【雙目測距】基於OpenCV的雙目攝像頭測距

一顆小樹x 發表於2020/05/28 23:06:28

4.4k+ 0 0

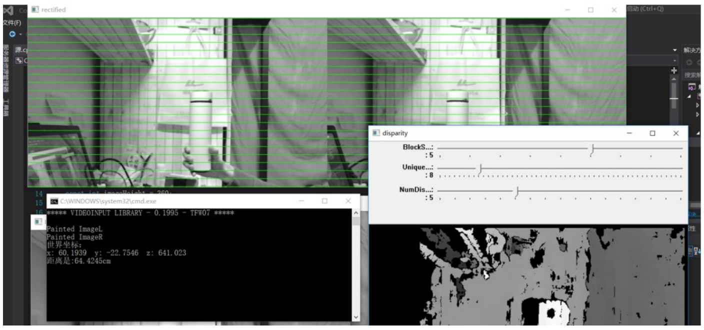
【摘要】 首先進行雙目攝像頭定標，獲取雙目攝像頭內部的參數後，進行測距；本文的雙目視覺測距是基於BM算法。注意：雙目定標的效果會影響測距的精準度，建議大家在做雙目定標時，做好一些（盡量讓誤差小）。

前言

首先進行雙目攝像頭定標，獲取雙目攝像頭內部的參數後，進行測距；本文的雙目視覺測距是基於BM算法。注意：雙目定標的效果會影響測距的精準度，建議大家在做雙目定標時，做好一些（盡量讓誤差小）。

一、雙目測距--輸入圖片

效果1：



```
***** VIDEOINPUT LIBRARY - 0.1995 - TFW07 *****

Painted ImageL
Painted ImageR
世界坐标:
x: 60.1939  y: -22.7546  z: 641.023
距离是:64.4245cm
```

推薦閱讀

3h精通OpenCV（九）-最簡單的人臉檢測

Opencv存圖讀圖

3h精通OpenCV（八）-形狀檢測

3h精通OpenCV（七）-顏色檢測

3h精通OpenCV（六）-圖像堆疊

3h精通OpenCV（五）-透視變換

3h精通OpenCV（四）-繪製形狀與文本

3h精通OpenCV（三）-重調大小與圖像裁剪

3h精通OpenCV（二）-基本功能

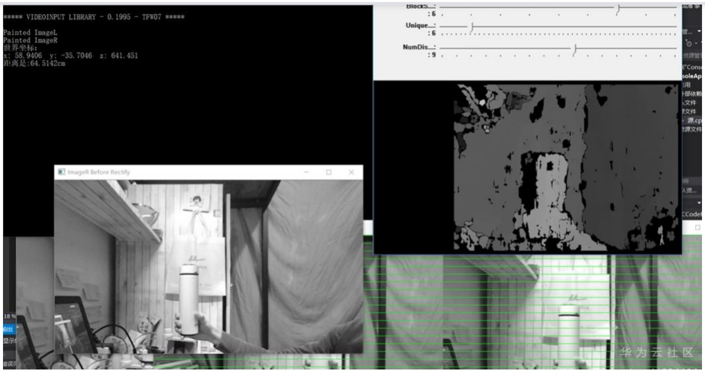
3h精通OpenCV（零）- 環境配置

相關產品

函數工作流

實時流計算服務

NAT網關



```
***** VIDEOINPUT LIBRARY - 0.1995 - TFW07 *****

Painted ImageL
Painted ImageR
世界坐标:
x: 58.9406  y: -35.7046  z: 641.451
距离是:64.5142cm
```

本人通過測試，誤差是1cm.

其中參數：BlockSize、UniquenessRatio、NumDisparities 根據實際情況來調整；

選擇C++運行效率高，BM算法可以自定義修改，比較靈活；嘗試過Python版的BM算法雙目測距，效果沒C++好。

源代碼：

```
/*           双目测距         */

#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace std;
using namespace cv;

const int imageWidth = 640;
const int imageHeight = 360;
Vec3f  point3;
float d;
Size imageSize = Size(imageWidth, image

Mat rgbImageL, grayImageL;
Mat rgbImageR, grayImageR;
Mat rectifyImageL, rectifyImageR;

Rect validROIL;//图像校正之后，会对图像进
Rect validROIR;
```

```
Point origin;          //鼠标按下的起始点
Rect selection;       //定义矩形选框
bool selectObject = false;    //是否选
int blockSize = 0, uniquenessRatio = 0,
Ptr<StereoBM> bm = StereoBM::create(16,

/*事先标定好的左相机的内参矩阵
fx 0 cx
0 fy cy
0  0  1
*/
Mat cameraMatrixL = (Mat_<double>(3, 3)
    0, 421.222568242056, 235.466208987
    0, 0, 1);
//获得的畸变参数
/*418.523322187048  0   0
-1.26842201390676   421.222568242056
344.758267538961    243.318992284899

Mat distCoeffL = (Mat_<double>(5, 1) <<
//[0.006636837611004,0.050240447649195]
/*事先标定好的右相机的内参矩阵
fx 0 cx
0 fy cy
0  0  1
*/
Mat cameraMatrixR = (Mat_<double>(3, 3)
    0, 419.795432389420, 230.6,
    0, 0, 1);

/*
417.417985082506    0   0
0.498638151824367   419.795432389420
309.903372309072    236.256106972796
*/ //2
Mat distCoeffR = (Mat_<double>(5, 1) <<
//[-0.038407383078874,0.23639280030161
Mat T = (Mat_<double>(3, 1) << -1.2101
//[-1.210187345641146e+02,0.5192354268
```

```
    -0.0362888948713456, -0.03864194686

/* 0.999341122700880    0.0006607480314
-0.00206388651740061    0.9992509896516
0.0362361815232777  0.0386913826603732

//Mat T = (Mat_<double>(3, 1) << -48.4,


//Mat rec = (Mat_<double>(3, 1) << -0.6

Mat R;//R 旋转矩阵
      /*****立体匹配*****/
void stereo_match(int, void*)
{
    bm->setBlockSize(2 * blockSize + 5)
    bm->setROI1(validROIL);
    bm->setROI2(validROIR);
    bm->setPreFilterCap(31);
    bm->setMinDisparity(0);  //最小视差
    bm->setNumDisparities(numDisparitie
    bm->setTextureThreshold(10);
    bm->setUniquenessRatio(uniquenessRa
    bm->setSpeckleWindowSize(100);
    bm->setSpeckleRange(32);
    bm->setDisp12MaxDiff(-1);
    Mat disp, disp8;
    bm->compute(rectifyImageL, rectifyI
    disp.convertTo(disp8, CV_8U, 255 /
    reprojectImageTo3D(disp, xyz, Q, tr
    xyz = xyz * 16;
    imshow("disparity", disp8);
}

/*****描述：鼠标操作回调*****/
static void onMouse(int event, int x, i
{
    if (selectObject)
    {
    }

    switch (event)
    {
    case EVENT_LBUTTONDOWN:   //鼠标左按
        origin = Point(x, y);
        selection = Rect(x, y, 0, 0);
        selectObject = true;
        //cout << origin << "in world c
          point3 = xyz.at<Vec3f>(origi
        point3[0];
        //cout << "point3[0]:" << point
        cout << "世界坐标：" << endl;
        cout << "x: " << point3[0] << '
         d = point3[0] * point3[0]+ poi
         d = sqrt(d);   //mm
        // cout << "距离是:" << d << "m

         d = d / 10.0;   //cm
         cout << "距离是:" << d << "cm"

        // d = d/1000.0;   //m
        // cout << "距离是:" << d << "m

        break;
    case EVENT_LBUTTONUP:    //鼠标左按
        selectObject = false;
        if (selection.width > 0 && sele
            break;
    }
}
/*****主函数*****/
int main()
{
    /*
    立体校正
    */
    Rodrigues(rec, R); //Rodrigues变换
    stereoRectify(cameraMatrixL, distC
    /*
    读取图片
    */
    rgbImageL = imread("image_left_1.jp
    cvtColor(rgbImageL, grayImageL, CV_
    rgbImageR = imread("image_right_1.j
    cvtColor(rgbImageR, grayImageR, CV_

    imshow("ImageL Before Rectify", gra
    imshow("ImageR Before Rectify", gra

    /*
    经过remap之后，左右相机的图像已经共面
    */
    remap(grayImageL, rectifyImageL, ma
    remap(grayImageR, rectifyImageR, ma

    /*
    把校正结果显示出来
    */
    Mat rgbRectifyImageL, rgbRectifyIma
    cvtColor(rectifyImageL, rgbRectifyI
    cvtColor(rectifyImageR, rgbRectifyI

    //单独显示
    //rectangle(rgbRectifyImageL, valid
    //rectangle(rgbRectifyImageR, valid
    imshow("ImageL After Rectify", rgbR
    imshow("ImageR After Rectify", rgbR

    //显示在同一张图上
    Mat canvas;
    double sf;
    int w, h;
    sf = 600. / MAX(imageSize.width, im
    w = cvRound(imageSize.width * sf);
    h = cvRound(imageSize.height * sf);
    canvas.create(h, w * 2, CV_8UC3);
```

```
        cvRound(validROIL.width*sf), cv
    //rectangle(canvasPart, vroiL, Scal
    cout << "Painted ImageL" << endl;

    //右图像画到画布上
    canvasPart = canvas(Rect(w, 0, w, h
    resize(rgbRectifyImageR, canvasPart
    Rect vroiR(cvRound(validROIR.x * sf
        cvRound(validROIR.width * sf),
    //rectangle(canvasPart, vroiR, Scal
    cout << "Painted ImageR" << endl;

    //画上对应的线条
    for (int i = 0; i < canvas.rows; i
        line(canvas, Point(0, i), Point
    imshow("rectified", canvas);

    /*
    立体匹配
    */
    namedWindow("disparity", CV_WINDOW_
    // 创建SAD窗口 Trackbar
    createTrackbar("BlockSize:\n", "dis
    // 创建视差唯一性百分比窗口 Trackbar
    createTrackbar("UniquenessRatio:\n"
    // 创建视差窗口 Trackbar
    createTrackbar("NumDisparities:\n",
    //鼠标响应函数setMouseCallback(窗口名
    setMouseCallback("disparity", onMou
    stereo_match(0, 0);

    waitKey(0);
    return 0;
}
```

流程說明：

雙目定標可以參考我這篇博客：https://guo-pu.blog.csdn.net/article/details/86602452

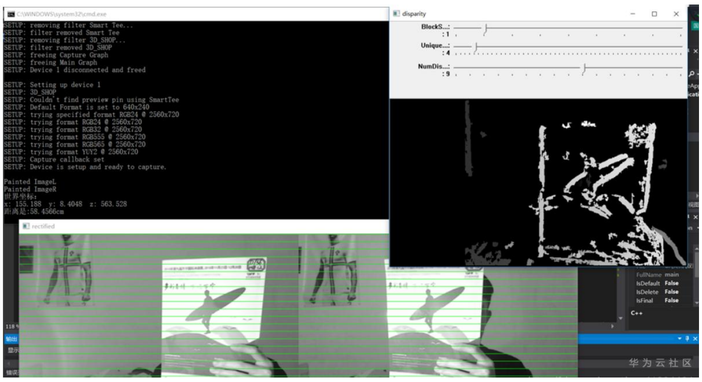
雙目數據轉化可以參考我這篇博客：https://guo-pu.blog.csdn.net/article/details/86710737

詳細講解攝像頭參數：

1）Mat cameraMatrixL 左相機的內參矩陣
2）Mat distCoeffL = (Mat_(5, 1) .... 左相機畸變參數 即 K1，K2，P1，P2，K3。
3） Mat cameraMatrixR 右相機的內參矩陣
4）Mat distCoeffR = (Mat_(5, 1) .... 右相機畸變參數 即 K1，K2，P1，P2，K3。
5) Mat T = (Mat_(3, 1) << -1.210187345641146e+02, 0.519235426836325, -0.425535566316217);// 相機的平移向量
6) Mat rec = (Mat_(3, 3) << 0.999341122700088.................. 相機的旋轉向量
一共6個相機參數，1、2是左相機的參數； 3、4是 右相機的參數； 5、6是相機（相對）整體的參數。

二、實時採集攝像頭數據，進行雙目測距

效果如下圖：



```
SETUP: removing filter Smart Tee...
SETUP: filter removed Smart Tee
SETUP: removing filter 3D_SHOP...
SETUP: filter removed 3D_SHOP
SETUP: freeing Capture Graph
SETUP: freeing Main Graph
SETUP: Device 1 disconnected and freed

SETUP: Setting up device 1
SETUP: 3D_SHOP
SETUP: Couldn't find preview pin using SmartTee
SETUP: Default Format is set to 640x240
SETUP: trying specified format RGB24 @ 2560x720
SETUP: trying format RGB24 @ 2560x720
SETUP: trying format RGB32 @ 2560x720
SETUP: trying format RGB555 @ 2560x720
SETUP: trying format RGB565 @ 2560x720
SETUP: trying format YUY2 @ 2560x720
SETUP: Capture callback set
SETUP: Device is setup and ready to capture.

Painted ImageL
Painted ImageR
世界坐标:
x: 155.188  y: 8.4048  z: 563.528
距离是:58.4566cm
```

源代碼:

```
/*********************************/
```

```
#include <iostream>
#include <math.h>

using namespace std;
using namespace cv;

const int imageWidth = 640;
const int imageHeight = 360;
Vec3f  point3;
float d;
Size imageSize = Size(imageWidth, image

Mat rgbImageL, grayImageL;
Mat rgbImageR, grayImageR;
Mat rectifyImageL, rectifyImageR;

Rect validROIL;//图像校正之后，会对图像进
Rect validROIR;

Mat mapLx, mapLy, mapRx, mapRy;     //
Mat Rl, Rr, Pl, Pr, Q;              //
Mat xyz;              //三维坐标

Point origin;         //鼠标按下的起始点
Rect selection;      //定义矩形选框
bool selectObject = false;    //是否选
int blockSize = 0, uniquenessRatio = 0,
Ptr<StereoBM> bm = StereoBM::create(16,

/*事先标定好的左相机的内参矩阵
fx 0 cx
0 fy cy
0  0  1
*/
Mat cameraMatrixL = (Mat_<double>(3, 3)
    0, 421.222568242056, 235.466208987
    0, 0, 1);
//获得的畸变参数
/*418.523322187048  0   0
```

```
//[0.006636837611004,0.050240447649195]
/*事先标定好的右相机的内参矩阵
fx 0 cx
0 fy cy
0  0  1
*/
Mat cameraMatrixR = (Mat_<double>(3, 3)
    0, 419.795432389420, 230.6,
    0, 0, 1);
/*
417.417985082506    0   0
0.498638151824367   419.795432389420
309.903372309072    236.256106972796
*/ //2
Mat distCoeffR = (Mat_<double>(5, 1) <<
//[-0.038407383078874,0.236392800301615
Mat T = (Mat_<double>(3, 1) << -1.21018
//[-1.210187345641146e+02,0.5192354268

//Mat rec = (Mat_<double>(3, 1) << -0.0
Mat rec = (Mat_<double>(3, 3) << 0.9993
    0.000660748031451783, 0.99925098965
    -0.0362888948713456, -0.03864194686

/* 0.999341122700880    0.0006607480314
-0.00206388651740061    0.9992509896516
0.0362361815232777  0.0386913826603732

//Mat T = (Mat_<double>(3, 1) << -48.4,


//Mat rec = (Mat_<double>(3, 1) << -0.0

Mat R;//R 旋转矩阵
      /*****立体匹配*****/
void stereo_match(int, void*)
{
    bm->setBlockSize(2 * blockSize + 5)
    bm->setROI1(validROIL);
```

```
        bm->setTextureThreshold(10);
        bm->setUniquenessRatio(uniquenessRa
        bm->setSpeckleWindowSize(100);
        bm->setSpeckleRange(32);
        bm->setDisp12MaxDiff(-1);
        Mat disp, disp8;
        bm->compute(rectifyImageL, rectifyI
        disp.convertTo(disp8, CV_8U, 255 /
        reprojectImageTo3D(disp, xyz, Q, tr
        xyz = xyz * 16;
        imshow("disparity", disp8);
    }

    /*****描述：鼠标操作回调*****/
    static void onMouse(int event, int x, i
    {
        if (selectObject)
        {
            selection.x = MIN(x, origin.x);
            selection.y = MIN(y, origin.y);
            selection.width = std::abs(x -
            selection.height = std::abs(y -
        }

        switch (event)
        {
        case EVENT_LBUTTONDOWN:   //鼠标左按
            origin = Point(x, y);
            selection = Rect(x, y, 0, 0);
            selectObject = true;
            //cout << origin << "in world c
              point3 = xyz.at<Vec3f>(origin
            point3[0];
            //cout << "point3[0]:" << point
            cout << "世界坐标：" << endl;
            cout << "x: " << point3[0] << '
             d = point3[0] * point3[0]+ poi
             d = sqrt(d);   //mm
            // cout << "距离是:" << d << "m
            // d = d/1000.0;   //m
            // cout << "距离是:" << d << "m

            break;
        case EVENT_LBUTTONUP:    //鼠标左按
            selectObject = false;
            if (selection.width > 0 && sele
                break;
        }
    }
    /*****主函数*****/
    int main()
    {
        /*
        立体校正
        */
        Rodrigues(rec, R); //Rodrigues变换
        stereoRectify(cameraMatrixL, distCo
            0, imageSize, &validROIL, &vali
        initUndistortRectifyMap(cameraMatri
        initUndistortRectifyMap(cameraMatri

        /*
        打开摄像头
        */
        VideoCapture cap;

            cap.open(1);

            cap.set(CV_CAP_PROP_FRAME_WIDTH
            cap.set(CV_CAP_PROP_FRAME_HEIGH

            if (!cap.isOpened())

            {

                cout << "摄像头打开失败!" <<

                return -1;
            cap >> frame;

            cout << "Painted ImageL" << enc
            cout << "Painted ImageR" << enc

            while (1) {

                double fScale = 0.5;

                Size dsize = Size(frame.col
                Mat imagedst = Mat(dsize, C

                resize(frame, imagedst, dsi
                char image_left[200];
                char image_right[200];
                frame_L = imagedst(Rect(0,
            //  namedWindow("Video_L", 1);
            //  imshow("Video_L", frame_L);

                frame_R = imagedst(Rect(64€
        //      namedWindow("Video_R", 2);
    //          imshow("Video_R", frame_R);
                cap >> frame;
                /*
                读取图片
                */
                //rgbImageL = imread("image
                cvtColor(frame_L, grayImage
                //rgbImageR = imread("image
                cvtColor(frame_R, grayImage

            //  imshow("ImageL Before Recti
            //  imshow("ImageR Before Recti

                /*
                经过remap之后，左右相机的图像
                */
                remap(grayImageL, rectifyIn
                remap(grayImageR, rectifyIn
                Mat rgbRectifyImageL, rgbRe
                cvtColor(rectifyImageL, rgb
                cvtColor(rectifyImageR, rgb

                //单独显示
                //rectangle(rgbRectifyImage
                //rectangle(rgbRectifyImage
            //  imshow("ImageL After Rectif
            //  imshow("ImageR After Rectif

                //显示在同一张图上
                Mat canvas;
                double sf;
                int w, h;
                sf = 600. / MAX(imageSize.w
                w = cvRound(imageSize.width
                h = cvRound(imageSize.heigh
                canvas.create(h, w * 2, CV_


                Mat canvasPart = canvas(Rec
                resize(rgbRectifyImageL, ca
                Rect vroiL(cvRound(validROI
                    cvRound(validROIL.width
                //rectangle(canvasPart, vro
            //  cout << "Painted ImageL" <<

                //右图像画到画布上
                canvasPart = canvas(Rect(w,
                resize(rgbRectifyImageR, ca
                Rect vroiR(cvRound(validROI
                    cvRound(validROIR.width
                //rectangle(canvasPart, vro
            //  cout << "Painted ImageR" <<

                //画上对应的线条
                for (int i = 0; i < canvas.
                    line(canvas, Point(0, i
                imshow("rectified", canvas)
```

```
            namedWindow("disparity", C\
            // 创建SAD窗口 Trackbar
            createTrackbar("BlockSize:\
            // 创建视差唯一性百分比窗口 T
            createTrackbar("UniquenessR
            // 创建视差窗口 Trackbar
            createTrackbar("NumDisparit
            //鼠标响应函数setMouseCallba
            setMouseCallback("disparity
            stereo_match(0, 0);

            waitKey(10);

        } //wheil
    return 0;
}
```

希望對你有幫助。

補充說明：

1.關於如何求出世界坐標?

1）x,y,z 是由

Vec3f point3;

point3 = xyz.at<Vec3f>(origin); 來轉化的。

cout << "x: " << point3[0] << " y: " << point3[1] << " z: " << point3[2] << endl;

2）x,y,z求平方和後開根號，是**兩點的距離公式**，即點（0，0，0）------雙目攝像頭的中心點，和點（x,y,z）進行兩點求距離。

人工智能 OpenCV

點贊 收藏

關注作者

作者其他文章

基於BM算法實現雙目測距（Python版）

MatLab 點雲可視化

GPGGA、GPRMC、GPGSV、GPGSA 數據格式定義

python讀取usb串口數據

Google Colab 開發記錄（5）TensorFlow開發之GPU...

評論（0）

登錄後可評論，請 登錄 或 註冊