MLRF Lecture 03

J. Chazalon, LRDE/EPITA, 2019

Local feature detectors

Lecture 03 part 02

Some classical detectors

Edge (gradient detectors)

- Sobel
- Canny

Corner

- Harris & Stephens
- FAST
- Laplacian of Gaussian, Difference of Gaussian, Determinant of Hessian

Blob

- MSER

Harris & Stephens Conclusion

Good features to track *aka* Shi-Tomasi *aka* Kanade-Tomasi

Remember the Harris-Stephens trick to avoid computing the eigenvalues?

$$M_c = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \underbrace{\det(A) - \kappa \operatorname{trace}^2(A)}_{\text{approximation}}$$

Well, nowadays, linear algebra is cheap, so **compute the real eigenvalues**.

Then filter using $min(\lambda_1, \lambda_2) > \lambda$, λ being a predefined threshold.

You get the Shi-Tomasi variant.

Jianbo Shi and Carlo Tomasi. Good features to track. In Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on, pages 593–600. IEEE, 1994.

Build your own edge/corner detector

Hessian matrix with block-wise summing

You just need eigenvalues λ_1 and λ_2 of the structure tensor

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} \frac{\partial I^2(x+u,y+v)}{\partial x} & \frac{\partial I(x+u,y+v)}{\partial x} \frac{\partial I(x+u,y+v)}{\partial y} \\ \frac{\partial I(x+u,y+v)}{\partial x} \frac{\partial I(x+u,y+v)}{\partial y} & \frac{\partial I^2(x+u,y+v)}{\partial y} \end{bmatrix}$$

$$= \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

```
dst = cv2.cornerEigenValsAndVecs(src, neighborhood_size, sobel_aperture)
dst = cv2.cornerMinEigenVal(src, neighborhood_size, sobel_aperture)
```

Harris summary

Pros

Translation invariant
⇒ Large gradients in both directions = stable point

Cons

Not so fast
⇒ Avoid to compute all those derivatives

Not scale invariant
⇒ Detect corners at different *scales*

Not rotation invariant
⇒ Normalization orientation

Corner detectors, binary tests
FAST

Features from accelerated segment test (FAST)

Keypoint detector used by ORB (described in part 3 of this lecture)

Segment test:
compare pixel ***P*** intensity I_p with surrounding pixels (circle of 16 pixels)

If ***n*** contiguous pixels are either

- all darker than $I_p - t$
- all brighter than $I_p + t$

then ***P*** is a detected as a corner

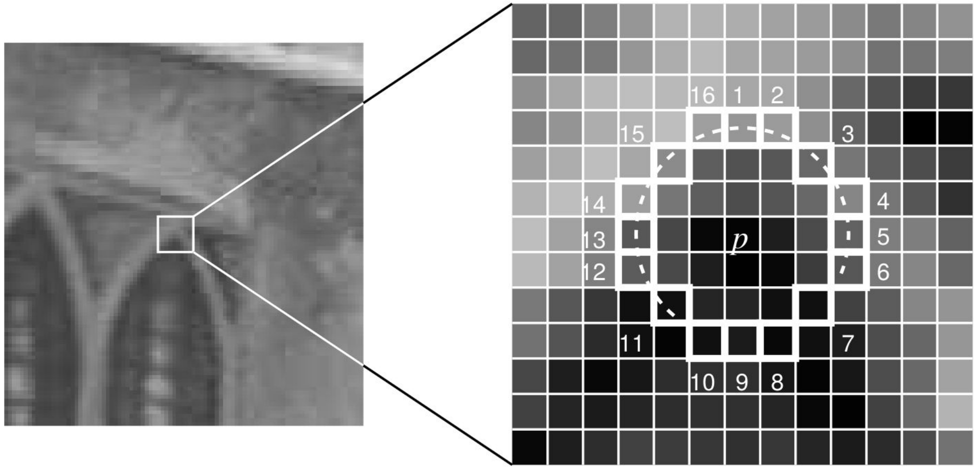


Figure 1. 12 point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at p is the centre of a candidate corner. The arc is indicated by the dashed line passes through 12 contiguous pixels which are brighter than p by more than the threshold.

E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in European conference on computer vision, 2006, pp. 430–443.

Tricks

1. **Cascading:** If n = 12 (¾ of the circle), then many non-corners can be discarded by testing pixels at the 4 compass directions. The full test is only applied to the candidates which passed the first test.
2. **Machine learning:** Learn on a dataset which pixels should be tested first to discard a non-corner as quickly as possible.
   Learn a decision tree, then compile the decisions as nested if-then rules.
3. How to perform **non-maximal suppression**?
   Need to assign a score ***V*** to each corner.
   ⇒ The sum of the absolute difference between the pixels in the contiguous arc and the centre pixel

$$V = \max\left(\sum_{x \in S_{\mathrm{bright}}} |I_{p\to x} - I_p| - t \ , \ \sum_{x \in S_{\mathrm{dark}}} |I_p - I_{p\to x}| - t \right)$$

FAST summary

Pros

Very fast

Authors tests:

- 20 times faster than Harris
- 40 times faster than DoG

Very robust to transformations (perspective in particular)

Cons

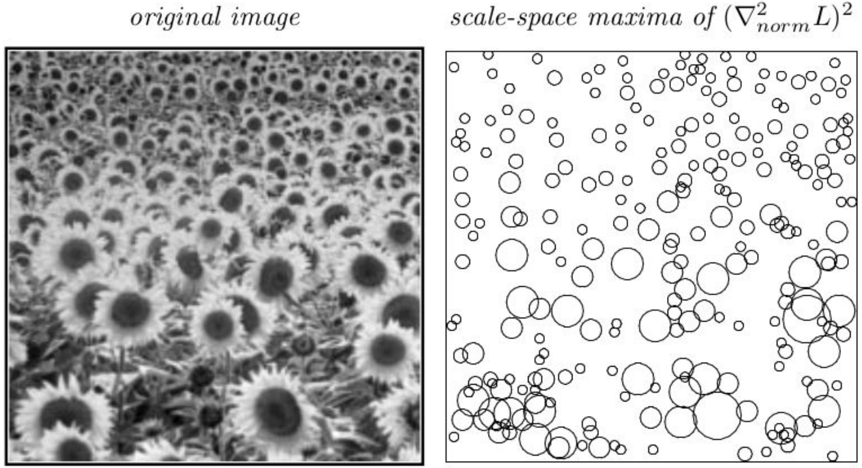
Very sensitive to blur

Corner detectors at different scales
DoG, LoG, DoH

Laplacian of Gaussian (LoG)

The theoretical, slow way.

If you need to remember only 1 thing:
it is a **band-pass filter** – it **detects objects of a certain size**.



T. Lindeberg, "Feature Detection with Automatic Scale Selection," Int. J. of Computer Vision, vol. 30, no. 2, p. 53, 1998.

Laplacian = second derivative

Second derivative of an image? Like Sobel… with 1 more derivation…

Taylor, again:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + O(h^4)$$

$$\overset{\text{add}}{+}\left[f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + O(h^4) \right]$$

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4)$$

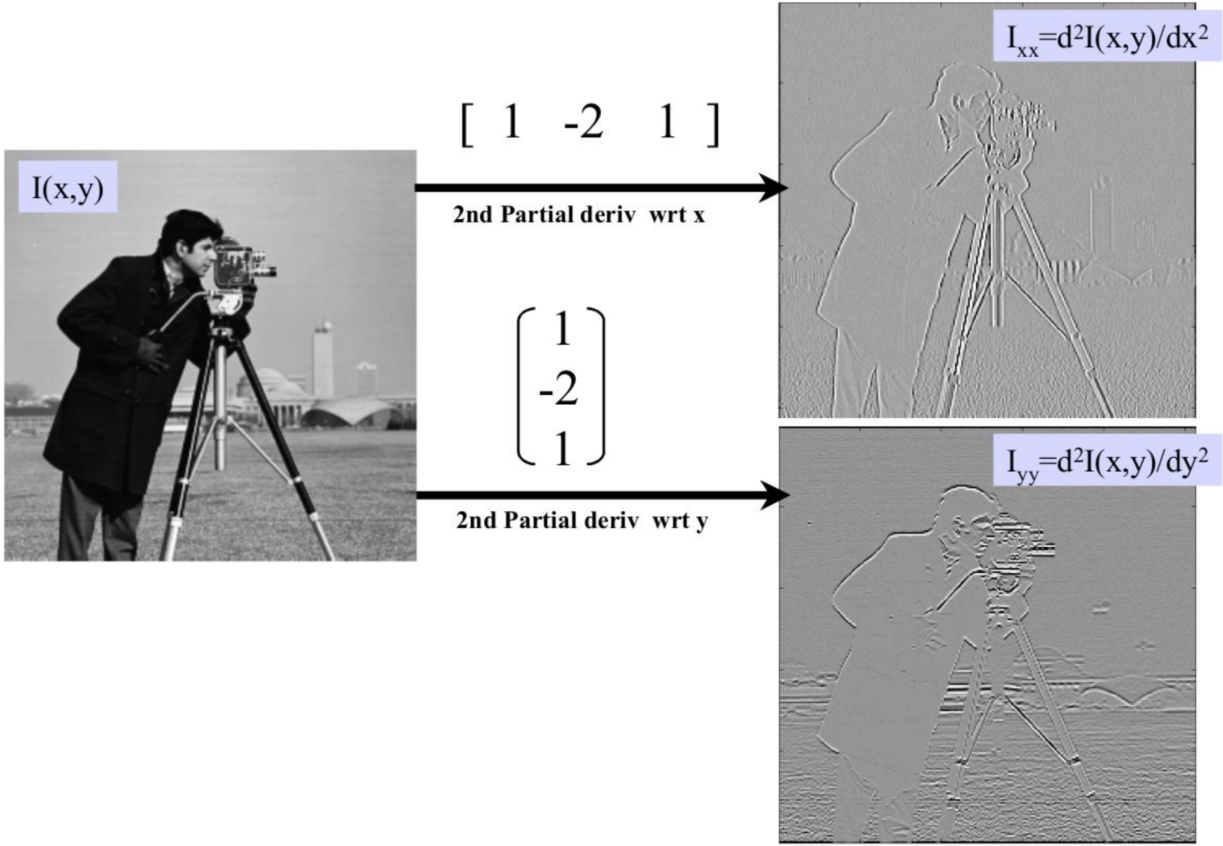
$$\frac{f(x-h) - 2f(x) + f(x+h)}{h^2} = f''(x) + O(h^2)$$

New filter: $\boldsymbol{I_{xx}}$ =

1	-2	1

$* \boldsymbol{I}$

Second partial derivatives of an image



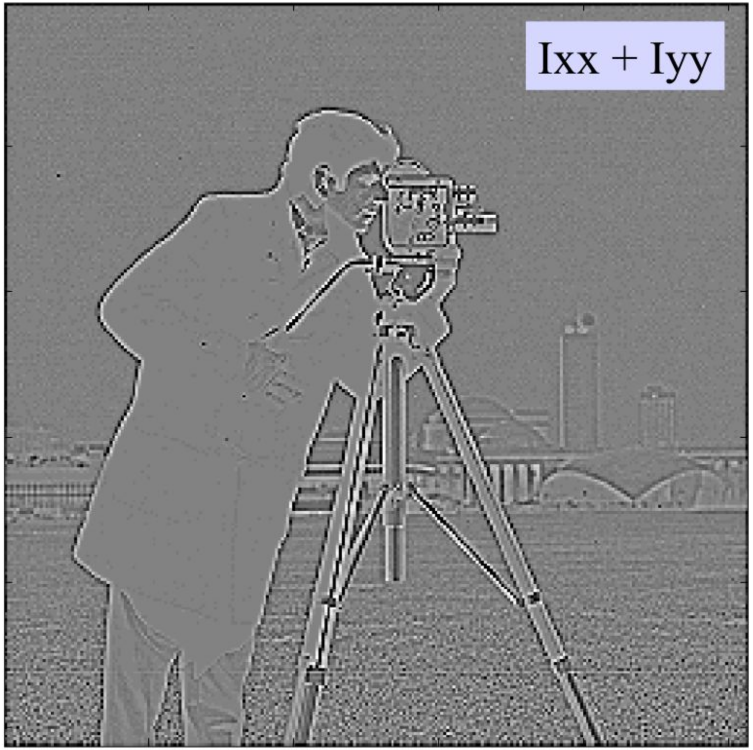
Laplacian filter ∇^2 I(x,y)

Edge detector, like Sobel but with 2nd derivatives

$$I_{xx}+I_{yy}=\left(\begin{bmatrix}1 & -2 & 1\end{bmatrix}+\begin{bmatrix}1\\-2\\1\end{bmatrix}\right)*I$$

$$=\begin{bmatrix}0 & 1 & 0\\1 & -4 & 1\\0 & 1 & 0\end{bmatrix}*I$$

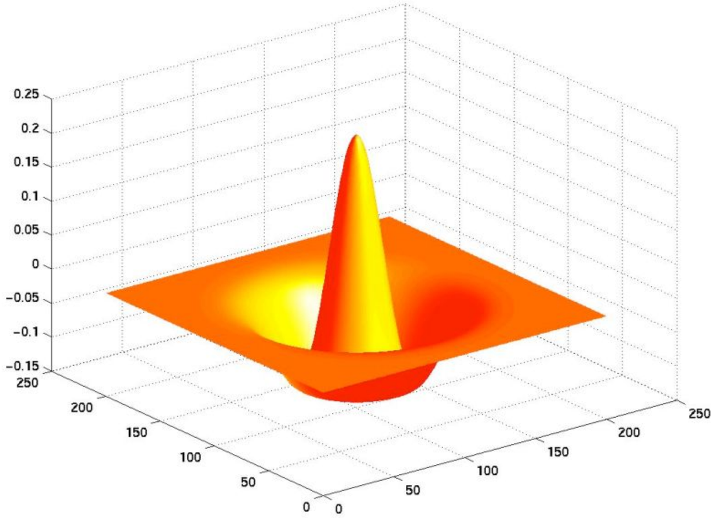
$$\begin{bmatrix}0 & 1 & 0\\1 & -4 & 1\\0 & 1 & 0\end{bmatrix}*I$$



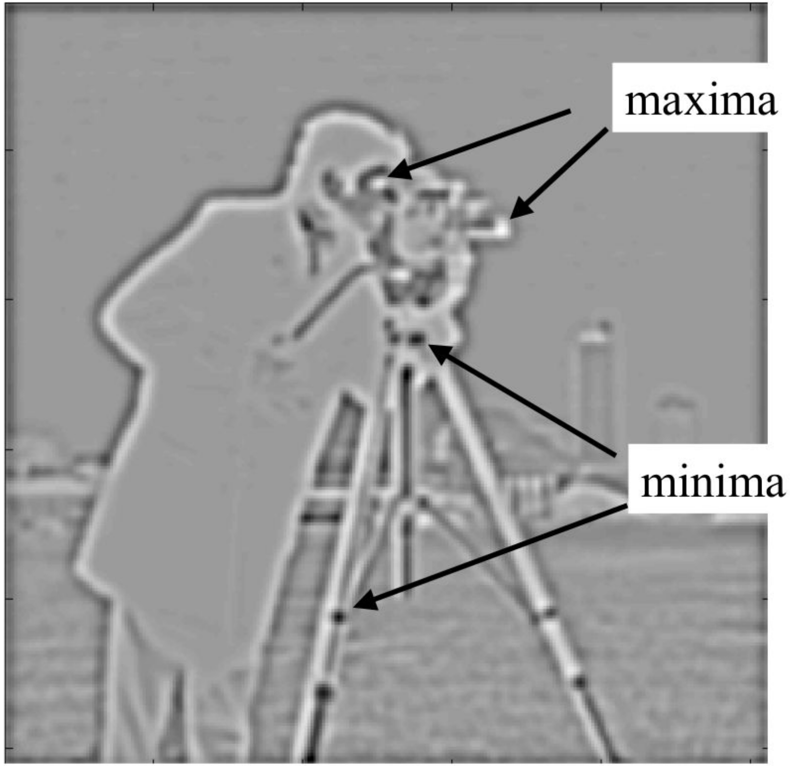
Laplacian of Gaussian

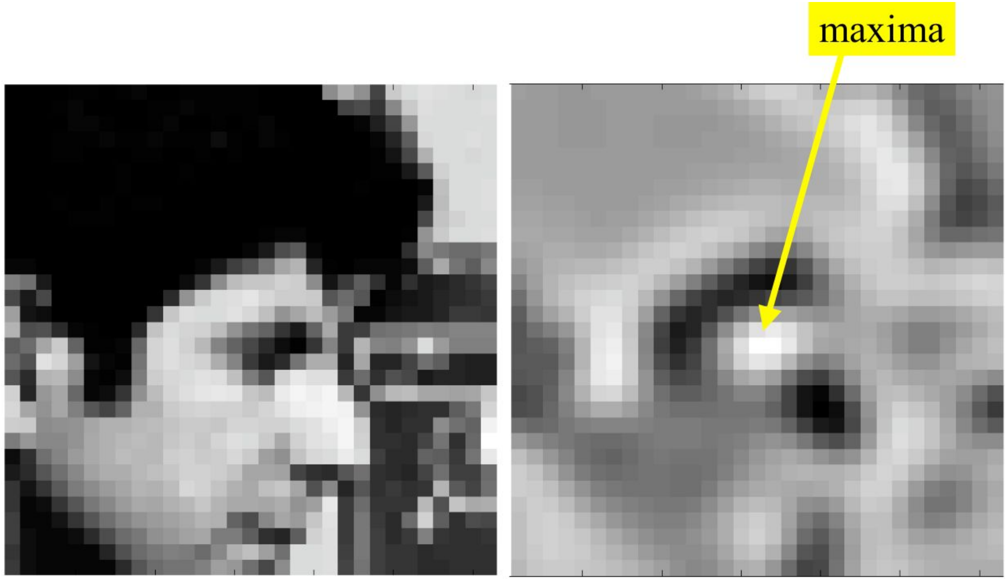
Second derivative of a Gaussian: **“Mexican hat”**

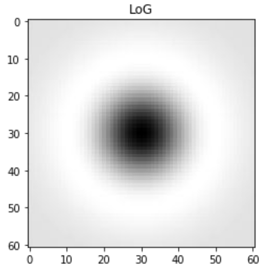
$$g''(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right)e^{-\frac{x^2}{2\sigma^2}}$$



LoG = detector of circular shapes





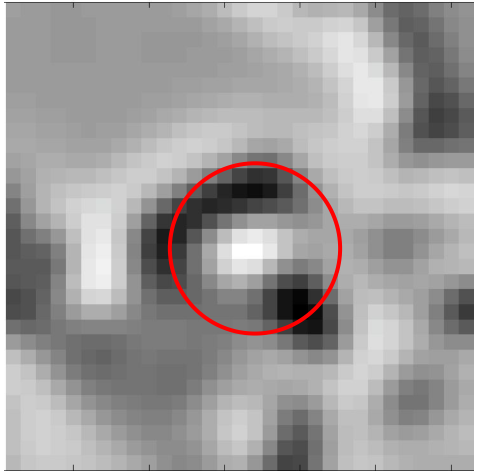


LoG = detector of circular shapes

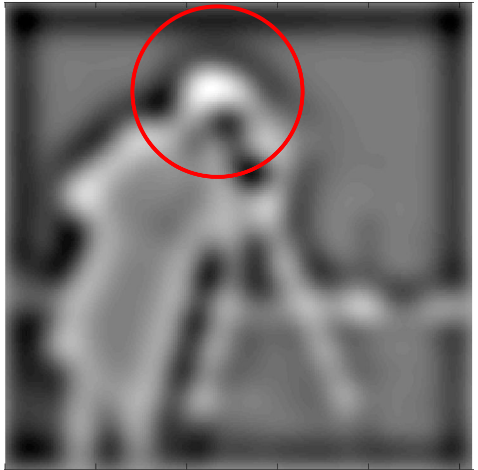
LoG filter extrema locates “blobs”

- maxima = dark blobs on light background
- minima = light blobs on dark background

Scale of blob (size ; radius in pixels) is determined by the **sigma** parameter of the LoG filter.



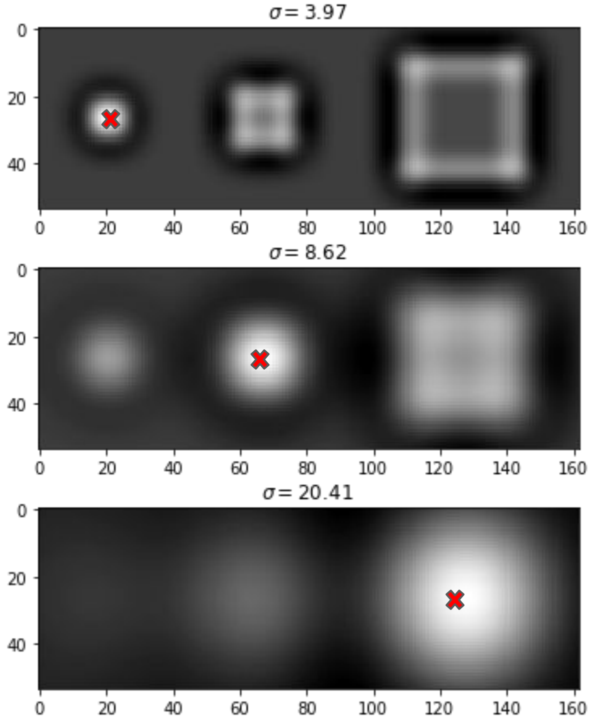
LoG σ=2

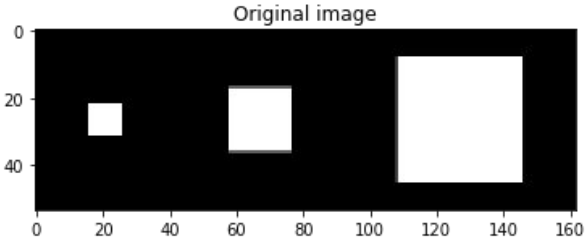


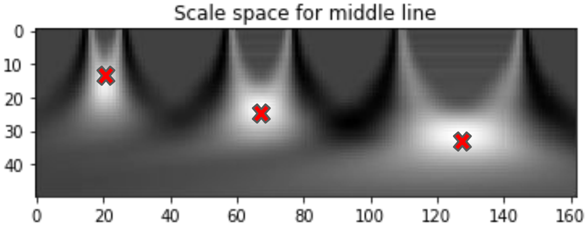
LoG σ=10

Detecting corners / blobs

Build a scale space representation: *stack of images (3D) with increasing sigma*







Then find local extremas in the scale space volume.

Difference of Gaussian (DoG)

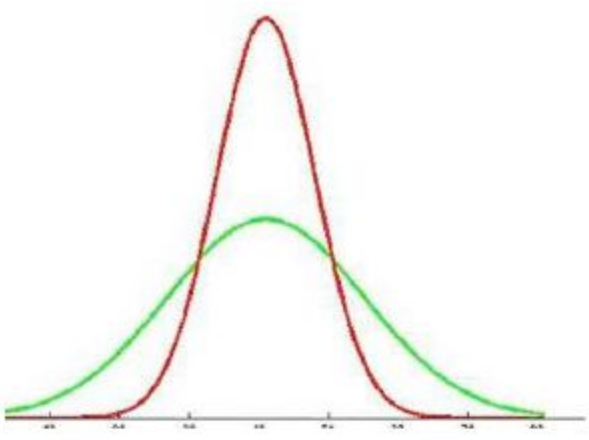
Fast approximation of LoG. Used by SIFT.

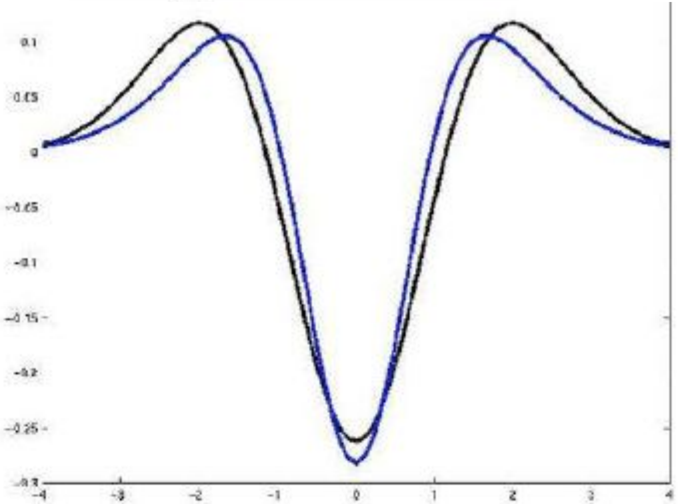
LoG can be approximate by a Difference of two Gaussians (DoG) at different scales.

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

Best approximation when:

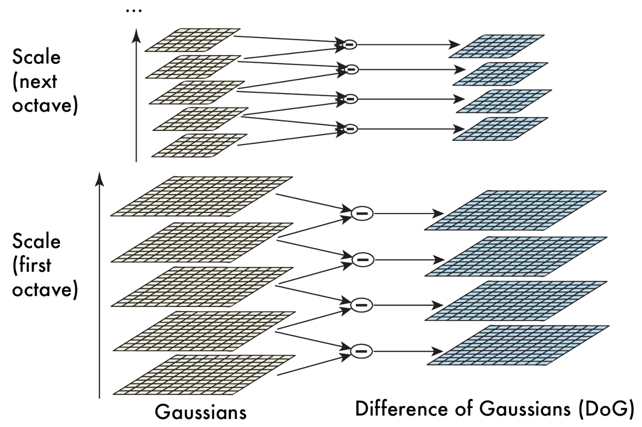
$\sigma_1 = \frac{\sigma}{\sqrt{2}},\ \sigma_2 = \sqrt{2}\sigma$





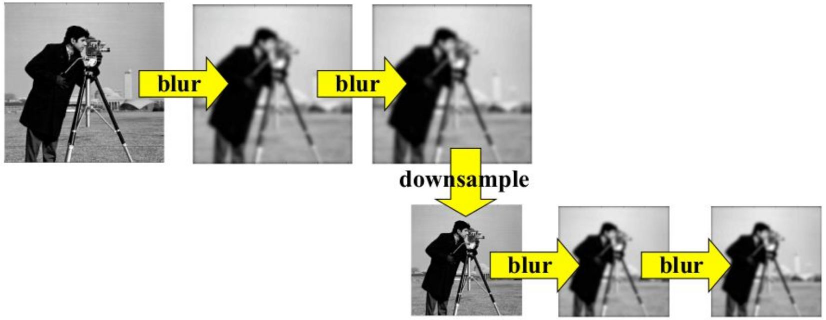
DoG scale generation trick

Illustration: D. Lowe



DoG computation

- "Octave" because frequency doubles between octaves
- If sigma = sqrt(2), then 3 levels per octave
- Downsample images for next octave (like double sized kernel)
- Compute the DoG between images



Crowley et.al., "Fast Computation of Characteristic Scale using a Half-Octave Pyramid." Proc International Workshop on Cognitive Vision (CogVis), Zurich, Switzerland, 2002.

DoG: Corner selection

Throw out weak responses and edges

Estimate gradients

- Similar to Harris, look at nearby responses
- Not whole image, only a few points! Faster!
- Throw out weak responses

Find cornery things

- Same deal, structure matrix, use det and trace information (SIFT variant)

→

D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004., see p. 12

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r} \qquad \mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Determinant of Hessian (DoH)

Faster approximation. Used by SURF.
Better resistance to perspective

Computes the scale-normalized determinant of the Hessian (strength of the curvature at a given point)

$$\boxed{\det H_{norm} L = \sigma^2 (L_{xx} L_{yy} - L_{xy}^2)}$$

⇒ Precompute ***Lxx***, ***Lyy***, ***Lxy***
⇒ Blur them with the right sigma while computing **det *H L***: 3 additions
⇒ normalize: different scales – same value range

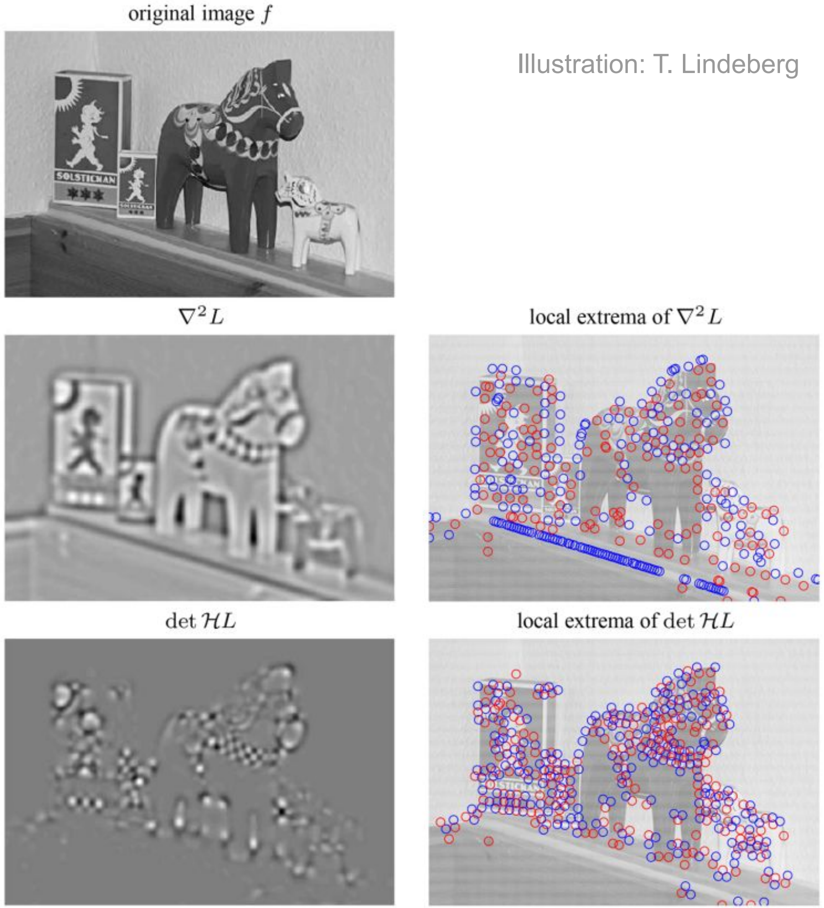
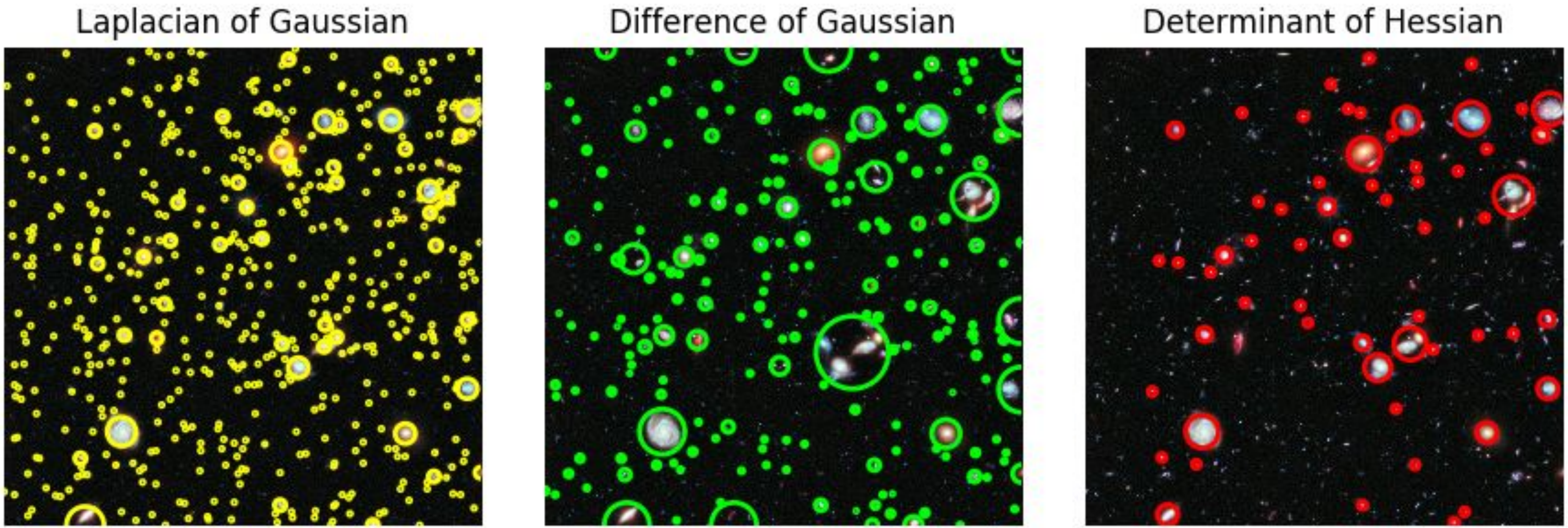


Illustration: T. Lindeberg

LoG vs DoG vs DoH



https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_blob.html

LoG, DoG, DoH summary

Pros

Very robust to transformations
- Perspective
- Blur

Adjustable size detector

Cons

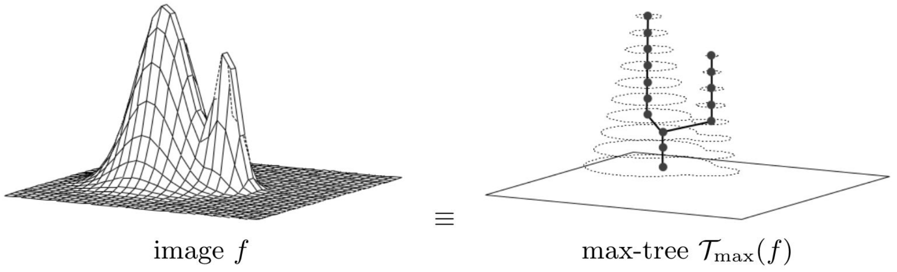
Slow

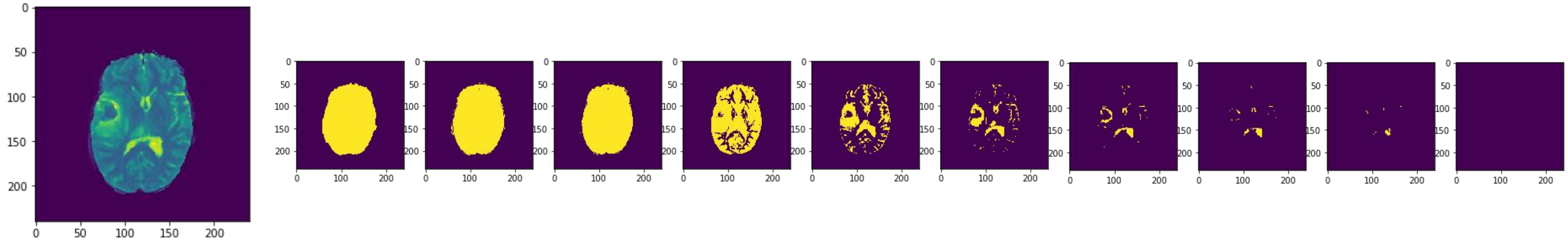
Blob detectors
MSER

Maximally Stable Extremal Regions (MSER)

Detects regions which are stable over thresholds.

1. **Create min- & max-tree of the image**
   tree of included components
   when thresholding the image
   at each possible level

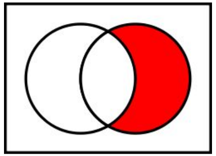




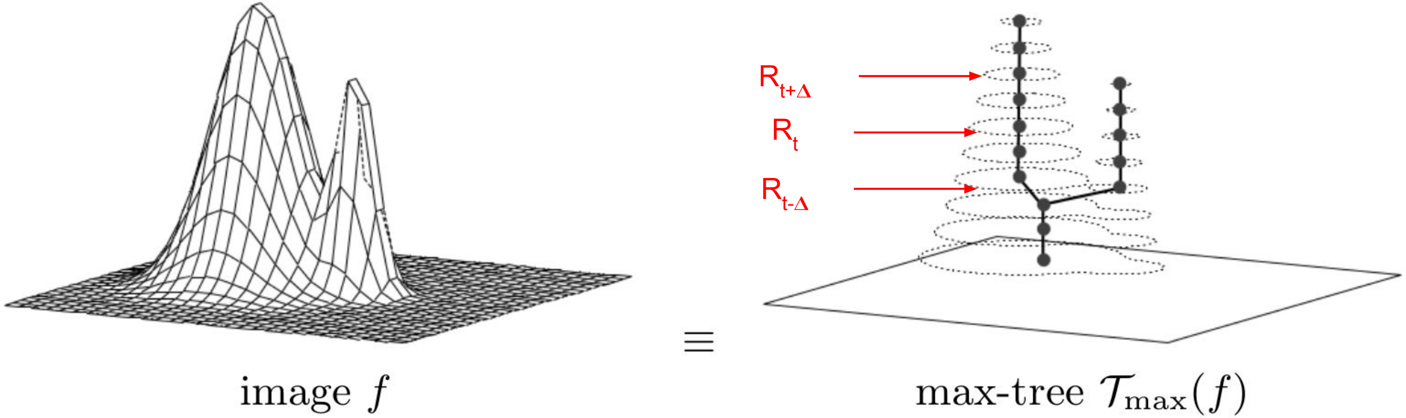
J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” Image and vision computing, vol. 22, no. 10, pp. 761–767, 2004.

Maximally Stable Extremal Regions (MSER)

2. **Select most stable regions** between t-Δ and t+Δ
   R_{t^*} is maximally stable iif $\mathbf{q(t) = |\, R_{t-\Delta} \setminus R_{t+\Delta} \,| \;/\; |\, R_t \,|}$
   as local minimum at t*



$|\, R \,|$ = card(R); Δ = parameter; $R_{t-\Delta} \setminus R_{t+\Delta}$ = set difference



MSER summary

Pros

Very robust to transformations

- Affine transformations
- Intensity changes

Quite fast

Cons

Does support blur

Local feature detectors
Conclusion

Local feature detectors: Conclusion

Harris Stephens: Can be very stable when combined with DoG

Shi-Tomasi: Assumes affine transformation (avoid it with perspective)

DoG: slow but very robust (perspective, blur, illumination)

DoH: faster than DoG, misses small elements, better with perspective.

FAST: very fast, robust to perspective change (like DoG), but blur quickly kills it

MSER: fast, very stable, good choice when no blur

Classification

Feature detector	Edge	Corner	Blob
Canny	X		
Sobel	X		
Harris & Stephens / Plessey / Shi–Tomasi	X	X	
Shi & Tomasi		X	
FAST		X	
Laplacian of Gaussian		X	X
Difference of Gaussians		X	X
Determinant of Hessian		X	X
MSER			X

https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)