PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Renderização baseada em pontos aplicada a seções geológicas

Miguel Basso Sanseverino

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Graduação em Ciência da Computação

Rio de Janeiro, Junho, 2022

Miguel Basso Sanseverino

Renderização baseada em pontos aplicada a seções geológicas

Relatório de Projeto Final, apresentado ao programa de Ciência da Computação da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Waldemar Celes Filho

Rio de Janeiro

Junho de 2022

“Se eu me ater ao ontem, o que será de mim amanhã?
Não importa.
Vou me desafiar hoje.”

Haruichi Furudate

Agradecimentos

Agradeço a minha família, pelo suporte incondicional que me foi dado durante a minha formação, pela motivação para superar as adversidades, e pela segurança dada para experimentar e buscar novos caminhos.
Agradeço também a meus amigos, que me motivaram a continuar indo em frente, e com quem compartilhei incontáveis momentos de felicidade.
À minha namorada, que me acompanhou e me apoiou mais que ninguém durante a faculdade.
À toda equipe do grupo MGEO do Tecgraf, que me auxiliou durante o desenvolvimento deste trabalho.

Resumo

Sanseverino, Miguel, Waldemar Celes Filho. Renderização baseada em pontos aplicada a seções geológicas. Rio de Janeiro, 2022. 28 páginas. Relatório Final de Projeto Final – Centro Técnico Científico, Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

O ReconMS é um software de modelagem de superfícies geológicas que emprega análises numéricas com nuvens de pontos. Este trabalho experimenta técnicas de visualização da superfície dessa nuvem de pontos, renderizando seus pontos como círculos com transparência variável, com o intuito de compor cores de regiões baseadas em alguns de seus atributos. Também são experimentadas técnicas de visualização volume dessa nuvens de pontos, aplicando funções de transferência para compor o volume da nuvem de acordo com a densidade da distribuição de suas propriedades. Os pontos são ordenados a cada frame na CPU para realizar a composição das transparências. A combinação das técnicas propostas gera uma visualização rica e detalhada da seção geológica, com desempenho interativo e que permite uma melhor exploração dessa nuvem de pontos.

Palavras-chave

Nuvem de pontos, renderização, volume, superfície, computação gráfica, seção geológica.

Abstract

Sanseverino, Miguel, Waldemar Celes Filho. Point based rendering based on geological cross sections. Rio de Janeiro, 2022. 28 pages. Centro Técnico Científico, Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

ReconMS is a geological surface modeling software that uses numerical analysis with point clouds. This work experiments with different surface visualization techniques applied in this point cloud, rendering each point as a circle with variable transparency and blending the colors of the points based on some of their attributes. A volume rendering technique is also experimented, using transfer functions to create the volume of the cloud, according to the given density of each property. The points are sorted in every CPU frame, in order to enable the transparency blending. The combination of both techniques creates a rich and highly detailed view of the geological cross section, in interactive time, enabling a better exploration of its data.

Keywords

Point cloud, rendering technique, volumetric render, surface render, computer graphics,

geological cross section, geological surface modeling.

1. Introdução

1.1 Motivação

O ReconMS [1] é um software, desenvolvido pelo Instituto Tecgraf da PUC-Rio, com o objetivo de realizar restaurações e transformações em seções geológicas, possibilitando a análise geológica dessas seções restauradas. O software possibilita diferentes métodos de visualização, incluindo visualização em 3D de uma série de seções arranjadas.

As restaurações e transformações geológicas são viabilizadas com a criação de uma malha de triângulos representando cada uma das diferentes camadas da seção. Essa malha é capaz de armazenar as informações que as transformações e restaurações infligem.

Além da apresentação de superfícies usando malhas de triângulos, o ReconMS usa um método numérico baseado em partículas para modelar reconstituições geológicas. O desafio do projeto foi possibilitar a visualização dessa nuvem densa de partículas. O objetivo do trabalho foi investigar e implementar técnicas de visualização 3D baseada em pontos (Point Based Rendering),

1. 2 Objetivos

O objetivo deste trabalho é desenvolver um programa que implemente métodos de visualização de pontos que serão usados para nortear uma atualização da visualização atual da nuvem de pontos dentro ReconMS. A implementação foi feita utilizando *OpenGL*, API gráfica na qual o ReconMS é desenvolvido, responsável por desenhar os pontos na tela e possibilitar a interação com a nuvem de pontos. A versão do OpenGL utilizada é a 4.5, com a utilização de *shaders* (*vertex e fragment)*.

Foi feito um estudo das limitações da visualização atual da nuvem de pontos no ReconMS, bem como um levantamento dos requisitos para uma visualização mais informativa e eficaz. Após isso, foram estudadas diferentes técnicas de renderização em 3D baseada em pontos, que servirão de base ou serão utilizadas parcialmente no trabalho, que incluem Renderização de Volumes e Renderização de Superfícies baseadas em pontos.

Foi feito uma experimentação desses diferentes métodos utilizando os dados exportados do ReconMS como base, a fim de determinar as especificidades da implementação do método junto com a sua fase inicial de desenvolvimento. Então, após a primeira versão, foi feita uma validação, análise dos resultados alcançados e, após isso, o desenvolvimento da versão final.

O escopo deste trabalho limita-se a renderizar a nuvem de pontos sem a implementação de uma estrutura de dados complexa, tendo em vista que o ReconMS possui as próprias estruturas que acoplam a nuvem de pontos, e que posteriormente a este trabalho podem ser adaptadas e otimizadas.

1.3 Contribuição

A visualização dessa nuvem de pontos dentro ReconMS é feita desenhando cada ponto como um pixel de tamanho 1, e de mesma cor. Essa forma de visualização é pouco ou nada informativa, e não diferencia os pontos pelas camadas das quais fazem parte, tão pouco os distingue por qualquer um de seus atributos (Figura 1).

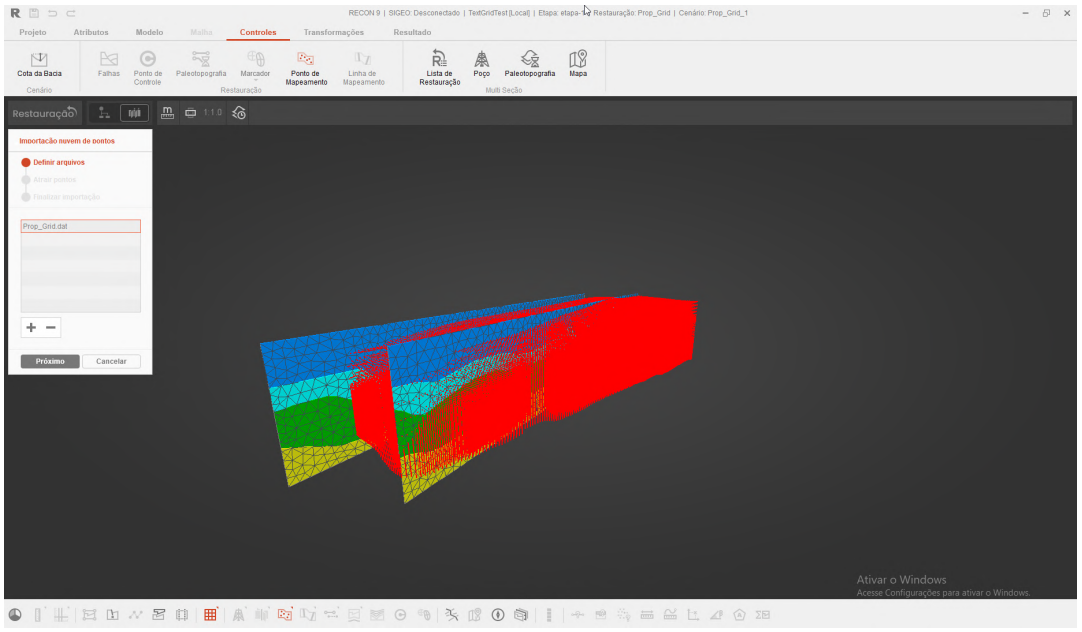

Figura 1. *Visualização atual da nuvem de pontos (em vermelho) dentro do ReconMS.*

Durante a realização deste trabalho, foram propostos diferentes métodos para a implementação da visualização da nuvem de pontos. Para cada método, foram realizados experimentos, visualizando diferentes atributos dentre os escolhidos pelos geólogos que participam do desenvolvimento do ReconMS. Esses experimentos foram utilizados para entender as relações entre os atributos, e como podem ser usados para formar uma visualização rica e detalhada da seção geológica.

A contribuição deste trabalho não se limita a nortear melhores técnicas de visualização. A visualização desta nuvem de pontos com ênfase em suas principais características pode possibilitar análises mais aprofundadas, que por sua vez podem gerar novas demandas de pesquisa e usabilidade.

1.4 Ambiente de Desenvolvimento

O trabalho foi desenvolvido visando investigar e implementar técnicas de visualização, de tal forma que as técnicas possam ser replicadas dentro do ambiente de visualização do ReconMS.

A implementação das técnicas foi realizada em um computador *Windows*, com uma placa de vídeo NVIDIA RTX 3050 e uma CPU i7 7700k. A linguagem utilizada para a implementação geral do trabalho foi C++, principal linguagem destinada para o desenvolvimento de aplicações gráficas utilizando OpenGL. Além disso, foi utilizado a API gráfica OpenGL para realizar a renderização. Para implementação das interfaces gráficas de usuário, foi utilizado a biblioteca ImGui. Por fim, foi utilizado a biblioteca GLM para realização de cálculos matemáticos.

2. Trabalhos Relacionados

A visualização de nuvens de pontos é um tema recorrente na área de computação gráfica. Nessa linha de pesquisa, existem diferentes enfoques para a visualização de dados em uma nuvem de pontos. Este trabalho baseia-se principalmente em duas técnicas de renderização. A primeira técnica em que este trabalho se baseia foi proposta para renderizar superfícies de malhas de triângulos detalhadas [9] , utilizando primitivas da API gráfica utilizada de baixo custo de renderização. A segunda técnica utilizada como base tem como foco a renderização de volumes de modelos [10], utilizando propriedades internas da nuvem para determinar a densidade daquela região, através de uma função de transferência [4].

Para criar e organizar o ambiente de desenvolvimento, foram usados como base *Learn OpenGL* [3], que implementam elementos essenciais para fundamentar o desenvolvimento, como a criação de uma cena simples usando OpenGL e GLFW, e criação das abstrações dos shaders.

Por fim, com o objetivo de reduzir o escopo deste trabalho à implementação e experimentação das técnicas de visualização, foi utilizado o software *Cloud Compare* [1], dedicado à visualização e manipulação de nuvens de pontos , para realizar os cálculos das normais e validar os testes iniciais de visualização.

3. Proposta

3.1 Configuração do Ambiente de Desenvolvimento

3.1.1 Ambiente

Para auxiliar na implementação e no desenvolvimento deste trabalho, foi configurado um ambiente utilizando a IDE *Microsoft Visual Studio 2019*, em que foi preparada uma cena simples utilizando OpenGL moderno, versão 4.6, *fragment* e *vertex shaders* e uma câmera *ArcBall.* Foram criadas classes para possibilitar esta visualização, através do OpenGL.

A classe *Shader* foi implementada seguindo o tutorial em *Learn OpenGL/Shaders*, de forma a receber os caminhos para os arquivos do *fragment shader* e do *vertex shader*, e gerar o programa utilizado pelo OpenGL. Essa implementação também adiciona o tratamento de erros dentro dos shaders.

3.1.2 Configuração dos Buffers

Para renderizar a nuvem de pontos, foi utilizado um *Vertex Buffer Object*, para armazenar os dados a serem renderizados, dentro de um *Vertex Array*, configurado com um *Vertex Attribute Array*, de forma a possibilitar que cada atributo no buffer possa ser acessado individualmente dentro dos shaders. São criados ao todo, 7 *attribute arrays,* que possuem as propriedades de posição, cor das fácies geológicas, id da camada, id da falha, porosidade, amplitude, mergulho da junta, e normais.

3.2 Seção Geológica

A nuvem de pontos visualizada neste trabalho representa uma seção geológica, obtida através de uma análise numérica. Uma seção geológica é constituída por camadas, de rochas, formadas ao longo do tempo e que acumulam-se uma sobre a outra. Esta seção geológica é visualizada, dentro do ReconMS, como uma malha de triângulos, sobre um plano vertical. Nela, são aplicados métodos para restaurar o deslocamento destas camadas, causado pela ocorrência de falhas geológicas.

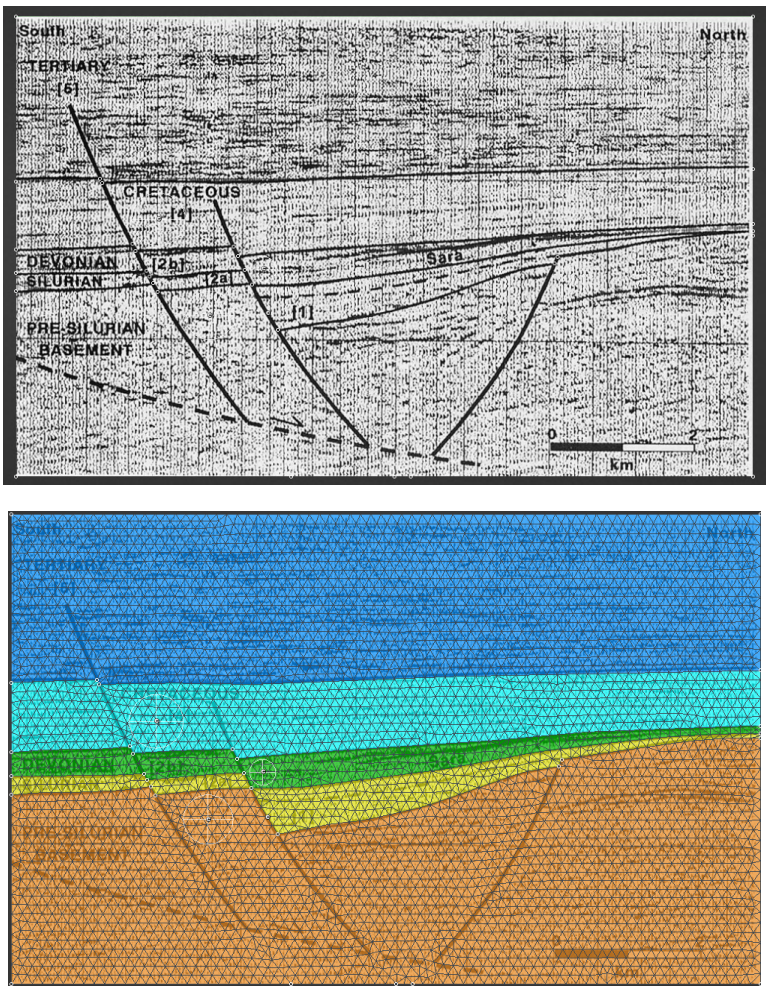

Figura 2. *Imagem gerada por uma análise sísmica de uma seção geológica (em cima), e sua representação dentro do ReconMS (em baixo).*

3.3 Nuvem de Pontos

O principal objeto de estudo deste trabalho é uma nuvem de pontos, com aproximadamente 570 mil pontos. Esta nuvem de pontos é salva em formato *.dat.* Dentro do arquivo, cada linha corresponde a um ponto, detalhado por 19 valores de propriedades específicas. Dentre essas, foram escolhidas 6 para serem mostradas dentro da visualização:

Coordenadas

Coordenadas do ponto no espaço 3D. Elas estão descritas como 3 valores *float*, *x, y, z,* na escala em que foram coletadas, e serão utilizadas para posicionar os pontos.

Camada

Camada da seção geológica a qual o ponto pertence.

Facie

Tipo de rocha que o ponto representa. É representado por um identificador do tipo *int*.

Falha

Id da falha presente, para o caso do ponto representar uma falha, e um id negativo caso contrário, representado por um valor *int.*

Porosidade

Quantidade de porosidade da rocha que este ponto representa.

Amplitude

Amplitude das ondas sísmicas que reverberam na rocha que este ponto representa.

Mergulho da Junta

Ângulo da junta de mergulho.

Para trabalhar com a nuvem de pontos, foi implementada uma classe *Cloud*, que possui os métodos necessários para realizar o *parsing* do arquivo original e popular uma estrutura de dados interna, reposicionar a nuvem, ordenar os pontos e prepará-los para serem desenhados. Ao importar a nuvem, os pontos são reduzidos a escalada selecionada, e o id que corresponde ao tipo de rocha do qual o ponto faz parte é substituído por uma cor, retirada de um dicionário de cores, importado do ReconMS.

Outro aspecto importante da estrutura da nuvem é obter os valores máximos, mínimos, assim como as medianas de cada atributo, para gerar as escalas de cores gradientes que representam cada propriedade. Para criar estas escalas, foi implementado uma função no *fragment shader*, que interpola um valor dentro de um intervalo entre duas cores:

```
vec3 gradient_scale(float value, float min_value, float max_value) {
        float n_value = value / (max_value - min_value);
        vec3 pct = vec3(float(n_value));
        pct.r = smoothstep(0.0, 1.0, n_value);
        pct.g = sin(n_value * PI);
        pct.b = pow(n_value, 0.5);
        return mix(colorA, colorB, pct);
}
```

Essa implementação gera o gradiente abaixo, que será utilizado nas visualizações.

Figura 3. *Gradiente gerado para criar a escala de cores das visualizações.*

3.4 Renderização de Superfície

A primeira técnica de renderização de pontos implementada é a visualização de superfície. Esta técnica tem como objetivo visualizar as superfícies da seção geológica e seus determinados horizontes de forma detalhada e precisa. Esta visualização se assemelha a como o ReconMS visualiza as seções geológicas utilizando malhas de triângulos. Elas são geradas como planos verticais em que são mostrados, na superfície, as diferentes camadas e falhas presentes. Visualizar a nuvem de pontos de forma semelhante às seções geológicas dentro do ReconMS é importante para garantir a corretude da visualização gerada, usando a visualização gerada pelo ReconMS como base.

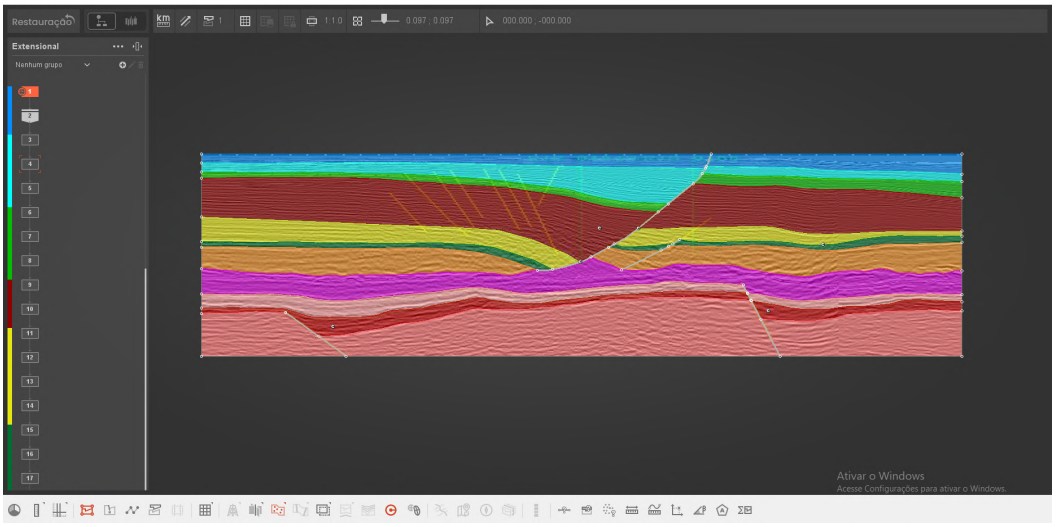

Figura 4. *Visualização de uma seção geológica dentro do ReconMS.*

A técnica utilizada para renderizar as superfícies da nuvem de pontos baseia-se na técnica descrita em *QSplat: A Multiresolution Point Rendering System for Large Meshes* [9] . O artigo descreve formas de renderizar malhas complexas utilizando *primitivas* das API's Gráficas, e propõe algoritmos e estruturas para otimizar a renderização e tratamento destas malhas. O *QSplat* é composto para

A técnica proposta utiliza o *QSplat,* uma representação de ponto no espaço, que são renderizados para compor a superfície. O artigo propõe e testa diferentes formas para representar estes pontos, como quadrados, círculos opacos, círculos com opacidade

variávle ou círculos combinados com elipses. Os *QSplats* possuem atributos de posição, material e normais, de forma que seja possível renderizá-los usando um modelo de iluminação.

A adaptação da técnica para este trabalho consiste em representar cada ponto da nuvem de pontos como uma das primitivas propostas, mediante testes de aceitabilidade. Iniciamos a implementação desenhando 4 pontos, usando a primitiva *GL_POINT*, em 4 cores diferentes, formando as arestas de um quadrado. Como a primitiva *GL_POINT* é desenhada utilizando um número de pixels para o ponto, é necessário que este tamanho varie com a distância da câmera para os pontos, fazendo com que seus tamanhos aumentem conforme a câmera se aproxima.

Para realizar esta variação, o tamanho do ponto é passado para os *vertex shader,* utilizando um *uniform*, e então o tamanjo final é calculado sobre um fator:

O tamanho do ponto e posição da câmera são passados para o *vertex shader*, dentro da *main.cpp*, e então são realizados os cálculos para determinar o tamanho do ponto dentro do *shader*. O vetor de posição da câmera é recebido pelo *layout* definido no *Vertex Array.*

main.cpp

```
shader.setFloat("pointSize", i_pointSize);
shader.setVec3("cameraPos", arcCamera.virtualPosition);
```

vertex.shader

```
layout(location = 0) in vec3 aPos;
uniform vec3 cameraPos;
uniform float pointSize;
...
float distanceToCamera = distance(aPos, cameraPos);
gl_PointSize = pointSize / distanceToCamera * factor;
```

Para visualizar os pontos como uma superfície, os mesmos devem ser desenhados de forma que a sobreposição entre eles preencha os espaços vazios com as cores que aquela região deveria ter. Para que os pontos não variem de tamanho no espaço físico, e ao aproximar a câmera da superfície, eles continuam compondo a superfície, os pontos devem ter um tamanho relativo à distância da câmera, de forma que o seu raio aumenta conforme a câmera se aproxima.

O principal objetivo da técnica utilizada é preencher regiões de pontos com suas respectivas contribuições de cores. Caso seja uma região de pontos de mesma cor, a área final tem a cor destes pontos. No caso de pontos com cores distintas, a cor final da região é a mistura destas cores. Essa composição de cores pode ser utilizando a função *smoothstep* da GLSL para variar a componente *alpha* dos pontos, conforme a distância para seus respectivos centros. Esta função utiliza a *Interpolação de Hermite*, que permite uma transição suave entre 2 valores dentro de um intervalo. Outra forma de atingir essa composição é através da aplicação de uma *Distribuição de Gauss*, também sobre a componente alpha, aplicando uma variação normal, utilizando como parâmetro a distância do a distância do fragmento para o centro do ponto. As duas técnicas empregadas são realizadas dentro do *fragment shader*.
Para utilizar a distribuição gaussiana, foi implementado a função *gauss* em GLSL.

```
float gauss(float x, float x0, float sx) {
    float x_x0 = x - x0;
    x_x0 = -1.0 / 2.0 * x_x0 * x_x0 / sx;
    float a = 1.0 / (pow(2.0 * PI * sx, 0.5));
    return a * exp(x_x0);
}
```

A chamada das funções *smoothstep* e *gauss* dentro do shader são ilustradas abaixo, obtendo os efeitos ilustrados na Figura 5.

```
alpha = gauss(n_dist, 1.0, gauss_factor);
```

ou

```
alpha = smoothstep(0.0, 1.0, n_dist);
```

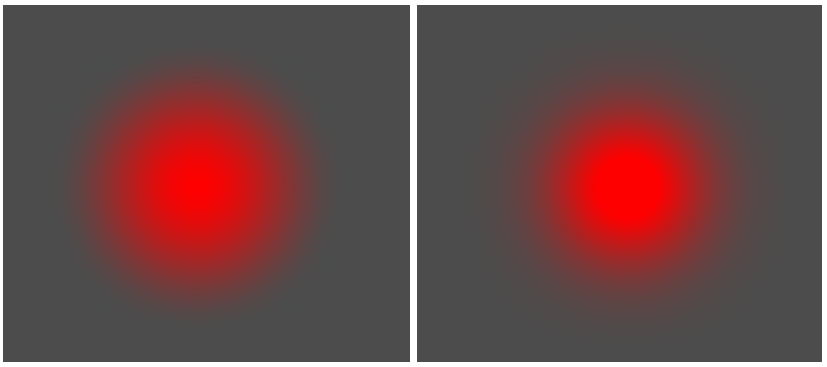

Figura 5. *Variação do canal alpha utilizando smoothstep (à equerda) e distribuição de Gauss (à direita).*

Dessa forma, cada ponto acaba sendo desenhado como um círculo opaco no centro e proporcionalmente mais transparente conforme se distancia do centro. Para conseguir a composição desejada, é utilizada a função de *blending* do OpenGL *glBlendFunc,* com os parâmetros *GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA.* (Figura 6).

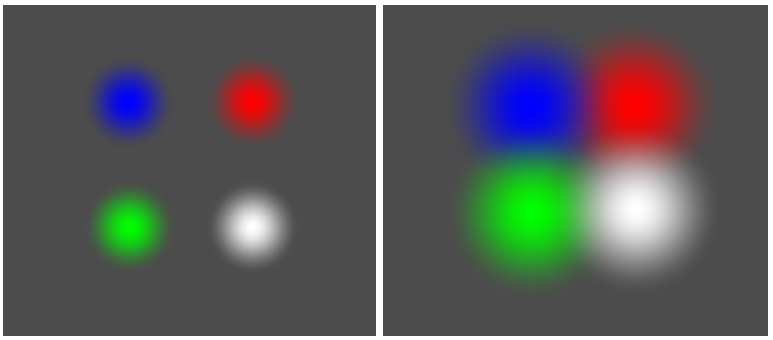

Figura 6. *Pontos desenhados como círculos com transparência relativa ao raio (usando smoothstep).*

Essa abordagem permite que pontos que estejam sobrepostos contribuam de forma equivalente para a cor final da região, e possibilita uma transição suave entre áreas de cores diferentes. Aplicando esta visualização para a nuvem de pontos, utilizando as cores das camadas obtemos os resultados ilustrados na figura 7.

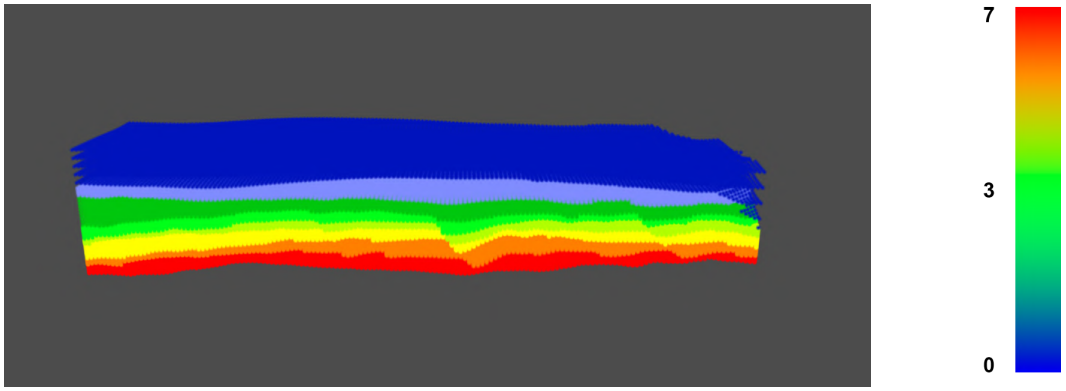

Figura 7. *Visualização dos pontos utilizando cor do ponto e sem normais.*

Além das propriedades presentes na nuvem, é necessário que os pontos que compõem a superfície possuam suas respectivas normais para que seja possível aplicar iluminação dentro da visualização. Utilizar técnicas de iluminação é importante para possibilitar a visualização das imperfeições e irregularidades da superfície. Como estes valores de normais não estão presentes no arquivo, faz-se necessário a geração destas normais. Para tal, foi utilizado um software externo chamado *Cloud Compare* [1], que possui as estruturas internas apropriadas para importar uma nuvem de pontos e calcular suas normais. O software permite a geração das normais de superfície, utilizando uma estrutura de *Octree*. Então, após a obtenção das normais para cada ponto, é possível aplicar um

modelo básico de iluminação. O modelo utilizado neste trabalho é o *Modelo de Reflexão de Phong,* e permite visualizar as imperfeições e irregularidades sobre as superfícies.

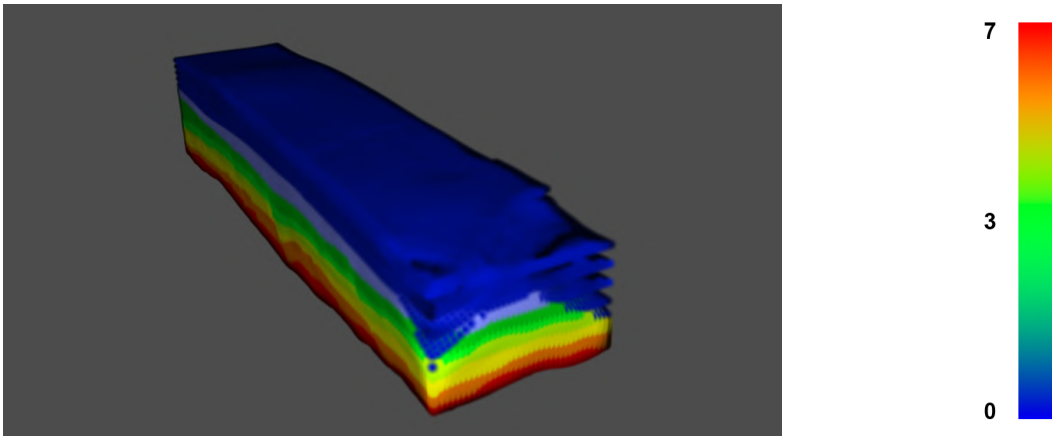

Figura 8. *Visualização da nuvem de pontos com as normais calculadas.*

Tendo em vista a funcionalidade do ReconMS de descompactar as camadas da seção geológica, as normais devem ser geradas corretamente não apenas para a superfície da seção, mas para cada superfície individual de cada camada. Utilizando os identificadores de cada camada, é possível apenas renderizar os pontos que fazem parte das camadas selecionadas, permitindo a funcionalidade de “esconder” uma camada, durante a visualização. Como o método numérico utilizado pelo *Cloud Compare* é apenas capaz de calcular as normais dos pontos da superfície da nuvem com precisão, os pontos não superficiais possuem normais imprecisas, que causam fragmentos e erros na visualização.

Para calcular corretamente as normais das superfícies internas, faz-se necessário realizar esse cálculo individualmente para cada camada. Como os pontos já possuem o *id* da camada da qual pertencem, eles são facilmente exportados utilizando um filtro para o *id* especificado. Então, após as normais serem calculadas individualmente por camada, é possível visualizar as superfícies independentes de forma correta.

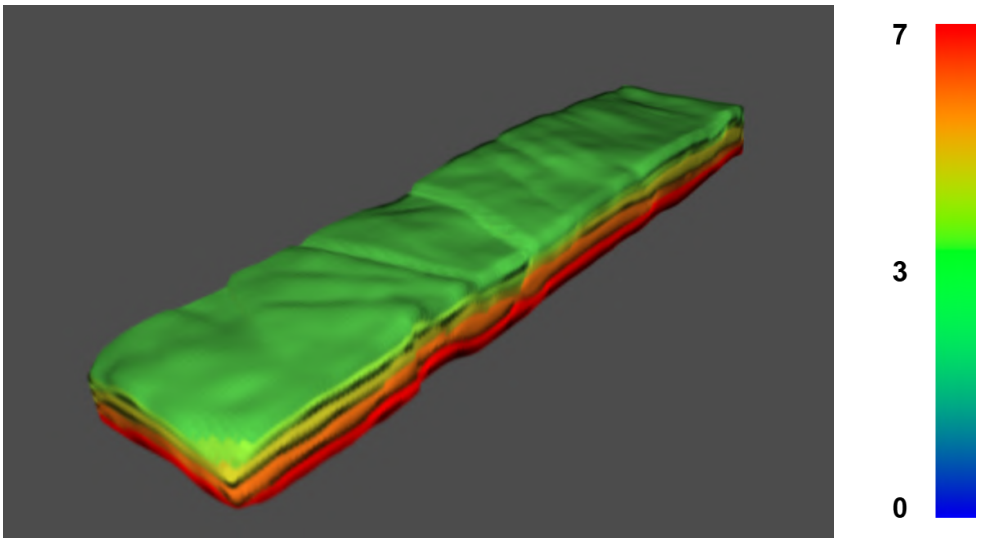

Figura 9. *Superfície das camadas internas, com as normais calculadas individualmente.*

O ponto desfavorável desta implementação é a imprecisão gerada nas normais presentes nas regiões de interseção entre duas camadas com as superfícies laterais da nuvem. Isso ocorre porque elas deixam de ser calculadas como parte de uma superfície horizontal, e são calculadas dentro das regiões de arestas destas camadas. Mesmo após a segmentação camada, e os cálculos de normais individuais, ainda existem fragmentos de normais invertidas, que aparecem como áreas escuras na visualização. A correção desses fragmentos consiste em inverter as normais cujo resultado do produto escalar com a direção da câmera são negativas. Isso faz com que as regiões invertidas fiquem visíveis, possibilitando a visualização da superfície por qualquer ângulo.

3.5 Ordenação dos Pontos

Apesar do resultado parcialmente correto, movimentar a câmera para o lado contrário da nuvem de pontos faz com que a superfície seja renderizada de forma incorreta, mostrando apenas os pontos da superfície oposta. Isso é causado pela ordem na qual os pontos dentro do *Buffer* são desenhados, na ordem em que foram inseridos. Caso a câmera se mova, os pontos devem ser reordenados, baseados na distância entre suas posições e a posição atual dela, de forma que são desenhados à frente dos pontos mais distantes.

O OpenGL realiza essa operação durante o teste de profundidade [6], descartando fragmentos que estão sendo sobrepostos por outros fragmentos com distância maior para a câmera. Entretanto, esse teste não pode ser utilizado dentro dessa implementação, já que a composição da superfície depende da sobreposição de fragmentos transparentes, que seriam descartados pelo teste de profundidade.

Para realizar esta ordenação, foi implementado uma função na classe *Cloud*, que usa o *standard sort* do C++, realizando a ordenação em relação à distância de cada ponto para a câmera. A ordenação é realizada utilizando o vetor de pontos que foi carregado do arquivo, em que cada elemento possui as propriedades do ponto. Essa implementação utiliza uma função de comparação própria, descrita abaixo:

```
float distanceP1 = glm::distance(p1.m_position, cameraPos);
float distanceP2 = glm::distance(p2.m_position, cameraPos);
return distanceP1 > distanceP2;
```

Após a ordenação, o *buffer* é substituído pelo vetor ordenado, e desenhado novamente (Figura 10). A ordenação em CPU é suficiente para realizar a visualização da superfície. A partir de qualquer localização da câmera, após realizar a ordenação dos pontos utilizando a interface gráfica, os pontos são desenhados na ordem correta, permitindo a composição correta da transparência.

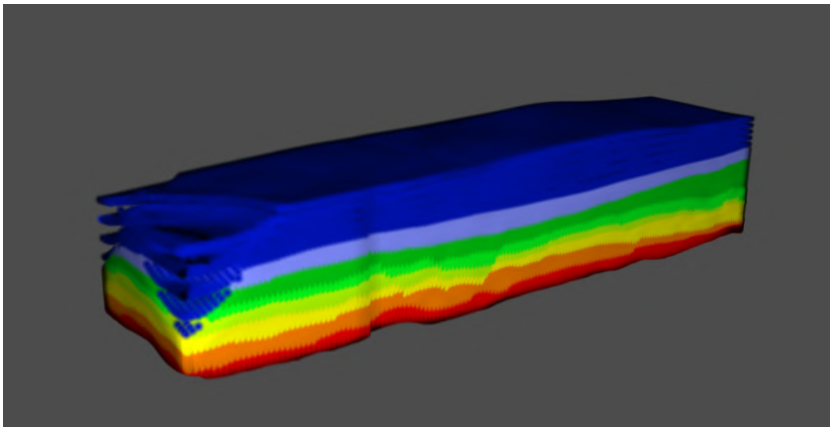

Figura 10. *Pontos sendo renderizados na ordem certa, após ordenamento, permitindo a visualização de qualquer "lado" da nuvem de pontos.*

3.6 Renderização de Volume

A visualização volumétrica tem como objetivo possibilitar a exibição de propriedades cuja distribuição se concentre no interior da nuvem de pontos. Ao visualizar apenas as superfícies, estes pontos não são visíveis, sendo sobrepostos por diversos outros pontos. Para possibilitar a visualização destes pontos e as variâncias de seus atributos, é necessário aplicar uma técnica de renderização volumétrica.

A técnica [4] e [10] utilizada como base para este trabalho foi proposta para possibilitar a visualização de volumes, utilizando texturas multidimensionais. Essa técnica abstrai a propriedade escolhida para ser visualizada volumetricamente como um valor de densidade. Dessa forma, é possível tornar visível apenas a densidade desejada dentro no volume. O escopo da técnica é proposto para renderizar volumes a partir de malhas de triângulos, e utilizando cortes paralelos para determinar a densidade de cada fragmento. A sua aplicação em uma nuvem de pontos consiste em variar a componente *alpha* utilizando uma função de transferência [4] . A função de transferência converte uma densidade daquele ponto em um valor alpha, de forma que a sobreposição de pontos em áreas com maior densidade gere um volume mais opaco. Os pontos são desenhados utilizando as mesmas primitivas da visualização de superfície, utilizando discos com opacidade gaussiana, e ativando o *blending* do OpenGL.

A função de transferência implementada normaliza o valor da densidade dentro do intervalo em que ela pertence. Então, esse valor normalizado é interpolado entre 0.0 e 1.0 para gerar um *alpha* correspondente. Caso o valor esteja abaixo da densidade selecionada para a visualização, ele é descartado (*alpha = 0.0)*.

```
float TransferFunction() {
    float n_value = value / (max_value - min_value);
    float maxView = threshold + range;
    float minView = threshold - range;
    if (insideRange) return smoothstep(0.0, 1.0, n_value);
    return 0.0;
}
```

Aplicando esta função de transferência, obtemos os seguintes resultados, visualizando o volume da propriedade de porosidade.

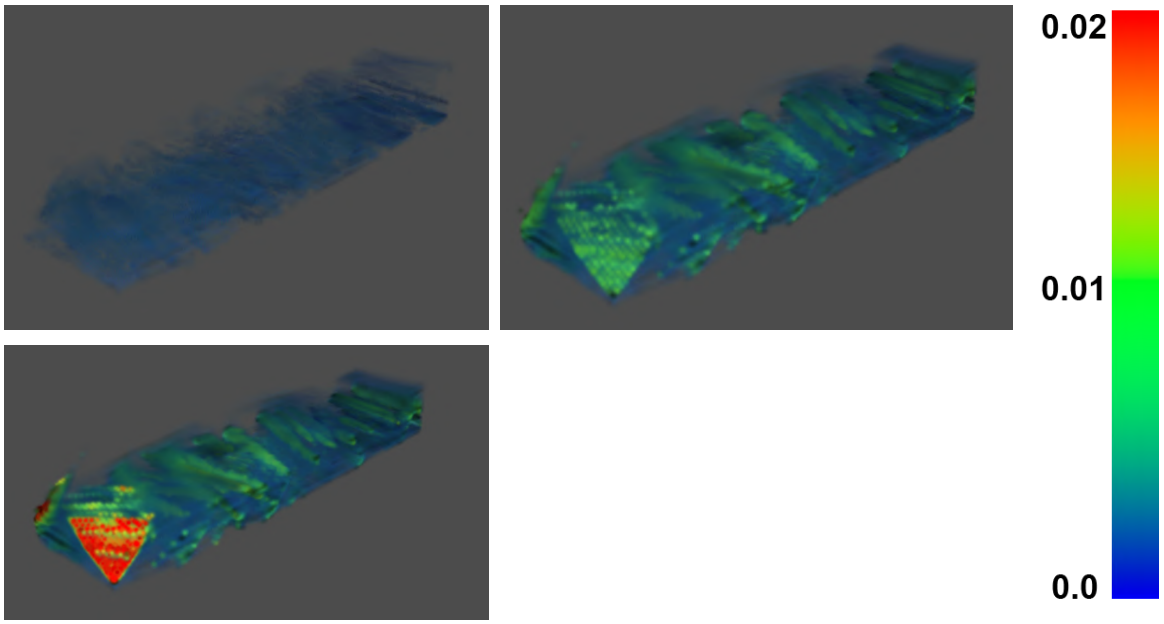

Figura 11. *Visualização de volume com as cores das camadas (em cima), e com as cores seguindo uma escala.*

4. Resultados

4.1 Visualização de Superfície

Testes e resultados obtidos com as técnicas de visualização de superfície implementadas:

O primeiro teste é realizado utilizando a propriedade de camada para exibir os pontos. A cama varia entre 0 e 7, e possui uma cor determinada pela escala à direita (Figura 12). Então, são destacadas (em branco) as falhas da seção geológica (Figura 12). É possível identificar que as falhas são corretamente localizadas. Além disso, é possível visualizar as superfícies individuais de cada camada, obtendo os resultados na Figura 13. Neste teste, é possível visualizar as rugosidades e irregularidades da superfície da camada laranja, assim como a região da falha .

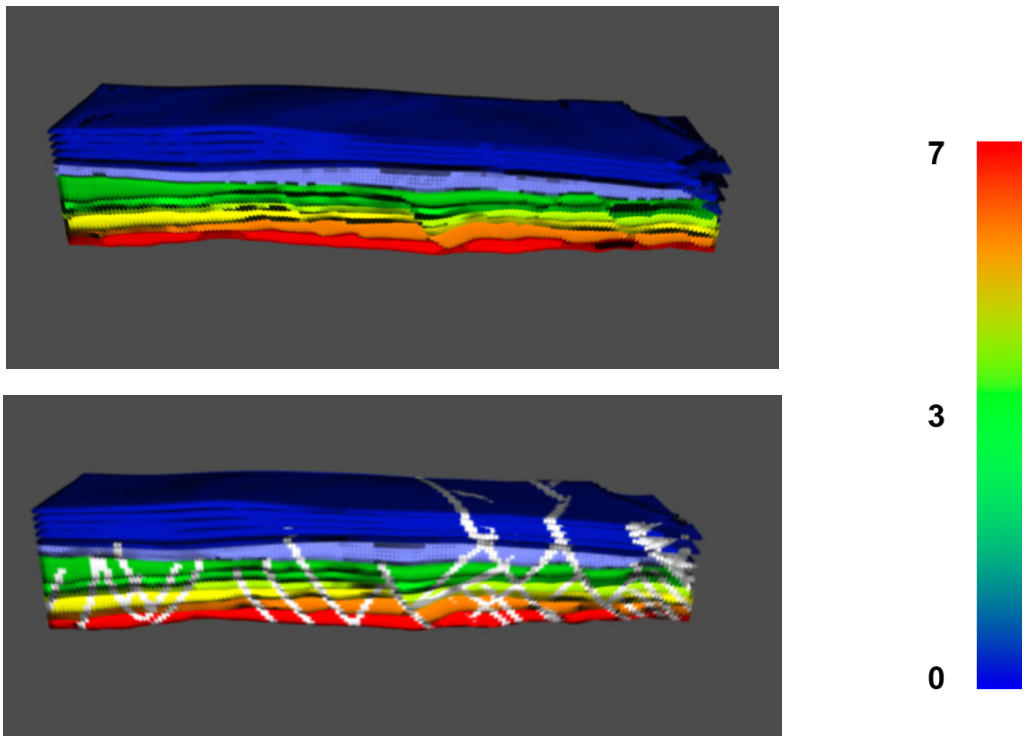

Figura 12. *Visualização da superfície da nuvem de pontos (em cima) e com destaque das falhas geológicas da seção em branco (em baixo).*

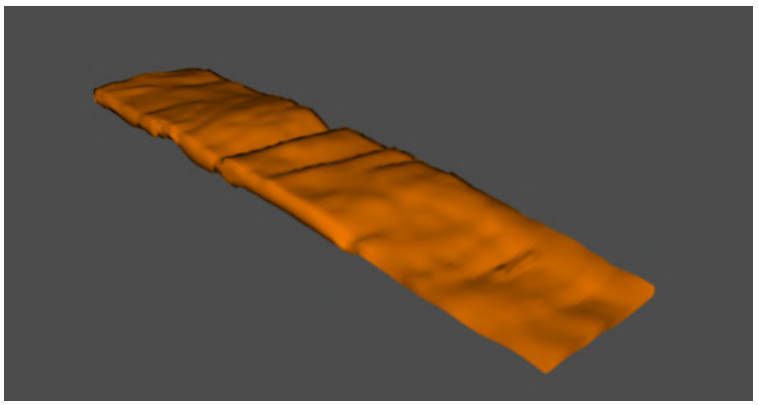

Figura 13. *Visualização de uma camada individual da seção geológica.*

No teste seguinte, foram visualizadas as cores que representam a propriedade da rocha. Este teste se assemelha ao primeiro, pois cada ponto já possui uma cor específica, e os resultados são similares. É possível também identificar que as regiões com folhas também aparecem nas superfícies laterais, bem como a sobreposição de finas camadas com diferentes tipos de rocha.

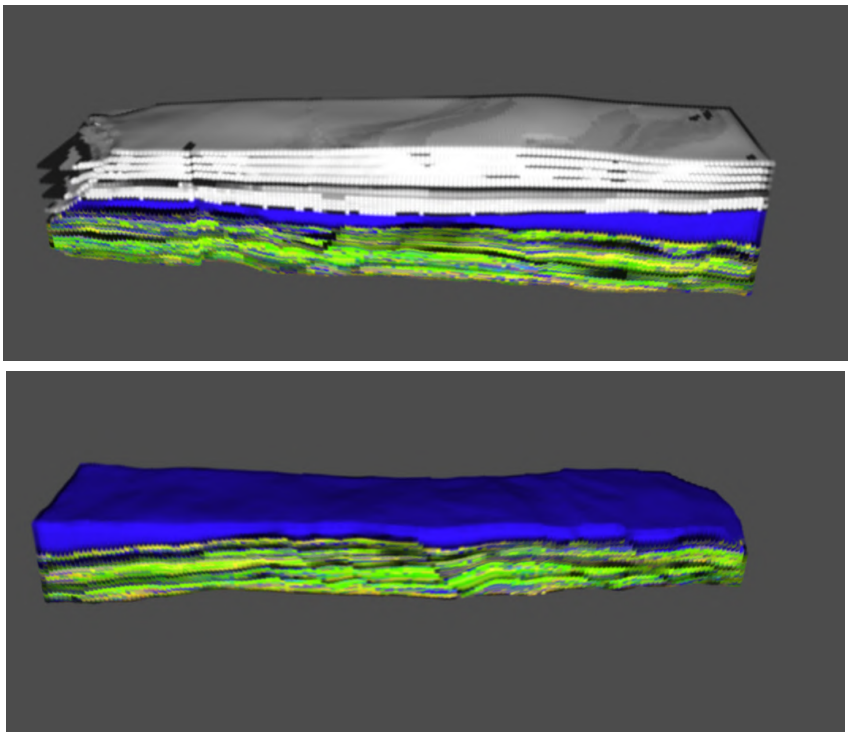

Figura 14. *Seção geológica visualizada com as cores de fácies de rocha, mostrando a nuvem de pontos original (em cima) e não mostrando as fácies indefinidas, cujo id = -999 (em baixo).*

O teste seguinte foi feito visualizando as propriedades de porosidade da seção geológica. Nesse caso, o intervalo visualizado foi de 0.0 até 0.02, e é possível identificar (Figura 15) o acúmulo de pontos com maiores porosidades justamente nas regiões em que há falhas.

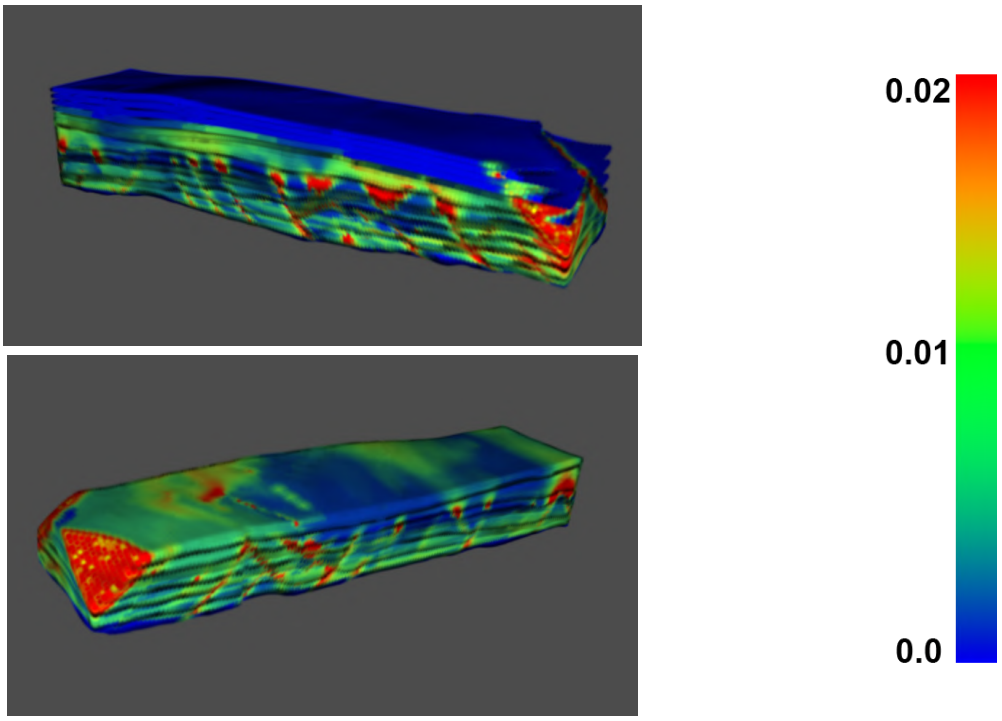

Figura 15. *Superfície da seção geológica visualizada pela propriedade de porosidade.*

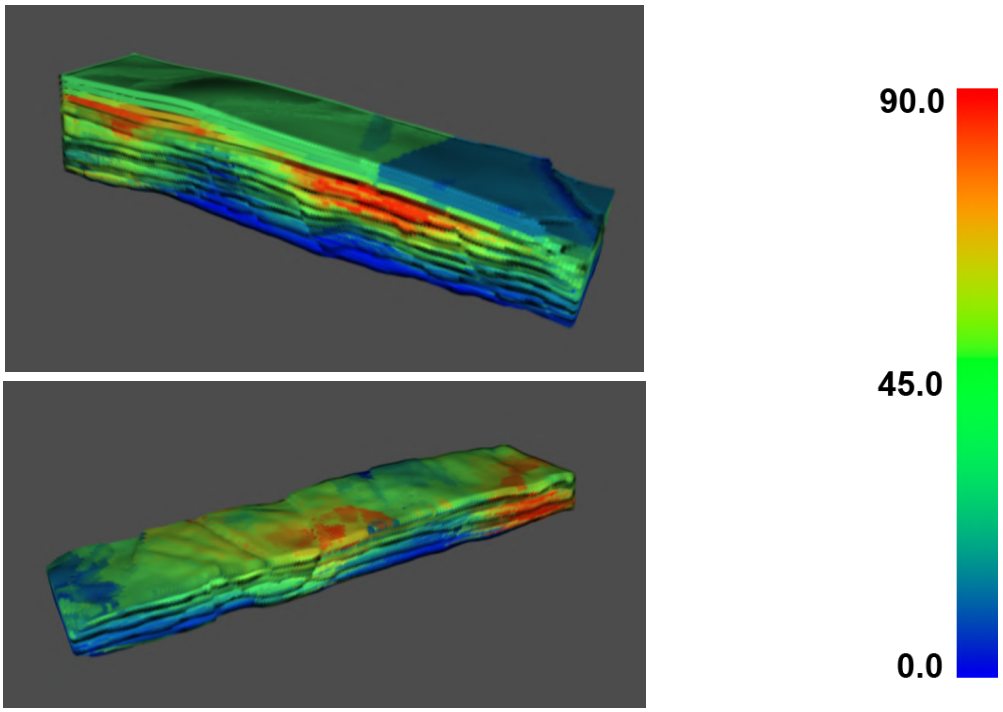

Figura 16. *Superfície da seção geológica visualizada pela propriedade de mergulho de falha.*

Com a aplicação dessas técnicas, os resultados obtidos, utilizando a resolução *1600x900*, levam entre 68 milissegundos e 75 milissegundos para serem gerados, obtendo uma média de 18 quadros por segundo. Separadamente, a ordenação dos pontos leva em média 35 milissegundos, enquanto a renderização mantém a média de 25 milissegundos, independente das condições. Esses tempos são considerados aceitáveis, por estarem bastante acima dos mínimos 6 frames por segundo, considerando tempo interativo.

4.2 Visualização de Volume

Resultados obtidos a partir da visualização de volume da nuvem de pontos, visualizando a propriedade de porosidade, que tem principal distribuição entre 0.0 e 0.020, com maior densidade de pontos no intervalo 0.0 e 0.012. Esta visualização permite que sejam visualizadas apenas as densidades selecionadas. A partir do volume, é possível identificar a horizontalidade das falhas, bem como confirmar o acúmulo de porosidade menor no interior da seção.

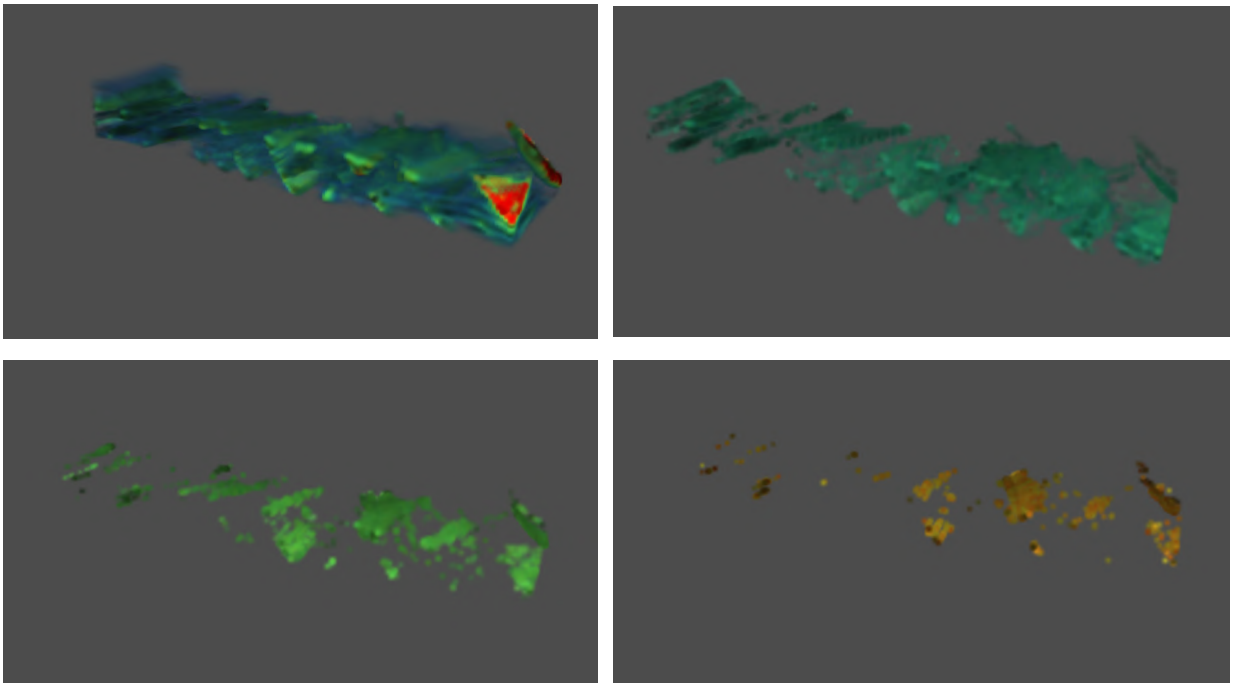

Figura 17. *Visualização volumétrica da nuvem de pontos, comparadas à visualização de superfície (em cima), com o intervalo da porosidade visível aumentando.*

No teste seguinte, é visualizado o volume da nuvem de pontos, utilizando a propriedade de mergulho da junta como densidade. Neste teste, é possível identificar o mergulho da junta se concentrando em algumas regiões (Figura 18) , e principalmente visualizar os volumes separados por densidades. Na primeira figura (superior esquerda) é

possível observar a usabilidade da visualização de volume, ao ver os os pontos com mergulho de junta menor quase transparentes, expondo o volume abaixo com maior densidade.

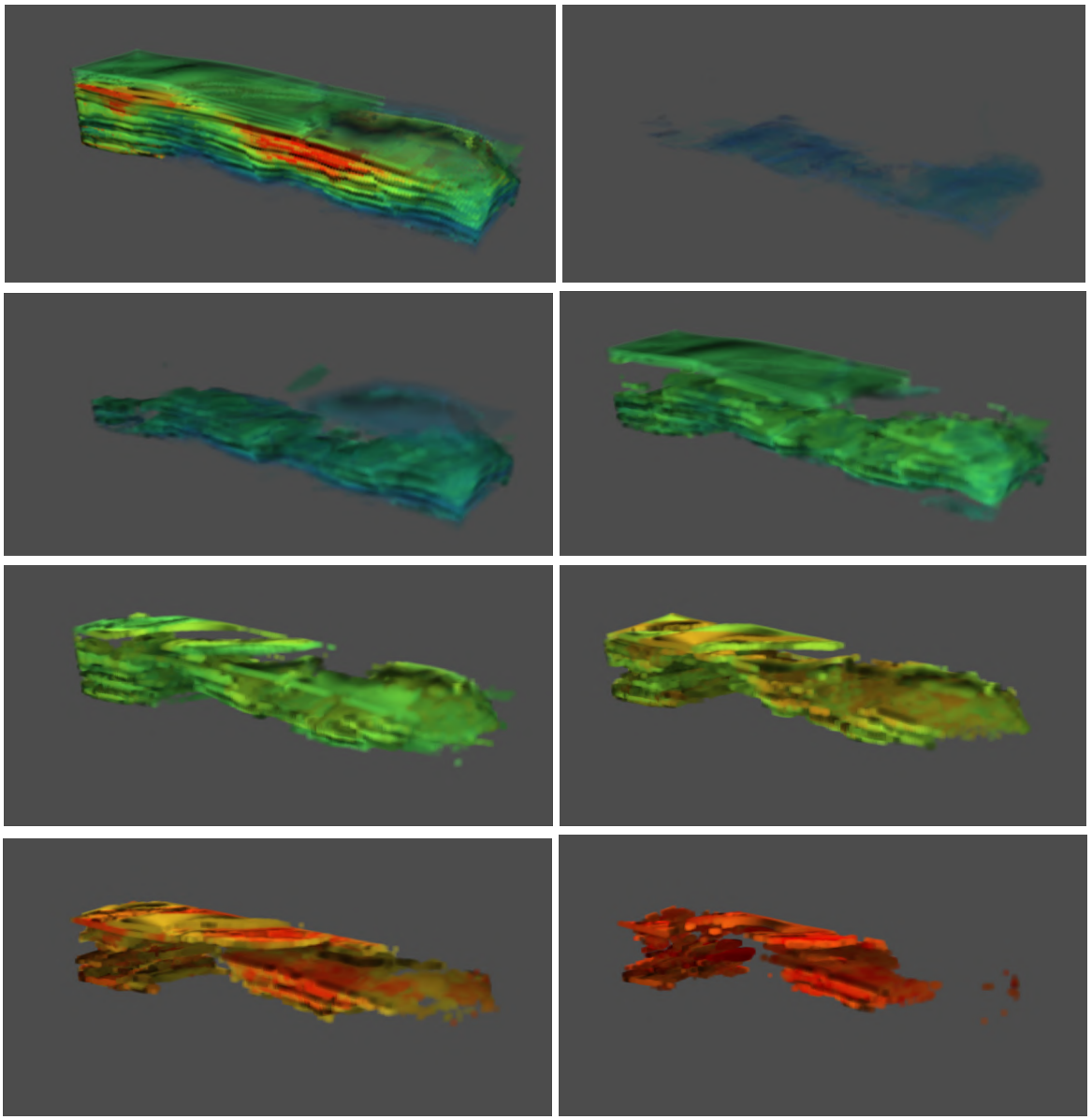

Figura 18. *Visualização volumétrica da nuvem de pontos, variando o mergulho da junta visível.*

O tempo médio de renderização dos pontos na visualização de superfície foi de 25 milissegundos, que somados aos 35 milissegundos da ordenação dos pontos, atinge 60 milissegundos, o que se traduz em aproximadamente 15 quadros por segundo.

Conclusão

Aplicar diferentes técnicas de renderização em uma nuvem de pontos permite uma melhor visualização das suas propriedades, forma e especificidades. No contexto específico da motivação deste trabalho, a aplicação das técnicas propostas para visualização da nuvem de pontos dentro ReconMS resultará em uma visualização mais detalhada. Este trabalho propôs duas técnicas de visualização distintas, baseados em trabalhos já fundamentados na linha de pesquisa da computação gráfica. Ambas as técnicas foram resultado da adaptação para o escopo deste trabalho, e o enfoque visando sua contribuição no ReconMS.

As duas visualizações, volumétrica e de superfície, apresentaram desempenhos próximos uma da outra, dentro da taxa de quadros por segundo (pelo menos 6 quadros por segundo) necessárias para ser considerada interativa. Há ainda, espaço para otimização das implementações. De acordo com os resultados obtidos, cerca de 50% do tempo necessário para realizar a renderização é ocupado com a ordenação dos pontos. Um caminho possível para reduzir esse tempo é a implementação da ordenação utilizando CUDA [2], de forma a usar um algoritmo de ordenação paralelo, e retirar os tempos necessários para substituir os *buffers* com o novo vetor de pontos ordenado.

Por fim, os resultados obtidos por este trabalho foram apresentados à equipe de geologia do ReconMS, e posteriormente validados, de forma que ambas as técnicas propostas enriquecem a visualização da nuvem de pontos dentro do software, e não apenas possibilitam sua melhor análise como também abrem precedente para a expansão da usabilidade destes dados.

Bibliografia

[1] *Cloud Compare, disponível* https://www.cloudcompare.org/main.html*,* acesso em Junho de 2022.

[2] *CUDA,* disponível em https://developer.nvidia.com/cuda-toolkit, acesso em Junho de 2022.

[3] DE VRIES*,* Joey, *Learn OpenGL,* disponível em https://learnopengl.com/, acesso em Junho de 2022.

[4] KNISS, Joe; KINDLMANN, Gordon; and HANSEN, Charles; *Multidimensional Transfer Functions for Interactive Volume Rendering,* IEEE, 2002.

[5] *OpenGL Reference Pages:* https://www.khronos.org/registry/OpenGL-Refpages/gl4/, acesso em Junho de 2022.

[6] *OpenGL Depth Test,* disponível em https://www.khronos.org/opengl/wiki/Depth_Test, acesso em Junho de 2022.

[7] PHONG, Bui Tong. *Illumination for Computer Generated Pictures*, 1975.

[8] *ReconMS*: https://www.tecgraf.puc-rio.br/publicproductdetail/pnM4xPEhSNnLN3Zfb , acesso em Setembro de 2021.

[9] RUSINKIEWICZ, Szymon; LEVOY, Marc; *Qsplat: a multiresolution point rendering system for large meshes.* Stanford University , 2000

[10] WITTENBRINK, Craig; MALZBENDER, Tom; *Opacity-weighted color interpolation for volume sampling*, IEEE Symposium on Volume Visualization, 1998.