**实验二 视觉传感器标定**

**【实验目的】**

1、了解开源图像处理库OpenCV的结构，掌握OpenCV的基本使用方法。

2、了解开源图像处理库OpenCV的基本模块功能，掌握常用图像处理方法。

3、掌握摄像机标定算法，学会使用OpenCV进行摄像机标定。

**【实验性质】**

验证性实验。

**【实验要求】**

1、C++集成开发环境（QT5或VS2015及以上）

2、OpenCV 4.x

3、单/双目摄像机

**【实验内容】**

1、掌握OpenCV开源图像库的基本使用方法

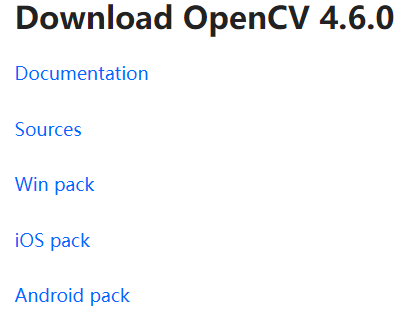
2、使用OpenCV进行单目摄像机标定

3、使用OpenCV进行双目摄像机标定

**【实验步骤】**

1、下载、安装、OpenCV 4.x

官网下载：https://opencv.org/opencv-4-6-0/



可根据操作系统平台选择相应版本下载。

2、摄像机标定原理

摄像机标定的目标，是要建立三维世界坐标系与二维图像坐标系之间的对应关系。在单目视觉中，这种对应关系是一对多的，即二维图像中的一个像素点对应着三维空间中的一条直线；在双目视觉中，可以通过两幅二维图像上的对应像素点计算得到三维世界坐标系与二维图像坐标系的一一对应关系。由此便可以得到物体的三维坐标值。

摄像机安装位置参数为外部参数，摄像机镜头畸变参数为非线性模型内部参数。标定过程即是求出摄像机的内部参数和外部参数，从而得到上述坐标转换的旋转矩阵和平移向量。

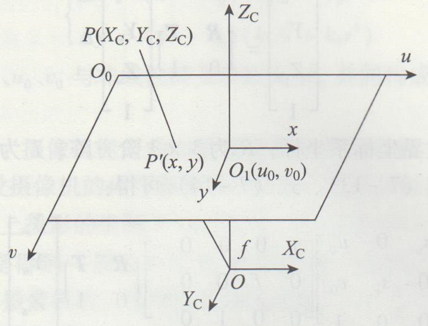


图2-1 摄像机成像模型

图中，摄像机坐标系为OXcYcZc，而计算机图像坐标系O0uv与图像平面坐标系O1xy的转换关系为：

u = sux + u0

v = svy + v0

写成矩阵形式为：

其中，su、sv分别为x、y轴方向单位长度对应的像素点数；u0、v0为镜头光学中心位置，单位为mm。

图像平面坐标系O1xy与摄像机坐标系OXcYcZc之间的关系可以通过共线方程得到：

将上述投影关系用矩阵形式表达如下：

上述参数及焦距f为摄像机线性模型内部参数。

摄像机坐标系与世界坐标系通过旋转矩阵R和平移向量T来转换。

其中，、、为世界坐标系，R为3X3阶方阵；T为3X1维平移向量矩阵。

其中，，与、、、、等摄像机内部参数有关，

，与旋转矩阵R和平移向量T等摄像机外部参数有关，A为3X4变换矩阵。

实际上，摄像机镜头成像并不是理想的透视成像，而是存在径向变形、偏心变形、薄棱镜变形等因素影响下产生不同程度的畸变。由于畸变因素作用，使空间点成像并不在线性模型描述的位置，而是在受到镜头失真影响而偏移的平面坐标。

其中，、为非线性畸变值，与像点在图像中的位置有关。一般镜头畸变同时存在径向畸变和切向畸变，切向畸变是由于透镜与摄像头传感器平面或图像平面不平行而产生的，多是由于透镜安装到镜头模组上的偏差导致，通常较小。实际应用中多只考虑径向畸变，径向畸变常用距图像中心径向距离的泰勒级数展开的前几项来表示：

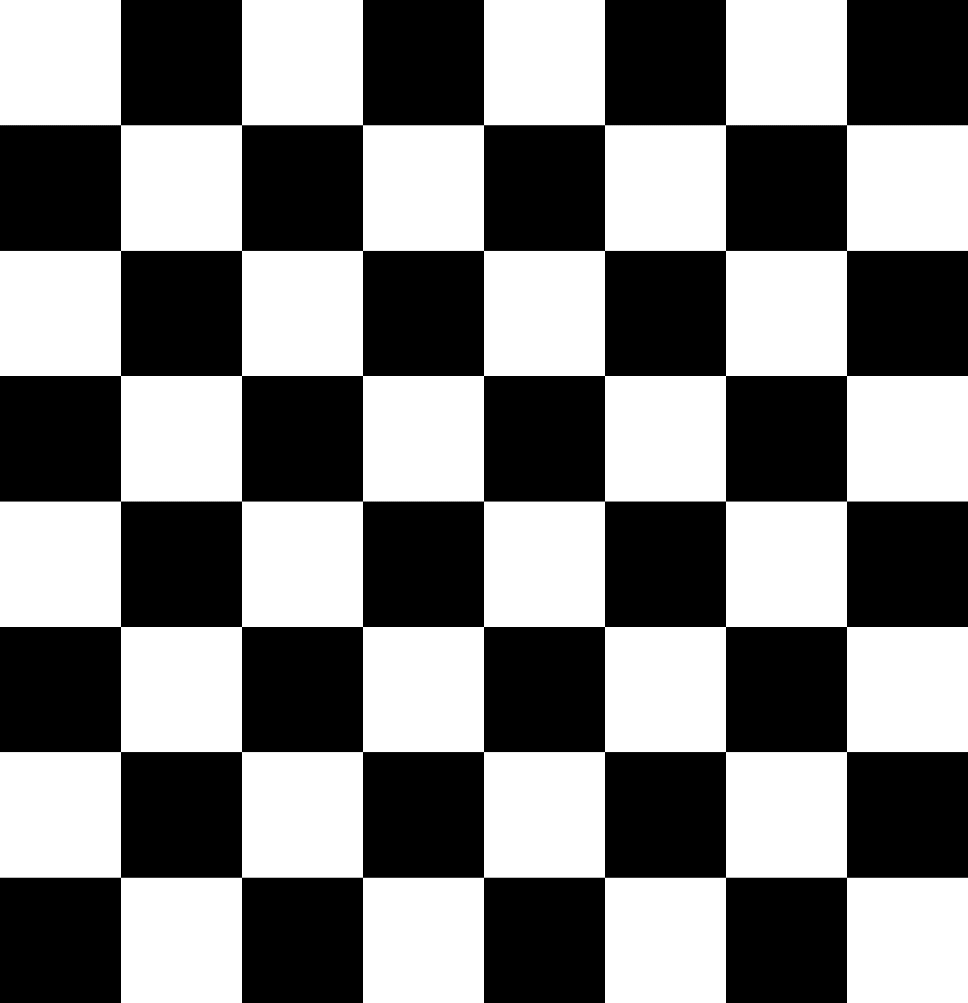
、为非线性畸变参数，。

摄像机的标定方法很多，大致可分为传统标定技术和自标定技术。其中，传统标定技术需要在摄像机前放置一个特定的标定物，并人为地提供一组已知坐标的特征基元，摄像机通过寻找这些已知特征的基元来实现标定。它用到射影几何方面的理论，是一种直接计算摄像机模型的方法。自标定技术则比较灵活，不需要特定的参照物，它利用环境的刚体性，通过对比多幅图像中的对应点来计算摄像机模型。就精度而言，还无法与传统标定技术相比。

OpenCV采用介于传统标定方法和自标定方法之间的一种方法，由张正友提出。这种方法不需要知道摄像机运动的具体信息，比传统标定方法更灵活，同时仍需要一个特定的标定物以及一组已知的特征基元的坐标，这一点不如自标定灵活。它通过在至少3个不同的位置获取标定物的图像，计算出摄像机所有的内外参数。

标定算法描述参考知乎：https://zhuanlan.zhihu.com/p/36371959

OpenCV源码在其sample/data目录下面一个自带的棋盘图(chessboard.png)，显示如下：



在标定的时候，算法要求提供的棋盘格的宽度与高度，还有他们的间隔距离。需要特别注意是这里的宽高是指他们的内部交叉点的个数，以上图为例，它的大小为7x7而不是8x8。间隔是指棋盘格之间的距离，可以用像素距离表示，也可以用实际毫米为单位表示。

3、标定常用函数

（1）findChessboardCorners() 棋盘格角点检测

bool findChessboardCorners( InputArray image, Size patternSize, OutputArray corners,

int flags = CALIB\_CB\_ADAPTIVE\_THRESH + CALIB\_CB\_NORMALIZE\_IMAGE );

第一个参数是输入的棋盘格图像（可以是8位单通道或三通道图像）；

第二个参数是棋盘格内部的角点的行列数（注意：不是棋盘格的行列数，如棋盘格的行列数分别为4、8，而内部角点的行列数分别是3、7，因此这里应该指定为cv::Size(3, 7)）；

第三个参数是检测到的棋盘格角点，类型为std::vectorcv::Point2f。

第四个参数flag，用于指定在检测棋盘格角点的过程中所应用的一种或多种过滤方法，可以使用下面的一种或多种，如果都是用则使用OR：

cv::CALIB\_CB\_ADAPTIVE\_THRESH：使用自适应阈值将图像转化成二值图像

cv::CALIB\_CB\_NORMALIZE\_IMAGE：归一化图像灰度系数(用直方图均衡化或者自适应阈值)

cv::CALIB\_CB\_FILTER\_QUADS：在轮廓提取阶段，使用附加条件排除错误的假设

cv::CALIB\_CV\_FAST\_CHECK：快速检测

（2）cv::drawChessboardCorners() 棋盘格角点的绘制

drawChessboardCorners( InputOutputArray image, Size patternSize,

InputArray corners, bool patternWasFound );

image源图像

patternSize，每一行每一列的角

corners，已经检测到的角

patternWasFound，findChessboardCorners的返回值

（3）find4QuadCornerSubpix() 对粗提取的角点进行精确化

find4QuadCornerSubpix(InputArray img, InputOutputArray corners, Size region\_size );

image源图像

corners，提供角点的初始坐标

region\_size： 搜索窗口的一般尺寸

（4）cornerSubPix() 亚像素检测

void cornerSubPix( InputArray image, InputOutputArray corners, Size winSize,

Size zeroZone, TermCriteria criteria );

image源图像

corners，提供角点的初始坐标，返回更加精确的点

winSize，搜索窗口的一般尺寸，如果winSize=Size(5,5),则search windows为11\*11

winSize,死区的一般尺寸，用来避免自相关矩阵的奇点，(-1,-1)表示没有死区

criteria，控制迭代次数和精度

（5）calibrateCamera() 求解摄像机的内在参数和外在参数

double calibrateCamera( InputArrayOfArrays objectPoints,

InputArrayOfArrays imagePoints,

Size imageSize,

InputOutputArray cameraMatrix,

InputOutputArray distCoeffs,

OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs,

int flags = 0,

TermCriteria criteria = TermCriteria(TermCriteria::COUNT +

TermCriteria::EPS, 30, DBL\_EPSILON) );

objectPoints，世界坐标，用vector<vector>，输入x,y坐标，z坐标为0

imagePoints，图像坐标，vector<vector>

imageSize，图像的大小用于初始化标定摄像机的image的size

cameraMatrix，内参数矩阵

distCoeffs，畸变矩阵

rvecs，位移向量

tvecs，旋转向量

flags，可以组合：

* CV\_CALIB\_USE\_INTRINSIC\_GUESS：使用该参数时，将包含有效的fx,fy,cx,cy的估计值的内参矩阵cameraMatrix，作为初始值输入，然后函数对其做进一步优化。如果不使用这个参数，用图像的中心点初始化光轴点坐标(cx, cy)，使用最小二乘估算出fx，fy（这种求法好像和张正友的论文不一样，不知道为何要这样处理）。注意，如果已知内部参数（内参矩阵和畸变系数），就不需要使用这个函数来估计外参，可以使用solvepnp()函数计算外参数矩阵。
* CV\_CALIB\_FIX\_PRINCIPAL\_POINT：在进行优化时会固定光轴点，光轴点将保持为图像的中心点。当CV\_CALIB\_USE\_INTRINSIC\_GUESS参数被设置，保持为输入的值。
* CV\_CALIB\_FIX\_ASPECT\_RATIO：固定fx/fy的比值，只将fy作为可变量，进行优化计算。当 CV\_CALIB\_USE\_INTRINSIC\_GUESS没有被设置，fx和fy的实际输入值将会被忽略，只有fx/fy的比值被计算和使用。
* CV\_CALIB\_ZERO\_TANGENT\_DIST：切向畸变系数（P1，P2）被设置为零并保持为零。
* CV\_CALIB\_FIX\_K1,…,CV\_CALIB\_FIX\_K6：对应的径向畸变系数在优化中保持不变。如果设置了CV\_CALIB\_USE\_INTRINSIC\_GUESS参数，就从提供的畸变系数矩阵中得到。否则，设置为0。
* CV\_CALIB\_RATIONAL\_MODEL（理想模型）：启用畸变k4，k5，k6三个畸变参数。使标定函数使用有理模型，返回8个系数。如果没有设置，则只计算其它5个畸变参数。
* CALIB\_THIN\_PRISM\_MODEL （薄棱镜畸变模型）：启用畸变系数S1、S2、S3和S4。使标定函数使用薄棱柱模型并返回12个系数。如果不设置标志，则函数计算并返回只有5个失真系数。
* CALIB\_FIX\_S1\_S2\_S3\_S4 ：优化过程中不改变薄棱镜畸变系数S1、S2、S3、S4。如果cv\_calib\_use\_intrinsic\_guess设置，使用提供的畸变系数矩阵中的值。否则，设置为0。
* CALIB\_TILTED\_MODEL （倾斜模型）：启用畸变系数tauX and tauY。标定函数使用倾斜传感器模型并返回14个系数。如果不设置标志，则函数计算并返回只有5个失真系数。
* CALIB\_FIX\_TAUX\_TAUY ：在优化过程中，倾斜传感器模型的系数不被改变。如果cv\_calib\_use\_intrinsic\_guess设置，从提供的畸变系数矩阵中得到。否则，设置为0。

（6）initUndistortRectifyMap() 计算畸变参数

void initUndistortRectifyMap(InputArray cameraMatrix,

InputArray distCoeffs,

InputArray R,

InputArray newCameraMatrix,

Size size,

int m1type,

OutputArray map1,

OutputArray map2)

cameraMatrix，摄像机内参数矩阵

distCoeffs， 摄像机的5个畸变系数，(k1,k2,p1,p2[,k3[,k4,k5,k6]])

R，在客观空间中的转换对象

newCameraMatrix，新的3\*3的浮点型矩矩阵

size，为失真图像的大小

m1type，第一个输出的map，类型为CV\_32FC1或CV\_16SC2

map1，x映射函数

map2，y映射函数

4、单目摄像机标定

（1）打开标定配置参数文件default.xml，设置参数：

* 棋盘格的宽度和高度（两个方向的角点数量），根据实际情况设置。

<!-- Number of inner corners per a item row and column. (square, circle) -->

<BoardSize\_Width>9</BoardSize\_Width>

<BoardSize\_Height>6</BoardSize\_Height>

* 每格的宽度

单元格的宽度应设置为实际的毫米数。

<!-- The size of a square in some user defined metric system (pixel, millimeter)-->

<Square\_Size>50</Square\_Size>

* 选择输入方式

程序提供了3种输入方式。如果摄像机已连接电脑，可以使用input\_camera方式。该方式只需要设置视频输入设备号，对于笔记本而言，通常0表示笔记本内置摄像头，1表示外置摄像头

<!-- The input to use for calibration.

To use an input camera -> give the ID of the camera, like "1"

To use an input video -> give the path of the input video, like "/tmp/x.avi"

To use an image list -> give the path to the XML or YAML file containing the list of

the images, like "/tmp/circles\_list.xml"

-->

<Input>"0"</Input>

* 编译OpenCV标定程序

（2）标定程序将实现以下功能

* 确定失真矩阵
* 确定摄像机矩阵
* 摄像机、视频和图像文件列表的输入
* 从XML/YAML文件进行配置
* 将结果保存到XML/YAML文件中
* 计算重投影误差

可以在OpenCV源代码库的samples/cpp/tutorial\_code/calib3d/camera\_calibration/文件夹中找到源代码。程序只有一个参数。其配置文件的名称。如果没有，它将尝试打开一个名为“default.xml”。

（3）运行标定程序，摄像机将拍摄25幅图片并识别角点，如图2



（4）标定结果保存在程序中指定的结果文件中

请在实验报告中说明文件中各项数据的意义。

5、双目摄像机标定

**【扩展思考】**

参考下面的内容，思考如何对图片进行校正？

cv::Mat cb\_final;

cv::Mat mapx = cv::Mat(cb\_source.size(), CV\_32FC1);

cv::Mat mapy = cv::Mat(cb\_source.size(), CV\_32FC1);

cv::Mat R = cv::Mat::eye(3, 3, CV\_32F);

undistort(cb\_source, cb\_final, cameraMatrix, distCoeffs);

cv::imwrite("../blizzard/res/calibration/cb\_final.png", cb\_final);

**【补充说明】**

1、OpenCV及OpenCV\_contrib模块功能

(1) OpenCV

* calib3d——主要包含相机标定和立体视觉等功能，例如物体位姿估计、三维重建、摄像机标定等。
* core——核心功能模块，主要包含opencv库的基础结构和基本操作，例如opencv基本数据结构、绘图函数、数组操作相关函数、动态数据结构等。
* dnn——深度学习模块，主要包括构建神经网络、加载序列化网络模型等。
* features2d——功能主要为处理图像特征点，例如特征检测、描述与匹配等。
* flann——Fast Library for Approximate Nearest Neighbors（快速近似最近邻库）的缩写，是高维的近似近邻快速搜索算法库，主要包含快速近似近邻搜索与聚类等。
* gapi——opencv4中新加模块，用来加速常规的图像处理，主要充当框架，并不是特定的计算机视觉算法。
* highgui——高层GUI，包含创建和操作显示图像的窗口、处理鼠标事件和键盘命令、提供图形交互可视化界面等。、
* imgcodecs——图像文件的读取与保存模块，主要用于图像文件的读取与保存，
* imgproc——图像处理模块，主要包括图像滤波、几何变换、直方图、特征检测与目标检测等。
* ml——机器学习模块，主要为统计分类、回归和数据聚类等。
* objdetect——目标检测模块，主要用于图像目标检测。
* photo——计算摄影模块，主要包含图像修复与去噪等。
* stiching——图像拼接模块，主要包含特征点寻找与匹配图像、估计旋转、自动校准、接缝估计等图像拼接相关内容。
* video——视频分析模块，主要包含运动估计、背景分离、对象跟踪等内容。
* videoio——视频文件读写模块，主要用于读取/写入视频或者图像序列。

(2) opencv\_contrib（扩展包）

* aruco——一个检测二进制marker的库。
* bgsegm——改进的背景-前景分割方法。
* bioinspired——受生物学启发的视觉模型和衍生工具，包含生物视觉系统模型（人类视觉系统和其他），它还提供利用这些仿生模型的衍生对象。
* ccalib——用于 3D 重建的自定义校准模式。
* cnn\_3dobj——3D目标识别和姿态估计API。
* cuda——GPU处理图像，CUDA-加速的计算机视觉算法，包含以下部分：
* cudaarithm——矩阵运算。
* cudabgsegm——背景分割。
* cudacodec——视频编/解码。
* cudafeatures2d——特征检测与描述。
* cudafilters——图像滤波，用于对2D图像执行各种线性或非线性滤波操作。
* cudaimgproc——图像处理。
* cudalegacy——传统支持
* cudaobjdetect——目标检测
* cudaoptflow——光流算法
* Cudastereo——CUDA-加速的立体视觉匹配算法
* cudawarping——CUDA快速图像变换
* cudev——设备层，实现CUDA版本的核心功能
* cvv——计算机视觉程序交互式可视化调试GUI。
* datasets——处理不同数据集的框架，用于处理不同数据集的类：加载数据、评估不同算法、包含基准测试等。
* dnn\_objdetect——用于目标检测的DNN。
* dnn\_superres——用于超分辨率的DNN，包含通过卷积神经网络放大图像的功能。
* dpm——基于可变形零件的模型。
* face——人脸分析。
* freetype——使用 freetype/harfbuzz 绘制 UTF-8 字符串。
* fuzzy——基于模糊数学的图像处理。
* hdf——分层数据格式 I/O 例程。
* hfs——用于高效图像分割的分层特征选择，包含一种高效的图像分割算法。
* img\_hash——提供提取图像散列的算法，以及快速计算出海量数据集中最相似图像的方法。
* line\_descriptor——在图像中寻找一种新的代表性信息，并为其提取和表示提供功能。特别地，与先前用于检测图像内相关元素的方法不同，提取线代替点；定义了一个新类，用于汇总线的特性，以便于重用和打印。
* optflow——光流算法。
* ovis——OGRE 3D 可视化。
* phase\_unwrapping——相位展开的API。
* plot——用于Mat 数据的绘图函数。
* quality——图像质量分析（IQA）API。
* reg——实现了参数化图像配准。实现的方法是直接对齐。
* rgbd——RGB-深度处理。
* saliency——提供一个独特的接口、一个独特的使用框架和插件服务器显著性算法。
* sfm——包含从2d图像执行3d重建的算法。
* shape——形状距离与匹配。
* stereo——立体对应算法。
* structured\_light——结构光API。
* superres——超级分辨率。
* surface\_matching——面匹配。
* text——场景文本检测和识别。
* tracking——追踪API，可用于插入多种算法并对其进行比较。
* videostab——包含一组可用于解决视频稳定问题的函数和类。
* viz——三维可视化工具。
* xfeatures2d——额外2D功能框架，对应前面的feature2d模块。
* ximgproc——扩展图像处理，对应前面的imgproc模块。
* xobjdetect——拓展目标检测，对应前面的objdetect模块。
* xphoto——其他照片处理算法，对应前面的photo模块。