

# notes for CV course from Tübingen University

## 一、Introduction (Andreas Geiger)

[lec\\_01\\_introduction.pdf](#)

### 1.1 Organization

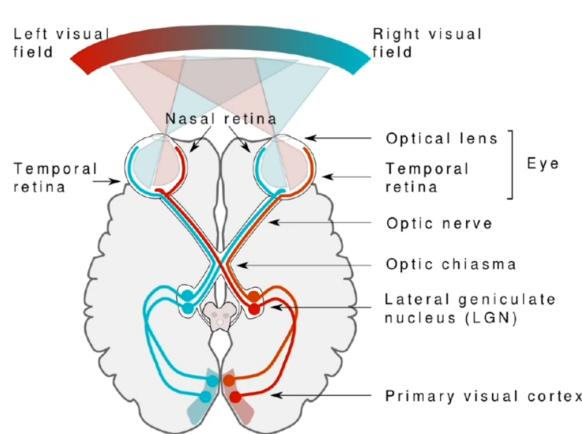
- what is CV?
  - 视觉是人类最出色的感知能力，我们尝试复制这种能力到机器中
  - 2D: detection, segmentation
  - 3D: reconstruct 3d from a collection of 2d images



### 1.2 Introduction

- what is AI?
  - "An attempt will be made to find how to make machines use **languages, form abstractions and concepts**, solve kinds of problems now reserved for humans, and **improve themselves.**"
    - ——from John McCarthy
      - ML
      - CV
      - CG
      - NLP
      - Robotics & Control
      - ...
- "Goal of Computer Vision is to convert light into meaning(geometric, semantic, ...)"

- 什么是“看见”？
  - “To now what is where by looking”
- 计算机视觉也是一门与生物医学（眼部视觉、神经学）交叉的学科

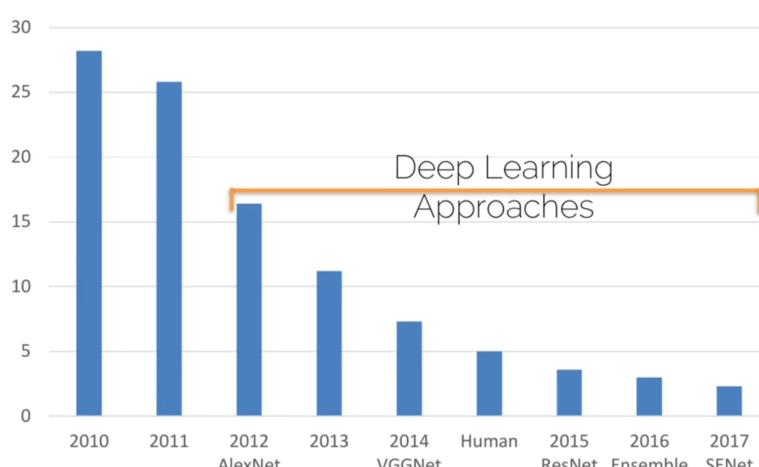


Over 50% of the processing in the **human brain** is dedicated to **visual information**.

- ■ 人脑中有超过 50% 的处理区域是关于视觉信息的
- 计算机视觉 VS 计算机图形学
  - CV 主要研究
    - 现实中的 3D 物体在投影为 2D 时已经损失了一个维度的数据，我们可以处理的只是一个二维矩阵
    - 我们想要还原现实场景中潜在的向量和 features
  - CG 主要关心
    - 数字化 3D 模型的物体、原材料、形状、几何、线条等；关心来源于该 3D 模型场景的 2D 图像

主要涉及，如何结合计算机图形学，利用现有的 2d 数据，还原本来的 3d 数据

- 计算机视觉 VS 图像处理
  -
- 计算机视觉 VS 机器学习
- 过去十几年提出的模型，错误率的排名

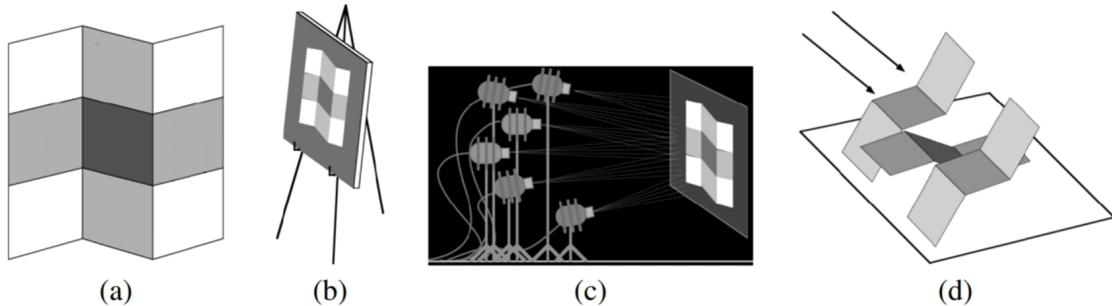


- 2015年提出的Resnet已经超过了人类的分类水准
- 计算机视觉VS人类视觉
  - 人类很容易学习获得视觉感知，因为人类可以在成长过程中，受到外界环境的影响（客观因素，主观教育），对于看到的图像进行分析
  - 机器所得到的只有一个数字矩阵，它没有任何可以依据的经验去解释所“看到”的图像



200	133	110	103	117	90	47	30	32	79	66	65
197	122	123	138	98	100	46	45	22	11	43	55
140	116	165	159	90	56	58	47	26	13	54	102
132	148	119	108	123	57	64	46	21	22	79	94
126	121	80	143	101	55	61	38	20	21	81	65
50	71	74	63	52	39	41	39	32	26	97	66
51	59	62	44	40	40	36	26	27	31	29	44
59	62	70	50	48	35	34	35	26	21	24	32
49	59	65	64	58	34	40	28	26	21	23	124
39	45	47	64	54	34	40	24	19	47	133	207
37	42	39	38	39	50	75	74	105	170	197	167
37	47	33	35	50	108	162	184	184	157	125	112
45	48	35	37	75	148	183	156	83	91	91	116
49	48	54	50	75	158	110	66	74	128	155	149
48	51	57	50	65	91	79	92	101	105	132	132
51	58	66	55	58	52	91	91	88	115	158	174
57	60	61	52	56	61	60	55	92	146	188	190
65	50	54	56	57	51	54	56	80	115	177	187
67	40	40	61	65	48	39	30	36	75	151	181
53	32	36	35	61	43	37	26	29	35	126	189
29	42	107	20	28	41	40	26	30	36	113	200
30	21	32	24	34	37	33	23	25	39	105	171
32	28	19	23	29	36	47	69	132	169	183	128
31	25	62	54	47	44	61	190	227	231	206	155
44	66	99	72	67	63	68	128	127	115	109	157
53	47	47	41	29	32	25	20	41	81	89	175
38	44	61	73	54	48	37	87	90	111	126	169
39	41	63	97	66	91	74	134	131	153	143	185
42	56	96	102	112	111	94	137	121	141	146	181
94	114	114	114	122	113	77	117	117	154	149	169
157	176	116	121	130	139	103	161	148	180	145	125
143	178	182	178	139	153	129	168	175	187	170	152
127	183	203	197	153	164	143	180	195	182	165	211
68	107	127	125	101	107	100	123	149	166	167	215

- The perception of shading and reflectance (adelsonPerceptionShadingReflectance1996)



- 为了解释一个二维图像的实际情况，包括反射、光线和形状，画家、灯光设计家、雕塑家用三种不同的方式将其还原
- 实际上，有无数种还原方法，这就是为什么CV解释二维图像非常困难

- Ames Room Illusion



- ■ 在不同的位置，同样的东西看起来大小不同

- Perspective Illusion



- ■ 上图中的小球似乎在违背我们认知的物理效应移动，但是实际上，这只是不同角度的图像产生的错觉  
■ 人们的大脑非常善于构建立体几何图形，会自动根据自己的认知还原实际场景的三维结构；但这并非总是准确的
- 

- Challenges

1. Viewpoint variation

- 对于同一个物体，来自不同视点的二维图像，像素矩阵可能会发生巨大的变化

2. Deformation

- 对于非刚性的对象，那么发生变形前后的二维图像，像素矩阵也会发生巨大变化

3. Occlusion

- 遮挡在日常生活中的二维图像中也是十分常见的，这对于像素矩阵的分辨也增加了困难

4. Illumination

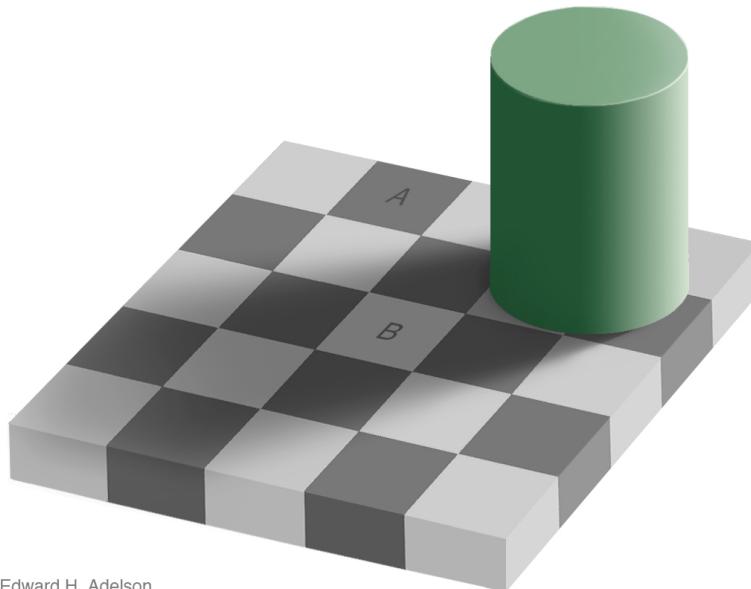
- 同一个视点，同样的场景，改变光源强弱、光源位置，也会使像素矩阵产生很大的变化

5. Motion

- 动作产生了模糊和画面变化等问题，但同样也是我们感兴趣的还原对象

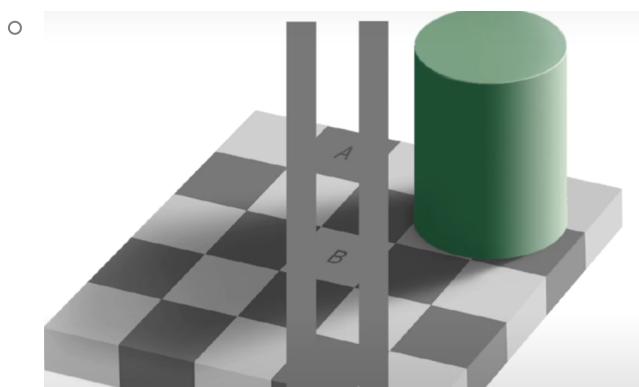
6. Perception VS Measurement

- [Source](#)



Edward H. Adelson

- 对于这张图，大多数人的第一反应是 A 方块会比 B 方块颜色更深，但是实际上如果用相似颜色的色条比较，会发现其实两个方块的颜色是完全相同的



- 因此，感知的结果和实际测量的结果也有可能不一样

## 7. Local Ambiguities

- 单从图像的局部来看，无法断定它究竟是什么东西，必须要联系上下文
- 因此，将场景整合在一起对于 CV 非常重要

## 8. Intra Class Variation

- 同一类东西，可能会有成千上万个不同的种类，但是它们都应该分为一类，因此同一类别内物体的变化识别也十分困难

## 1.3 History of CV

**Svetlana Lazebnik (UIUC):** Computer Vision: Looking Back to Look Forward

► <https://slazebni.cs.illinois.edu/spring20/>

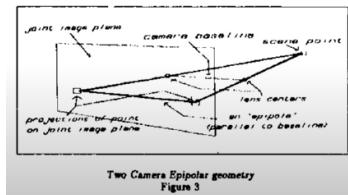
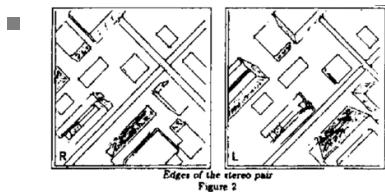
**Steven Seitz (Univ. of Washington):** 3D Computer Vision: Past, Present, and Future

► <http://www.youtube.com/watch?v=kyIzMr917Rc>

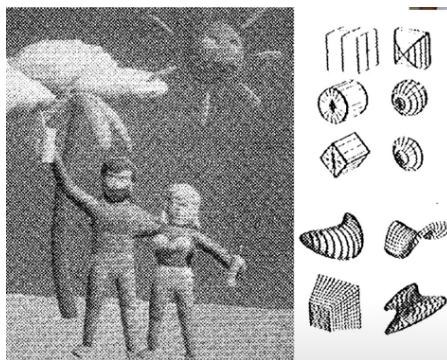
► <http://www.cs.washington.edu/homes/seitz/talks/3Dhistory.pdf>

- Pre-History
  - 1510: Perspectograph
    - 用一块非常透明的玻璃，与目标物体保持一个合适的距离，然后在玻璃上就会看到目标物体的投影，此时用画笔一点点画下来，就可以得到目标物体的投影
  - 1839: Daguerreotype
    - 用一些感光的化学物质作为底片，然后将其放在一个小黑箱中，只留下一个小孔，用机械装置控制小孔开关，在一定的曝光操作后，底片会显示出目标物体的投影
- Wave of development
  - 1960-1970: 用方块、线条和模型来拟合真实世界
  - 1970-1981: 低层次视觉：立体、流动、形状、阴影
  - 1985-1988: 神经网络、反向传播、自动驾驶模型
  - 1990-2000: 密集例题和多视点立体、MRFs
  - 2000-2010: 特征、描述符、大尺度的运动结构重建
  - 2010-now: 深度学习、大数据集、快速增长、商业化
- Some Brief History
  - 1957: Stereo
    - Gilbert Hobrough 演示了立体图像的模拟实现
    - 用于制作高程地图（摄影测量学），从二维图像还原原始地貌的真实高度
    - Raytheon-Wild B8 Stereomat 的诞生
  - 1958-1962: Rosenblatt's Perceptron
    - 第一个提出算法并实现了训练单个线性阈值神经元
    - 感知器的优化： $L(w) = - \sum_{n \in M} w^T x_n y_n$
    - 用于简单的二分类任务
  - 1963: Larry Roberts' Blocks World
    - 从二维图像中提取线条
    - 通过线条的拓扑结构来还原三维场景
    - 从二维图像中提取三维场景，而不采用二维图像的模式识别
    - 但是只用于简单（理想状态）的场景：
  - 1969: Minsky and Papert publish book——<Perceptrons>
    - 几个令人沮丧的结果
    - 表明单层感知器不能解决一些非常简单的问题（异或问题、计数）
    - 符号人工智能研究在 70 年代占据主导地位
    - 神经网络研究的衰退
  - 1970: MIT Copy Demo
    - 开发了一个“闭环”机器人系统
    - 系统涵盖了感知、规划和影响被感知物体的能力
    - Copy Demo 的任务是通过图像分析来构建一个由简单多面体积木组成的结构，这些积木的物理关系需要被确定下来，以制定拆卸和装配计划

- 这个项目对于实验环境具有非常高的敏感性，鲁棒性很差，当时被认为如果解决低水平视觉，这个问题也可以得到充分地解决——然而以今天的视角来看，无论是解决低水平视觉问题还是项目问题，都远不止这么简单
- 然而，它展示了视觉感知、规划和执行的能力过程
- 1970: Shape from Shading
  - 从简单的二维图像中还原三维
  - 假设 Lambertian 表面和恒定的 albedo
    - Lambertian 表面是一种理想化的表面模型，它具有完全漫反射的特性，无论从哪个角度观察，它都会均匀地反射光线（可以考虑为理想化的光照）
    - 物体表面的漫反射特性，从表面反射的光线占入射光的比例（实际上是反映表面的明亮度）
  - 应用平滑正则化来约束
- 1978: Intrinsic Images
  - 
  - 将图像分解为不同的内在二维层，例如反射率、阴影、形状、运动
  - 对下游任务有用，例如独立于阴影和照明的物体检测
- 1980: Photometric Stereo
  - 从多个 2D 图像中还原 3D 图像，以相同的 viewpoint 在不同的 illumination 条件下拍摄
  - 至少需要 3 张图片
  - 前所未有的细节和准确性
  - Lambertian assumption has been relaxed subsequently (这种方法也适用于非 Lambertian 表面)
- 1981: Essential Matrix
  - 
  - 将一种双视图几何定义为矩阵映射到 epipolar lines (极线)
  - 将对应像素匹配简化为一维问题
  - 可以从一组二维对应关系中估计出来
  - 之后一百年中的关键思想
- 1981: Binocular Scanline Stereo

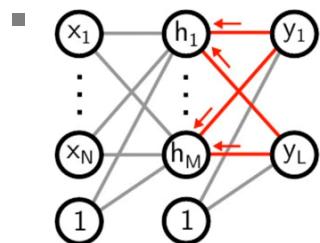


- 沿着极线关联所有的点
- 利用动态规划沿着扫描线（图像行）引入约束
- 这可以克服立体匹配过程中的模糊情况
- 但是由于所有的扫描线都是单独处理的，因此会出现扫描线间的伪影
- 1981: Dense Optical Flow
  - 
  - 视觉场景中物体、表面和边缘的运动
  - 通过密切地追踪每两帧之间的像素运动距离来测量
- 1984: Markov Random Fields (MRFs)
  - Markov 性质
    - 一个随机变量序列按时间先后关系依次排开的时候，第  $N + 1$  时刻的分布特性，与  $N$  时刻以前的随机变量取值无关（也就是说，未来的状态只与当前状态有关，与过去状态无关）
  - 随机场 ([马尔可夫随机场模型](#))
    - 随机场是一种描述多个位置上随机变量的随机模型。在 MRF 中，我们给每个位置赋予相空间的一个值，这些值构成了一个随机场
    - 例如，如果一块地里种的庄稼的种类仅与它邻近的地里种的庄稼地种类有关，而与其它地方的庄稼种类无关，那么这些庄稼地集合就构成了一个马尔科夫随机场
  - 对于先验概率的编码
  - 利用像素之间的相关性，实现图像分割、图像去噪、目标识别等
- 1980: Part-based Models



- 构建多种模块化的立体几何图形，组合生成复杂的立体三维图形

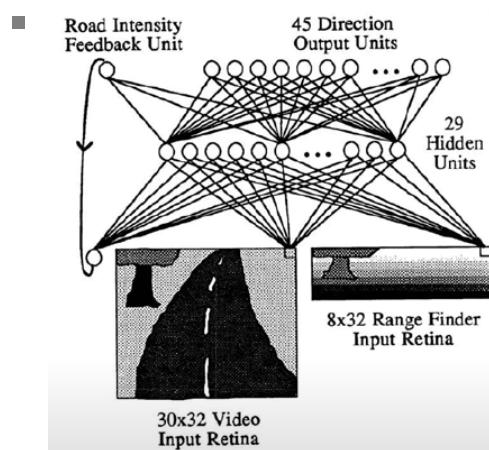
- 1986: Backpropagation Algorithm



- 深度网络中梯度的有效计算，网络权值
- 使基于梯度的学习应用于深度网络
- 1961年提出，但第一次证实是在1986年
- 至今仍然发挥着非常重要的作用

- 1986: Self-Driving Car VaMoRs

- 1988: Self-Driving Car ALVINN

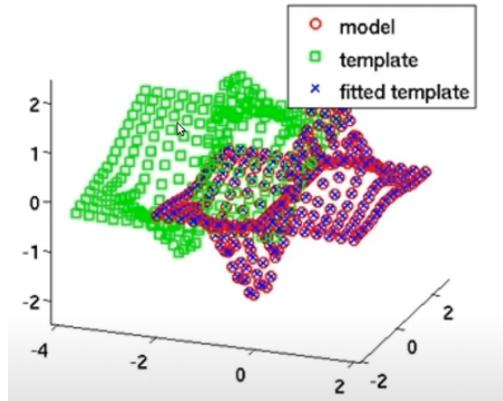


- 第一辆使用神经网络实现的自动驾驶算法模型
- 前瞻性的、基于视觉的自动驾驶
- 完全连接的神经网络地图道路图像
- 训练模拟道路图像
- 以每小时 70 英里的速度连续行驶 90 英里

- 1992: Structure from Motion

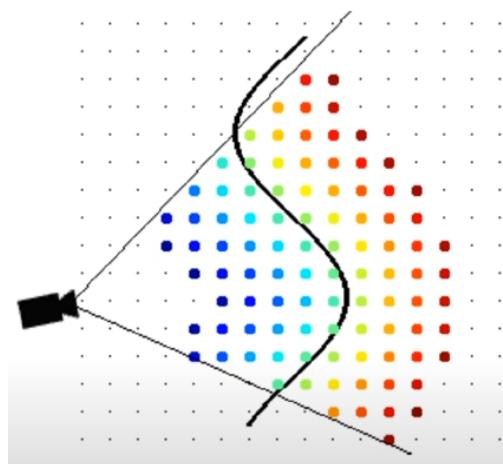
- 从静态场景的二维图像序列估计三维的结构
- 只需要 1 个摄像头

- 1992: Iterative Closest Points



- 通过迭代优化（刚性或非刚性）转换为两个点云
- 用于聚合来自不同扫描的部分 2D 或 3D 表面

- 1996: Volumetric Fusion



- 通过平均有符号距离值对多个隐式表示的表面进行聚合
- 网格提取作为后处理

- 1998: Multi-View Stereo

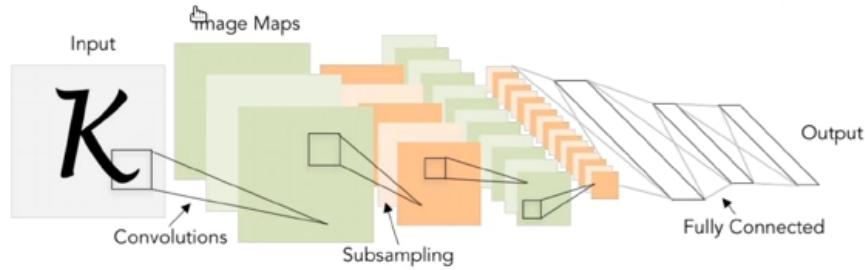
- 多视角三维重建，使用水平集方法输入图像
- 重建与图像匹配
- 适当的能见度
- 灵活的拓扑结构

- 1998: Stereo with Graph Cuts



- 流行的 MRF MAP 推理算法
- 第 1 个版本包括一元项和成对项
- 后来的版本还包括特定形式的高阶电位全局推理与扫描线立体

- 1998: Convolutional Neural Networks

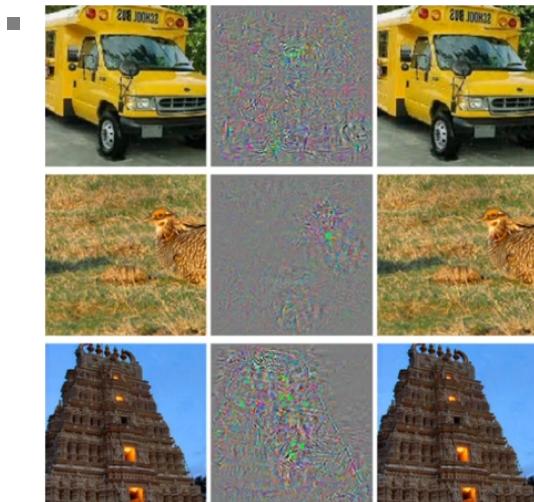


- 类似于 Neocognitron，但使用反向传播进行端到端训练
- 通过卷积和最大池化实现空间不变性
- 权重共享减少了参数
- Tanh/Softmax 激活
- 目前只在 MNIST 上取得了好的结果
- 1999: Morphable Models
  - 单视图 3D 面部重建
  - 200 激光面部扫描的线性结合
- 1999: SIFT
  - Scale Invariant Feature Transform (尺度不变特征转换)
  - 图像中显著局部特征的检测和描述
  - 为很多应用提供了基础
- 2006: Photo Tourism
  - 从互联网图片大规模 3D 重建
  - 关键组件: SIFT 特征匹配, 束调整
  - 微软 Photosynth(discount)
- 2007: PMVS

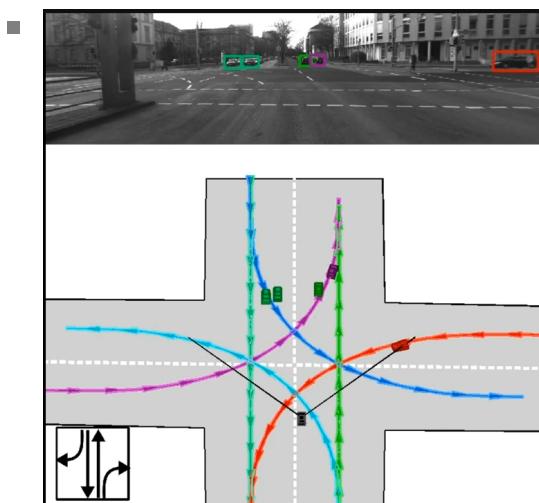


- Patch-based Multi View Stereo
- 基于补丁的多视图立体
- 各种大小物体的鲁棒重建
- 三维重建技术的性能不断提高
- 2009: Building Rome in a Day
  - 
  - 从非结构化的互联网图片集三维重建出地标建筑和城市
  - 将近 75000 篇图像，虽然是稀疏还原，但是细节精确度相当高
- 2011: Kinect
  - 主动式光三维传感
  - 利用机器学习的三维姿势评估
  - (人类体感交互摄像机)
- 2009-2012: ImageNet and AlexNet
  - ImageNet
    - 识别基准 (ILSVRC)
    - 1000 万张注释图片
    - 1000 个类别
  - AlexNet
    - 第一个赢得 ILSVRC 测试的神经网络，通过 GPU 训练、深度模型、数据集
- 2002-now: Golden Age of Datasets
  - Middlebury Stereo and Flow
  - KITTI, Cityscapes: Self-driving
  - PASCAL, MS COCO: Recognition
  - ShapeNet, ScanNet: 3D DL
  - Visual Genome: Vision/Language
  - MITOS: Breast cancer

- 2012-now: Synthetic Data
  - 标注真实的数据集是很费时费力的
  - 因此，合成数据集的研究激增
  - 创建 3D 资产的成本也很高
  - 但即使是非常简单的 3D 数据集也被证明对预训练非常有用
- 2014: Visualization
  - 提供对网络（黑箱）所学到的知识的见解
  - 最强烈地激活网络不同层的各种神经元的可视化图像区域
  - 发现更高的层次捕获更多抽象的语义信息
- 2014: Adversarial Examples



- 精确的图像分类器可能会被难以察觉的变化所欺骗
- 上图中所有图像都被归类为鸵鸟
- 2014: Generative Adversarial Networks
  - 深度生成模型 (VAEs、GANs) 产生了引人注目的图像
  - StyleGAN2 是最先进的
  - 结果显示，人脸与真实图像难以区分
- 2014: DeepFace
  - 基于模型的对其与深度学习在人脸识别中的结合
  - 第一个能达到人类水平的人脸识别性能的模型
- 2014: 3D Scene Understanding

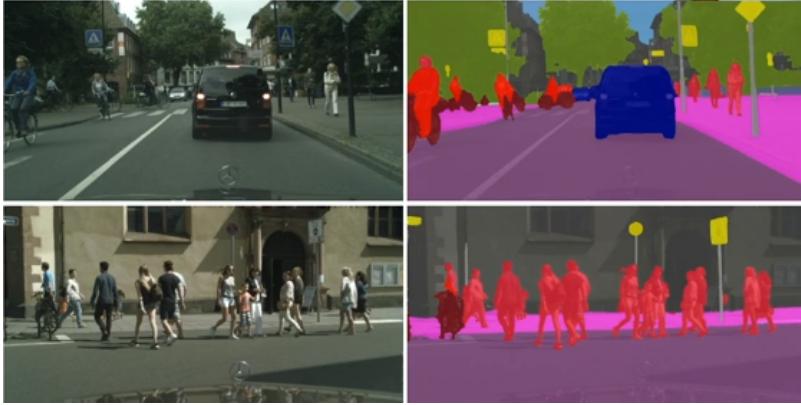


- 解析 RGB 与 RGBD 图像为整体 3D 场景表示

- 2014: 3D Scanning
  - 通过随机探索和奖励信号 (如游戏分数等) 学习策略 (状态 -> 行动)
  - 无其它监督
  - 在许多雅达利游戏中获得成功
  - 但有些游戏仍然很难
- 2016: Style Transfer
  - 

- 利用 ImageNet 预训练的深度网络, 从风格中分离出内容

- [体验网站](#)

- 2015-2017: Semantic Segmentation
  - 

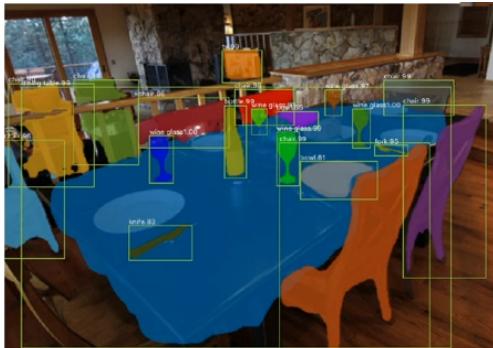
- 为每个像素分配语义分类

- 语义分割开始在具有挑战性的现实世界数据集 (例如 Cityscape) 上工作

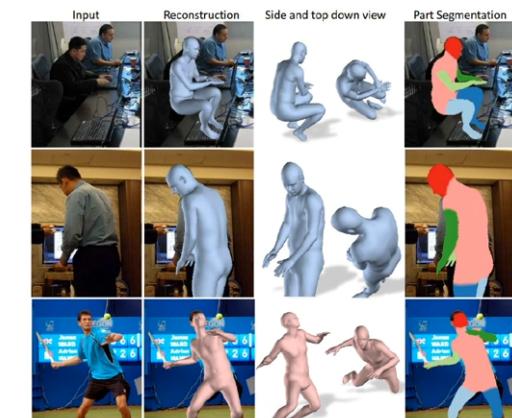
- 2015 年: FCN、SegNet

- 2016 年: DeepLab、FSO

- 2017 年: DeepLabv3

- 2017: Mask R-CNN
  - 

- 关节对象的深度神经网络检测和实例分割
- 输出“结构化对象”，而不仅仅是一个数字（类标签）
- 2017: Image Captioning
  - 将视觉与语言结合起来的兴趣与日俱增
  - 出现了几个新的任务：图片说明，视觉问答
  - 然而，模型仍然缺乏理解常识
- 2018: Human Shape and Pose



- 人体姿势、形状模型成熟
- 丰富的参数模型 (SMPL、STAR)
- 仅从 RGB 图像回归模型的姿势依赖的变形和服装
- 2016-2020: 3D Deep Learning
  - 应用体素、点云、网格作为表示
  - 预测 3D 模型，甚至从单独的 1 张图片

- 从过去到现在，计算机视觉的研究成果对我们生活的改变



- 目前仍然有的挑战
  - Un-/Self-Supervised Learning
    - 由于获得大型标注数据集还是十分困难，因此致力于研究无监督学习
  - Interactive Learning
    - 交互式学习算法
  - Accuracy (for self-driving)
  - Robustness and generalization
  - Inductive biases
    - 归纳偏差

- Understanding and mathematics
  - 对于理解黑盒网络非常重要
- Memory and compute
- Ethics and legal questions
  - 合法化和商业化等问题

## 二、Image Formation

[lec\\_02\\_image\\_formation.pdf](#)

### 2.1 Primitives and Transformations

- Primitives
  - points, lines, planes
- Transformations
  - some the most basic transformations
- Multiple View Geometry in CV

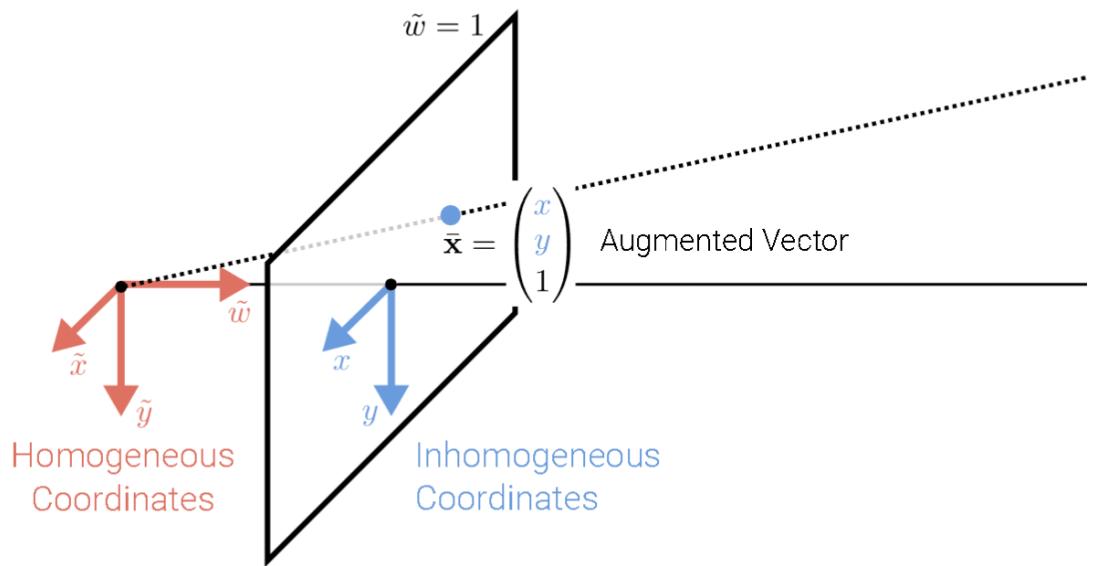
#### 2.1.1 Primitives

##### 2D points

齐次坐标和非齐次坐标的理解小结

[Inhomogeneous vs homogeneous coordinates](#)

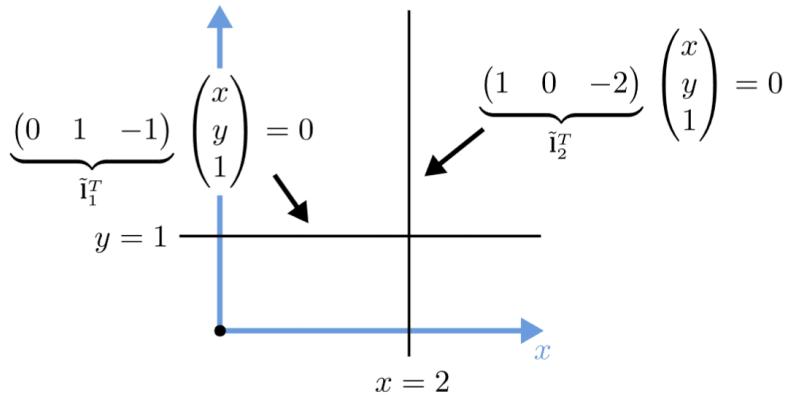
- 2d points
  - 非齐次坐标 (Inhomogeneous coordinates)
    - $x = \begin{pmatrix} x \\ y \end{pmatrix} \in R^2$
  - 齐次坐标 (homogeneous coordinates)
    - $\tilde{x} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in P^2$ 
      - 这里的  $P^2 = R^3 / \{(0, 0, 0)\}$ , 称为射影空间
      - 引入一个新的维度  $\tilde{w}$ , 使得某些几何变换可以用矩阵乘法简洁表达
      -
    - 一个非齐次坐标可以用若干等价的齐次坐标表示, 其中当  $\tilde{w} = 1$  时, 我们将这时的齐次坐标记作对应的增广向量 (augmented vector)  $\bar{x}$ 
      - $\tilde{x} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \bar{x}$
    - $\tilde{w} = 0$  时, 齐次坐标没有对应的非齐次坐标, 此时的点称为理想点 (ideal points) 或者无穷点
    - 齐次向量、非齐次向量、增广向量的映射关系 (这里的  $z$  轴实际上是  $\tilde{w}$  轴)



- 其实由图中可以理解，任何一个  $\tilde{w}$  维度下的  $x, y$  平面上表示的点其实都可以相互映射，一一对应

## 2D lines

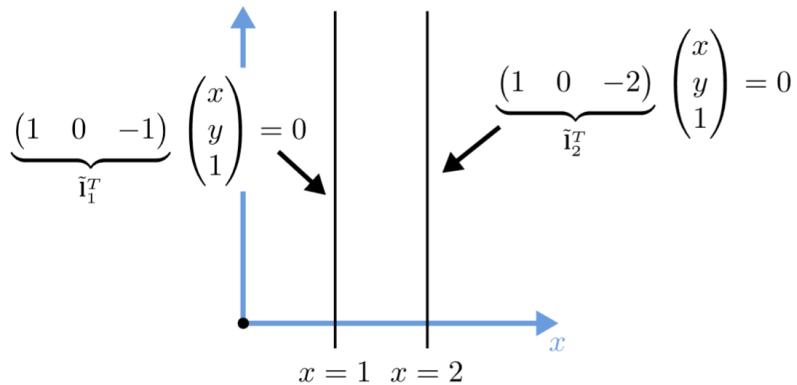
- 2d lines 也可以用齐次坐标  $\tilde{l} = (a, b, c)^T$  表示
  - $\{\bar{x} | \tilde{l}^T \bar{x} = 0\} \Leftrightarrow \{x, y | ax + by + c = 0\}$ 
    - 这里的  $\tilde{l}$  向量可以标准化为  $\tilde{l} = (n_x, n_y, d)^T = (n, d)^T$ , 其中  $\|n\|_2 = 1$ , 此时  $n = (n_x, n_y)$  向量是垂直于直线的法向量,  $d$  则是直线到原点的距离
    - 对于  $\tilde{l}_\infty = (0, 0, 1)^T$ , 是无穷远处的线, 它会穿过所有理想点, 且不能被标准化
- Cross product (叉乘) (外积) (向量积)
  - $a \times b = [a]_\times b = \begin{bmatrix} 0, -a_3, a_2 \\ a_3, 0, -a_1 \\ -a_2, a_1, 0 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$
  - 叉乘得到一个新的向量, 这个向量的方向垂直于原来两个向量的平面, 遵循右手法则; 长度等于原来两个向量的点乘 (内积) (数量积)
    - 注意
      - 向量  $a$  和  $b$  的叉乘, 等于  $[a]_\times$  和  $b$  的点乘, 其中  $[a]_\times$  是  $a$  的一个特殊的斜对角矩阵形式, 如上所示
- 2d lines arithmetic
  - 叉乘有什么用呢? 在齐次坐标表示下
    - 两条线的交点可以表示为:  $\tilde{x} = \tilde{l}_1 \times \tilde{l}_2 \setminus$
    - 经过两个点的线 (the line joining two points) 可以表示为:  $\tilde{l} = \bar{x}_1 \times \bar{x}_2$ 
      - 当然, 一般情况下我们使用 augmented vector 表示, 也可以用所有等价的齐次坐标表示



- 根据前面所述，2d lines 可以表示为一个 3 维向量  $\tilde{l}$  与 augmented vector 点乘的形式，因此  $y = 1$  和  $x = 2$  两条直线都可以拆分成这样的形式，其中  $\tilde{l}_1$  和  $\tilde{l}_2$  的叉乘结果如下

$$\circ \quad \tilde{l}_1 \times \tilde{l}_2 = \begin{pmatrix} (1 \times -2) - (-1 \times 0) \\ (-1 \times 1) - (0 \times -2) \\ (0 \times 0) - (1 \times 1) \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

- 因此，得到了图中交点的齐次坐标



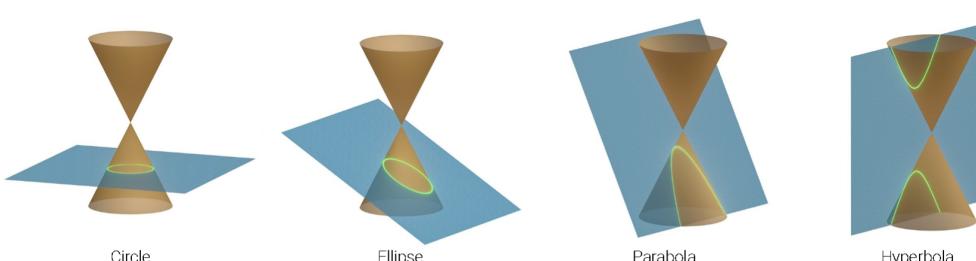
- 同理，我们也可以证明第二个等式

$$\circ \quad \text{图中 } \tilde{l}_1 \text{ 和 } \tilde{l}_2 \text{ 的叉乘结果为 } \tilde{l}_1 \times \tilde{l}_2 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$$

- 因此，两条平行线确实在无穷远处的点上相交

## 2D conics

- 实际上，2 维空间的一切几何对象都是基于点来描述的，线、面，其实是由一组具有特殊关系的点组成的集合；因此，我们只需要描述这种特殊关系，就可以描述任意类型的几何对象
- 更复杂的几何对象可以用多项式齐次方程来表示，例如圆锥截面可以用下面的二次齐次方程表示
  - $\{\bar{x}|\bar{x}^T Q \bar{x} = 0\}$



- (圆、椭圆、抛物线、双曲线)
  - 我们只需要调整Q矩阵就可以得到各种类型的圆锥截面
- 复杂的2维几何对象的齐次表示并不是本课程的重点，但却是多视角几何和相机校准的关键技术，如果感兴趣可以阅览教授的《Hartley and Zisserman》

## 3D points

- 3d points
  - 非齐次形式
    - $x = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in R^3$
  - 齐次形式
    - $\tilde{x} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} \in P^3$
    - 其中  $P^3 = R^4 / \{(0, 0, 0, 0)\}$
  - 非齐次与齐次的转换与 2d 的情况一致

## 3D planes

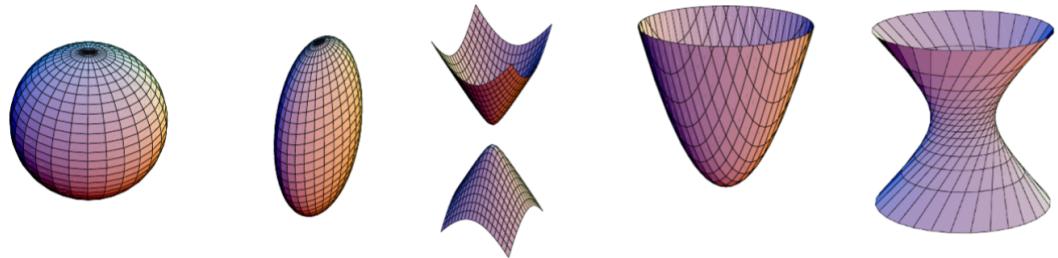
- 3d planes
  - $\tilde{m} = (a, b, c, d)^T$ 
    - $\{\bar{x} | \tilde{m}^T \bar{x} = 0\} \Leftrightarrow \{x, y, z | ax + by + cz + d = 0\}$
  - 同样的，我们也可以将  $\tilde{m}$  标准化，也即  $\tilde{m} = (n_x, n_y, n_z, d)^T = (n, d)^T$ ，其中  $\|n\|_2 = 1$ ；此时向量  $n$  就是平面的法平面向量， $d$  就是该平面与原点的距离
  - 对于  $\tilde{m}_\infty = (0, 0, 0, 1)^T$ ，就是无穷远处的平面，它经过所有的理想点，且不能被标准化

## 3D lines

- 3d lines 不如 2d lines 或者 3d planes 一样优雅，一种可能的表达方式维直线上两个点 p、q 的线性组合：
  - $\{x | x = (1 - \lambda)\vec{p} + \lambda\vec{q} \cap \lambda \in R\}$
- 然而，这种表示方式用 6 个参数来表示 4 个自由度，并不是一种理想的表示方法
- 可选的一种最小表示方法是将两个平面参数化，或者 Plücker 坐标
  - 详见第二章 2.1

## 3D quadrics

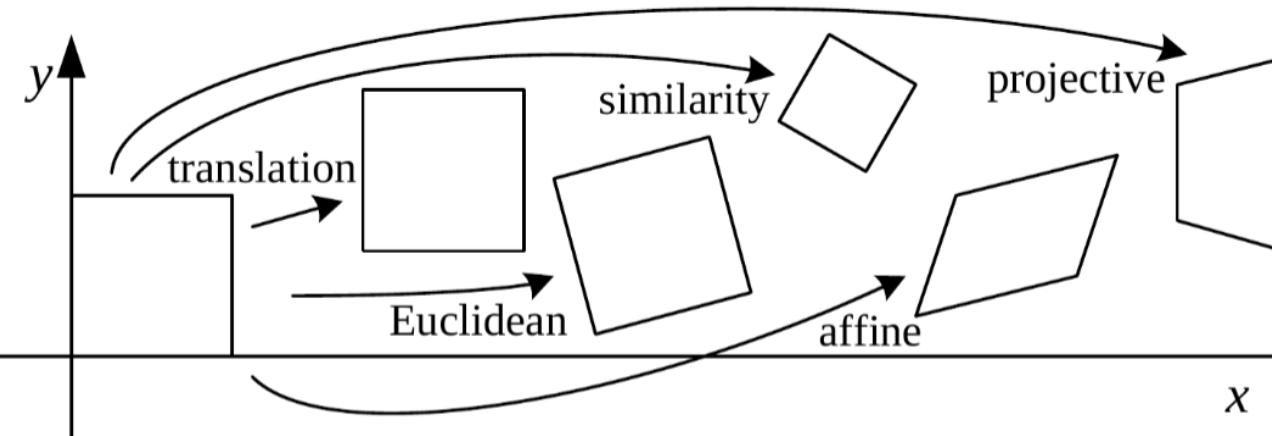
- 2d 的 conics 的 3d 模拟就是
  - $\{\bar{x} | \bar{x}^T Q \bar{x} = 0\}$



- 同样的，在这里也不展开叙述

## 2.1.2 Transformations

### 2d transformations



- **Translation** (2DoF)

$$- x' = x + t \Leftrightarrow \bar{x}' = \begin{bmatrix} \vec{I}, t \\ \vec{0}^T, 1 \end{bmatrix} \bar{x}$$

- 使用齐次表示，允许链式、逆变换

- 增广向量  $\bar{x}$  总是可以用一般的齐次向量  $\tilde{x}$  代替

- **Euclidean** (2d Translation + 2d Rotation, 3DoF)

$$- x' = Rx + t \Leftrightarrow \bar{x}' = \begin{bmatrix} \vec{R}, t \\ \vec{0}^T, 1 \end{bmatrix} \bar{x}$$

-  $\vec{R} \in SO(2)$  是一个标准正交旋转矩阵，其中  $RR^T = I$ 、 $\det(R) = 1$

- Euclidean 保留了点之间的欧几里得距离

- Translation 是一种特殊的 Euclidean

- **Similarity** (2d translation + scaled 2d rotation, 4DoF)

$$- x' = sRx + t \Leftrightarrow \bar{x}' = \begin{bmatrix} s\vec{R}, \vec{t} \\ \vec{0}^T, 1 \end{bmatrix} \bar{x}$$

-  $\vec{R} \in SO(2)$  是一个旋转矩阵， $s$  是一个任意缩放因子

- Similarity 只保留了线之间的夹角

- **Affine** (2d Linear transformation, 6DoF)

$$- x' = Ax + t \Leftrightarrow \bar{x}' = \begin{bmatrix} \vec{A}, \vec{t} \\ \vec{0}^T, 1 \end{bmatrix} \bar{x}$$

-  $\vec{A} \in R^{2 \times 2}$  是一个任意的  $2 \times 2$  矩阵

- Affine 只保留了平行线间的平行性
- Perspective (homography, 8DoF)
- $\tilde{x}' = \tilde{H}\tilde{x}$  ( $\bar{x} = \frac{1}{\tilde{w}}\tilde{x}$ )
- $\tilde{H} \in R^{3 \times 3}$  是一个任意的齐次  $3 \times 3$  矩阵
- Perspective 只保留了线段还是直的

DoF: Degrees of Freedom, 自由度

- 2d transformations on Co-vectors
  - 考虑最一般的变换 perspective:
    - $\tilde{x}' = \tilde{H}\tilde{x}$
    - 那么 2d lines 的变换如下
      - $\tilde{l}'\tilde{x}' = \tilde{l}'^T \tilde{H}\tilde{x} = (\tilde{H}^T \tilde{l}')^T \tilde{x} = \tilde{l}'^T \tilde{x} = 0$
    - 因此, 我们得到了
      - $\tilde{l}' = \tilde{H}^{-T}\tilde{l}$
  - 综上, 我们可以看出, 在类似于 2d line 或者 3d normal 的协向量的变换, 其变换矩阵可以表示为点变换矩阵的转置逆的变换矩阵

## summary

- Overview of 2d transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$[\mathbf{I} \quad \mathbf{t}]_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$[\mathbf{R} \quad \mathbf{t}]_{2 \times 3}$	3	lengths	
similarity	$[s\mathbf{R} \quad \mathbf{t}]_{2 \times 3}$	4	angles	
affine	$[\mathbf{A}]_{2 \times 3}$	6	parallelism	
projective	$[\tilde{\mathbf{H}}]_{3 \times 3}$	8	straight lines	

- 这些变换计算在组合、逆运算下是封闭的
- Overview of 3d transformations

○ Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	6	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	7	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{3 \times 4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{4 \times 4}$	15	straight lines	

- Direct Linear Transform for Homography Estimation (单应性估计)

-  $\chi = \{\tilde{x}_i, \tilde{x}'_i\}_{i=1}^N$  定义一组 2d 点 to 2d 点 的关系，通过一个 Homography (单应性) 矩阵  $\tilde{H}$ ，定义为  $\tilde{x}'_i = \tilde{H}\tilde{x}_i$

- 这些对应的向量是齐次的，它们的方向一致，但是大小不同

- 因此，上式也可以表示为  $\tilde{x}'_i \times \tilde{H}\tilde{x}_i = \vec{0}$

- 假如我们已经有了  $\tilde{x}$  和  $\tilde{x}'$ ，如何估计这里的单应性矩阵  $H$  呢？

- 如果用  $\tilde{h}_k^T$  表示  $\tilde{H}$  矩阵的第 k 行，则上式可以重新书写为下式

- \$\$

$$\begin{aligned} A[i]\tilde{h} = & \begin{bmatrix} \vec{0}^T & \tilde{x}_1^T & \tilde{x}_2^T & \dots & \tilde{x}_N^T \end{bmatrix} \\ & \begin{bmatrix} \tilde{x}'_1^T & \tilde{x}'_2^T & \dots & \tilde{x}'_N^T \end{bmatrix} \end{aligned}$$

- 现在我们得到了一个线性的系统，由已知值  $A_i$ ，欲求未知参数  $\tilde{h}$  –  $A$  矩阵的最后一行

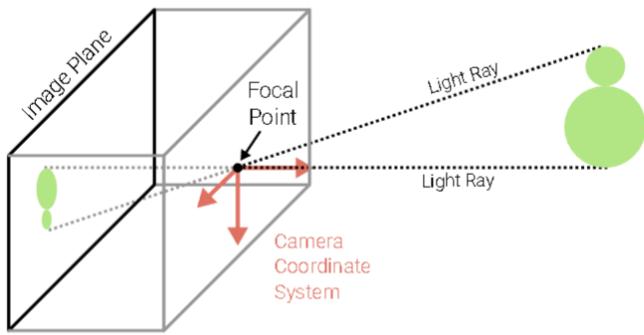
$$\begin{aligned} \begin{aligned} \tilde{h}^* = & \operatorname{argmin}_{\tilde{h}} \|\tilde{h}\|_2^2 + \lambda \|\tilde{h}\|_2 \\ = & \operatorname{argmin}_{\tilde{h}} \|\tilde{h}\|_2^2 + \lambda \|\tilde{h}\|_2 - \lambda \|\tilde{h}\|_2 \end{aligned} \end{aligned}$$

- 其中  $\lambda$  是正则化参数 – 由于  $\tilde{h}$  是齐次的，因此要有  $\|\tilde{h}\|_2^2 = 1$  的约束 – 求解过程对应一

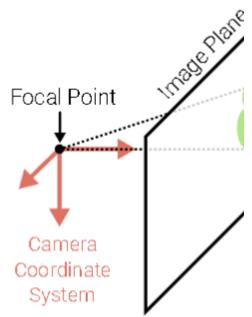
## 2.2 Geometric Image Formation

- 将现实中的 3d 物体转化为 2d 图像的经典技术：投影
  - 最早的摄影机：Pinhole Camera

## Physical Camera Model



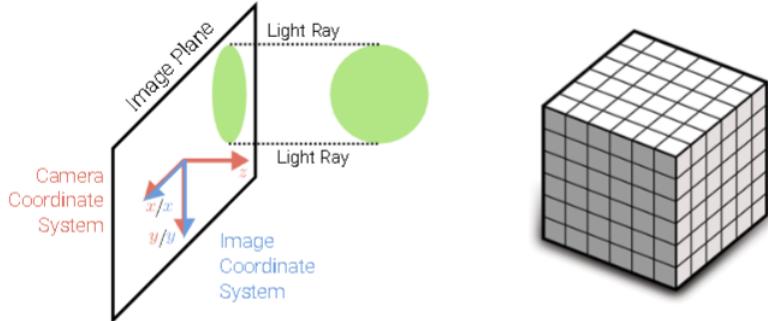
## Mathematical C



- 在本次讲座中，我们认为数学化的相机模型，像位于焦点 (Focal Point) 前面，而非后面；此时像与物体的方向是一致的，并且只要距焦点的距离相同，像的大小和形状是相同的，唯一的区别只是它们之间是镜像的

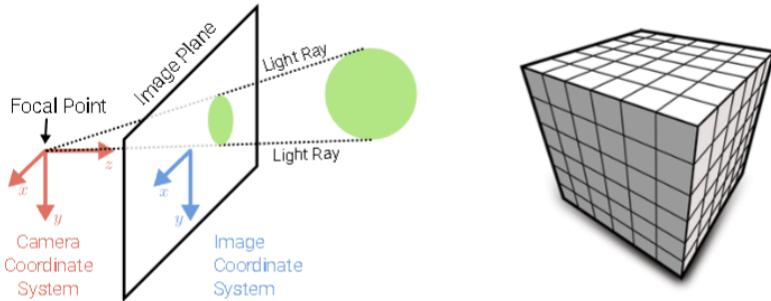
- Projection Models

- Orthographic Projection (正射投影)

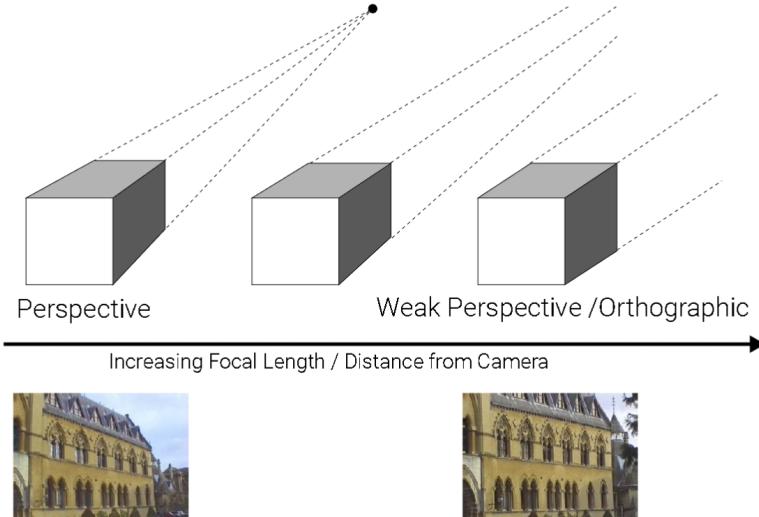


- 光线之间都是平行的
  - 能够非常完整的还原本来物体的形状和外观
  - 在现实中实现非常困难，但是可以用长焦镜头近似实现

- Perspective Projection (透视投影)



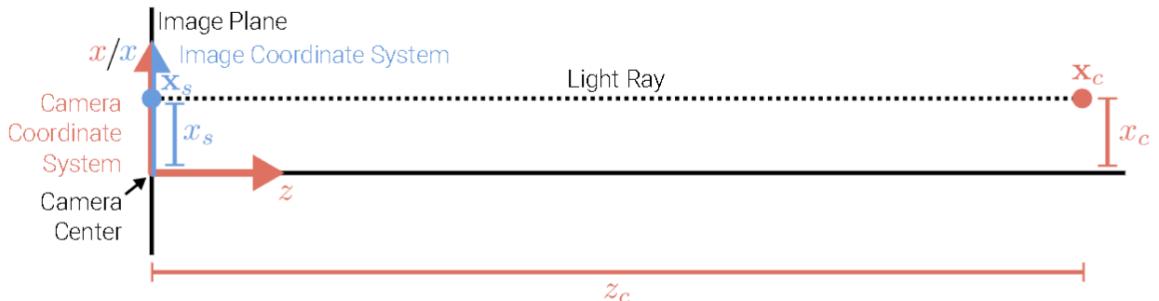
- 类似之前提到的小孔成像模型
  - 成像并非完整还原本来物体的形状和外观，会发生一定的变换
  - 手机、摄像机等的实现



可以理解为，现实中基本不存在理想的平行光源，物体的投影光线总是从一点出发的，该点即为焦点。随着与焦点的距离越远，像越大；因此在 $z$ 轴方向上，随着 $z$ 轴增大， $xoy$ 面的投影面积也逐渐增大；自然投影的物体形状和外观会有所变化。在距离焦点越远的地方取像，则受到的影响越小

- 考虑几何化的两种投影

- Orthographic Projection



- (注意 $y$ 轴没有画出，实际上指向画面向内的方向)

- $3d$  点  $x_c \in R^3$  映射到像素坐标  $x_s \in R^2$
  - 相机坐标系和像素坐标系的 $x$ 、 $y$ 轴是共享的
  - 光线平行于相机坐标系的 $z$ 轴
  - 在投影过程中， $z$ 坐标被丢弃， $x$ 、 $y$ 坐标保持不变

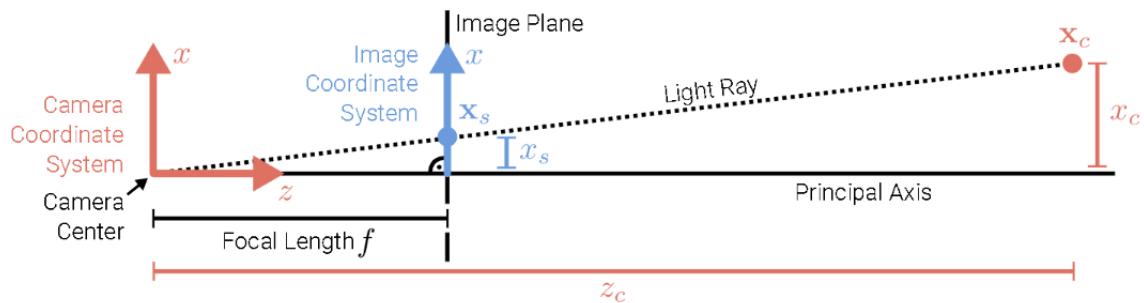
$$x_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_c \Leftrightarrow \bar{x}_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{x}_c$$

- 正射投影只是将 $3$ 维坐标中的 $z$ 分量去掉，从而在图像平面（屏幕）上得到相应的 $2$ 维点
  - 投影后，投影距离是无法恢复的

- Scaled Orthographic Projection

$$x_s = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \end{bmatrix} x_c \Leftrightarrow \bar{x}_s = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{x}_c$$

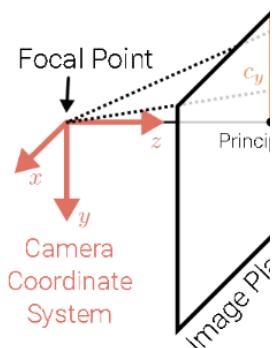
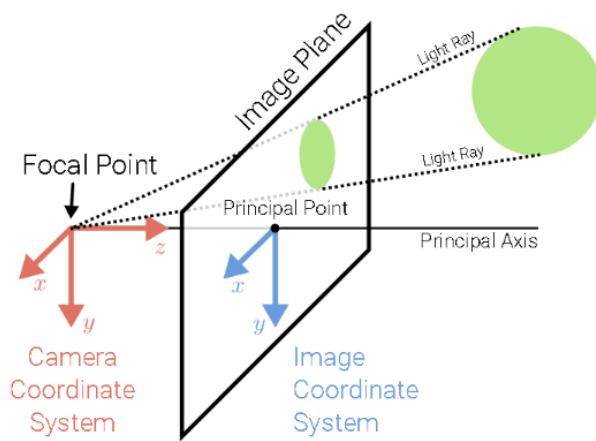
- 在实践中，世界真实坐标（可能以米为单位）必须缩放来适应图像传感器（以像素为单位）
- 缩放因子  $s$  的单位为  $px/m$  或  $px/mm$ ，用于将公制 3d 点转换为像素
- Perspective Projection



- (注意  $y$  轴没有画出，实际上指向画面向内的方向)
  - $3d$  点  $x_c \in R^3$  映射到像素坐标  $x_s \in R^2$
  - 光线穿过焦点、像素  $x_s$ 、点  $x_c$
  - 约定：主轴与  $z$  轴对齐
- $\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} \frac{fx_c}{z_c} \\ \frac{fy_c}{z_c} \end{pmatrix} \Leftrightarrow \tilde{x}_s = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{x}_c$ 
  - 在透视投影中，将相机坐标的分量乘焦距除以  $z$  分量，映射到图像平面上
  - 注意，在使用齐次坐标时，这个投影是线性的；投影后，不可能从图像中恢复 3 维点的距离
  - 焦距  $f$  的单位是 px，从而将  $3d$  坐标单位转换为像素单位
- Perspective Projection (Principle Point Offset)

Without Principal Point Offset

With Princip

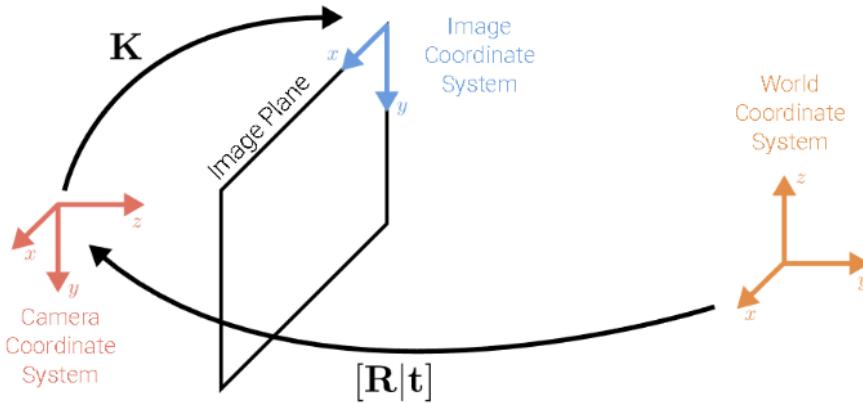


- 当我们以图像的中心点为原点建系时，图像的坐标可能出现负值，不便于计算
  - 因此，我们将原本的所有坐标点加上一个中心点的偏移，也即将建系中心点移至图片的左上角

- $\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} \frac{f_x x_c}{z_c} + \frac{s y_c}{z_c} + c_x \\ \frac{f_y y_c}{z_c} + c_y \end{pmatrix} \Leftrightarrow \tilde{x}_s = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{x}_c$ 
  - 这里  $3 \times 4$  的投影矩阵左侧的  $3 \times 3$  的矩阵称为校准矩阵 (calibration matrix)  $K$
  - $K$  存储了所有相机的内在参数，对于相机而言也称为内在矩阵
  - 这里， $x$  轴和  $y$  轴的焦距是独立的，允许不同的像素长宽比 (不一定必须为  $1:1$ )
  - 由于相机中传感器没有安装在与光轴垂直的方向上，因此可能会产生一个误差倾斜： $s$
  - 在实践中，我们通常令  $f_x = f_y$  并且设置  $s = 0$ ，但是模型  $c = (c_x, c_y)^T$

- Chain Transformations

- 使用齐次坐标表示的好处之一是可以方便地做 3d 变换
- 如果世界坐标系与相机坐标系并不重合，那么我们可以通过变换（平移和旋转）来链式转换到相机坐标系
- 也即我们可以推导出从世界坐标系  $\rightarrow$  相机坐标系  $\rightarrow$  屏幕（也即相机外在矩阵和相机内在矩阵合并的推导式）



- ■ 令  $K$  为校准矩阵， $[R t]$  为相机外在矩阵
  - $\tilde{x}_s = [K \ 0] \bar{x}_c = [K \ 0] \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \bar{x}_w = K [R \ t] \bar{x}_w = P \bar{x}_w$
- 注意， $3 \times 4$  的矩阵  $P$  可以被预先计算

- Full Rank Representation

- 有些时候我们最好使用  $4 \times 4$  的全秩矩阵
  - $\tilde{x}_s = \begin{bmatrix} K & 0 \\ p^T & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \bar{x}_w = \tilde{P} \bar{x}_w$
- 现在，齐次向量  $\tilde{x}_s$  就是一个 4d 向量，必须被标准化；其对应的 3d 非齐次坐标如下
  - $\bar{x}_s = \frac{\tilde{x}_s}{z_s} = (\frac{x_s}{z_s}, \frac{y_s}{z_s}, 1, \frac{1}{z_s})^T$
- 非齐次 4d 向量的第 4 个分量是逆深度，如果逆深度已知，由于  $P$  矩阵是一个满秩矩阵，则可以通过  $\tilde{x}_w = \tilde{P}^{-1} \tilde{x}_s$ ，从屏幕上的坐标计算出对应的世界坐标

- Lens Distortion

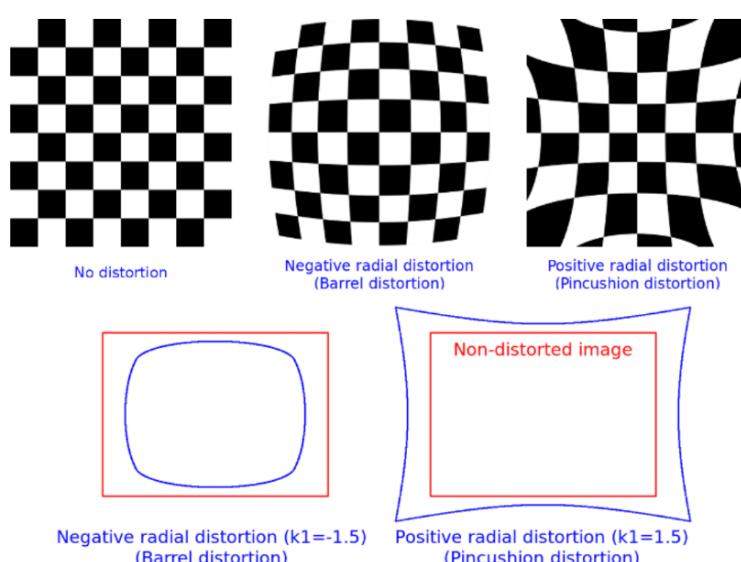
- 如果我们只使用一个针孔，那么我们在传感器上得到的光非常少，因此考虑使用镜头来收集更多的光

- 但是在实际应用中，相机镜头本身的特性会导致一些畸变，因此违背了直线投影（直线保持直线）的假设；
- 幸运的是，径向、切向畸变都可以相对容易的建模
- 设  $x = \frac{x_c}{z_c}$ ,  $y = \frac{y_c}{z_c}$ ,  $r^2 = x^2 + y^2$ ; 则畸变点的计算如下：

$$\mathbf{x}' = \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4)}_{\text{Radial Distortion}} \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} 2\kappa_3 xy + \kappa_4(r^2 + 2x^2) \\ 2\kappa_4 xy + \kappa_3(r^2 + 2y^2) \end{pmatrix}}_{\text{Tangential Distortion}}$$

$$\mathbf{x}_s = \begin{pmatrix} f_x x' + c_x \\ f_y y' + c_y \end{pmatrix}$$

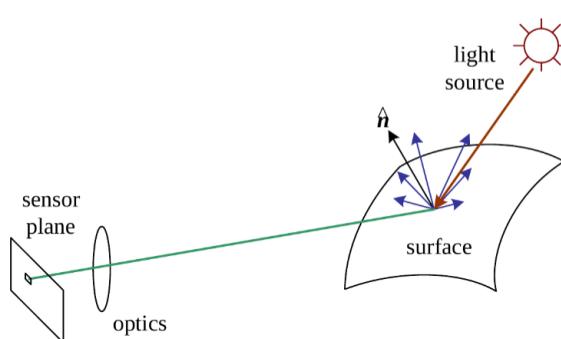
- 通过畸变模型使得图像不失真，这样投影模型就适用了；
- 更复杂的畸变模型必须用于广角镜头（如鱼眼）



## 2.3 Photometric Image Formation

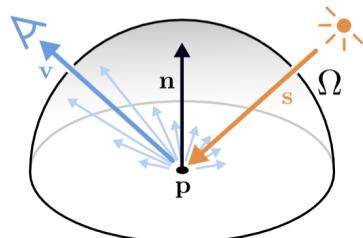
- 一张图片如何根据像素强度和颜色成像呢？

○

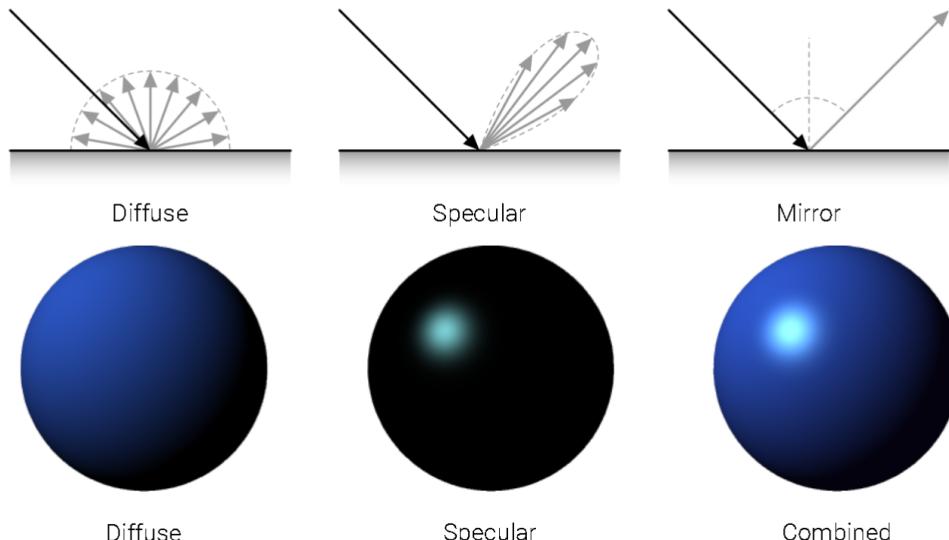


- 渲染方程 (Rendering Equation)

- 令  $\vec{p} \in R^3$  表示 3D 表面的点， $\vec{v} \in R^3$  表示观察方向， $\vec{r} \in R^3$  表示入射光方向
- 渲染方程描述了一束波长为  $\lambda$  的光  $L_{in}$  到达  $\vec{p}$  点后，反射多少光线到观察方向  $\vec{v}$  上
  - $L_{OUT}(\vec{p}, \vec{v}, \lambda) = L_{emit}(\vec{p}, \vec{v}, \lambda) + \int_{\Omega} BRDF(\vec{p}, \vec{r}, \vec{v}, \lambda) \cdot L_{in}(\vec{p}, \vec{r}, \lambda) \cdot (\vec{n}^T \vec{r}) d\vec{r}$



- - $\Omega$  是法线方向为  $\vec{n}$  的单位半球
  - 双向反射分布函数  $BRDF(\vec{p}, \vec{s}, \vec{v}, A)$  定义了光在不透明表面上的反射方式
  - 仅对发光表面有  $L_{emit} > 0$
  - Diffusion and Specular Reflection

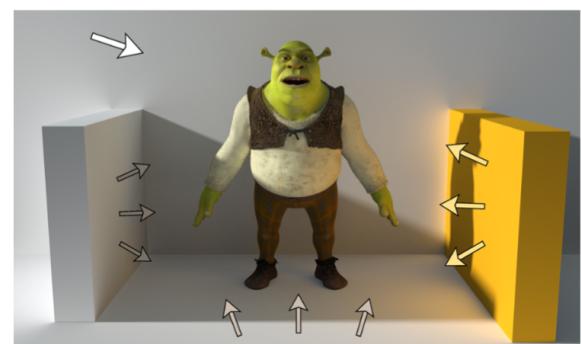


- - 典型的 BRDF 有漫反射 (Diffuse) 和镜面反射 (Specular) 两个分量
  - 漫反射分量在各个方向均匀地散射数值为常数的光，这导致了阴影，也即强度相对的平滑变化
  - 镜面分量强烈地依赖于出射光的方向
  - 但是实际上，BRDF 的讨论在现实中是非常复杂的，稍微细小的变化都会影响到整体 BRDF 值的变动
    - Fresnel Effect (菲涅尔效应)
      - 比如水面、湖面，当你看向远处时，水面呈现出近乎镜面的效果；当你垂直向下看时，水面几乎是透明的；因此，这位 BRDF 的计算有了更大的挑战：你必须考虑进去类似于水面这种材料的特性

- Global Illumination



Rendering with Direct Lighting

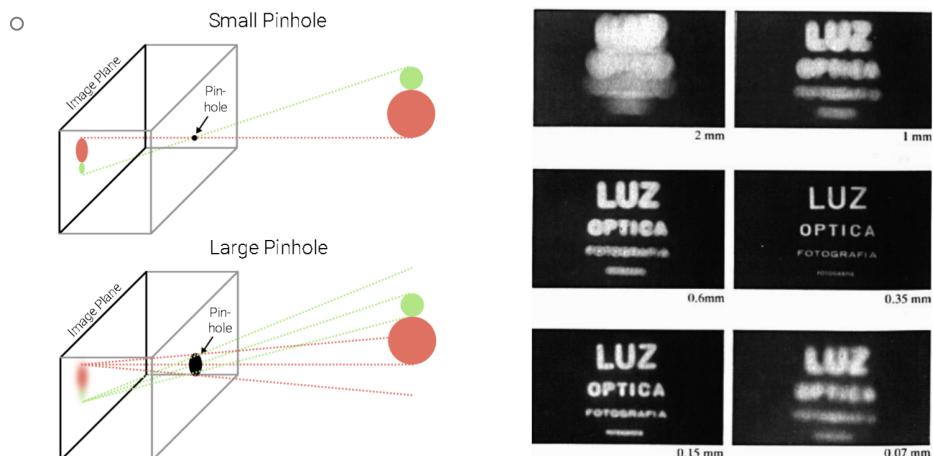


Rendering with Global Illumination

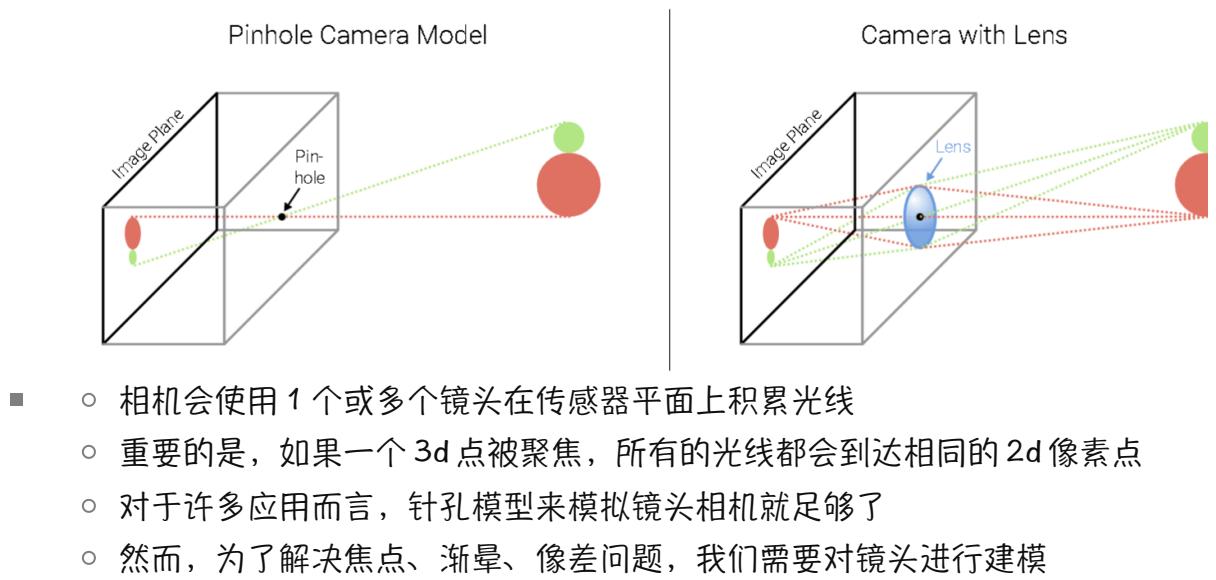
- 只考虑建模一次反射时不足以渲染复杂的场景的，光源会被遮挡物遮挡，光线会多次反射
- 这在 CV 中被称为全局照明 (Global Illumination) 技术，它也考虑了间接照明

### • Camera Lenses

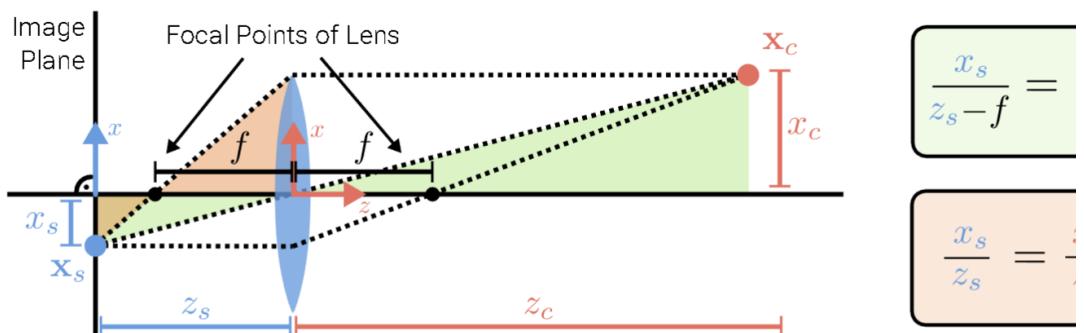
- 为什么需要镜头呢？
- 针孔过大或针孔过小都会导致图像模糊 (衍射、反射、太长的快门时间以至于运动模糊等...)



- 光学器件

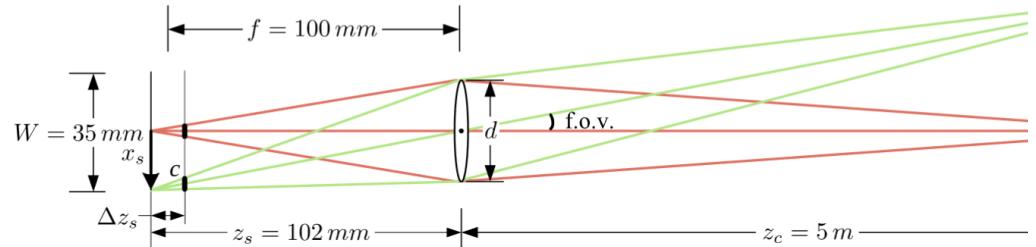


### ■ Thin Lens Model



- 最常见的一种建模，两篇都是向外凸的平面，称为球面透镜，通常用于近似
- 性质：平行于轴的光线通过焦点，通过中心的光线会保持原来的方向

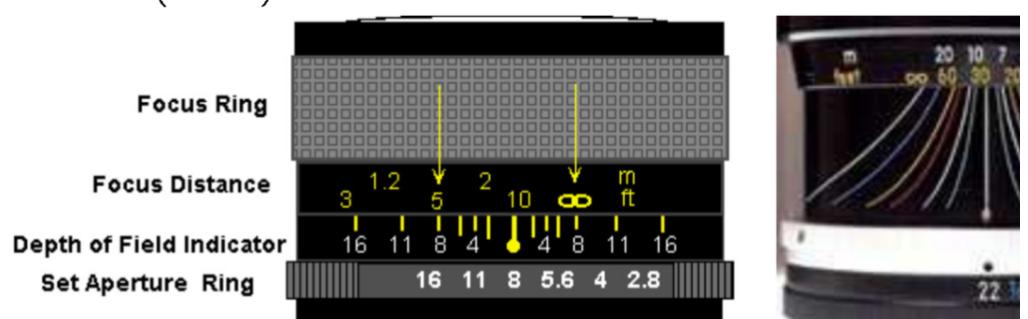
- 由 Snell 法则，当半径为  $R$ 、折射率为  $n$  时， $f = \frac{R}{2(n-1)}$
- DOF(Depth of Field) (景深)



- 如果  $f$  是镜头的焦距，有  $\frac{1}{Z_s} + \frac{1}{Z_c} = \frac{1}{f}$
- 当  $Z_c \rightarrow \infty$ ，我们有  $Z_s = f$  (镜头的焦距为  $f \approx$  小孔的距离为  $f$ )
- 如果一个图像平面脱焦，意味着一个 3d 点会投影到一个混淆圆  $c$  上



- 为了控制混淆圆的大小，我们可以改变 aperture (光圈大小)
- 光圈控制了能到达成像面的光线
- 光圈越小，则图像越清晰，但噪声越多



- 限制混淆圆  $c$  的允许深度变化称为景深，它是焦距  $f$  和镜头光圈 aperture 的函数
- 典型的数码单反摄像机有景深指示器
- 通常显示的  $f$  值定义为  $N = \frac{f}{d}$ ，也即镜头焦距和光圈直径的比值



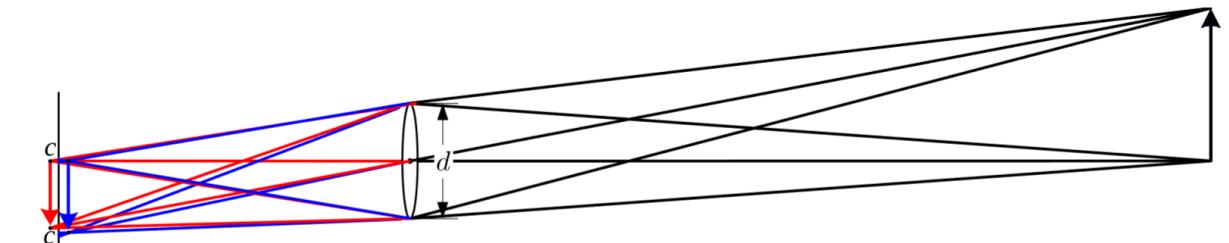
Aperture = f/1.4  
DOF = 0.8 cm

Aperture = f/4.0  
DOF = 2.2 cm

- DOF

- 最近和最远物体之间的锐利距离
- 减小光圈直径(增大 f 值)来提高 DOF 值

- Chromatic Aberration (色差)

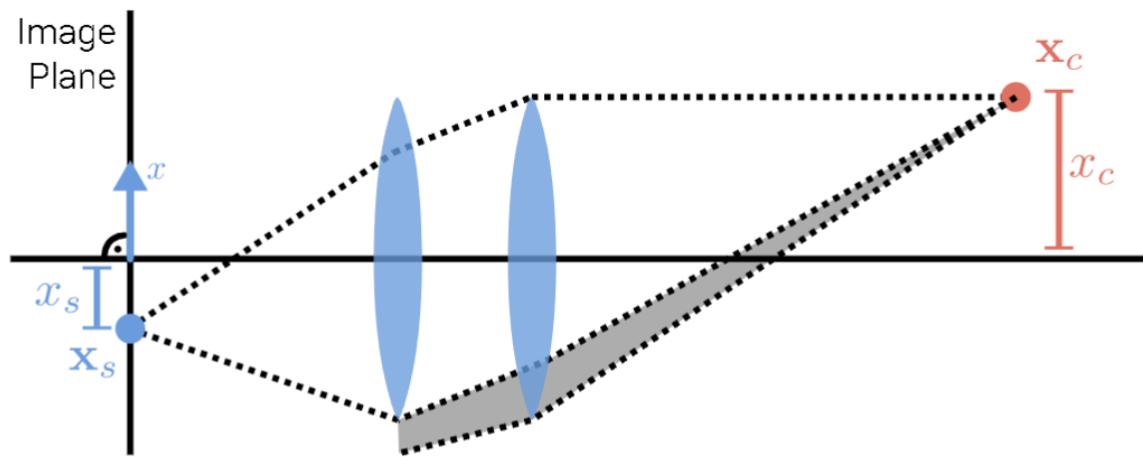


- ■ 玻璃的折射率随波长稍有不同
- 因此，简单的镜头会受到色差的影响
  - 色差，由于不同颜色的光的反射情况不同；因此会出现不同颜色的光汇聚的点之间有所偏移，在视觉上产生色差的现象
- 为了减少色差，镜头会由不同类型的玻璃元素涂层制成



► Top: High-quality lens      Bottom: Low-quality lens (blur, rainbow edges)

- Vignetting (渐晕)



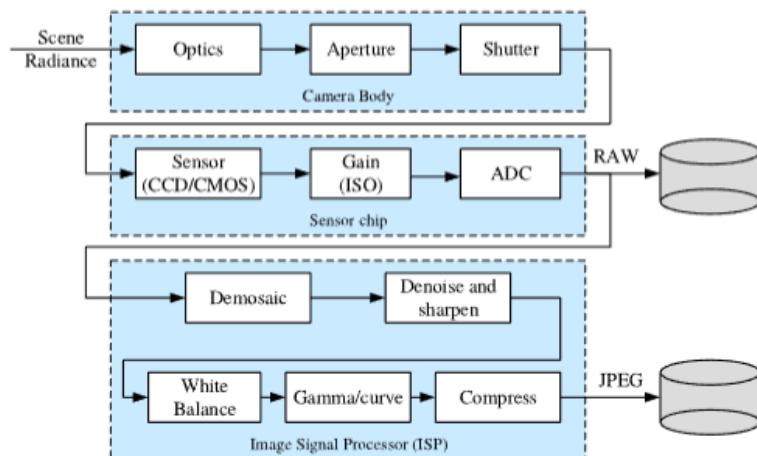
- ■ 渐晕是亮度向图像边缘下降的一个趋势
- 分为 2 种：自然渐晕和机械渐晕
  - 自然渐晕：物体表面和砸镜头光圈的缩短
  - 机械渐晕：光束的阴影部分永远不会到达图像
- 渐晕是可以完全消除的



- ○ 右侧是产生渐晕的图像
- 左侧是渐晕消除的图像

## 2.4 Image Sensing Pipeline

- 光线穿过镜头到达成像平面后发生的情况——



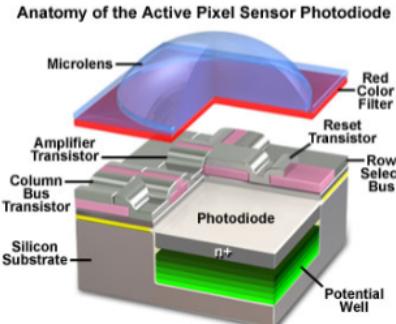
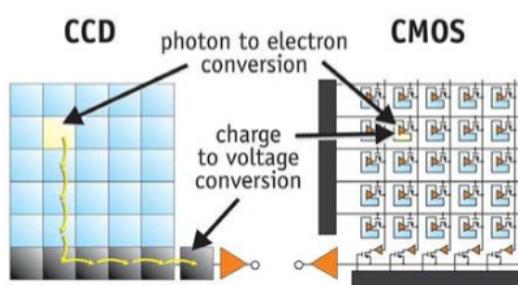
- 大体上可以分为 3 个阶段
  1. Camera Body
    - 光线->相机光学器件 -> 光圈 -> 快门
  2. Sensor chip
    - 传感器(CCD/CMOS) -> 聚集电流(ISO) -> ADC -> 原始图像
  3. Image Signal Processor(ISP)
    - 去噪-> 锐化 -> 白平衡 -> 亮度调整 -> 压缩 -> JPEG
- Shutter(快门)



- ■ 焦平面快门位于图像传感器/胶片的正前方
  - 大多数数码相机使用机械和电子快门的组合
  - 快门速度 (曝光时间) 控制有多少光到达传感器
  - 它确定图像是否过度/曝光不足, 模糊(blurry)或嘈杂(noisy)

- Sensor

- 



- CCDs
  - 将电荷从一个像素转移到另一个像素，并在输出节点将其转换为电压
- CMOSs
  - 图像将每个像素内的电荷直接转换为标准量程下的电压
  - 更厚的胶片 (35mm) 具有更好的感光性，可以实现更少的噪声

- Color Filter Arrays

- 

G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

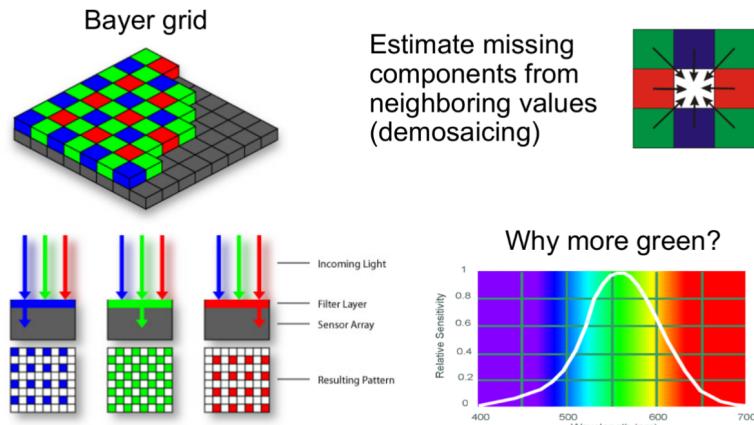
Bayer RGB Pattern

rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

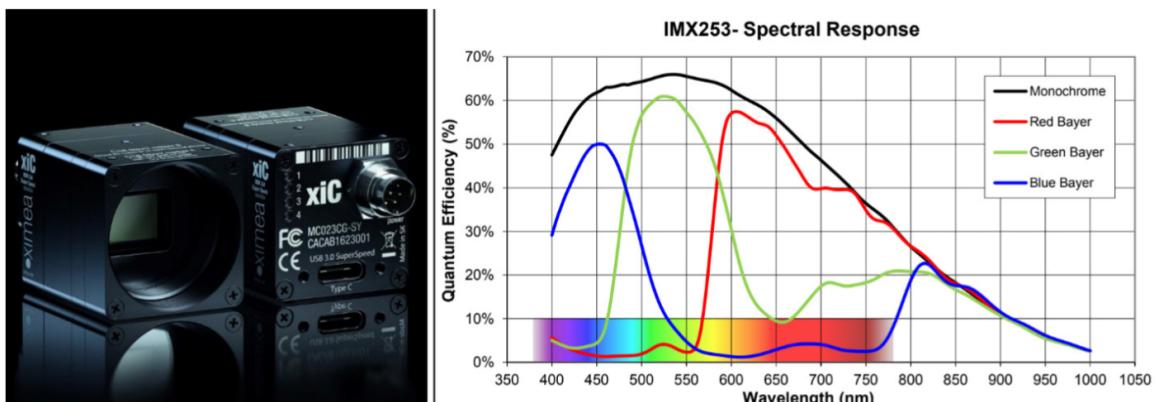
Interpolated Pixels

- 我们必须有一种方式来测量像素的颜色，通过在每个像素顶部位置放置一个小的颜色滤镜
- 每个像素的滤镜只对一种特定的颜色，因此我们需要考虑每个像素放置什么颜色的滤镜；因此提出 Bayer RGB 模式
- 由于人眼对绿色较为敏感，因此在每个 Bayer 单元中有两个绿色像素
- 每一个像素只能测量 RGB 三种颜色中的 1 种，因此需要从相邻的像素获取另外两种颜色的值并赋值给自身，这个过程称为 Demosaicing(去马赛克)

- 有许多种 Demosaicing 的算法

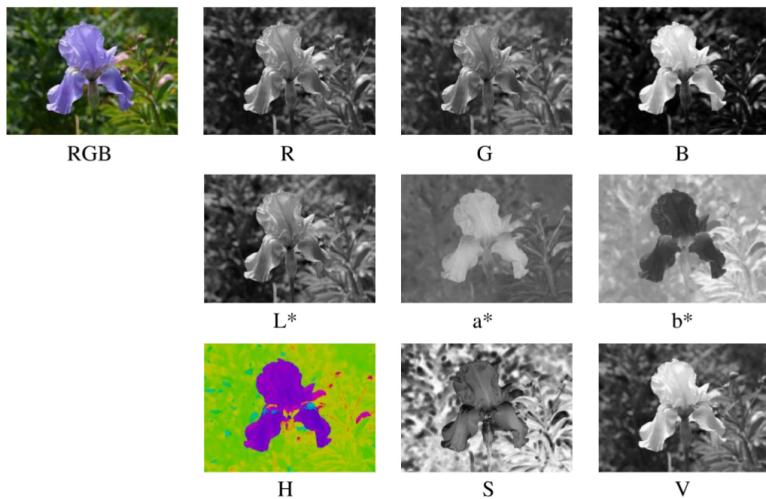


- 可以看到，我们在底片上面放置了很多种特定颜色的滤镜，这些滤镜只允许特定的颜色通过

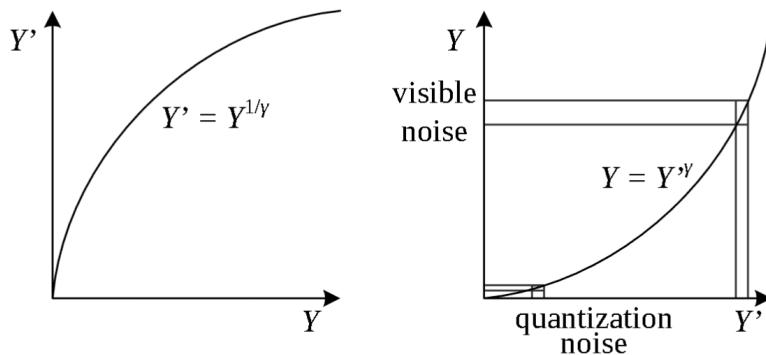


- 具体取决于相机所采用的光学器件对于光的灵敏度作为采集到的特定颜色光线的数值
- 不同相机的灵敏度曲线不同，可以看到它甚至不是严格划分到特定颜色区域的
  - $R = \int L(\lambda)S_R(\lambda)d\lambda$
- 根据相机灵敏度在光谱上的积分，得到最后的数值

- Color Spaces



- ■ H: 色调
- S: 饱和度
- V: 数值
  - 各种不同的色彩空间都在各种不同的实践中被应用
- Gamma Compression



- ■ 为了方便将这些图像更好的存储，我们需要对得到的连续化数值进行离散化
- 人类对于深色区域的强弱差异变化更为敏感
- 因此，一般我们会在加载前对颜色或强度进行如上述左图的非线性变化  $Y' = Y^{\frac{1}{\gamma}}$ ，并在加载时撤销此操作；这个过程称为 *Gamma Compression*
- Image Compression



- ■ 通常，亮度的压缩保真度会高于色度（因为人类对亮度的变化更敏感）
- 在经典的压缩算法，例如jpeg 种，使用基于  $8 \times 8$  像素 batch 的余弦离散或小波变换算法
- 离散余弦变化 (DCT) 是一种近似于 PCA 的自然图像处理方法
- DCT 的系数被量化为整数，可以用 Huffman 编码存储
- 最近，基于深度网络的压缩算法得到了发展

### 三、Structure from motion