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

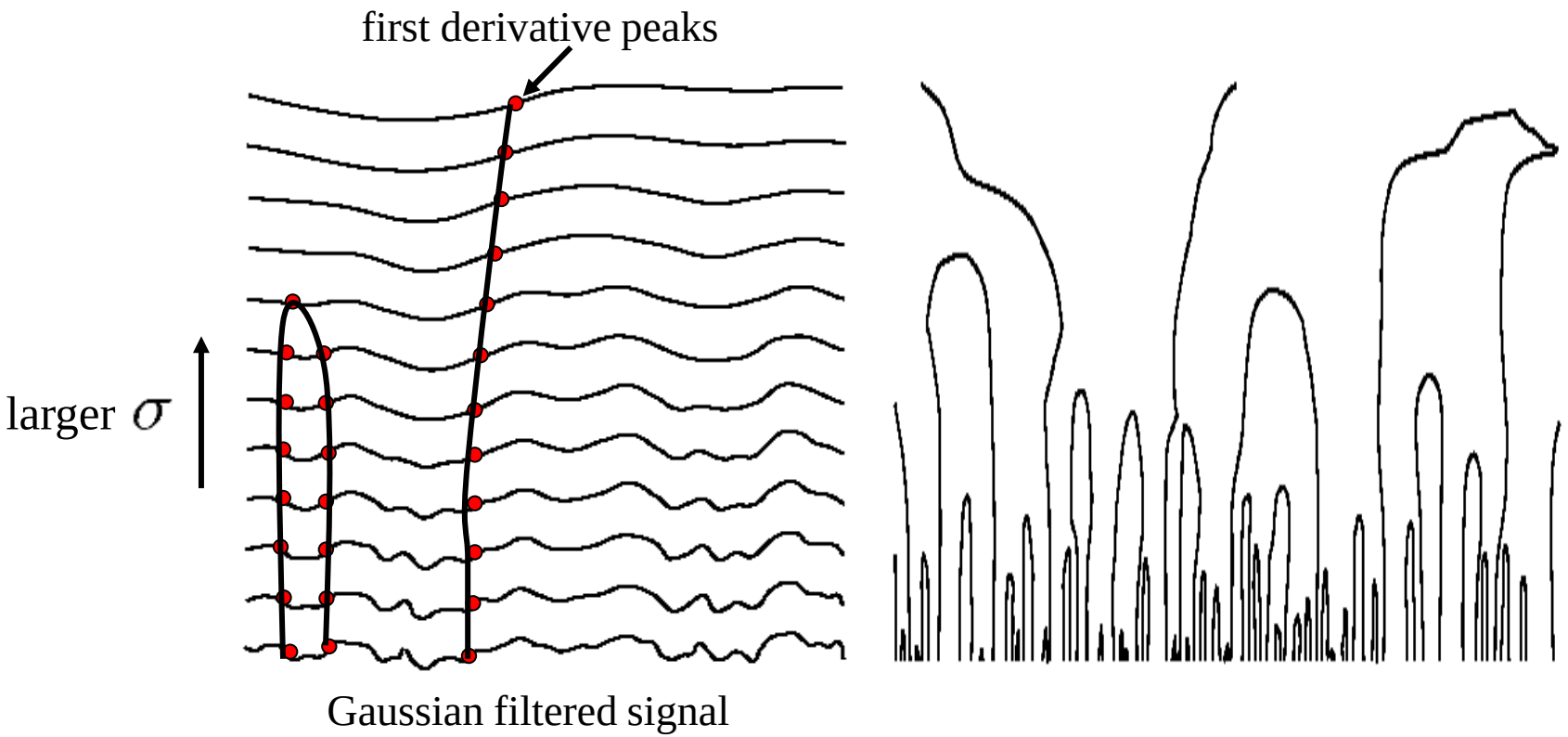
Canny with $\sigma = 1$

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”

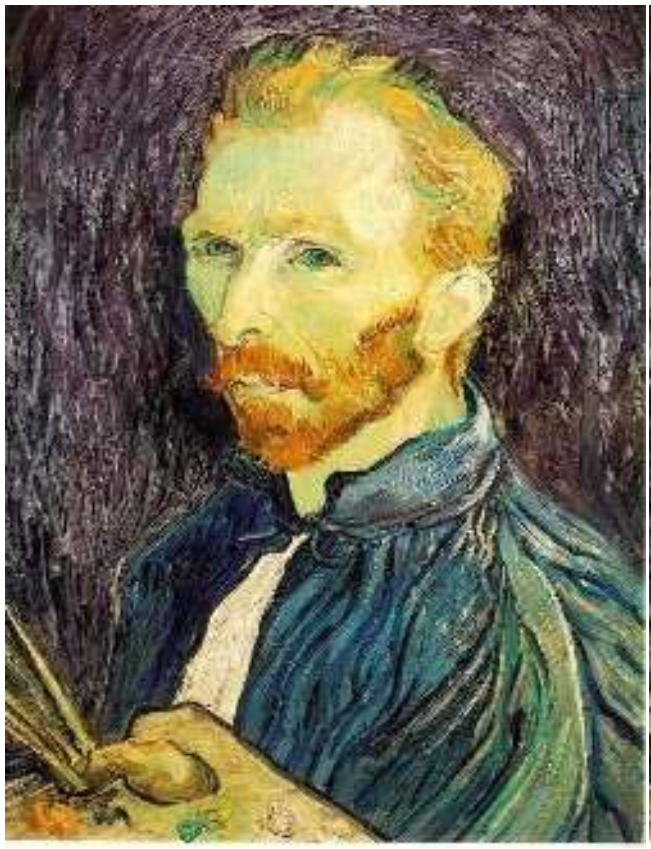
Source: S. Seitz

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

图像下采样



1/2

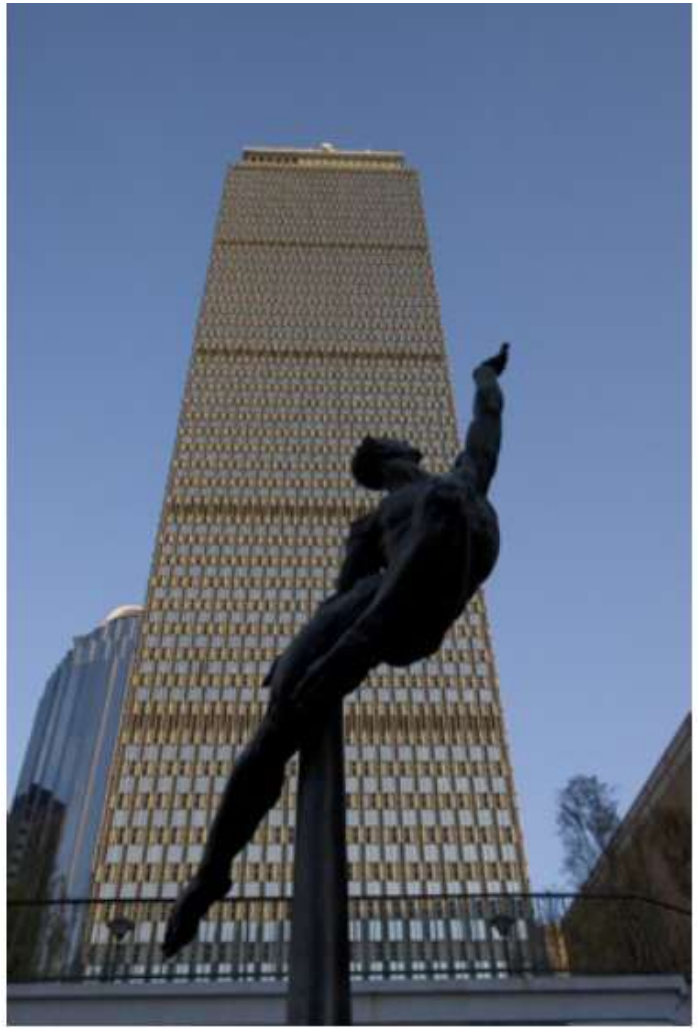
1/4 (2x zoom)

1/8 (4x zoom)

如果再上采样回来，为什么看起来这么粗糙?

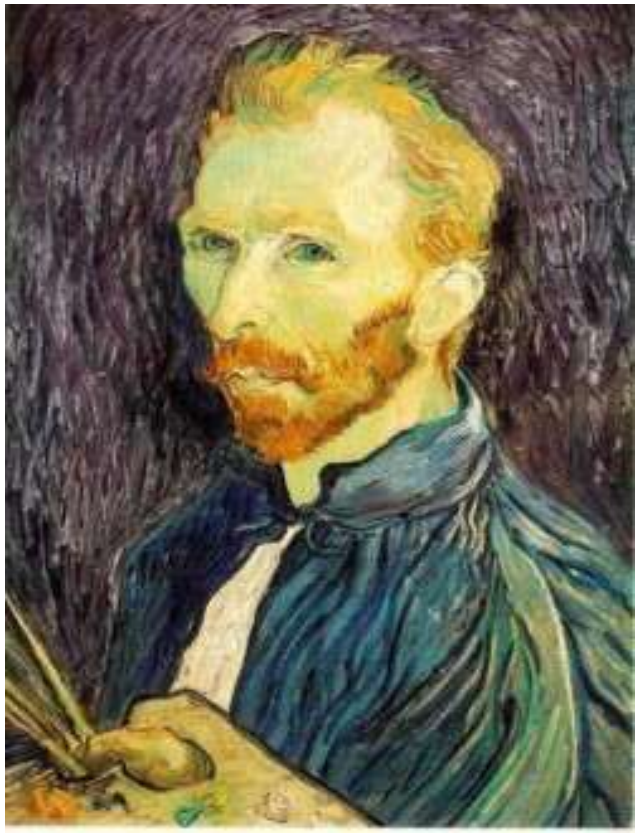
Source: S. Seitz

图像下采样：变形



Source: F. Durand

使用高斯模糊以后的下采样



Gaussian 1/2

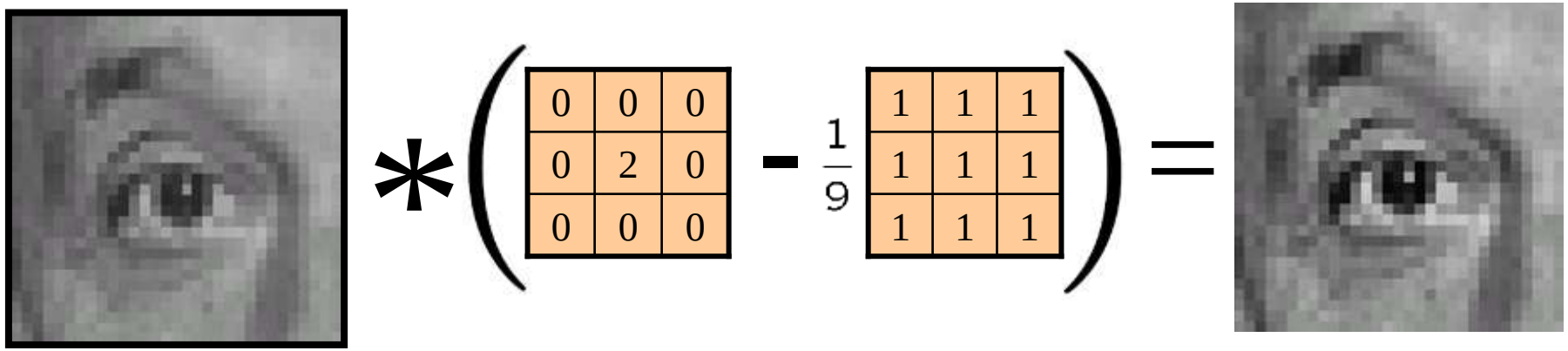
G 1/4

G 1/8

- 解决方案：先模糊，再下采样

Source: S. Seitz

线性滤波

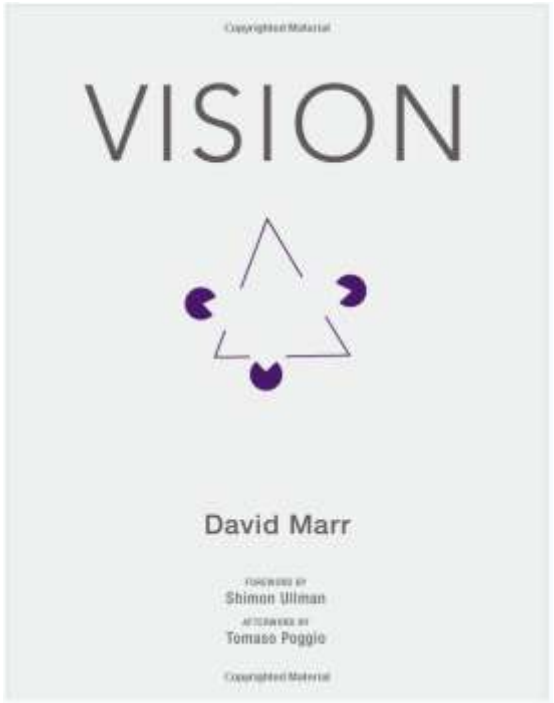


原图

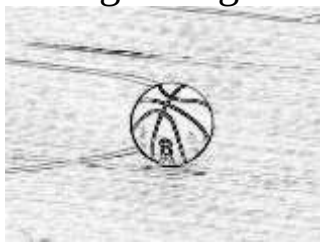
锐化后
(突出边缘)

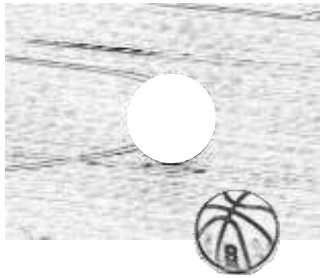
Source: D. Lowe

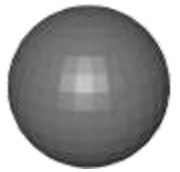
角点检测

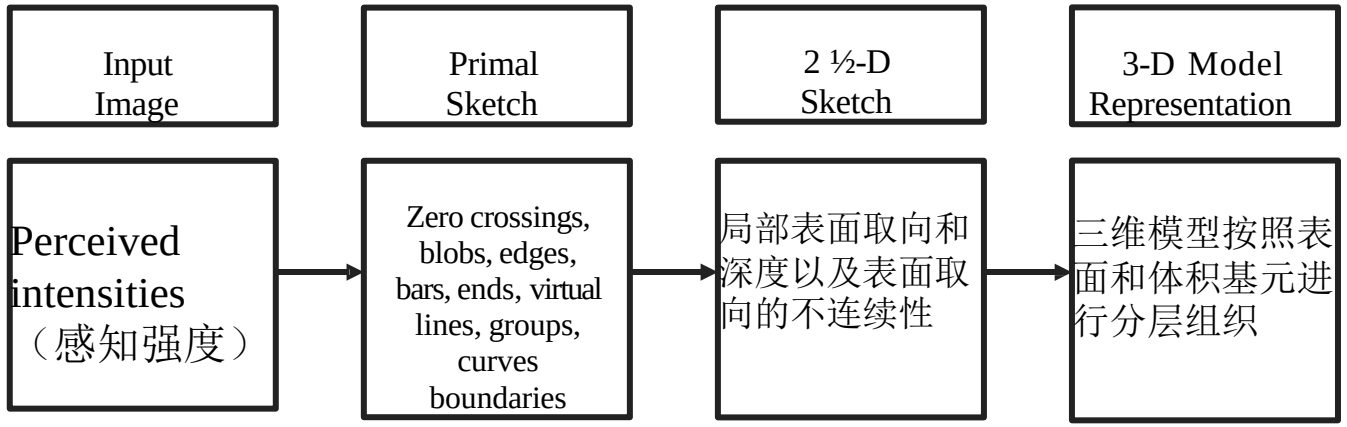


This image is CC0 1.0 public domain





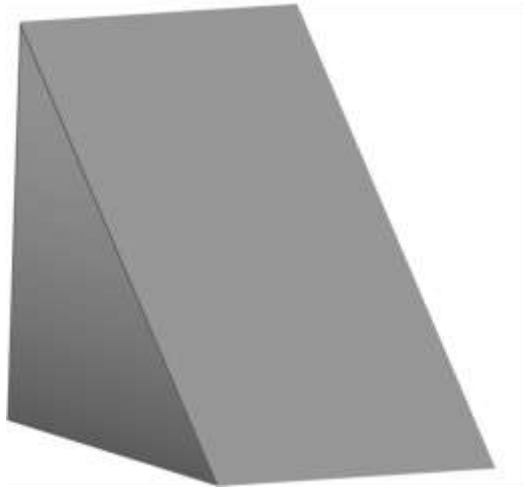




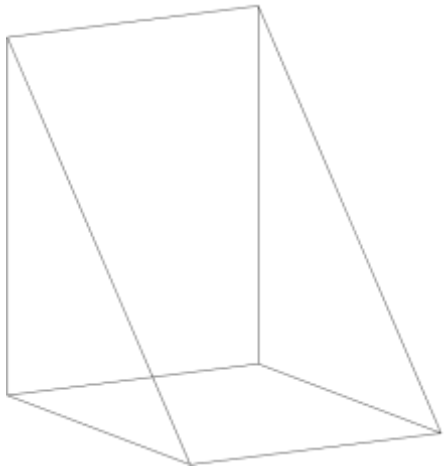
“二维半素描”：通过从不同视角获取的二维图像推断出的一些关于物体表面形状和结构的简化信息

Stages of Visual Representation, David Marr, 1970s

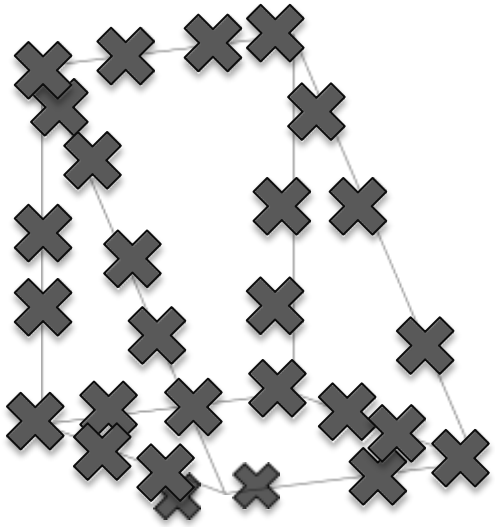
特征点



(a) Original picture



(b) Differentiated picture



(c) Feature points selected

从二维图像提取形状和三维信息

Lawrence Gilman Roberts, “Machine Perception of Three-Dimensional Solids”, 1963

全景图：整合多幅图

用一张图记录
一个场景！

轮转全景扫描

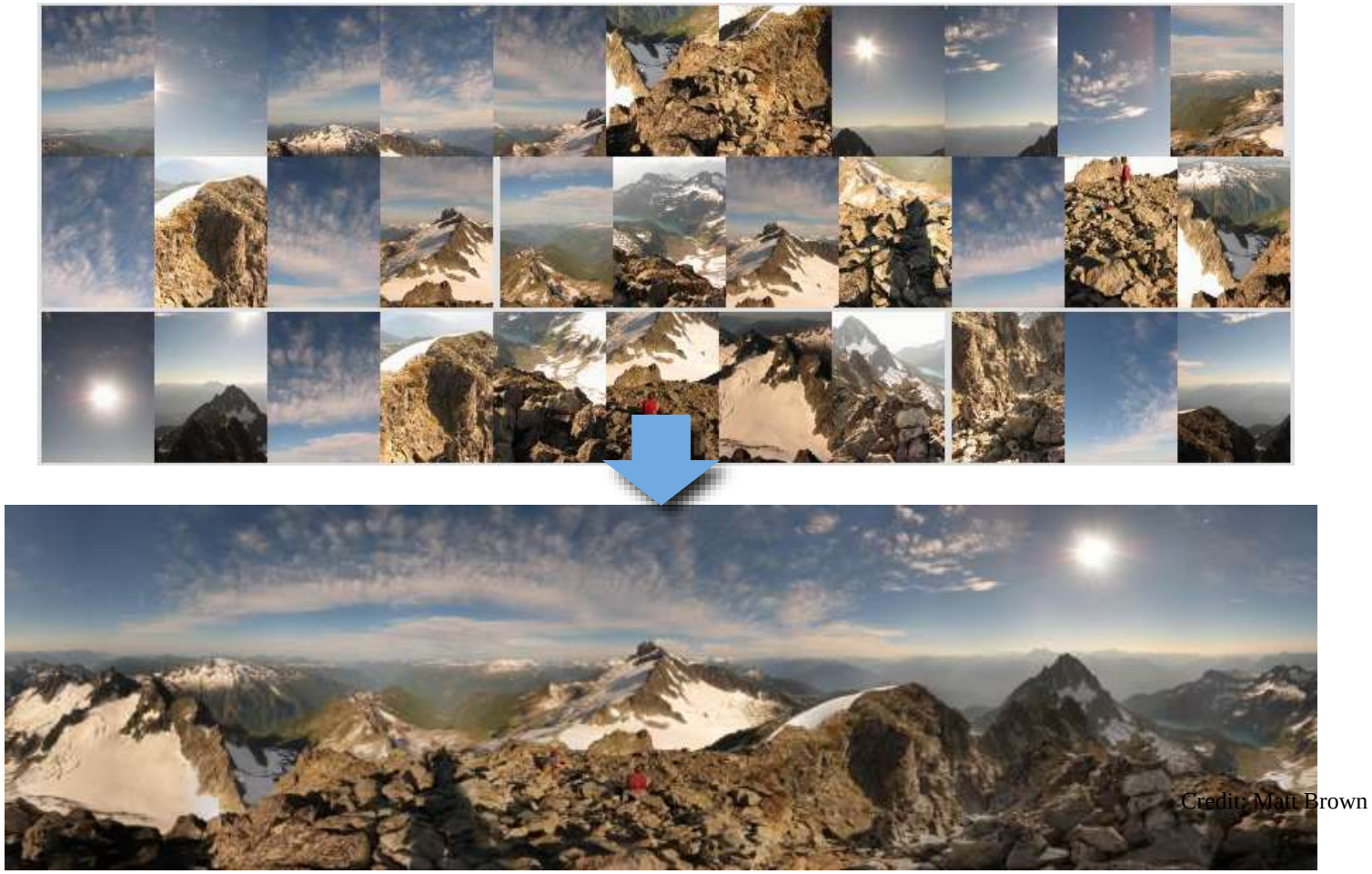
用一张图记录一个物体！

Rollout Photographs © Justin Kerr

http://research.famsi.org/kerrmaya.html

Also known as “cyclographs”, “peripheral images”

更大的场景…

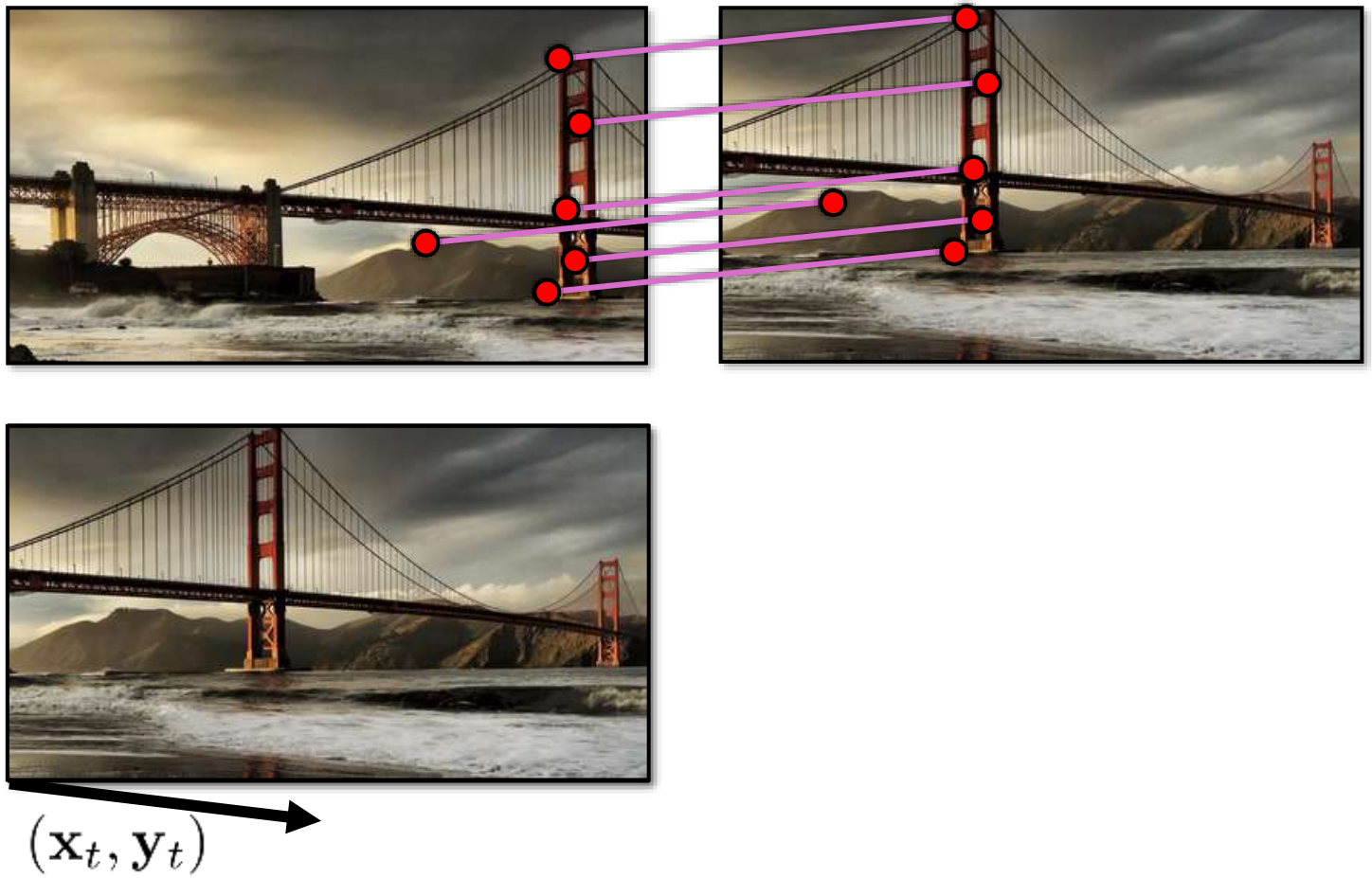


Credit: Matt Brown

如何自动获取全景图: 图像拼接

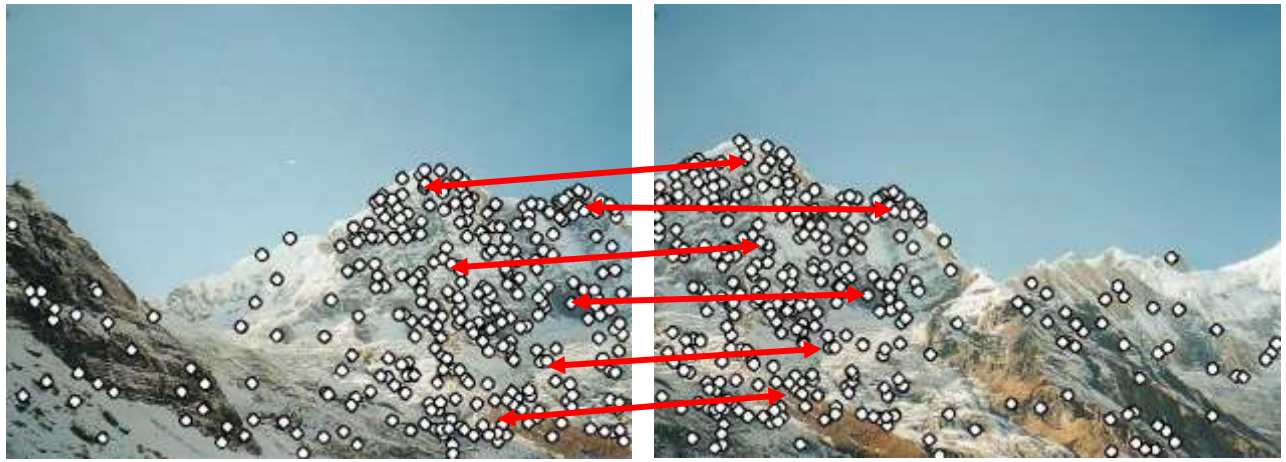
- 怎么把两张图像结合起来?


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？

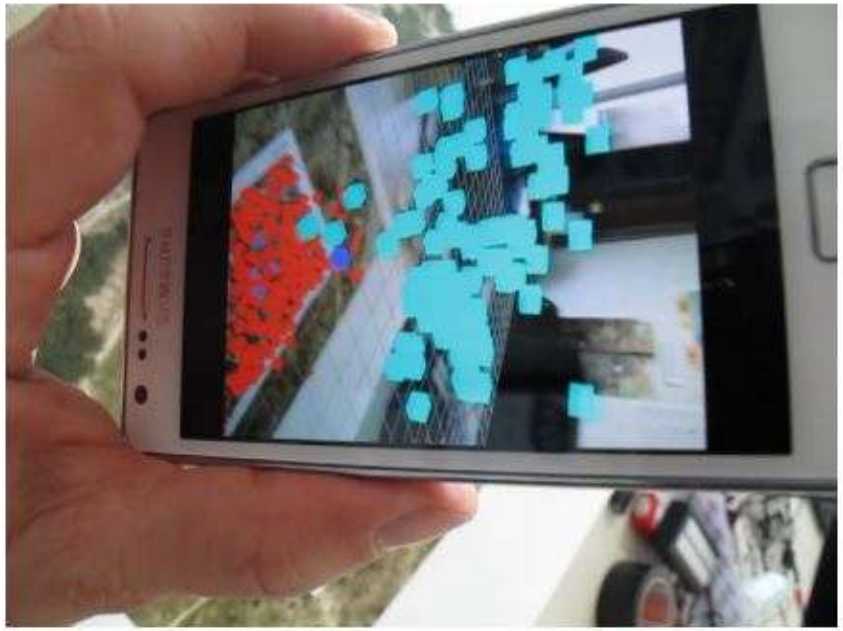
Step 1: 提取特征

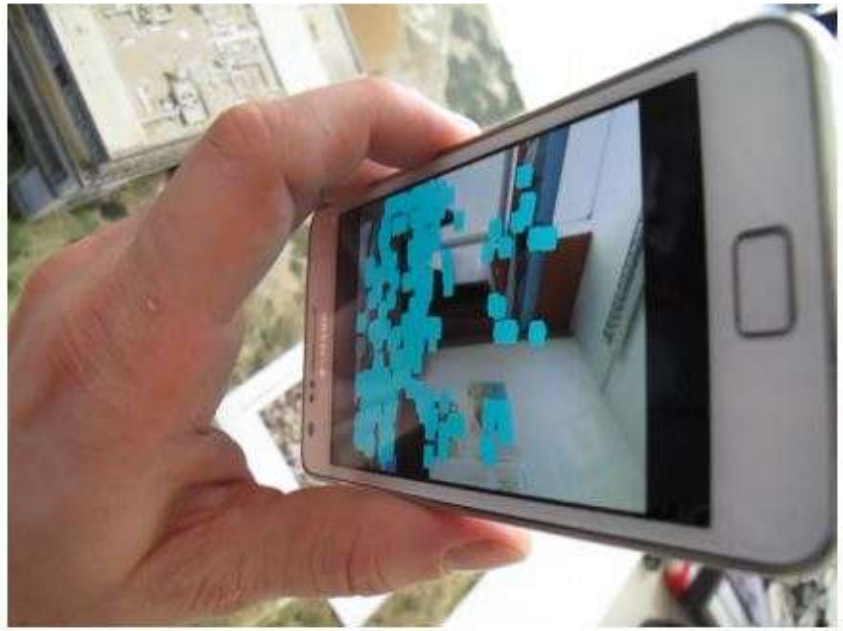
Step 2: 特征匹配

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

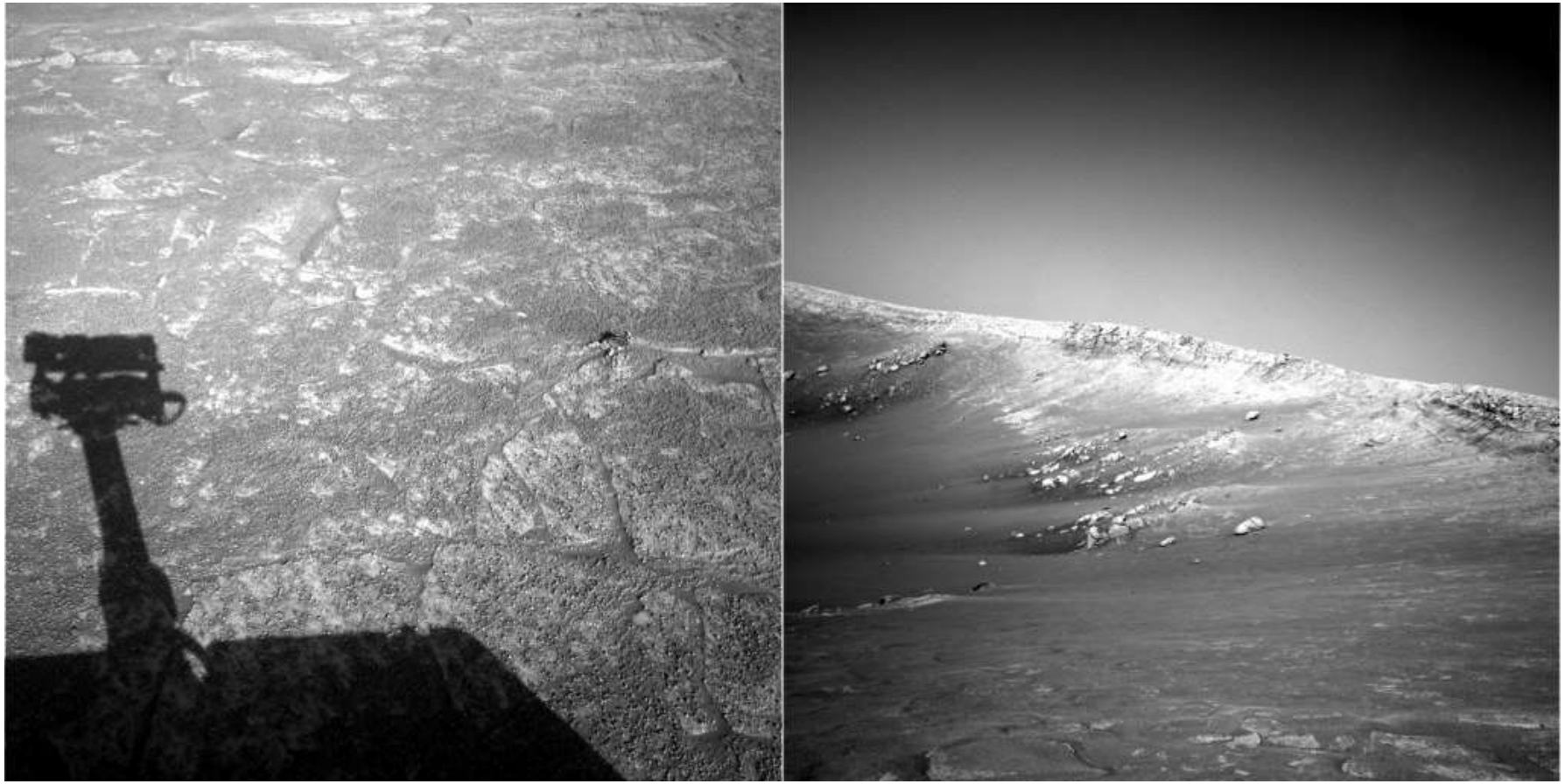
by swashford

加大难度: 视角变化

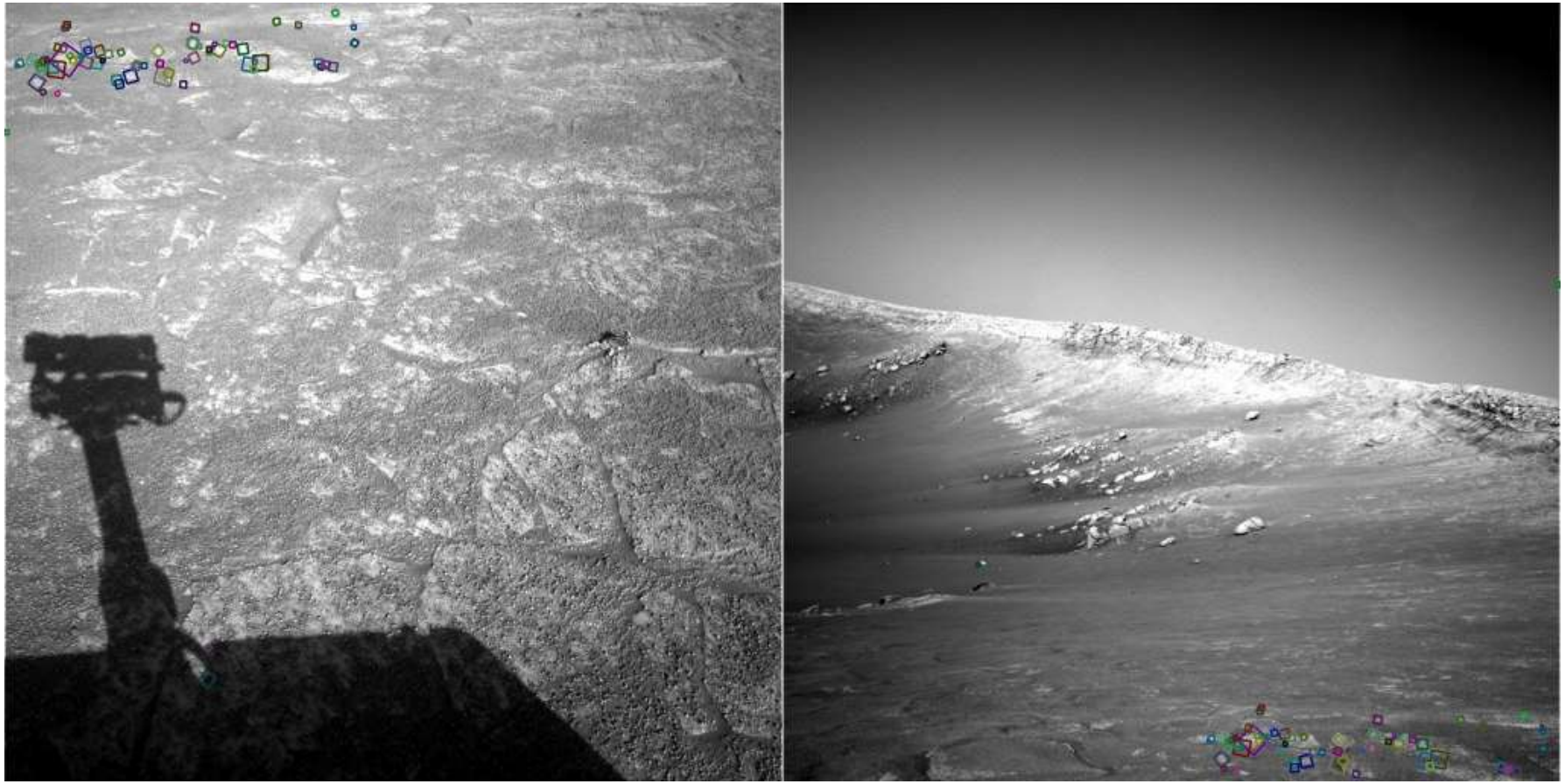
by Diva Sian

by scgbt

更难的情况：你能找到匹配的位置吗

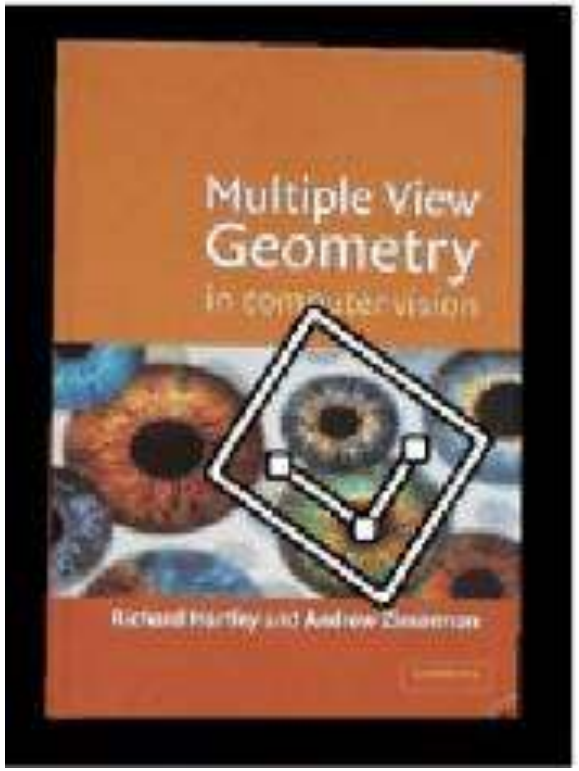


答案



NASA Mars Rover images
with SIFT feature matches

应用：物体识别与检测

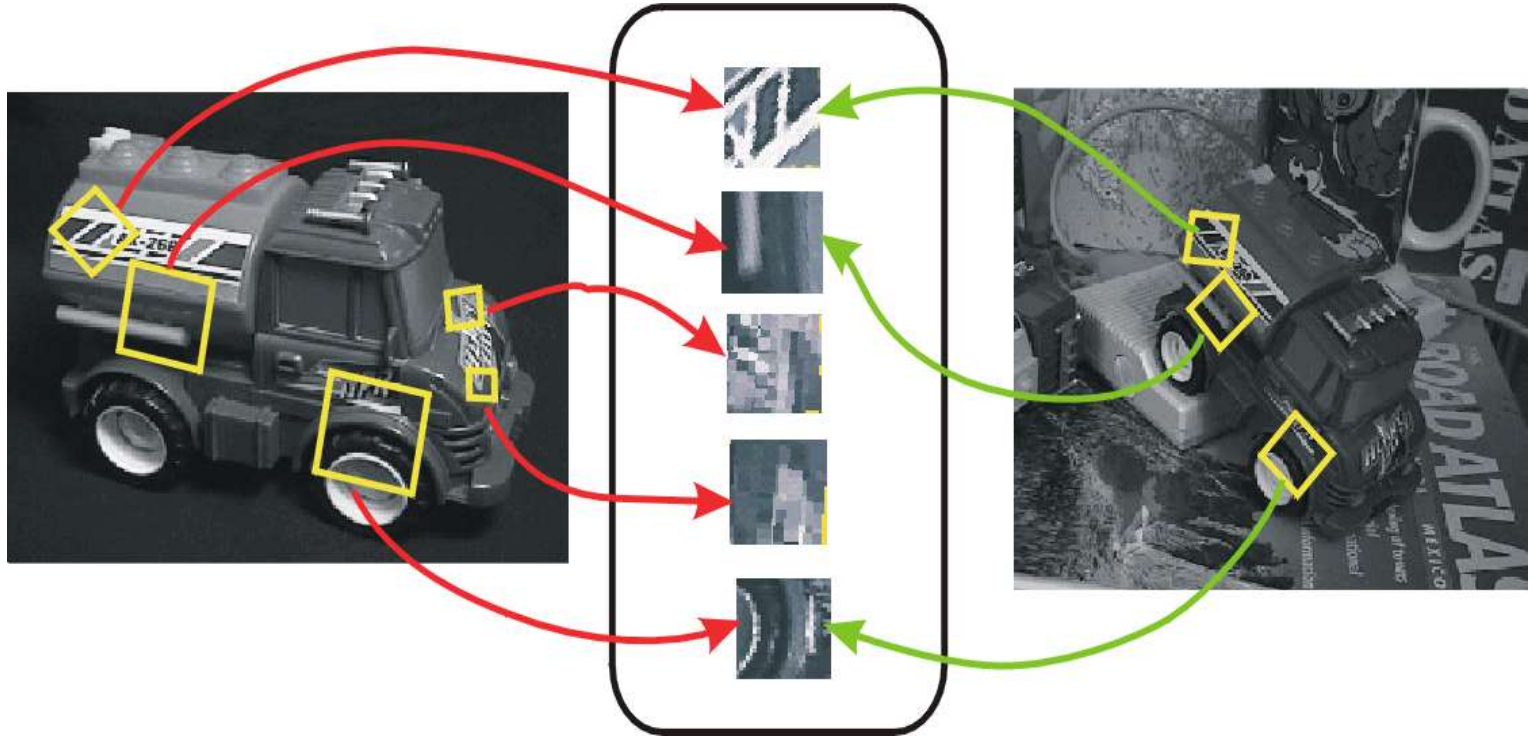


特征匹配

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：

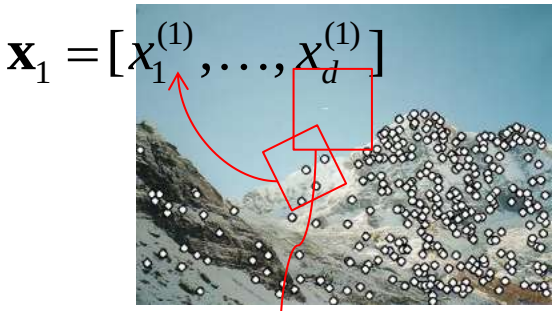
- 适当设计算法可以达到实时性处理

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



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

Kristen Grauman

什么样的特征是好的特征?

Uniqueness：唯一性

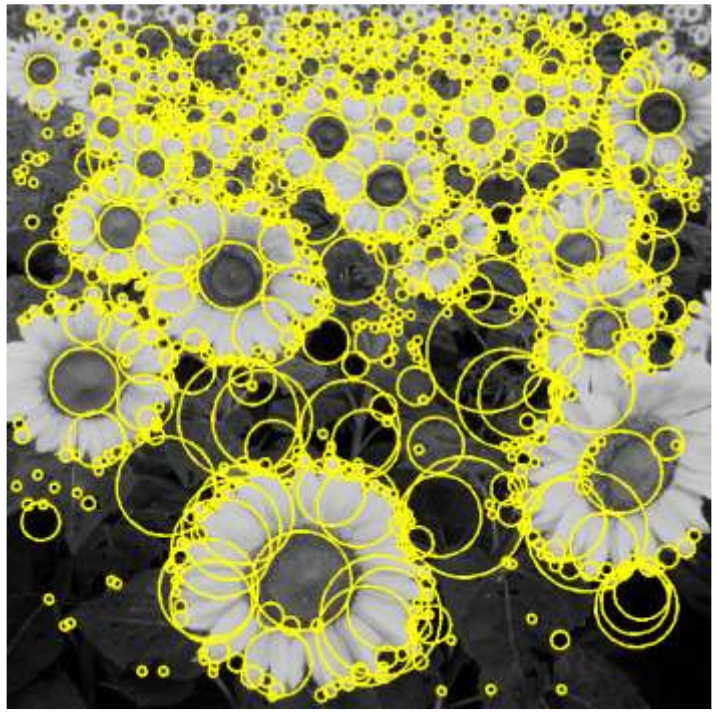
特征点在图像或图像集合中具有**独特**的外观或属性

- 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

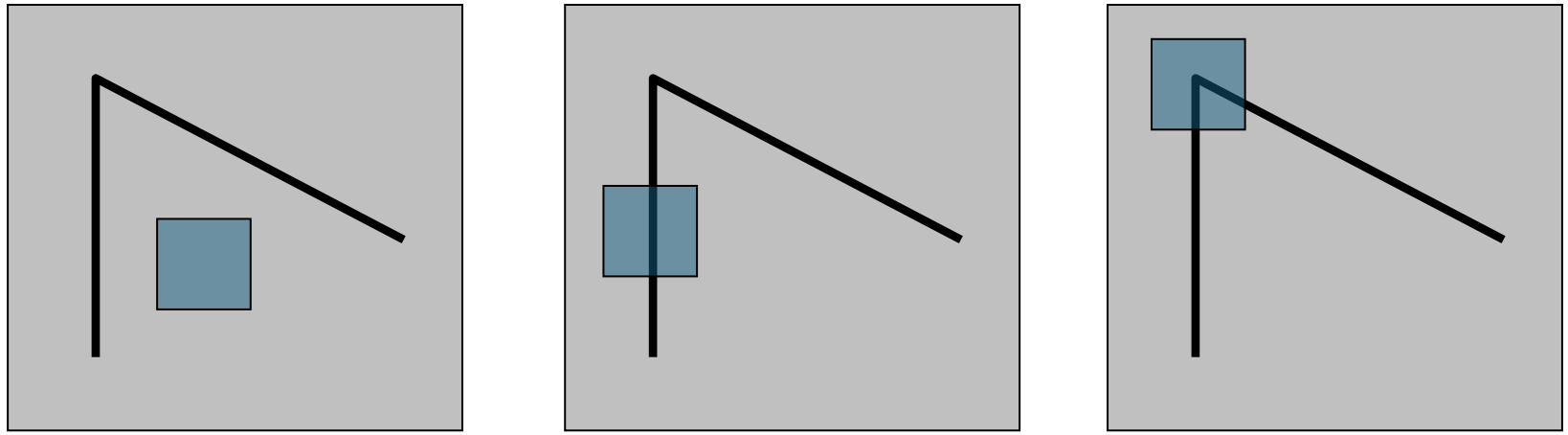
- 角点（Corners）：角点是图像中具有明显边缘转折的像素点。



衡量局部的“唯一性”

针对一个小的区域（滑动窗口）

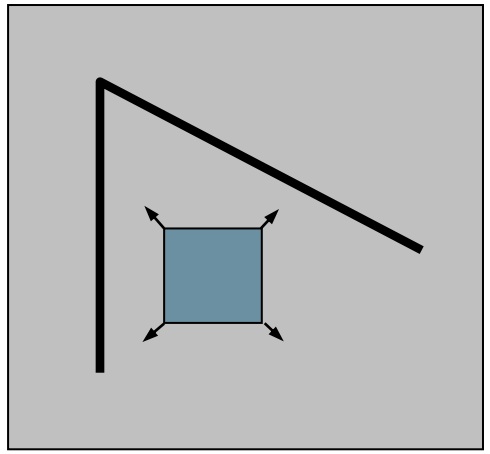
- 什么样的点附近区域代表更好的特征点?
- 我们从一个“物体”内去考虑“唯一性”
- 什么点在描述物体时更“独特”？



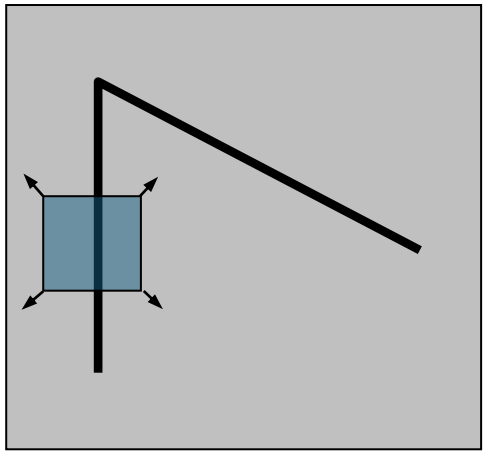
Credit: S. Seitz, D. Frolova, D. Simakov

衡量局部的“唯一性”

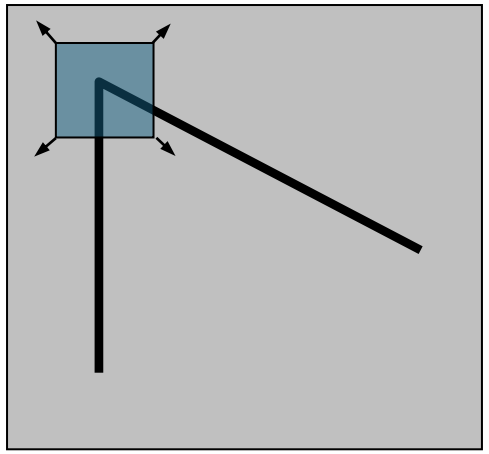
- 我们从窗口内提取特征（图像梯度、DoG）
- 当窗口向四周滑动，特征怎样变化?



“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

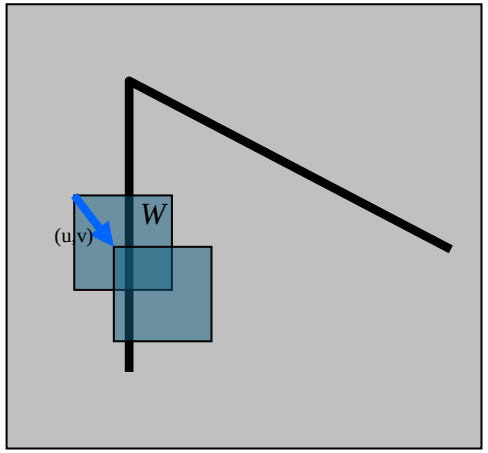
Harris 角点检测: 数学定义

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 $E(u,v)$:

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

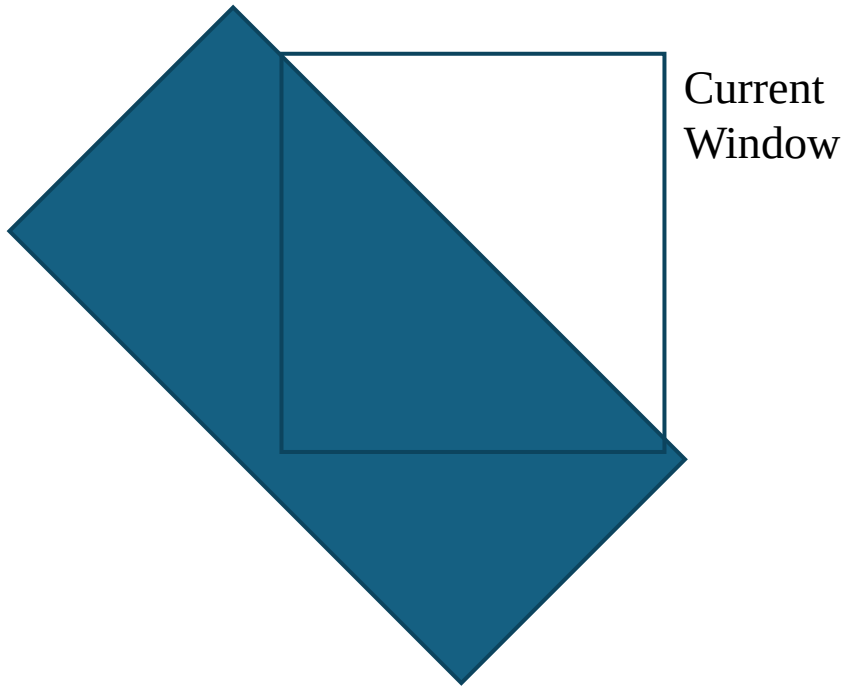


- SSD越高特征点越“唯一”

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

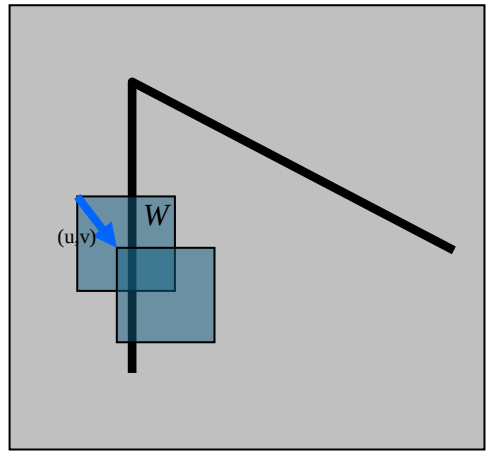
Harris 角点 – 为什么这么复杂?

- 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

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

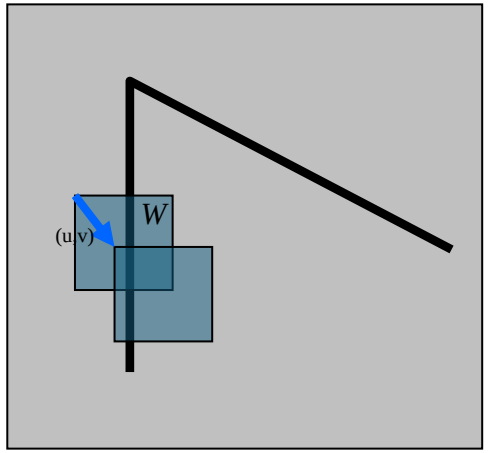
$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

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

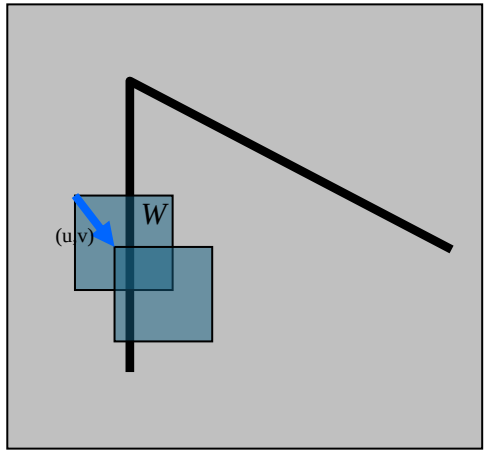
假设我们考虑窗口*W*滑动了 (*u*,*v*)

- 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

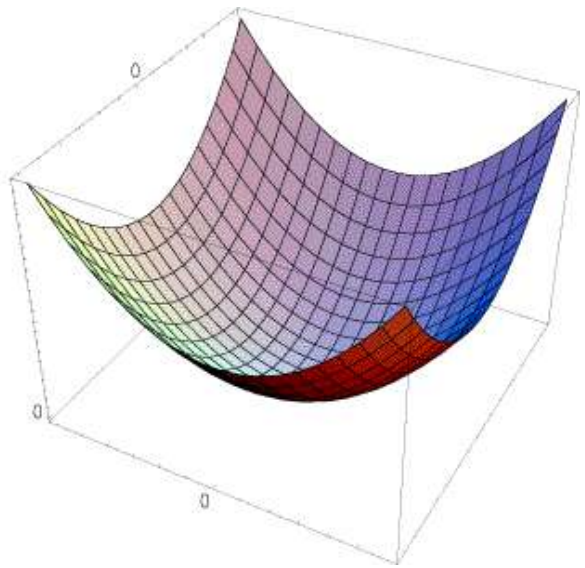
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

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

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

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

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

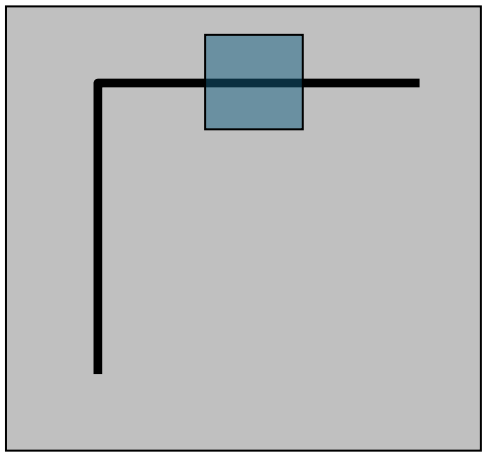


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

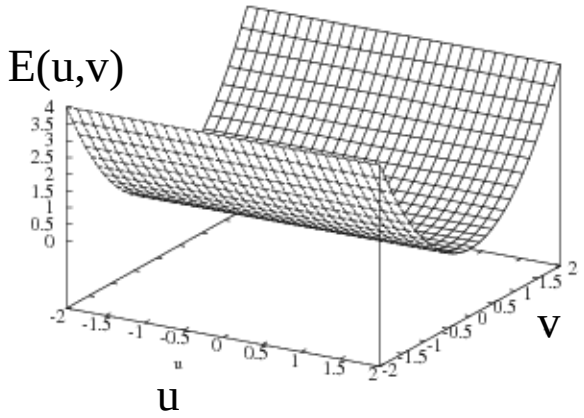
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



水平边缘: $I_x = 0$

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

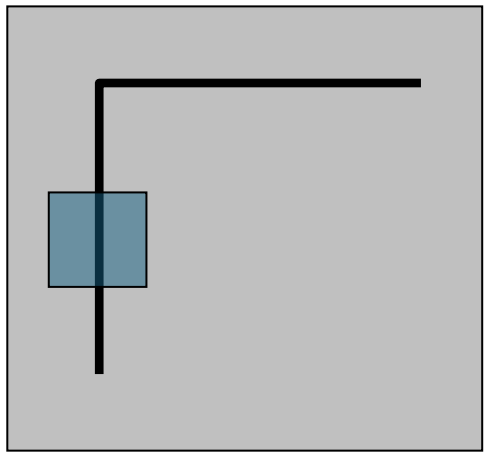


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

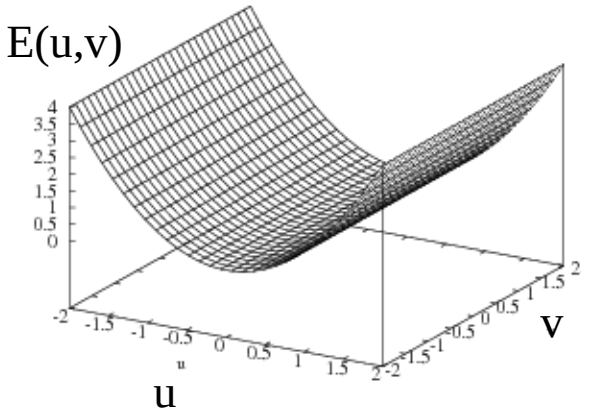
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



垂直边缘: $I_y = 0$

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

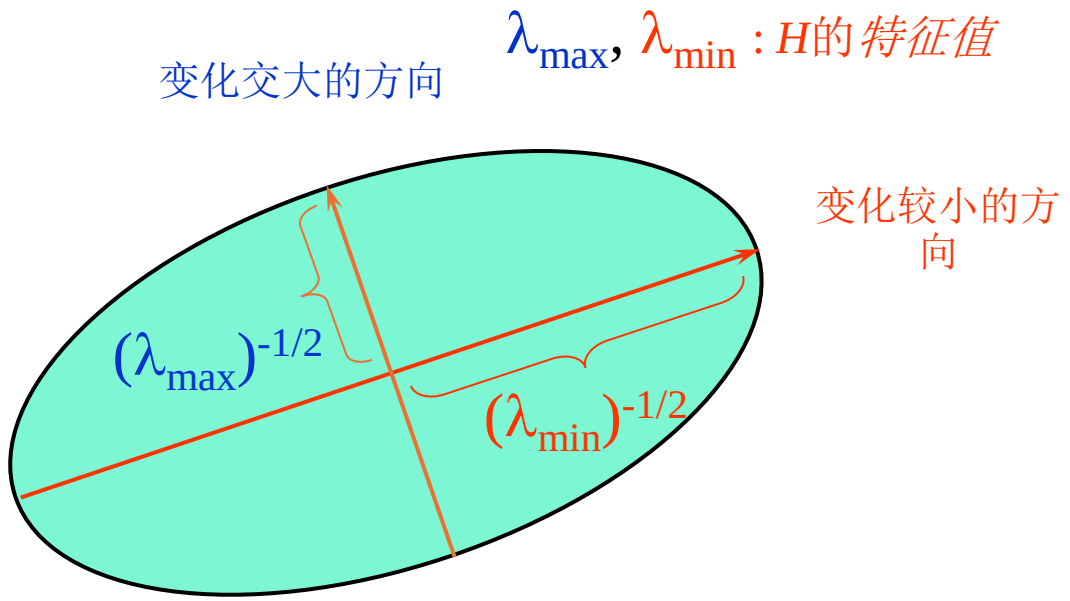


Harris 角点检测: 数学定义

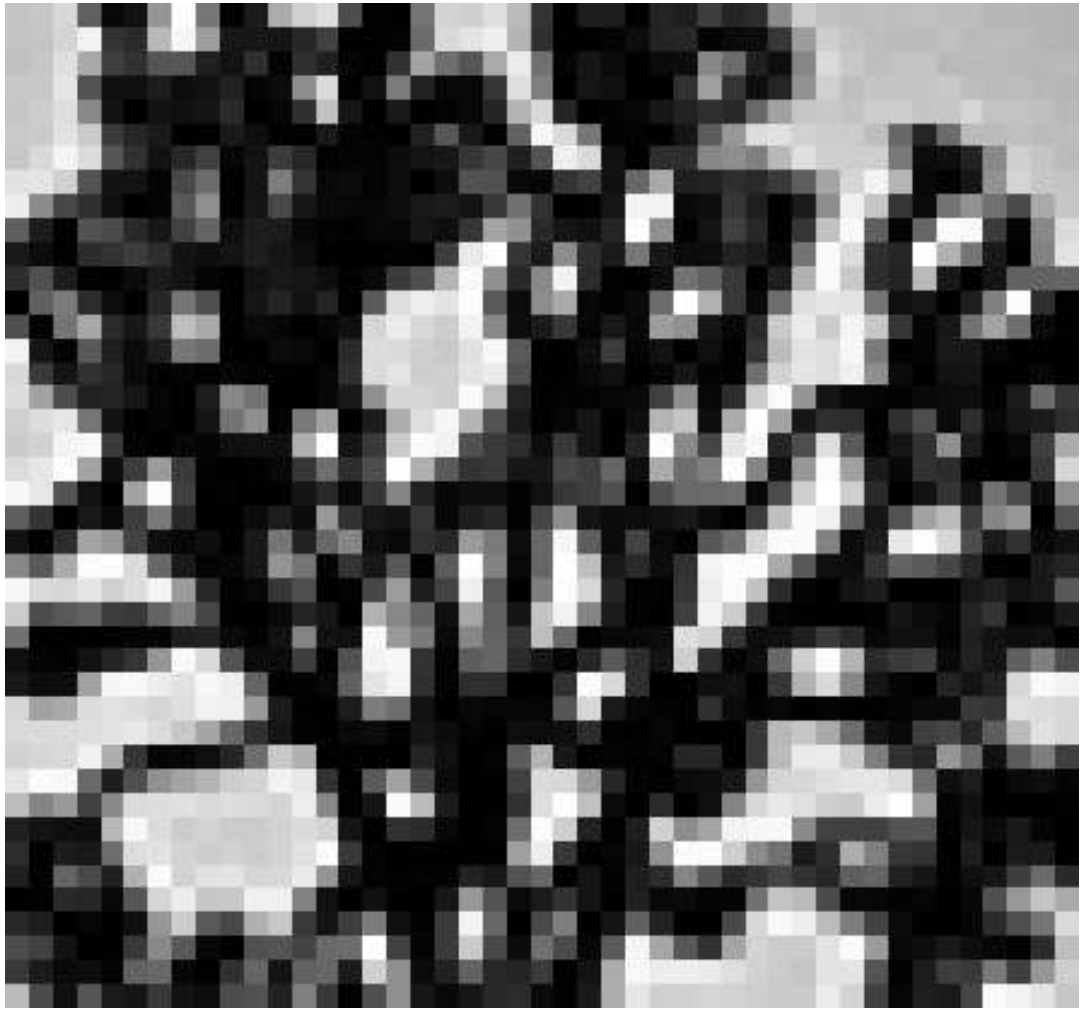
H 在各个方向上的变化速率可以通过H 的特征值和特征向量来可视化

椭圆方程:

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

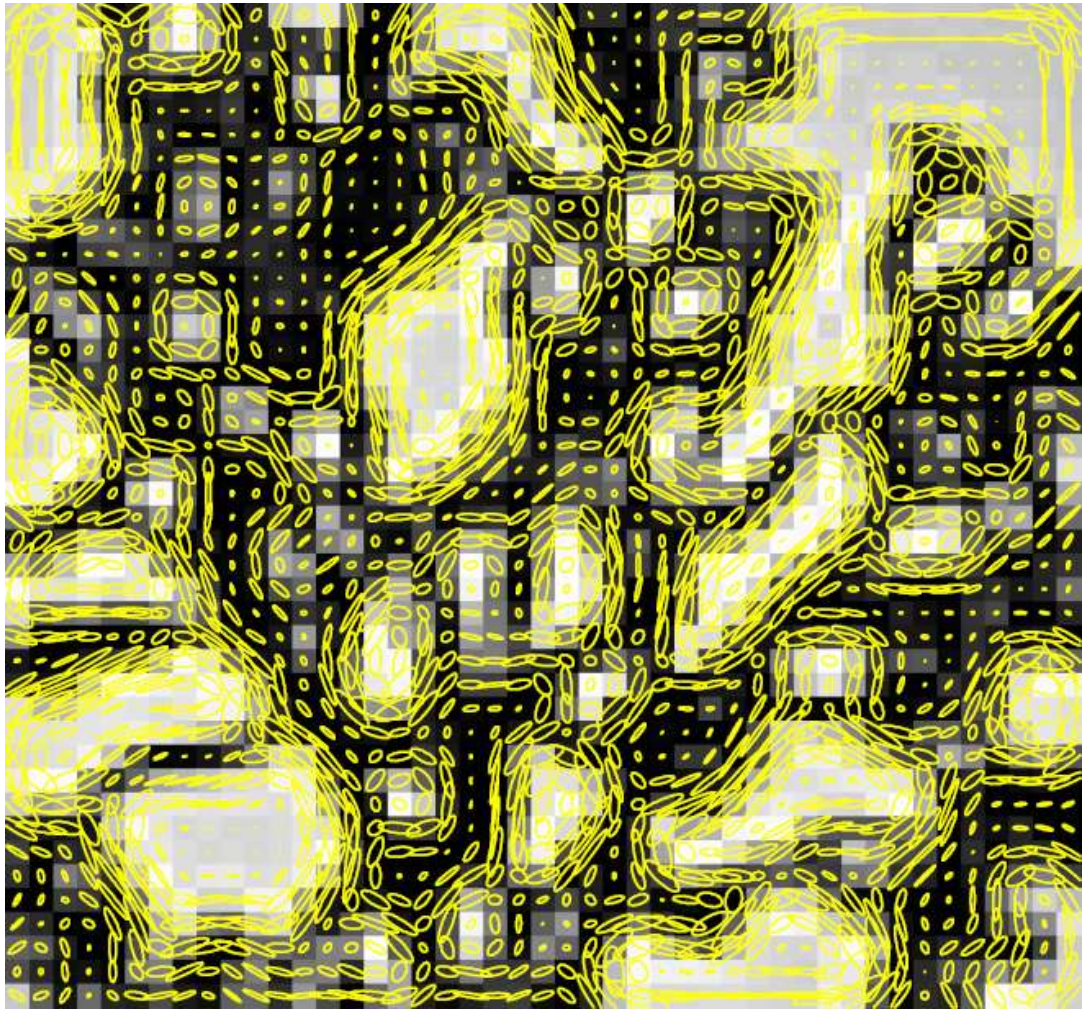


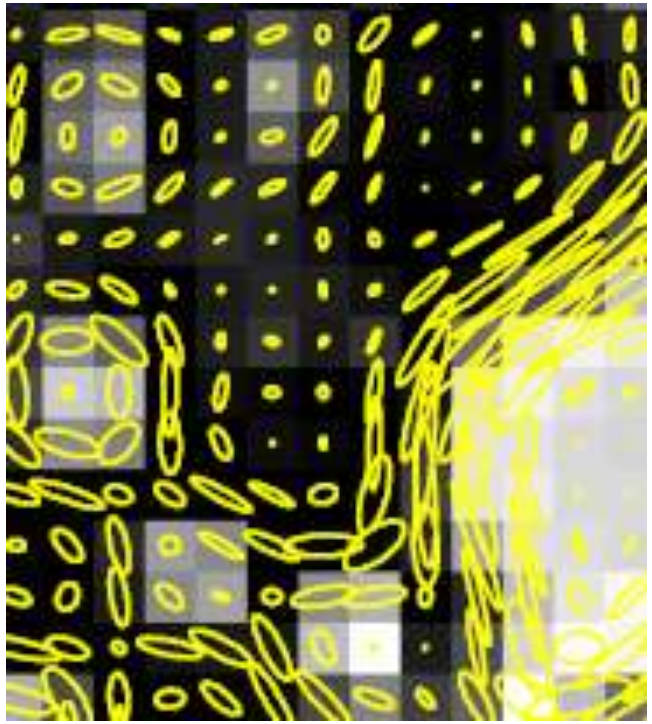
Visualizing M



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

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x**

- The eigenvalues are found by solving:

$$det(A - \lambda I) = 0$$

- In our case, ***A*** = ***H*** is a 2x2 matrix, so we have

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

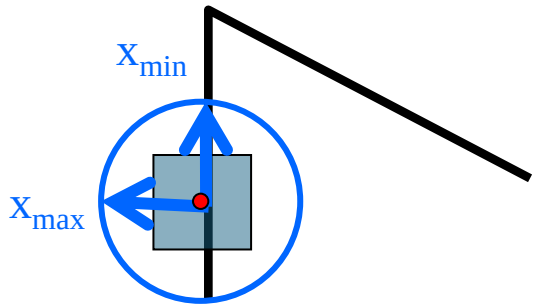
$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

Once you know λ, you find **x** by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

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?

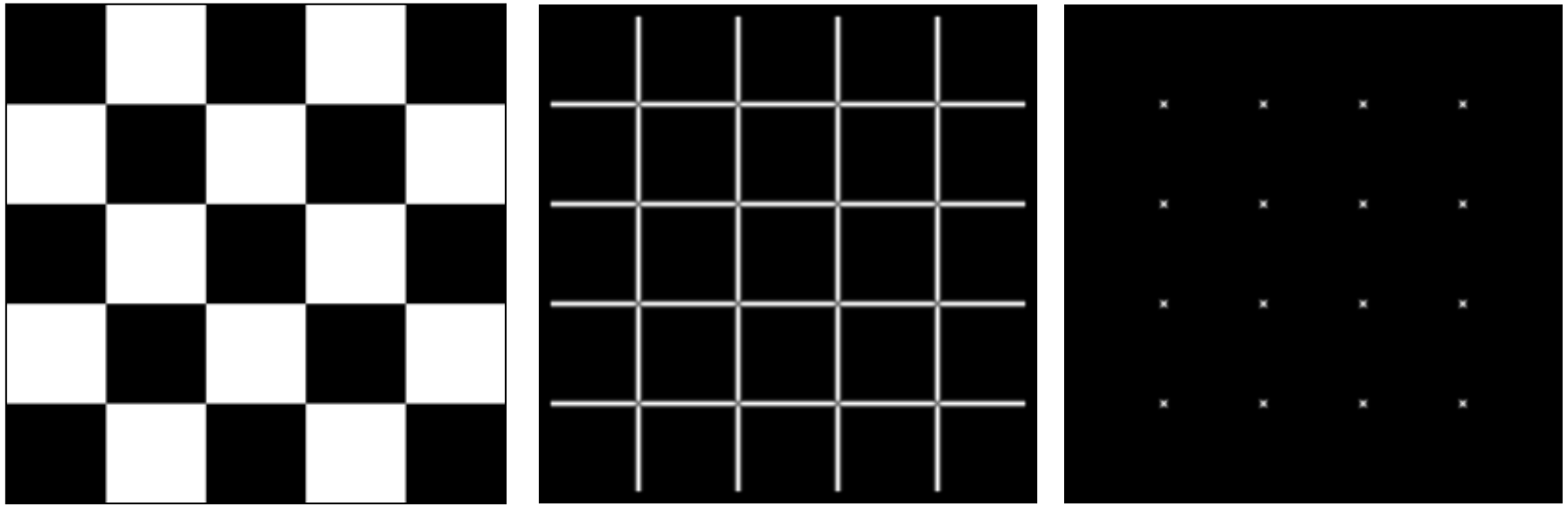
Harris 角点检测: 数学定义

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

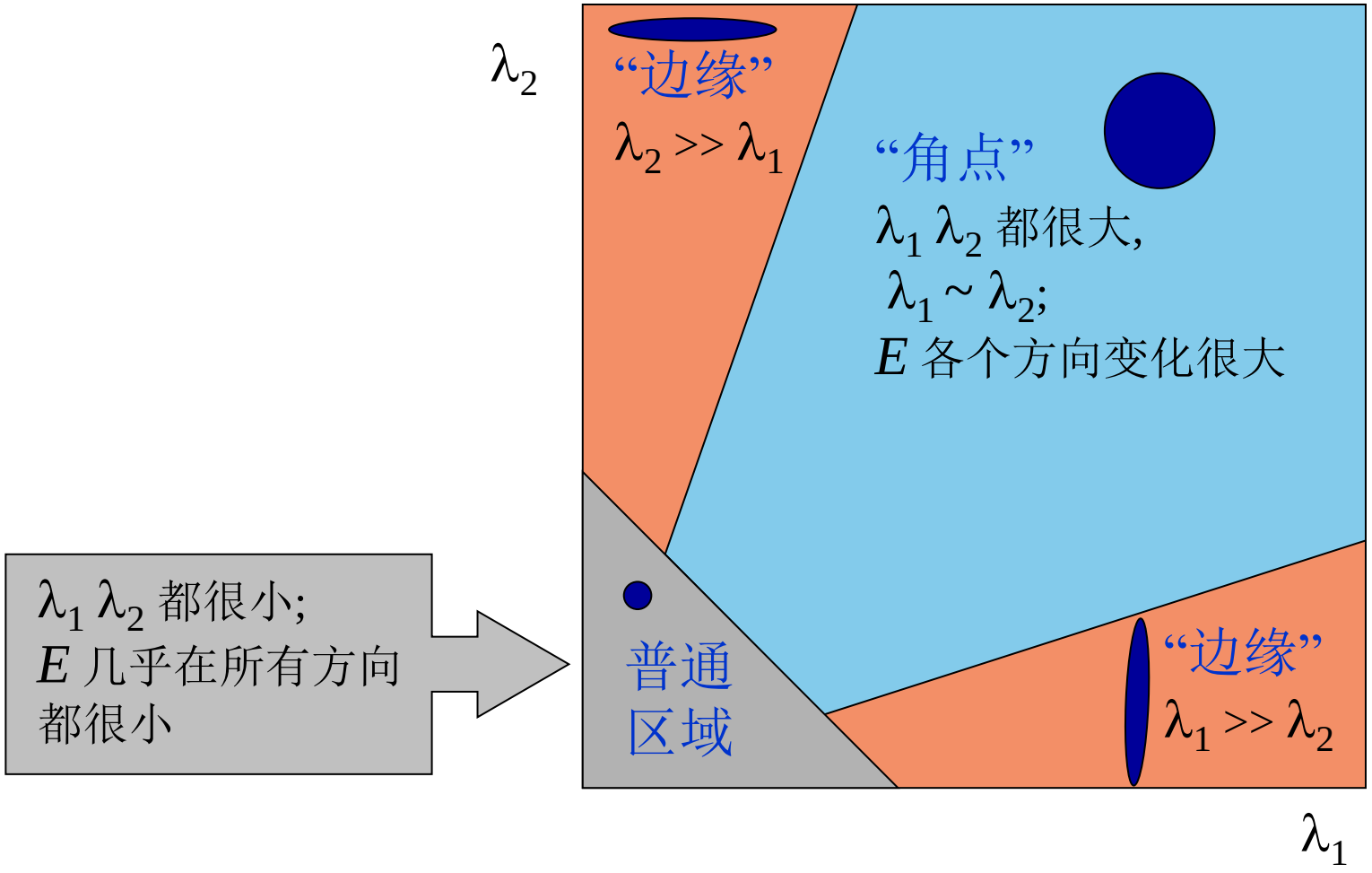
- 我们需要怎样的特征点?

我们需要$E(u,v)$ 在各个方向变化都很大

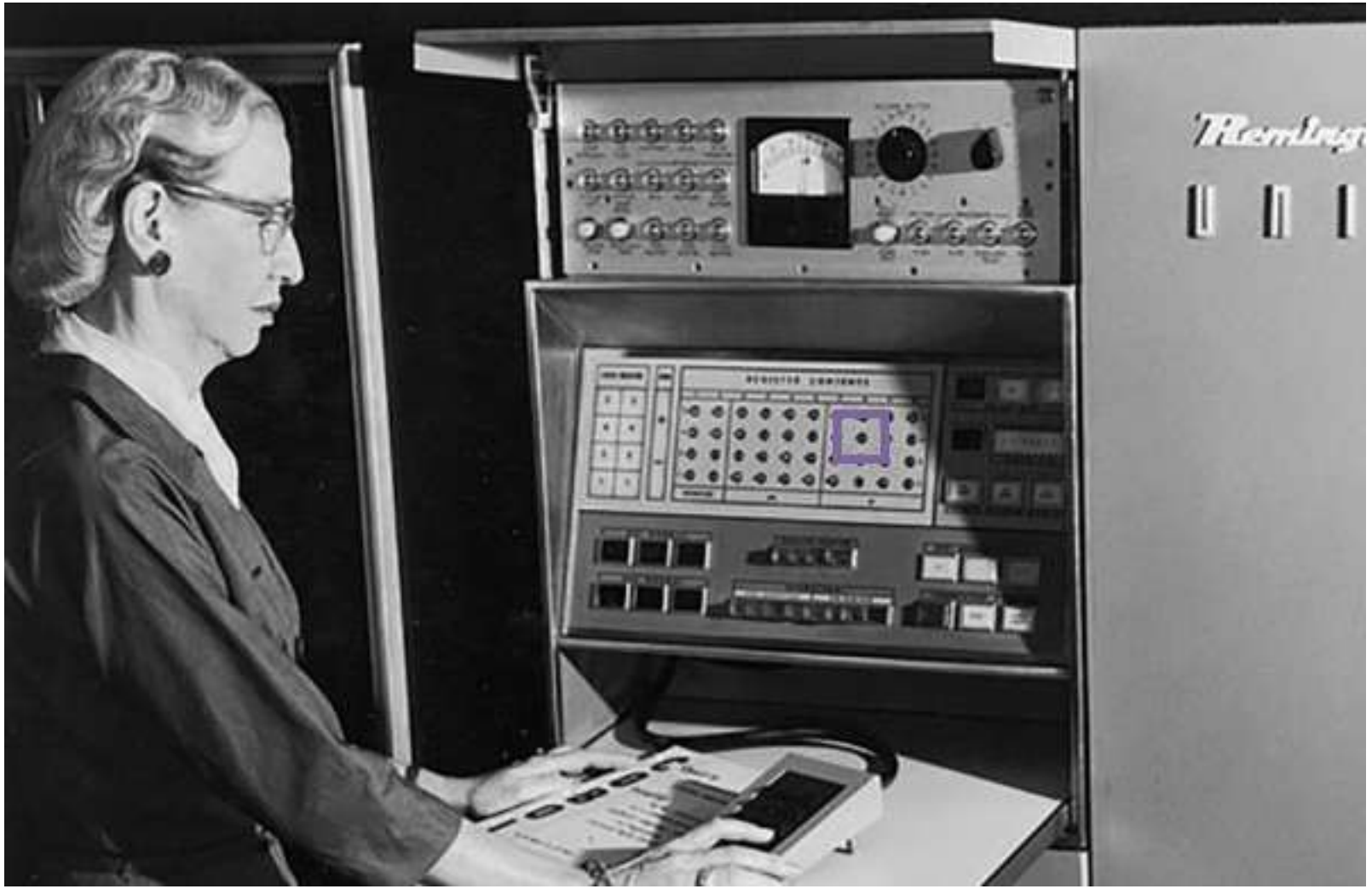
- $E(u,v)$ 在各个 $[u\ v]$ 上的最小值应该尽量大
- 这个最小值取决于H的最小特征值(λ_{min})



对各种情况分类

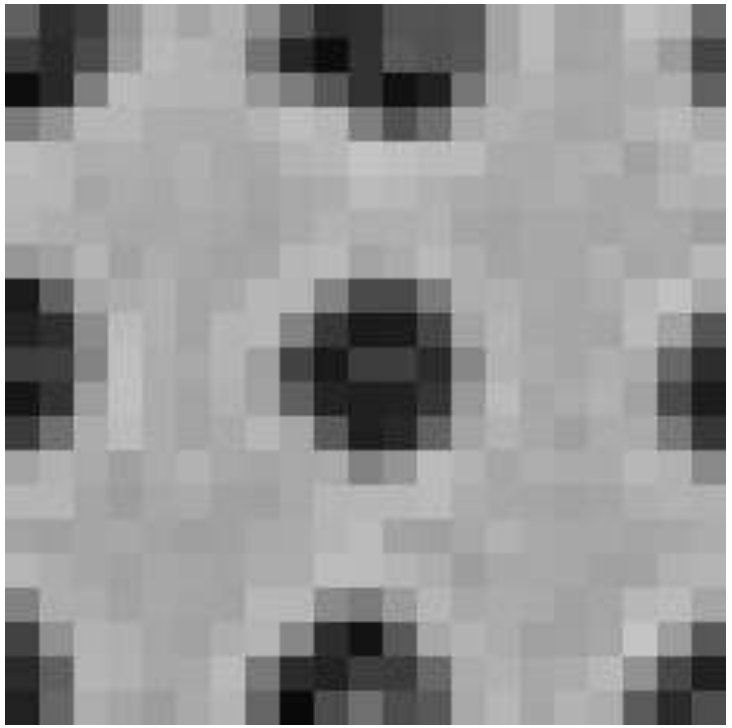

λ_2
“边缘”
$\lambda_2 >> \lambda_1$
“角点”
λ_1 λ_2 都很大,
$\lambda_1 \sim \lambda_2$;
E 各个方向变化很大
λ_1 λ_2 都很小;
E 几乎在所有方向都很小
普通区域
“边缘”
$\lambda_1 >> \lambda_2$
λ_1

Formalizing Corner Detection

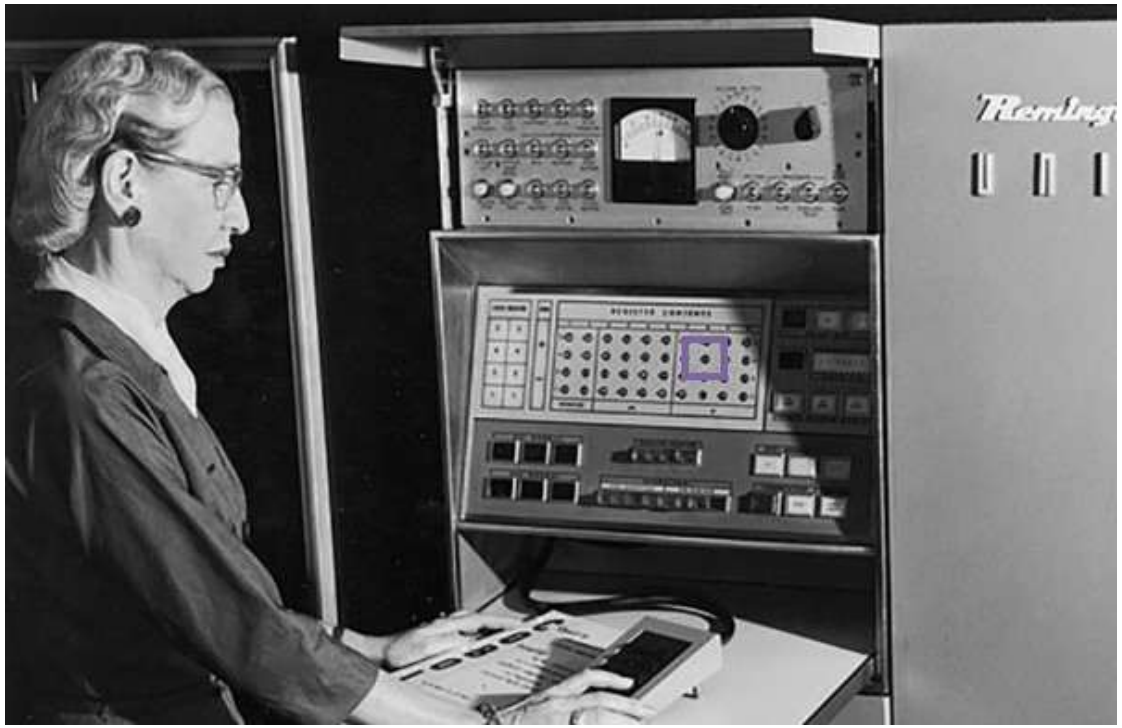


Formalizing Corner Detection

Zoom-In at x,y

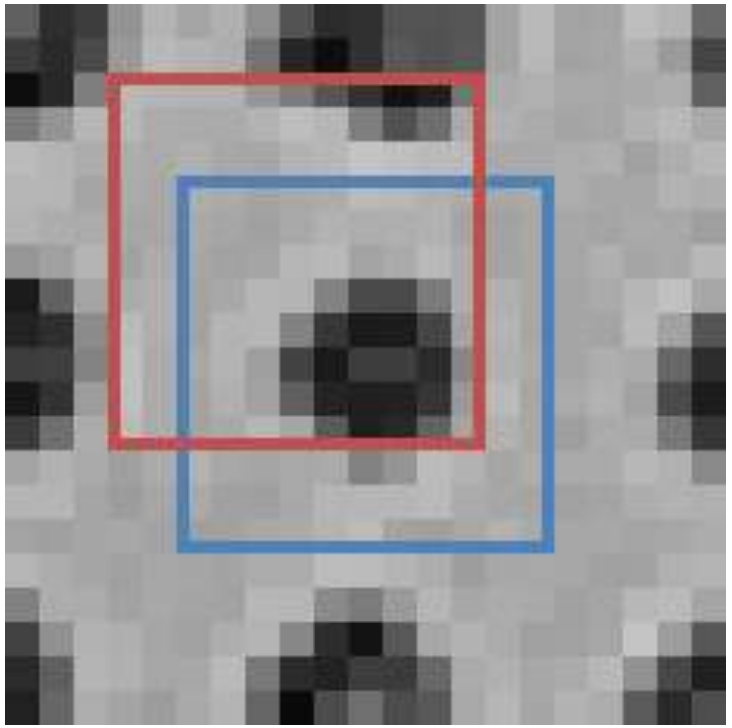


Original Image



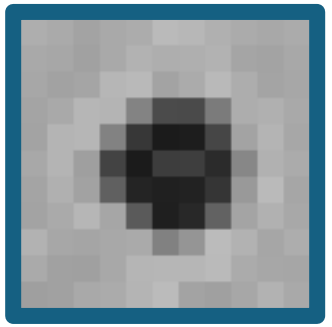
Formalizing Corner Detection

Zoom-In at x,y



Window without and with Offset

“Window”
At x+u, y+v
Here: u=-2,v=-3

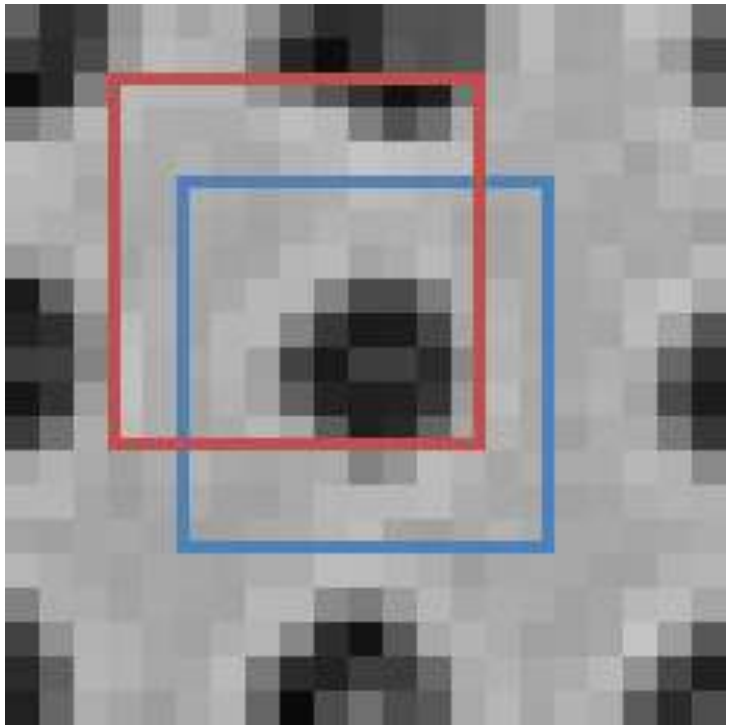


“Window”
At x, y

How might we measure similarity?

Formalizing Corner Detection

Zoom-In at x,y



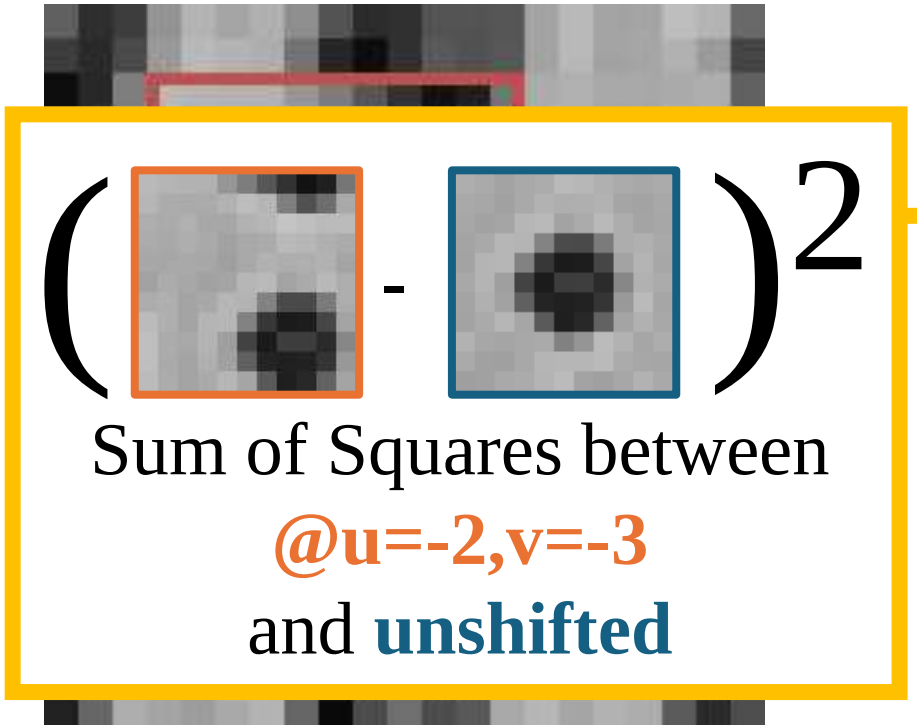
Error (Sum Sqs) for u,v offset

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

$$\left(\quad - \quad \right)^2$$

Formalizing Corner Detection

Zoom-In at x,y



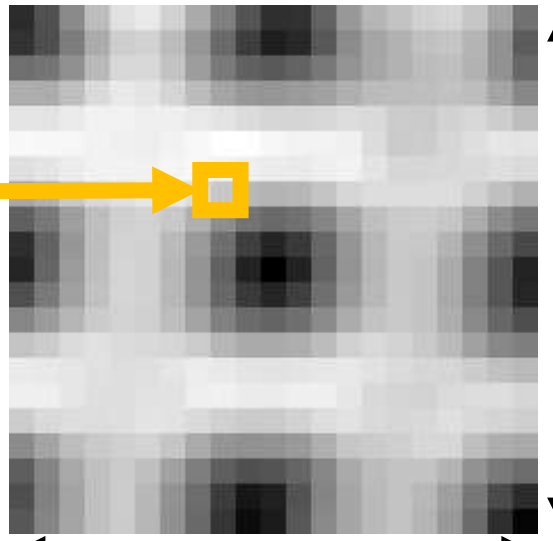
$$\left(\quad - \quad \right)^2$$

Sum of Squares between

@u=-2,v=-3

and **unshifted**

Error (Sum Sqs) for u,v offset



v=-10

v=0

v=10

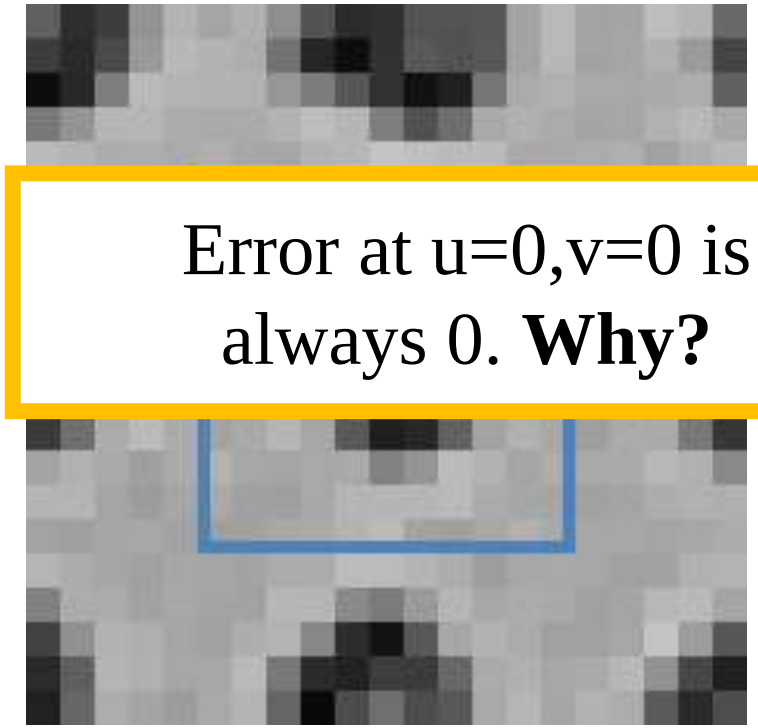
u = -10

u = 0

u = 10

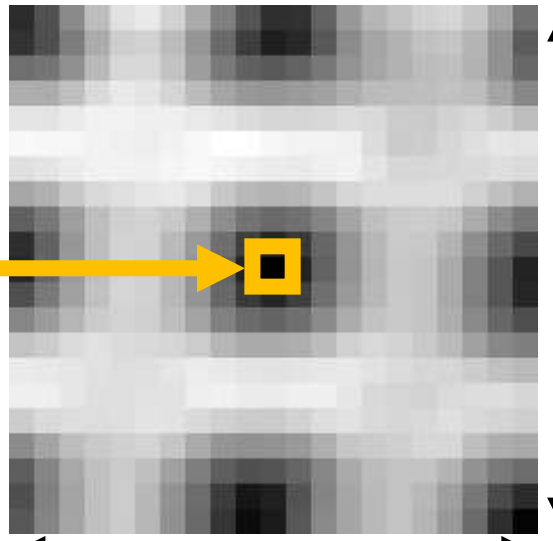
Formalizing Corner Detection

Zoom-In at x,y



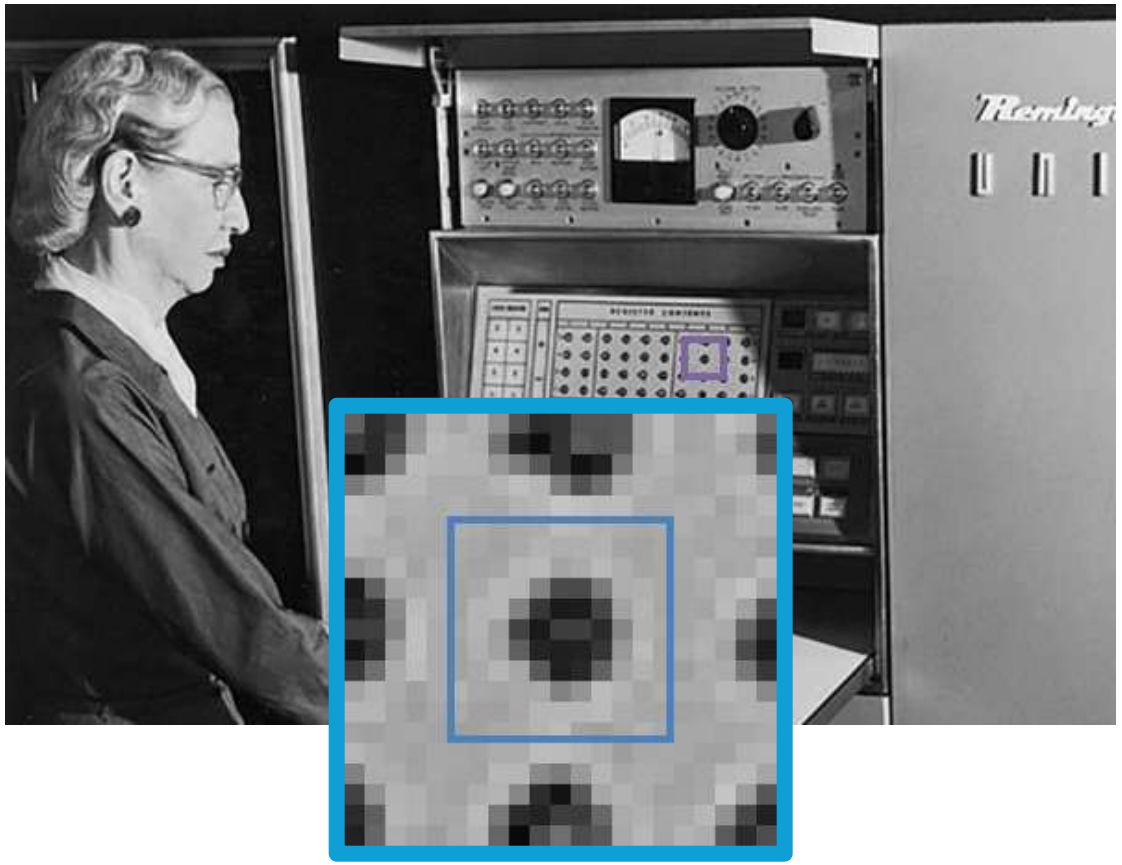
Error at u=0,v=0 is always 0. **Why?**

Error (Sum Sqs) for u,v offset



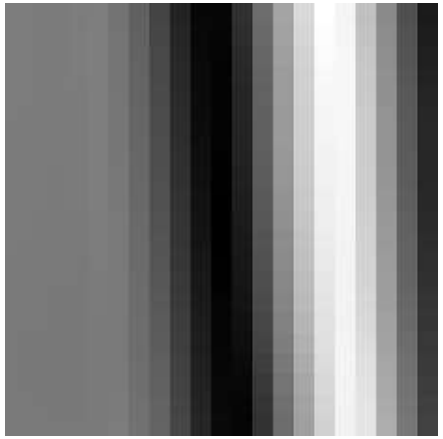
Match The Location and Plot

Original Image and Zoom-In

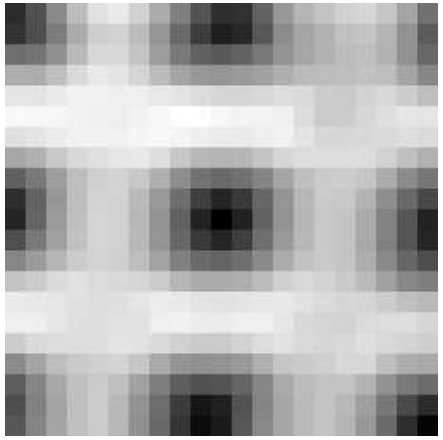


Error Options

A

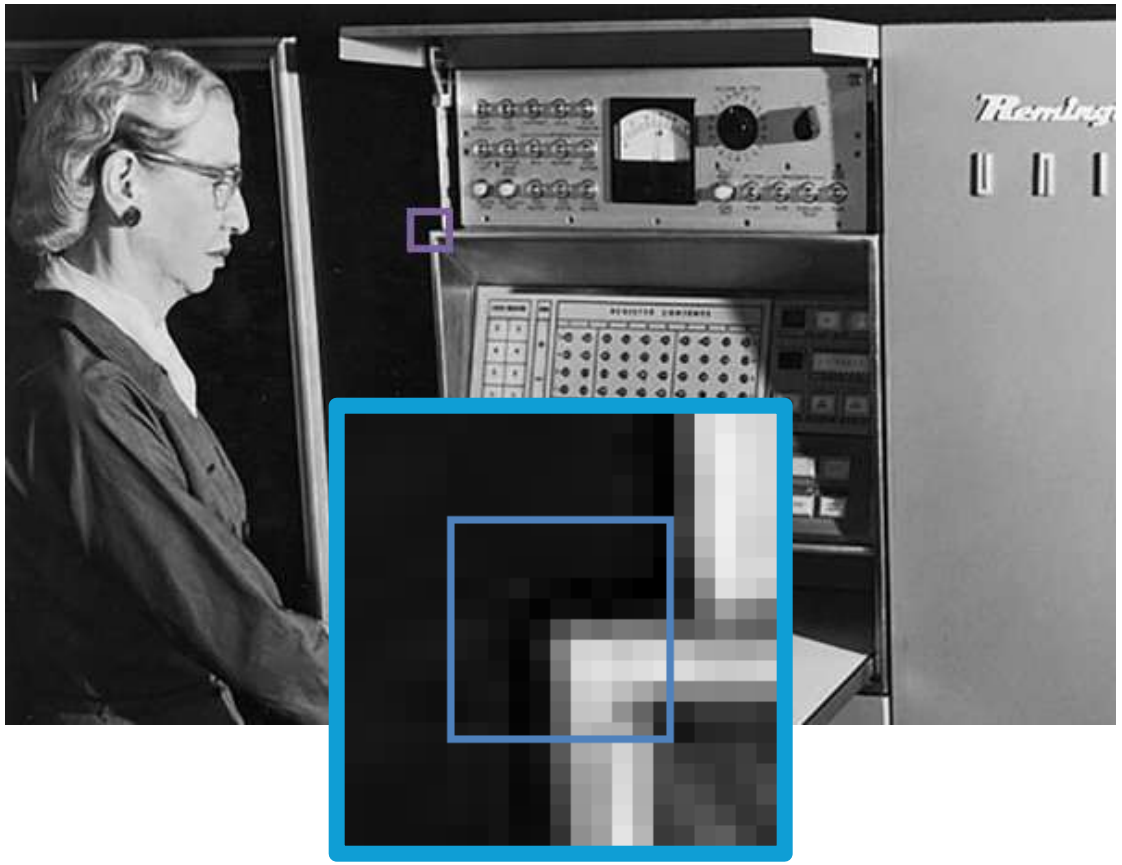


B



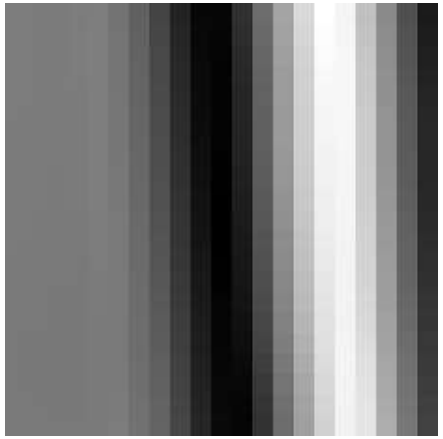
Match The Location and Plot

Original Image and Zoom-In

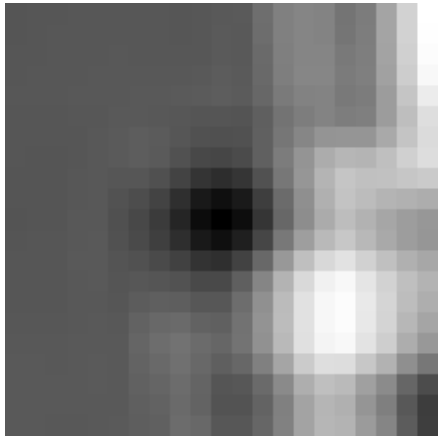


Error Options

A

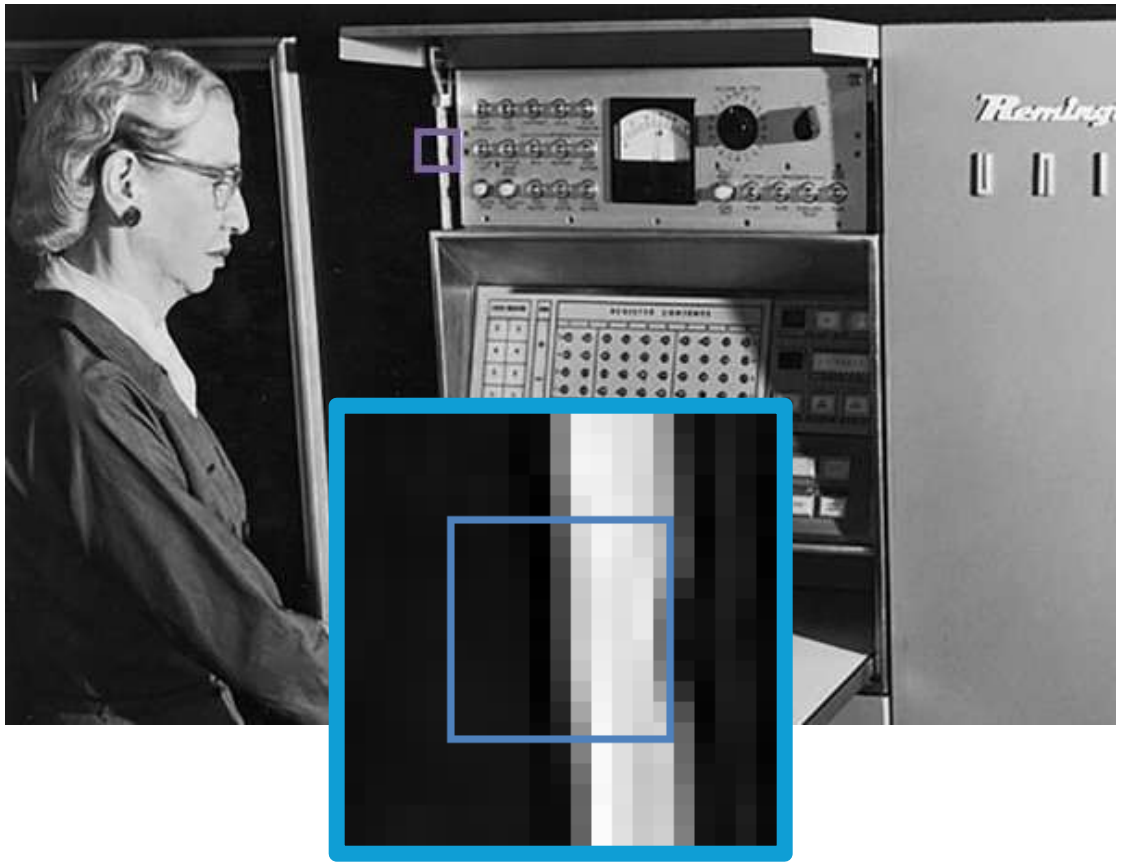


B



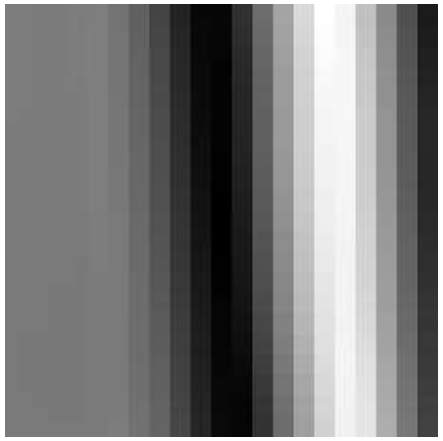
Match The Location and Plot

Original Image and Zoom-In

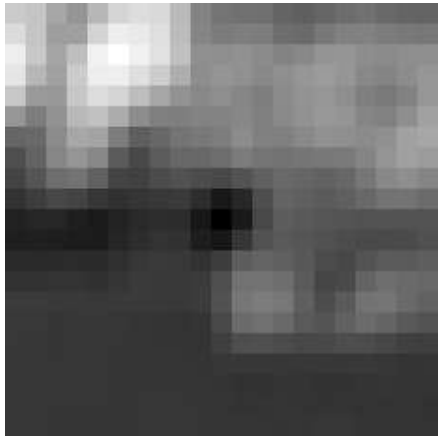


Error Options

A

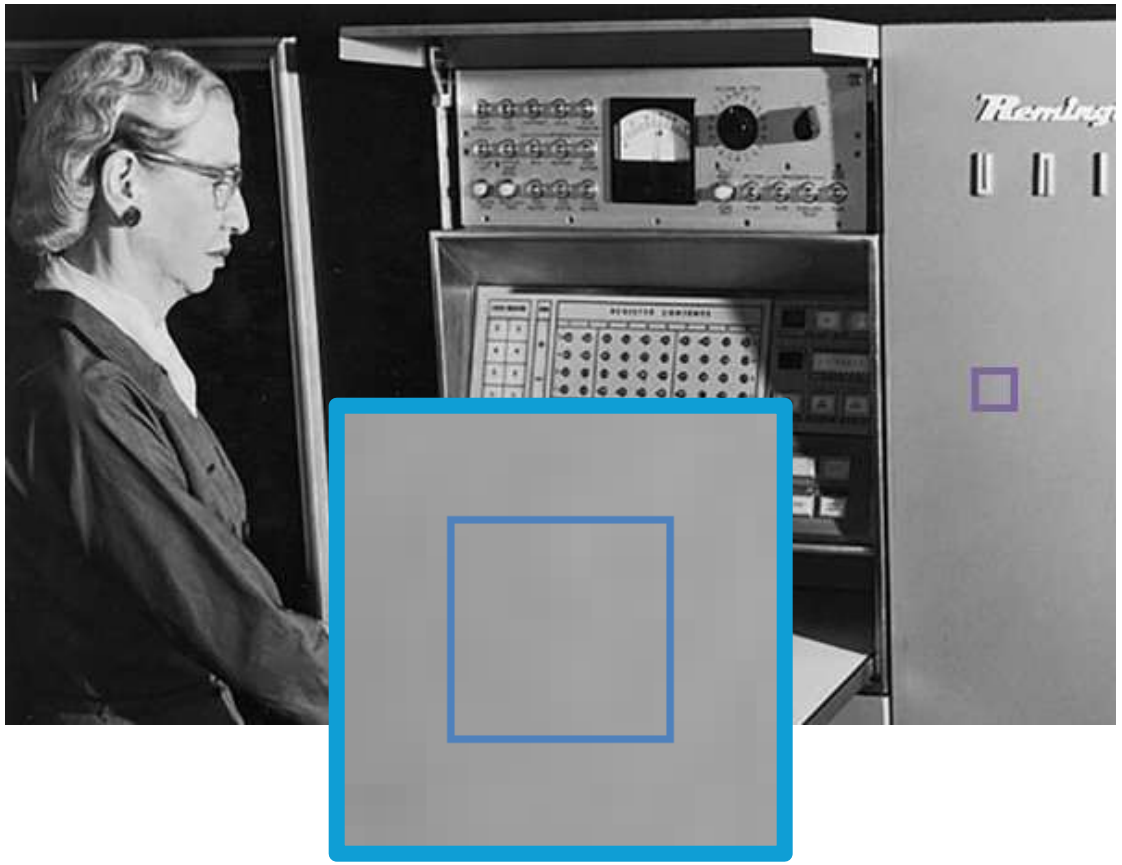


B



Match The Location and Plot

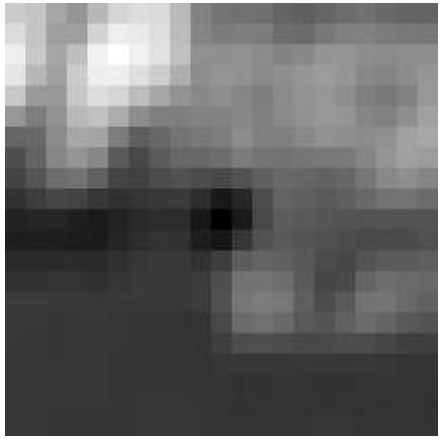
Original Image and Zoom-In



Error Options

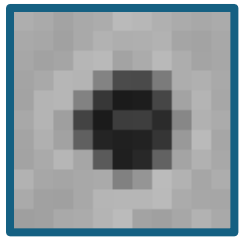
A

B



Ok But Back To Math

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

$$\left(\text{  } - \text{  } \right)^2$$

Shifting windows around is expensive!
We'll find a trick to approximate this.

Note: only need to get the gist

Aside: Taylor Series for Images

Recall Taylor Series – way of *linearizing* a function:

$$f(x+d) \approx f(x) + \frac{\partial f}{\partial x} d$$

Do the same with images, treating them as function of x, y

$$I(x+u, y+v) \approx I(x,y) + I_x u + I_y v$$

For brevity: Ix = Ix at point (x,y), Iy = Iy at point (x,y)

Formalizing Corner Detection

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

Taylor series expansion for I at every single point in window

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

Cancel

$$= \sum_{(x,y)\in W} \left(I_x u + I_y v\right)^2$$

Expand

$$= \sum_{(x,y)\in W} I_x^2 u^2 + 2 I_x I_y uv + I_y^2 v^2$$

For brevity: Ix = Ix at point (x,y), Iy = Iy at point (x,y)

Formalizing Corner Detection

By linearizing image, we can approximate E(u,v) with quadratic function of u and v

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

$$= [u,v]\boldsymbol{M}[u,v]^T$$

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y\in W} I_x^2 & \sum_{x,y\in W} I_x I_y \\ \sum_{x,y\in W} I_x I_y & \sum_{x,y\in W} I_y^2 \end{bmatrix}$$

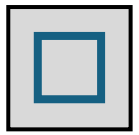
M is called the second moment matrix

Intuitively what is M?

Pretend gradients are *either* vertical or horizontal at a pixel (so Ix Iy = 0)

Obviously Wrong!

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} \approx \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

a,b both small: flat  $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$

One big, other small: edge  $\begin{bmatrix} 50 & 0 \\ 0 & 0.1 \end{bmatrix}$ or $\begin{bmatrix} 0.1 & 0 \\ 0 & 50 \end{bmatrix}$

a,b both big: corner  $\begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$

Intuitively what is M?

Pretend gradients are *either* vertical or horizontal at a pixel (so Ix Iy = 0)

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} \approx ? \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

a,b both small: flat

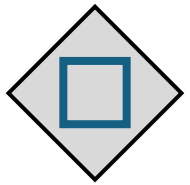
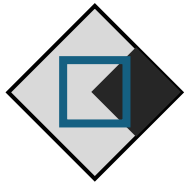


Image might be rotated by rotation θ!

One big, other small: edge

a,b both big: corner

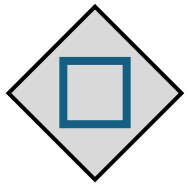


Intuitively what is M?

Pretend gradients are *either* vertical or horizontal at a pixel (so Ix Iy = 0)

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \boldsymbol{V^{-1}} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \boldsymbol{V}$$

a,b both small: flat



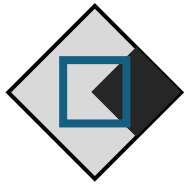
If image rotated by rotation θ / matrix **V**

One big, other small: edge

M will look like

$$\boldsymbol{V^{-1}} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \boldsymbol{V}$$

a,b both big: corner



So What Now?

Can calculate M at pixel, by summing nearby gradients, but need access to a and b.

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \boldsymbol{V^{-1}} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \boldsymbol{V}$$

Given **M**, can decompose it into eigenvectors **V** and eigenvalues λ_1, λ_2 with $\mathbf{M} = \boldsymbol{V^{-1}} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \boldsymbol{V}$.

Really slow. Why?

So What Now?

Can calculate M at pixel, by summing nearby gradients, but need access to a and b.

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \boldsymbol{V^{-1}} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \boldsymbol{V}$$

Instead: compute quantity R from **M**

$$R = \det(\boldsymbol{M}) - \alpha \operatorname{trace}(\boldsymbol{M})^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

Easy fast formula for 2x2

Fast – sum the diagonal

Empirical value, usually 0.04-0.06

So What Now?

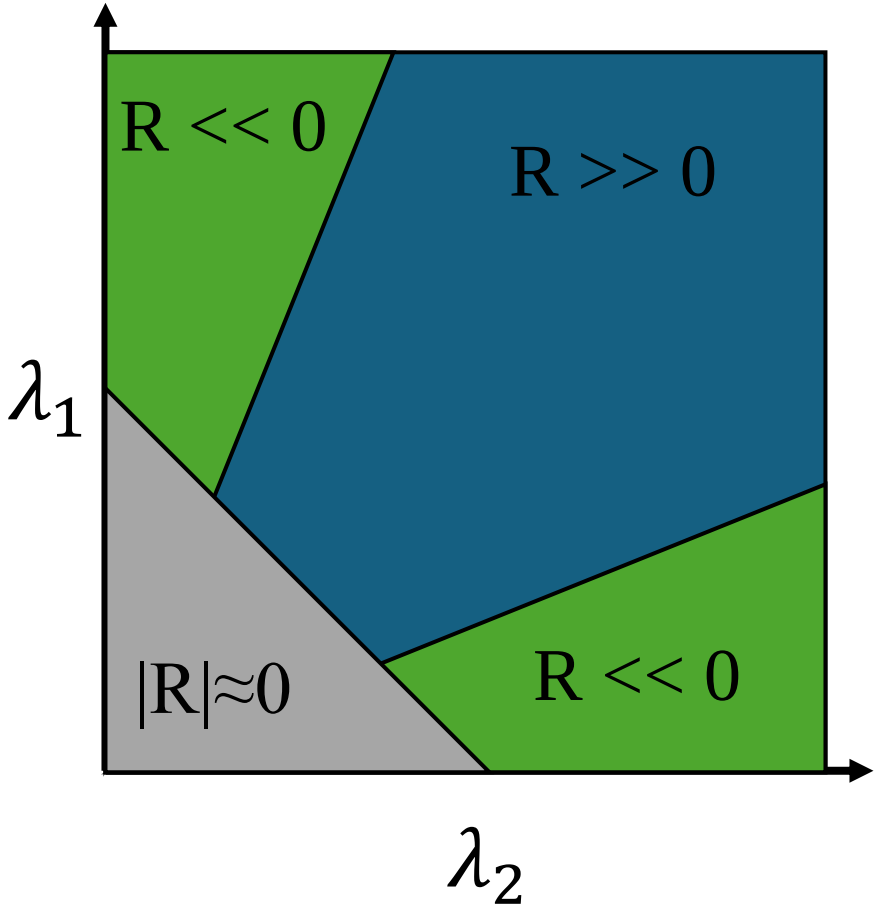
R tells us whether we're at a corner, edge, or flat

$$R = \det(\boldsymbol{M}) - \alpha \operatorname{trace}(\boldsymbol{M})^2 = \lambda_1\lambda_2 - \alpha\,(\lambda_1 + \lambda_2)^2$$

flat — $\lambda_1, \lambda_2 \approx 0$

edge — $\lambda_1 \gg \lambda_2 \gg 0$, $\lambda_2 \gg \lambda_1 \gg 0$

corner — $\lambda_1 \approx \lambda_2 \gg 0$



Remake of standard diagram from S. Lazebnik from original Harris paper.

What Do I Need To Know?

- Need to be able to take derivatives of image
- Need to be able to compute the entries of **M** at every pixel.
- Should know that some properties of **M** indicate whether a pixel is a corner or not.

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix}$$

In Practice

1. Compute partial derivatives Ix, Iy per pixel
2. Compute **M** at each pixel, using Gaussian weighting w

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

C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

Slide credit: S. Lazebnik

In Practice

1. Compute partial derivatives Ix, Iy per pixel
2. Compute **M** at each pixel, using Gaussian weighting w
3. Compute response function R

$$R = \det(\boldsymbol{M}) - \alpha\, trace(\boldsymbol{M})^2$$
$$= \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

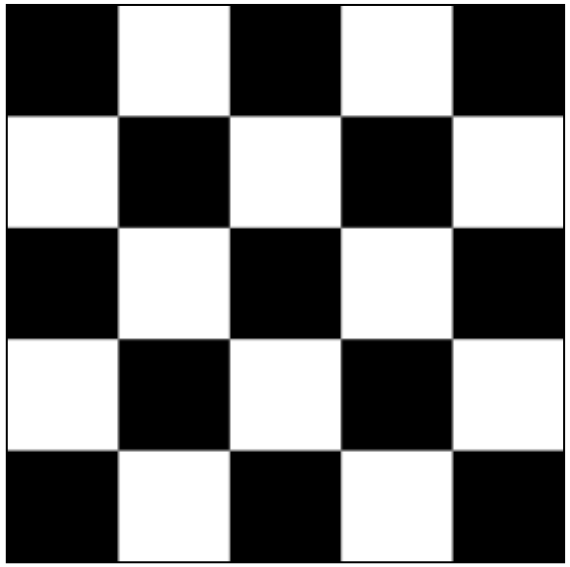
Slide credit: S. Lazebnik

Harris 角点检测

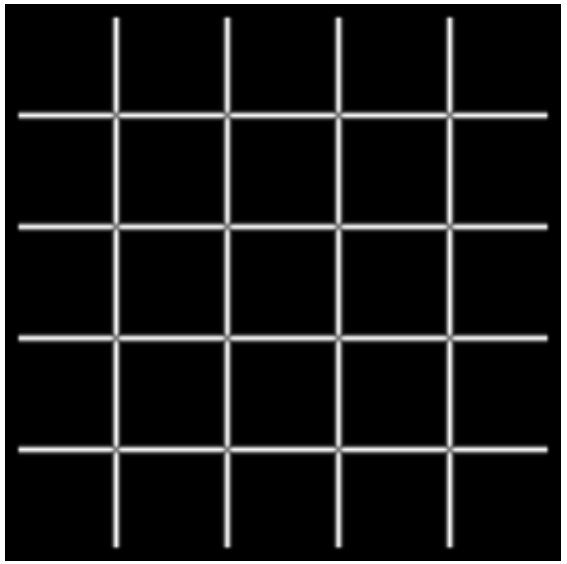
步骤:

- 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

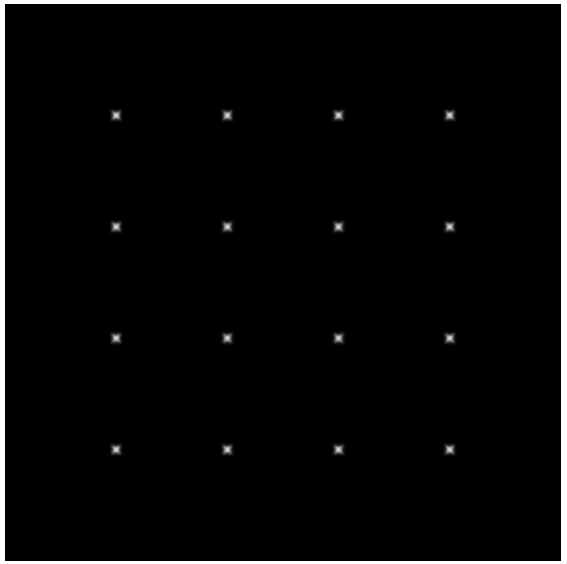
$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



I



$\lambda_{\max}$

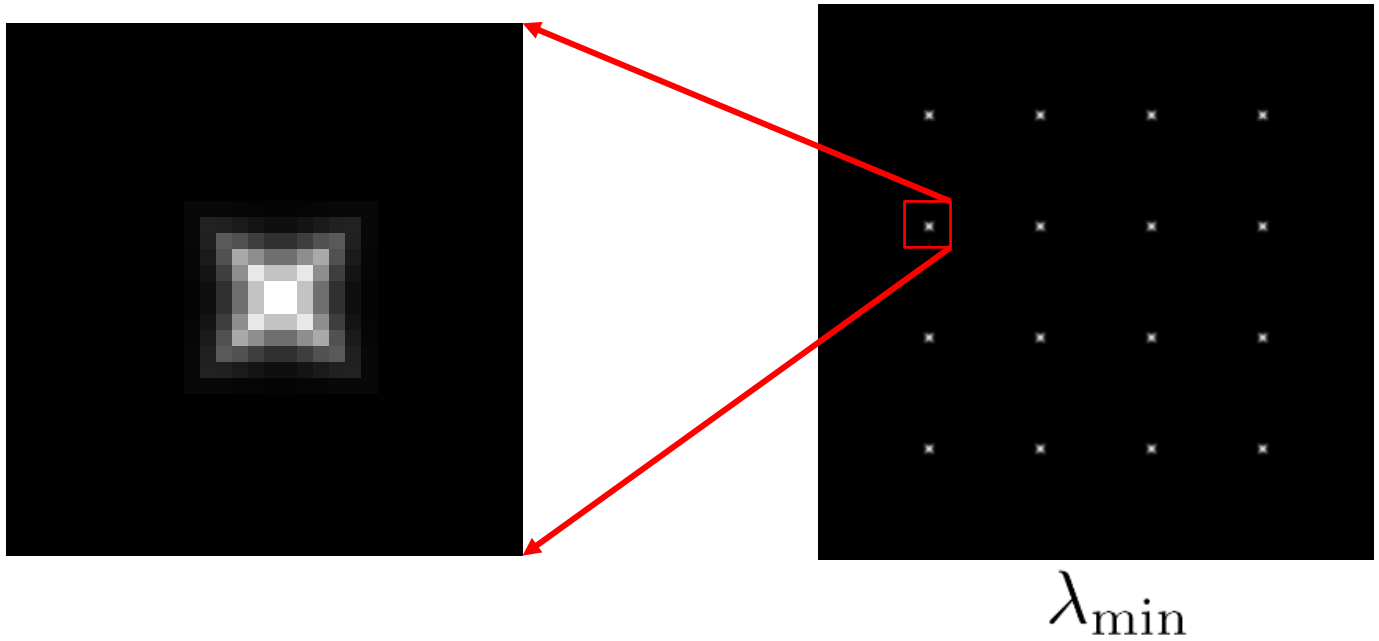


$\lambda_{\min}$

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}

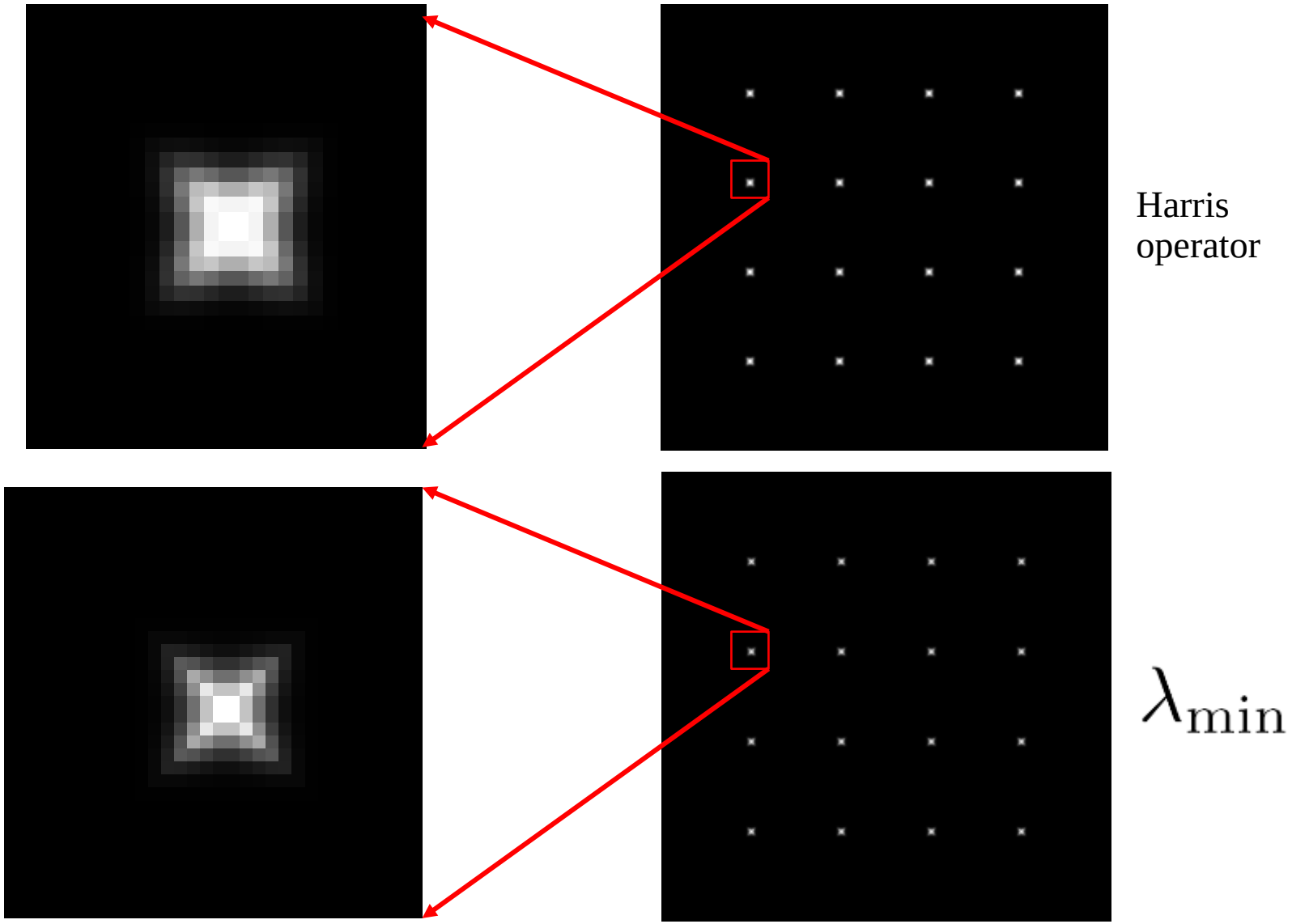
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

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

- 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



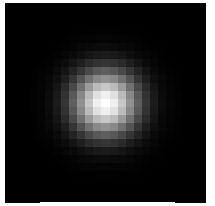
对图像梯度加权

- 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？

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

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



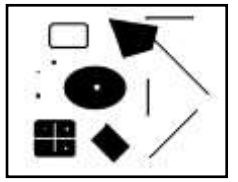
$w_{x,y}$

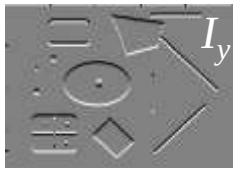
Harris Detector [Harris88]

- 构造二阶矩阵

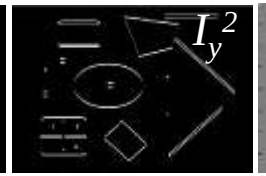
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. 图像梯度





2. 图像梯度的平方

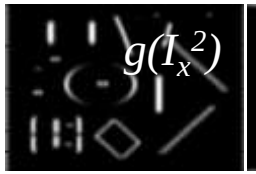


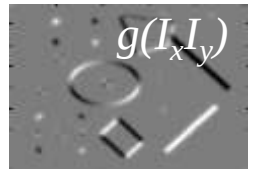
$$\det M = \lambda_1 \lambda_2$$

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

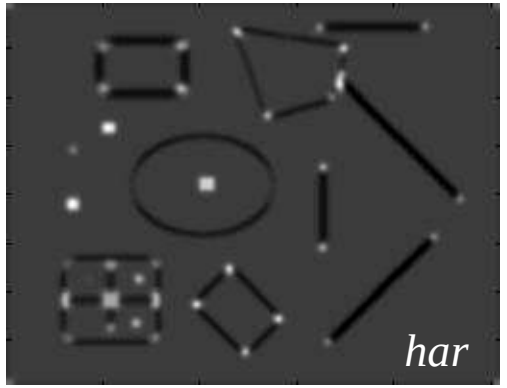
3. 高斯滤波

$g(s_I)$





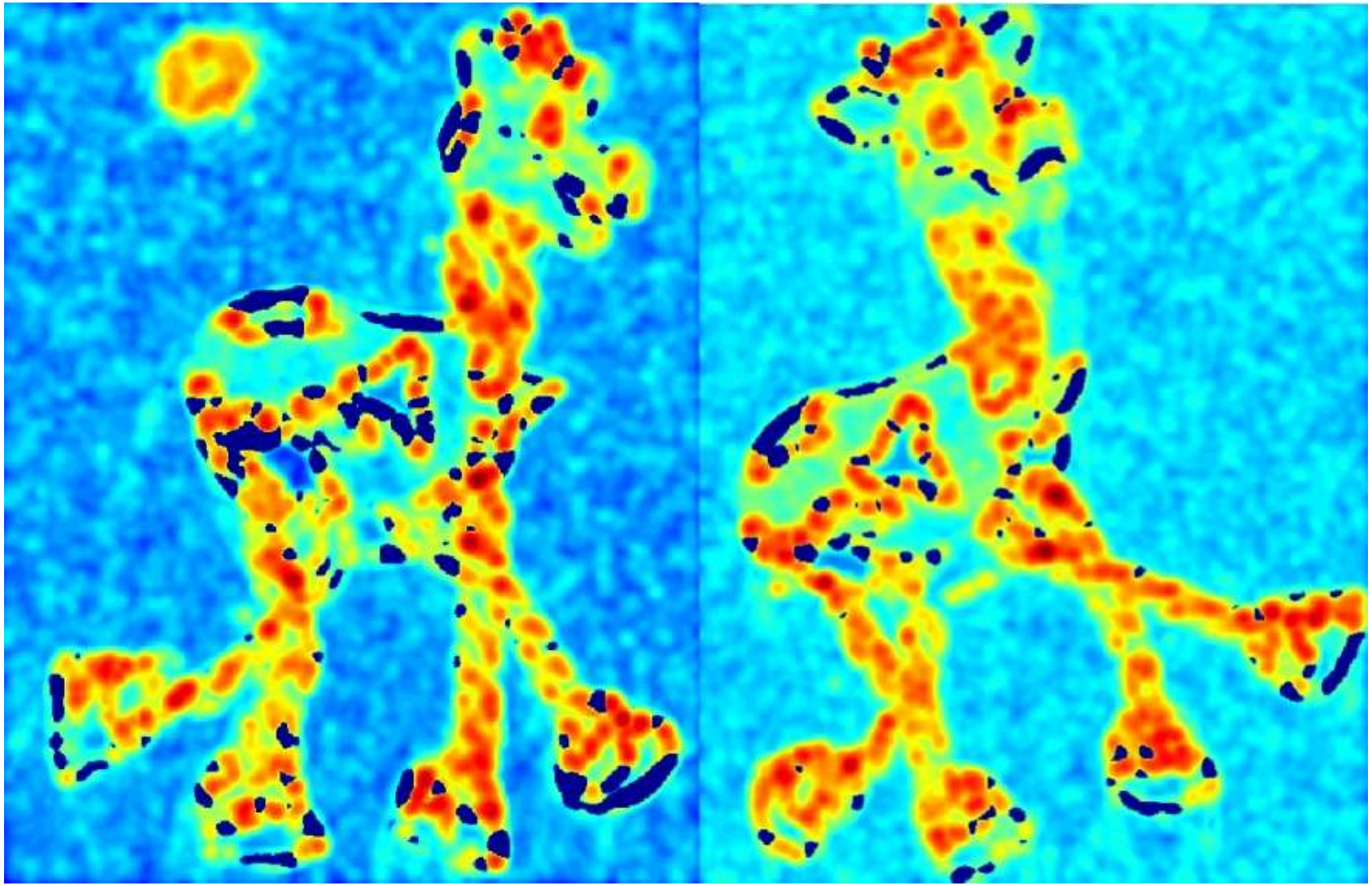
4. 找到最大特征值和最小特征值都足够大的点，或使用Harris算子



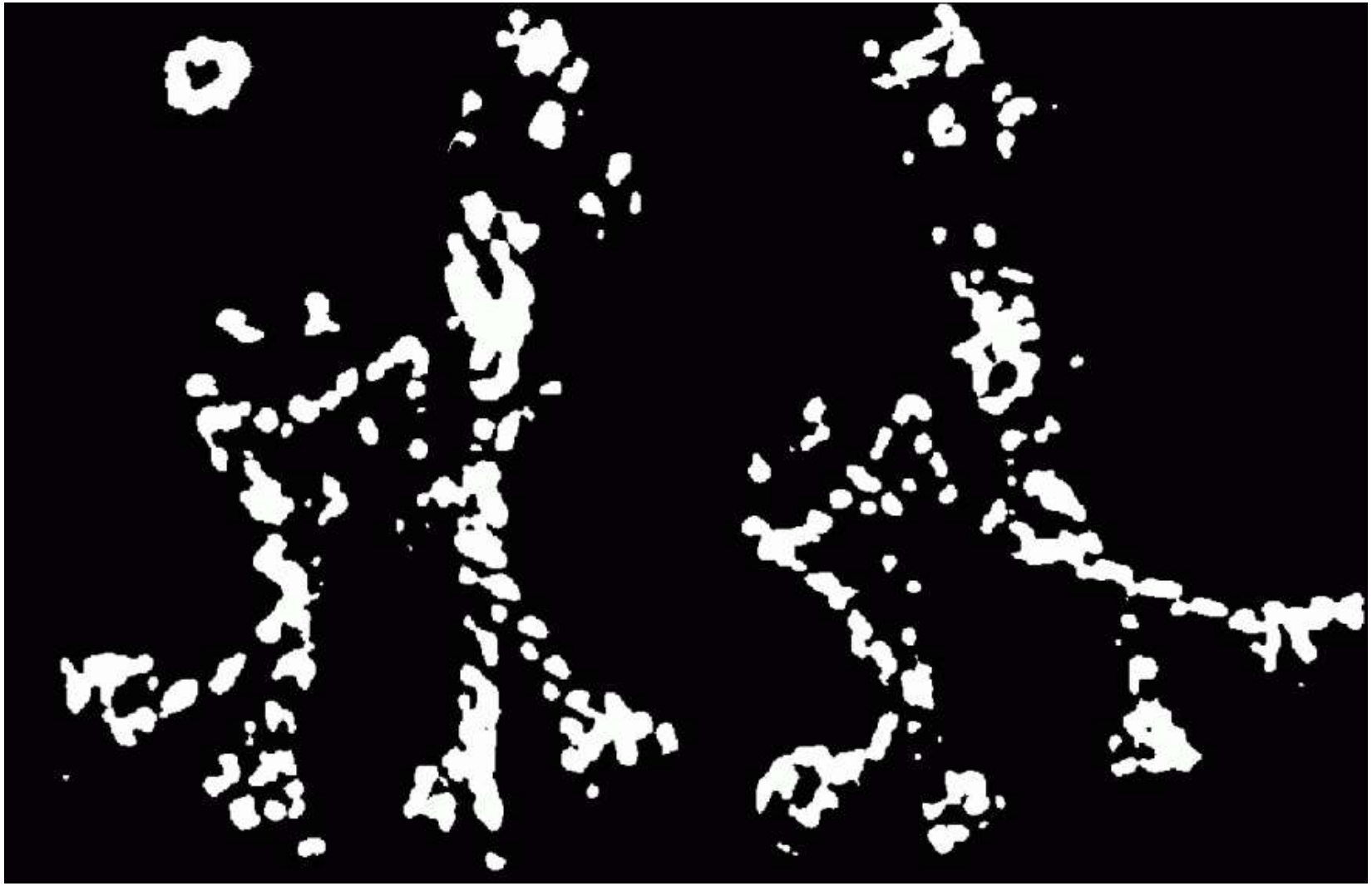
5. Non-maxima suppression 非局部最大抑制

Harris 角点检测

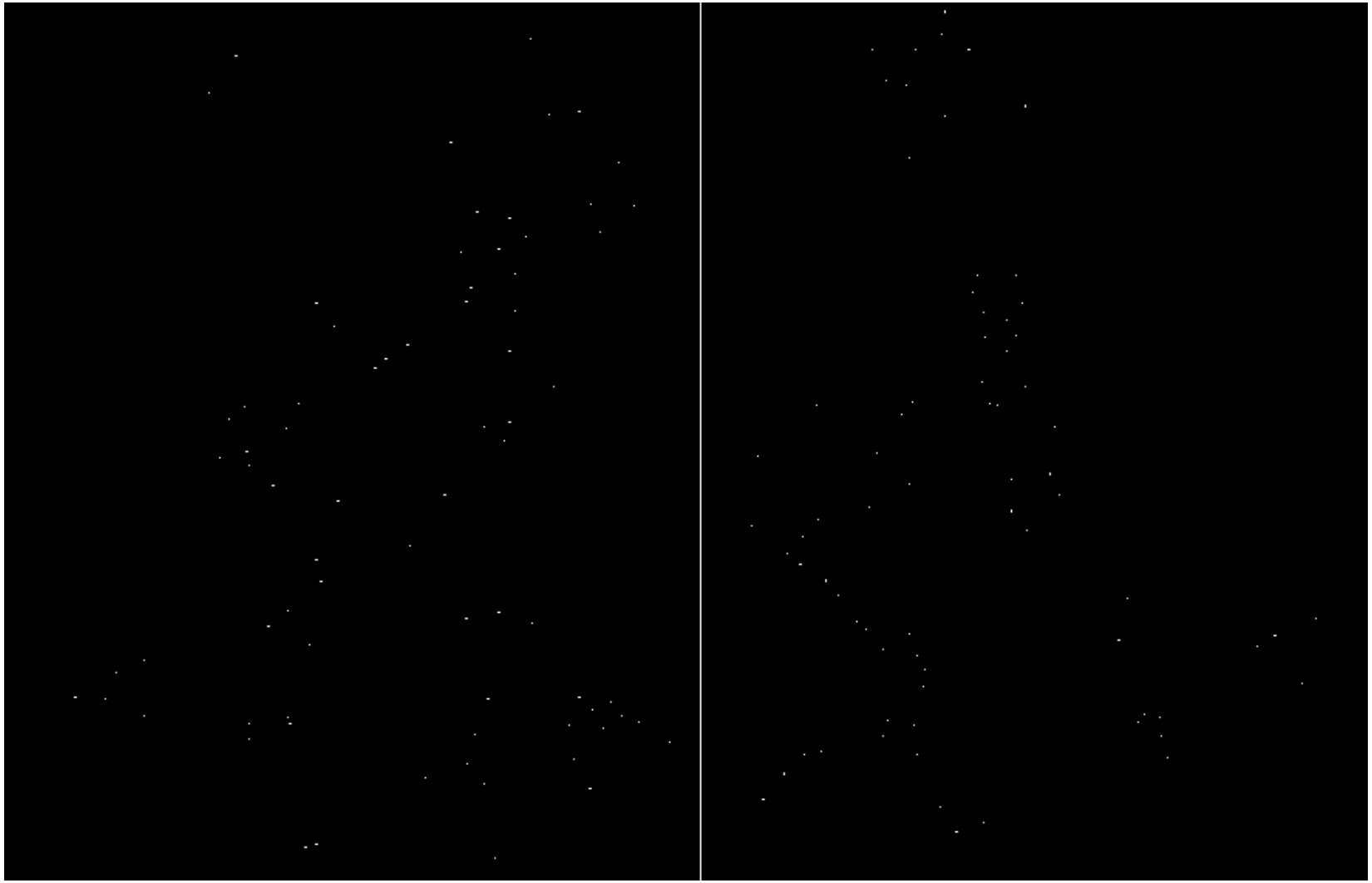
f value (red high, blue low)



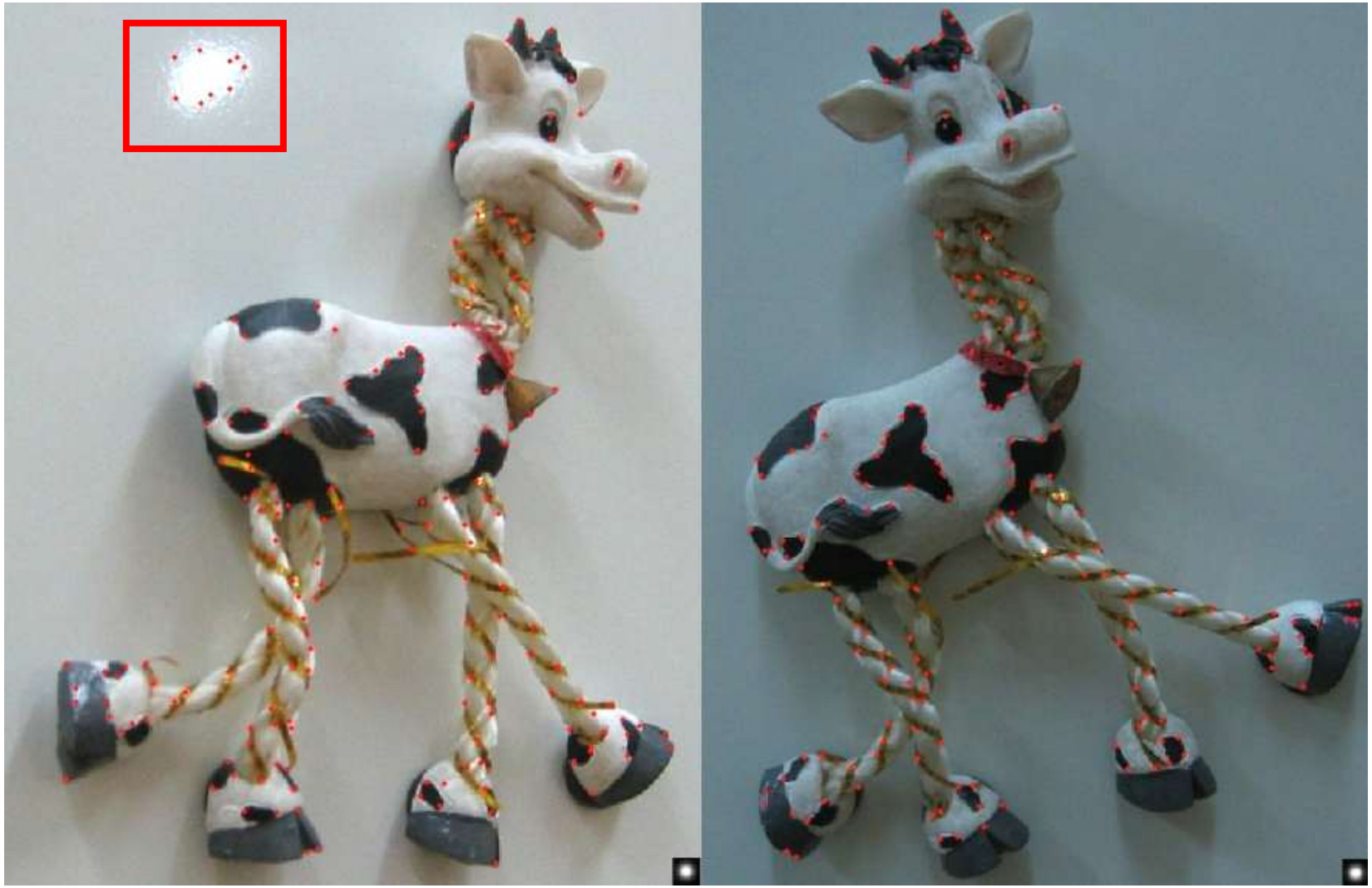
阈值处理 (f > value)



f 局部最大値 (non-max suppression)



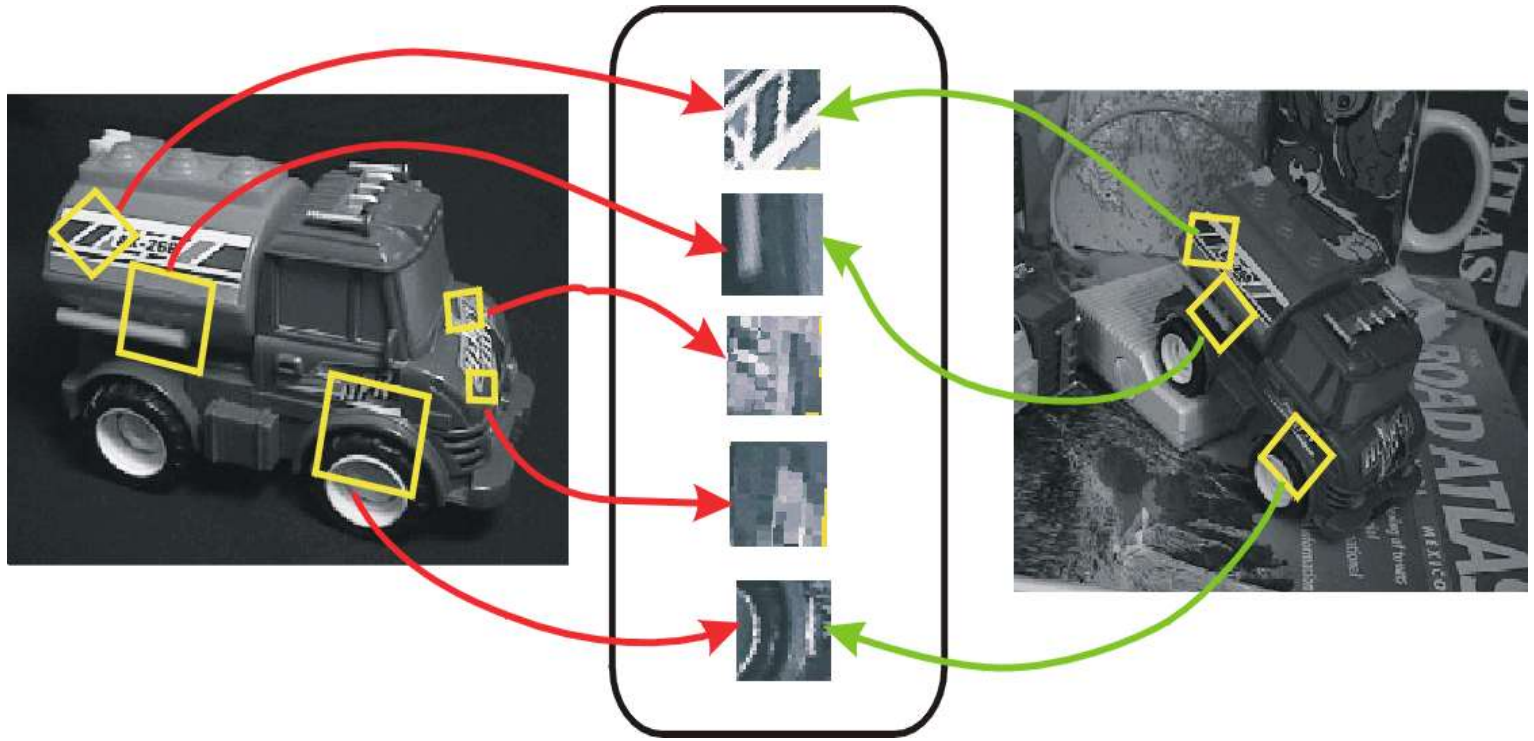
哈里斯特征点(in red): 我们还需要做点什么？



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

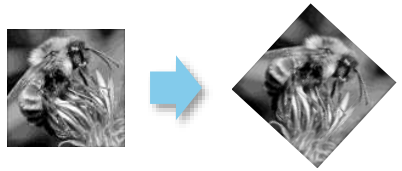
图像变换

找到在图像变换下仍然保持不变的图像特征

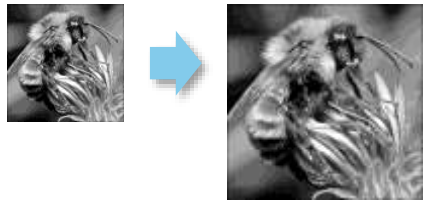
- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度, …

- 几何层面

旋转

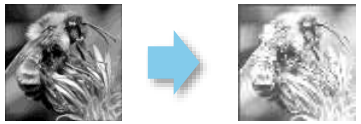


尺度



- 光学层面

强度变换

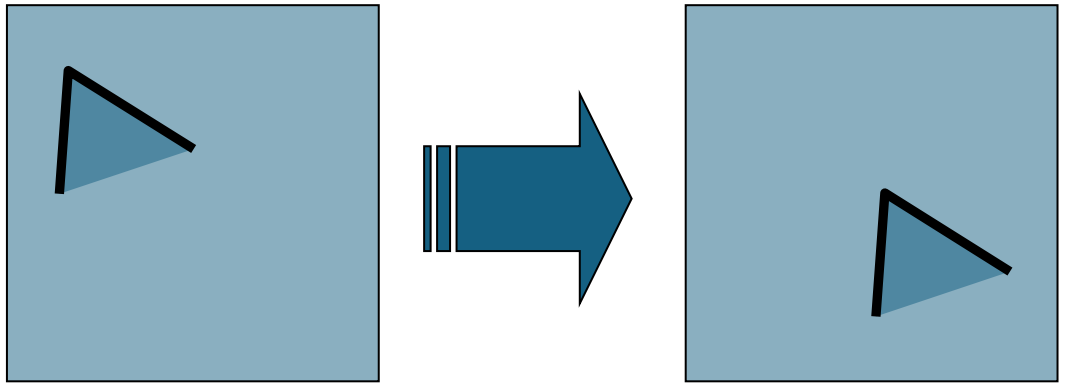


关键点要在这些图像变换中保持不变！

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

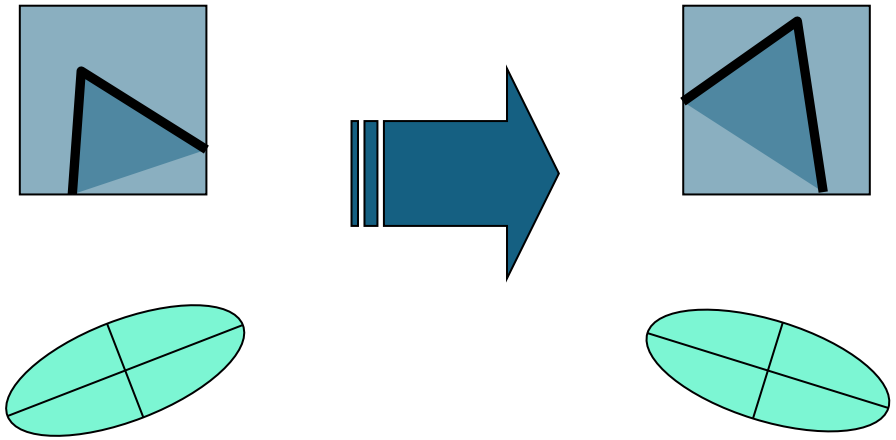
平移不变



- 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

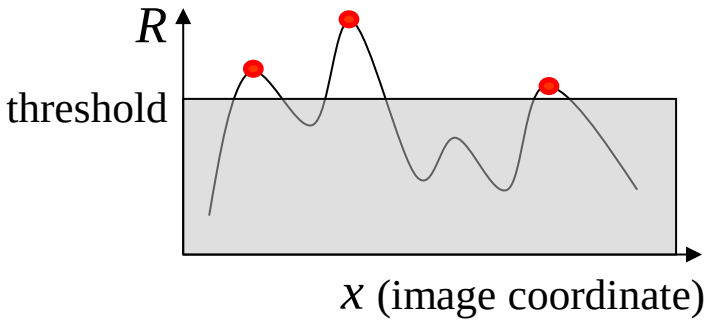
特征点位置在变换后等价

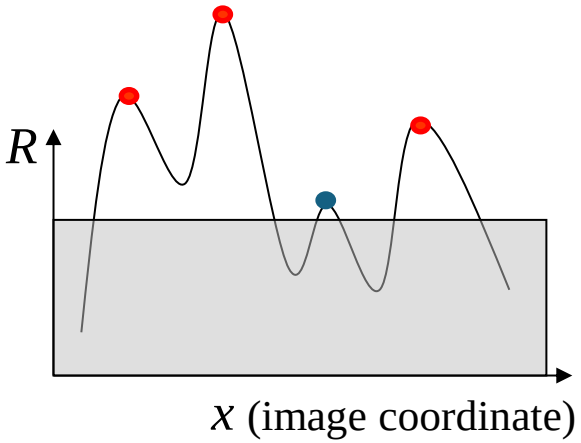
强度不变

$I \rightarrow a\,I + b$

•图像梯度→ 强度差 $I \rightarrow I + b$

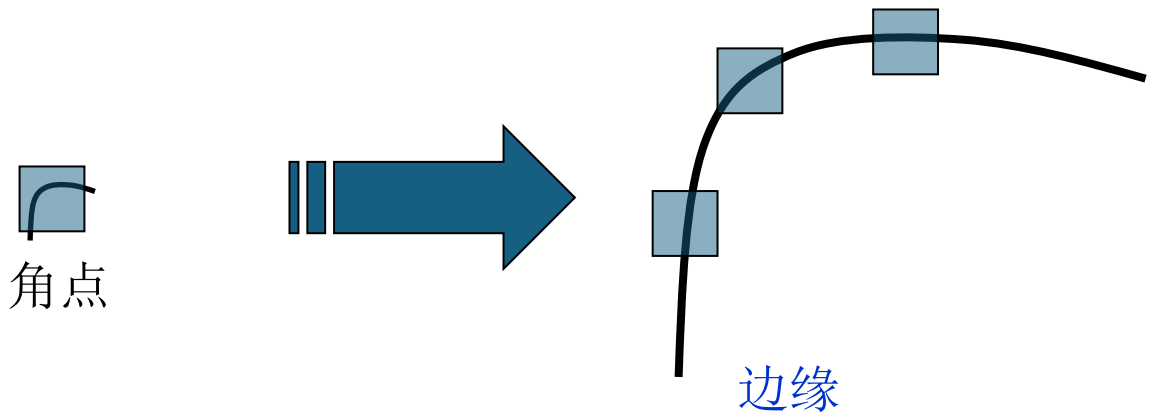
•强度缩放: $I \rightarrow a\,I$





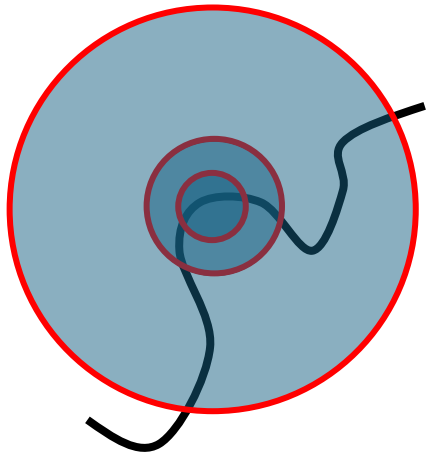
采取阈值法，对于强度变化部分不变

尺度（缩放、强度）不变



如何确保对缩放的不变性？

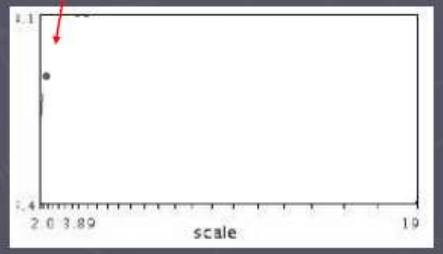
尺度不变性



核心思想：找到响应最大的缩放比例

Automatic scale selection

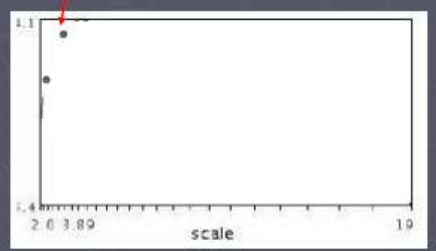
Lindeberg et al., 1996



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

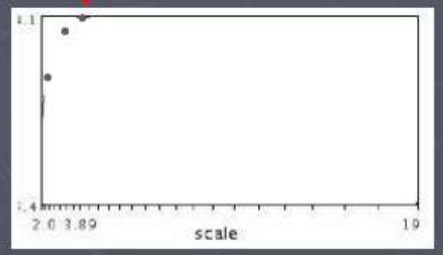
Slide from Tinne Tuytelaars

Automatic scale selection



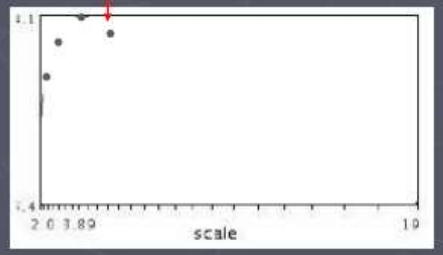
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



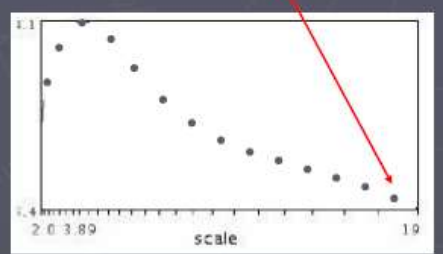
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



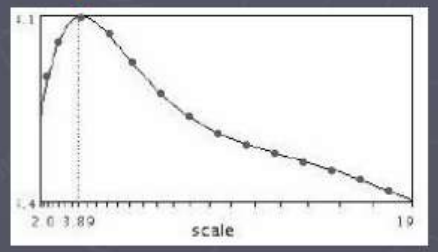
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



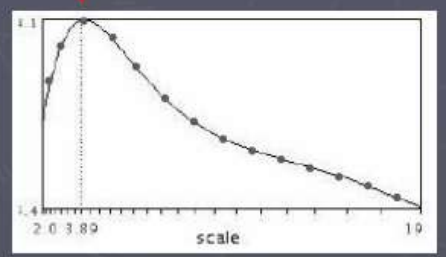
$f(I_{i\_j\_k}(x,\sigma))$

Automatic scale selection

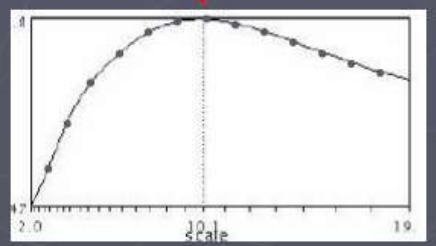


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection



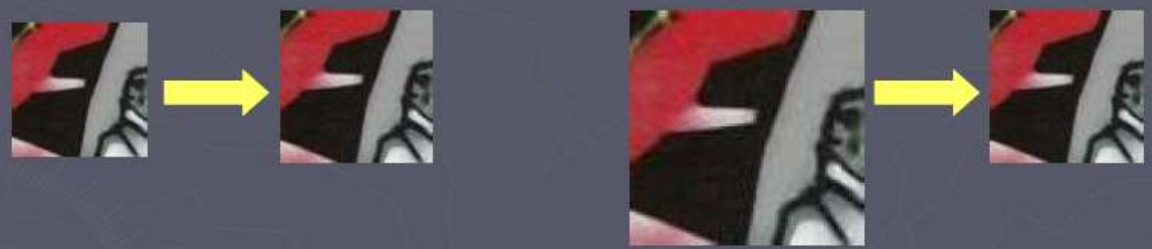
$f(I_{i_1 \ldots i_m}(x,\sigma))$



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

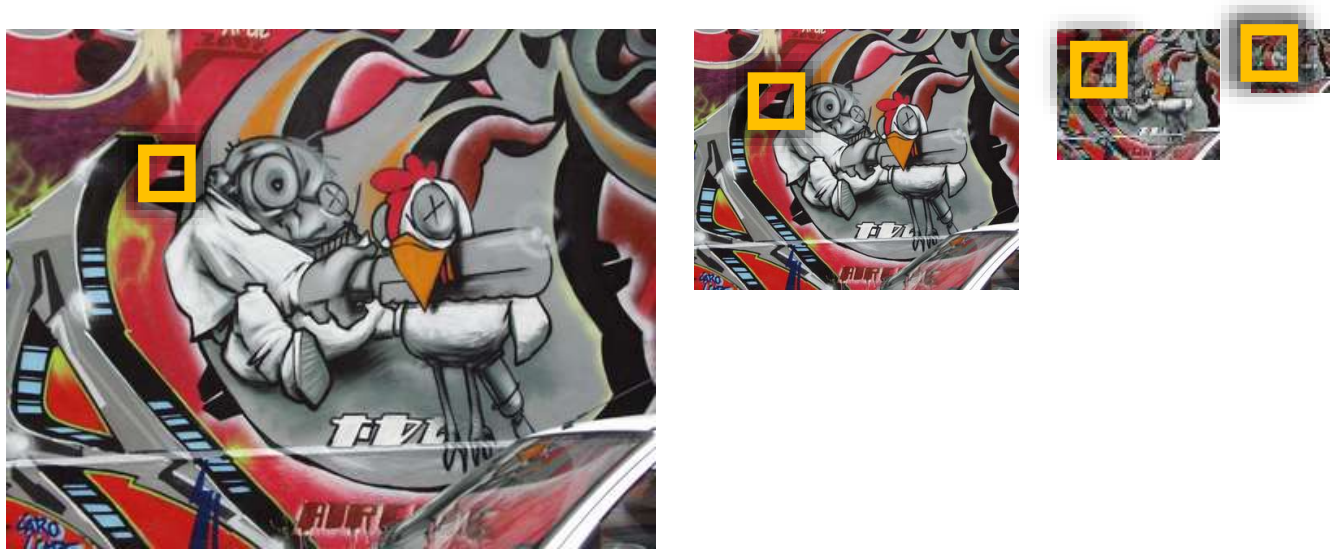
Automatic scale selection

Normalize: rescale to fixed size



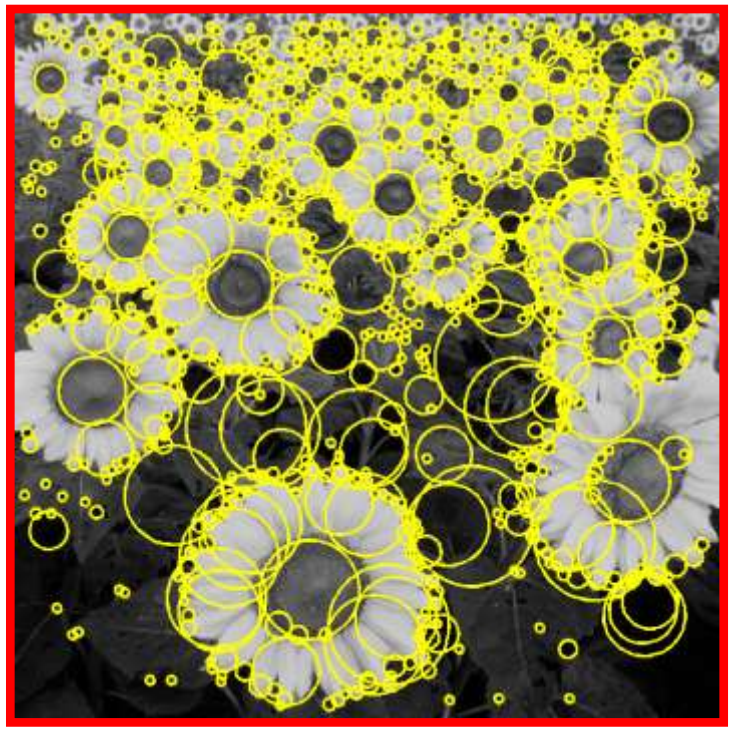
实现

- 用同样的窗口大小，但缩放图像
- **Gaussian Pyramid**



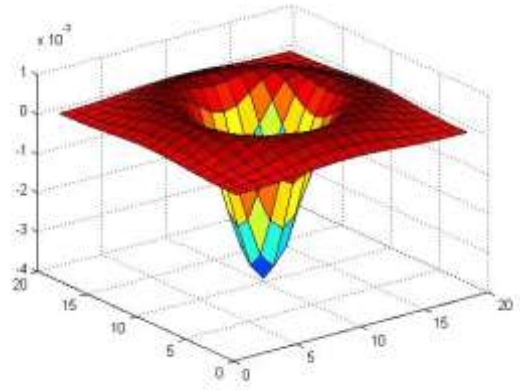
特征提取：斑点

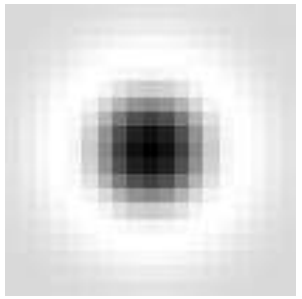
- 斑点（Blobs）：斑点是图像中的连续区域，其像素值具有明显的局部极值。



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

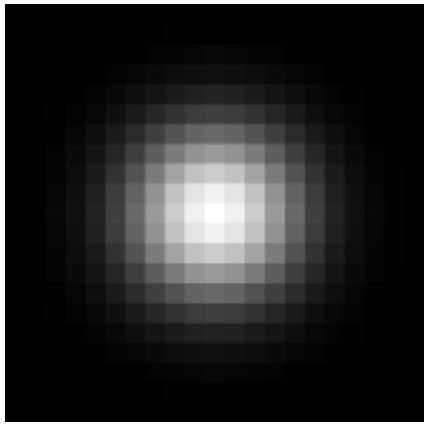
LoG 对旋转保证不变性

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

一个区域内高响应，周围低响应

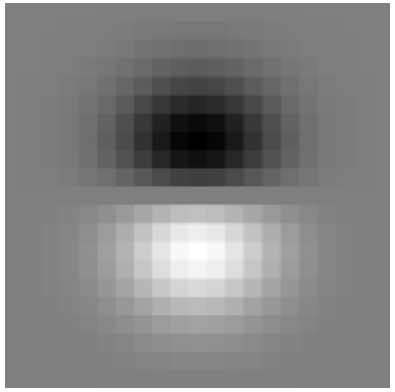
Gaussian Derivatives

Gaussian

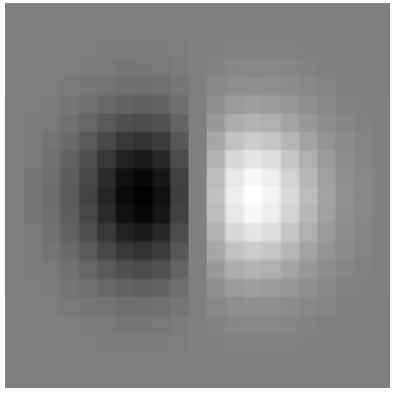


1st Deriv

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

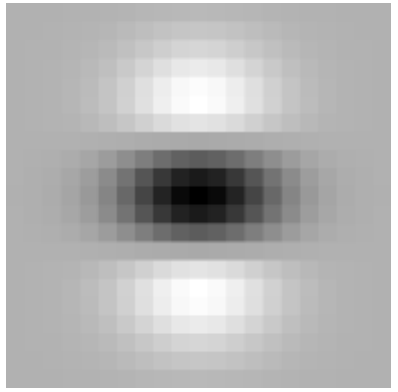


$\frac{\partial}{\partial x} g$

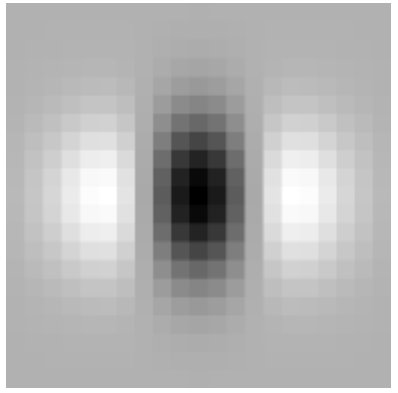


2nd Deriv

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

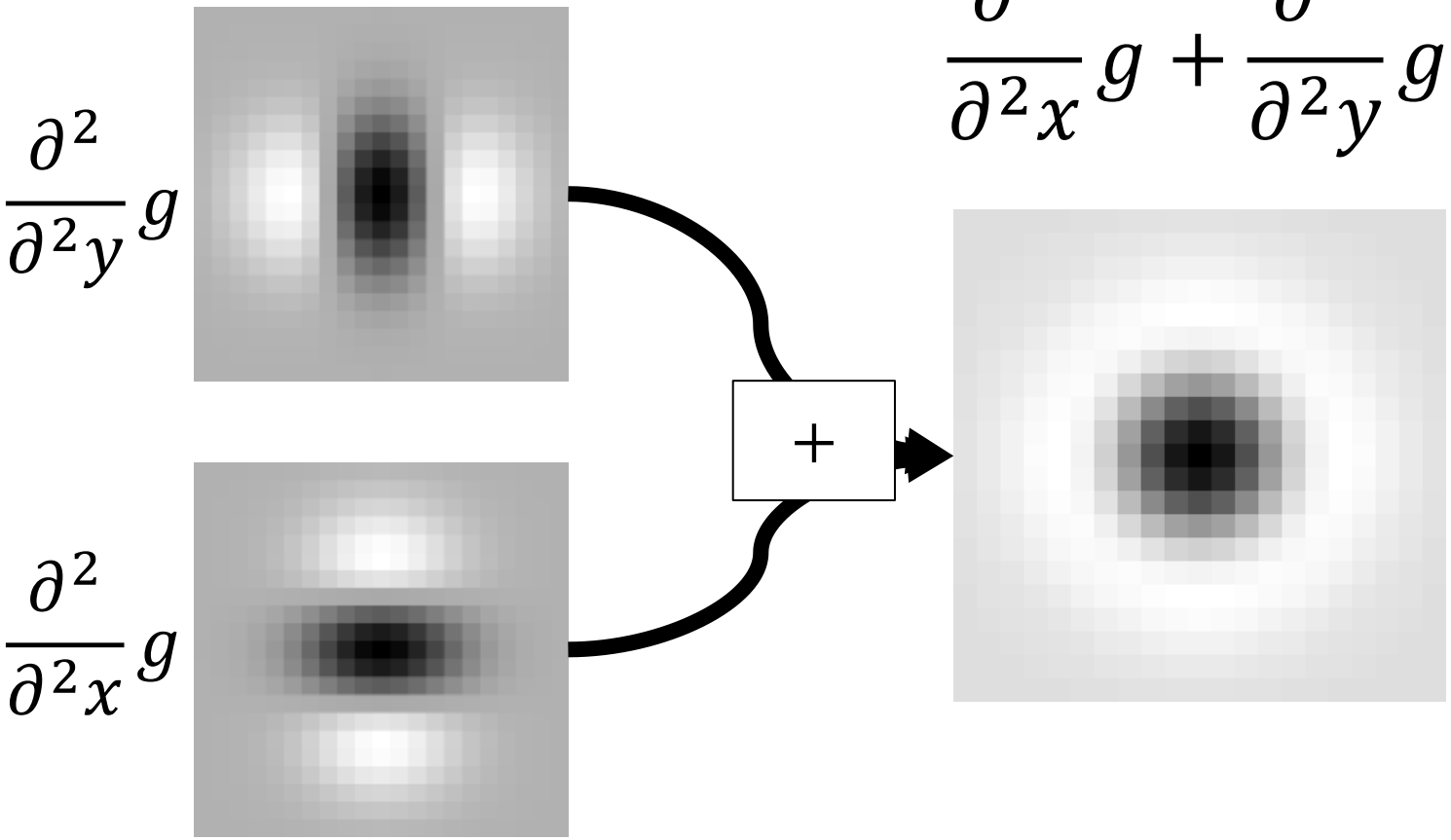


$\frac{\partial^2}{\partial^2 x} g$



Laplacian of Gaussian (LoG)

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



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

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

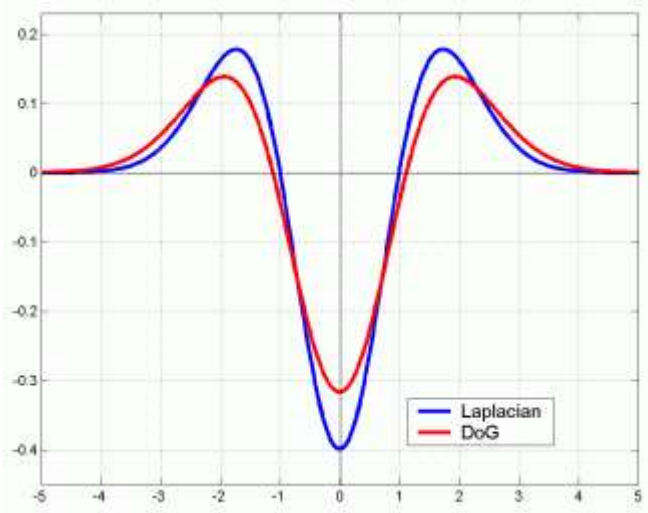
尺度不变性：对比 LoG DoG

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

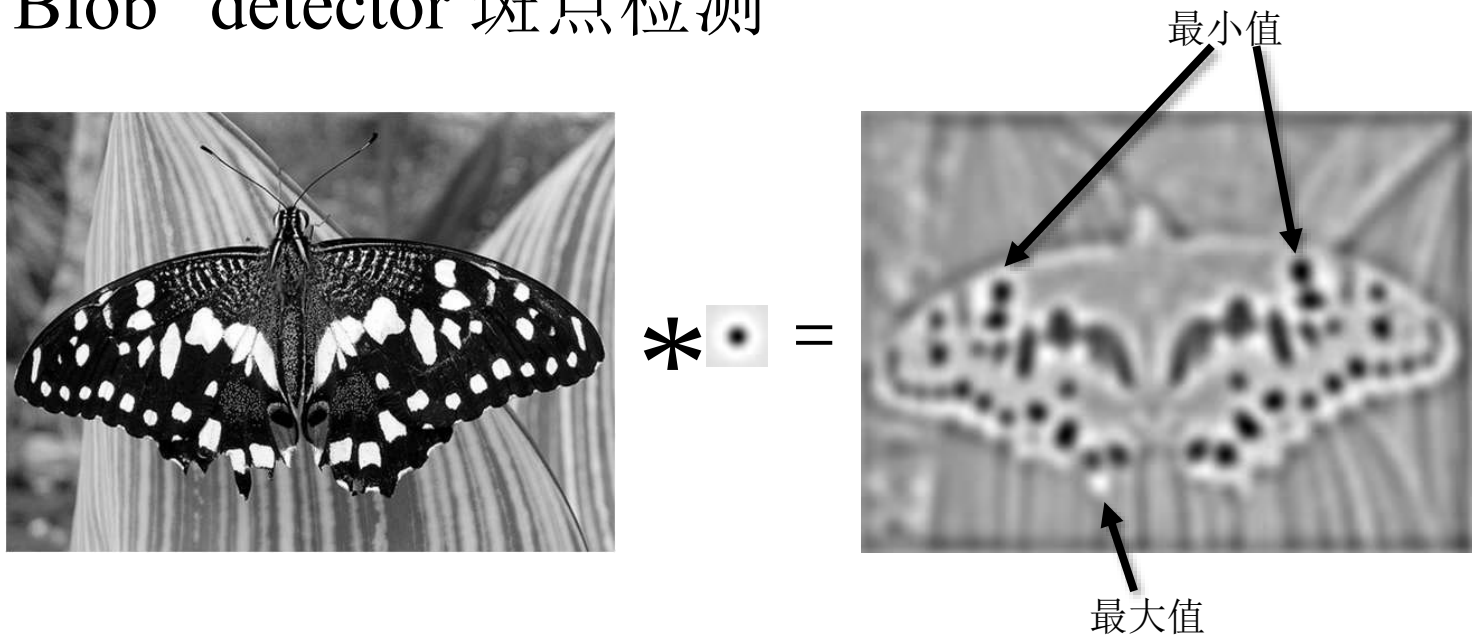


使用差分图像
边缘更细腻

LoG 和DoG 均对旋转保证不变性

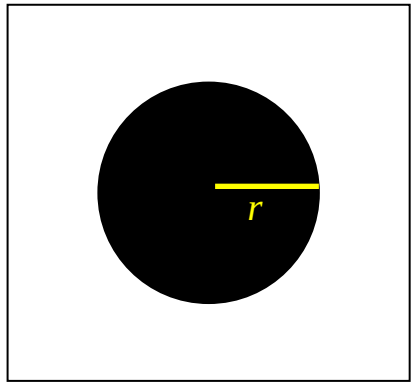
Laplacian of Gaussian （LoG）

- “Blob” detector 斑点检测

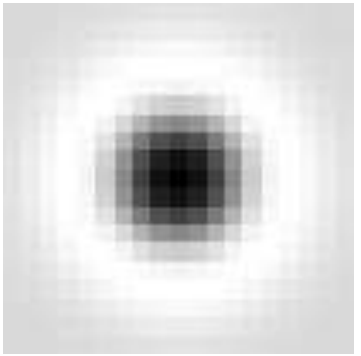


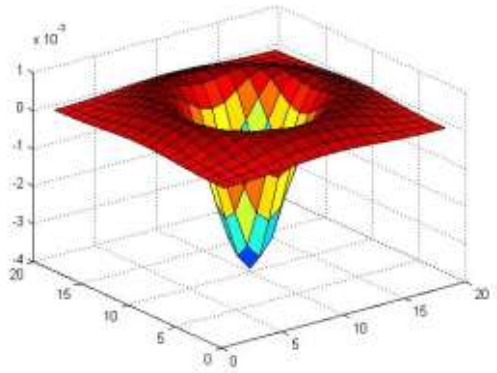
- 找到LoG 算子在空间和尺度上的最大值和最小值

特征尺度



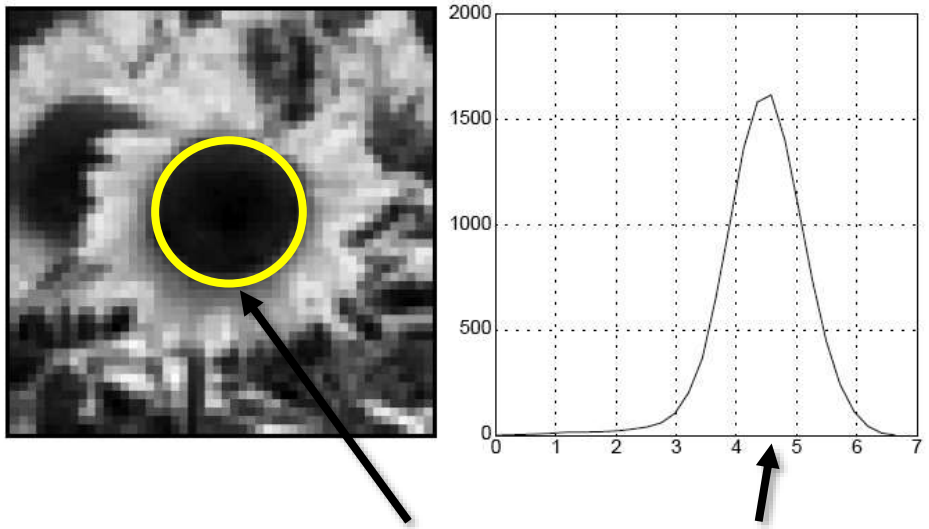
image





Laplacian

与角点相同，计算响应最大的尺度

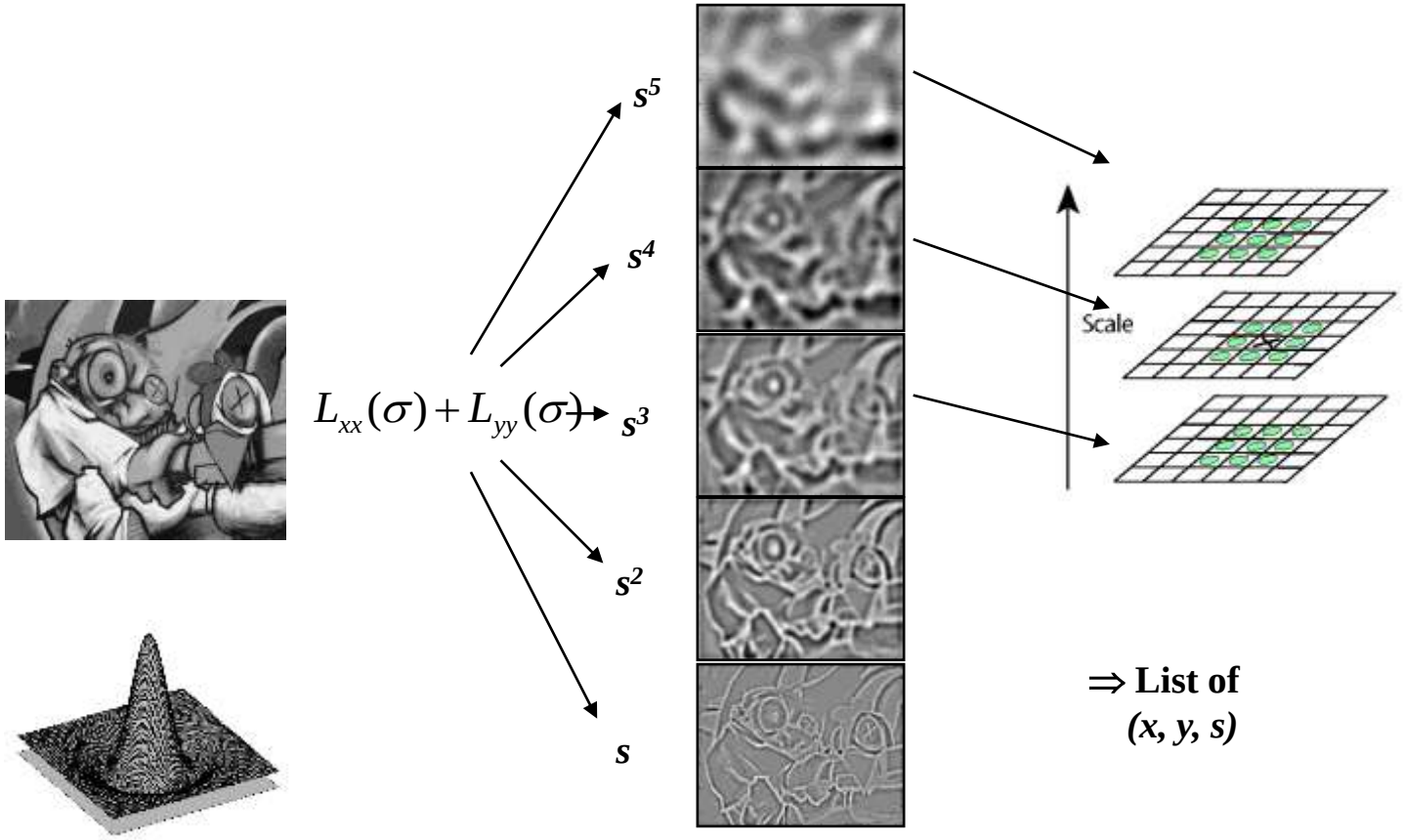


characteristic scale

- 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

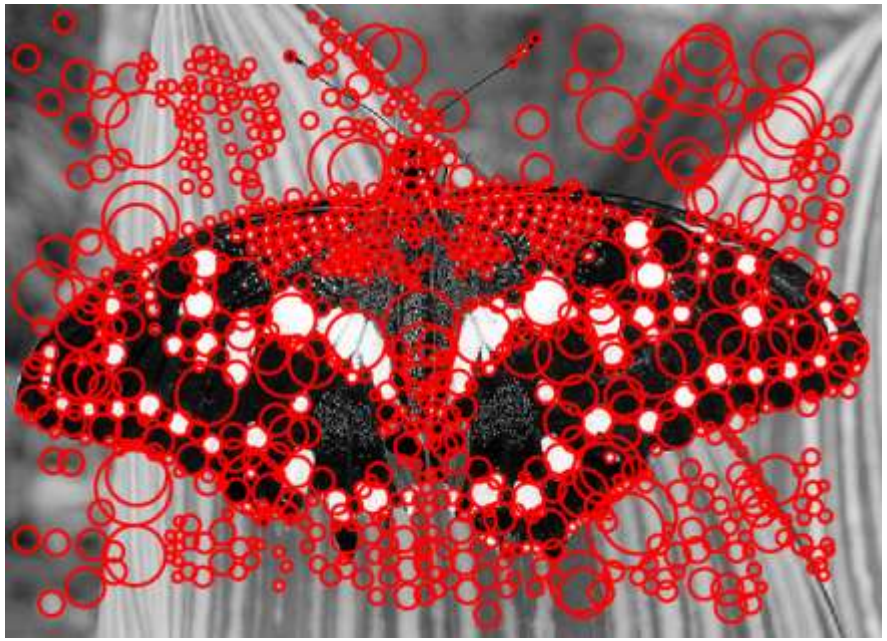


斑点检测

斑点检测：不同尺度

sigma = 11.9912

斑点检测：最终效果



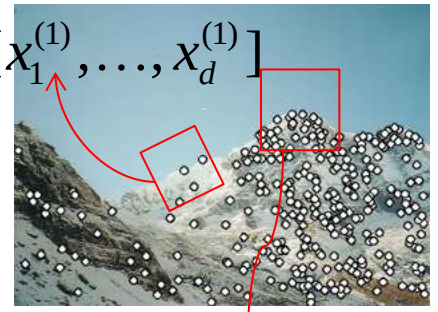
特征提取和匹配

基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点

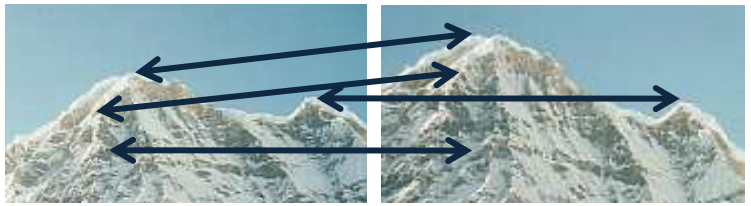
2) **解释 Description: 在关键点周围提取特征**

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



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

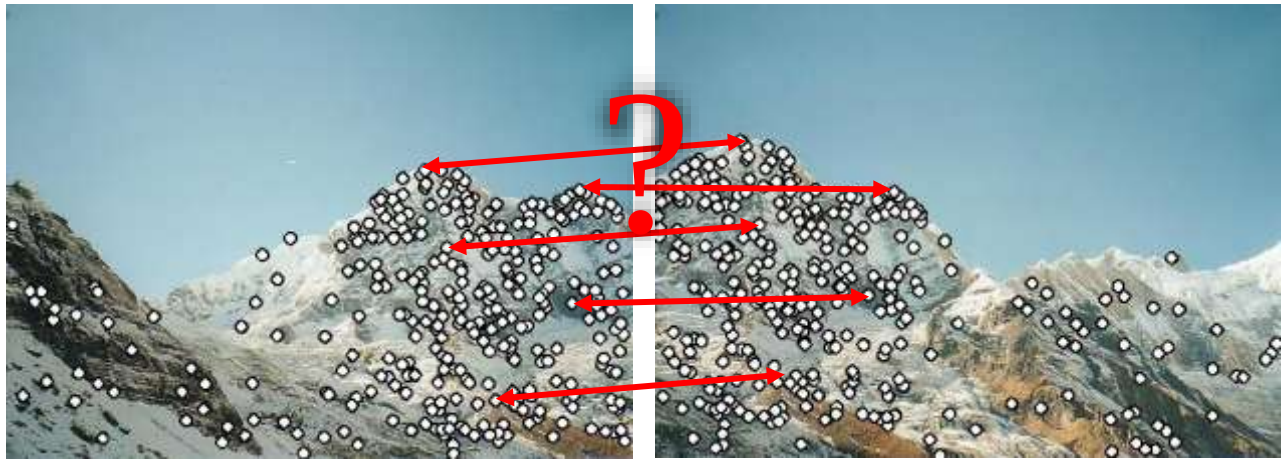
3) 匹配 Matching: 根据两个视角下的特征进行匹配



Kristen Grauman

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?**



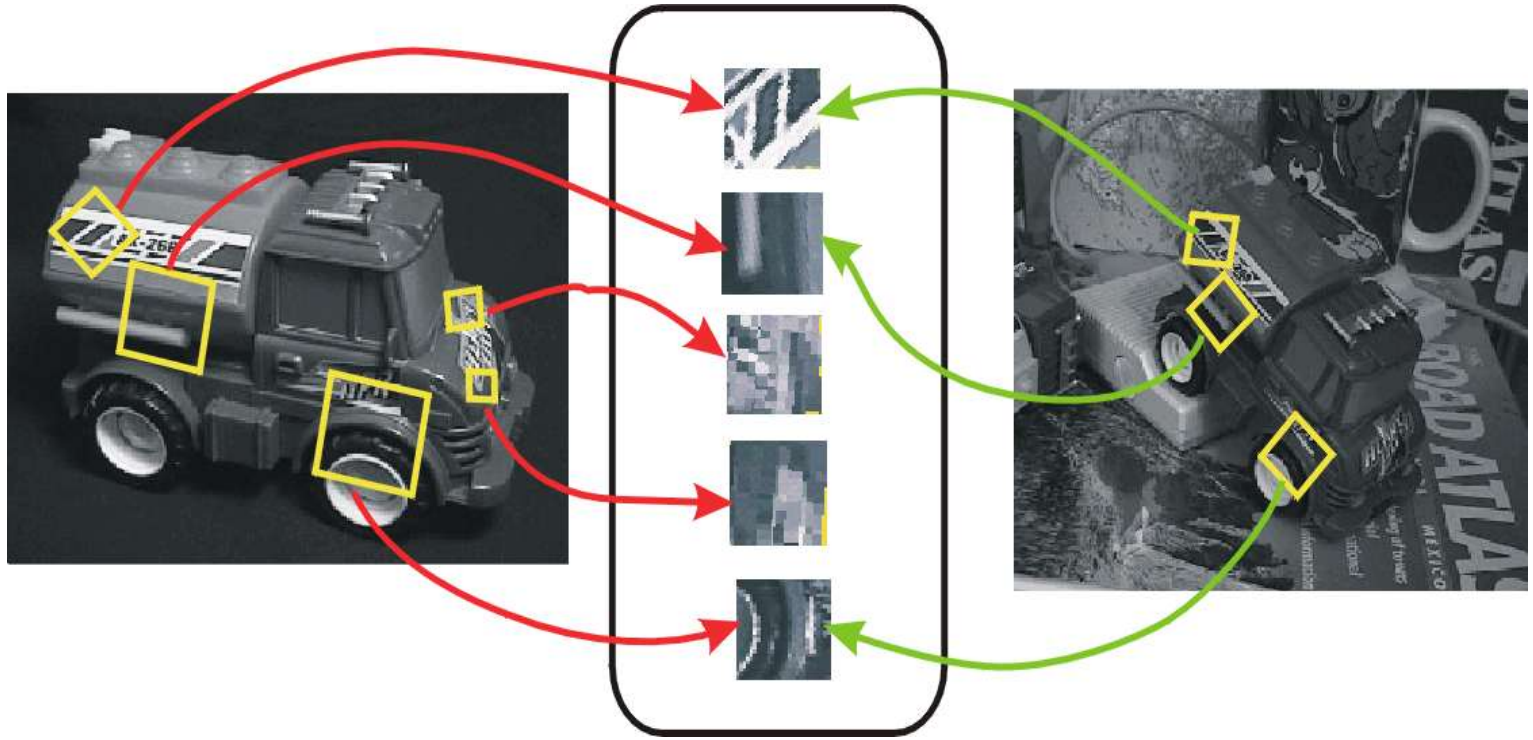
Answer: 使用特征提取器（特征描述子， descriptor）
怎么做?

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

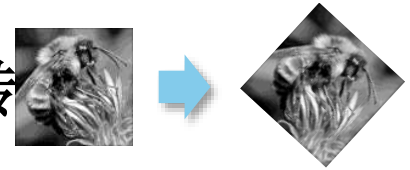
找到在图像变换下仍然保持不变的图像特征

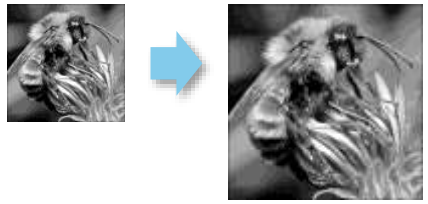
- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



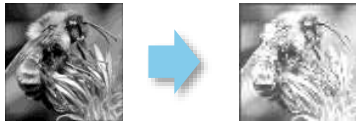
特征提取器

- 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

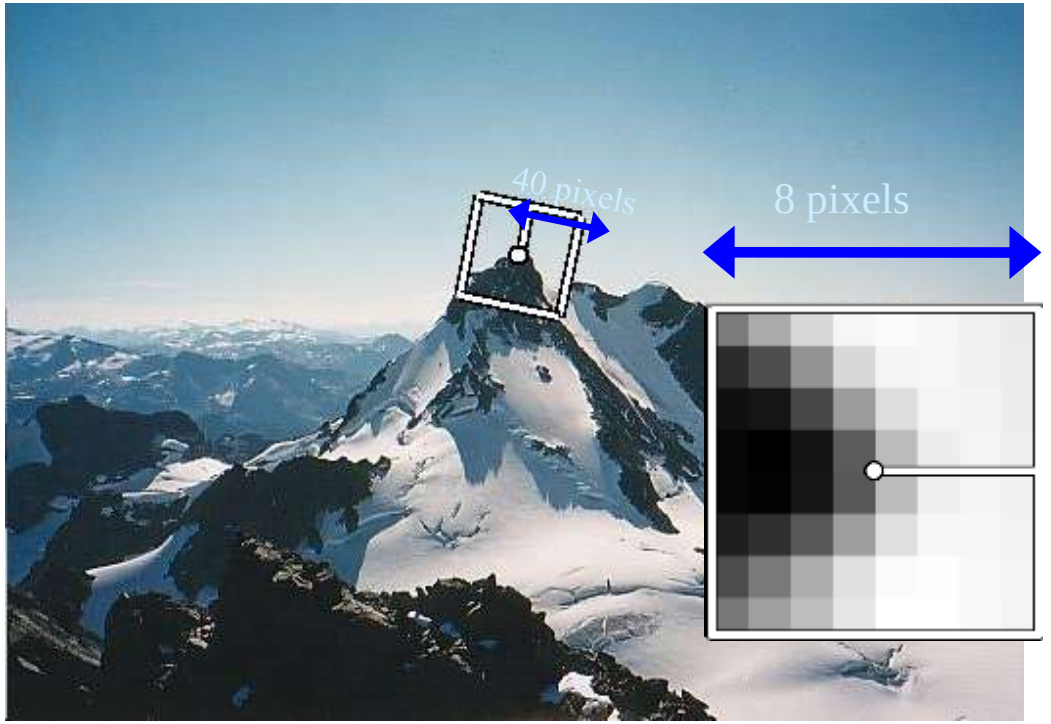
- 找到图像块的主要方向
  - 计算特征值 Eigenvalue
    - (最大的特征值)
  - 计算梯度方向
  - 旋转不同方向并提取特征

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

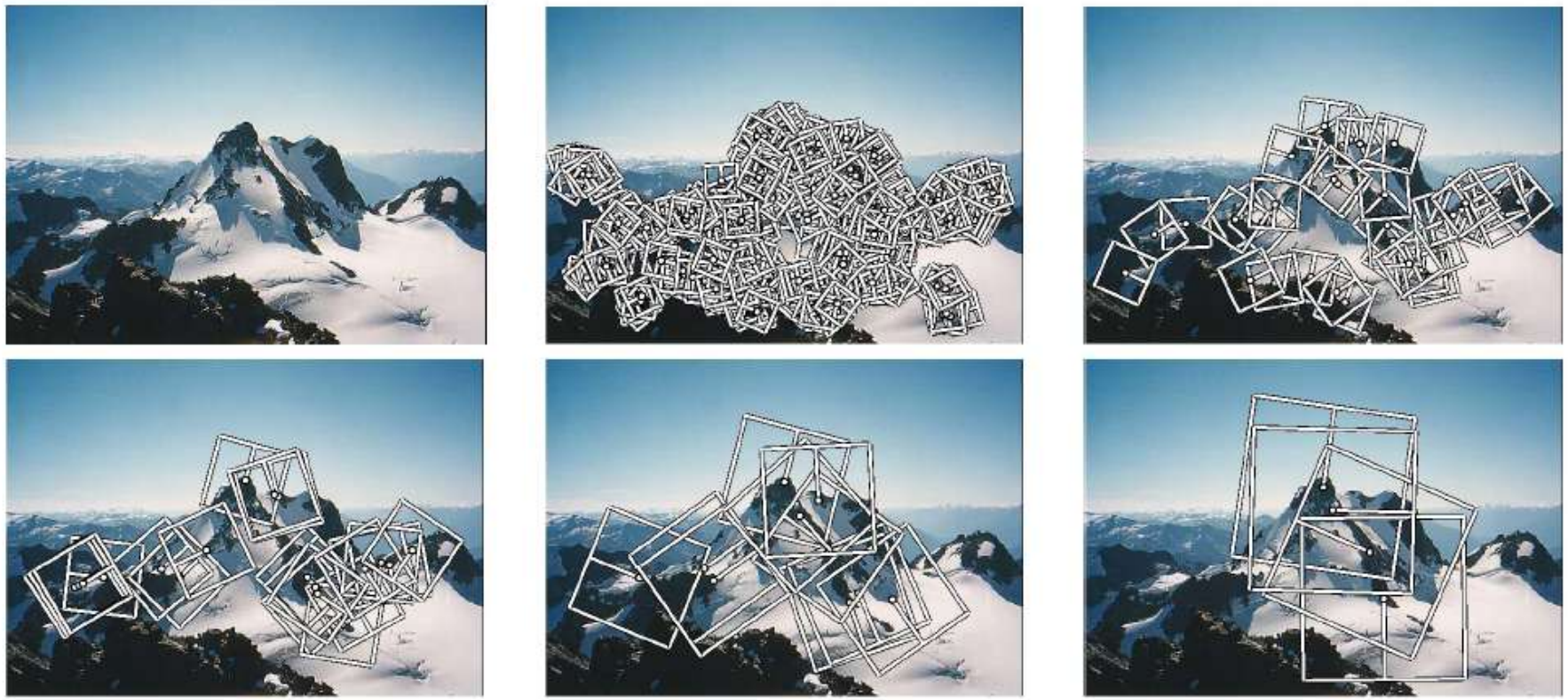
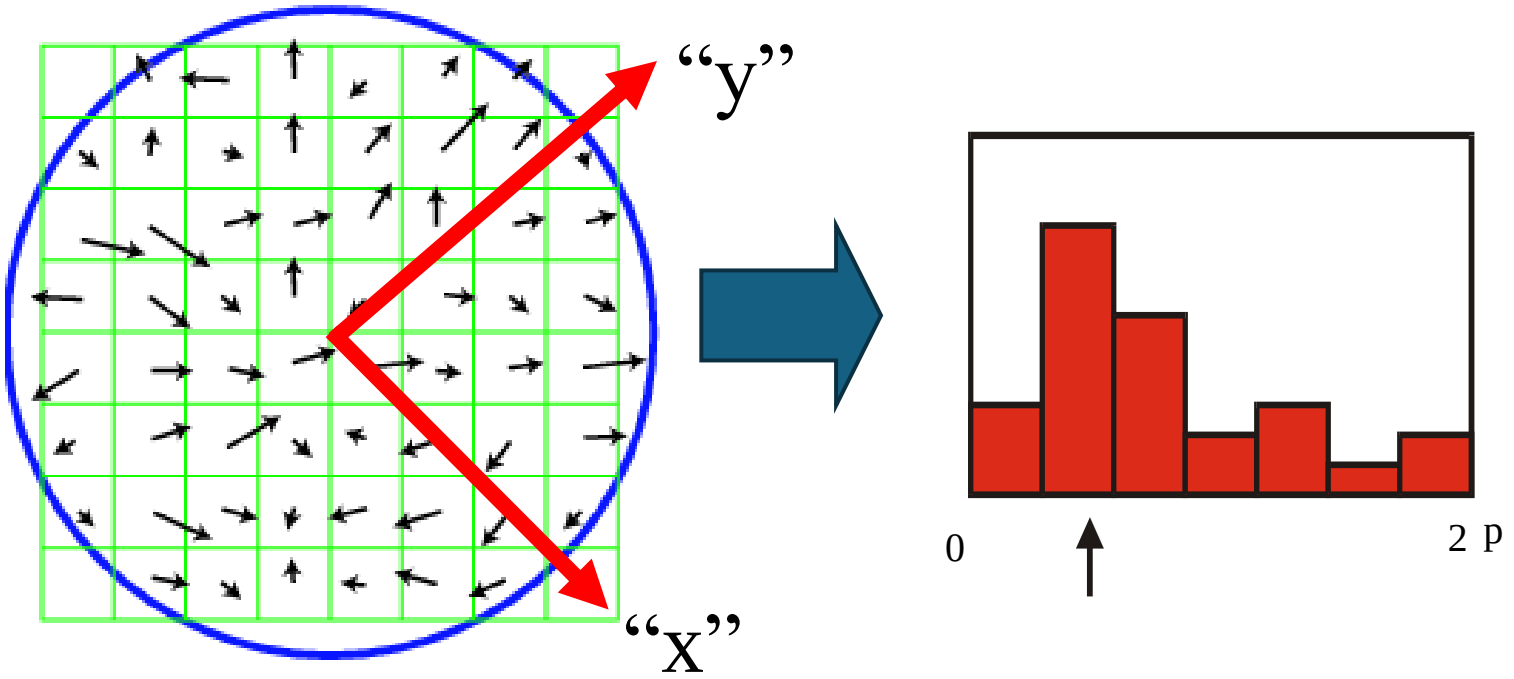


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

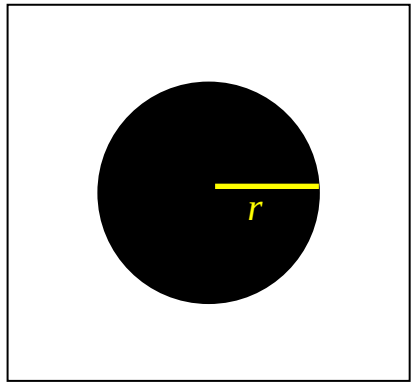
旋转不变：另一种方式

给定窗口，找到像素点梯度方向最多的方向

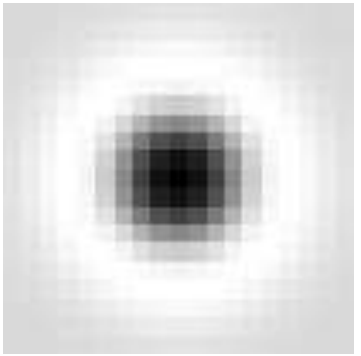


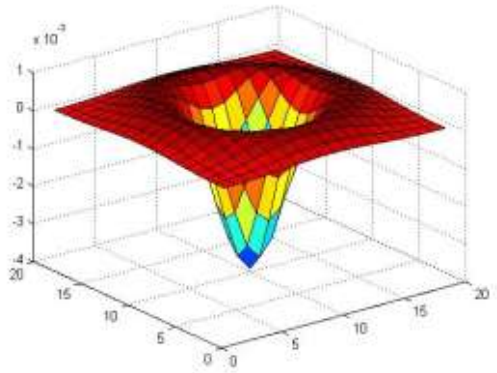
Slide credit: S. Lazebnik

特征尺度



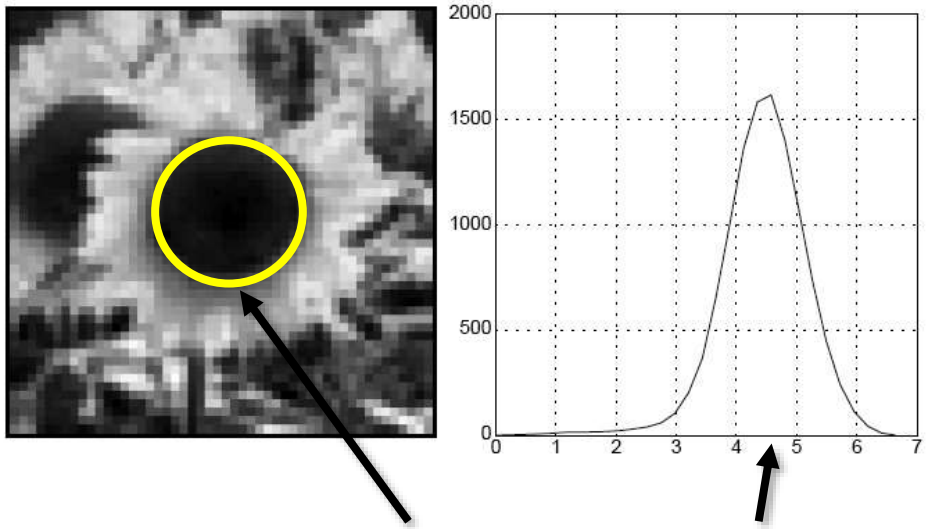
image





Laplacian

与角点相同，计算响应最大的尺度

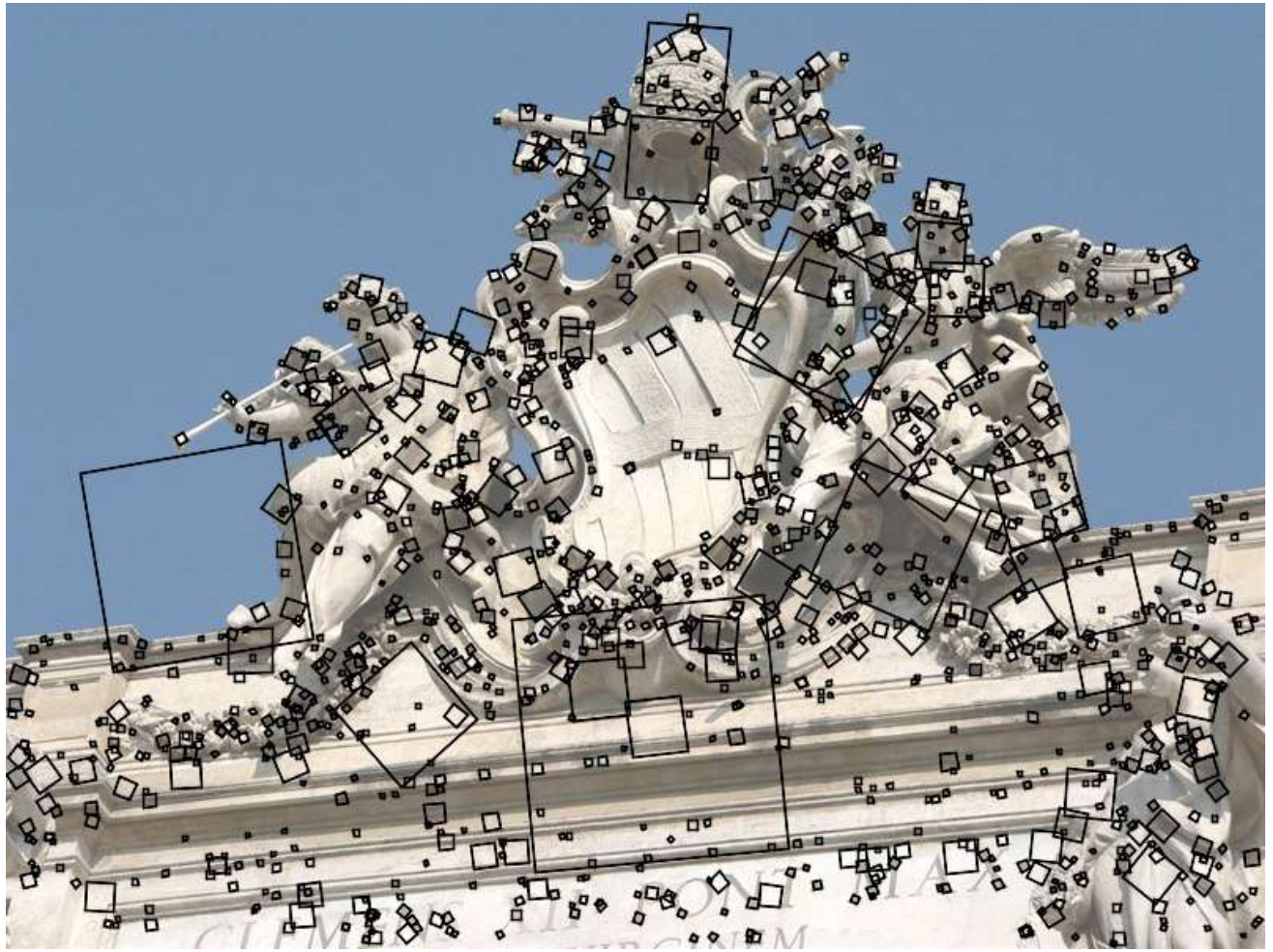


characteristic scale

- 首先找到特征尺度确定窗口大小

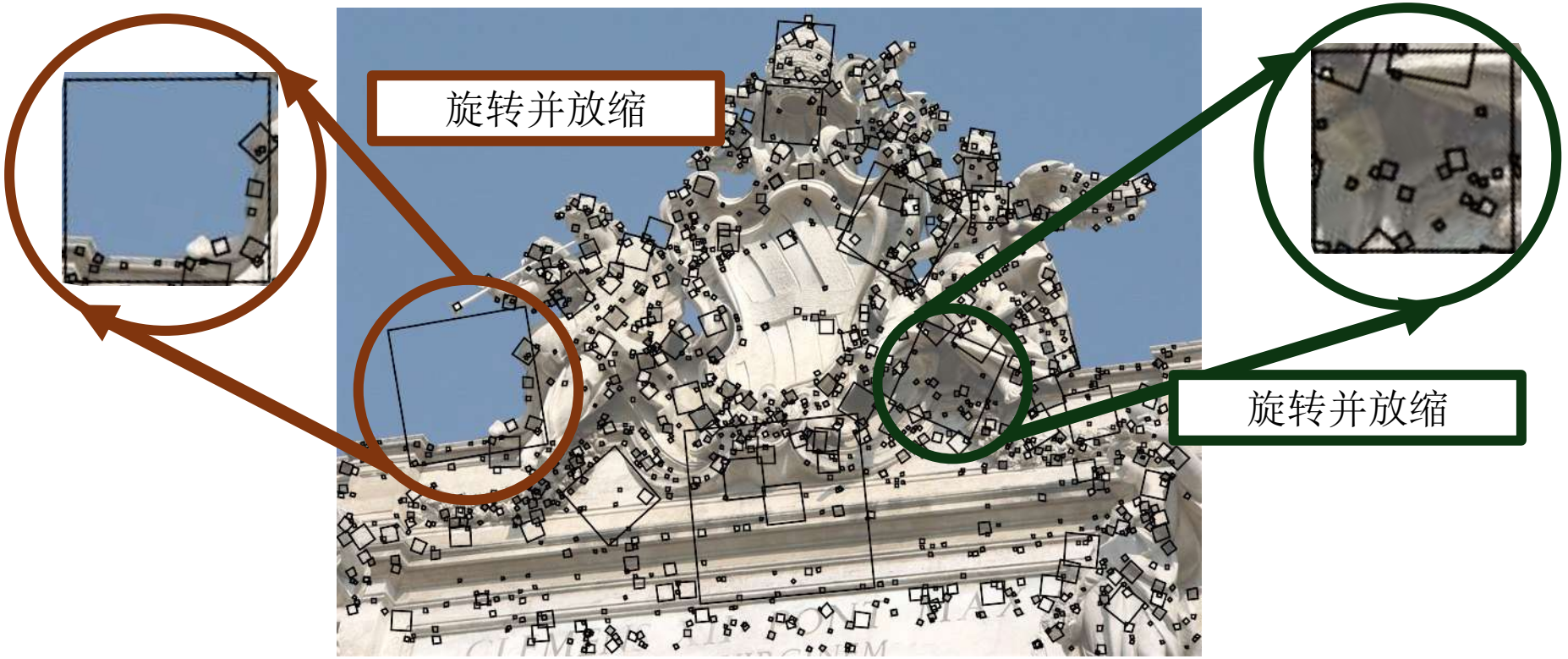
T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

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.

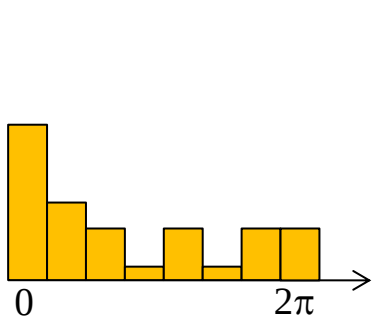
Scale Invariant Feature Transform

想法:

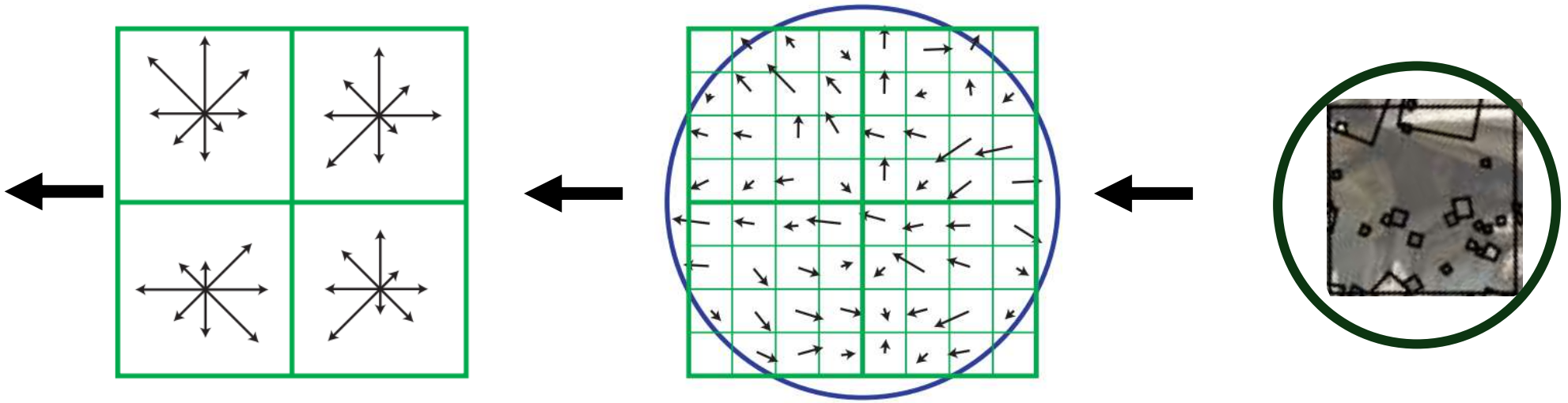
- 在检测到的特征周围取16x16的方形窗口
- 为每个像素计算边缘方向（梯度的角度 - 90度）
  - 减少亮度影响
- 排除弱边缘（梯度幅值低于阈值）
- 为剩余边缘方向创建直方图

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



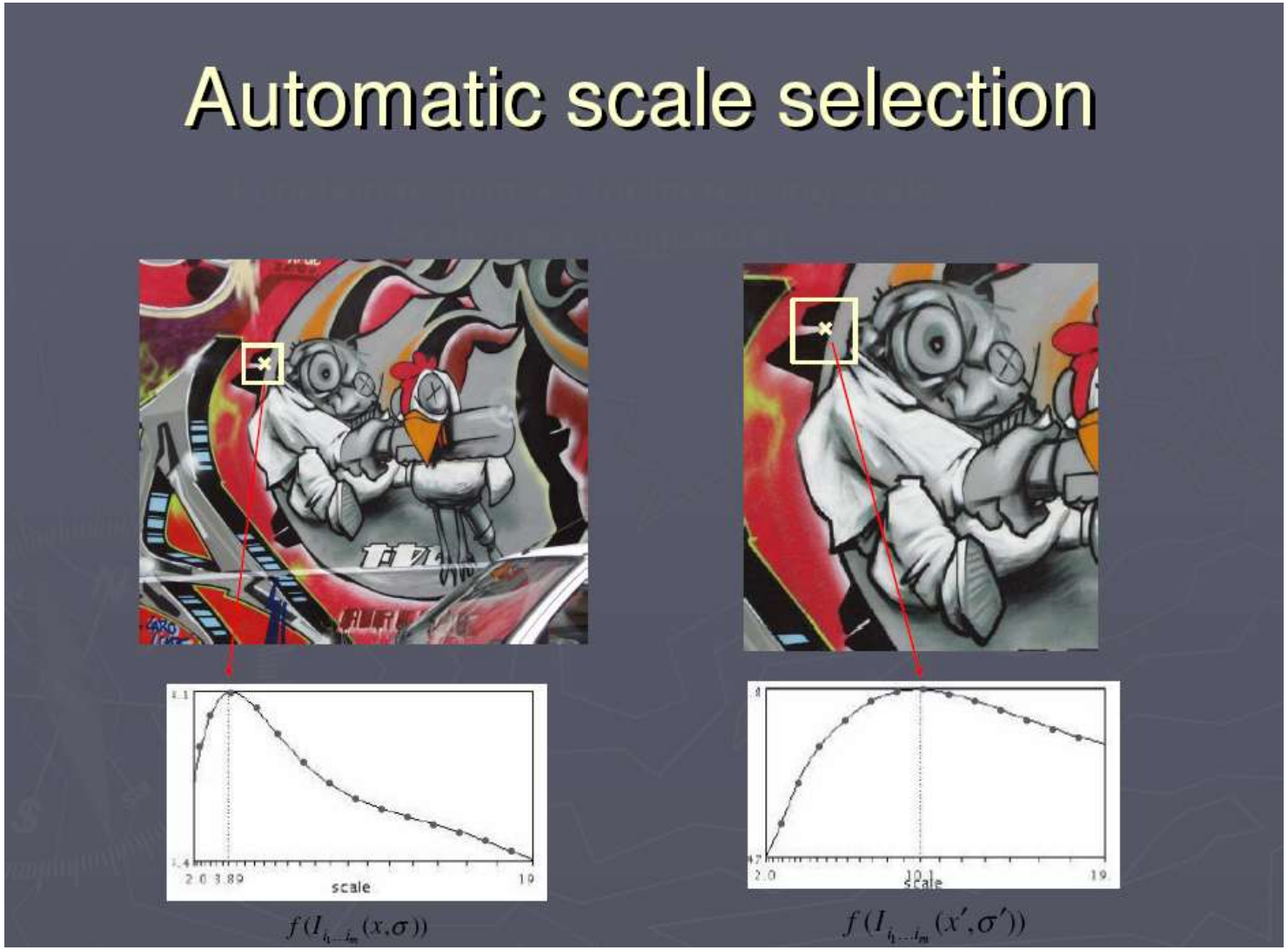
梯度角度直方图



SIFT Descriptors

- In principle: build a histogram of the gradients
- In reality: quite complicated
  - Gaussian weighting: smooth response
  - Normalization: reduces illumination effects
  - Clamping
  - Tons of more stuff

尺度不变性？特征尺度

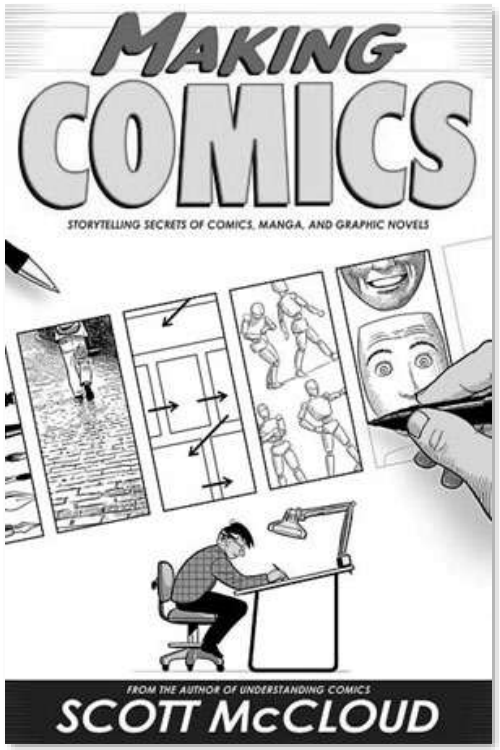

Automatic scale selection
scale
scale
$f(I_{i_1\ldots i_m}(x,\sigma))$
$f(I_{i_1\ldots i_m}(x',\sigma'))$

SIFT

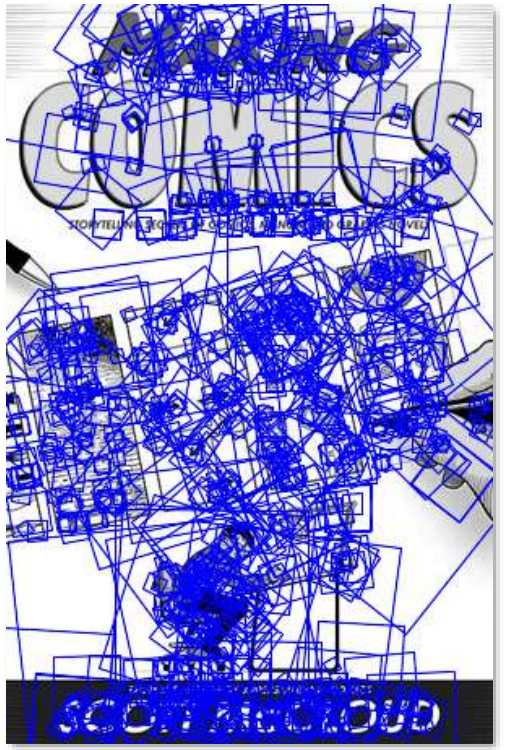
十分稳定的特征提取技术

- 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



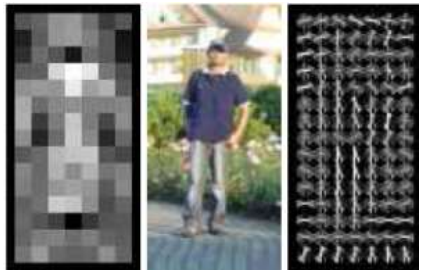
sift



868 SIFT 特征

其他特征提取器

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



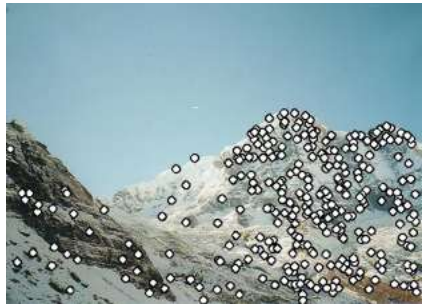
- FREAK: Fast Retina Keypoint
  - 用在SLAM 实时提取

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

    https://arxiv.org/abs/1603.09114

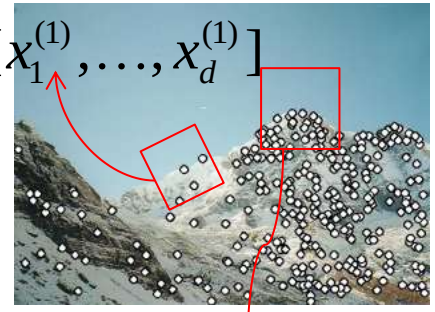
基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点



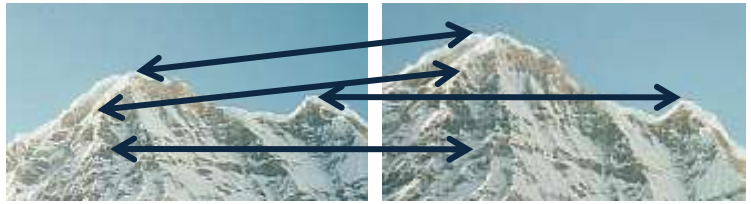
2) 解释 Description: 在关键点周围提取特征

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



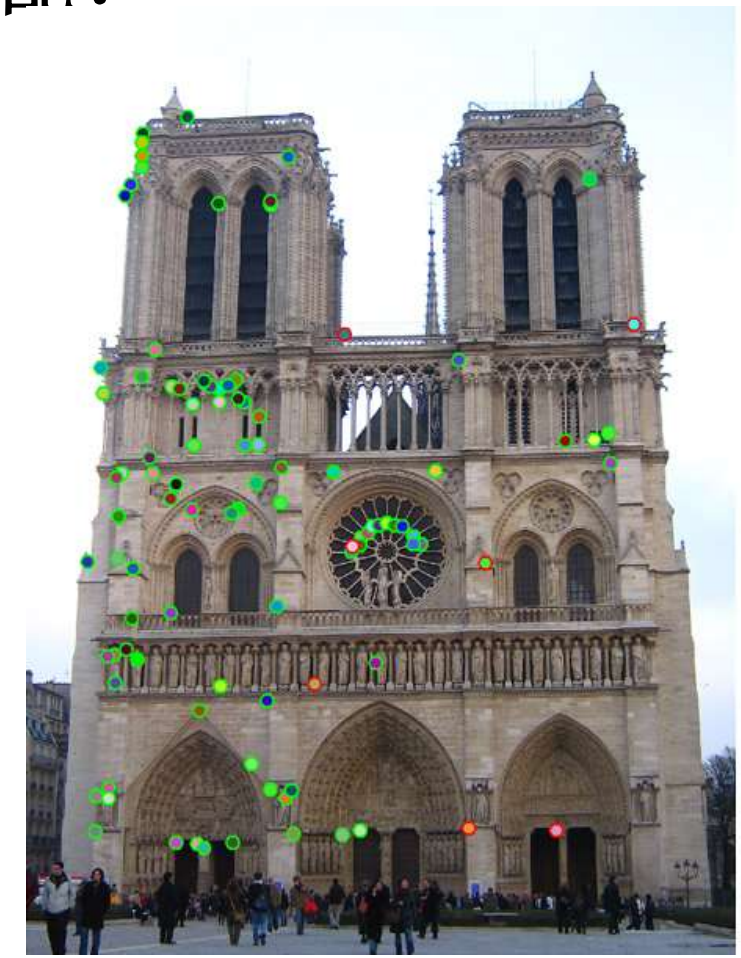
$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

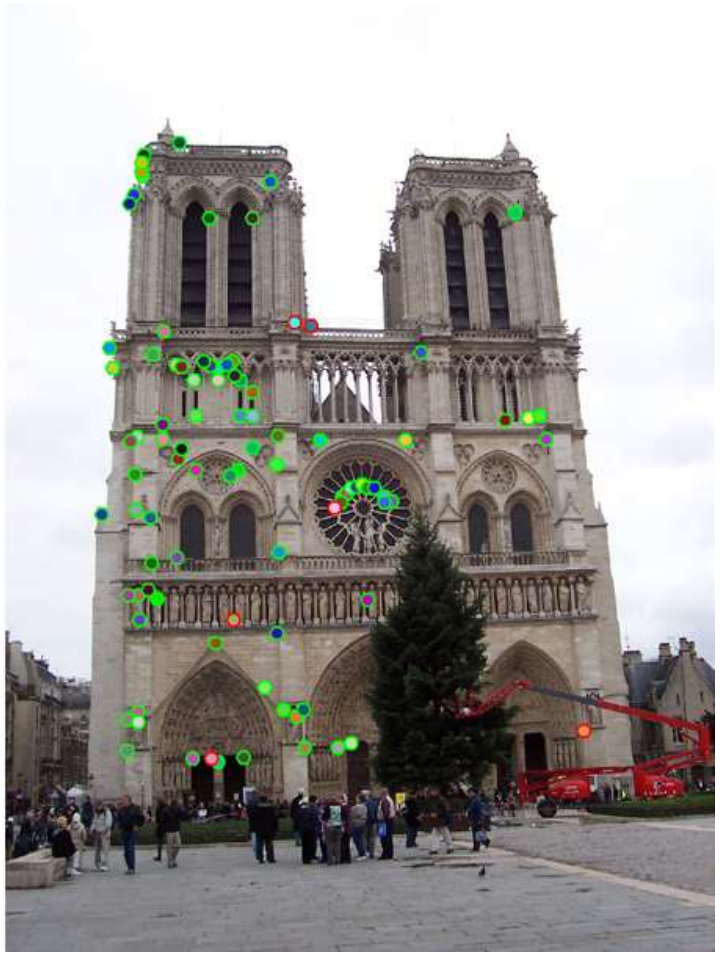
3) **匹配 Matching: 根据两个视角下的特征进行匹配**



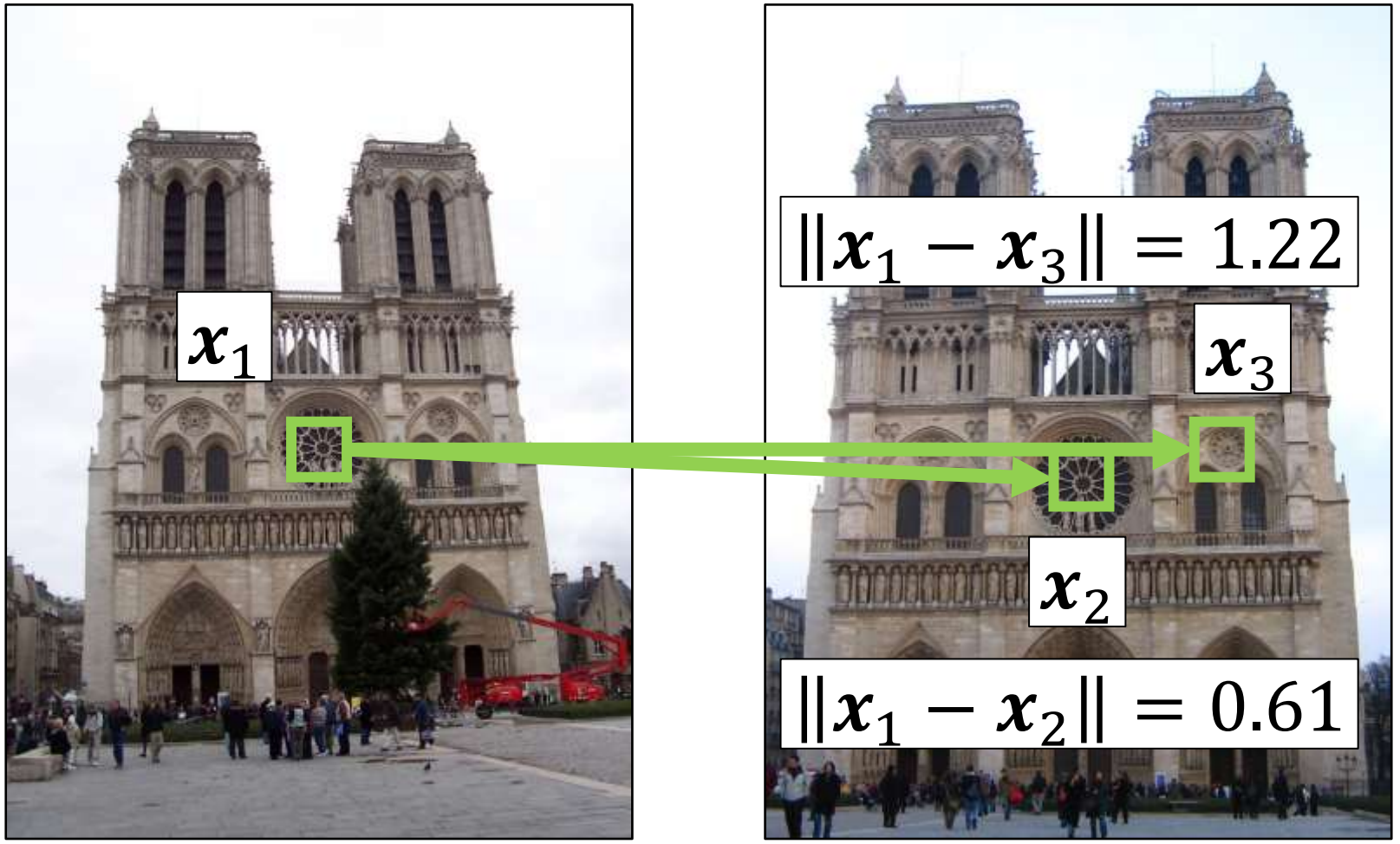
Kristen Grauman

怎么匹配？



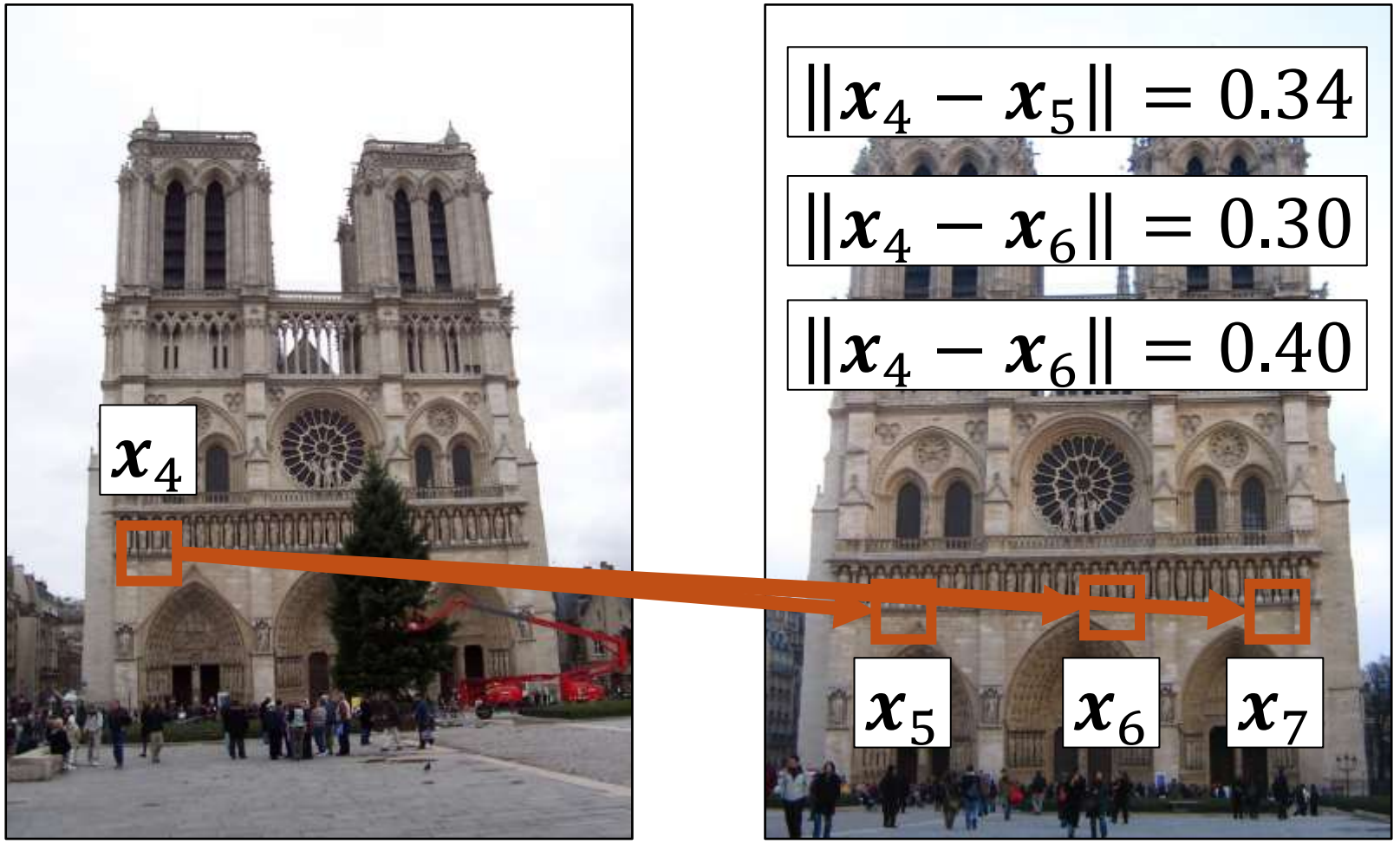


怎么匹配？



Example credit: J. Hays

怎么匹配？

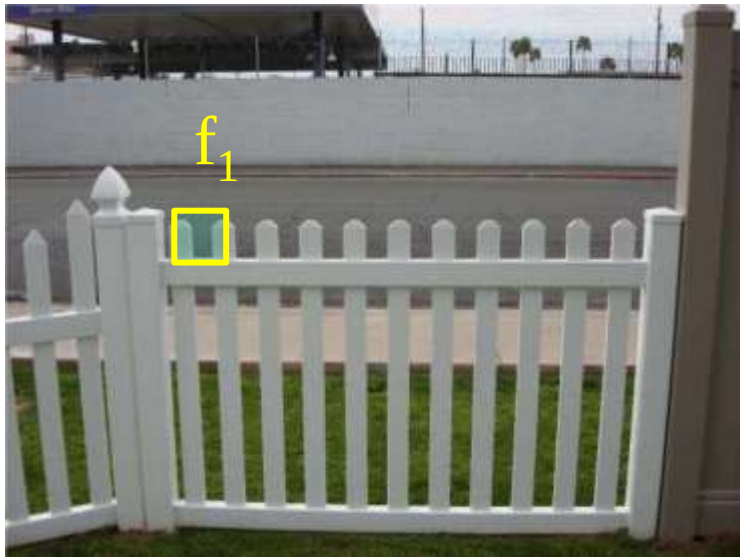


Example credit: J. Hays

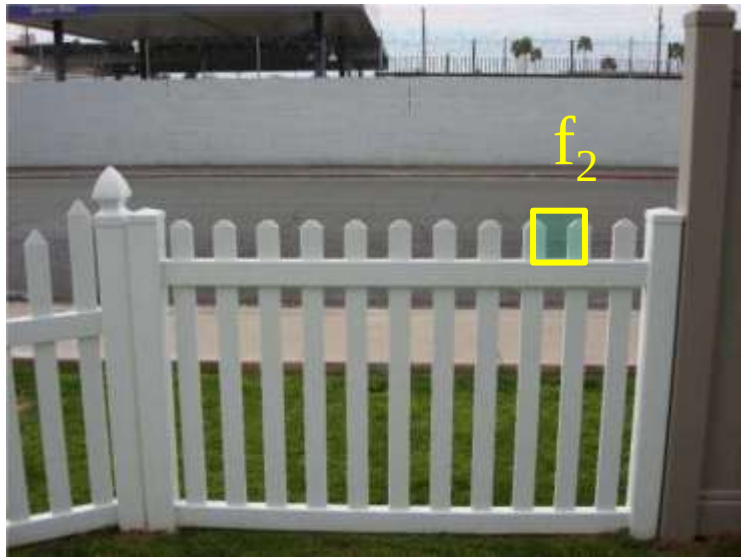
特征“距离”

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

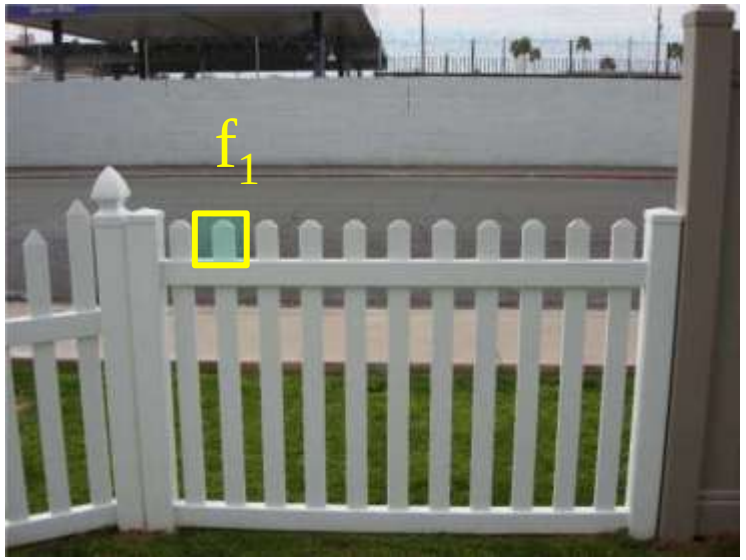


I_2

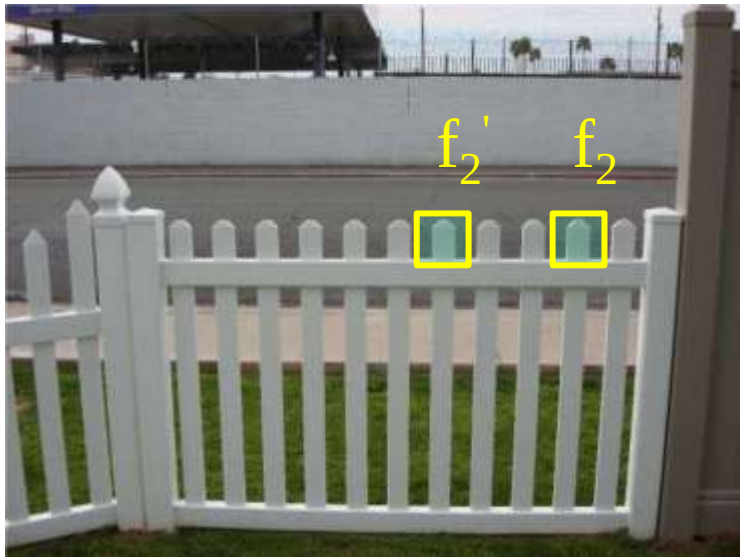
特征“距离”

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



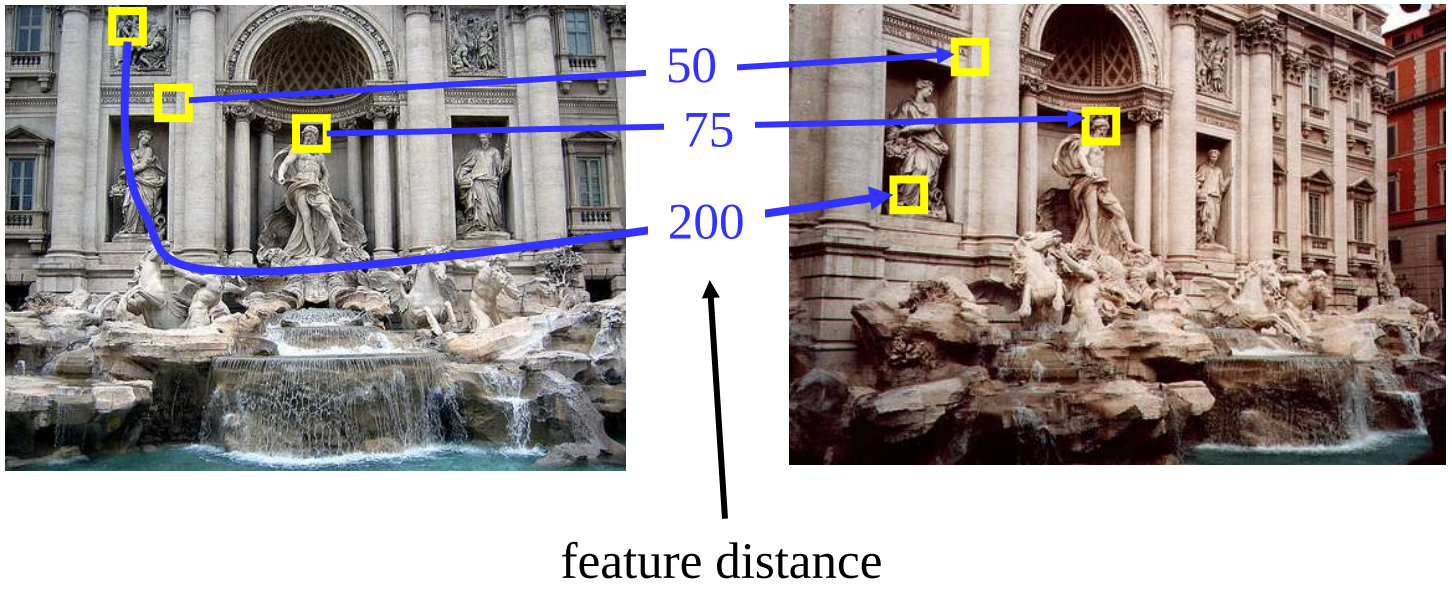
I_1



I_2

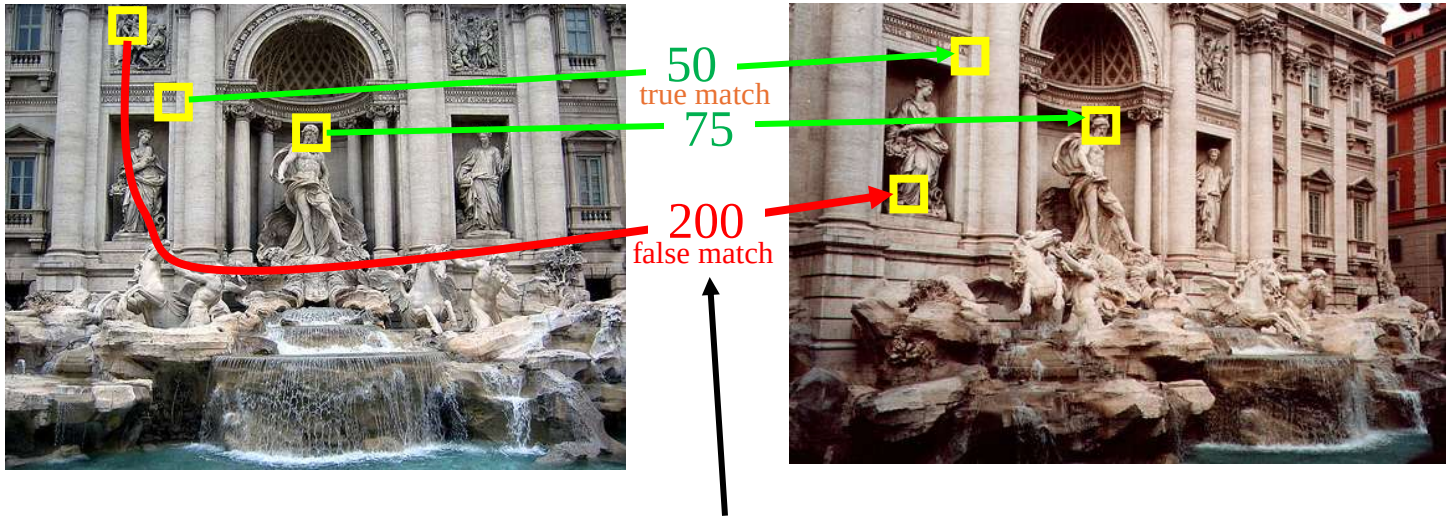
Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

How can we measure the performance of a feature matcher?

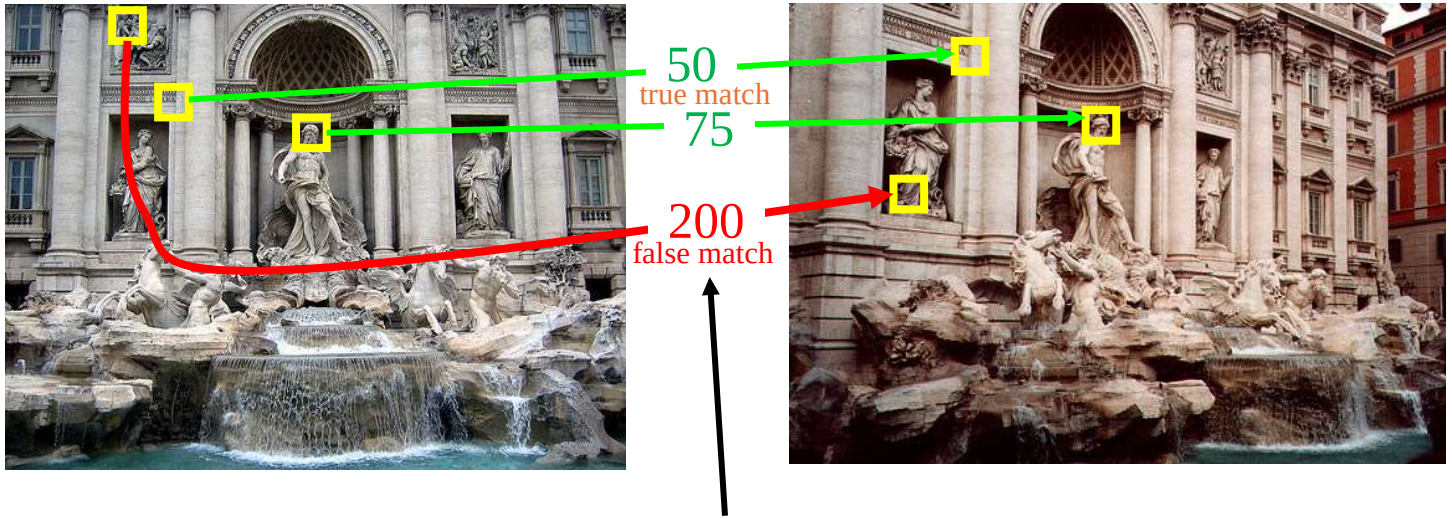


The distance threshold affects performance

- True positives = # of detected matches that survive the threshold that are correct
- False positives = # of detected matches that survive the threshold that are incorrect

True/false positives

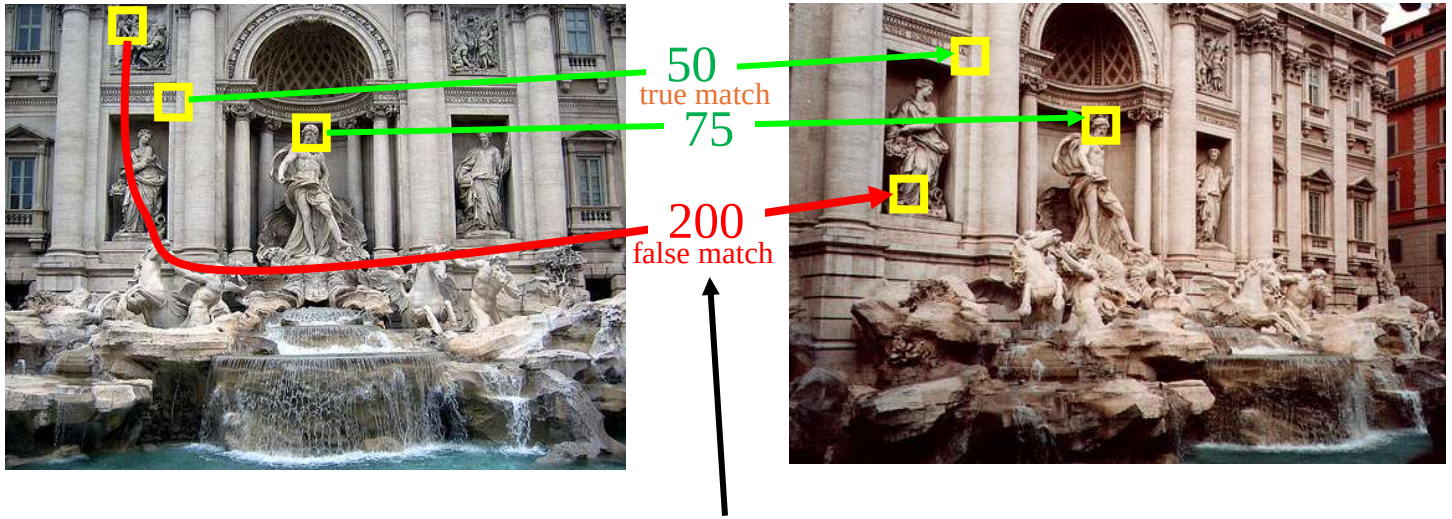
How can we measure the performance of a feature matcher?



Suppose we want to maximize true positives. How do we set the threshold? (We keep all matches with distance below the threshold.)

True/false positives

How can we measure the performance of a feature matcher?



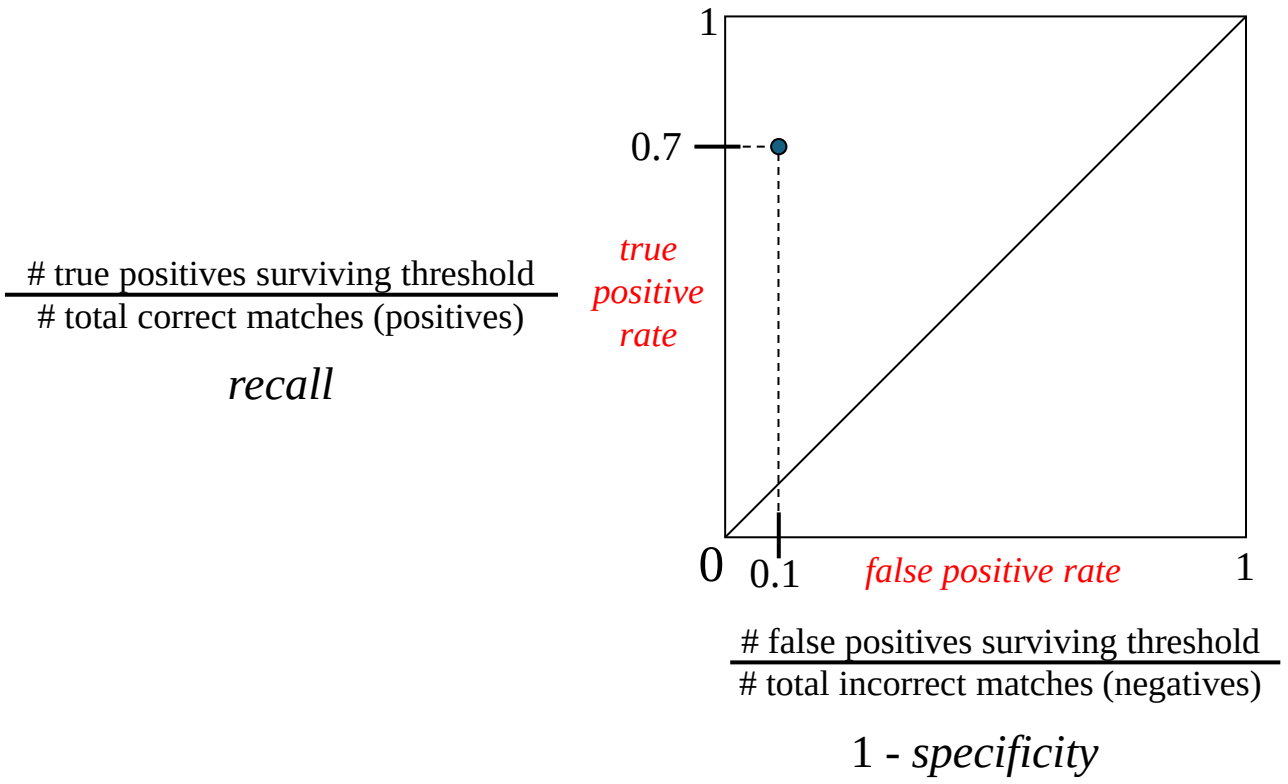
Suppose we want to minimize false positives. How do we set the threshold? (We keep all matches with distance below the threshold.)

Example

- Suppose our matcher computes 1,000 matches between two images
  - 800 are correct matches, 200 are incorrect (according to an oracle that gives us ground truth matches)
  - A given threshold (e.g., ratio distance = 0.6) gives us 600 correct matches and 100 incorrect matches that survive the threshold
  - True positive rate = 600 / 800 = ¾
  - False positive rate = 100 / 200 = ½

Evaluating the results

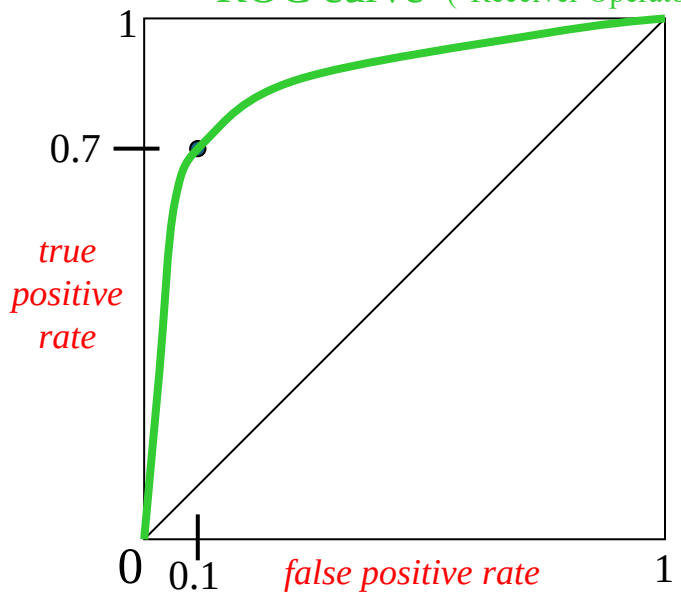
How can we measure the performance of a feature matcher?



Evaluating the results

How can we measure the performance of a feature matcher?

ROC curve ("Receiver Operator Characteristic")



$$\frac{\text{\# true positives surviving threshold}}{\text{\# total correct matches (positives)}}$$

recall

$$\frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

1 - *specificity*

Single number: Area Under the Curve (AUC)

E.g. AUC = 0.87

1 is the best

ROC curves – summary

- By thresholding the match distances at different thresholds, we can generate sets of matches with different true/false positive rates
- ROC curve is generated by computing rates at a set of threshold values swept through the full range of possible threshold
- Area under the ROC curve (AUC) summarizes the performance of a feature pipeline (higher AUC is better)

More on feature detection/description

http://www.robots.ox.ac.uk/~vgg/research/affine/
http://www.cs.ubc.ca/~lowe/keypoints/
http://www.vision.ee.ethz.ch/~surf/

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: K. Mikolajczyk and C. Schmid, Scale and Affine invariant interest point detectors. In IJC V 60(1):63-86, 2004. PDF
- *MSER*: J.Matas, O. Chum, M. Urban, and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. PDF
- *IBR & EBR*: T.Tuytelaars and L. Van Gool, Matching widely separated views based on affine invariant regions . In IJCV 59(1):61-85, 2004. PDF
- *Salient regions*: T. Kadir, A. Zisserman, and M. Brady, An affine invariant salient region detector. In ECCV p. 404-416, 2004. PDF
- *All Detectors - Survey*: T. Tuytelaars and K. Mikolajczyk , Local Invariant Feature Detectors - Survey. In CVG, 3(1):1-110, 2008. PDF

Region descriptors

- *SIFT*: D. Lowe, Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004. PDF

Performance evaluation

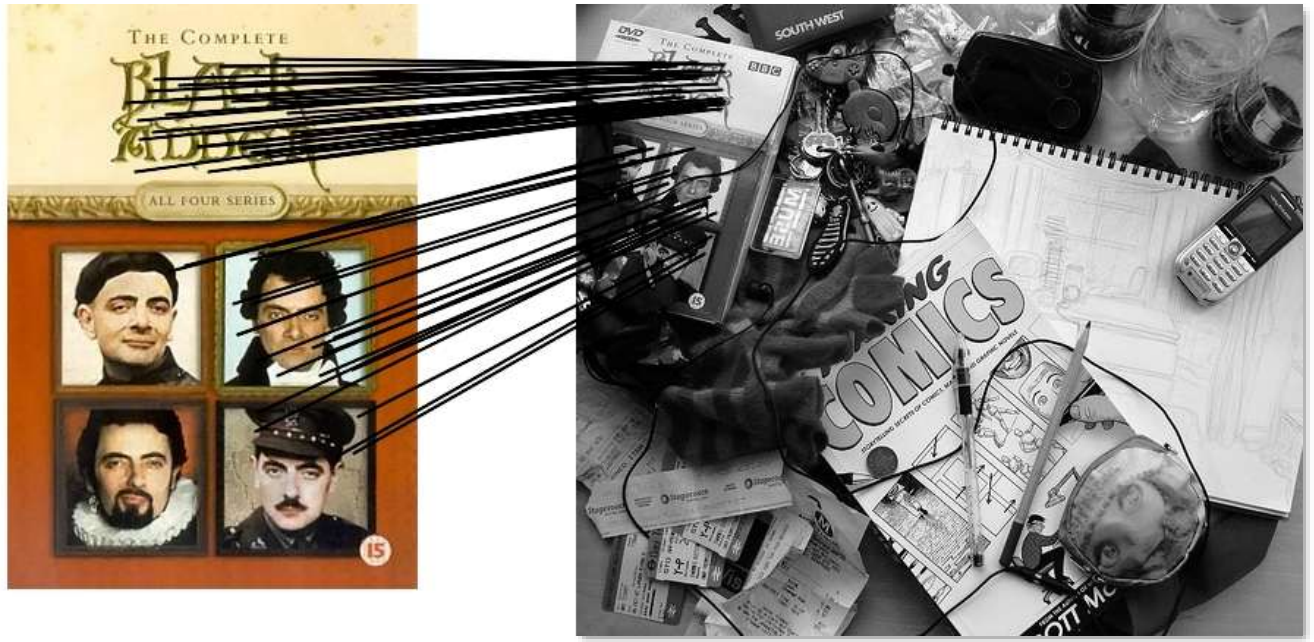
- K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, A comparison of affine region detectors. In IJCV 65(1/2):43-72, 2005. PDF
- K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. In PAMI 27(10):1615-1630 . PDF

Lots of applications

Features are used for:

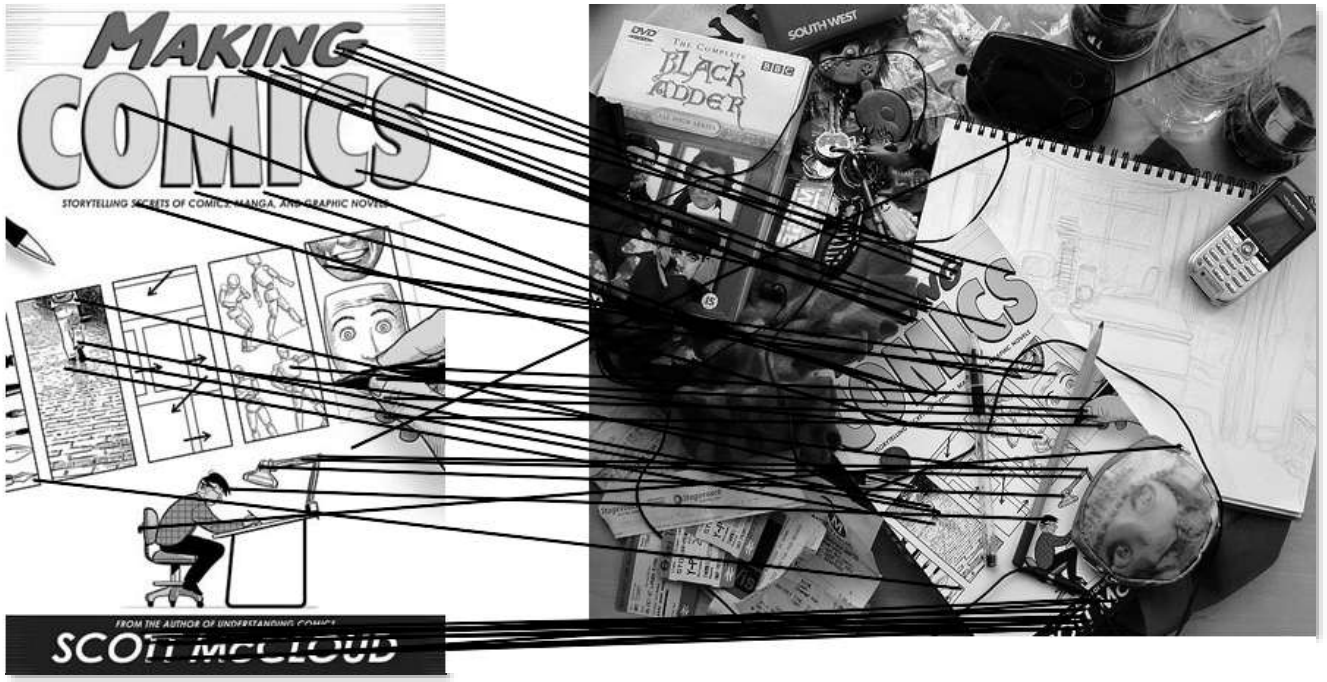
- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

特征匹配



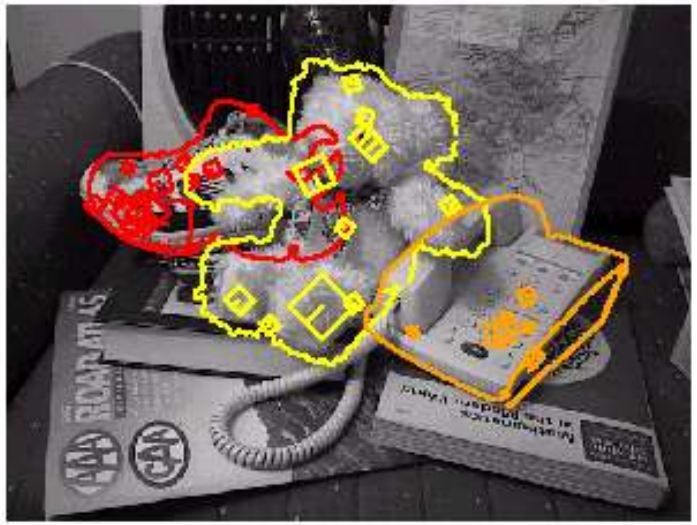
58 处匹配 (thresholded by ratio score)

特征匹配——Outliers

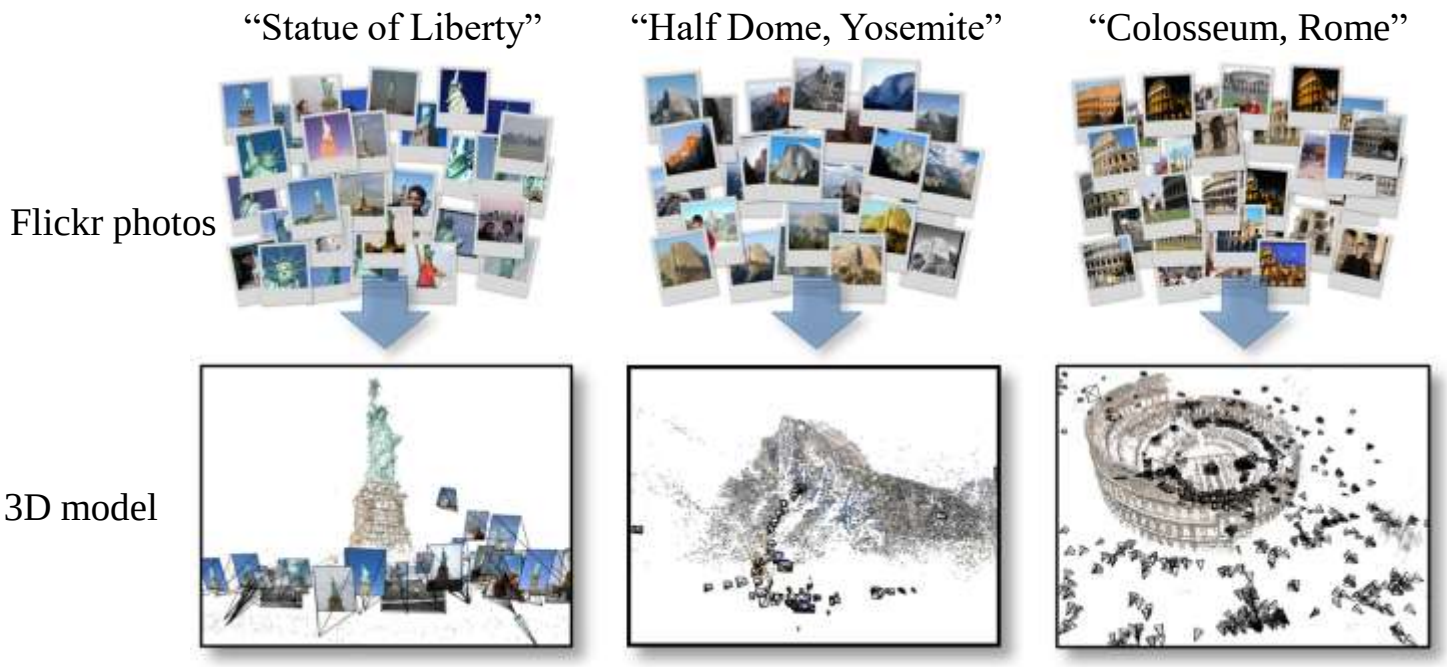


51 处匹配(thresholded by ratio score)

物体识别 (David Lowe)

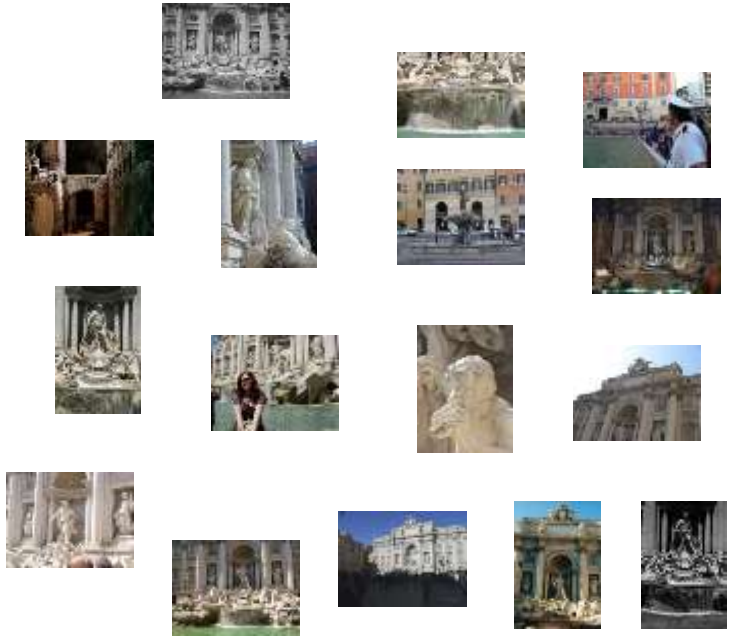


从网络照片重构一个场景



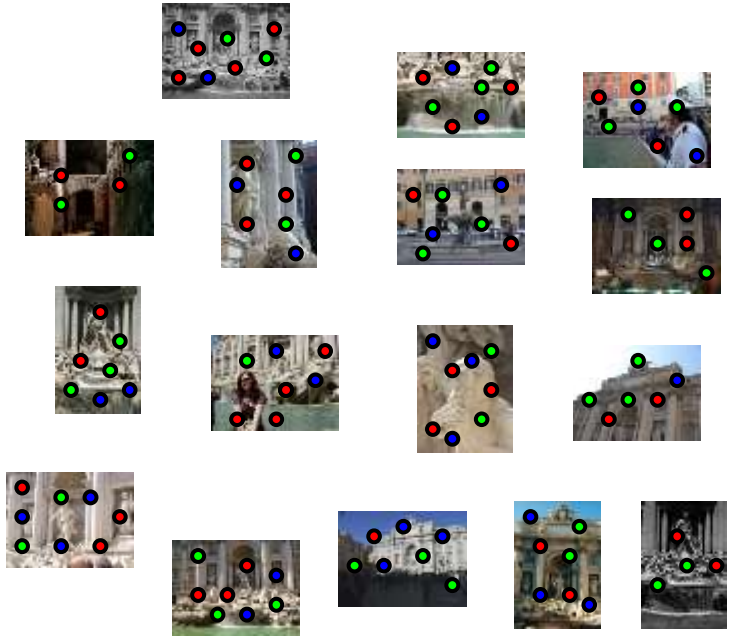
特征检测

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

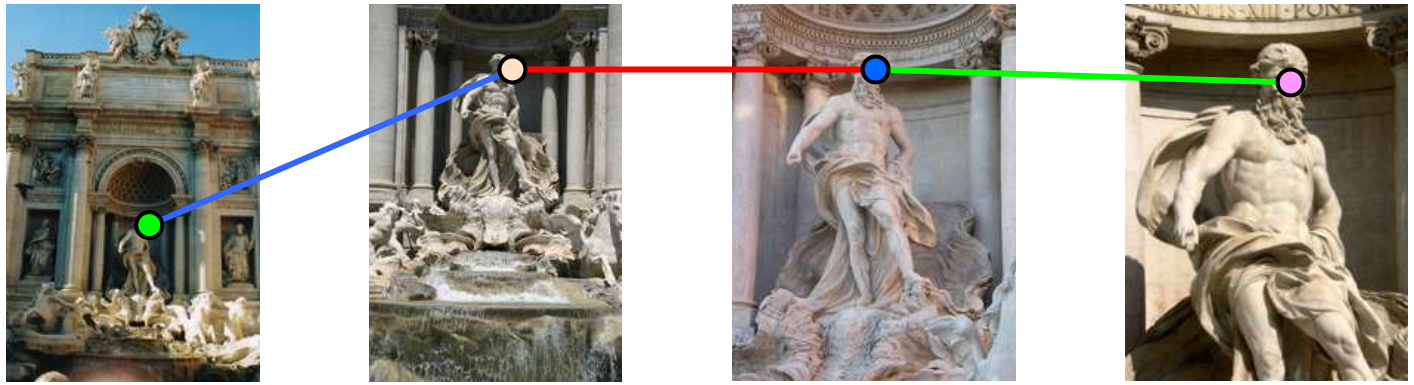
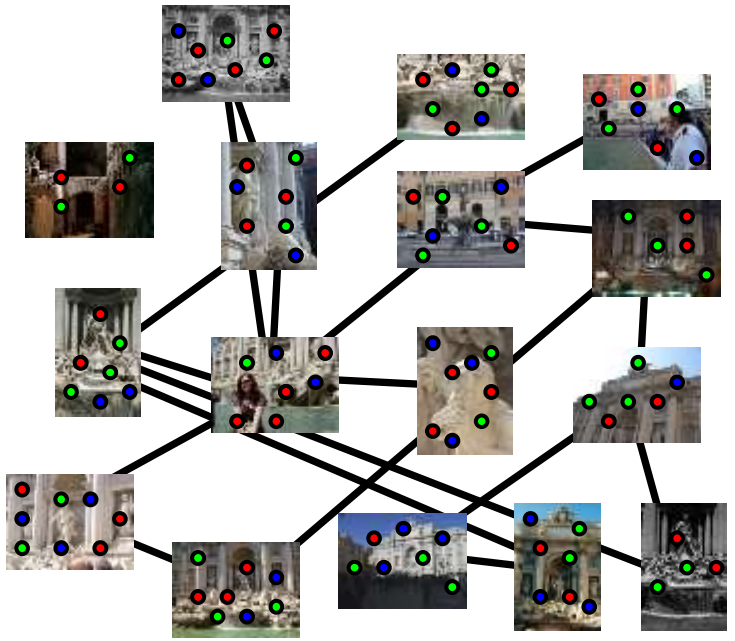


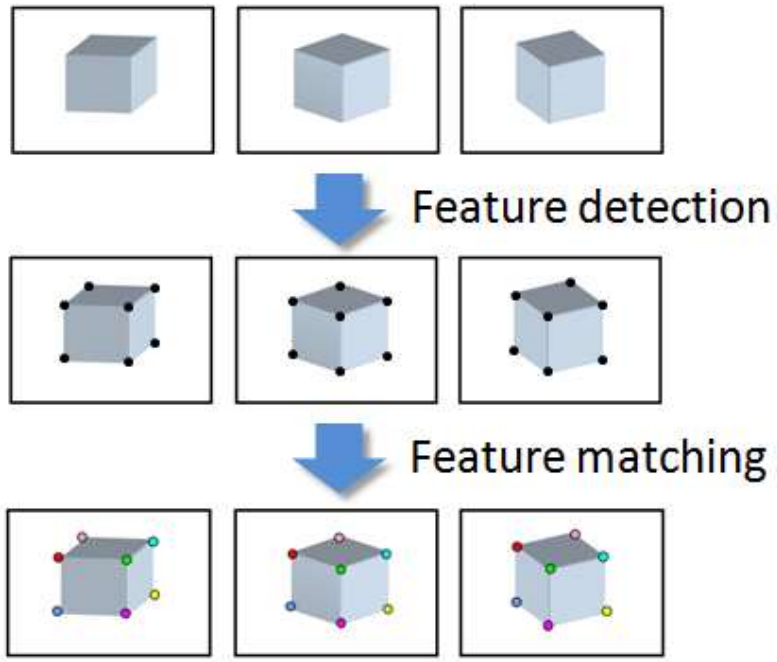
Image 1 Image 2 Image 3 Image 4

特征匹配

基于每对做特征匹配



计算场景结构



与二维图像整合方法一致

城镇级的三维重建

Reconstruction of Dubrovnik, Croatia, from ~40,000 images

场景与光照结合