UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

BIBLIOTECA PARA DETECÇÃO DE RELEVOS BASEADA EM LUZ ESTRUTURADA E PADRÃO DE PROJEÇÃO BINÁRIO

LUCAS EDUARDO SCHLÖGL

BLUMENAU
2018

LUCAS EDUARDO SCHLÖGL

BIBLIOTECA PARA DETEÇÃO DE RELEVOS BASEADA EM LUZ ESTRUTURADA E PADRÃO DE PROJEÇÃO BINÁRIO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, Mestre - Orientador

BLUMENAU
2018

BIBLIOTECA PARA DETECÇÃO DE RELEVOS BASEADA EM LUZ ESTRUTURADA E PADRÃO DE PROJEÇÃO BINÁRIO

Por

LUCAS EDUARDO SCHLÖGL

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, Mestre – Orientador, FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
Membro:	Prof. Maurício Capobianco Lopes, Doutor – FURB

Blumenau, 11 de julho de 2018

Dedico este trabalho a minha família e meus amigos.

AGRADECIMENTOS

Agradeço a minha mãe Sônia por todo o incentivo dado durante a graduação e ao meu pai Edson por também me apoiar durante este período. Além dos demais familiares que convivo diariamente: minha irmã Luana e meu padrasto Marlos.

Ao meu orientador Dalton Solano dos Reis por me apoiar não somente neste projeto, mas durante grande parte dessa graduação.

Ao professor Aurélio Hoppe, que foi um dos professores que mais contribuiu para minha formação acadêmica e profissional.

Aos meus colegas, tanto da vida pessoal quanto do trabalho e universidade: Gabrielle Laís, Joana Forster, Renan Fiedler, João Paulo Ros, Rodrigo Hulsenbeck, João Serodio, João Mantova, Mauricio Gemelli, Bruno Soares, Evandro Schmitz e a todos aqueles que possa ter esquecido.

Aos amigos e colegas que fiz no movimento estudantil da FURB, CALCOMP e DCE, que me ensinaram como realmente uma universidade funciona.

Aos professores com quem convivi quase que diariamente durante um grande período da minha jornada universitária como Luciana Araújo, Maurício Capobianco e Everaldo Grahl.

“What I can’t create, I can’t understand.”

Richard Feynman

RESUMO

Este trabalho apresenta o desenvolvimento de uma biblioteca para detecção de relevos baseada em luz estruturada e padrões binários que surge como alternativa ao trabalho iniciado por Storz (2017). Analisando as distorções dos padrões binários projetados sobre a cena e realizando a decodificação das imagens capturadas é possível realizar a reconstrução da cena em 3D com sucesso. Se comparado com Storz (2017) a biblioteca desenvolvida altera a forma de calibração para variáveis definidas como valor `Z` e limiares de branco e preto, eliminado o processo demorado baseado no tabuleiro de Xadrez. Para a visualização da cena reconstruída, é possível exportar a nuvem de pontos para os formatos `XYZ` e `PLY` além de ser possível obter o mapa de disparidade, sendo que com essas nuvens de pontos é possível importa-las em outros programas como Unity e Meshlab. Atualmente a biblioteca está desenvolvida em C++ e utiliza OpenCV para realizar as operações sobre as imagens e todo o código roda somente em CPU. A portabilidade da biblioteca também foi realizada com o objetivo de atingir a maior quantidade de sistemas operacionais possível, tendo em vista que a biblioteca de Storz (2017) funciona apenas no Windows. Analisando os resultados, foi possível notar que o algoritmo implementado é aproximadamente 310 vezes mais rápido se comparado com o a biblioteca de Storz (2017), porém o processo de captura de imagens é mais demorado em função de ter que capturar múltiplas imagens. A biblioteca desenvolvida não pode ser utilizada em cenários de tempo real, pois o tempo de captura das imagens leva aproximadamente 5 segundos. Também analisou-se o impacto da iluminação na reconstrução e pode verificar-se que o impacto foi mínimo.

Palavras-chave: Reconstrução 3D. Luz estruturada. Padrão binário. Tempo real.

ABSTRACT

This work presents the development of a relief reconstruction library based on structured light and binary patterns that emerges as an alternative to the work started by Storz (2017). By analyzing the distortions of the binary patterns projected onto the scene and performing the decoding of the captured images it is possible to successfully reconstruct the 3D scene. Compared with Storz (2017) the developed library changes the form of calibration for variables defined as `z` value and white and black thresholds, eliminating the time-consuming process of calibrating with chessboard. In order to view the reconstructed scene, it's possible to export the point clouds to `XYZ` and `PLY` files in addition to being able to get the disparity map, and with these point clouds it's possible to import them in other programs such as Unity and Meshlab. Currently the library is developed in C ++ and uses OpenCV to perform operations on the images and all code runs only on CPU. The library portability was also performed with the goal of reaching as many operating systems as possible, since the Storz (2017) library works only on Windows. Analyzing the results, it was possible to notice that the implemented algorithm is approximately 310 times faster when compared to the Storz (2017) library, but the process of image capture is longer due to having to capture multiple images. The developed library can't be used in real-time scenarios since the capture time of the images takes approximately 5 seconds. We also analyzed the impact of lighting on the reconstruction and it can be verified that the impact was minimal.

Key-words: 3D reconstruction. Structured light. Binary code pattern. Real time.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D – Três dimensões

CPU – Central Process Unit

FPS – Frames Per Second

GPGPU – General-Purpose Graphics Processing Unit

GPU – Graphics Processing Unit

RF – Requisito Funcional

RNF – Requisito Não Funcional

SUMÁRIO

1 INTRODUÇÃO

Com o avanço das técnicas de visão computacional e com a melhoria dos equipamentos óticos digitais de alta precisão, a área de reconstrução 3D tomou um grande impulso e deu origem à área que hoje se chama de digitalização tridimensional (CELANI; CANCHERINI, 2009, p. 1). Os campos de utilização são os mais variados possíveis, os trabalhos de Celani e Cancherini (2009) e Chen et al. (2010) citam a reconstrução 3D em áreas como mecânica industrial, navegação de robôs e até arquitetura. A reconstrução também pode ser utilizada em projetos educacionais, como o Projeto Caixa e Água (2015), que tem como objetivo realizar a reconstrução do relevo em tempo real simulando uma cidade destacando principalmente as características naturais como os recursos hídricos. Celani e Chancerini (2009, p. 3) demonstram *scanners* 3D que variam desde projetos caseiros que custam aproximadamente 110 dólares até projetos profissionais que custam 100.000 dólares. Chen et al. (2009, p. 1) afirma que a reconstrução 3D por luz estruturada tem como propriedades ser de baixo custo, precisa e com capacidade de ser em tempo real.

De acordo com o dicionário Michaelis em sua versão *online*, o significado de relevo são as diferentes formas da superfície terrestre, caracterizadas por saliências e depressões. Para se obter essas informações do relevo, é necessário realizar a extração das informações geométricas de uma cena para que ela possa ser construída computacionalmente. Essa reconstrução computacional das informações geométricas tridimensionais vem sendo estudada pela área de visão computacional. As técnicas de extração de informações geométricas a partir de ambientes podem ser classificadas em técnicas passivas e técnicas ativas. As técnicas ativas consistem em introduzir algum tipo de sinal no ambiente e as informações relativas à geometria daquele ambiente são extraídas através do comportamento do sinal. Já as técnicas passivas consistem em apenas coletar imagens e, com base na análise dessas imagens, tentar estimar ângulos, distâncias e posições (FERNANDES, 2005, p. 6). Dentro das técnicas ativas, a triangularização ativa é, segundo Fernandes (2005, p. 10), o método mais antigo para medir a profundidade dos pontos no espaço e é uma das técnicas mais comuns. Essa técnica utiliza-se da lei dos senos para medir ângulos e distâncias.

Conforme apresentado, é desenvolvido uma biblioteca que surge como alternativa ao trabalho iniciado por Storz (2017) com o objetivo de realizar a reconstrução 3D de uma cena, realizando a alteração das técnicas que foram propostas por Storz (2017) que consistem em: Calibração utilizando um tabuleiro de Xadrez, reconstrução por luz estruturada baseando-se em

uma projeção colorida baseada na sequência de De Brujin. O presente trabalho altera as técnicas para reconstrução por luz estruturada através de múltiplas imagens sendo os padrões binários.

1.1 OBJETIVOS

O objetivo deste trabalho é uma biblioteca para detecção de relevos utilizando a técnica de luz estruturada com padrões de projeção binários.

Os objetivos específicos são:

a) definir um padrão para ser projetado na superfície a ser reconstruída;
b) obter um tempo de reconstrução da cena para a utilização da biblioteca na reconstrução em tempo real;
c) gerar uma nuvem de pontos para que outros programas possam reconstruir o terreno.

1.2 ESTRUTURA

O presente trabalho está subdividido em cinco capítulos: introdução, fundamentação teórica, trabalho anterior, desenvolvimento e conclusões.

Na fundamentação teórica serão apresentados quatro trabalhos com funcionalidades semelhantes ao trabalho desenvolvido. Também serão apresentados os conhecimentos básicos na área de reconstrução 3D, padrões de projeção sobre a superfície e sobre a representação do relevo. No desenvolvimento serão apresentados detalhes técnicos da biblioteca, incluindo as especificações técnicas e algoritmos utilizados durante o processo. Por fim, no capítulo 4 são redigidas as conclusões finais alcançadas durante a produção deste trabalho juntamente com sugestões para futuras extensões.

2 FUNDAMENTAÇÃO TEÓRICA

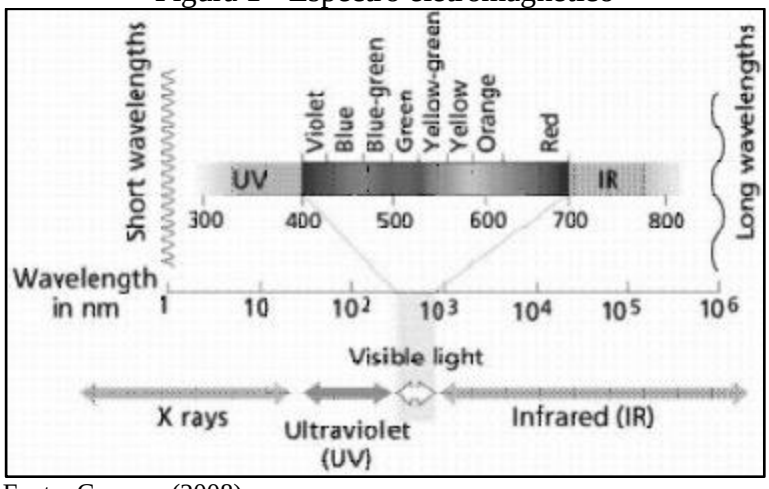
Este capítulo está organizado em três seções. A seção 2.1 aborda as formas de representação do relevo. A seção 2.2 descreve o processo de reconstrução 3D. Por fim, a seção 0 apresenta quatro trabalhos correlatos.

2.1 REPRESENTAÇÃO DE RELEVOS

No processo de representação de relevos, onde as características do relevo são representadas em duas dimensões, é necessário realizar a representação fiel dos dois principais elementos do relevo: a altitude e a declividade (diferença de altitude entre dois pontos). Com isso, tal representação pode ser realizada utilizando dois métodos, a carta hipsométrica e curvas de nível. Vale salientar que ambos os métodos podem ser utilizados em um mesmo mapa (CAMPOS, 2008).

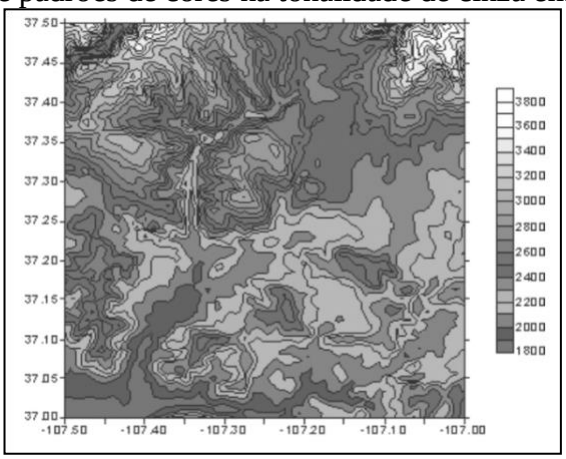
A carta hipsométrica utiliza cores para representar uma determinada classe de altitude em uma parte do relevo, sendo tais cores baseadas no espectro eletromagnético (Figura 1), estabelecendo que as cores seriam escolhidas do intervalo mais baixo para o mais alto de acordo com o espectro eletromagnético, partindo do verde até o vermelho (CAMPOS, 2008). Mendonça (2007) também corrobora com a ideia de Campos (2008), citando uma convenção de altitudes: as mais elevadas sendo representadas com tons de marrom, tons de amarelo para representar médias altitudes e o verde para baixas altitudes (Figura 2).

Figura 1 - Espectro eletromagnético



Fonte: Campos (2008)

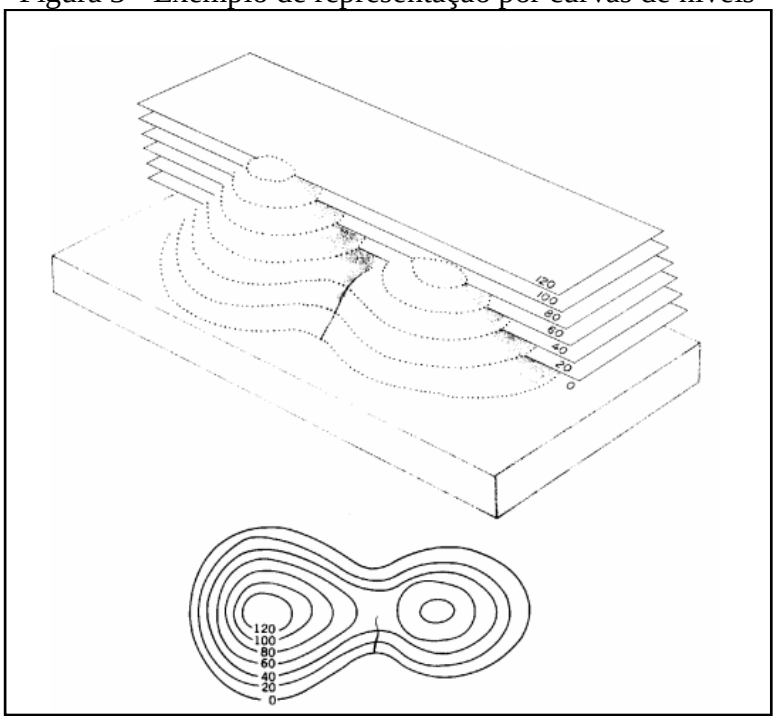
Figura 2 - Exemplo de padrões de cores na tonalidade de cinza em cartas hipsométricas



Fonte: Campos (2008).

Na representação através de curvas de nível, as variações de altitudes são representadas por linhas que demonstram intervalos verticais constantes (CAMPOS, 2008) no qual, de acordo com a Figura 3, é possível observar que cada linha representa uma diferença constante de 20 metros de altura. Para facilitar o entendimento do funcionamento das curvas de nível, que são representadas bidimensionalmente, Campos (2008) demonstra um exemplo tridimensional de como as diferenças de altitude são representadas. Quando uma linha está muito distante uma da outra significa que o terreno apresenta uma declividade suave naquele ponto e quando apresentam linhas muito próximas, uma grande declividade (MENDONÇA, 2007). Esse aspecto pode ser visto na parte central do terreno demonstrado na Figura 3, onde se observa a inexistência de linhas mais próximas, se comparadas com as laterais.

Figura 3 - Exemplo de representação por curvas de níveis



Fonte: Campos (2008).

2.2 RECONSTRUÇÃO 3D

Essa seção está dividida na subseção 2.2.1 que demonstra o funcionamento da reconstrução através da técnica de triangularização ativa, a subseção 2.2.2 trata sobre os padrões que são projetados sobre a cena e por último, a subseção 2.2.3 trata sobre a captura de imagens e a decodificação delas.

2.2.1 Reconstrução de relevo por triangularização ativa

As técnicas para obter informações de uma cena podem ser divididas em duas categorias: técnicas ativas e passivas. As técnicas passivas consistem em obter imagens de uma cena e, a partir da análise, estimar os ângulos, profundidades e as posições. Já as técnicas ativas, consistem na inserção de um sinal na cena, que pode ser luminoso ou ultrassônico para, com isso, realizar a extração das informações a partir da análise das alterações desse sinal (FERNANDES, 2005, p. 6).

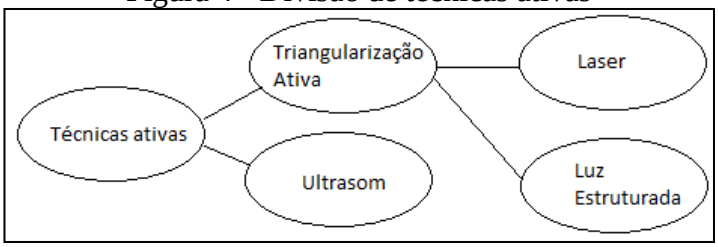
Ao trabalhar com técnicas baseadas em imagens, Fernandes (2005, p. 8) realiza uma enumeração de problemas que podem ocorrer:

a) problema de correspondência: quando se trabalha com mais de uma imagem, é necessário relacionar os mesmos pontos nas diferentes imagens;
b) as estruturas geométricas podem não estar bem definidas por problemas de oclusão (algum objeto na frente da cena): este problema pode dificultar a resolução para o problema de correspondência, tendo em vista que o mesmo ponto em uma imagem pode estar sendo ocultado em razão da oclusão;
c) problemas no ajuste do foco ou a baixa resolução das imagens;
d) distorção radial das imagens: é um problema introduzido pela lente da câmera e se caracteriza por distorcer a imagem à medida que os pontos se afastam do “centro” da imagem.

Dependendo das características da técnica utilizada, é necessário verificar se a calibração de parâmetros torna-se necessária ou não. Esses parâmetros podem ser internos (distorção radial, dimensão do pixel, distância focal e etc.) ou externos (posição da câmera, orientação e etc.) e podem ser informados totalmente, parcialmente ou calculados ao longo do processo. Nesse processo é necessário tomar grande cuidado, caso contrário o sistema poderá causar grandes erros nas medições estimadas (FERNANDES, 2005, p. 9).

Nas técnicas ativas, existem algumas subdivisões que se caracterizam pela inserção de diferentes tipos de sinais na cena como a inserção de sinais ultrassônicos ou então baseado na inserção de luzes ou lasers na cena. Essas subdivisões podem ser vistas na Figura 4.

Figura 4 - Divisão de técnicas ativas



Fonte: adaptado de Fernandes (2005).

Segundo Cerani e Cancherini (2009, p. 1), as técnicas ativas subdividem-se em três tipos básicos de tecnologias:

a) comparação de imagens de um mesmo objeto através de pontos diferentes (estereopar);
b) análise da deformação da luz projetada sobre um objeto (*luz estruturada*);
c) obtenção da distância e ângulo por meio da contagem de tempo que a radiação leva até tocar um objeto (*time of flight*).

Além disso, cada subdivisão tem alcances recomendados diferentes, que podem ser vistos no Quadro 1.

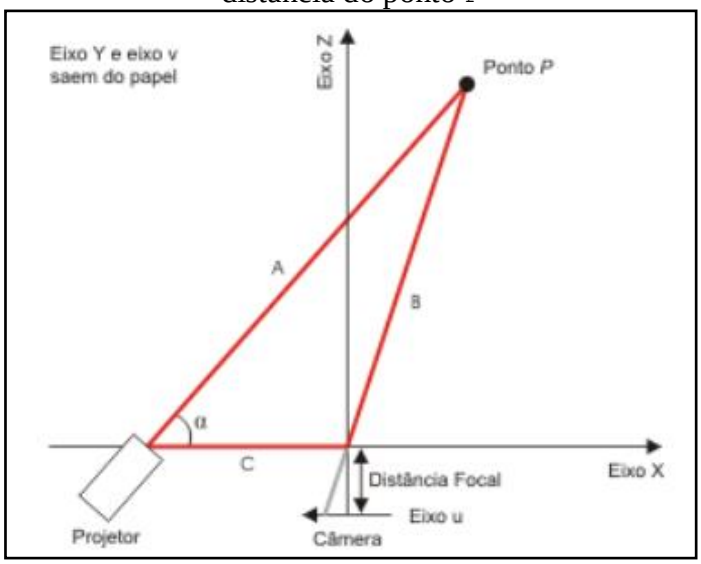
Quadro 1 - Subdivisões das técnicas ativas e seus alcances

Alcance / Técnica	Curto/médio (objeto)	Longo (edifício)	Muito longo (área urbana)
estereopar	X	X	X
luz estruturada	X	X	
time of flight		X	X

Fonte: adaptado de Cerani e Cancherini (2009, p. 2).

Para o desenvolvimento desta biblioteca, o método escolhido é o por triangularização ativa, utilizando uma luz estruturada como sinal para inserção na cena. De acordo com Fernandes (2005, p. 10), este é provavelmente o método mais antigo e o mais utilizado para a medição das formas geométricas presentes no ambiente. O método consiste em utilizar a lei dos senos para determinar o ângulo e o comprimento dos lados a fim de determinar a distância até um ponto. Como na técnica ativa alguns parâmetros como posição do projetor em relação a câmera e o ângulo são conhecidos, a distância até o ponto `P` que representa a profundidade de um ponto que pode ser obtida pela lei dos senos (Figura 5). Conhecendo essa distância, se faz necessário apenas a projeção de um padrão de luz conhecido (normalmente linhas) para que se calcule a deformação do padrão em relação à altura do ponto `P`.

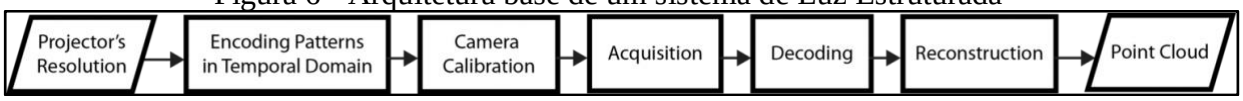
Figura 5 - Exemplo prático de como a lei dos senos pode ser utilizada para determinar a distância do ponto P



Fonte: Fernandes (2005).

Herakleous e Poullis (2016, p. 2) descrevem que o processo de reconstrução por luz estruturada envolve a projeção de padrões conhecidos em uma cena a ser reconstruída, enquanto realiza-se a captura por uma ou mais câmeras. Com o objetivo de reconstruir corretamente cada ponto na cena, o padrão projetado é codificado de um modo em que cada ponto pode ser unicamente identificado pelas câmeras, provendo um método eficiente para calcular com acurácia os pontos 3D. Herakleous e Poullis (2016, p. 3) também define uma arquitetura geral de como um sistema de luz estruturada é baseado, iniciando-se pelas configurações de projetor, codificação de padrões e calibração da câmera. Então se parte para parte de reconstrução, que inicia-se com o processo de aquisição de imagens, decodificação dos pontos e então a reconstrução usando a triangularização é realizada. Ao final, obtêm-se uma nuvem de pontos para que o resultado possa ser visto visualmente.

Figura 6 - Arquitetura base de um sistema de Luz Estruturada



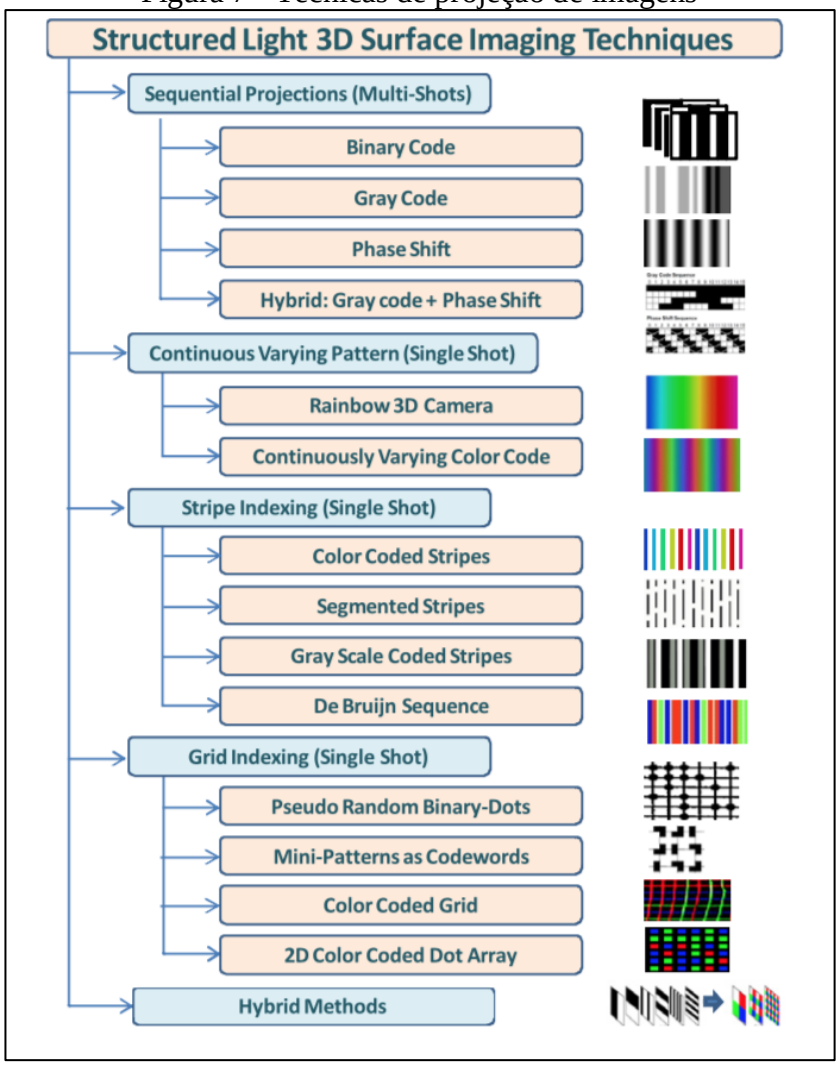
Fonte: Herakleous e Poullis (2016, p. 2)

2.2.2 Padrões de projeção

Chen et al. (2010, p. 2) descreve que a maioria dos sistemas de luz estruturada exige que sejam projetados padrões codificados sobre a cena e a escolha do padrão afeta diretamente no resultado da reconstrução 3D. Sendo assim, Geng (2010, p. 5) afirma que baseado na análise da distorção do padrão de luz projetado sobre uma cena em comparação com o padrão não distorcido, é possível estimar a forma 3D da cena com precisão. Inúmeras técnicas de projeção de padrões estão disponíveis sendo que cada uma tem suas vantagens e desvantagens de acordo

com um objetivo específico, sendo que esses padrões também podem ser combinados entre si para chegarem a um resultado (GENG, 2010, p. 5). Geng (2010, p. 6) classifica em cinco técnicas principais divididas na quantidade de fotos: Single Shots e Multi-Shots. De acordo com a classificação, o único padrão Multi-Shot é chamado de Sequential Projections sendo as técnicas nessa categoria caracterizadas principalmente pela projeção de padrões baseados em linhas e colunas nas escalas de cinza. Já os padrões de uma imagem (Single Shot) são divididos em Continous Varying Pattern, Stripe Indexing, Grind Indexing e são caracterizados por padrões baseados em um dicionário de palavras com o objetivo que diversas áreas da cena sejam subdivididas em padrões únicos. Há também a categoria híbrida que é resultante da combinação de diferentes tipos de padrões. A Figura 7 demonstra a divisão das técnicas conforme apresentado por Geng (2010). Herakleous e Poullis (2016, p. 3) afirmam que apesar de existirem diversas categorias de padrões que podem ser gerados, os mais populares são: Binary Code e Gray Code.

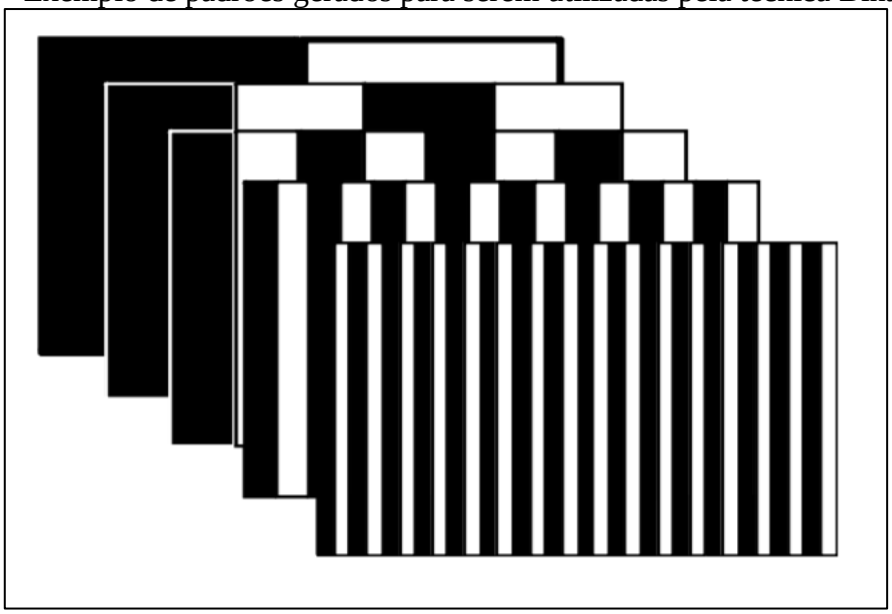
Figura 7 - Técnicas de projeção de imagens



Fonte: Geng (2010, p. 6)

De acordo com Geng (2010, p.7) a técnica Binary Code utiliza listras pretas e brancas para formar uma sequência de imagens a serem projetadas, onde cada ponto da superfície da cena possui um código binário único. Além disso, Geng (2010, p.7) afirma que essa técnica caracteriza-se por ser bem confiável e ser menos sensível ao tipo de superfície da cena pois apenas valores binários existem na cena. Entretanto para se conseguir uma boa reconstrução da superfície, uma grande sequência de padrões precisa ser projetado, fazendo com que a duração de captura das imagens seja alta se comparada com padrões de apenas uma imagem. Um exemplo de padrões gerados nessa técnica pode ser visto na Figura 8.

Figura 8 - Exemplo de padrões gerados para serem utilizadas pela técnica Binary Code

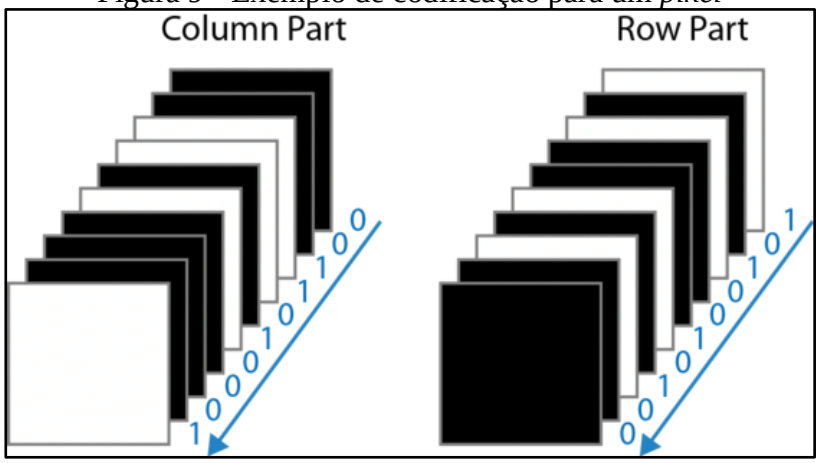


Fonte: Geng (2010, p. 7).

Na codificação binária (Binary Code), cada ponto contém um número binário onde em cada posição do número binário contém a informação da cor detectada naquele determinado ponto com base em um determinado padrão que estava sendo projetado, assumindo como padrão o valor 0 para pontos pretos e 1 para pontos brancos (HERAKLEOUS; POULLIS, 2016, p. 4). Um exemplo desse processo de codificação do número binário em um ponto `P` da imagem pode ser visto na Figura 9. Não muito diferente da técnica Binary Coding, a técnica Gray-code difere-se apenas por guardar o número de Gray em cada ponto da imagem. A diferença entre um número de Gray e um número binário podem ser vistas na Figura 10, onde nela é possível perceber que em dois padrões consecutivos, apenas um *bit* será diferente. Segundo Herakleous e Poullis (2016, p. 5), apesar de serem similares, a técnica de padrão baseada em Gray-code é melhor se comparada com a Binary code, pois em dois padrões sequenciais a largura das colunas geradas nos padrões serão maiores, reduzindo efeitos não esperados como *color*

bleeding no projetor ou nas imagens capturadas. Essa diferença na largura das colunas geradas pode ser vista na Figura 11.

Figura 9 - Exemplo de codificação para um *pixel*



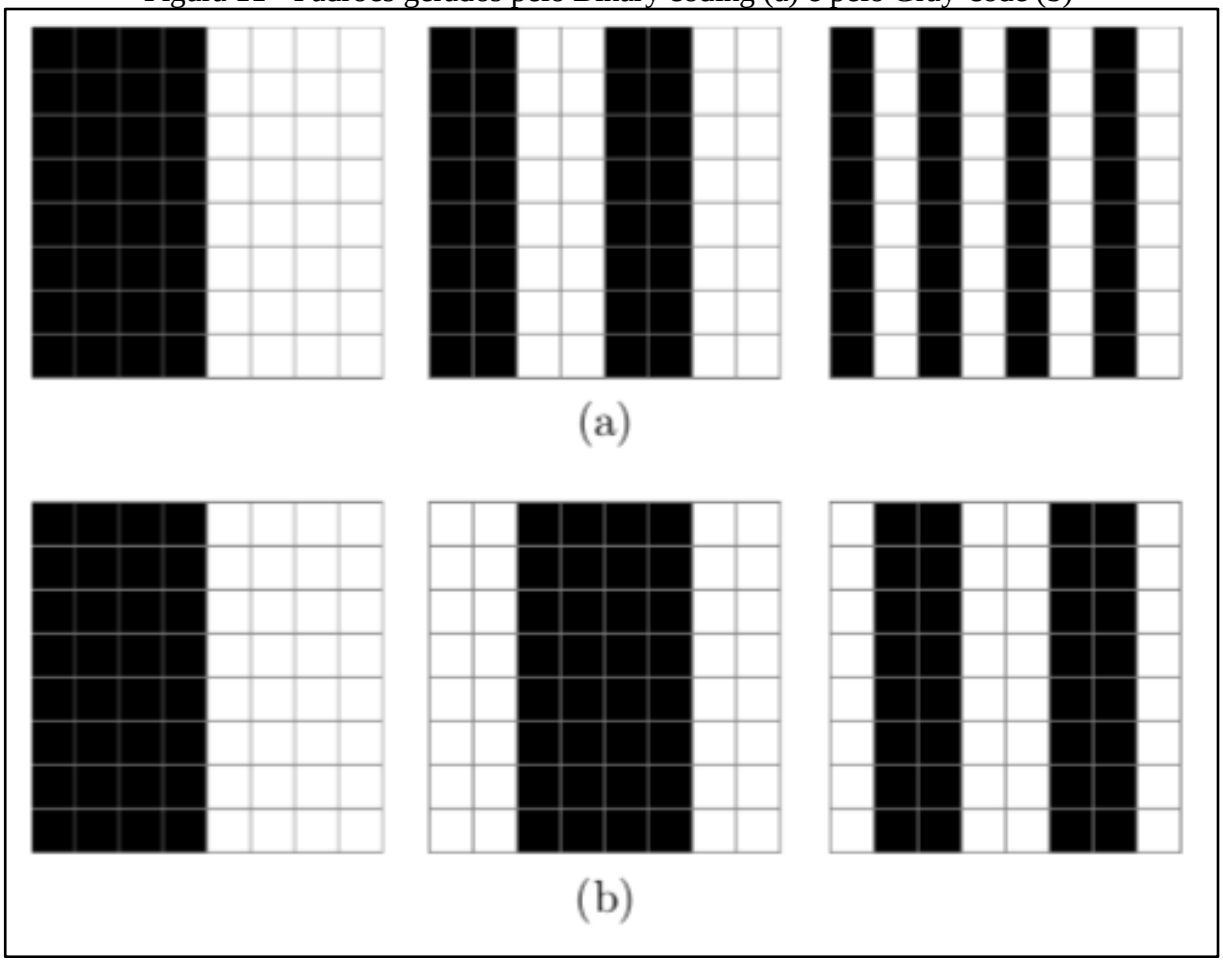
Fonte: Herakleous e Poullis (2016, p. 4)

Figura 10 - Número decimal e suas representações em Gray-code e Binary code

Decimal Value	Gray-code	Binary code
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Fonte: Herakleous e Poullis (2016, p. 5)

Figura 11 - Padrões gerados pelo Binary coding (a) e pelo Gray-code (b)



Fonte: Herakleous e Poullis (2016, p. 6)

2.2.3 Captura e decodificação de imagens

Para analisar a distorção que os padrões sofrem sobre uma cena, existe a necessidade de se capturar imagens contendo as distorções dos padrões projetados sobre uma cena. Para Herakleous e Poullis (2016, p. 9), o processo de captura precisa considerar:

a) o objeto precisa permanecer estático durante a captura;
b) as configurações do projetor e câmera devem ser ajustadas de acordo com a iluminação do ambiente;
c) caso o processo de aquisição de seja automático, é recomendado inserir um tempo de espera após cada captura da imagem, com o objetivo de que a última imagem seja capturada e armazenada corretamente, eliminando comportamentos não esperados.

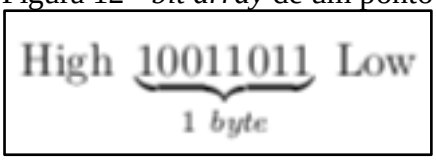
Feita a captura das imagens, inicia-se o procedimento de decodificação de cada ponto da imagem em um número decimal. Para Herakleous e Poullis (2016, p. 11), o processo de decodificação (de Binary code e Gray-code) segue os seguintes passos:

a) determinar se um ponto está branco ou preto (0 ou 1);

b) calcular a forma binária do ponto;
c) converter o número binário em um número decimal.

Esse procedimento é repetido pela quantidade de padrões que foram projetados sobre a cena. Em cada imagem capturada é realizada a decodificação do padrão e o o nível de importância dos padrões vai do primeiro até o ultimo, ou seja, o primeiro padrão é o que mais vai impactar no número decimal decodificado enquanto o último padrão será o que menos irá variar no número, conforme pode-se ver na Figura 12 se considerarmos a projeção de 8 padrões, que equivalem a 1 *byte*.

Figura 12 - *bit array* de um ponto



Fonte: Herakleous e Poullis (2016, p. 25)

2.3 TRABALHOS CORRELATOS

Neste capítulo são apresentados três trabalhos correlatos que possuem características semelhantes à proposta por esse trabalho. Na seção 2.3.1 é descrito o trabalho realizado por Li, Straub e Prautzsch (2004), que demonstra o processo para a detecção de relevos utilizando a triangularização ativa. A seção 2.3.2 apresenta o trabalho realizado por Mohan et al. (2011) demonstra uma abordagem diferente das que vinham sendo apresentadas na maioria dos trabalhos sobre técnicas de detecção de relevos passivas tendo em vista a situação adversa do cenário apresentado. A seção 2.3.3 descreve o trabalho de Pashaei e Mousavi (2013) que realiza o desenvolvimento de um *scanner* 3D de baixo custo utilizando a técnicas de visão estérea com luz estruturada. A seção 2.3.4 apresenta a biblioteca desenvolvida por Storz (2017) que utiliza luz estruturada e a projeção de um padrão baseado na sequência de De Brujin para realizar a reconstrução da cena.

2.3.1 Fast subpixel accurate reconstruction using color structured light

Li, Straub e Prautzsch (2004, p. 1) demonstram a criação de um método para a reconstrução de uma cena através da triangularização ativa, utilizando apenas uma imagem como a entrada dos dados, aumentando a robustez contra erros e alcançando uma melhor detecção das bordas. O trabalho utiliza como base para a triangularização uma câmera fotográfica como receptora da imagem e um projetor como emissor de luz estruturada (Figura 13). Segundo Li, Straub e Prautzsch (2004, p. 1), o projetor emite um conjunto de linhas sobre

o objeto alvo e a partir desse ponto, a câmera detecta a distorção das linhas, usando isso como base para extrair informações como a profundidade.

Figura 13 - Projetor e câmera utilizada para a triangularização

Fonte: Li, Straub, Prautzsch (2004, p. 1).

Do jeito que o trabalho foi projetado, é necessário realizar uma calibração do algoritmo, fornecendo dados explícitos como orientação e posição no espaço do projetor e da câmera antes que qualquer triangularização seja realizada. A partir da calibração, é necessário realizar a projeção de um padrão nos objetos da cena para que a triangularização tenha início. O artigo utiliza como padrão de projeção a sequência de De Brujin (Figura 14) que consiste em projetar várias linhas em diferentes cores baseado em um conjunto de números (LI; STRAUB; PRAUTZSCH, 2004, p. 2).

Figura 14 - Sequência de De Brujin

$$\mathbf{p}_i = (p_i^r, p_i^g, p_i^b) \in \{0,1\}^3$$

Fonte: Li, Straub e Prautzsch (2004, p. 2)

Na detecção das bordas, entre as linhas da sequência de De Brujin, surgem alguns problemas, como: luminosidade desfavorável, a textura dos objetos, qualidade da câmera e baixa resolução do projetor. O trabalho sugere limpar a imagem em três etapas (LI; STRAUB; PRAUTZSCH, 2004, p. 2):

a) cuidar do fenômeno chamado *crosstalk*, que é causado quando a luz ambiente interfere na luz emitida pelo projetor. Para corrigir, realiza-se uma normalização das imagens através de transformações lineares, para que a cor branca e preta realmente sejam essas cores;
b) remove-se áreas escuras da imagem, onde a intensidade da luz é menor que o limite definido pelo usuário;
c) um processo de suavização é aplicado se a imagem ainda contém muito ruído. Assumindo que a imagem contém um ruído Gaussiano, utiliza-se o filtro *low-pass* para redução do ruído.

Após reduzir os ruídos da imagem a detecção das bordas é realmente iniciada. Primeiramente é utilizado um algoritmo de programação dinâmica para a detecção das bordas que no final, gera uma matriz de custos. Com isso, consegue-se a rotulação das imagens projetadas (LI; STRAUB; PRAUTZSCH, 2004, p. 4).

Ao detectar as bordas com sucesso, é necessário calcular a profundidade das mesmas, e para isso, um padrão foi projetado no objeto. Com a projeção, é possível calcular as intersecções geradas com a borda e que acaba se transformando em uma *ray plane intersection*, que é facilmente computadorizada (LI; STRAUB; PRAUTZSCH, 2005, p. 4).

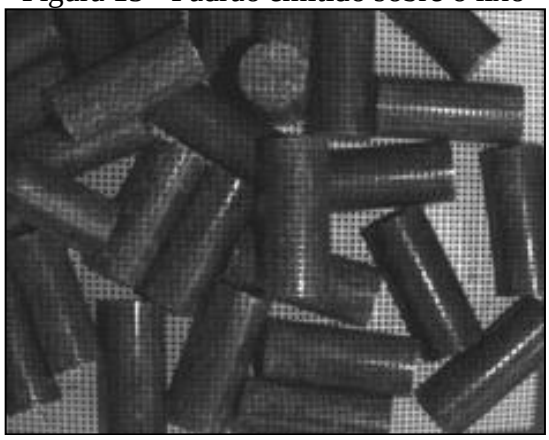
Os resultados obtidos por Li, Straub e Prautzsch (2005, p. 5) demonstram que houve uma grande melhora na precisão dos *subpixeis*, assim como melhorias no algoritmo que foi para o pós-processamento utilizando a programação dinâmica. A única desvantagem citada, é que há parâmetros que precisam ser configurados manualmente e há uma diminuição nas bordas detectadas. Segundo Li, Straub e Prautzsch (2005, p. 6), para se obter uma melhora nos resultados obtidos é sugerido que a interação com humanos no processo seja diminuída, principalmente na parte da calibração do projetor e da câmera, além disso, calcular automaticamente o limite de corte de cores baseando-se nas estatísticas de cores e intensidades de luz presentes na cena.

2.3.2 3D scanning of object surfaces using structured light and single camera image

Mohan et al. (2011, p. 1) apresentam um sistema para a detecção de objetos em uma cena com a finalidade de separação de lixo, tendo em vista que o uso de robôs nas fábricas, com essa finalidade, reduzirá custos para as empresas. Para detecção de relevos, os autores utilizam a técnica ativa baseada em luz estruturada, que utiliza LEDs para emitir um padrão na cena e uma câmera para analisar a mudança desse padrão.

Tendo em vista que em um cenário de separação de lixo a iluminação ambiente não é constante e existem diversos objetos com propriedades de reflexão da luz diferentes, essas características interferem no padrão que é projetado na cena, causando problemas. Os padrões baseados em cores têm problemas devido à grande incerteza do grau de reflexão dos itens, já os baseados em *time-multiplexed* acabam sendo muito lentos e necessitam de várias imagens. Porém, segundo o autor, o padrão baseado em palavras é o mais interessante para a solução. Após essa análise, o trabalho propôs a projetar um padrão de grades sendo projetados por emissores de luz LED, que pode ser visto na Figura 15 (MOHAN et al., 2011, p. 1).

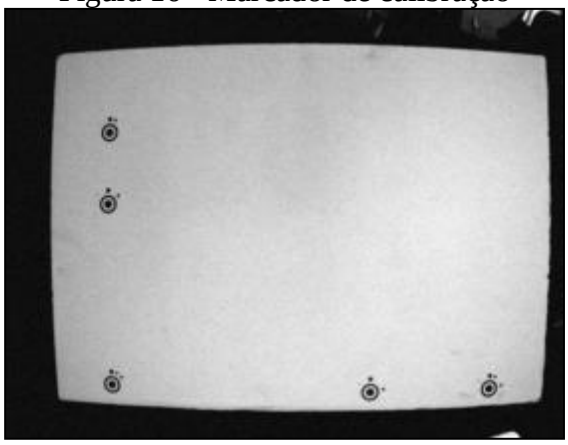
Figura 15 - Padrão emitido sobre o lixo



Fonte: Mohan et al. (2011).

Antes de emitir o padrão e começar a realizar o processo de extração das informações geométricas, o trabalho demonstra a realização de um processo fundamental, o de calibração do aparelho. Para realizar essa calibração, os autores utilizaram um padrão impresso em uma folha e colocaram diante da câmera para que ela identificasse aqueles pontos e configurasse os parâmetros necessários para a medição correta das formas geométricas. O padrão proposto pode ser visto na Figura 16.

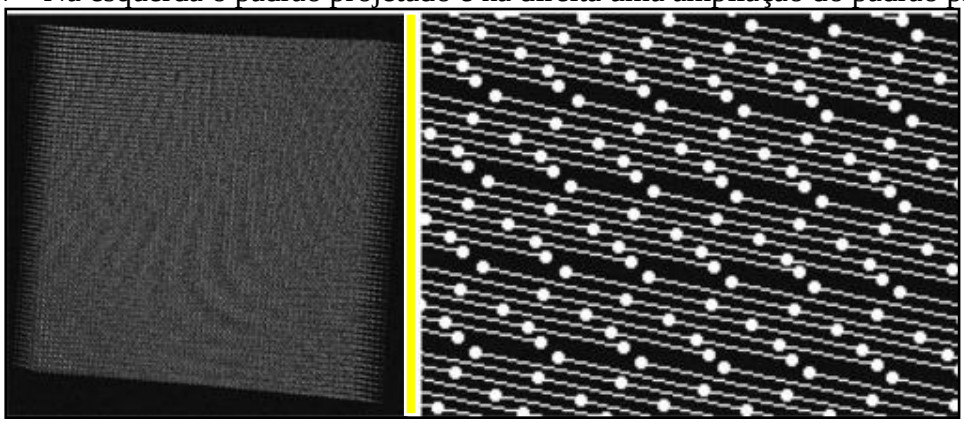
Figura 16 - Marcador de calibração



Fonte: Mohan et al. (2011).

Após a calibração, determinar o padrão que será projetado sobre a cena é ainda mais importante, tendo em vista que são extraídas informações de deformação em cima deste padrão. Neste trabalho, como a distância z (distância entre câmera e cena) já foi medida pela inserção do marcador na cena, para detectar a profundidade de um ponto, utiliza-se a técnica da interpolação, gerando assim uma coordenada (MOHAN et al., 2011, p. 4). O padrão projetado pelo trabalho pode ser visto na Figura 17, que mostra o padrão sobre toda a cena na esquerda, e na direita uma ampliação do padrão que foi projetado.

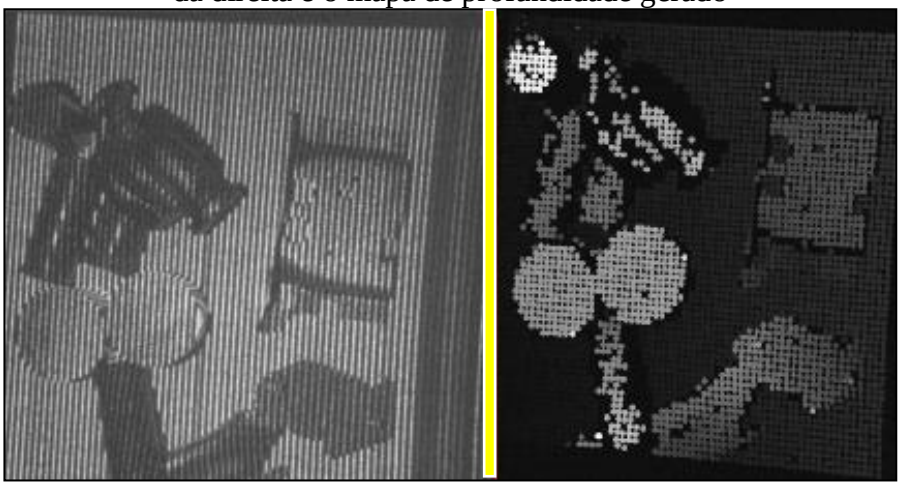
Figura 17 - Na esquerda o padrão projetado e na direita uma ampliação do padrão projetado



Fonte: Mohan et al. (2011).

Continuando o processo para a detecção do relevo, é calculado a profundidade de cada ponto onde o padrão foi projetado. A partir disso um mapa de profundidade é gerado. Pode-se ver na esquerda da Figura 18 a cena recebendo o padrão escolhido e a direita o mapa de altura que foi gerado após calcular a profundidade de cada ponto na cena.

Figura 18 - A imagem da esquerda mostra a cena com o padrão sendo projetado. Na imagem da direita é o mapa de profundidade gerado



Fonte: adaptado de Mohan et al. (2011).

O trabalho obteve uma taxa de erro menor que 5% ao detectar incorretamente a profundidade de um ponto, uma vez que houve problemas em relação ao padrão projetado que não foi o ideal. Na etapa de calibração também obtiveram problemas, onde para conseguir calibrar corretamente, foi necessário tirar duas fotos do marcador (MOHAN et al., 2011, p. 5). Para melhorar os resultados obtidos, espera-se melhorar o alcance da profundidade e a resolução combinando a técnica ativa com luz estruturada contendo um padrão de projeção melhor (MOHAN et al., 2011, p. 6).

2.3.3 Implementation of a low cost structured light scanner

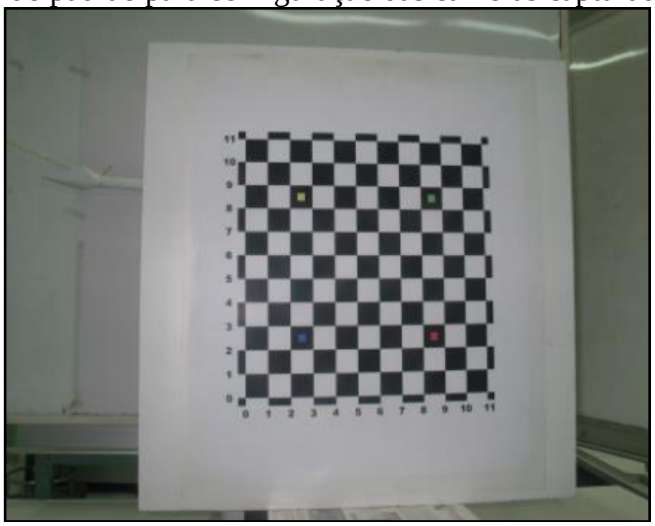
O trabalho de Pashaei e Mousavi (2013) apresenta o desenvolvimento de um *scanner* de baixo custo utilizando luz estruturada com o código de Gray (*Gray code*) e o deslocamento de várias linhas (*multi-line shift*). O equipamento é constituído de duas câmeras e um projetor que compõem o sistema de visão estéreo e pode ser observado na Figura 19.

Figura 19 - Dispositivo de captura e projeção

Fonte: Pashaei e Mousavi (2013, p. 5).

Antes do sistema iniciar, a configuração dos parâmetros intrínsecos e extrínsecos da câmera é realizada e, para isso, um padrão similar a um tabuleiro de xadrez é colocado em frente às câmeras. Inicialmente, as duas câmeras tiram fotos do padrão (Figura 20) de dois campos de visão distintos, tendo em vista que esse padrão já é conhecido. Os quadrados brancos e pretos são medidos utilizando a biblioteca Australis, resultando nos parâmetros intrínsecos para cada câmera (PASHAEI; MOUSAVI, 2013, p. 2).

Figura 20 - Foto do padrão para configuração das câmeras capturada câmera direita



Fonte: Pashaei e Mousavi (2013, p. 2).

Para fazer com que os parâmetros extrínsecos da matriz fundamental sejam obtidos, ambas as câmeras tiram uma foto, fazendo com que seja possível obter uma coordenada 3D precisa para cada câmera. Essas coordenadas são utilizadas para obter a matriz fundamental, que pode ser vista na Figura 21.

Figura 21 - Matriz fundamental obtida

$$E' = M_1 K_b M_2^T$$

Fonte: Pashaei e Mousavi (2013, p.3).

Feita a calibração, inicia-se o processo de extração das tiras de luz que estão sendo projetadas sobre o objeto. Segundo Pashaei e Mousavi (2013, p. 4), abordagens comuns para detecção dessas tiras se baseiam considerando o valor de Gray da imagem, por exemplo, o valor que contém o brilho mais alto é considerado o centro da tira. Porém o *scanner* desenvolvido utiliza a ideia de encontrar os cumes e desfiladeiros para encontrar o centro da tira através de uma equação resultante (Figura 22) do processo de derivação dos cumes e desfiladeiros encontrados.

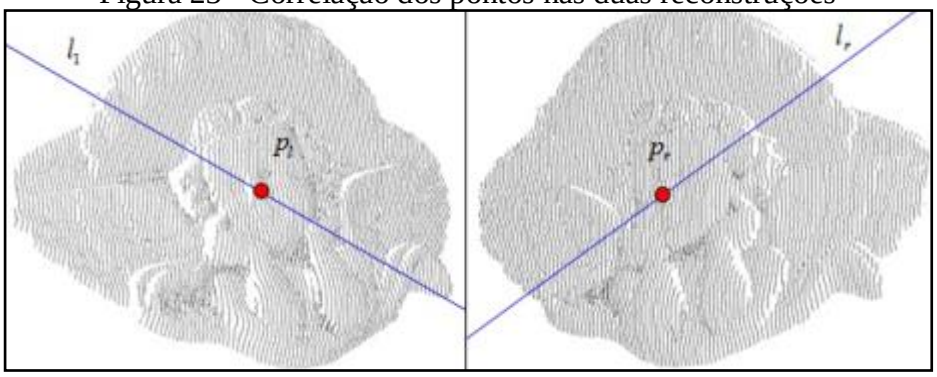
Figura 22 - Equação final utilizada para encontrar o centro da tira

$$(p_x, p_y) = (tm_x + x_0, tm_y + y_0)$$

Fonte: Pashaei e Mousavi (2013, p. 4).

Após detectados os pontos centrais das tiras, o padrão projetado sofre um deslocamento lateral para a direita sete vezes. Ao final, dezoito padrões são projetados sobre a imagem: sete deles sendo resultados do deslocamento, dois de *oversampling* do último padrão e mais dois para a normalização de cinza, obtendo um resultado robusto. Além disso, realiza-se a correlação dos pontos entre as reconstruções realizadas pela câmera da esquerda e pela da direita (Figura 23) a fim de se normalizar os dados obtidos e gerar uma única saída (PASHAEI; MOUSAVI, 2013, p. 5).

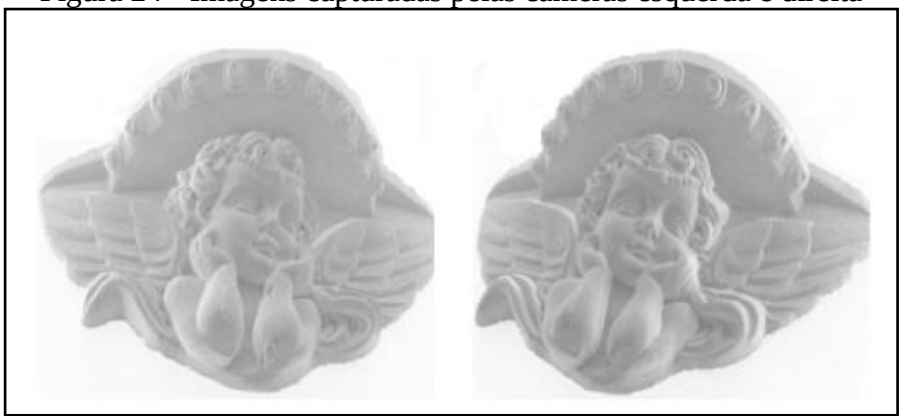
Figura 23 - Correlação dos pontos nas duas reconstruções



Fonte: Pashaei e Mousavi (2013, p. 5).

Baseados na integração de fotogrametria de baixo alcance juntamente com a combinação das técnicas de código de Gray com deslocamento das linhas, Pashei e Mousavi (2013, p. 6) obtiveram bons resultados. Desta forma conseguiram gerar nuvens de pontos densas e complexas baseadas em obras de artes complexas, obtendo taxas de erro de até 0,1%. Tal resultado pode ser visto nas Figuras Figura 24 e Figura 25.

Figura 24 - Imagens capturadas pelas câmeras esquerda e direita



Fonte: Pashaei e Mousavi (2013, p. 6).

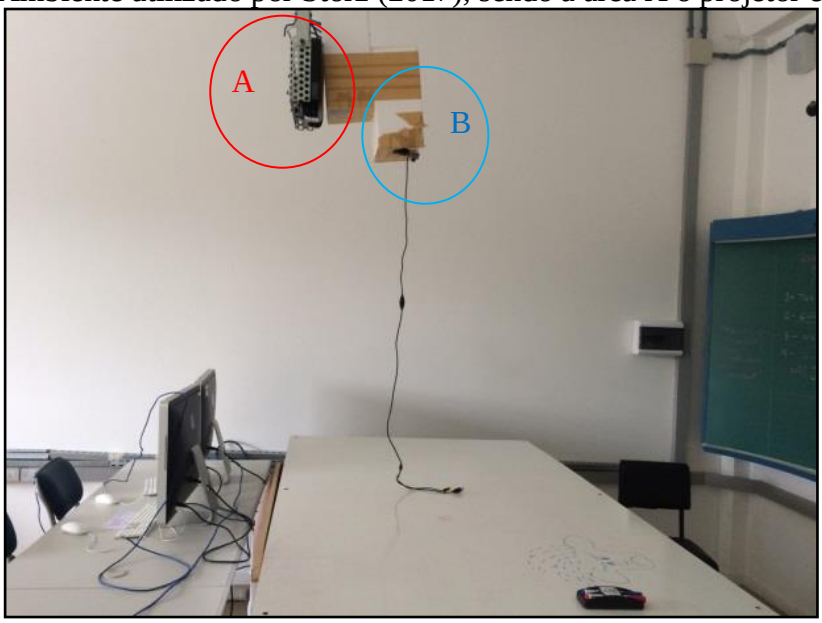
Figura 25 - Resultado da reconstrução 3D

Fonte: Pashaei e Mousavi (2013, p. 6).

2.3.4 Biblioteca para detecção de relevos

Inicialmente desenvolvida por Storz (2017), a biblioteca para detecção de relevos tem como principal objetivo a reconstrução de relevos em tempo real utilizando a técnica de reconstrução por luz estruturada através da emissão de um padrão conhecido, baseado na sequência de De Brujin, utilizando um projetor e uma câmera (Figura 26). A biblioteca utiliza como tecnologias: a linguagem de programação C++ 11 e a biblioteca de visão computacional OpenCV para realizar as operações sobre as imagens (STORZ, 2017, p. 43).

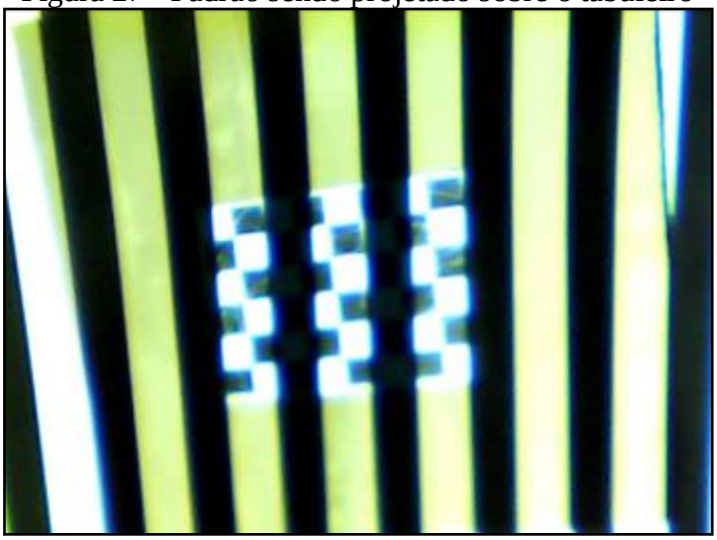
Figura 26 - Ambiente utilizado por Storz (2017), sendo a área A o projetor e B a câmera



Fonte: elaborado pelo autor.

O trabalho de Storz (2017) realiza três partes principais do processo de reconstrução: calibração, reconstrução e exibição dos resultados. O processo de calibração realizado por Storz (2017, p. 46) baseia-se no algoritmo de Moreno e Taubin (2012), removendo todas as dependências que não seriam utilizadas pela biblioteca, como por exemplo a interface gráfica. Nessa etapa, inicialmente projeta-se padrões sobre um tabuleiro de xadrez a fim de realizar a associação de uma coluna do padrão projetado com uma coluna do tabuleiro de xadrez, conforme pode ser visto na Figura 27. Após isso Storz (2017, p. 47) afirma que é realizada a detecção das bordas do tabuleiro a fim de encontrar tais pontos obtidos pela câmera em coordenadas do projetor. Comparando os resultados da calibração obtida com o trabalho de Moreno e Taubin (2012), Storz (2017, p. 59) verificou que as taxas de erros são semelhantes acrescentando como vantagem que o processo de calibração implementado não dependia de uma interface gráfica.

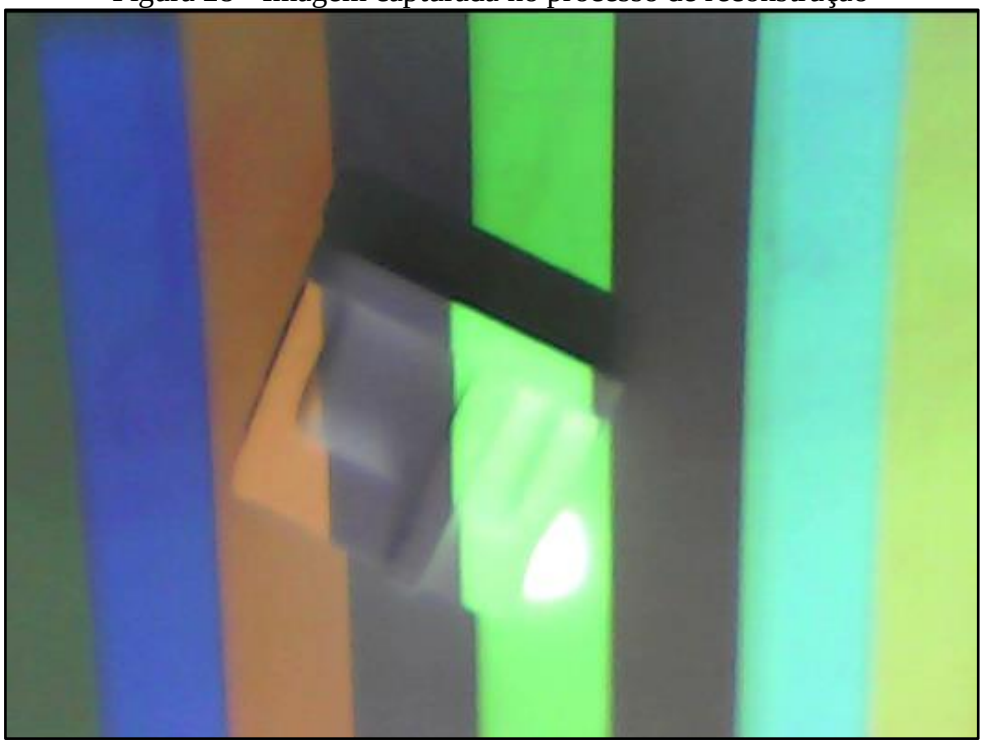
Figura 27 - Padrão sendo projetado sobre o tabuleiro



Fonte: Storz (2017, p. 46).

No processo de reconstrução por luz estruturada, uma das principais características do trabalho é a projeção de apenas um padrão e sendo assim, a biblioteca precisa apenas de uma imagem para realizar o processo de reconstrução. O padrão é baseado na sequência de De Brujin, e uma captura desse processo pode ser visto na Figura 28.

Figura 28 - Imagem capturada no processo de reconstrução

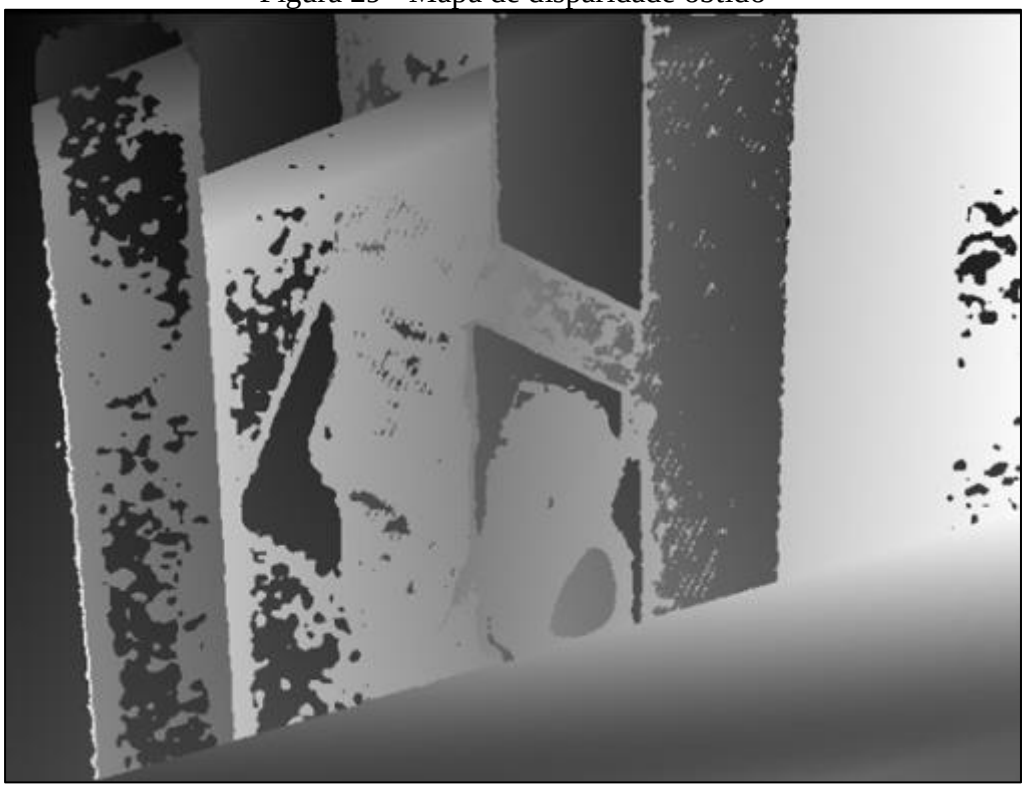


Fonte: Storz (2017, p. 60).

Para a demonstração da reconstrução, o trabalho de Storz (2017) utiliza apenas a apresentação através de um mapa de disparidade. Segundo Storz (2017, p. 63) a exibição da reconstrução realizada através do mapa de disparidade contém algum erro não identificado que

faz com que não haja altura no relevo detectado (Figura 29). Tal problema pode estar relacionado com a correlação dos *pixels* no algoritmo de reconstrução.

Figura 29 - Mapa de disparidade obtido



Fonte: Storz (2017, p. 63).

Em relação ao desempenho da biblioteca, Storz (2017, p. 65) afirma que para realizar o processo de reconstrução o tempo aproximado é de uma hora, impossibilitando a proposta original de que a biblioteca deveria realizar a reconstrução em tempo real. Para tentar minimizar este problema, Storz (2017, p. 66) sugere como principais pontos a se melhorar: a implementação das rotinas de reconstrução utilizando GPGPU e correção do erro que está gerando um mapa de disparidade errado, o que, segundo ele, pode fazer com que a velocidade e assertividade da biblioteca sejam melhoradas.

3 DESENVOLVIMENTO

Neste capítulo serão apresentadas as etapas do desenvolvimento da biblioteca. Na seção 3.1 são apresentados os requisitos da biblioteca. A seção 3.2 descreve as especificações através da utilização de diagramas. A seção 3.3 detalha a implementação. Por fim, a seção 3.4 demonstra os resultados que foram obtidos.

3.1 REQUISITOS

Os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) da biblioteca são:

a) utilizar uma câmera para capturar imagens de uma cena sobre a qual será realizado o reconhecimento do relevo (RF);
b) detectar a profundidade do relevo pelo método de triangularização ativa (RF);
c) gerar um *dataset* do relevo obtido com o objetivo de eser importado por outros programas (RF);
d) obter um tempo de reconstrução para que a biblioteca possa ser utilizada em um cenário de reconstrução em tempo real (RF);
e) utilizar a linguagem de programação C++ (RNF);
f) ser desenvolvida para rodar nos sistemas operacionais Windows, Linux e macOS (RNF);
g) utilizar o OpenCV para realizar as operações com as imagens (RNF);
h) remover a dependência do VisualStudio para compilar e executar a biblioteca (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção encontra-se uma descrição da estrutura da biblioteca, como o diagrama de classes e o diagrama de sequência da parte de reconstrução.

3.2.1 Diagrama de classes

Na Figura 30 é apresentado o diagrama de classes da biblioteca desenvolvida, podendo ser visto como as classes estão estruturadas e ligadas entre si. Nessa seção serão descritas as principais classes e suas relações.

A classe `SceneManager` é responsável por ser o ponto de entrada entre um programa e a biblioteca. Nessa classe é possível que tanto a biblioteca tire as fotos e projete os padrões como o programa que está utilizando a biblioteca realize isso e envie apenas as fotos capturadas.

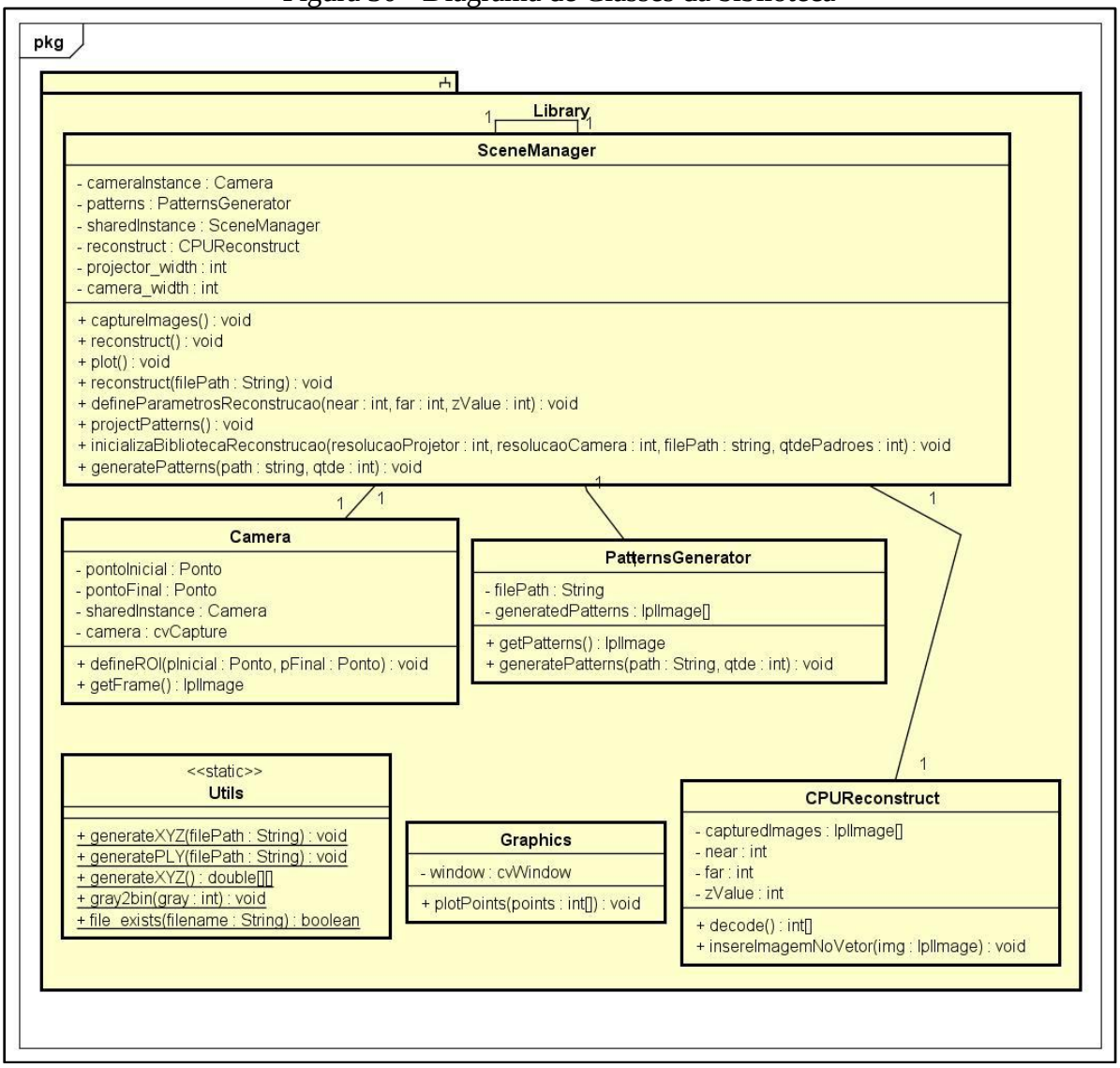
A classe `PatternsGenerator` é utilizada para gerar os padrões que serão projetados baseando-se na quantidade passada pelo parâmetro do método `generatePatterns`. A função da classe `Graphics` é de apenas auxiliar caso o programa que está utilizando a biblioteca ainda não tenha implementado a parte gráfica para visualizar um mapa de disparidade dos pontos que foram obtidos pela reconstrução.

Caso o programa que utilizará a biblioteca não esteja tirando as fotos necessárias, a biblioteca disponibiliza a classe `Camera` para projetar e capturar a distorção dos padrões. Nela existe o atributo `pontoInicial` e `pontoFinal` que servem para capturar apenas uma região de interesse, evitando capturar pontos que não estejam na área de projeção.

A classe `Utils` é utilizada como suporte, gerando arquivos da nuvem de pontos ou retornando a nuvem de pontos para quem chamar o método. Existe também o método `gray2bin` que é responsável por decodificar um código binário para um número representado por um inteiro.

A classe principal da biblioteca é a `CPUReconstruct`, que é responsável pelo algoritmo base de reconstrução através da luz estruturada. Tendo as imagens capturadas e os parâmetros o método `decode` pode ser chamado para que seja possível obter uma matriz com os pontos reconstruídos. Os atributos `near` e `far` servem para que no momento da reconstrução, caso algum ponto seja maior ou menor do que esses valores, ele seja descartado.

Figura 30 - Diagrama de Classes da biblioteca

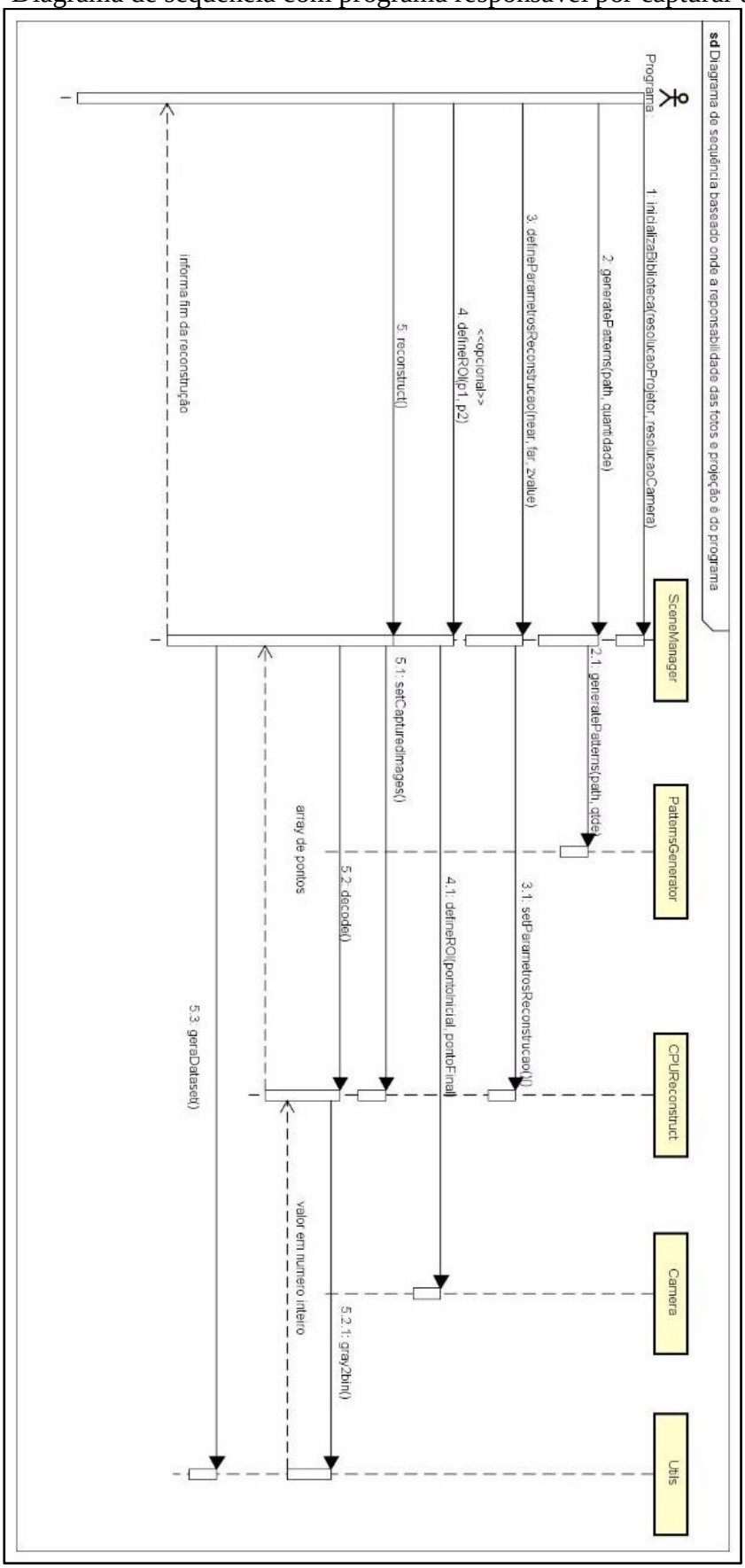


Fonte: elaborado pelo autor.

3.2.2 Diagrama de sequência

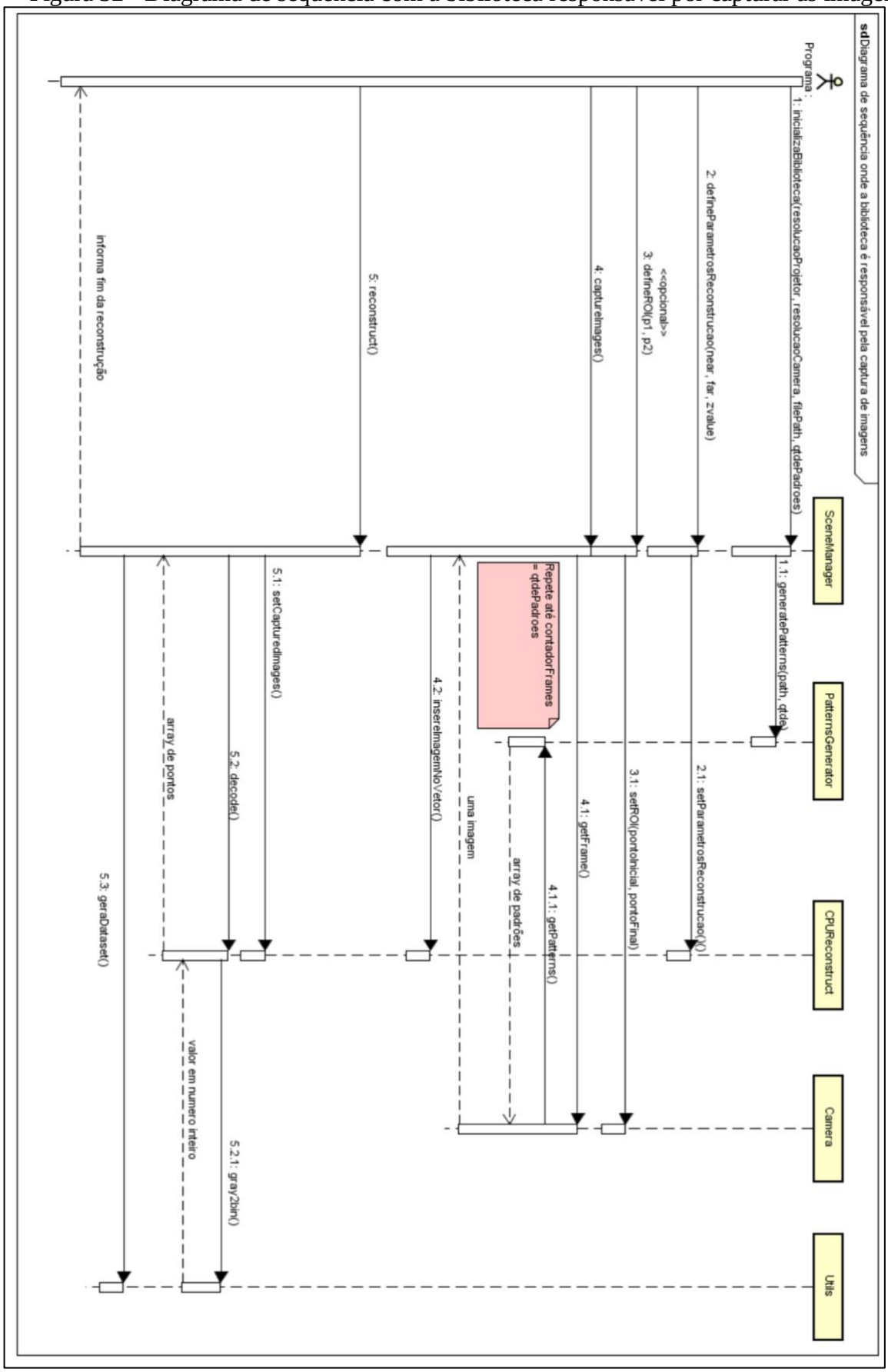
Tendo em vista que tanto o programa como a biblioteca podem ser os responsáveis por projetar e capturar as imagens, existem dois fluxos que podem acontecer para que o processo de reconstrução aconteça. Na Figura 31, apresenta-se o processo de reconstrução onde o programa que importa a biblioteca é responsável por projetar e capturar as distorções dos padrões e somente envia essas imagens à biblioteca. Já na Figura 32, a biblioteca é totalmente responsável pela projeção dos padrões, captura das distorções e pela reconstrução.

Figura 31 - Diagrama de sequência com programa responsável por capturar as imagens



Fonte: elaborado pelo autor.

Figura 32 - Diagrama de sequência com a biblioteca responsável por capturar as imagens



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

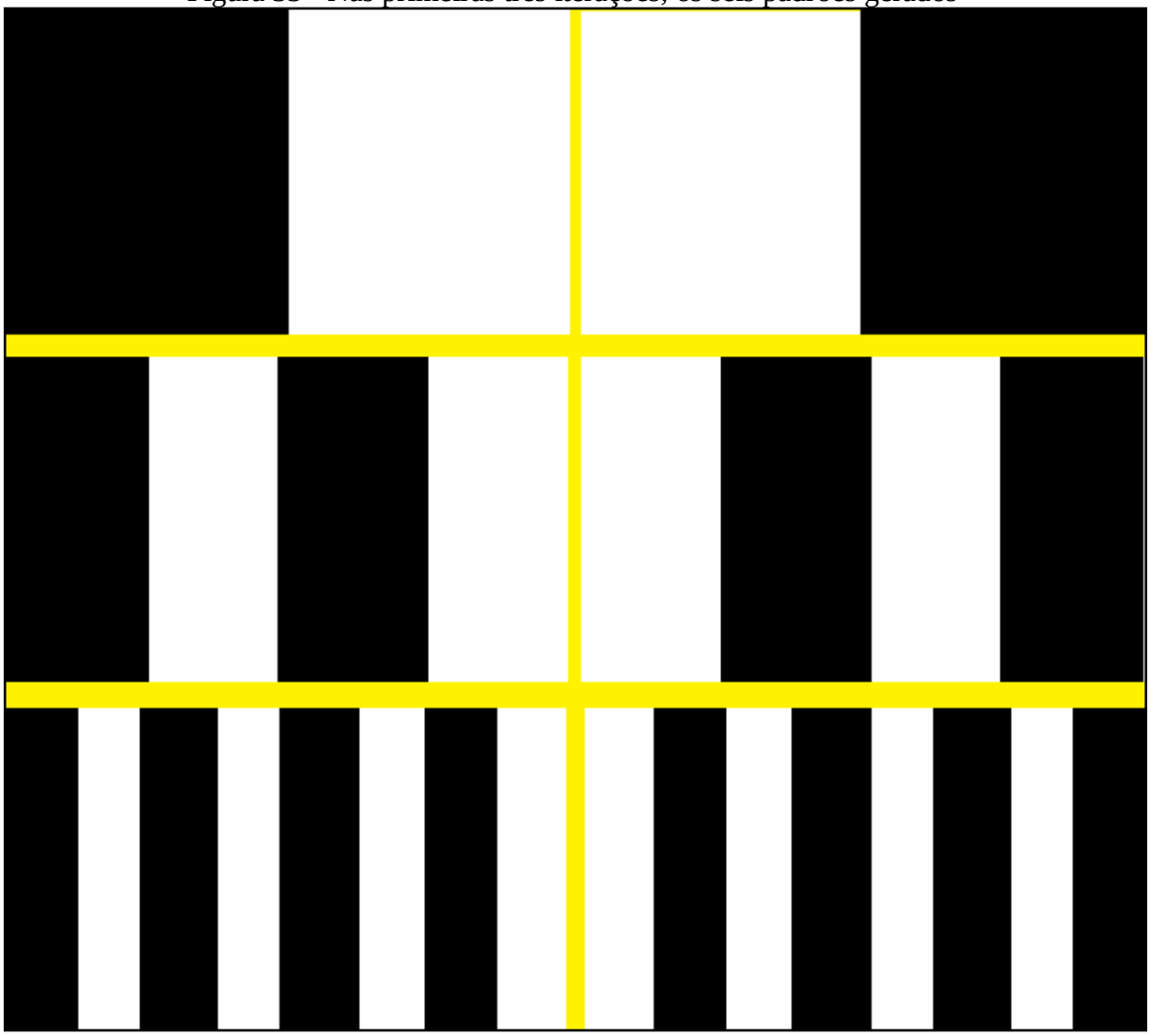
No desenvolvimento da biblioteca foi utilizada a linguagem de programação C++11 e o ambiente de desenvolvimento Visual Studio Community 2015 Update 3. Para compatibilidade entre plataformas um arquivo do tipo MAKEFILE foi gerado contendo a rotina de compilação da biblioteca, tirando essa responsabilidade do Visual Studio.

A seguir, descrevem-se os principais algoritmos e fórmulas utilizados como base para o desenvolvimento da biblioteca, assim como códigos que foram adaptados de seus autores para serem utilizados na aplicação.

3.3.1.1 Algoritmo de geração dos padrões

Antes de iniciar o processo de reconstrução, é preciso definir os padrões que serão projetados na cena. Para a geração dos padrões binários, o algoritmo foi baseado no proposto por Herakleous e Poullis (2016). O Quadro 2 demonstra o algoritmo criado. Na linha 1, para a geração dos padrões é necessário informar a quantidade de padrões e o tamanho do padrão. Nas linhas de 9 até 11, o algoritmo cria uma imagem totalmente preta e calcula a quantidade e o tamanho de cada coluna branca que será inserida nessa imagem. As linhas de 13 até 19 realizam a inserção das colunas brancas na imagem. Por fim, das linhas 20 até 25, ocorre a inversão do padrão e a inserção dos dois padrões no vetor de padrões. Esse algoritmo é repetido de acordo com a quantidade de padrões informadas pelo método. Na Figura 33 é demonstrado visualmente as primeiras três iterações do algoritmo e quais são os padrões resultantes.

Figura 33 - Nas primeiras três iterações, os seis padrões gerados



Fonte: elaborado pelo autor.

Quadro 2 - Algoritmo para geração dos padrões

```
1   IplImage generatePatterns(int numberOfPatterns, int patternWidth, int
2   patternHeight){
3
4   //cria na memoria um vetor de imagens que serao os padrões
5       IplImage patterns = IplImage[numberOfPatterns*2];
6
7   //laço para criar a quantidade dos padrões
8       for (int i = 1; i < numberOfPatterns+1; i++) {
9           Mat pattern(patternHeight, patternWidth, CV_8UC3, Scalar(255,
    255, 255));
10          int numberOfStripes = pow(2, i); //determina a quantidade de
    padrões
11          int stripeWidth = patternWidth / pow(2, i); //calcula o
    tamanho de cada coluna
12  //pra cada coluna que terá
13          for (int j = 0; j < numberOfStripes; j++) {
14  //calcula a margem da esquerda onde a coluna irá começar
15              int xMargin = j * stripeWidth;
16              cv::Rect rect(xMargin * 2, 0, stripeWidth,
    patternHeight);
17              cv::rectangle(pattern, rect, cv::Scalar(0, 0, 0),
18  CV_FILLED);
19          }
20  // inverte as cores do padrão
21          Mat invertedPattern = ~pattern;
22  //coloca no vetor de padrões
23          patterns[i-1] = pattern;
24          patterns[i] = invertedPattern;
25      }
26      return patterns;
27  }
```

Fonte: elaborado pelo autor.

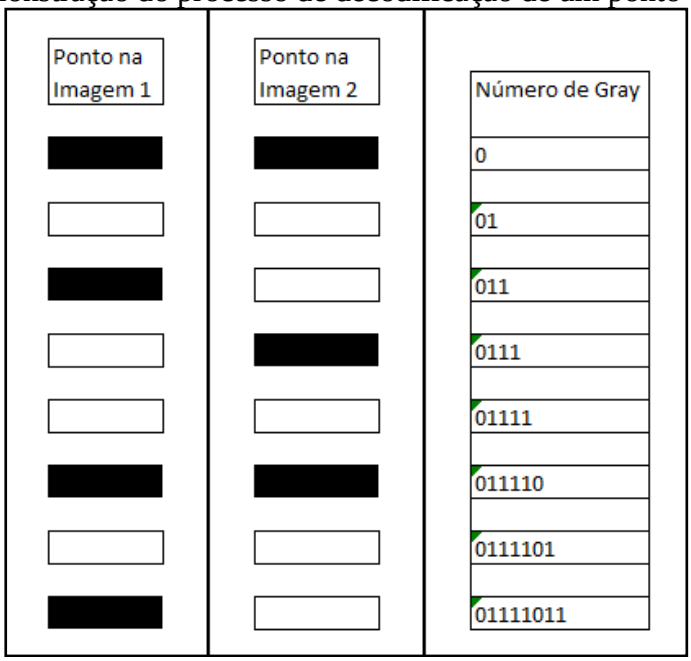
3.3.1.2 Algoritmo de reconstrução

Sabendo os padrões que foram projetados sobre o objeto, já é possível realizar a reconstrução 3D do objeto. O algoritmo de reconstrução baseia-se na decodificação da deformação dos padrões binários projetados sobre o objeto, descobrindo se em um determinado ponto era para ser branco ou preto. De acordo com Herakleous e Poullis (2016), ciar um padrão e inverter suas cores é um método fácil e eficiente para descobrir a intensidade da cor daquele ponto, a fim de saber se o objeto deformou o padrão ou não.

O Quadro 3 apresenta o processo de reconstrução. Na linha 2, um vetor é inicializado na memória e tem o seu tamanho baseado no tamanho da imagem e no tamanho de uma variável inteira. Após isso o algoritmo começa a percorrer o vetor de imagens capturadas obtendo a imagem com padrão normal e seu padrão invertido. Feito isso, das linhas 8 até 27 o algoritmo percorre cada ponto para realizar sua decodificação. A Figura 34 demonstra o processo de decodificação de um ponto visualmente. Assumindo que um ponto branco seja equivalente ao número 1 e o número preto seja 0, para cada ponto nas duas imagens é realizada uma operação

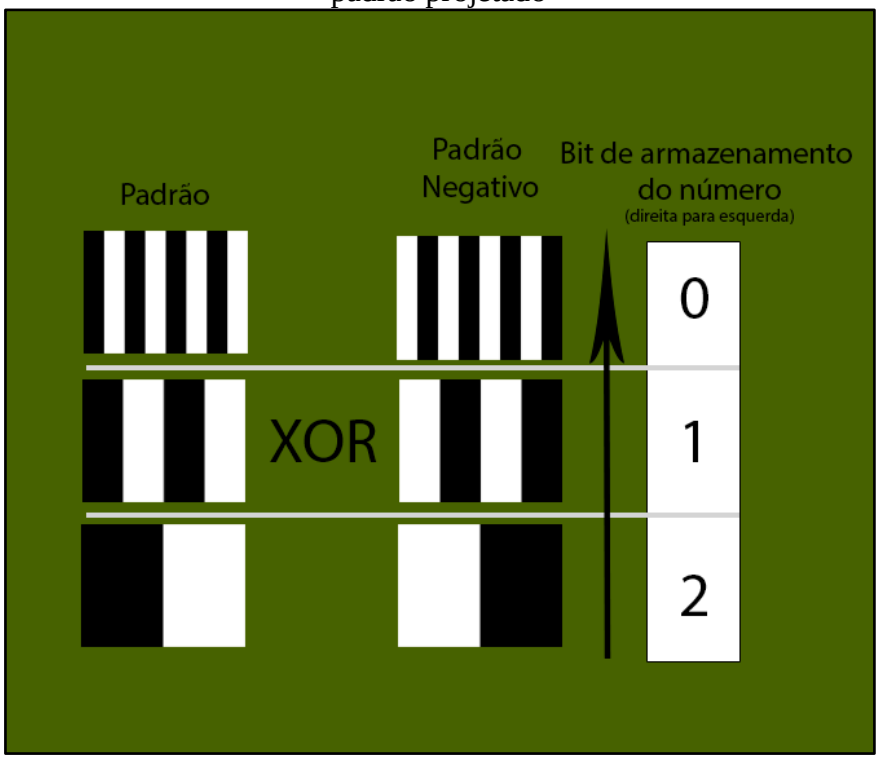
lógica OR, ou seja, se o branco for detectado em uma das imagens, aquele ponto sofreu uma distorção. Além disso, o *shift* realizado na linha 18 faz com que os primeiros padrões são os que contém um peso mais na hora de transformar o número binário gerado em um número decimal (Figura 35). Ao final desse processo, das linhas 28 até 32, o algoritmo realiza a conversão do binário para um número decimal.

Figura 34 - Demonstração do processo de decodificação de um ponto em oito padrões



Fonte: elaborado pelo autor.

Figura 35 – Representação do armazenamento do número binário em relação ao peso do padrão projetado



Fonte: elaborado pelo autor.

Quadro 3 - Algoritmo de reconstrução

```
1   //cria o vetor que irá armazenar o valor dos pixeis
2   int *reconstructVet = (int*)calloc(imgsCapturadas[0]->imageSize,
    sizeof(int));
3   for(int i = 0; i < NUM_IMAGES; i += 2){ // a cada dois padrões
4   //pega as duas imagens capturadas, 1 de cada padrão projetado
5       uchar* data1 = (uchar*)imgsCapturadas[i]->imageData;
6       uchar* data0 = (uchar*) imgsCapturadas [i+1]->imageData;
7   //para cada coluna e linha da imagem
8       for(int y = 0; y < imgsCapturadas [i]->height; y++){
9           for(int x = 0; x < imgsCapturadas [i]->width; x++) {
10              int u = y * imgsCapturadas [i]->widthStep + x *
    imgsCapturadas [i]->nChannels;
11  //se pixel ignorado, continua ignorando
12              if(reconstructVet [u] == -1){
13                  continue;
14              }
15              int p1 = data1[u] + data1[u+1] + data1[u+2];
16              int p0 = data0[u] + data0[u+1] + data0[u+2];
17  //desloca os numeros para esquerda a fim de que o novo 'bit' seja
    colocado logo abaixo
18              reconstructVet [u] = reconstructVet [u] << 1;
19              if(p1 – p0 > WHITE_THRESHOLD){
20                  reconstructVet [u] |= 1; //tem um pixel branco
21              } else if(p1 – p0 < BLACK_THRESHOLD){
22                  reconstructVet [u] |= 0; //tem um pixel preto
23              } else {
24                  reconstructVet [u] = -1; //ignora
25              }
26          }
27      }
28  // Converte os números para números inteiros
29  for(i = 0; i < imgsCapturadas [0]->imageSize; i++){
30      reconstructVet[i] = gray2bin(reconstructVet[i]);
31  }
32  }
```

Fonte: adaptado de Arroyo et al. (2011).

3.3.1.3 Algoritmo de captura das imagens

Caso o programa que esteja utilizando a biblioteca opte por deixar a captura de imagens a cargo da biblioteca, o algoritmo que irá projetar o padrão gerado e irá capturar a imagem será o descrito no Quadro 4. Inicialmente o algoritmo irá projetar e capturar a quantidade de fotos no parâmetro `NUM_IMAGES`. A partir disso, o algoritmo exibe um padrão que foi gerado anteriormente e espera um tempo de 125ms. O tempo de espera entre a projeção do padrão e a ação de capturar a imagem da câmera é essencial nesse algoritmo, sendo que o mesmo é variável de acordo com o projetor e câmera utilizada. Caso o número seja muito baixo, o projetor estará terminando de projetar o padrão e a câmera já estará capturando o padrão, causando uma anomalia no processo de reconstrução. Nas linhas 7 até 11, o algoritmo encerra esse procedimento de projeção e captura e armazena a imagem capturada.

Quadro 4 - Algoritmo de captura de imagens

```
1   for(int i=0;i<NUM_IMAGES;i++){
2       cvShowImage("mainWin", padroes[i] ); //projeta um padrão sobre a
    cena
3       cvWaitKey(125); //tempo de espera
4       cvGrabFrame(capture); //pega o ultimo frame do buffer
5       imgsCapturadas [NUM_IMAGES + 1] = cvRetrieveFrame(capture);
6   //copia a referencia da imagem e salva no vetor de imagens
    capturadas
7       imgsCapturadas [i] = cvCloneImage(imgsCapturadas [NUM_IMAGES +
    1]);
8   //salva a imagem capturada no filePath
9       sprintf(s,"resultados/%d.png",i+1);
10      cvSaveImage(s, imgsCapturadas [NUM_IMAGES + 1]);
11  }
```

Fonte: elaborado pelo autor.

No final do processo de reconstrução cada ponto da matriz estará codificado, sendo assim é necessário realizar a conversão deste para um número inteiro. Para um melhor desempenho, adaptou-se o algoritmo de Avins (2000), que pode ser visto no Quadro 5, tendo em vista que é um algoritmo linear (`O(N-1)`), pois a realização de um algoritmo quadrático (`O(2^N)`) nessa conversão poderia ser custosa ao processo de reconstrução. Se usarmos como base `N=8` teríamos um total de 2.150.400 instruções no algoritmo linear contra 78.643.200 de instruções do algoritmo quadrático. Caso um ponto não tenha sido reconhecido corretamente, ele terá o valor de -1 e não será convertido nessa etapa.

Quadro 5 - Algoritmo de conversão

```
1  int gray2bin(int gray){
2        unsigned int temp = gray ^ (gray>>8);
3        if(gray == -1) return -1; // adaptação para ignorar caso
   ponto esteja com erro
4        temp ^= (temp>>4);
5        temp ^= (temp>>2);
6        temp ^= (temp>>1);
7        return temp;
8  }
```

Fonte: adaptado de Avins (2000).

3.3.1.4 Algoritmo de persistência

Para importar a nuvem de pontos geradas em outros programas como o Unity ou o MeshLab, criou-se um arquivo no padrão `XYZ` contendo em cada linha uma coordenada `X`, `Y` e `Z` para cada ponto que esteja dentro dos parâmetros `near` e `far`. O Quadro 7 apresenta um trecho de um arquivo que foi gerado e como é o seu formato. Assim como nas demais etapas, caso um valor esteja -1, ele será ignorado. O Quadro 6 apresenta o algoritmo para gerar os pontos. Nas linhas iniciais do algoritmo (1 até 8) um arquivo com o nome `nuvem.xyz` é criado. Da linha 10 até a 24, o processo de salvar cada ponto nesse arquivo é realizado. Conforme a linha 16, caso a disparidade calculada seja 0 o ponto é ignorado para evitar que haja um erro de divisão por 0.

A variável `zScale` utilizada na linha 18, é um dos parâmetros principais de calibração da biblioteca (juntamente com os *thresholds* de branco e preto), fazendo com que a calibração através de tabuleiros do xadrez fosse removida. Esse parâmetro está baseado no ângulo entre o projetor e a câmera, fazendo com que quanto maior o ângulo, menor o parâmetro e quanto menor o ângulo, maior seja o valor desse parâmetro.

Quadro 6 - Algoritmo para gerar um arquivo XYZ com a nuvem de pontos

```
1   //cria o aquivo chamado nuvem.xyz que irá armazenar a nuvem de pontos
2   FILE *f = fopen(s,"w");
3   if(f == NULL) {
4       FILE *f = fopen("nuvem.xyz","w");
5       if(f == NULL) {
6           return 1;
7       }
8   }
9   //percorre a imagem inteira para imprimir cada ponto já normalizado em
    inteiro
10  for(int y = 0; y < imgsCapturadas[0]->height; y++){
11      for(int x = 0 ;x < imgsCapturadas[0]->width; x++) {
12          int u = y * imgsCapturadas[0]->widthStep + x * imgsCapturadas
    [0]->nChannels;
13          if(reconstructVet [u] == -1) continue; //ignora o pixel q n deu
    pra reconstruir
14  //calcula a disparidade
15          float disparidade = (float) reconstructVet
    [u]/(projector_width) - (float)x/camera_width;
16          if(disparidade == 0.0) continue; // faz a validação pq se for 0
    dá crash na divisão por 0
17  //calcula a altura baseada no parametro Z
18          float z = (float)zScale/( disparidade);
19  //imprime no arquivo as coordenadas na sequencia X Y Z caso esteja
    entre o near e far
20          if (z>=far && z<= near) {
21              fprintf(f,"%d %d %f\n",x,y,z);
22          }
23      }
24  }
```

Fonte: adaptado de Arroyo et al. (2011).

Quadro 7 - Exemplo de arquivo XYZ gerado

```
0 0 449.999969
1 0 443.076904
2 0 436.363617
3 0 457.142822
4 0 450.000031
5 0 472.131165
6 0 464.516144
7 0 457.142883
8 0 480.000000
9 0 472.131165
10 0 464.516144
11 0 488.135620
…
588 479 -76.595749
590 479 -77.837845
592 479 -78.260872
593 479 -78.474113
595 479 -79.778397
```

Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

Esta seção é dedicada a mostrar os experimentos realizados com a biblioteca desenvolvida. Na seção 3.4.1 é apresentada a metodologia para a realização dos experimentos realizados e a seção 3.4.2 apresenta o experimento em si. O experimento é dividido em análise de desempenho e qualidade da reconstrução. Já a seção 3.4.3 apresenta uma análise geral sobre o trabalho desenvolvido e por fim na seção 3.4.4 é realizada uma comparação com trabalhos correlatos.

3.4.1 Metodologia

Os experimentos foram realizados em um computador com 8 GB de memória RAM, em um sistema operacional Windows 10 64-bits utilizando um processador AMD Ryzen 5 1500x. A câmera utilizada para obter as fotos do experimento é uma Nikon P600 que contém uma lente 18mm-300mm. Para projetar os padrões utilizou-se um projetor LG Minibeam que tem uma resolução de 1280x720 pontos com brilho de até 550 lúmen. Para medir os pontos de luz, utilizou-se um luxímetro digital portátil[1].

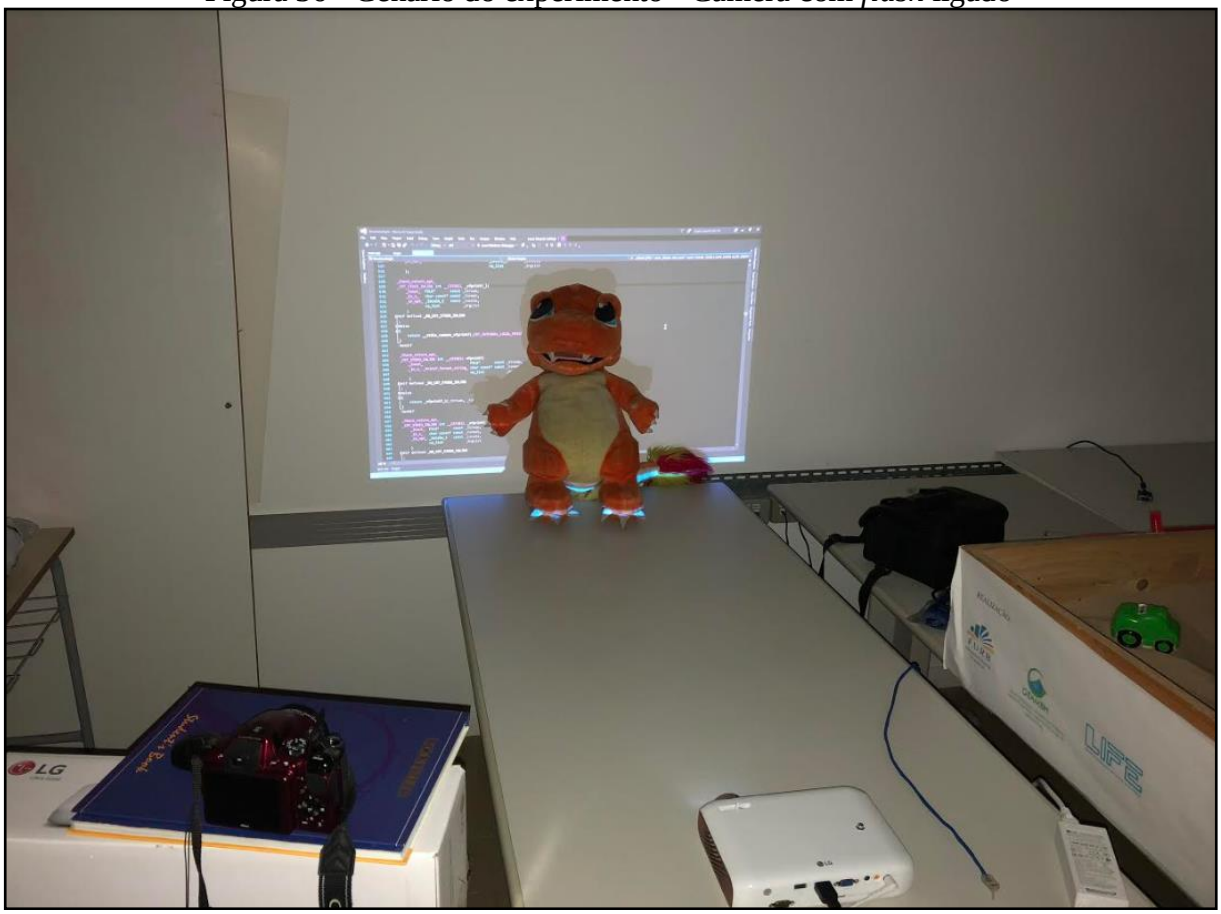
Para a utilização da biblioteca, foram realizados dois testes, um com um programa desenvolvido em C++ apenas chamando os métodos necessários e outro desenvolvido por Serodio (2018) que é feito em Unity, e quando importa a nuvem de pontos reconstrói um terreno e aplica um mapa hipsométrico.

3.4.2 Experimento

A fim de se realizar experimento e obter os resultados da reconstrução, configurou-se um cenário de teste que pode ser visto na Figura 36. Para os experimentos de reconstrução, realizou-se uma padronização nos parâmetros que podem ser vistos Quadro 8.

[1] Modelo semelhante: <https://www.americanas.com.br/produto/20522530/luximetro-digital-medidor-de-luminosidade-portatil-20000-lux>

Figura 36 - Cenário do experimento - Câmera com *flash* ligado



Fonte: elaborado pelo autor.

Quadro 8 - Parâmetros usados no experimento

Parâmetros	
Luminosidade	70 lux
Resolução da câmera	640x480
Zoom da lente	85mm
ISO da câmera	200
zScale	15
white_treshold	5
black_treshold	-5
projector_width	1280
camera_width	640
número de imagens obtidas (quantidade de padrões usados)	18

Fonte: elaborado pelo autor.

Com isso, iniciou-se o processo de projeção dos padrões e captura das imagens a fim de enviar essas fotos para a biblioteca analisar. Tendo em vista que não foi possível conectar a câmera diretamente com o computador, simulando uma *webcam*, as imagens foram capturadas manualmente e então passadas para o computador. Na Figura 37 é possível ver algumas imagens capturadas.

Para a verificação do resultado obtido pela reconstrução, importou-se a nuvem de pontos geradas pelo arquivo `XYZ` para dois programas, no Meshlab, que pode ser visto na Figura 38 e no programa desenvolvido na plataforma Unity por Serodio (2018) na Figura 39.

Figura 37 - Imagens capturadas pela câmera

(a) Décima imagem obtida

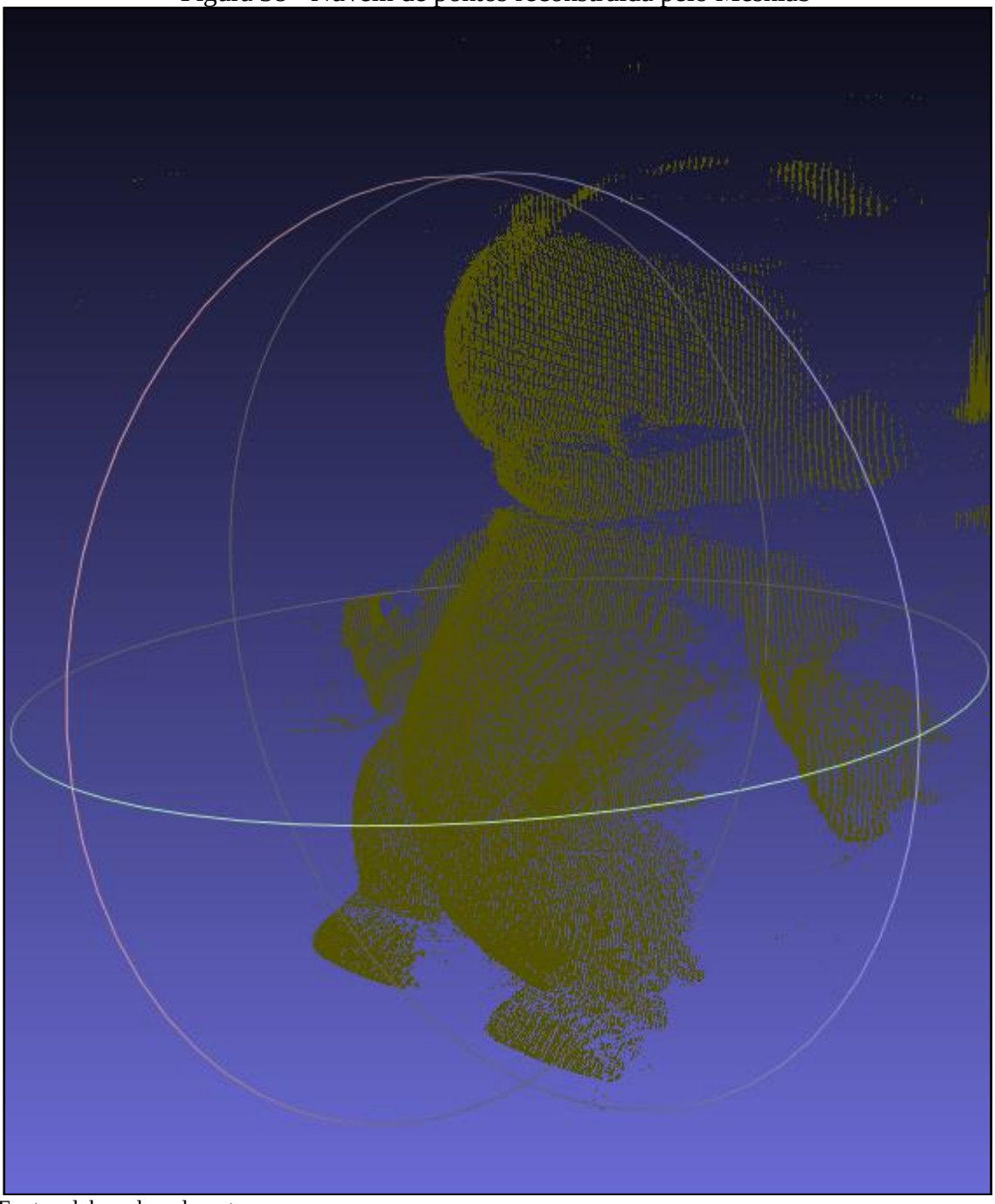
(b) Décima primeira imagem com padrão invertido da décima imagem

(c) Décima segunda imagem obtida

(d) Décima terceira imagem com padrão inverso da décima segunda imagem

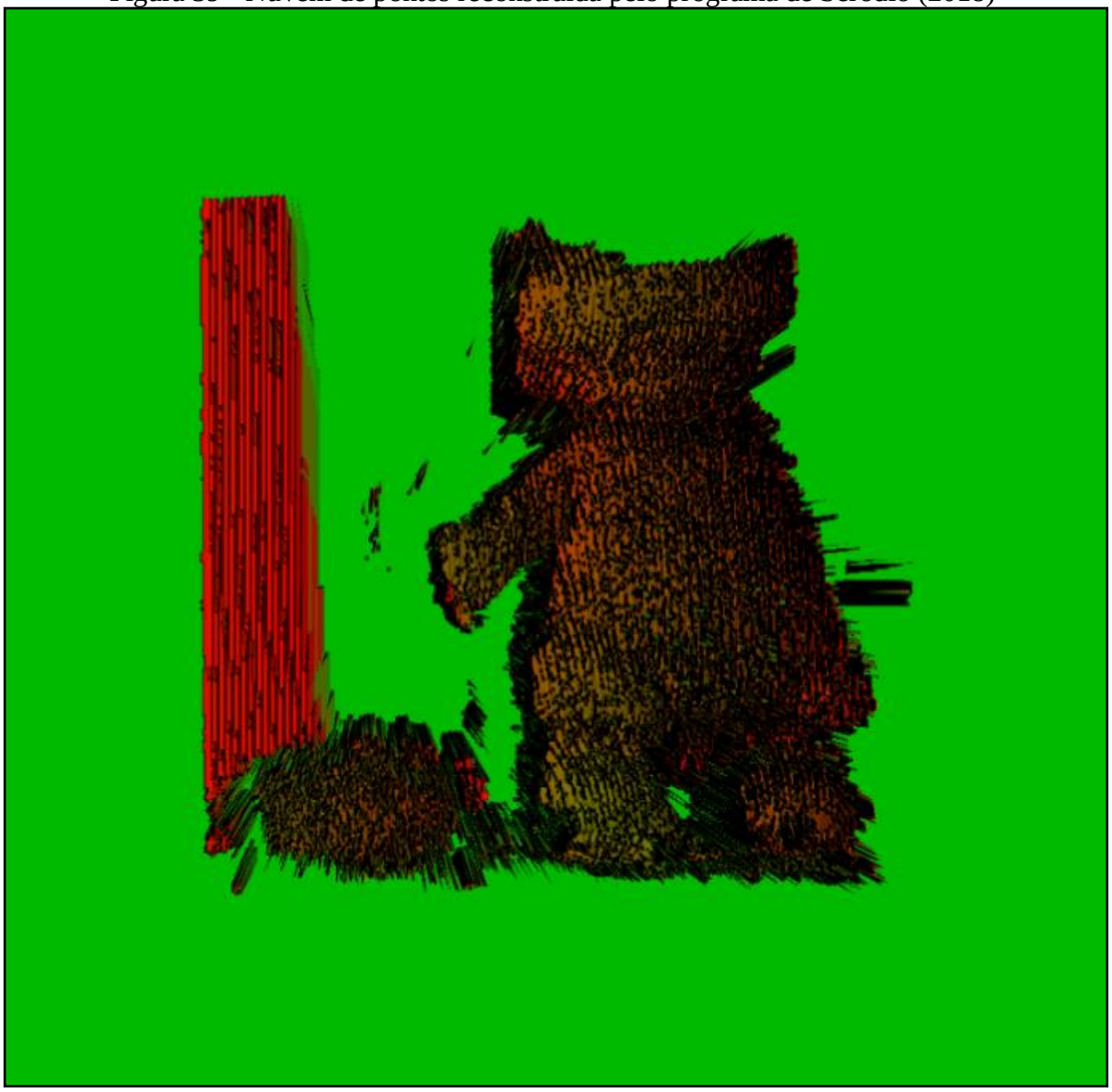
Fonte: elaborado pelo autor.

Figura 38 - Nuvem de pontos reconstruída pelo Meshlab



Fonte: elaborado pelo autor.

Figura 39 - Nuvem de pontos reconstruída pelo programa de Serodio (2018)



Fonte: elaborado pelo autor.

3.4.3 Análise

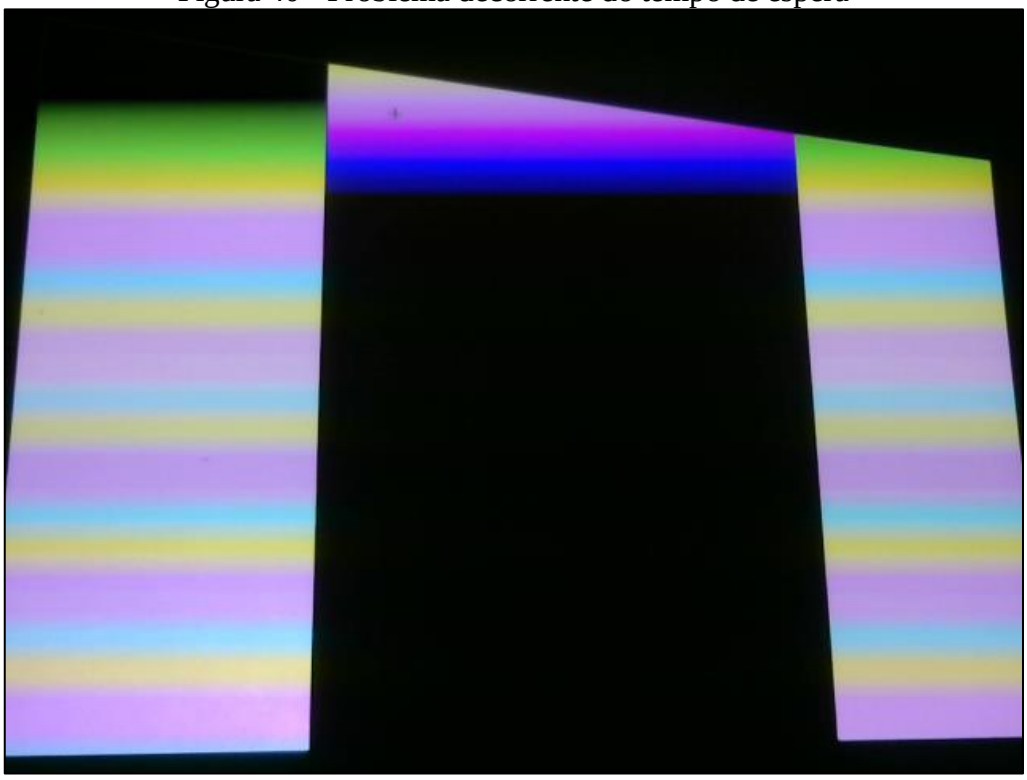
As análises foram subdivididas em duas seções que são separadas pela captura das imagens que são enviadas ou obtidas pela biblioteca e pelo processo de reconstrução.

3.4.3.1 Captura de imagens

A captura das imagens pode ser realizada tanto pela biblioteca quanto pelo programa que utiliza a biblioteca. Sendo assim, analisou-se o tempo de espera necessário para capturar a imagem assim que o padrão fosse projetado sobre uma superfície com o objetivo definir o intervalo de tempo entre a troca de padrões.

Durante essa análise (Tabela 1) detectou-se que o intervalo de tempo de espera na biblioteca entre a projeção e a captura não pode ser inferior a 75ms caso contrário as imagens não serão capturadas corretamente, tendo deformações no padrão de cores, conforme pode ser visto na Figura 40. No Unity, esse tempo é de no mínimo 190ms. Supõe-se que esse problema seja derivado do tempo de resposta do projetor e sua frequência de atualização.

Figura 40 - Problema decorrente do tempo de espera



Fonte: elaborado pelo autor.

Tabela 1 - Testes de tempo mínimo

Plataforma	Tempo de projeção (em milissegundos)	Tempo total (em milissegundos)	Aparição do problema
Biblioteca	50	1454	Sim
	60	1729	Sim
	75	1973	Não
	100	2460	Não
	125	2779	Não
	300	7710	Não
Unity	100	2539	Sim
	150	2878	Sim
	190	3540	Não
	250	4580	Não

Fonte: elaborado pelo autor.

3.4.3.2 Reconstrução

A análise do processo de reconstrução é dividida em três etapas: na velocidade do algoritmo utilizando diversas resoluções de imagens, na interferência da luz ambiente na nuvem de pontos resultante e na utilização de memória e processamento. Para realizar o teste de

desempenho do algoritmo, cronometrou-se o tempo que o processo inteiro de reconstrução levou, desde o carregamento das imagens para a memória do programa até a geração da nuvem de pontos. As resoluções das imagens e seus tempos podem ser vistos na Tabela 2, sendo tais dados mostrados graficamente na Figura 41. Com esses dados, pode-se verificar que o tempo de execução é linear e está diretamente ligado à quantidade total de pontos que a imagem tem.

Tabela 2 - Tempos de processamento do processo de reconstrução

Resolução da imagem	Total de pontos (altura * largura)	Tempo (em milissegundos)
680 x 480	307.200	869
1600 x 1200	1.920.000	6.558
5184 x 3456	17.915.904	43.559

Fonte: elaborado pelo autor.

Figura 41 - Tempo de processamento do algoritmo de reconstrução

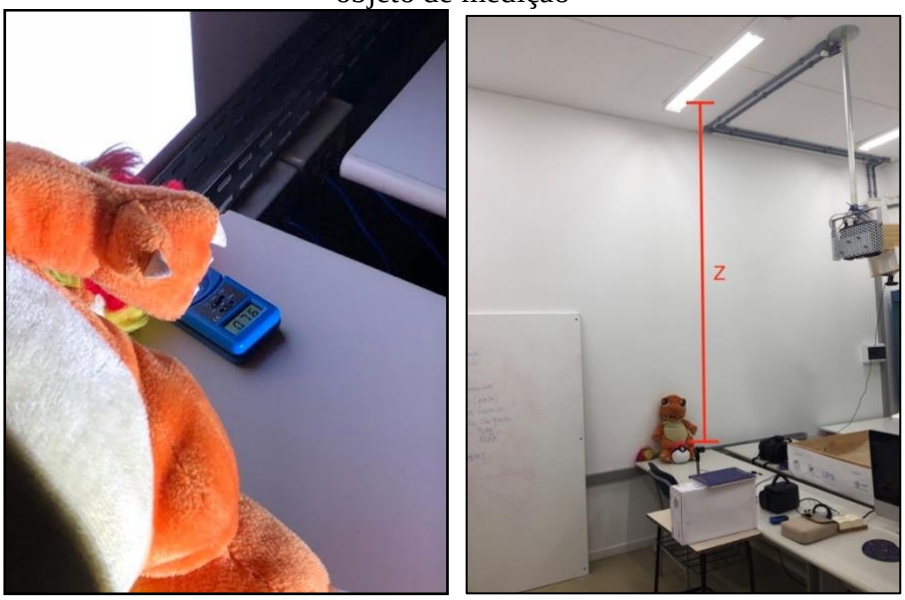
Fonte: elaborado pelo autor.

Outro teste realizado foi para verificar o quanto a variação da luz ambiente iria influenciar no resultado, sendo assim, realizou-se dois testes: um com 70 lux e outro com 90 lux. O medidor de luminosidade foi colocado ao lado do objeto com o projetor emitindo um padrão branco, podendo ser visto na Figura 42 a uma distância de 2,8m do teto, com o objetivo de normalizar o local em que o medidor ficaria. Após capturar as imagens com a câmera, a reconstrução foi realizada e as nuvens de pontos obtidas podem ser vistas nas FigurasFigura 43 eFigura 44.

No que se observou visualmente, ambas as nuvens de pontos parecem bem semelhantes e ao realizar a comparação no total de vértices, 238.009 vértices das capturas com 70 lux contra 237.933 vértices das capturas com 90 lux, verifica-se que uma variação de apenas 20 lux não influencia o resultado de maneira satisfatória. Essa variação de 20 lux representa duas luminárias com duas lâmpadas de 18 watts e temperatura de cor de 4000K localizadas no teto

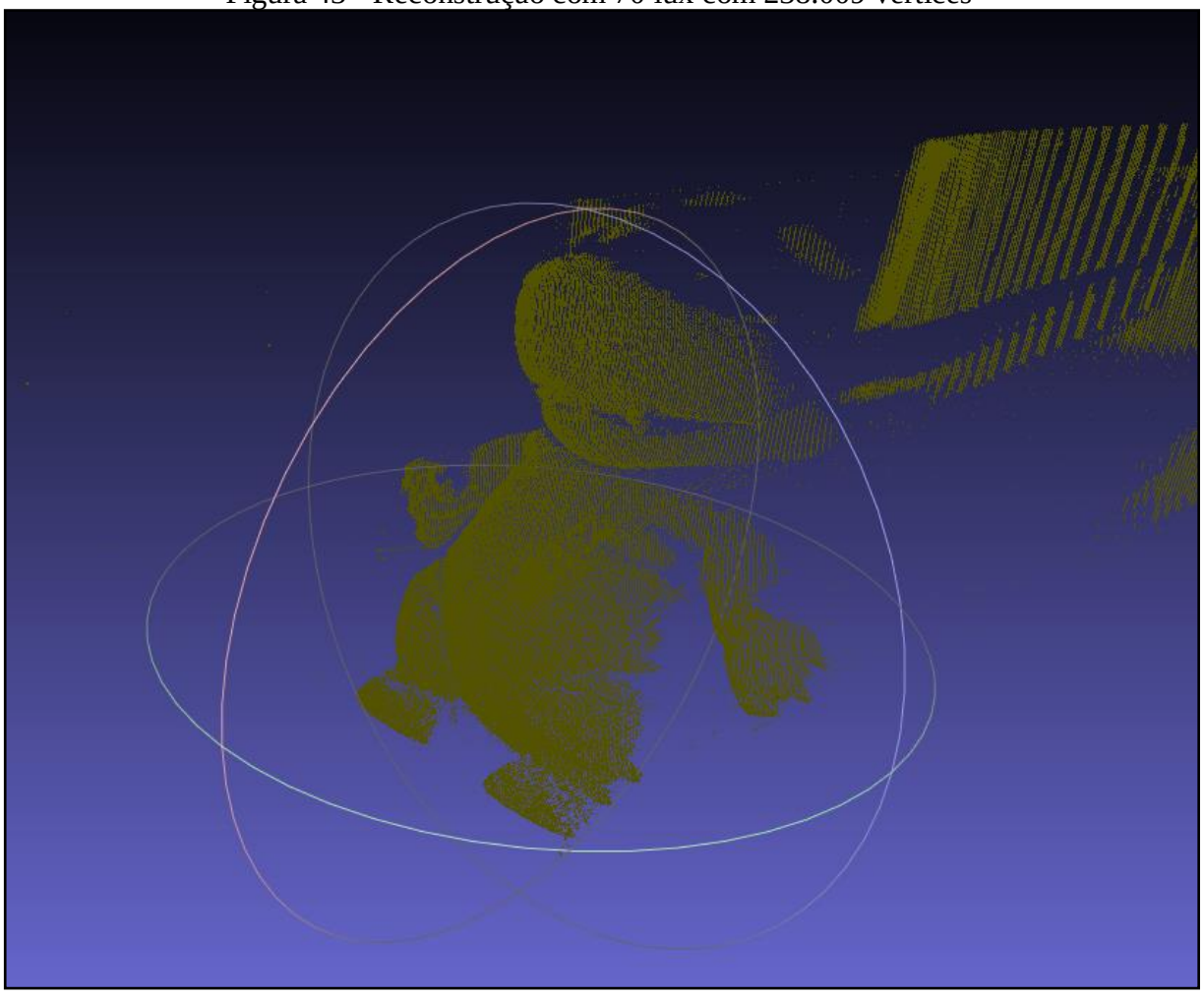
de sala com 11 metros por 6,5 metros do laboratório S-412 da Universidade Regional de Blumenau.

Figura 42 - a) Localização do medidor de luminosidade - b) Posição da luz em relação ao objeto de medição



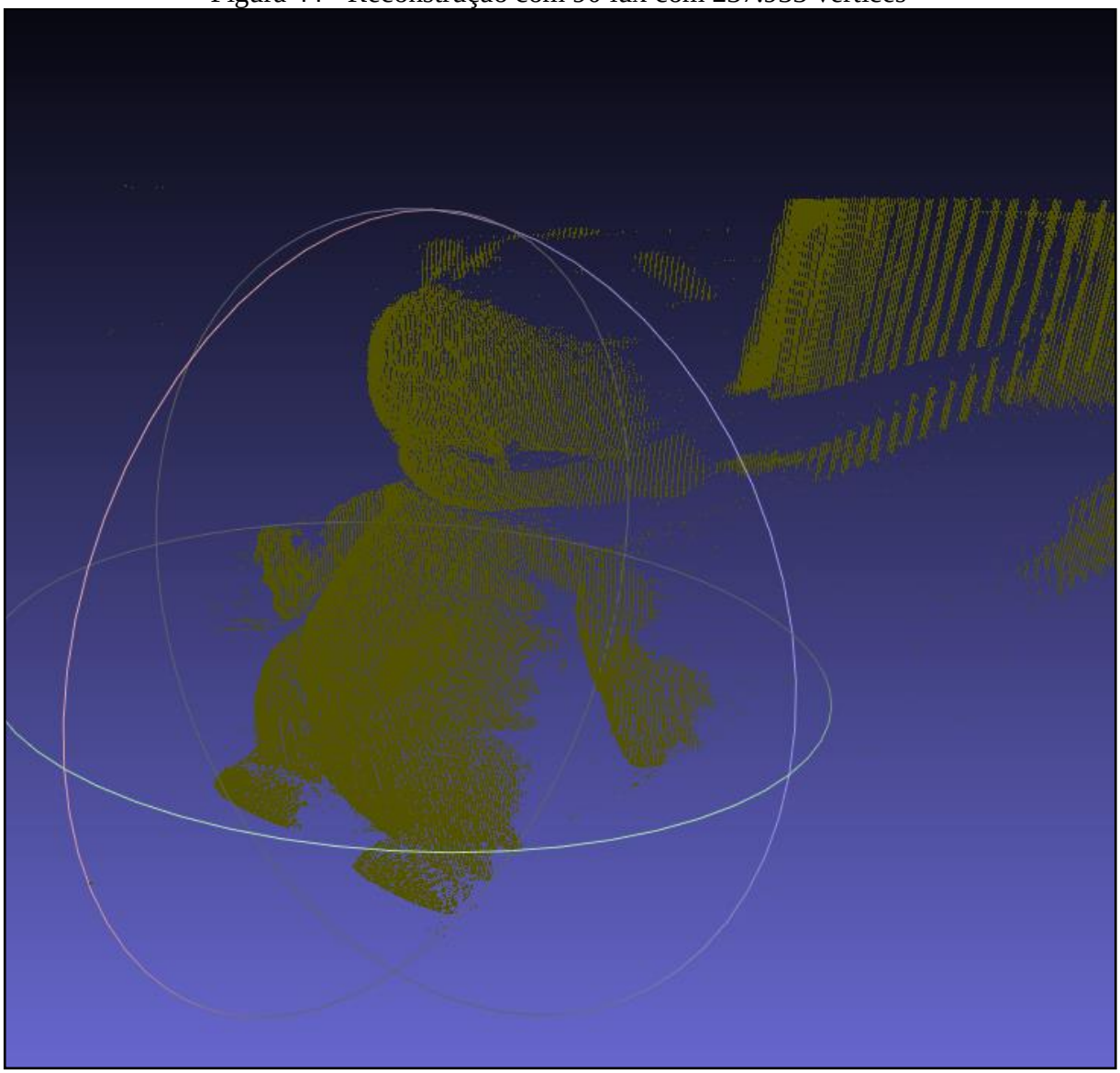
Fonte: elaborado pelo autor.

Figura 43 - Reconstrução com 70 lux com 238.009 vértices



Fonte: elaborado pelo autor.

Figura 44 - Reconstrução com 90 lux com 237.933 vértices

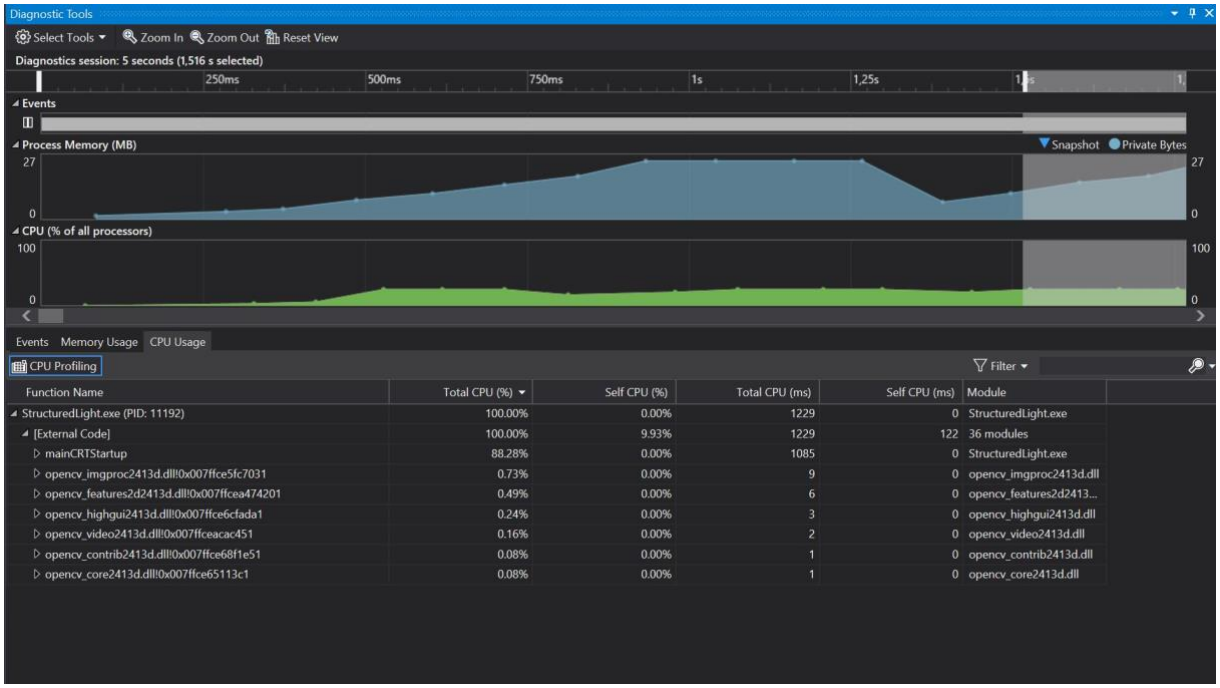


Fonte: elaborado pelo autor.

Para verificar a quantidade de memória utilizada e a porcentagem da CPU utilizada, utilizou-se a ferramenta de *debugger* do Visual Studio. O teste foi realizado com duas resoluções de imagens para verificar a quantidade de espaço utilizado na memória e verificar o uso da CPU durante o processo de reconstrução. Na resolução de 640x480 a utilização total da memória foi de 27MB que representa o ponto em que todas as imagens já haviam sido capturadas e estavam carregadas na memória. O processamento da CPU ficou em torno de 20% em todo o procedimento (Figura 45). Com a resolução de 5184x3456 o pico de memória foi de 1,1 GB no momento em que todas as imagens foram carregadas e o uso de CPU ficou em 25%, o que representa 100% de um núcleo do processador (Figura 46). Com isso é possível verificar que se a biblioteca for usada para processar imagens de baixa resolução existe a possibilidade de se conseguir reconstruir o relevo em tempo real utilizando o processamento na CPU e

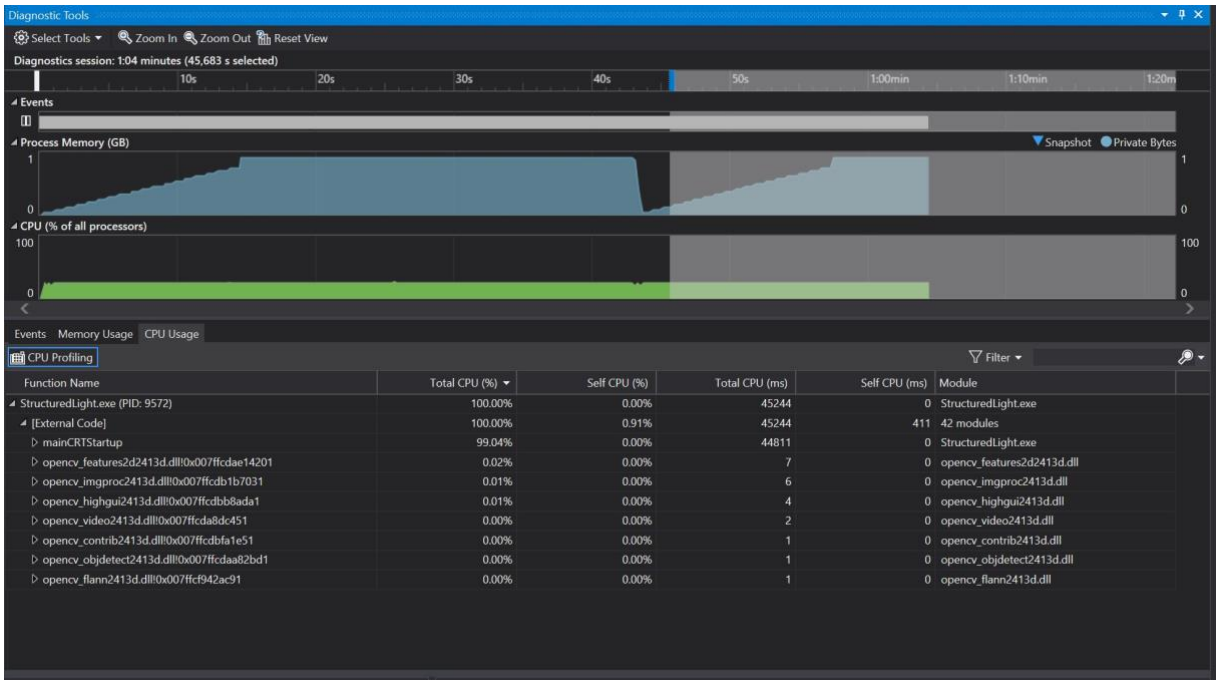
somente com um núcleo, entretanto se forem imagens de alta resolução existe a necessidade de implementar-se o processamento entre múltiplos núcleos a fim de realizar esta validação.

Figura 45 - Utilização de memória e CPU no processamento da imagem em resolução de 640x480



Fonte: elaborado pelo autor.

Figura 46 - Utilização de memória e CPU no processamento da imagem em resolução de 5184x3456



Fonte: elaborado pelo autor.

3.4.4 Comparação de trabalhos

Esta seção faz uma comparação da biblioteca desenvolvida com os trabalhos correlatos demonstrados na seção 0 juntamente com o projeto inicial que foi realizado por Storz (2017). O Quadro 9 apresenta a comparação entre as principais características.

Quadro 9 - Comparativo entre os trabalhos

Trabalhos correlatos / características	Li, Straub e Prautzsch (2004)	Mohan et al. (2011)	Pashaei e Mousavi (2013)	Storz (2017)	Ferramenta desenvolvida
Método de reconstrução	Luz estruturada	Luz estruturada	Luz estruturada + visão estérea	Luz estruturada	Luz estruturada
Tipo de padrão projetado	Sequência de De Brujin	N-ary Codes	Sequência de Gray + deslocamento	Codificação não formal baseada na sequência de De Brujin	Binary Code
Quantidade de imagens necessárias para reconstrução	1	1	Até 20	1	Até 20
Quantidade de câmeras	1	1	2	1	1
Geração da nuvem de pontos	Sim	Não	Sim	Não	Sim
Geração do mapa de disparidade	Não	Sim	Não	Sim	Sim
Calibração de câmera e projetor	Sim	Sim	Sim	Sim	Baseada em valores configurados pelo programa base

Fonte: elaborado pelo autor.

Por meio das informações demonstradas no Quadro 9, todos os trabalhos utilizam a técnica de reconstrução através de luz estruturada, sendo que Pashaei e Mousavi (2013) também incorporam visão estérea no processo para melhorar os resultados obtidos de cada câmera individualmente. Nos padrões projetados há uma grande diferença que é perceptível pela quantidade de imagens que precisam ser capturadas e o tipo de padrão que foi projetado. Trabalhos como Li, Straub e Prautzsch (2004), Mohan et al. (2011) e Storz (2017) precisam de apenas uma imagem pois utilizam padrões coloridos ou personalizados, sendo assim não há a necessidade de analisar a distorção das linhas projetadas diversas vezes como o Pashaei e Mousavi (2013) e o trabalho proposto fazem.

Na geração de uma saída, os trabalhos apresentam sempre uma alternativa entre nuvem de pontos e um mapa de disparidade, com exceção do trabalho proposto onde ambos estão presentes. Já na parte de calibração, apenas o trabalho proposto não realiza a calibração através do método convencional (tirar fotos do tabuleiro de xadrez) pois se baseia no parâmetro `zValue` que é informado pelo usuário durante a utilização da biblioteca.

Comparando a biblioteca atual com a iniciada por Storz (2017), Storz (2017, p. 66) afirma que o tempo necessário para realizar a reconstrução é próximo a uma hora, em testes realizados no mesmo computador apresentado na seção 3.4.1, o processo de reconstrução demorou em média quatro minutos e meio para uma imagem na resolução de 640x480 e se comparado com a biblioteca atual isso representa uma melhoria de aproximadamente 310 vezes tendo em vista que ela leva aproximadamente 870 milissegundos para realizar a reconstrução.

4 CONCLUSÕES

Este trabalho mostrou o desenvolvimento de uma biblioteca para realizar reconstrução de um relevo baseado no conceito de luz estruturada, utilizando um projetor e uma câmera. O objetivo da biblioteca é auxiliar no projeto Caixa e Água (2015) com a função de remover o Kinect e baratear o custo dos equipamentos utilizados. Embora a escolha do padrão de projeção não tenha sido a melhor, tendo em vista que existe a necessidade de projetar até 20 padrões, gastando um tempo precioso caso seja utilizada para reconstruir relevos em tempo real, a reconstrução obtida conseguiu demonstrar fidelidade à superfície que foi reconstruída.

Em relação a melhoria do trabalho iniciado por Storz (2017), a biblioteca proposta conseguiu corrigir os erros que o mesmo teve, além disso melhorar relativamente o desempenho sem a utilização de GPU. O padrão de projeção escolhido não foi o melhor pois o tempo de captura das imagens acaba inviabilizando a biblioteca para utilização tem tempo real, assim sendo, recomenda-se a mudança do padrão para que utilize somente uma imagem de captura.

Segundo Storz (2017, p. 66) a necessidade da criação de um *script* na plataforma Unity era necessária para que a nuvem de pontos pudesse ser importada e um terreno renderizado. Tal problema foi solucionado com a biblioteca desenvolvida por Serodio (2018), porém nesta biblioteca a forma de comunicação entre o programa Unity e biblioteca são precários, sendo realizados através de arquivos em disco.

Uma das principais vantagens no trabalho é a alteração do processo de calibração para poucas variáveis, apesar de que pode ser difícil encontrar tais parâmetros, se comparado com a biblioteca iniciada por Storz (2017).

4.1 EXTENSÕES

Sugere-se como extensões para trabalhos futuros:

a) alterar o padrão de projeção para utilizar uma imagem única;
b) alterar a calibração dos projetores que foi proposta por Storz (2017) para que a biblioteca receba as imagens e obtenha os parâmetros de calibração;
c) remover a dependência do OpenCV do projeto, a fim de tornar o projeto o mais independente possível;
d) melhorar a qualidade das capturas das imagens pela câmera com o objetivo de melhorar a qualidade do resultado;
e) remover o gargalo de captura das imagens;
f) utilizar a técnica de visão estérea com o objetivo de não precisar projetar um padrão para analisar a distorção, tendo em vista que para que o trabalho seja utilizado no

projeto Caixa e Água (2015) existe uma grande necessidade da reconstrução ser realizada em tempo real;

g) alterar a forma de comunicação da biblioteca com os programas que se comunicam com ela.

REFERÊNCIAS

ARROYO, Steven et al. **Kinect Like Camera**. [S.I], 2011. Disponível em: <https://code.google.com/archive/p/kinect-like-3d-camera>. Acesso em: 20 jun. 2018.

AVINS, Jerry. **DSP Trick: Gray Code Conversion**. [S.I], 2000. Disponível em: <https://dspguru.com/dsp/tricks/gray-code-conversion/>. Acesso em: 21 jun. 2018.

CAIXA E ÁGUA. **Conheça o projeto Caixa e Água**. Blumenau, 2015. Disponível em: <http://caixae-agua.blogspot.com.br/2016/07/conheca-o-projeto-caixae-agua.html>. Acesso em: 08 set. 2017.

CAMPOS, Antônio Carlos. **Representação do relevo nas cartas topográficas**. [S.I]. [2008?]. Disponível em: <http://www.cesadufs.com.br/ORBI/public/uploadCatalago/11210904042012Cartografia_Basica_Aula_17.pdf>. Acesso em: 08 set. 2017.

CELANI, Gabriela; CANCHERINI, Ramon. Digitalização tridimensional de objetos: um estudo de caso. In: CONGRESSO DA SOCIEDADE IBEROAMERICANA DE GRÁFICA DIGITAL, 13, 2009, São Paulo. **Proceedings...** São Paulo, 2009, p.309-311. Disponível em: <http://cumincades.scix.net/cgi-bin/works/Show?sigradi2009_1012>. Acesso em: 20 jun. 2018.

CHEN, Yanjun et al. Coded Structured Light Stripe Boundary Location with Sub-Pixel Accuracy Based on Weighted Centroid Method. **Journal of Biological Systems**, v. 18, n. spec01, p. 35-49, 2010. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/S0218339010003603>. Acesso em: 20 jun. 2018.

FERNANDES, Leandro Augusto Frata. **Estudo de Métodos para Extração de Formas e Realização de Medidas a Partir de Imagens.** 2005. 32 p. Trabalho Individual I (Programa de Pós-graduação em Informática) - Universidade Federal do Rio Grande do Sul. Disponível em: <http://www2.ic.uff.br/~laffernandes/projects/metrology/2005_UFRGS_TI/fernandes_TI_1159.pdf>. Acesso em: 05 set. 2017.

GENG, Jason. Structured-light 3D surface imaging: a tutorial. **Advances in Optics and Photonics**, v. 3, n. 2, p. 128-160, 2011. Disponível em: <http://aop.osa.org/abstract.cfm?URI=aop-3-2-128>. Acesso em: 13 jun. 2018.

HERAKLEOUS, Kyriakos; POULLIS, Charalambos. 3dunderworld-sls: An open-source structured-light scanning system for rapid geometry acquisition. **arXiv preprint arXiv:1406.6595**, 2014. Disponível em: <https://arxiv.org/abs/1406.6595>. Acesso em: 18 jun. 2018.

LI, Hao; STRAUB, Raphael; PRAUTZSCH, Hartmut. Fast subpixel accurate reconstruction using color structured ligth. In: PROCEEDINGS OF THE FOURTH IASTED INTERNATIONAL CONFERENCE ON VISUALIZATION, IMAGING AND IMAGE PROCESSING, 4., 2004, Marbella. **Proceedings…** Marbella, IASTED, 2004, p. 396–401. Disponível em: <http://www.hao-li.com/publications/papers/viip2004FSA.pdf>. Acesso em: 13 jun. 2018.

MENDONÇA, Cláudio. **Topografia (1):** Hipsometria e curvas de nível. São Paulo. 2007. Disponível em: <https://educacao.uol.com.br/disciplinas/geografia/topografia-1-hipsometria-e-curvas-de-nivel.htm>. Acesso em: 07 set. 2017.

MOHAN, San et al. 3D scanning of object surfaces using structured light and a single camera image. In: 2011 IEEE INTERNATIONAL CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING, 2011, Trieste. **Proceedings…** Trieste, IEEE, 2011, p. 151-156. Disponível em: <https://ieeexplore.ieee.org/document/6042450>. Acesso em: 13 jun. 2018.

MORENO, Daniel; TAUBIN, Gabriel. **Simple, Accurate, and Robust Projector-Camera Calibration**. Providence: Brown University, 2012. 8 p. Disponível em: <http://mesh.brown.edu/calibration/files/Simple, Accurate, and Robust Projector-Camera Calibration.pdf>. Acesso em: 20 jun. 2018.

PASHAEI, M.; MOUSAVI, S. M. Implementation of a low cost structured light scanner. **International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, v. XL-5, p. W2, 2013. Disponível em: <https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-5-W2/477/2013/isprsarchives-XL-5-W2-477-2013.pdf>. Acesso em: 13 jun. 2018.

SERODIO, Alex. **Unity + DLL**. [mensagem pessoal]. Mensagem recebida por <leschlogl@gmail.com> em 20 jun. de 2018.

STORZ, Kevin. **Biblioteca para detecção de relevos**. 2017. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.