EstimatePosture

構築: Doxygen 1.8.6

2015 年 12 月 19 日 (土) 20 時 04 分 43 秒

Contents

Chapter 1

クラス索引

1.1 クラス一覧

クラス・構造体・共用体・インターフェースの一覧です。

Chapter 2

ファイル索引

2.1 ファイル一覧

ファイル一覧です。

Chapter 3

クラス詳解

3.1 CoordinateTransform クラス

2D → 3D への座標変換

`#include <EstimatePosture.h>`

公開メンバ関数

- int **ct2Dto3D** (cv::Point **pushpt**, int **points**)

  ***3D* 空間座標へ変換**

3.1.1 詳解

2D → 3D への座標変換

EstimatePosture.h の 84 行目に定義があります。

3.1.2 関数詳解

3.1.2.1 int CoordinateTransform::ct2Dto3D (cv::Point *pushpt,* int *points*)

3D 空間座標へ変換

int CoordinateTransform::ct2dto3d() 関数.2D の座標データを距離データと合わせて 3D データに変換する

引数

pushpt	**Mouse()** (p. 20) 関数でクリックした静止画像の座標 (x,y)

戻り値

0 or -1.0 なら取得した店の距離データが 0 以外であったということ.-1 なら取得した点の距離データが 0 であったということ.-1 なら点を再取得する

CoordinateTransformFunc.cpp の 16 行目に定義があります。

参照先 cnt_getcoordinate, dist, f2, EstimatePosture::LeastSqureMethod(), MAX_POINTS, onedim, pt, pushpt, real_x, real_y, real_z, sumtime, time1th, time2, time2th, time3th (計 16 項目).

参照元 Mouse().

```
17 {
18     EstimatePosture ep; //インスタンス生成
19
20     ofstream ofs("data/RealPoints.dat"); //3 次元座標に変換した時の (x,y,z)
21
22     pt[cnt_getcoordinate] = pushpt; //クリック時に取得した点の座標をコピーする
23
24     onedim[cnt_getcoordinate] = pt[cnt_getcoordinate].x + 640 * pt[
       cnt_getcoordinate].y - 640; //2 次元で表されている点を 1 次元に変換する
25     cout << "pt[" << cnt_getcoordinate << "] => " << pt[cnt_getcoordinate] << " , onedim => " <<
       onedim[cnt_getcoordinate] << " , dist[" << onedim[cnt_getcoordinate] - 1 << "] => " <<
       dist[onedim[cnt_getcoordinate]] << endl;
26
27     //距離データが 0 だった場合座標取得をやり直す
28     if (dist[onedim[cnt_getcoordinate]] == 0){
29         cout << "距離データが 0 だったのでもう一度周辺をクリックしてください" << endl;
30         return -1;
31     }
32
33     WorldCoordinate = NuiTransformDepthImageToSkeleton((long)pt[
       cnt_getcoordinate].x, (long)pt[cnt_getcoordinate].y, dist[onedim[
       cnt_getcoordinate]] << 3, NUI_IMAGE_RESOLUTION_640x480);
34
35     real_x[cnt_getcoordinate] = WorldCoordinate.x*1000.0;
36     real_y[cnt_getcoordinate] = WorldCoordinate.y*1000.0;
37     real_z[cnt_getcoordinate] = WorldCoordinate.z*1000.0;
38
39     cout << "real_x[" << cnt_getcoordinate << "] => " << real_x[
       cnt_getcoordinate] << "[mm] real_y[" << cnt_getcoordinate << "] => " << real_y[
       cnt_getcoordinate] << "[mm] real_z[" << cnt_getcoordinate << "] => " << real_z[
       cnt_getcoordinate] << "[mm]" << endl;
40     cout << "================================================================================" << endl;
41
42     //0~8 までの点が揃ったらそれぞれの 3 次元座標をファイルへ出力し最小二乗法を行う
43     if (points == MAX_POINTS - 1){
44         for (int i = 0; i < MAX_POINTS; i++){
45             ofs << real_x[i] << " " << real_y[i] << " " << real_z[i] << endl;
46         }
47         //処理時間を計測する
48         f2 = 1000.0 / cv::getTickFrequency();
49         time2 = cv::getTickCount();
50         time2th = 0;
51
52         ep.LeastSqureMethod(real_x, real_y, real_z); //最小二乗法の要素計算
53
54         time2th = (cv::getTickCount() - time2)*f2;
55
56         cout << MAX_POINTS << "点目をクリックしてから回転角度を求めるまでの時間" << time2th << "[ms]" << endl;
57
58         sumtime = time1th + time2th - time3th;
59
60         cout << "処理にかかった時間 (gnuplot の処理時間を除く) は" << sumtime << "[ms] でした" << "\n" << endl;
61
62         cout << "============== メニュー ==============" << endl;
63         cout << "もう一度計測しますか??" << endl;
64         cout << "画像上でキーを入力してください" << endl;
65         cout << "1 : Yes" << endl;
66         cout << "2 : No" << endl;
67         cout << "====================================" << "\n" << endl;
68     }
69
70     cnt_getcoordinate++; //取得した点の数が 9 つになるまで座標を取得し続ける
71
72     return 0;
73 }
```

このクラス詳解は次のファイルから抽出されました:

- **EstimatePosture.h**
- **CoordinateTransformFunc.cpp**

3.2 EstimatePosture クラス

最小二乗法によって角度を計算するクラス

`#include <EstimatePosture.h>`

公開メンバ関数

- **EstimatePosture** ()

  EstimatePosture クラスのコンストラクタ
- void **LeastSqureMethod** (double **real_x**[**MAX_POINTS**-1], double **real_y**[**MAX_POINTS**-1], double **real_z**[**MAX_POINTS**-1])

  最小二乗法
- void **CalcYawRollPitch** (double **a**, double **b**, double **c**)

  回転角度の計算

公開変数類

- double **a**
- double **b**
- double **c**

  最小二乗法で求めた平面の係数 *a,b,c*

3.2.1 詳解

最小二乗法によって角度を計算するクラス

EstimatePosture.h の 97 行目に定義があります。

3.2.2 構築子と解体子

3.2.2.1 EstimatePosture::EstimatePosture ( )

EstimatePosture クラスのコンストラクタ

void **EstimatePosture::EstimatePosture()** (p. 7).EstimatePosture クラスのコンストラクタ

EstimatePostureFunc.cpp の 13 行目に定義があります。

参照先 a, b, c.

```
14 {
15     //最小二乗法に必要な要素の初期化
16     Szz = 0, Sxz = 0, Syz = 0, Sz = 0, Sxx = 0, Sxy = 0, Sx = 0, Syy = 0, Sy = 0;
17
18     //最小二乗法によって求めた係数 a,b,c の初期化
19     a = 0, b = 0, c = 0;
20 }
```

3.2.3 関数詳解

3.2.3.1 void EstimatePosture::CalcYawRollPitch (double *a*, double *b*, double *c*)

回転角度の計算

void **EstimatePosture::CalcYawRollPitch()** (p. 7) 関数.回転角度を計算する

引数

a	最小二乗法によって求めた平面の式 aX+bY+c=Z の係数
b	最小二乗法によって求めた平面の式 aX+bY+c=Z の係数

c	最小二乗法によって求めた平面の式 aX+bY+c=Z の係数

戻り値

EstimatePostureFunc.cpp の 76 行目に定義があります。

参照先 PI.

参照元 LeastSqureMethod().

```
77 {
78     char input = ' ';
79
80     yaw_rad = -atan(a);
81     yaw_deg = yaw_rad / PI*180.0;
82
83     roll_rad = atan2(-a, b);
84     roll_deg = roll_rad / PI*180.0;
85
86     pitch_rad = atan2(1, b);
87     pitch_deg = pitch_rad / PI*180.0;
88
89     cout << "\n";
90     cout << "====== 姿勢計測結果 ======" << endl;
91     cout << "Yaw : " << yaw_deg << "[deg]" << endl;
92     cout << "Roll : " << roll_deg << "[deg]" << endl;
93     cout << "Pitch : " << pitch_deg << "[deg]" << endl;
94     cout << "==========================" << "\n" << endl;
95 }
```

3.2.3.2 void EstimatePosture::LeastSqureMethod (double *real_x[MAX_POINTS-1],* double *real_y[MAX_POINTS-1],* double *real_z[MAX_POINTS-1]*)

最小二乗法

void **EstimatePosture::LeastSqureMethod()** (p. 8) 関数.最小二乗法の要素を計算する

引数

pushpt	**Mouse()** (p. 20) 関数でクリックした静止画像の座標 (x,y)

戻り値

0 or -1.0 なら取得した店の距離データが 0 以外であったということ.-1 なら取得した点の距離データが 0 であったということ.-1 なら点を再取得する

EstimatePostureFunc.cpp の 27 行目に定義があります。

参照先 a, b, c, CalcYawRollPitch(), f3, MAX_POINTS, Plot::PlotData(), real_x, real_y, real_z, time3, time3th.

参照元 CoordinateTransform::ct2Dto3D().

```
28 {
29     Plot pl; //Plot クラスのインスタンスを生成
30
31     ofstream ofs("data/abc.dat");
32
33     for (int i = 0; i < MAX_POINTS; i++){
34         Szz = Szz + real_z[i] * real_z[i];
35         Sxz = Sxz + real_x[i] * real_z[i];
36         Syz = Syz + real_y[i] * real_z[i];
37         Sz = Sz + real_z[i];
38         Sxx = Sxx + real_x[i] * real_x[i];
39         Sxy = Sxy + real_x[i] * real_y[i];
40         Sx = Sx + real_x[i];
41         Syy = Syy + real_y[i] * real_y[i];
42         Sy = Sy + real_y[i];
43     }
44
```

```
45      m1 = (cv::Mat_<double>(3, 3) << Sxx, Sxy, Sx, Sxy, Syy, Sy, Sx, Sy, MAX_POINTS);
46      m2 = (cv::Mat_<double>(3, 1) << Sxz, Syz, Sz);
47      A = m1.inv()*m2;
48
49      a = A.at<double>(0, 0);
50      b = A.at<double>(0, 1);
51      c = A.at<double>(0, 2);
52
53      ofs << "a = " << a << "\n" << "b = " << b << "\n" << "c = " << c << endl;
54
55      f3 = 1000.0 / cv::getTickFrequency();
56      time3 = cv::getTickCount();
57      time3th = 0;
58
59      pl.PlotData(a, b, c); //データをプロットする．必要ない場合はコメントアウトする
60
61      time3th = (cv::getTickCount() - time3)*f3;
62
63      cout << "\n";
64      cout << "gnuplot が作業を行っている時間" << time3th << "[ms]" << endl;
65
66      CalcYawRollPitch(a, b, c);
67  }
```

3.2.4 メンバ詳解

3.2.4.1 double EstimatePosture::a

EstimatePosture.h の 114 行目に定義があります。

参照元 EstimatePosture(), LeastSqureMethod().

3.2.4.2 double EstimatePosture::b

EstimatePosture.h の 114 行目に定義があります。

参照元 EstimatePosture(), LeastSqureMethod().

3.2.4.3 double EstimatePosture::c

最小二乗法で求めた平面の係数 a,b,c

EstimatePosture.h の 114 行目に定義があります。

参照元 EstimatePosture(), LeastSqureMethod().

このクラス詳解は次のファイルから抽出されました:

- **EstimatePosture.h**
- **EstimatePostureFunc.cpp**

3.3 Kinect クラス

Kinect 操作

`#include <EstimatePosture.h>`

公開メンバ関数

- ~**Kinect** ()
  デストラクタ
- void **initialize** ()
  Kinect の初期化

- void **run** ()

  Kinect による処理
- void **drawRGBImage** (cv::Mat &image)

  RGB 画像の描画
- void **drawDepthImage** (cv::Mat &image)

  Depth 画像の描画
- void **GetDepthData** (cv::Mat &image)

  Depth データの取得

3.3.1 詳解

Kinect 操作

EstimatePosture.h の 59 行目に定義があります。

3.3.2 構築子と解体子

3.3.2.1 Kinect::~Kinect ( )

デストラクタ

Kinect::~Kinect() (p. 10) 関数.デストラクタ

引数

KinectFunc.cpp の 14 行目に定義があります。

```
15 {
16     /* 終了処理 */
17     if (kinect != 0){
18         kinect->NuiShutdown(); //Kinect の終了処理
19         kinect->Release(); //インスタンスの解放
20     }
21 }
```

3.3.3 関数詳解

3.3.3.1 void Kinect::drawDepthImage (cv::Mat & *image*)

Depth 画像の描画

void **Kinect::drawDepthImage()** (p. 10) 関数.距離データを取得する

引数

image	Kinect が取得した画像

戻り値

KinectFunc.cpp の 103 行目に定義があります。

参照先 CAMERA_RESOLUTION, ERROR_CHECK.

参照元 run().

```
104 {
105     /* 距離画像準備. 距離画像がほしい場合のみ利用する */
106     //image = cv::Mat(height,width,CV_8UC1,cv::Scalar(0));
107 
108     /* 距離カメラのフレームデータを取得する */
109     NUI_IMAGE_FRAME depthFrame = { 0 }; //すべてのフレームを 0 で初期化
110     ERROR_CHECK(kinect->NuiImageStreamGetNextFrame(depthStreamHandle, 0, &depthFrame));
111 
112     /* 距離データを取得する */
113     NUI_LOCKED_RECT depthData = { 0 };
114     depthFrame.pFrameTexture->LockRect(0, &depthData, 0, 0);
115 
116     USHORT* depth = (USHORT*)depthData.pBits;
117 
118     for (unsigned int i = 0; i < (depthData.size / sizeof(USHORT)); ++i){
119         USHORT distance = ::NuiDepthPixelToDepth(depth[i]);
120 
121         LONG depthX = i%width;
122         LONG depthY = i / width;
123 
124         LONG colorX = depthX;
125         LONG colorY = depthY;
126 
127         //image.at<UCHAR>(colorY,colorX) = distance/8192.0*255.0;
128 
129         kinect->NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(
      CAMERA_RESOLUTION, CAMERA_RESOLUTION, 0, depthX, depthY, depth[i], &colorX, &colorY);
130     }
131     ERROR_CHECK(kinect->NuiImageStreamReleaseFrame(depthStreamHandle, &depthFrame));
132 }
```

3.3.3.2 void Kinect::drawRGBImage (cv::Mat & *image*)

RGB 画像の描画

void **Kinect::drawRGBImage()** (p. 11) 関数.RGB データを取得する

引数

image	Kinect が取得した画像

戻り値

KinectFunc.cpp の 81 行目に定義があります。

参照先 ERROR_CHECK.

参照元 run().

```
82 {
83     /* RGB カメラのフレームデータを取得する */
84     NUI_IMAGE_FRAME imageFrame = { 0 }; //すべてのフレームを 0 で初期化
85     ERROR_CHECK(kinect->NuiImageStreamGetNextFrame(imageStreamHandle, 0, &imageFrame)); //新しいフレームを取得
する
86 
87     /* フレームの画像データを取得する */
88     NUI_LOCKED_RECT colorData;
89     imageFrame.pFrameTexture->LockRect(0, &colorData, 0, 0);
90 
91     /* フレームの画像データを取得する */
92     image = cv::Mat(height, width, CV_8UC4, colorData.pBits);
93 
94     /* フレームデータを解放する */
95     ERROR_CHECK(kinect->NuiImageStreamReleaseFrame(imageStreamHandle, &imageFrame));
96 }
```

3.3.3.3 void Kinect::GetDepthData (cv::Mat & *image*)

Depth データの取得

void **Kinect::GetDepthData()** (p. 11) 関数.depthdata を dist[] へ読み込む.(x,y,z) のデータを作る

引数

image	Kinect が取得した画像

戻り値

KinectFunc.cpp の 139 行目に定義があります。

参照先 dist, ERROR_CHECK.

参照元 run().

```
140 {
141     FILE *fp1, *fp2; //ファイルポインタ
142     errno_t error1, error2; //errno_t 型の変数
143 
144     if (error1 = fopen_s(&fp1, "data/depthdata.txt", "w") != 0){
145         throw runtime_error("depthdata ファイルが開けません");
146     }
147 
148     if (error2 = fopen_s(&fp2, "data/xyz.txt", "w") != 0){
149         throw runtime_error("xyz.txt が開けません");
150     }
151 
152     /* 距離カメラのフレームデータを取得する */
153     NUI_IMAGE_FRAME depthFrame = { 0 }; //すべてのフレームを 0 で初期化
154     ERROR_CHECK(kinect->NuiImageStreamGetNextFrame(depthStreamHandle, 0, &depthFrame));
155 
156     /* 距離データを取得する */
157     NUI_LOCKED_RECT depthData = { 0 };
158     depthFrame.pFrameTexture->LockRect(0, &depthData, 0, 0);
159 
160     USHORT* depth = (USHORT*)depthData.pBits;
161 
162     for (unsigned int i = 0; i < (depthData.size / sizeof(USHORT)); ++i){
163         USHORT distance = ::NuiDepthPixelToDepth(depth[i]);
164 
165         fprintf(fp1, "%d\n", distance);
166         dist[i] = distance; //距離データを変数 dist[i] に格納
167     }
168 
169     /* (x,y,z) の組を作る */
170     int d = 0;
171     for (int y = 0; y < 480; y++){
172         for (int x = 0; x < 640; x++){
173             fprintf(fp2, "%d %d %d\n",x, y, dist[d]);
174             d++;
175         }
176     }
177 
178     fclose(fp1);
179     fclose(fp2);
180 
181     ERROR_CHECK(kinect->NuiImageStreamReleaseFrame(depthStreamHandle, &depthFrame));
182 }
```

3.3.3.4 void Kinect::initialize ( )

Kinect の初期化

void **Kinect::initialize()** (p. 12) 関数.Kinect の初期化

引数

@return	

KinectFunc.cpp の 52 行目に定義があります。

参照先 CAMERA_RESOLUTION, ERROR_CHECK.

参照元 main().

```
53 {
```

```
54      createInstance(); //createInstance() を呼び出す
55
56      /* Kinect の設定を初期化する */
57      ERROR_CHECK(kinect->NuiInitialize(NUI_INITIALIZE_FLAG_USES_COLOR | NUI_INITIALIZE_FLAG_USES_DEPTH)); //
    RGB-D カメラを扱うためのフラグ追加
58
59      /* RGB カメラの初期化 */
60      ERROR_CHECK(kinect->NuiImageStreamOpen(NUI_IMAGE_TYPE_COLOR,
    CAMERA_RESOLUTION, 0, 2, 0, &imageStreamHandle));
61
62      /* 距離カメラの初期化 */
63      ERROR_CHECK(kinect->NuiImageStreamOpen(NUI_IMAGE_TYPE_DEPTH,
    CAMERA_RESOLUTION, 0, 2, 0, &depthStreamHandle));
64
65      /* Near モード．使用しない場合はコメントアウトする */
66      ERROR_CHECK(kinect->NuiImageStreamSetImageFrameFlags(depthStreamHandle,
    NUI_IMAGE_STREAM_FLAG_ENABLE_NEAR_MODE));
67
68      /* フレーム更新のイベントハンドルを作成する */
69      streamEvent = ::CreateEvent(0, TRUE, FALSE, 0);
70      ERROR_CHECK(kinect->NuiSetFrameEndEvent(streamEvent, 0));
71
72      /* 指定した解像度の画面サイズを取得する */
73      ::NuiImageResolutionToSize(CAMERA_RESOLUTION, width, height);
74  }
```

3.3.3.5 void Kinect::run ( )

Kinect による処理

void **Kinect::run()** (p. 13) 関数.Kinect の処理を実行する

引数

@return	

KinectFunc.cpp の 189 行目に定義があります。

参照先 cnt_getcoordinate, drawDepthImage(), drawRGBImage(), f1, GetDepthData(), MAX_POINTS, Mouse(), points, sumtime, time1, time1th.

参照元 main().

```
190 {
191     cv::Mat image; //Kinect が取得した画像
192
193     cout << "============== メニュー ==============" << endl;
194     cout << "1：今写っている平面を計測する" << endl;
195     cout << "2：計測を終了する" << endl;
196     cout << "(計測の途中で終了することもできます)" << endl;
197     cout << "======================================" << endl << endl;
198
199     //メインループ
200     while (1){
201         /* データの更新を待つ */
202         DWORD ret = ::WaitForSingleObject(streamEvent, INFINITE);
203         ::ResetEvent(streamEvent); //次のイベントに備えてリセット
204
205         drawRGBImage(image); //RGB データの取得
206
207         drawDepthImage(image); //Depth データの取得
208
209         cv::namedWindow("動画像", CV_WINDOW_AUTOSIZE | CV_WINDOW_FREERATIO); //動画像が表示されるウインドウを定義
210         cv::imshow("動画像", image); //動画像を表示
211
212         int key = cv::waitKey(1); //動画像を表示し続ける
213
214         if (key == '2'){
215             cout << "終了します" << endl;
216             break;
217         }
218         else if (key == '1'){
219             sumtime = 0; //処理にかかる合計時間の初期化
220             f1 = 1000.0 / cv::getTickFrequency();
221             time1 = cv::getTickCount();
222             time1th = 0;
223
224             points = 0; //取得した点の数を初期化
225             cnt_getcoordinate = 0; //座標を取得した回数を初期化
226
```

```
227             cv::namedWindow("静止画像", CV_WINDOW_AUTOSIZE | CV_WINDOW_FREERATIO); //静止画像が表示されるウインドウを定義
228
229             cout << MAX_POINTS << "回クリックしてください" << endl;
230
231             cv::setMouseCallback("静止画像", Mouse); //マウスコールバック関数の設定
232
233             cv::imwrite("plot_img/rgbimage.bmp", image); //取得した静止画像を保存
234             cv::imshow("静止画像", image); //取得した静止画像を表示
235
236             time1th = (cv::getTickCount() - time1)*f1; //処理にかかった時間を計測
237             cout << "\"1\"キーを入力してから表示するまでの時間" << time1th << "[ms]" << endl << endl;
238
239             cout << "============================ 取得した点の座標 ============================" << endl;
240
241             GetDepthData(image); //距離データを取得
242         }
243     }
244 }
```

このクラス詳解は次のファイルから抽出されました:

- **EstimatePosture.h**
- **KinectFunc.cpp**

3.4 Plot クラス

gnuplot による画像を表示、保存する

`#include <EstimatePosture.h>`

公開メンバ関数

- void **PlotData** (double a, double b, double c)
  取得したデータをプロットする関数

3.4.1 詳解

gnuplot による画像を表示、保存する

EstimatePosture.h の 125 行目に定義があります。

3.4.2 関数詳解

3.4.2.1 void Plot::PlotData (double *a*, double *b*, double *c*)

取得したデータをプロットする関数

void **Plot::PlotData(double a,double b,double c)** (p. 14) 関数.取得したデータをプロットする

引数

a	最小二乗法によって求めた平面の式 aX+bY+c=Z の係数
b	最小二乗法によって求めた平面の式 aX+bY+c=Z の係数
c	最小二乗法によって求めた平面の式 aX+bY+c=Z の係数

戻り値

PlotFunc.cpp の 17 行目に定義があります。

参照元 EstimatePosture::LeastSqureMethod().

```
18 {
19     gnuplot = _popen("gnuplot", "w");
20     if (gnuplot == NULL){
21         cout << "gnuplot が開けません" << endl;
22     }
23
24     //a,b,c の値を送信
25     fprintf_s(gnuplot, "a = %f\n", a);
26     fprintf_s(gnuplot, "b = %f\n", b);
27     fprintf_s(gnuplot, "c = %f\n", c);
28
29     //それぞれの軸ラベルを付ける
30     fputs("set xlabel \"X-axis\"\n", gnuplot);
31     fputs("set ylabel \"Y-axis\"\n", gnuplot);
32     fputs("set zlabel \"Z-axis\"\n", gnuplot);
33
34     //クリックした 3D 座標と最小二乗法によって求めた平面をプロットする
35     fputs("splot \"data/RealPoints.dat\",a*x+b*y+c\n", gnuplot);
36
37     //デフォルトで表示された画像を保存する
38     fputs("set title \"Default\"\n", gnuplot);
39     fputs("set term jpeg\n", gnuplot);
40     fputs("set output \"plot_img/default.jpg\"\n", gnuplot);
41     fputs("rep\n", gnuplot);
42
43     //X-Y 平面でプロット.Z 軸と Z 軸のラベルを消す
44     fputs("unset zlabel\n", gnuplot);
45     fputs("unset ztics\n", gnuplot);
46     fputs("rep\n", gnuplot);
47
48     fputs("set view 0,0,1,1\n", gnuplot);
49     fputs("set title \"X-Y Plane\"\n", gnuplot);
50     fputs("set term jpeg\n", gnuplot);
51     fputs("set output \"plot_img/X-Y_Plane.jpg\"\n", gnuplot);
52     fputs("rep\n", gnuplot);
53
54     //X-Z 平面でプロット.Z 軸と Z 軸のラベル表示し、Y 軸と Y 軸のラベルを消す
55     fputs("set zlabel \"Z-axis\"\n", gnuplot);
56     fputs("set ztics\n", gnuplot);
57     fputs("unset ylabel\n", gnuplot);
58     fputs("unset ytics\n", gnuplot);
59     fputs("rep\n", gnuplot);
60
61     fputs("set view 90,0,1,1\n", gnuplot);
62     fputs("set title \"X-Z Plane\"\n", gnuplot);
63     fputs("set term jpeg\n", gnuplot);
64     fputs("set output \"plot_img/X-Z_Plane.jpg\"\n", gnuplot);
65     fputs("rep\n", gnuplot);
66
67     //X-Y 平面でプロット.Y 軸と Y 軸のラベルを表示し、X 軸と X 軸のラベルを消す
68     fputs("set ylabel \"Y-axis\"\n", gnuplot);
69     fputs("unset ytics\n", gnuplot);
70     fputs("unset xlabel\n", gnuplot);
71     fputs("unset xtics\n", gnuplot);
72     fputs("rep\n", gnuplot);
73
74     fputs("set view 0,0,1,1\n", gnuplot);
75     fputs("set title \"Y-Z Plane\"\n", gnuplot);
76     fputs("set term jpeg\n", gnuplot);
77     fputs("set output \"plot_img/Y-Z_Plane.jpg\"\n", gnuplot);
78     fputs("rep\n", gnuplot);
79
80     //ここまでの処理を gnuplot へ流し閉じる
81     fflush(gnuplot);
82     _pclose(gnuplot);
83 }
```

このクラス詳解は次のファイルから抽出されました:

- **EstimatePosture.h**
- **PlotFunc.cpp**

Chapter 4

ファイル詳解

4.1 CoordinateTransformFunc.cpp ファイル

CoordinateTransform クラスの実装ファイル

```
#include "EstimatePosture.h"
```

4.1.1 詳解

CoordinateTransform クラスの実装ファイル

日付

2014/03/26

著者

H.Shigehara

CoordinateTransformFunc.cpp に定義があります。

4.2 EstimatePosture.h ファイル

プロジェクトのヘッダファイル

```
#include <Windows.h>
#include <NuiApi.h>
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <direct.h>
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>
```

クラス

- class **Kinect**
  Kinect 操作
- class **CoordinateTransform**
  2D → 3D への座標変換
- class **EstimatePosture**
  最小二乗法によって角度を計算するクラス
- class **Plot**
  gnuplot による画像を表示、保存する

マクロ定義

- #define **MAX_POINTS** 9
  マウスクリック時に取得する点の数
- #define **PI** 3.14159265358979323846264338327 95
  円周率 π
- #define **ERROR_CHECK**(ret)
  エラーチェック

関数

- void **Mouse** (int event, int x, int y, int flags, void ∗param)
  cMouse::Mouse() 関数.コンストラクタ

変数

- unsigned short **dist** [307200]
  距離データを格納する配列
- cv::Point **pushpt** [**MAX_POINTS**]
  取得した点の座標
- int **points**
  取得した点の数
- cv::Point **pt** [**MAX_POINTS**]
  取得した座標のコピー
- int **cnt_getcoordinate**
  座標を取得した回数をカウント
- int **onedim** [**MAX_POINTS**]
  それぞれの天を一次元に直す
- double **real_x** [**MAX_POINTS**]
  3D 空間座標の*X*
- double **real_y** [**MAX_POINTS**]
  3D 空間座標の*Y*
- double **real_z** [**MAX_POINTS**]
  3D 空間座標の*Z*
- double **f1**
- double **f2**
- double **f3**
- int64 **time1**
- int64 **time2**
- int64 **time3**

- double **time1th**
  *"p"*キーを入力してから表示するまでの時間
- double **time2th**
  9 点目をクリックしてから回転角度を求めるまでの時間
- double **time3th**
  gnuplot が作業を行っている時間
- double **sumtime**
  処理にかかった時間の合計
- const NUI_IMAGE_RESOLUTION **CAMERA_RESOLUTION** = NUI_IMAGE_RESOLUTION_640x480
  Kinect の解像度

4.2.1 詳解

プロジェクトのヘッダファイル

日付

2014/03/26

著者

H.Shigehara

EstimatePosture.h に定義があります。

4.2.2 マクロ定義詳解

4.2.2.1 #define ERROR_CHECK(*ret*)

値:

```
if (ret != S_OK){ \
    stringstream ss; \
    ss << "faild " #ret " " << hex << ret << endl; \
    throw runtime_error(ss.str().c_str()); \
}
```

エラーチェック

EstimatePosture.h の 135 行目に定義があります。

参照元 Kinect::drawDepthImage(), Kinect::drawRGBImage(), Kinect::GetDepthData(), Kinect::initialize().

4.2.2.2 #define MAX_POINTS 9

マウスクリック時に取得する点の数

EstimatePosture.h の 29 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod(), Mouse(), Kinect::run().

4.2.2.3 #define PI 3.14159265358979323846264338327 95

円周率π

EstimatePosture.h の 30 行目に定義があります。

参照元 EstimatePosture::CalcYawRollPitch().

4.2.3 関数詳解

4.2.3.1 void Mouse (int *event,* int *x,* int *y,* int *flags,* void * *param*)

cMouse::Mouse() 関数.コンストラクタ

引数

Mouse.cpp の 14 行目に定義があります。

参照先 CoordinateTransform::ct2Dto3D(), MAX_POINTS, points, pushpt.

参照元 Kinect::run().

```
15 {
16     CoordinateTransform ct; //CoordinateTransform クラスの S インスタンスの生成
17
18     int check = 0; //距離データが 0 でないかチェックするフラグ.check == 0 なら距離データが 0 以外を表す.
19
20     if (event == CV_EVENT_LBUTTONDOWN){
21         if (points < MAX_POINTS){
22             pushpt[points] = cv::Point(x, y);
23             check = ct.ct2Dto3D(pushpt[points], points);
24             if (check == 0){
25                 points++;
26             }
27         }
28     }
29 }
```

4.2.4 変数詳解

4.2.4.1 const NUI_IMAGE_RESOLUTION CAMERA_RESOLUTION = NUI_IMAGE_RESOLUTION_640x480

Kinect の解像度

EstimatePosture.h の 143 行目に定義があります。

参照元 Kinect::drawDepthImage(), Kinect::initialize().

4.2.4.2 int cnt_getcoordinate

座標を取得した回数をカウント

main.cpp の 17 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::run().

4.2.4.3 unsigned short dist[307200]

距離データを格納する配列

main.cpp の 12 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::GetDepthData().

4.2.4.4 double f1

main.cpp の 23 行目に定義があります。

参照元 Kinect::run().

4.2.4.5 double f2

main.cpp の 23 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D().

4.2.4.6 double f3

main.cpp の 23 行目に定義があります。
参照元 EstimatePosture::LeastSqureMethod().

4.2.4.7 int onedim[MAX_POINTS]

それぞれの天を一次元に直す
main.cpp の 18 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D().

4.2.4.8 int points

取得した点の数
main.cpp の 15 行目に定義があります。
参照元 Mouse(), Kinect::run().

4.2.4.9 cv::Point pt[MAX_POINTS]

取得した座標のコピー
main.cpp の 16 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D().

4.2.4.10 cv::Point pushpt[MAX_POINTS]

取得した点の座標
main.cpp の 14 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D(), Mouse().

4.2.4.11 double real_x[MAX_POINTS]

3D 空間座標のX
main.cpp の 19 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.2.4.12 double real_y[MAX_POINTS]

3D 空間座標のY
main.cpp の 20 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.2.4.13 double real_z[MAX_POINTS]

3D 空間座標のZ

main.cpp の 21 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.2.4.14 double sumtime

処理にかかった時間の合計

main.cpp の 28 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::run().

4.2.4.15 int64 time1

main.cpp の 24 行目に定義があります。

参照元 Kinect::run().

4.2.4.16 double time1th

"p"キーを入力してから表示するまでの時間

main.cpp の 25 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::run().

4.2.4.17 int64 time2

main.cpp の 24 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D().

4.2.4.18 double time2th

9 点目をクリックしてから回転角度を求めるまでの時間

main.cpp の 26 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D().

4.2.4.19 int64 time3

main.cpp の 24 行目に定義があります。

参照元 EstimatePosture::LeastSqureMethod().

4.2.4.20 double time3th

gnuplot が作業を行っている時間

main.cpp の 27 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.3 EstimatePostureFunc.cpp ファイル

EstimatePosture クラスの実装ファイル

```
#include "EstimatePosture.h"
```

4.3.1 詳解

EstimatePosture クラスの実装ファイル

日付

2014/03/26

著者

H.Shigehara

EstimatePostureFunc.cpp に定義があります。

4.4 KinectFunc.cpp ファイル

Kinect クラスの関数の実装ファイル

```
#include "EstimatePosture.h"
```

4.4.1 詳解

Kinect クラスの関数の実装ファイル

日付

2014/03/26

著者

H.Shigehara

KinectFunc.cpp に定義があります。

4.5 main.cpp ファイル

クラスを使う側のファイル

```
#include "EstimatePosture.h"
```

関数

- void **Mouse** ()
- int **main** ()

変数

- unsigned short **dist** [307200]

  距離データを格納する配列
- cv::Point **pushpt** [**MAX_POINTS**]

  取得した点の座標
- int **points**

  取得した点の数
- cv::Point **pt** [**MAX_POINTS**]

  取得した座標のコピー
- int **cnt_getcoordinate**

  座標を取得した回数をカウント
- int **onedim** [**MAX_POINTS**]

  それぞれの天を一次元に直す
- double **real_x** [**MAX_POINTS**]

  3D 空間座標の*X*
- double **real_y** [**MAX_POINTS**]

  3D 空間座標の*Y*
- double **real_z** [**MAX_POINTS**]

  3D 空間座標の*Z*
- double **f1**
- double **f2**
- double **f3**
- int64 **time1**
- int64 **time2**
- int64 **time3**
- double **time1th**

  *"p"*キーを入力してから表示するまでの時間
- double **time2th**

  9 点目をクリックしてから回転角度を求めるまでの時間
- double **time3th**

  gnuplot が作業を行っている時間
- double **sumtime**

  処理にかかった時間の合計

4.5.1 詳解

クラスを使う側のファイル

日付

2014/03/26

著者

H.Shigehara

main.cpp に定義があります。

4.5.2 関数詳解

4.5.2.1 int main ( )

main.cpp の 32 行目に定義があります。

参照先 Kinect::initialize(), Kinect::run().

```
33 {
34     try{
35         Kinect kinect; //Kinect クラスのインスタンス kinect を生成
36 
37         kinect.initialize();
38 
39         _mkdir("data");
40         _mkdir("plot_img");
41 
42         kinect.run();
43 
44         exit(0);
45     }
46     catch (exception& ex){
47         cout << ex.what() << endl;
48     }
49     return 0;
50 }
```

4.5.2.2 void Mouse ( )

4.5.3 変数詳解

4.5.3.1 int cnt_getcoordinate

座標を取得した回数をカウント

main.cpp の 17 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::run().

4.5.3.2 unsigned short dist[307200]

距離データを格納する配列

main.cpp の 12 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::GetDepthData().

4.5.3.3 double f1

main.cpp の 23 行目に定義があります。

参照元 Kinect::run().

4.5.3.4 double f2

main.cpp の 23 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D().

4.5.3.5 double f3

main.cpp の 23 行目に定義があります。

参照元 EstimatePosture::LeastSqureMethod().

4.5.3.6 int onedim[MAX_POINTS]

それぞれの天を一次元に直す

main.cpp の 18 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D().

4.5.3.7 int points

取得した点の数

main.cpp の 15 行目に定義があります。

参照元 Mouse(), Kinect::run().

4.5.3.8 cv::Point pt[MAX_POINTS]

取得した座標のコピー

main.cpp の 16 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D().

4.5.3.9 cv::Point pushpt[MAX_POINTS]

取得した点の座標

main.cpp の 14 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Mouse().

4.5.3.10 double real_x[MAX_POINTS]

3D 空間座標のX

main.cpp の 19 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.5.3.11 double real_y[MAX_POINTS]

3D 空間座標のY

main.cpp の 20 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.5.3.12 double real_z[MAX_POINTS]

3D 空間座標のZ

main.cpp の 21 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.5.3.13 double sumtime

処理にかかった時間の合計

main.cpp の 28 行目に定義があります。

参照元 CoordinateTransform::ct2Dto3D(), Kinect::run().

4.5.3.14 int64 time1

main.cpp の 24 行目に定義があります。
参照元 Kinect::run().

4.5.3.15 double time1th

"p"キーを入力してから表示するまでの時間
main.cpp の 25 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D(), Kinect::run().

4.5.3.16 int64 time2

main.cpp の 24 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D().

4.5.3.17 double time2th

9 点目をクリックしてから回転角度を求めるまでの時間
main.cpp の 26 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D().

4.5.3.18 int64 time3

main.cpp の 24 行目に定義があります。
参照元 EstimatePosture::LeastSqureMethod().

4.5.3.19 double time3th

gnuplot が作業を行っている時間
main.cpp の 27 行目に定義があります。
参照元 CoordinateTransform::ct2Dto3D(), EstimatePosture::LeastSqureMethod().

4.6 Mouse.cpp ファイル

```
#include "EstimatePosture.h"
```

関数

- void **Mouse** (int event, int x, int y, int flags, void *param)
  cMouse::Mouse() 関数.コンストラクタ

4.6.1 関数詳解

4.6.1.1 void Mouse (int *event*, int *x*, int *y*, int *flags*, void * *param*)

cMouse::Mouse() 関数.コンストラクタ

引数

Mouse.cpp の 14 行目に定義があります。

参照先 CoordinateTransform::ct2Dto3D(), MAX_POINTS, points, pushpt.

参照元 Kinect::run().

```
15 {
16     CoordinateTransform ct; //CoordinateTransform クラスの S インスタンスの生成
17
18     int check = 0; //距離データが 0 でないかチェックするフラグ.check == 0 なら距離データが 0 以外を表す.
19
20     if (event == CV_EVENT_LBUTTONDOWN){
21         if (points < MAX_POINTS){
22             pushpt[points] = cv::Point(x, y);
23             check = ct.ct2Dto3D(pushpt[points], points);
24             if (check == 0){
25                 points++;
26             }
27         }
28     }
29 }
```

4.7 PlotFunc.cpp ファイル

PlotFunc クラスの実装ファイル

```
#include "EstimatePosture.h"
```

4.7.1 詳解

PlotFunc クラスの実装ファイル

日付

2014/04/01

著者

H.Shigehara

PlotFunc.cpp に定義があります。

Index