Computer vision HW3

There are two major questions for this assignment
- -Stereo system
- -Face-mask recognition

But because of time, the depth calculation is not complete

Stereo system :

Stereo system has two part

1.Calibration + Rectification

use some simple library

```cpp
#include <iostream>
#include <fstream>
#include <opencv2/opencv.hpp>

using namespace std;
using namespace cv;
```

```cpp
Size board_size = Size(9,6);    /* 標定板上每行、列的角點數 */
vector<Point2f> image_points_buf;  /* 快取每幅影象上檢測到的角點 */

/* 提取角點 */
if (0 == findChessboardCorners(imageInput,board_size,image_points_buf))
{
    cout<<"can not find chessboard corners!\n"; //找不到角點
    exit(1);
}
else
{
    Mat view_gray;
    cvtColor(imageInput,view_gray,CV_RGB2GRAY);
    /* 亞畫素精確化 */
    find4QuadCornerSubpix(view_gray,image_points_buf,Size(20, 20)); //對粗提取的角點進行精確化
    //cornerSubPix(view_gray,image_points_buf,Size(5,5),Size(-1,-1),TermCriteria(CV_TERMCRIT_EPS CV_TERMCRIT_ITER,30,0.1));
    /* 在影象上顯示角點位置 */
    drawChessboardCorners(view_gray,board_size,image_points_buf,false); //用於在圖片中標記角點
    drawChessboardCorners(imageInput,board_size,image_points_buf,true); //用於在圖片中標記角點
    imshow("Camera Calibration_gray", view_gray);//顯示圖片
    imshow("Camera Calibration_color", imageInput);
    waitKey();
```

use opencv lib
“findChessboardCorners” can find the corner of the board
“find4QuadCornerSubpix” further extract sub-pixel information from the preliminary extracted corner information
“drawChessboardCorners” used to draw the corner points that were successfully calibrated
“calibrateCamera” to calculate camera internal and external parameter coefficients
“projectPoints” to reproject the three-dimensional points in space
“initUndistortRectifyMap “ used to calculate distortion mapping
“remap” apply the obtained antipodes to the image
“undistort” achieve corrective effect, is same about “initUndistortRectifyMap “ + “remap”

```
Size image_size;  /* 影象的尺寸 */
vector<vector<Point2f>> image_points_seq; /* 儲存檢測到的所有角點 */
vector<vector<Point3f>> object_points; /* 儲存標定板上角點的三維座標 */
/*內外引數*/
Mat cameraMatrix=Mat(3,3,CV_32FC1,Scalar::all(0)); /* 攝像機內引數矩陣 */
Mat distCoeffs=Mat(1,5,CV_32FC1,Scalar::all(0)); /* 攝像機的5個畸變係數：k1,k2,p1,p2,k3 */
vector<Mat> tvecsMat;  /* 每幅影象的旋轉向量 */
vector<Mat> rvecsMat; /* 每幅影象的平移向量 */

image_points_seq.push_back(image_points_buf);  //儲存亞畫素角點
image_size.width = imageInput.cols;//讀入第一張圖片時獲取影象寬高資訊
image_size.height = imageInput.rows;

/* 初始化標定板上角點的三維座標 */
Size square_size = Size(10,10);  /* 實際測量得到的標定板上每個棋盤格的大小 */
vector<Point3f> tempPointSet;
for (int i=0; i<board_size.height; i++)
    {
        for (int j=0; j<board_size.width; j++)
            {
                Point3f realPoint;
                /* 假設標定板放在世界座標系中z=0的平面上 */
                realPoint.x = i*square_size.width;
                realPoint.y = j*square_size.height;
                realPoint.z = 0;
                tempPointSet.push_back(realPoint);
            }
    }
    object_points.push_back(tempPointSet);
```

```
/* 開始標定 */
calibrateCamera(object_points,image_points_seq,image_size,cameraMatrix,distCoeffs,rvecsMat,tvecsMat,0);

//對標定結果進行評價
//vector<Point3f> tempPointSet = object_points[0];
vector<Point2f> image_points2; /* 儲存重新計算得到的投影點 */

projectPoints(tempPointSet,rvecsMat[0],tvecsMat[0],cameraMatrix,distCoeffs,image_points2);
```

2. Disparity Map

```
Mat mapx = Mat(image_size,CV_32FC1);
Mat mapy = Mat(image_size,CV_32FC1);
Mat R = Mat::eye(3,3,CV_32F);

initUndistortRectifyMap(cameraMatrix,distCoeffs,R,cameraMatrix,image_size,CV_32FC1,mapx,mapy);

remap(imageInput,imageInput,mapx, mapy, INTER_LINEAR);
//undistort(imageInput,imageInput,cameraMatrix,distCoeffs);
```

```
cv::Ptr<cv::StereoBM> bm = cv::StereoBM::create(0, 21);

bm->setPreFilterType(CV_STEREO_BM_XSOBEL);  //CV_STEREO_BM_NORMALIZED_RESPONSE或者CV_STEREO_BM_XSOBEL
bm->setPreFilterSize(9);
bm->setPreFilterCap(31);
bm->setBlockSize(15);
bm->setMinDisparity(0);
bm->setNumDisparities(64);
bm->setTextureThreshold(10);
bm->setUniquenessRatio(5);
bm->setSpeckleWindowSize(100);
bm->setSpeckleRange(32);
bm->setROI1(leftROI);
bm->setROI2(rightROI);

bm->compute(left, right, disp);
normalize(disp, disp, 0, 255, CV_MINMAX, CV_8U);
```

Stereo reconstruction is to reconstruct 3D points from a calibrated stereo camera pair. It is a basic problem of computer vison. However, for omnidirectional camera, it is not very popular because of the large distortion make it a little difficult. Conventional methods rectify images to perspective ones and do stereo reconstruction in perspective images. However, the last section shows that recifying to perspective images lose too much field of view, which waste the advantage of omnidirectional camera, i.e. large field of view.

The first step of stereo reconstruction is stereo rectification so that epipolar lines are horizontal lines. Here, we use longitude-latitude rectification to preserve all filed of view, or perspective rectification which is available but is not recommended. The second step is stereo matching to get a disparity map. At last, 3D points can be generated from disparity map.

The API of stereo reconstruction for omnidirectional camera is omnidir::stereoReconstruct. Here we use an example to show how it works.

First, calibrate a stereo pair of cameras as described above and get parameters like K1, D1, xi1, K2, D2, xi2, rvec, tvec. Then read two images from the first and second camera respectively, for instance, image1 and image2, which are shown below.

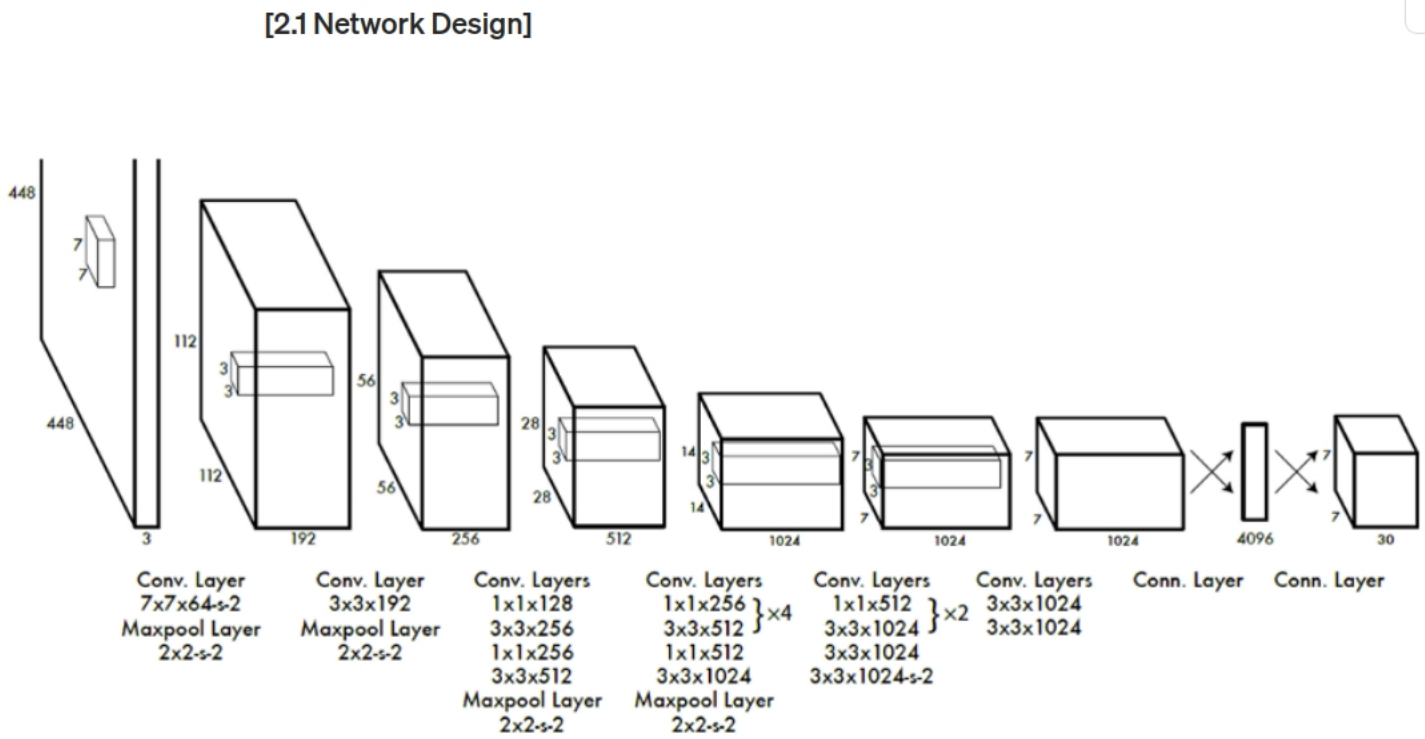
It can be observed that they are well aligned. The variable disMap is the disparity map computed by cv::StereoSGBM from imageRec1 and imageRec2. The disparity map of the above two images is:

Face-mask recognition :

I use yolov3 to do data train and image detect
about yolo

YOLO — You Only Look Once

yolo use CNN Architecture to get the characteristics of the target



“darknet” library is main part in yolo about object dectect

there three file to input

yolov3-tiny.cfg: many parameter by train data
obj.names: object dectect names
yolov3-tiny.weight: modle for object detection

```cpp
#include <cstdio>
#include <iostream>
#include <fstream>
#include <time.h>

#include <opencv2/opencv.hpp>

#define OPENCV
#include <yolo_v2_class.hpp>

using namespace std;
using namespace cv;
```

```cpp
string yoloCfgFile = "/home/james/Desktop/Computer_vision/HW3/image_files/mask_dataset/setting/yolov3-tiny.cfg";
string yoloWeightFile = "/home/james/Desktop/Computer_vision/HW3/image_files/mask_dataset/model/yolov3-tiny_5300.weights";
string yoloNameFile = "/home/james/Desktop/Computer_vision/HW3/image_files/mask_dataset/setting/obj.names";
```

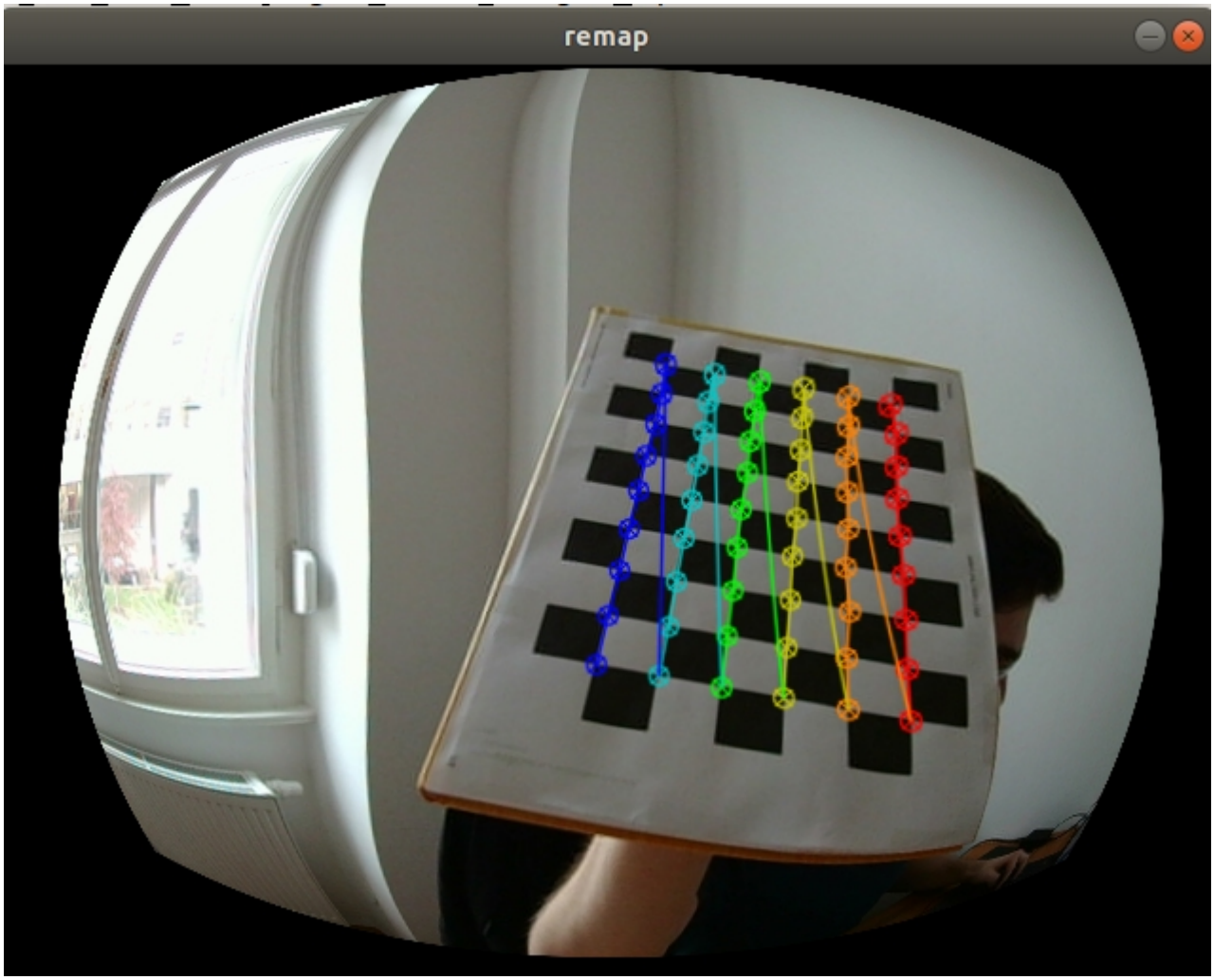
```
//Predict an image
vector<bbox_t> predict = detector.detect(image);

for (size_t i = 0; i < predict.size(); i++) {
    cout << boundingBoxInfo(predict[i]) << endl;
    drawBoundingBox(image, predict[i], classNames);
    //cout <<"probability = "<< predict[i].prob*100 <<" %" << endl;
}


end = clock();
double fps = 1 / ((double) (end - start)) * CLOCKS_PER_SEC;
```

result

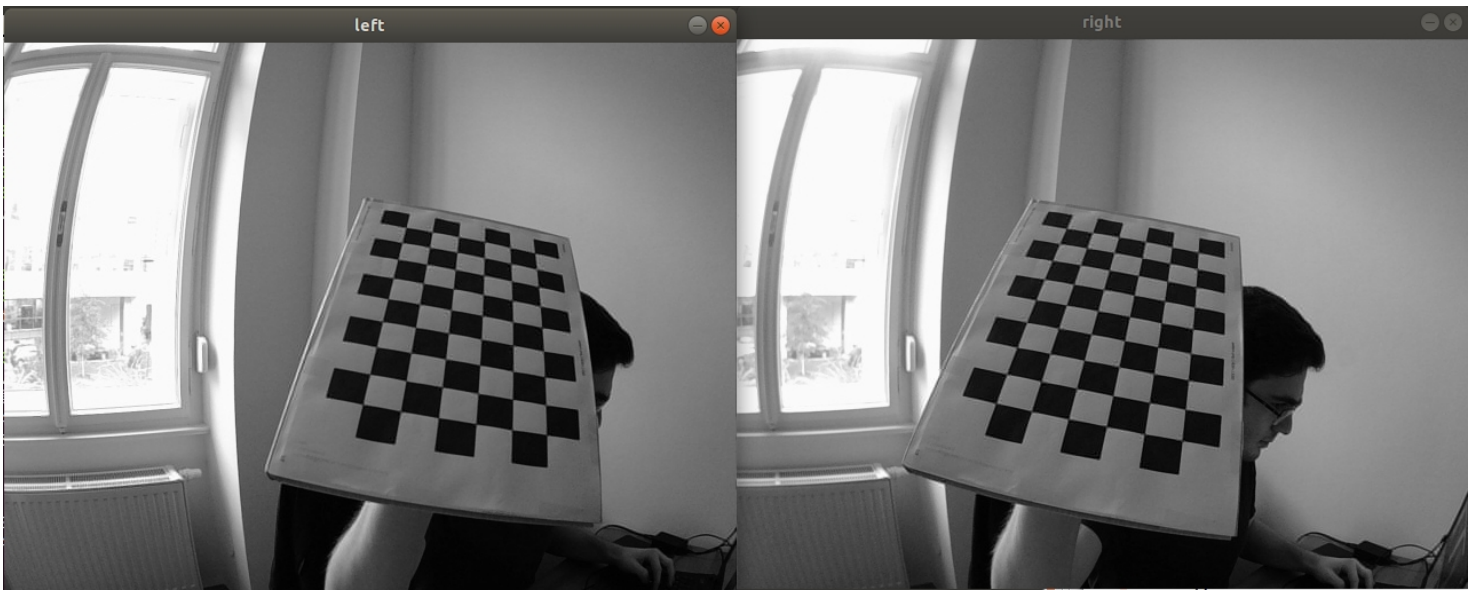
Stereo system :

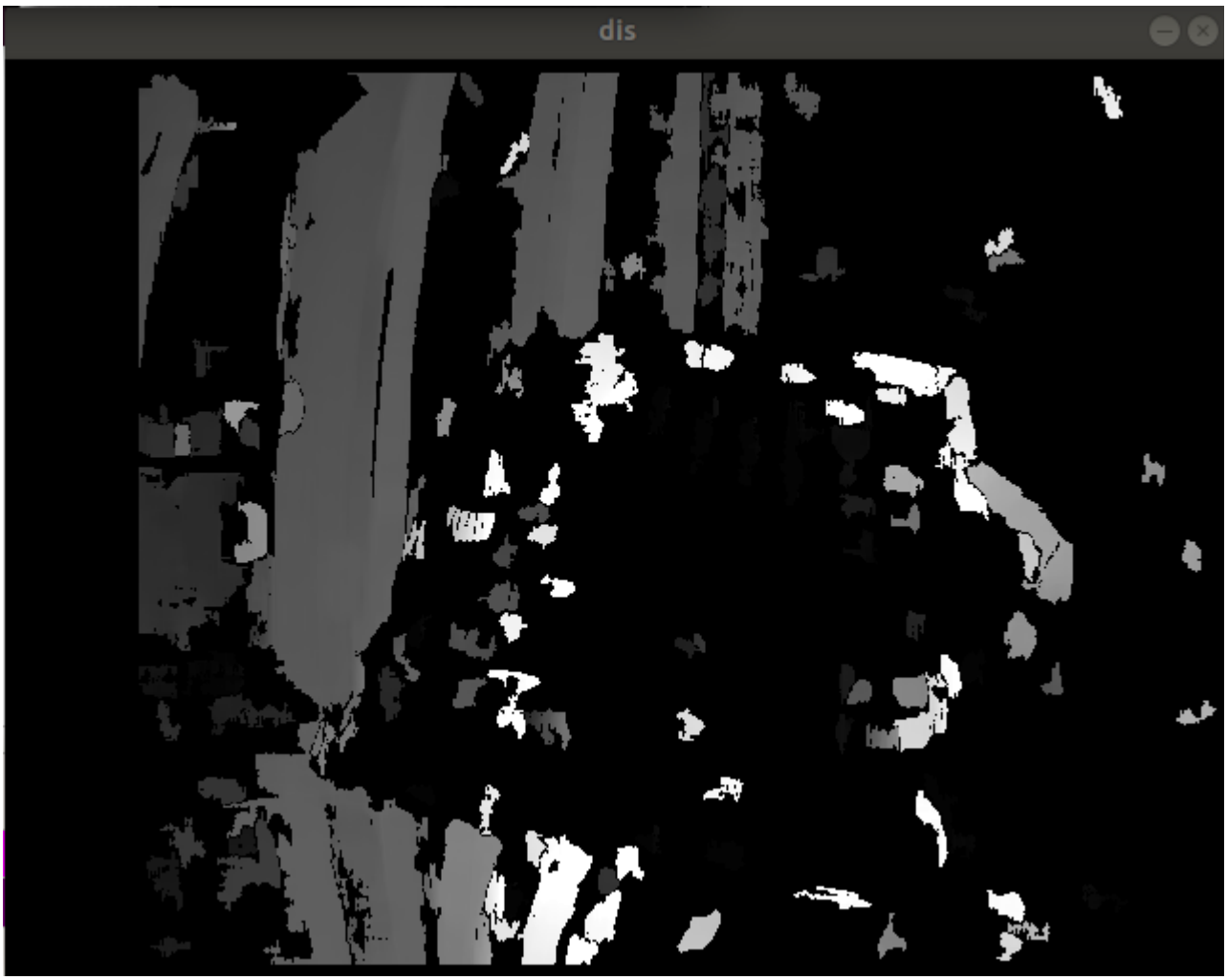
Calibration:

Rectification:

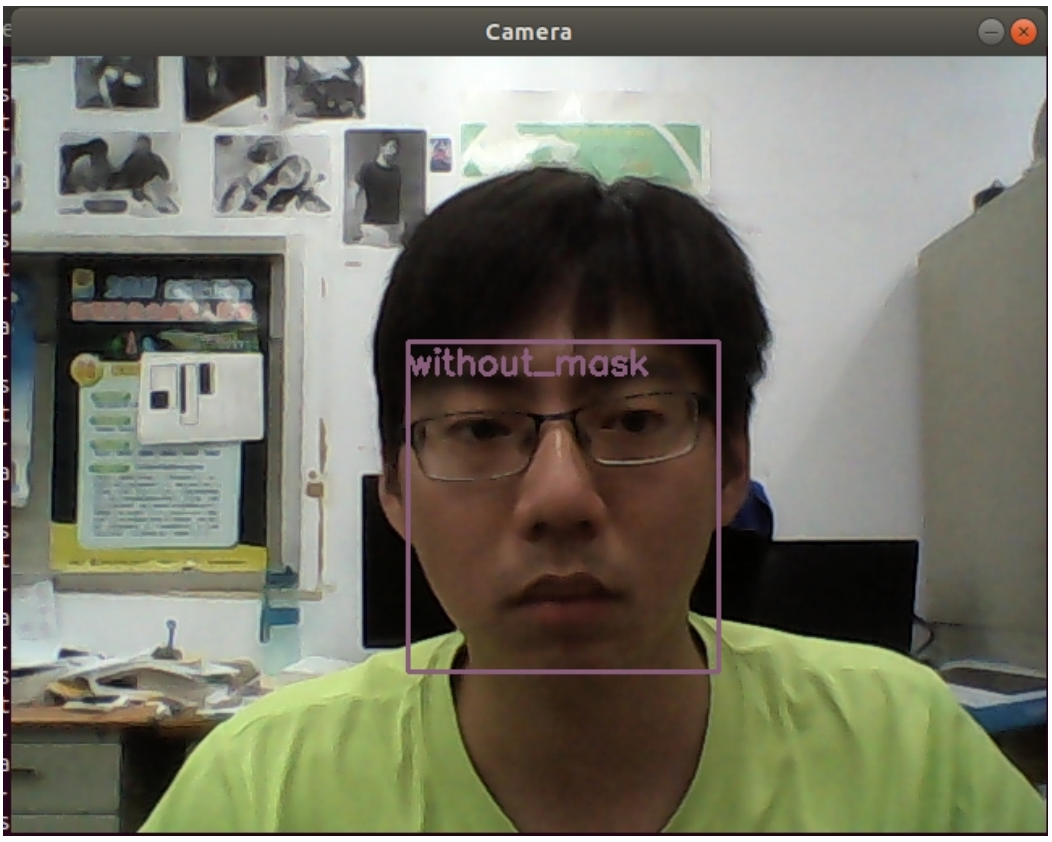


Disparity Map:



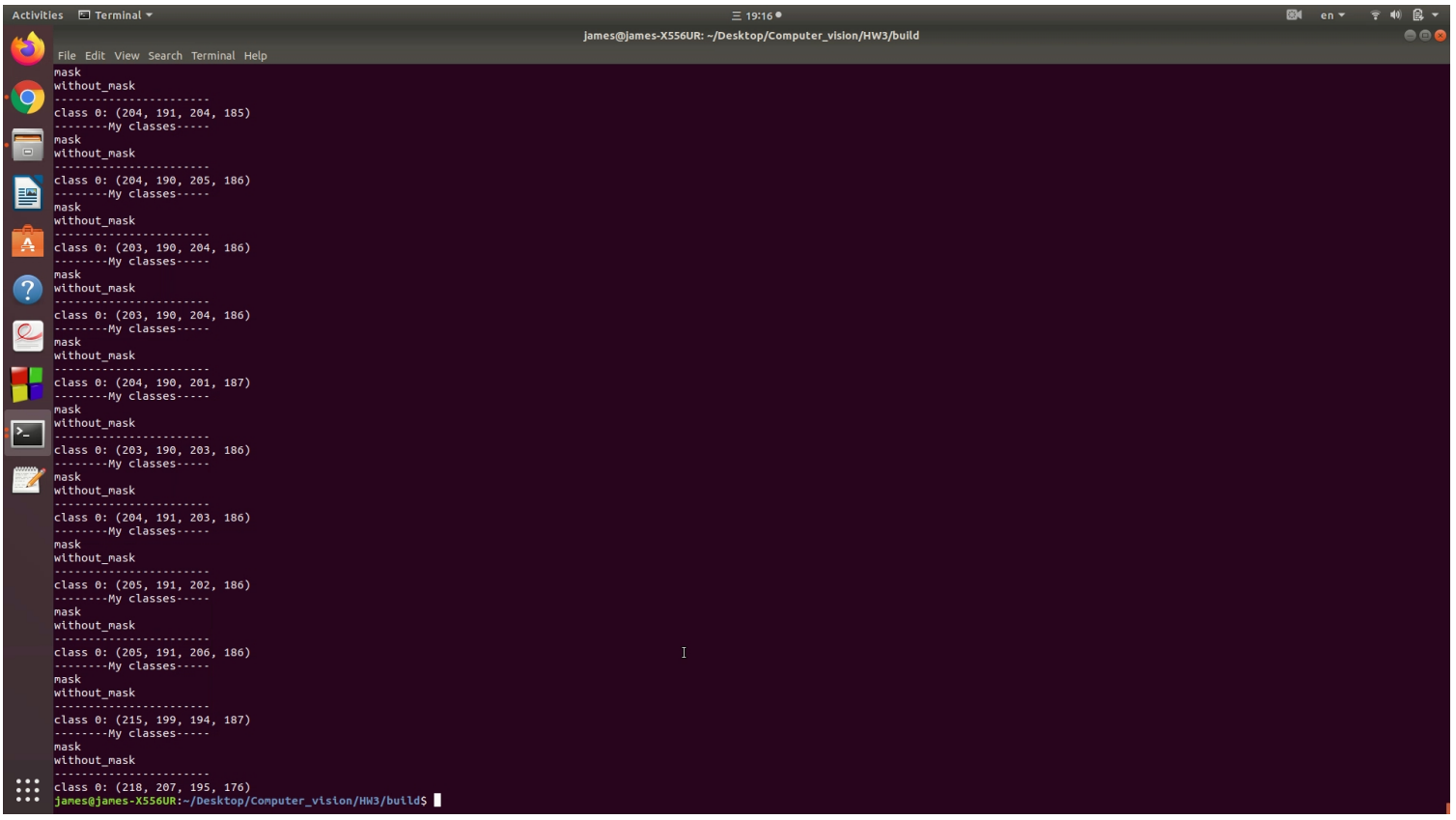


Face-mask recognition :




Camera
mask

video:


Activities
Terminal
19:16
en
james@james-X556UR: ~/Desktop/Computer_vision/HW3/build
File Edit View Search Terminal Help
mask
without_mask
----
class 0: (204, 191, 204, 185)
--------My classes-----
mask
without_mask
----
class 0: (204, 190, 205, 186)
--------My classes-----
mask
without_mask
----
class 0: (203, 190, 204, 186)
--------My classes-----
mask
without_mask
----
class 0: (203, 190, 204, 186)
--------My classes-----
mask
without_mask
----
class 0: (204, 190, 201, 187)
--------My classes-----
mask
without_mask
----
class 0: (203, 190, 203, 186)
--------My classes-----
mask
without_mask
----
class 0: (204, 191, 203, 186)
--------My classes-----
mask
without_mask
----
class 0: (205, 191, 202, 186)
--------My classes-----
mask
without_mask
----
class 0: (205, 191, 206, 186)
--------My classes-----
mask
without_mask
----
class 0: (215, 199, 194, 187)
--------My classes-----
mask
without_mask
----
class 0: (218, 207, 195, 176)
james@james-X556UR:~/Desktop/Computer_vision/HW3/build$