lecture 3: local features and matching

deep learning for vision

Yannis Avrithis

Inria Rennes-Bretagne Atlantique

Rennes, Nov. 2018 – Jan. 2019

logistics

- planning available for course evaluation including written exam and oral presentations

oral presentations

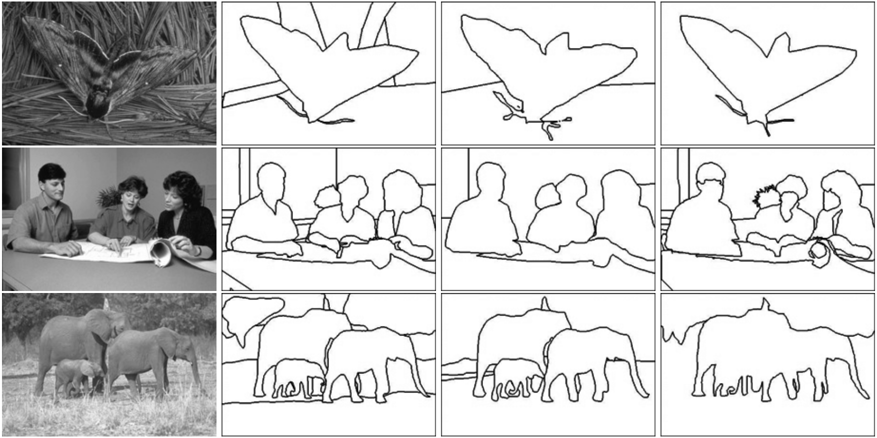
- teams of two
- instructions, paper list: `https://sif-dlv.github.io/oral`
- list is too long and "noisy"
- choose 2-5 papers, report your choice by December 16
- study and find more related work; find connections
- present on January 21
- focus presentation on ideas; not too detailed
- 8 min/talk, 4 min questions: total 20 min/team
- the class is your audience
- ask questions!

outline

derivatives
feature detection
spatial matching

derivatives

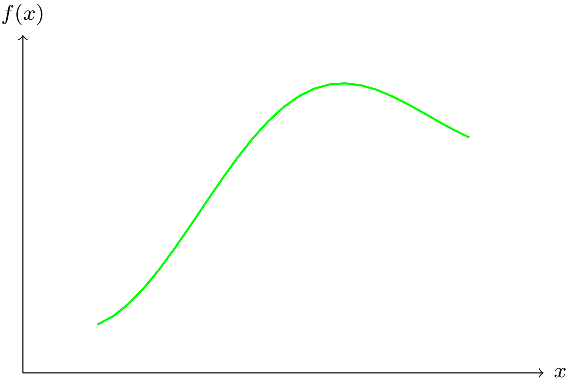
edges



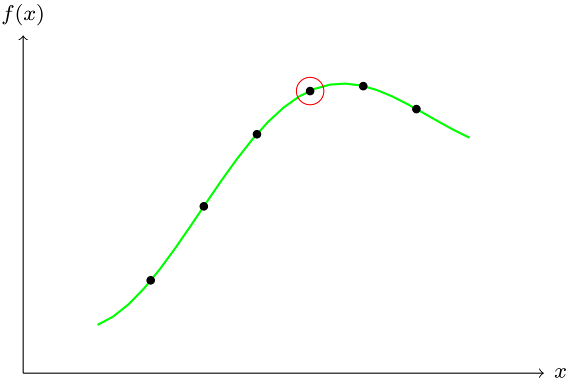
- connection between image recognition and segmentation
- database of human ‘ground truth’ to evaluate edge detection

Martin, Fowlkes, Tal, Malik. ICCV 2001. A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics.

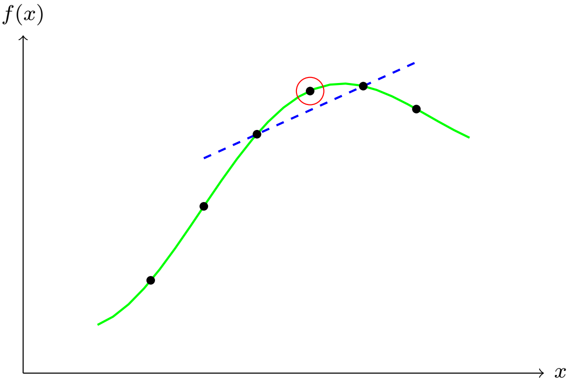
discrete derivative approximation



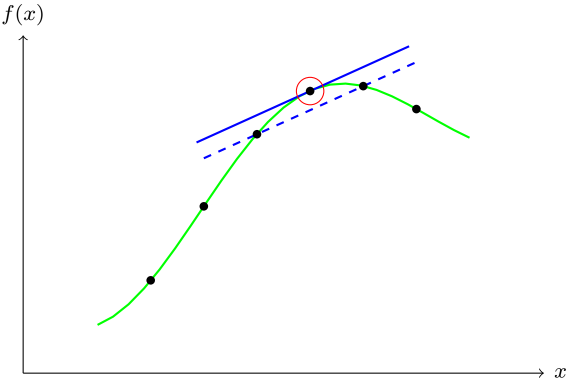
discrete derivative approximation



discrete derivative approximation

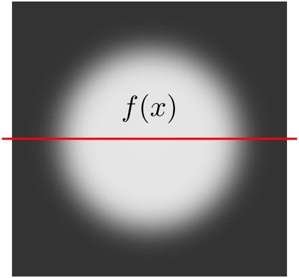


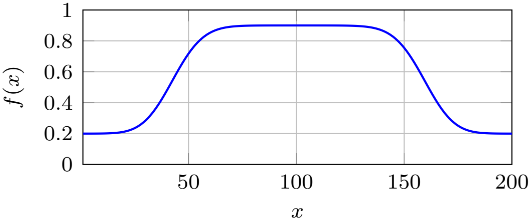
discrete derivative approximation



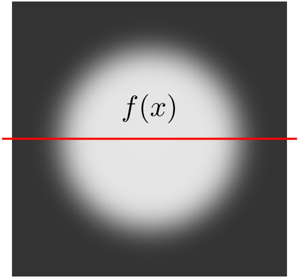
$$\frac{df}{dx}(x) \approx \frac{f(x+1) - f(x-1)}{2}$$

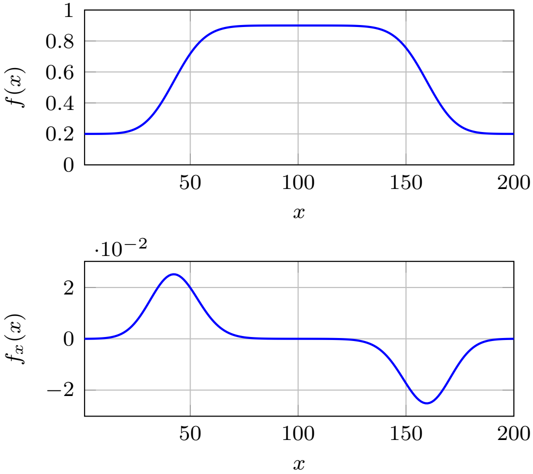
derivative in one dimension





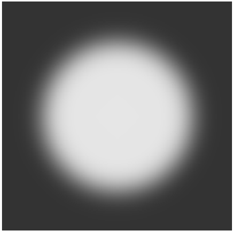
derivative in one dimension





$$f_x(x) := \frac{f(x+1) - f(x-1)}{2} = h * f, \quad h := \frac{1}{2}[1 \;\; 0 \;\; -1]$$

derivative in two dimensions: gradient

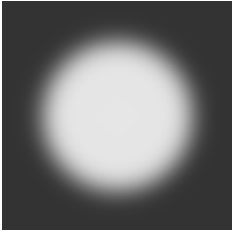


f

$$f_x := h_x * f$$
$$h_x := \tfrac{1}{2}[1 \;\; 0 \;\; -1]$$

$$f_y := h_y * f$$
$$h_y := \tfrac{1}{2}[1 \;\; 0 \;\; -1]^\top$$

derivative in two dimensions: gradient



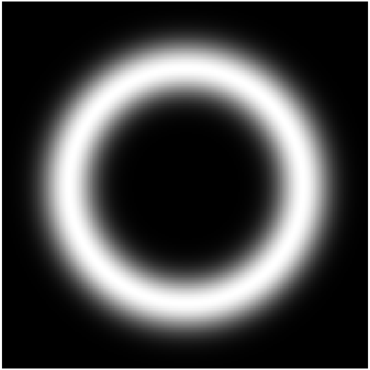
f

$\|(f_x, f_y)\|$

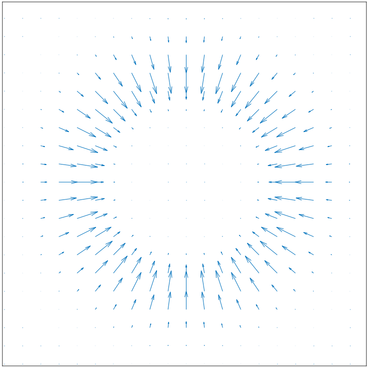
$f_x := h_x * f$
$h_x := \frac{1}{2}[1 \;\; 0 \;\; -1]$

$f_y := h_y * f$
$h_y := \frac{1}{2}[1 \;\; 0 \;\; -1]^\top$

gradient: magnitude and orientation



$\|(f_x, f_y)\|$

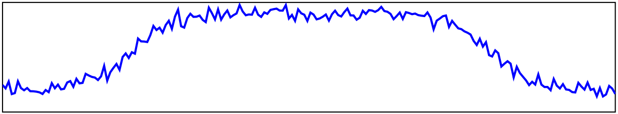


(f_x, f_y)

$$\nabla f(\mathbf{x}) := \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)(\mathbf{x}) \approx (h_x * f, h_y * f)(\mathbf{x}) = (f_x, f_y)(\mathbf{x})$$

noise

f

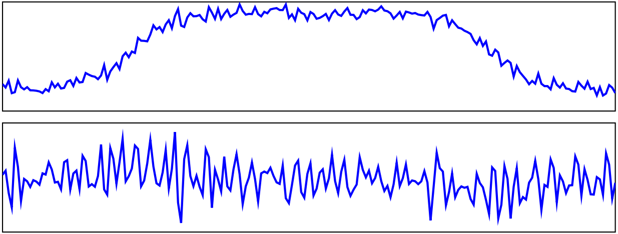


- Q: what happened to the edges?
- derivative is a high-pass filter: signal vanishes, noise remains

noise

f

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

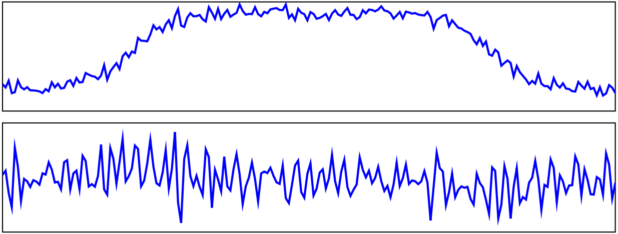


- Q: what happened to the edges?
- derivative is a high-pass filter: signal vanishes, noise remains

noise

f

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

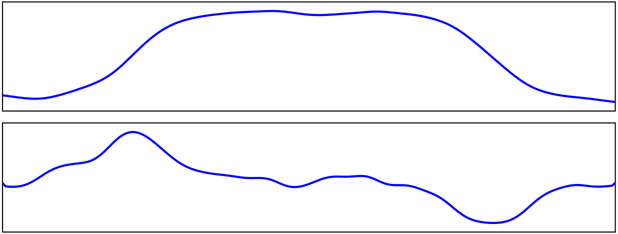


- Q: what happened to the edges?
- derivative is a high-pass filter: signal vanishes, noise remains

smoothing

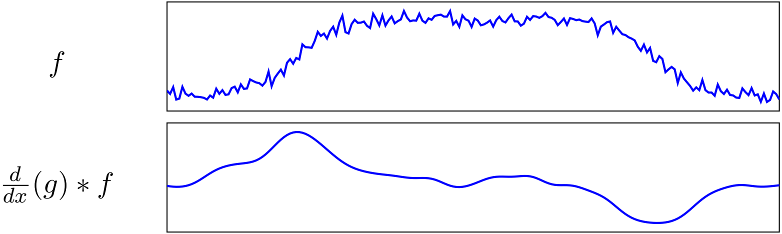
$g * f$

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



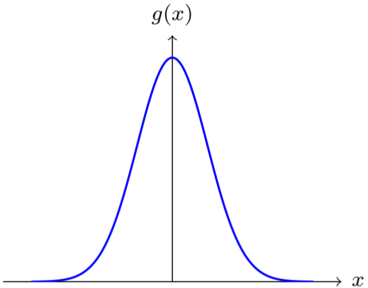
- smooth signal first
- that's better: edges recovered

filter derivative

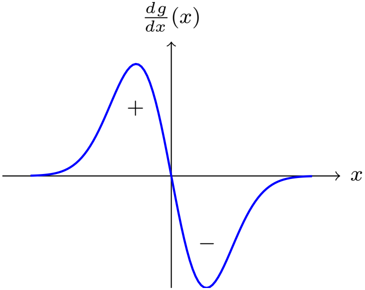


- this is equivalent to convolution with the filter derivative
- that's even better: filter is known in analytic form

1d Gaussian derivative



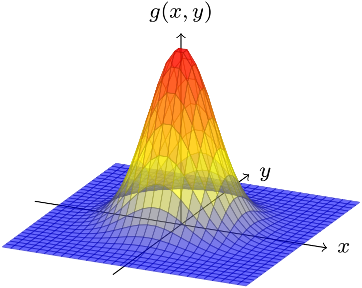
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

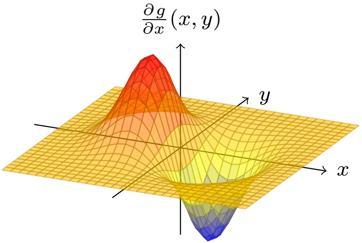


$$\frac{dg}{dx}(x) = -\frac{x}{\sigma^2} g(x)$$

- performs derivation and smoothing at the same time
- σ : "derivation scale"

2d Gaussian derivative





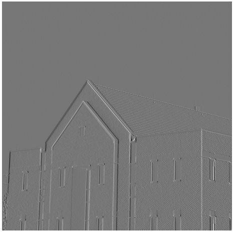
$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad g_x(x, y) := \frac{\partial g}{\partial x}(x, y) = -\frac{x}{\sigma^2} g(x, y)$$

- derivation in one direction, smoothing in both
- "derivative = convolution"

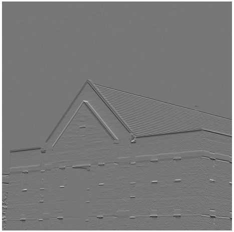
2d gradient

f

$\|(f_x, f_y)\|$



$f_x := h_x * f$



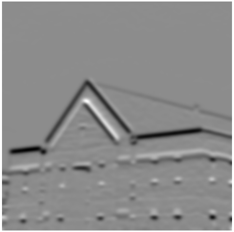
$f_y := h_y * f$

2d gradient by Gaussian derivative

f

$\|\nabla g * f\|$

$g_x * f$



$g_y * f$

why is gradient efficient comparing to Gabor?

- remember, the directional derivative of function f along vector $\mathbf{v}$ at point $\mathbf{x}$ is

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \mathbf{v} \cdot \nabla f(\mathbf{x}) = v_x \frac{\partial f}{\partial x}(\mathbf{x}) + v_y \frac{\partial f}{\partial y}(\mathbf{x})$$

- when $\mathbf{v}$ is a unit vector, the directional derivative is maximum when $\mathbf{v}$ points in the direction of the gradient
- does the same hold for the convolution with the Gaussian derivative?

why is gradient efficient comparing to Gabor?

- remember, the directional derivative of function f along vector $\mathbf{v}$ at point $\mathbf{x}$ is

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \mathbf{v} \cdot \nabla f(\mathbf{x}) = v_x \frac{\partial f}{\partial x}(\mathbf{x}) + v_y \frac{\partial f}{\partial y}(\mathbf{x})$$

- when $\mathbf{v}$ is a unit vector, the directional derivative is maximum when $\mathbf{v}$ points in the direction of the gradient
- does the same hold for the convolution with the Gaussian derivative?

why is gradient efficient comparing to Gabor?

- remember, the directional derivative of function f along vector $\mathbf{v}$ at point $\mathbf{x}$ is

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \mathbf{v} \cdot \nabla f(\mathbf{x}) = v_x \frac{\partial f}{\partial x}(\mathbf{x}) + v_y \frac{\partial f}{\partial y}(\mathbf{x})$$

- when $\mathbf{v}$ is a unit vector, the directional derivative is maximum when $\mathbf{v}$ points in the direction of the gradient
- does the same hold for the convolution with the Gaussian derivative?

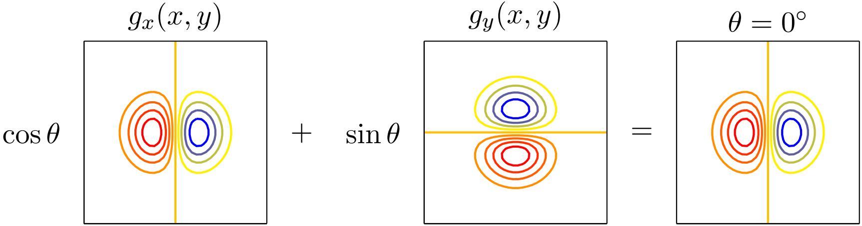
why is gradient efficient comparing to Gabor?

- remember, the directional derivative of function f along vector $\mathbf{v}$ at point $\mathbf{x}$ is

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \mathbf{v} \cdot \nabla f(\mathbf{x}) = v_x \frac{\partial f}{\partial x}(\mathbf{x}) + v_y \frac{\partial f}{\partial y}(\mathbf{x})$$

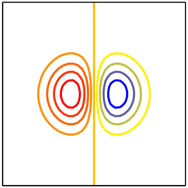
- when $\mathbf{v}$ is a unit vector, the directional derivative is maximum when $\mathbf{v}$ points in the direction of the gradient
- does the same hold for the convolution with the Gaussian derivative?

2d Gaussian derivative is steerable

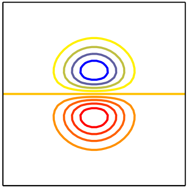


2d Gaussian derivative is steerable

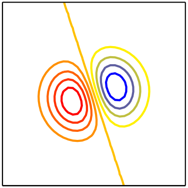
$g_x(x, y)$

$\cos\theta$  $+$ $\sin\theta$

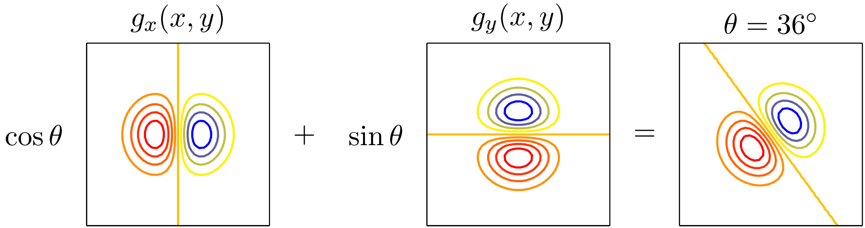
$g_y(x, y)$

 $=$

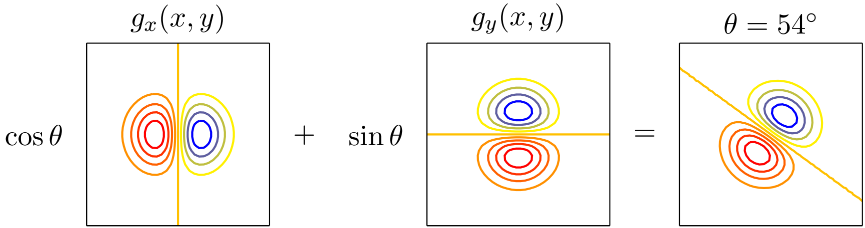
$\theta = 18°$



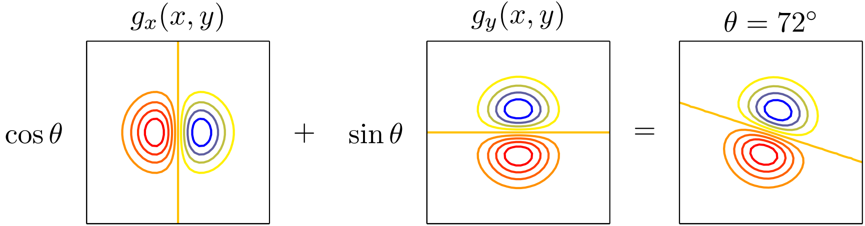
2d Gaussian derivative is steerable



2d Gaussian derivative is steerable



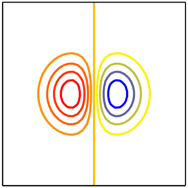
2d Gaussian derivative is steerable



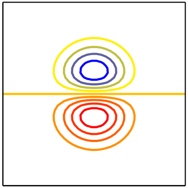
2d Gaussian derivative is steerable

$g_x(x, y)$ $\qquad$ $g_y(x, y)$ $\qquad$ $\theta = 90^\circ$

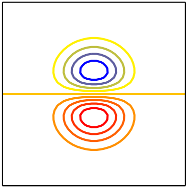
$\cos\theta$



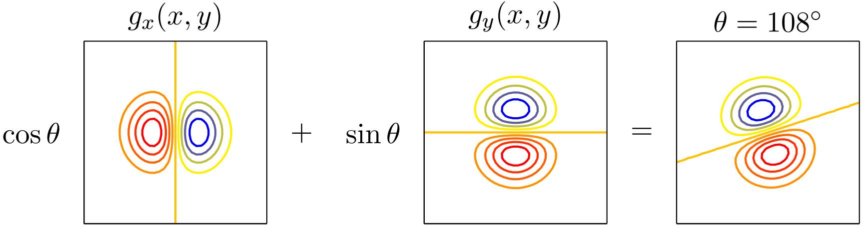
$+ \; \sin\theta$



$=$



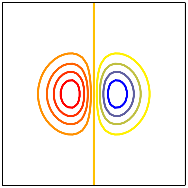
2d Gaussian derivative is steerable



2d Gaussian derivative is steerable

$g_x(x, y)$

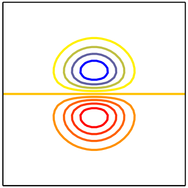
$\cos\theta$



$+$

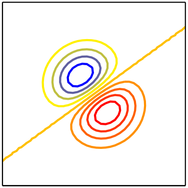
$g_y(x, y)$

$\sin\theta$

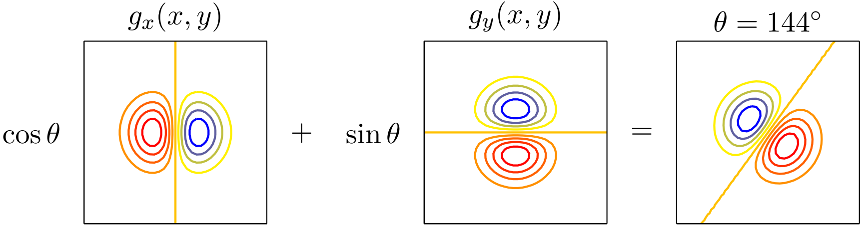


$=$

$\theta = 126^\circ$



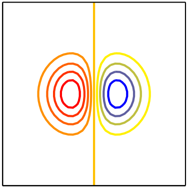
2d Gaussian derivative is steerable



2d Gaussian derivative is steerable

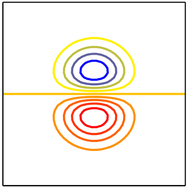
$g_x(x, y)$

$\cos\theta$



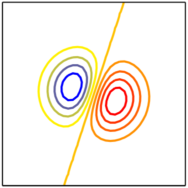
$+ \quad \sin\theta$

$g_y(x, y)$



$=$

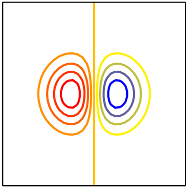
$\theta = 162^\circ$



2d Gaussian derivative is steerable

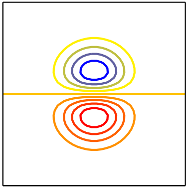
$\cos\theta$

$g_x(x, y)$



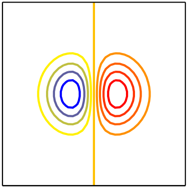
$+$ $\sin\theta$

$g_y(x, y)$



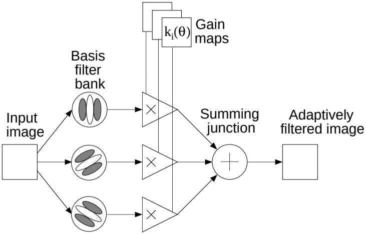
$=$

$\theta = 180^\circ$

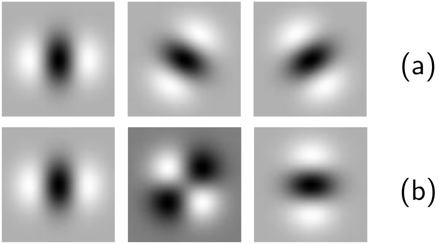


steerable filter

[Freeman and Adelson 1991]



architecture

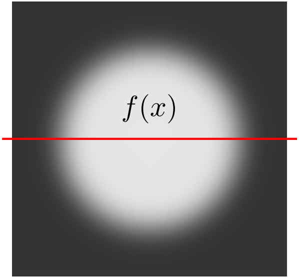


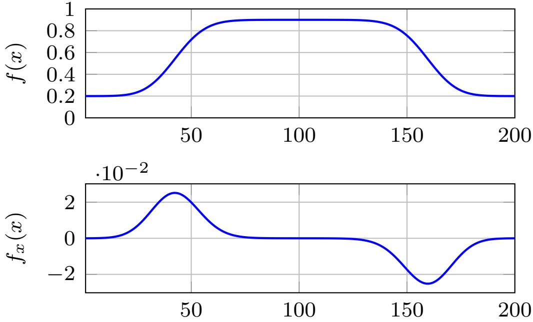
basis sets of $\frac{\partial^2 g}{\partial x^2}$

- an orientation-selective filter that can be expressed as a linear combination of a small basis set of filters
- the basis set can be (a) a set of rotated versions of itself, or (b) a set of separable filters

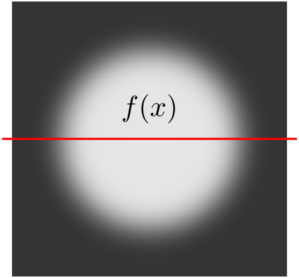
Freeman and Adelson. PAMI 1991. The Design and Use of Steerable Filters.

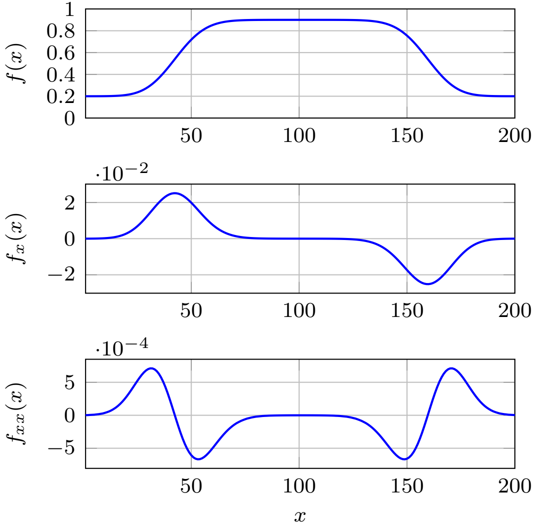
second derivative in one dimension





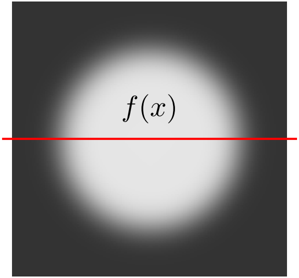
second derivative in one dimension

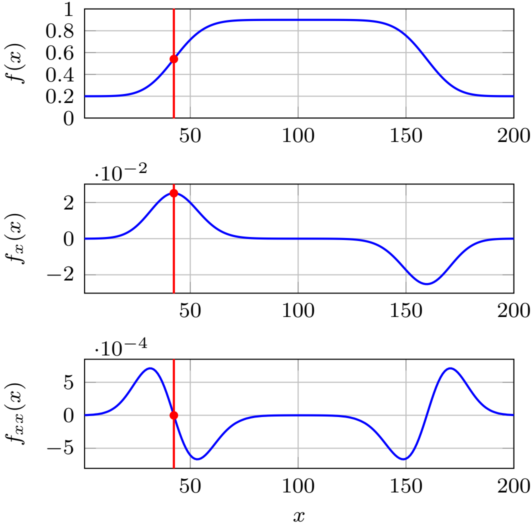




$$f_{xx}(x) := \frac{f(x-1) - 2f(x) + f(x+1)}{4} = h * f, \quad h := \frac{1}{4}[1 \;\; -2 \;\; 1]$$

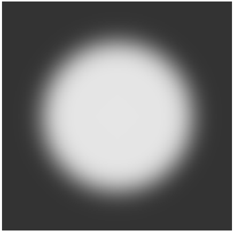
second derivative in one dimension





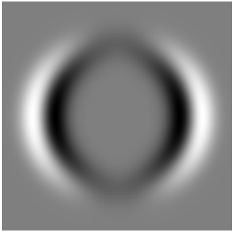
$$f_{xx}(x) := \frac{f(x-1) - 2f(x) + f(x+1)}{4} = h * f, \quad h := \frac{1}{4}[1 \;\; -2 \;\; 1]$$

second derivative in two dimensions: Laplacian

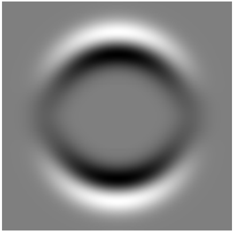


f

$f_{xx} + f_{yy}$



$$f_{xx} := h_{xx} * f$$
$$h_{xx} := \frac{1}{4}[1 \;\; -2 \;\; 1]$$



$$f_{yy} := h_{yy} * f$$
$$h_y := \frac{1}{4}[1 \;\; -2 \;\; 1]^\top$$

Laplacian operator

- discrete approximation

$$h_{xx} := \frac{1}{4}[1 \;\; -2 \;\; 1]$$
$$h_{yy} := \frac{1}{4}[1 \;\; -2 \;\; 1]^\top$$
$$h_L := h_{xx} + h_{yy} = \frac{1}{4}\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

- differential operator

$$\nabla^2 f(\mathbf{x}) := \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right)(\mathbf{x})$$
$$\approx (h_{xx} * f + h_{yy} * f)(\mathbf{x}) = (f_{xx} + f_{yy})(\mathbf{x})$$

Laplacian operator

- discrete approximation

$$h_{xx} := \frac{1}{4}[1 \ \ -2 \ 1]$$

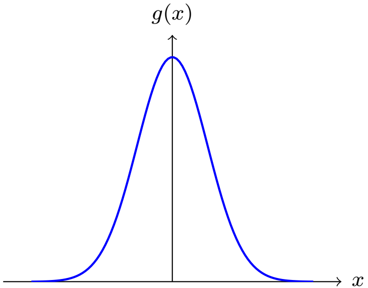
$$h_{yy} := \frac{1}{4}[1 \ \ -2 \ 1]^\top$$

$$h_L := h_{xx} + h_{yy} = \frac{1}{4}\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

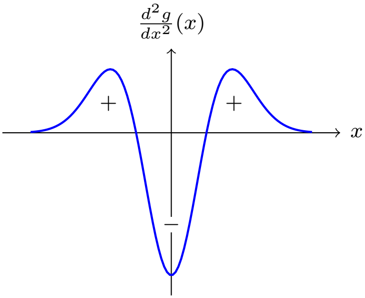
- differential operator

$$\begin{aligned} \nabla^2 f(\mathbf{x}) &:= \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)(\mathbf{x}) \\ &\approx (h_{xx} * f + h_{yy} * f)(\mathbf{x}) = (f_{xx} + f_{yy})(\mathbf{x}) \end{aligned}$$

1d Gaussian second derivative



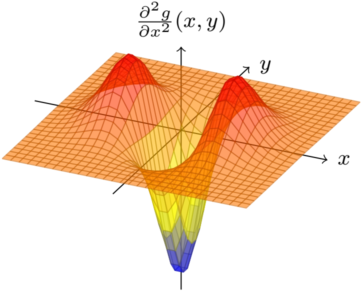
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

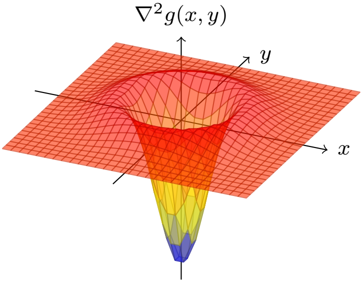


$$\frac{d^2g}{dx^2}(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) g(x)$$

- "center-surround" operator

2d Laplacian of Gaussian (LoG)





$$\frac{\partial^2 g}{\partial x^2}(x, y) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2} \right) g(x, y) \qquad \nabla^2 g(x, y) := \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right) (x, y)$$

- rotationally symmetric
- “mexican hat”

edge detection

f

$L_0(\nabla^2 g * f)$

$\|\nabla g * f\|$

$\nabla^2 g * f$

edge detection

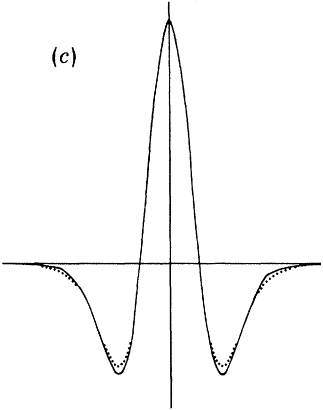
$$L_0(\nabla^2 g * f)$$

edge detection

$$L_0(\nabla^2 g * f)\|\nabla g * f\|$$

difference of Gaussians (DoG)

[Marr and Hildreth 1980]



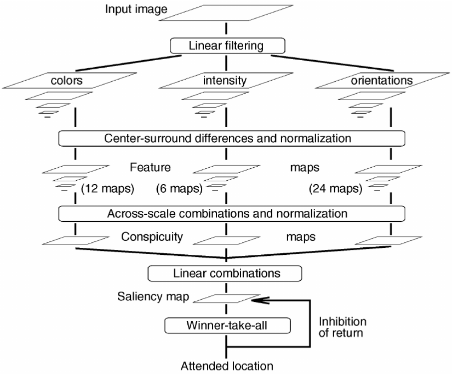
- studied the $\nabla^2 g$ operator as a model of retinal X-cells
- popularized it as a computational theory of edge detection
- hypothesized a biological implementation as a difference of Gaussians with $\sigma_1/\sigma_2 \approx 1.6$

Marr and Hildreth. RSL 1980. Theory of Edge Detection.

feature detection

saliency and visual attention

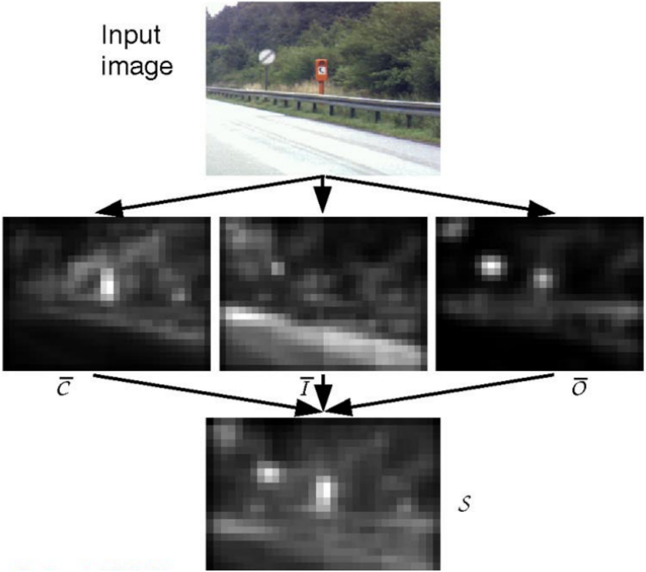
[Itti et al. 1998]



- visual attention system, inspired by the early primate visual system
- multiple scales, multiple features, center-surround, normalization and winner-take-all operations

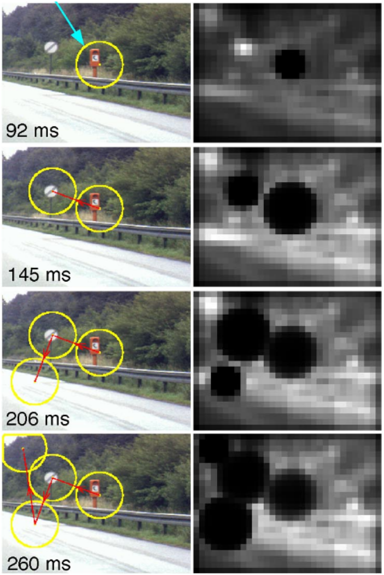
Itti, Koch and Niebur. PAMI 1998. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis.

saliency and visual attention



Itti, Koch and Niebur. PAMI 1998. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis.

saliency and visual attention



Itti, Koch and Niebur. PAMI 1998. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis.

scale change

scale change

scale change

scale change

scale change

scale change

scale change

- for every scale factor s, and for every point $\mathbf{x}$, the scaled image f' at the scaled point $\mathbf{x}' := s\mathbf{x}$ equals the original image f at the original point $\mathbf{x}$

$$f'(\mathbf{x}') = f'(s\mathbf{x}) = f(\mathbf{x})$$

scale space

scale space

scale space

scale space

scale space

scale space

scale space

[Witkin 1983]

- the scale-space F of f at point $\mathbf{x}$ and scale σ, and its n-th derivative with respect to some variable x, are defined as

$$F(\mathbf{x};\sigma) := [g(\cdot;\sigma) * f](\mathbf{x})$$

$$F_{x^n}(\mathbf{x};\sigma) := \frac{\partial^n F}{\partial x^n}(\mathbf{x};\sigma) = \left[\frac{\partial^n g}{\partial x^n}(\cdot;\sigma) * f\right](\mathbf{x})$$

- gradient

$$\nabla F \approx (F_x, F_y)$$

- Laplacian

$$\nabla^2 F \approx F_{xx} + F_{yy}$$

- we write derivatives but we only compute convolutions

scale space

[Witkin 1983]

- the scale-space F of f at point $\mathbf{x}$ and scale σ, and its n-th derivative with respect to some variable x, are defined as

$$F(\mathbf{x};\sigma) := [g(\cdot;\sigma) * f](\mathbf{x})$$
$$F_{x^n}(\mathbf{x};\sigma) := \frac{\partial^n F}{\partial x^n}(\mathbf{x};\sigma) = \left[\frac{\partial^n g}{\partial x^n}(\cdot;\sigma) * f\right](\mathbf{x})$$

- gradient

$$\nabla F \approx (F_x, F_y)$$

- Laplacian

$$\nabla^2 F \approx F_{xx} + F_{yy}$$

- we write derivatives but we only compute convolutions

Witkin. IJCAI 1983. Scale-Space Filtering.

scale space under scaling

[Witkin 1983]

- for every scale factor s, for every point $\mathbf{x}$, and for every scale σ, the scale-space F' at the point $\mathbf{x}' := s\mathbf{x}$ and scale $\sigma' := s\sigma$ equals the original scale-space F at the original point $\mathbf{x}$ and scale σ:

$$F'(\mathbf{x}'; \sigma') = F'(s\mathbf{x}, s\sigma) = F(\mathbf{x}; \sigma)$$

and we would like the same for their derivatives

Witkin. IJCAI 1983. Scale-Space Filtering.

scale-normalized derivatives

[Lindeberg 1998]

- remember, however,

$$\frac{dg}{dx}(x;\sigma) = -\frac{x}{\sigma^2} g(x;\sigma) \qquad \frac{d^2 g}{dx^2}(x;\sigma) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) g(x;\sigma)$$

$$F'_{x'}(\mathbf{x}';\sigma') = s^{-1} F_x(\mathbf{x};\sigma) \qquad F'_{x'x'}(\mathbf{x}';\sigma') = s^{-2} F_{xx}(\mathbf{x};\sigma)$$

- in general, we only have

$$F'_{x'^n}(\mathbf{x}';\sigma') = s^{-n} F_{x^n}(\mathbf{x};\sigma)$$

- solution: we normalize the n-th order derivative by σ^n

$$\hat{F}_{x^n}(\mathbf{x};\sigma) := \sigma^n F_{x^n}(\mathbf{x};\sigma) = \sigma^n \frac{\partial^n g}{\partial x^n}(\mathbf{x};\sigma) * f(\mathbf{x})$$

- then, indeed

$$\hat{F}'_{x'^n}(\mathbf{x}';\sigma') = \hat{F}_{x^n}(\mathbf{x};\sigma)$$

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

scale-normalized derivatives

[Lindeberg 1998]

- remember, however,

$$\frac{dg}{dx}(x;\sigma) = -\frac{x}{\sigma^2} g(x;\sigma) \qquad \frac{d^2 g}{dx^2}(x;\sigma) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) g(x;\sigma)$$

$$F'_{x'}(\mathbf{x}';\sigma') = s^{-1} F_x(\mathbf{x};\sigma) \qquad F'_{x'x'}(\mathbf{x}';\sigma') = s^{-2} F_{xx}(\mathbf{x};\sigma)$$

- in general, we only have

$$F'_{x'^n}(\mathbf{x}';\sigma') = s^{-n} F_{x^n}(\mathbf{x};\sigma)$$

- solution: we normalize the n-th order derivative by σ^n

$$\hat{F}_{x^n}(\mathbf{x};\sigma) := \sigma^n F_{x^n}(\mathbf{x};\sigma) = \sigma^n \frac{\partial^n g}{\partial x^n}(\mathbf{x};\sigma) * f(\mathbf{x})$$

- then, indeed

$$\hat{F}'_{x'^n}(\mathbf{x}';\sigma') = \hat{F}_{x^n}(\mathbf{x};\sigma)$$

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

scale-normalized derivatives

[Lindeberg 1998]

- remember, however,

$$\frac{dg}{dx}(x;\sigma) = -\frac{x}{\sigma^2} g(x;\sigma) \qquad \frac{d^2 g}{dx^2}(x;\sigma) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) g(x;\sigma)$$

$$F'_{x'}(\mathbf{x}';\sigma') = s^{-1} F_x(\mathbf{x};\sigma) \qquad F'_{x'x'}(\mathbf{x}';\sigma') = s^{-2} F_{xx}(\mathbf{x};\sigma)$$

- in general, we only have

$$F'_{x'^n}(\mathbf{x}';\sigma') = s^{-n} F_{x^n}(\mathbf{x};\sigma)$$

- solution: we normalize the n-th order derivative by σ^n

$$\hat{F}_{x^n}(\mathbf{x};\sigma) := \sigma^n F_{x^n}(\mathbf{x};\sigma) = \sigma^n \frac{\partial^n g}{\partial x^n}(\mathbf{x};\sigma) * f(\mathbf{x})$$

- then, indeed

$$\hat{F}'_{x'^n}(\mathbf{x}';\sigma') = \hat{F}_{x^n}(\mathbf{x};\sigma)$$

scale-normalized derivatives

[Lindeberg 1998]

- remember, however,

$$\frac{dg}{dx}(x;\sigma) = -\frac{x}{\sigma^2} g(x;\sigma) \qquad \frac{d^2 g}{dx^2}(x;\sigma) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) g(x;\sigma)$$

$$F'_{x'}(\mathbf{x}';\sigma') = s^{-1} F_x(\mathbf{x};\sigma) \qquad F'_{x'x'}(\mathbf{x}';\sigma') = s^{-2} F_{xx}(\mathbf{x};\sigma)$$

- in general, we only have

$$F'_{x'^n}(\mathbf{x}';\sigma') = s^{-n} F_{x^n}(\mathbf{x};\sigma)$$

- solution: we normalize the n-th order derivative by σ^n

$$\hat{F}_{x^n}(\mathbf{x};\sigma) := \sigma^n F_{x^n}(\mathbf{x};\sigma) = \sigma^n \frac{\partial^n g}{\partial x^n}(\mathbf{x};\sigma) * f(\mathbf{x})$$

- then, indeed

$$\hat{F}'_{x'^n}(\mathbf{x}';\sigma') = \hat{F}_{x^n}(\mathbf{x};\sigma)$$

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

normalized Laplacian and scale selection

- normalized Laplacian operator

$$\hat{\nabla}^2 F(\mathbf{x};\sigma) := \sigma^2 \nabla^2 F(\mathbf{x};\sigma) \approx \sigma^2 (F_{xx} + F_{yy})(\mathbf{x};\sigma)$$

- scale selection

$$\mathrm{scale}(\mathbf{x}) := \arg\max_{\sigma} |\hat{\nabla}^2 F(\mathbf{x};\sigma)|$$

$$\sigma^2 \frac{d^2 g}{dx^2}(x;\sigma) = \left(\frac{x^2}{\sigma^2} - 1\right) g(x;\sigma)$$

- let's try a blob centered at the origin, filter by a normalized LoG of varying scale σ, and measure the response at the origin

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

normalized Laplacian and scale selection

- normalized Laplacian operator

$$\hat{\nabla}^2 F(\mathbf{x};\sigma) := \sigma^2 \nabla^2 F(\mathbf{x};\sigma) \approx \sigma^2 (F_{xx} + F_{yy})(\mathbf{x};\sigma)$$

- scale selection

$$\mathrm{scale}(\mathbf{x}) := \arg\max_{\sigma} |\hat{\nabla}^2 F(\mathbf{x};\sigma)|$$

$$\sigma^2 \frac{d^2 g}{dx^2}(x;\sigma) = \left(\frac{x^2}{\sigma^2} - 1\right) g(x;\sigma)$$

- let's try a blob centered at the origin, filter by a normalized LoG of varying scale σ, and measure the response at the origin

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

normalized Laplacian and scale selection

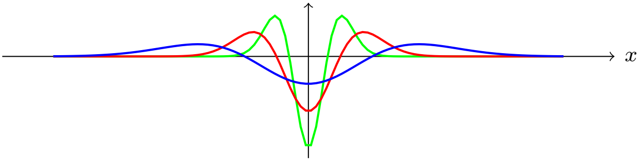
- normalized Laplacian operator

$$\hat{\nabla}^2 F(\mathbf{x};\sigma) := \sigma^2 \nabla^2 F(\mathbf{x};\sigma) \approx \sigma^2 (F_{xx} + F_{yy})(\mathbf{x};\sigma)$$

- scale selection

$$\mathrm{scale}(\mathbf{x}) := \arg\max_{\sigma} |\hat{\nabla}^2 F(\mathbf{x};\sigma)|$$

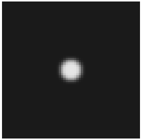
$$\sigma^2 \frac{d^2 g}{dx^2}(x;\sigma) = (\frac{x^2}{\sigma^2} - 1) g(x;\sigma)$$

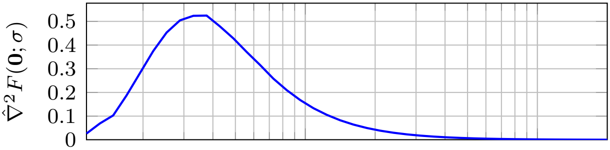


- let's try a blob centered at the origin, filter by a normalized LoG of varying scale σ, and measure the response at the origin

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

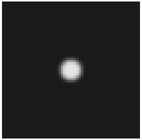
normalized Laplacian and scale selection

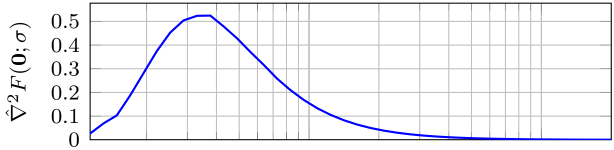


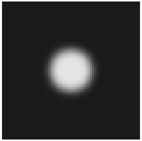


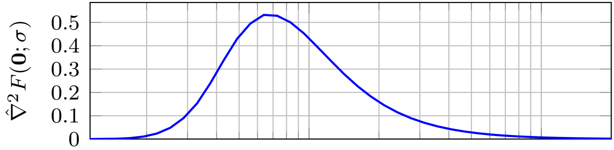
Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

normalized Laplacian and scale selection



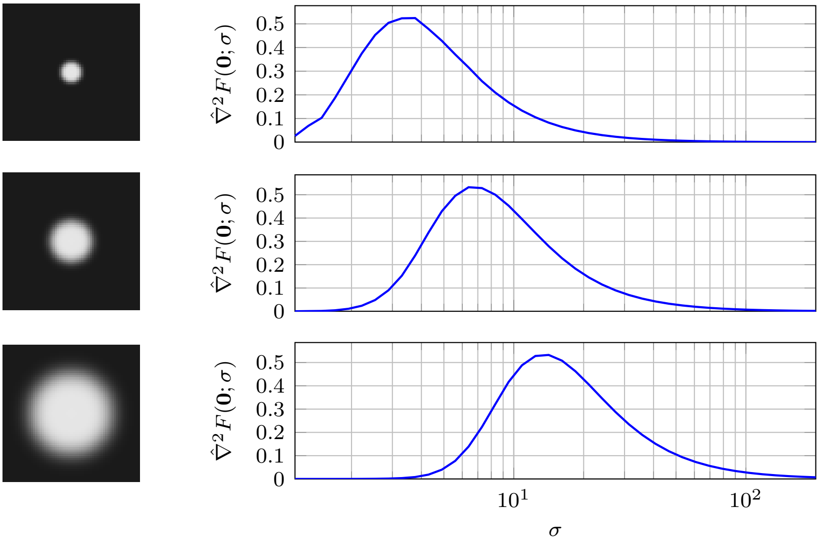






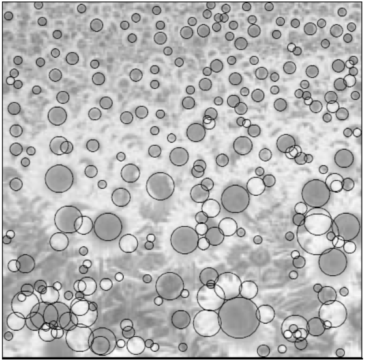
Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

normalized Laplacian and scale selection



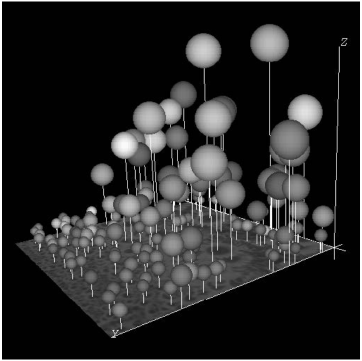
Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

blob detection



- convolution with a circular symmetric center-surround pattern in scale-space
- local maxima in scale-space yield positions and scales of blobs

blob detection



- convolution with a circular symmetric center-surround pattern in scale-space
- local maxima in scale-space yield positions and scales of blobs

Lindeberg. IJCV 1998. Feature Detection with Automatic Scale Selection.

difference of Gaussians

- Gaussian satisfies heat equation (try it!), hence finite difference approximation to $\frac{\partial g}{\partial \sigma}$ can be used

$$\sigma \nabla^2 g = \frac{\partial g}{\partial \sigma} \approx \frac{g(\mathbf{x}; k\sigma) - g(\mathbf{x}; \sigma)}{k\sigma - \sigma}$$

- then, difference of Gaussians approximates its normalized Laplacian

$$g(\mathbf{x}; k\sigma) - g(\mathbf{x}; \sigma) \approx (k-1)\sigma^2 \nabla^2 g,$$

incorporating scale normalization

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

difference of Gaussians

- Gaussian satisfies heat equation (try it!), hence finite difference approximation to $\frac{\partial g}{\partial \sigma}$ can be used

$$\sigma \nabla^2 g = \frac{\partial g}{\partial \sigma} \approx \frac{g(\mathbf{x}; k\sigma) - g(\mathbf{x}; \sigma)}{k\sigma - \sigma}$$

- then, difference of Gaussians approximates its normalized Laplacian

$$g(\mathbf{x}; k\sigma) - g(\mathbf{x}; \sigma) \approx (k-1)\sigma^2 \nabla^2 g,$$

incorporating scale normalization

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

difference of Gaussians

- Gaussian satisfies heat equation (try it!), hence finite difference approximation to $\frac{\partial g}{\partial \sigma}$ can be used

$$\sigma \nabla^2 g = \frac{\partial g}{\partial \sigma} \approx \frac{g(\mathbf{x}; k\sigma) - g(\mathbf{x}; \sigma)}{k\sigma - \sigma}$$

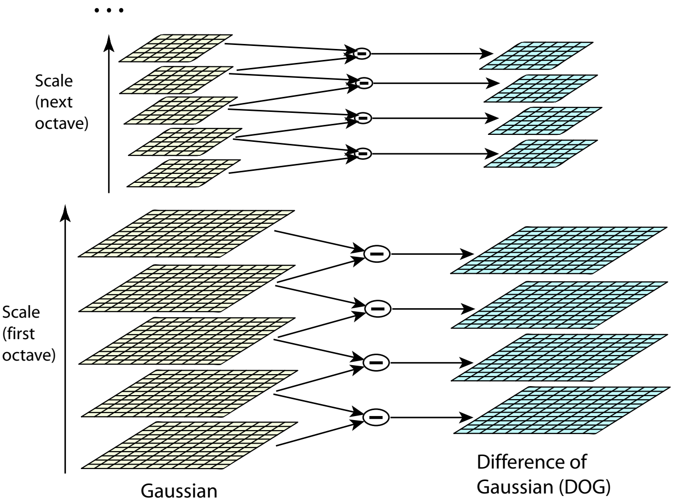
- then, difference of Gaussians approximates its normalized Laplacian

$$g(\mathbf{x}; k\sigma) - g(\mathbf{x}; \sigma) \approx (k-1)\sigma^2 \nabla^2 g,$$

incorporating scale normalization

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

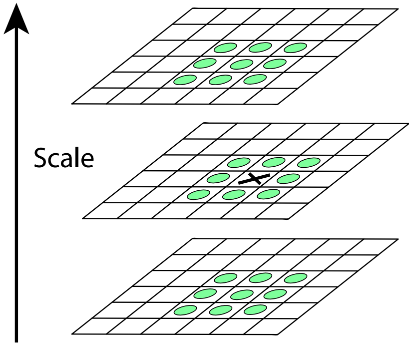
scale-space computation



- incrementally convolve with Gaussian, subsample at each octave

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

scale-space local extrema

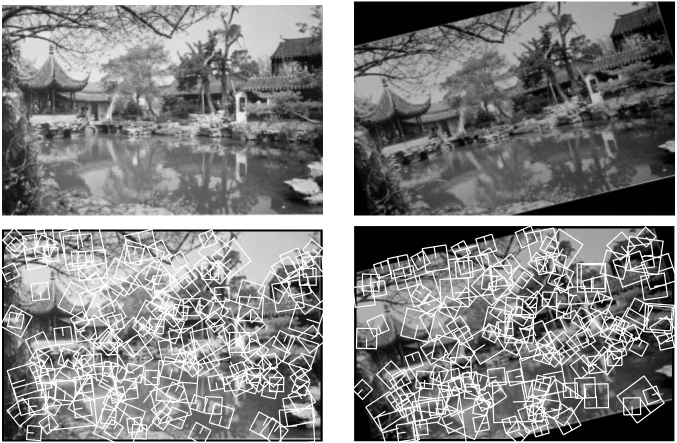


- local maxima among 26 neighbors selected
- accurately localized, edge responses rejected, orientation normalized

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

scale-invariant feature transform (SIFT)

[Lowe 1999]



- detected patches equivariant to translation, scale and rotation

Lowe. ICCV 1999. Object recognition from local scale-invariant features.

desired properties of local features

- **repeatable**: in a transformed image, the same feature is detected at a transformed position
- **distinctive**: different image features can be discriminated by their local appearance
- **localized**: relatively small regions, robust to occlusion

− *elongated*: edges, ridges

+ *isotropic*: blobs, extremal regions

+ *points*: corners and junctions

desired properties of local features

- **repeatable**: in a transformed image, the same feature is detected at a transformed position
- **distinctive**: different image features can be discriminated by their local appearance
- **localized**: relatively small regions, robust to occlusion

− *elongated*: edges, ridges

+ *isotropic*: blobs, extremal regions

+ *points*: corners and junctions

the Hessian matrix

- defined as

$$\hat{H}F(\mathbf{x},\sigma) := \sigma^2 \begin{pmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{pmatrix} (\mathbf{x},\sigma)$$

- the Laplacian is just its trace

$$\hat{\nabla}^2 F(\mathbf{x},\sigma) = \sigma^2 (F_{xx} + F_{yy})(\mathbf{x},\sigma) = \operatorname{tr} \hat{H}F(\mathbf{x},\sigma)$$

- where gradient magnitude is zero, f is locally maximized (concave), minimized (convex), flat, or has a saddle point depending on eigenvalues λ_1, λ_2 of the Hessian
- good for blobs: maximum for $\lambda_1, \lambda_2 < 0$, minimum for $\lambda_1, \lambda_2 > 0$
- however, still fires on edges

the Hessian matrix

- defined as

$$\hat{H}F(\mathbf{x},\sigma) := \sigma^2 \begin{pmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{pmatrix} (\mathbf{x},\sigma)$$

- the Laplacian is just its trace

$$\hat{\nabla}^2 F(\mathbf{x},\sigma) = \sigma^2 (F_{xx} + F_{yy})(\mathbf{x},\sigma) = \operatorname{tr} \hat{H}F(\mathbf{x},\sigma)$$

- where gradient magnitude is zero, f is locally maximized (concave), minimized (convex), flat, or has a saddle point depending on eigenvalues λ_1, λ_2 of the Hessian
- good for blobs: maximum for $\lambda_1, \lambda_2 < 0$, minimum for $\lambda_1, \lambda_2 > 0$
- however, still fires on edges

the Hessian matrix

- defined as

$$\hat{H} F(\mathbf{x}, \sigma) := \sigma^2 \begin{pmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{pmatrix} (\mathbf{x}, \sigma)$$

- the Laplacian is just its trace

$$\hat{\nabla}^2 F(\mathbf{x}, \sigma) = \sigma^2 (F_{xx} + F_{yy})(\mathbf{x}, \sigma) = \operatorname{tr} \hat{H} F(\mathbf{x}, \sigma)$$

- where gradient magnitude is zero, f is locally maximized (concave), minimized (convex), flat, or has a saddle point depending on eigenvalues λ_1, λ_2 of the Hessian
- good for blobs: maximum for $\lambda_1, \lambda_2 < 0$, minimum for $\lambda_1, \lambda_2 > 0$
- however, still fires on edges

the (windowed) second moment matrix

[Förstner 1986]

- defined as

$$\begin{aligned}\hat{\mu}F(\mathbf{x},\sigma) &:= w * \sigma^2 (\nabla F)(\nabla F)^\top(\mathbf{x},\sigma)\\ &= w * \sigma^2 \begin{pmatrix} F_x^2 & F_xF_y \\ F_xF_y & F_y^2 \end{pmatrix}(\mathbf{x},\sigma)\end{aligned}$$

  where w is another Gaussian at some higher integration scale; σ is called the derivation scale

- the (windowed) gradient is just its trace

$$w * \|\hat{\nabla} F(\mathbf{x},\sigma)\|^2 = w * \sigma^2 (F_x^2 + F_y^2)(\mathbf{x},\sigma) = \operatorname{tr} \hat{\mu}F(\mathbf{x},\sigma)$$

- good for edges, corners and junctions; again, depending on the eigenvalues $\lambda_1 \geq \lambda_2$

Förstner 1986. A Feature Based Correspondence Algorithm for Image Processing.

the (windowed) second moment matrix

[Förstner 1986]

- defined as

$$\begin{aligned}\hat{\mu}F(\mathbf{x},\sigma) &:= w * \sigma^2 (\nabla F)(\nabla F)^\top (\mathbf{x},\sigma) \\ &= w * \sigma^2 \begin{pmatrix} F_x^2 & F_x F_y \\ F_x F_y & F_y^2 \end{pmatrix} (\mathbf{x},\sigma)\end{aligned}$$

  where w is another Gaussian at some higher integration scale; σ is called the derivation scale

- the (windowed) gradient is just its trace

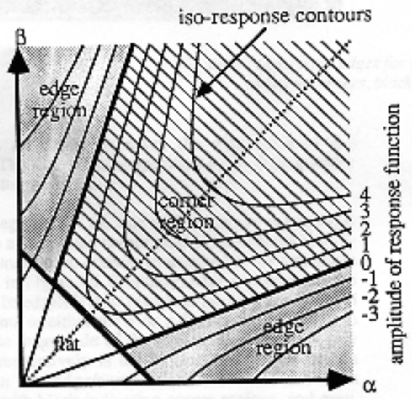
$$w * \|\hat{\nabla} F(\mathbf{x},\sigma)\|^2 = w * \sigma^2 (F_x^2 + F_y^2)(\mathbf{x},\sigma) = \operatorname{tr} \hat{\mu}F(\mathbf{x},\sigma)$$

- good for edges, corners and junctions; again, depending on the eigenvalues $\lambda_1 \geq \lambda_2$

Förstner 1986. A Feature Based Correspondence Algorithm for Image Processing.

Harris corners

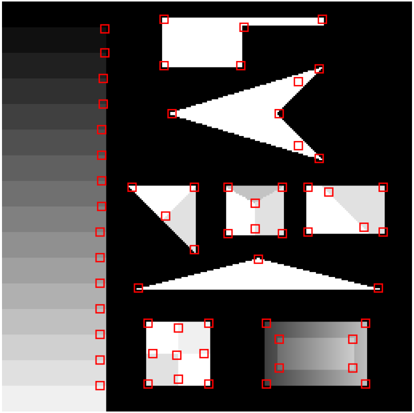
[Harris and Stevens 1988]



- if trace $\lambda_1 + \lambda_2$ is too low $\rightarrow$ flat
- if condition number λ_1 / λ_2 is too high $\rightarrow$ edge
- response function $r(\mu) = \det \mu - k \operatorname{tr}^2 \mu$

Harris and Stephens AVC 1988. A Combined Corner and Edge Detector.

Harris corners (and junctions)



corners

response

- response: positive on corners, negative on edges, zero otherwise
- detection: non-maxima suppression and thresholding

Harris and Stephens AVC 1988. A Combined Corner and Edge Detector.

motivation: local autocorrelation

- assume f is differentiable and ignore scale space
- assume an image patch at the origin defined by window w; how much does it change when we shift by $\mathbf{t}$?

$$E(\mathbf{t}) = \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - f(\mathbf{x}))^2$$

- quadratic form defined by $\mu = w * (\nabla f)(\nabla f)^\top$

Harris and Stephens AVC 1988. A Combined Corner and Edge Detector.

motivation: local autocorrelation

- assume f is differentiable and ignore scale space
- assume an image patch at the origin defined by window w; how much does it change when we shift by $\mathbf{t}$?

$$\begin{aligned} E(\mathbf{t}) &= \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - f(\mathbf{x}))^2 \\ &\approx \sum_{\mathbf{x}} w(\mathbf{x})(\mathbf{t}^\top \nabla f(\mathbf{x}))^2 \quad \text{(Taylor)} \end{aligned}$$

- quadratic form defined by $\mu = w * (\nabla f)(\nabla f)^\top$

Harris and Stephens AVC 1988. A Combined Corner and Edge Detector.

motivation: local autocorrelation

- assume f is differentiable and ignore scale space
- assume an image patch at the origin defined by window w; how much does it change when we shift by $\mathbf{t}$?

$$
\begin{aligned}
E(\mathbf{t}) &= \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - f(\mathbf{x}))^2 \\
&\approx \sum_{\mathbf{x}} w(\mathbf{x})(\mathbf{t}^\top \nabla f(\mathbf{x}))^2 \quad \text{(Taylor)} \\
&= \sum_{\mathbf{x}} w(\mathbf{x}) \mathbf{t}^\top (\nabla f(\mathbf{x}))(\nabla f(\mathbf{x}))^\top \mathbf{t}
\end{aligned}
$$

- quadratic form defined by $\mu = w * (\nabla f)(\nabla f)^\top$

motivation: local autocorrelation

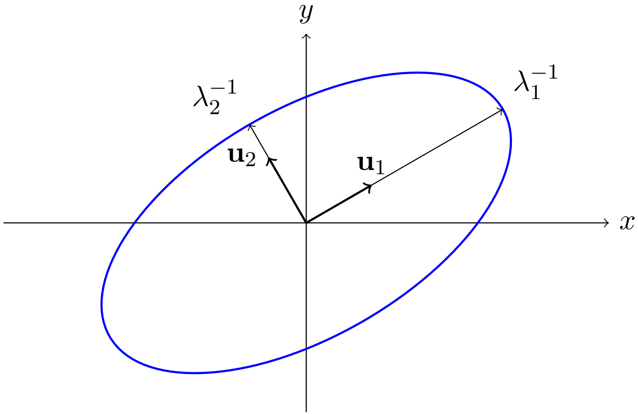
- assume f is differentiable and ignore scale space
- assume an image patch at the origin defined by window w; how much does it change when we shift by $\mathbf{t}$?

$$
\begin{aligned}
E(\mathbf{t}) &= \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - f(\mathbf{x}))^2 \\
&\approx \sum_{\mathbf{x}} w(\mathbf{x})(\mathbf{t}^\top \nabla f(\mathbf{x}))^2 \quad \text{(Taylor)} \\
&= \sum_{\mathbf{x}} w(\mathbf{x})\mathbf{t}^\top (\nabla f(\mathbf{x}))(\nabla f(\mathbf{x}))^\top \mathbf{t} \\
&= \mathbf{t}^\top (w * (\nabla f)(\nabla f)^\top(\mathbf{0}))\mathbf{t}
\end{aligned}
$$

- quadratic form defined by $\mu = w * (\nabla f)(\nabla f)^\top$

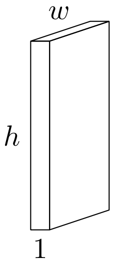
Harris and Stephens AVC 1988. A Combined Corner and Edge Detector.

quadratic form



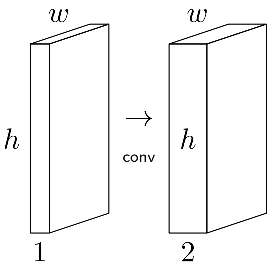
- locus of $(x\ y)^\top A (x\ y) = 1$, where A has eigenvectors $\mathbf{u}_1, \mathbf{u}_2$ and eigenvalues λ_1, λ_2

Harris pipeline



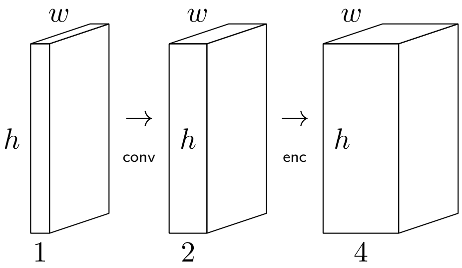
- 3-channel RGB input $\to$ 1-channel gray-scale
- compute gradient $\nabla F = (F_x, F_y)$ at derivation scale
- encode into tensor product $\nabla F \otimes \nabla F = (F_x^2, F_x F_y, F_x F_y, F_y^2)$
- average pooling by window w at integration scale
- compute point-wise nonlinear response function r

Harris pipeline



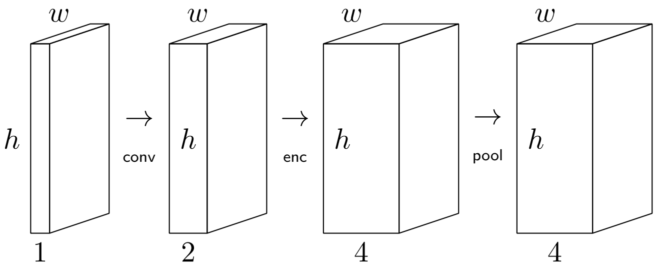
- 3-channel RGB input $\to$ 1-channel gray-scale
- compute gradient $\nabla F = (F_x, F_y)$ at derivation scale
- encode into tensor product $\nabla F \otimes \nabla F = (F_x^2, F_x F_y, F_x F_y, F_y^2)$
- average pooling by window w at integration scale
- compute point-wise nonlinear response function r

Harris pipeline



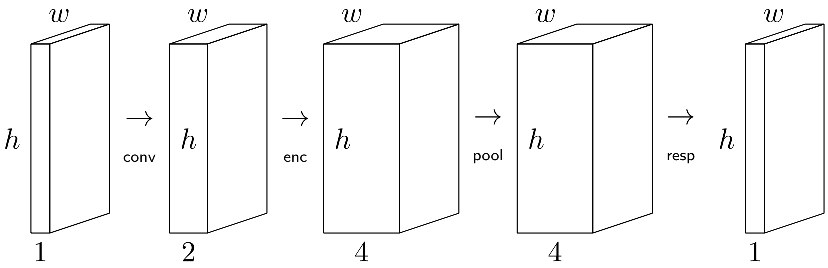
- 3-channel RGB input $\rightarrow$ 1-channel gray-scale
- compute gradient $\nabla F = (F_x, F_y)$ at derivation scale
- encode into tensor product $\nabla F \otimes \nabla F = (F_x^2, F_x F_y, F_x F_y, F_y^2)$
- average pooling by window w at integration scale
- compute point-wise nonlinear response function r

Harris pipeline



- 3-channel RGB input $\to$ 1-channel gray-scale
- compute gradient $\nabla F = (F_x, F_y)$ at derivation scale
- encode into tensor product $\nabla F \otimes \nabla F = (F_x^2, F_x F_y, F_x F_y, F_y^2)$
- average pooling by window w at integration scale
- compute point-wise nonlinear response function r

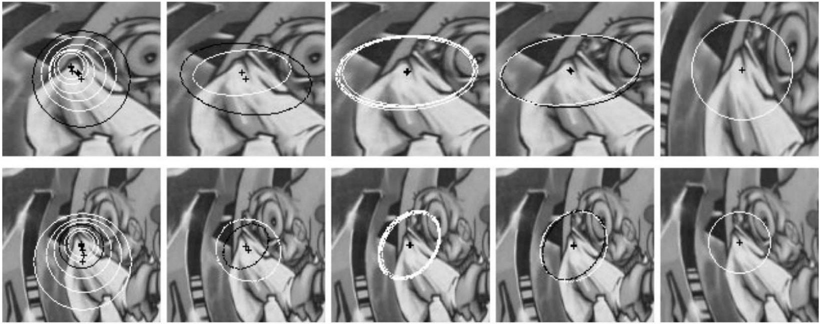
Harris pipeline



- 3-channel RGB input $\to$ 1-channel gray-scale
- compute gradient $\nabla F = (F_x, F_y)$ at derivation scale
- encode into tensor product $\nabla F \otimes \nabla F = (F_x^2, F_x F_y, F_x F_y, F_y^2)$
- average pooling by window w at integration scale
- compute point-wise nonlinear response function r

Harris affine & Hessian affine*

[Mikolajczyk and Schmid 2004]



- multi-scale Harris or Hessian detection, Laplacian scale selection
- iterative affine shape adaptation, based on Lindeberg
- Hessian-affine *de facto* standard on image retrieval for several years

Mikolajczyk and Schmid IJCV 2004. Scale & Affine Invariant Interest Point Detectors.

spatial matching

dense registration

[Lucas and Kanade 1981]

- for each location in an image, find a displacement with respect to another reference image
- appropriate for small displacements, *e.g.* stereopsis or optical flow

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

dense registration

[Lucas and Kanade 1981]

- for each location in an image, find a displacement with respect to another reference image
- appropriate for small displacements, *e.g.* stereopsis or optical flow

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

dense registration

[Lucas and Kanade 1981]

- for each location in an image, find a displacement with respect to another reference image
- appropriate for small displacements, *e.g.* stereopsis or optical flow

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

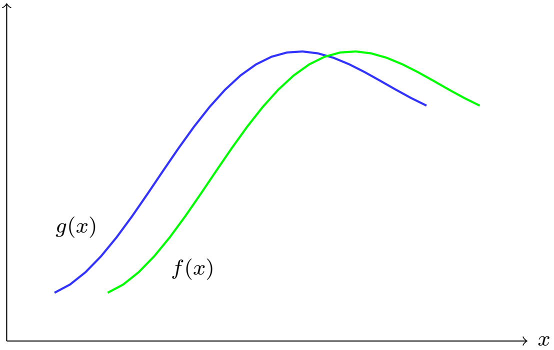
dense registration

[Lucas and Kanade 1981]

- for each location in an image, find a displacement with respect to another reference image
- appropriate for small displacements, *e.g.* stereopsis or optical flow

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

one dimension

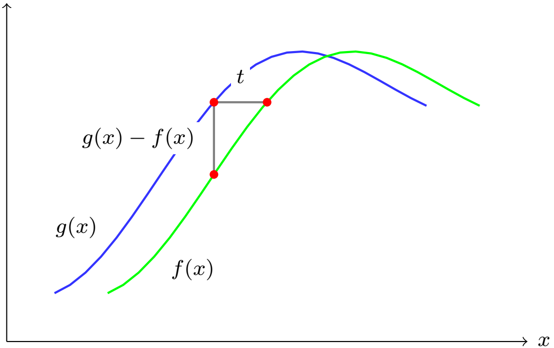


- assuming $g(x) = f(x + t)$ and t is small,

$$\frac{df}{dx}(x) \approx \frac{f(x+t) - f(x)}{t} = \frac{g(x) - f(x)}{t}$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

one dimension

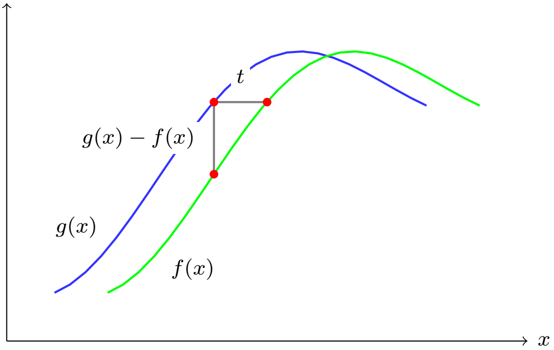


- assuming $g(x) = f(x + t)$ and t is small,

$$\frac{df}{dx}(x) \approx \frac{f(x+t) - f(x)}{t} = \frac{g(x) - f(x)}{t}$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

one dimension



- assuming $g(x) = f(x+t)$ and t is small,

$$\frac{df}{dx}(x) \approx \frac{f(x+t) - f(x)}{t} = \frac{g(x) - f(x)}{t}$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

two dimensions: least squares

- again, assume an image patch defined by window w; what is the error between the patch shifted by $\mathbf{t}$ in reference image f and a patch at the origin in shifted image g?

$$E(\mathbf{t}) = \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - g(\mathbf{x}))^2$$

- error minimized when gradient vanishes

$$\mathbf{0} = \frac{\partial E}{\partial \mathbf{t}} = \sum_{\mathbf{x}} w(\mathbf{x}) 2 \nabla f(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))$$

- least-squares solution

$$\left(w * (\nabla f)(\nabla f)^\top\right) \mathbf{t} = w * ((\nabla f)(g - f))$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

two dimensions: least squares

- again, assume an image patch defined by window w; what is the error between the patch shifted by $\mathbf{t}$ in reference image f and a patch at the origin in shifted image g?

$$E(\mathbf{t}) = \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - g(\mathbf{x}))^2$$
$$\approx \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))^2$$

- error minimized when gradient vanishes

$$\mathbf{0} = \frac{\partial E}{\partial \mathbf{t}} = \sum_{\mathbf{x}} w(\mathbf{x}) 2 \nabla f(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))$$

- least-squares solution

$$\left(w * (\nabla f)(\nabla f)^\top\right) \mathbf{t} = w * ((\nabla f)(g - f))$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

two dimensions: least squares

- again, assume an image patch defined by window w; what is the error between the patch shifted by $\mathbf{t}$ in reference image f and a patch at the origin in shifted image g?

$$\begin{aligned} E(\mathbf{t}) &= \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - g(\mathbf{x}))^2 \\ &\approx \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))^2 \end{aligned}$$

- error minimized when gradient vanishes

$$\mathbf{0} = \frac{\partial E}{\partial \mathbf{t}} = \sum_{\mathbf{x}} w(\mathbf{x}) 2 \nabla f(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))$$

- least-squares solution

$$\left(w * (\nabla f)(\nabla f)^\top\right) \mathbf{t} = w * ((\nabla f)(g - f))$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

two dimensions: least squares

- again, assume an image patch defined by window w; what is the error between the patch shifted by $\mathbf{t}$ in reference image f and a patch at the origin in shifted image g?

$$E(\mathbf{t}) = \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}+\mathbf{t}) - g(\mathbf{x}))^2$$
$$\approx \sum_{\mathbf{x}} w(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))^2$$

- error minimized when gradient vanishes

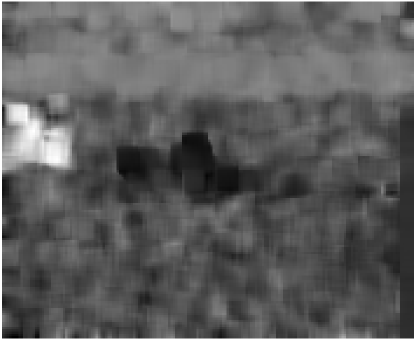
$$\mathbf{0} = \frac{\partial E}{\partial \mathbf{t}} = \sum_{\mathbf{x}} w(\mathbf{x}) 2 \nabla f(\mathbf{x})(f(\mathbf{x}) + \mathbf{t}^\top \nabla f(\mathbf{x}) - g(\mathbf{x}))$$

- least-squares solution

$$\left(w * (\nabla f)(\nabla f)^\top\right) \mathbf{t} = w * ((\nabla f)(g - f))$$

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

dense optical flow



- camera follows background, two objects at opposite horizontal directions
- motion noisy on uniform regions

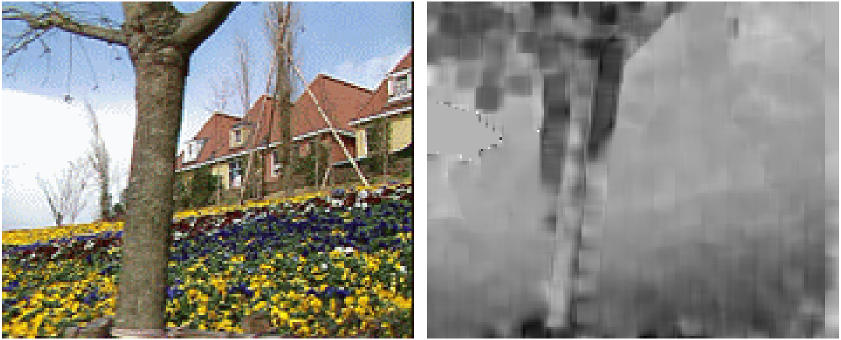
Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

dense optical flow

- camera follows background, two objects at opposite horizontal directions
- motion noisy on uniform regions

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

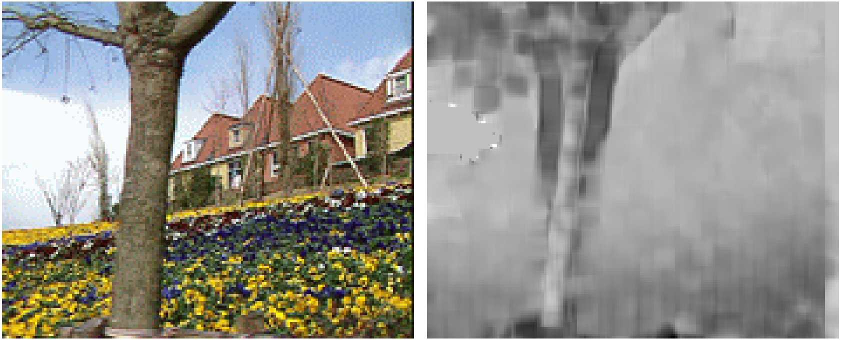
dense optical flow



- parallax: tree closer to viewer than background
- stable on textured regions
- window size visible on edges

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

dense optical flow

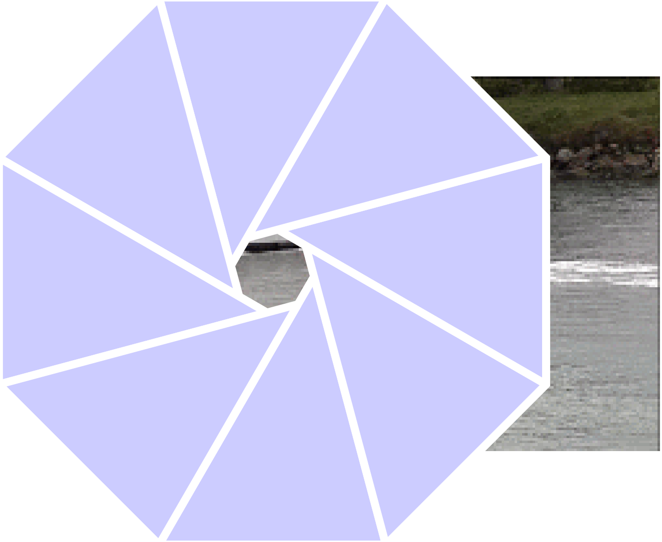


- parallax: tree closer to viewer than background
- stable on textured regions
- window size visible on edges

Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

the aperture problem

the aperture problem



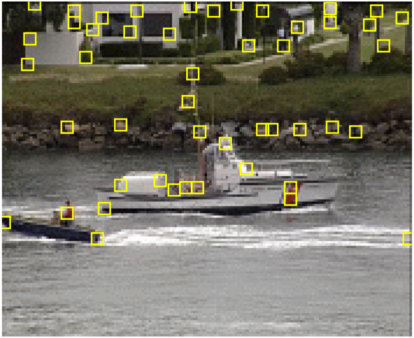
feature point tracking

[Tomasi and Kanade 1991]

- linear system can be solved reliably if matrix μ is well-conditioned: λ_1/λ_2 is not too large
- detect feature points at local maxima of response $\min(\lambda_1, \lambda_2)$

Tomasi and Kanade 1991. Detection and Tracking of Point Features.

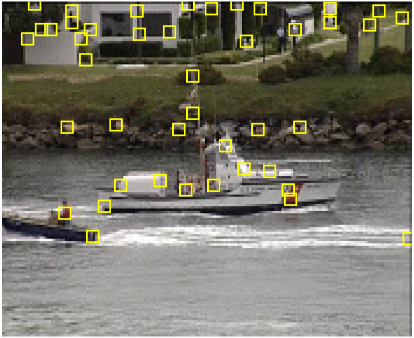
feature point tracking



- uniform regions are not tracked now
- nearly same response as Harris corners
- Q: why do we need the window? what should the size be?

Tomasi and Kanade 1991. Detection and Tracking of Point Features.

feature point tracking



- uniform regions are not tracked now
- nearly same response as Harris corners
- Q: why do we need the window? what should the size be?

Tomasi and Kanade 1991. Detection and Tracking of Point Features.

wide-baseline matching

- in dense registration, we started from a local “template matching” process and found an efficient solution based on a Taylor approximation
- both make sense for small displacements
- in wide-baseline matching, every part of one image may appear anywhere in the other
- we start by pairwise matching of local descriptors without any order and then attempt to enforce some geometric consistency according to a rigid motion model

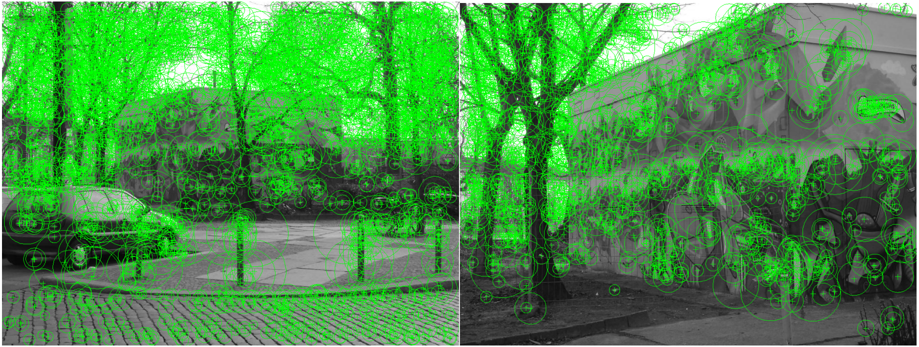
wide-baseline matching

- in dense registration, we started from a local "template matching" process and found an efficient solution based on a Taylor approximation
- both make sense for small displacements
- in wide-baseline matching, every part of one image may appear anywhere in the other
- we start by pairwise matching of local descriptors without any order and then attempt to enforce some geometric consistency according to a rigid motion model

wide-baseline matching

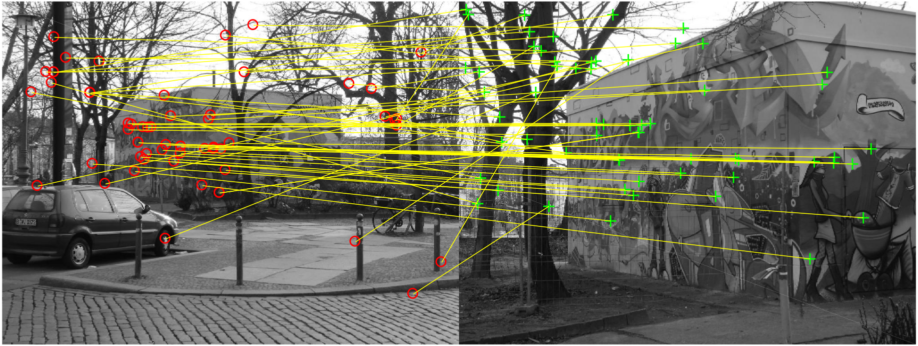
- a region in one image may appear anywhere in the other

wide-baseline matching



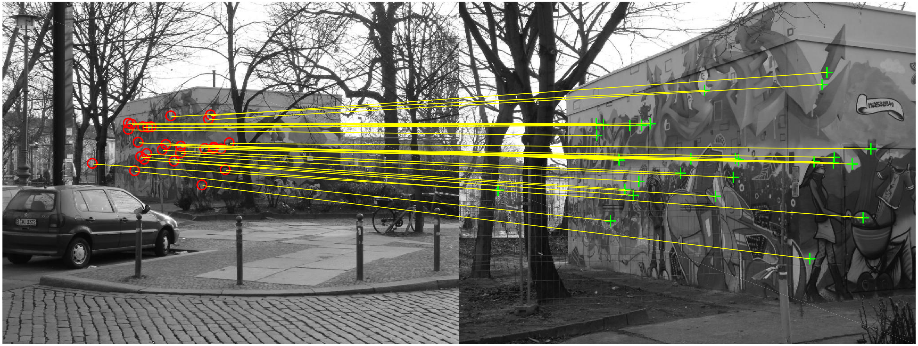
- features detected independently in each image

wide-baseline matching



- tentative correspondences by pairwise descriptor matching

wide-baseline matching



- subset of correspondences that are ‘inlier’ to a rigid transformation

descriptor extraction

for each detected feature in each image

- construct a local histogram of gradient orientations
- find one or more dominant orientations corresponding to peaks in the histogram
- resample local patch at given location, scale, affine shape and orientation
- extract one descriptor for each dominant orientation

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

descriptor matching

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

descriptor matching

- detect features

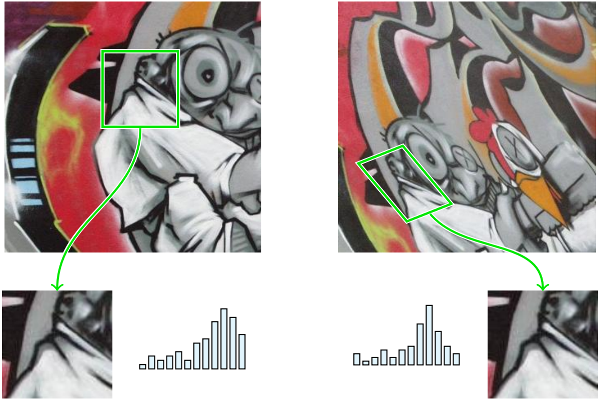
Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

descriptor matching

- detect features - find dominant orientation, resample patches

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

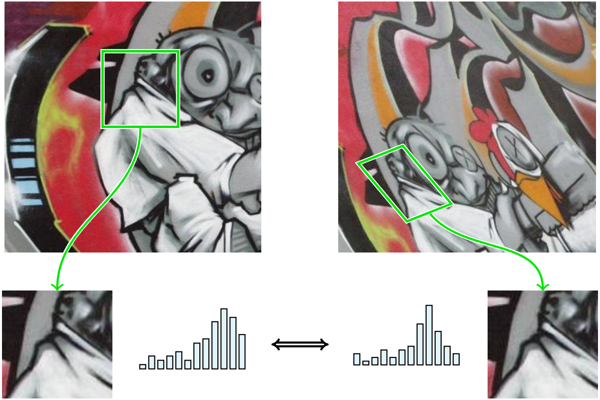
descriptor matching



- detect features - find dominant orientation, resample patches - extract descriptors

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

descriptor matching



- detect features - find dominant orientation, resample patches - extract descriptors - match pairwise

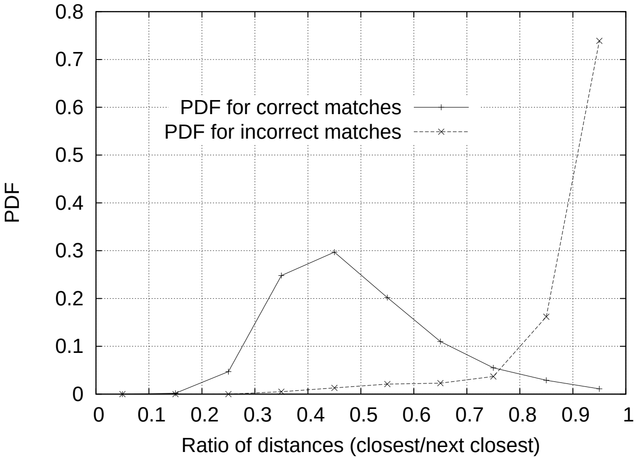
Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

descriptor matching

- for each descriptor in one image, find its two nearest neighbors in the other
- if ratio of distance of first to distance of second is small, make a correspondence
- this yields a list of tentative correspondences

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

ratio test



- ratio of first to second nearest neighbor distance can determine the probability of a true correspondence

Lowe. IJCV 2004. Distinctive Image Features From Scale-Invariant Keypoints.

spatial matching

why is it difficult?

- should allow for a geometric transformation
- fitting the model to data (correspondences) is sensitive to outliers: should find a subset of *inliers* first
- finding inliers to a transformation requires finding the *transformation* in the first place
- correspondences have gross error
- inliers are typically less than 50%

geometric transformations

- two images f, f' are equal at points $\mathbf{x}, \mathbf{x}'$

$$f(\mathbf{x}) = f'(\mathbf{x}')$$

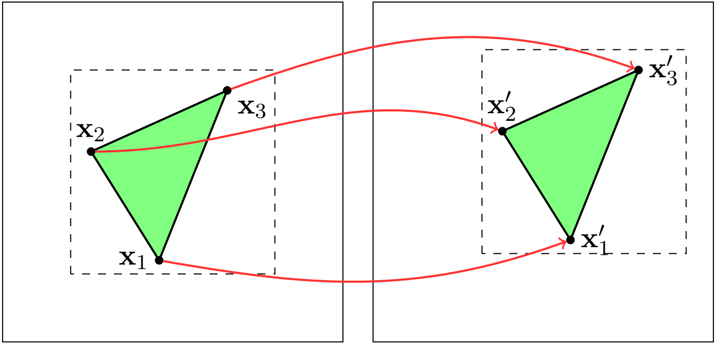
- $\mathbf{x}$ is mapped to $\mathbf{x}'$

$$\mathbf{x}' = T(\mathbf{x})$$

- T is a bijection of $\mathbb{R}^2$ to itself:

$$T : \mathbb{R}^2 \to \mathbb{R}^2$$

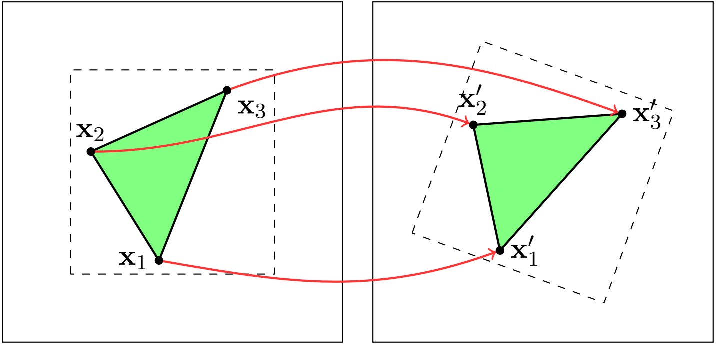
geometric transformations



- translation: 2 degrees of freedom

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

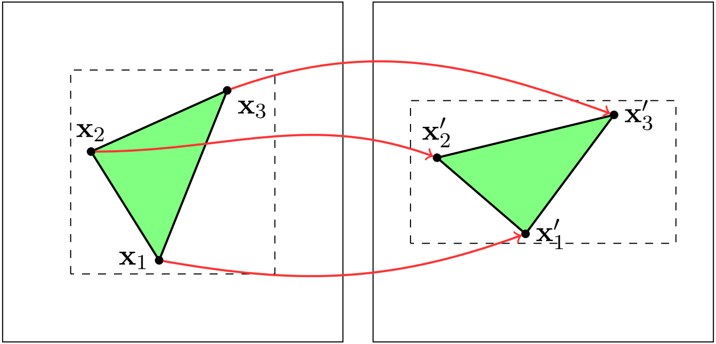
geometric transformations



- rotation: 1 degree of freedom

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

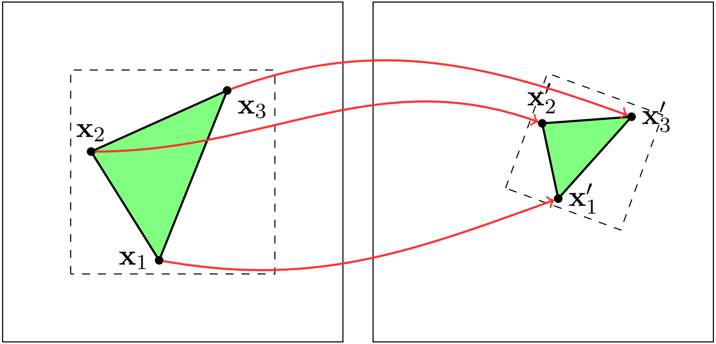
geometric transformations



- scale: 2 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

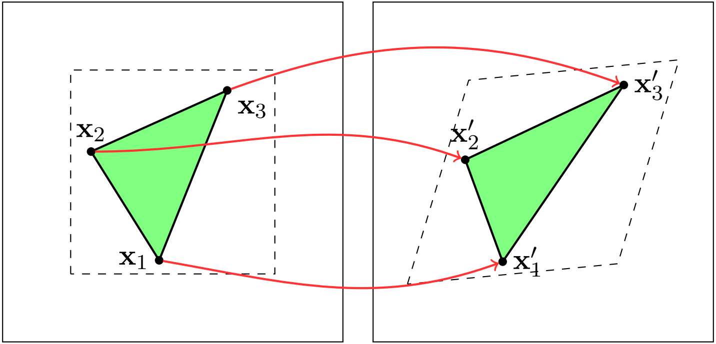
geometric transformations



- similarity: 4 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} r\cos\theta & -r\sin\theta & t_x \\ r\sin\theta & r\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

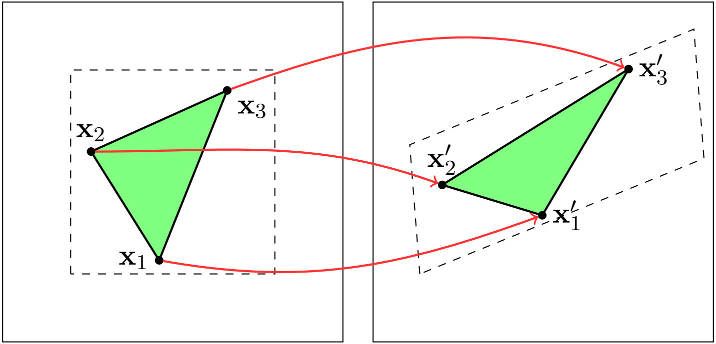
geometric transformations



- shear: 2 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & b_x & 0 \\ b_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

geometric transformations



- affine: 6 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

however

- details don't matter; in all cases, the problem is transformed to a linear system (why?)

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

  where $\mathbf{A}, \mathbf{b}$ contain coordinates of known point correspondences from images f, f' respectively, and $\mathbf{x}$ contains our model parameters
- we need $n = \lceil d/2 \rceil$ correspondences, where d are the degrees of freedom of our model
- let's take the simplest model as an example: fit a line to two points

however

- details don't matter; in all cases, the problem is transformed to a linear system (why?)

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

  where $\mathbf{A}, \mathbf{b}$ contain coordinates of known point correspondences from images f, f' respectively, and $\mathbf{x}$ contains our model parameters
- we need $n = \lceil d/2 \rceil$ correspondences, where d are the degrees of freedom of our model
- let's take the simplest model as an example: fit a line to two points

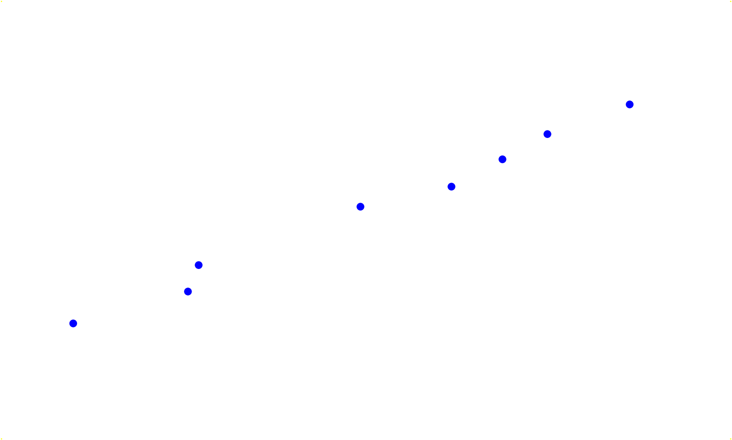
however

- details don't matter; in all cases, the problem is transformed to a linear system (why?)

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

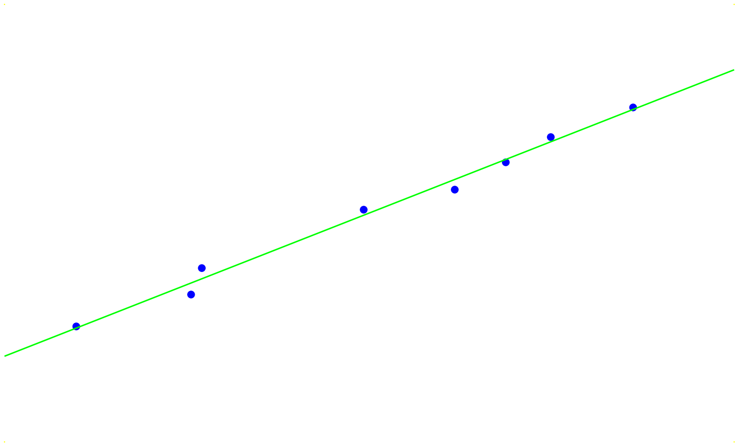
  where $\mathbf{A}, \mathbf{b}$ contain coordinates of known point correspondences from images f, f' respectively, and $\mathbf{x}$ contains our model parameters
- we need $n = \lceil d/2 \rceil$ correspondences, where d are the degrees of freedom of our model
- let's take the simplest model as an example: fit a line to two points

least squares and gross outliers



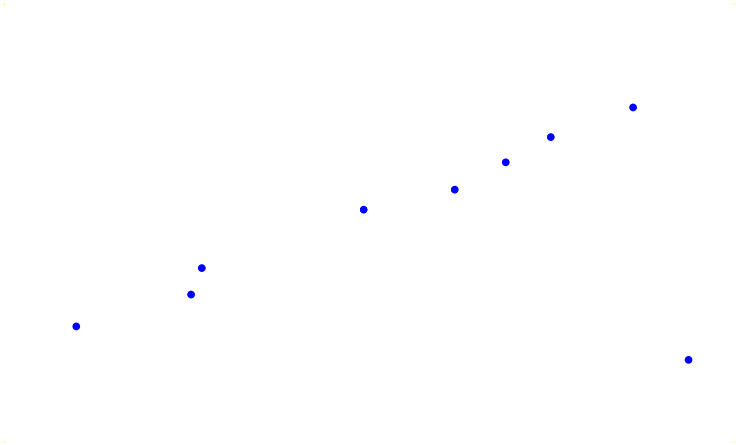
- clean data, no outliers : least squares fit ok

least squares and gross outliers



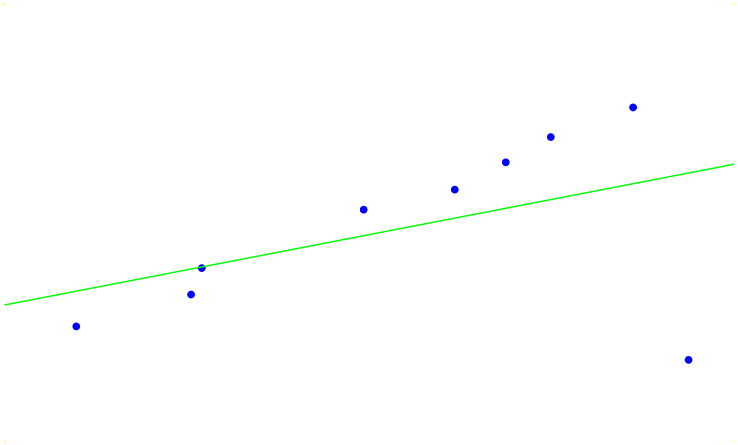
- clean data, no outliers : least squares fit ok

least squares and gross outliers



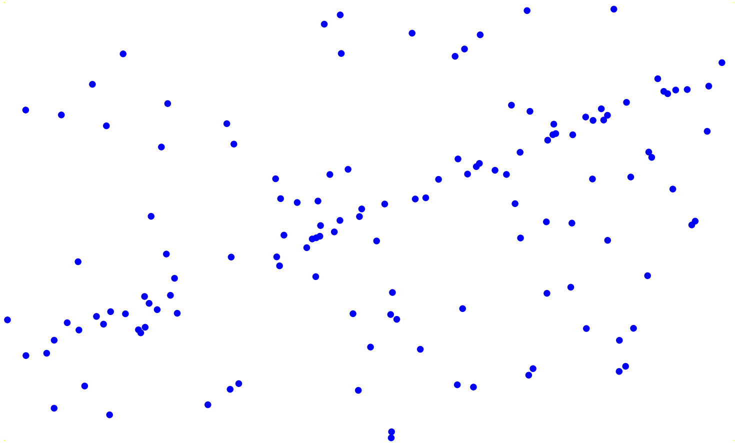
- one gross outlier : least squares fit fails

least squares and gross outliers



- one gross outlier : least squares fit fails

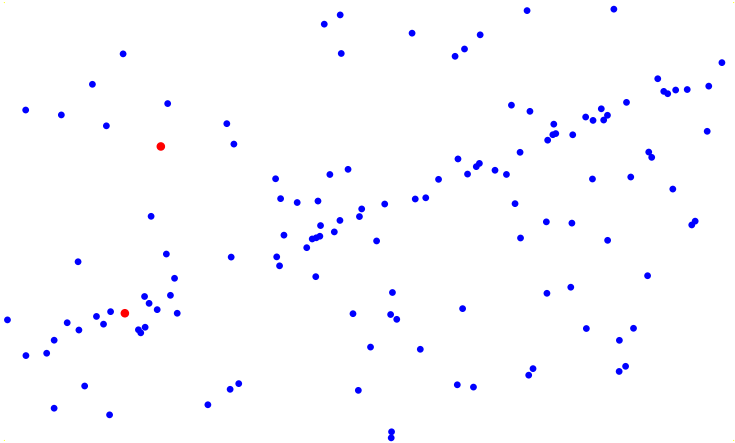
random sample consensus (RANSAC)



- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

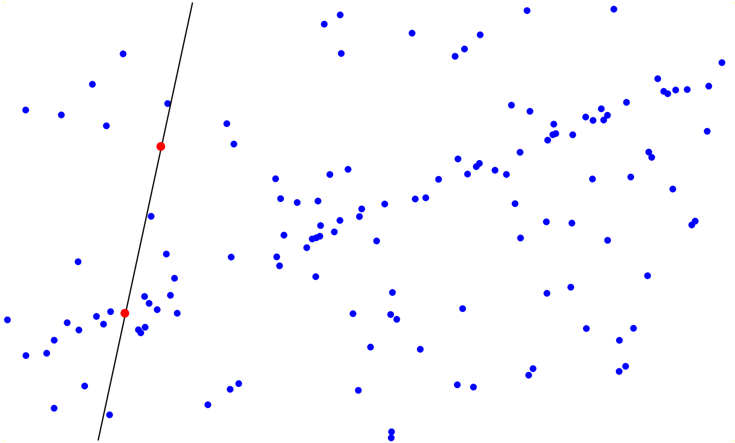
random sample consensus (RANSAC)



- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

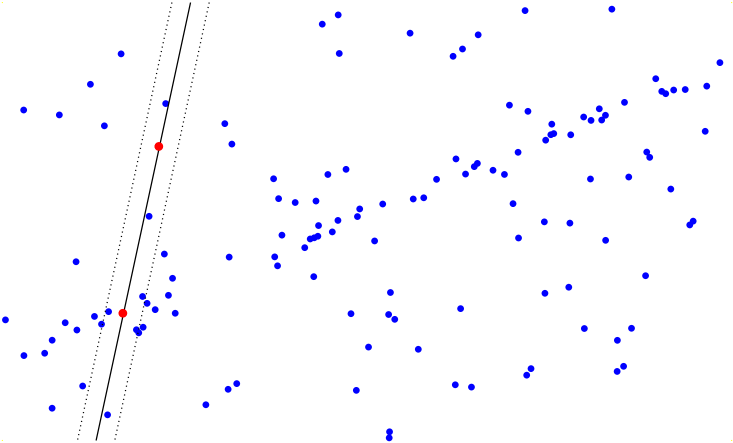
random sample consensus (RANSAC)



- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

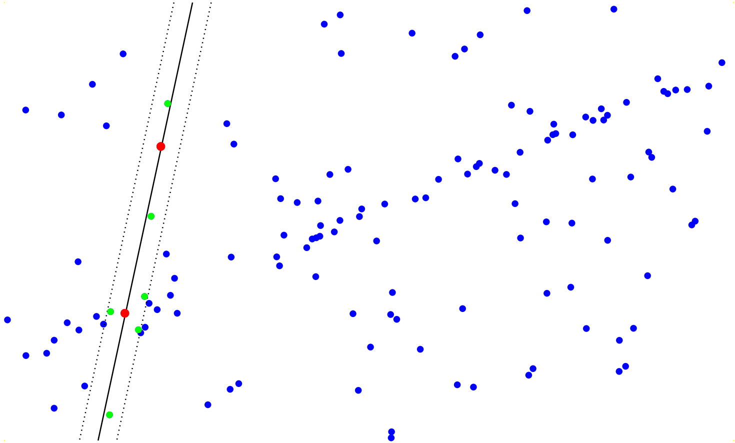
random sample consensus (RANSAC)



- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

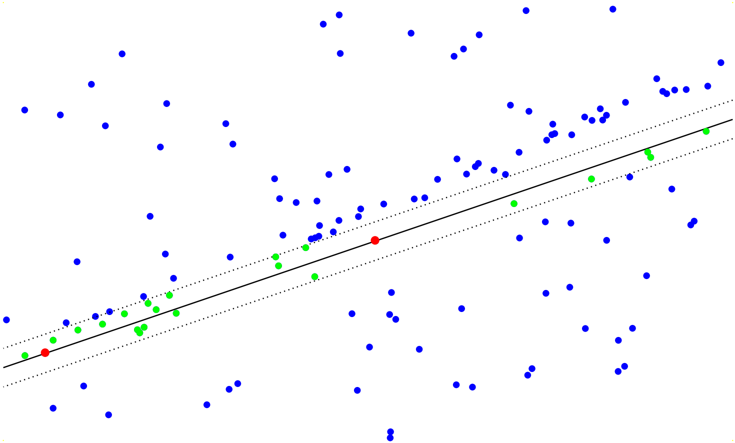
random sample consensus (RANSAC)



- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

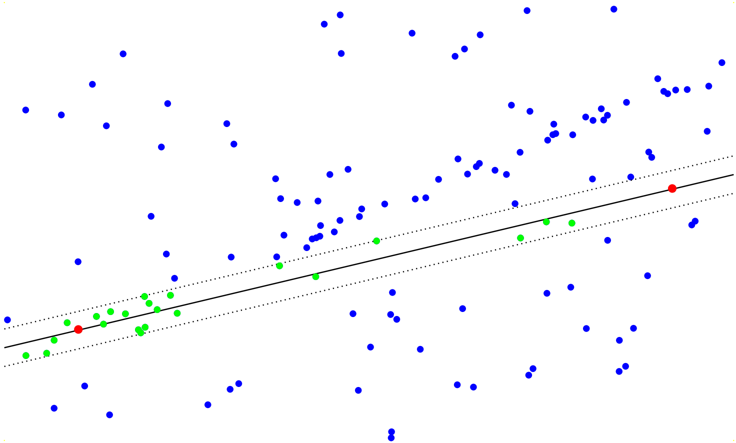
random sample consensus (RANSAC)



- repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

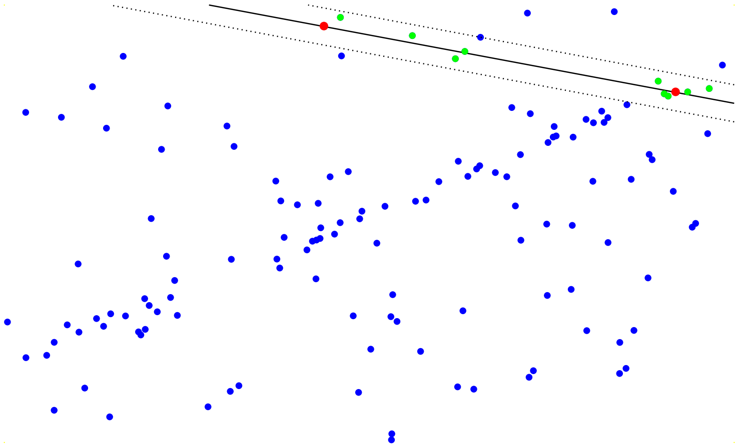
random sample consensus (RANSAC)



- repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

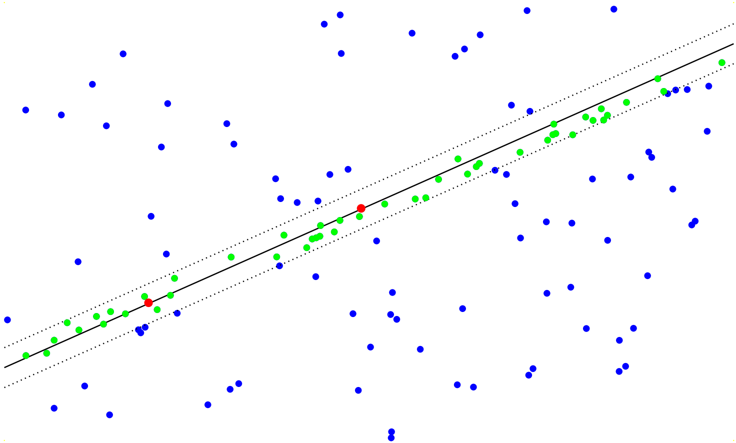
random sample consensus (RANSAC)



- repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

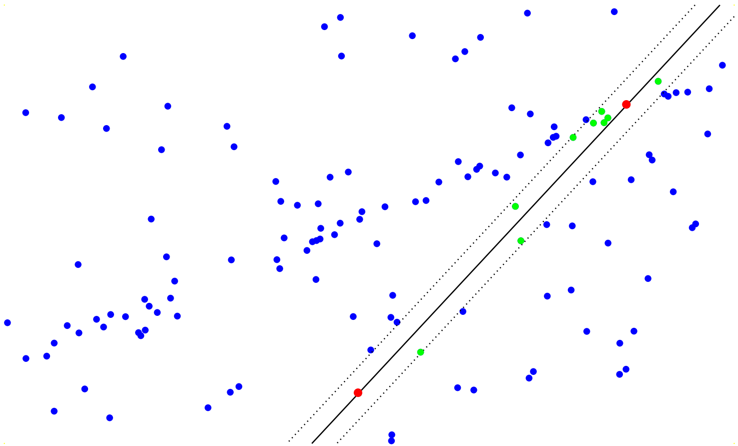
random sample consensus (RANSAC)



- repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

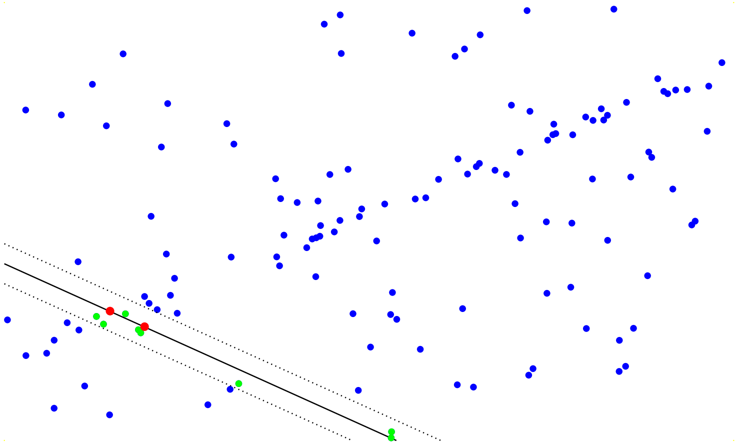
random sample consensus (RANSAC)



- repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

random sample consensus (RANSAC)



- repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

random sample consensus (RANSAC)*

[Fischler and Bolles 1981]

- X: data (tentative correspondences)
- n: minimum number of samples to fit a model
- $s(x;\theta)$: score of sample x given model parameters θ
- repeat
  - hypothesis
    - draw n samples $H \subset X$ at random
    - fit model to H, compute parameters θ
  - verification
    - are data consistent with hypothesis? compute score $S = \sum_{x \in X} s(x;\theta)$
    - if $S^* > S$, store solution $\theta^* := \theta$, $S^* := S$

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

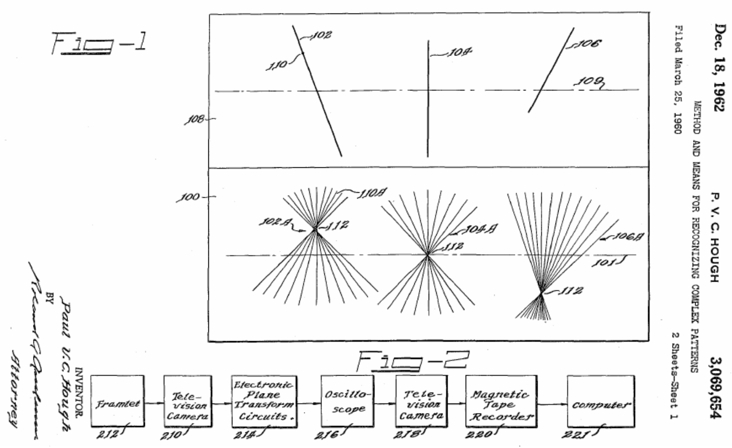
RANSAC issues*

- inlier ratio w unknown
- too expensive when minimum number of samples is large (*e.g.* $n > 6$) and inlier ratio is small *e.g.* $w < 10\%$): 10^6 iterations for 1% probability of failure

Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Hough transform*

[Hough 1962]

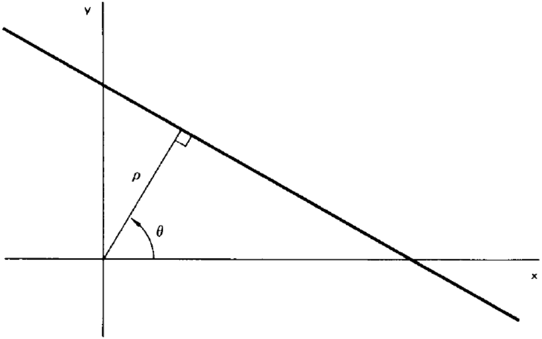


- detect lines by a voting process in parameter space
- slope-intercept parametrization unbounded for vertical lines

Hough. US Patent 1962. Method and Means for Recognizing Complex patterns.

Hough transform*

[Duda and Hart 1972]



- polar parametrization makes parameter space bounded
- discusses generalization to analytic curves; space exponential in number of parameters
- equivalent to Radon transform, but makes sense for sparse input

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

Hough transform*

idea

- n samples are needed to fit a model (*e.g.* 2 points for a line)
- but even one sample brings some information
- in the space of all possible models, vote for the ones that satisfy a given sample
- collect votes from all samples, and seek for consensus

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

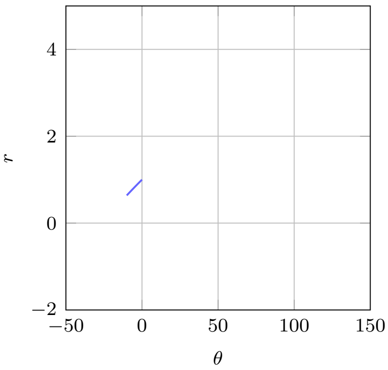
Hough transform*

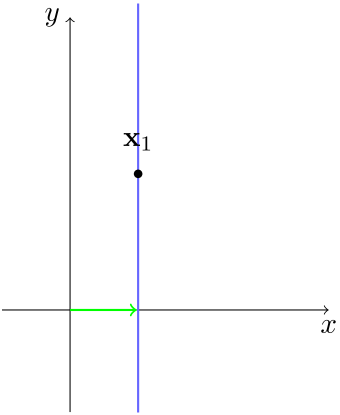
idea

- n samples are needed to fit a model (*e.g.* 2 points for a line)
- but even one sample brings some information
- in the space of all possible models, vote for the ones that satisfy a given sample
- collect votes from all samples, and seek for consensus

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

voting in parameter space*



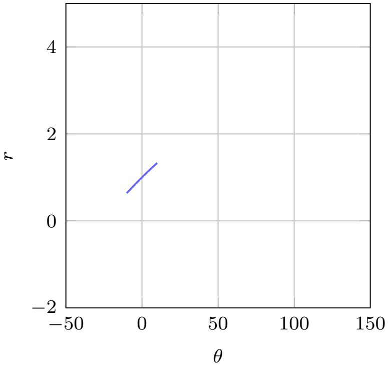


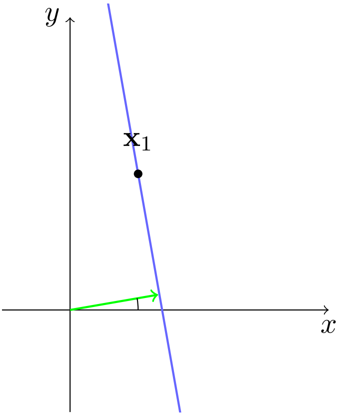
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



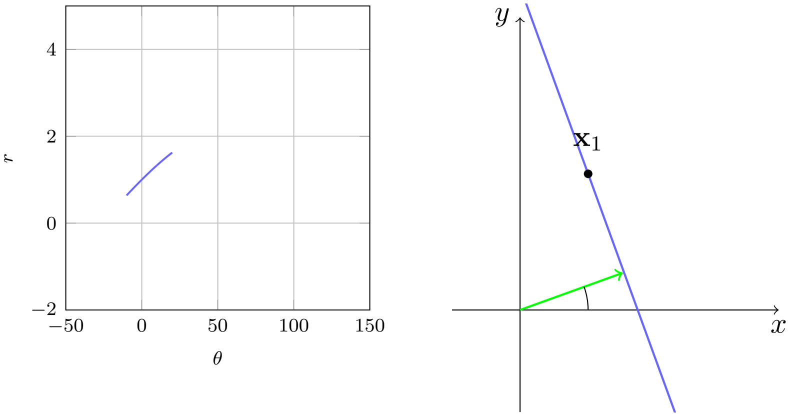


- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*

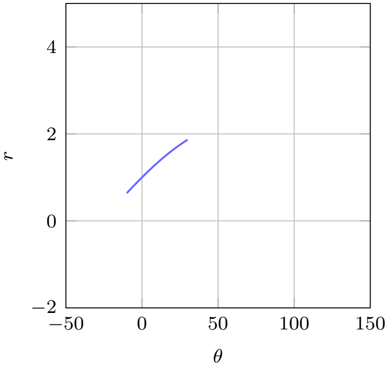


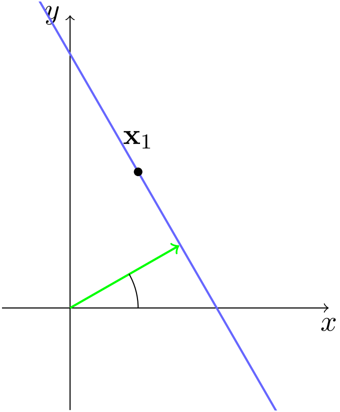
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



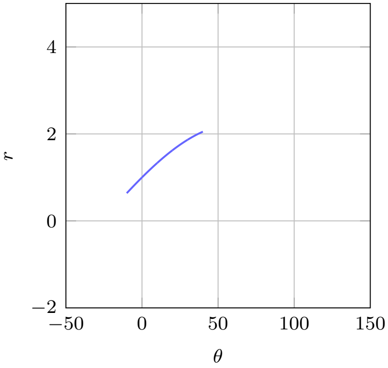


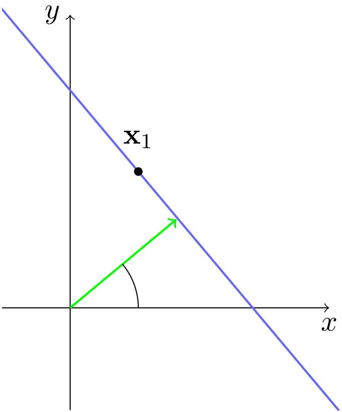
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



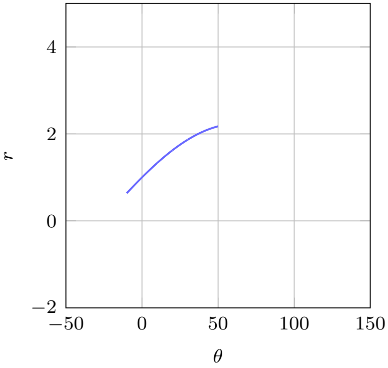


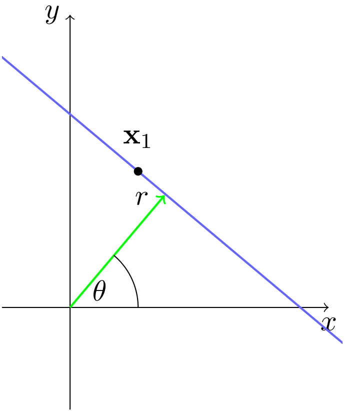
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



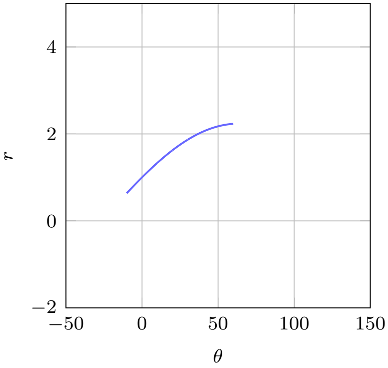


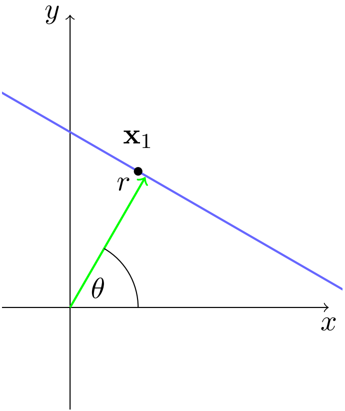
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



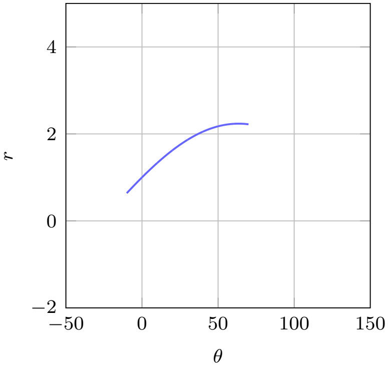


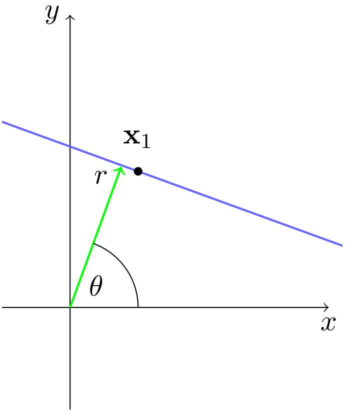
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



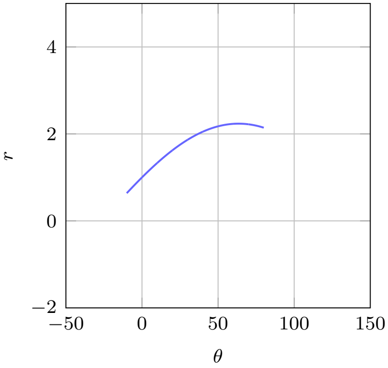


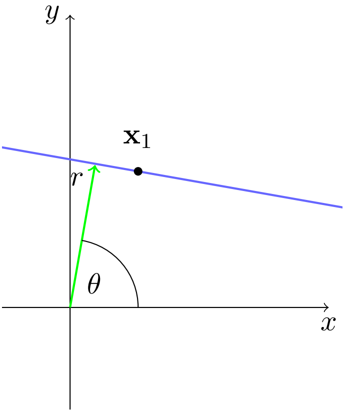
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



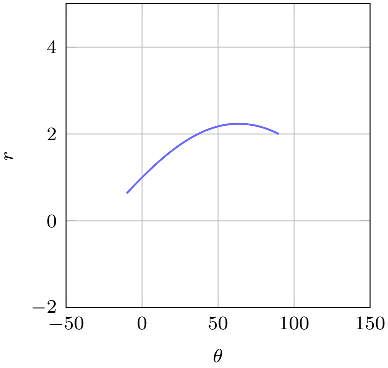


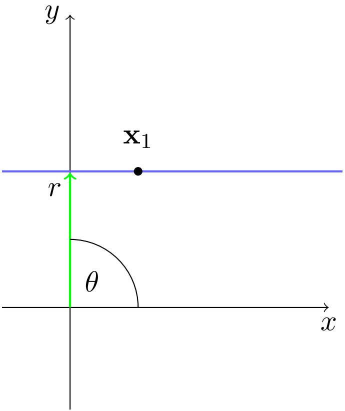
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



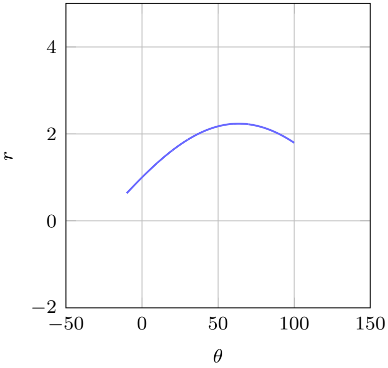


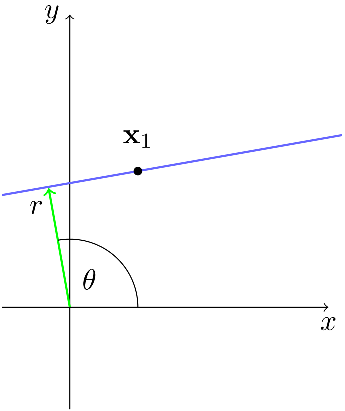
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*

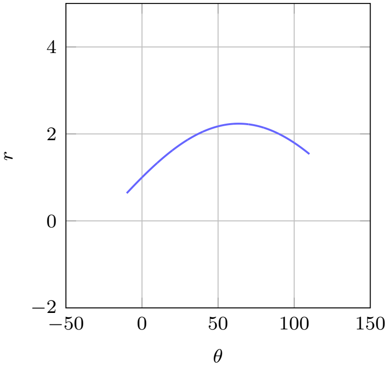


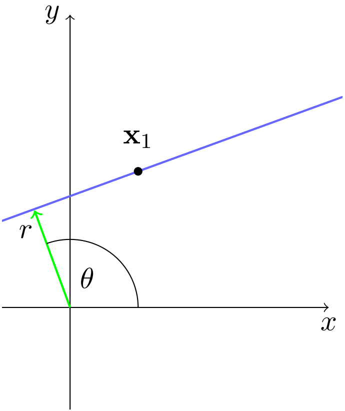


- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

voting in parameter space*



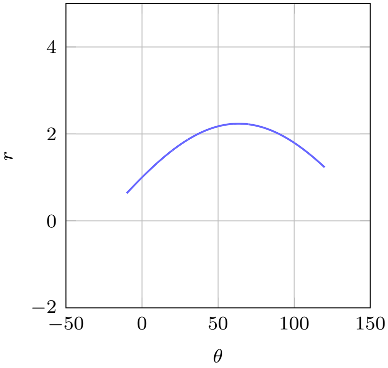


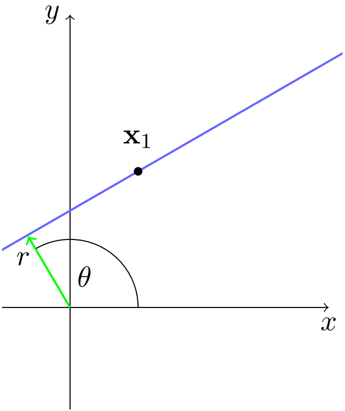
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



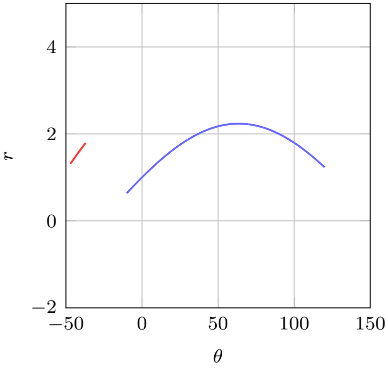


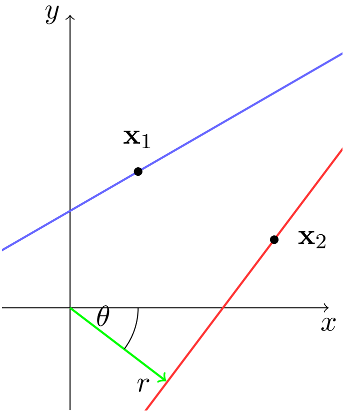
- all lines through $\mathbf{x}_1 = (x_1, y_1)$ are defined by (r, θ) that satisfy

$$r = x_1 \cos\theta + y_1 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



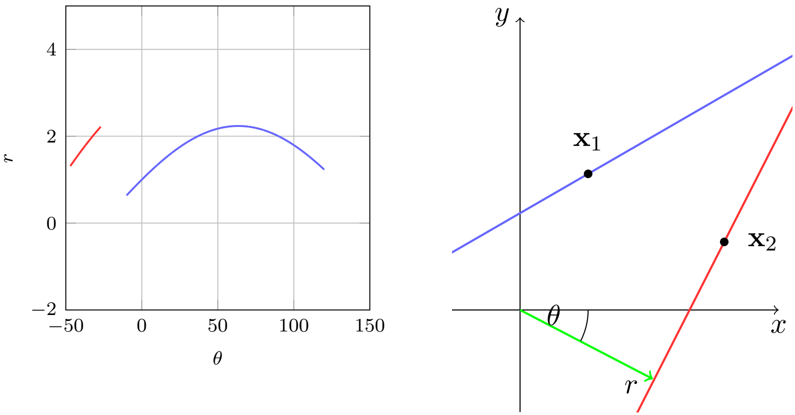


- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*

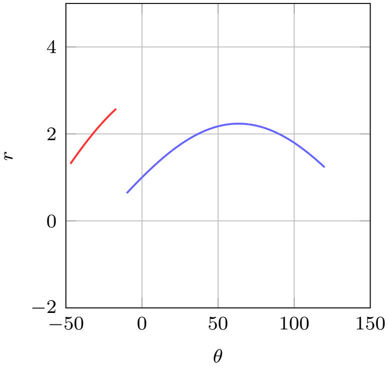


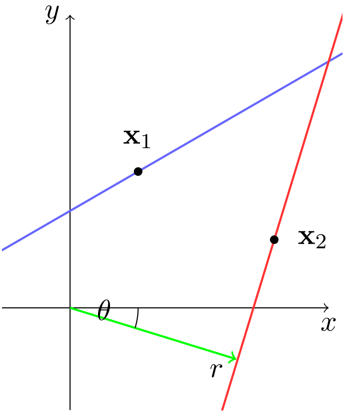
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



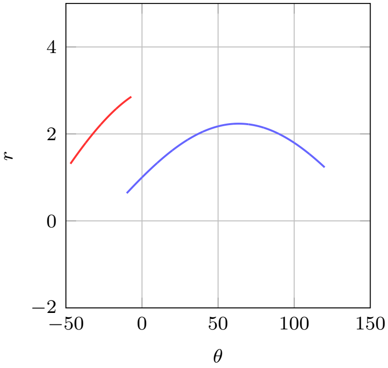


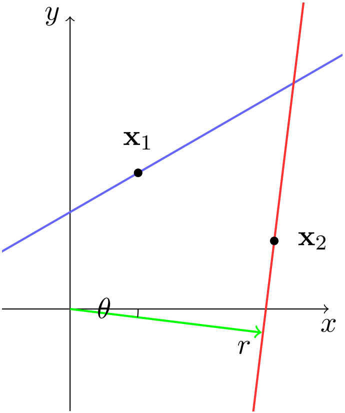
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



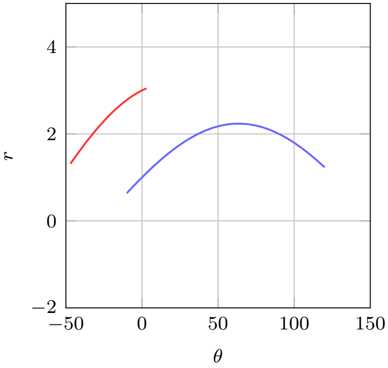


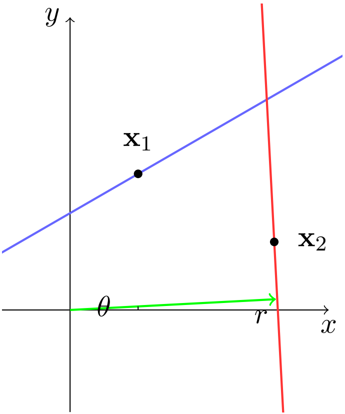
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



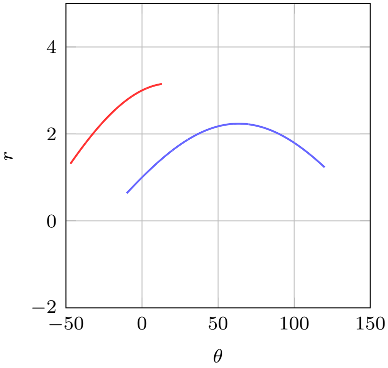


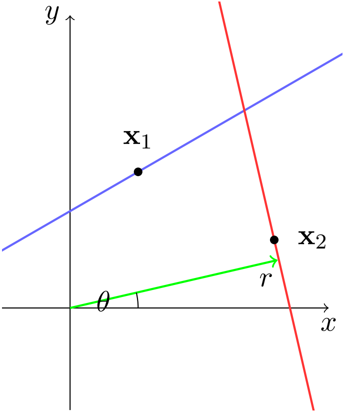
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



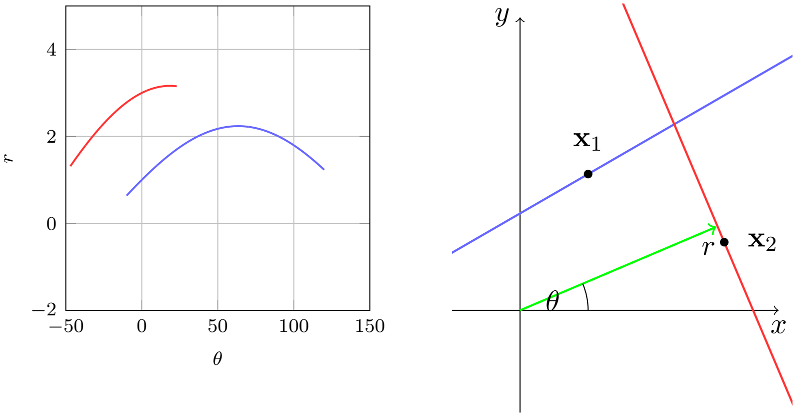


- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*

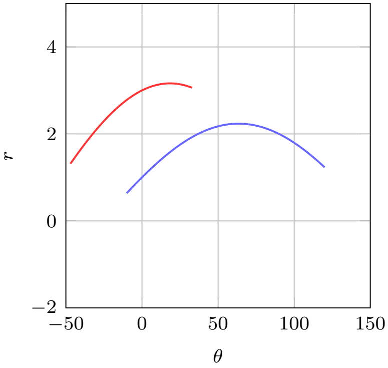


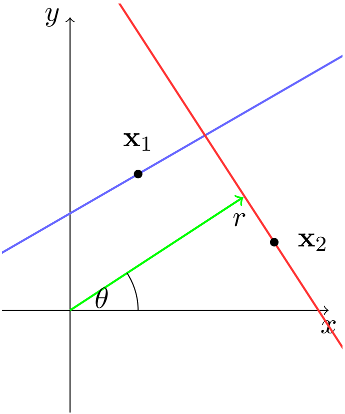
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos \theta + y_2 \sin \theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



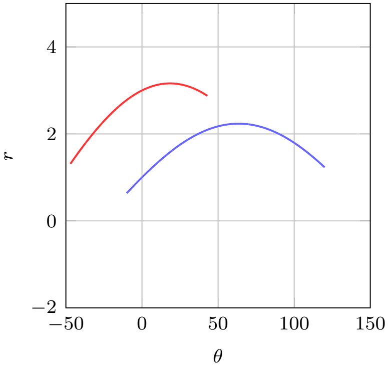


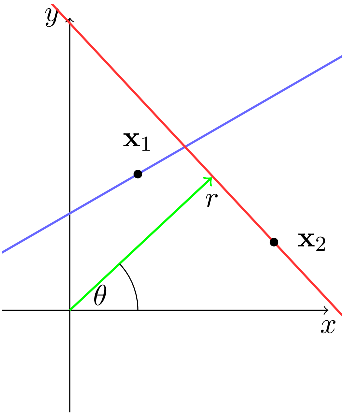
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



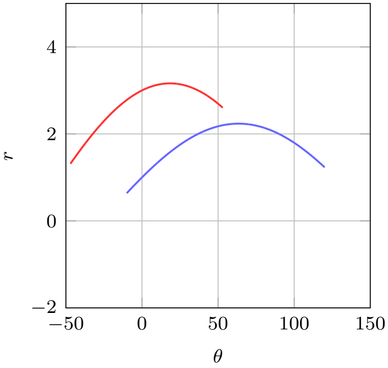


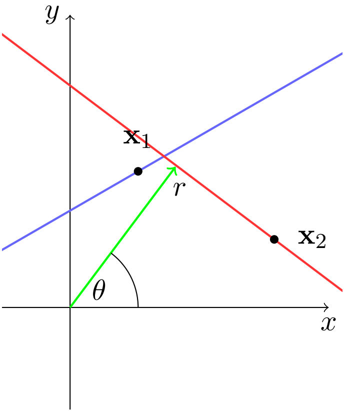
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



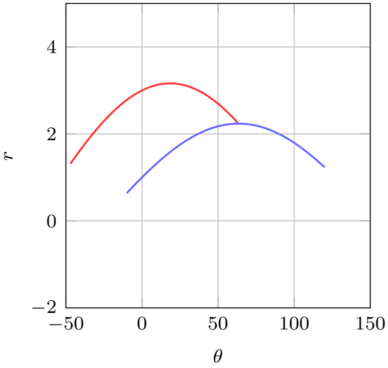


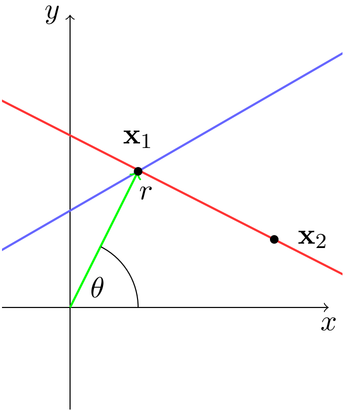
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



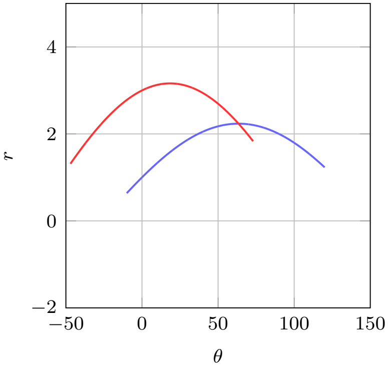


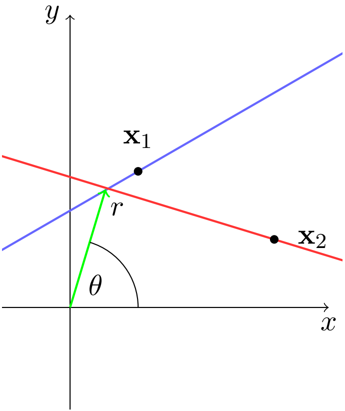
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



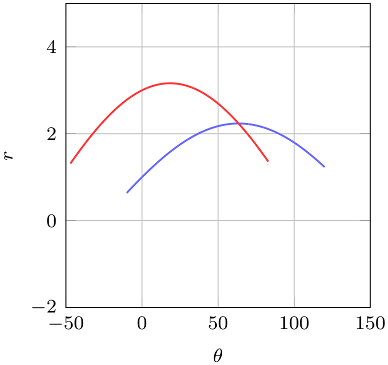


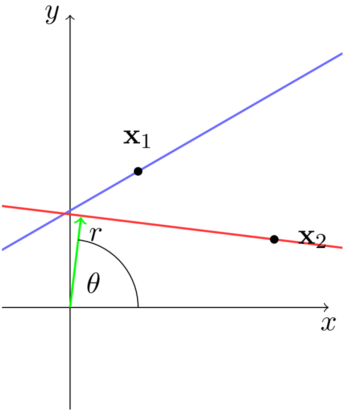
- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

voting in parameter space*



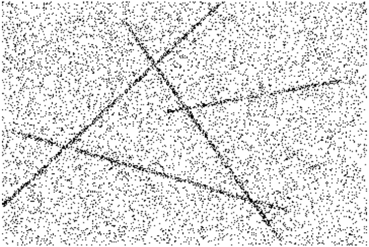


- all lines through $\mathbf{x}_2 = (x_2, y_2)$ are defined by (r, θ) that satisfy

$$r = x_2 \cos\theta + y_2 \sin\theta$$

Duda and Hart. CACM 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures.

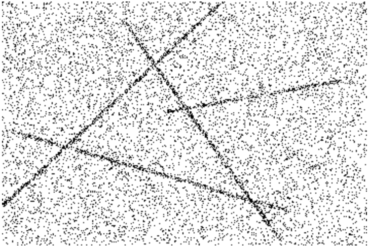
line detection*



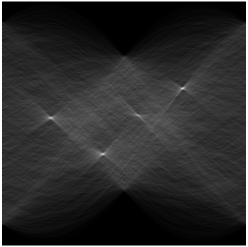
points

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

line detection*



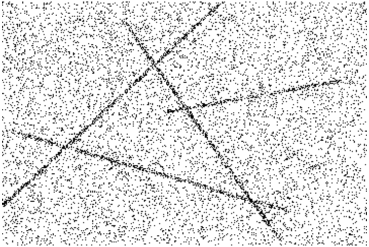
points



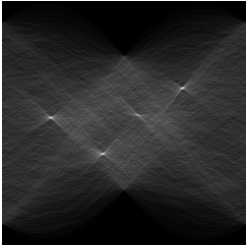
accumulator

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

line detection*



points

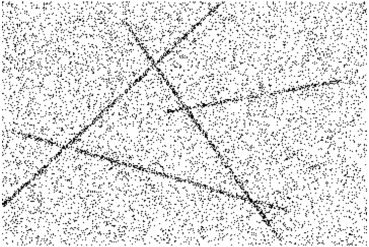


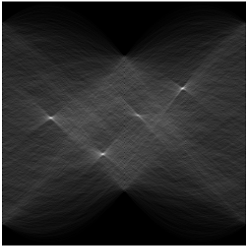
accumulator

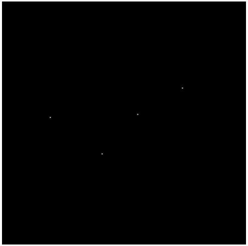
thresholding

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

line detection*

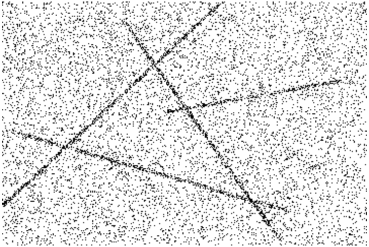

points


accumulator

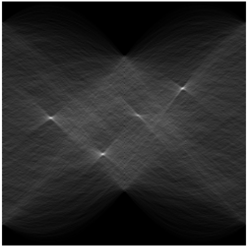

local maxima

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

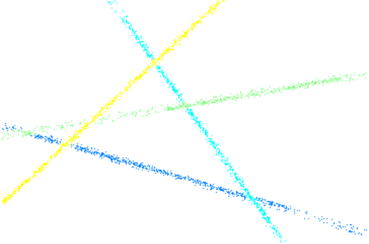
line detection*



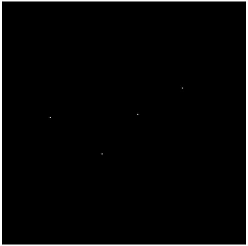
points



accumulator



labels



local maxima

Duda and Hart. CACM 1972 Use of the Hough Transformation to Detect Lines and Curves in pictures.

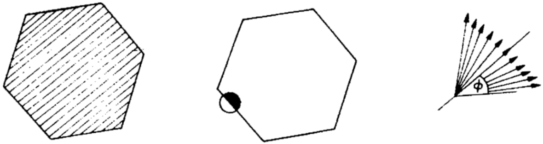
Hough voting*

- X: data
- n: number of model parameters
- A: n-dimensional accumulator array, initially zero
- hypotheses: for each sample $x \in X$
    - for each set of model parameters θ consistent with x
        - voting: increment $A[\theta]$
- "verification":
    - threshold A, relative to maximum
    - non-maxima suppression: detect local maxima

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

generalized Hough transform*

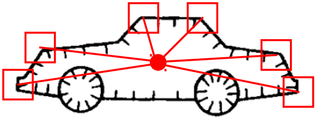
[Ballard 1981]



- generalize to arbitrary shapes
- similarity transformation, 4d parameter space: translation, scaling, rotation
- use gradient orientation to reduce number of votes per sample

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

translation space*

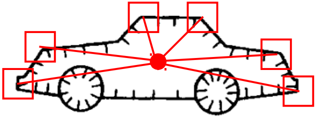


model image

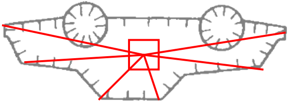
- model: record coordinates relative to reference point
- test: each point votes for all possible coordinates of reference point, which are reversed

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

translation space*



model image

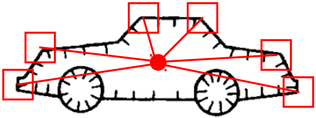


test image

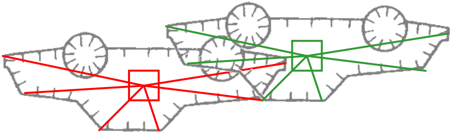
- model: record coordinates relative to reference point
- test: each point votes for all possible coordinates of reference point, which are reversed

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

translation space*



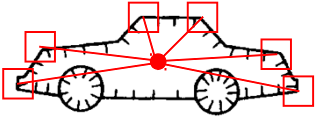
model image



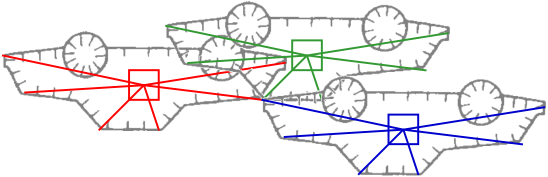
test image

- model: record coordinates relative to reference point
- test: each point votes for all possible coordinates of reference point, which are reversed

translation space*



model image

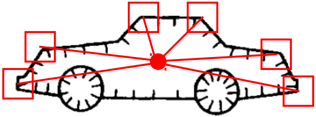


test image

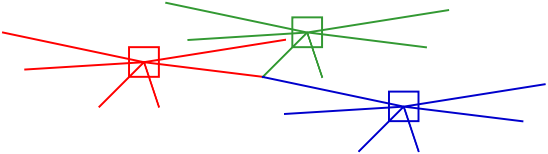
- model: record coordinates relative to reference point
- test: each point votes for all possible coordinates of reference point, which are reversed

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

translation space*



model image

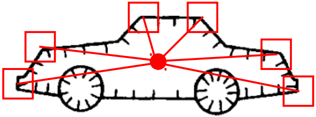


test image

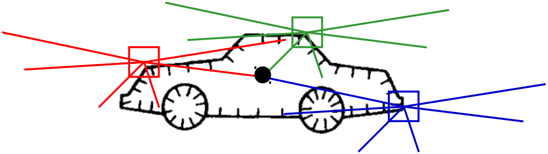
- model: record coordinates relative to reference point
- test: each point votes for all possible coordinates of reference point, which are reversed

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

translation space*



model image

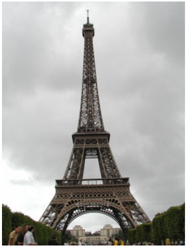


test image

- model: record coordinates relative to reference point
- test: each point votes for all possible coordinates of reference point, which are reversed

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

Eiffel tower detection*



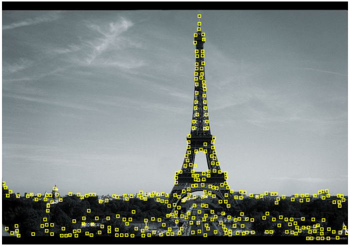
model image

test image

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

Eiffel tower detection*

model image points

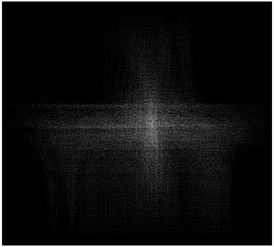


test image points

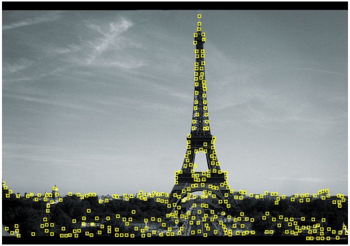
Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

Eiffel tower detection*

model image points



accumulator

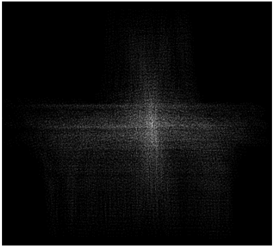


test image points

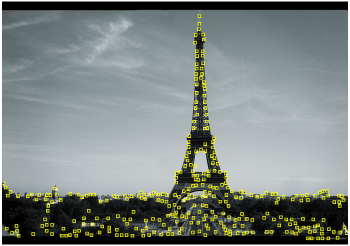
Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

Eiffel tower detection*

model image points



accumulator



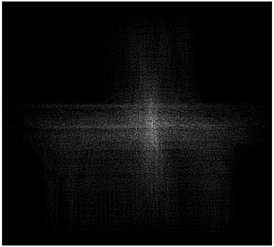
test image points

local maxima

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

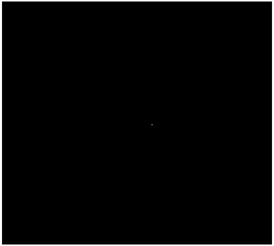
Eiffel tower detection*

model image points



accumulator

detected location



local maxima

Ballard. PR 1981. Generalizing the Hough Transform to Detect Arbitrary shapes.

Hough is (sparse) cross-correlation*

- model points H, test points X as signals

$$h[\mathbf{n}] = \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - \mathbf{h}]$$

$$x[\mathbf{n}] = \sum_{\mathbf{x} \in X} \delta[\mathbf{n} - \mathbf{x}]$$

- for each test point $\mathbf{x} \in X$
  - for each translation $\mathbf{x} - \mathbf{h}$ consistent with $\mathbf{x}$ (for $\mathbf{h} \in H$)
    - voting: increment accumulator A at $\mathbf{x} - \mathbf{h}$
- in symbols

$$A = \sum_{\mathbf{x} \in X} \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - (\mathbf{x} - \mathbf{h})]$$

Hough is (sparse) cross-correlation*

- model points H, test points X as signals

$$h[\mathbf{n}] = \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - \mathbf{h}]$$

$$x[\mathbf{n}] = \sum_{\mathbf{x} \in X} \delta[\mathbf{n} - \mathbf{x}]$$

- for each test point $\mathbf{x} \in X$
  - for each translation $\mathbf{x} - \mathbf{h}$ consistent with $\mathbf{x}$ (for $\mathbf{h} \in H$)
    - voting: increment accumulator A at $\mathbf{x} - \mathbf{h}$
- in symbols

$$A = \sum_{\mathbf{x} \in X} \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - (\mathbf{x} - \mathbf{h})]$$

Hough is (sparse) cross-correlation*

- model points H, test points X as signals

$$h[\mathbf{n}] = \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - \mathbf{h}]$$

$$x[\mathbf{n}] = \sum_{\mathbf{x} \in X} \delta[\mathbf{n} - \mathbf{x}]$$

- for each test point $\mathbf{x} \in X$
  - for each translation $\mathbf{x} - \mathbf{h}$ consistent with $\mathbf{x}$ (for $\mathbf{h} \in H$)
    - voting: increment accumulator A at $\mathbf{x} - \mathbf{h}$
- in symbols

$$A = \sum_{\mathbf{x} \in X} \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - (\mathbf{x} - \mathbf{h})]$$

Hough is (sparse) cross-correlation*

- model points H, test points X as signals

$$h[\mathbf{n}] = \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - \mathbf{h}]$$

$$x[\mathbf{n}] = \sum_{\mathbf{x} \in X} \delta[\mathbf{n} - \mathbf{x}]$$

- for each test point $\mathbf{x} \in X$
  - for each translation $\mathbf{x} - \mathbf{h}$ consistent with $\mathbf{x}$ (for $\mathbf{h} \in H$)
    - voting: increment accumulator A at $\mathbf{x} - \mathbf{h}$
- in symbols

$$A = \sum_{\mathbf{x} \in X} \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - (\mathbf{x} - \mathbf{h})]$$

Hough is (sparse) cross-correlation*

- model points H, test points X as signals

$$h[\mathbf{n}] = \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - \mathbf{h}]$$

$$x[\mathbf{n}] = \sum_{\mathbf{x} \in X} \delta[\mathbf{n} - \mathbf{x}]$$

- for each test point $\mathbf{x} \in X$
  - for each translation $\mathbf{x} - \mathbf{h}$ consistent with $\mathbf{x}$ (for $\mathbf{h} \in H$)
    - voting: increment accumulator A at $\mathbf{x} - \mathbf{h}$
- in symbols - try it!

$$A = \sum_{\mathbf{x} \in X} \sum_{\mathbf{h} \in H} \delta[\mathbf{n} - (\mathbf{x} - \mathbf{h})] = \sum_{\mathbf{k}} x[\mathbf{k}] h[\mathbf{k} - \mathbf{n}]$$

local shape*

[Lowe 2004]

- a SIFT feature is determined by location, scale and orientation; a single feature correspondence can yield a 4-dof similarity transformation
- hypotheses: sparse Hough voting in 4-dimensional space; each correspondence casts a single vote in a hash table
- verification: on each bin with at least 3 votes, find inliers, form linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ and fit a 6-dof affine transformation by least-squares

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Lowe. ICCV 1999. Object recognition from local scale-invariant features.

local shape*

[Lowe 2004]

- a SIFT feature is determined by location, scale and orientation; a single feature correspondence can yield a 4-dof similarity transformation
- hypotheses: sparse Hough voting in 4-dimensional space; each correspondence casts a single vote in a hash table
- verification: on each bin with at least 3 votes, find inliers, form linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ and fit a 6-dof affine transformation by least-squares

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Lowe. ICCV 1999. Object recognition from local scale-invariant features.

local shape*

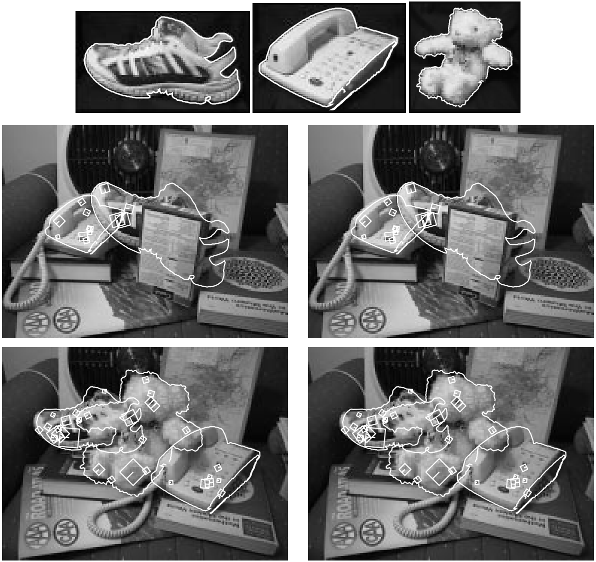
[Lowe 2004]

- a SIFT feature is determined by location, scale and orientation; a single feature correspondence can yield a 4-dof similarity transformation
- hypotheses: sparse Hough voting in 4-dimensional space; each correspondence casts a single vote in a hash table
- verification: on each bin with at least 3 votes, find inliers, form linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ and fit a 6-dof affine transformation by least-squares

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

Lowe. ICCV 1999. Object recognition from local scale-invariant features.

object recognition*

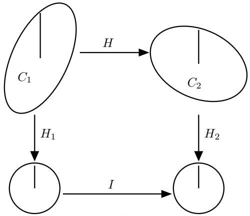


Lowe. ICCV 1999. Object recognition from local scale-invariant features.

fast spatial matching*

[Philbin et al. 2007]

Transformation	dof	Matrix
translation + isotropic scale	3	$\begin{bmatrix} a & 0 & t_x \\ 0 & a & t_y \end{bmatrix}$
translation + anisotropic scale	4	$\begin{bmatrix} a & 0 & t_x \\ 0 & b & t_y \end{bmatrix}$
translation + vertical shear	5	$\begin{bmatrix} a & 0 & t_x \\ b & c & t_y \end{bmatrix}$



- same idea, a single feature correspondence can yield a transformation that can be 3,4,5-dof
- but now use RANSAC where there is only one hypothesis per correspondence; all hypotheses can be enumerated and verified
- again, 6-dof fitting on inliers in the end
- so Hough can be seen as filtering of hypotheses by agreement

Philbin, Chum, Isard, Sivic and Zisserman. CVPR 2007. Object Retrieval With Large Vocabularies and Fast Spatial Matching.

object retrieval*

- image retrieval based on a bag-of-words representation
- fast spatial verification performed on top-ranking images

Philbin, Chum, Isard, Sivic and Zisserman. CVPR 2007. Object Retrieval With Large Vocabularies and Fast Spatial Matching.

summary

- derivatives as convolution
- edges: gradient magnitude and Laplacian
- scale-space and scale selection
- blobs: normalized Laplacian
- corners/junctions: windowed second moment matrix
- dense registration / sparse feature tracking
- wide-baseline matching by local features
- robust fitting: RANSAC*, Hough transform*
- Hough as cross-correlation*
- local shape for global transformation hypotheses*