# 《计算机视觉 相机标定实验》指导书

课程名称：计算机视觉

实验学时：16 学时

综合性、设计性实验：□是 🗹否

面向专业和班级：软件学院 2019级

学生人数： 108

任务编制人：彭绍武

# 实验目的

本次实验课程的目的是了解相机标定以及增强现实技术（AR）的原理，通过编译、运行相机标定以及深度图像计算，加深对算法的理解。

# 实验环境

根据自己电脑的操作系统（window/Linux/MacOSX）配置支持OpenGL的OpenCV。

# 实验内容

# 相机标定实验介绍

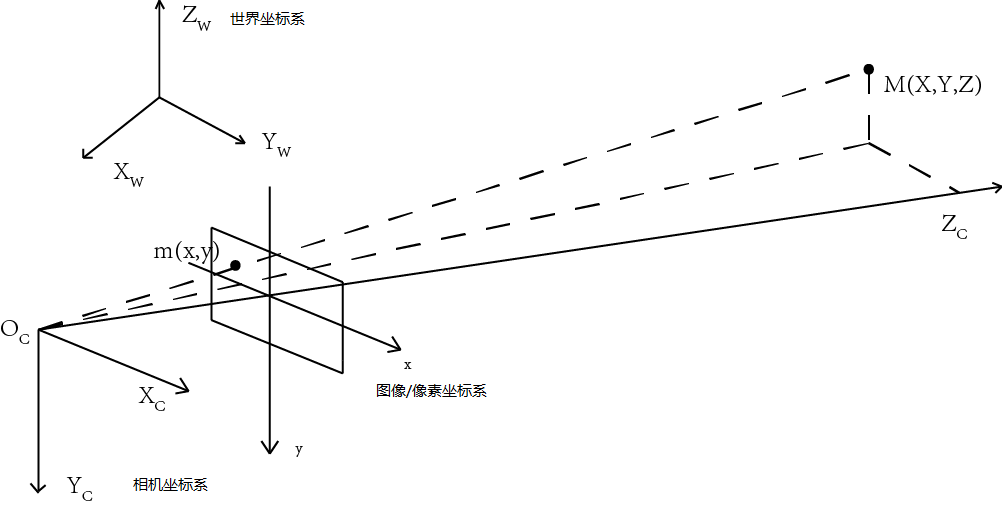
## 相机成像原理

在计算机视觉中，我们使用相机模型将三维空间点与二维图像点联系起来。相机模型分为很多种，可以按照线性模型和非线性模型进行分类。

1. **线性模型与相机坐标系**

最简单的线性模型也就是针孔模型，理想的针孔模型成像与实际物体是颠倒的，在相机模型中，将成像平面与物体放于同一侧，以消除颠倒，如图1所示。可以得出：在空间中的点M(X,Y,Z)投影到相机平面上的点m(x,y)，这个过程满足一个相似关系，这个相似关系也是后面推导过程的基础。

相机的成像过程涉及到四个坐标系，分别是世界坐标系、相机坐标系、图像坐标系以及像素坐标系，M点通过这四个坐标系转换到m。



**图1**

**世界坐标系：**是客观三维世界的绝对坐标系，也称客观坐标系。因为数码相机安放在三维空间中，我们需要世界坐标系这个基准坐标系来描述数码相机的位置，并且用它来描述安放在此三维环境中的其它任何物体的位置，用（XW, YW, ZW）表示其坐标值。

**相机坐标系：**也称光心坐标系,以相机的光心OC为坐标原点，XC 轴和YC 轴分别平行于图像坐标系的 XC 轴和YC 轴，相机的光轴为ZC 轴，用（XC, YC, ZC）表示其坐标值。

从世界坐标系到相机坐标系的转换用齐次坐标可以表示为：

 （1）

其中，R 为3 × 3正交旋转矩阵，t 为三维平移向量。

**图像坐标系：**以图像平面的中心为原点，X轴和Y 轴分别平行于图像平面的两条垂直边，用( x , y )表示其坐标值。图像坐标系是用物理单位（例如毫米）表示像素在图像中的位置。

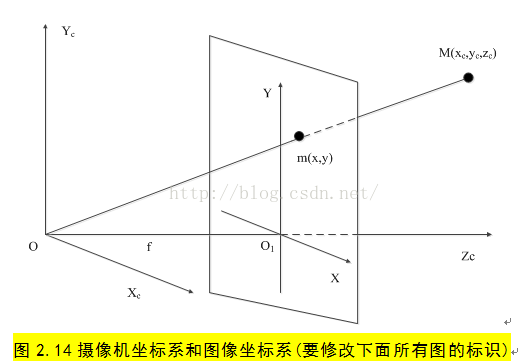


图2

如图2，相机的焦距为f,根据相似关系可得：

 （2）

则从相机坐标系到图像坐标系的齐次坐标转换可以表示为：

 （3）

**像素坐标系：**以图像的左上角顶点为原点，U轴和V轴分别于图像坐标系的X轴和Y轴平行，使用(u,v)来表示坐标值，单位为像素。

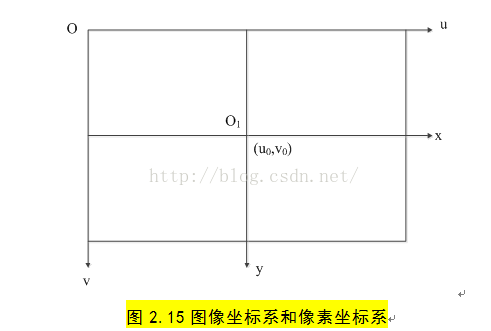


图3

假设每一个像素在U轴和V轴方向上对应的物理尺寸分别为dx和dy，像素坐标与图像坐标的对应关系为：

 （4）

则从图像坐标系到到像素坐标系的齐次坐标转换可以表示为:

 (5)

综上，由世界坐标（三维空间点）到像素坐标（二维平面点）的坐标转换可以表示为：

 （6）

其中矩阵K的参数中只包含焦距、主点坐标等只由相机的内部结构决定，因此称 K 为内部参数矩阵，ax,ay,u0,v0叫做内部参数，M中包含的旋转矩阵和平移向量是由相机坐标系相对于世界坐标系的位置决定的，因此称M为相机的外部参数矩阵，R和t叫做外部参数，M 叫投影矩阵。

1. **非线性模型与镜头畸变**

前面提到的针孔模型是一个较为理想的模型，但是实际上由于镜头加工精度问题，物象并不满足相似关系，相机平面上实际成像与理想成像存在着不同程度的畸变，畸变程度从画面中心到边缘依次递增。镜头畸变分为两类：径向畸变和切向畸变。

**径向畸变：**沿着透镜半径方向分布的畸变，常见的的径向畸变分为枕行畸变和桶形畸变，前者常见于长焦镜头中，后者常见于广角镜头中，如图4。

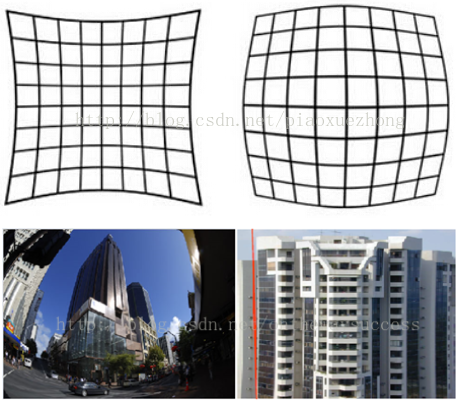


图4

可以使用泰勒级数展开式来作为畸变的数学模型，对畸变成像的中的点(x0,y0)，以及矫正后的点(xcorrect,ycorrect)，有：

 （7）

其中k1，k2，k3表示径向畸变系数，通常使用前两项K1，K2，对于畸变很大的镜头，如鱼眼镜头，可以增加使用k3来进行描述。

**切向畸变：**是由于透镜本身与成像平面不平行而产生的，如图5。

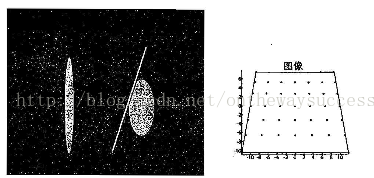


图5

切向畸变可以表示为：

 （8）

其中p1，p2为切向畸变系数。

# 实验详细设计与实现

本节将详细介绍实验的设计与实现，本实验的实验步骤为：

## 配置opencv

备注：熟悉windows开发的同学可以跳过此节。不用VS IDE而是其它开发工具如QT等亦可，仅需注意使用OpenCV即可。

### 下载opencv安装包

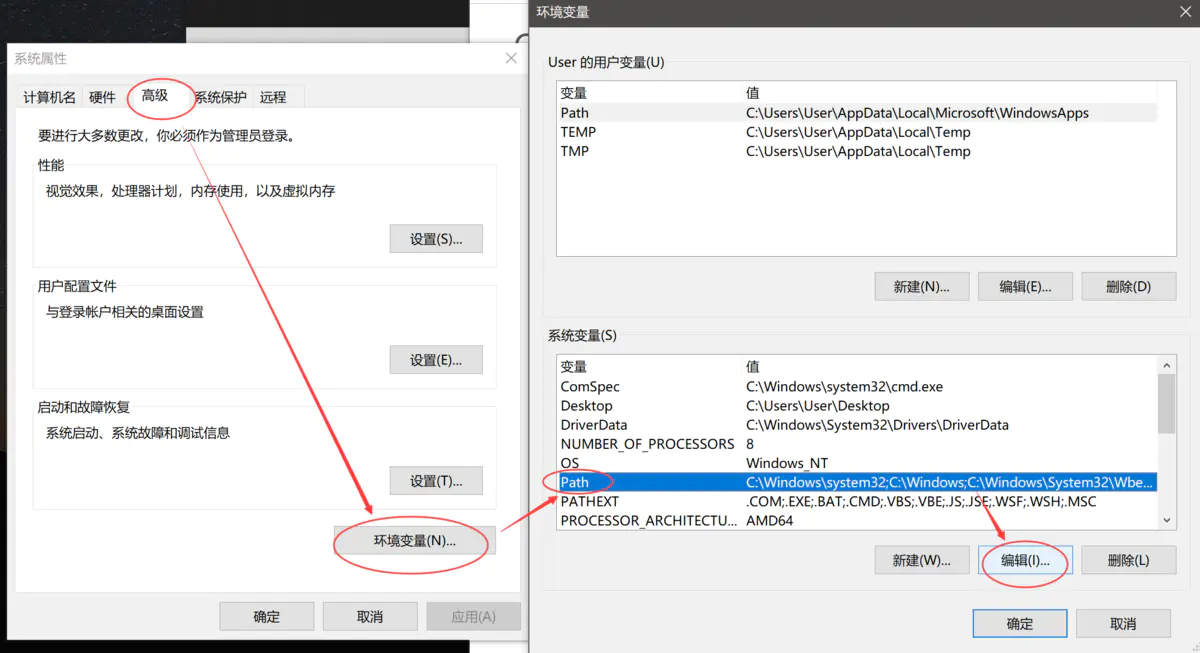
下载链接：[http://opencv.org/releases/](https://links.jianshu.com/go?to=http%3A%2F%2Fopencv.org%2Freleases%2F" \t "_blank)

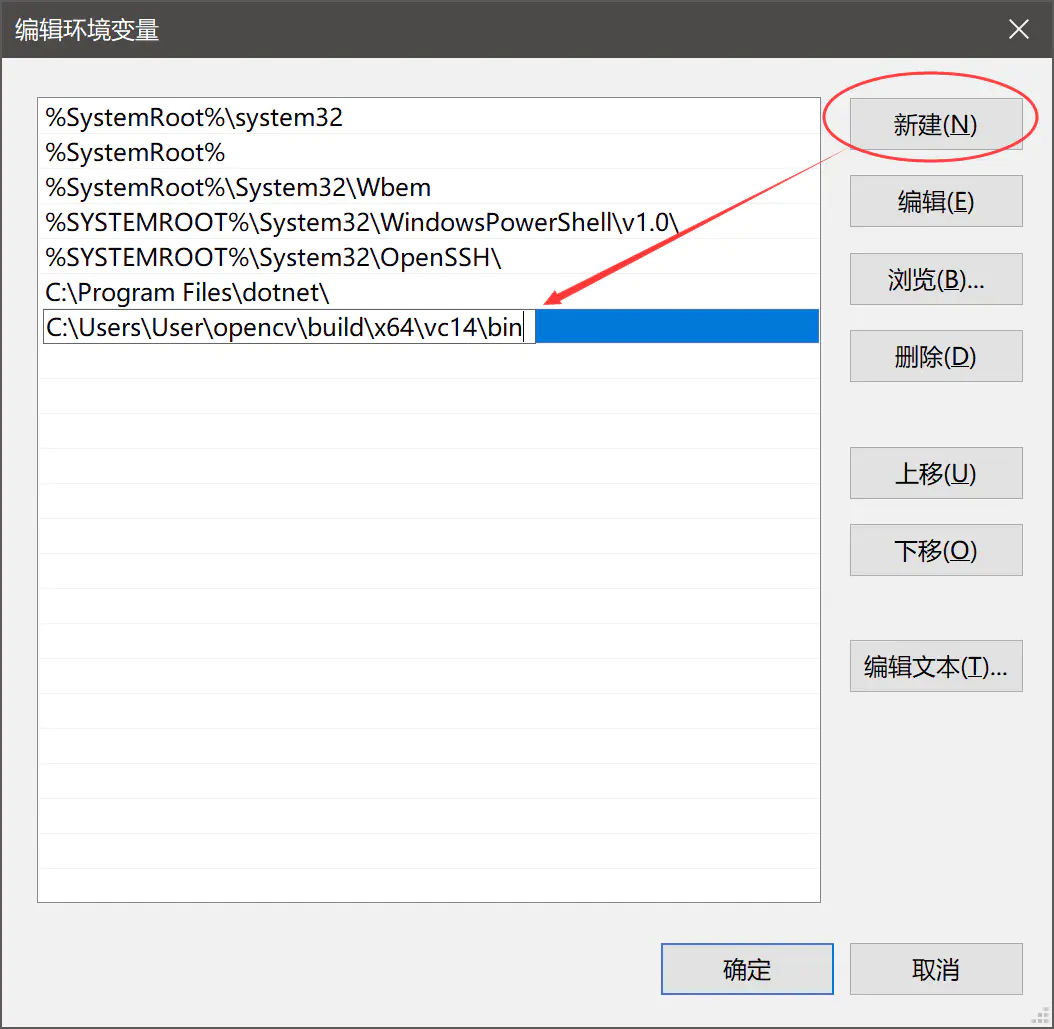
选择3.4.1版本的Win Pack

下载好后，双击运行exe程序解压，选择一个解压后的目录。( 其会自动生成一个opencv目录，不需要单独再创一个opencv目录。

### 配置系统变量（WIN10）

解压完毕后，将其写入系统的环境变量（此电脑->属性->高级系统设置->高级->环境变量）中，向Path中添加一条："解压后opencv所在的文件夹目录\opencv\build\x64\vc14\bin",例如将其解压在C:\Users\User\下，则添加：C:\Users\User\opencv\build\x64\vc14\bin。

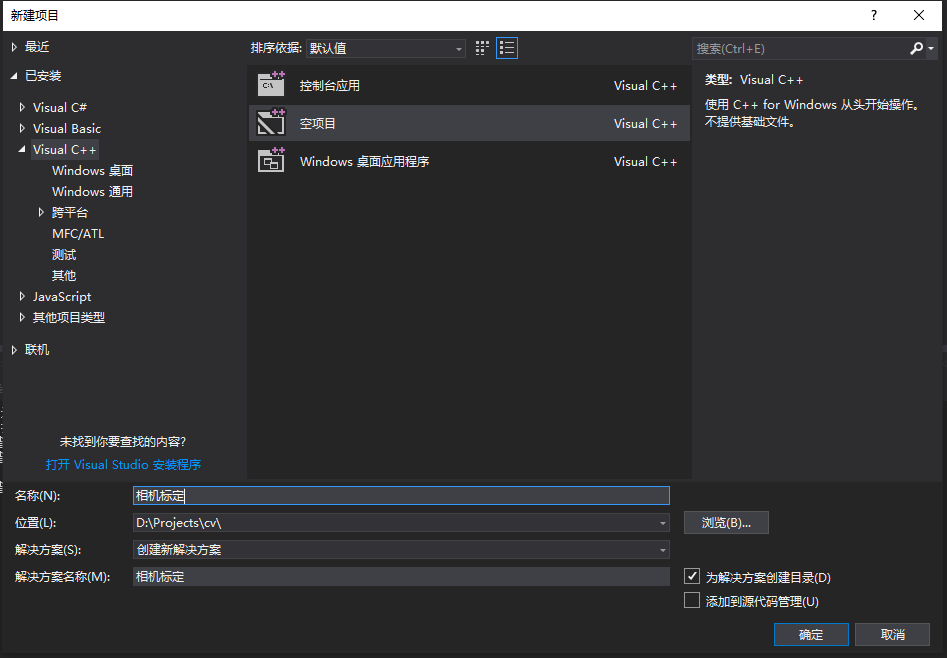




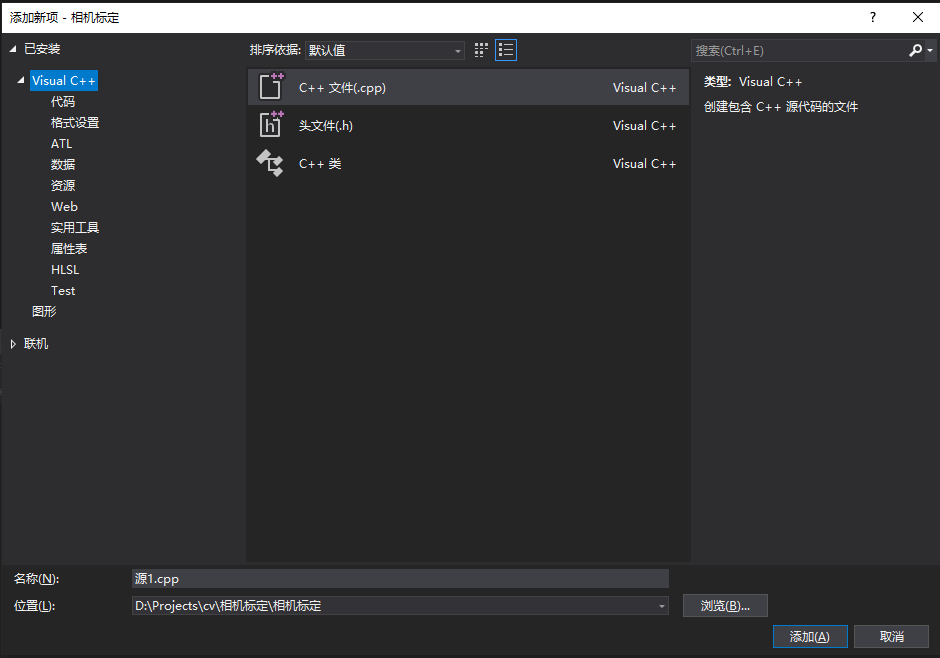
### IDE配置。

例如C++开发环境是Visual Studio 2017，相应的项目创建配置流程如下：

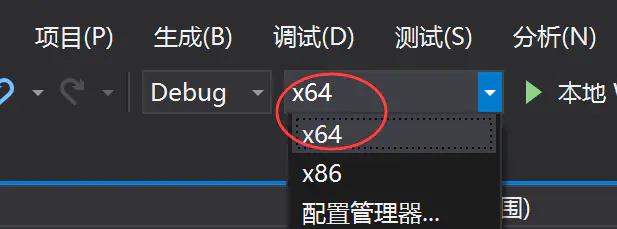
1. 打开Visual Studio2017，新建一个空项目（Ctrl+Shift+N）。



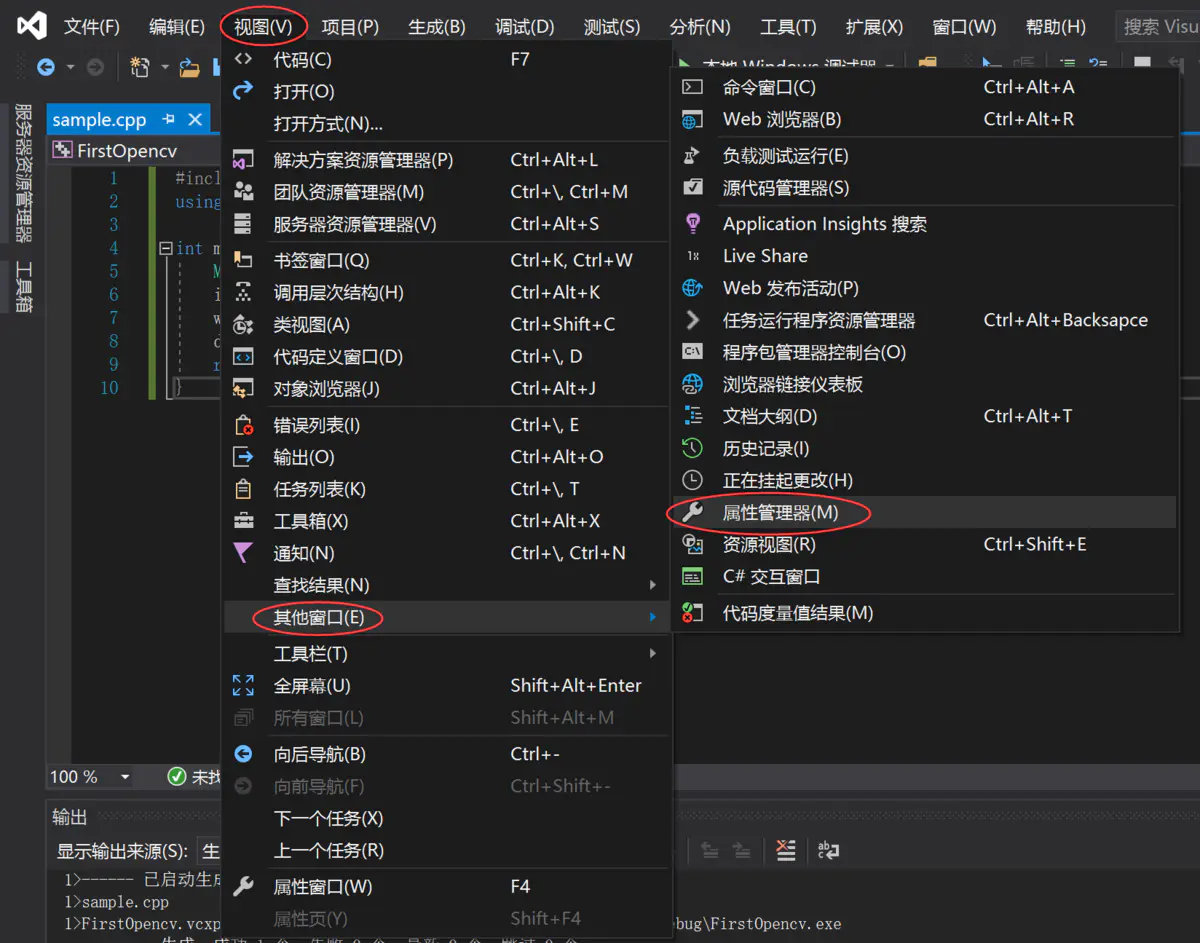
1. 新建一个C++文件（Ctrl+Shift+A）



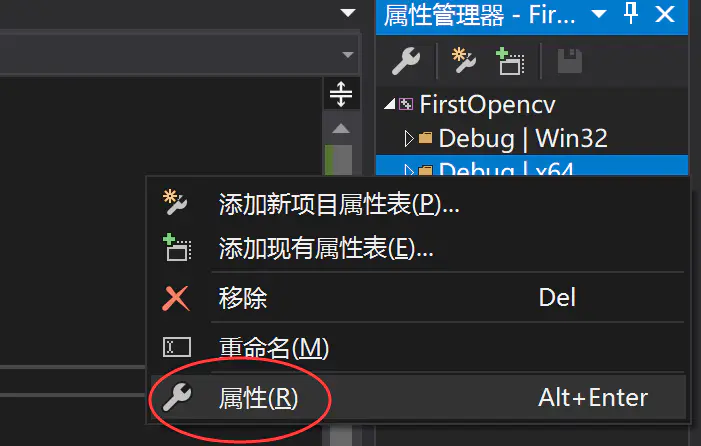
1. 将菜单栏下的Debug后的x86改为x64



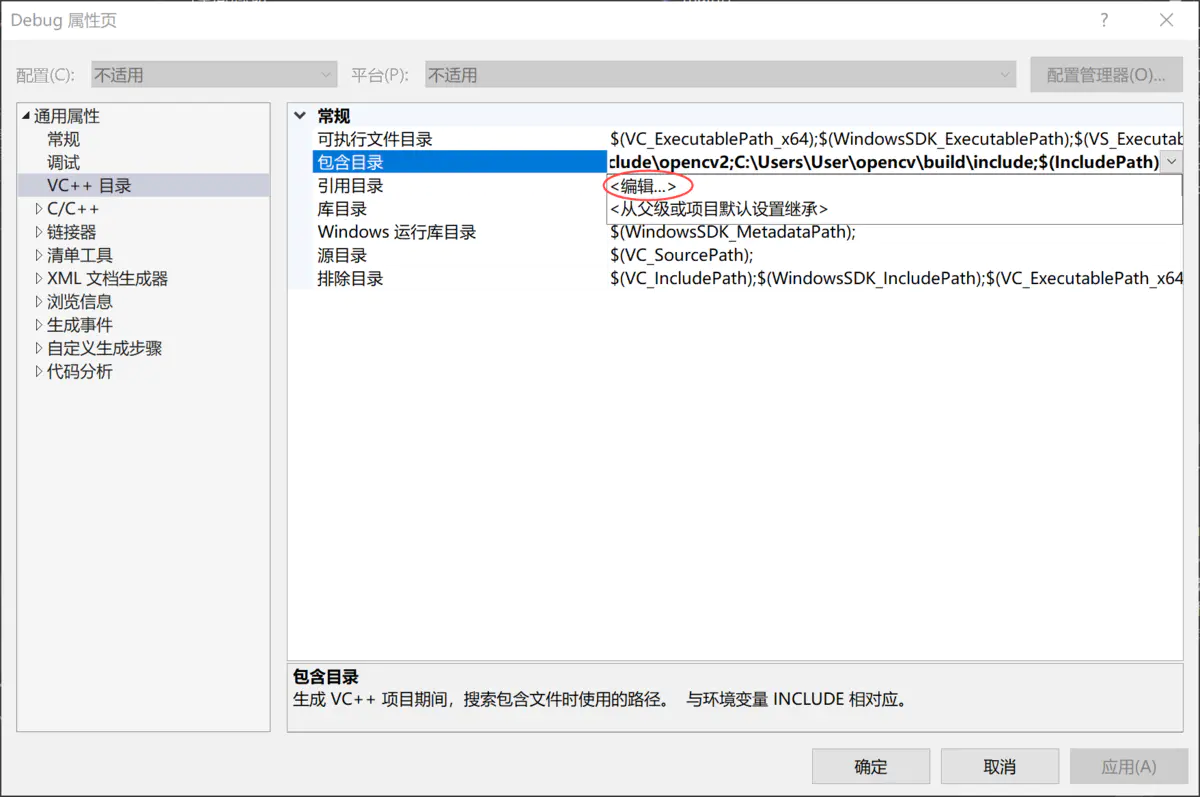
1. 打开菜单栏中的视图(View) -> 其他窗口(Other Windows) -> 属性管理器(Property Manager)



1. 右键Debug|x64 打开属性



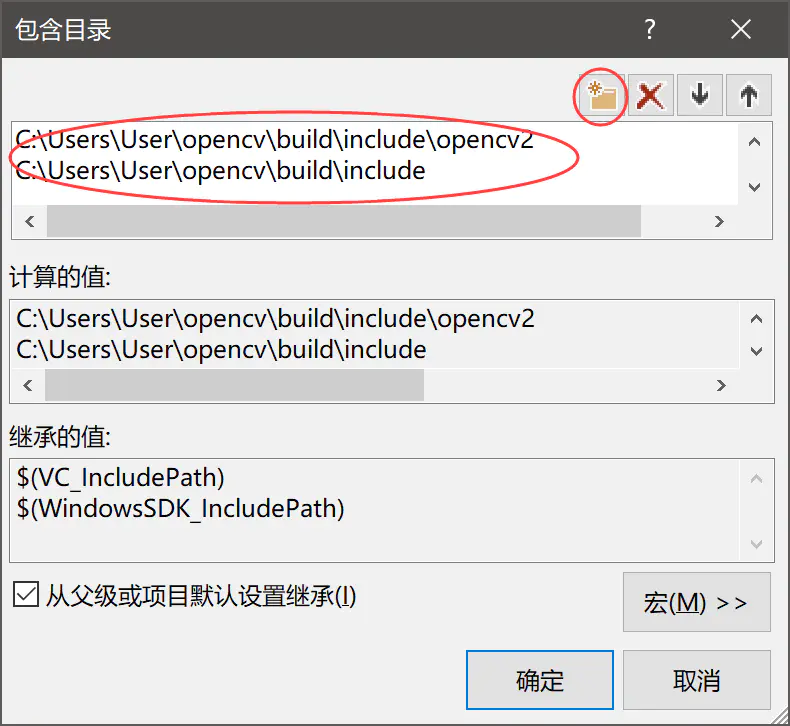
1. 编辑包含目录



1. 添加两条目录路径并确定（注意路径是你opencv安装的路径）

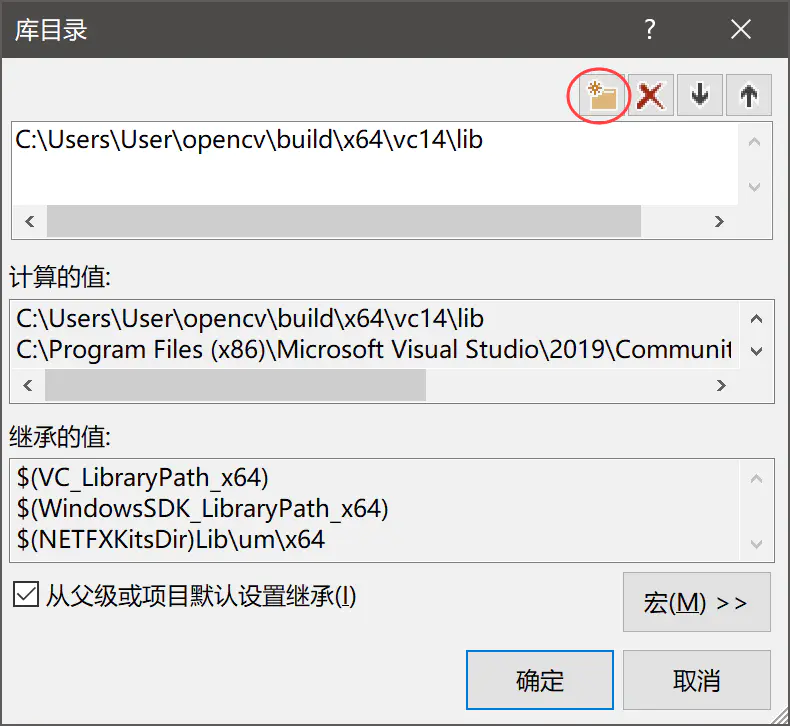
C:\Users\User\opencv\build\include

C:\Users\User\opencv\build\include\opencv2

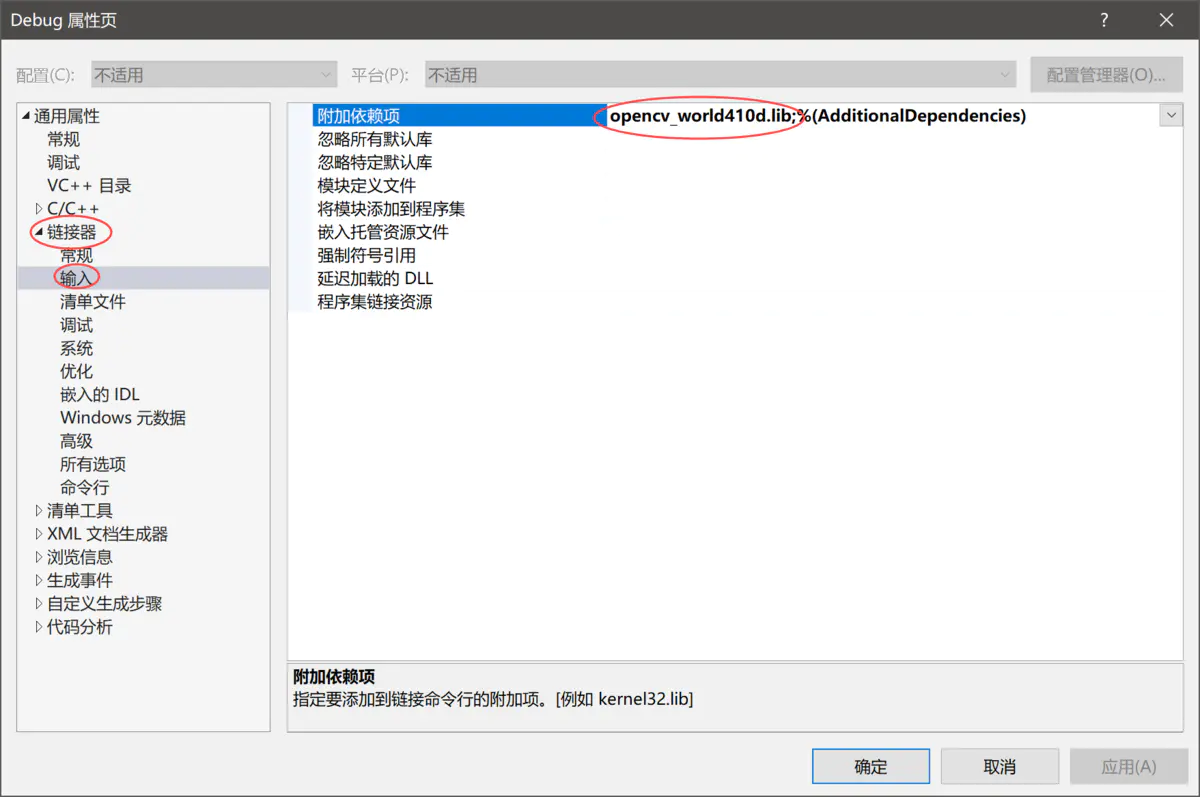


1. 同理，添加库目录，如图所示。

C:\Users\User\opencv\build\x64\vc14\lib



1. 添加附加依赖项。链接器->输入->附加依赖项

将opencv\_world341d.lib输入（注意opencv版本，3.4.1是341，其他版本同理）

## 相机标定流程

对相机的标定也就是求出相机的内参、外参以及畸变参数。使用OpenCV实现相机标定法的流程：

1. 准备标定图片

标定图片是固定相机位置，近距离拍摄标定板在不同位置、不同角度、不同姿态下的图片，理论上两张不同角度的图片就能够求解参数，为了提升精度，一般选用经验值20张。标定板需要是黑白相间的矩形构成的棋盘图，制作精度要求较高，如下图所示：

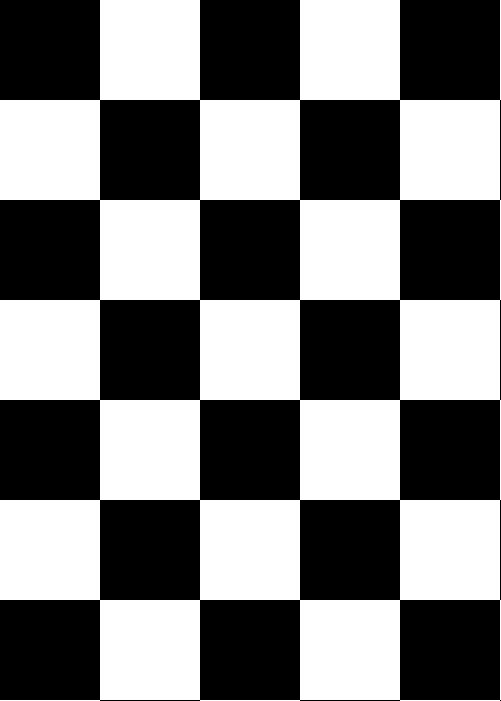


图6

1. 对每一张标定图片，提取角点信息

需要使用findChessboardCorners函数提取角点，这里的角点专指的是标定板上的内角点，这些角点与标定板的边缘不接触。

1. //! finds checkerboard pattern of the specified size in the image
2. CV\_EXPORTS\_W **bool** findChessboardCorners( InputArray image, Size patternSize,
3. OutputArray corners,
4. **int** flags=CALIB\_CB\_ADAPTIVE\_THRESH+CALIB\_CB\_NORMALIZE\_IMAGE );

其中， Image是棋盘图Mat图像，必须是8位的灰度或者彩色图像； patternSize是每个棋盘图上内角点的行列数，一般情况下，行列数不要相同，便于后续标定程序识别标定板的方向； corners用于存储检测到的内角点图像坐标位置； flags用于定义棋盘图上内角点查找的不同处理方式，有默认值。

1. 对每一张标定图片，进一步提取亚像素角点信息

为了提高标定精度，需要在初步提取的角点信息上进一步提取亚像素信息，降低相机标定偏差，常用的方法是cornerSubPix

1. //! adjusts the corner locations with sub-pixel accuracy to maximize the certain cornerness criteria
2. CV\_EXPORTS\_W **void** cornerSubPix( InputArray image, InputOutputArray corners,
3. Size winSize, Size zeroZone,
4. TermCriteria criteria );

另一个方法是使用find4QuadCornerSubpix函数，这个方法是专门用来获取棋盘图上内角点的精确位置的。

1. //! finds subpixel-accurate positions of the chessboard corners
2. CV\_EXPORTS **bool** find4QuadCornerSubpix(InputArray img, InputOutputArray corners, Size region\_size);
3. 在棋盘标定图上绘制找到的内角点用于观察

使用drawChessboardCorners函数绘制被成功标定的角点：

1. //! draws the checkerboard pattern (found or partly found) in the image
2. CV\_EXPORTS\_W **void** drawChessboardCorners( InputOutputArray image, Size patternSize,
3. InputArray corners, **bool** patternWasFound );
4. 相机标定

获取棋盘标定图的内角点图像坐标之后，就可以使用calibrateCamera函数进行标定：

1. //! finds intrinsic and extrinsic camera parameters from several fews of a known calibration pattern.
2. CV\_EXPORTS\_W **double** calibrateCamera( InputArrayOfArrays objectPoints,
3. InputArrayOfArrays imagePoints,
4. Size imageSize,
5. CV\_OUT InputOutputArray cameraMatrix,
6. CV\_OUT InputOutputArray distCoeffs,
7. OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs,
8. **int** flags=0, TermCriteria criteria = TermCriteria(
9. TermCriteria::COUNT+TermCriteria::EPS, 30, DBL\_EPSILON) );

其中输入：objectPoint和imagePoints分别是三维空间点和对应的二维图像点；cameraMatrix为相机内参矩阵，distCoeffs为相机畸变矩阵；输出：rvecs和tvecs分别是标定图像的旋转矩阵和平移矩阵的向量（vector<Mat>类型）；criteria为终止迭代的条件。

1. 对标定结果进行评价

通过计算得到的相机内参和外参，对三维空间点重新进行投影，得到标定后的图像坐标，计算图像坐标于原角点坐标之间的偏差，偏差越小则标定效果越好。

1. //! projects points from the model coordinate space to the image coordinates. Also computes derivatives of the image coordinates w.r.t the intrinsic and extrinsic camera parameters
2. CV\_EXPORTS\_W **void** projectPoints( InputArray objectPoints,
3. InputArray rvec, InputArray tvec,
4. InputArray cameraMatrix, InputArray distCoeffs,
5. OutputArray imagePoints,
6. OutputArray jacobian=noArray(),
7. **double** aspectRatio=0 );
8. 查看标定效果——利用标定结果对棋盘图进行矫正

使用undistort函数实现图像畸变矫正：

1. //! corrects lens distortion for the given camera matrix and distortion coefficients
2. CV\_EXPORTS\_W **void** undistort( InputArray src, OutputArray dst,
3. InputArray cameraMatrix,
4. InputArray distCoeffs,
5. InputArray newCameraMatrix=noArray() );

矫正的效果如图7。

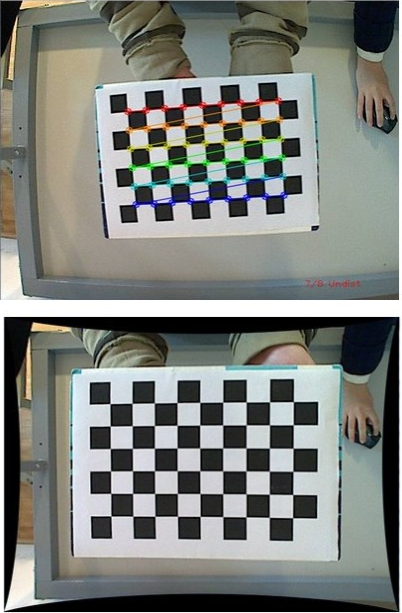
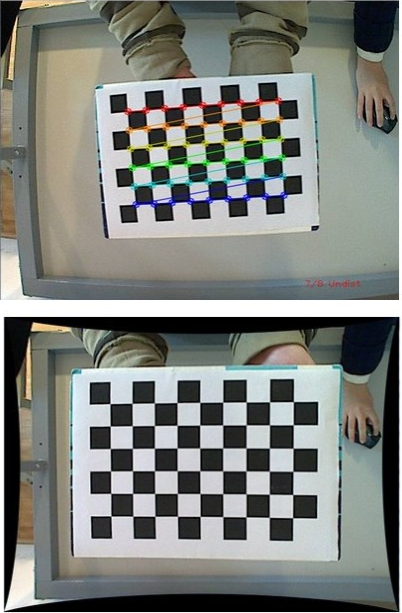


图7

## 相机标定注意事项

（1）棋盘格的选取

行列的格子数不要相同，便于后续标定程序识别标定板的方向，如图8。



图8

（2）标定图像的拍摄

1、为了提升精度，实验使用20张标定图像

2、棋盘平面和相机平面的夹角要小于45°

3、棋盘在图像中的比例不小于图像的20%

4、棋盘摆放角度和位置要尽可能的多样并覆盖整个图像

（3）打印设置

准备一张A4纸进行打印，为方便定标，注意棋盘方格边长可固定为诸如30mm、25mm等真实尺寸。自行查阅word文档的标尺设置方式。

# 单目相机三维点云（深度图像计算）重建实验

分别参考<https://blog.csdn.net/weixin_45358184/article/details/111645985>，和<https://blog.csdn.net/sinat_39411798/article/details/80453642>，使用标定好的相机参数，从两张图像，计算F矩阵，求解深度数据，形成三维点云。实验过程，分别使用已标定参数和不使用标定参数，验证两者重建的差异。

**3.1 三维重建过程**

(1) 相机标定获得相机内参数矩阵

(2) 特征点匹配

在计算机视觉中，特征提取和描述子匹配是一个基础的过程，并且用在许多方法中来执行各种各样的操作。例如，检测图像中一个目标的位置和方向，或者通过给出一个查询图像在大数据图像中找到相似的图像。从本质上讲，提取意味着在图像中选择点，使得获得好的特征，并且为它们计算一个描述子。一个描述子是含有多个数据的向量，用来描述在一个图像中围绕着特征点的周围环境。不同的方法有不同的长度和数据类型来表示描述子矢量。匹配是使用它的描述子从另外一个图像中找到一组与之对应的特征。OpenCV提供了非常简单和有效的方法支持特征提取和匹配。

1. SurfFeatureDetectordetector();
2. vector<KeyPoint> keypoints1, keypoints2;
3. detector.detect(img1, keypoints1);
4. detector.detect(img2, keypoints2);
5. // computing descriptors
6. SurfDescriptorExtractor extractor;
7. Mat descriptors1, descriptors2;
8. extractor.compute(img1, keypoints1, descriptors1);
9. extractor.compute(img2, keypoints2, descriptors2);
10. // matching descriptors
11. BruteForceMatcher<L2<float>> matcher;
12. vector<DMatch> matches;
13. matcher.match(descriptors1, descriptors2, matches);
14. 计算基础矩阵F和本征矩阵E

本征矩阵E，是单几何意义上的，与成像仪无关，它将左摄像机观测到的点P的物理坐标和右摄像机观测到的相同的点的位置关联起来。

基础矩阵F，则是将一台摄像机的像平面上的点在图像坐标（像素）上的坐标和另一台摄像机的像平面上的点关联起来。

OpenCV提供findFundamentalMat函数找到基础矩阵，K为相机的内参矩阵。  
Mat\_<double> E = K.t() \* F \* K;

1. 根据本征矩阵的旋转和平移分量构造投影矩阵对P0和P1

根据本征矩阵E用SVD分解得到旋转矩阵R和平移向量t，检查旋转矩阵R是否有效，根据旋转矩阵的标准正交特性判断旋转矩阵的有效性。然后在旋转矩阵有效的情况下构造投影矩阵P0和P1。

1. 有效特征点三角化实现重建

 已知（u,v）为空间点在物理成像平面中的坐标，则（u,v,1）为空间点在物理成像平面中的齐次坐标。根据两个不同的归一化图像平面中的坐标u和u1，以及投影矩阵P0和P1，可以得到该点的三维空间坐标。  
Mat\_<double> X = LinearLSTriangulation(u,P,u1,P1);

# 考核评价

任务：相机标定；三维重建

评分标准：

1. A+：任务全部实现，实验报告详细，配图清晰；对三维重建结果进行初步的比较，分析。
2. A：任务全部实现
3. 少实现一个任务，评分下降一级；