

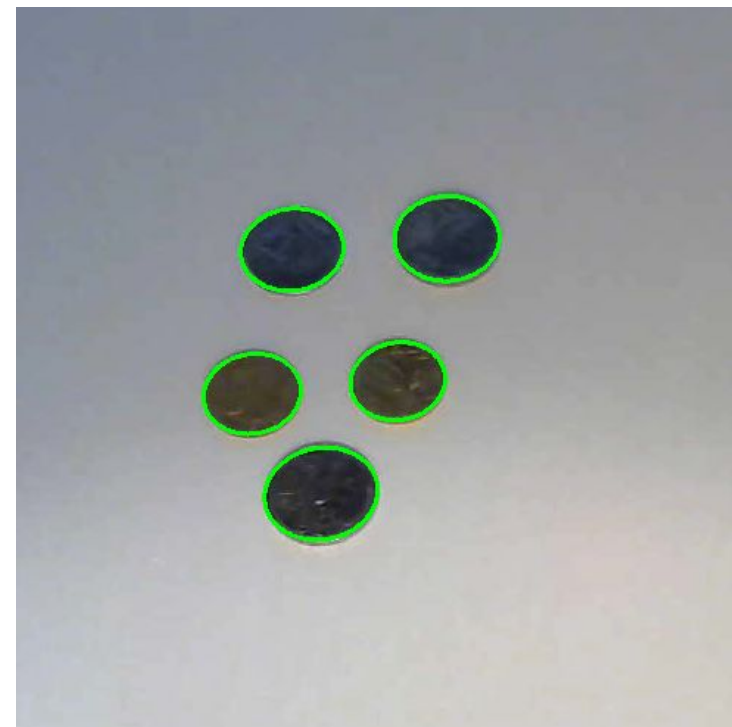
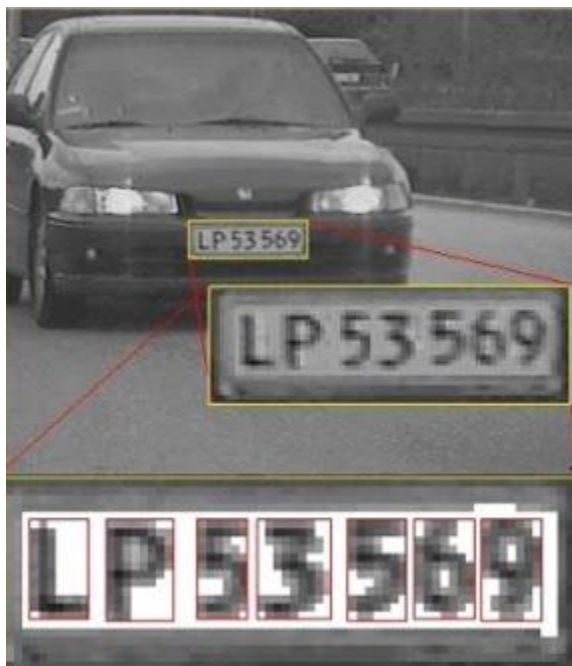
# Компьютерное зрение

Классический подход

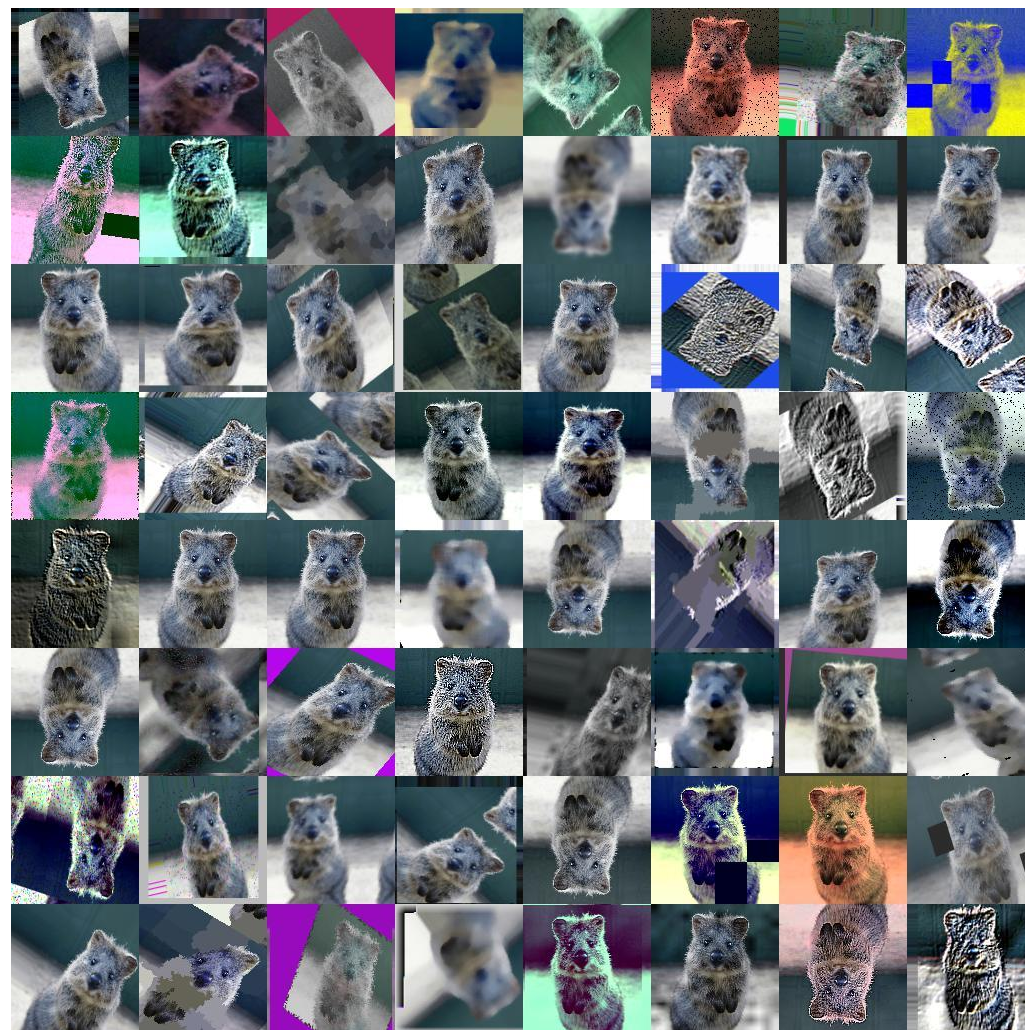
# Classical computer vision

- Преобразования над изображениями
  - Повышение контрастности, цветокоррекция
- Можно решить несложные, четко поставленные задачи
  - Найти номер автомобиля, вырезать однотонный фон
- Работает быстро
  - Может применяться на слабых девайсах

# Примеры задач



# Аугментации при обучении CNN





# Что такое изображение?

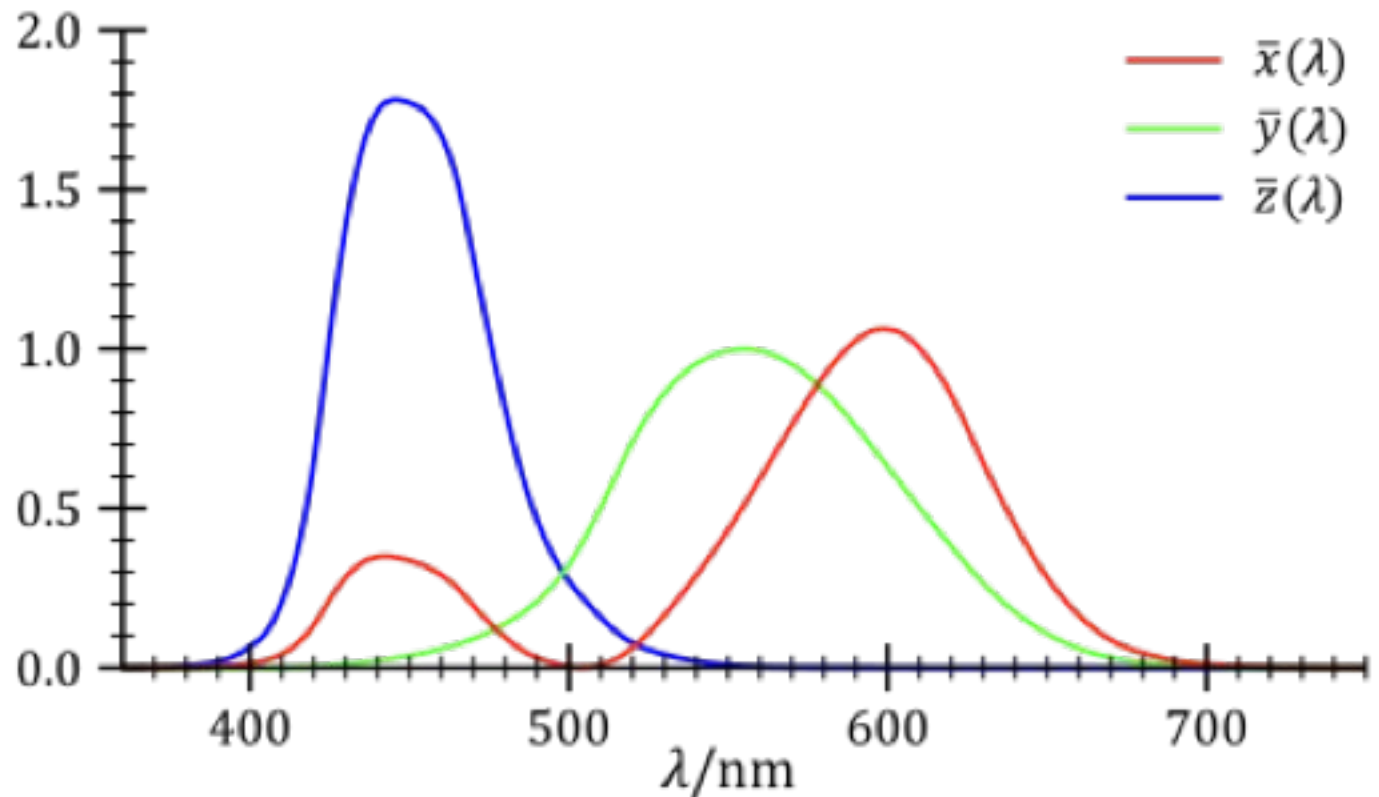


```
array([[116, 129, 143],  
       [117, 130, 144],  
       [117, 130, 144],  
       ...,  
       [200, 187, 189],  
       [198, 187, 189],  
       [197, 186, 188]],  
  
       [[116, 129, 143],  
        [117, 130, 144],  
        [117, 130, 144],  
        ...,  
        [200, 187, 189],  
        [198, 187, 189],  
        [197, 186, 188]],  
  
       [[116, 129, 143],  
        [117, 130, 144],  
        [117, 130, 144],  
        ...,  
        [200, 187, 189],  
        [198, 187, 189],  
        [197, 186, 188]],
```

Shape: (478, 718, 3)

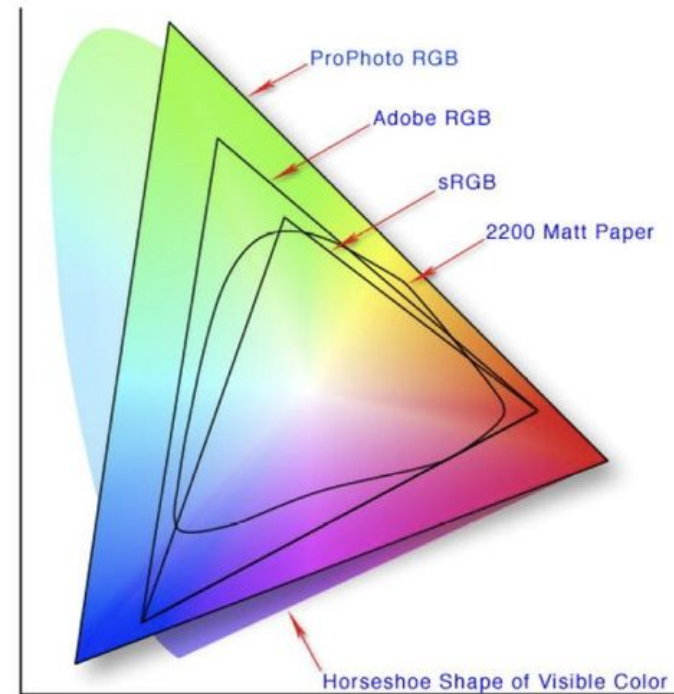
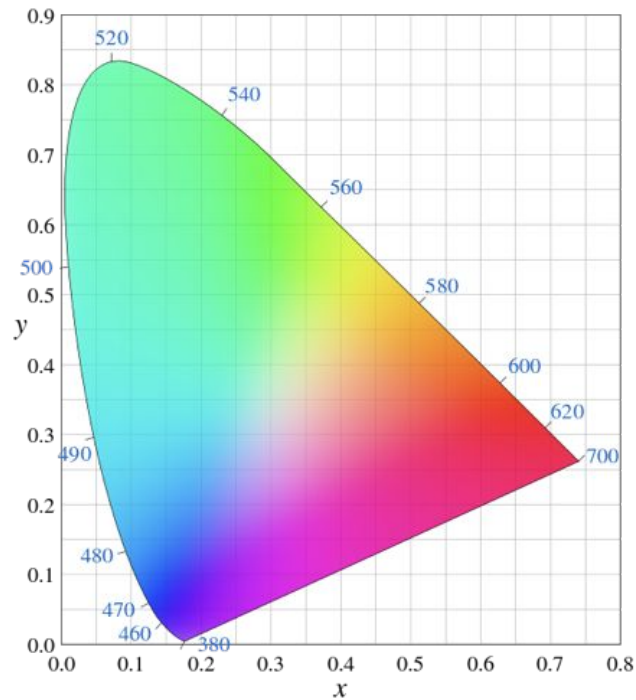
# Что такое цвет?

- Человек является трихроматом
- Сетчатка глаза имеет три вида рецепторов (колбочек)



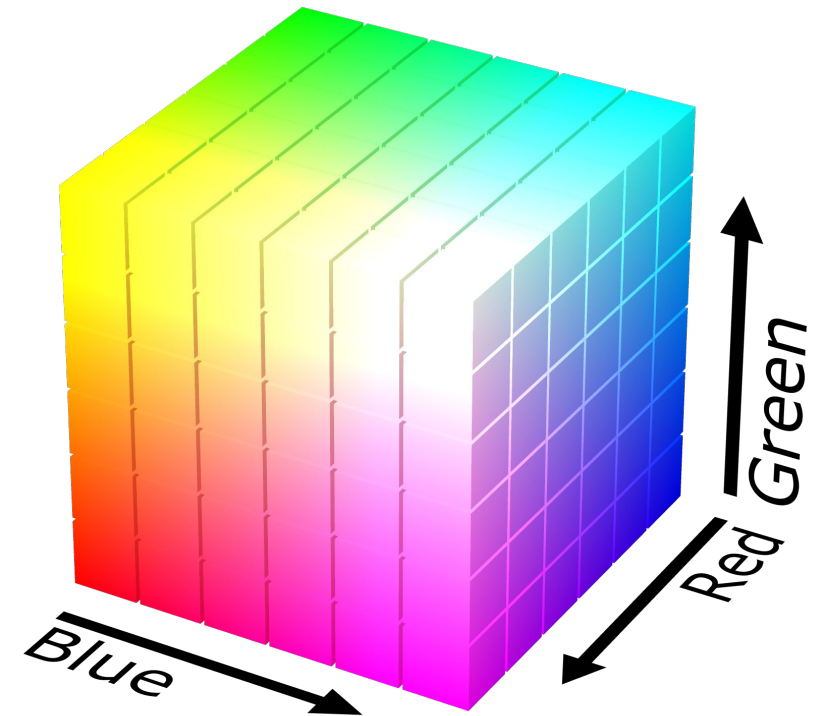
# Световая модель XYZ

- Самая полная из существующих
- Остальные охватывают только часть и пытаются приблизить ее



# Цветовое пространство RGB

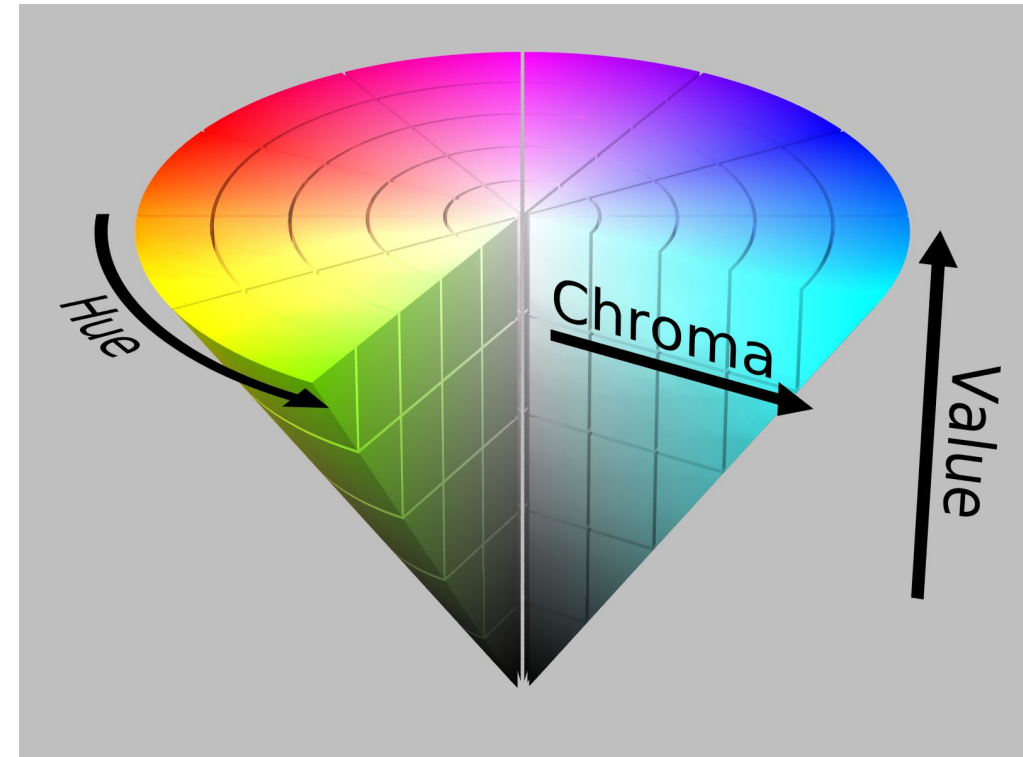
- Самый распространенный формат
- Цвет – комбинация Red, Green, Blue
- Значения каждого цвета в диапазоне  $[0, 255]$
- На разных мониторах выглядит по-разному
- Дистанции между цветами не соответствуют человеческому восприятию







# Цветовое пространство HSV

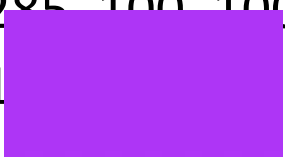
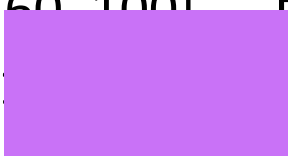
- Самый понятный формат
- Цвет это комбинация
  - Тона (Hue) - напр. зеленый или красный
  - Насыщенности (Chroma) - насыщенный или тусклый
  - Яркость (Value) – насколько темный
- Содержит меньше оттенков чем RGB



# HSV vs RGB в дизайне

- Подобрать аналогичный сочетающийся цвет





•   в HSV  
• [255, 0, 0] [255, 102, 102] в RGB

•   в HSV  
• [285, 100, 100] [217, 100, 100] в RGB



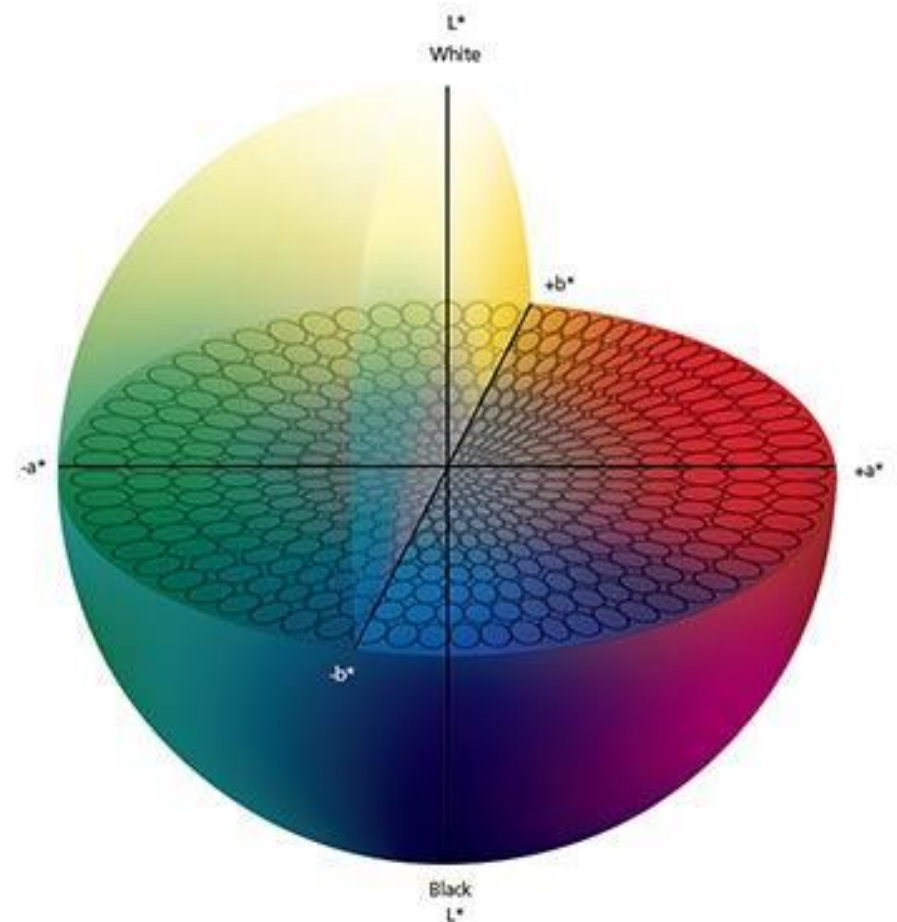
# Дистанция между цветами

## Euclidean distance in RGB

COLOR 1	COLOR 2	DISTANCE
		25.94
		21.40

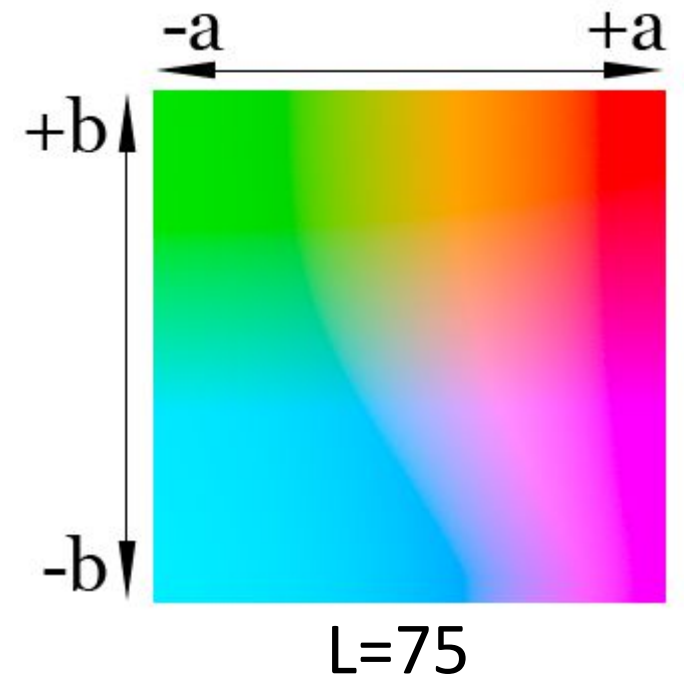
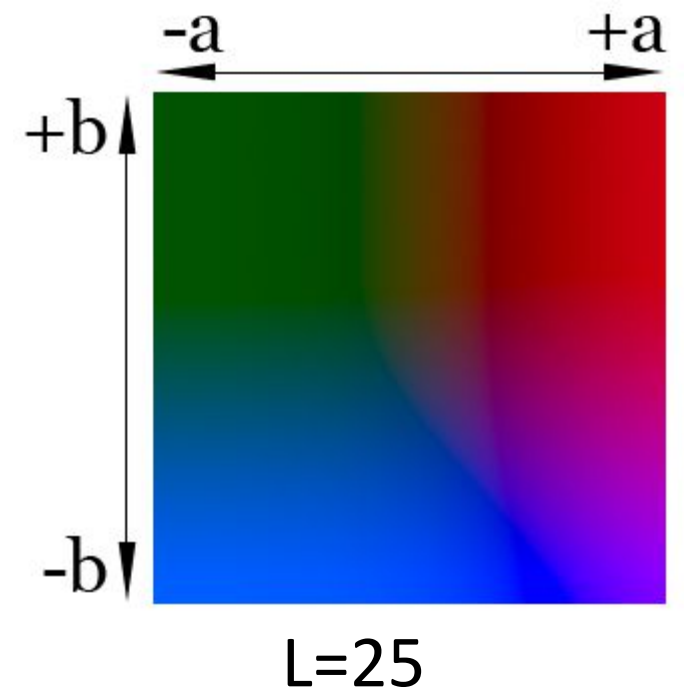
# Цветовое пространство LAB

- Максимально приближенное к человеческому восприятию
- Выглядит одинаково на всех устройствах
- Позволяется задавать эталонный белый цвет
- Имеет свою метрику для дистанции между цветами DeltaE2000



# Цветовое пространство LAB

- Параметр светлоты  $L$  изменяется от 0 до 100 (от самого тёмного до самого светлого)
- Хроматическая составляющая (тон и насыщенность) изменяется по двум координатам  $a$  (от красного до зелёного) и  $b$  (от синего до жёлтого)

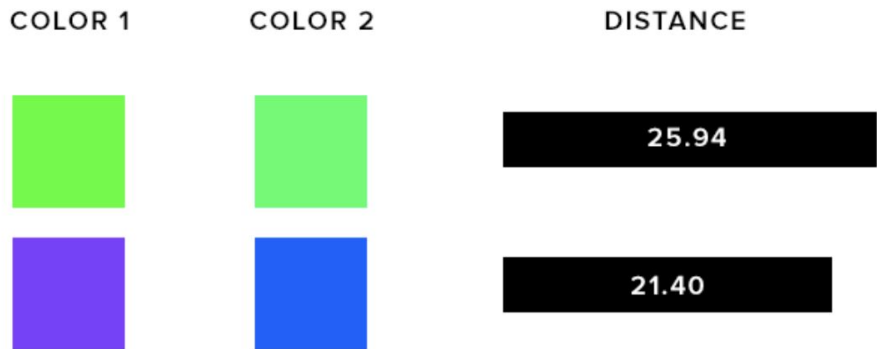




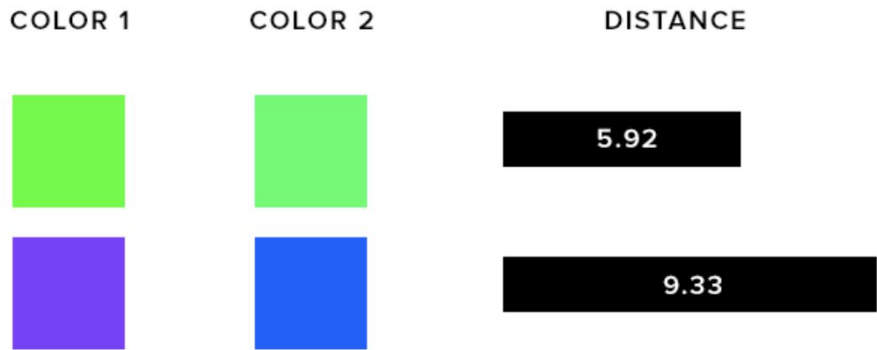
# DeltaE metric

Delta E	Perception
<= 1.0	Not perceptible by human eyes.
1 - 2	Perceptible through close observation.
2 - 10	Perceptible at a glance.
11 - 49	Colors are more similar than opposite
100	Colors are exact opposite

## Euclidean distance in RGB



## DeltaE distance in LAB

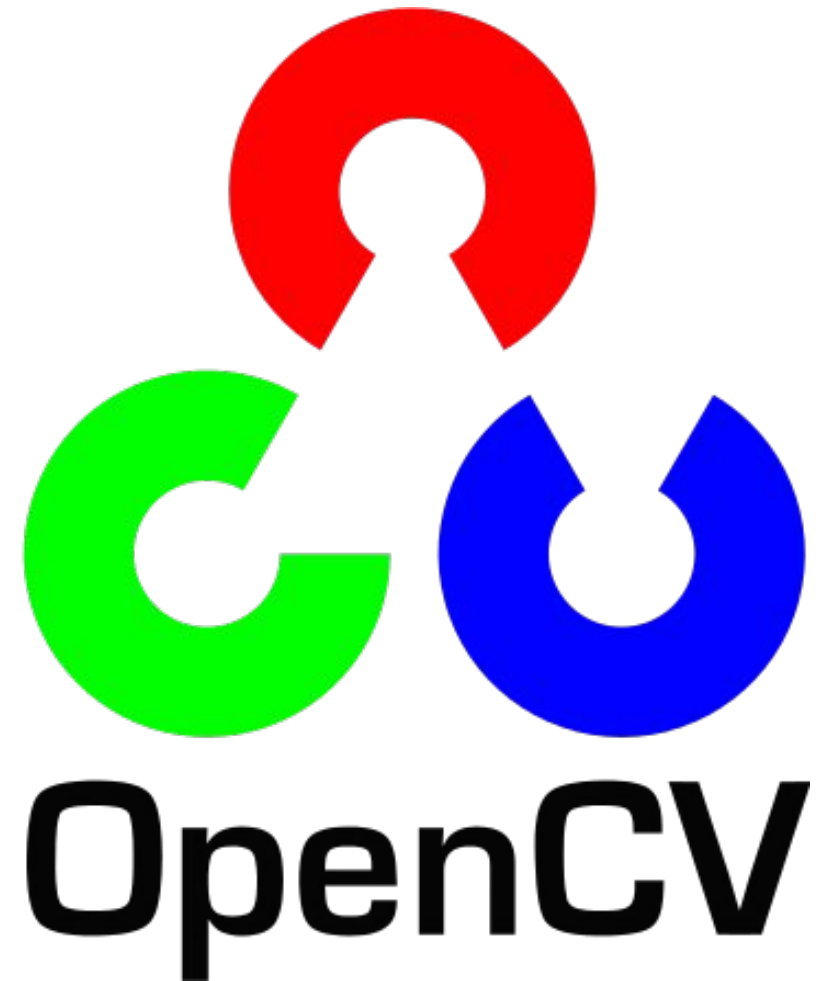


# Выводы:

- Если не требуются возможности HSV или LAB, то RGB будет достаточно
- HSV помогает удобно выделять цвета и находить цветовые сочетания
- Если ваша задача требует точной метрики для цветов – используйте LAB

# Преобразование изображений. OpenCV

- Разработчик Itseez (Российская компания, Нижний Новгород)
- Есть под C++, Java, Python
- Содержит функции для преобразований изображений и видео



# Типы операций

- Поточечные операции
  - Применяются независимо к каждому пикселю
- Глобальные операции
  - Используют информацию всего изображения сразу
- Локальные операции
  - Используют информацию в окрестности точки
  - Основаны на применении сверток

# Линейные операции с матрицей

- Прибавление числа к матрице

- $g(x) = f(x) + b$
- $g(x) = f(x) + b(x)$

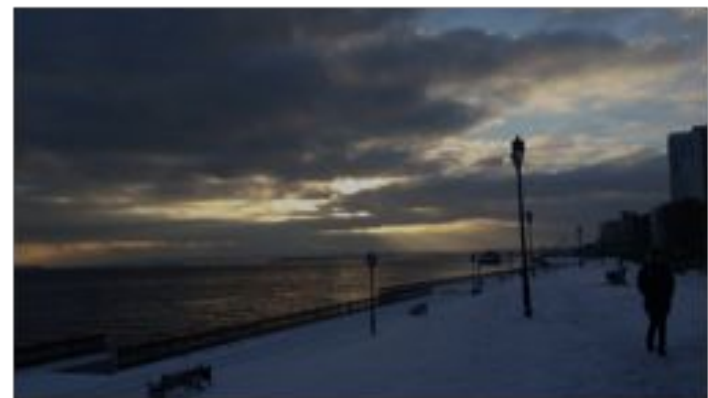
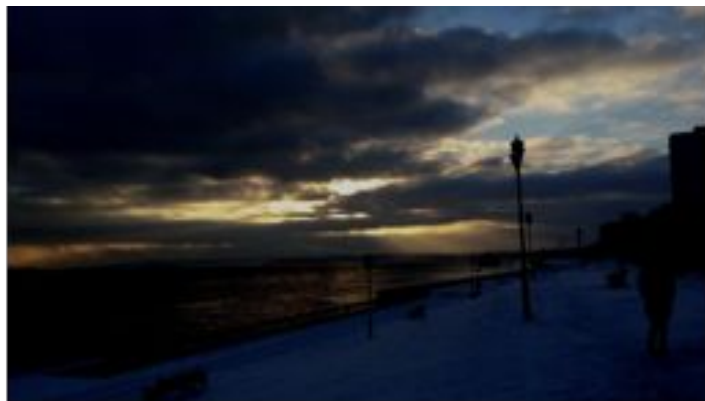
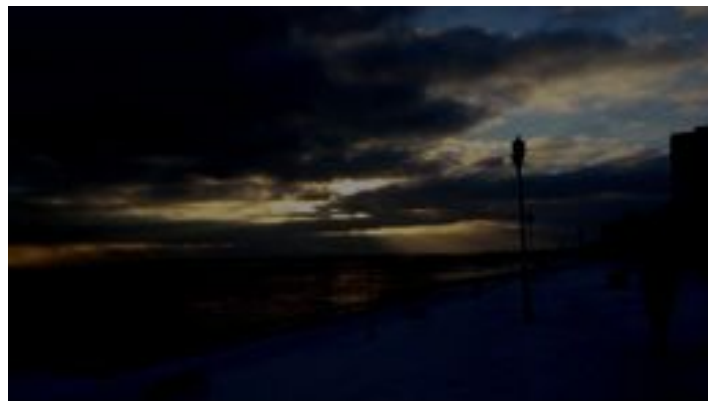
Увеличивает общую яркость

- Умножение элементов матрицы

- $g(x) = a * f(x)$
- $g(x) = a(x) * f(x)$

Увеличивает контрастность





# Threshold (пороговое преобразование)

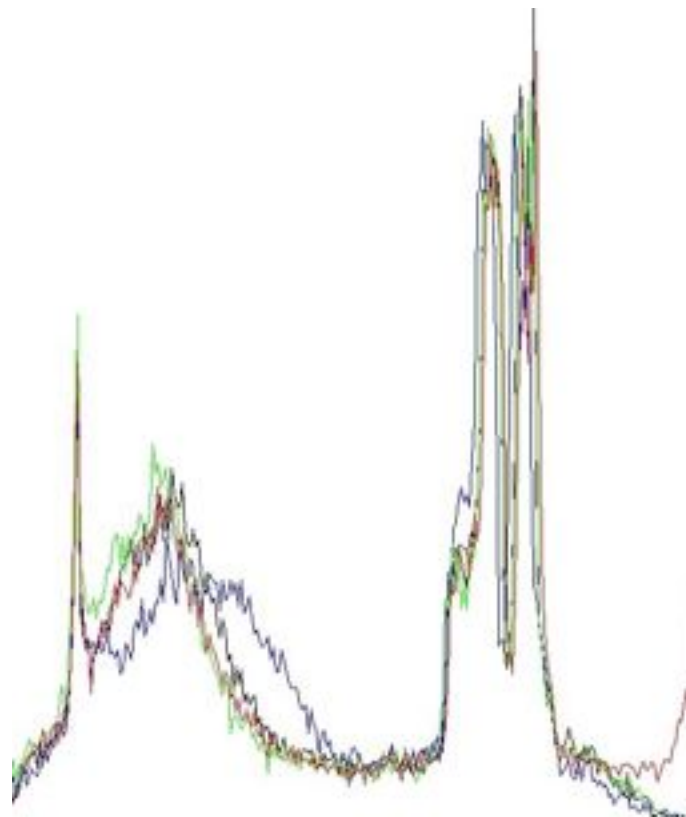
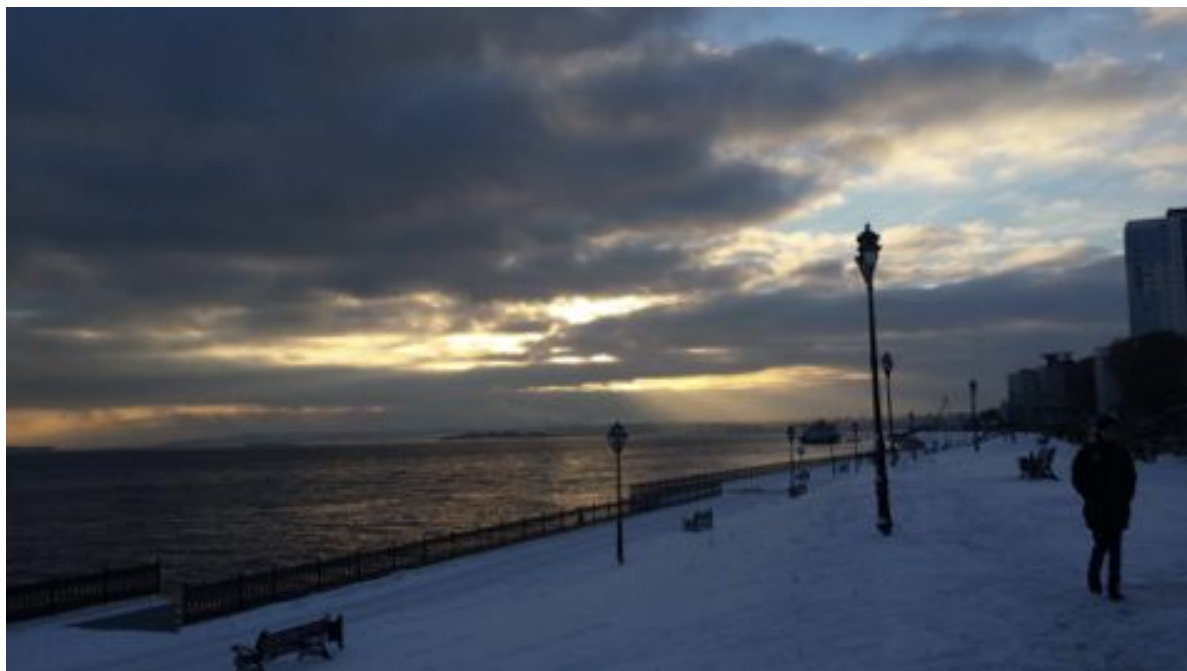
- Выделяет часть изображения, которая лежит в заданном цветовом диапазоне
- Удобнее задавать диапазон в HSV

```
weaker = np.array([0,0,100])  
stronger = np.array([10,255,255])  
  
mask = cv2.inRange(img, weaker, stronger)
```

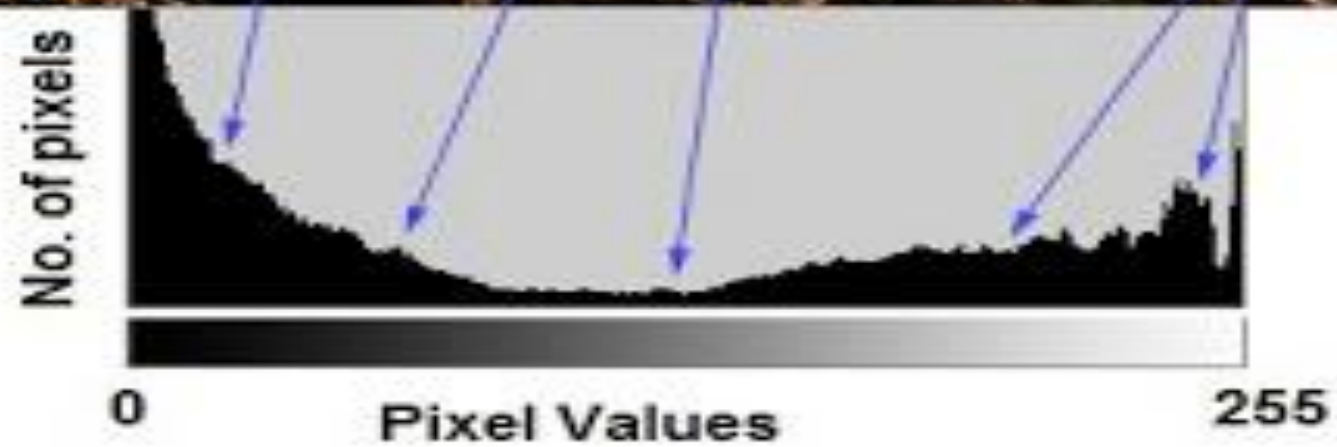


# Глобальные преобразования. Гистограммы

- Производится подсчет количества пикселей с определенной яркостью

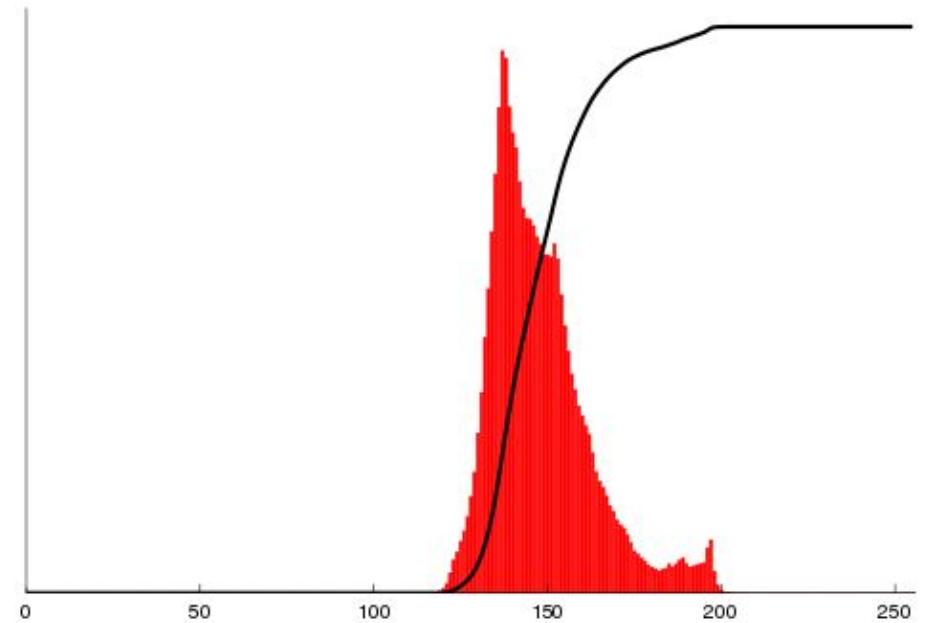






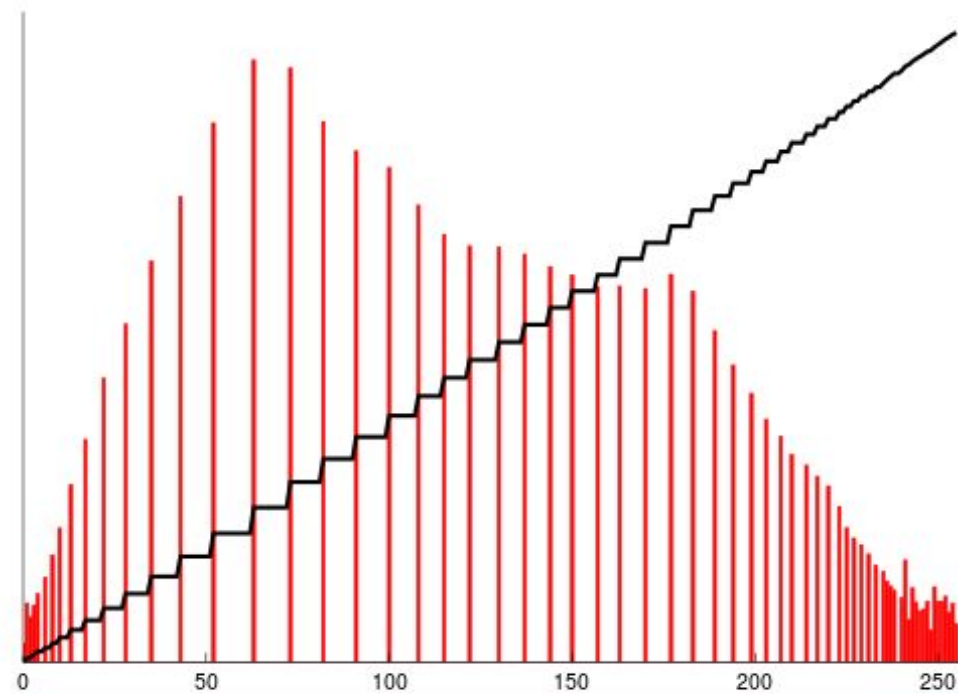
# Histogram equalization

- Метод повышения контрастности изображения





# Histogram equalization 2



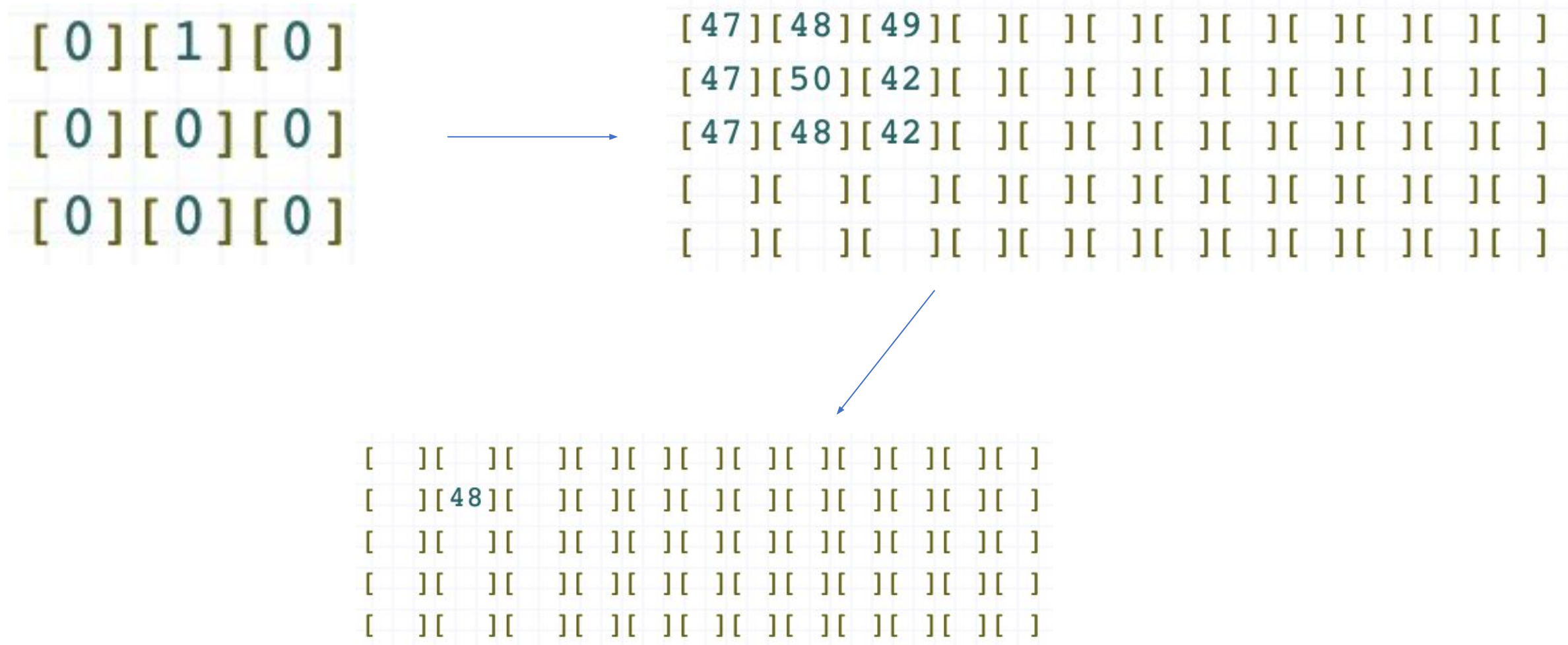
# Локальные операции. Свертка

- Свёртка (англ. convolution) — это операция, показывающая «схожесть» одной функции с отражённой и сдвинутой копией другой.
- Для нас свертка это матрица  $k$  на  $k$
- Центральный элемент будем называть якорем
- Перемножаем элементы из свертки с пикселями изображения и записываем сумму в якорь

[	]	[	]	[	]
[	]	[я]	[	]	
[	]	[	]	[	]

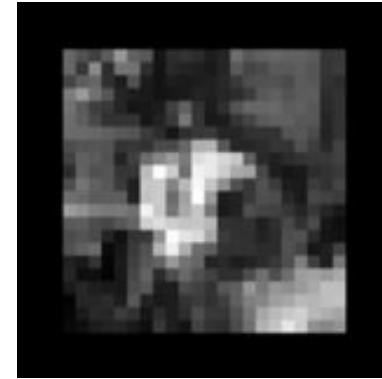
[0]	[1]	[0]
[0]	[0]	[0]
[0]	[0]	[0]

# Свертка. Пример подсчета



# Проблема границы

- Стратегии как преодолеть эту проблему
- Добавить рамку (padding) размера  $k$ 
  - Заполнить нулями
  - Зеркальное отражение
  - Продлить значения с границы



# Average filter

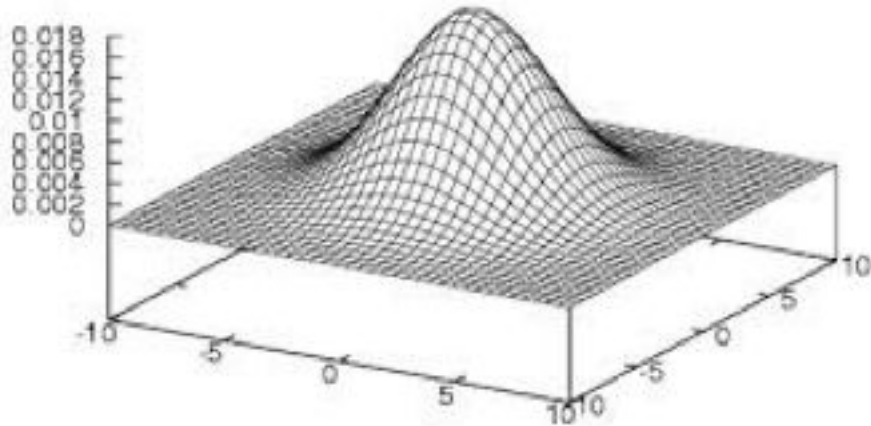
$$\frac{1}{K} \begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \\ 1 & \dots & 1 \end{bmatrix}$$





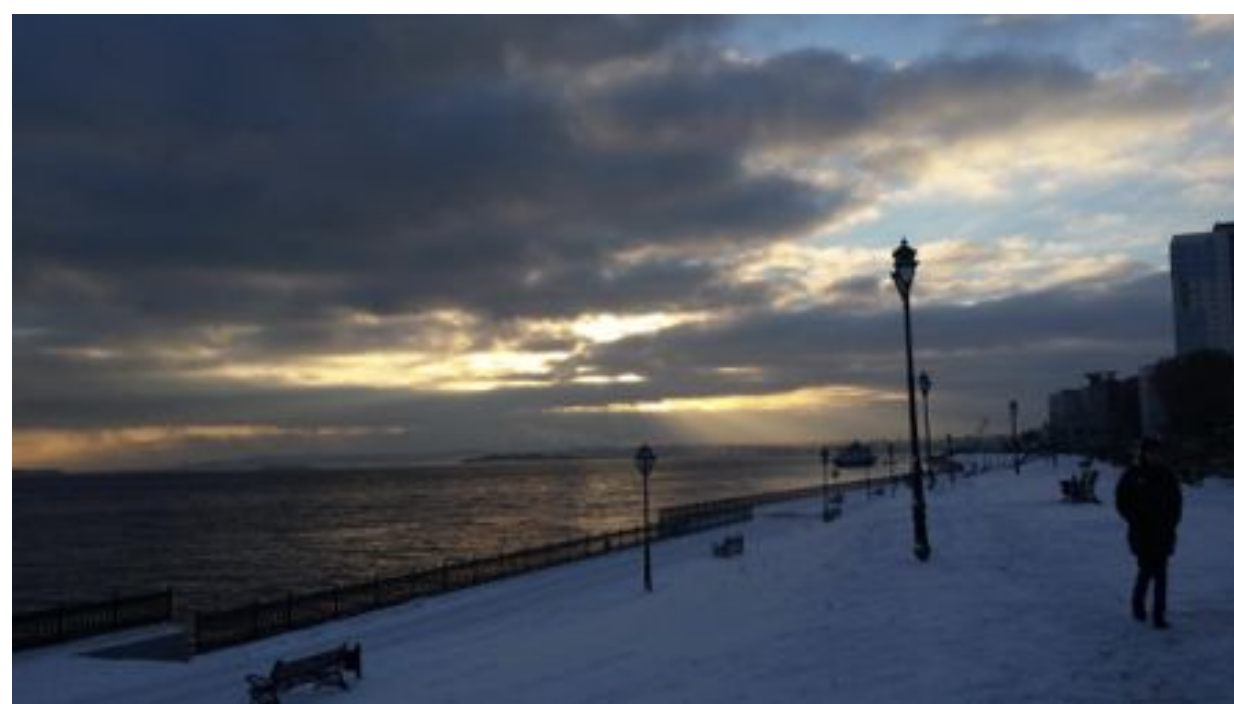
# Gaussian filter

$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$



# Median filter

- Берем kernel 3x3
- Под ядром будет 9 пикселей
- В качестве нового значения берем медиану
  - Сортируем значения и берем значение в центре массива
- Удаляет шум а не размазывает его



# Оператор Собеля

- Дифференциальный оператор
- Разность яркости
- Выделение границ

$$\frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

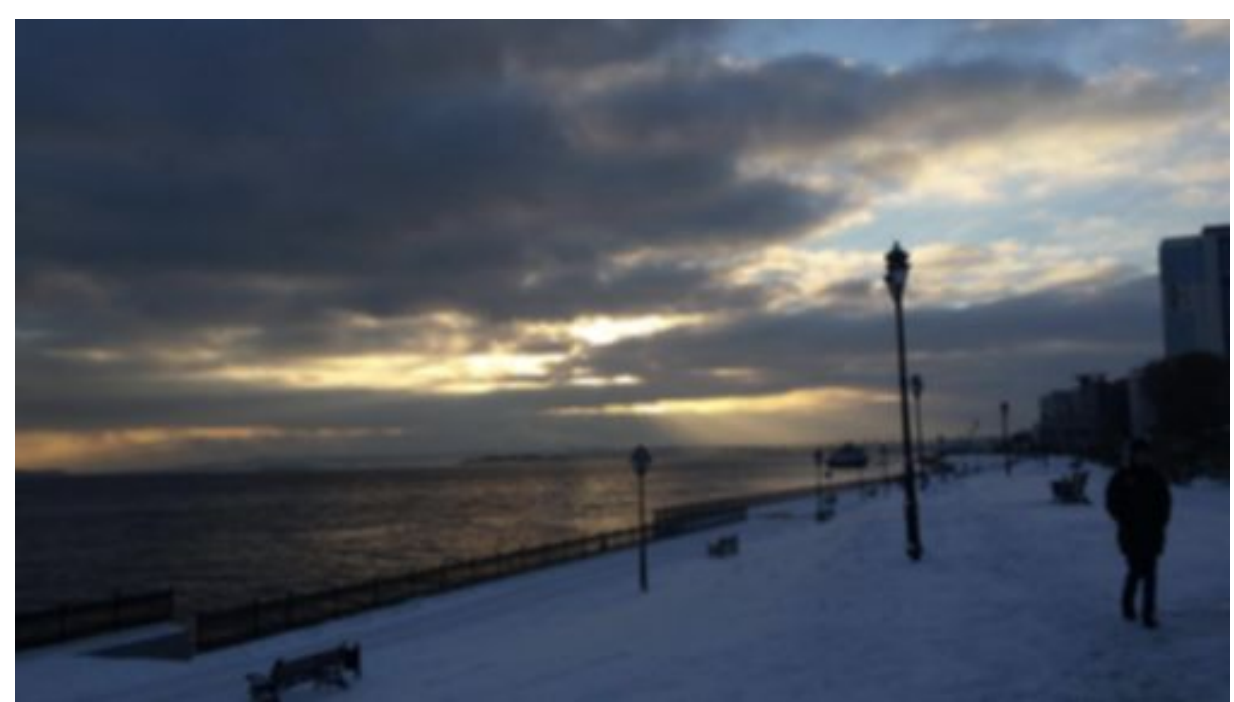




# Оператор Лапласа

- Также выделение границ
- Сразу по всем направлениям

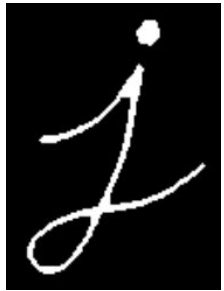
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



# Морфологические преобразования



Исходное изображение



Эрозия (cv2.erode), утоньшение



Дилатация (cv2.dilate), утолщение

# Морфологические преобразования 2



Исходное изображение



Открытие (`cv2.morphologyEx(img, cv2.MORPH_OPEN)`)  
Сначала эрозия потом дилатация



Заккрытие (`cv2.morphologyEx(img, cv2.MORPH_CLOSE)`)  
Сначала дилатация потом эрозия