“感知”

从 几何模型：基于几何原理和相对位置关系的模型

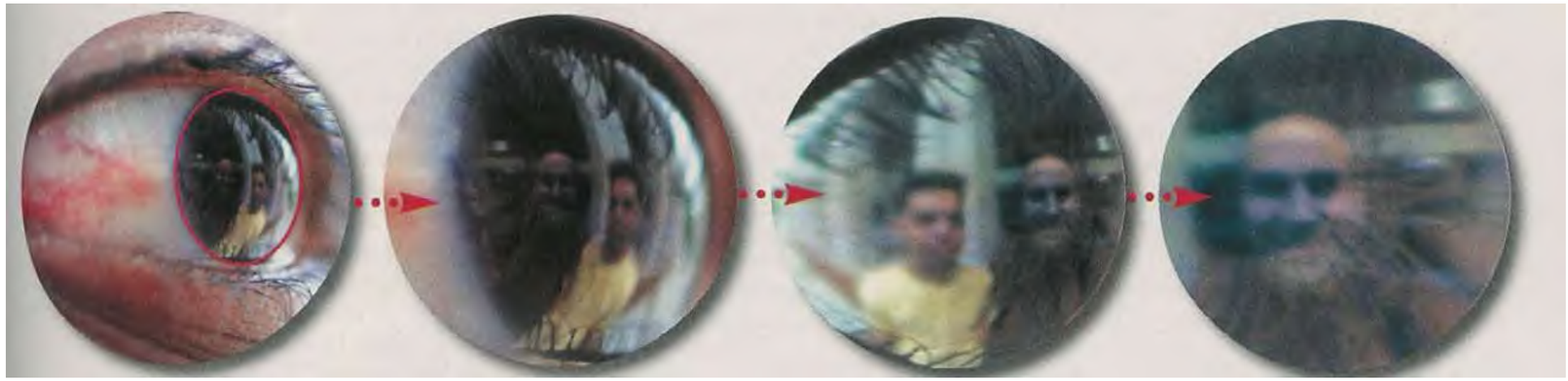
到 统计模型：基于数据分布和统计推断的模型

图像图形学的最终目标

- Forensics：取证

Source: Nayar and Nishino, “Eyes for Relighting”

Source: Nayar and Nishino, “Eyes for Relighting”



Source: Nayar and Nishino, “Eyes for Relighting”

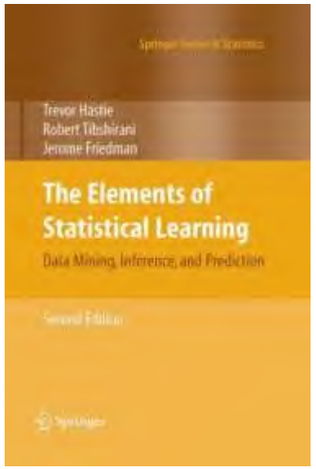
图像图形学的最终目标

https://www.bilibili.com/bangumi/play/ep28950?t=717
《攻壳机动队》

感知与理解

https://www.bilibili.com/video/BV1EV411y7g4
《神探夏洛克》

Pointers



Useful book (Free too!):
The Elements of Statistical Learning
Hastie, Tibshirani, Friedman
https://web.stanford.edu/~hastie/ElemStatLearn/

Useful set of data:
UCI ML Repository
https://archive.ics.uci.edu/ml/datasets.html

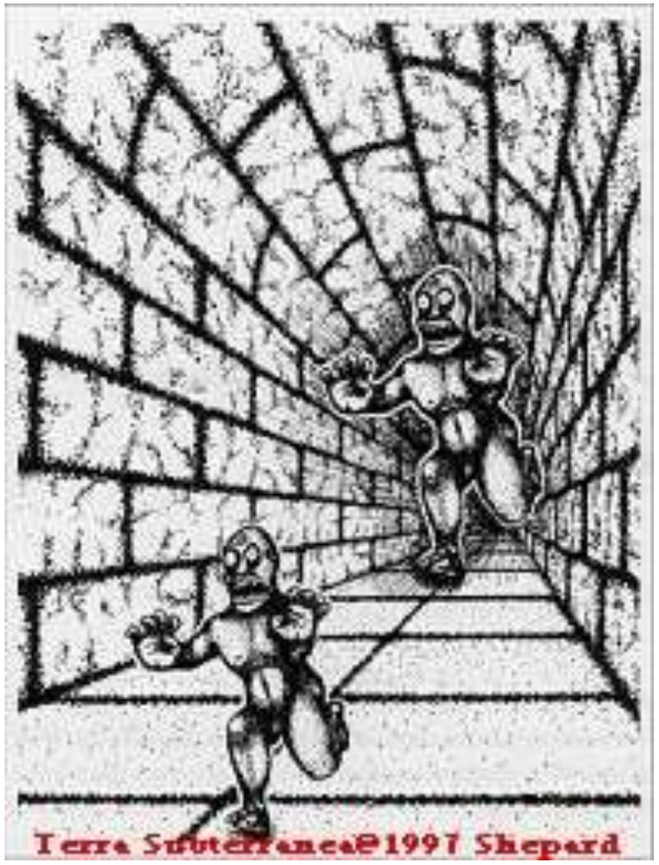
“感知”

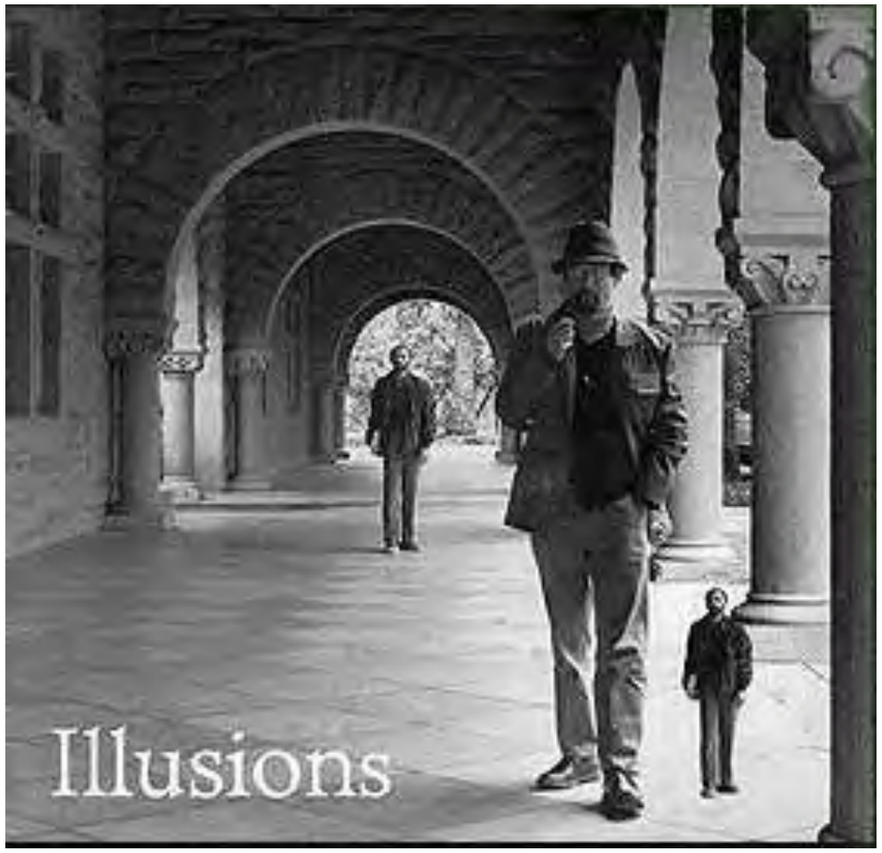
为什么 需要 统计模型？

让我们从自己的感知系统开始!

视觉系统存在错觉——透视几何系统

视觉系统存在错觉——透视线索





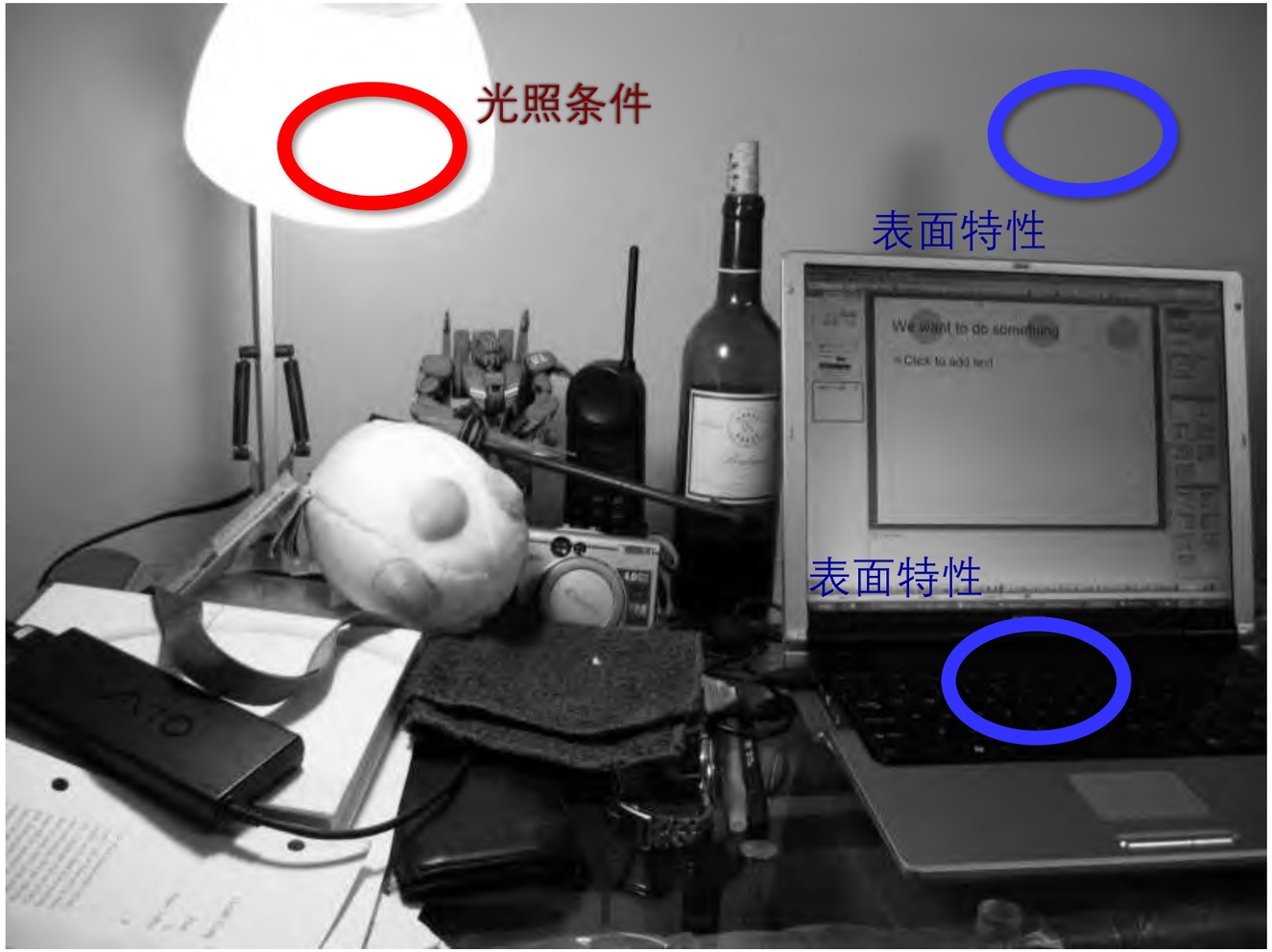
蓝黑还是白金?

消失的圆点

With left eye shut, look at the cross on the left. At the right distance, the circle on the right should disappear (Glassner, 1.8).

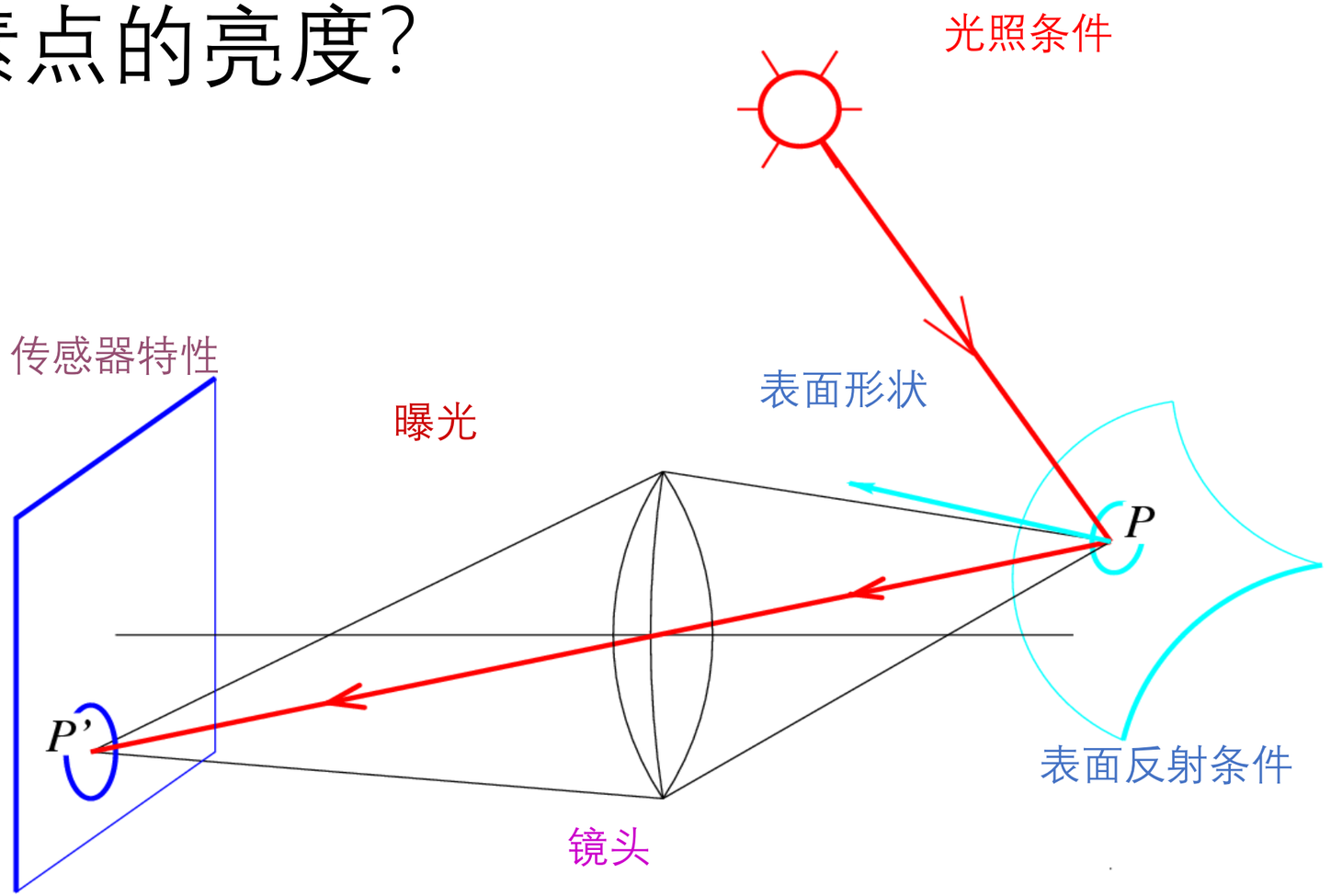
Radiometry：辐射度量

- 什么决定了像素点的亮度?
- 相机参数
- 光照条件：光源的强度和方向会直接影响像素点的亮度。
- 物体表面特性：物体表面的反射率和漫反射性质会决定光线被多少吸收和多少反射到观察者或相机。



辐射度量

- 什么决定了像素点的亮度?



Slide by L. Fei-Fei

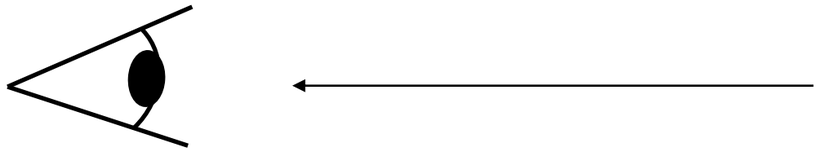
什么是 光?

光的感知

- 怎样把辐射转换为“色彩”?
- 我们能看到哪些颜色?

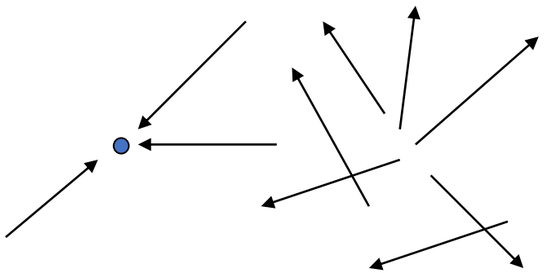
光是一种电磁辐射，它属于电磁波谱的一部分

- R(λ) 代表其能量 (单位为watts)
  - λ是波长



光场

- 光场是指在空间中传播的电磁辐射波动。它描述了光波的传播方向、波长和振幅。

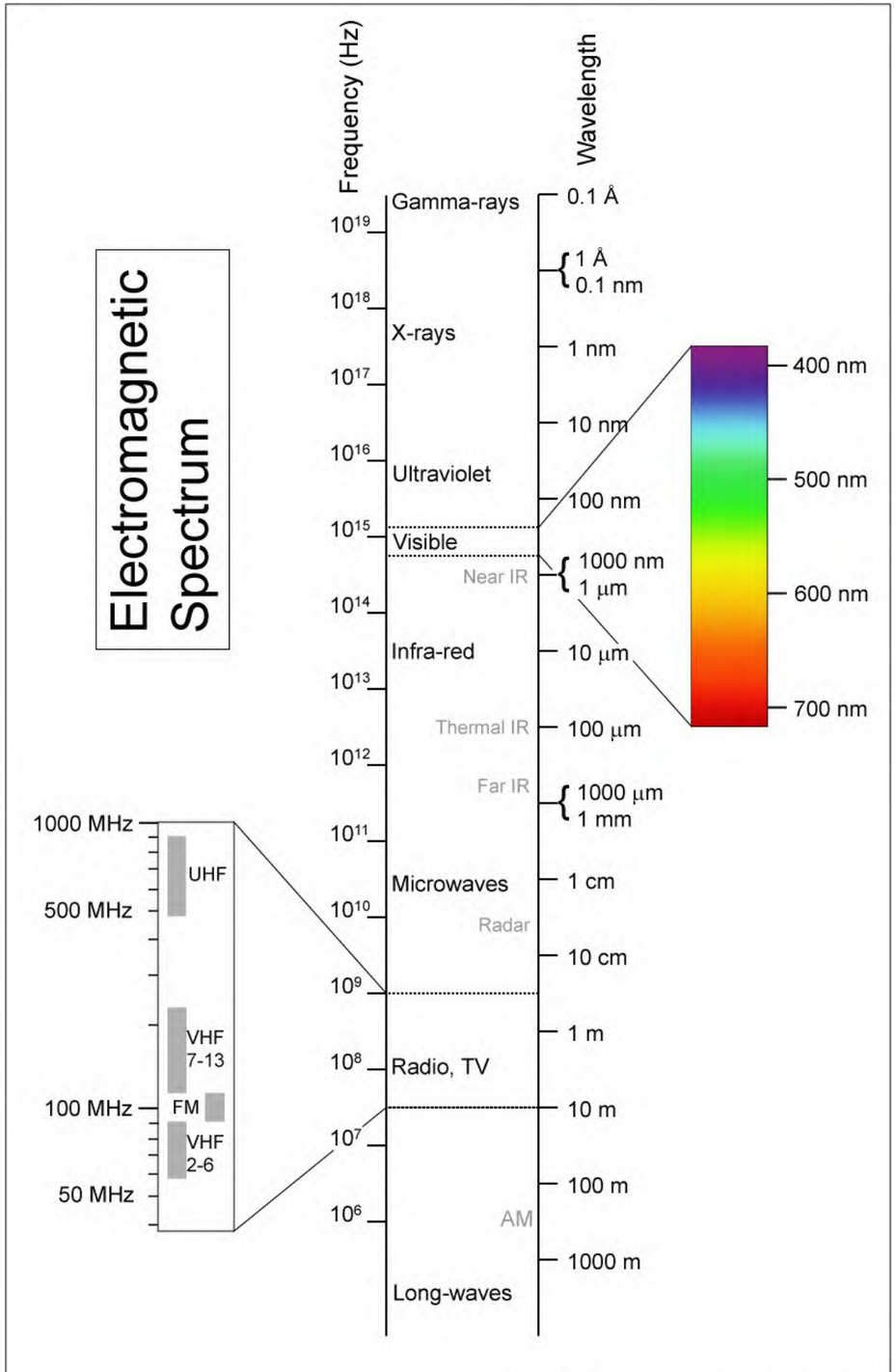


plenoptic function 定义描述光的函数:

$$R(X, Y, Z, \theta, \phi, \lambda, t)$$

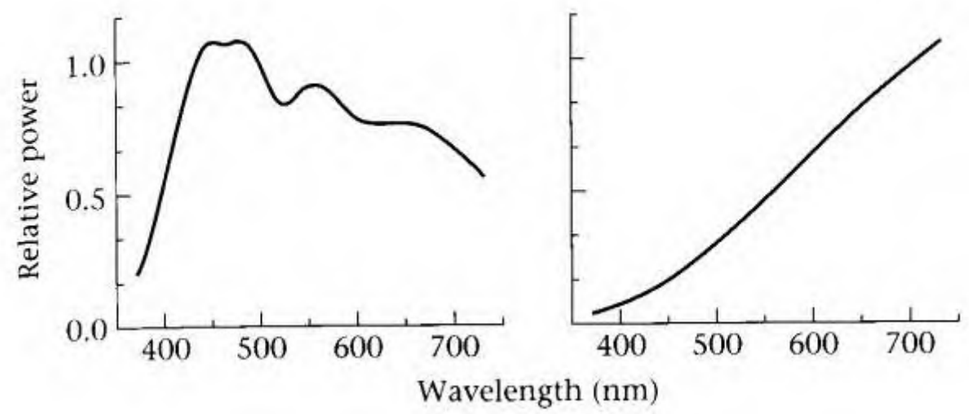
可见光

我们只能看到一部分波段的光辐射



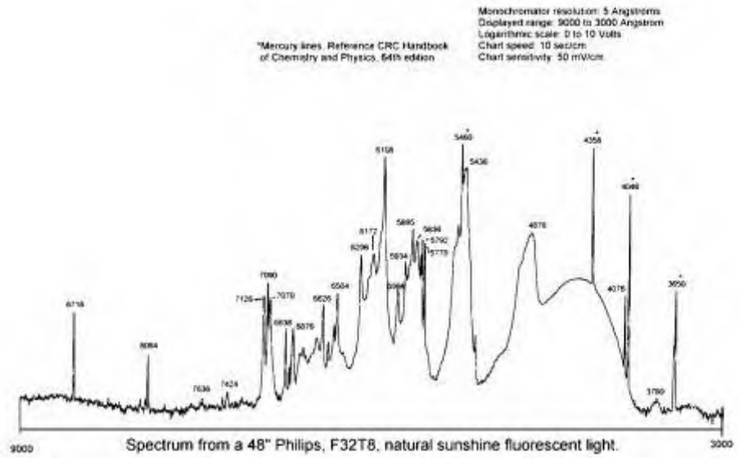
光谱

- 我们看到的外表是可见光对应的光谱
  - 光谱对应的是每个波长的强度



日光

钨丝灯

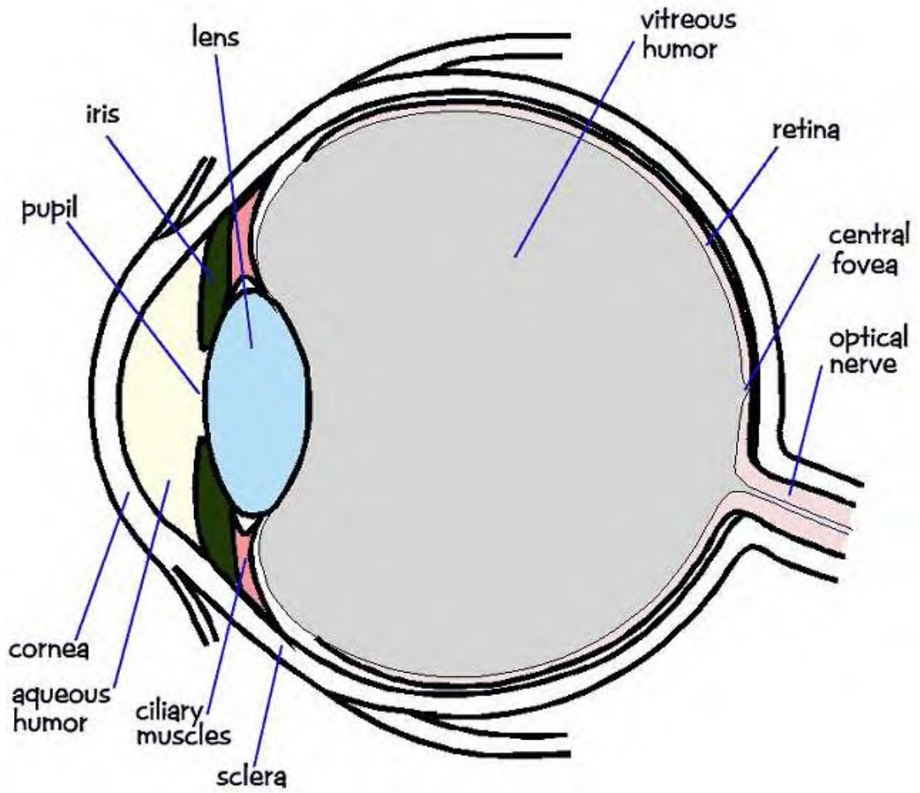


荧光灯

颜色就是从这种光谱通过复杂的形式转换过来的

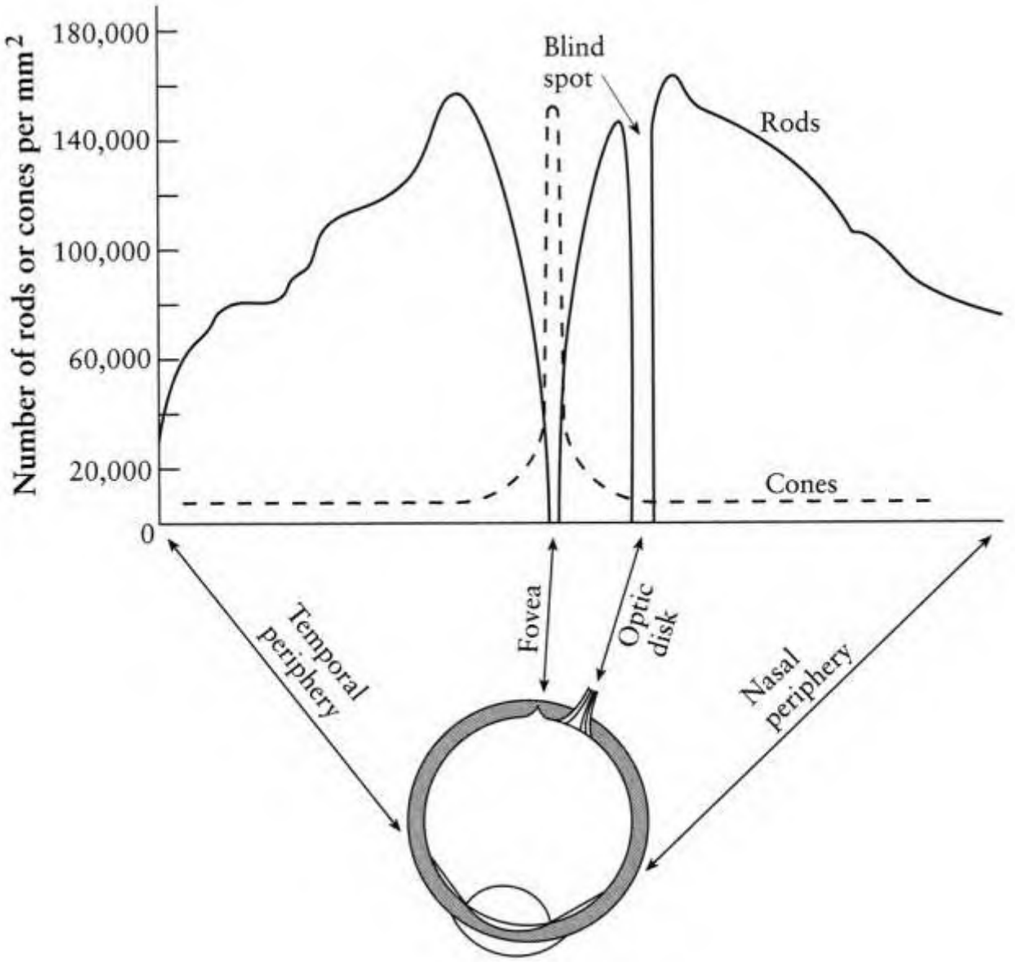
人类的视觉系统

- 颜色感知
  - 光打到视网膜上，由感光细胞捕捉到
    - Rods 视杆 和 cones 视锥
  - 这些感光细胞把光谱转换为颜色



视杆 与 视锥

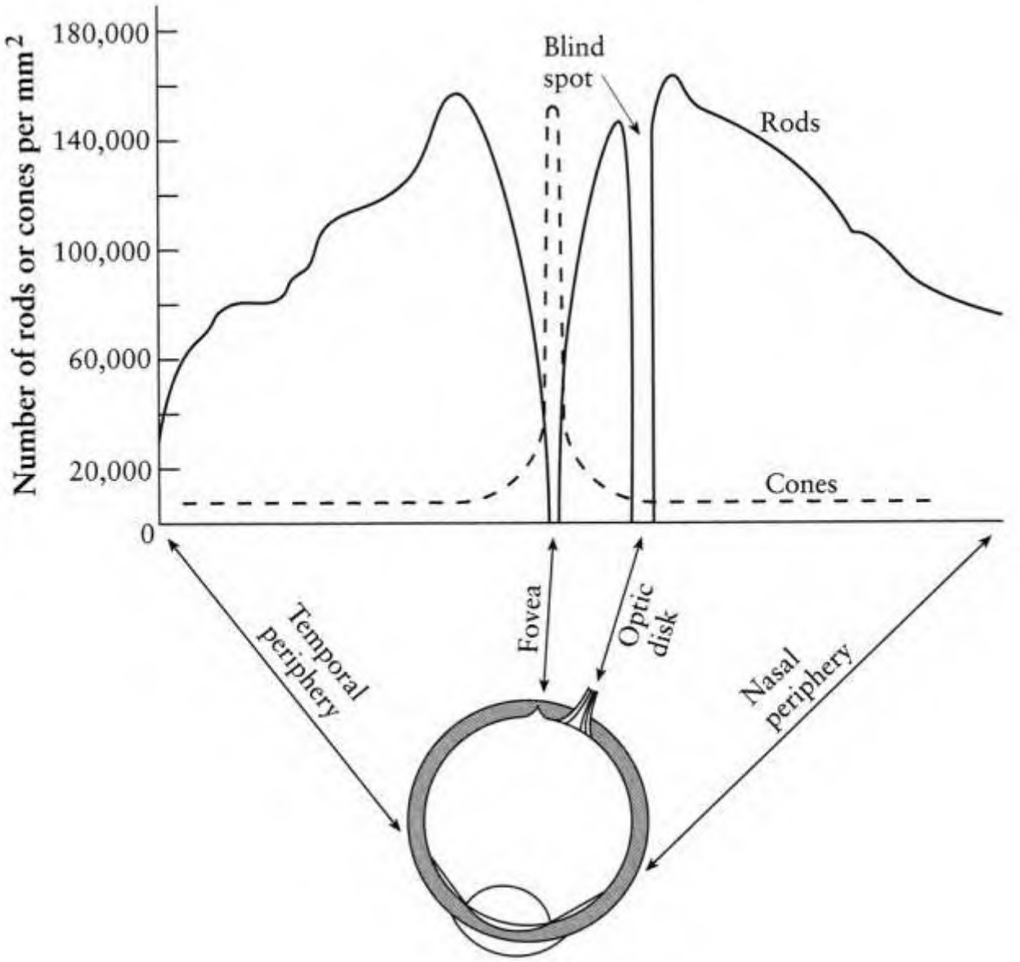
- 视杆与视锥是不均匀地分布在视网膜上的
  - 视杆对光的强度更敏感，视锥对颜色更敏感。明亮光照下，视锥细胞主导，而在暗光下，视杆细胞起主导作用
  - Fovea **中间凹**: 视网膜上的一个小区域(1 or 2°)，包含大量视锥细胞（**没有视杆**），特别适用于高分辨率的颜色感知。它在颜色识别和高分辨率视觉中发挥关键作用。
  - 不在中央凹区域的部分，对于感知低光强度或运动检测非常有用，但视力锐度较低。



盲点

- 神经盘是视网膜上的一个区域，位于视网膜背部，是视觉信息传输的起点。它是视神经纤维束聚集的地方，这些纤维将视觉信号传送到大脑的视觉中枢。

- **视神经盘上没有感光细胞**，因此在视觉场景中，与视神经盘对应的区域被视为视野中的**盲点**。这是我们视觉系统中的一个盲区，无法感知光或图像。

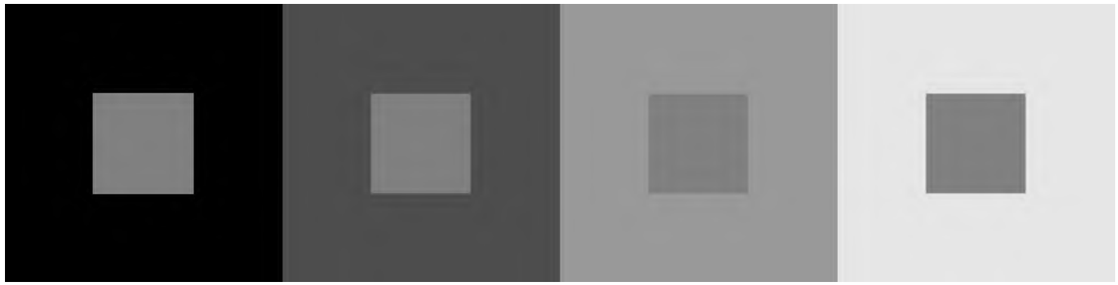


Recall：消失的圆点

With left eye shut, look at the cross on the left. At the right distance, the circle on the right should disappear (Glassner, 1.8).

亮度、对比度

- 亮度与周围区域有光
  - 对比度: 图像或场景中不同区域之间的明暗差异。通常通过比较相邻像素的亮度差异。高对比度的区域中的像素之间有明显的亮暗差异。



  - 光照恒定: 在视觉感知中，恒定性是指在不同光照条件下，物体的颜色和亮度看起来保持相对恒定。

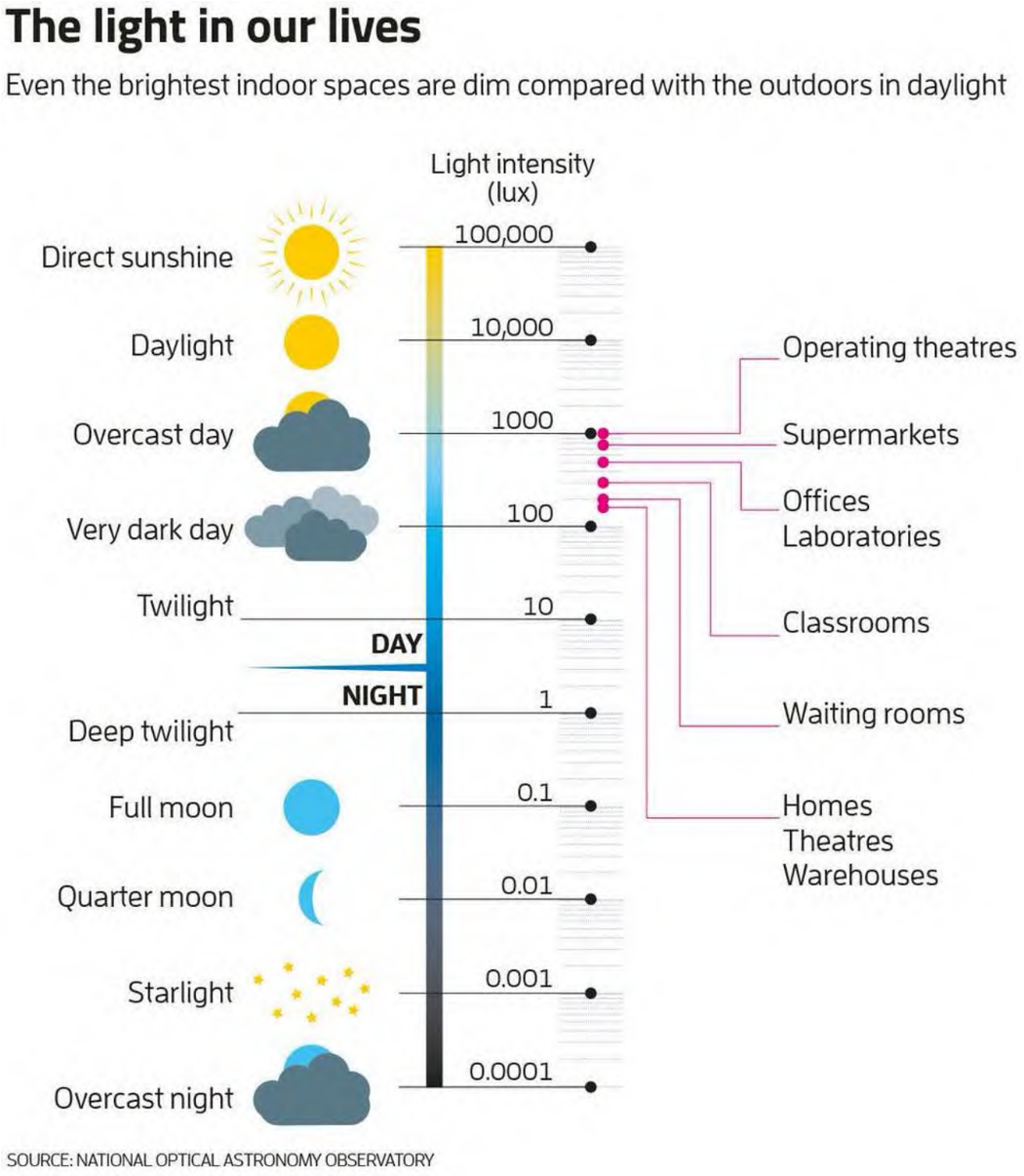
人类的光感系统是动态的

- 我们的视觉系统有有一个很**大**的**动态**的区间
  - 我们可以同时感受亮的和暗的物体
  - 我们能感受**指数级**的不同光强
  - 我们的视觉系统可以**自适应调节**

动态视觉系统

一张白纸，在明亮的室外和在没有月亮的晚上，亮度相差1,000,000,000倍

但是在一个场景内，我们可以轻松感受10倍左右的亮度差别



https://threader.app/thread/1134003178515701762

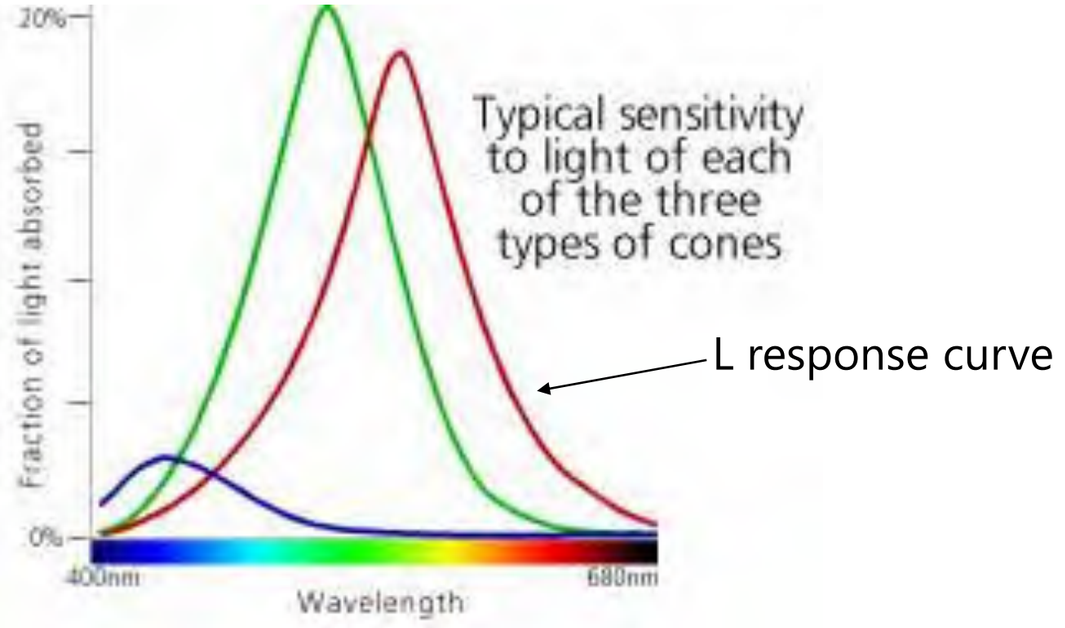
动态视觉系统

如果我们一直对整个范围都非常敏感，那么我们将无法在场景中区分亮度级别。

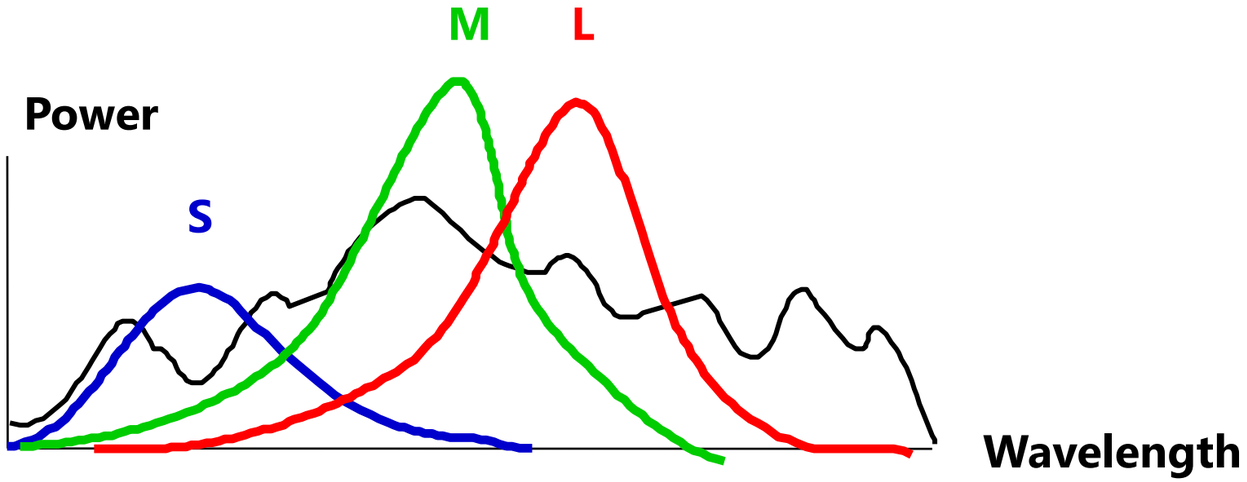
视觉系统通过限制其响应的“动态范围”以匹配当前的整体或“环境”光水平来解决这个问题。

颜色感知

- 三种色锥感受颜色
  - 每种色锥对光谱中的区域更敏感
    - 这些区域相交
    - Short (S) 对应 **蓝色**
    - Medium (M) 对应 **绿色**
    - Long (L) 对应 **红色**
  - 总体而言，我们对红色和绿色更敏感
    - 但每个人的敏感度不同，也会随年龄变化
  - 色盲的原因是至少一种色锥出现了退化

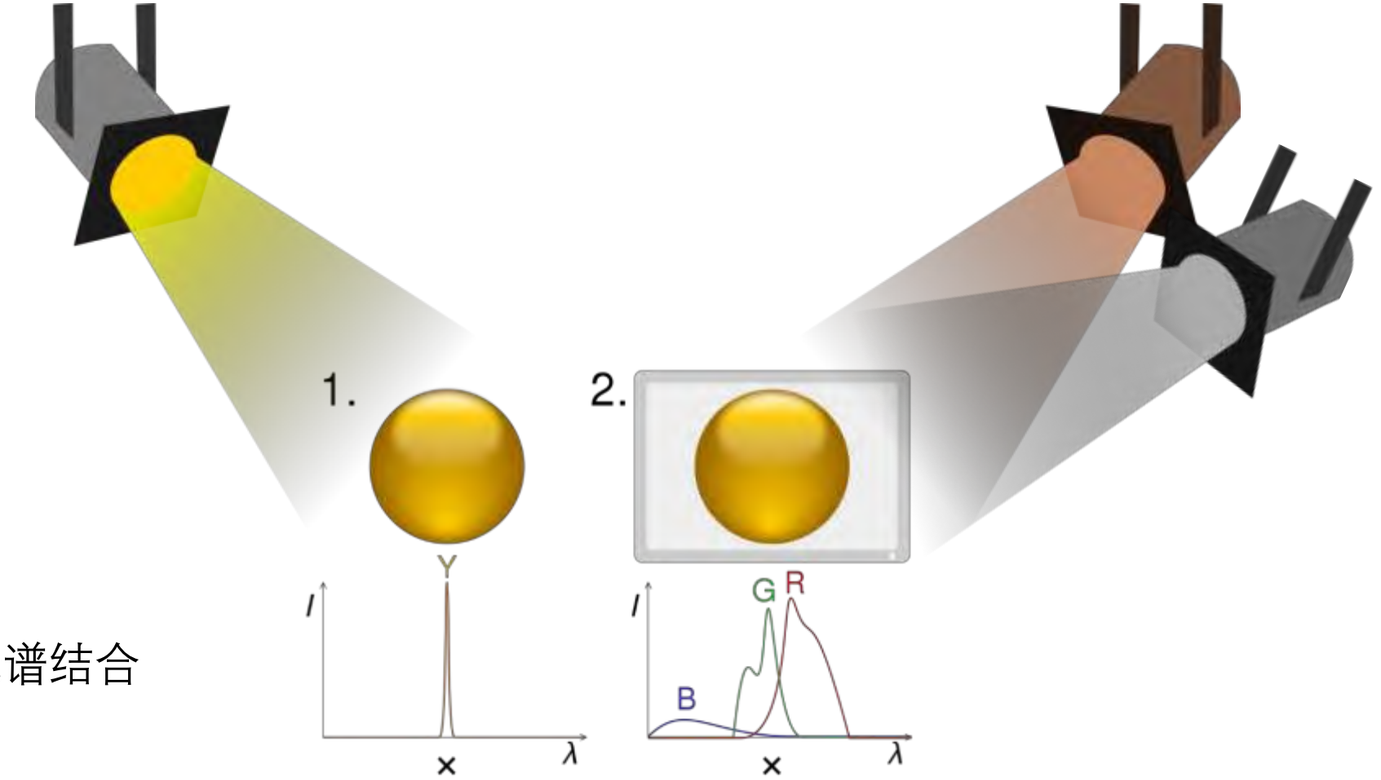


颜色感知



- 视杆与视锥根据光谱感受颜色
  - 在波长上做强度的积分
    - 每个视锥得到一个数字
  - Q: 我们可以直接拿视锥的感受强度来表示光谱吗?
  - A: 不行，会丢失信息
    - 不同的光谱在某些情况下可能完全无法被分辨
      - 这种光谱被称作 metamers **条件等色**

条件等色



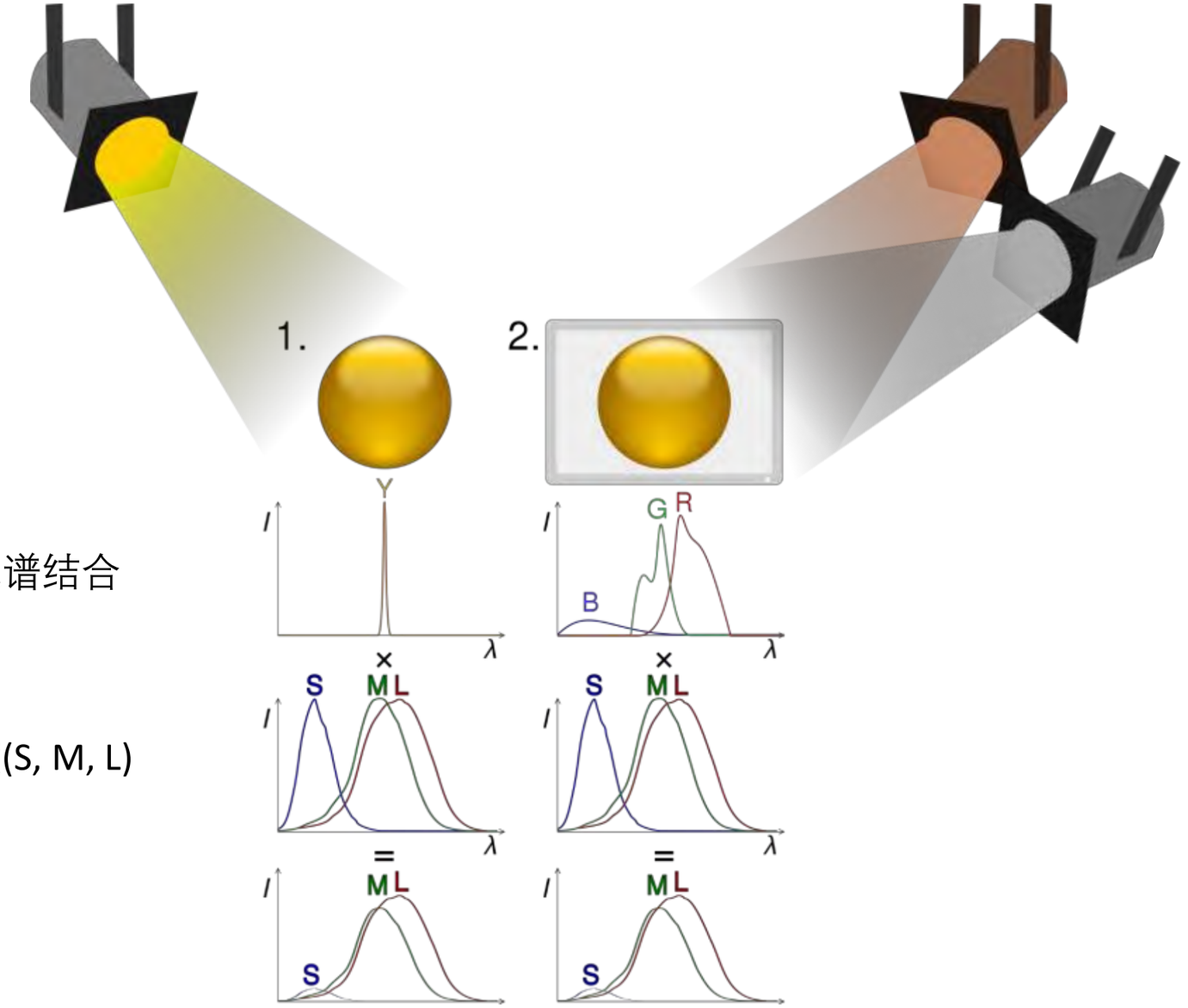
1. 与光源的光谱结合

条件等色

1. 与光源的光谱结合

2. 光锥敏感度(S, M, L)

3. 1与2结合



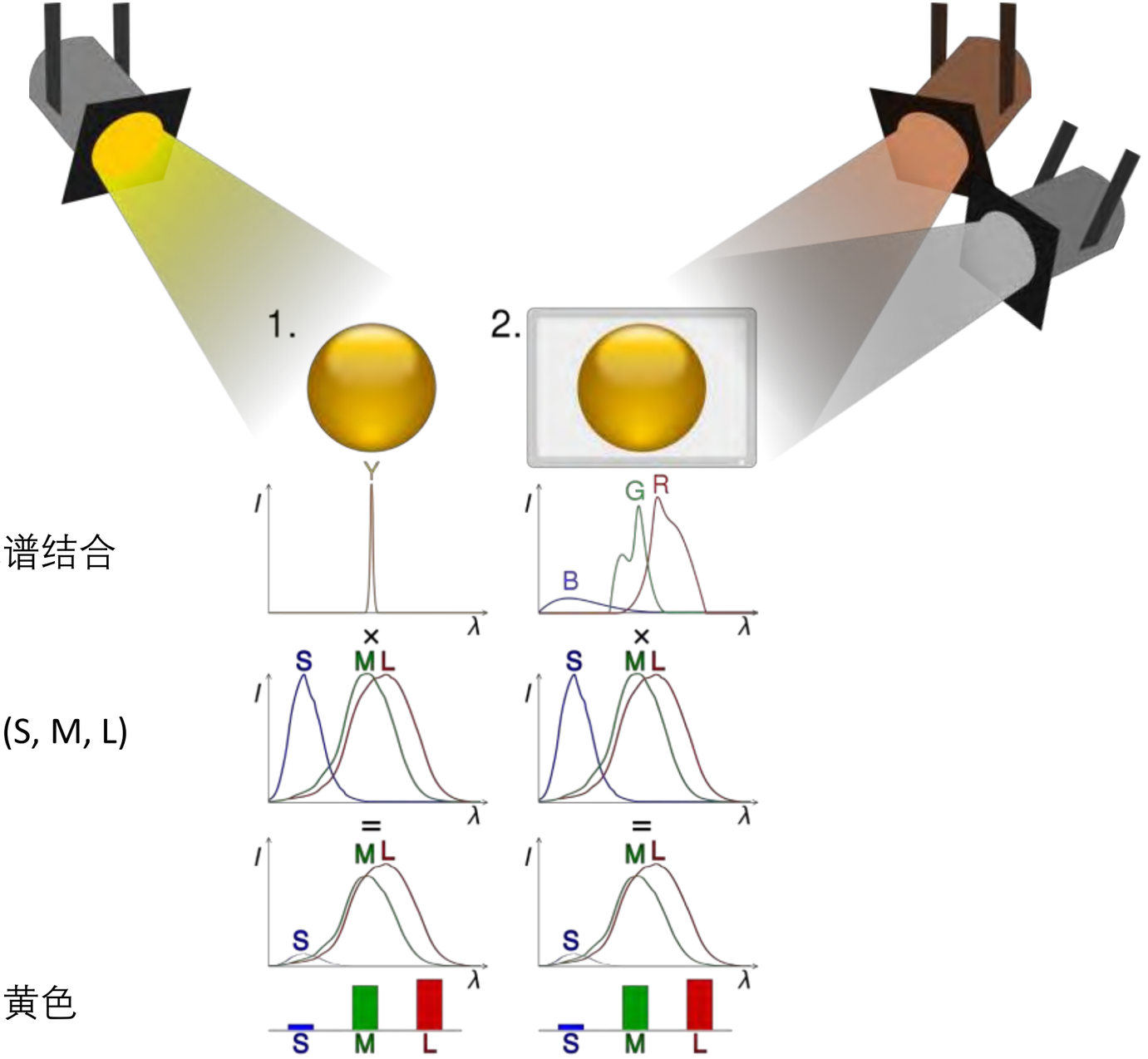
条件等色

1. 与光源的光谱结合

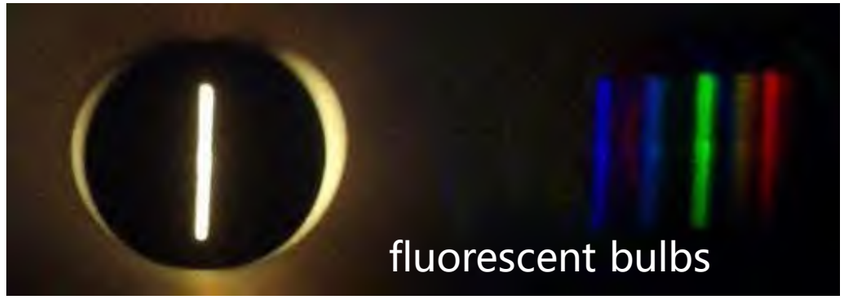
2. 光锥敏感度(S, M, L)

3. 1与2结合

4. 观测颜色：黄色

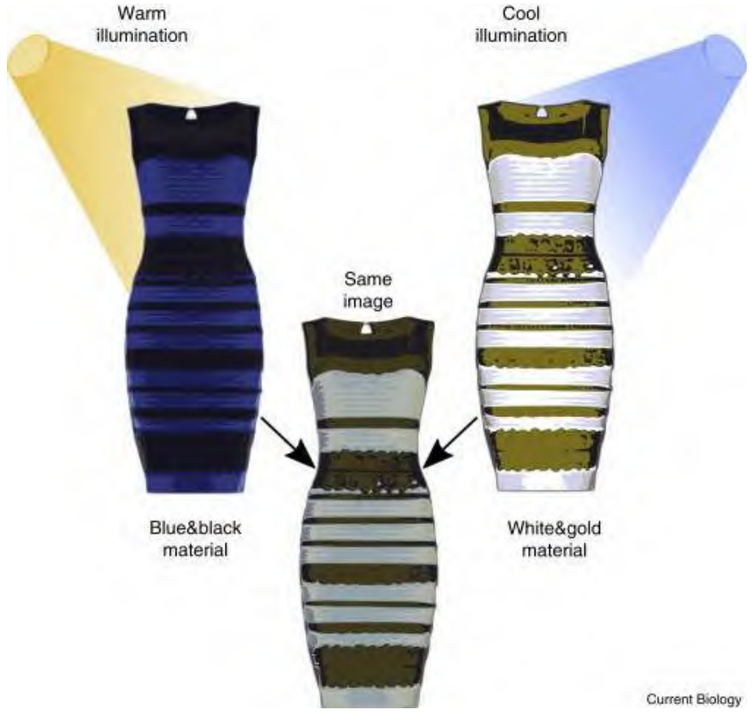


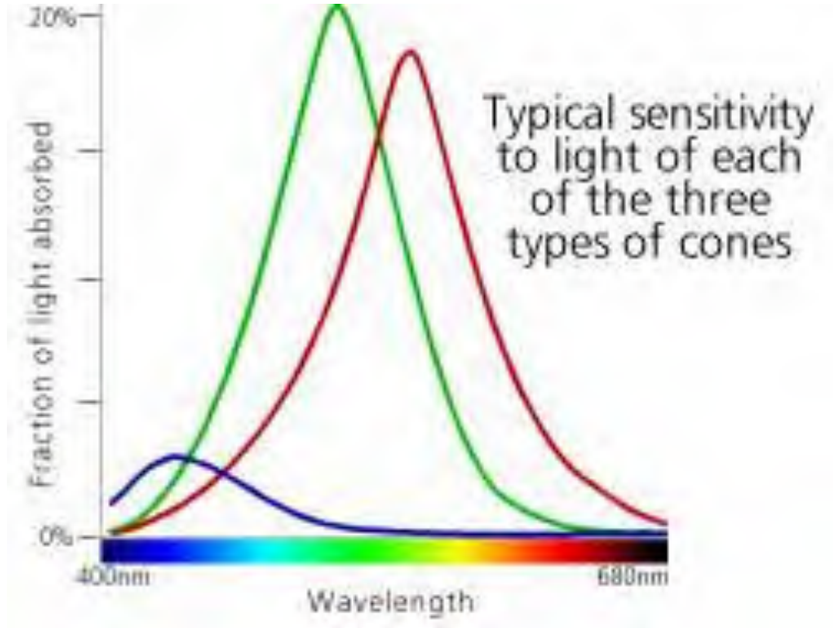
用分光镜（Spectroscope）区分不同的灯泡



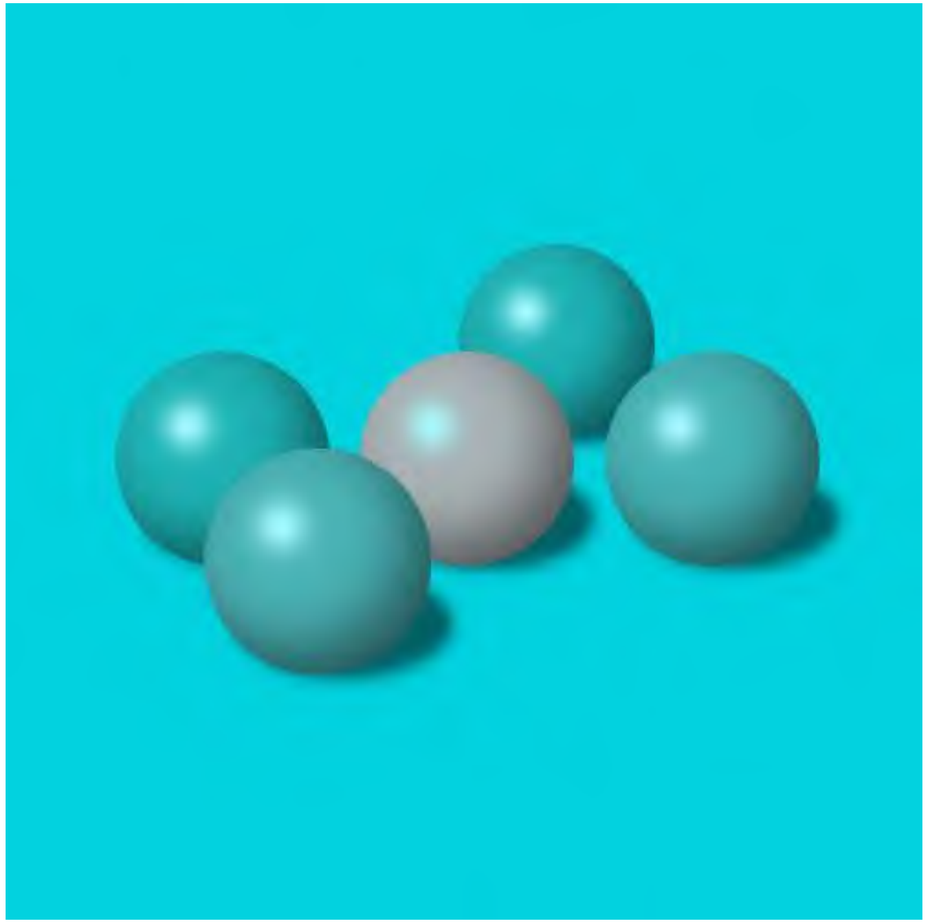
http://www.chemistryland.com/CHM107Lab/Exp7/Spectroscope/Spectroscope.html

蓝黑还是白金?

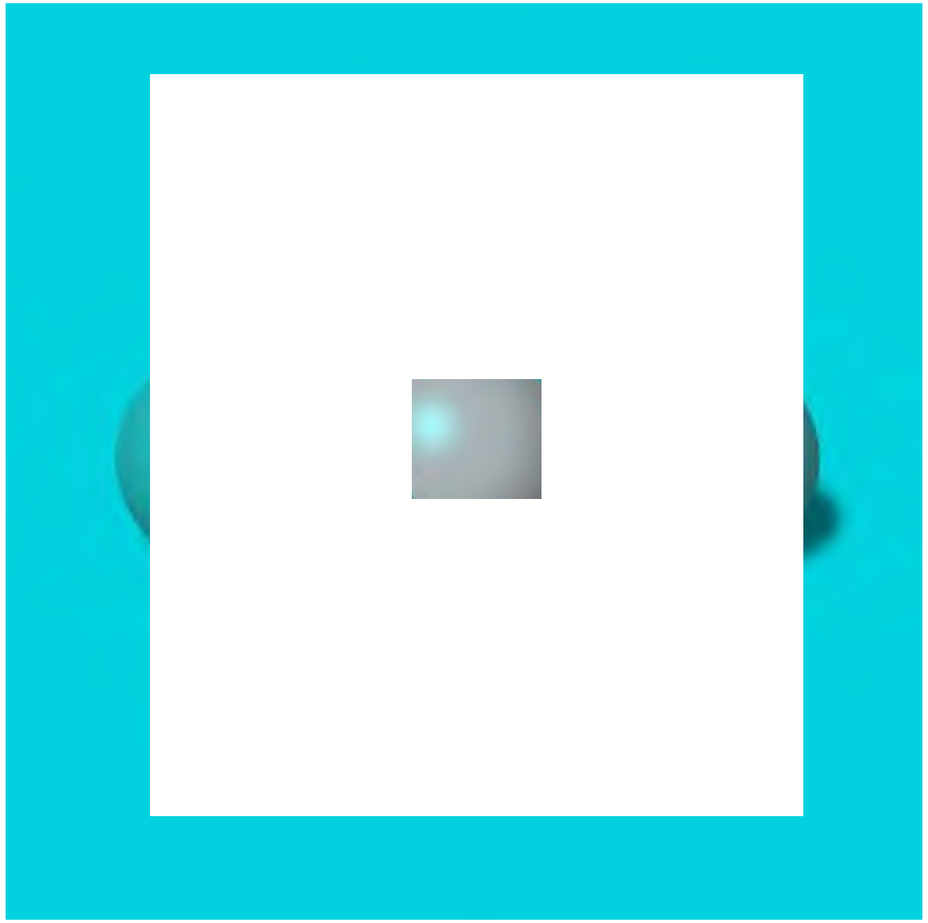

Warm
illumination
Cool
illumination
Same
image
Blue&black
material
White&gold
material
Current Biology


20%
0%
Fraction of light absorbed
400nm
680nm
Wavelength
Typical sensitivity
to light of each
of the three
types of cones

中间的球是什么颜色？

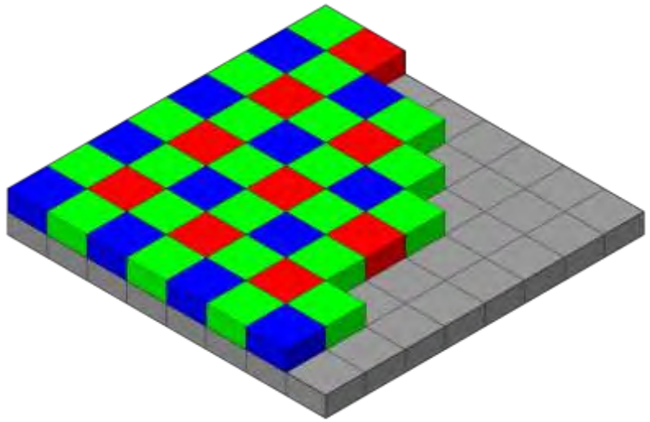


中间的球是什么颜色？



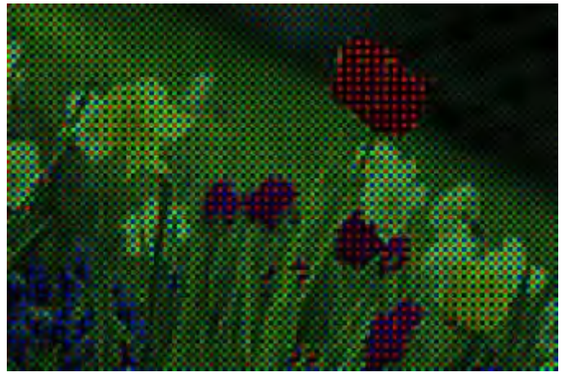
相机模型的色彩

- *Bayer filter 贝尔滤镜* 分隔入射的光线，将光分为红、绿和蓝三个通道。每个像素只能透过其中一种颜色的滤镜，因此，传感器上的每个像素只能感知一种颜色的信息。



Bayer filter pattern in front of sensor

- "*demosaicked*" color image:估计每个像素的完整彩色信息，以重建彩色图像



What the camera sees
("raw" image)

Demosaicked image

https://en.wikipedia.org/wiki/Bayer_filter

早期的图像上色

- 在发明彩色胶卷以前，Sergey Prokudin-Gorsky 根据不同光照下的黑白图像曝光来计算彩色图像。

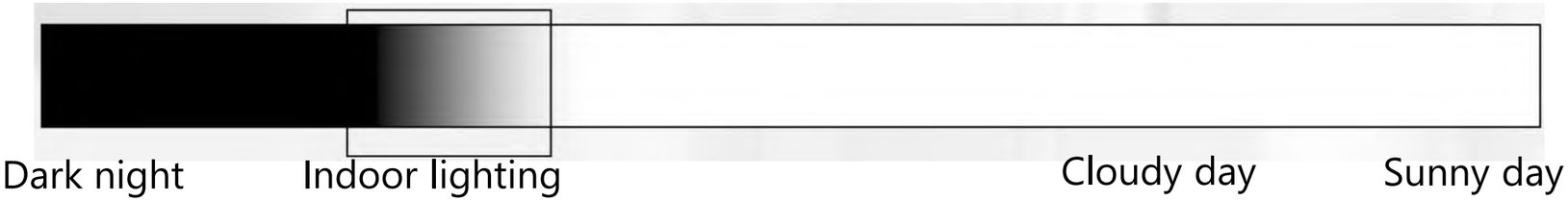
Blue, Green, Red exposures

Combined color image (1911)

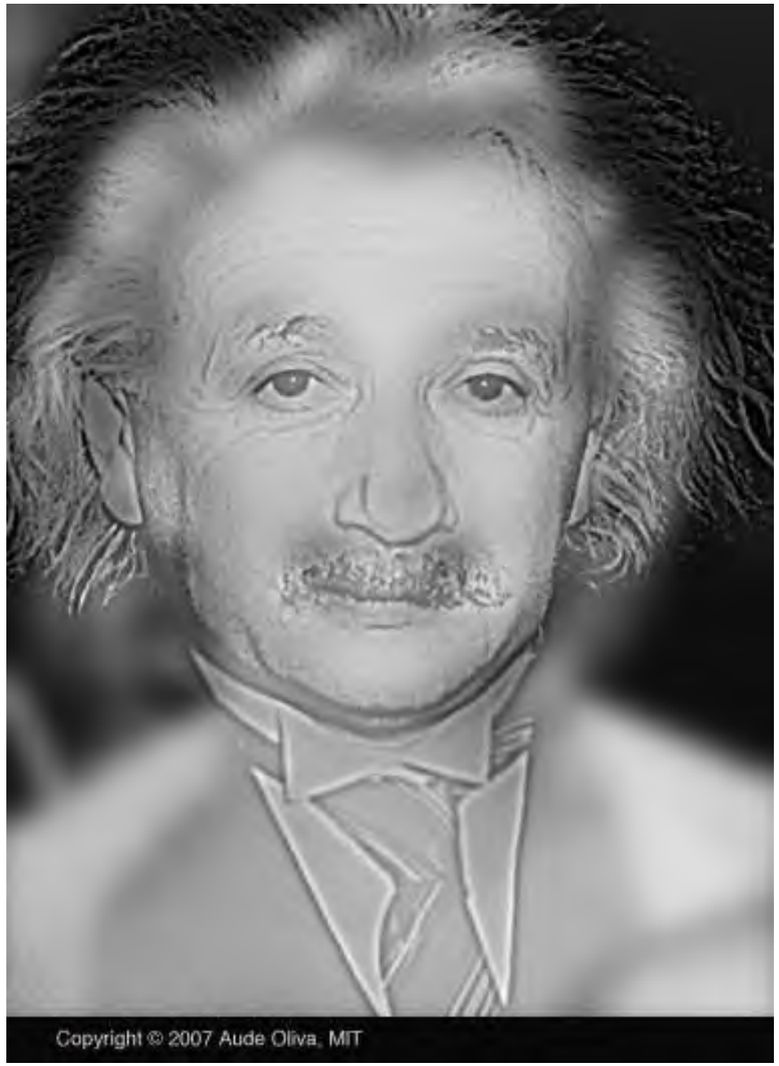
Recall: 动态视觉系统 “尺度”

如果我们一直对整个范围都非常敏感，那么我们将无法在场景中区分亮度级别。

视觉系统通过限制其响应的“动态范围”以匹配当前的整体或“环境”光水平来解决这个问题。



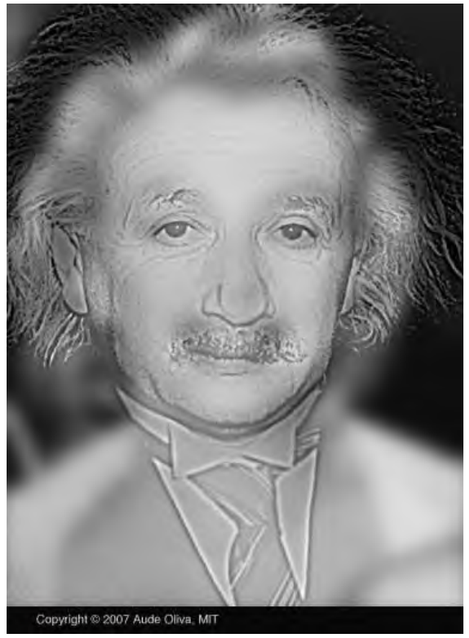
另一个错觉：他是谁？

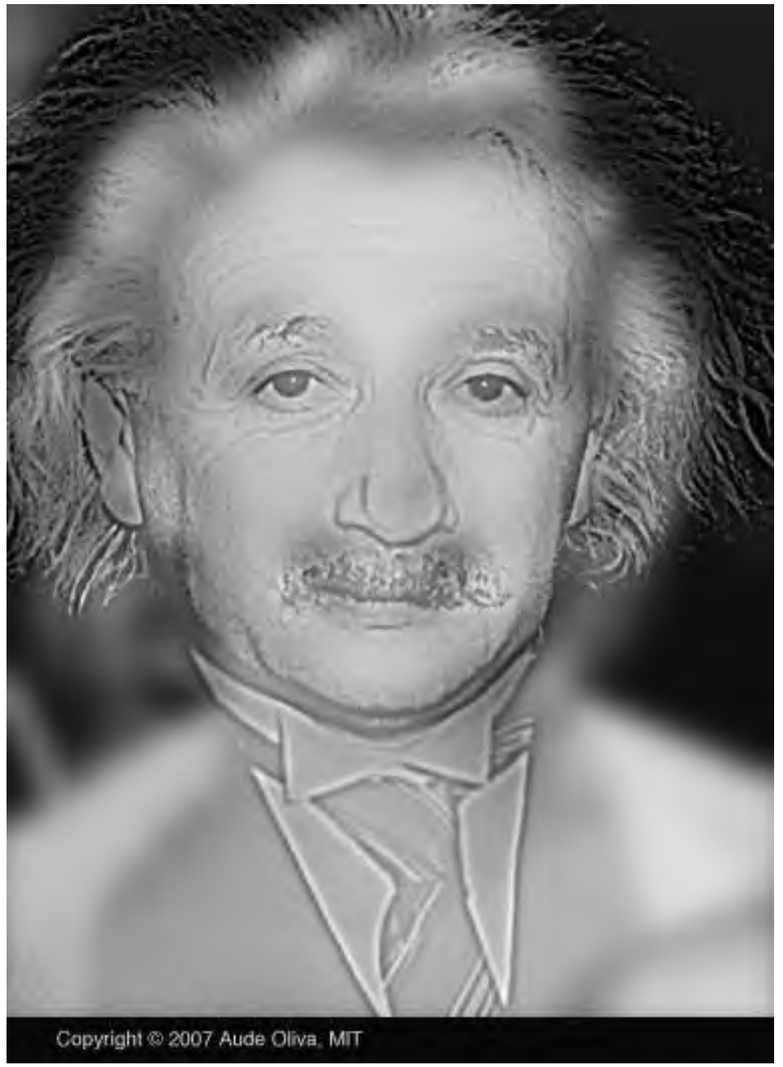


他又是谁?

其实是同一张图!

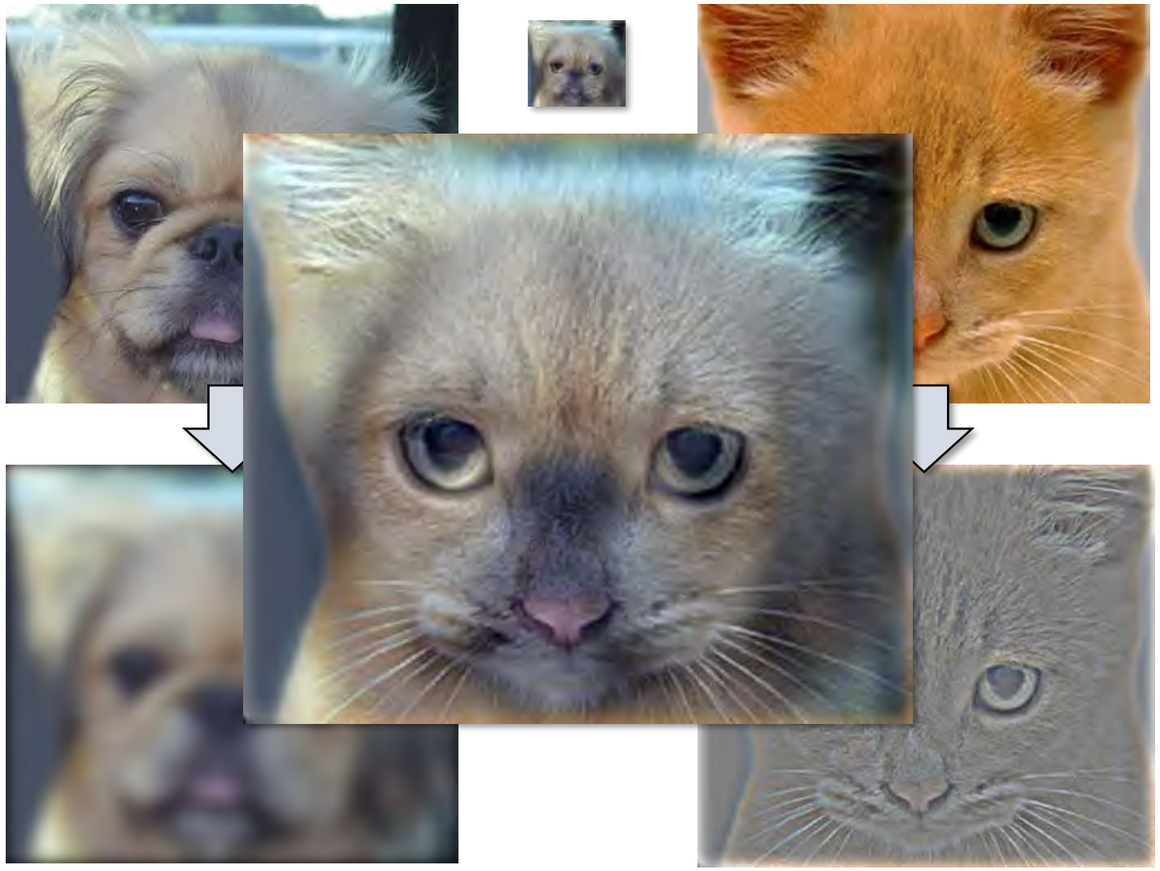
他是谁?


Copyright © 2007 Aude Oliva, MIT


Copyright © 2007 Aude Oliva, MIT

Homework1

形成原理



在不同尺度下
有不同的关注点

模糊

边缘

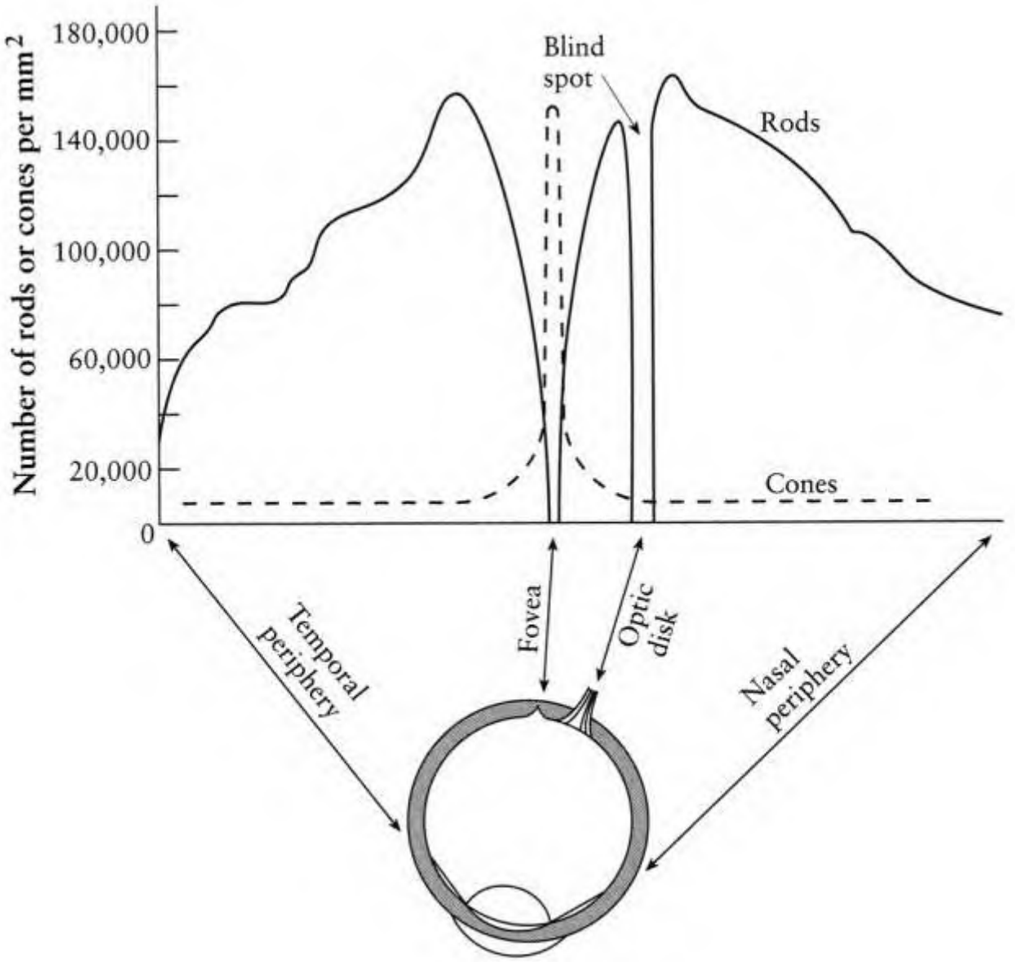
Homework1

- 自选**2**组图做前几页slides中提到的合成图测试
- 熟悉Python基础图像处理操作、感受多尺度视觉感知的变化
- Deadline： **2023/10/19**

回顾——光照感知

视杆 与 视锥

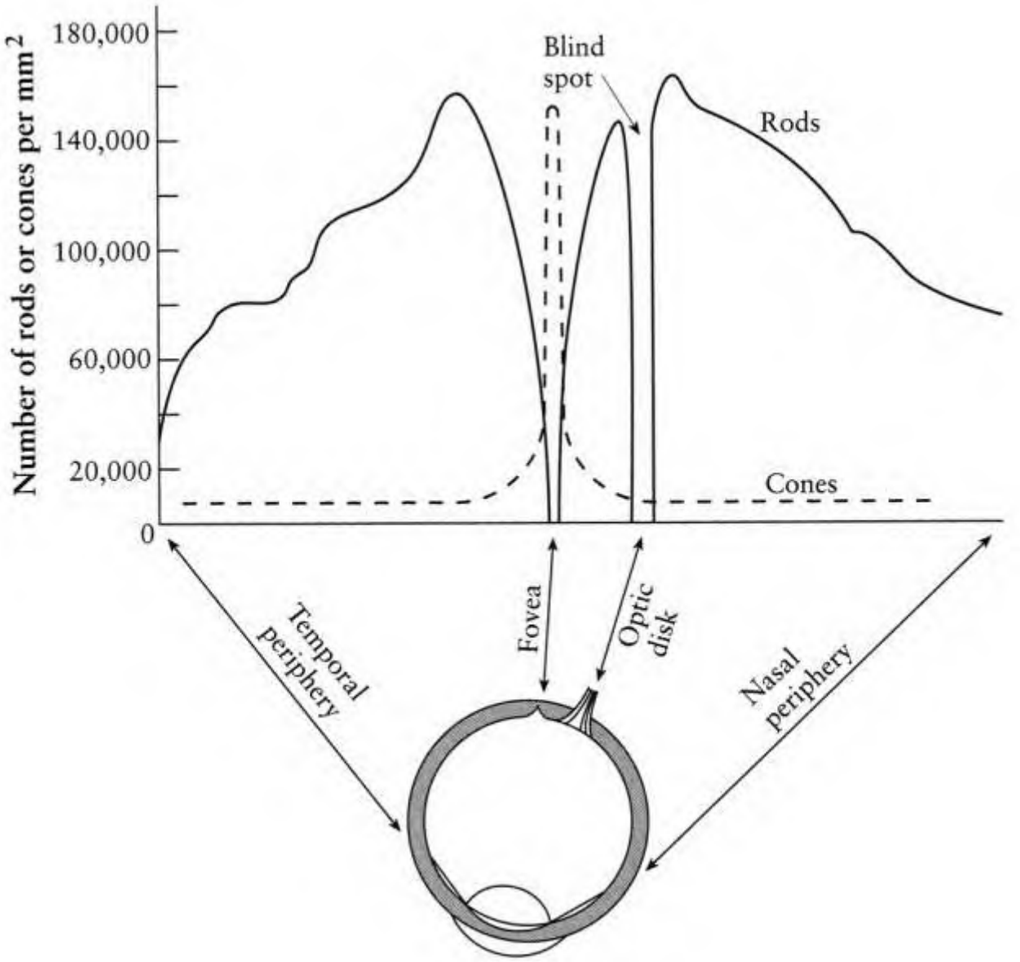
- 视杆与视锥是不均匀地分布在视网膜上的
  - 视杆对光的强度更敏感，视锥对颜色更敏感。明亮光照下，视锥细胞主导，而在暗光下，视杆细胞起主导作用
  - Fovea **中间凹**: 视网膜上的一个小区域(1 or 2°)，包含大量视锥细胞（**没有视杆**），特别适用于高分辨率的颜色感知。它在颜色识别和高分辨率视觉中发挥关键作用。
  - 不在中央凹区域的部分，对于感知低光强度或运动检测非常有用，但视力锐度较低。



盲点

- 神经盘是视网膜上的一个区域，位于视网膜背部，是视觉信息传输的起点。它是视神经纤维束聚集的地方，这些纤维将视觉信号传送到大脑的视觉中枢。

- **视神经盘上没有感光细胞**，因此在视觉场景中，与视神经盘对应的区域被视为视野中的**盲点**。这是我们视觉系统中的一个盲区，无法感知光或图像。

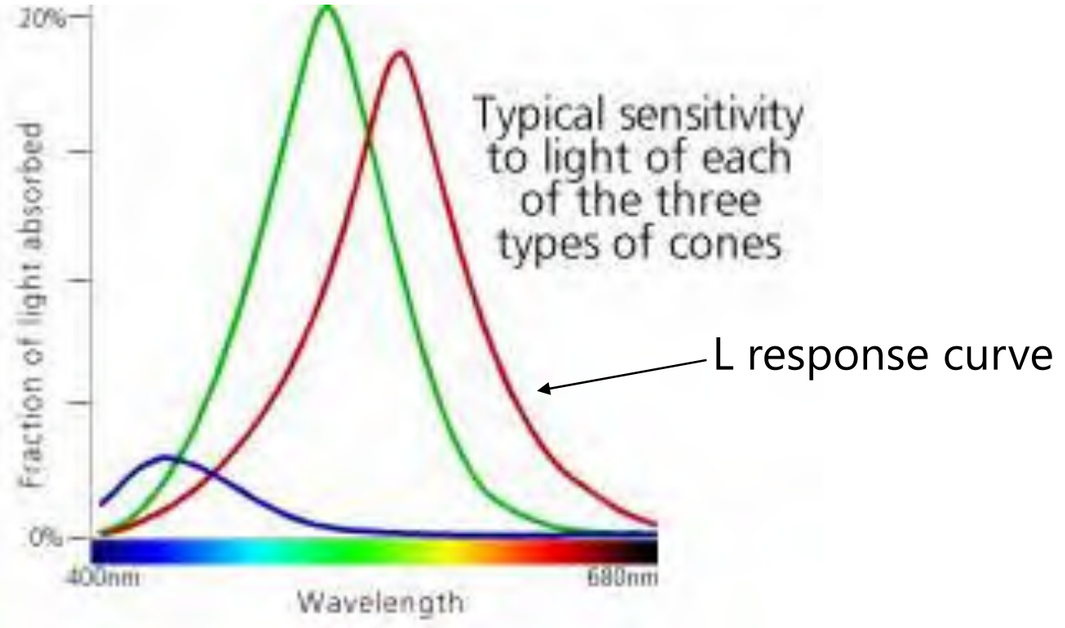


Recall：消失的圆点

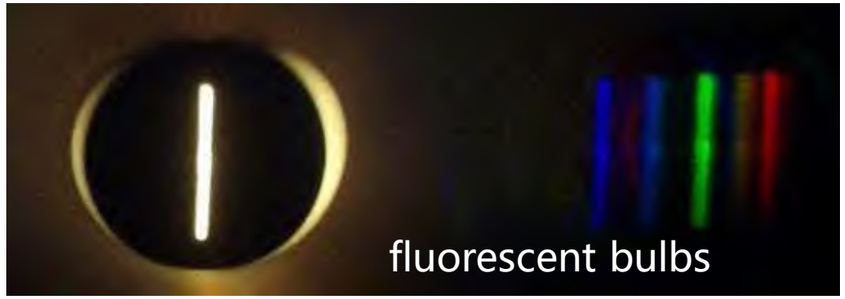
With left eye shut, look at the cross on the left. At the right distance, the circle on the right should disappear (Glassner, 1.8).

颜色感知

- 三种色锥感受颜色
  - 每种色锥对光谱中的区域更敏感
    - 这些区域相交
    - Short (S) 对应 **蓝色**
    - Medium (M) 对应 **绿色**
    - Long (L) 对应 **红色**
  - 总体而言，我们对红色和绿色更敏感
    - 但每个人的敏感度不同，也会随年龄变化
  - 色盲的原因是至少一种色锥出现了退化

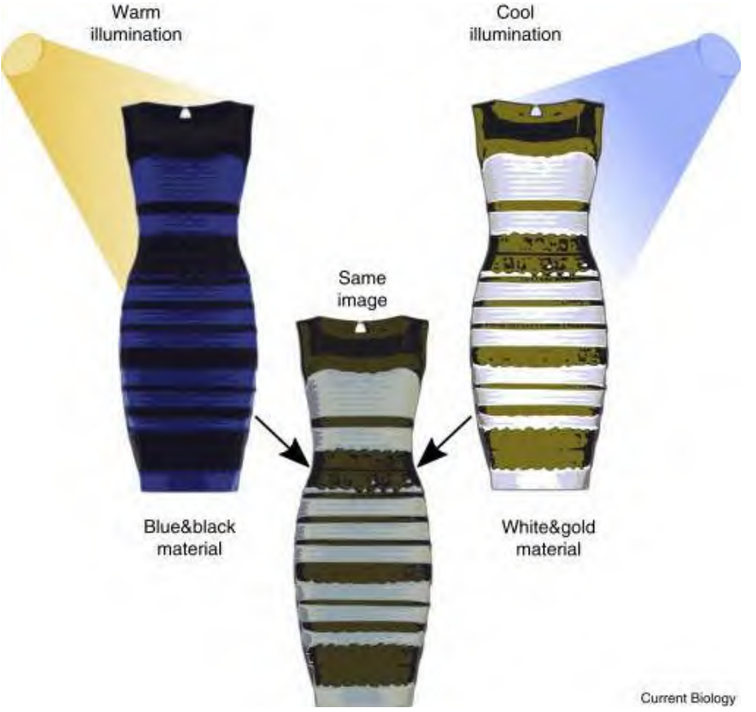


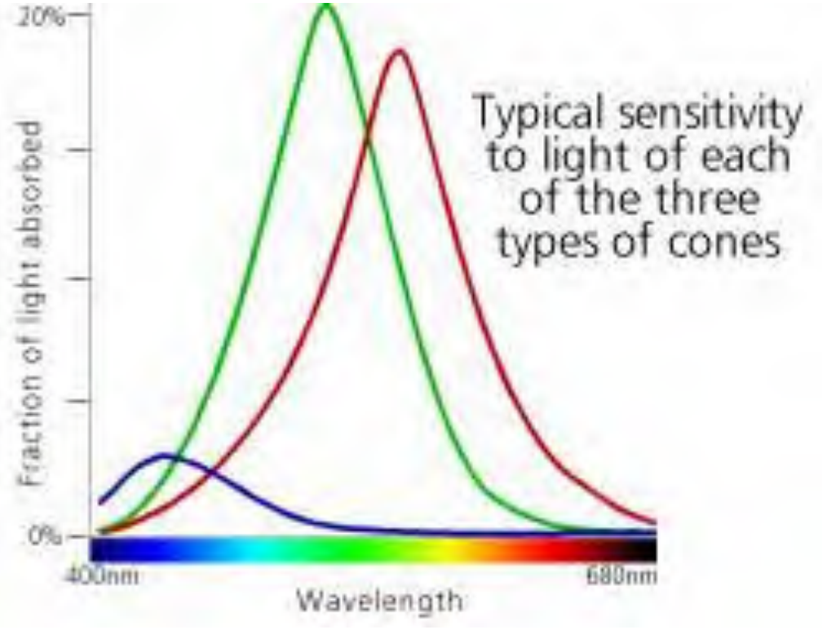
用分光镜（Spectroscope）区分不同的灯泡

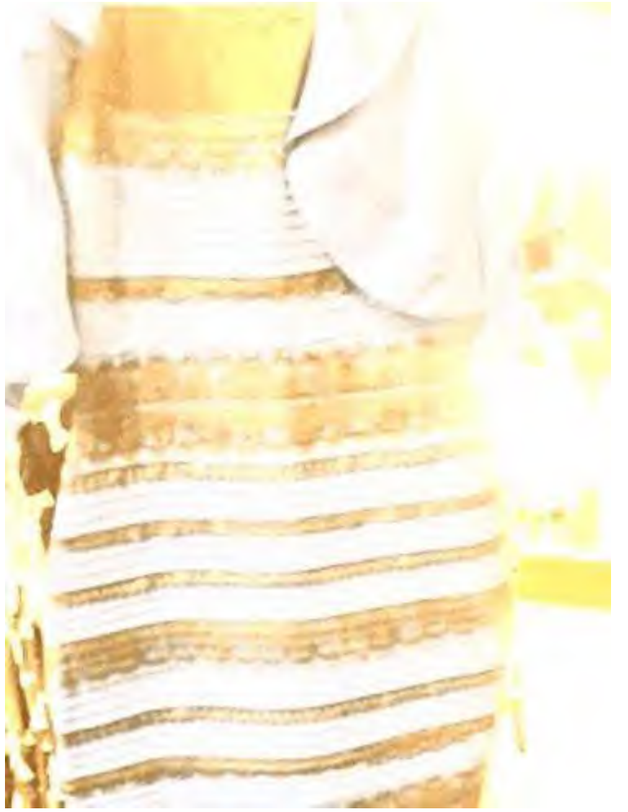


http://www.chemistryland.com/CHM107Lab/Exp7/Spectroscope/Spectroscope.html

蓝黑还是白金?


Warm
illumination
Cool
illumination
Same
image
Blue&black
material
White&gold
material
Current Biology


20%
0%
Fraction of light absorbed
400nm
680nm
Wavelength
Typical sensitivity
to light of each
of the three
types of cones



“感知”

计算机系统对数字图像和视频进行分析和理解的能力，以获得有关物体、场景、特征和动作的信息。感知的目标是模拟和模仿人类视觉系统，使计算机能够理解和解释图像中的内容。

“感知”

特征信息提取、对象识别、场景理解、运动分析、三维深度感知、
上下文感知 .....

“感知”

从 几何模型：基于几何原理和相对位置关系的模型

到 统计模型：基于数据分布和统计推断的模型

计算机如何感知?

- 让我们以人为例...
  - “音容笑貌”
  - “你掌心的痣，我总记得在哪里”

计算机如何感知?

- 让我们以人为例...
  - “音容笑貌”
  - “你掌心的痣，我总记得在哪里”

“模式”
Pattern

$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

“特征”
Feature

计算机如何感知?

- 让我们以人为例...
  - “音容笑貌”
  - “你掌心的痣，我总记得在哪里”

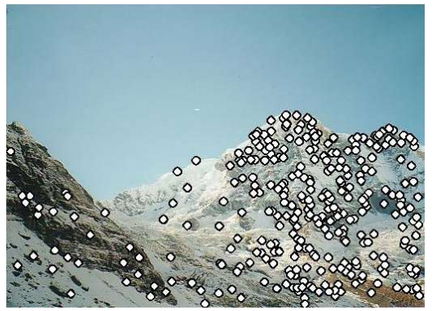
“模式”
Pattern

$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

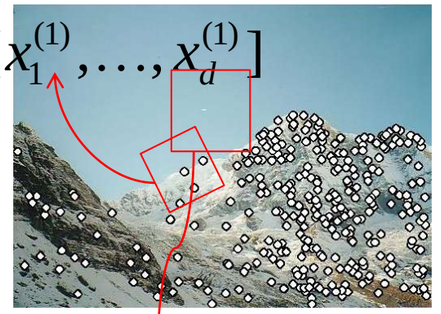
“特征”
Feature

计算机的视觉认知：基于特征匹配的识别

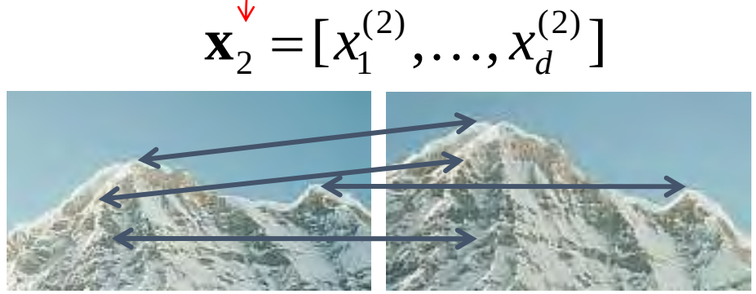
1) 检测 Detection: 找到图中的关键点



2) 解释 Description: 在关键点周围提取特征



3) 匹配 Matching: 根据两个视角下的特征进行匹配



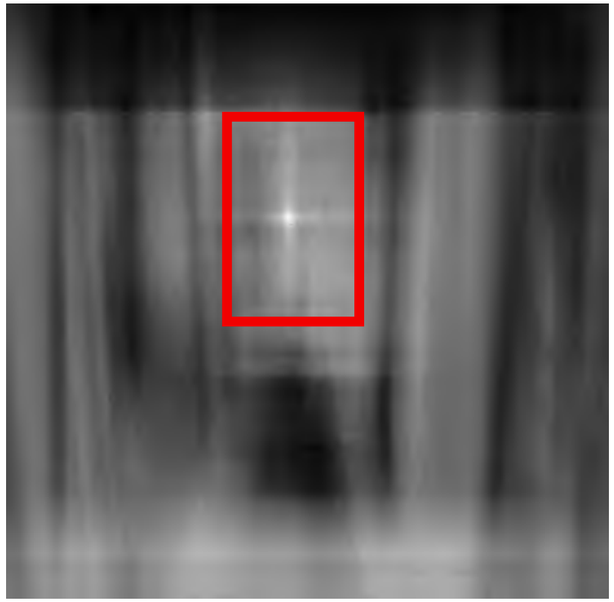
Kristen Grauman

挑战是什么？

一把椅子

找出图像中的椅子

特征匹配度热图



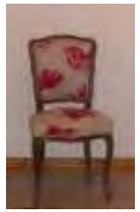
这样做有什么问题?

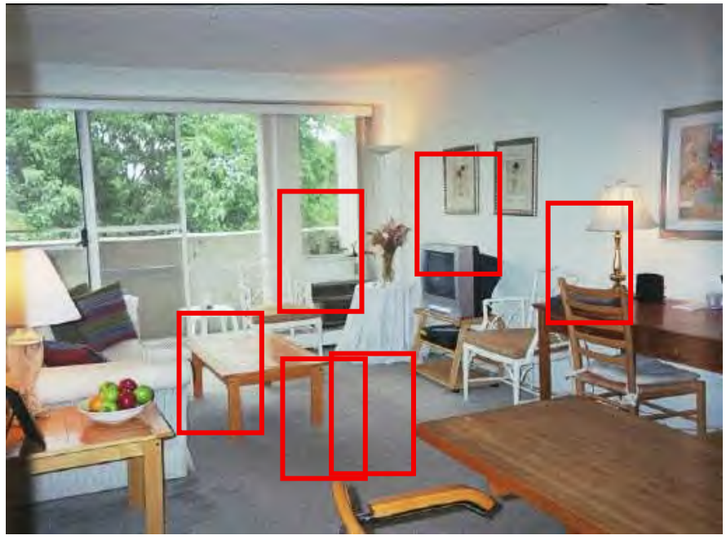
- 视觉信息质量可能较低
  - 模糊、遮挡、噪音

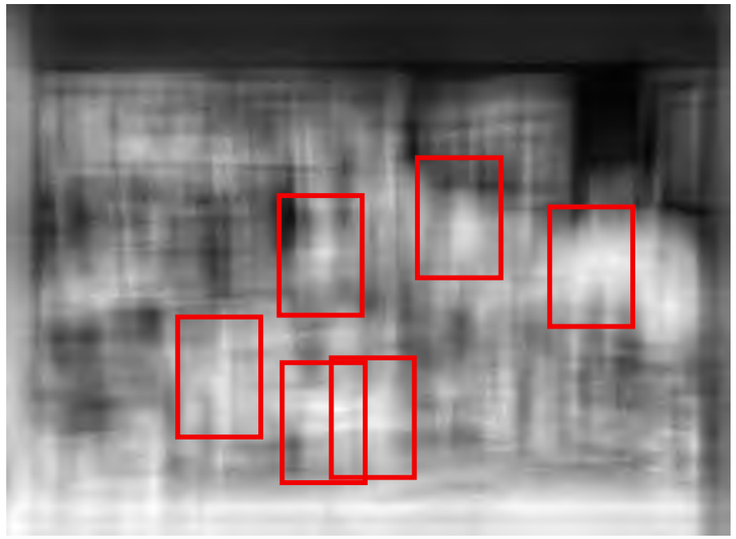
- 实时性、通用性挑战
  - 外观多变、视觉错觉

挑战是什么？

找到图中的椅子







绝大多数目标框都没有目标物体

目标物体在新场景下不会一成不变

通用性挑战：不同视角

Michelangelo 1475-1564

通用性挑战：不同光照条件

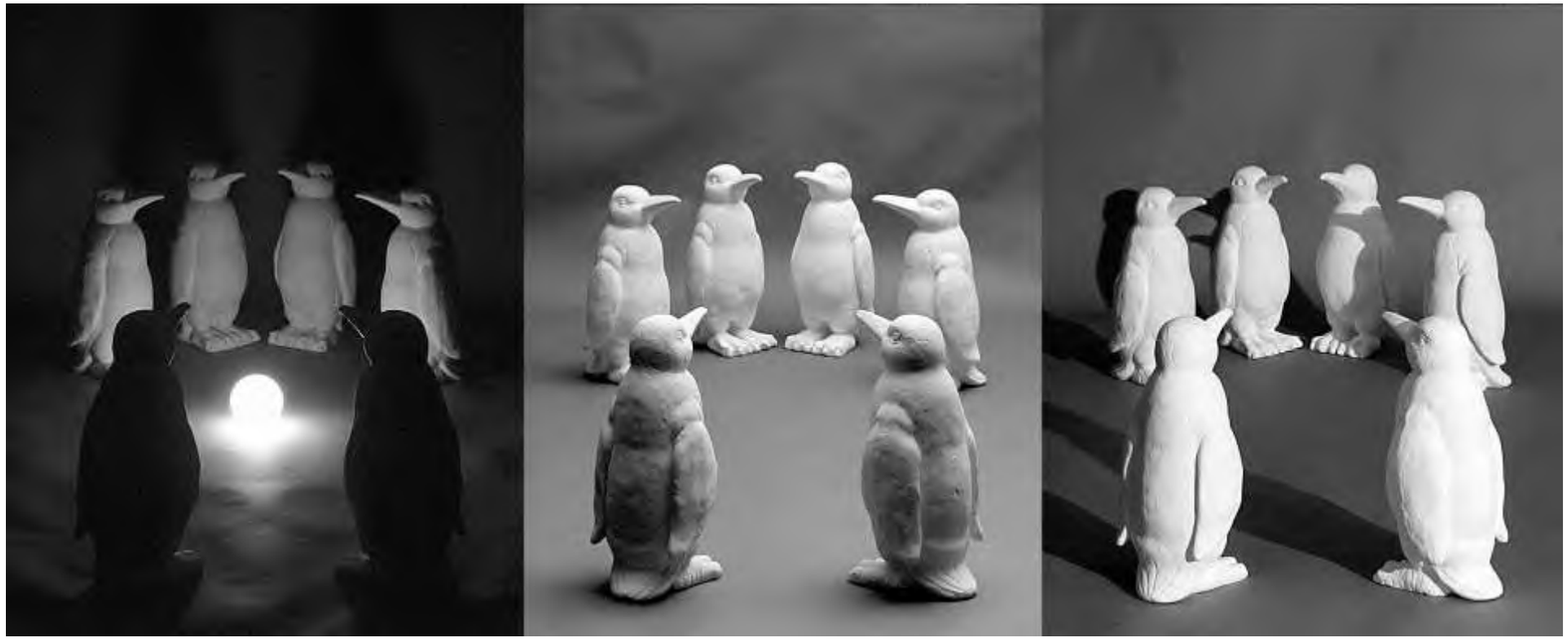


image credit: J. Koenderink

通用性难点：类内差异

同类目标外观差异很大

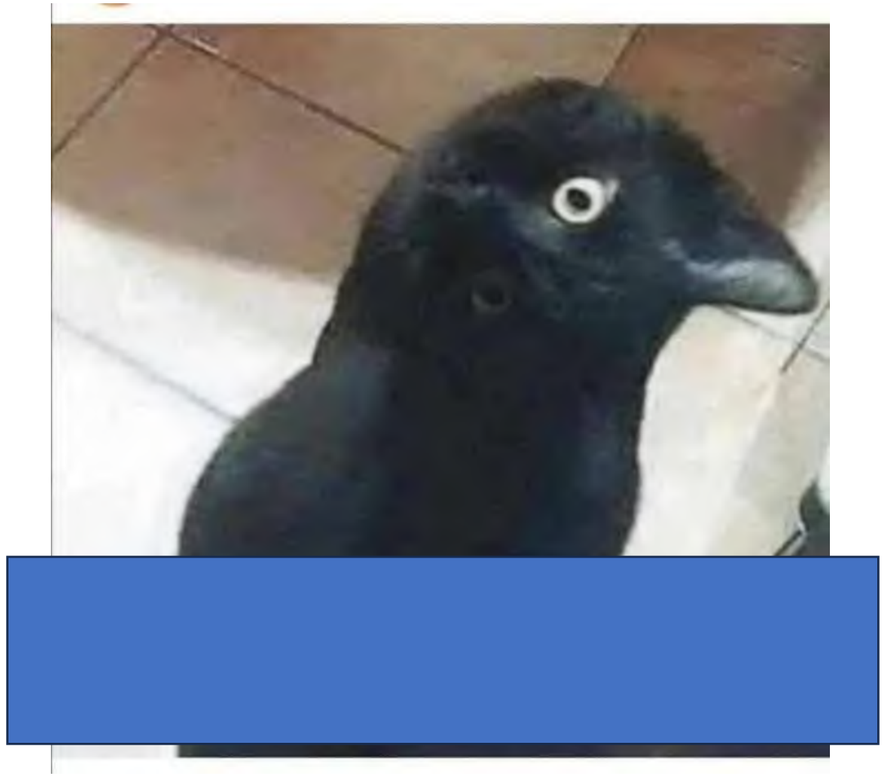
通用性挑战：模糊与局部歧义

单独局部无法推断内容

slide credit: Fei-Fei, Fergus & Torralba

通用性难点：极端任务

Brady, M. J., & Kersten, D. (2003). Bootstrapped learning of novel objects. J Vis, 3(6), 413-422



https://www.dcard.tw/f/funny/p/233833012

视觉错觉：感知对于人们来说同样有挑战

- 感知本身就是一个定义模糊的问题
  - 三维世界与二维世界的感知区别
  - 感知需要大量知识储备

视觉错觉下的可乐瓶
Artist Julian Beever

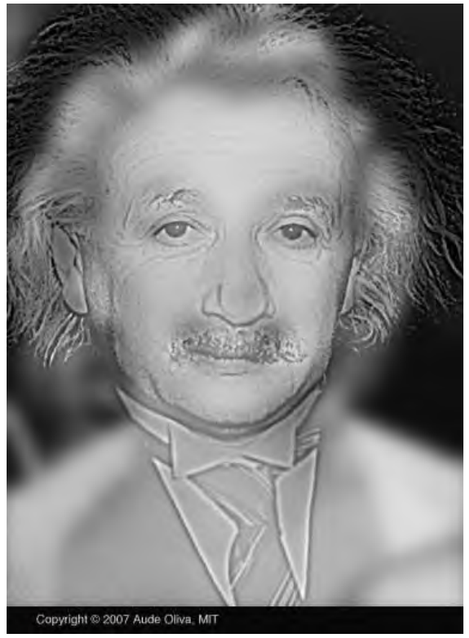
Image source: F. Durand

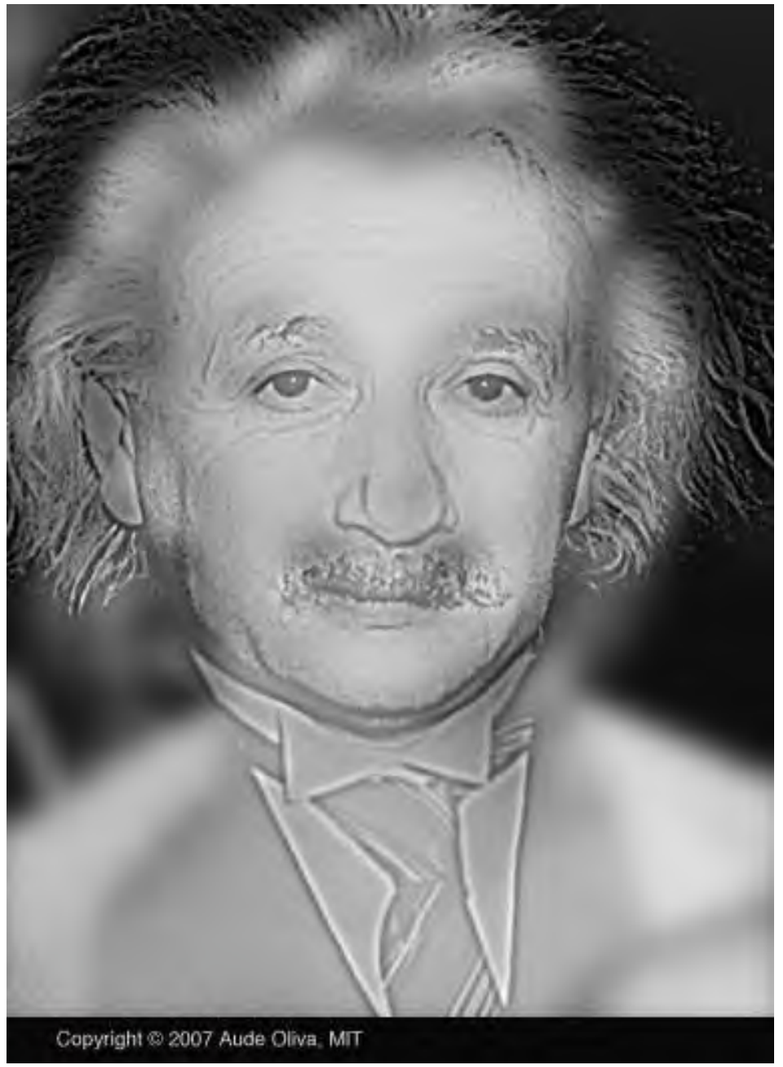
视觉错觉

视觉错觉

With left eye shut, look at the cross on the left. At the right distance, the circle on the right should disappear (Glassner, 1.8).

尺度问题


Copyright © 2007 Aude Oliva, MIT


Copyright © 2007 Aude Oliva, MIT

尺度问题: Wagon-wheel 效应

https://en.wikipedia.org/wiki/Wagon-wheel_effect

人类如何解决这些困难？

- 多次认证
  - 灵活调整自身状态
  - 总有一个最好的尺度、没有错觉的视角

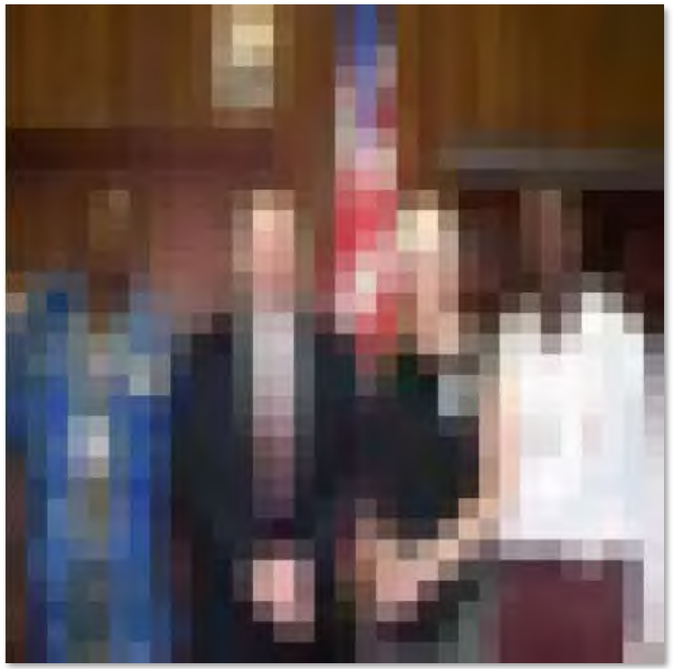
- 记忆、推理
  - 经验、智能

人类的优势：调整视角


MAKE POVERTY HISTORY
CoolOpticalIllusions.com


CoolOpticalIllusions.com

人类的优势：经验推理



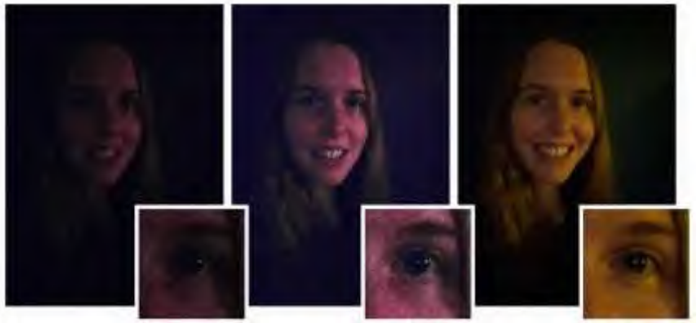
Source: “80 million tiny images” by Torralba, et al.

计算机的优势：图像处理

- 增强图像的 (“Computational Photography”)

超分辨率 **放大，放大，再放大！**
(source: 2d3)

背景虚化 **柔光双摄！** (source: Google Research Blog)



亮度调整 **照亮你的美！**
(credit: Hasinoff et al., SIGGRAPH ASIA 2016)

图像补全 **一键修图！**
(image credit: Hays and Efros)

计算机的优势：感知可以利用大量的冗余

一个房子也许认不出来…

一百个房子认出来了！

Source: S. Lazebnik

“感知”

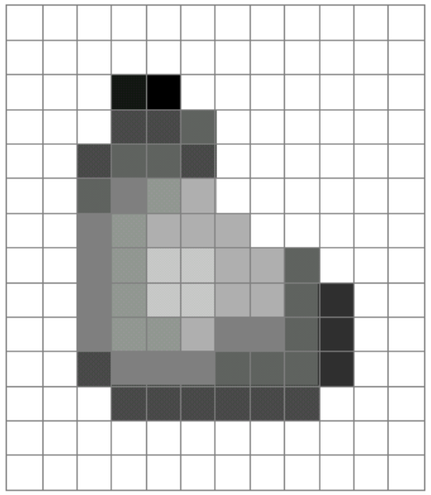
从 几何模型：基于几何原理和相对位置关系的模型

到 **统计模型：基于数据分布和统计推断的模型**

我们需要统计模型快速合理地进行感知推断

什么是图像?

- 代表光强度的矩阵



=

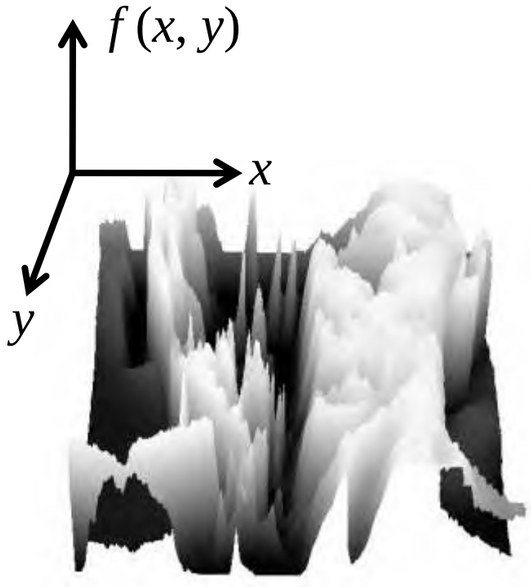
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(一般用8-bit（byte）整数表示每个值: 0 = black, 255 = white)

什么是图像?

- 可以认为是R^2 到 R的映射:
  - $f(x,y)$ 给出(x,y)的强度

snoop



3D view

图像变换

- 可以应用到图像的变换函数

$g(x,y) = f(x,y) + 20$

$g(x,y) = f(-x,y)$

- 接下来我们会具体介绍一个十分常用的变换 卷积 *convolution* (线性滤波)
  - 是图像处理的基础操作
  - 也是后续处理实时性和通用性的基础单元

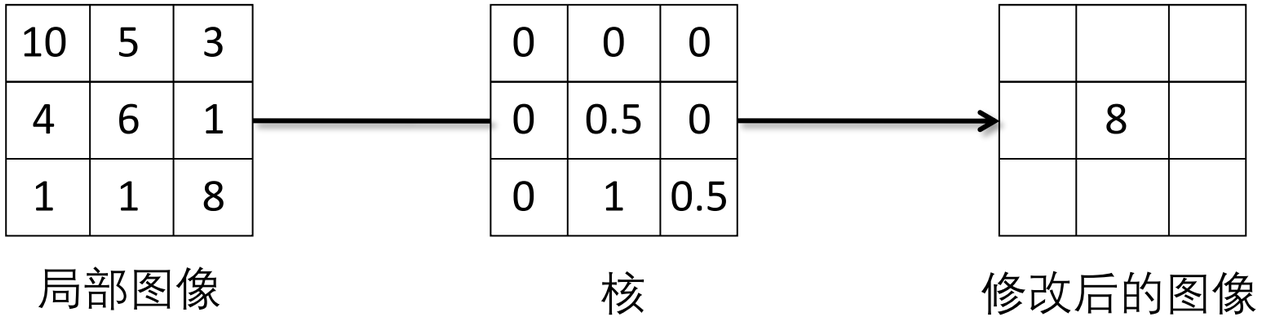
图像滤波

- 基于局部图像和某个函数修改像素的值

Source: L. Zhang

线性图像滤波

- 一种简单的图像滤波：线性滤波(cross-correlation, convolution)
  - 用像素点周围的值的线性组合来替代自身的像素值
- 函数可以被 核“kernel” (or “mask”, “filter”)表示



Source: L. Zhang

Cross-correlation 互相关

F： 图像, H： 核 (大小 2*k*+1 x 2*k*+1), G： 输出图像

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

这就是 **cross-correlation 互相关**：

$$G = H \otimes F$$

- 可以认为是图像局部与核的相关性（内积）

Convolution 卷积

- 与互相关类似，不过水平垂直反转

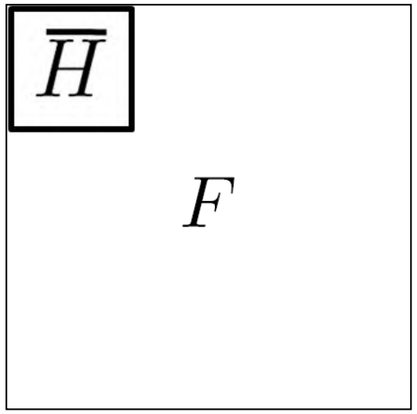
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

这就是 **convolution** 卷积:

$$G = H * F$$

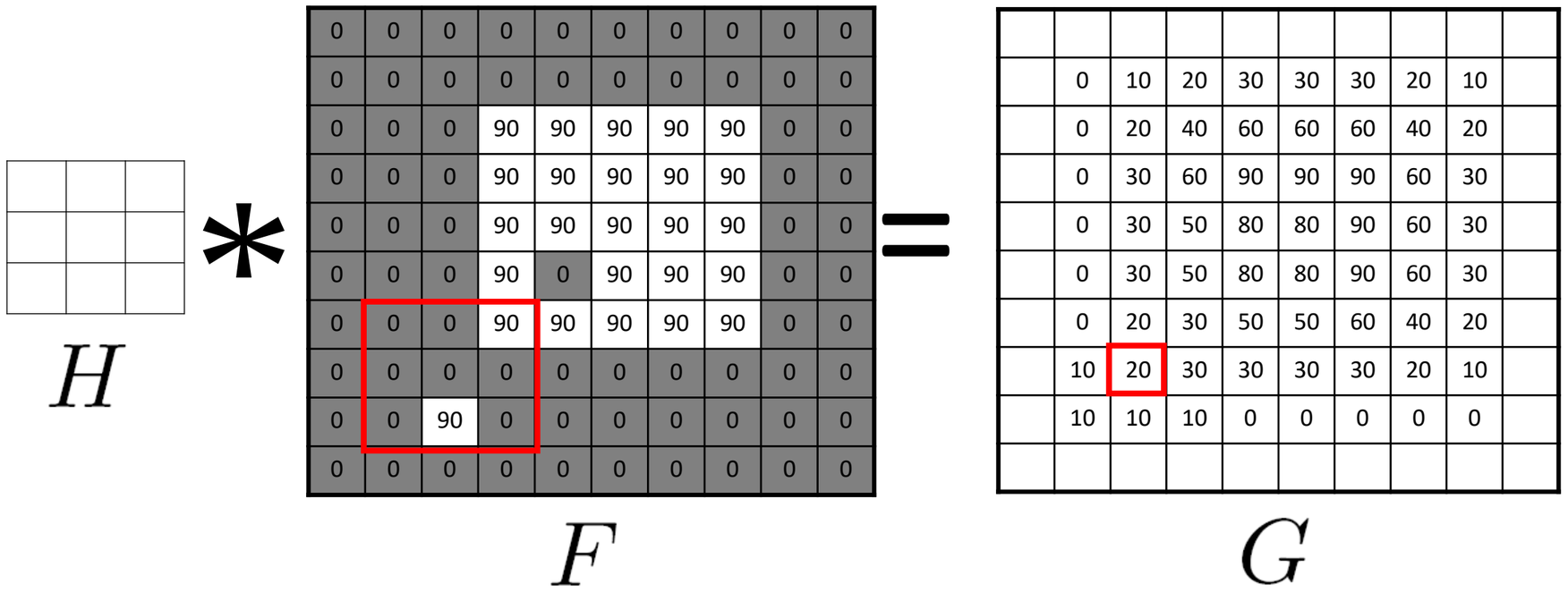
- 卷积具有 **交换律** 和 **结合律**

卷积

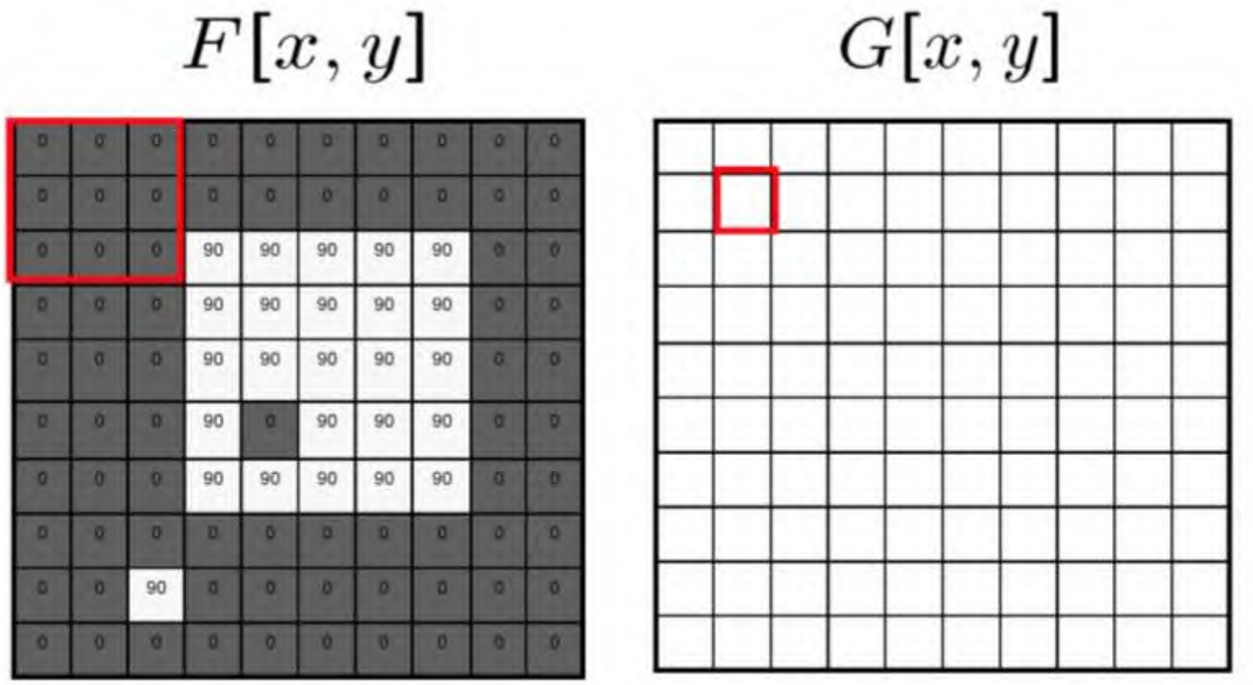


Adapted from F. Durand

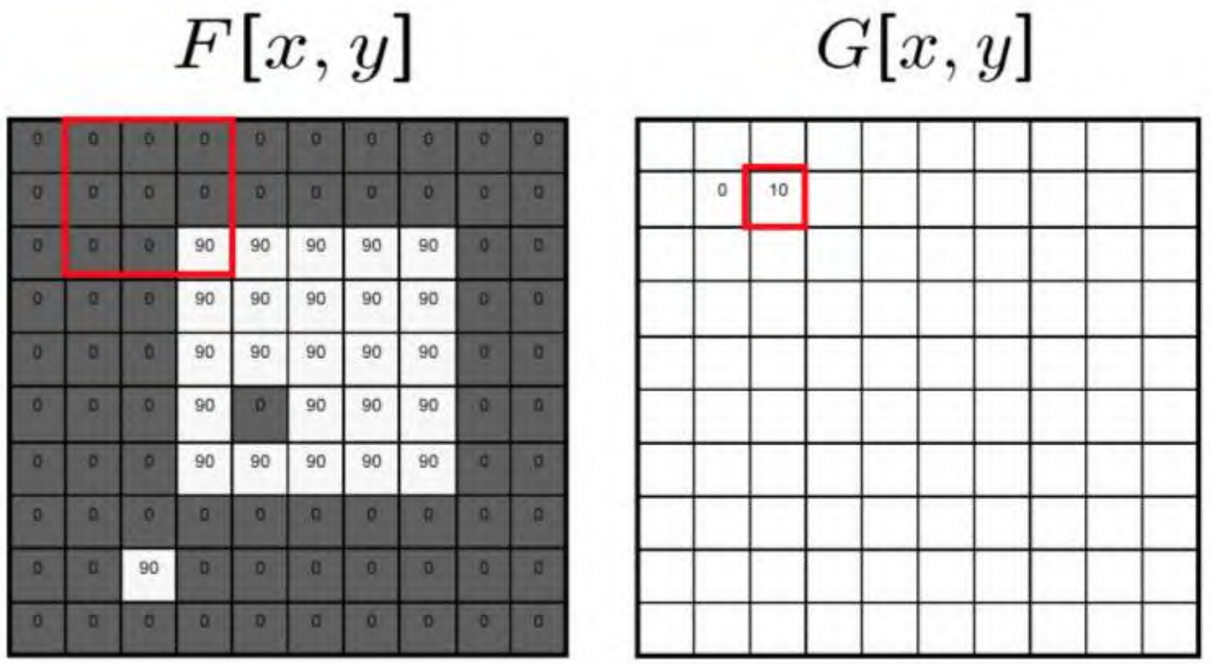
Mean filtering 平均滤波



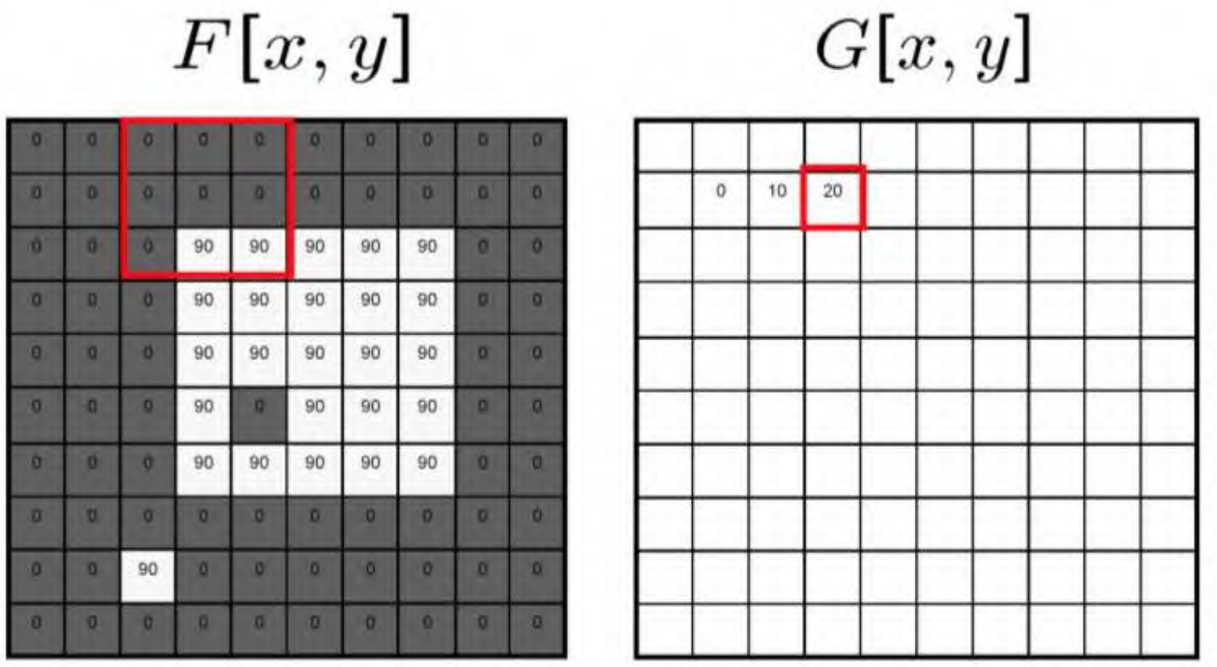
平均滤波


$F[x, y]$
$G[x, y]$
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

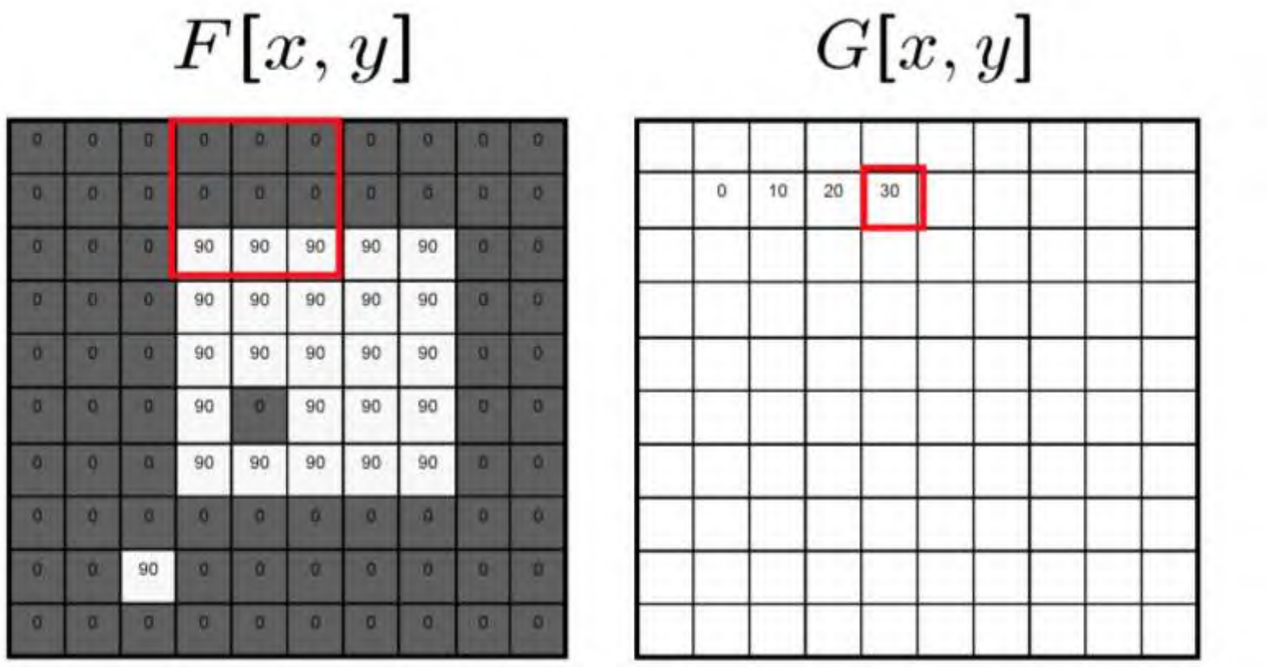
平均滤波


$F[x, y]$
$G[x, y]$
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 10

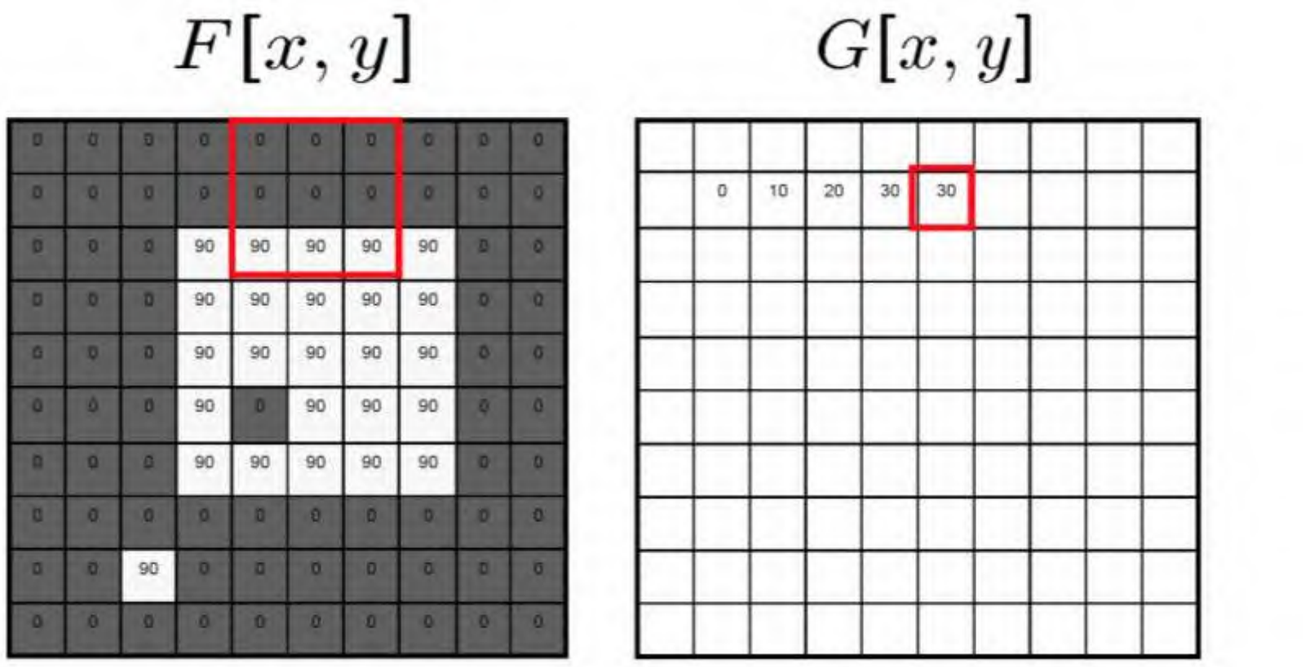
平均滤波


$F[x, y]$
$G[x, y]$
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 10 20

平均滤波


$F[x, y]$
$G[x, y]$
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 10 20 30

平均滤波


$F[x, y]$
$G[x, y]$
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 10 20 30 30

平均滤波

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

线性滤波

原图

$$*\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Source: D. Lowe

线性滤波

原图

$*$

0	0	0
0	1	0
0	0	0

$=$

同一张图

Source: D. Lowe

线性滤波

原图

*

0	0	0
1	0	0
0	0	0

Source: D. Lowe

线性滤波

原图

$$*\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

向左移动一个像素

Source: D. Lowe

线性滤波

原图

$$* \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

模糊（平均滤波）

Source: D. Lowe

线性滤波

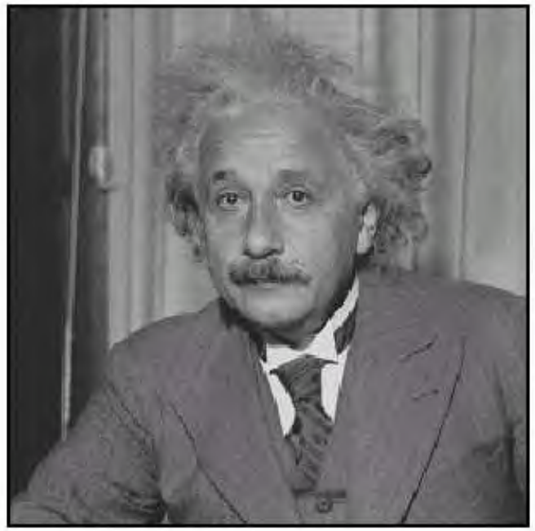
原图

$$*\left(\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix}-\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\right)=$$

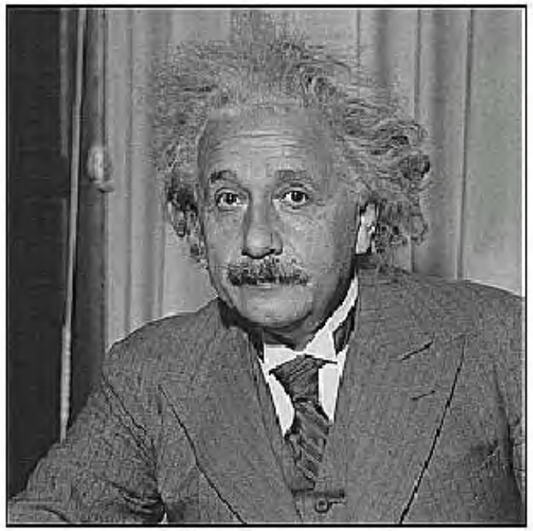
锐化后
(突出边缘)

Source: D. Lowe

图像锐化



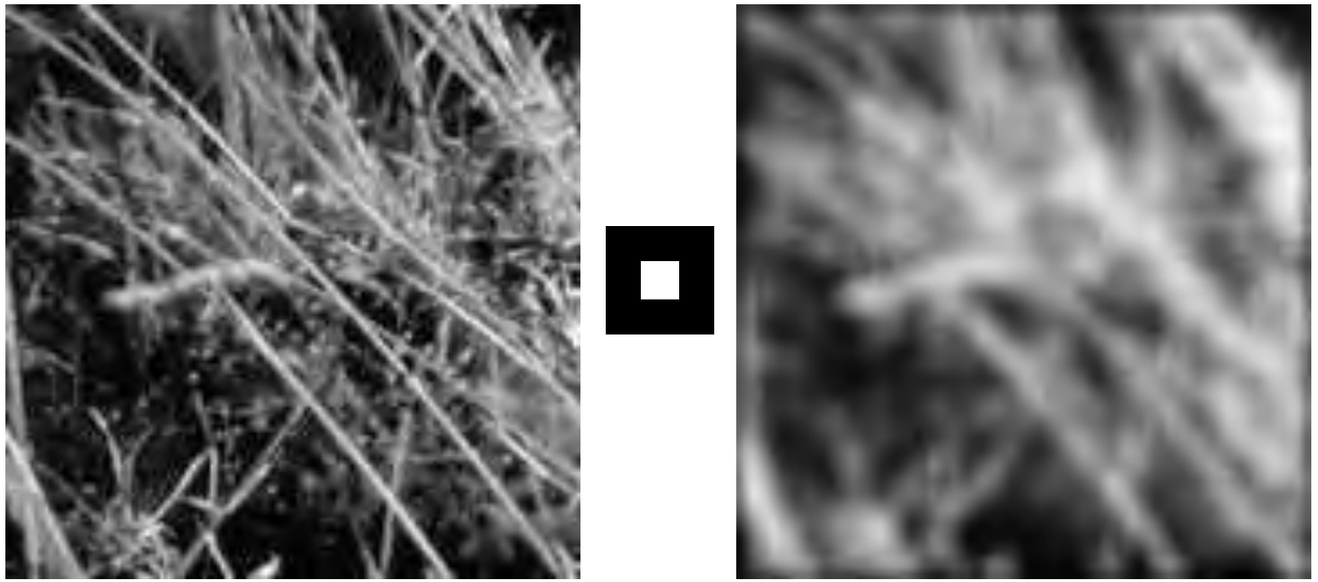
before



after

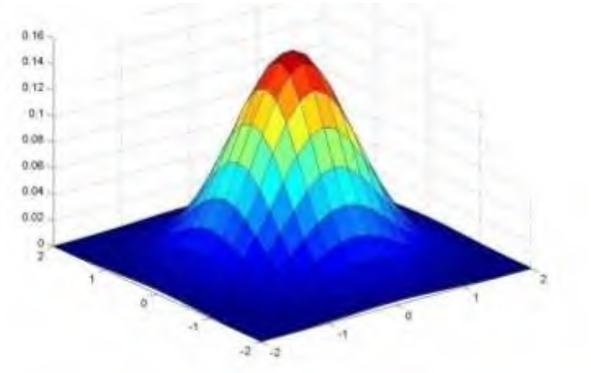
Source: D. Lowe

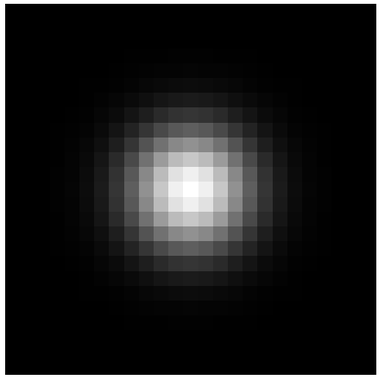
图像模糊



Source: D. Forsyth

高斯核

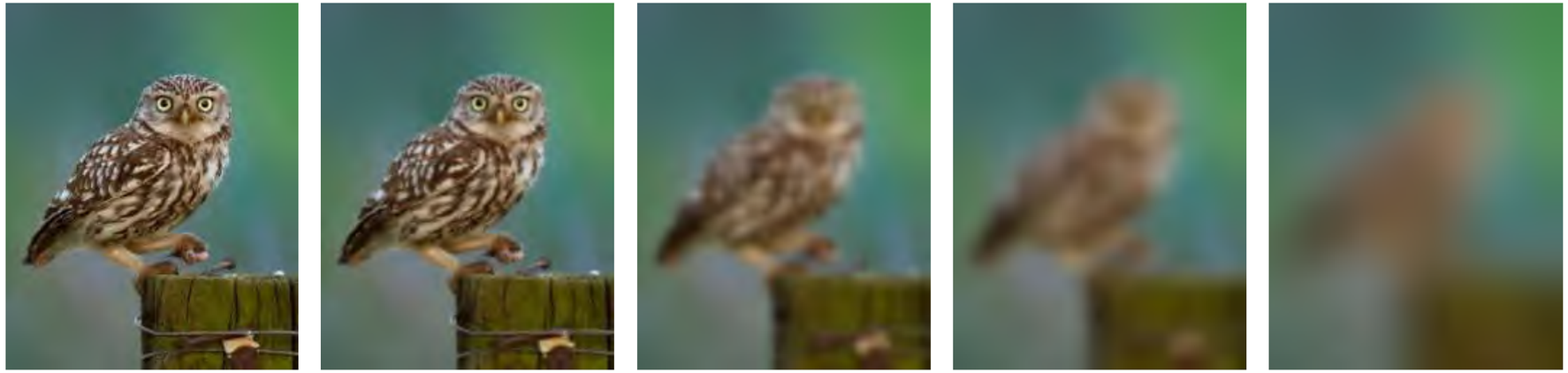




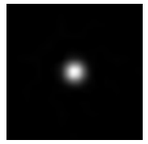
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

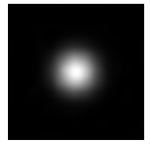
高斯滤波



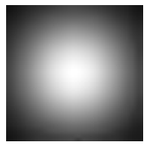
σ = 1 pixel



σ = 5 pixels

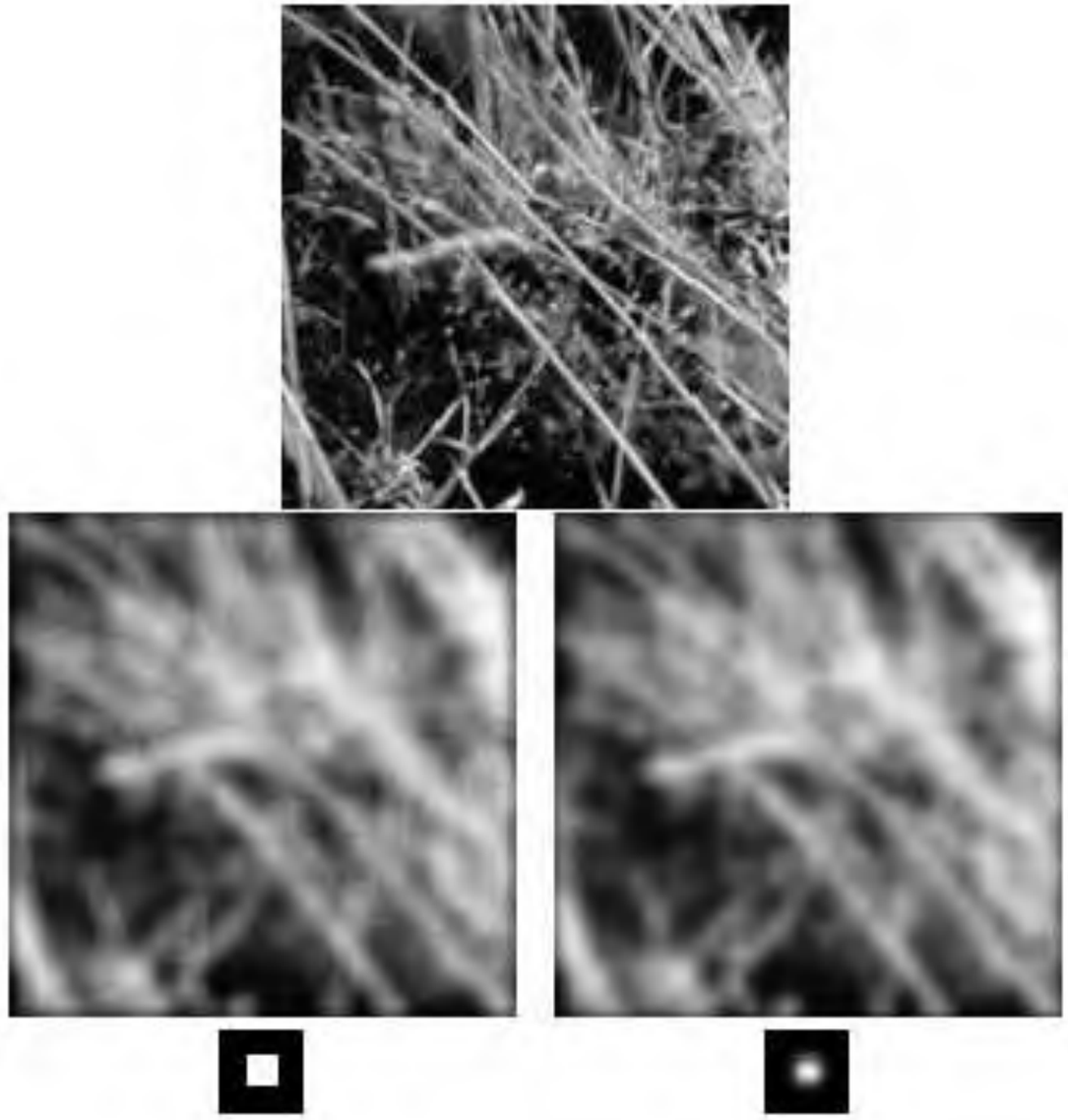


σ = 10 pixels



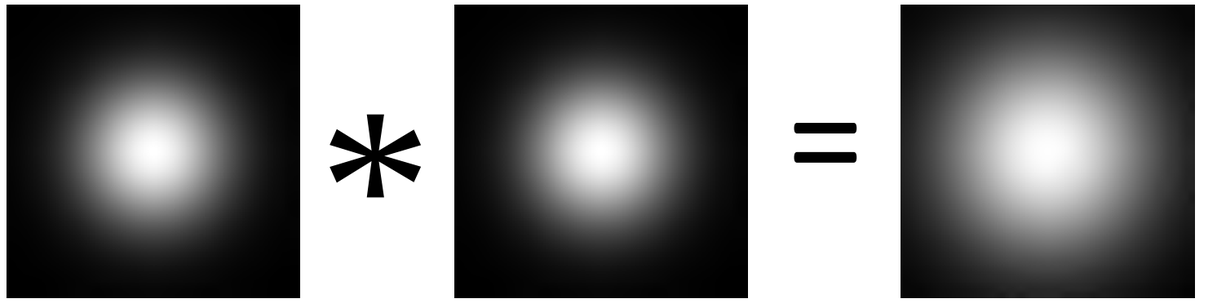
σ = 30 pixels

平均滤波 vs 高斯滤波



高斯滤波

- 去掉 “high-frequency 高频” 图像 (low-pass filter 低通滤波)
- 如果高斯滤波自我叠加



- 核宽为σ　卷积后核宽为$\sigma\sqrt{2}$

Source: K. Grauman

锐化

- 模糊带走了什么?

(“细节提取”：高通滤波 ***high-pass filter***)

把细节加回来:

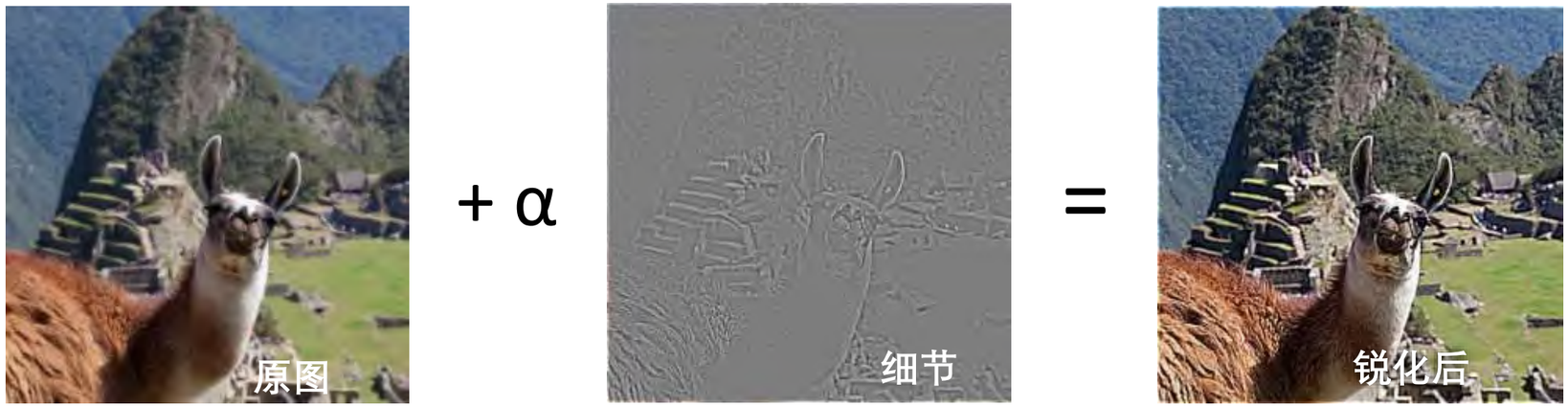


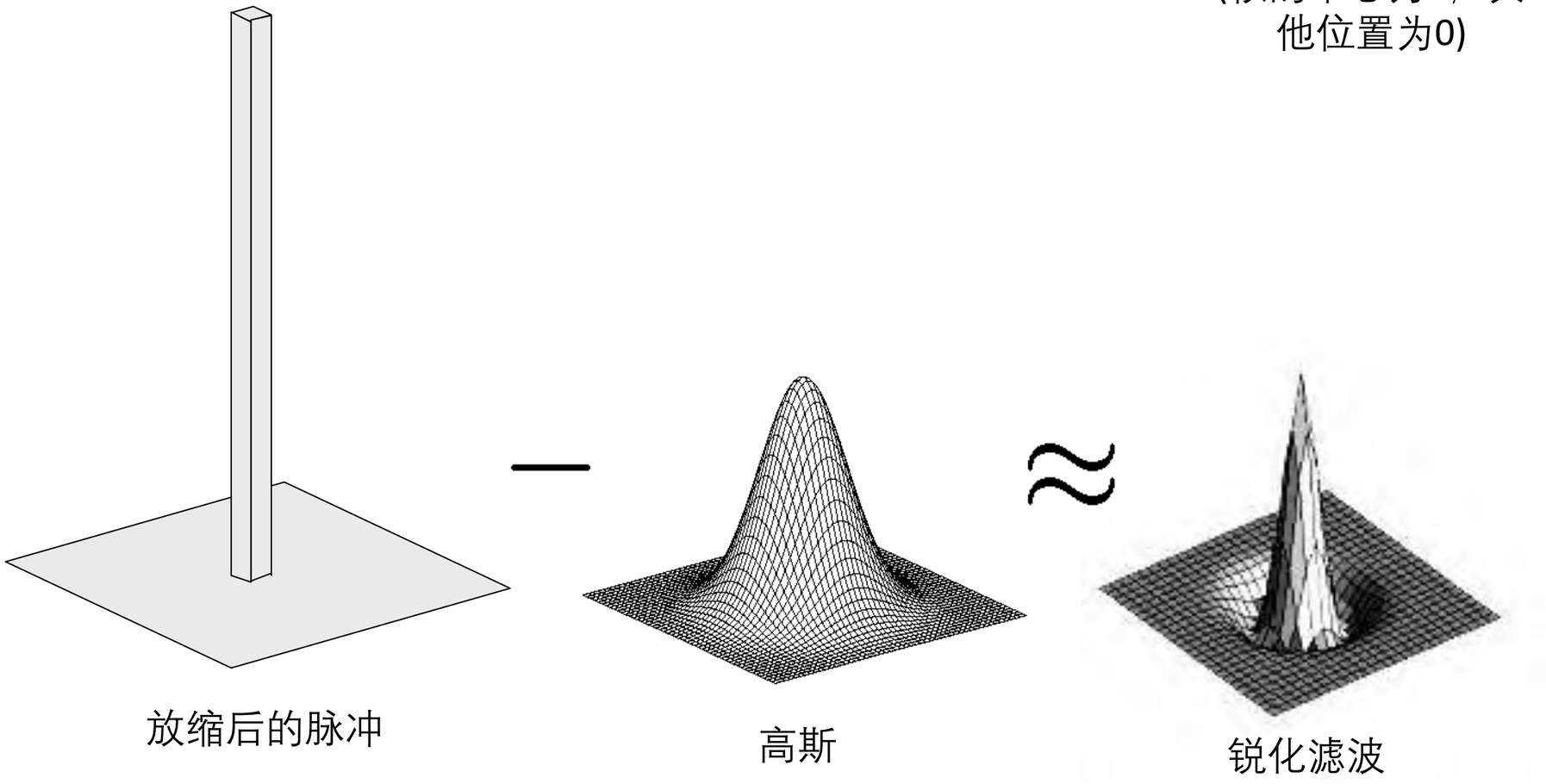
Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

锐化滤波

$$F + \alpha\left(F - \underbrace{F * H}_{\text{模糊后的图像}}\right) =$$

图像

单位脉冲
(核的中心为1，其他位置为0)



放缩后的脉冲

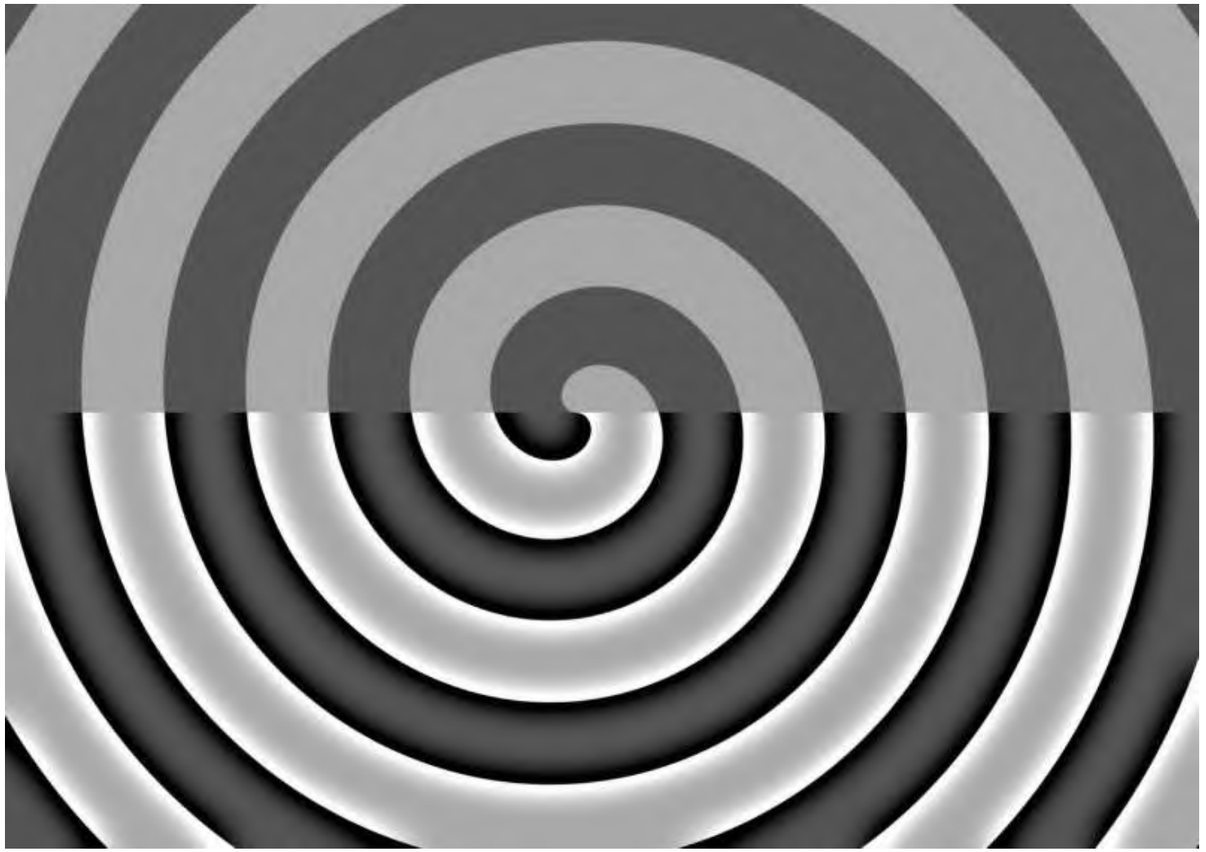
高斯

锐化滤波

锐化滤波

锐化前

锐化后



“光学”卷积

镜头晃动

Source: Fergus, *et al.* “Removing Camera Shake from a Single Photograph”, SIGGRAPH 2006

虚化模板

Source: https://www.diyphotography.net/diy_create_your_own_bokeh/

回顾——“感知”、卷积

计算机如何感知

- 计算机系统对数字图像和视频进行分析和理解的能力，以获得有关物体、场景、特征和动作的信息。感知的目标是模拟和模仿人类视觉系统，使计算机能够理解和解释图像中的内容。

- 让我们以人为例...
  - “音容笑貌”
  - “你掌心的痣，我总记得在哪里”

“模式”
Pattern

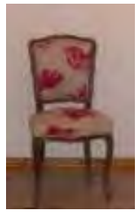
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

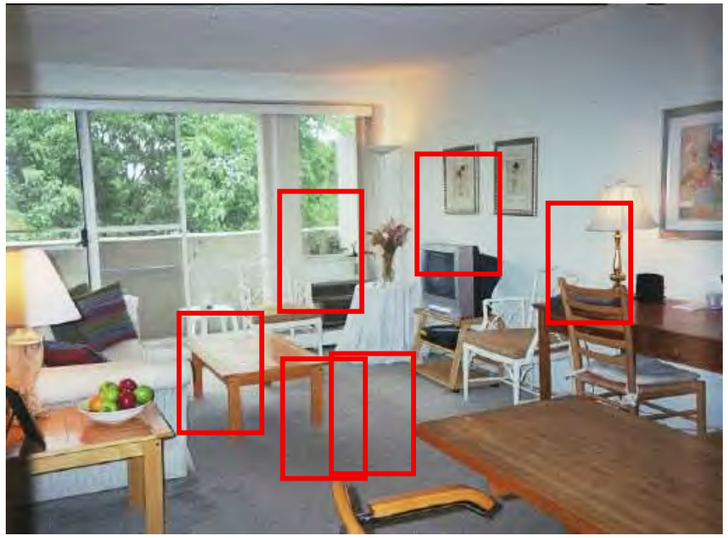
“特征”
Feature

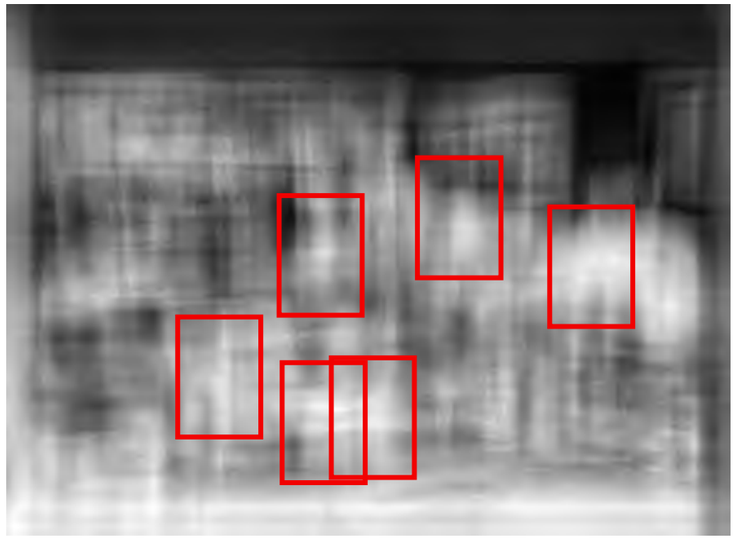
挑战是什么？

- 视觉信息质量可能较低
  - 模糊、遮挡、噪音
- 实时性、通用性挑战
  - 外观多变、视觉错觉

找到图中的椅子







绝大多数目标框都没有目标物体

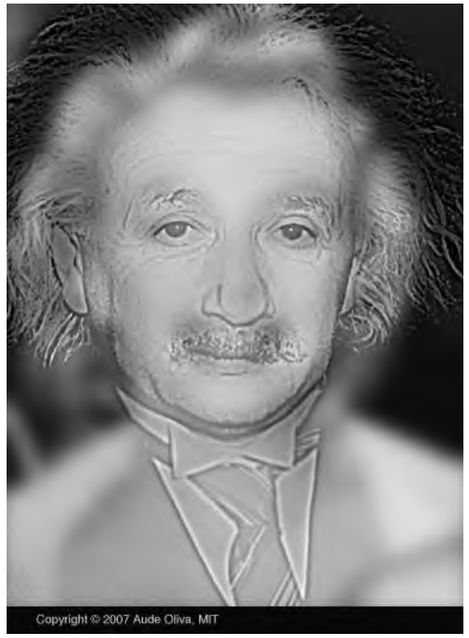
目标物体在新场景下不会一成不变

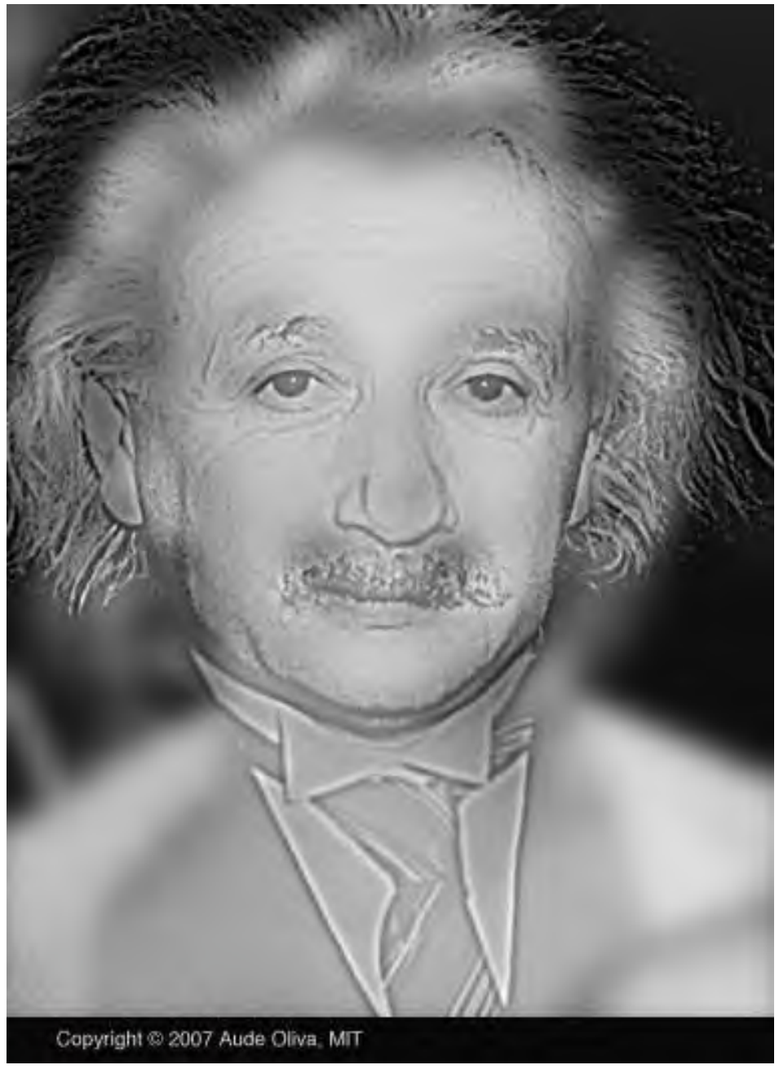
通用性挑战：模糊与局部歧义

单独局部无法推断内容

slide credit: Fei-Fei, Fergus & Torralba

尺度问题

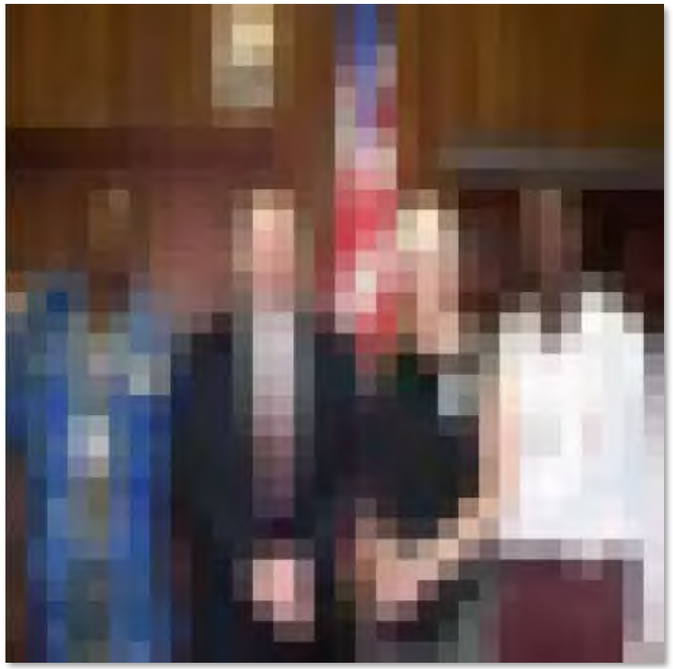

Copyright © 2007 Aude Oliva, MIT


Copyright © 2007 Aude Oliva, MIT

人类如何解决这些困难？

- 多次认证
  - 灵活调整自身状态
  - 总有一个最好的尺度、没有错觉的视角

- 记忆、推理
  - 经验、智能

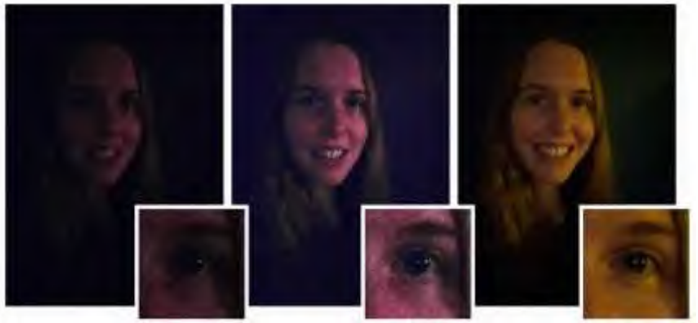


计算机的优势：图像处理

- 增强图像的 (“Computational Photography”)

超分辨率 **放大，放大，再放大！**
(source: 2d3)

背景虚化 **柔光双摄！** (source: Google Research Blog)



亮度调整 **照亮你的美！**
(credit: Hasinoff et al., SIGGRAPH ASIA 2016)

图像补全 **一键修图！**
(image credit: Hays and Efros)

计算机的优势：感知可以利用大量的冗余

一个房子也许认不出来...

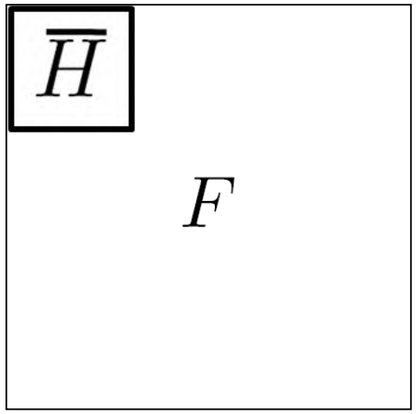
一百个房子认出来了！

Source: S. Lazebnik

统计模型：基于数据分布和统计推断的模型

我们需要统计模型快速合理地进行感知推断

卷积



Adapted from F. Durand

线性滤波

原图

$$* \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

模糊（平均滤波）

Source: D. Lowe

线性滤波

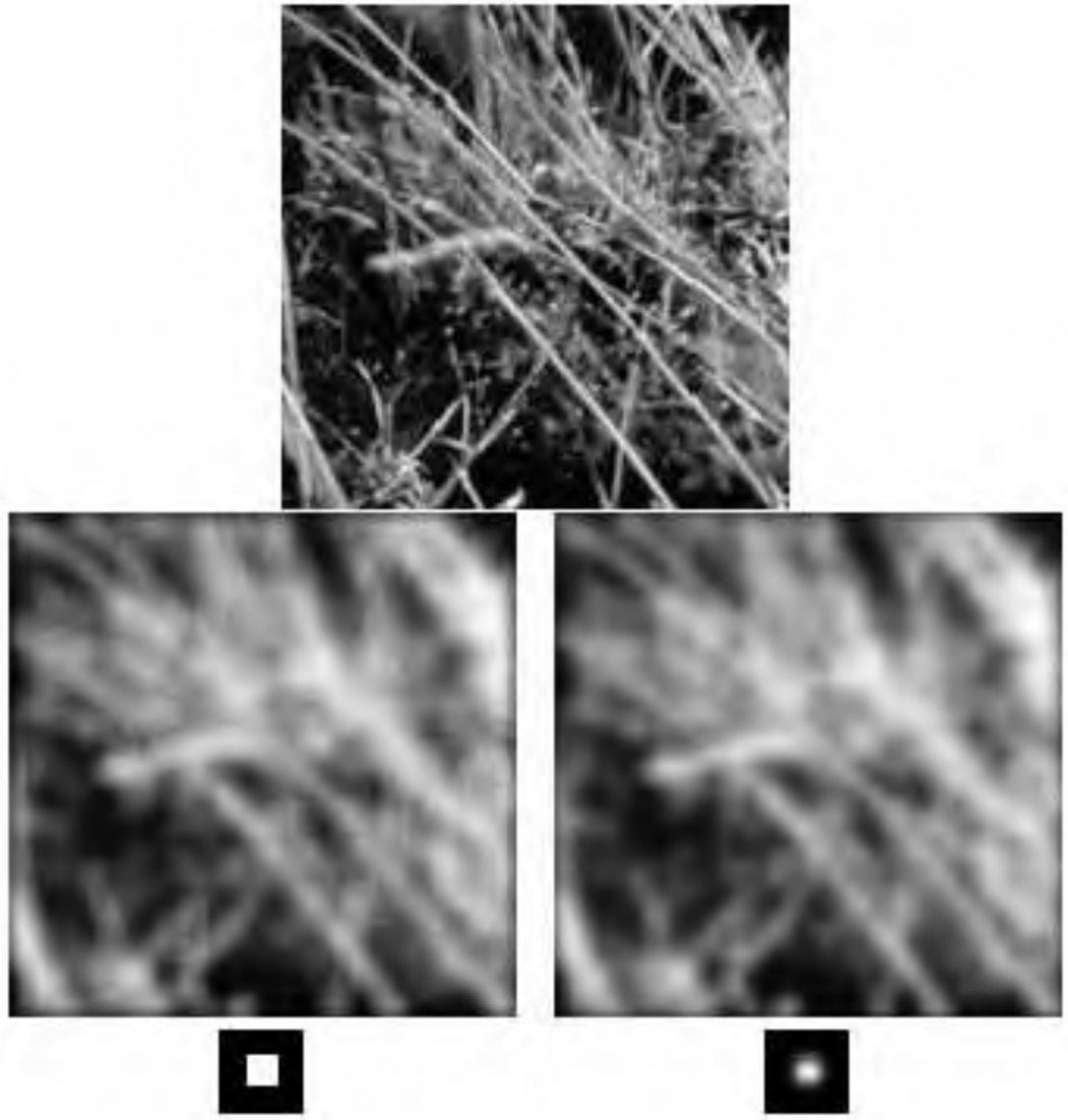
原图

$$*\left(\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix}-\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\right)=$$

锐化后
(突出边缘)

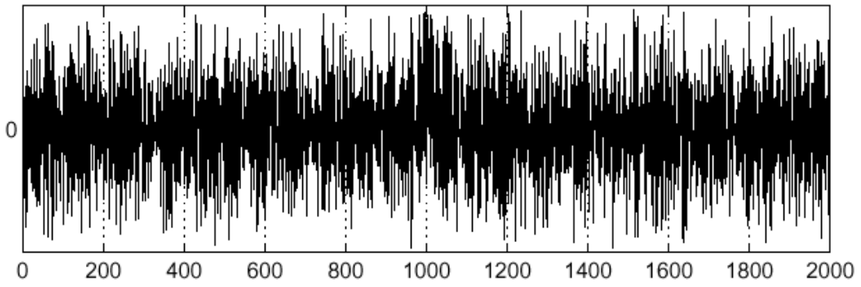
Source: D. Lowe

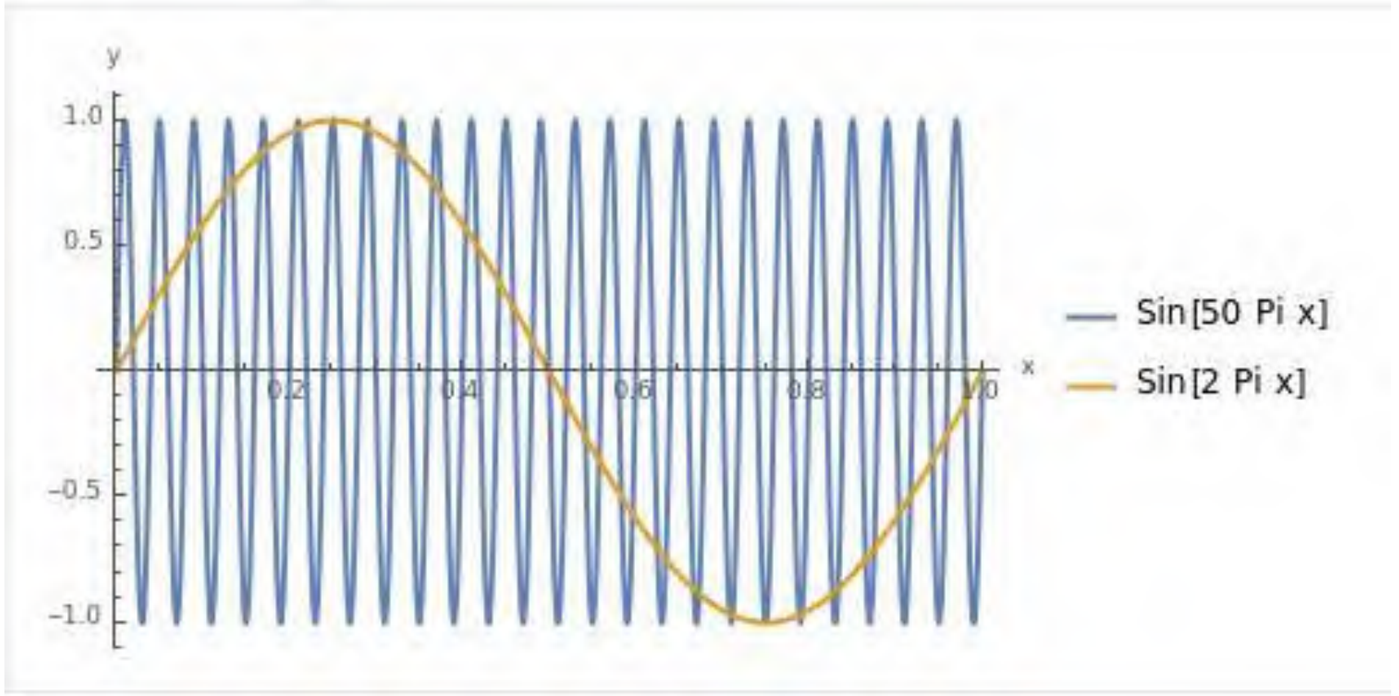
平均滤波 vs 高斯滤波

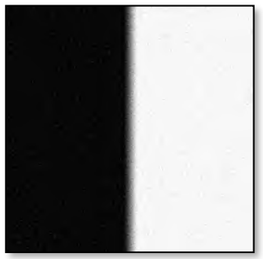


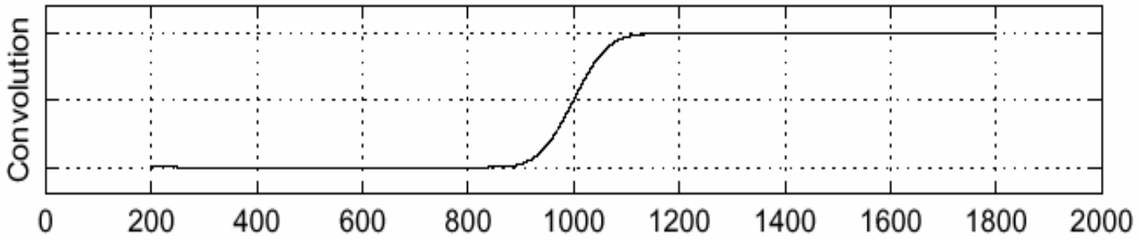
低频信号 vs 高频信号

在图像处理中，高频信号通常代表图像中的细节和边缘，而低频信号则代表图像的基础结构和平滑区域。



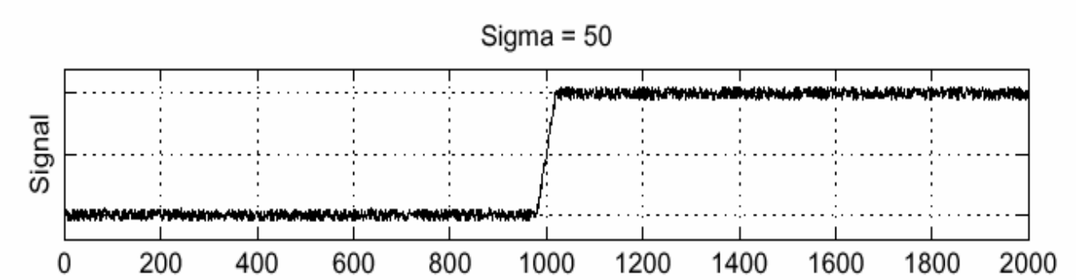




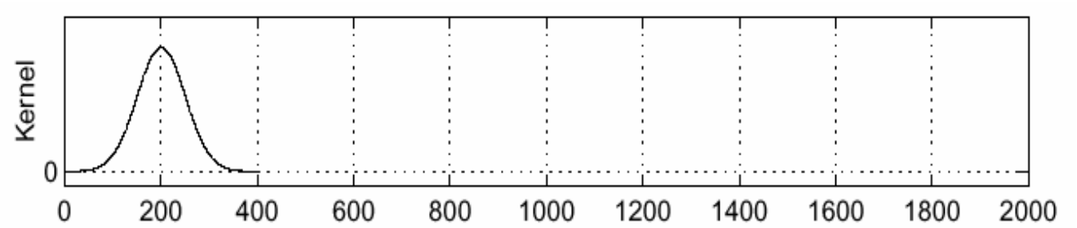


高斯滤波的平滑效果

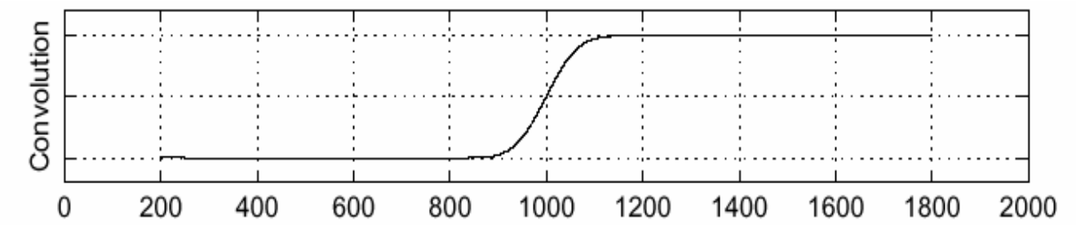
f



h



$f * h$



Source: S. Seitz

锐化

- 模糊带走了什么?

(“细节提取”：高通滤波 ***high-pass filter***)

把细节加回来:

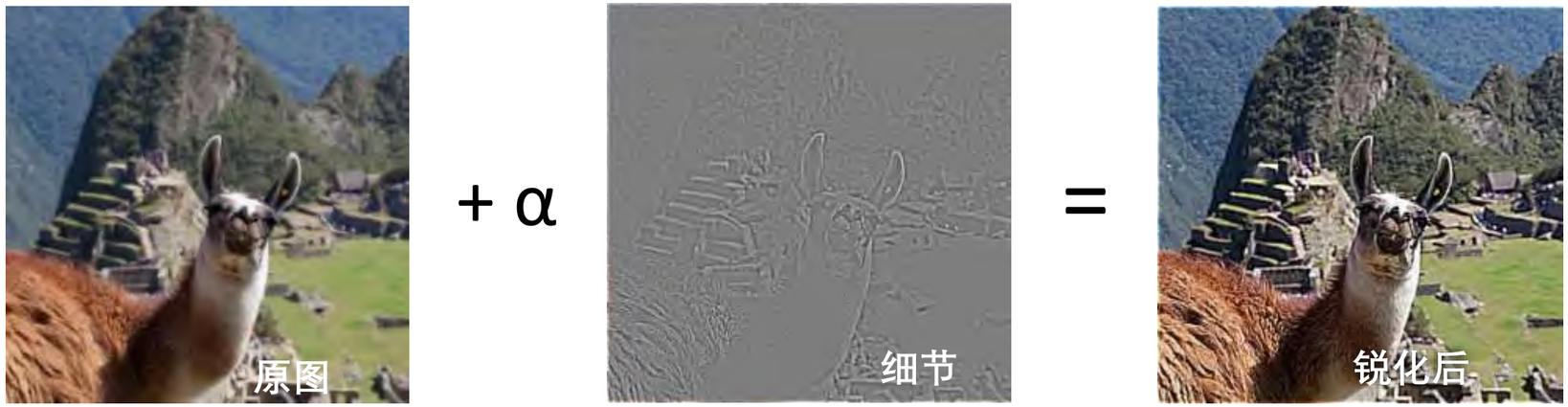
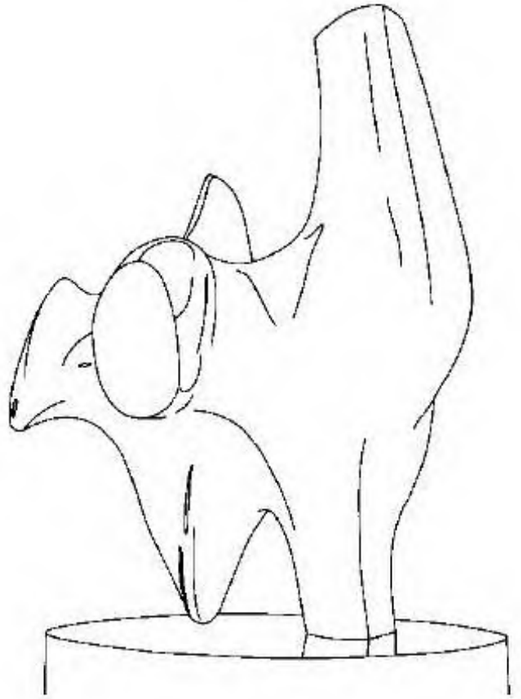


Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

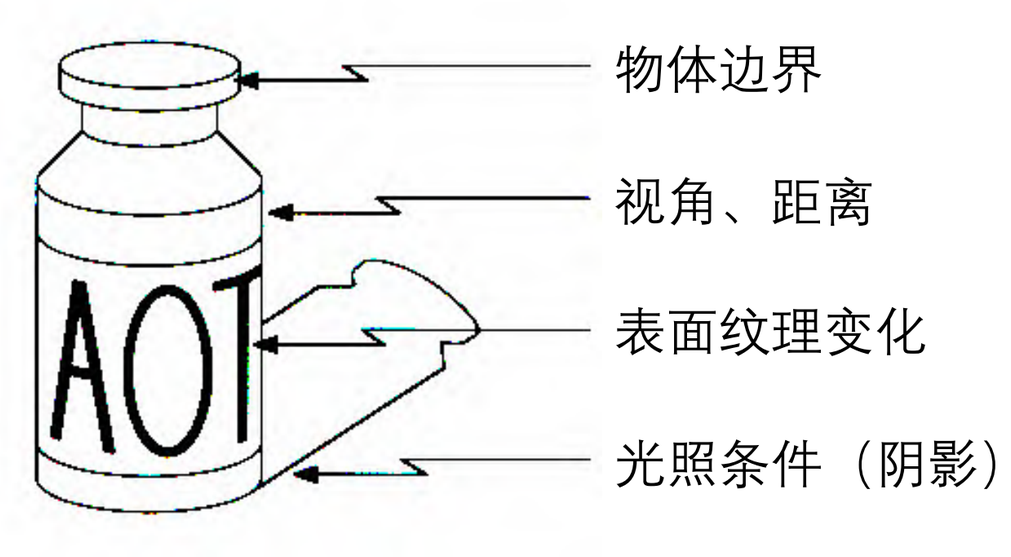
边缘检测

Edge detection 边缘检测



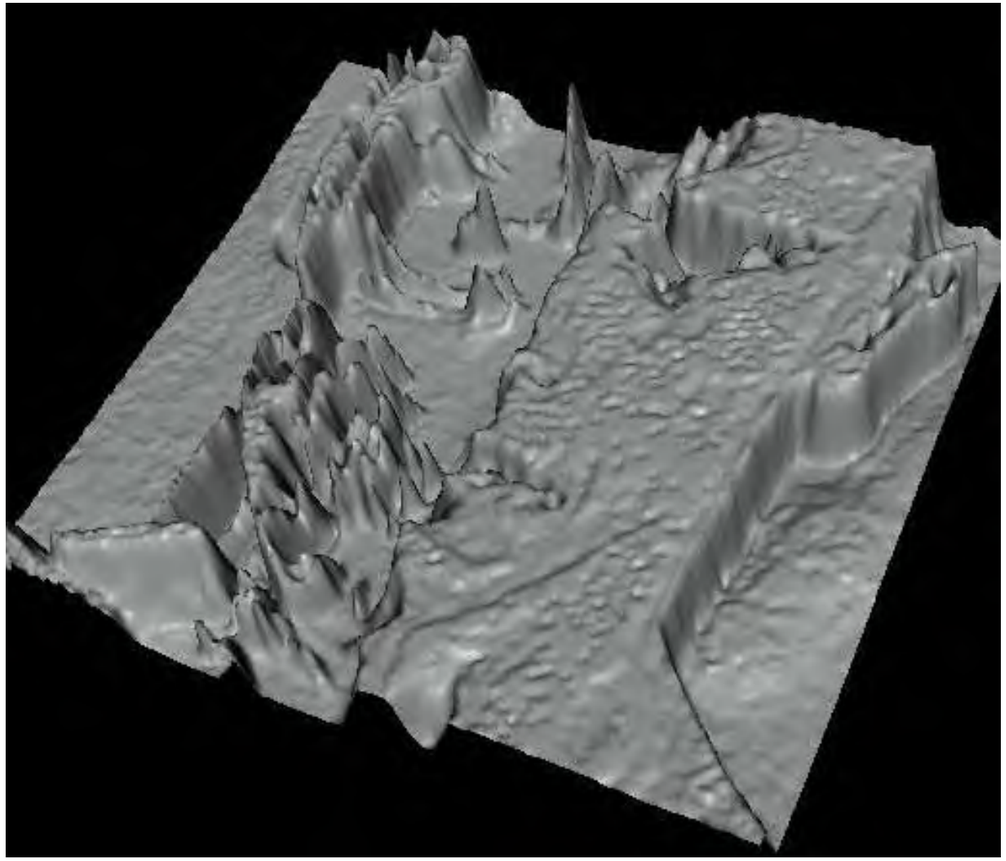
- 把2维图像转成曲线的集合
  - 可以提取场景中的显著特征
  - 比像素的信息量更大（简图）

什么是边缘？



- 很多因素可以导致边缘的出现

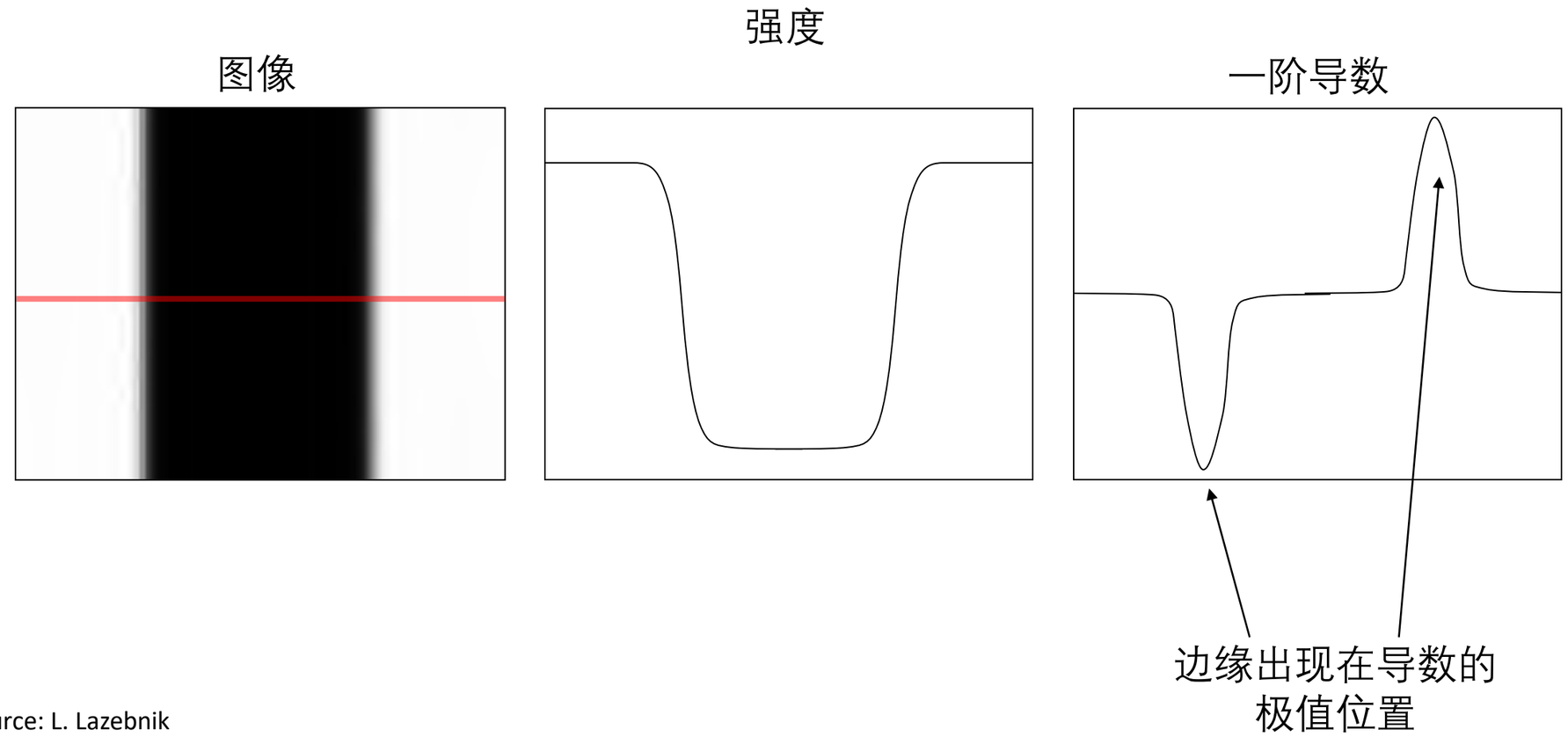
回顾：图像可以认为是一个像素位置到像素值的映射函数



- 边缘可以认为是那些“悬崖”

定义边缘

- 边缘时图像强度发生突变的区域



Source: L. Lazebnik

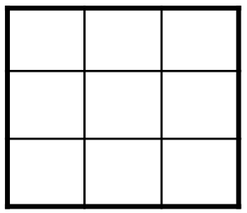
Image derivatives： 图像斜率

- 如何计算图像F[x,y]的“斜率”?
  - Option 1: 把F转换为连续函数 *f*
  - Option 2: 做离散梯度discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

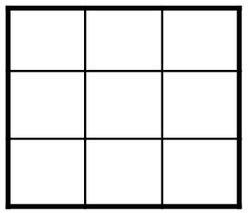
可以用卷积实现

$\frac{\partial f}{\partial x}$:



H_x

$\frac{\partial f}{\partial y}$:



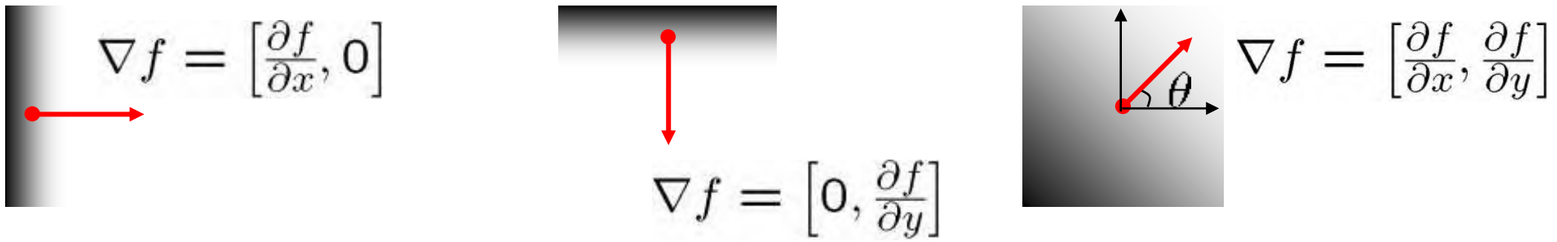
H_y

Source: S. Seitz

Image gradient： 图像梯度

- 图像梯度定义为: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

  这些梯度会指出哪些方向图像会发生剧烈的变化



梯度的强度定义了边缘的强度:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

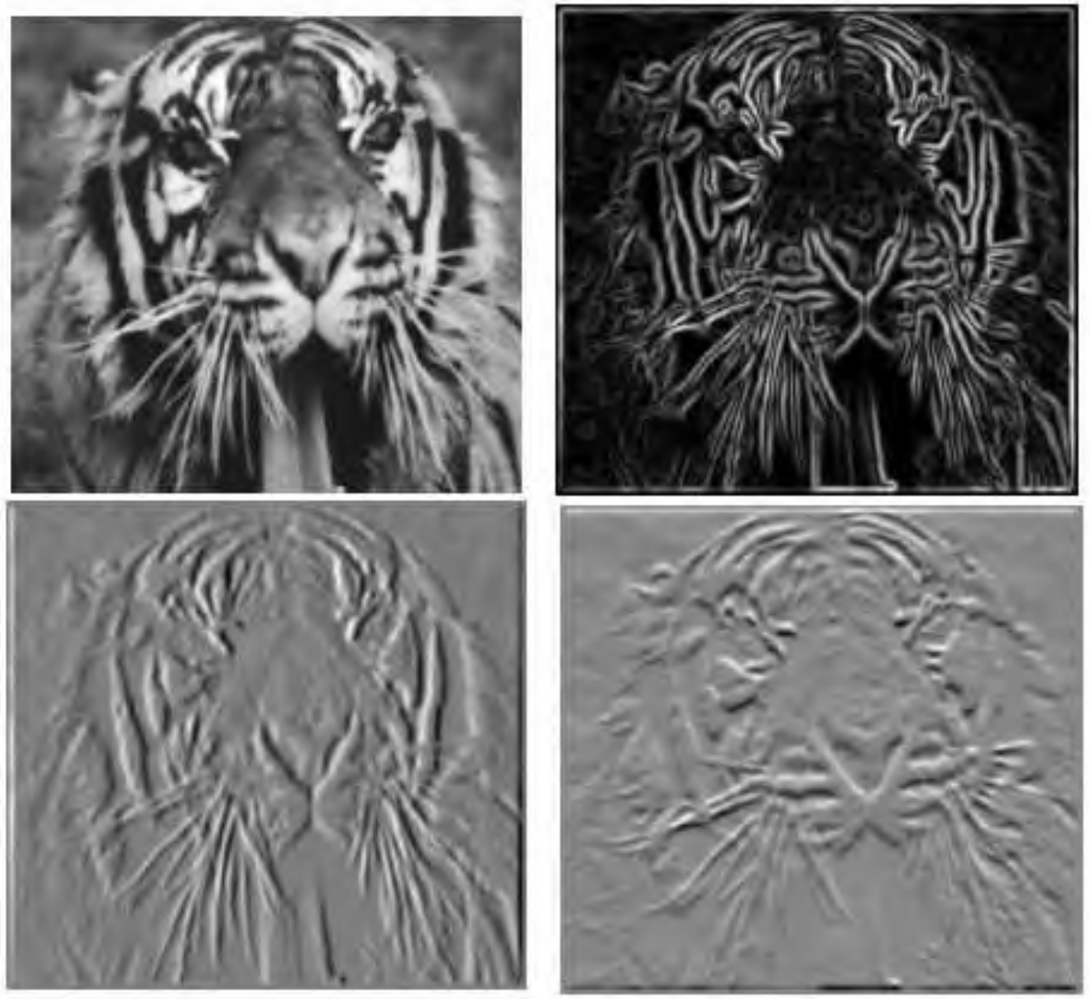
我们可以计算梯度的方向:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- 梯度的方向可以反应边缘的方向，他们是什么关系呢?

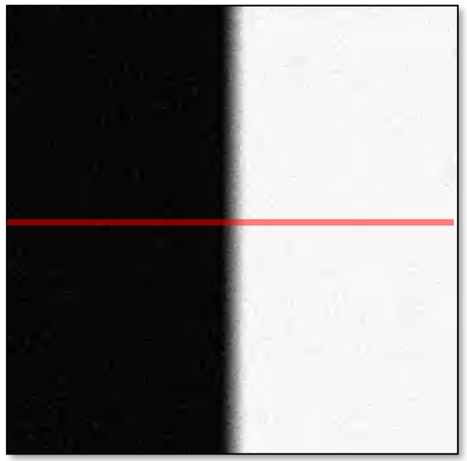
Source: Steve Seitz

图像梯度



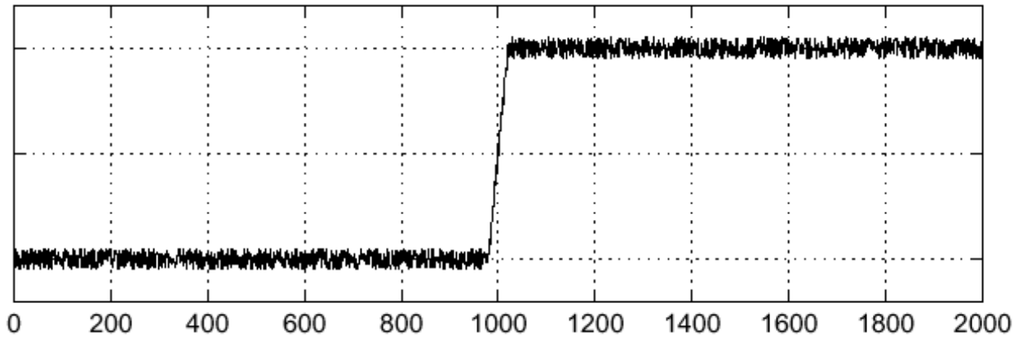
Source: L. Lazebnik

噪声的影响

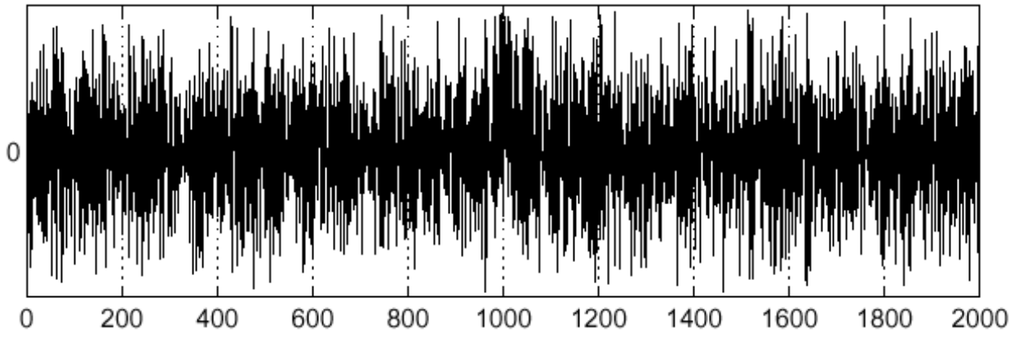


有噪声的图像

$f(x)$



$\frac{d}{dx}f(x)$



哪里是边缘?

Source: S. Seitz

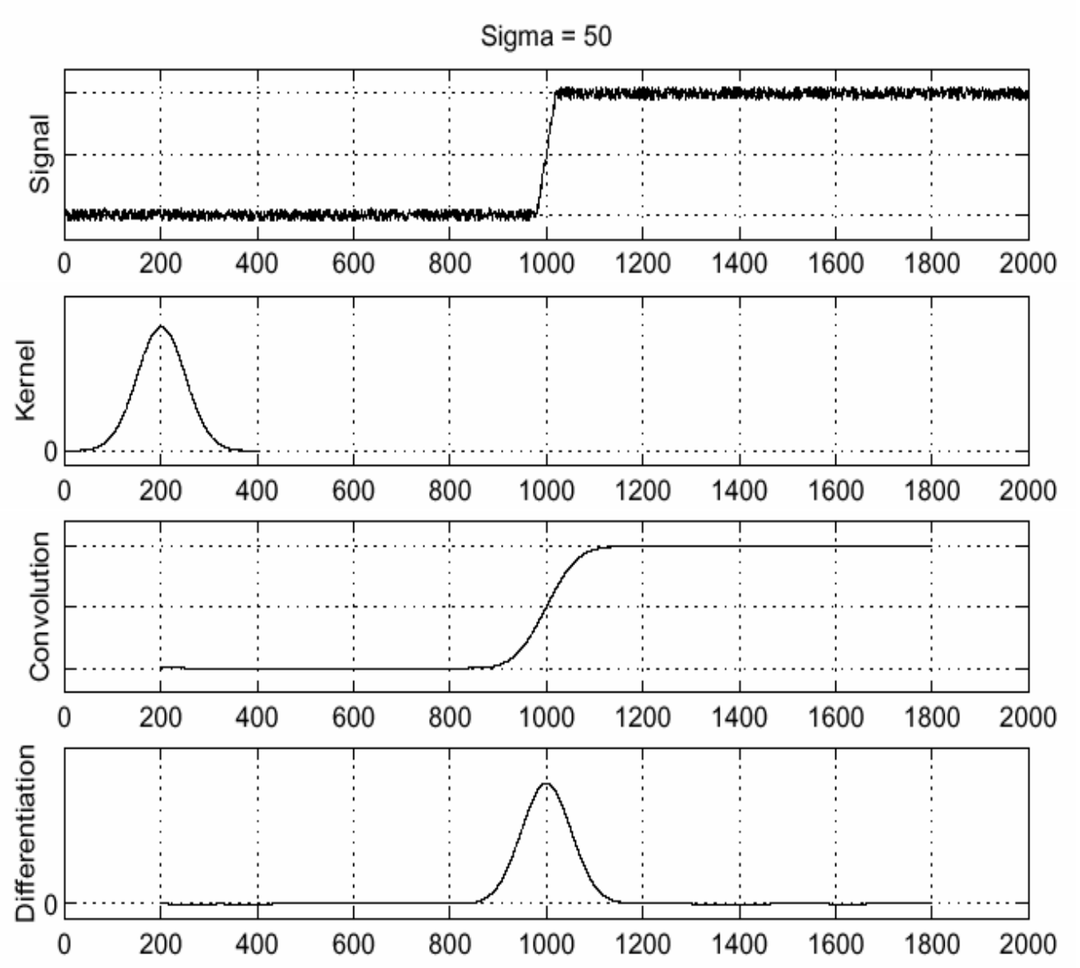
解决方案：高斯模糊

f

h

$f * h$

$\frac{d}{dx}(f * h)$



如果要找边缘，找峰值

$\frac{d}{dx}(f * h)$

Source: S. Seitz

Image with Edge

Edge Location

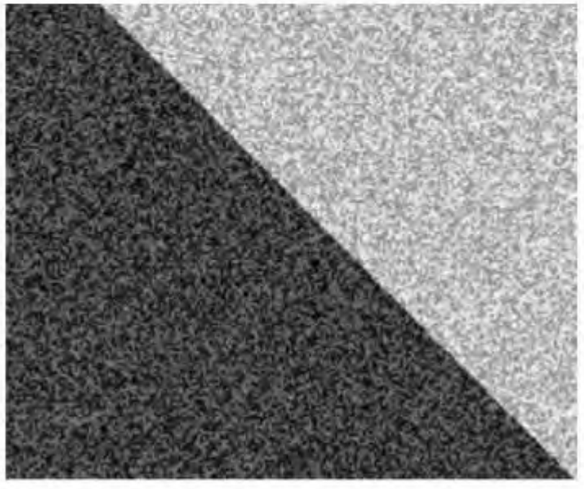
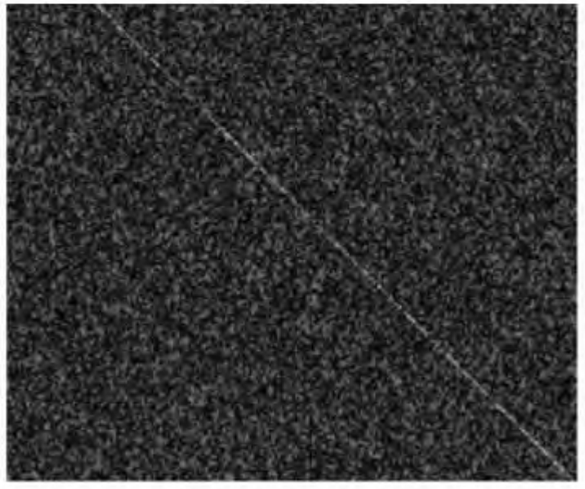
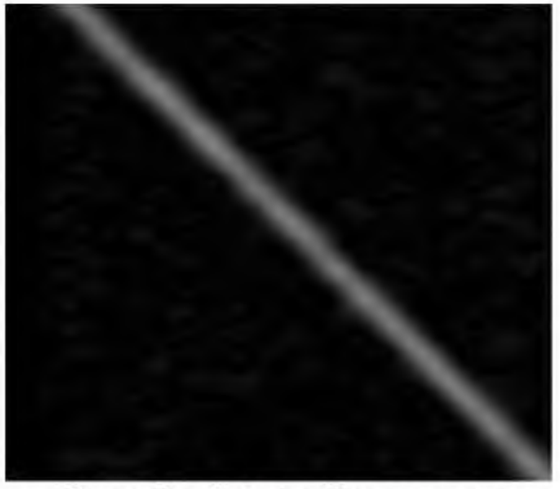


Image + Noise



Derivatives detect
edge *and* noise



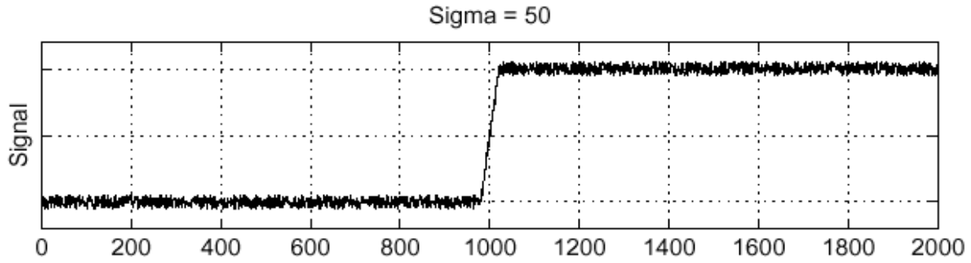
Smoothed derivative removes
noise, but blurs edge

卷积是符合交换律的

- 计算梯度可以用卷积实现，而卷积有交换律: $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$

- 我们可以把计算梯度和模糊结合在一起·
  - Derivative of Gaussian （DoG）

f

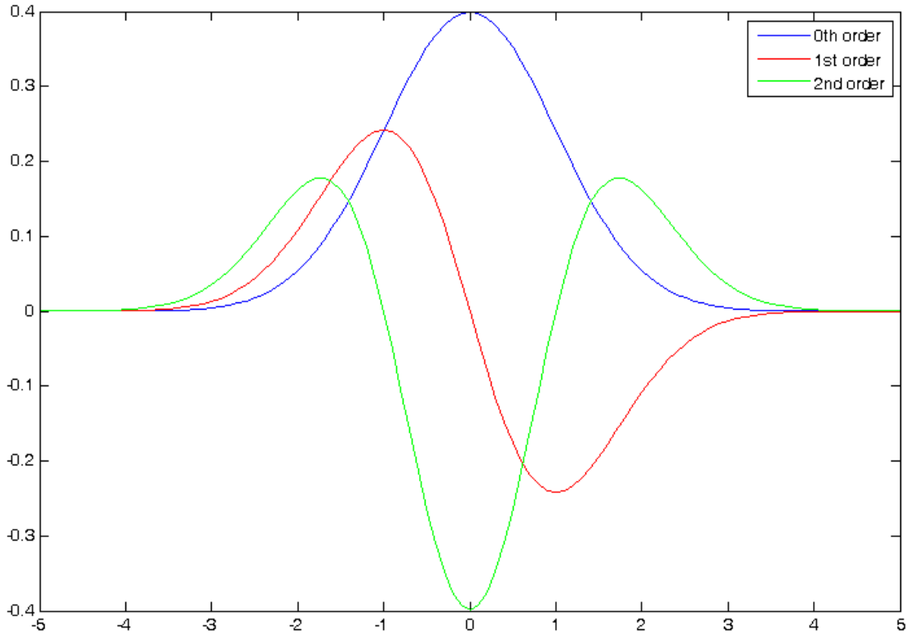


Source: S. Seitz

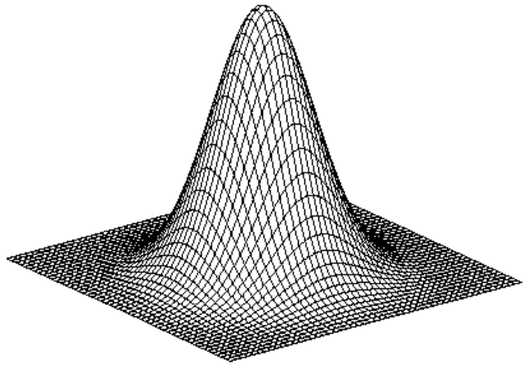
一维高斯函数和DoG

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma}\left(\frac{x}{\sigma}\right) G_\sigma(x)$$

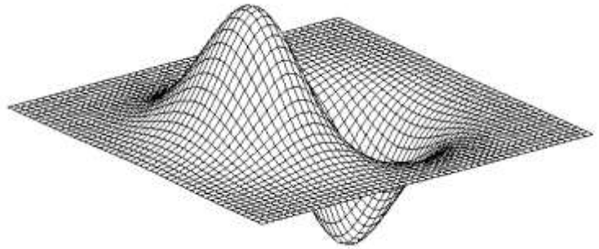


二维高斯函数和DoG



Gaussian

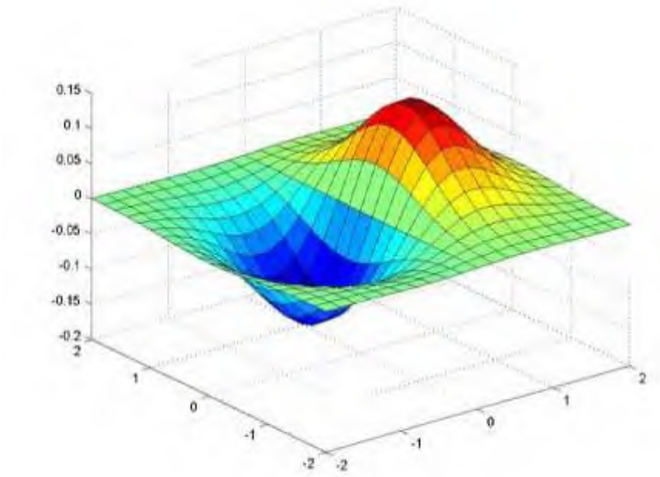
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



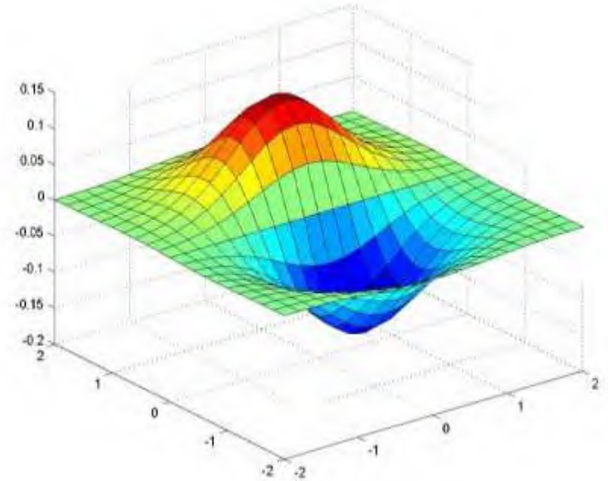
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

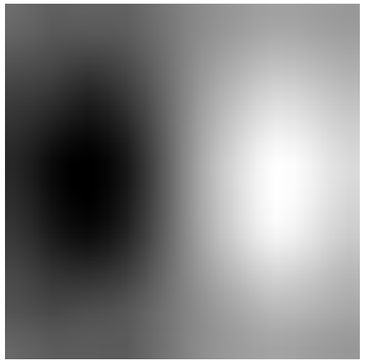
不同方向的DoG

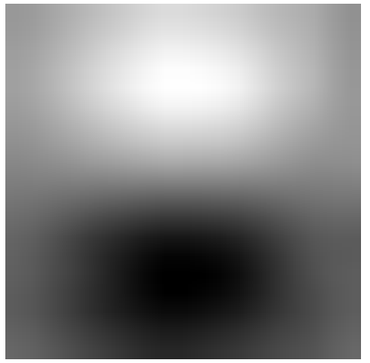


x-direction



y-direction





Sobel 算子

- 经常用来替换DoG

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

s_x

$\frac{1}{8}$

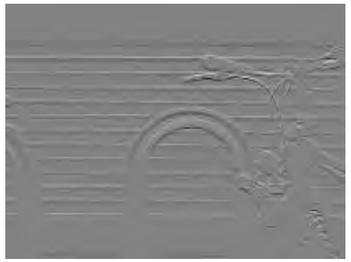
1	2	1
0	0	0
-1	-2	-1

s_y

- Sobel算子通常比DoG更加计算效率，因为它使用的是简单的3x3卷积核，而DoG通常需要更大的卷积核和更多的计算。
  - 计算简单，速度快。对于基础的边缘检测任务通常足够有效。
  - 对噪声比较敏感。可能无法检测到更复杂或微妙的边缘。

Sobel算子





Source: Wikipedia

效果

原图

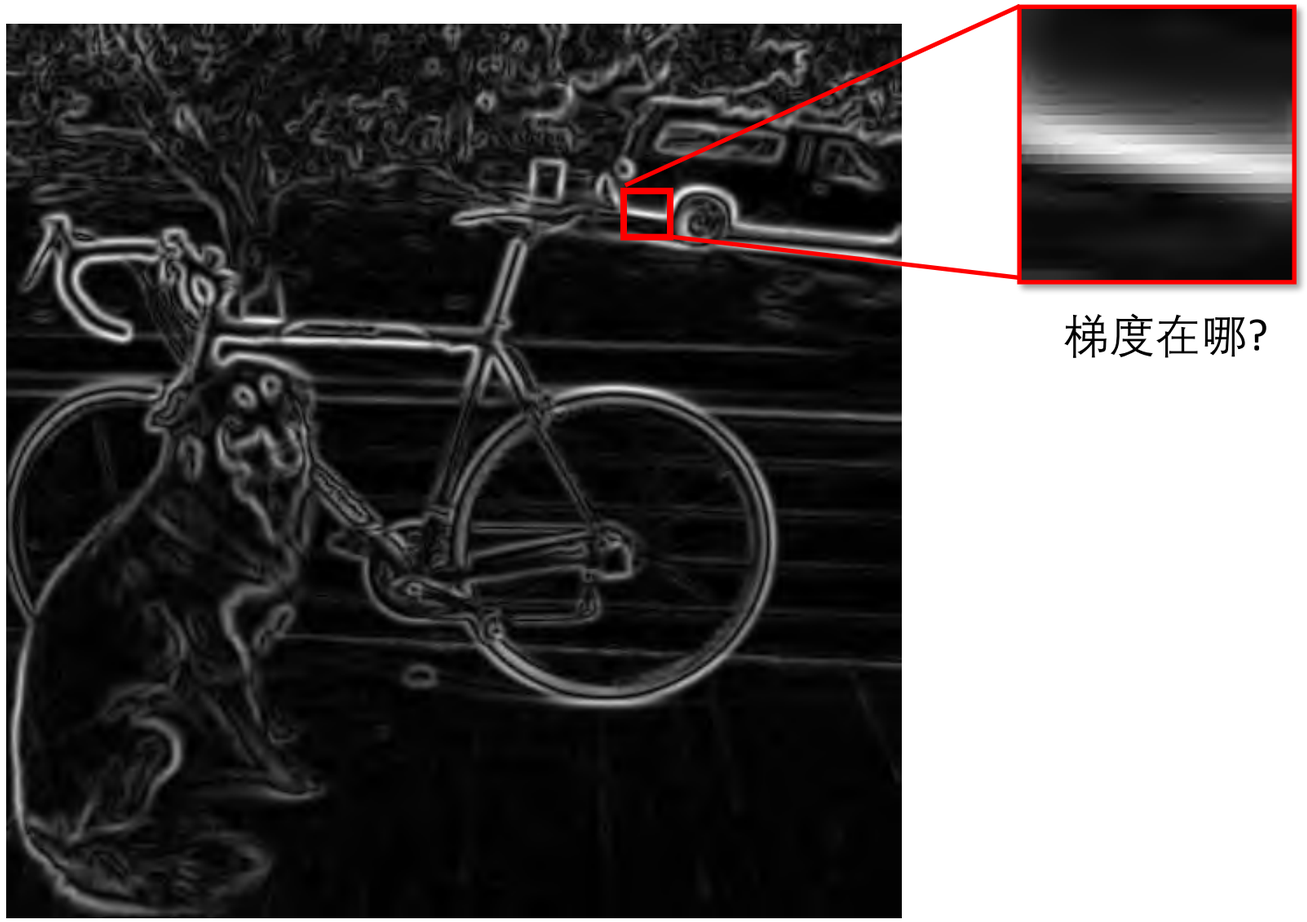
Demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Image credit: Joseph Redmon

找出边缘

平滑梯度幅度

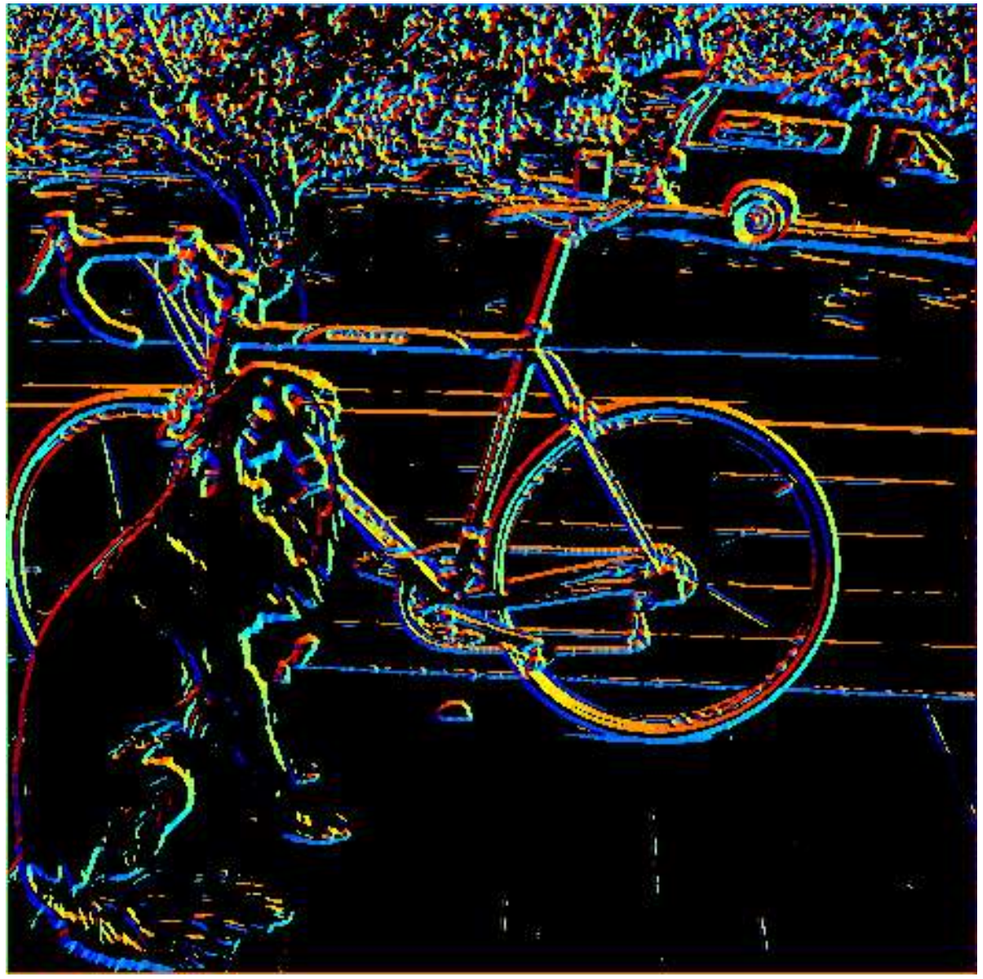
找到梯度



梯度在哪?

利用阈值

找到每个像素的方向



theta = atan2(gy, gx)

Gradient orientation angle

Image with Edge

Edge Location

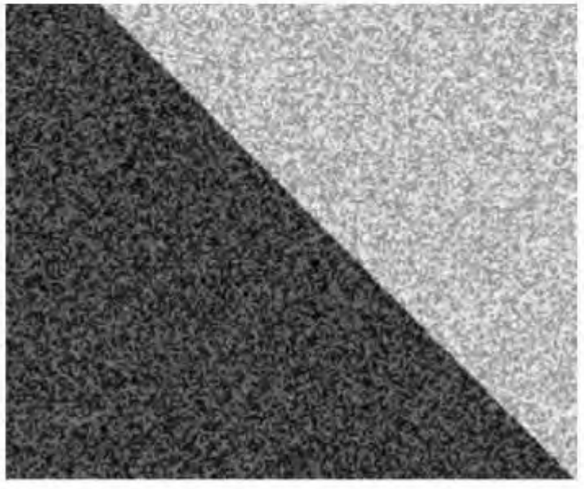
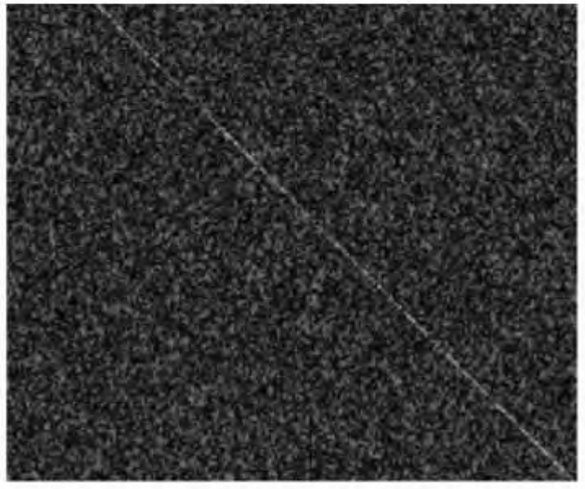
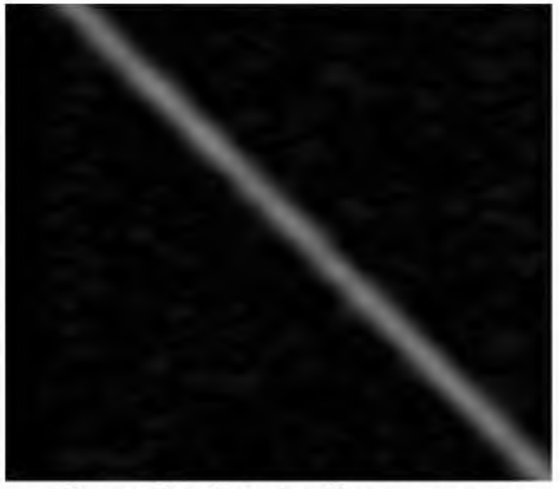


Image + Noise



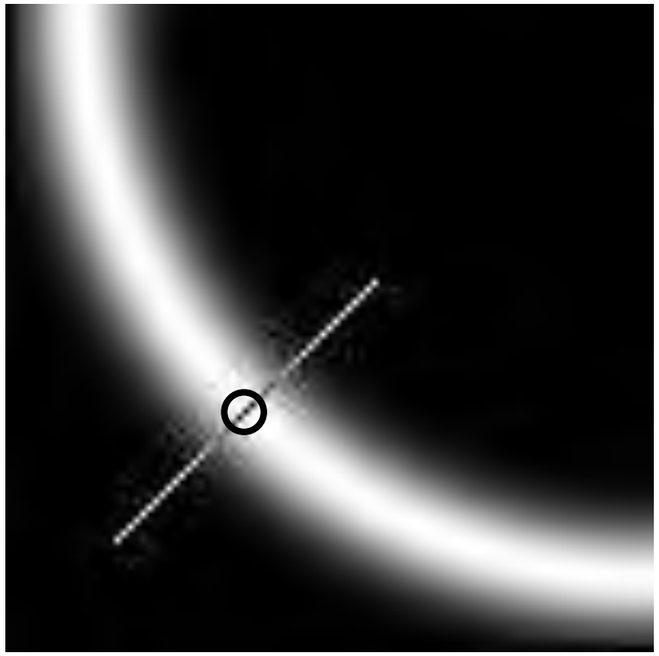
Derivatives detect
edge *and* noise

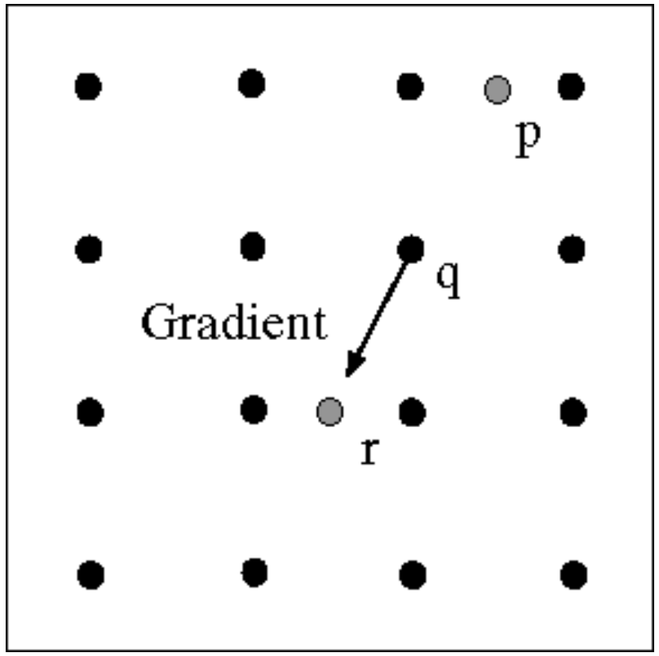


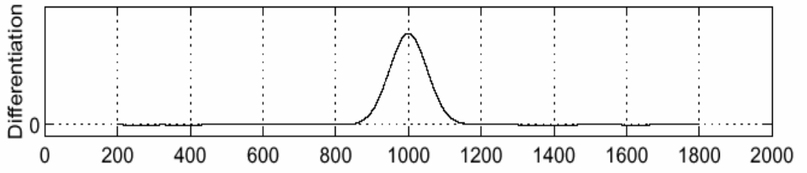
Smoothed derivative removes
noise, but blurs edge

Non-maximum suppression: 非极大抑制

将“厚边缘”细化为“细边缘”







- 对于图像中的每一个像素点，确定其梯度的方向，计算该方向子区域的梯度幅度，在该方向上，如果该像素点的梯度幅度不是局部最大值，则将其设置为0。

Before Non-max Suppression

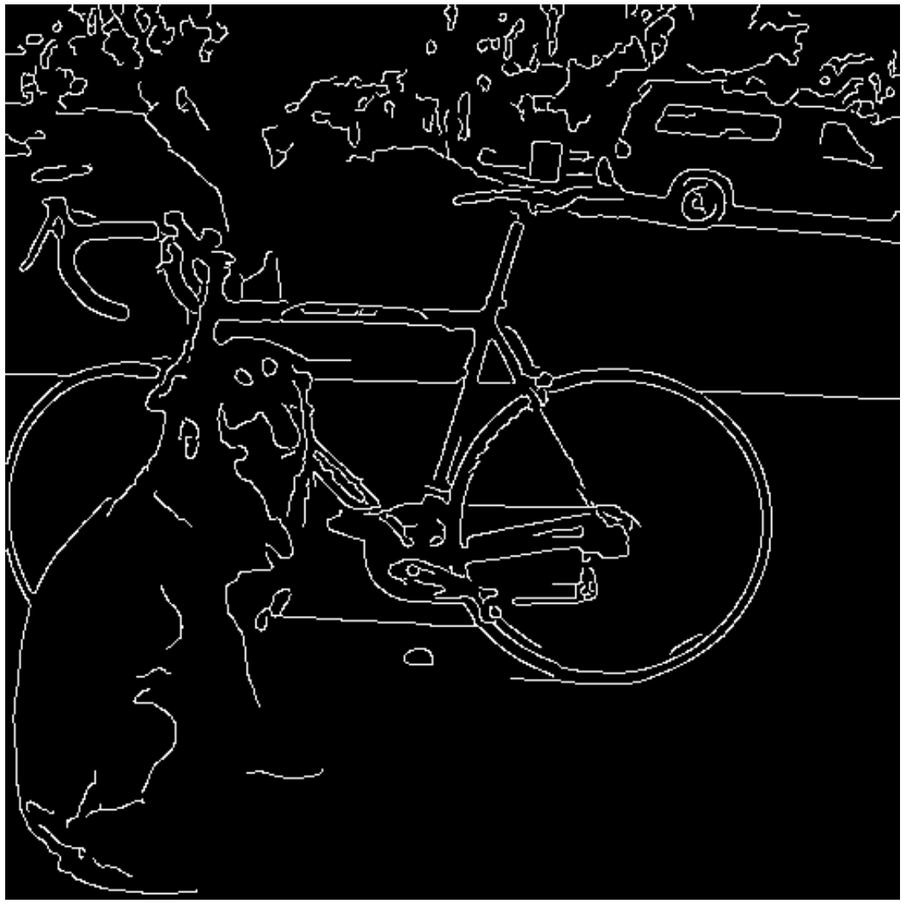
After Non-max Suppression

通过阈值得到边缘

- 噪声依然存在
- 我们想要一些明显边缘
- 2 个阈值, 3 种情况
  - R > T: 强边缘
  - R < T but R > t: 弱边缘
  - R < t: 非边缘
- 为什么要两个阈值?

连接边缘

- 强边缘一定是边缘
- 如果弱边缘与强边缘联通则也是边缘
- 通过周围八个像素点进行搜索



J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

MATLAB: `edge(image,'canny')`

Canny边缘检测

1. 使用DoG对图像预处理

2. 找到梯度的幅度和方向

3. Non-maximum suppression

4. 连接边缘:
   - 定义高低两个阈值，得到弱边缘和强边缘
   - 用强边缘作为初始边缘，用弱边缘连接强边缘

Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny边缘检测器

original

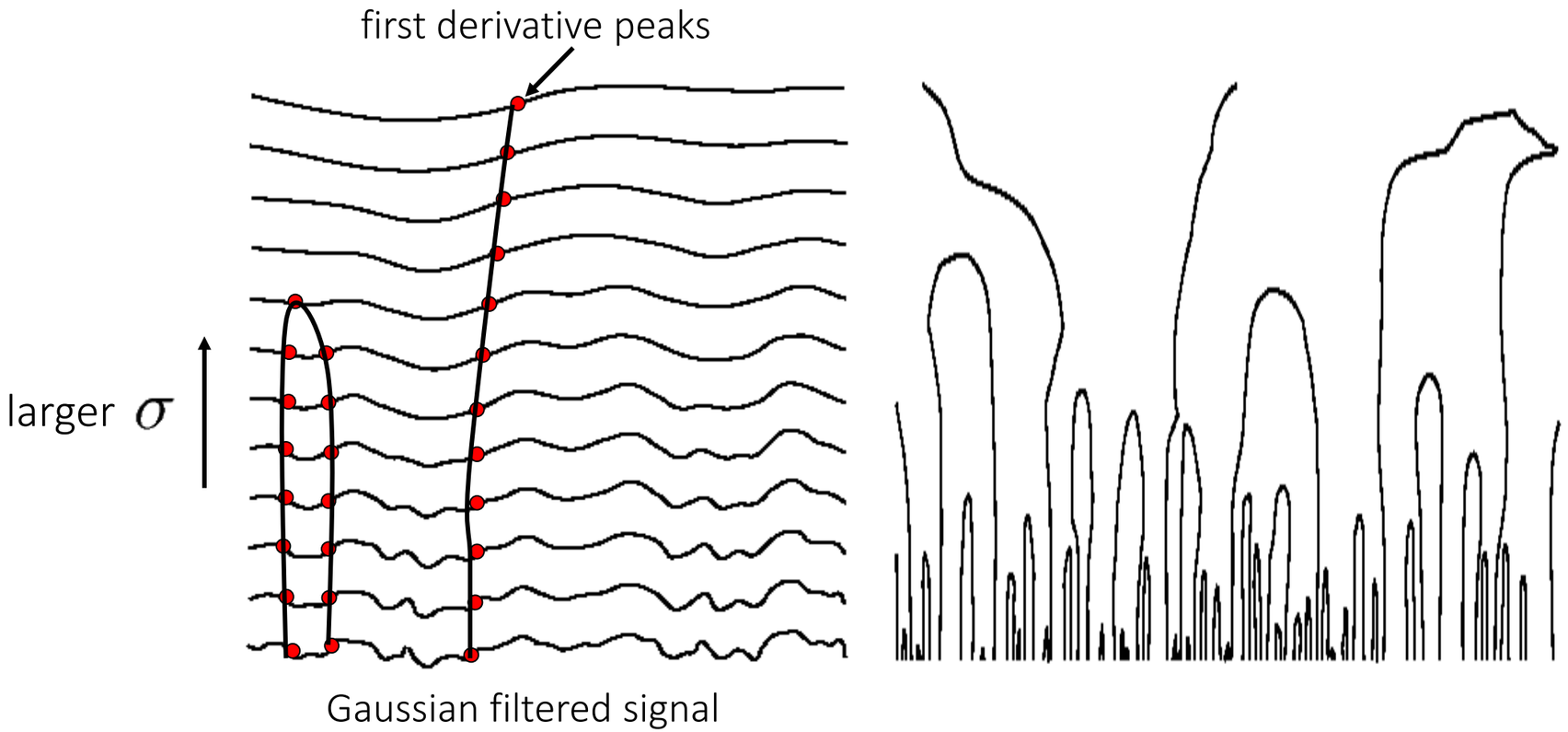
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- 高斯滤波带宽σ影响最终结果
  - 高带宽边缘更“大尺度”
  - 低带宽边缘更“细粒度”

Source: S. Seitz

尺度空间[Witkin 83]

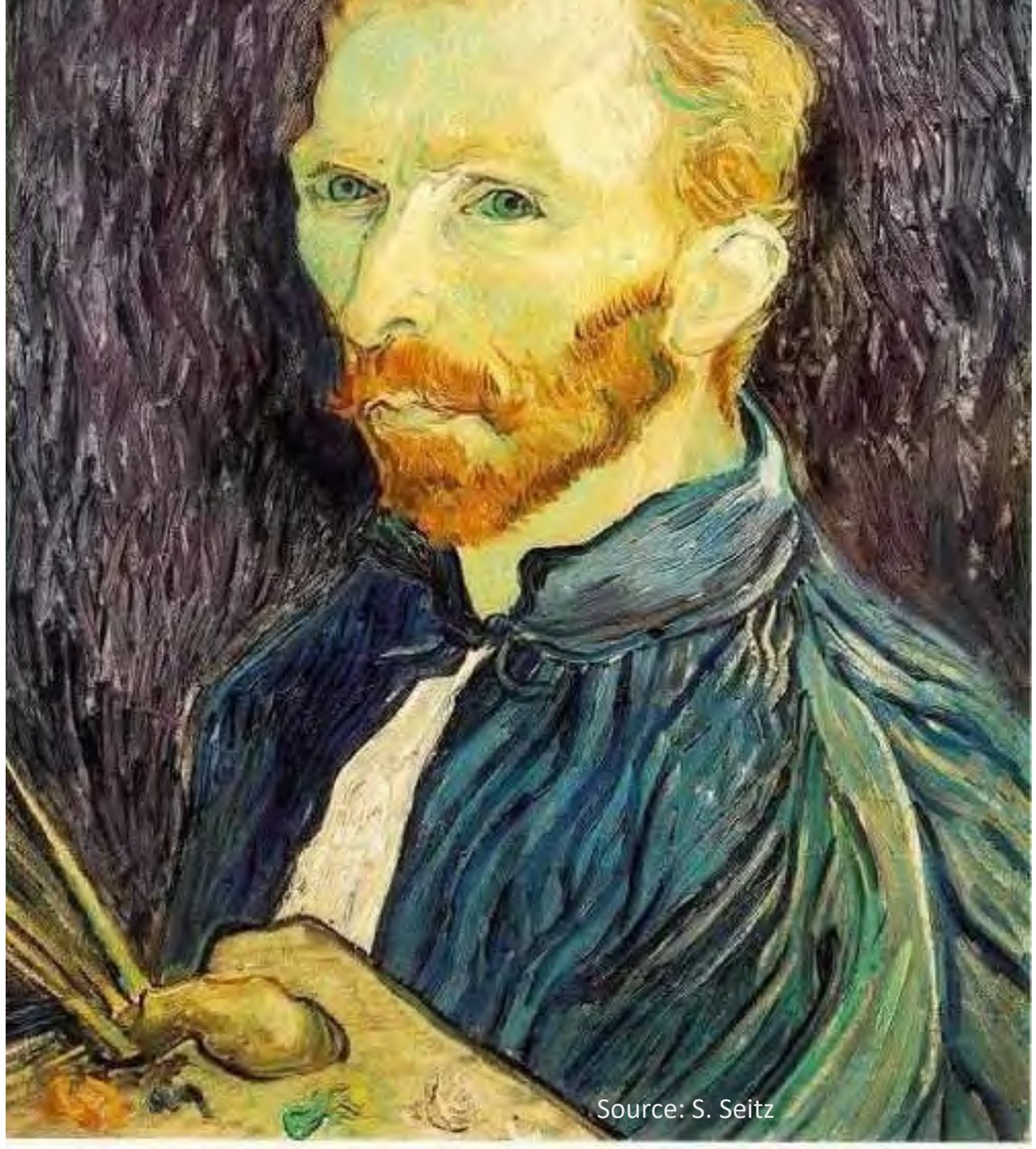


- 更高的尺度下（更大的带宽）：
  - 边缘的位置可能改变
  - 边缘可能合并
  - 但边缘不会再分开

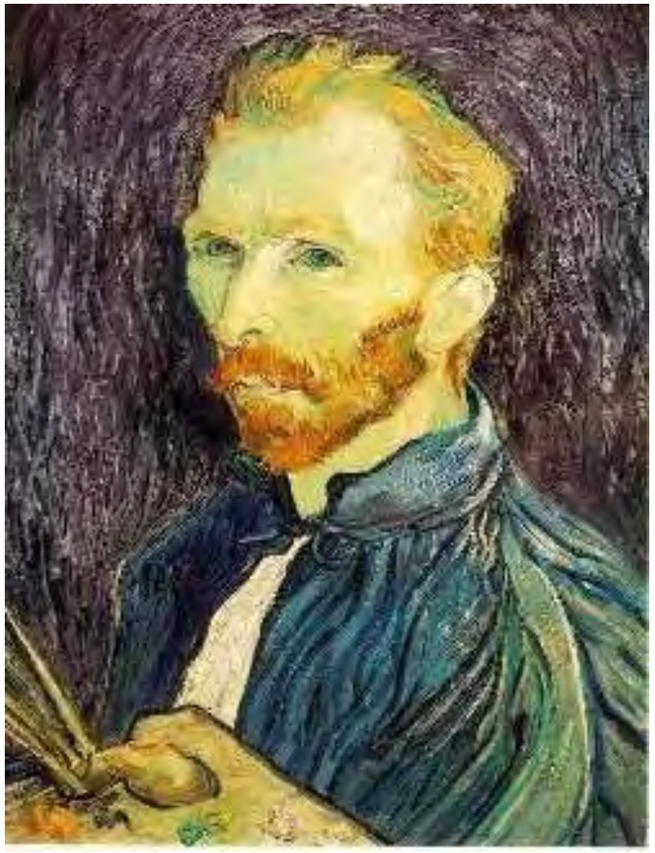
采样与插值

图像尺度

如果图像太大了放不到屏幕里，我们怎样制作一张更小的图像？



Sub-sampling 下采样



1/4

1/8

隔一行隔一列扔掉一半的图像

- 图像下采样 *image sub-sampling*

Source: S. Seitz

图像下采样

1/2

1/4 (2x zoom)

1/8 (4x zoom)

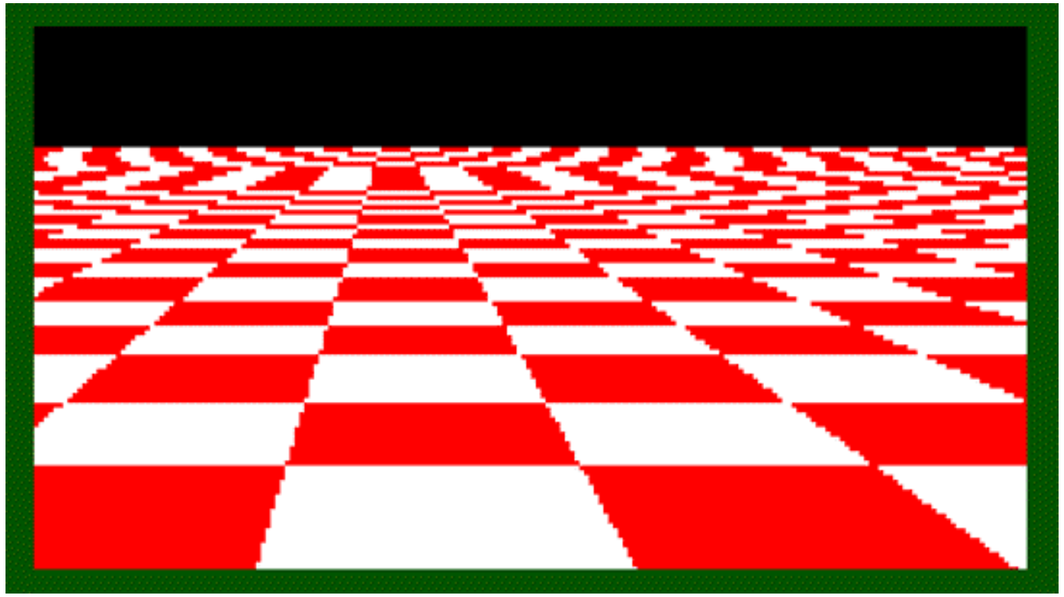
如果再上采样回来，为什么看起来这么粗糙?

Source: S. Seitz

图像下采样：变形

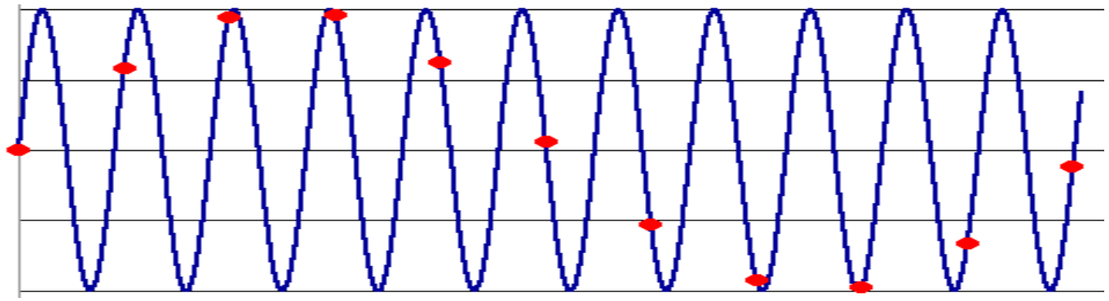
Source: F. Durand

生成图像下采样：变形



Source: L. Zhang

Aliasing：混淆

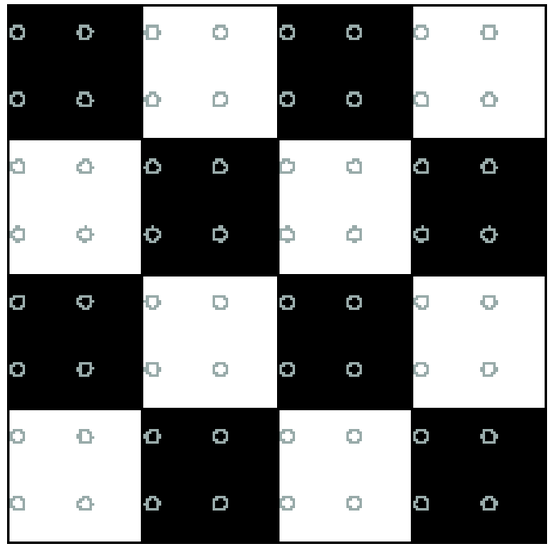


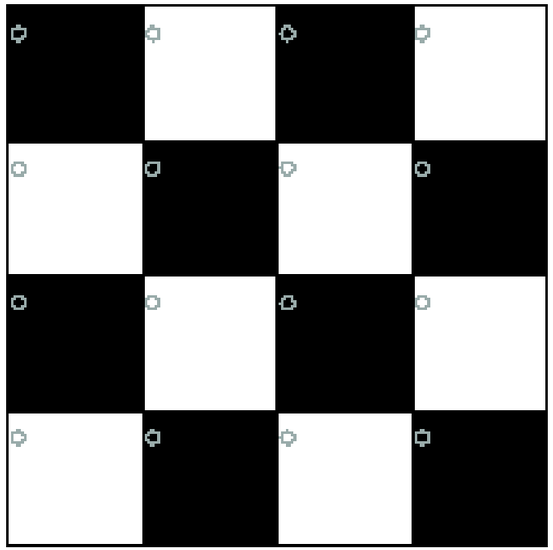
- 当采样率不够高的时候，有可能会导致采样的信息有问题
- 导致得到的信号出现问题（频率变低）导致 *alias 混淆*

- 如何正确地采样？我们需要正确地理解信号、图像的结构
- 傅里叶变换的相关知识
  - “But what is the Fourier Transform? A visual introduction.” https://www.youtube.com/watch?v=spUNpyF58BY
- 为了避免混淆:
  - 采样率要不小于图像最大频率的2倍
    - 也就是说，每个周期至少两个采样
  - **最小采样率也被称为Nyquist rate**

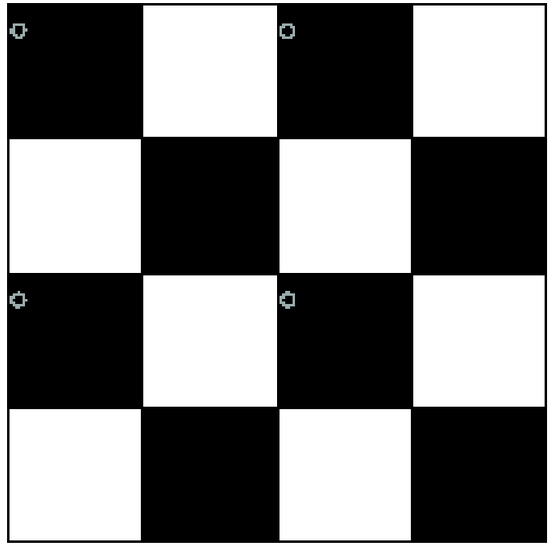
Source: L. Zhang

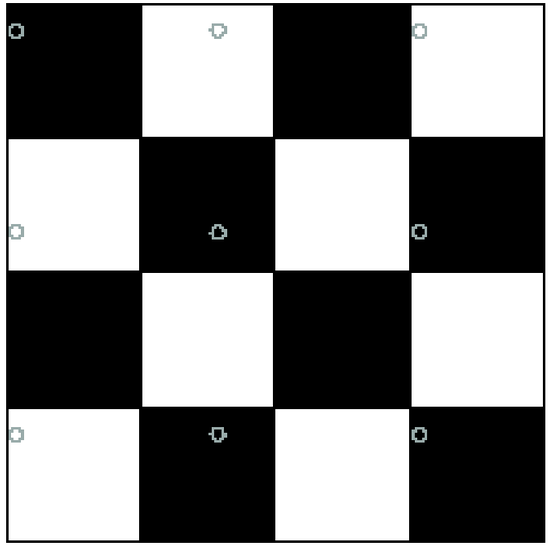
Nyquist limit – 2D的例子





好的采样





坏的采样

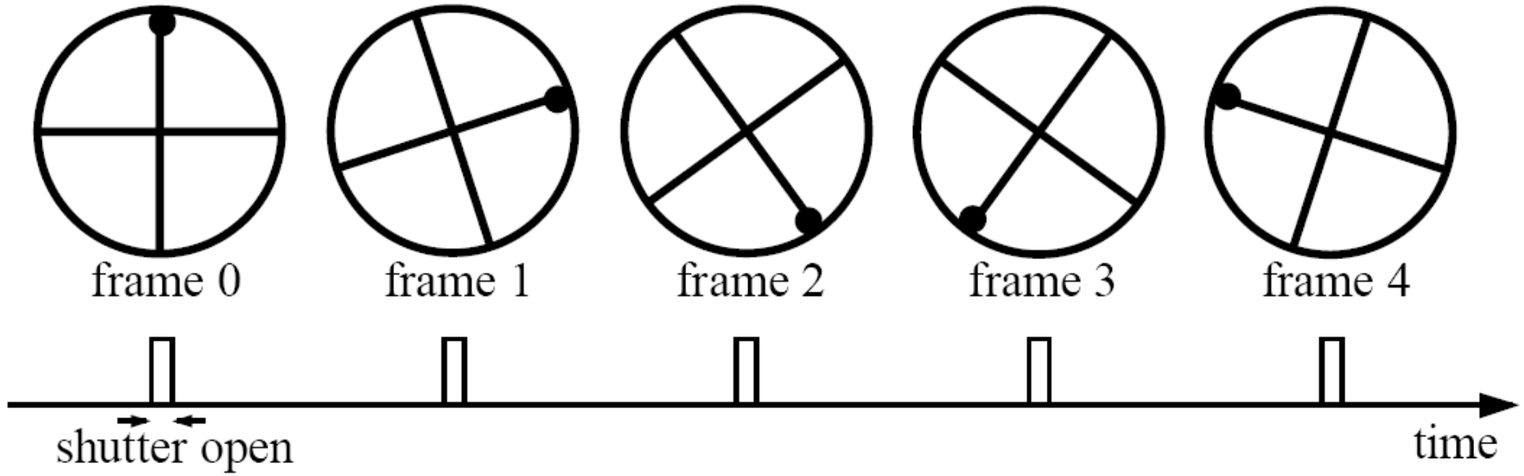
Wagon-wheel effect

https://en.wikipedia.org/wiki/Wagon-wheel_effect

Wagon-wheel effect

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

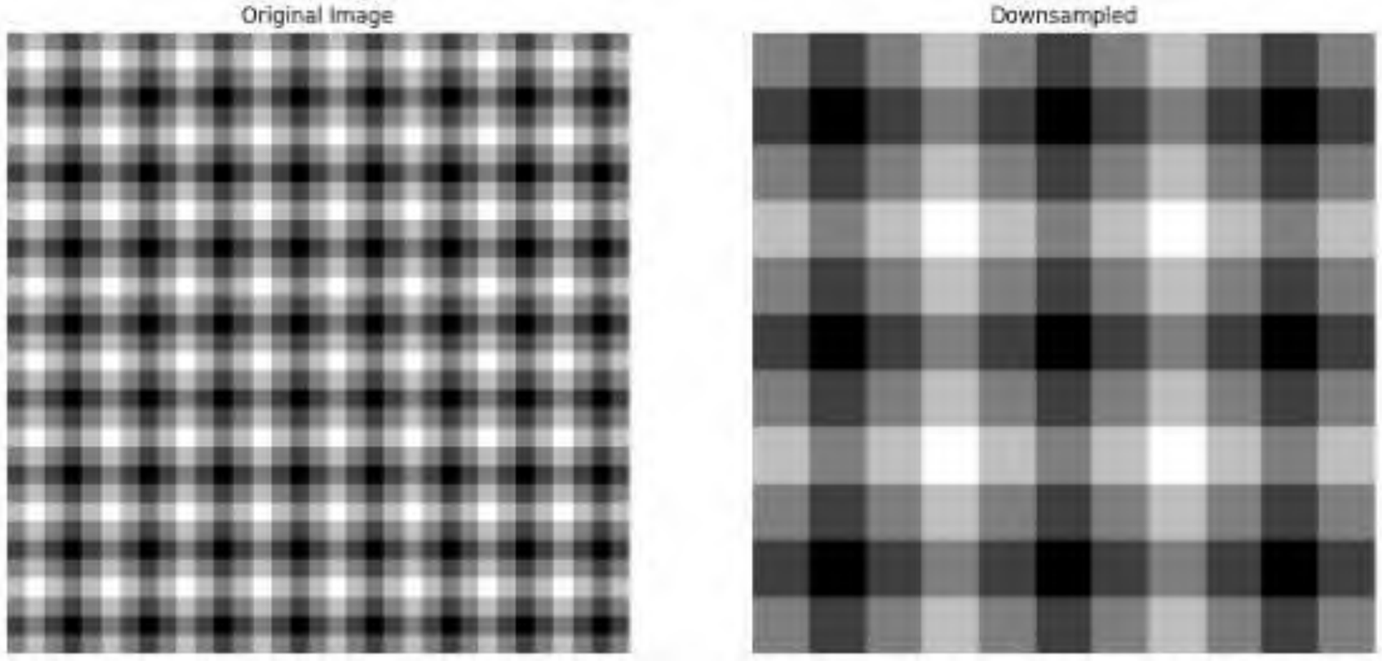
(See http://www.michaelbach.de/ot/mot-wagonWheel/index.html)

Source: L. Zhang

Aliasing 混淆

- 当下采样的比例大于2的时候
  - 原来的像中的有些内容频率可能过高，导致出现混淆

- 怎么解决这个问题?



高斯模糊预处理

Gaussian 1/2

G 1/4

G 1/8

- 解决方案：先模糊，再下采样

Source: S. Seitz

使用高斯模糊以后的下采样

Gaussian 1/2

G 1/4

G 1/8

- 解决方案：先模糊，再下采样

Source: S. Seitz

如果不这么做…

1/2

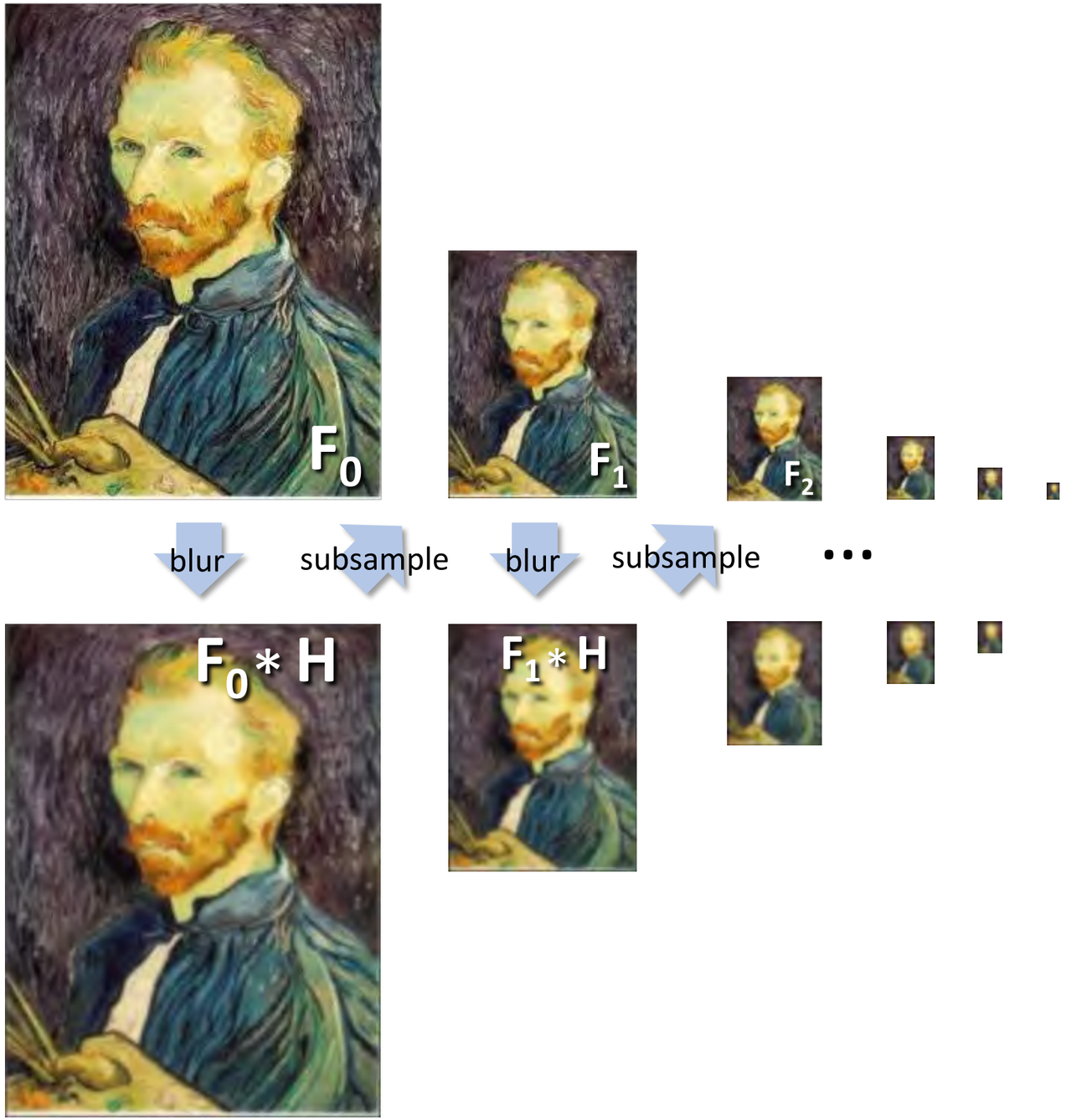
1/4 (2x zoom)

1/8 (4x zoom)

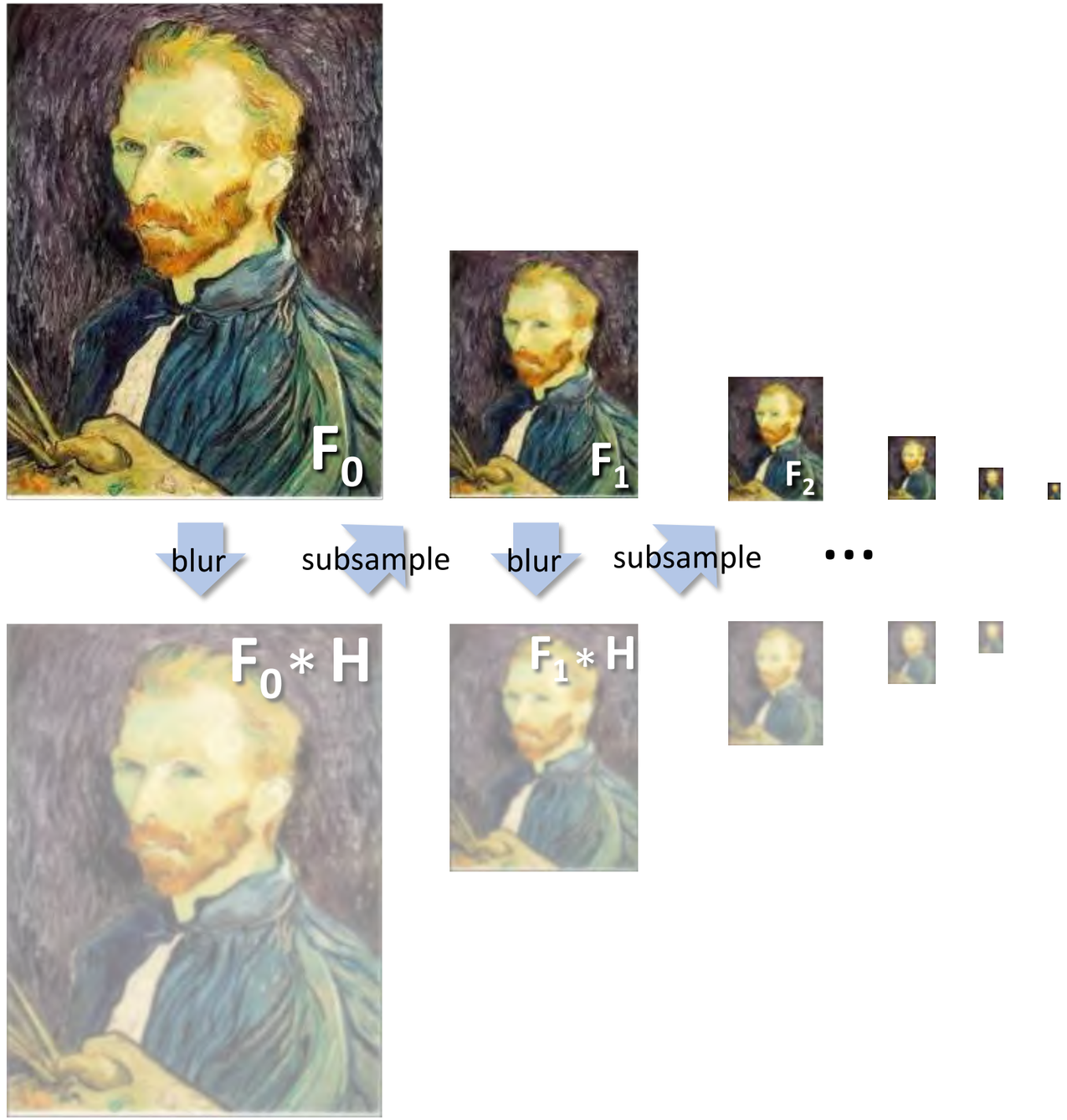
Source: S. Seitz

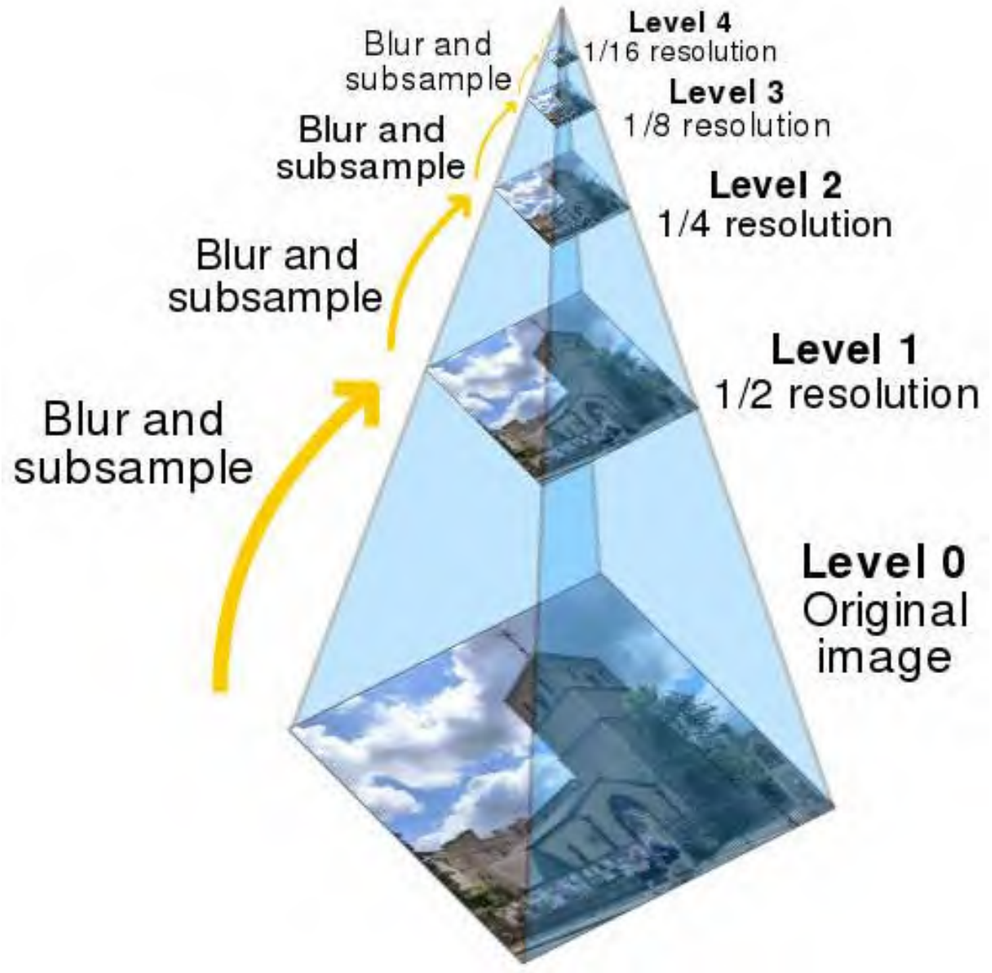
高斯模糊

- 解决方案：先模糊，再下采样

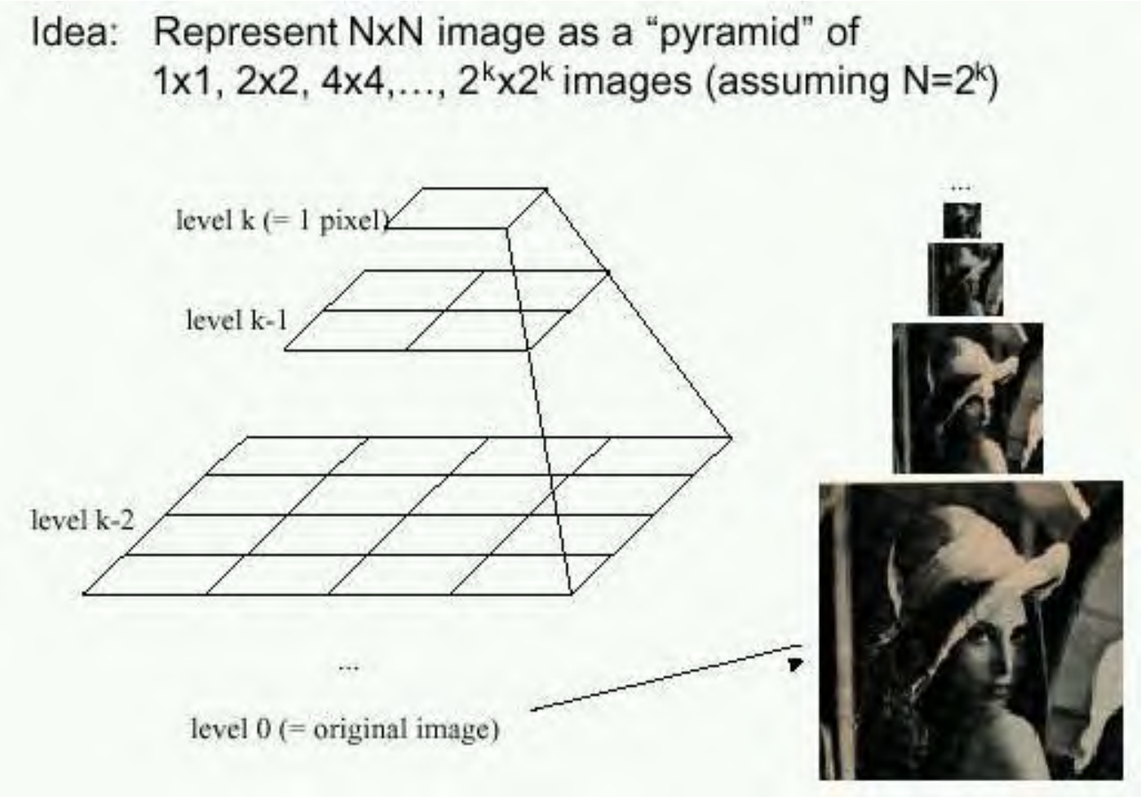


Gaussian pyramid
高斯金字塔




Blur and subsample
Blur and subsample
Blur and subsample
Blur and subsample
Level 4
1/16 resolution
Level 3
1/8 resolution
Level 2
1/4 resolution
Level 1
1/2 resolution
Level 0
Original image

Gaussian pyramids [Burt and Adelson, 1983]

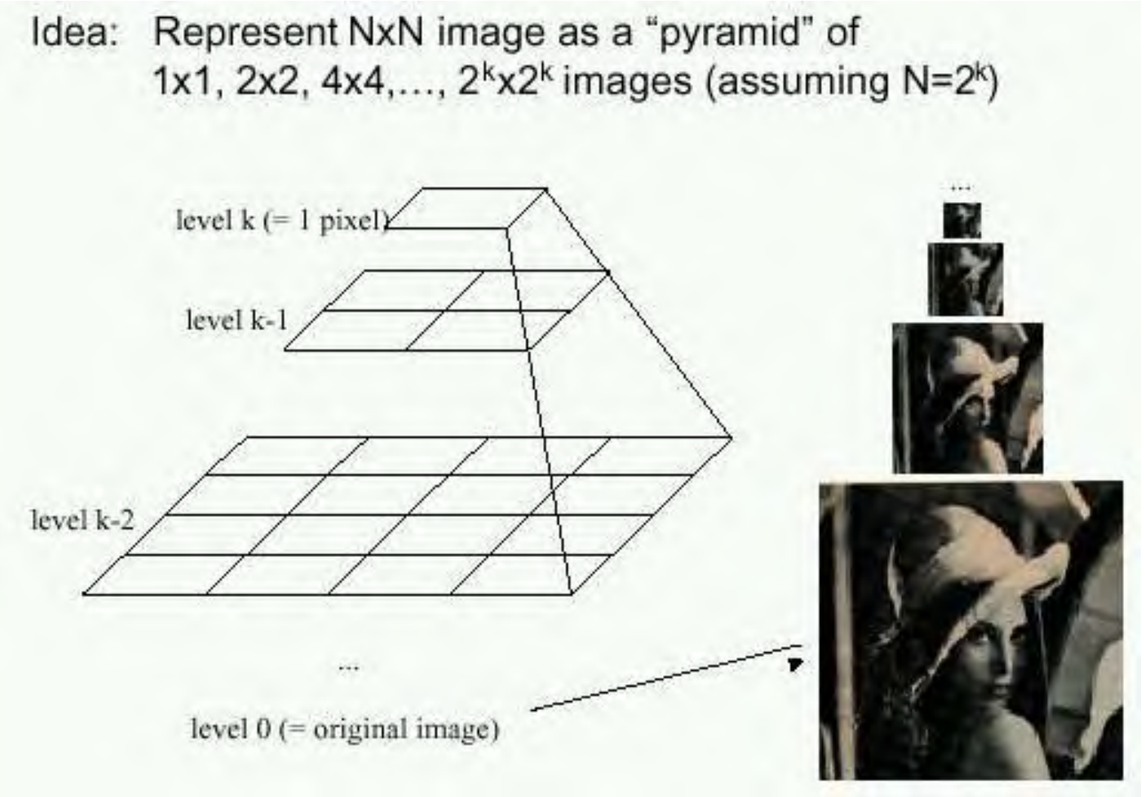


- *mip map* [Williams, 1983] 纹理映射和反走样的技术。它包含了一系列预先计算和存储的图像，每个图像都是原始纹理的一个下采样版本。
- *wavelet transform 在不同频率成分和不同位置的数学变换。*

我们未来还会再遇到这个金字塔结构

Source: S. Seitz

Gaussian pyramids [Burt and Adelson, 1983]

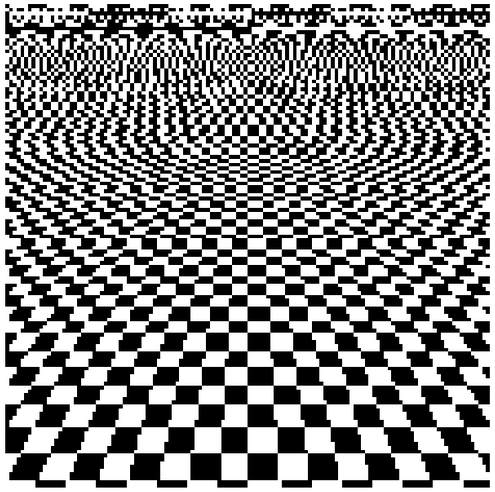


- 高斯金字塔相比原来多占用了多少空间?

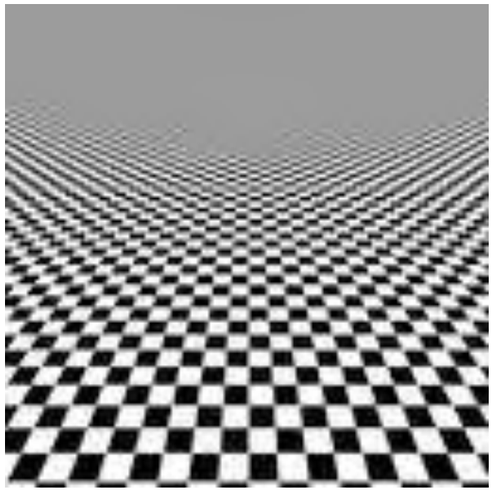
Source: S. Seitz

Gaussian pyramid

使用高斯金字塔的效果



Naïve subsampling

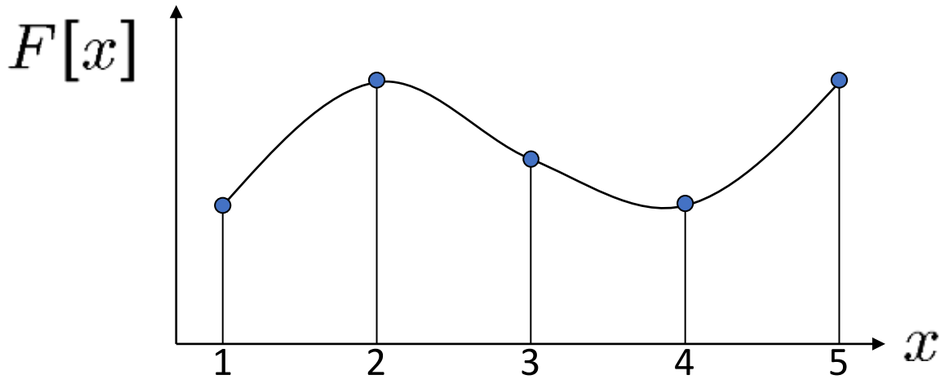


Proper prefiltering
(“antialiasing”)

Upsampling 上采样

- 如果图像太小了:
- 怎么把它放大十倍?
- 最简单的方法:
  重复每行每列
  十次
- (“Nearest neighbor interpolation”)
  最近邻插值

图像插值



d = 1

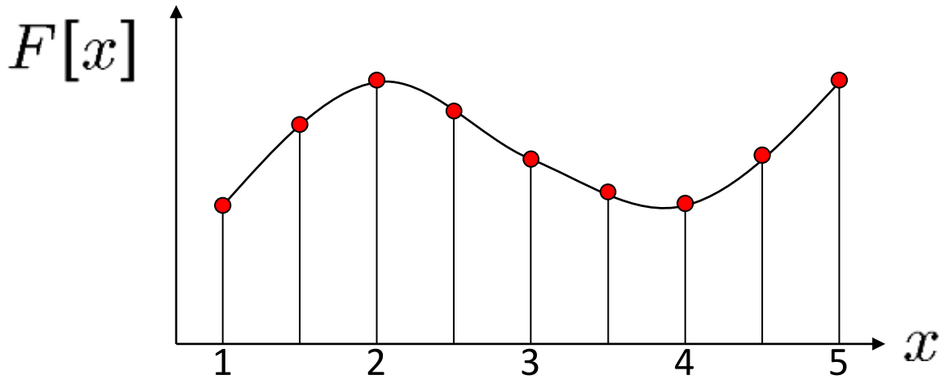
数字图像可以认为是连续信号的“量化”:

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- 每一个点都是连续信号上的离散采样
- 我们是否可以用采样的值还原整个函数?

Adapted from: S. Seitz

图像插值



d = 0.5

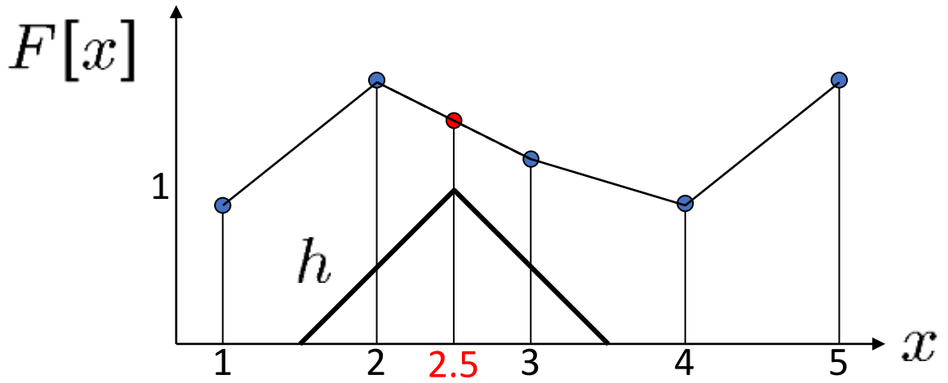
数字图像可以认为是连续信号的“量化”:

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- 每一个点都是连续信号上的离散采样
- 我们是否可以用采样的值还原整个函数?

Adapted from: S. Seitz

重构滤波器



d = 1 in this example

- 当我们不知道 f 的时候
  - 猜一个近似函数: $\tilde{f}$
  - 依然可以用滤波的形式来做
  - 把 F 转换为连续函数，保留整数点的值:

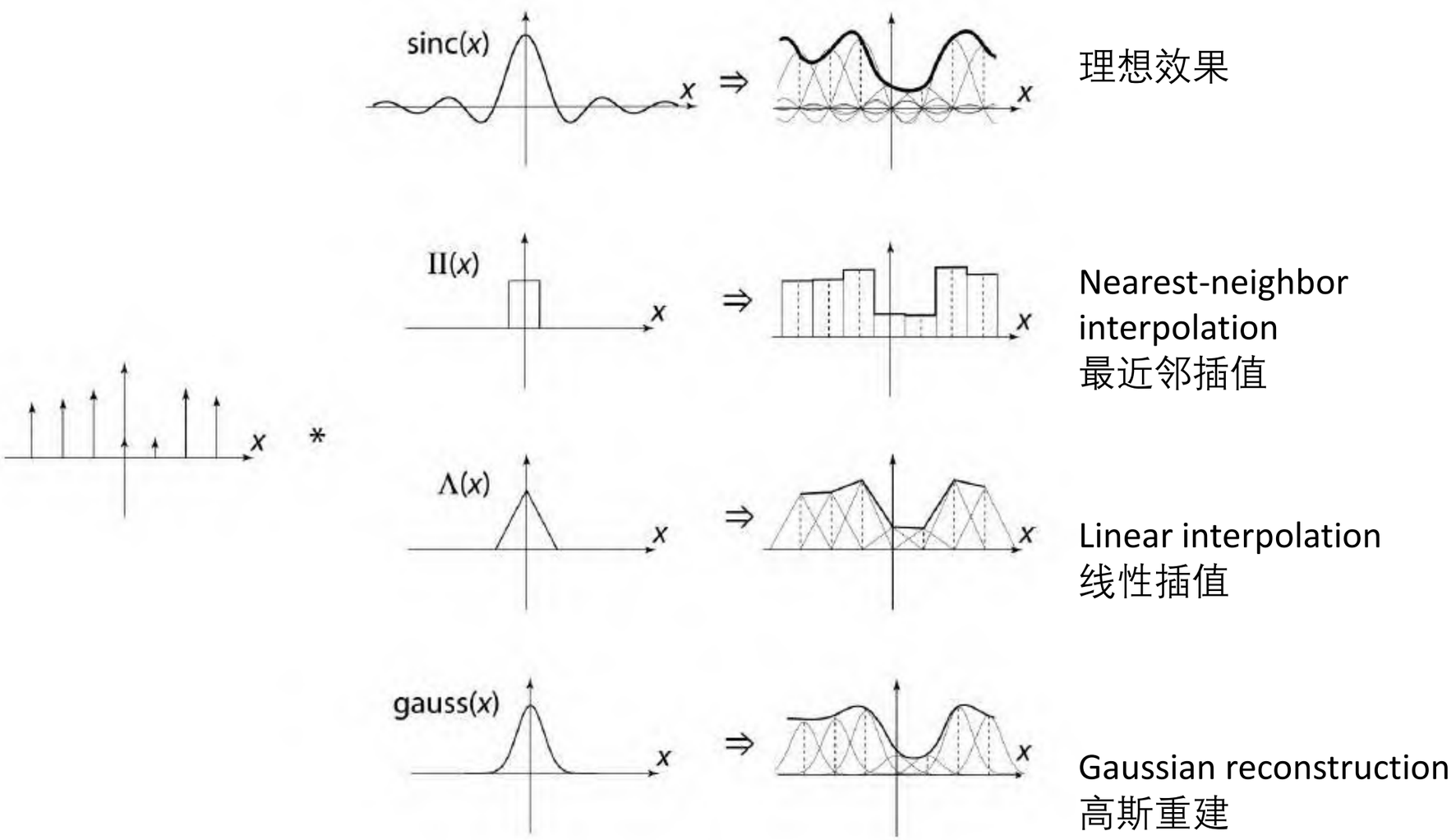
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - *用整数点估计非整数点的值，卷积 h 来实现，* 称之为 重建滤波

$$\tilde{f} = h * f_F$$

Adapted from: S. Seitz

重构滤波器



理想效果

Nearest-neighbor interpolation
最近邻插值

Linear interpolation
线性插值

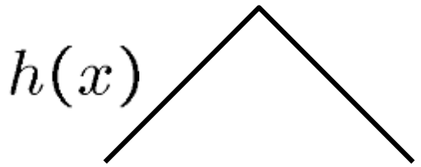
Gaussian reconstruction
高斯重建

Source: B. Curless

重构滤波器

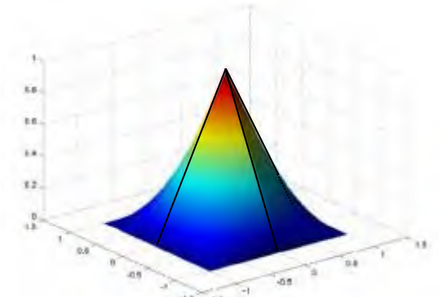
- 二维情形:

$h(x)$

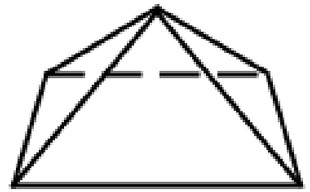


performs
linear interpolation

$h(x,y)$

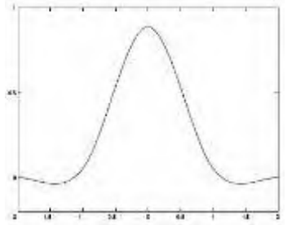


(tent function) performs
bilinear interpolation



更好的滤波产生更好的图像

- **Bicubic** 一般是最好的选择
- 请选择 数值分析 相关资料来学习具体内容



Cubic reconstruction filter

$$r(x)=\frac{1}{6}\begin{cases}(12-9B-6C)|x|^3+(-18+12B+6C)|x|^2+(6-2B) & |x|<1\\ (-B-6C)|x|^3+(6B+30C)|x|^2+(-12B-48C)|x|+(8B+24C) & 1\le|x|<2\\ 0 & otherwise\end{cases}$$

图像插值

原图:  x 10

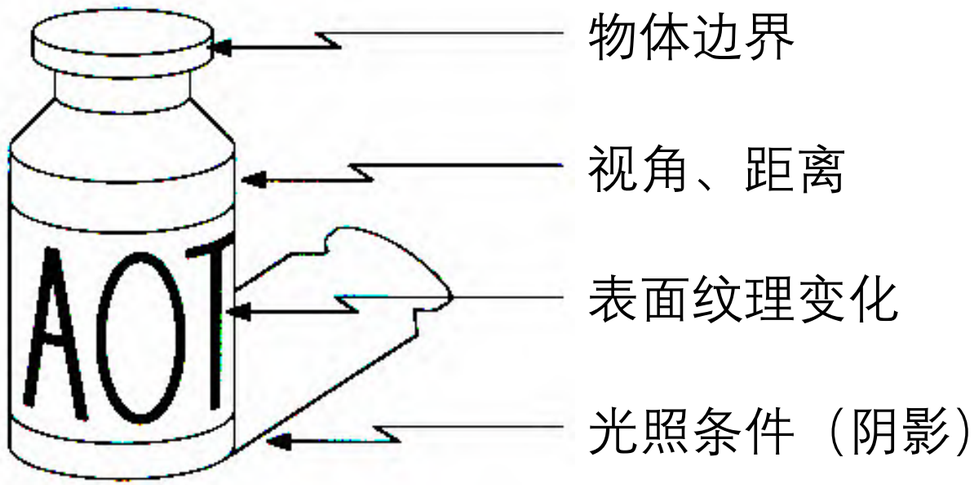
Nearest-neighbor interpolation
最近邻插值

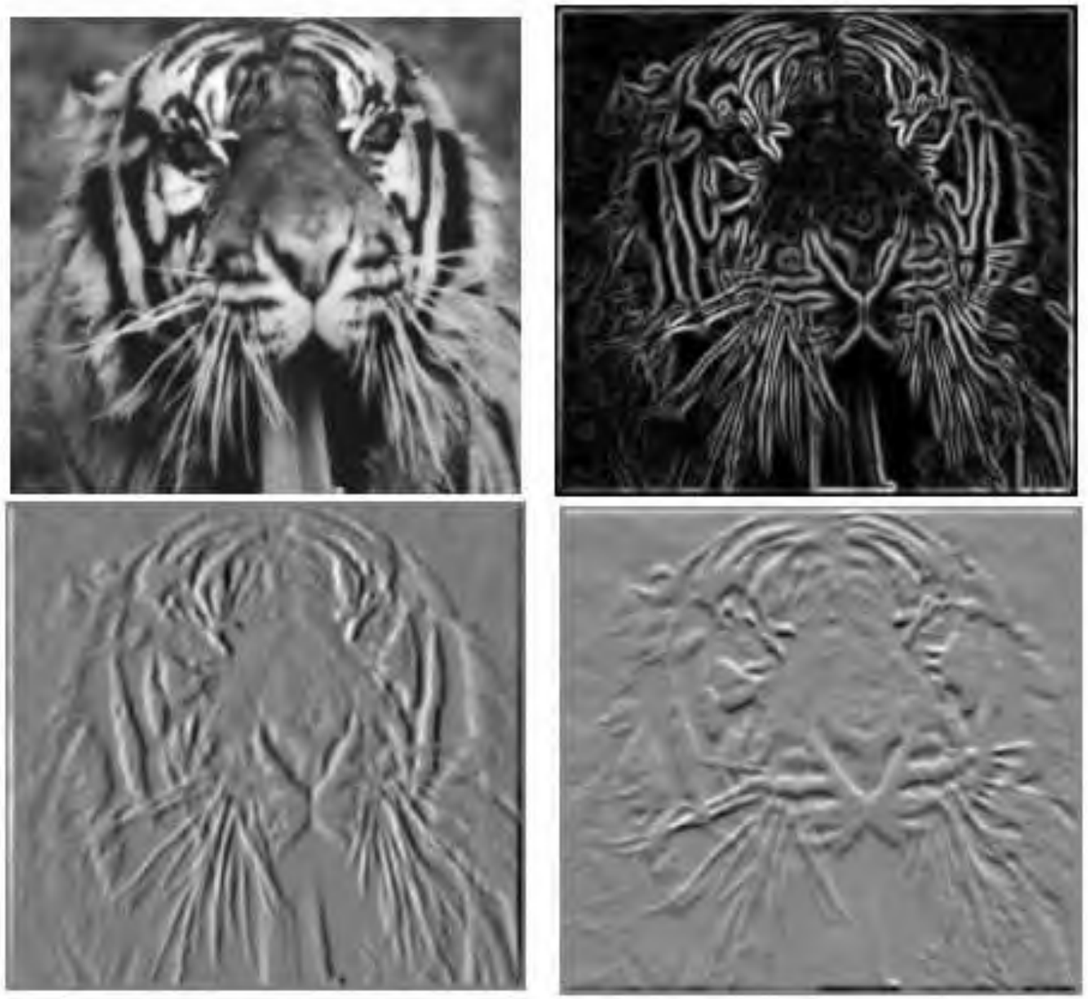
Bilinear interpolation
双线性插值

Bicubic interpolation
双三次插值

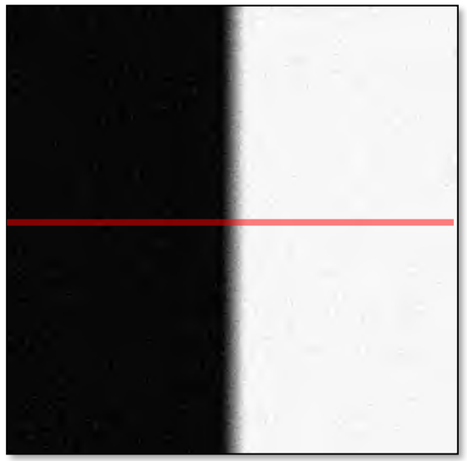
回顾——边缘检测

边缘的定义、图像梯度



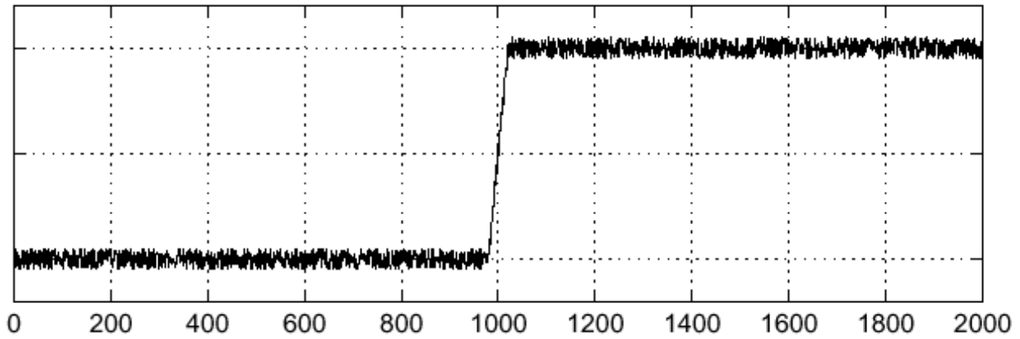


噪声的影响——高斯滤波预处理

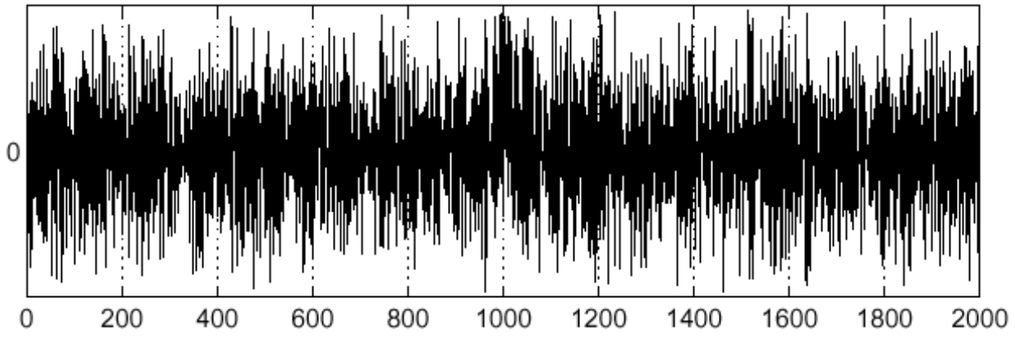


有噪声的图像

$f(x)$



$\frac{d}{dx}f(x)$



Source: S. Seitz

Image with Edge

Edge Location

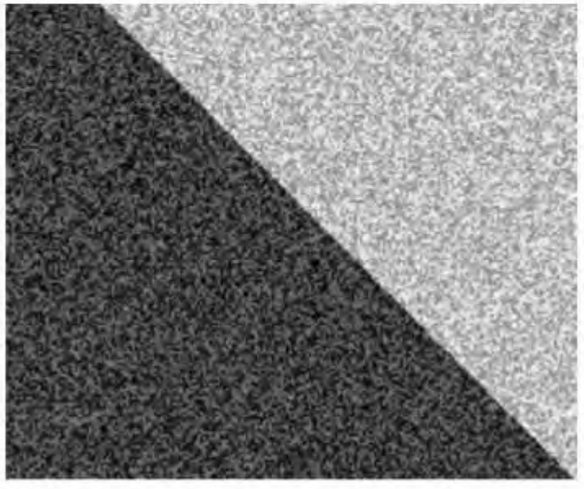
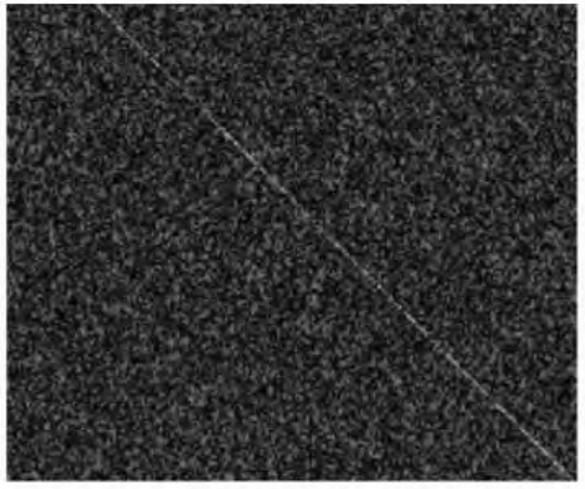
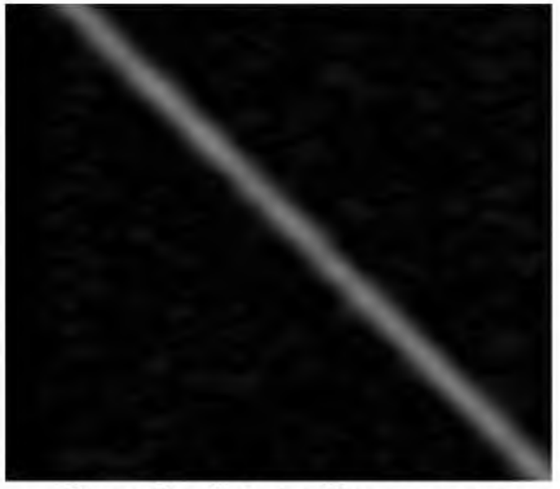


Image + Noise



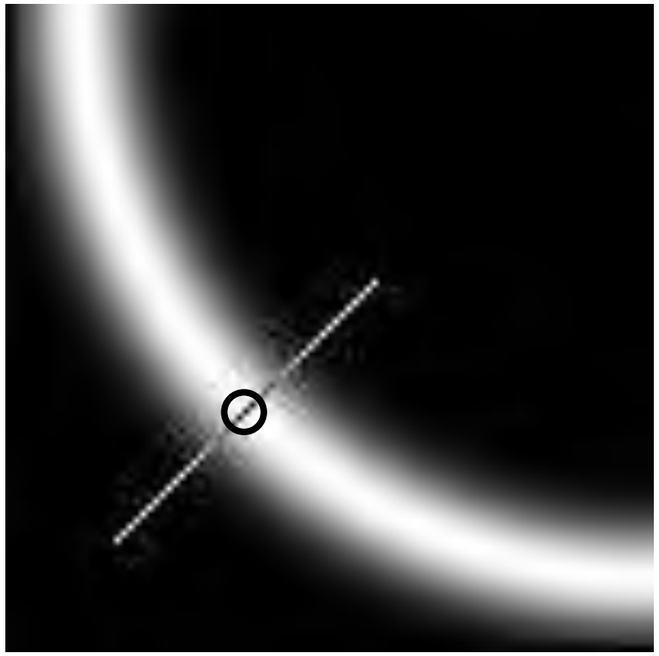
Derivatives detect
edge *and* noise

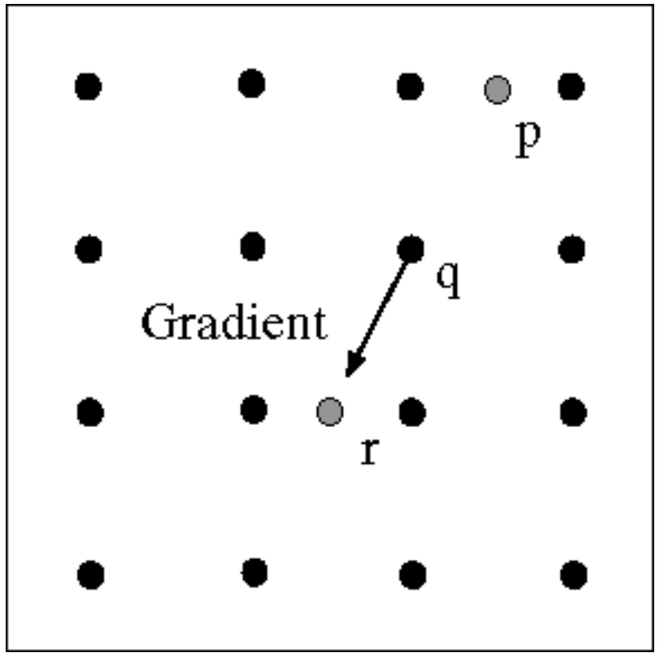


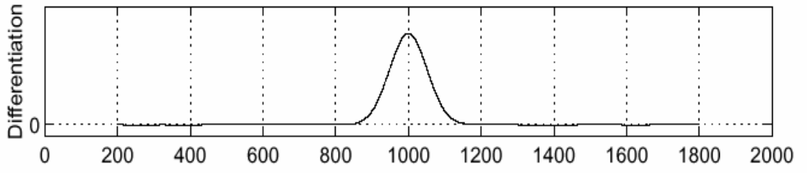
Smoothed derivative removes
noise, but blurs edge

Non-maximum suppression: 非极大抑制

将“厚边缘”细化为“细边缘”







- 对于图像中的每一个像素点，确定其梯度的方向，计算该方向子区域的梯度幅度，在该方向上，如果该像素点的梯度幅度不是局部最大值，则将其设置为0。

通过阈值得到边缘、连接边缘

- 2 个阈值, 3 种情况
  - R > T: 强边缘
  - R < T but R > t: 弱边缘
  - R < t: 非边缘
- 强边缘一定是边缘
- 如果弱边缘与强边缘联通则也是边缘
- 通过周围八个像素点进行搜索

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

MATLAB: `edge(image,'canny')`

Canny边缘检测

1. 使用DoG对图像预处理

2. 找到梯度的幅度和方向

3. Non-maximum suppression

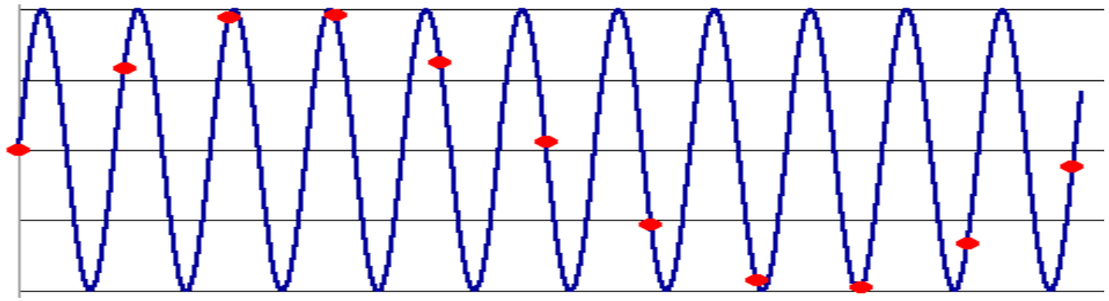
4. 连接边缘:
   - 定义高低两个阈值，得到弱边缘和强边缘
   - 用强边缘作为初始边缘，用弱边缘连接强边缘

Source: D. Lowe, L. Fei-Fei, J. Redmon

回顾——采样与插值

下采样、低分辨率下 的 混淆现象

Aliasing：混淆



- 当采样率不够高的时候，有可能会导致采样的信息有问题
- 导致得到的信号出现问题（频率变低）导致 *alias 混淆*

- 为了避免混淆:
  - 采样率要不小于图像最大频率的2倍
    - 也就是说，每个周期至少两个采样
  - **最小采样率也被称为**Nyquist rate

Source: L. Zhang

使用高斯模糊以后的下采样

Gaussian 1/2

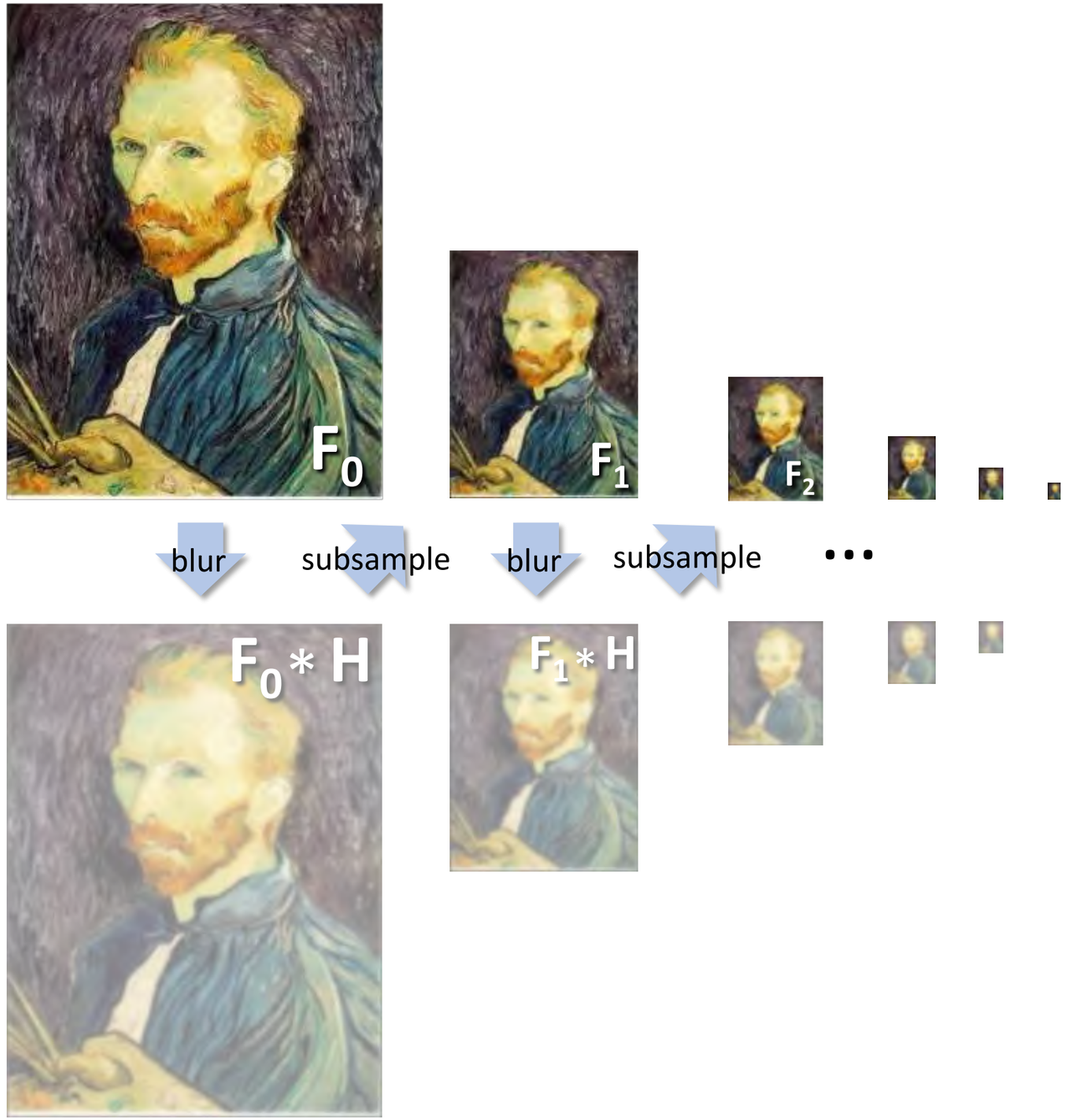
G 1/4

G 1/8

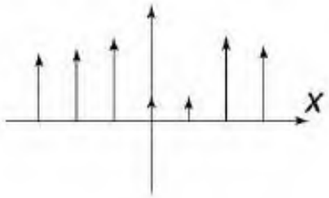
- 解决方案：先模糊，再下采样

Source: S. Seitz

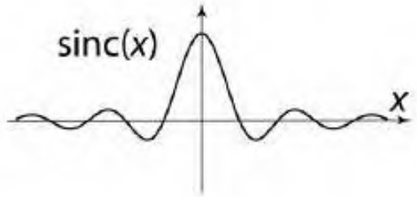
Gaussian pyramid
高斯金字塔



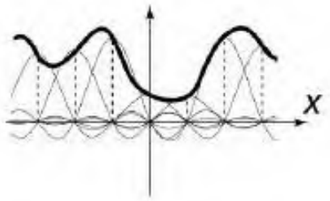
上采样：重构滤波器



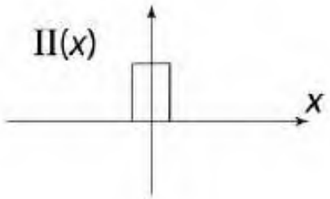
*



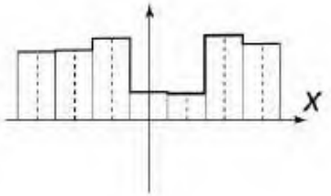
⇒



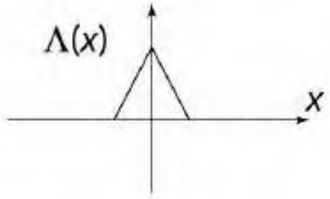
理想效果



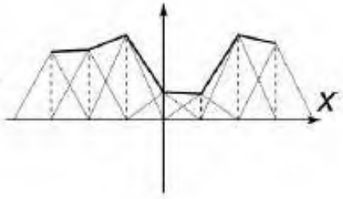
⇒



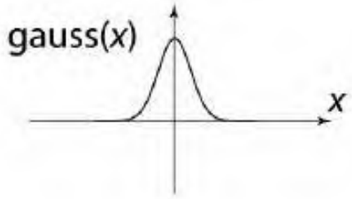
Nearest-neighbor interpolation
最近邻插值



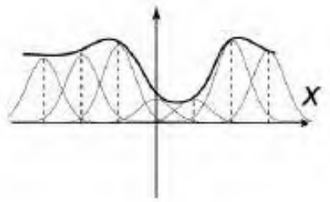
⇒



Linear interpolation
线性插值



⇒



Gaussian reconstruction
高斯重建

Source: B. Curless

图像插值

原图:  x 10

Nearest-neighbor interpolation
最近邻插值

Bilinear interpolation
双线性插值

Bicubic interpolation
双三次插值

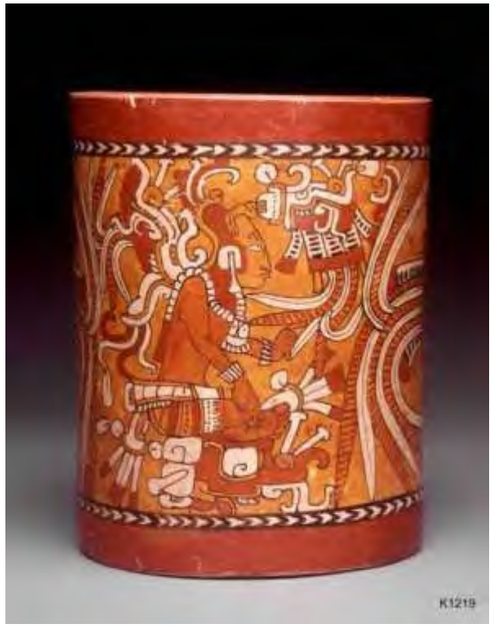
角点检测

全景图：整合多幅图

用一张图记录一个场景!

轮转全景扫描

用一张图记录一个物体!



Rollout Photographs © Justin Kerr
http://research.famsi.org/kerrmaya.html

Also known as “cyclographs”, “peripheral images”

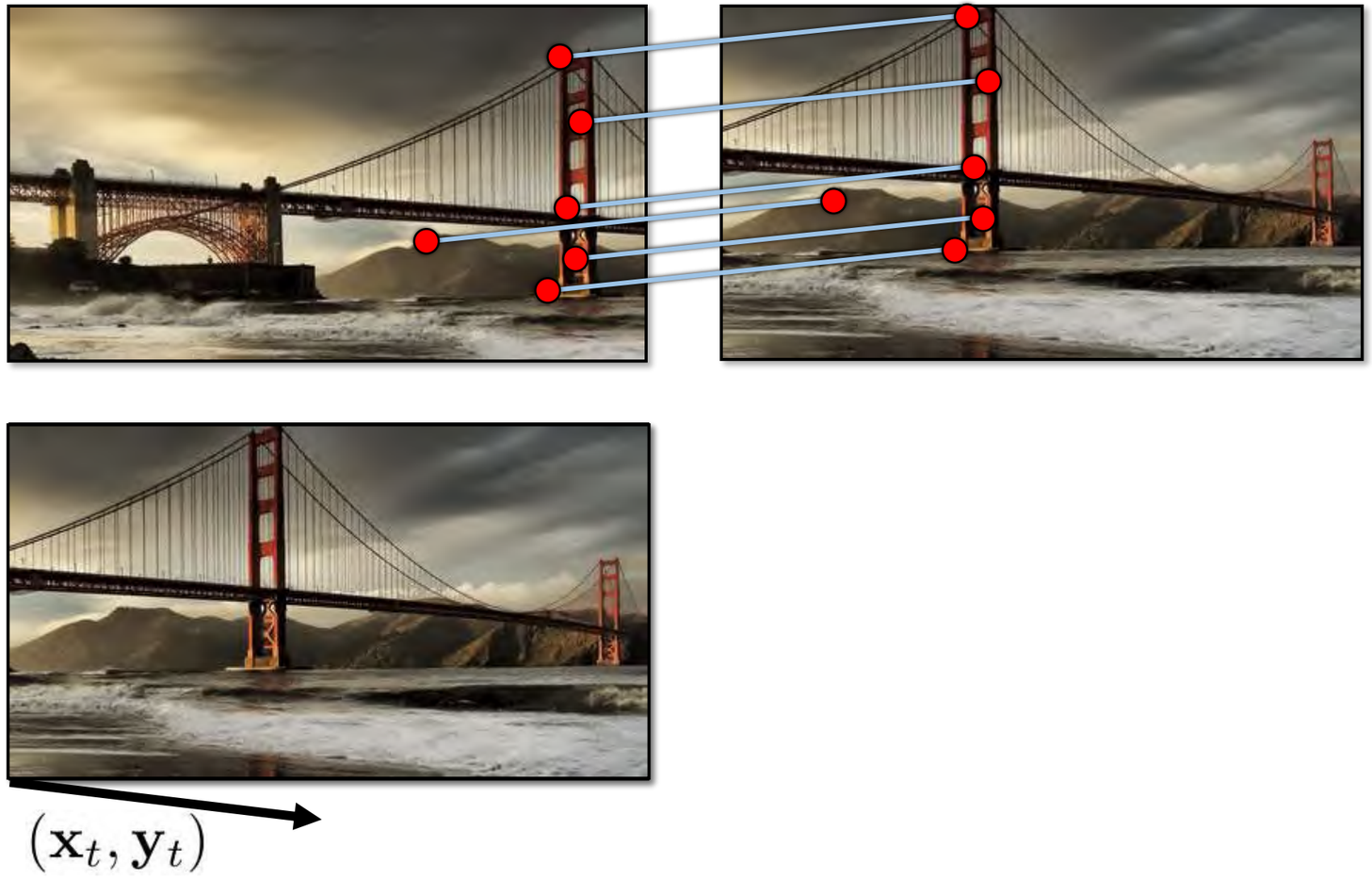
更大的场景…

Credit: Matt Brown

如何自动获取全景图: 图像拼接

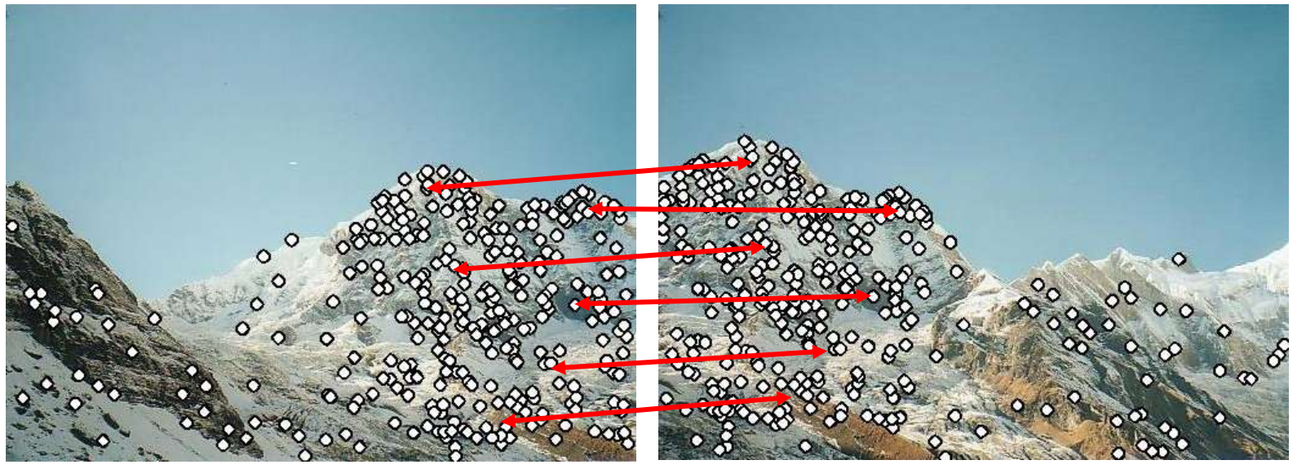
- 怎么把两张图像结合起来?


图像拼接



为什么提取特征

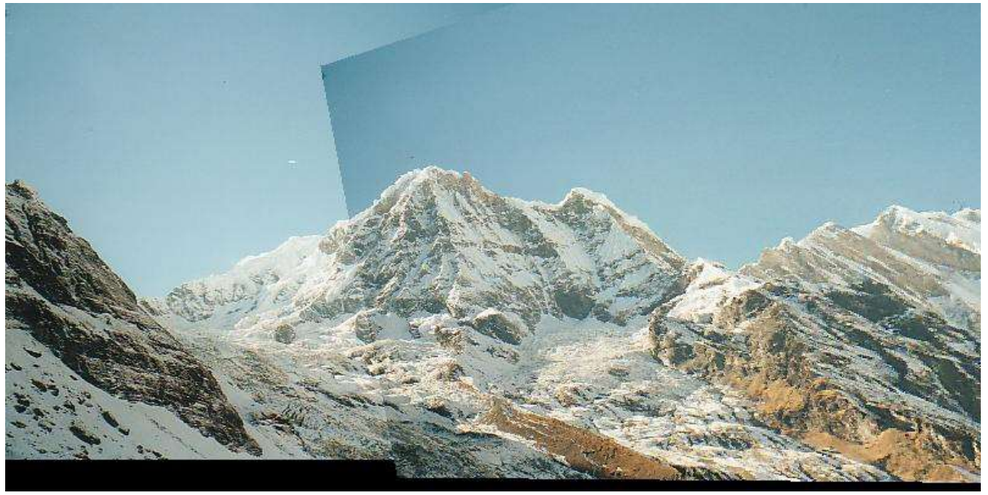
- 怎么把两张图像结合起来?



Step 1: 提取特征
Step 2: 特征匹配

为什么提取特征

- 怎么把两张图像结合起来?



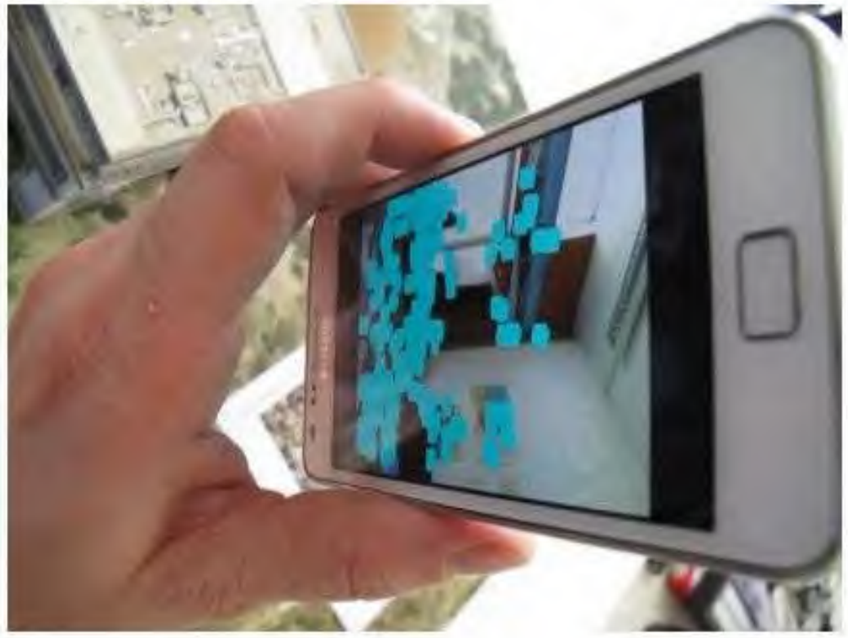
Step 1: 提取特征

Step 2: 特征匹配

Step 3: 缝合图像

应用: Visual SLAM （Simultaneous Localization and Mapping）

- “视觉同时定位与地图构建”：在未知环境中构建地图，同时跟踪代理在该环境中的位置。



图像匹配

by Diva Sian

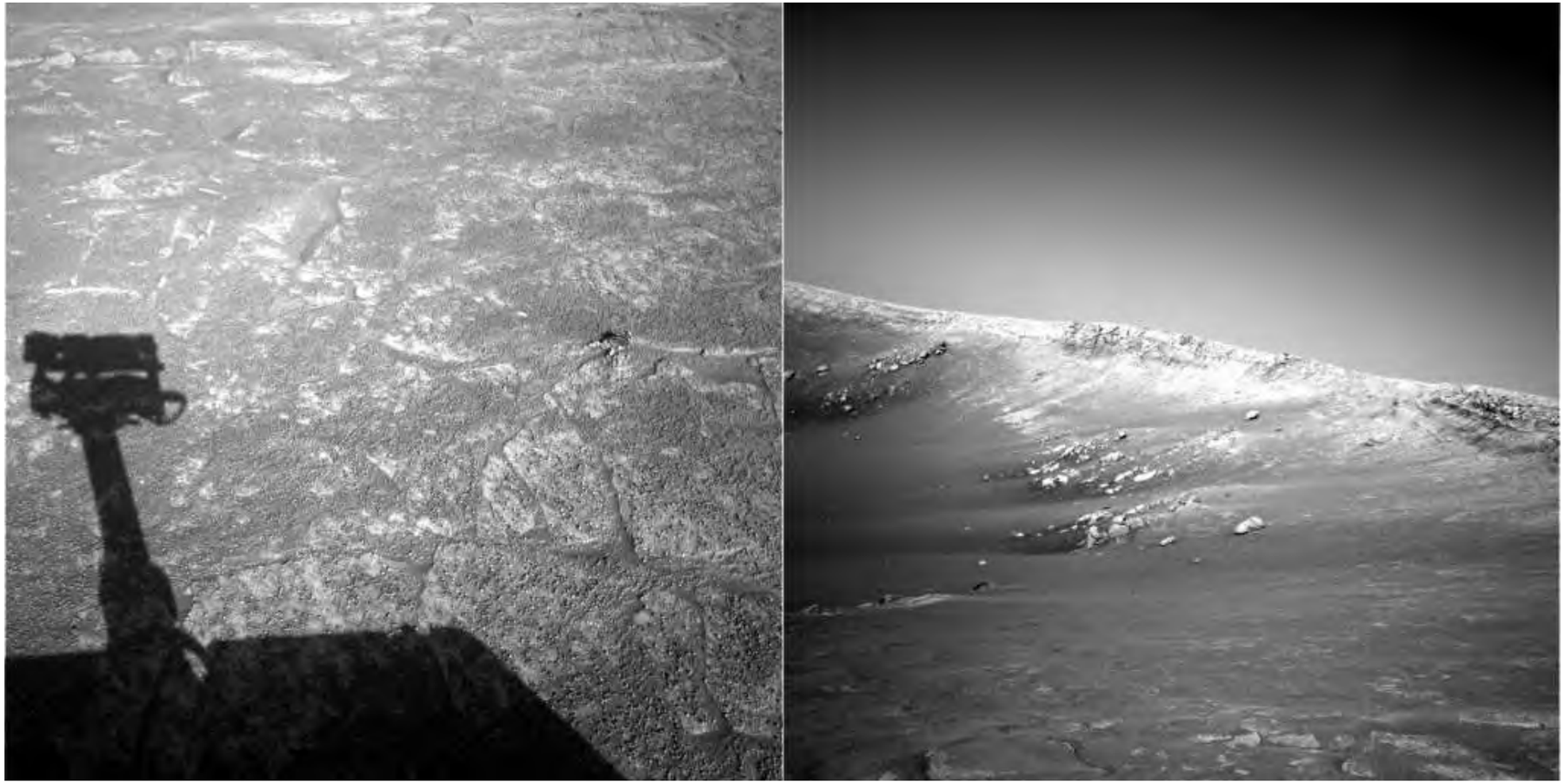
by swashford

加大难度: 视角变化

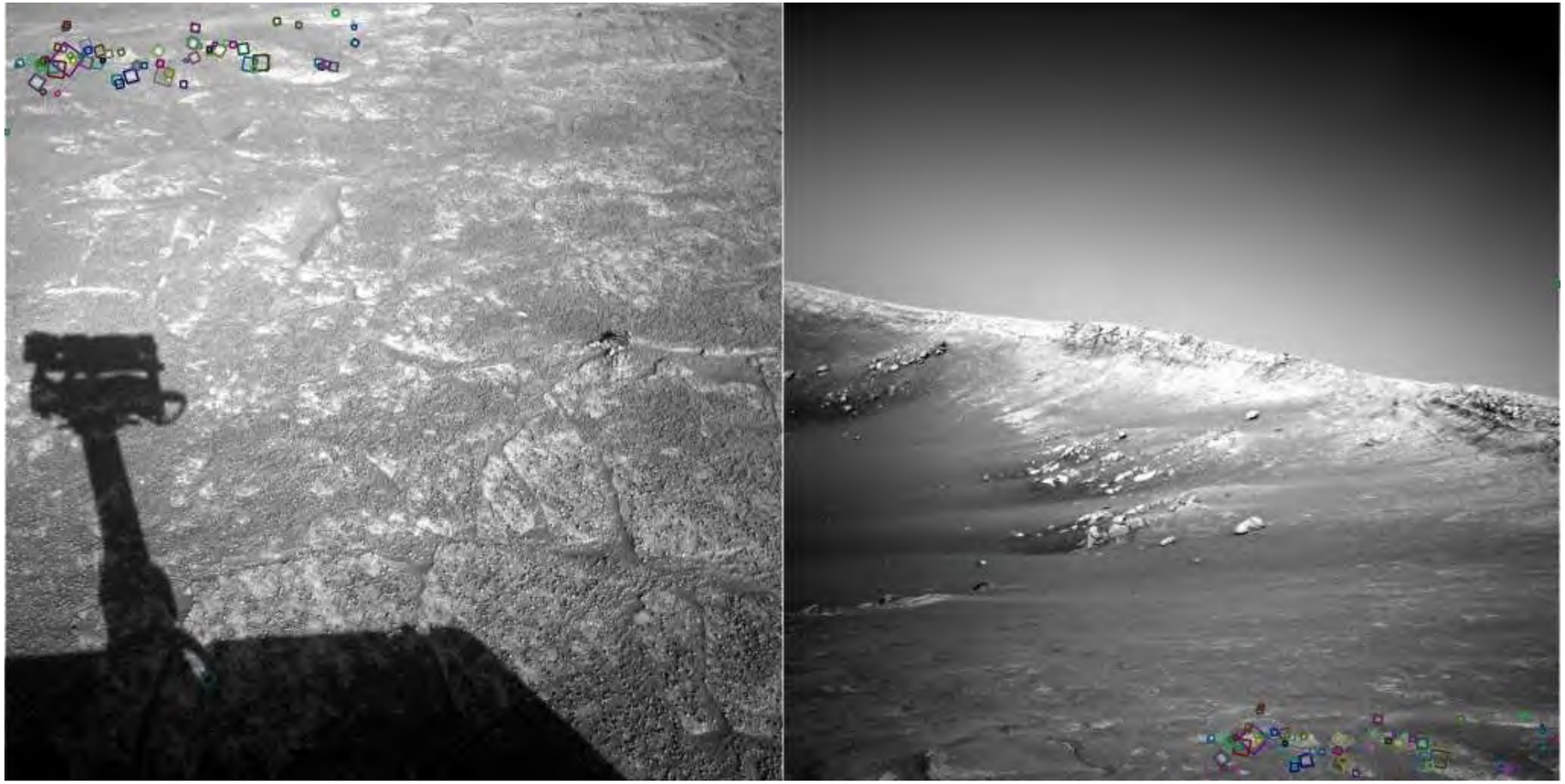
by Diva Sian

by scgbt

更难的情况：你能找到匹配的位置吗

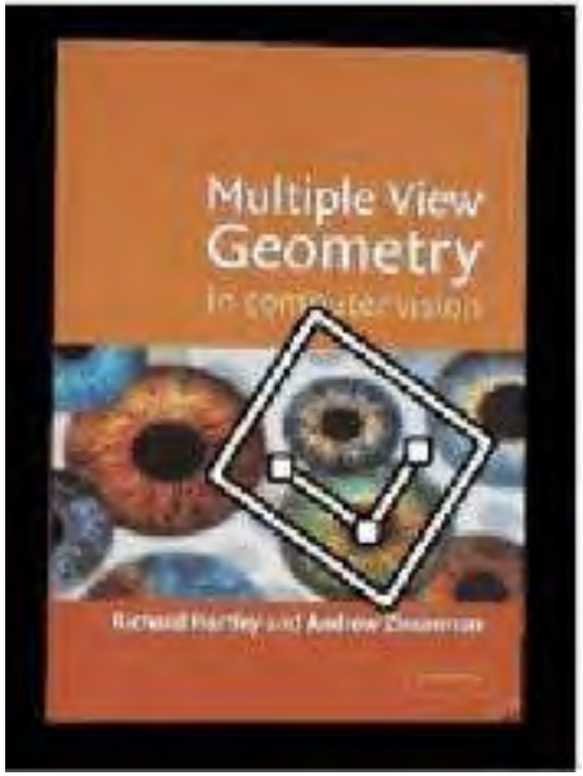


答案



NASA Mars Rover images
with SIFT feature matches

应用：物体识别与检测

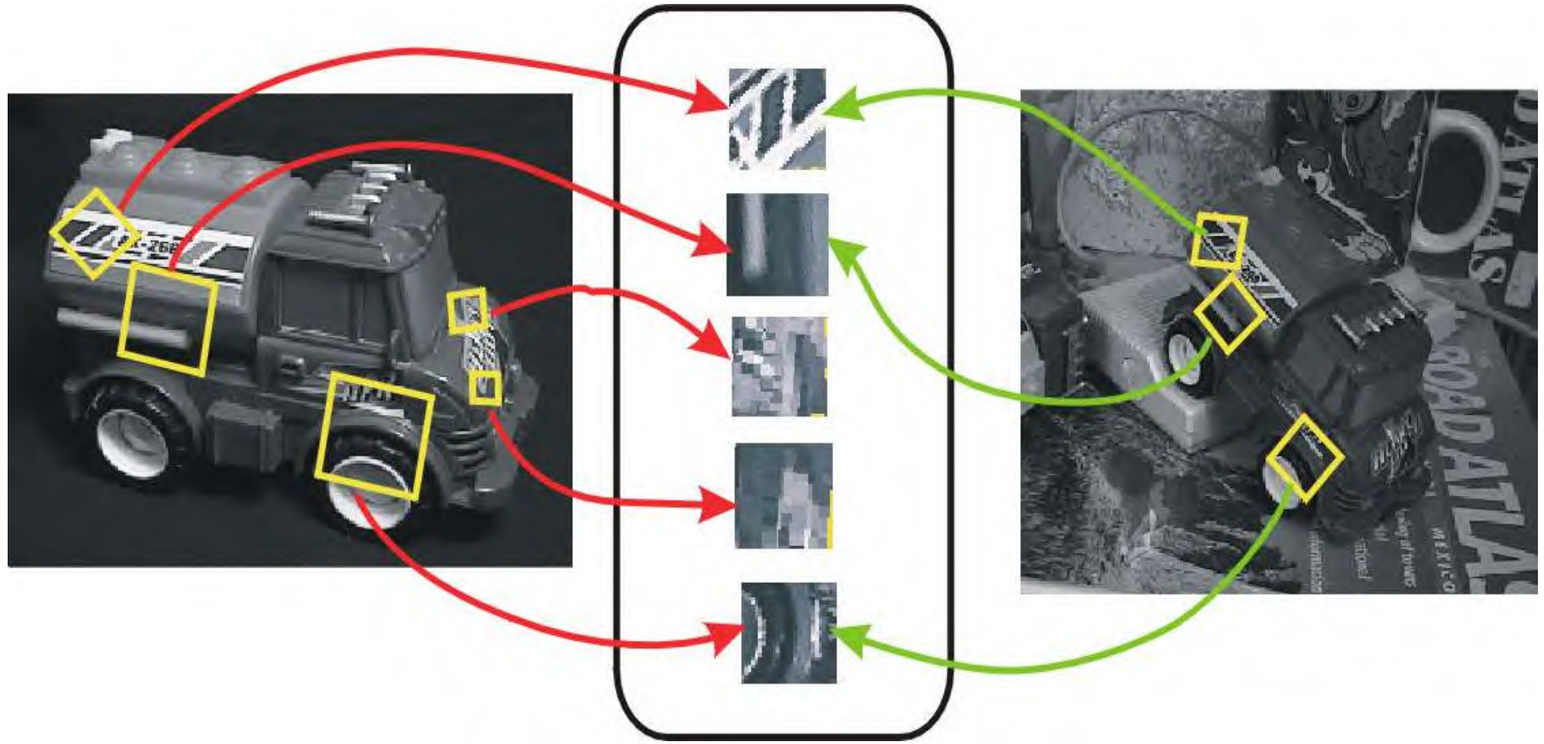


特征匹配

不变局部特征

找到在图像变换下仍然保持不变的图像特征

- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



特征提取器

局部特征在感知领域的优势

局部鲁棒:

- 局部特征对遮挡等鲁棒（总有没被遮挡的部分）

数量上:

- 局部特征数量更多（回顾：视觉冗余）

可区分性:

- 拥有重组信息量，可以区分不同的物体

效率:

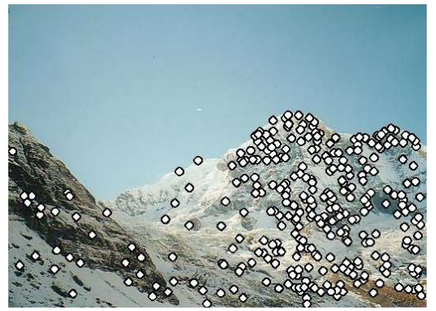
- 适当设计算法可以达到实时性处理

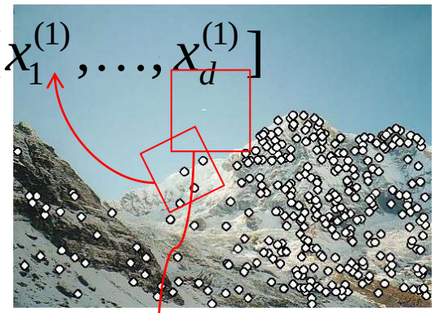
回顾 计算机视觉的“感知”：基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点

2) 解释 Description: 在关键点周围提取特征

3) 匹配 Matching: 根据两个视角下的特征进行匹配





$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

Kristen Grauman

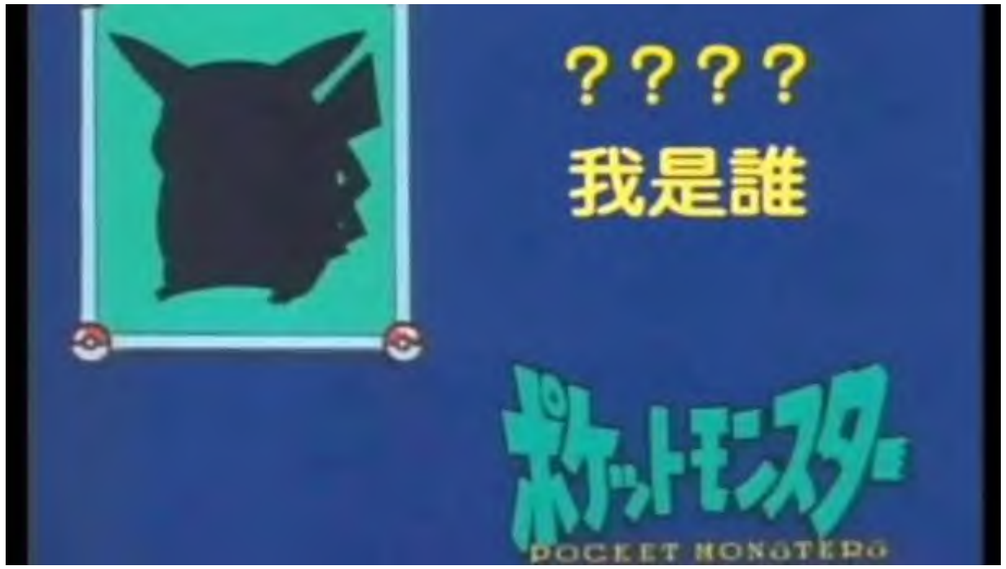
什么样的特征是好的特征?

Uniqueness：唯一性

特征点在图像或图像集合中具有**独特**的外观或属性

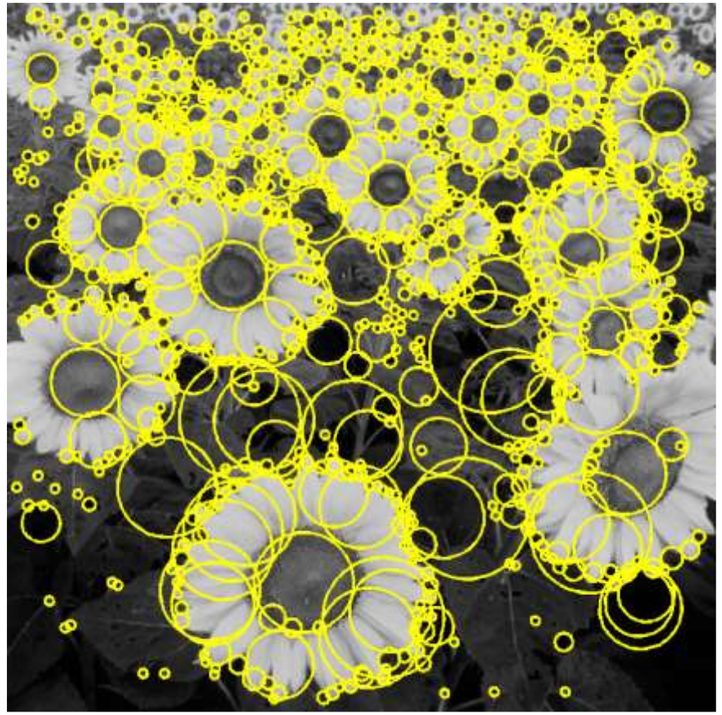
- 使其容易与其他特征点区分开来，不会产生混淆

怎样定义“独特”？



特征提取：角点

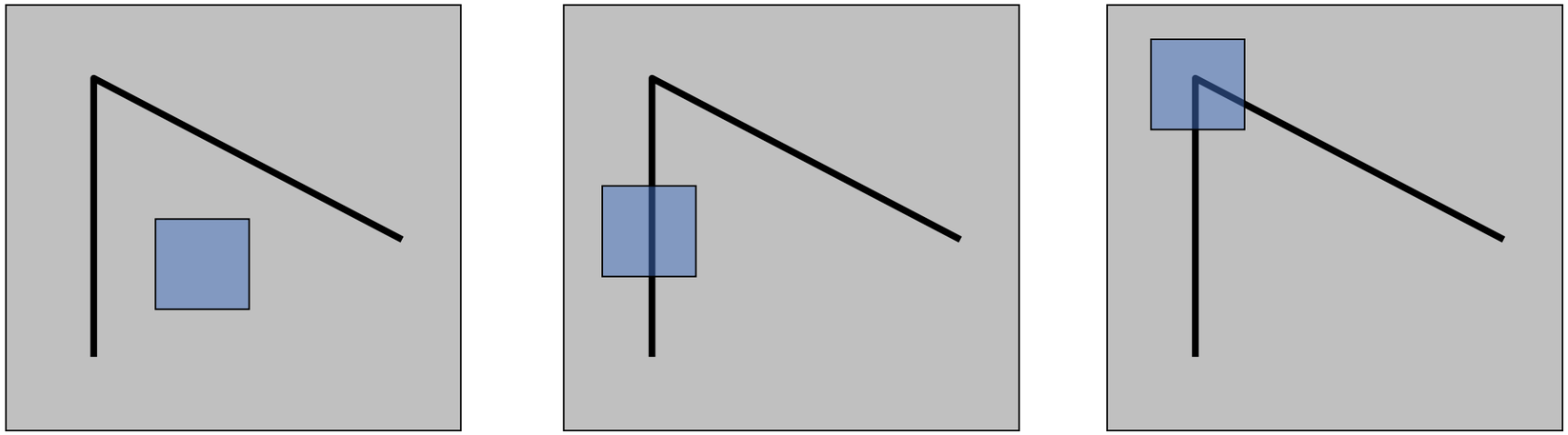
- 角点（Corners）： 角点是图像中具有明显边缘转折的像素点。



衡量局部的“唯一性”

针对一个小的区域（滑动窗口）

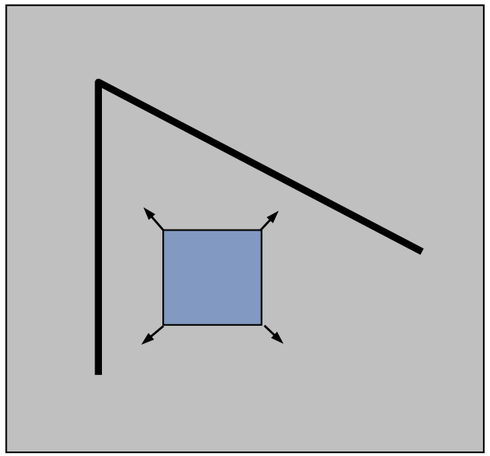
- 什么样的点附近区域代表更好的特征点?
- 我们从一个“物体”内去考虑“唯一性”
- 什么点在描述物体时更“独特”?



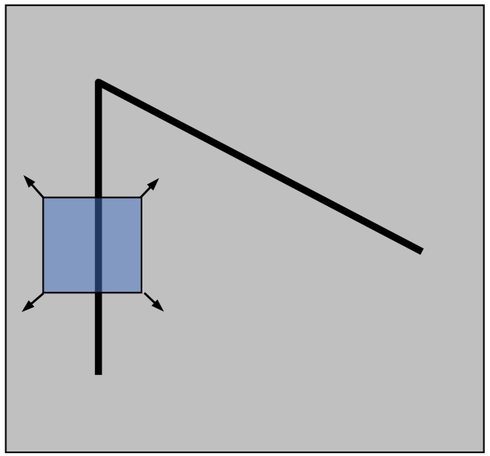
Credit: S. Seitz, D. Frolova, D. Simakov

衡量局部的“唯一性”

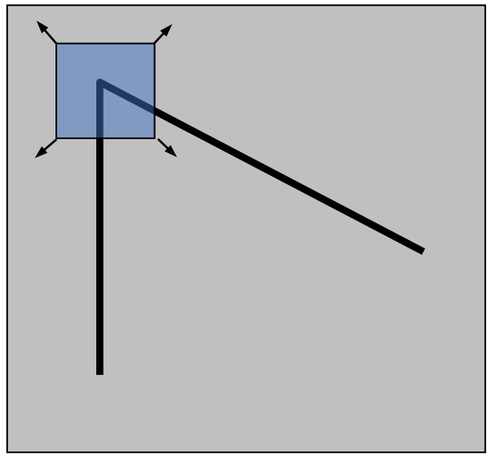
- 我们从窗口内提取特征（图像梯度、DoG）
- 当窗口向四周滑动，特征怎样变化?



“flat” 普通区域:
所有方向都没有过多变化



“edge” 边缘:
边缘区域没有变化

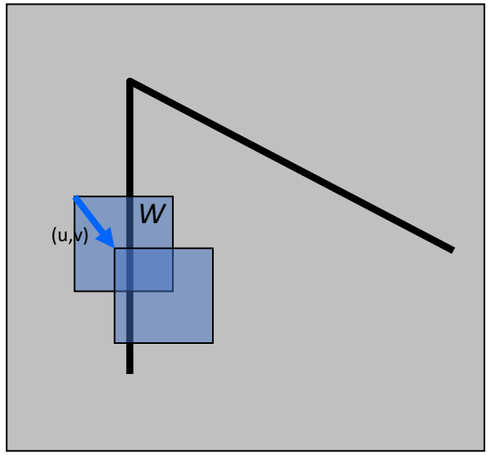


“corner” 角点:
在各个方向都有交大的变化

Credit: S. Seitz, D. Frolova, D. Simakov

Harris 角点检测: 数学定义

假设我们考虑窗口*W* 滑动了 (*u,v*)



- 我们对比每个对应像素，计算梯度差异，并求和，summing up the squared differences (SSD)
- SSD 定义为 *E*(*u,v*):

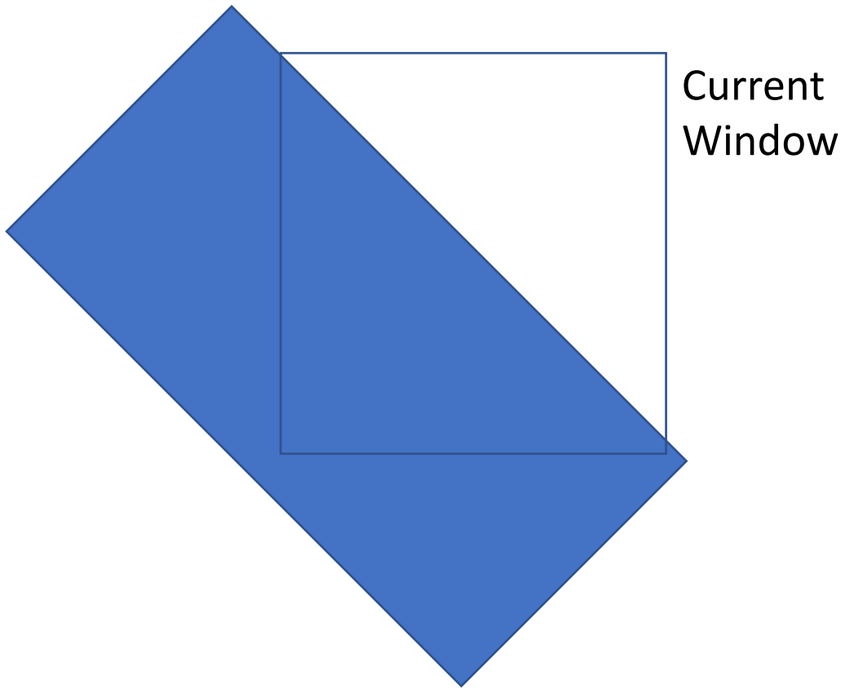
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

- SSD越高特征点越“唯一”

Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". *Alvey Vision Conference*.

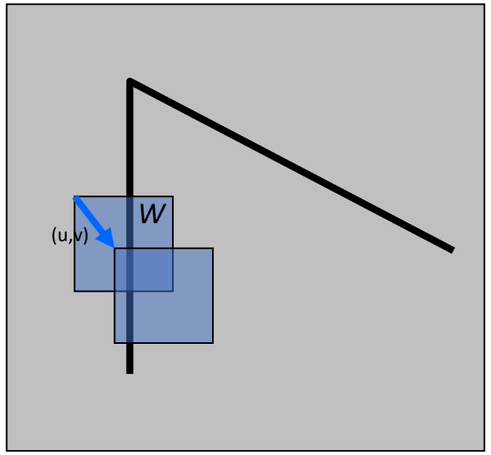
Harris 角点 – 为什么这么复杂?

- 为什么不直接根据xy方向梯度计算幅值，取最大的点?
  - 对角线的点可能满足这样的计算方式



Harris 角点检测: 数学定义

假设我们考虑窗口*W* 滑动了 (*u,v*)



- 我们对比每个对应像素，计算梯度差异，并求和，summing up the squared differences (SSD)
- SSD 定义为 *E*(*u,v*):

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

- SSD越高特征点越“唯一”
- 如果我们对每个窗口、每个偏移(u,v)都去计算，则会比较慢

Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". *Alvey Vision Conference*.

Small motion assumption：“小运动假设”

对 *I* 做泰勒展开（离散、邻域 下的 粗糙定义）：

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

如果 (*u*, *v*) 很小, 则高阶项可以近似消除

$$\begin{aligned} I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

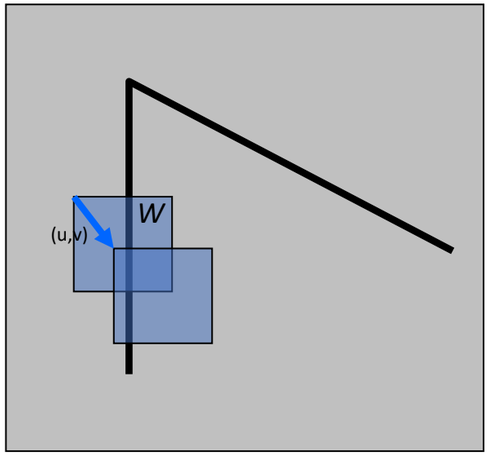
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

小运动假设允许我们使用更简单的数学模型来描述复杂的运动。

Harris 角点检测: 数学定义

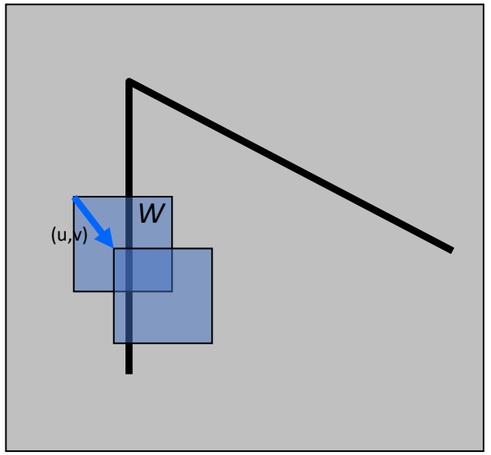
假设我们考虑窗口*W* 滑动了 (*u,v*)

- SSD *E(u,v)*:



$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\end{aligned}
$$

Harris 角点检测: 数学定义



假设我们考虑窗口*W* 滑动了 (*u,v*)

- SSD *E(u,v)*:

$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

- *E*(*u,v*) 可以被近似表示为二次误差函数

二阶矩阵

二维函数*E*(*u*,*v*) 的表面可以局部近似为二次形式

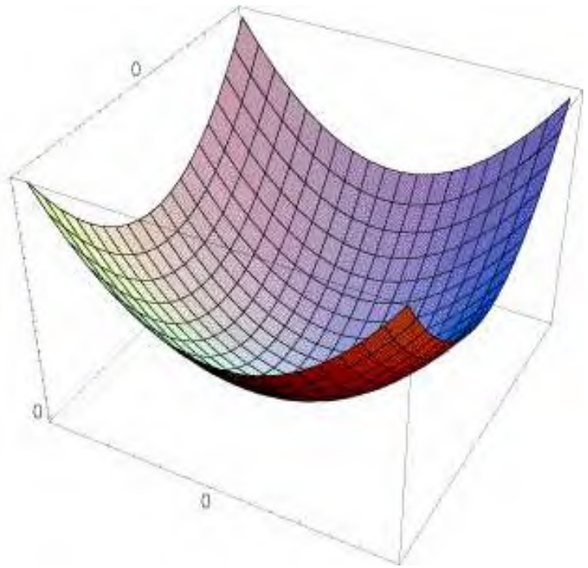
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

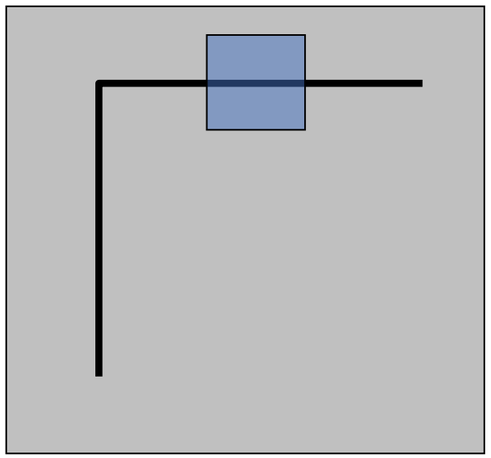


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

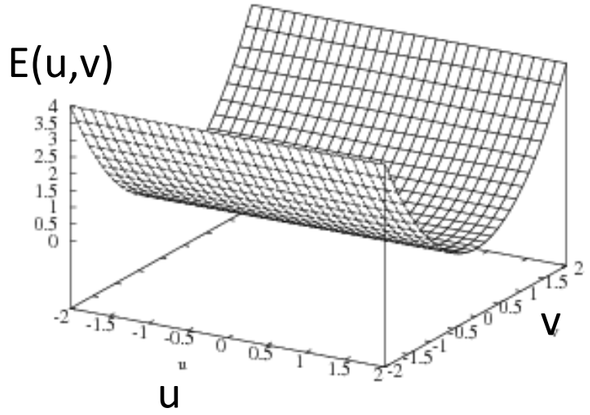
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



水平边缘: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

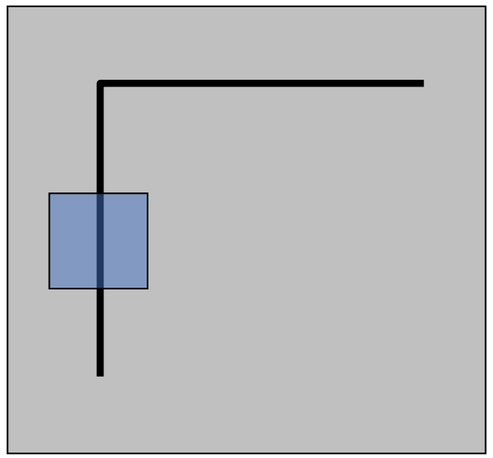


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

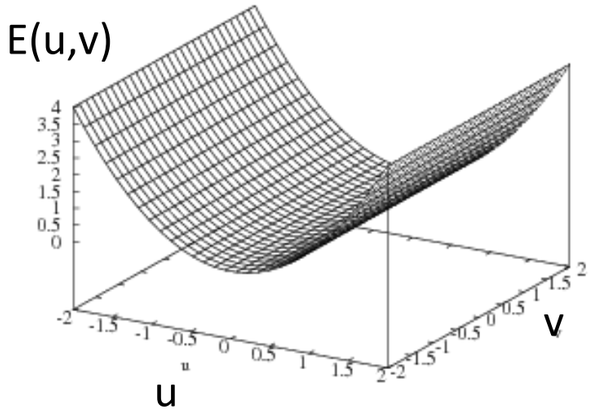
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



垂直边缘: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

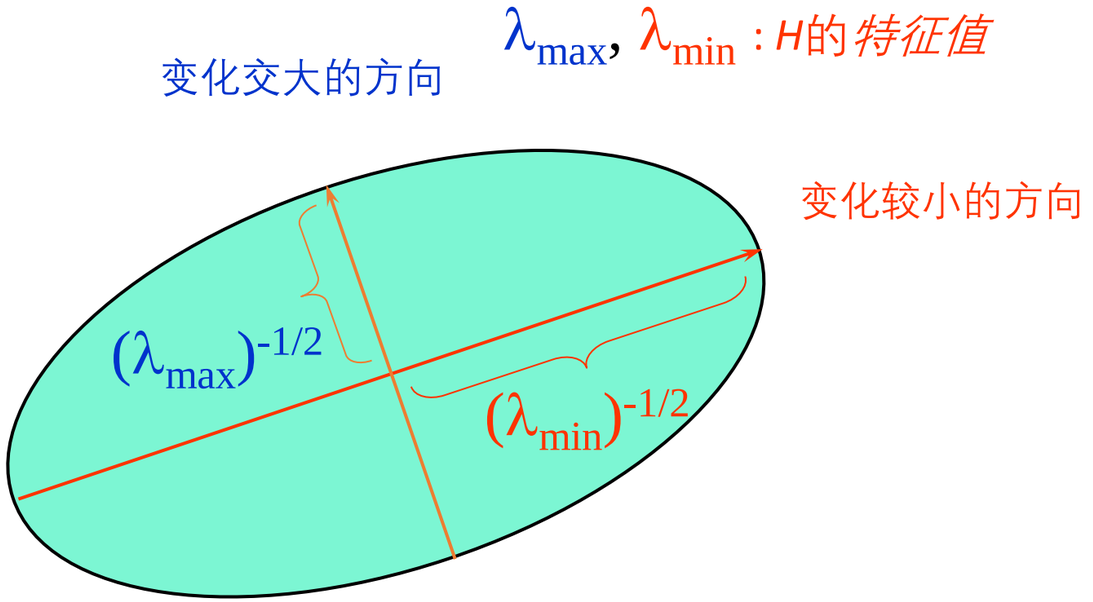


Harris 角点检测: 数学定义

H 在各个方向上的变化速率可以通过*H* 的特征值和特征向量来可视化

椭圆方程:

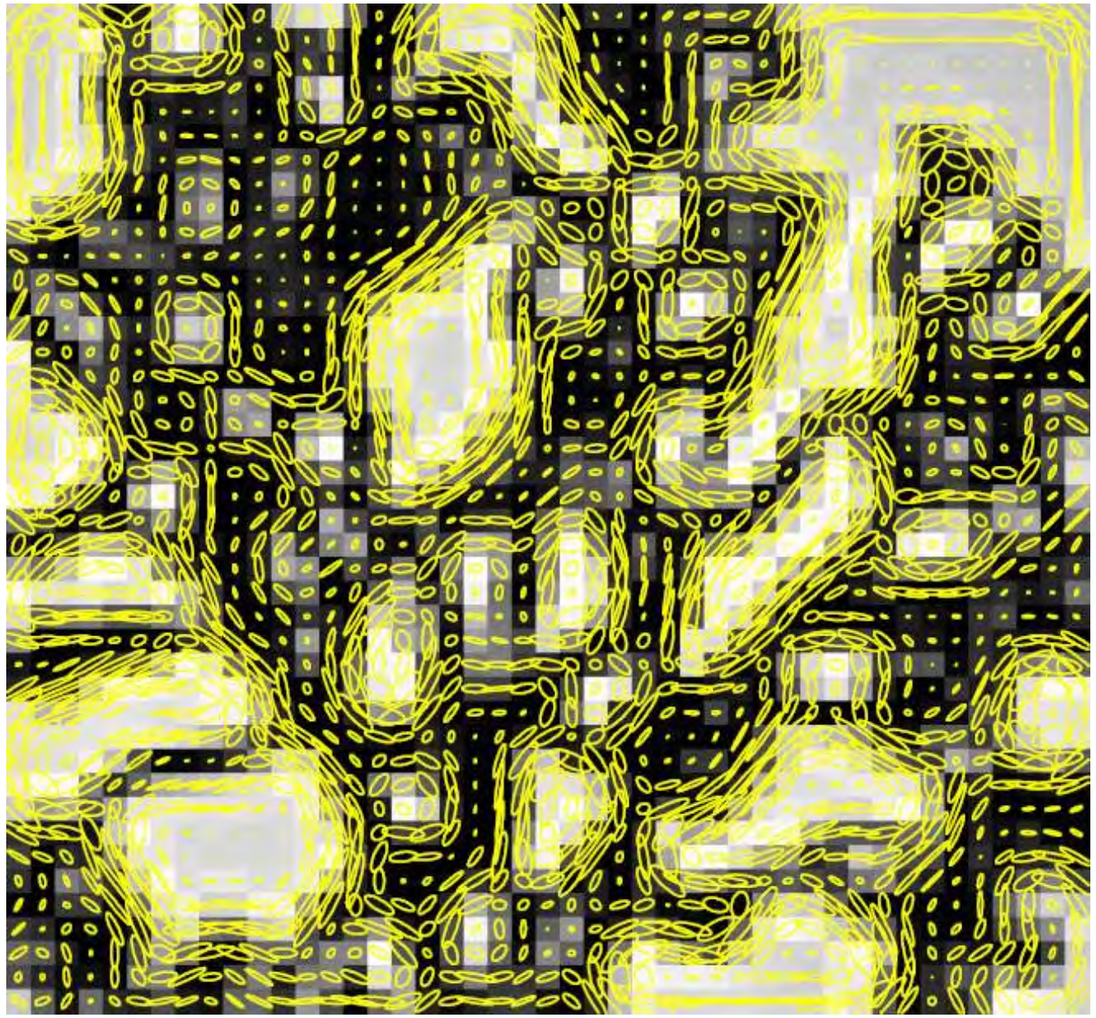
$$[u \quad v] \; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

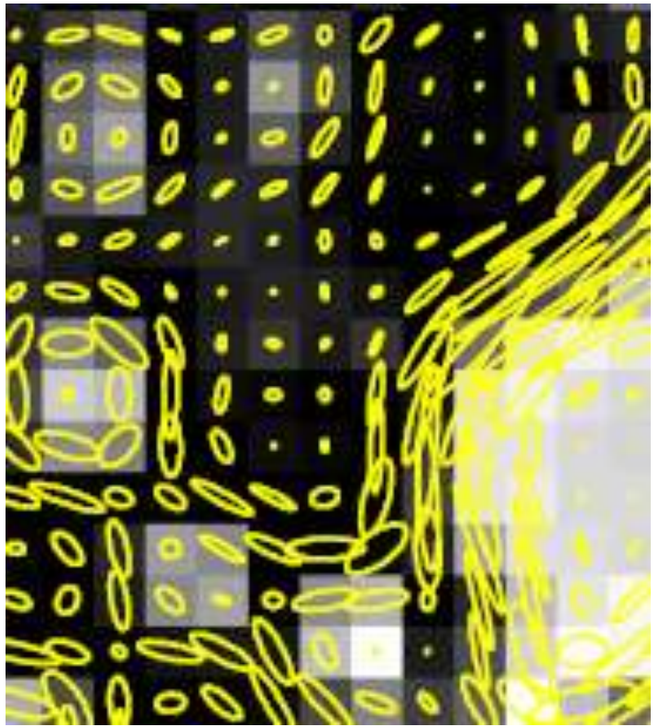


Visualizing M

Slide credit: S. Lazebnik

Visualizing M



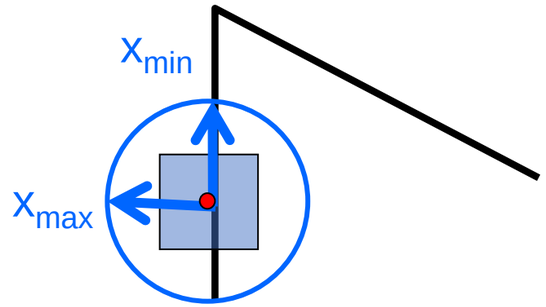


Technical note: M is often best *visualized* by first taking inverse, so long edge of ellipse goes along edge

Slide credit: S. Lazebnik

Harris 角点检测: 数学定义

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$



$$Hx_{\max} = \lambda_{\max} x_{\max}$$

$$Hx_{\min} = \lambda_{\min} x_{\min}$$

H的特征值与特征向量

- 定义了最大和最小的SSD（E）变化方向
- x_{max} = *E* 变化最大的方向
- λ_{max} = 最大方向上变化的幅度x_{max}
- x_{min} = *E* 变化最小的方向
- λ_{min} = 最小方向上变化的幅度x_{min}

Harris 角点检测: 数学定义

λ_{max}, x_{max}, λ_{min}, x_{min} 与特征方向、变化幅度有何关联?

- 我们需要怎样的特征点?

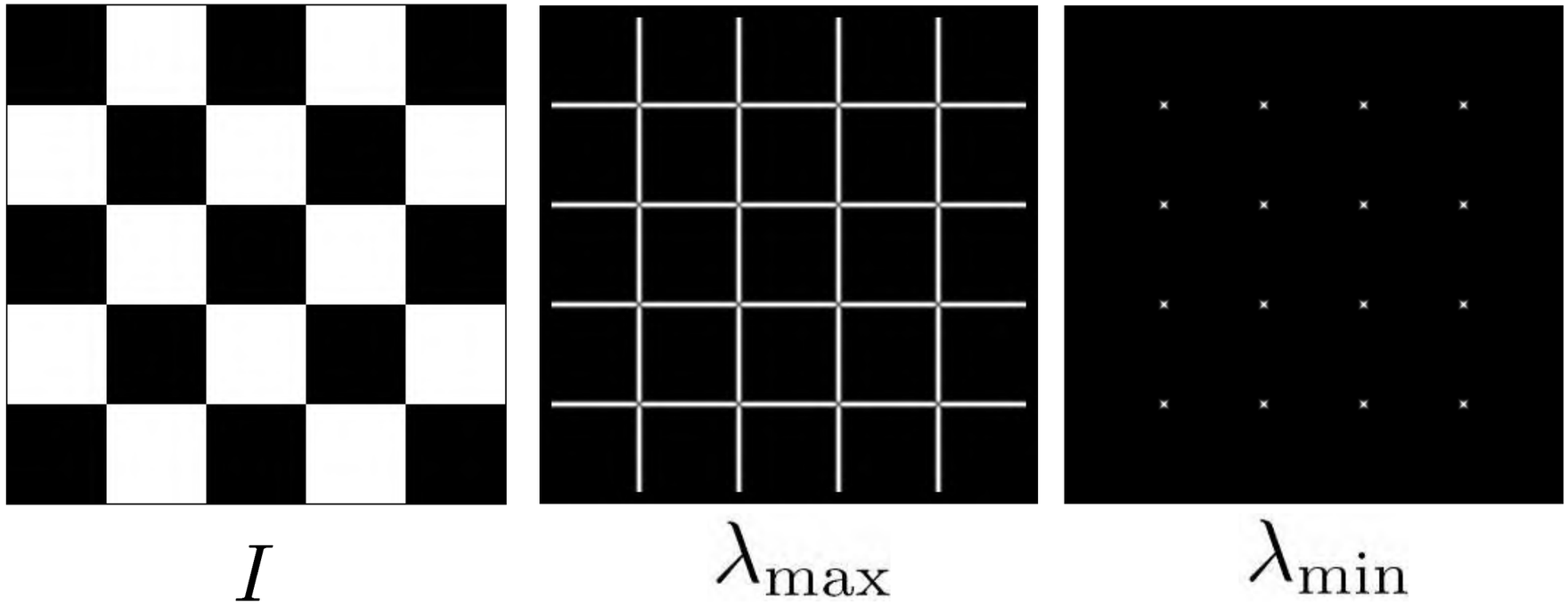
Harris 角点检测: 数学定义

λ_{max}, x_{max}, λ_{min}, x_{min} 与特征方向、变化幅度有何关联?

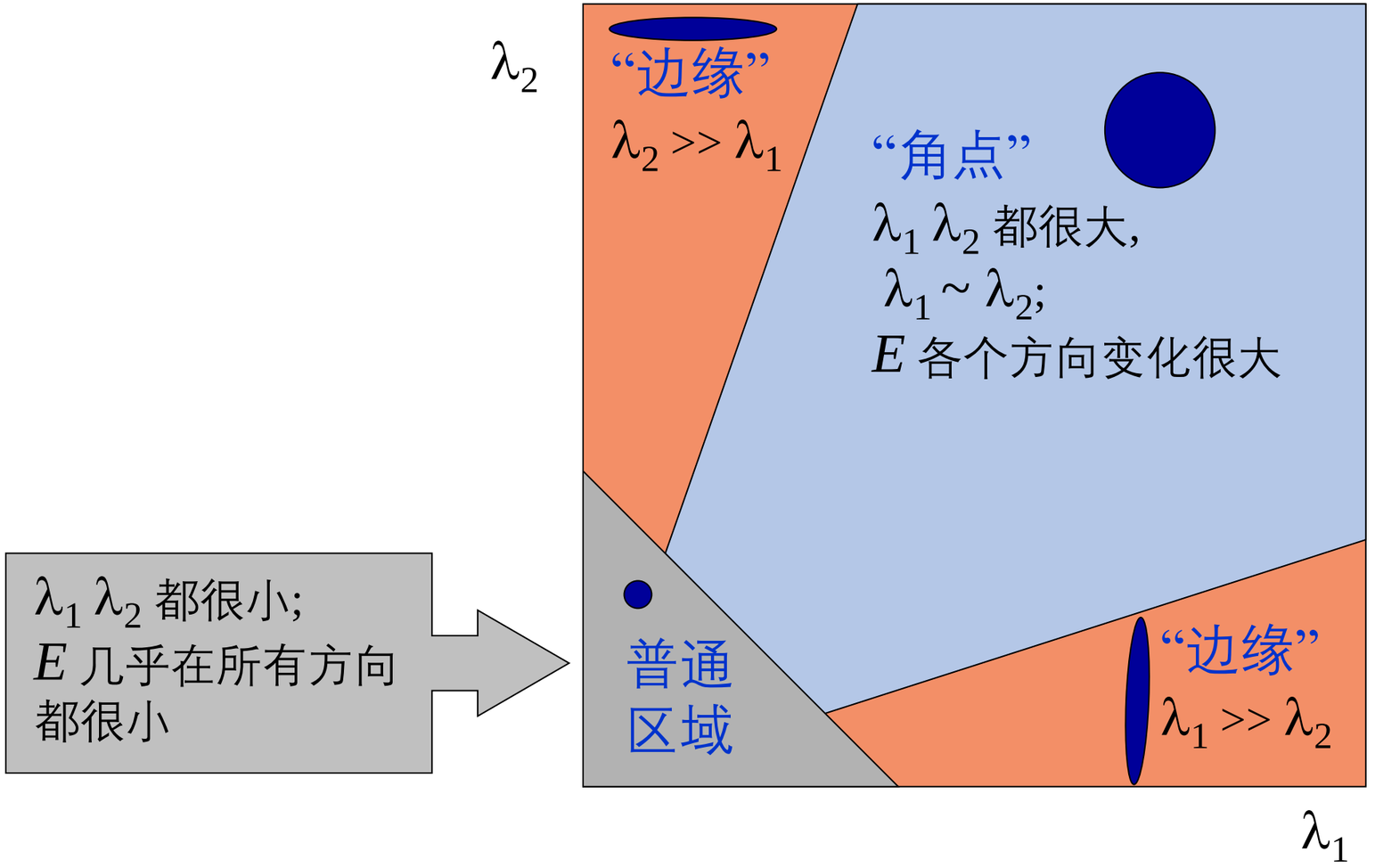
- 我们需要怎样的特征点?

我们需要$E(u,v)$ 在各个方向变化都很大

- $E(u,v)$ 在各个 $[u\ v]$ 上的最小值应该尽量大
- 这个最小值取决于H的最小特征值(λ_{min})



对各种情况分类

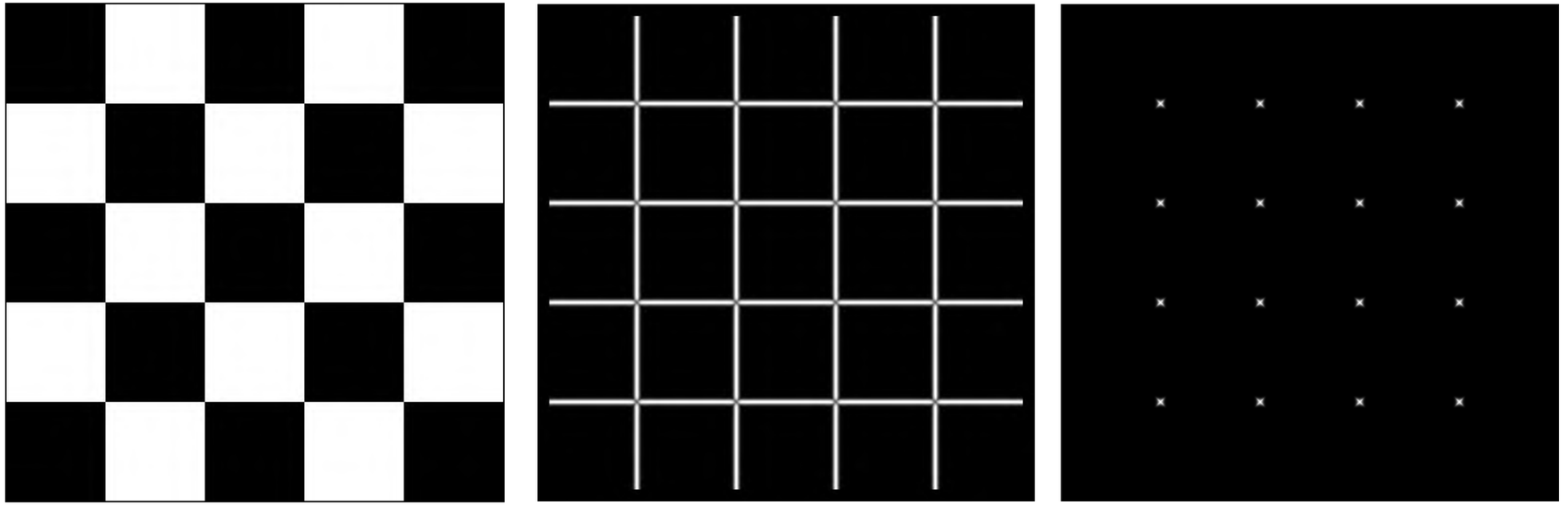

λ_2
“边缘”
$\lambda_2 >> \lambda_1$
“角点”
λ_1 λ_2 都很大，
$\lambda_1 \sim \lambda_2$;
E 各个方向变化很大
λ_1 λ_2 都很小;
E 几乎在所有方向都很小
普通区域
“边缘”
$\lambda_1 >> \lambda_2$
λ_1

Harris 角点检测

步骤:

- 计算每个点的梯度
- 对于每个像素:
  - 根据梯度计算*H*
  - 计算特征值
  - 根据阈值找出角点 (λ_{min} > threshold)
- 选择 λ_{min} 为局部最大值的点为特征点

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

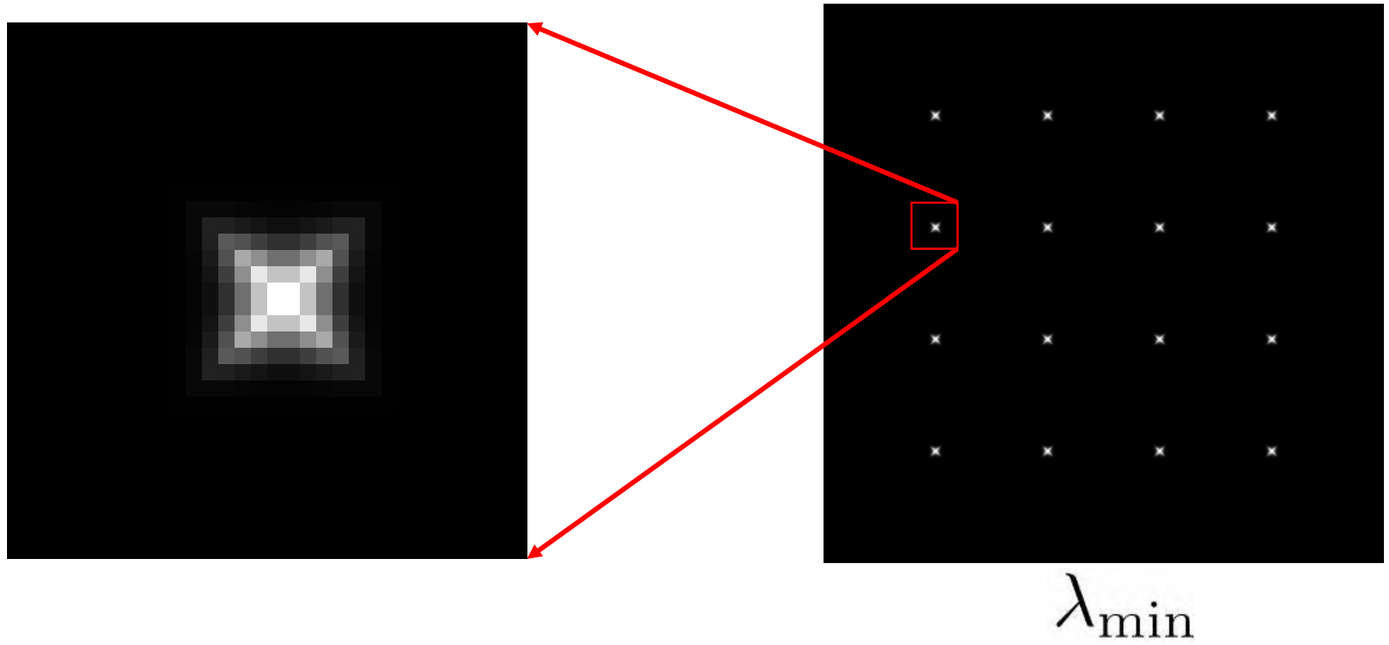


I $\lambda_{\max}$ $\lambda_{\min}$

Harris 角点检测

步骤:

- 计算每个点的梯度
- 对于每个像素:
  - 根据梯度计算H
  - 计算特征值
  - 根据阈值找出角点 (λ_{min} > threshold)
- 选择 λ_{min} 为局部最大值的点为特征点



The Harris operator： Harris 算子

“Harris operator” 特征点提取类似求λ_{min}

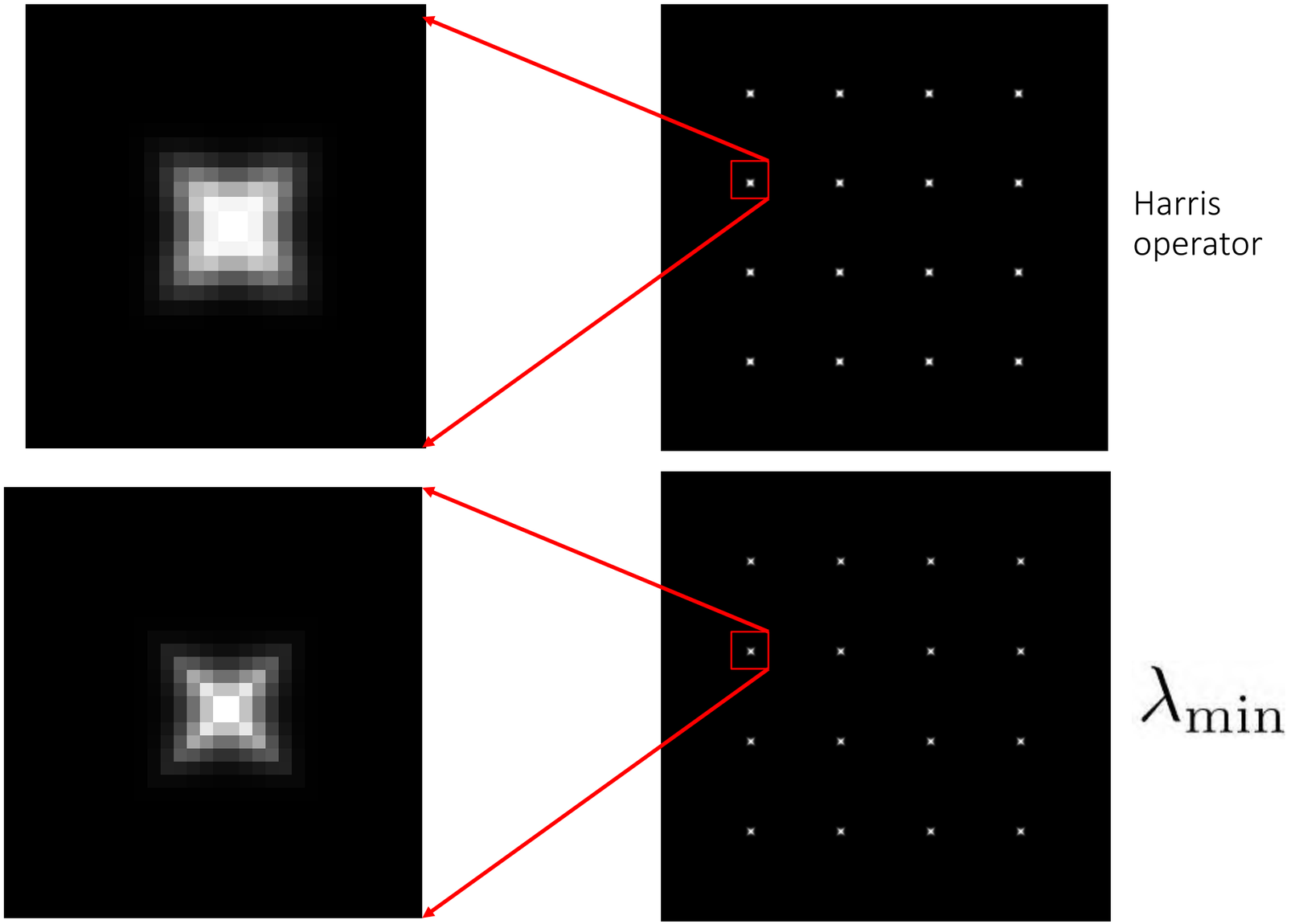
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

- 迹 *trace(H) = h_{11} + h_{22}*
- 与求λ_{min}相似，但是所需计算量更小
- 变式：

$$R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \det(M) - k \cdot \mathrm{tr}(M)^2$$

Harris 算子



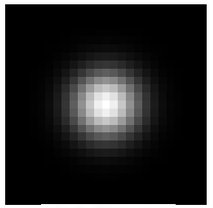
对图像梯度加权

- 实际应用中，我们会对窗口内梯度进行加权处理
  - 离中心越远的像素点权重越低

- 跟什么操作很类似?

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



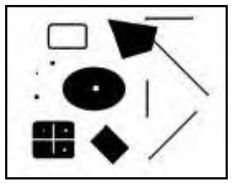
$w_{x,y}$

Harris Detector [Harris88]

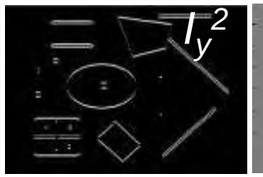
- 构造二阶矩阵

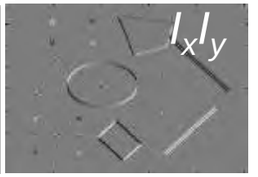
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. 图像梯度



2. 图像梯度的平方





$$\det M = \lambda_1 \lambda_2$$

$$\text{trace}\, M = \lambda_1 + \lambda_2$$

3. 高斯滤波

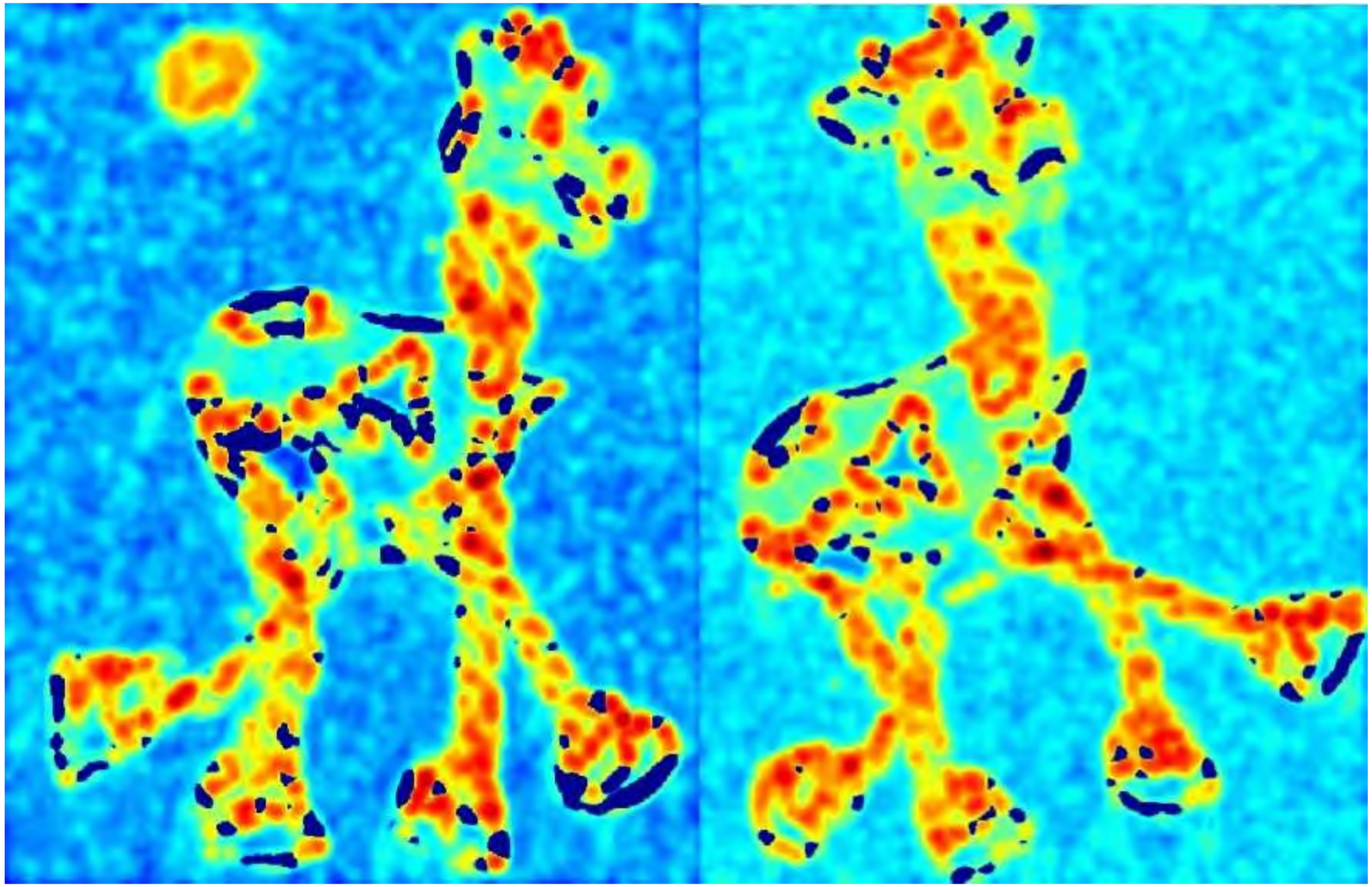
$g(s_I)$

4. 找到最大特征值和最小特征值都足够大的点，或使用Harris算子

5. Non-maxima suppression 非局部最大抑制

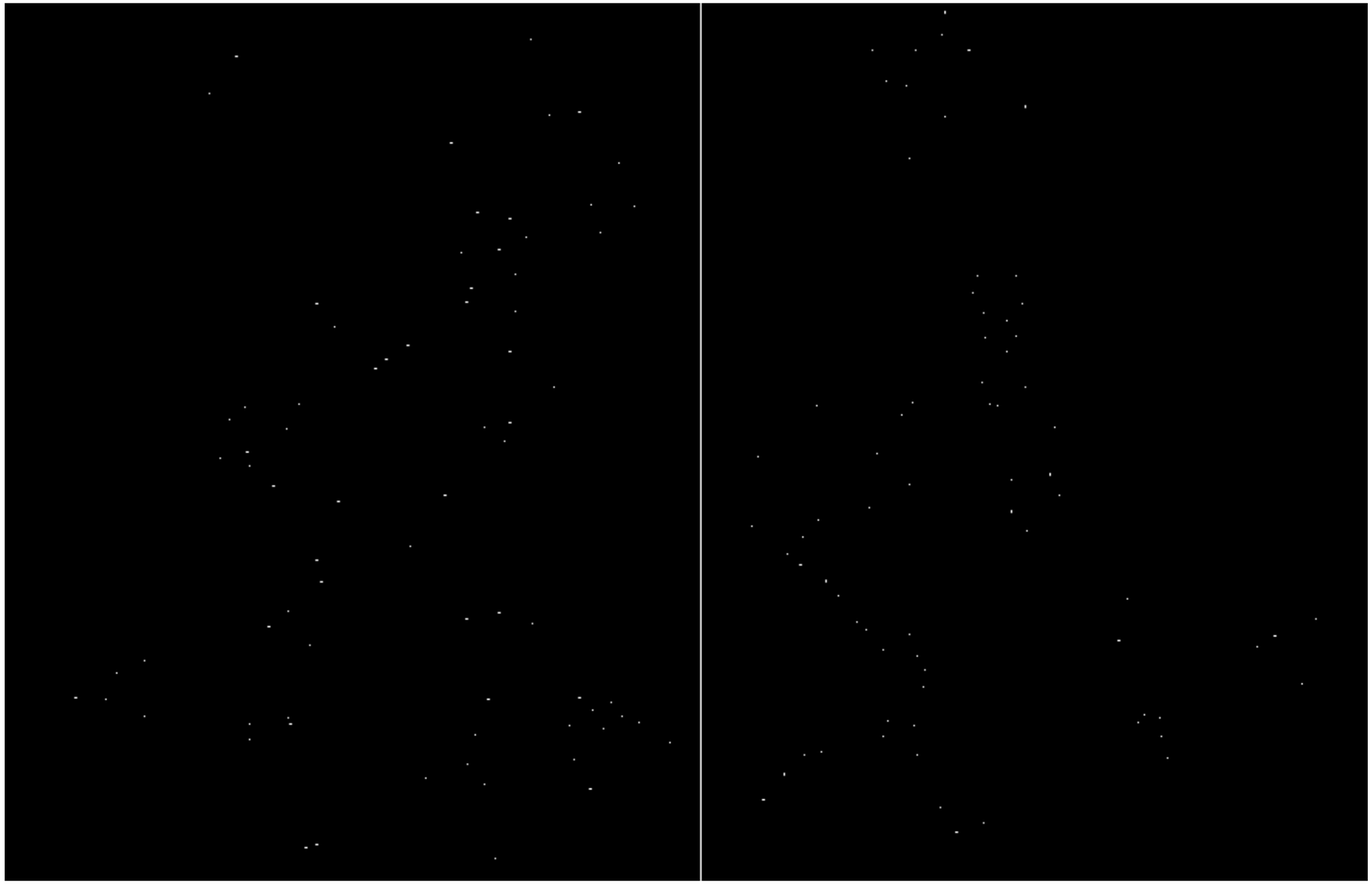
Harris 角点检测

f value (red high, blue low)

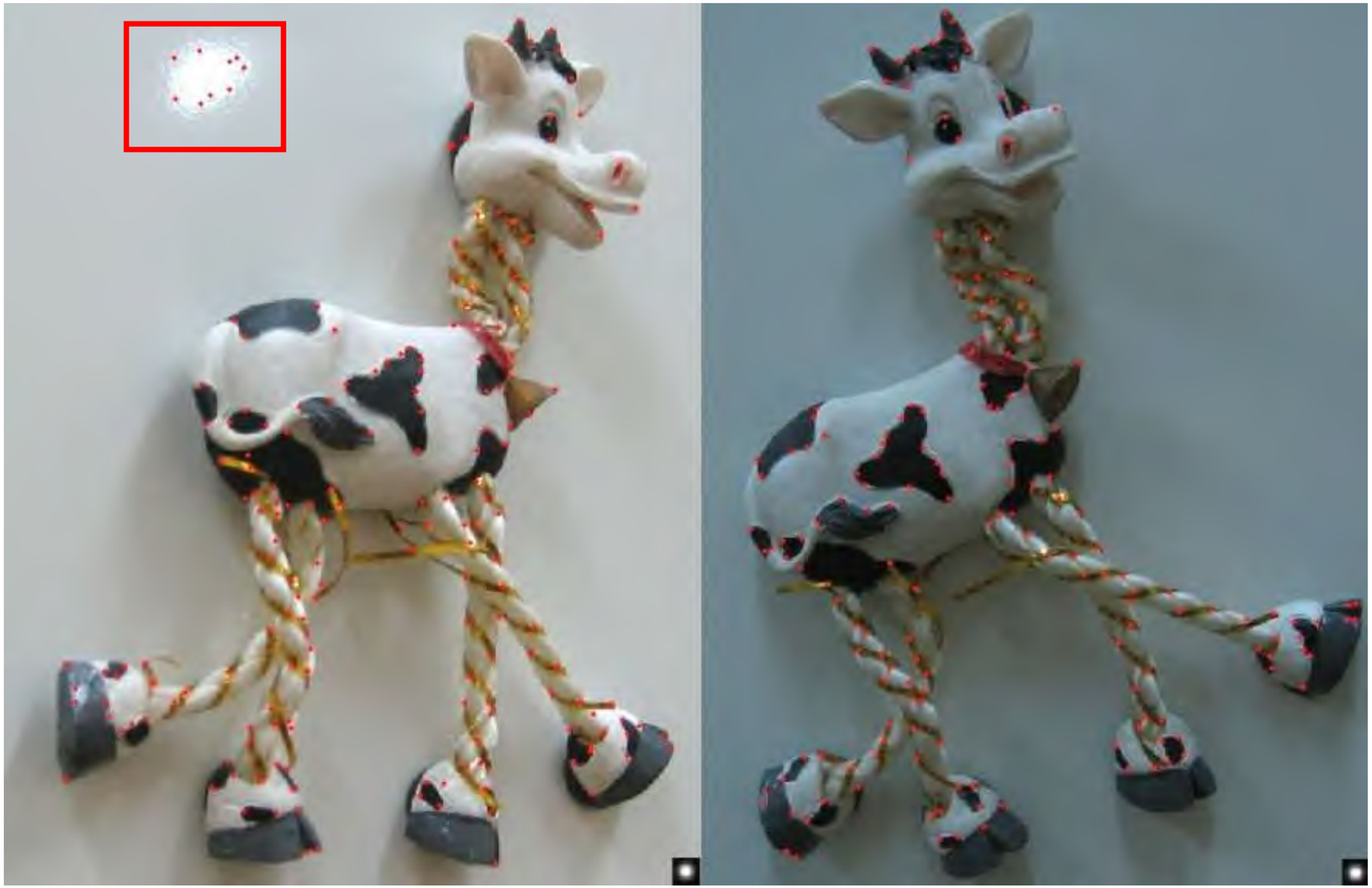


阈值处理 (f > value)

f 局部最大値 (non-max suppression)



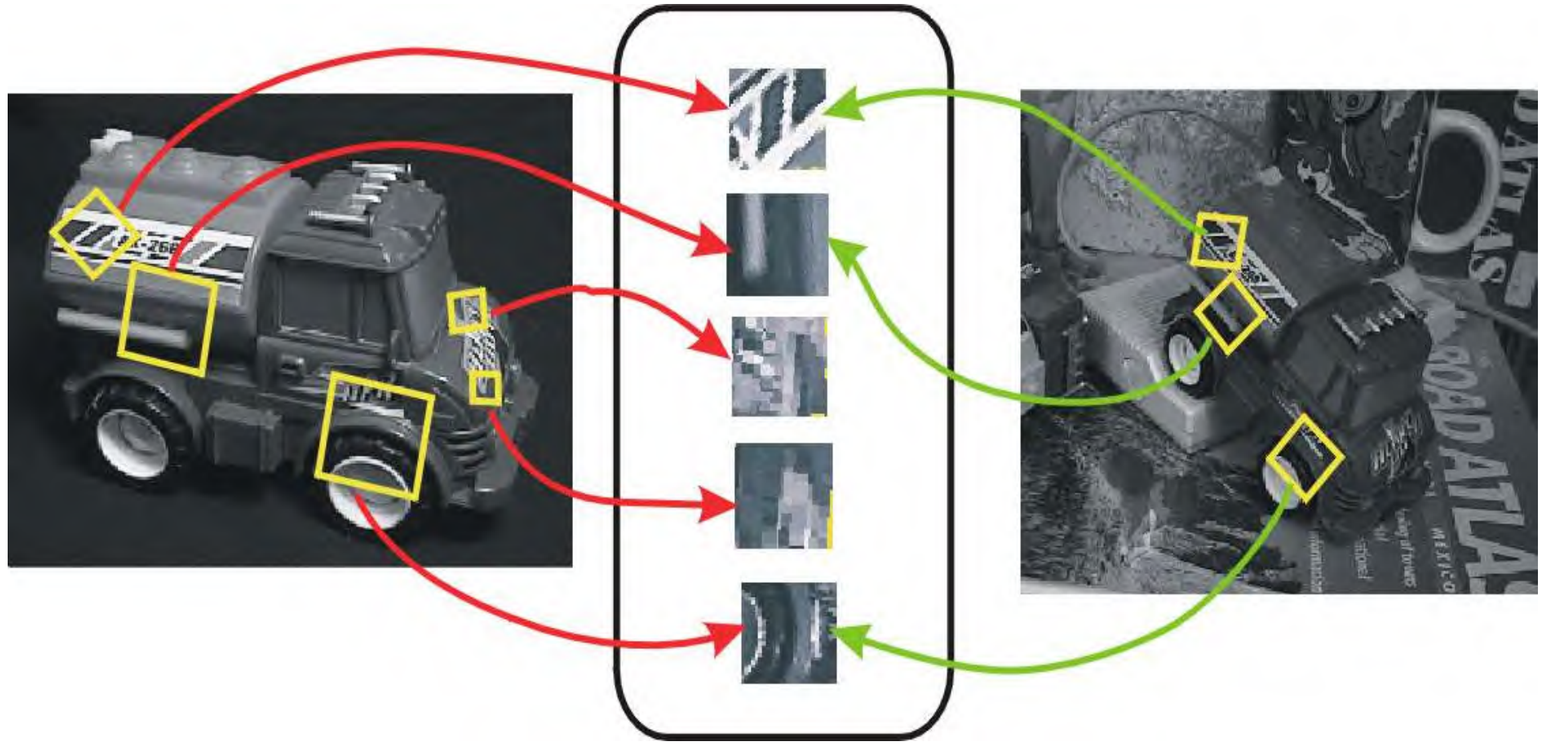
哈里斯特征点(in red): 我们还需要做点什么?



回顾：不变局部特征

找到在图像变换下仍然保持不变的图像特征

- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



特征提取器

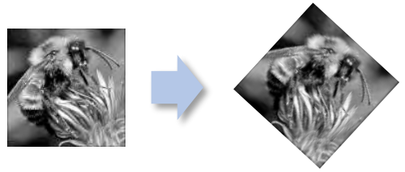
图像变换

找到在图像变换下仍然保持不变的图像特征

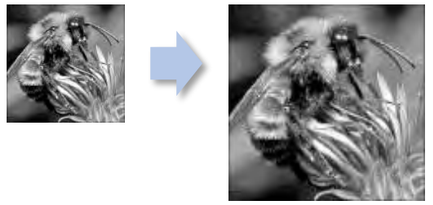
- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度, ...

- 几何层面

旋转



尺度



- 光学层面

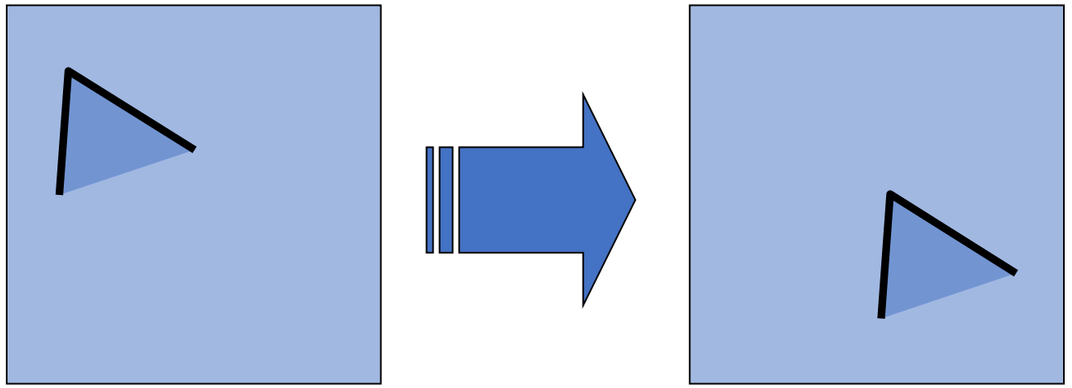
强度变换

关键点要在这些图像变换中保持不变！

不变性与等价性

- 我们希望特征点在不同的光学变换下不变（*invariant*），在不同的几何变换下等价（*equivariant*）
  - **不变性**: 图像强度变了但是关键点的位置不变
  - **等价性**: 图像如果有几何层面的变化，则特征也会得到同样的几何变化

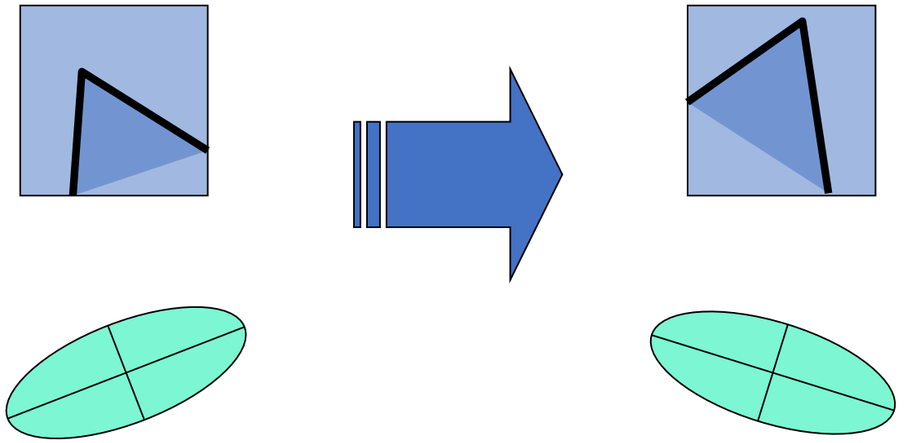
平移不变



- 窗口信息不变，在变换后仍然可以被检测到

特征点位置在变换后等价

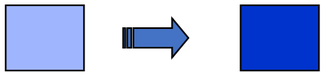
旋转不变



虽然图像做了旋转变换，但H对应椭圆公式形状不变

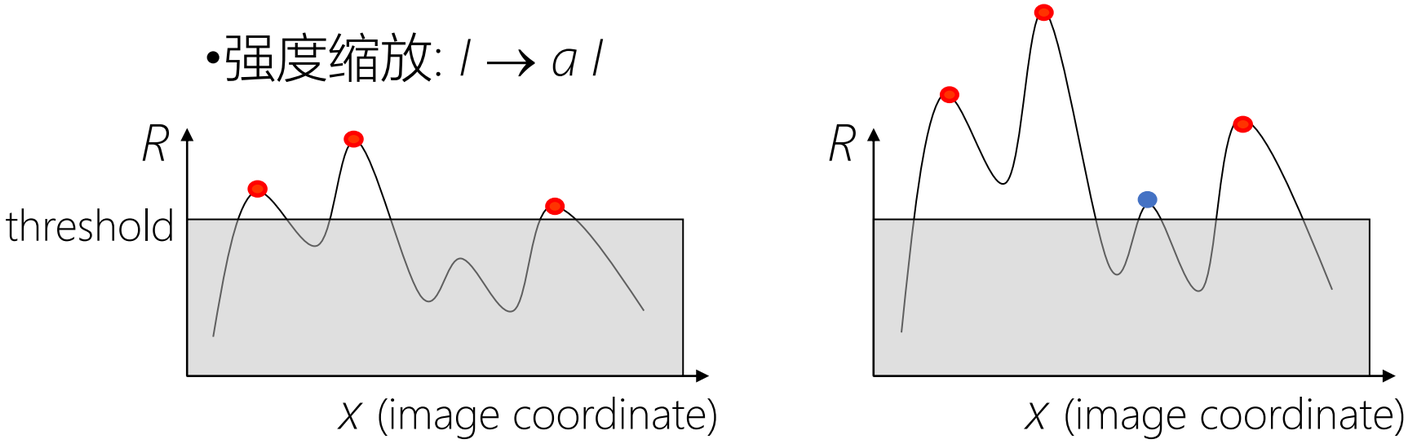
特征点位置在变换后等价

强度不变



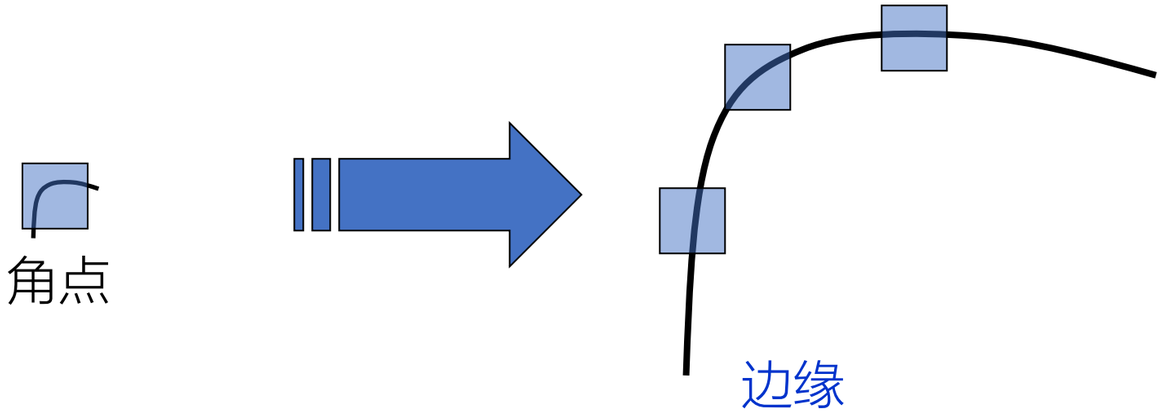
$I \rightarrow a\,I + b$

- 图像梯度→ 强度差 $I \rightarrow I + b$
- 强度缩放: $I \rightarrow a\,I$



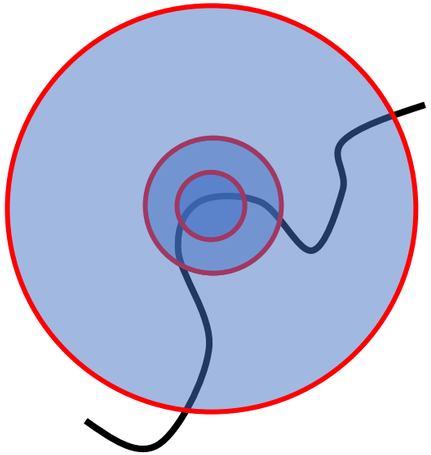
采取阈值法，对于强度变化部分不变

尺度（缩放、强度）不变



如何确保对缩放的不变性？

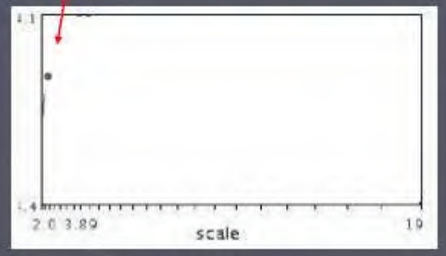
尺度不变性



核心思想：找到响应最大的缩放比例

Automatic scale selection

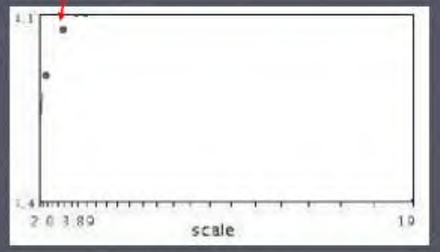
Lindeberg et al., 1996



$f(I_{i_1 \ldots i_m}(x,\sigma))$

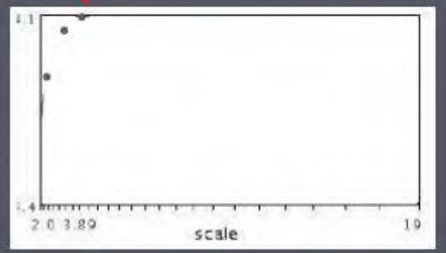
Slide from Tinne Tuytelaars

Automatic scale selection



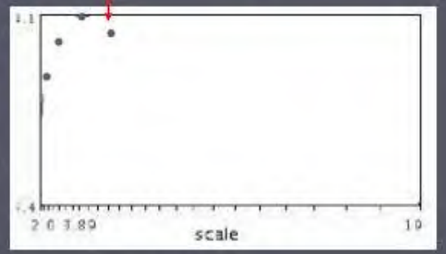
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



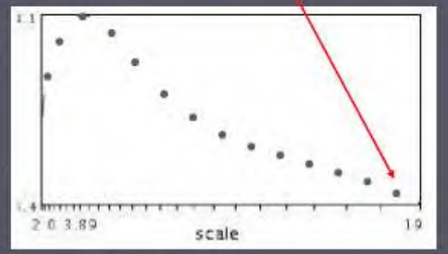
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



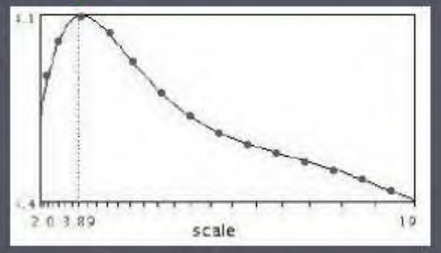
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



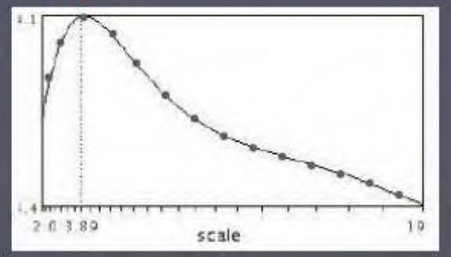
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection

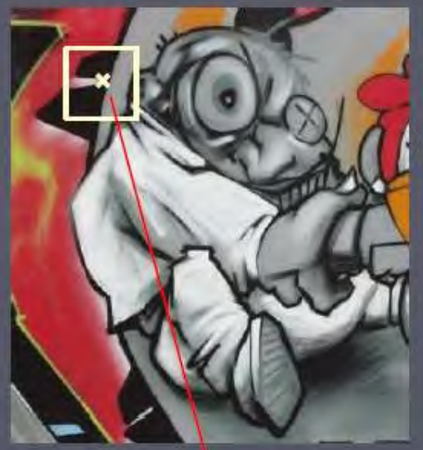


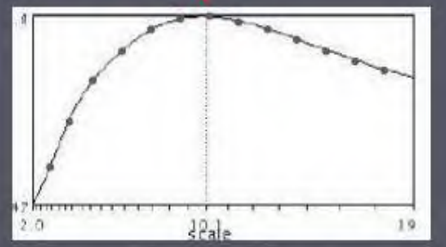
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

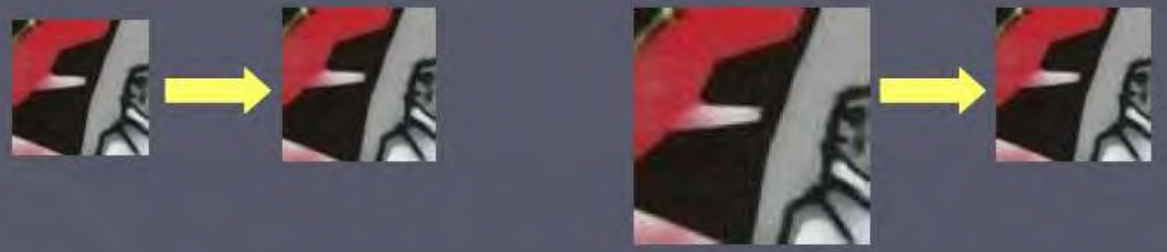




$f(I_{i_1 \ldots i_m}(x',\sigma'))$

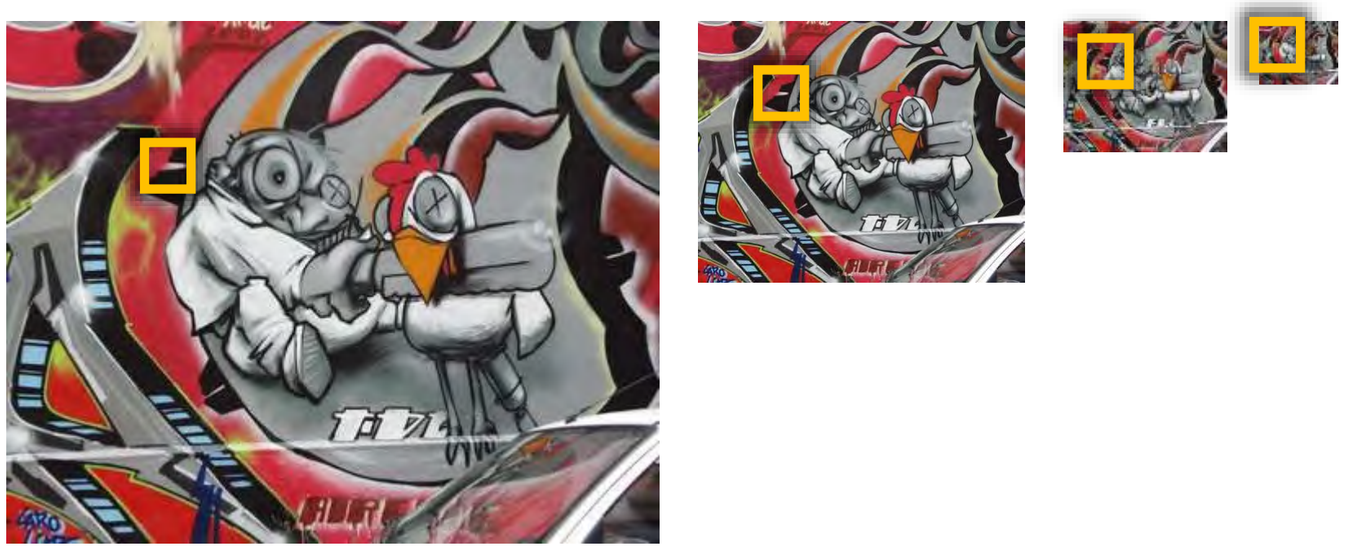
Automatic scale selection

Normalize: rescale to fixed size



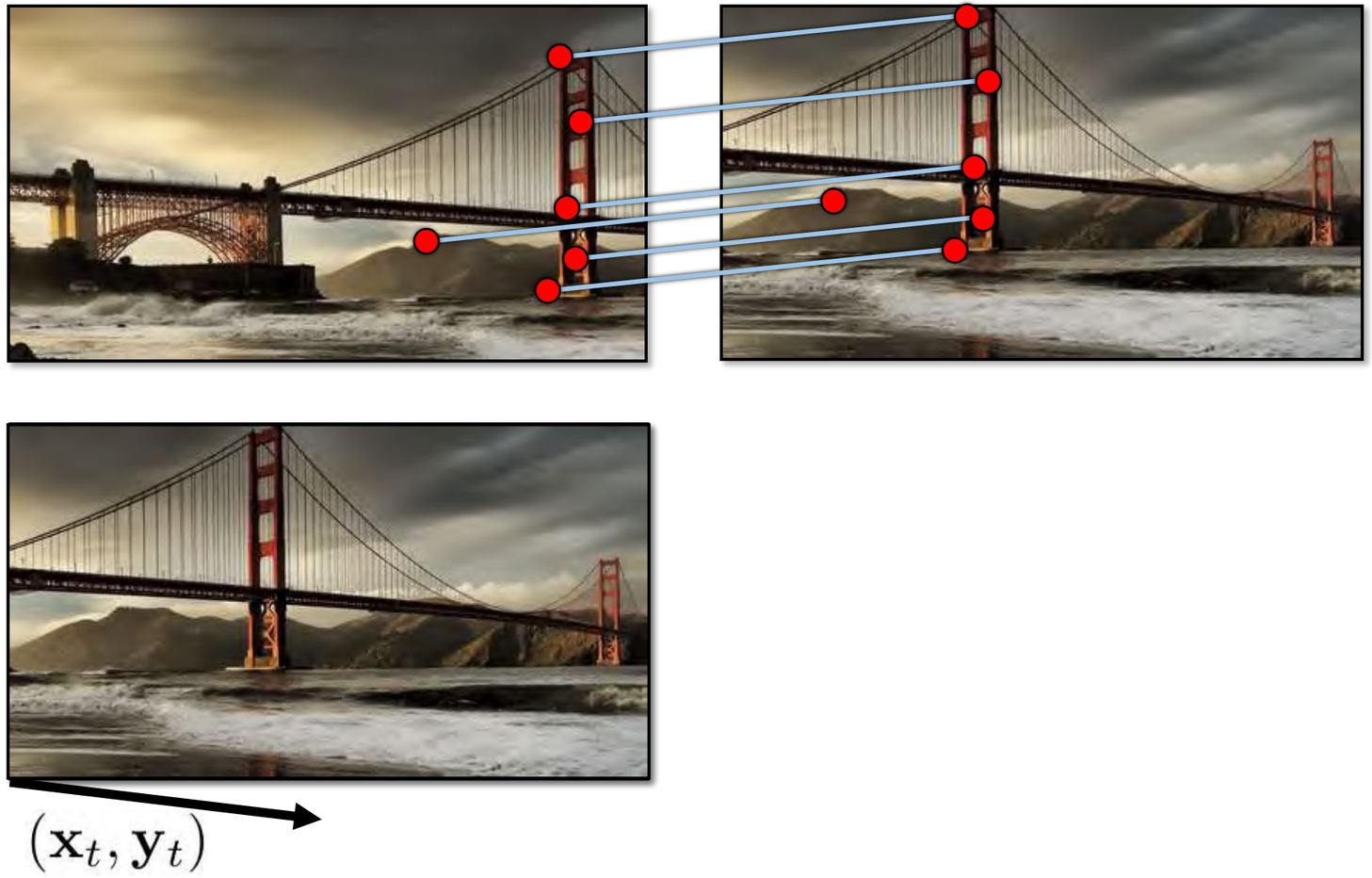
实现

- 用同样的窗口大小，但缩放图像
- Gaussian Pyramid



回顾——角点检测

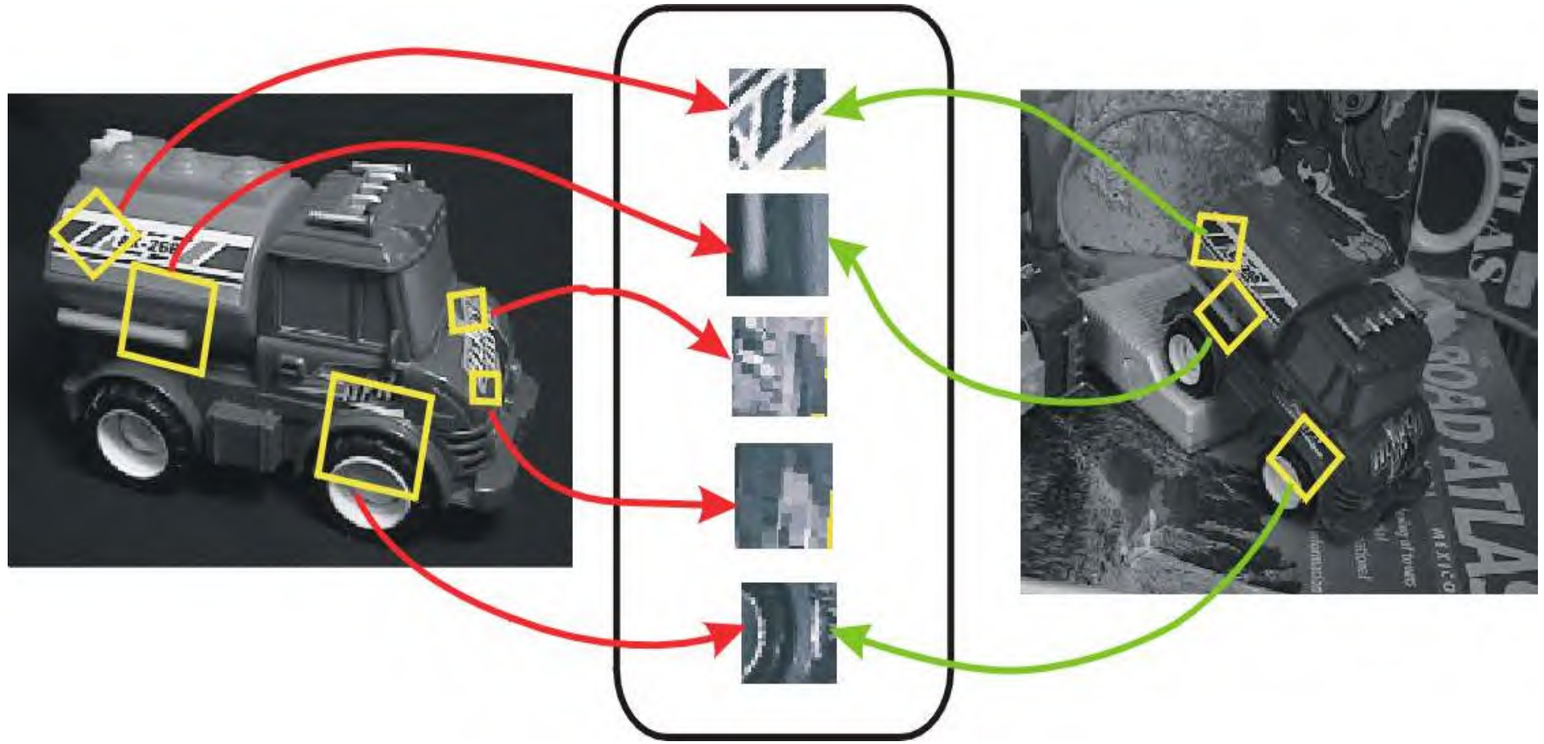
引入：全景图中的图像拼接



不变局部特征

找到在图像变换下仍然保持不变的图像特征

- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



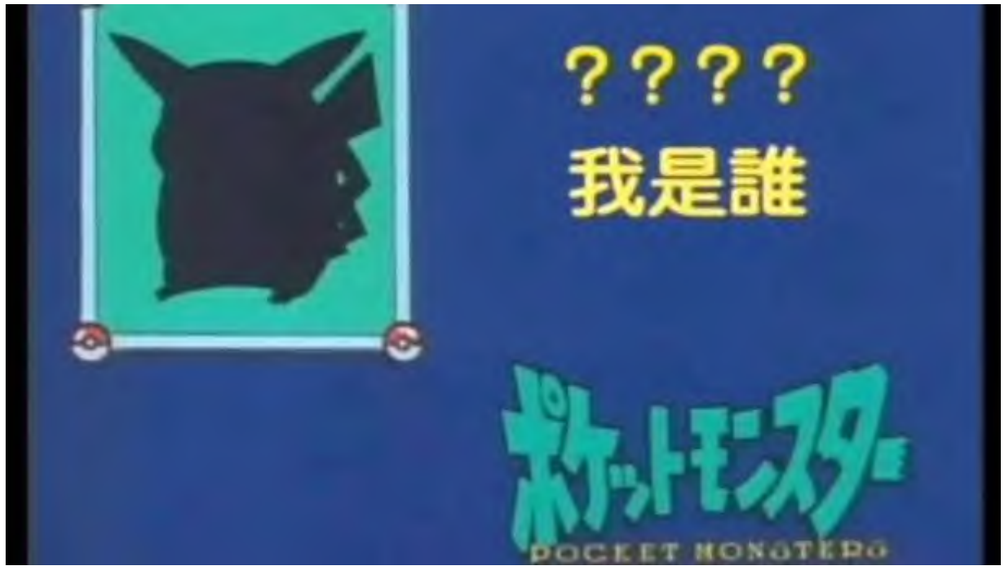
特征提取器

Uniqueness：唯一性

特征点在图像或图像集合中具有**独特**的外观或属性

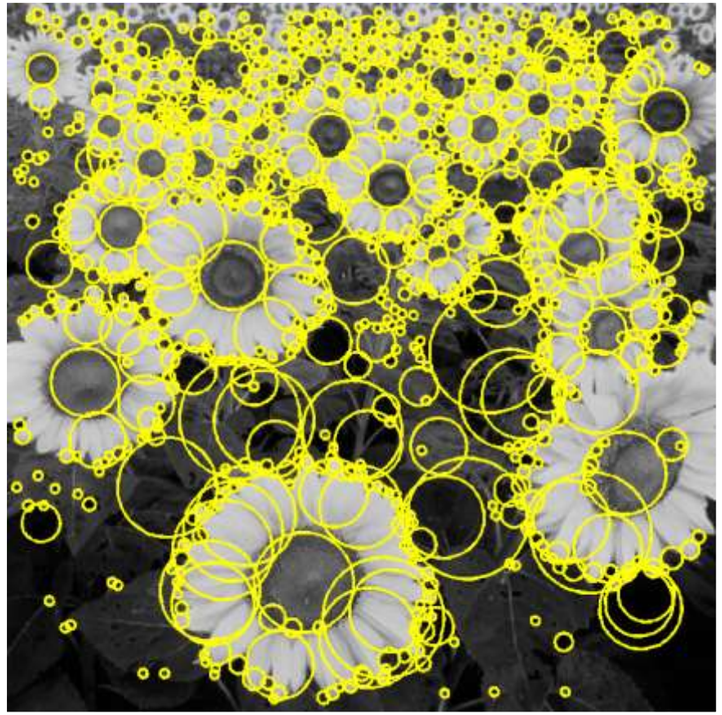
- 使其容易与其他特征点区分开来，不会产生混淆

怎样定义“独特”？



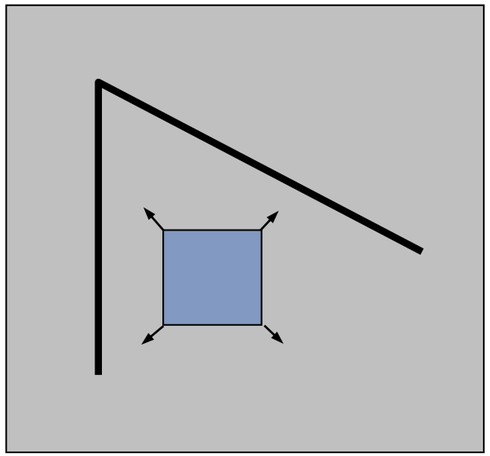
特征提取：角点

- 角点（Corners）： 角点是图像中具有明显边缘转折的像素点。

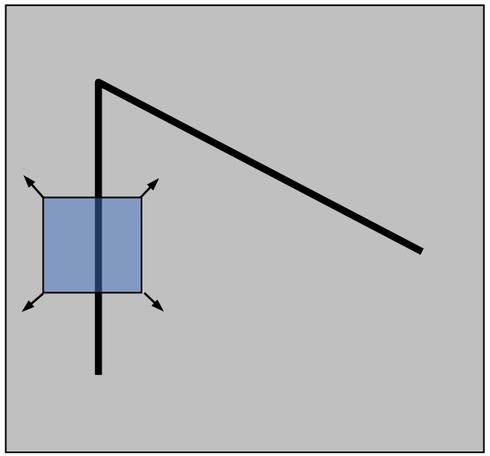


衡量局部的“唯一性”

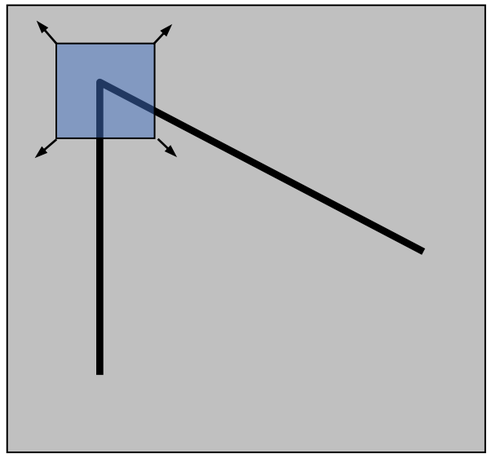
- 我们从窗口内提取特征（图像梯度、DoG）
- 当窗口向四周滑动，特征怎样变化?



“flat” 普通区域:
所有方向都没有过多变化



“edge” 边缘:
边缘区域没有变化



“corner” 角点:
在各个方向都有交大的变化

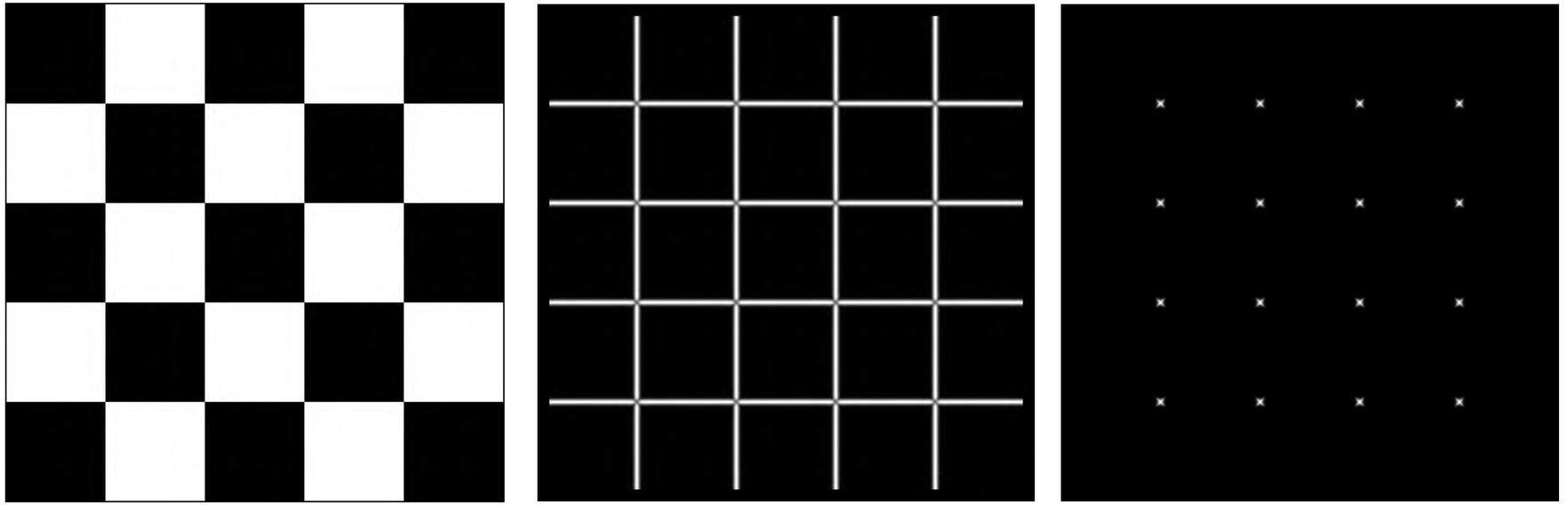
Credit: S. Seitz, D. Frolova, D. Simakov

Harris 角点检测

步骤:

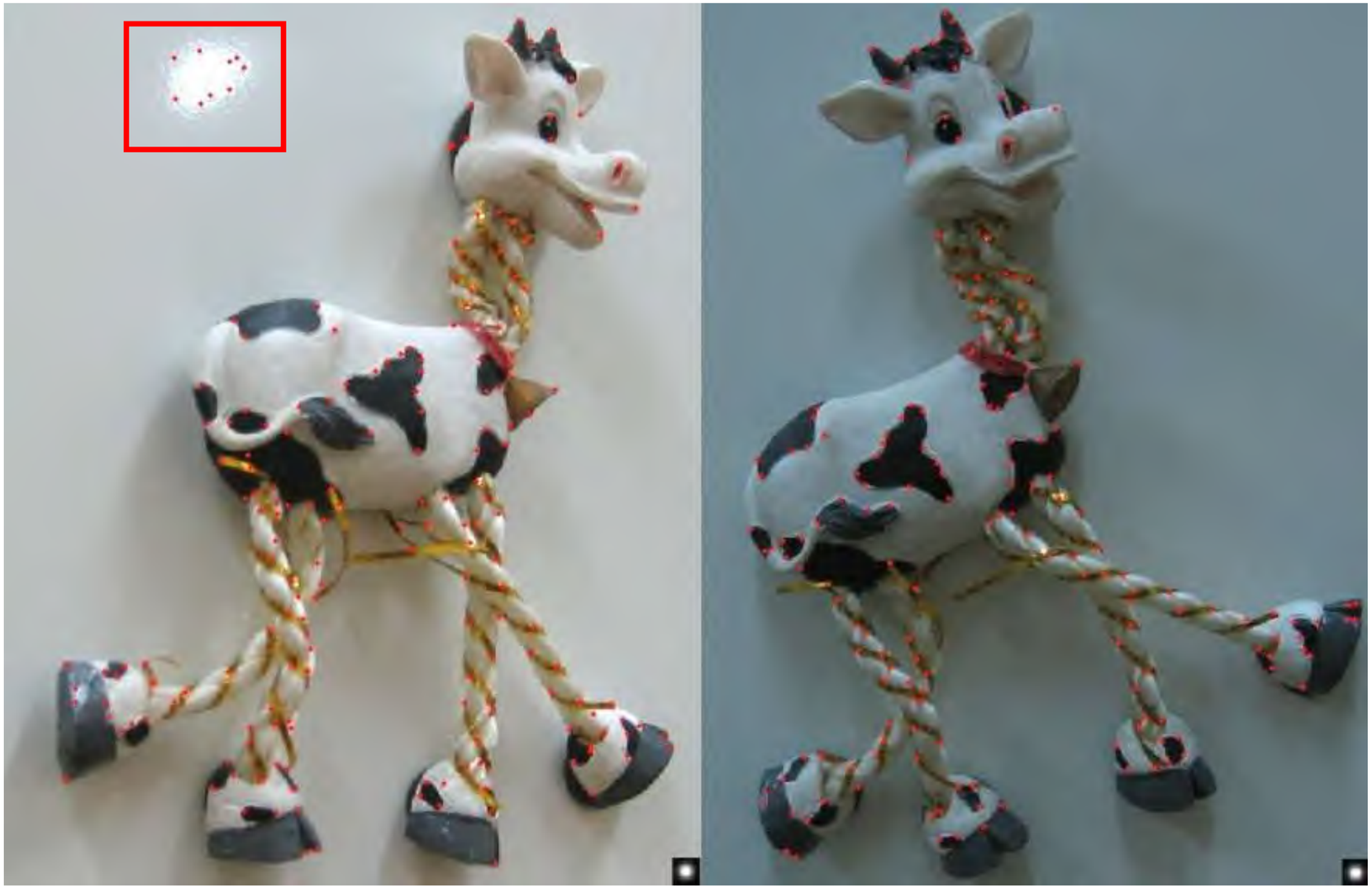
- 计算每个点的梯度
- 对于每个像素:
  - 根据梯度计算*H*
  - 计算特征值
  - 根据阈值找出角点 (λ_{min} > threshold)
- 选择 λ_{min} 为局部最大值的点为特征点

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

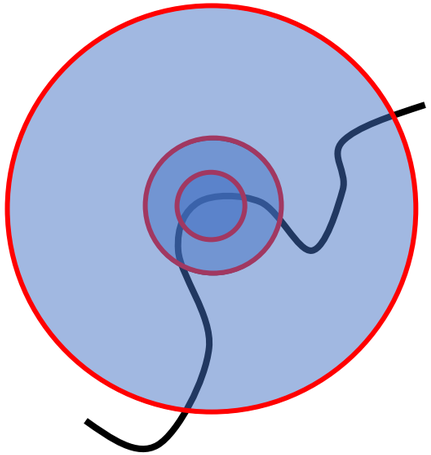


I $\lambda_{\max}$ $\lambda_{\min}$

哈里斯特征点(in red): 我们还需要做点什么?

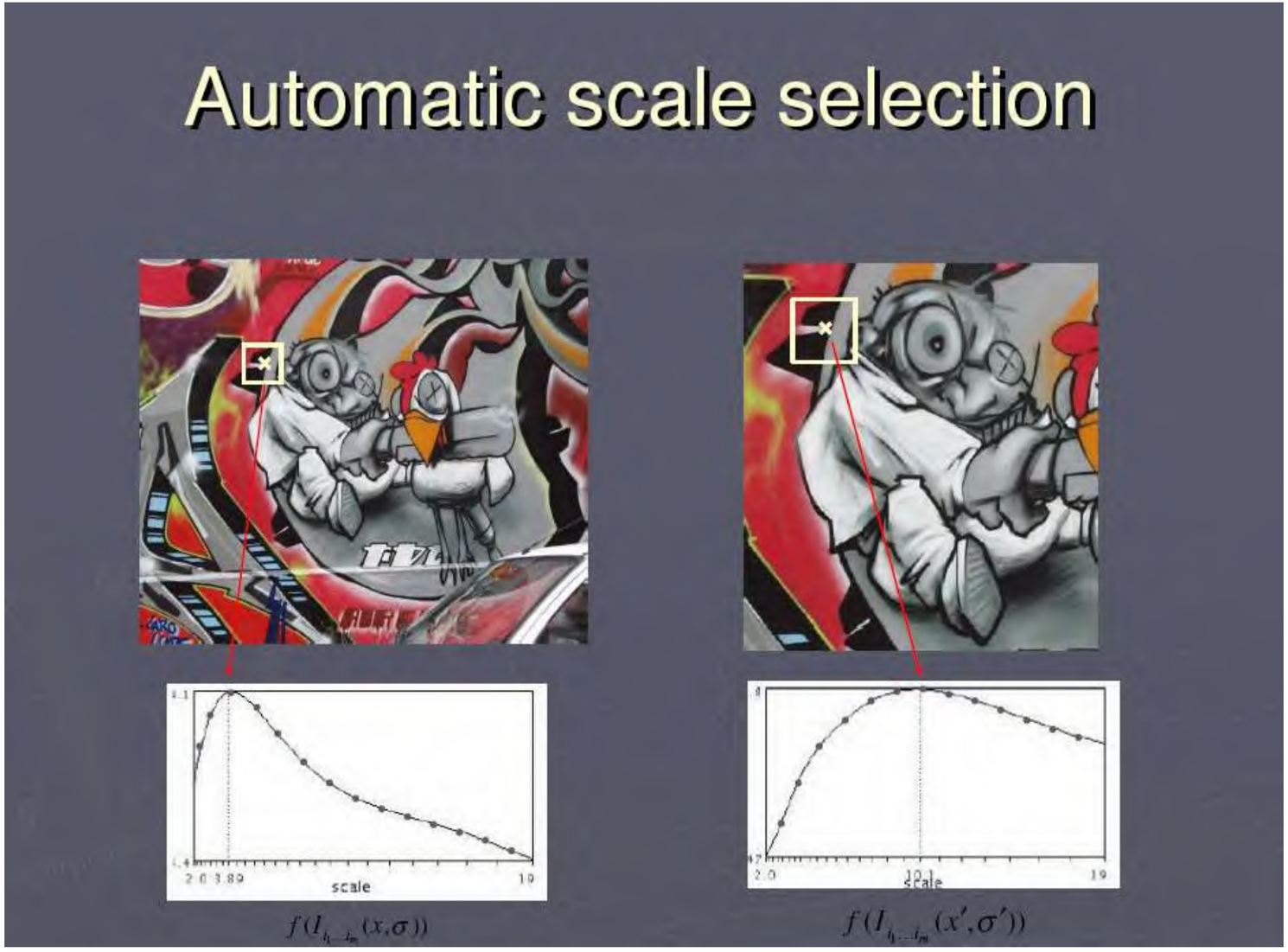


尺度不变性



核心思想：找到响应最大的缩放比例

特征尺度

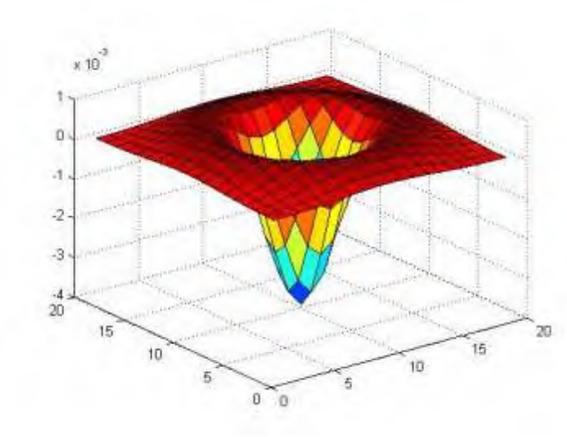

Automatic scale selection
scale
scale
$f(I_{i_1 \ldots i_m}(x,\sigma))$
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

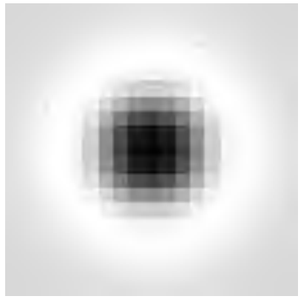
特征提取：斑点

- 斑点（Blobs）：斑点是图像中的连续区域，其像素值具有明显的局部极值。

计算斑点的方式

- The *Laplacian （二阶导数） of Gaussian* (*LoG*)





高亮点位于图像中心的图像，而图像的边缘和角点等特征会被突出显示

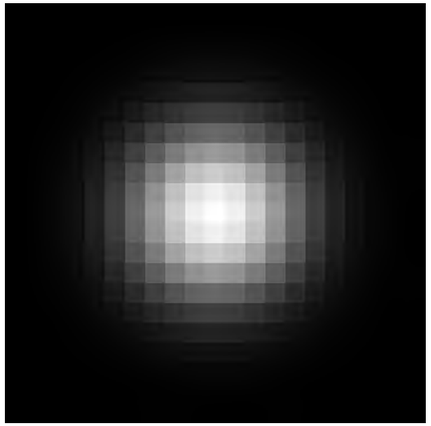
LoG 对旋转保证不变性

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

一个区域内高响应，周围低响应

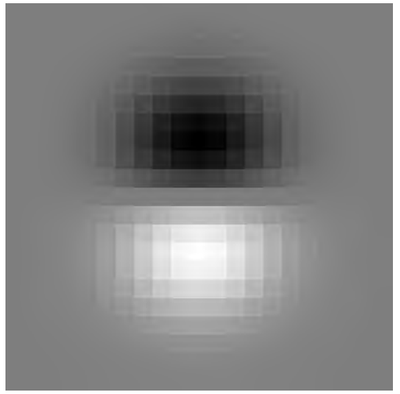
Gaussian Derivatives

Gaussian

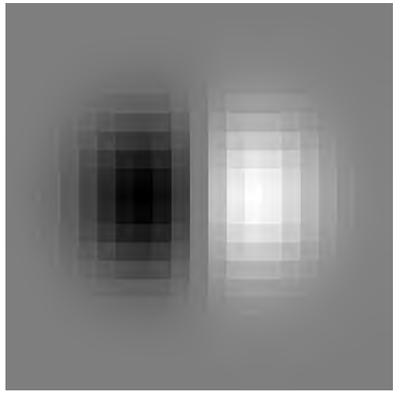


1st Deriv

$$\frac{\partial}{\partial y} g$$

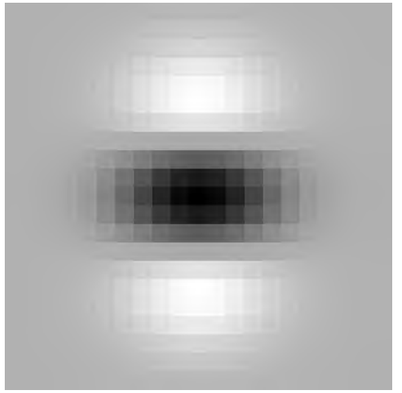


$$\frac{\partial}{\partial x} g$$

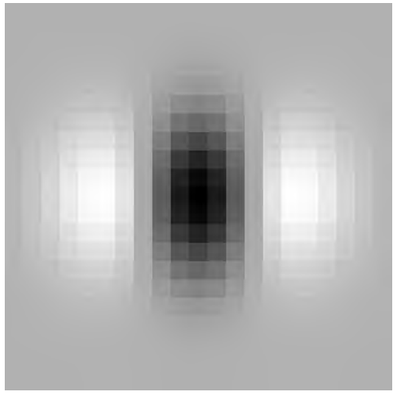


2nd Deriv

$$\frac{\partial^2}{\partial^2 y} g$$



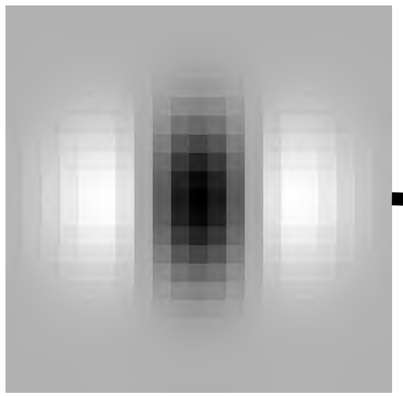
$$\frac{\partial^2}{\partial^2 x} g$$



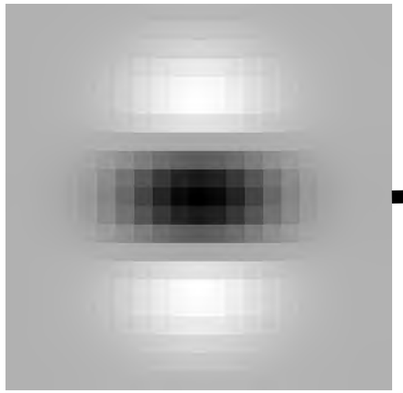
Laplacian of Gaussian (LoG)

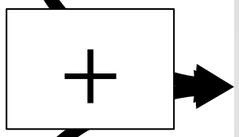
$$\frac{\partial^2}{\partial^2 x} g + \frac{\partial^2}{\partial^2 y} g$$

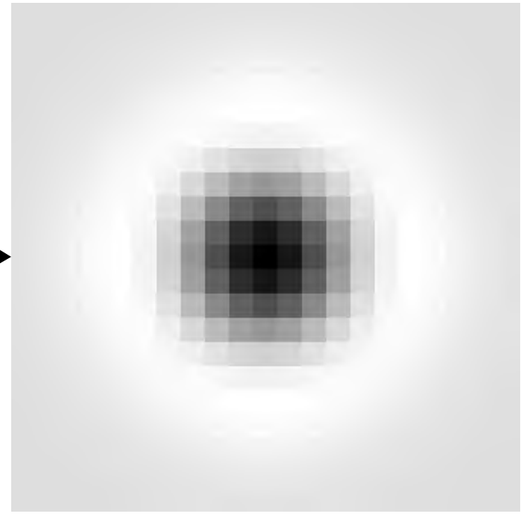
$$\frac{\partial^2}{\partial^2 y} g$$



$$\frac{\partial^2}{\partial^2 x} g$$





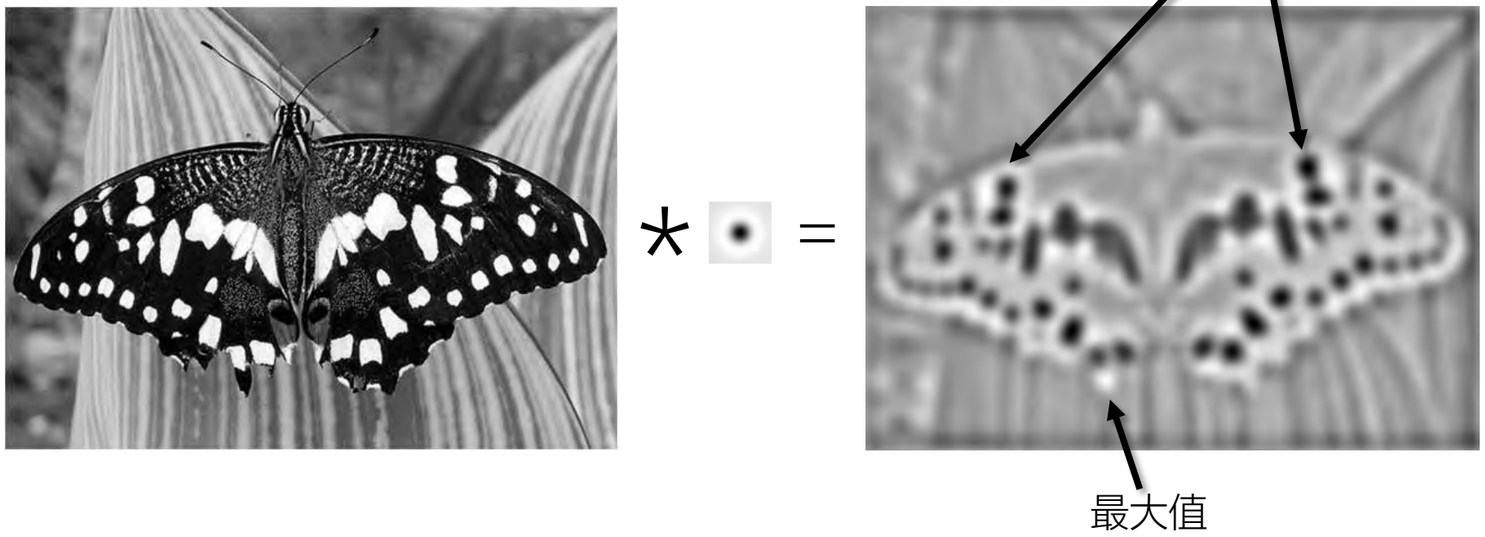


Slight detail: for technical reasons, you need to scale the Laplacian of Gaussian if you want to compare across sigmas.

$$\nabla^2_{norm} = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial^2 y} g \right)$$

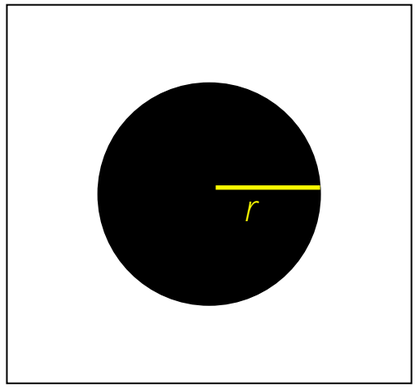
Laplacian of Gaussian （LoG）

- “Blob” detector 斑点检测

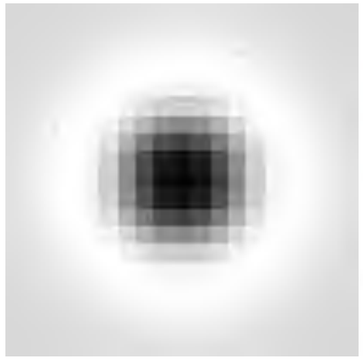


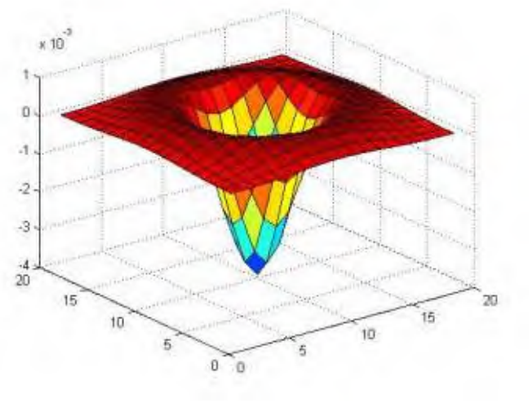
- 找到LoG 算子在空间和尺度上的最大值和最小值

特征尺度



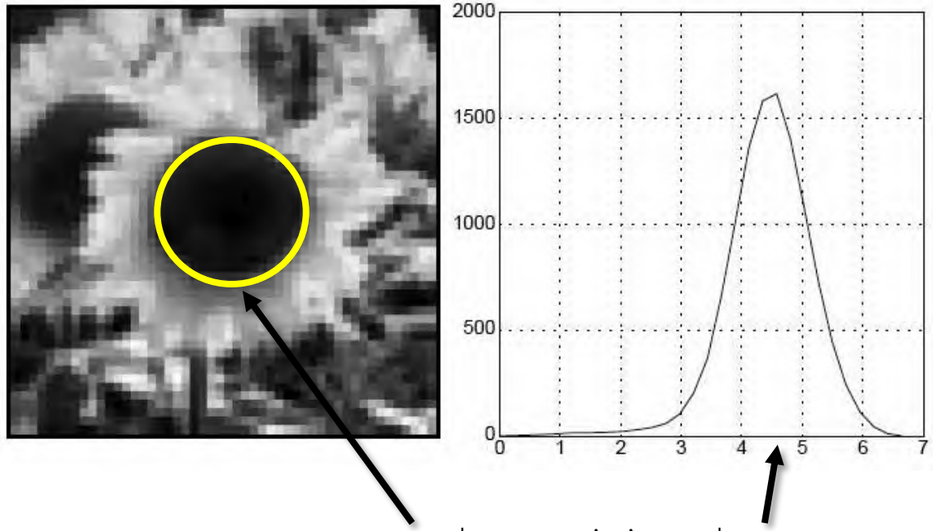
image





Laplacian

与角点相同，计算响应最大的尺度

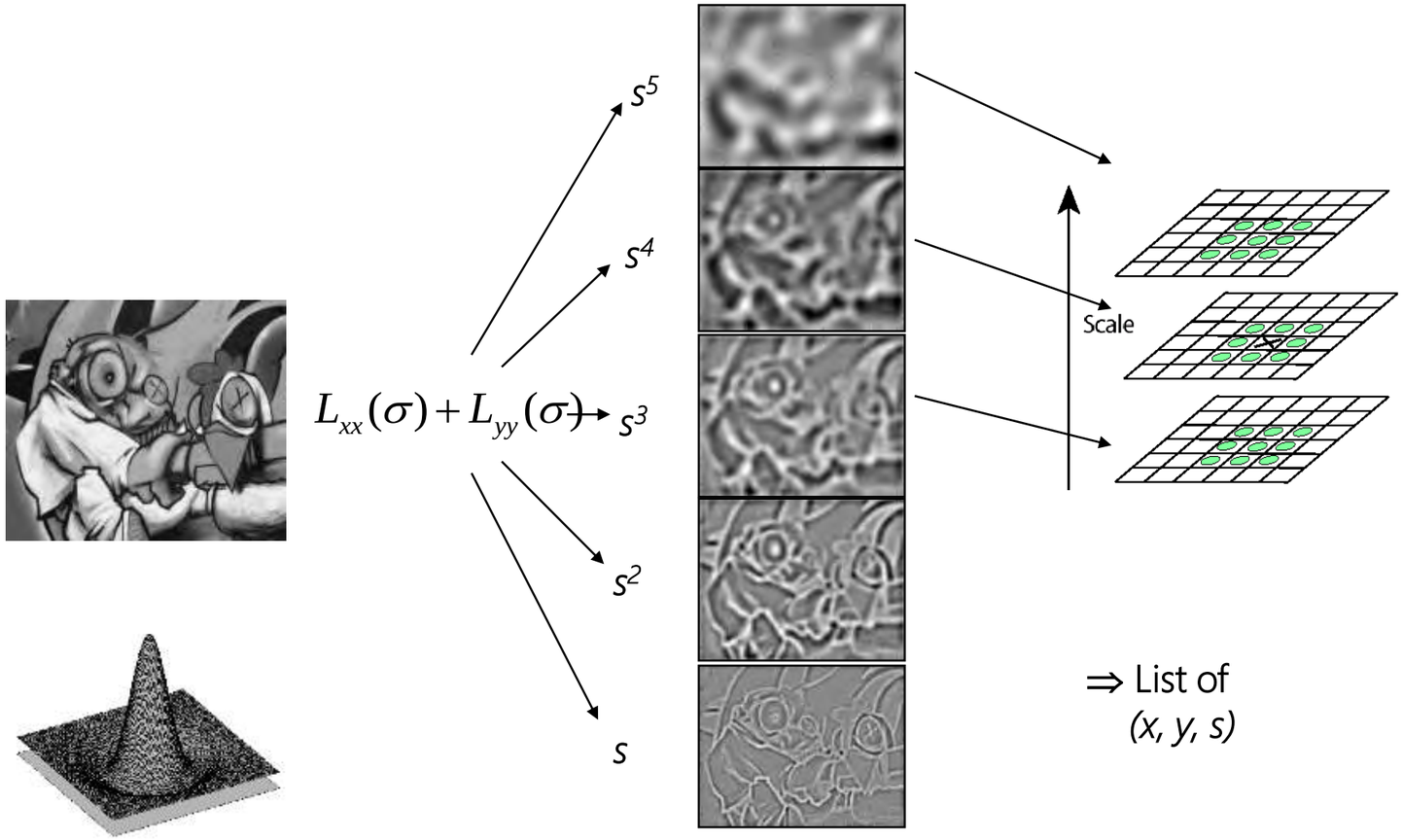


characteristic scale

- 在什么尺度下我们能在Laplacian算子下得到最大值?

T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

斑点检测：多尺度

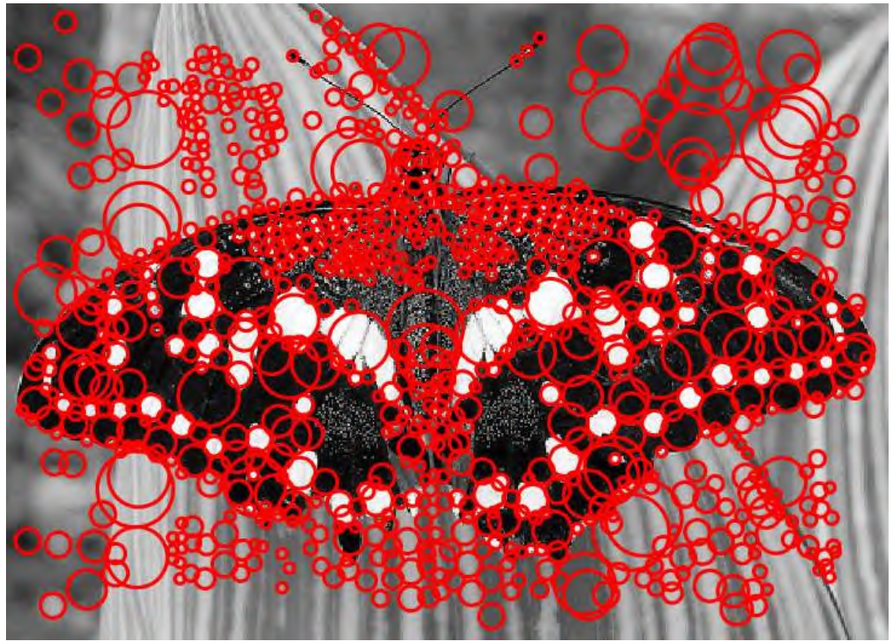


斑点检测

斑点检测：不同尺度

sigma = 11.9912

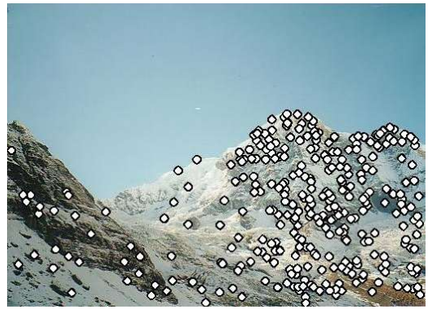
斑点检测：最终效果



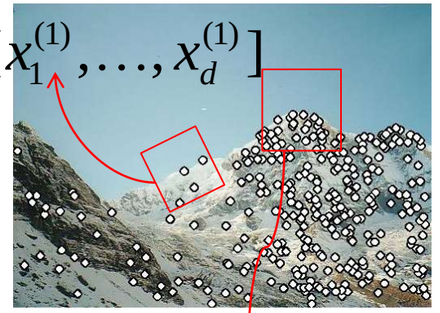
特征提取和匹配

基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点

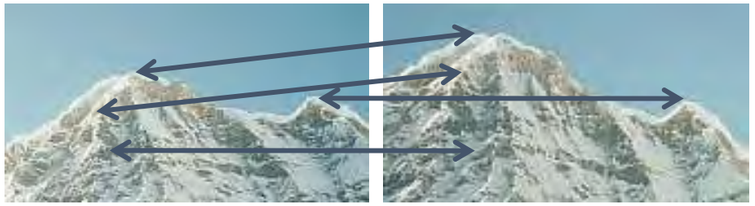


2) **解释 Description: 在关键点周围提取特征**



$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

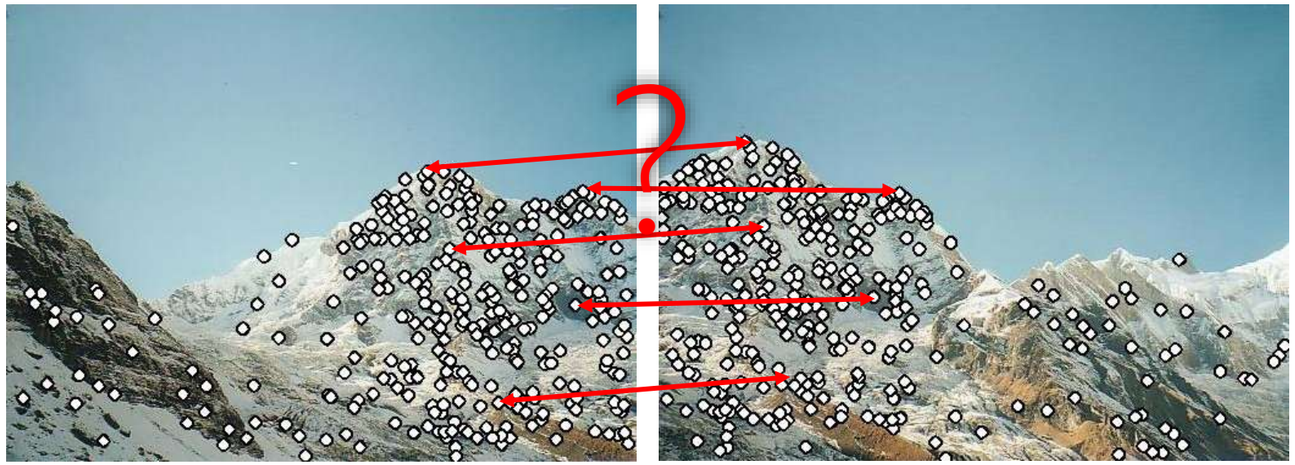
3) 匹配 Matching: 根据两个视角下的特征进行匹配



Kristen Grauman

特征提取器

我们已经知道怎么检测关键点了
下一个问题: **怎么提取关键点信息?**



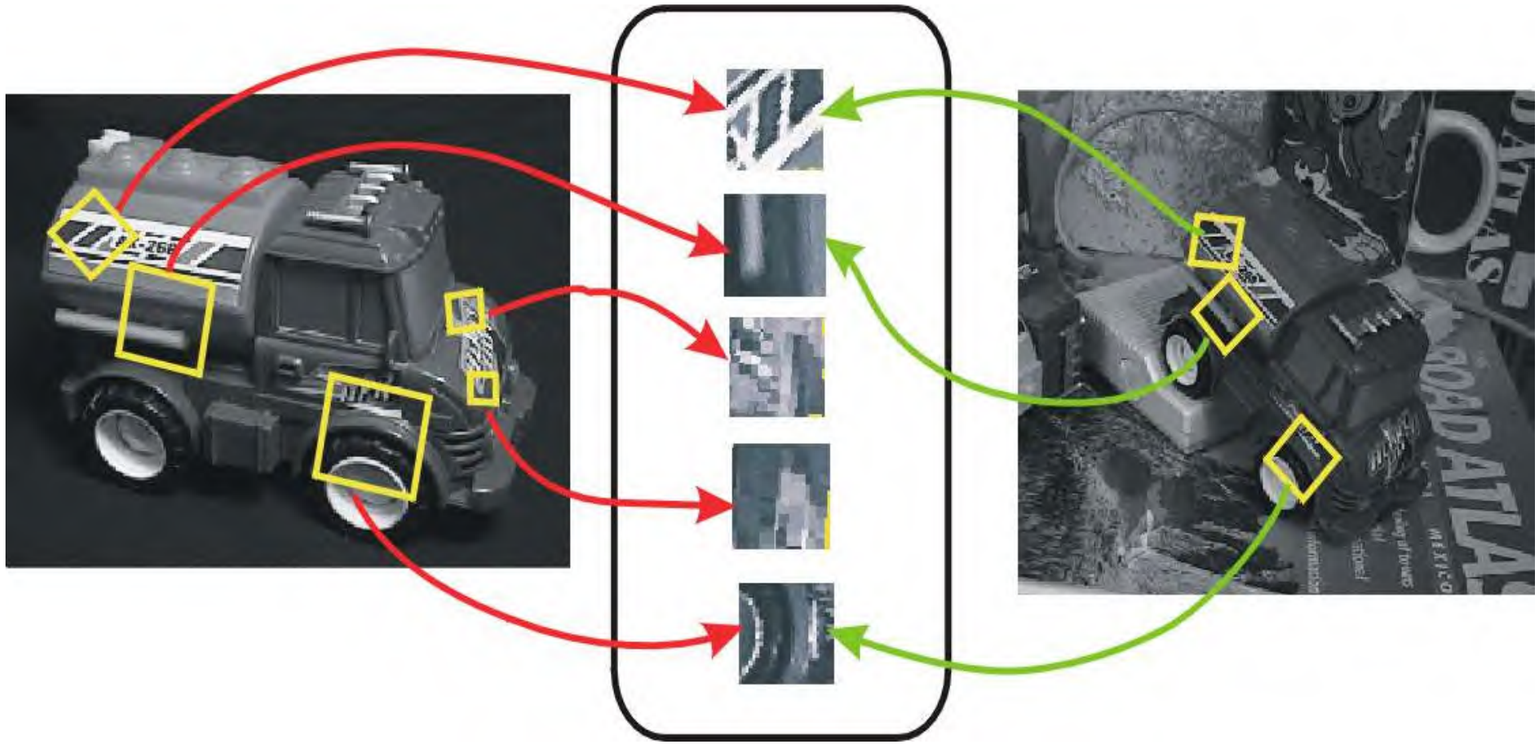
Answer: 使用特征提取器（特征描述子，descriptor）
怎么做?

1. 使用关键点周围的像素
2. 使用如SIFT等保证不变性的特征提取器

回顾：不变局部特征

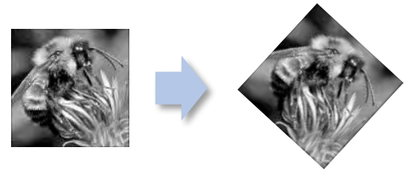
找到在图像变换下仍然保持不变的图像特征

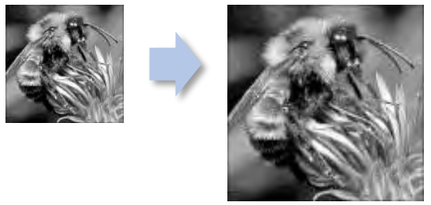
- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



特征提取器

- 几何层面

旋转 

尺度 

- 光学层面

强度变换 

描述特征

Image – 40

1/2 size, rot. 45°
Lightened+40

全图

100x100 crop
at Glasses

特征提取器的旋转不变性

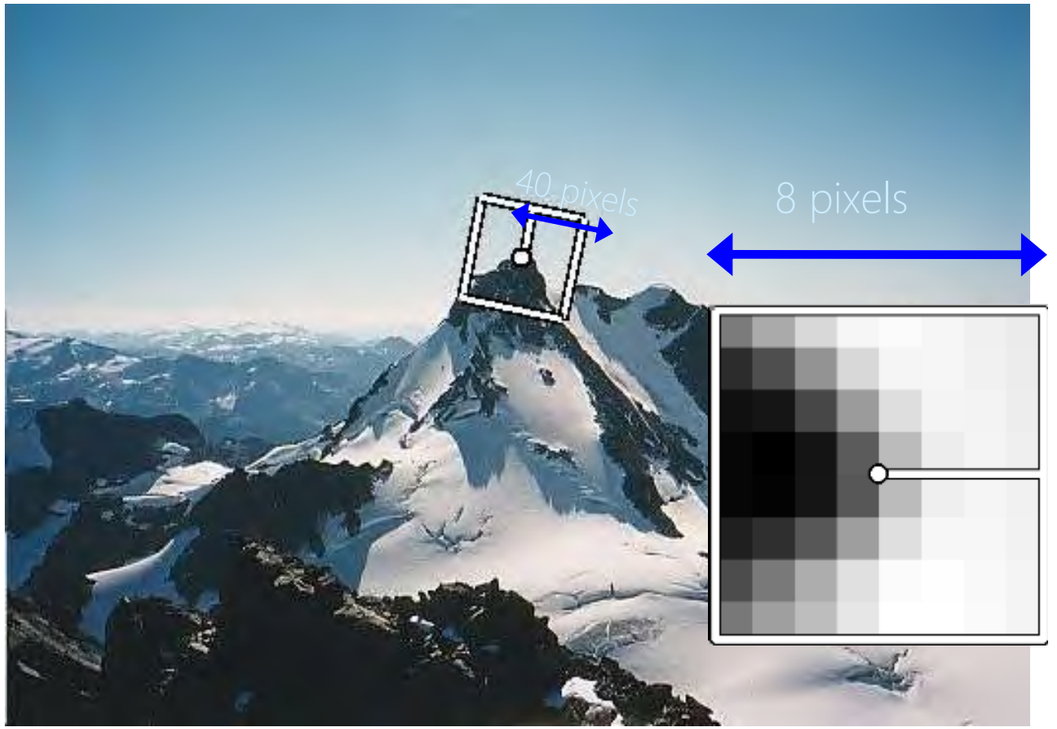
- 找到图像块的主要方向
  - 计算特征值 Eigenvalue
    - (最大的特征值)
  - 计算梯度方向
  - 旋转不同方向并提取特征

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

使用 40x40 方形窗口提取特征

- 缩放至1/5
- 旋转至水平
- 将 8x8 的方形窗口作为特征
- 减去均值和方差保证强度不变性



Adapted from slide by Matthew Brown

不同尺度下的特征提取

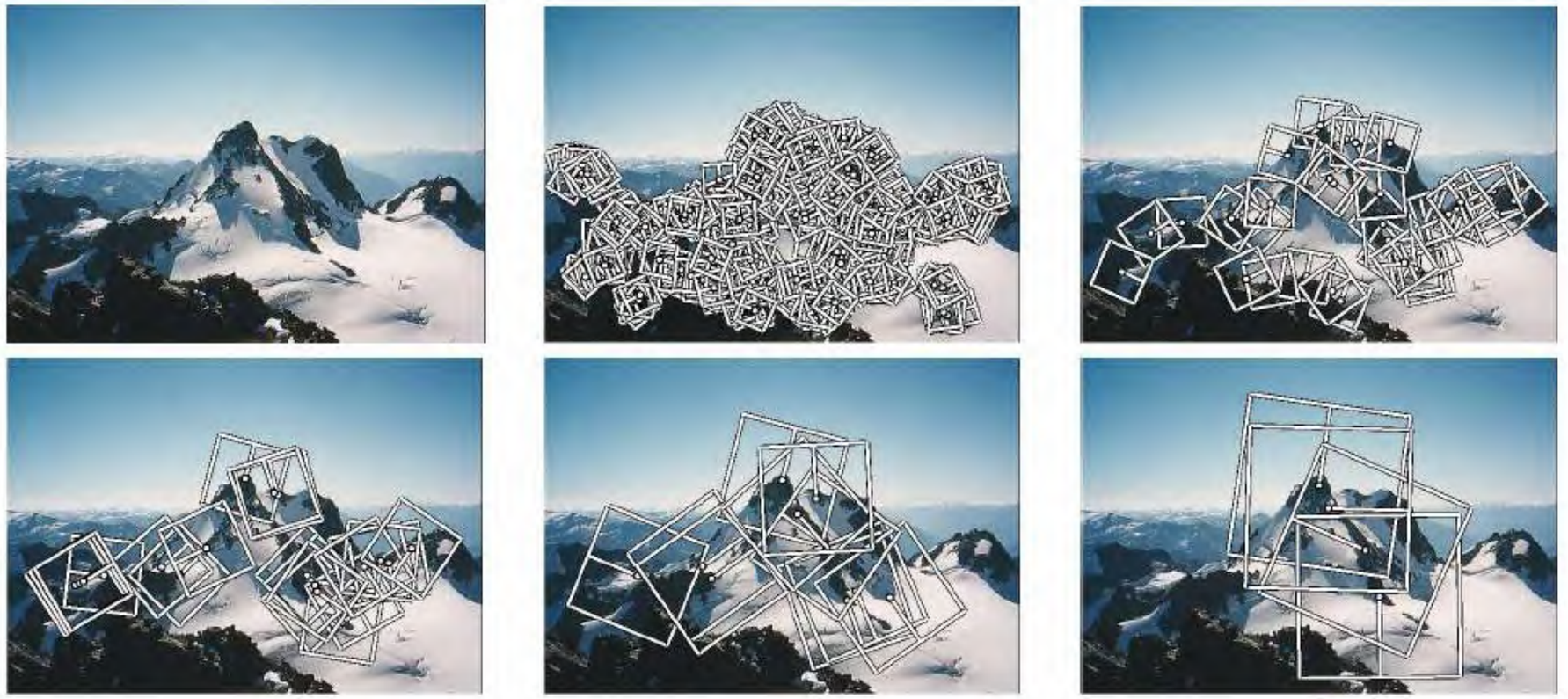
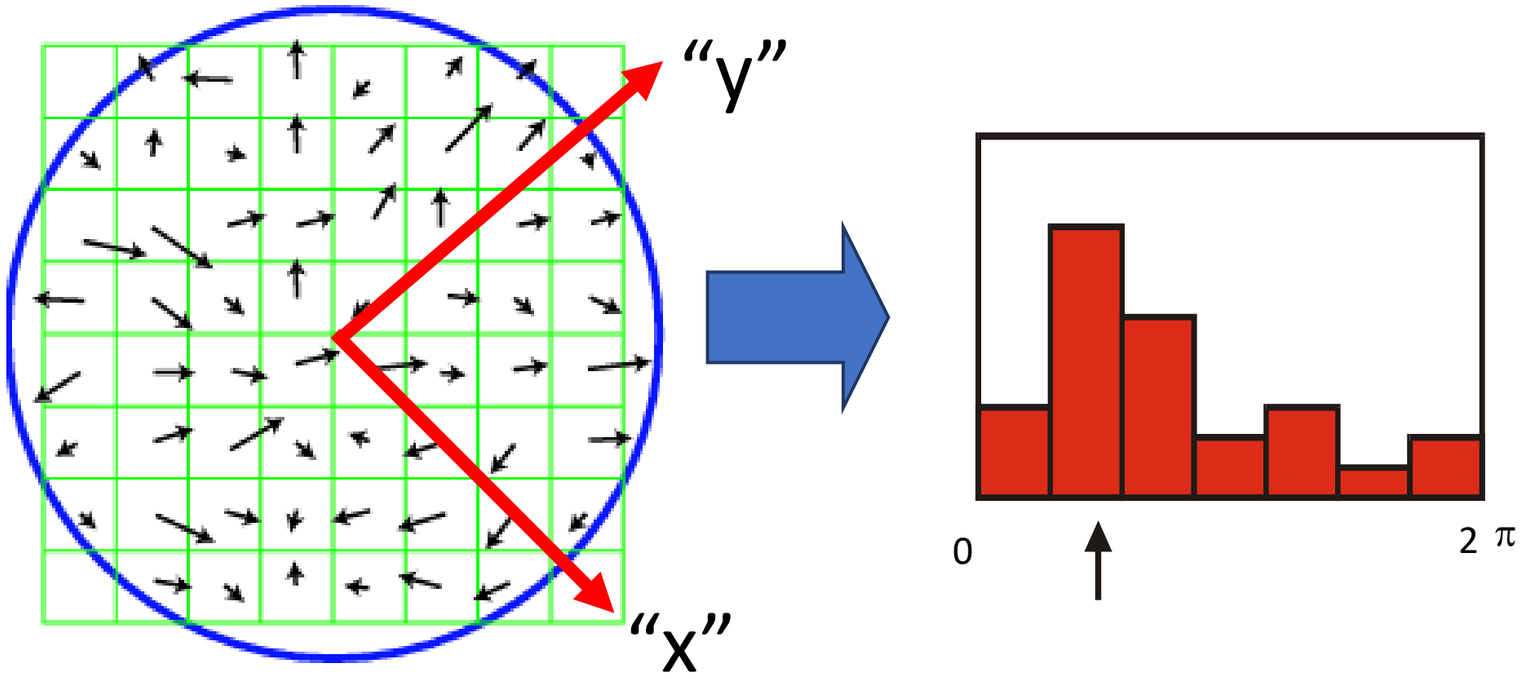


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

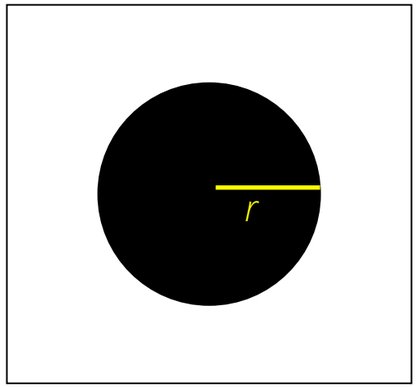
旋转不变：另一种方式

给定窗口，找到像素点梯度方向最多的方向

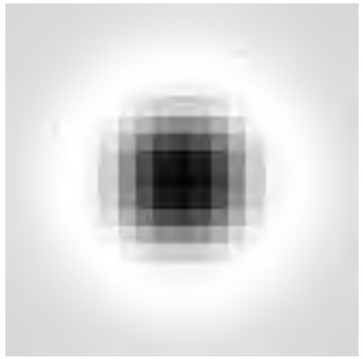


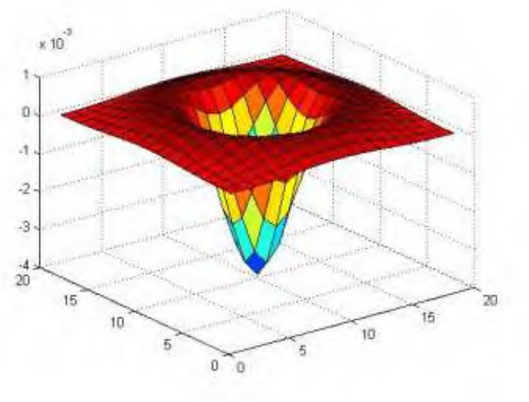
Slide credit: S. Lazebnik

特征尺度



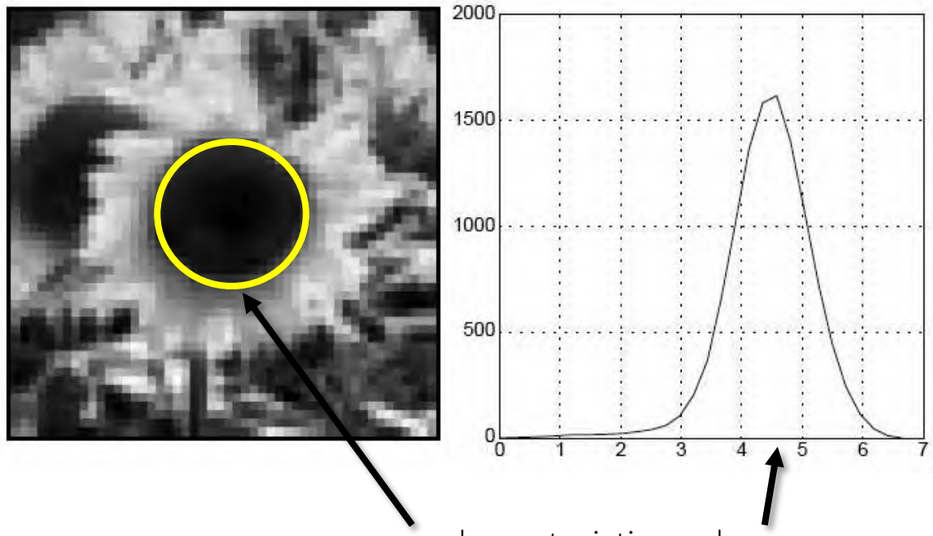
image





Laplacian

与角点相同，计算响应最大的尺度

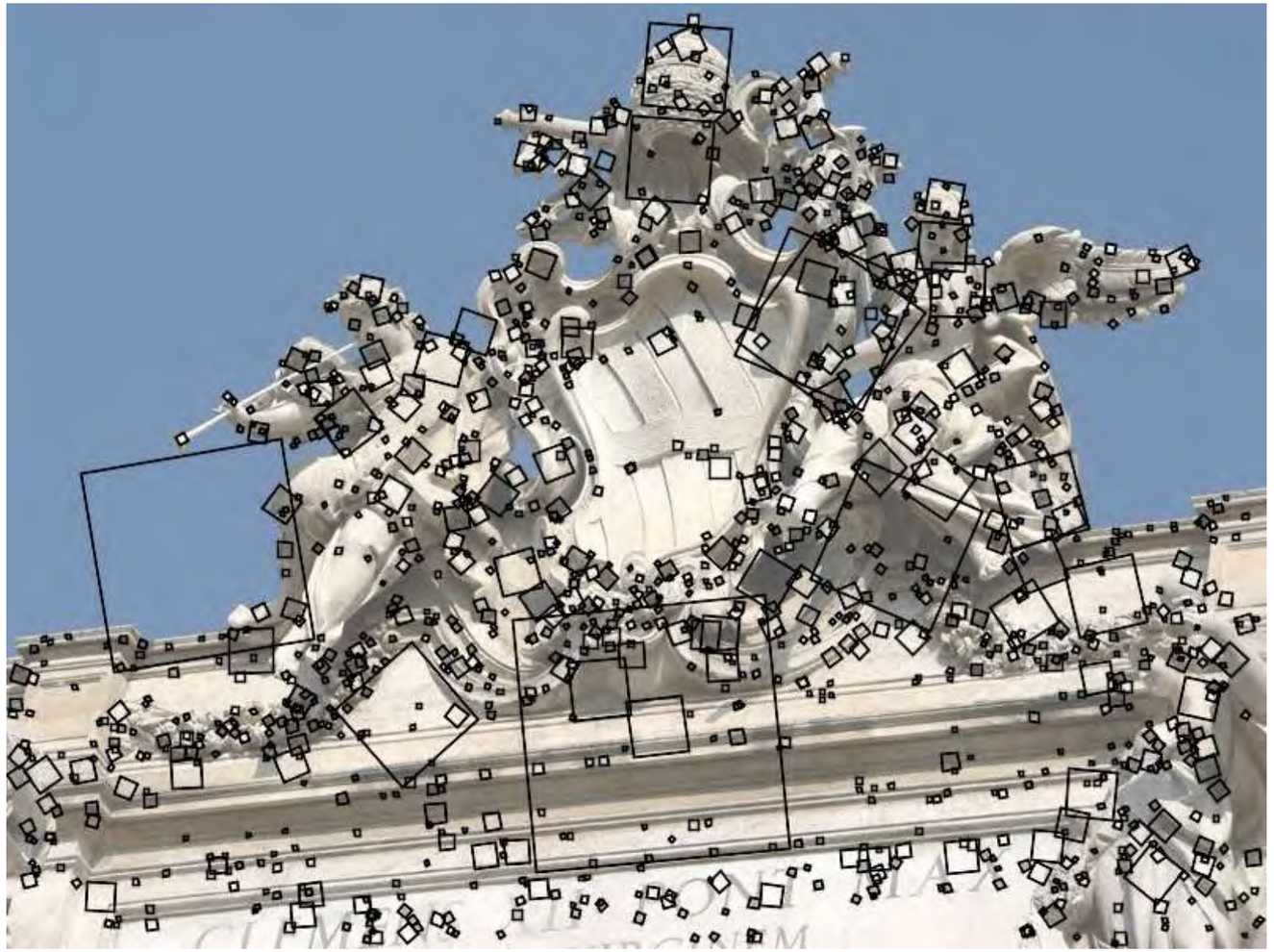


characteristic scale

- 首先找到特征尺度确定窗口大小

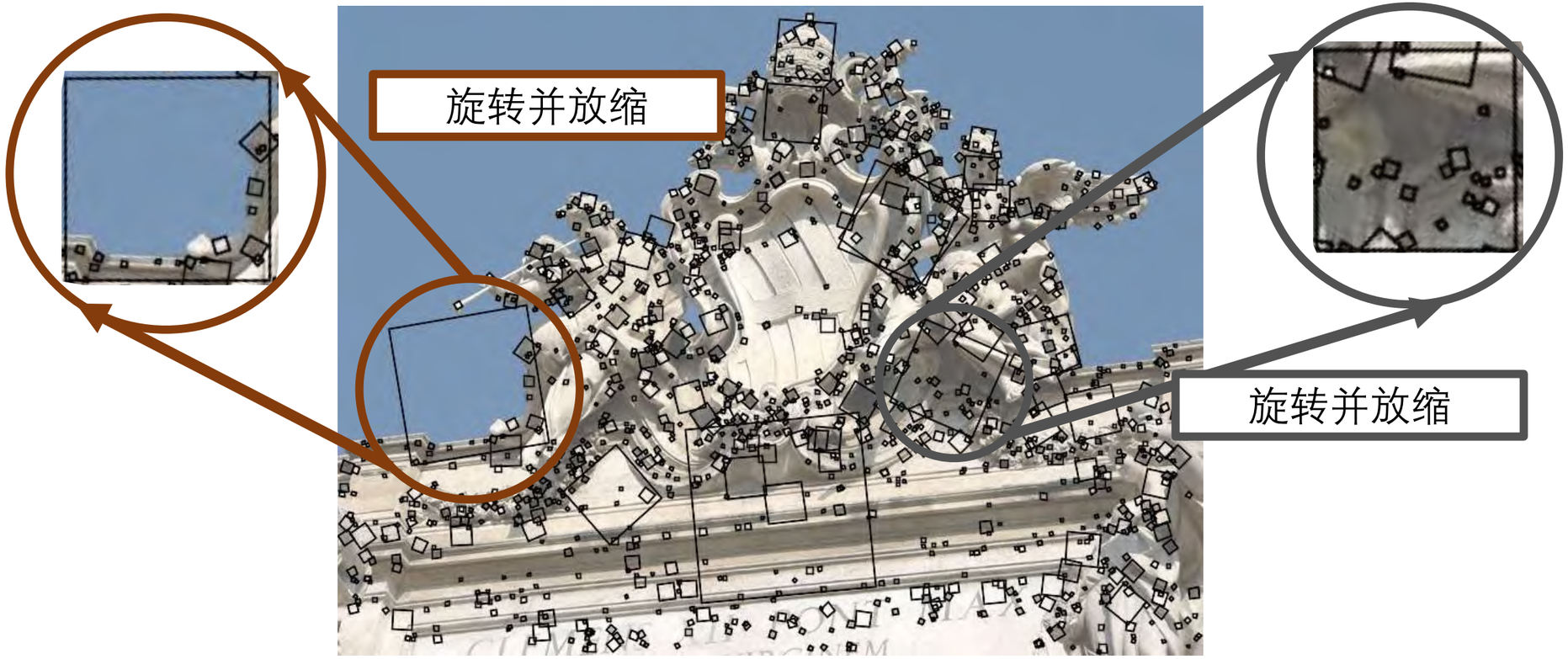
T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

尺度与旋转



Picture credit: S. Lazebnik. Paper: David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

尺度与旋转



Picture credit: S. Lazebnik. Paper: David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

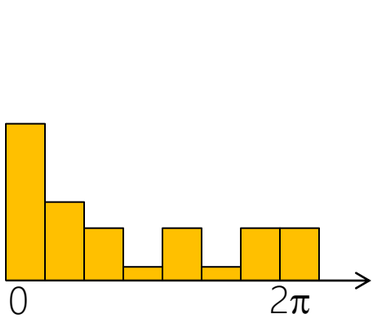
Scale Invariant Feature Transform

想法:

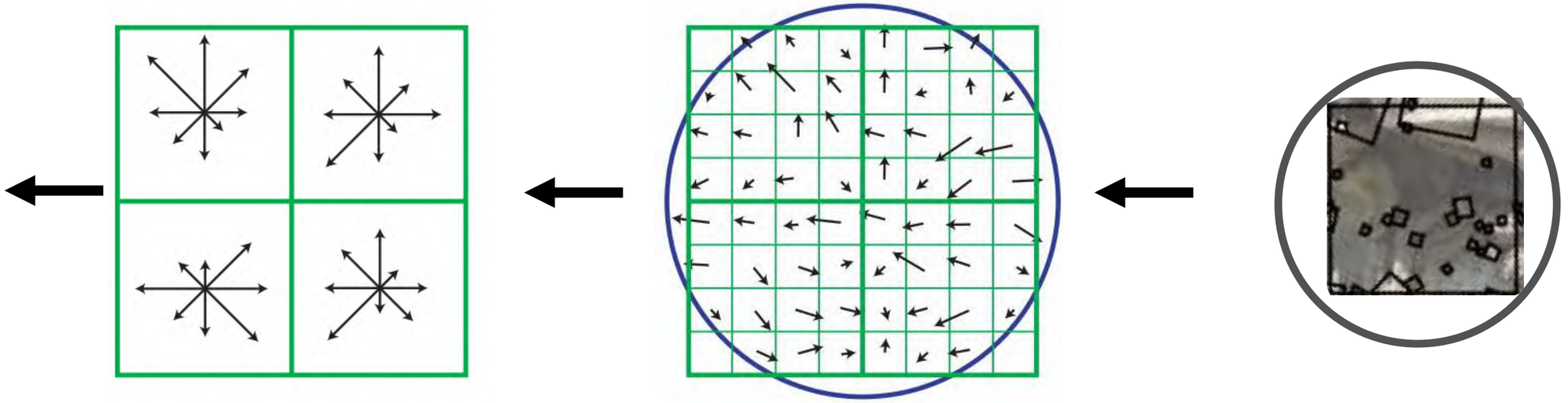
- 在检测到的特征周围取16x16的方形窗口
- 为每个像素计算边缘方向（梯度的角度 - 90度）
  - 减少亮度影响
- 排除弱边缘（梯度幅值低于阈值）
- 为剩余边缘方向创建直方图

做法:

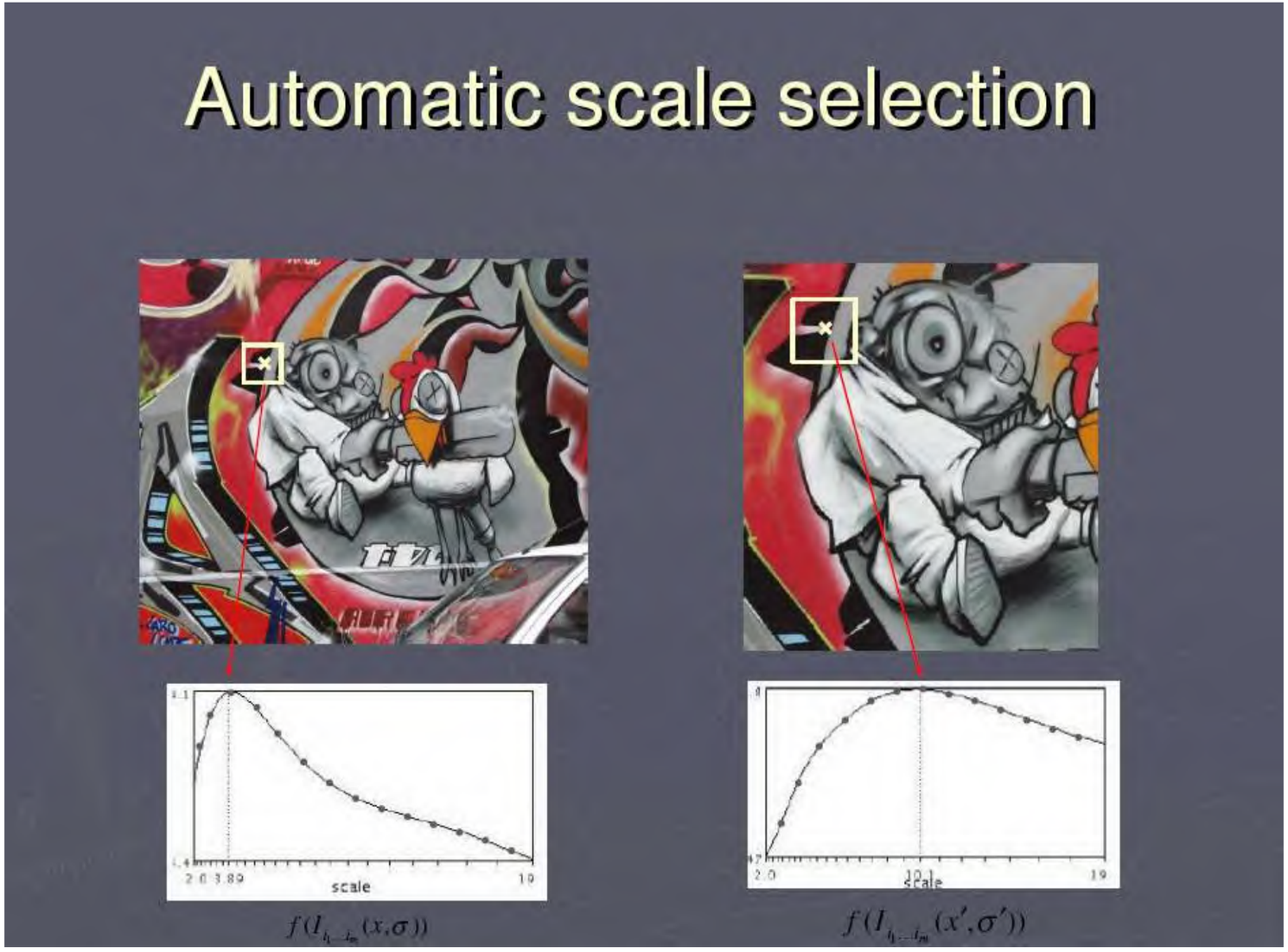
- 把16x16的窗口分成4x4 的格点 (在这里画的是2x2)
- 计算每个点的方向直方图
- 16 个格点 * 8 个方向 = 128 维的特征



梯度角度直方图



尺度不变性？特征尺度

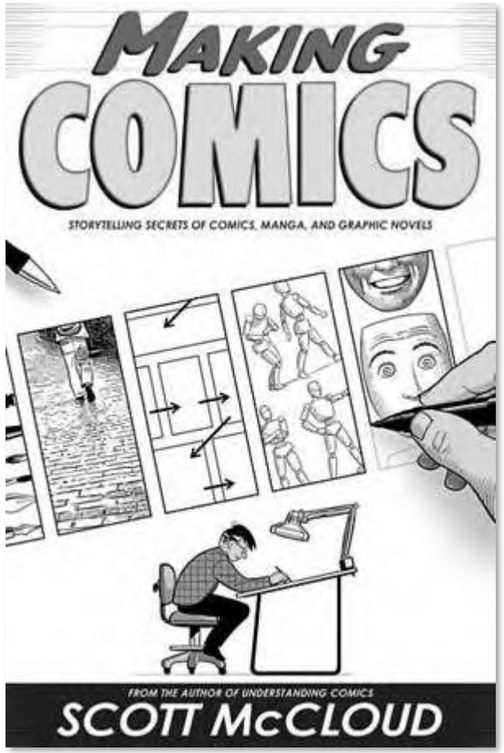

Automatic scale selection
scale
scale
$f(I_{i_1 \ldots i_m}(x,\sigma))$
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

SIFT

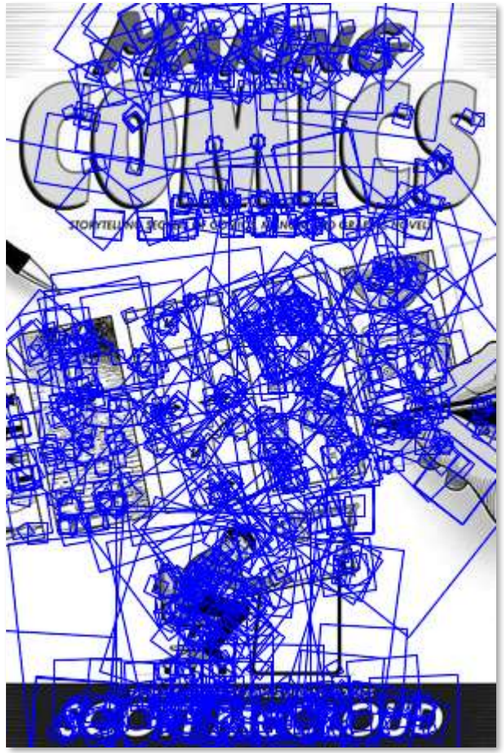
十分稳定的特征提取技术

- 视角变化(60 度以内的平面旋转)
- 光照变化(日景、夜景)
- 速度快!

SIFT



sift



868 SIFT 特征

其他特征提取器

- HOG: Histogram of Gradients (HOG)
  - 滑动窗口, 行人检测

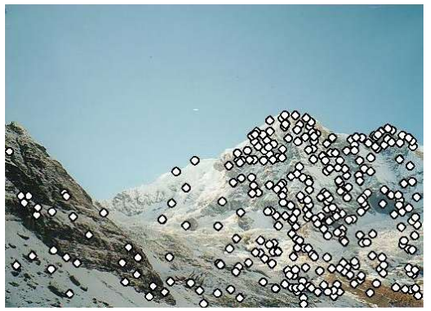
- FREAK: Fast Retina Keypoint
  - 用在SLAM 实时提取

- LIFT: Learned Invariant Feature Transform
  - 与深度学习结合，提取各种特征

    https://arxiv.org/abs/1603.09114

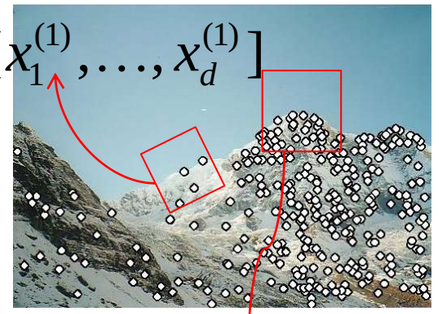
基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点



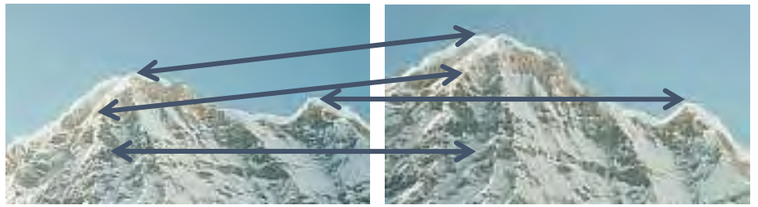
2) 解释 Description: 在关键点周围提取特征

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



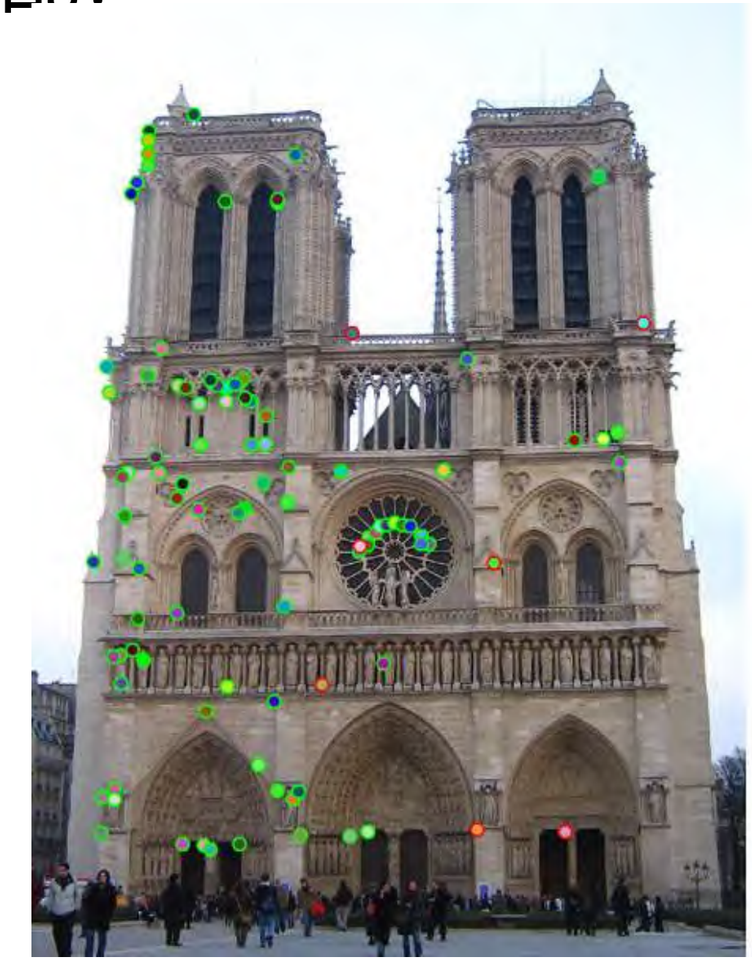
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

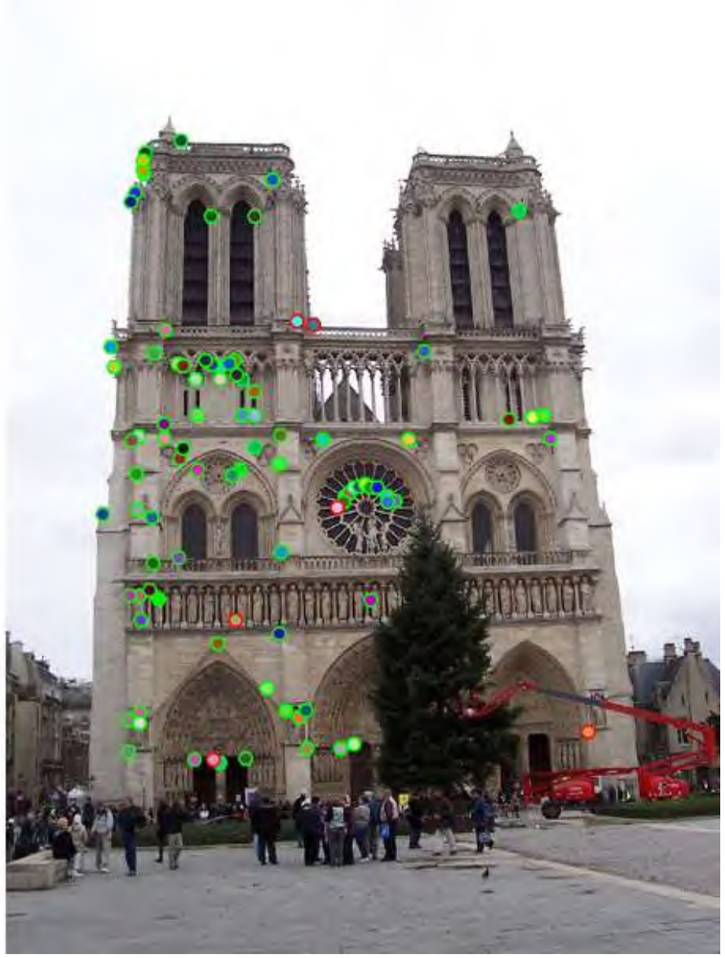
3) 匹配 Matching: 根据两个视角下的特征进行匹配



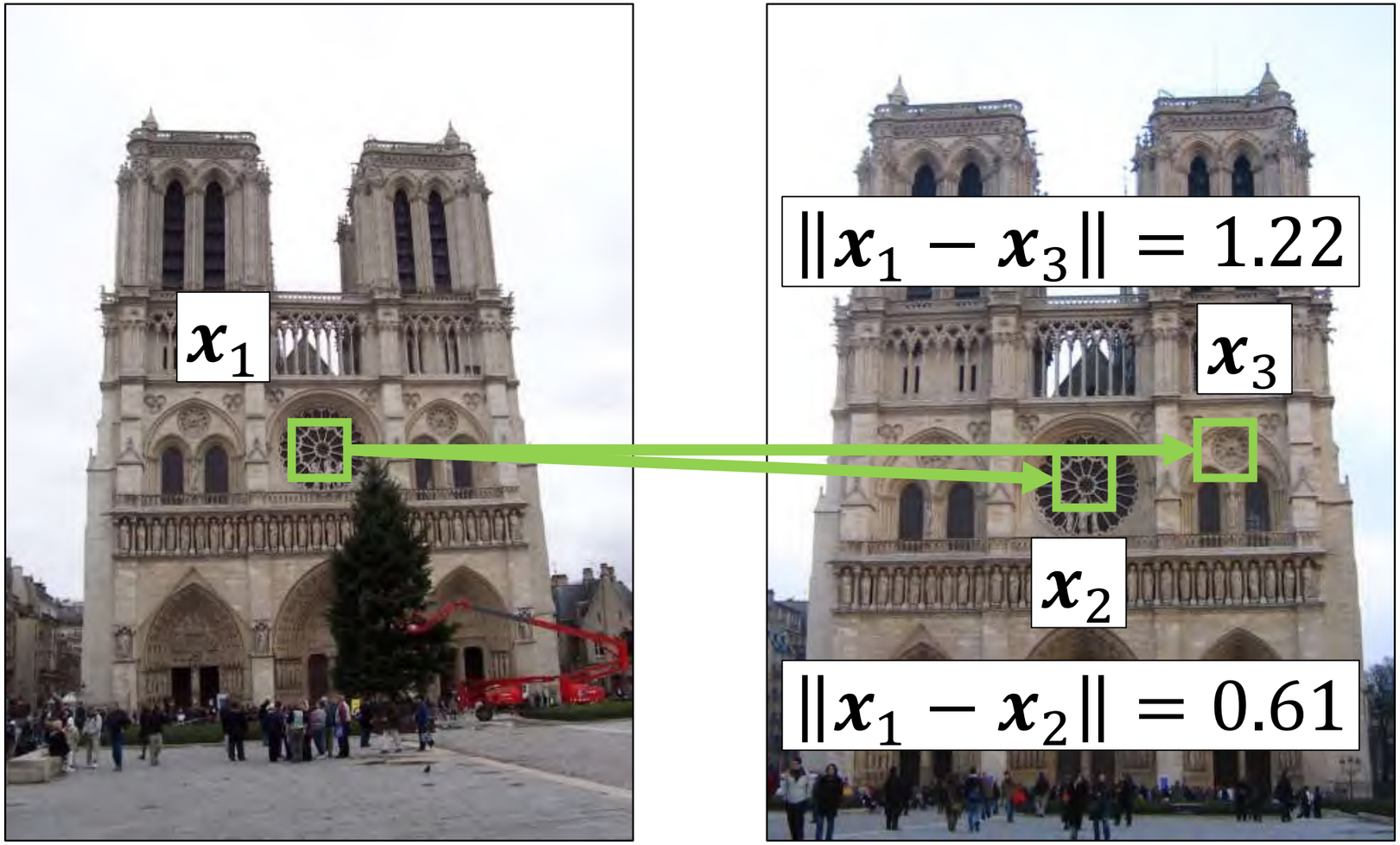
Kristen Grauman

怎么匹配?



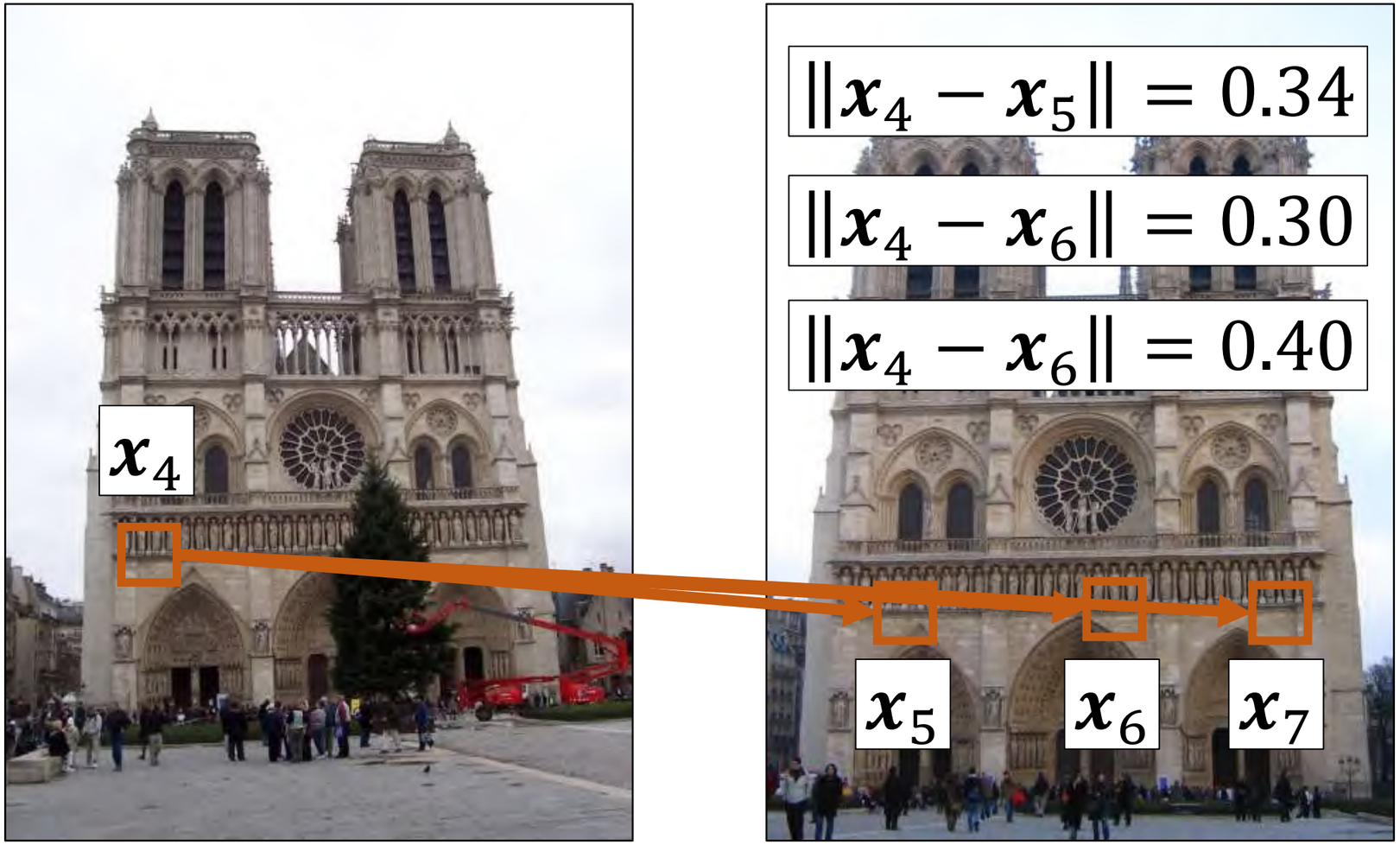


怎么匹配?



Example credit: J. Hays

怎么匹配?

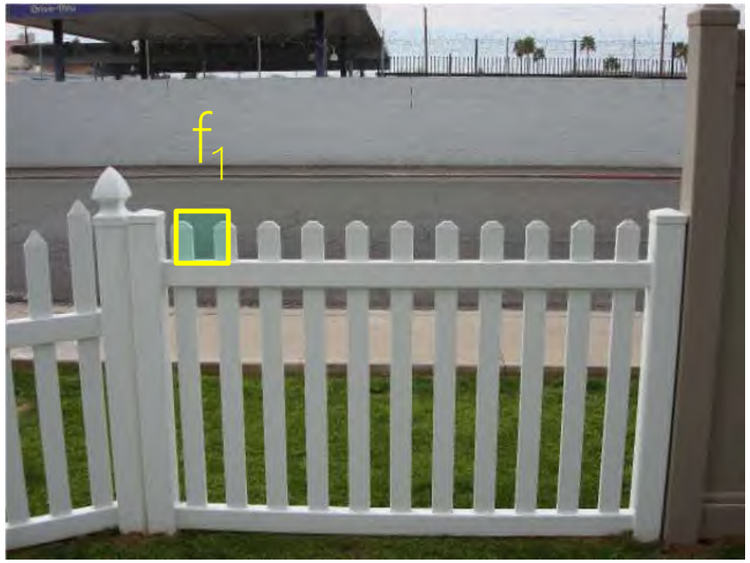


Example credit: J. Hays

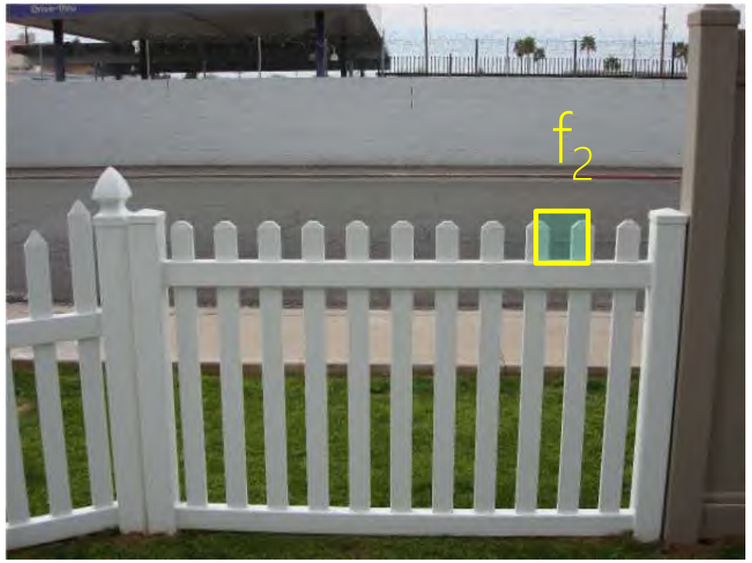
特征“距离”

怎么根据相似度匹配特征 f_1, f_2?

- 简单方法: L_2 distance, $\| f_1 - f_2 \|$
- 模糊匹配效果不好 (阈值法)



I_1

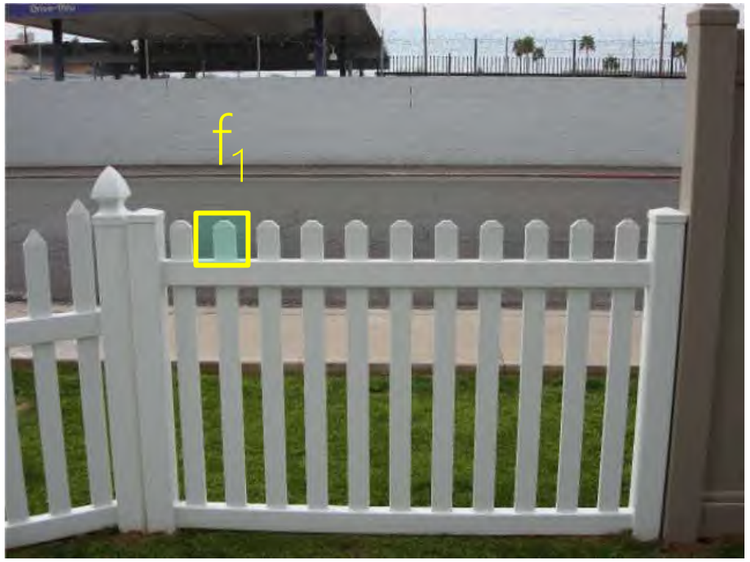


I_2

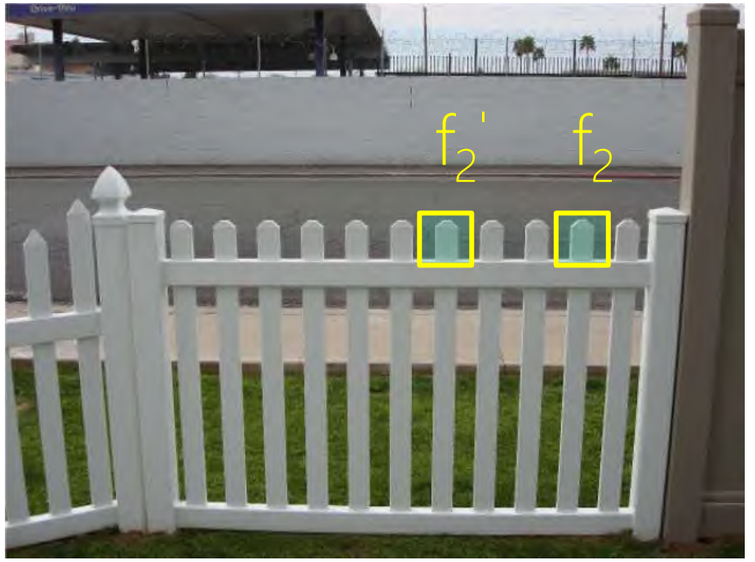
特征“距离”

怎么根据相似度匹配特征 f_1, f_2?

- 高级方法 2nd Nearest Neighbor Trick： $= \| f_1 - f_2 \| / \| f_1 - f_2' \|$
  - f_2 是 I_2 最匹配 f_1 的像素点
  - f_2' 是 I_2 第二匹配 f_1 的像素点
  - 模糊匹配效果较好:如果一个特征与其最近邻的距离与其与第二近邻的距离相差很大，那么这个匹配很可能是正确的。相反，如果这两个距离相差不大，那么这可能是一个误匹配

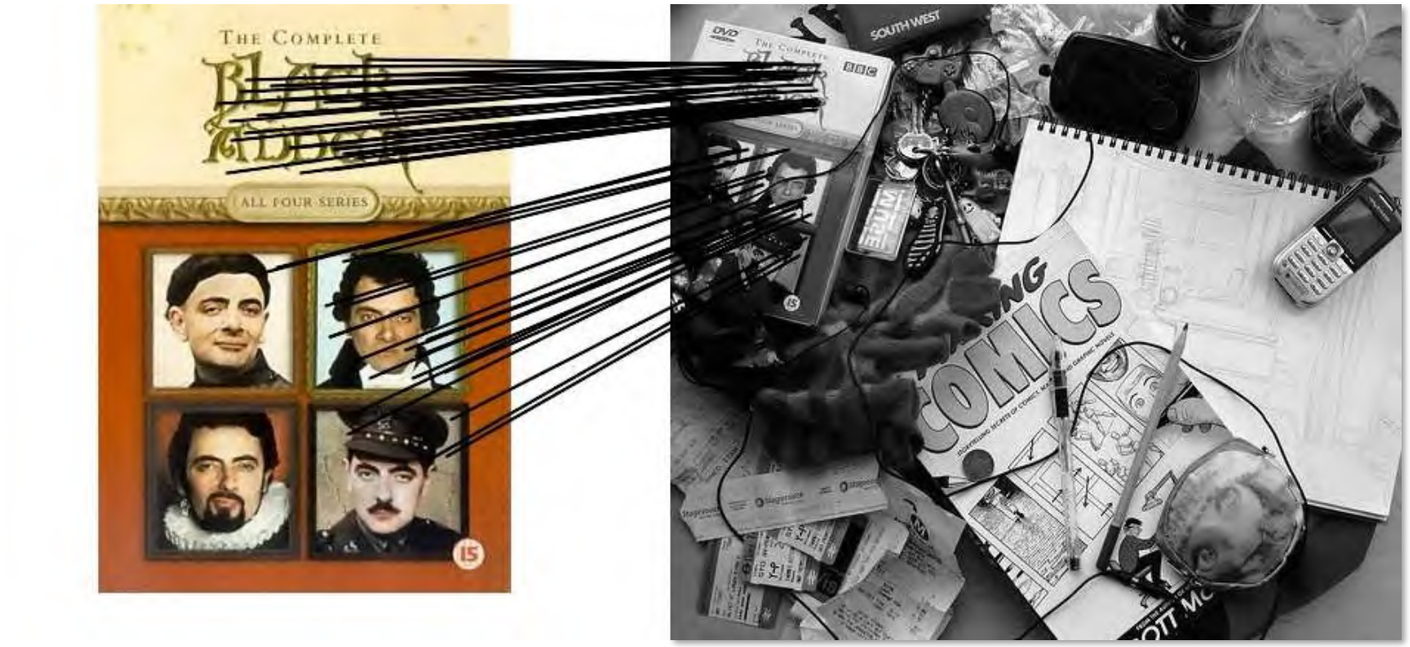


I_1



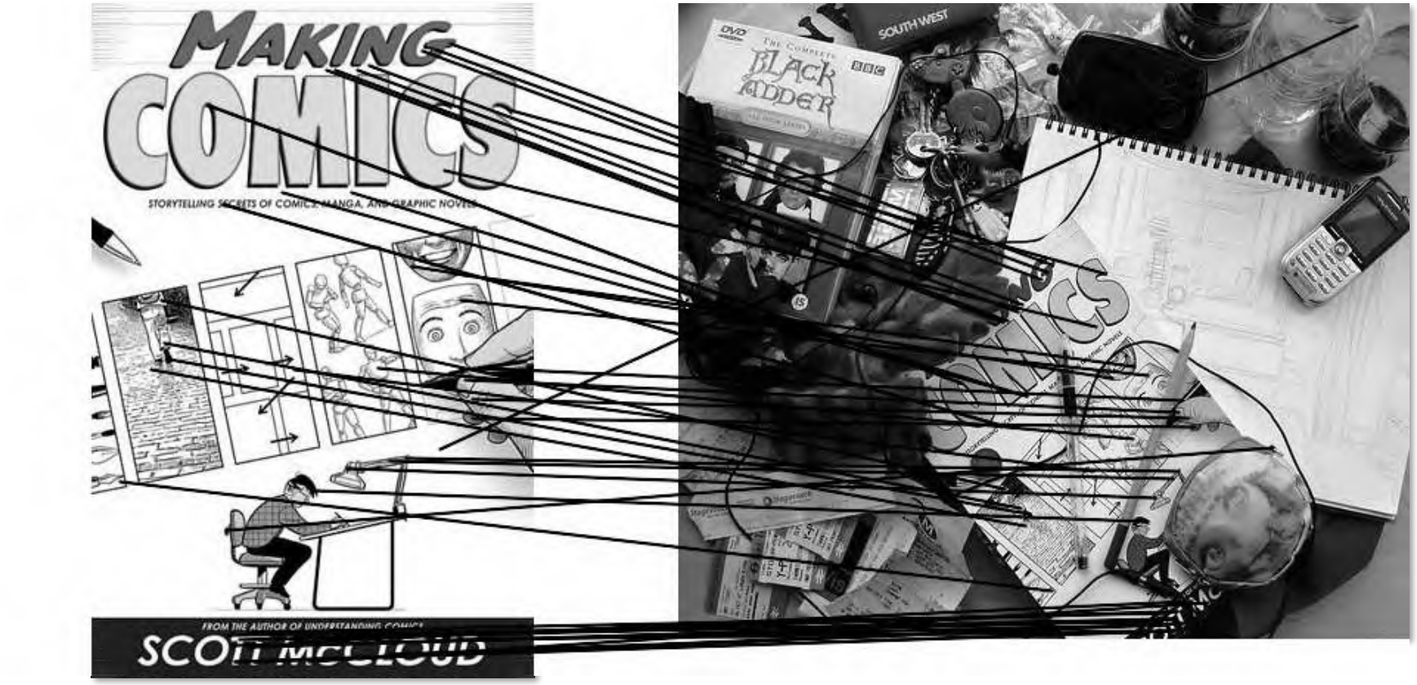
I_2

特征匹配



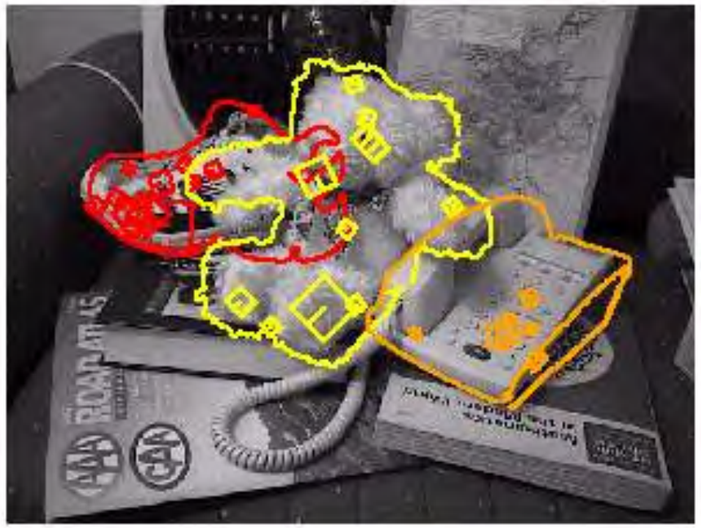
58 处匹配 (thresholded by ratio score)

特征匹配——Outliers

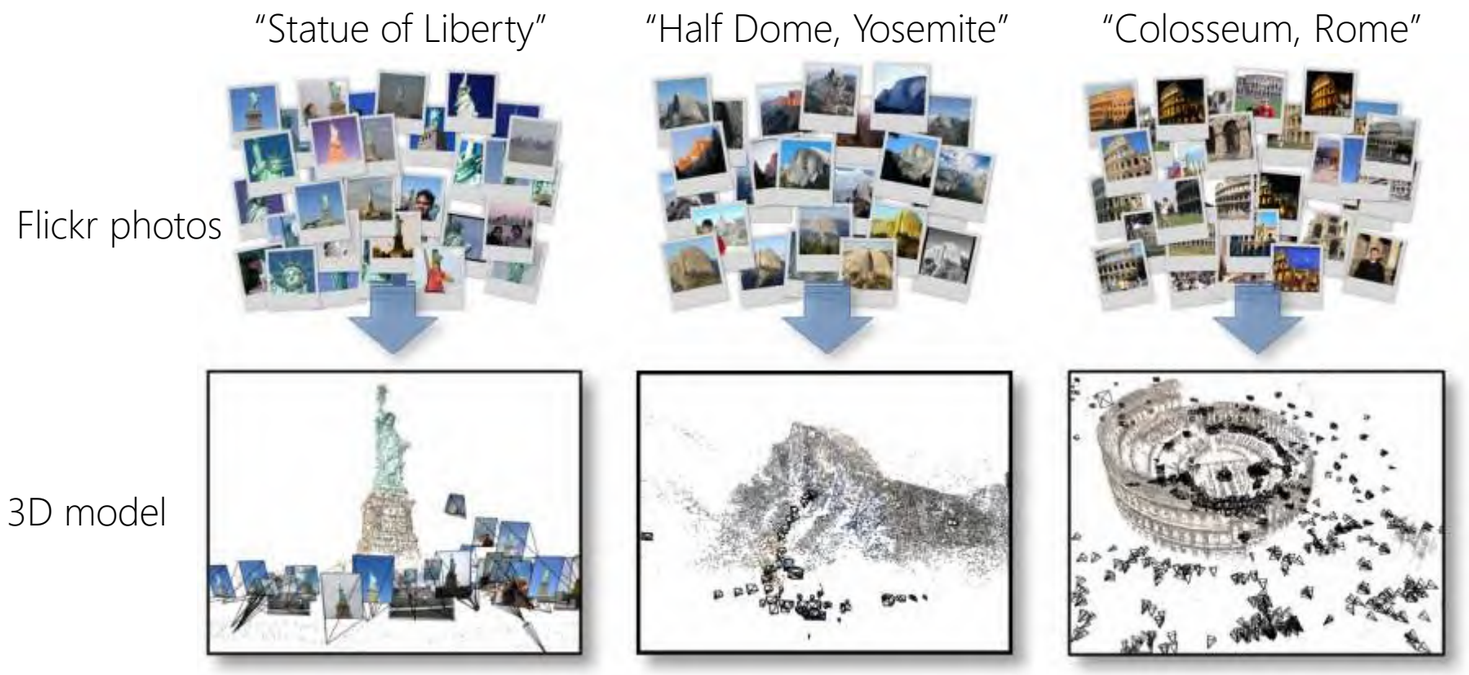


51处匹配(thresholded by ratio score)

物体识别 (David Lowe)

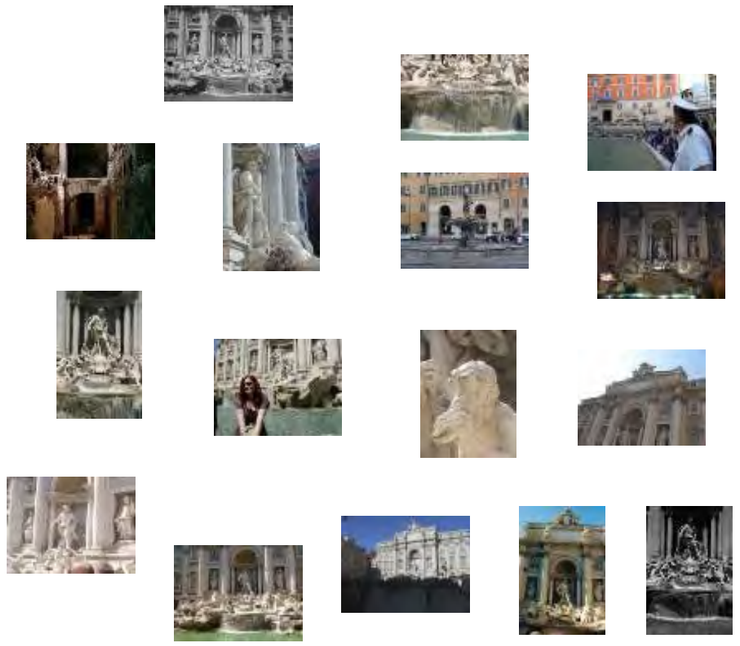


从网络照片重构一个场景



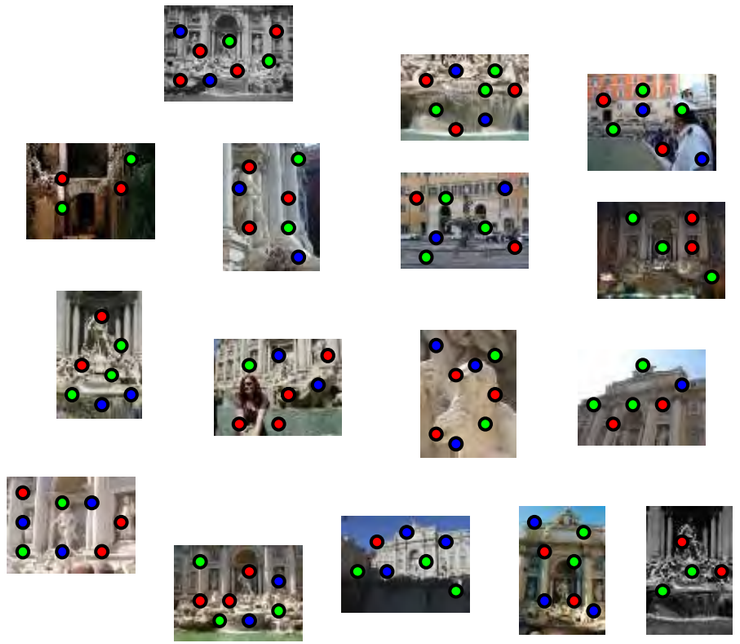
特征检测

使用SIFT检测特征 [Lowe, IJCV 2004]



特征检测

使用SIFT检测特征 [Lowe, IJCV 2004]



相似度关联

- 把不同图像的关键点通过相似程度联系起来

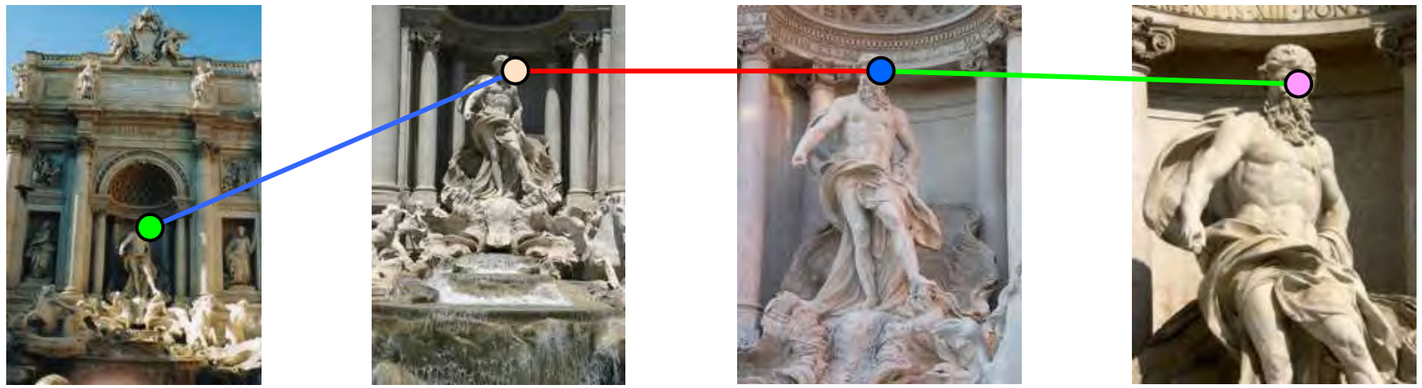
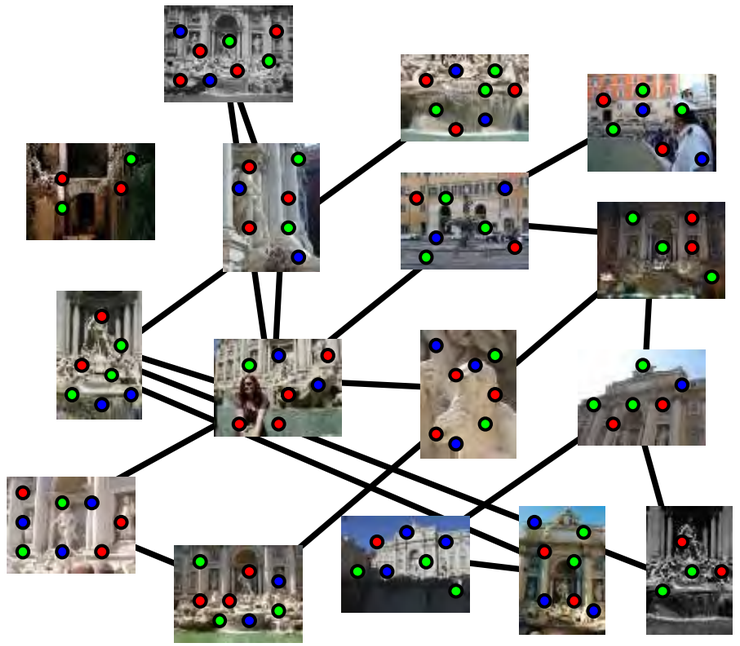


Image 1 Image 2 Image 3 Image 4

特征匹配

基于每对做特征匹配

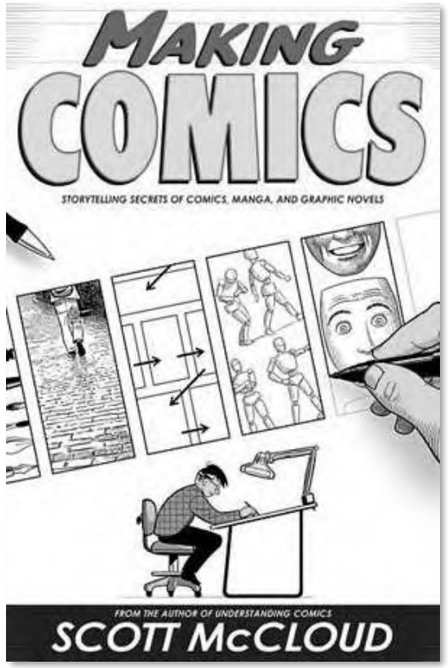


图像变换

全景图：图像对齐

为什么这样拼接下端的栏杆没有对齐?

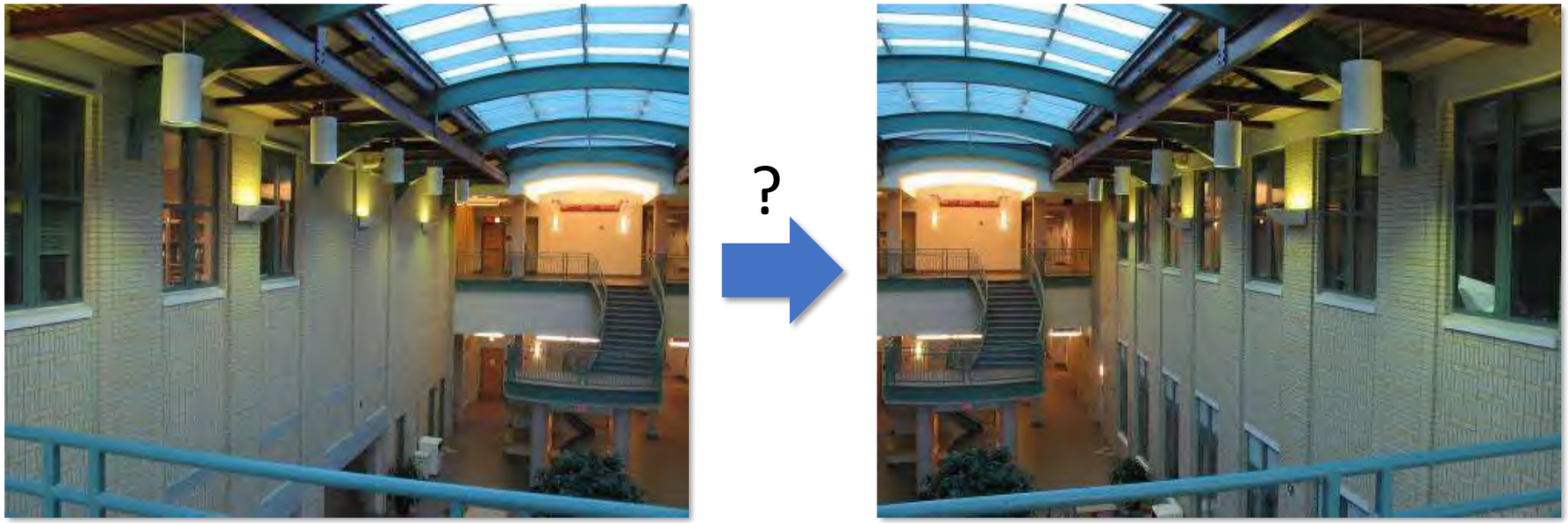
匹配的图像之间的几何关系是什么?



?

答案：相似性变换（平移、旋转、均匀缩放）

这两张图像之间的几何关系是什么?



这两张图像之间的几何关系是什么?

对制作全景图很重要!

首先，我们需要知道这种转换是什么。

其次，我们需要弄清楚如何使用特征匹配来计算它。

图像变换

图像滤波：更改图像**值域**

$$g(x) = T(f(x))$$

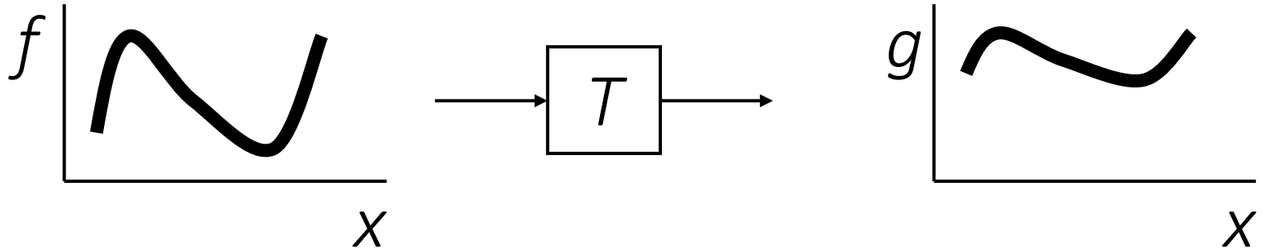
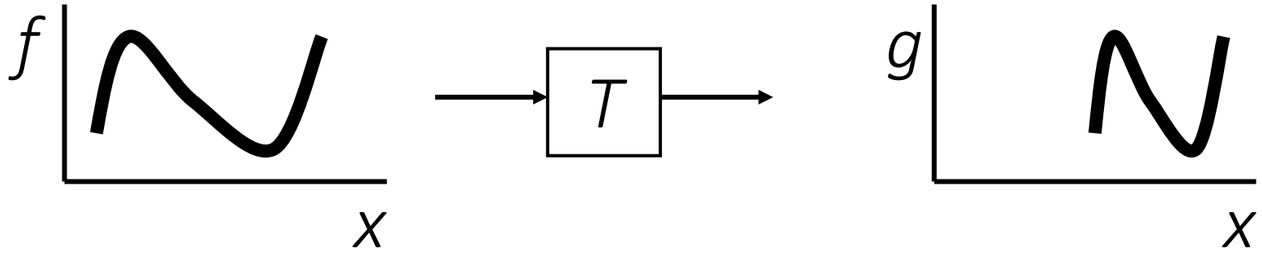


Image warping **图像扭曲**：更改图像的**定义域（domain）**

$$g(x) = f(T(x))$$

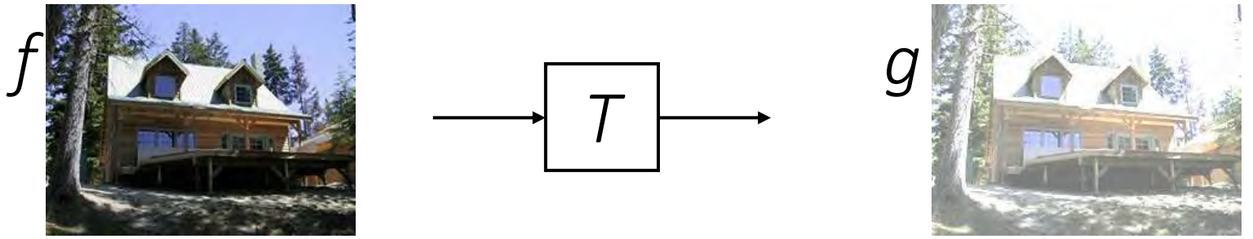


Slide credit: A. Efros

图像变换

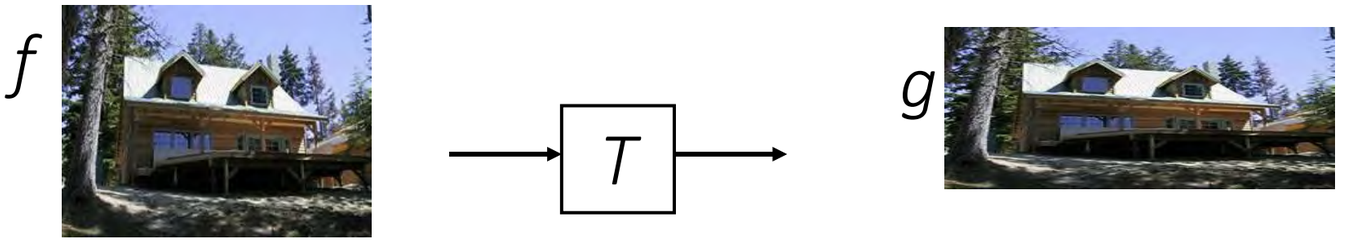
图像滤波：更改图像**值域**

$$g(x,y) = T(f(x,y))$$



图像扭曲：更改图像的**定义域**

$$g(x,y) = f(T(x,y))$$



Slide credit: A. Efros

参数化（全局）扭曲（**Warping**）

参数化扭曲的示例

Translation 平移

Rotation 旋转

Aspect 长宽比

Affine 仿射

Perspective 透视

Cylindrical 圆柱

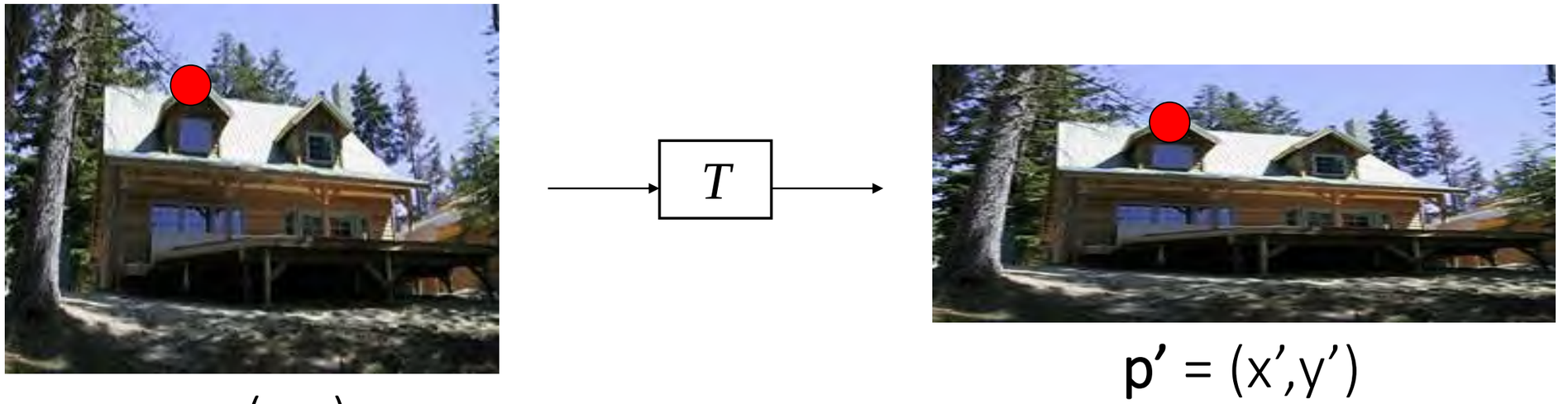
Slide credit: A. Efros

参数化（全局）扭曲

T 被描述为一个坐标变换机

$$\boldsymbol{p}' = T(\boldsymbol{p})$$

Note: 无论图像的内容是什么，T 都是相同的，且参数量很少，对图像的每个像素应用相同的变换



Slide credit: A. Efros

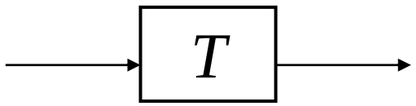
参数化（全局）扭曲

我们暂时只关注 线性变换

$$\boldsymbol{p' \equiv Tp}$$

T:矩阵; p, p’: 2D点。

p = (x,y)

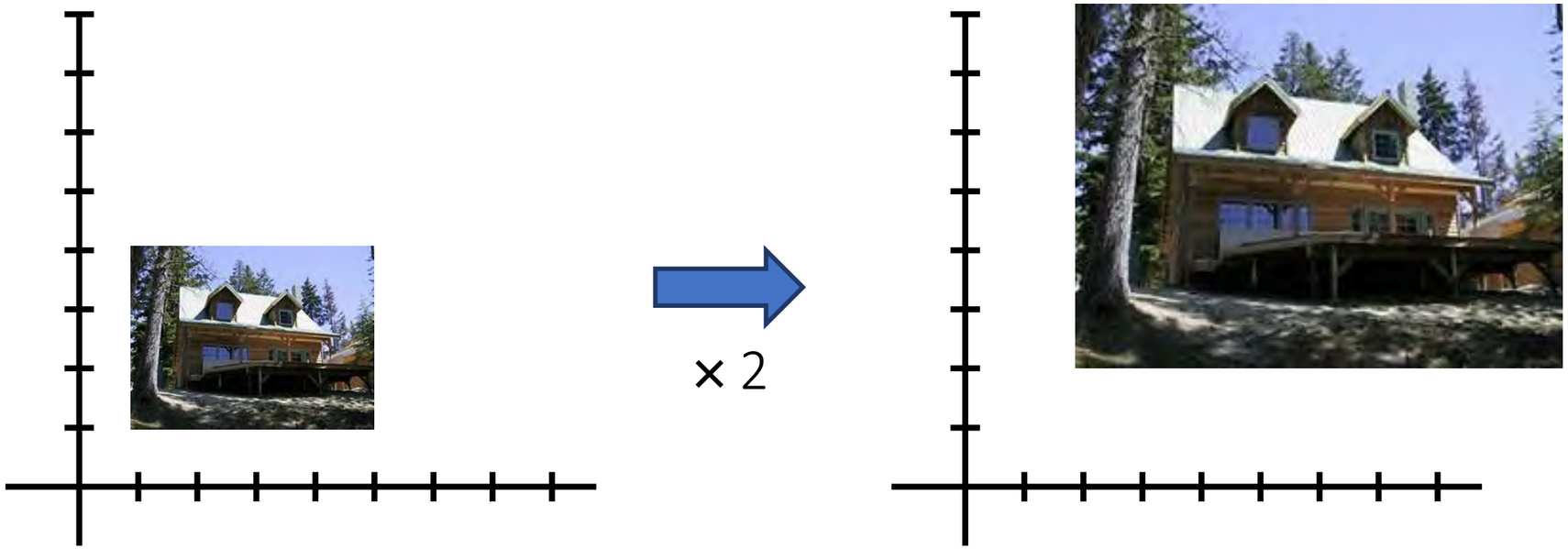


p’ = (x’,y’)

Slide credit: A. Efros

放缩

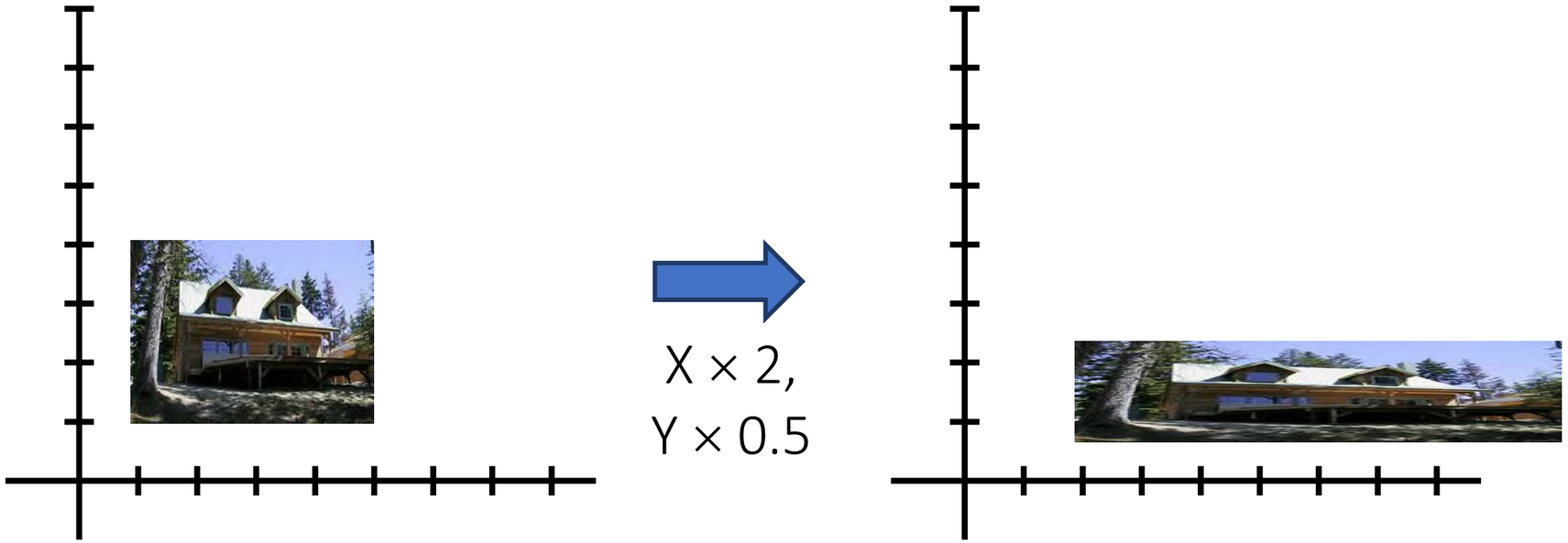
放缩 会把所有像素的坐标 (x,y) 乘以一个定值



Slide credit: A. Efros

放缩

放缩 会把所有像素的坐标 (x,y) 乘以一个定值



Slide credit: A. Efros

放缩

放缩变换的 T 是怎样的?

$$x' = ax$$

$$y' = by$$

矩阵形式:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{放缩矩阵 } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

S的逆矩阵是?

Slide credit: A. Efros

2D 旋转

旋转矩阵

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

R_θ的逆矩阵是？ $\boldsymbol{I = R_\theta^T R_\theta}$

Slide credit: A. Efros

其他 T 为 2x2 矩阵能做的变换

Identity 等价变换

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear 倾斜

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

其他 T 为 2x2 矩阵能做的变换

Before

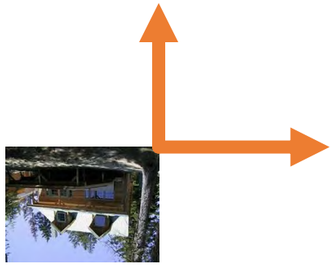
After

2D Y轴 镜像

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Before

After



2D 反转

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

T 为 2x2 矩阵时

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

T 变换后

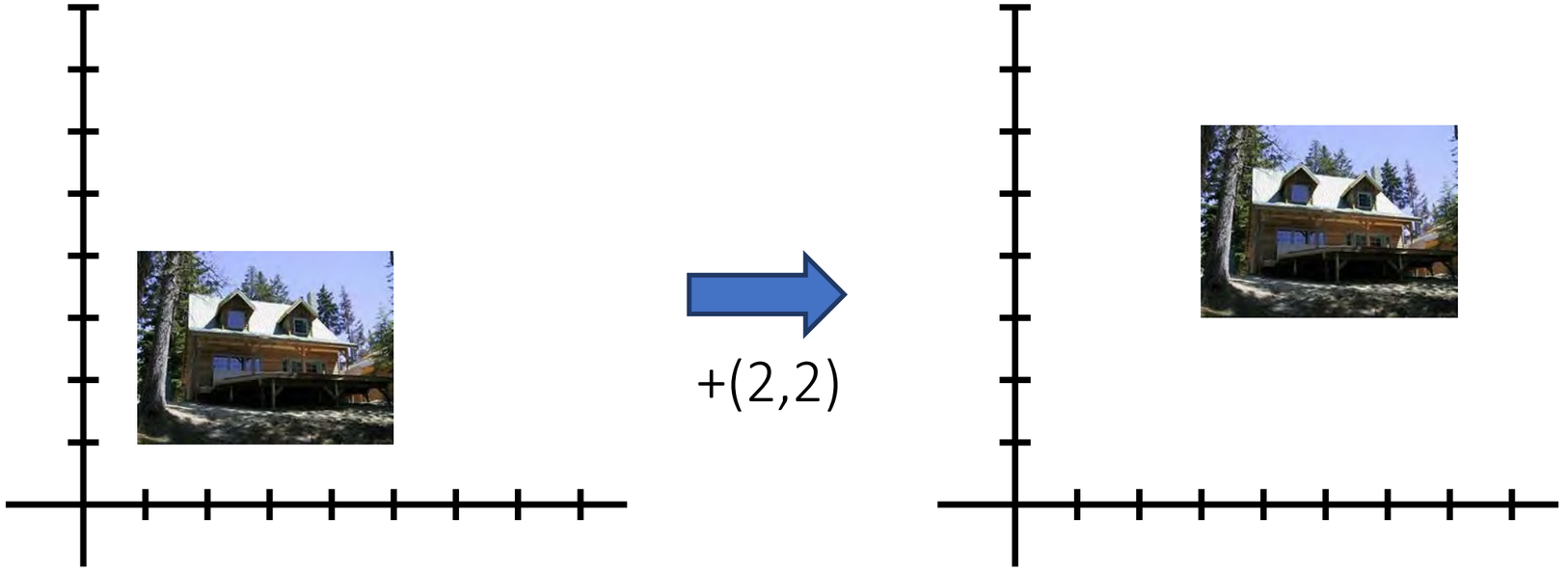
- 原点不变: **0 = T0**
  - 线还是线
- 平行线仍然平行

T 为 2x2 矩阵不能做的变换

平移怎么做?

$x' = x + t_x$, $y' = y + t_y$

怎样用线性变换表示?

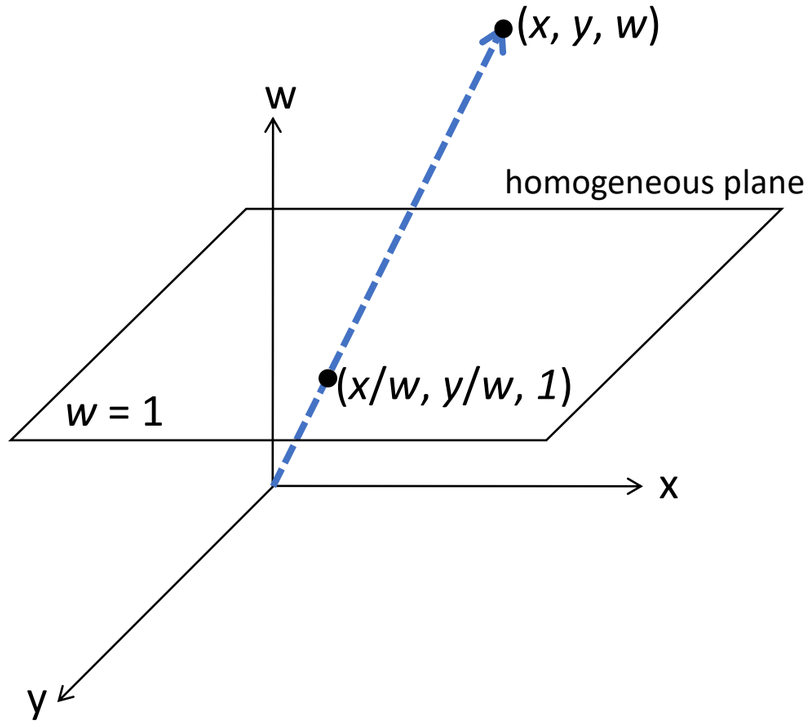


齐次坐标

Trick: 增加一维坐标:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

齐次图像坐标



齐次坐标到图像坐标的转换

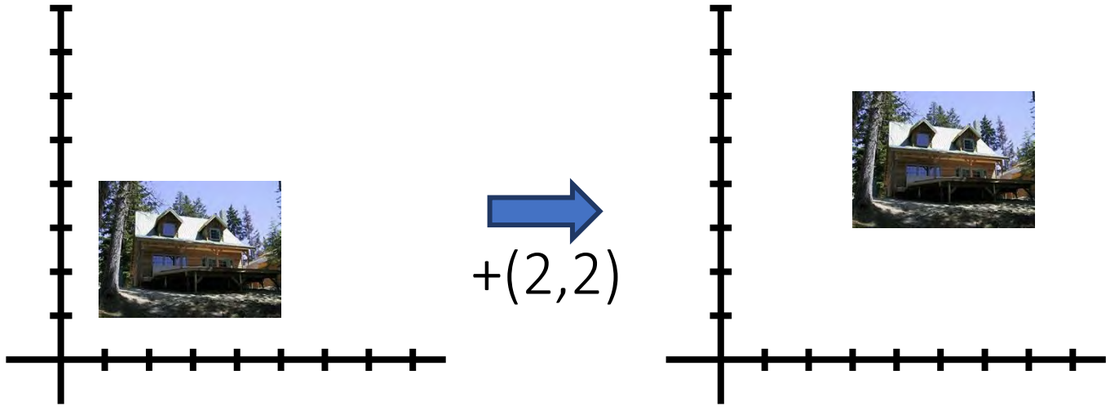
$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

图像齐次坐标

平移怎么做?

$x' = x + t_x$, $y' = y + t_y$

$$\begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Slide credit: A. Efros

Affine transformations：仿射变换

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

任何 由 3x3 矩阵表示的变换，其最后一行为 [0 0 1]我们称之为仿射变换。

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

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

平移

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

放缩

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

2D旋转

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

倾斜

仿射变换

- 仿射变换可以认为是以下变换的组合 ...
  - 线性变换
  - 平移

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

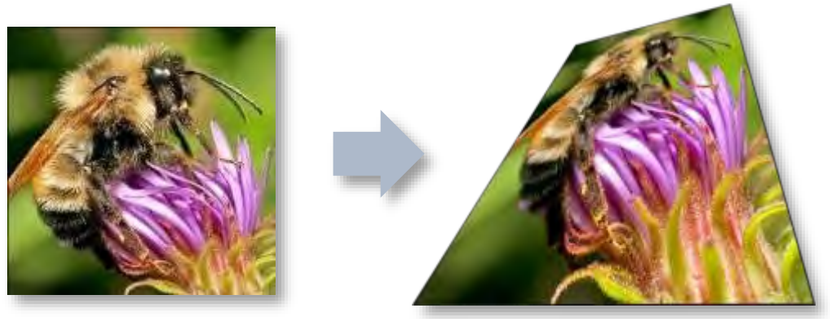
- 仿射变换的性质:
  - 原点变换后不一定在原点
  - 线仍然是线
  - 平行线仍然平行
  - 长度相对比例会保留
  - 仿射变换的组合还是仿射变换

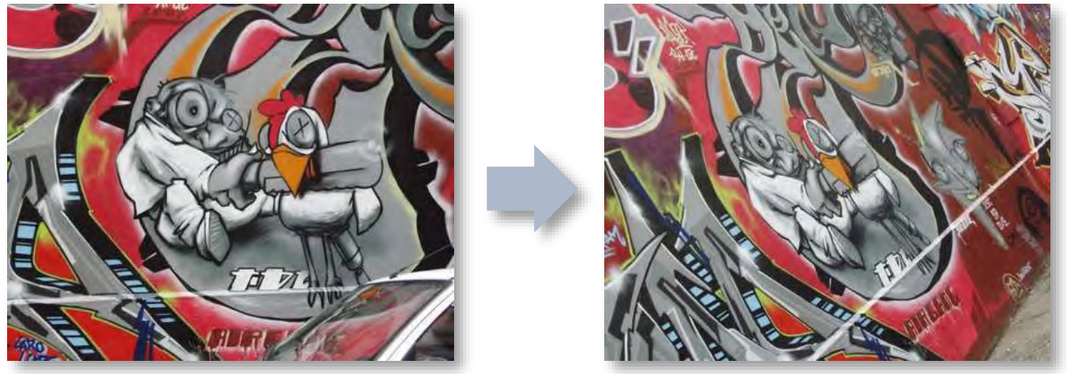
这是仿射变换吗?

Projective Transformations 投影变换
aka Homographies 同构变换
aka Planar Perspective Maps 平面透视映射

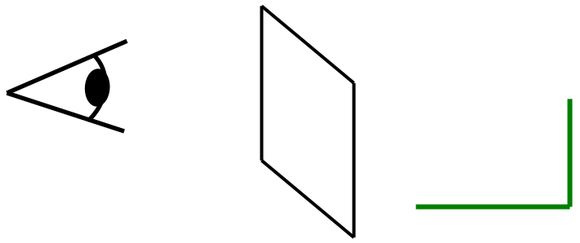
$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

这种变换称为同构变换





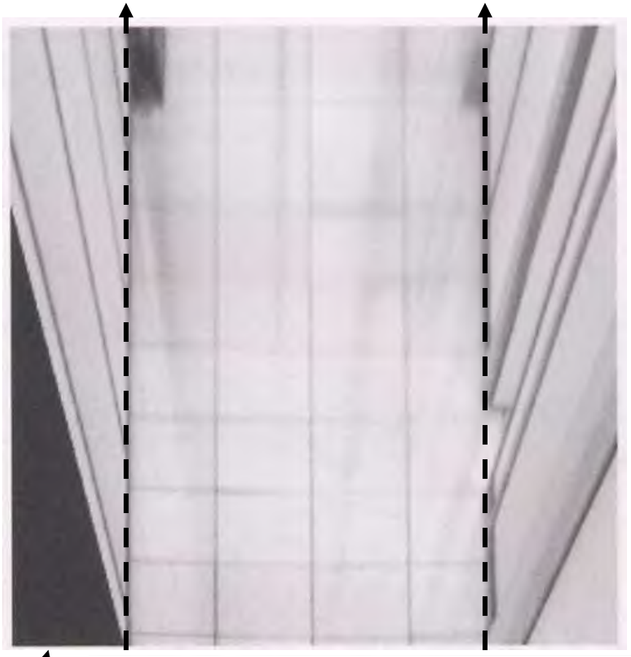
同构变换

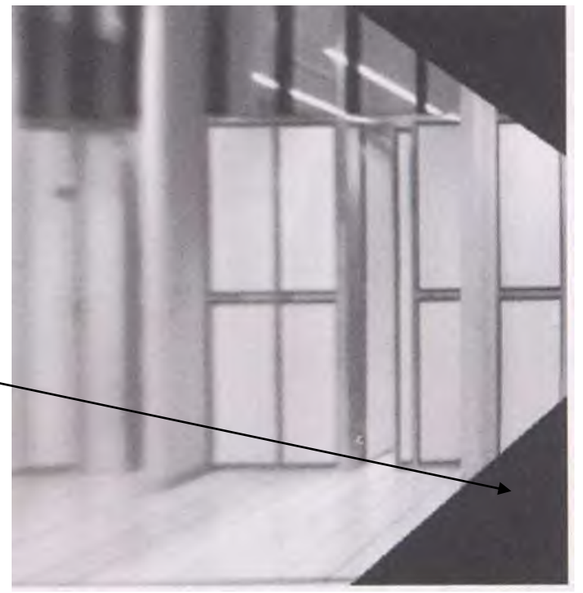


正面视角的图像

$\mathbf{H}_1$

$\mathbf{H}_2$





黑色区域
代表没有
像素

同构变换: 全景图是从不同视角采集的

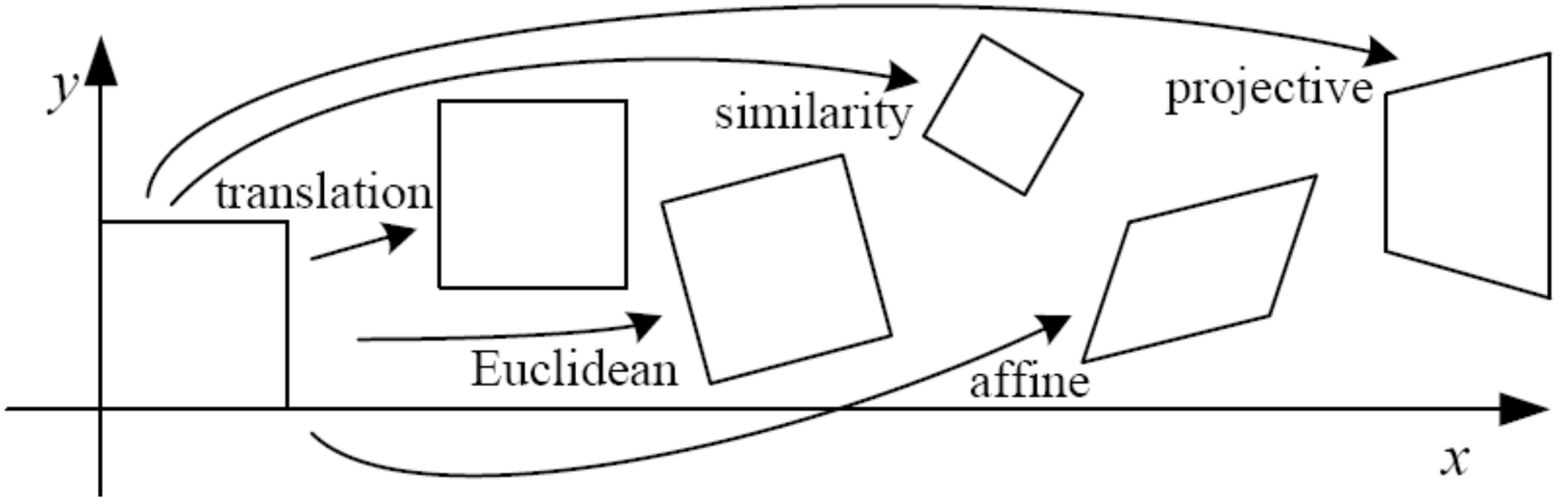
同构变换

- 同构变换包括...
  - 仿射变换
  - 透视变换

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

- 性质:
  - 原点变换后不一定是原点
  - 线仍然是线
  - 平行线不一定平行
  - 比例可能变化
  - 同构变换的组合还是同构变换

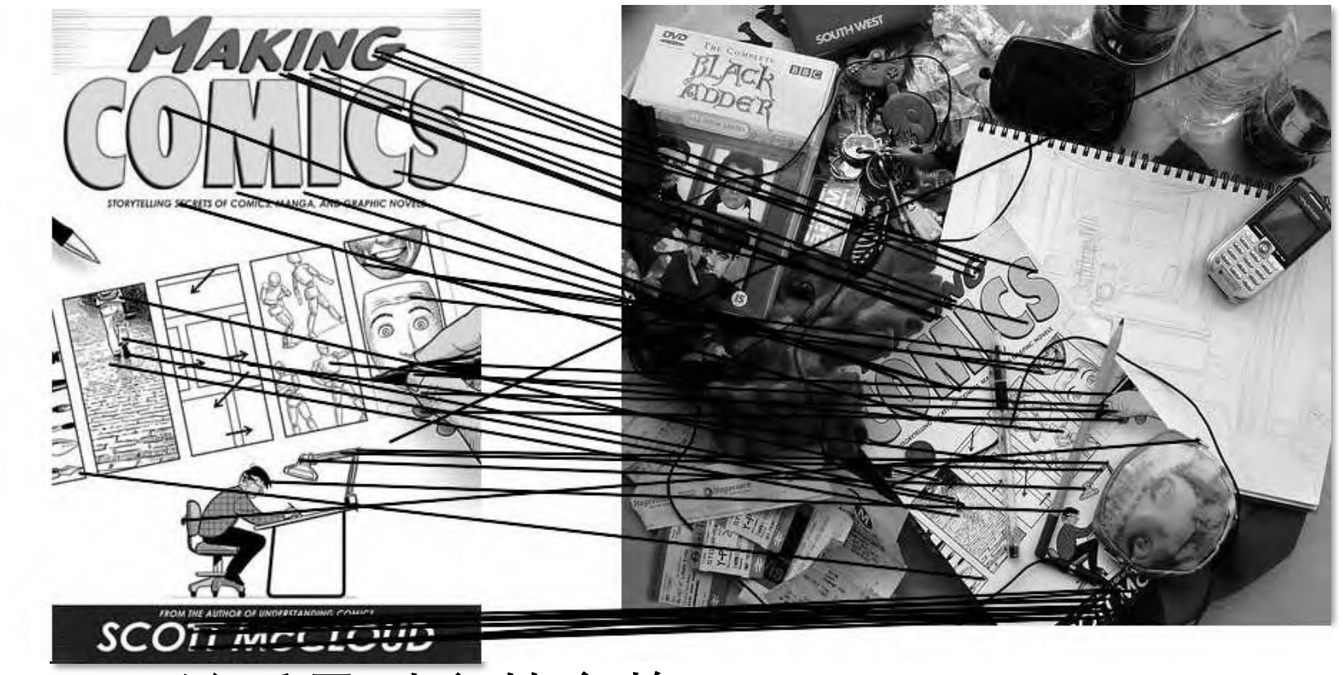
2D 图像变换



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

计算变换矩阵

- 给出两张图像和对应的特征点匹配
  - 怎样计算从A到B的变换？



- 找到跟特征匹配关系最对应的变换

计算变换矩阵

?

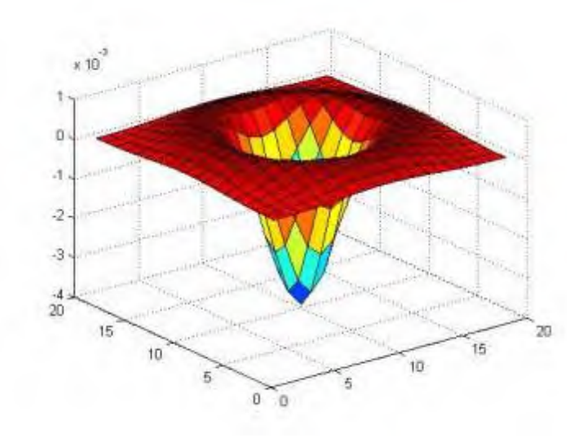
回顾——斑点检测

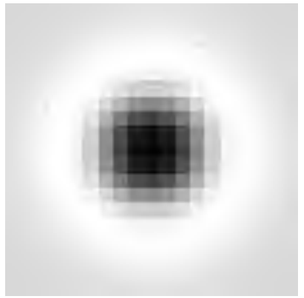
特征提取：斑点

- 斑点（Blobs）：斑点是图像中的连续区域，其像素值具有明显的局部极值。

计算斑点的方式

- The *Laplacian （二阶导数） of Gaussian* (*LoG*)





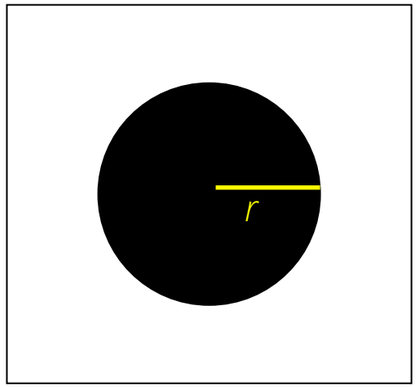
高亮点位于图像中心的图像，而图像的边缘和角点等特征会被突出显示

LoG 对旋转保证不变性

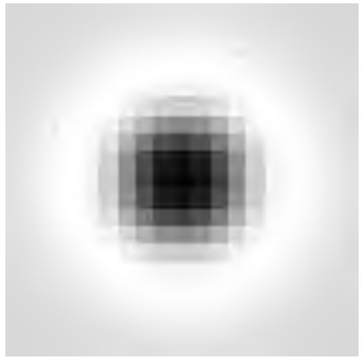
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

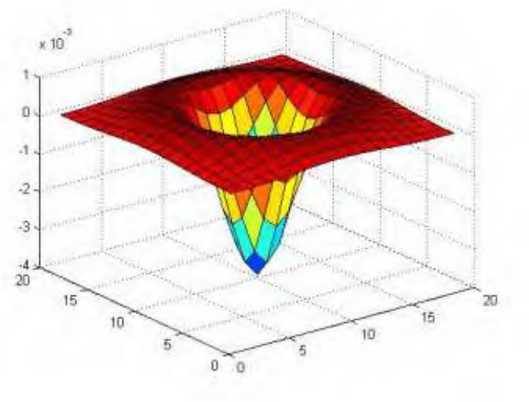
一个区域内高响应，周围低响应

特征尺度



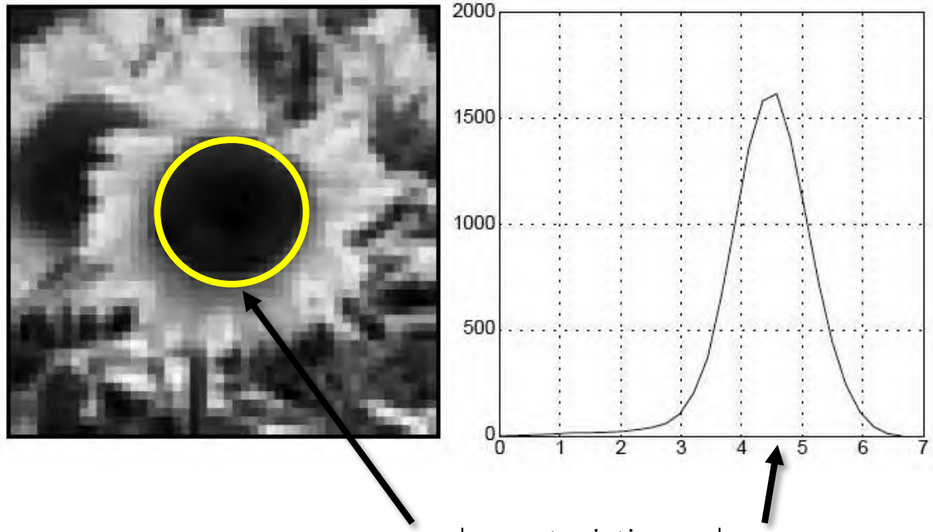
image





Laplacian

与角点相同，计算响应最大的尺度

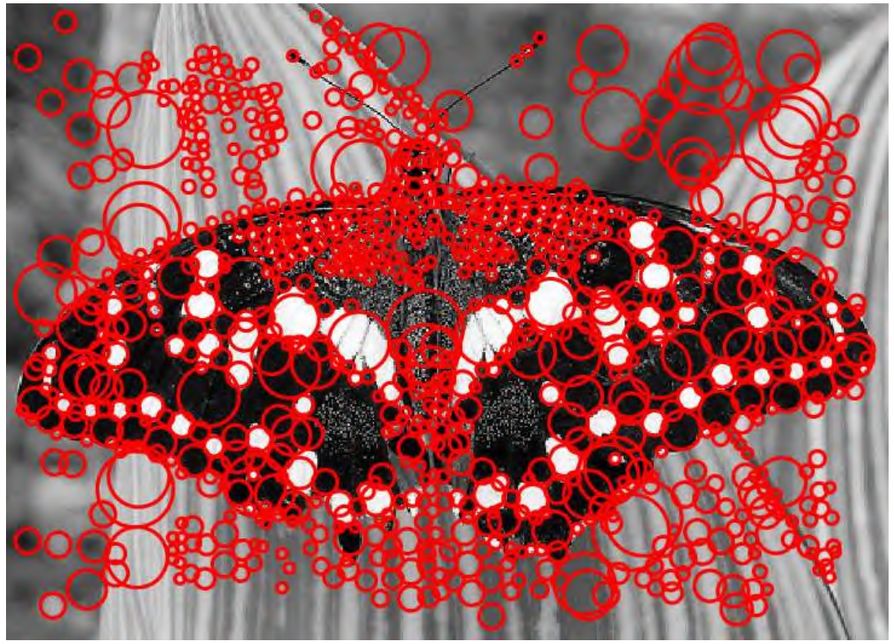


characteristic scale

- 在什么尺度下我们能在Laplacian算子下得到最大值?

T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

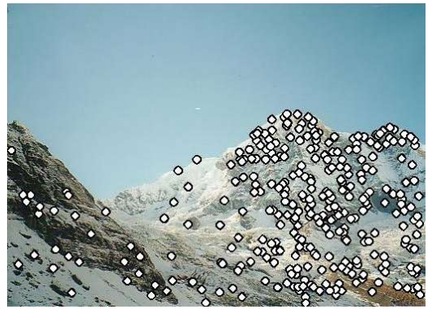
斑点检测：最终效果



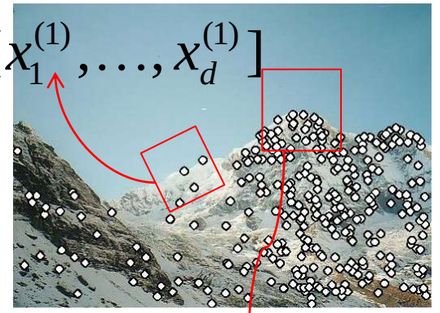
回顾———特征提取和匹配

基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点

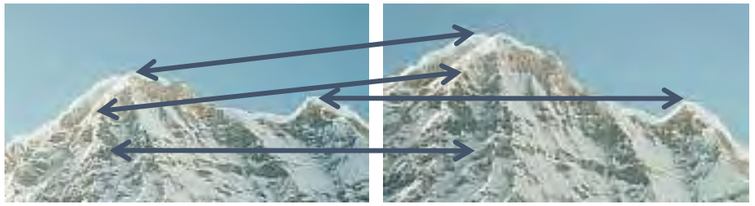


2) 解释 Description: 在关键点周围提取特征



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) 匹配 Matching: 根据两个视角下的特征进行匹配

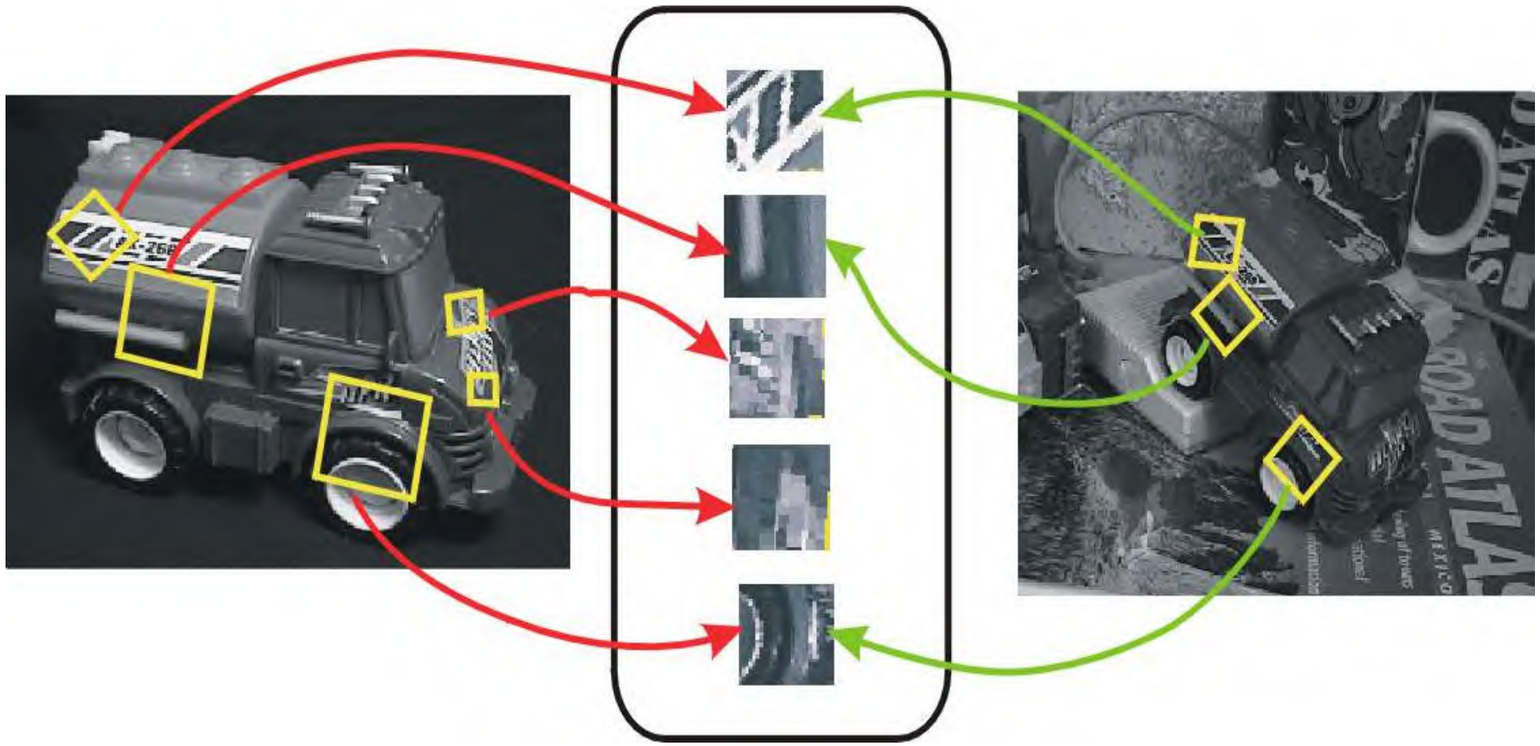


Kristen Grauman

回顾：不变局部特征

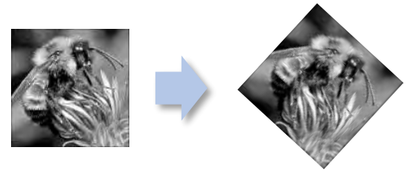
找到在图像变换下仍然保持不变的图像特征

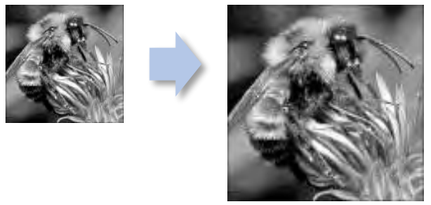
- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



特征提取器

- 几何层面

旋转 

尺度 

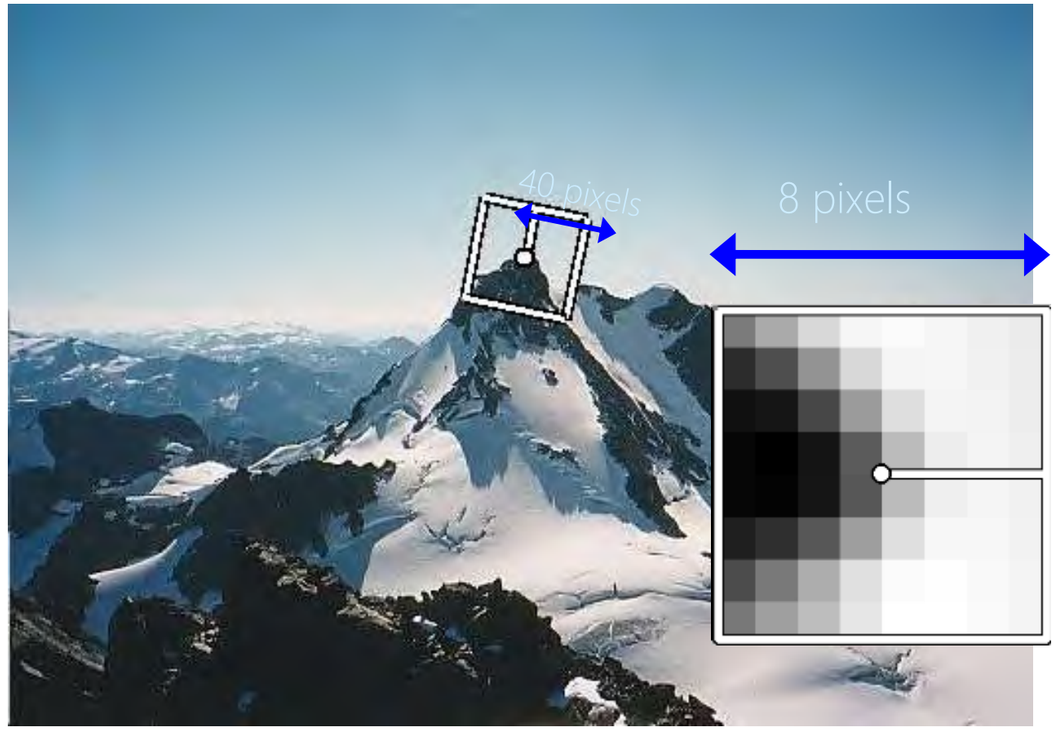
- 光学层面

强度变换 

Multiscale Oriented PatcheS descriptor 特征分解确定方向

使用 40x40 方形窗口提取特征

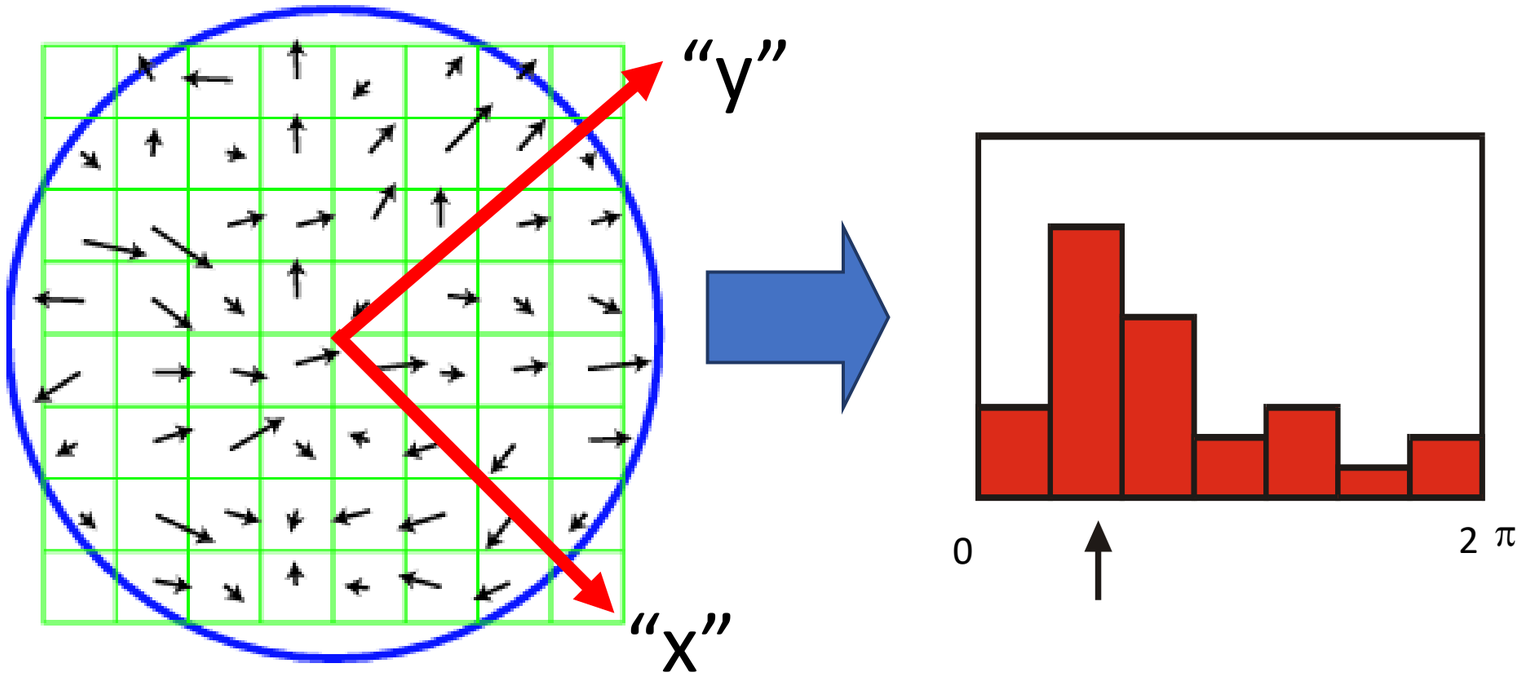
- 缩放至1/5
- 旋转至水平
- 将 8x8 的方形窗口作为特征
- 减去均值和方差保证强度不变性



Adapted from slide by Matthew Brown

Scale Invariant Feature Transform

用特征尺度确定窗口尺度
给定窗口，找到像素点梯度方向最多的方向
根据梯度方向旋转各个窗口



Slide credit: S. Lazebnik

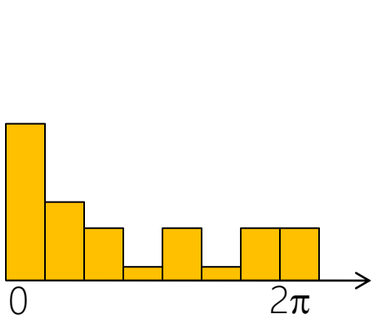
Scale Invariant Feature Transform

想法:

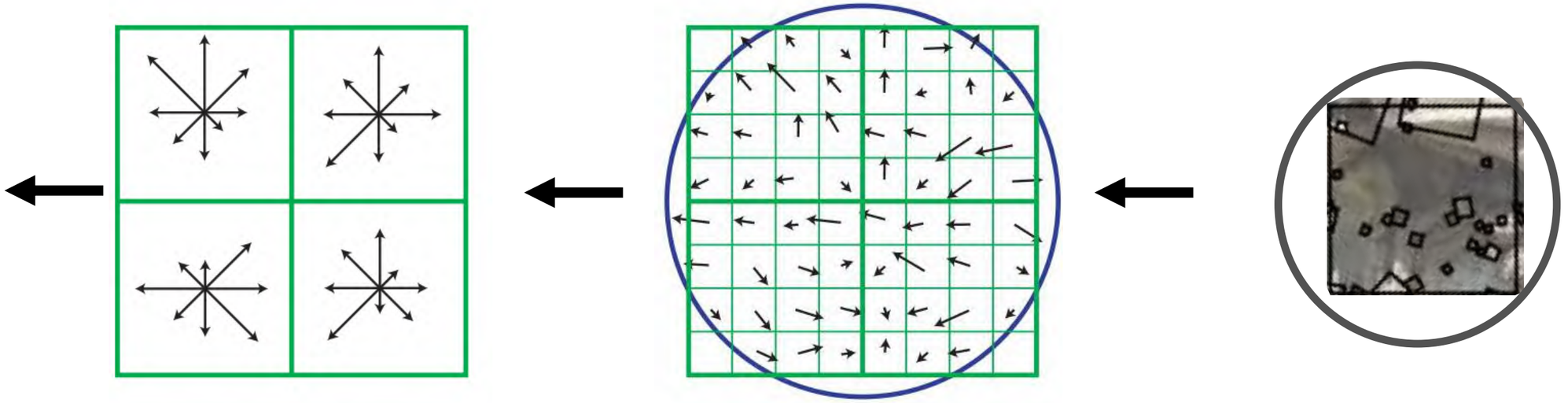
- 在检测到的特征周围取16x16的方形窗口
- 为每个像素计算边缘方向（梯度的角度 - 90度）
  - 减少亮度影响
- 排除弱边缘（梯度幅值低于阈值）
- 为剩余边缘方向创建直方图

做法:

- 把16x16的窗口分成4x4 的格点 (在这里画的是2x2)
- 计算每个点的方向直方图
- 16 个格点 * 8 个方向 = 128 维的特征



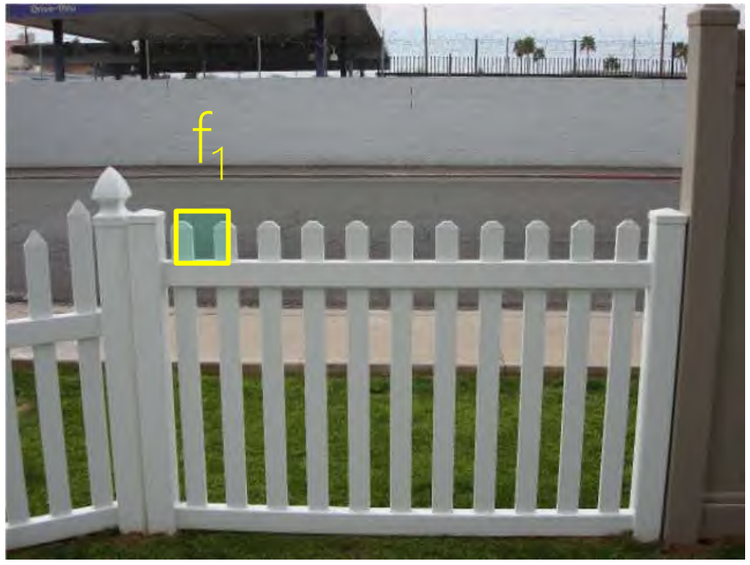
梯度角度直方图



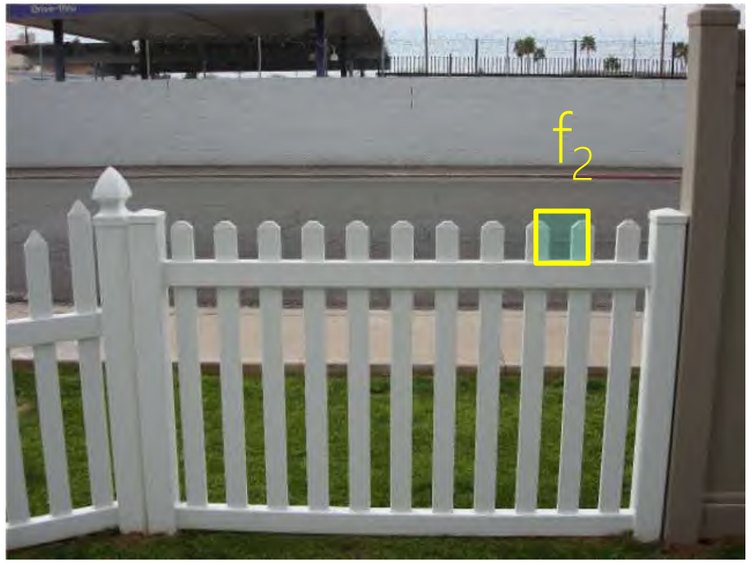
特征匹配——计算特征距离

怎么根据相似度匹配特征 f_1, f_2?

- 简单方法: L_2 distance, $\| f_1 - f_2 \|$
- 模糊匹配效果不好 (阈值法)



I_1

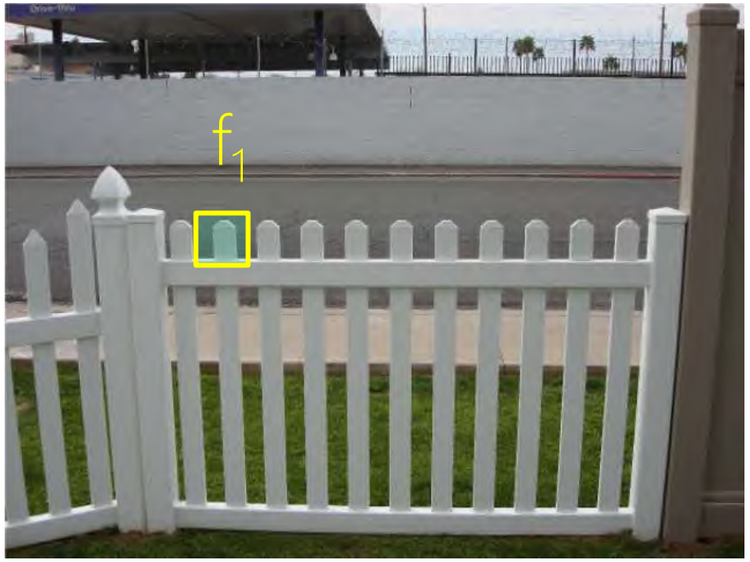


I_2

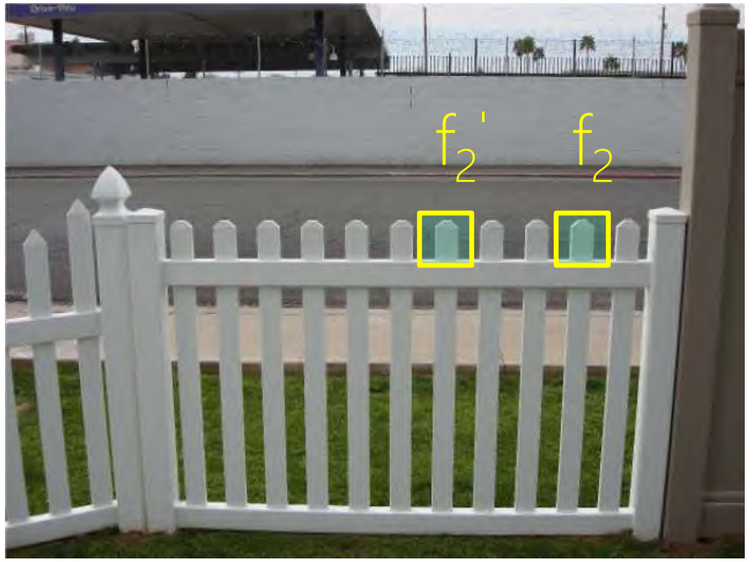
次近邻比率匹配

怎么根据相似度匹配特征 f_1, f_2?

- 高级方法 2nd Nearest Neighbor Trick： $= \| f_1 - f_2 \| / \| f_1 - f_2' \|$
  - f_2 是 I_2 最匹配 f_1 的像素点
  - f_2' 是 I_2 第二匹配 f_1 的像素点
  - 模糊匹配效果较好:如果一个特征与其最近邻的距离与其与第二近邻的距离相差很大，那么这个匹配很可能是正确的。相反，如果这两个距离相差不大，那么这可能是一个误匹配



I_1



I_2

回顾——图像变换

这两张图像之间的几何关系是什么?

对制作全景图很重要!

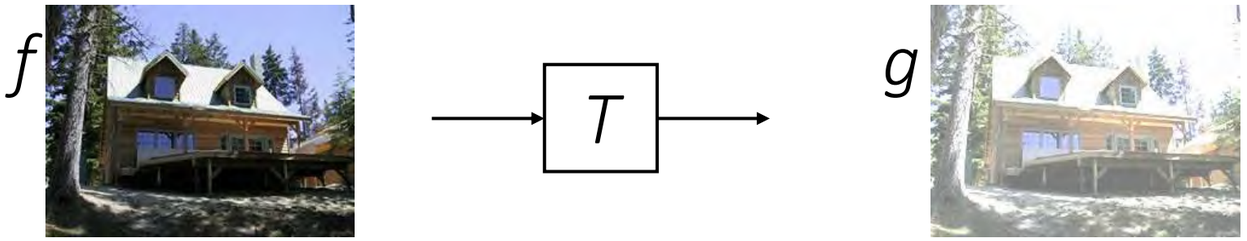
首先，我们需要知道这种转换是什么。

其次，我们需要弄清楚如何使用特征匹配来计算它。

图像变换

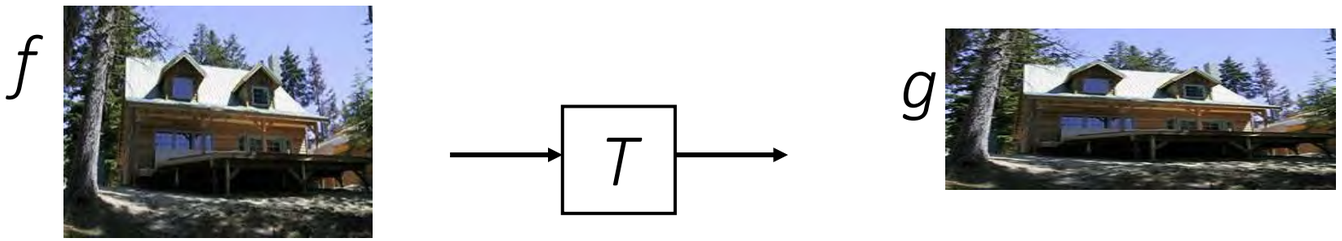
图像滤波：更改图像**值域**

$$g(x,y) = T(f(x,y))$$



图像扭曲：更改图像的**定义域**

$$g(x,y) = f(T(x,y))$$



我们暂时只关注 线性变换

$$\boldsymbol{p' \equiv Tp}$$

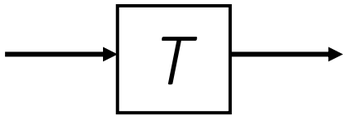
T:矩阵; p, p’: 2D点。

Slide credit: A. Efros

T 为 2x2 矩阵能做的变换

放缩矩阵

f 



g 

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

旋转矩阵

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

T 为 2x2 矩阵能做的变换

Identity 等价变换

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear 倾斜

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

T 为 2x2 矩阵能做的变换

Before

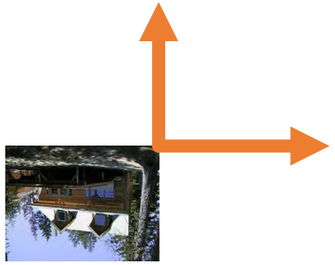
After

2D Y轴 镜像

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Before

After



2D 反转

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

T 为 2x2 矩阵时

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

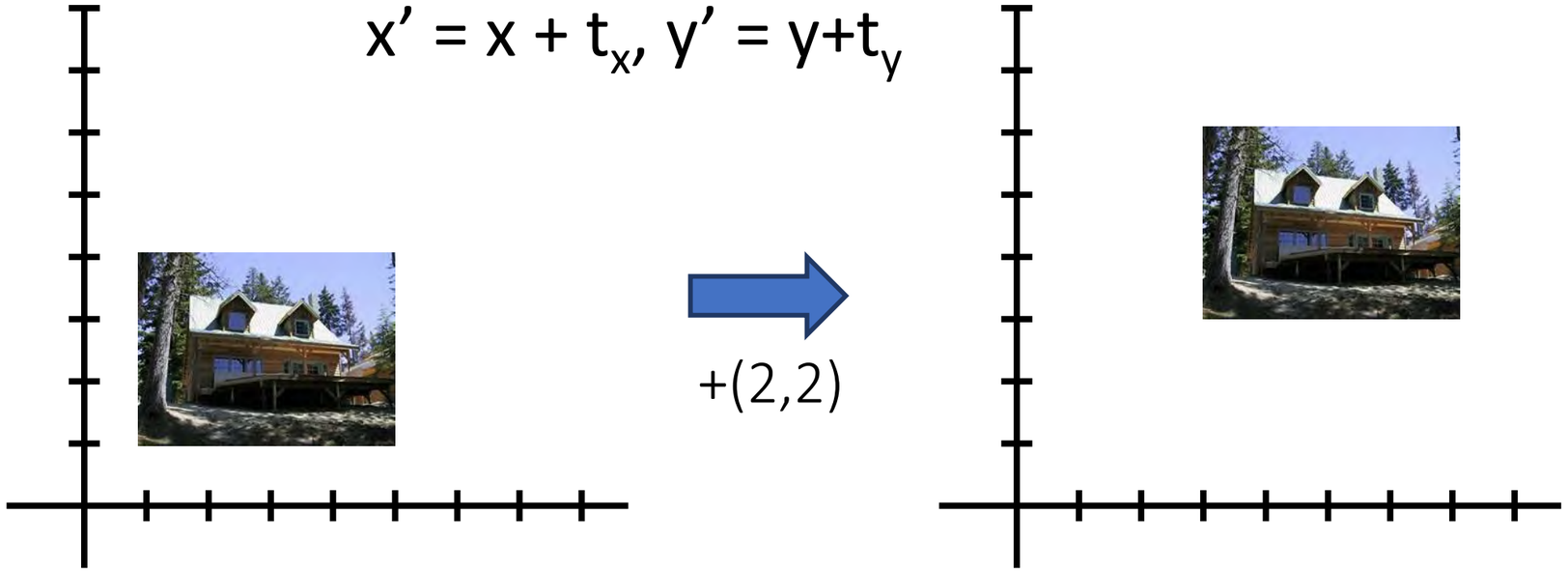
T 变换后

- 原点不变: **0 = T0**
  - 线还是线
- 平行线仍然平行

T 为 2x2 矩阵不能做平移变换，引入齐次坐标

平移怎么做?

$x' = x + t_x$, $y' = y + t_y$



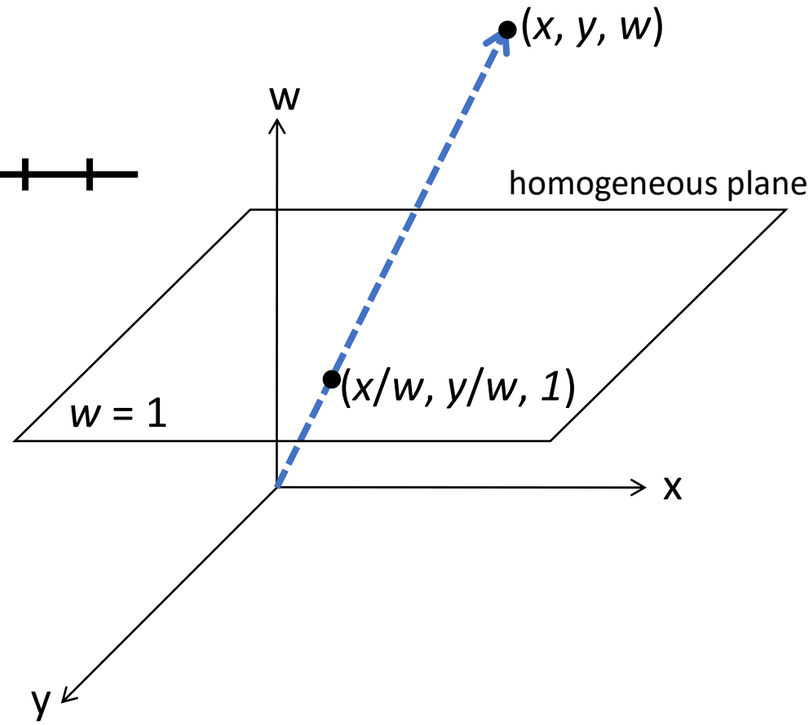
Trick: 增加一维坐标:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

齐次图像坐标

齐次坐标到图像坐标的转换

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$



仿射变换

- 仿射变换可以认为是以下变换的组合 ...
  - 线性变换
  - 平移

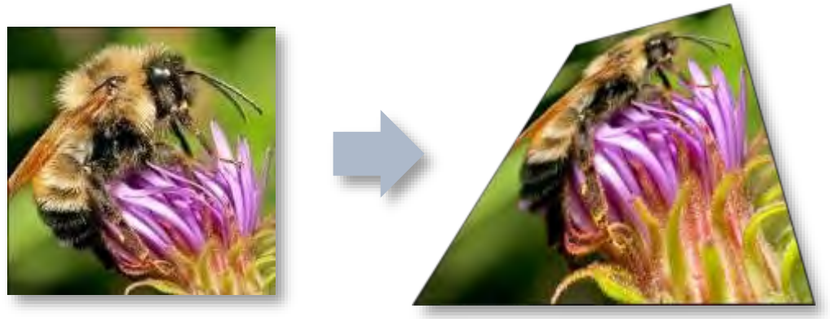
$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

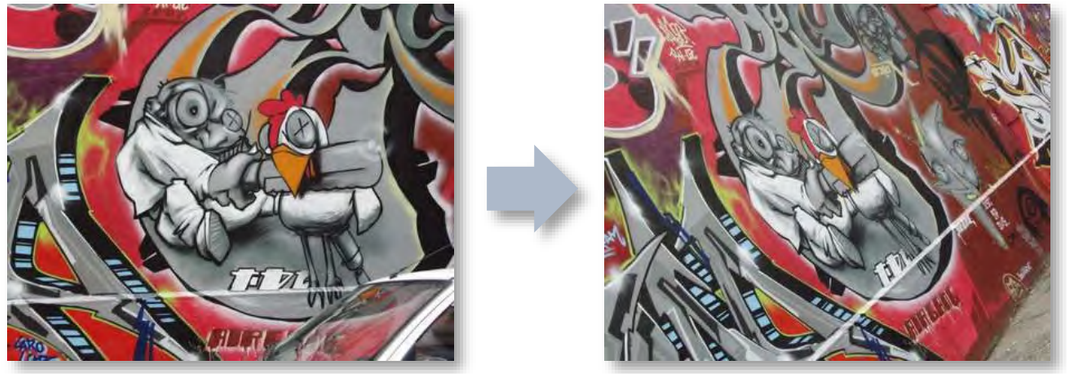
- 仿射变换的性质:
  - 原点变换后不一定在原点
  - 线仍然是线
  - 平行线仍然平行
  - 长度相对比例会保留
  - 仿射变换的组合还是仿射变换

Projective Transformations 投影变换
aka Homographies 同构变换
aka Planar Perspective Maps 平面透视映射

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

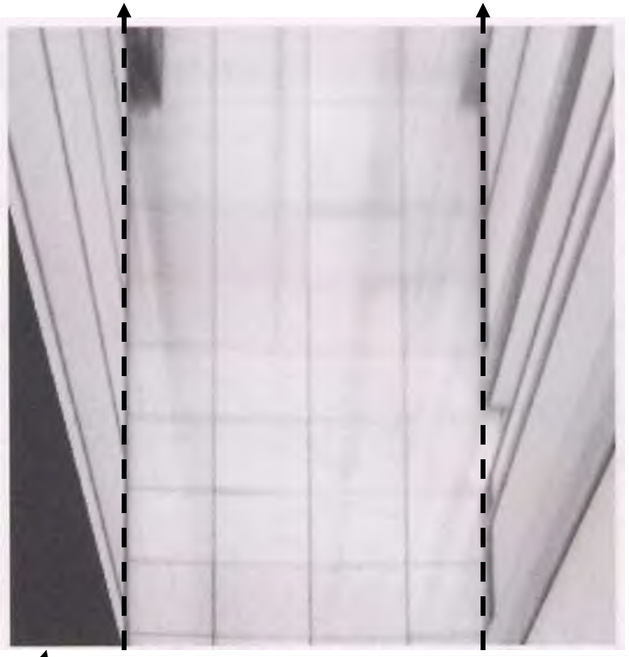
这种变换称为同构变换



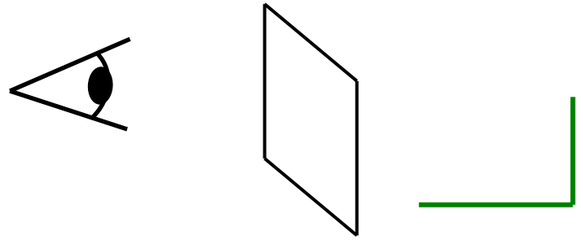


同构变换：透视投影——近大远小

$\mathbf{H}_1$



$\mathbf{H}_2$



正面视角的图像

黑色区域
代表没有
像素

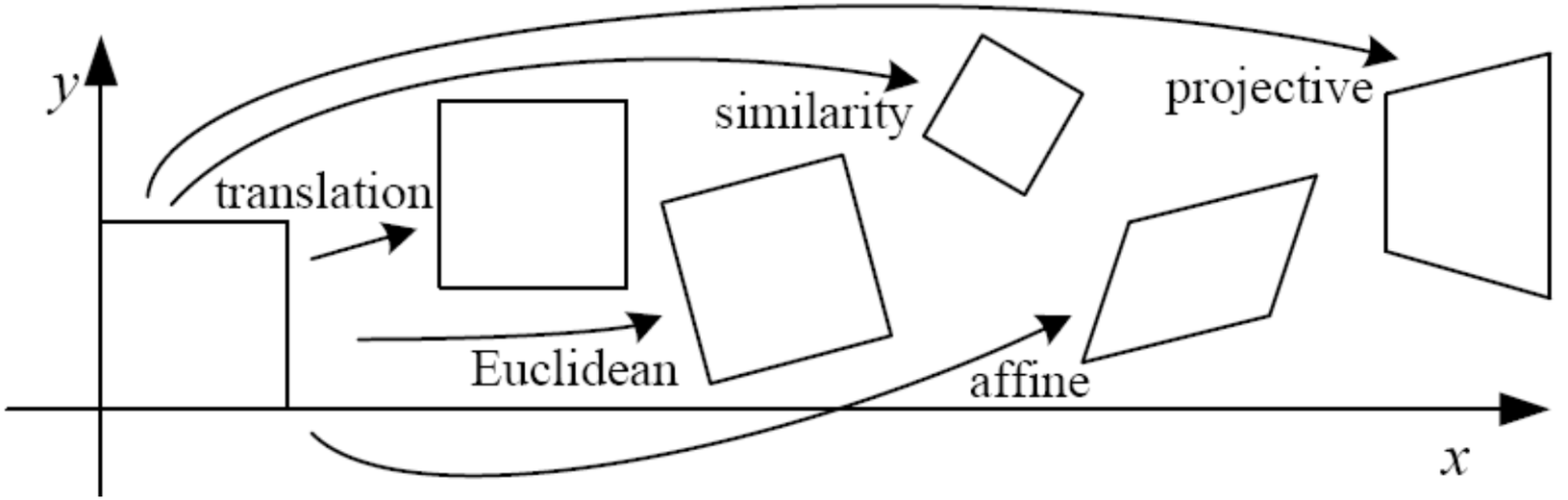
同构变换

- 同构变换包括...
  - 仿射变换
  - 透视变换

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

- 性质:
  - 原点变换后不一定是原点
  - 线仍然是线
  - 平行线不一定平行
  - 比例可能变化
  - 同构变换的组合还是同构变换

2D 图像变换

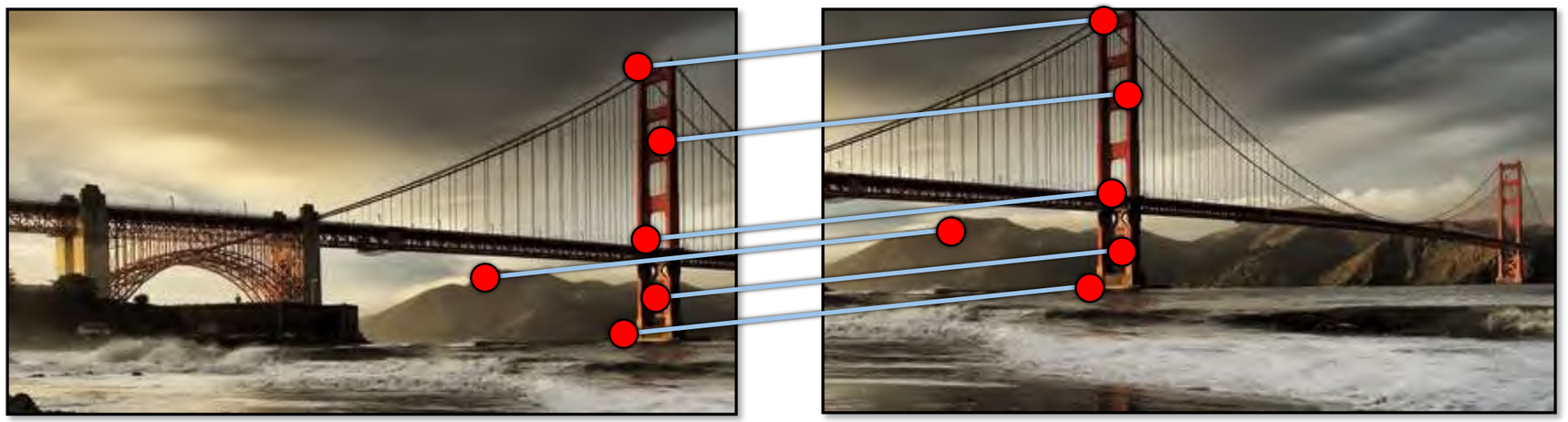


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

同构变换等于视角表你换 全景图是从不同视角采集的

如何计算变换参数?

简单情况：平移

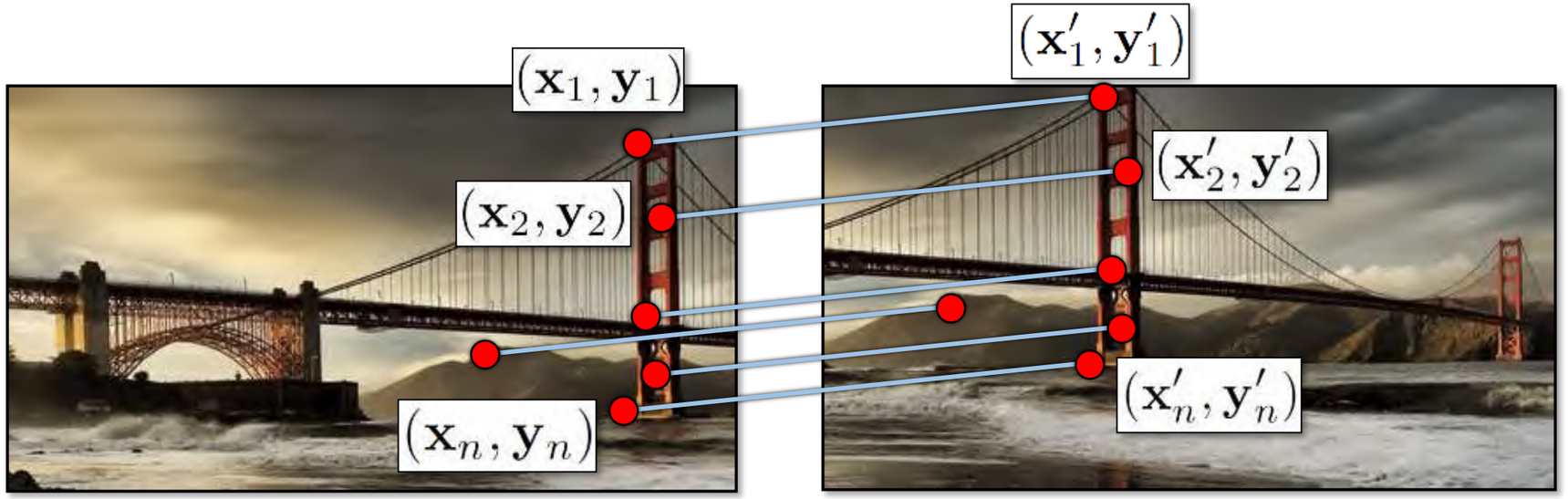


$(\mathbf{x}_t, \mathbf{y}_t)$

怎么求解？

$(\mathbf{x}_t, \mathbf{y}_t)$

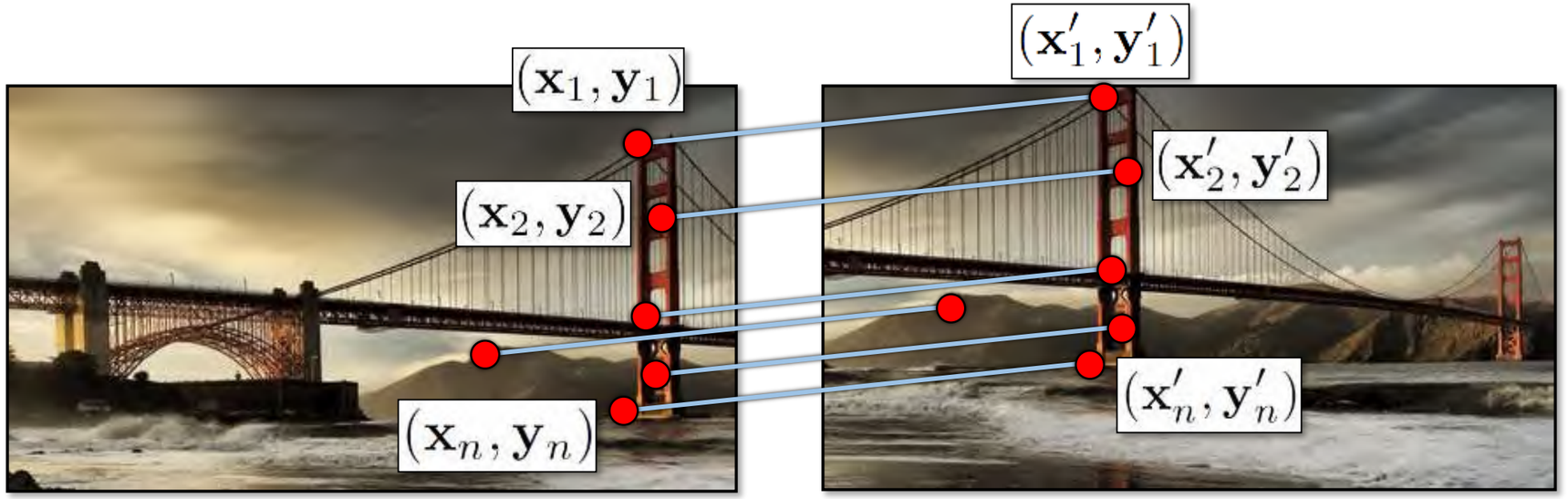
简单情况：平移



不匹配度 i = $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

简单情况：平移



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- 线性方程组
  - 已知数是什么？未知数是什么？
  - 有多少未知数？每次匹配有多少方程？

最小二乘

$$\mathbf{A}\mathbf{t} = \mathbf{b}$$

- 找到 t 最小化

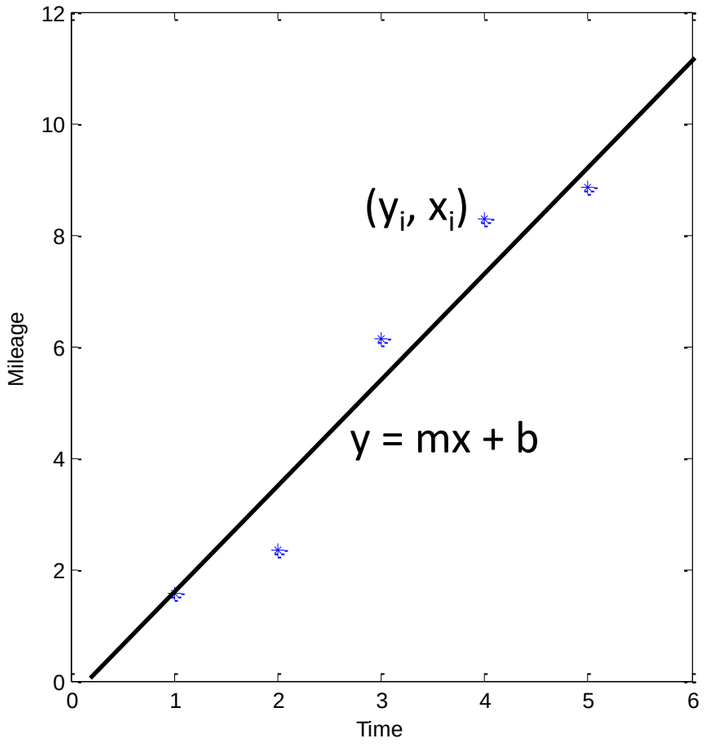
$$||\mathbf{A}\mathbf{t} - \mathbf{b}||^2$$

- 标准解

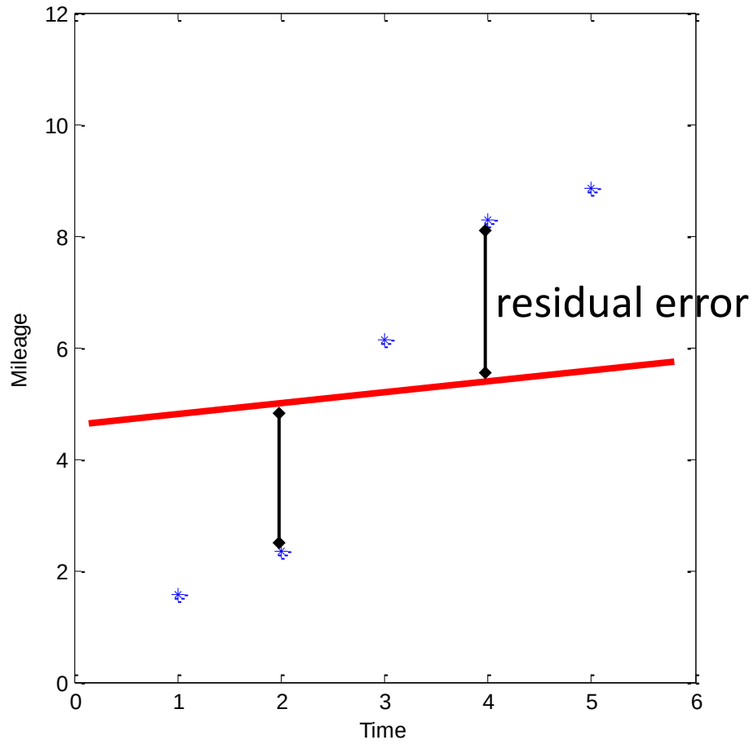
$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{t} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

最小二乘：线性回归

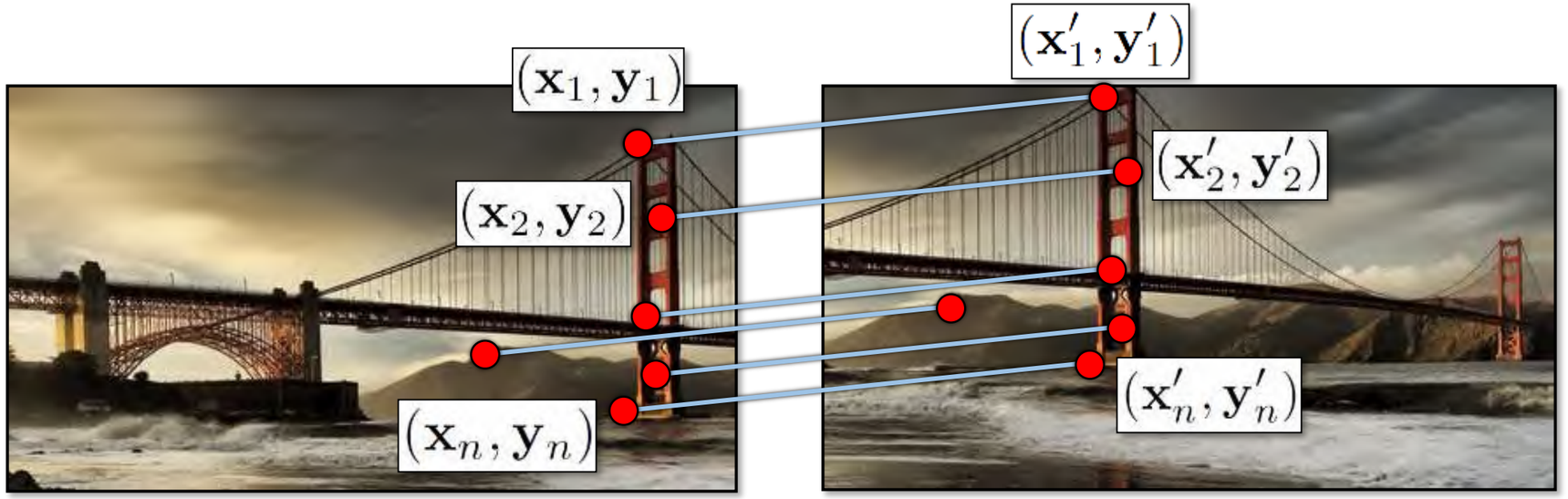


最小二乘：线性回归



$$\text{Cost}(m, b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

简单情况：平移



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- 问题：方程多于未知数
  - “过度确定的”方程组
  - 最小二乘解

最小二乗解

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

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

求解同构映射

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

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

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

不是线性的!

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

求解同构映射

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

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

求解同构映射

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

图像对齐算法

给出两张图像 A 和 B

1. 提取 A 和 B 的特征
2. 匹配 A 和 B 的特征
3. 根据最小二乘解计算A和B之间的同构映射

“感知”

计算机系统对数字图像和视频进行分析和理解的能力，以获得有关物体、场景、特征和动作的信息。感知的目标是模拟和模仿人类视觉系统，使计算机能够理解和解释图像中的内容。

特征信息提取、对象识别、场景理解、运动分析、三维深度感知、上下文感知 .....

什么是 “识别”?

Next few slides adapted from Li, Fergus, & Torralba’s excellent short course on category and object recognition

什么是 “识别”?

- Verification: is that a lamp?
  - 验证：那是一盏路灯吗?

什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
  - 检测：人们在哪里?

什么是 “识别”?

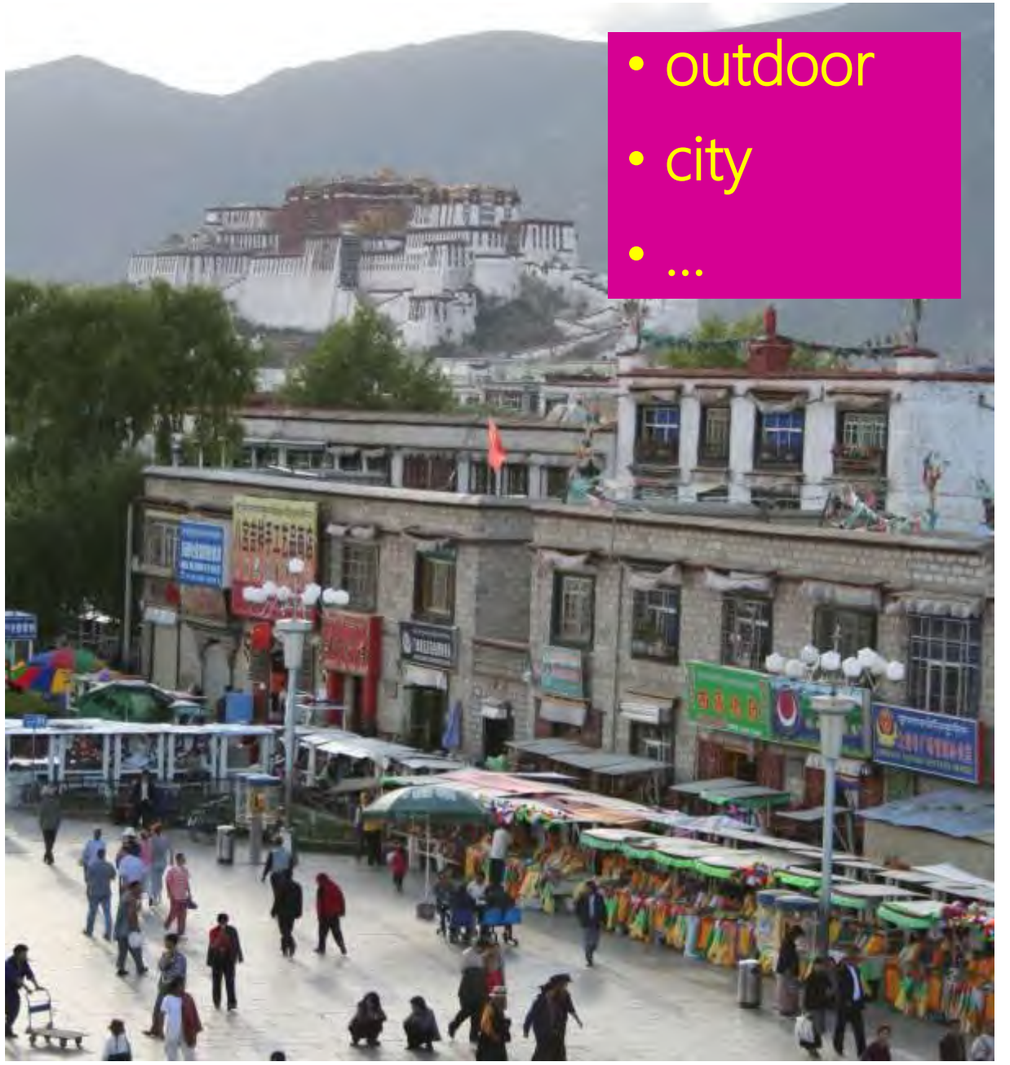
- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
  - 辨认：那是布达拉宫吗?

什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
  - 对象分类

什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
- Scene and context categorization
  - 场景识别与理解



什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
- Scene and context categorization

- Activity / Event Recognition
  - 动作、事件识别

难点：不同视角

Michelangelo 1475-1564

难点：不同光照条件

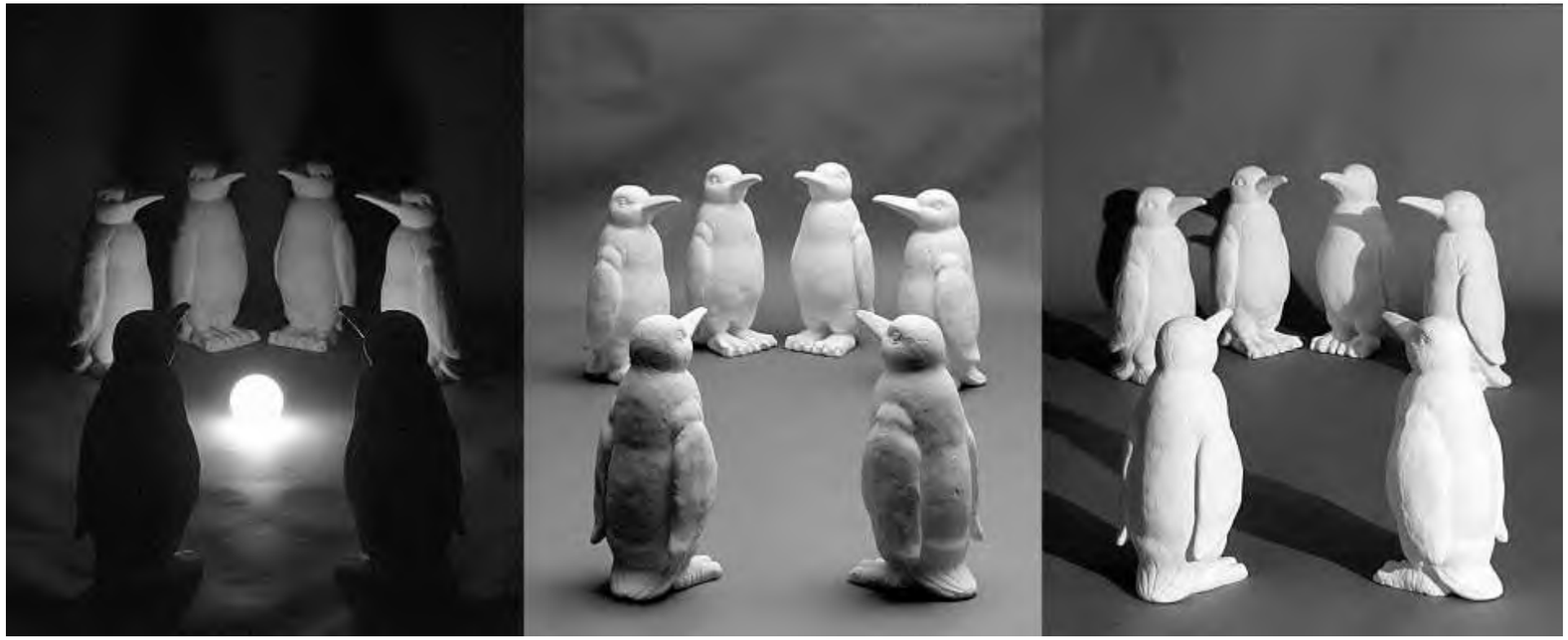
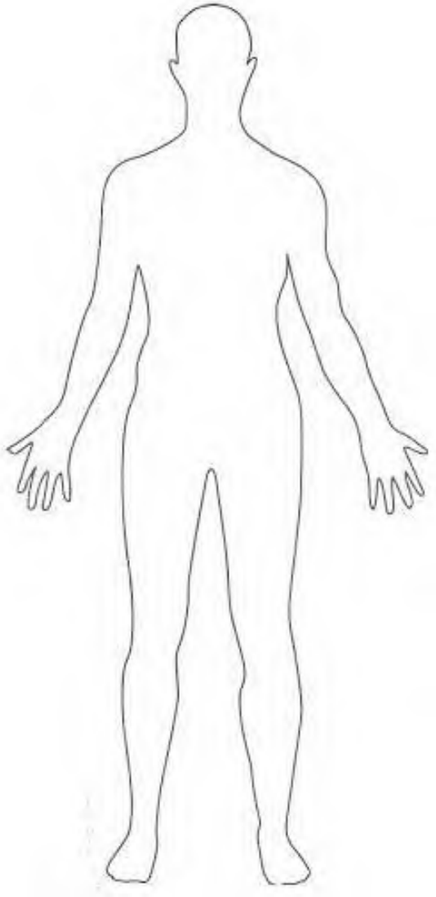


image credit: J. Koenderink

难点：尺度

难点：不同形式




London 2012
London 2012
Punch Layout
Arabian
Double Front
Roundoff,
Back Handspring
Roundoff,
One-and-a-half Stepout
© Corbis/Getty Images

难点：遮挡

Magritte, 1957

难点：背景伪装

Kilmeny Niland. 1995

难点：类内差异

同类目标外观差异很大

难点：类内差异

让我们从最简单的分类开始

分类最简单的情况：二分类

我们会用一个特征向量
代表这张图

分类最简单的形式

记住所有类别的训练样本

If this:
cat.

If this:
dog.

If this:
hippo.

分类最简单的形式

这么做有什么问题?

Rule: if this,
then cat

似乎跟记忆不太一样?

分类最简单的形式：最近邻

训练图像
和对应标签

测试图像

$\boldsymbol{x}_1$ ← $D(\boldsymbol{x}_1, \boldsymbol{x}_T)$ → $\boldsymbol{x}_T$

$D(\boldsymbol{x}_N, \boldsymbol{x}_T)$

...

$\boldsymbol{x}_N$

(1) 计算特征向量的距离（特征匹配）(2) 找到训练集里最相似的样本 (3) 使用最相似样本的标签.

最近邻算法

训练 ($\mathbf{x}_i$,y_i): 记住所有训练样本

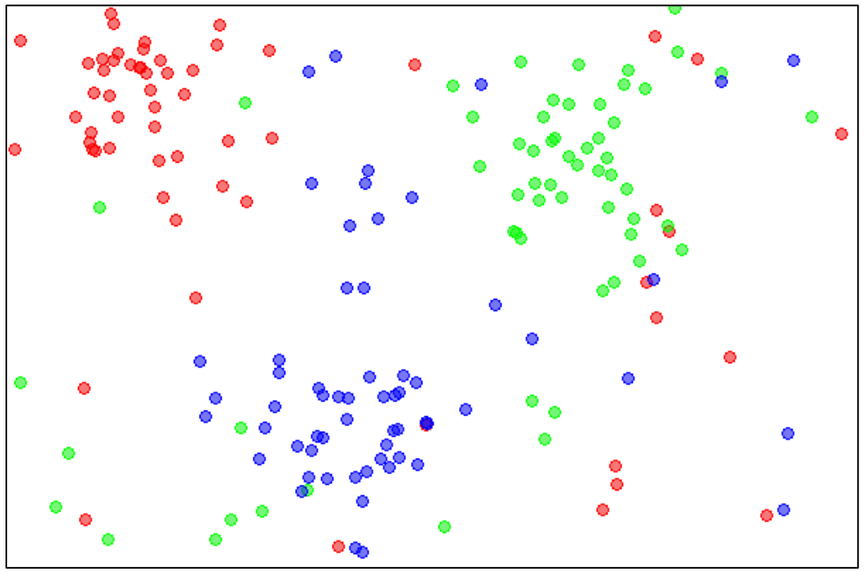
测试 (x):

```
bestDist, prediction = Inf, None
for i in range(N):
    if dist(x_i,x) < bestDist:
        bestDist = dist(x_i,x)
        prediction = y_i
```

最近邻算法

可能出现什么问题?

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

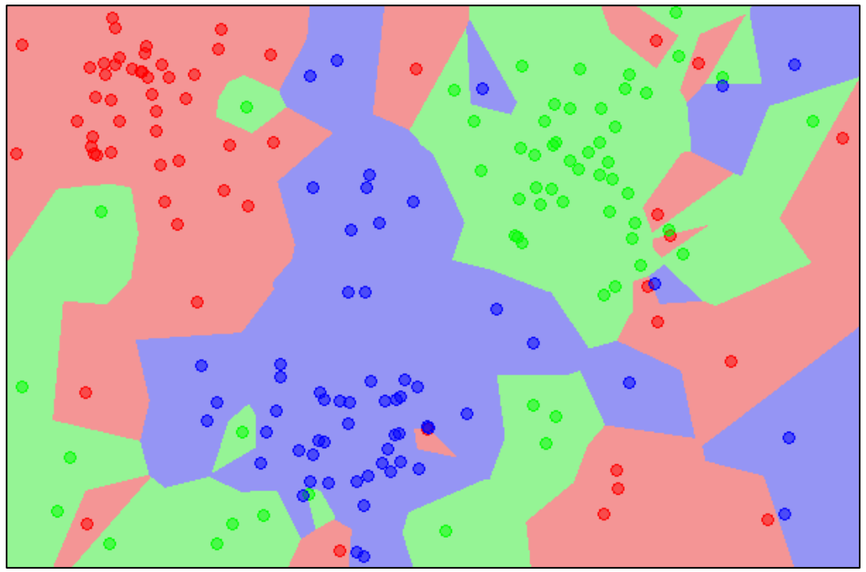
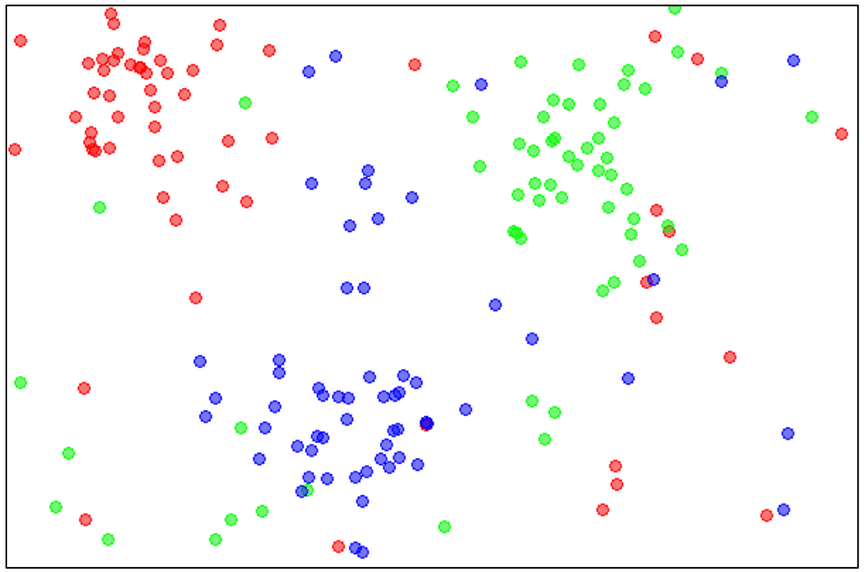


Diagram Credit: Wikipedia

K近邻

找到前K近邻样本，然后投票决定标签

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

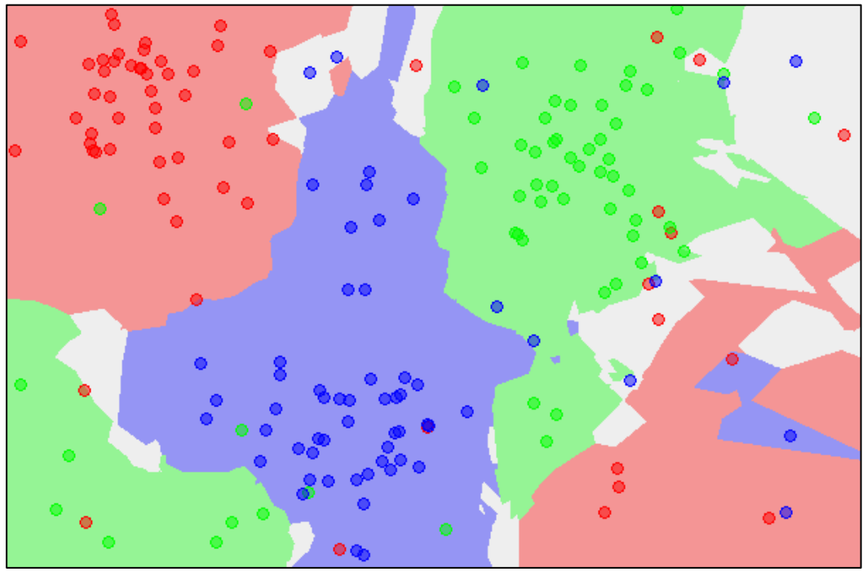
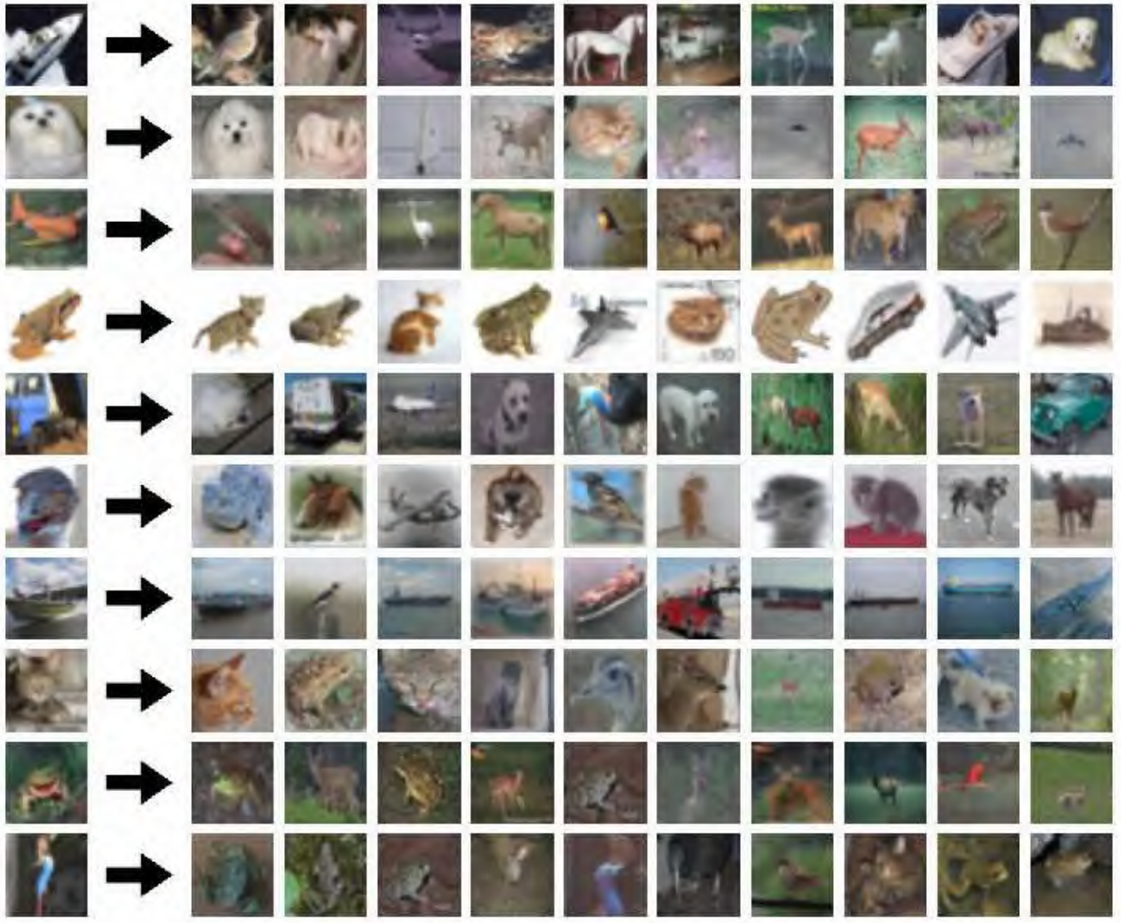


Diagram Credit: Wikipedia

What does this look like?

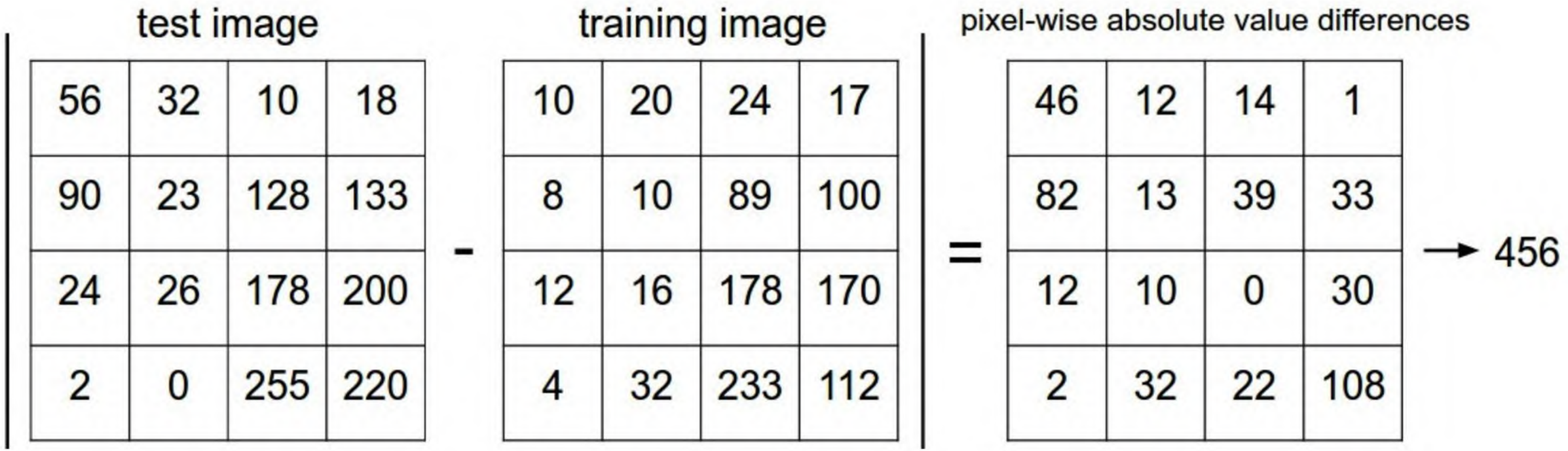


What does this look like?

How to Define Distance Between Images

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

Where I_1 denotes image 1, and p denotes each pixel



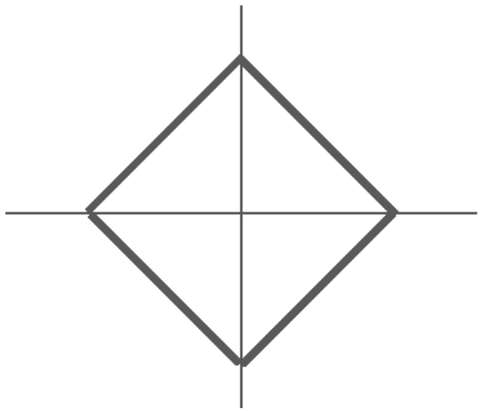
Slides from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

Choice of distance metric

- Hyperparameter

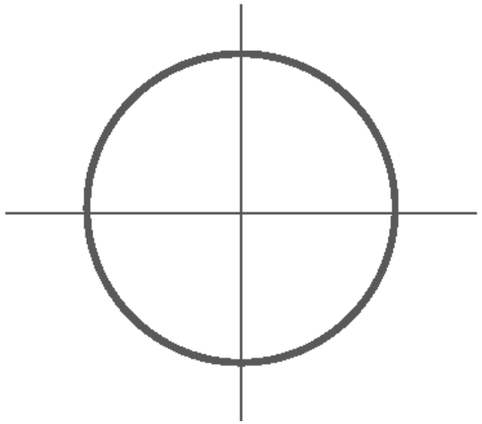
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



- Two most commonly used special cases of p-norm

$$||x||_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \qquad p \geq 1, x \in \mathbb{R}^n$$

Slide composited from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

K = 1

L2 (Euclidean) distance

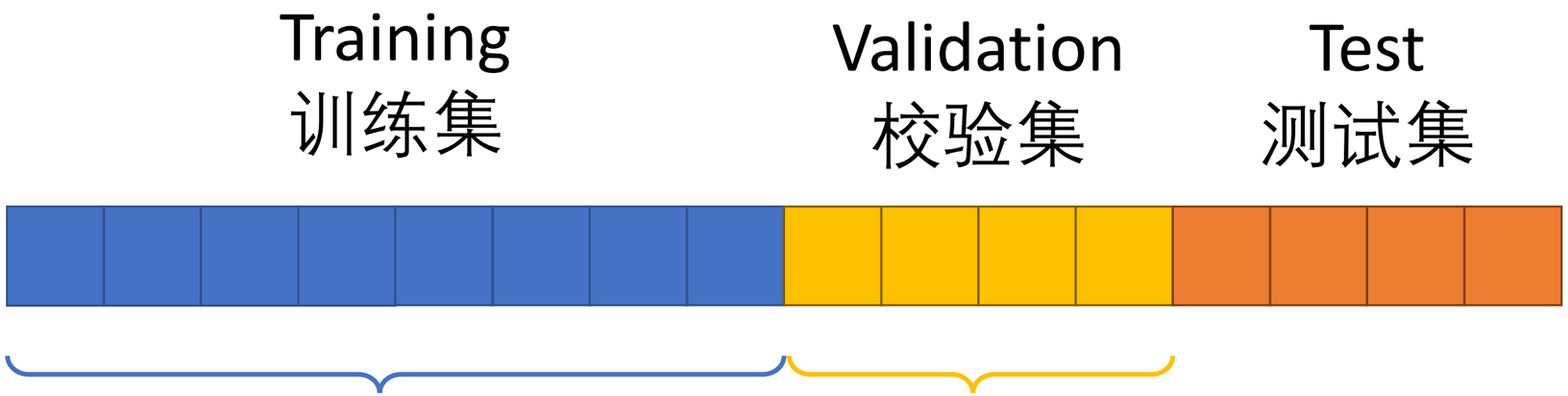
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

K = 1

Demo: http://vision.stanford.edu/teaching/cs231n-demos/knn/

K近邻

怎样确定距离度量? 怎样确定K?



用训练集做查找库

用校验集确定 k
和度量方式

K近邻

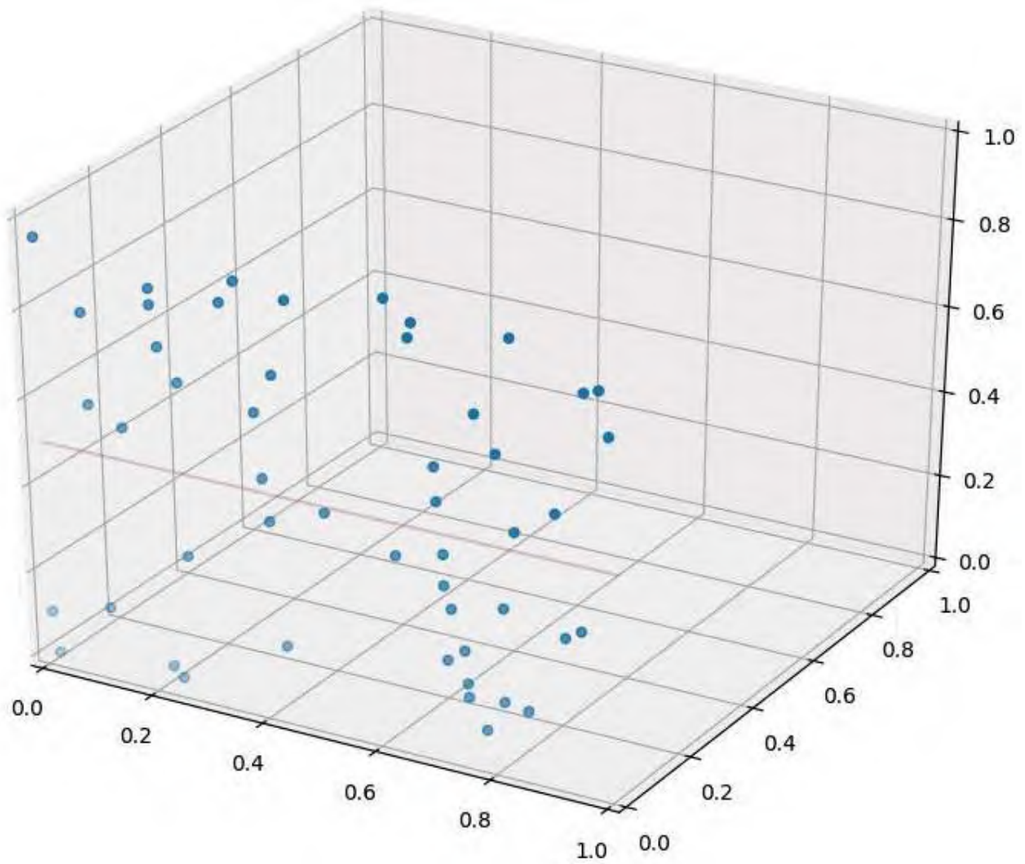
- 虽然没有学习过程，但通常是有效的
  - 不学习参数和结构，没有实际的训练过程
- 对于每个任务都使用相同的算法
- 当数据点数量趋近于无穷时，错误率保证最多比数据上的最优解差2倍
  - **局部决策**：K-NN在决策时只考虑局部信息，即输入点的K个最近邻。因为它是基于局部信息做出决策的，所以当数据量很大时，它能够捕捉到数据的微小细节和模式。
  - **大数定律**：当训练数据量增加时，每个点的近邻都更加可能代表真实的数据分布。这意味着K-NN的决策边界变得更加准确。
  - 当K同时增长并且K/N趋近于0，其中N是数据点的数量
  - K-NN的错误率会收敛到贝叶斯最优错误率
  - **看似很好，但是有个严重的问题…**

KNN的问题：距离度量

- 测试时性能糟糕
- 基于整体图像的距离度量可能非常不直观

KNN的问题：维度的诅咒

- 随着维度数量的增加，相同数量的数据变得更为稀疏。
  - 数据点之间距离变得均匀
- 我们所需的数据量随维度数量指数级增长。
  - 决策“边界”不稳定



Animation from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

用最小二乘法解决分类问题

将分类视为回归问题: x_i 是图像特征; y_i 为 1 如果图像是猫, 为 0 如果图像不是猫. 最小化均方误差.

训练 $(\mathbf{x}_i, y_i)$:

$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x_i} - y_i\|^2$$

推断 (x):

$$\boldsymbol{w}^T \boldsymbol{x} > t$$

Rifkin, Yeo, Poggio. *Regularized Least Squares Classification* (http://cbcl.mit.edu/publications/ps/rlsc.pdf). 2003

Redmon, Divvala, Girshick, Farhadi. *You Only Look Once: Unified, Real-Time Object Detection.* CVPR 2016.

线性模型

设定：三分类模型



模型：每个类别一个权重

$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

全部参数: $\boldsymbol{W_{3xF}}$ where **x** is in R^F

线性模型

Cat weight vector	0.2	-0.5	0.1	2.0	1.1	56
Dog weight vector	1.5	1.3	2.1	0.0	3.2	231
Hippo weight vector	0.0	0.3	0.2	-0.3	-1.2	24
						2
						1

W

x_i

-96.8	Cat score
437.9	Dog score
61.95	Hippo score

Wx_i

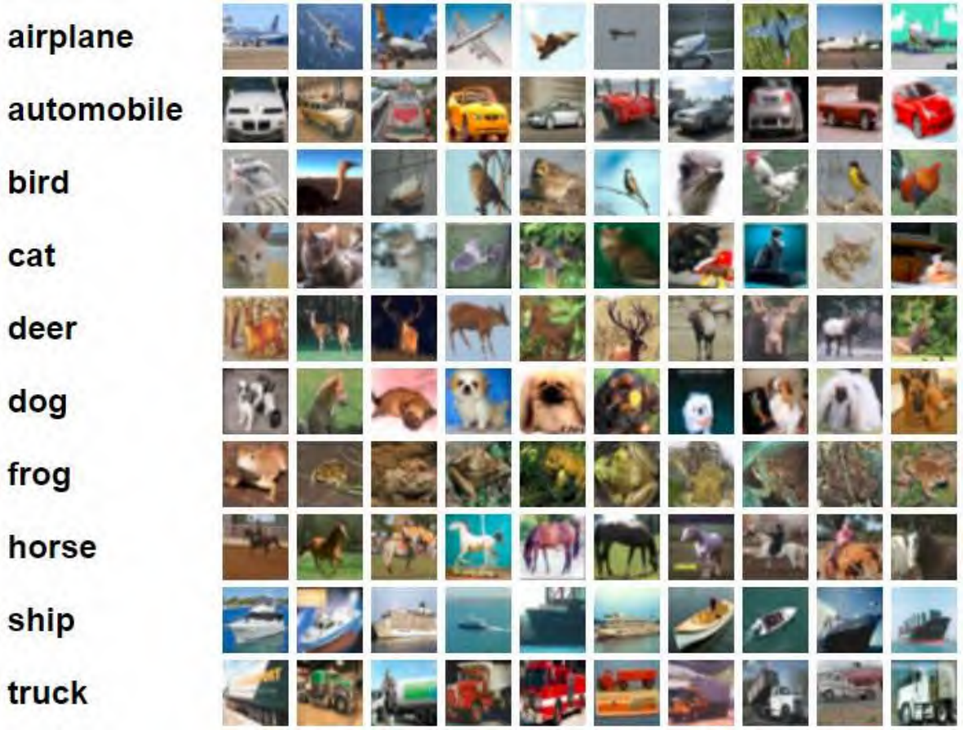
权重矩阵可以认为是“评分函数”

预测结果是一个向量，
代表每个类别的评分

Diagram by: Karpathy, Fei-Fei

视觉层面的直观理解

CIFAR 10:
32x32x3 Images, 10 Classes



- 把每张图像的像素平铺作为特征向量
- 拟合一个10分类的线性模型
- 根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

Slide credit: Karpathy & Fei-Fei

猜类别?

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.
让我们可视化$\mathbf{w}$

Deer or Plane?

Diagram credit: Karpathy & Fei-Fei

猜类别?

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.
让我们可视化$\mathbf{w}$

Ship or Dog?

Diagram credit: Karpathy & Fei-Fei

可视化线性分类器

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

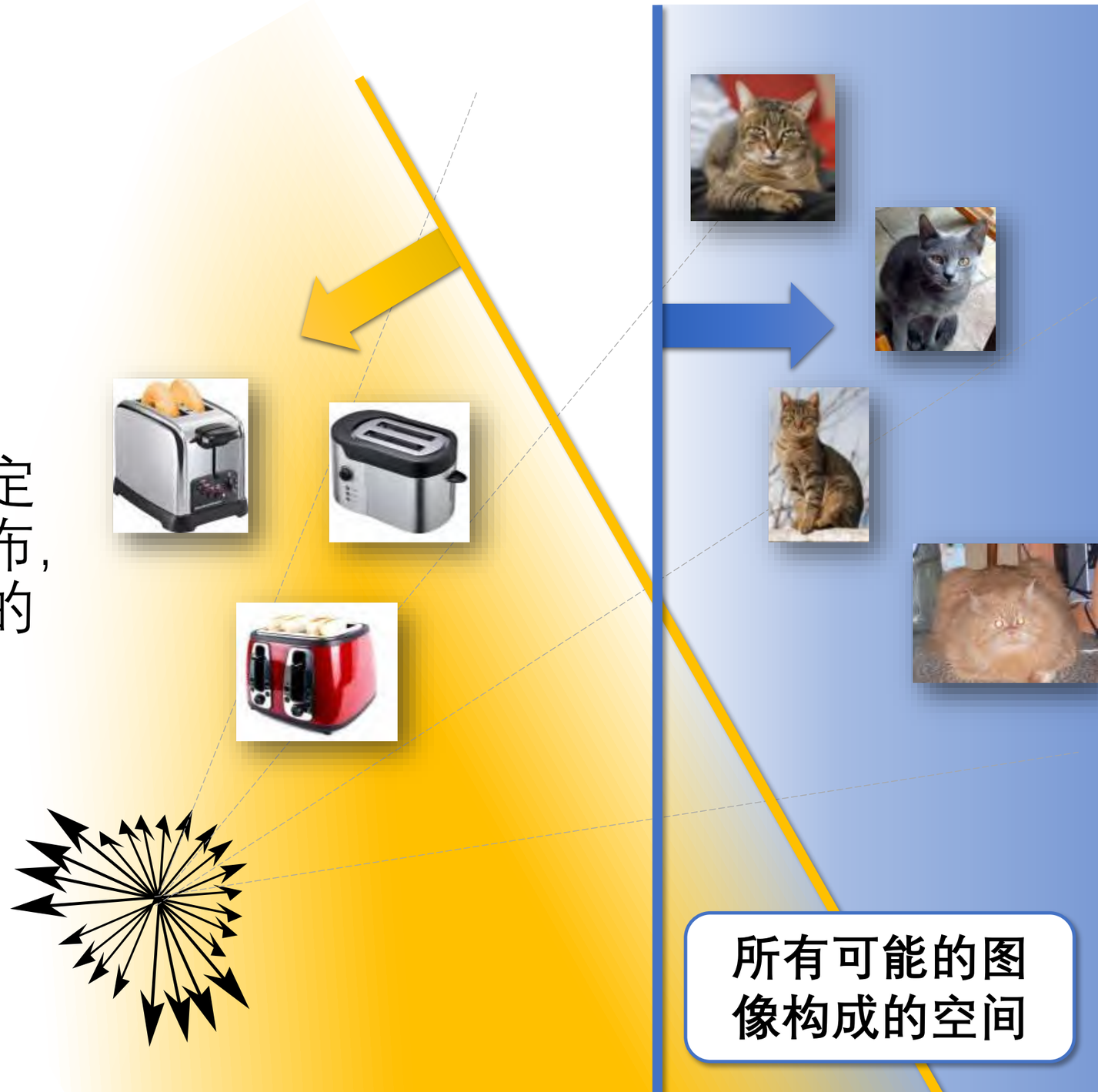
Diagram credit: Karpathy & Fei-Fei

解释：几何角度

- 参数为每个类定义一个超平面:

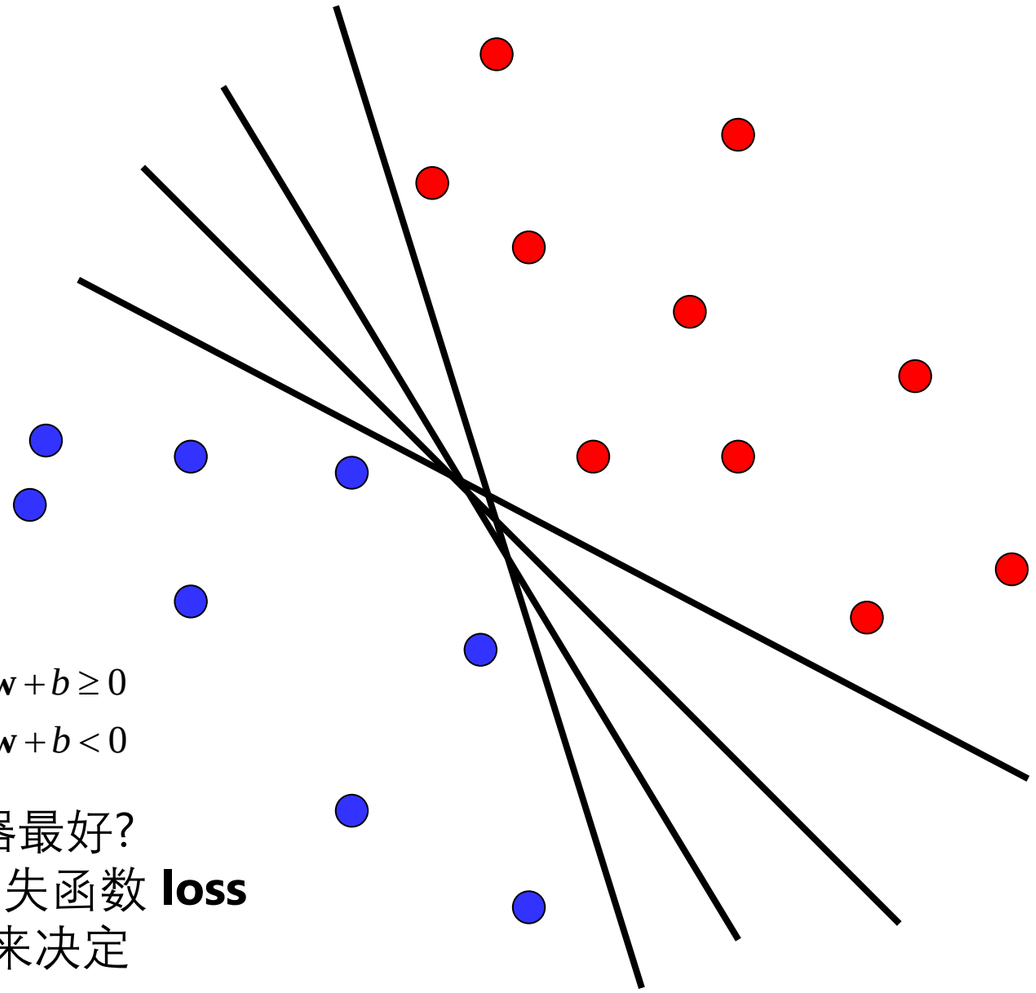
$$f(x_i, W, b) = Wx_i + b$$

- 我们可以将每个类的得分视为定义了一个与其距离成正比的分布，距离是指从对应的超平面到点的距离。



解释：几何角度

- 找到线性分类器（找平面）可以分开正负样本



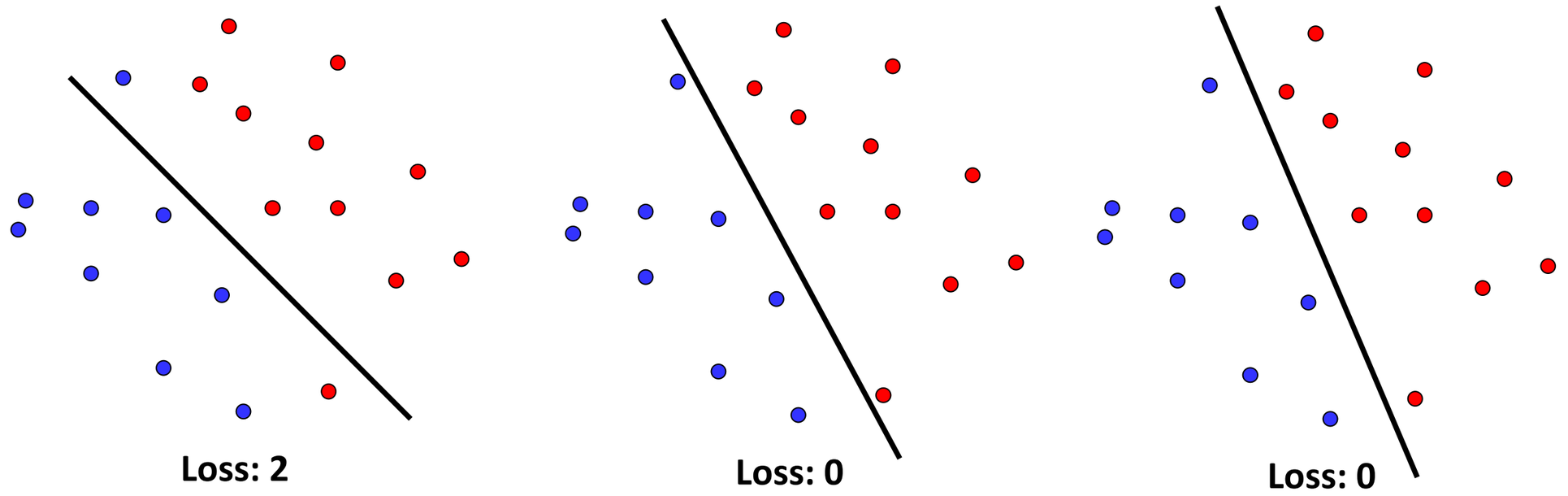
$\mathbf{x}_i$ positive： $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative： $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

哪个分类器最好?
我们需要一个损失函数 **loss function** 来决定

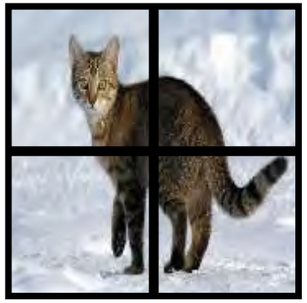
什么是好的损失函数?

- 检视分类错的样本
  - 问题：样本是离散的，我们没办法区分相似的分类器
  - 我们需要更好的 **泛化性** *generalization*
  - 还要处理多于两类的情况



如何设计损失函数 —— 最大间距

这样有什么问题?



Loss: dog score – cat score

怎么定义“dog” vs “cat”的评分?

$$(\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1$$

避免优化到负无穷

$$\max(0, (\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1)$$

$(\boldsymbol{Wx})_1$	-96.8	Cat score
$(\boldsymbol{Wx})_2$	437.9	Dog score
$(\boldsymbol{Wx})_3$	61.95	Hippo score

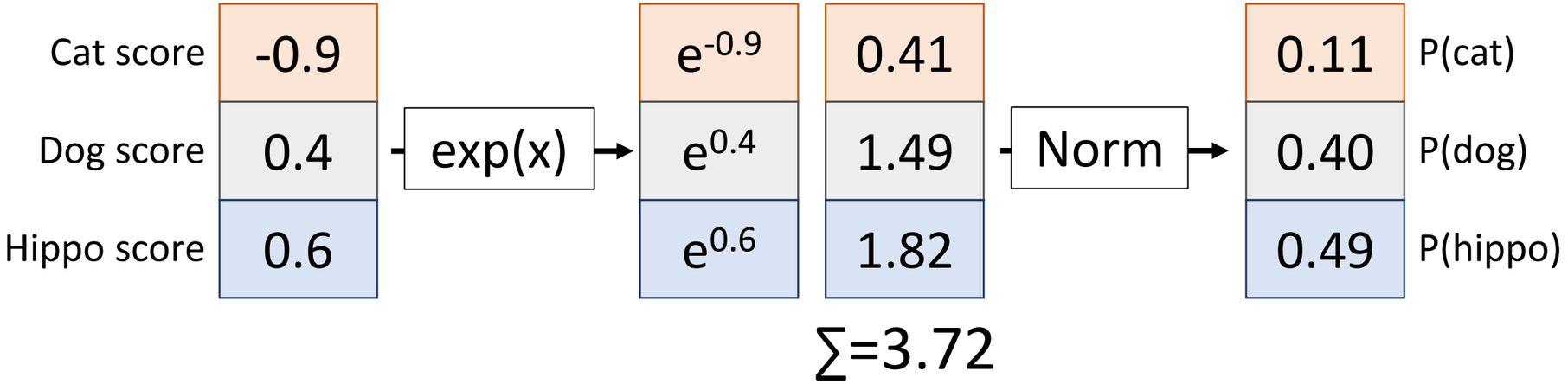
$$\boldsymbol{Wx}$$

预测结果是一个向量,
第j个值代表j类别的评分

Diagram by: Karpathy, Fei-Fei

Softmax

把评分转化为“概率分布”



P(class j): $\frac{\exp((Wx)_j)}{\sum_k \exp((Wx)_k)}$

Softmax

推断时 (x): $\arg\max_{k} (\boldsymbol{W}\boldsymbol{x})_k$ (找到评分最高的那一类)

P(class j): $\frac{\exp((Wx)_j)}{\sum_k \exp((Wx)_k)}$

为什么我们在测试时可以省略 exp/sum exp ?

Softmax

推断时 (x): $\arg\max_k (\boldsymbol{W}\boldsymbol{x})_k$ (找到评分最高的那一类)

训练时 (x_i,y_i):

$$\arg\min_{\boldsymbol{W}} \boldsymbol{\lambda}\|\boldsymbol{W}\|_2^2 + \sum_i^n -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)$$

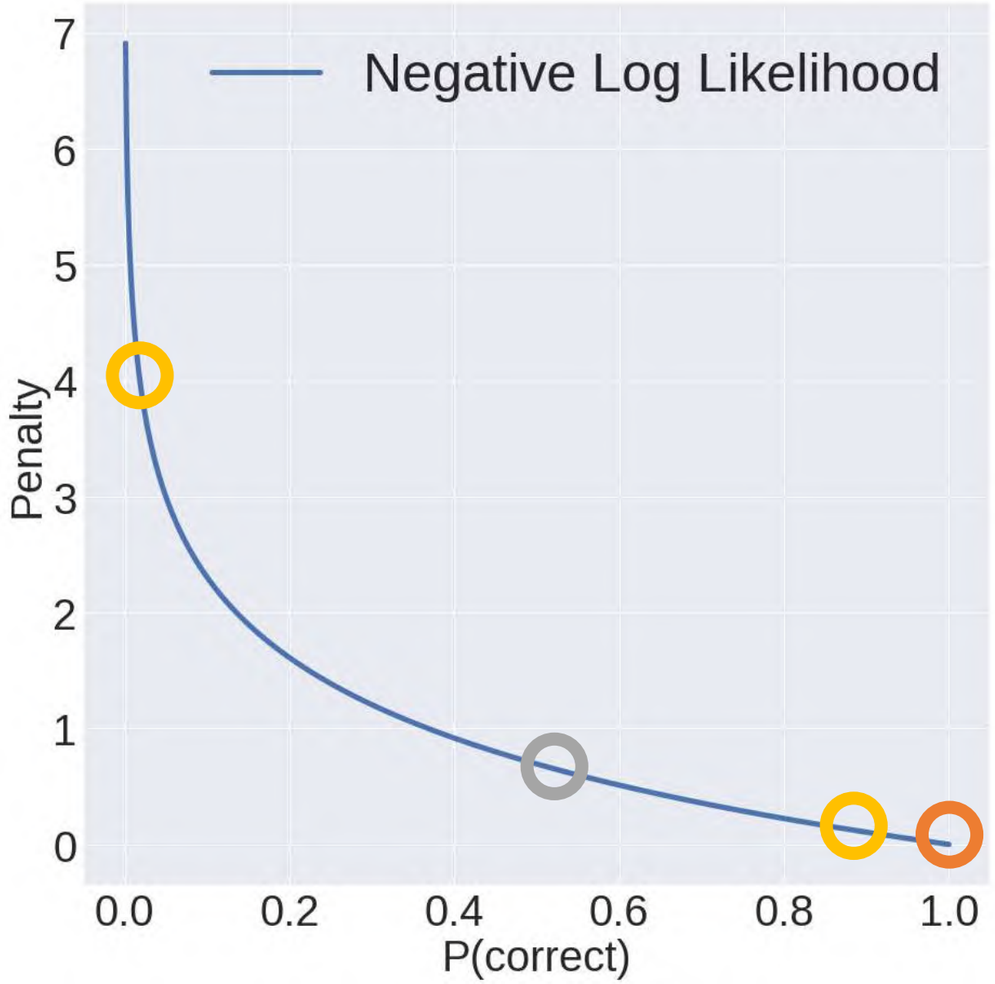
P(correct class)

Regularization
正则项

对所有训练样本

对正确分类对应的 negative log-likelihood 进行损失优化

Softmax的优点



P(correct) = 0.05:
3.0 penalty

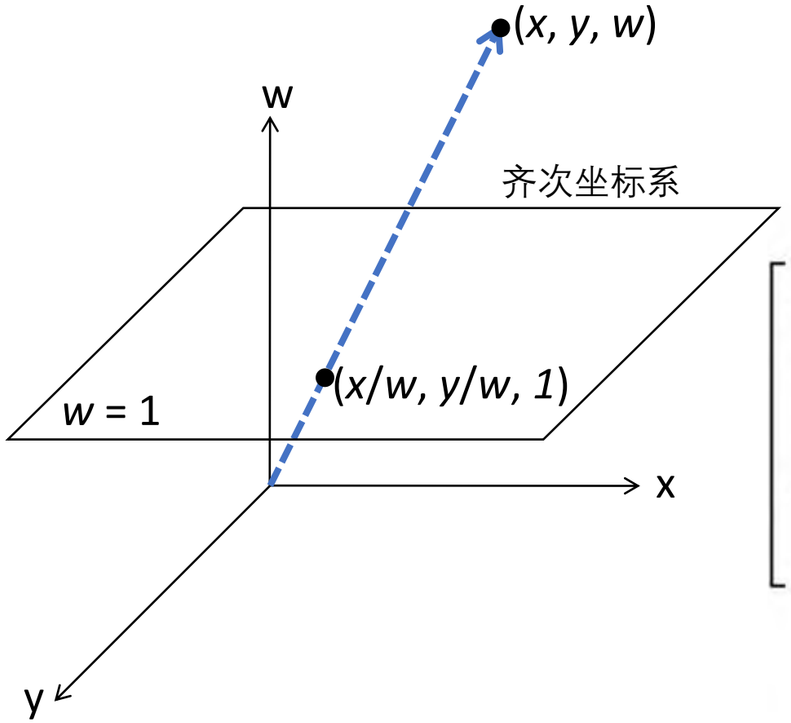
P(correct) = 0.5:
0.11 penalty

P(correct) = 0.9:
0.11 penalty

P(correct) = 1:
No penalty!

回顾——变换回归、图像分类

求解同构映射



$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$\mathbf{A}$ (2n × 9) $\mathbf{h}$ (9) $\mathbf{0}$ (2n)

分类最简单的形式：最近邻

训练图像
和对应标签

测试图像

$\boldsymbol{x}_1$ ← $D(\boldsymbol{x}_1, \boldsymbol{x}_T)$ → $\boldsymbol{x}_T$

$D(\boldsymbol{x}_N, \boldsymbol{x}_T)$

...

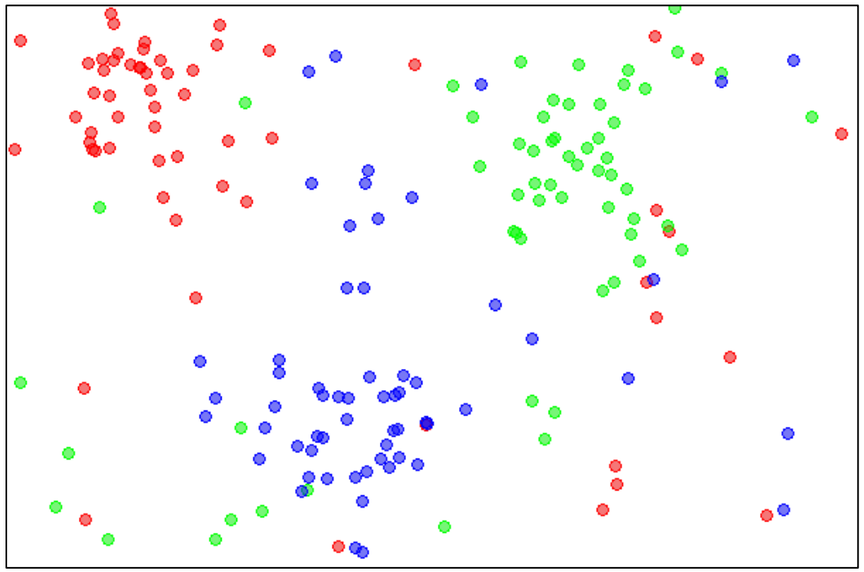
$\boldsymbol{x}_N$

(1) 计算特征向量的距离（特征匹配）(2) 找到训练集里最相似的样本 (3) 使用最相似样本的标签.

K近邻

找到前K近邻样本，然后投票决定标签

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

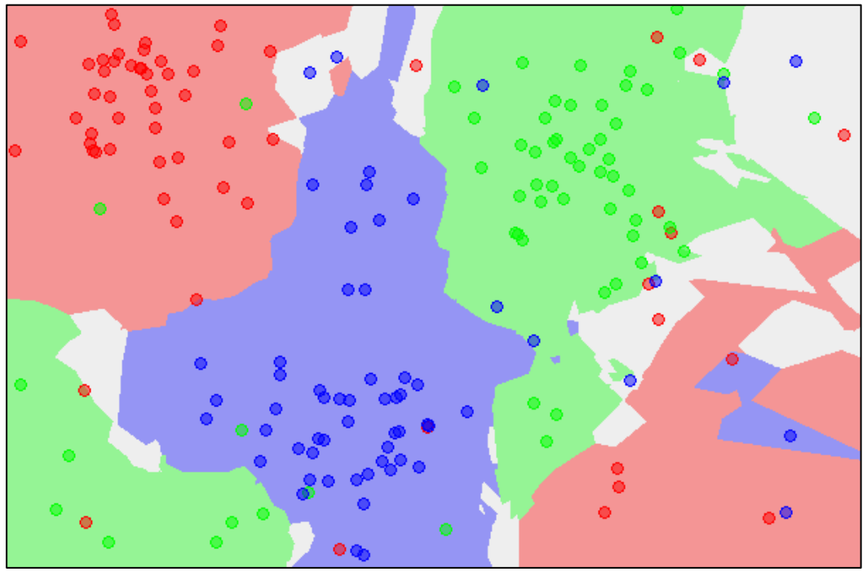


Diagram Credit: Wikipedia

K近邻

- 虽然没有学习过程，但通常是有效的
  - 不学习参数和结构，没有实际的训练过程
- 对于每个任务都使用相同的算法
- 当数据点数量趋近于无穷时，错误率保证最多比数据上的最优解差2倍
  - **局部决策**：K-NN在决策时只考虑局部信息，即输入点的K个最近邻。因为它是基于局部信息做出决策的，所以当数据量很大时，它能够捕捉到数据的微小细节和模式。
  - **大数定律**：当训练数据量增加时，每个点的近邻都更加可能代表真实的数据分布。这意味着K-NN的决策边界变得更加准确。
  - 当K同时增长并且K/N趋近于0，其中N是数据点的数量
  - K-NN的错误率会收敛到贝叶斯最优错误率
  - **看似很好，但是有效率问题，且性能一般**

线性模型

设定：三分类模型



模型：每个类别一个权重

$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

全部参数: $\boldsymbol{W_{3xF}}$ where **x** is in R^F

可视化线性分类器

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

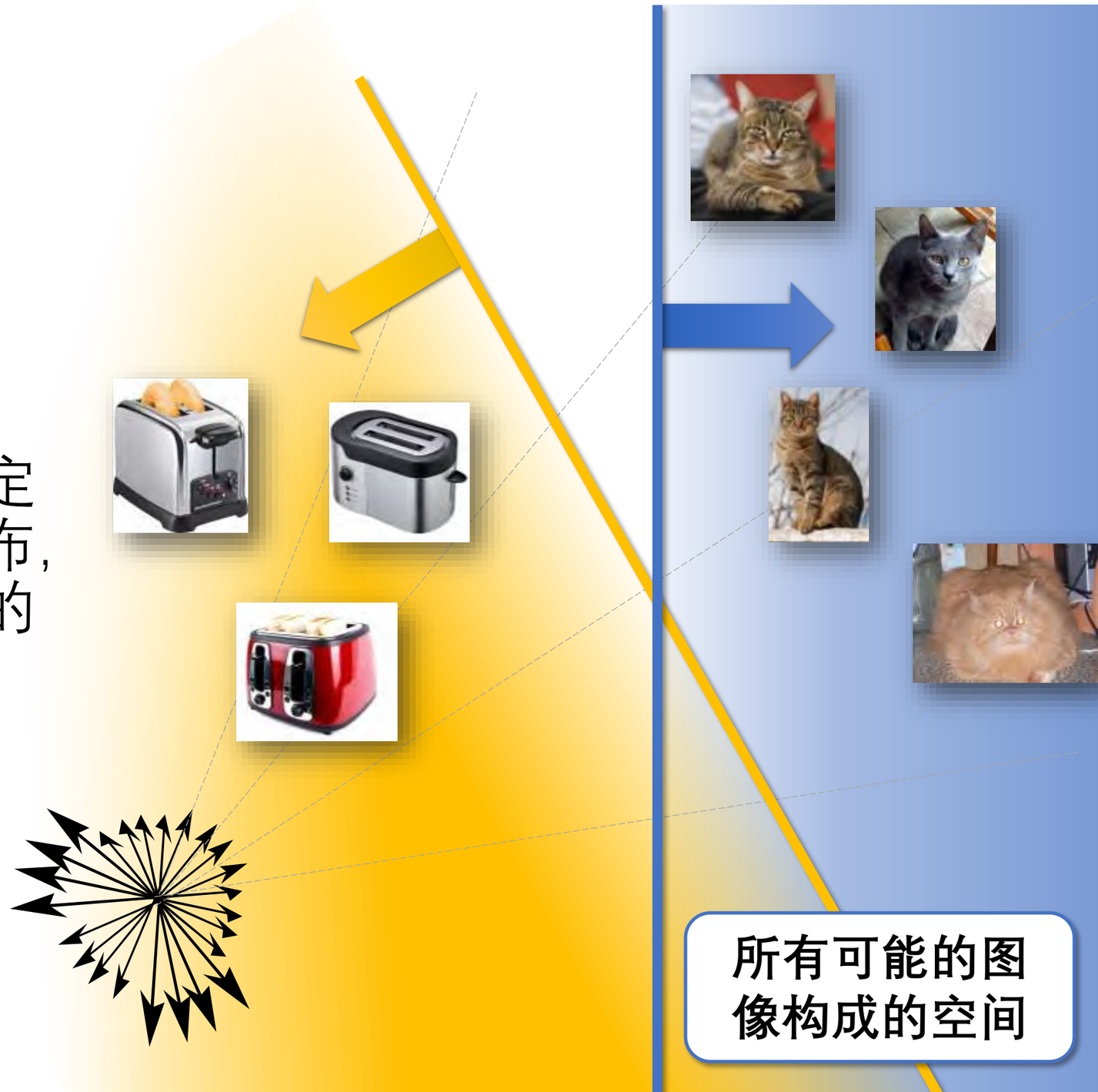
Diagram credit: Karpathy & Fei-Fei

解释：几何角度

- 参数为每个类定义一个超平面:

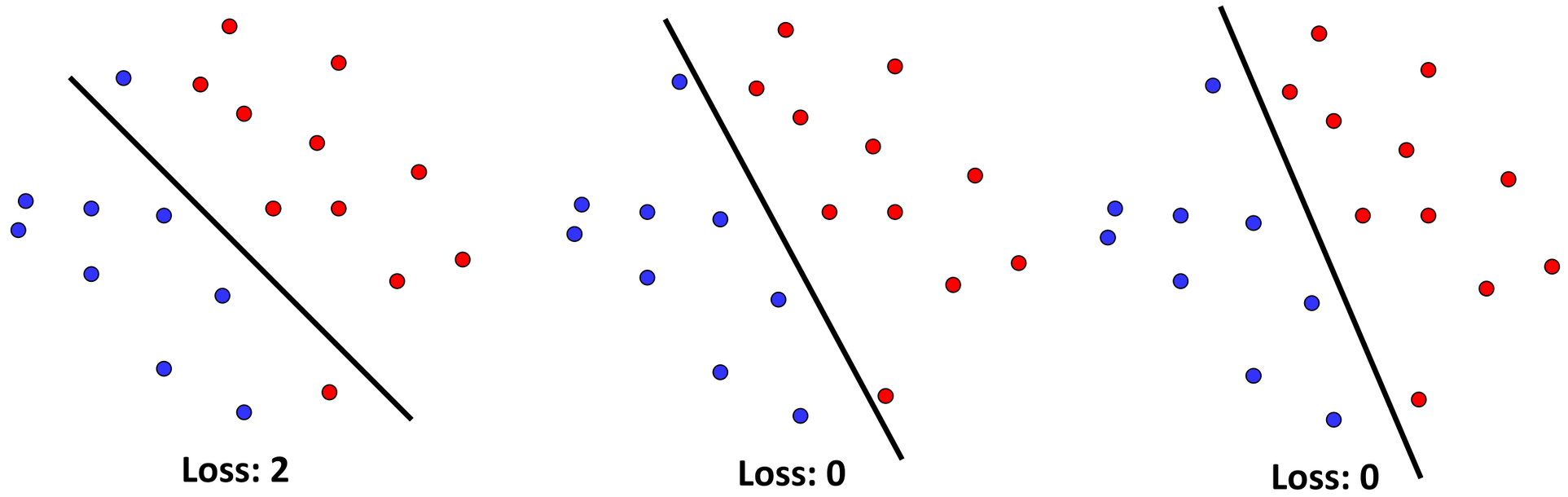
$$f(x_i, W, b) = Wx_i + b$$

- 我们可以将每个类的得分视为定义了一个与其距离成正比的分布，距离是指从对应的超平面到点的距离。



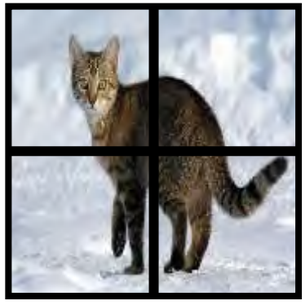
什么是好的损失函数?

- 检视分类错的样本
  - 问题：样本是离散的，我们没办法区分相似的分类器
  - 我们需要更好的 **泛化性** *generalization*
  - 还要处理多于两类的情况



如何设计损失函数 —— 最大间距

这样有什么问题?



Loss: dog score – cat score

怎么定义“dog” vs “cat”的评分?

$$(\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1$$

避免优化到负无穷

$$\max(0, (\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1)$$

$(\boldsymbol{Wx})_1$	-96.8	Cat score
$(\boldsymbol{Wx})_2$	437.9	Dog score
$(\boldsymbol{Wx})_3$	61.95	Hippo score

$$\boldsymbol{Wx}$$

预测结果是一个向量,
第j个值代表j类别的评分

Diagram by: Karpathy, Fei-Fei

Softmax

推断时 (x): $\arg\max_k (\boldsymbol{W}\boldsymbol{x})_k$ (找到评分最高的那一类)

训练时 (x_i,y_i):

$$\arg\min_{\boldsymbol{W}} \boldsymbol{\lambda}\|\boldsymbol{W}\|_2^2 + \sum_i^n -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)$$

P(correct class)

Regularization
正则项

对所有训练样本

对正确分类对应的 negative log-likelihood 进行损失优化

Cross-entropy loss 交叉熵

$$f(x_i, W) = Wx_i \quad \text{(score function)}$$

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \qquad L_i = -f_{y_i} + \log\sum_j e^{f_j}$$

我们称 L_i 为 交叉熵 *cross-entropy loss*

$P(y_i \mid x_i; W)$

i.e. we're minimizing the negative log likelihood.

损失函数

- 交叉熵损失仅仅是一种可能的损失函数。
  - 它的一个好属性是它将得分重新解释为概率，这些概率具有自然的含义。

- 支持向量机（SVM，最大间距）损失函数过去也很受欢迎。
  - 但目前，交叉熵是最常见的分类损失。

怎么优化线性模型?

目标：找到参数**w** 最小化损失函数 L.

$$\arg\min_{\boldsymbol{w}\in R^N} L(\boldsymbol{w})$$

$$L(\boldsymbol{W}) = \boldsymbol{\lambda}\|\boldsymbol{W}\|_{\boldsymbol{2}}^{\boldsymbol{2}} + \sum_{i=1}^{n} -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k))}\right)$$

不同的损失函数L:

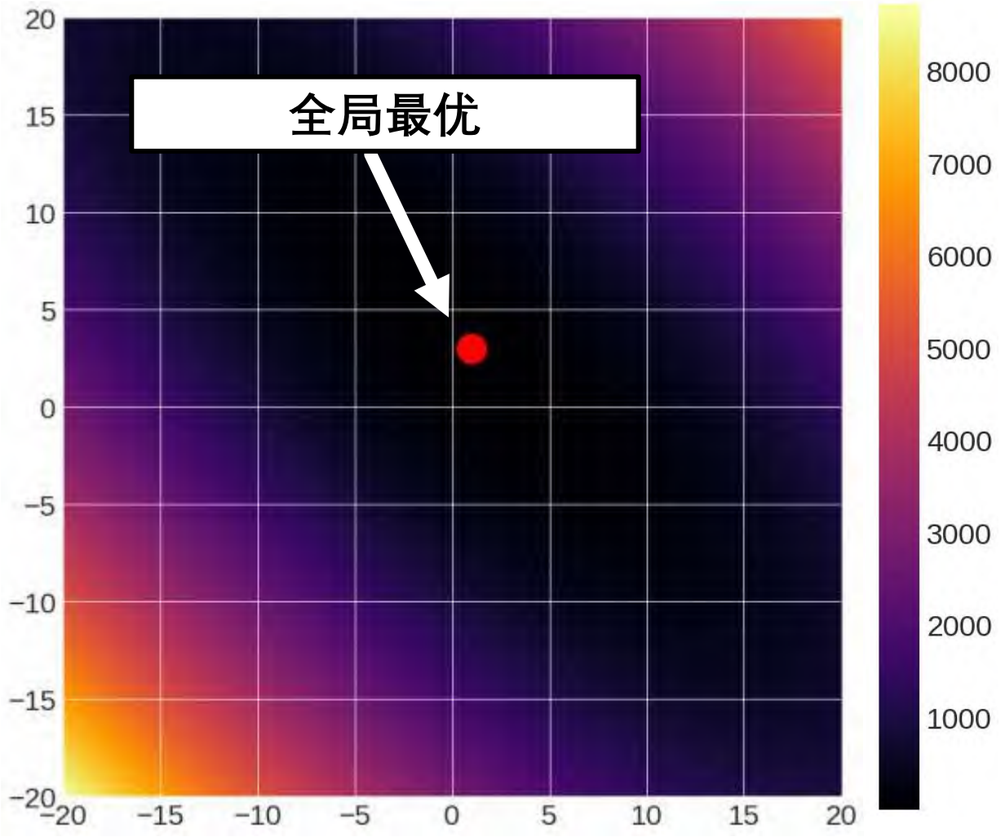
$$L(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n}\left(y_i - \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}}\right)^2$$

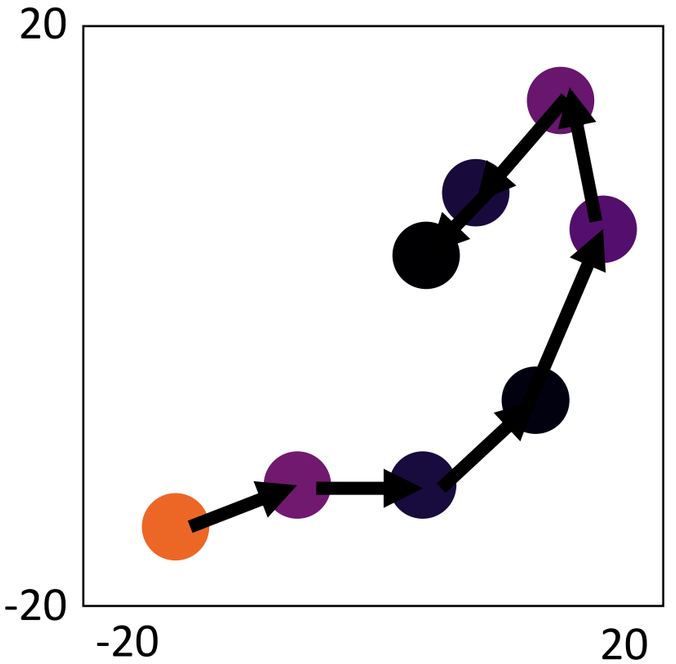
$$L(\boldsymbol{w}) = C\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n}\max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_{\boldsymbol{i}})$$

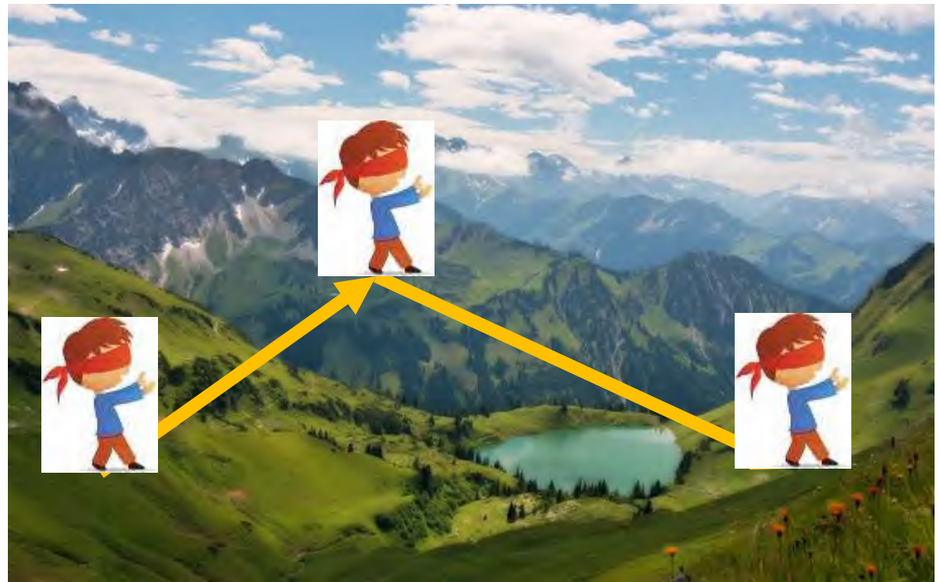
怎么优化?

- 每个点代表函数的值
  （大小参照右边热度图）
- 类似“爬山”与“下山”

$$f(x,y) = (x+2y-7)^2 + (2x+y-5)^2$$

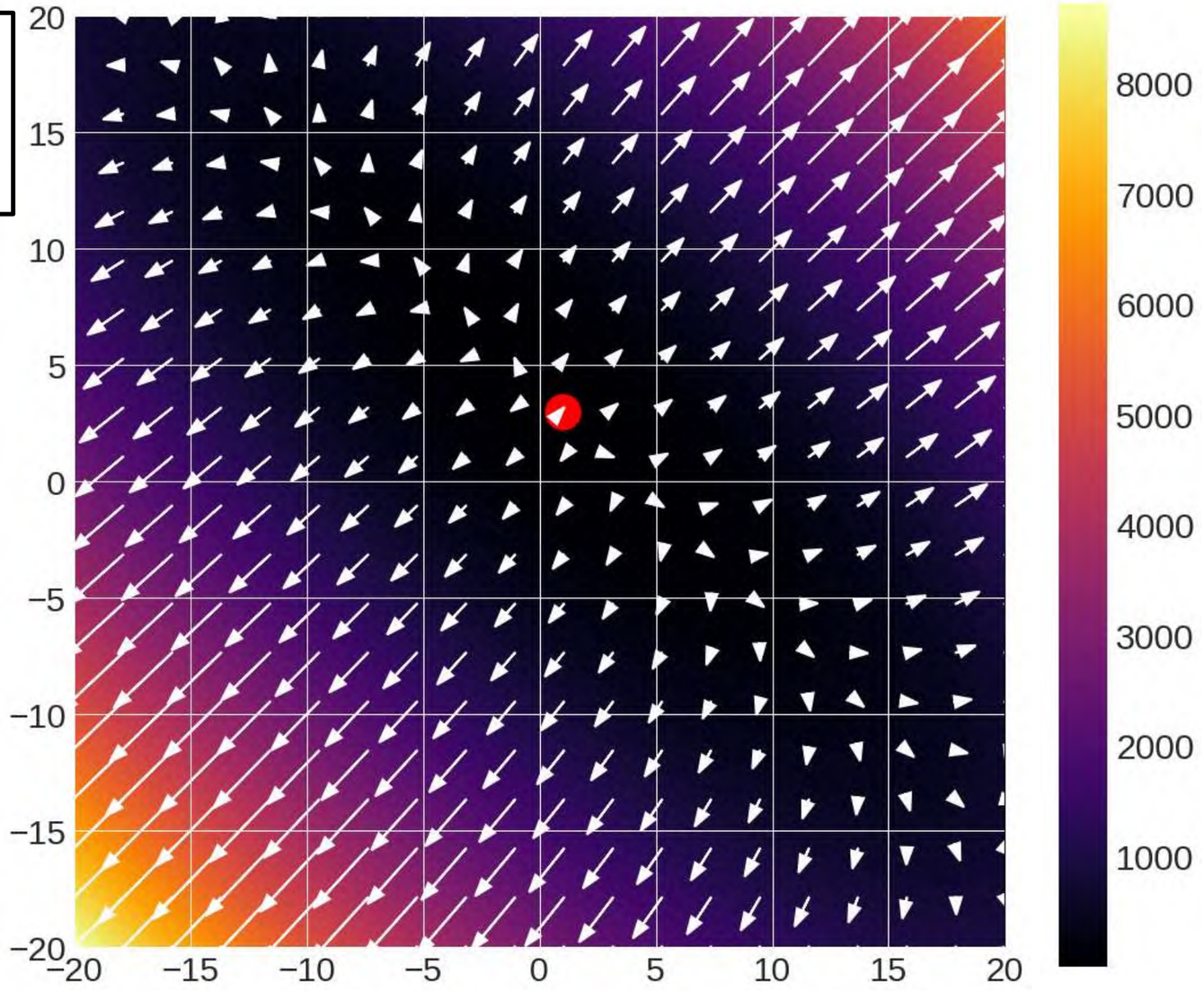






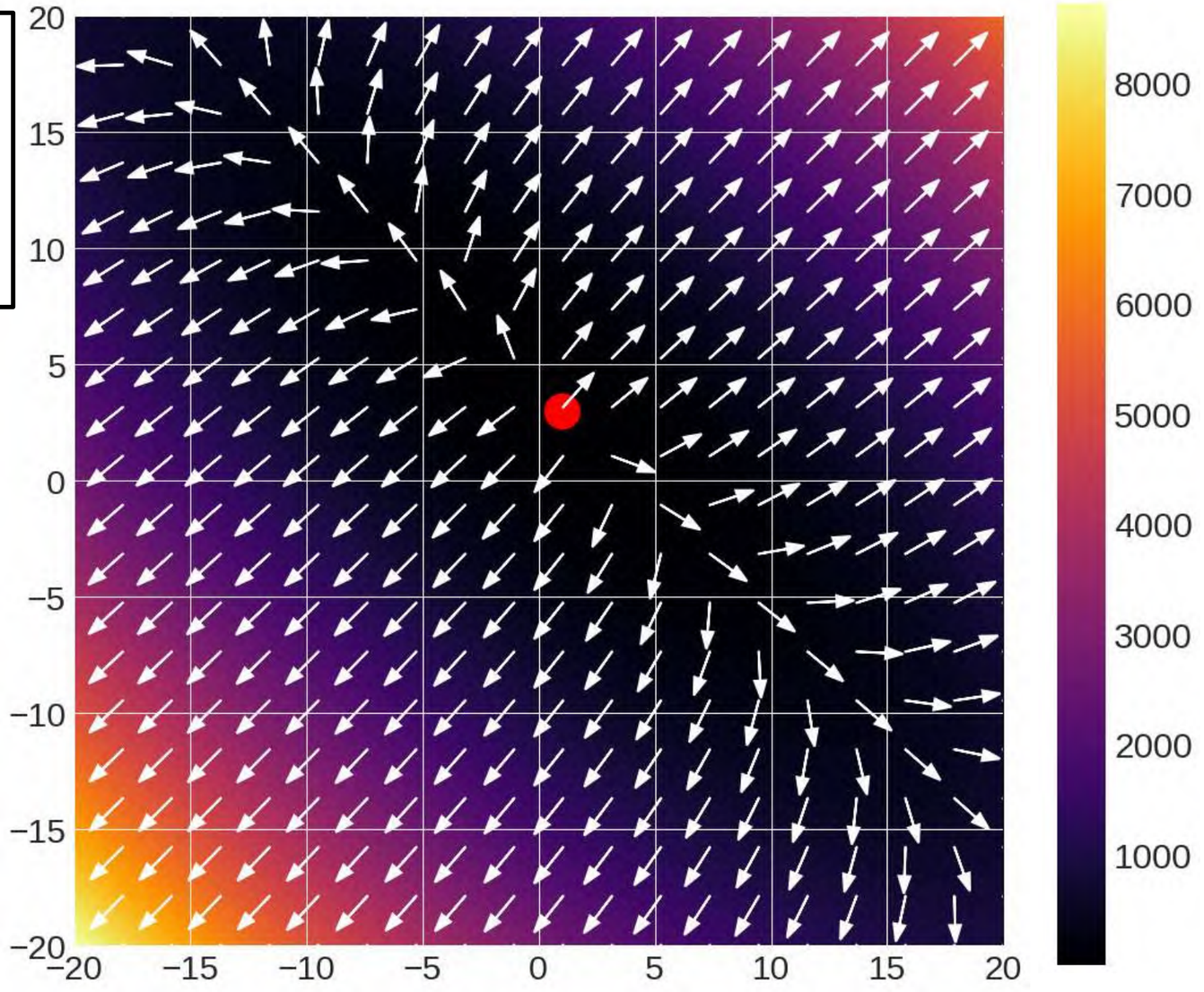
怎么优化？使用梯度

箭头：
梯度 **gradient**



怎么优化？使用梯度

箭头:
梯度方向
(单位长度)



怎么优化？使用梯度

目标: $\arg\min_{\boldsymbol{w}} L(\boldsymbol{w})$

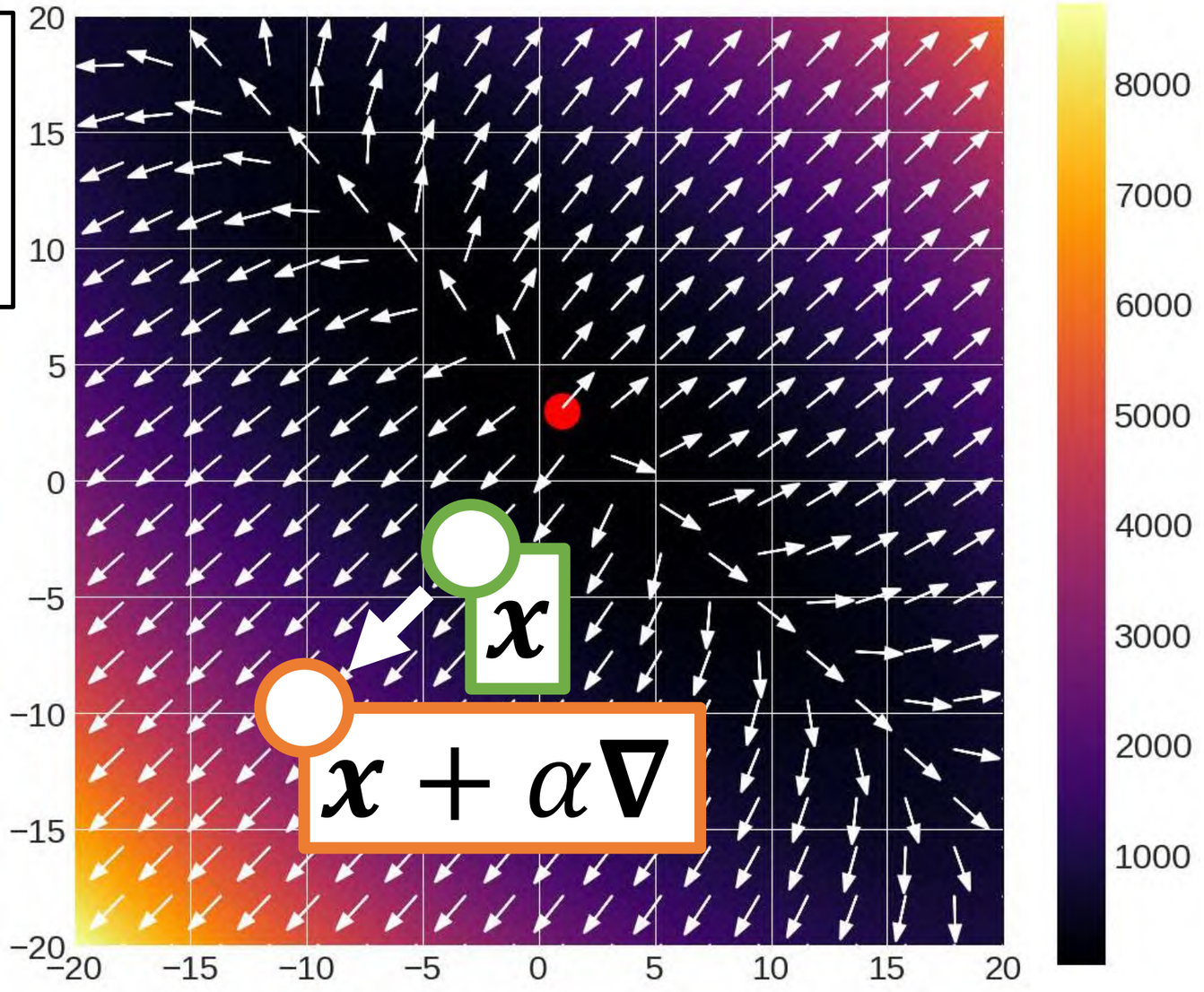
计算梯度: $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \begin{bmatrix} \partial L/\partial \boldsymbol{x}_1 \\ \vdots \\ \partial L/\partial \boldsymbol{x}_N \end{bmatrix}$

哪个更大(当α较小时)?

$$L(\boldsymbol{w}) \quad \begin{matrix} \leq ? \\ > ? \end{matrix} \quad L(\boldsymbol{w} + \alpha \nabla_{\boldsymbol{w}} L(\boldsymbol{w}))$$

怎么优化？使用梯度

箭头:
梯度方向
(单位长度)



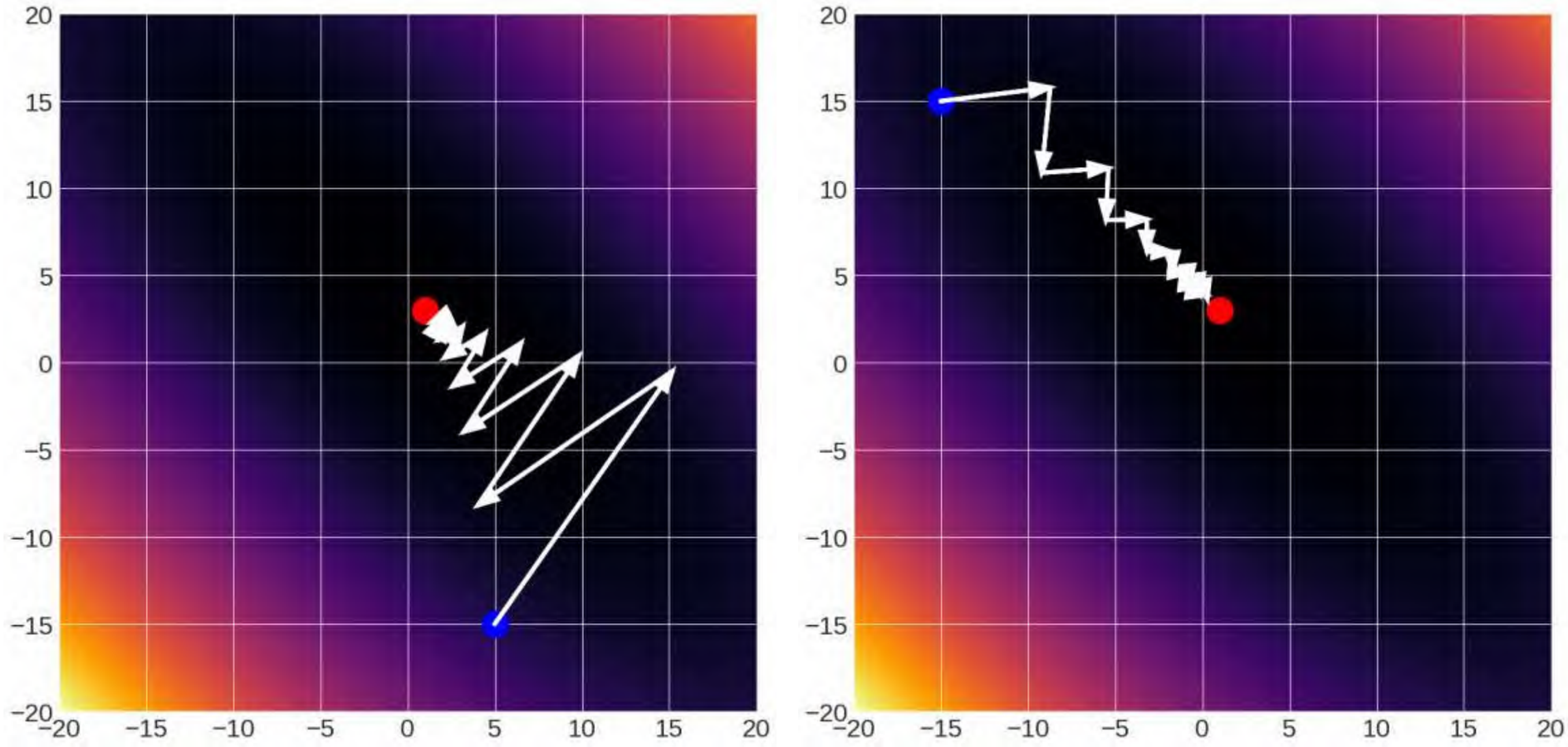
梯度下降

每一步优化，向梯度的反方向移动

w0 = *initialize***()** #initialize
for iter in range(numIters):
    g = $\nabla_{\boldsymbol{w}} L(\boldsymbol{w})$ #eval gradient
    w = **w** + -*stepsize*(iter)***g** #update w
return **w**

梯度下降

基于蓝色起始点，$w_{i+1} = w_i + -9.8x10^{-2} \times gradient$

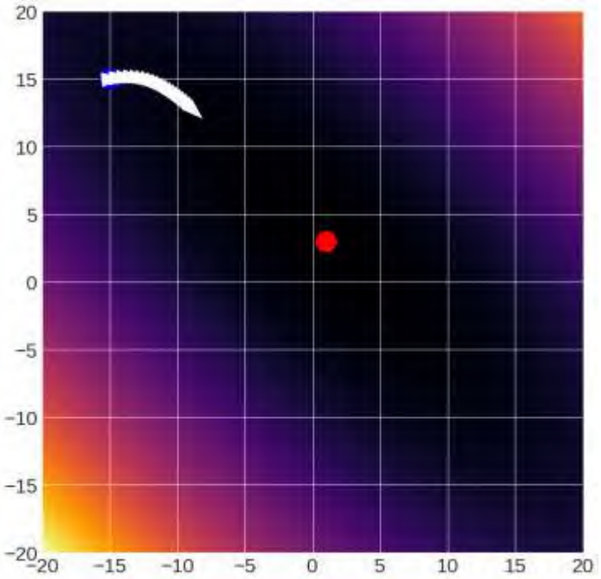


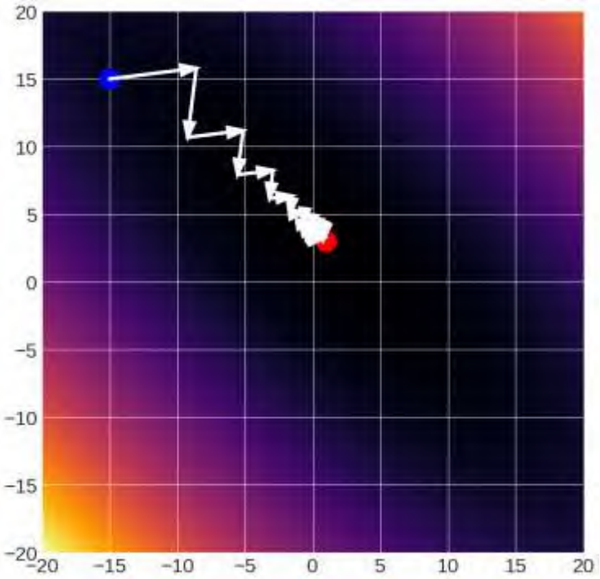
梯度下降

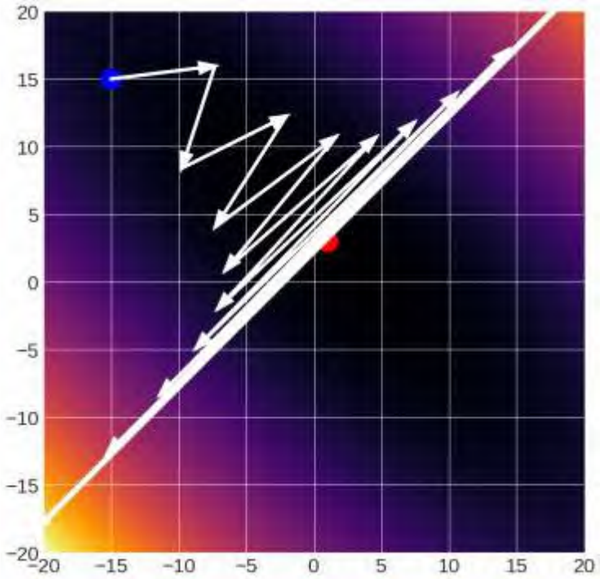
步长 Step size (也叫做 学习率**learning rate / lr**)

十分重要的参数

$1x10^{-2}$	$10x10^{-2}$	$12x10^{-2}$
falls short	converges	diverges

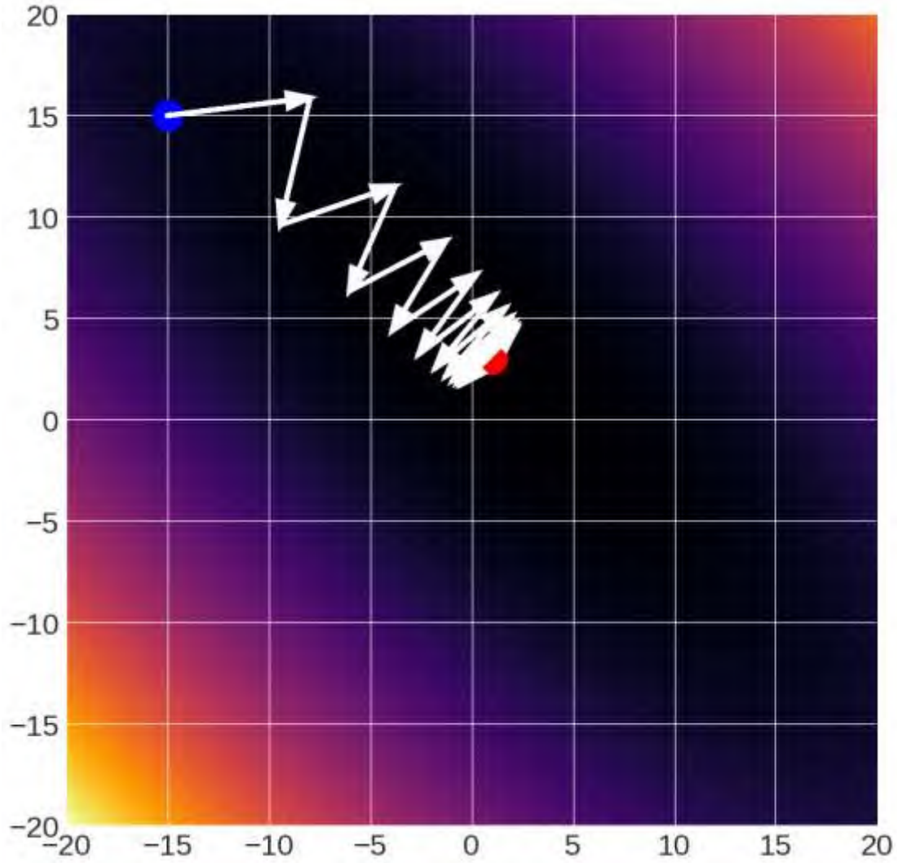






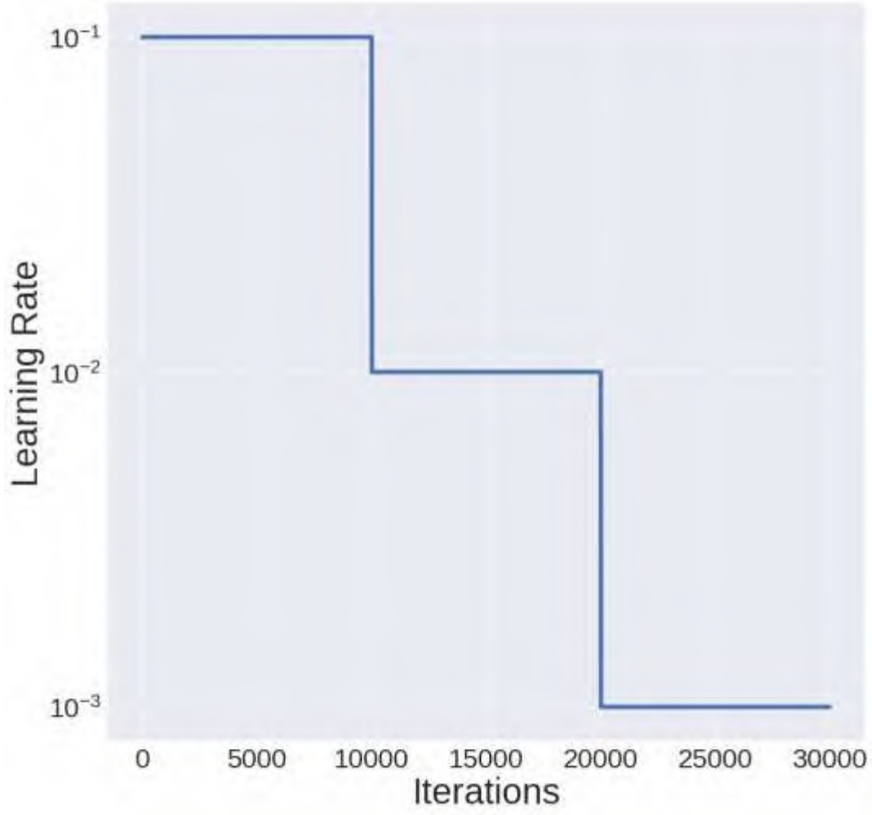
梯度下降

$11x10^{-2}$:oscillates
(Raw gradients)



梯度下降

使用初始的 lr, 每N步以后乘以f



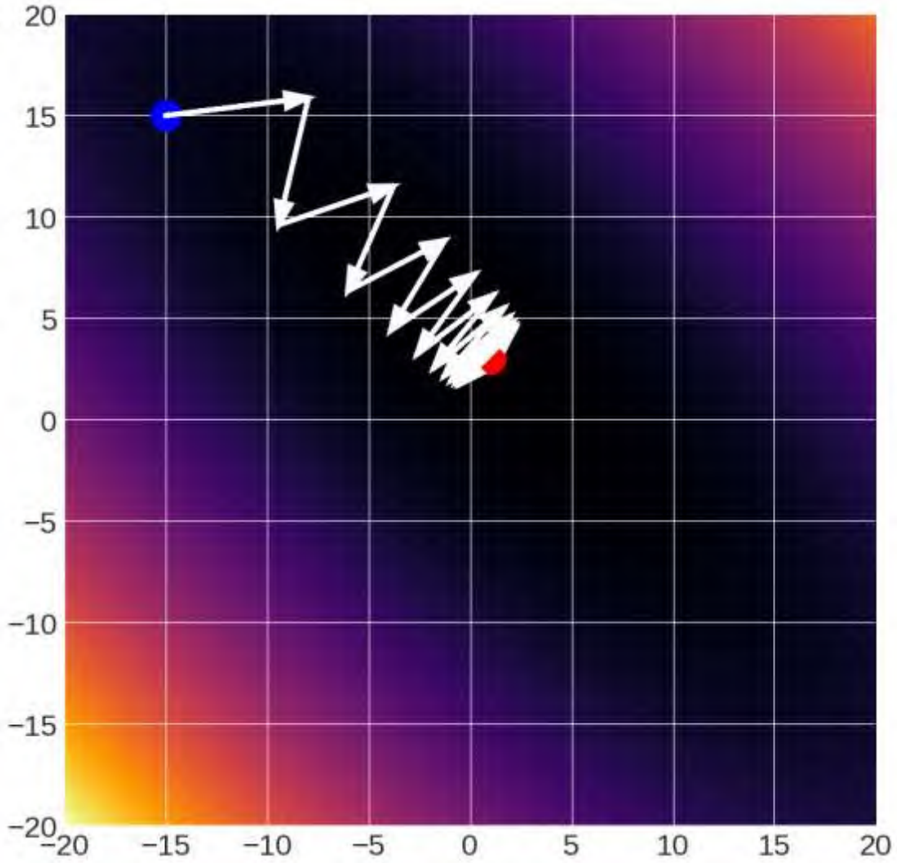
init_lr = 10^{-1}

f = 0.1

N = 10K

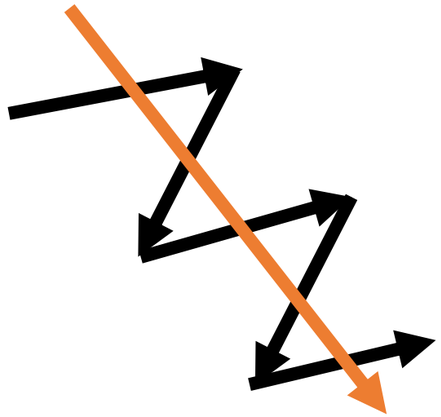
梯度下降

$11x10^{-2}$:oscillates
(Raw gradients)



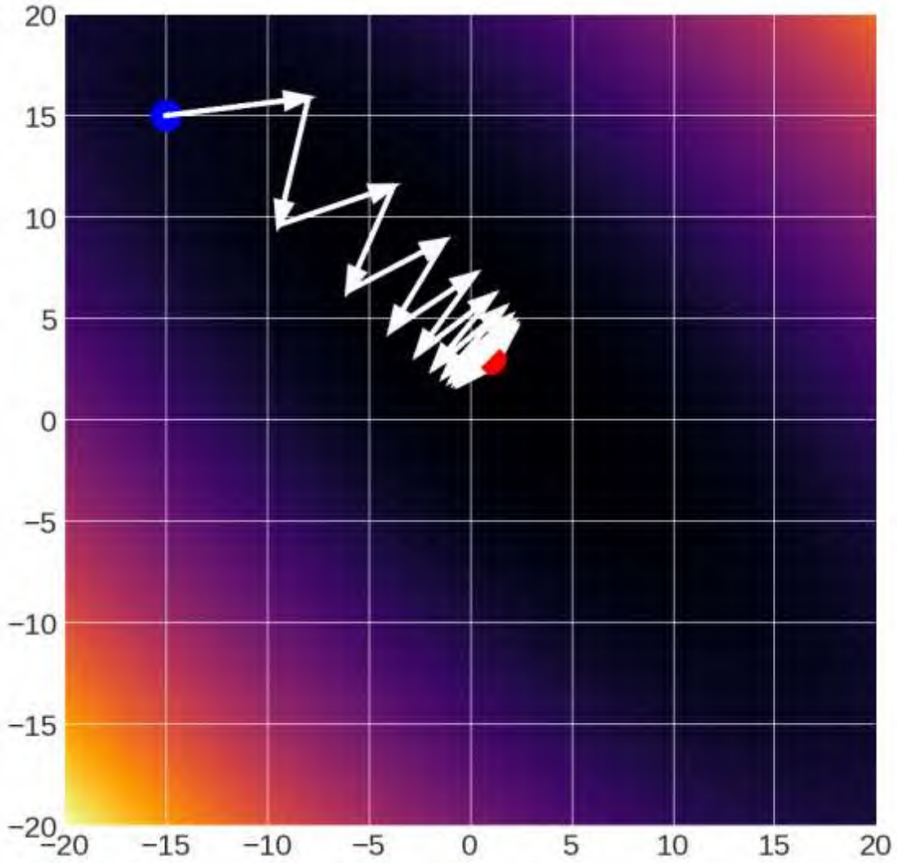
“平均梯度”

记录历史梯度，并且与当前步梯度取加权平均， 也被称为动量法 “momentum”

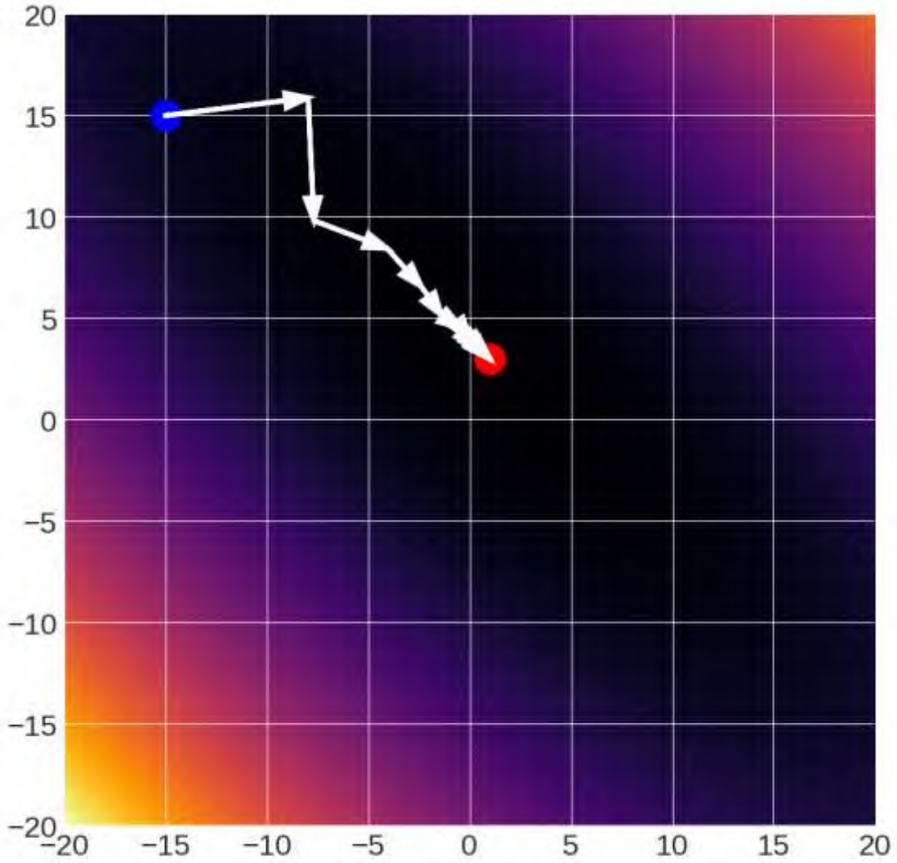


梯度下降

$11x10^{-2}$:oscillates
(Raw gradients)

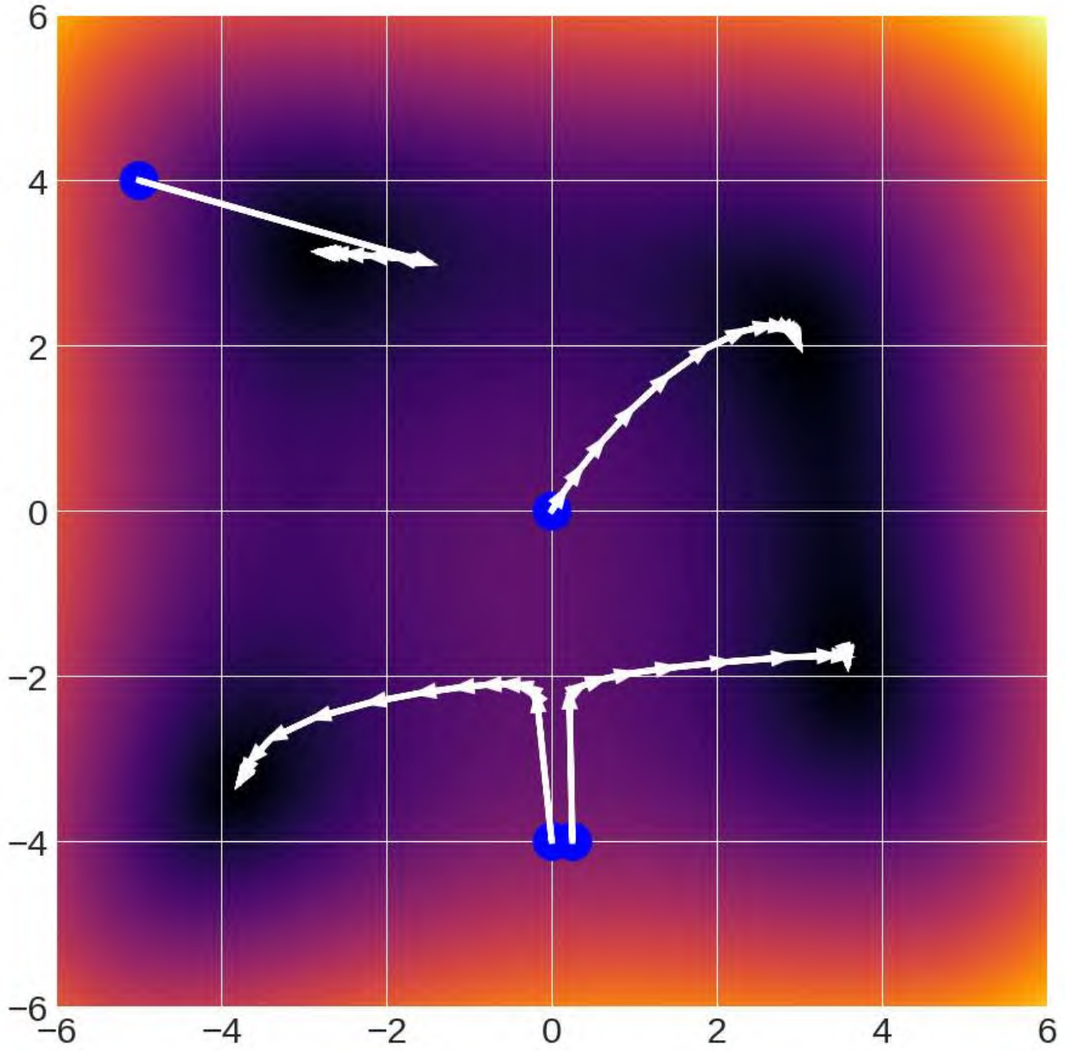


$11x10^{-2}$
(0.25 momentum)



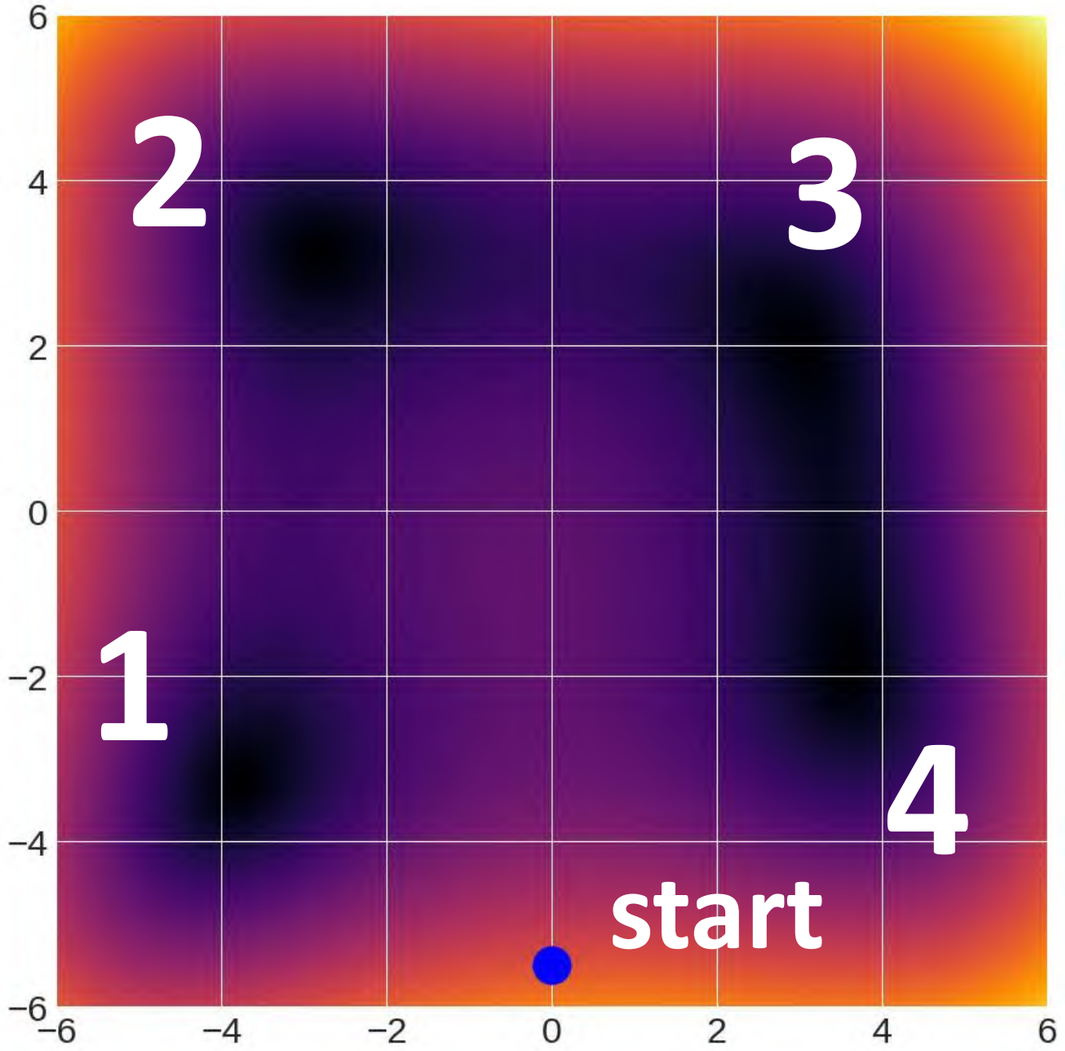
梯度下降

多个极小值时
→
梯度下降会找到
局部极小



梯度下降

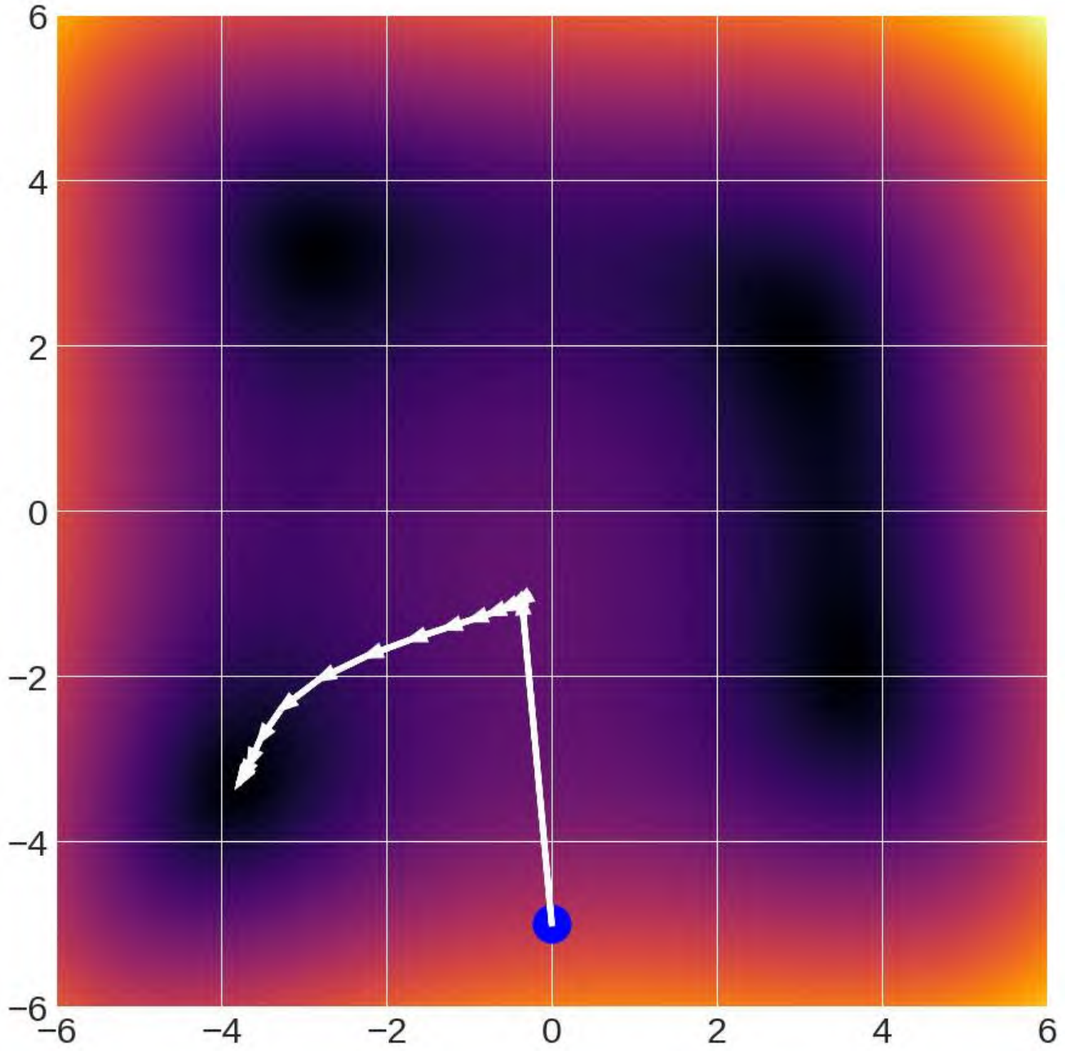
如果我们从蓝色点开始，会走向哪个局部最小？



梯度下降

许多函数是凸的
convex: 局部最小
就是全局最小

但是大多数函数并
不是



经验

- **一般来说**:小批量随机梯度下降（SGD）+ 动量+ 逐步调整的学习率
- 存在其他的更新规则（例如，AdamW）。在某些问题上，它们经常表现得更好。

损失函数

- 交叉熵损失仅仅是一种可能的损失函数。
  - 它的一个好属性是它将得分重新解释为概率，这些概率具有自然的含义。

- 支持向量机（SVM，最大间距）损失函数过去也很受欢迎。
  - 但目前，交叉熵是最常见的分类损失。

- 交叉熵可能更容易过拟合，特别是当模型在训练数据上获得非常高的分类准确性时。
- 最大间距可能更加健壮，因为它关注的是那些难以正确分类的样本，并试图确保它们被正确分类，而不是关注所有样本。

过拟合、欠拟合 与 模型复杂度

多项式回归: 给定 x, 预测 y

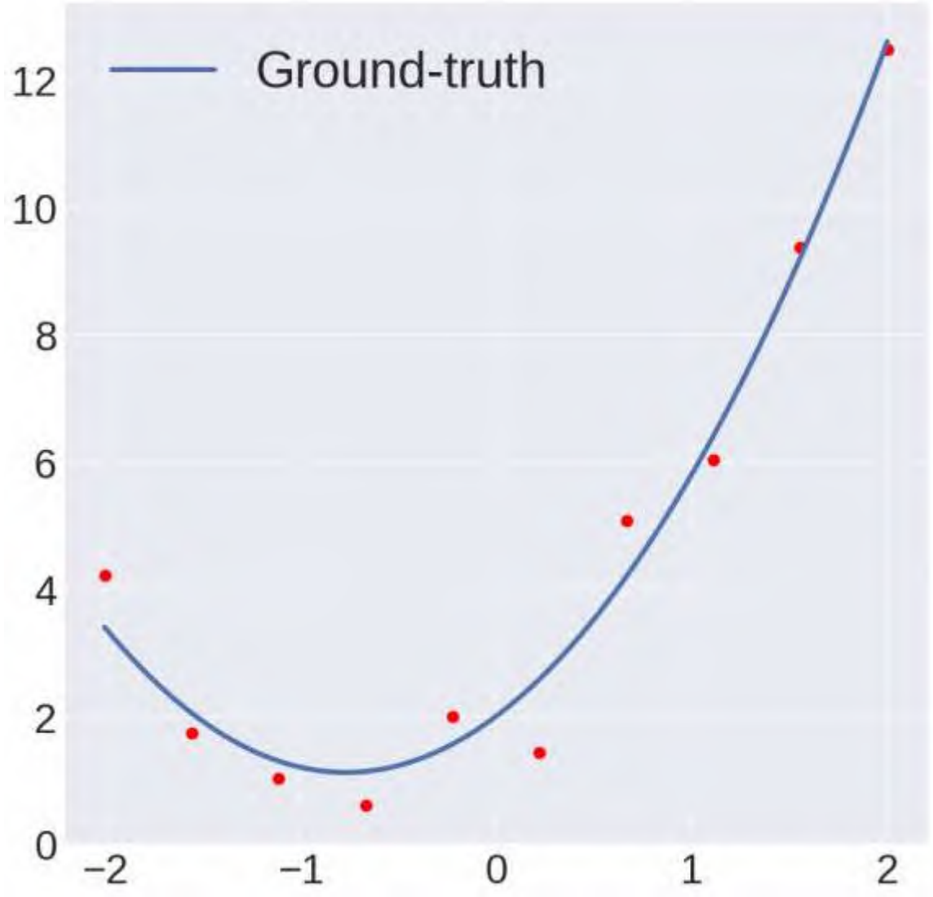
$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1^F & \cdots & x_1^2 & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_N^F & \cdots & x_N^2 & x_N & 1 \end{bmatrix} \begin{bmatrix} w_F \\ \vdots \\ w_2 \\ w_1 \\ w_0 \end{bmatrix}$$

我们有一个输入矩阵 X，其中包含了各种多项式次数的数据（例如，x, x^2, x^3 等）。这样，模型可以选择最佳的多项式次数来拟合数据。

权重W: 每个多项式度数都有一个相应的权重,这些权重决定了每个多项式度数在模型中的重要性。

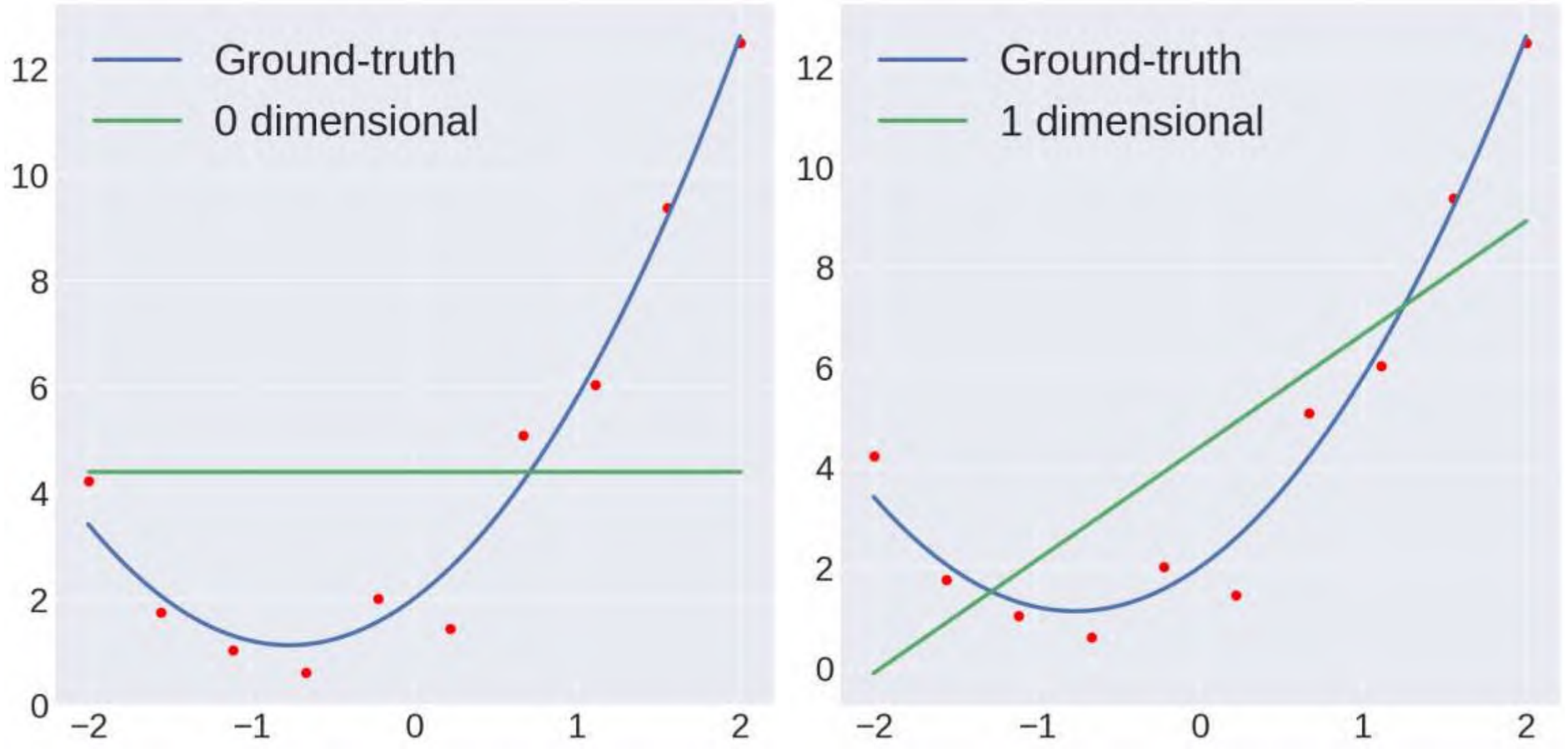
过拟合、欠拟合 与 模型复杂度

Model: $1.5x^2 + 2.3x+2 + N(0,0.5)$

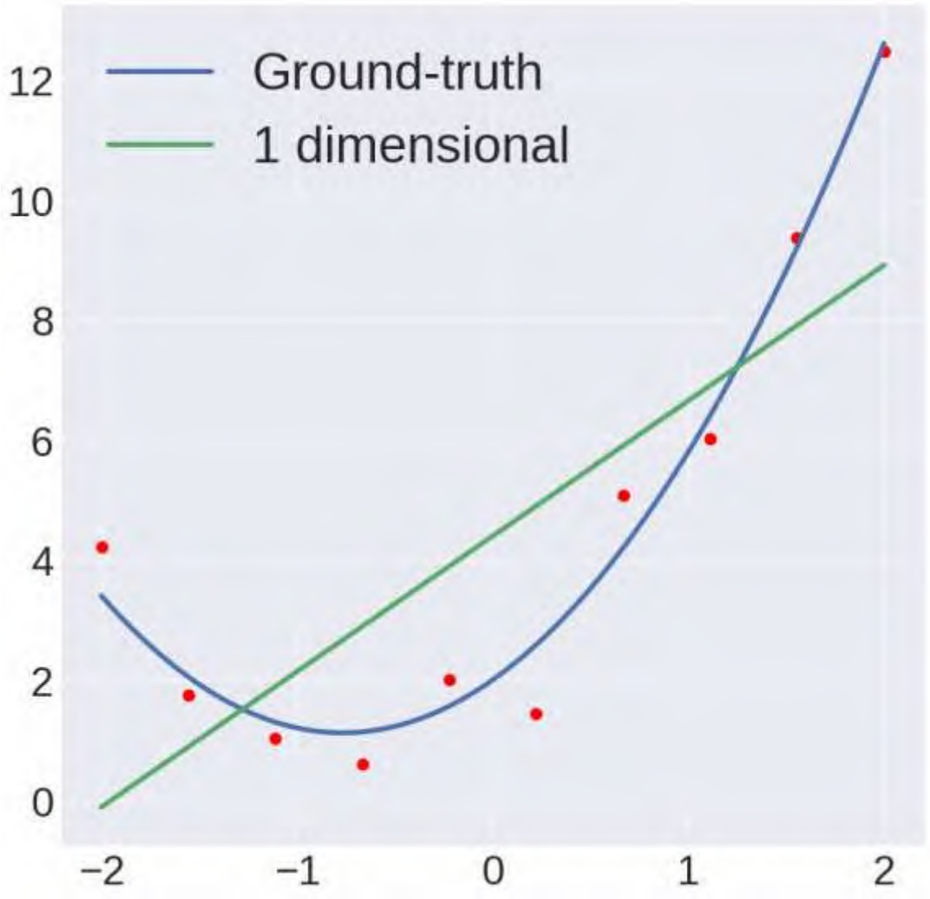


欠拟合

数据: $1.5x^2 + 2.3x+2 + N(0,0.5)$



欠拟合

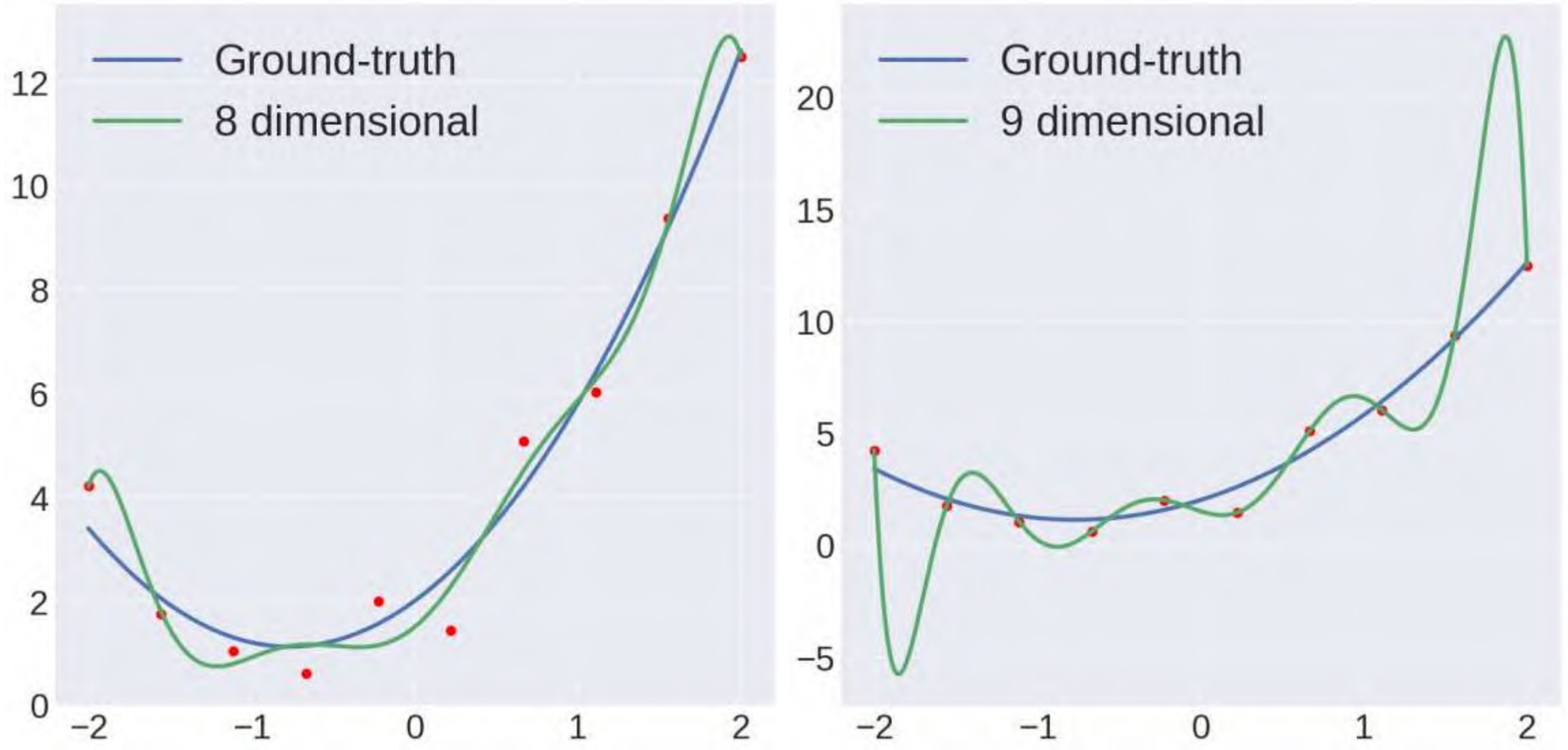


模型没有足够的参数或复杂度去拟合数据。即使在训练数据上，模型的表现也可能不尽如人意。

Bias 偏差 (statistics): 模型固有的错误

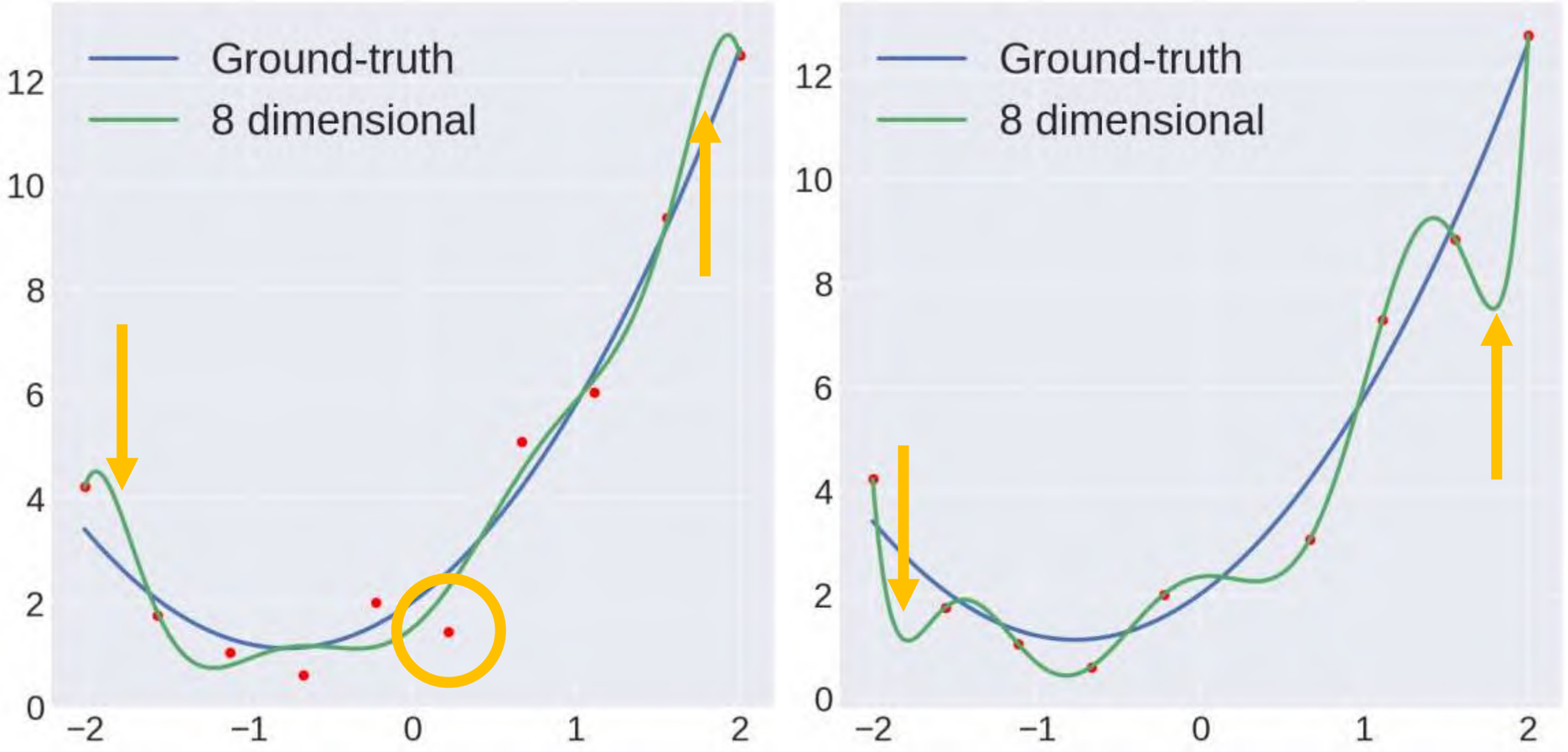
过拟合

数据: $1.5x^2 + 2.3x+2 + N(0,0.5)$



过拟合

高方差 variance:过高的方差意味着模型对训练数据的微小变化非常敏感 移除 **一个数据点**, 模型就会出现很大变化



模型复杂度

$$\arg\min_{W} \lambda \|W\|_2^2 + \sum_{i=1}^{n} \underbrace{-\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)}_{\text{交叉熵损失}}$$

Regularization 正则化:
对模型的权重添加惩罚

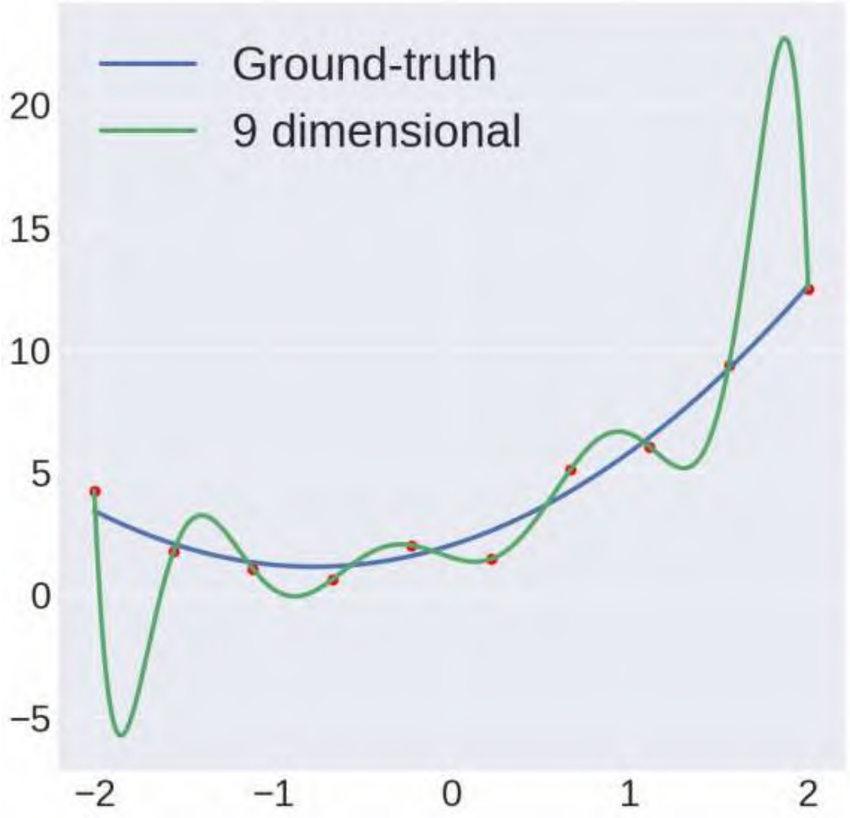
模型复杂性是指模型的容量或能力，它决定了模型可以适应多复杂的数据分布。
复杂的模型可能在训练数据上拟合得很好，但在新数据上可能表现得不好，
这被称为过拟合。

Model 1: $0.01*x_1 + 1.3*x_2 + -0.02*x_3 + -2.1x_4 + 10$

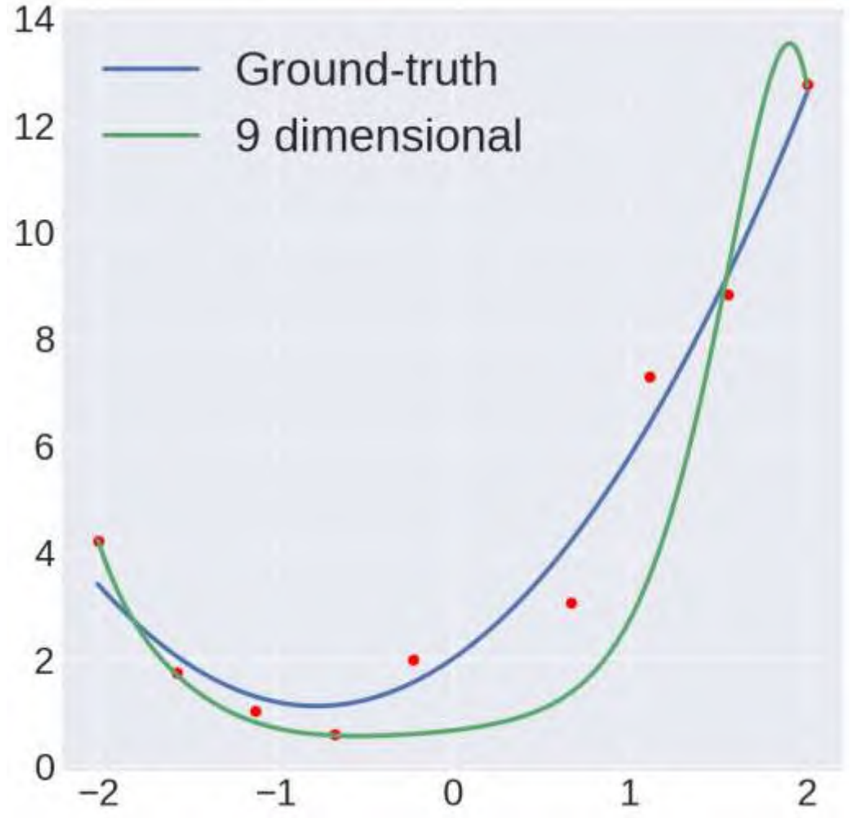
Model 2: $37.2*x_1 + 13.4*x_2 + 5.6*x_3 + -6.1x_4 + 30$

正则化

没有正则化:
（过）拟合所有点



使用正则化:
无法拟合所有点



总的来说

当模型在新数据上表现不佳时，通常是由三种误差组合而成的：偏差、方差和固有误差。

1. Bias: 偏差描述的是模型的简化性，它是模型假设与真实数据之间的差异。一个高偏差的模型可能太简单，无法捕捉到数据的真实模式。
2. Variance: 方差描述的是模型对训练数据的敏感性。高方差模型在不同的训练数据集上可能会有很大的差异。这通常是因为模型过于复杂，试图捕捉训练数据中的每一个小细节，甚至包括噪声。
3. Inherent: 固有误差是与数据本身相关的误差，与模型无关。例如，数据可能包含噪声，或者某些特征无法被测量。

通常，减少偏差可能会增加方差，反之亦然。选择合适的模型复杂度是关键。

欠拟合与过拟合

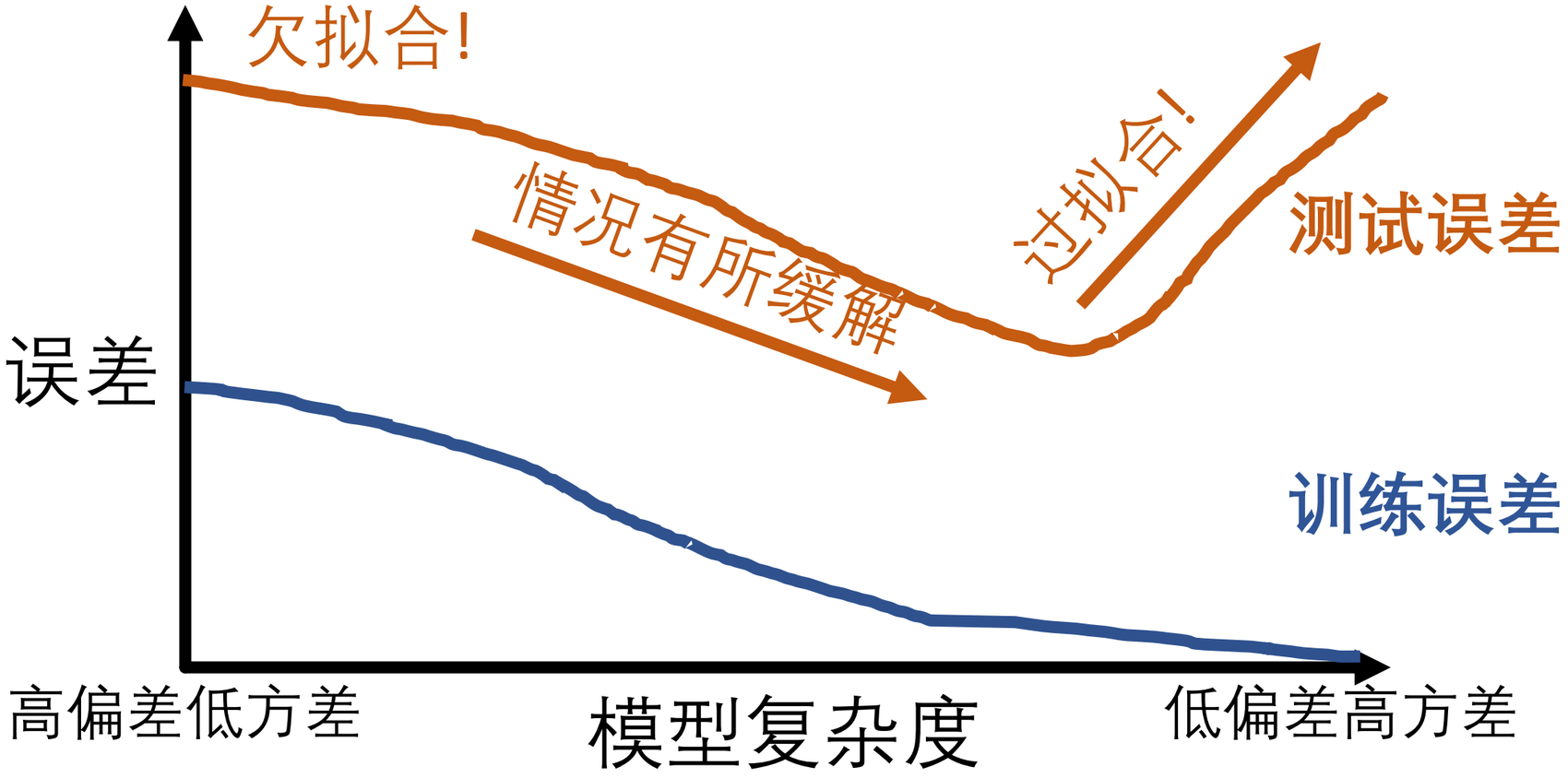


Diagram adapted from: D. Hoiem

欠拟合与过拟合

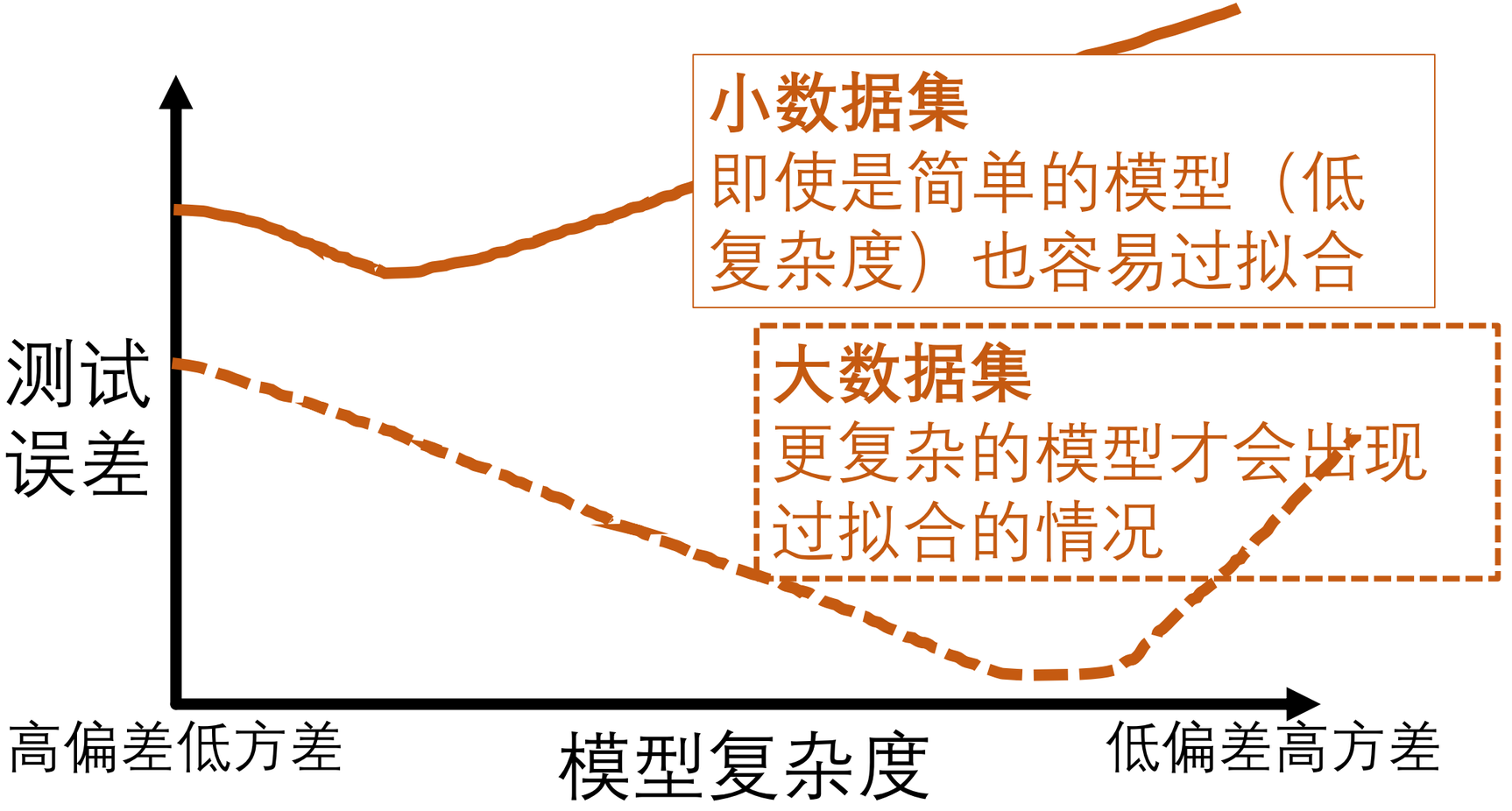
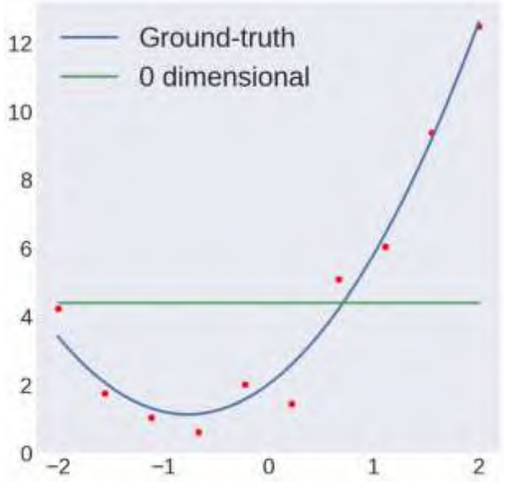
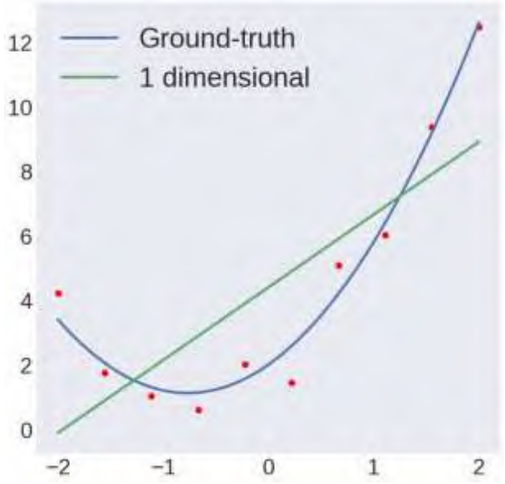


Diagram adapted from: D. Hoiem

欠拟合



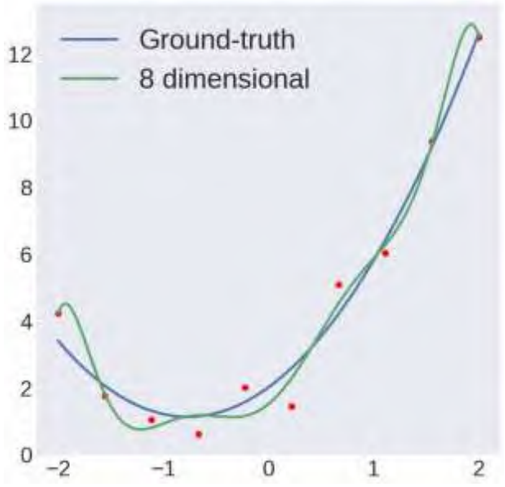


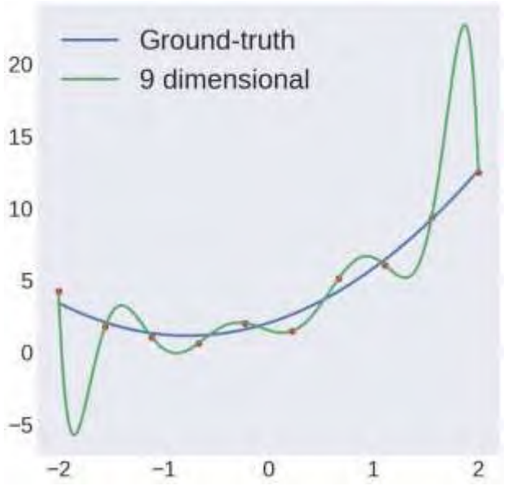
当模型的复杂度过低时，即使在训练数据上也不能得到满意的拟合效果。这通常意味着模型太简单，不能捕获数据的真实模式。

解决方案:

- **增加更多特征**：更多的特征可以增加模型的复杂度，使其更容易捕获到数据中的模式。
- **使用更强大的模型**：如从线性模型升级到多项式模型或使用深度学习模型。
- **减少正则化**：正则化是为了避免过拟合而添加的惩罚项，但如果模型欠拟合，则可以减少或移除正则化。

过拟合





模型在训练数据上表现很好，但在验证数据上表现较差，通常是因为模型太复杂，过度拟合了训练数据中的噪声或异常值。

解决方案:

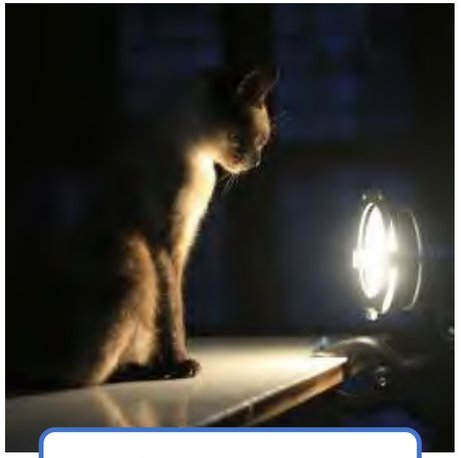
- **增加更多数据**：更多的训练数据可以帮助模型学习到更广泛的数据分布，从而减少过拟合。
- **使用简单模型**：选择一个复杂度较低的模型，避免过度拟合。
- **增加正则化**：正则化可以防止模型过于复杂，从而限制其拟合能力。

建议：首先确保模型有足够的能力进行拟合（即避免欠拟合），然后再考虑如何避免过拟合。

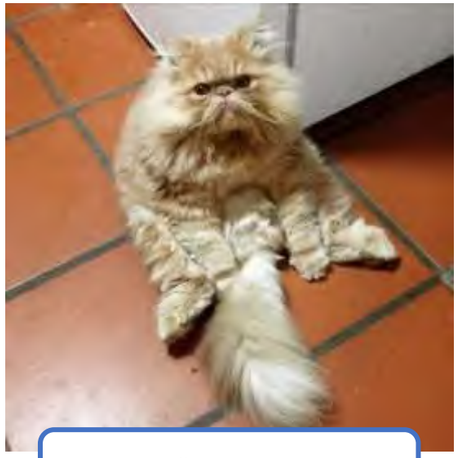
回顾：外观的巨大差异让识别问题困难倍增

- 相同类别的物体图像差异极大

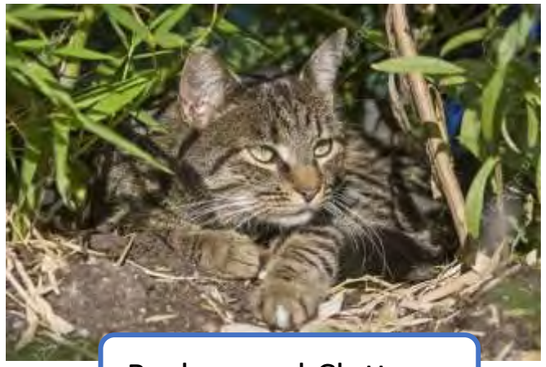
Viewpoint Variation



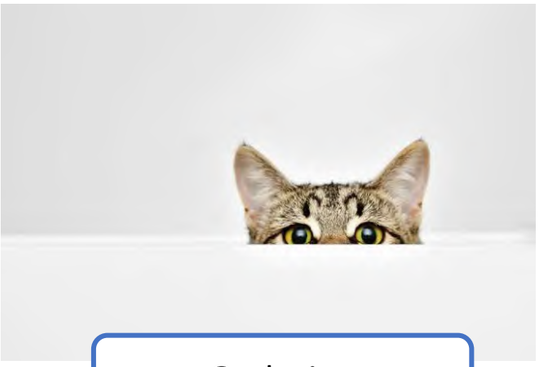
Lighting Variation



Deformation



Background Clutter



Occlusion

问题：欠约束 Under-constrained

- **不同的现实情境可以产生相同的图像**：这意味着从同一个图像出发，我们可能会得到多种不同的解释或解决方案。
- **通常我们不能计算“正确”的答案**：由于存在多种可能的现实情境，我们通常不能确定一个图像的“正确”解释。但我们可以计算最有可能的答案。
- **我们需要某种先验来加以条件**：由于问题是欠约束的，我们需要某种先验知识或假设来缩小可能的答案范围。
- **我们可以从数据中学习这个先验**：通过训练数据，我们可以学习到这些先验知识或假设，以更好地解决问题。

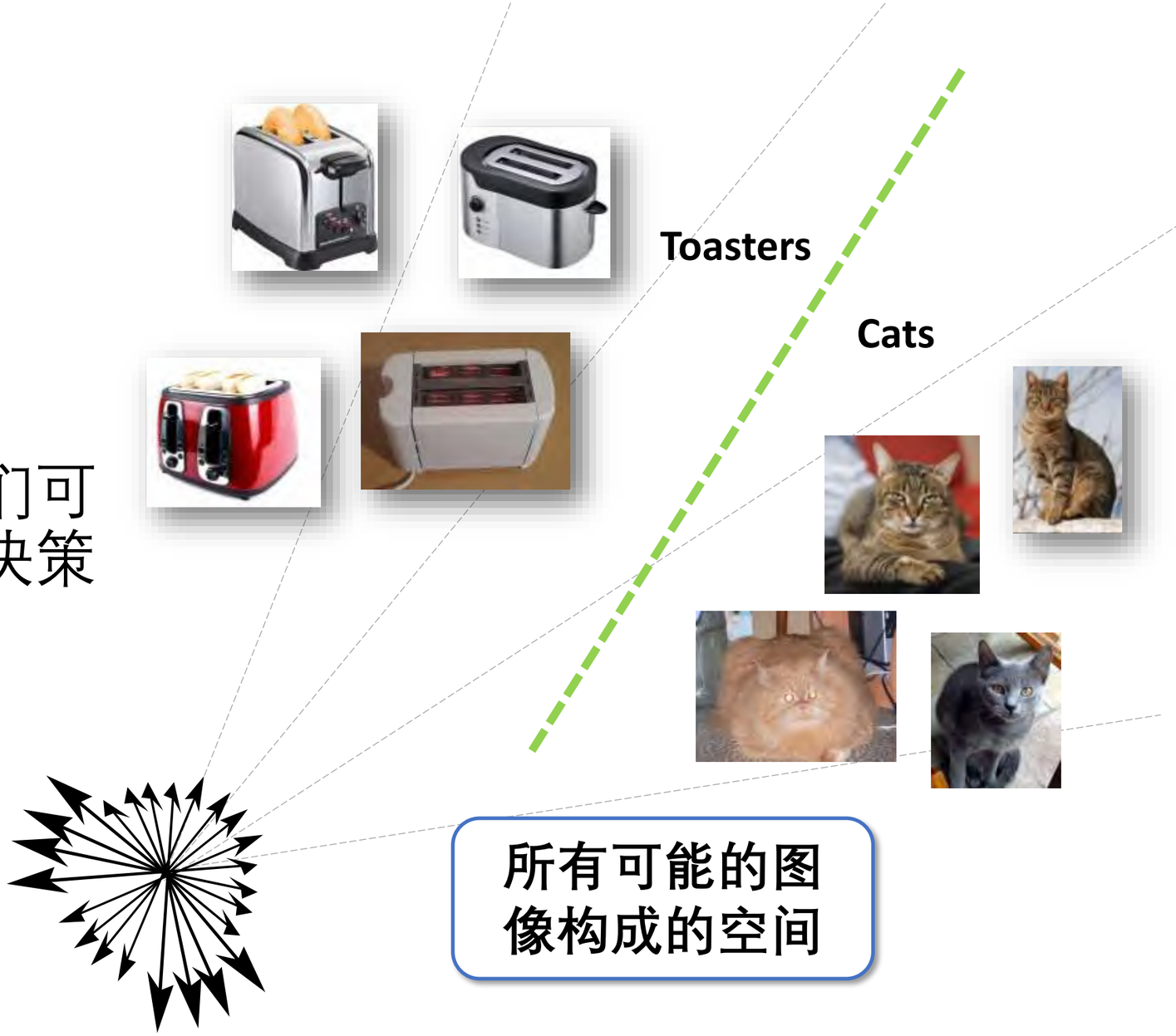
$$f(x) = \underset{\ell_x}{argmax}\ P(\ell_x | data)$$

将图像视为高维向量

- **图像只是一组数字**：每幅图像都可以看作是一个由像素值组成的数字数组。对于灰度图像，这些数字通常是0到255之间的值。对于彩色图像，每个像素通常包含三个值（RGB）。
- **将图像转换为向量**：为了便于计算和处理，我们可以将这些数字堆叠成一个向量。例如，对于一个100x100的灰度图像，我们可以将它转换为一个10,000维的向量。
- **训练数据变为高维点集**：当我们有多幅图像时，每幅图像都可以转换为一个高维向量。因此，我们的训练数据集就可以看作是在高维空间中的一组点。

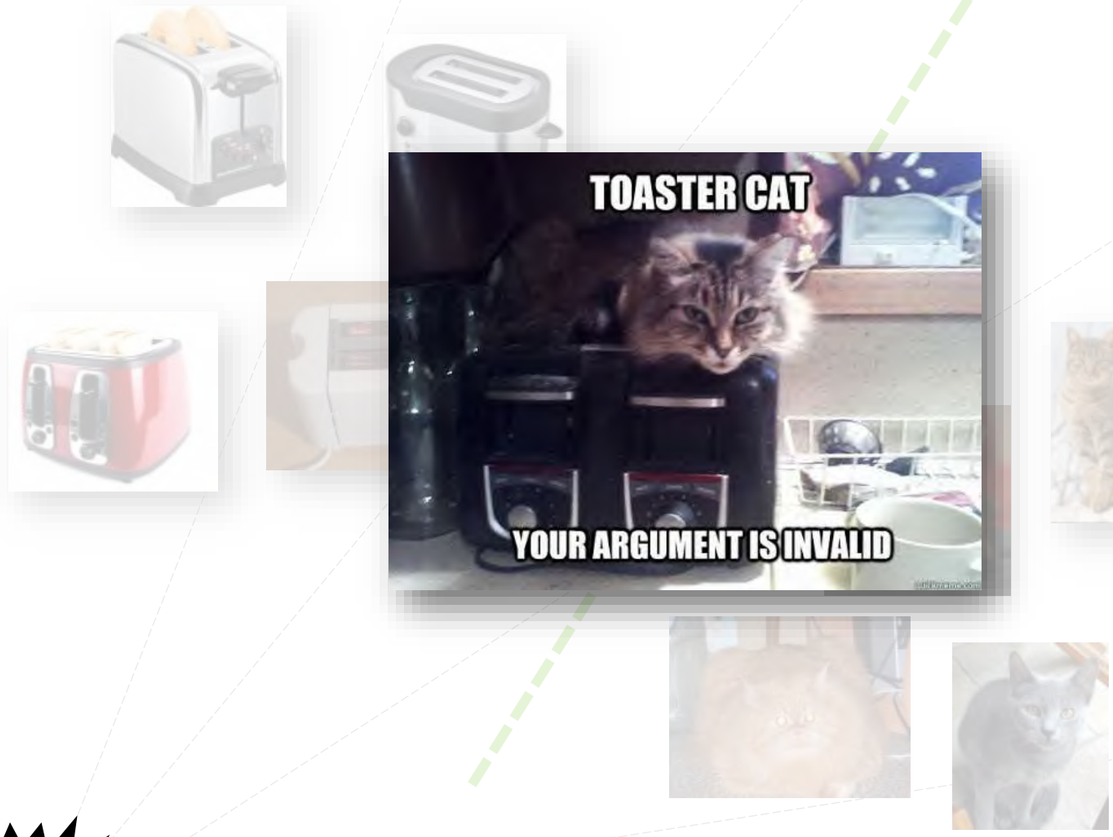
将图像视为高维向量

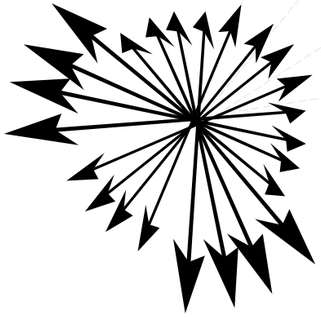
- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。



将图像视为高维向量

- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。





所有可能的图像构成的空间

将图像视为高维向量

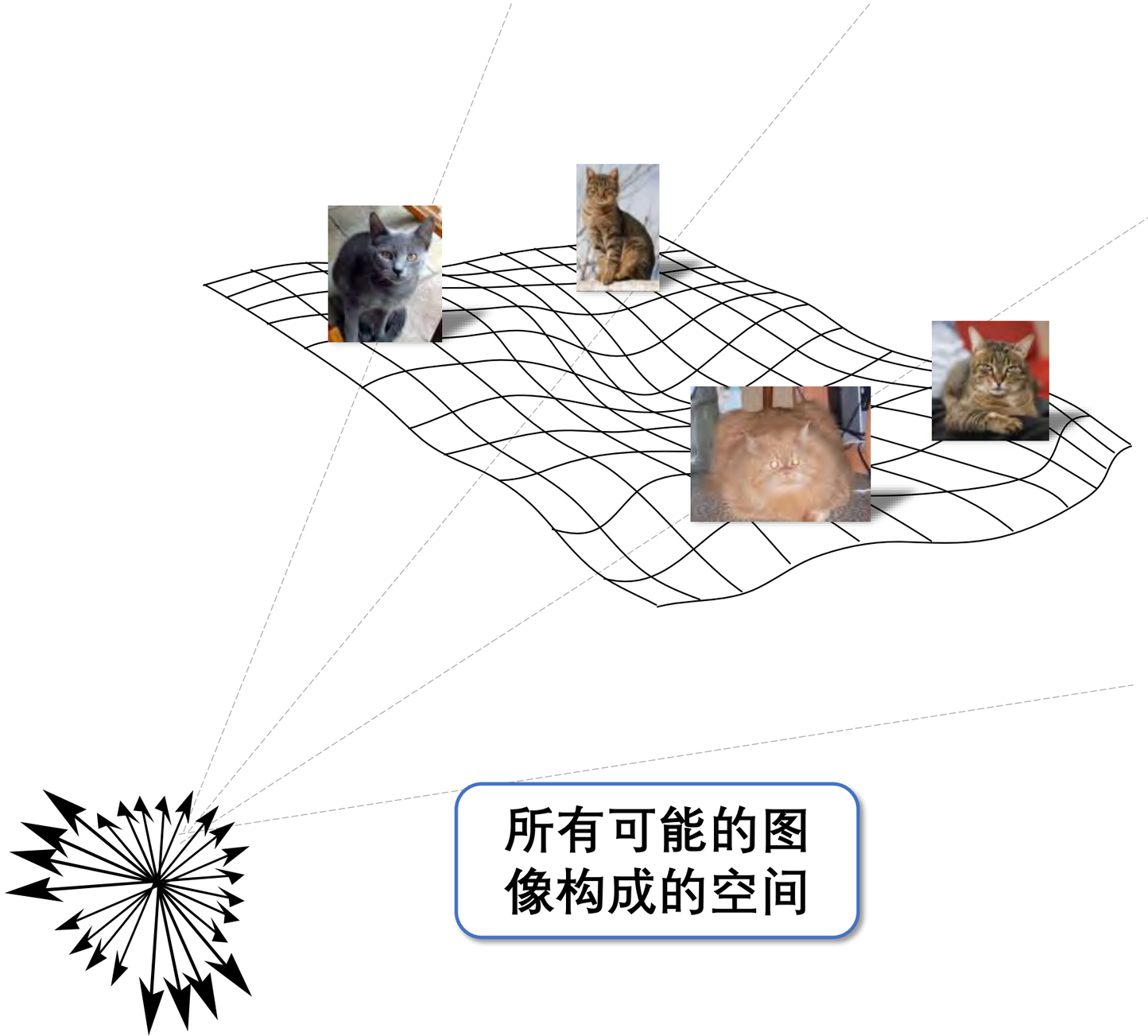
- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。

或者

- 为每个类别定义一个概率分布。
  - 来允许toaster-cat的存在。

图像的高维性 与 降维表示

- 一个图像有多少维度?
  - 图像的所有像素和它们的颜色通道。
  - iPhone X 照片:
    - 4032 x 3024 pixels
    - 3 colors
    - 36,578,304 pixels (36.5 Mega pixels)
- 在实际应用中，虽然图像可能具有高度的维度，但许多图像都倾向于聚集在某种低维结构或流形上。
- 降维可以帮助我们提取图像的关键特征，并用更简洁的方式表示图像。



所有可能的图像构成的空间