ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ: БАЗОВАЯ ТЕОРИЯ

Мария Шеянова (masha.shejanova@gmail.com),
Вячеслав Мурашкин

Автор большей части материалов

ВЯЧЕСЛАВ МУРАШКИН

Был разработчиком-исследователем в «Яндексе», теперь работает в Google.

План занятия

- примеры применения Computer Vision
- цифровое представление изображений и пространство цветов
- посмотрим на изображения в питоне
- виды сжатия изображений
- свёртки и фильтры
- (если хватит времени) свёртки и фильтры с помощью питона

Рекомендуемые материалы

- книжка
- стэнфодрский курс
- подробный видеокурс
- канал в ютубе про нейросетки

Примеры задач CV

Фильтры для обработки изображений

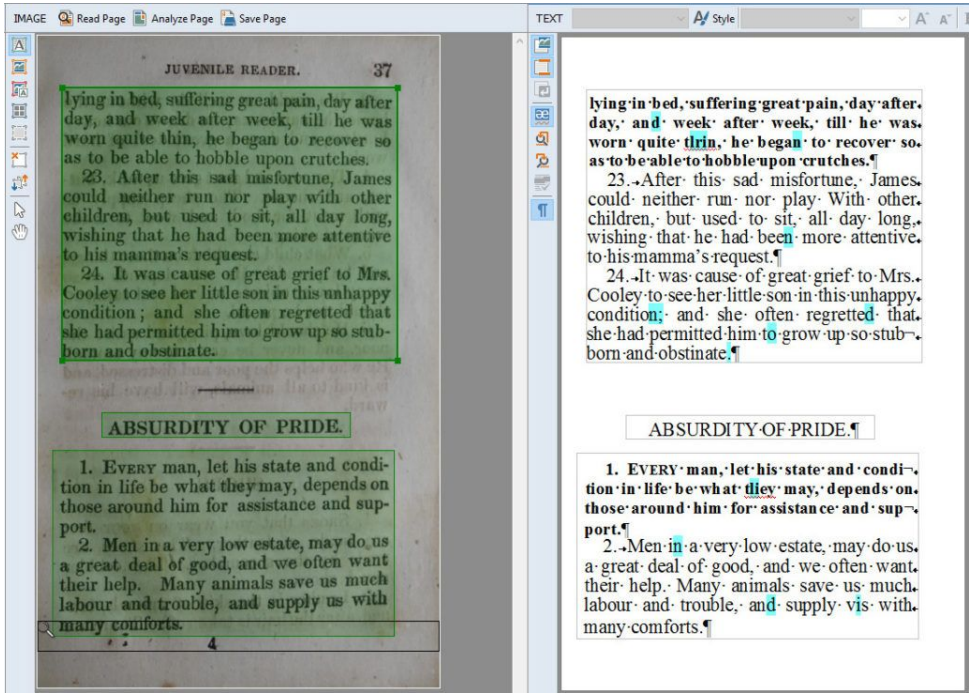
noisy lena

Gaussian filter

Удаление шума на изображении

Семантическая сегментация изображений

Распознавание символов



Цифровое представление изображений

растр vs. вектор

Растр

– описание изображения на уровне точек (пикселей);

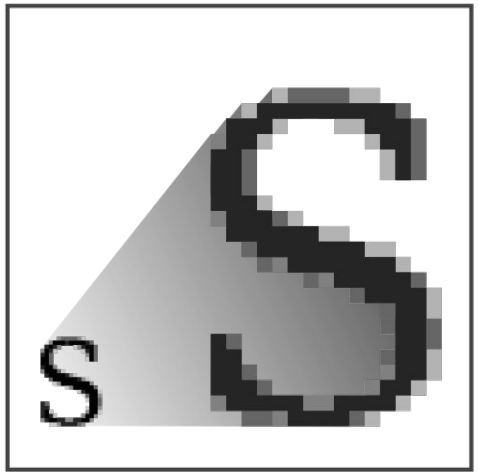
– размер изображения ограничен числом пикселей.

Вектор

– описание изображения на уровне фигур и их свойств;

– размер изображения может быть произвольным.

растр vs. вектор



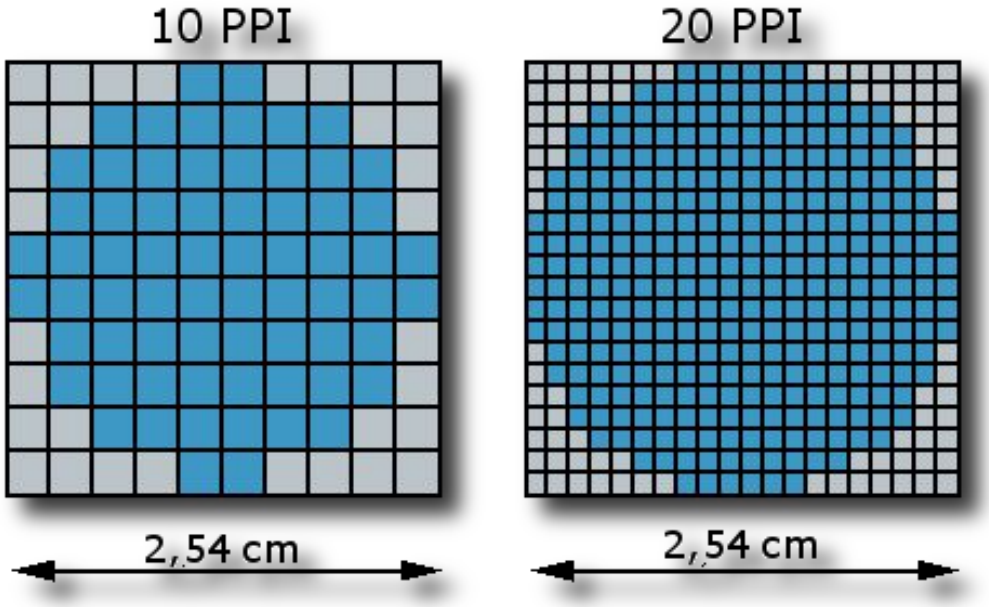
РАСТР

.jpeg .gif .png

ВЕКТОР

.svg

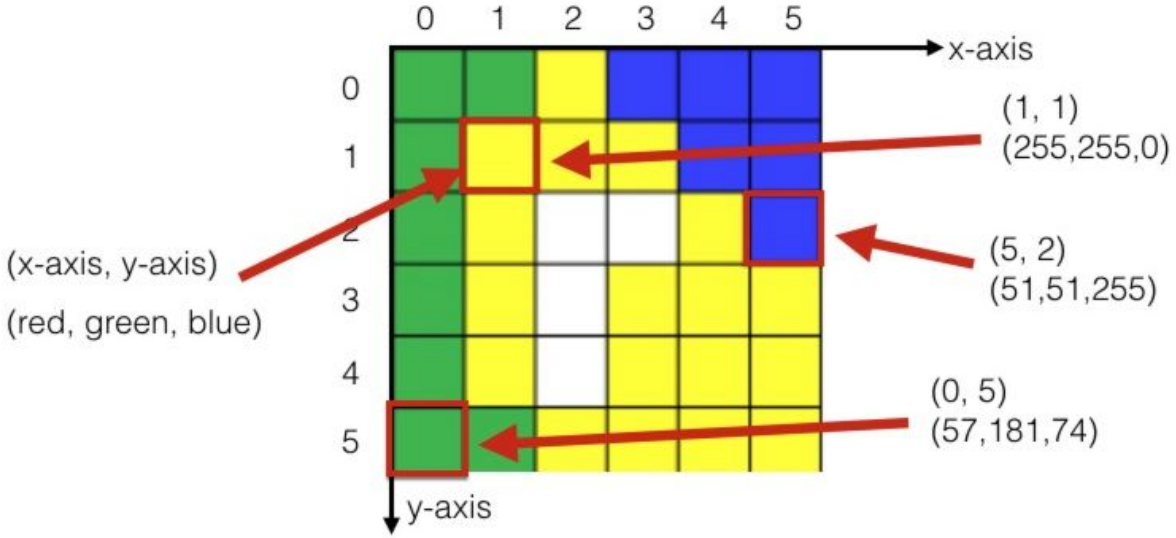
Разрешение



Каналы и динамический диапазон

- каждый пиксель изображения кодируется одним или несколькими значениями (каналами)
- стандартный диапазон значений в каждом канале: от 0 до 255 (один байт или 8 бит)
- для представления черно-белого изображения достаточного одного канала (передача яркости пикселя)
- цветные изображения, как правило, содержат 3 канала

Растровое представление изображения



Пространства цветов

RGB

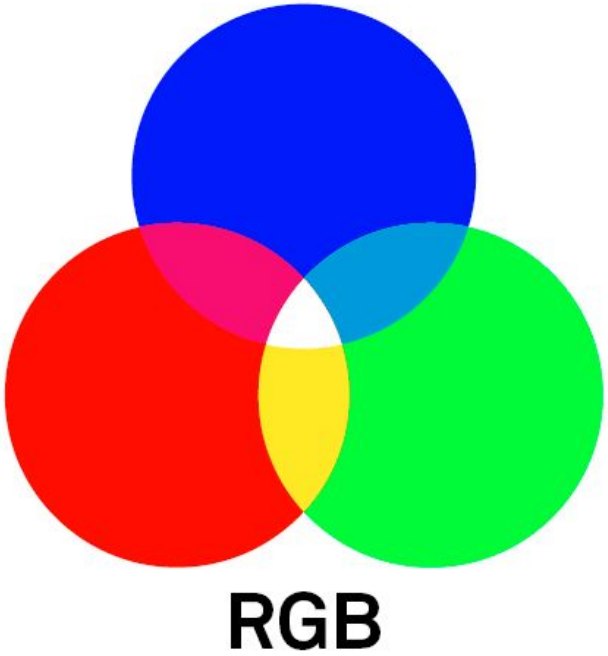
RGB - Red Green Blue

- наиболее распространенное представление цветного изображения
- выбор основных цветов обусловлен восприятием цвета сетчатки глаза
- 3 канала

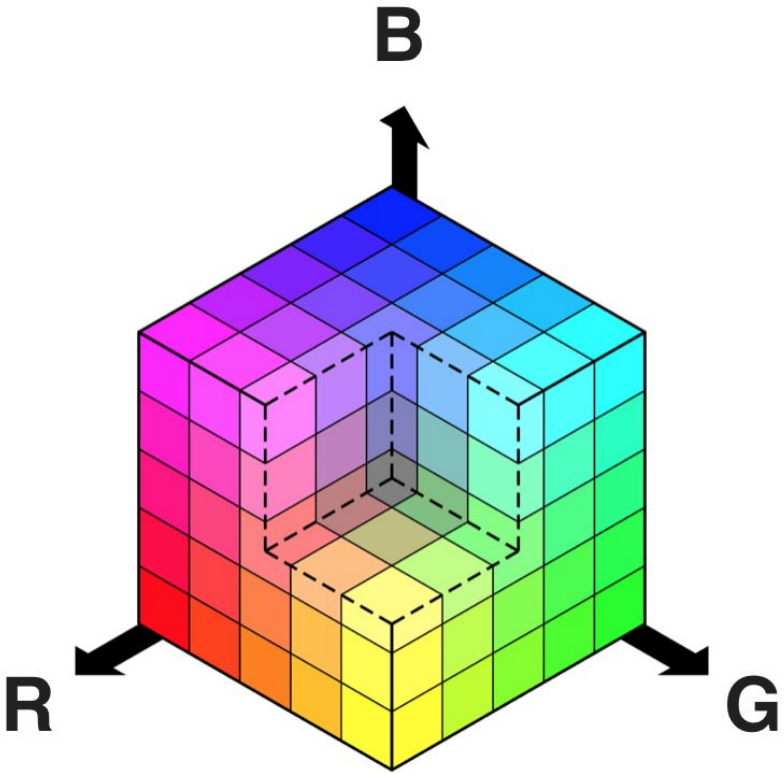
Примеры цветов:

- белый: (255, 255, 255)
- чёрный: (0, 0, 0)
- жёлтый: (255, 255, 0)

RGB



RGB



CMYK

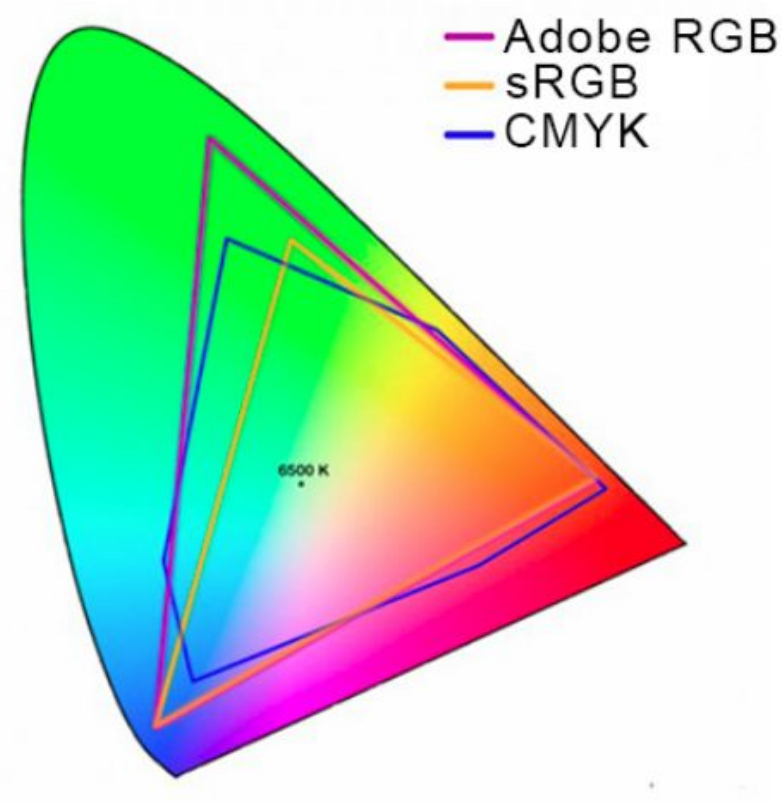
CMYK - Cyan Magenta Yellow Black

- 4 канала;
- в основном используется в полиграфии;
- цветовой охват меньше в сравнении с RGB.

CMYK VS RGB

sRGB - standard RGB

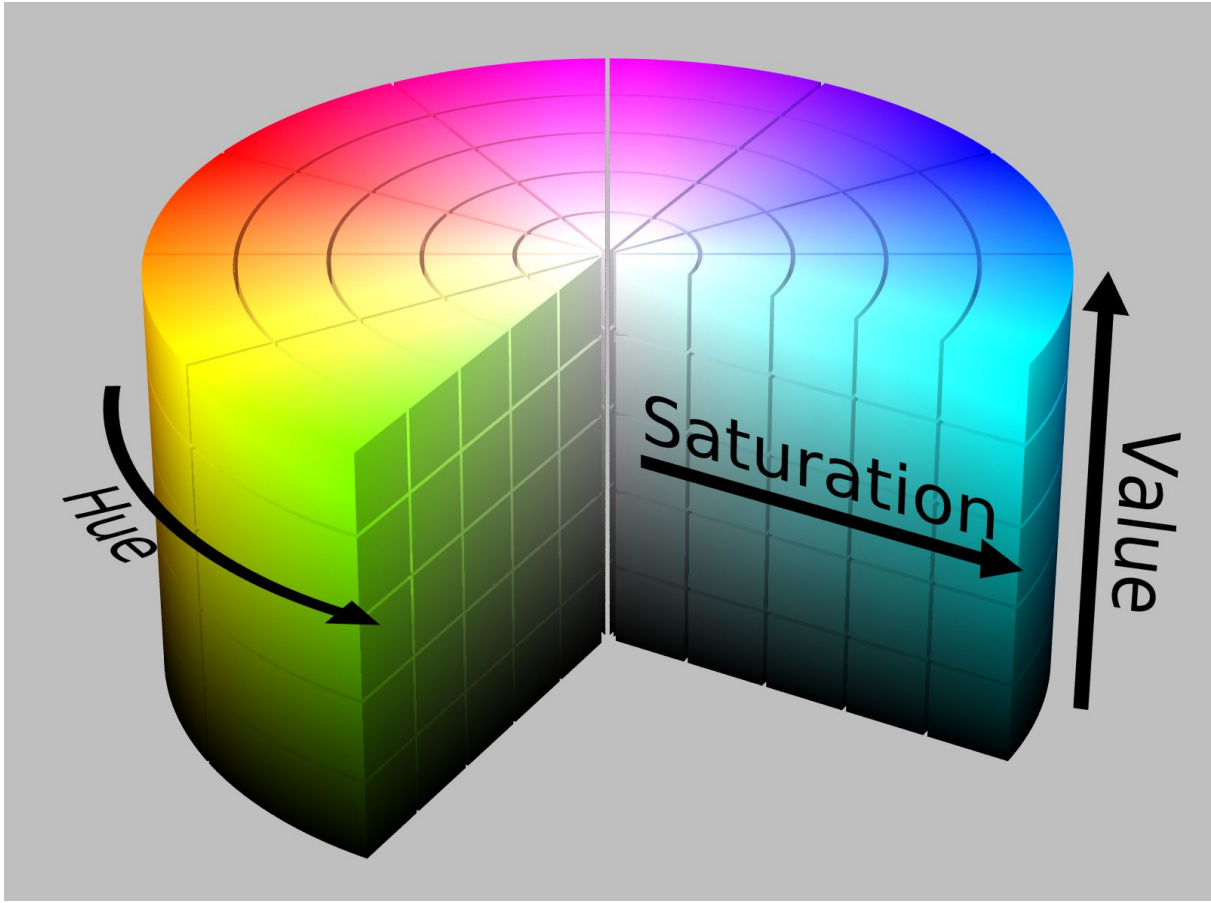
In Adobe RGB (1998), colors are specified as [R,G,B] triplets, where each of the R, G, and B components have values ranging between 0 and 1.



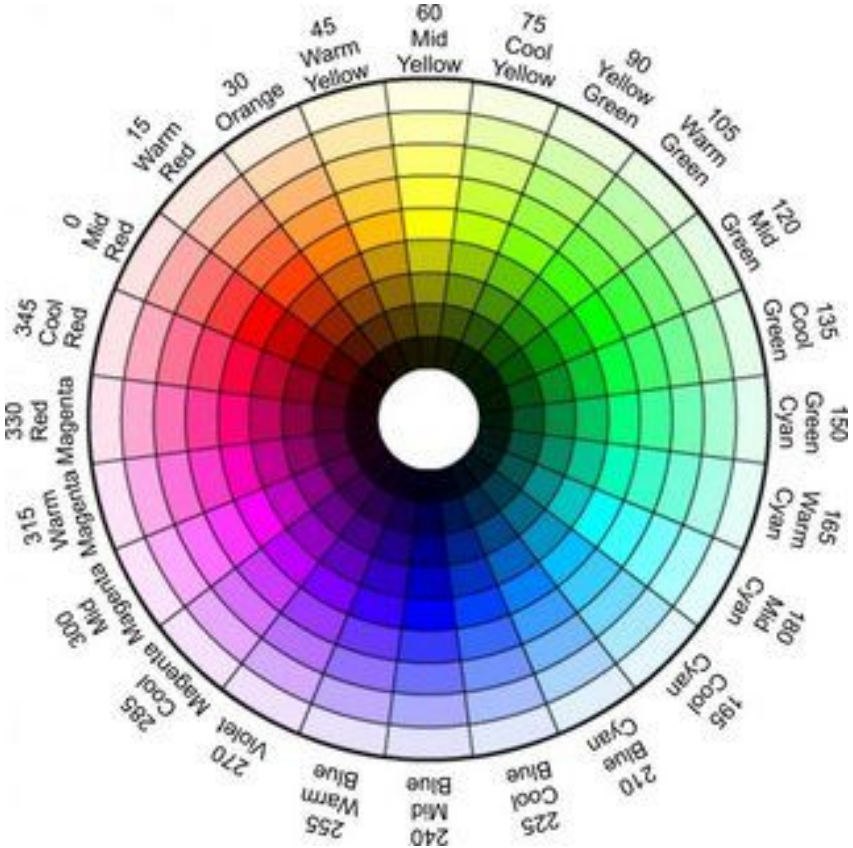
HSV

Hue **S**aturation **V**alue

- Hue — цветовой тон
- Saturation - насыщенность цвета
- Value - интенсивность



HSV: цвет


0
Mid
Red
15
Warm
Red
30
Orange
45
Warm
Yellow
60
Mid
Yellow
75
Cool
Yellow
90
Yellow
Green
105
Warm
Green
120
Mid
Green
135
Cool
Green
150
Green
Cyan
165
Warm
Cyan
180
Mid
Cyan
195
Cool
Cyan
210
Blue
Cyan
225
Cool
Blue
240
Mid
Blue
255
Warm
Blue
270
Violet
285
Cool
Magenta
300
Mid
Magenta
315
Warm
Magenta
330
Red
345
Cool
Red

Алгоритм перехода RGB → HSV

$$V \leftarrow max(R, G, B)$$

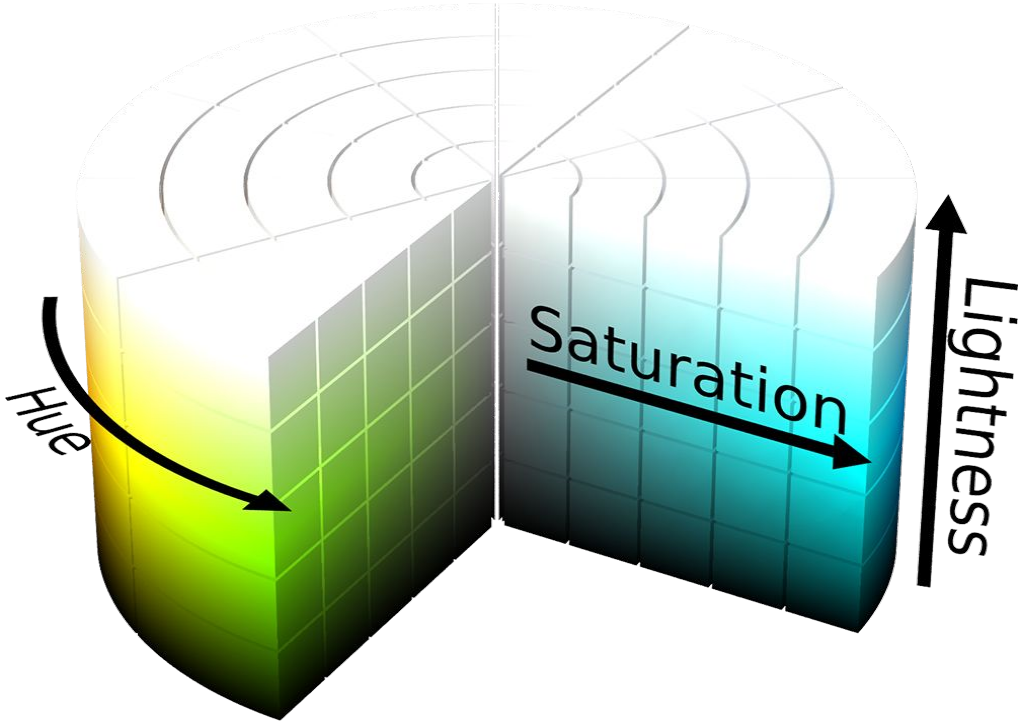
$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - min(R, G, B)) & \text{if } V = B \end{cases}$$

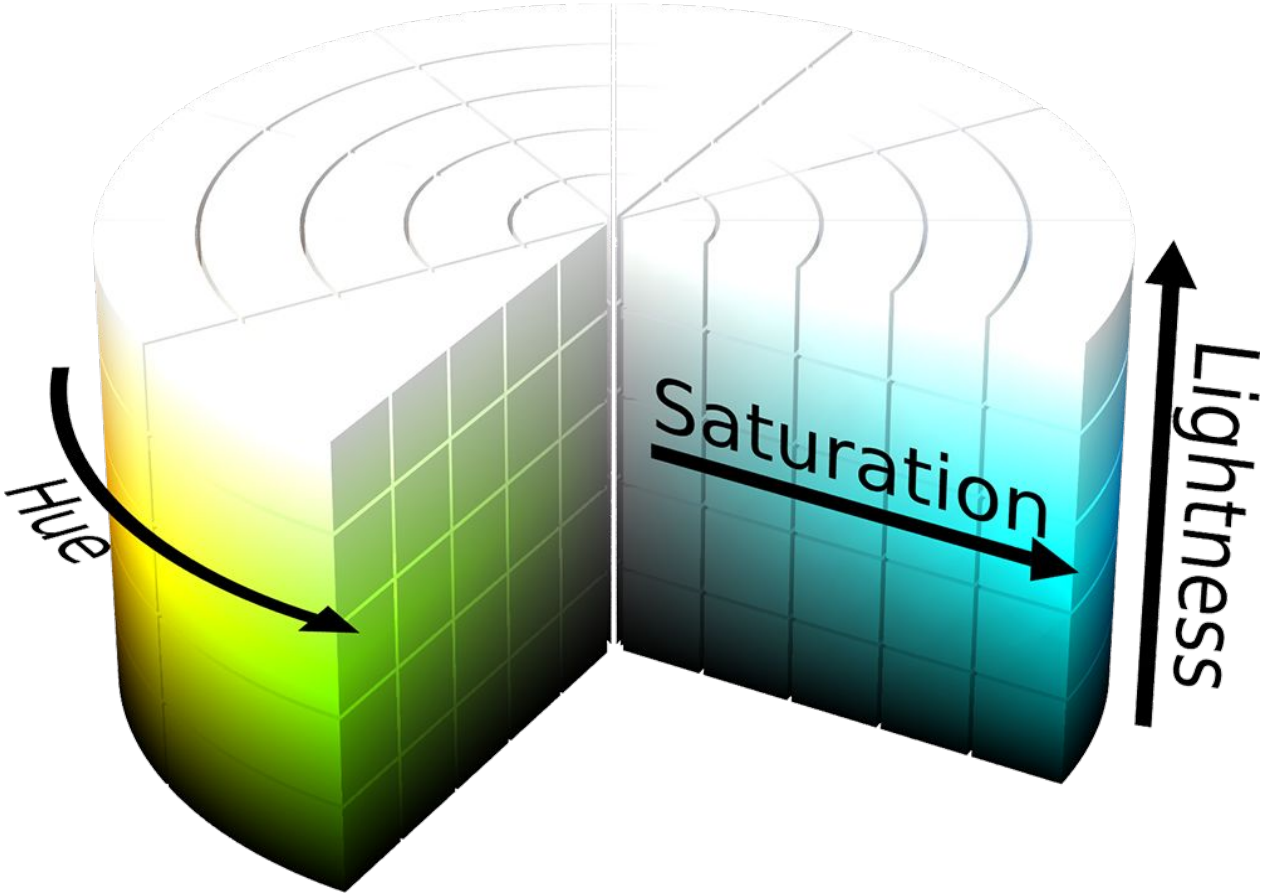
HSL

Hue **S**aturation **L**ightness

- Hue — цветовой тон
- Saturation — насыщенность цвета
- Lightness — яркость



HSL



Алгоритм перехода RGB → HLS

$$V_{max} \leftarrow max(R, G, B)$$

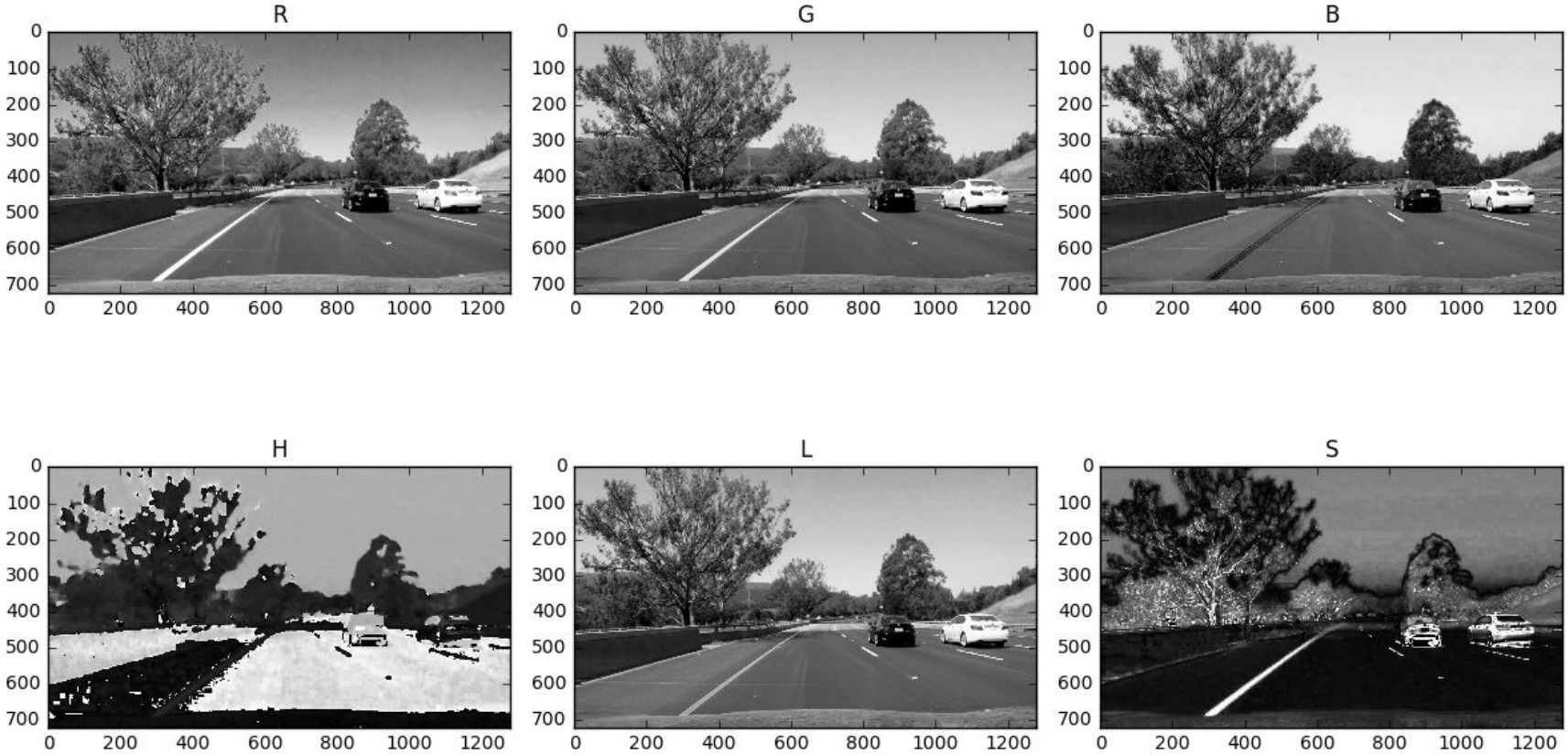
$$V_{min} \leftarrow min(R, G, B)$$

$$L \leftarrow \frac{V_{max} + V_{min}}{2}$$

$$S \leftarrow \begin{cases} \frac{V_{max} - V_{min}}{V_{max} + V_{min}} & \text{if } L < 0.5 \\ \frac{V_{max} - V_{min}}{2 - (V_{max} + V_{min})} & \text{if } L \geq 0.5 \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V_{max} - V_{min}) & \text{if } V_{max} = R \\ 120 + 60(B - R)/(V_{max} - V_{min}) & \text{if } V_{max} = G \\ 240 + 60(R - G)/(V_{max} - V_{min}) & \text{if } V_{max} = B \end{cases}$$

HSL VS RGB


R
G
B
H
L
S
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

Форматы сжатия

Сжатие изображений

- Матричное представление RGB требовательно к ресурсам памяти
- Каждый пиксель занимает 3 x 8bit = 24bit памяти
- Изображение 1024x768 занимает 2,4 Mb памяти
- Представление изображения в виде матриц RGB, как правило, содержит избыточную информацию;
- Сжатие изображений осуществляется за счёт уменьшения объема избыточной информации
- Сжатие бывает с потерями (восстановленное после сжатия изображение может отличаться от исходного) и без потерь (гарантируется что восстановленное после сжатия изображение совпадает с исходным)

Форматы изображений: JPEG

JPEG (Joint Photographic Experts Group)

- сжатие с потерями — восстановленное изображение не является точной копией исходного;
- уровень сжатия является параметром алгоритма;
- ориентировочный коэффициент сжатия цветного изображения: 10:1 — 20:1;
- использует особенность восприятия изображения человеческим глазом, связанную с большей чувствительностью к изменению яркости пикселей и меньшей чувствительностью к небольшому изменению цвета.

Форматы изображений: PNG

PNG (Portable Network Graphic)

- сжатие без потерь;
- палитра цветов изображения хранятся в таблице;
- для каждого пикселя указывается индекс цвета из палитры;
- ориентировочный коэффициент сжатия цветного изображения: 2,5:1.

Свёртки

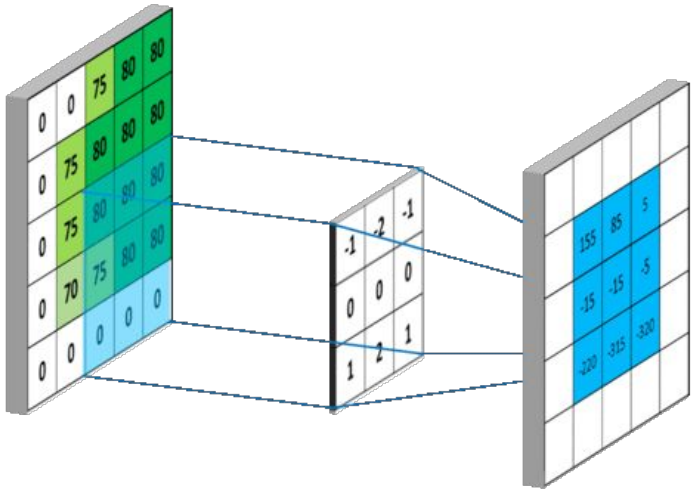
Что такое свёртка

Давайте пока представим, что мы работаем с чёрно-белыми изображениями (всего один канал).

Проходимся по всему изображению квадратной матрицей, поэлементно умножая яркость пикселей на её числа, а потом складываем результат.

Ссылка на анимацию.

Видео-объяснение про свёртки.



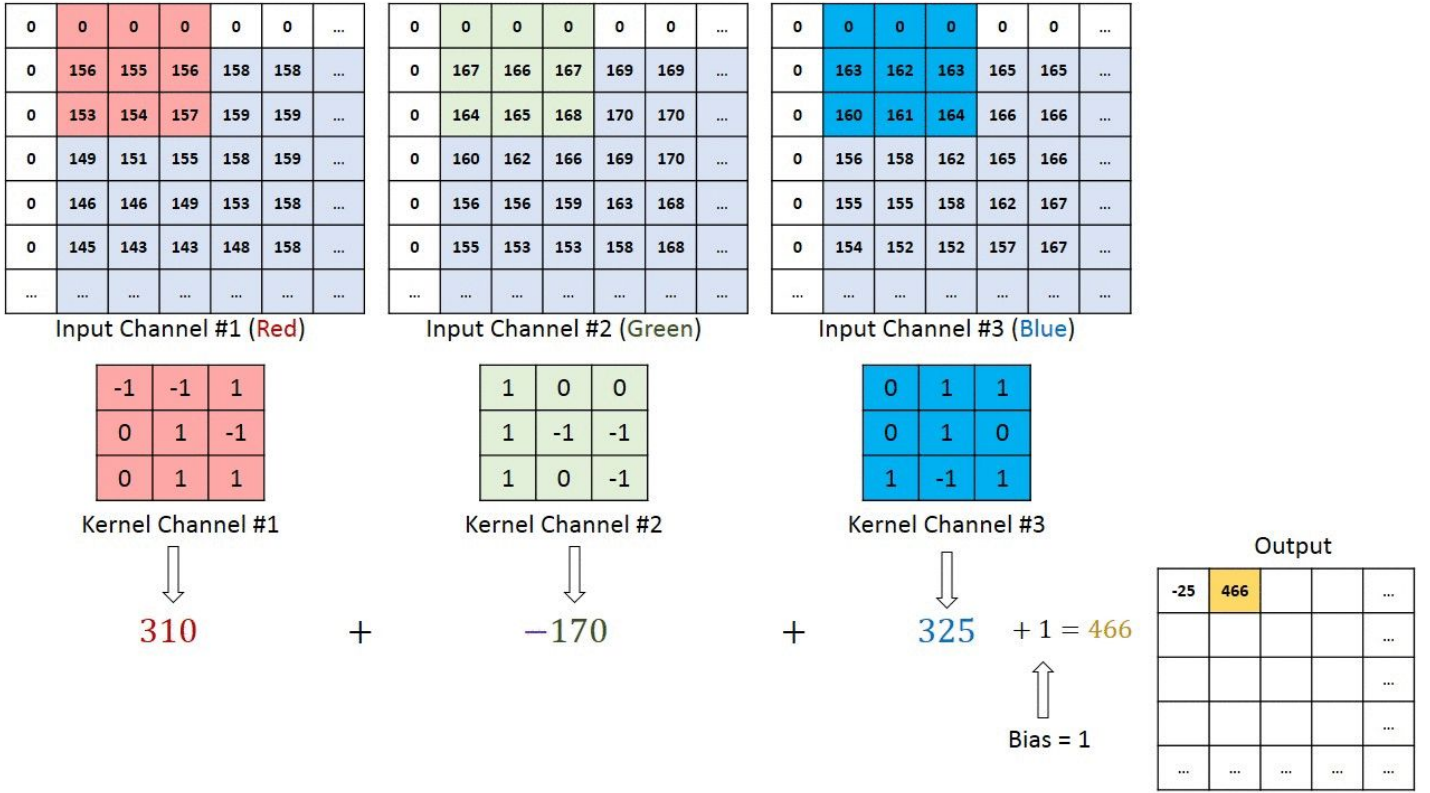
Свёртка: код для одномерной свёртки

```
/*
 * Размер выходной последовательности равен M + N - 1
 */
double * conv(double * x, int N, double * h, int M)
{
    double * result = new double[N + M - 1];
    memset(result, 0, sizeof(double) * (N + M - 1));

    for (int i = 0; i < N + M - 1; i++)
    {
        for (int j = 0; j < M; j++)
        {
            if (i-j >= 0 && i-j < N)
                result[i] += x[i - j] * h[j];
        }
    }

    return result;
}
```

Свёртка на цветных картинках



Делаем всё то же самое, но со всеми каналами, а потом складываем.

Анимация.

Фильтры

Фильтры: идея

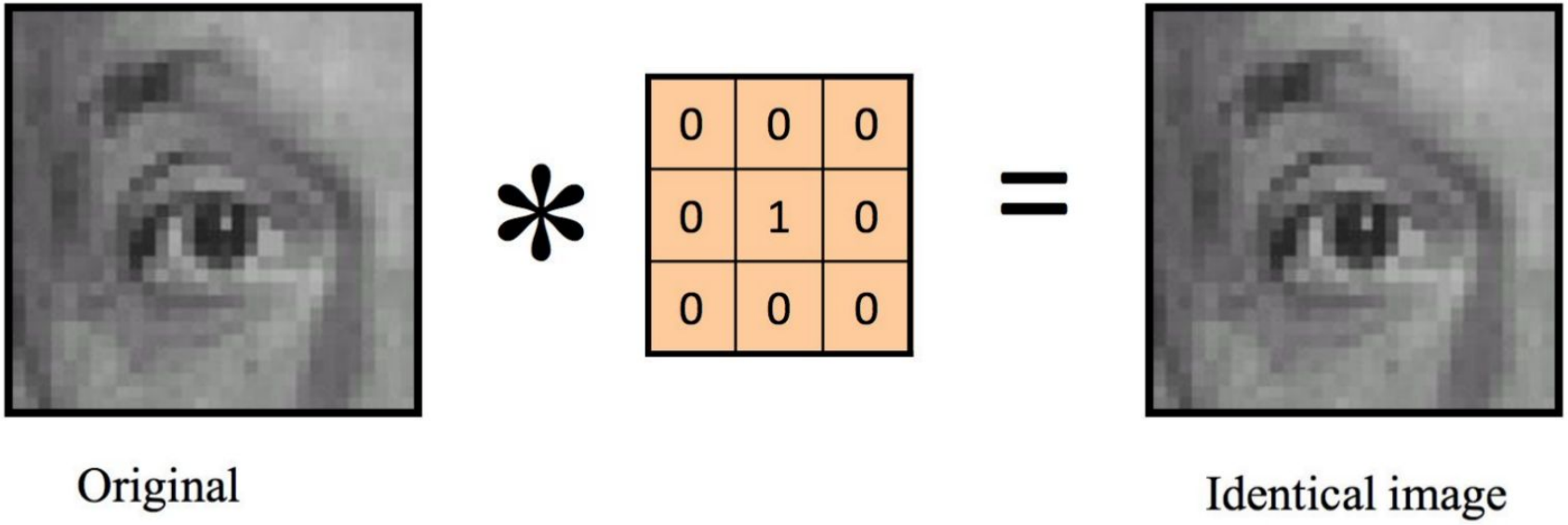
Проходимся по всему изображению, и меняем значения его пикселей

- в зависимости их значения
- в зависимости от их соседних пикселей

Таким образом, можно, например:

- выделять вертикальные / горизонтальные границы
- размывать изображение / делать его более чётким
- сдвигать изображение

Пример фильтра: идентичное отображение



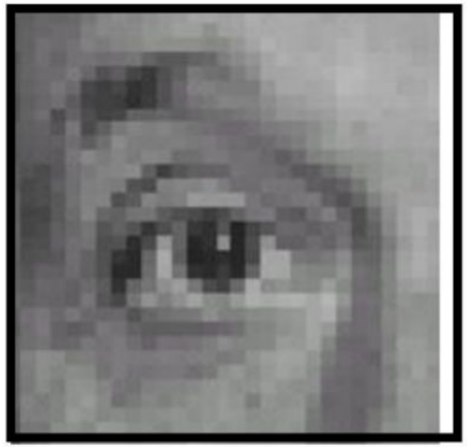
Пример фильтра: смещение

Original

*

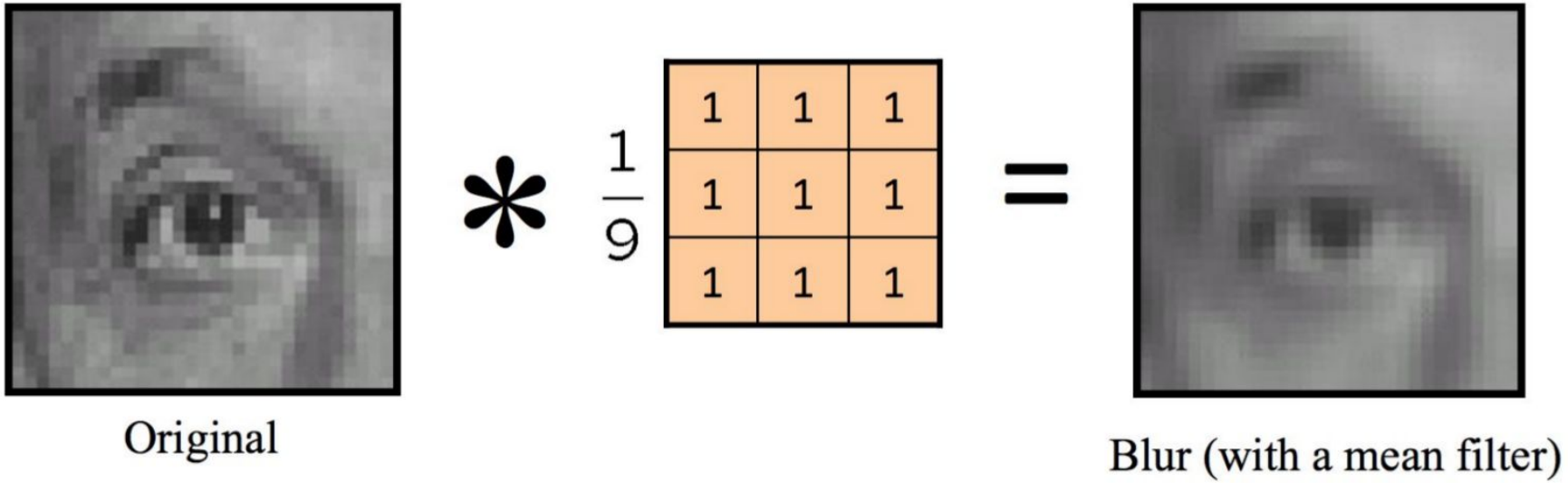
0	0	0
1	0	0
0	0	0

=



Shifted left
By 1 pixel

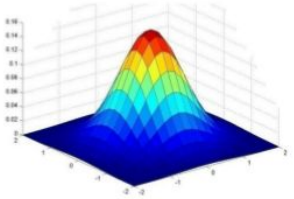
Пример фильтра: размытие

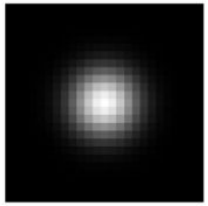


Original

Blur (with a mean filter)

Размытие - фильтр Гаусса

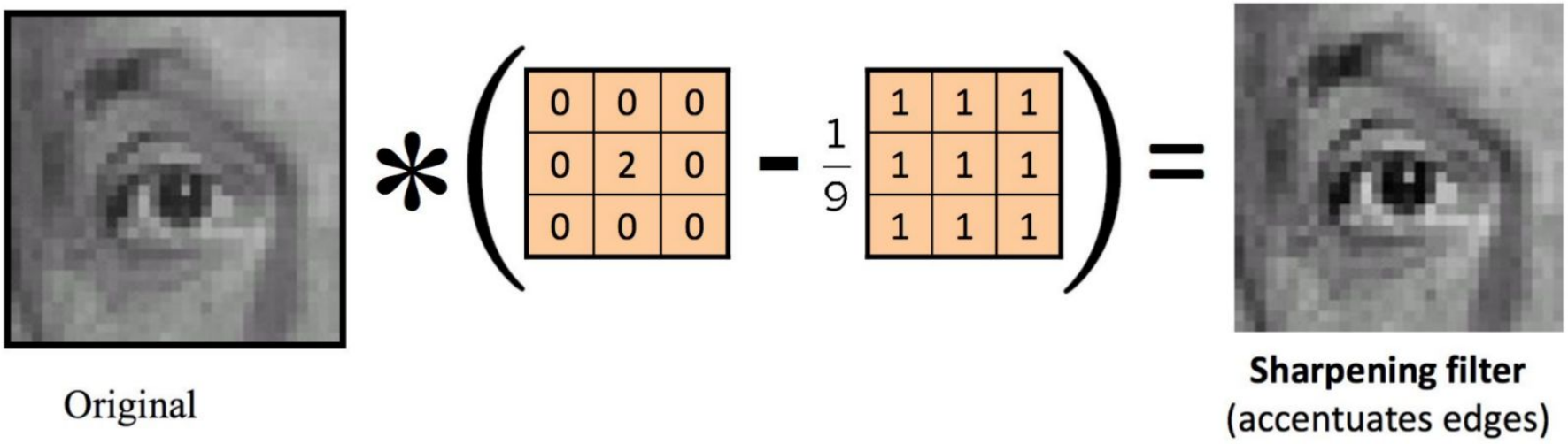




$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Пример фильтра: выделение границ (sharpen)



Выделение границ - оператор Лапласа

-1	-1	-1
-1	8	-1
-1	-1	-1

Выделение границ - оператор Собеля

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

x filter

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

y filter

Выделение границ - оператор Собеля

Original

Sobel X

Sobel Y

Оператор Собеля - градиент

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = arctan\frac{g_y}{g_x}$$

g, g_x, g_y - длина вектора градиента и его составляющих: **насколько сильный контраст**

theta - угол наклона градиента в полярной системе координат: **ориентация ребра**