Corners and Edge Detectors [1]

Lecture 07

See Sections 2.3.4 and 2.4 in
Reinhard Klette: Concise Computer Vision
Springer-Verlag, London, 2014

ccv.wordpress.fos.auckland.ac.nz

[1] See last slide for copyright information.

Agenda

Corner

A *corner* in an image I is given at a pixel p where two edges of different directions intersect.

Detected corners provide important information for localizing and understanding shapes in 3D scenes

Corner Detection Using the Hessian Matrix

A *corner* is characterized by high curvature of intensity values

Curvature can be measured by second-order derivatives

Hessian matrix at pixel location p:

$$\mathbf{H}(p) = \begin{bmatrix} I_{xx}(p) & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) \end{bmatrix}$$

Consider eigenvalues λ_1 and λ_2 of this matrix:

1. Magnitude of both eigenvalues is "large": a corner
2. One large and one small eigenvalue: a step edge
3. Two small eigenvalues: a low-contrast region.

Trace of a Matrix, Determinant, and Eigenvalues

Trace $\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}$ of an $n \times n$ matrix $\mathbf{A} = (a_{ij})$

Determinant $\det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21}$ of a 2×2 matrix $\mathbf{A} = (a_{ij})$ and

$$\det(\mathbf{A}) = a_{11}a_{22}a_{33}+a_{12}a_{23}a_{31}+a_{13}a_{21}a_{32}-a_{13}a_{22}a_{31}-a_{12}a_{21}a_{33}-a_{11}a_{23}a_{32}$$

of a 3×3 matrix $\mathbf{A} = (a_{ij})$

Eigenvalues of $n \times n$ matrix $\mathbf{A}$: n solutions of *characteristic polynomial*

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

$\mathbf{I}$ is $n \times n$ identity matrix

Square matrix $\mathbf{A}$: Determinant equal to product of eigenvalues, and trace equal to sum of eigenvalues

Corner Detector by Harris and Stephens (*Harris detector*)

Rather than considering $\mathbf{H}(p)$, just use products of first-order derivatives of smoothed version $L(p,\sigma) = [I * G_\sigma](p)$ for some $\sigma > 0$:

$$\mathbf{G}(p,\sigma) = \begin{bmatrix} L_x^2(p,\sigma) & L_x(p,\sigma)L_y(p,\sigma) \\ L_x(p,\sigma)L_y(p,\sigma) & L_y^2(p,\sigma) \end{bmatrix}$$

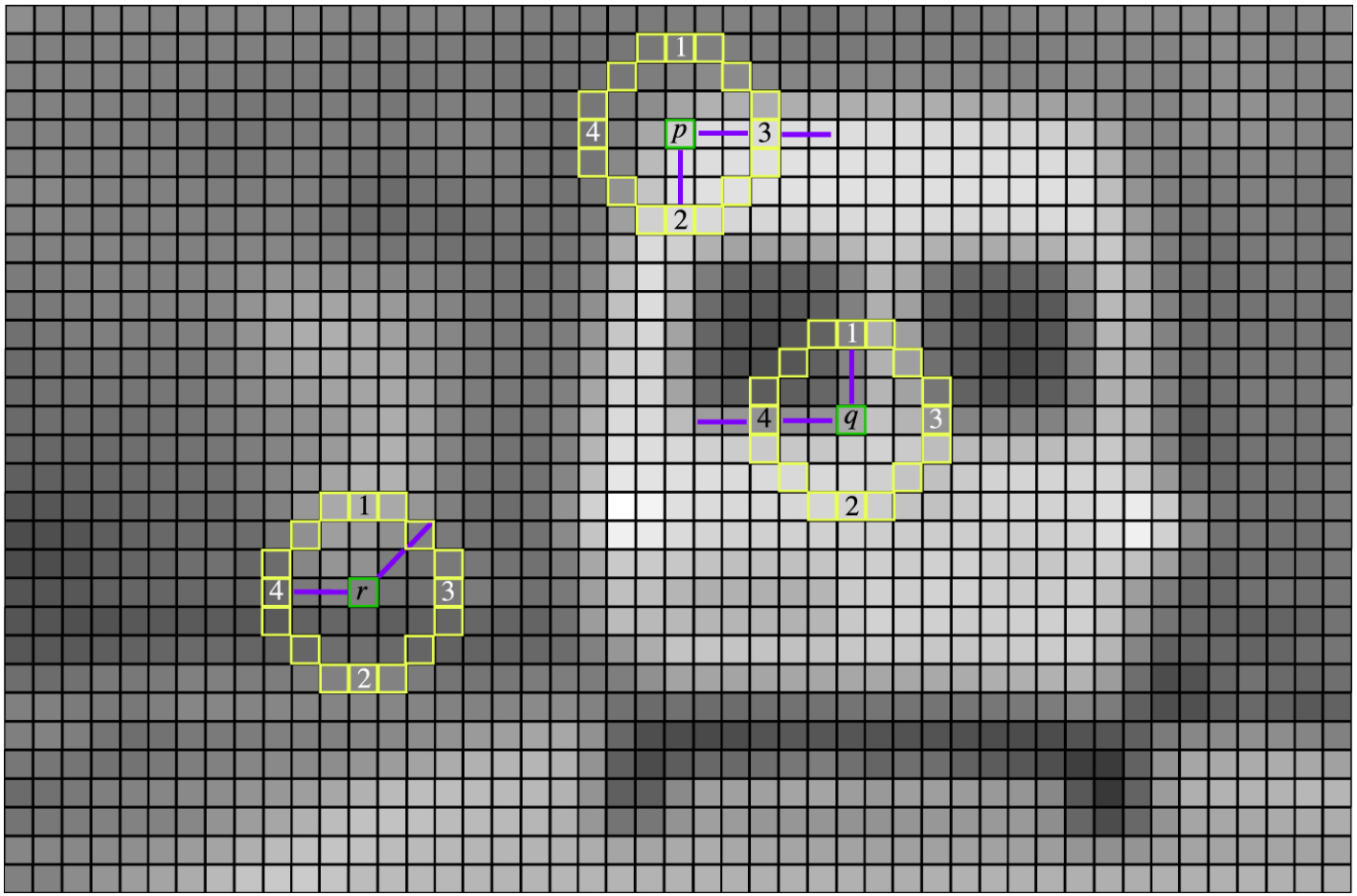
Eigenvalues λ_1 and λ_2 of $\mathbf{G}$ represent changes in intensities in orthogonal directions in image I

Instead of calculating eigenvalues, consider the *cornerness measure*

$$\mathcal{H}(p,\sigma,a) = \det(\mathbf{G}) - a \cdot \mathrm{Tr}(\mathbf{G}) = \lambda_1\lambda_2 - a \cdot (\lambda_1 + \lambda_2)$$

for small $a > 0$ (e.g. $a = 1/25$)

Corners in FAST



p, q, and r are at intersections of edges

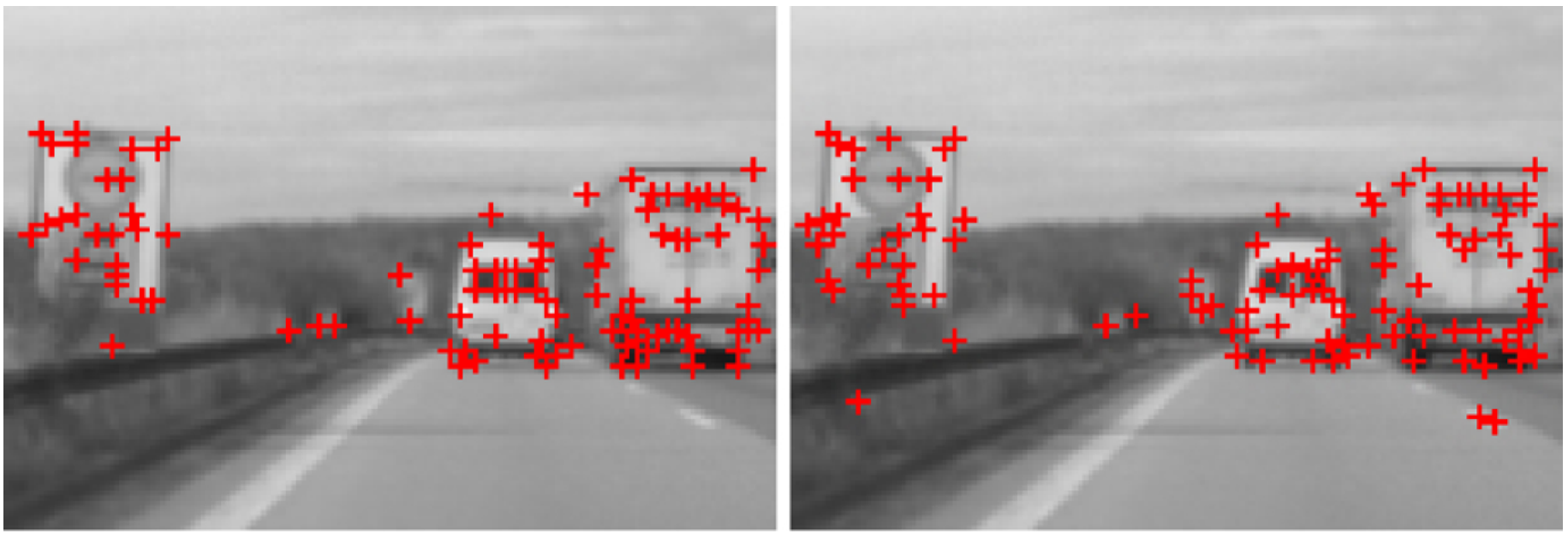
Directions of those edges are indicated by *blue lines*

Discrete circles (of 16 pixels) as used for FAST corner detector

Features from an Accelerated Segment Test (FAST)

FAST identifies a corner by considering image values on a digital circle around the given pixel location p

Cornerness test: Value at centre pixel is darker (or brighter) compared to more than 8 ... 11 subsequent pixels on the circle, and "similar" to values at remaining pixels on the circle



Left: Detected corners using the Harris detector
Right: Corners detected by FAST

Agenda

Estimating Gradient Magnitude and Direction

Canny operator maps a scalar image into a binary edge map of one-pixel-wide edge segments

Output depends on used gradient estimator, thresholds T_{low} and T_{high}, $0 < T_{low} < T_{high} < G_{\max}$, and a chosen scale σ for Gaussian smoothing

Let I be the already smoothed input image, after applying a convolution with a Gauss function G_σ of scale $\sigma > 0$

Apply a basic gradient estimator (e.g. Sobel operator) for $[I_x, I_y]^\top$

Use estimate $g(p) = \sqrt{I_x^2 + I_y^2}$ or $g(p) = |I_x| + |I_y|$ for gradient magnitude and estimate $\theta(p) = \operatorname{atan2}(I_y, I_x)$ for gradient direction

Estimates $\theta(p)$ rounded to multiples of $\pi/4$ by $(\theta(p) + \pi/8) \operatorname{modulo} \pi/4$

Non-maxima Suppression and Outline of Edge Following

In a step of *non-maxima suppression* it is tested whether a value $g(p)$ is maximal in the (now rounded) direction $\theta(p)$

For example, if $\theta(p) = \pi/2$, i.e. the gradient direction at $p = (x, y)$ is downward, then $g(p)$ is compared against $g(x, y-1)$ and $g(x, y+1)$, the values above and below of p

If $g(p)$ is not larger than the values at both of those adjacent pixels, then $g(p)$ becomes 0

Final step of *edge following*:

Trace paths of pixel locations p with $g(p) > T_{low}$, mark path as being an edge; start such a trace at a location p with $g(p) \geq T_{high}$. – See next slide.

Details for Edge Following

Scan Ω left-to-right, top-down

Arrive at a not yet marked pixel p with $g(p) \geq T_{high}$:

1. Mark p as an edge pixel
2. If $q \in A_8(p)$ with $g(q) > T_{low}$ then mark q as an edge pixel
3. Call q now p and go back to Step 2.

Search for next start pixel p until end of Ω is reached. Used selection principle in Step 2 influences the created paths.

Both used thresholds define *hysteresis* :

Following q may not have a value above T_{high}, but it had at least one predecessor on the same path with value above T_{high}; this "positive" history supports the decision at q: $g(q) > T_{low}$ is sufficient for continuation

Agenda

LoG Edge Detector

Applying the Laplacian for a Gauss-filtered image can be done in one step of convolution, based on the theorem

$$\nabla^2(G_\sigma * I) = I * \nabla^2 G_\sigma$$

where $*$ denotes the convolution of two functions, and I is assumed to be twice differentiable; the zero-crossings define the edges

$$\textbf{LoG} = \text{Laplacian of Gaussian}$$

Proof: Apply twice the following rule

$$D(F * H) = D(F) * H = F * D(H)$$

of convolutions where D denotes a derivative, and F and H are differentiable functions; also note that convolution is commutative:

$$D^2(G_\sigma * I) = D^2(I * G_\sigma) = D(D(I * G_\sigma)) = D(I * D(G_\sigma)) = I * D^2(G_\sigma)$$

Laplacian of Gaussian (LoG)

Thus: Calculation of LoG of I only needs one convolution with $\nabla^2 G_\sigma$

For deriving a filter kernel for $\nabla^2 G_\sigma$, first we calculate the first partial derivatives of

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

e.g. with respect to x:

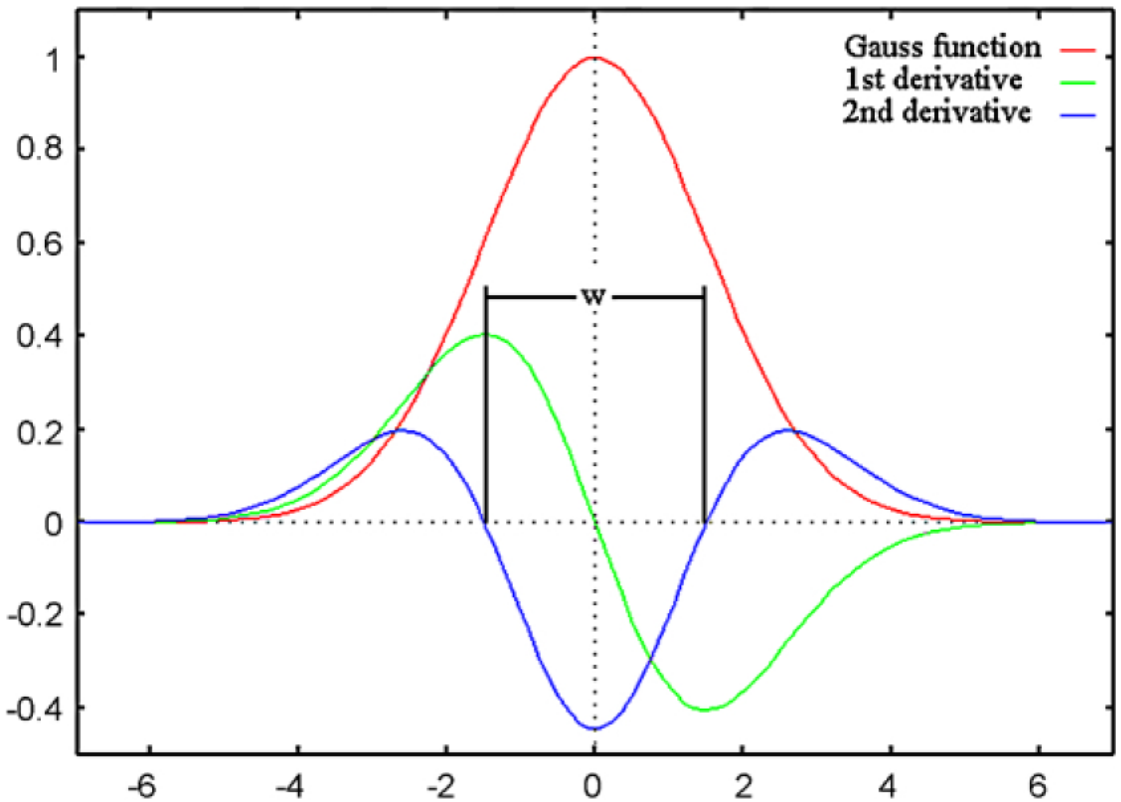
$$\frac{\partial G_\sigma}{\partial x}(x, y) = \frac{1}{2\pi\sigma^2} \cdot \frac{-2x}{2\sigma^2} \cdot e^{-(x^2+y^2)/2\sigma^2} = -\frac{x}{2\pi\sigma^4} e^{-(x^2+y^2)/2\sigma^2}$$

Repeat the derivative for x and y and obtain the LoG

$$\nabla^2 G_\sigma(x, y) = \frac{1}{2\pi\sigma^4} \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^2} \right) e^{-(x^2+y^2)/2\sigma^2}$$

also known as *Mexican hat function* (in fact, an "inverted Mexican hat")

Inverted Mexican Hat



1D Cuts through the function graph of G and its subsequent derivatives
[2D Gauss function is rotationally symmetric with respect to origin $(0, 0)$]

Advice on Sampling the LoG Kernel

Task: sample the LoG into a $(2k+1) \times (2k+1)$ filter kernel

What is an appropriate value for k?

Estimate σ for the given images; appropriate value of k follows from σ:

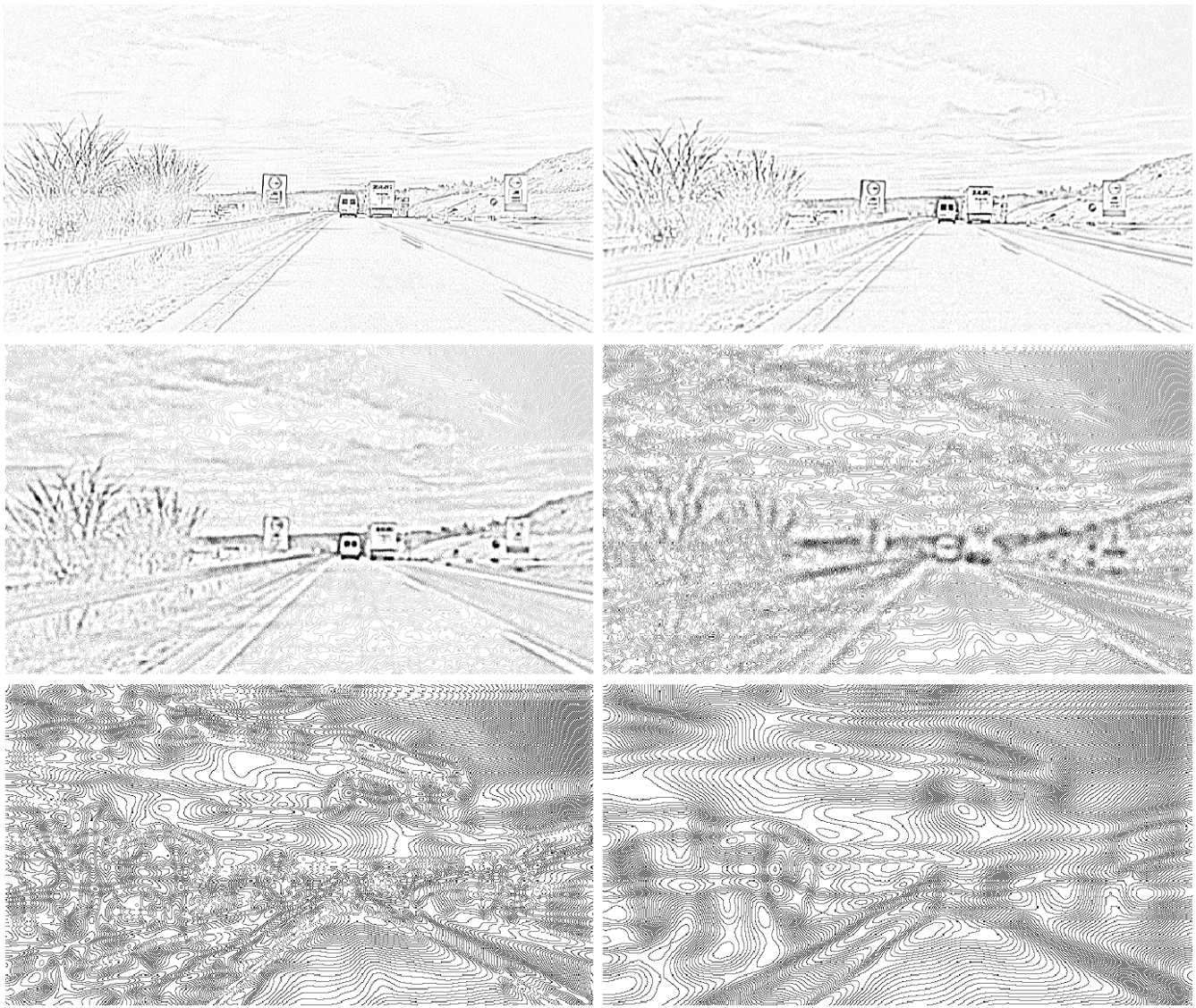
1. Parameter w defined by zero-crossings of $\nabla^2 G_\sigma(x, y)$
2. Let $\nabla^2 G_\sigma(x, y) = 0$ and $y = 0$:
   zero-crossings at $x^2 = 2\sigma^2$, i.e. $x_1 = -\sqrt{2}\sigma$ and at $x_2 = +\sqrt{2}\sigma$
3. Thus: $w = |x_1 - x_2| = 2\sqrt{2}\sigma$
4. Sample in window of size $3w \times 3w = 6\sqrt{2}\sigma \times 6\sqrt{2}\sigma$
5. $2k+1 \times 2k+1 = \mathrm{ceil}\big(6\sqrt{2}\sigma\big) \times \mathrm{ceil}\big(6\sqrt{2}$

where ceil is the smallest integer equal to, or larger than the argument

Increasingly Smoothed Images: $\sigma = 0.5, 1, 2, 4, 8,$ or 16

Six layers in the *Gaussian scale space* of image `Set1Seq1`

Six Layers in LoG Scale Space: $\sigma = 0.5, 1, 2, 4, 8,$ or 16



LoG Scale Space

Laplacians of images shown on Page 18 represent six layers in the *LoG scale space* of image `Set1Seq1`

Linear scaling was applied to all the shown images for making the intensity patterns visible.

The *scale* is defined by the used standard deviation σ.

As in a Gaussian scale space, each layer is defined by the scale σ.

It is common to generate subsequent layers by starting with an initial scale σ, and then using subsequent scales

$$a^n \cdot \sigma$$

for $a > 1$ and $n = 0, 1, \ldots, m$

Difference of Gaussians (DoG)

The *difference of Gaussians* (DoG) is a common approximation of LoG

See Page 6 for layer L in the Gaussian scale space.

DoG defined by initial scale σ and scaling factor $a > 1$:

$$D_{\sigma,a}(x, y) = L(x, y, \sigma) - L(x, y, a\sigma)$$

As for LoG, edges (following the step-edge model) are at zero-crossings

Relation between LoG and DoG:

$$\nabla^2 G_\sigma(x, y) \approx \frac{G_{a\sigma}(x, y) - G_\sigma(x, y)}{(a - 1)\sigma^2}$$

with $a = 1.6$ as a recommended parameter for approximation

DoG*s* are used in general as time-efficient approximations of LoG*s*

Calculations for Relation between LoG and DoG

The DoG represents an approximation of a convolution of I with the partial derivative

$$\begin{aligned}\frac{\partial G_\sigma}{\sigma} &= -\frac{1}{\pi\sigma^3}\cdot e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{x^2+y^2}{2\pi\sigma^5}\cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \sigma\cdot\left(\frac{x^2+y^2-2\sigma^2}{\sigma^2}\right)\cdot\frac{1}{2\pi\sigma^4}\cdot e^{-(x^2+y^2)/2\sigma^2} \\ &= \sigma\cdot\nabla^2 G_\sigma\end{aligned}$$

Using

$$\frac{\partial G_\sigma}{\sigma} \approx \frac{G_{a\sigma}(x,y)-G_\sigma(x,y)}{a\sigma-\sigma} = \frac{G_{a\sigma}(x,y)-G_\sigma(x,y)}{(a-1)\sigma}$$

it follows that

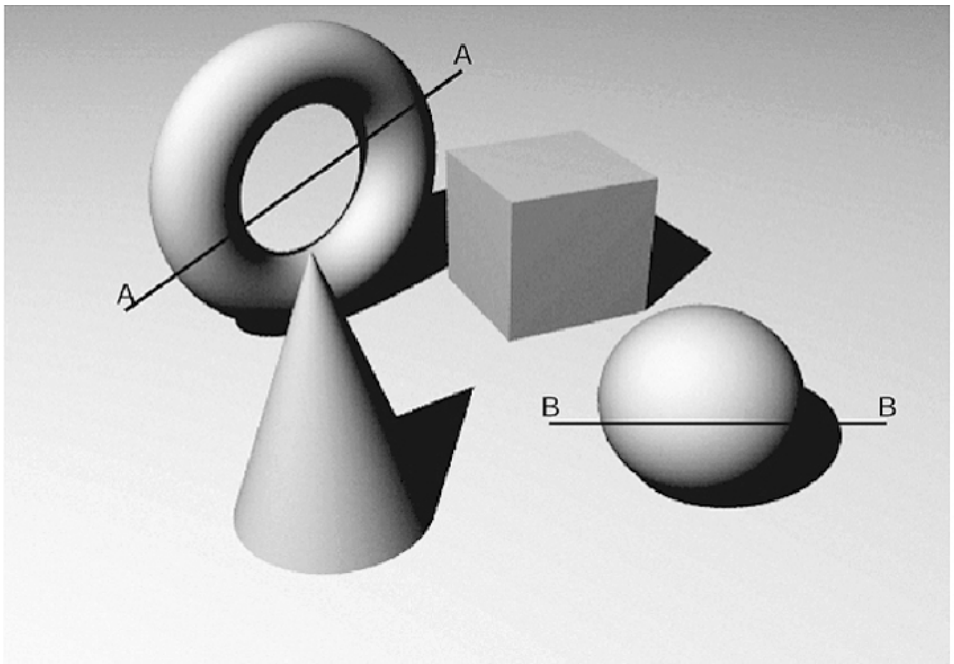
$$G_{a\sigma}(x,y)-G_\sigma(x,y) \approx (a-1)\sigma\frac{\partial G_\sigma}{\sigma} = (a-1)\sigma^2\cdot\nabla^2 G_\sigma$$

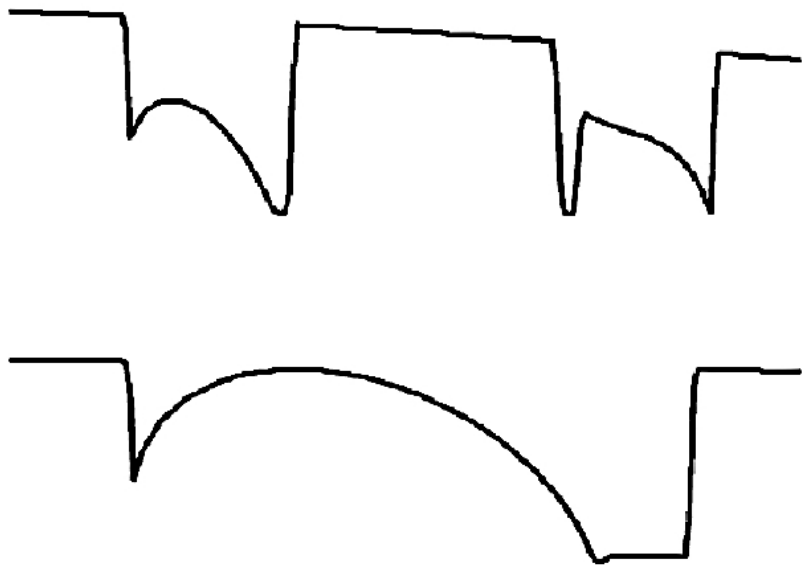
Left: Log, and Right: DoG Scale Space

$\sigma = 0.5$, $a_n = 1.6^n$; figure shows results for $n = 1$, $n = 3$, and $n = 5$

Agenda

Step-Edge or Phase-Based Model?

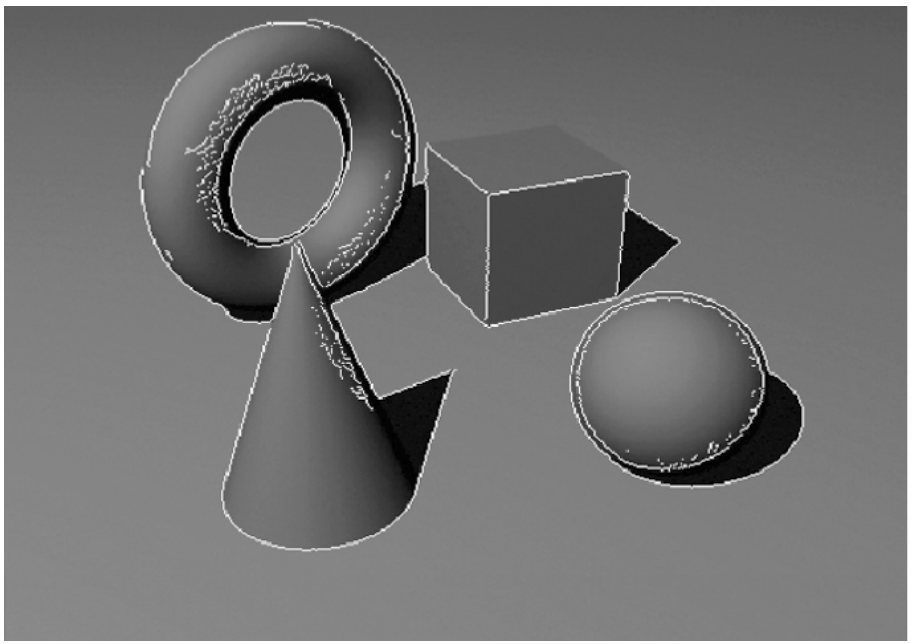


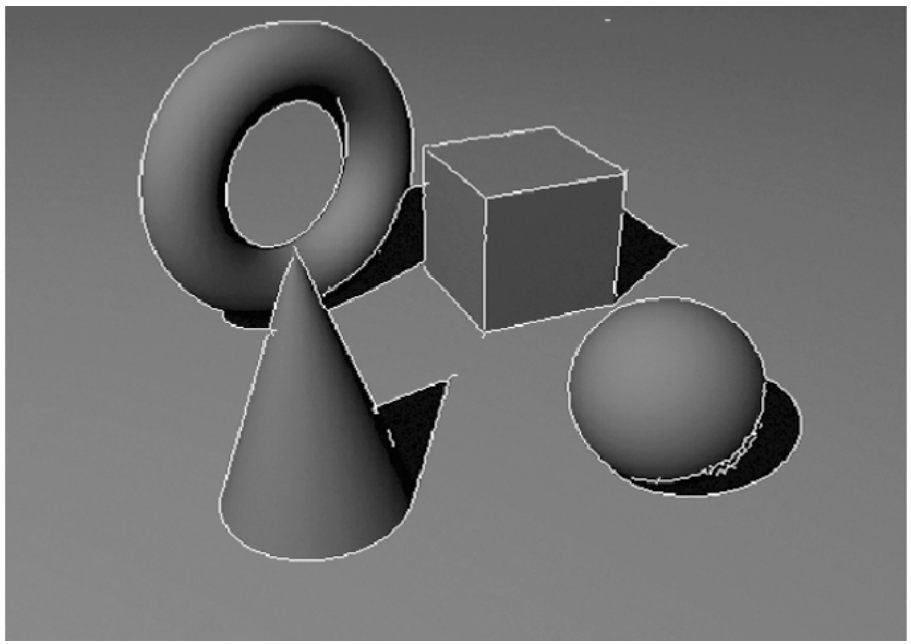


Left: Synthetic input image
Right: Intensity profiles, Section A–A (*top*), Section B–B (*bottom*)

Differences between step-edge operators and phase-based operators are better visible for a simple synthetic input image as the one shown above

Detected Edges



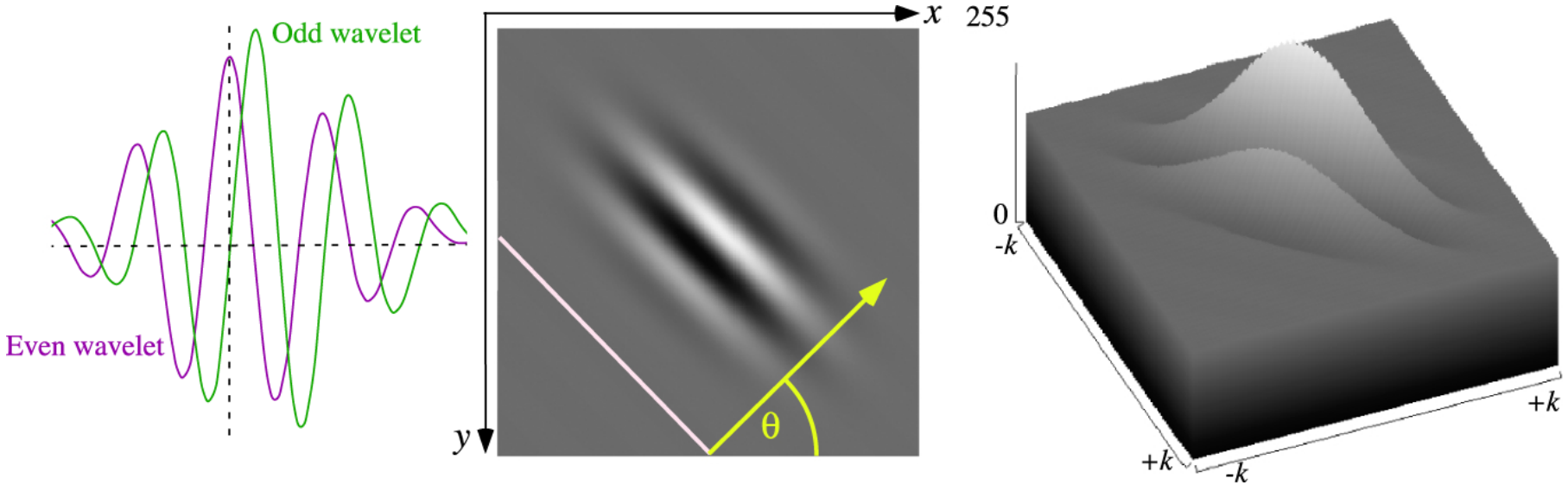


Left: Edges detected by the Canny operator (i.e. use of step-edge model)

Right: Edges detected by the Kovesi algorithm (i.e. phase-based)

Gabor Wavelets

Local analysis of frequency components not based on wave patterns that run uniformly through the whole $(2k+1) \times (2k+1)$ window but rather *wavelets*, such as *Gabor wavelets*, which are sine or cosine waves modulated by a Gauss function of some scale σ



Left: Two 1D cuts through an odd and an even Gabor wavelet
Middle: Gabor wavelet in $(2k+1) \times (2k+1)$ window with direction θ
Right: 3D surface plot; decreasing amplitudes around a centre point

Odd and Even Gabor Wavelets

Odd wavelet: Generated from a sine wave, thus having value 0 at the origin
Even wavelet: Generated from a cosine wave; maximum at the origin

Recall:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

and use rotation (θ is orthogonal to the stripes in the Gabor wavelets)

$$\begin{aligned} u &= x\cos\theta + y\sin\theta \\ v &= -x\sin\theta + y\cos\theta \end{aligned}$$

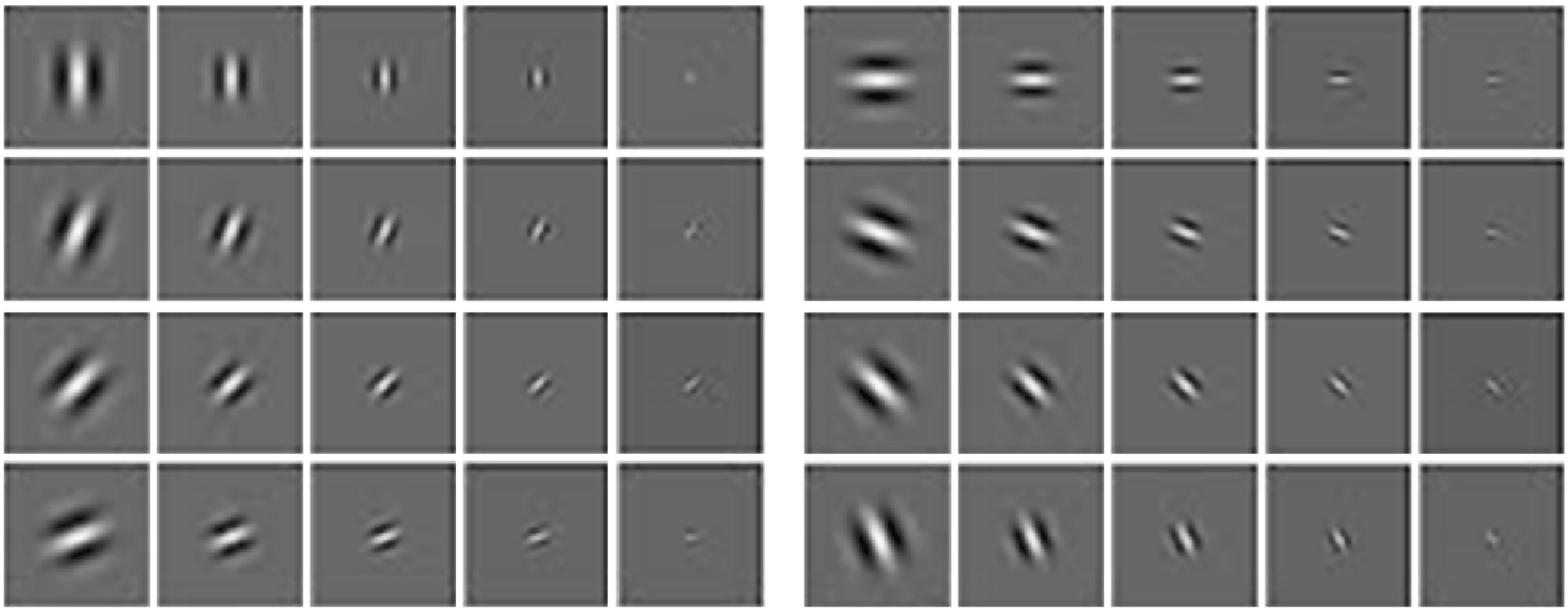
Gabor pair

$$\begin{aligned} g_{even}(x,y) &= G_\sigma(u,\gamma v)\cdot\cos\left(2\pi\frac{u}{\lambda}+\psi\right) \\ g_{odd}(x,y) &= G_\sigma(u,\gamma v)\cdot\sin\left(2\pi\frac{u}{\lambda}+\psi\right) \end{aligned}$$

for phase-offset $\psi \geq 0$, wavelength $\lambda > 0$, spatial aspect ratio $\gamma > 0$

Preparing for the Algorithm

Kovesi algorithm applies a set of n square Gabor pairs centred at the current pixel location $p = (x, y)$



Here: $n = 40$ and illustrating only one function (say, the odd wavelet) for each pair; the Kovesi algorithm uses $n = 24$ as default

Local Convolution

A local convolution with each Gabor pair defines one complex number

We obtained n complex numbers with amplitude r_h and phase α_h

Ideal phase congruency measure; z be the sum of all $[r_h, \alpha_h]^\top$ vectors:

$$\mathcal{P}_{phase}(p) = \frac{\mathrm{pos}(\|z\|_2 - T)}{\sum_{h=1}^{n} r_h + \varepsilon}$$

Small positive number ε: for cases where $\sum_{h=1}^{n} r_h$ becomes very small

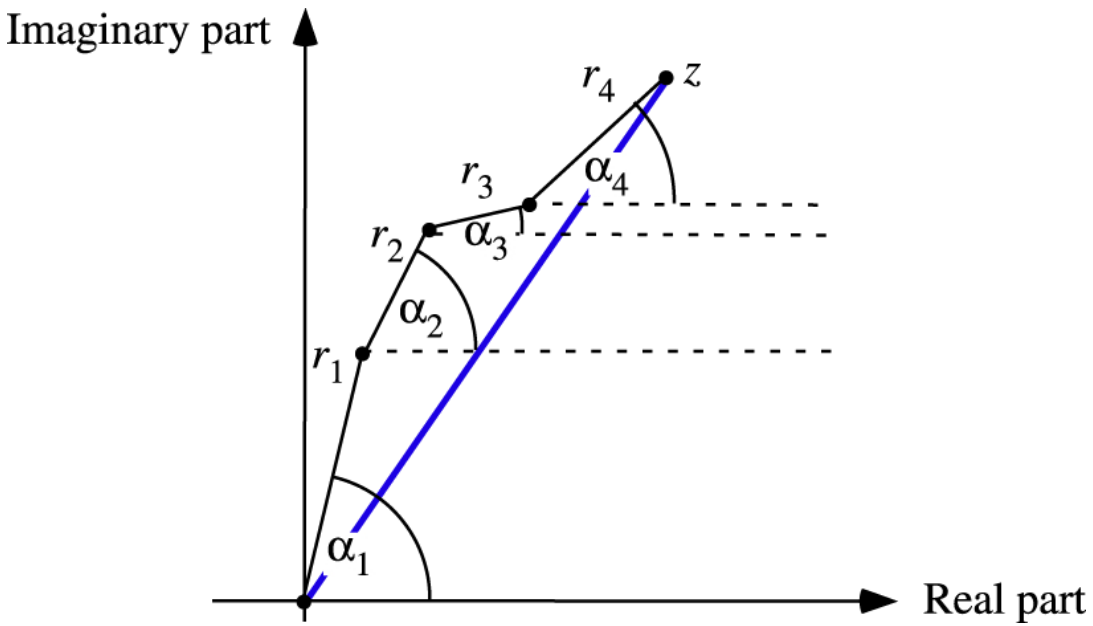
$T > 0$ be the estimated sum of all noise responses over all AC components

Function pos returns the argument if positive and 0 otherwise

Thus

$$0 \leq \mathcal{P}_{phase}(p) \leq 1$$

Sum of Vectors



If all into the same direction (i.e. phase congruency) then a large value of $\|z\|_2$ compared to the sum of all amplitudes

Processing at One Pixel

Analysing phase congruency at a given pixel location $p = (x, y)$:

1. Apply at p the set of convolution masks of $n = m_1 \cdot m_2$ Gabor pairs producing n complex numbers (r_h, α_h)
2. Calculate phase congruency measures $\mathcal{P}_i(p)$, $1 \leq i \leq m_1$, by only using m_2 complex numbers (r_h, α_h) defined for direction θ_i
3. Calculate directional components X_i and Y_i for $1 \leq i \leq m_1$ by

$$[X_i, Y_i]^\top = \mathcal{P}_i(p) \cdot \big[\sin(\theta_i), \cos(\theta_i)\big]^\top$$

4. Consider *covariance matrix* of directional components:

$$\begin{bmatrix} \sum_{i=1}^{m_1} X_i^2 & \sum_{i=1}^{m_1} X_i Y_i \\ \sum_{i=1}^{m_1} X_i Y_i & \sum_{i=1}^{m_1} Y_i^2 \end{bmatrix}$$

Calculate eigenvalues λ_1 and λ_2; let $\lambda_1 \geq \lambda_2$
(Note the correspondence to the approximated 2×2 Hessian matrix)

Detection of Edge Pixels

Magnitude of λ_1 indicates significance of a local feature (an edge, corner, or another local feature)

If λ_2 also of large magnitude then we have a corner

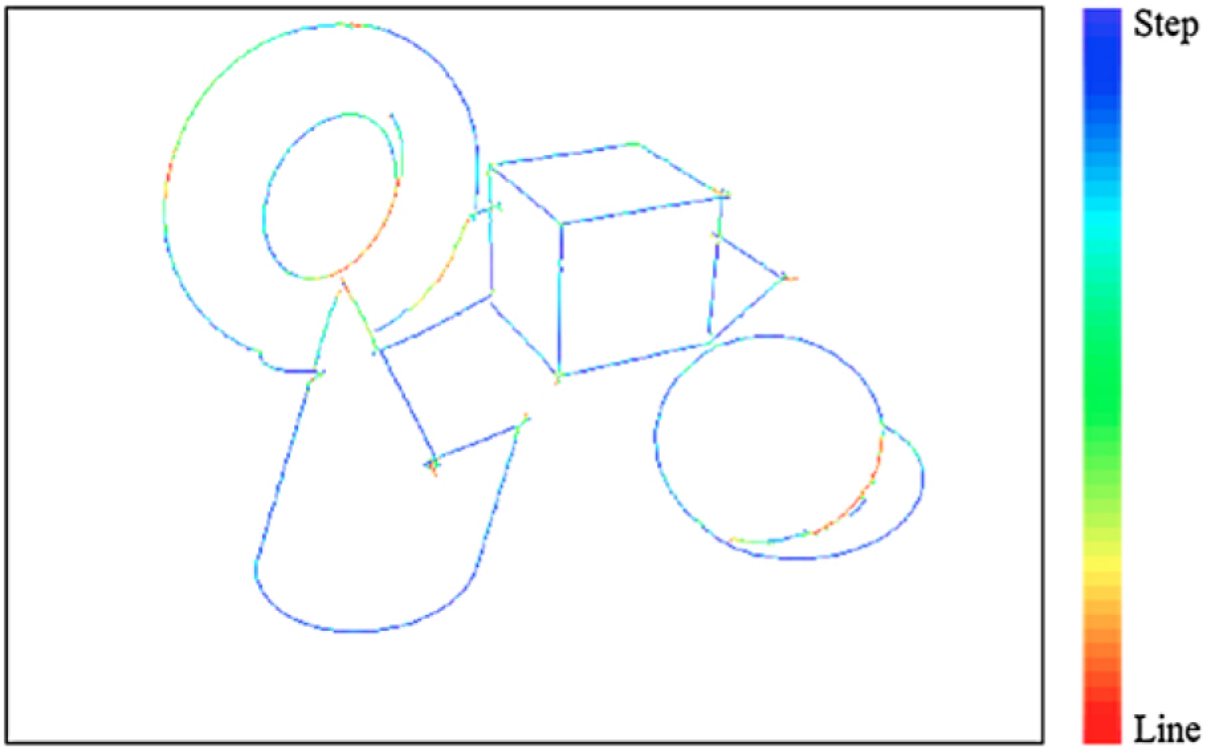
Principle axis corresponds with the direction of the local feature

Apply procedure for all $p \in \Omega$: Array of λ_1 values defines *raw results*; ignore p if value below a chosen cut-off threshold (say, 0.5)

Perform non-maxima suppression in this array, possibly combined with hysteresis thresholding, i.e. set to zero all values that do not define a local maximum in their (say) 8-neighbourhood

Pixels having non-zero values after the non-maxima suppression are the identified edge pixels

Edge or Line Pixel ?



Colour-code for a scale between “Step” and “Line”

Having two eigenvalues as results for each pixel, these two values can also be used for classifying a detected feature; the figure illustrates just an example

Copyright Information

This slide show was prepared by Reinhard Klette
with kind permission from Springer Science+Business Media B.V.

The slide show can be used freely for presentations.
However, *all the material* is copyrighted.

In case of citation: just cite the book, that's fine.