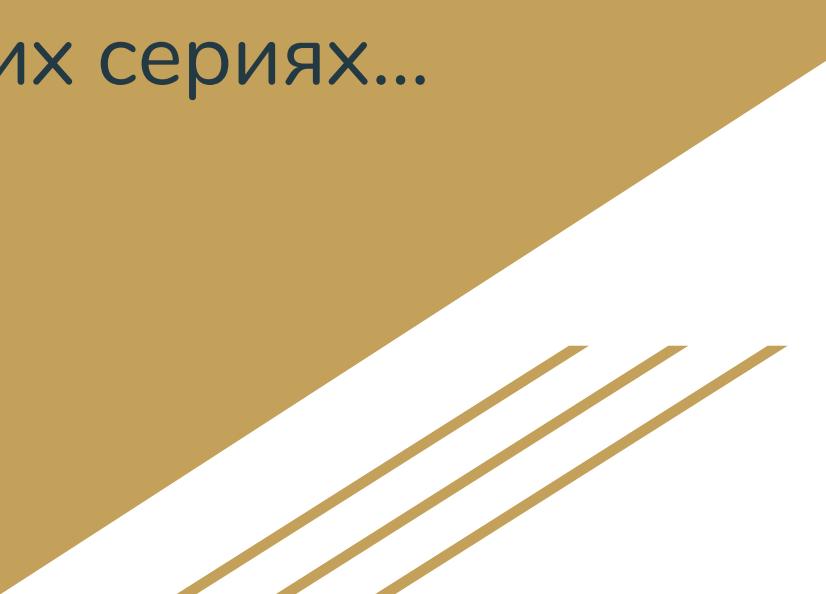


# Компьютерное зрение

Лекция 2. Фильтры и границы.  
Ресайзинг, фильтрация, Canny edge detector.

30.05.2020  
Руслан Алиев



В предыдущих сериях...

# Низкоуровневое компьютерное зрение

---

Простейшие манипуляции  
с изображениями

- Размер
- Цвет
- Яркость

Извлечение признаков

- Границы
- Ориентированные  
границы
- Сегментации



# Среднеуровневое компьютерное зрение

---

Картина <-> Картина

- Panoramas

Картина <-> Мир

- Стерео изображения
- Structured light
- Определение  
расстояния  
(фотограмметрия)

Картина <-> время

- Optical flow



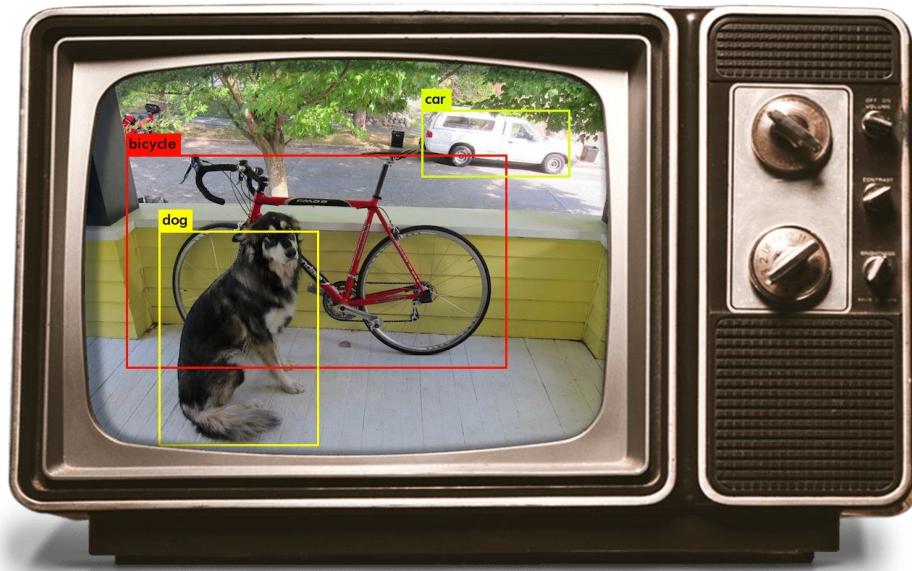
# Высокоуровневое компьютерное зрение

---

Появляется семантика!

## Приложения

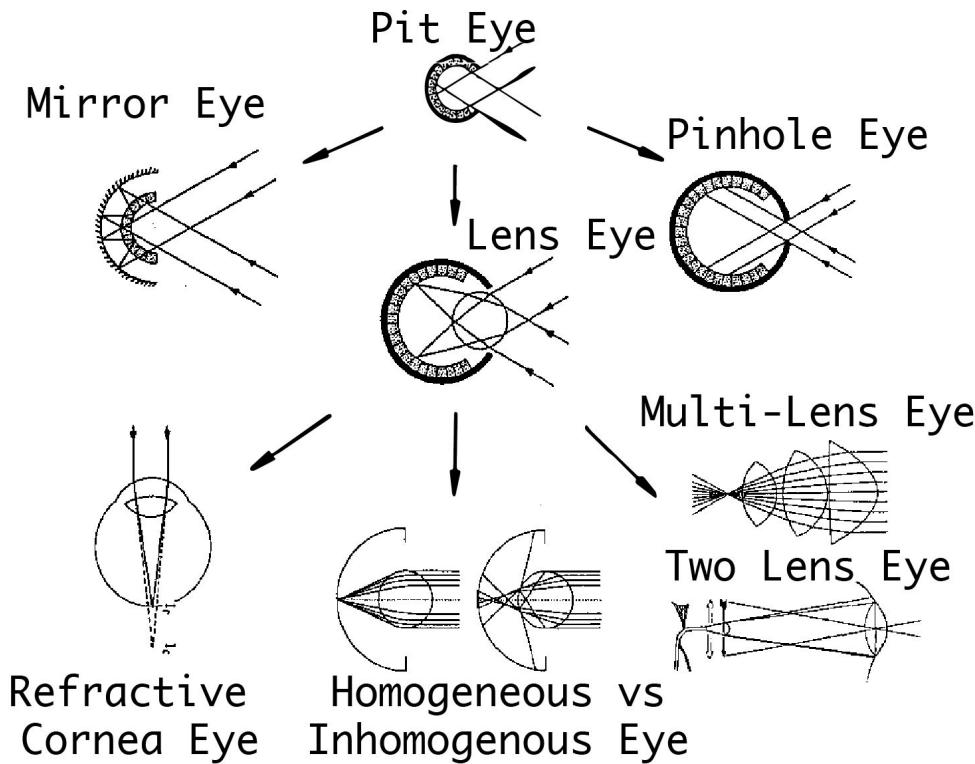
- Поиск изображений
- Беспилотные автомобили
- Наблюдение
- Диагностика заболеваний
- ...?



# Зрение - важный эволюционный инструмент

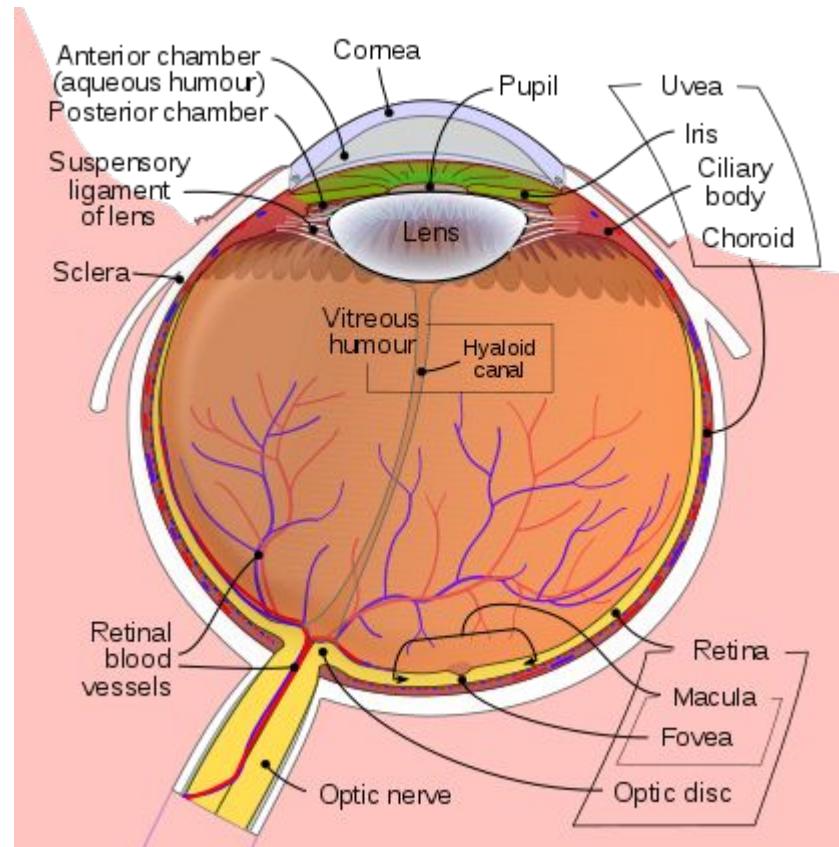
Простые глаза: есть у 28 из 33 классов животных

Сложные глаза: есть у 6 из 33 классов животных, но при этом 96% всех видов



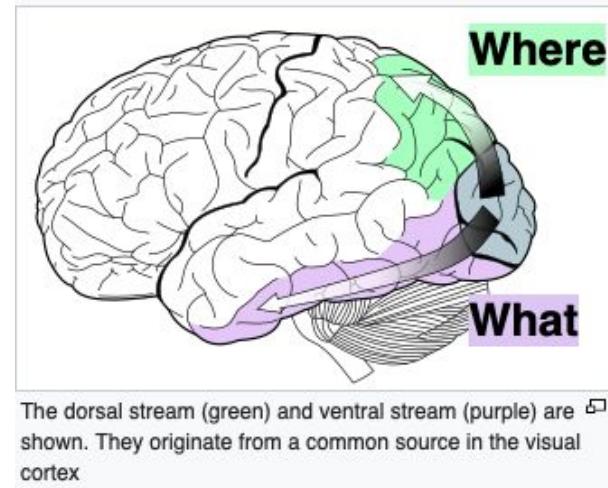
# Человеческие глаза

- Свет проходит через глаз
  - Хрусталик фокусирует свет
- Падает на сетчатку
- Возбуждает палочки и колбочки
- Сигнал передает через оптический нерв в визуальную кору



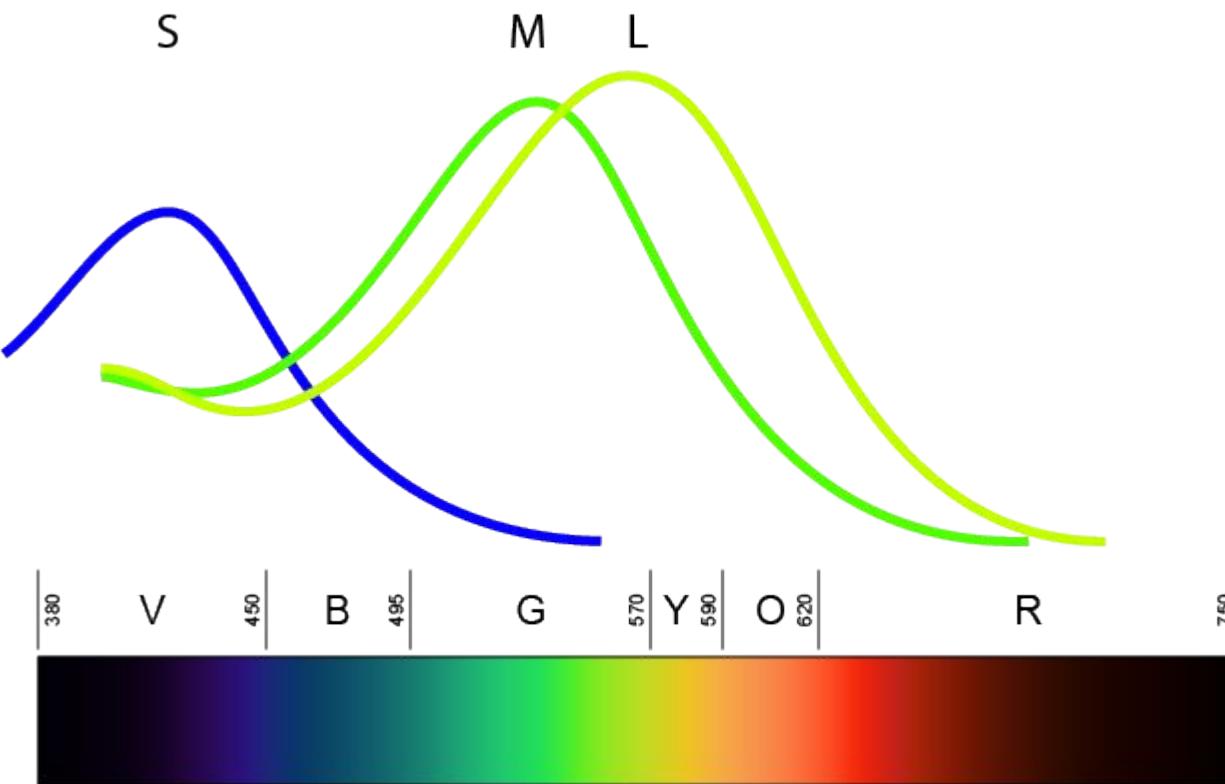
# Вентральное/дорсальное разделение

- Вентральная система: распознавание объектов
- Дорсальная: зрение связанное с движением и планированием
- Большая часть зрения неосознаваема



# Палочки и колбочки (свет!)

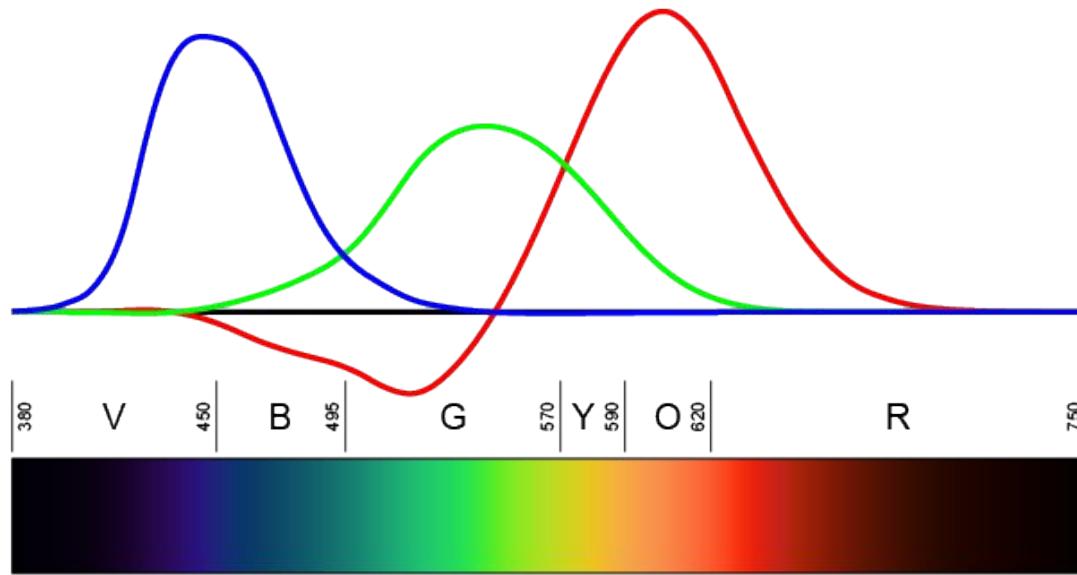
---



# CIE 1931

---

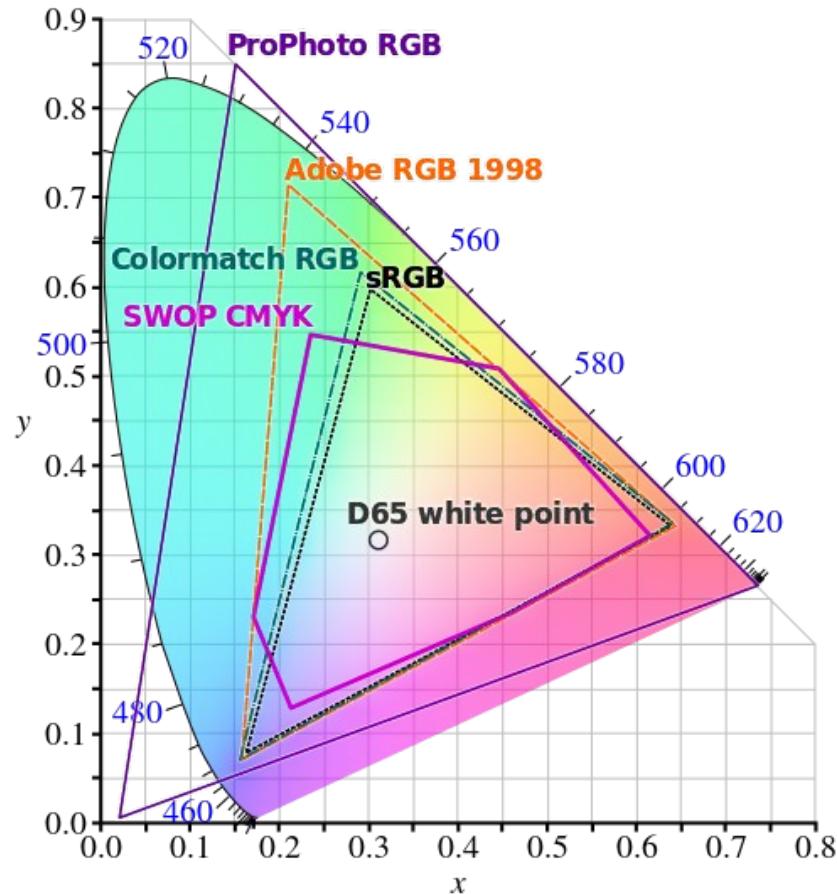
- Любой цвет как комбинация 3 базовых цветов
- Участники выбирали очень похожие комбинации



# Цветовые пространства

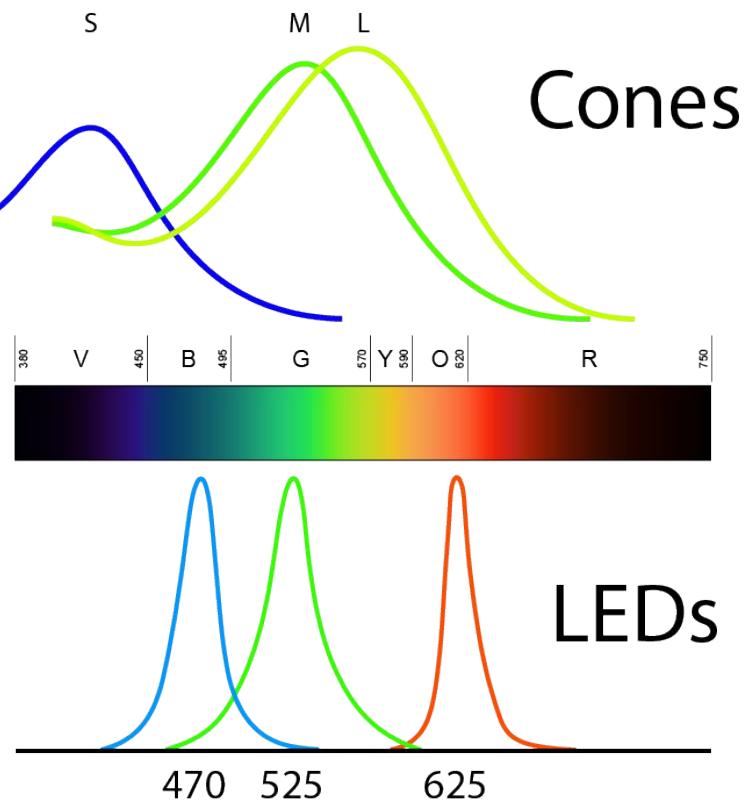
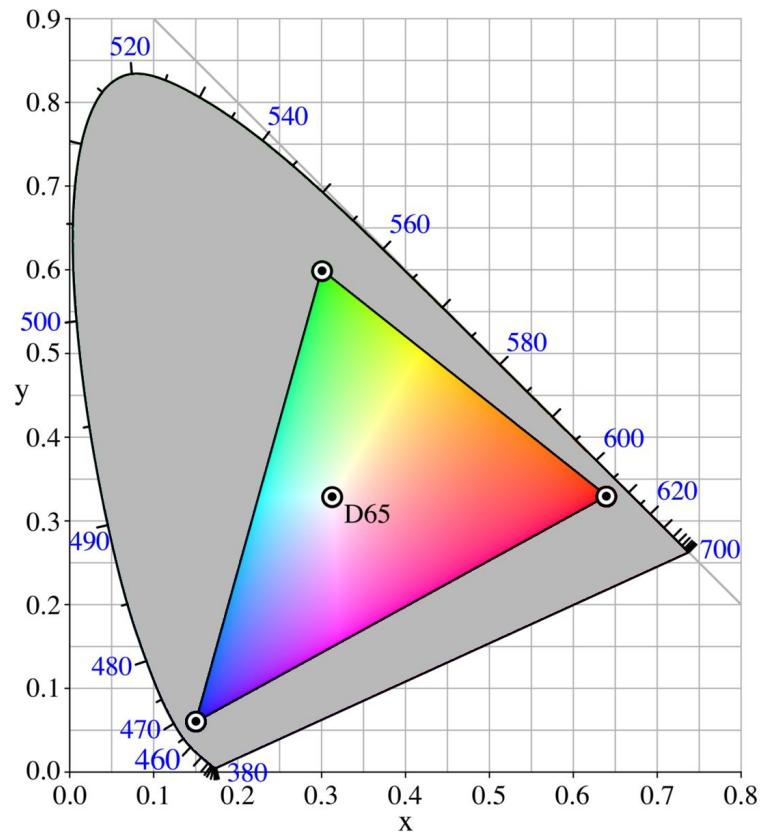
---

- Определяем цветовое пространство, выбирая базовые цвета
- Все возможные цвета - это гамма
- Не все цвета представимы



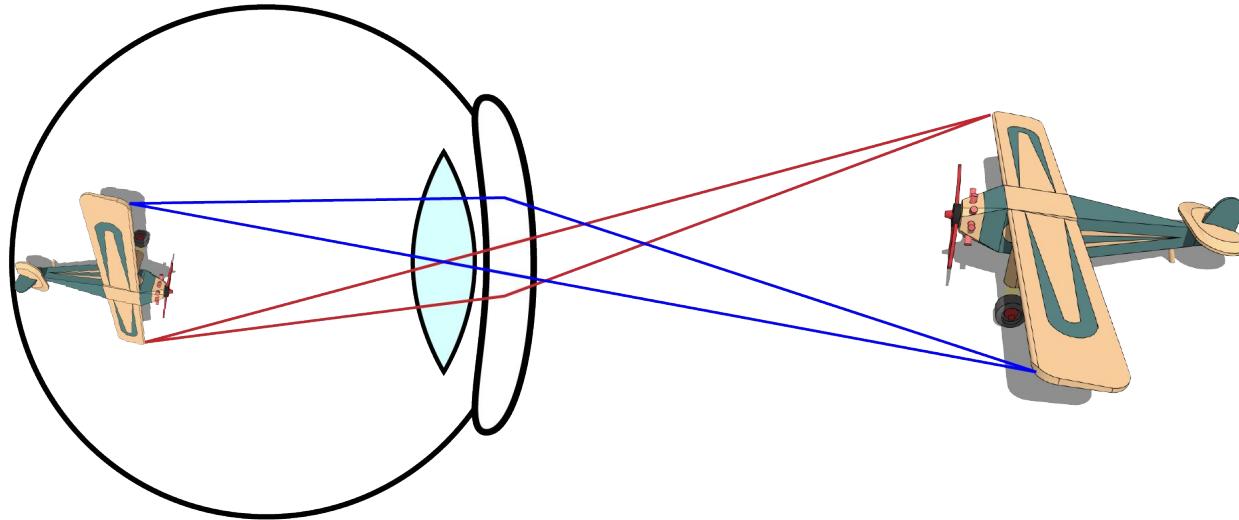
# sRGB (1996, HP+Microsoft)

---



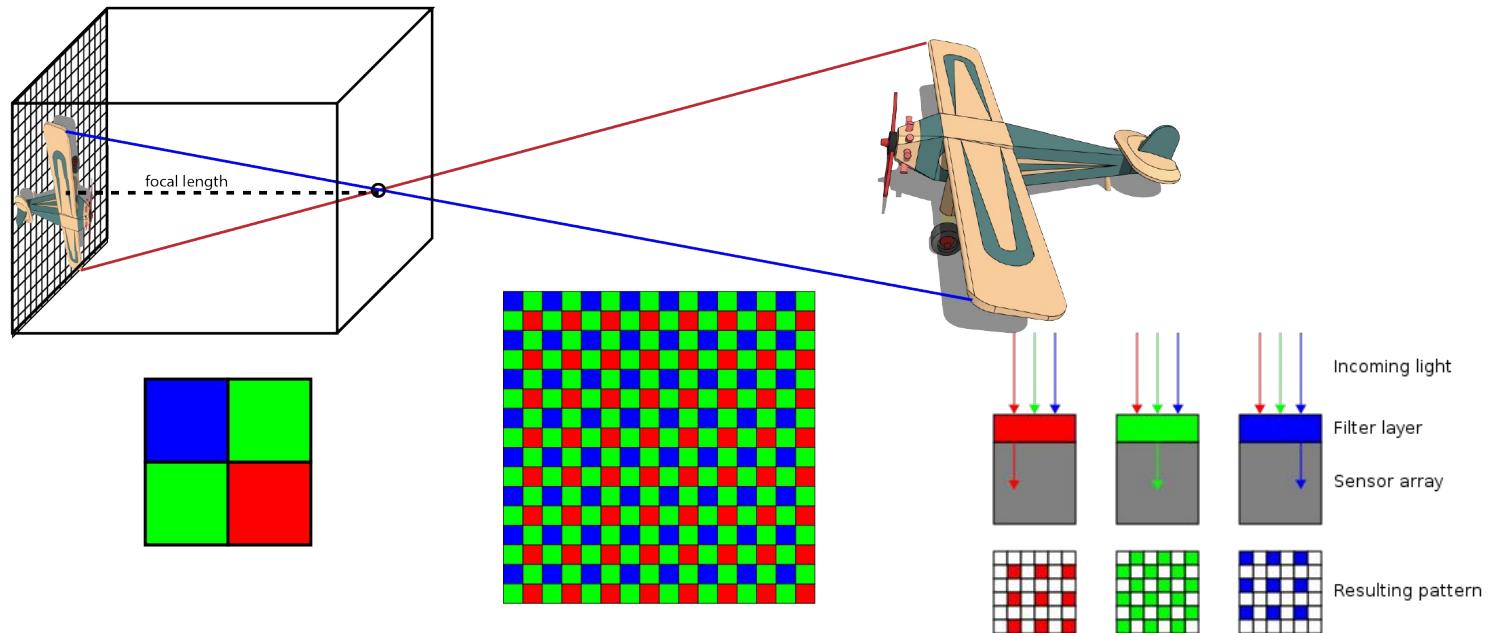
# Глаза: проекция на сетчатку

---



# Камера: проекция на светочувствительную матрицу

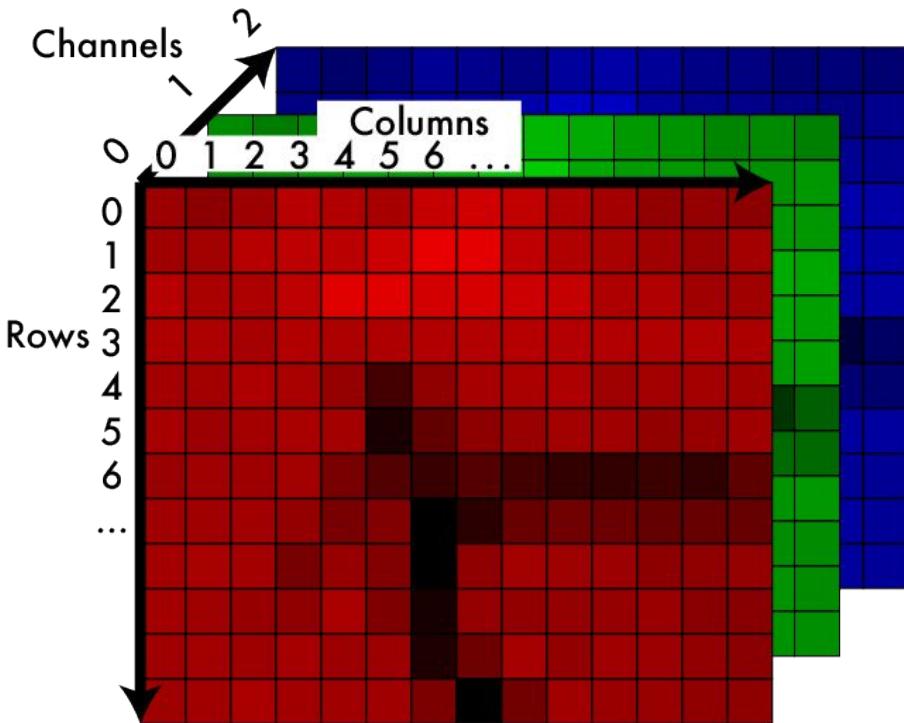
---



# Изображение: трехмерный тензор

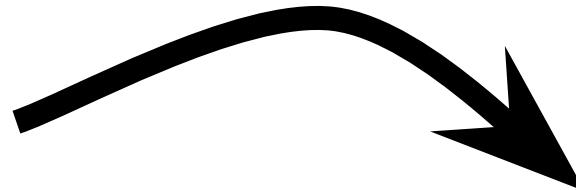
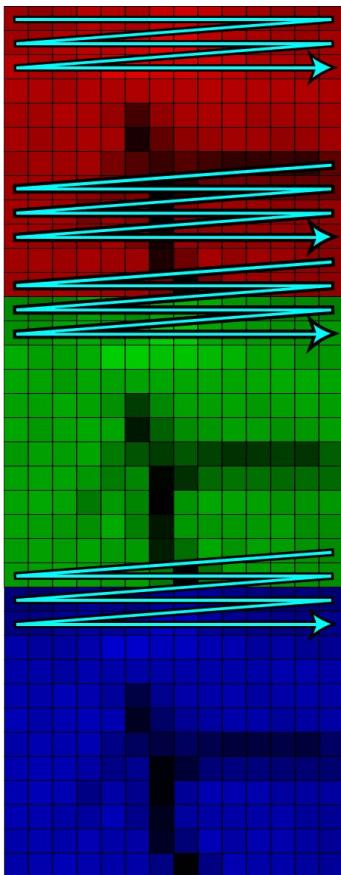
Нумерация пикселей:

- Размер:  $W \times H \times C$
  - Индекс:  $(x, y, z)$ 
    - Столбец  $x$ , строка  $y$ , канал  $z$
  - Выбор индексирования опционален
- numpy:
- Строка, столбец, канал

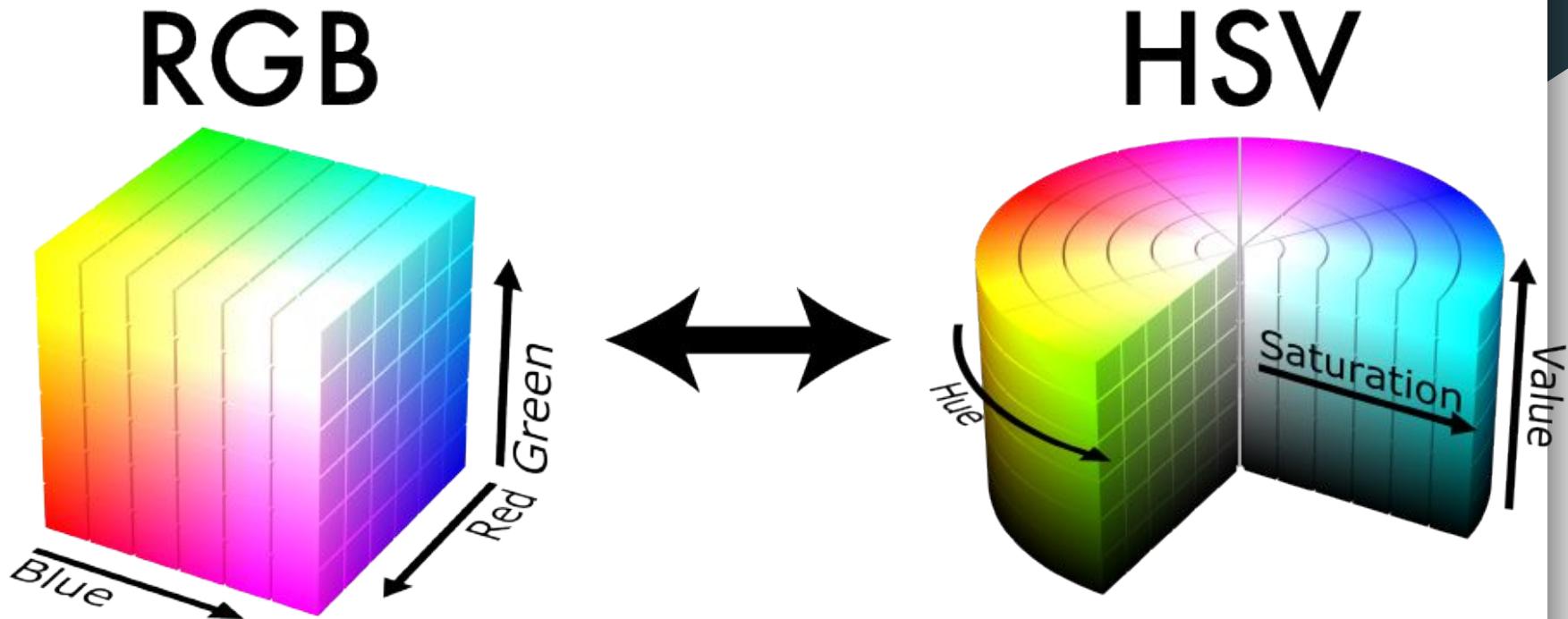


# CHW для хранения в 1д массиве

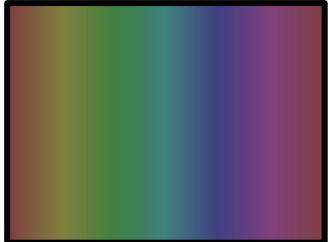
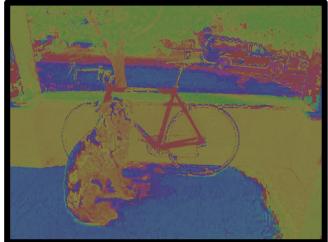
---



HSV - больше соответствует человеческому восприятию



# Можно делать фокусы



# Nearest neighbor интерполяций

---

$f(x,y,z) = \text{Im}(\text{round}(x), \text{round}(y), z)$

- Лестничные артефакты
- $z$  - все еще целое число



# Билинейная интерполяция

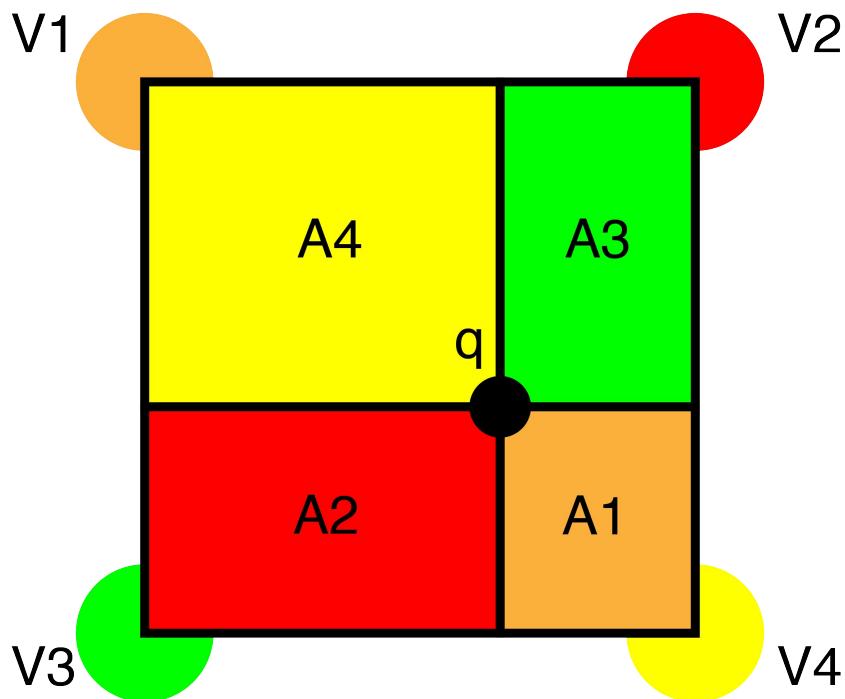
В этот раз будем искать ближайших соседей по прямоугольнику

Идея такая же: взвешенная сумма на основе противолежащих прямоугольников

$$Q = V1 * A1 + V2 * A2 + V3 * A3 + V4 * A4$$

Все еще нужно нормализовывать!

Или не нужно? 🤔



## Билинейная интерполяция

$$q_1 = V1 * d2 + V2 * d1$$

$$q_2 = V3 * d2 + V4 * d1$$

$$q = q_1 * d4 + q_2 * d3$$

$$q = (V1 * d2 + V2 * d1) * d4 + (V3 * d2 + V4 * d1) * d3$$

$$q = V1 * d2 * d4 + V2 * d1 * d4 + V3 * d2 * d3 + V4 * d1 * d3$$

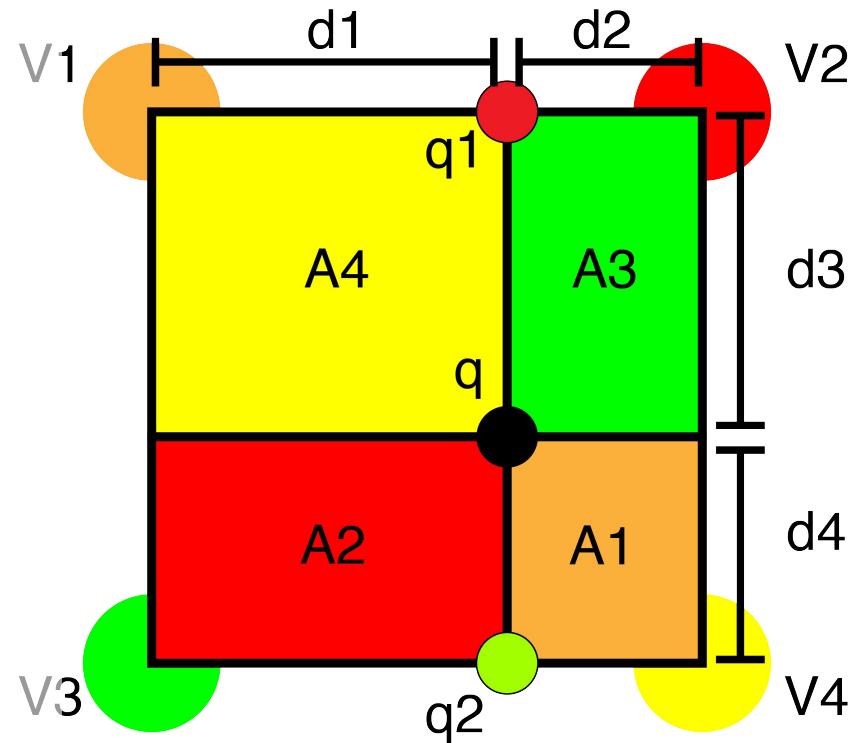
Помним:

$$A1 = d2 * d4$$

$$A2 = d1 * d4$$

$$A3 = d2 * d3$$

$$A4 = d1 * d3$$

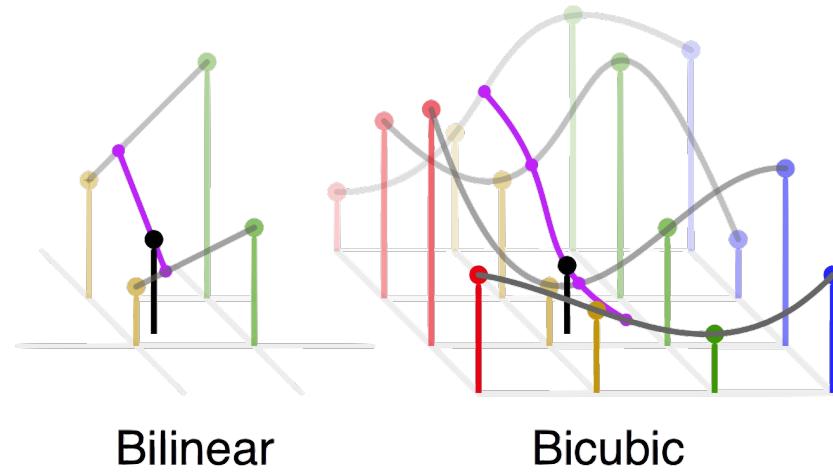


$$q = V1 * A1 + V2 * A2 + V3 * A3 + V4 * A4 \text{ (ура!)}$$

# Бикубическая интерполяция

---

- Кубическая интерполяция  
4 кубических  
интерполяций
- Более гладкая, нету  
артефактов “звездочек”
- Смотрим на 16 соседей
- Полином 3 степени:
  - $f(x) = a + bx + cx^2 + dx^3$

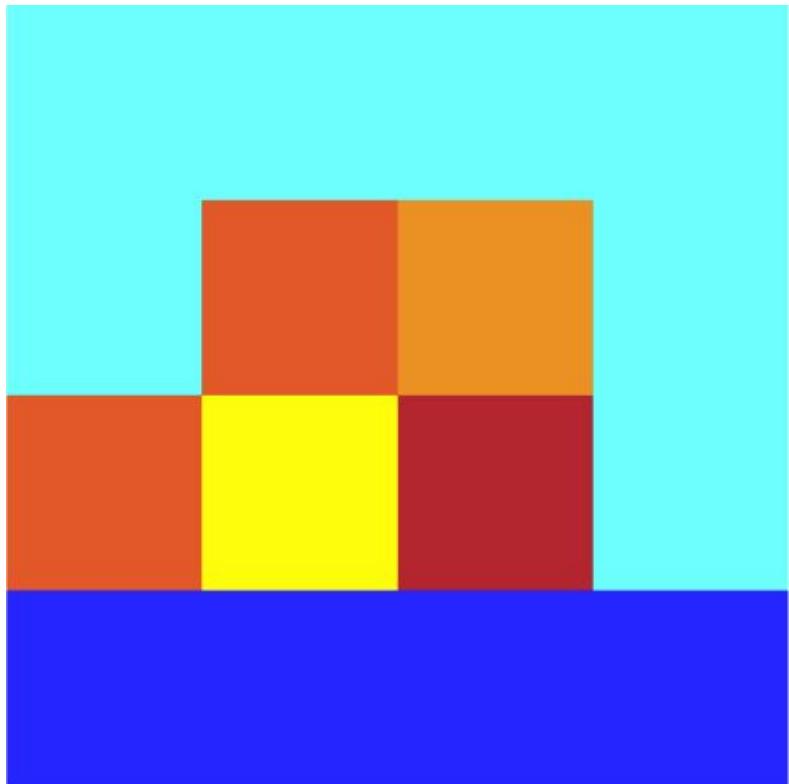


# Ресайзинг

# Ресайзинг

---

Допустим мы хотим  
увеличить размер этого  
изображения  
(это закат)

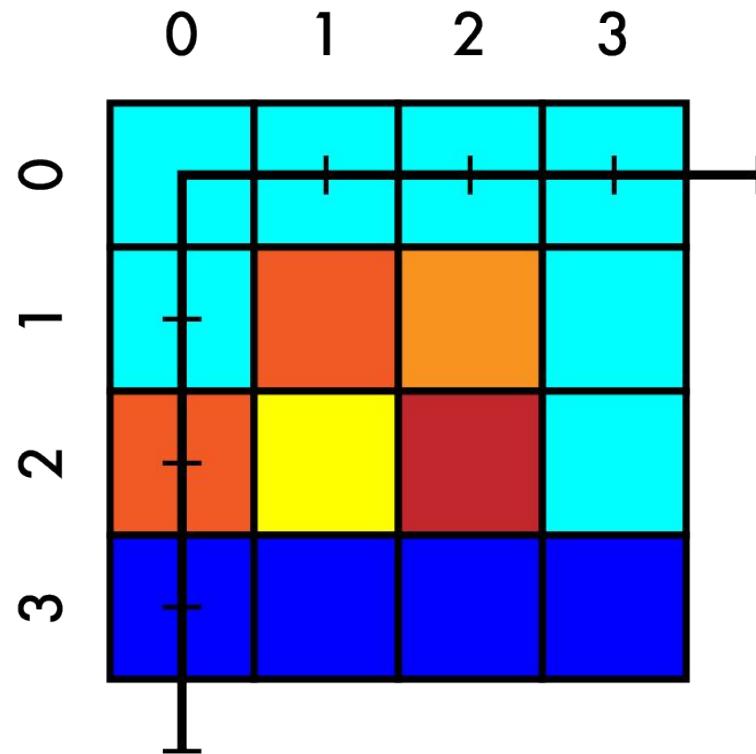


# Ресайзинг

---

Допустим мы хотим  
увеличить размер этого  
изображения  
(это закат)

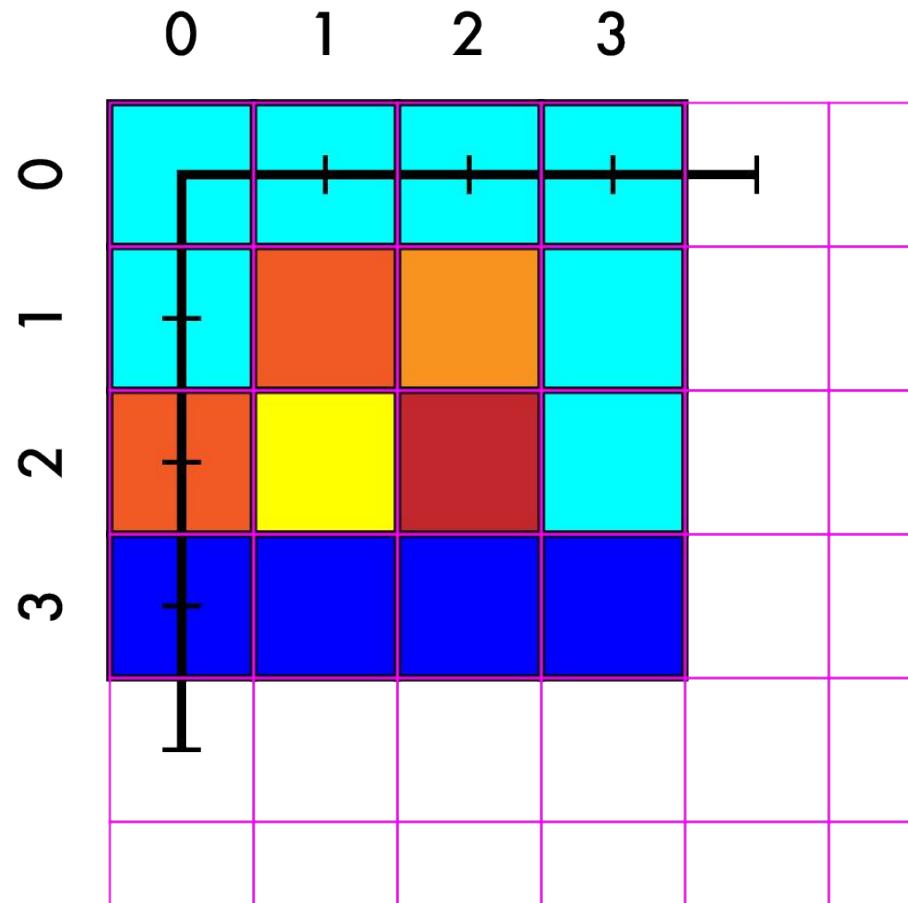
$4 \times 4 \rightarrow 7 \times 7$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

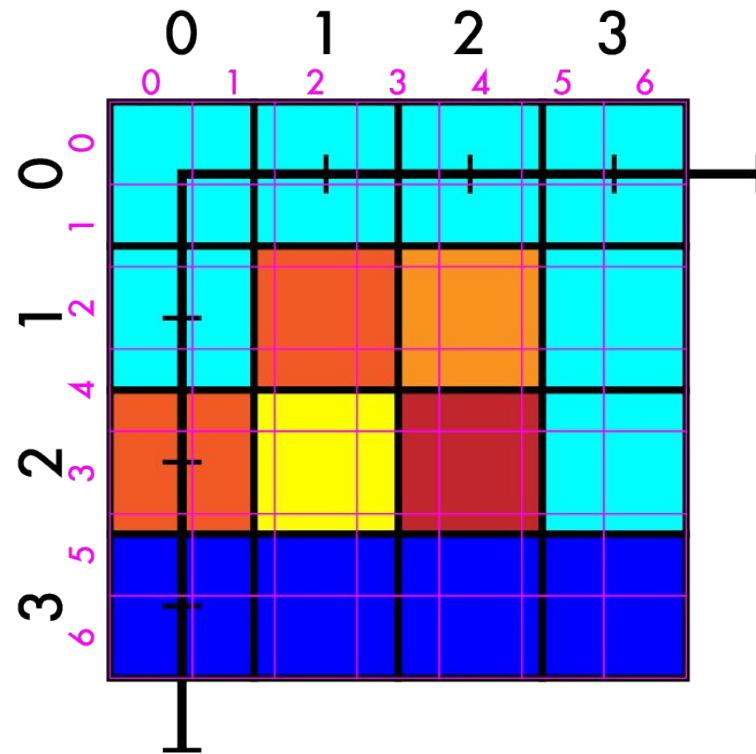
---

- Создать новое изображение



# Ресайзинг 4x4 -> 7x7

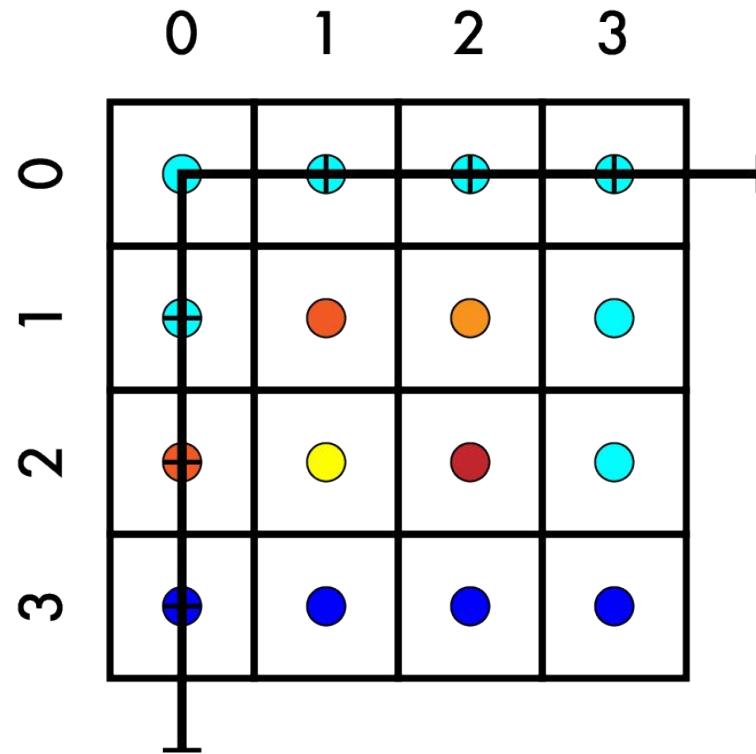
- Создать новое изображение
  - Сматчить координаты



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

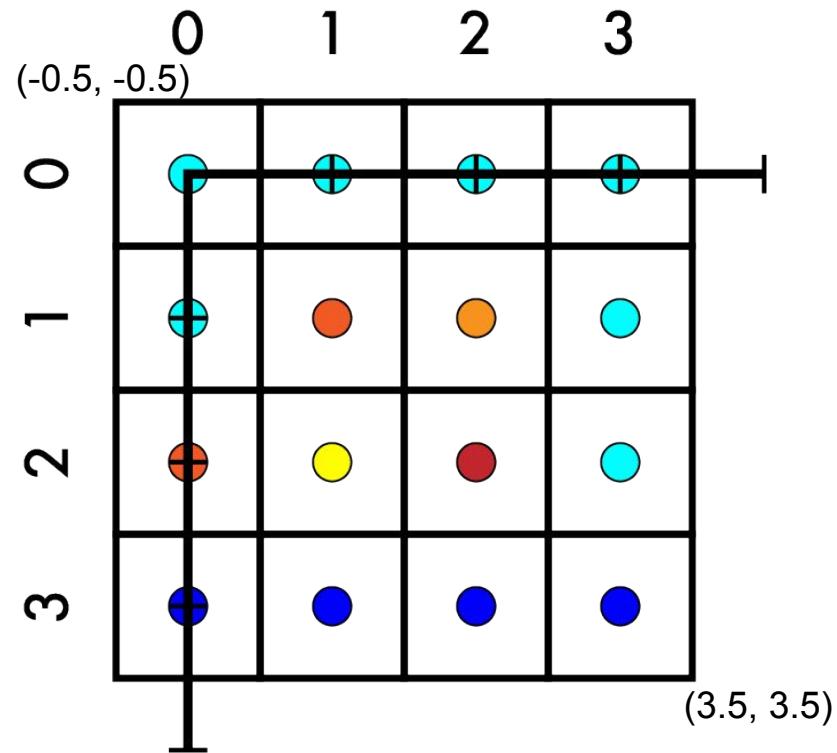
- Создать новое изображение
- Сматчить координаты



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

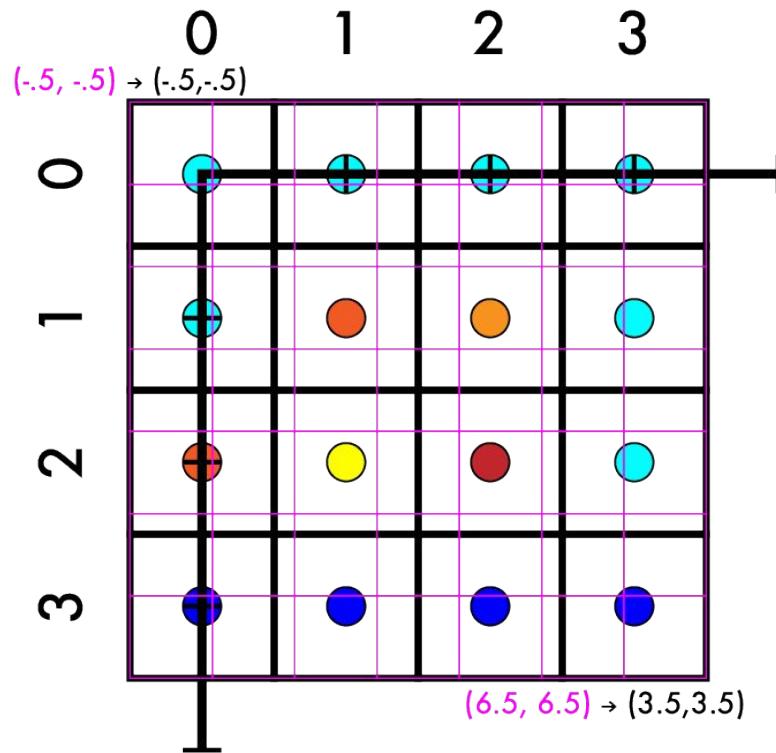
---

- Создать новое изображение
- Сматчить координаты



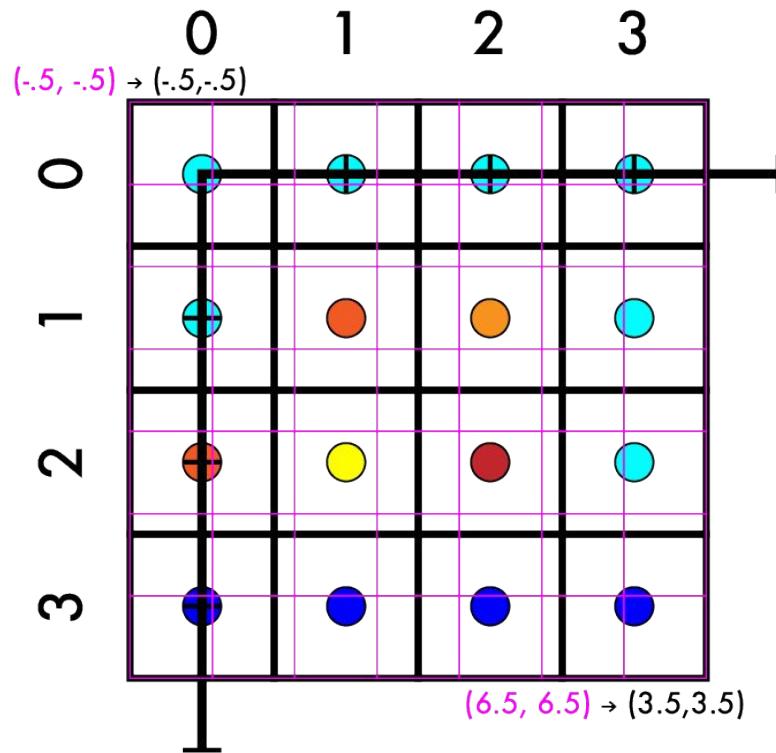
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -.5 + b = -.5$
  - $a * 6.5 + b = 3.5$



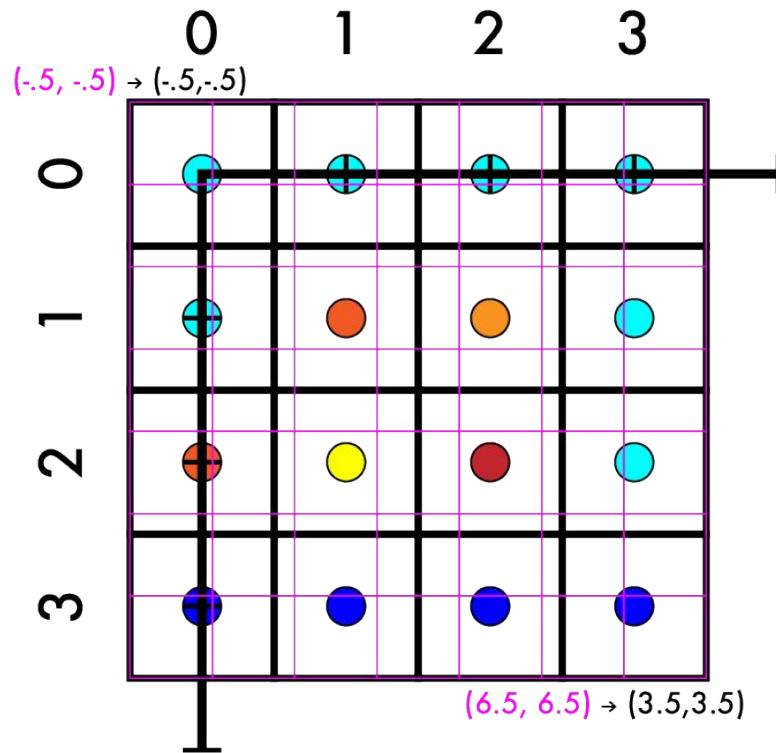
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -.5 + b = -.5$
  - $a * 6.5 + b = 3.5$ 
    - $a * 7 = 4$



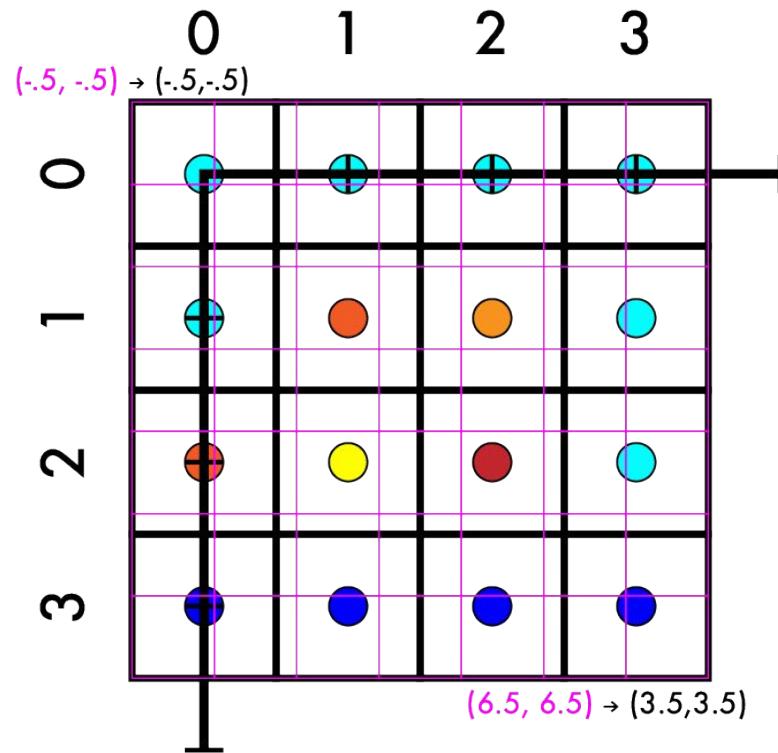
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -0.5 + b = -0.5$
  - $a * 6.5 + b = 3.5$ 
    - $a * 7 = 4$
    - $a = 4/7$



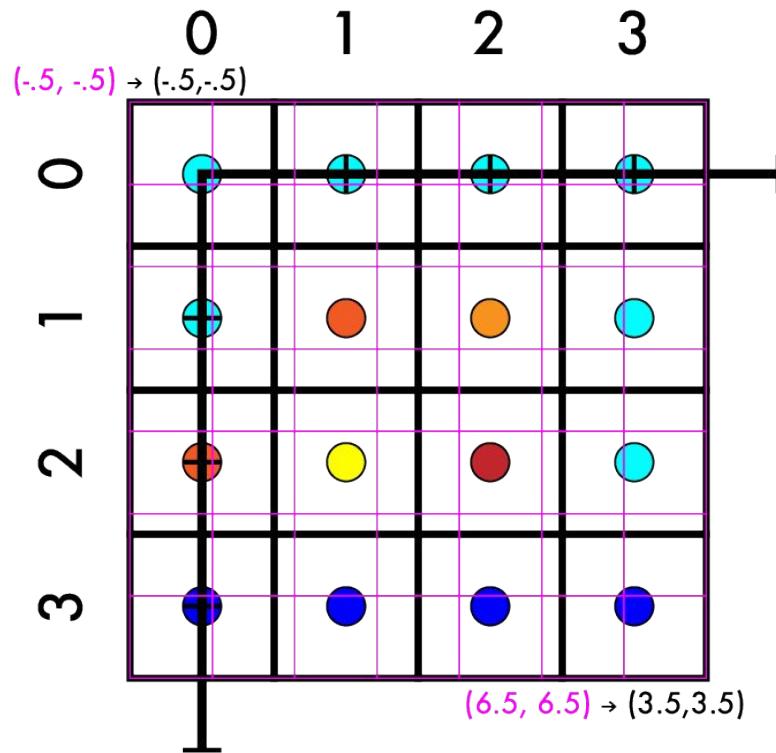
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -.5 + b = -.5$
  - $a * 6.5 + b = 3.5$
  - $a = 4/7$



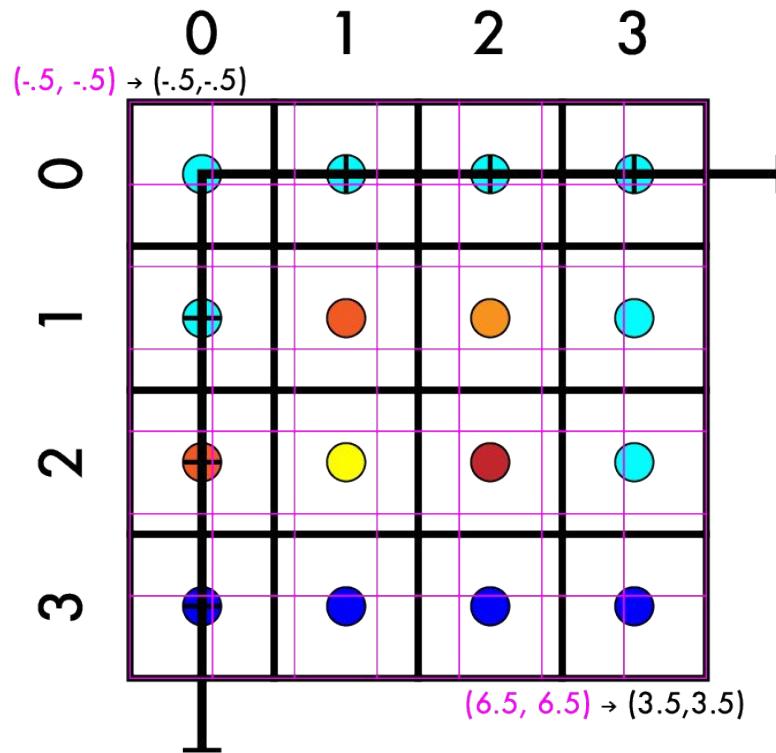
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -.5 + b = -.5$
  - $a * 6.5 + b = 3.5$
  - $a = 4/7$ 
    - $a * -.5 + b = -.5$



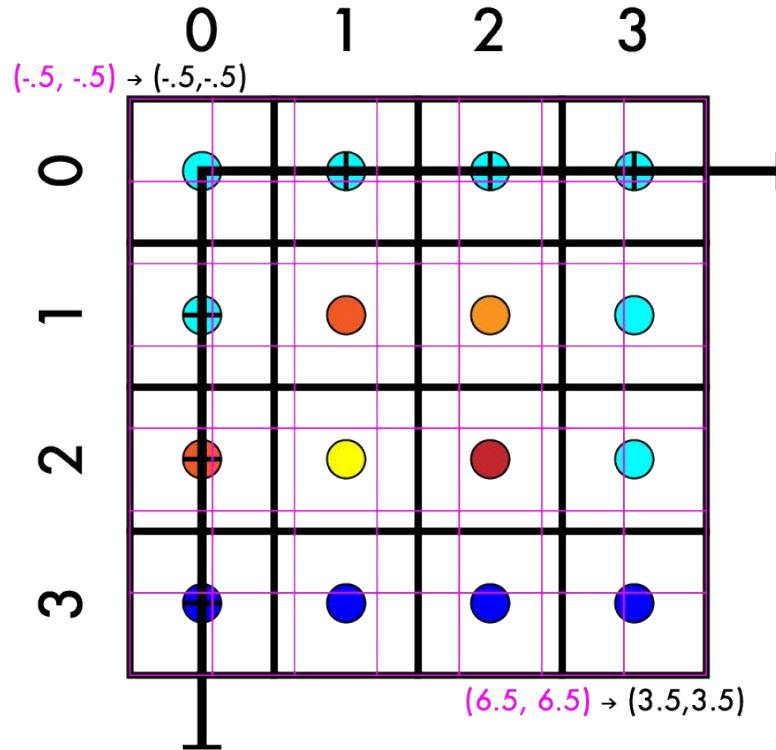
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -0.5 + b = -0.5$
  - $a * 6.5 + b = 3.5$
  - $a = 4/7$ 
    - $a * -0.5 + b = -0.5$
    - $4/7 * -1/2 + b = -1/2$



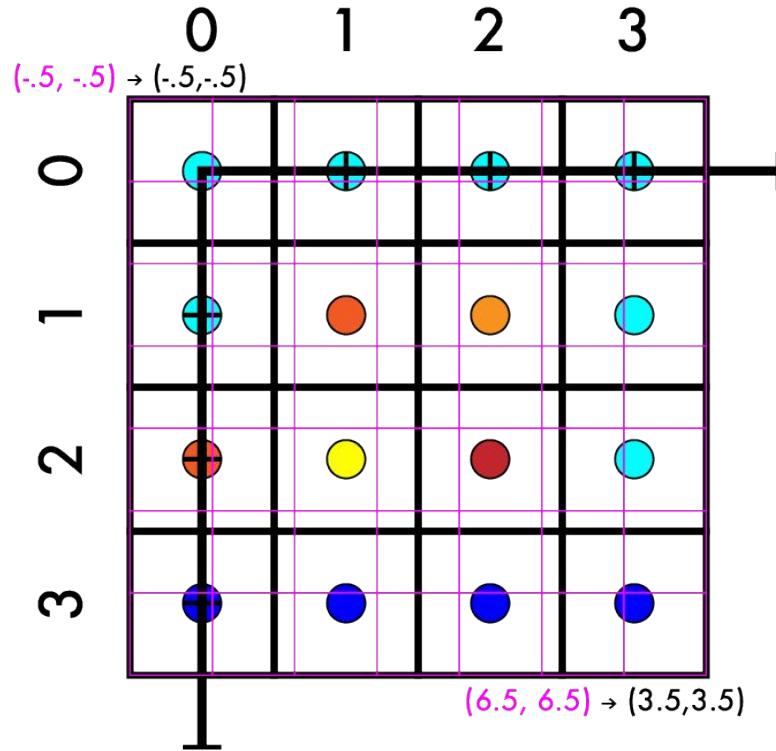
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -.5 + b = -.5$
  - $a * 6.5 + b = 3.5$
  - $a = 4/7$ 
    - $a * -.5 + b = -.5$
    - $4/7 * -1/2 + b = -1/2$
    - $-4/14 + b = -7/14$



# Ресайзинг 4x4 -> 7x7

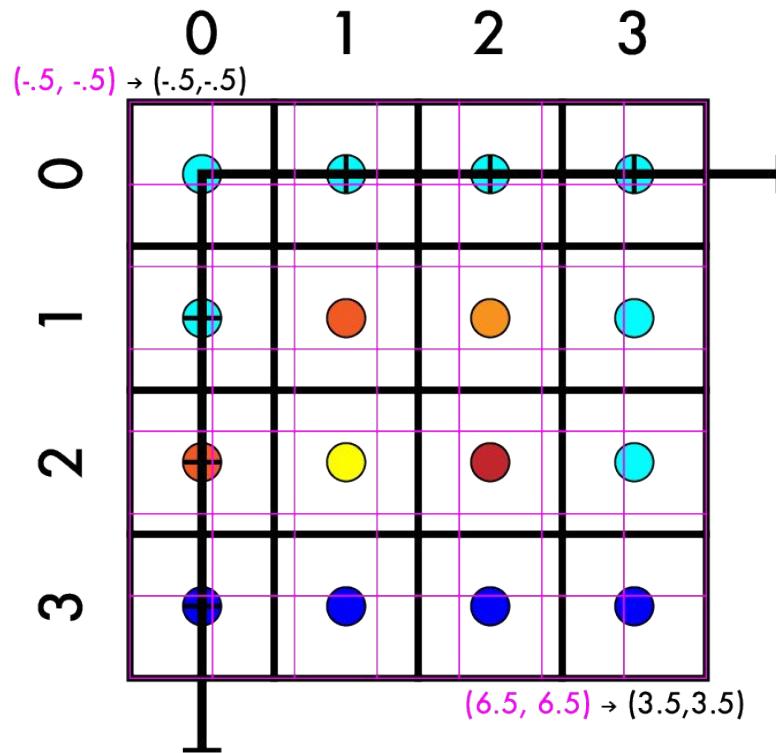
- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -0.5 + b = -0.5$
  - $a * 6.5 + b = 3.5$
  - $a = 4/7$ 
    - $a * -0.5 + b = -0.5$
    - $4/7 * -1/2 + b = -1/2$
    - $-4/14 + b = -7/14$
    - $b = -3/14$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

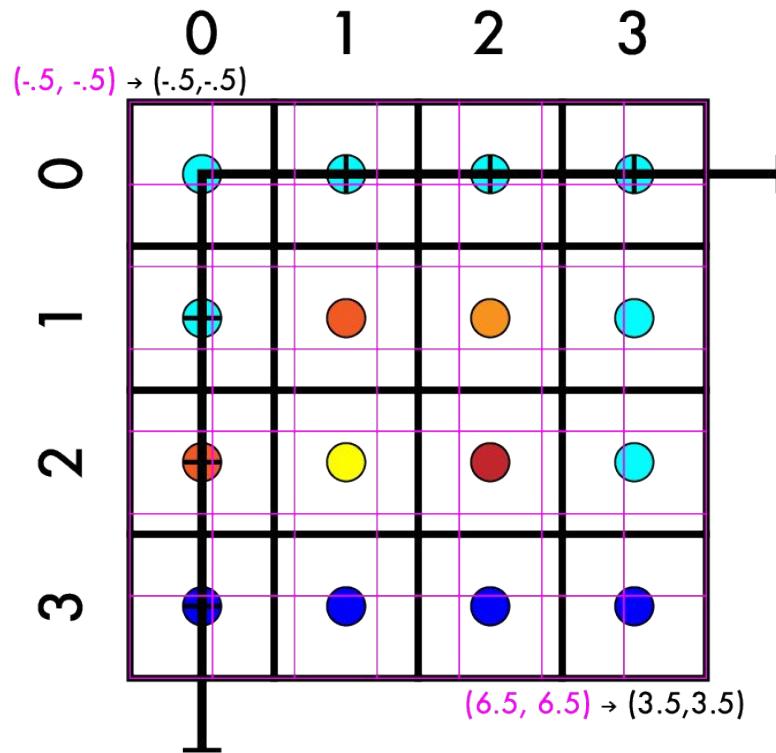
- Создать новое изображение
- Сматчить координаты
  - Система уравнений
  - $aX + b = Y$
  - $a * -0.5 + b = -0.5$
  - $a * 6.5 + b = 3.5$
  - $a = 4/7$
  - $b = -3/14$



# Ресайзинг 4x4 -> 7x7

---

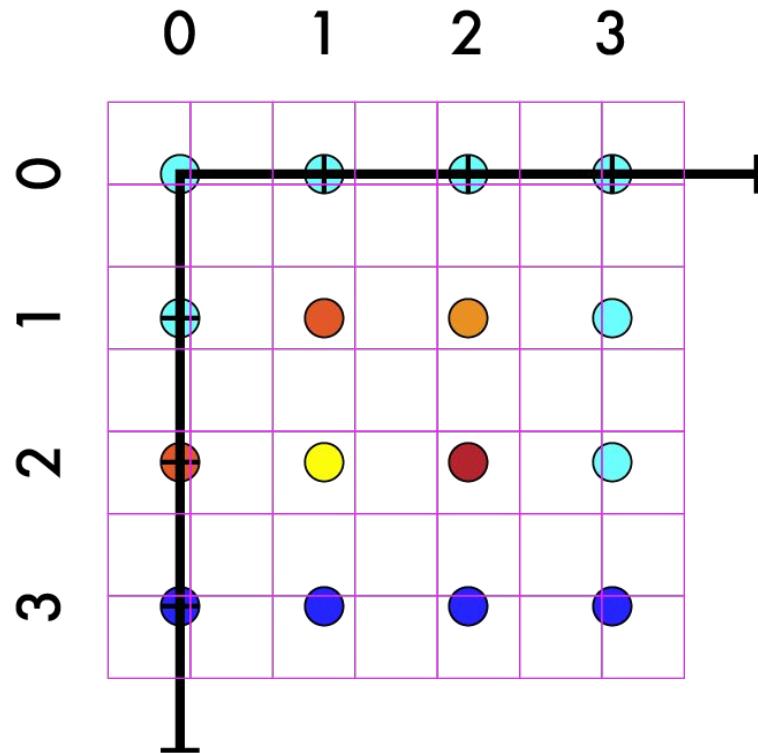
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$



# Ресайзинг 4x4 -> 7x7

---

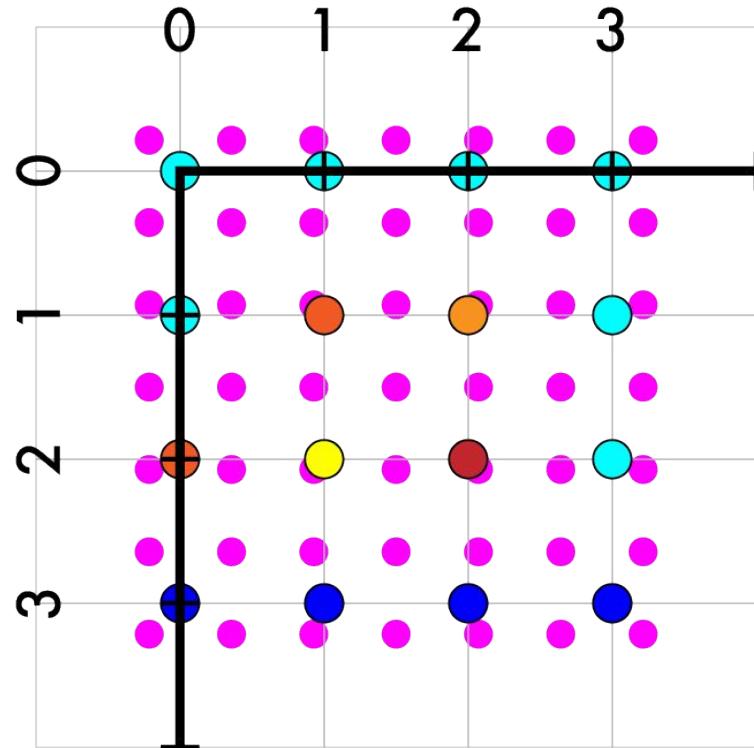
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

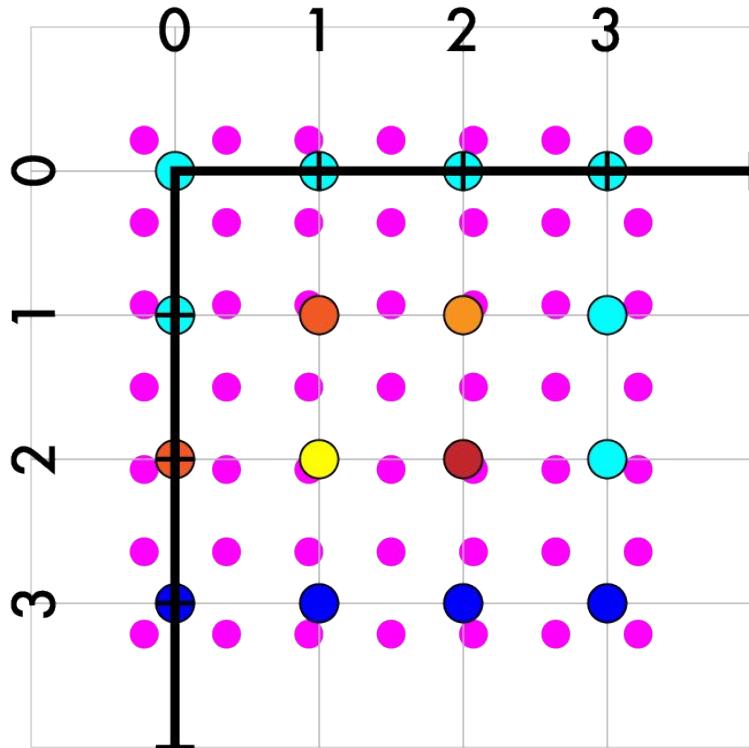
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

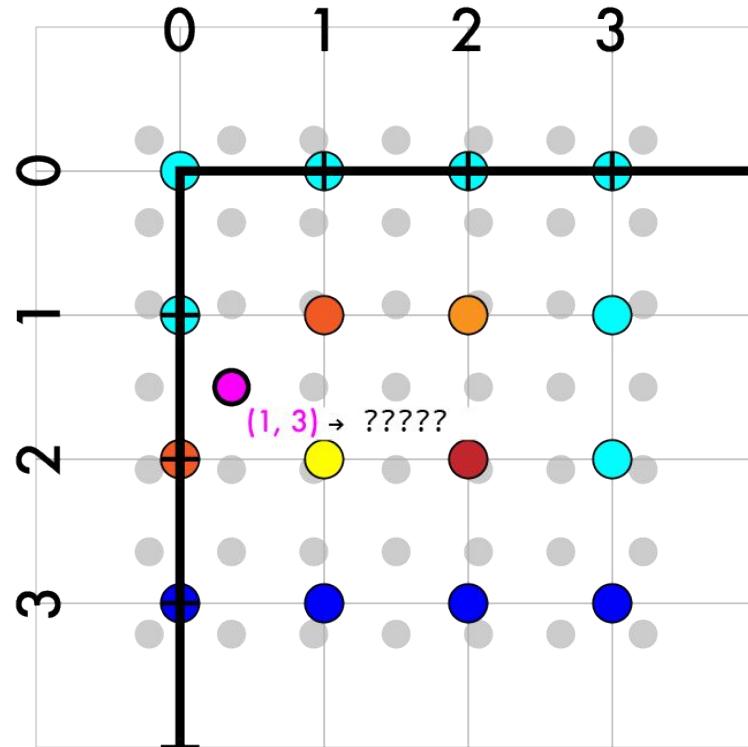
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

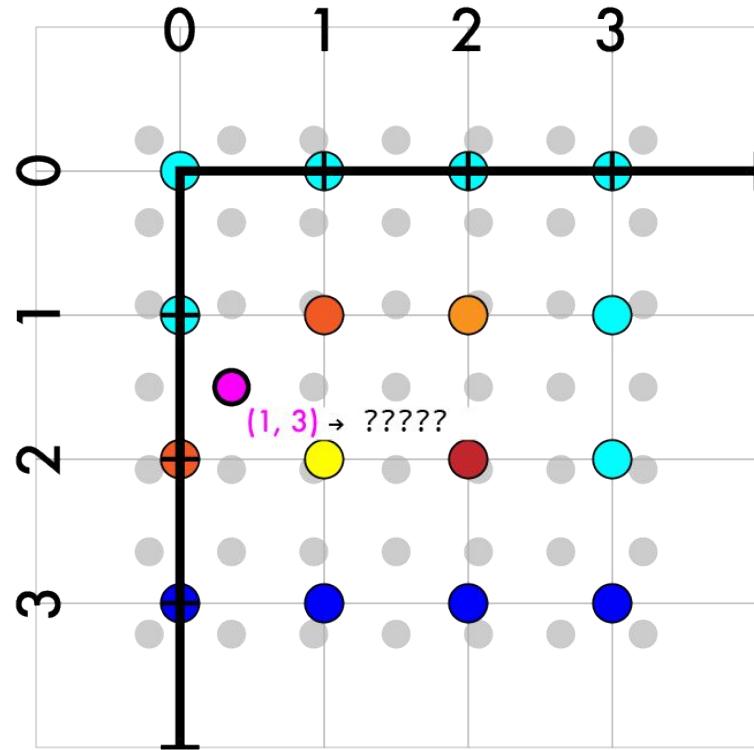
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3)$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

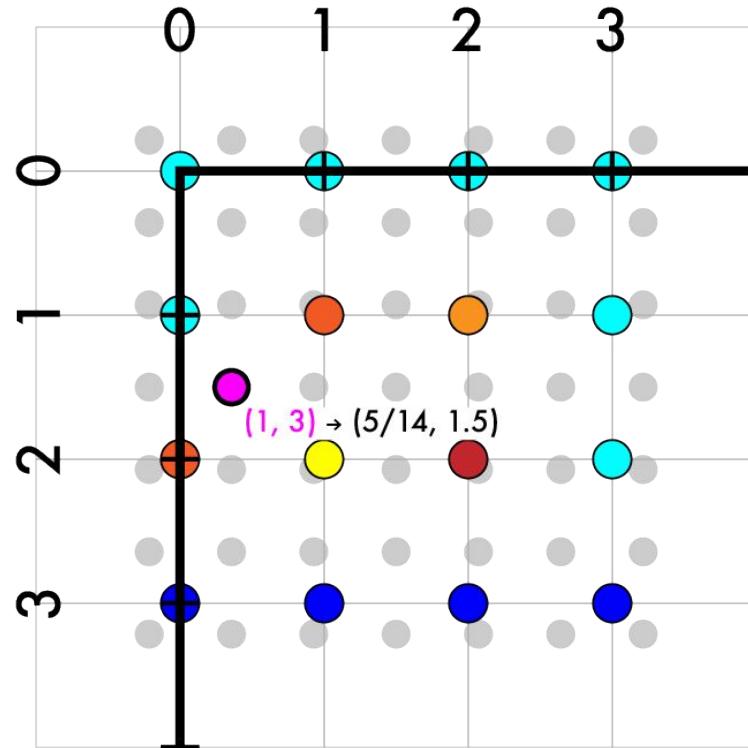
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3)$
  - $4/7 * 1 - 3/14$
  - $4/7 * 3 - 3/14$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

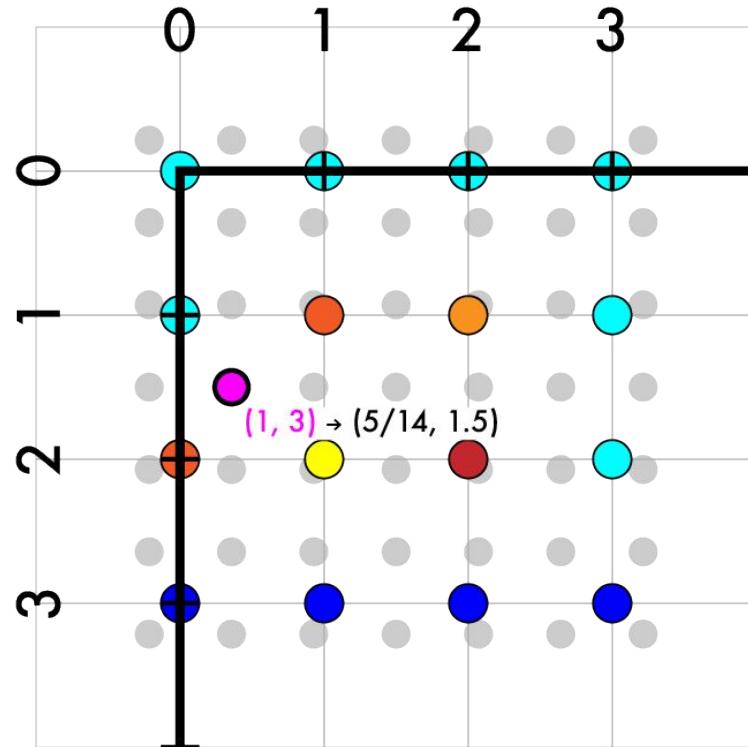
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3)$
  - $4/7 * 1 - 3/14$
  - $4/7 * 3 - 3/14$
  - $(5/14, 3/2)$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

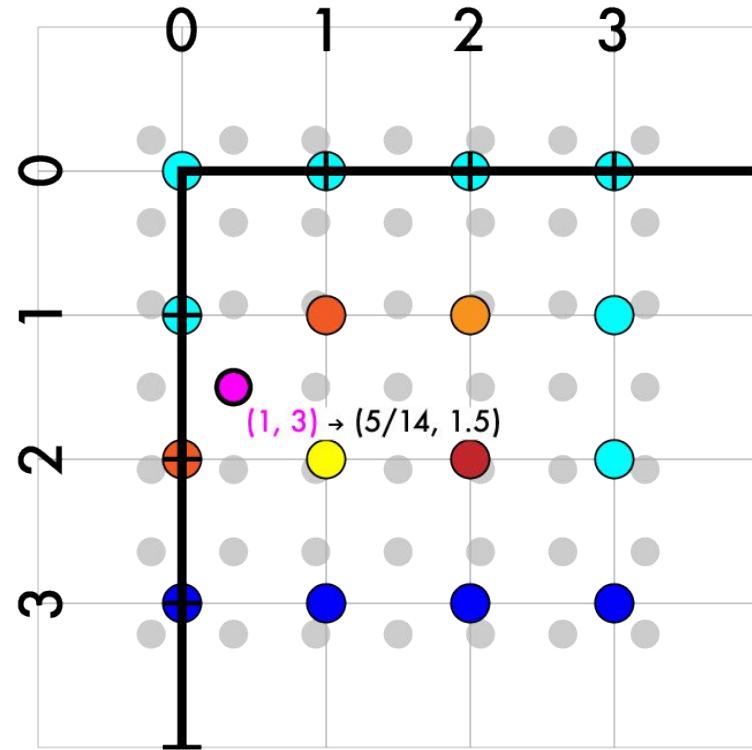
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

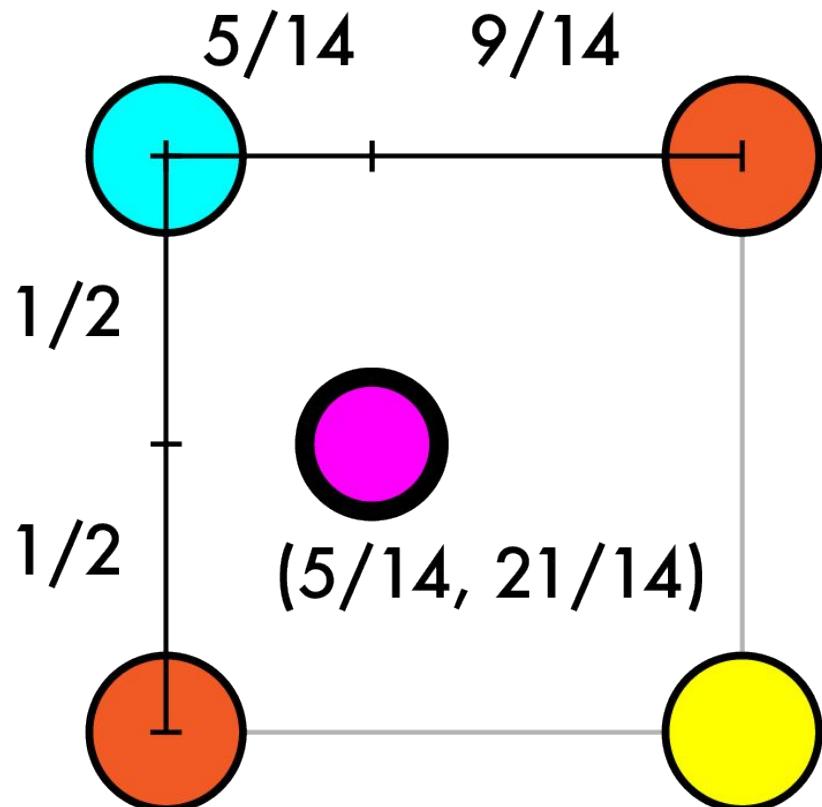
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

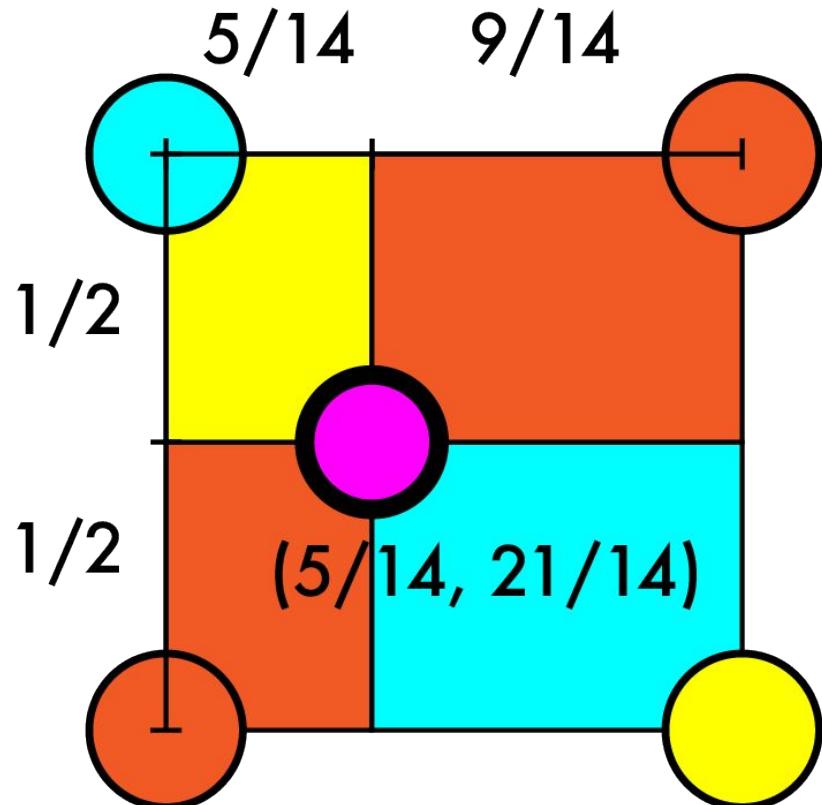
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

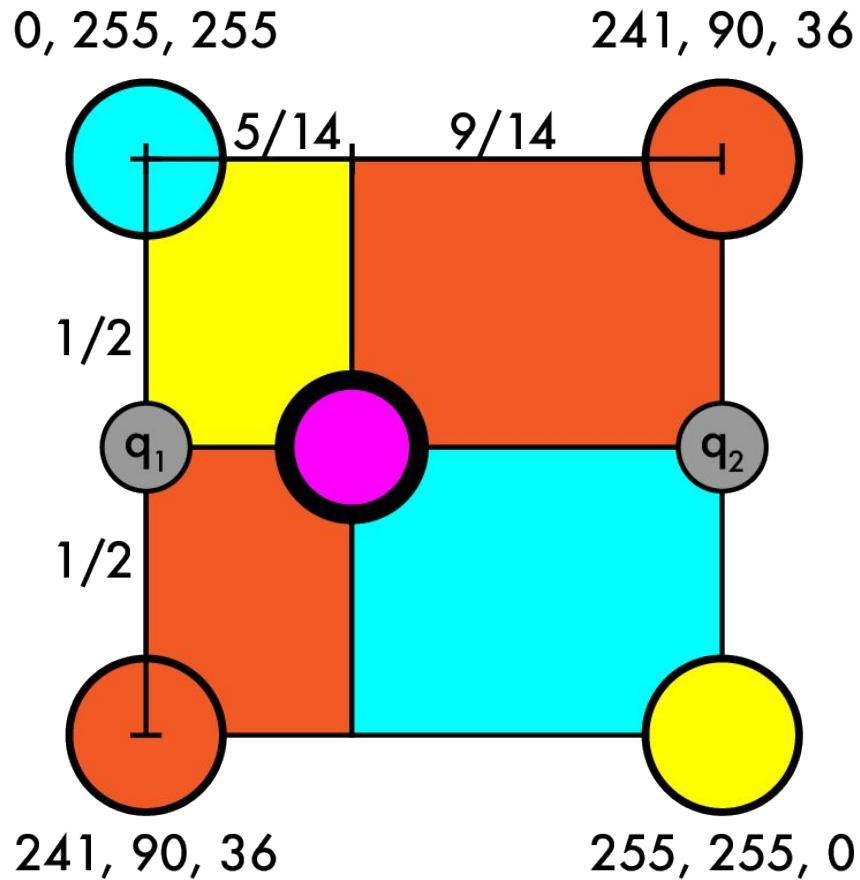
---

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения



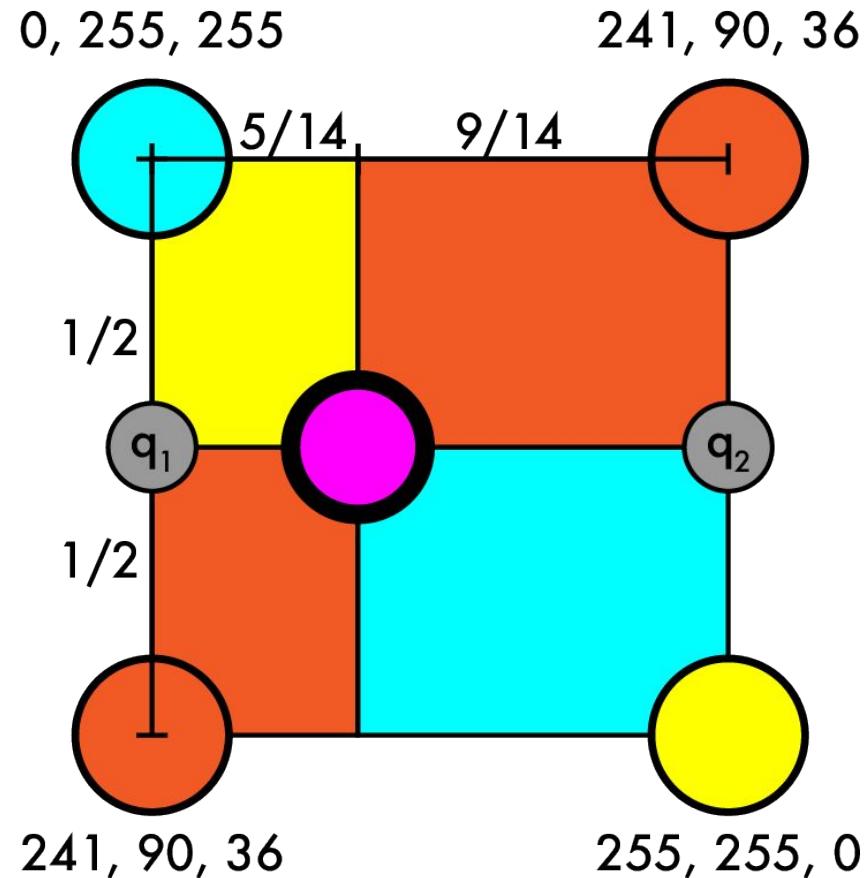
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - Прямоугольники
    - $q_1, q_2$



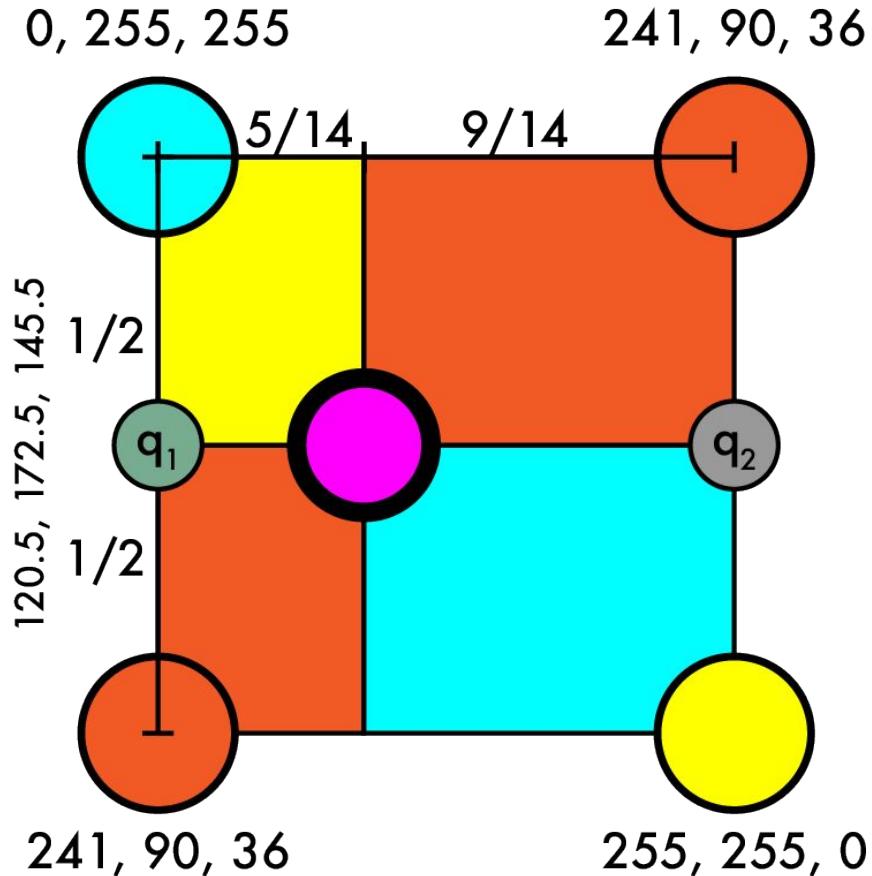
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q_1 = r1, g1, b1$
    - $r1 = .5*0 + .5*241$
    - $g1 = .5*255 + .5*90$
    - $b1 = .5*241 + .5*36$



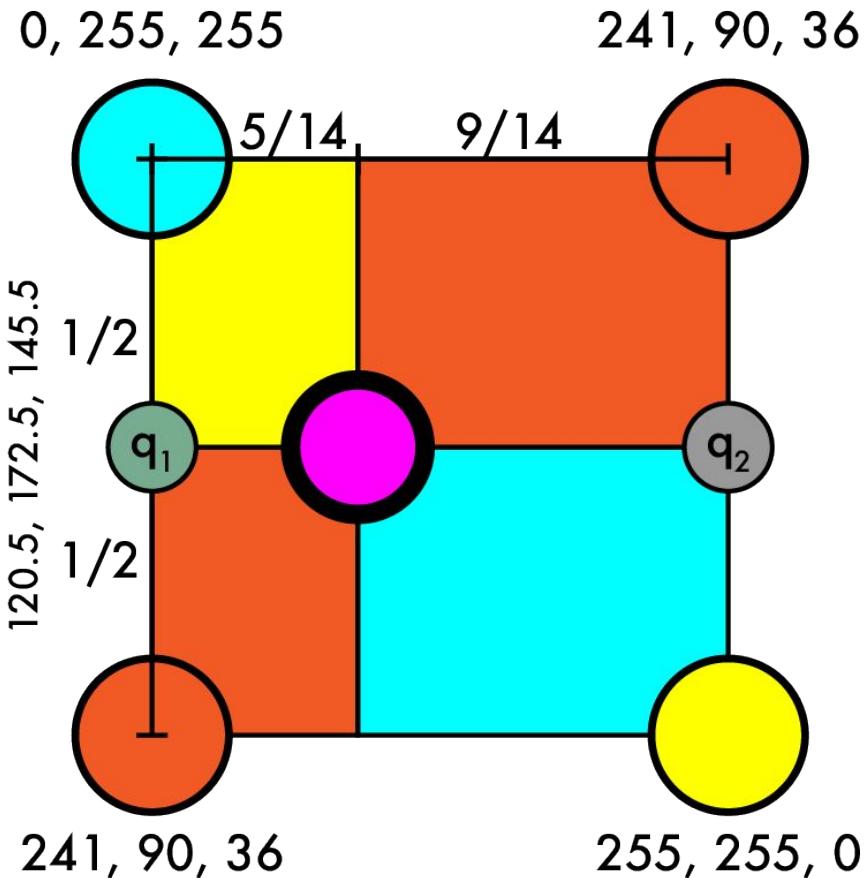
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q_1 = (120.5, 172.5, 145.5)$
    - $q_2 = r2, g2, b2$



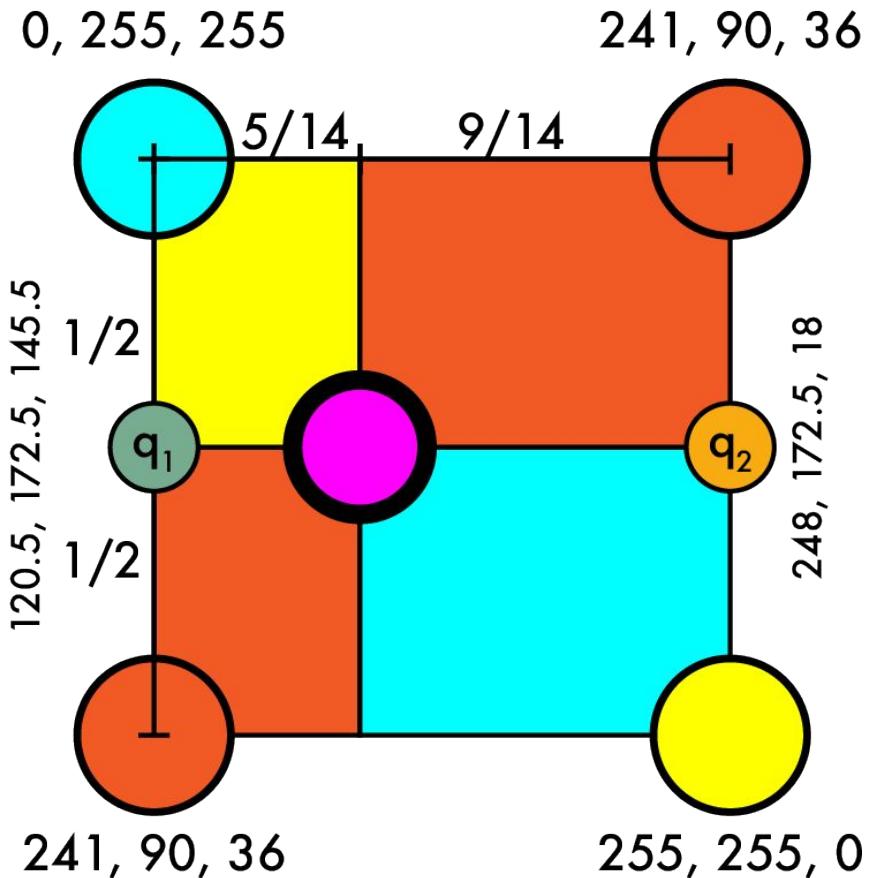
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q_1 = (120.5, 172.5, 145.5)$
    - $q_2 = r2, g2, b2$
    - $r2 = .5*241 + .5*255$
    - $g2 = .5*90 + .5*255$
    - $b2 = .5*36 + .5*0$



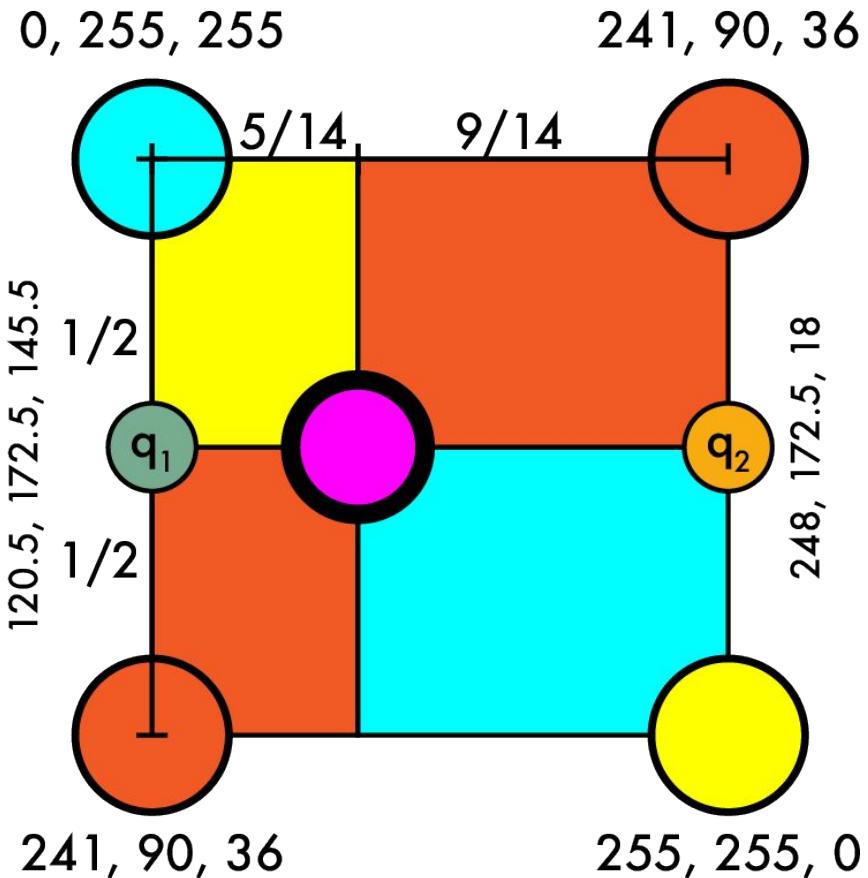
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q_1 = (120.5, 172.5, 145.5)$
    - $q_2 = (248, 172.5, 18)$



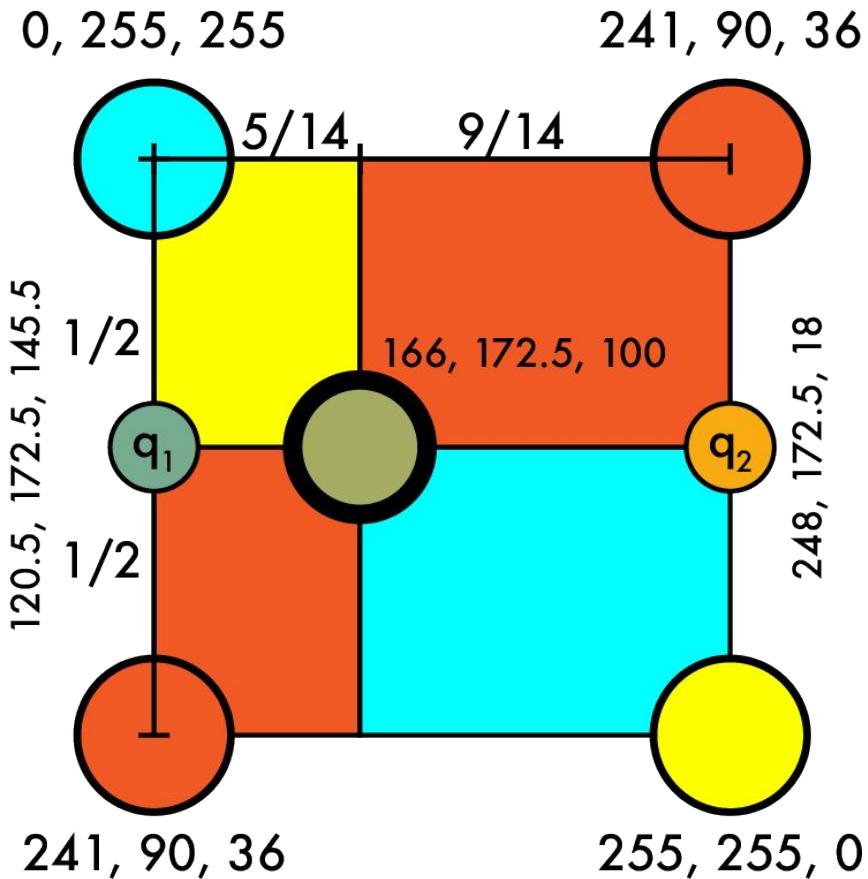
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q_1 = (120.5, 172.5, 145.5)$
    - $q_2 = (248, 172.5, 18)$
    - $q = r, g, b$
    - $w = 9/14 * w_1 + 5/14 w_2$



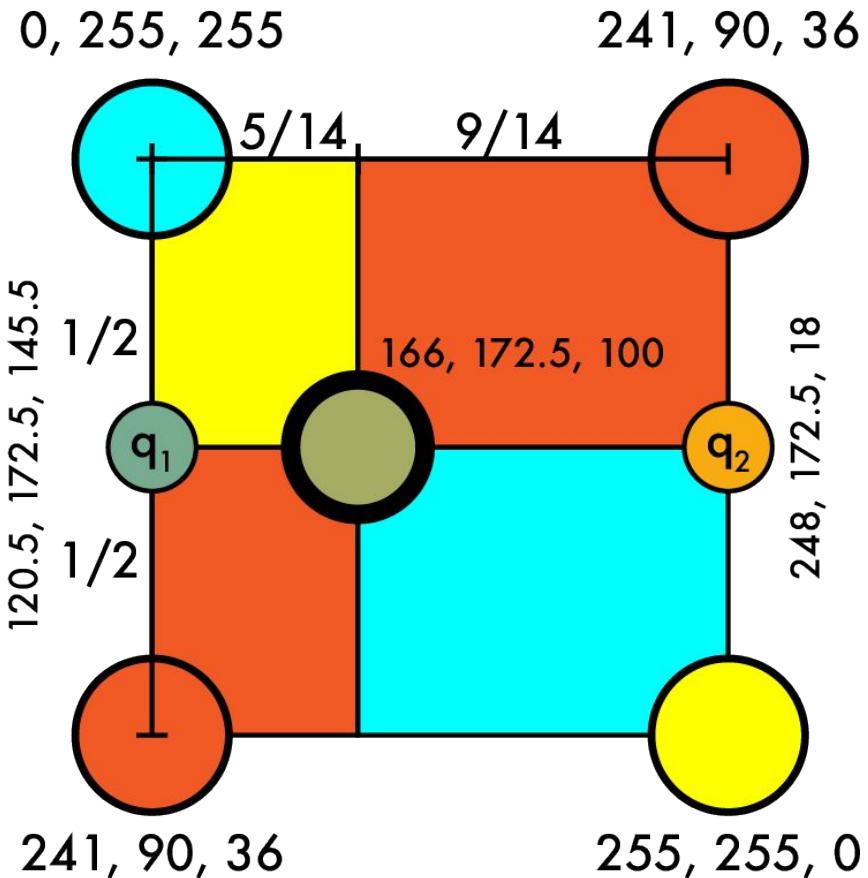
# Ресайзинг 4x4 -> 7x7

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q_1 = (120.5, 172.5, 145.5)$
    - $q_2 = (248, 172.5, 18)$
    - $q = (166, 172.5, 100)$



# Ресайзинг 4x4 -> 7x7

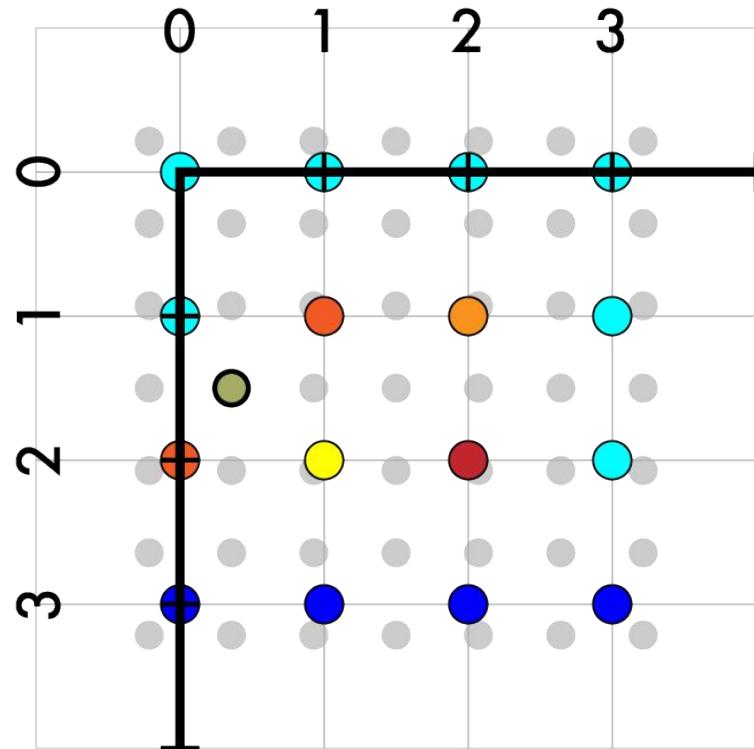
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q = (166, 172.5, 100)$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

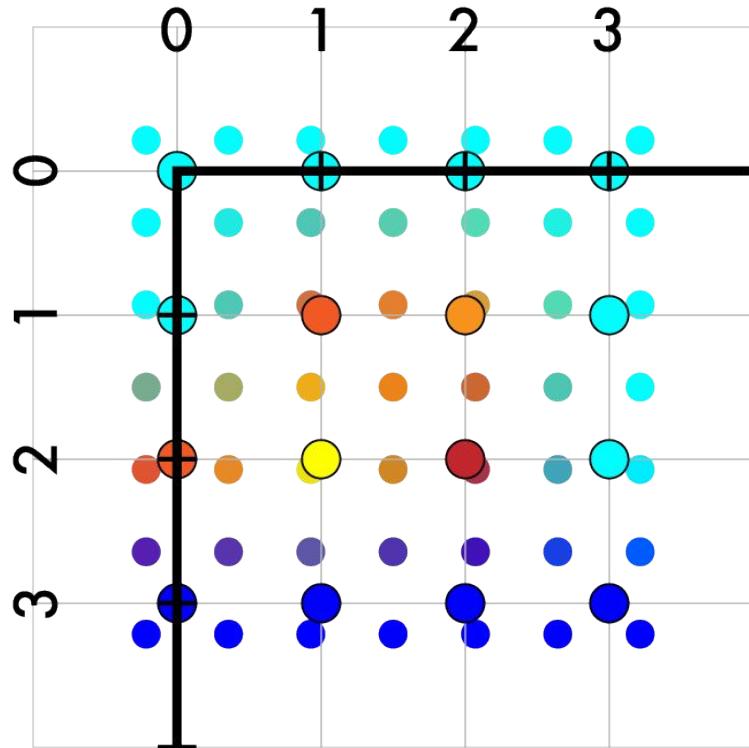
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q = (166, 172.5, 100)$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

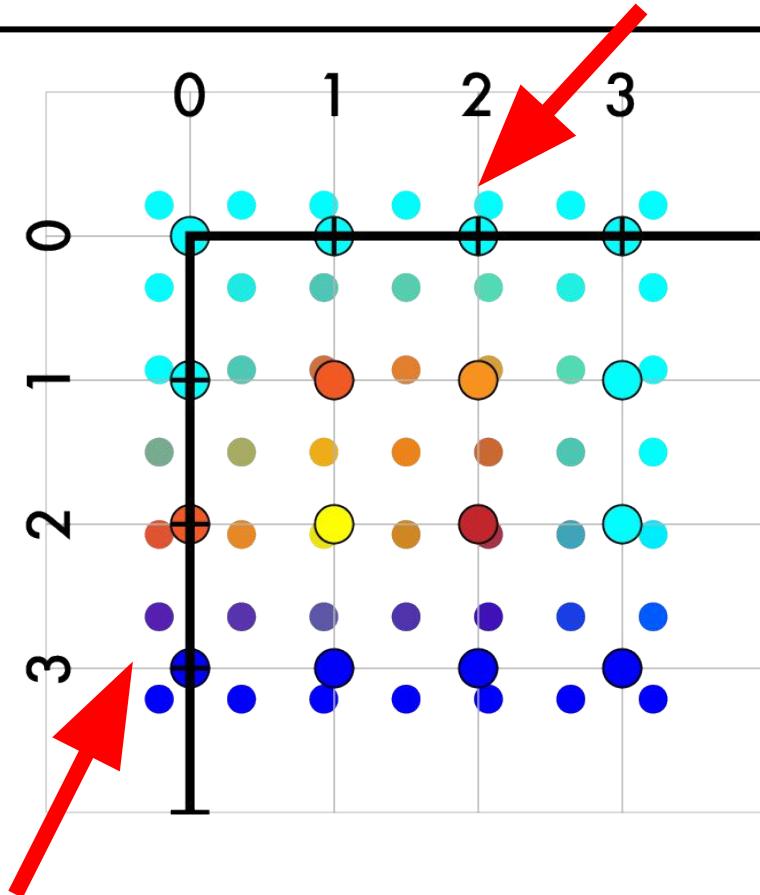
---

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q = (166, 172.5, 100)$



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

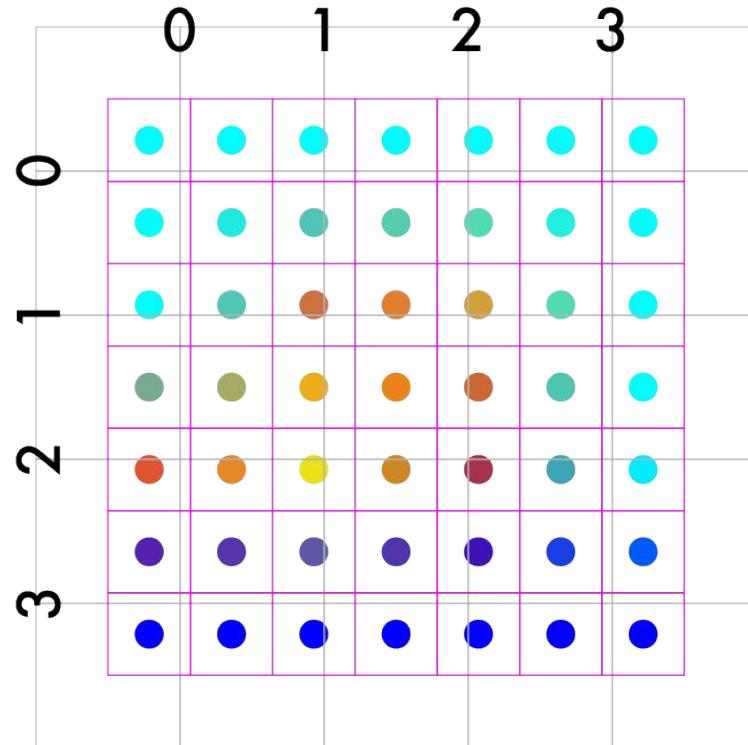
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q = (166, 172.5, 100)$
- На внешних краях использовать паддинг!



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

---

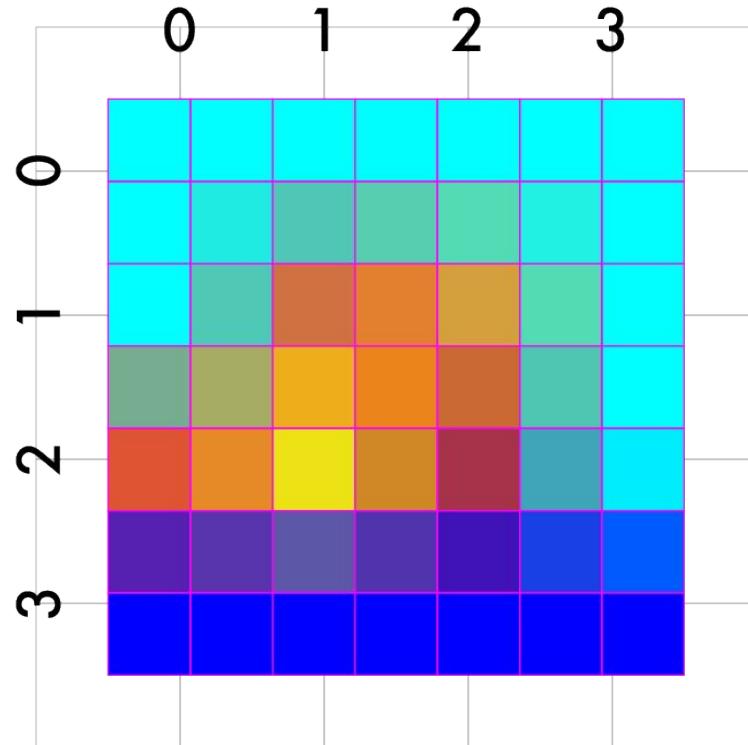
- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q = (166, 172.5, 100)$
- На внешних краях использовать паддинг!



# Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

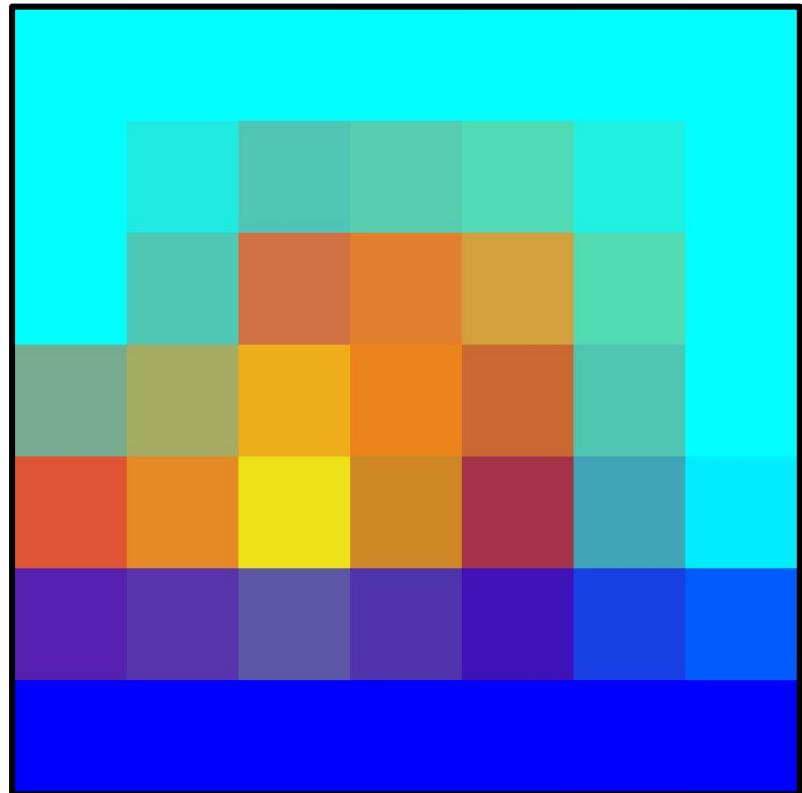
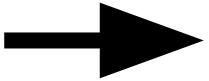
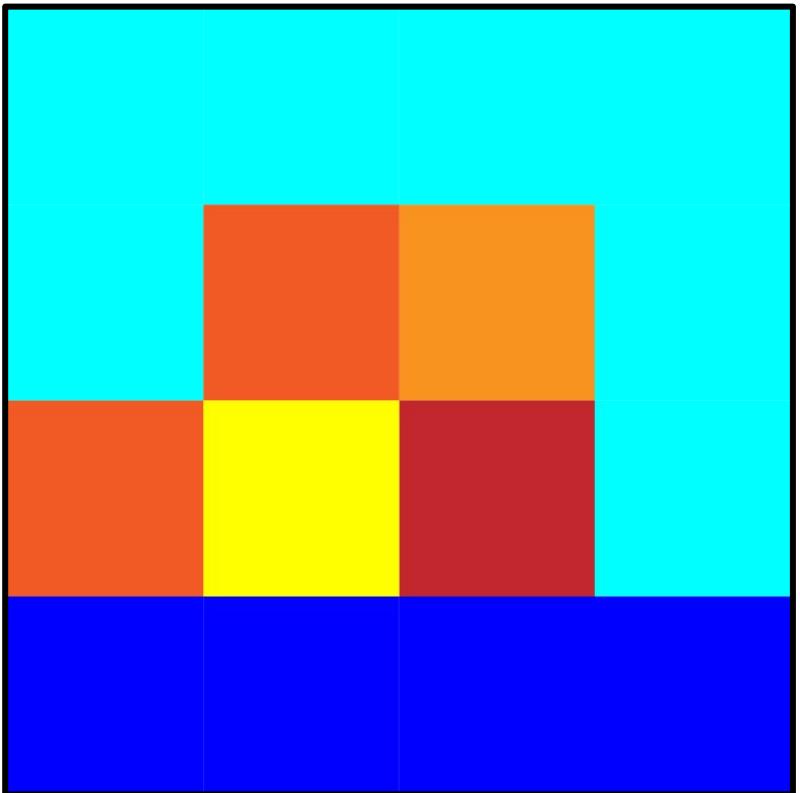
---

- Создать новое изображение
- Сматчить координаты
  - $4/7 X - 3/14 = Y$
- Проитерироваться по новым точкам
  - Отобразить в старые координаты
  - $(1, 3) \rightarrow (5/14, 3/2)$
  - Интерполировать старые значения
    - $q = (166, 172.5, 100)$
- На внешних краях использовать паддинг!



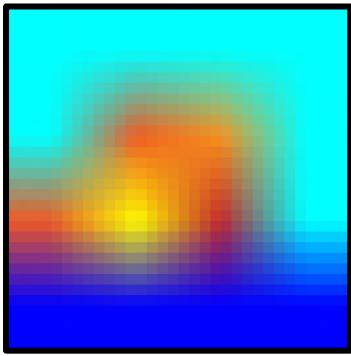
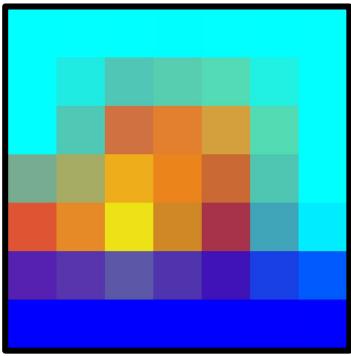
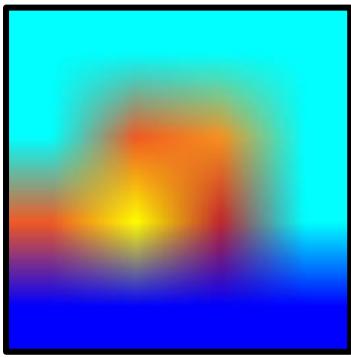
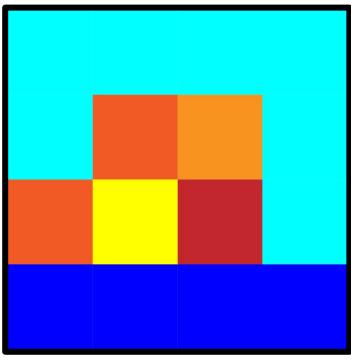
Вуаля!

---



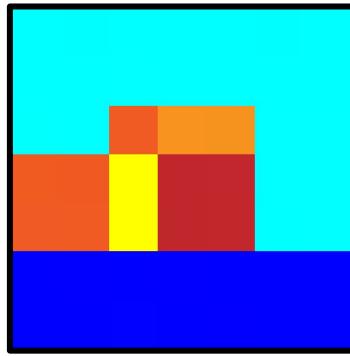
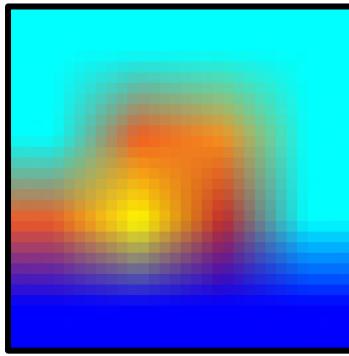
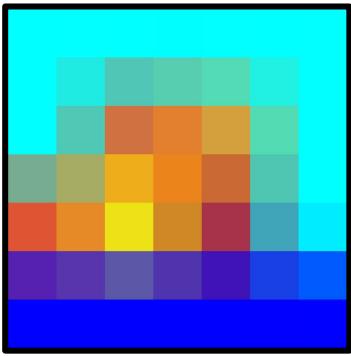
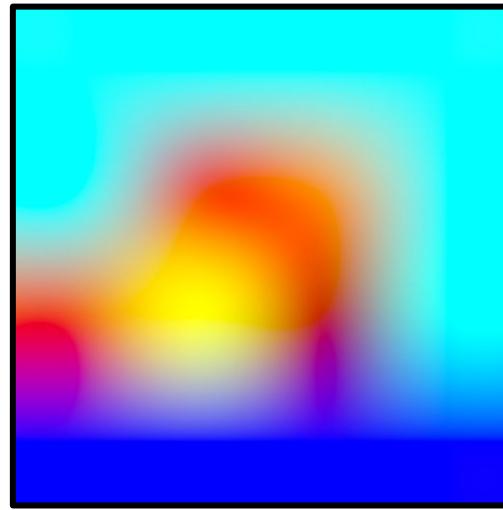
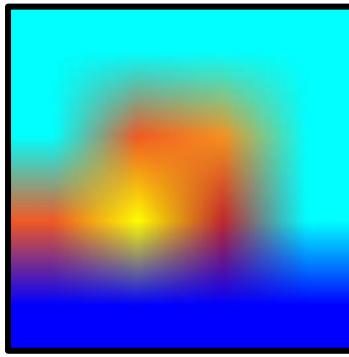
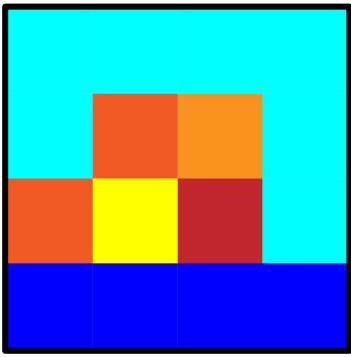
# Различные масштабы

---



# Различные методы

---



# Уменьшение изображения

---



448x448 -> 64x64

---



448x448 -> 64x64

---



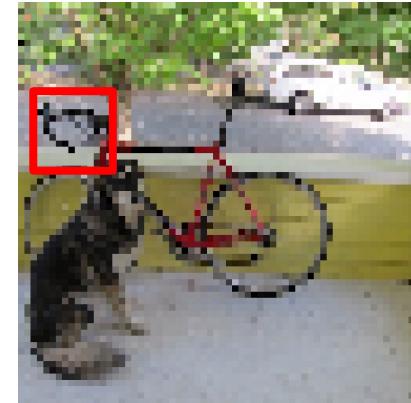
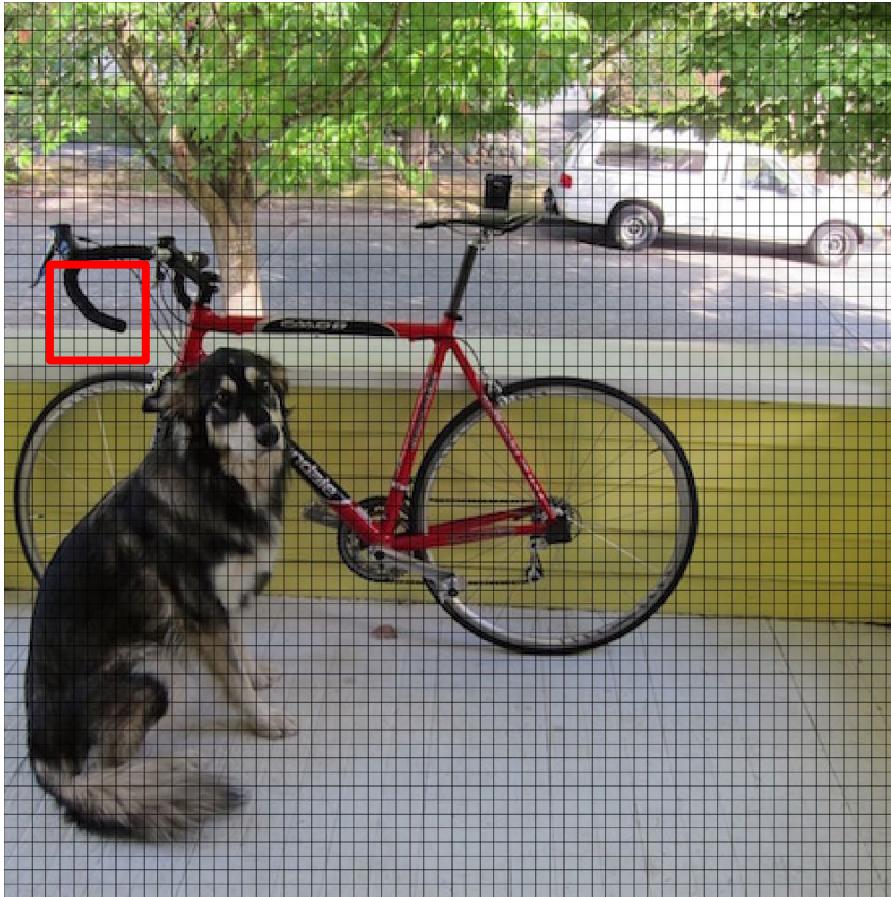
448x448 -> 64x64

---



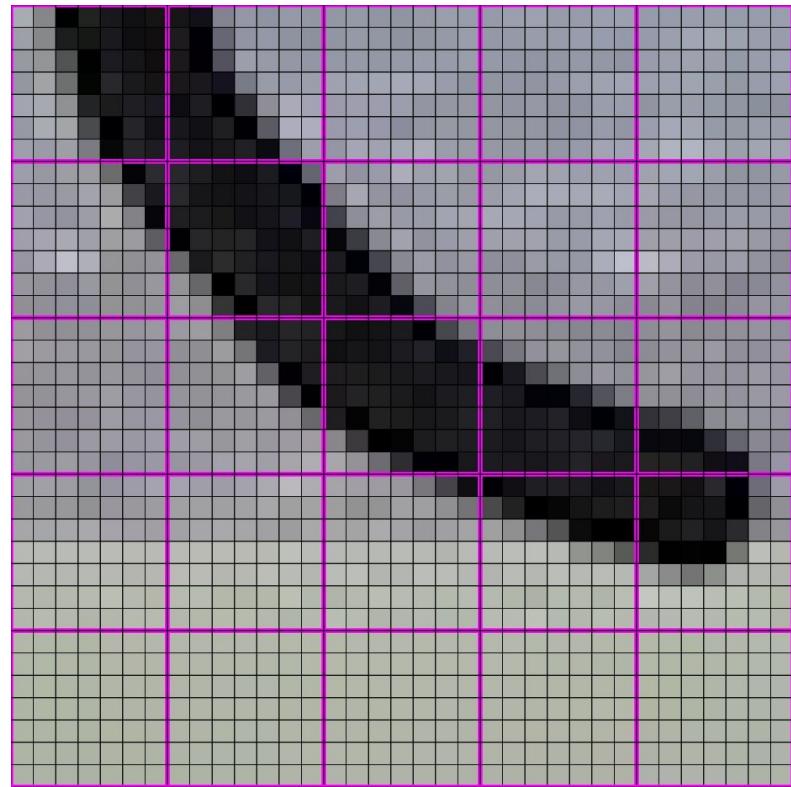
448x448 -> 64x64

---



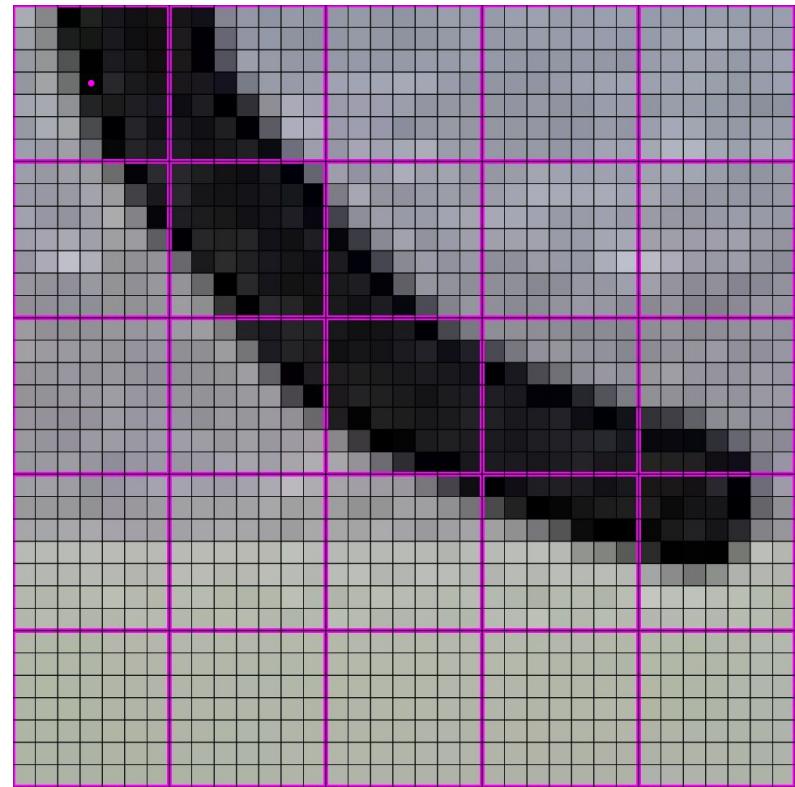
448x448 -> 64x64

---



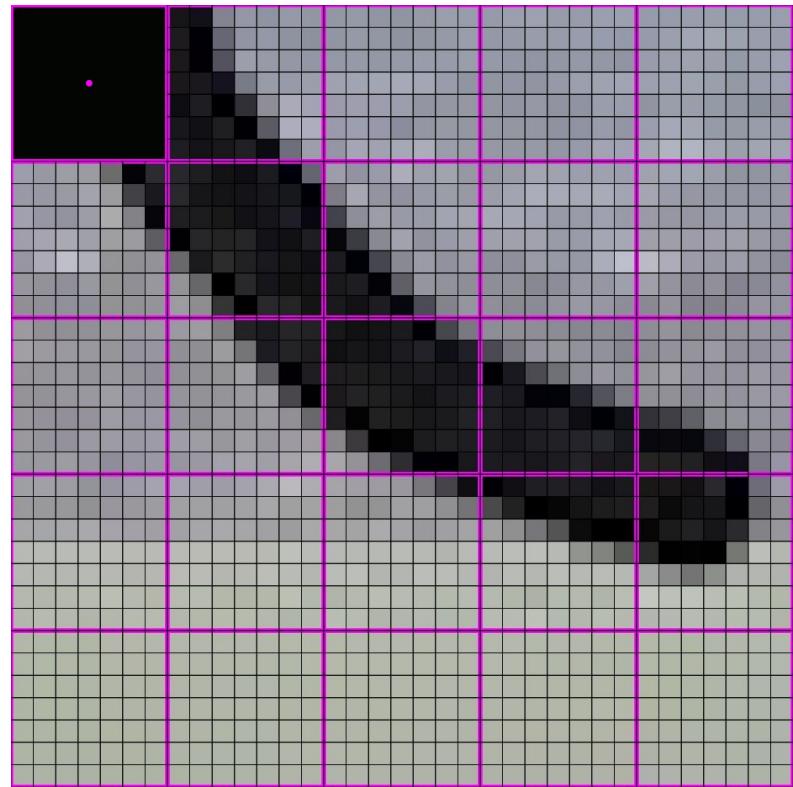
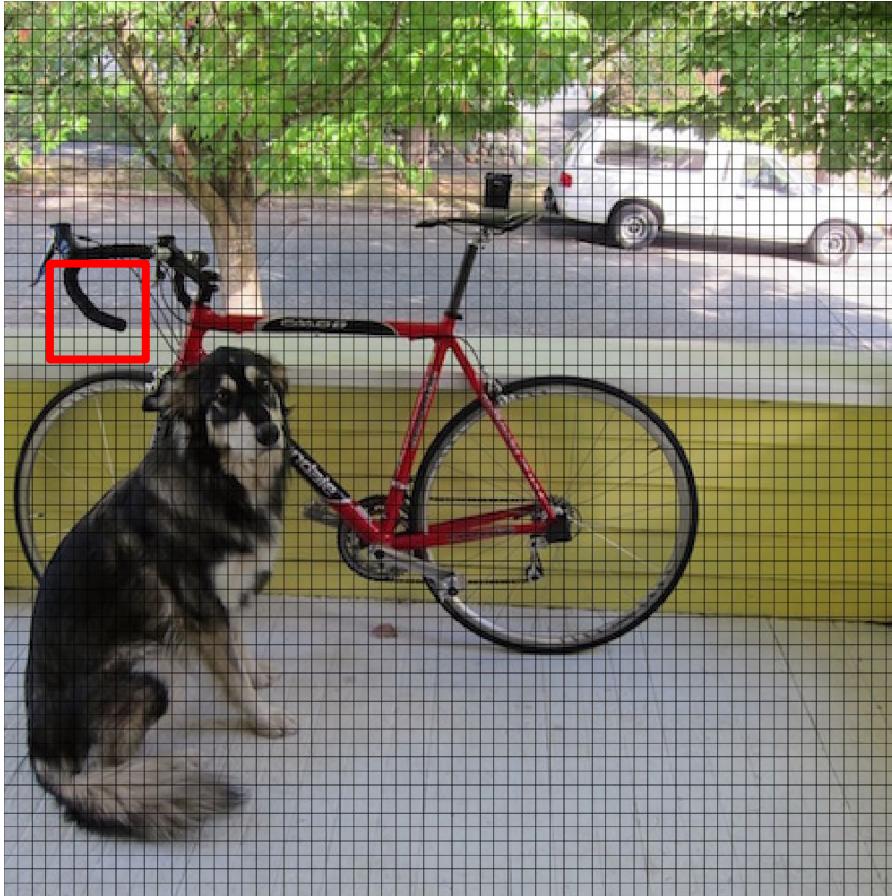
448x448 -> 64x64

---



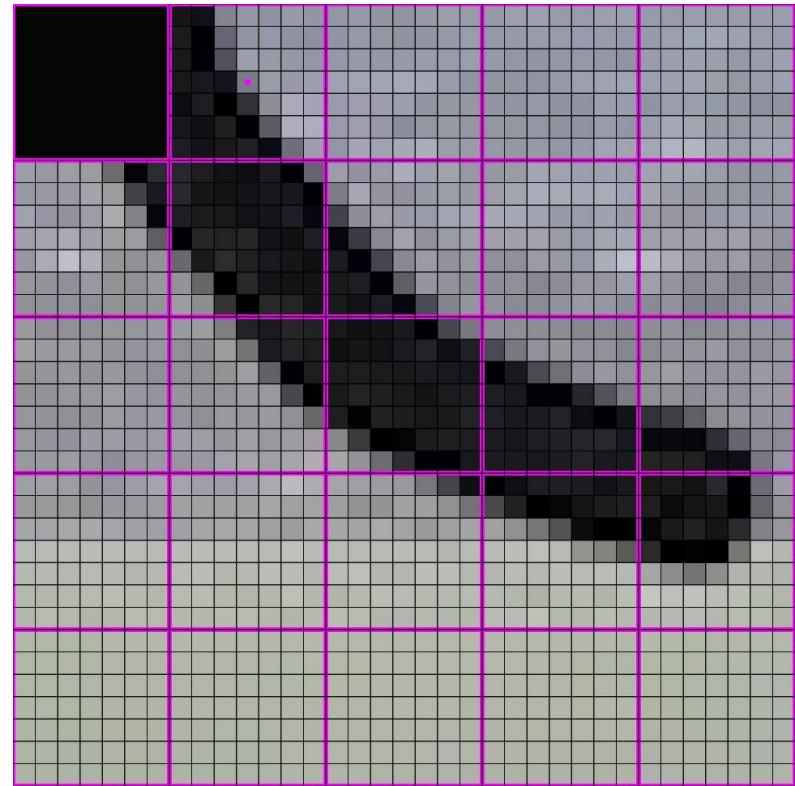
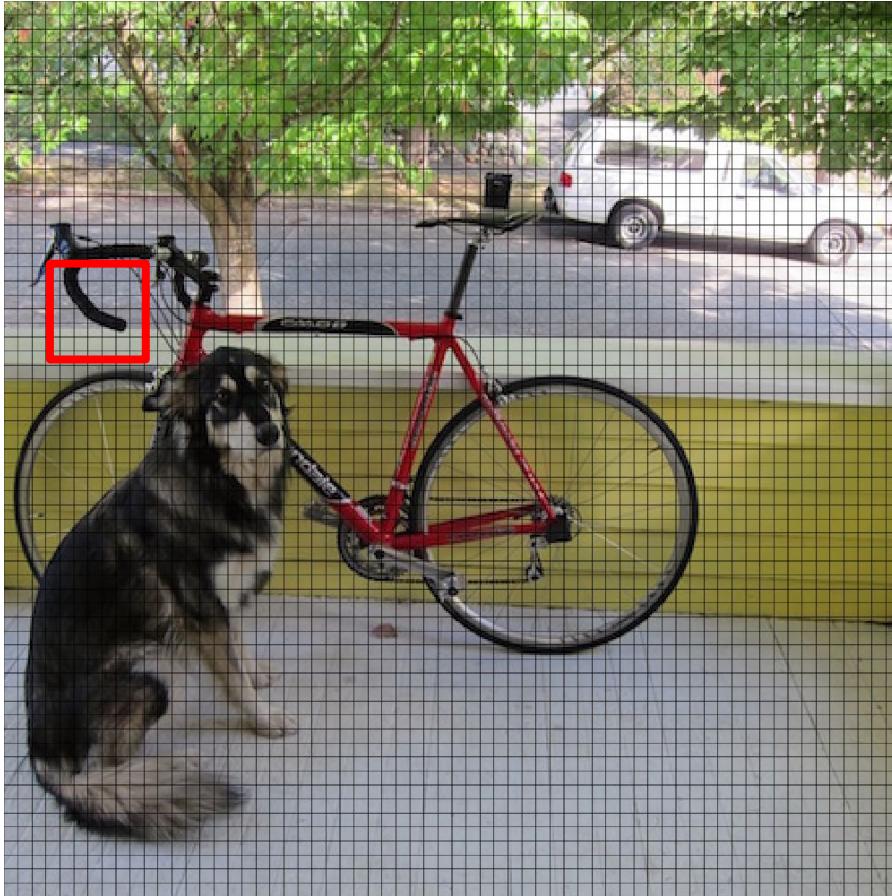
448x448 -> 64x64

---



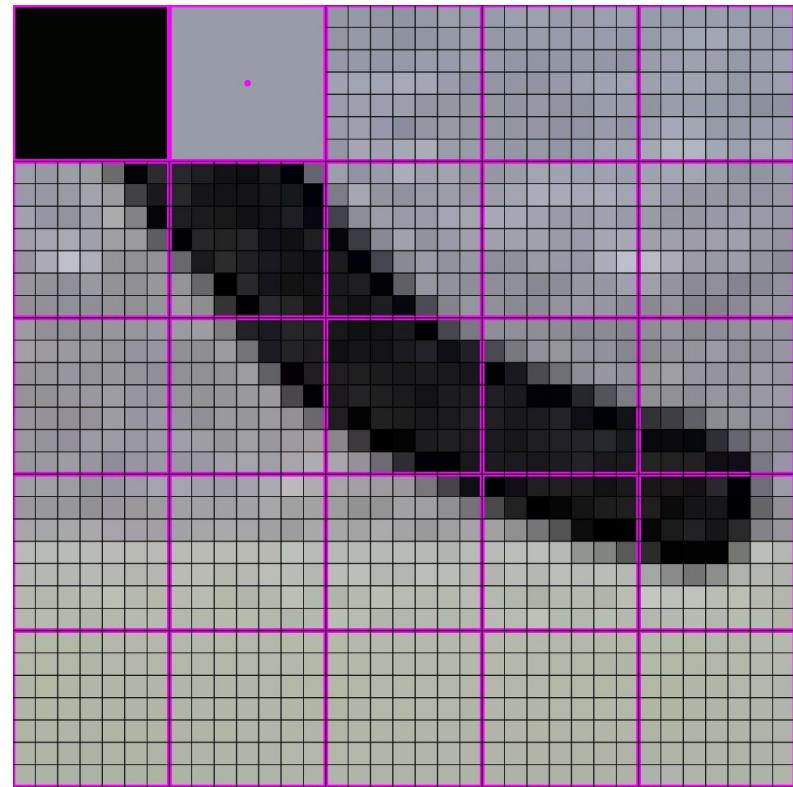
448x448 -> 64x64

---



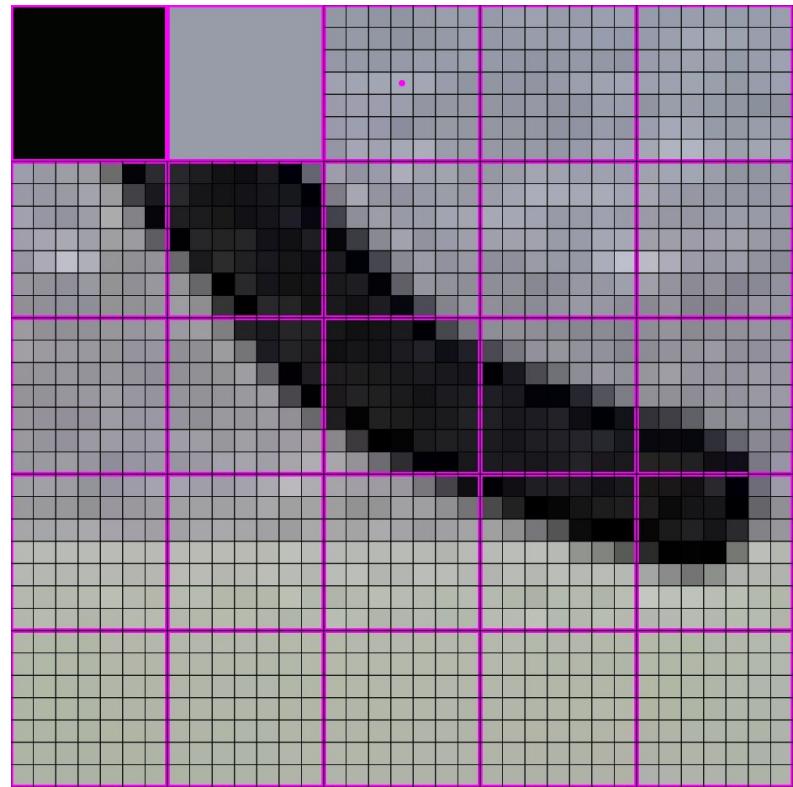
448x448 -> 64x64

---



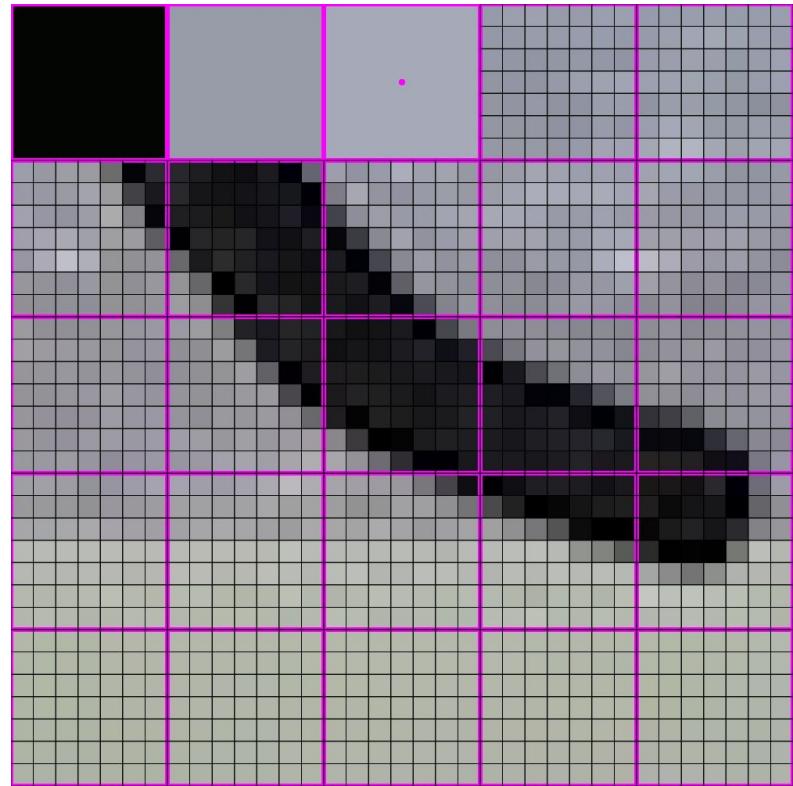
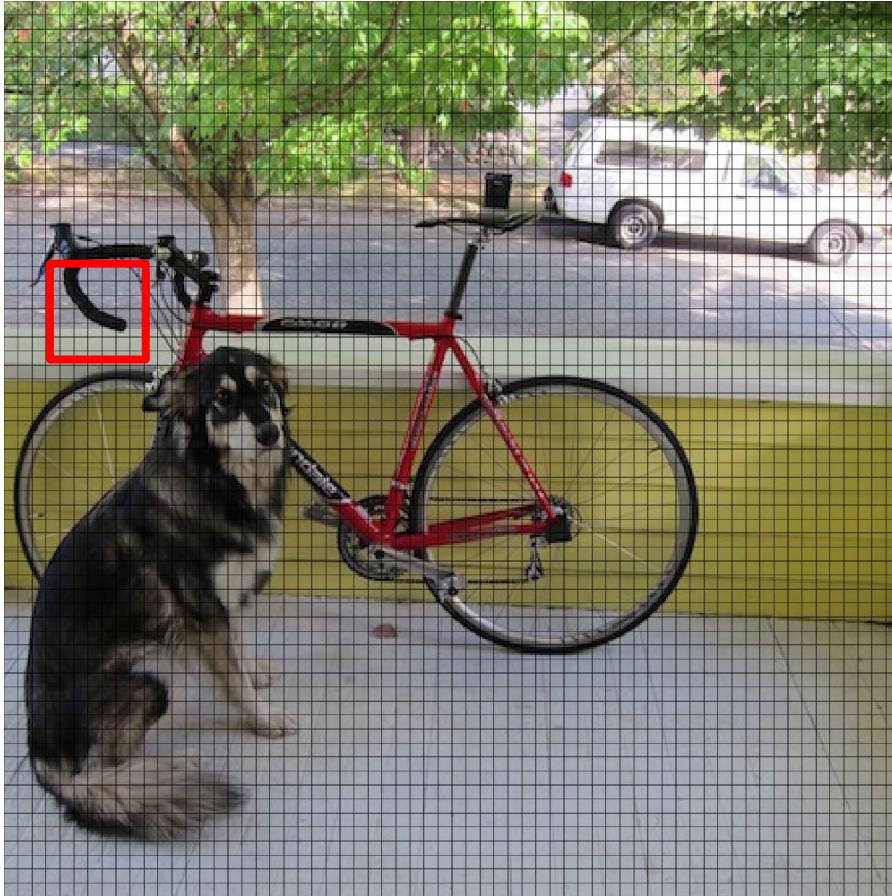
448x448 -> 64x64

---



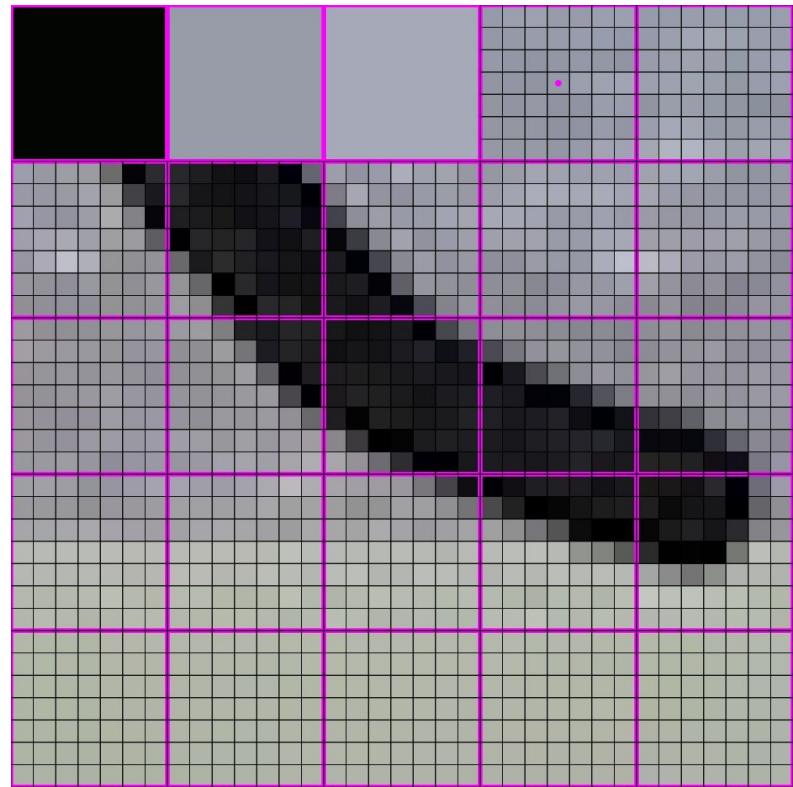
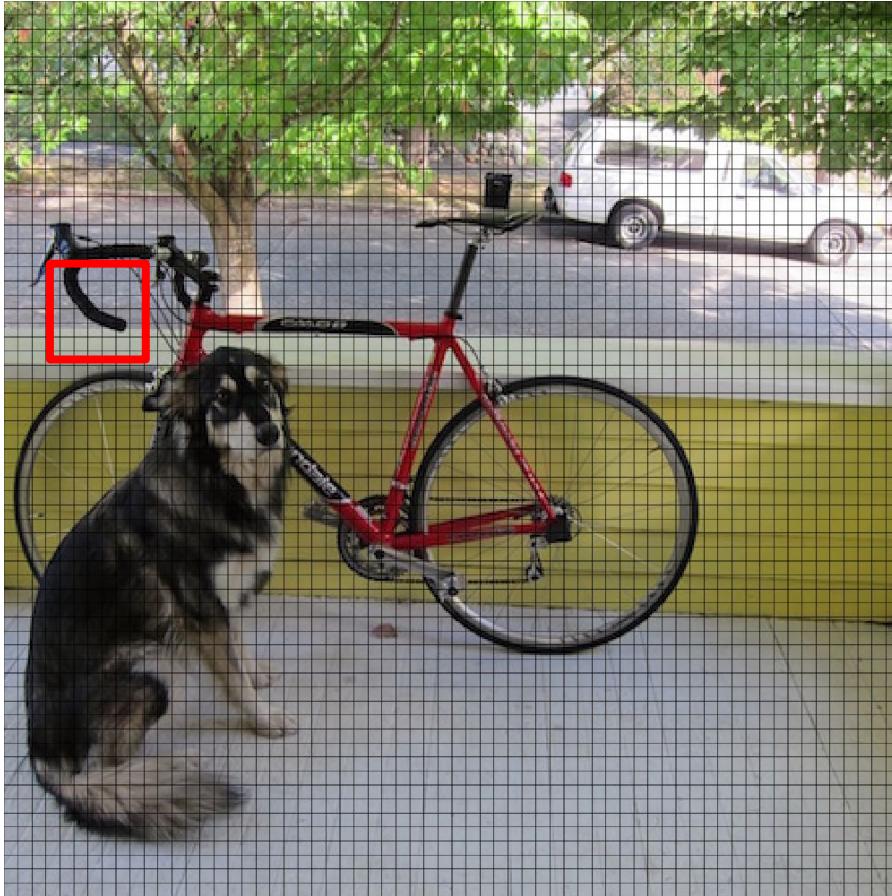
448x448 -> 64x64

---



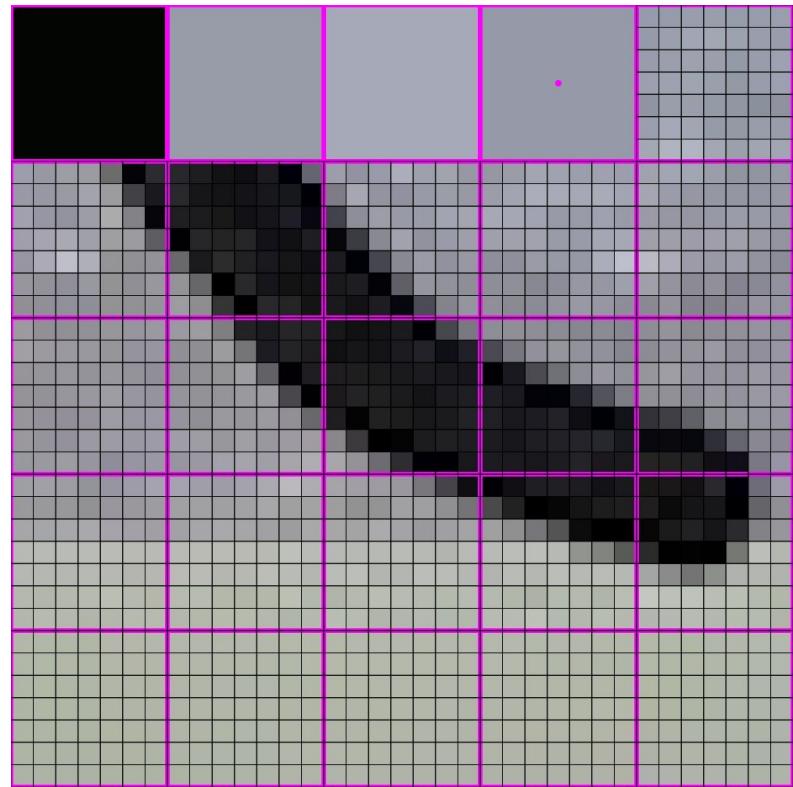
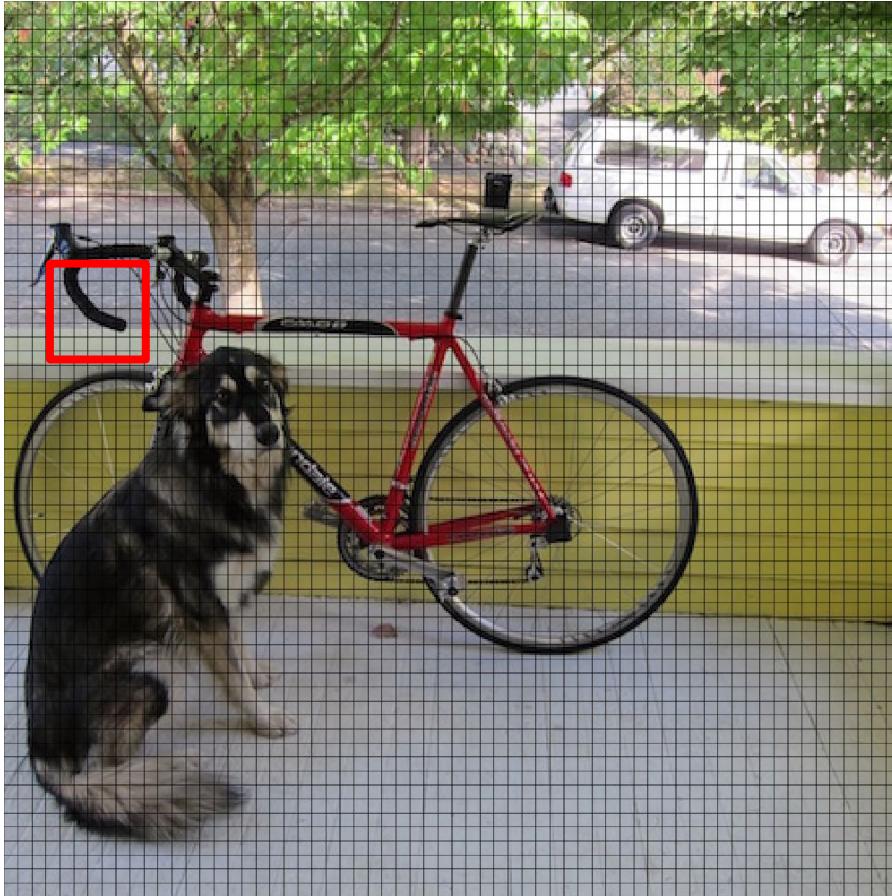
448x448 -> 64x64

---



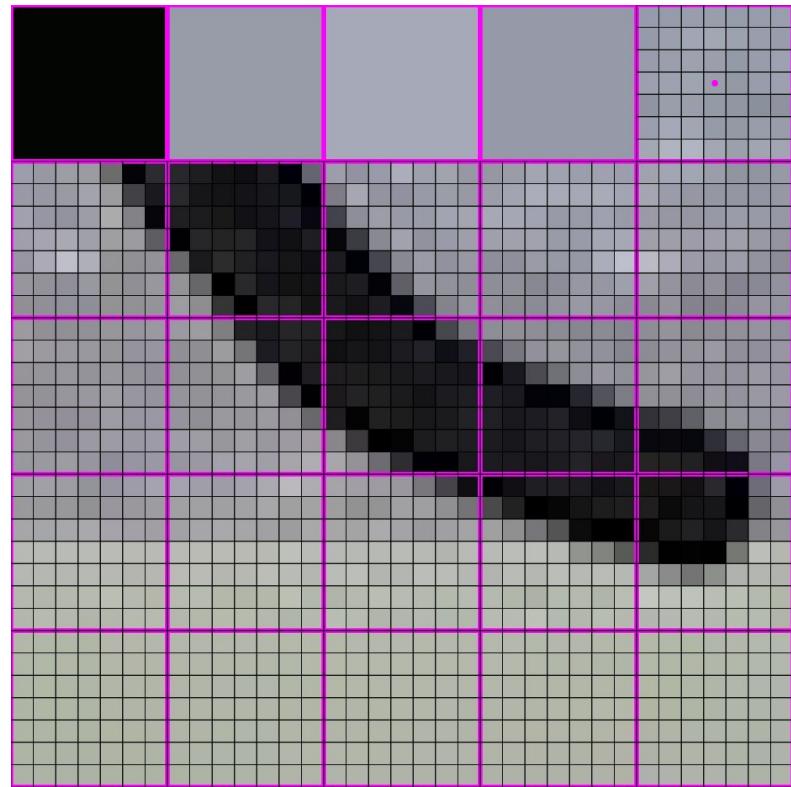
448x448 -> 64x64

---



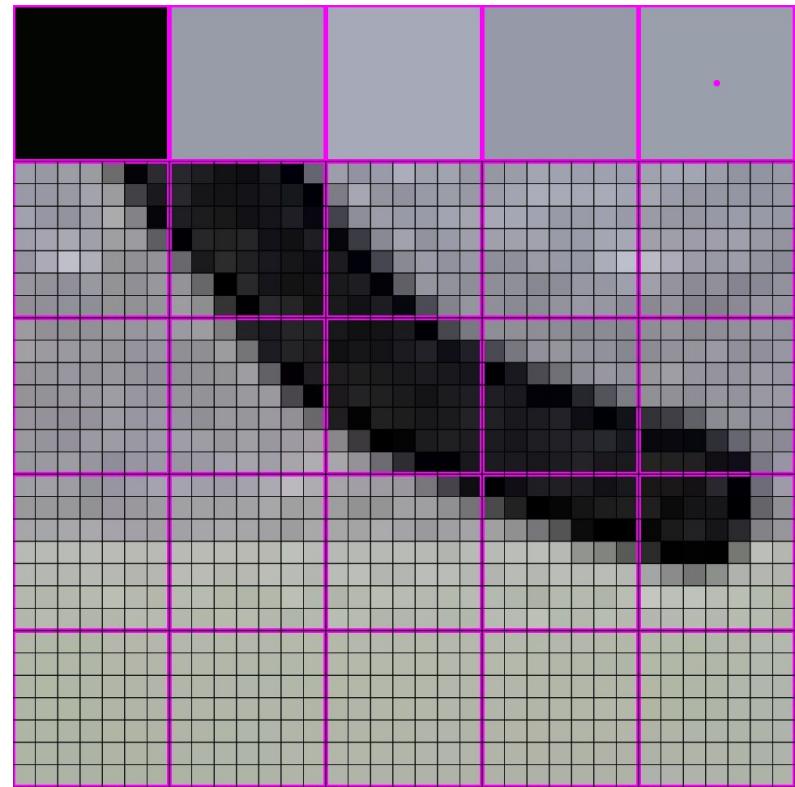
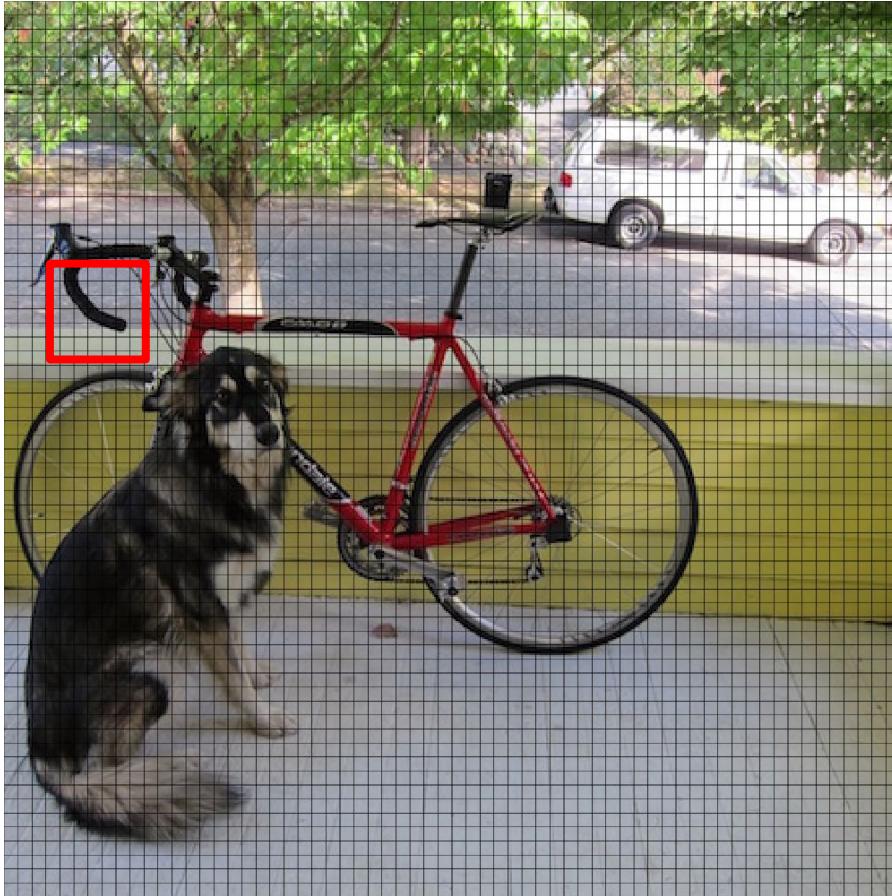
448x448 -> 64x64

---



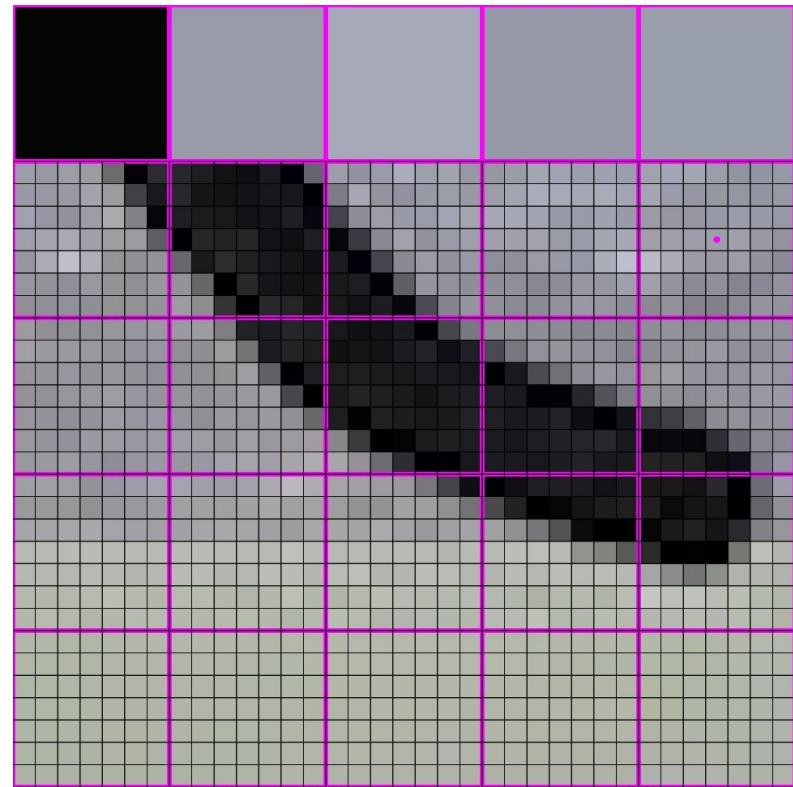
448x448 -> 64x64

---



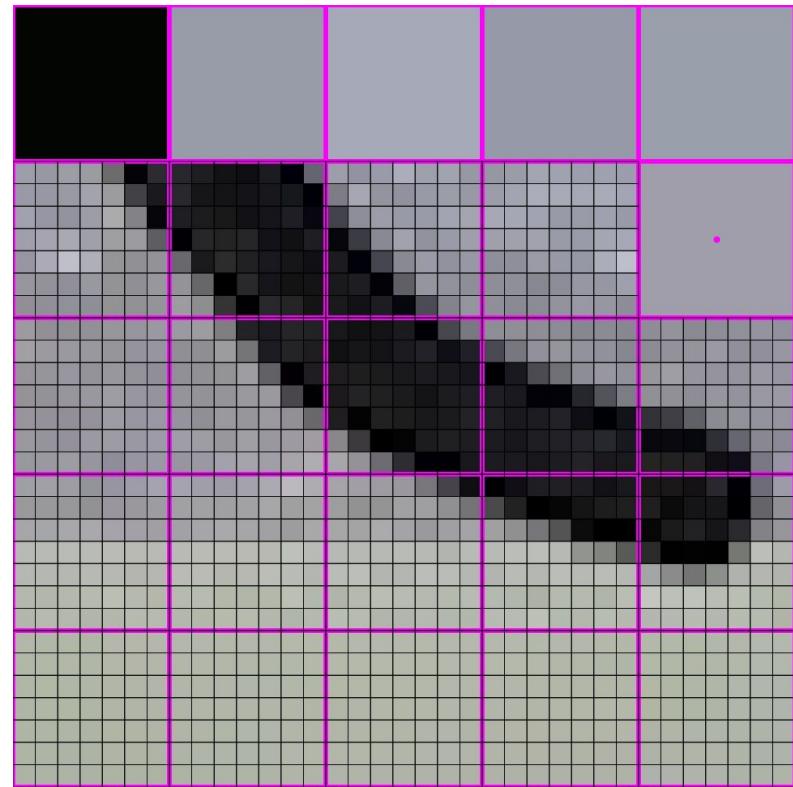
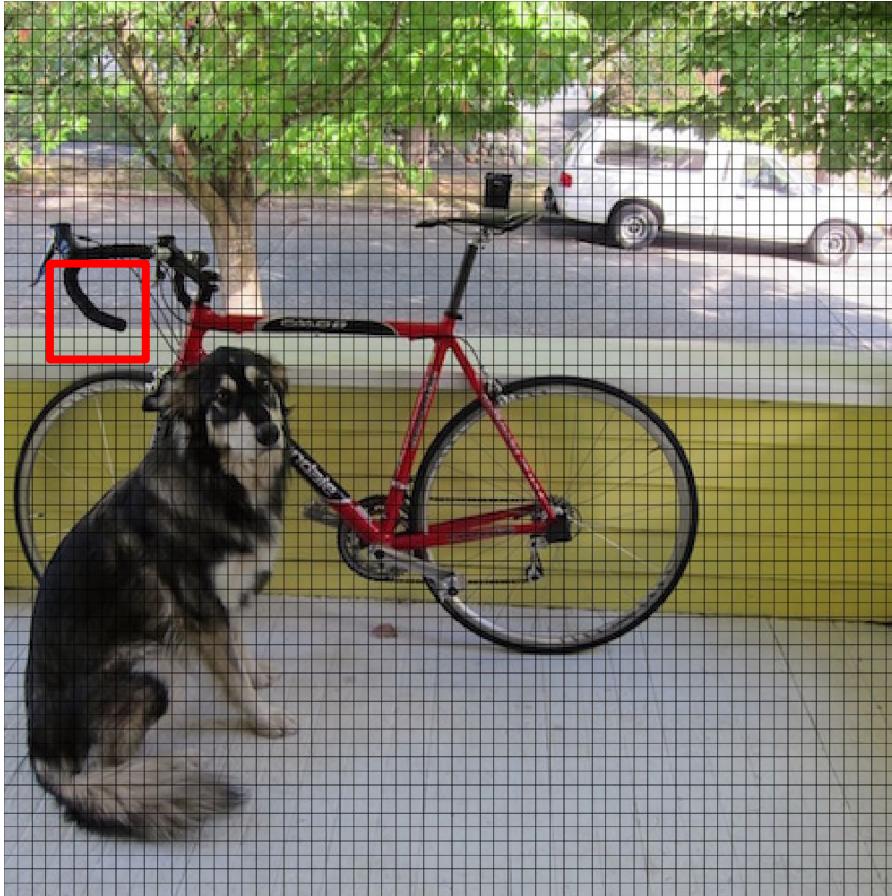
448x448 -> 64x64

---



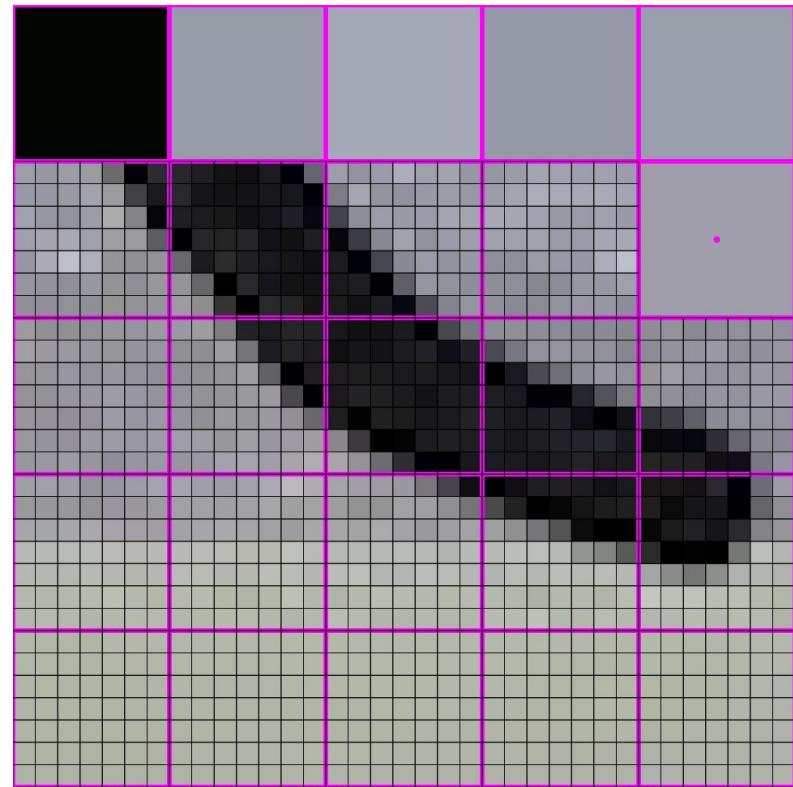
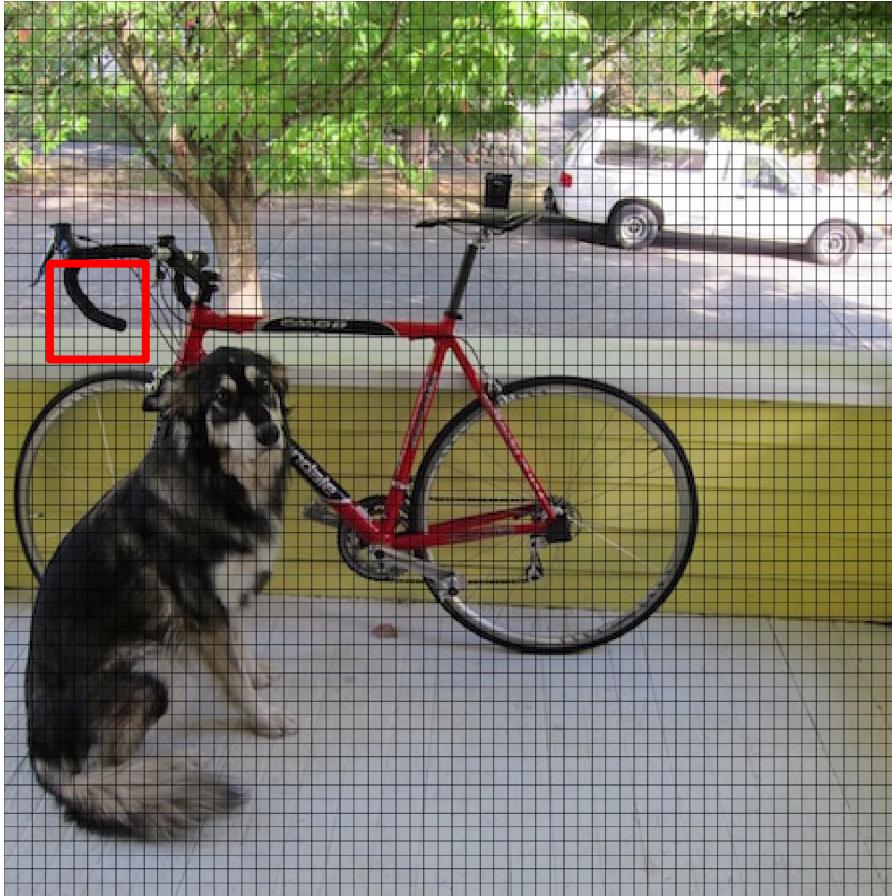
448x448 -> 64x64

---



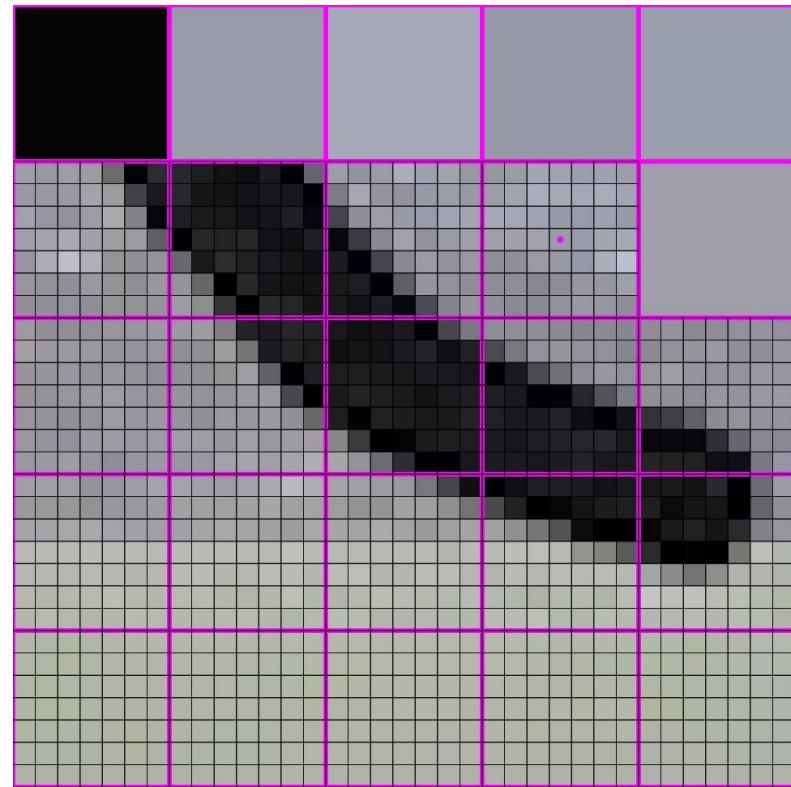
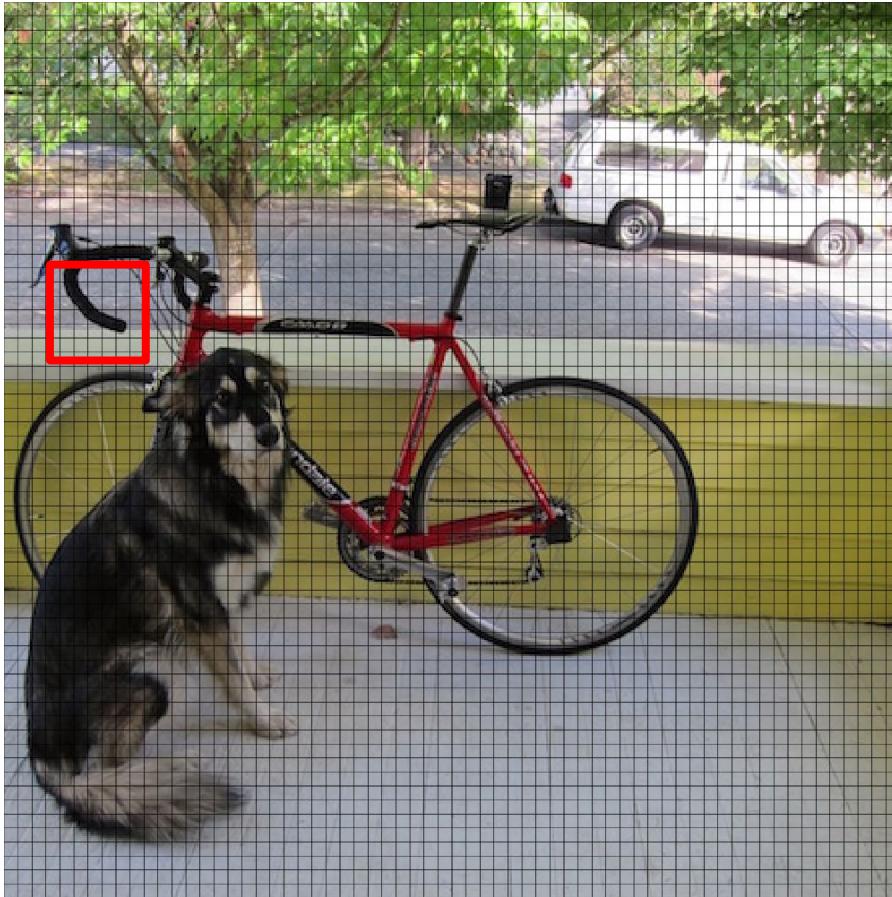
448x448 -> 64x64

---



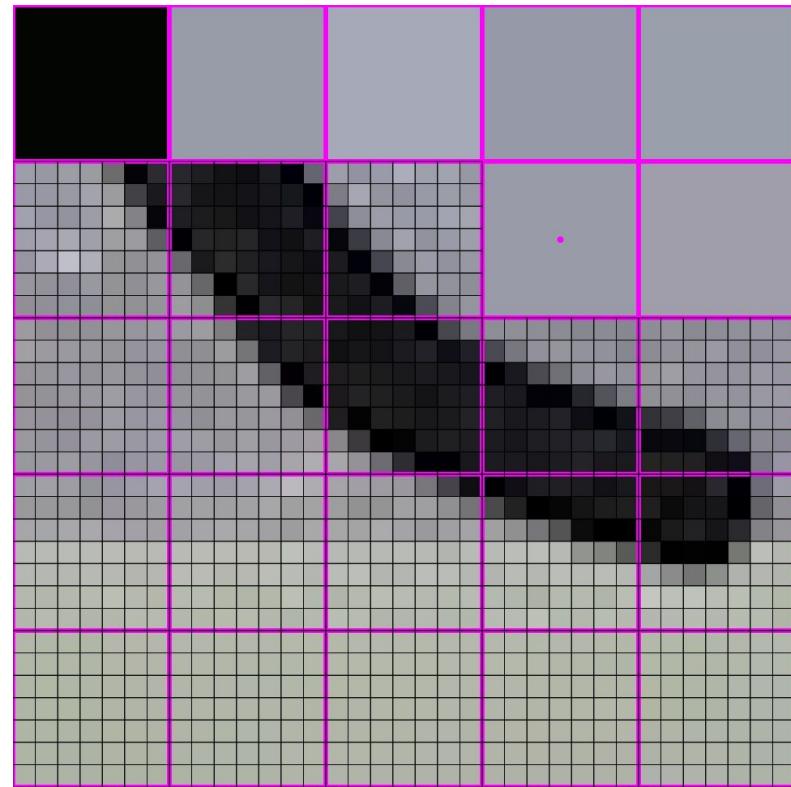
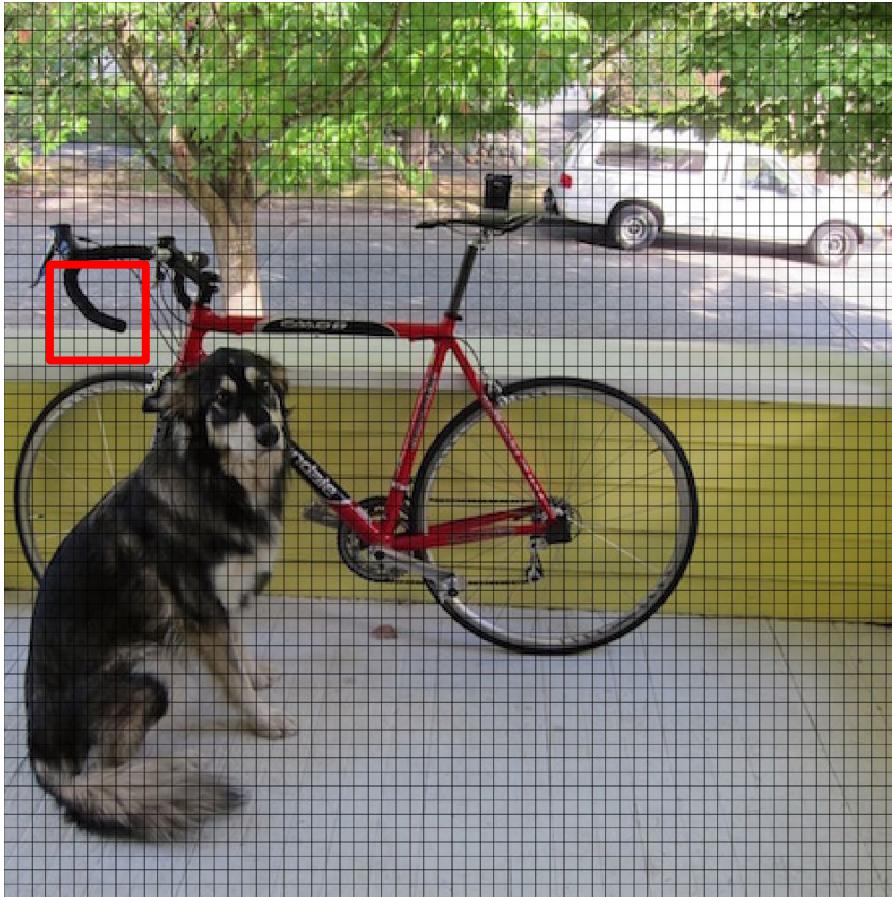
448x448 -> 64x64

---



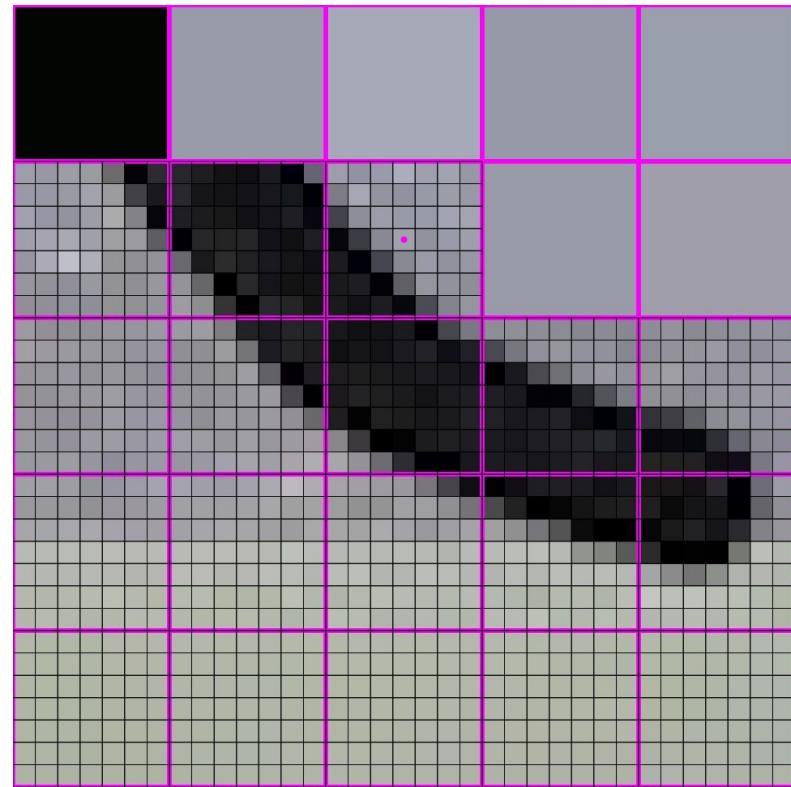
448x448 -> 64x64

---



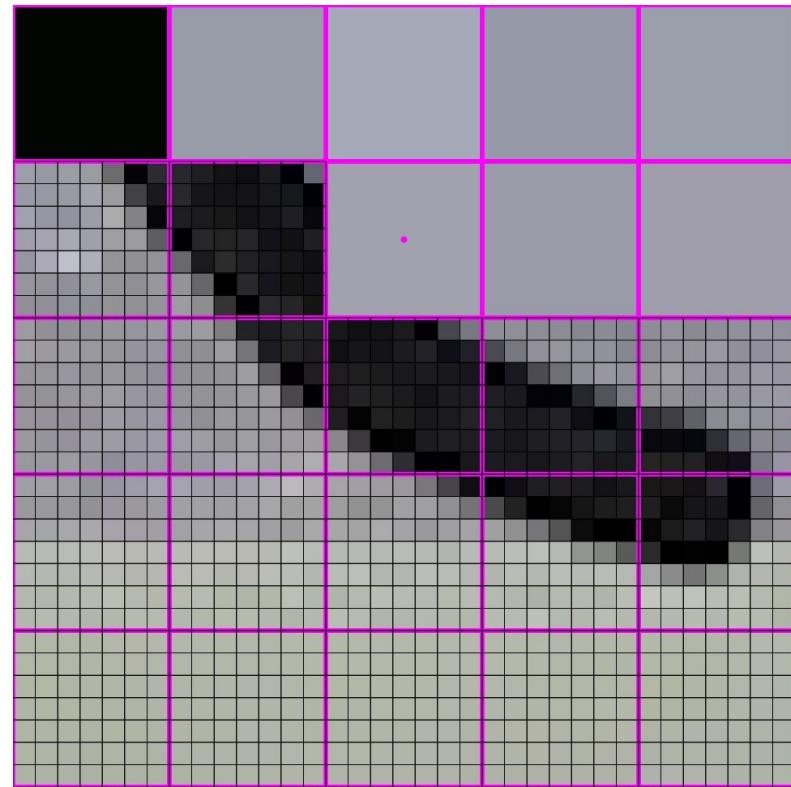
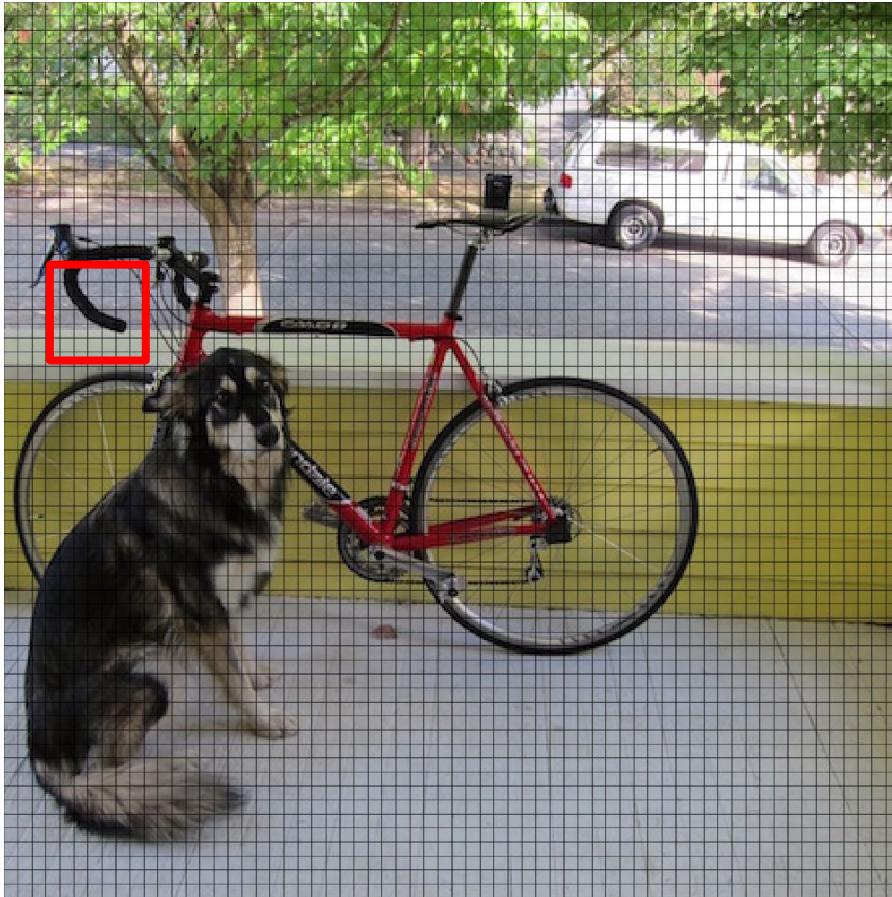
448x448 -> 64x64

---



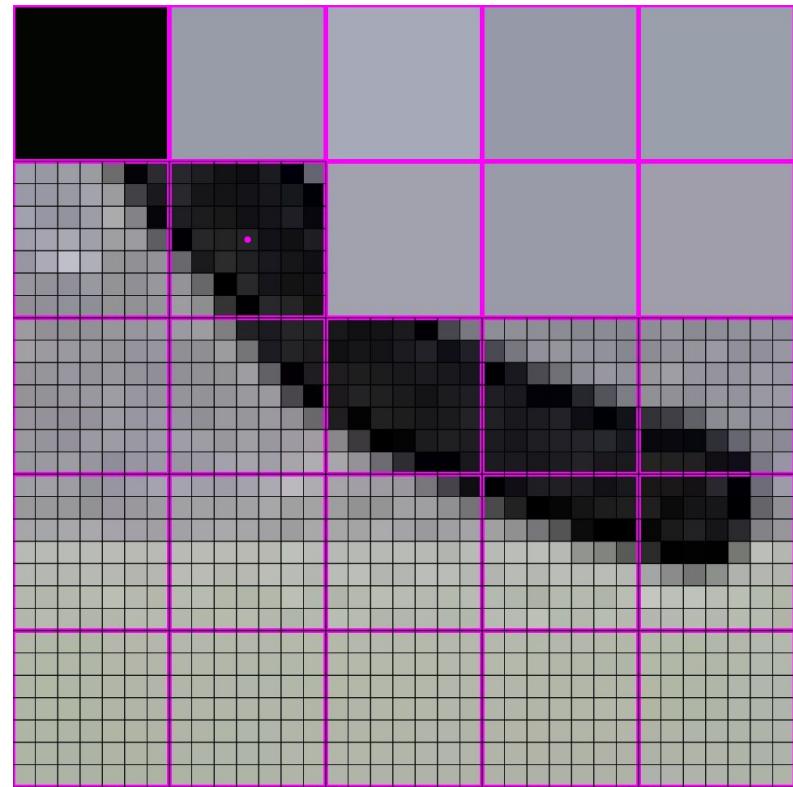
448x448 -> 64x64

---



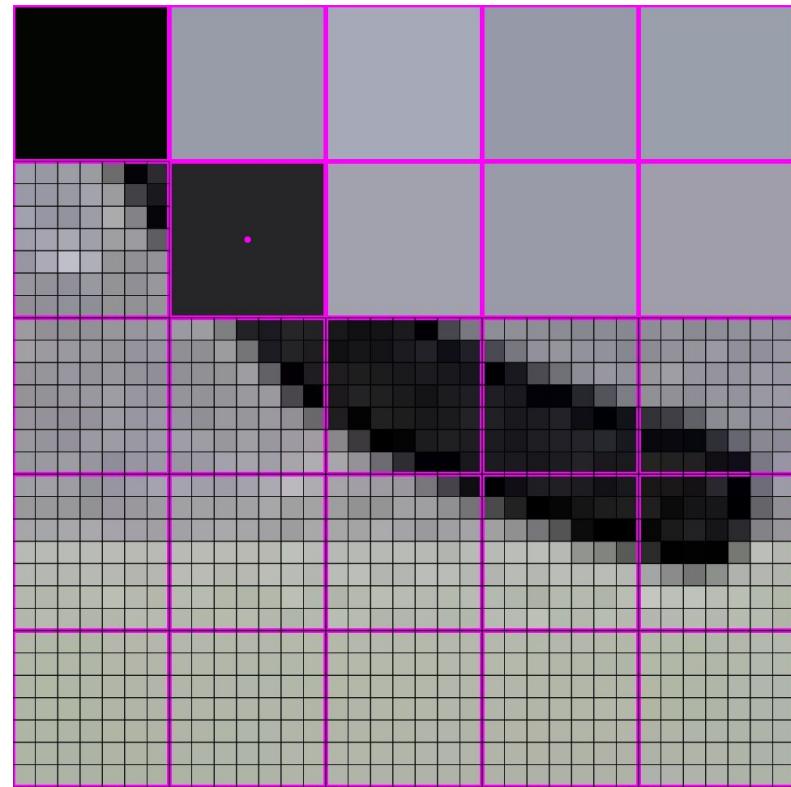
448x448 -> 64x64

---



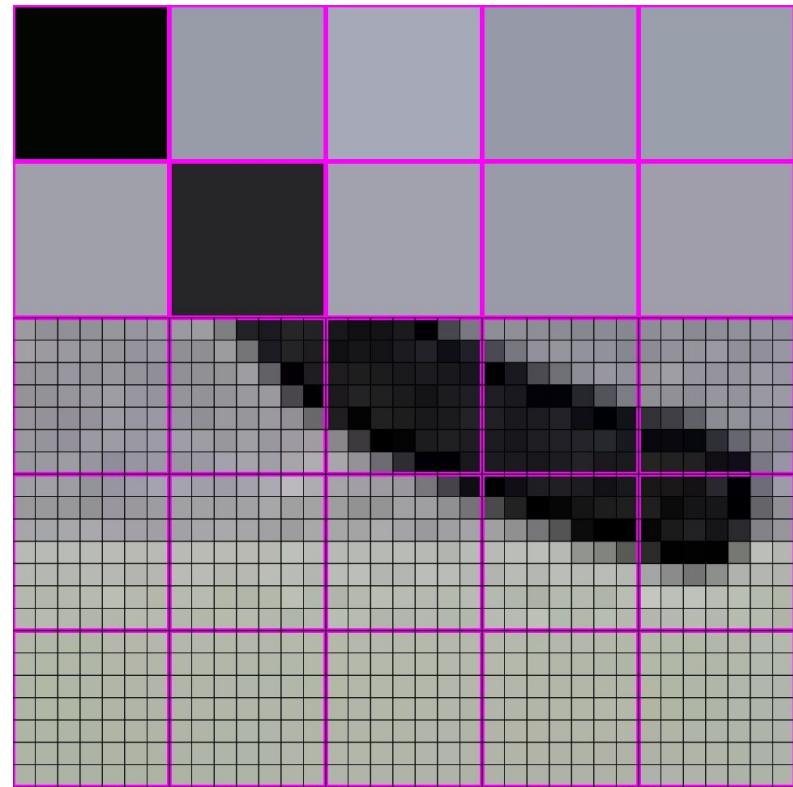
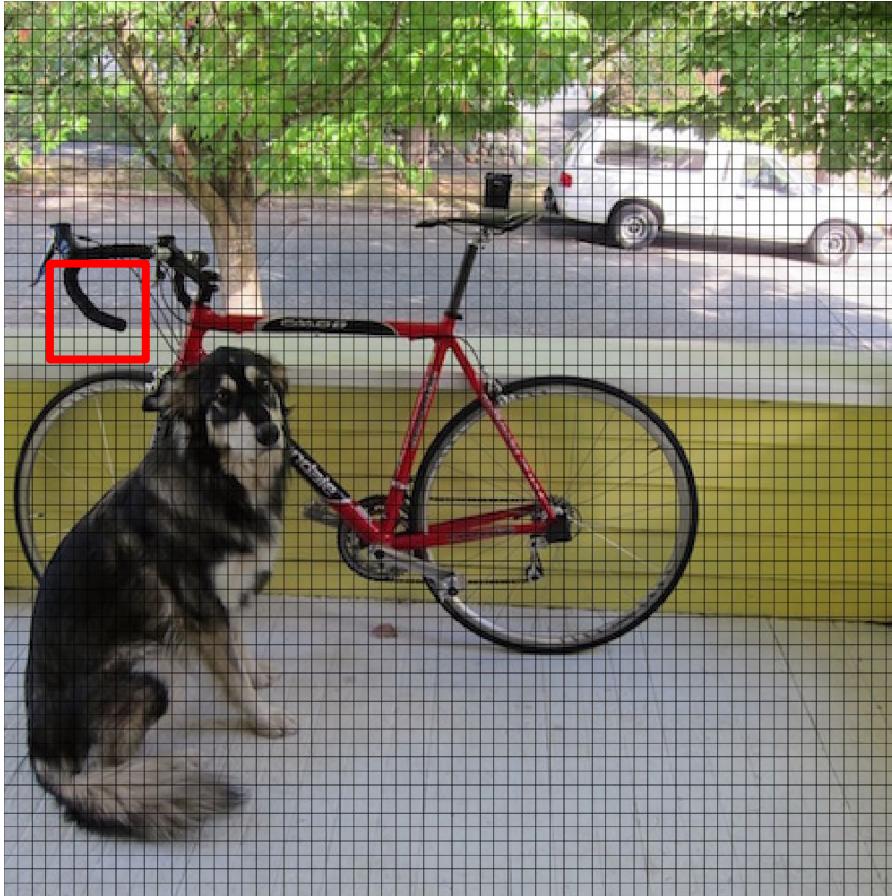
448x448 -> 64x64

---



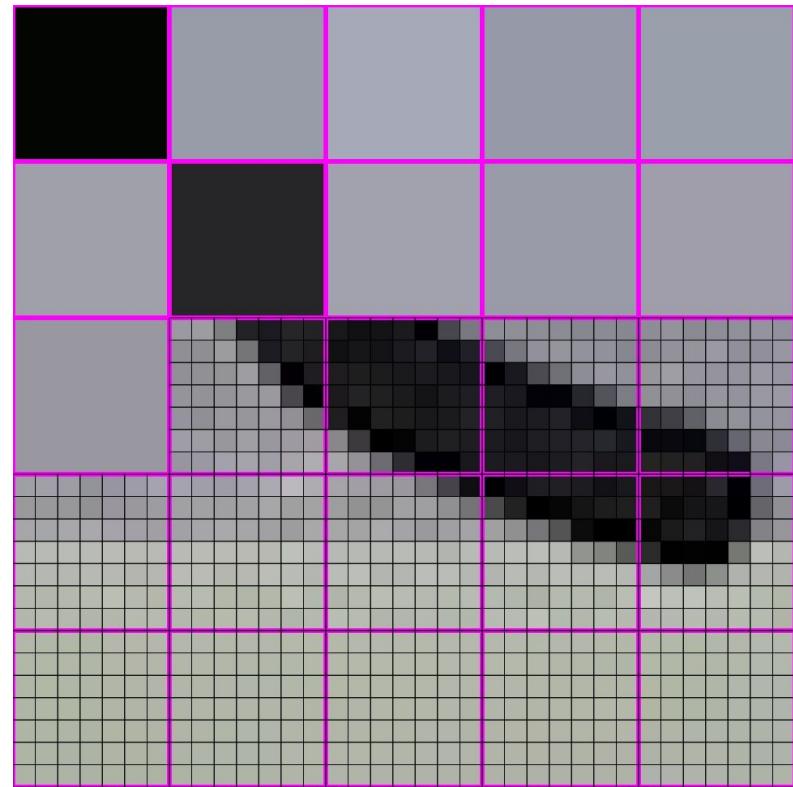
448x448 -> 64x64

---



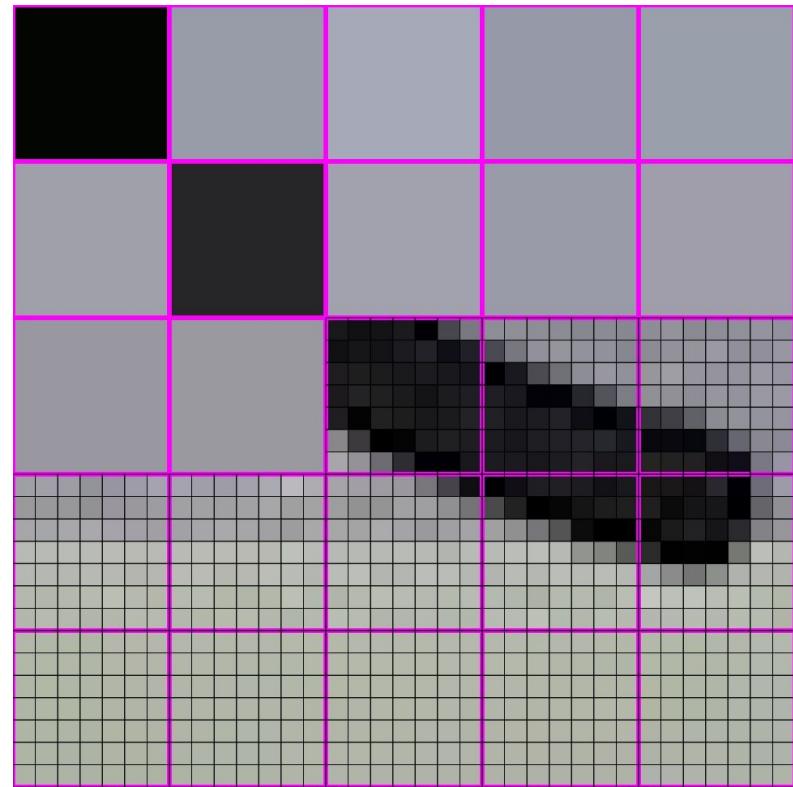
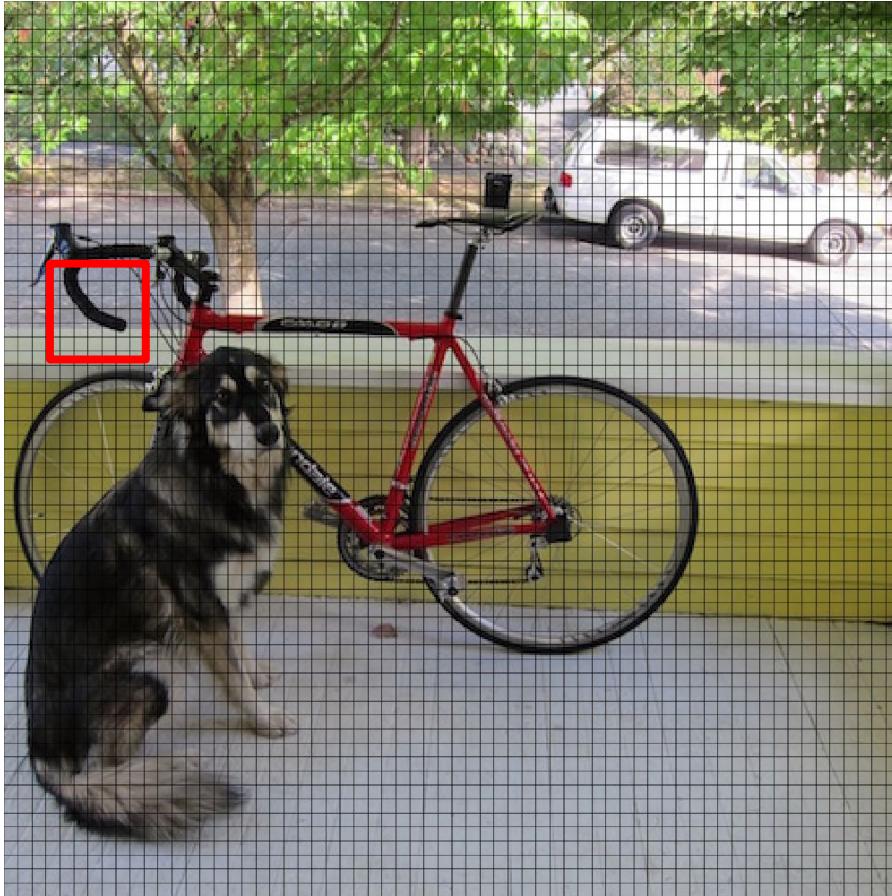
448x448 -> 64x64

---



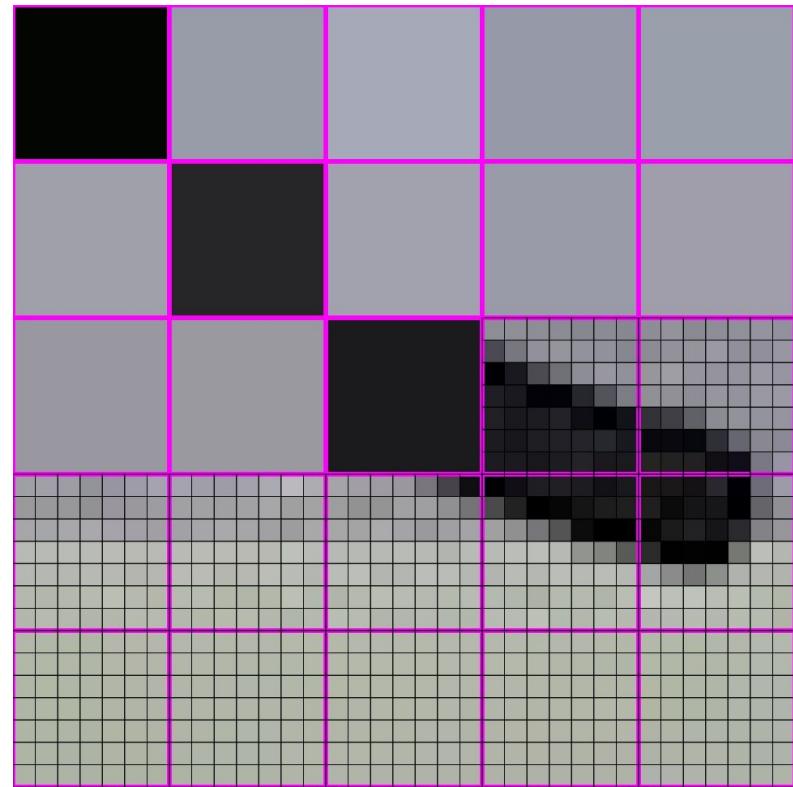
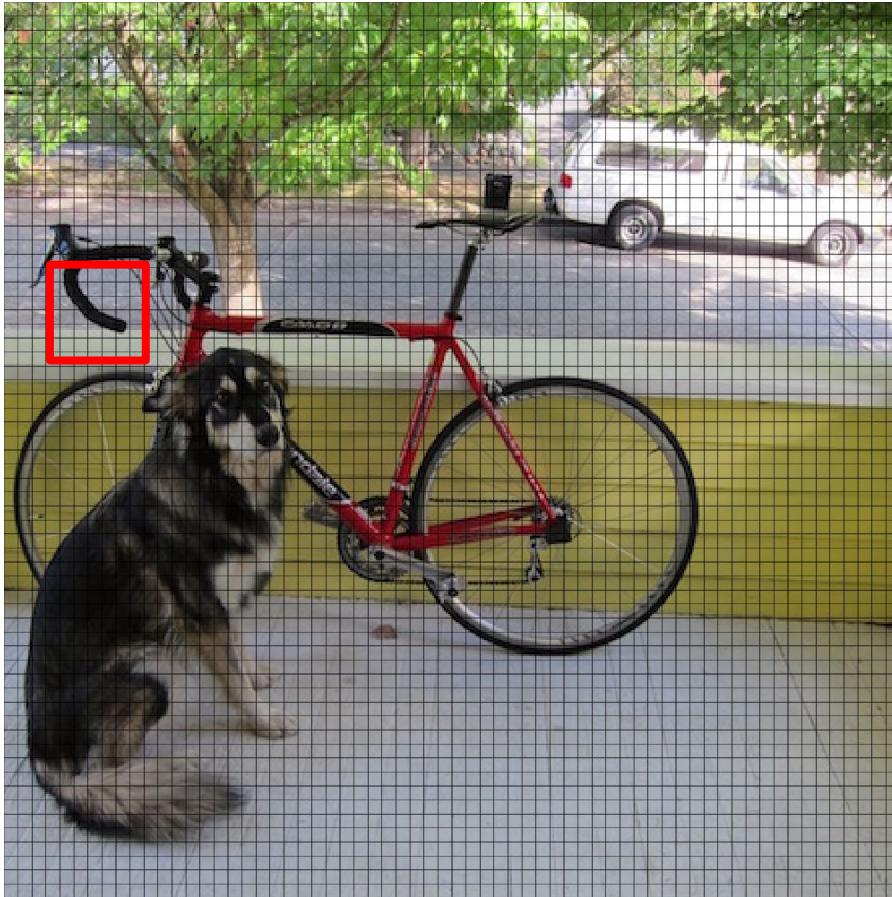
448x448 -> 64x64

---



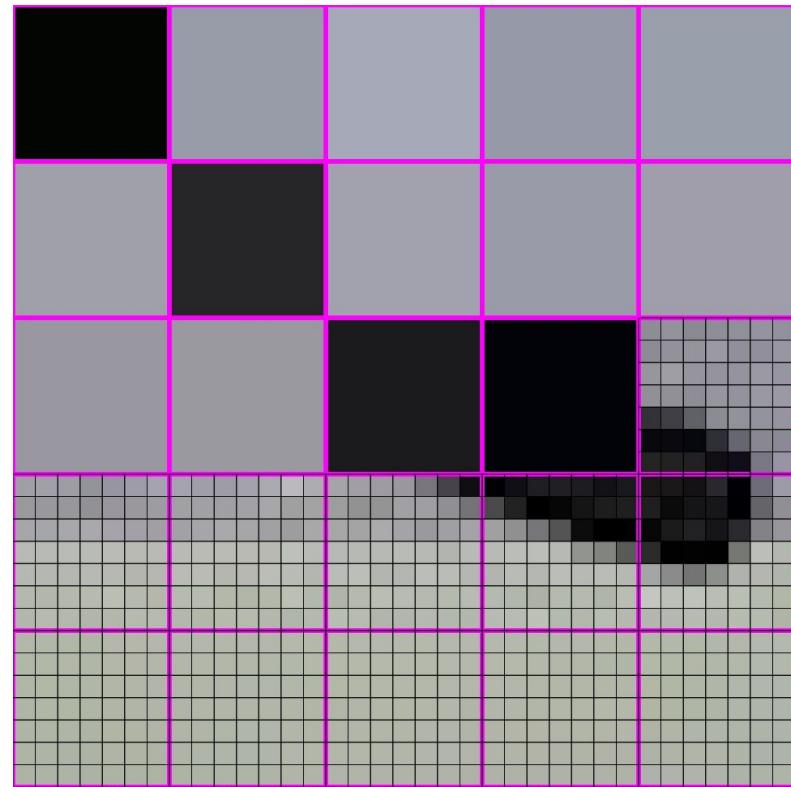
448x448 -> 64x64

---



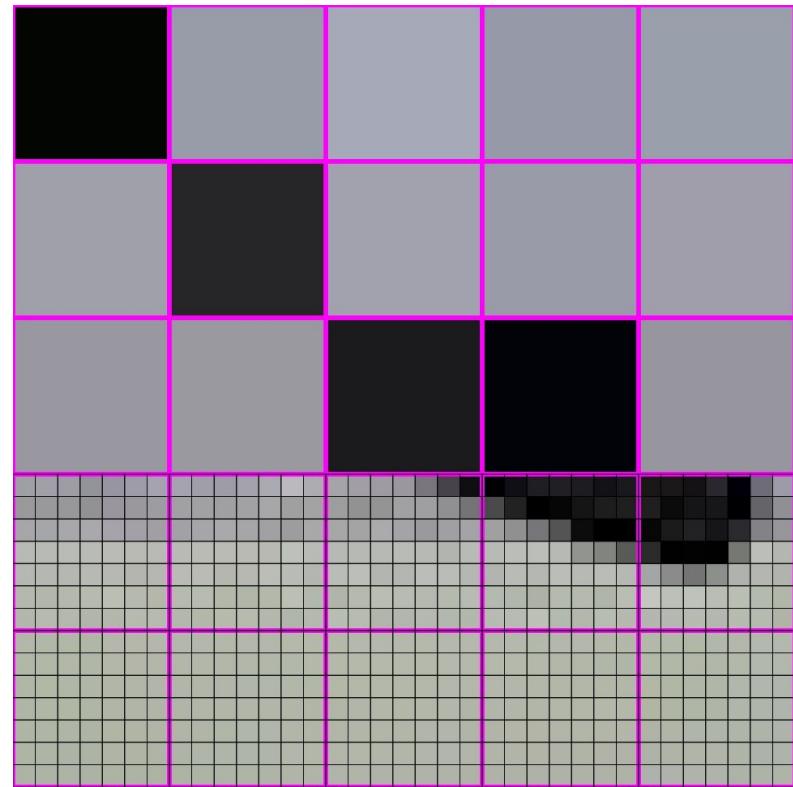
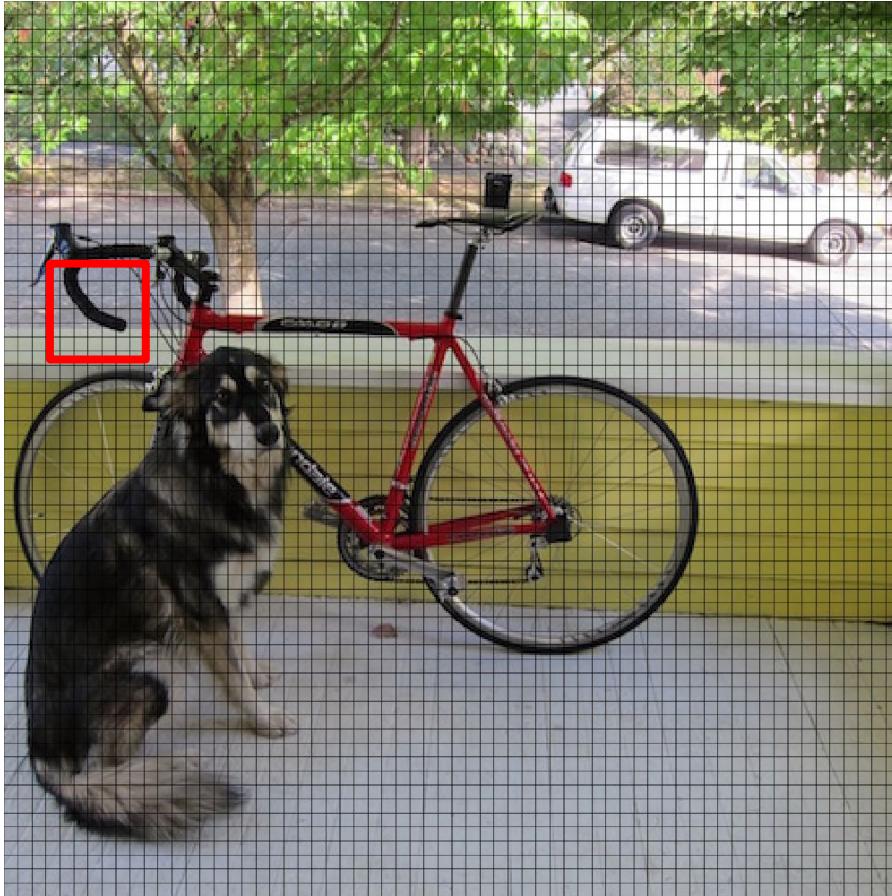
448x448 -> 64x64

---



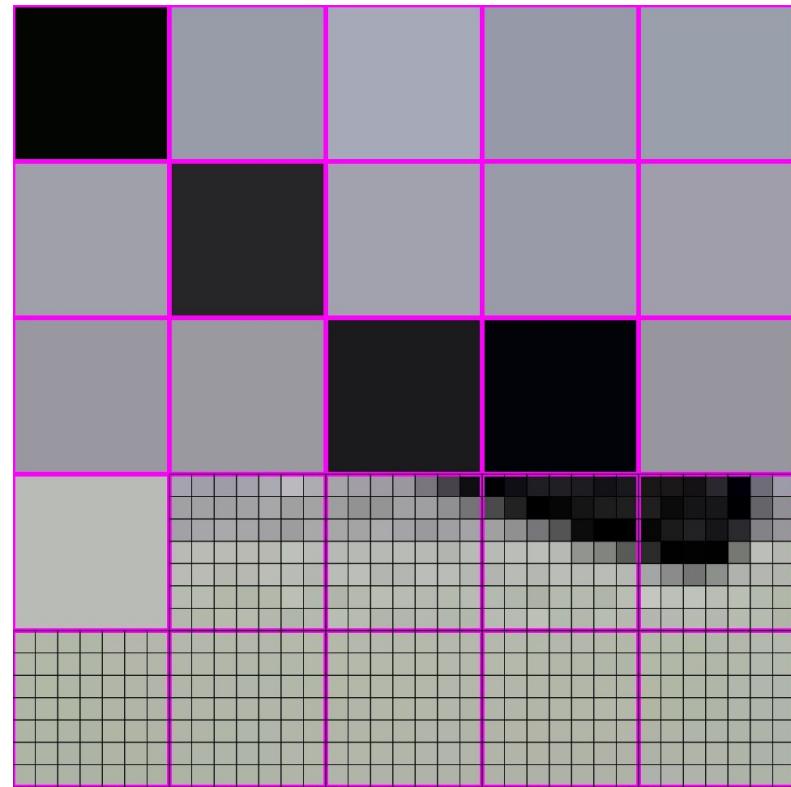
448x448 -> 64x64

---



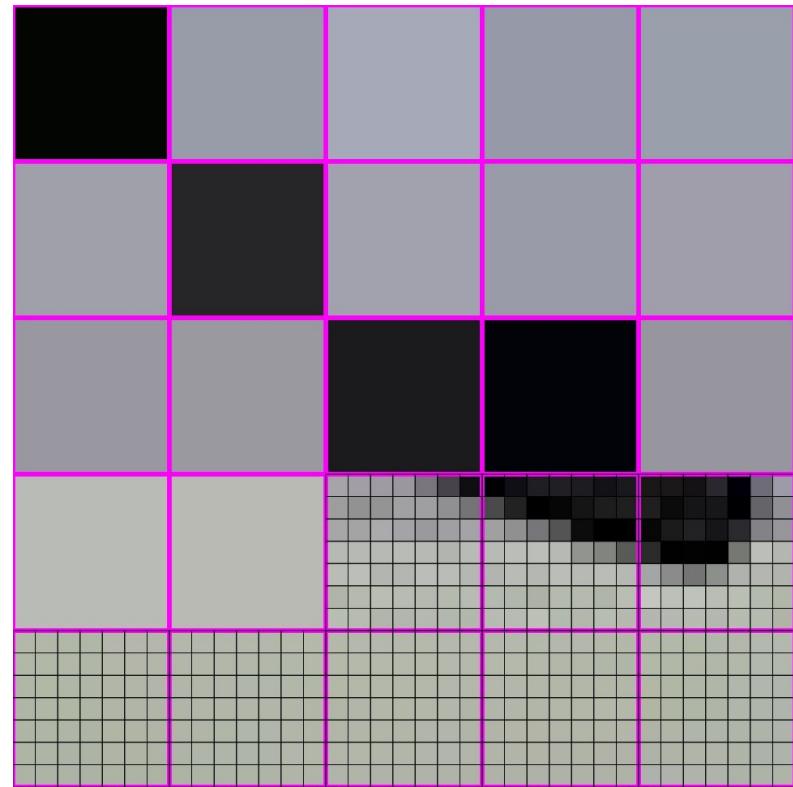
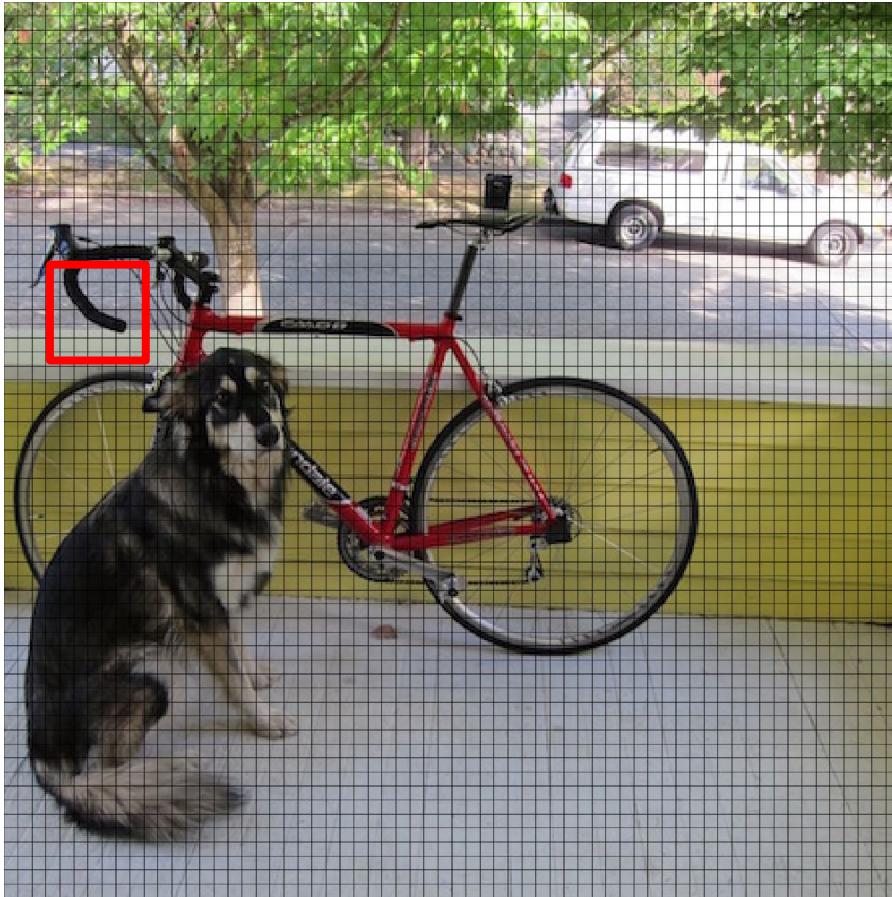
448x448 -> 64x64

---



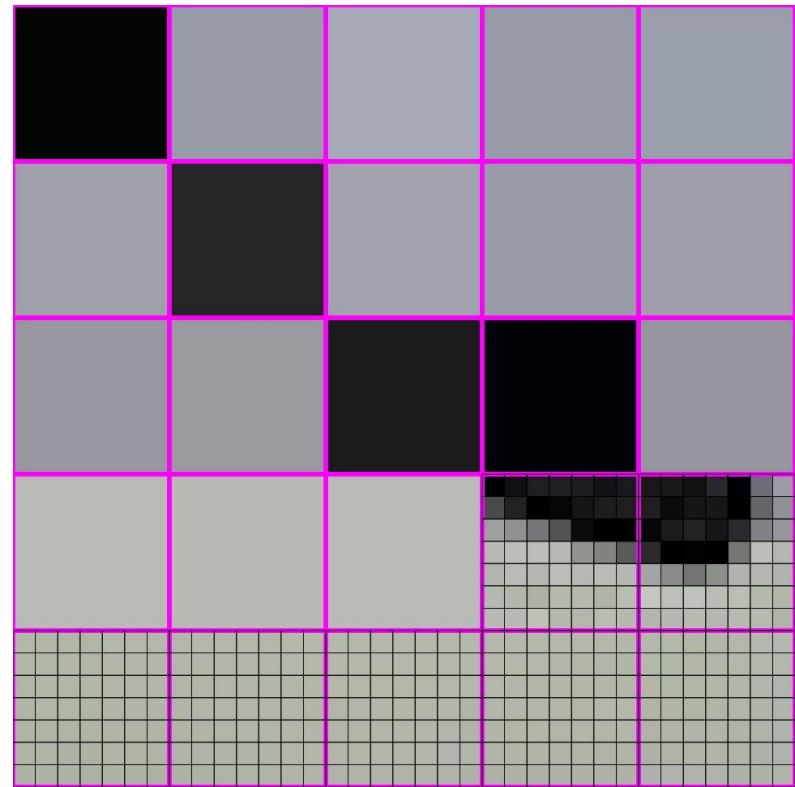
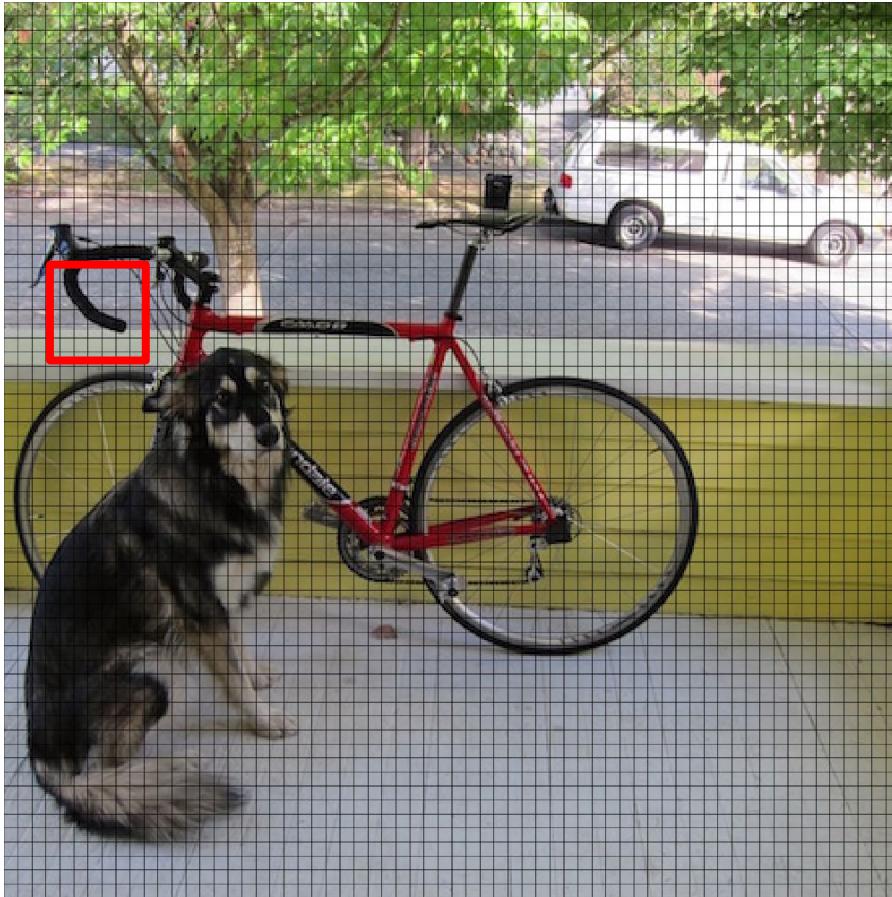
448x448 -> 64x64

---



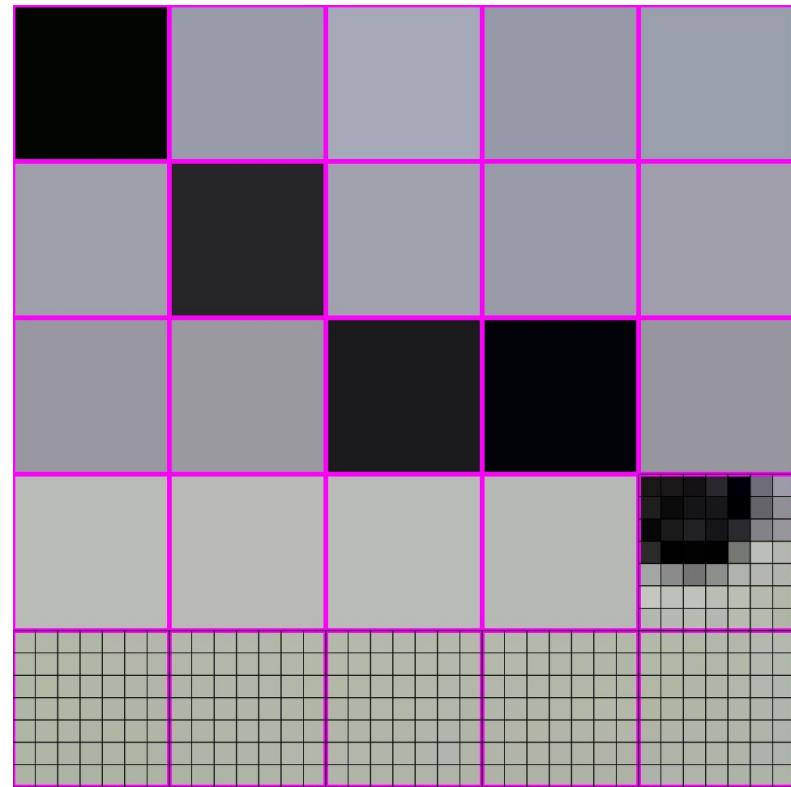
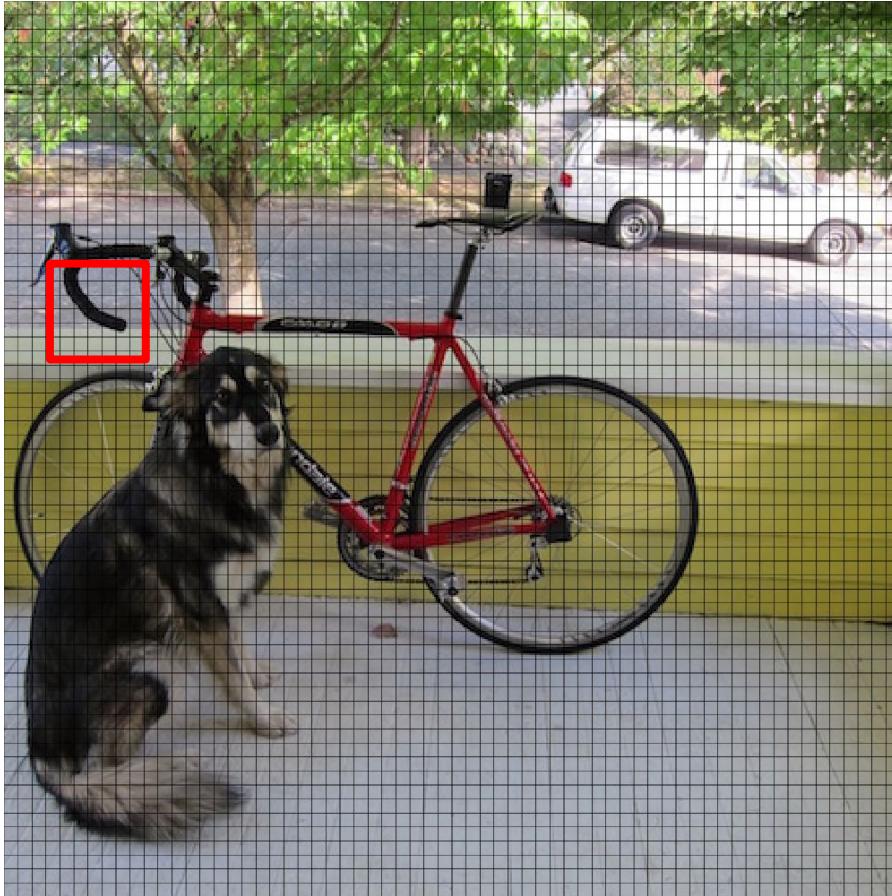
448x448 -> 64x64

---



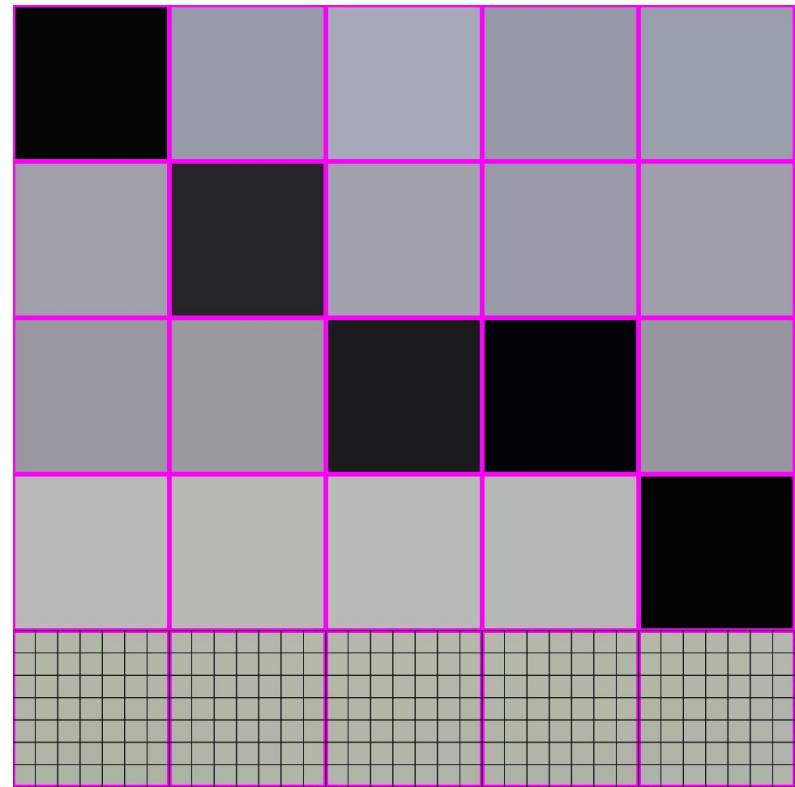
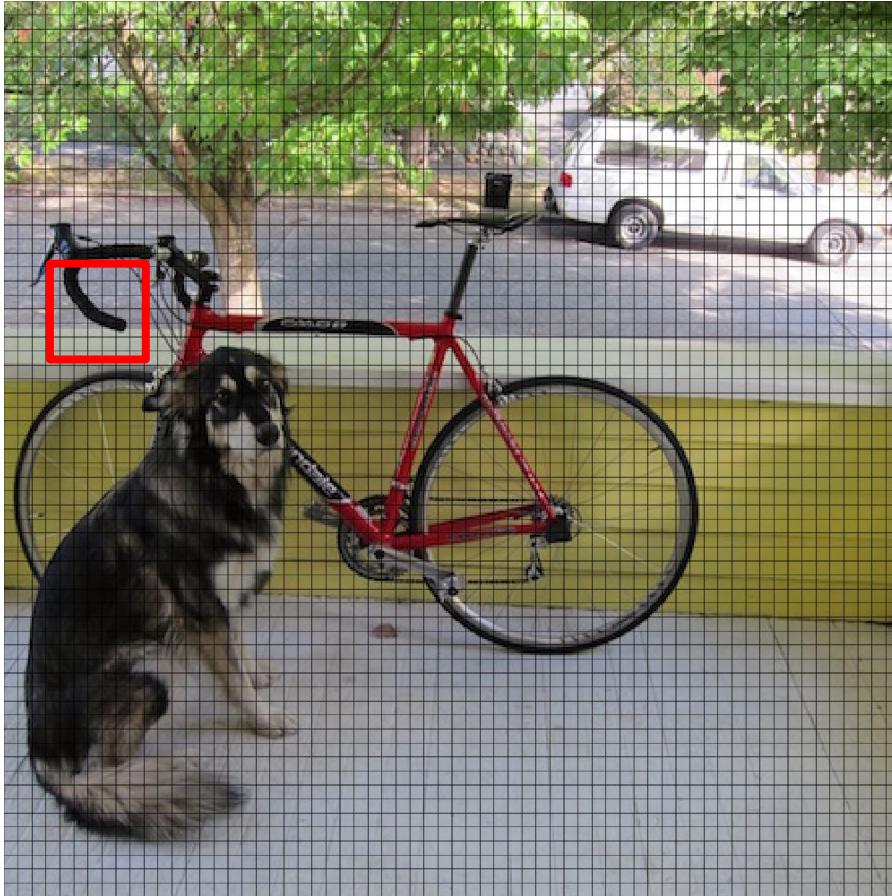
448x448 -> 64x64

---



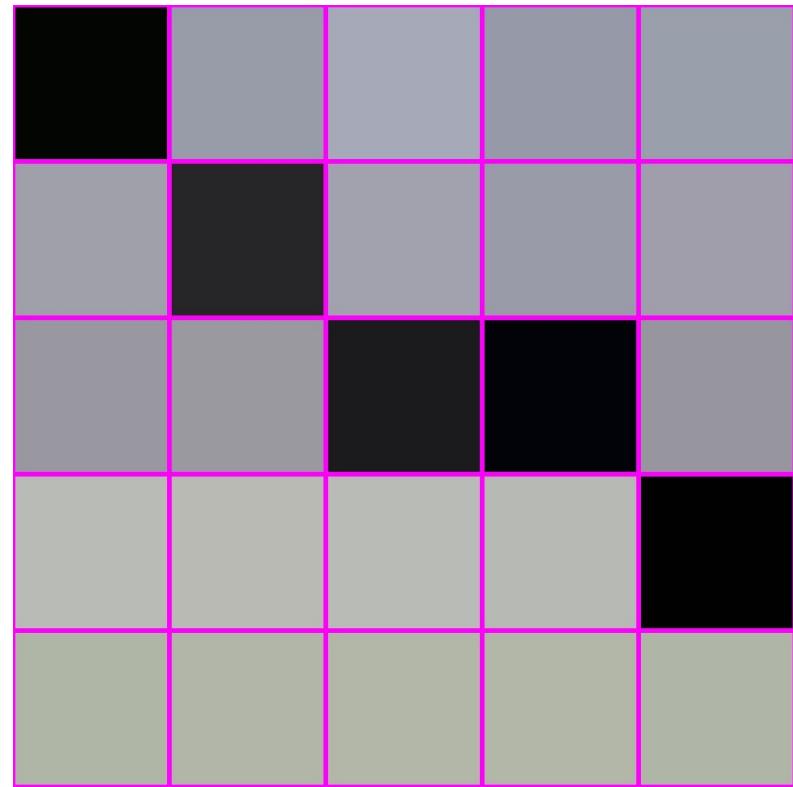
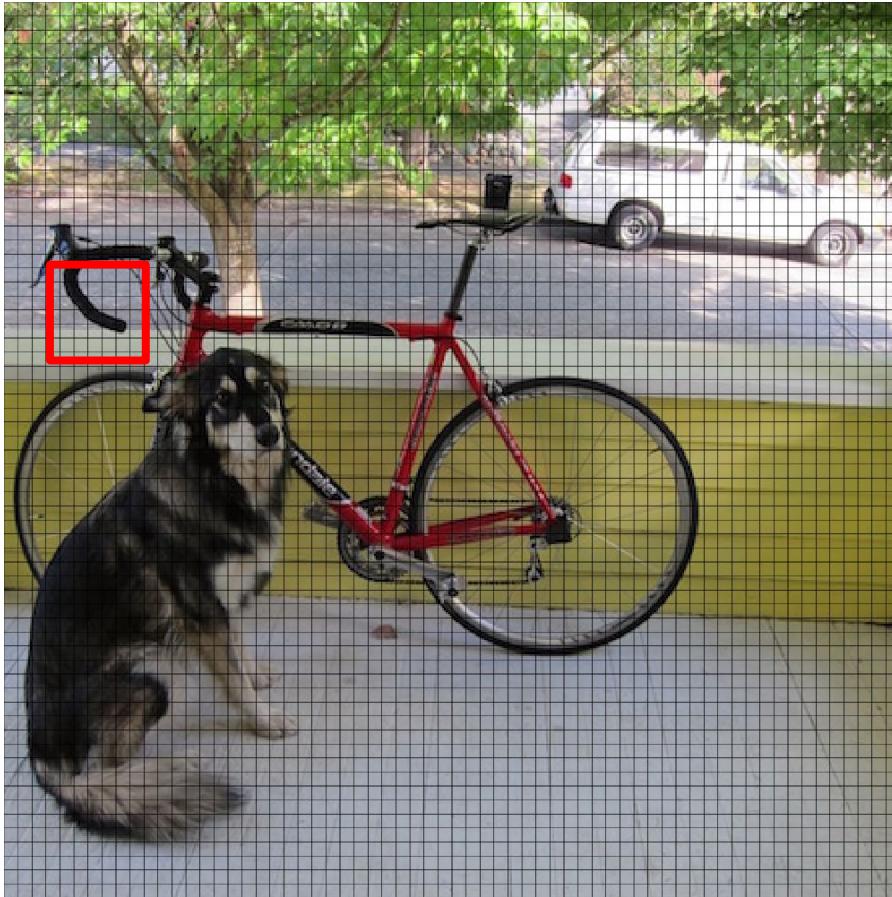
448x448 -> 64x64

---



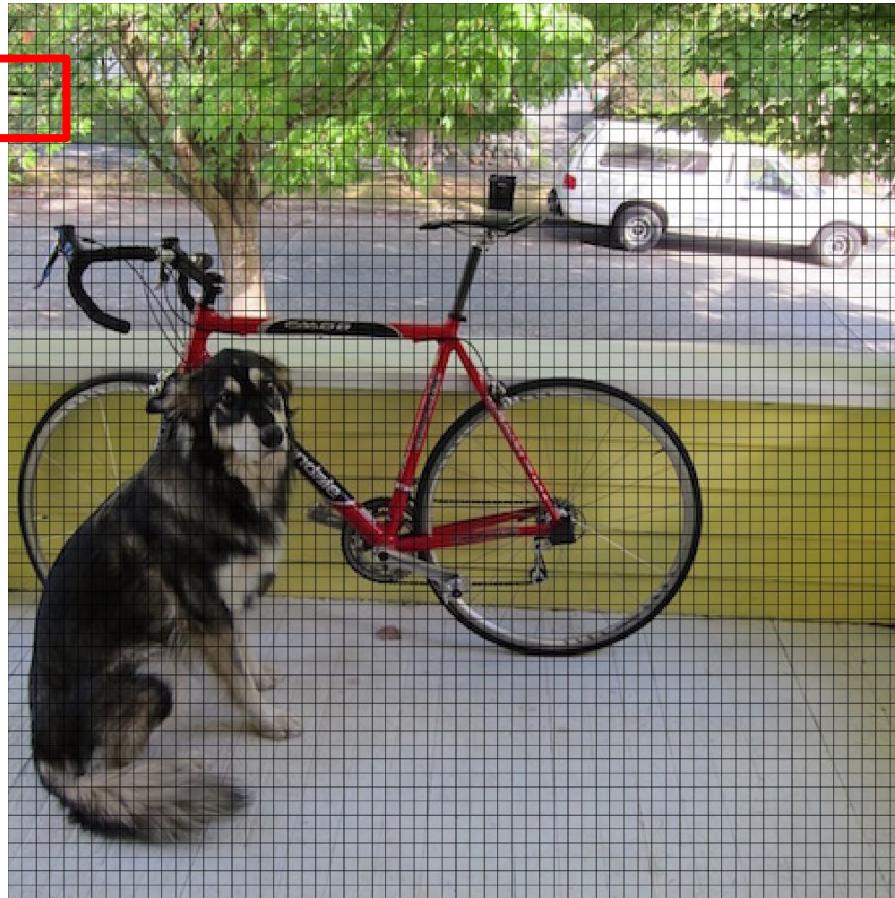
448x448 -> 64x64

---



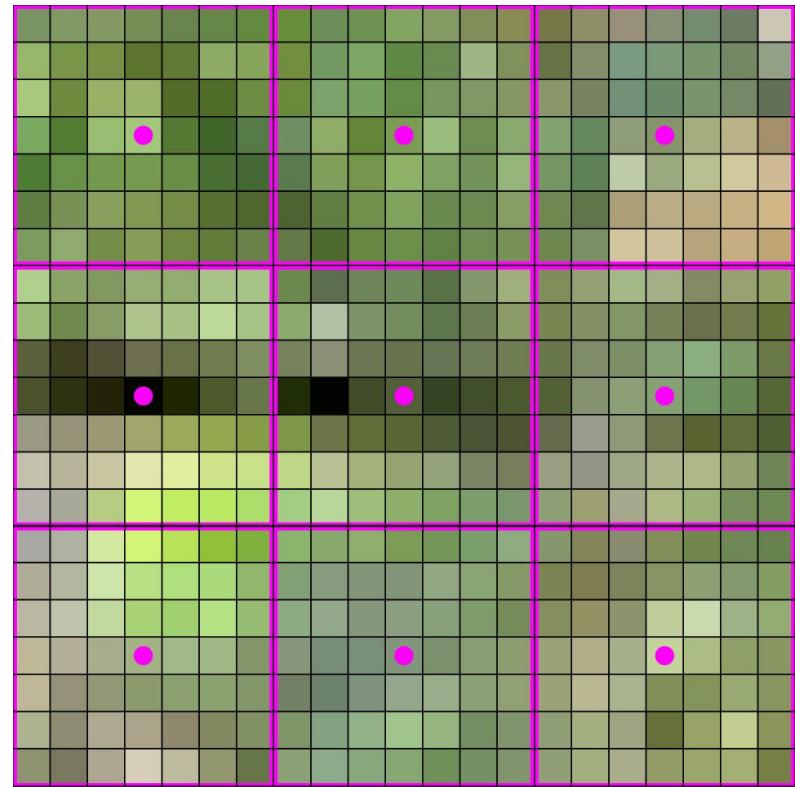
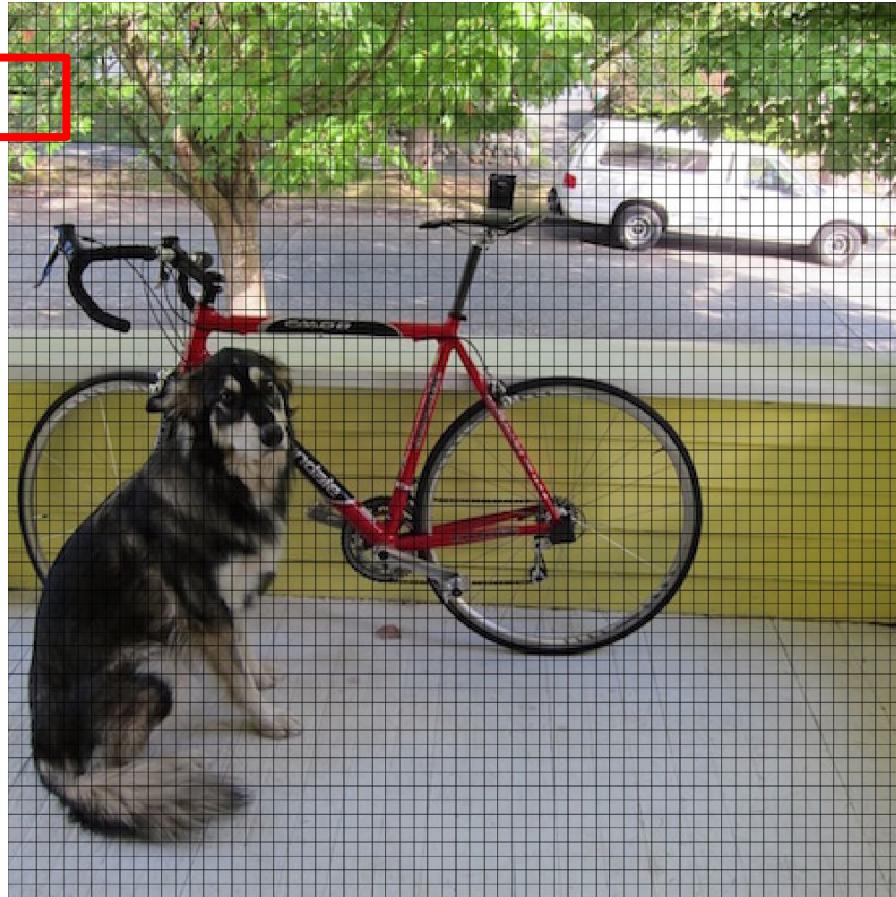
448x448 -> 64x64

---



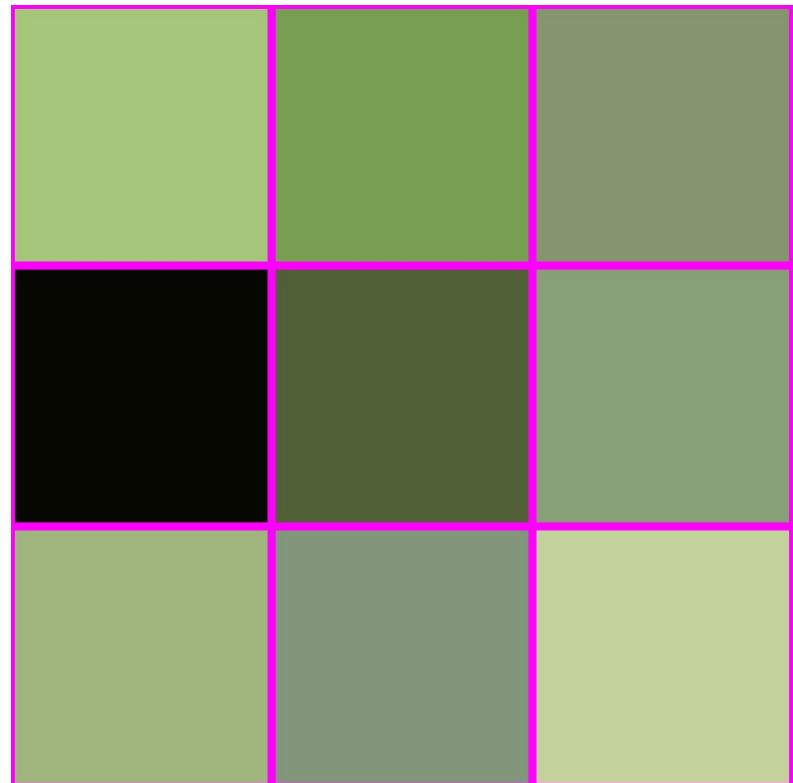
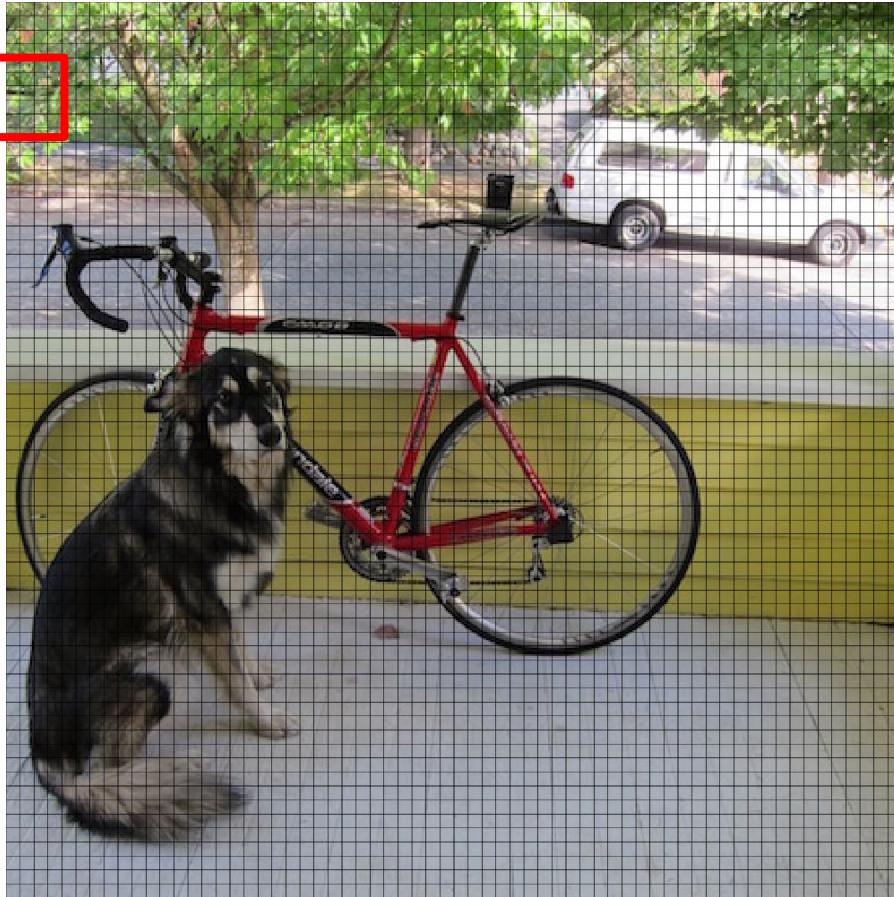
448x448 -> 64x64

---



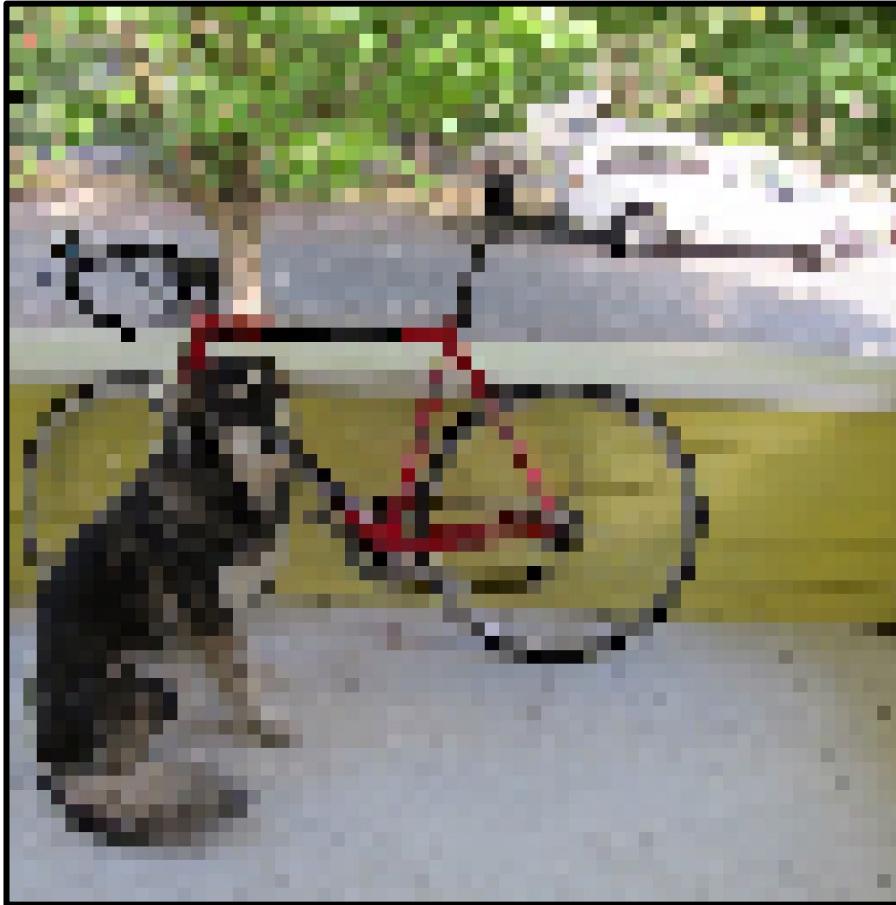
448x448 -> 64x64

---



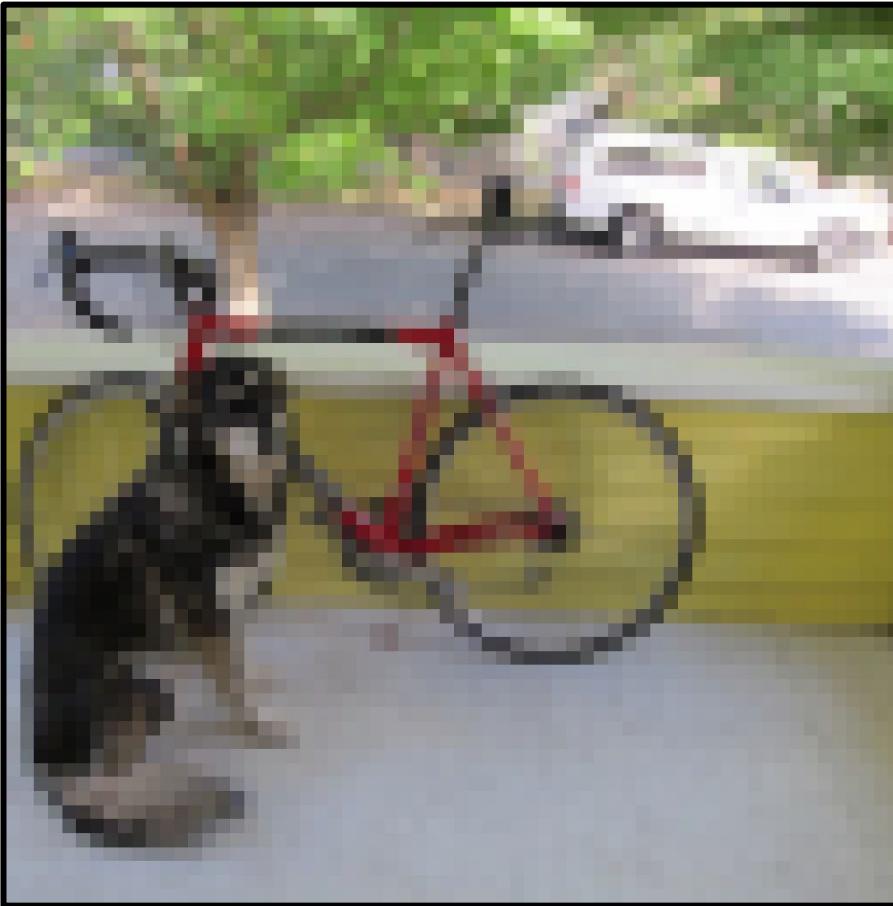
А можно лучше?

---



Можно!

---



# Гораздо лучше!

---



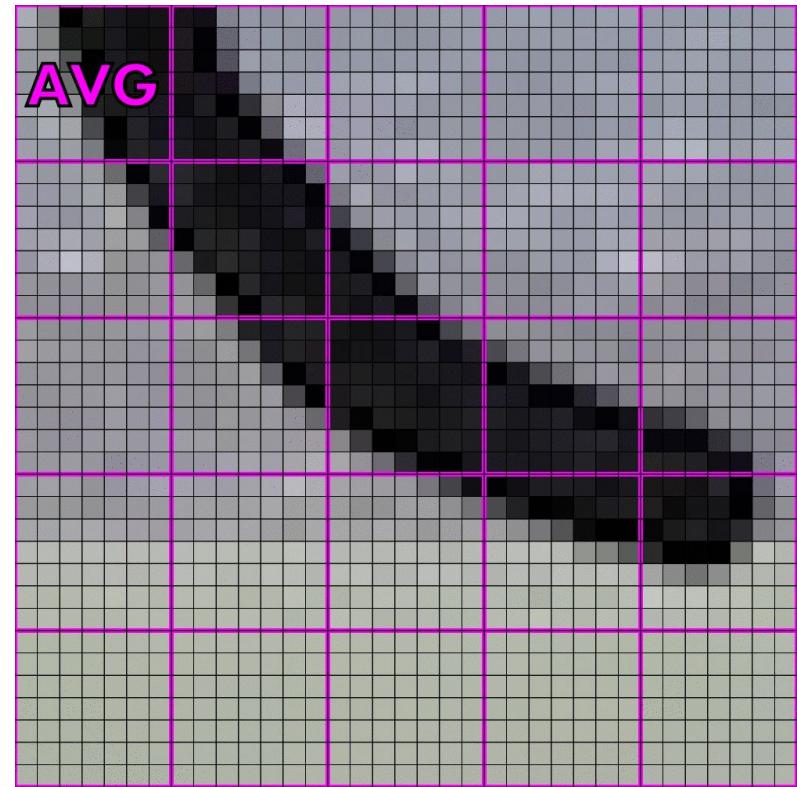
Как?

---



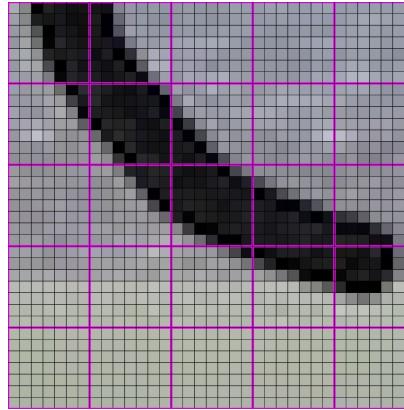
# Как? Усредняем!

---

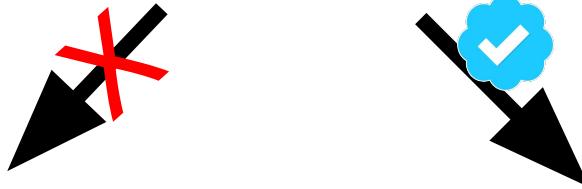
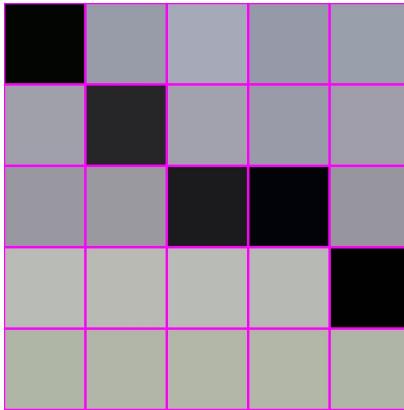


# Как? Усредняем!

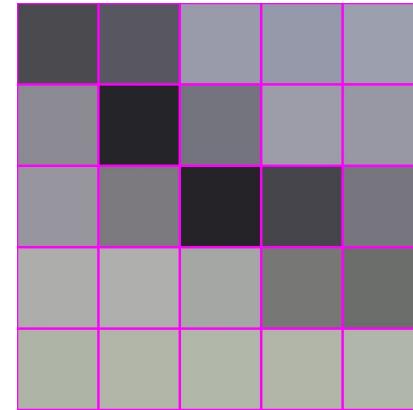
---



“interpolation”

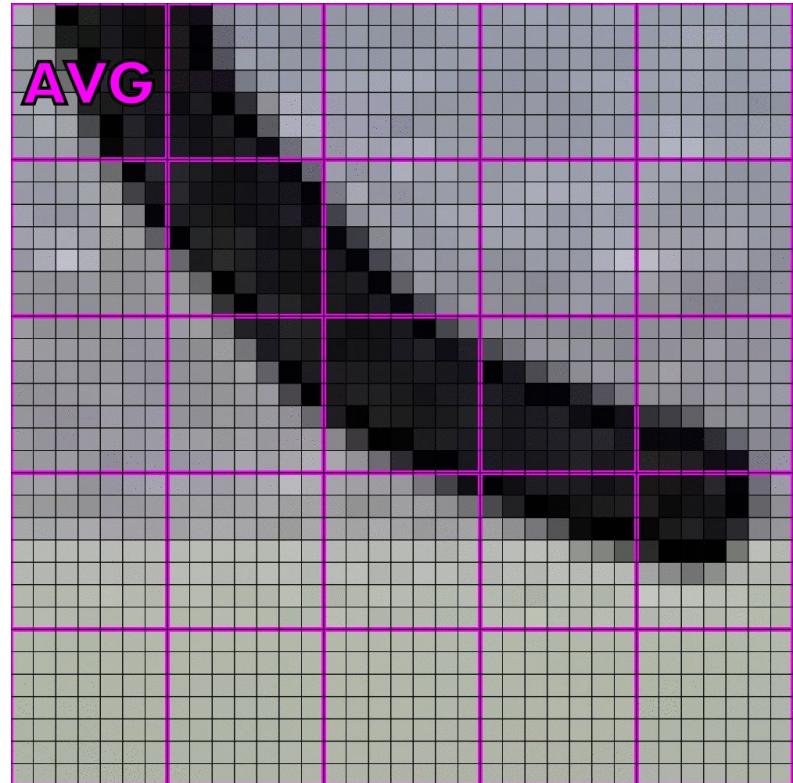


averaging



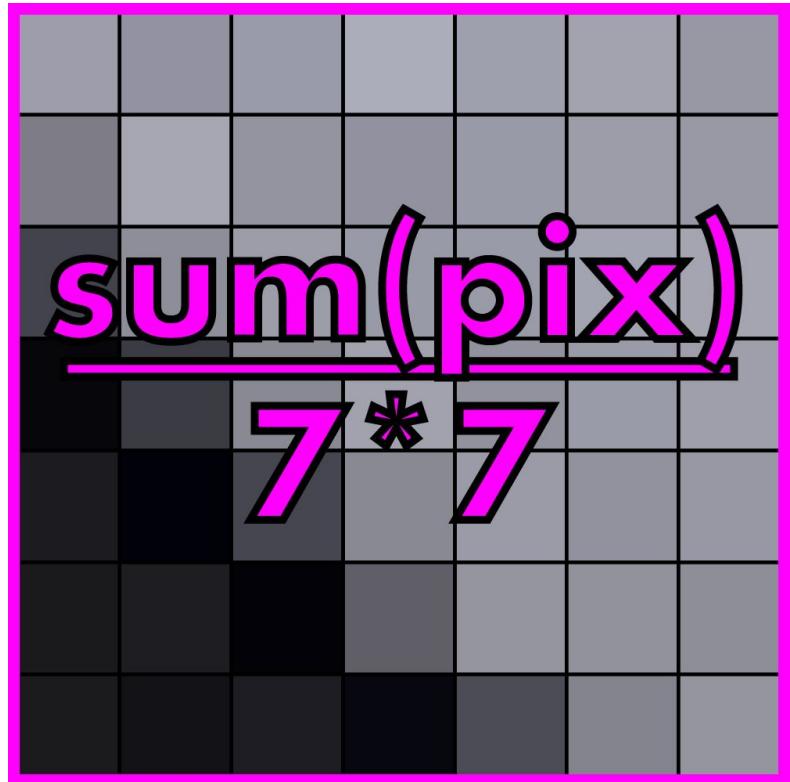
# Как? Усредняем!

---



## Усреднение - взвешенная сумма

---



## Усреднение - взвешенная сумма

$$\text{sum} \left[ \frac{1}{49} \right]$$

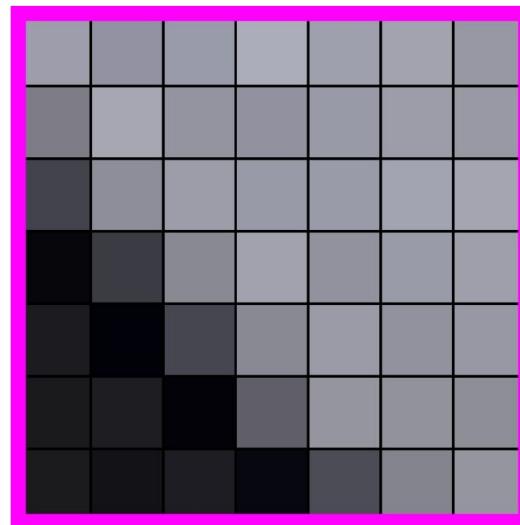
# Назовем эту операцию “конволюцией”

---

*Фильтр или ядро*

$\frac{1}{49}$

$1 \times$						
$1 \times$						
$1 \times$						
$1 \times$						
$1 \times$						
$1 \times$						
$1 \times$						



# Конволюция на больших изображениях

---

$\frac{1}{49}$

$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							
$1 \times$							

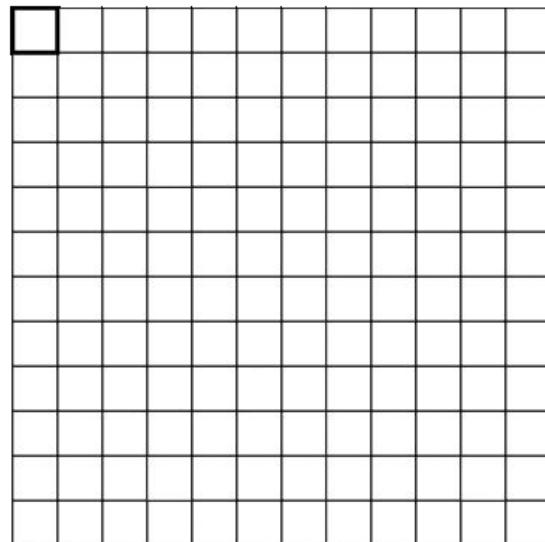
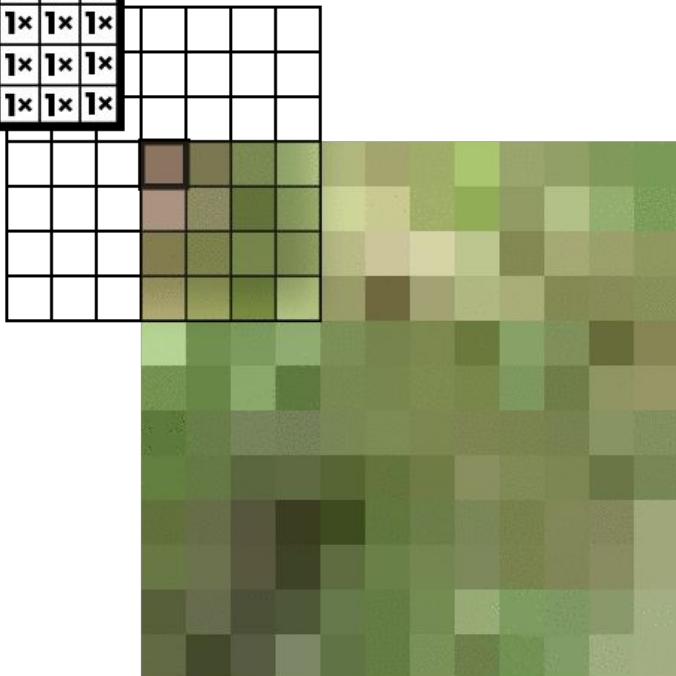


# Фильтр скользит вдоль изображения

---

1  
49

1x							
1x							
1x							
1x							
1x							
1x							
1x							
1x							

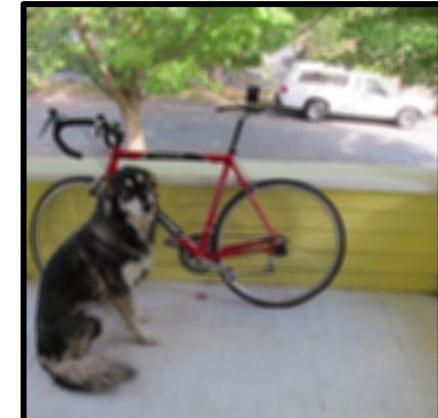
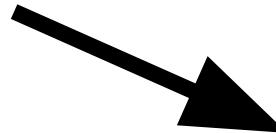


# Конволюция на больших изображениях

---

$\frac{1}{49}$

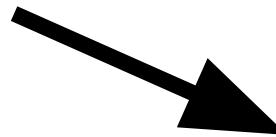
1x							
1x							
1x							
1x							
1x							
1x							
1x							
1x							



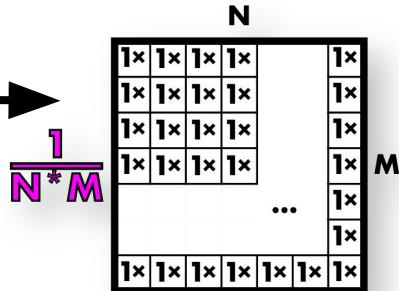
# Это называется box filter

$\frac{1}{49}$

1x						
1x						
1x						
1x						
1x						
1x						
1x						

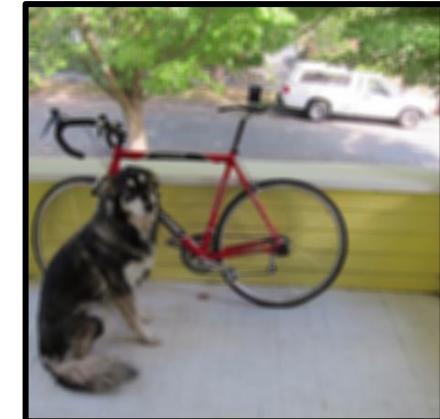
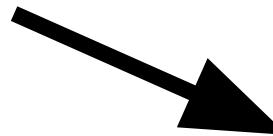


Box filters



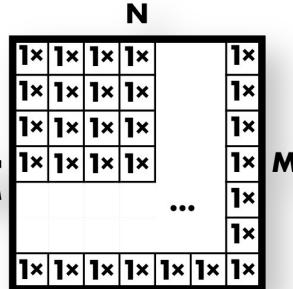
# Box filter сглаживает изображение

1  
49

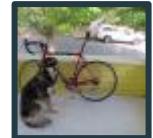
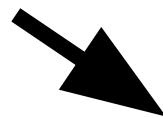


## Box filters

1  
N \* M

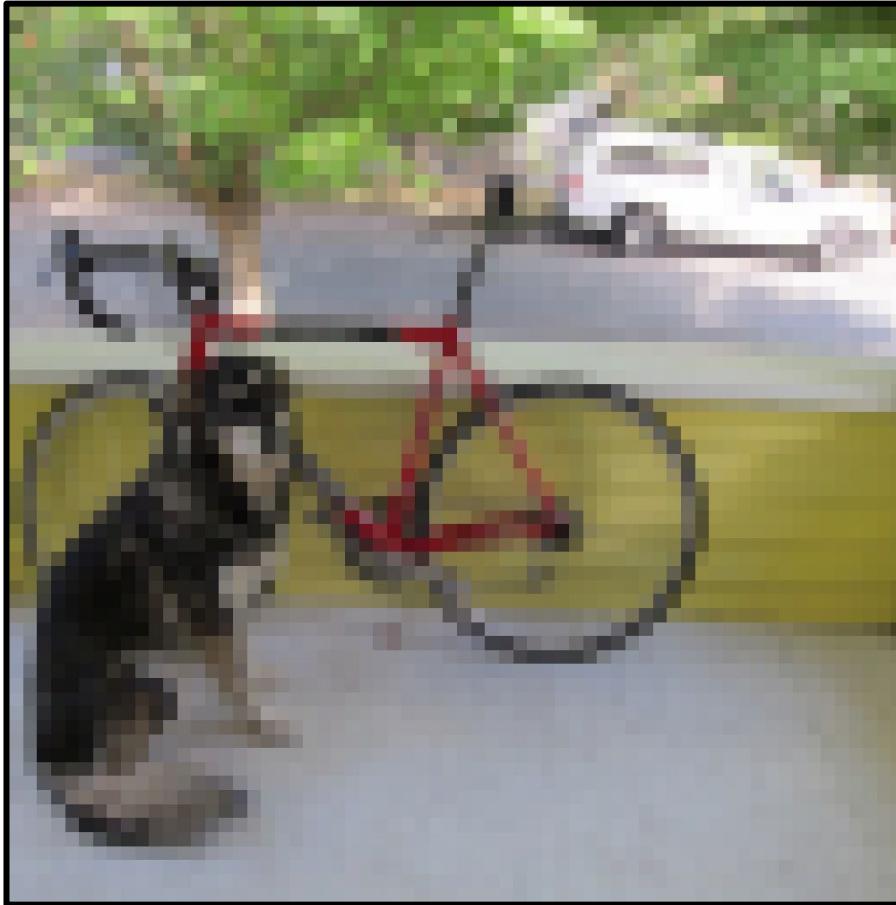


Теперь можно заресайзить сглаженное изображение



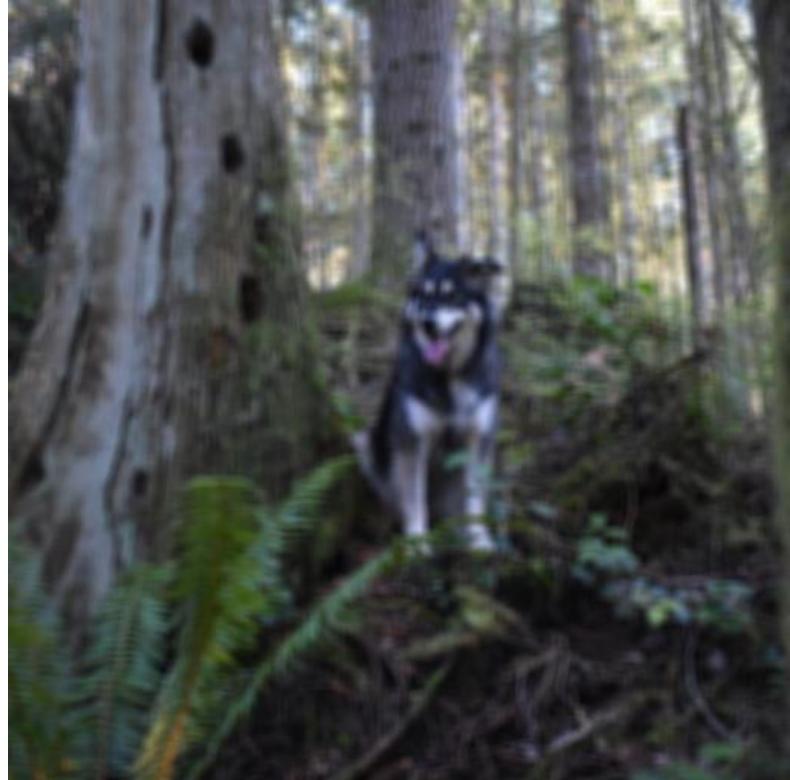
Лучше!

---



# У бокс фильтров есть артефакты

---

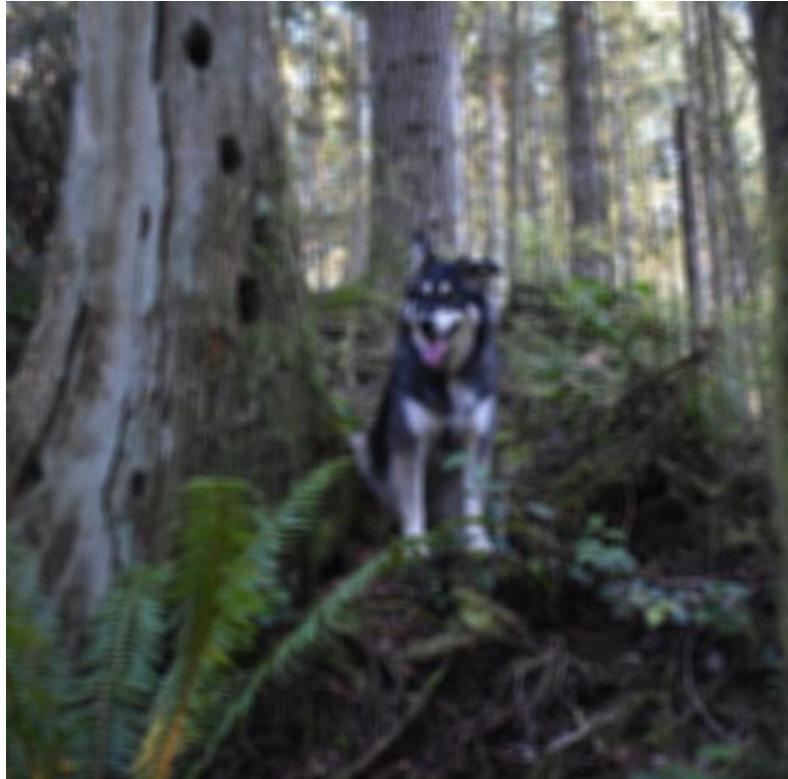
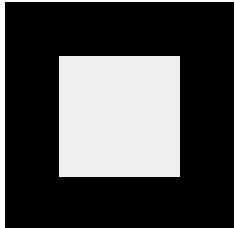


У бокс фильтров есть артефакты

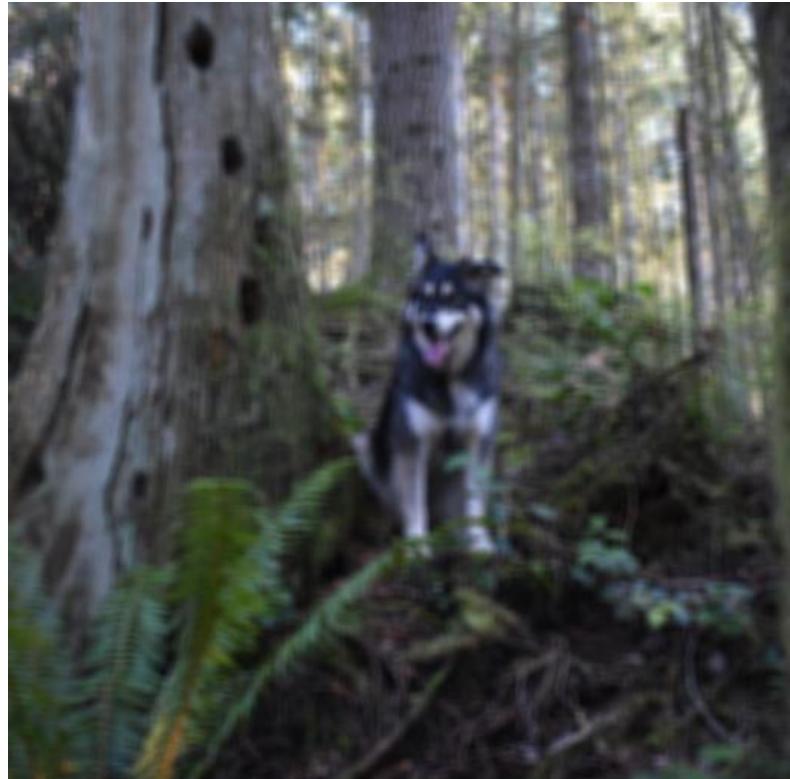
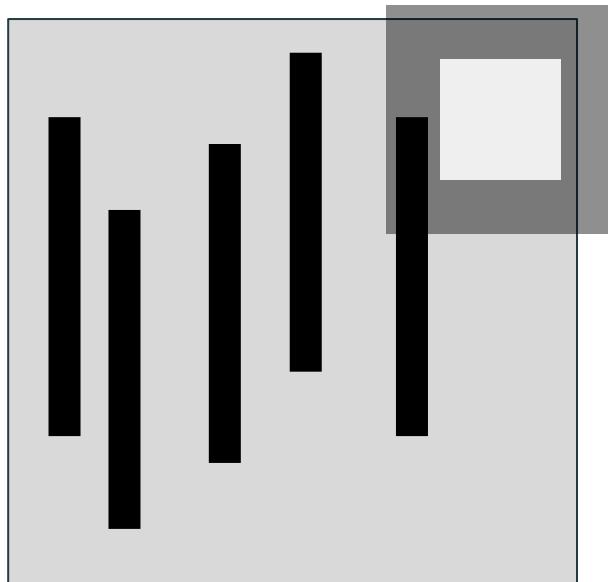


# Бокс фильтры: вертикальные и горизонтальные полосы

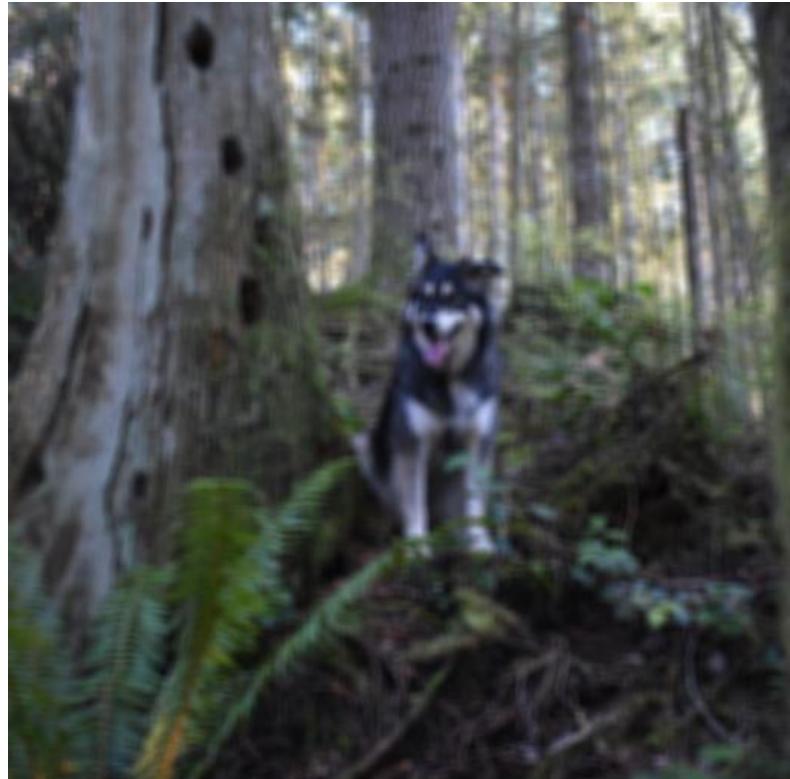
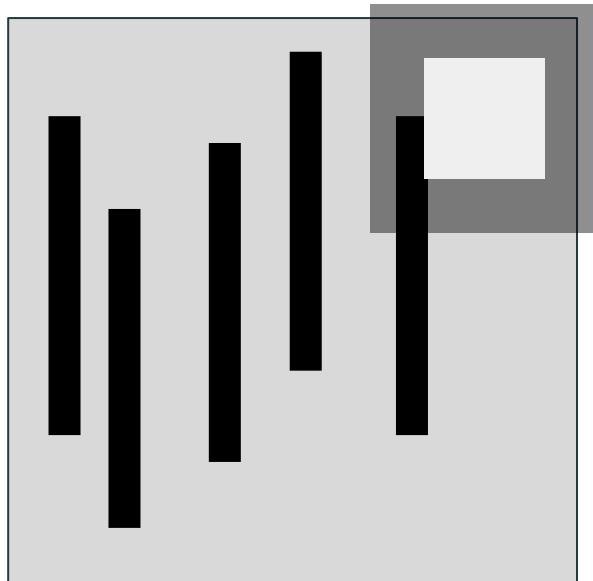
---



## Бокс фильтры: вертикальные и горизонтальные полосы

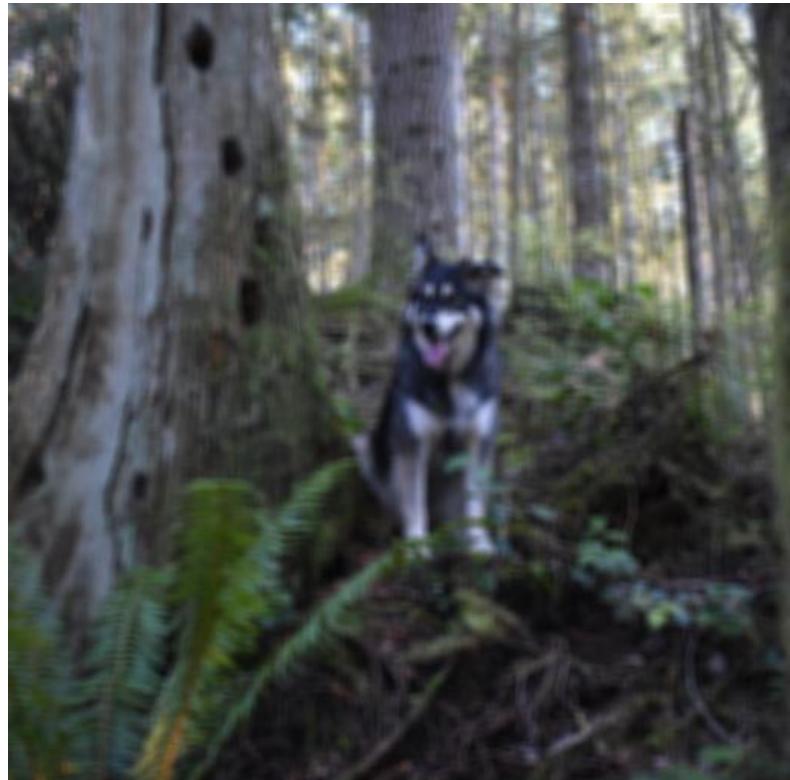
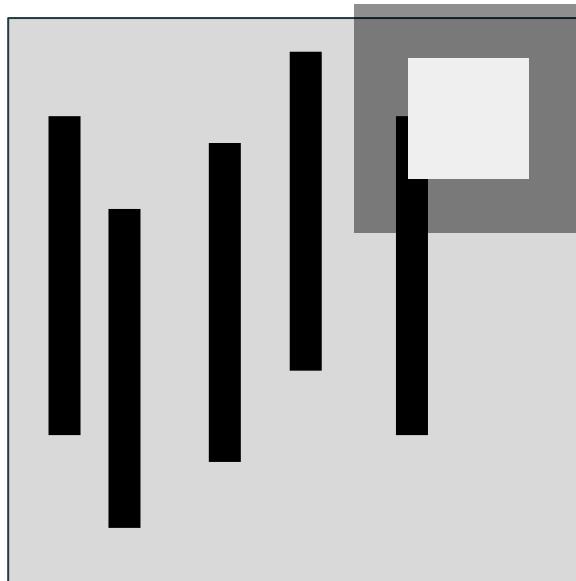


## Бокс фильтры: вертикальные и горизонтальные полосы



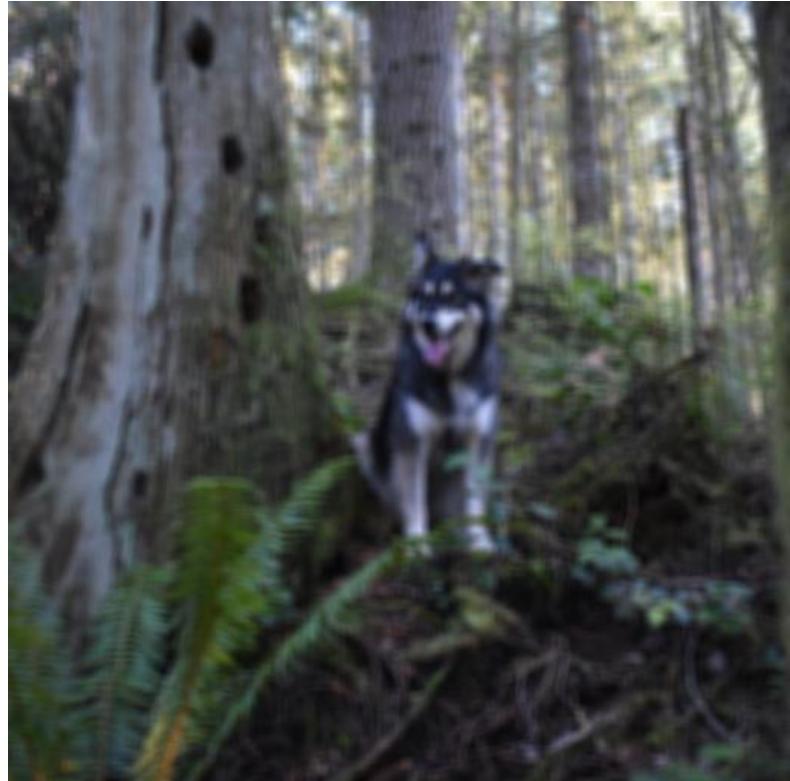
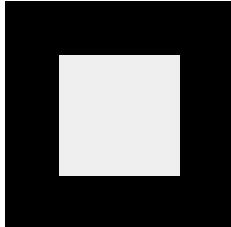
## Бокс фильтры: вертикальные и горизонтальные полосы

---



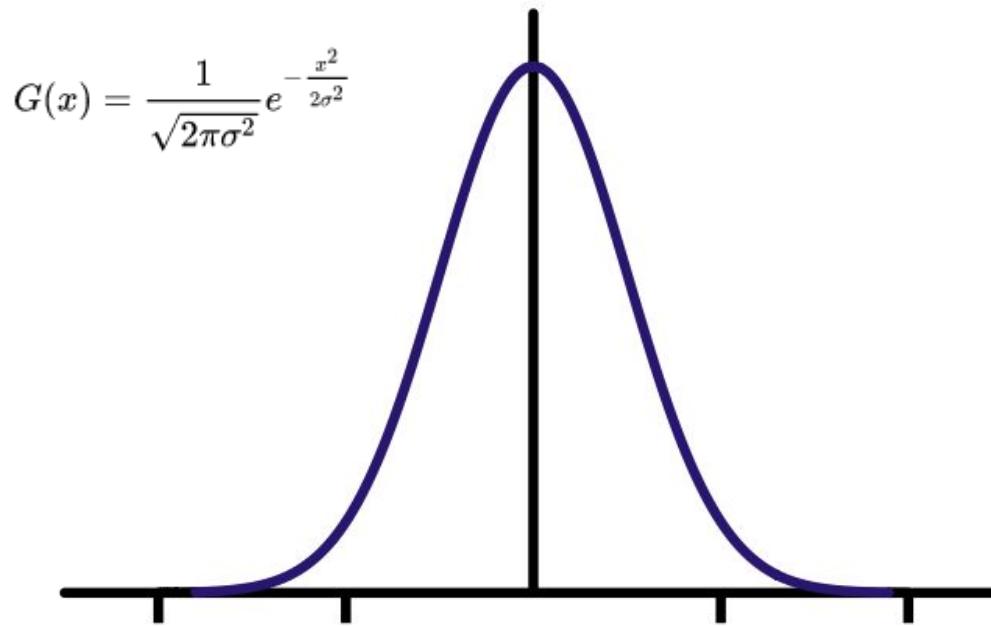
# Мы хотим более “гладкий” кернел

---



# Гауссиана

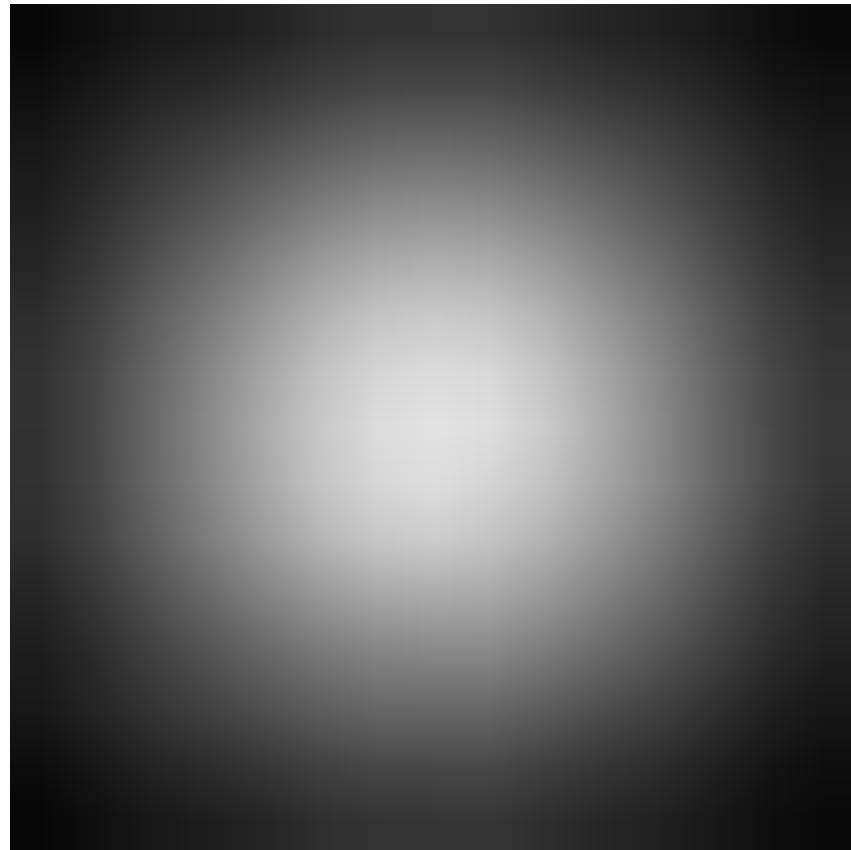
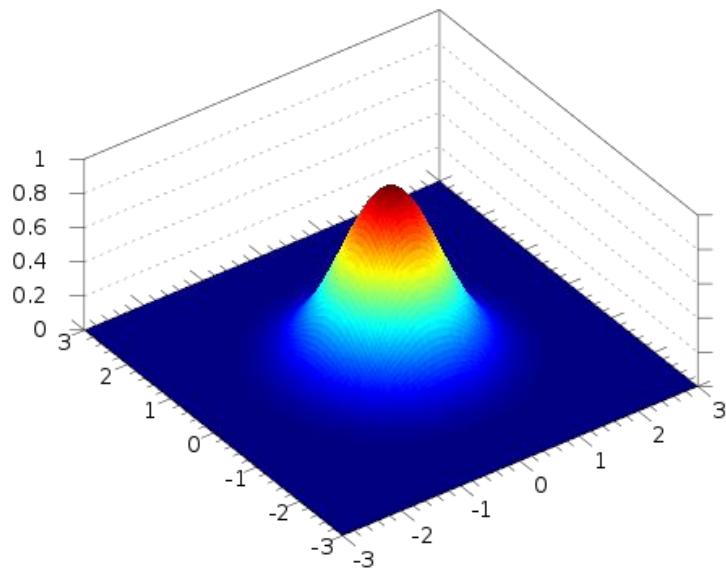
---



## 2д гауссиана

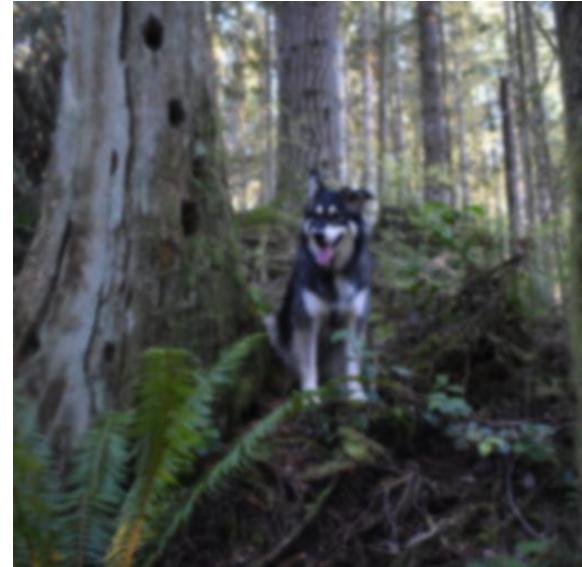
---

$$g(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$



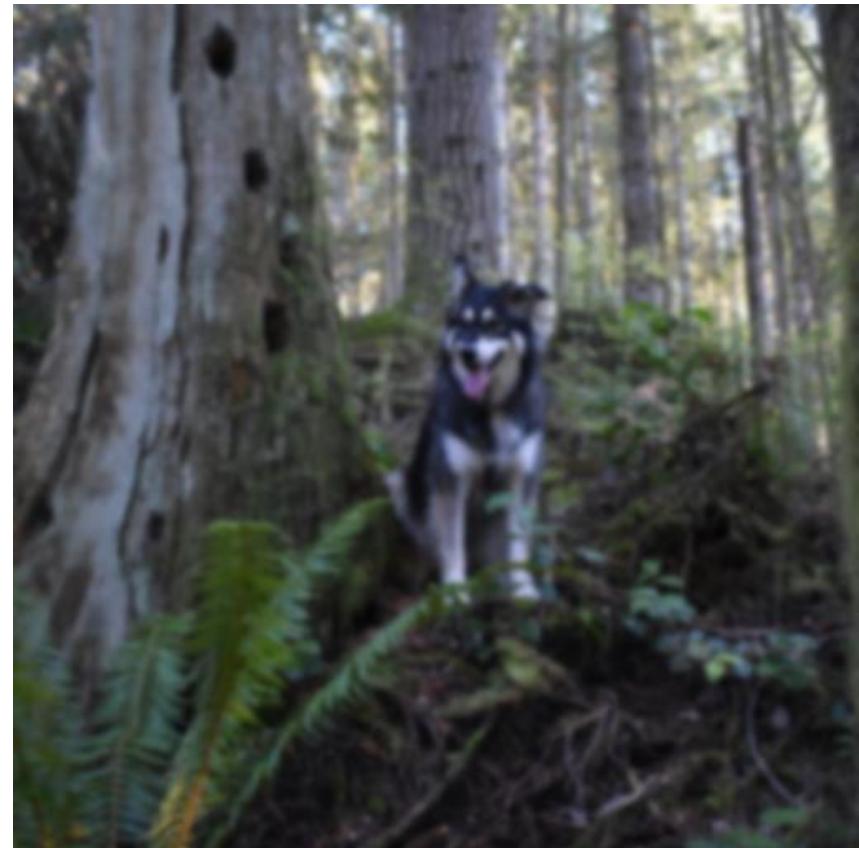
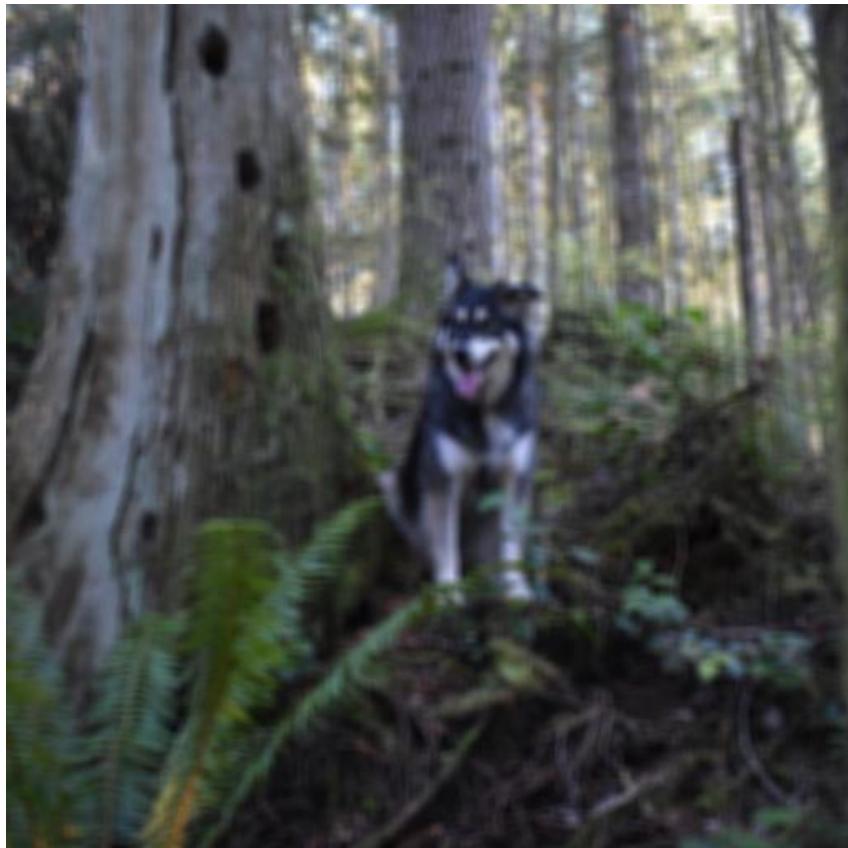
# Лучшее сглаживание с Гауссианой

---

 $*$  $=$ 

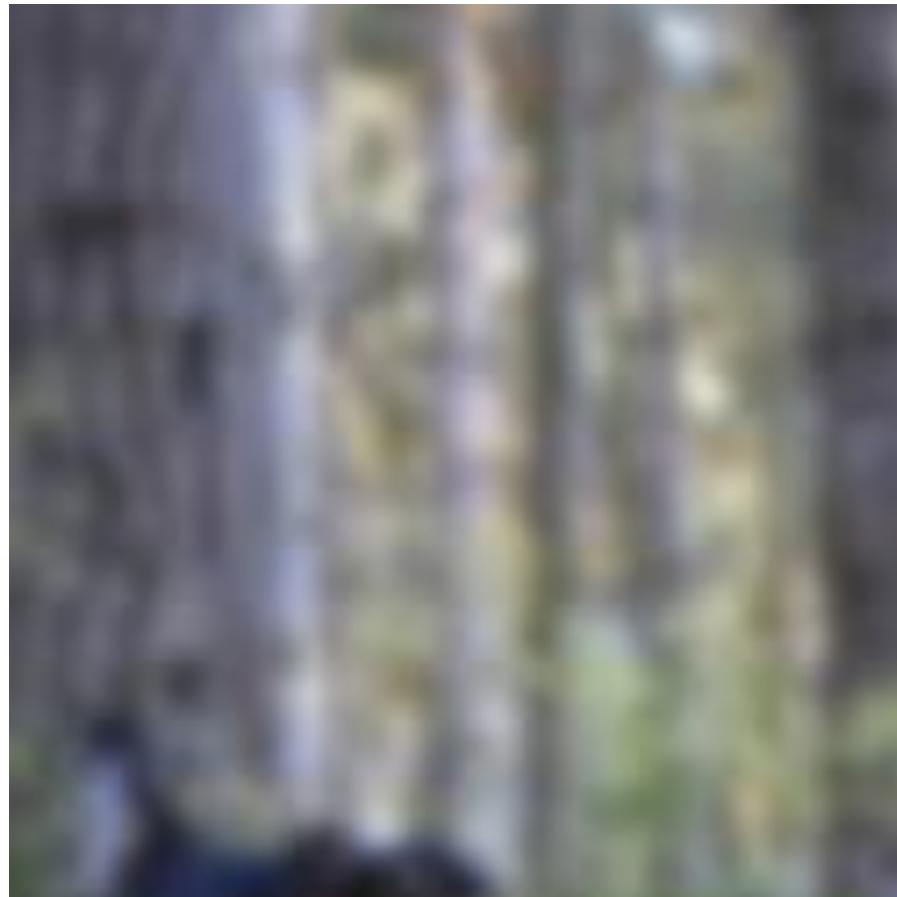
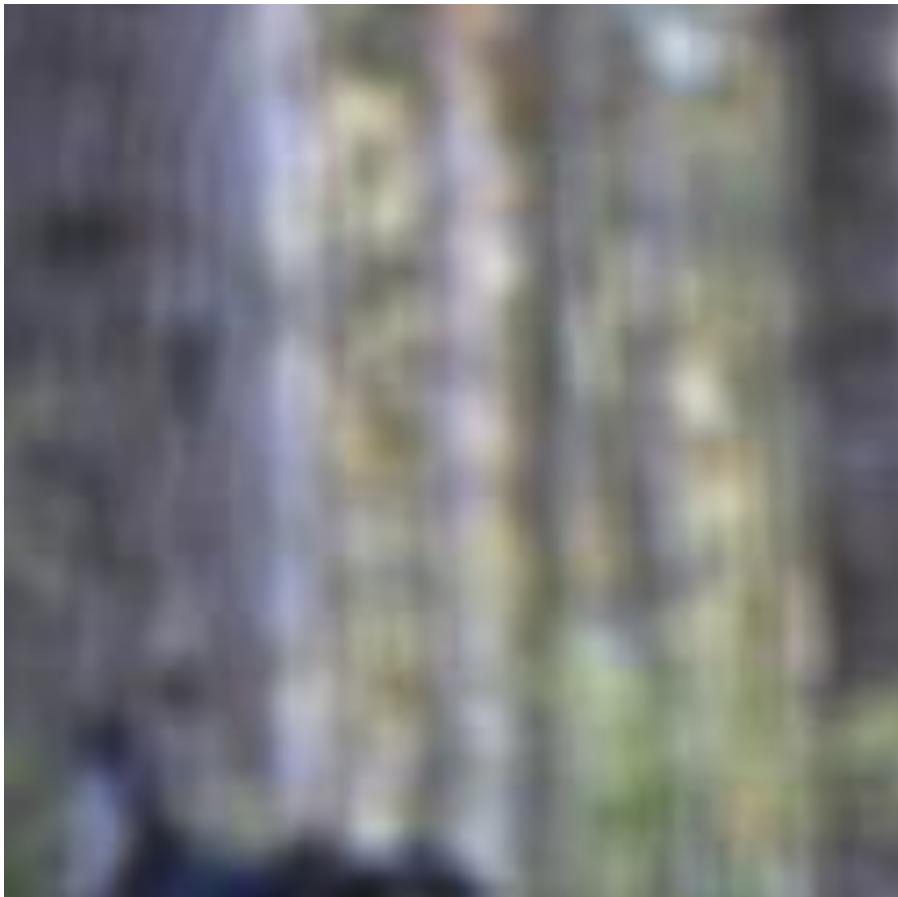
# Лучшее сглаживание с Гауссианой

---



# Лучшее сглаживание с Гауссианой

---



# Медианный фильтр

Brightness Values

	-1	0	1	← i
-1	10	30	5	
0	20	200	20	
1	15	10	30	
↑				
j				



Brightness Values in Order

5 10 10 15 20 20 30 30 200



Median

Mean



3x3

Gaussian



5x5

Median

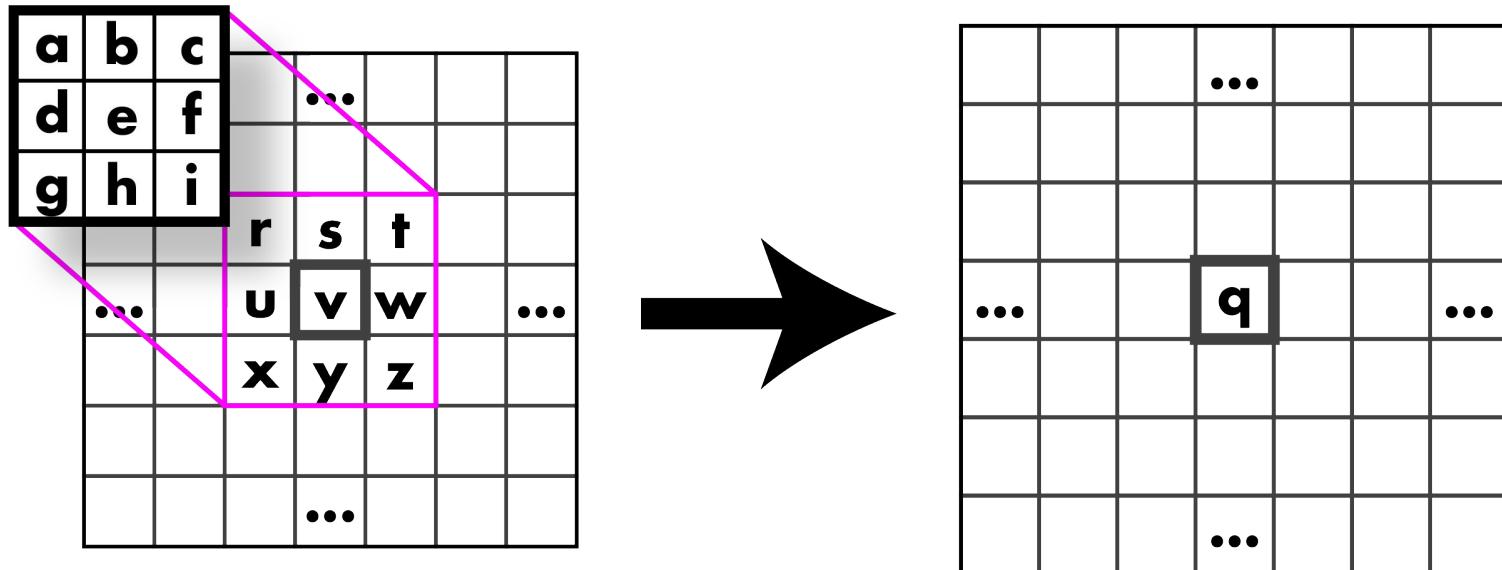


7x7



# Так, что же такое конволюция?

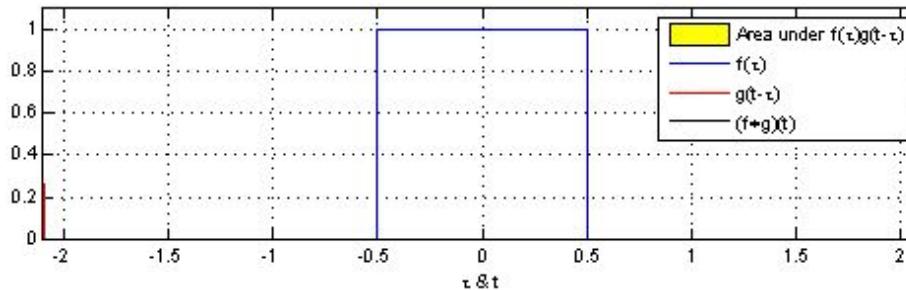
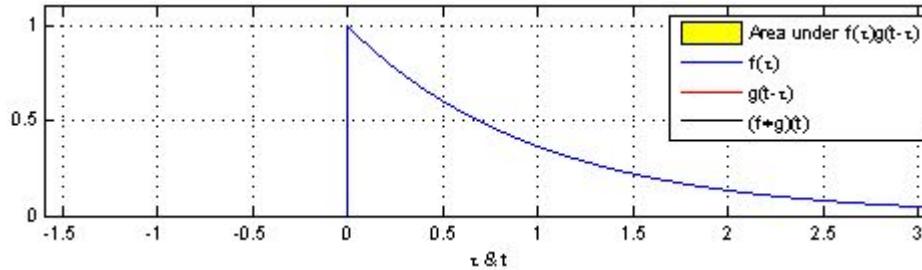
---



# Так, что же такое конволюция?

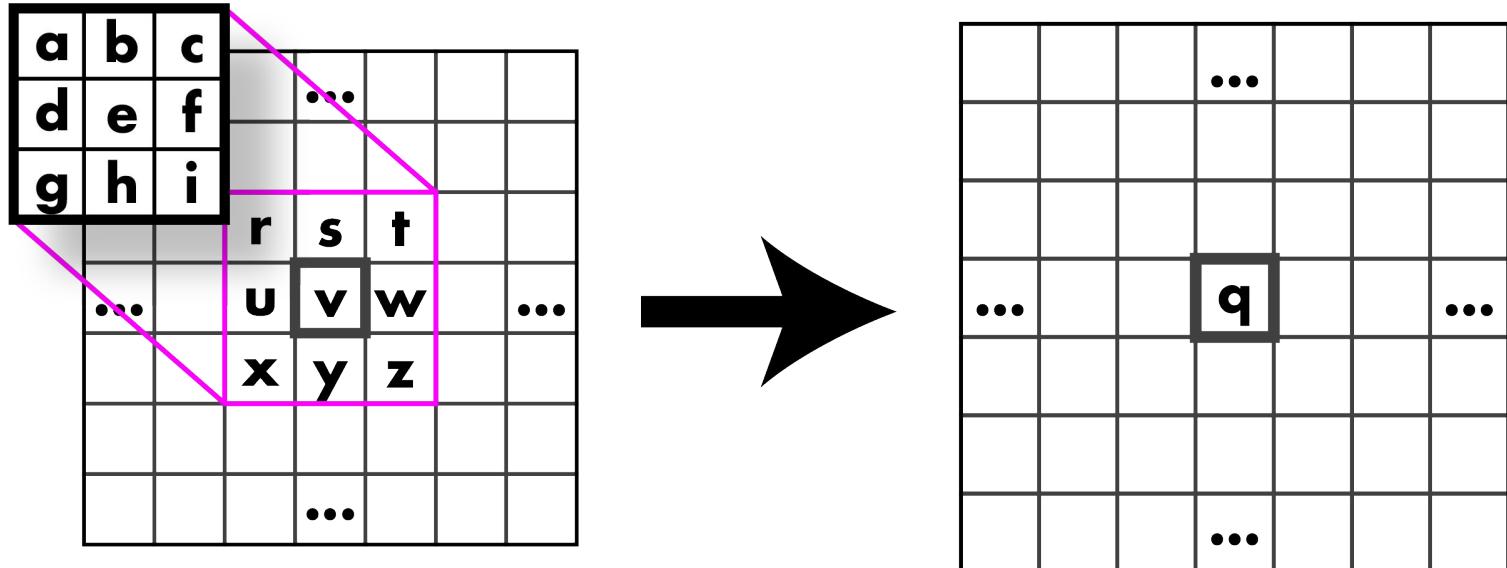
---

$$(f * g)(x) \stackrel{\text{def}}{=} \int_{\mathbb{R}^n} f(y) g(x - y) dy = \int_{\mathbb{R}^n} f(x - y) g(y) dy.$$



# Так, что же такое конволюция?

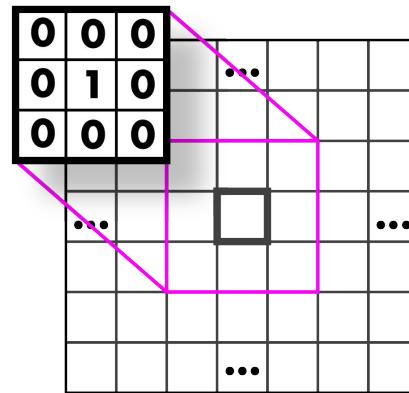
---



$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

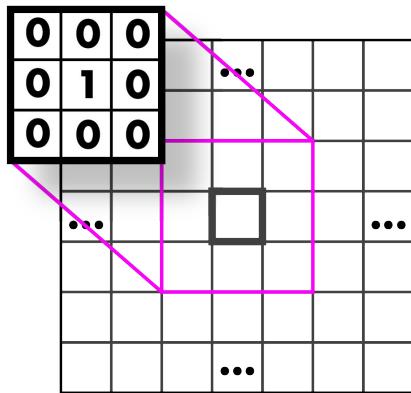
# Что будет делать такой кернел?

---



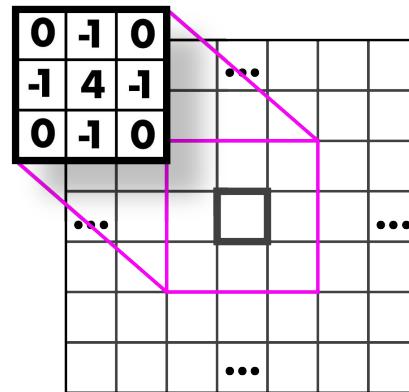
# Identity Kernel: не делает ничего!

---



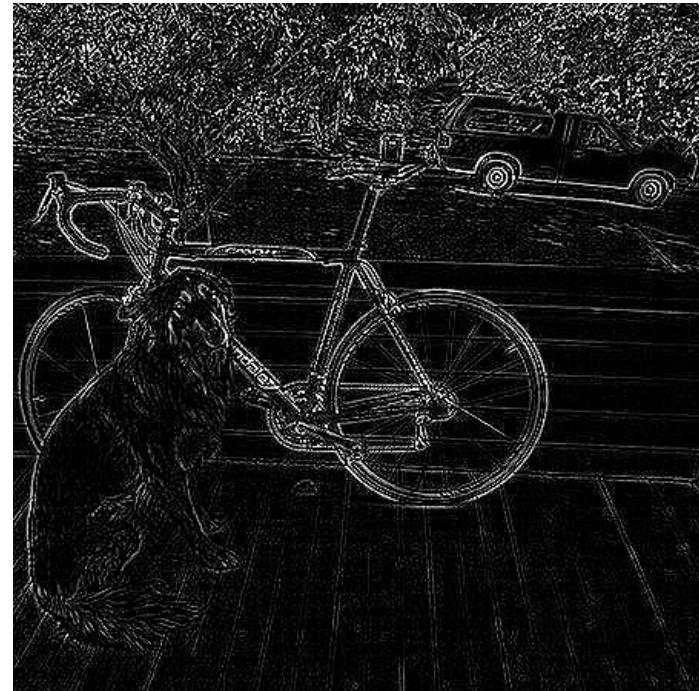
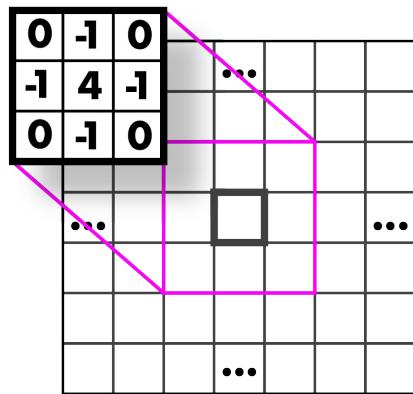
# Что будет делать такой кернел?

---



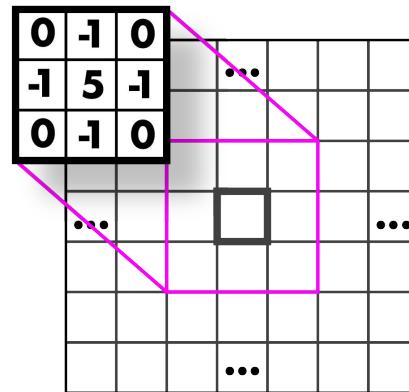
# Highpass Kernel: находит границы

---



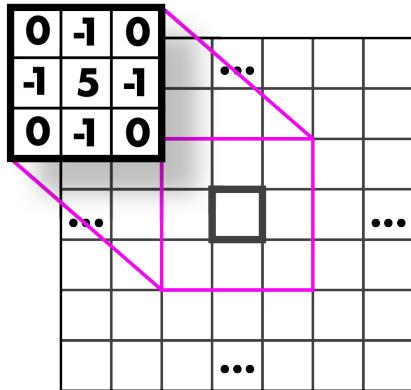
# Что будет делать такой кернел?

---



# Sharpen Kernel: заостряет!

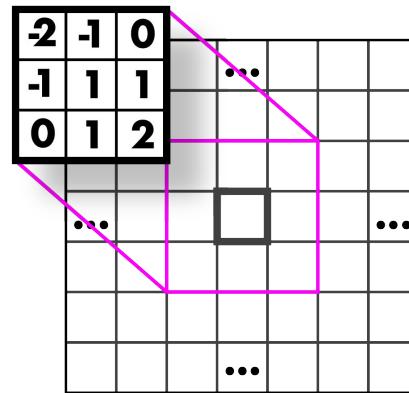
---



Note: sharpen = highpass + identity!

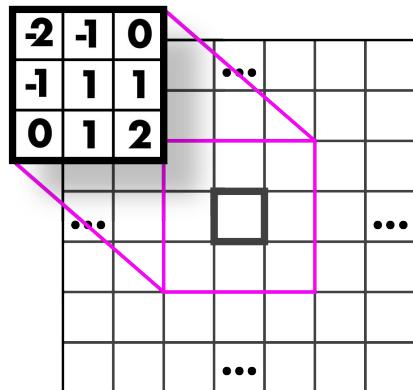
# Что будет делать такой кернел?

---



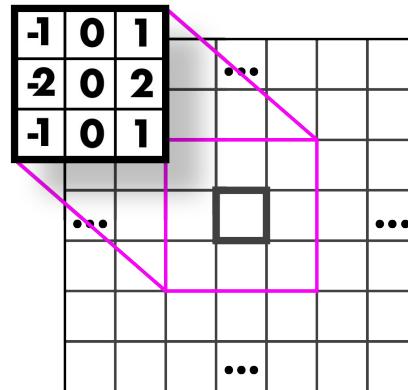
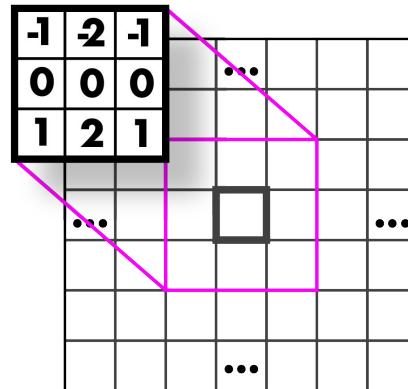
# Emboss Kernel: стилизация

---



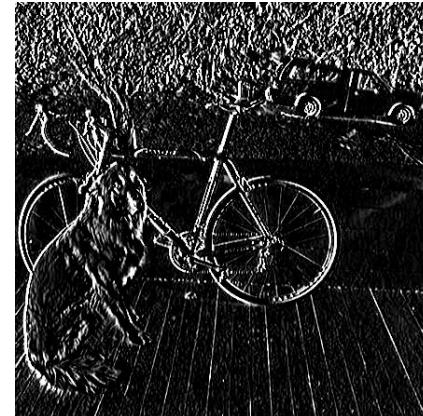
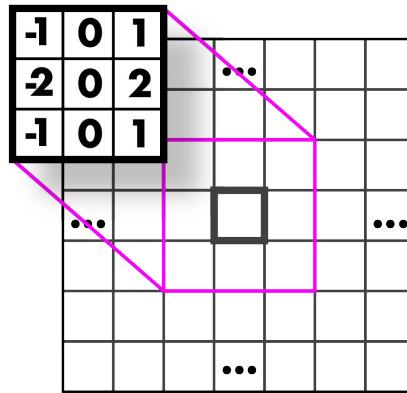
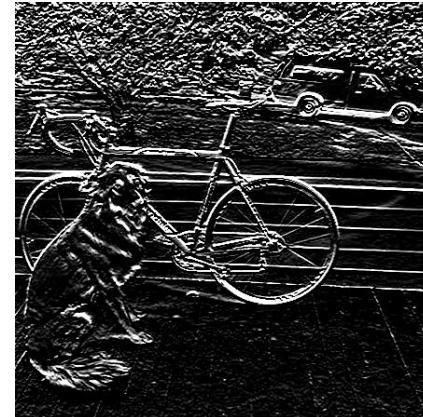
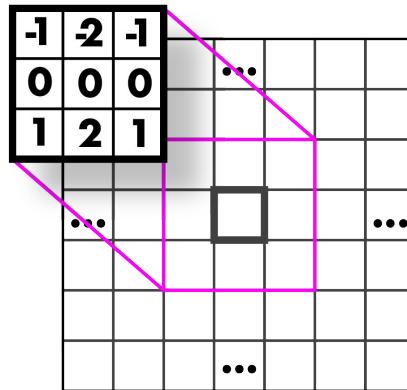
# Что будет делать такой кернел?

---



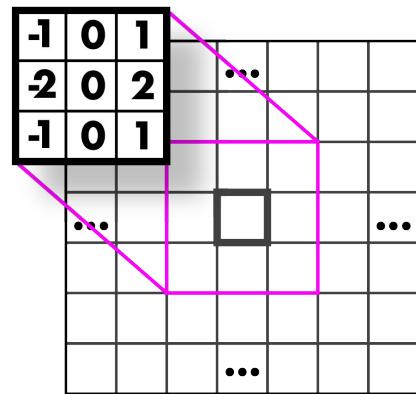
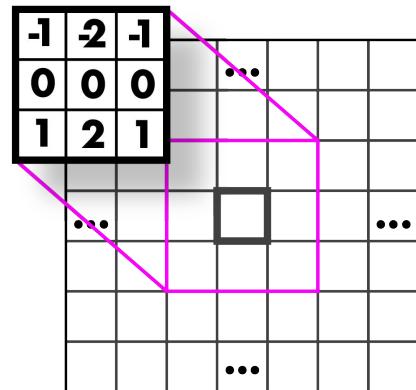
# Sobel Kernels: границы и ...

---



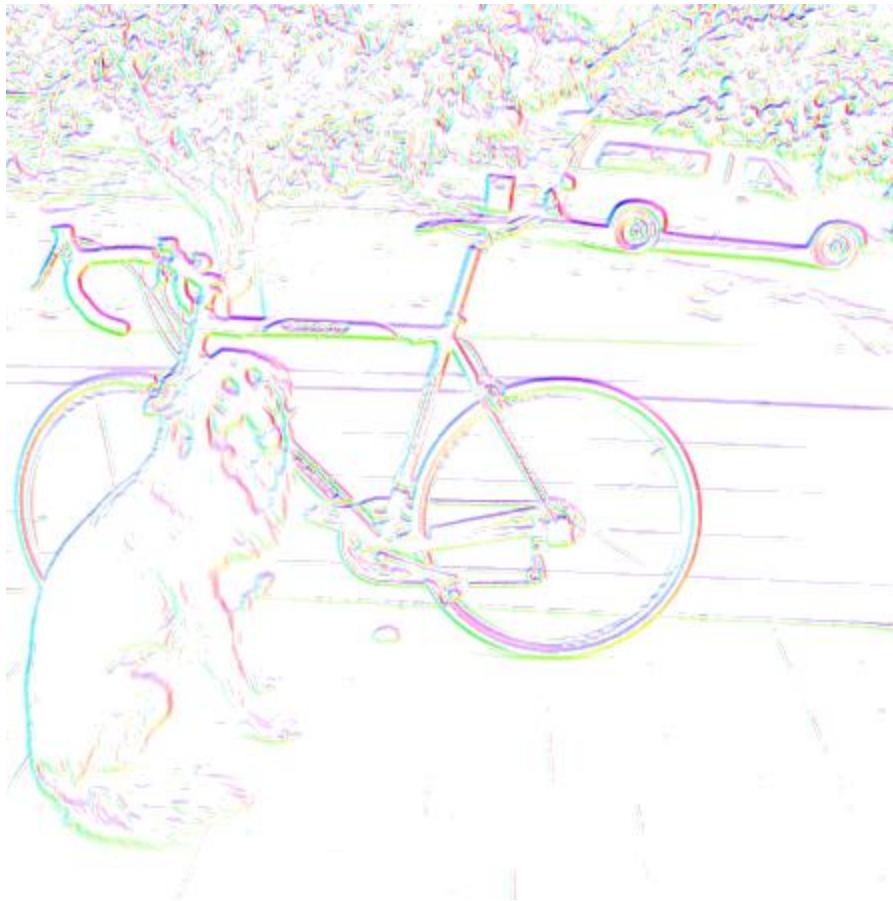
# Sobel Kernels: границы и градиенты!

---



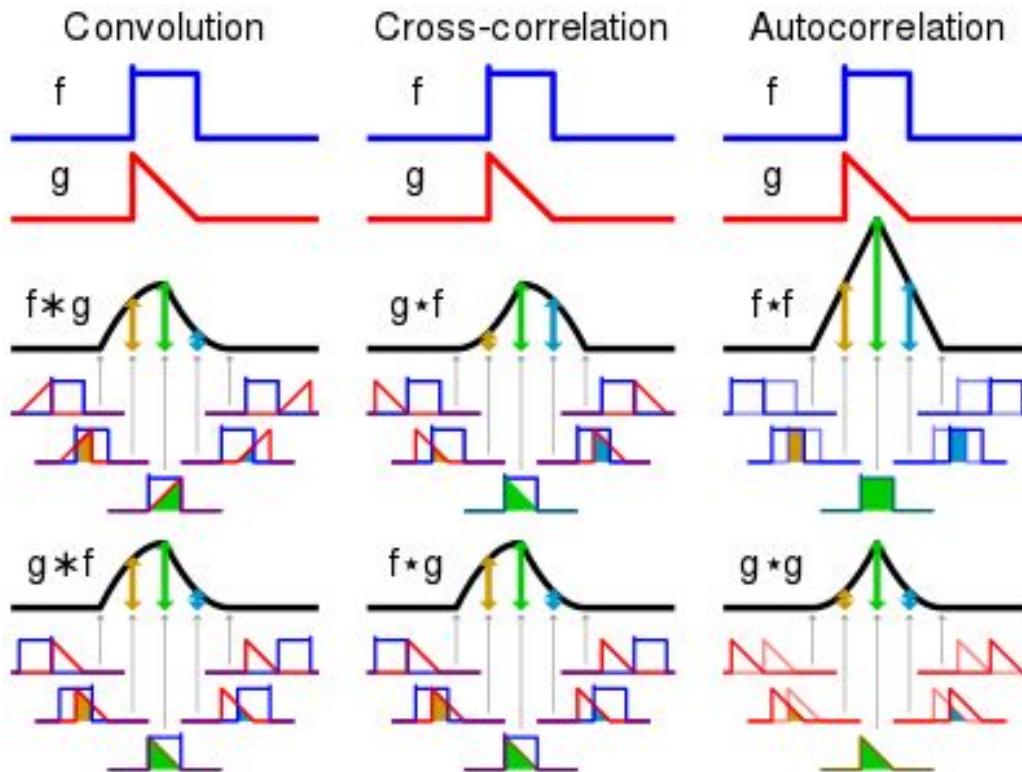
# Sobel Kernels: границы и градиенты!

---

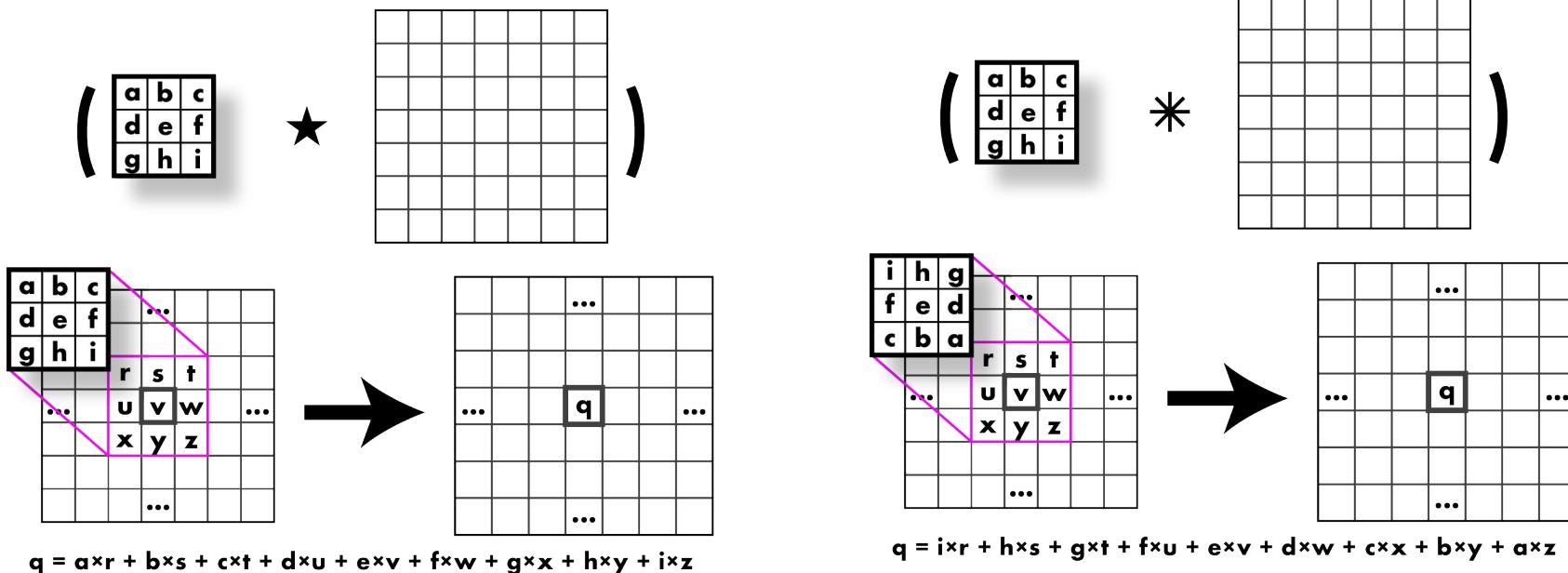


# Кросскореляция VS конволюция

---



# Кросскореляция VS конволюция



# Что можно делать с конволюциями?

---

## Классные математические свойства!

- Коммутативность
  - $A * B = B * A$
- Ассоциативность
  - $A * (B * C) = (A * B) * C$
- Дистрибутивность
  - $A * (B + C) = A * B + A * C$
- Умножение на скаляр
  - $x(A * B) = (xA) * B = A * (xB)$

## Сепарабельные фильтры

---

Можно декомпозировать конволюции!

- 2d гауссиана это просто комбинация двух 1d гауссиан
  - Быстрее выполнять две 1д конволюции

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

## Что можно делать с этим конволюциями?

---

- Blurring
- Sharpening
- Edges
- Features
- Derivatives
- Super-resolution
- Classification
- Detection
- Image captioning
- ...

## Что можно делать с этим конволюциями?

---

- Blurring
- Sharpening
- Edges
- Features
- Derivatives
- Super-resolution
- Classification
- Detection
- Image captioning
- ...

Конволюция - **самая** важная  
операция в компьютерном  
зрении!

## Что можно делать с этим конволюциями?

---

- Blurring
- Sharpening
- Edges
- Features
- Derivatives
- Super-resolution
- Classification
- Detection
- Image captioning
- ...

# 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)

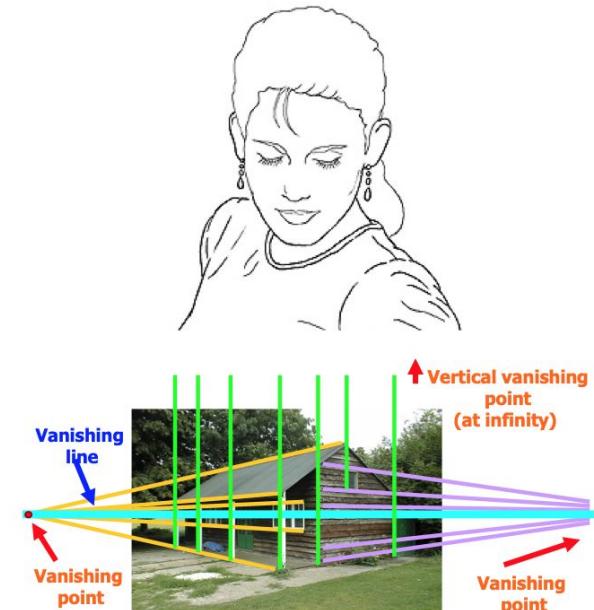


Source: D. Lowe

# Зачем нам нужны границы изображения?

---

- Выделять признаки, распознавать/детектировать/сегментировать объекты
- Геометрические преобразования: находить точку съемки, восстанавливать 3D модель изображения



# Откуда берутся границы объектов

---



Source: D. Hoiem

# Откуда берутся границы объектов

---



Разрыв поверхности

Разрыв глубины

Изменение цвета

Изменения освещения

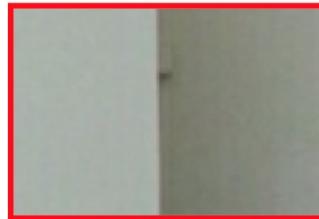
Source: D. Hoiem

# Откуда берутся границы объектов

---



Разрыв глубины



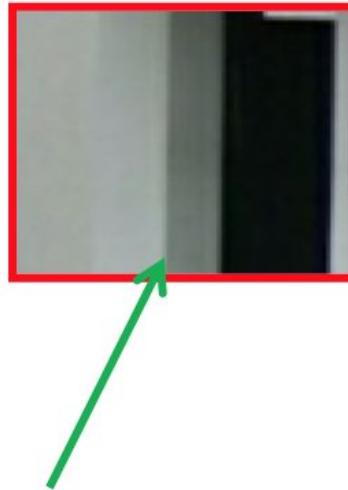
Source: D. Hoiem

# Откуда берутся границы объектов

---



Разрыв поверхности



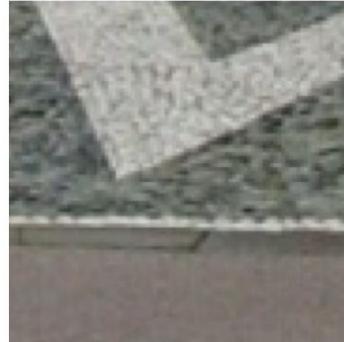
Source: D. Hoiem

# Откуда берутся границы объектов

---



Изменение цвета

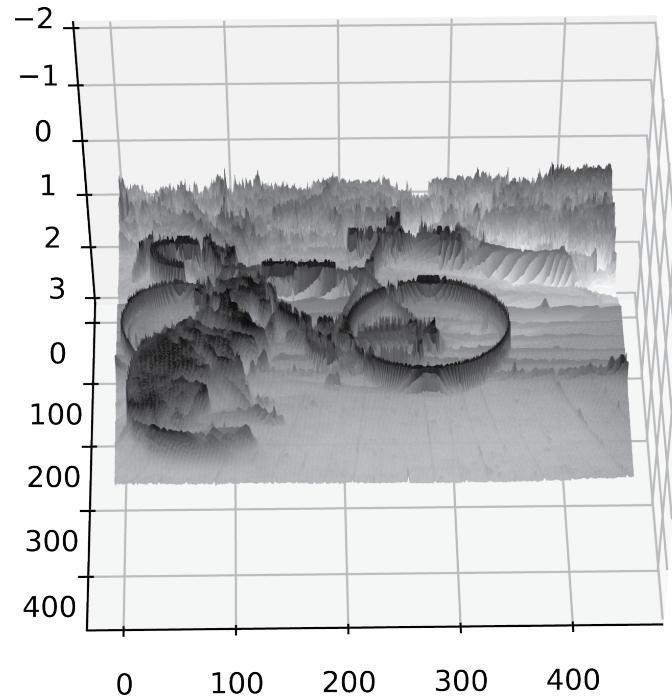
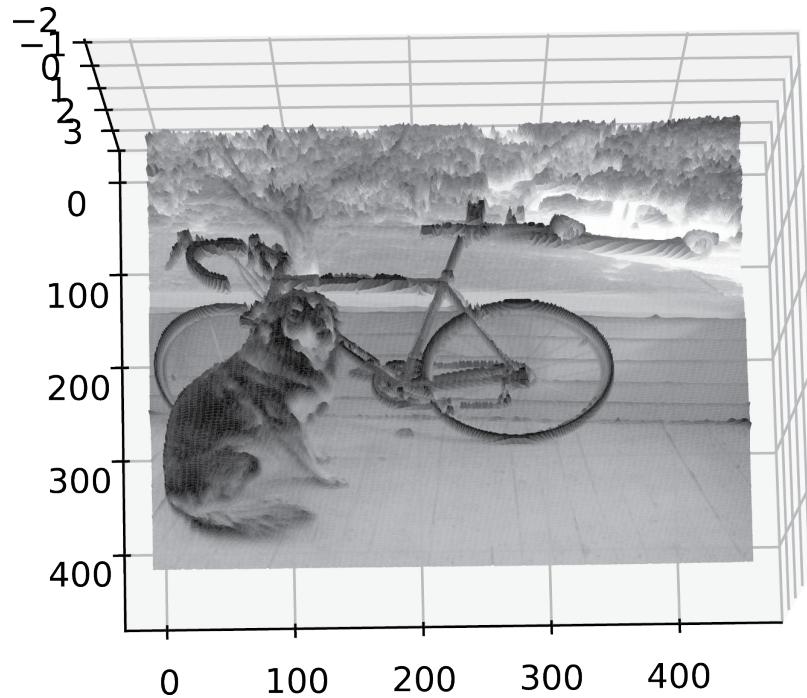


Source: D. Hoiem

# Что такое граница?

---

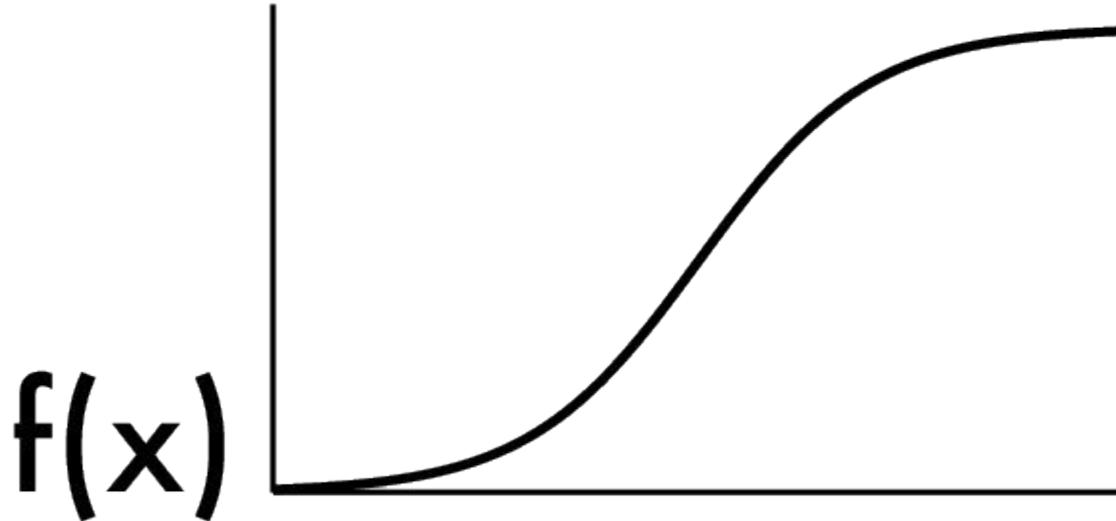
- Изображение - это функция
- Границы - резкие изменения значения этой функции



# Что такое граница?

---

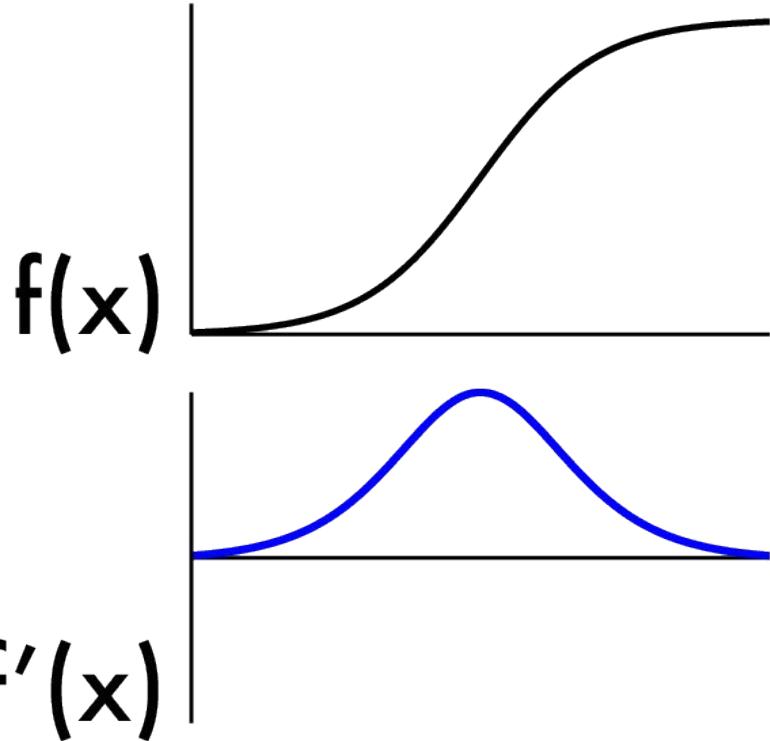
- Изображение - это функция
- Границы - резкие изменения значения этой функции



# Нахождение краев

---

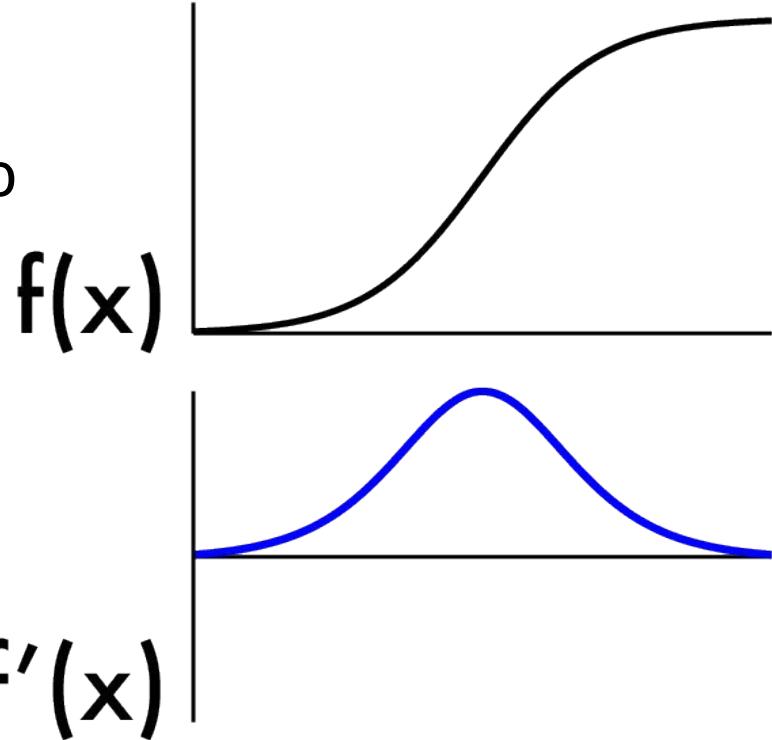
- Можем взять производную
- Граница = большое значение производной



# Производные изображения

---

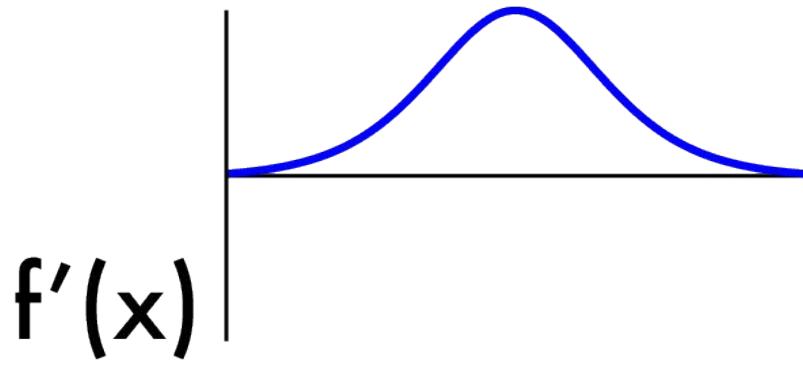
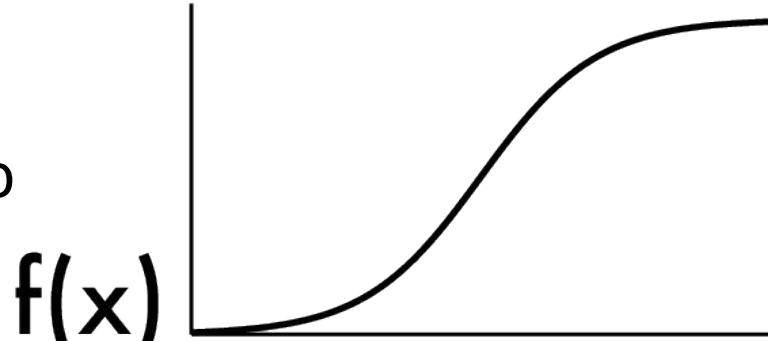
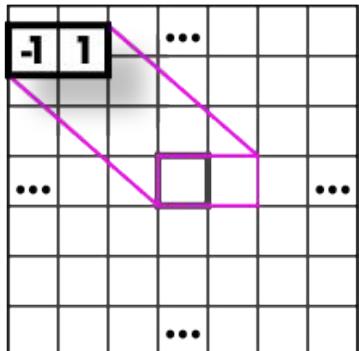
- Вспомним, что:
  - $f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$ .
- У нас нет настоящей функции
- Нужно оценивать производную
- Скажем:  $h = 1$
- Как это будет выглядеть?



# Производные изображения

---

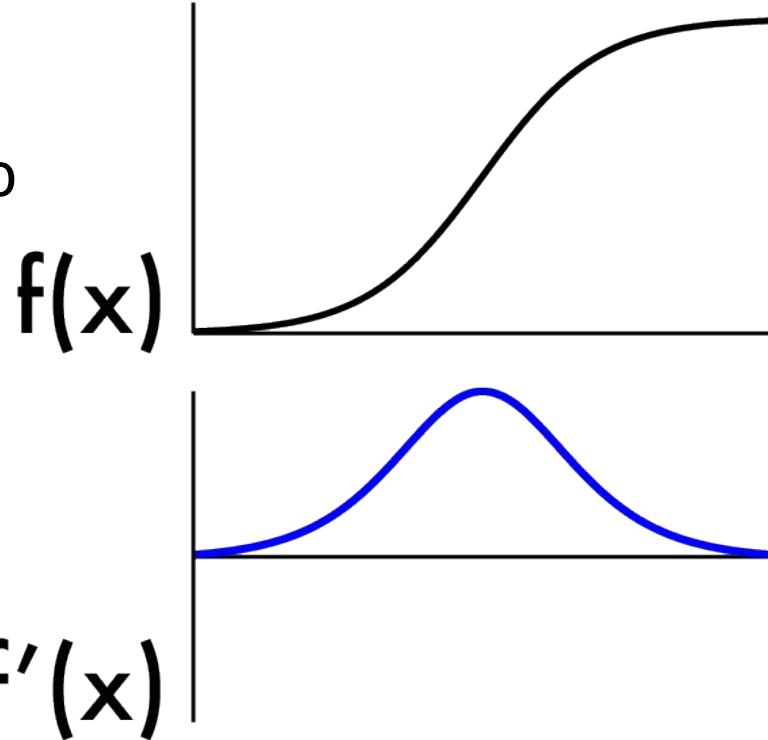
- Вспомним, что:
  - $f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$ .
- У нас нет настоящей функции
- Нужно оценивать производную
- Скажем:  $h = 1$
- Как это будет выглядеть?



# Производные изображения

---

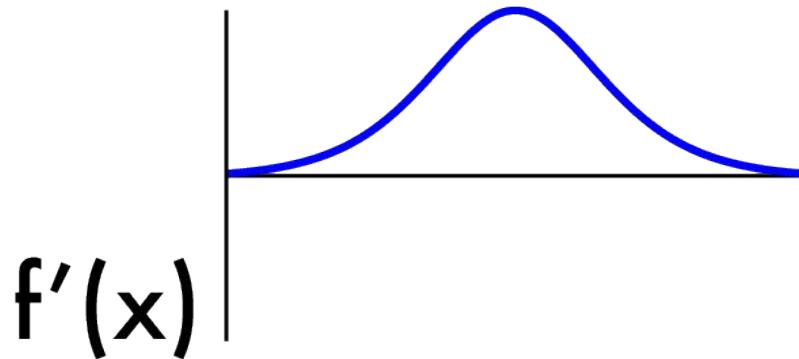
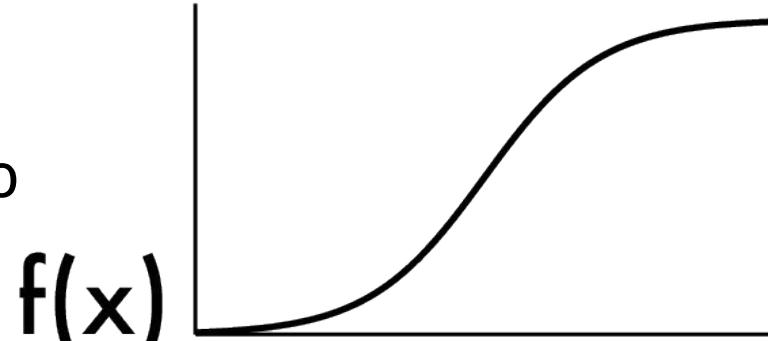
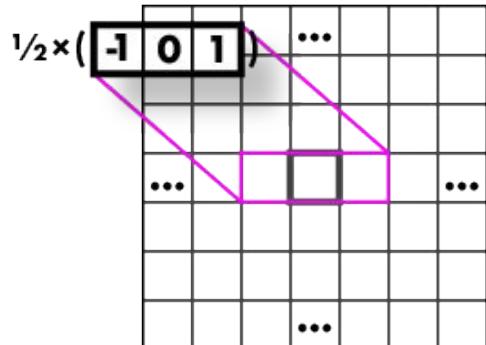
- Вспомним, что:
  - $f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$ .
- У нас нет настоящей функции
- Нужно оценивать производную
- Скажем:  $h = 2$
- Как это будет выглядеть?



# Производные изображения

---

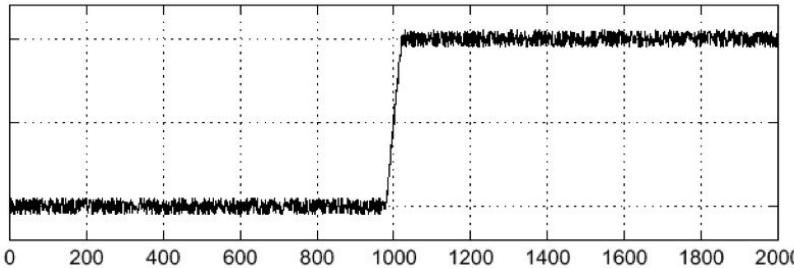
- Вспомним, что:
  - $f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$ .
- У нас нет настоящей функции
- Нужно оценивать производную
- Скажем:  $h = 2$
- Как это будет выглядеть?



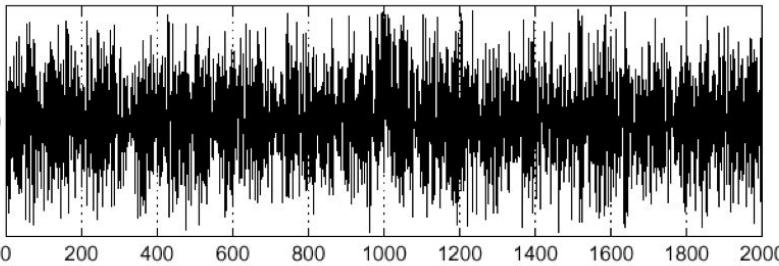
# Границы в зашумленных функциях

---

$f(x)$

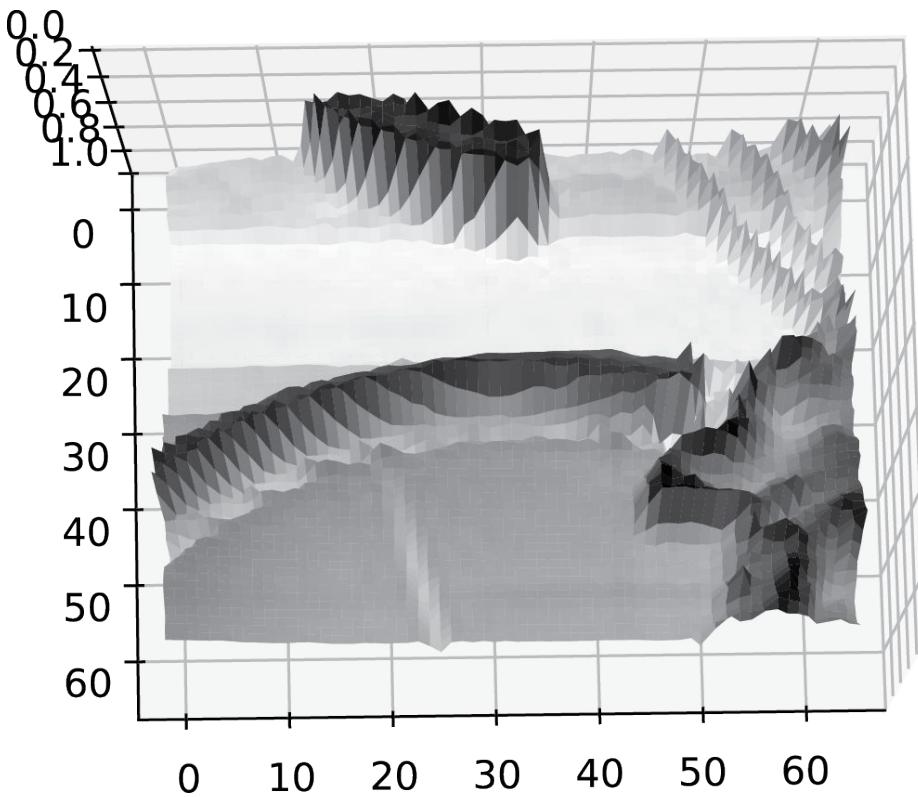
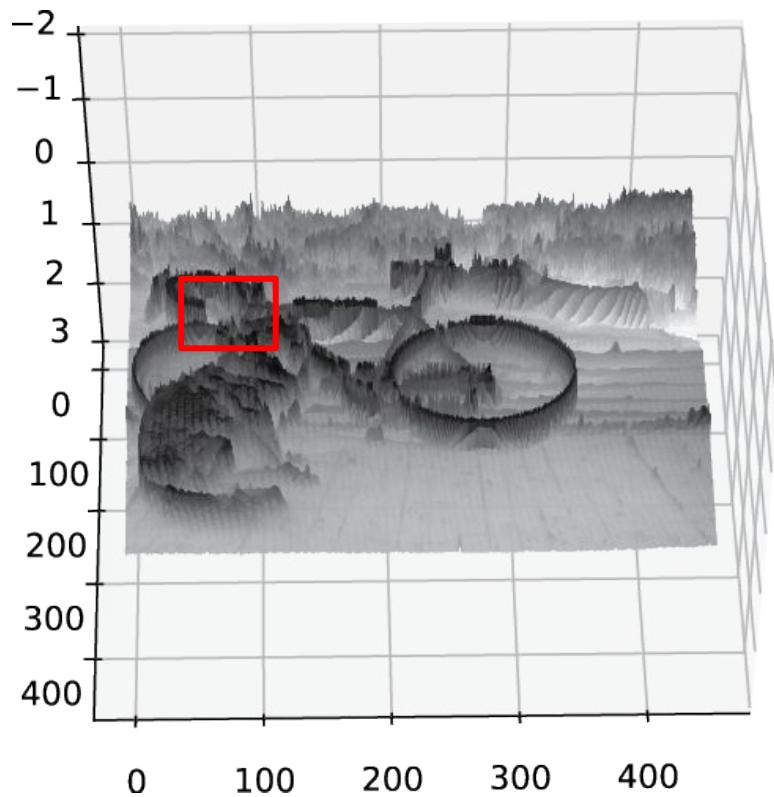


$\frac{d}{dx}f(x)_0$



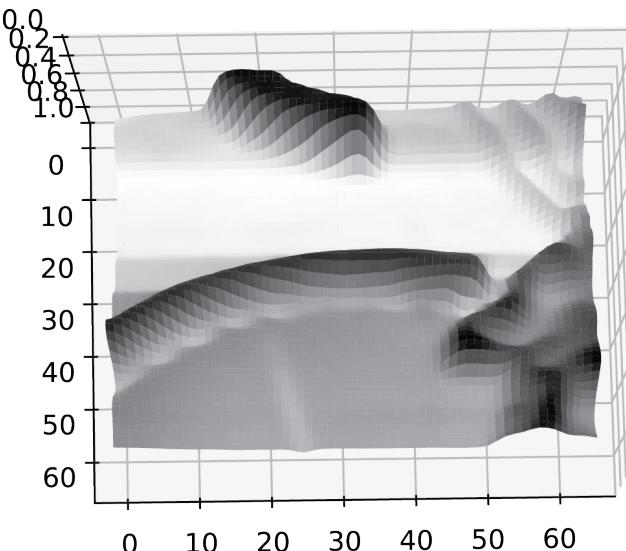
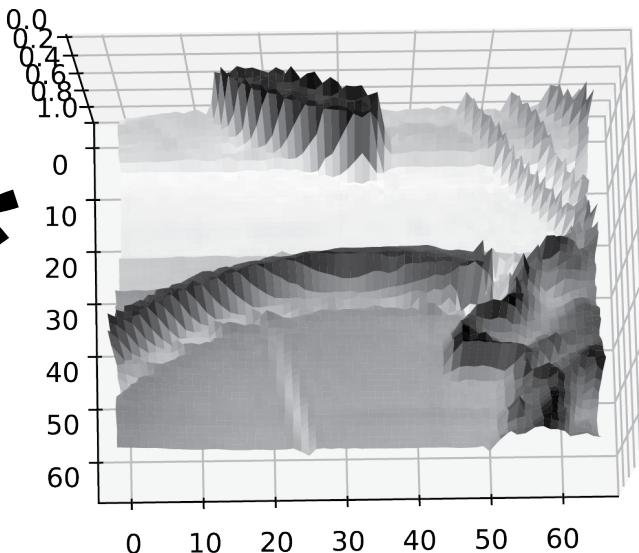
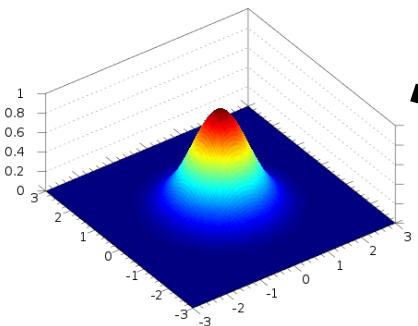
# Изображения шумные!

---



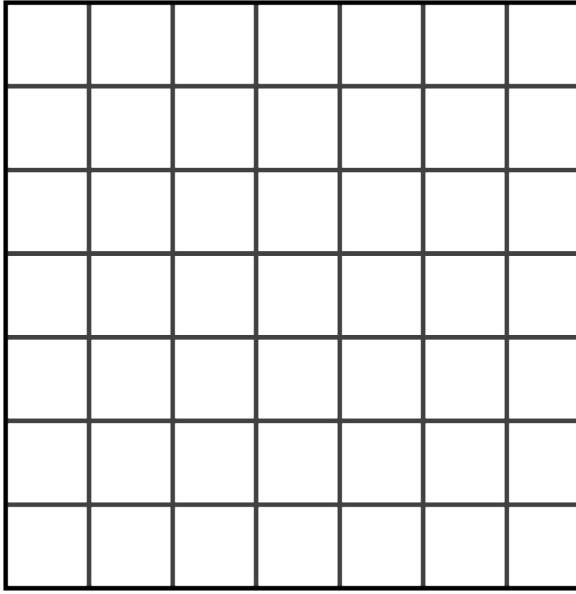
# Но мы уже знаем, как сглаживать!

---



Сглаживаем, потом берем производную

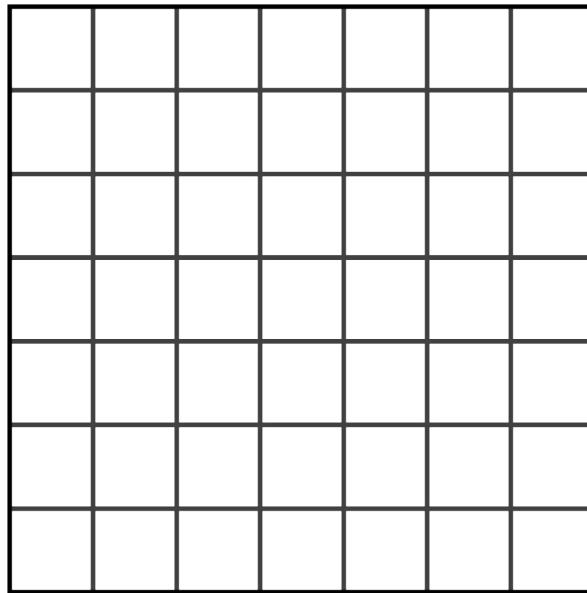
---

$$\frac{1}{2} \times (-1 \ 0 \ 1) * \left( \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \right) *$$

$$)$$

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times (-1 \ 0 \ 1) * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} *$$



Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

A diagram illustrating a convolution operation. At the top, a horizontal vector  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$  is multiplied by a vertical column of three  $3 \times 3$  matrices. The result is a  $2 \times 2$  output matrix. Below this, a pink arrow points from the  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$  vector to the top-left cell of the  $3 \times 3$  input matrix. The input matrix has values  $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ . The output matrix has a value of 2 in its top-left cell and zeros elsewhere.

$$\frac{1}{2} \times \begin{bmatrix} 2 & & \\ & & \\ & & \end{bmatrix}$$

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

A diagram illustrating a convolution operation. At the top, a horizontal vector  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$  is shown next to a 3x3 input matrix:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

An arrow points from the vector to the input matrix. Below the input matrix, a large black arrow points to the right. To the right of the arrow, a 2x0 matrix is shown:

$$\frac{1}{2} \times \begin{bmatrix} 2 & 0 \\ \vdots & \vdots \end{bmatrix}$$

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

The diagram illustrates a convolution operation. A 3x3 input matrix is shown below, with the central element (2) highlighted by a magenta box. Above it, a 1x3 kernel is shown, also with its central element (0) highlighted by a magenta box. An arrow points from the kernel to the input matrix, indicating the receptive field of the kernel's center. To the right, the result of the convolution is shown as a 1x3 output vector:  $\frac{1}{2} \times [2 \ 0 \ -2]$ .

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

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times (-1 \ 0 \ 1) * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The diagram illustrates a convolution operation. At the top, a horizontal vector  $(-1 \ 0 \ 1)$  is multiplied by a vertical  $3 \times 3$  input matrix. The result is then scaled by  $\frac{1}{2}$ . Below this, a large black arrow points to the right, indicating the flow of the computation. To the left of the arrow, the input matrix is shown again, with the first row  $(-1 \ 0 \ 1)$  highlighted in pink. This row is being multiplied by the kernel. The result of this multiplication is  $2$ , which is then scaled by  $\frac{1}{2}$  to produce the final output value  $1$ .

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

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times (-1 \ 0 \ 1) * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Diagram illustrating the convolution operation:

Input Matrix (3x3):

-1	0	1	1
2	4	2	
1	2	1	

Kernel (1x3):

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

Output Matrix (2x2):

2	0	-2	
4	0		

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

The diagram illustrates a convolution step. A 1x3 kernel  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$  is applied to a 3x3 input matrix  $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ . The result of this convolution is then multiplied by  $\frac{1}{2}$  to produce the final output.

The input matrix is shown with a pink outline around the central element 4, indicating the receptive field of that output unit. The output matrix is  $\frac{1}{2} \times \begin{bmatrix} 2 & 0 & -2 \\ 4 & 0 & -4 \\ 0 & 0 & 0 \end{bmatrix}$ .

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

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

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

Сглаживаем, потом берем производную

---

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

A diagram illustrating a convolution operation. A 3x3 input matrix (bottom) is multiplied by a 1x3 kernel (left). The result is a 2x3 output matrix (right). The input matrix has values 1, 2, 1 in the first row, 2, 4, 2 in the second row, and 1, 2, 1 in the third row. The kernel has values -1, 0, 1. The output matrix has values 2, 0, -2 in the first row, 4, 0, -4 in the second row, and 2, 0, 0 in the third row. A pink arrow points from the input matrix to the output matrix.

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

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

Сглаживаем, потом берем производную

---

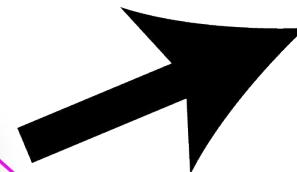
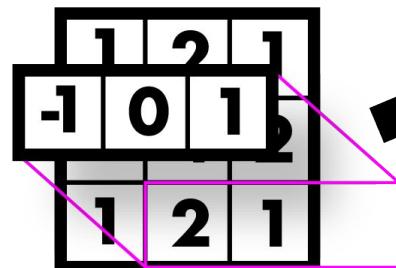
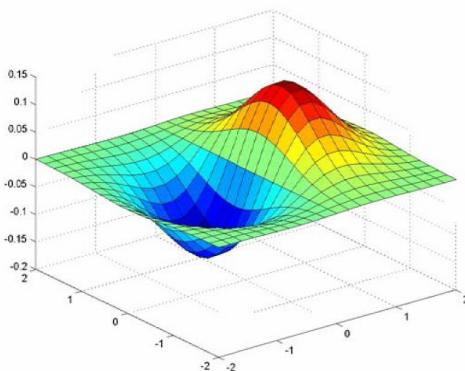
$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

The diagram illustrates a convolution operation. On the left, a 3x3 input matrix (gray background) is multiplied by a 1x3 kernel (black border). The result is a 3x1 output vector (black border). A large black arrow points from the input to the output. A pink line highlights the bottom-right element of the input matrix, which is highlighted with a pink box.

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

# Фильтр Собеля: сглаживание и производная

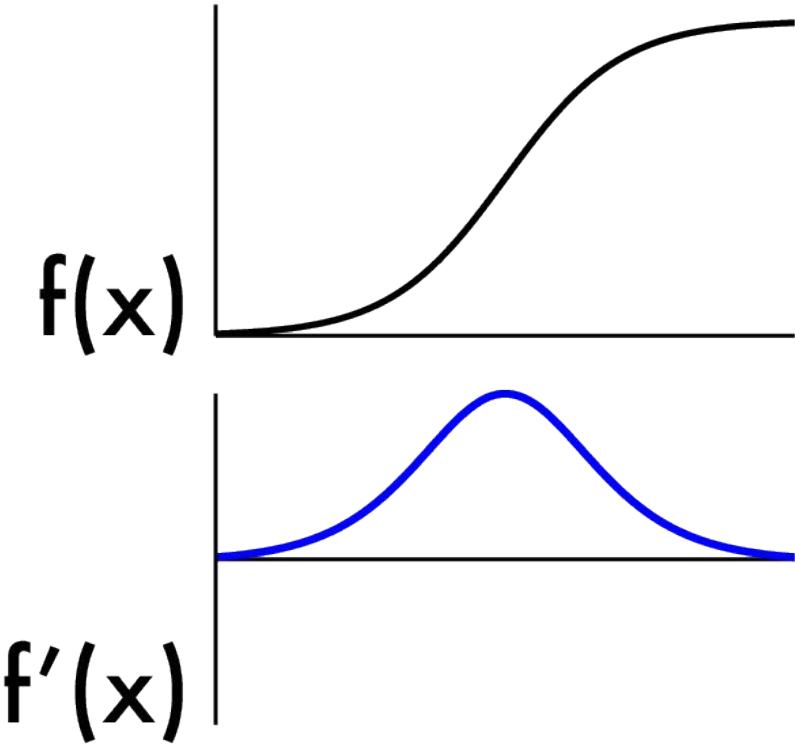
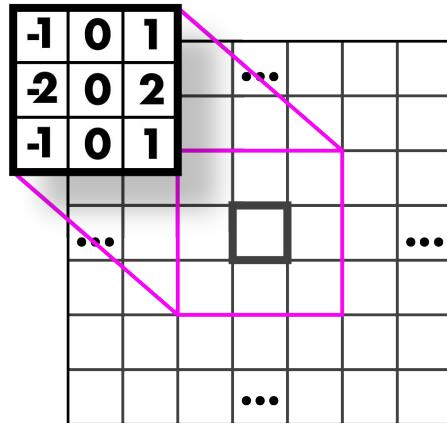
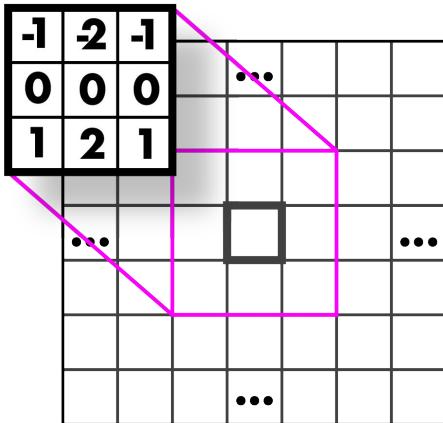
$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$



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

# Нахождение границы

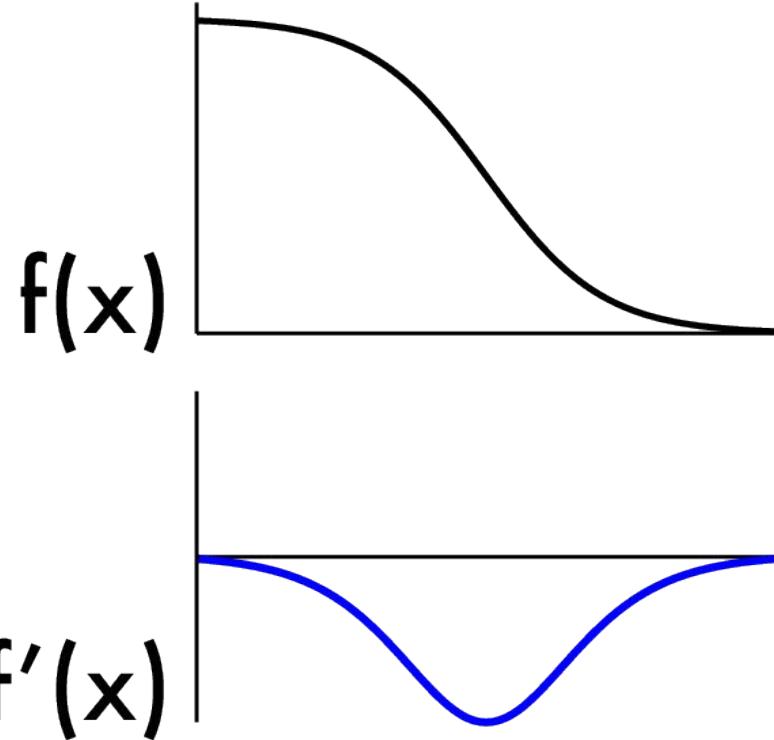
- Можно брать производные
- Находить большие значения
- Фильтр Собеля
- Но...



# Нахождение границ

---

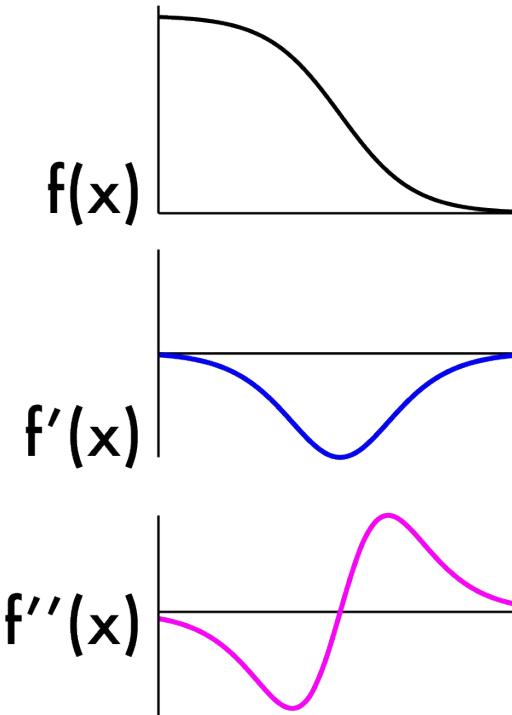
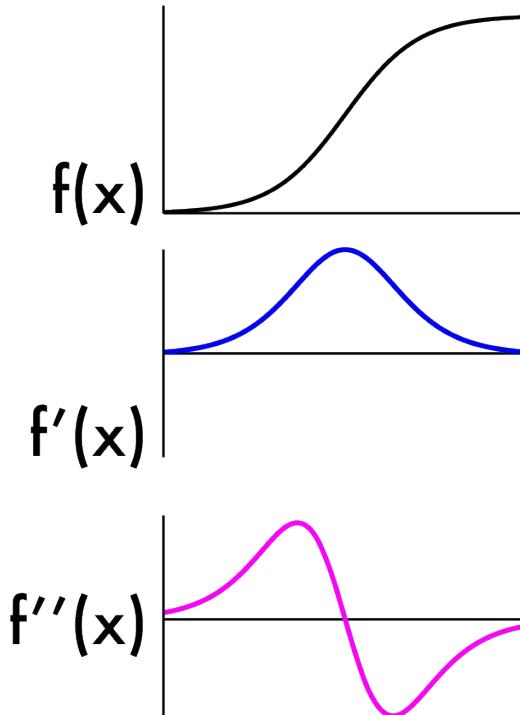
- Можно брать производные
- Находить большие значения
- Фильтр Собеля
- Но...
- Функция может возрастать или убывать
- Хотим находить экстремумы



## Вторая производная

---

- Пересекает ноль в точке экстремума



# Лапласиан (вторая производная)!

- Пересекает ноль в точке экстремума

- Вспомним:

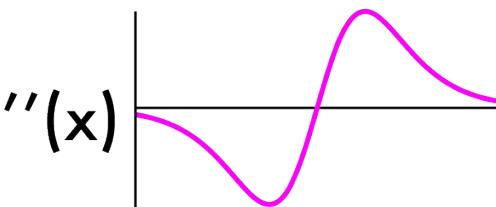
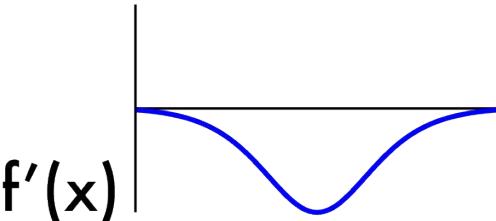
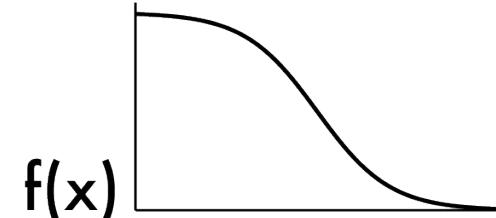
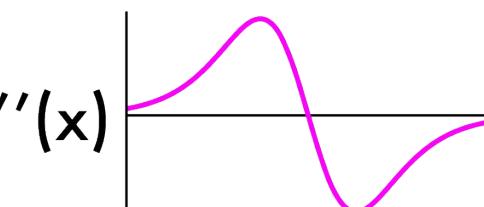
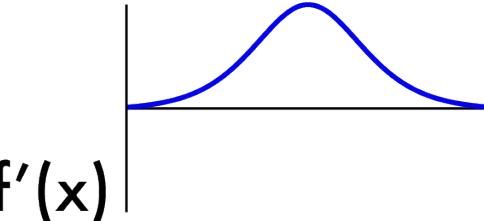
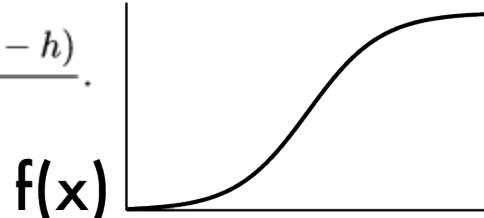
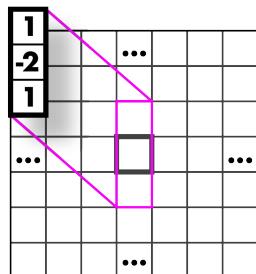
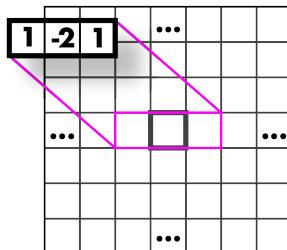
$$- f''(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

- Лапласиан:

$$- \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- И снова,

оцениваем  $f''(x)$ :



# Лапласиан (вторая производная)!

---

- Лапласиан:

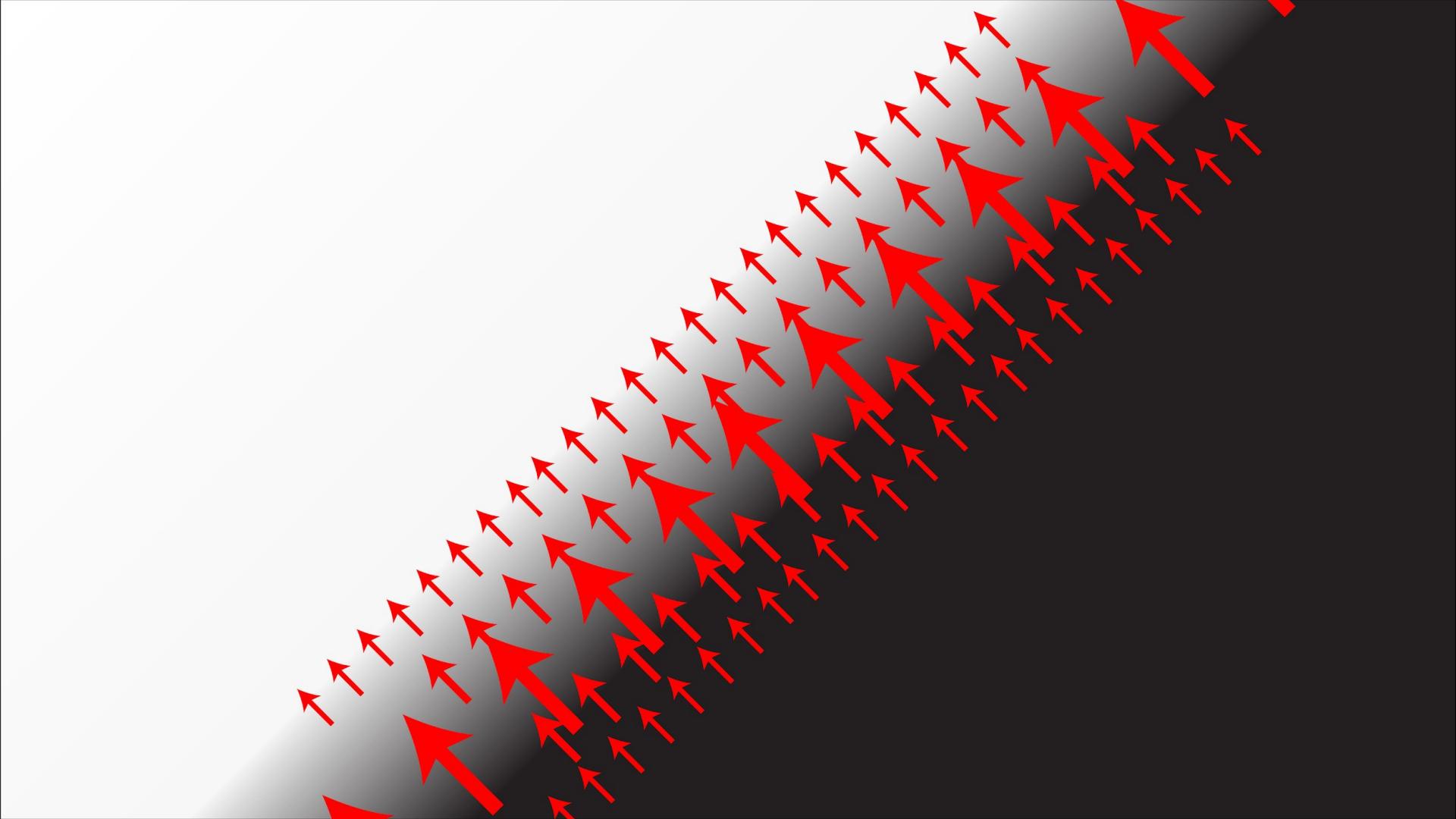
- $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

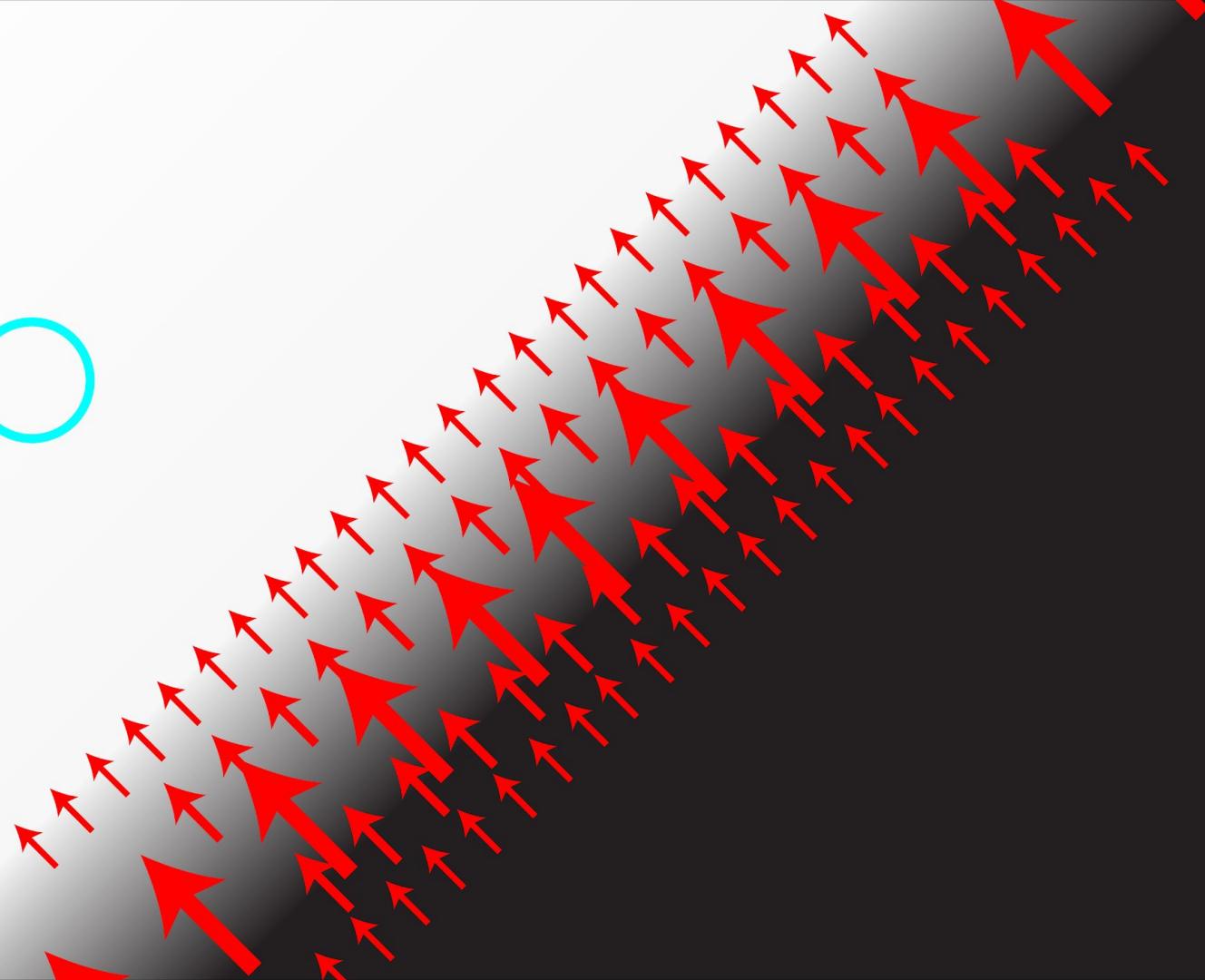
# Лапласиан (вторая производная)!

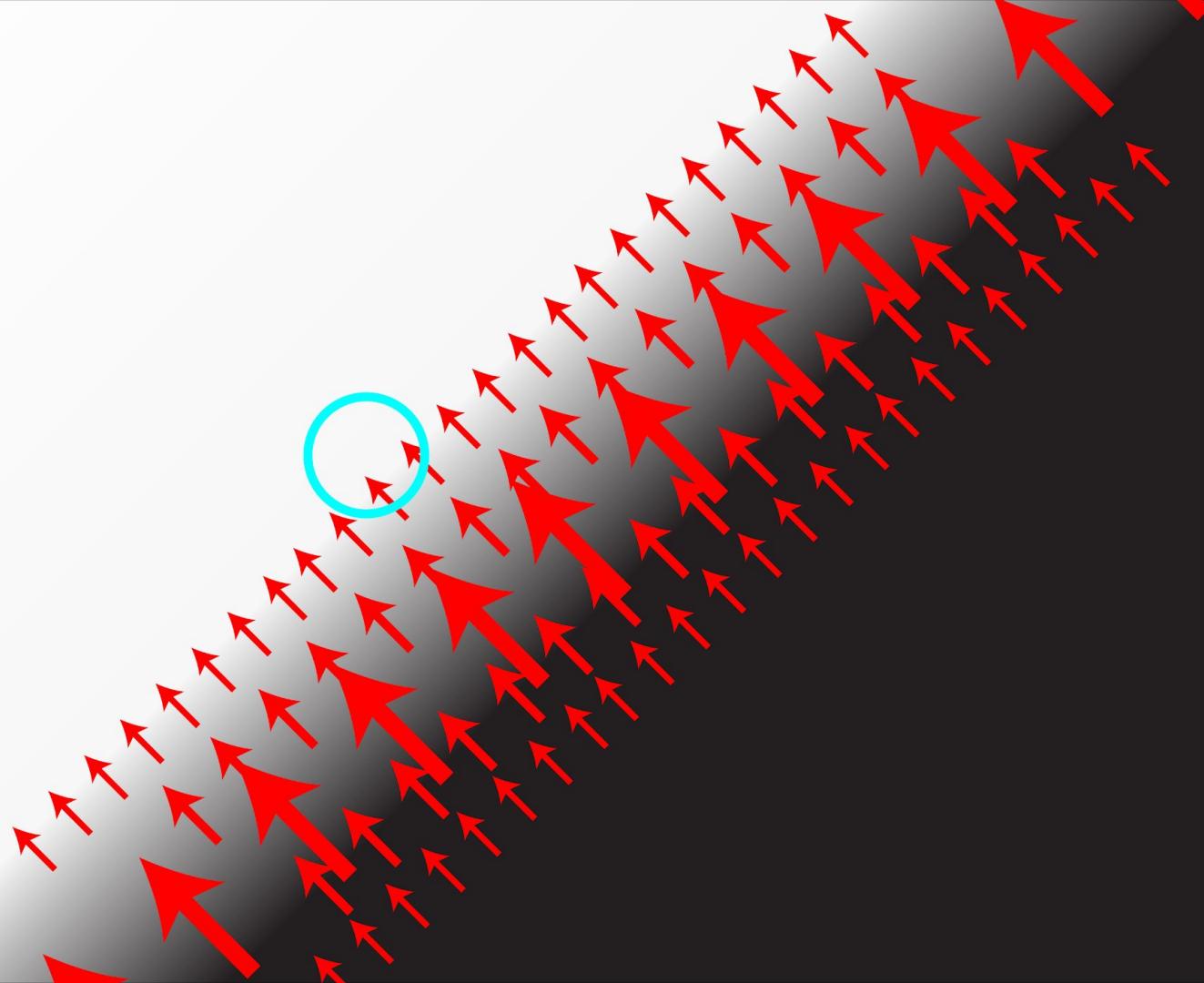
---

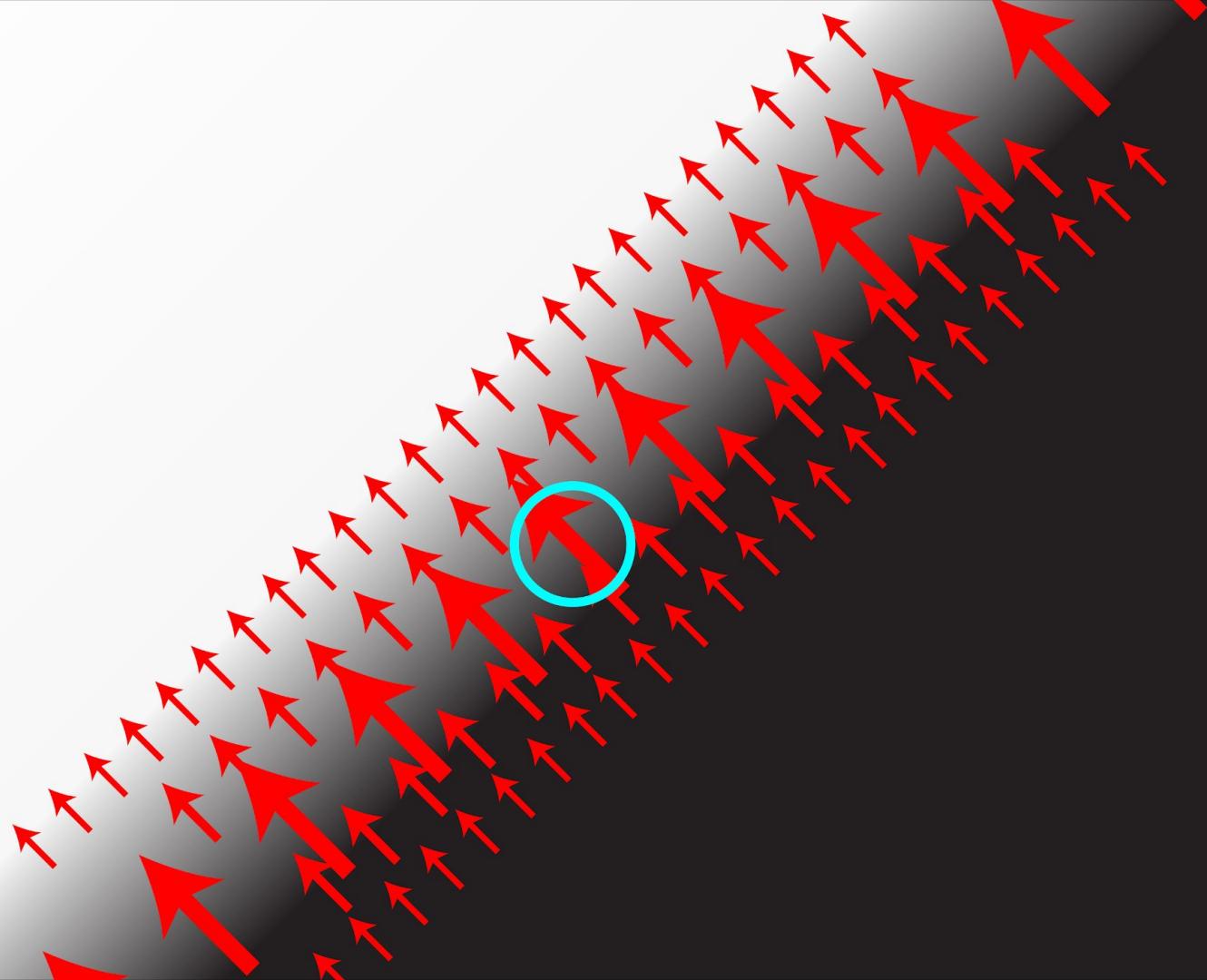
- Лапласиан:
  - $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Измеряет дивергенцию
  - Поток вектора-градиента через маленькую область

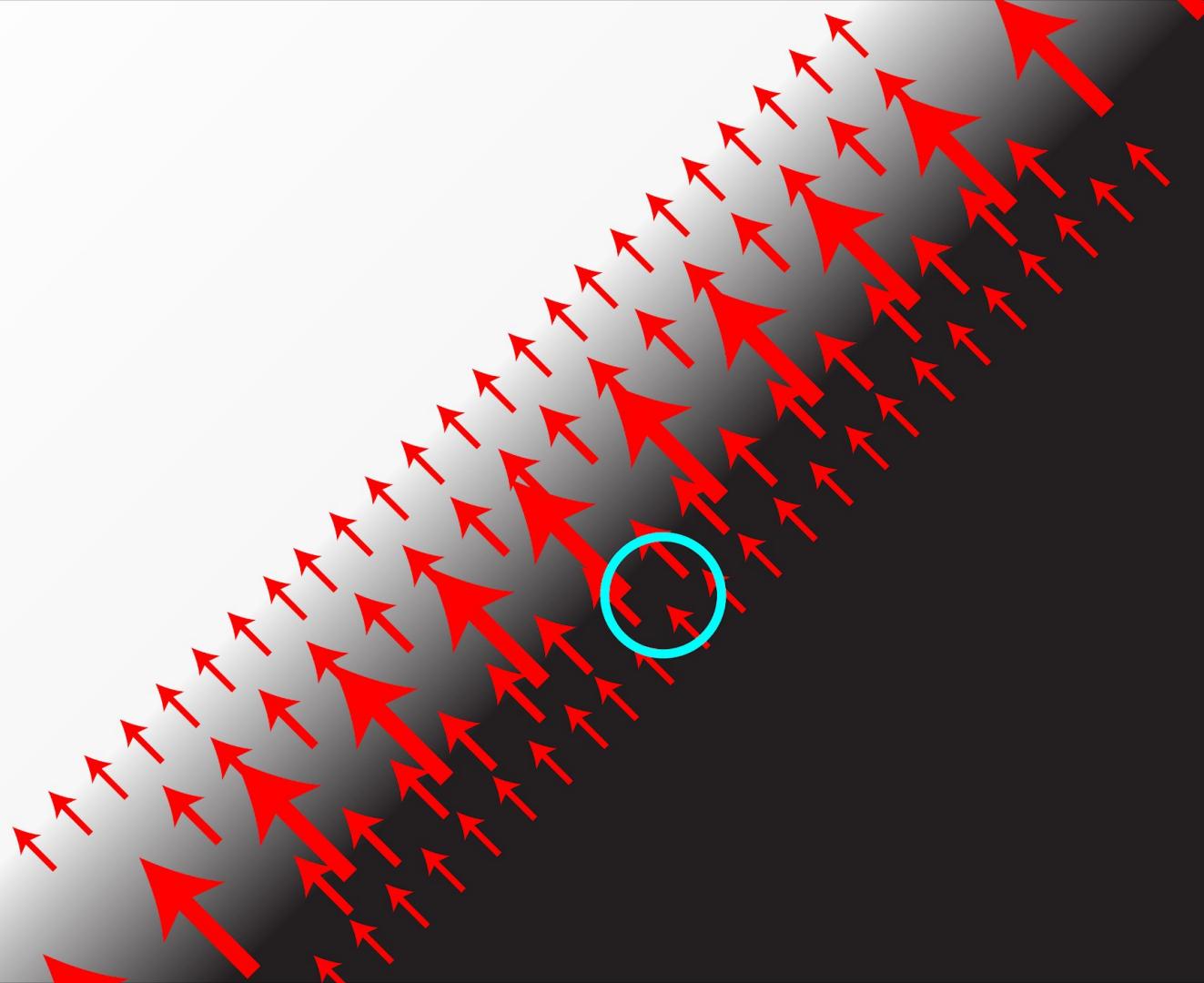








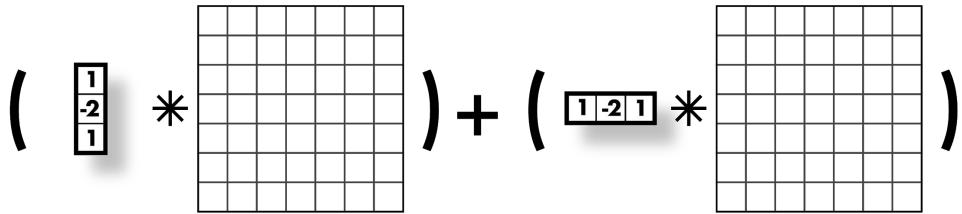




# Лапласиан (вторая производная)!

- ### - Лапласиан:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

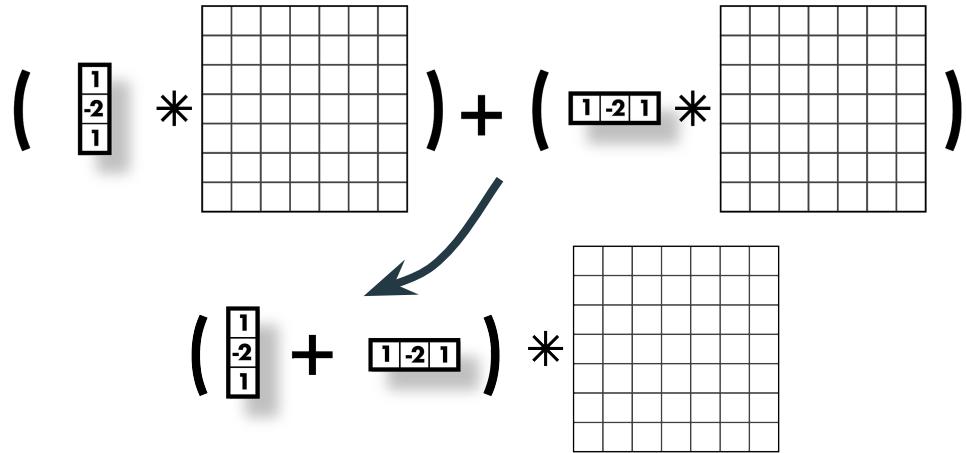


# Лапласиан (вторая производная)!

---

- Лапласиан:

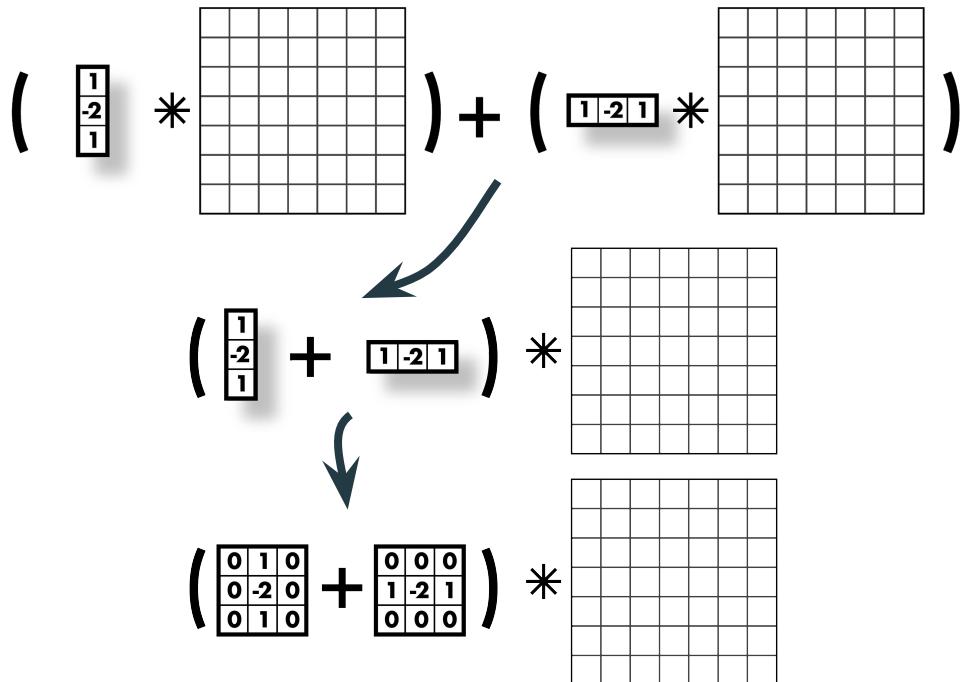
- $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$



# Лапласиан (вторая производная)!

- Лапласиан:

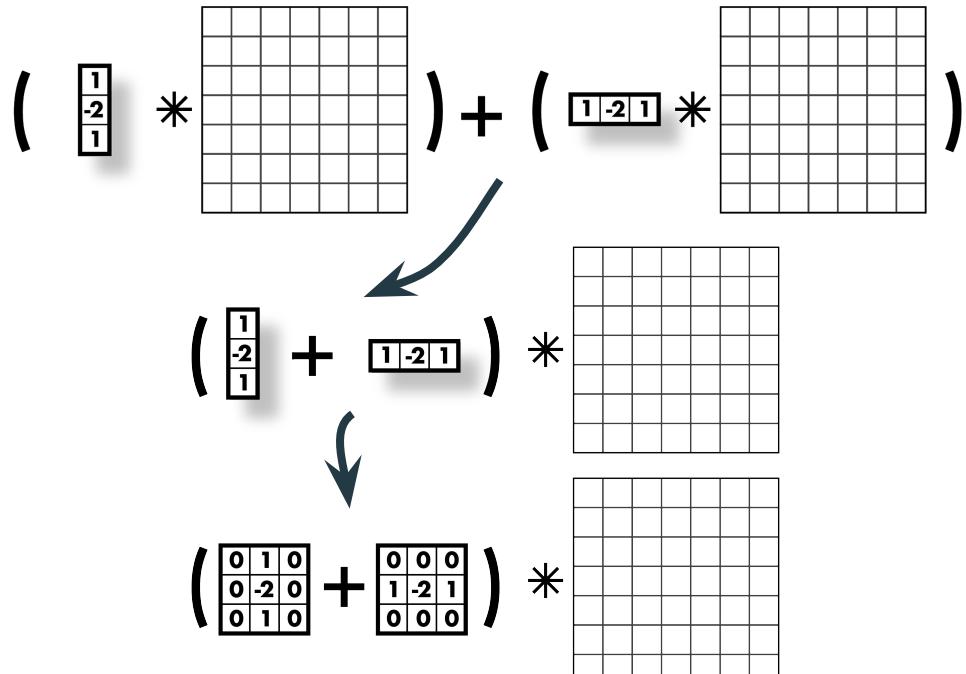
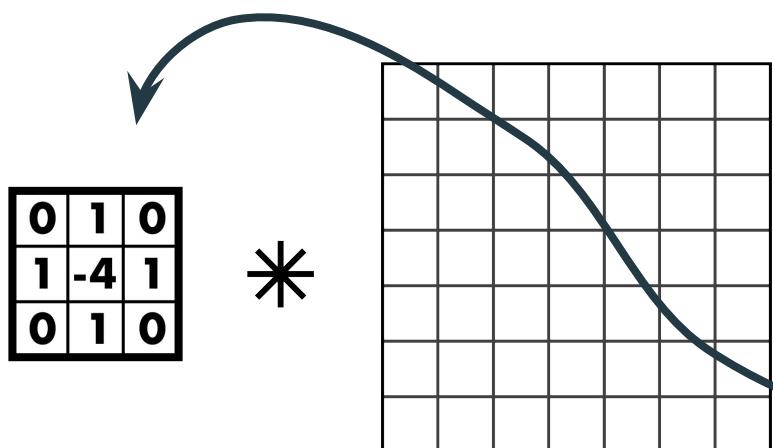
$$- \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$



# Лапласиан (вторая производная)!

- Лапласиан:

- $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

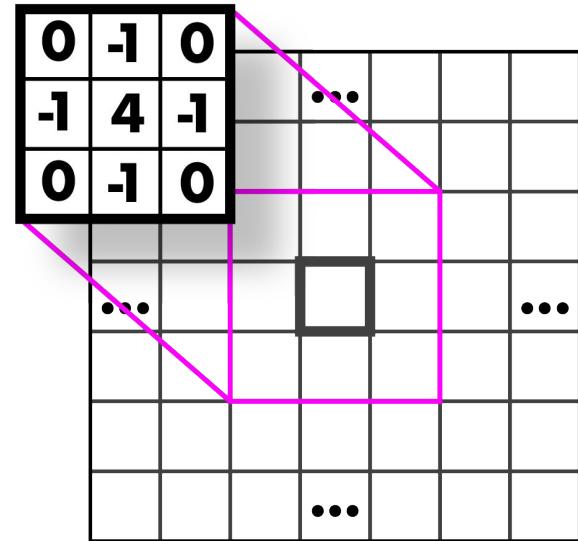
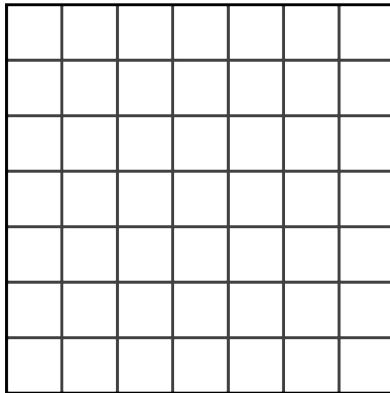


# Лапласиан (вторая производная)!

- Лапласиан:
  - $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Отрицательный Лапласиан, -4 в середине
- Положительный Лапласиан --->

0	1	0
1	-4	1
0	1	0

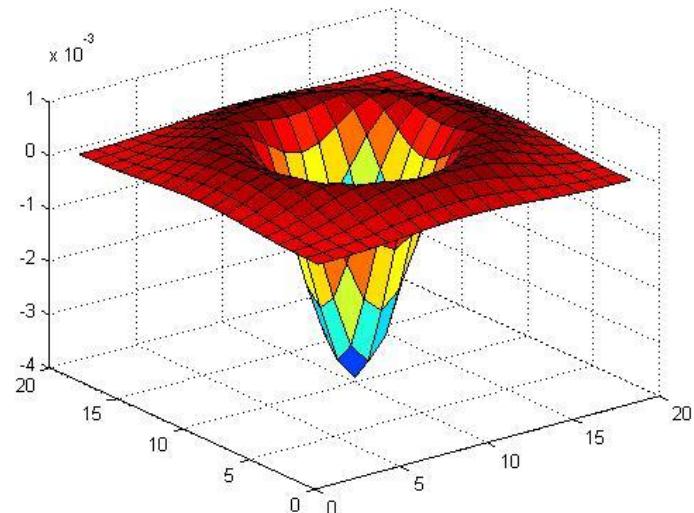
\*



## Лапласианы также чувствительны к шуму

---

- И снова, используем сглаживание
- Можем использовать один совместный кернел
- Laplacian of Gaussian, LoG
- Хорошая аппроксимация при  
5x5 - 9x9 кернелях



## Еще один детектор границ

---

- Изображение это функция
  - Есть высокочастотные и низкочастотные компоненты
  - Преобразование Фурье?
- Границам соответствуют высокие частоты
- Может мы хотим находить границы определенного размера (т.е. определенной частоты)

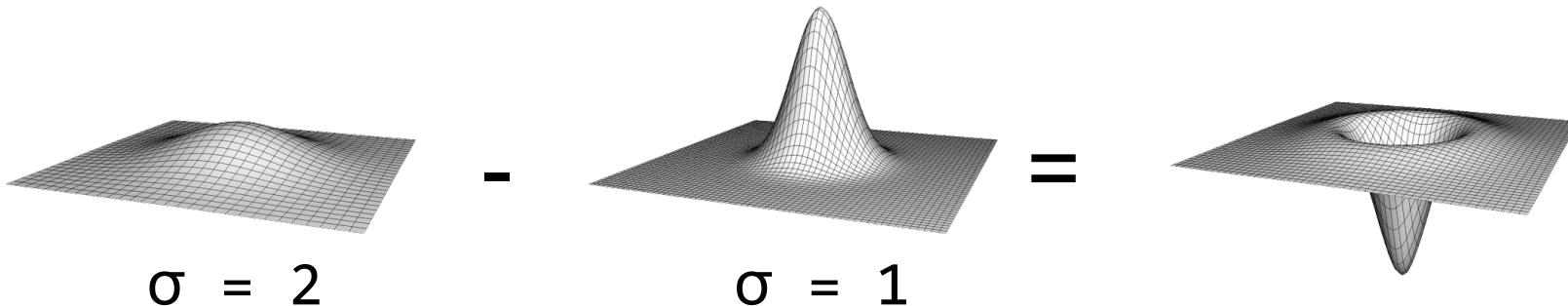
# Difference of Gaussian (DoG)

---

- Гауссиана это фильтр низких частот
- Убивает компоненты с частотой  $f < \sigma$
- $(g * I)$  - низкочастотные компоненты
- $I - (g * I)$  высокочастотные компоненты
- $g(\sigma_1) * I - g(\sigma_2) * I$ 
  - Компоненты между этими частотами ("среднечастотные")
- $g(\sigma_1) * I - g(\sigma_2) * I = [g(\sigma_1) - g(\sigma_2)] * I$

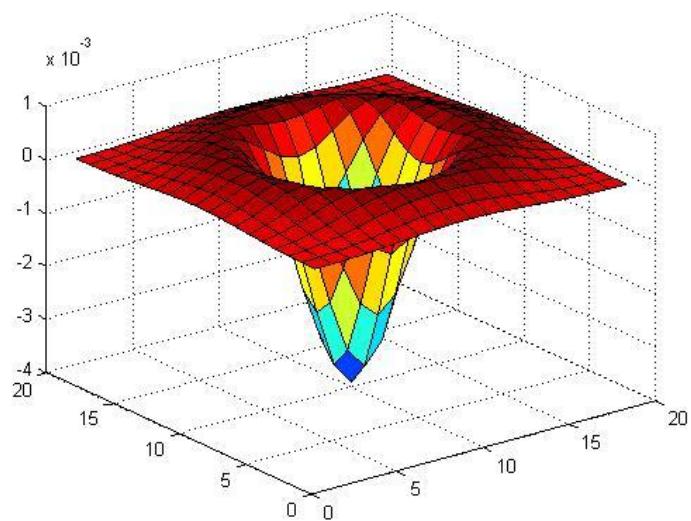
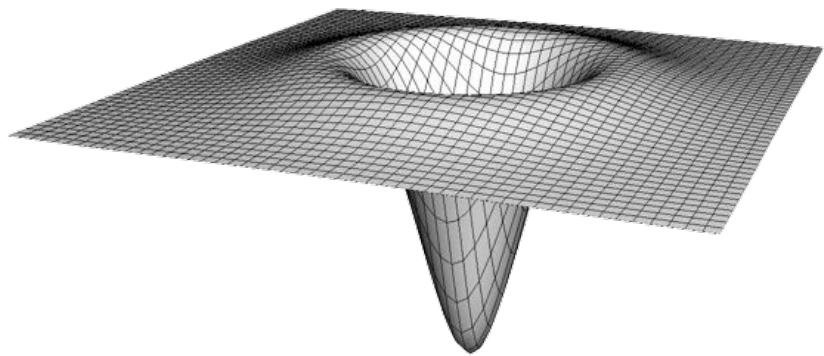
# Difference of Gaussian (DoG)

---



# Difference of Gaussian (DoG)

---



# DoG (1 - 0)

---



## DoG (2 - 1)

---



## DoG (3 - 2)

---



## DoG (4 - 3)

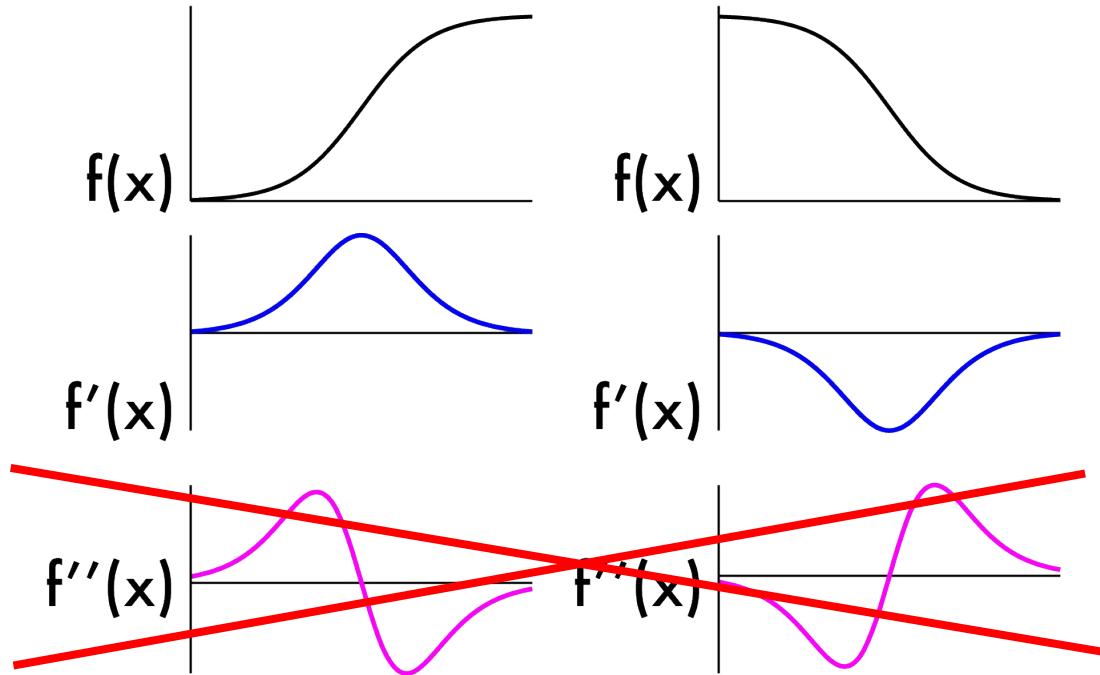
---



## Другой подход: магнитуда градиентов

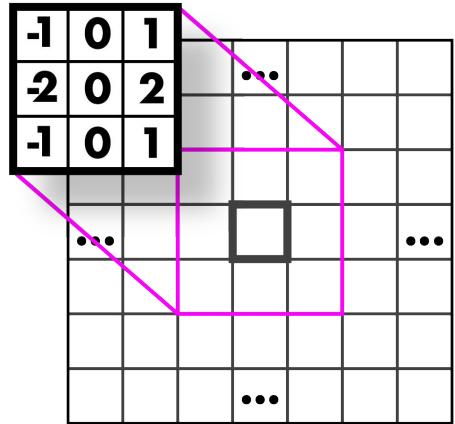
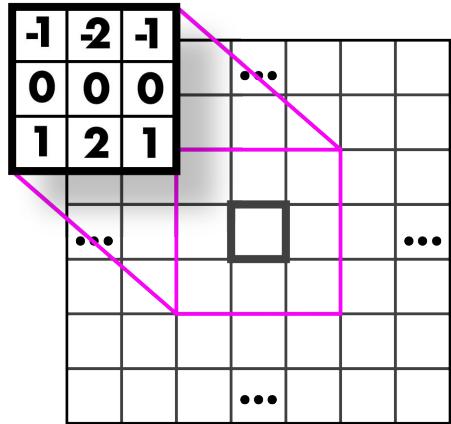
---

- Не нужны вторые производные
- Просто берем магнитуду градиентов



# Другой подход: магнитуда градиентов

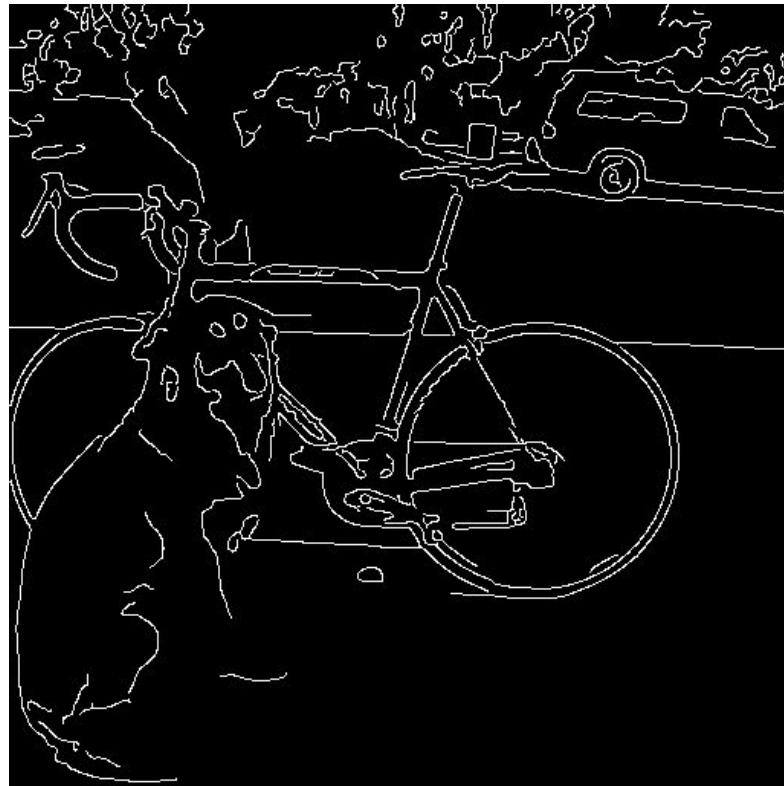
---





# Хотим узкие границы

---



# Детектор границ Кэнни

---

- Ваш первый image processing pipeline!
  - Олдскульное CV целиком состоит из пайплайнов

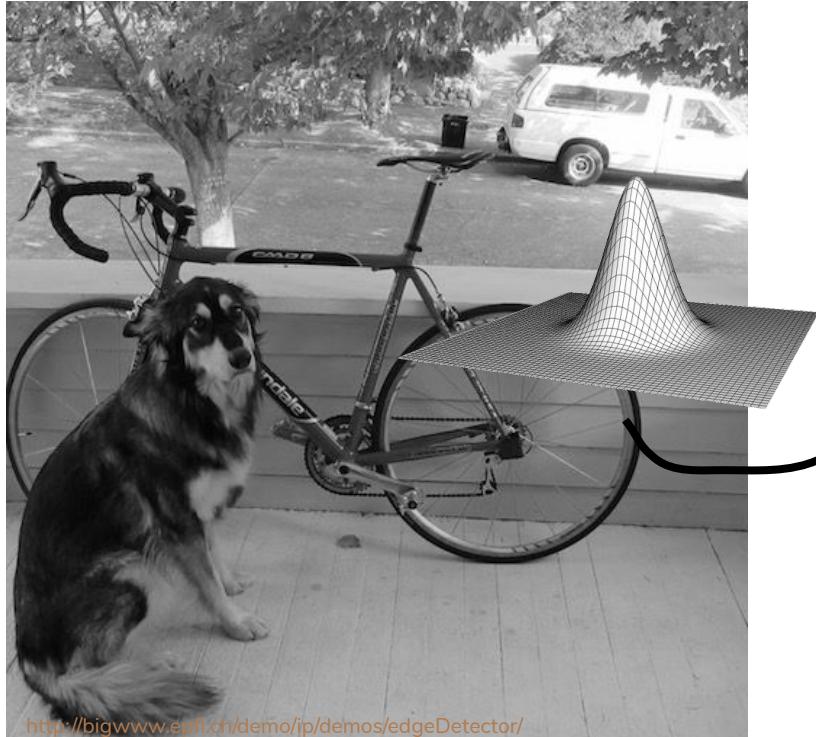
Алгоритм:

- Сгладить изображение
- Посчитать направление и магнитуду градиентов
- Non-maximum suppression
- Двойной трешхолдинг (strong, weak edges)
- Соединение компонент (гистерезис)

# Сглаживание изображений

---

- Уже знаем как - гауссианы!

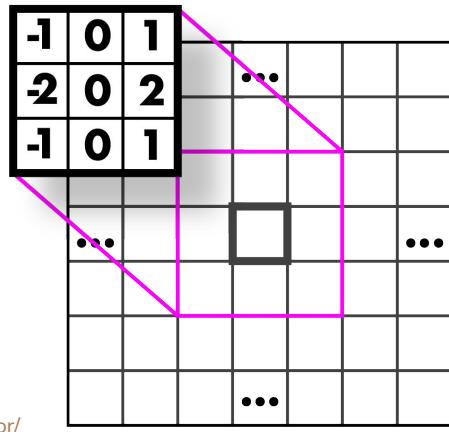
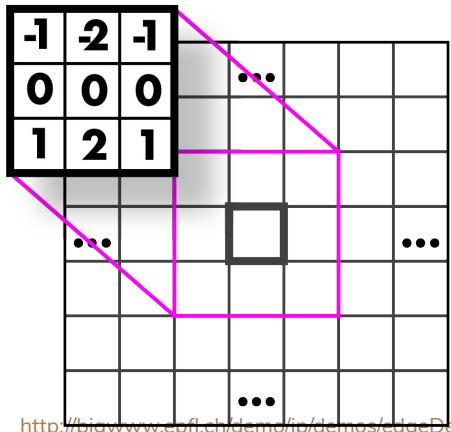


<http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/>



# Направление и магнитуда градиентов

- Фильтр Собеля



# Non-maximum suppression

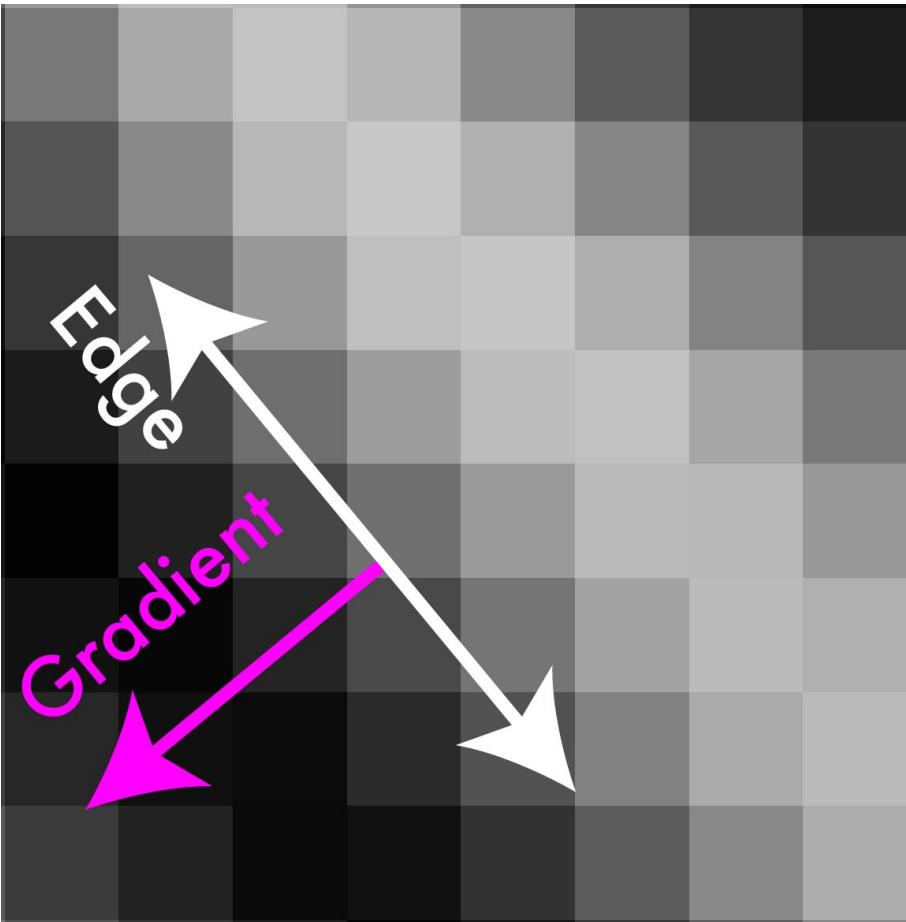
---

- Хотим границы “шириной” в один пиксель
- Сравниваем с соседними пикселями
- Смотрим максимальное ли значение



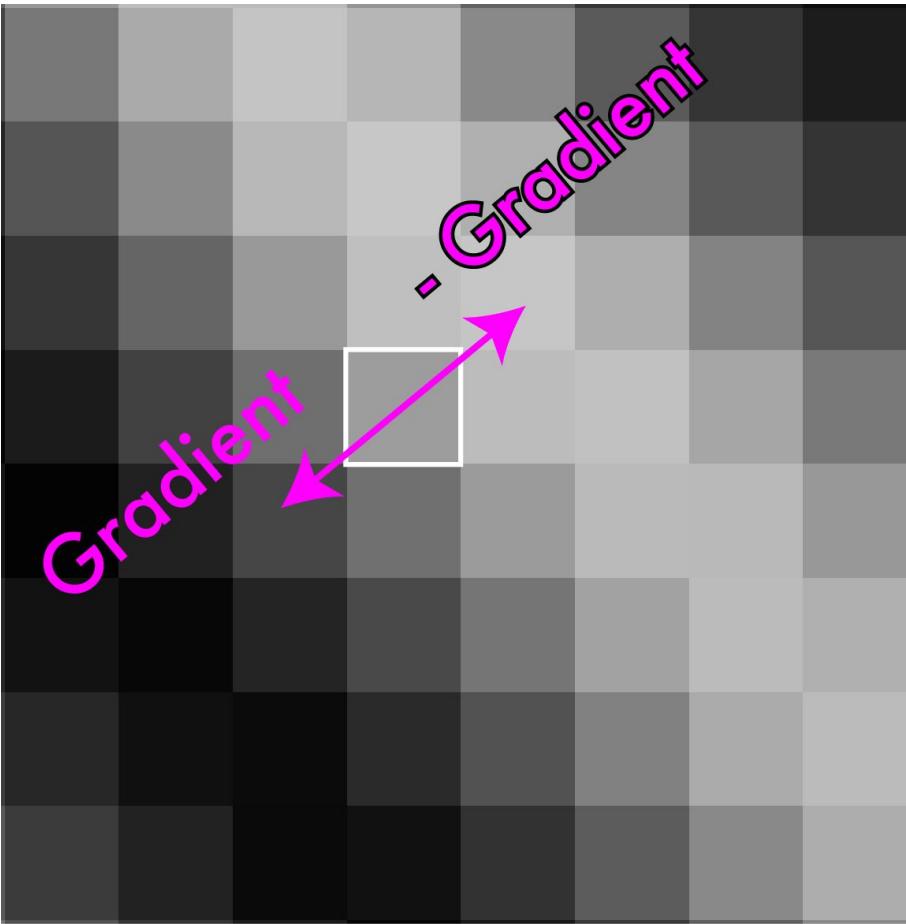
# Non-maximum suppression

---



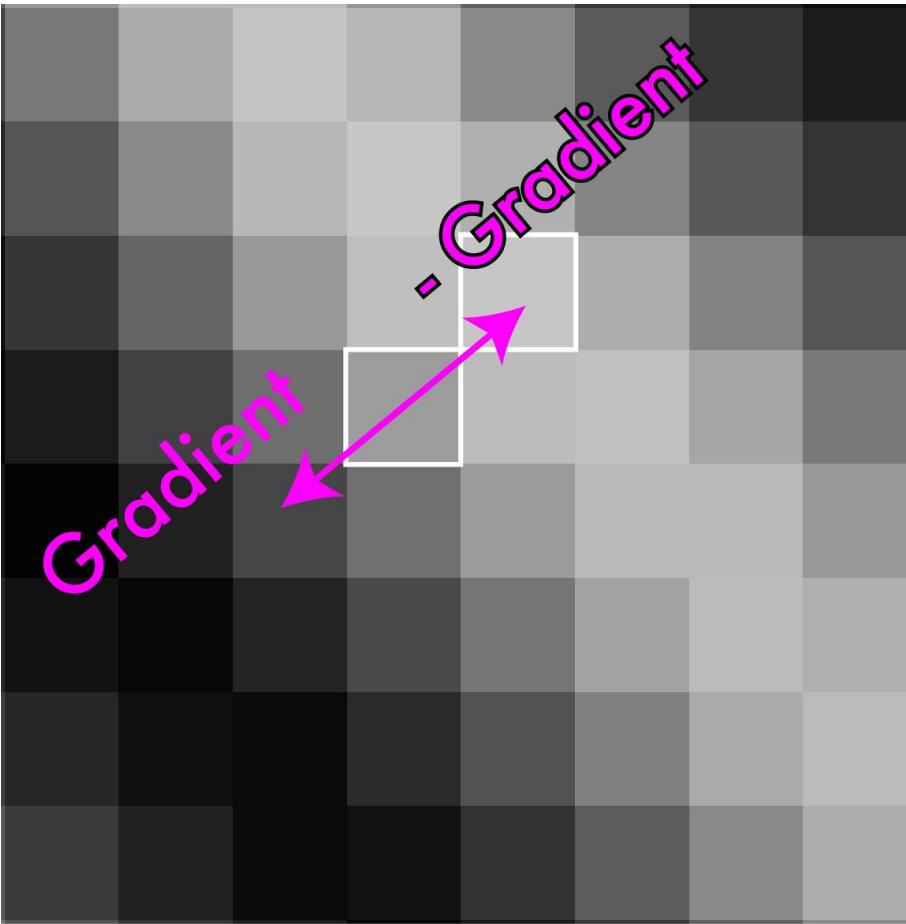
# Non-maximum suppression

---



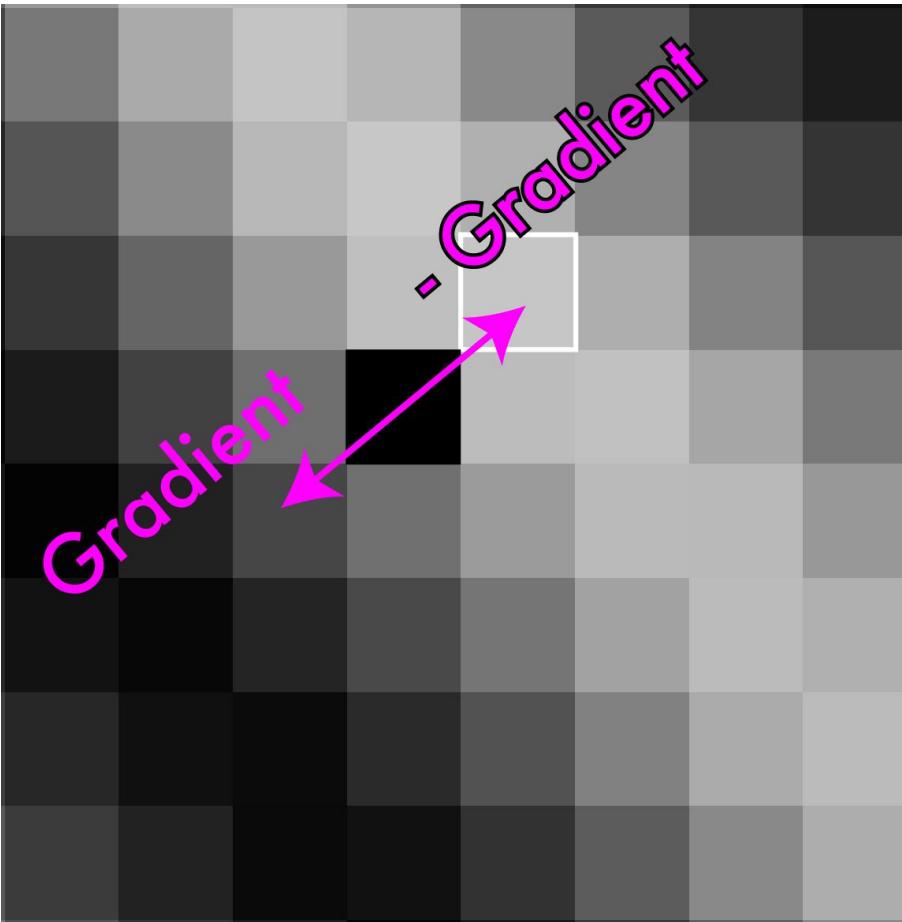
# Non-maximum suppression

---



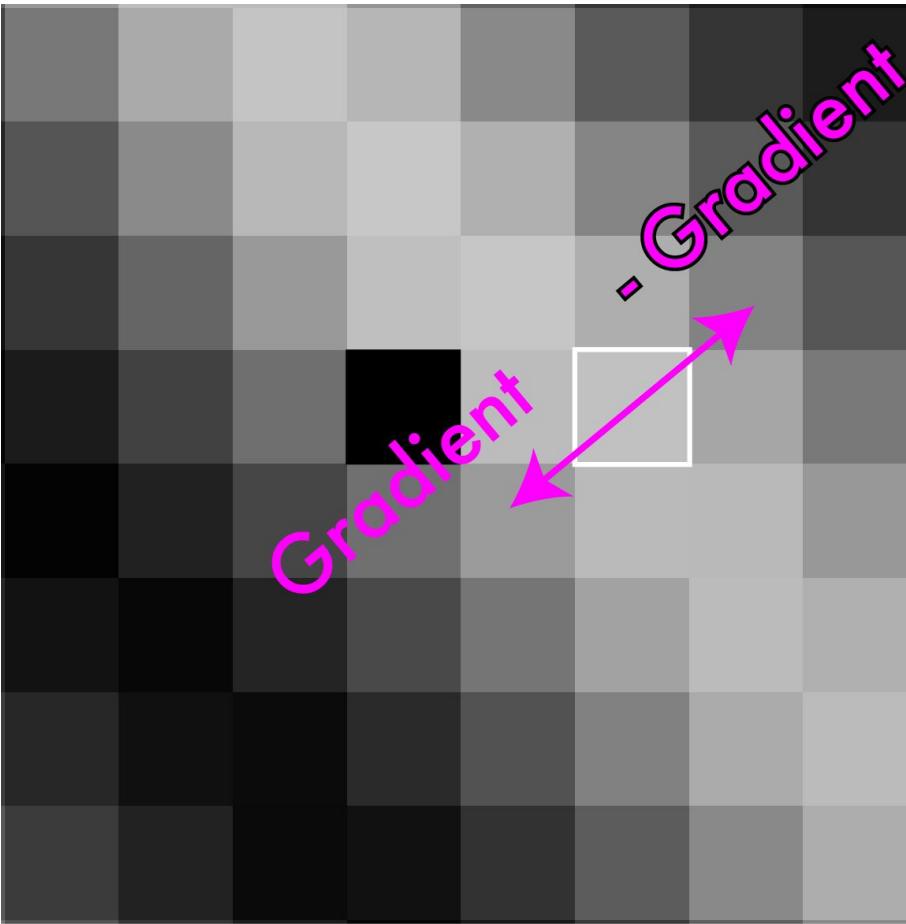
# Non-maximum suppression

---



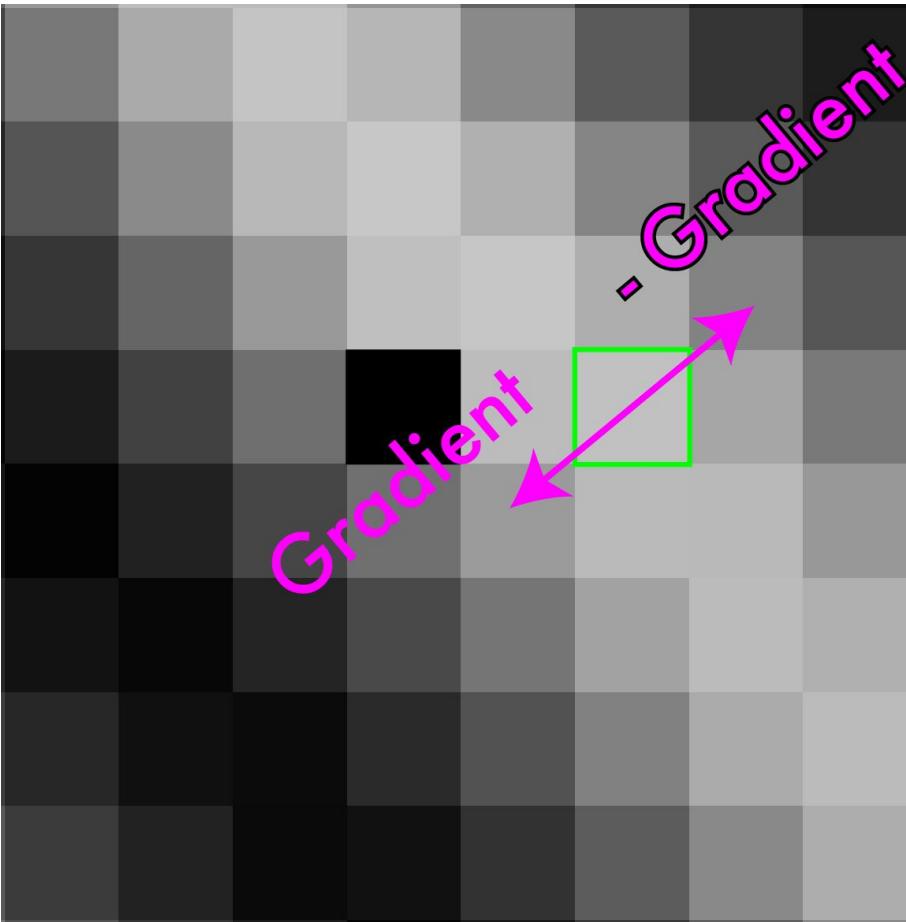
# Non-maximum suppression

---



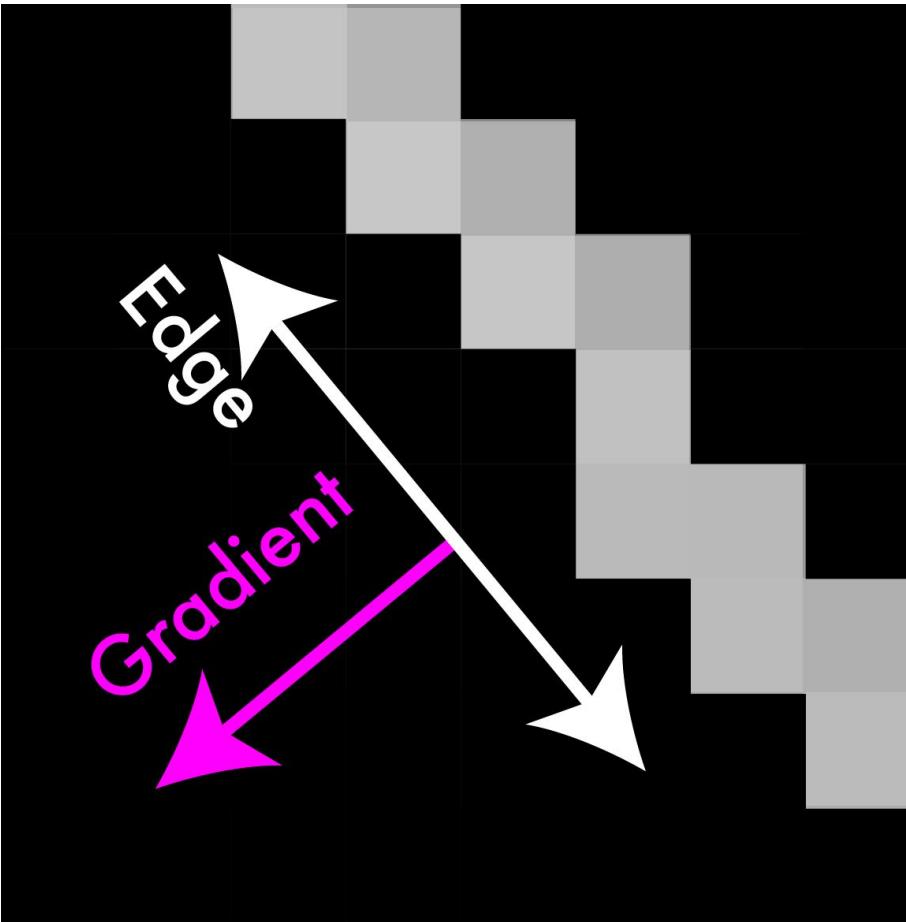
# Non-maximum suppression

---



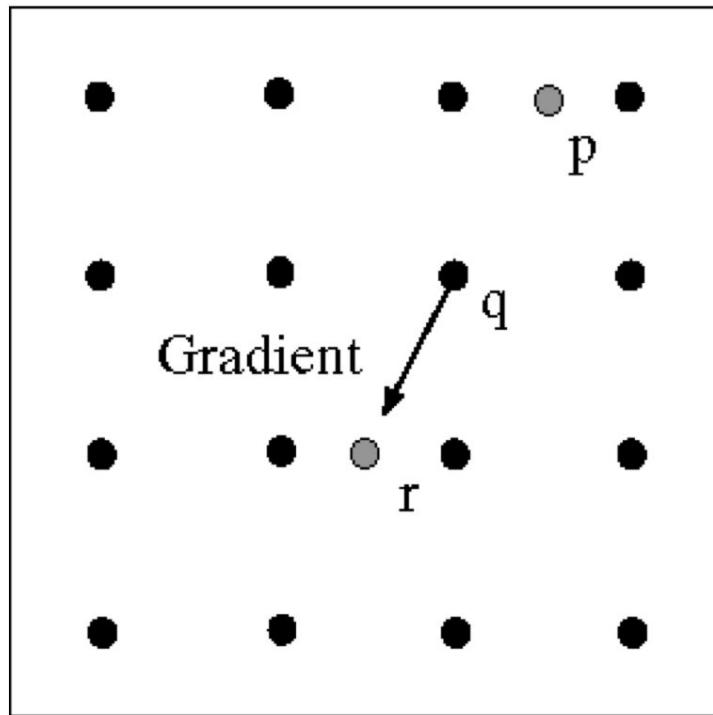
# Non-maximum suppression

---



# Non-maximum suppression

---



По-хорошему надо  
интерполировать!



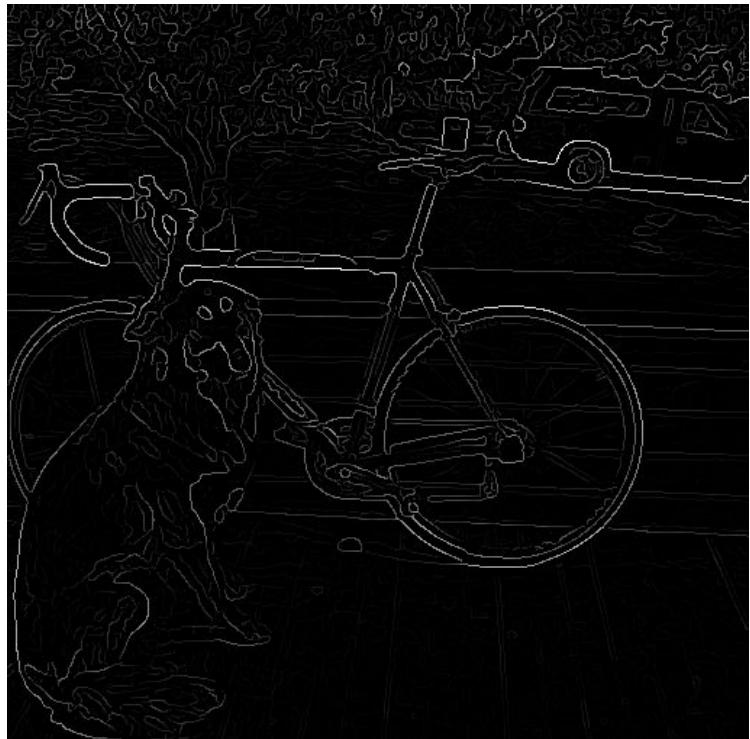
# Non-maximum suppression



# Трешхолдинг границ

---

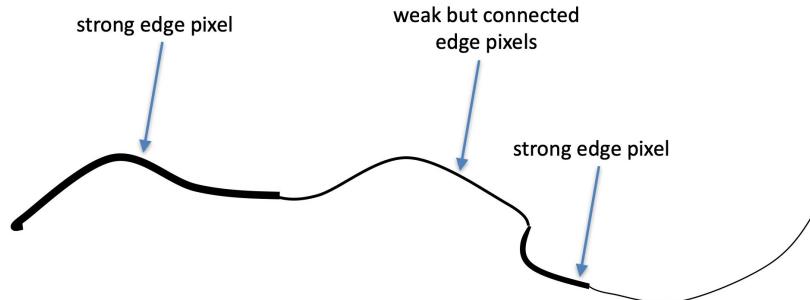
- Хотим бинарную маску (граница/не граница)
- Все еще зашумлено
- 2 порога, 3 случая
  - $R > T$ : strong edge
  - $R < T$ , но  $R > t$ : weak edge
  - $R < t$ : no edge
- Почему 2 порога?



# Соединяем их!

---

- strong edges - точно граница
- weak edges - граница, тогда и только тогда, когда соединен с strong edges
- Смотрим на соседние пиксели (8 ближайших)

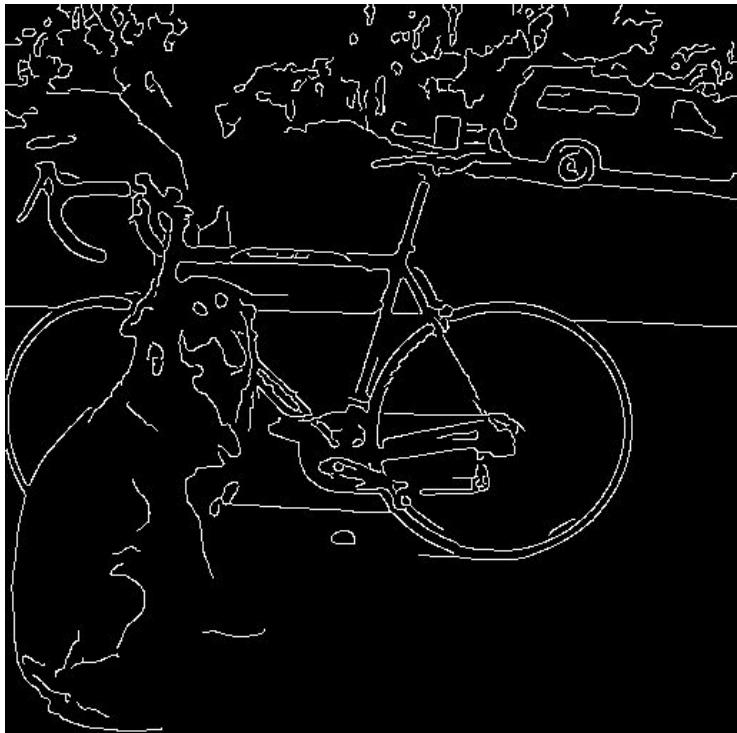


Source: S. Seitz

# Соединяем их!

---

- strong edges - точно граница
- weak edges - граница, тогда и только тогда, когда соединен с strong edges
- Смотрим на соседние пиксели (8 ближайших)



# Детектор границ Кэнни

---

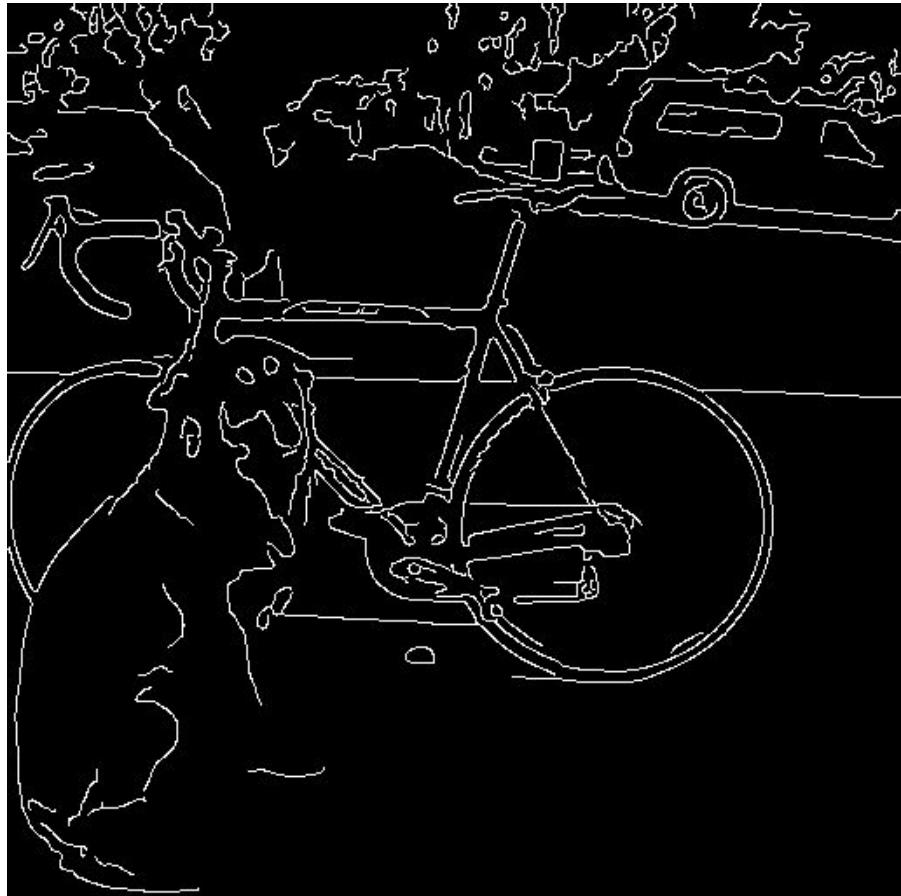
- Ваш первый image processing pipeline!
  - Олдскульное CV целиком состоит из пайплайнов

Алгоритм:

- Сгладить изображение
- Посчитать направление и магнитуду градиентов
- Non-maximum suppression
- Двойной трешхолдинг (strong, weak edges)
- Соединение компонент (гистерезис)

Гиперпараметры: sigma, значения порогов

# Canny Edge Detection



# Признаки изображения

## Что можно делать с конволюциями?

---

- Blurring
- Sharpening
- Edges
- Features
- Derivatives
- Super-resolution
- Classification
- Detection
- Image captioning
- ...

# Признаки!

- Наиболее информативные регионы
- Способы описания этих регионов
- Полезно для:
  - Матчинга
  - Распознавания
  - Детектирования

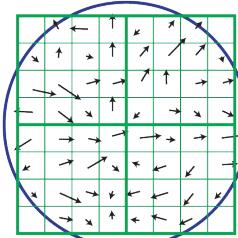
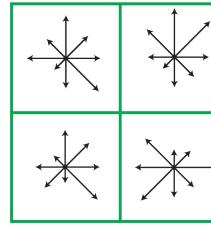
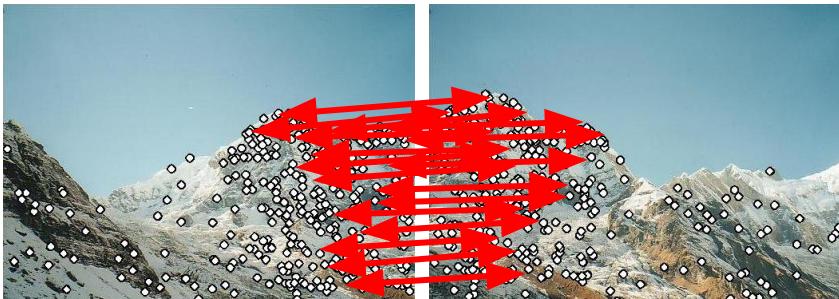
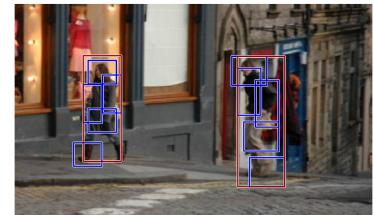


Image gradients



Keypoint descriptor



# Что есть хороший признак?

---

- Хотим найти патчи в изображение, которые полезные или имеют какой-то смысл
- Для объекта хотим найти неповторимый патч именно для этого объекта
- Для панорам, хотим находить патчи которые можно легко найти и сопоставить во всех изображениях
- Хорошие признаки уникальны!
  - Можем легко найти “тот же самый” признак
  - Не перепутать с другими признаками

# Насколько близки два патча

---

- Сумма квадратов разности
- Изображения I, J
- $\sum_{x,y} (I(x,y) - J(x,y))^2$

# Как найти хороший признак

---

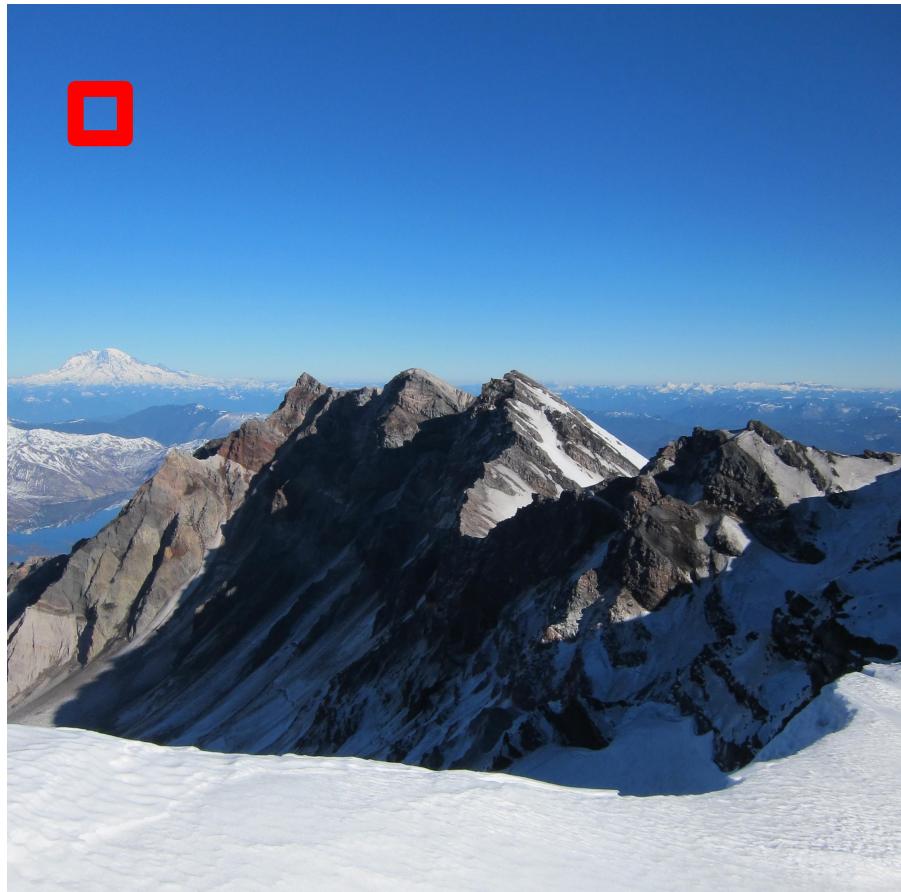
- Допустим мы делаем панораму
- Хотим патчи в одном изображения сматчить с патчами в другом
- Связь “один к одному”



# Как найти хороший признак

---

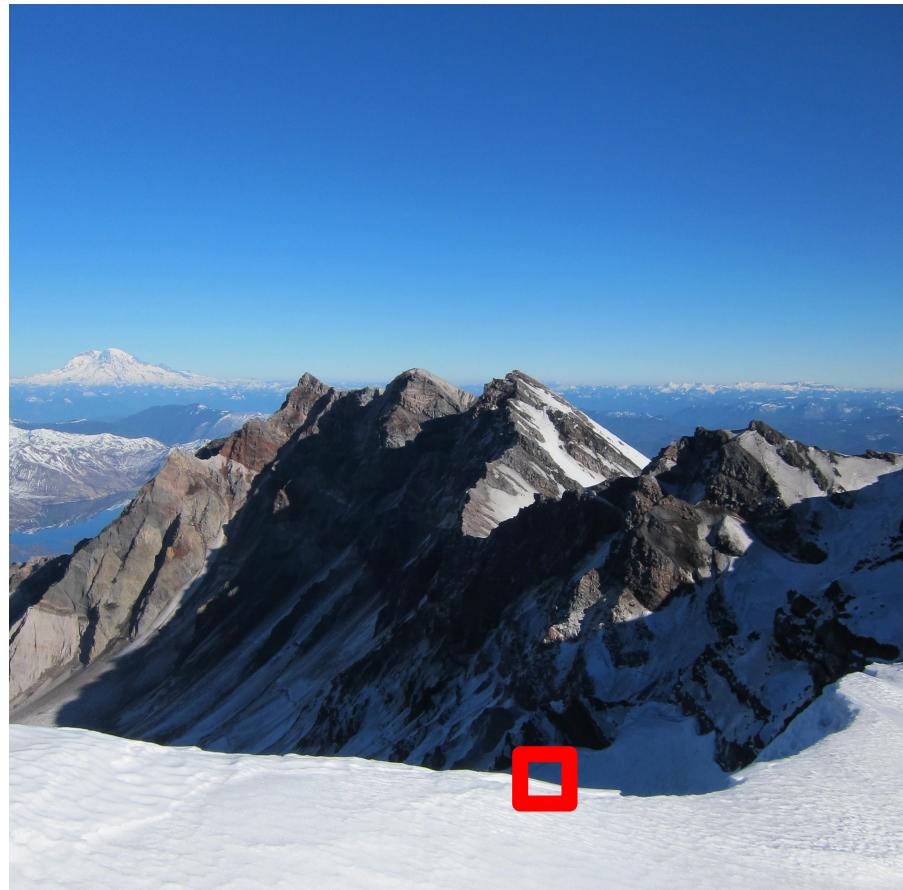
- Небо: плохо
  - Очень маленькая вариация
  - Неразличимо



# Как найти хороший признак

---

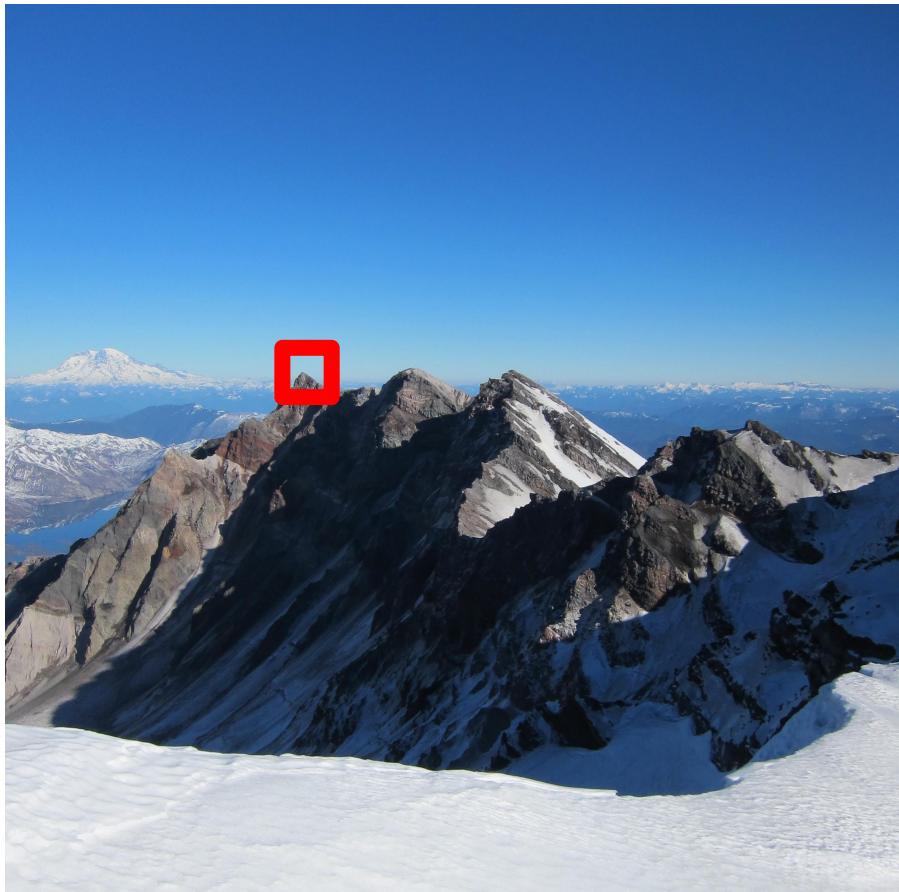
- Небо: плохо
  - Очень маленькая вариация
  - Неразличимо
- Граница: получше
  - Вариация в одном направлении
  - Может сматчиться с любым другим патчом вдоль этого направления



# Как найти хороший признак

---

- Небо: плохо
  - Очень маленькая вариация
  - Неразличимо
- Граница: получше
  - Вариация в одном направлении
  - Может сматчиться с любым другим патчом вдоль этого направления
- Углы: супер!
  - Угол ункален



# Как найти хороший признак

---

- Хотим патч уникальный в изображении
- Можем посчитать расстояние между патчем и каждым другим патчем
- Много вычислений!



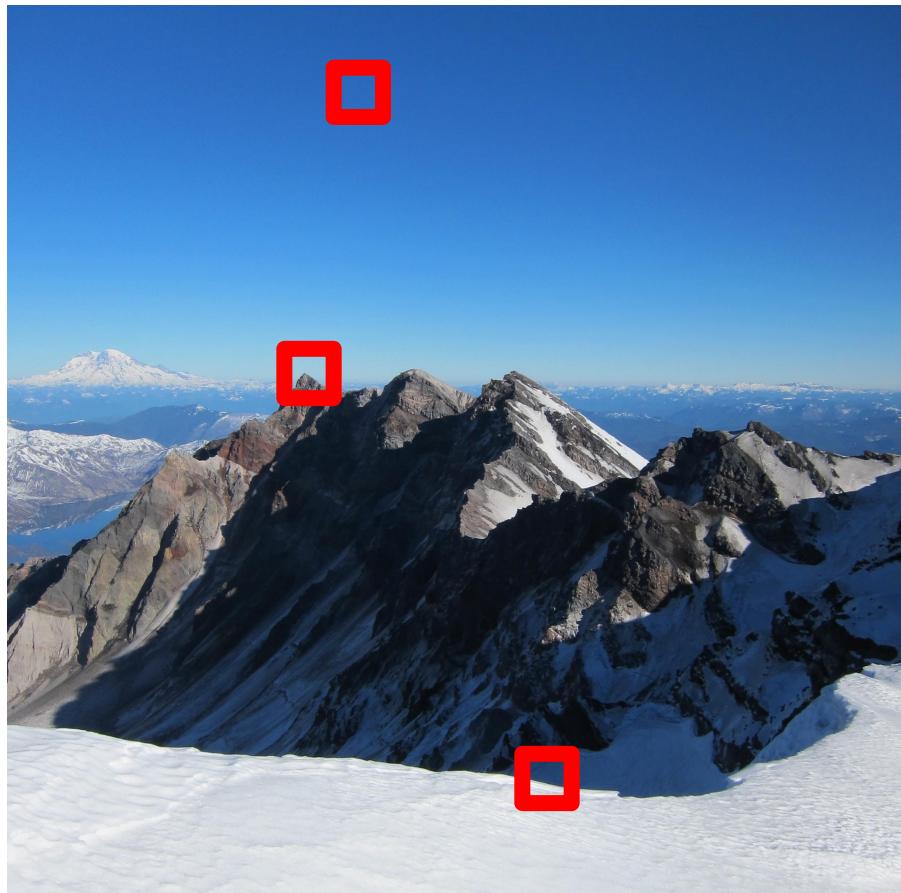
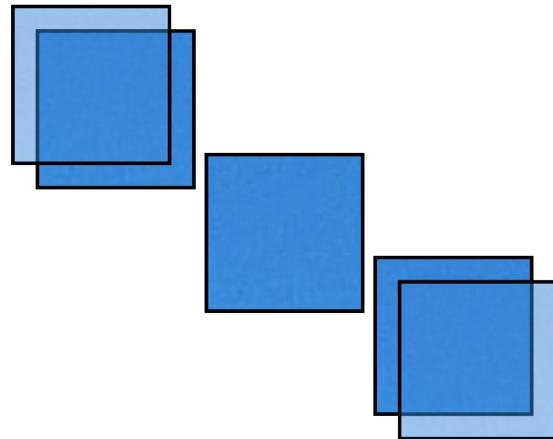
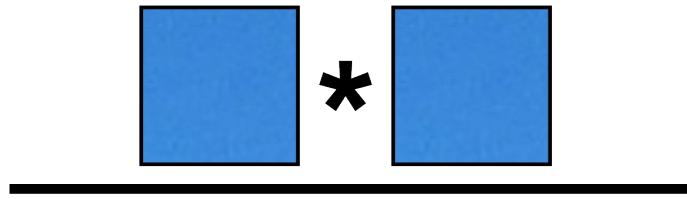
# Как найти хороший признак

---

- Хотим патч уникальный в изображении
- Можем посчитать расстояние между патчем и каждым другим патчем
- Много вычислений!
- Воспользуемся автокорреляцией:
  - Насколько хорошо изображение матчится со смещенной версией самого себя?
- $\sum_d \sum_{x,y} (I(x,y) - I(x+d_x, y+d_y))^2$
- Мера непохожести

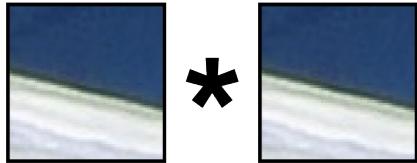
# Мера непохожести

Небо: везде низкая

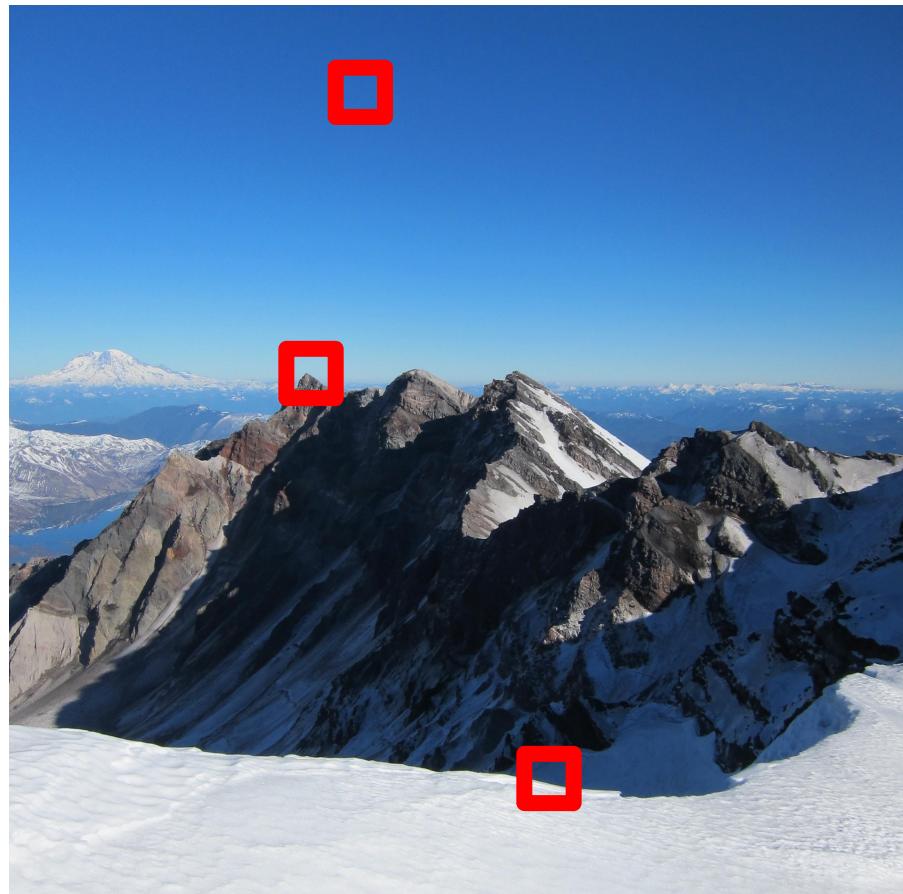
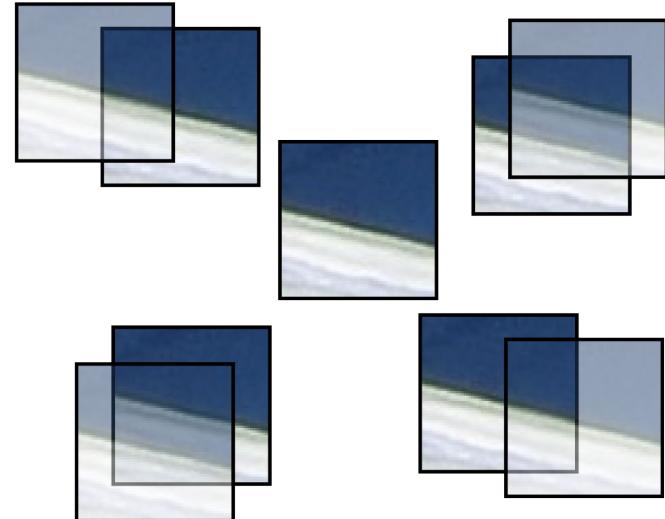


# Мера непохожести

Край: низкая вдоль границы



\*

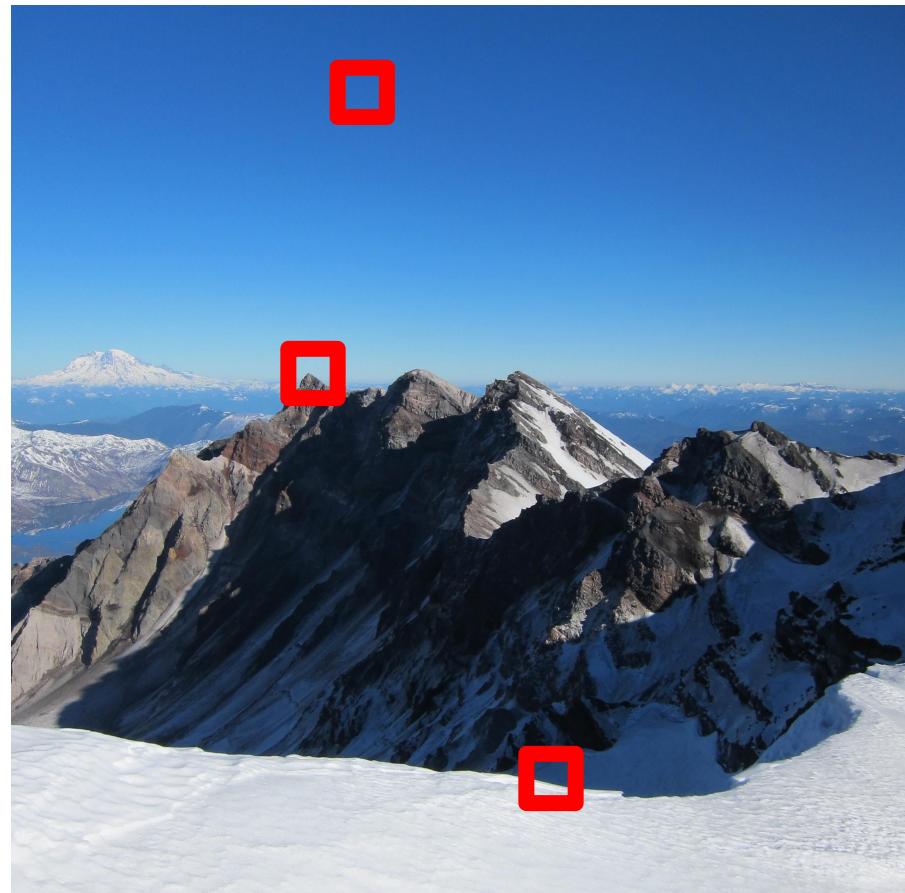


# Мера непохожести

Угол: везде высокая



\*



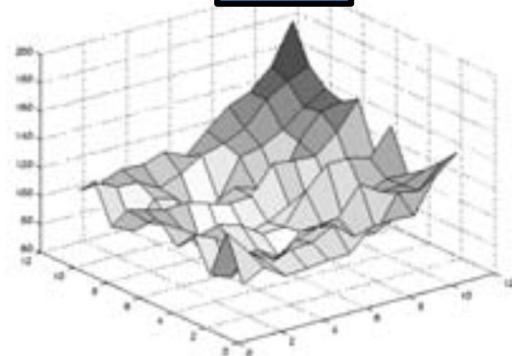
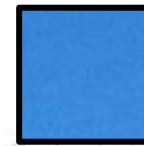
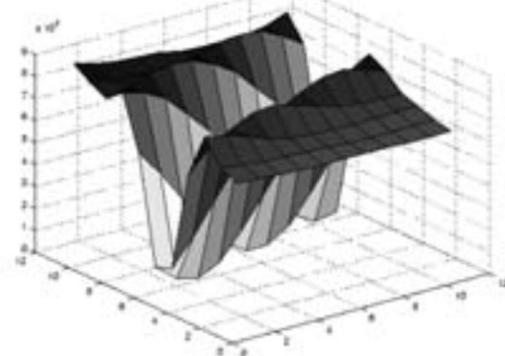
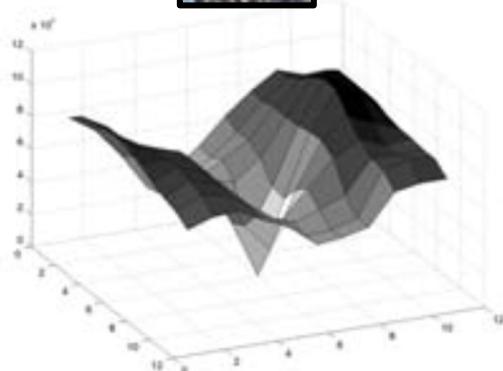
# Мера непохожести

---

Небо: везде низкая

Край: низкая вдоль границы

Угол: везде высокая





It's coding time!