

Melhorias na biblioteca para detecção de relevos

Aluno: Lucas Eduardo Schlögl

Orientador: Dalton Solano dos Reis

Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Trabalhos Correlatos
- Biblioteca de Storz (2017)
- Requisitos
- Especificação
- Implementação
- Resultados e discussões
- Conclusões e sugestões

Introdução

- Melhoria dos equipamentos;
- Extrair informações geométricas da cena;
- Técnicas ativas e passivas;
- Custo de equipamentos:
 - De 110 dólares até 100.000 dólares;
- Projeto Caixa e Água (2015).

Objetivos

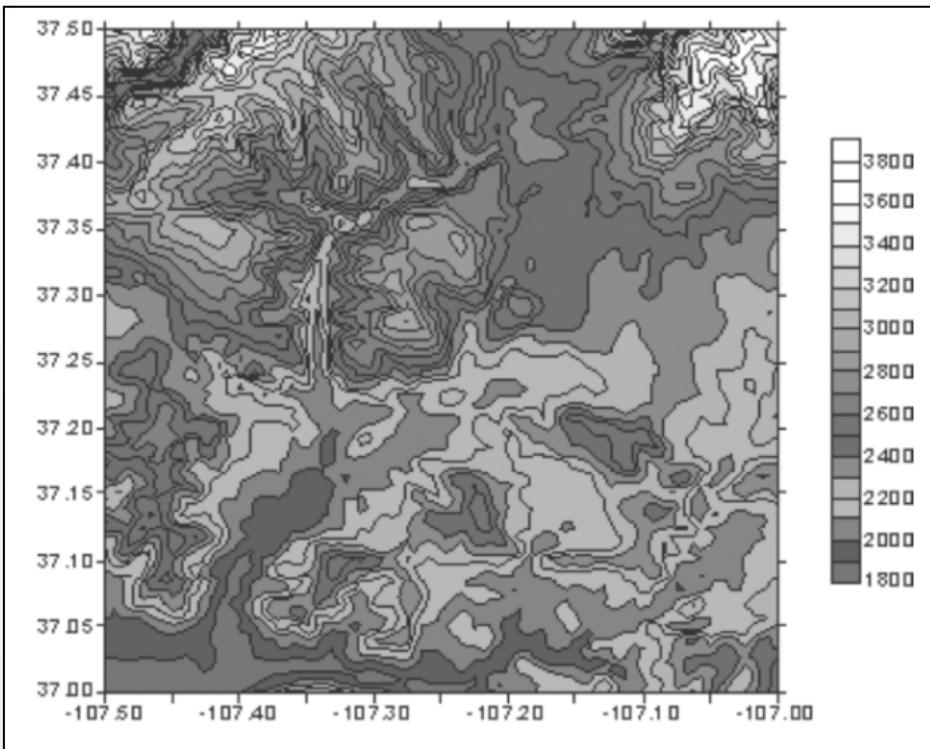
O objetivo é desenvolver melhorias na biblioteca para detecção de relevos iniciada por Storz (2017).

- Corrigir os erros encontrados no processo de reconstrução do trabalho de Storz (2017);
- Melhorar o desempenho no processo de reconstrução;
- Gerar uma nuvem de pontos para que outros programas possam reconstruir o terreno.

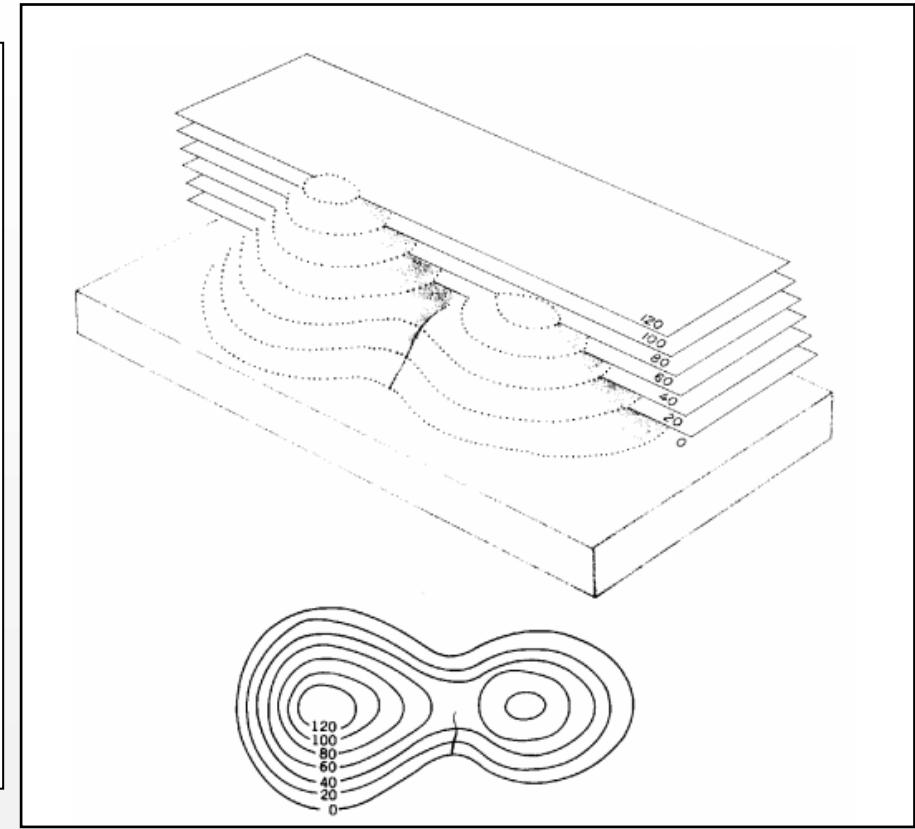
Fundamentação Teórica

- Representação de relevos
- Triangularização ativa
- Padrões de projeção

Representação do Relevo



Fonte: Campos (2008).

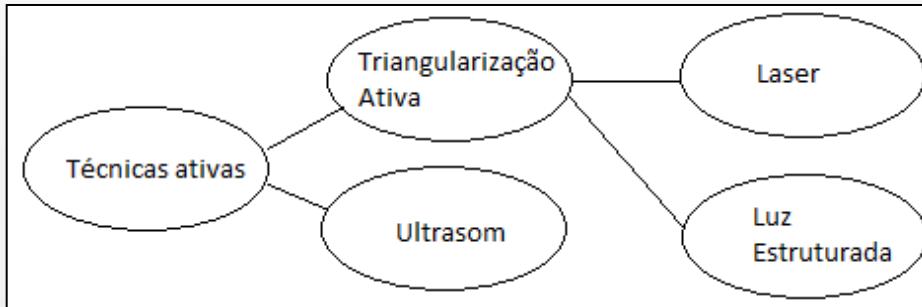


Fonte: Campos (2008).

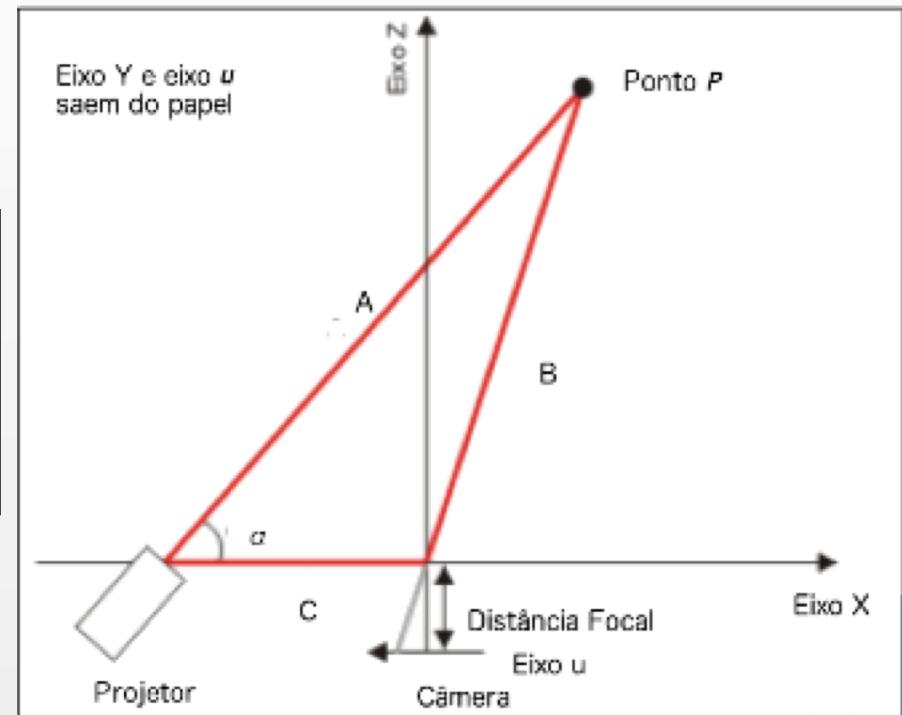
Reconstrução 3D

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

Fonte: Cerani e Cancherini (2009).

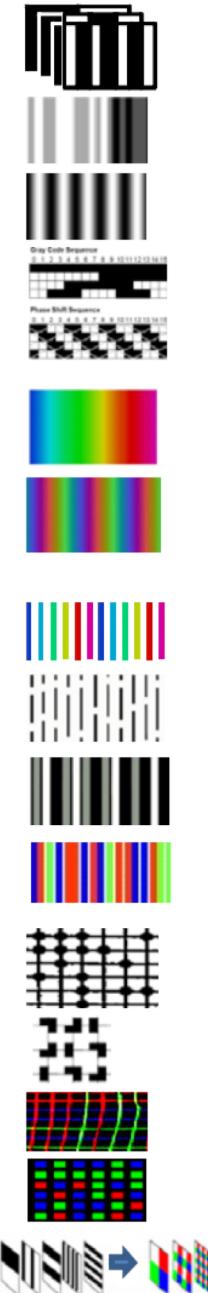
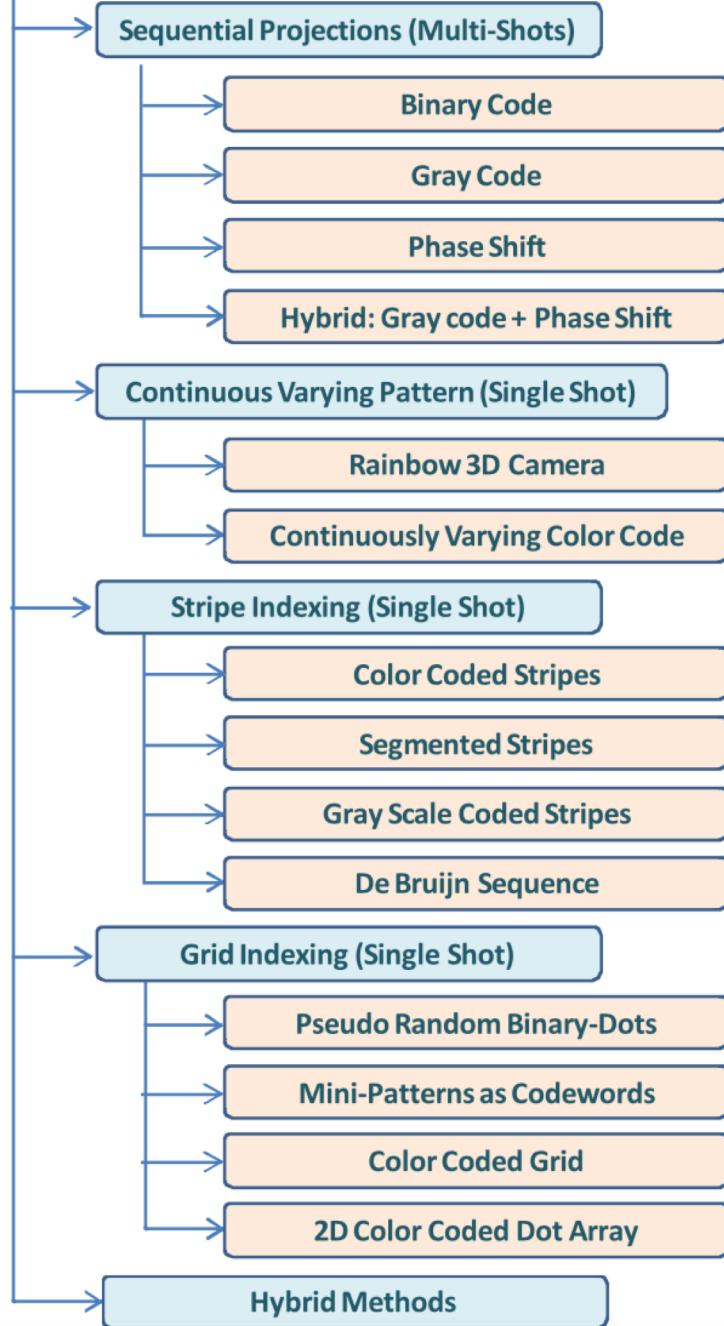


Fonte: Adaptado de Fernandes (2005).



Fonte: Fernandes (2005).

Structured Light 3D Surface Imaging Techniques



Padrões de Projeção

Fonte: Geng (2010).

Trabalhos Correlatos

- Li, Straub e Prautzsch (2004) - Fast subpixel accurate reconstruction using color structured light

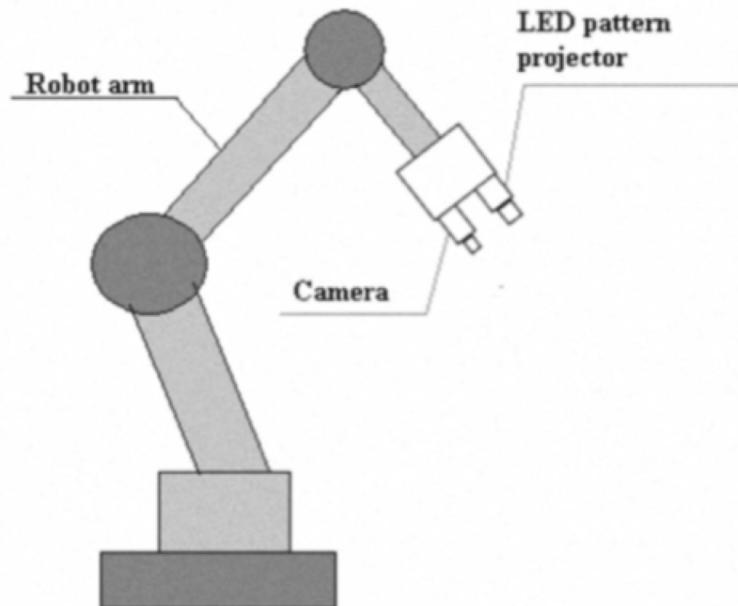


Fonte: Li, Straub e Prautzsch (2004).

Trabalhos correlatos / características	Li, Straub e Prautzsch (2004)
Método de reconstrução	Luz estruturada
Tipo de padrão projetado	Sequência de De Brujin
Quantidade de imagens necessárias para reconstrução	1
Quantidade de câmeras	1
Geração da nuvem de pontos	Sim
Geração do mapa de disparidade	Não
Calibração de câmera e projetor	Sim

Trabalhos Correlatos

- Mohan et al. (2011) - 3D scanning of object surfaces using structured light and single camera image



Trabalhos correlatos / características	Mohan et al. (2011)
Método de reconstrução	Luz estruturada
Tipo de padrão projetado	N-ary Codes
Quantidade de imagens necessárias para reconstrução	1
Quantidade de câmeras	1
Geração da nuvem de pontos	Não
Geração do mapa de disparidade	Sim
Calibração de câmera e projetor	Sim

Trabalhos Correlatos

- Pashaei e Mousavi (2013) - Implementation of a low cost structured light scanner



Fonte: Pashaei e Mousavi (2013).

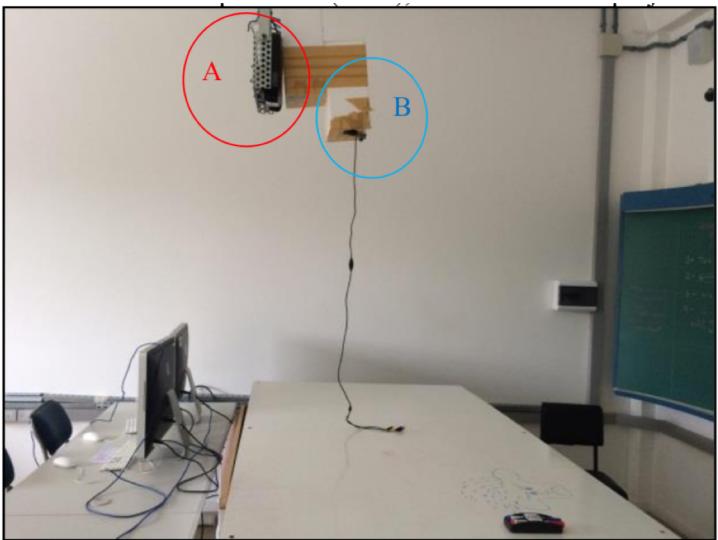
Trabalhos correlatos / características	Pashaei e Mousavi (2013)
Método de reconstrução	Luz estruturada + visão estérea
Tipo de padrão projetado	Sequência de Gray + deslocamento
Quantidade de imagens necessárias para reconstrução	Até 20
Quantidade de câmeras	2
Geração da nuvem de pontos	Sim
Geração do mapa de disparidade	Não
Calibração de câmera e projetor	Sim

Biblioteca de Storz (2017)

- Calibração baseada em analisar um tabuleiro de Xadrez;
- Padrão de projeção baseado na sequência de De Bruijn;
- Aproximadamente 4 minutos e 30 segundos para processar na resolução 640x480;
- Geração de mapa de disparidade.

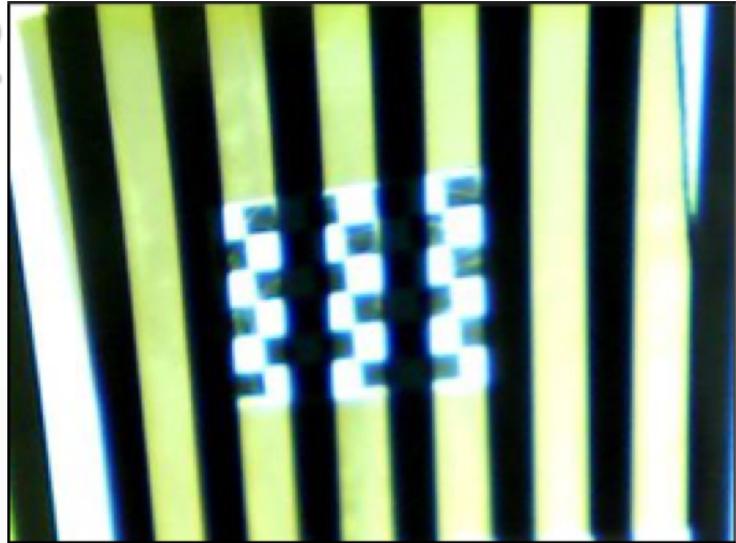
Biblioteca de Storz (2017)

1



Fonte: autor.

2



Fonte: Storz (2017).

3



Fonte: Storz (2017).

Requisitos

- a) utilizar uma câmera para capturar imagens de uma cena que será realizada o reconhecimento do relevo (RF);
- b) detectar a profundidade do relevo pelo método de triangularização ativa (RF);
- c) gerar um *dataset* do relevo obtido a fim de ser importado pelo Unity (RF);
- d) obter um tempo de reconstrução para que seja possível utilizar a biblioteca 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 Visual Studio para compilar e executar a biblioteca (RNF).

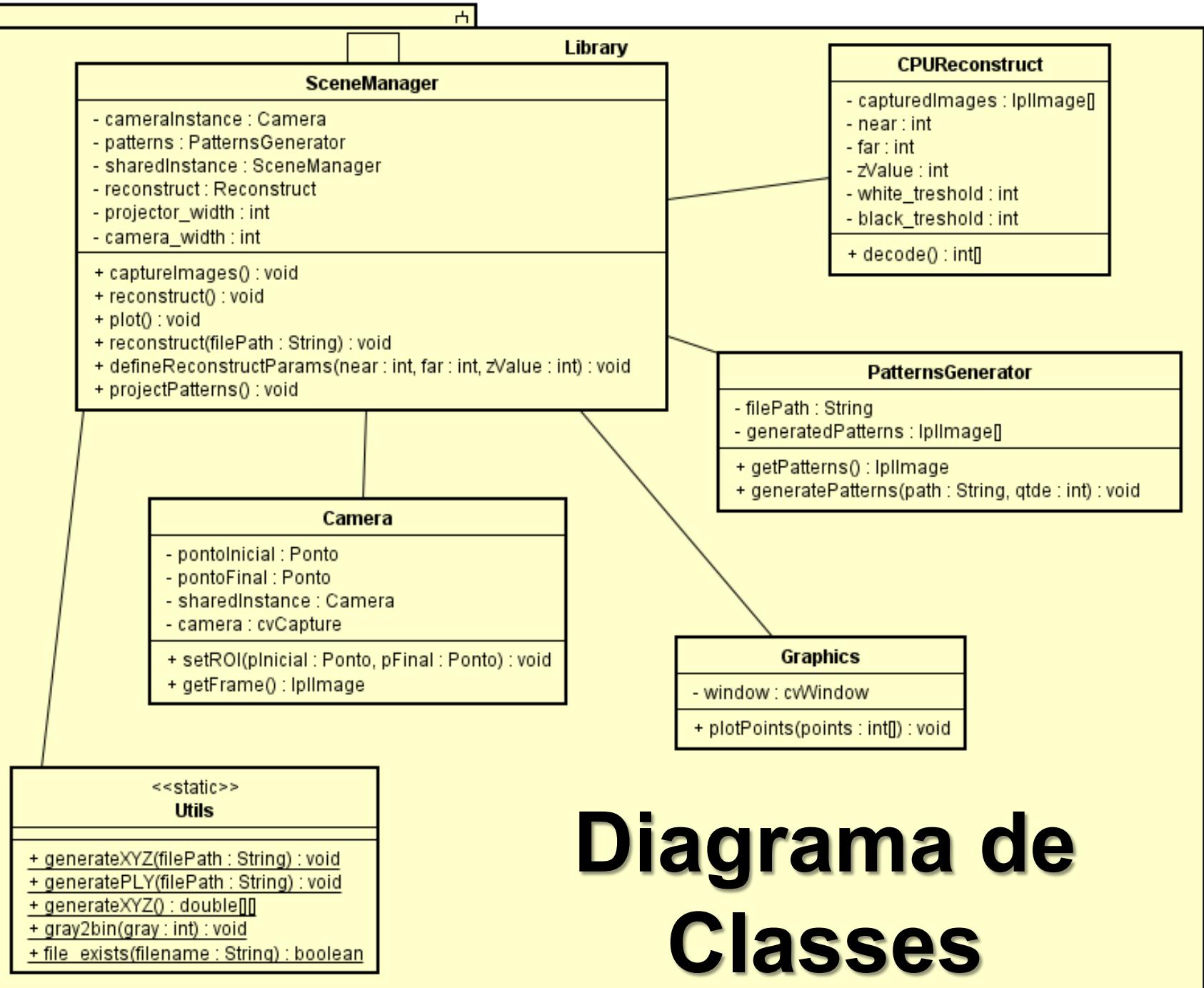
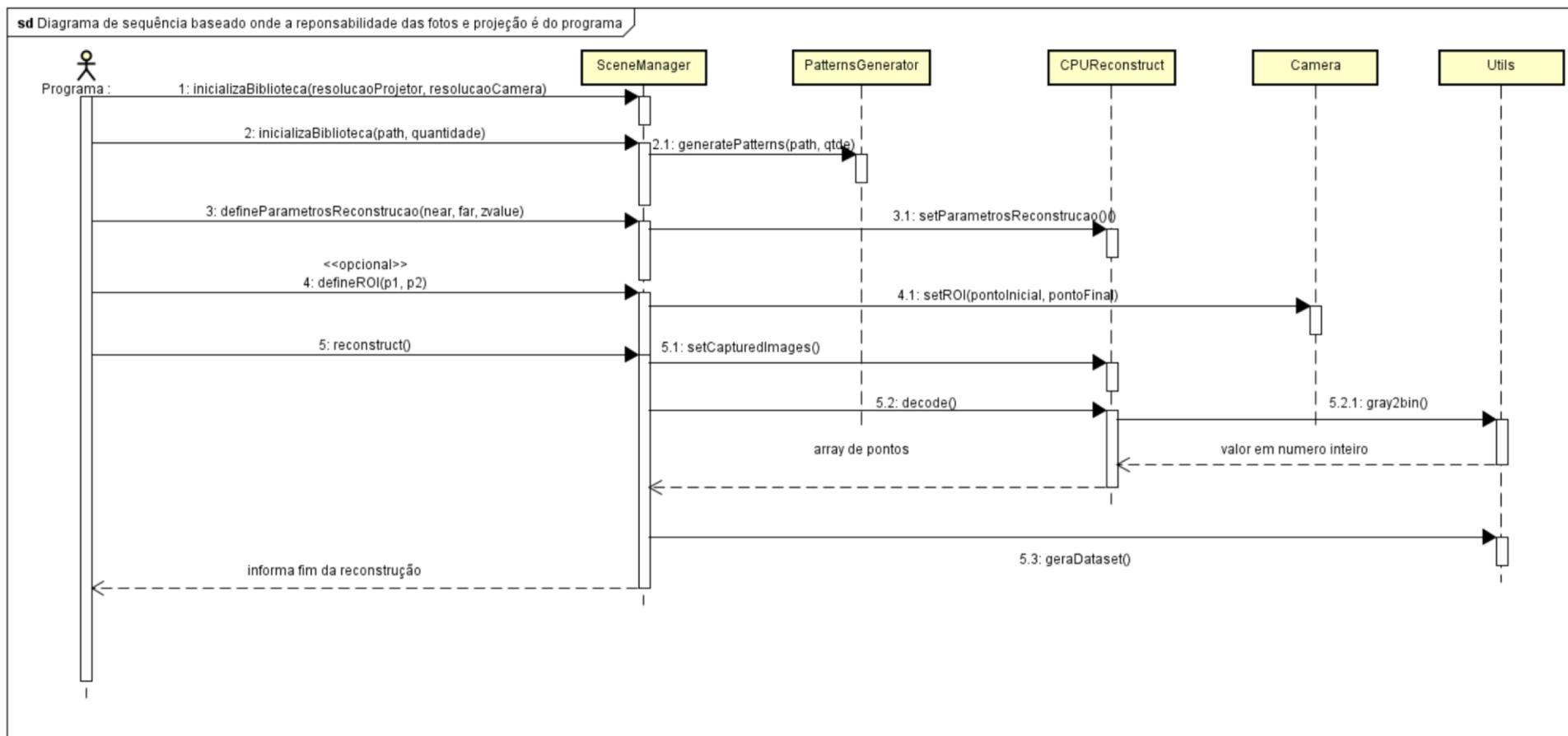


Diagrama de Classes

Diagrama de Sequência



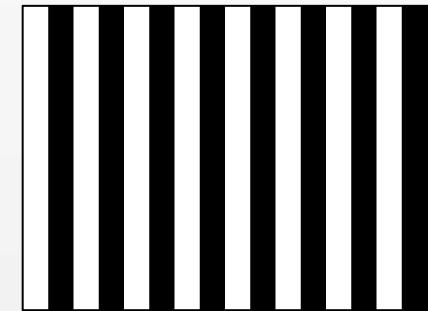
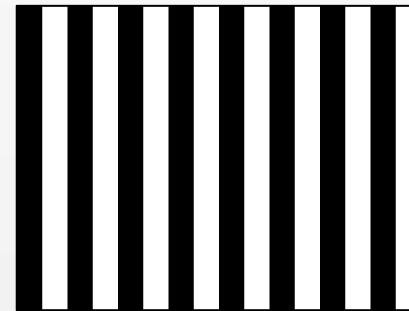
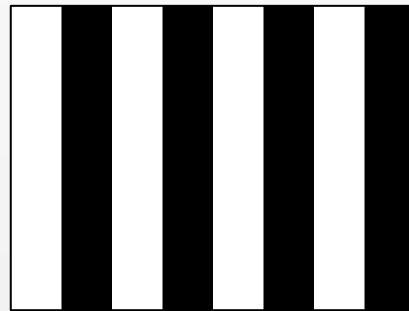
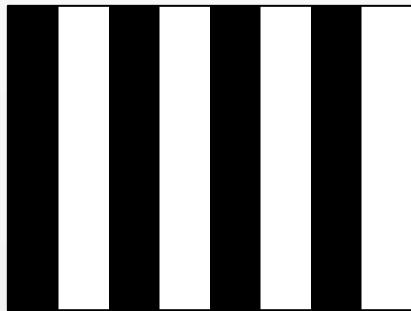
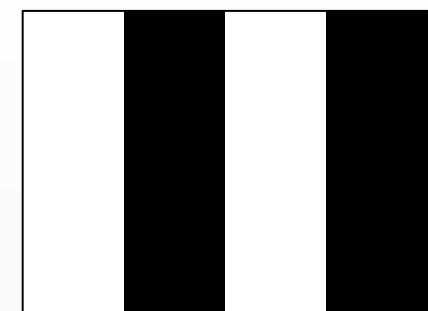
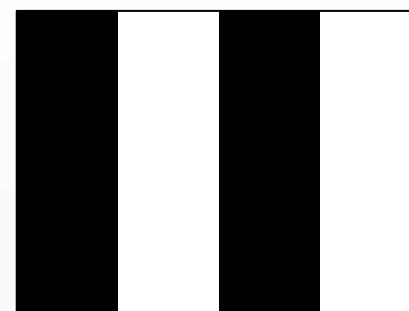
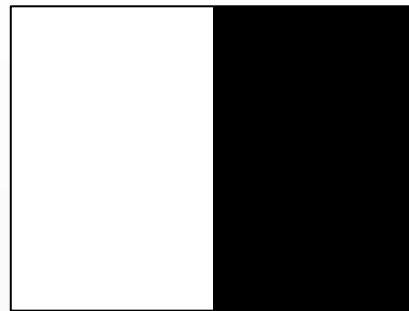
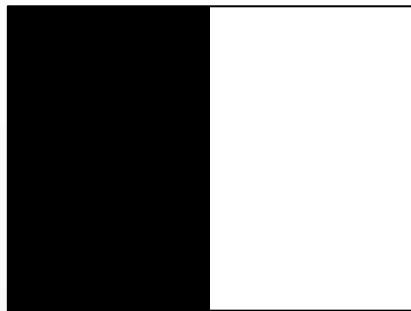
Implementação

- Geração dos padrões
- Captura de imagens
- Reconstrução
- Geração de *dataset XYZ*
- Utilização da biblioteca

Geração dos Padrões

```
1 void generatePatterns(int numberOfPatterns, int patternWidth, int patternHeight, string filePath) {  
2  
3     int counter = 1;  
4     for (int i = 1; i < numberOfPatterns + 1; i++) {  
5         Mat generatedPattern(patternHeight, patternWidth, CV_8UC3, Scalar(255, 255, 255));  
6  
7         int numberofStripes = pow(2, i);  
8         int stripeWidth = patternWidth / pow(2, i);  
9  
10        for (int j = 0; j < numberofStripes; j++) {  
11            int xMargin = j * stripeWidth;  
12            cv::Rect rect(xMargin * 2, 0, stripeWidth, patternHeight);  
13            cv::rectangle(generatedPattern, rect, cv::Scalar(0, 0, 0), CV_FILLED);  
14        }  
15  
16        std::stringstream stringStreamPattern;  
17        stringStreamPattern << filePath << "\\\" << counter << ".jpg";  
18        std::string stringPath = stringStreamPattern.str();  
19  
20        imwrite(stringPath, generatedPattern);  
21        counter++;  
22        Mat invertedPattern = ~generatedPattern;  
23  
24        std::stringstream stringStreamPatternInverted;  
25        stringStreamPatternInverted << filePath << "\\\" << counter << ".jpg";  
26        std::string stringPathInverted = stringStreamPatternInverted.str();  
27  
28        imwrite(stringPathInverted, invertedPattern);  
29        counter++;  
30  
31    }  
32  
33}
```

Geração dos Padrões



Quantidade de padrões = 4

Algoritmo de captura de imágenes

```
ptions ▾ void getFrames (int numberofImages, string patternsPath, string saveImagePath){  
2     vector<IplImage *> capturedImages(numberOfImages+2);  
3     vector<IplImage *> patterns(numberOfImages);  
4  
5     for (int i = 0; i < numberofImages; i++) {  
6         std::stringstream stringStreamPatterns;  
7         stringStreamPatterns << patternsPath << "\\\" << i + 1 << ".jpg";  
8         std::string stringPath = stringStreamPatterns.str();  
9  
10        patterns[i] = cvLoadImage(stringPath.c_str());  
11        if (!patterns[i]) {  
12            printf("Couldn't load pattern: %s\n", stringPath);  
13            return;  
14        }  
15    }  
16    }  
17  
18  
19    cvStartWindowThread();  
20    cvNamedWindow("mainWin", CV_WINDOW_NORMAL);  
21    cvSetWindowProperty("mainWin", CV_WND_PROP_FULLSCREEN, CV_WINDOW_FULLSCREEN);  
22  
23    cvWaitKey(100);  
24  
25    CvCapture* capture = cvCaptureFromCAM(CV_CAP_ANY);  
26    if (!capture) {  
27        fprintf(stderr, "Camera not found \n");  
28        getchar();  
29        exit(0);  
30    }  
31    cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 480);  
32    cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 640);  
33
```

Algoritmo de captura de imagens

```
35 //Display images and capture
36 for (int i = 0; i<numberOfImages; i++) {
37
38     // show the image
39     cvShowImage("mainWin", patterns[i]);
40
41     // pause if time == 0, wait key
42     cvWaitKey(200);
43
44     // capture the image
45     cvGrabFrame(capture);
46     capturedImages [numberOfImages + 1] = cvRetrieveFrame(capture);
47
48     capturedImages [i] = cvCloneImage(capturedImages [numberOfImages + 1]);
49
50     std::stringstream stringStreamCapture;
51     stringStreamCapture << saveImagePath << "\\\" << i + 1 << ".jpg";
52     std::string stringPathCaptured = stringStreamCapture.str();
53
54     cvSaveImage(stringPathCaptured.c_str(), capturedImages [numberOfImages + 1]);
55
56 }
57
58 cvReleaseCapture(&capture);
59
60 cvDestroyWindow("mainWin");
61
62 }
```

Algoritmo de reconstrução

```
1 int* reconstruct(int white_threshold, int black_threshold, vector<IplImage *> imgsCapturadas) {  
2  
3     int *matriz = (int*)calloc(imgsCapturadas[0]->imageSize, sizeof(int));  
4     if (matriz == NULL) {  
5         printf("Erro ao inicializar matriz de pontos\n");  
6         return NULL;  
7     }  
8     for (int i = 0; i < imgsCapturadas.size(); i += 2) {  
9         uchar* data1 = (uchar*)imgsCapturadas[i]->imageData;  
10        uchar* data0 = (uchar*)imgsCapturadas[i + 1]->imageData;  
11        for (int y = 0; y < imgsCapturadas[i]->height; y++) {  
12            for (int x = 0; x < imgsCapturadas[i]->width; x++) {  
13                int u = y*imgsCapturadas[i]->widthStep + x*imgsCapturadas[i]->nChannels;  
14                if (matriz[u] == -1) { continue; }  
15                int gray1 = data1[u] + data1[u + 1] + data1[u + 2];  
16                int gray0 = data0[u] + data0[u + 1] + data0[u + 2];  
17  
18                matriz[u] = matriz[u] << 1;  
19  
20                if (gray1 - gray0 > white_threshold) {  
21                    matriz[u] |= 1;  
22                } else if (gray1 - gray0 < black_threshold) {  
23                    matriz[u] |= 0;  
24                } else {  
25                    matriz[u] = -1;  
26                }  
27            }  
28        }  
29    }  
30    for (int i = 0; i < imgsCapturadas[0]->imageSize; i++) {  
31        matriz[i] = gray2bin(matriz[i]);  
32    }  
33    return matriz;  
34}
```

Geração do dataset XYZ

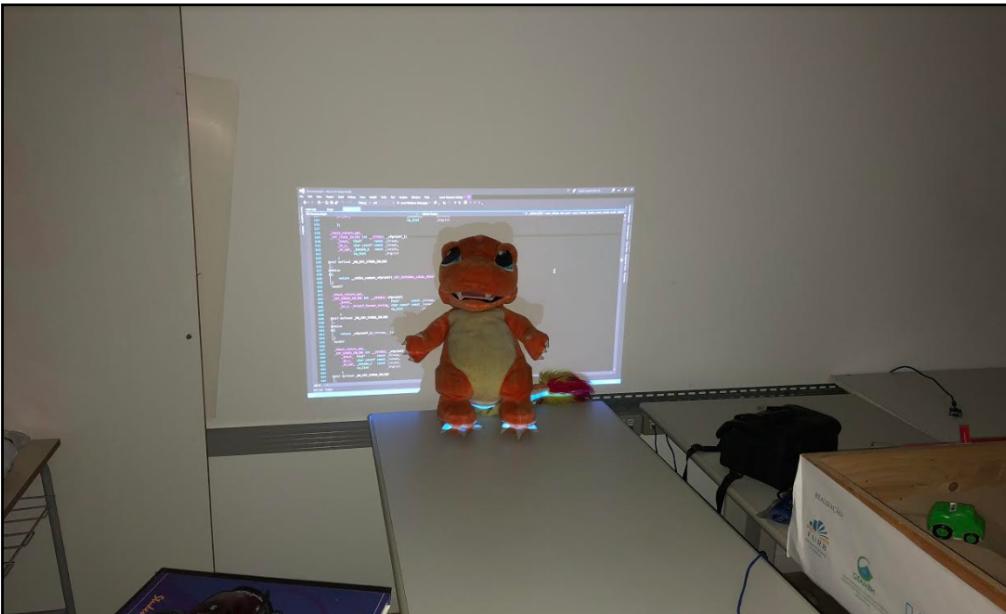
```
1 void generateXYZ(vector<IplImage *> imgsCapturadas, int *matriz, string filePath,
2                               int proj_width, int cam_width, int zScale) {
3
4     std::stringstream stringstream;
5     stringstream << filePath << "\\nuvem_pontos.xyz";
6     std::string stringPath = stringstream.str();
7
8     FILE *f = fopen(stringPath.c_str(), "w");
9     if (f == NULL) {
10        printf("Erro ao abrir %s\nTentando diretorio local...\n", stringPath);
11        FILE *f = fopen("nuvem_pontos.xyz", "w");
12        if (f == NULL) {
13            printf("Erro ao salvar nuvem de pontos!\n");
14            return;
15        }
16    }
17
18    for (int y = 0; y < imgsCapturadas[0]->height; y++) {
19        for (int x = 0; x < imgsCapturadas[0]->width; x++) {
20
21            int u = y * imgsCapturadas[0]->widthStep + x * imgsCapturadas[0]->nChannels;
22
23            if (matriz[u] == -1) { continue; }
24
25            float disp = (float)matriz[u] / (proj_width)-(float)x / cam_width;
26            // printf("%f\n", disp);
27            if (disp == 0.0) { continue; }
28            float z = (float)zScale / (disp);
29
30            fprintf(f, "%d %d %f\n", x, y, z);
31        }
32    }
33
34    fclose(f);
35
36 }
```

0 0 449.999969	0 0 449.999969
1 0 443.076904	1 0 443.076904
2 0 436.363617	2 0 436.363617
3 0 457.142822	3 0 457.142822
4 0 450.000031	4 0 450.000031
5 0 472.131165	5 0 472.131165
6 0 464.516144	6 0 464.516144
7 0 457.142883	7 0 457.142883
8 0 480.000000	8 0 480.000000
9 0 472.131165	9 0 472.131165
10 0 464.516144	10 0 464.516144
11 0 488.135620	11 0 488.135620
...	...

Utilização da biblioteca

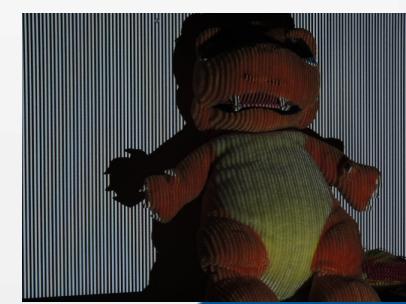
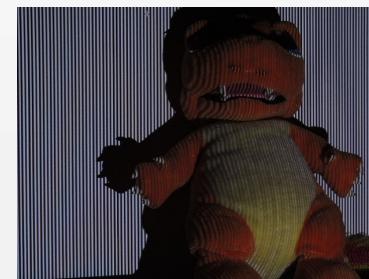
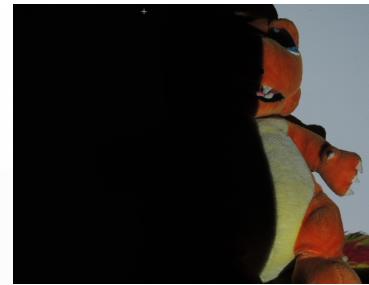
```
1 printf("Projector Width:\n");
2 cin >> proj_width;
3
4 printf("Camera Width:\n");
5 cin >> cam_width;
6
7 printf("Z Scale:\n");
8 cin >> zScale;
9
10 printf("White treshold:\n");
11 cin >> white_treshold;
12
13 printf("Black treshold:\n");
14 cin >> black_treshold;
15
16 capturedImages = getFrames(18, "c:\\temp\\fotos");
17 matriz = reconstruct(white_treshold, black_treshold, capturedImages);
18 generateXYZ(capturedImages, matriz, "C:\\temp", proj_width, cam_width, zScale);
19
```

Resultados

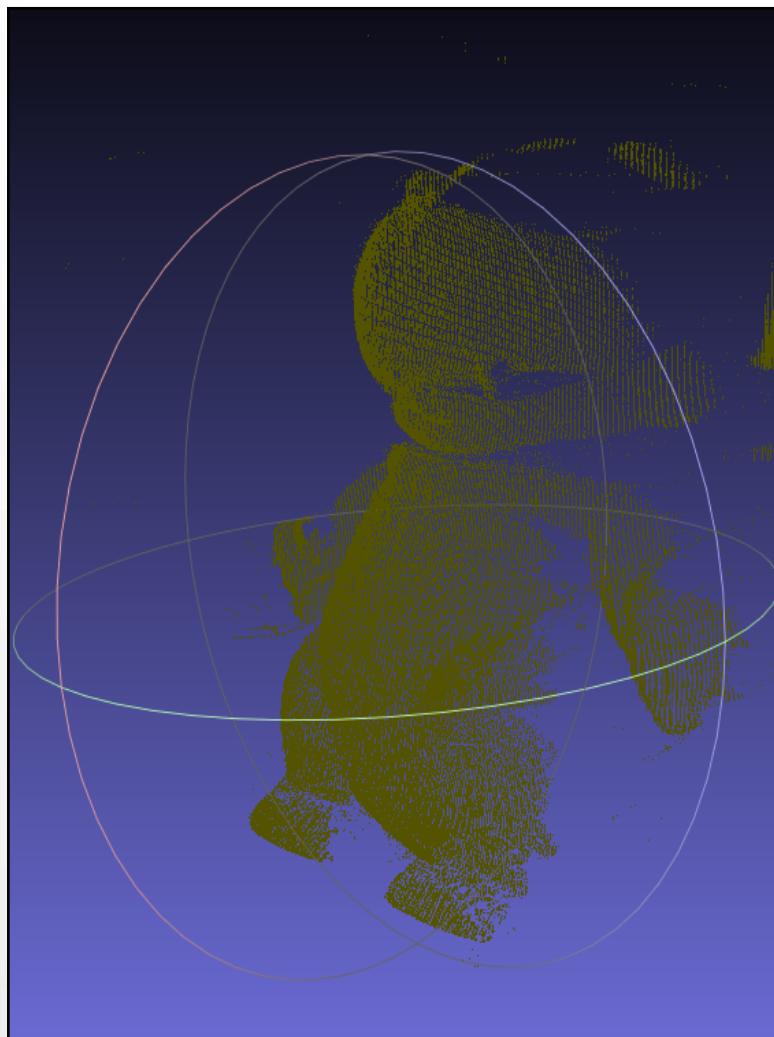


Parâmetros

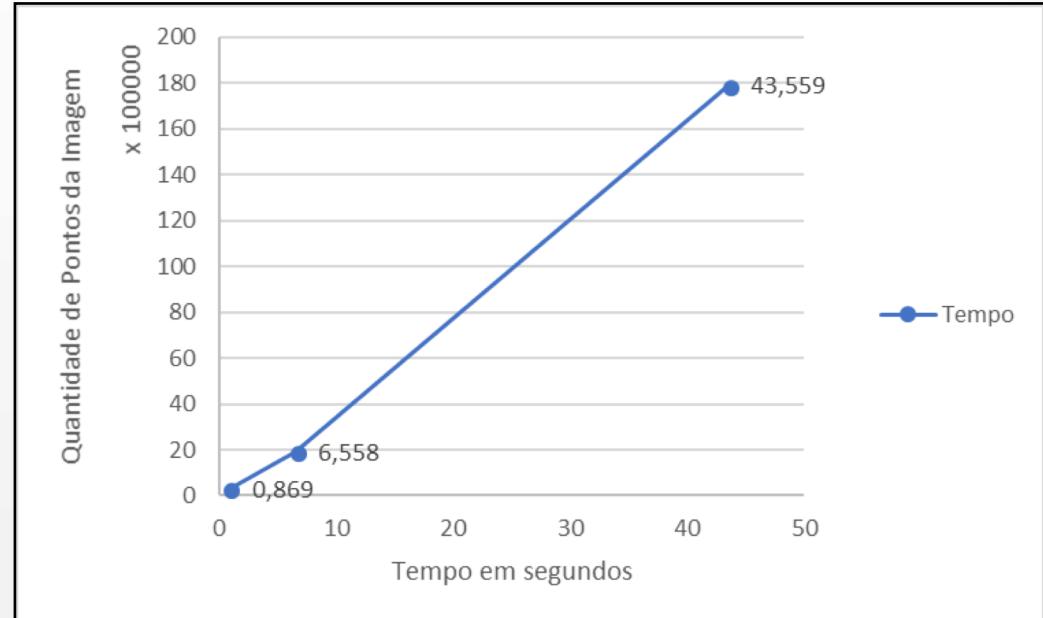
Luminosidade	70 lux
Resolução da câmera	640x480
Zoom da lente	85mm
ISO da câmera	200
zScale	15
white threshold	5
black threshold	-5
projector_width	1280
camera_width	640
número de imagens obtidas (quantidade de padrões usados)	18



Resultados



Resolução da imagem (altura * largura)	Total de pontos	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



Discussões

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 Bruijn	N-ary Codes	Sequência de Gray + deslocamento	Codificação não formal baseada na sequência de De Bruijn	Sequência de Gray
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

Tempo de reconstrução

Storz (2017):
aproximadamente 4 minutos e 30 segundos

Biblioteca atual: 810 milissegundos

310 vezes mais rápido

Conclusões

- Tempo de reconstrução aceitável para reconstrução em tempo real;
- Tempo de captura das imagens não aceitável para tempo real;
- Comunicação com a plataforma Unity é precária;
- Pouca necessidade de configuração de parâmetros.

Extensões

- Alterar o padrão de projeção para imagens únicas;
- Melhorar a qualidade das imagens capturadas pela câmera com o objetivo de melhorar o resultado;
- Remover a dependência do OpenCV;
- Remover o gargalo no processo de captura das imagens;
- Alterar a forma de comunicação com o Unity.

Melhorias na biblioteca para detecção de relevos

Aluno: Lucas Eduardo Schlögl

Orientador: Dalton Solano dos Reis