

waeceo的专栏

目录视图

摘要视图

RSS 订阅

个人资料



waeceo

访问：46065次

积分：838

等级：BLOG 3

排名：千里之外

原创：32篇

转载：24篇

译文：0篇

评论：25条

文章搜索

文章分类

window2003权限管理及设置 (1)

opencv (5)

opengl (1)

win7 (1)

qt (1)

sql (3)

C# (4)

USB (1)

CyUSB (1)

CyPress (1)

USB通讯 (1)

java (13)

数据类型 (0)

摄像头 (3)

rgb (1)

avicap (1)

灰度值 (2)

avicap32 (1)

graphics (2)

【观点】人工智能会不会取代开发它的人？

CSDN日报20170410 ——《未经检视的人生不值得活》

【福利】入门

技术初探

博客搬家，有礼相送

世界坐标系和相机坐标系,图像坐标系的关系

标签：[世界坐标系](#) [摄像机坐标系](#) [图像坐标系](#)

2016-01-25 15:3511619人阅读评论(4)收藏举报

分类：[opencv \(4\)](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

一、四个坐标系简介和转换

相机模型为以后一切标定算法的关键，只有这边有相当透彻的理解，对以后的标定算法才能有更好的理解。本人研究了好长时间，几乎每天都重复看几遍，最终才会明白其推导过程。

我觉得首先我们要理解相机模型中的四个平面坐标系的关系：像素平面坐标系（ u,v ）、像平面坐标系（图像物理坐标第 (x,y) ）、相机坐标系（ X_c,Y_c,Z_c ）和世界坐标系（ X_w,Y_w,Z_w ），在每一篇介绍相机模型的文章中都有介绍。

我刚开始理解时，看着那一堆的公式十分的头晕，我相信很多初学者和我一样，但仔细想想，只不过是，我们假设了一些参数，使四个坐标系之间的坐标联系起来，这样我们就可以从拍摄的图片上一个点坐标一路反推出世界中的那个点的坐标，这样就达到了我们的目的，三维重建。而那些我们假设的参数，就是我们要标定的内外参数。



1、像素坐标与像平面坐标系之间的关系

确定他们的关系之前，我们可以假设每一个像素在 u 轴和 v 轴方向上的物理尺寸为 dx 和 dy 。仔细看下他们的模型可以推出以下公式（这个还是比较好理解的）：

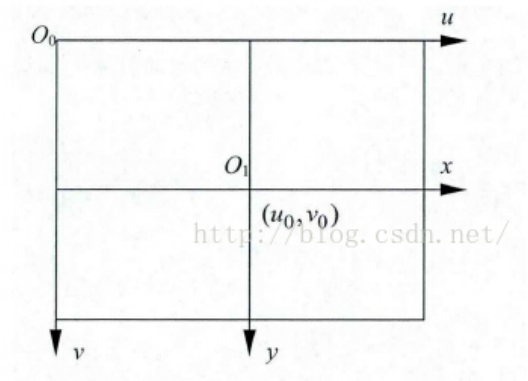


图 2.1 图像坐标系

[AForge](#) (1)
[build path](#) (1)
[classloader](#) (1)
[webroot](#) (1)
[classpath](#) (1)
[JUnit](#) (1)
[测试](#) (1)
[eclipse](#) (1)
[数学](#) (1)
[计算机视觉](#) (6)
[双目视觉](#) (2)
[摄像机](#) (2)
[spring](#) (9)
[jar](#) (1)
[个人提升](#) (2)
[Linux](#) (1)
[springmvc](#) (7)
[mybatis](#) (1)
[Redis](#) (1)
[javascript](#) (1)
[机器学习](#) (1)
[mysql](#) (0)
[dubbo](#) (1)
[git](#) (2)
[idea](#) (1)
[maven](#) (1)
[sso](#) (1)
[学习心得](#) (1)
[web通讯](#) (1)
[笛卡尔积](#) (1)
[并发编程实践](#) (2)
[elasticSearch](#) (2)

文章存档

[2017年04月](#) (2)
[2017年03月](#) (5)
[2017年02月](#) (2)
[2017年01月](#) (5)
[2016年12月](#) (13)

展开

阅读排行

[世界坐标系和相机坐标系](#) (11575)
[C#与USB设备通信](#) (4852)
[C#读取摄像头并对图像做](#) (3614)
[eclipse配置spring\(最新\)](#) (2584)
[张正友标定法的真实理解](#) (2350)
[Spring AOP配置中的问题](#) (2349)
[镜头畸变矫正](#) (1742)
[Eclipse中如何添加JUnit](#) (1441)
[C#读取摄像头处理图片A](#) (1210)
[Sql的优化](#) (885)

$$u = \frac{x}{dx} + u_0$$

$$v = \frac{y}{dy} + v_0$$

解释：1、dx,dy,u0,v0其实都是我们假设出来的参数，dx,dy表示感光芯片上像素大小，是连接像素坐标系和真实尺寸坐标系的，u0,v0是图像平面中心，最终是得到内外参数。

得出这个公式后我们可以运用线性代数的知识把方程用矩阵形式表示：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

当然我们也可以另一种矩阵形式表示：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} dx & 0 & -u_0 dx \\ 0 & dy & -v_0 dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

2、相机坐标系与世界坐标系之间的关系

这两个坐标系之间的关系我们可以旋转矩阵R和平移矩阵T来得到以下关系：

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = L_w \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

公式4

解释：1、在这个公式中，R为3*3矩阵，T为3*1，0为(0,0,0)，简化用Lw表示后为4*4矩阵。

3、成像投影关系（相机坐标系与像平面坐标系）

评论排行

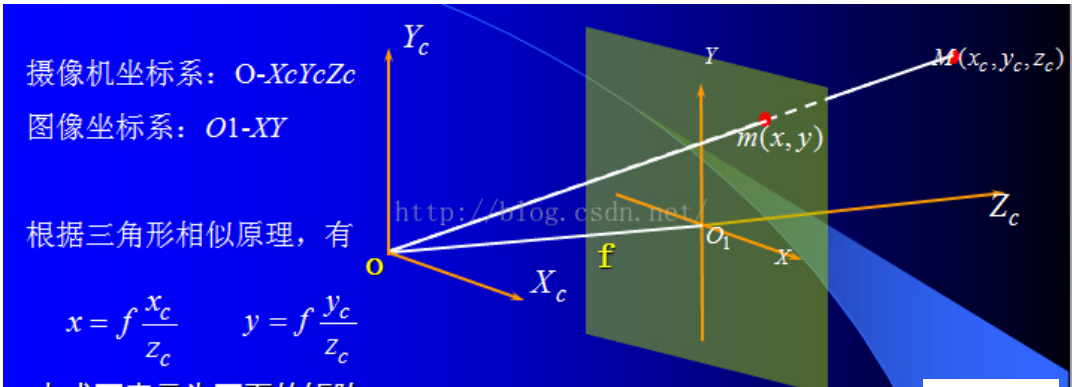
- C#与USB设备通信 (13)
- 世界坐标系和相机坐标系 (4)
- 镜头畸变矫正 (3)
- eclipse配置spring(最新 (2)
- Eclipse中如何添加JUnit j (1)
- 摄像机成像模型 (1)
- padding-top、margin-to (0)
- Sql复杂查询 (0)
- 面向对象三大特点：封装 (0)
- Java与其他语言的对比 (0)

推荐文章

- * 【《Real-Time Rendering 3rd》提炼总结】(一) 全书知识点总览
- * CSDN日报20170409 —— 《扯蛋的密码规则》
- * Shader2D: 一些2D效果的Shader实现
- * 一个屌丝程序员的人生（六十一）
- * 自定义控件三部曲视图篇（三）——瀑布流容器WaterFallLayout实现
- * 面向服务的体系架构（SOA）——架构篇

最新评论

- 世界坐标系和相机坐标系,图像坐
woshishazhu96: 我也觉得讲的
很好，起码我从零到看懂了大
概，不过2楼说得对，前面希望
补充，还有，后面的讲了旋转的
R，然...
- 世界坐标系和相机坐标系,图像坐
woshishazhu96: 我也觉得讲的
很好，起码我从零到看懂了大
概，不过2楼说得对，前面希望
补充，还有，后面的讲了旋转的
R，然...
- C#与USB设备通信
bdb1018: 楼主您好；我用
myDev = usbDevices as
CyUSBDevice; 得到的myDe...
- C#与USB设备通信
baidu_37552881: 问下，你那
cyusb.dll 从哪里弄的
- C#与USB设备通信
Handsome五爷: 求cyapi.dll的文
档呀。。。
- C#与USB设备通信
Handsome五爷: 能不能贴出
cyusb的下载连接呀？
- 世界坐标系和相机坐标系,图像坐
阿木寺: 关于坐标系，文章介绍了
两种，内容很赞，但文中的第二
篇介绍所涉及的图像坐标
(x,y)->像素坐标(u,...
- Eclipse中如何添加JUnit.jar包
范晓权: 使用文章，用到了。
- C#与USB设备通信
waeceo: @qq_25159513:没有
识别到设备吗？前提是你确定
设备的USB正常使用



在相机模型中我们可以得到以下公式：

$$\begin{cases} \frac{x}{f} = \frac{X_c}{Z_c} \\ \frac{y}{f} = \frac{Y_c}{Z_c} \end{cases} \Rightarrow \begin{cases} Z_c \cdot x = f \cdot X_c \\ Z_c \cdot y = f \cdot Y_c \end{cases}$$

公式5

解释：1、
同样我们用矩阵形式表示：

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

公式6

4、得到公式

而我们可以将以上公式综合一下就可以得到：

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{像素与像平面}} \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}}_{\text{相机与世界}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$
$$= L \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & l_3 & l_4 \\ l_5 & l_6 & l_7 & l_8 \\ l_9 & l_{10} & l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

因此，内参数矩阵可以表示为：

镜头畸变矫正

RaxKT: 非常感谢您的详细介绍,明白啦,另外在学习张正友校订算法,遇到边缘校正不理想的问题(在输出分辨率与输入...

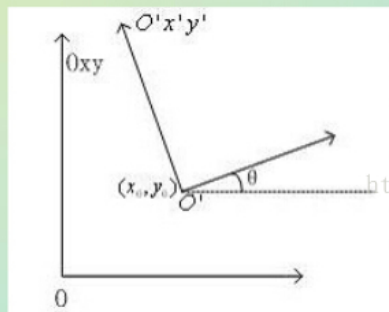
$$\begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix}$$

外参矩阵可以表示为： $\begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix}$ ，由旋转矩阵R和平移向量T组成

当然在好多资料上都有这种做法：

① 坐标变换



首先从二维坐标变换进行理解

$$\begin{cases} x = x' \cos \theta - y' \sin \theta + x_0 \\ y = x' \sin \theta + y' \cos \theta + y_0 \end{cases}$$

矩阵表示

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

齐次坐标

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & x_0 \\ \sin \theta & \cos \theta & y_0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$r_1^2 = r_2^2 = 1$$

$$r_1 \cdot r_2 = 0$$

图像数字化

O_1 在 u, v 中的坐标为 (u_0, v_0)
像素在轴上的物理尺寸为 dx, dy

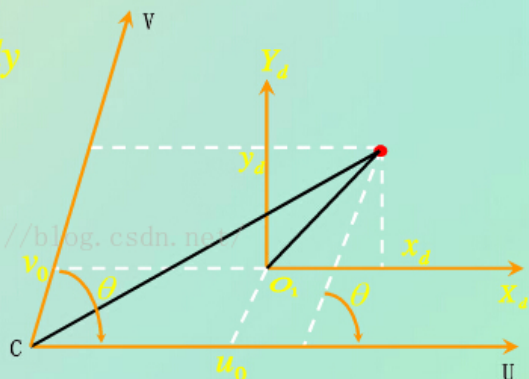
Affine Transformation :

$$u = u_0 + \frac{x_d}{dx} - \frac{y_d \cot \theta}{dx}$$

$$v = v_0 + \frac{y_d}{dy \sin \theta}$$

齐次坐标形式:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & -f_u \cot \theta & u_0 \\ 0 & f_v / \sin \theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \quad \text{其中} \quad f_u = \frac{1}{dx}, f_v = \frac{1}{dy}$$



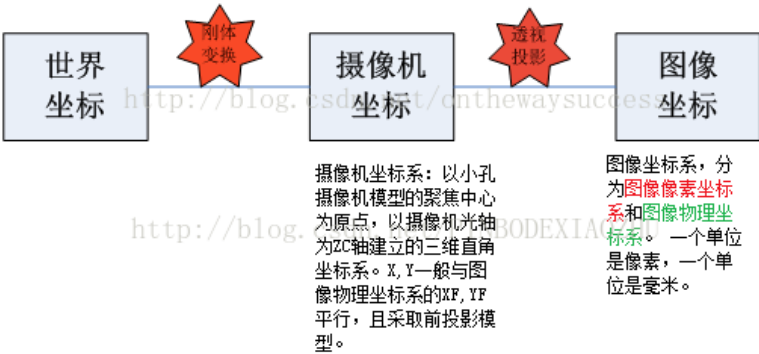
上图中表示的情况是像素坐标系和图像物理坐标系的两个坐标轴不是平行的关系，像素坐标系的两个坐标轴也不是垂直90°的关系，而图像物理坐标系的两个坐标轴是垂直关系。

所以，我们在转换两个坐标轴的坐标之间的关系时就必须考虑像素坐标系两个坐标轴之间的夹角了。就有了上面的不同的内参矩阵，理解了就好了。

二、图像坐标：我想和世界坐标谈谈(B)

玉米将在这篇博文中，对图像坐标与世界坐标的这场对话中涉及的第二个问题：谈话方式，进行总结。世界坐标是怎样变换进摄像机，投影成图像坐标的呢？

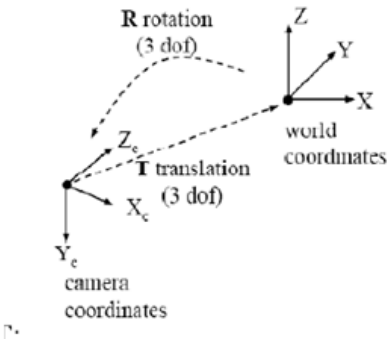
玉米做了一个简单的图示，在这里做一个提纲。图中显示，世界坐标系通过刚体变换到达摄像机坐标系，然后摄像机坐标系通过透视投影变换到达图像坐标系。可以看出，世界坐标与图像坐标的关系建立在刚体变换和透视投影变换的基础上。为了奖励刚体变和透视投影变换沟通了“世界上最远的距离”，玉米在图上奖励了他们两朵小红花。哈哈



首先，让我们来看一下刚体变换是如何将世界坐标系与图像坐标系联系起来的吧。这里，先对刚体变换做一个介绍：

刚体变换(rigidbody motion):三维空间中，当物体不发生形变时，对一个几何物体作旋转，平移的运动，称之为刚体变换。

因为世界坐标系和摄像机坐标都是右手坐标系，所以其不会发生形变。我们想把世界坐标系下的坐标转换到摄像机坐标下的坐标，如下图所示，可以通过刚体变换的方式。空间中一个坐标系，总可以通过刚体变换转换到另外一个坐标系的。转一转，走一走，就到另外一个坐标系下了。以前可能是面朝大海，经过平移旋转，最终可能只能面朝冰山了，哈哈



下面让我来看一下，二者之间刚体变化的数学表达。

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$
$$X_c = RX + T$$

其中，X_C代表摄像机坐标系，X代表世界坐标系。R代表旋转，T代表平移。R、T与摄像机无关，所以称这两个参数为摄像机的外参数(extrinsic parameter)可以理解两个坐标原点之间的距离，因其受x,y,z三个方向上的分量共同控制，所以其具有三个自由度。

R则为分别绕XYZ三轴旋转的效果之和。如下面所示：

绕X轴旋转 **r1**

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

绕Y轴旋转 **r2**

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

绕Z轴旋转 **r3**

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$R=r1*r2*r3$ 其由三个方向的 θ 控制,故具有三个自由度。

好了,刚体变换就讲完了。大家应该都了解,世界坐标系到摄像机坐标系之间的转换过程了吧。

接下来,让我们看看摄像机坐标下的坐标如何投影到图像坐标系下,最终变为照片中的一个像素。这其中包含两个过程:一是从摄像机坐标到“空间图像坐标”(x,y)所发生的透视投影;二是从“连续图像坐标”到“离散图像坐标”(u,v)。后者我们已经在第一篇博文中解释过。所以在这里,主要介绍一下透视投影。

透视投影(perspective projection): 用中心投影法将形体投射到投影面上,从而获得的一种较为接近视觉效果的单面投影图。有一点像皮影戏。它符合人们心理习惯,即离视点近的物体大,离视点远的物体小,不平行于成像平面的平行线会相交于消隐点(vanish point)。

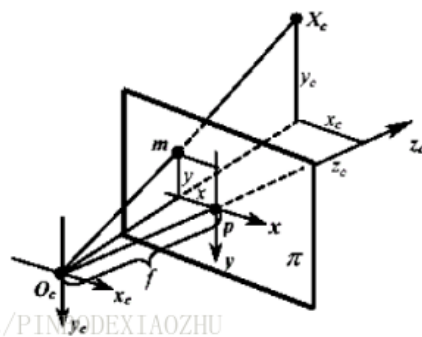
啰嗦这么多,其实大家看看示意图,看看公式,秒懂。

3.1 成像几何

基本模型

$$\begin{cases} x = \frac{fx_c}{z_c} \\ y = \frac{fy_c}{z_c} \end{cases}$$

$$z_c \mathbf{m} = \begin{pmatrix} fx_c \\ fy_c \\ z_c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbf{X}_c$$



O_c 是摄像机中(光)心

Z_c 是摄像机主(光)轴

$f=||O_c-p||$ 是摄像机焦距

以图中B(X_B, Y_B)点为例,在小孔成像摄像机模型下(几何分析的最常用模型)。这里的 f 为摄像机的焦距,其属于摄像机的内参数(intrinsic parameter)。其在成像平面上的投影点b(x_b, y_b)的坐标利用简单的相似三角形比例关系很容易求出:

$$x_b = \frac{f}{Z_B} X_B$$

$$y_b = \frac{f}{Z_B} Y_B$$

上面两式也阐明了摄像机坐标与图像坐标之间的透视投影关系。

好吧,现在玉米已经把图像坐标与世界坐标之间的这场对话所需经历的三个波折的过程加以了解释。即:刚体变换、透视投影、(x,y)换(u,v)(ps.这个在上一篇博文中讲过)。接下来玉米用一张图把三个过程连接起来。实现从世界坐标(X,Y,Z)到(u,v)之间的转换。让图像坐标与世界坐标直接对话。

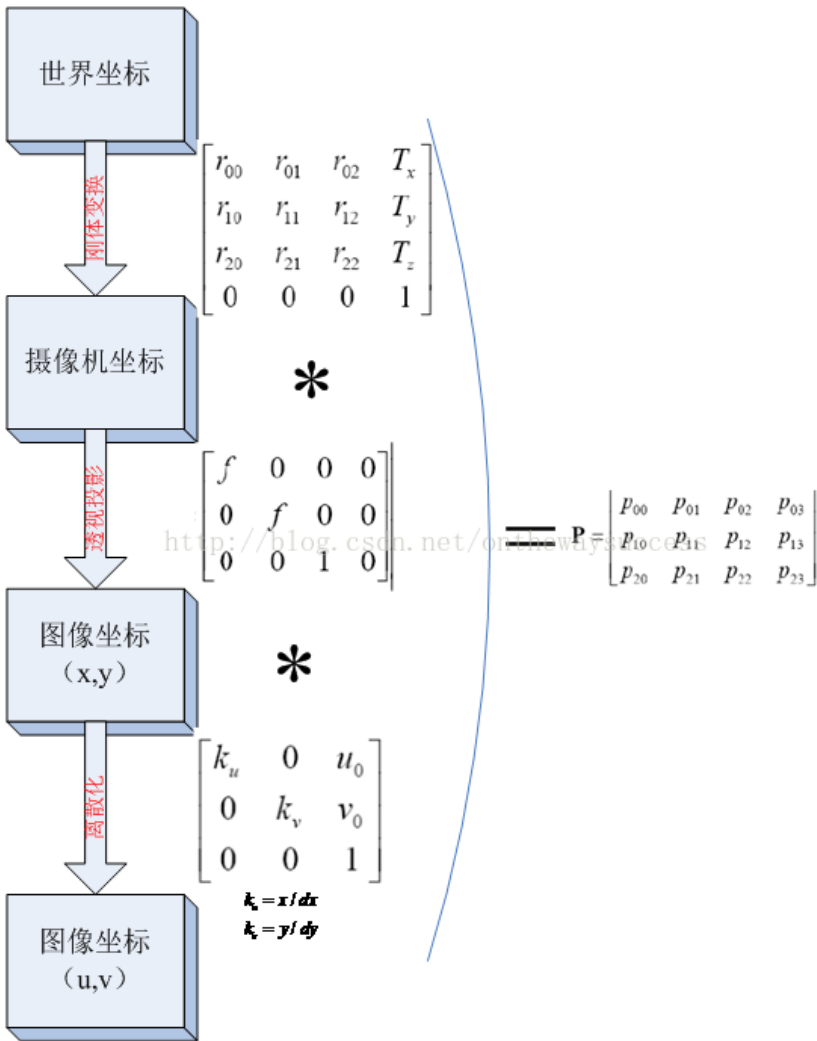
下图中的转换关系，都是用齐次坐标表达的，大家会发现这样的表达非常整洁。
其实这张图显示的过程还有一个名字：摄像机模型(camera model)。其实也就是摄像机的几何模型了。
将三者相乘，可以把这三个过程和在一起，写成一个矩阵：

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix}$$

P就是世界坐标到图像坐标的直接联系人，P就表示了一个投影相机，有下面公式：

$$\tilde{\mathbf{w}} = \mathbf{P}\tilde{\mathbf{X}}$$

注意在表示齐次坐标时，需要在符号上面加个小帽子。除去齐次坐标控制位P₂₃，P具有11个自由度。



摄像机模型及其中涉及的坐标系等，是弄清3D重建几何框架的基础。可以把它们视为基本运算关系。后面对于三维重建几何框架的推导，都是要用到三个基本坐标系和摄像机模型的。

我的同类文章

opencv (4)

• 张正友标定法的真实理解

2016-01-25

阅读 2329

• vs2010配置opencv2.4.8以及..

2016-01-24

阅读 426

• C#读取摄像头处理图片AFor...

2015-12-08


阅读 1208

• 在win7环境下安装Qt配置Op...

2015-01-24

阅读 375

参考知识库




算法与数据结构知识库


15420 关注 | 2320 收录


猜你在找


- 使用Cocos2d-x 开发3D游戏
- unity中屏幕和世界坐标系区别
- CityEngine城市建模视频教程（GIS思维邀你与3D大师的约会）
- Unity插件之NGUI学习8 Table和NGUI尺寸转换为世界坐标
- unity3D一游戏/AR/VR在线就业班 C#入门（二）
- 解决Unity鼠标坐标点转成世界坐标系坐标点
- 使用Cocos开发一款简单的3D VR抓钱游戏
- 在世界坐标系中放置物体SetTransform函数
- 用LayaAir引擎开发HTML5的3D与VR游戏（入门基础）【
- OpenGL立方体在世界坐标系中_缩放_旋转_平移_顶点片

查看评论

- 4楼 woshishazhu96 2017-04-03 14:07发表
- 

我也觉得讲的很好，起码我从零到看懂了大概，不过2楼说得对，前面希望补充，还有，后面的讲了旋转的R，然后就没有讲平移的T
- 3楼 woshishazhu96 2017-04-03 14:06发表
- 

我也觉得讲的很好，起码我从零到看懂了大概，不过2楼说得对，前面希望补充，还有，后面的讲了旋转的R，然后就没有讲平移的T。
- 2楼 阿木寺 2017-01-11 15:53发表
- 

关于坐标系，文章介绍了两种，内容很赞，但文中的第二篇介绍所涉及的图像坐标（x,y）->像素坐标（u,v）不是很详细，还望可以补充完整
- 1楼 dacheng_jsj 2016-10-20 14:15发表
- 

详细透彻，通俗易懂

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
- VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
- BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
- Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack
- FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
- Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
- Angular Cloud Foundry Redis Scala Django Bootstrap

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved 

□