FREIE UNIVERSITÄT BERLIN

Computer Vision

Lecture 2: Edge Detection

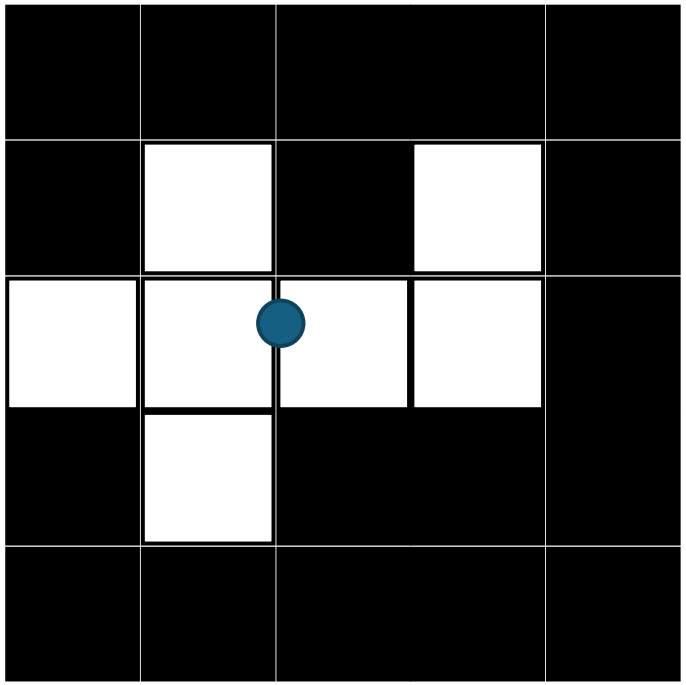
30.10.2024

Manuel Heurich, Tim Landgraf

List of Topics

• Introduction • **Convolution, Edge Detection** • Color Histograms, HoG	Computer Vision Fundamentals
• Optic Flow • Hough Transform • SIFT / SURF	Conventional Computer Vision
• Introduction to Neural Information Processing • Convolutional Neural Networks • Image Classification, Object Detection • Vision Transformers, ConvMixer • Semantic Segmentation • Pose Estimation • Recurrent Neural Networks, Image Captioning	Deep Learning (Supervised)
• Generative Models • Unsupervised feature extraction	Unsupervised Learning

Blob to Center of Mass



$$c = \frac{\begin{pmatrix}18\\20\end{pmatrix}}{7} = \begin{pmatrix}2{,}5\\2{,}9\end{pmatrix}$$

Collect all foreground pixels' positions

$$B = \{(x_1, y_1)^T, (x_2, y_2)^T, \dots, (x_n, y_n)^T\}$$

Compute average

$$c = \frac{\sum_i \begin{pmatrix}x_i\\y_i\end{pmatrix}}{n}$$

Tracking

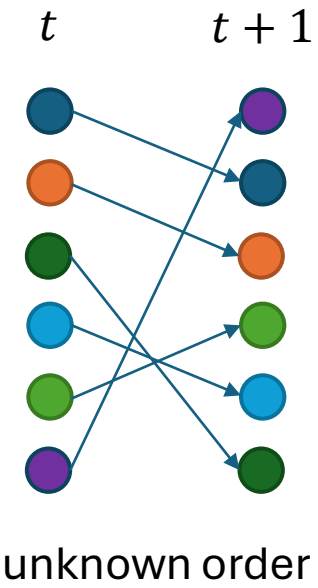
- Given: set of detections

$$D^{(t)} = \{p_i\}; i = 1 \dots N, ; p_i = \begin{pmatrix} x_i^{(t)} \\ y_i^{(t)} \\ \dots \end{pmatrix}$$

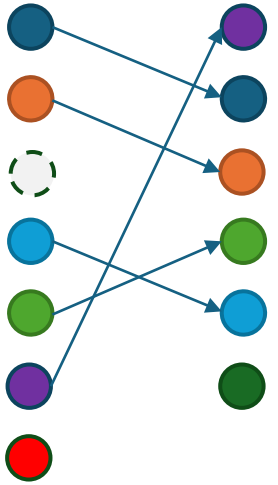
  at time t and $t+1$

- Find: the mapping $M: D_i^{(t)} \rightarrow D_j^{(t+1)}$ that retains the identity of the objects

- Note: in general, objects may (re, dis-)appear any time

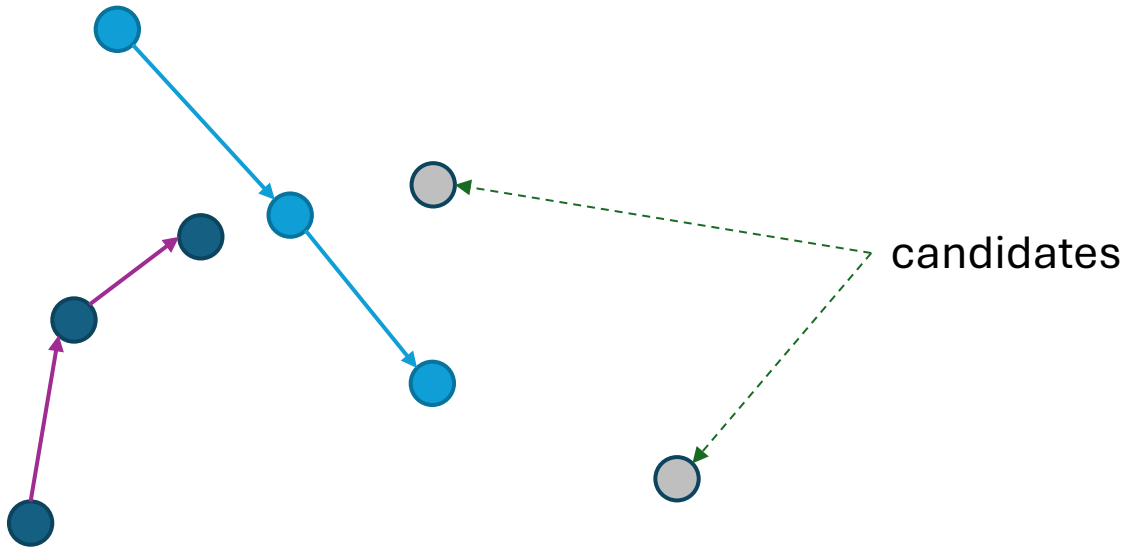


unknown order

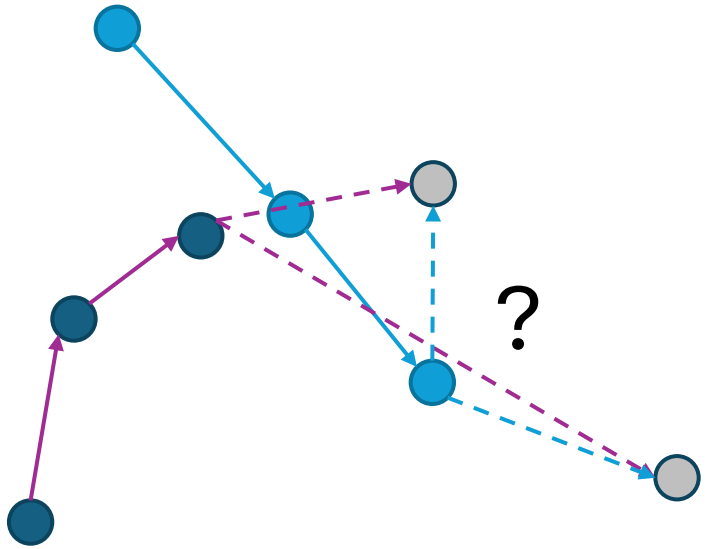


missing and novel detections

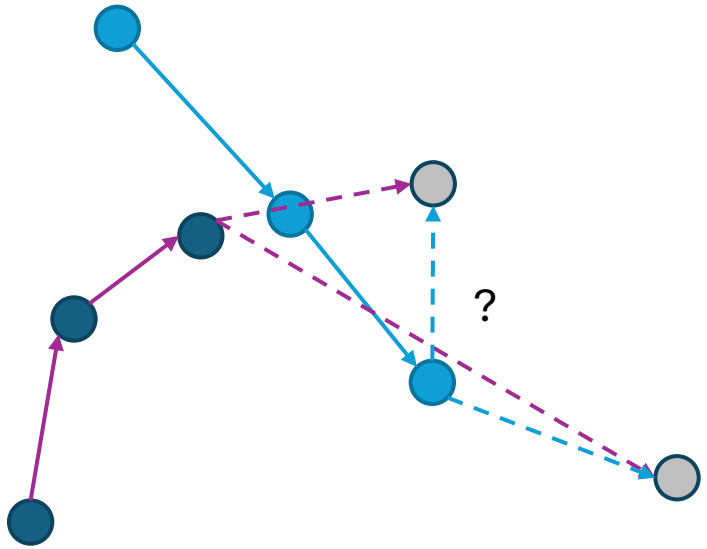
Simplest tracking algorithm



Simplest tracking algorithm



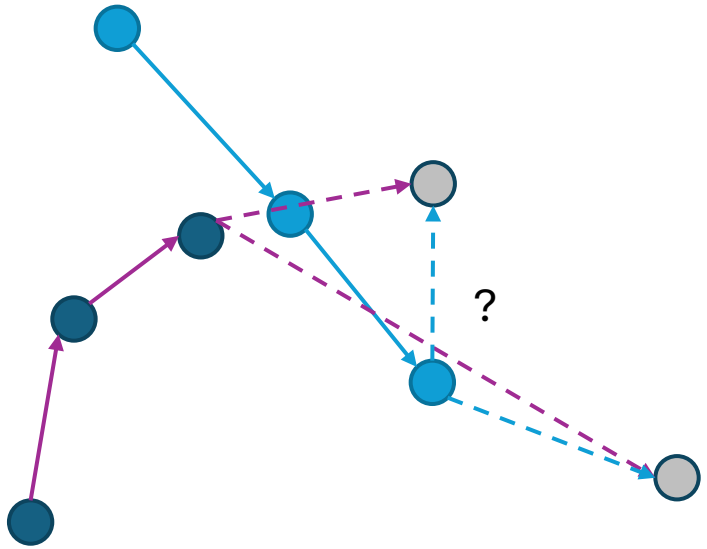
Simplest tracking algorithm



Algorithm:

```
C = get_candidates()
for last detection d in D
        c = get_closest_candidate(d,C)
        M.add(d,c)
        C.remove(c)
```

Simplest tracking algorithm

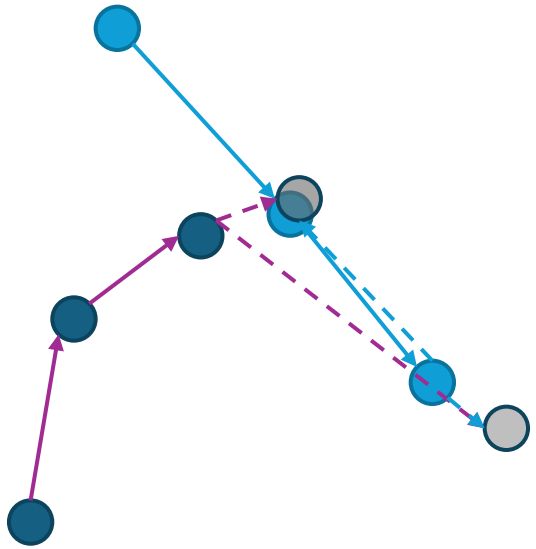


When does this **fail**?

Is the results **invariant to the order** of our last detections?

Will those problems become relevant when tracking a **single object**?

Simplest tracking algorithm



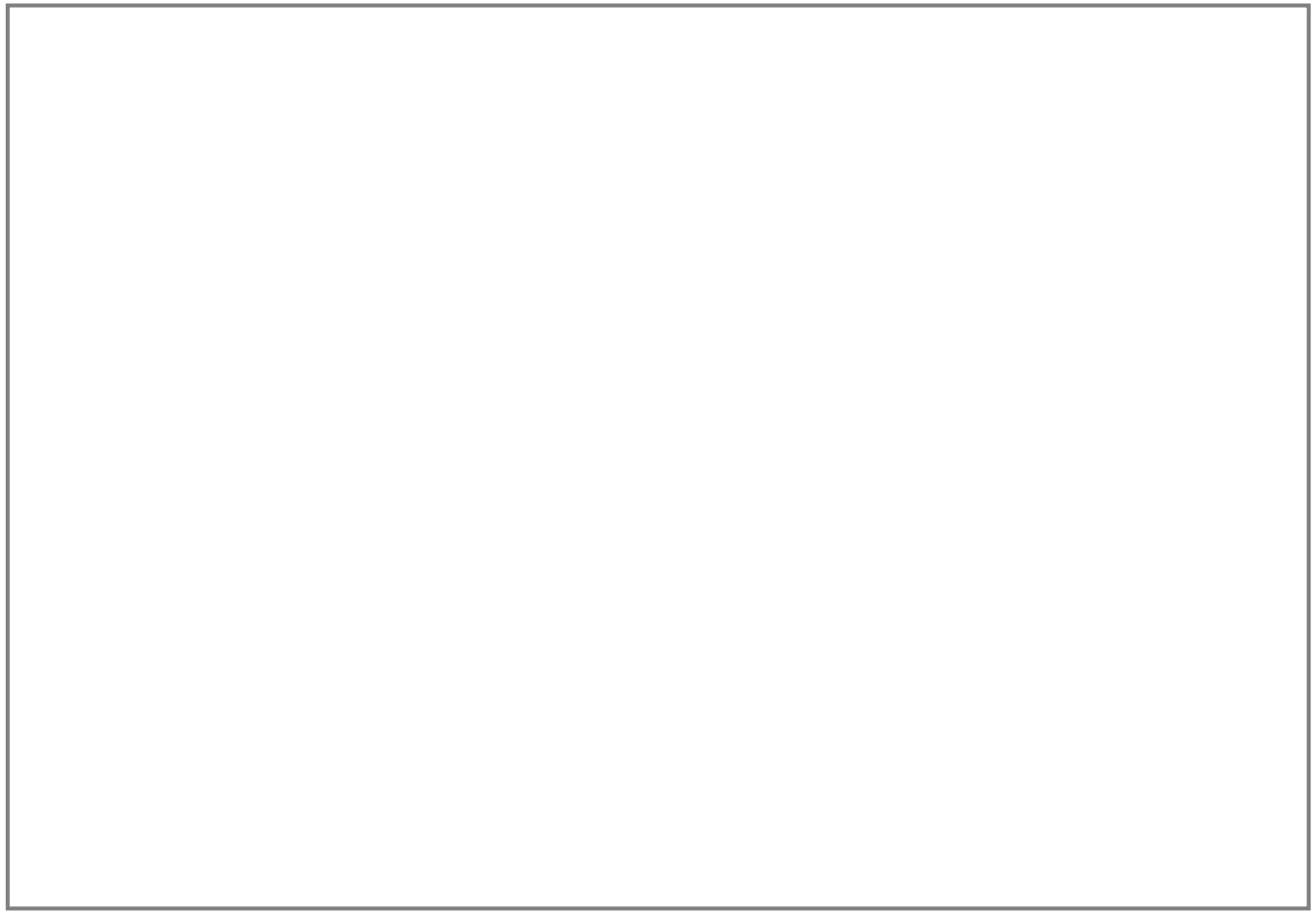
What can we **change in our recording setup** to improve tracking quality?

Ex. 1.3: track colored objects

- use your color detection and connected components algorithm
- implement simplest tracking algorithm
- draw history of all previous points on frame

Source: https://www.youtube.com/watch?v=IOPb5BKT6HY

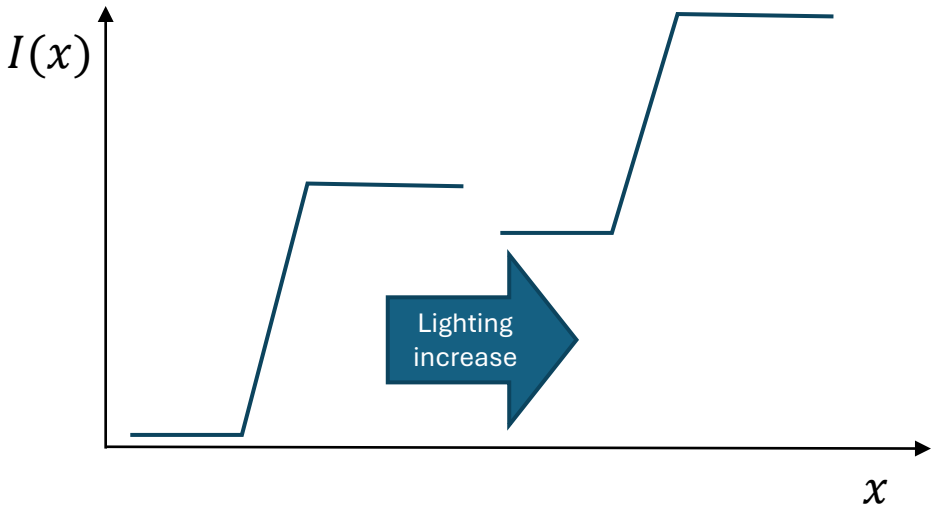
Today: EDGES



White eagle on white background

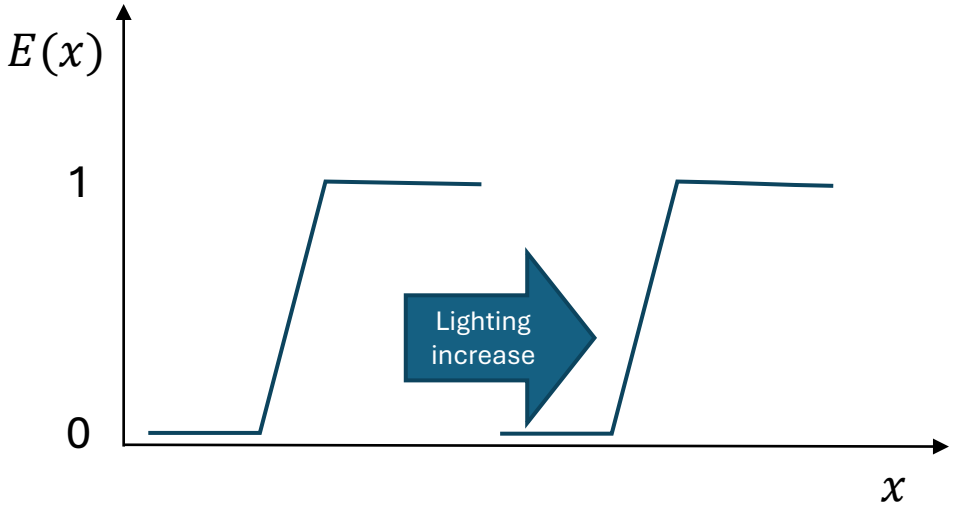
Edges are robust features

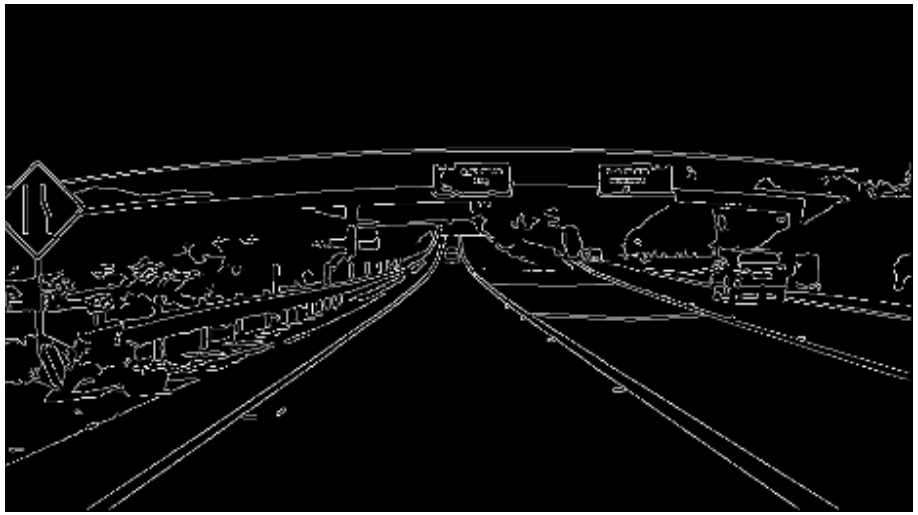
- global lighting changes introduce brightness shifts, but local brightness differences remain stable



Edges are robust features

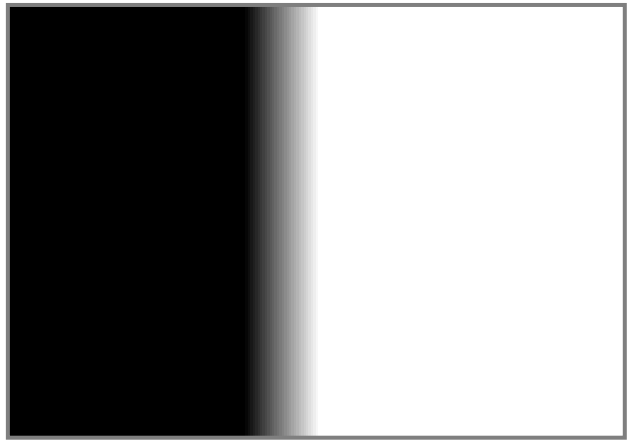
- global lighting changes introduce brightness shifts, but local brightness differences remain stable
- edge image = binary image
- detect local brightness changes!

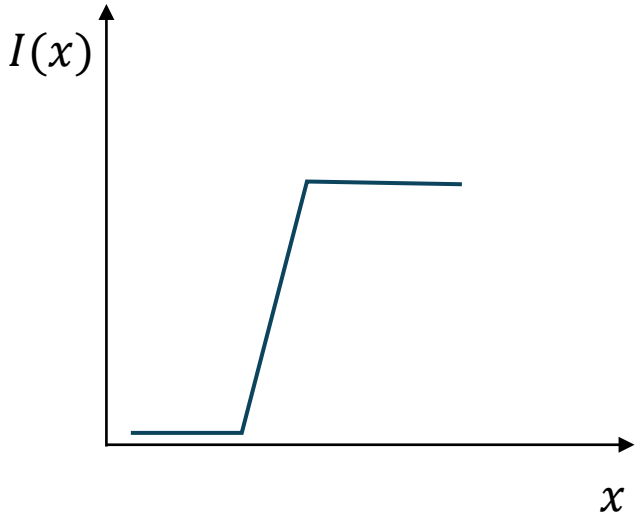




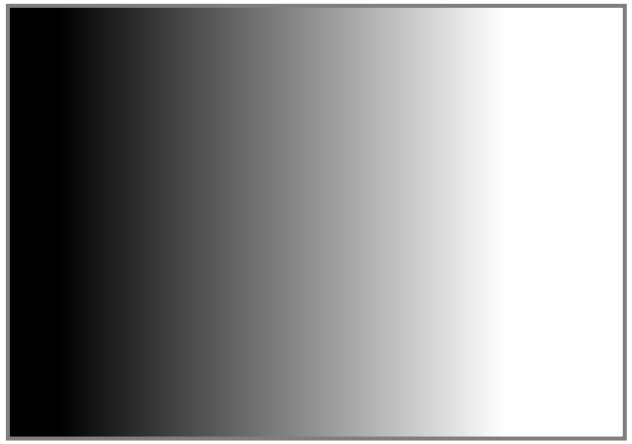
Seibert, A., Hähnel, M., Tewes, A. and Rojas, R., 2013, June. Camera based detection and classification of soft shoulders, curbs and guardrails. In *2013 IEEE Intelligent Vehicles Symposium (IV)* (pp. 853-858). IEEE.

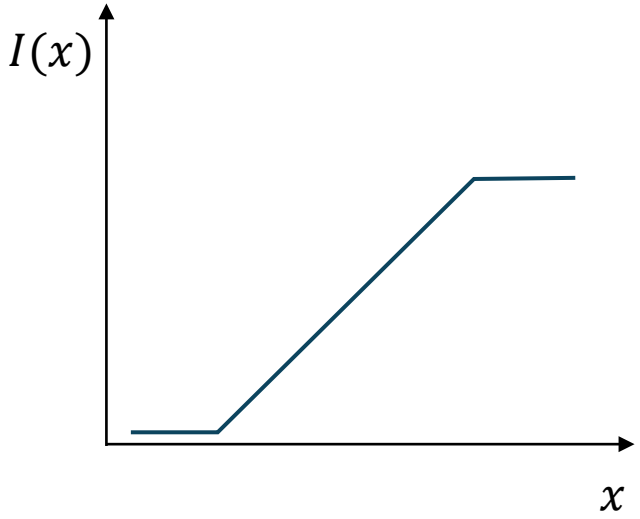
Properties of brightness changes





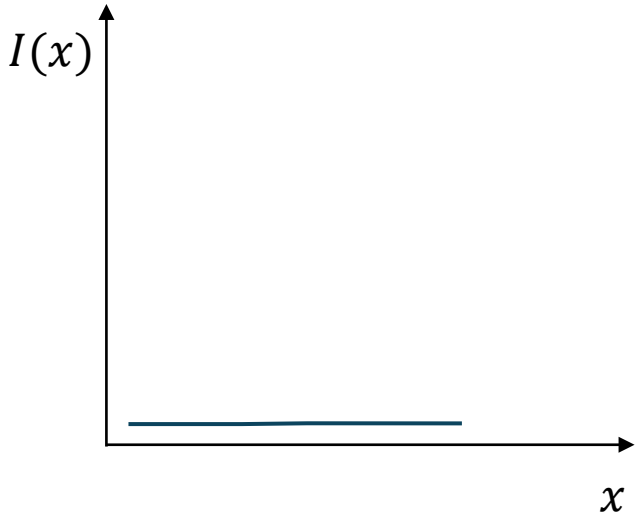
High slope



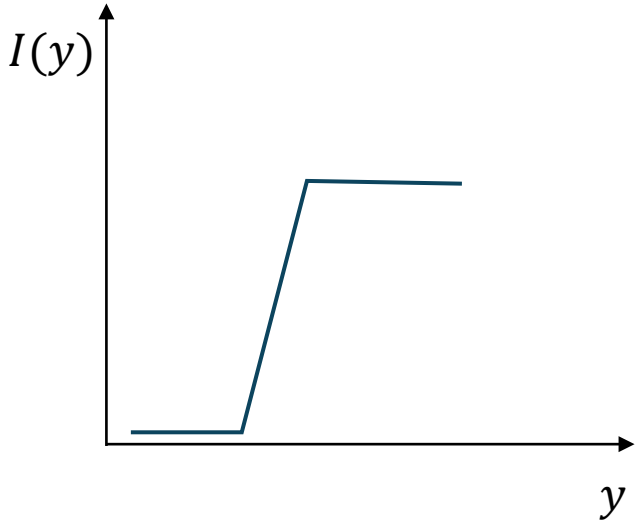


Low slope

Properties of brightness changes



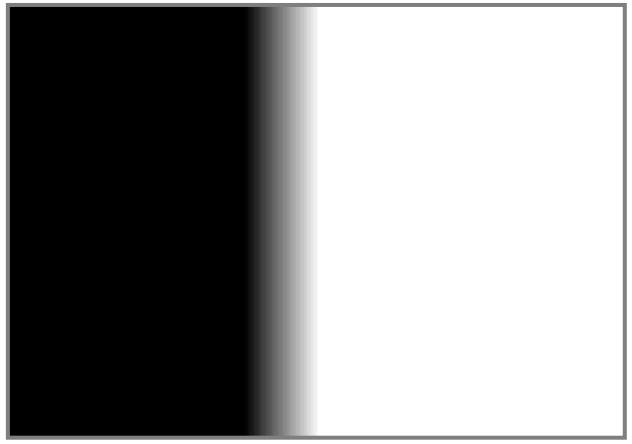
No slope in x direction

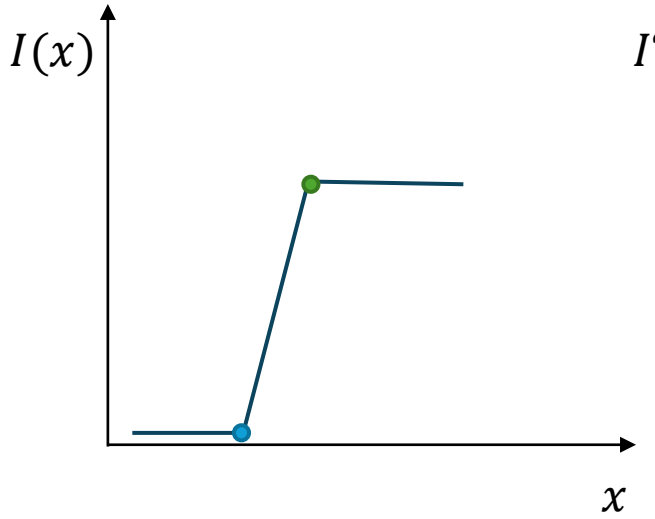


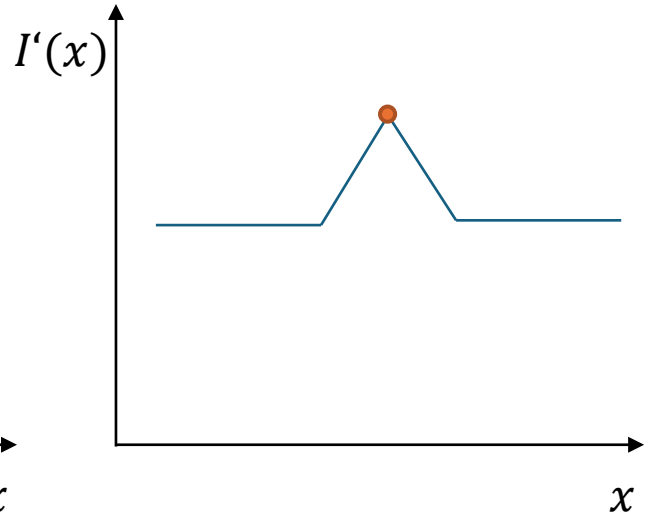
High slope in y direction

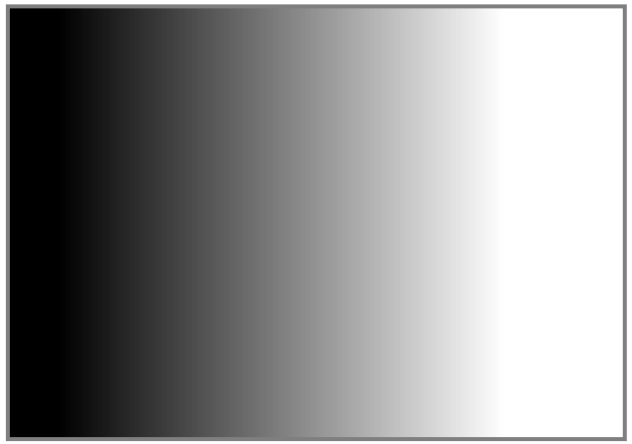
Image Derivative

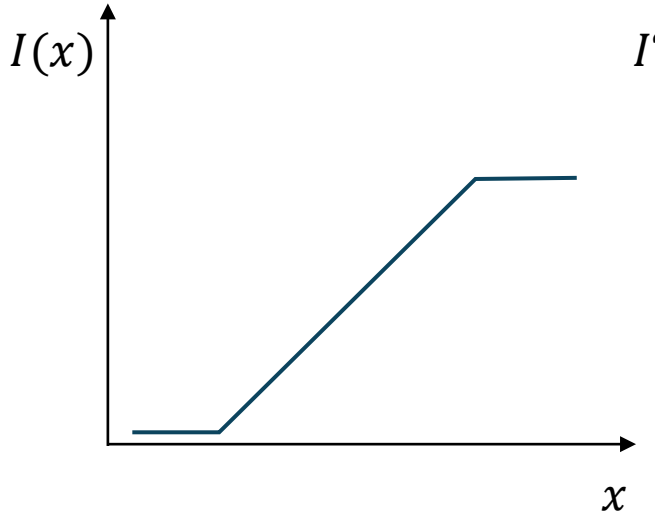
$$I'(x) = \frac{I(x+\Delta) - I(x)}{\Delta}$$











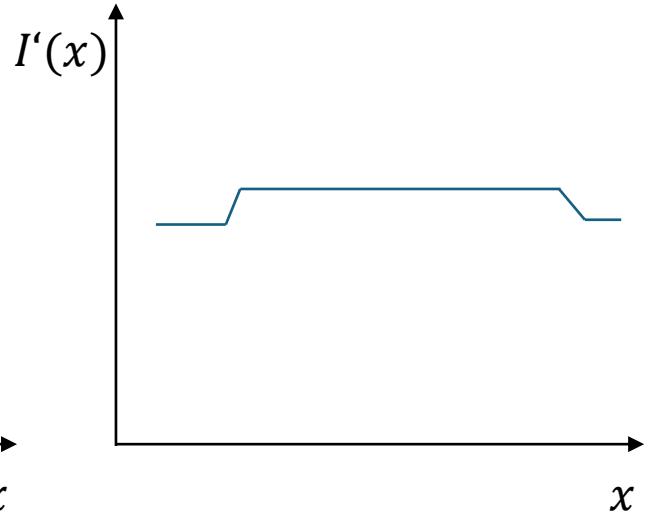
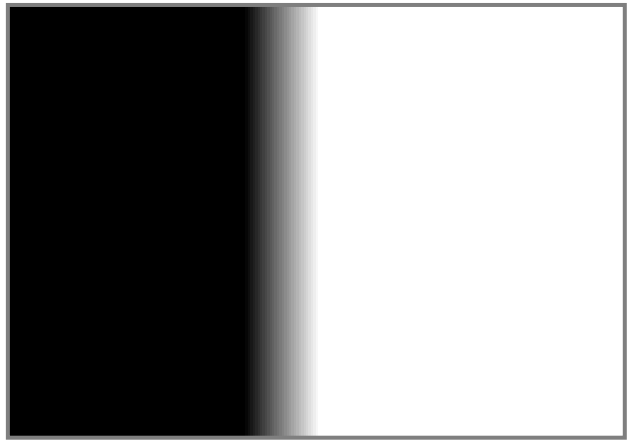
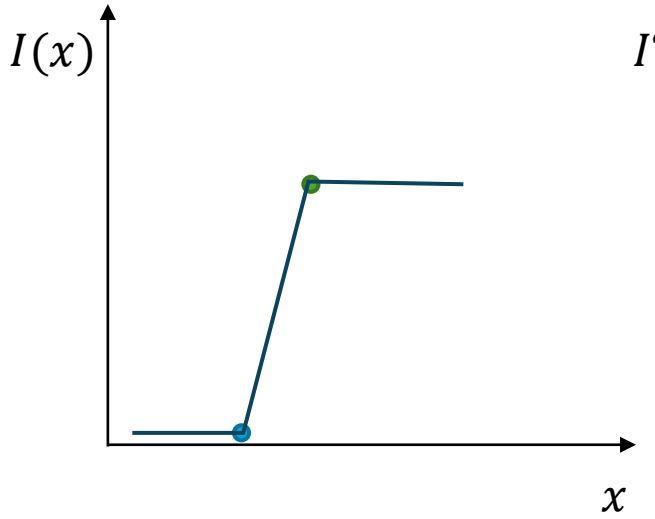
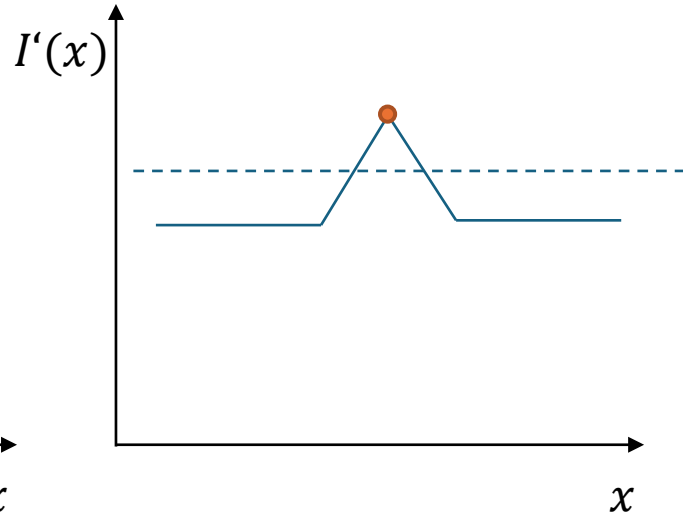


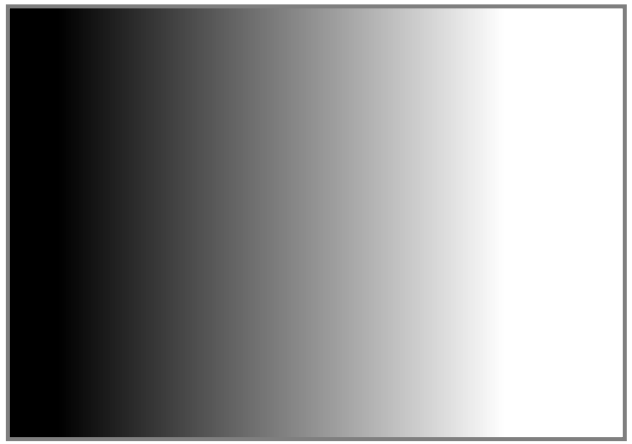
Image Derivative

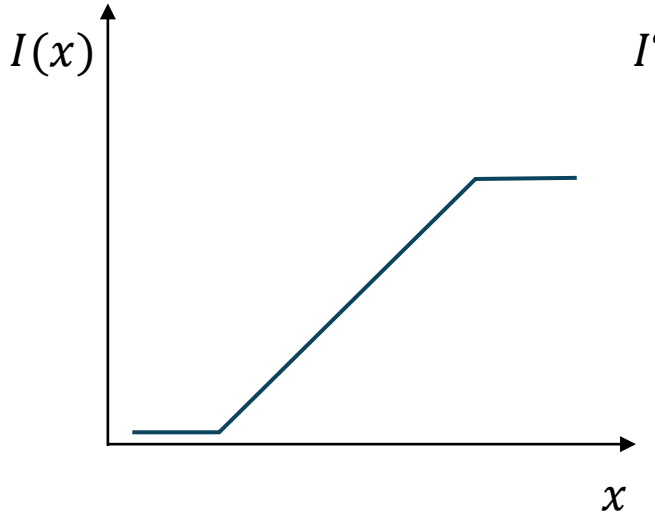
$$I'(x) = \frac{I(x+\Delta) - I(x)}{\Delta}$$

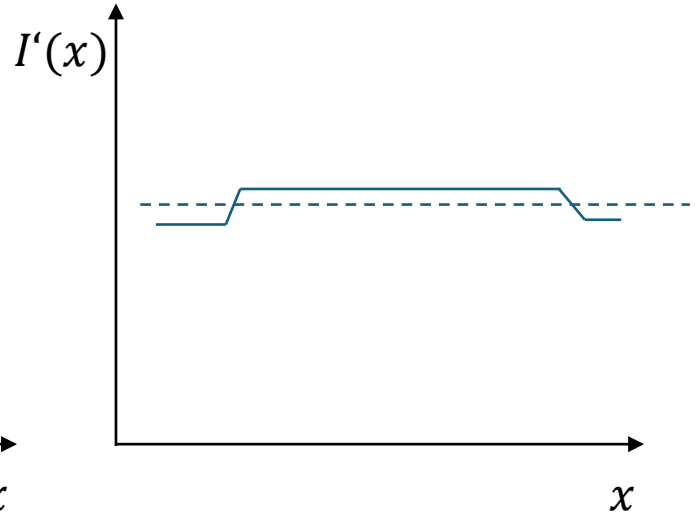








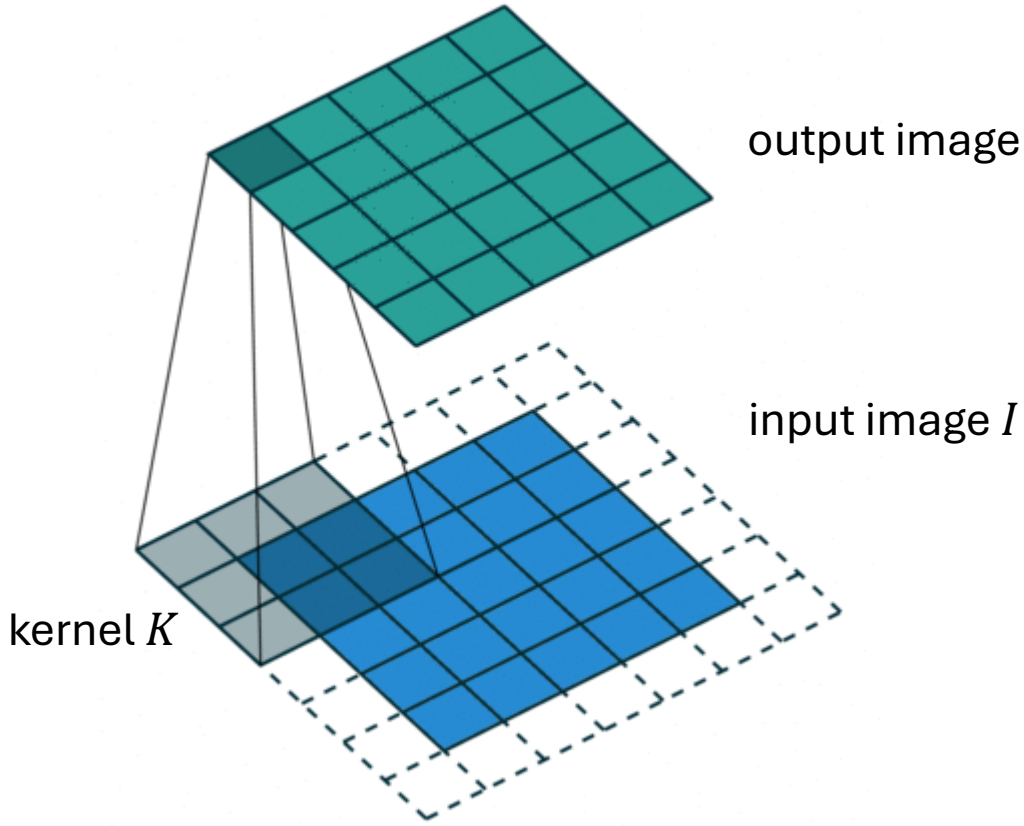




Computer Vision 101: Convolution

$$(I * K)_{x,y} = \sum_{-W}^{W} \sum_{-H}^{H} I_{x-w,y-h} K_{w,h}$$

$$= I^{T}_{N(x,y)} K$$



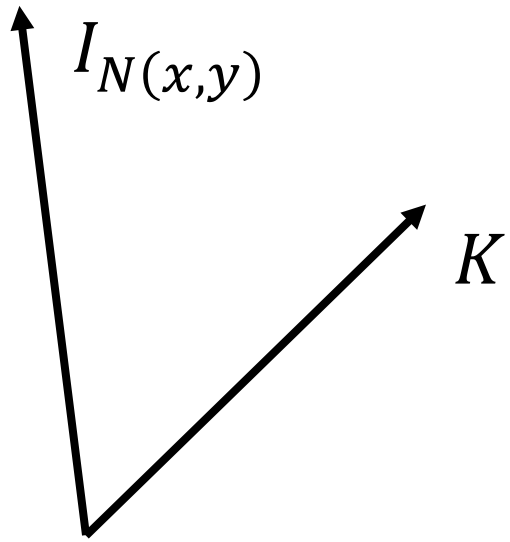
Notion 1: weighted sum

https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

Computer Vision 101: Convolution

$$(I * K)_{x,y} = \sum_{-W}^{W} \sum_{-H}^{H} I_{x-w,y-h} K_{w,h}$$

$$= I_{N(x,y)}^{T} K$$



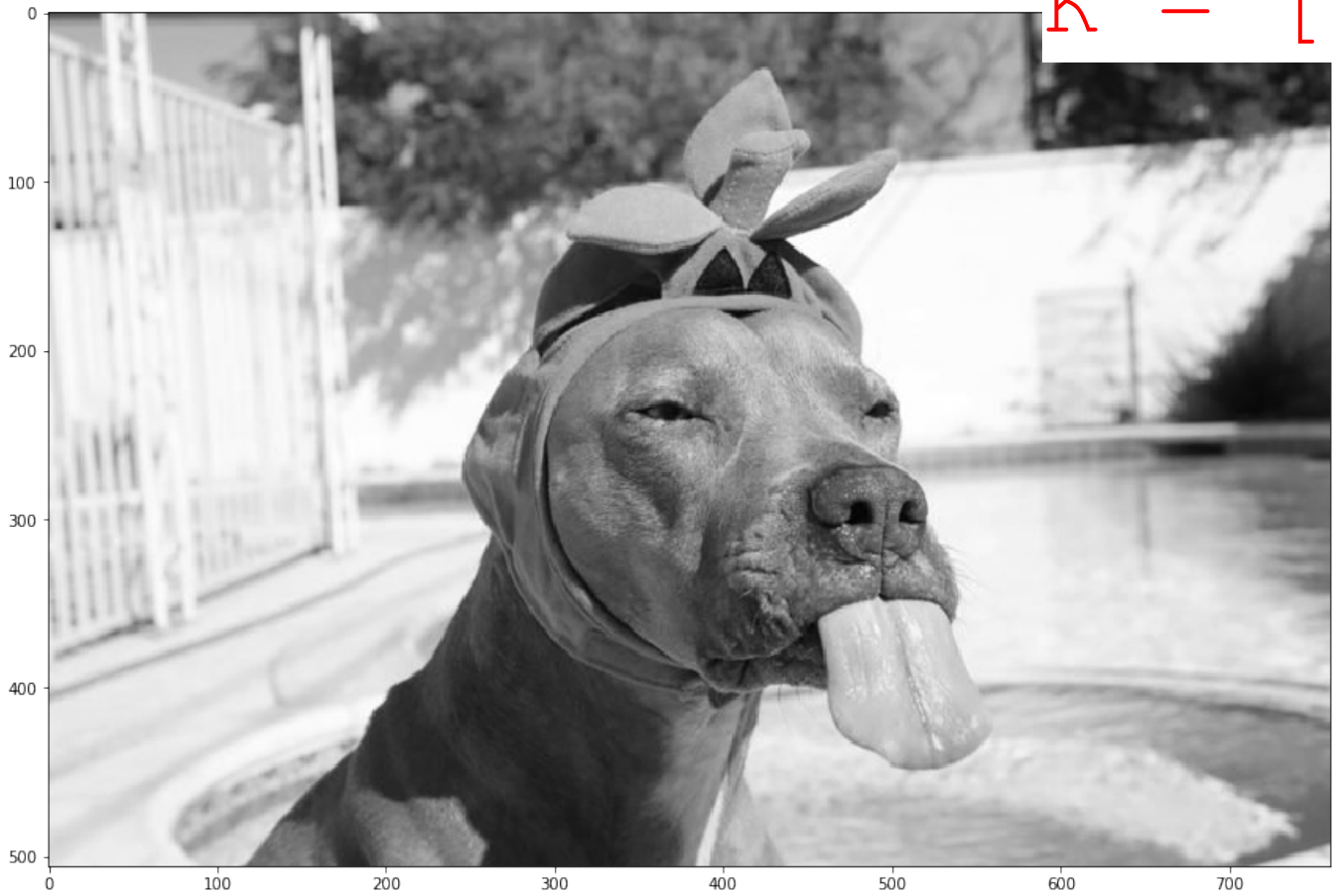
Notion 2: pattern matching via dot product

Convolution

```
K = [[1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1]]
```

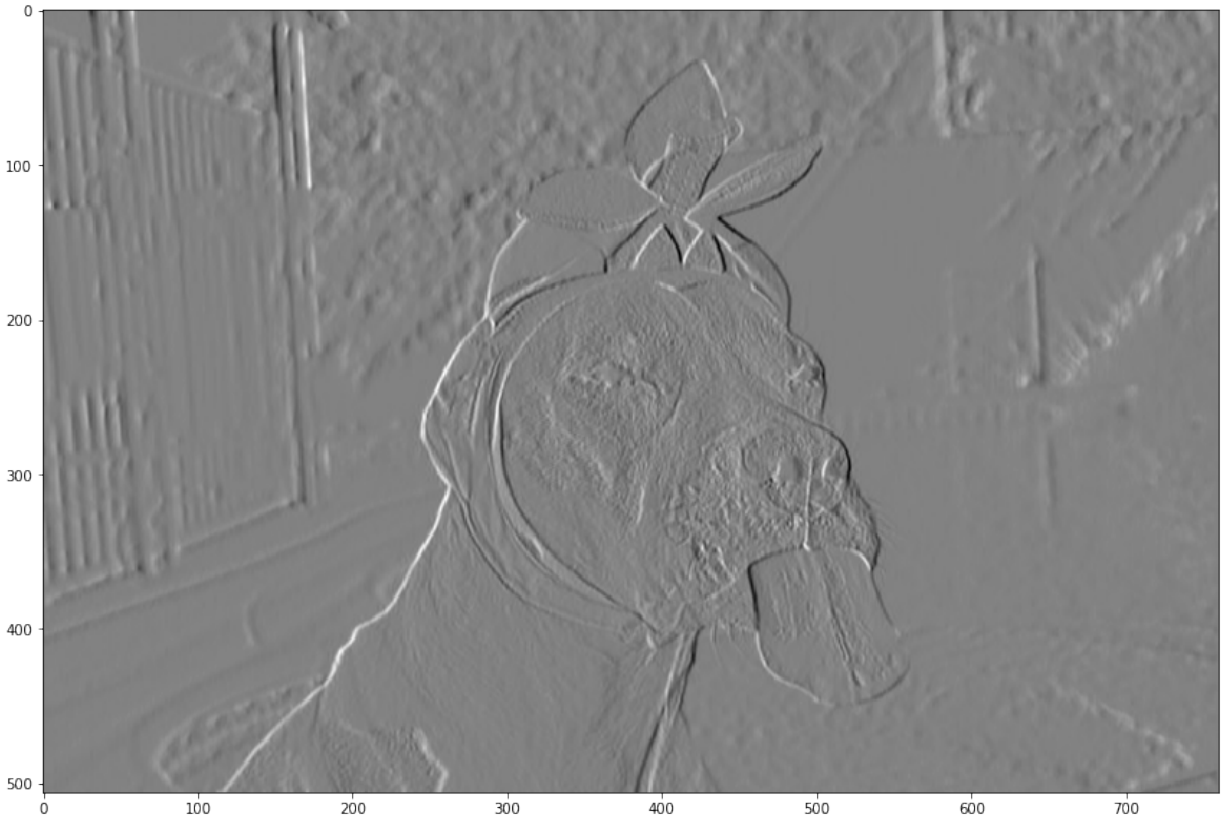
„Detect“ edges with Convolution

```
K = [[???]]
```



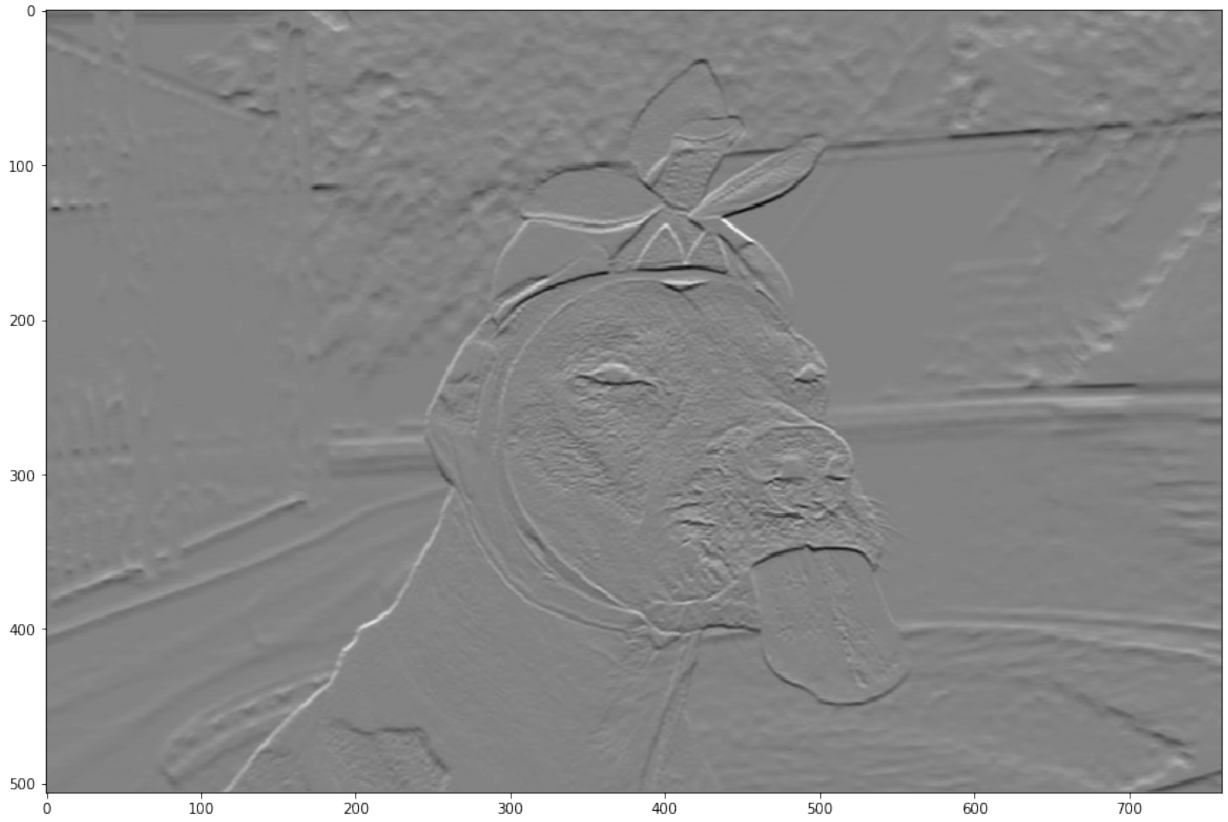
„Detect“ edges with Convolution

```
K = [[-1, 0, 1],
     [-1, 0, 1],
     [-1, 0, 1]]
```



„Detect“ edges with Convolution

$$K = [[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]^T$$



Sobel Kernel, Sobel Filter

```
Sx =[[-1, 0, 1],
     [-2, 0, 2],
     [-1, 0, 1]]
```

```
Sy =[[-1, -2, -1],
     [ 0,  0,  0],
     [ 1,  2,  1]]
```

Convolve for derivative in x-direction

$$G_x = (I * S_x)$$

Convolve for derivative in y-direction

$$G_y = (I * S_y)$$

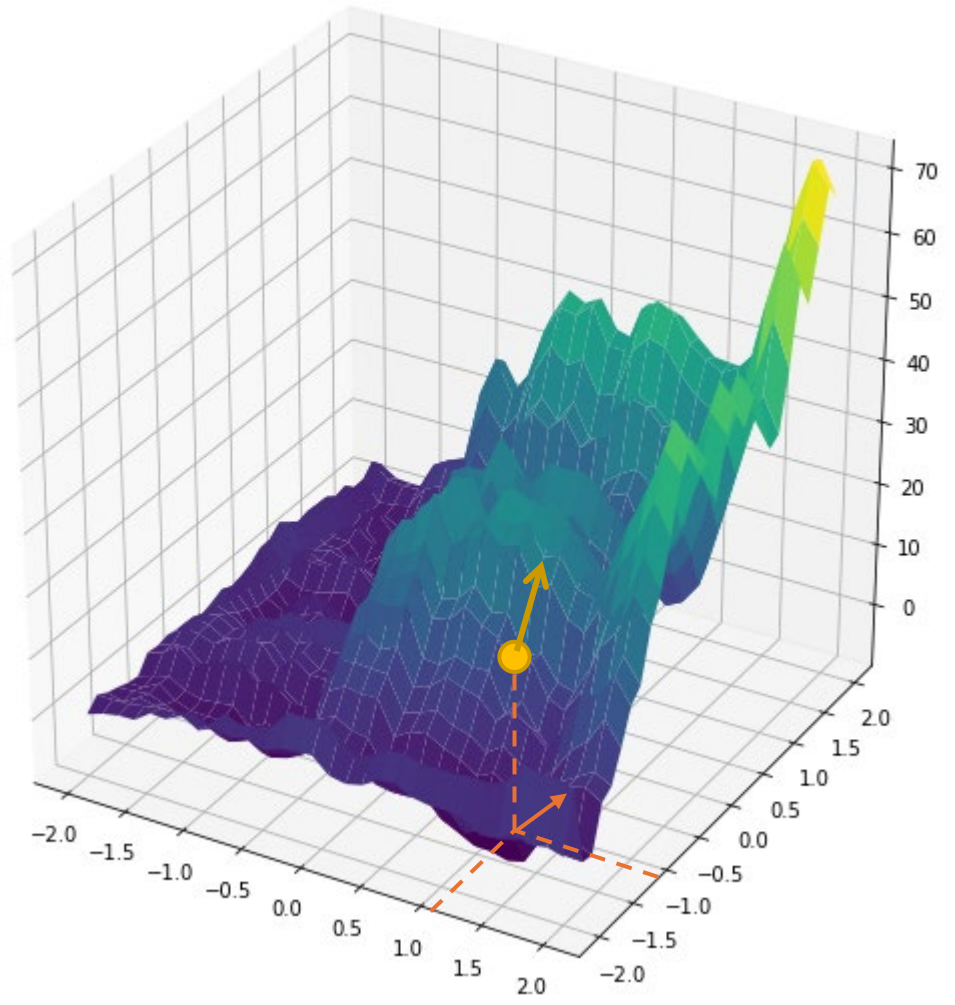
The Gradient

- The vector of partial image derivatives

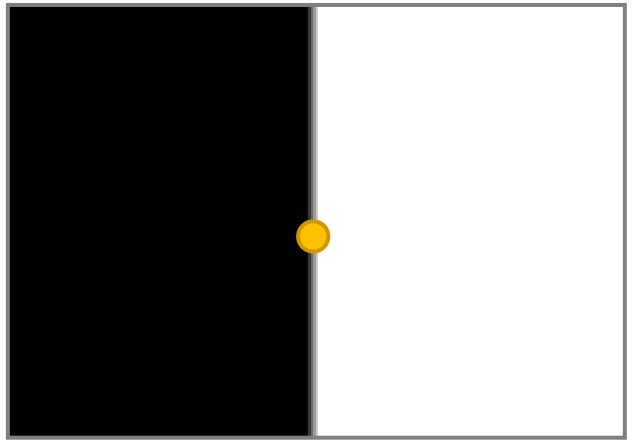
$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta \mathrm{y}} \end{pmatrix}(x,y)$$

is called the gradient.

- The gradient is pointing to the direction of maximum slope and is orthogonal to the edge's direction

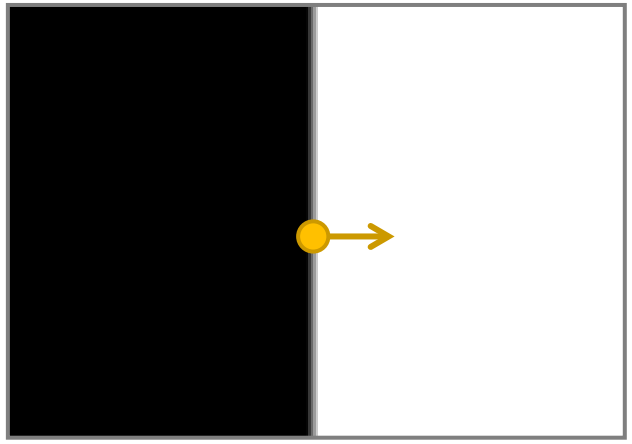


Computing the Image Gradient



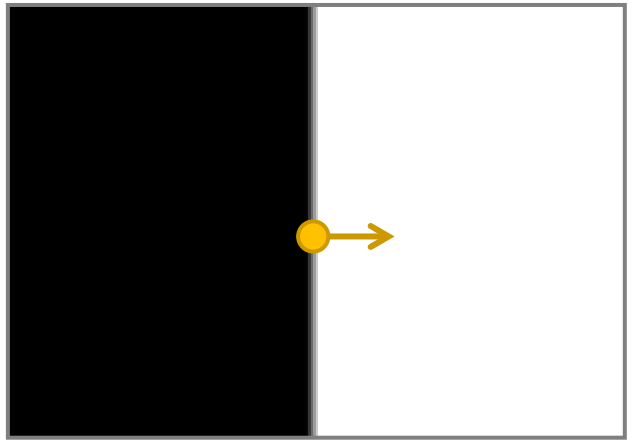
$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{pmatrix}(x,y) = \begin{pmatrix} ? \\ ? \end{pmatrix}$$

Computing the Image Gradient

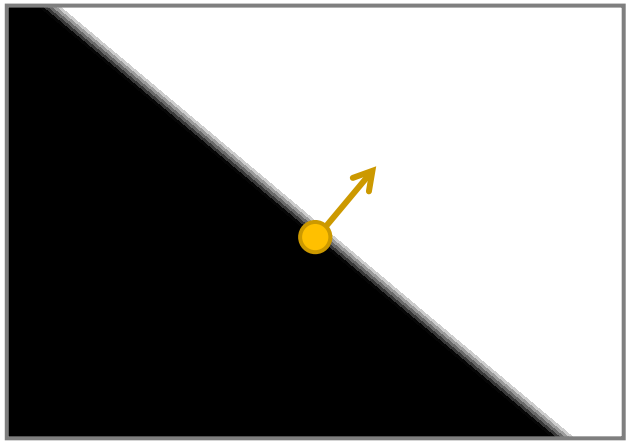


$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{pmatrix}(x,y) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Computing the Image Gradient

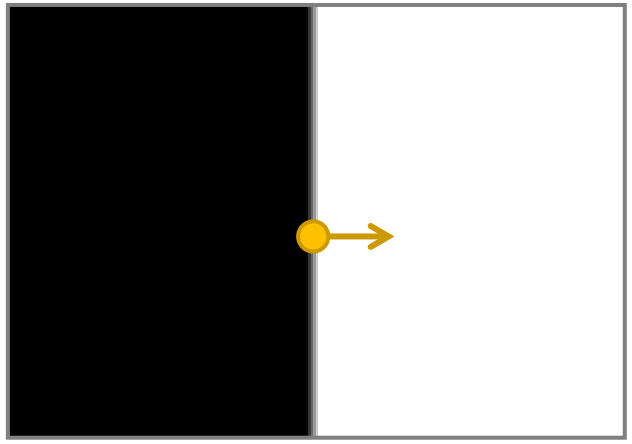


$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{pmatrix}(x,y) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{pmatrix}(x,y) = \begin{pmatrix} ? \\ ? \end{pmatrix}$$

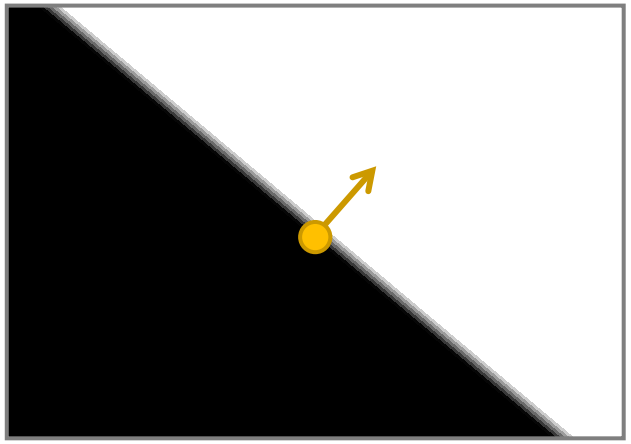
Computing the Image Gradient



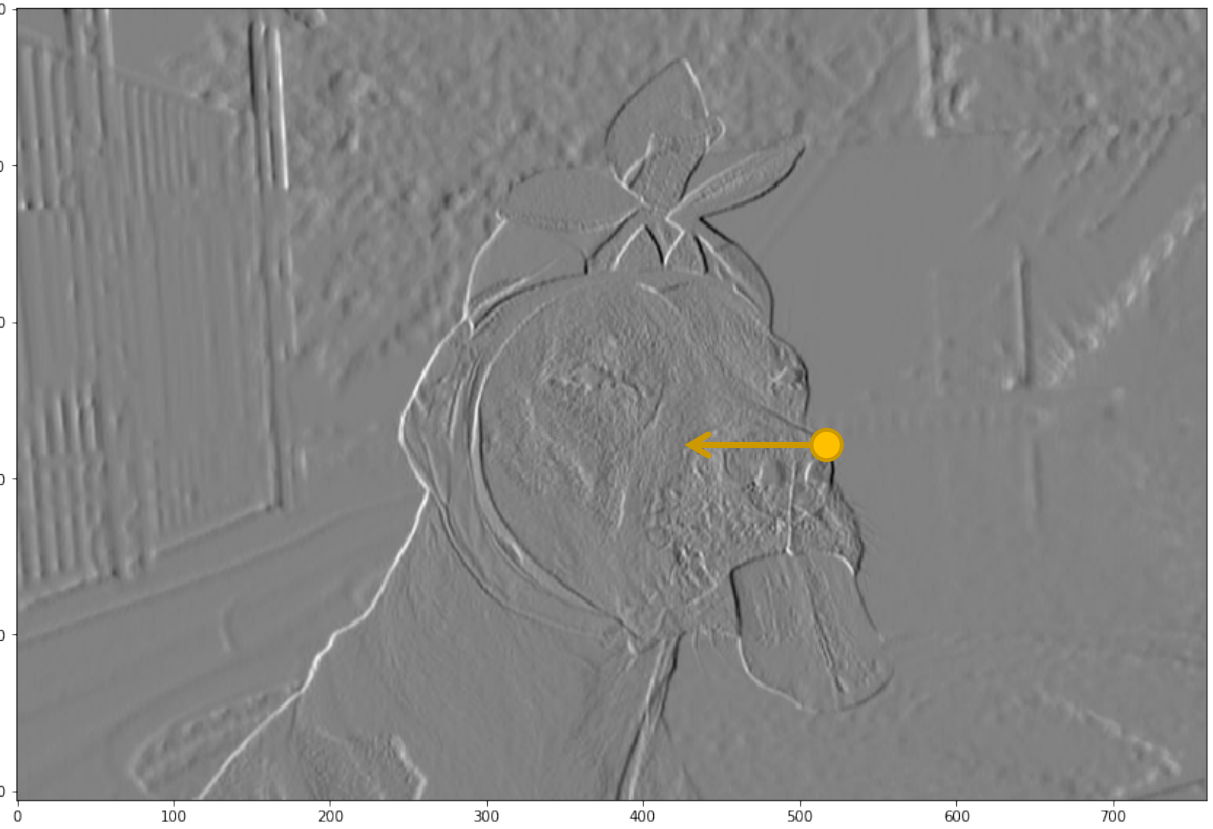
$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{pmatrix}(x,y) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

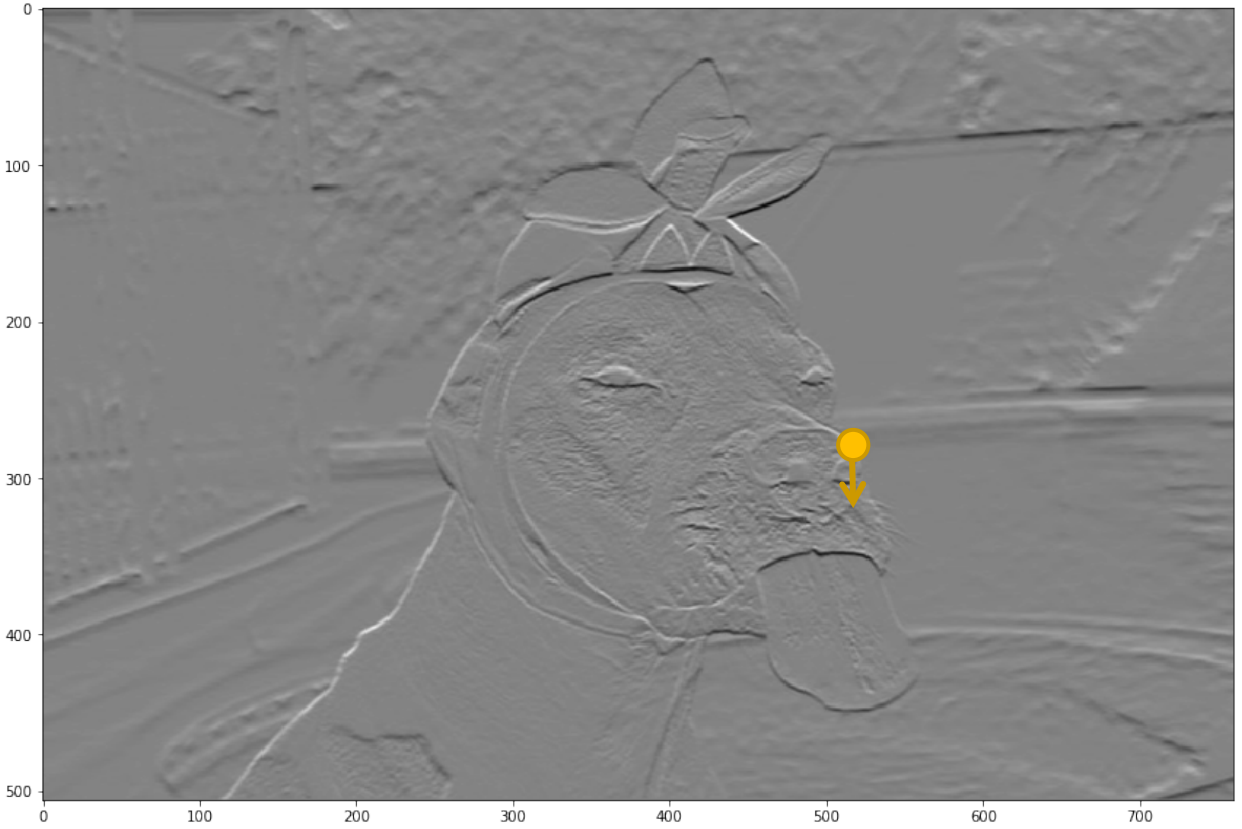
Use Sobel filter to estimate the image gradient

$$\nabla I(x,y) = \begin{pmatrix} G_x \\ G_y \end{pmatrix}(x,y)$$



$$\nabla I(x,y) = \begin{pmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{pmatrix}(x,y) = \begin{pmatrix} ? \\ ? \end{pmatrix}$$


0
100
200
300
400
500
600
700


0
100
200
300
400
500
0
100
200
300
400
500
600
700

Canny‘s Edge Detection

Cited by 29138

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

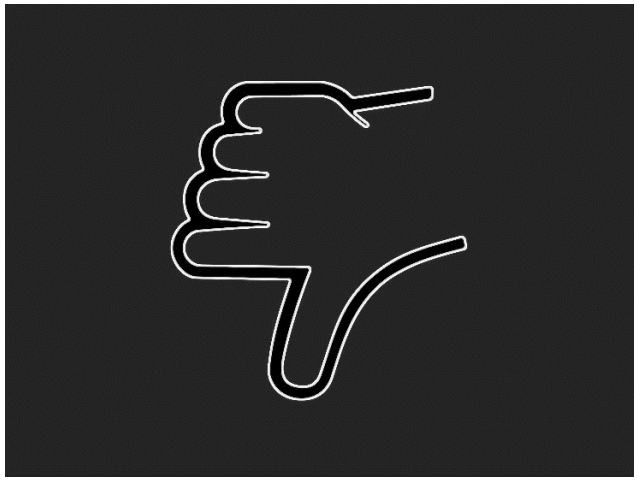
Abstract—**This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure that the detector has only one response to a single edge. We use the criteria in numerical optimization to derive detectors for several common image features, including step edges. On specializing the analysis to step edges, we find that there is a natural uncertainty principle between detection and localization performance, which are the two main goals. With this principle we derive a single operator shape which is optimal at any scale. The optimal detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image. We extend this simple detector using operators of several widths to cope with different signal-to-noise**

detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge contours or edge points in the image plane. Several modern theories of stereopsis assume that images are preprocessed by an edge detector before matching is done [19], [20]. Beattie [1] describes an edge-based labeling scheme for low-level image understanding. Finally, some novel methods have been suggested for the extraction of three-dimensional information from image contours, namely shape from contour [27] and shape from texture [31].

In all of these examples there are common criteria rel-

Requirements for a good edge detector

- detection of edge with **low error rate** (low false positives and false negative rates)
- edge point should be **localized** accurately on the center of the edge
- **no double detections**!



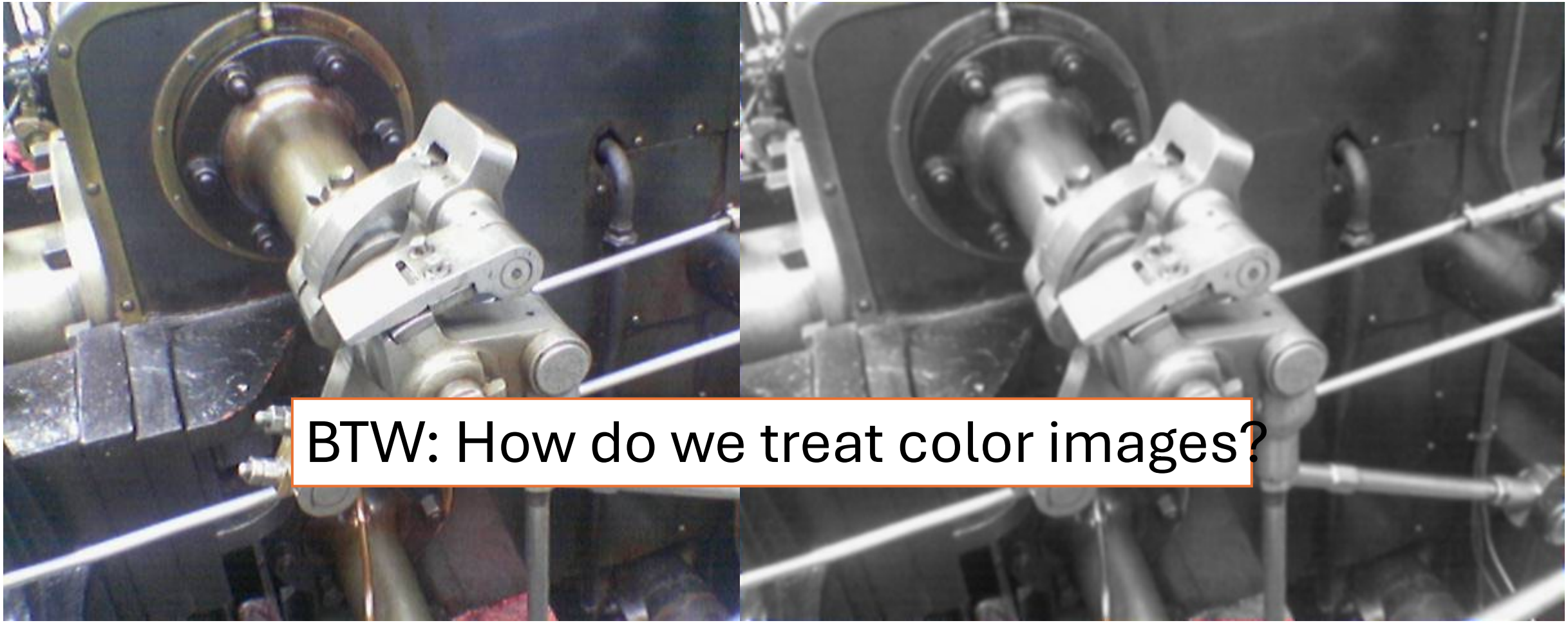
FREIE UNIVERSITÄT BERLIN

Outline of Canny‘s Edge Detection

- **convolve** with Gaussian to smooth the image and remove noise
- **convolve** again with Sobel to compute the image gradient
- apply **non-maximum suppression**
- use **two thresholds**: 1) for strong and 2) weak edges
- **hysteresis**: track edge along edge direction and suppress weak edges not connected to strong edges

Gaussian Smoothing

$$K = [? \ ? \ ?]$$



BTW: How do we treat color images?

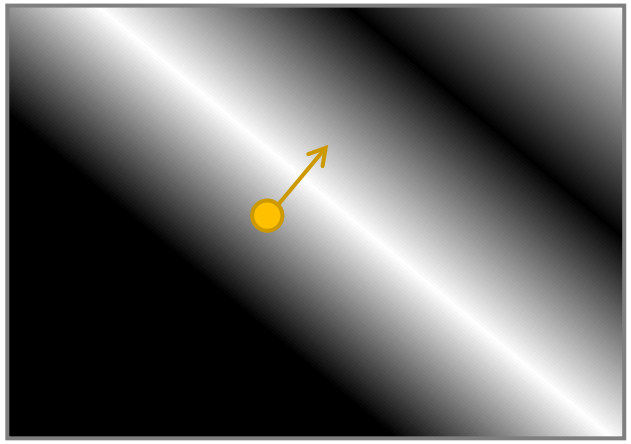
Image gradient magnitude and orientation

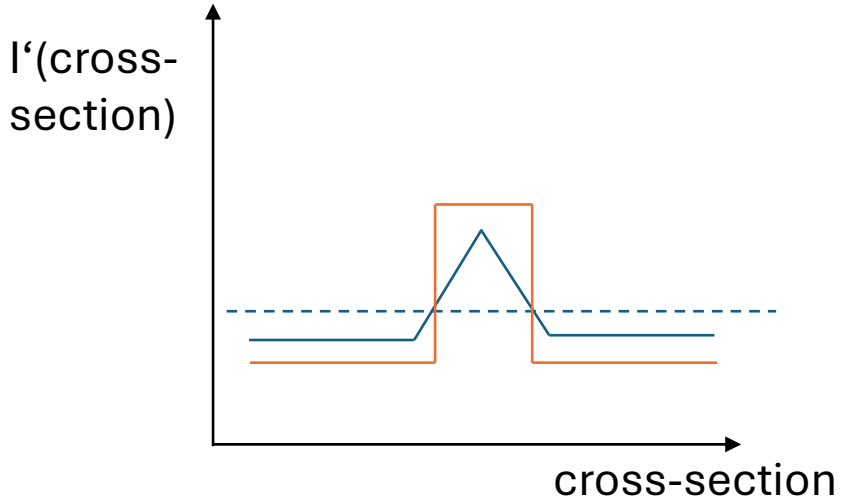
- Convolve with Sobel: $Gx = I * S_x$ and $G_y = I * S_y$ ✓
- The **gradient magnitude** is the length of the gradient vector
- atan2 returns the edge orientation between 0 and 2π
- θ is then rounded to eight discrete directions (corresponding to the 8-neighborhood)

$$G = \sqrt{G_x^2 + G_y^2} \qquad \Theta = atan2(G_y, G_x)$$

Non-maximum suppression: edge thinning

- Scan over all pixels, compare edge magnitude with that of its neighbors (as indicated by gradient orientation)
- Suppress if not greater





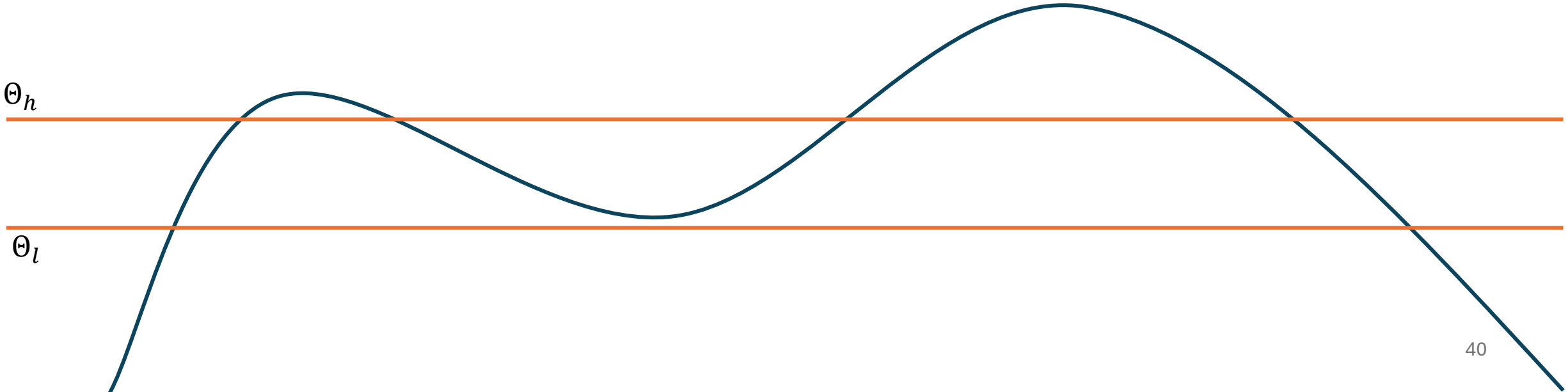
Non-maximum suppression: edge thinning

- Scan over all pixels, compare edge magnitude with that of its neighbors (as indicated by gradient orientation)
- Suppress if not greater
- **Q**: For which gradient direction would the center pixel not be suppressed?

5	10	10
11	10	6
10	10	3

Weak or strong edges?

- Two thresholds
- one defining strong edges, one for weak edges, else: no edge!
- Strong pixels: should be in the result set, weak pixels only if they are connected to strong edges!



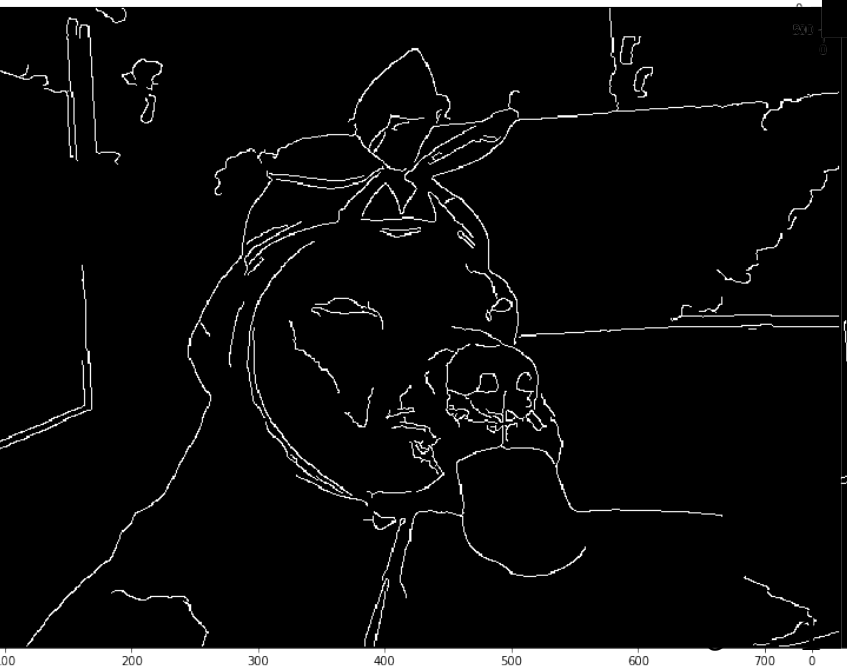
Effect of Low/High Thresholds

(using relative thresholds here:
0.4 means 40% of the maximum
gradient magnitude)

$\sigma = 1, l = 0.4, h = 0.4$

$\sigma = 1, l = 0.4, h = 0.6$

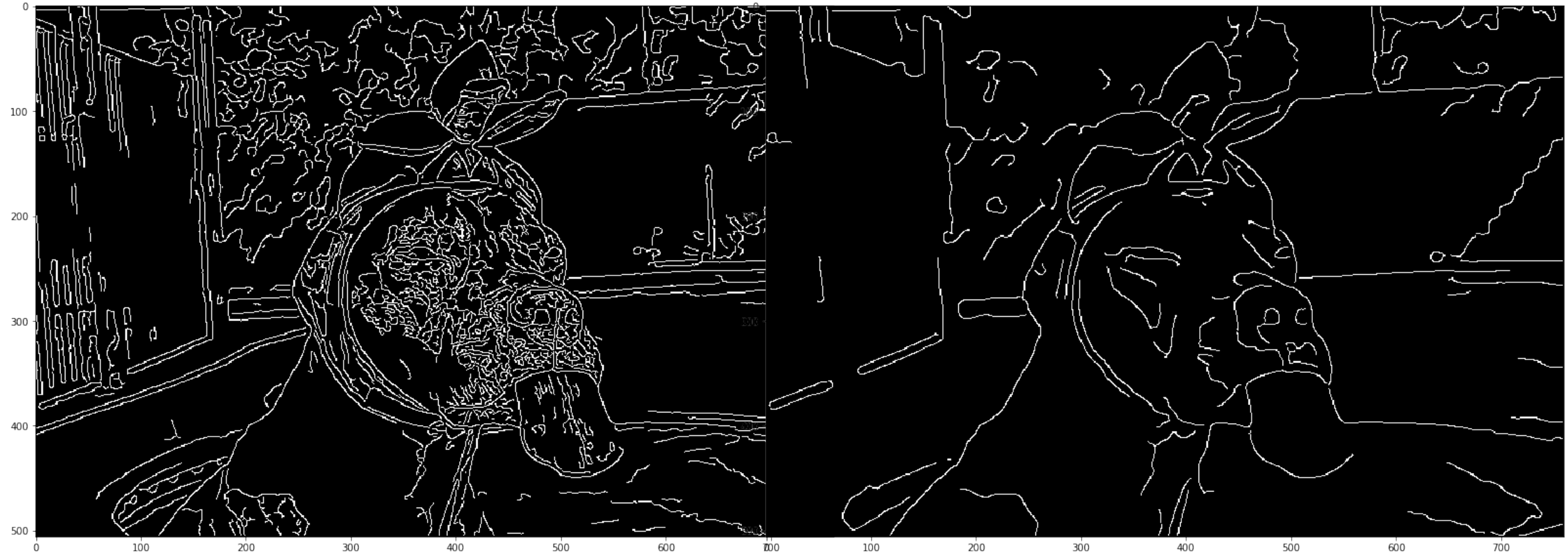
$\sigma = 1, l = 0.2, h = 0.6$



$\sigma = 1, l = 0.2, h = 0.4$

$\sigma = 1, l = 0.2, h = 0.2$

Effect of smoothing



$\sigma = 1$ $\sigma = 3$

FREIE UNIVERSITÄT BERLIN

Summary (aka “what will likely show up in the finals”)

- “Edges” = binary concept
  - Which properties can we use to detect them?
  - Requirements for “good” edge detectors
- Image derivatives via convolution with various kernels (e.g. Sobel)
  - What is convolution? What is it used for?
  - What is the Sobel kernel used for?
- Edge detection with Canny’s edge detector
  - Describe the steps
  - What are expected effects when changing parameters such as $\sigma, \theta_h, \theta_l$, etc?