MIN-VIS-2016

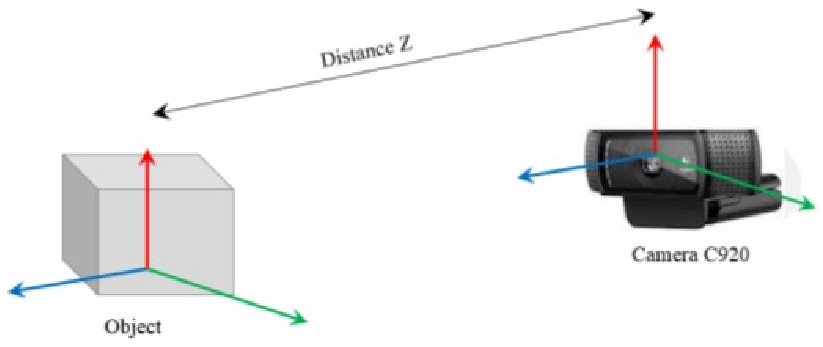
ASSIGNMENT 11 – GROUP4

Group Members:

- Sandio Fongang R
- Armaan Rustami

For exercise (3) You must present your calibration results. To validate your results, use the obtained intrinsic parameters to estimate the distance of an object as shown in the following figure.

Give a demo to proof your results!



This assignment, the first part, must be handed in before November the 25th 2016.

2. Solution

a. Estimate distance of an object from camera:

In order to estimate distance of an object from camera, here are some Steps we took:.

- Calibrate camera and save camera parameter , for calibration we used the chessboard which its width is 14 and height is 9 . *Figure1*

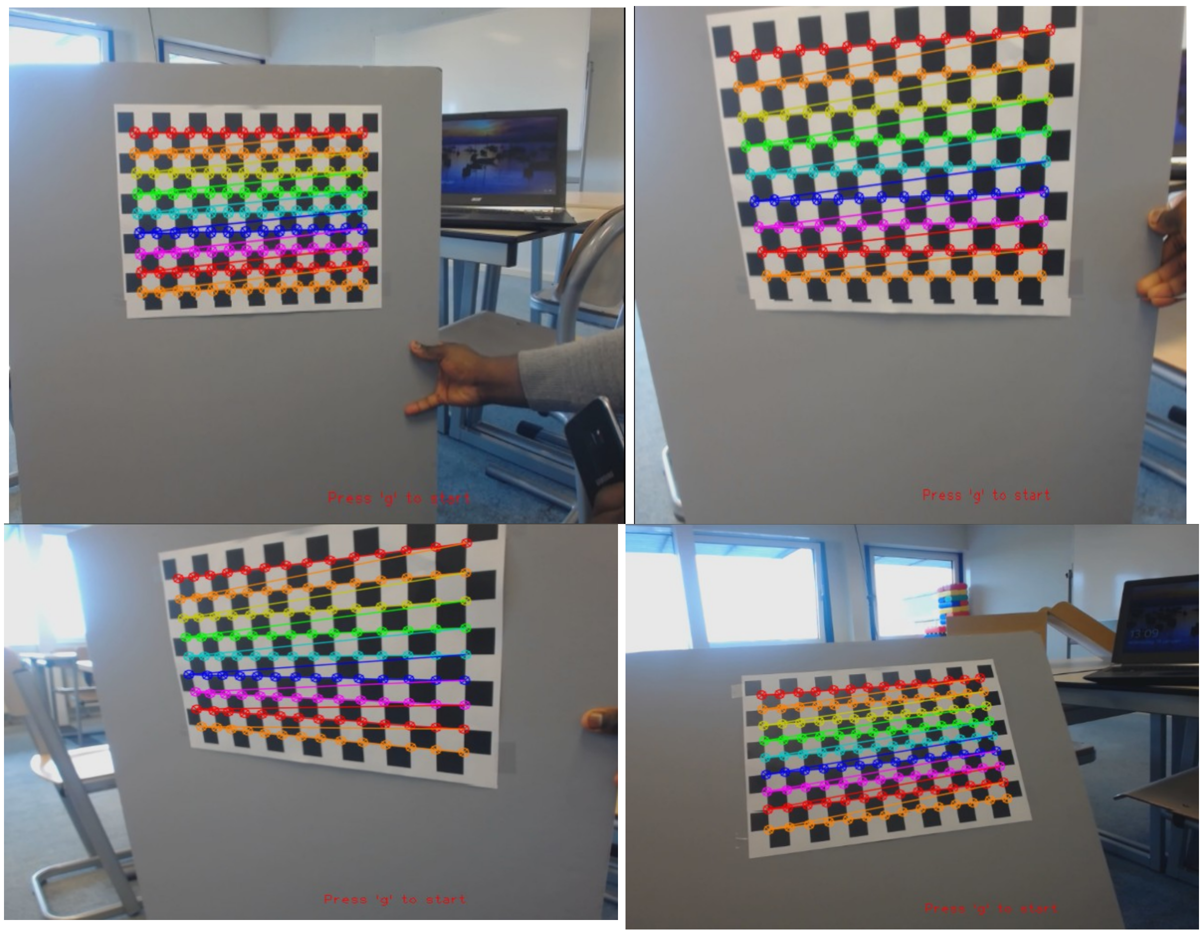
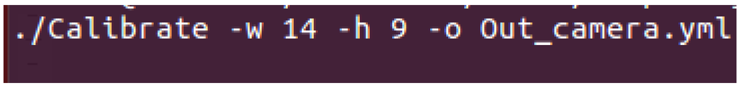


Figure 1: "Calibrated Pictures"

- we used the following command for calibration:



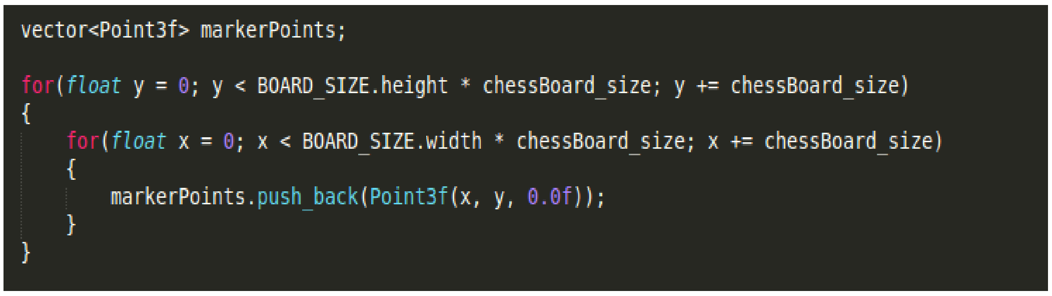
```
./Calibrate -w 14 -h 9 -o Out_camera.yml
```

 -w <board_width> :he number of inner corners per one of board dimension\n"

 -h <board_height> :the number of inner corners per another board dimension\n"

-o <out_camera>]: the output file name for intrinsic [and extrinsic] parameters

- after getting camera parameters, the next step is to use these parameters in find the distance of object from camera.
- Initially create marker points using board height and width size (14,9) with the chessboard square size , in our case its 2.6 cm. *figure2*

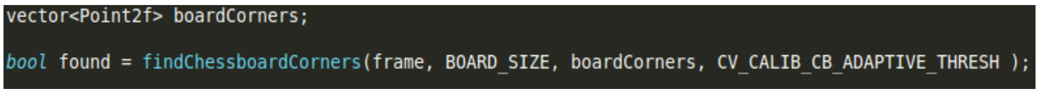


```
vector<Point3f> markerPoints;

for(float y = 0; y < BOARD_SIZE.height * chessBoard_size; y += chessBoard_size)
{
    for(float x = 0; x < BOARD_SIZE.width * chessBoard_size; x += chessBoard_size)
    {
        markerPoints.push_back(Point3f(x, y, 0.0f));
    }
}
```

Figure 2:   Create marker points

- Find the corners of chessboard using built-in open-CV findChessboardCorners function.*figure3*



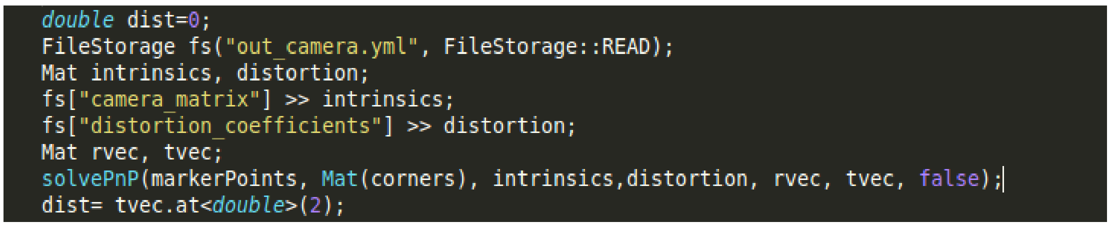
```
vector<Point2f> boardCorners;

bool found = findChessboardCorners(frame, BOARD_SIZE, boardCorners, CV_CALIB_CB_ADAPTIVE_THRESH );
```

Figure 3:   Find chessboard Corners

- After all , we use open-CV built-in solvepnp function which is mostly used for pose estimation, to use solvepnp we need camera_matrix and distortion_coefficients which can find it from camera calibration output file (out_camera.yml),so we read from file and assign them to local variables.

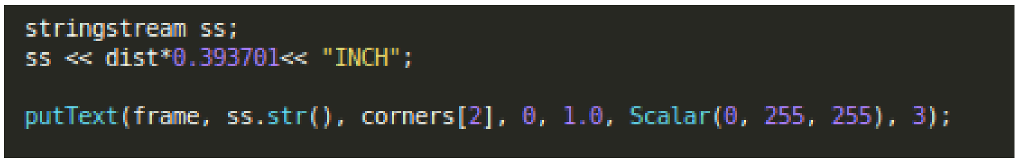
Moreover, we need marker points and corners then we pass them to solvepnp parameter and in translation vector (tvec) which we receive in 2 index we get the distance in cm *figure4*



```
double dist=0;
FileStorage fs("out_camera.yml", FileStorage::READ);
Mat intrinsics, distortion;
fs["camera_matrix"] >> intrinsics;
fs["distortion_coefficients"] >> distortion;
Mat rvec, tvec;
solvePnP(markerPoints, Mat(corners), intrinsics,distortion, rvec, tvec, false);
dist= tvec.at<double>(2);
```

Figure 4: read camera parameters from file and get distance in cm

- finally we use the result and multiply with 0.393701 to show our result in inch and draw it in screen.*figure5*

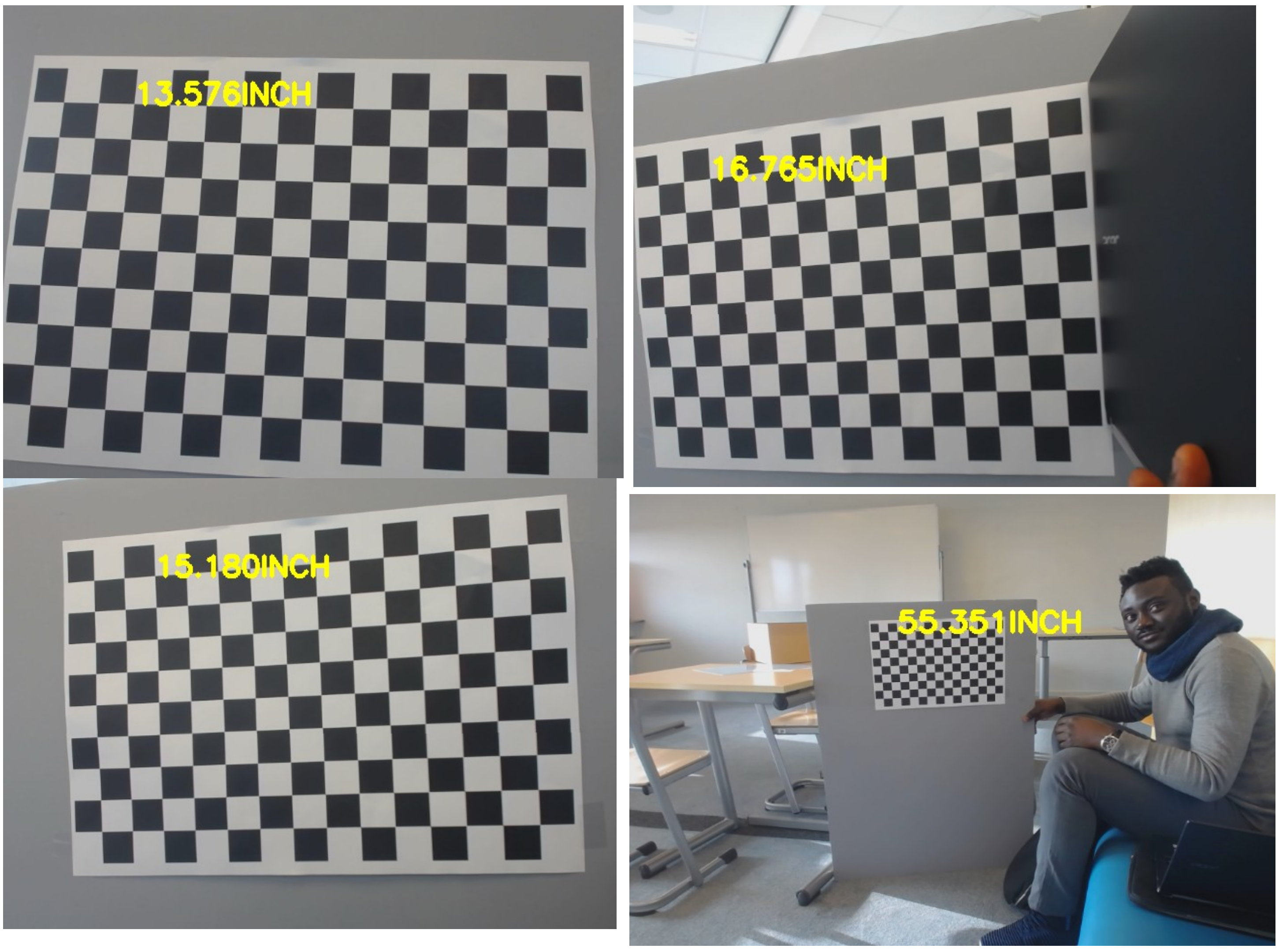


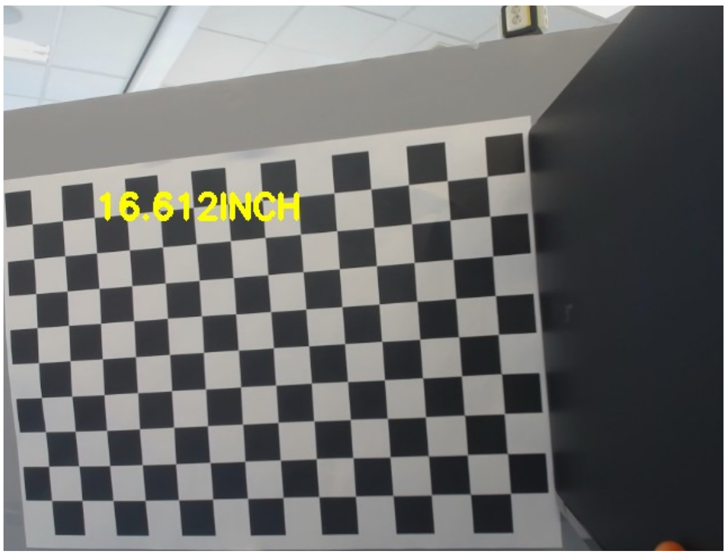
```
stringstream ss;
ss << dist*0.393701<< "INCH";

putText(frame, ss.str(), corners[2], 0, 1.0, Scalar(0, 255, 255), 3);
```

Figure 5: convert cm to inch and print on screen

Output:




16.612INCH