目录

第一章 Introduction

计算机视觉给予计算机顶尖猎杀者的通行证 (眼睛).

这门课重点在三维重建

规范: 网上的东西都引用一下为好

一、 What is Computer Vision

1. A standard computer vision system

摄像机 (Camera) 照一个有光源 (Lighting) 的场景 (Scene) 通过视觉算法 (Vision Software) 转换为场景描述 (Scene Description).

识别场景中属性

(1) 3D 形状

(2) 辨识物体

(3) 发生事件

2. Computer vision tasks

(1) Reconstruction(三维重建)

(2) Understanding(图像理解)

(3) Synthesis(图像合成)

3. Why is computer vision hard

计算机所视为一个矩阵 (所以线性代数特别多), 需要从数字中提取信息.

人工智能现阶段瓶颈在我们对人脑认识并不清晰. 视觉是个基本功能, 但我们未知其如何具体实现的. 计算机与人类各有优劣, 需要探寻其不同的具体实现方式.

如何把问题转换成数学问题.

4. Human perception

Shortcomings& 一些视错觉: 明暗分辨, 光流, 人对三维旋转的感知需要深度 (eg: 遮挡关系).

Advantages: 理解与想象的能力, 可以从较少信息的图像获取较多的信息.

二、 What is Vision Used For

1. Applications

(1) 人脸识别: 点云 (3D 信息, 安全), 红外

DeepFake
(2) AI 换脸: 危险.jpg, 开源但被下了.

Augmented reality
(3) 现实增强 : 美颜, 特效 (确定关键点来改变) eg:mediapipe(开源)

Factory Automation,Vision Inspection
(4) 工业检测

Optical Character Recognition(OCR)
(5) 光学文字识别

Video Surveillance
(6) 监控

Human Computer Interaction Optical Mouse Motion Sensing Games Motion Capture
(7) 人机交互 : 光学鼠标 , 体感游戏 (识别骨架), 动作捕捉

Digital Human
(8) 数字人 : 三维建模, 动作捕捉

Sports Broadcasting
(9) 体育分析

3D Street View
(10) 3D 街景 : 以前做不到, 但现在可以真 3D 街景了

(11) VR Tour

Visual Localization and Navigation
(12) 视觉导航 : 用视觉定位.

Autonomous Navigation
(13) 自动导航 : 视觉定位

Robot Perception
(14) 机器人感知

Autonomous Driving
(15) 自动驾驶

Free Viewpoint Video
(16) 全息呈现

Medical Image Analysis
(17) 医学影像分析

2. Vision research

计算机视觉三大顶级会议:CVPR,ICCV,ECCV
图形学顶会:ACM_SIGGRAPG

三、 Course Overview

1. Target

进入这个领域, 用数学描述以解决问题 (线性代数与优化).PS:Lab 很硬

2. Overview

(1) Basic.(Lec.02 - Lec.04)

- a. Lec.02: $\overset{\text{图像形成}}{\text{Image formation}}$
  - i. Geometric formation: Camera model
  - ii. $\overset{\text{光度}}{\text{Phototmetric}}$ formation: $\overset{\text{上色}}{\text{Shading}}$, color, $\overset{\text{传感器}}{\text{sensors}}$
- b. Image $\overset{\text{处理}}{\text{processing}}$
  - i. Image $\overset{\text{滤波}}{\text{filtering}}$
  - ii. Image resize and reshape
- c. Model $\overset{\text{拟合}}{\text{fitting}}$ and $\overset{\text{优化}}{\text{optimization}}$
  - i. Model fitting
  - ii. Mathematical optimization

(2) Reconstruction.(Lec.05 - Lec.08)

- a. $\overset{\text{特征匹配}}{\text{Feature matching}}$ and $\overset{\text{运动估计}}{\text{motion estimation}}$
  - i. Feature matching
  - ii. Motion estimation
- b. Image alignment and stitching
  - i. Image transformation
  - ii. Image $\overset{\text{拼接}}{\text{stitching}}$
- c. $\overset{\text{运动恢复结构}}{\text{Structure from motion}}$
  - i. 从视频恢复点云
  - ii. 视觉定位
  - iii. $\overset{\text{同步定位建图}}{\text{Simultaneous localization and mapping}}$ (SLAM)
- d. 3D reconstruction(稠密)(the most difficult)
  - i. $\overset{\text{立体匹配}}{\text{Stereo matching}}$
  - ii. Surface Representations
  - iii. Surface Extraction from Point Clouds

(3) Understanding. (Lec.9 - Lec.11)

- a. Deep learning
- b. $\overset{\text{识别}}{\text{Recognition}}$
  - i. $\overset{\text{分类}}{\text{Classification}}$
  - ii. $\overset{\text{语义分割}}{\text{Semantic Segmentation}}$
  - iii. Object $\overset{\text{检测}}{\text{Detection}}$

iv. $\overset{\text{实例分割}}{\text{Instance Segmentation}}$

v. Human pose estimation

vi. Action recognition

c. 3D deep learning

(4) Synthesis.(Lec.12 - Lec.13)

a. Computational photography I

i. $\overset{\text{高动态范围成像}}{\text{High dynamic range(HDR)}}$

ii. $\overset{\text{超分辨率}}{\text{Super-resolution}}$

iii. Image-to-image translation

b. Computational photography 2

i. 虚拟视点合成:$\overset{\text{局域神经网络}}{\text{Neural rendering}}$

四、 Review of Linear Algebra

1. Vectors

(1) 长度与方向 $\vec{a}$

(2) 范数 $\|\vec{a}\|$, 归一化 $\hat{a} = \vec{a}/\|\vec{a}\|$

(3) 平行四边形法则 $\vec{a} + \vec{b}$

(4) 内积/点积 $\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta$,$\cos\theta = \hat{a}\hat{b}$

(5) 向量投影 $\vec{b}_\perp = k\hat{a}$, $k = \|\vec{b}\| \cos\theta$

2. Matrix

(1) $A_{m\times n} = \begin{pmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{pmatrix}$

(2) 矩乘 $A_{m\times n} \times B_{m\times p}$, 性质

a. $(AB)C = A(BC)$

b. $A(B + C) = AB + AC$

c. $(A + B)C = AC + BC$

(3) 单位矩阵 $I_{3\times 3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $AA^{-1} = A^{-1}A = I$, $(AB)^{-1} = B^{-1}A^{-1}$

(4) 矩阵乘向量是: 矩阵所有列的加权求和 or 向量的坐标变换

eg:

a. $\overset{\text{缩放}}{\text{scale}}$: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

i. $s_x = 0.5, s_y = 0.5$ 变小

ii. $s_x = -1, s_y = 1$ 关于 y 镜像

b. 剪切: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & a \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$, 相当于 $x' = s_x x + ay$

c. 旋转: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$, θ 是关于正 x 的角度.

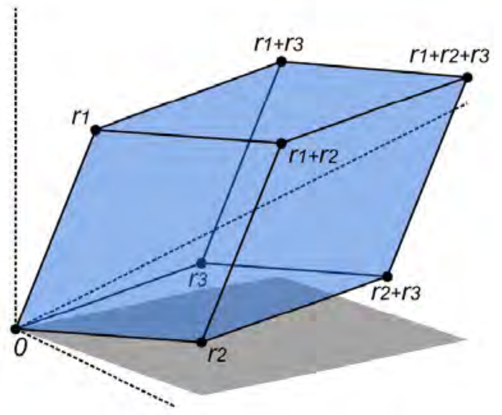
d. 平移/仿射: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$

or using $\overset{\text{齐次坐标}}{\text{homogenous coordinates}}$: $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

(5) 逆矩阵 T^{-1}

(6) determinant: $det(A) = \sum_{\sigma \in S_n} \left(\text{sgm}(\sigma) \prod_{i=1}^{n} a_{i,\sigma_i} \right)$

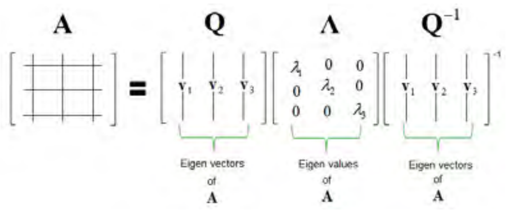
行列式是三维向量张成的体积, 是变换的大小?



(7) $\overset{\text{特征向量与特征值}}{\text{Eigenvectors and Eigenvalues}}$

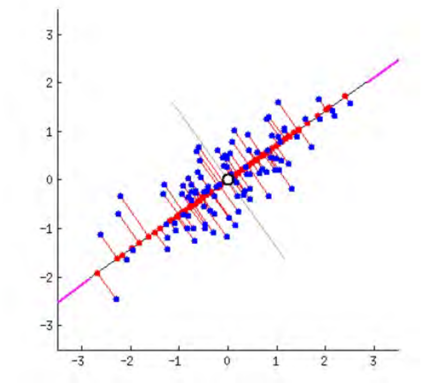
a. $Ax = \lambda x$, A 对特征向量 x 的变换等价于对 x 的缩放, 没改变 x 的方向

b. $A = Q\Lambda Q^{-1}$, 直接调用函数实现



特征值不一定是实数 (可能是复数), 仅 A 为对称时才是实数

c. 应用: 主成分分析. 一堆点找一个方向, 在这个方向上这些点的方差最大.



将点记为 A, 最大方向就是 $A^{-1}A$ 最大特征值的特征向量

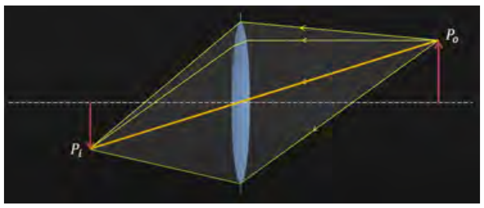
第二章　Image formation

一、　Camera and lens

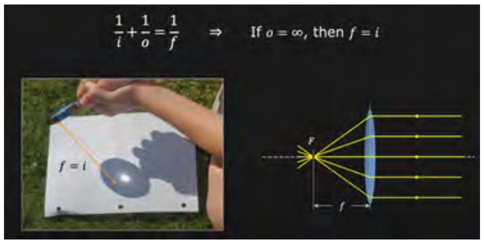
1.　Image formation

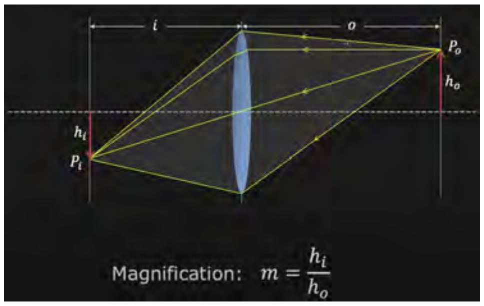
(1) Pinhole camera: $\overset{\text{光圈}}{\text{aperture}}$(小孔成像), 过小会发生衍射, 且通光量过少.

(2) Lens: 汇聚光线, $\frac{1}{i}+\frac{1}{o}=\frac{1}{f}$, 可用平行光测量 f, $\overset{\text{放大率}}{\text{Magnification}}$: $m=\frac{h_i}{h_o}=\frac{i}{o}$, 现阶段 $\overset{\text{像距}}{\text{i}}$ 基本上等于 $\overset{\text{焦距}}{\text{f}}$, 同时焦距也决定了视野.



(a) Lens

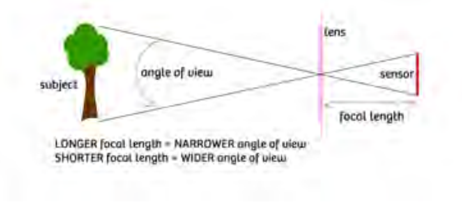

$\frac{1}{i}+\frac{1}{o}=\frac{1}{f}$ ⇒ If $o=\infty$, then $f=i$
$f=i$
F
f
(b) f

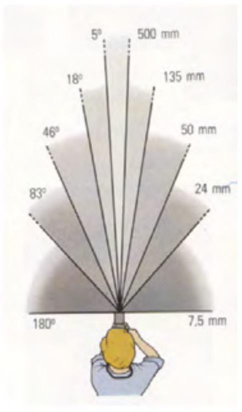


(c) Magnification

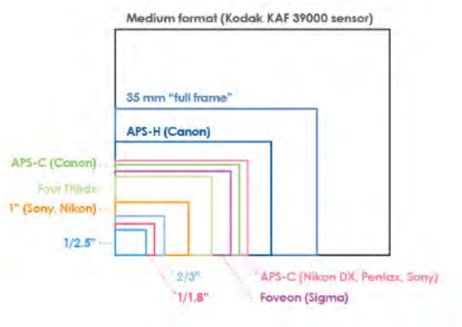
图 2.1: Lens

(3) $\overset{\text{视野}}{\text{Field of View}}$(FOV), 不同焦距也影响着背景的变化


lens
subject
angle of view
sensor
focal length
LONGER focal length = NARROWER angle of view
SHORTER focal length = WIDER angle of view
(a) FOV



(b) 焦距与视野



(c) 镜头与视野

图 2.2: FOV

F 数
(4) Aperture: $D = \frac{f}{N}$, N is called F-Number.

散焦
(5) Lens defocus, 会成斑.

$$b = \frac{D}{i'}\left|i' - i\right|, b \propto D \propto \frac{1}{N}$$

b 是斑的大小.

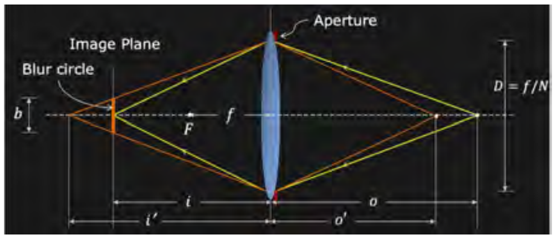


图 2.3: Lens defocus

对焦
(6) Focusing

景深
(7) Depth of Field(DOF)

$$c = \frac{f^2(o - o_1)}{No_1(o - f)}$$
$$c = \frac{f^2(o_2 - o)}{No_2(o - f)}$$
$$(DOF)o_2 - o_1 = \frac{2of^2cN(o - f)}{f^4 - c^2N^2(o - f)^2}$$

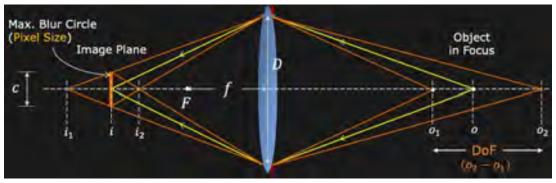


图 2.4: DOF

虚化背景
To blur the background:

a. Large aperture
b. Long focal length
c. Near foreground
d. Far Background

二、 Geometric image formation

相机模型
Camera model describes the geometric relation between 3D world and 2D image

1. Pin-hole camera model

透视变换
(1) Perspective Projection

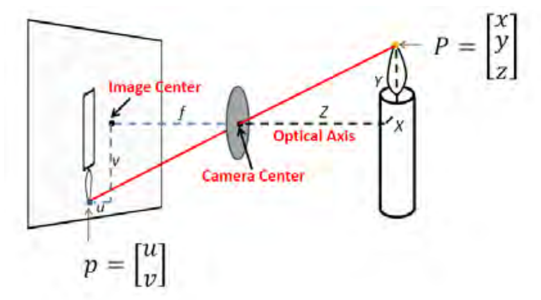

$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
Image Center
f
Z
Y
X
Optical Axis
v
u
Camera Center
$p = \begin{bmatrix} u \\ v \end{bmatrix}$

图 2.5: Pin-hole

$$p = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \end{bmatrix}$$

齐次坐标
(2) Homogeneous coordinates

Also called projective coordinates.

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \cong \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ 1 \end{bmatrix}$$

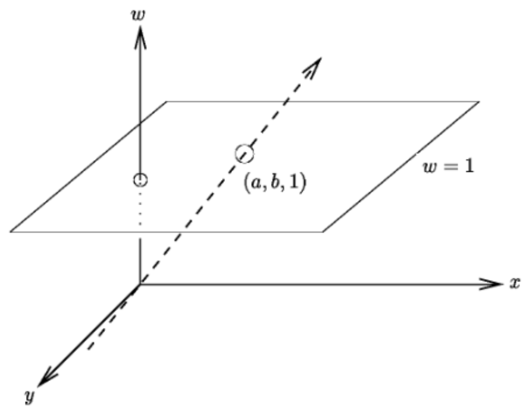


图 2.6: Homogeneous coordinates

将二维的点升维到三维过原点的射线. 二维的平移可以等效于三维的旋转.

(3) Visualize Perspective Projection

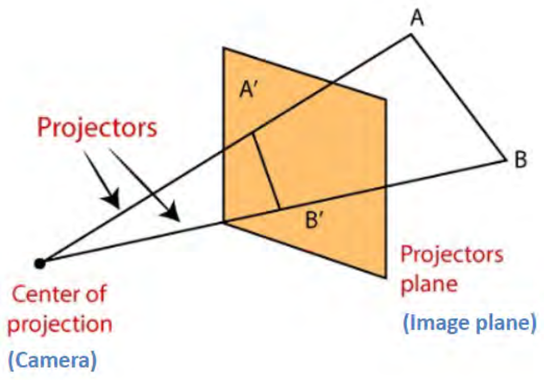


图 2.7: Perspective Projection

(4) 信息损失

a. 深度

b. 长度 (因为损失深度)

c. 角度

d. 非双射, 一个二平面可以对应无数三维物体

(5) 信息保留

a. 线性: 直线还是直线

(6) $\overset{\text{灭点}}{\text{Vanishing}}$ points

一组平行的线的交点, 可以在画外 or 无穷远处

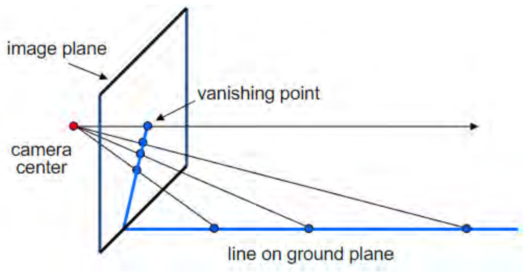


图 2.8: Vanishing points

Properties:

a. 任意两条平行线会交于一个灭点

b. $\overset{\text{camera center}}{C}$ $\overset{\text{vanishing point}}{V}$ 平行于三维的线

可以获得平行线的朝向 or 相机的朝向

(7) $\overset{\text{地平线}}{\text{Vanishing lines}}$

一个平面内所有平行线的灭点的连线, 地平线与相机中心构成的平面与原平面平行.

(8) Perspective $\overset{\text{畸变}}{\text{distortion}}$

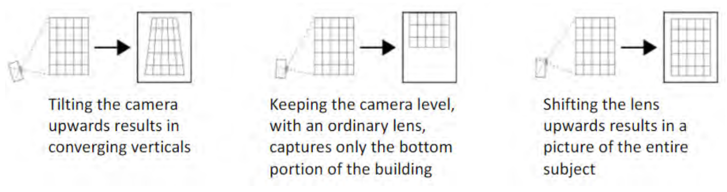

Tilting the camera upwards results in converging verticals
Keeping the camera level, with an ordinary lens, captures only the bottom portion of the building
Shifting the lens upwards results in a picture of the entire subject

图 2.9: Perspective distortion

解决方案: 移轴镜头

越边界越大, 不是因为镜头

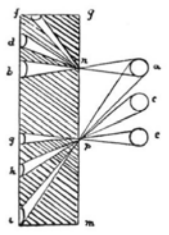


图 2.10: Perspective distortion

(9) Distortion

是因为镜头造成的, 可以后期矫正的

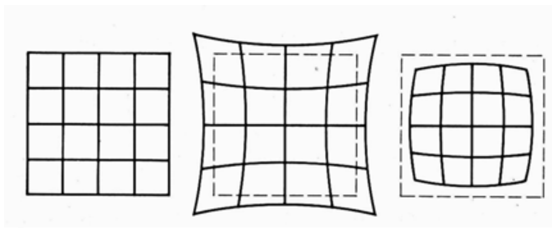


图 2.11: 无畸变, 枕形畸变, 筒形畸变

$$r^2 = {x'}_n^2 + {y'}_n^2$$
$$x'_d = x'_n(1 + k_1 r^2 + k_2 r^4)$$
$$y'_d = y'_n(1 + k_1 r^2 + k_2 r^4)$$

2. Orthographic projection

正交投影, 简单但不符合实际, 只是种近似.

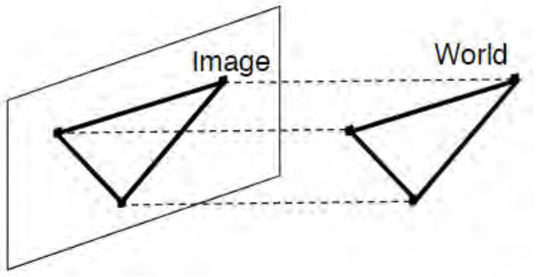


图 2.12: 正交投影

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

三、 Photometric image formation

1. Image sensor

每个像素是光强关于时间的积分.

(1) CMOS(Complimentary Metal-Oxide Semiconductor): 数码底片, 光-> 电, 进行记录

(2) CCD(Charge Coupled Device): 用的少了

2. Shutter

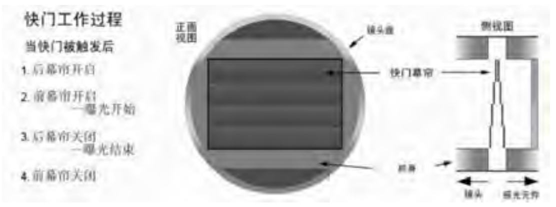


图 2.13: 快门

(1) Shutter speed: 快门/曝光时间

(2) Shutter effect: 不同曝光时间的效应

(3) IOS: 传感器敏感度

(4) rolling shutter effect: 快门一行一行打开, 导致物体结构被破坏了

3. Color sensing in camera

颜色和波长有关

(1) Color spaces: RGB(所有颜色是由 R,G,B 混合而成), HSV($\overset{\text{色调}}{\text{Hue}}$, $\overset{\text{饱和度}}{\text{Saturation}}$, value)

贝尔滤波器
(2) Bayer filter

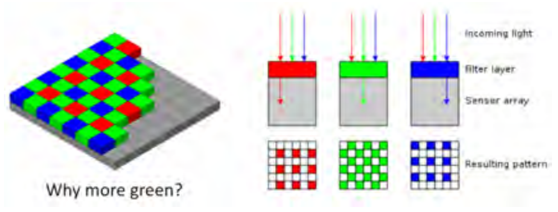


图 2.14: Bayer filter

旁边颜色用插值获取, 绿色多因为人眼对绿光更敏感, 绿光多信噪比更好.

第三章　Image processing

一、　Image processing basics

1. Example

(1) Increasing $\overset{\text{对比度}}{\text{contrast}}$ with ”S curve”, $ouput(x, y) = f(input()x, y)$

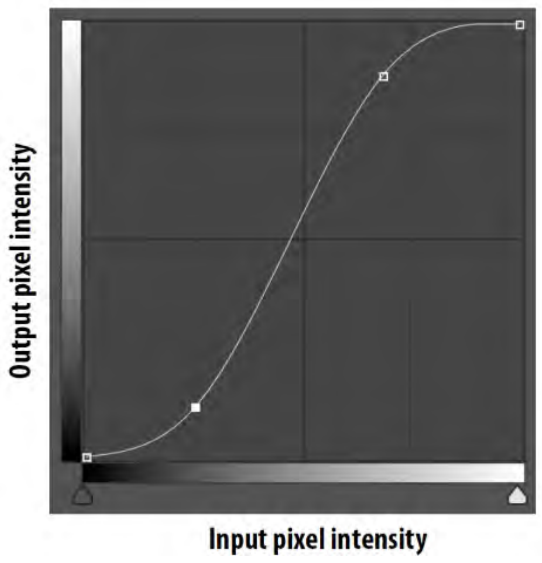


图 3.1: Increasing contrast

(2) 整体变亮

(3) $\overset{\text{模糊}}{\text{Blur}}$: 去噪

(4) ”smarter” blur

(5) $\overset{\text{锐化}}{\text{Sharpen}}$

(6) $\overset{\text{边缘检测}}{\text{Edge detection}}$

2. Review: convolution

$$\overset{\text{output signal}}{(f * g)(x)} = \int_{-\infty}^{\infty} \overset{\text{filter}}{f(y)}\ \overset{\text{input signal}}{g(x-y)}\ dy$$

卷积意义: 对输入的每个点, 将其周围的值取出, 并用卷积核相乘, 再积分 (离散就直接相加).

(1) 卷积核需要翻转 (但一般就是对称的所以忽略了翻转).

(2) 在亮度相差不大的情况下, 图像中与卷积和越相似的地方响应越大.

3. Discrete 2D convolution

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} \overset{\text{filter}}{f(i,j)} I(x-i, y-j)$$

会让输出变小

4. Padding

要在图边缘填充像素

(1) Zero values

(2) Edge values

(3) Symmetric

eg:

(1) box blur: 取平均

(2) Gaussian blur: 加权平均

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

The bigger σ is, the more blurry pic will be.

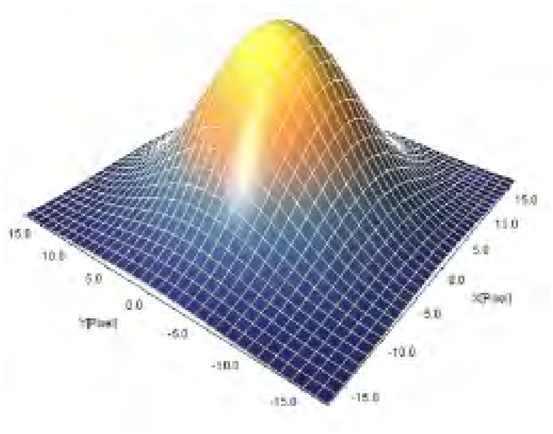


图 3.2: Gaussian

(3) Sharpens: 增加对比度

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

a. I is original image: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

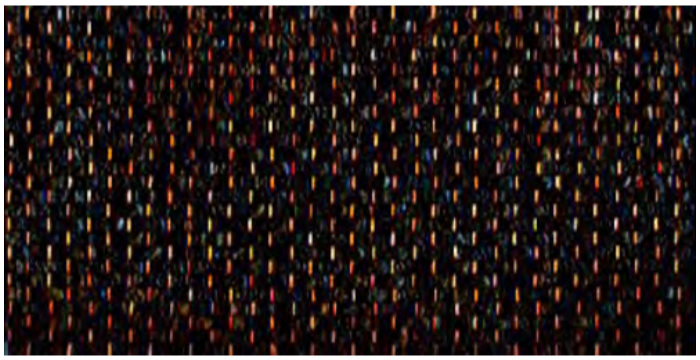
b. High frequencies in image $I = I - blur(I)$, (原本减去低频就是高频)

c. Sharpened image $= I + (I - bule(I))$. (加强高频)

(4) Gradient detection filters(梯度边缘检测)

a. Extracts horizontal gradients(检测竖直边缘): $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

b. Extracts vertical gradients(检测水平边缘): $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$


(a) Horizontal gradients

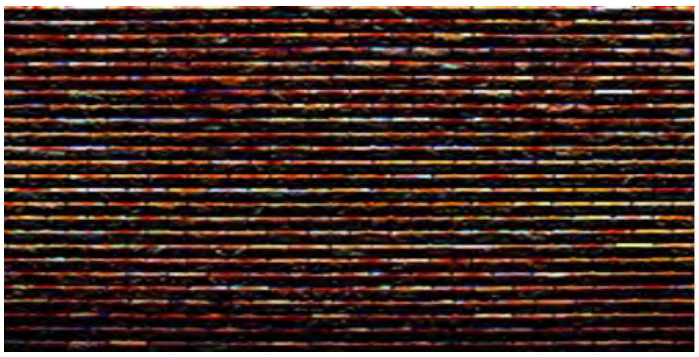

(b) Vertical gradients

图 3.3: Gradient detection filters

(5) Bilateral filter(双边滤波器): 图像变得更加光滑 (去噪), 但保留边缘信息

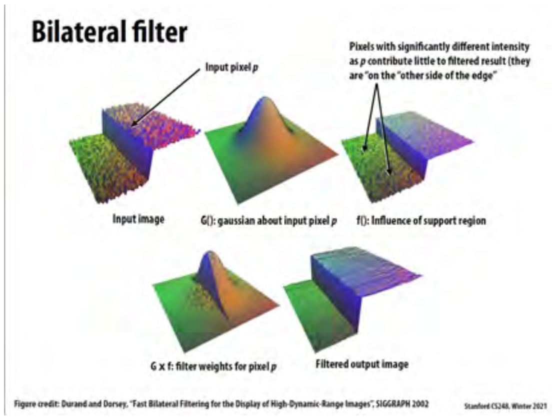

Bilateral filter
Input pixel p
Pixels with significantly different intensity as p contribute little to filtered result (they are "on the "other side of the edge"
Input image
G(): gaussian about input pixel p
f(): Influence of support region
G x f: filter weights for pixel p
Filtered output image
Figure credit: Durand and Dorsey, "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images", SIGGRAPH 2002
Stanford CS248, Winter 2021

图 3.4: Bilateral filter

二、 Image sampling(采样)

1. Image resizing

$\overset{\text{pixels}}{\text{像素}}$ 尺寸与 $\overset{\text{inches}}{\text{物理}}$ 尺寸, 物理尺寸有意义需要 $\overset{\text{resolution}}{\text{分辨率}}$, 单位: 像素/英寸. 人眼分辨率是 300 dip

2. Down-sampling

Reducing image size.

(1) Problem: $\overset{\text{走样}}{\text{Aliasing}}$(artifacts due to sampling)

a. 会出现摩尔纹.

b. Wagon Wheel Illusion(False Motion)

(2) Reason: Signals are changing too fast but $\overset{\text{采样}}{\text{sampled}}$ too slow.

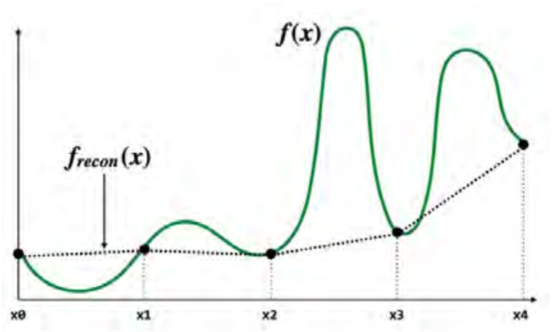


图 3.5: Aliasing

(3) Mathematically describe: $f = \frac{1}{T}$

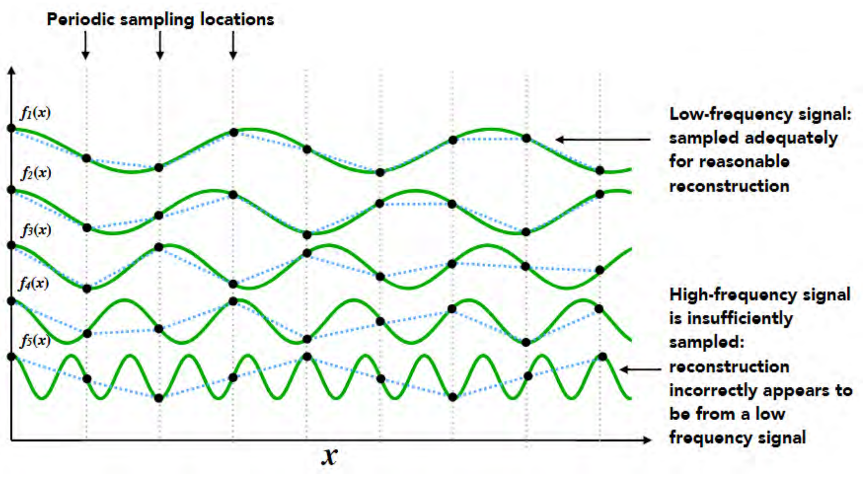


图 3.6: Frequencies & Sampling

Higher Frequencies Need Faster Sampling.

3. Fourier Transform

Represent a function as a weighted sum of sines and cosines. To obtain the frequencies of arbitrary signals.

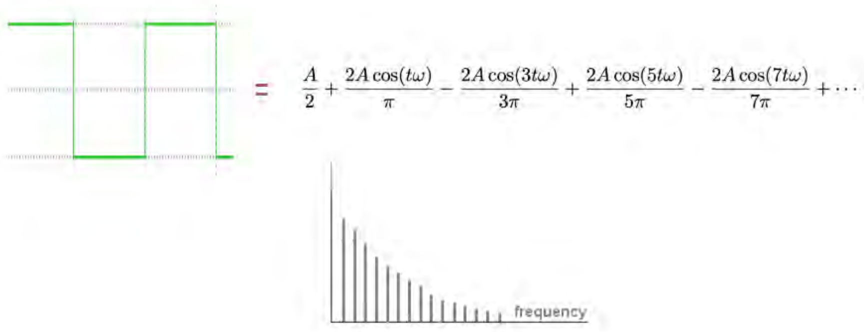

$= \frac{A}{2} + \frac{2A\cos(t\omega)}{\pi} - \frac{2A\cos(3t\omega)}{3\pi} + \frac{2A\cos(5t\omega)}{5\pi} - \frac{2A\cos(7t\omega)}{7\pi} + \cdots$
frequency

图 3.7: Fourier

$$x : \text{space}, u : \text{frequency}, e^{i\theta} = \cos\theta + i\sin\theta$$

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi ux}\,dx$$
$$f(x) = \int_{-\infty}^{\infty} F(u)e^{i2\pi ux}\,du$$

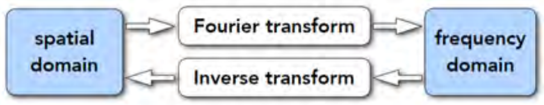
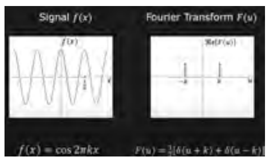


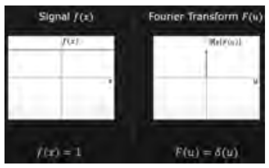
图 3.8: Fourier Transform

对 Fourier Transform 的频率而言, 中间频率低, 两边频率高.
频率大小对应在这个频率下 sin cos 与原图像拟合的多少? 也可以看作原图像变化的速度.
频域信号 $F(u)$ 代表原来的信号在 u 频率上的大小.
二维频谱 $F(u,v)$, 代表原来信号沿 x,y 两个方向分别在 u,v 频率上的大小. (并不与原坐标相对应)

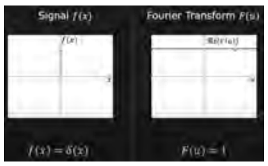
图 3.9: Examples(一维)



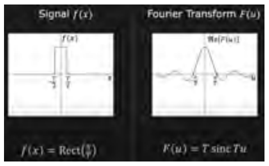
(a) cosinusoids
(余弦信号)



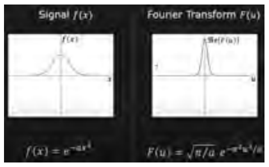
(b) Constant function
(常数)



(c) Dirac function
(δ, 冲击信号)



(d) Box function
(窗口)



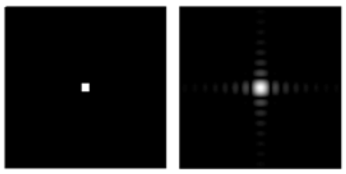
(e) Gaussian
(二者都是 Gaussian)

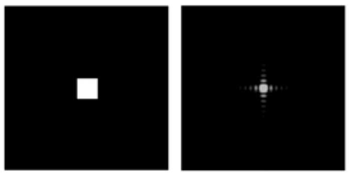
Convolution Theorem(卷积定理)

$$\begin{array}{ccc} \overset{\text{空间域}}{\text{Spatial Domain}} & & \overset{\text{频域}}{\text{Frequency Domain}} \\ \underbrace{g(x)=f(x)*h(x)}_{\text{Convolution}} & \Longleftrightarrow & \underbrace{G(u)=F(u)H(u)}_{\text{Multiplication}} \\ \underbrace{g(x)=f(x)h(x)}_{\text{Multiplication}} & \Longleftrightarrow & \underbrace{G(u)=F(u)*H(u)}_{\text{Convolution}} \end{array}$$

二维也成立

图 3.10: Examples(二维)


(a) Box filter=low-pass filter (均值滤波器)


(b) Box filter (Wider kernel=lower frequency)

4. Sampling

Sampling a signal = multiply the single by a Dirac $\overset{\text{脉冲信号}}{\text{comb}}$ function.

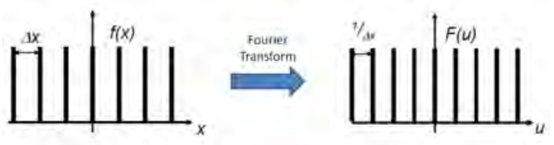


图 3.11: a Dirac comb function

Sampling = Repeating Frequency Contents.

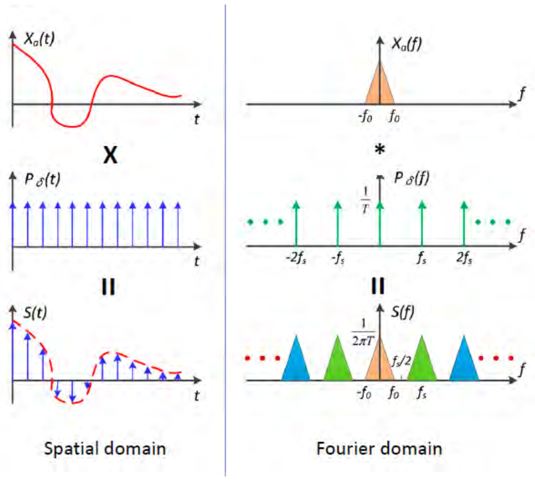


图 3.12: Sampling

Aliasing = Mixed Frequency Contents(频谱混叠)

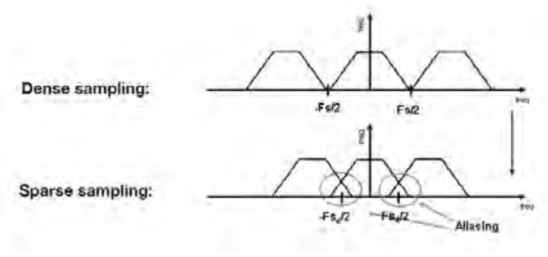


图 3.13: Aliasing

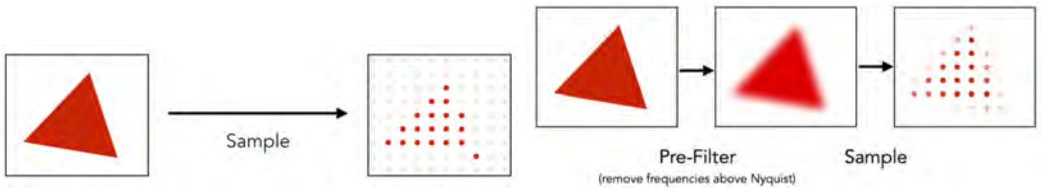
可将高频信号过滤后采样

5. Nyquist-Shannon theorem(奈奎斯特采样定理)

由此可计算采样最小的频率.
Consider a band-limited signal: has no frequencies above f_0.
The signal can be perfectly reconstructed if sampled with a frequency larger than $2f_0$.
由此 Anti-alisaing = Filtering, then Sampling.

(1) Convolve the image with low-pass filters (e.g. Gaussian)

(2) Sample it with a Nyquist rate



(a) Regular Sampling　　(b) Anti-aliased Sampling

三、 Image magnification

1. Interpolation(插值)

一维

(1) Nearest-neighbor interpolation(最近邻)

Not continuous, not smooth

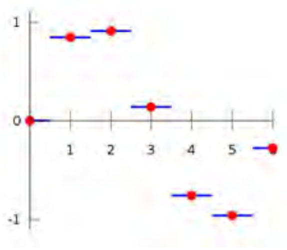


图 3.15: Nearest-neighbor interpolation

(2) Linear interpolation(线性)

Continuous, not smooth

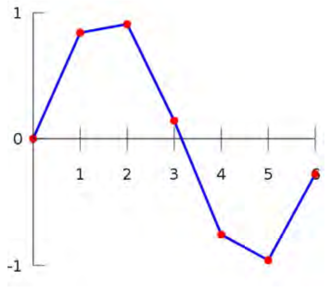


图 3.16: Linear interpolation

(3) Cubic interpolation(三次多项式, 三次样条)

Continuous, smooth

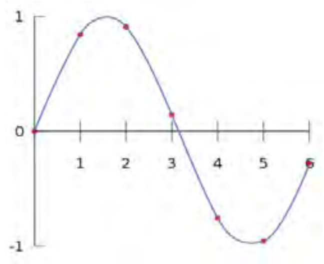


图 3.17: Cubic interpolation

For each interval: $y = ax^3 + bx^2 + cx + d$(每一段一个多项式)

二维

(1) Bilinear Interpolation

将二维插值转换为几次一维的插值, 一般用这个, generally bilinear is good enough

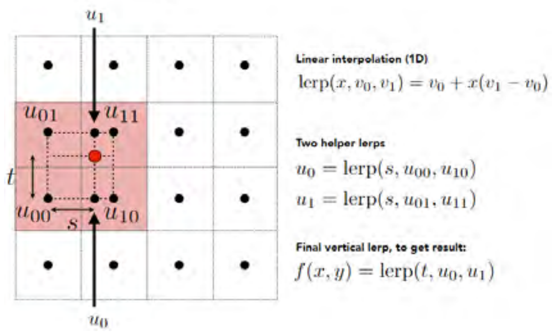

u_1
u_{01}
u_{11}
t
u_{00}
s
u_{10}
u_0
Linear interpolation (1D)
$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$
Two helper lerps
$u_0 = \text{lerp}(s, u_{00}, u_{10})$
$u_1 = \text{lerp}(s, u_{01}, u_{11})$
Final vertical lerp, to get result:
$f(x, y) = \text{lerp}(t, u_0, u_1)$

图 3.18: Bilinear interpolation

(2) Bicubic Interpolation

$p(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$

Super-Resolution(超分辨率): 神经网络

2. Seam Carving for Content-Aware Image Resizing(有内容的裁剪)

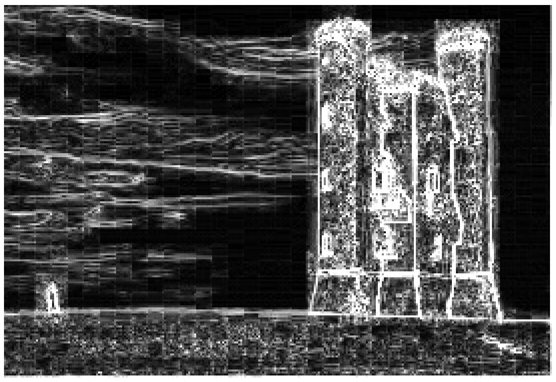
(1) Basic idea: (裁掉变化小的地方)remove unimportant pixels

(2) Important of pixel: edges are important. Egde energy $E(I) = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|$

方式: 用边缘检测滤波做卷积 (相当于获得梯度) 后相加, 即可得到 Egde energy 图像.

eg:

(a) Egde energy pre



(b) Egde energy

(3) Seam carving: Find connected path of pixels from top to bottom of which the edge energy is minima(除去的是连续的一条曲线, 且梯度最小)

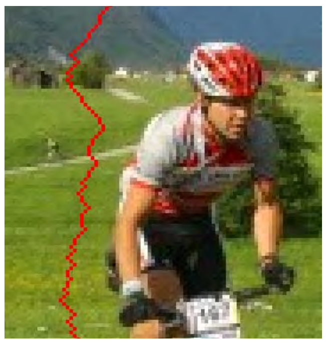


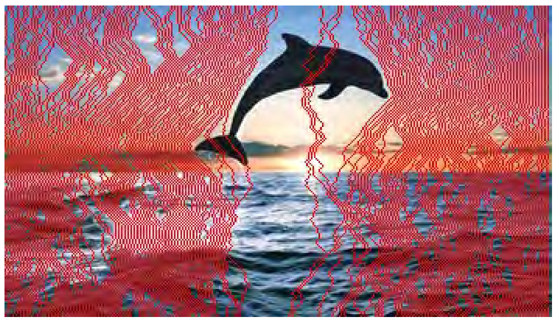
图 3.20: seam

方式: DP $M(i,j)$ 代表从上到下经过这个点的最小梯度, 有 $M(i,j) = E(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$

3. Seam insertion(接缝插值)

Find k seams to insert. Then interpolate pixels.

eg:


(a) Seam insertion pre

(b) Seam insertion

第四章　Model Fitting and Optimization

一、　Optimization(优化)

$$\begin{aligned}\text{minimize } & f_0(x)\\ \text{subject to } & f_i(x) \le 0, i = 1, \cdots, m\\ & g_i(x) = 0, i = 1, \cdots, p\end{aligned}$$

(1) $x \in R^n$(vector) is $\overset{\text{优化变量}}{\text{variable}}$ to be chosen.

(2) $f_0(x)$ is the $\overset{\text{目标函数}}{\text{objective}}$ function to be minimized.

(3) $f_1, \cdots, f_m$ are the inequality $\overset{\text{不等式约束条件}}{\text{constraint}}$ functions.

(4) $g_1, \cdots, g_m$ are the equality $\overset{\text{等式约束条件}}{\text{constraint}}$ functions.

eg: Image deblurring.

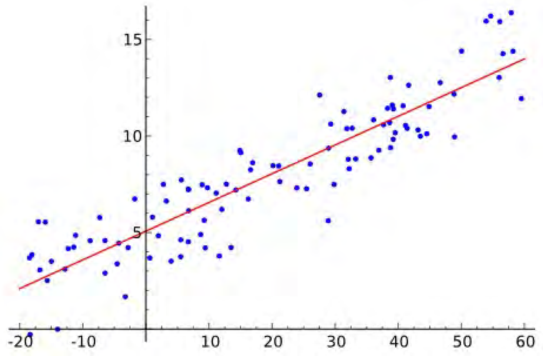
$$\begin{aligned}&\min_X \|Y - F * X\|_2^2 \text{(用范数衡量向量大小)}\\ &Y \text{ :Blurred image}\\ &F \text{ :Blur kernel}\\ &X \text{ :Sharp image}\end{aligned}$$

二、　Model fitting

1.　Model

A model describes the relationship between input and output(输入与输出的关系).
eg: linear model

$$\begin{aligned}&b = a^T x\\ &a \text{ is input}\\ &b \text{ is output}\\ &x \text{ is model parameter(参数)}\end{aligned}$$



2. Model fitting(Learning)

To estimate model parameters from data.

A typical approach: Minimize the $\overset{\text{最小均方误差}}{\text{Mean Square Error}}$(MSE).

$$\hat{x} = \arg\min_x \sum_i (b_i - a_i^T x)^2$$

Why use MSE?

3. Maximum Likelihood Estimate(MLE, 极大似然估计)

参数能最大化这组数据出现的概率.(目标函数是用噪声分布推导而出的)

(1) 假设数据噪声是 Gaussian 分布.(最常用的是 Gaussian, 因为各种随机变量的和可以证明是一个 Gaussian? 所以在噪声未先验的情况下 Gaussian 最保险)

$$b_i = a_i^T x + n, n \sim G(0, \sigma)$$

(2) 给定 x, 观测到 (a_i, b_i) 的可能性为:

$$P[(a_i, b_i)|x] = P[b_i - a_i^T x] \propto \exp(-\frac{(b_i - a_i^T x)^2}{2\sigma^2})$$

P 就是 likelihood(可能性, 不是概率)

(3) 给很多点

$$\begin{aligned} & P[(a_1, b_1)\ldots|x] \\ =& \prod_i P[(a_i, b_i)|x] \\ =& \prod_i P[b_i - a_i^T x] \\ \propto & \exp(-\frac{(b_i - a_i^T x)^2}{2\sigma^2}) = \exp(-\frac{\|Ax-b\|_2^2}{2\sigma^2}) \end{aligned}$$

(4) MLE= 寻找最大化 $P[(a_1, b_1)\ldots|x]$ 的 x, 即

$$\begin{aligned} \hat{x} &= \arg\max_x P[(a_1, b_1)\ldots|x] \\ &= \arg\max_x \exp(-\frac{\|Ax-b\|_2^2}{2\sigma^2}) \\ &= \arg\min_x \|Ax-b\|_2^2 \end{aligned}$$

(5) MSE=MLE with Gaussian noise assumption(高斯假设下的 MLE)

三、 Numerical methods

1. Problems have analytical solution

对有解析解的问题
eg:Linear MSE
求导, 让导数等于 0. 但矩阵求导比较复杂, 需要网上去搜 ().

$$\hat{x} = \arg\min_x \|Ax - b\|_2^2$$

对上式求导:

$$\hat{x}' = A^T(Ax - b)$$

令其为 0, 有:

$$A^T Ax = A^T b$$

求解即得 $\hat{x}$

2. No analytical solution

使用数值方法: 梯度下降 (Gradient descent).

$$F(x_0) > F(x_1) > \cdots > F(x_k) > \ldots$$

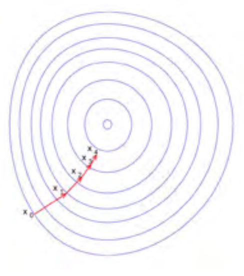


图 4.1: Gradient descent

Algorithm 1 Gradient descent

```
1: $x \leftarrow x_0$                                        ▷ Init
2: while not converge do
3:     h $\leftarrow$descending_direction($x$)                ▷ determine the direction
4:     $\alpha$ $\leftarrow$descending_step($x, h$)          ▷ determine the step
5:     x $\leftarrow x + \alpha h$                          ▷ update the parameters
6: end while
```

So how to find h and α

2.1 Taylor expansion

(1) First-order approximation

$$F(x_k + \Delta x) \approx F(x_k) + J_F \Delta x$$
$$\mathbf{J} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

J_F is Jacobian matrix, also as first-order derivative.

(2) Second-order approximation

$$F(x_k + \Delta x) \approx F(x_k) + J_F \Delta x + \frac{1}{2}\Delta x^T H_F \Delta x$$
$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

H_F is Hessian matrix, also as second-order derivative.

2.2 Gradient descent(GD)

假设目标函数近似为:

$$F(x_k + \Delta x) \approx F(x_k) + J_F \Delta x$$

当 $J_F \Delta x < 0$ 时, 目标函数会变小.
Steepest descent method(最速梯度下降法)
当 Δx 与 J_F(一阶近似) 方向完全一致, 即 $\Delta x = -J_F^T$ 时, 下降最快.
Determine the step size:

(1) Exact line search(一个个去试)

(2) Backtracking algorithm(回溯法)(求一个可接受的解)

- Initialize α with a big value.
- Decrease α until $\Phi(\alpha) \le \Phi(0) + \gamma \Phi'(0)\alpha$

  γ is a parameter, $0 < \gamma < 1$

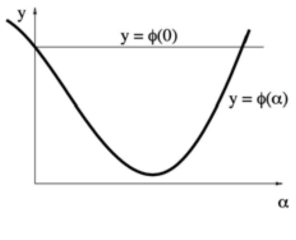


图 4.2: $\Phi(\alpha) = F(\mathrm{x} + \alpha \mathrm{h})$, x and h fixed, $\alpha \ge 0$

- Advantage: 简单, 较远时效果好.
- Disadvantge: 越接近收敛越慢, 计算浪费多.

2.3 Newton method

$$F(x_k + \Delta x) \approx F(x_k) + J_F \Delta x + \frac{1}{2} \Delta x^T H_F \Delta x$$

Find Δx to minimize $F(x_k + \Delta x)$,(求导)

$$H_F \Delta x + J_F^T = 0$$

The optimal direction(Newton step),

$$\Delta x = -H_F^{-1} J_F^T$$

- Advantage: 在最小点附近收敛更快
- Disadvantge: Hessian 需要更大计算量, 甚至无法计算

2.4 Gauss-Newton method(仅求解非线性最小二乘)

$$\begin{aligned} \hat{x} &= \arg\min_x F(x) \\ &= \arg\min_x \|R(x)\|_2^2 \end{aligned}$$

$$R(x) = \begin{bmatrix} b_1 - f_x(a_1) \\ b_2 - f_x(a_2) \\ \vdots \\ b_n - f_x(a_n) \end{bmatrix}$$

$R(x)$ is residual vector(残差向量)
展开 $R(x)$,

$$\begin{aligned} \|R(x_k + \Delta x)\|_2^2 &\approx \|R(x_k) + J_R \Delta x\|_2^2 \\ &= \|R(x_k)\|_2^2 + 2R(x_k)^T J_R \Delta x + \Delta x^T J_R^T J_R \Delta x \end{aligned}$$

The optimal Δx satisfies,(求导)

$$J_R^T J_R \Delta x + J_R^T R(x_k) = 0$$

J_R is the Jacobian of $R(x)$, $J_F^T = J_R^T R(x_k)$
The optimal direction,

$$\Delta x = -(J_R^T J_R)^{-1} J_R^T R(x_k)$$

Newton step: $\Delta x = -H_F^{-1} J_F^T = -H_F^{-1} J_R^T R(x_k)$
Gauss-Newton use $J_R^T J_R$ to approximate Hessian H_F.(只有目标函数为平方和时才可用)

- Advantage: 收敛快, 不需要计算 Hessian
- Disadvantge: 若 $J_R^T J_R$ 不满秩, 算法会不稳定.

2.5　Levenberg-Marquardt

LM employ regularization to overcome this

$$\Delta x = -(J_R^T J_R + \lambda I)^{-1} J_R^T R(x_k)$$

For all $\lambda > 0$, $J_R^T J_R + \lambda I$ must be positive-definite(正定)

(1) The effect of λ

- $\lambda \to \infty$: Gradient descent, and stepsize is small.
- $\lambda \to 0$: Gauss-Newton step.

(2) How to determine λ

- Init λ as a large number(Gradient descent)
- Update in every iteration
- When decreases obviously, $\lambda \downarrow$
- When does not decrease obviously, $\lambda \uparrow$

Advantage:

- 启动快 ($\lambda \uparrow$),
- 收敛快 ($\lambda \downarrow$),
- 不会退化 ($J_R^T J_R + \lambda I$ is always positive-definite),
- LM=Gradient descent + Gauss-Newton

2.6　Constrained Optimization(课外了解)

- Objective + constraints
- Different methods for different types of constraints

3.　Local minimum and global minimum

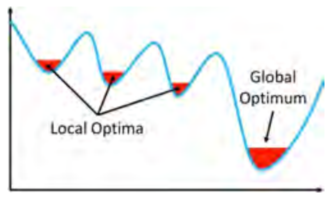


图 4.3: Local minimum and global minimum

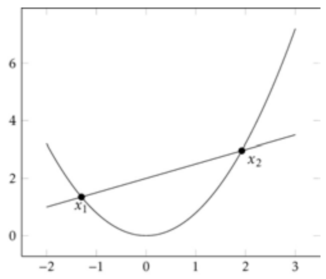


图 4.4: Convex optimization

凸函数与凸优化是非常重要的 (因为凸函数的局部最优就是全局最优), 把目标函数写成凸函数也有许多方法, 课外学习 ()

四、 Robust estimation(鲁棒估计)

1. Outliers(离群值)

- Inlier: obeys the model assumption
- Outlier: differs significantly from the assumption

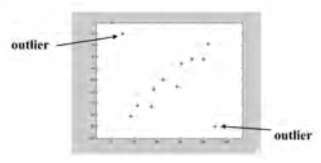


图 4.5: 离群值

2. Reduce the effect of Outliers

(1) Use other loss function to replace MSE, like L1, Huber. They are called robust functions.

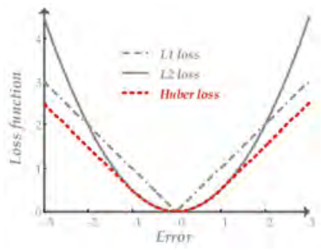


图 4.6: other loss function

(2) RANSAC: Random Sample Concensus(随机一致采样)

- The most powerful method to handle outliers.
- Key ideas:
  - The distribution of inliers is similar while outliers differ a lot.(好的数据的分布是近似的)
  - Use data point pairs to vote.(用点对连线进行投票积分, 求到最好的线后再用最小二乘跑一遍)

3. well-posed problem & ill-posed problem(适定问题与病态问题)

Well-posed problem is

(1) a solution exists;

(2) the solution is unique;

(3) the solution's behavior changes continuously with the initial conditions. (stable)

当上面三者有一不满足时, 就是 ill-posed problem.
For the solution is not unique, eg: $Ax = b$ when A is singular matrix.
How to make the soultion unique? Use prior knowledge to add more constraints(使用先验条件添加更多约束).

3.1 L2 regularization(正则)

(1) L2 norm: $\|x\|_2 = \sum_i x_i^2$

(2) L2 regularization:

$$\min_x \|Ax - b\|_2^2$$
$$s.t. \|x\|_2 \leq 1$$

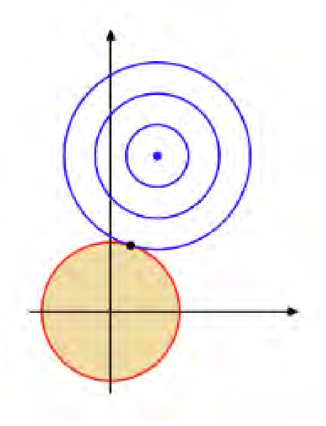


图 4.7: L2

- 把 x 往原点去拉.
- 让参数趋近于零, 以此抑制没必要的小参数.

3.2 L1 regularization

(1) L1 norm: $\|x\|_1 = \sum_i |x_i|$

(2) L1 regularization:

$$\min_x \|Ax - b\|_2^2$$
$$s.t. \|x\|_1 \leq 1$$

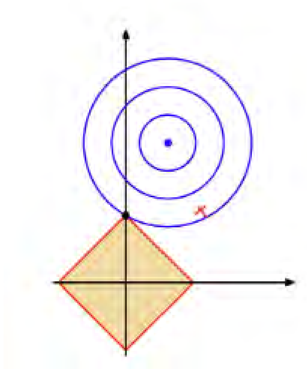


图 4.8: L1

- 有 L2 的优点.
- 让一些变量真的变为 0(稀疏, sparse).

4. Overfitting and underfitting(过拟合与欠拟合)

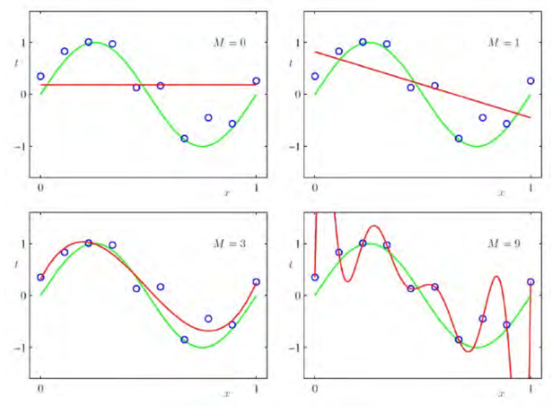


图 4.9: Overfitting and underfitting

五、 Graphcut and MRF

1. Image labeling problems

eg: Image segmentation(分割)

1.1 Images as Graphs

- A vertex for each pixel
- An edge between each pair
- Graph Notation: G=(V, E)
- Each edge is weighted by the affinity or similarity between its two vertices(边权以像素相似性定义)

1.2 Measuring affinity

Let i and j be two pixels whose features are f_i and f_j

- Pixel Dissimilarity

$$S(f_i, f_j) = \sqrt{\sum_k (f_i k - f_j k)^2}$$

- Pixel Affinity(亲和性)

$$w(i,j) = A(f_i, f_j) = e^{-\frac{1}{2\sigma^2} S(f_i, f_j)}$$

1.3 Graph cut

Cut $C = (V_A, V_B)$ is a partition of vertices V of a graph G into two disjoint subsets V_A and V_B.
Cost of Cut: Sum of weights of cut-set edges

$$cut(V_A, V_B) = \sum_{u \in V_A, v \in V_B} w(u,v)$$

让这个代价最小. 因为是离散的所以不能用上节课的优化方式.

min-cut 不过可以用最大流算法求最小割来解决.

Problem: 解不一定有意义 (如把一个点视为一个子图). 由此可见目标函数不够好.

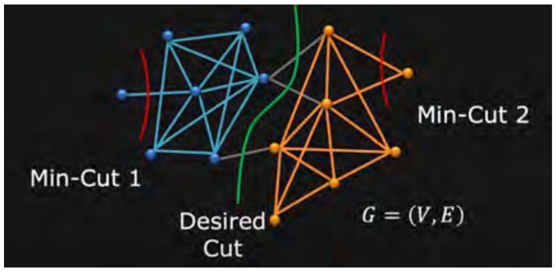


图 4.10: min-cut

Normalized cut Normalizing for size of segments

$$NCut(V_A, V_B) = \frac{cut(V_A, V_B)}{assoc(V_A, V)} + \frac{cut(V_A, V_B)}{assoc(V_B, V)}$$
$$assoc(V_a, V) = \sum_{u \in V_A, v \in V} w(u,v)$$

- 为 NP-Complete

- Approximate solution by eigenvalue decomposition.(但可以用特征值分解方法求近似解)

2. Markov Random Field

2.1 Markov chains

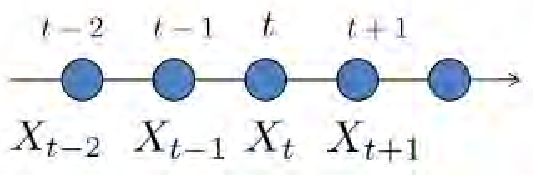


图 4.11: Markov chains

Markov chains 用来描述一个系统随时间推移的状态转移。

在 Markov chains 中，系统在每个时刻都处于某一个状态，并且下一个状态只与当前状态有关，不受之前状态的影响。

2.2 Markov Random Field

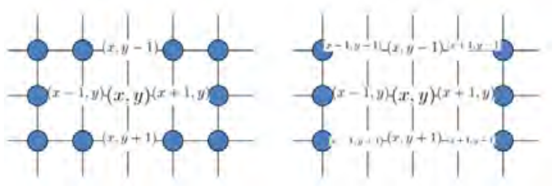


图 4.12: Markov random field

$$P(I_{x,y}|I_{-(x,y)}) = P(I_{x,y}|I_{N(x,y)})$$

$-(x,y)$ 为其他所有点, $N(x,y)$ 为与之相邻的点.
每个像素的连续性只取决于与之相邻的像素. 即相邻变量之间有约束.

2.3 Segmentation by MRF

Joint probability(联合概率分布)(此下 x,y 与上文 x,y 的意义并不相同)

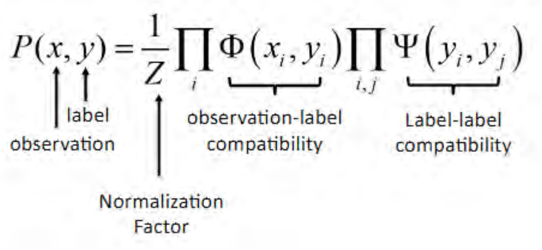

$P(x,y)=\frac{1}{Z}\prod_i \Phi(x_i,y_i)\prod_{i,j}\Psi(y_i,y_j)$
label
observation
observation-label compatibility
Label-label compatibility
Normalization Factor

图 4.13: Segmentation by MRF

- x 为观测
- y 为标签

- Z 为常数
- $\prod_i \Phi(x_i, y_i)$ 所有像素的某个量相乘, 描述每个像素标签与观测的关系.
- $\prod_{i,j} \Psi(y_i, y_j)$ 相邻像素标签之间的关系.

需要最大化这个值, 等效于最小化其-ln, 即
Energy funciton

$$E(x, y) = \sum_i \varphi(x_i, y_i) + \sum_{i,j} \psi(y_i, y_j)$$

(1) Unary potentials(单一) $\varphi(x_i, y_i)$
    - 每个像素自己标签的概率.

(2) Pairwise potentials(成对) $\psi(y_i, y_j)$
    - 成对像素标签中的一致性.

2.4 Example

(1) Unary potentials(单一) $\varphi(x_i, y_i) = (bool)(x_i \neq y_i)$
    - 像素是否与标签一致, 不一致给 loss

(2) Pairwise potentials(成对) $\psi(y_i, y_j) = (bool)(y_i = y_j)w_{ij}$
    - 标签相同给 loss

How to solve?

- 最大流求最小割 Maxflow/Min-Cut
- 优化问题

2.5 Advantage

很好用, 用的很广泛.

2.6 Disadvantages

多个标签只能近似求解.

第五章　Image Matching and Motion Estimation

一、　Image matching

找两个图像之间点与点的关系.

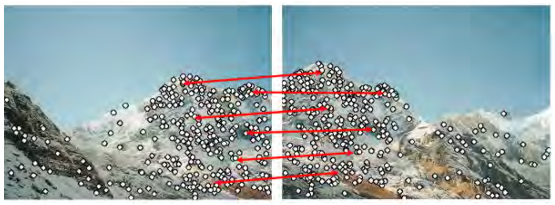


图 5.1: Image matching

Application: 几乎所有 CV 的问题都可以转换为此问题.

(1) Image alignment

(2) 3D reconstruction

(3) Motion tracking

(4) Object recognition

(5) Indexing and database retrieval

(6) Robot navigation

(7) ... other

Main Components of Feature matching

(1) Detection(检测): Identify the interest points.

(2) Description(描述): Extract(提取) vector feature descriptor(描述子) surrounding each interest point.

(3) Matching: Determine correspondence between descriptors in two views.

1.　Detection

寻找点的唯一性.

1.1 Local measures of uniqueness

衡量的是点的领域. 唯一性可以看作移动这个领域后的变化.

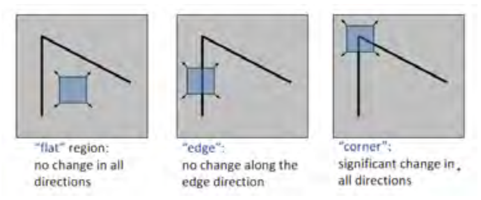

"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions

图 5.2: uniqueness

数学上可以用梯度衡量, 坐标描述的是领域中梯度的分布.

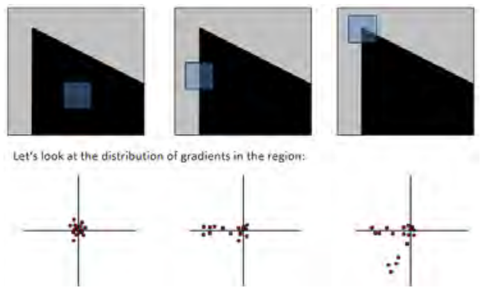


图 5.3: measure uniqueness

Principal Component Analysis(主成分分析) 第一个主成分是方差最大的方向, 第二个主成分是与第一个主成分垂直的方差最大的方向.

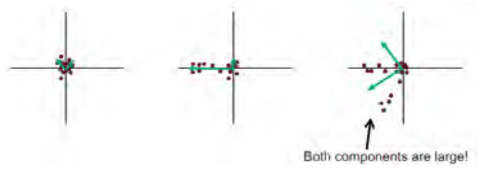


图 5.4: PCA

Corner detection

(1) Compute the covariance matrix(协方差矩阵) at each point.

$$H = \sum_{u,v} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ i_x I_y & I_y^2 \end{bmatrix}$$
$$I_x = \frac{\partial f}{\partial x}$$
$$I_y = \frac{\partial f}{\partial y}$$

$w(u,v)$ is typically Gaussian weights

(2) Compute eigenvalues(特征值).

$$H = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\lambda_{\pm} = \frac{1}{2}((a+d) \pm \sqrt{4bc + (a-d)^2})$$

(3) Classify points using eigenvalues of H:

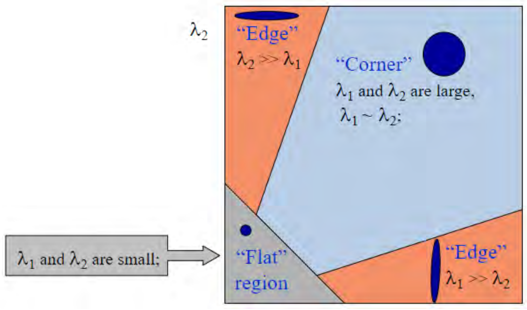

λ_2
"Edge"
$\lambda_2 >> \lambda_1$
"Corner"
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;
λ_1 and λ_2 are small;
"Flat" region
"Edge"
$\lambda_1 >> \lambda_2$

图 5.5: Corner detection

The Harris operator 算特征值开销太大, Harris corner detector 用以来代替

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{det(H)}{trace(H)}$$

$$det(\begin{bmatrix} a & b \\ c & d \end{bmatrix}) = ad - bc\text{为行列式}$$

$$trace(\begin{bmatrix} a & b \\ c & d \end{bmatrix}) = a + d\text{为迹, 即对角线之和 (需对称矩阵)}$$

f 叫做角点检测的响应值 (corner response)
Harris detector

(1) Compute derivatives(梯度/导数) at each pixel

(2) Compute covariance matrix H(协方差矩阵) in a Gaussian window around each pixel

(3) Compute corner response(响应值) function f

(4) Threshold f

(5) Find local maxima of response function(nonmaximum suppression)

但要如何保证两幅图检测出来点的重复性 (repeatable)?
因此想要 f 对图像的变换具有不变性 (invariant).

Harris detector: Invariance properties image transformations

(1) photometric: 亮度变换

(2) Geometric

a. 平移

b. 旋转

c. 缩放

Harris detector(讨论对应点的响应值)

(1) photometric transformation $I \longrightarrow aI + b$

a. $I \longrightarrow I + b$ 梯度不变, f 不变

b. $I \longrightarrow aI$ 梯度改变, f 改变

所以是部分改变

(2) Image translation(平移) f 不变

(3) Image rotation(旋转) f 不变

(4) Scaling(缩放) f 改变 (图像变大, 领域中梯度改变了)

如何正确选择窗口大小? 逝 ()
不同窗口扫下来, 最大 f 应该一样, 但尺度不同, 所以选最大的. 用 image pyramid 缩小图像, 用相同的领域去扫这个 pyramid, 以此确定最好的尺度.

1.2 Blob detector(斑点检测)

斑点有较大的二阶导数 (极值点)
用拉普拉斯算子 (Laplacian)$\Delta = \nabla^2$, $I_{xx} + I_{yy}$ 卷积计算,

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

Laplacian of Gaussian 因为其对噪声十分敏感, 所以用卷积核 Laplacian of Gaussian (LoG) filter 来计算, 以抑制噪声.

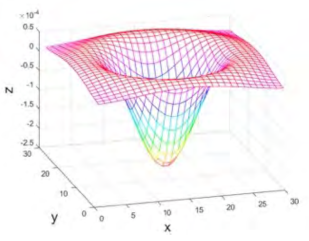


图 5.6: Laplacian of Gaussian

$$\nabla^2(f * G) = f * \nabla^2 G$$
$$\therefore \text{use LoG } \nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$
$$G \text{ is 2D Gaussian function}$$

LoG 尺度 (scale) 取决于 Gaussian 的 σ.

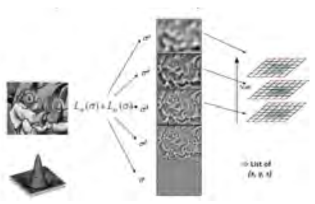


图 5.7: Scale of LoG

尺度选择: 用不同 σ 去试 (), 也是找最大的 f 值的 σ.

Difference-of-Gaussian(DoG) $\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$(LoG 可用两个 Gaussian 的差作近似)

Best approximation when $\sigma_1 = \frac{\sigma}{\sqrt{2}}, \sigma_2 = \sqrt{2}\sigma$.

不同尺度下, 用不同 σ 计算再两两相减得出.

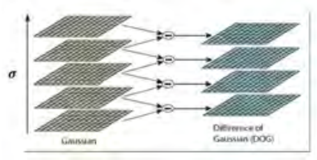


图 5.8: Scale of DoG

DoG 一般比 LoG 高效 (Gaussian 比较快)

2. Description

2.1 Feature descriptors(构造描述子)

一般是个 vector(向量)

Describe an image patch 描述像素块, 直接把一个像素块表示为一个向量. (这并不好, 因为其不变性非常差.)

SIFT descriptor Scale Invariant Feature Transform (SIFT), 最常用. 本质上是对 patch 内图像梯度的统计. SIFT 就是用 patch 内梯度朝向分布作描述子.

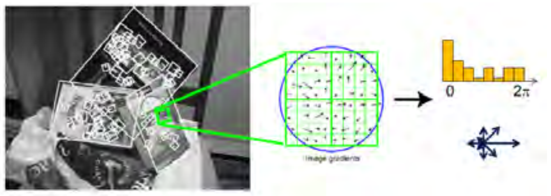


图 5.9: SIFT

不变性

(1) 对亮度有较好不变性

(2) 缩放有不变性 (因为其特殊检测阶段的手段确定了缩放不变, 而不是其本身有不变性)

(3) 旋转有不变性 (Orientation Normalization, 对朝向进行了归一化)

a. Compute orientation histogram(求朝向)

b. Select dominant orientation(选择主朝向)

c. Normalize: rotate to fixed orientation(归一化)

流程

(1) Run DoG detector

a. Find maxima in location/scale space

b. Remove edge points

(2) Find dominate orientation(朝向归一化)

(3) For each (x,y,scale,orientation), create descriptor

2.2 一些其他检测器与描述子 detectors and descriptors

SIFT 已经是传统的效果的极限了 (不传统指神经网络)
其他有些是为了更快之类的.

(1) HOG: Histogram of oriented gradients

- Dalal & Triggs, 2005

(2) SURF: Speeded Up Robust Features

- Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, ”SURF: Speeded Up Robust Features”, Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008

(3) FAST (corner detector)

- Rosten. Machine Learning for High-speed Corner Detection, 2006.

(4) ORB: an efficient alternative to SIFT or SURF

- Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011

(5) Fast Retina Key- point (FREAK)

- A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. CVPR 2012 Open Source Award Winner.

3. Matching

3.1 Feature distance

用距离衡量相似 $\|f_1 - f_2\|$, 但如果存在重复会 WA.

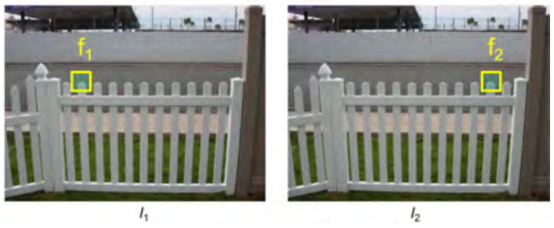


图 5.10: Feature distance

Ratio test Ratio score=$\frac{\|f_1-f_2\|}{\|f_1-f_2'\|}$, 奇异点比值为 1, 这个点就不太好, 那就不要了.

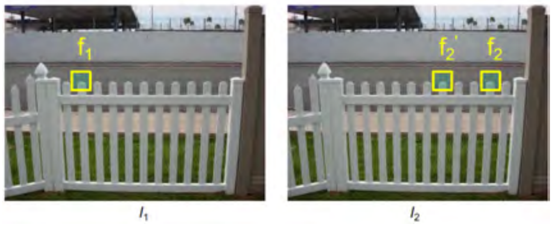


图 5.11: Ratio test

Mutual nearest neighbor (双向最近邻) 双向寻找, 都是最接近的才是最接近的. 如果两次不一样, 那就不要这个点了.

4. Deep learning for feature matching

训练网络来解决 (), 后面讲.

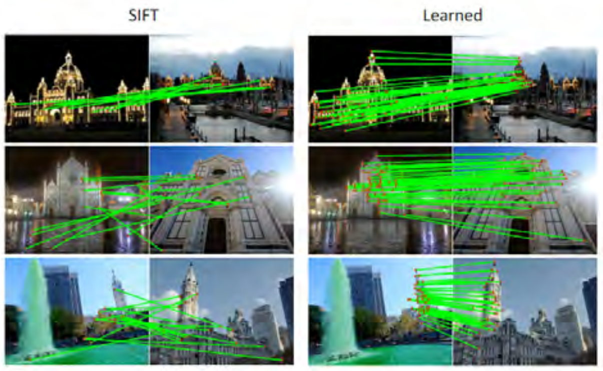


图 5.12: SIFT vs Learned

二、 Motion Estimation

视频中相邻帧的运动关系.

1. Seeing motion from a static picture

一些奇怪的图, 看着图在动, 其实是人眼在动, 导致看到的图在动. 一般运动都是看亮度有没有改变.

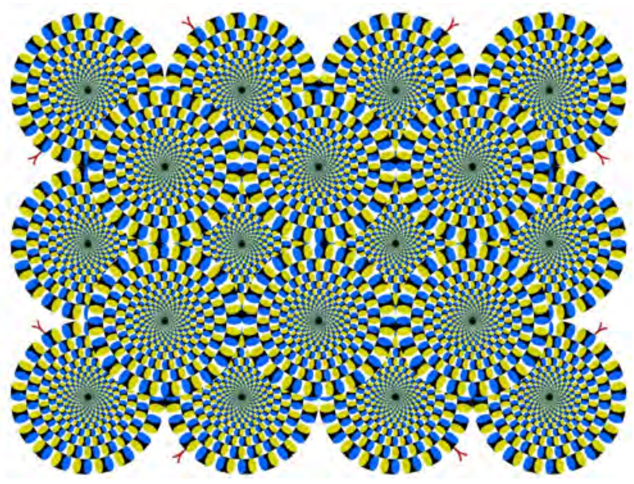


图 5.13: a static picture

2.　The cause of motion

(1) 场景中物体运动.

(2) 相机运动.

(3) 相机与场景一起运动.

(4) 场景相机都不动, 但光线变化.(会影响运动检测)

3.　Motion estimation problems

(1) Feature-tracking: 跟踪每个特征点在每一帧的位置. (稀疏)

(2) Optical flow(光流): 恢复图像中每一个像素的运动. (稠密, 本质上是个向量场, 表示像素的移动)

4.　Lucas-Kanade method

4.1　Motion estimation

把图像记为 $I(x, y, t)$.

有 $I(x, y, t)$ 与 $I(x, y, t+1)$, 求解位移. 即通过第一张图位置估计运动以此求解第二张图的位置, 本质上是求解每个点对应的平移向量.

4.2　Lucas-Kanade 的假设

(1) Small motion: points do not move very far(前后帧点运动较小)

(2) Brightness constancy: same point looks the same in every frame(前后帧亮度值不会有太大变化)

(3) Spatial coherence(空间连续性): points move like their neighbors(相邻点运动会比较一致)

4.3　求解

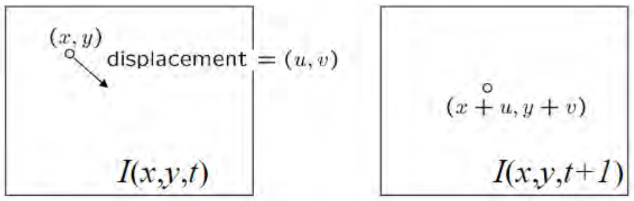


图 5.14: The brightness constancy

要求解 u 与 v

Brightness Constancy Equation:

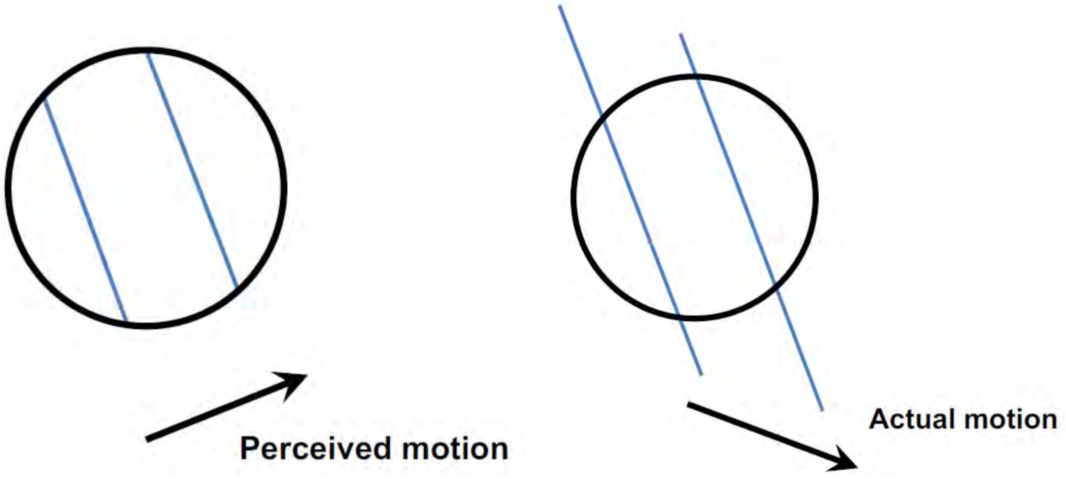
$$I(x, y, t) = I(x + u, y + v, t + 1)$$

Taylor expansion assuming small motion:(I_x 是图像关于 x 的导数, I_t 用关于时间的两次的变化做差得到)

$$\begin{aligned} I(x+u, y+v, t+1) &\approx I(x,y,t) + I_x u + I_y v + I_t \\ I(x+u, y+v, t+1) - I(x,y,t) &= I_x u + I_y v + I_t \\ \therefore I_x u + I_y v + I_t &\approx 0 \\ \therefore \nabla I \begin{bmatrix} u \\ v \end{bmatrix} + I_t &= 0 \end{aligned}$$

(方程数量不足, 那么方程无法确定在哪? 自由度在何处?) 若 (u, v) 满足此方程, $\exists u', v'$, 有 $\nabla I \begin{bmatrix} u' \\ v' \end{bmatrix} = 0$, 则 $(u + u', v + v')$ 也满足此方程. 所以垂直此方向的移动是确定不了的.

The aperture problem　孔径问题, 只看局部是确定不了运动的.



(a) pre　　(b) real

图 5.15: The aperture problem

Spatial coherence, 取周围的点联立建方程求解 u_i,v_i. 假设周围的点 (u, v) 相同. eg:5×5 共 25 点, 即

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$\underset{25\times 2}{A} \quad \underset{2\times 1}{d} = \underset{25\times 1}{b}$$

不一定要完全相等, 就是求 $\min_d \|Ad - b\|^2$, 即求解

$$(A^T A)d = A^T b$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ 为领域内梯度的协方差矩阵, 但当 $A^T A$ 不满秩无唯一解, 所以要求 λ_1 与 λ_2 不为 0, 最好都比较大, 这样求逆会稳定一些. (与 Harris corner detector 相似, 所以角点是比较好跟踪的点)

4.4　Errors in Lukas-Kanade

当假设被违反就会产生很大的误差

(1) Brightness constancy is not satisfied.(光照巨变)

(2) The motion is not small(有东西快速乱窜)

(3) A point does not move like its neighbors(边缘的点/特别小的物体)

Revisiting the small motion assumption　可以减小分辨率缓解此假设的违反.

Coarse-to-fine optical flow estimation　从粗到细的光流估计.

根据上一层光流估计的结果改变下一层让其更满足小运动假设. (Image warping)

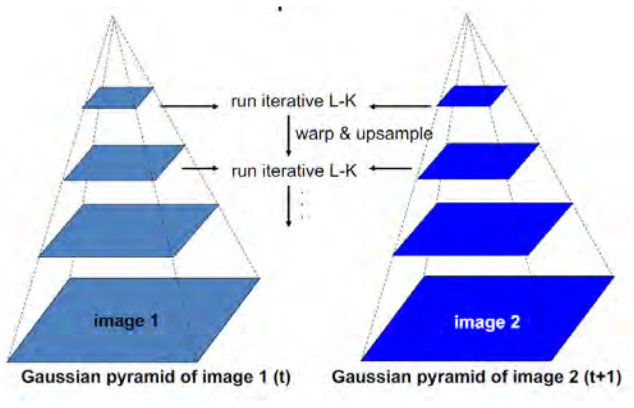

run iterative L-K
warp & upsample
run iterative L-K
image 1
image 2
Gaussian pyramid of image 1 (t)
Gaussian pyramid of image 2 (t+1)

图 5.16: Coarse-to-fine

5.　光流可视化

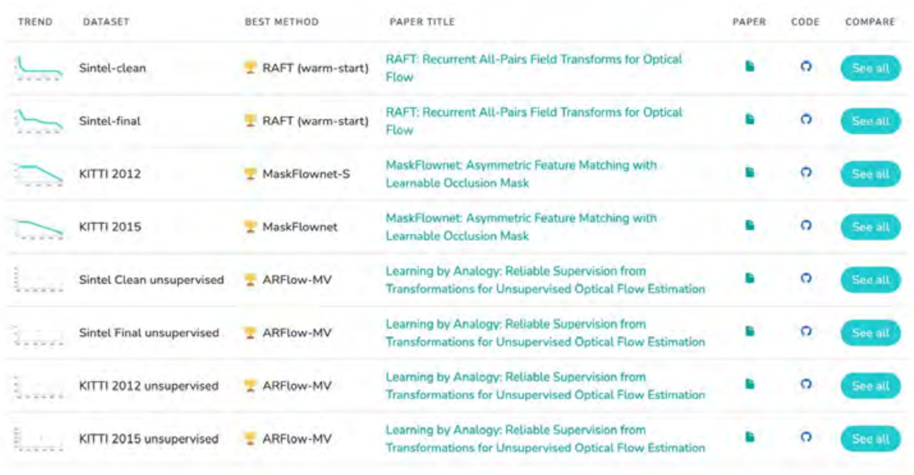
Flow quality evaluation

用 HVS 颜色表示光流的方向, 大小为强度.

常用数据集: Middlebury flow page

6. Leaderbord

来自 paperswithcode.com



TREND	DATASET	BEST METHOD	PAPER TITLE	PAPER	CODE	COMPARE
	Sintel-clean	RAFT (warm-start)	RAFT: Recurrent All-Pairs Field Transforms for Optical Flow	▪	○	See all
	Sintel-final	RAFT (warm-start)	RAFT: Recurrent All-Pairs Field Transforms for Optical Flow	▪	○	See all
	KITTI 2012	MaskFlownet-S	MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask	▪	○	See all
	KITTI 2015	MaskFlownet	MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask	▪	○	See all
	Sintel Clean unsupervised	ARFlow-MV	Learning by Analogy: Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation	▪	○	See all
	Sintel Final unsupervised	ARFlow-MV	Learning by Analogy: Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation	▪	○	See all
	KITTI 2012 unsupervised	ARFlow-MV	Learning by Analogy: Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation	▪	○	See all
	KITTI 2015 unsupervised	ARFlow-MV	Learning by Analogy: Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation	▪	○	See all

图 5.17: Leaderbord

虽用 DL, 但思想来自经典方法.

7. 应用

(1) Video stabilization(去抖动, 因为所有像素的运动多半是相机运动)

(2) Video 压缩 (用光流压缩信息)

(3) Video denoising(去噪, 把滤波扩展到时序)

第六章　Image stitching

图像拼接应用

(1) Panorama(全景图)

(2) 360 VR

一、　Image Warping

一种图像变换.

- Image filtering: 改变强度
- Image warping: 改变形状

1.　Parametric (global) warping

一种全局坐标变换.

$$p' = T(p)$$

(1) Linear Transforms = Matrices (线性变换)

(2) Affine Transformatio (仿射变换)

a. Affine map = linear map + translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

b. Using homogenous coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

(3) Projective Transformation (投影变换)(Homography)(单应)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$ 通常范数为 1, 所以其一共有 8 个自由度, 这种问题称之为 up to scale.

其描述为相机平面的改变. Change projection plane (pp).

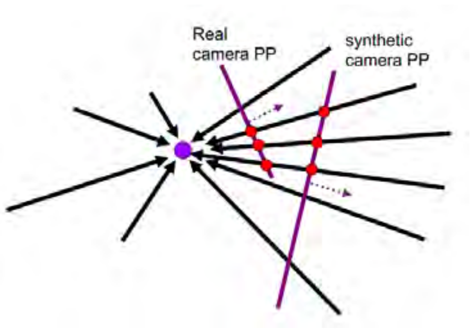


图 6.1: Change projection plane (pp)

Homography 仅有的情况

a. 相机中心不动, 相机转. (所以拍全景图只能转)

b. 相机中心移动, 但场景是个平面.

(4) Summary of 2D transformations

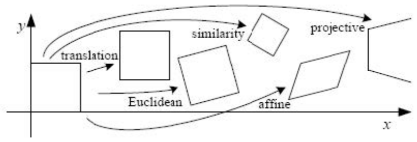


图 6.2: 2D transformations

(5) Inverse Transform

$$T^{-1}$$

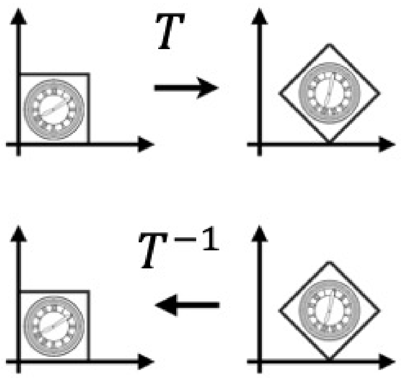


图 6.3: Inverse Transform

2. Implementing image warping

实现图像翘曲. Give a coordinate transform $(x', y') = T(x, y)$, and a source image $f(x, y)$.

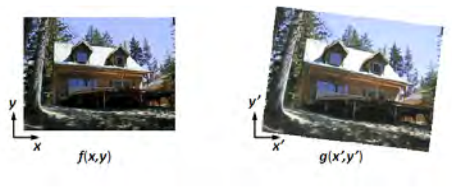


图 6.4: image warping

2.1 Forward Warping

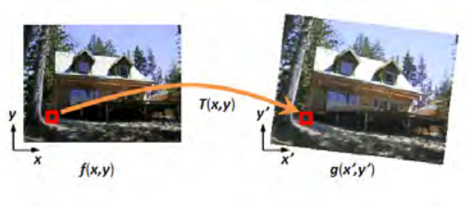


图 6.5: Forward Warping

如果算得的值非整数就强行取整, 问题: 有些地方没值.

2.2 Inverse Warping

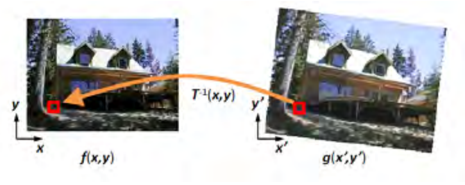


图 6.6: Inverse Warping

反着做保证有值, 如果没值可以插值. (这种高效一点)

Interpolation

(1) nearest neighbor

(2) bilinear(一般这个就够了)

(3) bicubic

(4) sinc

二、 Image Stitching

1. Compute transformation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

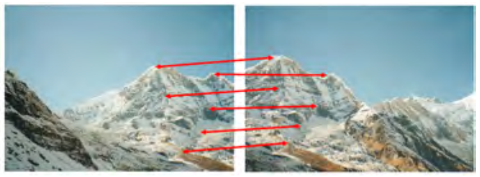


图 6.7: Image Stitching

(1) Image matching(每对点有个等式)

(2) 求解 T

1.1 Affine transformations

每队点两个方程, 共 6 自由度, 所以要三个方程.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax+by+c \\ dx+ey+f \\ 1 \end{bmatrix}$$

即

$$\begin{aligned} x' &= ax+by+c \\ y' &= dx+ey+f \end{aligned}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

对 n 对点 (n matches)

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

$$\underset{2n\times 6}{A} \quad \underset{6\times 1}{t} = \underset{2n\times 1}{b}$$

如果 $n \geq 3$, 寻找 t, 有最小 $\|At-b\|^2$, 即

$$A^TAt = A^Tb$$
$$t = (A^TA)^{-1}A^Tb$$

1.2　Projective transformations

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

即

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$
$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$
$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

For n matches,

$$\underset{2n\times 9}{\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{A}} \underset{9\times 1}{\underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{t}} = \underset{2n\times 1}{\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{b}}$$

且 $\|h\| = 1$
如果 $n \geq 4$, 寻找 h, 有最小 $\|Ah-0\|^2$, 即寻找 $\hat{h}$, $\hat{h} = (A^TA)$ 的最小特征值对应的特征向量.

2. RANSAC

(1) Randomly choose s samples

Typically s = minimum sample size that lets you fit a model

(2) Fit a model (e.g., transformation matrix) to those samples

(3) Count the number of inliers that approximately fit the model

(4) Repeat N times

(5) Choose the model that has the largest set of inliers

(6) least squares fit, find average translation vector over all inliers

(7) 对接缝进行处理, 找接缝附近变化最小的点 (Seam insertion)

3. Panorama

3.1 create

(1) Warp all images to a reference image

(2) merge them

但越边缘越小.

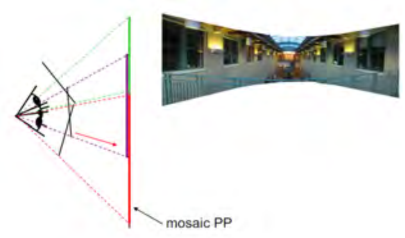


图 6.8: Panorama

3.2 Cylindrical Panoramas

可以以相机为中心构造圆柱面, 把其投影到柱面上.

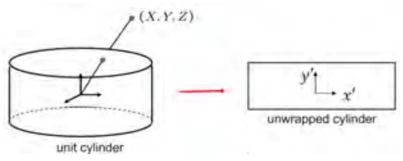


图 6.9: Cylindrical projection

$$x' = r\tan^{-}1(\frac{x}{f})$$
$$y' = \frac{ry}{\sqrt{x^2+f^2}}$$

(x', y') 为柱面坐标, (x, y) 为平面坐标, r 为柱半径, f 为焦距, 其坐标原点都已移至图像中心. 且相机旋转在柱面上等效于平移.

Problem: Drift 误差会累计.

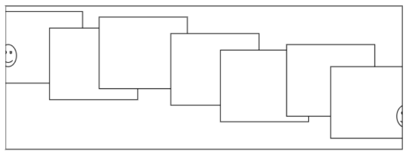


图 6.10: Drift

将首尾特征匹配修正, 转换为优化问题.

(1) small (vertical) errors accumulate over time

(2) apply correction so that sum = 0 (for 360° pan.)

因为投影到了柱面, 所以直线会边弯.

第七章　Structure from Motion

通过相机运动重建物体.

Problems tobe noticed

(1) How the camera maps the 3D points in the world onto its image plane. (camera model)(如何把三维坐标映射为二维坐标)

(2) How tocompute the position and orientation of the camera w.r.t. the world coordinate frame. (camera calibration and pose estimation)(如何恢复相机参数, 位姿)

(3) How to reconstruct the unknown 3D structure from images. (structure from motion)(恢复场景)

一、　Camera model

几何变换

1. Image Formation

(1) 相机定位 (Coordinate Transformation, 坐标变换, 从世界坐标系到相机坐标系)

(2) 快门 (Perspective Projection, 透视投影变换)

(3) 图像形成 (Image Plane to Image Sensor Mapping, 从物理坐标到像素坐标)

- Extrinsic Matrix(外参): Coordinate Transformation
- Intrinsic Matrix(内参): Perspective Projection + Image Plane to Image Sensor Mapping

1.1　Coordinate Transformation

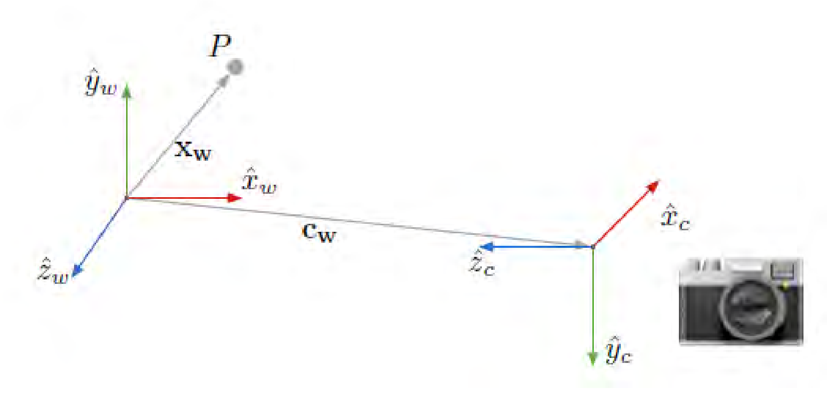


图 7.1: Coordinate Transformation

$$x_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \Longrightarrow x_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

需要知道相机位置与朝向 (位姿). Position c_w 为世界中心到相机中心的向量, Orientation R 为旋转矩阵, 且 R 为正交矩阵.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{array}{l} \longrightarrow \text{Row 1: Direction of } \hat{x}_c \text{ in world coordinate frame} \\ \longrightarrow \text{Row 2: Direction of } \hat{y}_c \text{ in world coordinate frame} \\ \longrightarrow \text{Row 3: Direction of } \hat{z}_c \text{ in world coordinate frame} \end{array}$$

现在对一世界坐标 P, 其坐标为 x_w, 有: (即旋转后平移一下)

$$x_c = R(x_w - c_x) = Rx_w + t$$
$$t = -Rc_w$$

$$x_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$\tilde{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\therefore \text{外参矩阵} M_{ext} = \begin{bmatrix} R_{3\times 3} & \text{t} \\ 0_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.2　Perspective Projection

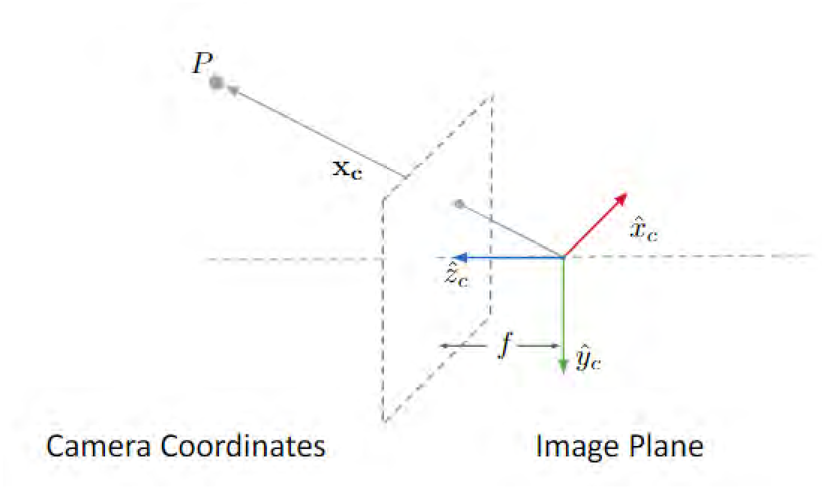


图 7.2: Perspective Projection

$$x_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \Longrightarrow x_i = f \cdot \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{bmatrix}$$

1.3 Image Plane to Image Sensor Mapping

从毫米坐标变为像素坐标. 需要分辨率.

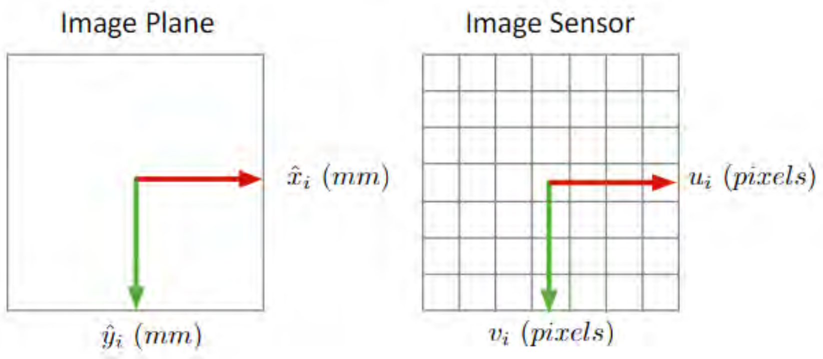


图 7.3: Image Plane to Image Sensor Mapping

m_x 与 m_y 为方向上的分辨率.

$$u = m_x x_i = m_x f \frac{x_c}{z_c}$$
$$v = m_y y_i = m_y f \frac{y_c}{z_c}$$

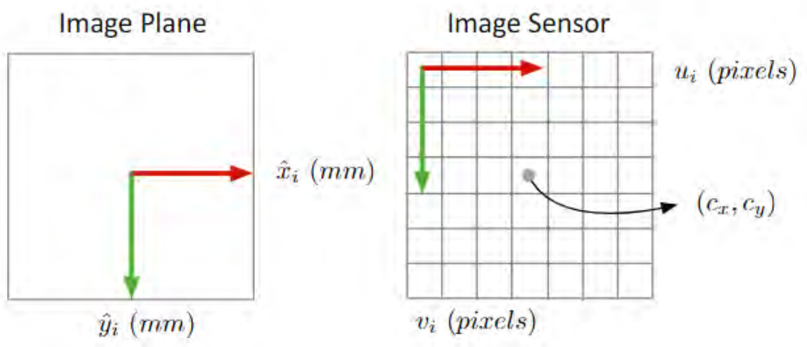


图 7.4: Principle Point

但二者原点不一样, 需要有偏移量 (c_x, c_y)(一般情况是图像中心, 但如果图像被裁切过, 图像中心会改变)

$$u = m_x f \frac{x_c}{z_c} + c_x$$
$$v = m_y f \frac{y_c}{z_c} + c_y$$

综上所述, (令 $f_x = m_x f, f_y = m_y f$)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$\therefore \text{内参矩阵} M_{int} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2. Projection Matrix P

再综上所述, 最终的投影矩阵 P:

$$\tilde{x}_c = M_{ext}\tilde{x}_w \text{ World toCamera}$$
$$\tilde{u} = M_{int}\tilde{x}_c \text{ Camera to Pixel}$$
$$\therefore \tilde{u} = M_{int}M_{ext}\tilde{x}_w = P\tilde{x}_w$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

二、 Camera calibration

相机标定

1. Camera Calibration Procedure

(1) Capture an image of an object with known geometry. (拍摄一已知物体, 已知角点世界坐标)

(2) Identify correspondences between 3D scene points and image points. (检测与匹配, 得到角点在图像中二维坐标)

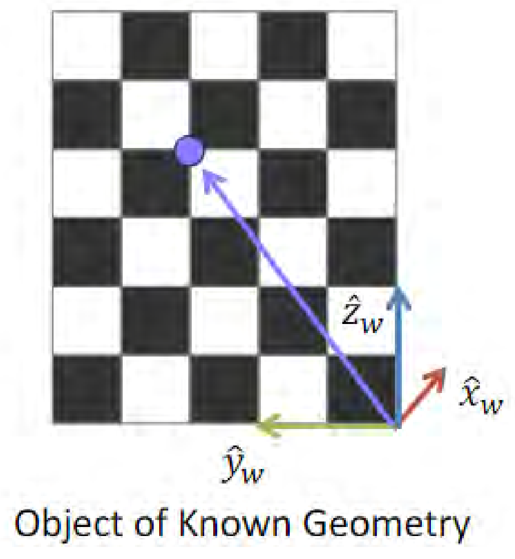


图 7.5: 标定板

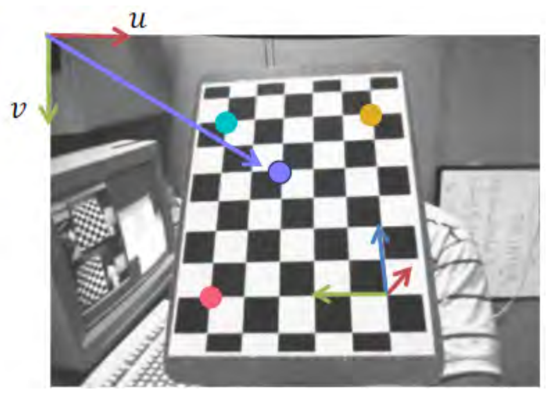


图 7.6: 拍照

(3) For each corresponding point i in scene and image (从已知坐标求解投影矩阵 P)

$$\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ w^{(i)} \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ z_w^{(i)} \\ 1 \end{bmatrix}$$

$$u^{(i)} = \frac{p_{11}x_w^{(i)} + p_{12}y_w^{(i)} + p_{13}z_w^{(i)} + p_{14}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

$$v^{(i)} = \frac{p_{21}x_w^{(i)} + p_{22}y_w^{(i)} + p_{23}z_w^{(i)} + p_{24}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

(4) Rearranging the terms(最少 6 组)

$$\underset{\substack{A \\ \text{Known}}}{\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1x_w^{(1)} & -u_1y_w^{(1)} & -u_1z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1x_w^{(1)} & -v_1y_w^{(1)} & -v_1z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_ix_w^{(i)} & -u_iy_w^{(i)} & -u_iz_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_ix_w^{(i)} & -v_iy_w^{(i)} & -v_iz_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_nx_w^{(n)} & -u_ny_w^{(n)} & -u_nz_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_nx_w^{(n)} & -v_ny_w^{(n)} & -v_nz_w^{(n)} & -v_n \end{bmatrix}} \underset{\substack{\vec{p} \\ \text{Unknown}}}{\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(5) Solve for p

$$A\vec{p} = 0$$

2. Property of Projection Matrices: Scale

投影矩阵 P 仅按尺度定义, 即哪怕知道了物体与相片的具体对应, 相机的参数仍有无穷多种解. 例如只要视场角相同, 不同焦距配上合适距离可以拍摄出一致的图像.

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv k \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}, \quad k \neq 0$$

$$\therefore P \equiv kP$$

所以需要一定约束.

- Option 1: $p_{34} = 1$
- Option 2: $\|\vec{p}\|^2 = 1$, 一般用此

求为

$$\min_{\vec{p}} \|A\vec{p}\|^2, \text{ when } \|\vec{p}\| = 1$$

可以证明 $A^T A$ 最小特征值 λ 对应的特征向量为最优的 $\vec{p}$, 然后通过 $\vec{p}$ 重构出 P.

3. Decompose Projection Matrices to Intrinsic and Extrinsic Matrices

将 P 分解为内参矩阵与外参矩阵.

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M_{int}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{ext}}$$

K 为上三角矩阵, R 为正交矩阵, 可以通过 “QR factorization” 从 KR 分解出唯一的 K 与 R.

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = KR$$

然后有

$$\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = K\vec{t}$$

$$\therefore \vec{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

4. Camera Distortions

相机会因为镜头有非透视投影导致的畸变, 这些畸变可以被公式描述, 得到畸变参数后, 也可以通过优化消除. 但这里忽略了.

5. Visual Locaization Problem

视觉定位问题 (求解外参, 默认内参已知)

(1) 找到对应关系 (3D-2D): 同样通过图像匹配

(2) 已知 3D 世界坐标与对应 2D 相机坐标后求解外参 (一般假设内参已知)

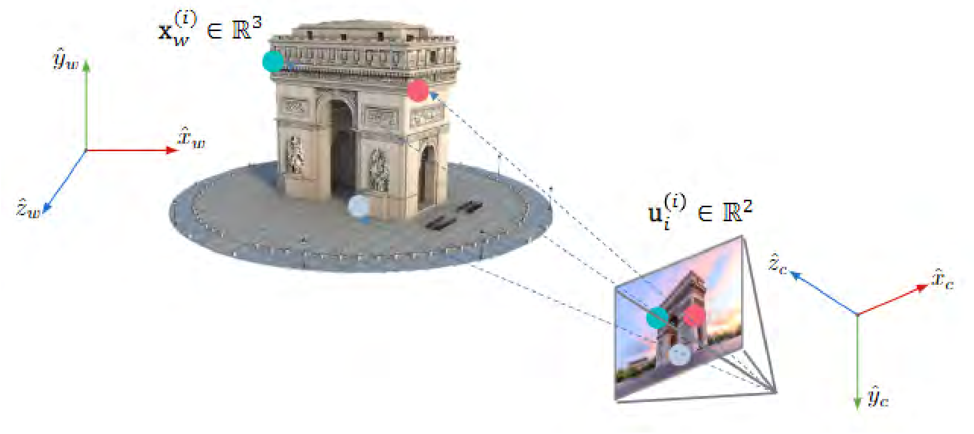

$\mathbf{x}_w^{(i)} \in \mathbb{R}^3$
$\hat{y}_w$
$\hat{x}_w$
$\hat{z}_w$
$\mathbf{u}_i^{(i)} \in \mathbb{R}^2$
$\hat{z}_c$
$\hat{x}_c$
$\hat{y}_c$

图 7.7: 3D-2D

6. Perspective-n-Point problem(PnP)

给定三维点坐标与对应图像的位置, 求解相机的位置与旋转.
旋转矩阵虽然是 3x3 但只有三个角度参数, 再加上三个平移参数, 统共为 6 自由度, 所以问题也称 6DoF

6.1 Direct Linear Transform (DLT)

直接硬解, 但不常用, 因为内参已知

$$\underset{\text{Known}}{\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix}} \equiv \underset{\text{Unknown}}{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}} \underset{\text{Known}}{\begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ z_w^{(i)} \\ 1 \end{bmatrix}}$$

6.2 P3P

只用 3 对点求解 (不是唯一解), 等价于求解 OA,OB,OC 的长度, 即如何把 abc 套到 O-ABC 之中. (纯几何) Oa, Ob, Oc 已知, ABC 坐标已知.

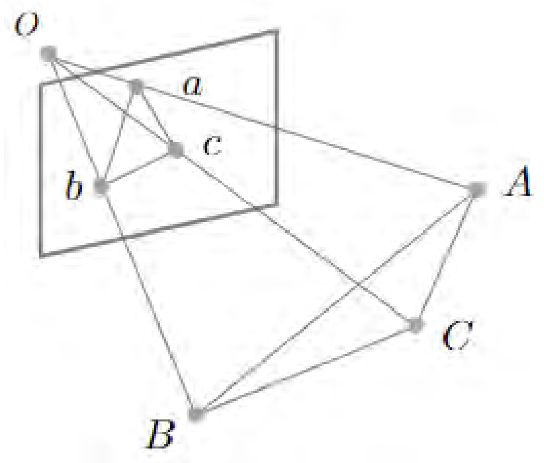


图 7.8: P3P

(1) 根据余弦定理

$$\begin{aligned} OA^2 + OB^2 - 2OA \cdot OB \cdot \cos\langle a,b\rangle &= AB^2 \\ OB^2 + OC^2 - 2OB \cdot OC \cdot \cos\langle b,c\rangle &= BC^2 \\ OA^2 + OC^2 - 2OA \cdot OC \cdot \cos\langle a,c\rangle &= AC^2 \end{aligned}$$

(2) 同除 OC^2, 令 $x = \frac{OA}{OC}, y = \frac{OB}{OC}$

$$\begin{aligned} x^2 + y^2 - 2xy\cos\langle a,b\rangle &= \frac{AB^2}{OC^2} \\ y^2 + 1^2 - 2xy\cos\langle b,c\rangle &= \frac{BC^2}{OC^2} \\ x^2 + 1^2 - 2xy\cos\langle a,c\rangle &= \frac{AC^2}{OC^2} \end{aligned}$$

(3) 令 $v = \frac{AB^2}{OC^2}, u = \frac{BC^2}{AB^2}, w = \frac{AC^2}{AB^2}$

$$\begin{aligned} x^2 + y^2 - 2xy\cos\langle a,b\rangle - v &= 0 \\ y^2 + 1^2 - 2xy\cos\langle b,c\rangle - uv &= 0 \\ x^2 + 1^2 - 2xy\cos\langle a,c\rangle - wv &= 0 \end{aligned}$$

(4) 消去 v 并重排, 得二元二次方程组

$$\begin{aligned} (1-u)y^2 - ux^2 - \cos\langle b,c\rangle y + 2uxy\cos\langle a,b\rangle + 1 &= 0 \\ (1-w)x^2 - wy^2 - \cos\langle a,c\rangle x + 2wxy\cos\langle a,b\rangle + 1 &= 0 \end{aligned}$$

$$\begin{aligned} \text{Known: } & \cos\langle a,b\rangle, \cos\langle b,c\rangle, \cos\langle a,c\rangle, u, w \\ \text{Unknown: } & x, y \end{aligned}$$

有四个解, 两个在正面, 两个在背面, 所以用额外一对点验证是哪一个解

6.3 PnP

优化问题, 最小化 reprojection error (重投影误差)

$$\min_{R,t} \sum_i \|p_i - K(RP_i + t)\|^2$$

$$p_i\text{为 2D 点}$$

$$K(RP_i + t)\text{为 3D 对应 2D 的关系}$$

使用 P3P 初始化, 用 Gauss-Newton 优化. 得到的解应该是最准确的.
注:

- R 必须正交, 或用轴角直接写出 R(减少变量数, 且保证正交)
- 收敛解非全局最优解, 需要初始化

一个比较好的流程是用 P3P 的 RANSAC, 然后最小化一下误差.

7. Object Pose Estimation

已知 2D, 3D 点与 2D 旋转求 3D 相对相机的位置与旋转, 需找 3D 与 2D 对应关系. 可以直接手动定义关键点然后转化为关键点匹配问题, 后求 PnP 问题.

三、 Structure from Motion(SfM)

给多张图, 获得三维点云以及相机位姿.

1. Solving SfM

(1) Assume intrinsic matrix K is known for each camera. (假设内参已知)

(2) Find a few reliable corresponding points. (寻找一些点到点的对应关系, 特征匹配)

(3) Find relative camera position t and orientation R. (求解相机相对平移与旋转)

(4) Find 3D position of secne points. (三角化交点找到三维空间具体位置)

2. Epipolar geometry

对极几何, 描述一 3D 点在不同 2D 视角下的几何关系, 通过此反求出视角.

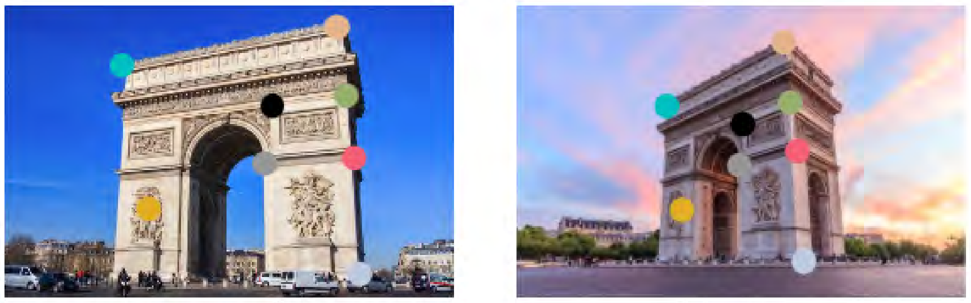


图 7.9: 匹配

(1) base line(基线): the line connecting the center of cameras. (相机中心的连线)

(2) Epipole(极点): Image point of origin/pinhole of one camera as viewed by the other camera. (基线与相平面的角点) e_l 与 e_r 为极点, 对每一对点, 其是唯一的.

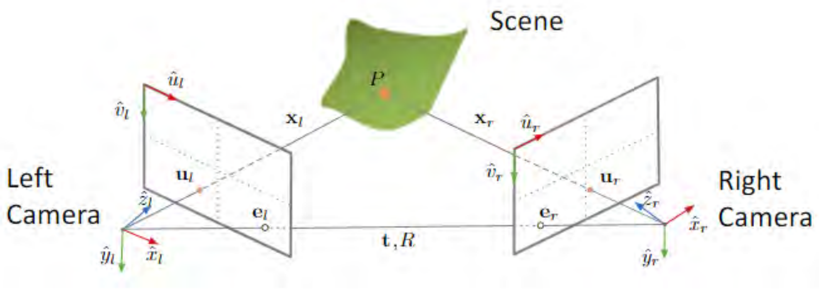


图 7.10: Epipole

(3) Epipolar Plane of Scene Point P: The plane formed by camera origins (O_l and O_r), epipoles (e_l and e_r) and scene point P. (相机中心与 P 构成的平面, 对每个 P, 其也是唯一的)

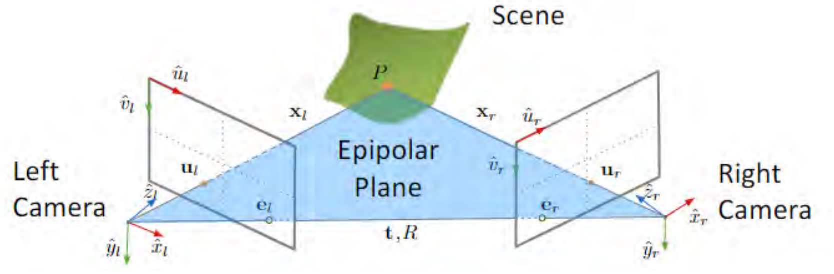


图 7.11: Epipolar Plane

2.1 Epipolar Constraint

极线约束, 目的是描述 e_l 与 e_r 的关系.

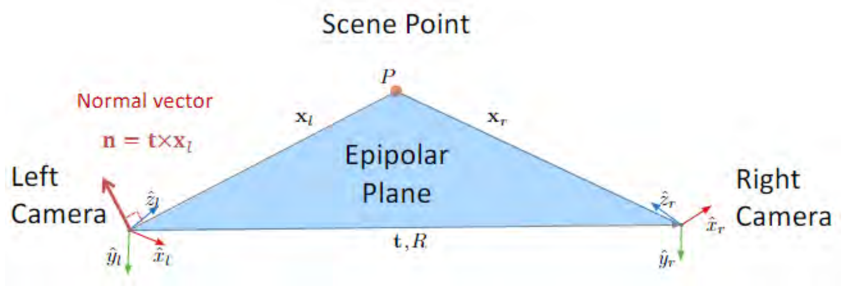


图 7.12: Epipolar Constraint

$$P \text{ 相对于 Left Camera 的坐标: } \vec{x}_l$$
$$P \text{ 相对于 Right Camera 的坐标: } \vec{x}_r$$
$$\text{相机基线: } \vec{t}$$
$$\text{相机相对旋转: } R$$
$$\therefore \text{极平面法向量: } \vec{n} = \vec{t} \times \vec{x}_l$$

$$\therefore \vec{x}_l \cdot (\vec{t} \times \vec{x}_l) = 0$$

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \begin{bmatrix} t_y z_l - t_z y_l \\ t_z x_l - t_x z_l \\ t_x y_l - t_y x_l \end{bmatrix} = 0$$

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \underbrace{\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}}_{T_\times} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = 0$$

易知:

$$\vec{x}_l = R\vec{x}_r + \vec{t}$$

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

代入得:

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \left(\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}_{T_\times \cdot \vec{t} = \vec{t} \times \vec{t} = 0} \right) = 0$$

$$\therefore \begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \underbrace{\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}}_{E} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

Essential Matrix(本质矩阵): $E = T_\times R$

The Decomposition (分解) of Essential Matrix E: 若 E 已知, 可以获得 $\vec{t}$ 与 R. (具体方法略去)

2.2　Find Essential Matrix E

$$\vec{x}_l^T E \vec{x}_r = 0$$

但仅已知二维坐标, $\vec{x}_l$ 与 $\vec{x}_r$ 未知. 但其可以使用二维坐标表示, 即透视变换.

$$z_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} z_l u_l \\ z_l v_l \\ z_l \end{bmatrix} = \begin{bmatrix} f_x^{(l)} x_l + z_l o_x^{(l)} \\ f_y^{(l)} y_l + z_l o_y^{(l)} \\ z_l \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} \\ 0 & f_y^{(l)} & o_y^{(l)} \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Camera Matirx } K_l} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

$$z_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} \\ 0 & f_y^{(l)} & o_y^{(l)} \\ 0 & 0 & 1 \end{bmatrix}}_{K_l} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

$$z_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} \\ 0 & f_y^{(r)} & o_y^{(r)} \\ 0 & 0 & 1 \end{bmatrix}}_{K_r} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}$$

$$\therefore \vec{x}_l^T = \begin{bmatrix} u_l & v_l & 1 \end{bmatrix} z_l {K_l^{-1}}^T$$

$$\vec{x}_r = K_r^{-1} z_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix}$$

由此得

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

$$\begin{bmatrix} u_l & v_l & 1 \end{bmatrix} z_l {K_l^{-1}}^T \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} K_r^{-1} z_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

虽然 z_l 与 z_r 未知, 但因为右为 0, 所以可以忽略, 得:

$$\begin{bmatrix} u_l & v_l & 1 \end{bmatrix} {K_l^{-1}}^T \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} K_r^{-1} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

令 $E = K_l^T F K_r$,

$$\therefore \begin{bmatrix} u_l & v_l & 1 \end{bmatrix} \underbrace{\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}}_{\text{Fundamental Matirx } F} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

也有 Scale 的问题, 所以也需要 $\|F\| = 1$ 的约束.

3. Relative Camera Pose Estimation

(1) For each correspondence i, write out epipolar constraint.

$$\underbrace{\begin{bmatrix} u_l^{(i)} & v_l^{(i)} & 1 \end{bmatrix}}_{\text{Known}} \underbrace{\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}}_{\text{Unknown}} \underbrace{\begin{bmatrix} u_r^{(i)} \\ v_r^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}} = 0$$

Then expand the matrix to get linear equation:

$$\left(f_{11} u_r^{(i)} + f_{12} v_r^{(i)} + f_{13}\right) u_l^{(i)} + \left(f_{21} u_r^{(i)} + f_{22} v_r^{(i)} + f_{23}\right) v_l^{(i)} + f_{31} u_r^{(i)} + f_{32} v_r^{(i)} + f_{33} = 0$$

(2) Rearrange terms to form a linear system.

$$\underbrace{\begin{bmatrix} u_l^{(1)} u_r^{(1)} & u_l^{(1)} v_r^{(1)} & u_l^{(1)} & v_l^{(1)} u_r^{(1)} & v_l^{(1)} v_r^{(1)} & v_l^{(1)} & u_r^{(1)} & v_r^{(1)} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_l^{(i)} u_r^{(i)} & u_l^{(i)} v_r^{(i)} & u_l^{(i)} & v_l^{(i)} u_r^{(i)} & v_l^{(i)} v_r^{(i)} & v_l^{(i)} & u_l^{(i)} & u_r^{(i)} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_l^{(m)} u_r^{(m)} & u_l^{(m)} v_r^{(m)} & u_l^{(m)} & v_l^{(m)} u_r^{(m)} & v_l^{(m)} v_r^{(m)} & v_l^{(m)} & u_l^{(m)} & u_r^{(m)} & 1 \end{bmatrix}}_{\substack{A \\ \text{Known}}} \underbrace{\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}}_{\substack{\vec{f} \\ \text{Unknown}}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$A\vec{f} = 0$$

(3) Find least squares solution for fundamental matrix F.

$$min_{\vec{f}}\left\|A\vec{f}\right\|, \text{ when } \left\|\vec{f}\right\| = 1$$

对 A^TA 使用特征值分解, 取其最小特征值对应特征向量就是我们的解.

Rearrange solution $\vec{f}$ to form the fundamental matrix F.

(4) Compute essential matrix E from known left and right intrinsic camera matrices and fundamental matrix F.

$$E = k_l^T F K_r$$

(5) Extract R and $\vec{t}$ from E.(Using Singular Value Decomposition)

$$E = T_\times R$$

4. Triangulation

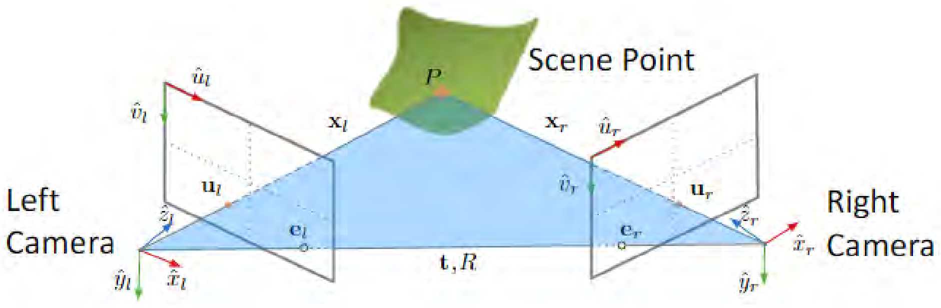


图 7.13: Triangulation

已知相机相对位姿与内参, 使用三角化测定三维的点位置.

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} & 0 \\ 0 & f_y^{(l)} & o_y^{(l)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} & 0 \\ 0 & f_y^{(r)} & o_y^{(r)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\text{and} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} & 0 \\ 0 & f_y^{(l)} & o_y^{(l)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} & 0 \\ 0 & f_y^{(r)} & o_y^{(r)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\therefore \tilde{u}_l = P_l \tilde{x}_r$$
$$\tilde{u}_r = M_{int_r} \tilde{x}_r$$

$$\therefore \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

Rearranging the terms:

$$\underbrace{\begin{bmatrix} u_r m_{31} - m_{11} & u_r m_{32} - m_{12} & u_r m_{33} - m_{13} \\ v_r m_{31} - m_{21} & v_r m_{32} - m_{22} & v_r m_{33} - m_{23} \\ u_l p_{31} - p_{11} & u_l p_{32} - p_{12} & u_l p_{33} - p_{13} \\ v_l p_{31} - p_{21} & v_l p_{32} - p_{22} & v_l p_{33} - p_{23} \end{bmatrix}}_{A_{4\times 3}} \underbrace{\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}}_{\vec{x}_r} = \underbrace{\begin{bmatrix} m_{14} - m_{34} \\ m_{24} - m_{34} \\ p_{14} - p_{34} \\ p_{24} - p_{34} \end{bmatrix}}_{\vec{b}_{4\times 1}}$$

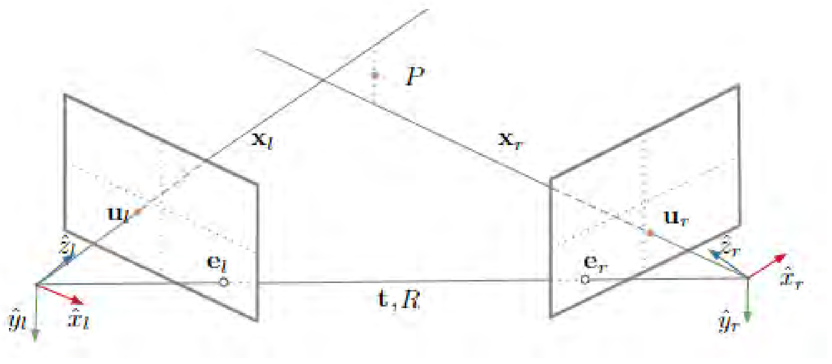


图 7.14: not exactly intersect

Generally, rays x_l and x_r will not exactly intersect. Find least squares solution by:

$$\vec{x}_r = (A^T A)^{-1} A^T \vec{b}$$

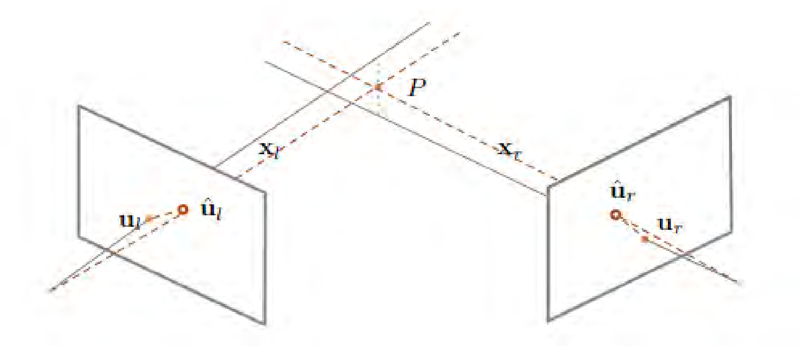


图 7.15: Minimize reprojection error

Or minimize reprojection error(重投影误差):

$$cost(\mathrm{P}) = \|\vec{u}_l - \hat{u}_l\|^2 + \|\vec{u}_r - \hat{u}_r\|^2$$

5. Sequential Structure from Motion(Multi-frame structure from motion)

序列式重建, 一帧一帧去建.

(1) Initialize camera motion and scene structure. (前比较近的两帧先得到三维点云, 以第一帧做世界坐标系)

(2) For each additional view:

a. Determine projection matrix of new camera using all the known 3D points that are visible in its image. (视觉定位估计新相机位姿)

b. Refine and extend structure: compute new 3D points, reoptimize existing points that are also seen by this camera. (新增特征点三角化加入三维点云)

(3) Refine structure and motion: Bundle Adjustment.(集束调整, 后处理)

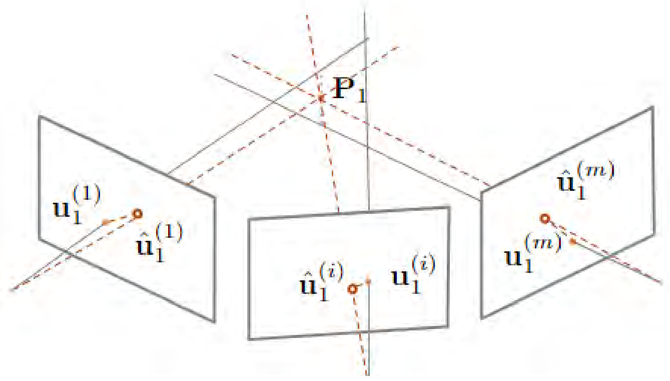


图 7.16: Bundle adjustment

Refining 3D points and camera parameters by minimizing reprojection error over all frames.(最小化所有点的重投影误差并优化相机位姿, m 帧, n 关键点)

$$E(P_{proj}, \vec{P}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left\| u_j^{(i)} - P_{proj}^{(i)} \vec{P}_j \right\|^2$$

第八章　Depth estimation and 3D reconstruction

深度估计与三维重建 (稠密重建)

一、　Depth estimation

1.　Introduction

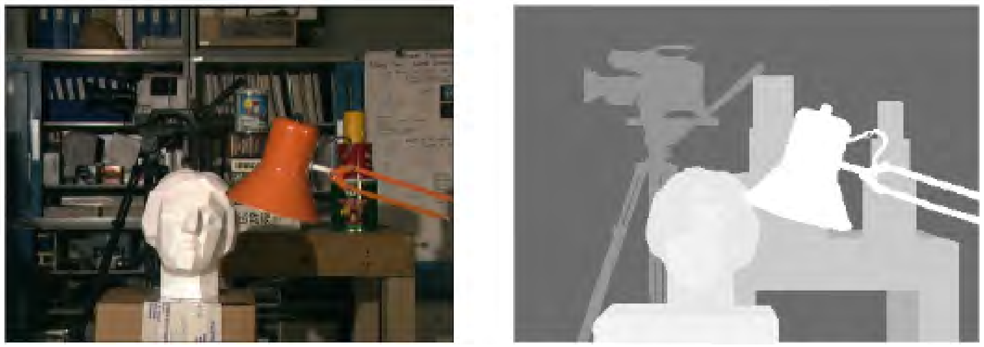


图 8.1: Compute depth maps

深度为离相机中心的距离, 注意是垂直距离还是直线距离 (不同算法所给不同, 本节课是垂直距离).

1.1　Depth sensing

(1) 避障

(2) 人脸识别 (确保扫描的不是图像, 是三维的物体)

(3) 人机交互, 体感游戏

1.2　Active depth sensing

主动式深度传感器

(1) LiDAR: Light Detection and Ranging (雷达, 测光反射的时间以测距离, ToF)

a. 昂贵

b. 准确

c. 最大可 360°

(2) Structured light (结构光)

(3) Active stereo (主动式立体显像)

1.3 Passive depth sensing

被动式深度感知

(1) Stereo (立体视觉)

(2) Monocular (单目估计, 依赖神经网络)

2. Stereo vision

依靠双眼成像视差判断距离, 近的视差大, 远的视差小.

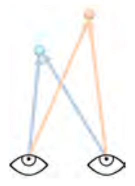


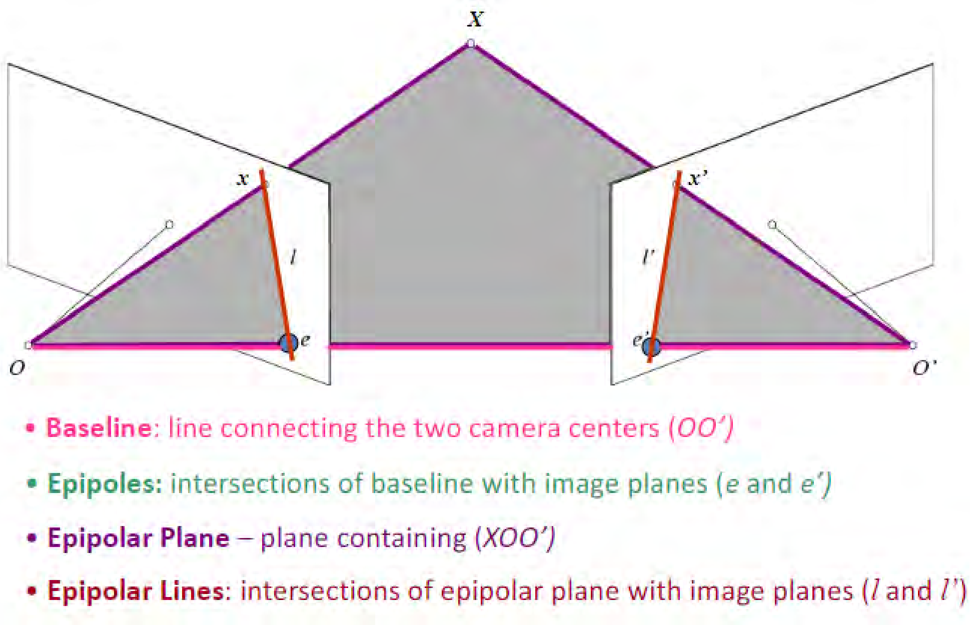
图 8.2: 视差

使用双目摄像机

(1) Find 2D-2D correspondences.

(2) Triangulate.

问题是需要每个像素的匹配与相对高效的速率. 可使用光流解决, 但不够好, 因其并没有双目先验 (点到点关系有极线约束).



- **Baseline:** line connecting the two camera centers (*OO'*)
- **Epipoles:** intersections of baseline with image planes (*e* and *e'*)
- **Epipolar Plane** – plane containing (*XOO'*)
- **Epipolar Lines:** intersections of epipolar plane with image planes (*l* and *l'*)

图 8.3: Epipolar geometry

在这里使用了 Epipolar Lines(极线).

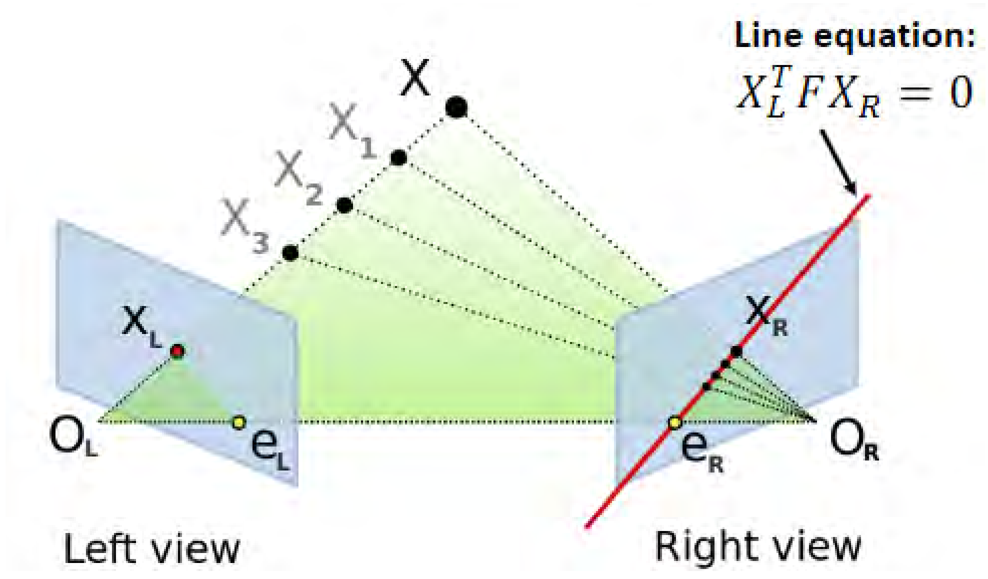


图 8.4: Epipolar line

$$X_L^T F X_R = 0$$

给定 X_L, X_R 必须满足这个方程. 这个方程就是上文的极线约束.

3. Basic stereo matching algorithm

3.1 For each pixel in the first image

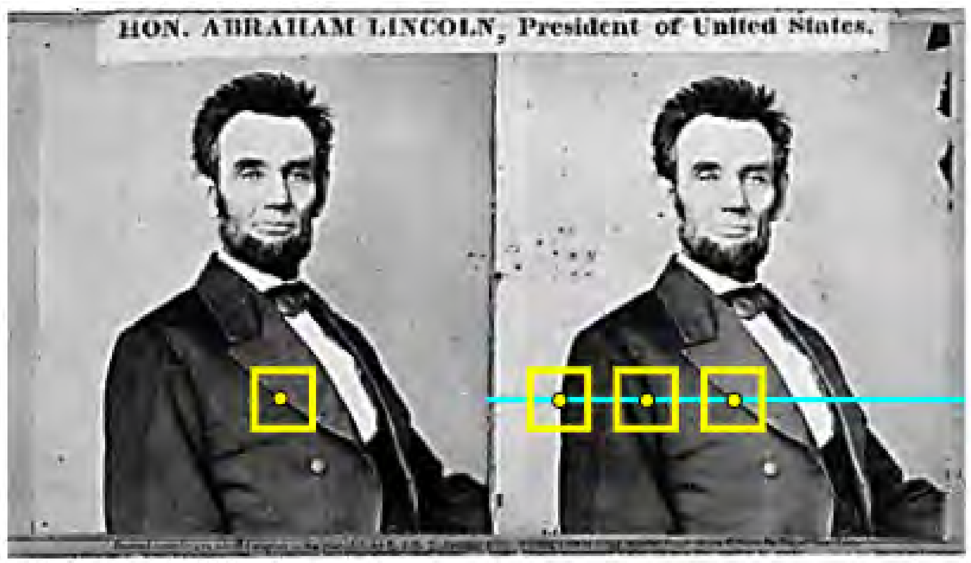


图 8.5: Basic stereo matching algorithm

(1) Find corresponding epipolar line in the right image.

(2) Search along epipolar line and pick the best match.

3.2 Simplest case: Parallel images

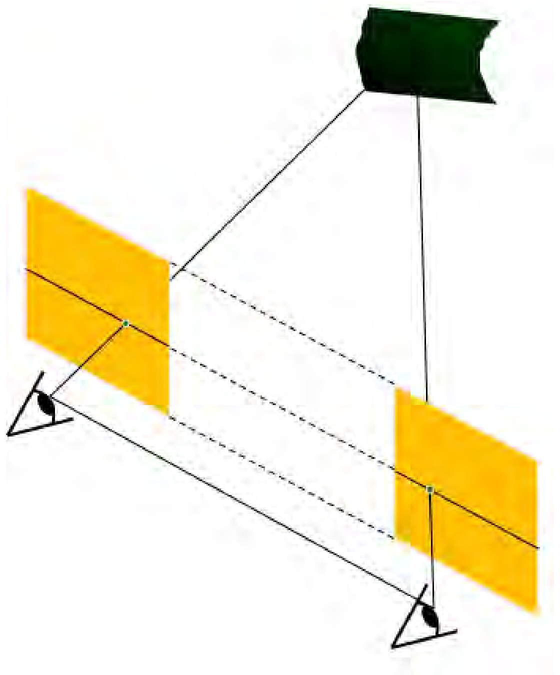


图 8.6: Parallel images

epipolarlines are horizontal scanlines.

(1) Image planes of cameras are parallel to each other and to the baseline.

(2) Camera centers are at same height.

(3) Focal lengths are the same.

(4) Then, epipolar lines fall along the horizontal scan lines of the images.

3.3 Depth from disparity(视差)

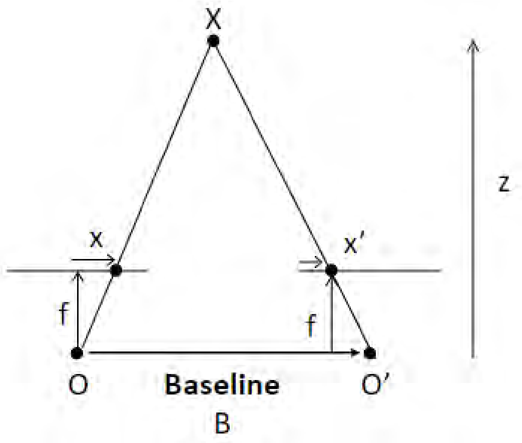


图 8.7: Disparity

$$\text{Disparity} = x - x' = \frac{B \cdot f}{z}$$

视差与深度严格成反比.

3.4 Stereo image rectification

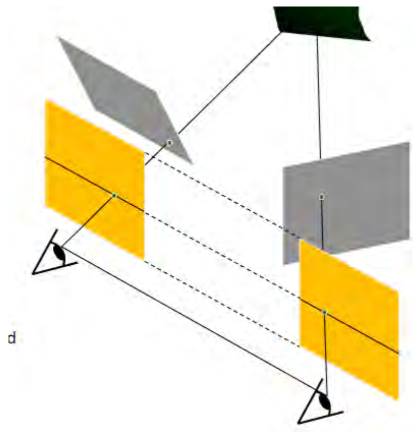


图 8.8: rectification

(1) Reproject image planes onto a common plane parallel to the line between camera centers.

(2) Two homographies (3x3 transform), one for each input image reprojection.

4. Stereo matching algorithms

4.1 Match pixels in conjugate epipolar lines

(1) Assume brightness constancy.

(2) This is a challenging problem.

(3) Hundreds of approaches.

4.2 Best match minimizes the dissimilarity

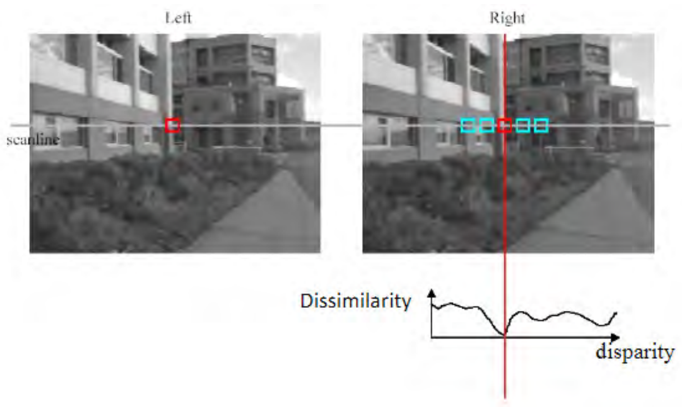


图 8.9: minimizes the dissimilarity

4.3 Popular matching scores

(1) SSD (Sum of Squared Differences)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$$

(2) SAD (Sum of Absolute Differences)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|$$

(3) ZNCC (Zero-mean Normalized Cross Correlation)

$$\frac{\sum_{x,y}(W_1(x,y) - \bar{W}_1)(W_2(x,y) - \bar{W}_2)}{\sigma_{W_1}\sigma_{W_2}} \text{ where } \bar{W}_i = \frac{1}{n}\sum_{x,y} W_i,\ \sigma_{W_i} = \sqrt{\frac{1}{n}\sum_{x,y}(W_i - \bar{W}_i)^2}$$

对亮度有不变性.

4.4 Window size

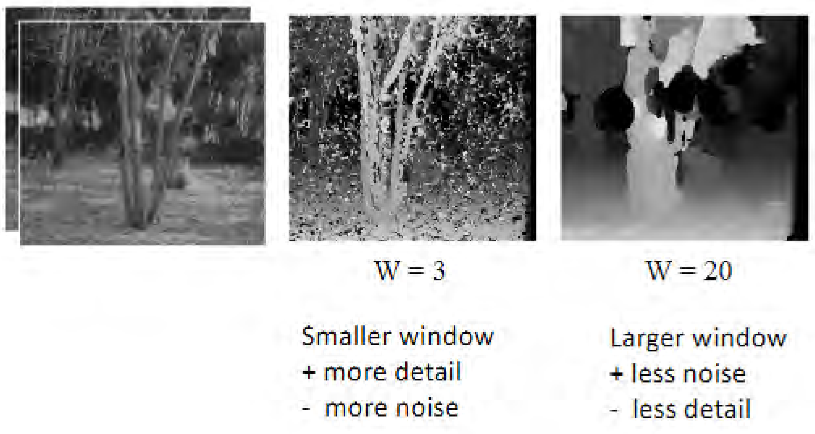
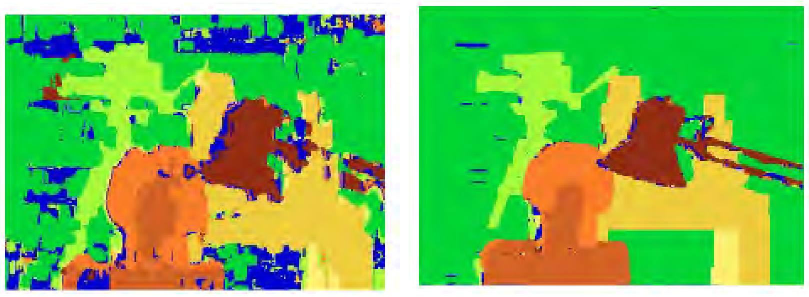


图 8.10: Window size

但即使是最佳 Window size 效果也不好, 但加图像分割 (MRF, Markov 随机场) 后效果显著.



(a) best window size (b) Graph cuts-based method

5. Stereo as energy minimization

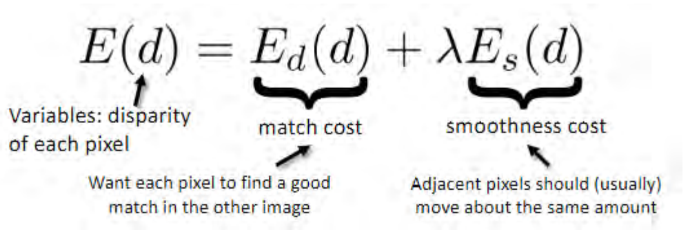

$E(d) = \underbrace{E_d(d)}_{\text{match cost}} + \lambda \underbrace{E_s(d)}_{\text{smoothness cost}}$
Variables: disparity of each pixel
match cost
smoothness cost
Want each pixel to find a good match in the other image
Adjacent pixels should (usually) move about the same amount

图 8.12: energy minimization

- $E_d(d) = \sum_{(x,y)\in I} C\,(x, y, d(x, y))$ eg: SSD, SAD, ZNCC(只跟每个像素各自深度有关系)
- $E_s(d) = \sum_{(p,q\in\varepsilon)} V(d_p, d_q)$(跟相邻像素深度有关系)
  - ε: set of neighboring pixels.
  - V
    - L_1 distance: $V(d_p, d_q) = |d_p - d_q|$
    - “Potts model”: $V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$ (对边缘巨大差别不会过于光滑)

这是一个离散优化问题, 假设 MRF(ε) 仅在一条线上, 可以用 DP 求解. 若 MRF(ε) 是二维的随机场, 可以转化为最小割问题求解, 或使用机器学习的 belief 模型.

6. Stereo reconstruction pipeline

(1) Calibrate cameras(相机标定)

(2) Rectify images(图像矫正)

(3) Compute disparity(计算视差)

(4) Estimate depth(估计深度)

6.1 Choosing the stereo baseline

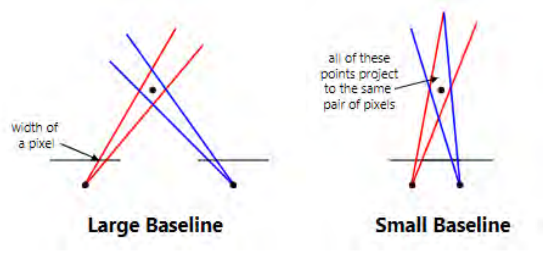


图 8.13: Baseline

- Too small: large depth error
- Too large: difficult search problem

6.2 What will cause errors

- Camera calibration errors
- Poor image resolution
- Occlusions(遮挡)
- Violations of brightness constancy (specular reflections)
- Large motions
- Textureless regions(无纹理)

7. Active stereo with structured light

就像二维码, 给表面人为增加结构光, 使匹配更容易. 一般打的是红外的结构光 (人眼不可见, 但可用红外相机), 甚至投影仪可以当作一个相机 (投影的结构光已知).

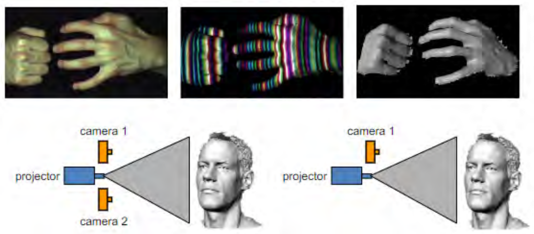


图 8.14: structured light

8. Multi-view stereo

Advantages:

(1) Can match windows using more than 1 neighbor, giving **a stronger constraint**.

(2) If you have lots of potential neighbors, can **choose the best subset** of neighbors to match per reference image.

(3) Can reconstruct a depth map for each reference frame, and the merge into a **complete 3D model**.

8.1 Basic idea

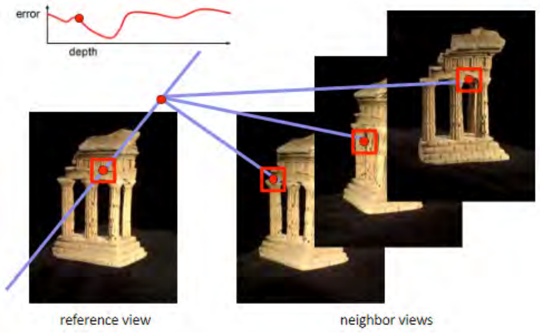


图 8.15: Multi-view stereo

Compute the error for each depth value for each point in the reference image. Find the depth value that gives the smallest error. 源图片某深度后投影到其他图片获得一个 block, 用此 block 与源图片算 ZNCC, 最小的 ZNCC 对应的即为正确的深度.

8.2 Plane-Sweep

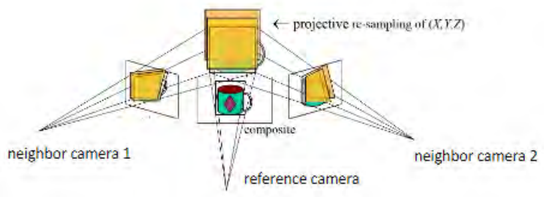


图 8.16: Plane-Sweep

如何高效计算所有像素的 cost. 每个点所有深度的 cost 构成一个向量, 所有点的向量一起构成一个三维块, 每一个深度平面可以一起投影到图像 (单应) 可以得到 Cost Volumes(误差块), 用其可以得到深度信息. 效率没有显著提高 (复杂度未变).

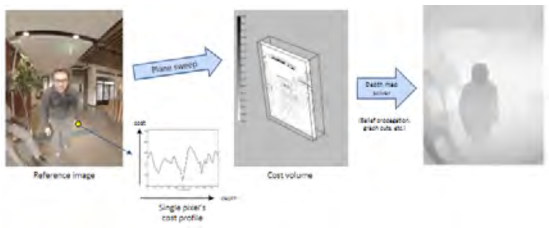

Plane sweep
Reference image
cost
depth
Single pixel's cost profile
Cost volume
Depth map solver
(Belief propagation, graph cuts, etc.)

图 8.17: Cost Volumes

8.3 PatchMatch

图像块匹配. (一个随机化算法)

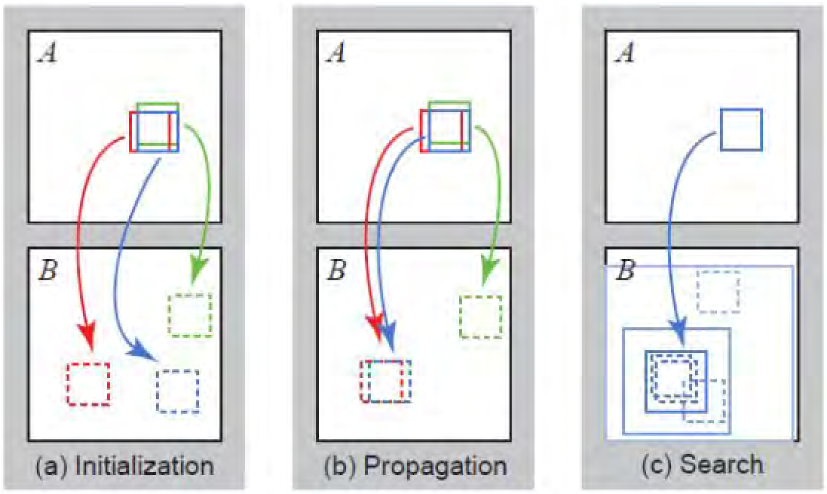


图 8.18: PatchMatch

(1) Random initialization: Each pixel is given a random patch offset as initialization. (每个图像块随机进行匹配)

(2) Propagation(传播): Each pixels checks if the offsets from neighboring patches give a better matching patch. If so, adopt neighbor's patch offset. (相邻图像块的匹配是否更优, 如果优就更换匹配)

(3) Local search: (扰动以寻找更优的匹配)

a. Each pixels searches for better patch offsets within a concentric radius around the current offset.

b. The search radius starts with the size of the image and is halved each time until it is 1.

(4) Go to Step 2 until converge.

In MVS, replace patch offsets by depth values in the above algorithm.

二、 3D reconstruction

1. 3D Representations

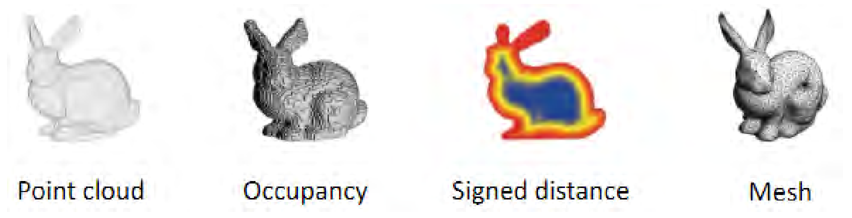


图 8.19: 3D Representations

- Point cloud (点云)
- Volume(体素, voxel) (非常占用空间)
  - Occupancy (占用)
  - Signed Distance Function(SDF) (点到物体表面的距离)
  - Truncated Signed Distance Function (TSDF) (截断距离场): Truncation SDF' s distance value to $[-1, 1]$ (太远的不记录具体值, 就为 1 了)
- Mesh (网格): usually triangle mesh. (甚至可以贴材质)

Texture Mapping(纹理贴图): 每个网格顶点给一个坐标 (O-uv), 以此绘制二维纹理图.

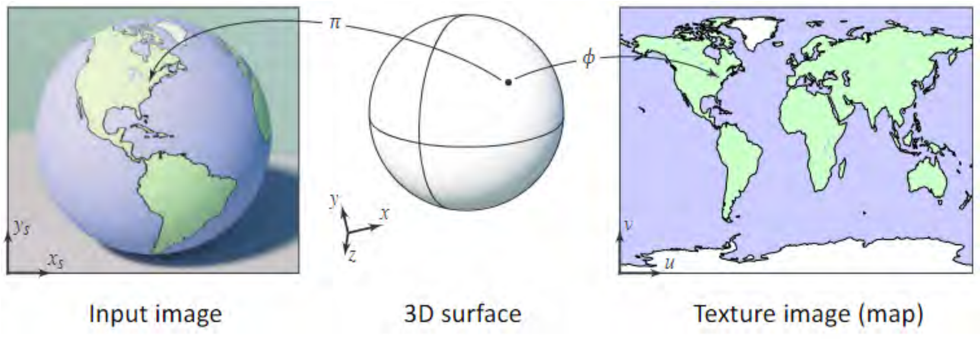


图 8.20: Texture Mapping

三维重建就是为了获得 Mesh 与对应的纹理图.

2. 3D surface reconstruction

Fuse depth maps into a complete 3D mesh. (将多张深度图融合成为三维网格)

(1) Depth maps $\longrightarrow$ TSDF/occupancy volume (深度图到体素)

a. Kinest Fusion

b. Poisson reconstruction $\star$

(2) TSDF/occupancy volume $\longrightarrow$ mesh (体素到网格)

为何从点云到体素再到网格?

(1) 操作简单

(2) 去噪简单 (在体素上去噪)

2.1 Poisson reconstruction

将深度图转化为体素.

(1) 将深度图化为点云. (从二维到三维的反投影)

$$\begin{cases} z = depth(i,j) \\ x = \frac{(j-e_x)\times z}{f_x} \\ y = \frac{(i-e_y)\times z}{f_y} \end{cases}$$

(2) 计算每个点的法向量 (normal vector): 取周围一些点拟合平面计算法向量 or 进行三维的主成分分析, 取最小的主成分.

(3) 将点转化为体素

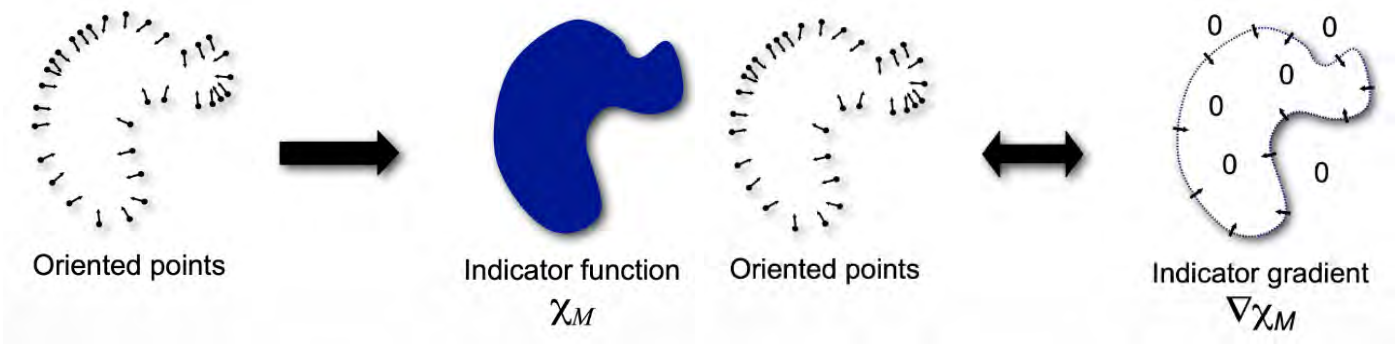


图 8.21: Represent surface

$$\chi_M(p) = \begin{cases} 1 & \text{if } p \in M \\ 0 & \text{if } p \notin M \end{cases}$$

a. Represent the oriented points by a vector field $\vec{V}$

b. Find the function χ whose gradient best approximates $\vec{V}$ by minimizing:

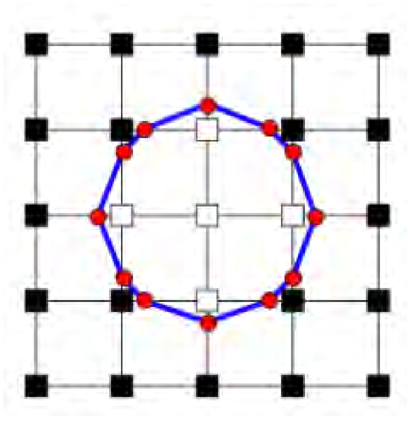
$$\min_{\chi} \left\| \nabla \chi - \vec{V} \right\|$$

c. Applying the divergence operator, we can transform this into a Poisson problem: (使用泊松方程求解)

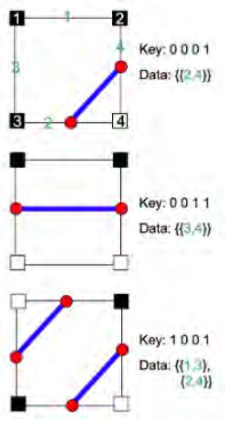
$$\nabla \times (\nabla \chi) = \nabla \times \vec{V} \Longleftrightarrow \Delta \chi = \nabla \times \vec{V}$$

2.2 Marching cubes

从体素提取网格.



(a) Marching Squares (2D)



(b) look-up table

图 8.22: Marching Squares (2D)

For each grid cell with a sign change

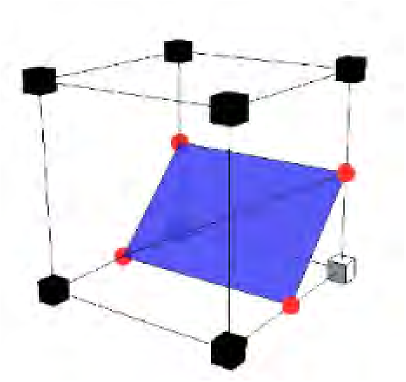
(1) Create one vertex on each grid edge with a sign change.

(2) Connect vertices by lines.

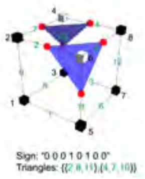
a. Lines should not intersect

b. Use a pre-computed look-up table

(3) Location of the vertex can be determined by linear interpolation of SDF values.



(a) Marching Squares (3D)



(b) look-up table

图 8.23: Marching Squares (3D)

For each grid cell with a sign change

(1) Create one vertex on each grid edge with a sign change

(2) Connect vertices by triangles

   a. Triangles should not intersect
   b. Much more complicated than 2D cases
   c. But it still has look-up table
      - $2^8 = 256$ sign configurations
      - 15 configurations in total (considering symmetry)

第九章　Deep Learning

一、　Machine learning

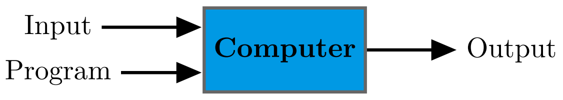


图 9.1: Traditional Programming

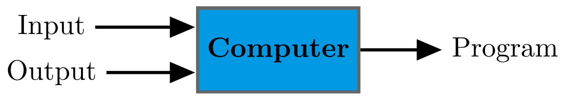


图 9.2: Machine Learning

1.　Important concepts

1.1　Model

A mathematical description that explains the relation between input x and output y. (输入输出的映射关系)
Defined as a function $y = f_W(x)$ with parameters dentoed by W.
Problem types:

- Regression(回归): y is a real number.
- Classification(分类): y is a discrete label.

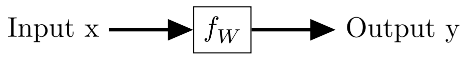


图 9.3: Model

1.2　Supervised learning(监督学习)

Find f_W from labeled data.
labeled data: exisiting (x,y) pairs, called training data.
Tow phases:

(1) Training: given labeled data, find f_W. (model fitting, 模型拟合)

(2) Testing: given f_W and a new x, find y. (inference, 模型推理)

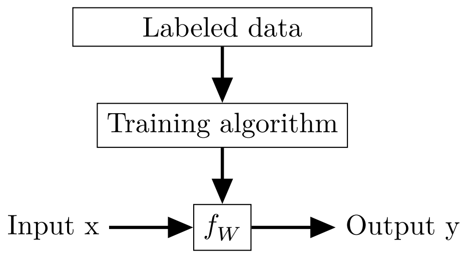


图 9.4: Supervised learning

1.3　General pipeline

(1) Define the problem: What is x and y.

(2) Collect training data: (x,y) pairs.

(3) Design a model: What is the form of f_W.

(4) Model training

a. Define a loss function $l(W)$.

b. Find optimal W that minimizes $l(W)$ using some optimzation algorithm.

(5) Testing(application): Given a new data x, predict y with the learning f_W.

Example:

- 回归: 最小二乘
- 分类: 图像分类

二、 Linear classifier(线性分类器)

1. Classification model

- Input x: image
- Output y: class scores

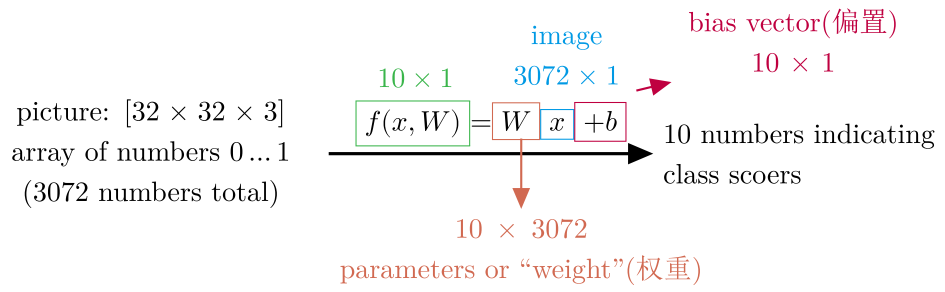


图 9.5: Linear classifier

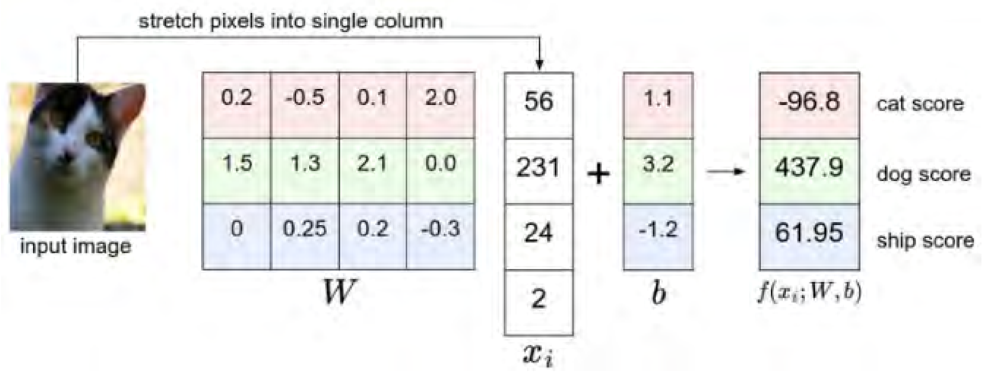


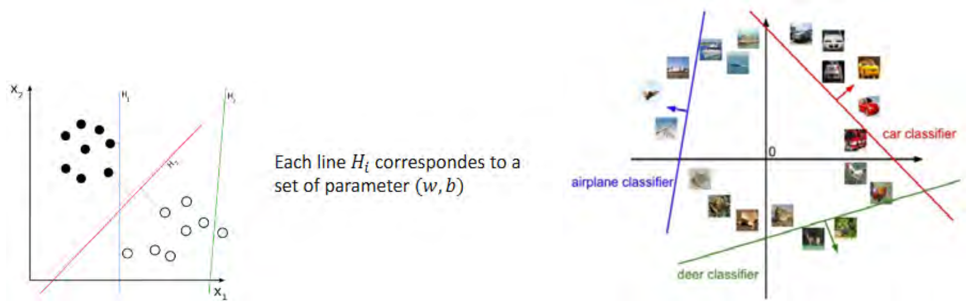
图 9.6: Example with an image with 4 pixels and 3 classes

2. Interpret linear classifier

When is the score $w^T x + b$ large?

- When x (image) is similar to w (weights). (w 与 x 越像分数越大)
- w represents how an object should look like. (w 是 W 中一类所对应的一行)

We can decide the class label by thresholding the score, then $w^T x + b = 0$ is called decision boundary.



(a) 2-class　　(b) Multi-class

图 9.7: Linear classifier in 2D

若是在 2D 中, 相近的点表示一个类别, 分类器的作用就是将这些类别划分, 高维情况同理.

3. Training: find good parameters

(1) Difficult to handcraft w.

(2) Define a loss (objective) function that quantifies the “goodness” (likelihood) of w.

(3) Solve the optimization problem.

4. Loss function of classification

对分类不能使用 MSE loss, 因为 labels 是离散的而分数是连续的.
To define a loss function, class labels can be viewed as probabilities. (将离散类别转化为概率)

(1) Convert scores to probabilities

(2) Compute cross entropy between predicted and true probabilities

4.1 Convert scores to probabilities

$$\text{Softmax operator: } \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \dots, K$$

z_i 带代表的是类别的分数.

4.2 Cross entropy(交叉熵) as loss function

cross entropy, 在统计上用于衡量分布的距离. 熵越小, 分布越一致.
分布 S, L 的交叉熵 D(S,L) 为

$$D(S, L) = -\sum_i L_i \log S_i$$

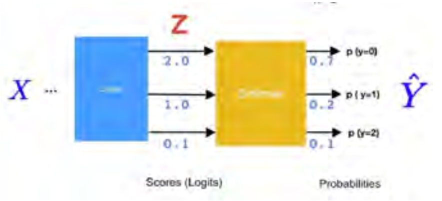
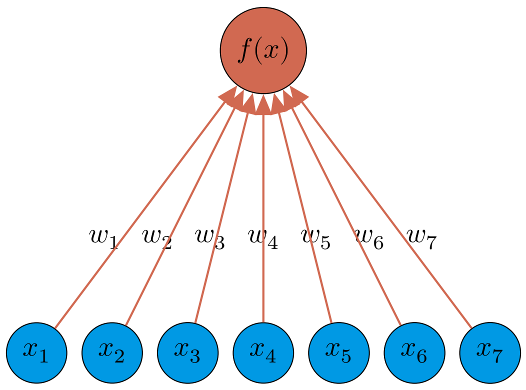


图 9.8: Convert scores to probabilities

三、 Neural networks

1. Graphical representation

Perceptron(感知机, 最简单的网络, 为线性分类器)



$f(x)$
w_1 w_2 w_3 w_4 w_5 w_6 w_7
x_1 x_2 x_3 x_4 x_5 x_6 x_7

图 9.9: Perceptron

2. Limitation of linear model

However, the data may be NOT linear separable.

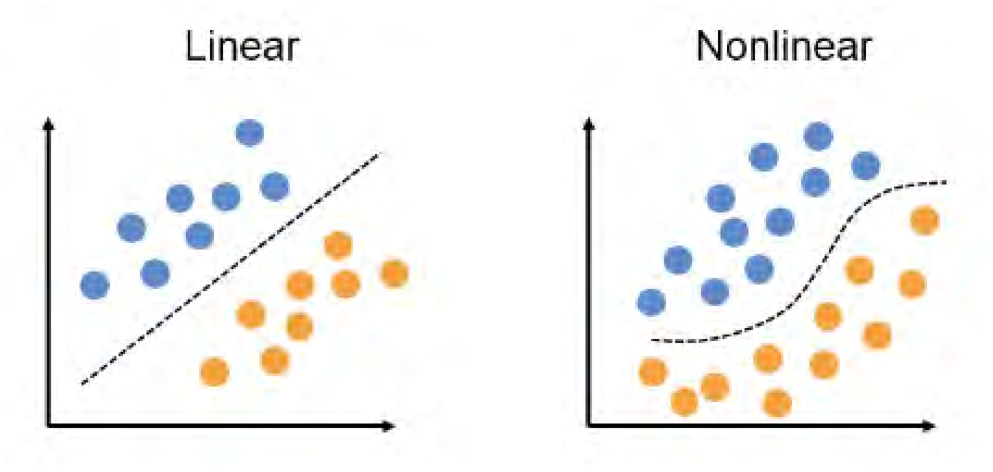


图 9.10: Limitation of linear model

3. Activation functions(激活函数)

To model nonlinearity, add a nonlinear transform, named activation functions.

$$f(x) = \sigma(w^T x + b)$$

Examples:

- Rectified Linear Unit (ReLU): $\sigma(z) = \max(0, z)$ (**Electrical pulse** cannot be negative)
- Sigmoid:$\sigma(z) = \frac{1}{1+e^{-z}}$ (Map values to probability)

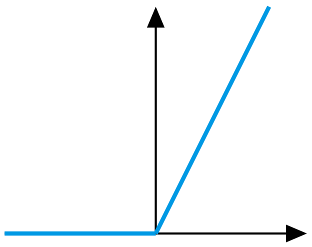


图 9.11: ReLU

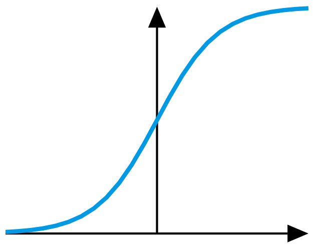


图 9.12: Sigmoid

4. Multiple outputs

To represent multiple outputs.

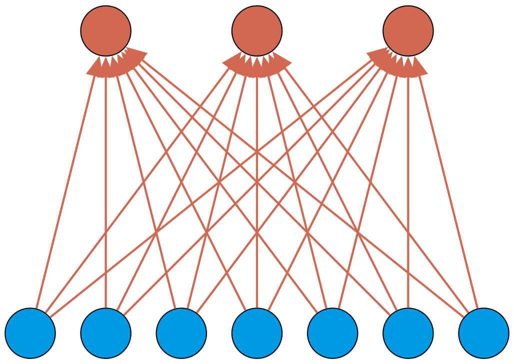


图 9.13: Multiple outputs

$$f(x) = \sigma(Wx + b),\ W \text{ is matrix}$$

5. Multi-layer perceptron

$$f(x) = \sigma(W_2\sigma\left(W_1x + b_1\right) + b_2)$$

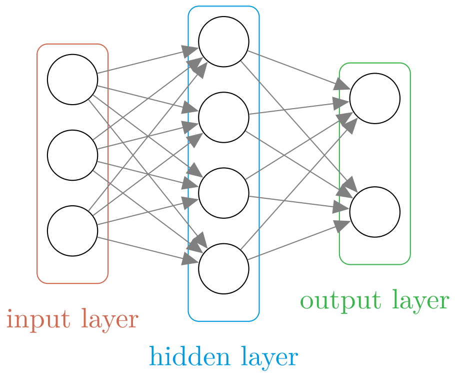


图 9.14: Multi-layer perceptron

可以多层叠加, 构成复杂神经网络. 若无激活函数, 所有分层将和为同一层, 即分层无效.

6. Neural networks

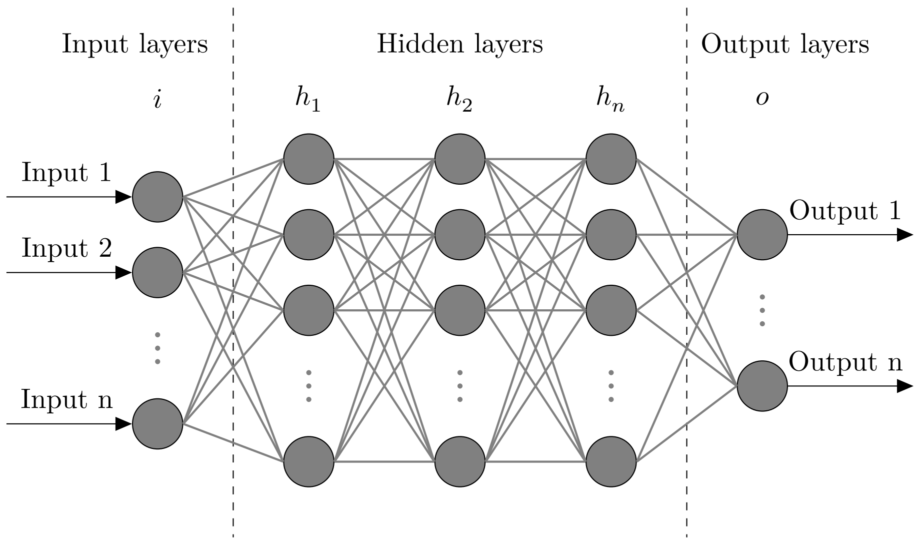


图 9.15: Neural networks

输入, 隐层, 输出, 边对应着权重.

(1) (Artificial) neural networks -linear functions chained together and separated by non-linear activation functions. (神经网络就是线性分类器的叠加, 用非线性激活函数分隔)

(2) Used to regress the function that maps input to output.

(3) A computational model inspired by human nervous system.

为何叫神经网络: 与人脑应该比较接近 (主要是人脑运作机制不清楚)

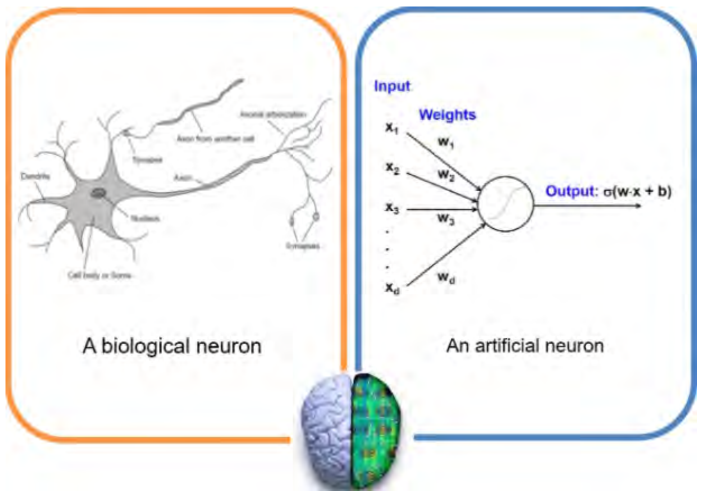


图 9.16: Artificial neuron

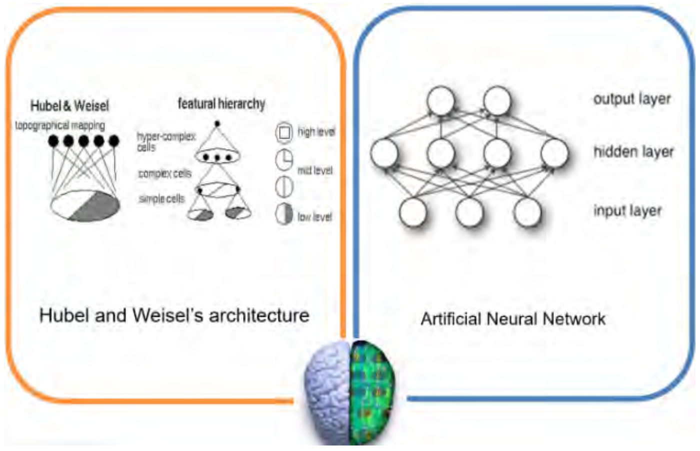


图 9.17: Artificial neural networks

7. Deep Neural networks

(1) Deep neural networks: neural networks with many layers

(2) A network with more layers mean is able to represent more complex function

(3) The more the better?

a. More parameters to learn: need more data

b. More difficult to train: “gradient vanishing”

8. Fully connected layer(全连接层)

所有结点与上一层链接.

$$f(x) = \sigma(Wx + b)$$

四、 Convolutional neuarl networks

卷积神经网络

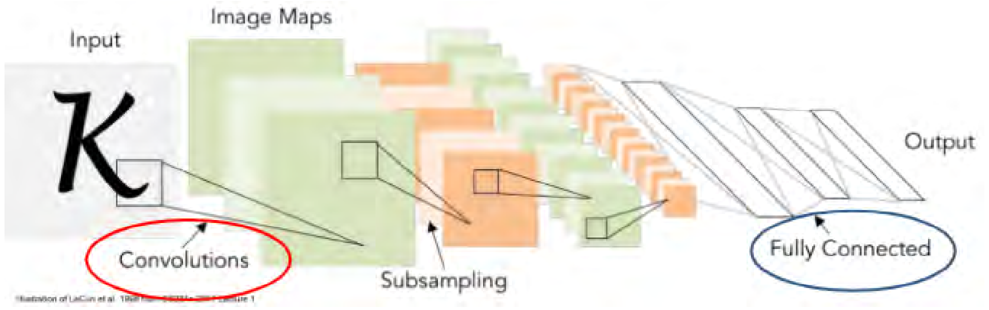


图 9.18: Convolutional neural networks

1. Reason

1.1 Locally connected network

全连接参数过多, 使用局部链接.

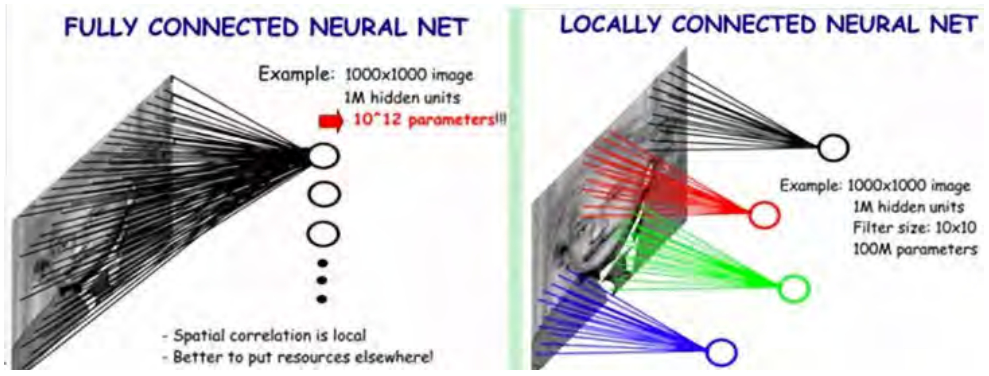


图 9.19: Locally connected network

1.2 Weight sharing

局部链接参数还是太多, 使用相同的参数处理链接. Why is weight sharing a valid assumption?

- Reminder: weights represent how an object should look like. (全局相似)

- Should not change with its position in the image(Shift invariance property, 平移的不变性)

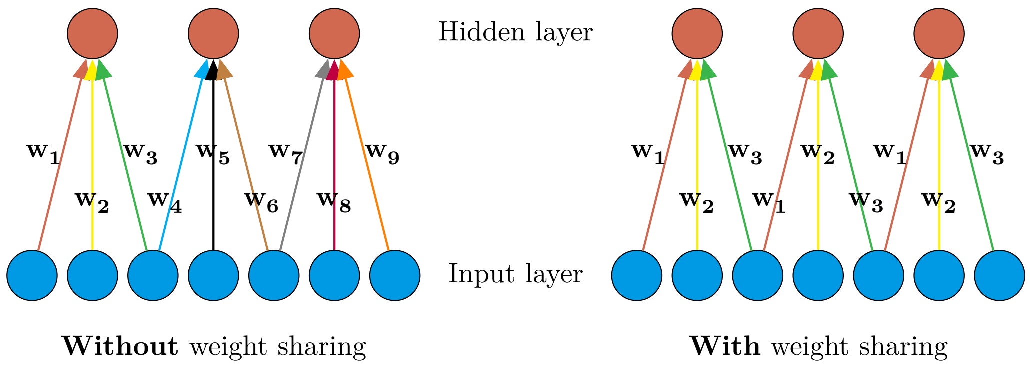


图 9.20: weight sharing

1.3　Convolution

Local connectivity + weight sharing = convolution

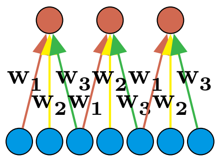


图 9.22: Convolution

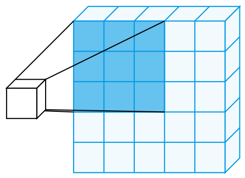


图 9.21: 2D convolution

$$y = \sigma(x \otimes w + b)$$

2. Convolution

2.1　2D convolution

The size of output is $ImageSize - FilterSize + 1$.

2.2　Padding and stride

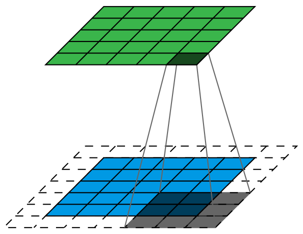


图 9.23: **Padding**: adding zero pixels around the border

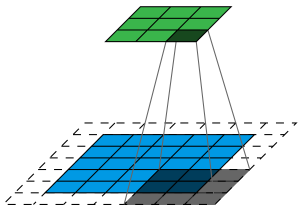


图 9.24: **Stride**: step size large than 1 in convolution

The size of output is $\frac{ImageSize - FilterSize + Padding * 2}{Stride} + 1$

2.3 Convolution of multi-channel image(3D tensor)

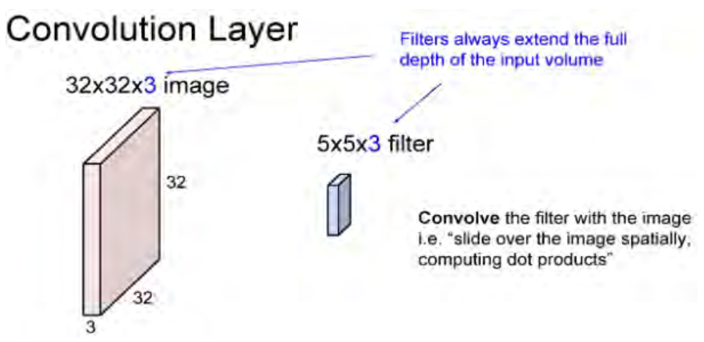

Convolution Layer
Filters always extend the full depth of the input volume
32x32x3 image
5x5x3 filter
32
32
3
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

图 9.25: 3D Convolution

The Number of weight is $5 \times 5 \times 3 + 1 = 76$ (is 3072 for fully-connected layer)

3. Convolution layer

- Input: image
- Output: feature map (activation map)(特征图/响应图)
- Parameters: filter

Could be more than 1 filter. The output are multiple feature maps. Number of filters = number of feature maps

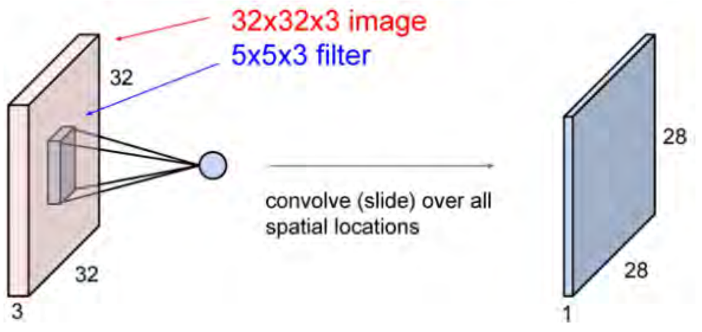


图 9.26: Convolution layer

4. Convolution neural network(CNN)

CNN is a sequence of convolution layers separated by activation functions.

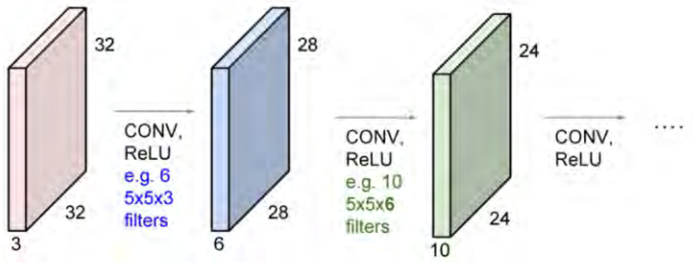


图 9.27: Convolution neural network

Reminder: neural network is a sequence of linear functions seperated by activation functions.

4.1 Receptive fields((可视域))

A convolution layer with kernel size K, each element in the output depends on K x K receptive field in the input.

For successive convolution with K kernel size and L layers, the receptive field is 1 + L * (K -1).

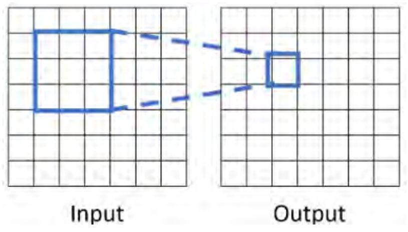


图 9.28: Receptive fields

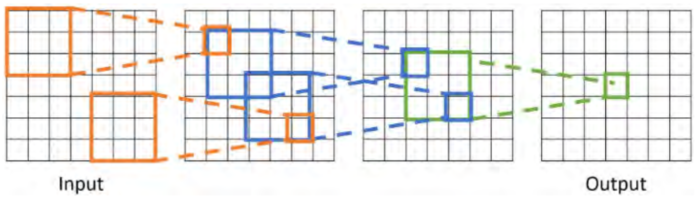


图 9.29: successive convolution

4.2 Pooling(池化) layer

Pooling: combine outputs of a filter at different locations.

Pooling operators:

- Max pooling
- Average pooling

Pooling is to Aggregate spatial information and Makes feature maps smaller and more manageable. (整合信息并缩小特征图)

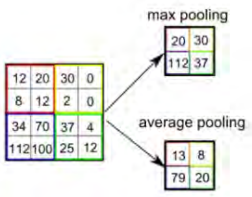
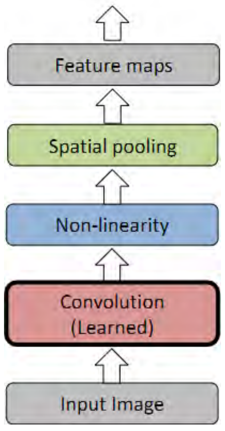
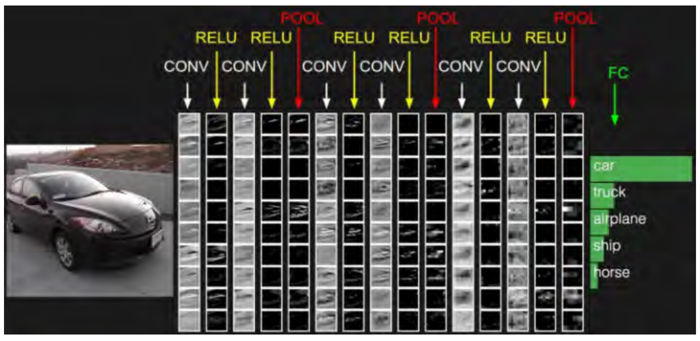


图 9.30: Pooling operators



(a) CNN



(b) CNN for classification

结构设计比较玄学, 总的来说是利用图像特性让参数变小.

卷积核可视化, 层级越高核越像特征的样貌.

五、 Training neural networks

Training: find network weights w to minimize the error between true training labels y_i and estimated labels $f_w(x_i)$

$$L(w) = \frac{1}{n}\sum_i l(y_i, f_w(x_i))$$

For Example:

- L2 loss for regression
- Cross-entropy loss for classification

Minimization can be done by gradient descent(梯度下降) provided f is differentiable(可微).This training method is called back-propagation(反向传播).

1. Gradient descent

A CNN as composition of functions

$$f_w(x) = f_L(\dots(f_2(f_1(x; w_1); w_2))\dots; w_L)$$

Parameters

$$w = (w_1, w_2, w_L)$$

Loss function

$$L(w) = \frac{1}{n}\sum_i l(y_i, f_w(x_i))$$

Gradient descent

$$w^{t+1} = w^t - \eta_t \frac{\partial L}{\partial w}(w^t)$$

$$\begin{aligned} w^{t+1} &: \text{ New weight} \\ w^t &: \text{ Old weight} \\ \eta_t &: \text{ Learning rate} \\ \frac{\partial L}{\partial w}(w^t) &: \text{ Gradient} \end{aligned}$$

2. Backpropagation

Use chain rule to calculate gradients of loss L w.r.t. weights w.

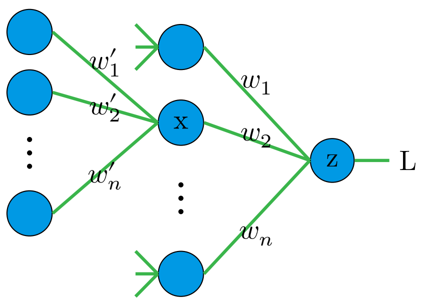


图 9.32: Backpropagation

$$\frac{\partial L}{\partial w_i} = \frac{\partial z}{\partial w_i}\frac{\partial L}{\partial z}$$
$$\frac{\partial L}{\partial w_i'} = \frac{\partial x}{\partial w_i'}\frac{\partial z}{\partial x}\frac{\partial L}{\partial z}$$

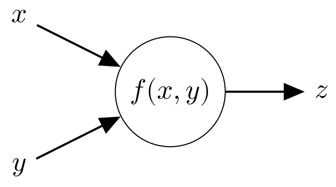
Algorithm:

(1) Forward data through the network, get loss.

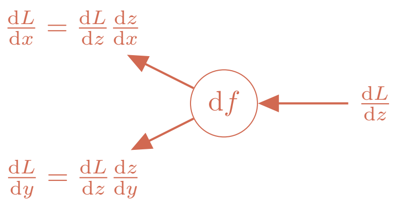
(2) Backprop to calculate the gradients.

(3) Update the parameters using the gradient.

(4) Go to step 1 if not converged.



(a) Forwardpass


$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$
df
$\frac{dL}{dz}$
$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$

(b) Backwardpass

3. Stochastic gradient descent(SGD)

Sometimes the training data is too large, e.g. millions of images in ImageNet. Calculate loss and gradients over all images in each iteration is too expensive.

Stochastic gradient descent(随机梯度下降): only calcualte loss and gradients on a batch of randomly sampled images.Stochastic gradients approximate the real gradients.

$$\hat{L}(w) = \frac{1}{n}\sum_{i\in\Omega} l(y_i, f_w(x_i)),\ SG = \frac{\partial \hat{L}}{\partial w}(w^t)$$

4. Architecture and hyper-parameters

4.1 Hyper-parameters(超参数)

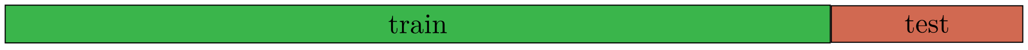
The number of layers to use, the number of filters in each layers, the better batch size and learning rate.
To set them, there is one option: try them all and see what works best. (逝)

4.2 Data split

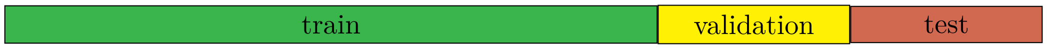
(1) Choose hyper-parameters that work best on the data Bad: Can always decrease training loss by using a larger netwrok. (过拟合数据)

Your Dataset

(2) Split data into **train** and **test**, choose hyper-parameters thatwork best on test data. Bad: no idea how it will perform on new data. (过拟合 test)



(3) Split data into **train**, **validation**(验证集) and **test**, choose hpyer-parameters on validation and evaluate on test. Great! Called **cross validation**(交叉验证)



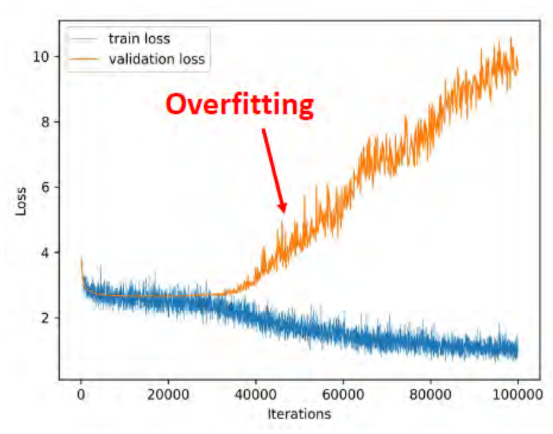


图 9.34: Overfit on train

loss 的波动是因为梯度选择是随机的 (SGD). 可以提前停止防止 overfit.

4.3 To pick hyper-parameters

(1) Train for original model.

(2) Validate to find hyper-parameters.

(3) Test to understand generalizability.

5. Overfitting

Overfit 原因之一是网络过于复杂, 使用正则来使一些参数无效化. 或者数据量过少, 有噪声, 或有 corner case.

5.1 Regularization

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max\left(0, f\left(x_i;W\right)_j - f\left(x_i;W\right)_{y_i}+1\right) + \boxed{\lambda R(W)}$$

In commom use:

- L2: $R(W) = \sum_k \sum_l W_{k,l}^2$
- L1: $R(W) = \sum_k \sum_l \left|W_{k,l}\right|$

5.2 Dropout

Dropout 可以证明与 L2 Regularization 作用相似.

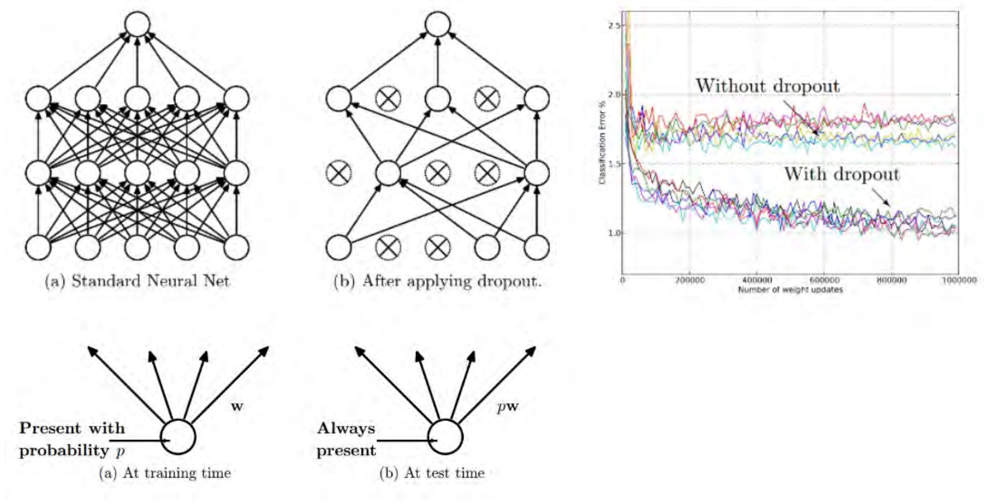


图 9.35: Dropout

5.3　Data augmentation(数据增广)

人为创造数据

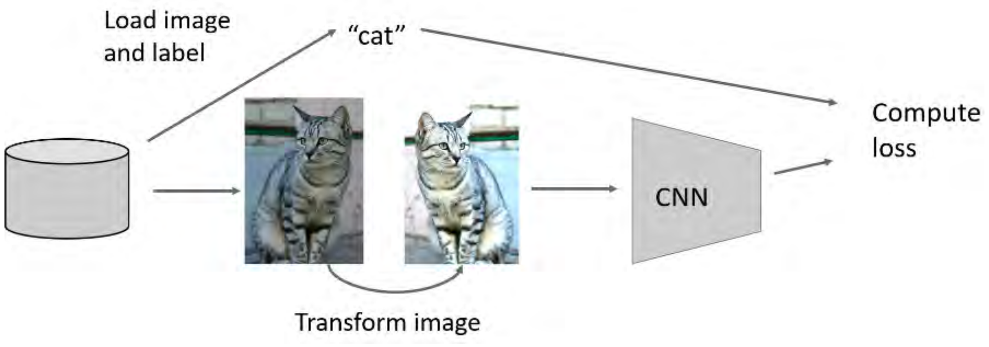


图 9.36: Data augmentation

改变亮度, 模糊, 几何变换, 裁剪图像等.

5.4　To prevent overfitting

- Cross validation and early stop.
- Regularization or dropout.
- Data augmentation.

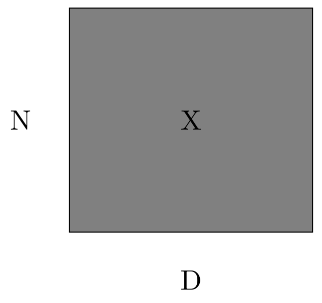
6.　Batch Normalization(归一化)

Normalize the outputs of a layer so they have zero mean and unit variance. To reduce internal covariate shift.

$$\widehat{x}^{(k)}=\frac{x^{(k)}-\mathrm{E}\left[x^{(k)}\right]}{\sqrt{\operatorname{Var}\left[x^{(k)}\right]}}$$

Input: $x \in N \times D$, N is channels, D is $B * H * W$ for a batch of images.

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$
$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j}-\mu_j)^2$$
$$\hat{x}_{i,j} = \frac{x_{i,j}-\mu_j}{\sqrt{\sigma_j^2+\varepsilon}}$$



Usually inserted after FC / Conv and before nonlinearity.
Advantages:

- Much easier to train.
- Allow higher learning rate, faster convergence.
- More robust to initialization.

7. Deep learning frameworks

Support fast development of deep learning algorithms.
Various modules to build a network: operations, layers, losses.
Various optimizers to train networks with CPU/GPU/Multiple GPUs.

- Caffe 早
- TensorFlow 便于部署, 工程化
- PyTorch 上手容易, 研究使用多

六、 Network architectures

历史

(1) 以前数据集小, 有过拟合, 且网络难以训练.

(2) ImageNet: 巨大的图像数据集.

(3) GPU: 矩阵并行, NVIDIA 的 cuda.

(4) AlexNet: 在 ImageNet 展示了深度学习的强大, 其在图像分类上减到了 16%, 甚至到 7%

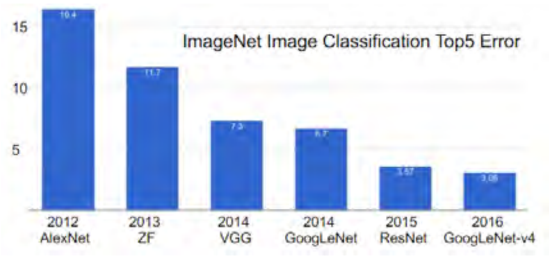


图 9.37: Progress on ImageNet

现在 ResNet 用的多.

(5) ResNet: 增加层数但 loss 变大.

设计一种网络, 使加一层但 loss 不会变大, 尝试训练出一种空白层 (identity map) 过于困难, 于是构造并训练一种空白层, 相当于用网络去学习输入到输出的残差, 而不是直接输入到输出, 如此就叫残差网络. 即使用残差逼近原来的输出.

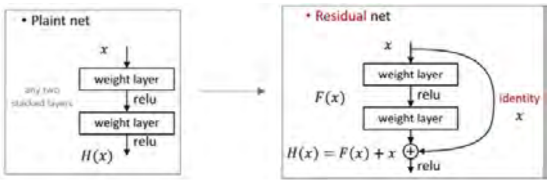


图 9.38: ResNet

梯度消失: 层数过多时梯度会越来越小以至于消失.

残差网络减弱了梯度消失.

(6) DenseNet: 每层输入会连到前面的输出, 相当于更多的残差.

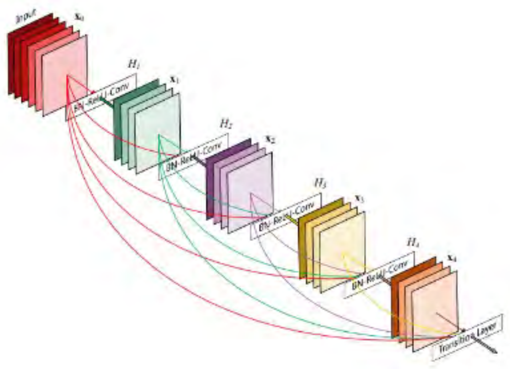


图 9.39: DenseNet

(7) MobileNets: 快! 可在移动端上运行.

(8) Neural architecture search: 自己调自己, 自己试自己. (套娃)

第十章　Recognition(识别)

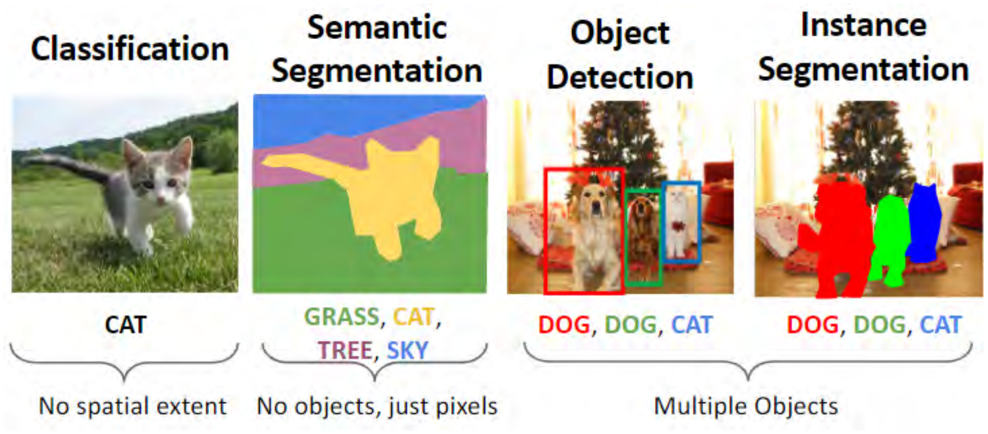


图 10.1: Recognition

一、　Semantic segmentation(语义分割)

Input: image

Output: Label each pixel in the image with a category label. Do not differentiate instances.

1.　Sliding window(移动窗口)

Very inefficient and Limited receptive fields. 效率低下且只有局部信息.

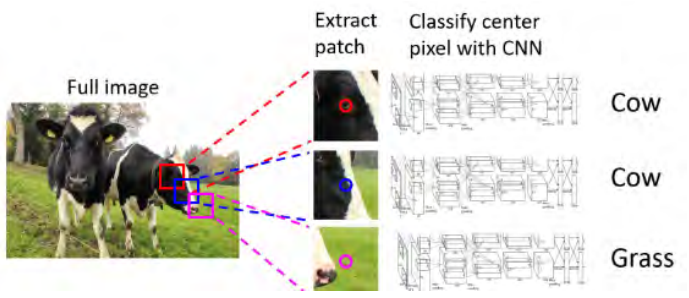


图 10.2: Sliding window

2.　Fully convolutional network(FCC)

都是卷积的网络. 从图像到图像, 一次进行整张图的估计, Loss function 就是每个像素的交叉熵 (cross-entropy) 之和.

tag 是手工标的, 正所谓 “多少人工就有多少智能”x

缺点: 可视域仅与层数成线性关系, 且高分辨率下消耗昂贵.

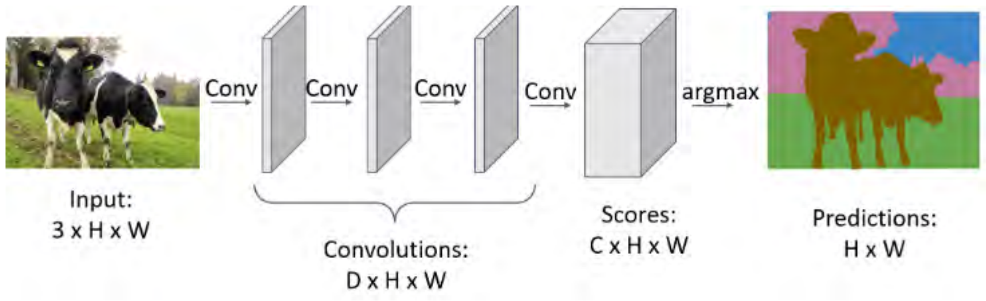


图 10.3: Fully convolutional network

To improve efficiency and receptive field:

- Downsampling: pooling, strided conv
- Unpooling

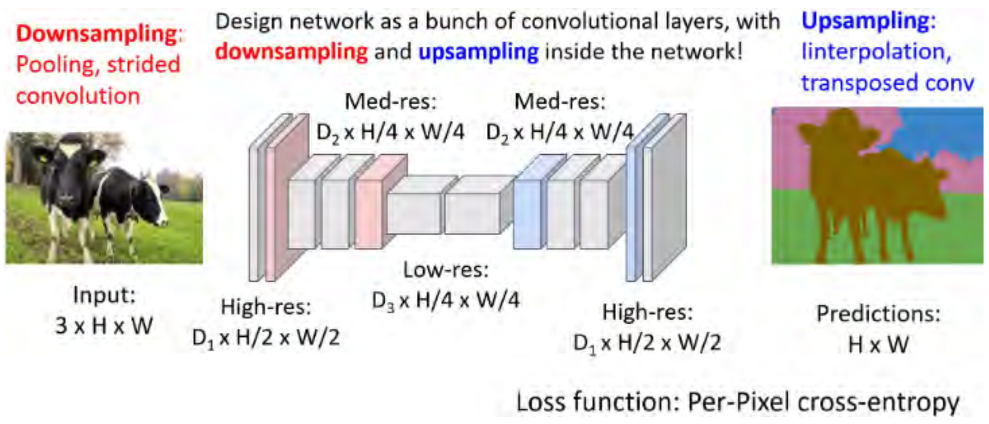

Downsampling: Pooling, strided convolution
Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!
Upsampling: linterpolation, transposed conv
Med-res: D_2 x H/4 x W/4
Med-res: D_2 x H/4 x W/4
Input: 3 x H x W
High-res: D_1 x H/2 x W/2
Low-res: D_3 x H/4 x W/4
High-res: D_1 x H/2 x W/2
Predictions: H x W
Loss function: Per-Pixel cross-entropy

图 10.4: improvement

2.1 Unpooling

- Bed of Nails (填零)
- Nearest Neighbor (填周围的)
  - Bilinear Interpolation (双线性)
  - Bicubic Interpolation (双三次)
- Transposed convolution (让网络去学习插值)

3. U-Net

Skip connection between downsampling and upsampling stages. (使用跳过层来保留信息) 也叫 hourglass-net

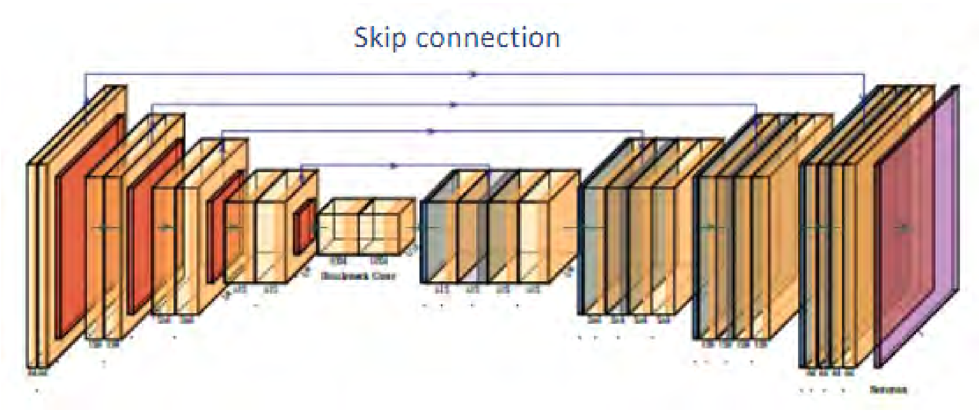


图 10.5: U-Net

4. DeepLab

FCN + CRF, 现阶段最好的方式

DCNN(膨胀卷积): 间隔卷积, 让卷积核更大但算量减少?

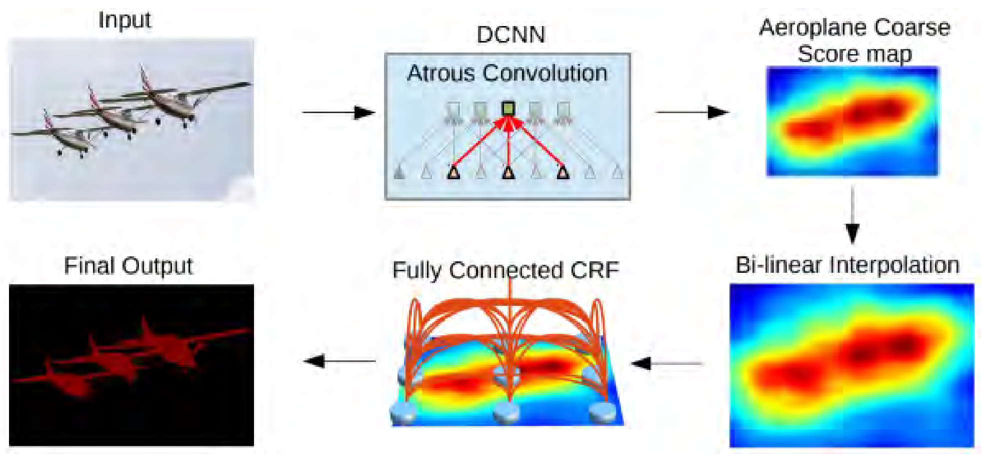


图 10.6: DeepLab

4.1 Conditional random field(CRF)

Energy function:

$$E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)$$

$$\text{Unary potential: } \theta_i(x_i) = -\log P(x_i) \text{ where } P(x_i) \text{ is the score map given by network}$$

$$\text{Pairwise potential: } \theta_{ij}(x_i, x_j) = \mu(x_i, x_j)\left[w_1 \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_{\beta^2}}\right) + w_2 \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2}\right)\right]$$

$$\text{where } \mu(x_i, x_j) = \begin{cases} 1 & x_i \neq x_j \\ 0 & \text{otherwise} \end{cases}$$

Pairwise potential: 权重取决于像素是否一致, 若不一致但分数相近也要减小惩罚, 最后捣腾为这个式子.

5. Evaluation metric(衡量测度)

Per-pixel Intersection-over-union (IoU), 交集除并集.
mIoU is mean IoU of each categories.

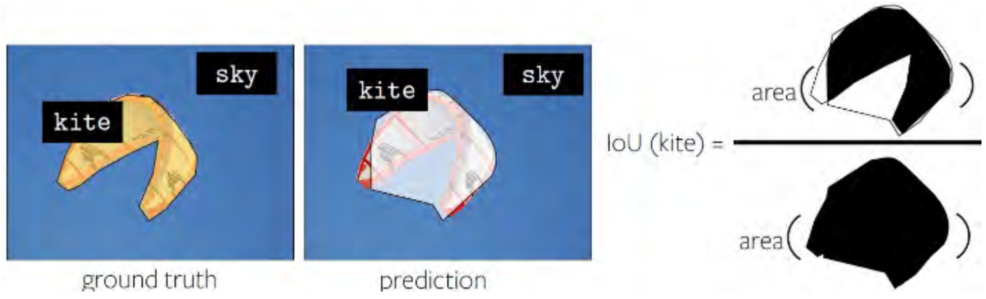


图 10.7: IoU

二、 Object detection

Input: image
Output: A set of bounding boxes that denote objects.
Bounding box (bbox):

- Class label
- Location (x,y)
- Size (w,h)

1. Detecting a single object

Treat localization as a regression problem.

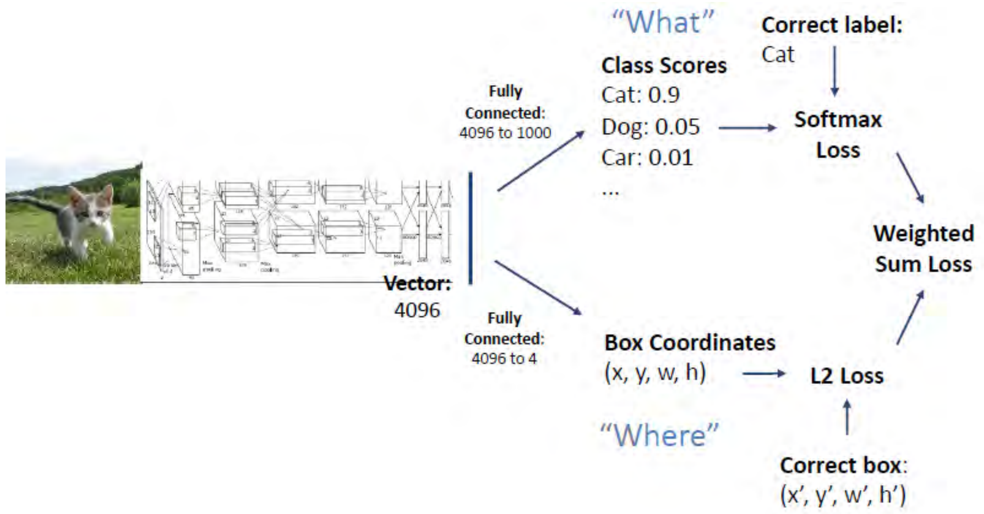


图 10.8: Detecting a single object

2. Detecting multiple objects

Bounding box 的数量并不一定, 这是最难以解决的地方.

2.1 Sliding window

~~没错又是我~~Apply a CNN to many different crops of the image, CNN classifies each crop as object or background. (直接检测一些框, 名 proposal)

但候选的框太多了, 无法穷举.

2.2 Region proposals

Find a small set of boxes that are likely to cover all objects. Often based on heuristics (eg: by over-segmentation). Relatively fast to run.

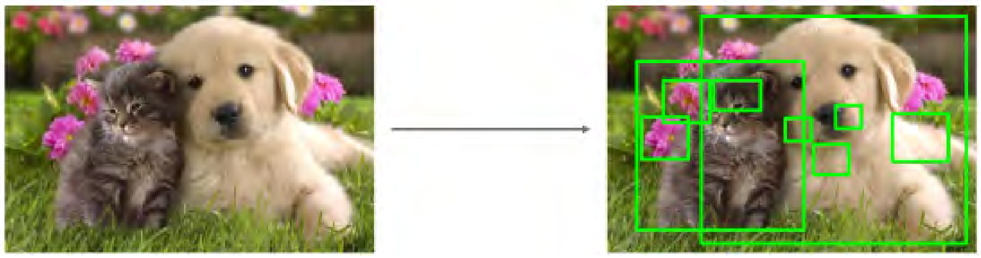


图 10.9: Region proposals

2.3 R-CNN

(1) Run region proposal method to compute ~2000 region proposals.

(2) Resize each region to 224x224 and run through CNN.

(3) Predict class scores and bbox transform.

(4) Use scores to select a subset of region proposals to output.

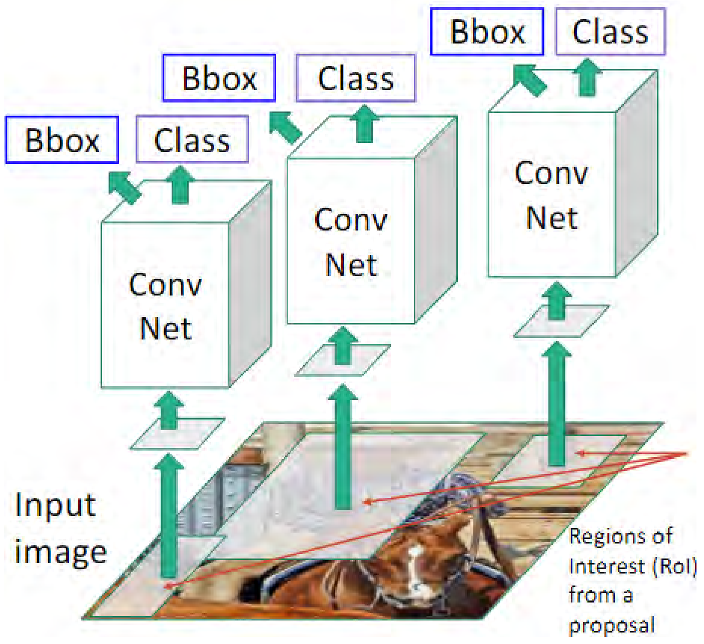


图 10.10: R-CNN

3. Evaluation metric

3.1 For detecting a single object

$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

- IoU>0.5 is decent.
- IoU>0.7 is pretty good.
- IoU>0.9 is almost perfect.

3.2 For Detecting multiple objects

- Positives and negatives:
  - True positive (TP)
  - False positive (FP)
  - True negative (TN)
  - False negative (FN)
- Precision(精度) = TP/(TP+FP)
- Recall(召回率) = TP/(TP+FN)

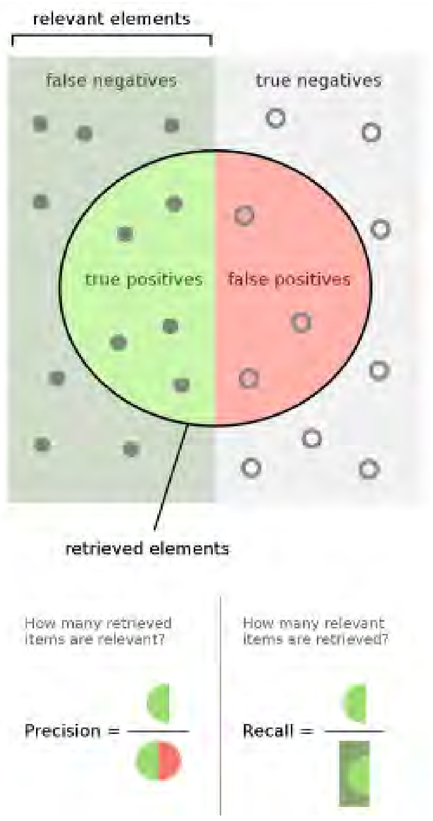


图 10.11: Precision and recall

3.3 Mean Average Precision (mAP)

(1) Run object detector on all test images.

(2) For each category, compute Average Precision(AP) = area under Precision vs Recall Curve.

a. For each detection (highest score to lowest score).

i. If it matches some ground-truth box with IoU > 0.5, mark it as positive and eliminate the ground-truth.

ii. Otherwise mark it as negative.

iii. Plot a point on PR Curve.

b. Average Precision (AP) = area under PR curve.

c. Mean Average Precision (mAP) = average of AP for each category.

d. For “COCO mAP”: Compute mAP for each IoU threshold(阈值) (0.5, 0.55, 0.6, ..., 0.95) and take average.

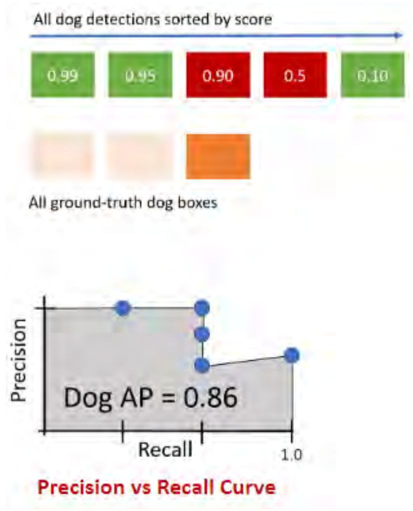

All dog detections sorted by score
0.99
0.95
0.90
0.5
0.10
All ground-truth dog boxes
Precision
Dog AP = 0.86
Recall
1.0
Precision vs Recall Curve

图 10.12: Mean Average Precision

To get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no “false positive” detections ranked above any ”true positives”.

3.4 Non-Max Suppression(非极大值抑制)

Object detectors often output many overlapping detections. (把和最大分数重合超过阈值的框去了)

(1) Select the highest-scoring box.

(2) Eliminate lower-scoring boxes with IoU > threshold.

(3) If any boxes remain, goto 1.

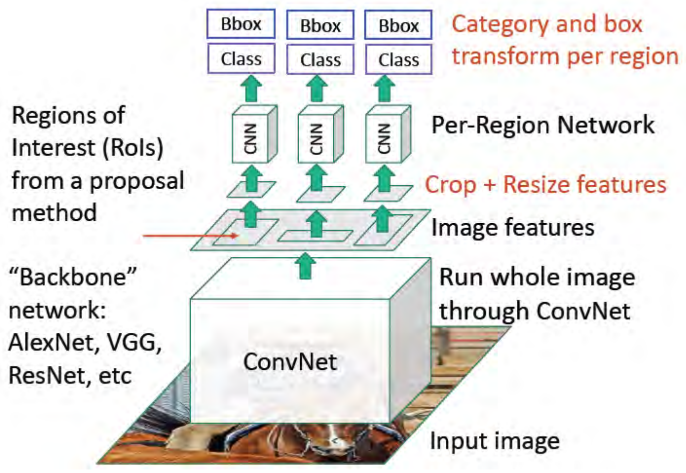

Bbox
Class
Bbox
Class
Bbox
Class
Category and box transform per region
Regions of Interest (RoIs) from a proposal method
CNN
CNN
CNN
Per-Region Network
Crop + Resize features
Image features
“Backbone” network: AlexNet, VGG, ResNet, etc
Run whole image through ConvNet
ConvNet
Input image

图 10.13: Fast R-CNN

4. Fast R-CNN

R-CNN 不够快 (本质是每个框单独过网络), 就有了 Fast R-CNN.

(1) “Backbone”(骨干网络): 特征提取, 先整张图过一次网络得特征图, 以此再用 proposal 过可以减少每次的网络深度.

(2) Regions of Interest (RoIs) pooling: 将不同的 proposal 同化

5. Faster R-CNN/Two-stage

Use CNN to select proposals, called Region Proposal Network (RPN). (加速了 proposal 生成的部分)

(1) Run once per image

  a. Backbone network

  b. RPN

(2) Run once per region

  a. Crop features RoI pool/align

  b. Predict object class

  c. Predict bbox offset

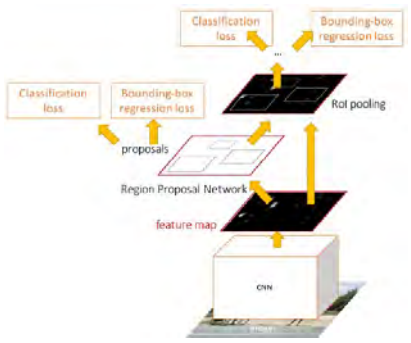


图 10.14: Faster R-CNN

5.1 RPN

Imagine an anchor box(算是候选的 bbox) of fixed size at each point in the feature map.
先生成 k 个框 (预先给予了), 再过网络判断 k 个框的好坏 (RPN). 过网络同时也调整了最好的 box.

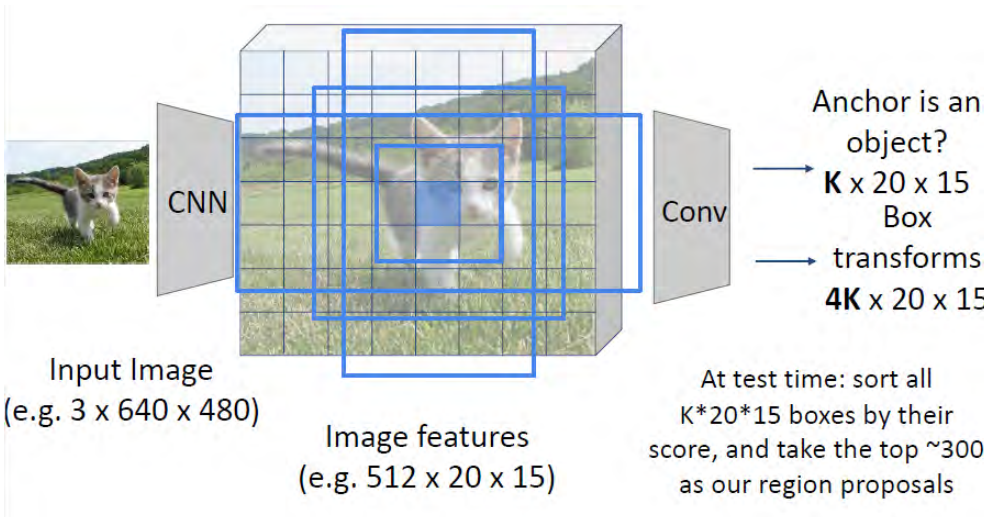


图 10.15: RPN

6. Single-stage object detection

RPN: Classify each anchor as object / not object.
Single-Stage Detector: Classify each object as one of C categories (or background).
(相当于直接在分类时判断类别)

6.1 YOLO detector

原作者只到了 v4

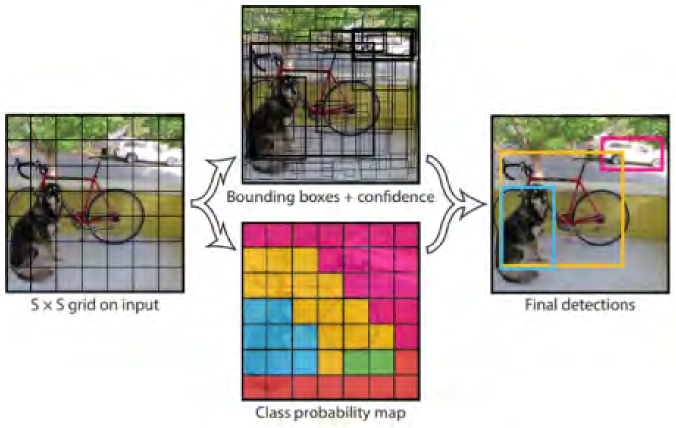


图 10.16: YOLO

7. Two-stage or Single-stage

Two-stage is generally more accurate. Single-stage is faster.

三、 Instance segmentation(实例分割)

Detect all objects in the image, and identify the pixels that belong to each object (Only things). Perform object detection, then predict a segmentation mask for each object.

1. Mask R-CNN

Faster R-CNN + additional head

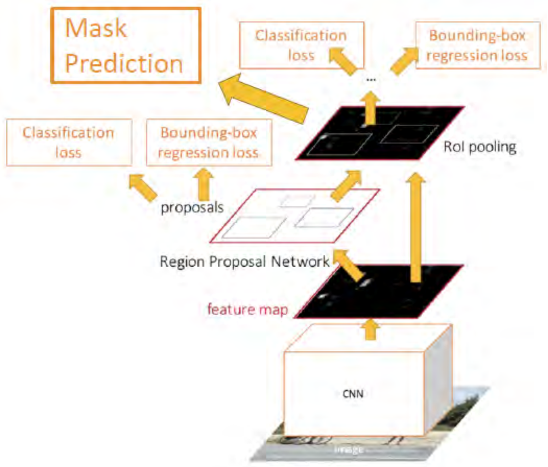


图 10.17: Mask R-CNN

2. Deep snake

传统的算法. 训网络调整框.

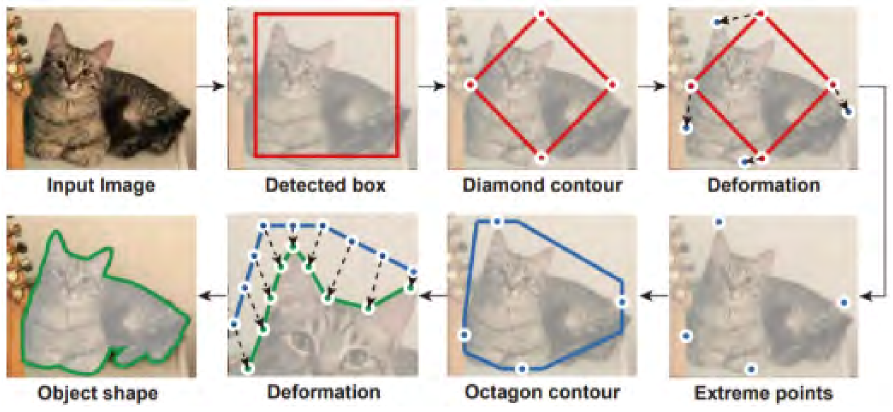


图 10.18: Deep snake

3. Panoptic segmentation(全景分割)

语义 + 实例 (背景也要分类)

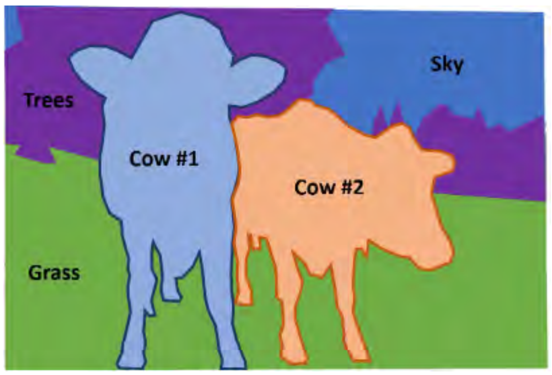


图 10.19: Panoptic segmentation

4. Datasets

Microsoft COCO (现在最常用的数据集)

- 118K train
- 5K val
- Contains
  - Object detection
  - Keypoints
  - Instance seg
  - Panoptic seg

四、 Human pose estimation

1. Represent the pose of a human

Represent the pose of a human by locating a set of keypoints.
e.g. 17 keypoints:

- Nose
- Left / Right eye
- Left / Right ear
- Left / Right shoulder
- Left / Right elbow
- Left / Right wrist
- Left / Right hip
- Left / Right knee
- Left / Right ankle

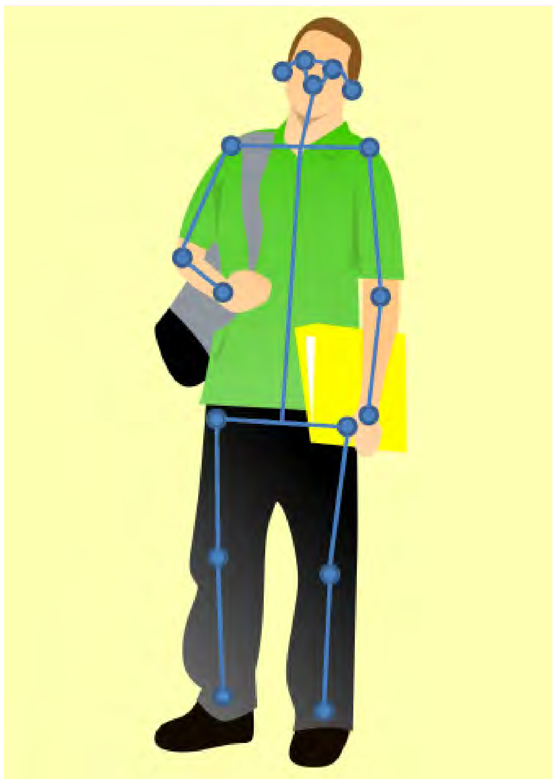


图 10.20: Represent the pose of a human

2. Human pose estimation with depth sensors

Microsfot Kinect, 基于深度相机识别人体关键部位.

3. Single human

3.1 Directly predict

Directly predict joint locations (a 17*2 vector).
图转向量需要很多全连接层, 难以训练, 图像到坐标的映射也难以学习.

3.2 heatmap

Represent joint location as the heatmap(热力图).
一个 heatmap 代表一个关键点. 输出变成图了, 只需要卷积了.

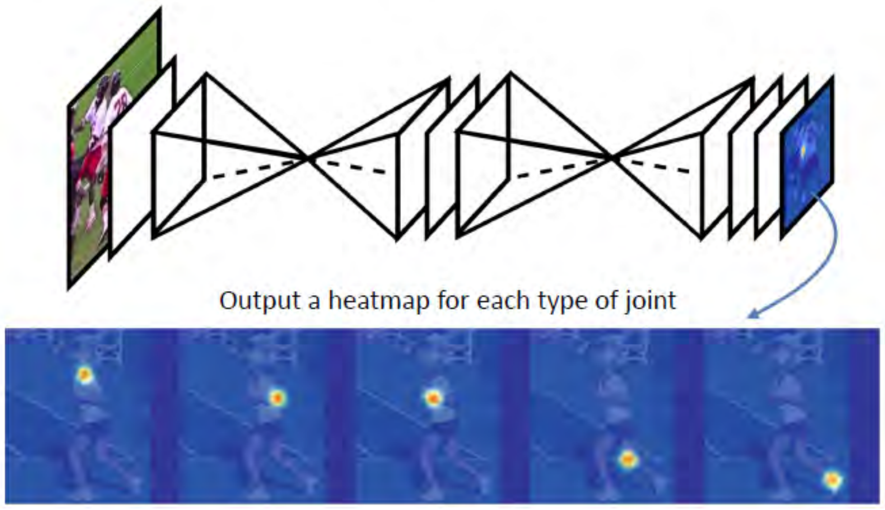


图 10.21: heatmap

4. Multiple humans

4.1 Top-down

Detect humans and detect keypoints in each bbox. Example: Mask R-CNN. 最大问题还是慢了.

4.2 Bottom-up

Detect keypoints and group keypoints to form humans.Example: OpenPose.

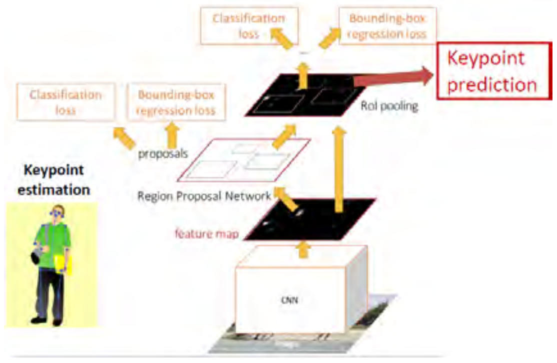


图 10.22: Mask R-CNN-pose

(1) Part affinity fields.

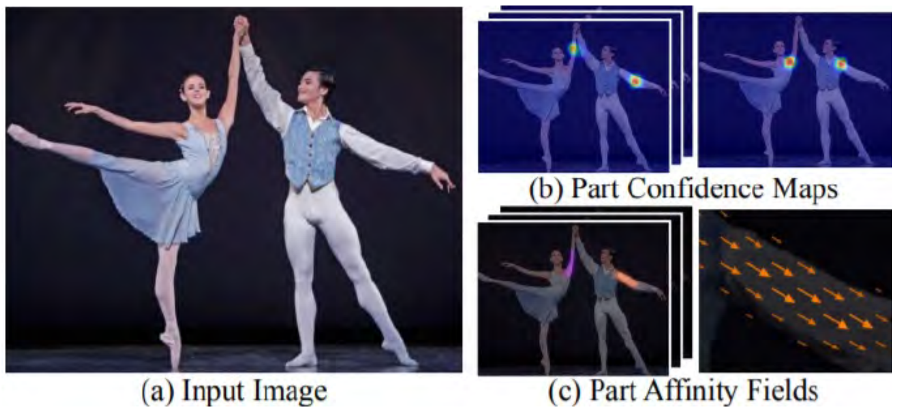


图 10.23: Part affinity fields

(2) Link parts based on part affinity fields.

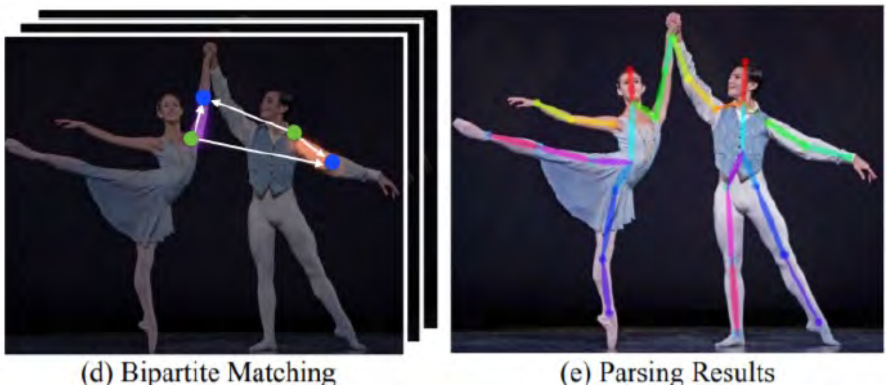


图 10.24: Link parts

5. Top-down or bottom-up

Top-down is generally more accurate. Bottom-up is faster. (Bottom-up 在有遮挡下也有优势)

五、 Other tasks

1. Video classification

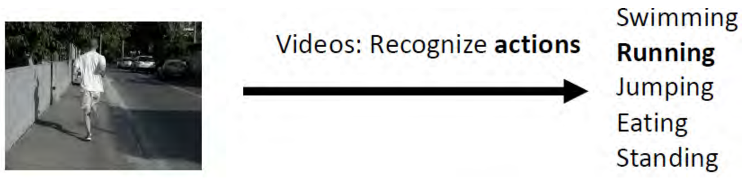


图 10.25: Video classification

Use 3D CNN. (时间也是个维度)

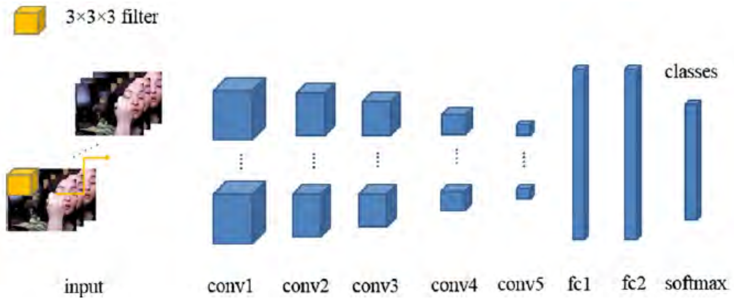


图 10.26: 3D CNN

2. Temporal action localization

Given a long untrimmed video sequence, identify frames corresponding to different actions. Generate proposals then classify.

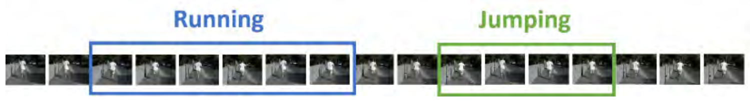


图 10.27: Temporal action localization

3. Spatial-temporal detection

Given a long untrimmed video, detect all the people in space and time and classify the activities they are performing.

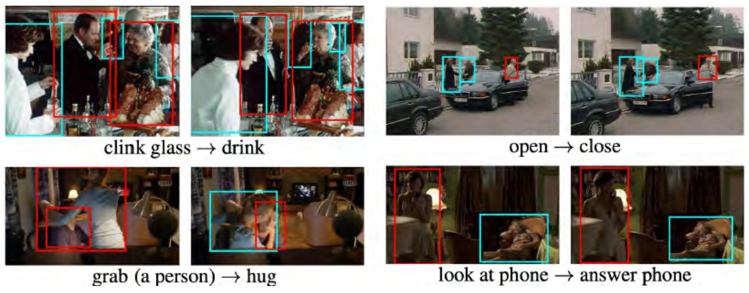


图 10.28: Spatial-temporal detection

4. Multi-object tracking

Identify and track objects belonging to one or more categories without any prior knowledge about the appearance and number of targets.

5. Papers with code

paperswithcode.com
去看本领域较优的方法, 可以参考 (它只是个非专业网站, 排名有局限性, 不可全信), 需要进一步的挖掘的.

第十一章　3D Deep Learning

一、　Deep Learning for 3D Reconstruction

1.　Feature matching

1.1　Learning to estimation pose

Using networks to directly predict pose is not good. (不准确且不稳定)
网络的功能之一是记忆, 这个与泛化性相关.
Use deep learning to improve feature matching.

1.2　Recap: featrue matching pipline

$$\begin{tikzcd}
\boxed{\text{Images}} \arrow[r] & \boxed{\text{Extract Features}} \arrow[r] & \boxed{\text{Match Features}}
\end{tikzcd}$$

1.Detection　　　　3. Matching
2.Description

图 11.1: Feature matching pipline

1.3　Why deep learning

Traditional feature detectors and descriptors are handcrafted (人工定义) (e.g. DoG, SIFT, ...)
Limitations: Geometry only, no semantics, no semantics, not robust.

1.4　Deep learning for feature matching

Example: SuperPoint

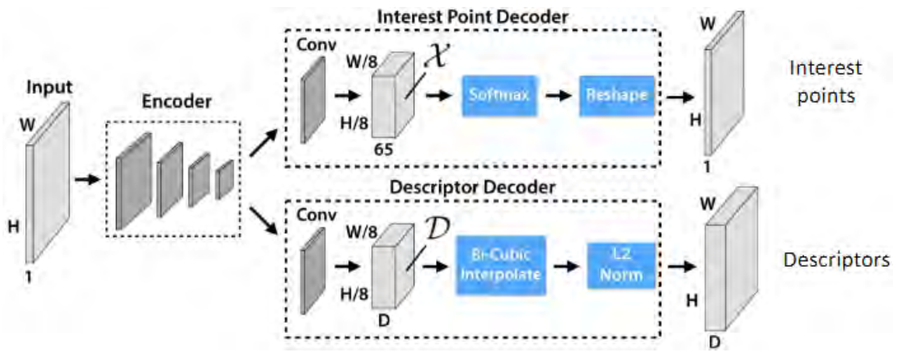


图 11.2: SuperPoint

(1) CNN-based detectors: Representing feature point locations by heatmaps (热力图).

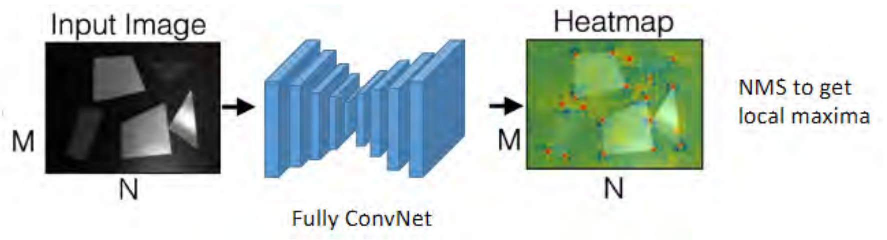


图 11.3: CNN-based detectors

For training detectors:

a. Training CNNs to detect corners.

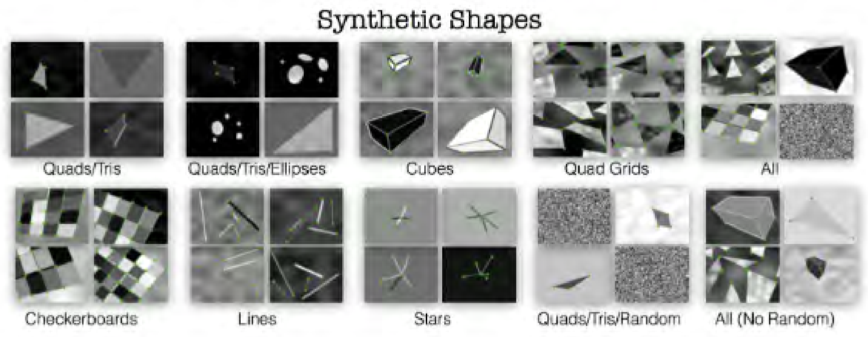


图 11.4: detect corners

$$L_{loc} = \sum_i \|p_t^* - \hat{p}_t\|_2$$

b. Training CNNs to enforce repeatability (不变性)

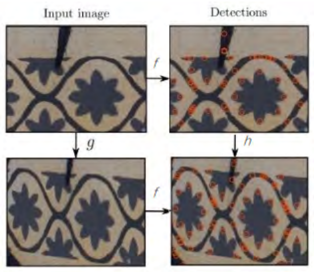


图 11.5: enforce repeatability

i. Warp image

ii. Enforce equivariance

$$\min_f \frac{1}{n} \sum_{i=1}^{n} \|f(h(I)) - g(f(I))\|^2$$

(2) CNN-based descriptors: Extract descriptors from CNN feature maps.

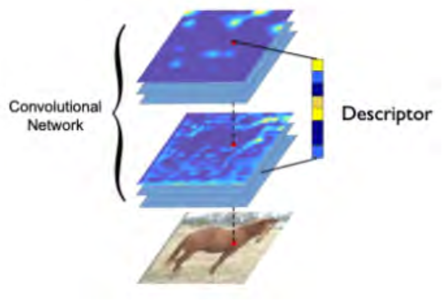


图 11.6: CNN-based descriptors

For training descriptors: Training descriptors by metric learning

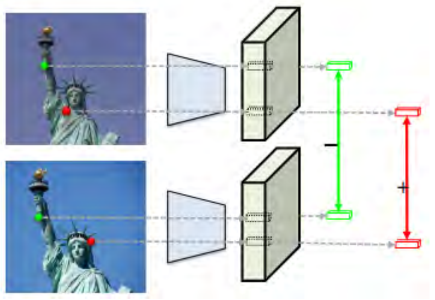


图 11.7: metric learning

a. Contrastive loss (最小化对应点距离, 最大化不对应点的距离)

$$L_{pos} = \frac{1}{N}\sum_{i=1}^{N} \|F_I(A) - F_{I'}(P)\|^2$$

$$L_{neg} = \frac{1}{N}\sum_{i=1}^{N} \max(0, m - \|F_I(A) - F_{I'}(N)\|)^2 \text{ 取 max 避免无限制的优化}$$

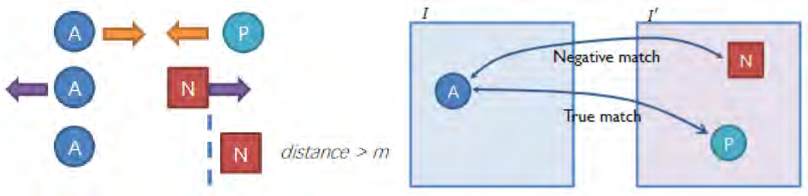


图 11.8: Contrastive loss

b. Triplet loss

$$L_{tri} = \frac{1}{N}\sum_{i=1}^{N} \max(0, m + \|F_I(A) - F_{I'}(P)\| - \|F_I(A) - F_{I'}(N)\|)^2 \text{ 也同样避免无限制的优化}$$

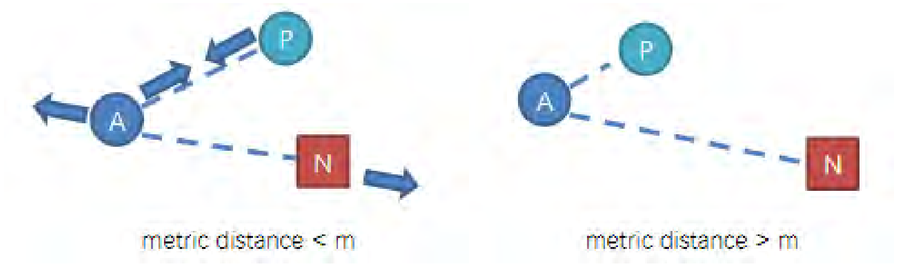
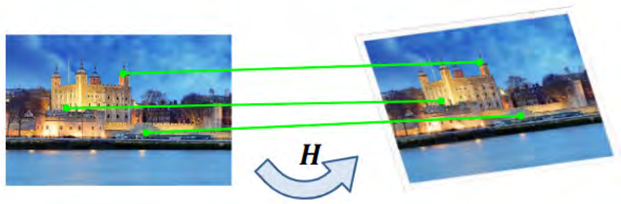


图 11.9: Triplet loss

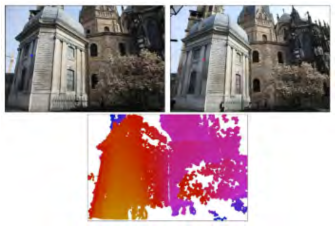
二者效果差不多, 主要是收敛上的差别.

Where is training data from?

- Synthetic data. (但数据不真实)
- Use MVS.



(a) Synthetic data



(b) MVS

The state of the art is SuperGlue.

2. Object Pose Estimation

Estimate the 3D location and orientation of an object realtive to the camera frame.

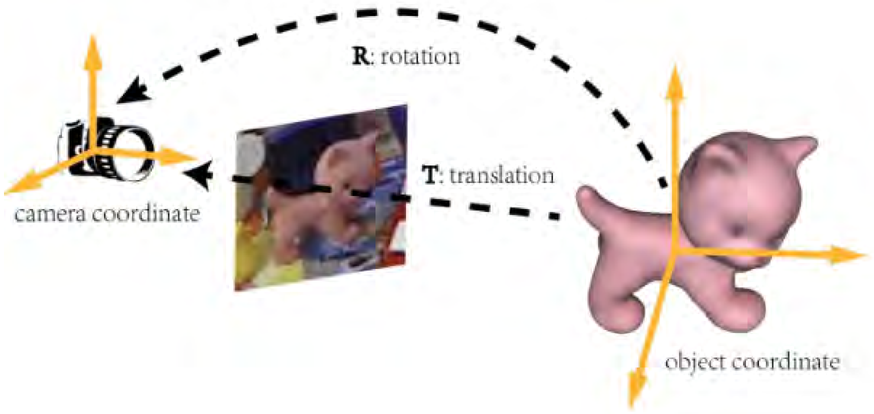


图 11.11: 3D location and orientation

Applications: robot grasping, autonomous driving, augmented reality (AR).
Similar to visual localization:

(1) Find 3D-2D correspondences.

(2) Solve R and t by perspective-n-point (PnP) algorithm.

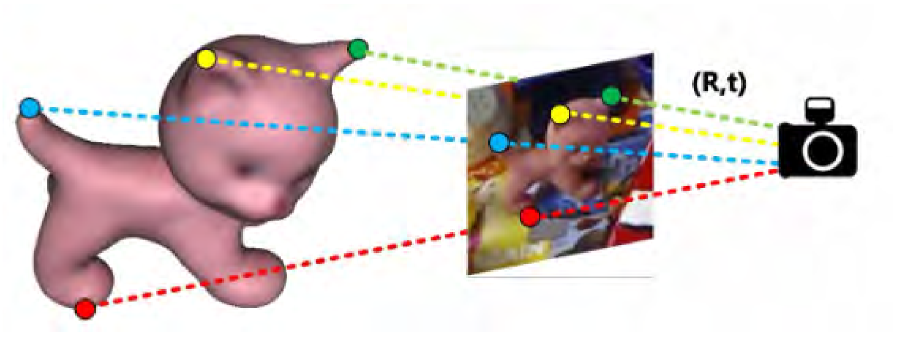


图 11.12: Object Pose Estimation

2.1 Feature-matching-based methods

(1) Reconstruct object SfM model by input multi-view images.

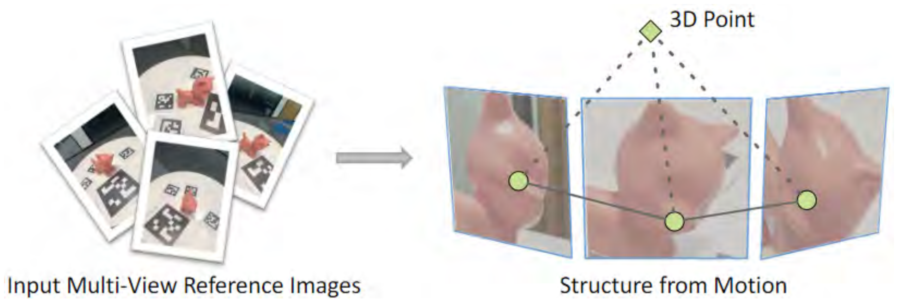


图 11.13: Reconstruct object SfM model

(2) Obtain 2D-3D correspondeces by lifting 2D-2D matches to 3D

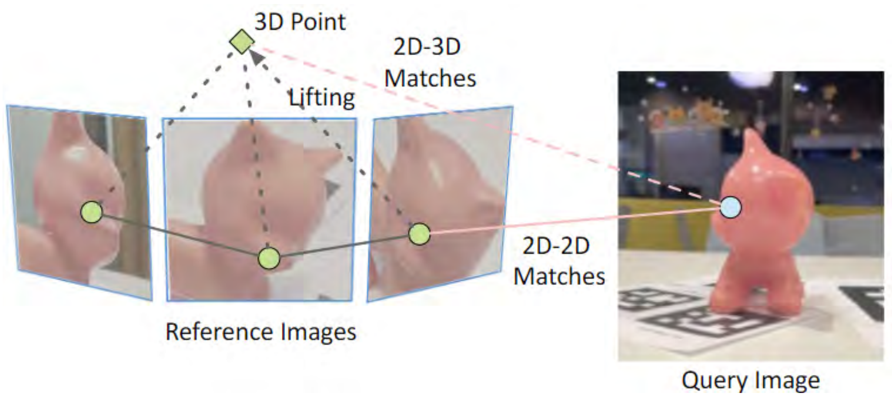


图 11.14: Obtain 2D-3D correspondeces

(3) Object pose of query image can be solved by PnP.

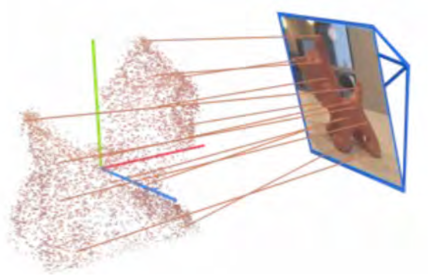


图 11.15: Object pose

(4) Futher reading list

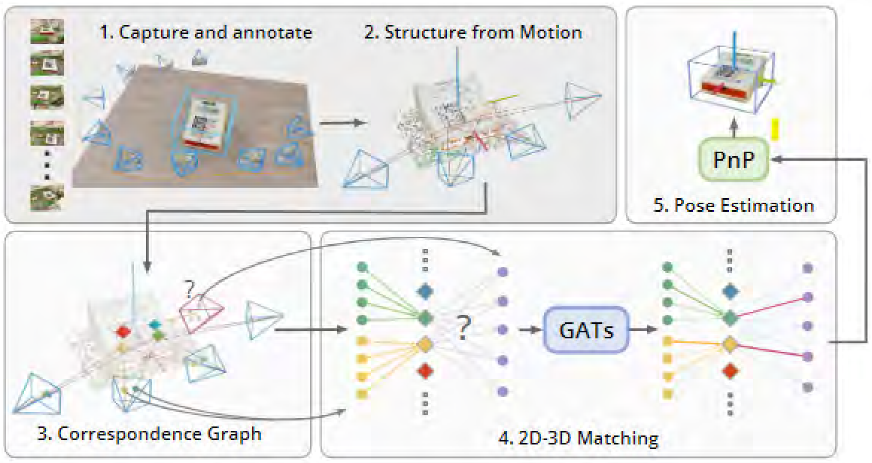


图 11.16: Futher reading list

2.2　Direct Pose Regression Methods

Directly regressing object pose of query image using a neural network. Need to render a large amount of images for training.

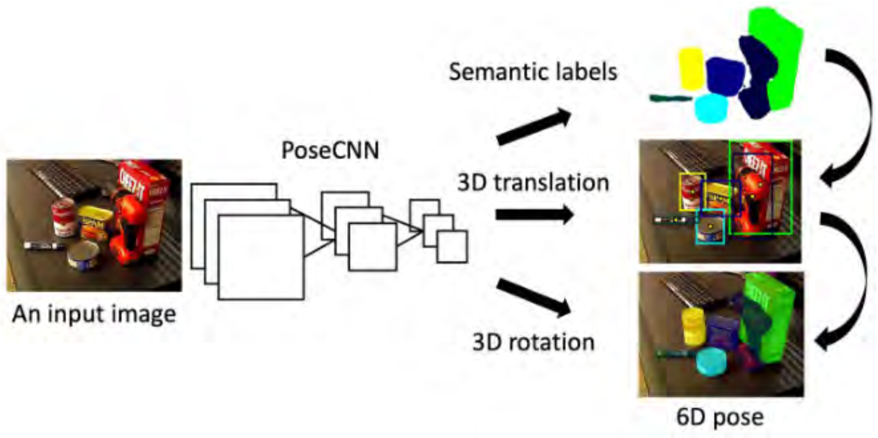


图 11.17: Direct Pose Regression Methods

2.3 Keypoint detection methods

Using a CNN to detect pre-defined keypoints. Also need to render a large amount of images for training.

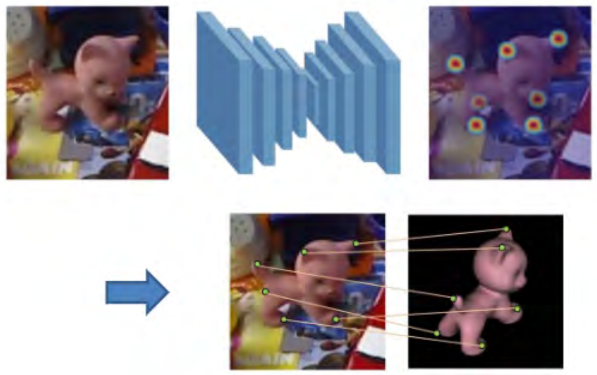


图 11.18: Keypoint detection methods

2.4 Pose refinement methods

Given estimated pose by above mentioned methods as an initialization. Refine the pose by optimization to obtain more accurate results.

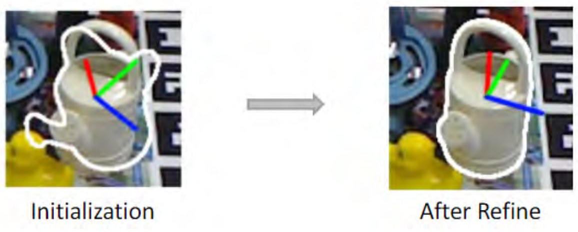


图 11.19: Pose refinement methods

3. Human Pose Estimation

3.1 3D Human Pose Estimation

- 2D Human Pose Estimation: Localize human joints (keypoints -elbows, wrists, etc) in images.
- 3D Human Pose Estimation: Estimate a 3D x, y, z coordinate for each joint from image.

3.2 Marker-based MoCap system

Optical motion capture (MoCap) system.

图 11.20: MoCap

3.3　Markerless MoCap

Multiview 3D Human Pose Estimation.

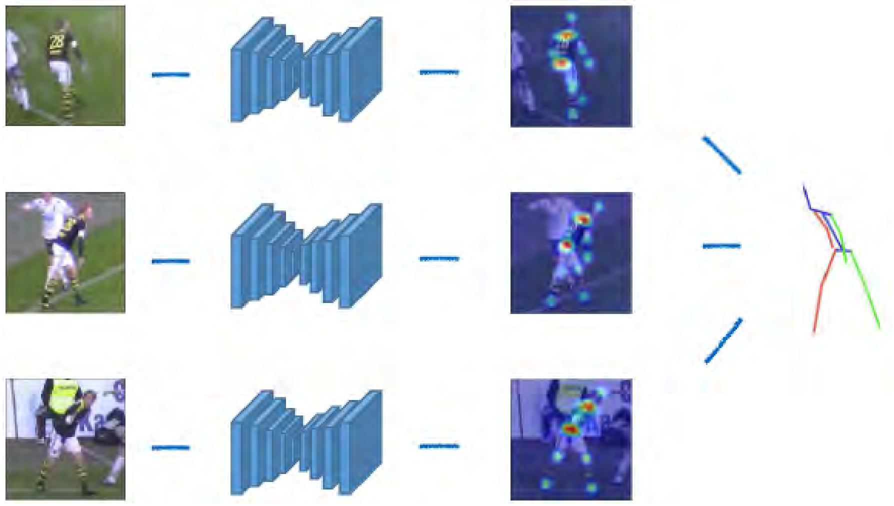


图 11.21: Markerless MoCap

3.4　Monocular 3D Human Pose Estimation

Estimating 3D human pose using a single camera. (如何设计网络, 与如何获取数据, 或者获取人体参数化模型参数, 往往是关节的角度)

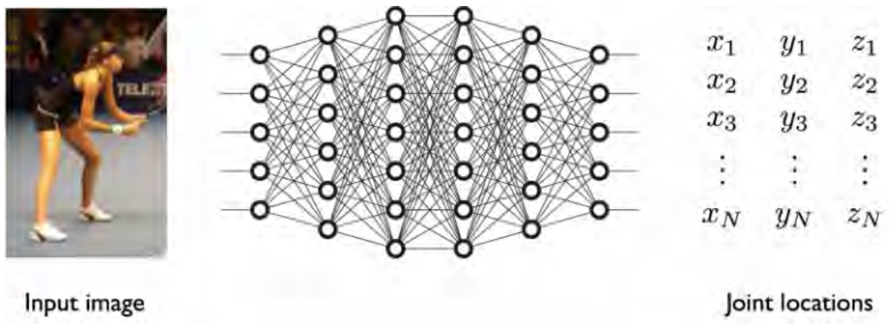


图 11.22: Monocular 3D Human Pose Estimation

Mehta, Dushyant, et al. ”Vnect: Real-time 3d human pose estimation with a single rgb camera.” ACM Transactions on Graphics (TOG) 36.4 (2017): 1-14. 与 https://github.com/mkocabas/VIBE.

4.　Dense Reconstruction

4.1　Traditional pipeline

(1) Compute depth map per image (multi-view stereo)

(2) Fuse the depth maps into a 3D surface (Poisson reconstruction)

(3) Texture mapping

4.2　Recap: multi-view stereo

Find the depth value that gives the smallest error.

4.3 Learning MVS

Challenges for traditional methods:

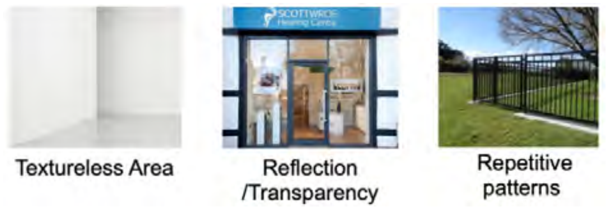


图 11.23: Challenges

MVSNet: cost volume from CNN features

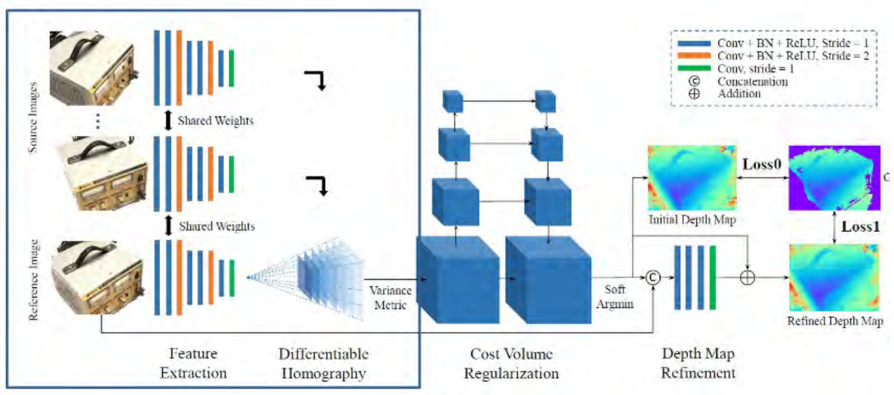


图 11.24: MVSNet

Can we improve the mesh quality by comparing the rendered images with the input images?

Yes, but not easy. The rendering process is not differentiable. And mesh representation is not a good representation for optimization.

5. Single image to 3D

Infer 3D representation from just single image. (Depth, Point Cloud, Meshm Volume)

5.1 Monoculer depth estimation

Using network to guess depth from single image. 使用深度摄像机的照片做训练数据.

5.2 Single-view shape estimation

3D shape (point cloud, mesh) from a single image. (目前还不是很行, 且单目信息有限, 现训练数据较少)

二、 Deep Learning for 3D Understanding

1. 3D Representations

Explicit Representations and Implicit Representations

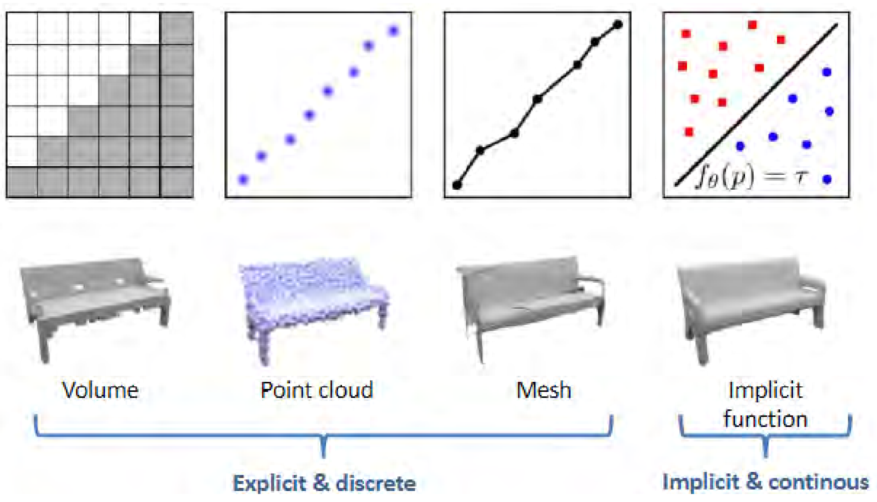


图 11.25: 3D Representations

1.1 Implicit Representations

A simple example: $f(p) = 1$ represents a sphere where

$$f(p) = p_1^2 + p_2^2 + p_3^2$$

Generally, the implicit function can be:

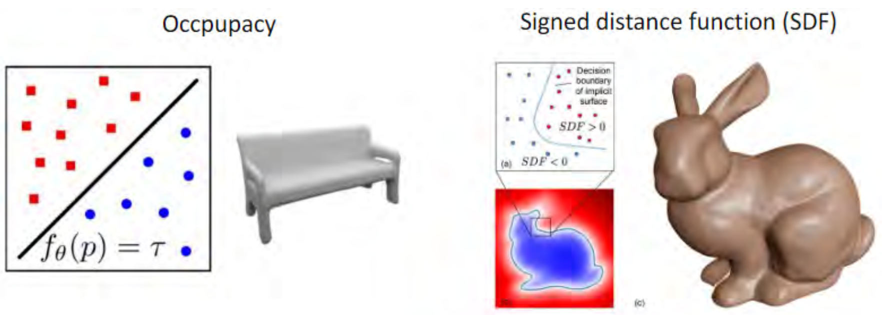

Occpupacy
Signed distance function (SDF)
$f_\theta(p) = \tau$
Decision boundary of implicit surface
$SDF > 0$
$SDF < 0$

图 11.26: Implicit Representations

But in practice, the form of $f_\theta(p)$ is very hard to define.

1.2 Implicit Neural Representations

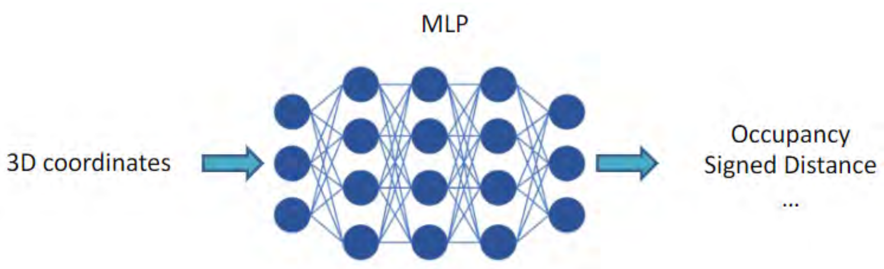


图 11.27: Implicit Neural Representations

1.3 Neural Radiance Fields (NeRF)

Representing scenes as radiance fields.

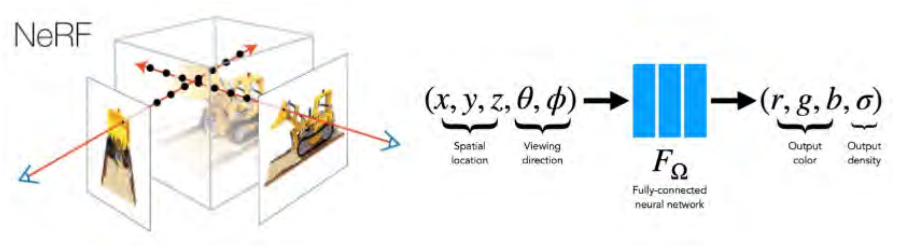

NeRF
$(x, y, z, \theta, \phi) \rightarrow F_\Omega \rightarrow (r, g, b, \sigma)$
Spatial location
Viewing direction
Fully-connected neural network
Output color
Output density

图 11.28: NeRF

The radiance filed can be converted to images by volume rendering, which is differentiable.

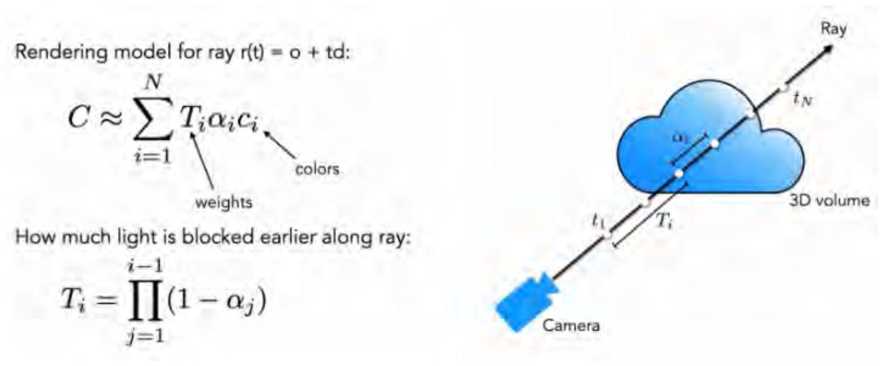

Rendering model for ray r(t) = o + td:
$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$
colors
weights
How much light is blocked earlier along ray:
$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$
Ray
t_N
α_i
3D volume
t_1
T_i
Camera

图 11.29: volume rendering

Reconstructing NeRF from images by minimizing rendering loss.

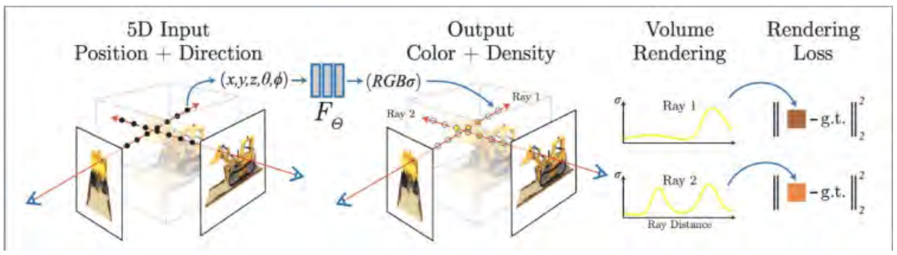


图 11.30: Reconstructing NeRF from images

But NeRF has poor surface reconstruction quality.

1.4 NeuS

Replacing density field by SDF.

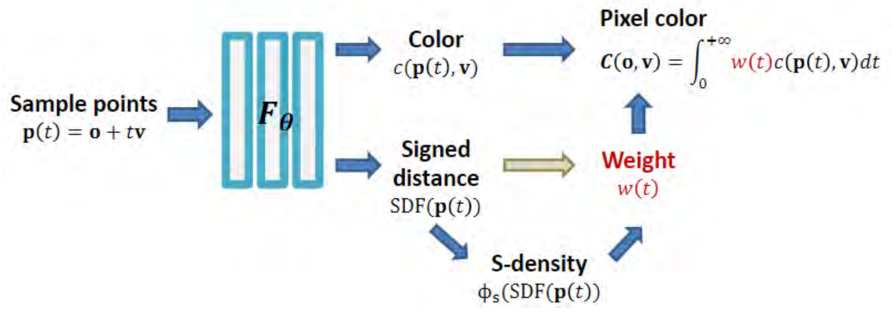

Sample points
$\mathbf{p}(t) = \mathbf{o} + t\mathbf{v}$
F_θ
Color
$c(\mathbf{p}(t), \mathbf{v})$
Signed distance
$\mathrm{SDF}(\mathbf{p}(t))$
S-density
$\phi_s(\mathrm{SDF}(\mathbf{p}(t)))$
Weight
$w(t)$
Pixel color
$C(\mathbf{o}, \mathbf{v}) = \int_0^{+\infty} w(t)c(\mathbf{p}(t), \mathbf{v})dt$

图 11.31: NeuS

2. 3D Classification

- Input: 3D shape
- Output: Class

2.1 Multi-view CNNs

(1) Given an input shape

(2) Render with multiple virtual cameras

(3) The rendered images are passed through CNNs

(4) All image features are pooled and then passed through CNNs and to generate final predictions

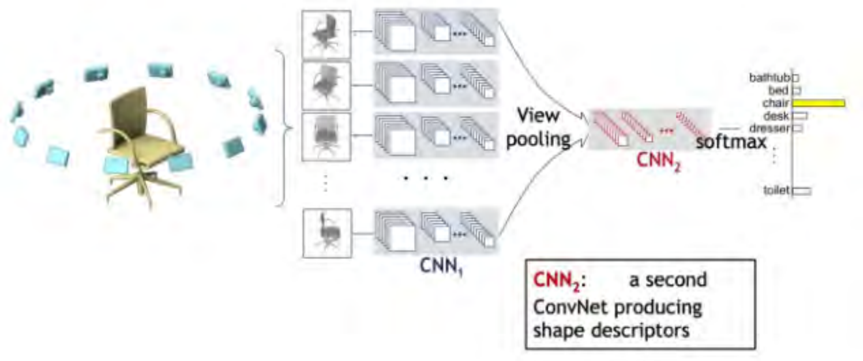


图 11.32: Multi-view CNNs

3. Deep learning on volumetric data

3.1 3D ConvNets

Deep learning on volumetric data. Process volumetric data using 3D convlution.
Challenge: High space/time complexity of high resolution voxels: $O(N^3)$.

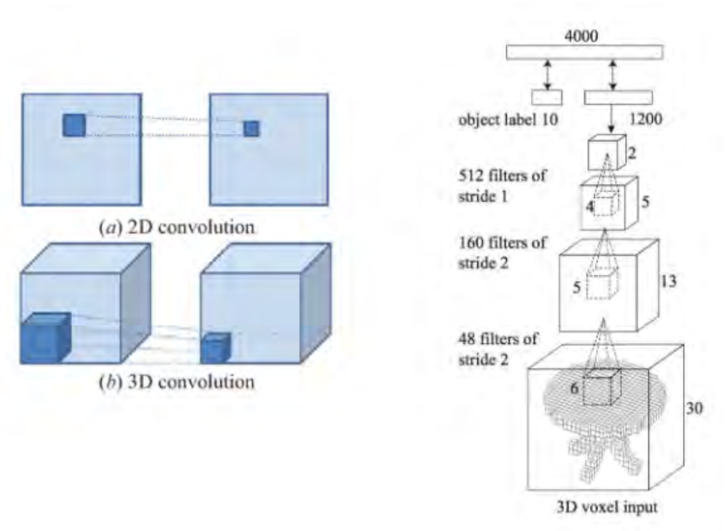


图 11.33: 3D ConvNets

3.2 Sparse ConvNets

Idea: using sparity of 3D shapes. (三维空间中被占据的部分是比较少的)

(1) Store the sparse surface signals (Octree, 八叉树)

(2) Constrain the computation near the surface

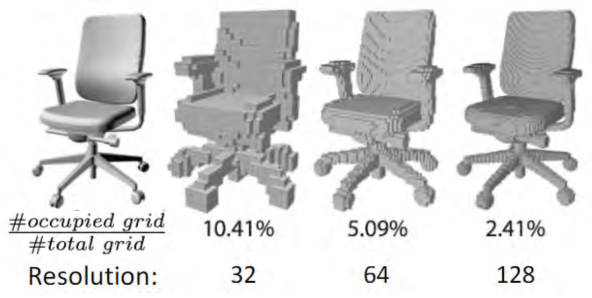
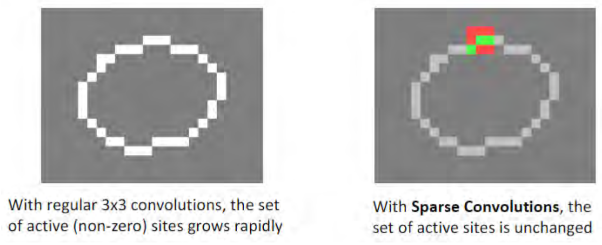


图 11.34: sparity of 3D shapes

Sparse convolution: compute inner product only at the active sites (nonzero entries)



With regular 3x3 convolutions, the set of active (non-zero) sites grows rapidly
With Sparse Convolutions, the set of active sites is unchanged

图 11.35: Sparse ConvNets

4. Deep learning on point clouds

Challenges:

- Point cloud is unrasterized data (非栅格化)
- Convolution cannot be applied.

4.1 PointNet

A point cloud processing architecture for multiple tasks (classification, detection, segmentation, registration, etc).

Point cloud: N orderless points, each represented by a D dim coordinate.

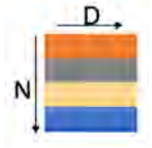


图 11.36: 2D array representation

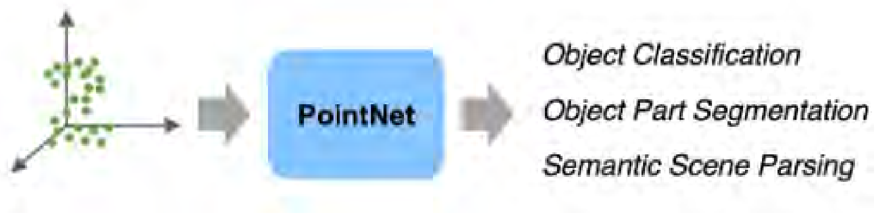


图 11.37: PointNet

PointNet Classification and Segmentation Architecture.

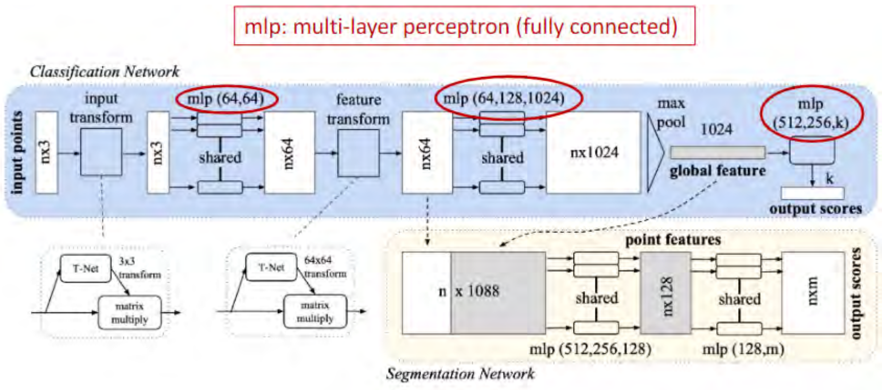


图 11.38: PointNet Network

(1) Challenge 1: the point set is order-less. The output needs to be invariant to N! permutations. (对点的顺序需要有不变性)

Solution: max pooling makes the output invariant to the order of input points.

(2) Challenge 2: the output should be invariant to rigid transformation of points. (对模型的位置需要有不变性)

Solution: estimate the transformation using another network (T-Net).

4.2 PointNet++

Limitations of PointNet: No local context for each point.
Three Parts:

(1) Sampling: Sample anchor points by Farthest Point Sampling (FPS)

(2) Grouping: Find neighbourhood of anchor points

(3) Apply PointNet in each neighborhood to mimic convolution

5. 3D semantic segmentation

Input: sensor data of a 3D scene (RGB/depth/point cloud .....)
Output: Label each point in point cloud with category label.
Possible solution: directly segmenting the point cloud

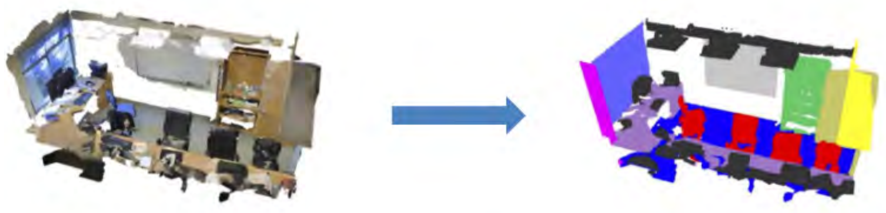


图 11.39: 3D semantic segmentation in point cloud

Input: sensor data of a 3D scene (RGB/depth/point cloud .....)
Output: Label each point in point cloud with category label.
Possible solution: fuse 2D segmentation results in 3D

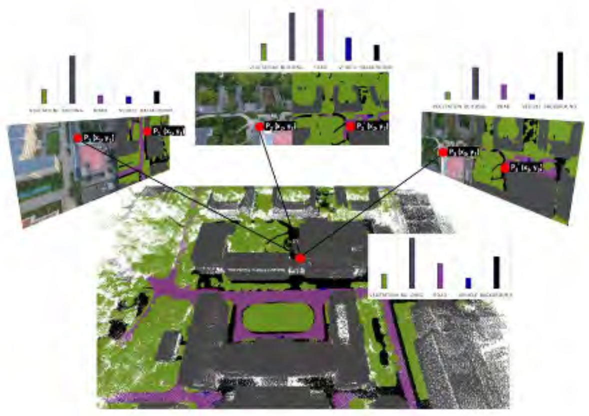


图 11.40: 3D semantic segmentation in multi-view

6. 3D Object Detection

Detecting 3D objects in 3D data.

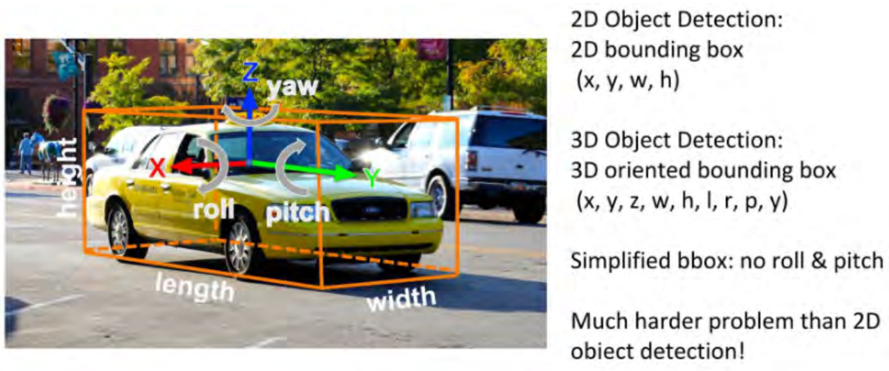


图 11.41: 3D bounding boxes

(1) Classify sliding windows. Con: 3D CNNs are very costly in both memory and time.

(2) PointRCNN: RCNN for point cloud

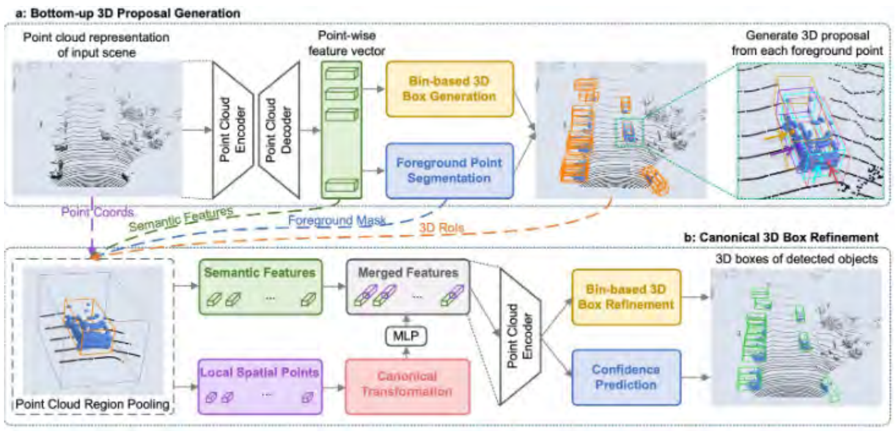

a: Bottom-up 3D Proposal Generation
Point cloud representation of input scene
Point Cloud Encoder
Point Cloud Decoder
Point-wise feature vector
Bin-based 3D Box Generation
Foreground Point Segmentation
Generate 3D proposal from each foreground point
Point Coords.
Semantic Features
Foreground Mask
3D RoIs
b: Canonical 3D Box Refinement
Point Cloud Region Pooling
Semantic Features
Merged Features
MLP
Local Spatial Points
Canonical Transformation
Point Cloud Encoder
Bin-based 3D Box Refinement
Confidence Prediction
3D boxes of detected objects

图 11.42: PointRCNN

(3) Frustum PointNets: Using 2D detectors to generate 3D proposals. (视锥)

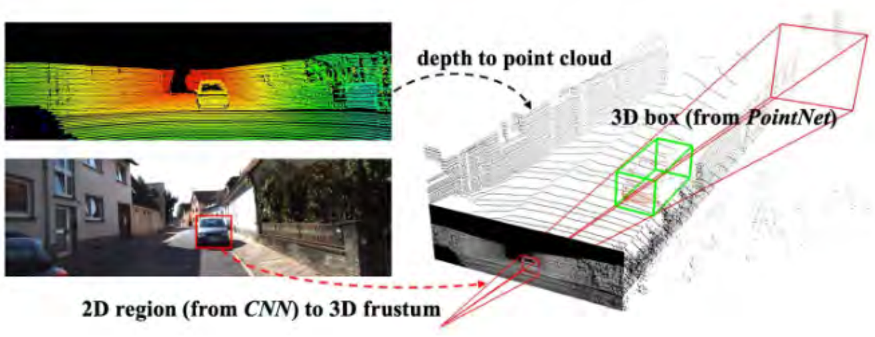


图 11.43: Frustum PointNets

7. 3D Instance Segmentation

Input: 3D point cloud

Output: instance labels of 3D points

7.1　Top-down approach

(1) Run 3D detection

(2) Run segmentation in each 3D bbox

7.2　Bottom-up approach

Group (cluster) points into different objects

8.　Datasets for 3D Objects

(1) Large-scale Synthetic Objects: ShapeNet

(2) Fine-grained Part: PartNet (ShapeNetPart2019)

- Fine-grained (towards mobility)
- Instance-level
- Hierarchical

(3) Large-scale Synthetic Scenes: SceneNet

For Out door 3D Scenes

(1) KITTI: LiDAR data, labeled by 3D bboxes

(2) Semantic KITTI: LiDAR data, labeled per point

(3) Waymo Open Dataset: LiDAR data, labeled by 3D b.boxes

第十二章　Computational Photography

Data recorded by sensor is not the final image. Computational Photography: arbitrary computation to the final image.

一、　High Dynamic Range Imaging (HDR)

高动态范围成像

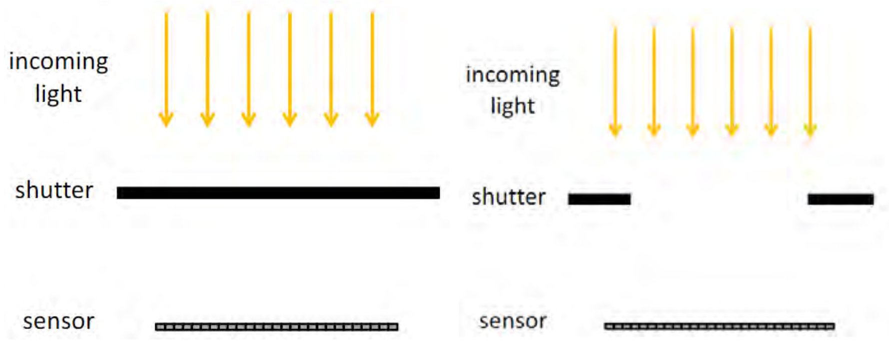
图 12.1: HDR

1.　Exposure

Roughly speaking, the “brightness” of a captured image given a fixed scene.
Exposure = Gain(增益) x Irradiance x Time
Gain is controlled by the ISO. Irradiance is controlled by the aperture. Time is controlled by the shutter speed.

1.1　shutter speed

Side-effects of shutter speed: moving scene elements appear blurry.

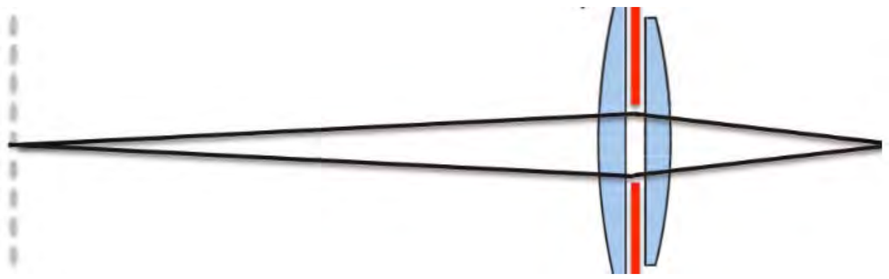


(a) shutter speed close　　(b) shutter speed open

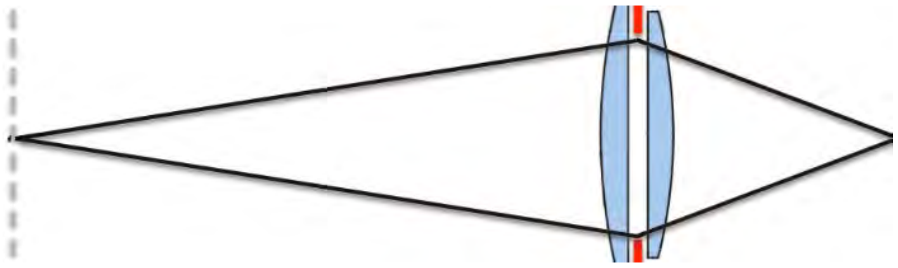
图 12.2: shutter speed

1.2　aperture

Side-effects of aperture: depth of field (bokeh)



(a) small aperture



(b) big aperture

图 12.3: aperture

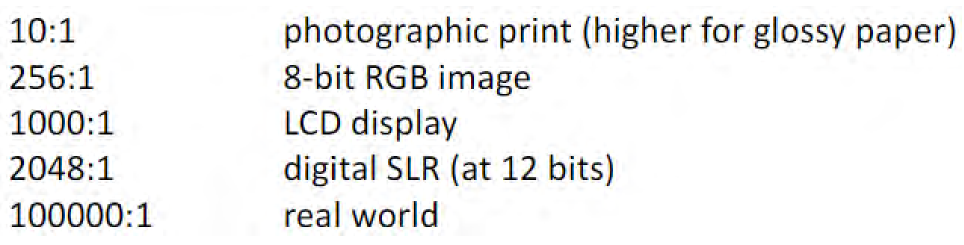
1.3　ISO

ISO 越大图越亮, 但噪声也越多.

When taking a photo, the averaged exposure should be at the middle of the sensor’s measurement range. So that the photo has both bright and dark parts with details.

2.　Dynamic range

The ratio between the largest and smallest values of a certain quantity (e.g., brightness).



10:1	photographic print (higher for glossy paper)
256:1	8-bit RGB image
1000:1	LCD display
2048:1	digital SLR (at 12 bits)
100000:1	real world

图 12.4: Dynamic range

3.　Key idea

(1) Exposure bracketing: Capture multiple LDR images at different exposures.

(2) Merging: Combine them into a single HDR image.

3.1 Image formation model

Suppose scene radiance for image pixel (x, y) is $L(x, y)$.

$$I(x, y) = clip\left[t_i \cdot L(x, y) + noise\right]$$

clip 截断超过相机的数值. Exposure time is t_i.

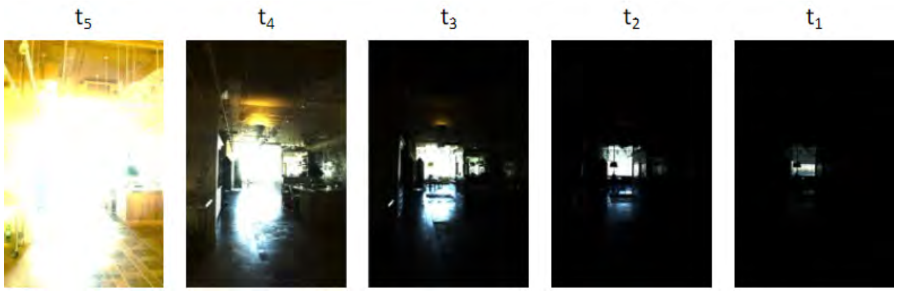


图 12.5: Exposure time

3.2 Merging images

For each pixel:

(1) Find “valid” pixels in each image.

$$\text{(noise)}0.05 < \text{pixel} < 0.95\text{(clipping)}$$

(2) Weight valid pixel values appropriately.

$$\text{weight} = \frac{\text{pixel value}}{t_i}$$

(3) Form a new pixel value as the weighted average of valid pixel values

3.3 Merging result

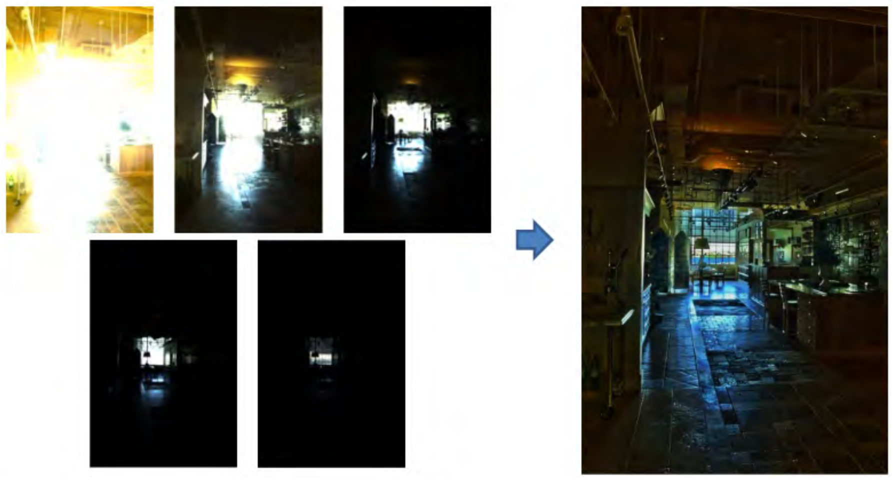


图 12.6: Merging result

4. Display the HDR image

how to display the HDR image (e.g. 12-bit) on a SDR(standard dynamic range, e.g., 8-bit) device? tone mapping.

4.1 Tone mapping

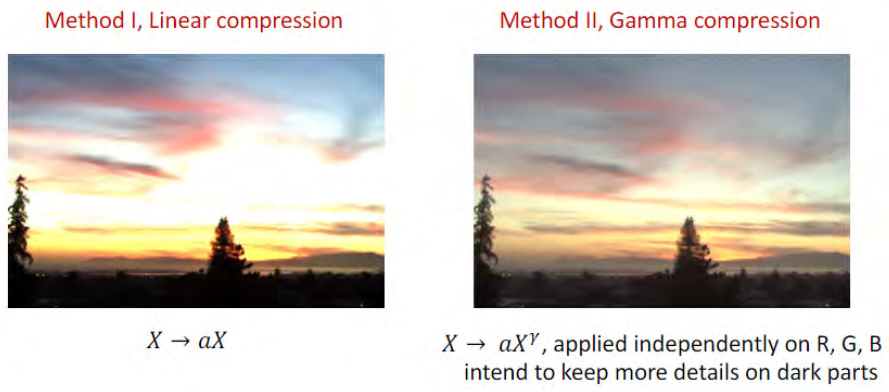

Method I, Linear compression
Method II, Gamma compression
$X \to aX$
$X \to aX^{\gamma}$, applied independently on R, G, B
intend to keep more details on dark parts

图 12.7: Tone mapping

Gamma compression:$y = ax^{\gamma}$

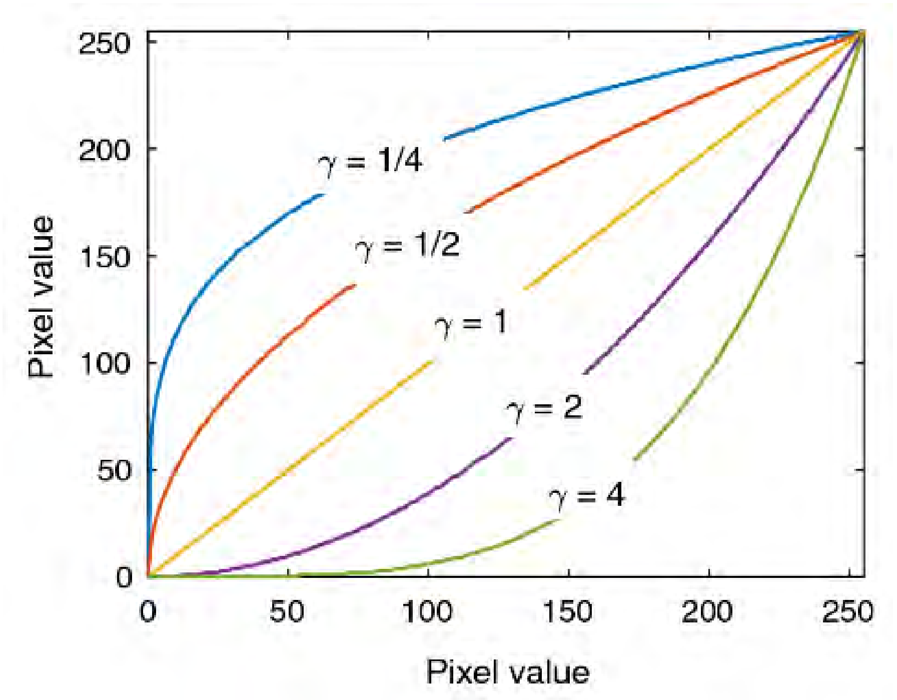
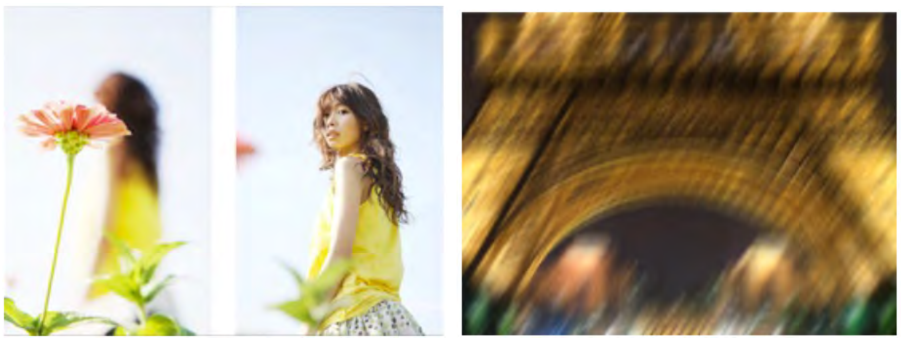


图 12.8: Gamma compression

二、 Deblurring

1. Reason of blurring

- Defocus: the subject is not in the depth of view.
- Motion blur: moving subjects or unstable camera.



(a) Defocus　　(b) Motion blur

图 12.9: Blurring

2. Get a clear image

Accurate focus, Fast shutter speed

- Large aperture
- High ISO

One of the reasons why SLR cameras and lenses are expensive.

(1) Use hardware

- tripod
- not portable!

(2) Use hardware

- optical image stabilization
- IMU
- expensive!

3. Modeling image blur

The blurring process can be described by convolution. The blurred image is called convolution kernel.

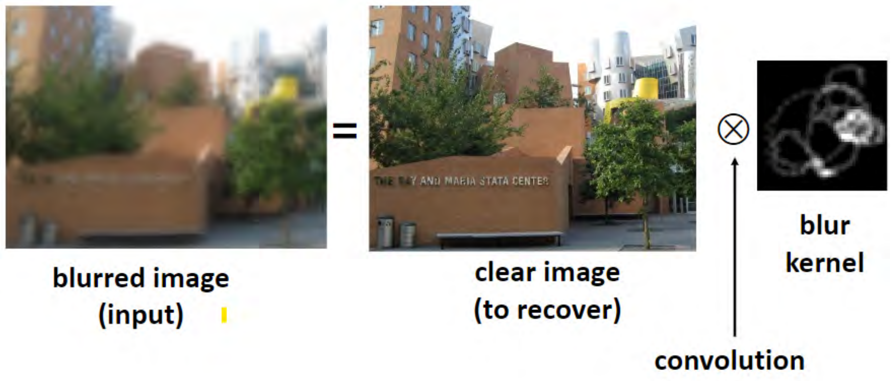


图 12.10: Modeling image blur

The blur pattern of defocusing depends on the aperture shape. The blur pattern of shaking depends on the camera trajectory.

Deblurring = Deconvolution

4. Non-blind image deconvolution(NBID)

$$G = F \bigotimes H$$

- G: The captured image (known)
- F: Image to be solved (unknown)
- H: Convolution kernel (known)

4.1 Frequency domain deconvolution

Convolution in the spatial domain = product in the frequency domain.

$$\mathcal{F}(G) = \mathcal{F}(F \bigotimes H) = \mathcal{F}(F) \times \mathcal{F}(H)$$

Spatial deconvolution = division in frequency domain.

$$F = \mathcal{F}^{-1}\left(\mathcal{F}(G) \div \mathcal{F}(H)\right)$$

Usually called inverse filter Inverse filter.

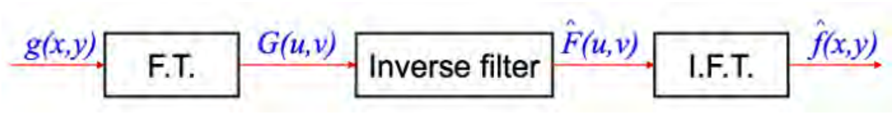


图 12.11: Inverse filter

(但是因为危险的除法)

The blurred kernel is generally a low-pass filter. Inverse filter will amplify high frequency components. Inverse filter will also amplify noise.

$$\begin{aligned} G(u,v) =& H(u,v)F(u,v) + N(u,v) \\ \hat{F}(u,v) =& \frac{G(u,v)}{H(u,v)} = F(u,v) + \frac{N(u,v)}{H(u,v)} \end{aligned}$$

4.2 Wiener Filter

Suppress high frequency when reverse filtering

$$\begin{aligned} \hat{F}(u,v) =& W(u,v)G(u,v) \\ W(u,v) =& \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)} \end{aligned}$$

- If $K = 0$ then $W(u,v) = \frac{1}{H(u,v)}$, i.e. an inverse filter.
- If $K \gg |H(u,v)|$ for large u, v, then high frequencies are attenuated.
- K is set to a constant scalar which is determined empirically.

4.3 Deconvolution by optimization

Blurred image generation process:

$$G = F \bigotimes H + N$$

Assuming that the noise N is Gaussian noise, the likelihood can be measured by the mean square error:

$$MSE = \left\|G - F \bigotimes H\right\|_2^2 = \sum_{ij} \left(G_{ij} - \left[F \bigotimes H\right]_{ij}\right)^2$$

But deconvolution is ill-posed, non-unique solution. Need prior information about the solution.
Prior of natural image:

(1) Natural images are generally smooth in segments

(2) Gradient map is sparse

(3) Adding L1 regularization makes the image gradient sparse.

Objective function = likelihood function + regular term

$$\min_F \left\|G - F \bigotimes H\right\|_2^2 + \left\|\nabla F\right\|_1$$

5. Blind image deconvolution(BID)

The convolution kernel is also unknown. Obviously more difficult. Need more prior knowledge!

5.1 Kernel prior

Blur kernel is non-negative and sparse. Optimized objective function:

$$\begin{aligned} &\min_F \left\|G - F \bigotimes H\right\|_2^2 + \lambda_1 \left\|\nabla F\right\|_1 + \lambda_2 \left\|H\right\|_1 \\ &s.t. H \geq 0 \end{aligned}$$

三、 Colorization

Colorization refers to the process of adding color to a monochrome picture or video with the aid of a computer. There are two main ways to color grayscale images:

(1) Sample-based colorization: use sample image.

(2) Interactive colorization: paint brush interactively.

1. Sample-based colorization

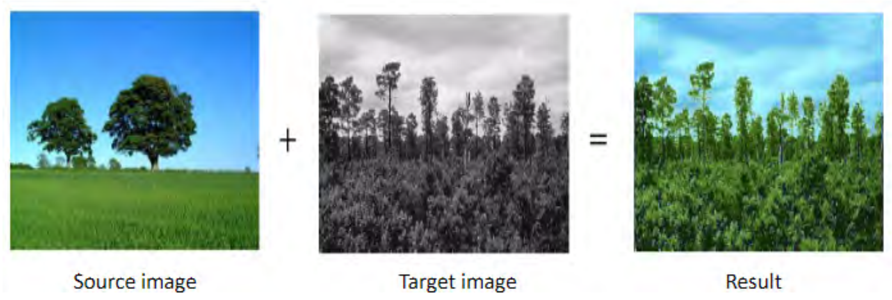


图 12.12: Sample-based colorization

Scan the target image, for each pixel:

(1) Find the best matching point in the sample (e.g. considering the brightness and the standard deviation with neighboring pixels)

(2) Assign the color of the matching point to the pixel

2. Interactive colorization

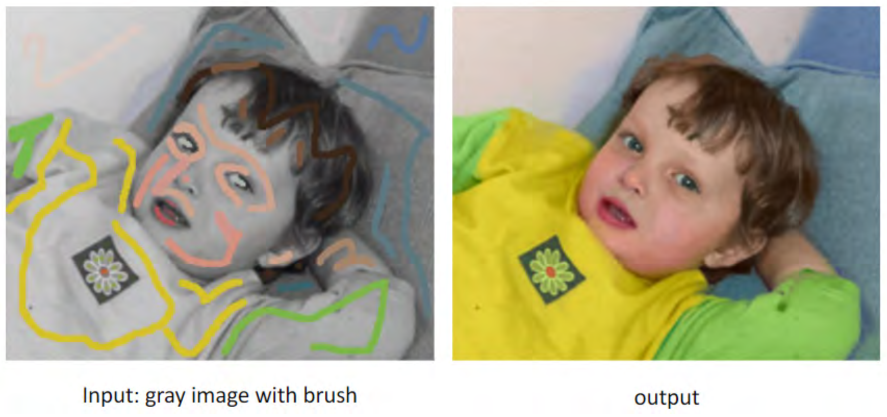


图 12.13: Interactive colorization

For two adjacent pixels, if the brightness is similar, then the color should also remain similar. Based on this assumption, the coloring problem is transformed into an optimization problem:

$$J(U) = \sum_r \left(U(r) - \sum_{s \in N(r)} w_{rs} U(s) \right)^2$$

- $U(r), U(s)$: RGB values of pixel r, s.
- $N(r)$: neighborhood pixels of pixel r.
- w_{rs}: Weight that measures similarity between r and s.

Constraint: User-specified colors of brushed pixels keep unchanged.

3. The modern approach

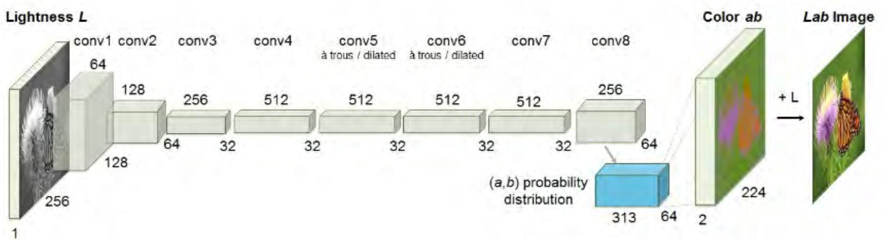


图 12.14: Convolutional Neural Networks

3.1 Loss function for image synthesis

Reconstruction loss:

$$L(\Theta) = \|F(X;\Theta) - Y\|^2$$

Problem with reconstruction loss:

- Cannot handle the case with multiple solutions
- Cannot measure if an image is realistic

3.2 Generative Adversarial Network (GAN)

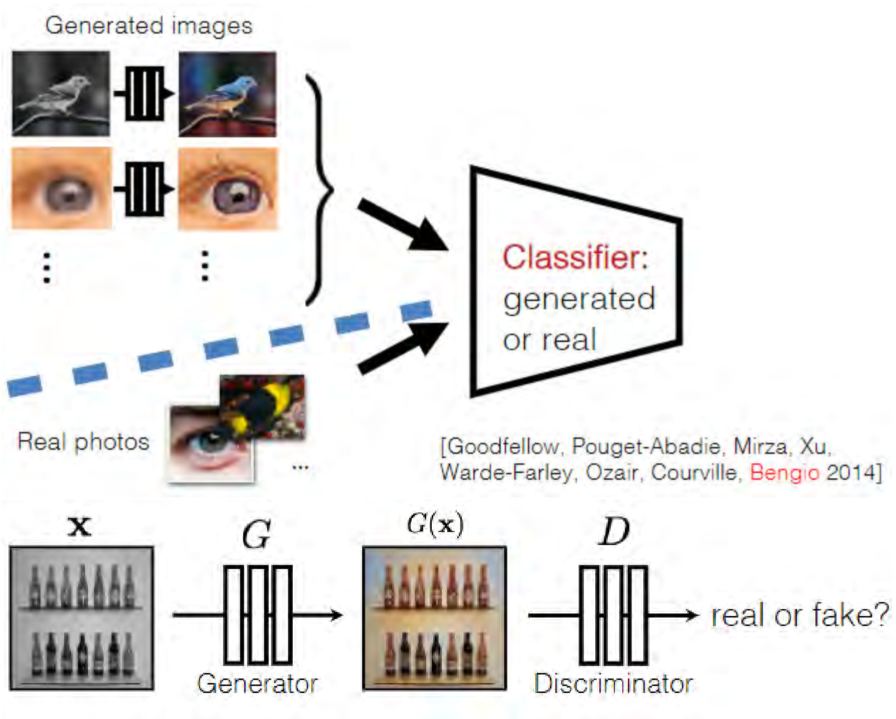

Generated images
Classifier: generated or real
Real photos
[Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio 2014]
x
G
G(x)
D
Generator
Discriminator
real or fake?

图 12.15: GAN

G tries to synthesize fake images that fool **D**. **D** tries to identify the fakes.

Training discriminator:

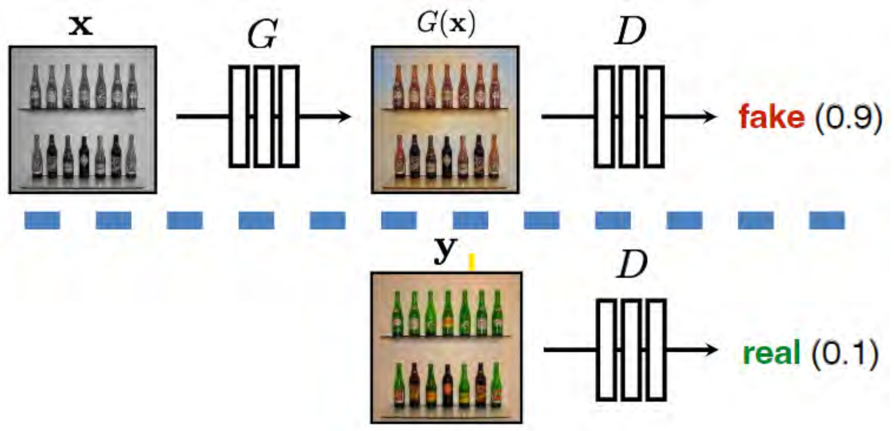


图 12.16: Training discriminator

$$\arg\max_{D} \mathbb{E}_{\mathbf{x},\mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$

Training generator:

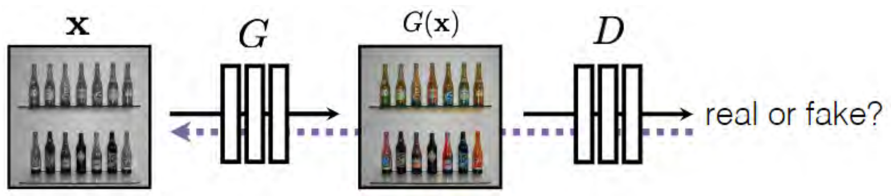


图 12.17: Training generator

G tries to synthesize fake images that fool **D**:

$$\arg\min_{G} \mathbb{E}_{\mathbf{x},\mathbf{y}} [\log D(G(\mathbf{x}))]$$

But for true, we training both of them together:

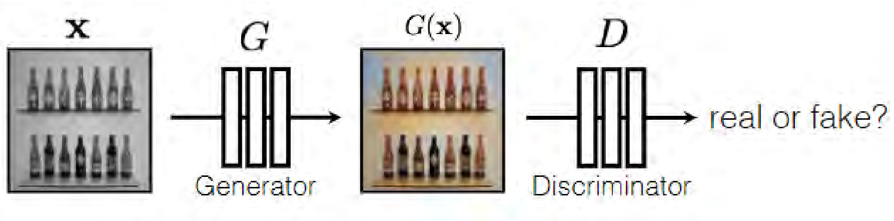


图 12.18: Training

G tries to synthesize fake images that fool the best **D**:

$$\arg\min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))\right]$$

GAN 很难调, 难以收敛. 最近的 vae 与 diffusion model 缓解 GAN 的训练问题. 以及 GAN 的改进 Cycle-GAN, 解决了缺少成对训练数据的问题.

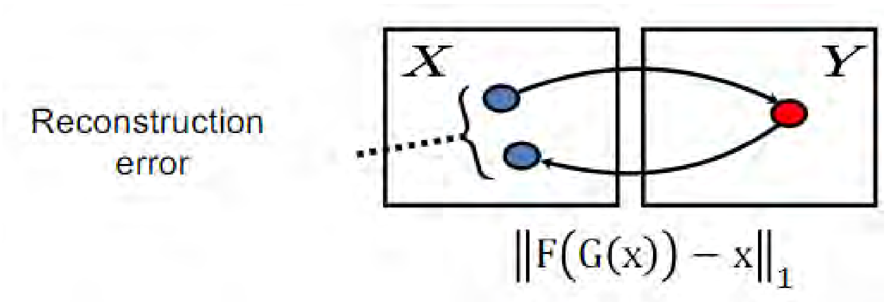


图 12.19: Cycle-GAN

D can be viewed as a loss function to train G, called adversarial loss.

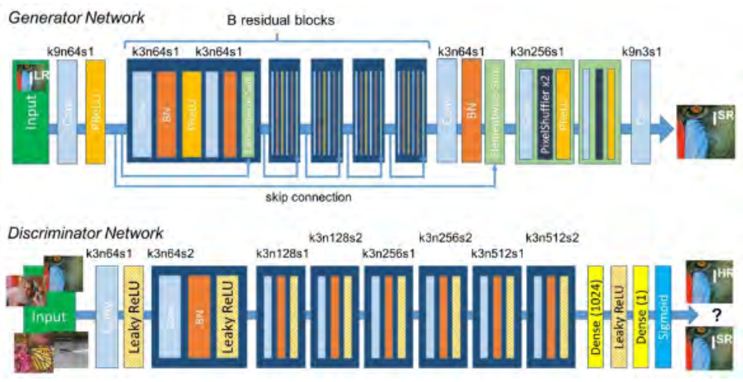
- Learned instead of being hand-designed.
- Can be applied to any image synthesis task.

4. More image sysnthesis tasks

- Image to Image Translation.
- Style transfer.
- Text-to-Photo.
- Image dehazing.
- Customized gaming.

四、 Super Resolution

1. Super Resolution using GAN



$$l^{SR} = \underbrace{l_{\mathrm{X}}^{SR}}_{\substack{\text{reconstruction loss}\\ \text{(likelihood)}}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\substack{\text{adversarial loss}\\ \text{(regularizer)}}}$$

$$\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2 \qquad \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

图 12.20: Super Resolution using GAN

五、 Image-based Rendering (渲染)

1. Rendering

Rendering: from 3D models to images

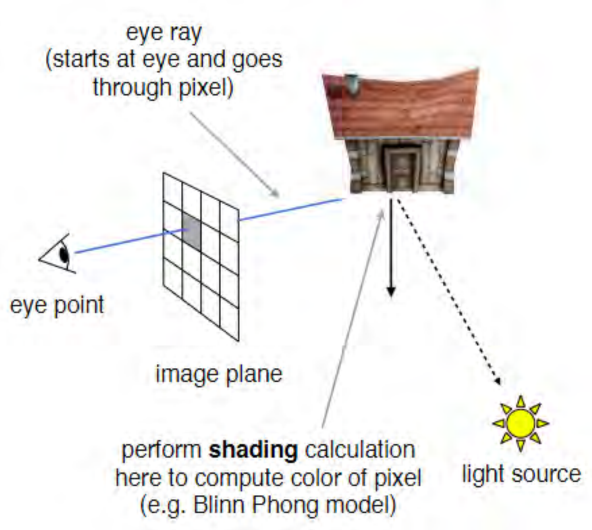

eye ray
(starts at eye and goes through pixel)
eye point
image plane
perform **shading** calculation here to compute color of pixel (e.g. Blinn Phong model)
light source

图 12.21: Rendering

2. Image-based rendering

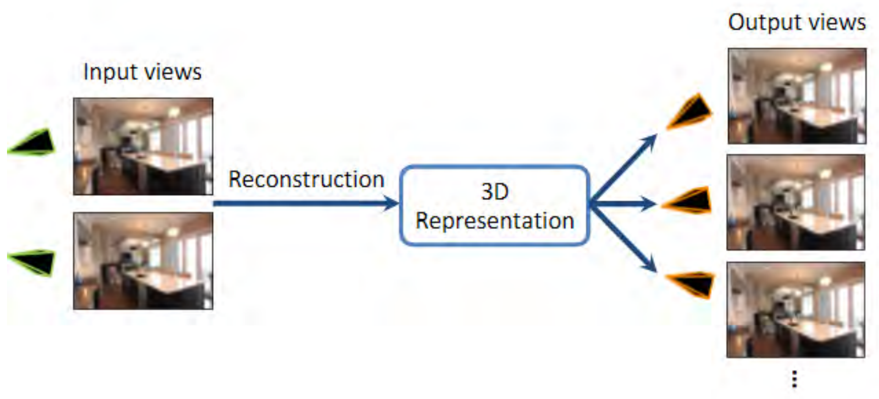


图 12.22: Image-based rendering

3. Applications

(1) 3D photo

(2) VR tour

(3) Bullet time effect

(4) Free-viewpoint video

(5) Immersive telepresence

(6) Embodied AI: training agents in simulated environments

4. Different representations

(1) Textured mesh

(2) Light fields and Lumigraphs

(3) Depth warping

(4) Multi-plane images

(5) Neural Radiance Fields

4.1 Surface-based representations

3D mesh with texture map
Mesh reconstruction pipeline

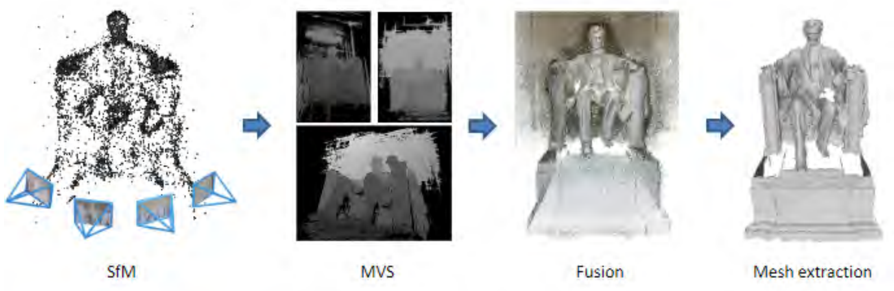


图 12.23: Surface-based representations

Limitations:

- High-quality mesh reconstruction is difficult in many cases.
- Cannot represent very complex scenes.

4.2 Light Fields

The Plenoptic Function (7D) depicts light rays passing through:

- at any location (p_x, p_y, p_z).
- at any viewing angle (θ, ϕ).

- for every wavelength (λ).
- for any time (t).

5D function by ignoring λ and t.

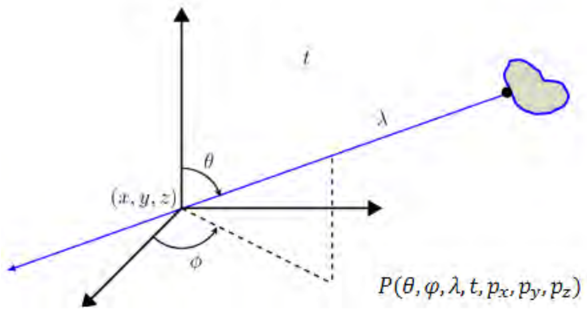


图 12.24: Light Fields

Lumigraph: approximate a light field as a 4D function (用两平面上的点确定一条光线).

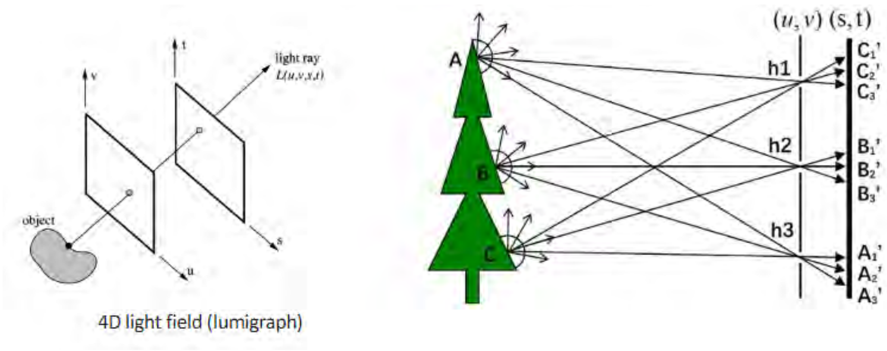


图 12.25: Lumigraph

Capture the light field with a dense camera array.
Novel view synthesis by light field interpolation.

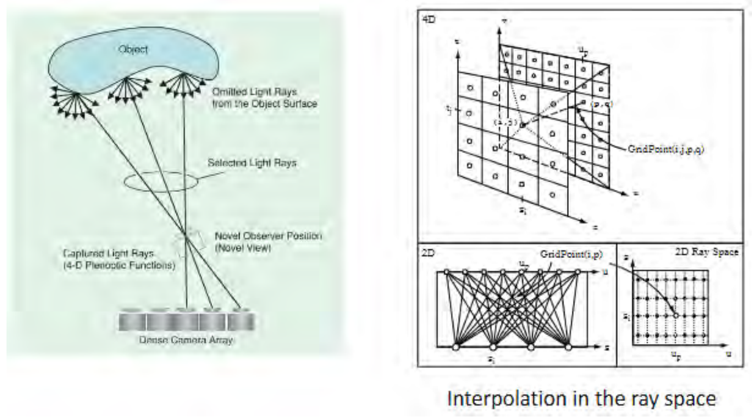

Object
Omitted Light Rays from the Object Surface
Selected Light Rays
Novel Observer Position (Novel View)
Captured Light Rays (4-D Plenoptic Functions)
Dense Camera Array
4D
GridPoint(i,j,p,q)
2D
GridPoint(i,p)
2D Ray Space
Interpolation in the ray space

图 12.26: light field interpolation

Limitations:

- Need a dense array of cameras
- Can only synthesize a limited range of viewpoints

4.3 Multi-Plane Image (MPI)

A set of front-parallel planes at a fixed range of depths Each plane encodes an RGB color image C_d and an alpha/transparency map α_d.

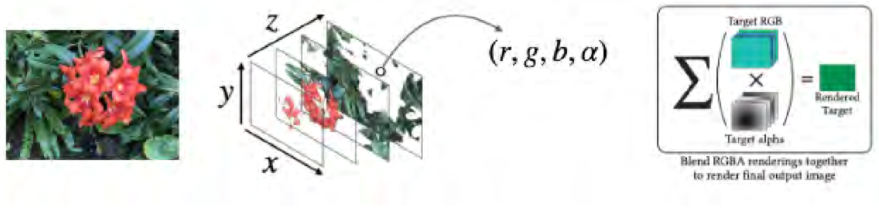


图 12.27: Multi-Plane Image (MPI)

DeepView: View synthesis with learned gradient descent.

Neural Volumes: Learning Dynamic Renderable Volumes from Images. An encoder-decoder network that transforms input images into a 3D volume representation.

Advantages:

- Can represent very complex scenes
- Realistic reflections / specularity / transparency

Limitations:

- Discrete 3D volume requires large storage size for high-resolution rendering.

4.4 Implicit Neural Representations

Implicit Representations is function $f_\theta(p) = \tau$. (使用函数表示, 需要转换才能看得到)

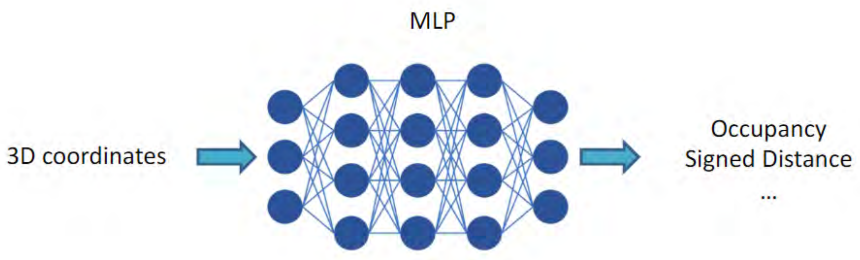


图 12.28: Implicit Neural Representations

使用网络储存表达.

Neural Radiance Fields (NeRF)

Representing scenes as continuous density and color fields.

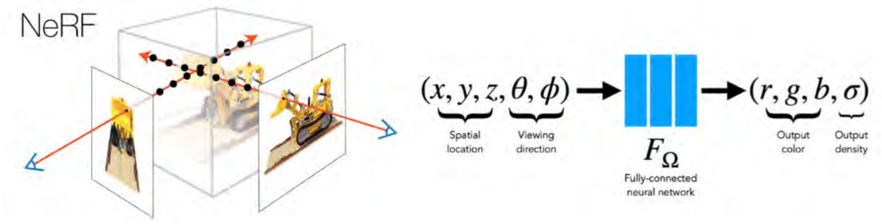

NeRF
$(x, y, z, \theta, \phi) \to F_\Omega \to (r, g, b, \sigma)$
Spatial location
Viewing direction
Fully-connected neural network
Output color
Output density

图 12.29: NeRF

不同方向上看去的点不一样, 输入的是一个视角射线, 输出是一个点的 rgb 与透明度. 而且它是连续的.
Volume rendering, which is differentiable.

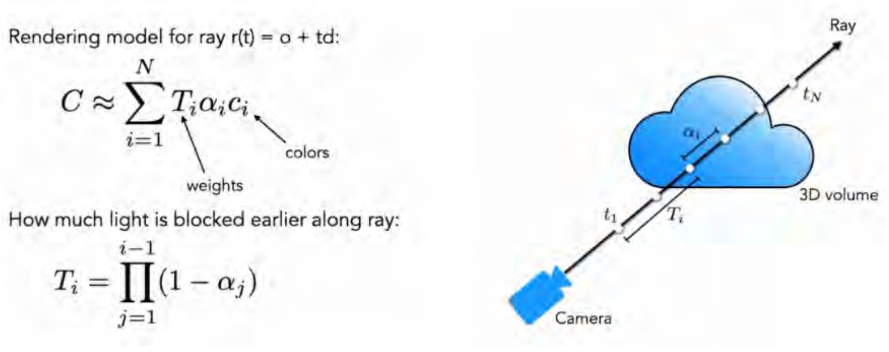

Rendering model for ray r(t) = o + td:
$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$
weights
colors
How much light is blocked earlier along ray:
$T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$
Ray
t_N
α_i
3D volume
t_1
T_i
Camera

图 12.30: Volume rendering

Learning NeRF from images

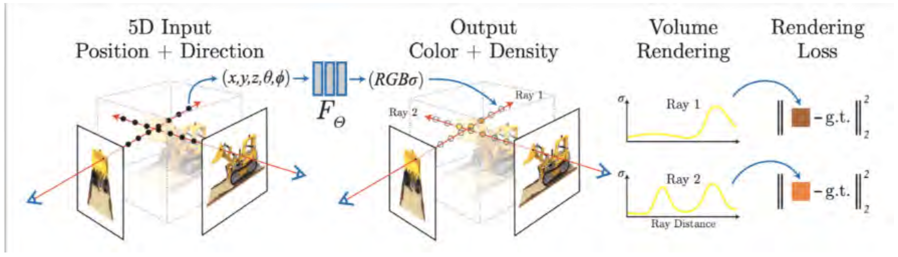


图 12.31: Learning NeRF from images

可微, 所以梯度下降是可用的.
Advantage:

- The cost to represent a 3D scene is low: 10MB.
- It is a continuous and higher-resolution representation.
- It can handle reflection, specularity and non-Lambertian scene well.

Limitation:

- Optimizing a MLP network needs about 1 day.
- Render one novel view needs 30 seconds.
- Only for statistic scenes and constant illumination.

Improvement:

- PlenOctree for NeRF Rendering, Basic idea:
  - Offline inference: space-for-time.
  - Reduce memory cost by Octree.
- NeRF in the wild, Inprovement of NeRF on:
  - Varying illumination of input images
  - Removing dynamic objects in the scene. e.g. Peoples
- NeuralBody: Reconstruct dynamic human body from sparse input views.
- Deformable NeRF: Neural radiance field to model dynamic scenes.