The National Higher School of
Artificial Intelligence
المدرسة الوطنية العليا للذكاء الاصطناعي

Course
« Computer Vision»

Sid-Ahmed Berrani

2024-2025

IV. Edge Detection: The Canny Detector

- Probably the most widely used edge detector in computer vision.
- It uses the best properties of the gradient operator and the Laplacian.
- In 1986, Canny tried to find the **optimal** edge detector, assuming a perfect step edge in Gaussian noise. Optimal means:

1. **Low error rate:** All edges should be found, and there should be no spurious responses.
2. **Edge points should be well localized:** The edges located must be as close as possible to the true edges
3. **Single edge point response:** The detector should not identify multiple edge pixels where only a single edge point exists.

IV. Edge Detection: The Canny Detector

- The essence of Canny's work was in expressing the preceding three criteria mathematically and then attempting to find optimal solutions to these formulations.
- Given a filter *F*, two objective functions have been defined:

  1. $\Lambda(F)$: Large if *F* produces good **localization.**
  2. $\sum(F)$: Large if *F* produces good **detection.**

Problem:

Find the best filter *F* that maximizes the compromise criterion: $\Lambda(F)\sum(F)$.

With the additional constraint that a single peak should be generated at a step-edge.

IV. Edge Detection: The Canny Detector

- The algorithm:
    - Smooth the image with a 2D Gaussian filter.
    - Compute Image gradient using Sobel operator (for example).
    - Find gradient magnitude at each pixel.
    - Find gradient orientation at each pixel.
    - Compute 1D Laplacian along the gradient direction at each pixel.
    - Find zero-crossings in Laplacian to find the edge location.

IV. Edge Detection: The Canny Detector

- The algorithm:
  - Smooth the image with a 2D Gaussian filter.
  - Compute Image gradient using Sobel operator (for example).
  - Find gradient magnitude at each pixel.
  - Find gradient orientation at each pixel.
  - Apply a non-maxima suppression approach (the idea is to select the single maximum point across the width of an edge).
  - Threshold the non-maxima suppressed gradient image to reduce false edge points – Hysteresis thresholding is used here.

IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

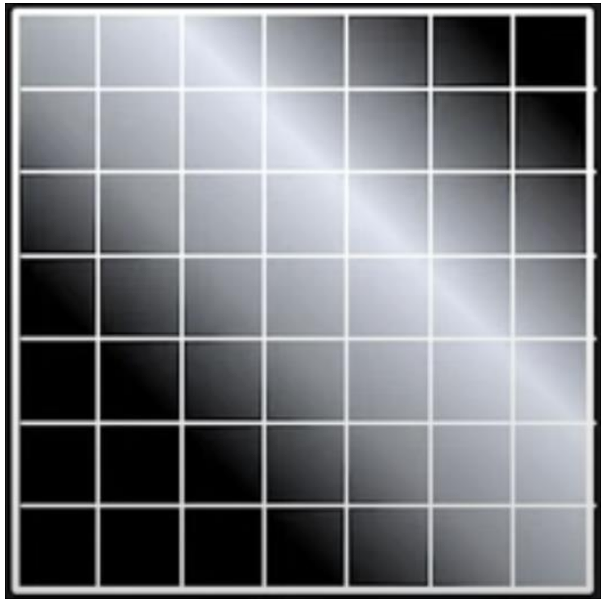
$$\boldsymbol{n_\sigma * I}$$

- Compute Image gradient using Sobel operator (for example):

$$\boldsymbol{\nabla n_\sigma * I}$$

- Find gradient magnitude at each pixel:

$$\boldsymbol{\|\nabla n_\sigma * I\|}$$



IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

$$\boldsymbol{n_\sigma} * \boldsymbol{I}$$

- Compute Image gradient using Sobel operator (for example):

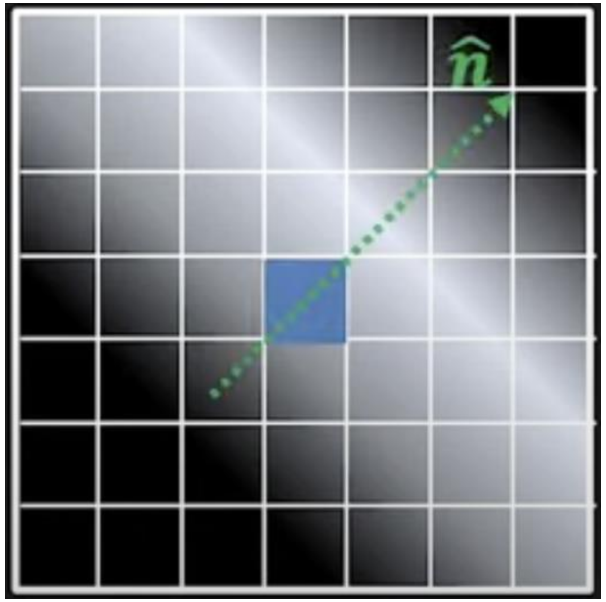
$$\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}$$

- Find gradient magnitude at each pixel:

$$\|\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}\|$$

- Find gradient direction at each pixel:

$$\hat{\boldsymbol{n}} = \frac{\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}}{\|\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}\|}$$



IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

$$\boldsymbol{n_\sigma * I}$$

- Compute Image gradient using Sobel operator (for example):

$$\boldsymbol{\nabla n_\sigma * I}$$

- Find gradient magnitude at each pixel:

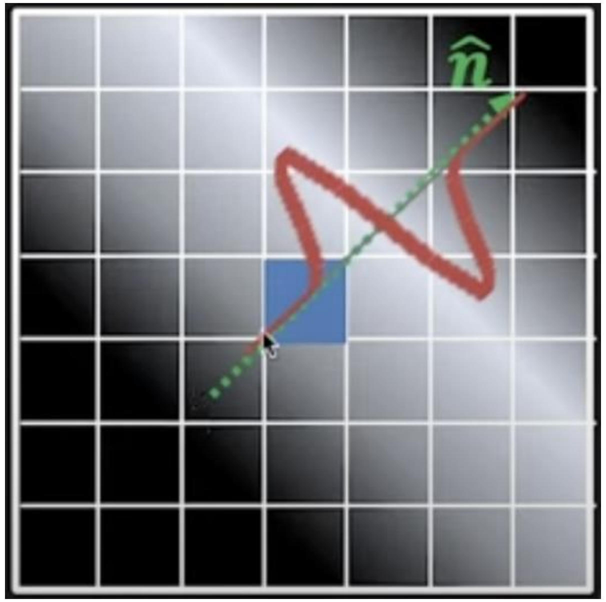
$$\boldsymbol{\|\nabla n_\sigma * I\|}$$

- Find gradient direction at each pixel:

$$\boldsymbol{\hat{n} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}}$$

- Compute 1D Laplacian along the gradient direction $\boldsymbol{\hat{n}}$ at each pixel:

$$\boldsymbol{\frac{\partial^2 (n_\sigma * I)}{\partial \hat{n}^2}}$$



IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

$$\boldsymbol{n}_{\sigma} * \boldsymbol{I}$$

- Compute Image gradient using Sobel operator (for example):

$$\nabla \boldsymbol{n}_{\sigma} * \boldsymbol{I}$$

- Find gradient magnitude at each pixel:

$$\|\nabla \boldsymbol{n}_{\sigma} * \boldsymbol{I}\|$$

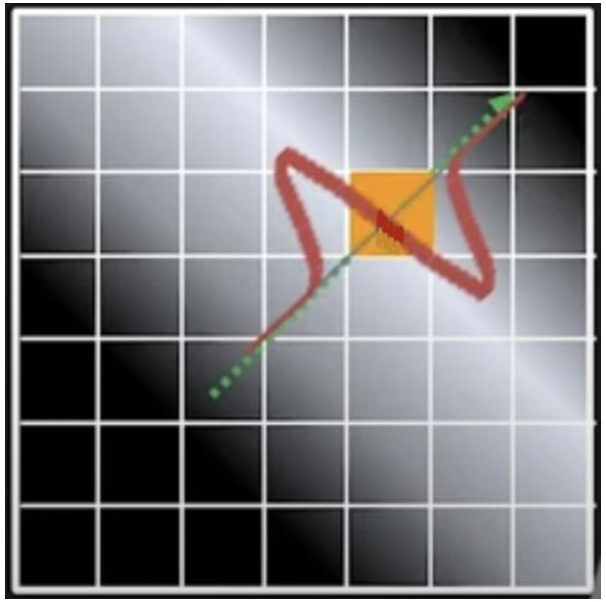
- Find gradient direction at each pixel:

$$\widehat{\boldsymbol{n}} = \frac{\nabla \boldsymbol{n}_{\sigma} * \boldsymbol{I}}{\|\nabla \boldsymbol{n}_{\sigma} * \boldsymbol{I}\|}$$

- Compute 1D Laplacian along the gradient direction $\widehat{\boldsymbol{n}}$ at each pixel:

$$\frac{\partial^2 (\boldsymbol{n}_{\sigma} * \boldsymbol{I})}{\partial \widehat{\boldsymbol{n}}^2}$$

- Find zero-crossings in Laplacian to find the edge location.



IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

$$\boldsymbol{n_\sigma * I}$$

- Compute Image gradient using Sobel operator (for example):

$$\boldsymbol{\nabla n_\sigma * I}$$

- Find gradient magnitude at each pixel:

$$\boldsymbol{\|\nabla n_\sigma * I\|}$$

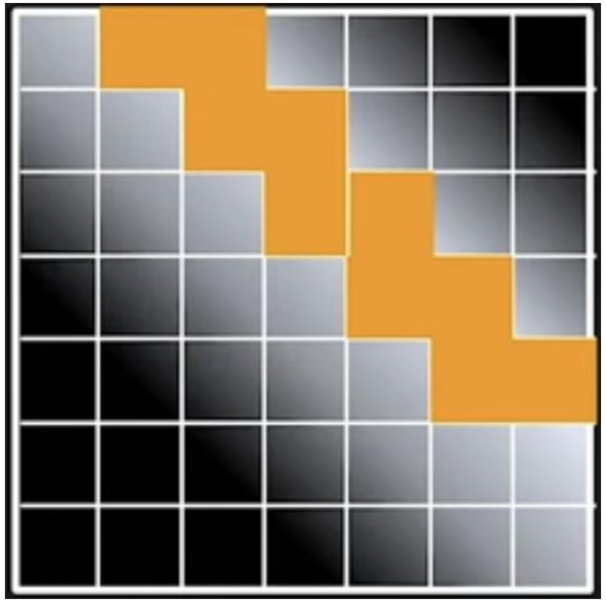
- Find gradient direction at each pixel:

$$\boldsymbol{\hat{n} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}}$$

- Compute 1D Laplacian along the gradient direction $\boldsymbol{\hat{n}}$ at each pixel:

$$\boldsymbol{\frac{\partial^2 (n_\sigma * I)}{\partial \hat{n}^2}}$$

- Find zero-crossings in Laplacian to find the edge location.



IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

$$\boldsymbol{n_\sigma} * \boldsymbol{I}$$

- Compute Image gradient using Sobel operator (for example):

$$\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}$$

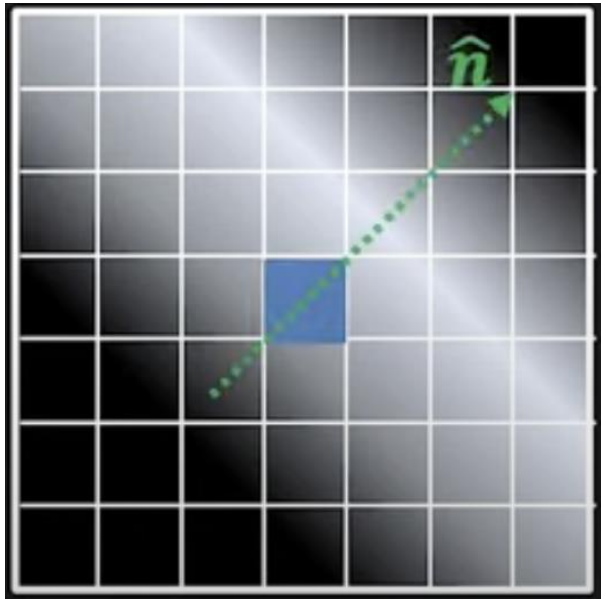
- Find gradient magnitude at each pixel:

$$\|\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}\|$$

- Find gradient direction at each pixel:

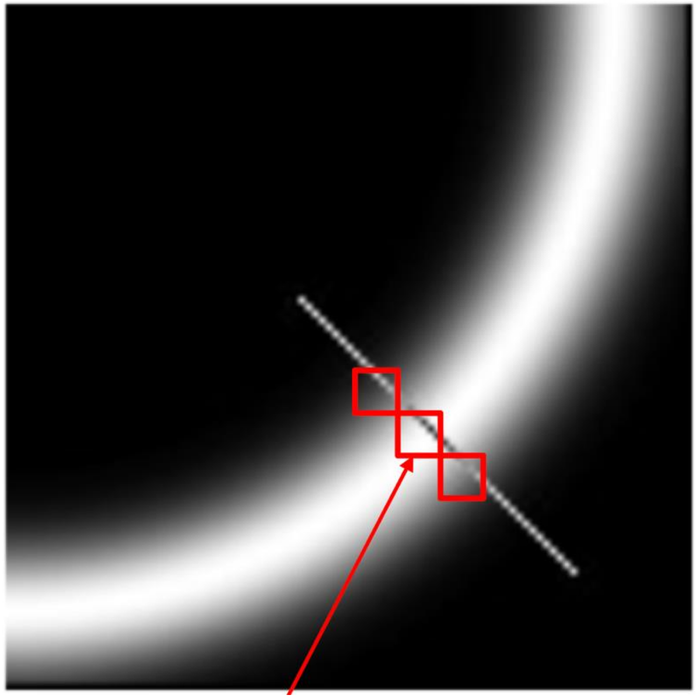
$$\widehat{\boldsymbol{n}} = \frac{\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}}{\|\nabla \boldsymbol{n_\sigma} * \boldsymbol{I}\|}$$

- Apply the non-maxima suppression: For each pixel check the neighbors depending on the direction of the gradient



IV. Edge Detection: The Canny Detector

- The non-maxima suppression



If the intensity of the gradient of a pixel is less than at least one of the two neighbors along the gradient direction, then it must be removed (i.e. gradient magnitude set to zero)

IV. Edge Detection: The Canny Detector

- Smooth the image with a 2D Gaussian filter:

$$\boldsymbol{n_\sigma * I}$$

- Compute Image gradient using Sobel operator (for example):

$$\boldsymbol{\nabla n_\sigma * I}$$

- Find gradient magnitude at each pixel:

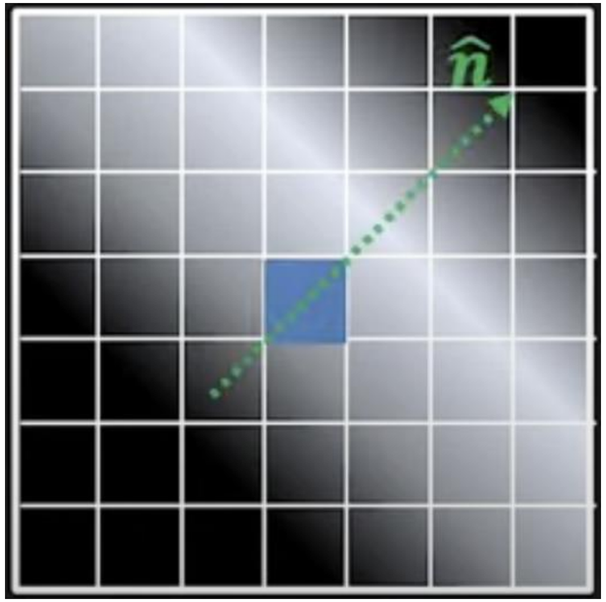
$$\boldsymbol{\|\nabla n_\sigma * I\|}$$

- Find gradient direction at each pixel:

$$\boldsymbol{\hat{n} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}}$$

- Apply the non-maxima suppression: For each pixel check the neighbors depending on the direction of the gradient.

- Apply hysteresis thresholding to reduce false edge points.



IV. Edge Detection: The Canny Detector

Thresholding:

- Hysteresis-based: Using two thresholds ($T_0 < T_1$)

$\|\nabla I(x,y)\| < T_0$	Definitely not an edge
$\|\nabla I(x,y)\| \geq T_1$	Definitely an edge
$T_0 \leq \|\nabla I(x,y)\| < T_1$	Is an edge if a neighboring pixel is definitely an edge

IV. Edge Detection: The Canny Detector

Step 1: Noise reduction with Gaussian blur – smooth the image using a Gaussian filter.

Step 2: Compute gradient magnitude and direction.

Step 3: Keep only the local maxima in the gradient direction – this step thins the edges. Can be presented as **“a non-maxima suppression”** or **“a 1D Laplacian operator”.**

Step 4: Apply hysteresis thresholding to reduce false edge points.

IV. Edge Detection: The Canny Detector

- An example: The Lena image

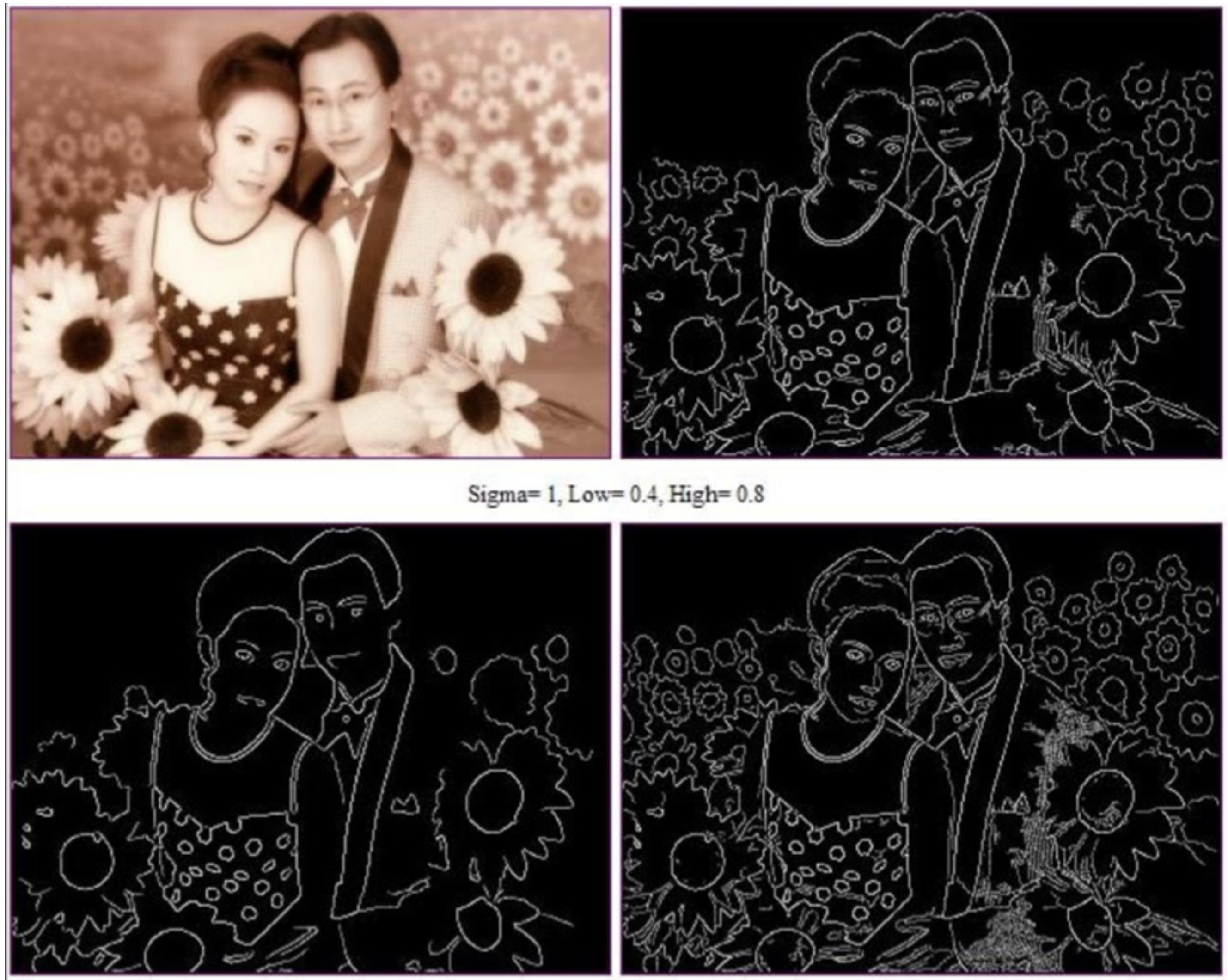
$\sigma = 1$

$\sigma = 2$

$\sigma = 4$

IV. Edge Detection: The Canny Detector

- Another example



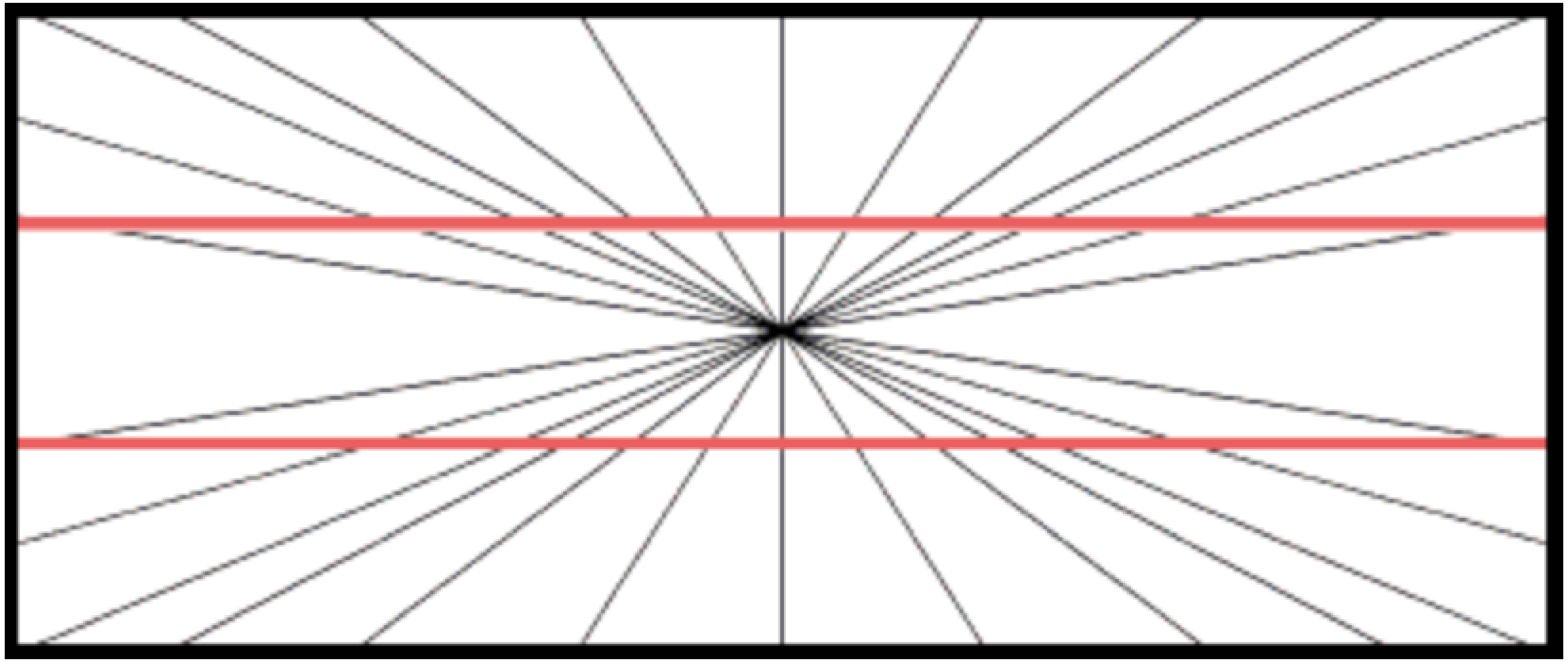
Sigma= 1, Low= 0.4, High= 0.8

Sigma= 2, Low= 0.4, High= 0.8

Sigma= 1, Low= 0.3, High= 0.7

V. Edge Detection: Few Optical Illusions

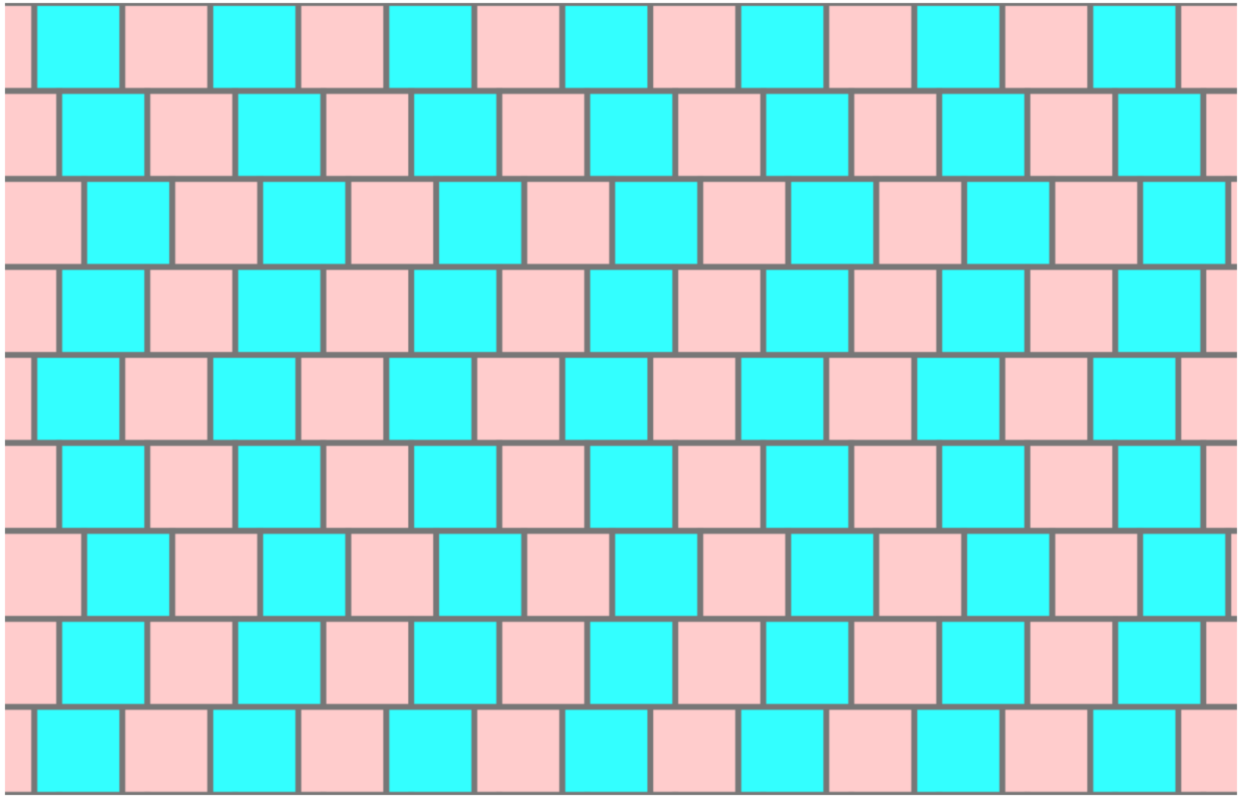
- The standard Hering illusion (1861).



https://michaelbach.de/ot/ang-hering/

V. Edge Detection: Few Optical Illusions

- The Café wall illusion (1979).

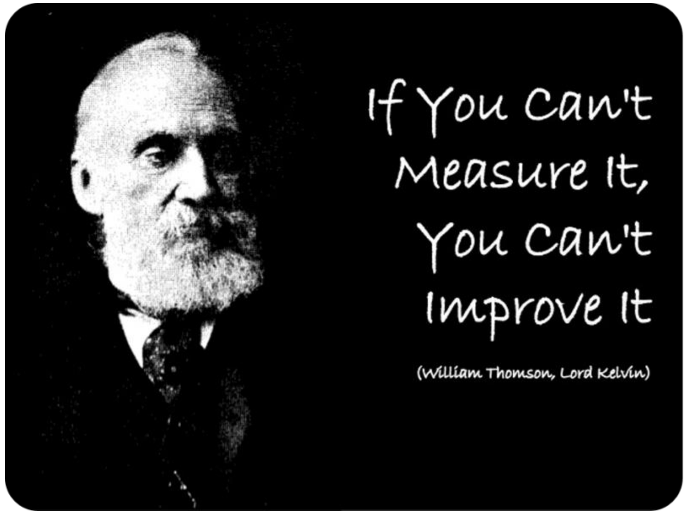


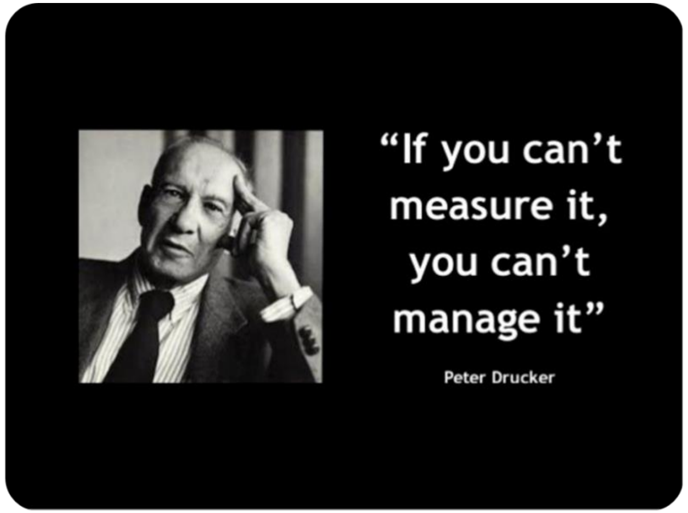
https://michaelbach.de/ot/ang-cafewall/index.html

V. Edge Detection Evaluation

Why Evaluate Edge Detectors?

- Compare performance of different methods.
- Choose the most suitable detector for a target application.
- Ensure reliability in real-world tasks.





V. Edge Detection Evaluation

The 1st approach: Qualitative evaluation

⇒ Based on visual inspection:

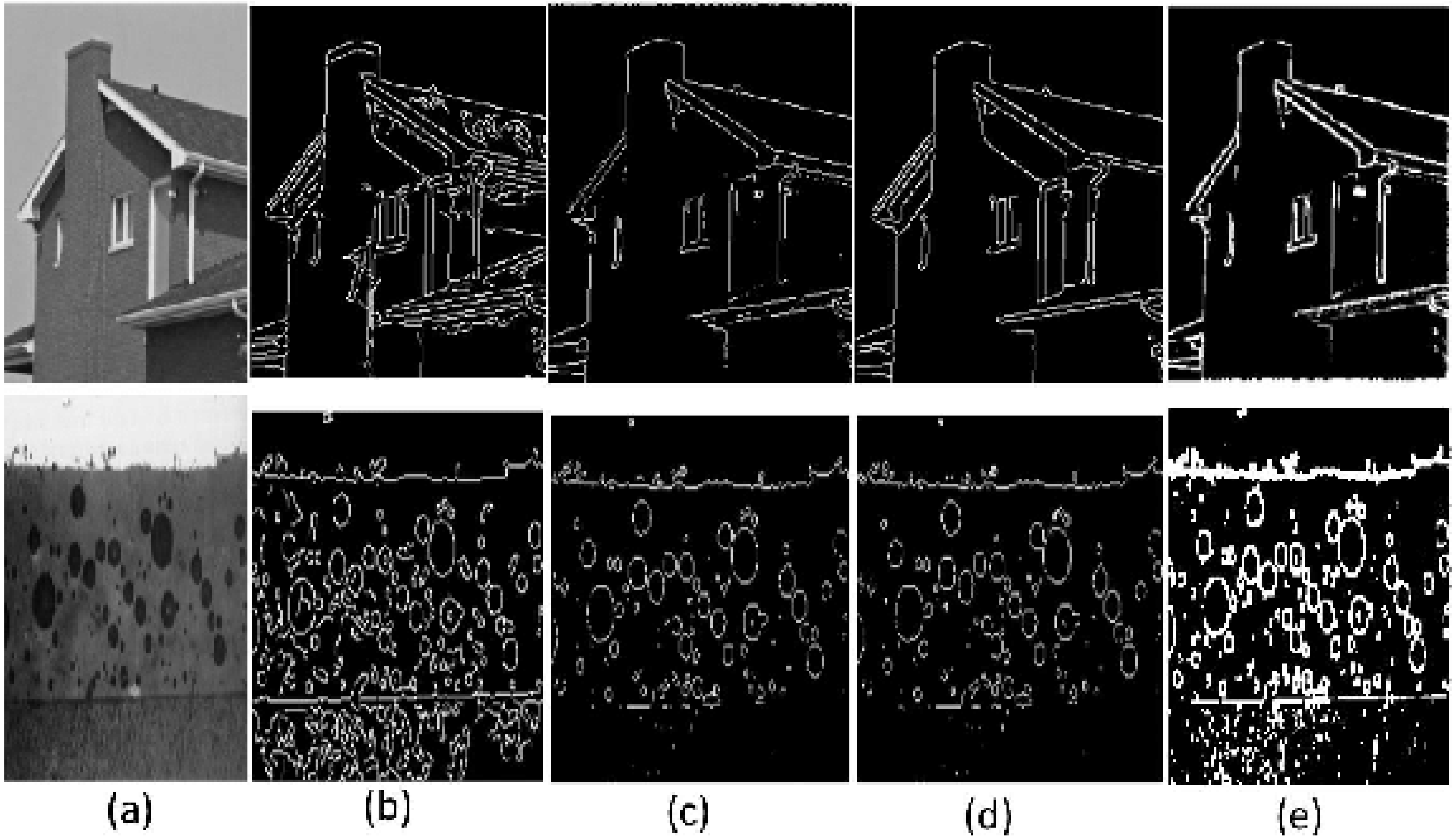
- Are the edges **clean**, **thin**, and **well-localized**?
- Are object **boundaries complete** and continuous?
- Is there too much **noise** (false edges)?
- Do the results match **human perception**?

Common practice: overlay edge map on original image.

⇒ **Quick, intuitive, but subjective.**

V. Edge Detection Evaluation

The 1st approach: Qualitative evaluation



V. Edge Detection Evaluation

The 2nd approach: Quantitative Evaluation

- Assumes a ground-truth edge map is available

V. Edge Detection Evaluation

The 2nd approach: Quantitative Evaluation — Key Metrics

- **Precision, Recall, F1-score**
- **ROC Curve & AUC**
- **Boundary Displacement Error (BDE)**: Measures the average distance between detected edges and ground truth edges. Lower BDE → better alignment with true boundaries.
- **Pratt's Figure of Merit (FOM)**: see next slide.
- **Edge Localization Error:** Measures how precisely edges are detected in terms of position, especially important in high-resolution images or critical applications.

⇒ **Objective, repeatable, and statistically measurable.**

V. Edge Detection Evaluation

The 2nd approach: Quantitative Evaluation — Key Metrics

- Pratt's Figure of Merit (FOM)

$$FOM = \frac{1}{\max(N_d, N_g)} \sum_{i=1}^{N_d} \frac{1}{1 + \alpha d_i^2}$$

N_d: number of detected edge pixels

N_g: number of ground truth edge pixels

d_i: distance from each detected edge pixel to the nearest ground truth

α: a constant (typically 1/9)

=> Values close to 1 indicate good performance.

V. Edge Detection Evaluation

In addition, computational metrics can also be used:

- Runtime performance
- Memory usage
- Scalability to large images

$\Rightarrow$ These can be important in real-time or resource-constrained systems.

V. Edge Detection Evaluation

How is Ground-Truth Created?

- Manual annotation by humans (pixel-accurate)
- Multiple annotators: consensus or averaging
- Synthetic images with known edge structure (ideal for testing)

Available datasets with ground-truth:

- BSDS500: Natural images, human-labeled contours
- NYUDv2: RGB-D indoor scenes with edges
- PASCAL Boundaries: Object-level boundaries
- SBD: Semantic boundaries from PASCAL VOC
- BSDS300: Classic version, still cited

V. Edge Detection Evaluation

Best Practices in Evaluation:

- Align image resolutions (ground-truth and prediction)
- Allow small tolerance in edge location
- Avoid data leakage (do not evaluate on training data)
- Combine quantitative and qualitative evaluation