```cpp
//James Rogers Nov 2020 (c) Plymouth University
#include <iostream>
#include <fstream>
#include <opencv2/opencv.hpp>
#include <stdio.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    //Calibration file paths (you need to make these)
    string intrinsic_filename = "../intrinsics.xml";
    string extrinsic_filename = "../extrinsics.xml";

    //================================================Load Calibration
Files================================================
    //This code loads in the intrinsics.xml and extrinsics.xml calibration files, and
creates: map11, map12, map21, map22.
    //These four matrices are used to distort the camera images to apply the lense
correction.
    Rect roi1, roi2;
    Mat Q;
    Size img_size = {640,480};

    FileStorage fs(intrinsic_filename, FileStorage::READ);
    if(!fs.isOpened()){
        printf("Failed to open file %s\n", intrinsic_filename.c_str());
        return -1;
    }

    Mat M1, D1, M2, D2;
    fs["M1"] >> M1;
    fs["D1"] >> D1;
    fs["M2"] >> M2;
    fs["D2"] >> D2;

    fs.open(extrinsic_filename, FileStorage::READ);
    if(!fs.isOpened())
    {
        printf("Failed to open file %s\n", extrinsic_filename.c_str());
        return -1;
    }
    Mat R, T, R1, P1, R2, P2;
    fs["R"] >> R;
    fs["T"] >> T;

    stereoRectify( M1, D1, M2, D2, img_size, R, T, R1, R2, P1, P2, Q,
CALIB_ZERO_DISPARITY, -1, img_size, &roi1, &roi2 );

    Mat map11, map12, map21, map22;
    initUndistortRectifyMap(M1, D1, R1, P1, img_size, CV_16SC2, map11, map12);
    initUndistortRectifyMap(M2, D2, R2, P2, img_size, CV_16SC2, map21, map22);

    //================================================Stereo SGBM
Settings================================================
    //This sets up the block matcher, which is used to create the disparity map. The
various settings can be changed to
    //obtain different results. Note that some settings will crash the program.

    int SADWindowSize=5;            //must be an odd number >=3
    int numberOfDisparities=256;    //must be divisable by 16
```

```
    Ptr<StereoSGBM> sgbm = StereoSGBM::create(0,16,3);
    sgbm->setBlockSize(SADWindowSize);
    sgbm->setPreFilterCap(63);
    sgbm->setP1(8*3*SADWindowSize*SADWindowSize);
    sgbm->setP2(32*3*SADWindowSize*SADWindowSize);
    sgbm->setMinDisparity(0);
    sgbm->setNumDisparities(numberOfDisparities);
    sgbm->setUniquenessRatio(10);
    sgbm->setSpeckleWindowSize(100);
    sgbm->setSpeckleRange(32);
    sgbm->setDisp12MaxDiff(1);
    sgbm->setMode(StereoSGBM::MODE_SGBM);

    //================================================Main Program
Loop==============================================
    enum MODE {CALIBRATION, EXECUTION};
    MODE current_mode = EXECUTION;
    int ImageNum=0;
    if (current_mode == CALIBRATION){
        ImageNum=30; //current image index
    } else {
        ImageNum = 0;
    }

    while (1){
        //Load images from file (needs changing for known distance targets)
        Mat Left,Right;
        if (current_mode == CALIBRATION){
            Left =imread("../Task4/Distance Targets/left" +to_string(ImageNum)+"cm.jpg");
            Right=imread("../Task4/Distance Targets/right"+to_string(ImageNum)+"cm.jpg");
        } else {
            Left =imread("../Task4/Unknown Targets/left" +to_string(ImageNum)+".jpg");
            Right=imread("../Task4/Unknown Targets/right"+to_string(ImageNum)+".jpg");
        }
        cout<<"Loaded image: "<<ImageNum<<endl;

        //Distort image to correct for lens/positional distortion
        remap(Left, Left, map11, map12, INTER_LINEAR);
        remap(Right, Right, map21, map22, INTER_LINEAR);

        //Match left and right images to create disparity image
        Mat disp16bit, disp8bit;
        sgbm->compute(Left, Right, disp16bit);                                  //compute
16-bit greyscalse image with the stereo block matcher
        disp16bit.convertTo(disp8bit, CV_8U, 255/(numberOfDisparities*16.)); //Convert
disparity map to an 8-bit greyscale image so it can be displayed (Only for imshow, do not
use for disparity calculations)

        //================================Your code goes
here==============================
        int BF = 62426;
        //380,250 <- coords to extract
        //display images untill x is pressed
        double intensity_avg = 0;
        for (int i = 0;i < 3;i++){
            for (int j = 0; j < 3; j++){
                intensity_avg += (int)disp16bit.at<ushort>(249+i,379+j);
            }
        }
        intensity_avg /= 9;
        cout << intensity_avg << endl;
        double distance_prediction = BF/intensity_avg;
        cout << distance_prediction <<"cm" << endl;
```

```
        while(waitKey(10)!='x')
        {
            imshow("left", Left);
            imshow("right", Right);
            imshow("disparity", disp8bit);
        }
        //move to next image
        if (current_mode == CALIBRATION){
            ImageNum+=10;
            if(ImageNum>150)
            {
                ImageNum=30;
            }
        } else {
            ImageNum +=1;
            if (ImageNum>7){
                ImageNum=0;
            }
        }
    }
    return 0;
}
```