Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Guilherme Ferreira Gusmão**

**Matrícula 2012405**

**Simulador de LiDAR para Geração de Nuvens de Pontos Sintéticas**

**Documentação para a disciplina Projeto Final de**

**Programação**

Orientador: Prof. Alberto Barbosa Raposo

Rio de Janeiro

Dezembro de 2021

Sumário

[**Introdução:** 2](#_Toc89131393)

[**Especificação do Sistema:** 3](#_Toc89131394)

[*Objetivo:* 3](#_Toc89131395)

[*Requisitos:* 3](#_Toc89131396)

[*Casos de Uso:* 4](#_Toc89131397)

[*Especificação da Tecnologia:* 5](#_Toc89131398)

[**Projeto do Programa:** 6](#_Toc89131399)

[*Arquitetura:* 7](#_Toc89131400)

[*Diagrama de Organização dos Dados:* 10](#_Toc89131401)

[*Organização do Programa:* 11](#_Toc89131402)

[**Teste** **Efetuado**: 12](#_Toc89131403)

[*Roteiro*: 12](#_Toc89131404)

[*Possibilidades de Melhorias no Projeto*: 13](#_Toc89131405)

# **Introdução:**

A tecnologia de imagem 3D é capaz, por meio do uso de sensores e câmeras especializadas, coletar dados de um objeto ou região alvo e processá-los no computador para gerar dados métricos precisos, com resolução capaz de atingir a ordem de milimetros. Uma forma de exibir esses conjuntos de dados é em conjuntos de pontos no mesmo sistema de coordenadas com informações geoespaciais e/ou métricas, chamados de nuvens de pontos.

Dentre os sistemas de imagens 3D, o dispositivo Light Detection and Ranging (LiDAR), um sensor óptico ativo de imageamento baseado em luz, vem recebendo cada vez mais atenção, tanto do setor industrial quanto do meio acadêmico. A precisão e velocidade de aquisição de dados, juntamente com a capacidade de incorporá-los em veículos não tripulados ou operados remotamente, permitiram uma gama de aplicações interessantes, como carros autônomos realizando mapeamento em tempo real de seus arredores usando sensores LiDAR com redes de machine learning ajudando com a interpretação do ambiente.

Embora muitas aplicações que usam sensores de imagem 3D, como LiDAR, sejam bem-sucedidas, existem alguns problemas associados à tecnologia:

• A coleta de dados brutos depende diretamente de quantos sensores são usados ​​e do tamanho do alvo verificado. Portanto, quanto maior for o alvo, mais lenta e mais cara será a operação.

• Existem situações impossíveis de serem testadas devido a restrições de segurança e preservação do bem-estar humano, como na aquisição de dados para o treinamento de veículos autônomos em casos de acidente automobilístico e atropelamento de pedestre ou animal.

• A varredura feita em ambientes hostis tem um risco maior de danificar o equipamento, como no caso de inspeções submarinas.

• O hardware e o equipamento de tecnologia ainda são caros no presente momento.

• Devido à enorme quantidade de pontos normalmente presentes em nuvens de pontos, a segmentação semântica (anotação de dados com rótulos de classificação para cada ponto ou pixel) torna-se onerosa e demorada.

Esses aspectos dificultam aplicações que precisam de grandes conjuntos de datasets, como a criação de dados para treinamento em deep learning. Muitos são os esforços para contornar ou resolver essa barreira. Uma solução em voga é o desenvolvimento de simuladores de sensores, também conhecidos como escaneamento virtual. A técnica consiste na interação de um simulador de sensores de imageamento com um cenário virtual 3D, gerando dados sintéticos com anotação automática ou semiautomática.

Uma vez que, em uma simulação, o ambiente e as situações podem ser controlados, é possível reproduzir vários cenários e experimentos, mesmo aqueles com riscos potenciais para equipamentos ou pessoas. Consequentemente, esses procedimentos geram mais conjuntos de dados de nuvens de pontos em um tempo menor, dependendo apenas do poder computacional disponível. Esses simuladores também podem ser úteis na educação, permitindo a geração de dados em diversas situações laboratoriais, permitindo que os alunos entendam melhor as tecnologias de imagem antes de usar dispositivos reais.

# **Especificação do Sistema:**

O simulador de LiDAR apresentado aqui, apelidado de optScan, vem de um histórico de pesquisa submarina, principalmente na área de petróleo e gás. Como as profundezas do oceano são um ambiente muito hostil, somado ao equipamento caro, é impossível adquirir uma quantidade adequada de dados para aplicações como a criação de conjuntos de dados para treinar sistemas de deep learning. Assim, a geração de nuvens de pontos sintéticas ajudaria a criar mais dados para prova de conceito, melhor planejamento de aquisição de dados em campo e aumento de dados na área submarina.

O simulador utiliza um modelo híbrido de sensor (baseado em medições e em erros) para gerar conjuntos de dados sintéticos com anotação semiautomática com fácil adaptação para diferentes campos de pesquisa. Seu sensor virtual calibrado por comparação com um sensor real garante maior robustez neste processo.

O sistema foi construído de forma modular, constituído de 3 principais blocos: Um construtor de cenários virtuais baseado em objetos 3D; um modelo matemático do comportamento de um LiDAR real a partir daqui chamado de sensor virtual; e por fim um módulo capaz de processar os dados na forma de nuvens de pontos.

## *Objetivo:*

Este trabalho tem como objetivo o desenvolvimento de um simulador de sistema LiDAR capaz de receber informações sobre objetos 3D e posições do sensor virtual dentro de um cenário virtual. Ao final do processamento desses dados pelo sistema, deveremos ter como saída uma nuvem de pontos sintética com informações registradas de forma semiautomática.

## *Requisitos:*

A especificação de requisitos tem como objetivo obter produtos de software de melhor qualidade que satisfaçam às reais necessidades dos clientes dentro de prazo e orçamento adequados. A partir dela todas as demais atividades serão realizadas: projeto, codificação e testes. A seguir são listados os requisitos do sistema (tabela 1).

|  |  |
| --- | --- |
| Requisito Funcionais do Sistema | |
| RF-1 | O sistema precisa de três módulos, sendo um construtor de cenários virtuais, um sensor virtual e um gerador de nuvens de pontos. |
| RF-2 | O sistema deve ser feito de tal forma, que a substituição dos módulos por outras ferramentas similares mantenha seu pipeline atual. |
| RF-3 | A saídas de cada módulo devem respeitar o padrão estipulado neste documento |
| RF-4 | O cenário virtual necessita de uma interface que facilite a construção de cenas simples a complexas pelo usuário |
| RF-5 | O cenário virtual deve permitir o usuário definir a classe de cada objeto que ele define em cena |
| RF-6 | O sensor virtual deve ser capaz de ser modificado para receber parâmetros metrológicos do LiDAR que se deseja simular |
| RF-7 | O sensor virtual deve gerar os dados como um sensor real, produzindo uma saída a partir da distância entre a posição do sensor virtual até a superfície do objeto alvo |
| RF-8 | O gerador de nuvens de pontos sintéticos deve ser capaz de organizar os pontos gerados pelo sensor virtual em nuvens de pontos, tal como num sistema LiDAR real |
| RF-9 | O gerador deve ser capaz de juntar todas as nuvens de pontos geradas em uma varredura como um único modelo 3D |

Tabela - Requisitos do Sistema

## *Casos de Uso:*

A partir dos requisitos funcionais identificados, foi elaborado um Diagrama de Casos de Uso para ilustrar as tarefas que o sistema deve cumprir para o usuário. Este diagrama é exibido na Figura 1, abaixo:

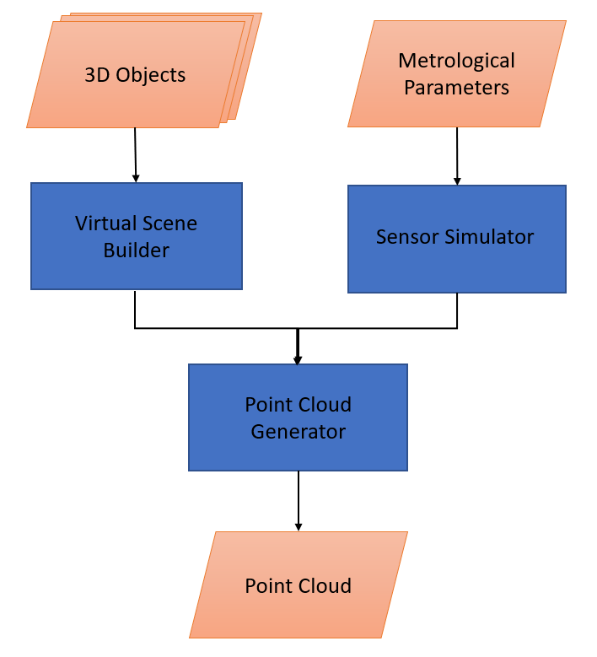


Figura 1 – Fluxograma de Casos de Uso: Paralelogramos são dados de entrada e/ou saída, retângulos são módulos.

**Virtual Scene (Cenário Virtual):** É necessário ter construtor de cenas em que o usuário possa construir o modelo alvo e escolher os locais de escaneamento do sensor. Depois disso, o módulo deve produzir arquivos com características de cena e um caminho de varredura de sensor que pode ser usado em outros módulos para gerar os dados brutos e nuvens de pontos sintéticos.

**Sensor Simulator (Sensor Virtual)**: O dispositivo principal em um sistema de imageamento 3D é o seu sensor de imagem, pois ele irá ditar quais métodos de aquisição de dados podem ser usados. A contraparte digital do sensor de imagem é o simulador de sensor: um modelo matemático baseado no comportamento e nas técnicas de aquisição de dados de um sensor designado. No caso de métodos de imagem 3D para gerar nuvens de pontos, isso se traduz em como emular a aquisição de dados brutos por um sensor real, seja a partir de métodos de imagem ótica 3D passivos ou ativos, na forma de imagens ou distâncias. O fator crucial para uma geração de nuvem de pontos robusta é a qualidade do comportamento modelado no simulador de sensor, uma vez que os dados brutos reais também são processados ​​por algoritmos computacionais para criar as nuvens de pontos do alvo.

Esses simuladores de LiDAR podem ser facilmente divididos em três categorias:

• “Modelo de medição do sensor”: Modelos baseados em uma técnica de medição real do sensor e a orientação do sensor na cena.

• “Modelo de erro do sensor”: Modelos baseados nos erros estatísticos percebidos nas medições de um sensor real, considerando erros sistemáticos e aleatórios.

• “Modelo híbrido do sensor”: Modelos que combinam os dois métodos anteriores.

**Point Cloud Generator (Gerador de Nuvens de Pontos):** Neste módulo, o objetivo é pegar as saídas obtidas pelo sensor virtual após escanear o cenário virtual e processar estes dados de forma a criar um conjunto de pontos ordenados espacialmente, gerando assim as nuvens de ponto. Esta etapa é comum tanto ao escaneamento com equipamentos de imageamento real quanto aos simuladores, pois os dados brutos são ambos processados por algoritmos computacionais.

## *Especificação da Tecnologia:*

Como visto na seção anterior, o optScan é um sistema composto por três grandes módulos. Sendo assim, o pipeline é feito com uso de mais de um software que funcionam em conjunto para alcançar o objetivo final – uma nuvem de pontos sintética. A seguir, as especificações da tecnologia de cada módulo serão aprofundadas.

a. Construtor de cenário virtual:

Este módulo foi desenvolvido com a engine de jogos Unity 3D na versão 2018.3.14f1. As ferramentas presentes nela permitem ao usuário construir facilmente uma cena virtual com objetos 3D.

b. Modelo do Sensor Virtual:

Para sensor virtual foi criado um modelo híbrido, ou seja, uniu-se um modelo baseado em medições com um modelo baseado em erros para aumentar o realismo durante a aquisição de dados brutos.

Para o modelo de medição, foi necessário replicar a técnica de tempo de voo (ToF), que é o princípio de operação do LiDAR. Essas técnicas utilizam o tempo que leva para um sinal sair e retornar ao sistema e calcula a distância percorrida pelo sinal até atingir uma superfície reflexiva, como visto na Figura 2.

Diagrama

Descrição gerada automaticamente

Figura 2 - Princípio de funcionamento de Tempo de Voo em um LiDAR.

A distância pode ser encontrada com a seguinte equação na maioria das aplicações típicas:

|  |  |
| --- | --- |
|  | (1) |

Onde d é a distância percorrida, t é o tempo de voo medido e c é a velocidade da luz.

Uma boa maneira de desenvolver o modelo de comportamento de ToF foi utilizando algoritmos de raytracing, uma família de computação gráfica de algoritmos para renderização de imagens. Com esta motivação, foi escolhido a engine de raytracing OptiX 5.0 da Nvidia, pois a API permite o uso de um enorme conjunto de ferramentas e algoritmos de rastreamento de raios com alto grau de paralelismo, ou seja, o processamento de cada raio disparado pode ser realizado de forma independente, reduzindo o tempo total de cálculos computacionais. A linguagem de programação é CUDA com wrapper C++.

c. Gerador de Nuvens de Pontos Sintéticos:

Este módulo tem dois blocos internos feito em diferentes arquiteturas:

- O bloco simulation: responsável pelo processamento dos dados brutos obtidos pelo sensor virtual durante o escaneamento do cenário virtual. Ele foi projetado com o Optix 5.0.

- O bloco merge: As nuvens de pontos sintéticas individuais geradas no bloco anterior precisam ser unificadas em um único modelo 3D. Ele foi projetado com a biblioteca de algoritmos geométricos Geogram.

Abaixo, a tabela 2 indica os requerimentos mínimos para o sistema.

|  |  |
| --- | --- |
| Hardware | Requisitos Mínimos |
| Placa Gráfica | GPUs da NVIDIA com Compute Capability 3.0 (Kepler) ou superior |
| CPU | Arquitetura X64 com suporte ao conjunto de instruções SSE2 |
| Memória | 8 GB de RAM |
| Sistema Operacional | Windows 7/8.1/10 64-bit;  Mac OS 10.9 ou superior |

Tabela 2 - Requisitos mínimos

# **Projeto do Programa:**

A partir dos requisitos do sistema e de seus casos de uso, é possível projetar a arquitetura, a organização do sistema e a organização dos dados

## *Arquitetura:*

O optScan, como visto anteriormente, é um sistema composto por três grandes módulos. Sendo assim, o pipeline é feito com uso de mais de um software que funcionam em conjunto para alcançar o objetivo final – uma nuvem de pontos sintética.

a. Construtor de cenário virtual:

As ferramentas presentes na Unity 3D permitem ao usuário construir uma cena virtual com objetos 3D, onde cada um desses objetos possui propriedades importantes que serão importadas para os outros módulos, como posição, rotação e escala no mundo virtual. Outra propriedade importante do objeto é o Id, um número de identificação único para cada classe de objeto presente na cena atual. O objeto recebe esta etiqueta especial quando o usuário o coloca na cena. Esta propriedade será passada para a nuvem de pontos automaticamente, atribuindo rótulos de classe para cada ponto, tornando mais rápido para métodos de deep learning realizar a segmentação semântica.

Após a construção da cena, o usuário pode usar o objeto câmera no Unity para escolher as posições que o sensor virtual fará a varredura posteriormente nos outros módulos. Cada nuvem de pontos gerada posteriormente a partir deste caminho deve se parecer com a imagem vista no recurso de visualização da câmera do Unity. A Figura 3 mostra um exemplo, conforme as câmeras são posicionadas para definir o caminho do sensor do usuário para digitalizar uma árvore de Natal de um poço de petróleo.

Tela de jogo de vídeo game

Descrição gerada automaticamente com confiança média

Figura 3 - Construção de uma cena simples com objetos 3D e posicionamento da câmera na Unity.

Finalmente, o módulo processa a cena e produz três tipos de arquivos:

• Descrição da cena: Contém as quatro propriedades mencionadas anteriormente para cada objeto da cena virtual.

• Modelos 3D OBJ: No caso do optScan, um dos módulos necessita de arquivos 3D com a extensão .obj. Assim, o módulo converte os modelos 3D permitidos no Unity em .obj.

• Caminho do sensor: contém a posição e a orientação de cada câmera definida como a posição de varredura do sensor pelo usuário.

b. Modelo do Sensor Virtual:

Algumas adaptações tiveram que ser feitas nas funções e características do algoritmo de raytracing para melhor modelar o simulador de sensor nas condições desejadas. As principais adaptações foram:

• Plano esférico: os sensores LiDAR geralmente fazem a varredura de uma área dentro de um arco de circunferência, com um ângulo de inclinação definido pelo fabricante. Como o raytracing usa um plano cartesiano, a conversão para um plano esférico foi necessária. Isso traz distorções cônicas típicas de varredura com LiDAR.

• Ângulo de disparo: Como resultado da mudança para o plano esférico, o número de raios disparados depende do tamanho da tela de simulação e do passo angular definido pelo usuário. Assim, os raios agora respeitam um ângulo de azimute máximo (horizontal) e um ângulo de elevação máximo (vertical). Isso emula os ângulos operacionais de um LiDAR.

• Alcance máximo de detecção: A distância máxima dos raios disparados é definida pelo usuário de acordo com a distância de operação do LiDAR. Isso significa que objetos fora desse limite não são registrados pelo sensor virtual, como ocorre com um sensor real.

• Raycasting: Para esta etapa, os reflexos dos raios não estão sendo usados, pois dependem dos materiais do objeto e de outros problemas mais complexos, como dispersão de luz, para reproduzir esses fenômenos na simulação de forma adequada. Apenas a parte de raycasting do algoritmo de raytracing está sendo usada (traçando um caminho entre o raio e a colisão de um objeto).

Para o modelo de erro, foi realizada uma abordagem empírica por meio de testes experimentais com um sensor LiDAR real. Por exemplo, para o modelo URG-04LX-UG01 da Hokuyo constatou-se que as propriedades estatísticas desse conjunto obtiveram um erro por equação de distância com ajuste de curva, mostrada a seguir com RMSE de 3,69 mm.

|  |  |
| --- | --- |
|  | (2) |

Além disso, um ruído gaussiano N (0,3) foi adicionado para representar os erros aleatórios nas medições.

c. Gerador de Nuvens de Pontos Sintéticos:

O processo compreende duas etapas, usando aplicativos internos do módulo. Primeiro, o bloco Simulation recria os alvos digitais matematicamente dentro do OptiX por meio dos arquivos criados no módulo Cenário Virtual. Em seguida, o escaneamento virtual é realizado pelo módulo Sensor Virtual, que obtém os dados brutos da cena quando um raio disparado por ele colide com um objeto. A distância de intercessão obtida a partir dessas colisões são os dados brutos deste processo de imagem 3D. Dadas essas distâncias, a posição 3D da colisão pode ser calculada, criando assim o ponto em relação à superfície colidida pelo raio. Este vetor com coordenadas (x, y, z) é calculado da seguinte maneira:

|  |  |
| --- | --- |
|  | (3) |

onde é o vetor de origem do raio, é o vetor de direção obtido dos vetores de orientação (Up, Right, Forward) e ângulos do sensor, d é a distância de interseção da colisão, o erro é dado pela equação (2), e o ruído é o erro gaussiano aleatório. Esses dados são então organizados na forma de nuvens de pontos sintéticos para cada posição definida no módulo de Cena Virtual, conforme ilustrado na Figura 4.

Calendário

Descrição gerada automaticamente

Figura 4 - No topo, visão da câmera no construtor de cenas em que o sensor virtual foi posicionado. Embaixo, as nuvens de pontos resultantes de cada posição.

Na maioria dos procedimentos, essas nuvens de pontos sintéticas individuais precisam ser unificadas em um único modelo 3D. Portanto, entra-se na segunda etapa deste módulo, o aplicativo Merge. O aplicativo recebe como entrada as nuvens de pontos selecionadas pelo usuário dentre as geradas na simulação e, em seguida, os dados são combinados aos pares, iniciando-se um processo de comparação entre os pontos existentes nessas nuvens de pontos provisórias. O objetivo desta comparação é definir a existência de pontos redundantes, ou seja, pontos no conjunto de dados com posição 3D correspondente ou dentro de um limite de proximidade definido pelo usuário. Este limite irá variar entre 0 mm e 1 mm. A eficiência deste método dependerá dos parâmetros metrológicos configurados no módulo Sensor Virtual. Os pontos redundantes são então agrupados e eliminados da nuvem de pontos provisória. Essa filtragem é chamada de decimação espacial.

O processo de combinação, comparação e dizimação será repetido até que todo o conjunto de dados seja transformado em uma nuvem de pontos única. No final, a nuvem de pontos é salva nos formatos XYZ e PLY.

Cada ponto nessas nuvens tem as seguintes propriedades:

• Coordenadas 3D: Define a posição do ponto em relação ao cenário virtual construído (x, y, z). São as informações básicas de uma nuvem de pontos. A visualização dos pontos é mostrada na Figura 5a.

• Vetor normal: São vetores perpendiculares à superfície dos objetos detectados. Usado para definições de iluminação de materiais e útil em várias rotinas de segmentação de dados. A visualização do vetor em ação é mostrada na Figura 5b.

• Cor: Definido na forma de vetores RGB. Útil para verificar a correspondência da nuvem de pontos com o modelo 3D. Alguns sistemas LiDAR já possuem câmeras no espectro visível para obter informações de cores, pois o sensor sozinho não consegue adquirir esses dados. A visualização da cor é mostrada na Figura 5c.

• Número de Identificação: Este número é um identificador único definido pelo usuário ao criar a cena, conforme mencionado anteriormente. Essas informações facilitam a visualização de diferentes objetos dentro da mesma nuvem, o que também facilita os processos de catalogação e segmentação de dados. A visualização do Id é mostrada na Figura 5d.

Mapa

Descrição gerada automaticamente

Figura 5 - Propriedades diferentes da nuvem de pontos sintéticos: (a) Pontos 3D; (b) Vetores normais; (c) Cor; (d) Id.

## *Diagrama de Organização dos Dados:*

No diagrama de fluxo de dados, na figura 6, é possível ver:

– Dados de entrada e/ou saída (representado pelos paralelogramos) – são os dados necessários para funcionamento e eventuais resultados dos módulos e blocos internos.

– Fluxo dos dados (representados pelas setas) – representam como os dados

fluem;

– Módulos (representado pelos retângulos) – responsáveis pela transformação dos dados.

– Blocos internos (representado pelos losangos) – aplicações internas do sistema responsáveis por processar os dados na forma de nuvem de pontos.

Diagrama, Forma

Descrição gerada automaticamente

Figura 6 – Fluxograma de dados: Paralelogramos são dados de entrada e/ou saída, retângulos são módulos e losangos são blocos internos.

## *Organização do Programa:*

Como mencionado, o programa é dividido em três módulos. Devido a proposta do trabalho que originou o optScan, ele não se trata de uma única arquitetura, mas um sistema composto por diversas arquiteturas com saídas processadas de forma a manterem o funcionamento do pipeline visto no diagrama de Organização de Dados. A seguir são apresentados as principais classes e funções de cada módulo.

1. *Virtual Scene Builder:*

Como a Unity 3D já providencia todas as ferramentas necessárias para manipulação de objetos 3D, somente foi necessário criar scripts para criar os arquivos de saída necessários para os próximos módulos.

**Data.cs:** Permite usuário acrescentar o label de identificação de classe e nome do objeto.

**Gerenciador.cs:** Esta classe é responsável por preparar o caminho de câmera e a conversão dos objetos 3D para o formato .OBJ. Ele obtém a posição e orientação de cada objeto do tipo câmera e converte para o sistema de coordenadas do NVIDIA Optix. Além disso, cria arquivos de comando para os próximos módulos da seguinte lista de comandos:

|  |  |
| --- | --- |
| Comando | Descrição |
| -m | --mesh <mesh\_file> | Especifica o objeto a ser carregado. <mesh\_file> é o caminho do objeto. |
| -c | --camera <camera\_file> | Especifica o caminho com as informações do caminho de câmera. <camera\_file> é o caminho deste arquivo. |
| --position <x,y,z> | Especifica a posição do objeto em relação ao mundo em coordenas cartesianas. |
| --rotation <x,y,z,angle> | Especifica a rotação do objeto na forma de um quartenion. |
| --scale <float> | Especifica a escala do objeto |
| --id <int> | Especifica a identificação de classe do objeto definida pelo usuário |

Tabela 3 - Comandos gerados pelo Gerenciador.cs

1. *Sensor Simulator:*

Todo o modelo comportamental do sensor encontra-se em três programas CUDA.

**closesthit.cu:** responsável pelos acertos mais próximos da origem do raio disparado pelo raygeneration.cu para a superfície de um objeto.Os modelos de erros também são acrescentados aqui.

**raygeneration.cu:** responsável por gerar o raio com as características corretas baseadas em parâmetros metrológicos de um LiDAR real.

**miss.cu:** responsável por tratar os raios disparados que não encontram nenhum objeto para colisão.

1. *Synthetic Point Cloud Generator*:

Este modulo possui duas grandes etapas, dentro das seguintes classes:

**Application.cpp:** toda a inicialização da engine do OptiX, a criação do contexto com as informações e características pertinentes da cena virtual gerada e do sensor virtual.

**mergeVertices.cpp:** responsável pelo processo de unificação das nuvens de pontos individuais por meio de um processo de decimação e combinação.

Após o uso da Unity 3D para facilitar na construção dos arquivos de cenários, o pipeline dos módulos seguintes pode ser todo chamado por um simples arquivo .bat, como o do código abaixo.

@Echo off

ECHO \*\*\*\*Running optScan

CALL "%~dp0optScan.exe" --configure "samples\config\_commands.txt"

ECHO \*\*\*\*Running mergeVertices

CALL %~dp0mergeVertices.exe -c "%~dp0data\config.txt"

ECHO \*\*\*DONE

PAUSE

CLS

EXIT

Desde que as dll do OptiX e do Geogram estejam na pasta do release, não há necessidade do usuário instalar estas bibliotecas.

# **Teste** **Efetuado**:

Para verificar a qualidade da simulação e dos dados gerados ao final do processo, foi feito um processo de validação por comparação com um sensor real. Desta forma, ao final dos experimentos, é possível usar métricas de erro de calibração, tais como num sensor real.

## *Roteiro*:

1 - O processo começa com a análise dos parâmetros metrológicos do sensor de referência escolhido.

2 – Configura-se o módulo Sensor Virtual com os parâmetros metrológicos do LiDAR e uma cena virtual baseada em um objeto cilíndrico simples com baixa refletância.

3 - Tanto a varredura em campo quanto a virtual devem ser feitas com seus respectivos sensores posicionados de modo que o raio central do LiDAR atinja o centro do objeto em

a mesma altura.

4 - Uma série de varreduras foram realizadas, resultando em dados brutos na forma de interseção distâncias para o LiDAR e sua contraparte digital.

5 - Esses dados brutos foram plotados contra seus ângulos de disparo de raios, resultando em mapas de interseção.

6 – Avaliação deve ser feita em todas as faixas de operação que o LiDAR possuir.

o LiDAR.

7 - Uma vez que esses mapas foram obtidos, o erro quadrático médio (RMSE) entre os dados virtuais e os dados de referência são calculados para cada distância de interesse.

Como exemplo, na figura 7 seguem os mapas de intercessão para para o modelo URG-04LX-UG01 da Hokuyo mencionado como exemplo na seção sobre o modelo de erro.

Gráfico

Descrição gerada automaticamente

Figura 7 - Mapas de intersecção: (a) A 700 mm do raio central do sensor; (b) A 1500 mm do raio central do sensor. Em ambos os gráficos, os pontos de interseção do Hokuyo LiDAR são marcados com 'o', enquanto os do Simulador de Sensor (modelo híbrido) são marcados com "x"

Neste caso, para a posição do raio central do sensor a 700 mm, o RMSE encontrado foi de 7,22 mm, enquanto para a posição do raio central do sensor a 1500 mm, o RMSE foi de 14,13 mm. Isso mostra a robustez de todo o simulador para este caso, principalmente do modelo de comportamento do sensor, que se comportou de forma semelhante a um sensor LiDAR real. Além disso, todo esse processo funciona como um processo de calibração para o sensor virtual em comparação com o sensor de referência, assumindo que ele foi previamente calibrado corretamente.

## *Possibilidades de Melhorias no Projeto*:

Possíveis melhorias seriam:

- Substituir os módulos para arquiteturas mais coerentes entre si, por exemplo, todos utilizando a mesma linguagem de programação;

- Implementar mais propriedades LiDAR, como divergência do feixe, tamanho do feixe e número de raios por ângulo;

- Desenvolvimento de um módulo baseado em CCD para capturar informações de cores de forma realista;

- Usar a recursividade dos algoritmos de raytracing para incluir reflexos de luz com base no material do objeto.