

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。

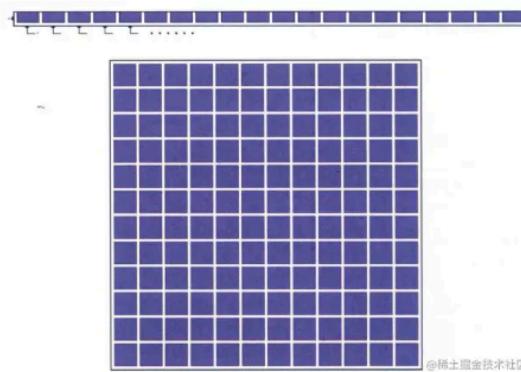
光电二极管

被光照射时能溢出电子的设备

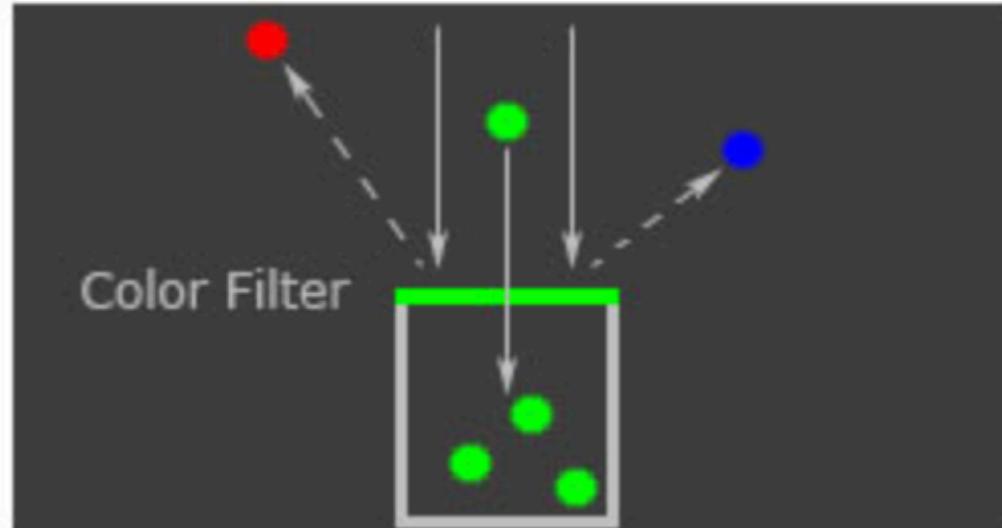
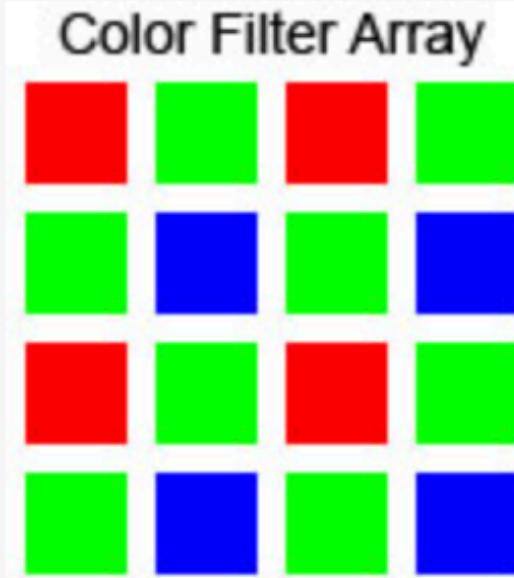
阵列传感器

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。



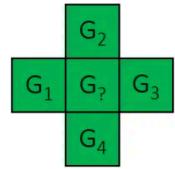
Bayer 传感器阵列



courtesy of cambridgeincolour.com

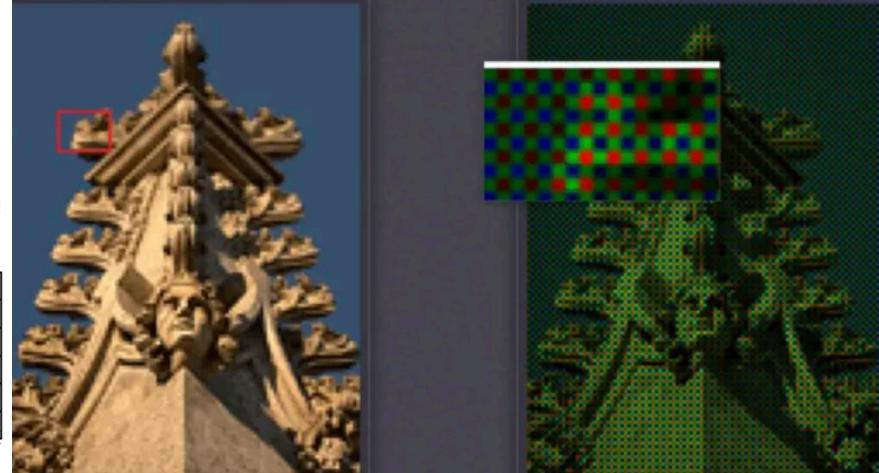
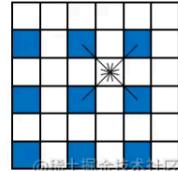
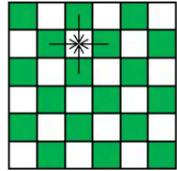
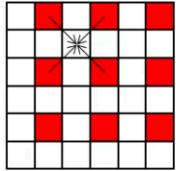
这样，每个传感器能够获取一种颜色的光强。以此生成的图片叫做raw图

RGB 图像的合成: 去马赛克



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

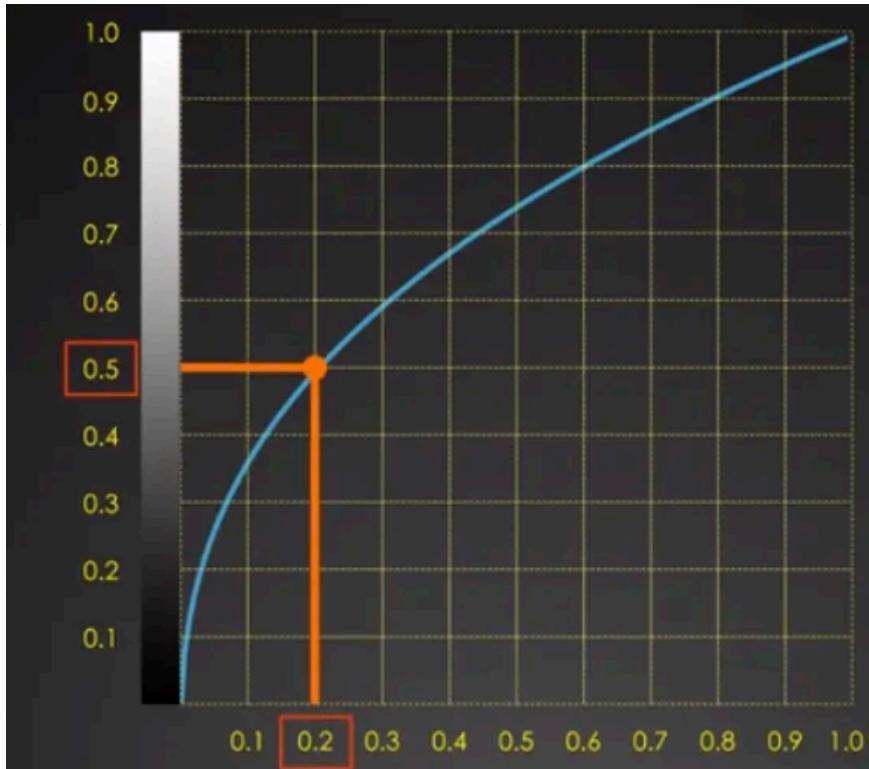
@稀土掘金技术社区



亮度平衡： gamma矫正

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。

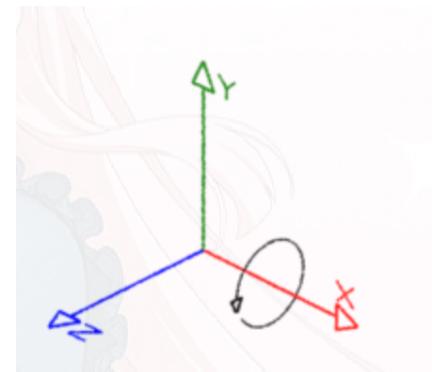
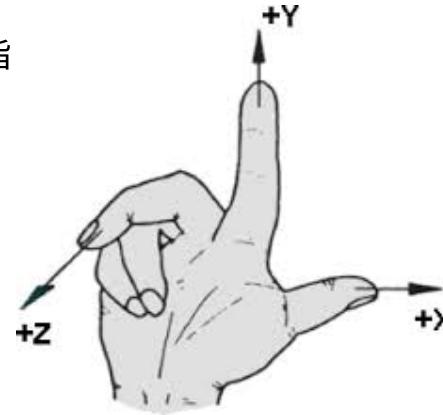
gamma：gamma矫正时，衡量曲线的参数。gamma越大，图像越亮，越关注暗处细节

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 表示参数，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 3×3 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：

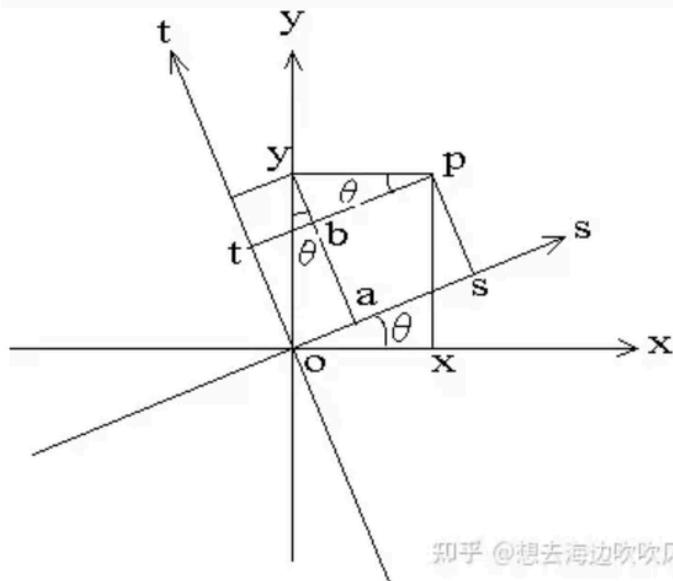
- **正交性:** $R^T R = I$ ，即矩阵的转置等于其逆。
- **行列式为 1:** $|R| = 1$ ，表示旋转不改变空间的体积。

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 θ 度示例：



知乎 @想去海边吹吹风

■ 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 1×3 的列向量，表示两个坐标系原点之间的平移关系。它可以表示为：

一个简单的例子如下，通过 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。

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

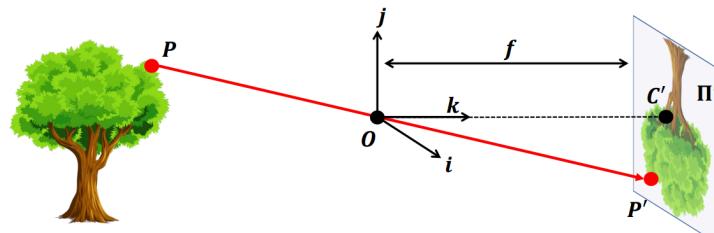
$$\begin{bmatrix} {}^R_x \\ {}^R_y \\ {}^R_z \\ 1 \end{bmatrix} = {}^F_T \begin{bmatrix} {}^F_x \\ {}^F_y \\ {}^F_z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3 \times 3} & {}^R_t_{3 \times 1} \\ O_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} {}^F_x \\ {}^F_y \\ {}^F_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^F_x \\ {}^F_y \\ {}^F_z \\ 1 \end{bmatrix}$$

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。

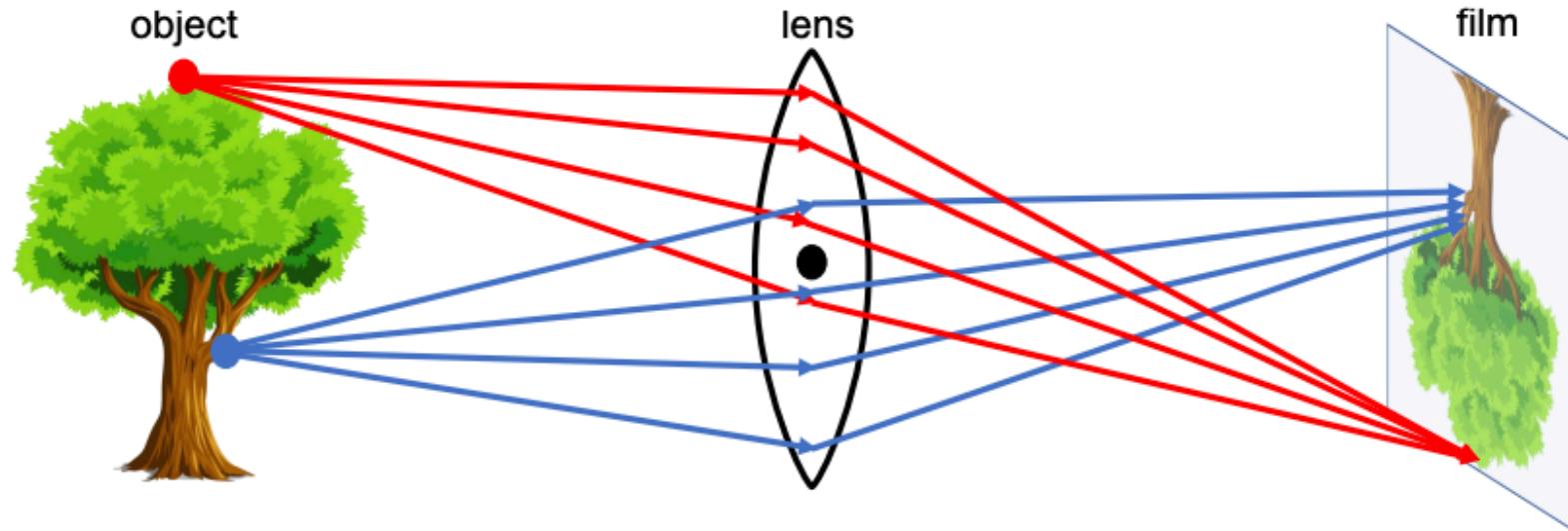
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x X_c / Z_c + c_x \\ f_y Y_c / Z_c + c_y \end{bmatrix}$$



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

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

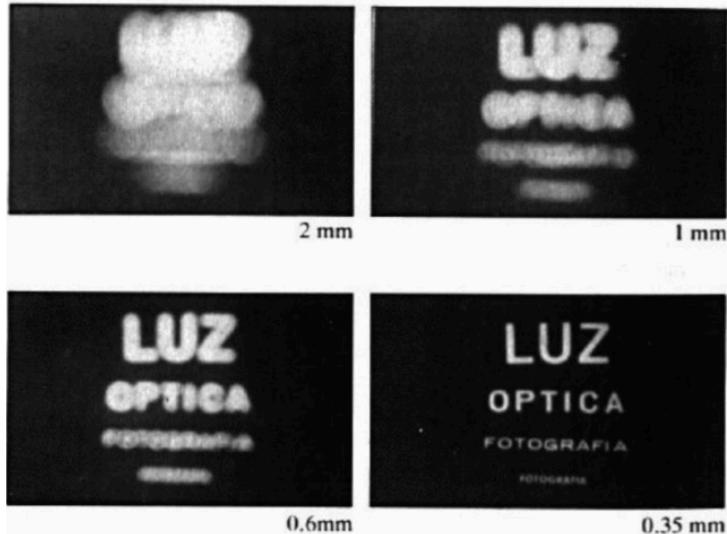
凸透镜成像



- 在小孔成像中, 成像越清晰, 透过小孔的光线越少, 画面越暗

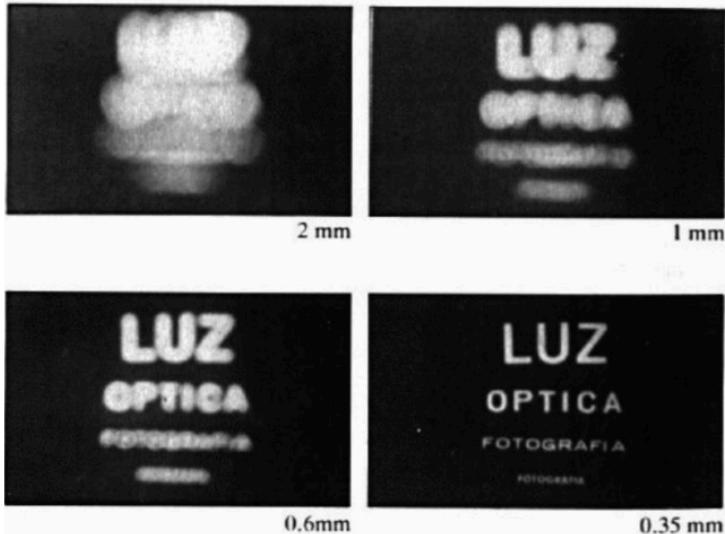
凸透镜成像

- 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)

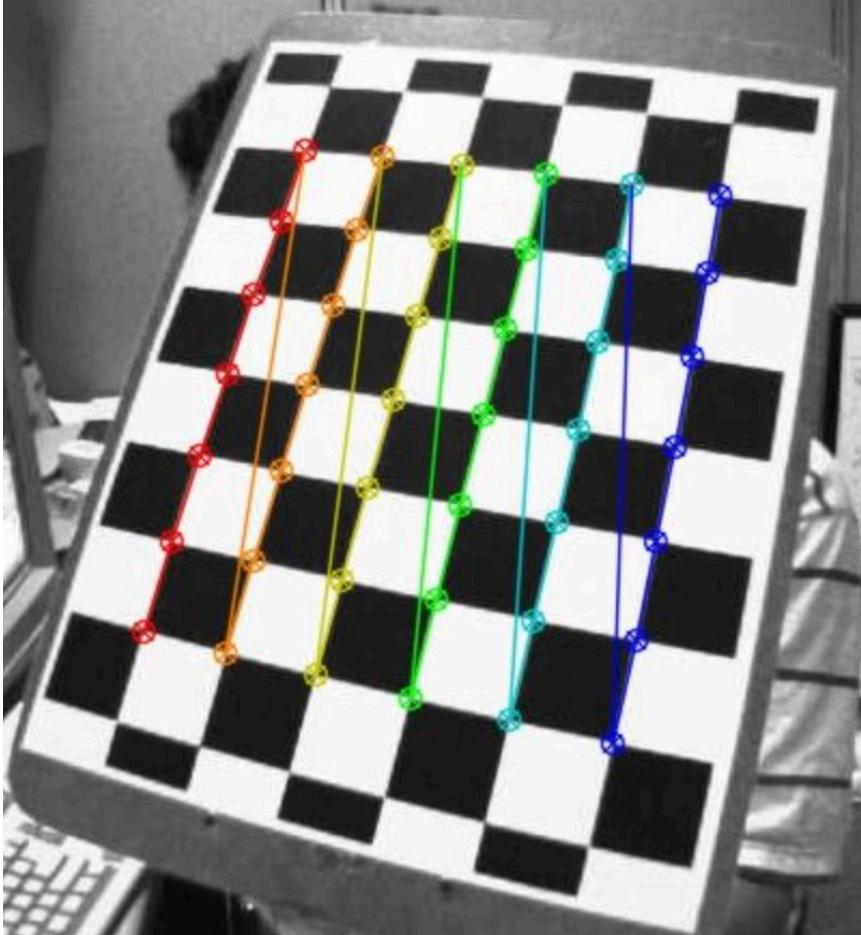


凸透镜成像

- 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。

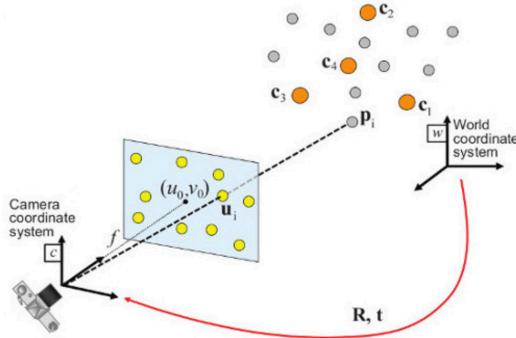
imageSize: 输入图像的尺寸（宽度和高度）

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。

卡尔曼滤波

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{w}_k$$

卡尔曼滤波是一种用于线性动态系统中的递归估计方法，常用于状态估计问题，例如目标跟踪、导航、传感器融合等。它通过结合系统的预测模型和噪声观测数据来估计系统的状态，并随着时间的推移逐步更新这些估计值。

卡尔曼滤波分为两个主要阶段：预测和更新。

基本概念

- 状态变量 (\mathbf{x}_k)：表示系统在时刻 k 的内部状态。
- 状态转移模型 (\mathbf{F}_k)：描述系统状态如何从时刻 $k - 1$ 变化到时刻 k 。

其中， \mathbf{F}_k 是状态转移矩阵，表示从 $k - 1$ 到 k 的状态变化。 \mathbf{w}_k 是过程噪声，假设其均值为零，协方差为 \mathbf{Q}_k 。

- 观测模型 (\mathbf{H}_k)：描述如何从状态估计得到观测值。

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

其中， \mathbf{H}_k 是观测矩阵，由于建立观测量和状态量之间的关系。 \mathbf{v}_k 是测量噪声，假设其均值为零，协方差为 \mathbf{R}_k 。

- 噪声协方差矩阵：

- \mathbf{Q}_k ：过程噪声协方差矩阵。
- \mathbf{R}_k ：测量噪声协方差矩阵。

- 状态协方差 (\mathbf{P}_k)：描述状态的不确定性

卡尔曼滤波

(1) 预测阶段

在预测阶段，卡尔曼滤波器根据之前的状态估计来预测当前时刻 k 的状态和协方差。

状态预测公式：

$$\hat{\mathbf{x}}_k^- = \mathbf{F}_k \hat{\mathbf{x}}_{k-1}$$

- $\hat{\mathbf{x}}_k^-$ 是预测的状态估计值（先验估计）。
- \mathbf{F}_k 是状态转移矩阵，表示从 $k - 1$ 到 k 的状态变化。

协方差预测公式：

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

- \mathbf{P}_k^- 是预测的状态协方差矩阵（先验协方差）。
- \mathbf{P}_{k-1} 是前一时刻状态估计的协方差矩阵。
- \mathbf{Q}_k 是过程噪声的协方差。

(2) 更新阶段

在更新阶段，卡尔曼滤波器利用观测数据来修正预测的状态和协方差，从而得到更准确的估计值。

卡尔曼增益公式

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1}$$

- \mathbf{K}_k 是卡尔曼增益矩阵，确定如何在观测值和预测状态之间加权。

卡尔曼滤波

(2)更新阶段

状态更新公式

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$$

- $\hat{\mathbf{x}}_k$ 是更新后的状态估计值（后验估计）。
- $\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-$ 是预测误差，即测量值和预测观测的差值，也叫做残差。

协方差更新公式

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$$

- \mathbf{P}_k 是更新后的状态协方差矩阵（后验协方差）。
- \mathbf{I} 是单位矩阵。
- \mathbf{H} 是观测矩阵，能够将状态变量映射到观测空间。

卡尔曼滤波流程总结

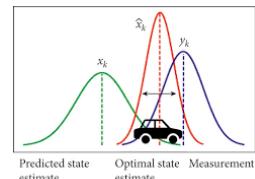
1. 预测阶段：

- 预测状态： $\hat{\mathbf{x}}_k^- = \mathbf{F}_k \hat{\mathbf{x}}_{k-1}$
- 预测协方差： $\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k$

2. 更新阶段：

- 计算卡尔曼增益：
- $\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
- 更新状态估计： $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$
- 更新协方差估计： $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$

这个递归过程使得卡尔曼滤波能够在噪声环境下对系统的状态进行实时跟踪。



二维运动的卡尔曼滤波实例

参数定义

k : 当前时间 , $k - 1$: 上一阶段时间

x, y : x 和 y 方向的位置

\dot{x}, \dot{y} : x 和 y 方向的速度

可以表示为

$$\mathbf{x}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1}$$

这里 \mathbf{x}_k 表示当前状态量 \mathbf{x}_{k-1} 上一时刻的状态量。

由此得到状态转移矩阵 F 。

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

二维运动的卡尔曼滤波实例

观测分析 (H)

由前面分析的观测方程如下：

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k$$

由此得到的观测矩阵为 H

在推导观测模型时，我们假设我们只测量位置而不测量速度。

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

所以测量量为：

$$\mathbf{z}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

因此，为了建立观测量和状态量之间的联系，我们可以将测量模型写成；

二维运动的卡尔曼滤波实例

过程噪声协方差矩阵 Q

假设测量位置 x 和 y 都是独立的，因此我们可以忽略它们之间的任何相互作用，使协方差 x 和 y 为 0。 Q 可以表达为：

$$\mathbf{Q} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \\ \sigma_x^2 & 0 & \sigma_x \sigma_{\dot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_y \sigma_{\dot{y}} \\ \sigma_{\dot{x}} \sigma_x & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & \sigma_{\dot{y}} \sigma_y & 0 & \sigma_{\dot{y}}^2 \end{bmatrix}$$

可以重写为与时间相关

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \sigma_a^2$$

测量噪声协方差矩阵 R

同，测量噪声协方差 R 可表示为：

$$\mathbf{R} = \begin{bmatrix} x & y \\ \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \frac{\Delta t^4}{4} & 0 \\ 0 & \frac{\Delta t^4}{4} \end{bmatrix}$$

拓展卡尔曼滤波

卡尔曼滤波仅适用于线性系统，若是变量间关系设计到非线性，显然无法使用简单的矩阵运算描述变量间关系。

此时需要采用**拓展卡尔曼**进行建模，同时使用**雅可比矩阵**来描述变量间的关系，但是本质思想不变。

EKF步骤如下：

(1)状态预测：利用非线性系统的状态方程进行预测。

(2)线性化：通过在当前状态估计处对非线性方程进行一阶泰勒展开，得到线性化模型。

(3)更新：根据观测值更新状态估计和协方差矩阵。

主要公式

1. 预测步骤

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1})$$

$$P_k^- = F_k P_{k-1} F_k^T + Q_k$$

其中 f 为状态转移函数， F_k 是状态转移函数的雅可比矩阵， P_k^- 是预测协方差矩阵， Q_k 是过程噪声协方差矩阵。

2. 更新步骤：

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-))$$

$$P_k = (I - K_k H_k) P_k^-$$

其中 H_k 是观测方程的雅可比矩阵， R 是测量噪声协方差矩阵， z_k 是测量值。

拓展卡尔曼滤波 状态预测方程的线性化

对于状态转移方程 $f(x)$:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1})$$

它被线性化为:

$$F_k = \frac{\partial f}{\partial x} \Big|_{\hat{x}_{k-1}}$$

其中 F_k 是状态转移函数的雅可比矩阵，表示对当前状态的线性近似。

观测方程的线性化:

观测方程 $h(x)$:

$$z_k = h(x_k) + v_k$$

被线性化为:

$$H_k = \frac{\partial h}{\partial x} \Big|_{\hat{x}_k^-}$$

其中 H_k 是观测方程的雅可比矩阵，表示对观测模型的线性近似。