3D感知技术与实践

3D传感器原理

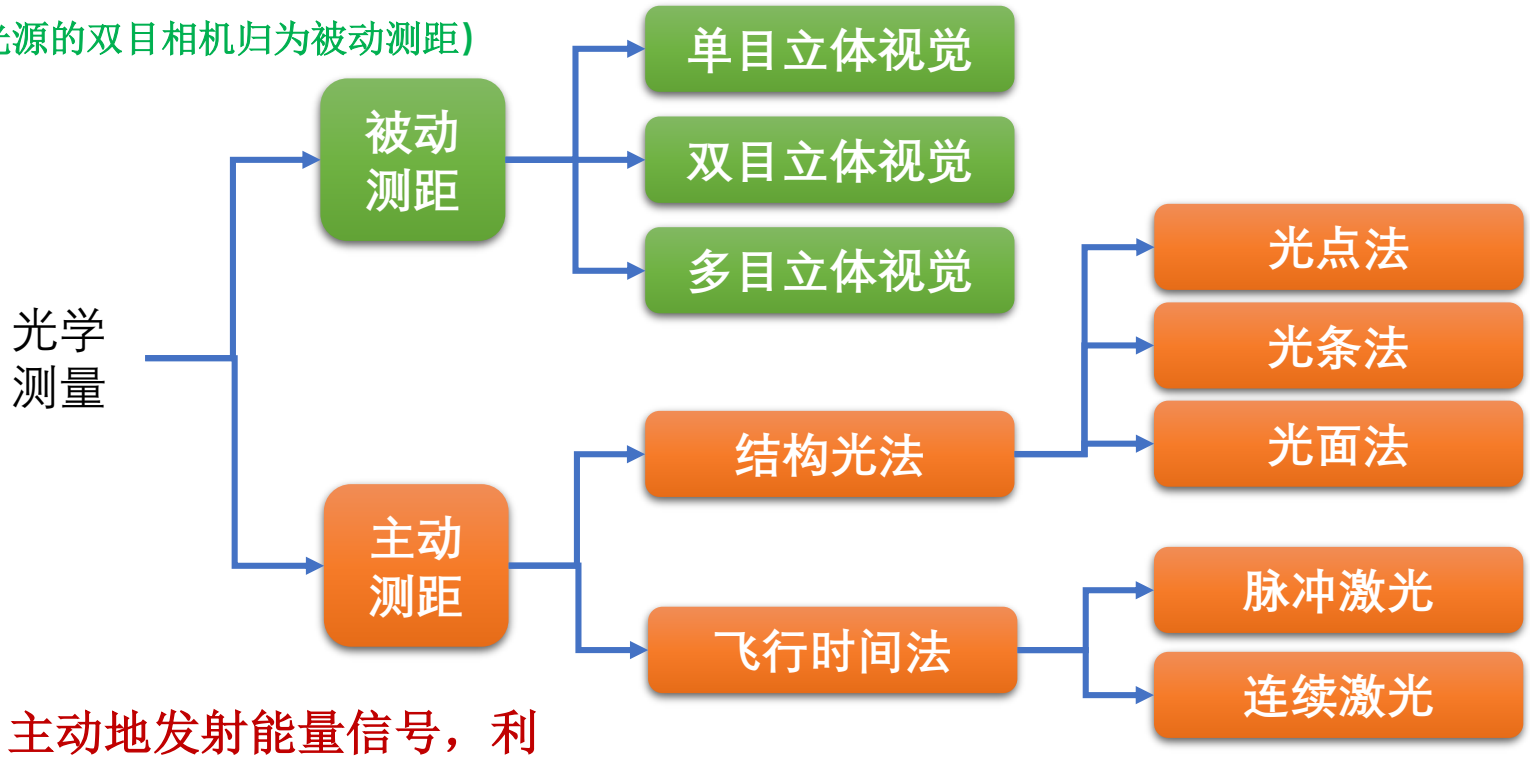
内容概要

- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

3D光学测量方法概述

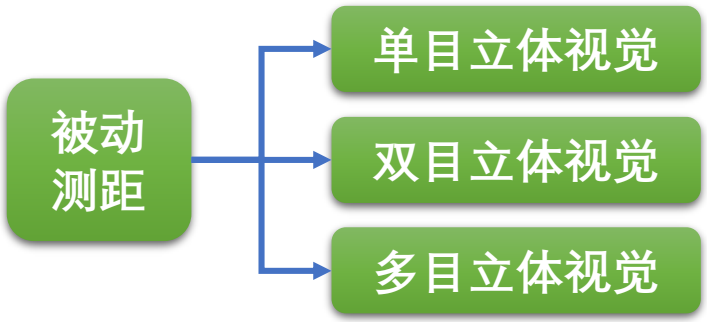
利用自然光在物体表面反射形成的2D图片进行3D信息的重建

（带有补光光源的双目相机归为被动测距）



人为地、主动地发射能量信号，利用物体表面反射特性及信号的传播特性实现对物体的3D信息的测量

3D光学测量方法概述——被动测距方法



单目测距

- 聚焦法——利用相机焦距可变的特性，变化焦距使被测量物体处于聚焦位置，然后利用成像公式计算出被测量物体距离相机的距离
- 离焦法——不需要被测量物体处于焦距上，而是根据标定好的离焦模型计算物体与相机之间的距离

测量方法简单，成本低廉，但存在精度不高，测量范围小等问题

双目立体视觉

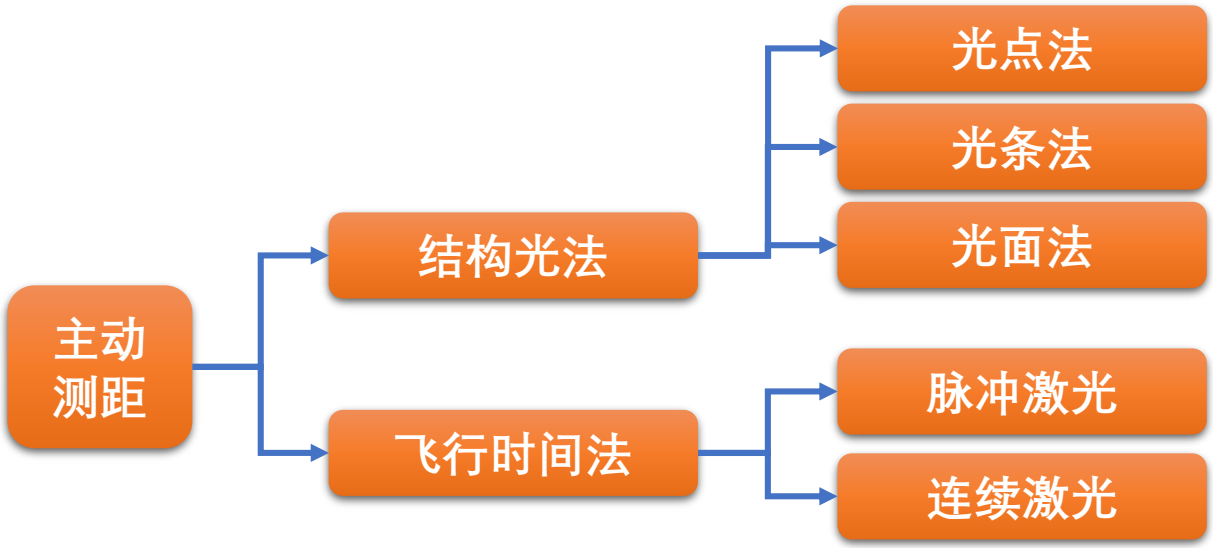
在两个视点观察同一个场景，然后利用匹配算法计算两个图像像素的位置偏差进行3D测距

与人眼结构相类似，测距精准

多目立体视觉

双目的扩展——多个视点观察同一个场景，进行多次匹配计算实现距离测量

3D光学测量方法概述——主动测距方法



结构光法

投射特定纹理光线到被测物体，通过相机拍摄物体表面的投影纹理，经过计算得到距离。

飞行时间（ToF）法

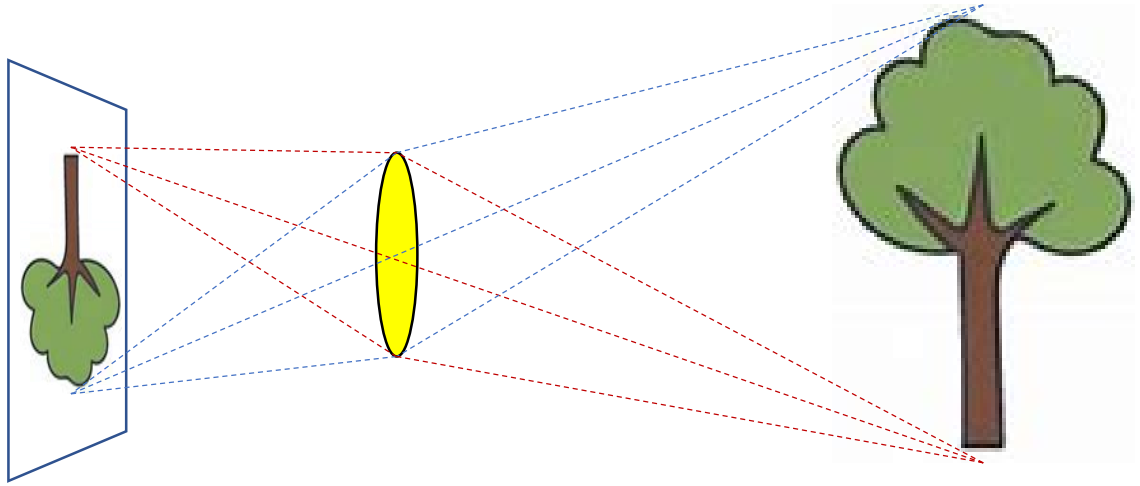
将调制光信号发射到物体表面，通过测量接收反射信号与发射信号的时间差计算物体表面到相机的距离。基于以上两种原理的相机目前已经成为消费级产品

内容概要

- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

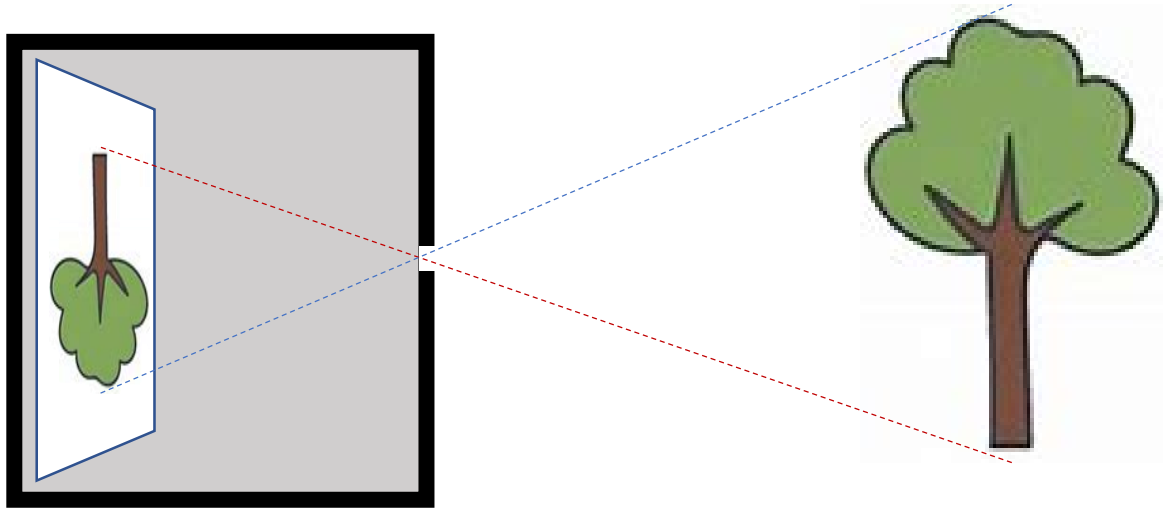
3D视觉原理——针孔相机模型

- 相机结构一般用透镜模型解释
- 穿过镜头中心的光线没有改变方向，镜头中心称为“光心”



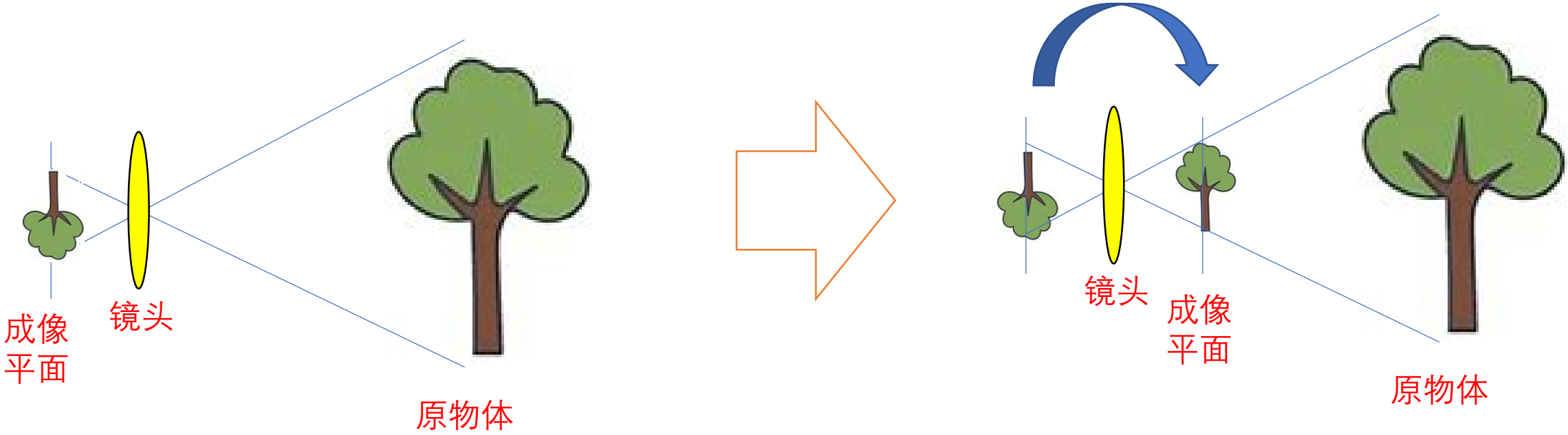
3D视觉原理——针孔相机模型

- 透镜模型可以简化为针孔模型——将透镜替换成小孔
- 光心对应小孔中心



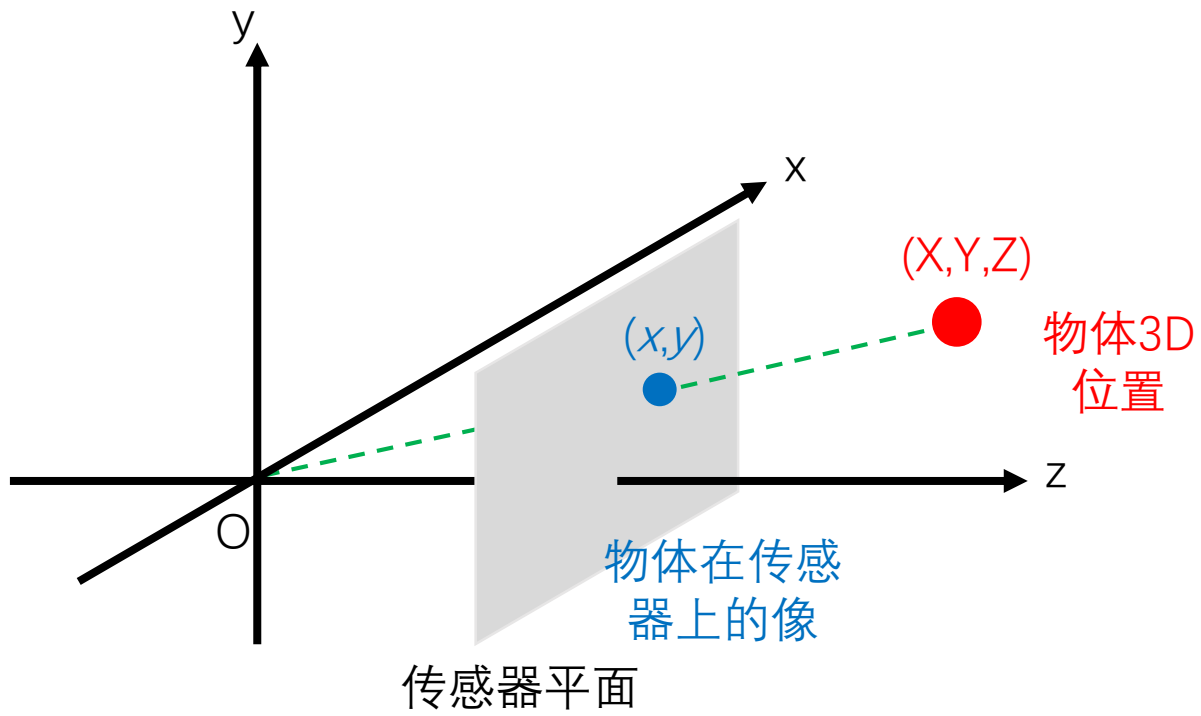
3D视觉原理——针孔相机模型

一般为了分析简单,将成像平面画在对称位置,这样图像不再颠倒



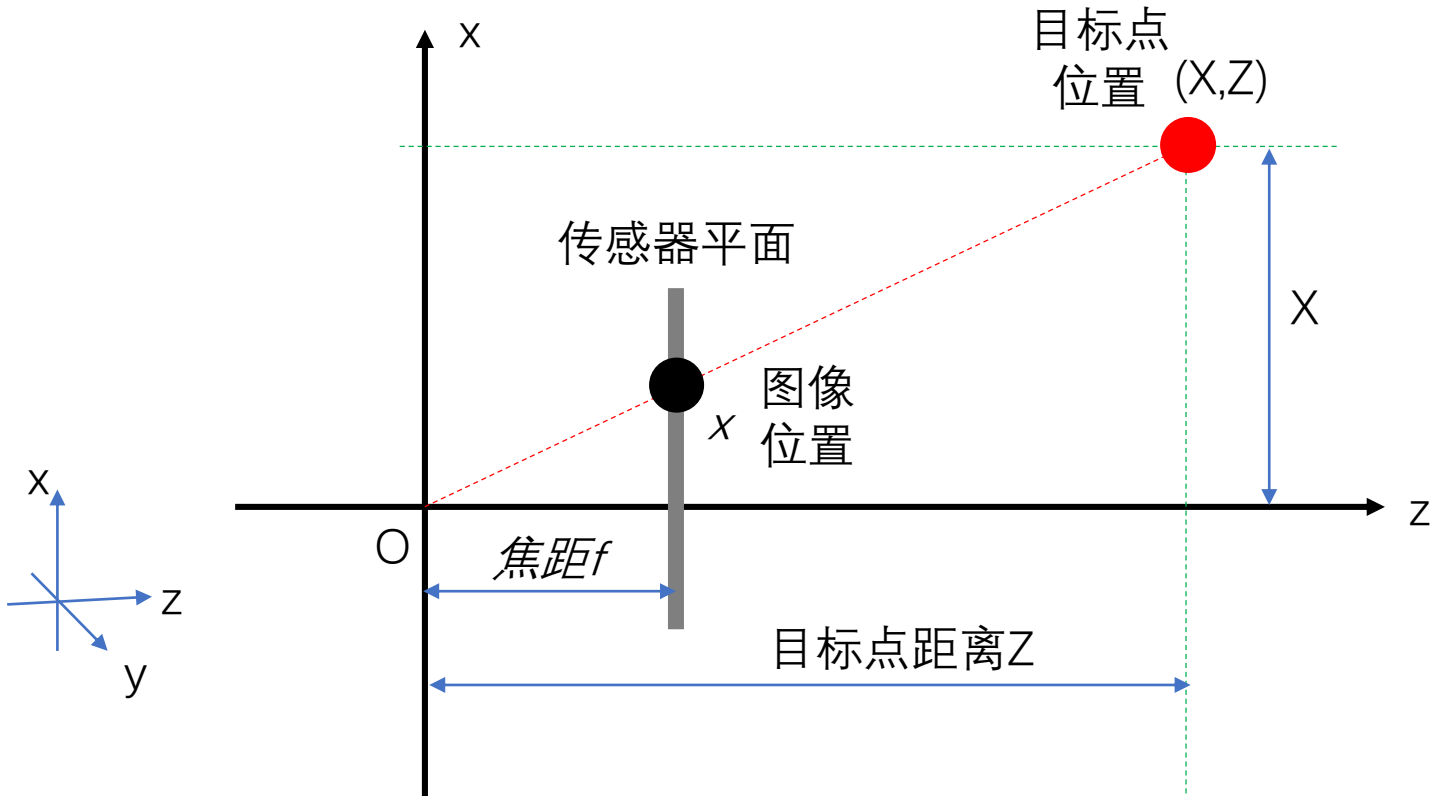
3D视觉原理——针孔相机模型

- 空间的3D点和图像传感器上的位置关系通过上面的图给出
- 传感器平面上的图像点看成是从空间点（X，Y，Z）到原点的连线和传感器平面的交点



3D视觉原理——针孔相机模型

- 利用相似三角形能够看出图像传感器平面上的像素位置和3D空间点的位置关系



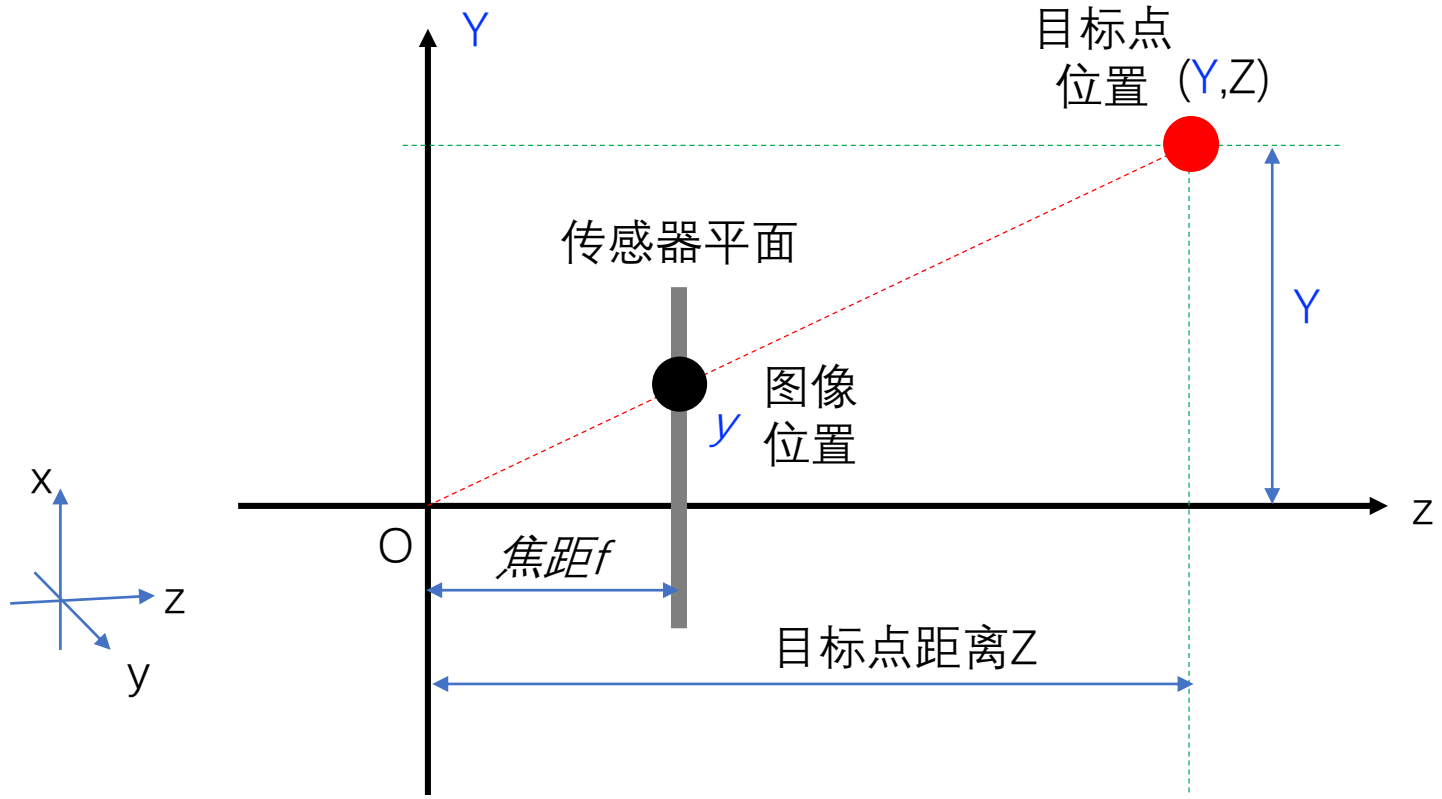
相似三角形 $\frac{X}{x}=\frac{Z}{f}$

$$X = x\frac{Z}{f}$$

如果有Z信息的话,可以从图像上的坐标x得到物体的3D坐标X

3D视觉原理——针孔相机模型

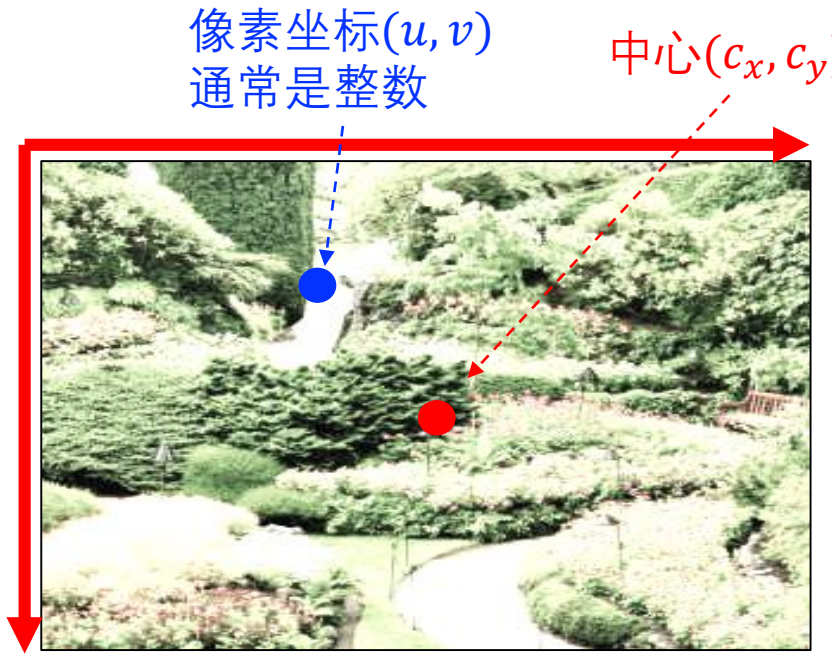
- 之前的公式作用于X坐标，它同样可以用于Y坐标



X,Y坐标都以相同方式获得

$$\begin{cases} X = x\dfrac{Z}{f} \\ Y = y\dfrac{Z}{f} \end{cases}$$

3D视觉原理——针孔相机模型



实际图像和前面的模型还有些差别

- 图像的坐标系原点在左上角，镜头中心对应的像素坐标(c_x,c_y)不是(0,0)
- 在计算像素坐标(u,v)和3D物理坐标关系前，先平移$(u-c_x,v-c_y)$
- 图像像素坐标一般是整数，如何和3D物理坐标联系起来？
- 需要根据传感器物理尺寸将像素坐标乘以比例系数变成物理坐标

需要乘以比例系数s_x和s_y：

$$\begin{cases} x = s_x(u-c_x) \\ y = s_y(v-c_y) \end{cases}$$

代入上式

$$\begin{cases} X = x\frac{Z}{f} \\ Y = y\frac{Z}{f} \end{cases}$$

$$\begin{cases} X = (u-c_x)\frac{Z}{f/s_x} \\ Y = (v-c_y)\frac{Z}{f/s_y} \end{cases}$$

其中比例系数(s_x,s_y)和f合并得到

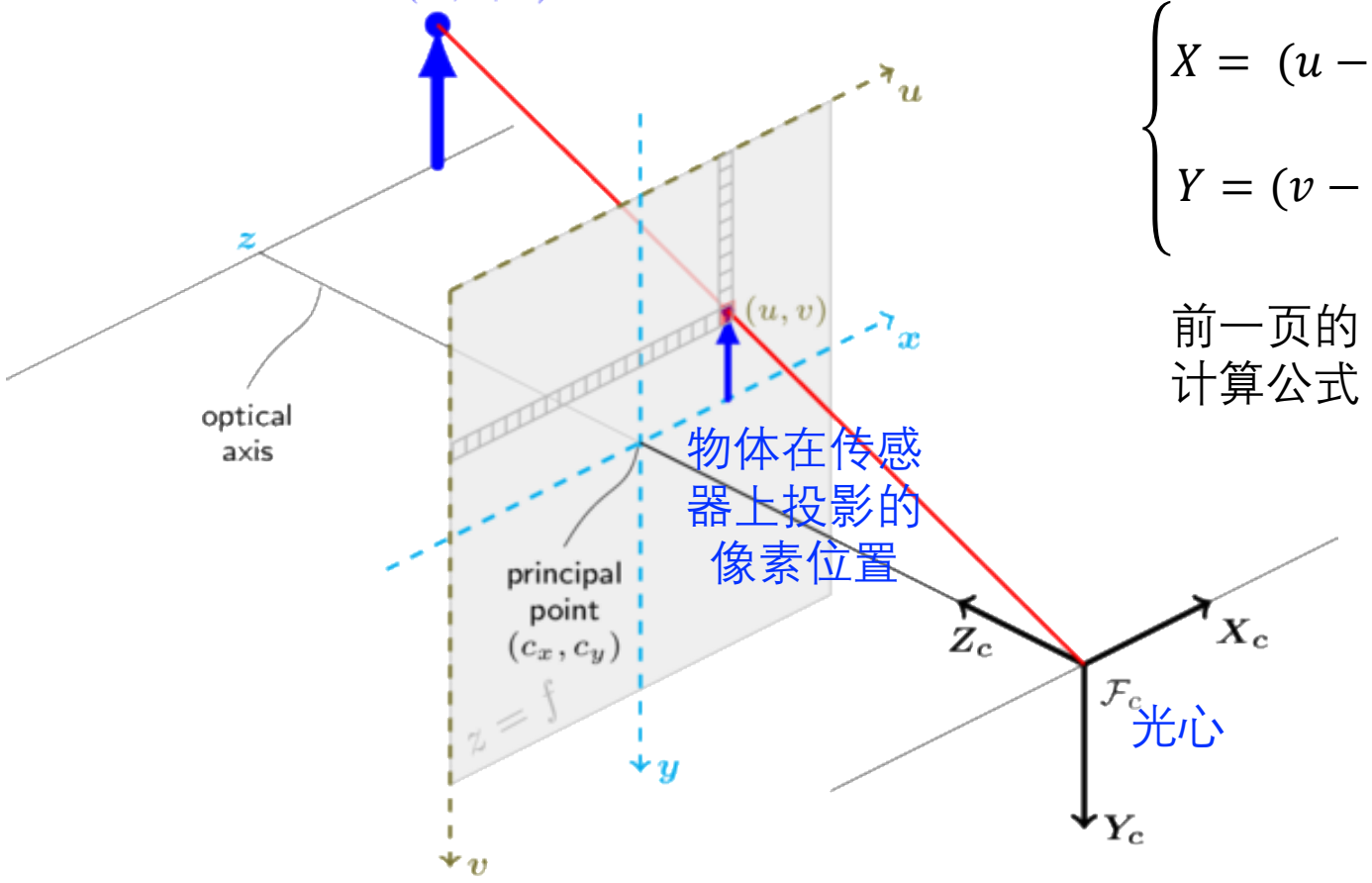
$$\begin{cases} X = (u-c_x)\frac{Z}{f_x} \\ Y = (v-c_y)\frac{Z}{f_y} \end{cases}$$

其中$\{f_x,f_y,c_x,c_y\}$被称为“相机内参”

这里$f_x = f/s_x$、 $f_y = f/s_y$

3D视觉原理——针孔相机模型

- 下图完整画出图像像素位置和空间点的坐标之间关系
- 这里用符号(u,v)而不是(x,y)描述图像坐标是为了强调(u,v)是整数像素坐标



物体在传感器上投影的像素位置

光心

$$\begin{cases} X = (u-c_x)\dfrac{Z}{f_x} \\ Y = (v-c_y)\dfrac{Z}{f_y} \end{cases}$$

前一页的计算公式

下面以矩阵形式给出相机图像和3D坐标之间的关系

$$Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

像素坐标(齐次坐标)　内参矩阵　3D物理坐标

3D视觉原理——针孔相机模型

具体计算例子：

拍摄深度图照片上，物体上某个角点的像素位置是（100，200），该像素点对应的物体距离：Z=1.2M

相机参数为$f_x = 310, f_y = 250, c_x = 160, c_y = 120$，求改像素点对应物体的空间3D坐标

$$\begin{cases} X = (u - c_x)\dfrac{Z}{f_x} = (100 - 160)\dfrac{1.2}{310} = -0.232 \\ Y = (v - c_y)\dfrac{Z}{f_y} = (200 - 120)\dfrac{1.2}{250} = 0.384 \end{cases}$$

于是得到该物体角点的空间3D坐标为：$(-0.232, 0.384, 1.2)$

代码例子——将3D相机的数据转换成点云坐标

$$\begin{cases} X = Z\dfrac{u - c_x}{f_x} \\ Y = Z\dfrac{v - c_y}{f_y} \end{cases}$$

红色部分是和Z无关的，可以预先计算出来

```
class dep_to_pcl_c
{
public:
    float *tab_u;
    float *tab_v;

    dep_to_pcl_c()
    {
        tab_u=new float[IMG_SZ];
        tab_v=new float[IMG_SZ];

        for (long v=0;v<IMG_HGT;v++)
        {
            for (long u=0;u<IMG_WID;u++)
            {
                long n=u+v*IMG_WID;
                tab_u[n]=((float)(u-CX))/((float)FX);
                tab_v[n]=((float)(v-CY))/((float)FY);
            }
        }
    }
```

```
    void operator()(float *img_dep, float *img_pcl)
    {
        float *p =img_pcl;
        float *q =img_dep;
        float *qe=img_dep+IMG_SZ;
        long n=0;
        while(q<qe)
        {
            float z=*q;
            *p=z*tab_u[n]; p++;
            *p=z*tab_v[n]; p++;
            *p=z;          p++;
            q++;
            n++;
        }
    }

    ~dep_to_pcl_c()
    {
        delete []tab_u;
        delete []tab_v;
    }
};
```

计算X
计算Y
Z（不用计算）

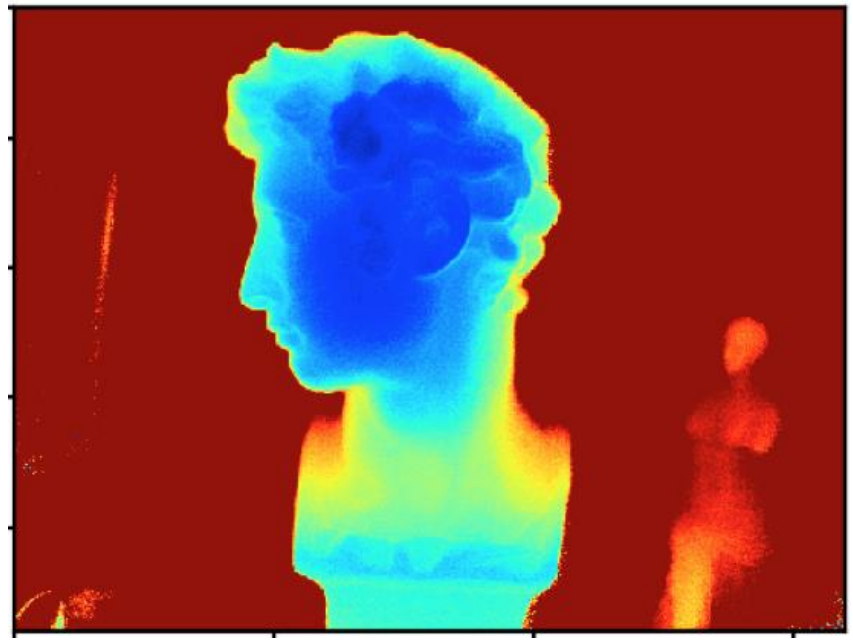
作业描述——深度图转换成点云

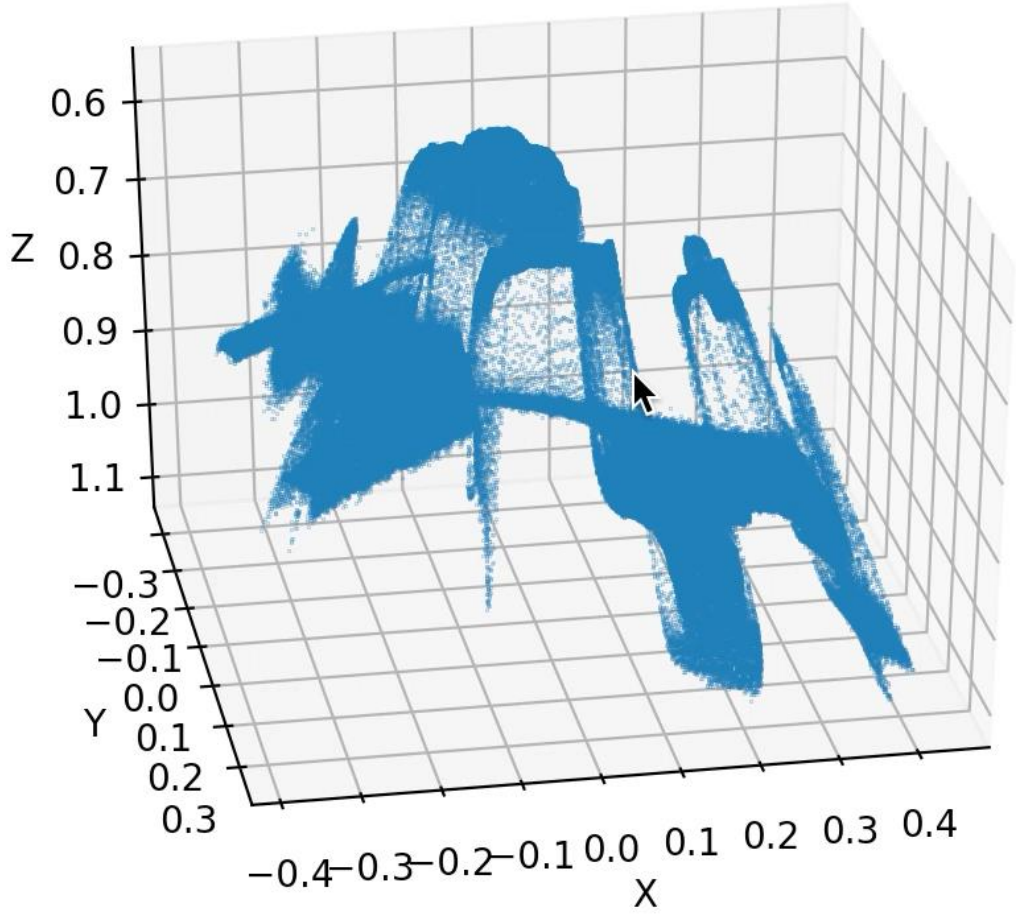
- 从一个深度图生成点云，图片数据是img_dep_640x480.csv，对应图像矩阵img
- 图像矩阵img的第i行j列内存放的是该像素对应的物体距离值Z，
- 需要你根据之前的公式将计算该深度图数据中每个像素对应的空间坐标（X，Y，Z）
- 相机参数是：
  - CAM_FX,CAM_FY　　= 795.209,793.957　# 相机的fx/fy参数
  - CAM_CX,CAM_CY　　= 332.031,231.308　# 相机的cx/cy参数
- 输出数据存放在pc.csv文件内，每行3个浮点数，对应一个像素的XYZ值
- 为帮助你编程，提供一个python的代码框架：partial_code_depth_to_pc.py，你可以通过修改这个代码实现作业要求
- 作业提交的是生成的数据文件pc.csv

- 另外注意：pc.csv文件是文本文件，你可以用任意的文本编辑器打开参看内容

作业描述——深度图转换成点云

- 转换结果参考





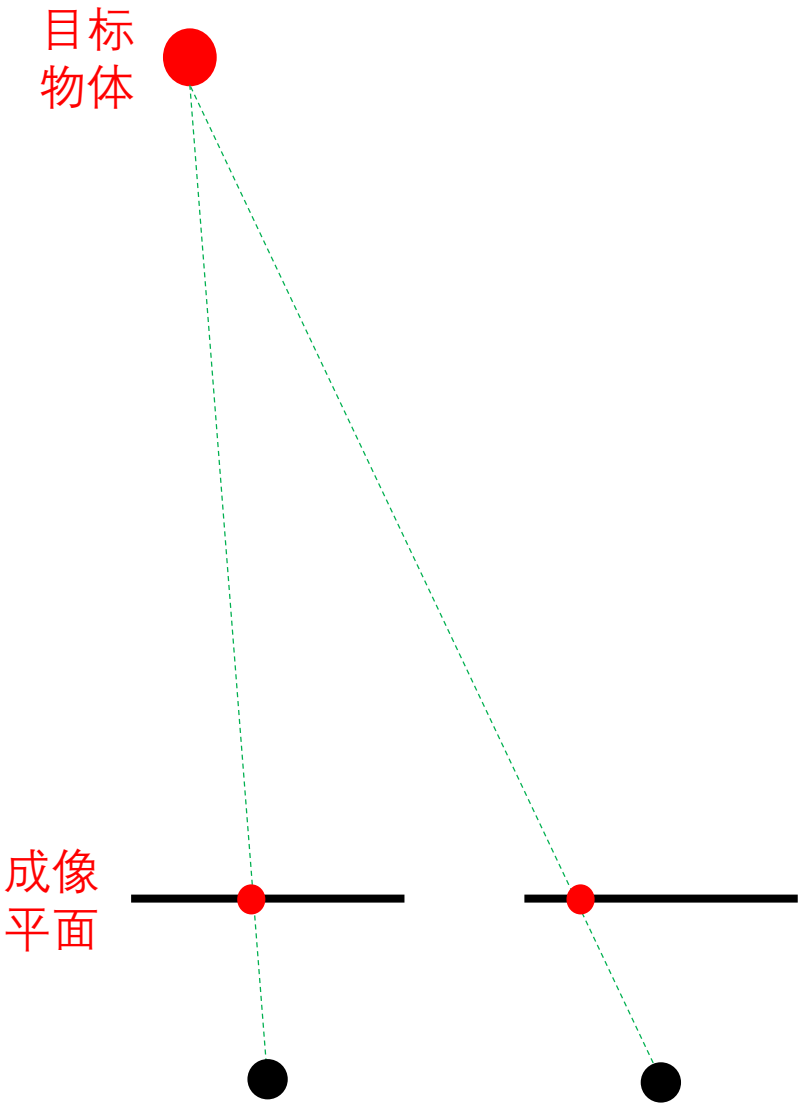
内容概要

- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

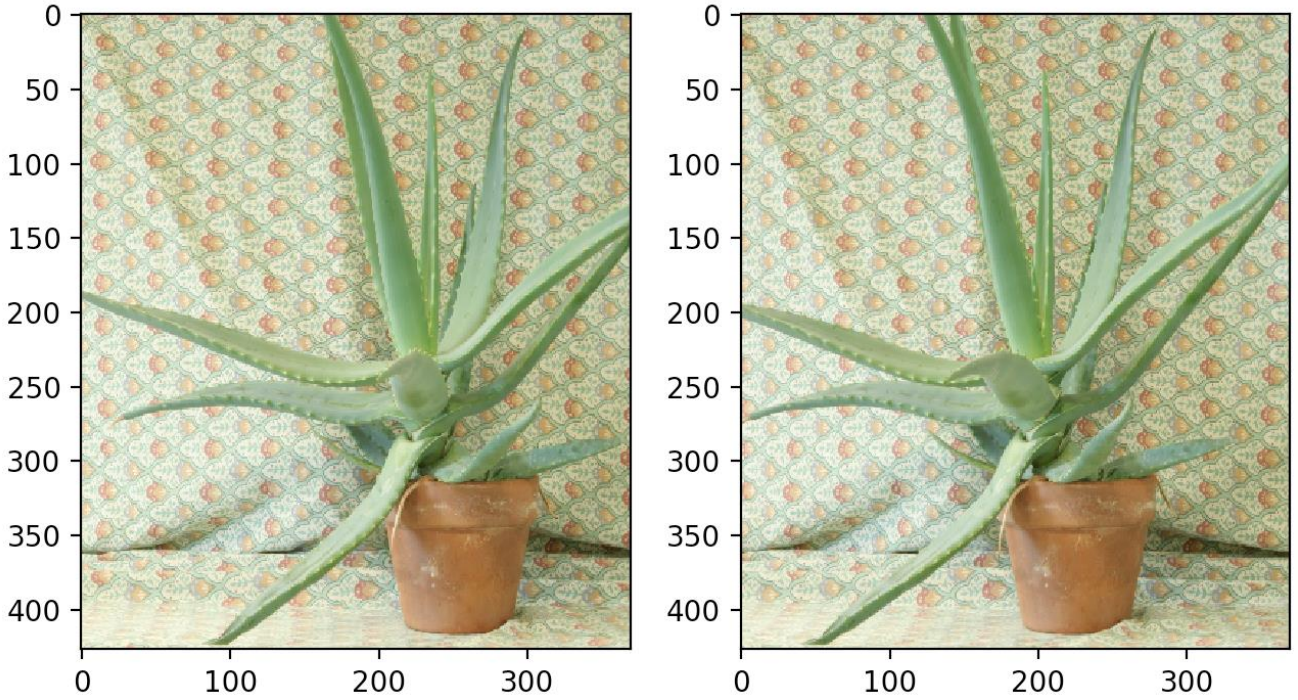
双目3D视觉原理

- 前面可以看到,如果有每个像素点对应的物体的距离信息Z,则可以通过针孔相机模型,计算对应物体的三维坐标(X,Y,Z)
- 如果只有RGB图,如何得到距离信息?

双目3D视觉原理——从视差到3D坐标

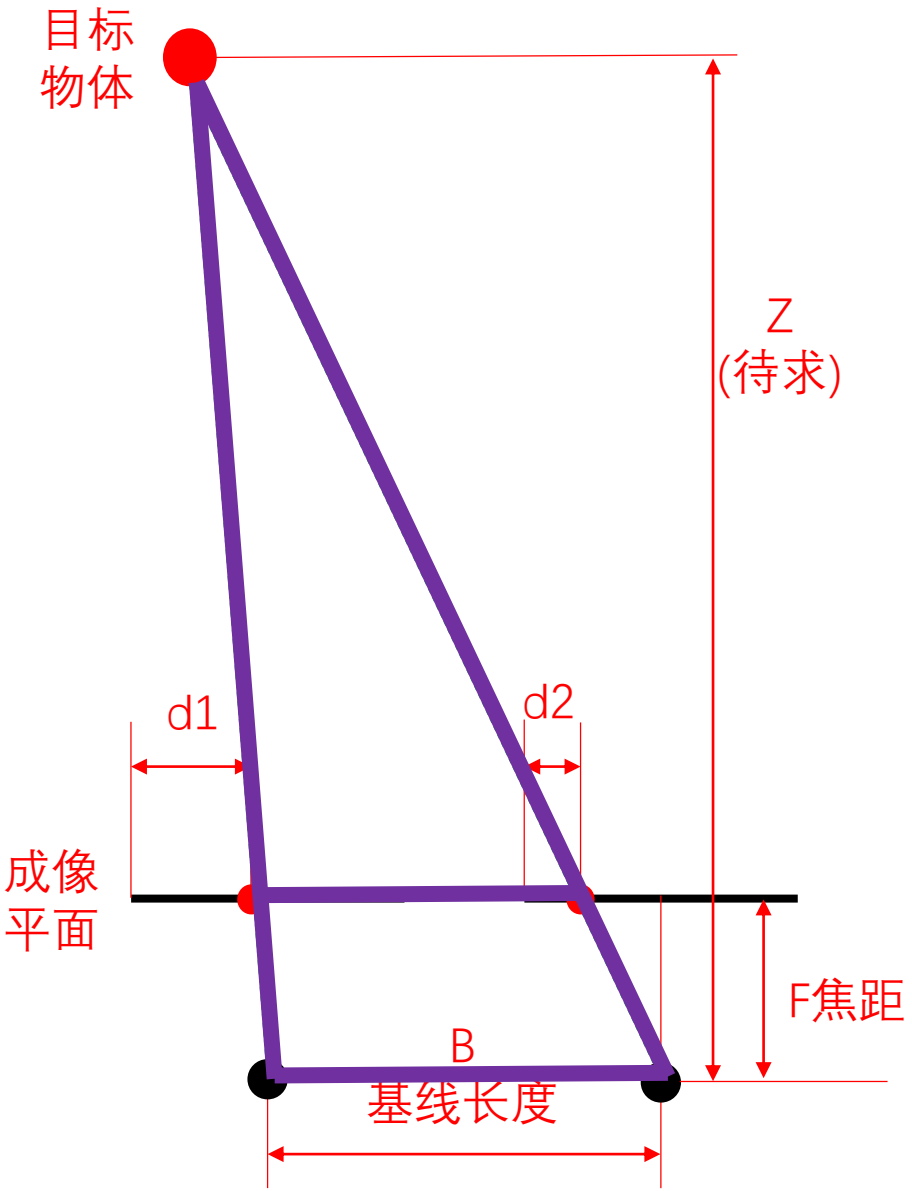


- 这里只讨论最简单的情况,两个相机内参相同, 两个成像平面在相同平面上,
- 两个相机的图像中,对应相同物体的位置只有左右偏移
- 例子所示(画图)

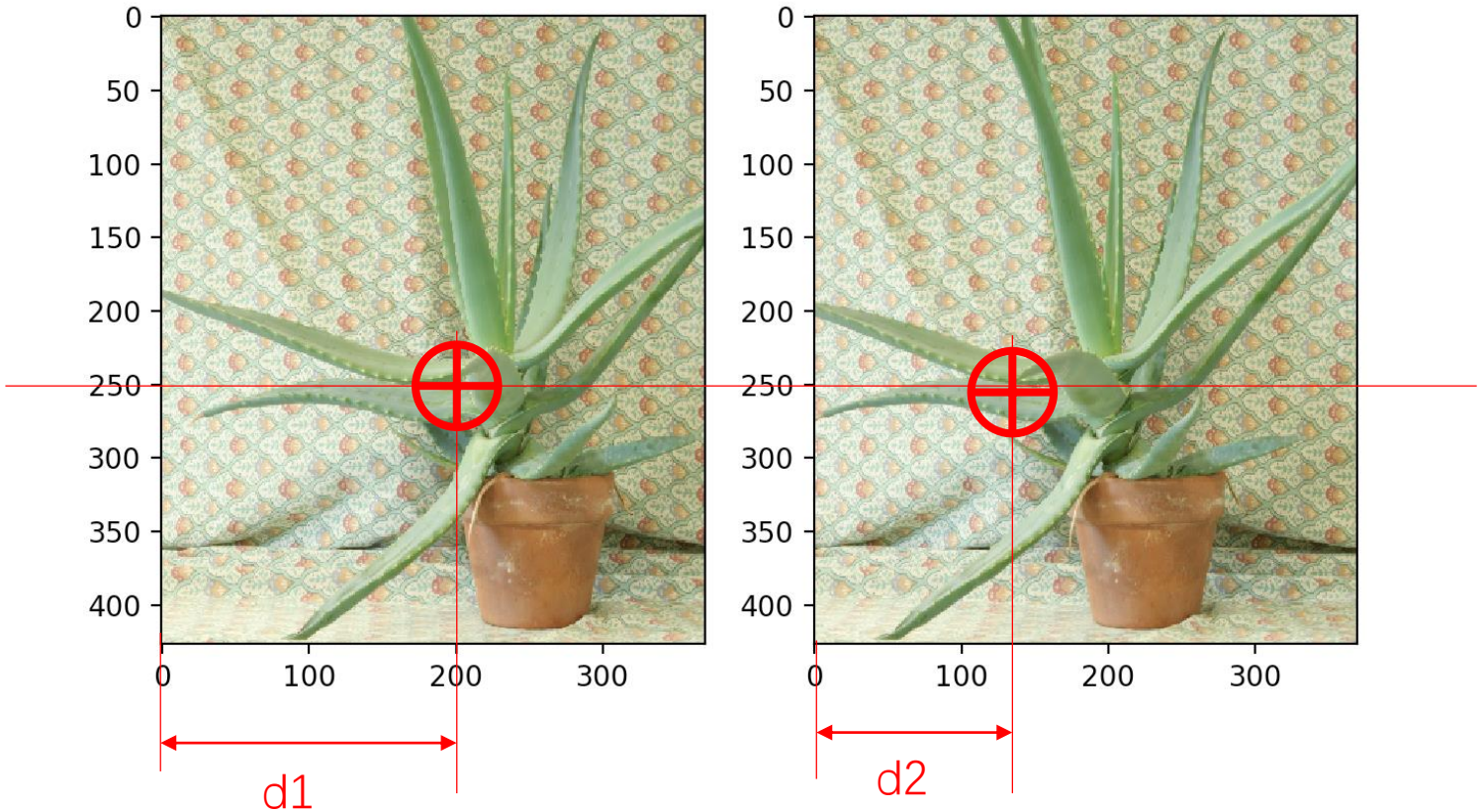


- 同样的物体，在左图上位置偏右，右图上位置偏左

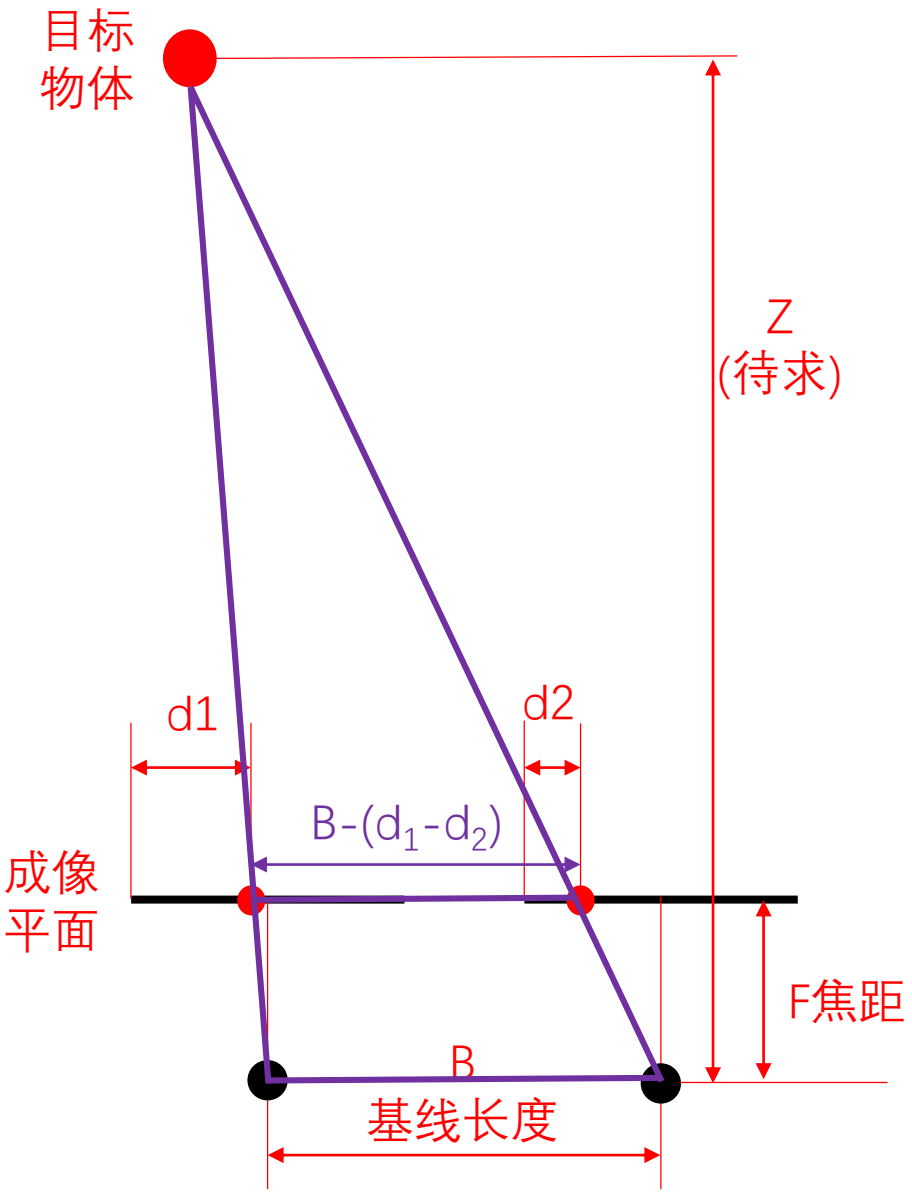
双目3D视觉原理——从视差到3D坐标



- d1和d2是同一物体在两图的位置
- 通过视差可以得到距离——使用相似三角形计算



双目3D视觉原理——从视差到3D坐标



相似三角形性质

$$\frac{Z-f}{B-(d_1-d_2)}=\frac{Z}{B}$$

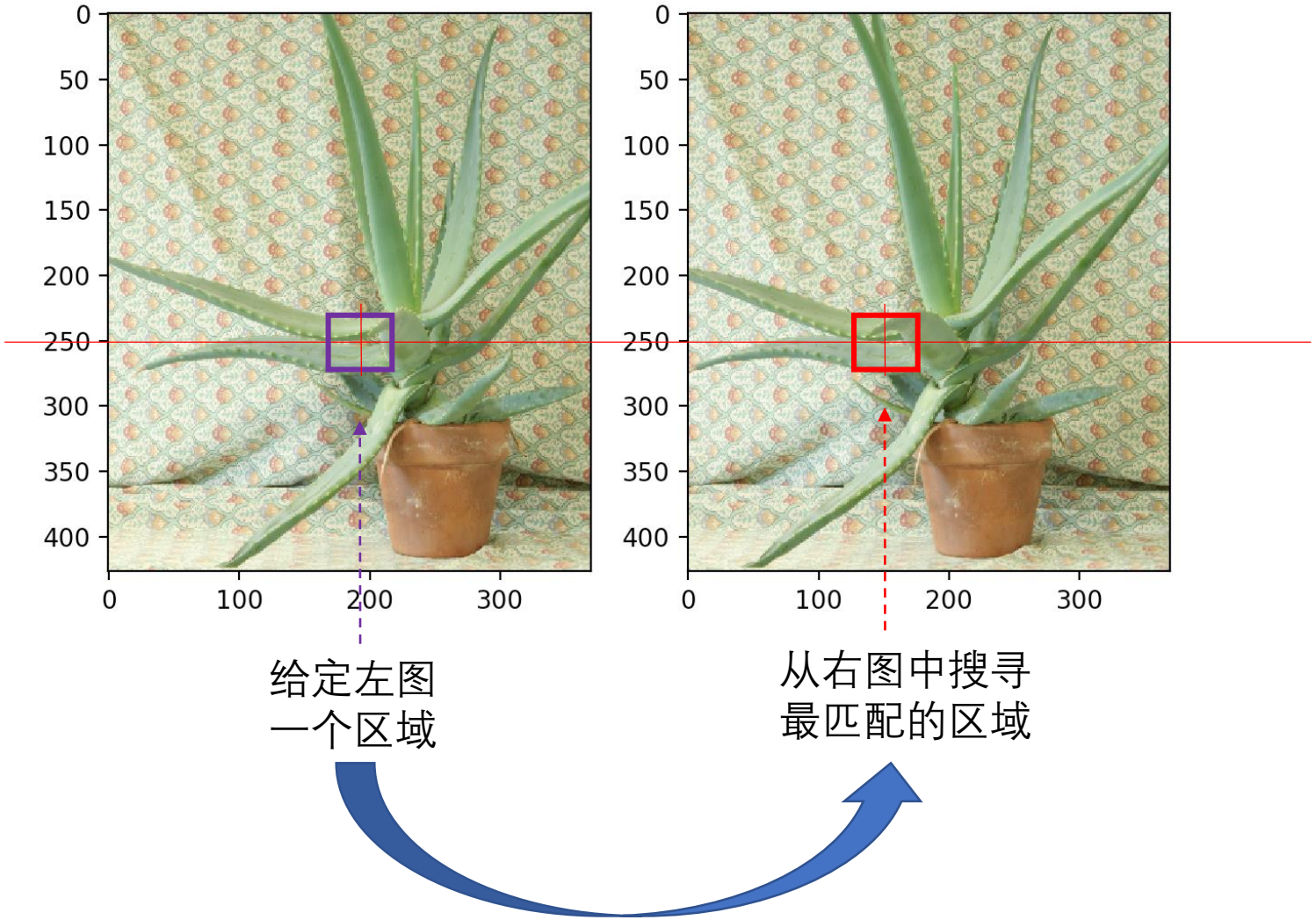
$$Z=\frac{Bf}{\Delta d}$$

通过视差计算
距离Z的公式

$$\Delta d=d_1-d_2$$

视差

双目3D视觉原理——视差匹配查找



在有图中,相同高度,左右滑动搜索框,每次滑动,计算匹配度,直到找出最有匹配结果

通过视差计算距离Z的公式

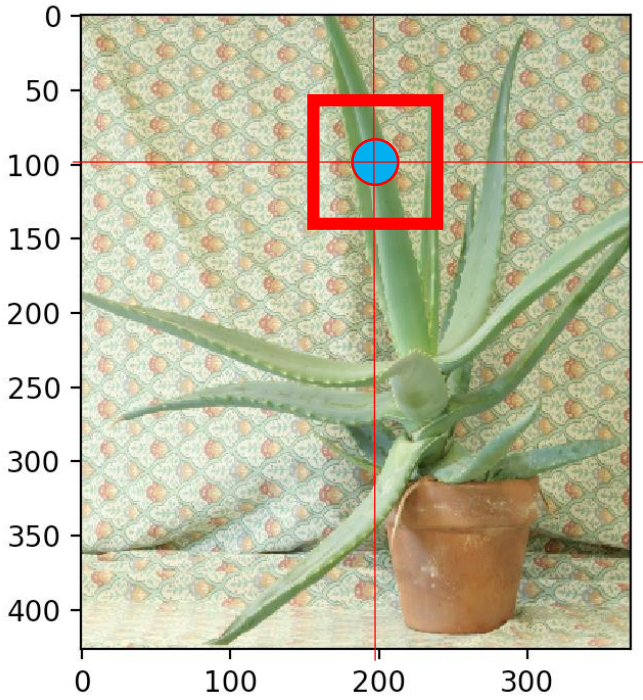
$$Z = \frac{B \cdot \mathrm{f}}{\Delta d}$$

$$\Delta d = d_1 - d_2$$

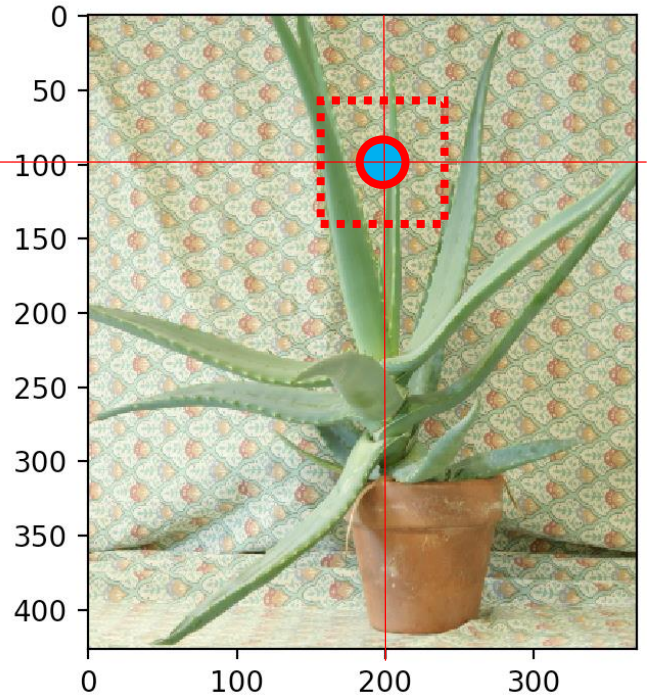
上面的Δd通过图片的匹配搜索得到

$$\Delta d = argmin_d f(x, y, d)$$

双目3D视觉原理——双目视觉计算细节



左图像
程序中为矩阵imgL



右图像
程序中为矩阵imgL

要求

- 计算左图像(200,100)位置图像在右图中的位置

搜索时用到的图像特性

- 左右图像中同一物体对应的纵坐标相同，仅仅横坐标不同，并且右图像中物体“偏左”移动。
- 假设左右图像对应像素的水平偏移不超过30像素

上述特性和假设帮助降低搜索的难度

搜索方案

- 在右图中，从(200,100)位置开始搜索，依次检查右图位置(200,100)、(200-1,100)、(200-2,100)… (200-30,100)的图像，看哪一个位置和左图(200,100)位置图像匹配度最高。
- 假设右图(200-d,100)位置匹配度最高，这就意味着在左图(200,100)处像素对应的物体在左右眼图像中的视差为d

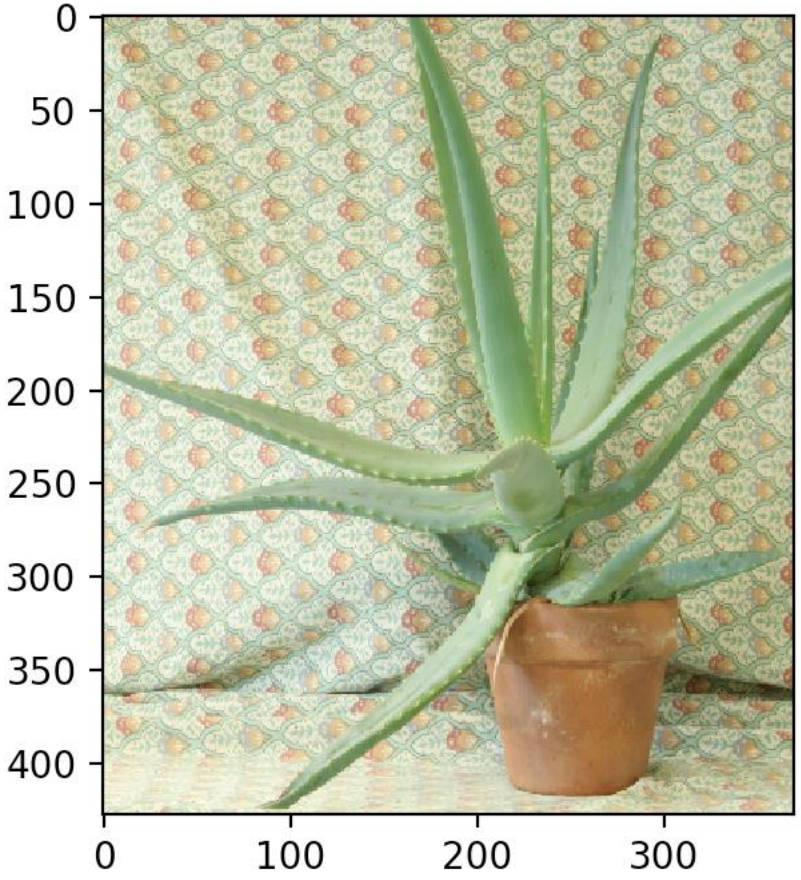
匹配度如何计算?

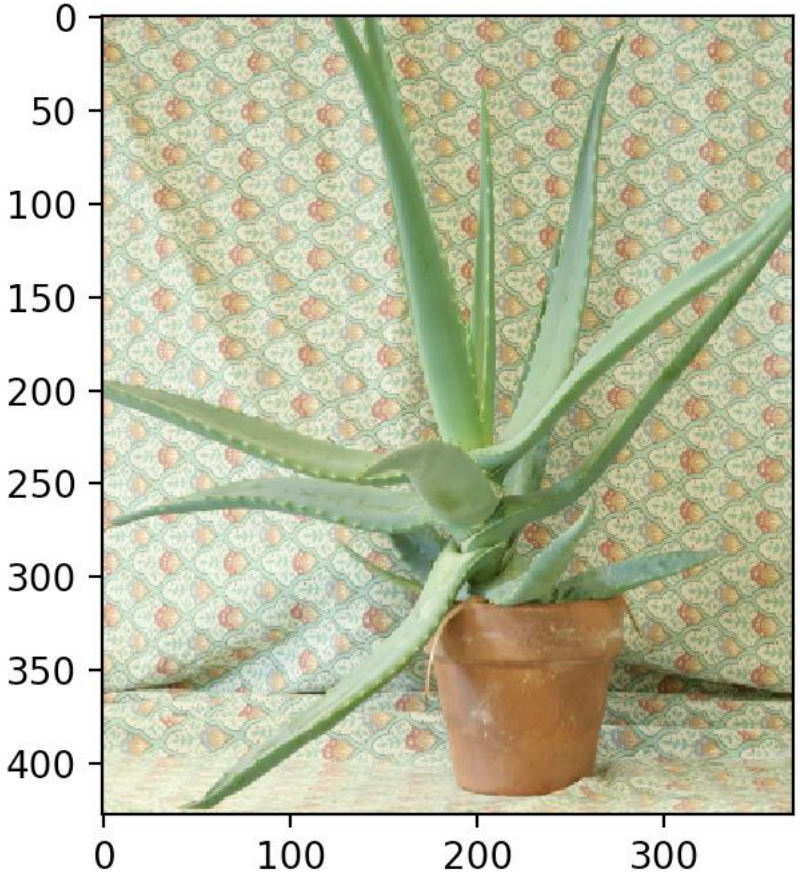
- 差别通过像素的RGB3 通道的色彩差的绝对值之和表示，该值越小，匹配程度越高
- 为了提高匹配的准确性，我们需要匹配左图以 (200,100) 为中心的一个窗口内所有的像素，计算该窗口内每个像素和右图搜索位置对应窗口内所有像素的色彩差的绝对值的和（SAD），作为匹配度指标$f(d)$，如右边公式所示
- 注意：公式中窗口尺寸参数W根据图像特性设置，比如这里的例子可以设W=2。

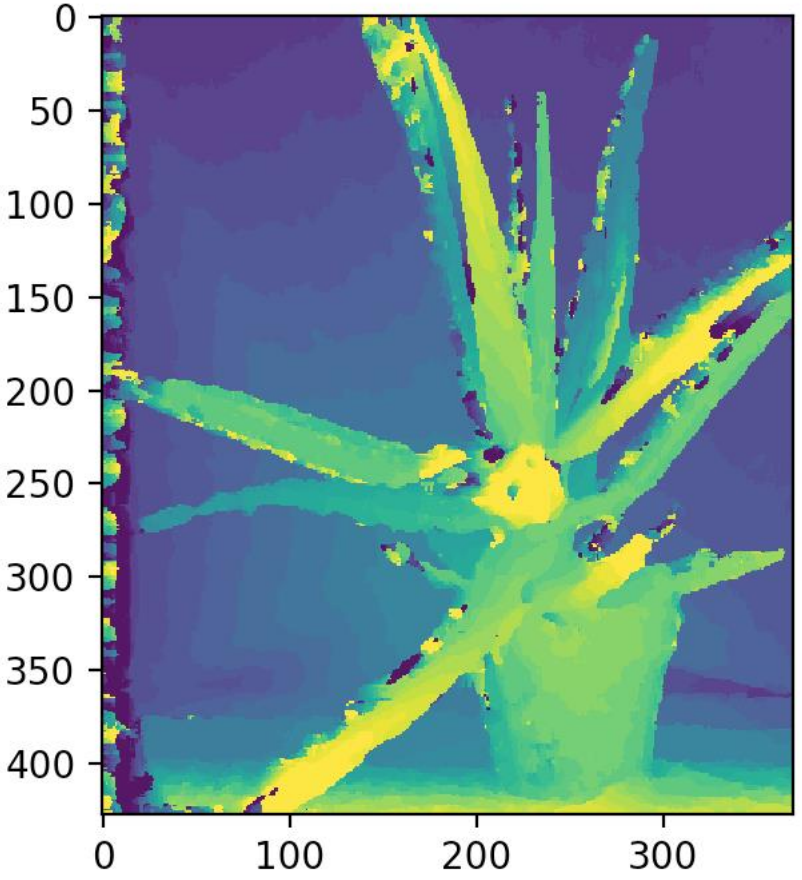
左图(x,y)位置像素和右图(x,y,d)位置像素的差异计算公式（SAD）：

$$f(x,y,d)=\Sigma_{d_x=-W}^{W}\Sigma_{d_y=-W}^{W}|I_L(x+dx,y+dy)-I_R(x-d+dx,y+dy)|$$

双目3D视觉原理——从视差到3D坐标

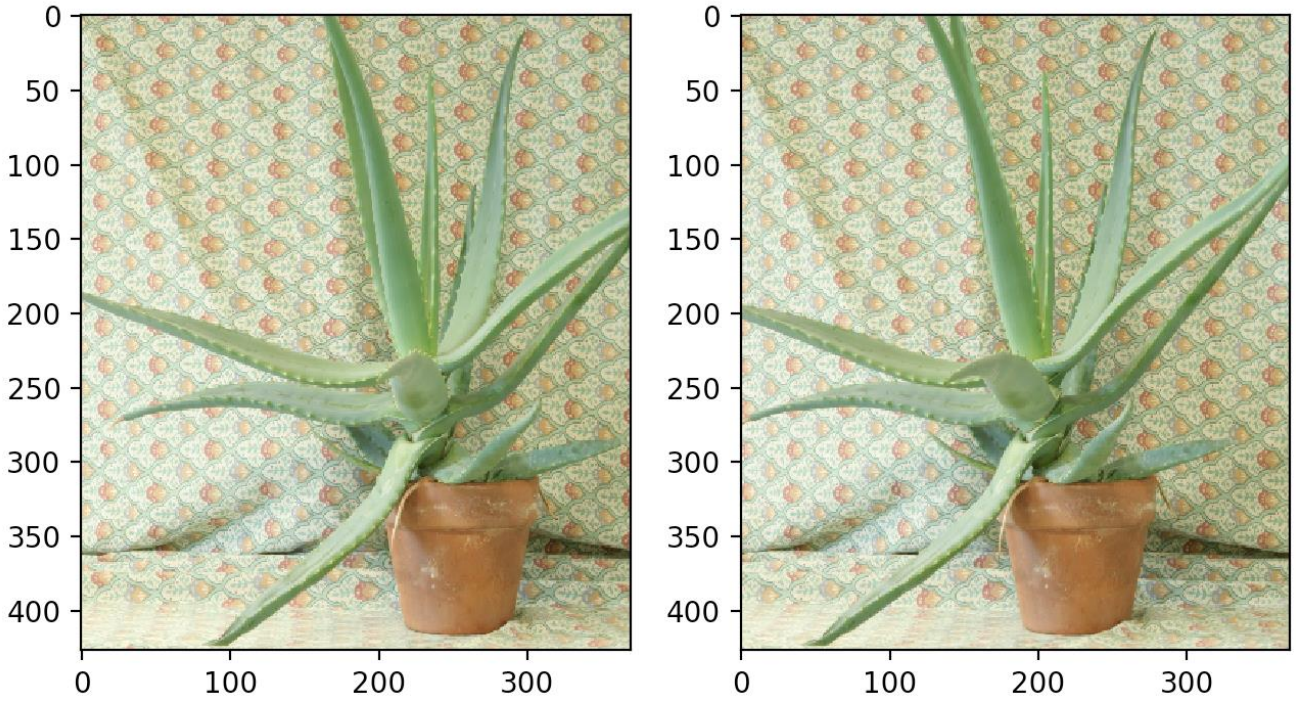


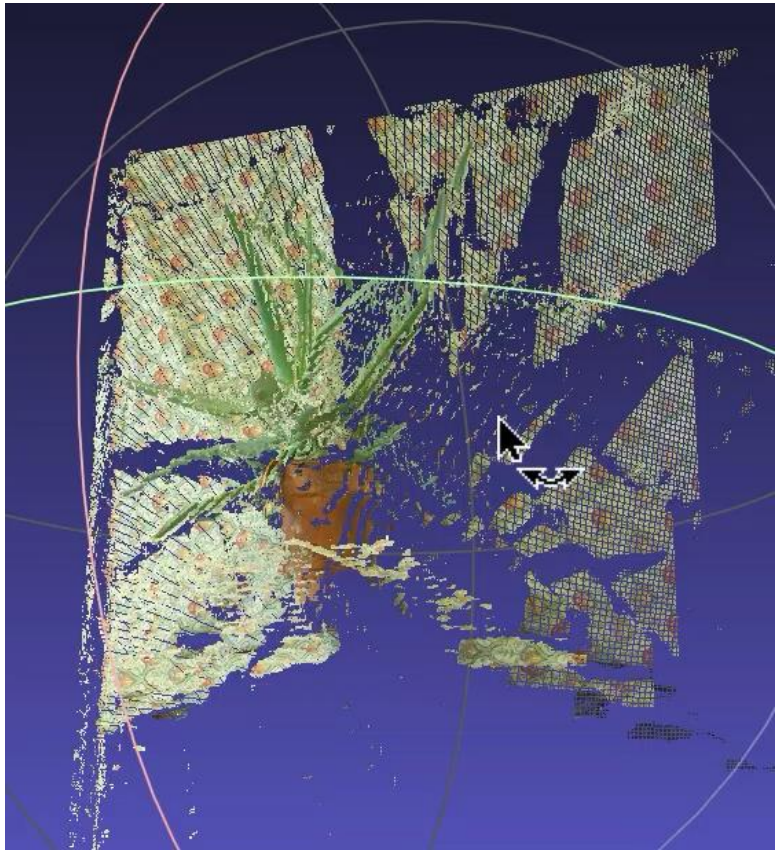




- 使用SAD算法计算视差图
- （SAD指图像差的绝对值之和）
- 图的左侧边沿不准，因为两个像素视野范围不同

双目3D视觉原理——从视差到3D坐标

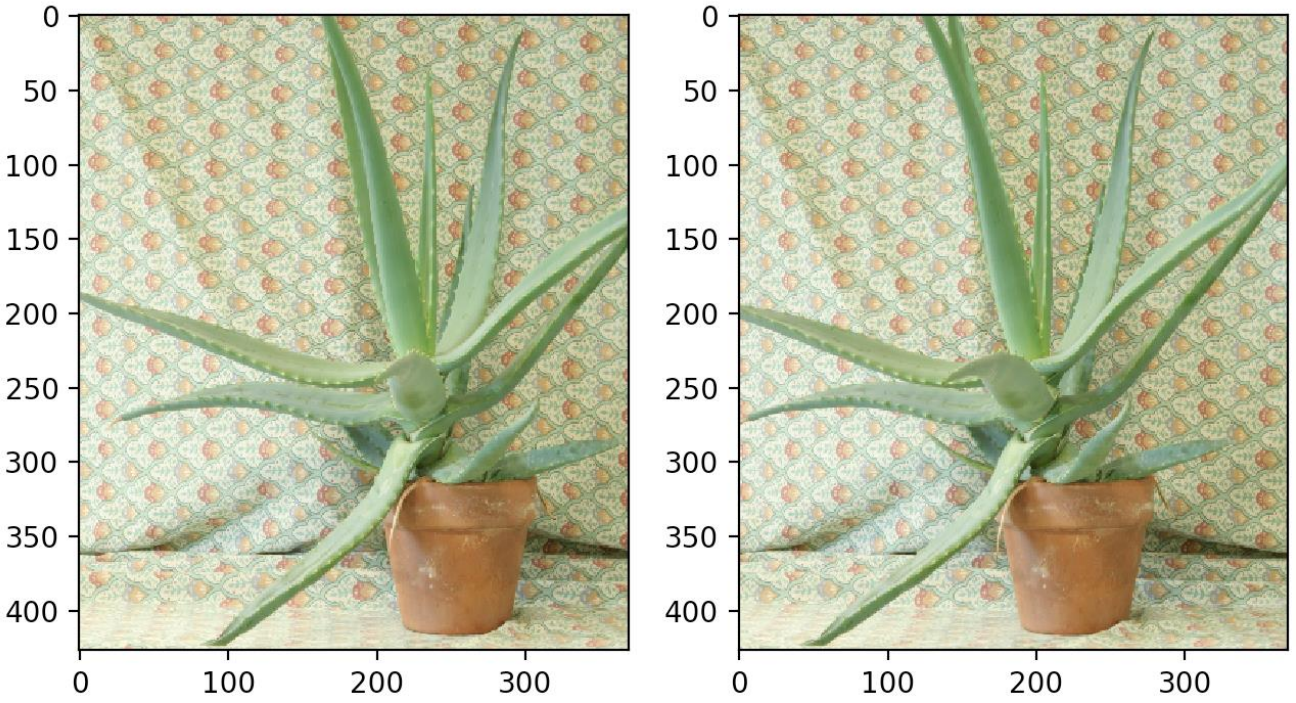


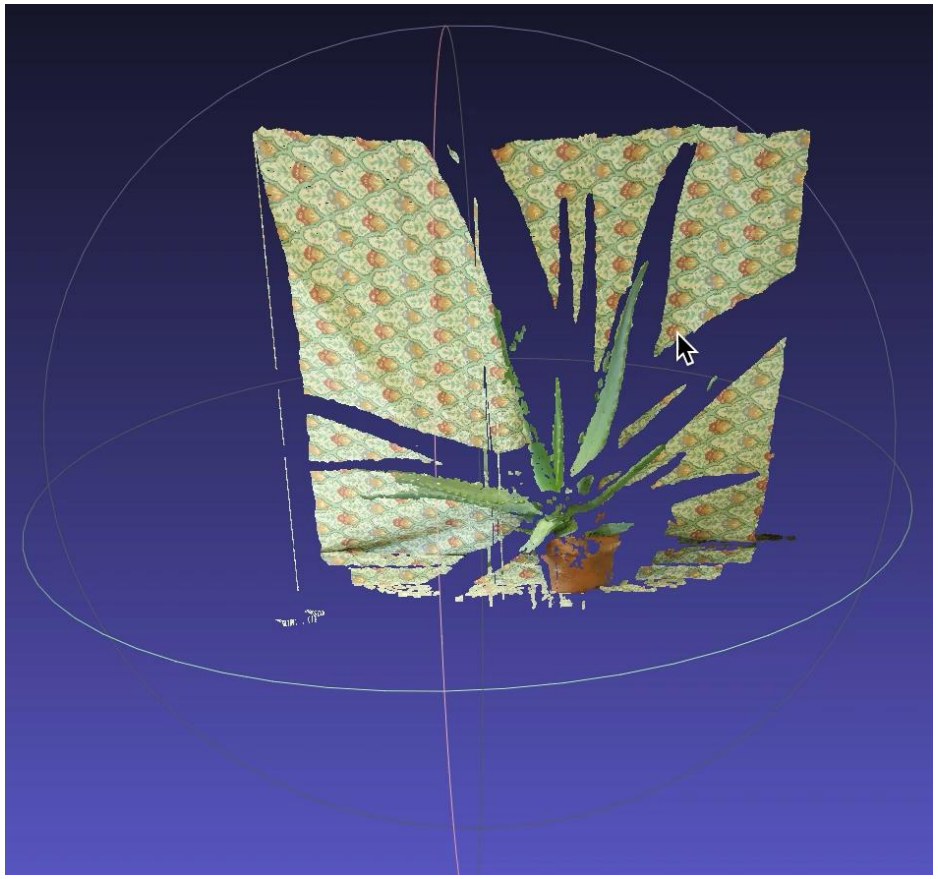


$$Z = \frac{B \cdot \mathrm{f}}{\Delta d}$$

- 使用SAD算法计算视差后转成点云（右边公式）
- 由于匹配不够精确,转成立体图后能看到“噪声”
- 图中Z距离“分片”，是因为视差计算结果是整数

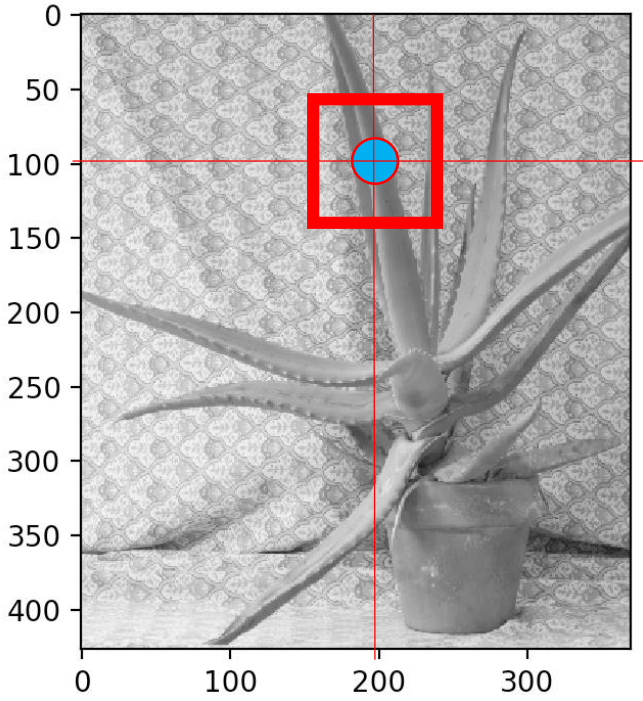
3D视觉原理——双目视觉，从视差到3D坐标



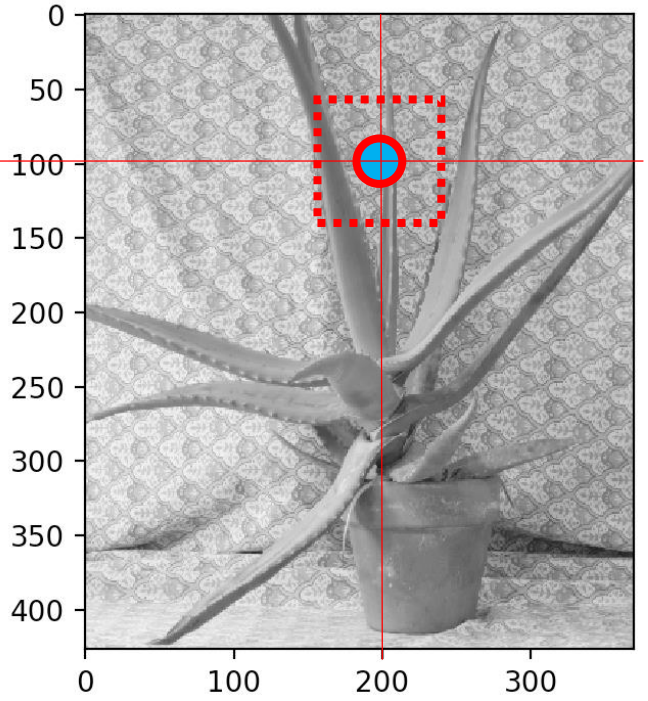


使用opencv内带的计算视差的API,得到更好的结果

双目3D视觉原理——双目视觉计算作业描述



左图像
程序中为矩阵imgL



右图像
程序中为矩阵imgL

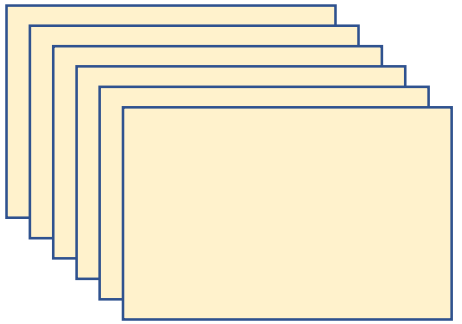
要求：

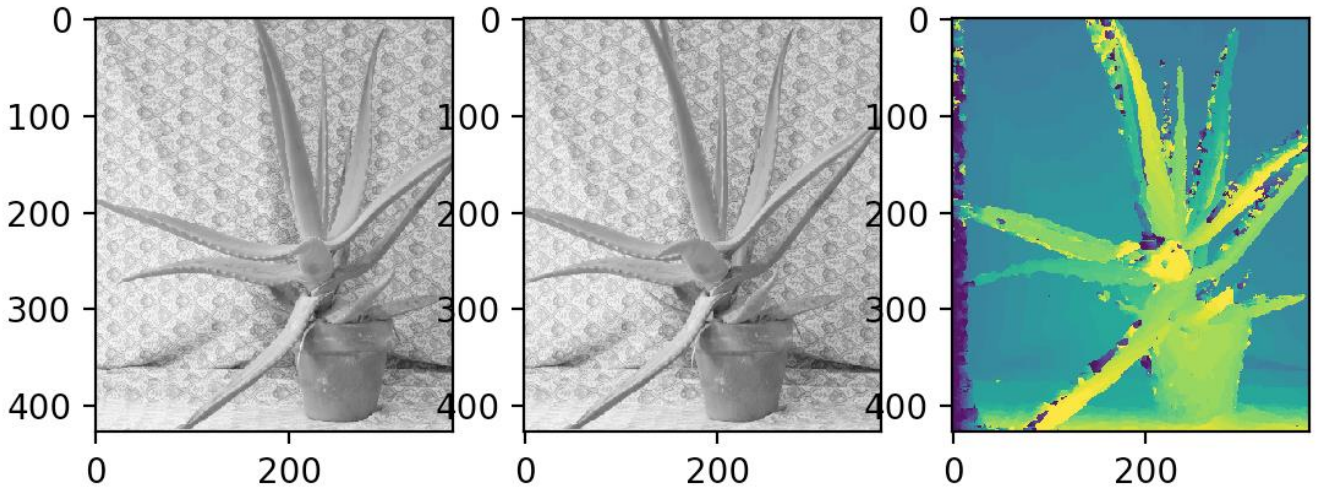
1. 编写程序，对左图（存储于文件aL_gray.csv中）中的每个像素，计算在右图（存储与文件aR_gray.csv中）中的最佳匹配位置，将匹配像素的水平位置差（取绝对值）存储在文件match.csv中
2. 文件match.csv代表的矩阵尺寸和被匹配图像一样大，里面每个元素的值是通过像素匹配得到的坐标差。
3. 为便于开发，提供了partial_code_stereo_match.py程序，可以通过修改改程序实现代码。
4. 作业提交match.csv文件即可

双目3D视觉原理——双目视觉计算作业提示


```python
# 构建一系列平移后的图img_shift
img_shift=np.zeros((D,HGT,WID))
for d in range(D):
    img_shift[d,:,:]=np.roll(imgR, d,axis=1)

# 计算左图和一系列平移后的右图的差,差别取绝对值
img_diff=np.abs(img_shift-imgL)

# 对图像差计算窗口平滑
for n in range(img_diff.shape[0]):
    img_diff[n,:,:]=cv2.boxFilter(img_diff[n,:,:],-1,(W,W))

# 逐个像素求最匹配的平移量
imgD=np.zeros((HGT,WID))
imgD=np.argmin(img_diff,axis=0)
```

img_shift

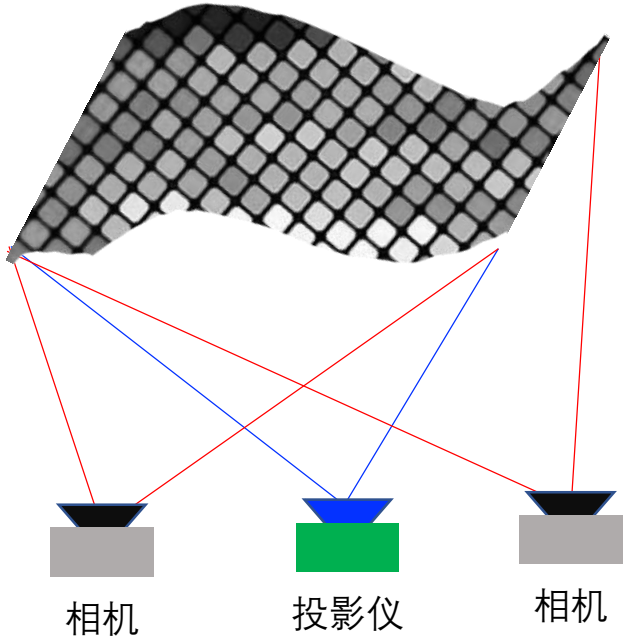




双目3D视觉原理——双目视觉

不适用于单调缺乏纹理的场景。因为双目视觉根据视觉特征进行图像匹配，所以对于缺乏视觉特征的场景 (如天空、白墙、沙漠等) 会出现匹配困难甚至匹配失败

双目3D视觉原理——双目视觉

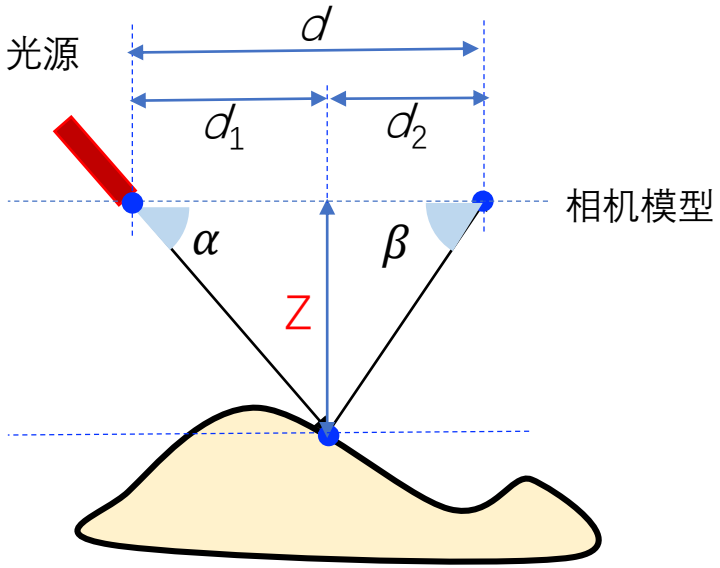


- 利用投影在物体上的图案，帮助两个相机计算双目匹配
- 有多种可能的投影图案，一个方案是使用伪随机散斑

内容概要

- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

结构光3D相机 ——点光源



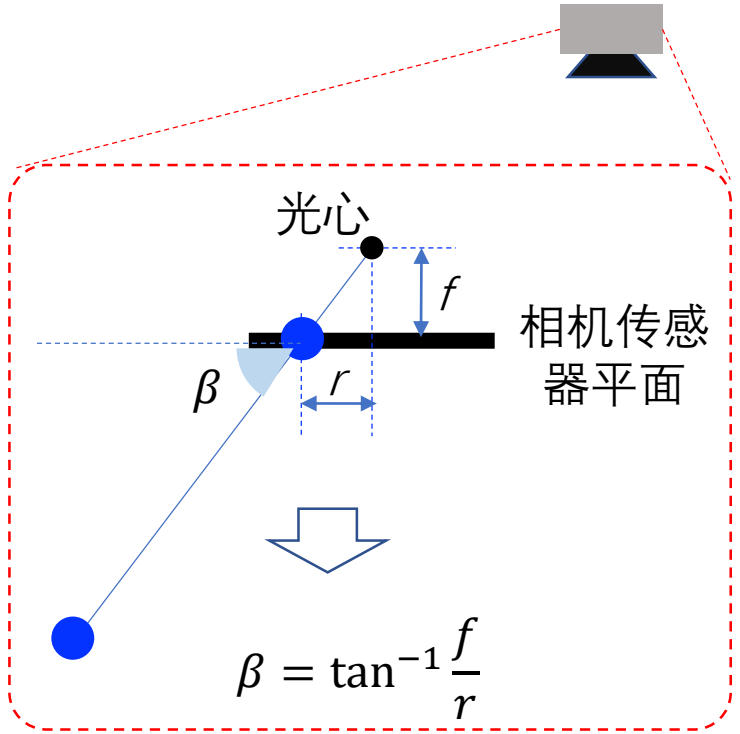
$$\frac{Z}{\tan\alpha}+\frac{Z}{\tan\beta}=d$$

$$Z=d\frac{\sin\alpha\sin\beta}{\sin(\alpha+\beta)}$$

物体表面光点到相机传感器平面的距离

β如何得到

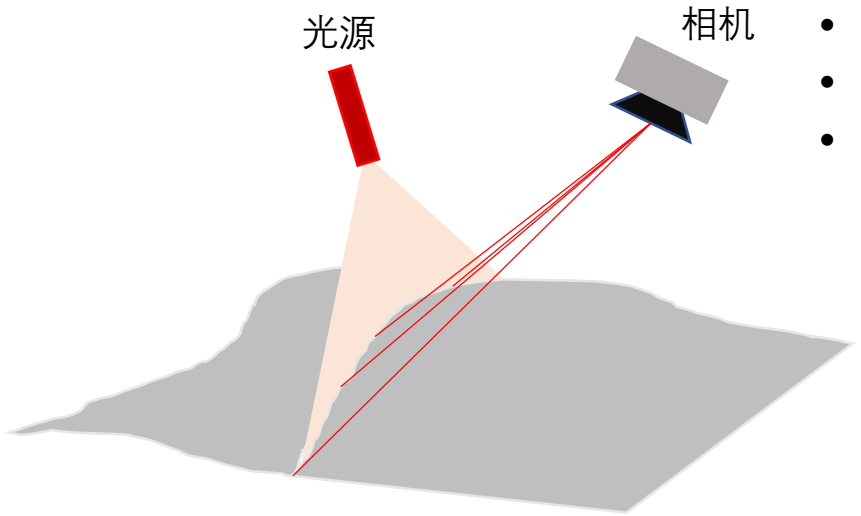
从图像中光点位置x计算β



$$\beta=\tan^{-1}\frac{f}{r}$$

通过光源转动进行光点扫描就能够计算出物体表面各个位置的距离Z值

结构光3D相机——线光源



- (x,y)是相机传感器平面上的点（已知）
- (X,Y,Z)是和(x,y)对应的空间点的坐标（未知）
- f是相机内参

$$\begin{cases} AX+BY+CZ+D=0 \\ X=\dfrac{xZ}{f} \\ Y=\dfrac{yZ}{f} \end{cases}$$

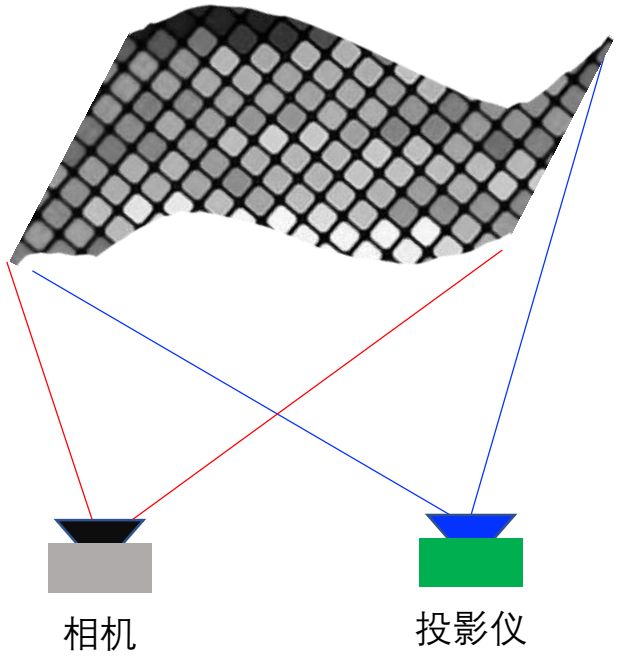
光源在相机坐标系中的方程

针孔相机模型

针孔相机模型带入平面方程

$$\begin{cases} Z=-\dfrac{Df}{Ax+By+Cf} \\ X=-\dfrac{Dx}{Ax+By+Cf} \\ Y=-\dfrac{Dy}{Ax+By+Cf} \end{cases}$$

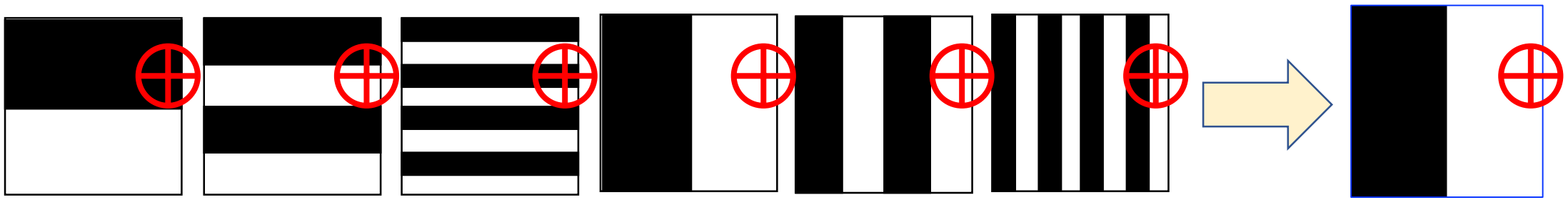
3D数据获取技术——结构光编码



投影一个面到物体，接收相机通过图案编码识别出是投影仪哪个角度的“射线”，并计算距离。

如何设计投影的图案编码，以及如何识别编码？

时间和空间编码方式

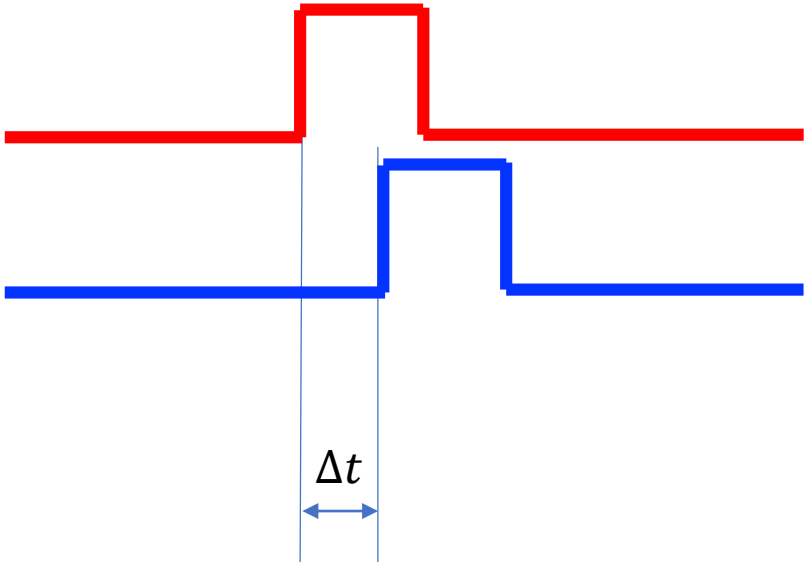


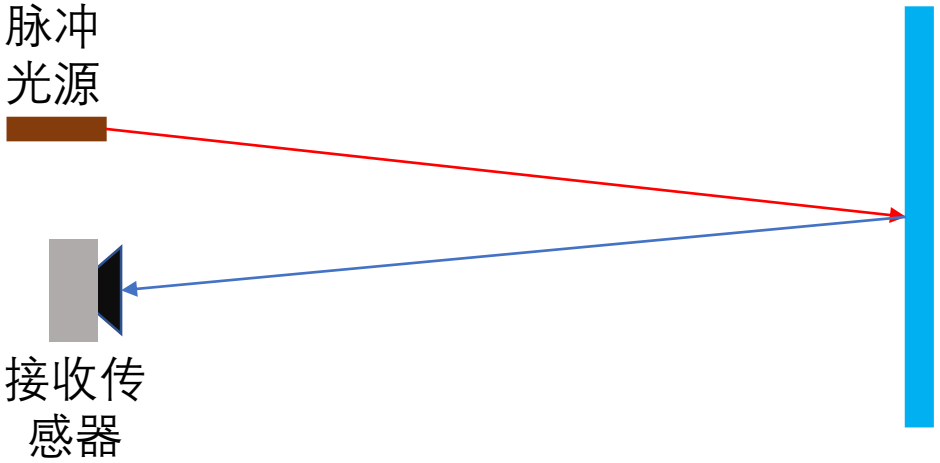
上图十字线对应的闪烁编码是101000，通过识别这一闪烁编码获知特定像素对应的投影图像中的位置，并进一步得到投影光线的角度

内容概要

- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

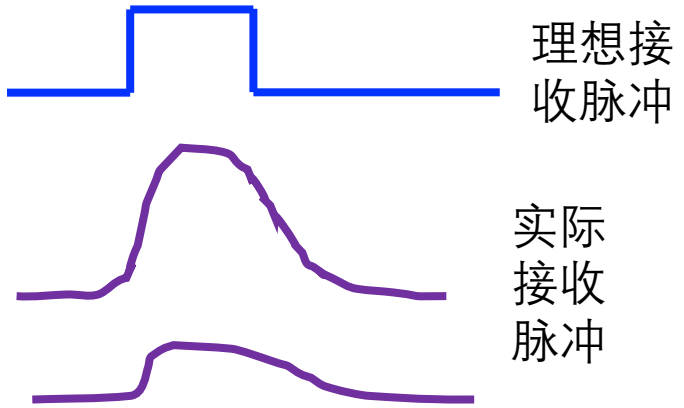
3D数据获取技术——TOF 3D成像原理（PTOF）



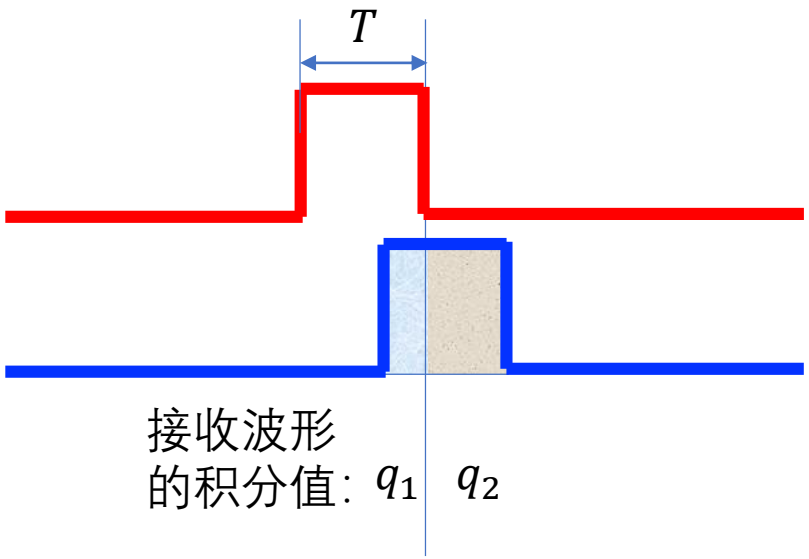


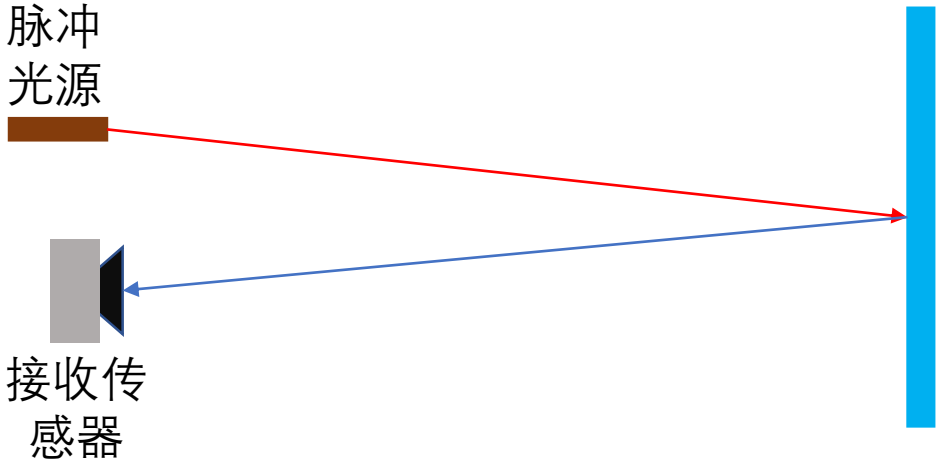
- 通过时间差计算得到总的路径长度：$c\Delta t$
- 当光源位置和相机光心几乎重合时，到反射面距离为$d = \frac{c\Delta t}{2}$

- 需要精确测量上升沿时间差收到距离、接收脉冲失真等影响
- 更加稳定的方案是检测整个脉冲波形面积比例



3D数据获取技术——TOF 3D成像原理（PTOF）





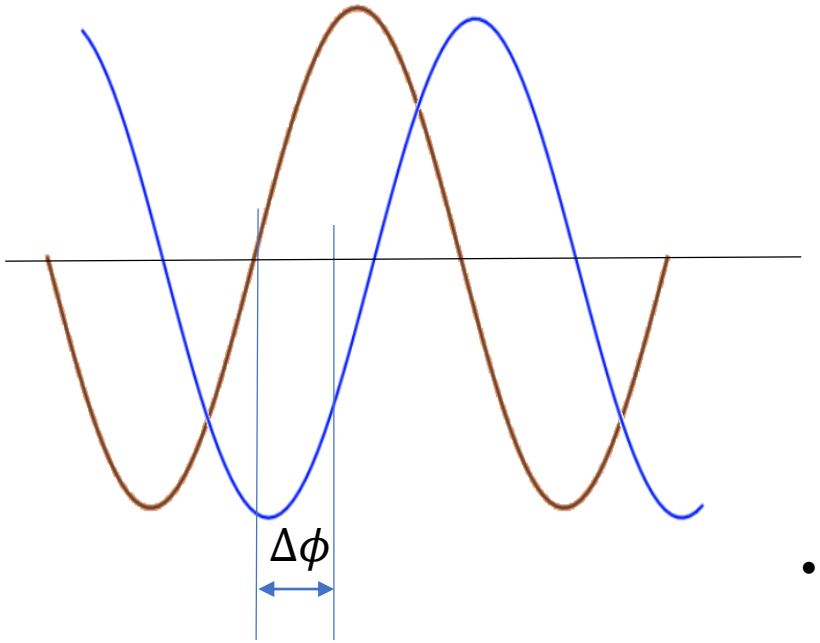
- 用积分比值反映延迟信息：$r = \frac{q_2}{q_1+q_2}$

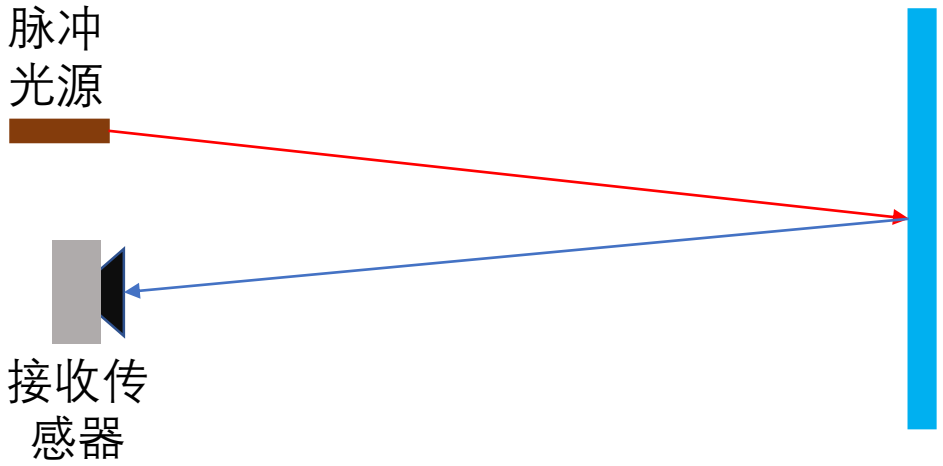
  $r = 0$对应$q_2 = 0$，接收发送信号重合，没有延迟

  $r = 1$对应$q_1 = 0$，接收发送信号延迟T

- 延迟量计算：$\Delta t = T\frac{q_2}{q_1+q_2}$
- 光飞行距离：$cT\frac{q_2}{q_1+q_2}$，考虑光的来回，物体到相机距离是：$d = \frac{cT}{2}\frac{q_2}{q_1+q_2}$
- 为提高精度，需要多个脉冲重复测量，将结果平均

3D数据获取技术——TOF 3D成像原理（CWTOF）



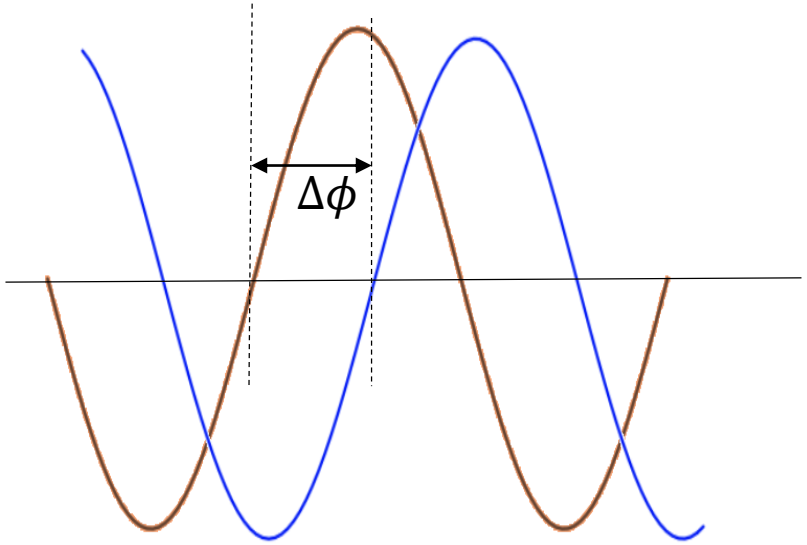


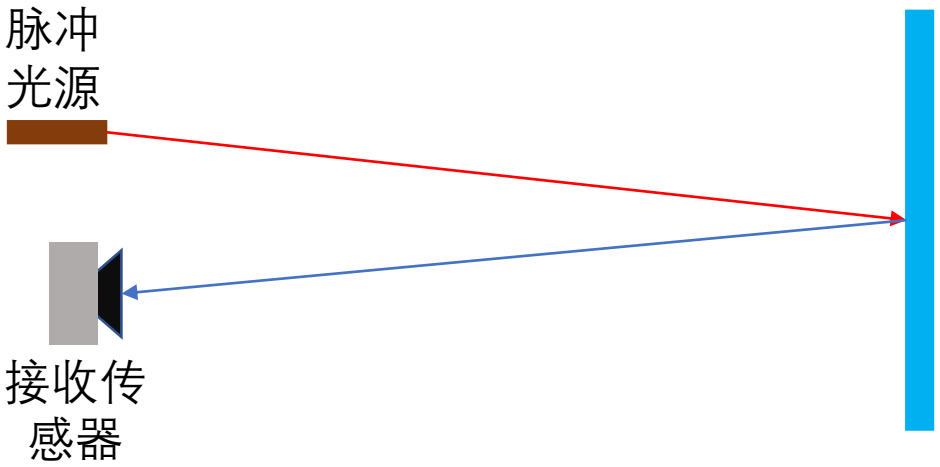
- 接收信号：$r(t) = 1 + a\cos(\omega t - \phi)$

飞行时间带来的相移

- 相移转成时间延迟：$\Delta t = T\frac{\phi}{2\pi}$（其中T是调制信号周期）
- 光飞行距离：$c\Delta t = cT\frac{\phi}{2\pi} = \frac{c}{F}\frac{\phi}{2\pi}$
- 考虑光的来回，物体到相机距离是：$d = \frac{c\phi}{4\pi F}$
- 如何计算相位？

3D数据获取技术——TOF 3D成像原理（CWTOF）



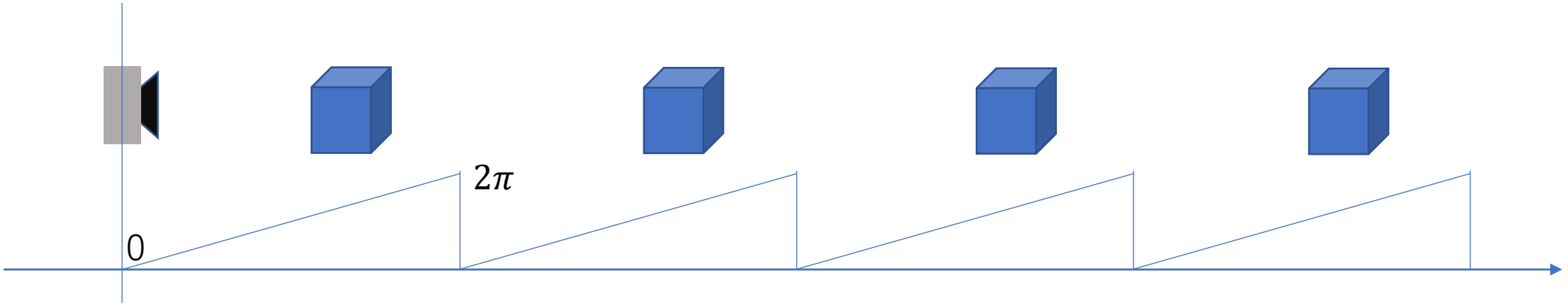


- 相位测量方法：通过积分器得到接收信号$r(t)$和若干特定参考信号$g_\theta(t)=\cos(\omega t+\theta)$的相关值
- 相关计算公式：$s_\theta=\lim_{t\to\infty}\frac{1}{t}\int_{-\frac{t}{2}}^{\frac{t}{2}}r(t)g_\theta(t)dt=\frac{a}{2}\cos(\phi+\theta)$
- 设置参考信号$g_\theta(t)$和发射信号相差θ分别为：0、0.5π 、π、1.5π，送入相关电路分别得到：
  $s_0=\frac{a}{2}\cos(\phi)$、 $s_{0.5\pi}=-\frac{a}{2}\sin(\phi)$、 $s_\pi=-\frac{a}{2}cos(\phi)$ 、 $s_{1.5\pi}=\frac{a}{2}\sin(\phi)$
- 上面对相关值分析表明

  相位差：$\phi=\operatorname{atan}\left(\frac{s_{1.5\pi}-s_{0.5\pi}}{s_0-s_\pi}\right)$　　以及　接收信号强度：$a=\frac{\sqrt{(s_{1.5\pi}-s_{0.5\pi})^2+(s_0-s_\pi)^2}}{2}$
- 计算过程复杂，但额外的好处——相关器就像带通滤波器，进一步降低了干扰

3D数据获取技术——TOF 3D成像原理

模糊度问题

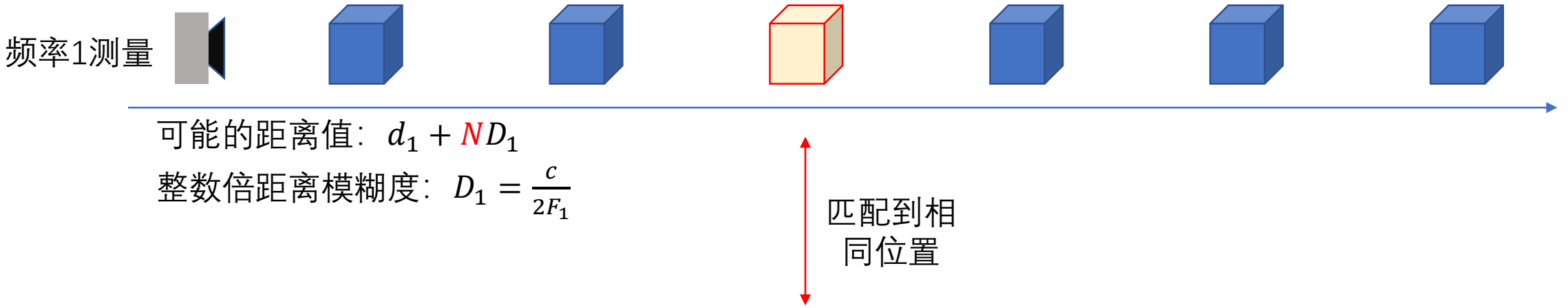


- 物体到相机的距离计算公式：$d = \frac{c\phi}{4\pi F}$
- 相位差从0变化到2π，对应距离从0变到$\frac{c}{2F}$
- 对于更远距离的信号，相位“折回”到$0 \sim 2\pi$区间，导致模糊度:
- 没有模糊度条件下，最长测量距离$\frac{c}{2F}$
- 频率F越低、无模糊度测量距离越长（消除模糊度的一个方案）
  - F越低、固定的相位测量分辨率对应的距离分辨率越差

$$d_N = \frac{c\phi}{4\pi F} + N\frac{c}{2F}$$

物体可能的距离有多个可能

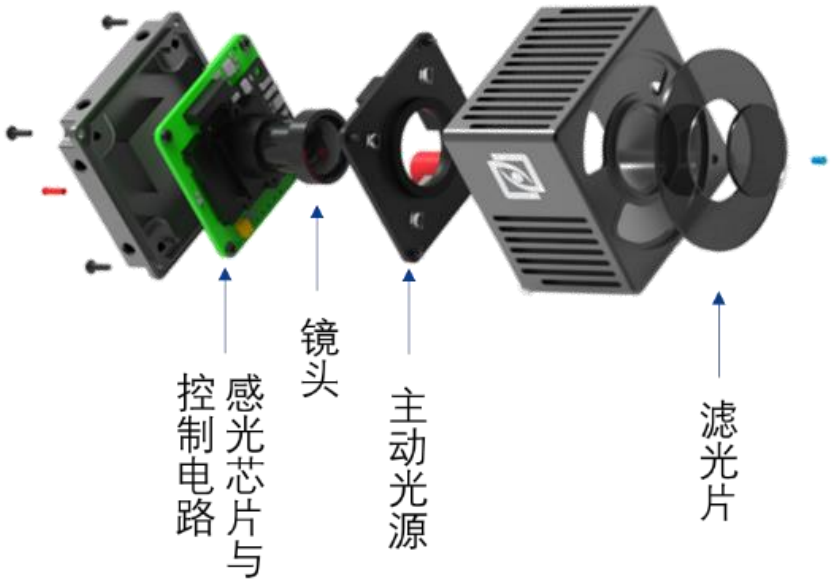
3D数据获取技术——TOF 3D成像原理

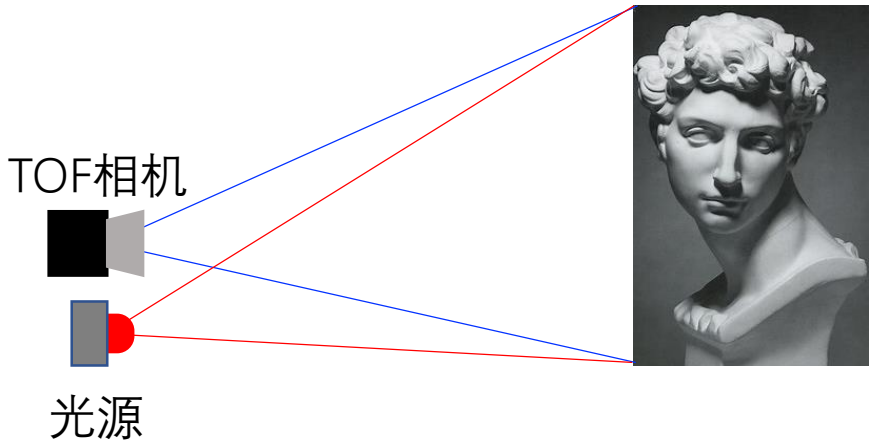


- 利用两个距离匹配去除距离模糊度
- 通过这一方式，等效的调制频率是F_1和F_2的最大公约数F
- 由于F足够小，整数倍距离模糊度对应的距离更长

3D数据获取技术——TOF 3D成像原理

- 光源照亮一片区域
- 接收器通过镜头将反射光聚焦到传感器芯片不同位置
- 传感器芯片上内置鉴相电路计算各像素相位差，对应到特定“视线方向”的物体距离
- 一般光源和TOF相机镜头尽可能靠近测量的光线飞行距离大约是物体到镜头距离的1/2

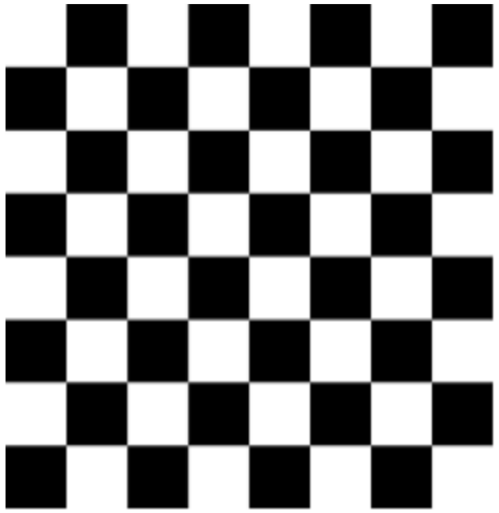


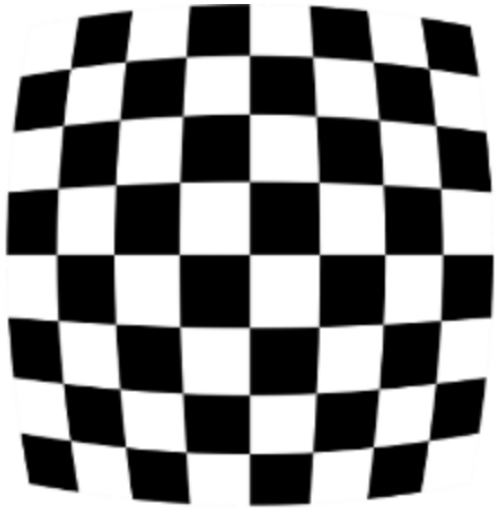


内容概要

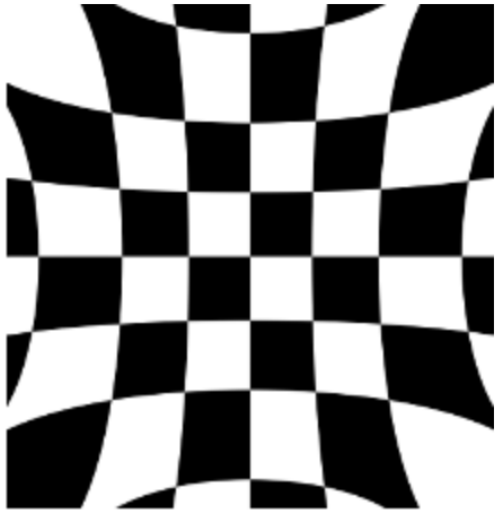
- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

镜头畸变矫正





桶形失真



枕形失真

$$\begin{bmatrix} x_{corr} \\ y_{corr} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix}$$

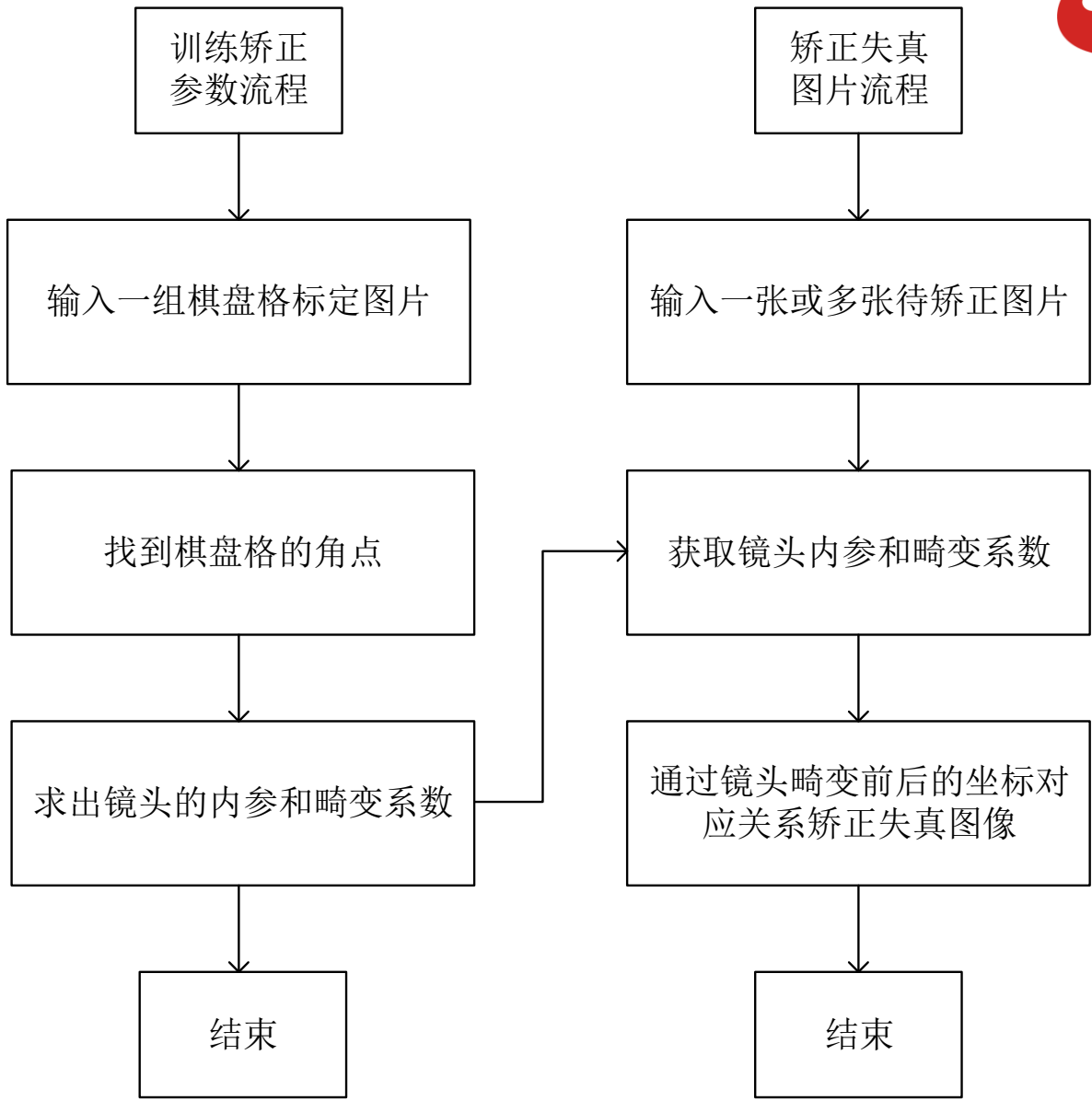
通过畸变模型
调整像素位置

其中:

- (x, y)是带有失真的像素坐标
- $r^2 = x^2 + y^2$是像素到图形中心距离
- (x_{corr}, y_{corr})是带有像素坐标修正值

失真矫正参数$\{k_1, k_2, k_3, p_1, p_2\}$

镜头畸变矫正

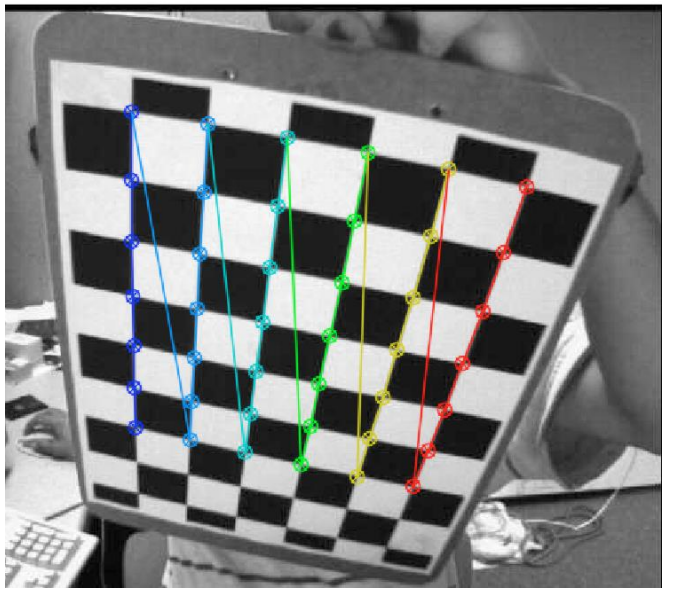

训练矫正
参数流程
输入一组棋盘格标定图片
找到棋盘格的角点
求出镜头的内参和畸变系数
结束
矫正失真
图片流程
输入一张或多张待矫正图片
获取镜头内参和畸变系数
通过镜头畸变前后的坐标对
应关系矫正失真图像
结束

基于opencv的镜头校准例程

参考代码：demo_calib.cv2.py，来自opencv的官方例程

```
30  for fname in images:
31      print(fname)
32      img = cv2.imread(fname)
33      gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
34
35      # 角点粗检测
36      ret, corners = cv2.findChessboardCorners(gray, (7,6),None)
37      if not ret:
38          continue
39      else:
40          objpoints.append(objp)
41
42      # 角点精检测
43      corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
44      imgpoints.append(corners2)
```

通过粗检测/精细检测两级，得到棋盘格角点的坐标（像素坐标）



```
58  ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)
59  print('mtx:',mtx)       # 内参矩阵
60  print('dist:',dist)     # 失真参数
```

计算矫正模型参数

$$\begin{bmatrix} x_{corr} \\ y_{corr} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix}$$

$$\begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}$$

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

基于opencv的镜头校准例程

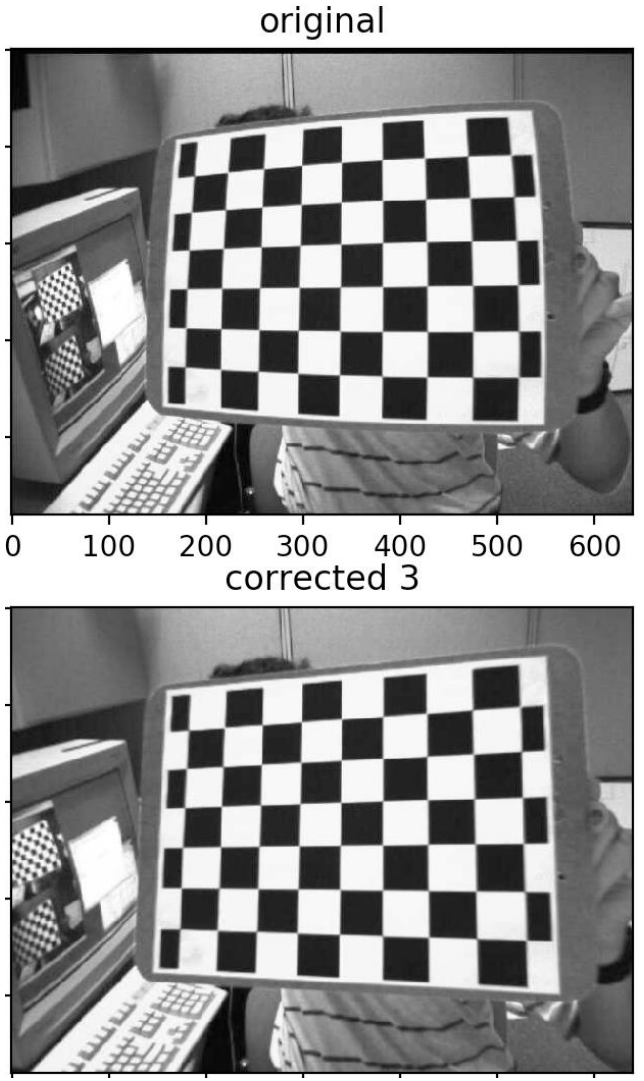
- 图像矫正方法
- 注意入参对应的两个矩阵
- 原始图像中弯曲的边界已经直了
- undistort有多种调用方法，请参考opencv文档

```
104     dst = cv2.undistort(img,mtx,dist)
```

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

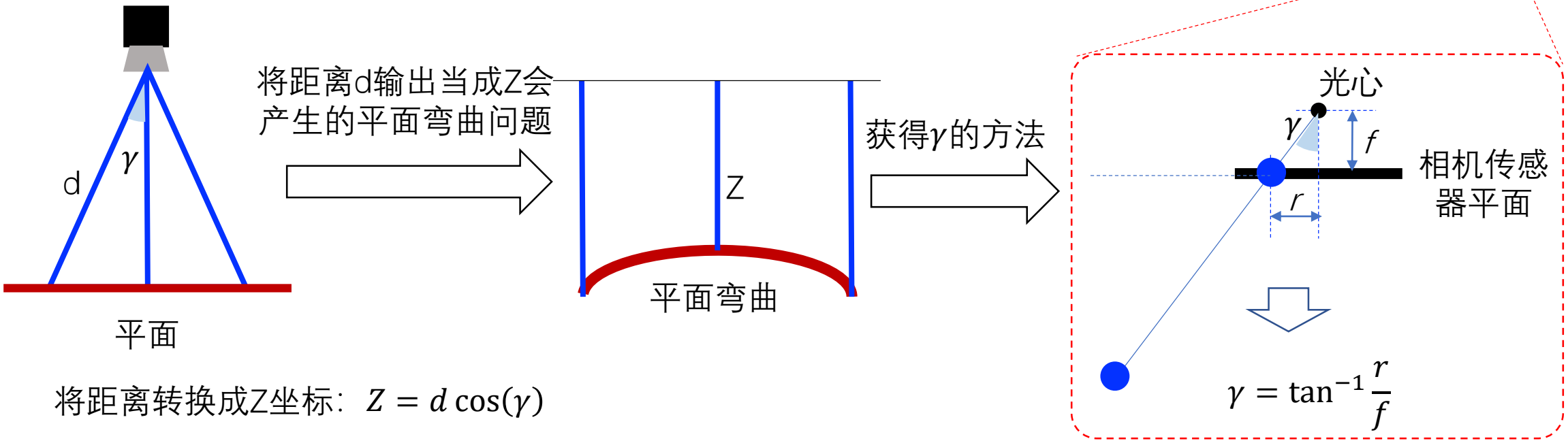
$$\begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}$$

$$\begin{bmatrix} x_{corr} \\ y_{corr} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix}$$



TOF 3D数据从距离数据计算Z

- 如果深度相机输出的数据是像素到镜头的距离，那么需要按照视线 “射线”进行额外的转换
- 如果将距离d输出当成Z会产生的平面弯曲问题



将距离转换成Z坐标：$Z = d\cos(\gamma)$

内容概要

- 3D光学测量方法概述
- 针孔相机模型
- 双目3D视觉原理
- 结构光3D成像原理
- TOF 3D成像原理
- 图像畸变与矫正
- RGB-D融合概述

RGB-D融合

为从深度相机得到的每个点云“染色”

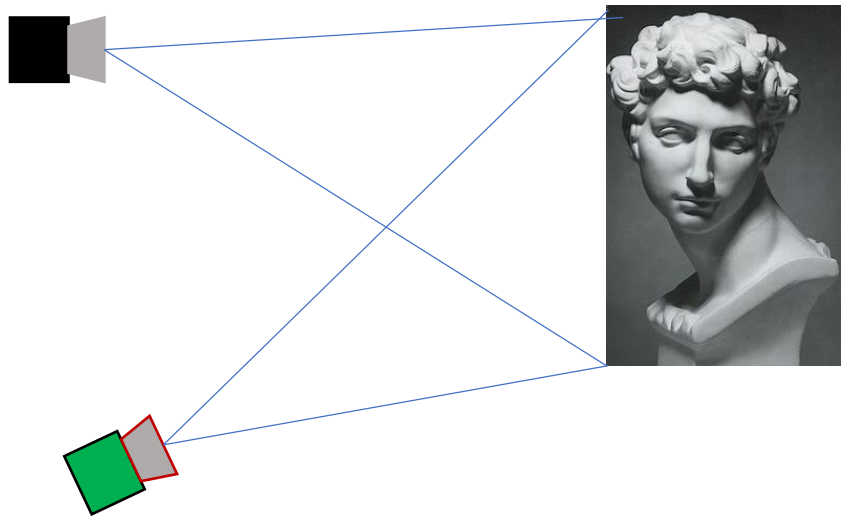
RGB相机中的像素位置
（和TOF相机坐标系
下的点云对应）

TOF相机坐标系
下的点云坐标

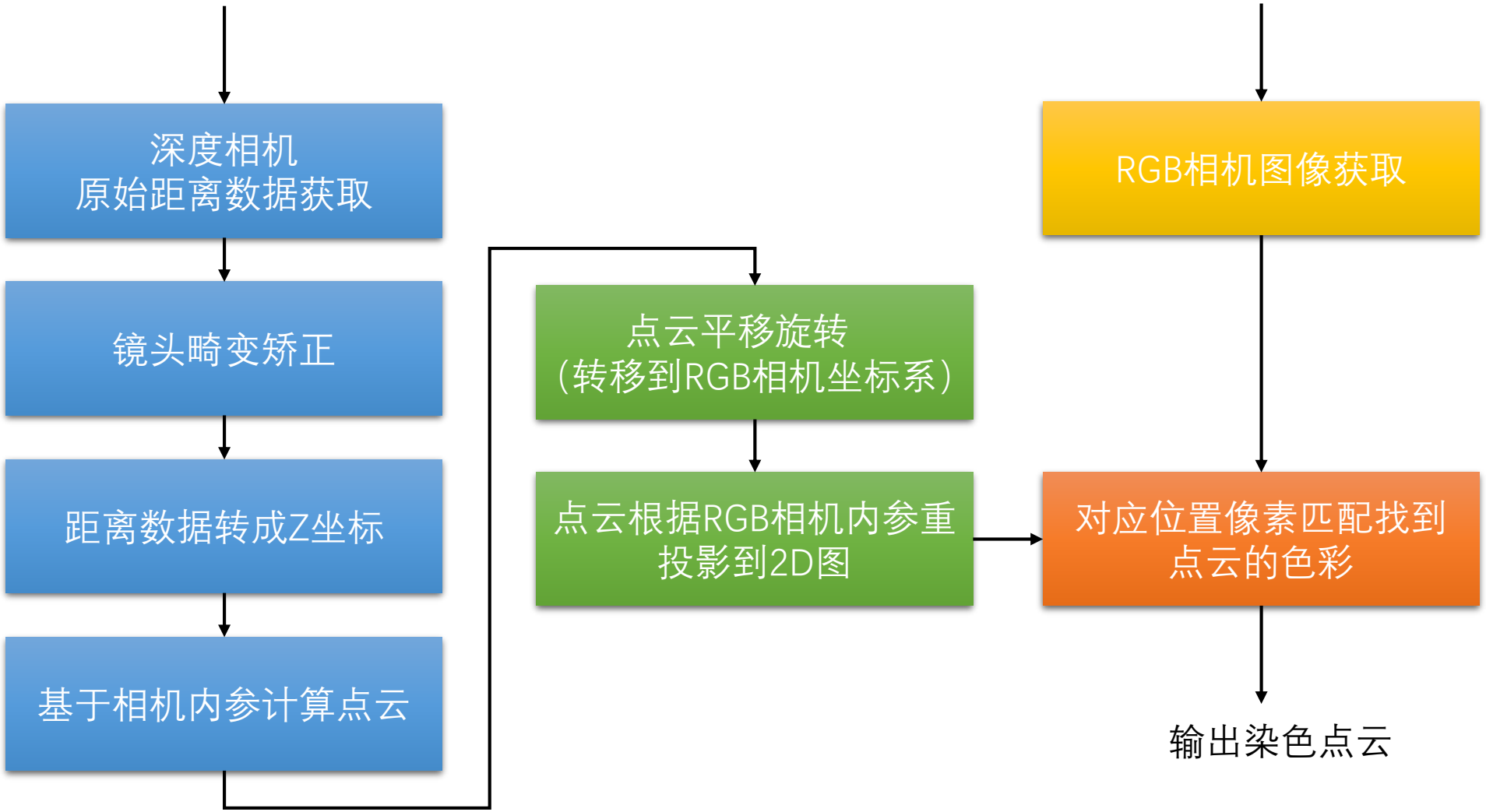
$$Z\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

RGB相机
的内参

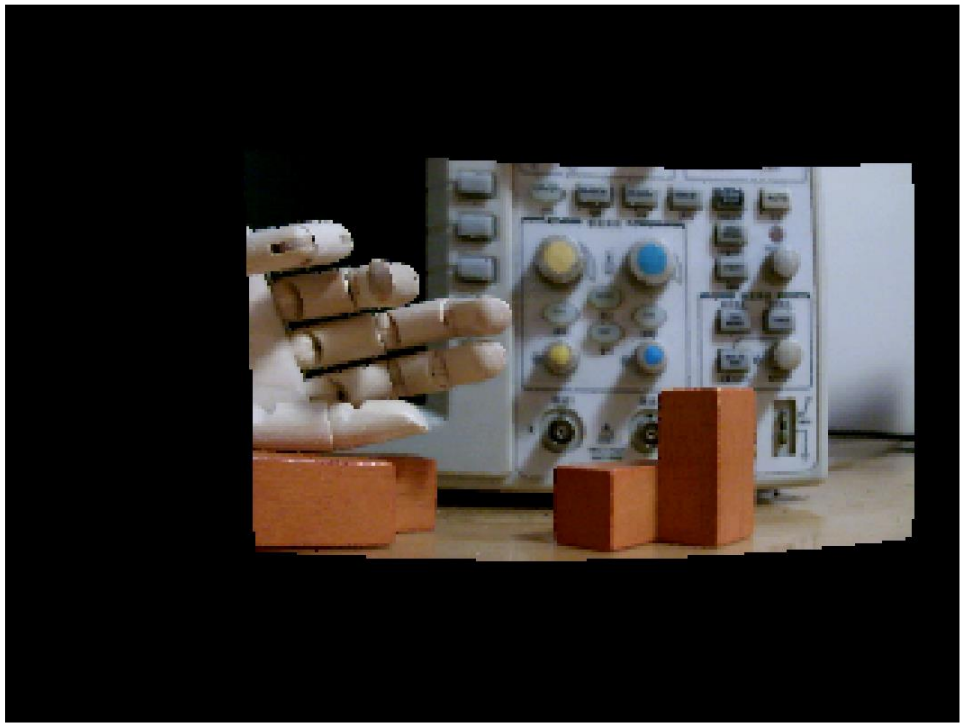
TOF相机坐标系转到
RGB相机坐标系

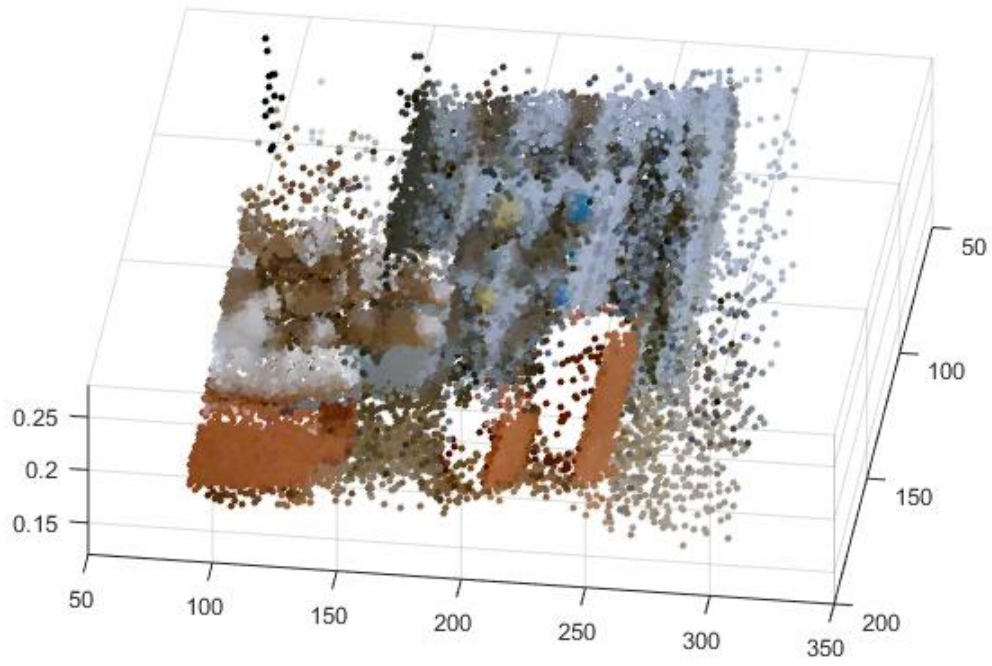


RGB-D融合流程


深度相机
原始距离数据获取
镜头畸变矫正
距离数据转成Z坐标
基于相机内参计算点云
点云平移旋转
（转移到RGB相机坐标系）
点云根据RGB相机内参重
投影到2D图
RGB相机图像获取
对应位置像素匹配找到
点云的色彩
输出染色点云

RGB-D融合——RGB图色彩映射到深度图的结果





END