CEEL - ISSN 2596-2221
Universidade Federal de Uberlândia
05 a 09 de Dezembro de 2022

COMPARAÇÃO ENTRE MODELOS TRIDIMENSIONAIS UTILIZANDO A DISTÂNCIA DE HAUSDORFF

Edgard Afonso Lamounier Júnior[1], Gabriel Alves Caixeta Custódio[1]

[1] FEELT – Universidade Federal de Uberlândia

***Resumo* – Em projetos que utilizam uma biblioteca grande de objetos modelados em três dimensões, como subestações de energia elétrica, simulações e jogos eletrônicos, é possível que objetos bem parecidos, semelhantes ou duplicados ocupem espaço desnecessário na biblioteca.**

Assim, a existência de um método que consiga comparar modelos tridimensionais é extremamente útil, pois é mais eficiente que a comparação visual e facilita a detecção de modelos clonados.

Diante dessa questão, o objetivo deste trabalho é o estudo de uma ferramenta que compara a amostra de duas de malhas através do cálculo da distância de Hausdorff e discutir a viabilidade do programa em uma situação parecida com a citada acima.

***Palavras-Chave* – Distância de Hausdorff, Amostragem de malhas, Modelos 3D, Comparação.**

COMPARISON BETWEEN THREE-DIMENSIONAL MODELS USING HAUSDORFF DISTANCE

***Abstract* – In projects that use a large library of three-dimensional objects, such as electrical substation, simulation and videogames, it's possible that similar or duplicated objects take unecessary space in the library.**

So, the existence of a method that compares three-dimensional models is extremely useful, because it is more eficiente than a visual comparison and make easy the detection of clone models.

Faced with this question, the goal of this article is the study of a tool that compares two meshes sampling with calculation of Hausorff distance and the program viability in a similar situation previously cited.

***Keywords* – Hausdorff distance, Meshes sampling, 3D models, Comparison.**

I. INTRODUÇÃO

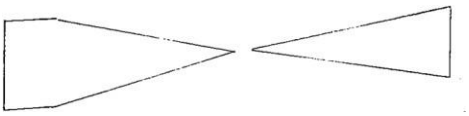
Quando falamos sobre distâncias entre polígonos convexos, e queremos encontrar a menor distância entre eles, o que geralmente fazemos é calcular a distância do vértice pertencente ao polígono A mais próximo do polígono B até o vértice pertencente a B mais próximo de A, e o mesmo vale para conjuntos que pode ser escrito na seguinte relação lógica [1]:

$$D(A, B) = \min_{a\epsilon A}(\min_{b\epsilon B}(d(a, b))) \quad (1)$$

Sendo D(A,B) a menor distância entre os polígonos ou conjuntos A e B e d(a,b) a distância entre o ponto ou vértice *a* pertencente a A até o ponto ou vértice *b* pertencente a B.

Porém, essa métrica é insatisfatória, quando passamos a considerar o conjunto todo, como no exemplo a seguir:

Figura 1: dois polígonos, utilizados para a compreensão da motivação para o uso da distância de Hausdorff [2]



Observando a menor distância entre os polígonos, podemos dizer que eles estão próximos, mas se considerarmos os pontos mais distantes entre eles, podemos ver que a métrica de menor distância entre dois polígonos não diz nada a respeito da posição entre eles.

A distância de Hausdorff, pode nos dar uma ideia melhor sobre isso, e é definida como a distância máxima de um conjunto até o ponto de um segundo conjunto mais próximo do primeiro [1], e tem esse nome em homenagem ao matemático Félix Hausdorff (1868-1942). A distância de Hausdorff direcionada de um conjunto ou polígono A para B é denotada por:

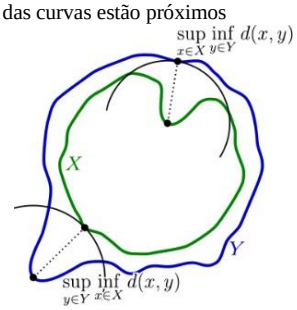
$$h(A, B) = \max_{a\epsilon A}(\min_{b\epsilon B}(d(a, b))) \quad (2)$$

Um ponto importante é que a distância de Hausdorff não é simétrica, ou seja, *h(A, B)* é diferente de *h(B, A)*, dessa forma, para obtermos a distância de Hausdorff *entre* A e B, deve ser feito o cálculo das duas distâncias direcionadas (A para B e B para A) e utilizar o maior valor entre elas [6]:

$$H(A, B) = \max(h(A, B), h(B, A)) \quad (3)$$

Quando as extremidades dos conjuntos estão bem próximas, essa distância tende a diminuir, como podemos ver entre o conjunto X e Y na Figura 2:

Figura 2: Demonstração da distância de Hausdorff quando os pontos das curvas estão próximos



Fazendo uma análise lógica, quanto mais próximos estiverem dois polígonos, a distância de Hausdorff entre eles tende a ser zero, dessa forma, quando os dois polígonos são iguais e estão alinhados, essa distância será igual a zero.

Baseado nisso, podemos também usar esta métrica para fazer a comparação entre dois modelos tridimensionais que estiverem alinhados, realizando o cálculo desta distância entre as duas malhas, que são compostas por polígonos. [7]

Dessa forma, este artigo tem como objetivo a pesquisa e um estudo de uma ferramenta ou método de comparação de modelos tridimensionais utilizando a distância de Hausdorff.

II. METODOLOGIA

Para o estudo e análise da comparação entre dois modelos tridimensionais, optamos por utilizar a ferramenta Metro [3], pois ela utiliza o cálculo da distância de Hausdorff entre duas malhas triangulares, incluindo os vértices, arestas e faces, indicando o nível de similaridade entre elas, além disso, a documentação deste programa já foi citada mais de 500 vezes [7], que mostra a confiabilidade e relevância deste método.

O Metro é um software livre, desenvolvido e distribuído gratuitamente, disponível no repositório da Visual and Computer Graphics Library (VCGLib) [3] no github, o programa é quase inteiramente desenvolvido nas linguagens C e C++.

A utilização básica do Metro é feita através do prompt de comando do Windows ou terminal do Linux ou macOS. O comando é feito em uma linha única, que tem a seguinte estrutura:

Metro file1 file2 [opts]

Onde:

file1 e *file2* são dois arquivos de entrada contendo modelos tridimensionais, e só são aceitos os formatos OFF, PLY e STL, se o usuário utilizar um formato diferente, uma mensagem de erro 'Unknown type' será mostrada;

[opts] são parâmetros opcionais, dentre os quais, destacamos os seguintes:

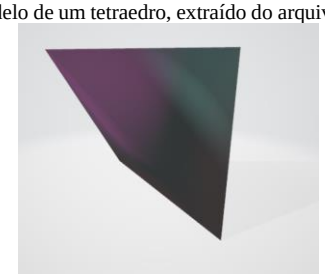
-v - Desabilita a amostragem das distâncias de entre os vértices das malhas

-e - Desabilita a amostragem das arestas

-f - Desabilita a amostragem das faces

No primeiro teste, utilizamos o modelo de um tetraedro para demonstrar o funcionamento da ferramenta.

Figura 3: Modelo de um tetraedro, extraído do arquivo tetraascii.ply



Ao fazer a chamada do arquivo no comando Metro tetrascii.ply tetrascii.ply , o programa é executado, resultando na seguinte saída, que será dividida em etapas para melhor explicação:

Figura 4: Saída de Metro tetrascii.ply tetrascii.ply (1/3) no prompt de comando do Windows

```
C:\Users\gabri\Desktop\hausdorff>Metro tetraascii.ply tetraascii.ply
------------------------------
        Metro V.4.07
    http://vcg.isti.cnr.it
  release date: May 11 2007
------------------------------

read mesh `tetraascii.ply'
read mesh `tetraascii.ply'
Mesh info:
 M1: 'tetraascii.ply'
       vertices        4
       faces           4
       area          13.8564
       bbox (-1.0000 -1.0000 -1.0000)-( 1.0000  1.0000  1.0000)
       bbox diagonal 3.464102
 M2: 'tetraascii.ply'
       vertices        4
       faces           4
       area          13.8564
       bbox (-1.0000 -1.0000 -1.0000)-( 1.0000  1.0000  1.0000)
       bbox diagonal 3.464102
```

Primeiramente, o programa lê as duas malhas fornecidas na entrada e mostra detalhes e informações sobre cada uma delas, como o número de vértices, faces e área, como os dois arquivos são o mesmo, os dados são iguais (Figura).

Na próxima etapa (Figura), o programa realiza o cálculo da distância de Hausdorff da malha 1 para a malha 2, e depois o cálculo na direção contrária, da malha 2 para a malha 1, pois a distância de Hausdorff pode não ser simétrica, mas, neste caso, como utilizamos o mesmo modelo para as duas entradas, o resultado dos dois cálculos é o mesmo.

Figura 5: Saída de Metro tetrascii.ply tetrascii.ply (2/3) pelo prompt de comando do Windows

```
Forward distance (M1 -> M2):
target # samples      : 40
target # samples/area : 2.886751
Vertex sampling
Edge sampling
Similar Triangles face sampling

distances:
  max  : 0.000000 (0.000000  wrt bounding box diagonal)
  mean : 0.000000
  RMS  : 0.000000
# vertex samples         4
# edge samples          27
# area samples           4
# total samples         35
# samples per area unit: 2.525907


Backward distance (M2 -> M1):
target # samples      : 40
target # samples/area : 2.886751
Vertex sampling
Edge sampling
Similar Triangles face sampling

distances:
  max  : 0.000000 (0.000000  wrt bounding box diagonal)
  mean : 0.000000
  RMS  : 0.000000
# vertex samples         4
# edge samples          27
# area samples           4
# total samples         35
# samples per area unit: 2.525907
```

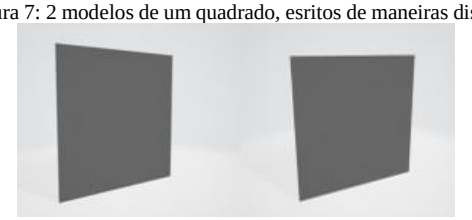
A saída final do código resulta no valor máximo entre os dois cálculos, que é a distância de Hausdorff entre as duas malhas. O resultado neste teste é esperado, pois sabemos que os dois arquivos são iguais, logo, a distância de Hausdorff deve ser zero (Figura). Além disso, é mostrado o tempo de execução do comando e o número de amostras por segundo realizadas. Nos próximos testes, serão mostradas apenas as saídas finais, onde estão os dados necessários para este estudo.

Figura 6: Saída de Metro tetrascii.ply tetrascii.ply (3/3) pelo prompt de comando do Windows

```
Hausdorff distance: 0.000000 (0.000000  wrt bounding box diagonal)
  Computation time  : 73 ms
  # samples/second  : 958.904064
```

A seguir, realizamos o teste com dois modelos idênticos de quadrados, porém, escritos de forma diferente:

Figura 7: 2 modelos de um quadrado, esritos de maneiras distintas [2]



Fazemos o teste utilizando o comando Metro quad.ply quad2.ply, arquivis extraídos do repositório do VCGLib. Ao abrirmos os dois arquivos em um editor de texto, podemos ver que o conteúdo dos dois são diferentes:

Figura 8: Arquivo de texto de quad.ply

```
ply
format ascii 1.0
comment VCGLIB generated
element vertex 4
property float x
property float y
property float z
element face 2
property list uchar int vertex_indices
end_header
-1 -1 0
 1 -1 0
 1  1 0
-1  1 0
3 0 1 2
3 0 2 3
```

Figura 9: Arquivo de texto de quad2.ply

```
ply
format ascii 1.0
comment VCGLIB generated
element vertex 5
property float x
property float y
property float z
element face 4
property list uchar int vertex_indices
end_header
-1 -1 0
 1 -1 0
 1  1 0
-1  1 0
0 0 0
3 0 1 4
3 1 2 4
3 2 3 4
3 3 0 4
```

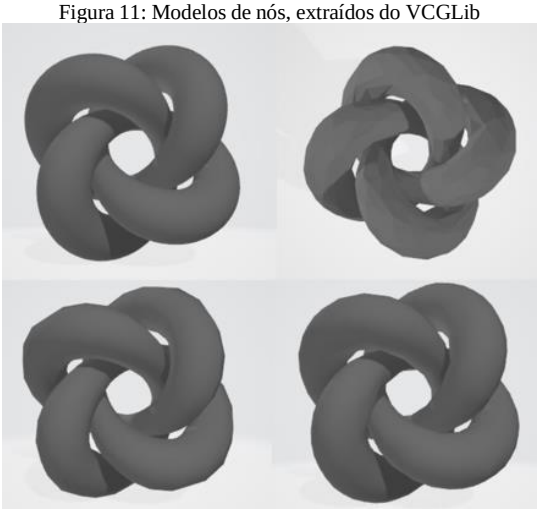
Figura 10: Resultado final do comando Metro quad.ply quad2.ply

```
Hausdorff distance: 0.000000 (0.000000  wrt bounding box diagonal)
  Computation time  : 61 ms
  # samples/second  : 1327.868789
```

Como podemos ver, o resultado do índice de Hausdorff é zero, o que significa que o programa consegue identificar a semelhança entre modelos com formas iguais, mas topologias diferentes, ou seja, propriedades geométricas diferentes que resultam na mesma malha.

O próximo teste é feito com modelos semelhantes, que são criados a partir da simplificação de um deles (knot_orig.ply). Nestes, casos, a simplificação da topologia altera o formato das malhas, assim, os quatro modelos seguintes não são completamente idênticos:

Figura 11: Modelos de nós, extraídos do VCGLib



O primeiro nó é o original, enquanto os outros três são simplificações diferentes dele, e, o segundo modelo também está no formato STL (da esquerda para direita, de cima para baixo).

Figura 12: Resultados das comparações entre as simplificações e o modelo original

```
Hausdorff distance: 2.229879 (0.008226  wrt bounding box diagonal)
  Computation time  : 411 ms
  # samples/second  : 1202919.650894

Hausdorff distance: 2.300590 (0.008486  wrt bounding box diagonal)
  Computation time  : 417 ms
  # samples/second  : 1195170.207021

Hausdorff distance: 1.671167 (0.006162  wrt bounding box diagonal)
  Computation time  : 446 ms
  # samples/second  : 1117486.494007
```

Estes são os resultados do comando do metro para o cálculo da distância de Hausdorff entre o primeiro modelo e suas simplificações. Como podemos ver, o resultado não é nulo, pois a mudança na topologia feita também alterou a forma da malha, logo, os dois modelos são todos diferentes, porém, semelhantes.

O último caso é um teste feito com modelos completamente diferentes e com uma maior complexidade (maior número de vértices, arestas e faces). O primeiro modelo (Figura) tem o formato de uma mão, extraído de Hand.ply [4], enquanto o segundo tem a forma de um pé, extraído do arquivo Foot.ply [5] (Figura).

Figura 13: Modelo 3D de uma mão [3]

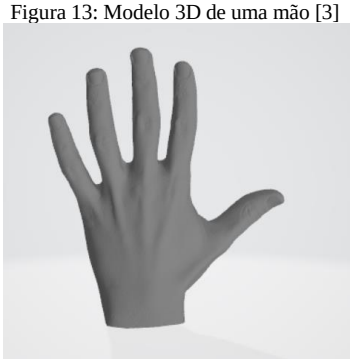


Figura 14: Modelo 3D de um pé [4]

Utilizando o comando no terminal Metro Hand.ply Foot.ply -e -f , obtemos um resultado (Figura) que corresponde às expectativas, pois os modelos são completamente diferentes, mas o mais notável é o tempo de processamento para esse cálculo, que leva vários minutos, mesmo desabilitando a amostragem de arestas e faces.

Figura 15: Resultado final de Metro Hand.ply Foot.ply -e -f

```
Hausdorff distance: 184.891327 (0.398561  wrt bounding box diagonal)
  Computation time  : 559616 ms
  # samples/second  : 892.308255
```

III. CONCLUSÃO

Após os testes realizados, podemos concluir que realmente o cálculo da distância de Hausdorff pode ser utilizado para fazer a comparação entre dois modelos tridimensionais, e o programa Metro é funcional. Através dos primeiro e segundo casos, podemos ver que o resultado é aquele esperado (nulo) quando comparamos dois modelos iguais, e, no segundo caso, é mostrado que isso independe da topologia e da forma como foi escrito o arquivo, apenas da forma da malha, um ponto positivo para a ferramenta.

No terceiro caso, podemos observar que a resolução a partir da simplificação afeta na comparação pelo Metro, o que é compreensível, pois claramente há uma mudança visual entre os modelos. Além disso, um dos modelos simplificados está no formato STL, diferente do formato PLY como o original, mas não houve nenhum problema durante o cálculo do índice, o que mostra que podemos utilizar formatos diferentes para isso. O problema é que esta versão do Metro só aceita três tipos de formato (OFF, STL e PLY) e não consegue comparar utilizando outros formatos bastante utilizados como FBX, GITF e MAX.

No último caso, utilizando modelos mais complexos, podemos reparar que o resultado da distância de Hausdorff foi um número grande, mas também é esperado, já que os dois modelos têm poucas semelhanças entre si, além disso, é notável que o tempo de processamento do programa levou alguns minutos, mesmo utilizando apenas amostragem de vértices, e a tendência, é que o tempo aumente conforme a complexidade do objeto.

Outro ponto importante, é que o Metro facilita a comparação entre os arquivos sem precisar abri-los e realizar uma comparação visual humana, mas esse processo só pode ser feito em pares. Por isso, se o usuário deseja fazer a comparação entre vários modelos de uma biblioteca, é necessário a criação de um script que possa realizar vários comandos do Metro automaticamento, ou então, fazer uma pré-seleção manual para reduzir a biblioteca. Outro porém, é que a complexidade dos modelos também deve ser levada em consideração ao fazer várias comparações em grande escala, o que levaria um tempo cada vez maior, que pode tornar a utilização da ferramenta inviável em certos casos.

REFERÊNCIAS

[1] N. Grégoire, M. Bouillot, *Hausdorff distance between convex polygons*. Acedido em: 24 de setembro de 2022,em: http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html

[2] M. J. Atallah (1983). *A linear time algorithm for the Hausdorff distance between convex polygons*, Information Processing Letters, v.17 p 207-209.

[3] P. Cignoni, C. Rocchini, R. Scopigno, *Metro: measuring error on simplified surfaces*, Computer Graphics Forum, Blackwell Publishers, vol. 17(2), junho de 1998, p. 167-174. Disponível em: http://vcg.sf.net

[4] "Hand" (https://skfb.ly/osFzZ) by 1-3D.com is licensed under Creative Commons Attribution-ShareAlike (http://creativecommons.org/licenses/by-sa/4.0/).

[5] "Foot Ankle Model" (https://skfb.ly/ooLPJ) by bzhang2021 is licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/).

[6] I.-S. Kim, W. McLean, *Computing the Hausdorff distance between two sets of parametric curves*, Communications of Korean Mathematical Society, publicado em 31 de outubro de 2013, acedido em 25 de setembro de 2022 em: http://koreascience.or.kr/article/JAKO201334064306689.page

[7] *Measiring the difference between two meshes*, Meshlan Stuff, publicado em 10 de janeiro de 2010, disponível em:

http://meshlabstuff.blogspot.com/2010/01/measuring-difference-between-two-meshes.html?m=1, acedido em 25 de setembro de 2022.