

# Lecture 9

BanGVision! It's MyCode!!!

# 工业相机成像原理

## 光电效应

金属被光线照射时，会产生电子

光强越大，电子越多。

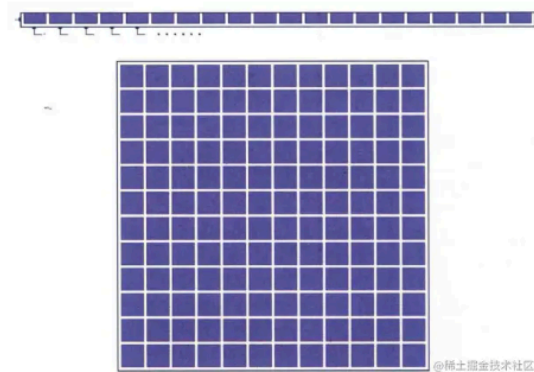
## 光电二极管

被光照射时能溢出电子的设备

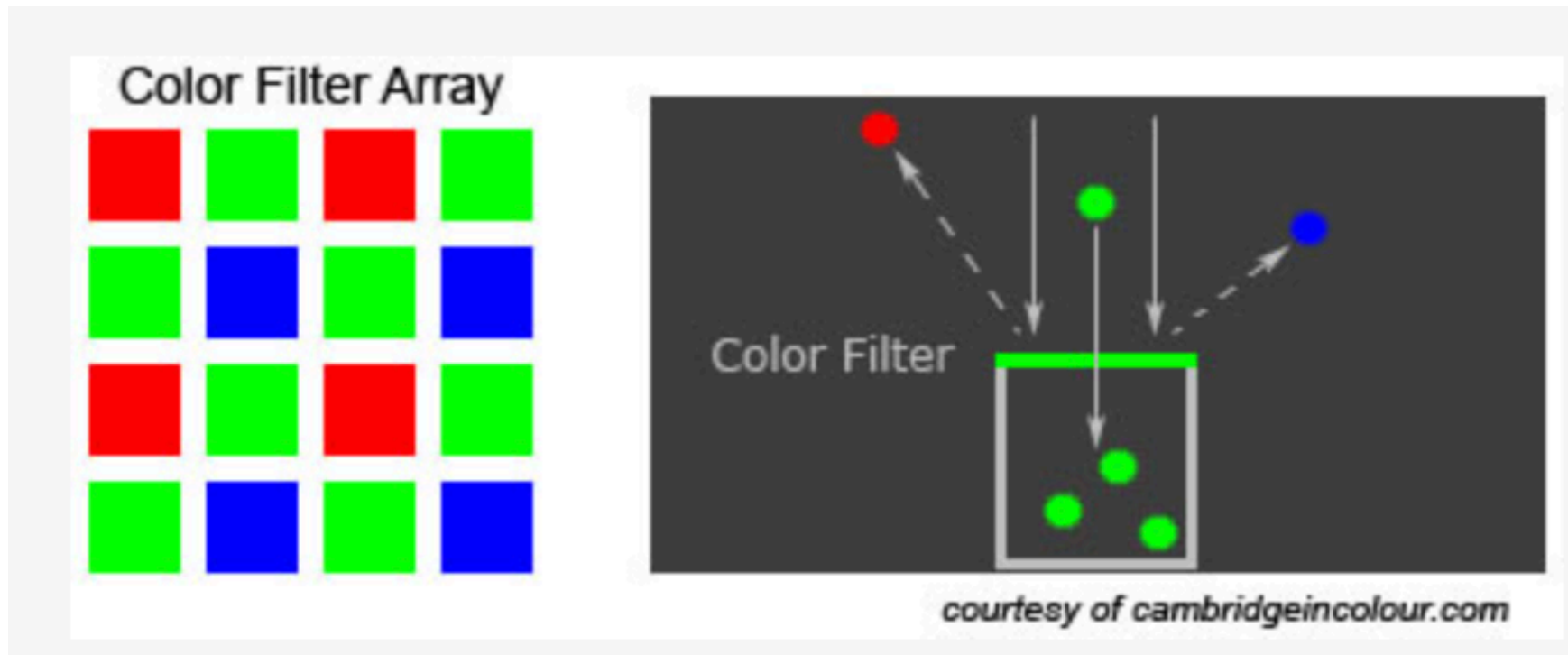
# 阵列传感器

由多个传感器构成的传感器阵列

- 势阱：储存光电二极管溢散电子的设备
- 颜色滤镜：透过指定颜色光的滤镜，一般由染料或薄膜干涉实现。

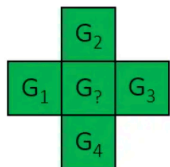


# Bayer 传感器阵列



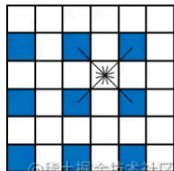
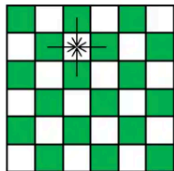
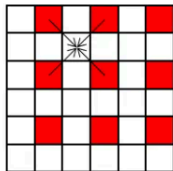
这样，每个传感器能够获取一种颜色的光强。以此生成的图片叫做raw图

# RGB 图像的合成: 去马赛克

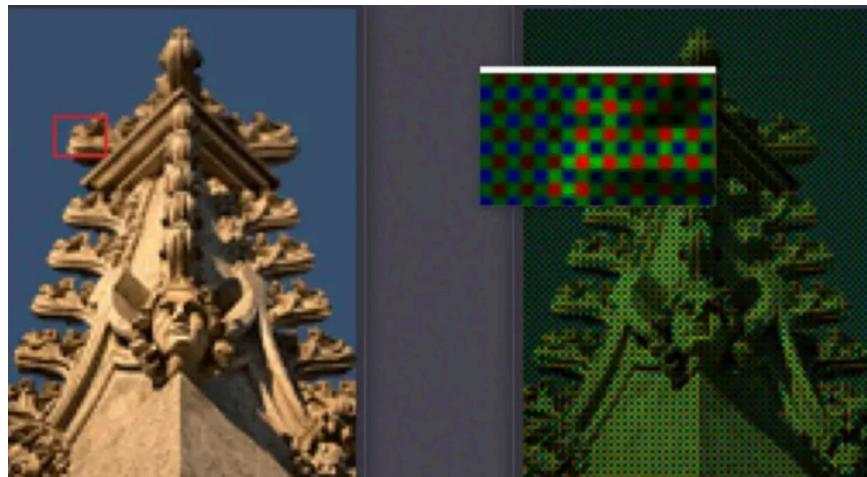


$$G_? = \frac{G_1 + G_2 + G_3 + G_4}{4}$$

@稀土掘金技术社区



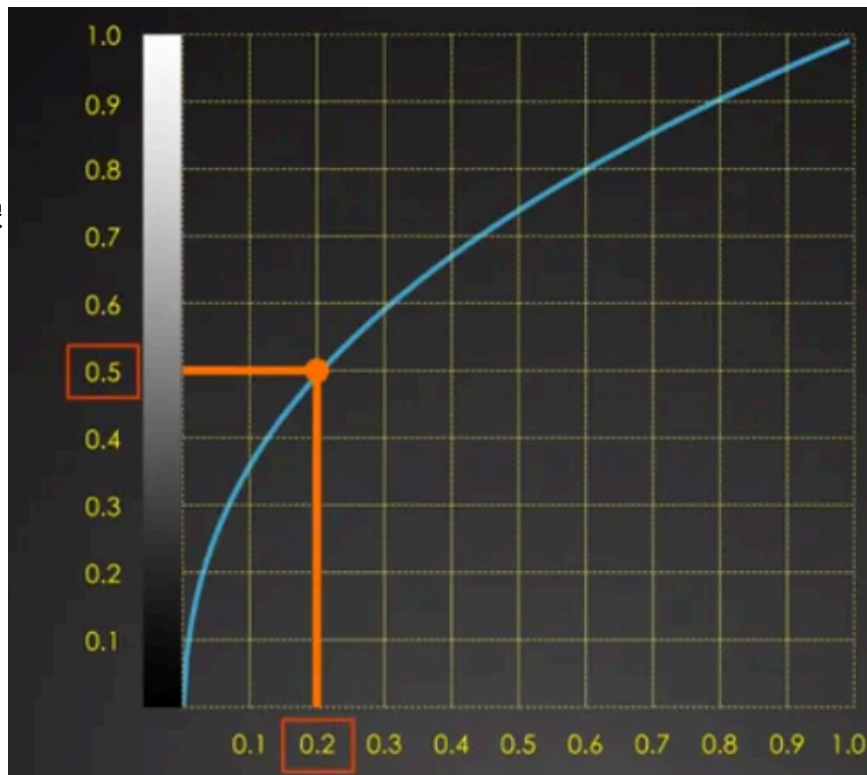
@稀土掘金技术社区



# 亮度平衡： gamma校正

人眼对与暗处的细节更加敏感。

如果令图像亮度（灰度）与光强呈线形关系，会让图像看起来更暗。



# 重要参数

**靶面尺寸**：传感器最大尺寸。一般表示方法为斜对角线长度(如1/2.8'表示传感器斜对角线长度为2.8分之一英寸) **镜头靶面尺寸应该至少不小于传感器的尺寸。**

**动态范围**：相机能够捕捉的最大光强。一般与势阱最大容量有关

**焦距**：平行光入射时，光线汇聚处与镜头中心的距离。

**曝光时间**：每一帧图像上，每一个传感器接收光线的时间。曝光时间越长，二极管产生的电荷越多，图像越亮

**增益**：传感器读取势阱电压，转为数字信号时，乘上的系数。

**gamma**：gamma矫正时，衡量曲线的参数。gamma越大，图像越亮，越关注暗处细节

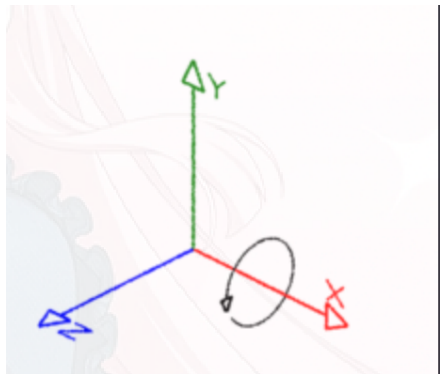
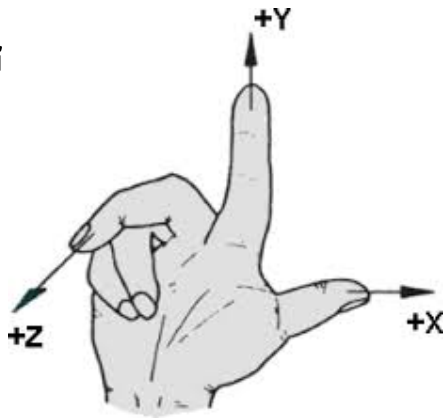


# 相机标定

# 右手坐标系

以右手手心为坐标原点，大拇指朝右，食指朝上，中指对着自己

- 大拇指为x轴
- 食指为y轴
- 中指为z轴



# 坐标转换

## 参数定义规范

$${}^A_Bx$$

$A$  表示相对坐标系， $B$  表示当前坐标系， $x$  表示参量，这里指  $x$  坐标。

这里表示  $B$  坐标系相对于  $A$  坐标系的  $x$  坐标值。

对于  ${}^A_BT$  这个坐标变换阵，采用矩阵左乘的计算方式，表示的是从坐标系  $A$  到坐标系  $B$  的坐标系变换。

三维坐标变换阵的通式如下：

$$T = \begin{bmatrix} R_{3 \times 3} & t_{3 \times 1} \\ O_{1 \times 3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

坐标变换阵主要由 **旋转矩阵**(rotation matrix)  $R$  和 **平移向量**(translation)  $t$  组成，此外还有一部分齐次坐标。

其中**旋转矩阵**  $R$  是一个  $3 \times 3$  的正交矩阵，描述在三维空间中的旋转。旋转矩阵具有以下性质：

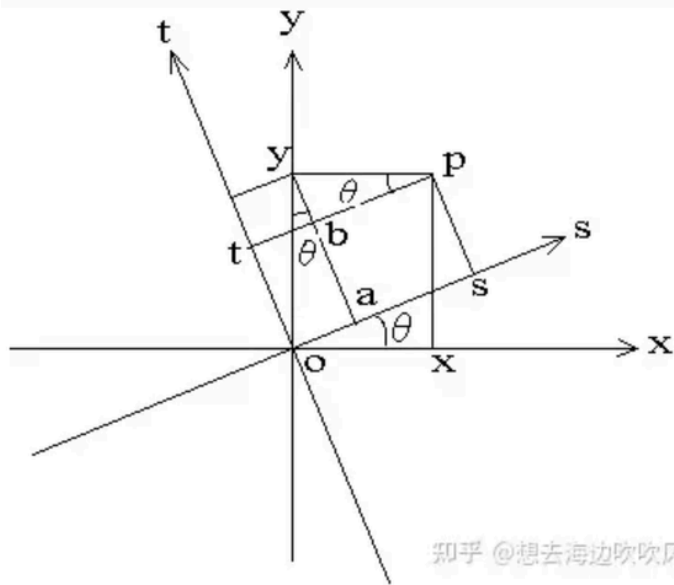
- **正交性**： $R^T R = I$ ，即矩阵的转置等于其逆。
- **行列式为 1**： $|R| = 1$ ，表示旋转不改变空间的体积。

**旋转矩阵表示的是两个坐标系之间坐标轴的空间指向的旋转关系。**

# 坐标转换

常见的旋转矩阵可以通过绕坐标轴旋转的方式构造

2维平面的坐标系旋转 $\theta$ 度示例：



- Example: 绕  $x$ 、 $y$ 、 $z$  轴的三维旋转矩阵分别为：

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 坐标转换

平移向量  $t$  是一个  $1 \times 3$  的列向量，表示两个坐标的坐标原点之间的平移关系。它可以表示为：

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

一个简单的例子如下，通过  ${}^R_F T$  完成了从扇叶坐标系  $F$  坐标值 到 机器人坐标系  $R$  坐标值的坐标变换。

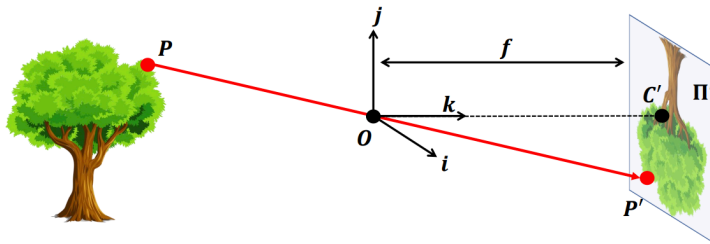
$$\begin{bmatrix} {}^R x \\ {}^R y \\ {}^R z \\ 1 \end{bmatrix} = {}^R_F T \begin{bmatrix} {}^F x \\ {}^F y \\ {}^F z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3 \times 3} & {}^R_F t_{3 \times 1} \\ O_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} {}^F x \\ {}^F y \\ {}^F z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^F x \\ {}^F y \\ {}^F z \\ 1 \end{bmatrix}$$

**齐次坐标** 为了将平移和旋转结合在一个统一的框架下，使用齐次坐标表示。齐次坐标的引入使得在数学上更方便地处理三维变换。齐次坐标将三维点  $(x, y, z)$  扩展为四维点  $(x, y, z, 1)$ 。

# 小孔相机 & 相机投影

- $f_x, f_y$ : 相机焦距 (why two focal length?)
- $X_c, Y_c, Z_c$ : 相机坐标系中的点
- $u, v$ : 成像平面上对应点的坐标
- $s$ : 缩放尺度。对齐像素坐标系与成像平面坐标系。

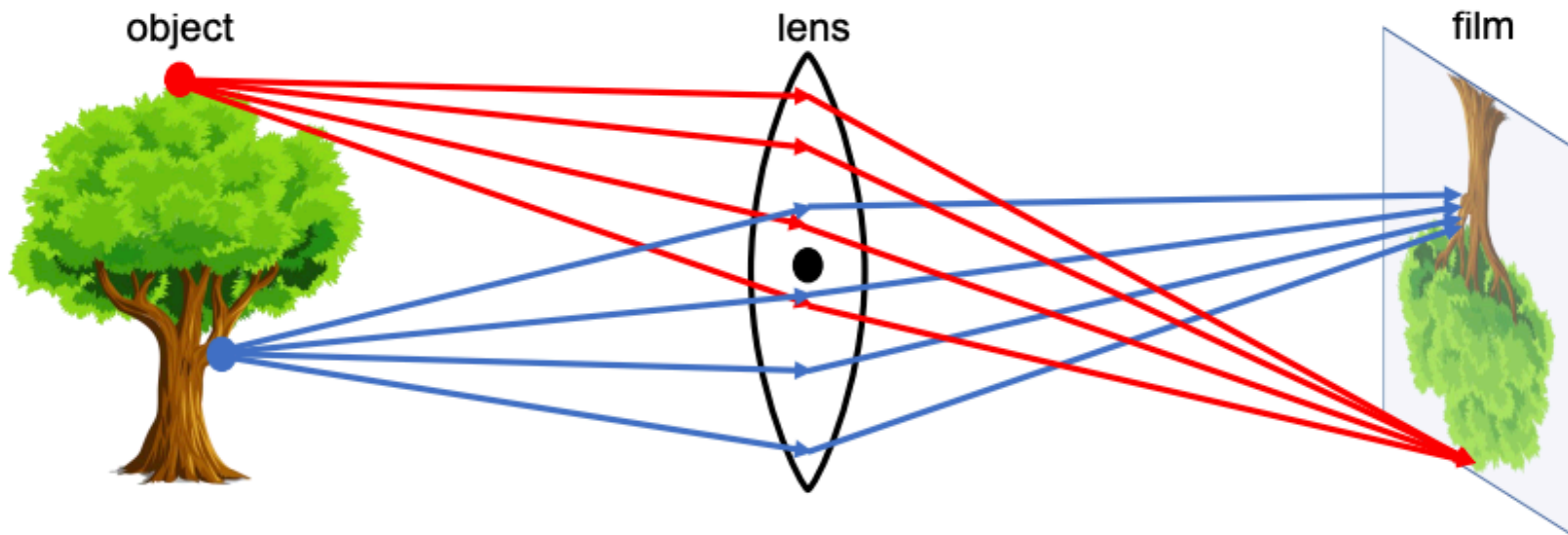
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x X_c / Z_c + c_x \\ f_y Y_c / Z_c + c_y \end{bmatrix}$$



$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}.$$

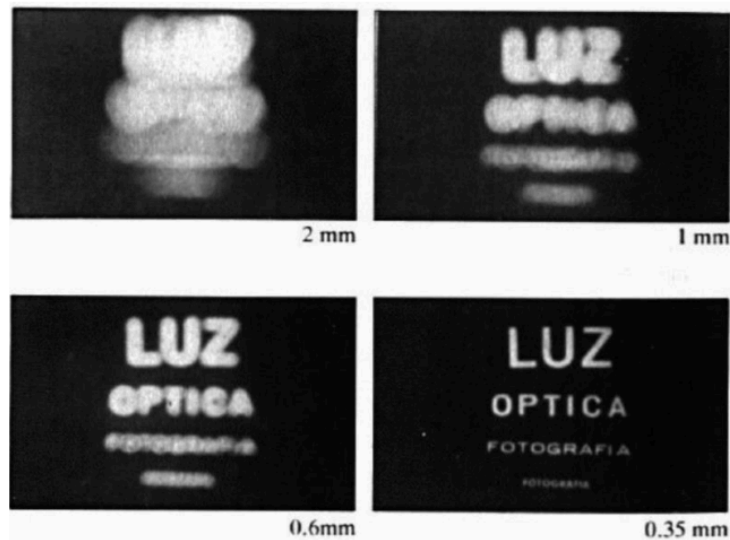
# 凸透镜成像



- 在小孔成像中, 成像越清晰, 透过小孔的光线越少, 画面越暗

# 凸透镜成像

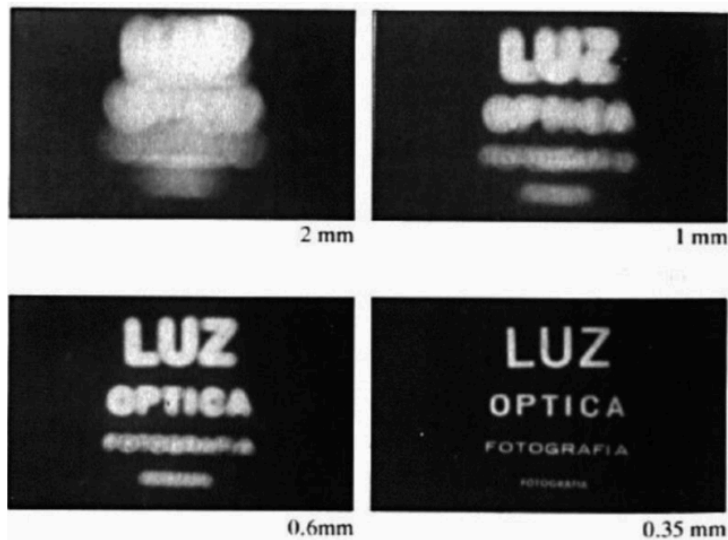
- 凸透镜成像时，需要保证成像平面位于光线汇聚处。  
(对于近距离物体，光线汇聚处不是焦距)



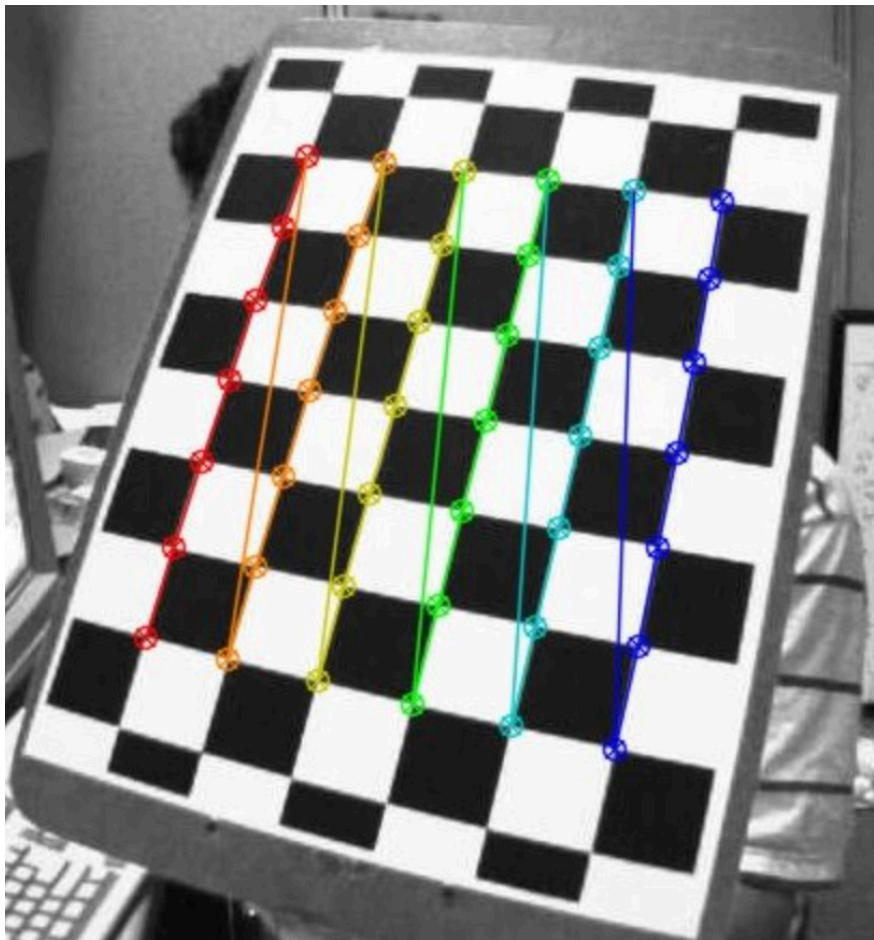


# 凸透镜成像

- 凸透镜成像时，需要保证成像平面位于光线汇聚处.  
(对于近距离物体，光线汇聚处不是焦距)
- 景深：相机能够清晰成像的有效工作距离



# 相机标定



# 相机标定

`cv::calibrateCamera` 是 OpenCV 中用于相机标定的函数，主要用于计算相机内参矩阵和畸变系数。

```
double cv::calibrateCamera(  
    const std::vector<std::vector<cv::Point3f>>& objectPoints,  
    const std::vector<std::vector<cv::Point2f>>& imagePoints,  
    cv::Size imageSize,  
    cv::Mat& cameraMatrix,  
    cv::Mat& distCoeffs,  
    std::vector<cv::Mat>& rvecs,  
    std::vector<cv::Mat>& tvecs,  
    int flags = 0,  
    cv::TermCriteria criteria = cv::TermCriteria(cv::TermCriteria::EPS + cv::TermCriteria::COUNT, 30, DBL_EPSILON)  
);
```

objectPoints: 3D点的集合，通常是世界坐标系中的物体点

imagePoints: 2D点的集合，对应于图像中检测到的物体点。

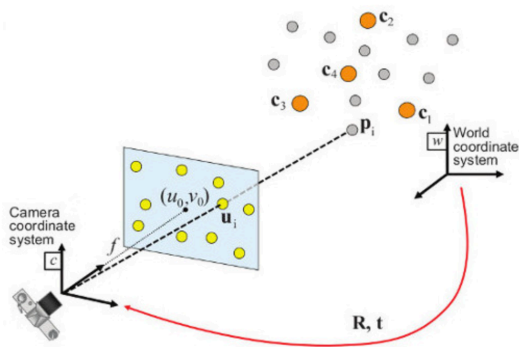
imageSize: 输入图像的尺寸（宽度和高度）

cameraMatrix: 输出参数，表示相机内参矩阵。 distCoeffs: 输出参数，表示相机的畸变系数。

rvecs: 输出参数，旋转向量（相机相对于世界坐标系的旋转）。 tvecs: 输出参数，平移向量。

# pnp解算

pnp 算法通过建立相机像素平面上特征点的2D信息和实际物体对应特征点的3D坐标信息，完成了目标坐标系和相机坐标系之间的**位置**和**姿态**的解算



```
bool cv::solvePnP(  
    const std::vector<cv::Point3f>& objectPoints,  
    const std::vector<cv::Point2f>& imagePoints,  
    const cv::Mat& cameraMatrix,  
    const cv::Mat& distCoeffs,  
    cv::Mat& rvec,  
    cv::Mat& tvec,  
    bool useExtrinsicGuess = false,  
    int flags = cv::SOLVEPNP_ITERATIVE  
);
```

objectPoints: 3D点的集合，表示物体在空间中的位置。

imagePoints: 对应的2D点集合，表示在图像中的位置。

cameraMatrix: 相机内参矩阵，包含焦距和主点信息。

distCoeffs: 相机畸变系数，描述镜头的畸变情况。

rvec: 输出参数，表示相机的旋转向量。

tvec: 输出参数，表示相机的平移向量。

useExtrinsicGuess: 是否使用外部猜测值