Introduction to Computer Vision

2. Convolutions & filters, template matching, edge detection

02.11.22

Mikhail Belyaev

Skoltech
Skolkovo Institute of Science and Technology

Outline

- **Convolutions recap**
- Correlation & template matching
- Edge detection

Convolutions: some math

Question: what is g for moving average?

$$(f * g)(x, y) = \sum_{x'=0}^{X} \sum_{y'=0}^{Y} f(x', y') g(x - x', y - y')$$

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

This operation & the corresponding matrix are also called filters, kernels, convolutional matrices.

Convolutions: moving average

How does it modify images?

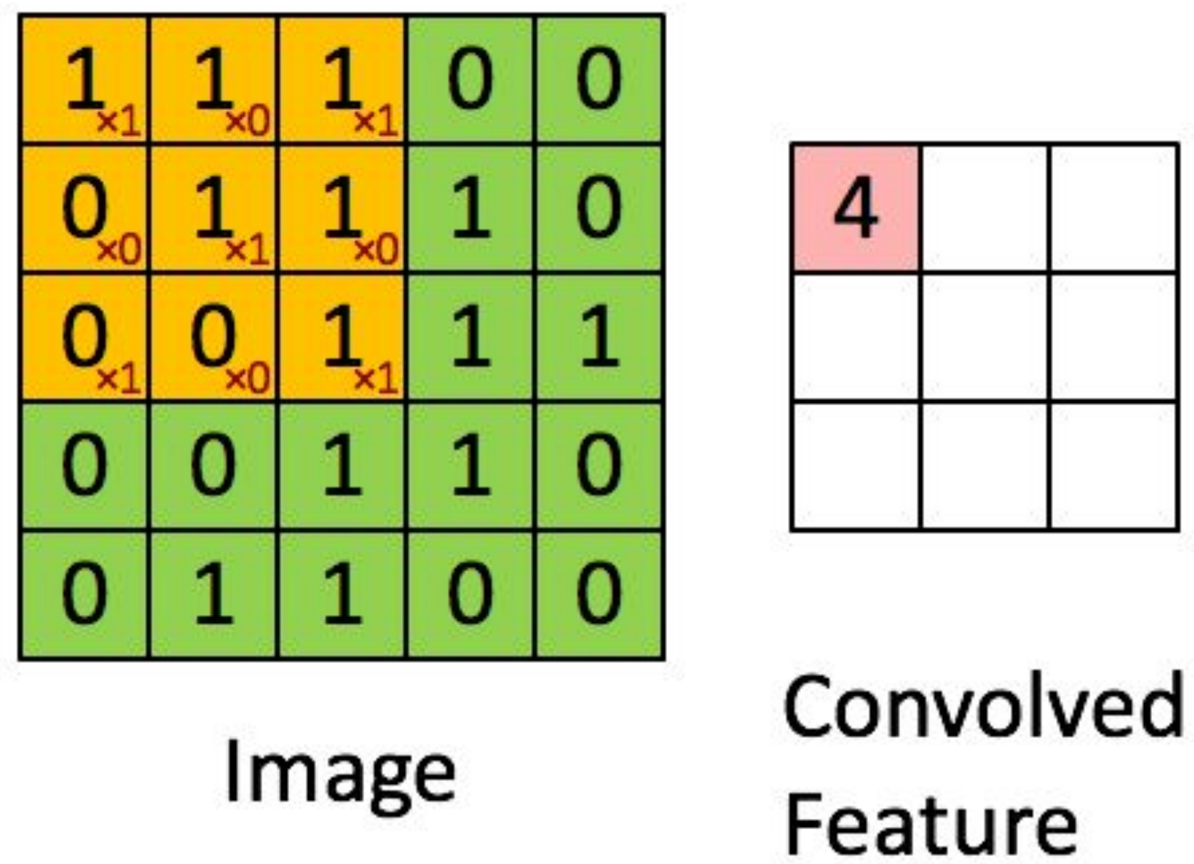
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Original image

Smoothed image

Source: J. Niebles

Convolutions in 2D



Source: [Stanford deep learning tutorial.]()

Outline

- Convolutions recap
- **Correlation & template matching**
- Edge detection

Skoltech
Skolkovo Institute of Science and Technology

Convolution vs Correlation

Cross correlation is another important operation we can apply to images:

$$(f \otimes h)(x, y) = \sum_{x'=0}^{X} \sum_{y'=0}^{Y} f(x', y') g(x' - x, y' - y)$$

Is it equivalent to convolution?

Skoltech
Skolkovo Institute of Science and Technology

Convolution vs Correlation

Cross correlation is another important operation we can apply to images:

$$(f \otimes h)(x, y) = \sum_{x'=0}^{X} \sum_{y'=0}^{Y} f(x', y') g(x' - x, y' - y)$$

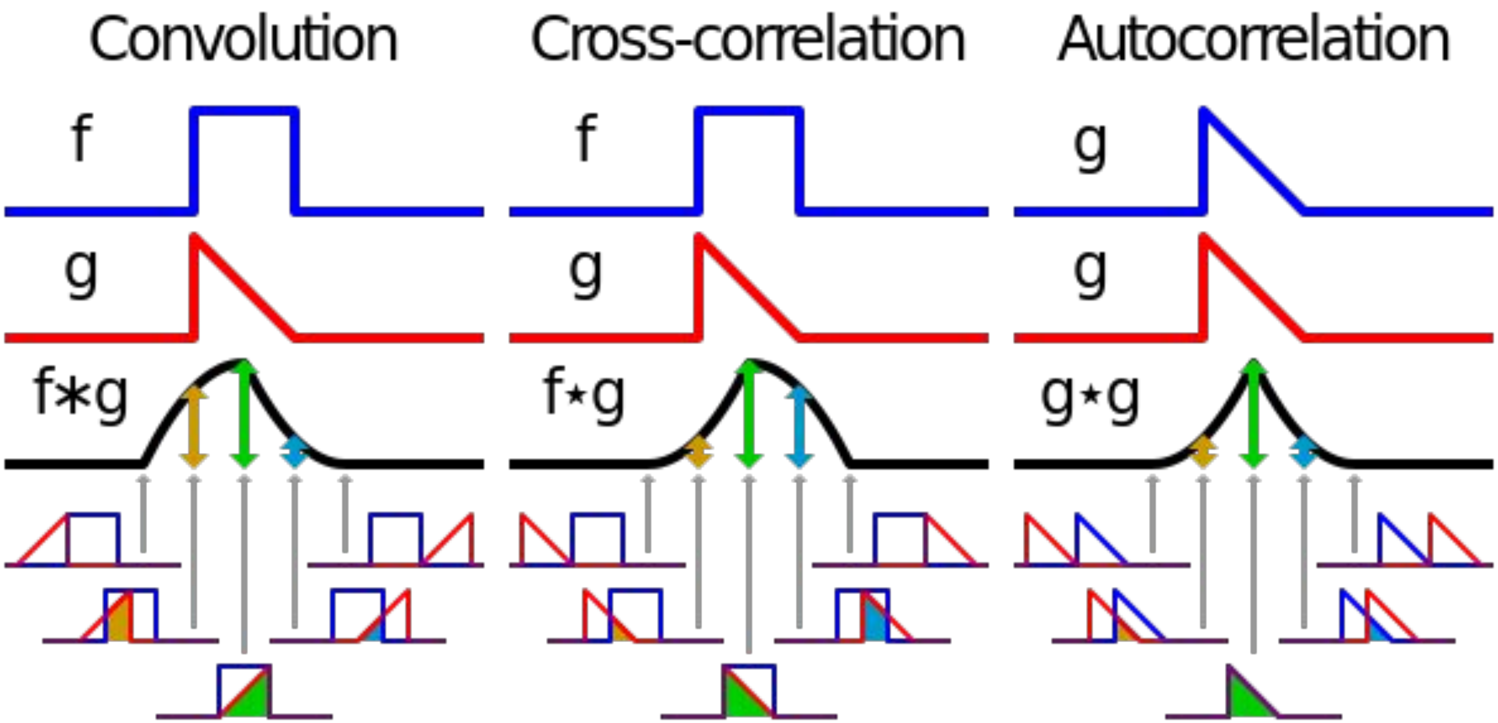
Is it equivalent to convolution? Not exactly!

$$(f * h)(x, y) = \sum_{x'=0}^{X} \sum_{y'=0}^{Y} f(x', y') g(x - x', y - y')$$

See also this blog post:
https://glassboxmedicine.com/2019/07/26/convolution-vs-cross-correlation/

Skoltech
Skolkovo Institute of Science and Technology

Convolution vs Correlation



Convolution = Cross-correlation with the flipped version of the kernel (along both axes for 2D)

Convolution vs Correlation

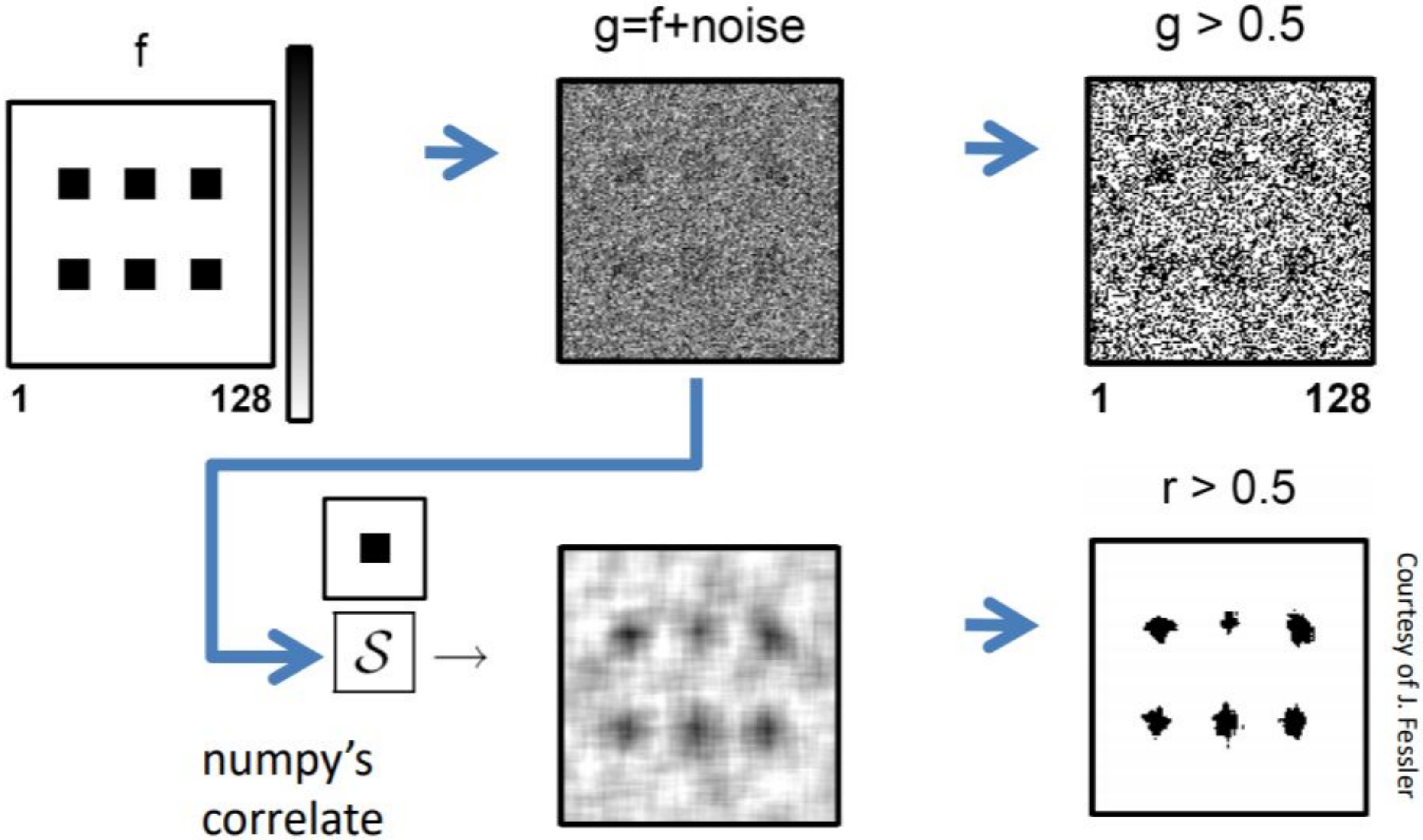
```python
def convolve(a, v, mode='full'):
    """
    Returns the discrete, linear convolution of two one-dimensional sequences.

    The convolution operator is often seen in signal processing, where it
    models the effect of a linear time-invariant system on a signal [1]_.  In
    probability theory, the sum of two independent random variables is
    distributed according to the convolution of their individual
    distributions.
    a, v = array(a, copy=False, ndmin=1), array(v, copy=False, ndmin=1)
    if (len(v) > len(a)):
        a, v = v, a
    if len(a) == 0:
        raise ValueError('a cannot be empty')
    if len(v) == 0:
        raise ValueError('v cannot be empty')
    return multiarray.correlate(a, v[::-1], mode)
```

https://github.com/numpy/numpy/blob/b235f9e701e14ed6f6f6dcba885f7986a833743f/numpy/core/numeric.py#L837-L844

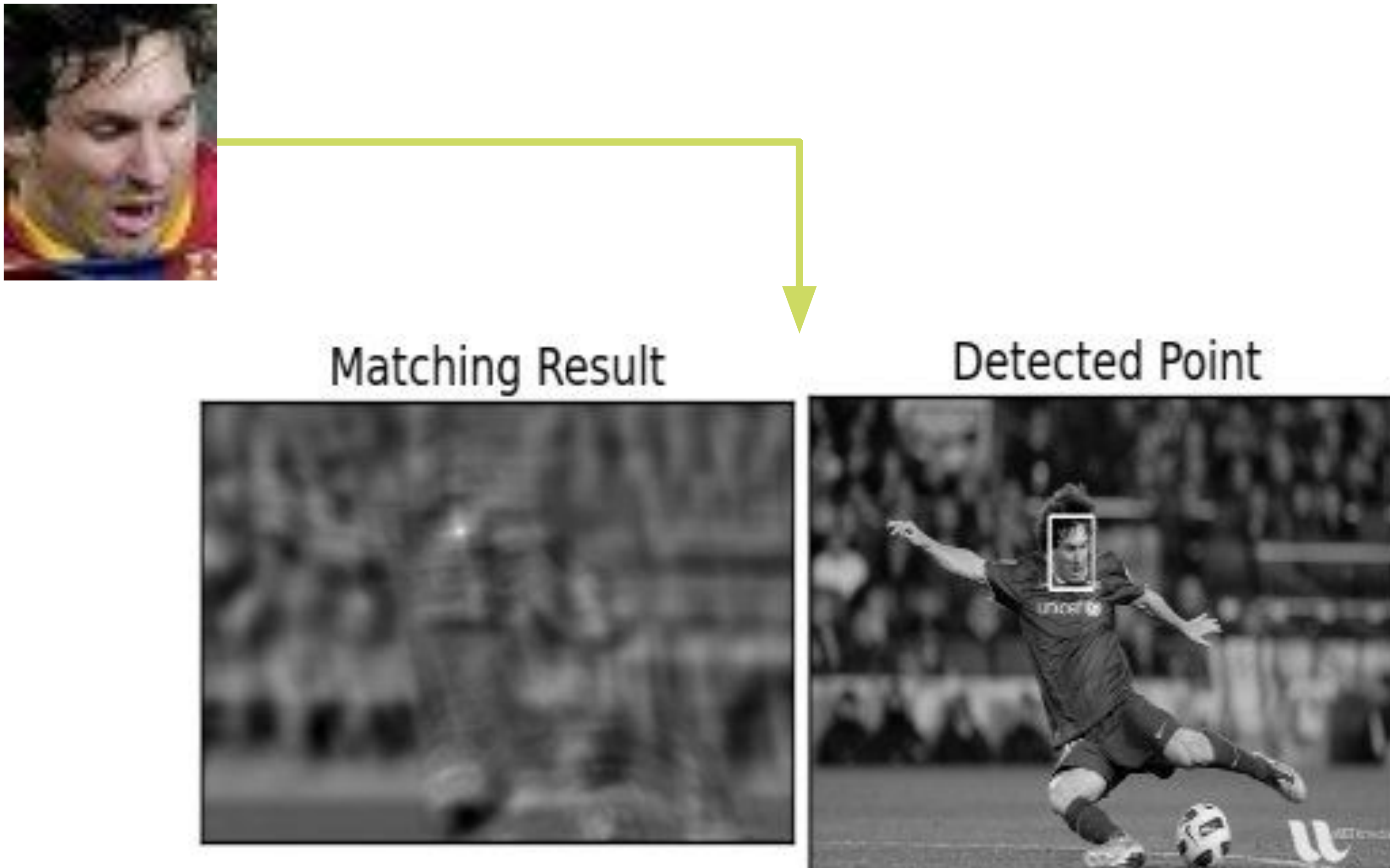
Skoltech
Skolkovo Institute of Science and Technology

Convolution vs Correlation



Source: J. Niebles

Template matching



https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html

Template matching

How can we estimate that a part of the image is **similar** to the template?

Template matching

How can we estimate that a part of the image is **similar** to the template?

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

Template matching

How can we estimate that a part of the image is **similar** to the template?

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x', y+y'))^2$$

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))$$

Skoltech
Skolkovo Institute of Science and Technology

Template matching

How can we estimate that a part of the image is **similar** to the template?

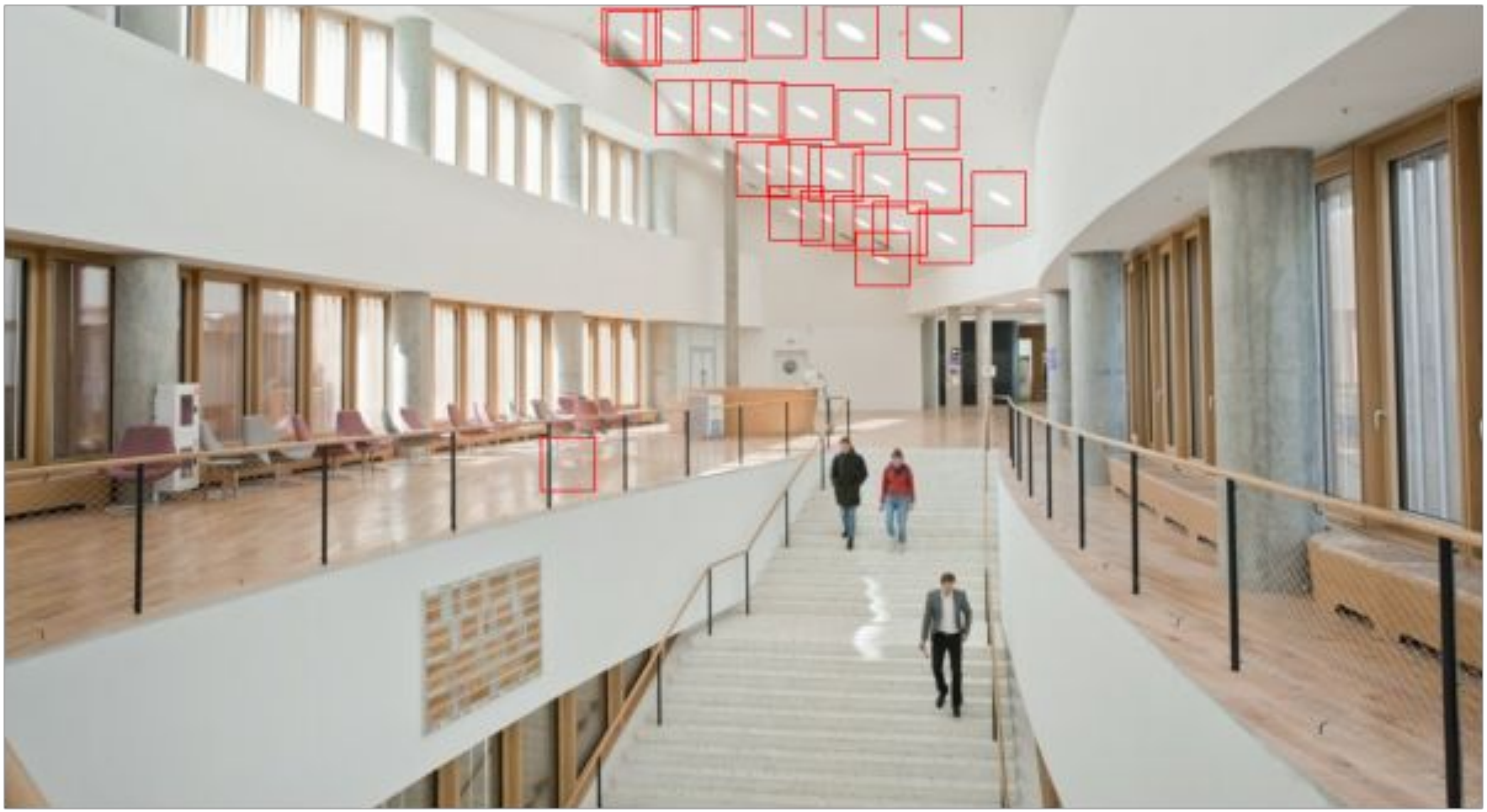
$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

Template matching

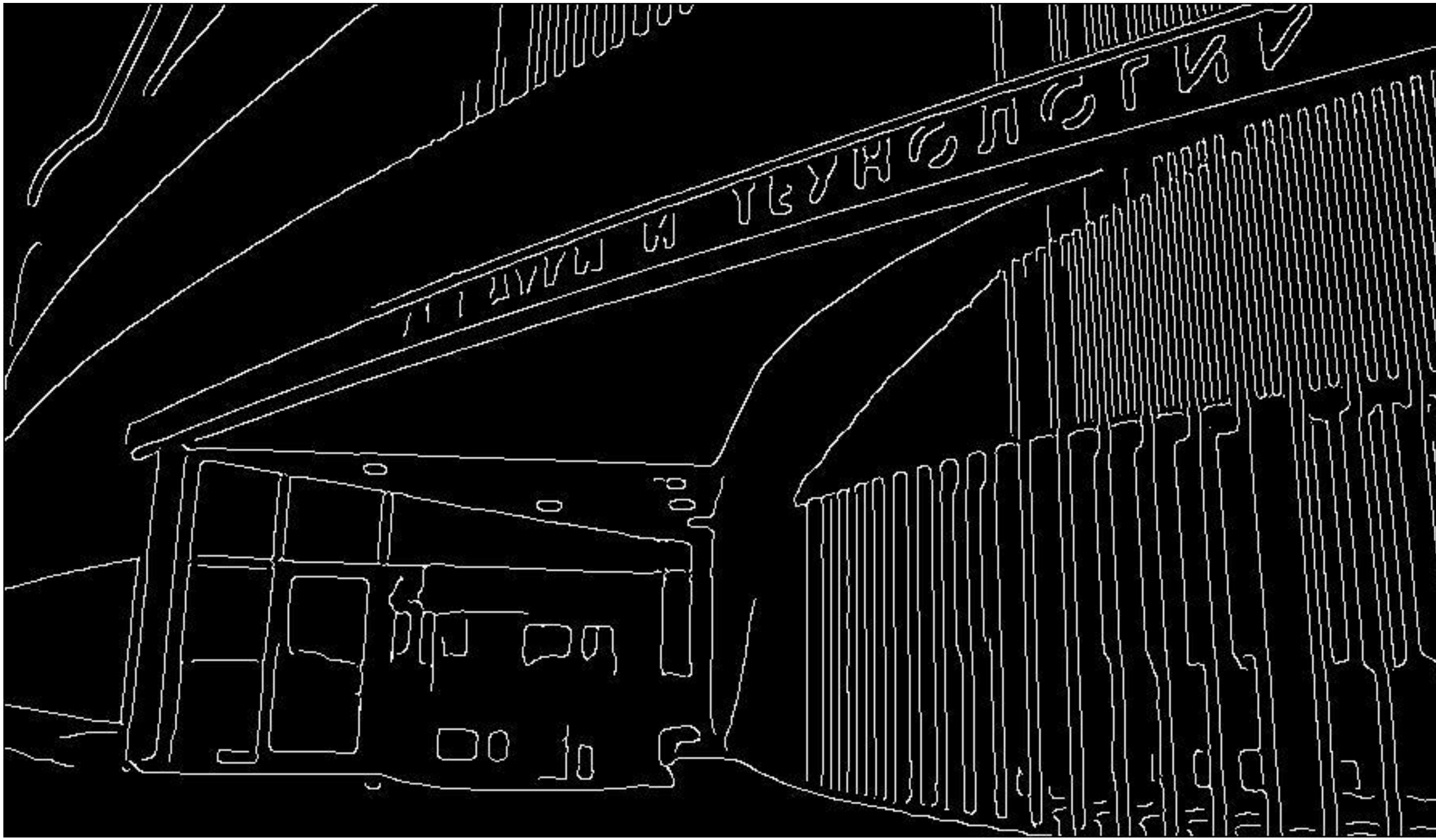
Exercise: count lamps using template matching!



Outline

- Convolutions recap
- Correlation & template matching
- **Edge detection**

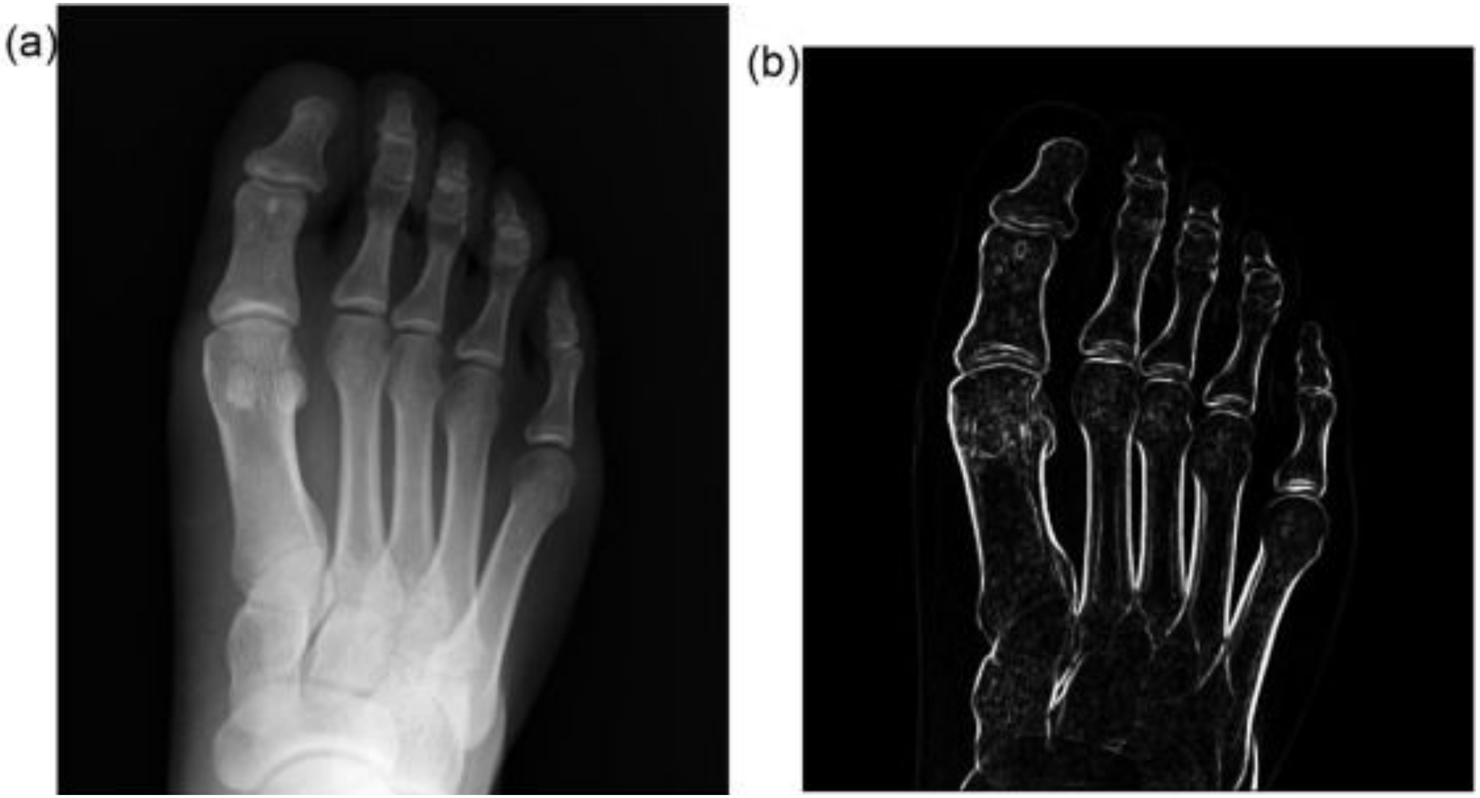
Why do we need to detect edges?



Why do we need to detect edges?

Skoltech
Skolkovo Institute of Science and Technology

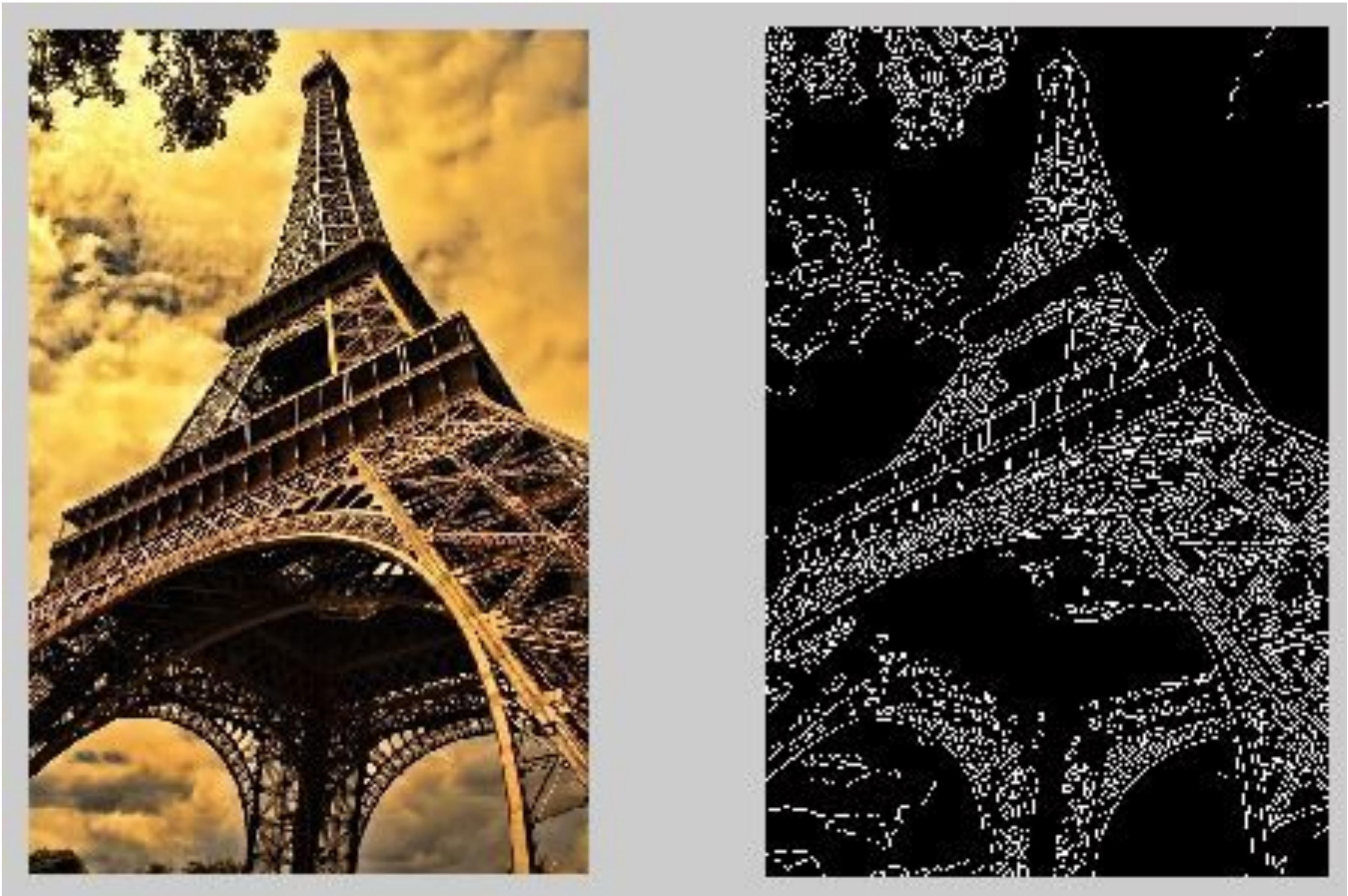
Why do we need to detect edges?



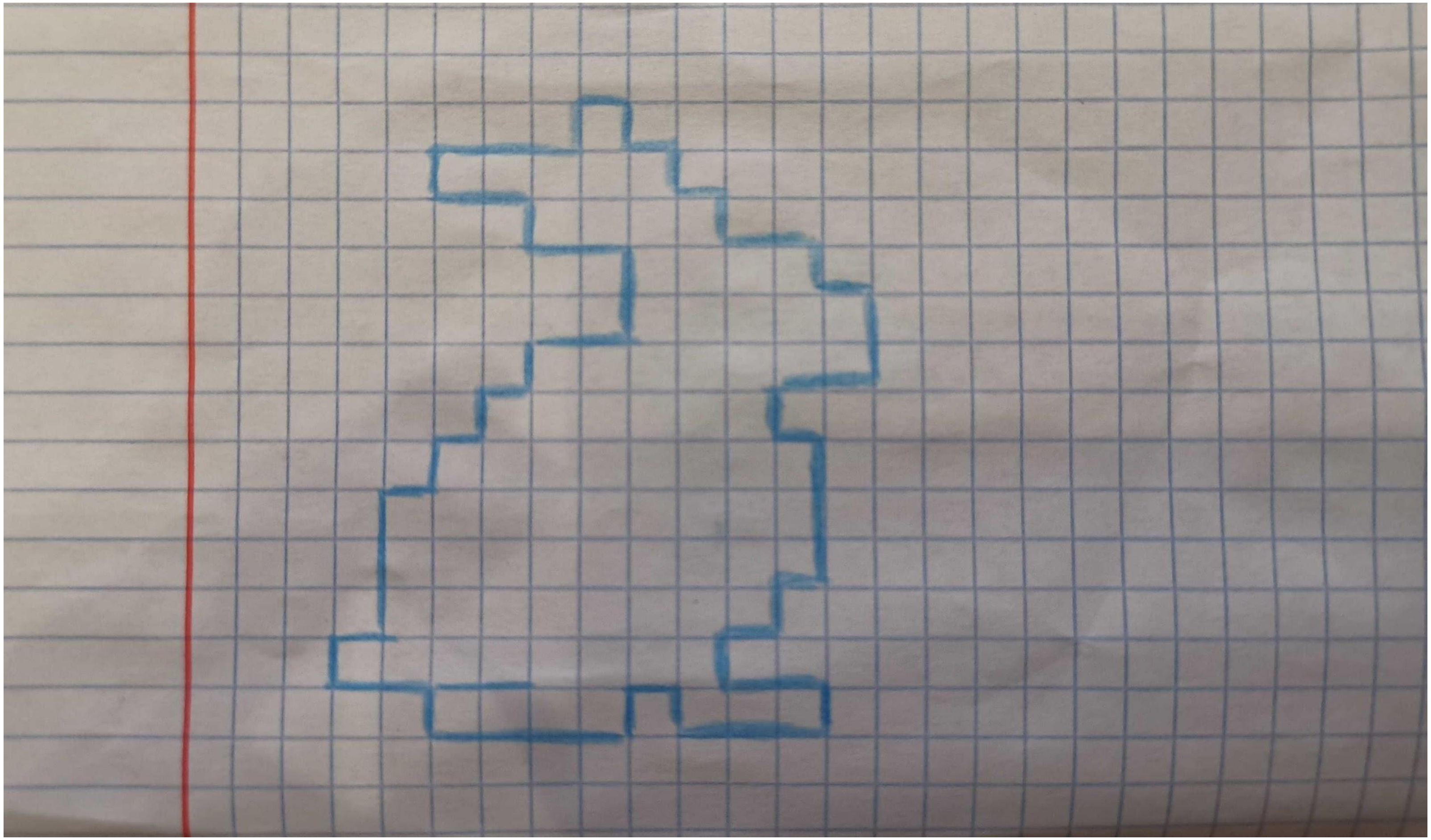
Source: Edge detection in medical images with quasi high-pass filter based on local statistics

Skoltech
Skolkovo Institute of Science and Technology

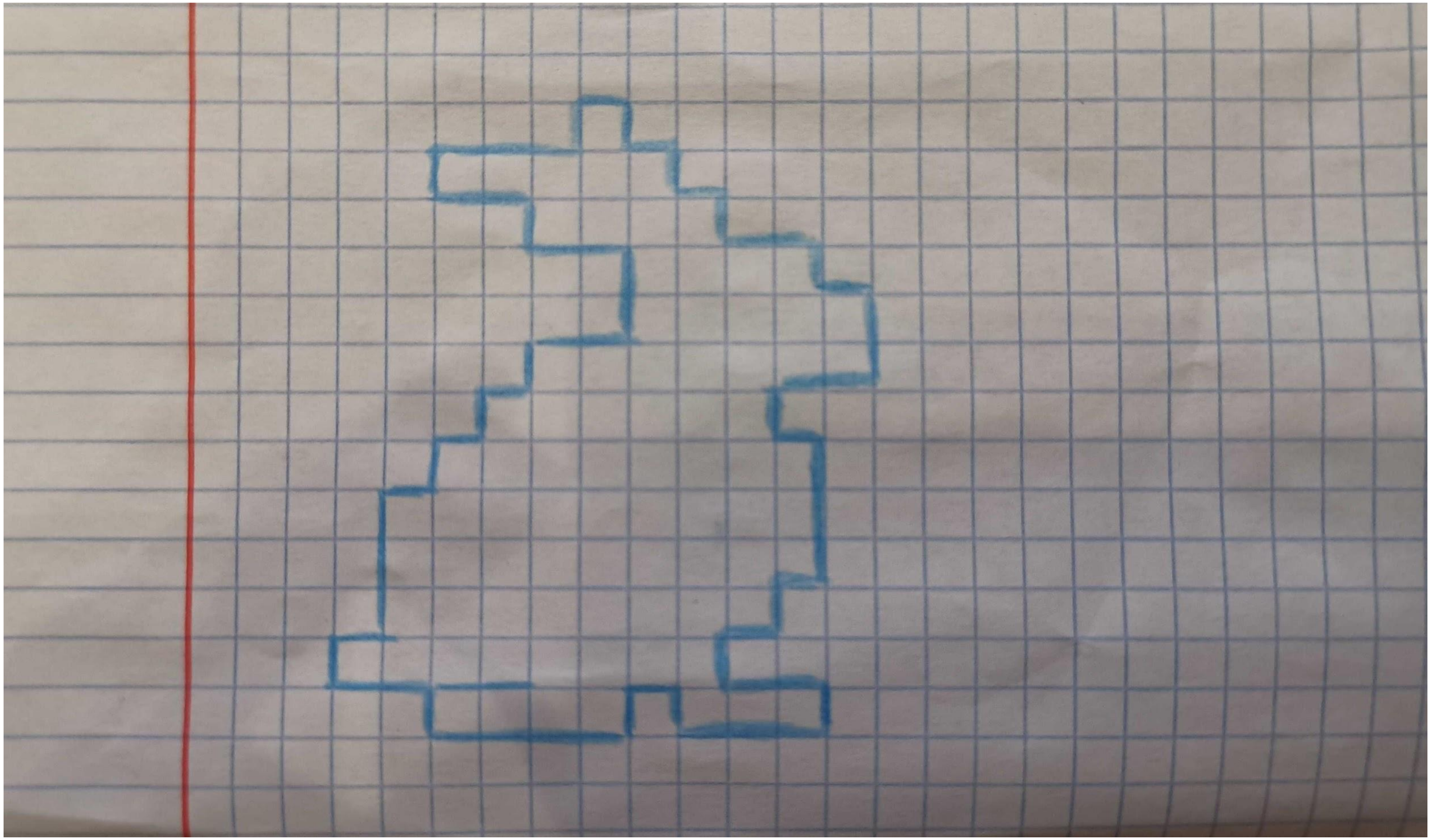
Why do we need to detect edges?



Why do we need to detect edges?

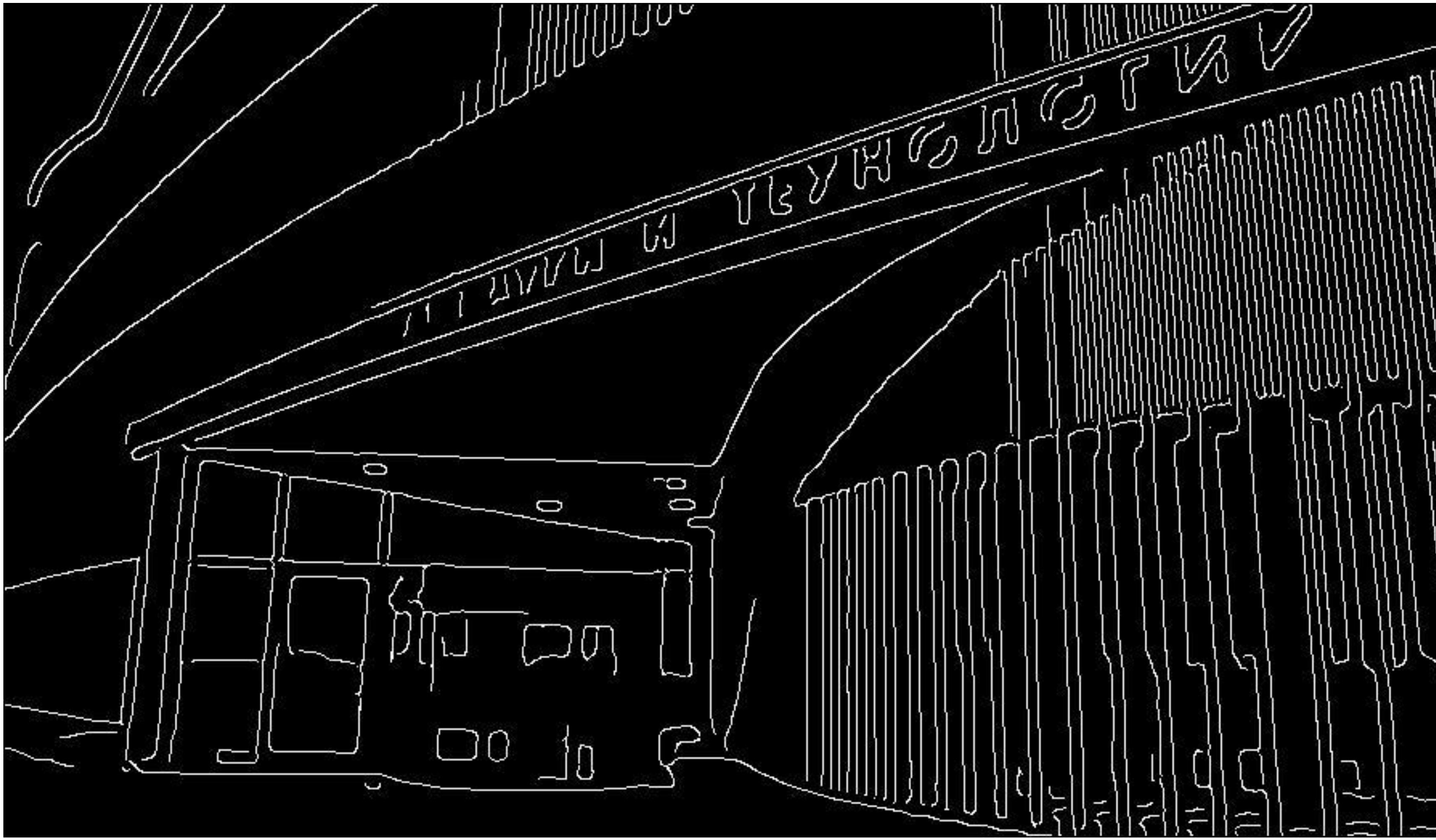


Why do we need to detect edges?



- Edges contain a lot of information about the image
- Generated by discontinuities, edges give a lot of information about separate objects, or their separate parts

Where do edges come from?



Please name 3 “sources” of edges on this photo

Skoltech
Skolkovo Institute of Science and Technology

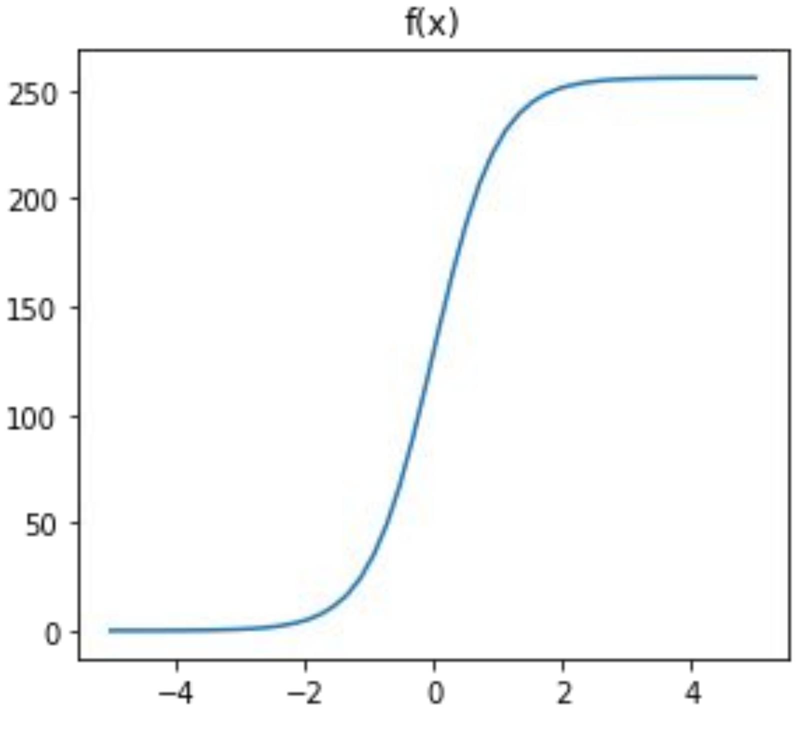
Where do edges come from?

Please name 3 “sources” of edges on this photo

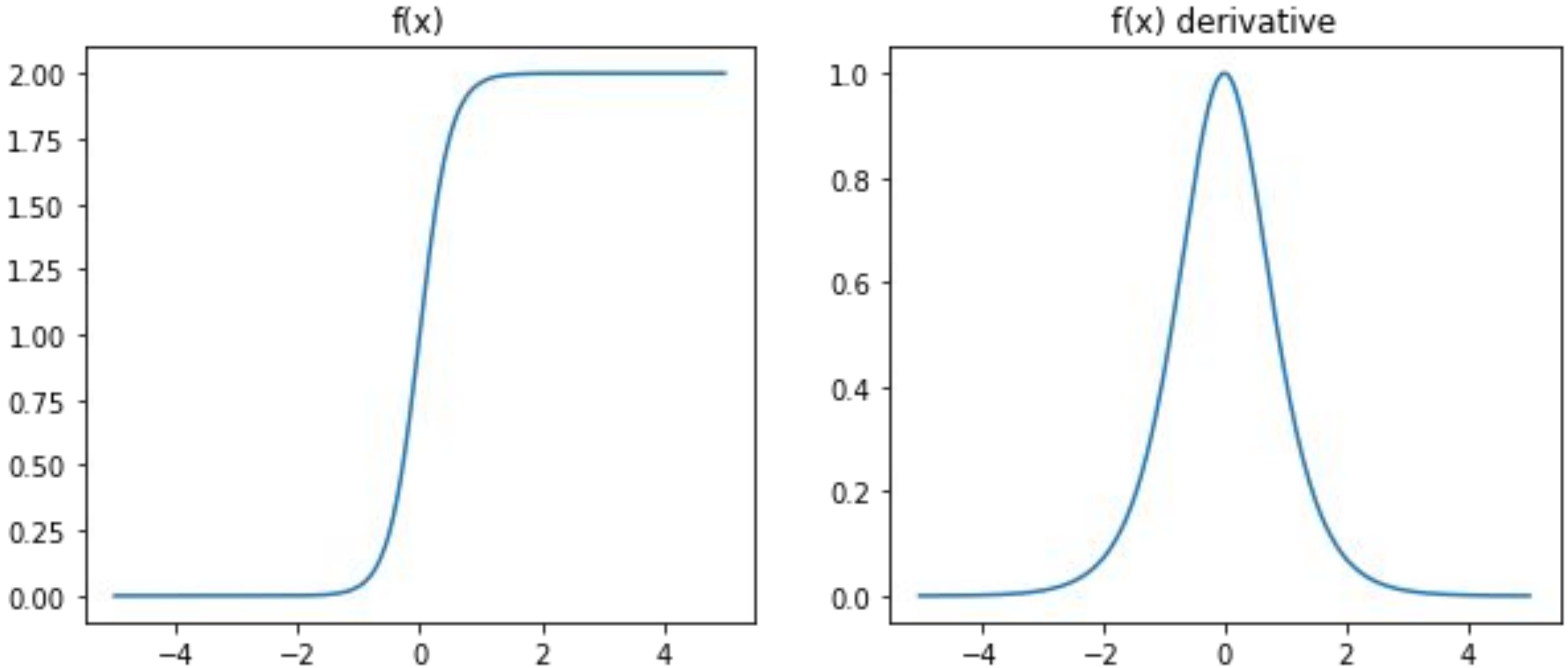
Where do edges come from?

1. Surface / Depth
2. Colors
3. Illumination

How to find discontinuities of a function?



How to find discontinuities of a function?



Derivative is a good way to find edges!

But what can we do with 2D discrete functions?

Skoltech
Skolkovo Institute of Science and Technology

Convolutions: examples of filters

How does it work?

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

Source: Wikipedia

Convolutions: examples of filters

How does it work?

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

Source: Wikipedia

Convolutions: examples of filters

How does it work?

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Source: Wikipedia

Separable convolutions

What is the time complexity of a naive convolution algorithm for an image of shape (n, m) and a kernel of shape (k, l)?

Skoltech
Skolkovo Institute of Science and Technology

Separable convolutions

Sometimes kernels can be decomposed into outer product of two vectors:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Separable convolutions

Sometimes kernels can be decomposed into outer product of two vectors:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

In this case we can exploit the associativity of convolution:

$$f \star (v \star w) = f \star (vw) = (f \star v) \star w$$

Skoltech
Skolkovo Institute of Science and Technology

Separable convolutions

Sometimes kernels can be decomposed into outer product of two vectors:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

In this case we can exploit the associativity of convolution:

$$f \star (v \star w) = f \star (vw) = (f \star v) \star w$$

What is the time complexity of a naive *separable* convolution algorithm for an image of shape (n, m) and a kernel of shape (k, l)?

Sobel filter

The x-derivative of smoothed image

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Gaussian smoothing

Differentiation

Sobel filter

Approximation of differentiation for both directions on the image:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

Horizontal changes

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Vertical changes

Skoltech
Skolkovo Institute of Science and Technology

Sobel filter

Approximation of differentiation for both directions on the image:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

Horizontal changes

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Vertical changes

Gradient magnitude $\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$

Gradient direction $\mathbf{\Theta} = \operatorname{arctan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$

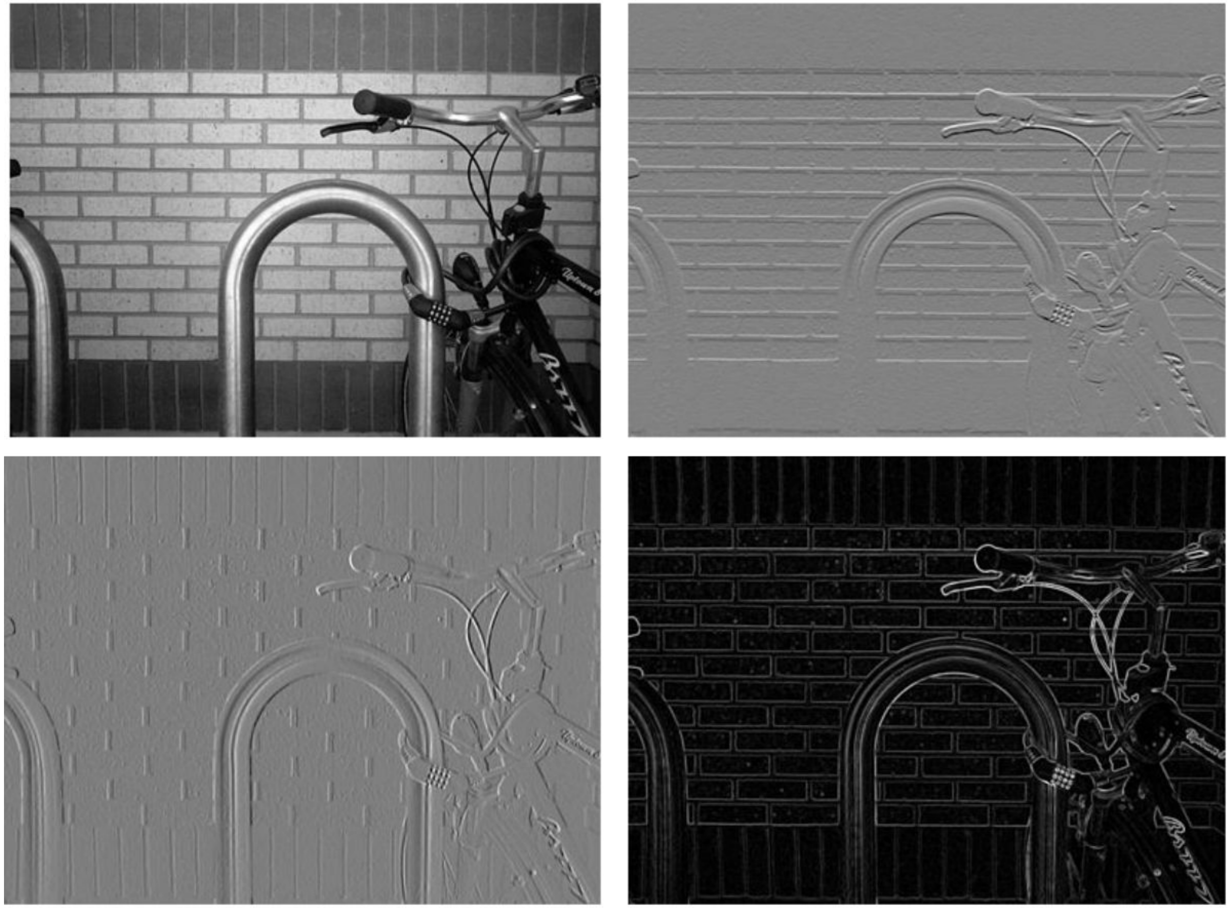
Skoltech
Skolkovo Institute of Science and Technology

Sobel filter (example)

Grayscale image

y-derivative

G_y



x-derivative

G_x

Gradient magnitude

G

Source: Wikipedia

Edge detection by Sobel

Algorithm:

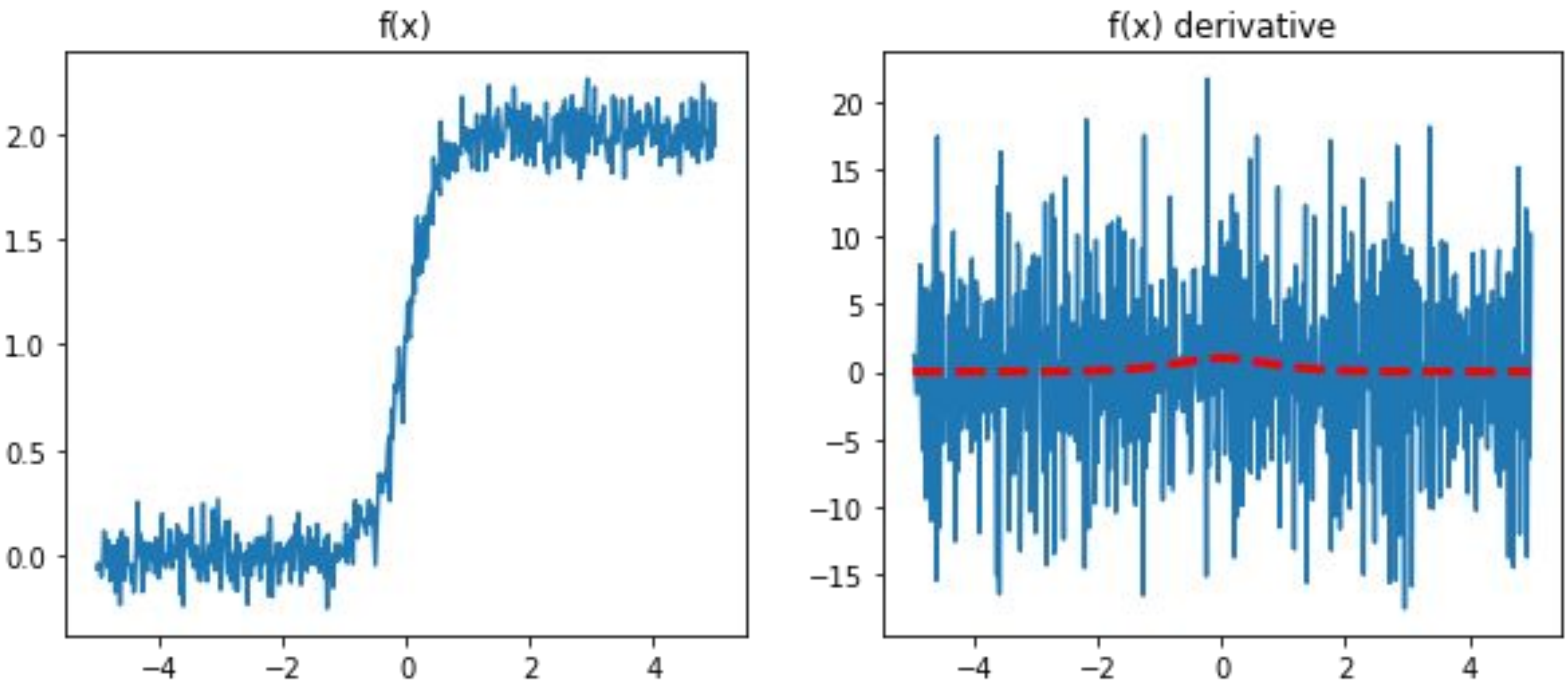
1. Convolve the image with filters G_X and G_y to estimate image derivatives.

2. Calculate gradient magnitude $\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$

Note: these edges aren't binary

Skoltech
Skolkovo Institute of Science and Technology

Sobel filter (issues)

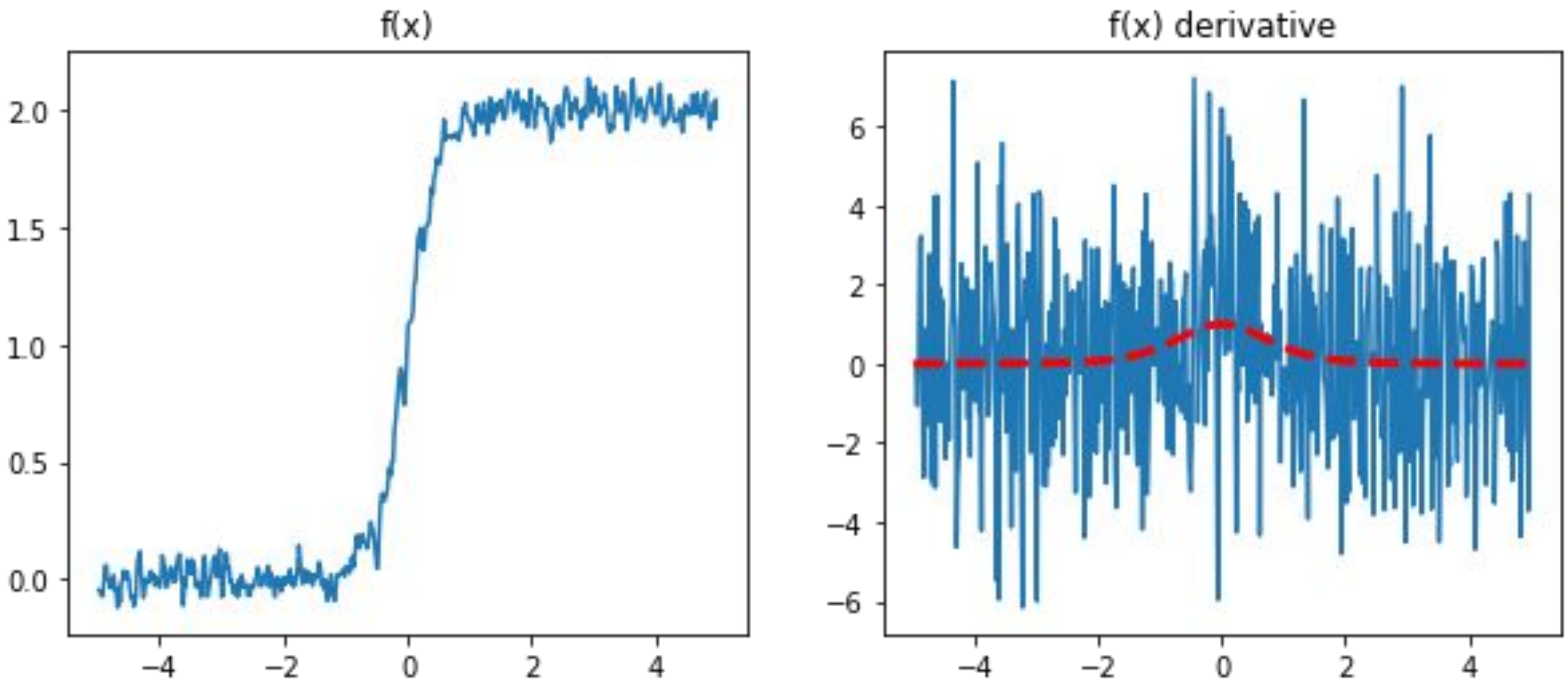


Numerical derivatives are sensitive to noise (dashed red line represents the analytical derivative).

What can we do with our image?

Skoltech
Skolkovo Institute of Science and Technology

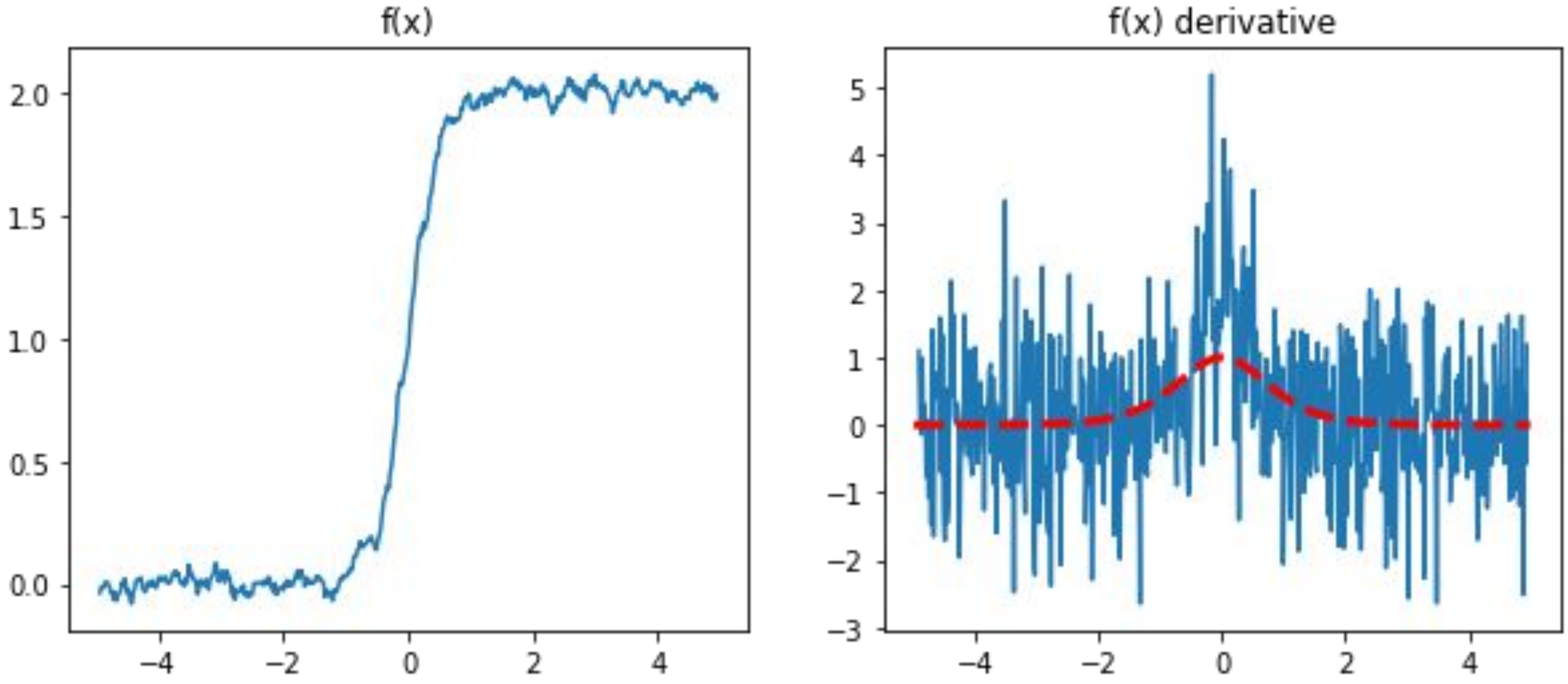
Sobel filter (issues)



Smoothing! For example, moving average with window=3

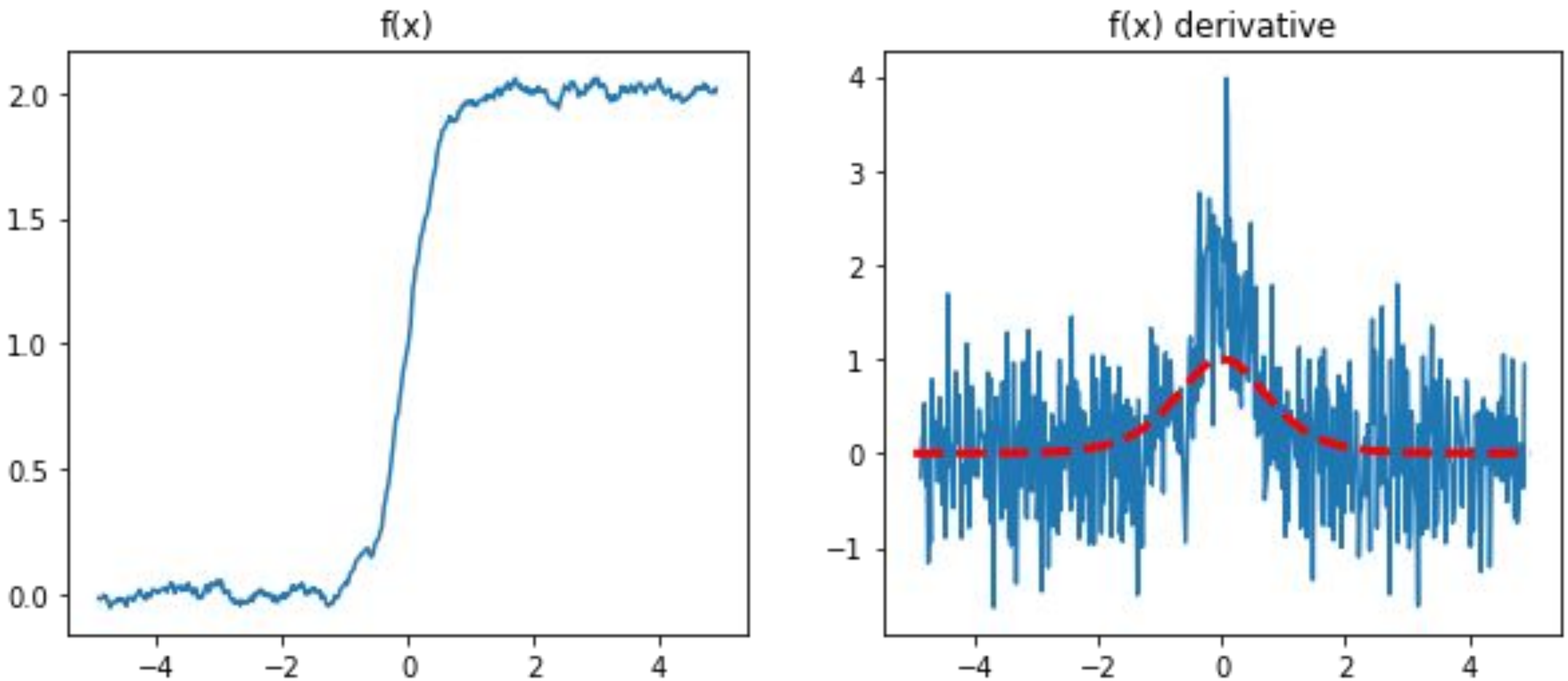
Skoltech
Skolkovo Institute of Science and Technology

Sobel filter (issues)



Smoothing! For example, moving average with window=7

Sobel filter (issues)



Smoothing! For example, moving average with window=11

Edge detection - desired features

- Detection of edge with low false negative error rate, which means that the detection should accurately catch as many edges shown in the image as possible.

- The edge point should accurately localize on the center of the edge.

- A given edge in the image should only be marked once, and where possible, image noise should not create false edges (low false positive errors rate).

Skoltech
Skolkovo Institute of Science and Technology

Canny edge detector - steps

1. Noise suppression (e.g. **gaussian filter**, median filter)

2. Gradient magnitude and direction (e.g. via Sobel filter)

3. **Non-Maximum Suppression**

4. **Hysteresis thresholding and connectivity analysis**

J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

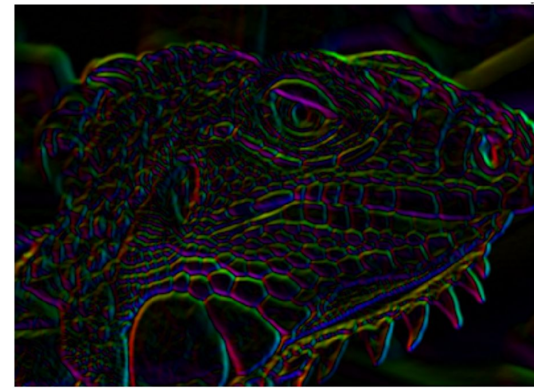
Skoltech
Skolkovo Institute of Science and Technology

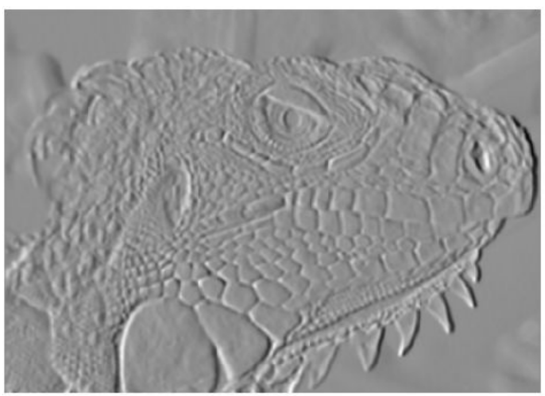
Canny edge detector (step-by-step)

Noise suppression + gradient calculation

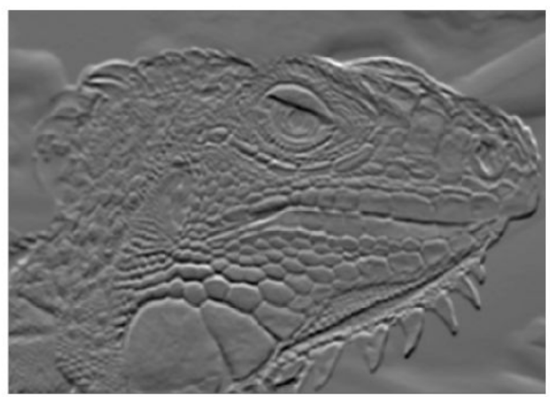
orig. image

gradient orientation





X-Derivative of Gaussian



Y-Derivative of Gaussian

Gradient Magnitude

Source: J. Hayes

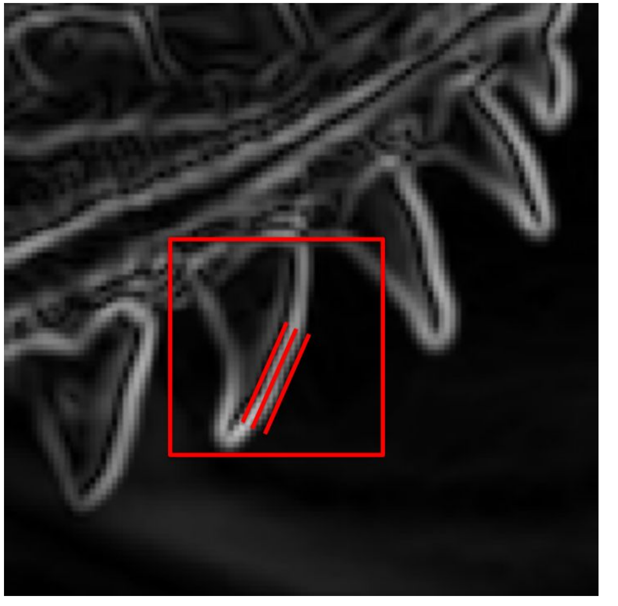
Skoltech

Skolkovo Institute of Science and Technology

Canny edge detector (step-by-step)

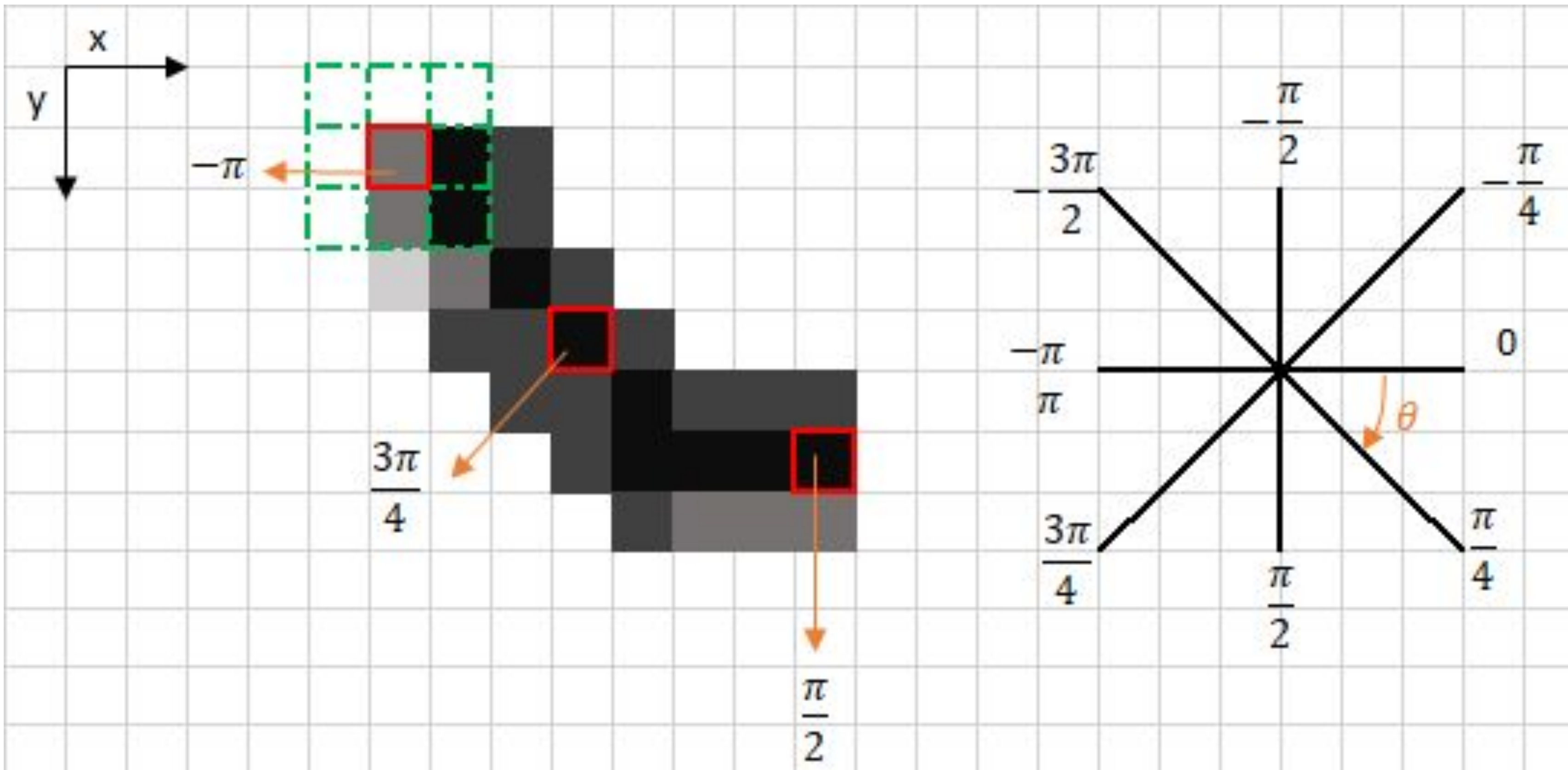
Non-Maximum Suppression

- Edge occurs where gradient reaches a (local) maxima
- Consider only 8 angle directions (e.g. 45º, 90º, 135º, ...)
- Suppress all pixels in each direction which are not maxima



Canny edge detector (step-by-step)

Non-Maximum Suppression

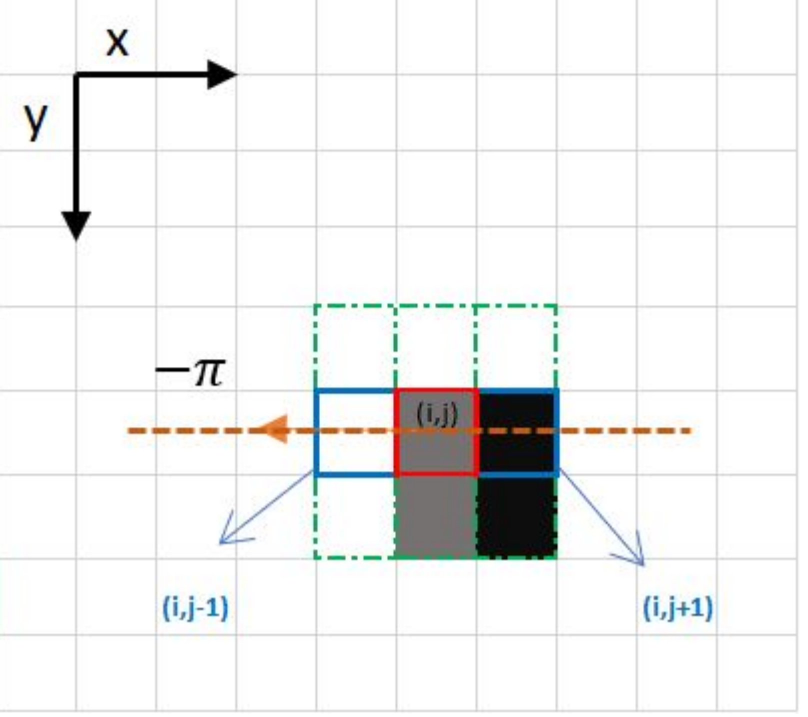


Skoltech
Skolkovo Institute of Science and Technology

Canny edge detector (step-by-step)

Non-Maximum Suppression

- the pixel **(i, j)** is being processed
- Gradient orientation is *approximately* -π (orange line).
- We consider pixels on the same **gradient direction**: **(i, j-1)** and **(i, j+1)**
- if **(i, j)** is more intense than these two neighbors, then it is kept
- overwise, it is suppressed (set to 0)

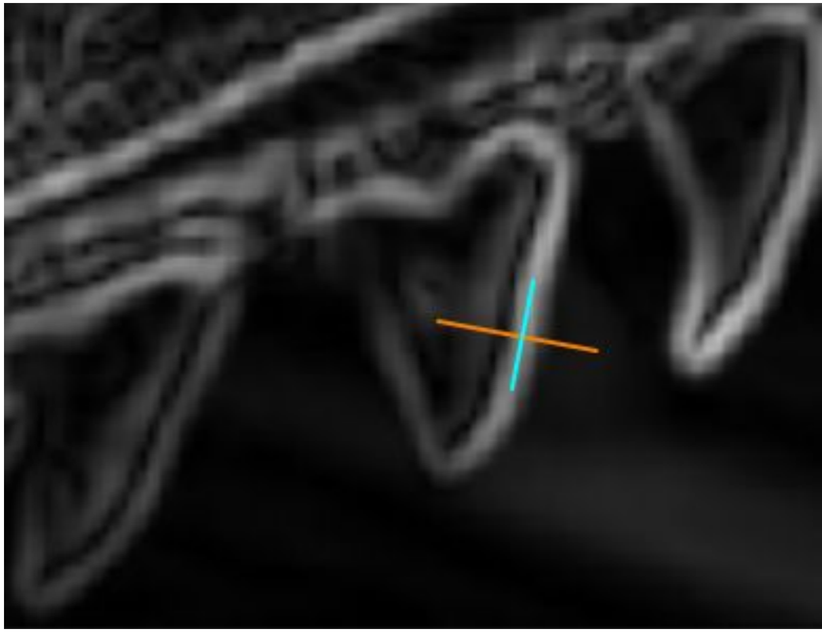


Skoltech
Skolkovo Institute of Science and Technology

Canny edge detector (step-by-step)

Non-Maximum Suppression

- the pixel **(i, j)** is being processed
- Gradient orientation is *approximately* -π (orange line).
- We consider pixels on the same **gradient direction**: **(i, j-1)** and **(i, j+1)**
- if **(i, j)** is more intense than these two neighbors, then it is kept
- overwise, it is suppressed (set to 0)



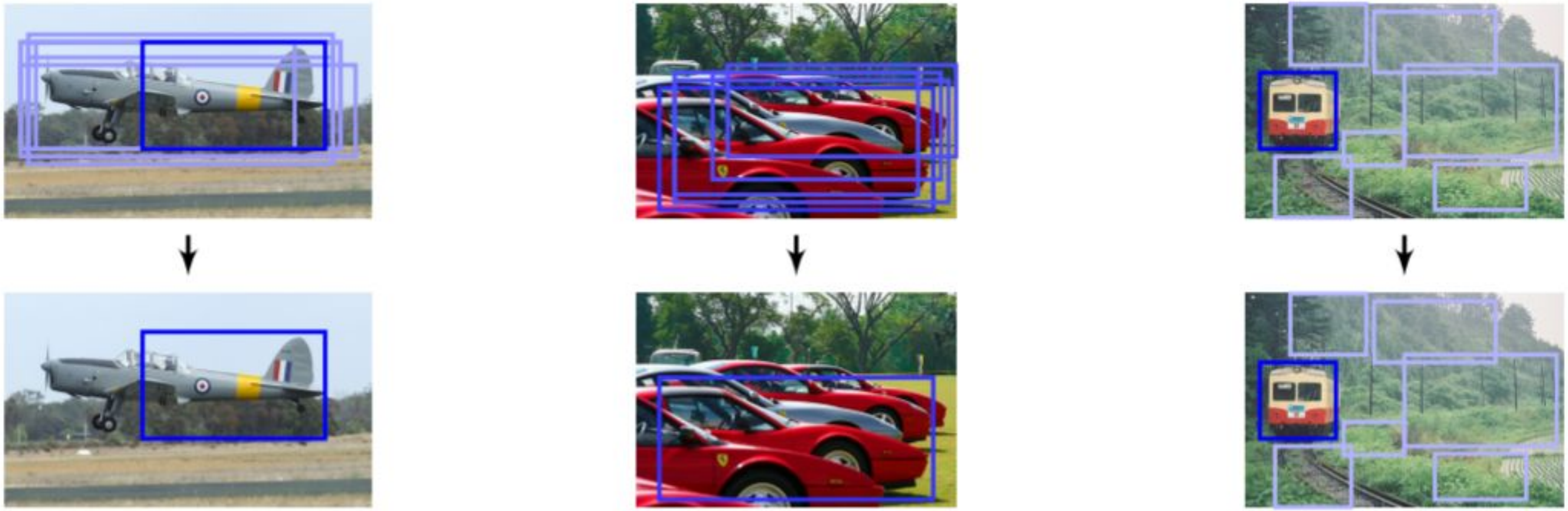
The blue line is aligned with the edge, so we don't want to apply NMS along this direction

Skoltech
Skolkovo Institute of Science and Technology

Canny edge detector (step-by-step)

Non-Maximum Suppression - a side note

- NMS (the same idea, but different algorithms) is very useful for object detection



Source: Non-Maximum Suppression for Object Detection by Passing Messages between Windows

Canny edge detector (step-by-step)

Non-Maximum Suppression

Gradient Magnitude > Non-Maximum Suppression

Canny edge detector (step-by-step)

Non-Maximum Suppression

Gradient Magnitude › Non-Maximum Suppression

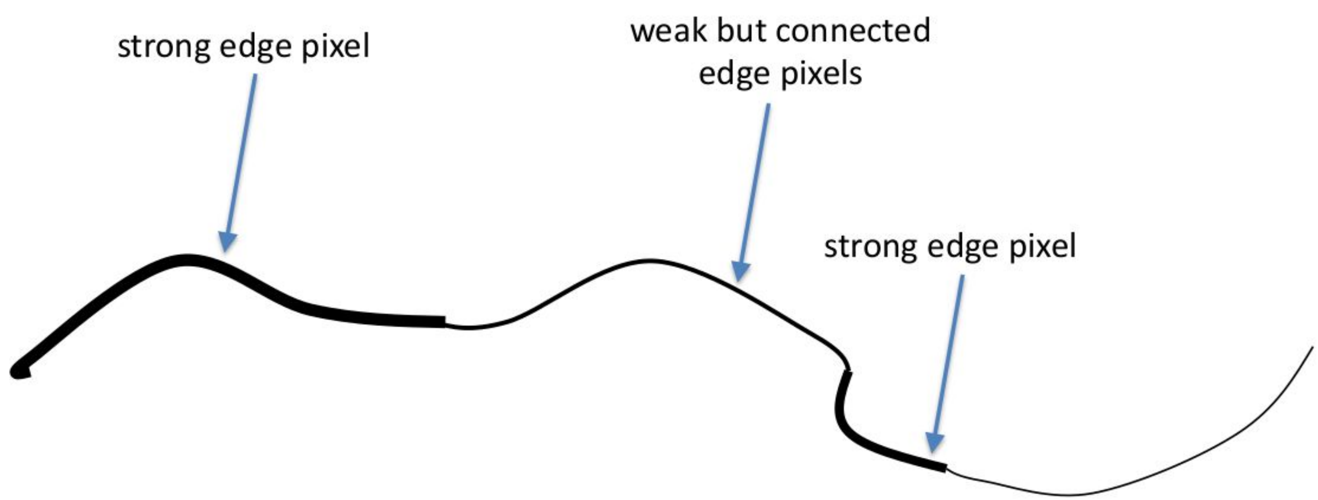
Canny edge detector (step-by-step)

Hysteresis thresholding and connectivity analysis

- Define two thresholds: *Low* and *High*
  - pixel < *Low* :: **not an edge**
  - pixel > *High* :: **strong edge**
  - *Low* < pixel < *High* :: **weak edge**

Canny edge detector (step-by-step)

Hysteresis thresholding and connectivity analysis



- Re-declare **weak edge** as **strong edge** if it is in the same **connected component** with a strong edge
- Re-declare **weak edge** as **not an edge** if it has no strong edges in the connected component

Skoltech
Skolkovo Institute of Science and Technology

Canny edge detector (step-by-step)

Non-Maximum Suppression

Non-Maximum Suppression

Hysteresis thresholding

Canny edge detector

Hysteresis thresholding

Skoltech
Skolkovo Institute of Science and Technology

Edge detection for a noisy image

Practical exercise: Sobel filter (in jupyter).

Skoltech
Skolkovo Institute of Science and Technology