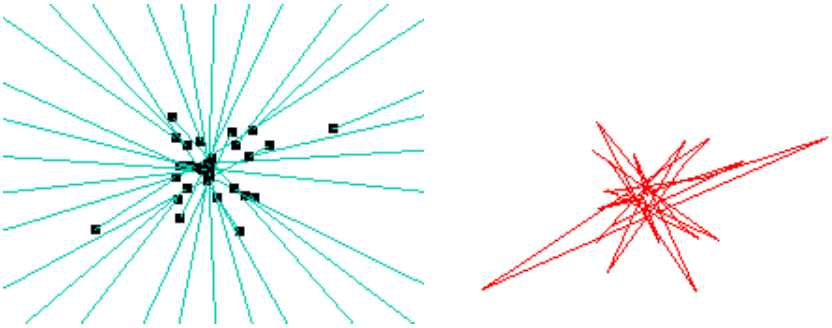
Tabela 4.3: Ponto de observação - toro.

Nas Figuras 4.19 e 4.20 podemos perceber o comportamento do polígono observador gerado nas malhas cônicas e circulares.



4.19(a): Pontos de interseção 4.19(b): Polígono

Figura 4.19: Polígono observador na malha circular - toro



4.20(a): Pontos de interseção 4.20(b): Polígono

Figura 4.20: Polígono observador na malha cônica - toro

Pelos resultados obtidos percebemos que em malhas circulares o polígono observador esta mas concentrado e proporciona um a melhor aproximação para o ponto observador. Como ultimo exemplo utilizaremos a malha do peixe que é planar, porém não foi obtida pela processo de otimização. Neste exemplo planar exibiremos o comportamento do polígono observador em malhas mais complexas.

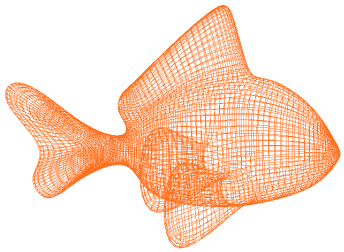
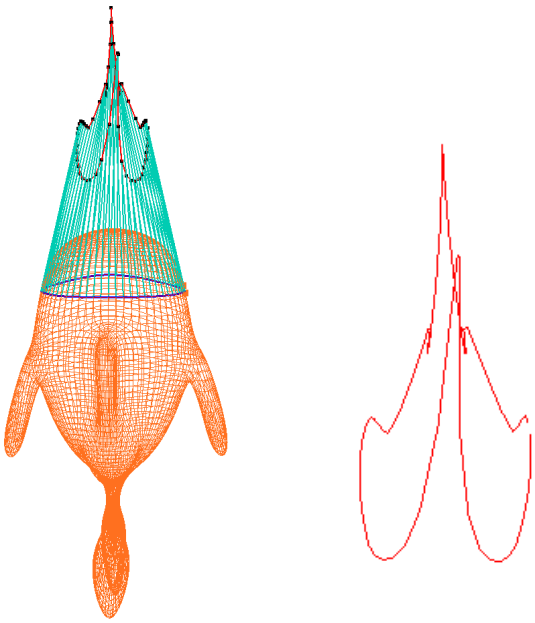


Figura 4.21: Malha Peixe

Para essa malha a curva é gerada sorteando uma face e percorreremos a faixa utilizando as Half-Edges. Na Figura 4.22 temos uma curva gerada sorteando uma face e o polígono formado pelas suas interseções.

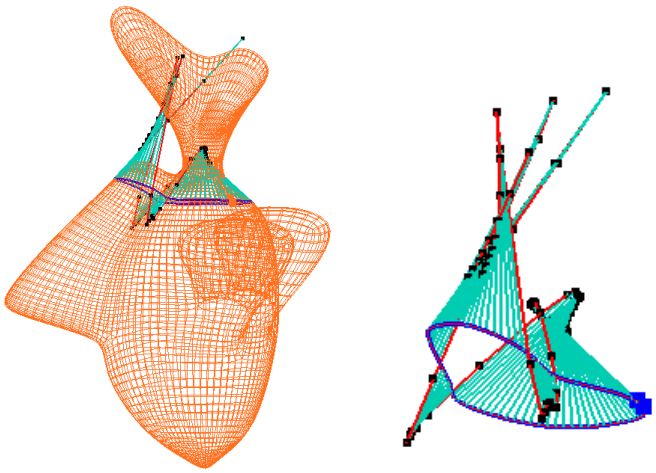


4.22(a): Pontos de interseção

4.22(b): Polígono observador

Figura 4.22: Interseções campo de Darboux paralelo - peixe curva 1

Na Figura 4.23, geramos uma nova curva utilizando outra face. Nesta Figura temos em azul a curva construída em verde o campo de Darboux paralelo e em vermelho o polígono gerado pelas interseções.



4.23(a): Pontos de interseção 4.23(b): Polígono

Figura 4.23: Interseções campo de Darboux paralelo - peixe curva 2

Na tabela 4.4 temos a avaliação da função $G(C)$.

Curva 1	0.0904303
Curva 2	0.0766663

Tabela 4.4: $G(C)$ no peixe.

O peixe é um bom exemplo de uma malha planar em que é possível ver seu polígono observador. Isto mostra que em malhas apenas planares as curvas contidas e faixas podem ter o um comprimento do polígono observador maior e não se aproximar de um ponto. Para estas curvas avaliamos a função $H(C)$. Na tabela 4.5 temos a avalia da função $H(C)$ nas curvas utilizadas nesta seção, sobre as malhas cônicas.

Malha	$H(C)$
hiperbolóide	0.558433
Figura 4.15	0.36705
Toro	0.22272

Tabela 4.5: Avaliação $H(C)$ malhas cônicas

Na Tabela 4.6 temos os resultados de $H(C)$ nas malhas circulares.

Malha	$H(C)$
hiperbolóide	0.211733
Figura 4.15	0.0849712
Toro	0.0109187

Tabela 4.6: Avaliação $H(C)$ malhas circulares

Avaliamos também a função $H(C)$ na malha planar do peixe (Tabela 4.7).

Malha	$H(C)$
Peixe curva 1	2.546228
Peixe curva 2	24.6164

Tabela 4.7: Avaliação $H(C)$ peixe

Em geral pelos resultados visto a malha circular gera um polígono observador com comprimento menor indicando que a curva esta mais próxima de uma silhueta real. Na avaliação da função $H(C)$ podemos notar melhores resultados em malhas circulares.

4.4.2 Processo de subdivisão

Nesta seção buscaremos aprimorar a silhueta utilizando um processo de subdivisão. Em alguns dos exemplos trabalhados, a curva pode não estar contida em faixas (ver Figura 4.24)

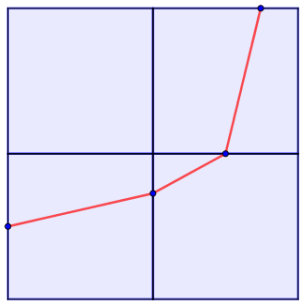
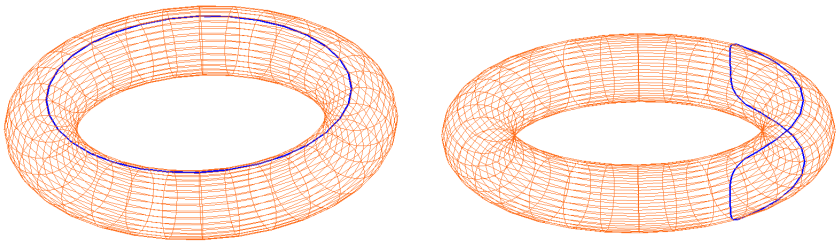


Figura 4.24: Silhueta fora de uma faixa.

Neste caso, estes valores serão retirados do cálculo e marcaremos a quantidade de vezes em que esta situação ocorre. Esperamos que utilizando um método de subdivisão, os resultados melhorem e desta forma a silhueta terá uma melhor qualidade. Para isso aplicaremos o método de Catmul-Clark [7], para refinar a malha em alguns exemplos. Iniciaremos medindo as curvas no Toro.



4.25(a): Curva - 1 4.25(b): Curva - 2

Figura 4.25: Curvas no Toro

Estas curvas (Figura 4.24) foram construídas utilizando um ponto de observação escolhido, para curva 1 escolhemos $(0, -47, 0)$ e para curva 2 o ponto $(-47, 0, 0)$. Para a curva 2 temos pontos em que a silhueta está fora da faixa, estes pontos serão retirados do cálculo e continuaremos a soma quando a curva voltar a estar dentro e uma faixa. Desta forma a curva a ser avaliada encontra-se na Figura 4.26.

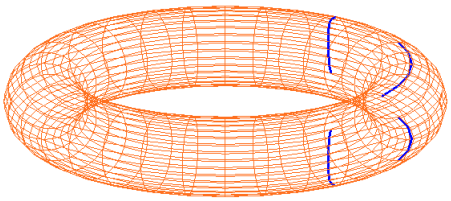


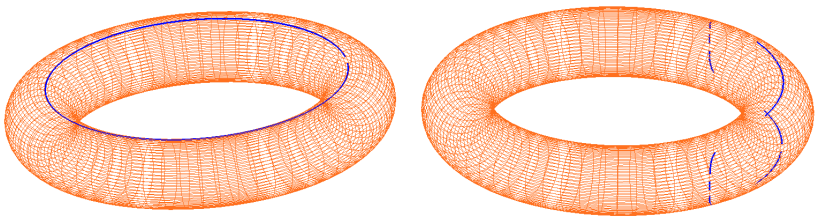
Figura 4.26: Silhueta avaliada em azul

A curva completa utiliza 51 pontos porém, apenas 23 estão dentro de faixas. Abaixo temos os resultados obtidos da avaliação destas duas curvas nas malhas cônicas e circulares.

malha	Cônica	Circular
curva 1	0.00311197	0.00489073
curva 2	0.00375565	0.00318893

Tabela 4.8: Valor de $H(C)$ antes da subdivisão.

Analisaremos agora o que ocorre quando aplicamos uma subdivisão na malha. Na Figura 4.27 temos as curvas a serem avaliadas depois da subdivisão. Utilizamos o mesmo observador para gerar as curvas depois da subdivisão.



4.27(a): Curva - 1 4.27(b): Curva - 2

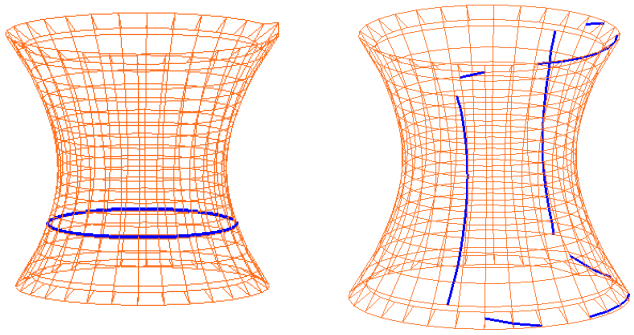
Figura 4.27: Curvas no Toro - subdivisão

Na curva dois foram utilizados 103 pontos porém apenas 42 estão dentro de alguma faixa. Na Tabela 4.9 temos o resultado da avaliação.

malha	Cônica	Circular
curva 1	0.00166762	0.00197128
curva 2	0.00371903	0.00356944

Tabela 4.9: Valor de $H(C)$ depois de 1 subdivisão - toro.

Agora avaliaremos algumas curvas no hiperbolóide. Utilizamos o observador no ponto $(0, 0, 3)$ para a curva 1 e $(0, 5, 0)$ para curva 2 (ver Figura 4.28).



4.28(a): Curva - 1 4.28(b): Curva - 2

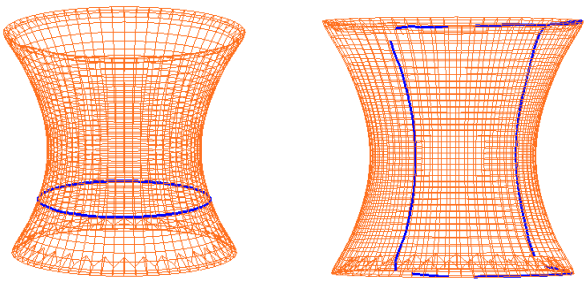
Figura 4.28: Curvas no hiperbolóide

Para a curva dois foram gerados 29 pontos e apenas 20 estão dentro de alguma faixa. Na tabela 4.10 temos os resultados nas malhas cônicas e circulares.

malha	Cônica	Circular
curva 1	0.227177	0.398607
curva 2	1.411325	2.0787

Tabela 4.10: Valor de $H(C)$ antes da subdivisão - hiperbolóide.

Aplicamos uma vez a subdivisão e analisaremos os resultados (Figura 4.29).



4.29(a): Curva - 1 4.29(b): Curva - 2

Figura 4.29: Curvas no hiperbolóide - subdivisão

Para estas novas curvas obtivemos os resultado apresentados na tabela 4.11

malha	Cônica	Circular
curva 1	0.0869319	0.146266
curva 2	0.426775	1.43718

Tabela 4.11: Valor de $H(C)$ depois de 1 subdivisão - hiperbolóide.

De forma geral, o processo de subdivisão reduziu o valor de H(C) nos dois exemplos verificados, com exceção do caso do toro na curva 2 - malha circular. Este resultado indica que se considerarmos uma maior amostragem de pontos sobre a silhueta, obteremos uma curva de melhor qualidade.

5
Conclusão e trabalhos futuros

Apresentamos dois tipos de malhas utilizados para a modelagem de estruturas de vidro. Descrevemos com detalhes um processo de otimização utilizado para gerar estas malhas e medidas de qualidade para as mesmas.

Definimos uma curva silhueta discreta pelo ponto de vista da geometria diferencial afim. Buscamos pontos de observação para alguns tipos de curvas sobre malhas planares e medimos a qualidade da curva utilizando uma relação entre campo de vetores.

Para trabalhos futuros podemos tentar definir o campo de vetores de Darboux paralelo sobre faces triangulares, permitindo assim avaliar curvas sobre qualquer tipo de malha.

Referências bibliográficas

[1] Cipolla, R. and Giblin, P. **Visual Motion of Curves and Surfaces**. 2000. Cambridge University Press.

[2] Crane, K. and Wardetzky, M. . **A Glimpse into Discrete Differential Geometry**. Notices of the American Mathematical Society 64.10 (2017).

[3] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. **Geometric modeling with conical meshes and developable surfaces**. 2006. In ACM SIGGRAPH 2006 Papers (SIGGRAPH '06). ACM, New York, NY, USA, 681-689.

[4] Craizer, M. and Pesco, S. **Affine geometry of equal-volume polygons in 3-space.** Computer Aided Geometric Design 57 (2017): 44-56.

[5] Lopes, H. and Tavares, G. **Structural operators for modeling 3-manifolds**. 1997. In Proceedings of the fourth ACM symposium on Solid modeling and applications (SMA '97). ACM, New York, NY, USA, 10-18.

[6] Mäntylä M. **An Introduction to Solid Modeling**. 1987. Computer Science Press, Inc., New York, NY, USA.

[7] Catmull, E.; Clark, J. (1978). **Recursively generated B-spline surfaces on arbitrary topological meshes** . Computer-Aided Design. 10 (6): 350. 1978

[8] Estrutura de vidro. Disponível em: seele.com. Acessado em 21/08/2018.

[9] Pottmann, Helmut and Wallner, Johannes (2008).**The focal geometry of circular and conical meshes**. Advances in Computational Mathematics 29(2008), 249-268.

[10] Bonnans, Joseph-Frédéric, et al. **Numerical optimization: theoretical and practical aspects**. Springer Science and Business Media, 2006.

[11] Manfredo do Carmo **Geometria diferencial de curvas e superfícies**. 2012. Rio de Janeiro, Brasil. Editora SBM.

[12] Davis, D. **Generic affine differential geometry of curves in $\mathbb{R}^n$**. Proceedings of the Royal Society of Edinburgh Section A: Mathematics 136.6 (2006): 1195-1205

[13] Izumiya, S. and Sano, T. **Generic affine differential geometry of plane curves**. 1998. Proceedings of the Edinburgh Mathematical Society, 41(2), 315-324.

[14] Craizer, M., Saia. M. J., Sánchez,L.F,(2016). **Equiaffine Darboux frames for codimension 2 submanifolds contained in hypersurfaces** Journal of the Mathematical Society of Japan. 69,1331-1352.

[15] Fundamentos Básicos de Computação Gráfica. Disponível em: www.miscelaneadoconhecimento.com. Acessado em 21/08/2018.

[16] Hoffmann, Tim **Discrete diferential geometry of curves and surfaces**. COE Lecture Note 18 (2009).

A
Códigos em MATLAB

A.1
f_{det}

```
function f_det()
for i = 1:mf

    v1 = [x((F(i,1)-1)*3+1),x((F(i,1)-1)*3+2),x((F(i,1)-1)*3+3)];

    v2 = [x((F(i,2)-1)*3+1),x((F(i,2)-1)*3+2),x((F(i,2)-1)*3+3)];

    v3 = [x((F(i,3)-1)*3+1),x((F(i,3)-1)*3+2),x((F(i,3)-1)*3+3)];

    v4 = [x((F(i,4)-1)*3+1),x((F(i,4)-1)*3+2),x((F(i,4)-1)*3+3)];

    e0 = v(v2,v1);

    e1 = v(v3,v2);

    e2 = v(v4,v3);

    e3 = v(v1,v4);

    g0 = acos((e0)*transpose(-e3));

    g1 = acos((-e0)*transpose(e1));

    g2 = acos((-e1)*transpose(e2));

    g3 = acos((-e2)*transpose(e3));

    a = det([e0;-e2;-e3]);    % c1
```

```
    b = det([e0;-e2;e1]);     % c

    c = det([e0;-e3;e1]);     % c3

    d = det([-e2;-e3;e1]);    % c4

    f = f + a*a + b*b + c*c + d*d;  % detrerminantes
end
```

A.2

f_{fair}

```
function f_fair();
ffair = 0;
 ang=0;
for i = 1:mv-1

a1 = [x(3*(i-1)+1),x(3*(i-1)+2),x(3*(i-1)+3)];
a2 = [x(3*(S(i,1)-1)+1),x(3*(S(i,1)-1)+2),x(3*(S(i,1)-1)+3)];
a3 = [x(3*(S(i,2)-1)+1),x(3*(S(i,2)-1)+2),x(3*(S(i,2)-1)+3)];
e1=v(a2,a1);
e1=e1/norm(e1);
e2=v(a3,a1);
e2=e2/norm(e2);

if S(i,3) == 0

a4 = [0,0,0];
k2=0;
else

a4 = [x(3*(S(i,3)-1)+1),x(3*(S(i,3)-1)+2),x(3*(S(i,3)-1)+3)];
e3=v(a4,a1);
e3=e3/norm(e3);
end

if S(i,4) == 0

a5 = [0,0,0];
```

```
else

a5 = [x(3*(S(i,4)-1)+1),x(3*(S(i,4)-1)+2),x(3*(S(i,4)-1)+3)];
e4=v(a5,a1);
e4=e4/norm(e4);

end
c = norm(a2-2*a1+a4);
d = norm(a3-2*a1+a5);

ffair = ffair + c*c + d*d;

end
```

A.3

f_{close}

```
function f_close()

fclose = 0;

for i = 0:mv-1 % proximidade da malha  ok

    v1 = [x(3*i+1),x(3*i+2),x(3*i+3)];

    v2 = [V(i+1,1),V(i+1,2),V(i+1,3)];

    fclose = fclose + norm(v1-v2)*norm(v1-v2);

end
```

A.4

f_{angle} **malha circular**

```
function f_angle()
for i = 1:mf

    v1 = [x((F(i,1)-1)*3+1),x((F(i,1)-1)*3+2),x((F(i,1)-1)*3+3)];

    v2 = [x((F(i,2)-1)*3+1),x((F(i,2)-1)*3+2),x((F(i,2)-1)*3+3)];
```

```
    v3 = [x((F(i,3)-1)*3+1),x((F(i,3)-1)*3+2),x((F(i,3)-1)*3+3)];

    v4 = [x((F(i,4)-1)*3+1),x((F(i,4)-1)*3+2),x((F(i,4)-1)*3+3)];

    e0 = v(v2,v1);

    e1 = v(v3,v2);

    e2 = v(v4,v3);

    e3 = v(v1,v4);

    g0 = acos((e0)*transpose(-e3));

    g1 = acos((-e0)*transpose(e1));

    g2 = acos((-e1)*transpose(e2));

    g3 = acos((-e2)*transpose(e3));

    % circular
e1=(g0 + g2 - pi) * (g0 + g2 - pi);
e2=(g1 + g3 - pi)*(g1 + g3 - pi);
       h = h + (e1*e1)+(e2*e2);
end
```

A.5 f_{angle} malha Cônica

```
function f_angle()
for i = 1:mv-1

a1 = [x(3*(i-1)+1),x(3*(i-1)+2),x(3*(i-1)+3)];

% primeiro vizinho
a2 = [x(3*(S(i,1)-1)+1),x(3*(S(i,1)-1)+2),x(3*(S(i,1)-1)+3)];

% segundo vizinho
a3 = [x(3*(S(i,2)-1)+1),x(3*(S(i,2)-1)+2),x(3*(S(i,2)-1)+3)];
```

```
e1=v(a2,a1);
e1=e1/norm(e1);
e2=v(a3,a1);
e2=e2/norm(e2);

 k1=acos(e1*transpose(e2));
if S(i,3) == 0

a4 = [0,0,0];
k2=0;
else
% terceiro vizinho
a4 = [x(3*(S(i,3)-1)+1),x(3*(S(i,3)-1)+2),x(3*(S(i,3)-1)+3)];
e3=v(a4,a1);
e3=e3/norm(e3);
 k2=acos(e2*transpose(e3));
end

if S(i,4) == 0

a5 = [0,0,0];
k3=pi/2;
k4=pi/2;
else
% quarto vizinho
a5 = [x(3*(S(i,4)-1)+1),x(3*(S(i,4)-1)+2),x(3*(S(i,4)-1)+3)];
e4=v(a5,a1);
e4=e4/norm(e4);
k3=acos(e3*transpose(e4));
k4=acos(e4*transpose(e1));

end

e=(k1+k3)-(k2+k4);
ang = ang + e*e;

end
```