Компьютерное зрение

Лекция 1.
Базовая теория. Свертки изображений. Детекция границ.

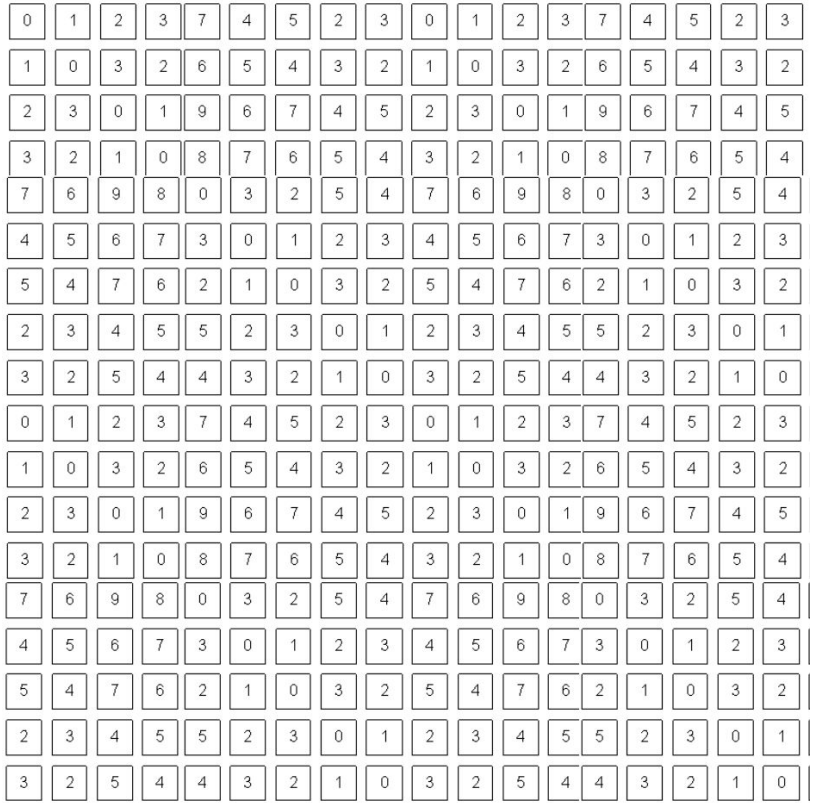
28.05.2020

Руслан Рахимов

Что такое компьютерное зрение?

Что такое компьютерное зрение?

Что такое компьютерное зрение?

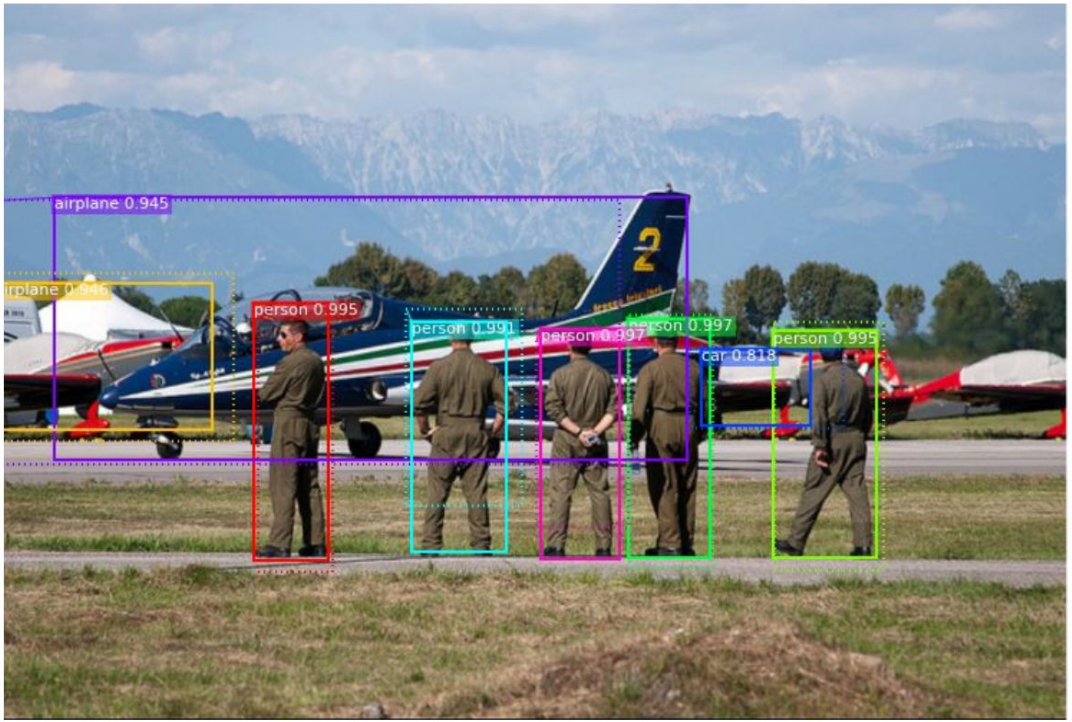


Задача компьютерного зрения дать компьютерам восприятие на уровне человека (или лучше)

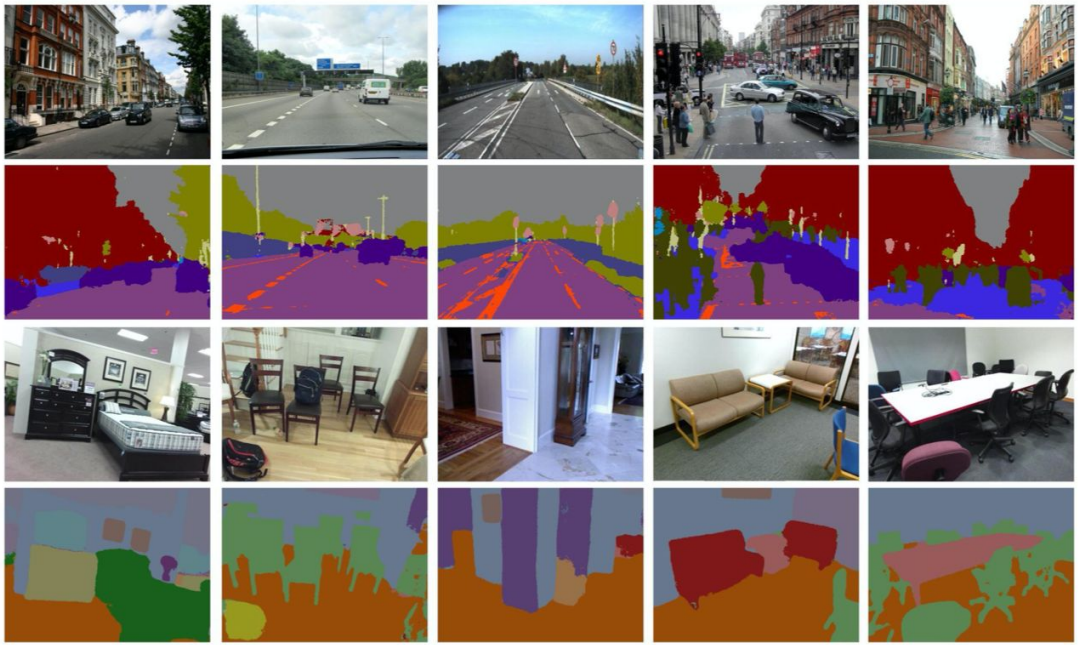
Примеры задач CV

Классификация изображений

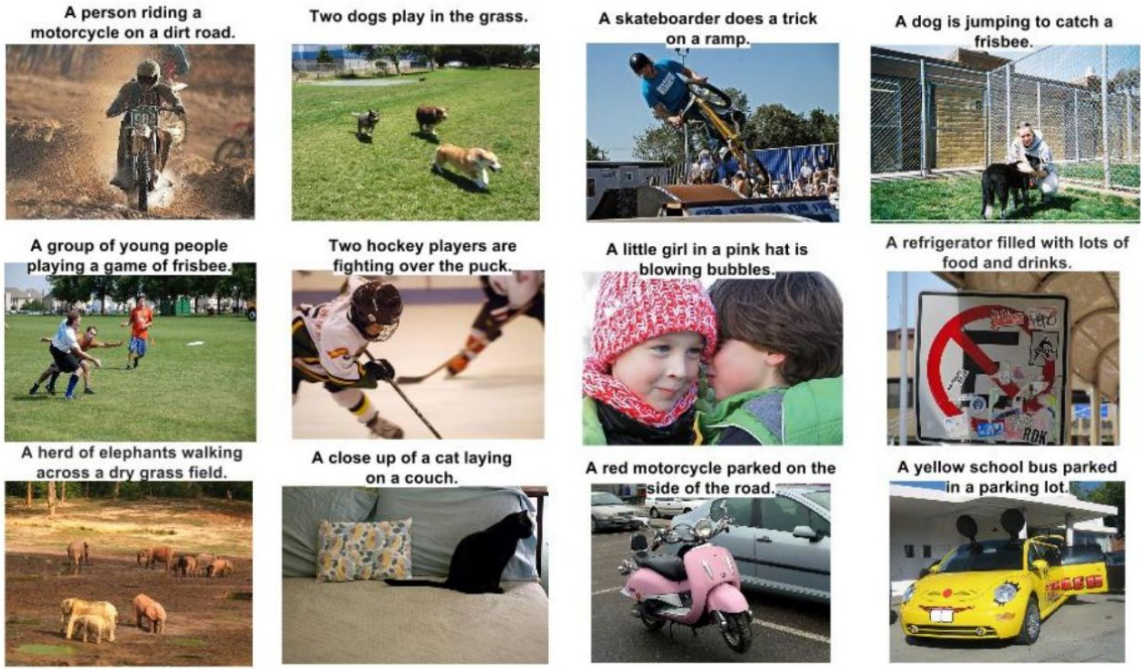
Детекция объектов



Семантическая сегментация изображения



Аннотация изображений



В программе курса:

— Базовая теория: форматы, преобразование изображений с помощью фильтров, знакомство с библиотекой OpenCV.

— Извлечение признаков, PCA разложение, гистограммы цвета и градиента, особые точки.

— Сегментация изображений, детекция объектов на изображении - классические подходы.

— Сверточные сети для обработки изображений. Основные блоки. Знакомство с фреймворком PyTorch.

— Архитектуры для классификации изображений. Аугментация данных, тюнинг обученных сетей.

— Проблемы при обучении нейронных сетей и как их решать.

— Сегментация изображений, детекция объектов на изображении с помощью нейросетей.

— Распознание лиц. Две постановки задачи - классификация и верификация. Задача Metric Learning.

— Латентное представление изображений. Autoencoders и Variational Autoencoders.

— Генеративно-состязательные модели.

— Рекуррентные сети. Архитектура RNN, LSTM, GRU. Генерация описания по изображению.

Что такое изображение?

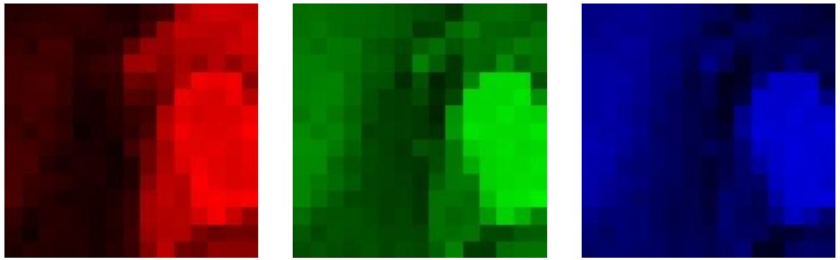
A (color) image is a 3D tensor of numbers.

Что такое изображение?

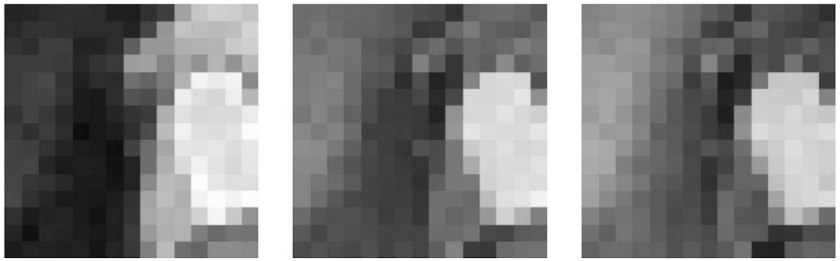
color image patch

How many bits are the intensity values?

red green blue



colorized for visualization



actual intensity values per channel

Each channel is a 2D array of numbers.

Что такое изображение?

F

Filtering

$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$

G

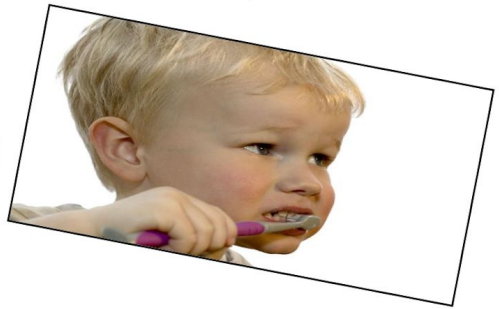
changes *range* of image function

F

Warping

$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$

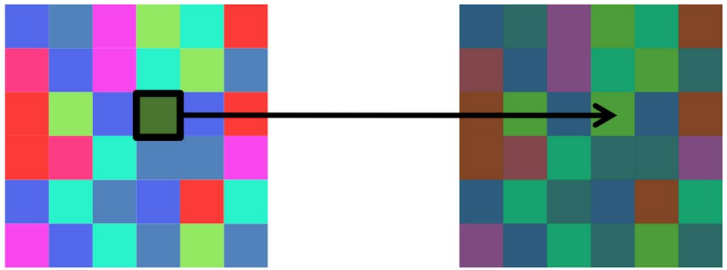
G



changes *domain* of image function

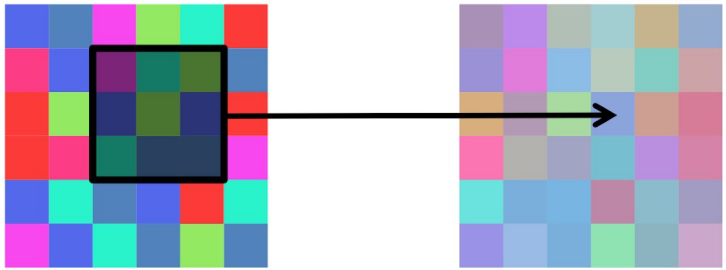
Какие преобразования можем проводить?

Point Operation



point processing

Neighborhood Operation



“filtering”

Примеры point processing

original

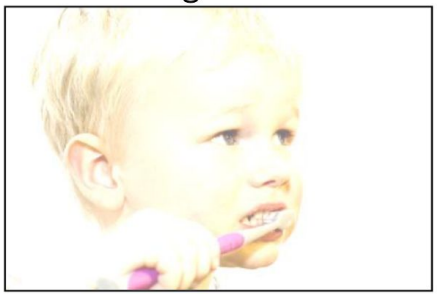
darken

lower contrast

non-linear lower contrast

invert

lighten



raise contrast

non-linear raise contrast

Примеры point processing

original

x

darken

$x - 128$

lower contrast

$\frac{x}{2}$

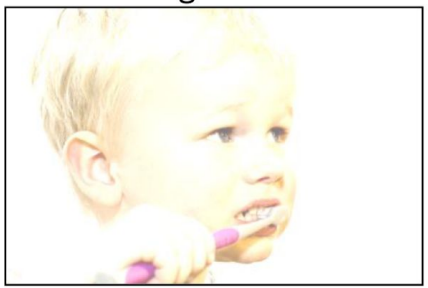
non-linear lower contrast

$\left(\frac{x}{255}\right)^{1/3} \times 255$

invert

$255 - x$

lighten



$x + 128$

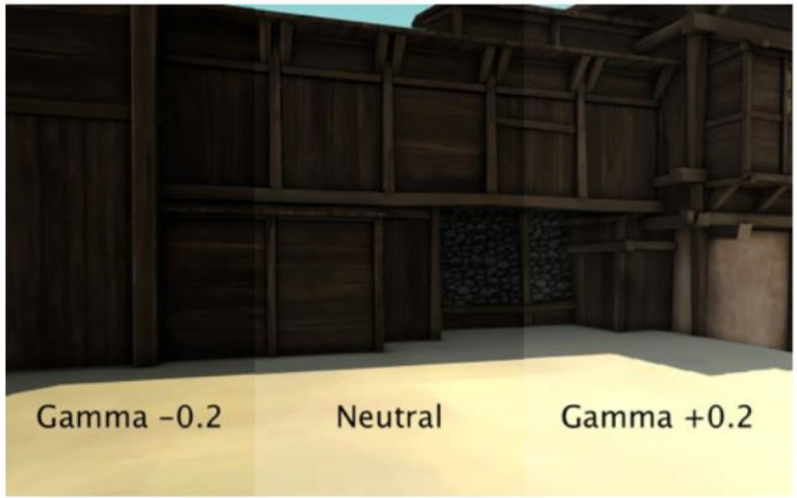
raise contrast

$x \times 2$

non-linear raise contrast

$\left(\frac{x}{255}\right)^{2} \times 255$

И много других преобразований…



Linear shift-invariant image filtering

Linear shift invariant filtering

- заменим каждый пиксель линейной комбинацией соседей (и самого себя)
- комбинация задается ядром
- одинаковое ядро применятся ко всем пикселям изображения

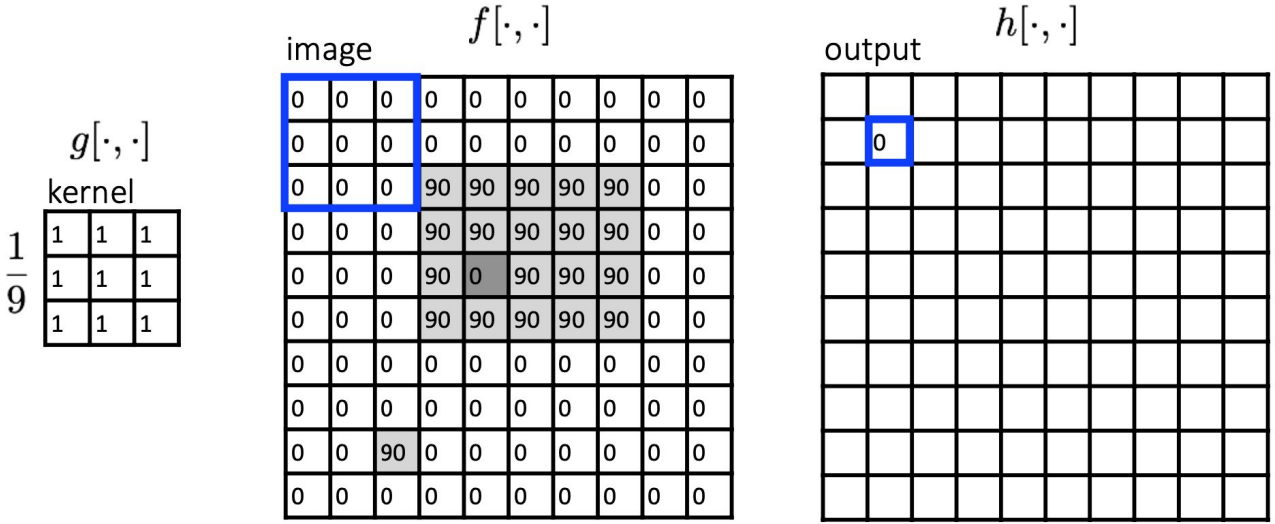
Example: box filter

Ядро $g[\cdot,\cdot] = \frac{1}{9}$

1	1	1
1	1	1
1	1	1

- заменяет пиксель локальным средним
- обладает сглаживающим (blurring) эффектов

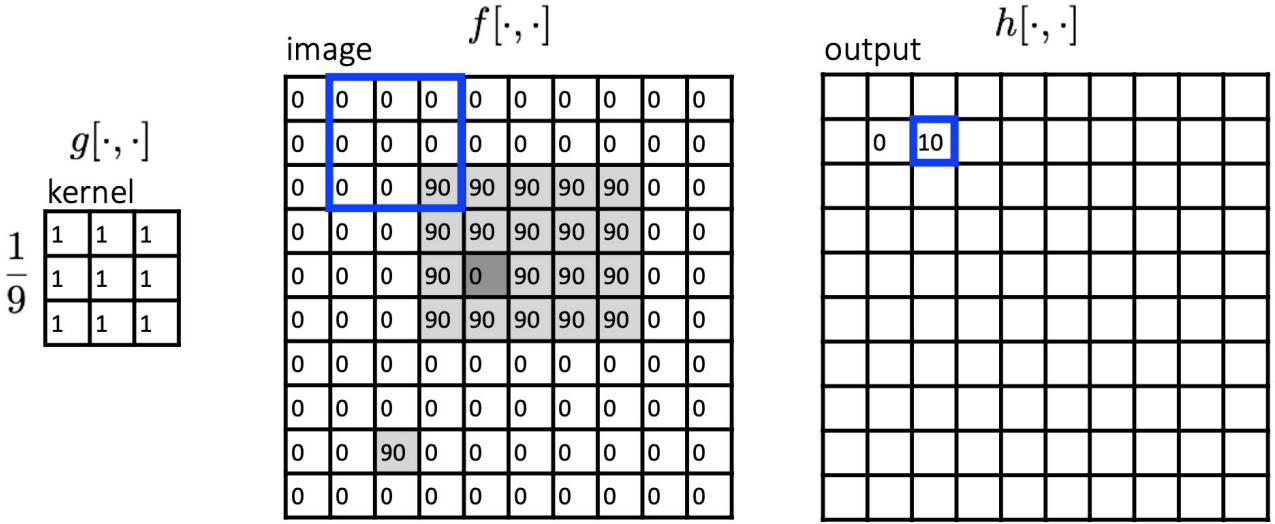
Применим box filter



$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

output — filter — image (signal)

Применим box filter



$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

output filter image (signal)

Применим box filter

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

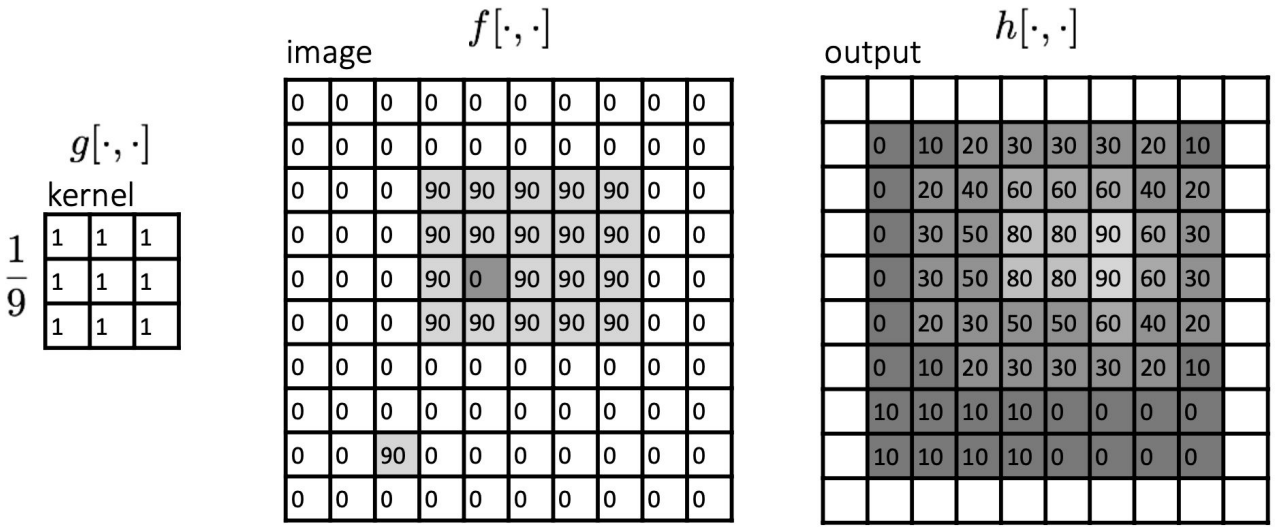
output

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

output — filter — image (signal)

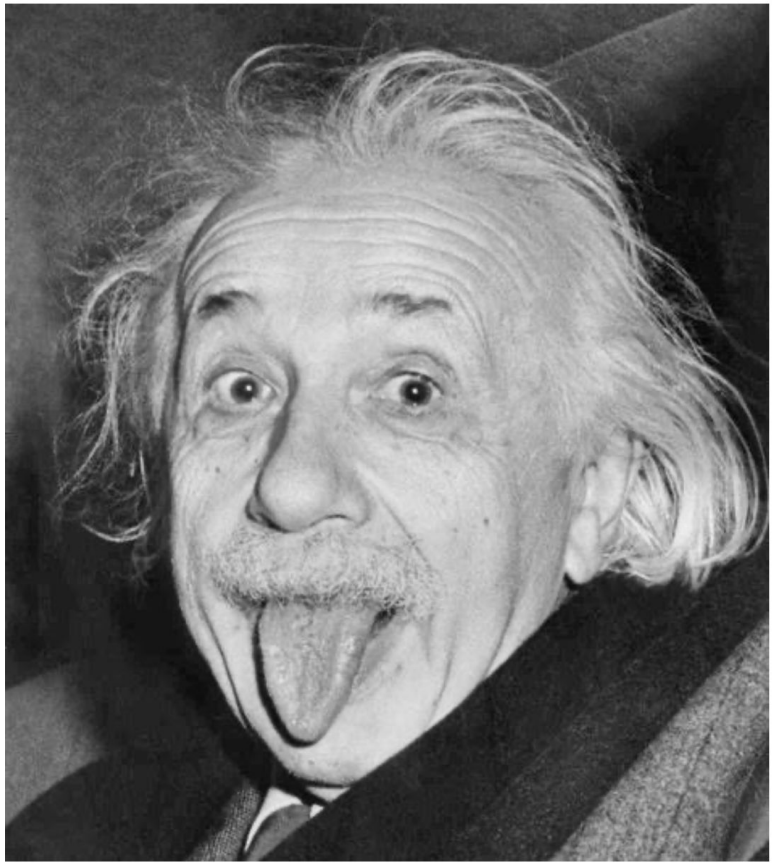
Применим box filter

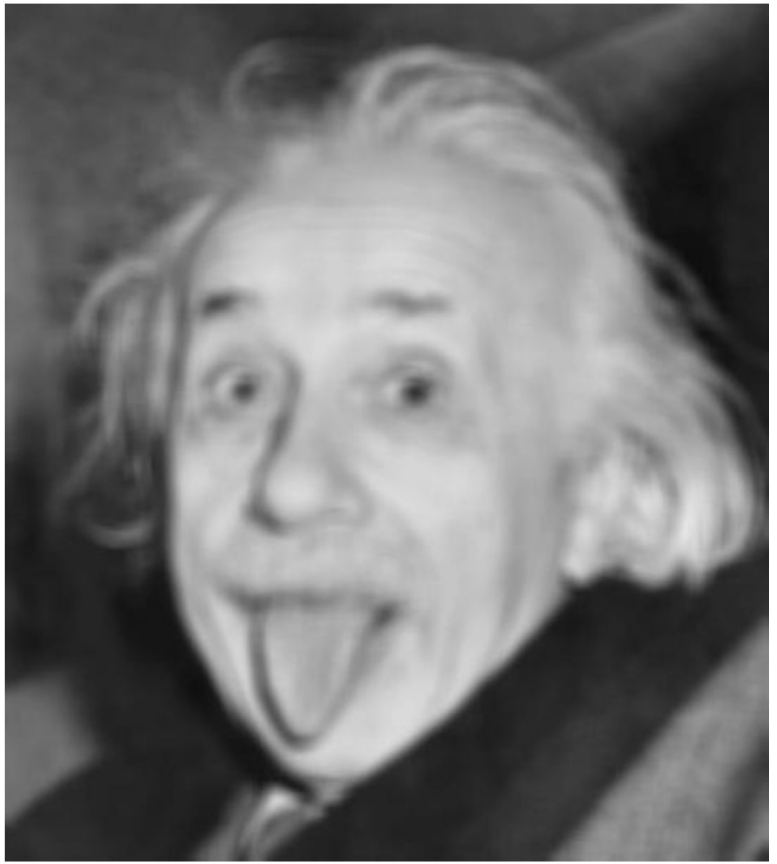


$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

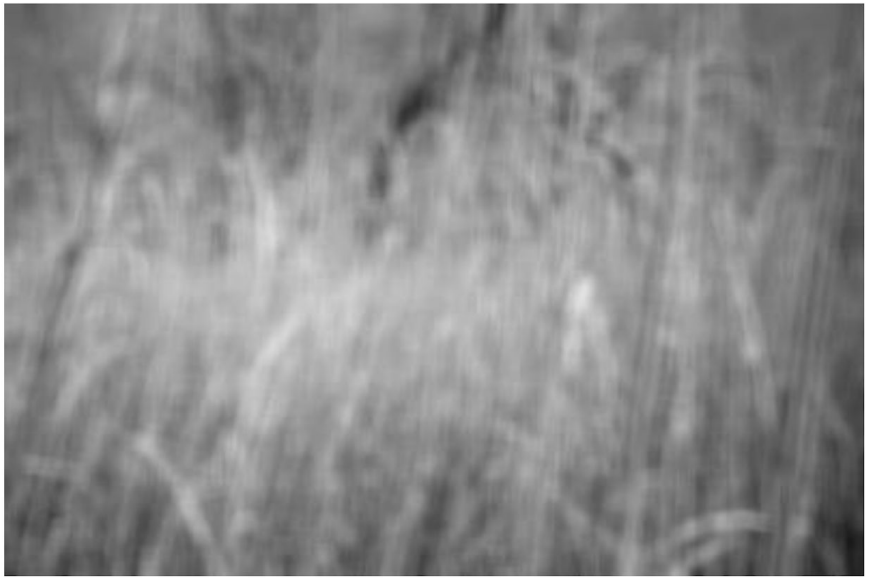
output filter image (signal)

Реальный пример





Реальный пример



Реальный пример

Convolution
(свертка)

Свертка 1D непрерывного сигнала

Definition of filtering as convolution:

notice the flip

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-y)dy$$

filtered signal

filter

input signal

Свертка 1D непрерывного сигнала

Definition of filtering as convolution:

notice the flip

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-y)dy$$

filtered signal

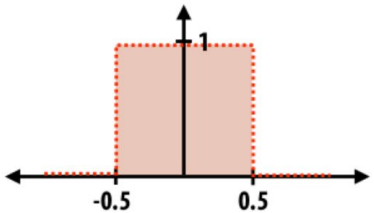
filter

input signal

Consider the box filter example:

1D continuous box filter

$$f(x) = \begin{cases} 1 & |x| \leq 0.5 \\ 0 & otherwise \end{cases}$$



filtering output is a blurred version of g

$$(f * g)(x) = \int_{-0.5}^{0.5} g(x-y)dy$$

Свертка 2D дискретного сигнала

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image

filter

input image

Свертка 2D дискретного сигнала

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image — filter — input image

If the filter $f(i, j)$ is non-zero only within $-1 \leq i, j \leq 1$, then

$$(f * g)(x, y) = \sum_{i,j=-1}^{1} f(i, j) I(x - i, y - j)$$

Свертка vs Корреляция

Definition of discrete 2D convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

Definition of discrete 2D correlation:

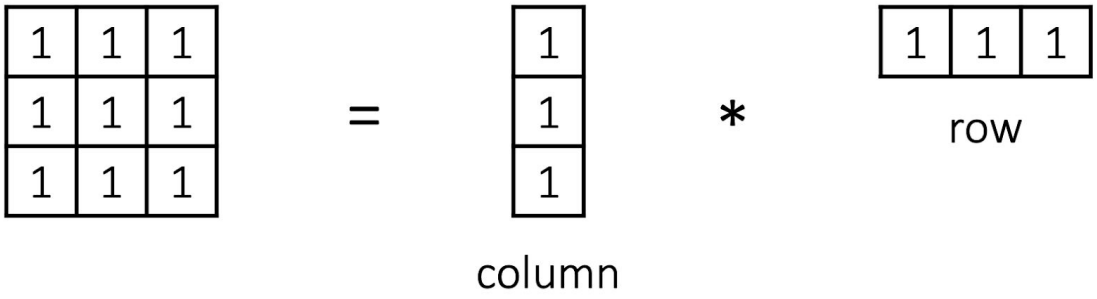
notice the lack of a flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x + i, y + j)$$

Сепарабельные фильтры

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

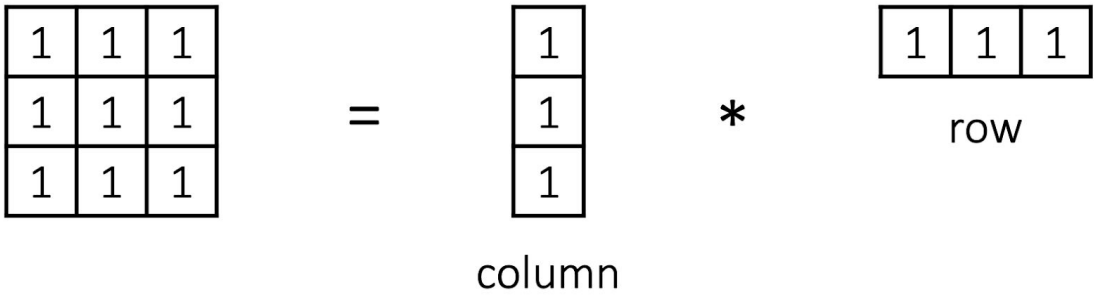
example:
box filter



Сепарабельные фильтры

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

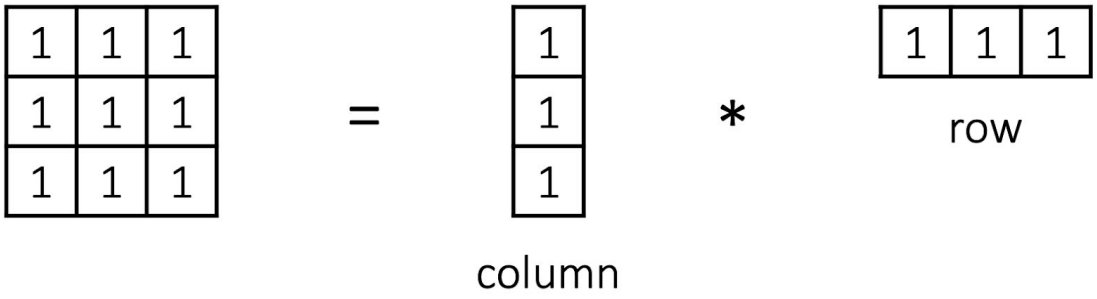
example:
box filter



Сепарабельные фильтры

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter



2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has M x M pixels and the filter kernel has size N x N:

- What is the cost of convolution with a non-separable filter?
- What is the cost of convolution with a separable filter?

Сепарабельные фильтры

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

=

1
1
1

column

*

1	1	1

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has M x M pixels and the filter kernel has size N x N:

- What is the cost of convolution with a non-separable filter? → $M^2 \times N^2$
- What is the cost of convolution with a separable filter? → $2 \times N \times M^2$

Больше фильтров...

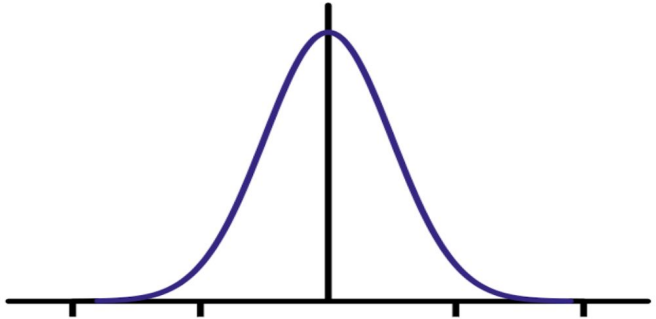
original

3x3 box filter

Гауссовский фильтр

- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$



- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?

Гауссовский фильтр

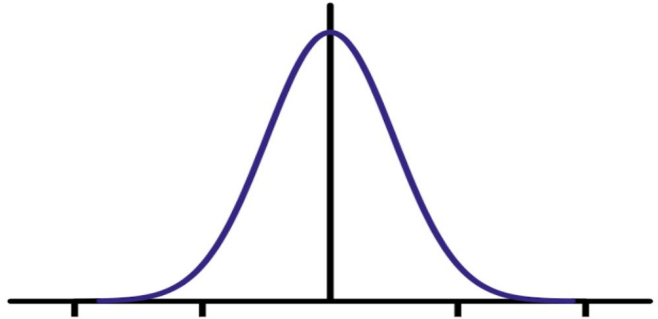
- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?

- usually at 2-3σ

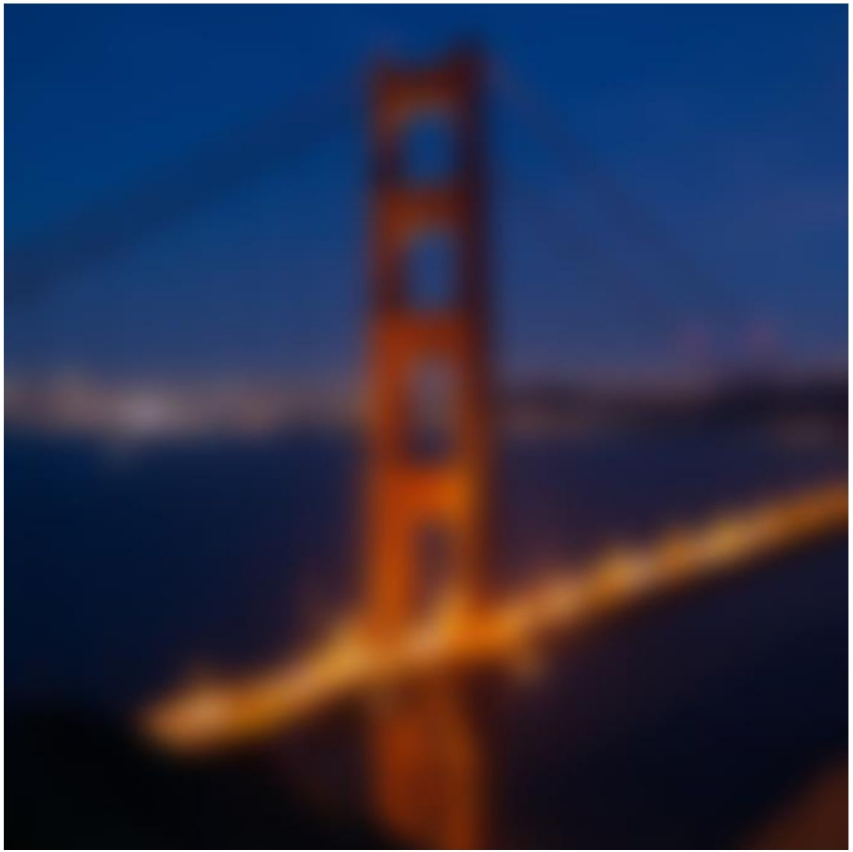


Is this a separable filter? Yes!

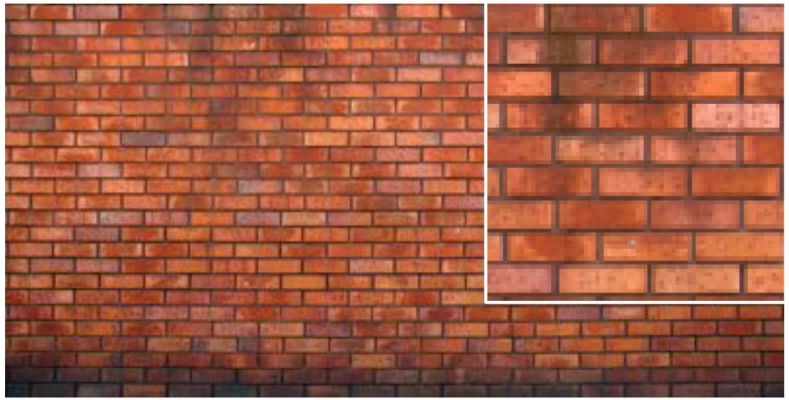
kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

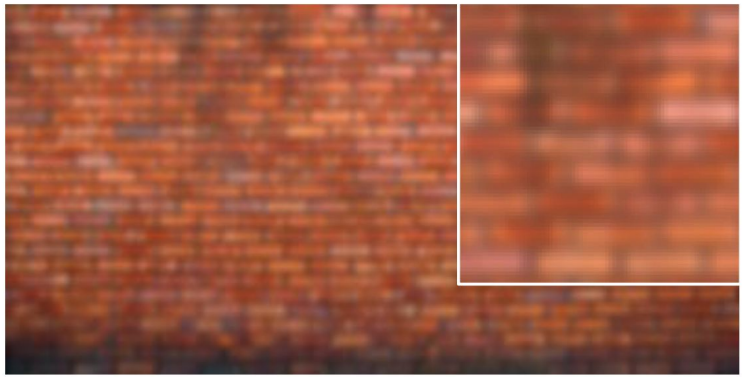
Гауссовский фильтр



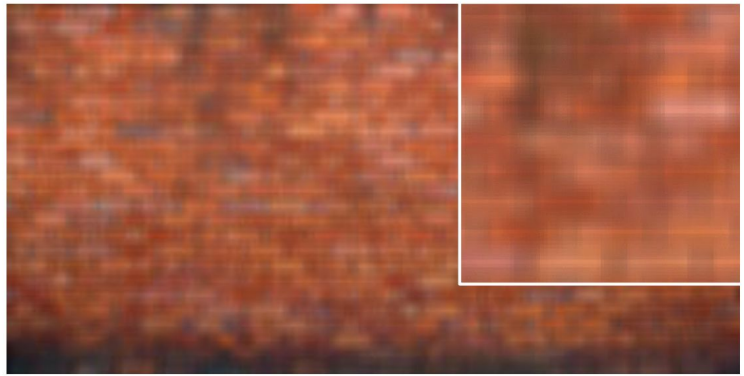
Гауссовский vs box фильтр



original

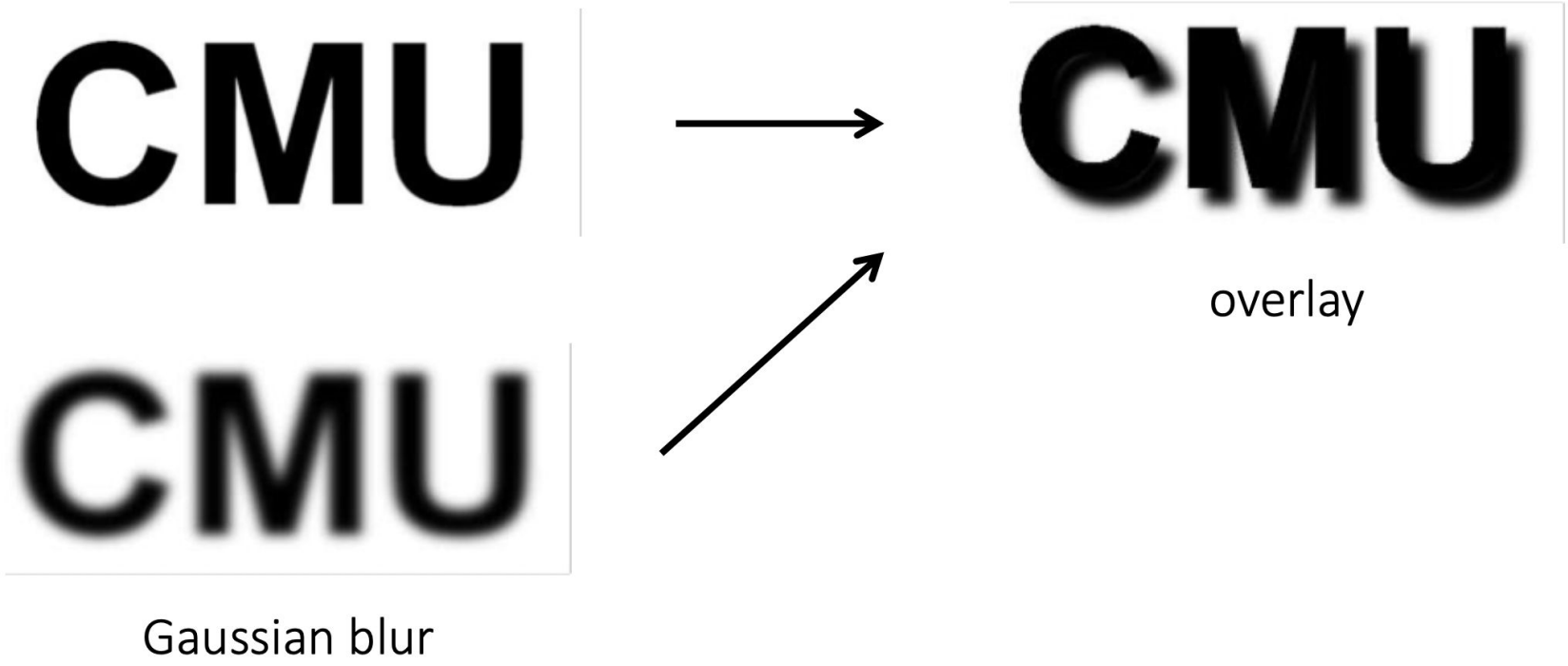


7x7 Gaussian



7x7 box

Создаем мягкую тень



Другие фильтры

input

filter

0	0	0
0	1	0
0	0	0

input

filter

0	0	0
0	0	1
0	0	0

Другие фильтры

input

filter

0	0	0
0	1	0
0	0	0

output

unchanged

input

filter

0	0	0
0	0	1
0	0	0

output

shift to left
by one

Другие фильтры

input

filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Другие фильтры

input

filter

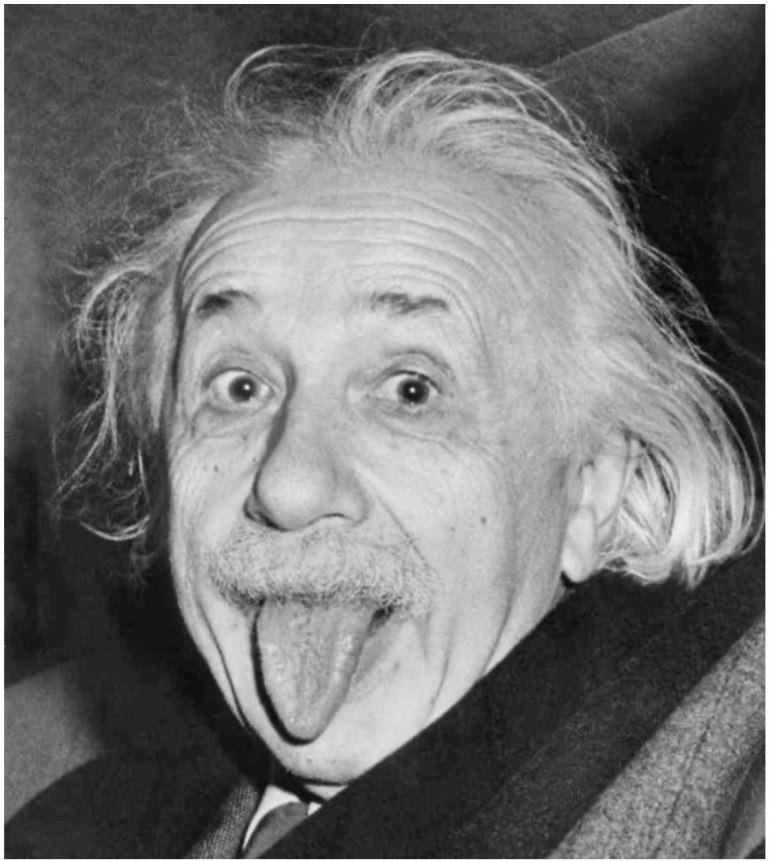
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

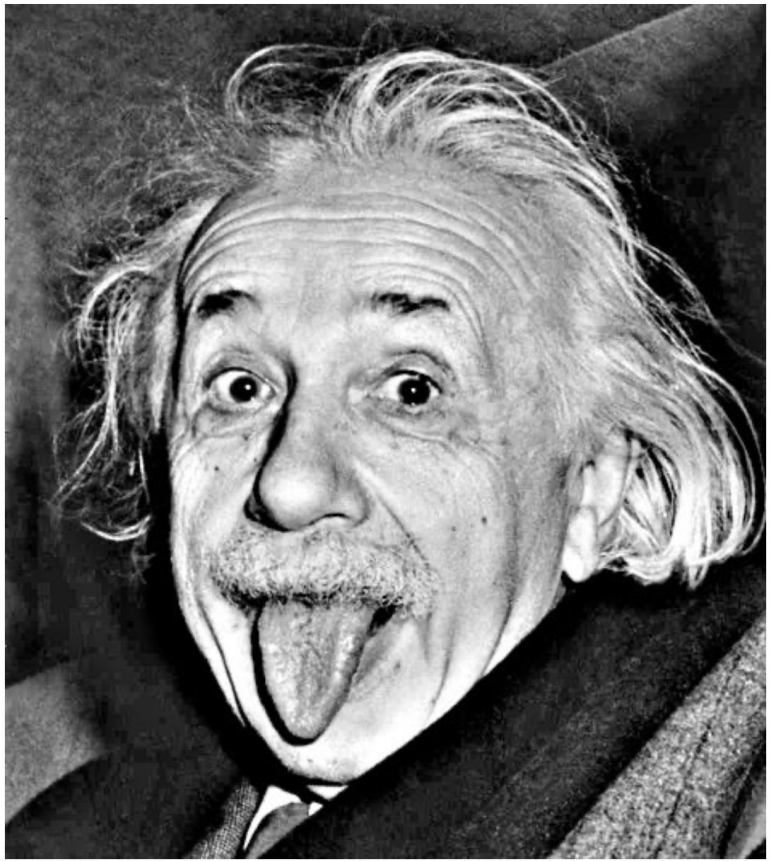
output

sharpening

- do nothing for flat areas
- stress intensity peaks

Примеры sharpening





Примеры sharpening

Padding

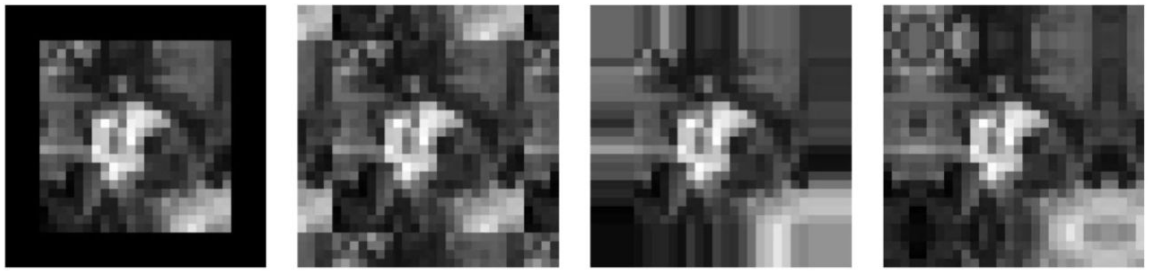
При свертке проблематично считать значения на краях изображения

Какие есть варианты?

Padding

При свертке проблематично считать значения на краях изображения

Какие есть варианты?



Ищем границы изображения

Edge detection

Задача: выделить резкие изменения (разрывы) в изображении

Интуитивно понятно, что большая часть информации об изображении содержится в краях:

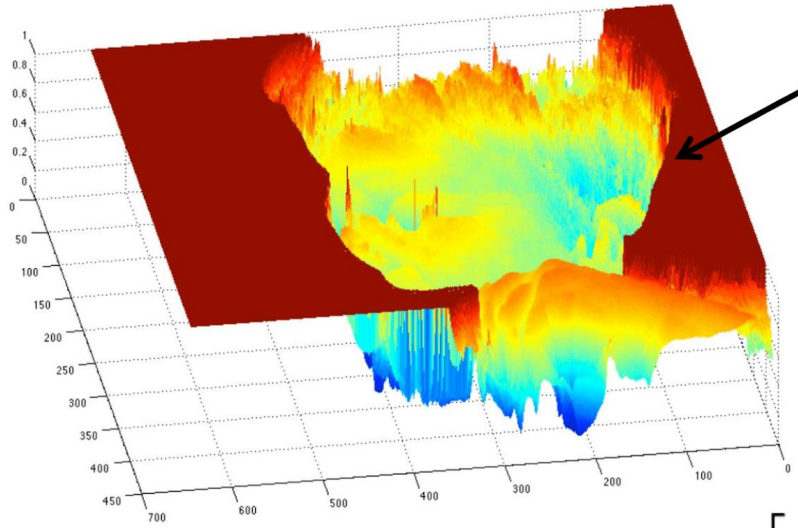
- Соответствует человеческому восприятию мира
- Компактнее, чем пиксели

Идеал: рисунок художника (но ему легче - он уже знает, какой объект рисует, есть априорная информация

Image edges (границы изображения)

grayscale image

$f(\boldsymbol{x})$



Very sharp discontinuities in intensity.

domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

Как находить края края?

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

How do you differentiate a discrete image (or any other discrete signal)?

Как находить края края?

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.

Метод конечных разностей

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Метод конечных разностей

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

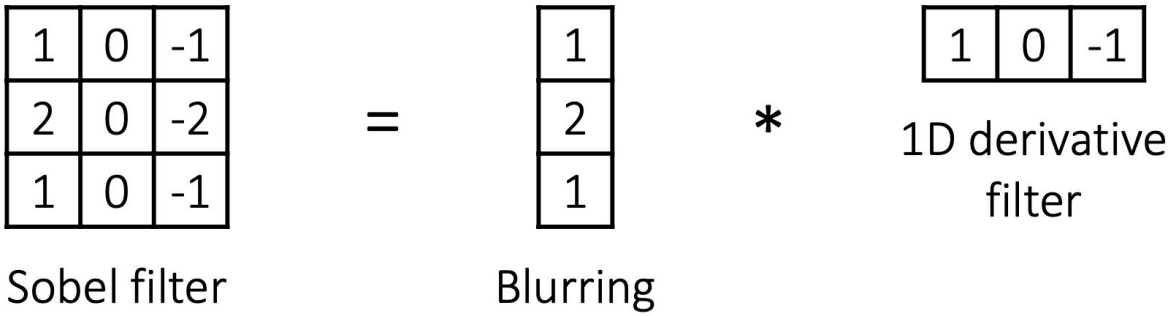
For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

1D derivative filter

1	0	-1

Фильтр Собеля



Фильтр Собеля

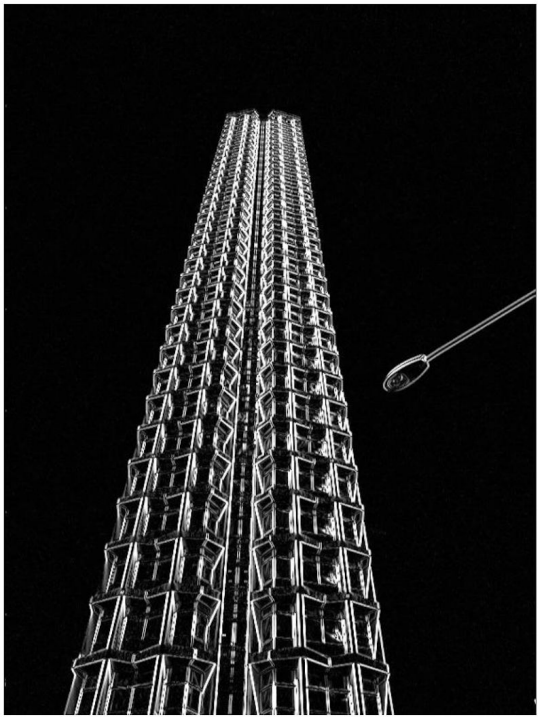
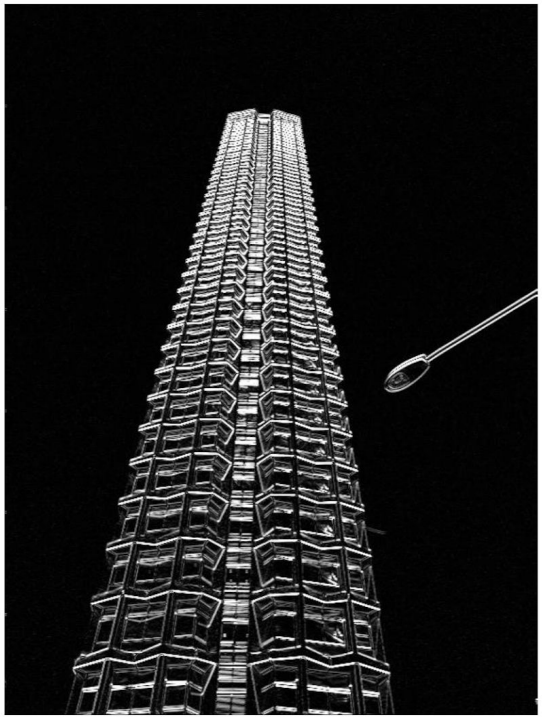
Horizontal Sober filter:

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

Vertical Sobel filter:

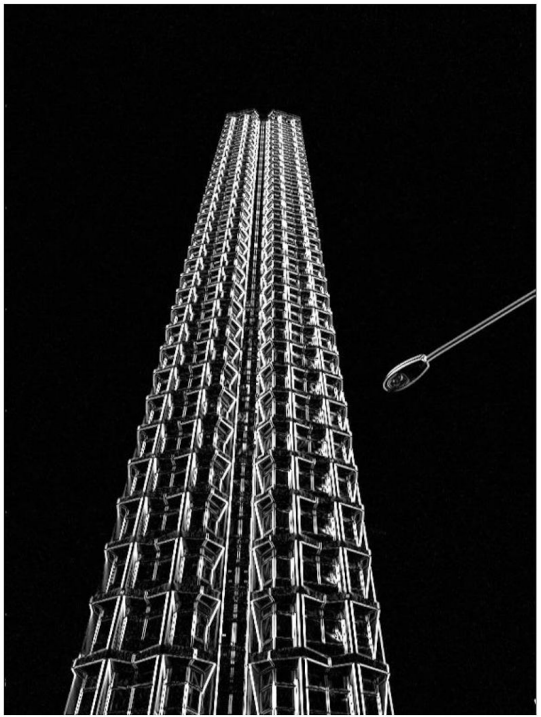
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Пример

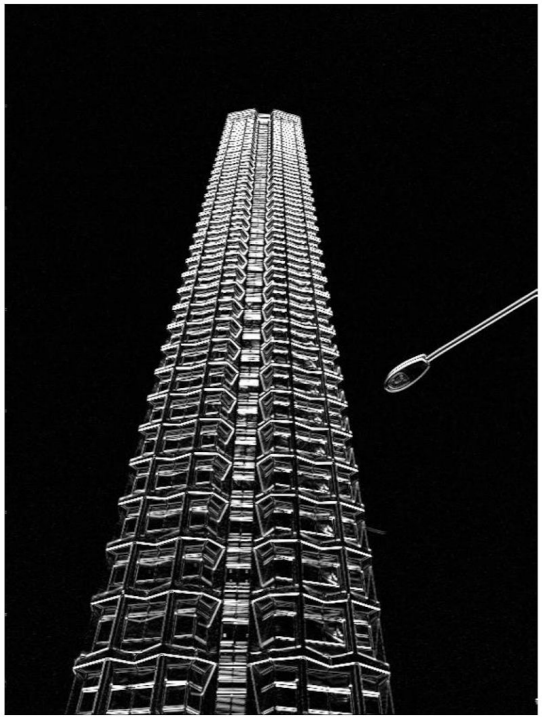




Пример

original

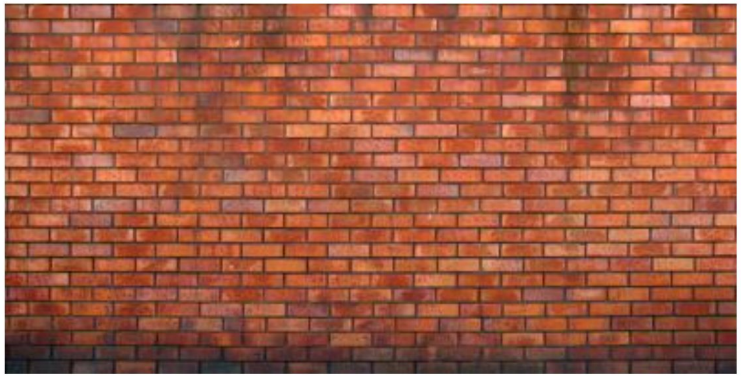


horizontal Sobel filter



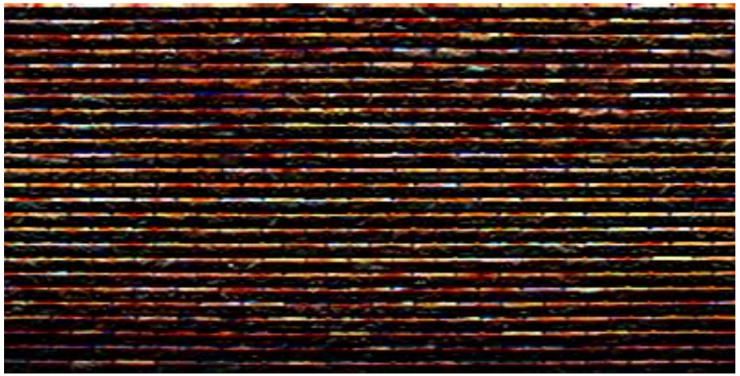
vertical Sobel filter

Пример



original

horizontal Sobel filter



vertical Sobel filter

Другие фильтры для взятия производной

Sobel

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

Градиенты изображения

1. Select your favorite derivative filters.

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Градиенты изображения

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

Градиенты изображения

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

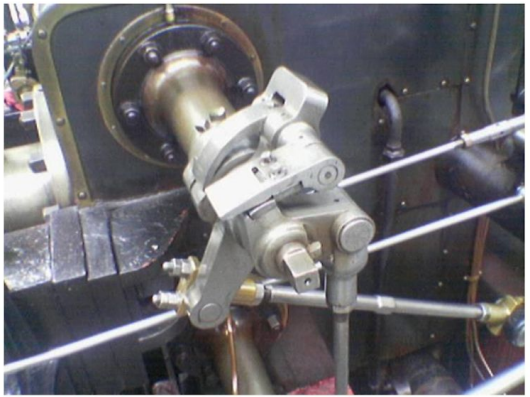
3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right) \qquad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

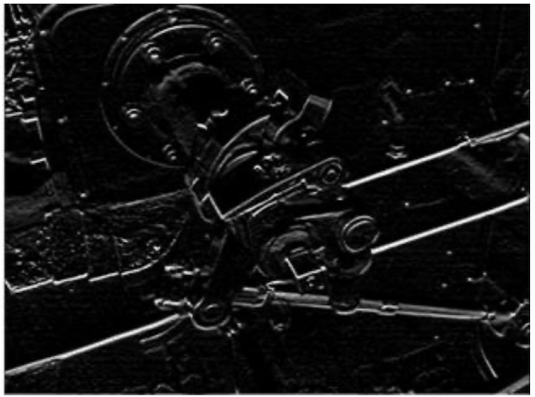
gradient direction amplitude

Пример

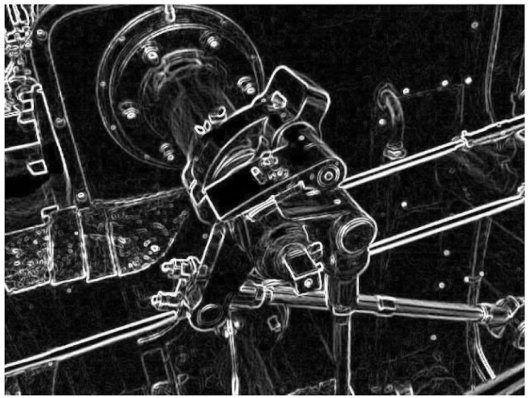
original



vertical derivative



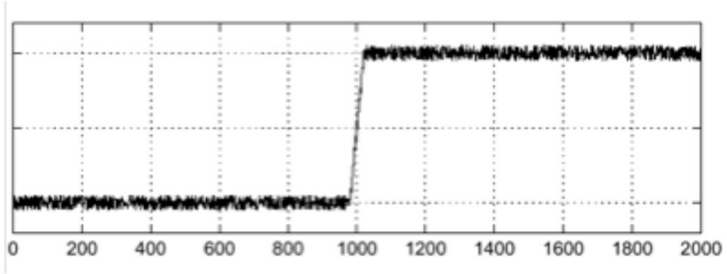
gradient amplitude



horizontal derivative

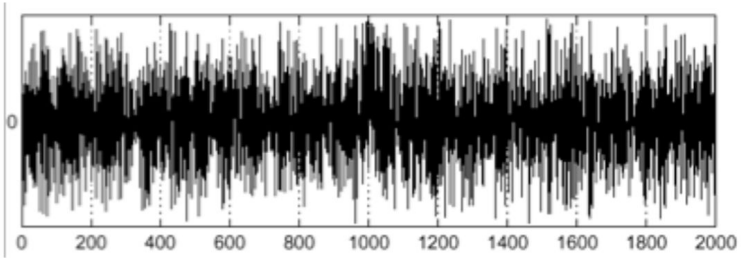
Ищем границы сигнала

intensity plot



Using a derivative filter:

derivative plot



What's the problem here?

Производные чувствительны к шуму

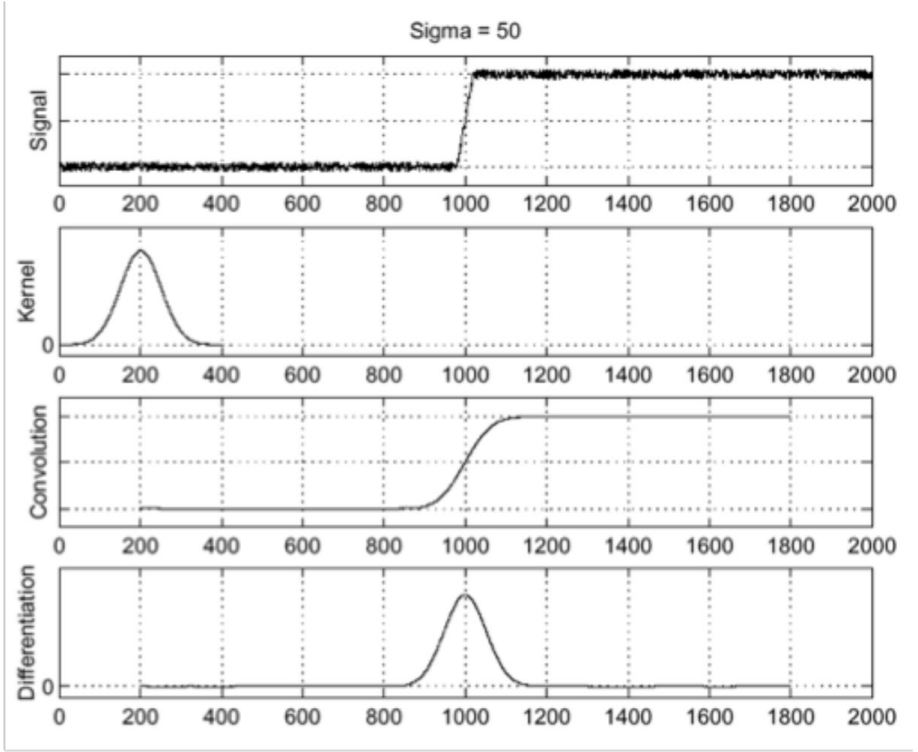
When using derivative filters, it is critical to blur first!

input

Gaussian

blurred

derivative of blurred



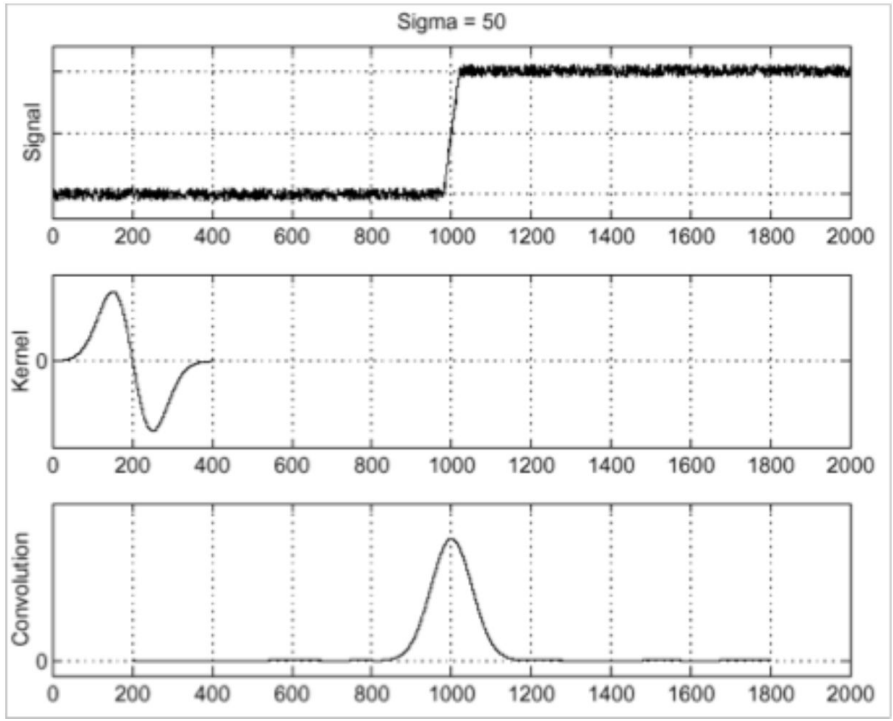
Derivative of Gaussian (DoG) filter

Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

input

derivative of Gaussian

output (same as before)



Фильтр Лапласа

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

→ 1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

→ Laplace filter

1	-2	1

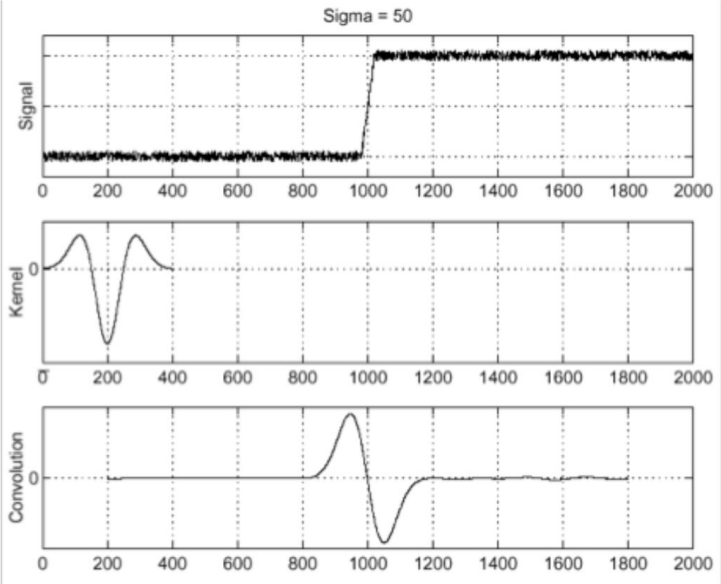
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



“zero crossings” at edges

LoG vs Laplace

Laplacian of Gaussian filtering

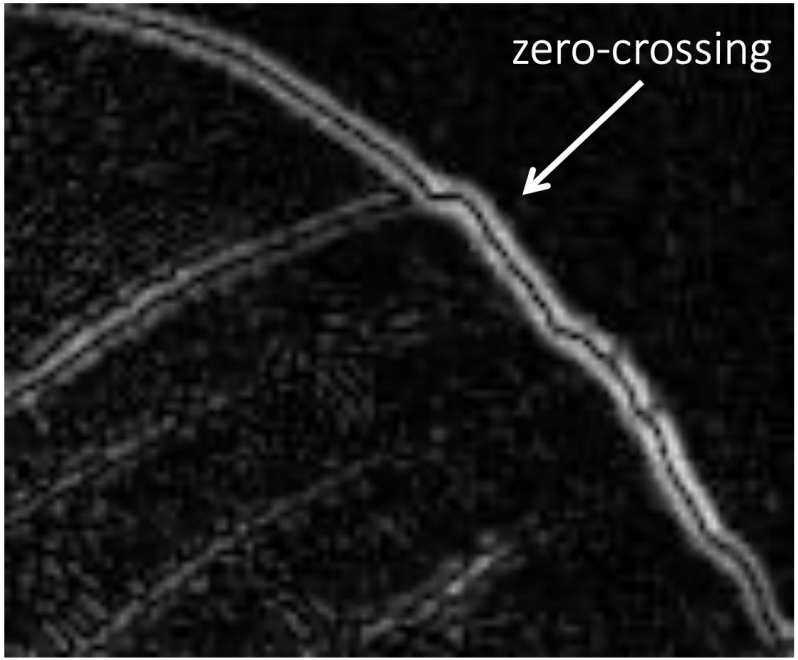
Laplace filtering

LoG vs DoG

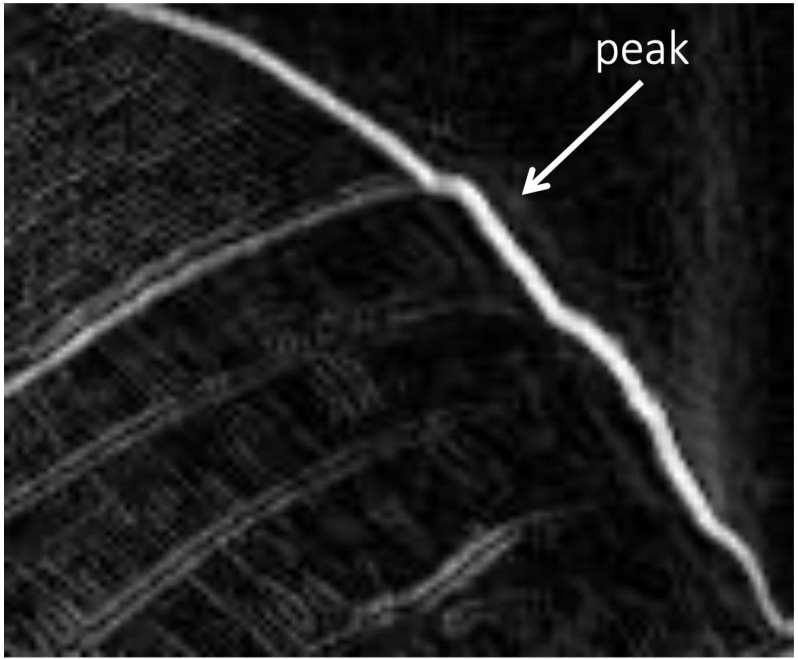
Laplacian of Gaussian filtering

Derivative of Gaussian filtering

LoG vs DoG



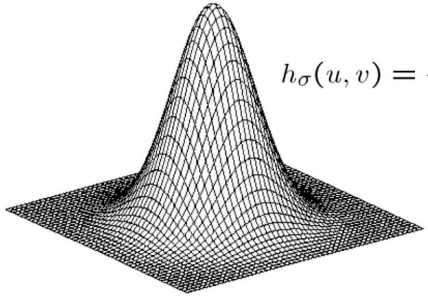
Laplacian of Gaussian filtering



Derivative of Gaussian filtering

Zero crossings are more accurate at localizing edges (but not very convenient).

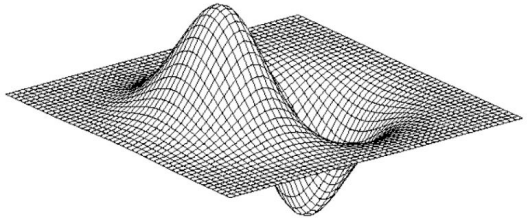
2D Гауссовские фильтры



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Gaussian

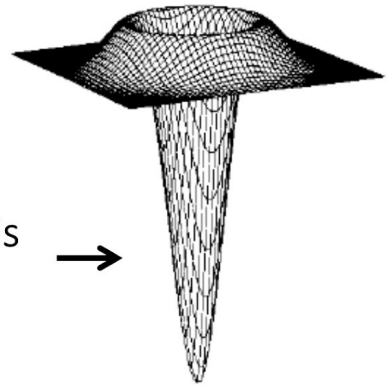
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Derivative of Gaussian

$$\nabla^2 h_\sigma(u,v)$$

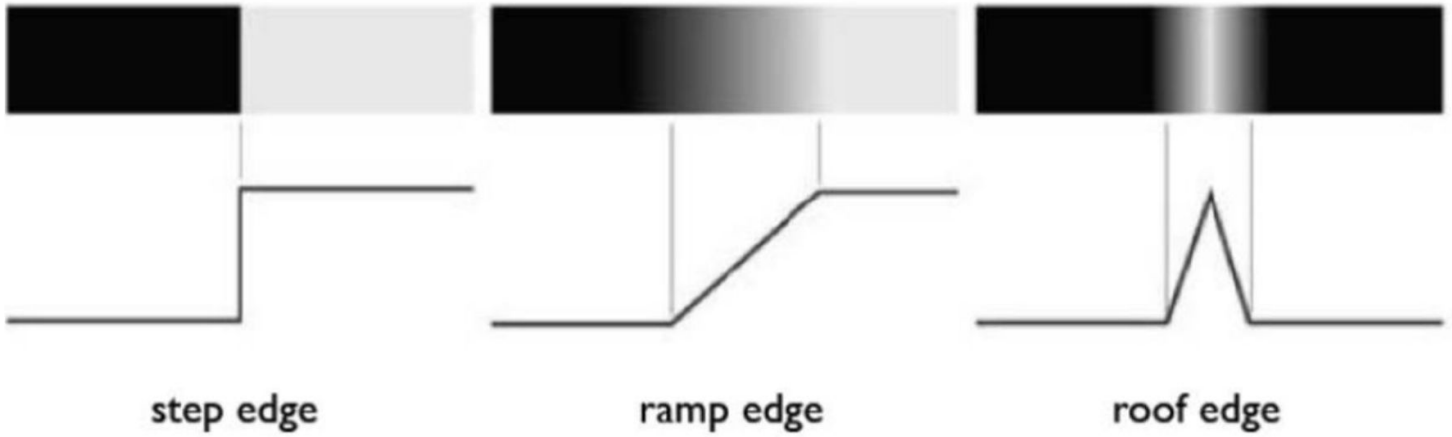
how does this relate to this lecture's cover picture? →



Laplacian of Gaussian

Детектор границ Кэнни

Минусы оператора Собеля



- Плохая локализация (в одной области множество пикселей принимаются за границу)
- Не все направления равнозначны из-за порогового значения - диагональные границы чаще пропускаются чем горизонтальные или вертикальные

Детектор границ Кэнни

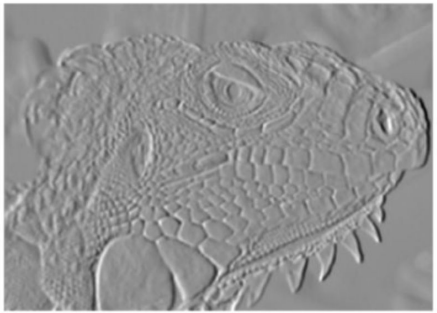
Наиболее широко используемой детектор границ в компьютерном зрении

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

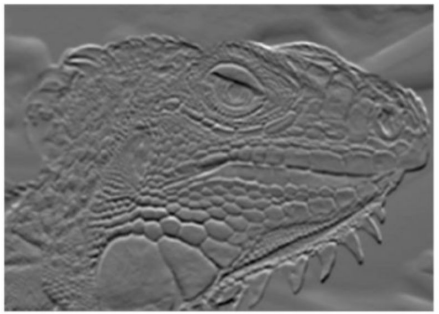
Детектор границ Кэнни

1. Применяем оператор Собеля - находим магнитуду и направление производной
2. Non-maximum suppression (делаем границы тонкими)
3. Гистерезис (Пороговое отсечение и связывание)
    - Два порога - низкий и высокий
    - Высокий позволяет начать инициализировать кривую, обозначающую границу, низкий позволяет продолжать ее

1. Применяем оператор Собеля



X-Derivative of Gaussian



Y-Derivative of Gaussian

Gradient Magnitude

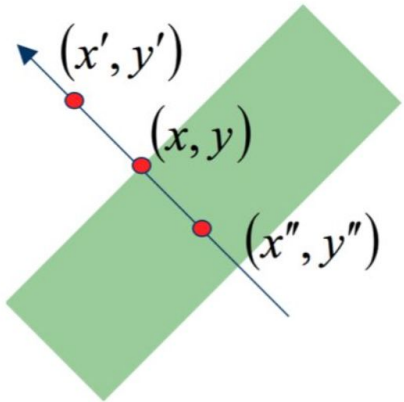
2. Non maximum suppression

- Края возникают, когда градиент достигает максимума
- Не будем рассматривать точку как границу, если ее градиент не максимален (даже если градиент проходит пороговое значение)
- Убираем пиксели как границы во всех направлениях, если их градиенты не максимальны

Позволяет добиться single response

2. Non maximum suppression

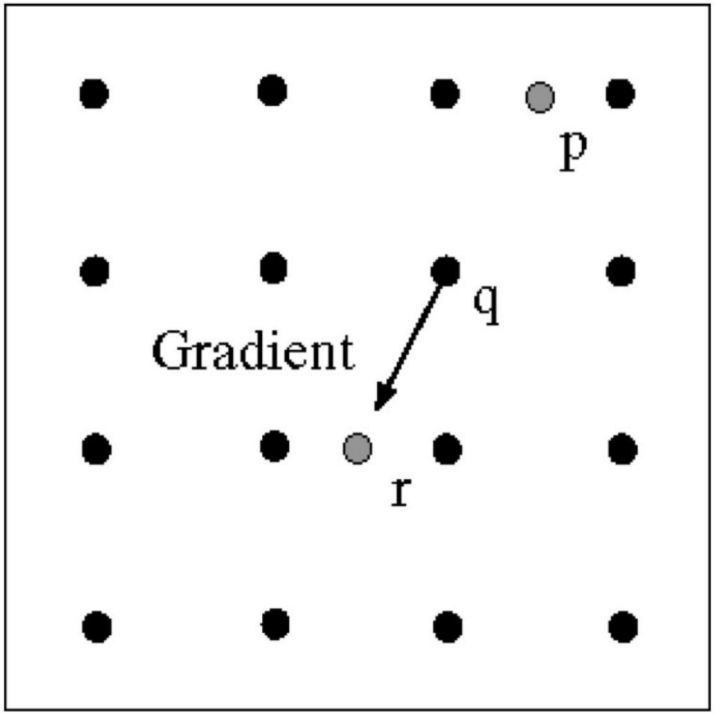
$|\nabla G|(x, y)$ это градиент в пикселе (x, y)



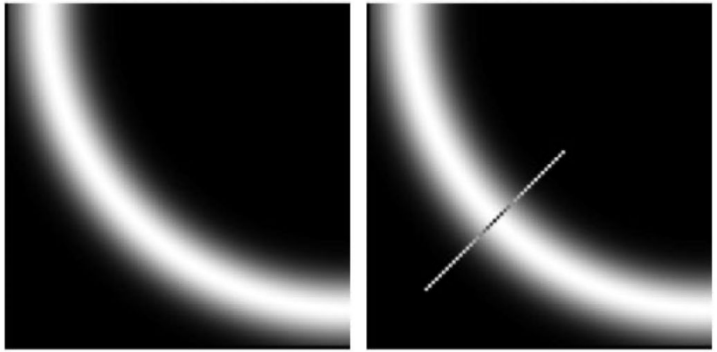
$$M(x,y) = \begin{cases} |\nabla G|(x,y) \text{ if } |\nabla G|(x,y) > |\nabla G|(x',y') \\ \quad\quad\quad\quad\quad\quad\quad\quad \&|\nabla G|(x,y) > |\nabla G|(x'',y'') \\ 0 \quad\quad\quad\quad\quad\quad\quad\quad \text{otherwise} \end{cases}$$

x' and **x**" are the neighbors of **x** along normal direction to an edge

2. Non maximum suppression



Интерполируем, чтобы получить значение градиента в p и r.

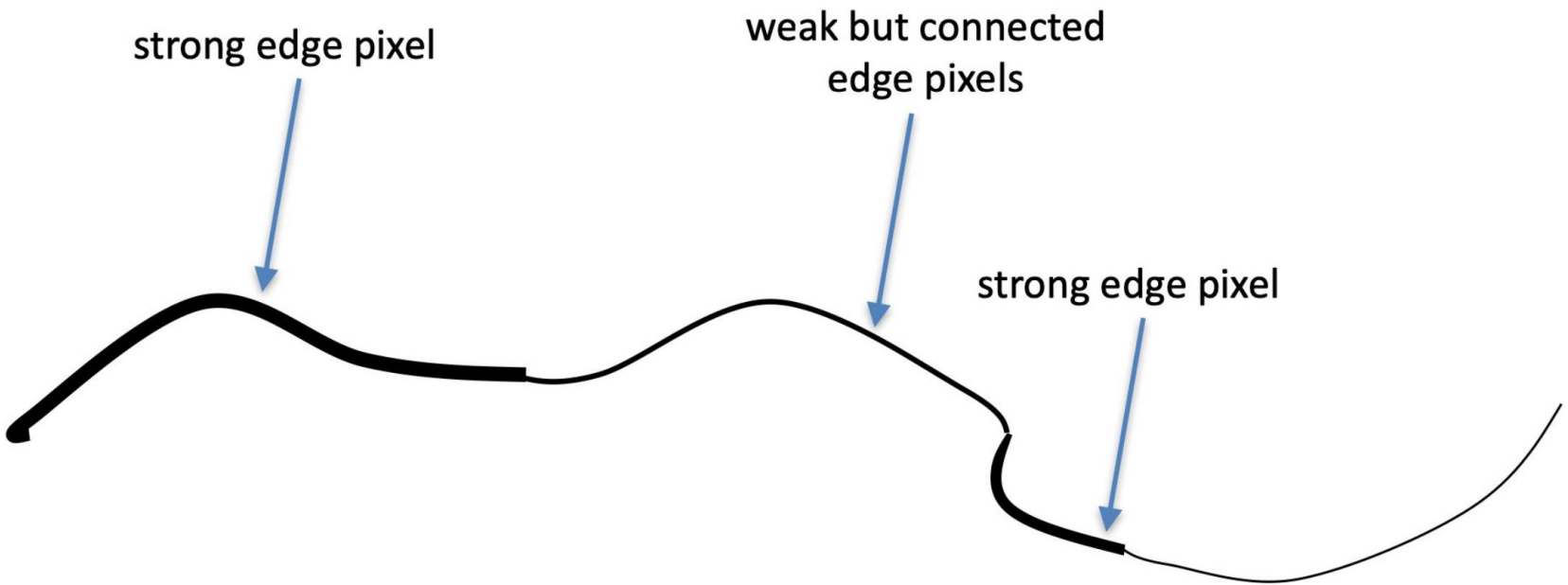


2. Non maximum suppression

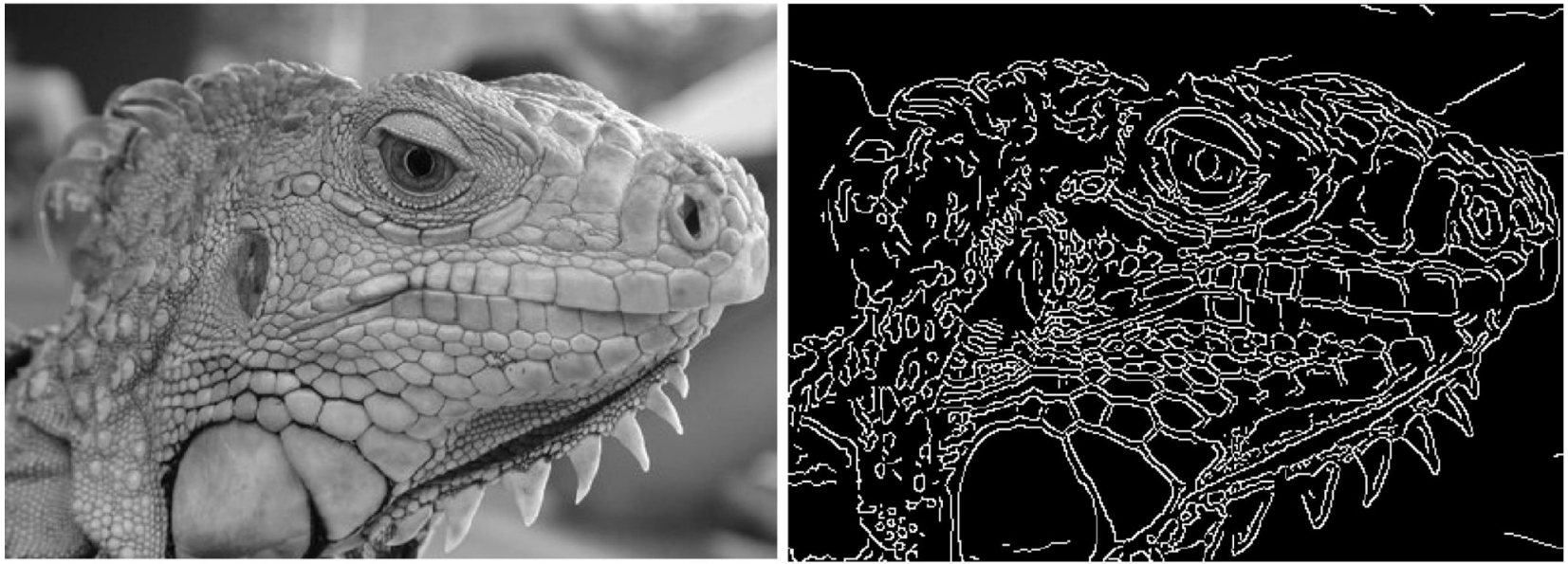
Before

After

Гистерезис


strong edge pixel
weak but connected
edge pixels
strong edge pixel

Результат



Влияние ширины фильтра Гаусса

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

Выбор сигмы зависит от того, что хочется получить:

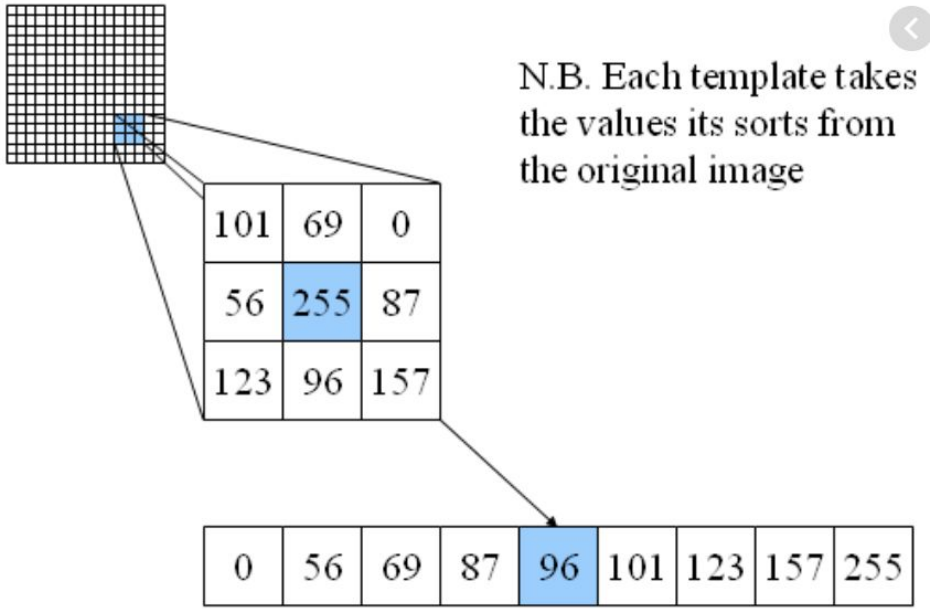
1. Большая сигма позволяет находить границы больших объектов
2. Маленькая сигма позволяет находить маленькие границы

Домашние задание №1

Имплементируем детектор границ Canny

Нелинейные фильтры

Медианный фильтр



Медианный фильтр

Билатеральный фильтр

Gaussian filter

Bilateral filter

Билатеральный фильтр

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

g(i,j) - результат применения фильтра в точке i,j
f(k,l) - значение исходного изображения в точке k,l
w(i,j,k,l) - вес, с которым учитывается значение f(k,l)

Билатеральный фильтр

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}\right)$$

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j)-f(k,l)\|^2}{2\sigma_r^2}\right)$$

d - фактор расстояния от точки i,j до k,l
r - “похожесть” значений исходного изображения в точке i,j и k,l
sigma - параметры фильтра

Билатеральный фильтр

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

w - учитывает как расстояние между точками (i,j) и (k,l), так и “похожесть” значений исходного изображения в точках (i,j) и (k,l)

Немного теории цвета

Цвет - часть ЭМ волны

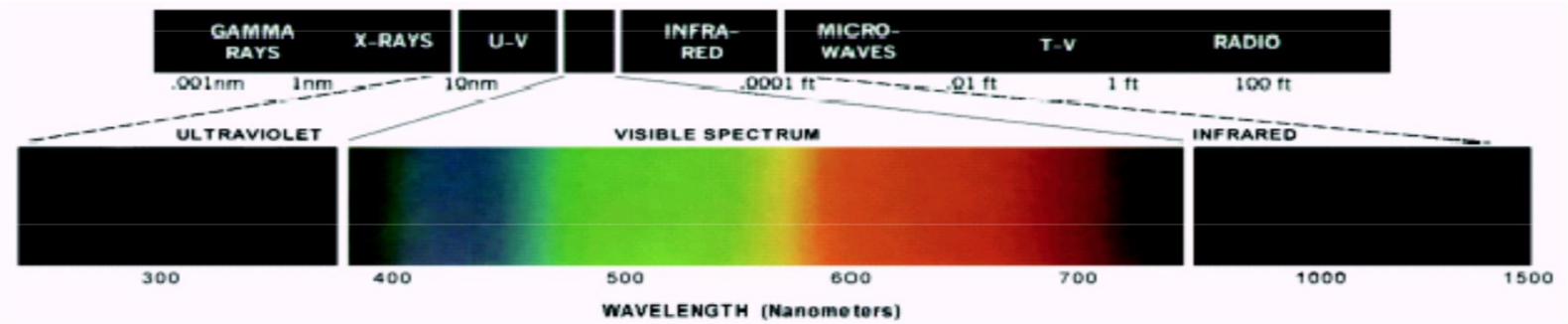


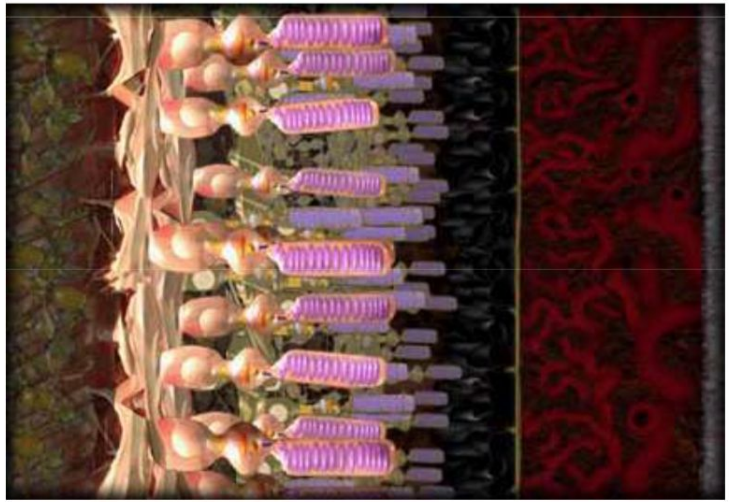
FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lamp Business Division.)

Illuminating and Reflecting Light

- Излучающие источники (первичный свет):
  - Излучают свет (солнце, монитор)
  - Воспринимаемый цвет зависит от частоты ЭМ волны
  - Аддитивное правило: R+G+B=White
- Отражающие источники (вторичный свет):
  - Отражают поступающий свет (краска, одежда)
  - Воспринимаемый цвет зависит от отраженного цвета (= emitted freq - absorbed freq)
  - Вычитающие правило: R+G+B=Black

Человеческое восприятие цвета

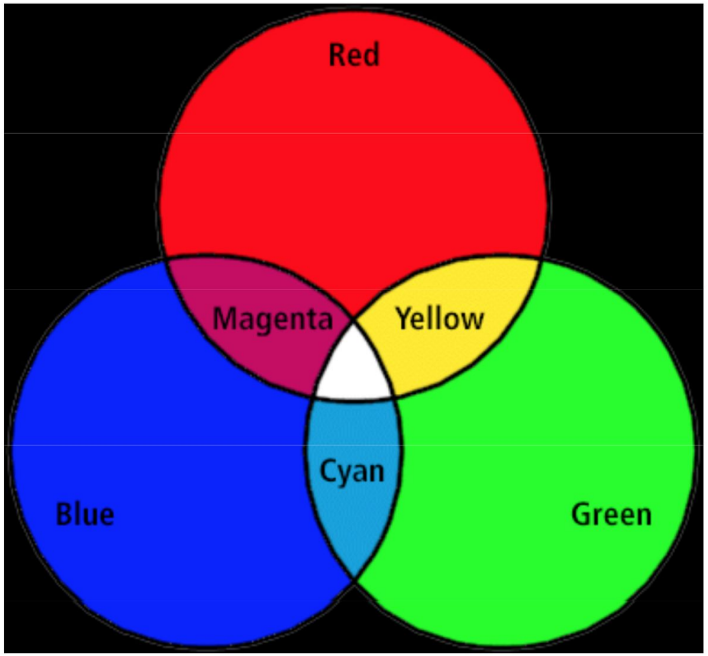
- Сетчатка содержит рецепторы
  - Колбочки
    - Дневной свет, оттенок цвета
    - R, G, B колбочки
  - Палочки
    - Ночной свет, яркость
- Чувствительность к свету
  - Luminance (brightness)
    - яркость
  - Chrominance
    - Hue (color tone)
      - Оттенок цвета, доминирующая длина волны
    - Saturation (color purity)
      - Относительная насыщенность цвета или количество белого света смешанного с оттенком



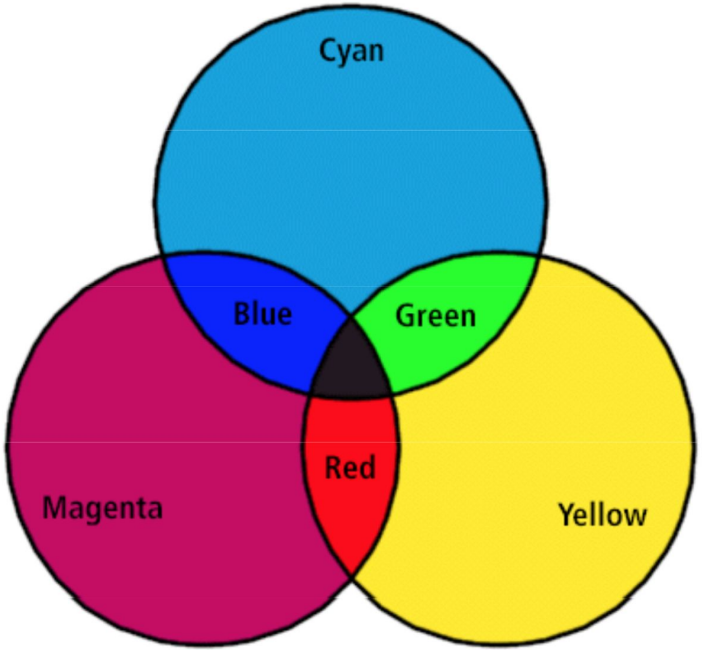
Трихроматическая теория цвета

- Любой цвет может быть получен смешиванием трех основных цветов в правильном отношении
- Основные цвета для излучающих источников:
  - R, G, B
  - R + G + B = white
- Основные цвета для отражающих источников:
  - Cyan, Magenta, Yellow (CMY)
  - В принтерах CMYK (=CMY + blacK)
  - R + G + B = black

RGB vs CMY



Magenta = Red + Blue
Cyan = Blue + Green
Yellow = Green + Red



Magenta = White - Green
Cyan = White - Red
Yellow = White - Blue

Цветное изображение

Red

Green

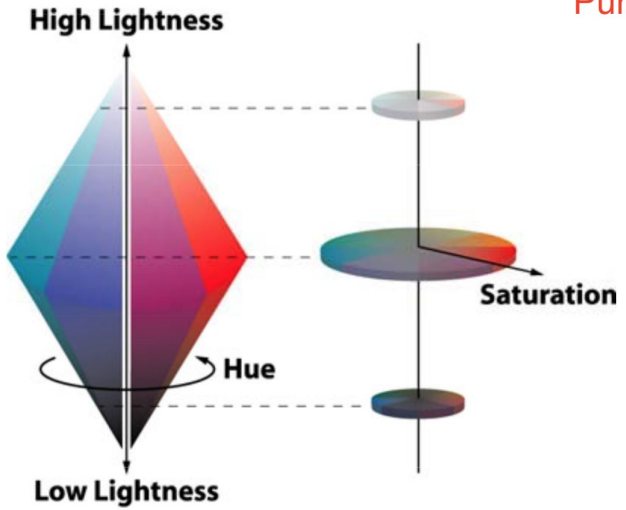
Blue

Три атрибута цвета

- Luminance (brightness) ← Intensity
- Chrominance
  - Hue (color tone) and Saturation

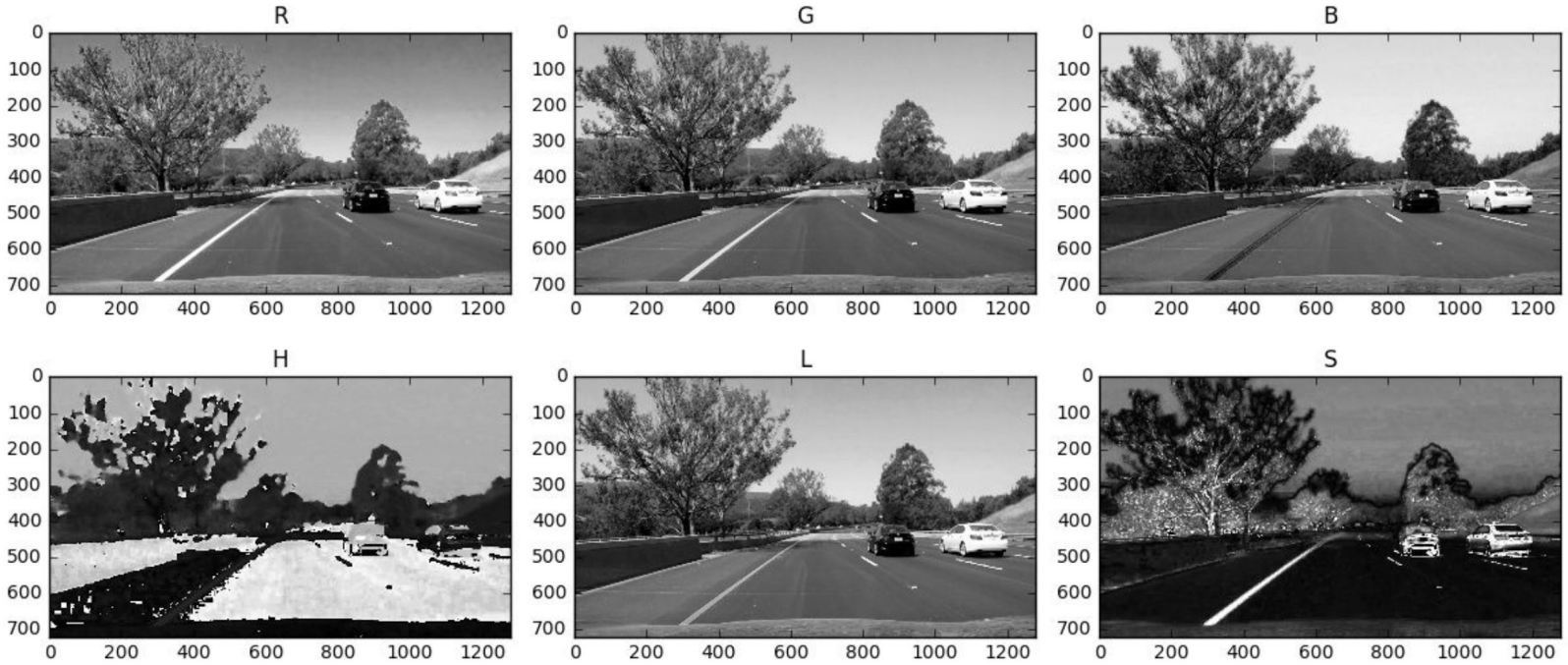
Dominant wavelength

Purity



HSV vs RGB

HSV vs RGB



Модель цвета

- Определите 3 первичных или вторичных цвета
  - RGB
  - CMY
- Либо определите luminance и chrominance
  - HSB(=HSV) or HSL = hue, saturation, value
  - YIQ~YUV (аналоговое ТВ)
  - YCbCr (цифровое ТВ)
- Выберите амплитуду
  - 8 битов на компоненту или 24 бита на пиксель

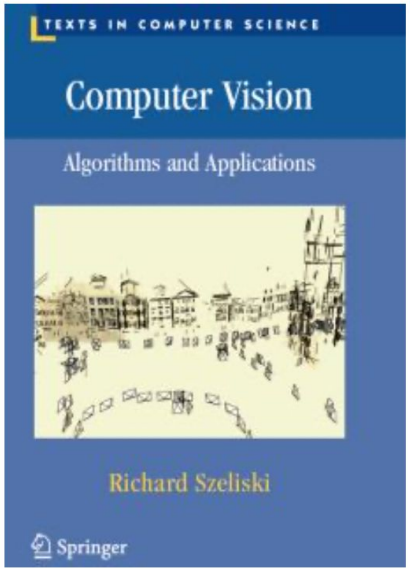
Квантизация света

24 bits -> 8 bits

Где смотреть полезную информацию

Рекомендуемые материалы

Computer vision. Algorithms and Applications,
Richard Szeliski
Доступна онлайн



Рекомендуемые материалы

ods.ai

- #cv
- #deep_learning
- #article_essence

Рекомендуемые материалы

Twitter

- Yann LeCun
- Andrej Karpathy
- Pieter Abbeel