Шумоподавление

Причины возникновения шума:

- Несовершенство измерительных приборов
- Хранение и передача изображений с потерей данных

Шум фотоаппарата

Сильное сжатие JPEG

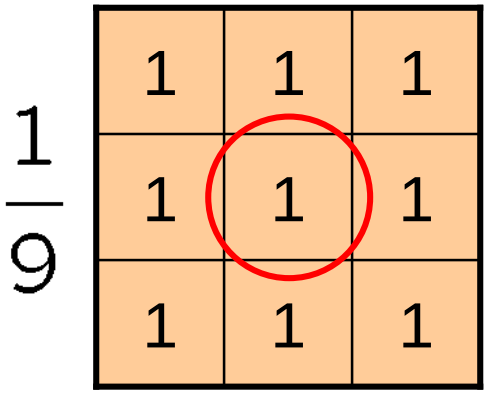
Подавление шума

Пусть дана камера и статичная сцена, требуется подавить шум. Шум случайная величина.

Простейший вариант: усреднить несколько кадров

Усреднение

- Заменим каждый пиксель взвешенным средним по окрестности
- Веса обозначаются как *ядро фильтра*
- Веса для усреднения задаются так:



“box filter”

Пример самого простого фильтра для шумоподавления

Определение свертки

Будем считать, что задано исходное полутоновое изображение *I*, и обозначим интенсивности его пикселей *I* (*x*, *y*). Линейный фильтр определяется вещественнозначной функцией *F*, заданной на растре. Данная функция называется ядром фильтра, а сама фильтрация производится при помощи операции дискретной свертки (взвешенного суммирования) по следующей формуле:

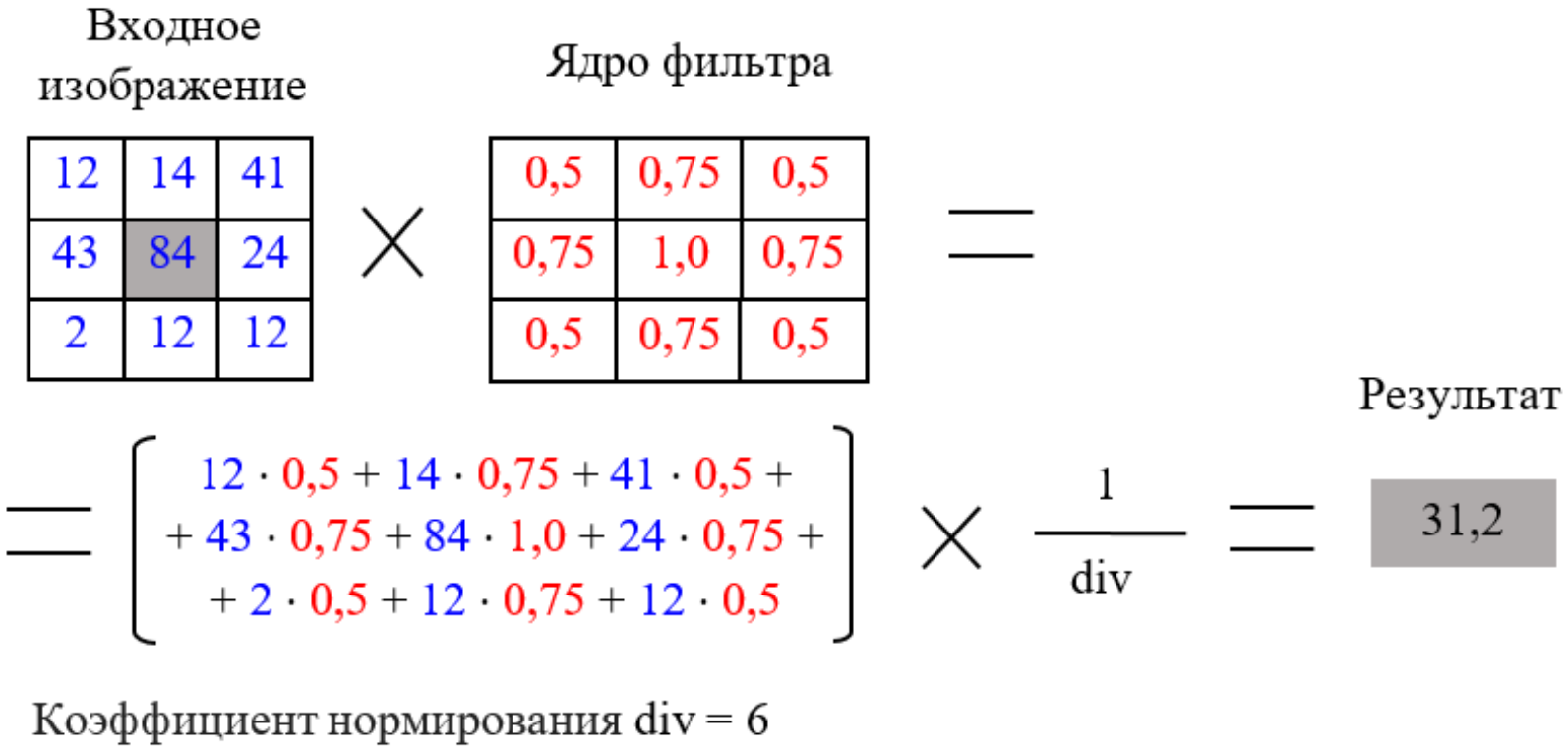
$$I'(x,y) = \sum_i \sum_j F(i,j) \cdot I(x+i, y+j)$$

Свёртка — это операция **вычисления нового значения** заданного пикселя, при которой **учитываются значения окружающих его соседних пикселей**.

Главным элементом свёртки является т.н. **ядро свёртки** — это матрица (произвольного размера и отношения сторон; чаще всего используется квадратная матрица (по-умолчанию, 3x3)).

Определение свертки

Ядро свёртки – это числовая матрица с подобранными коэффициентами. Эти коэффициенты называют степенями влияния или “ценностями”, определяющими вес пикселей из рассматриваемой окрестности.

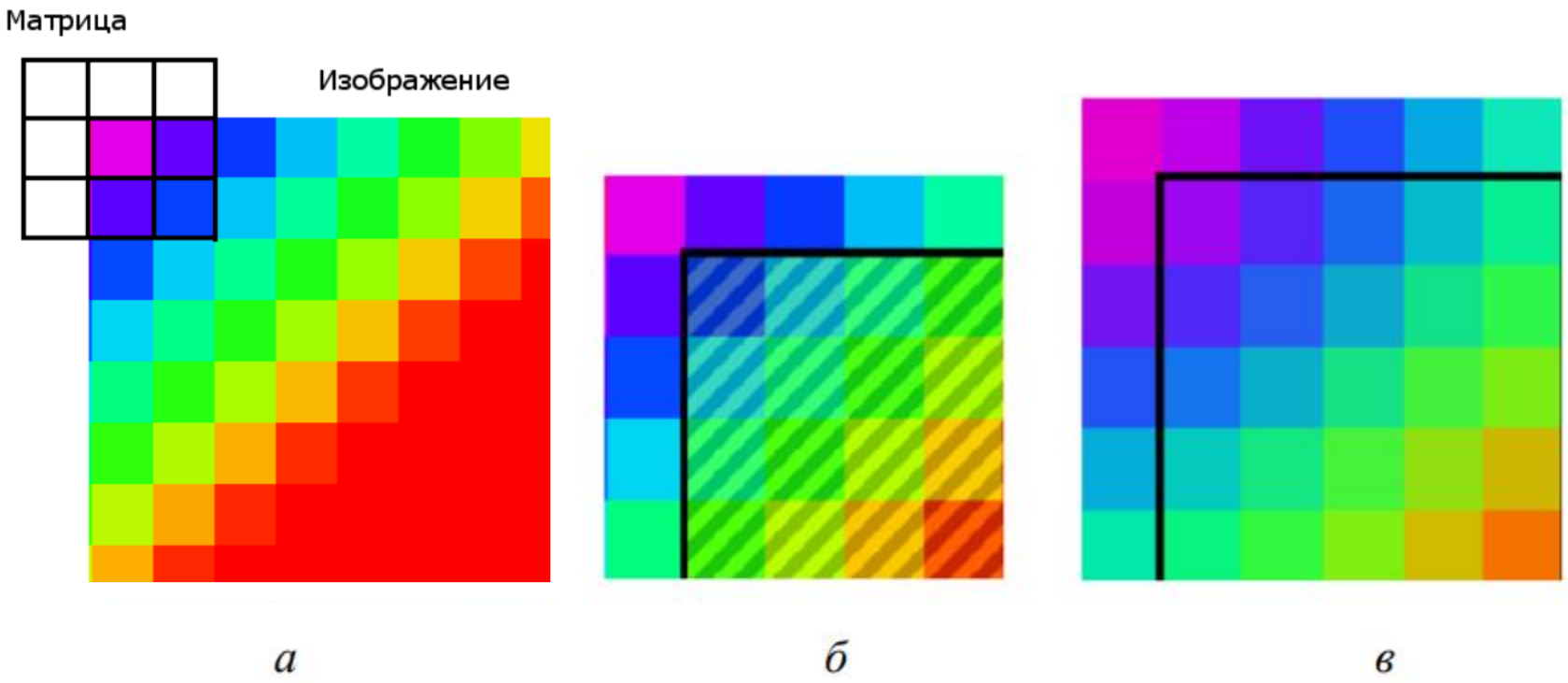


Источник [Матричные фильтры обработки изображений / Хабр (habr.com)]

Хорошая визуализация процессов свертки [Визуальное объяснение ядер изображений (setosa.io)]

Граничные условия

Эта проблема актуальна для всех матричных фильтров



Граничные условия: а – исходное изображение; б – обрезка граничных пикселей; в – дублирование граничных пикселей

Источник Матричные фильтры обработки изображений / Хабр (habr.com)

Для вычисления сверток в библиотеке OpenCV присутствует функция **filter2D**():

```
void filter2D(const Mat& src, Mat& dst, int ddepth,
        const Mat& kernel, Point anchor=Point(-1, -1),
        double delta=0, int borderType=BORDER_DEFAULT)
```

```
const float kernelData[] = { -1.0f, -1.0f, -1.0f,
                             -1.0f, 9.0f, -1.0f,
                             -1.0f, -1.0f, -1.0f };
const Mat kernel(3, 3, CV_32FC1, (float*)kernelData);
filter2D(src, dst, -1, kernel);
```

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

Какое изображение получим в итоге применения фильтра с таким ядром

?

Основные свойства операции свертки

- **Линейность:** filter($f_1 + f_2$) = filter(f_1) + filter(f_2)
- **Инвариантность к сдвигу:** не зависит от сдвига пиксела: filter(shift(f)) = shift(filter(f))
- **Коммутативность**: $a * b = b * a$

  Нет никакой разницы между изображением и ядром фильтра
- **Ассоциативность**: $a * (b * c) = (a * b) * c$

  Последовательное применение фильтров: $(((a * b_1) * b_2) * b_3)$

  Эквивалентно применению такого фильтра: a * ($b_1 * b_2 * b_3$)
- **Дистрибутивность** по сложению:

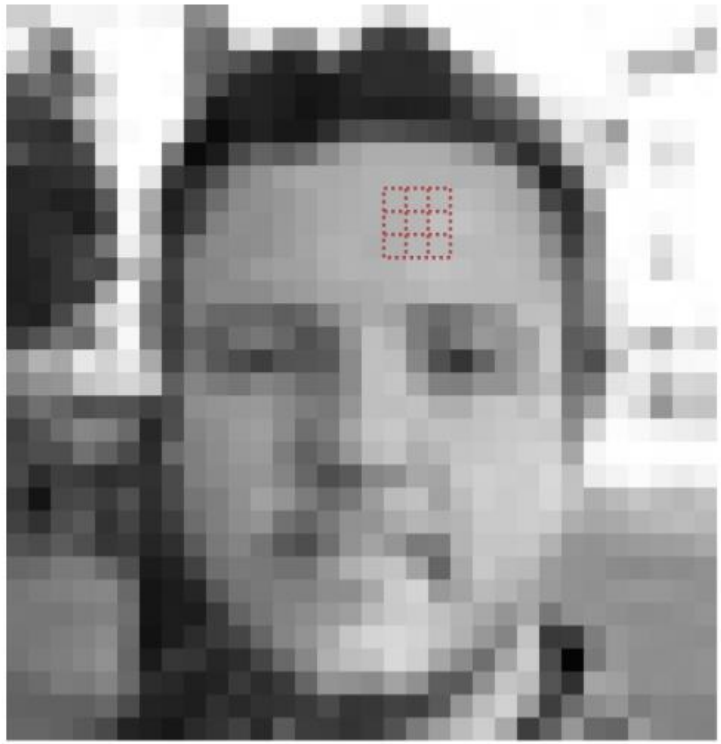
  $a * (b + c) = (a * b) + (a * c)$
- Домножение на скаляр можно вынести за скобки: $ka * b = a * kb = k\,(a * b)$
- Единица: e = […, 0, 0, 1, 0, 0, …], $a * e = a$

0	-1	0
-1	5	-1
0	-1	0

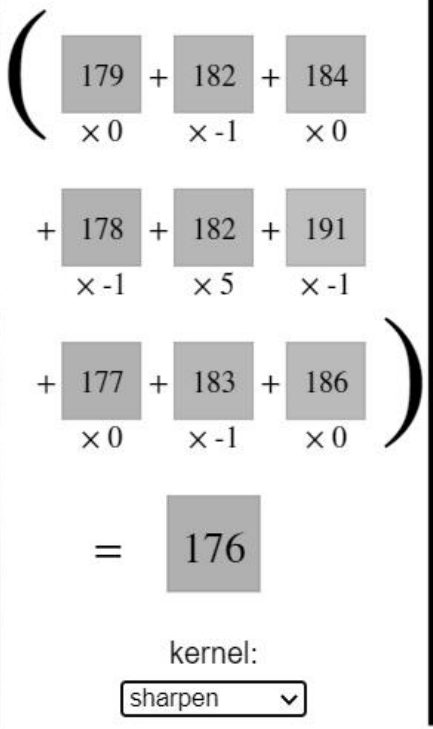
Какое изображение получим в итоге применения фильтра с таким ядром

?

повышения контраста



input image



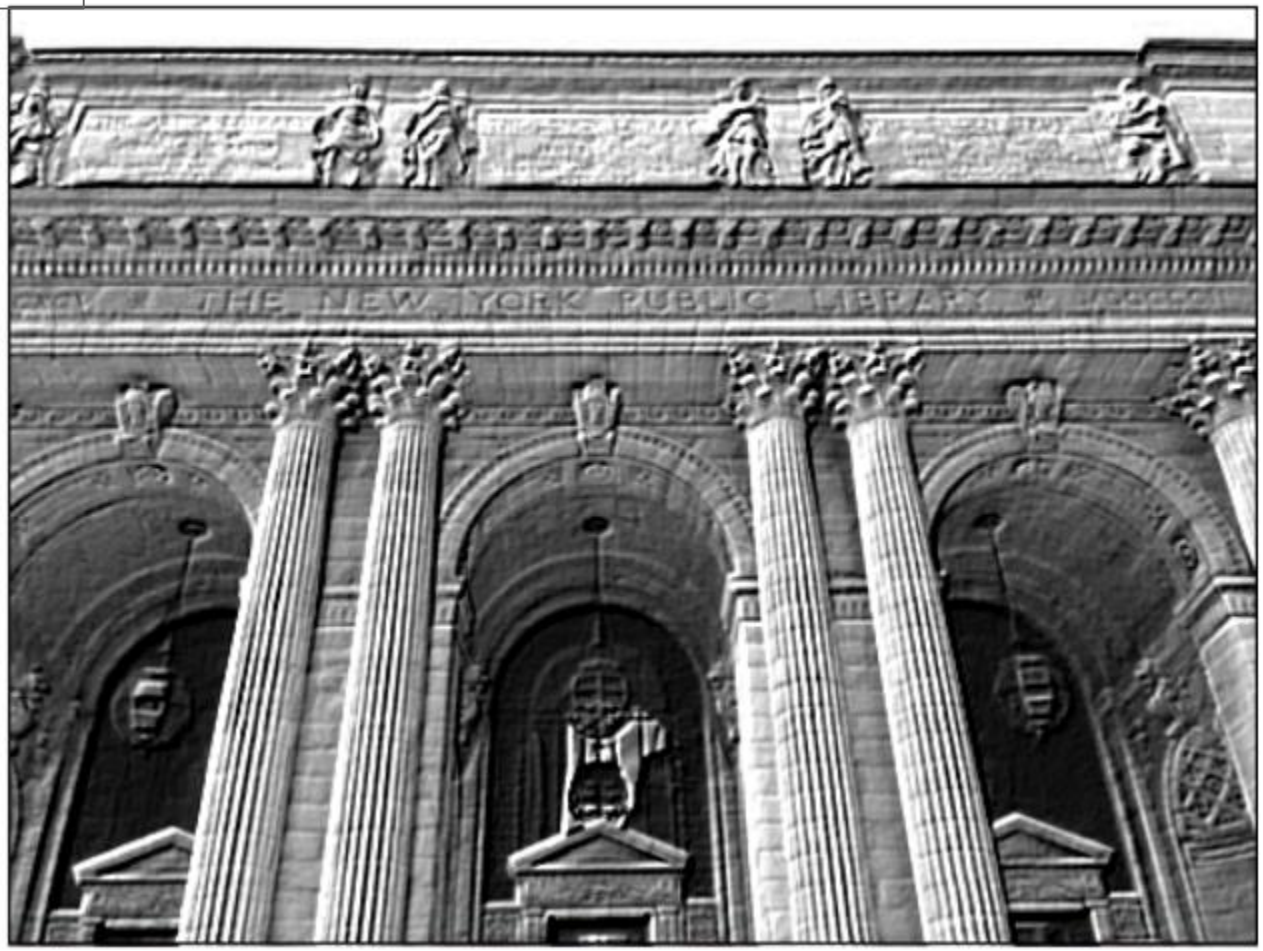
output image

Пример фильтра улучшения чёткости

$$\text{kernel} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

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

Ядро тиснения создает иллюзию глубины, подчеркивая различия пикселей в заданном направлении. В данном случае в направлении вдоль линии от левого верхнего угла к правому нижнему.



Фильтры. Сглаживание изображений

Сглаживание или размытие – это одна из самых простых и часто используемых операций обработки изображений. Как правило, размытие применяется, чтобы уменьшить шумы или артефакты.

В библиотеке OpenCV реализовано несколько функций размытия изображения, к примеру:

- blur(),
- boxFilter(),
- GaussianBlur()
- medianBlur()

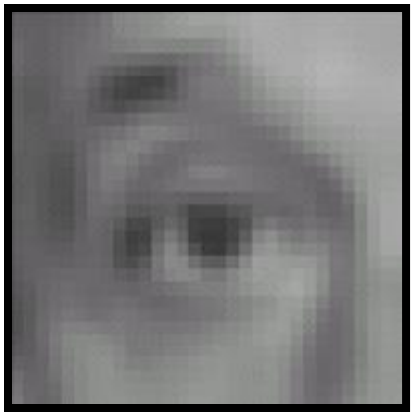
Фильтры. Сглаживание изображений

```
void blur(const Mat& src, Mat& dst, Size ksize,
      Point anchor=Point(-1, -1),
      int borderType=BORDER_DEFAULT)

void boxFilter(const Mat& src, Mat& dst, int ddepth,
      Size ksize, Point anchor=Point(-1, -1), bool
      normalize=true, int borderType=BORDER_DEFAULT)

void GaussianBlur(const Mat& src, Mat& dst, Size ksize,
      double sigmaX, double sigmaY=0,
      int borderType=BORDER_DEFAULT)

void medianBlur(const Mat& src, Mat& dst, int ksize)
```

Простейшие фильтры. Blur

Original

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



Blur (with a box filter)

$$K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \cdots & & & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

Source: D. Lowe

Функция ***blur()*** выполняет размытие посредством вычисления свертки исходного изображения с ядром K:

$$K = \frac{1}{kSize.width \cdot kSize.height} \cdot \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}.$$

Функция ***boxFilter*** использует ядро более общего вида:

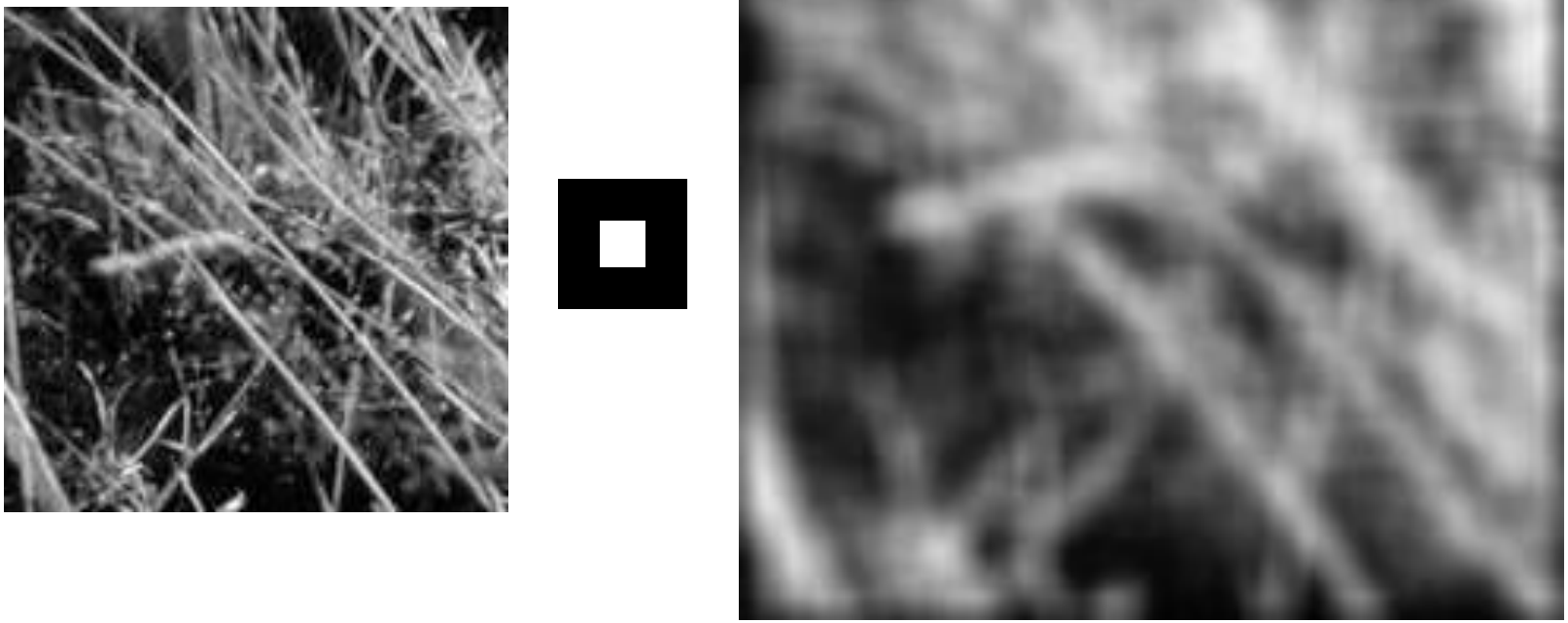
$$K = \alpha \cdot \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix},$$

где $\alpha = 1$, если ядро не нормализовано, а в случае нормализованного ядра функция ***boxFilter***() перерождается в функцию ***blur***().

Сглаживание с box-фильтром

Результат сглаживание с помощью усреднения отличается от расфокусированного изображения.

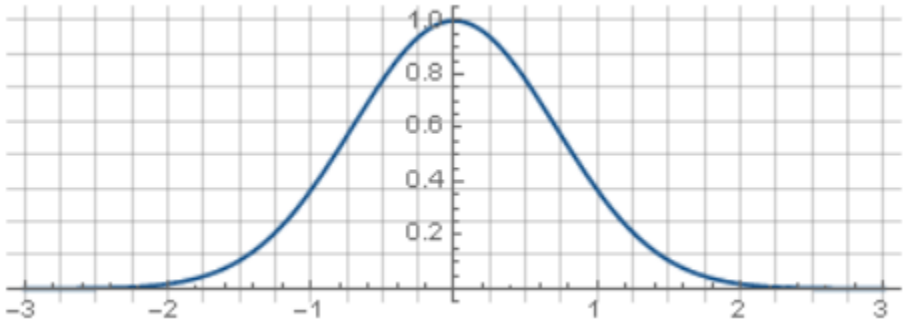
Точка света, наблюдаемая с расфокусированного объектива, выглядит как кружок света, а усреднение дает квадрат.



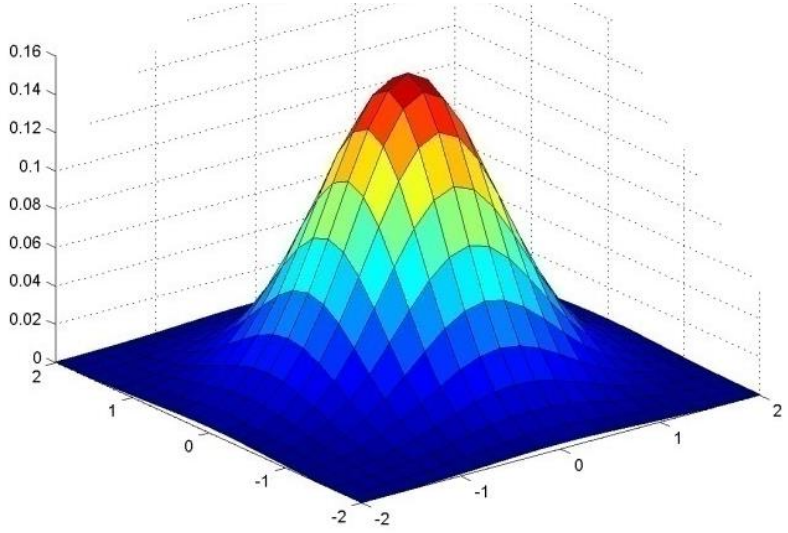
Source: D. Forsyth

Фильтр Гаусса (Gaussian Blur)

Фильтр называется Gaussian, потому что он строится из функции, известной как гауссиана,



$$e^{-x^2}$$

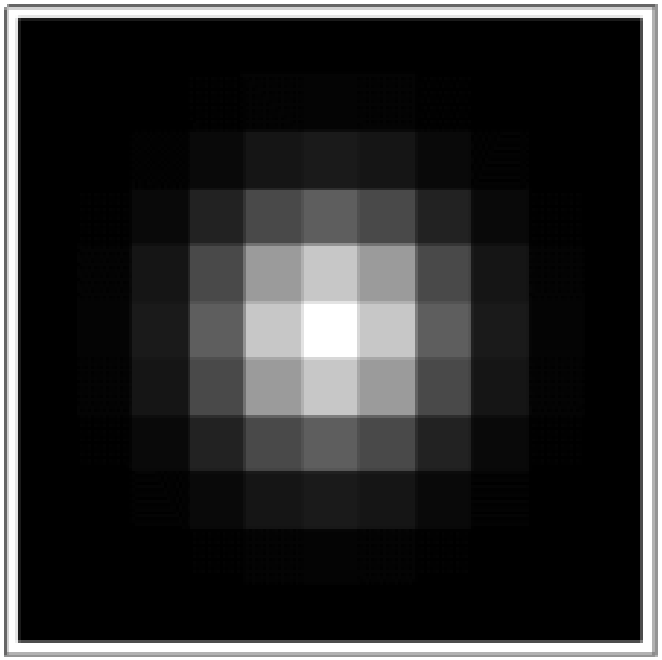


$$e^{-(x^2+y^2)}$$

Фильтр Гаусса (Gaussian Blur)

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

Для нормализации по координатам введена **сигма** (*среднеквадратическое отклонение*), которая определяет степень сжатия гауссианы



Ядро фильра Гаусса

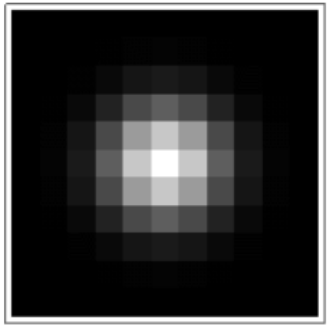
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

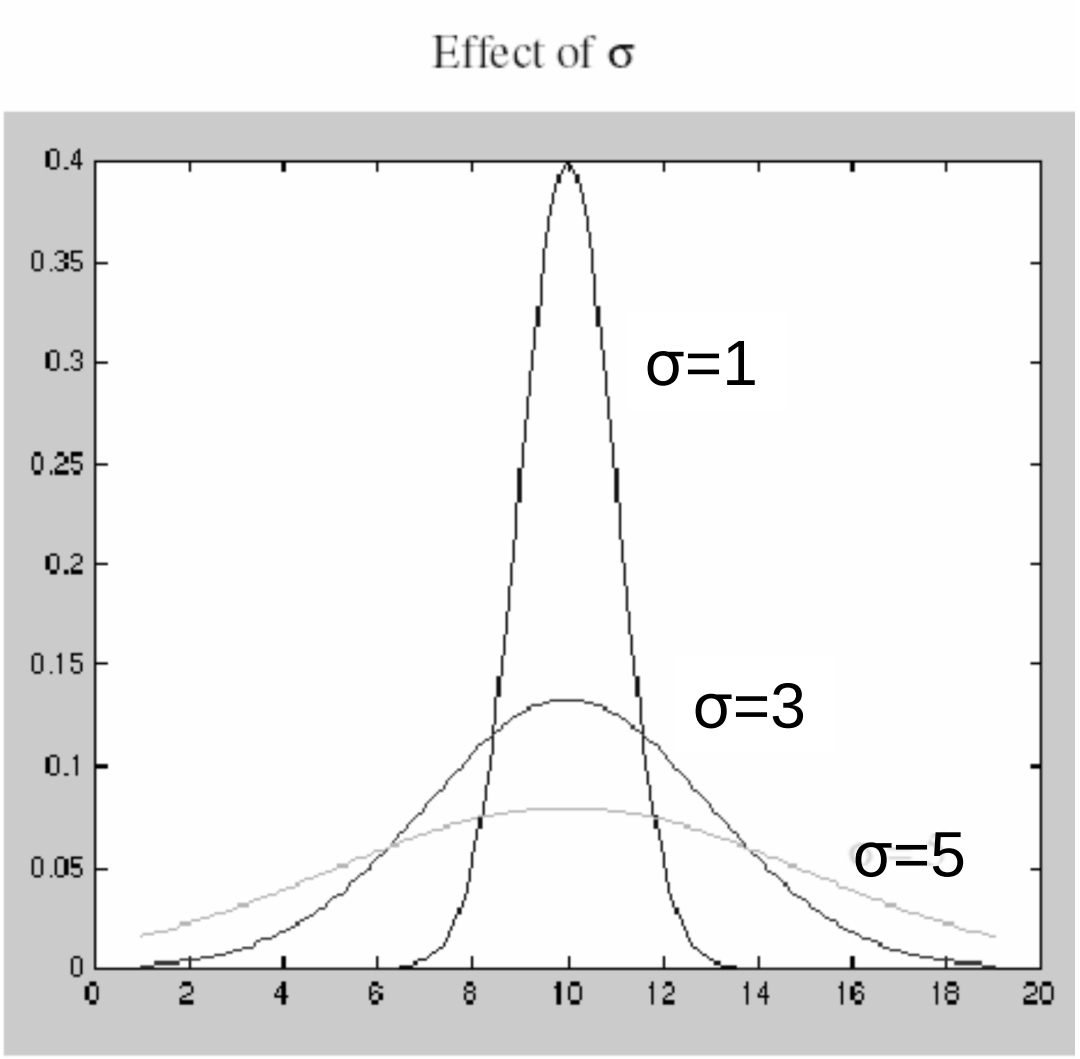
5 x 5, σ = 1

Source: C. Rasmussen

void **GaussianBlur**(const Mat& src, Mat& dst, Size ksize,
double sigmaX, double sigmaY=0,
int borderType=BORDER_DEFAULT)

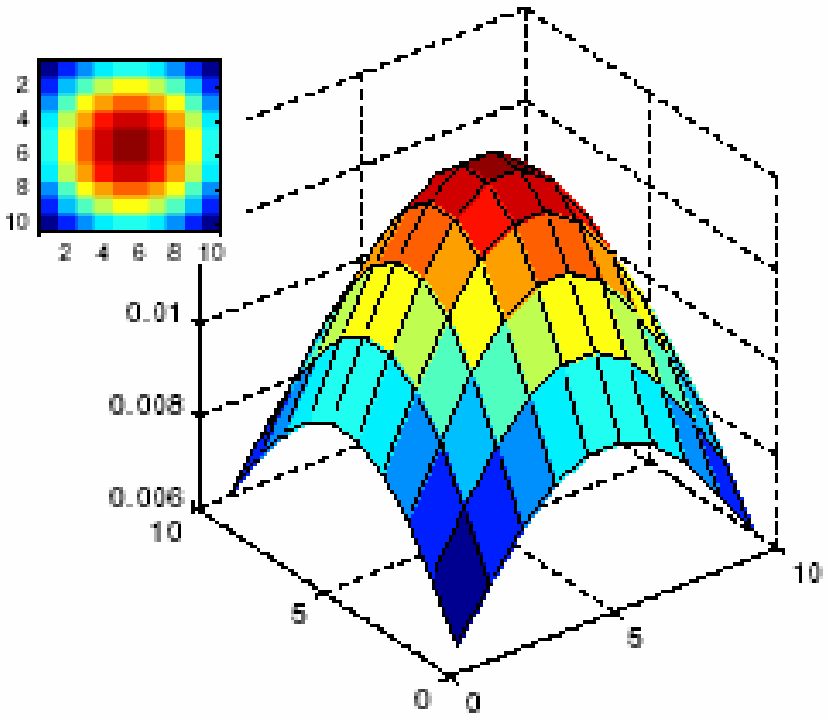
зависимость размера сглаживающей области от сигма



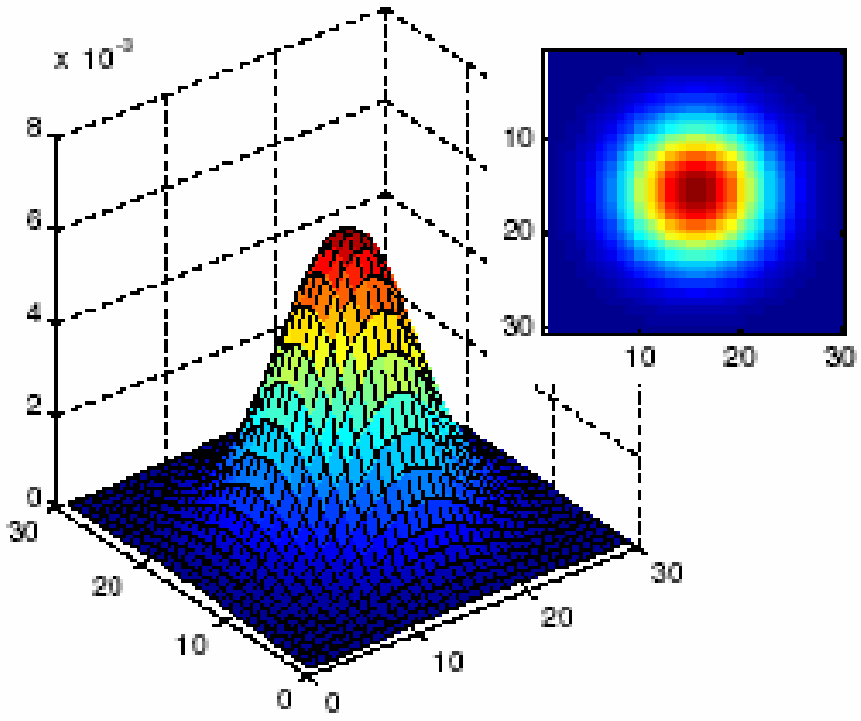


Slide by S. Lazebnik

Выбор размера ядра



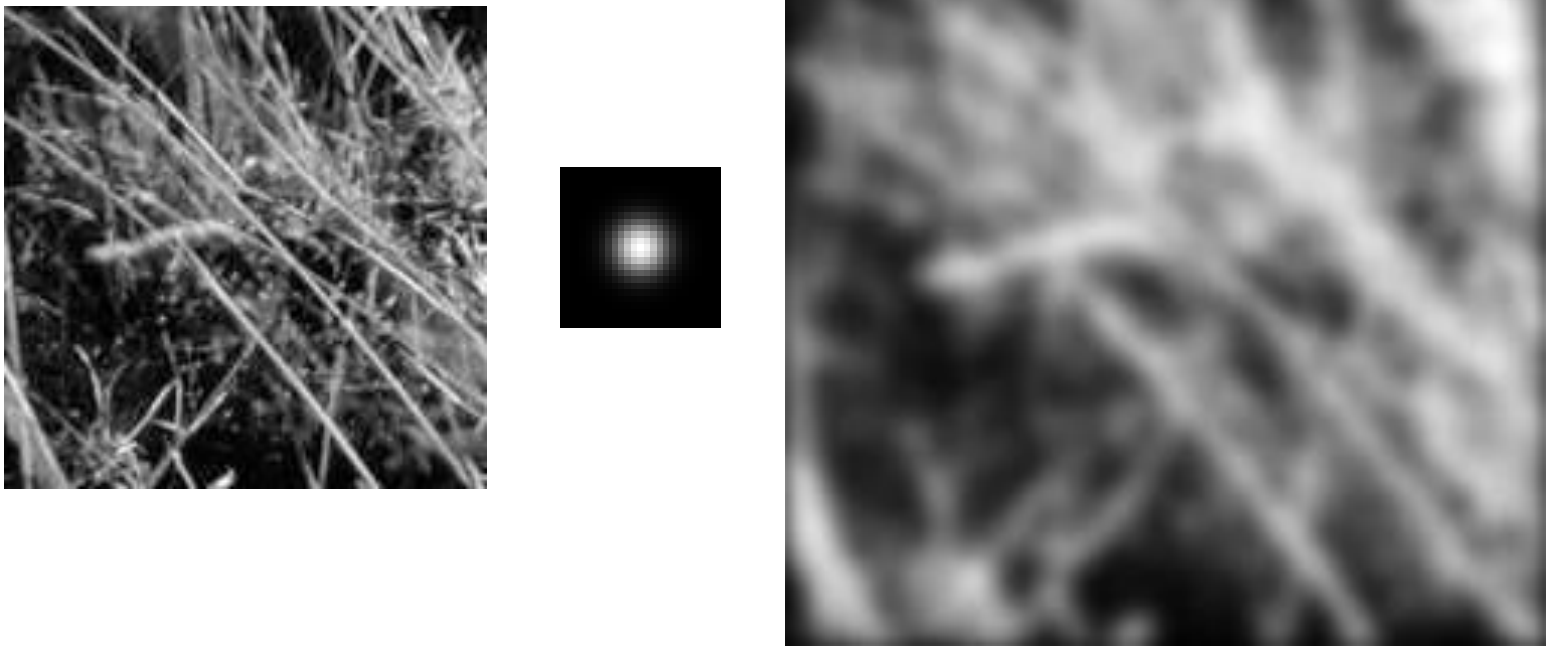
$\sigma = 5$ kernel 10x10



$\sigma = 5$ kernel 30x10

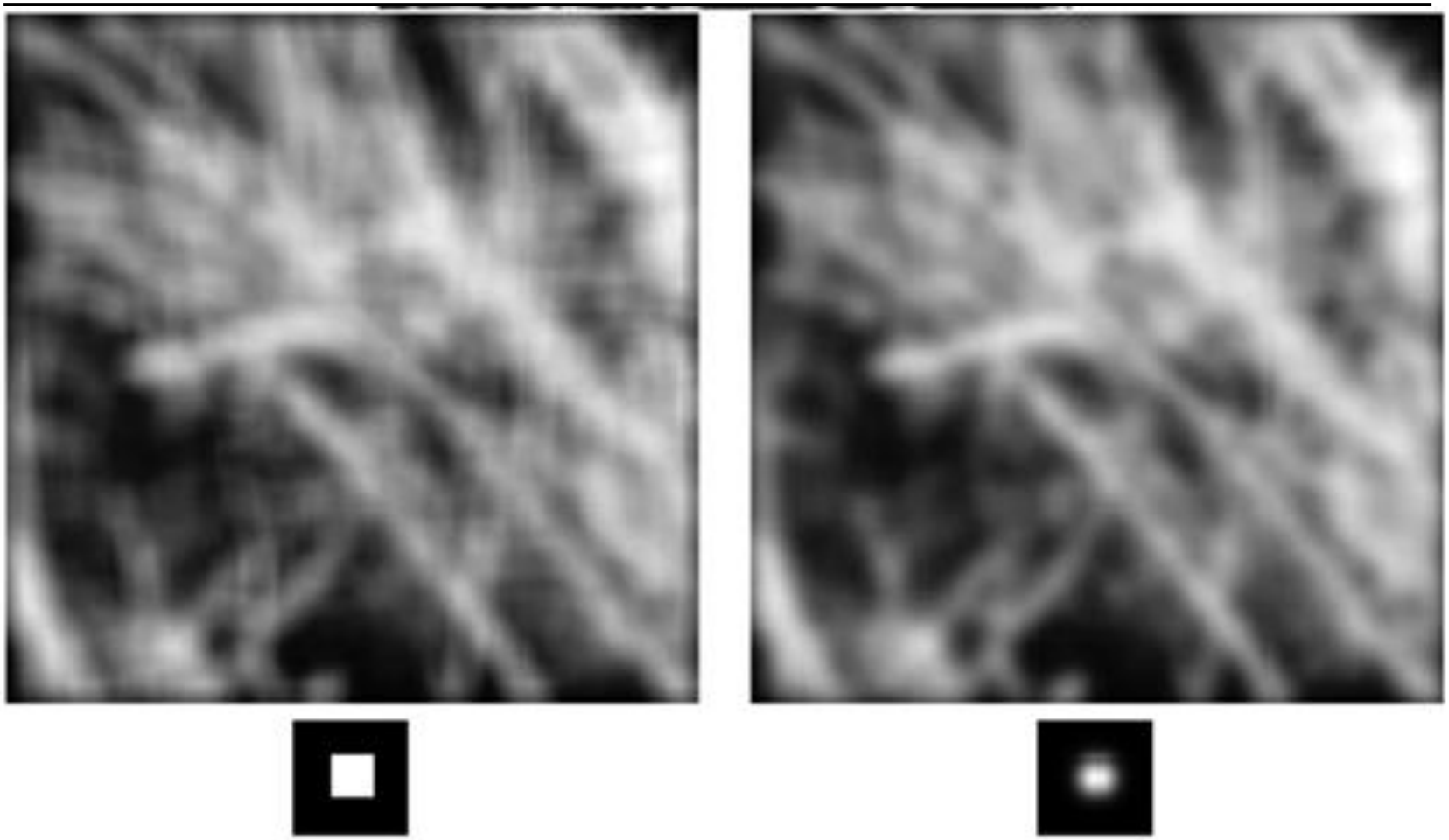
Source: K. Grauman

Сглаживание фильтром гаусса



Гауссовский фильтр — это **фильтр нижних частот**, он ослабляет высокочастотные сигналы

Сравнение box-фильтра и фильтра Гаусса



В цифровых камерах низкого класса, в том числе во многих камерах мобильных телефонов, обычно используется размытие по Гауссу, чтобы скрыть шум изображения, вызванный более высокой светочувствительностью ISO.

Виды шума

Original

Salt and pepper noise

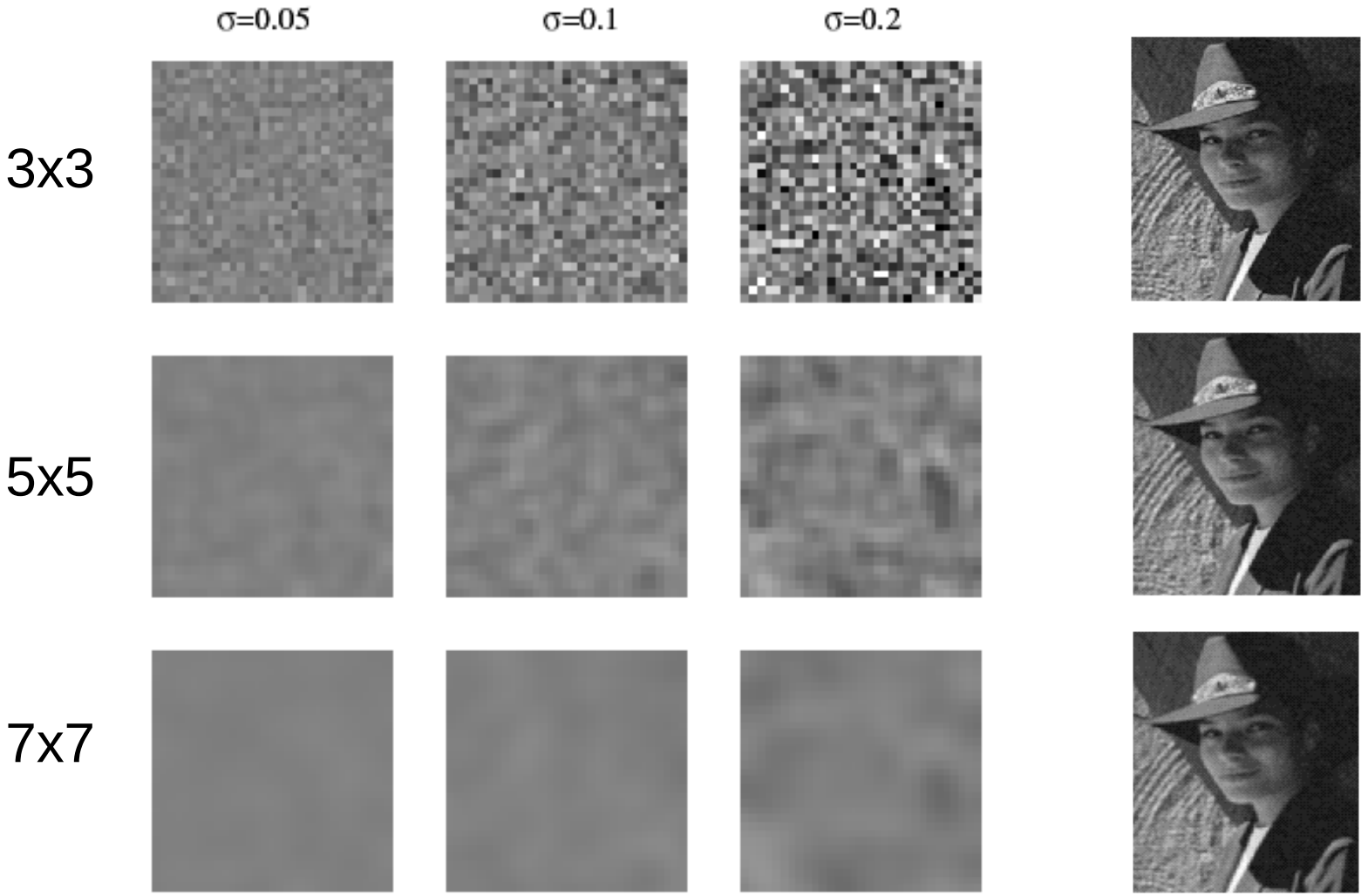
Impulse noise

Gaussian noise

- **Соль и перец**: случайные черные и белые пиксели
- **Импульсный:** случайные белые пиксели
- **Гауссов**: колебания яркости, распределенные по нормальному закону

Source: S. Seitz

Подавление гауссова шума



Сглаживание фильтрами большого радиуса подавляет шум, но размывает изображение

Slide by S. Lazebnik

Подавление шума «соль и перец»

3x3

5x5

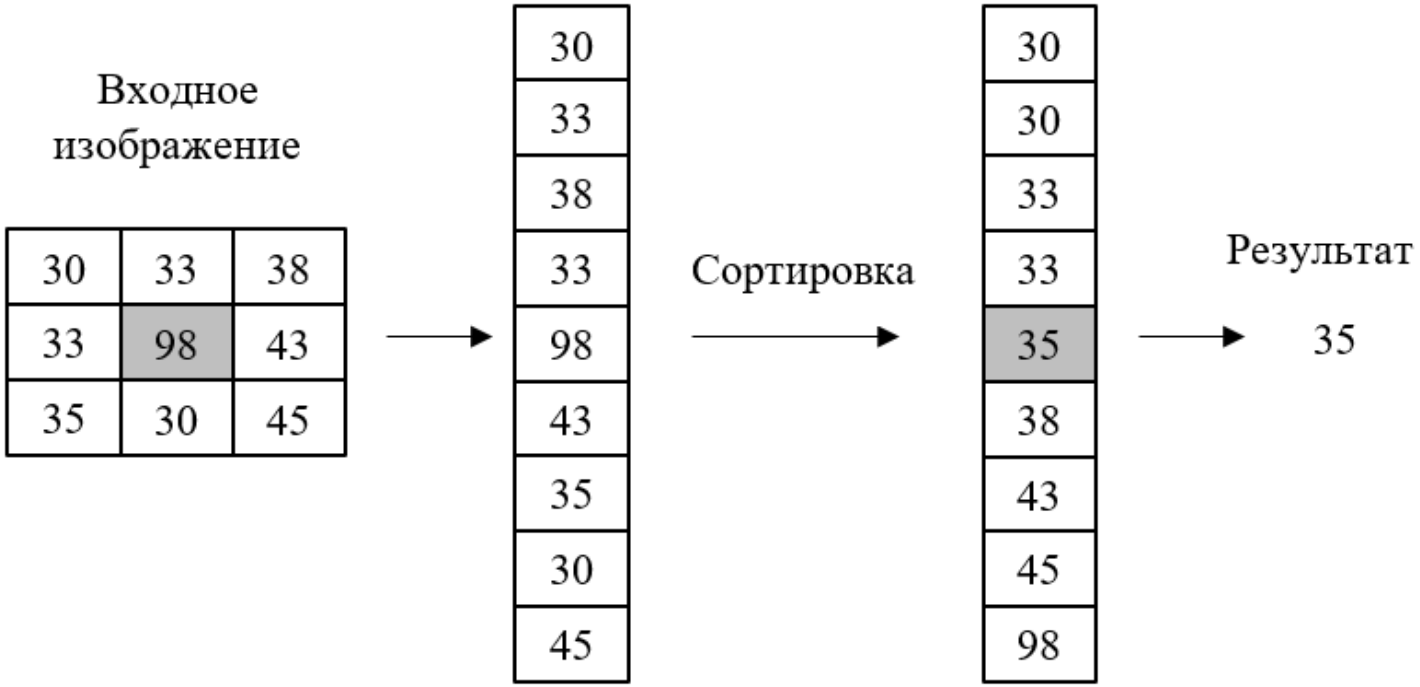
7x7

Чем плох результат?

Slide by S. Lazebnik

Медианный фильтр

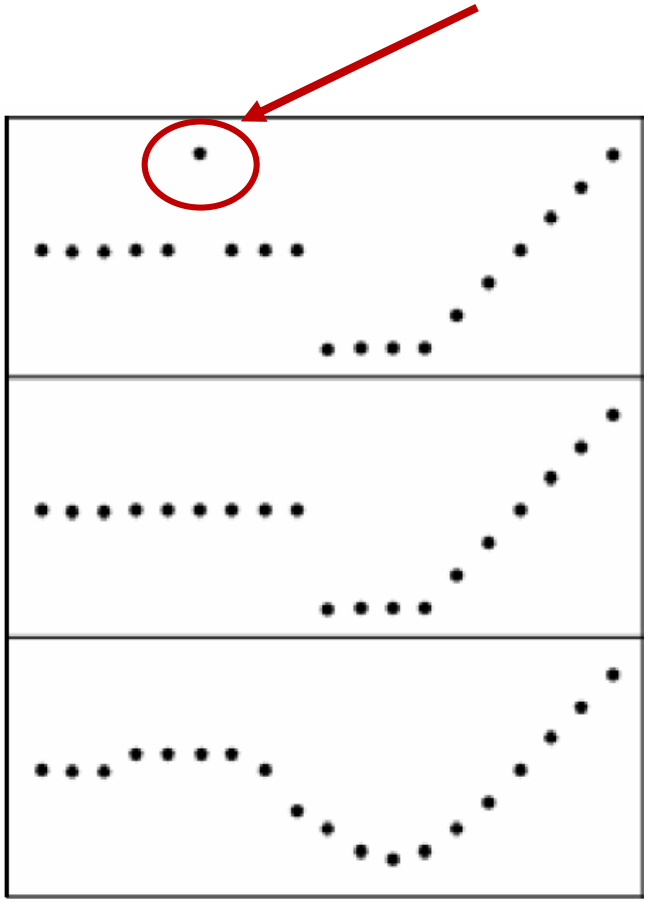
Фильтр работает с матрицами различного размера, но в отличие от матрицы свёртки, размер матрицы влияет только на количество рассматриваемых пикселей.
Пиксели, которые «попадают» в матрицу вокруг текущего пикселя, сортируются, и выбирается серединное значение из отсортированного массива. По сути, определяется медиана в отсортированном наборе данных. Это значение и является выходным для текущего пикселя.



Медианный фильтр

В чем преимущество медианного фильтра перед фильтром гаусса?

Устойчивость к выбросам (outliers)



входное

медианный

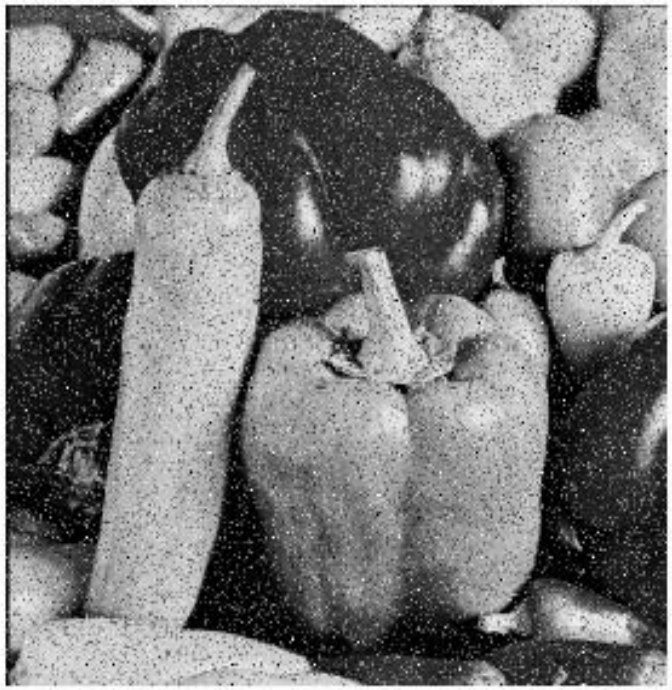
гаусса

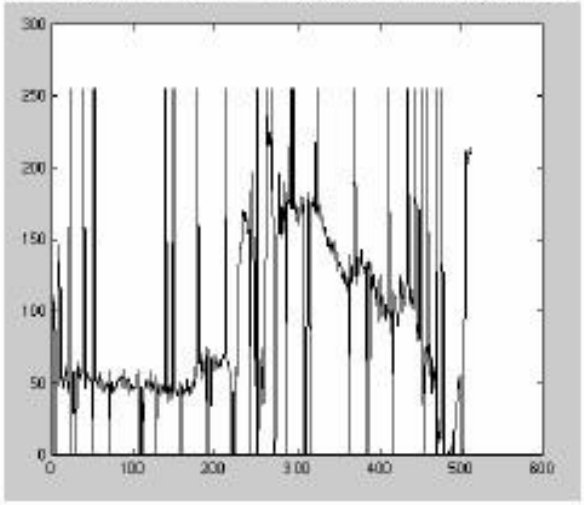
Преимущество медианной фильтрации перед фильтрами размытия (***blur***(), ***box***(), ***GaussianBlur***()) заключается в том, что «битый» пиксель на темном фоне будет заменен на темный, а не «размазан» по окрестности.

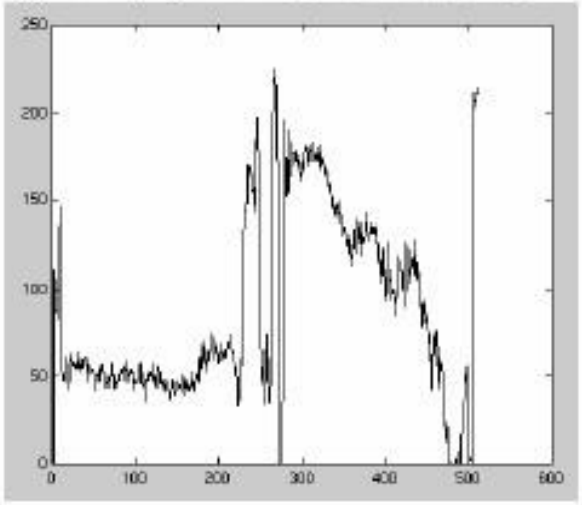
Source: K. Grauman

Salt-and-pepper noise

Median filtered



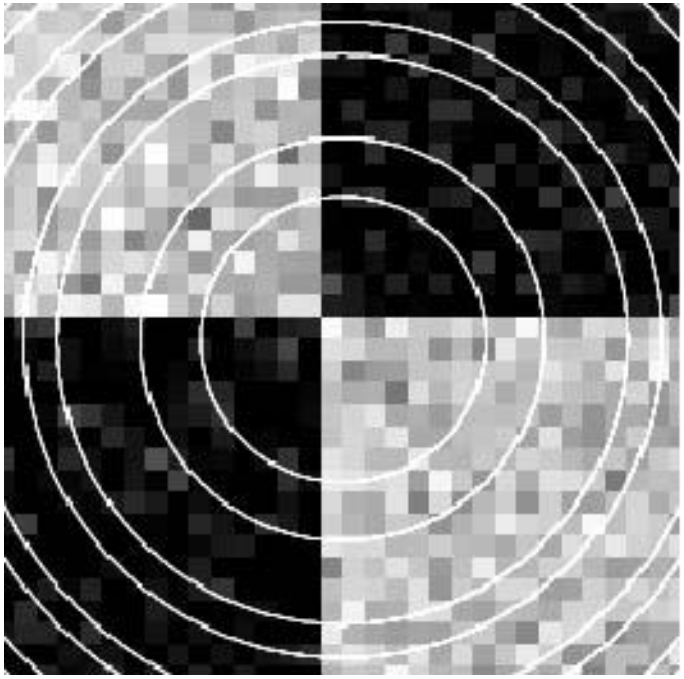


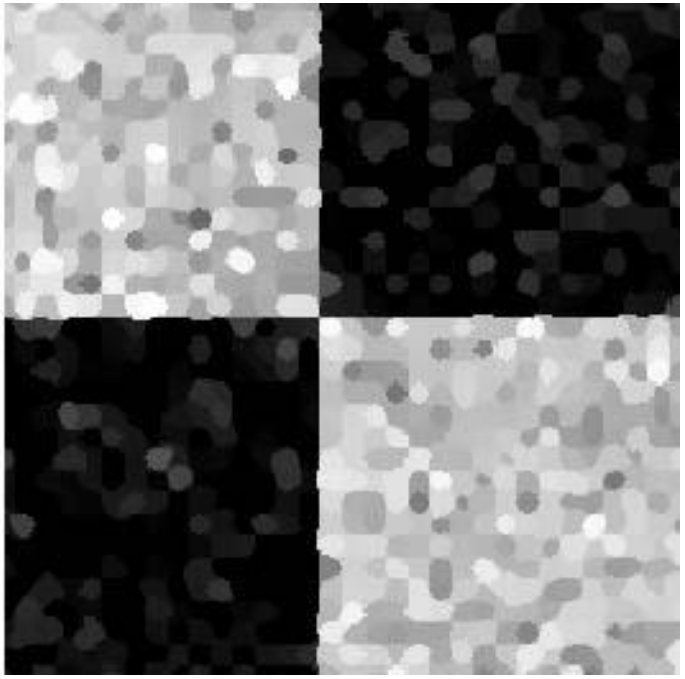


Source: M. Hebert

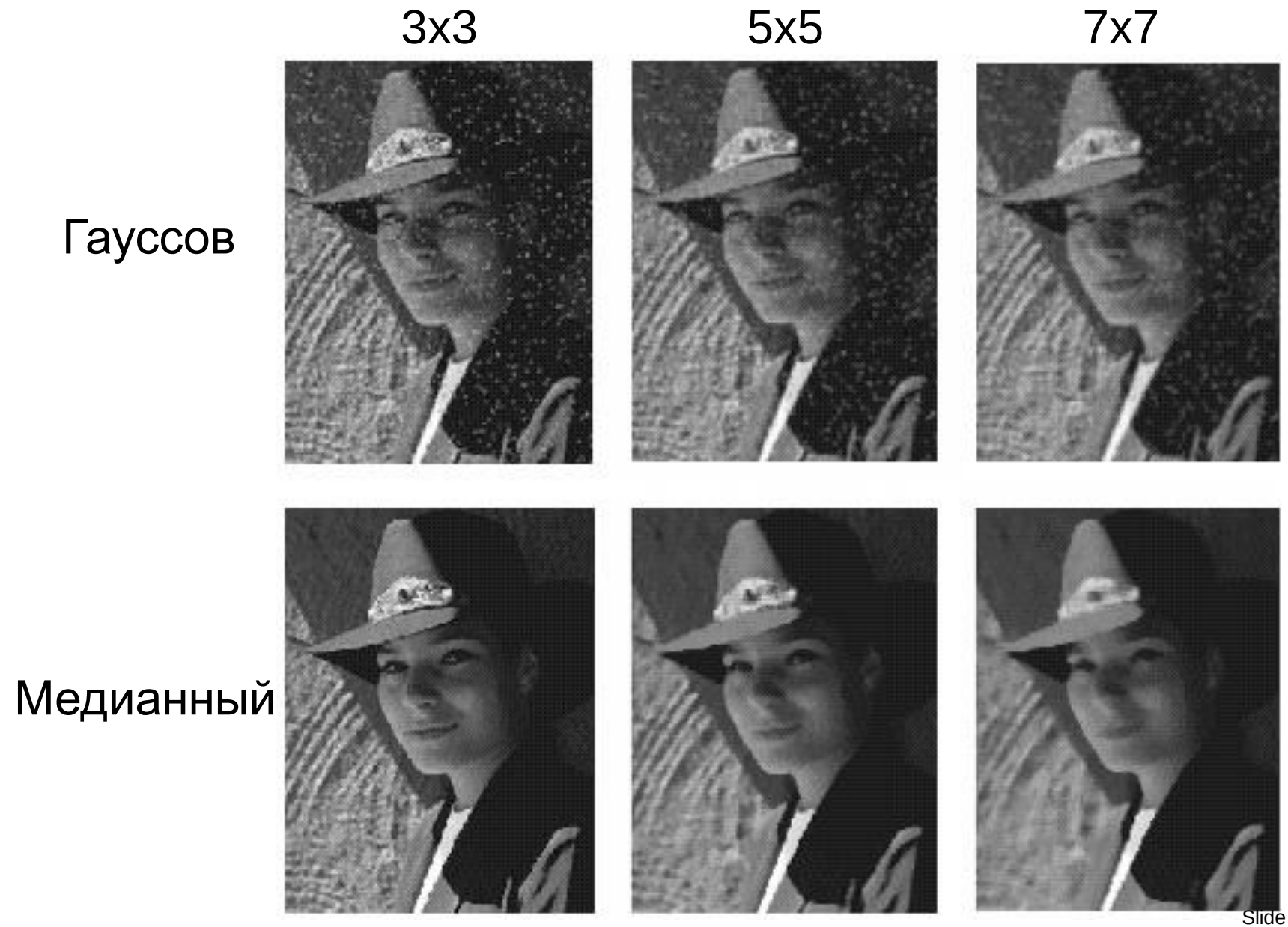
Медианный фильтр

Результат применения медианного фильтра к изображению с артефактами в виде тонких светлых линий.





Сравнение фильтров

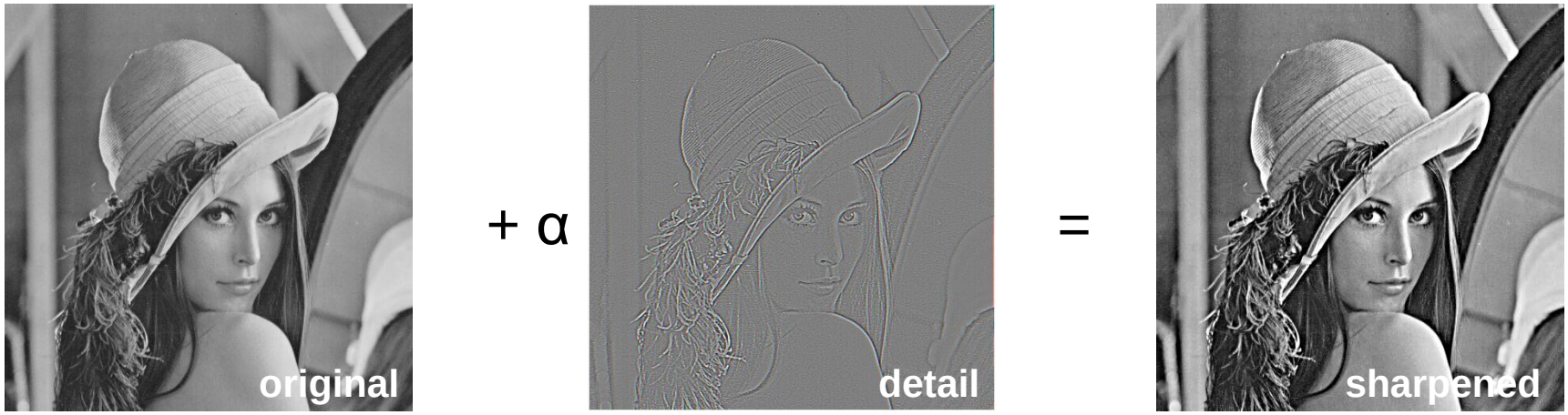


Slide by S. Lazebnik

Повышение резкости

Что теряется при сглаживании?

Добавим дополнительно высокие частоты:



Slide by S. Lazebnik

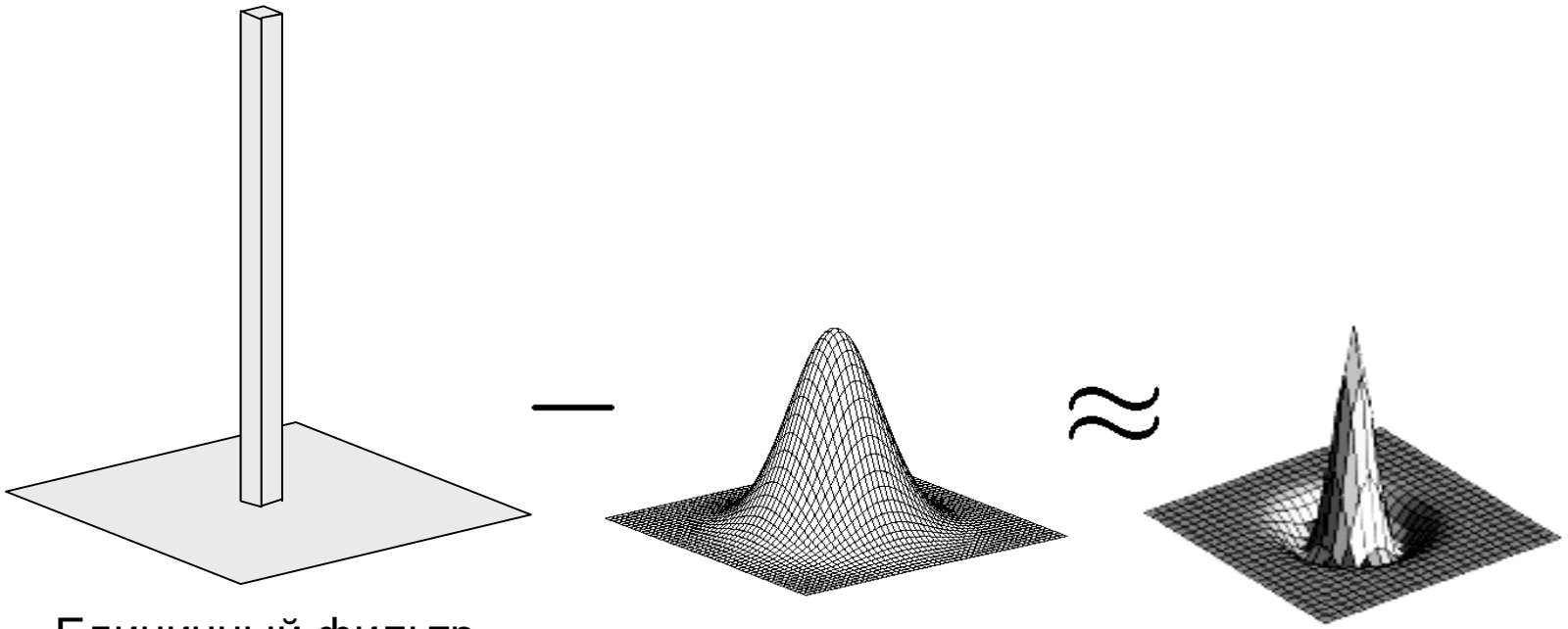
Повышение резкости (Фильтр Unsharp)

$$f + \alpha(f - f * g) = (1+\alpha) f - \alpha f * g = f * ((1+\alpha)e - \alpha g)$$

изображение

сглаженное изображение

Единичный Фильтр



Единичный фильтр

Гауссов

Лапласиан гауссиана

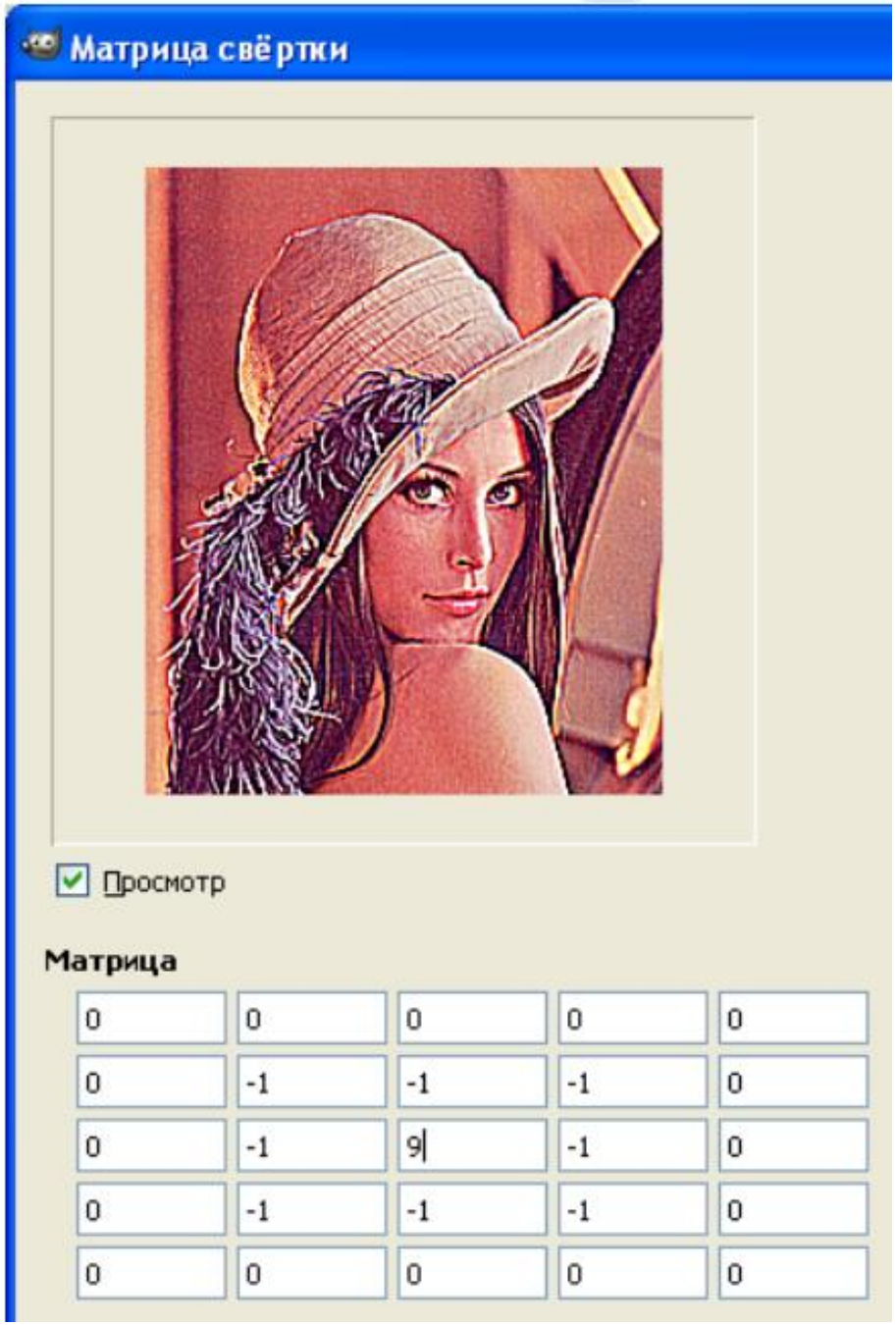
Slide by S. Lazebnik

Пример фильтра улучшения чёткости

Ядро свертки

$$\frac{1}{10}\cdot\begin{Vmatrix} -1 & -2 & -1 \\ -2 & 22 & -2 \\ -1 & -2 & -1 \end{Vmatrix}$$

В некоторых графических редаторвх (*в частности в программе GIMP*) есть фильтр «Матрица свёртки», который упрощает поиск необходимого матричного преобразования.



Тиснение

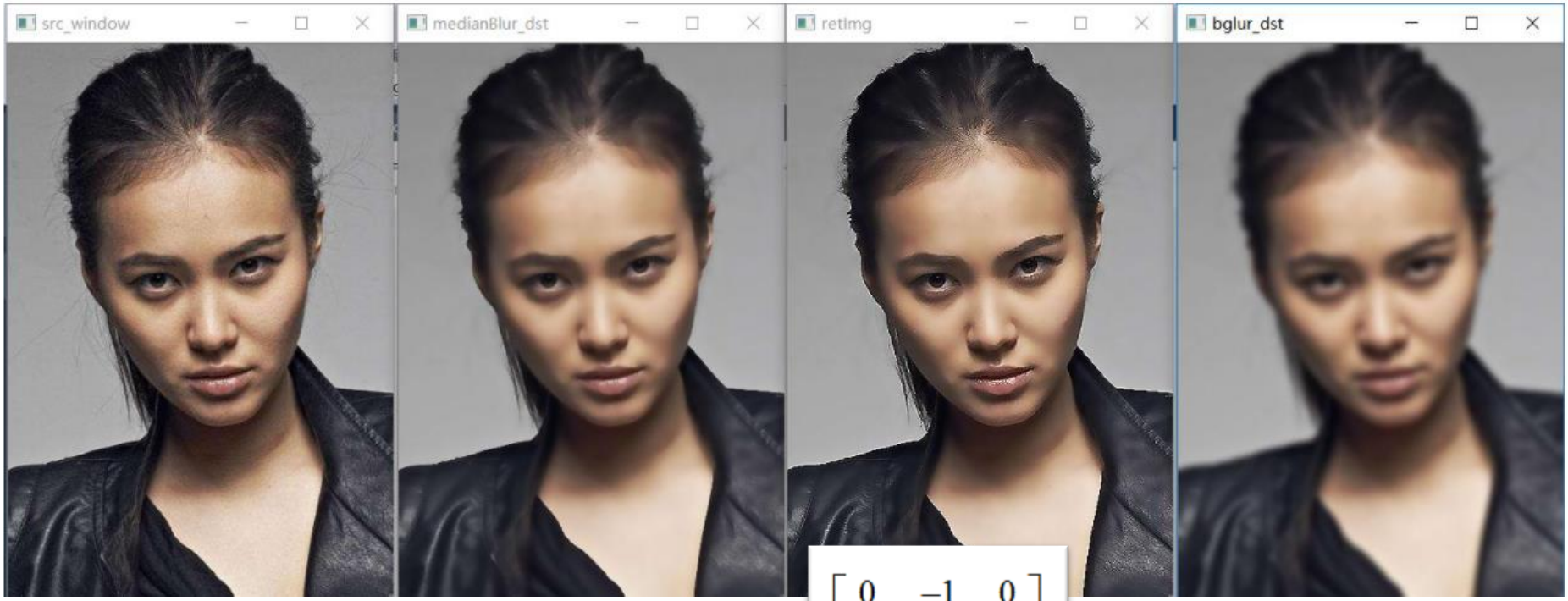
$$\begin{Vmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{Vmatrix}$$ либо

$$\begin{Vmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 0 \end{Vmatrix}$$

Выделение границ

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

Примеры применения фильтров

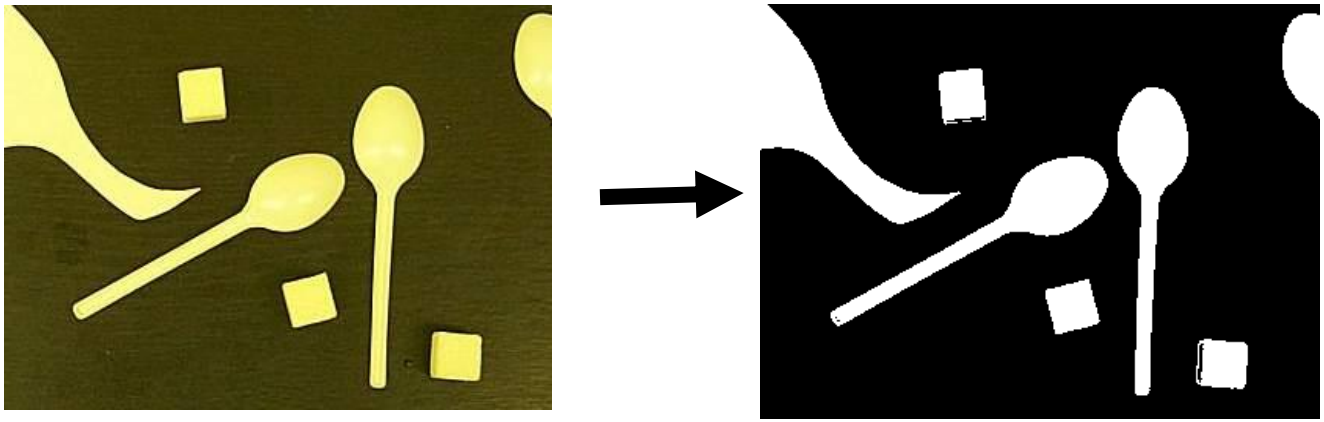


$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

1. Исходное изображение
2. **medianBlur**(src, dst, 3)
3.
```
Mat kernal = (Mat_<int>(3, 3) << 0, -1, 0, -1, 5, -1, 0, -1, 0);
        filter2D(dst, retImg, dst.depth(), kernal, Point(-1, -1), 0);
```
4.
```
GaussianBlur(src, gblur, Size(15, 15), 3, 3)
```

Свертка ядра и обработка изображений - Русские Блоги (russianblogs.com)

Бинаризация изображений

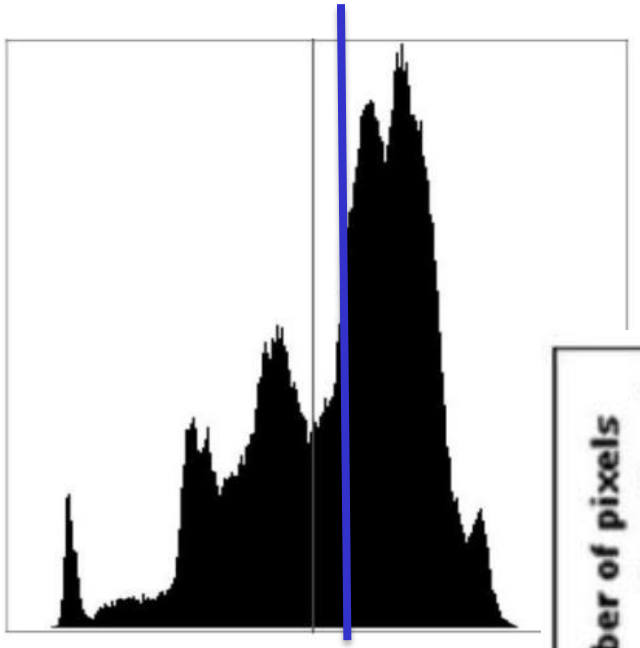


- Бинаризация – построение бинарного изображения по полутоновому / цветному
  - *пороговая фильтрация, которая сводится к выделению областей, яркость которых выше/ниже некоторого порога*
- Пиксель бинарного изображения может значения 0 и 1

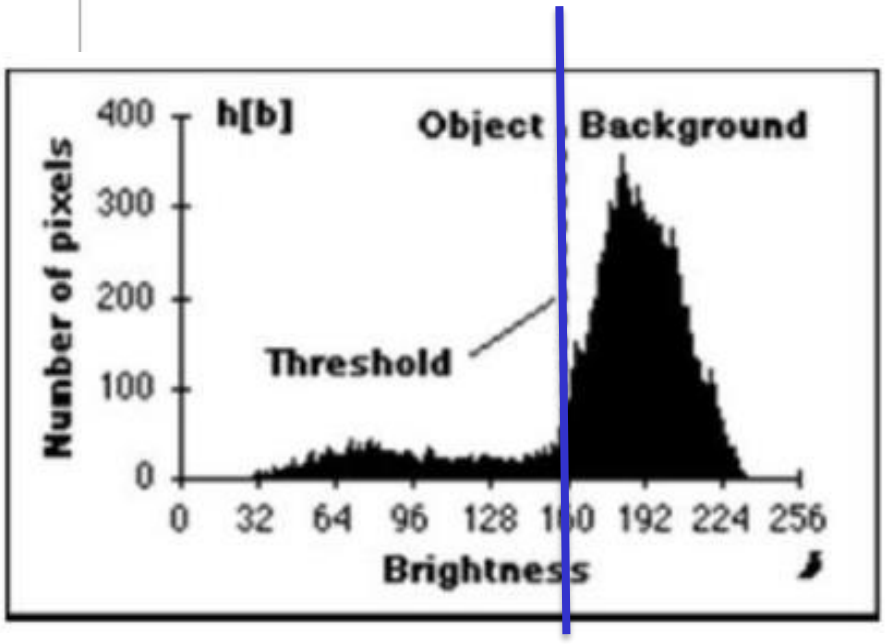
Смысл

- Разделить изображение на фон и контрастные объекты

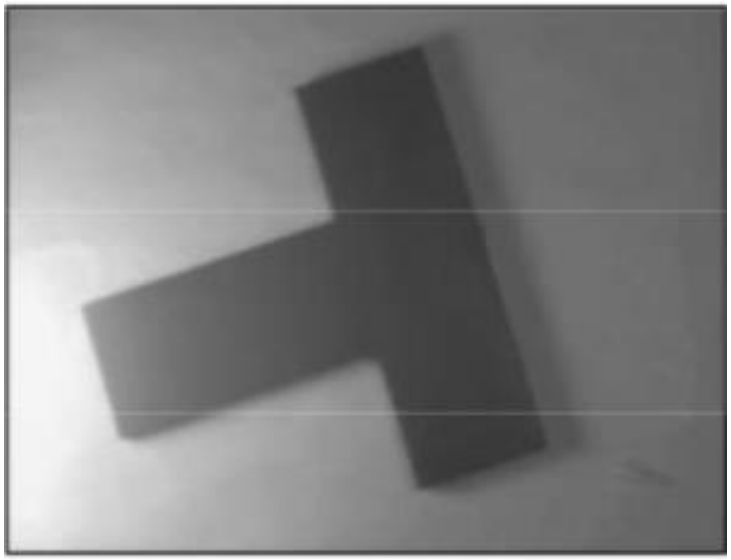
Бинаризация изображений

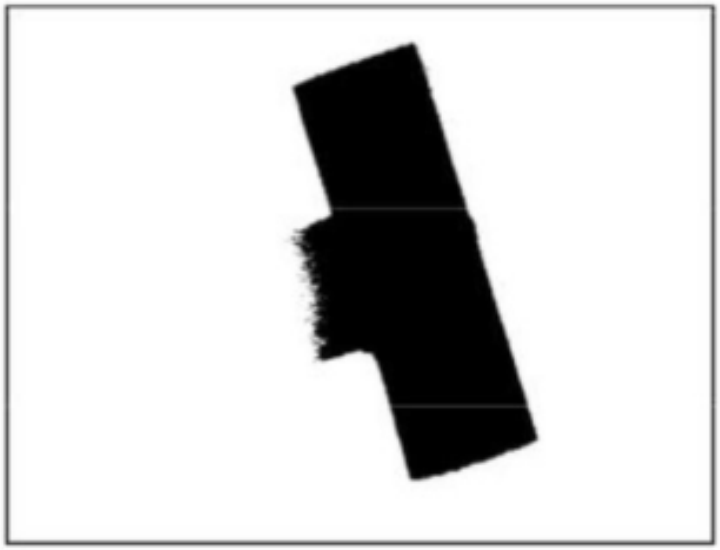


Пороговая бинаризация



Пороговая бинаризация





В данном случае простая пороговая бинаризация привела к неудовлетворительному результату

Адаптивная бинаризация

Необходима в случае неравномерной яркости фона/объекта.

- Для каждого пикселя изображения $I(x, y)$:
  1. В окрестности пикселя радиуса r высчитывается индивидуальный для данного пикселя порог T;
  2. Если $I(x, y) > T + C$, результат 1, иначе 0;

Варианты выбора T:

- $T = mean$
- $T = median$
- $T = (min + max) / 2$

Адаптивная бинаризация

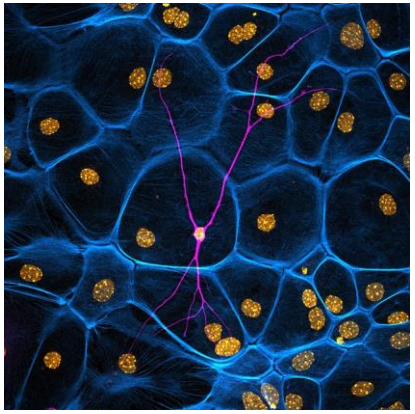
Исходное

r=7, C=0

r=7, C=7

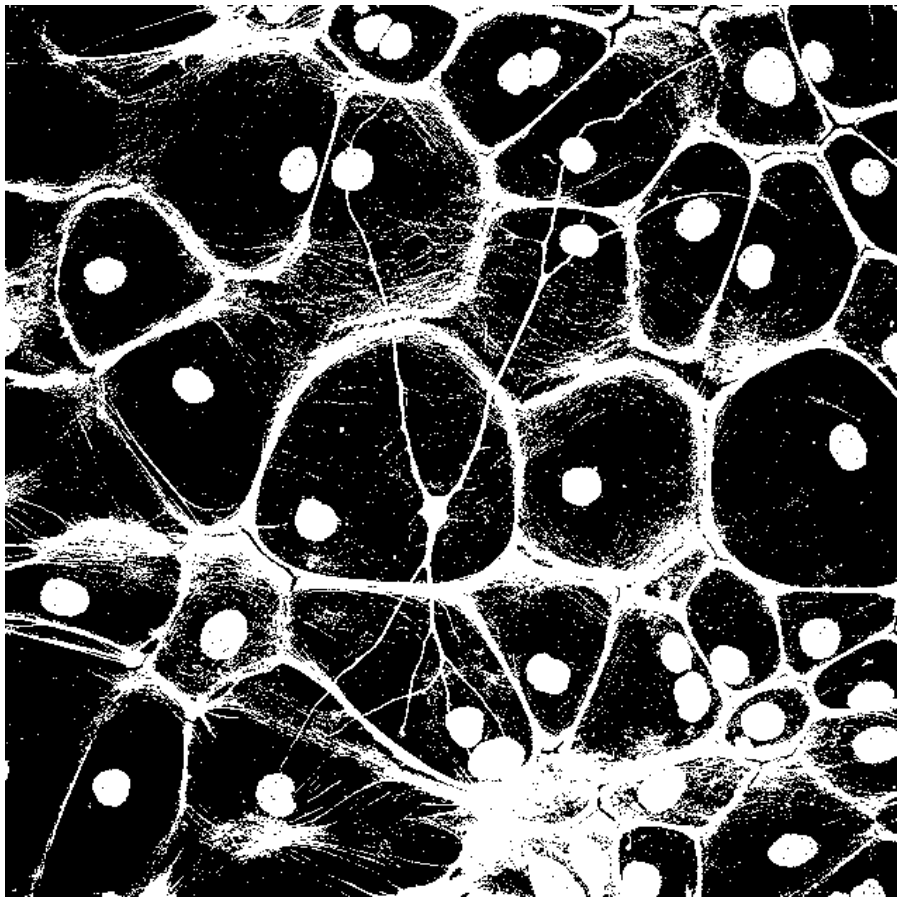
r=75, C=10

r – радиус окрестности пикселя,
C – производная константа

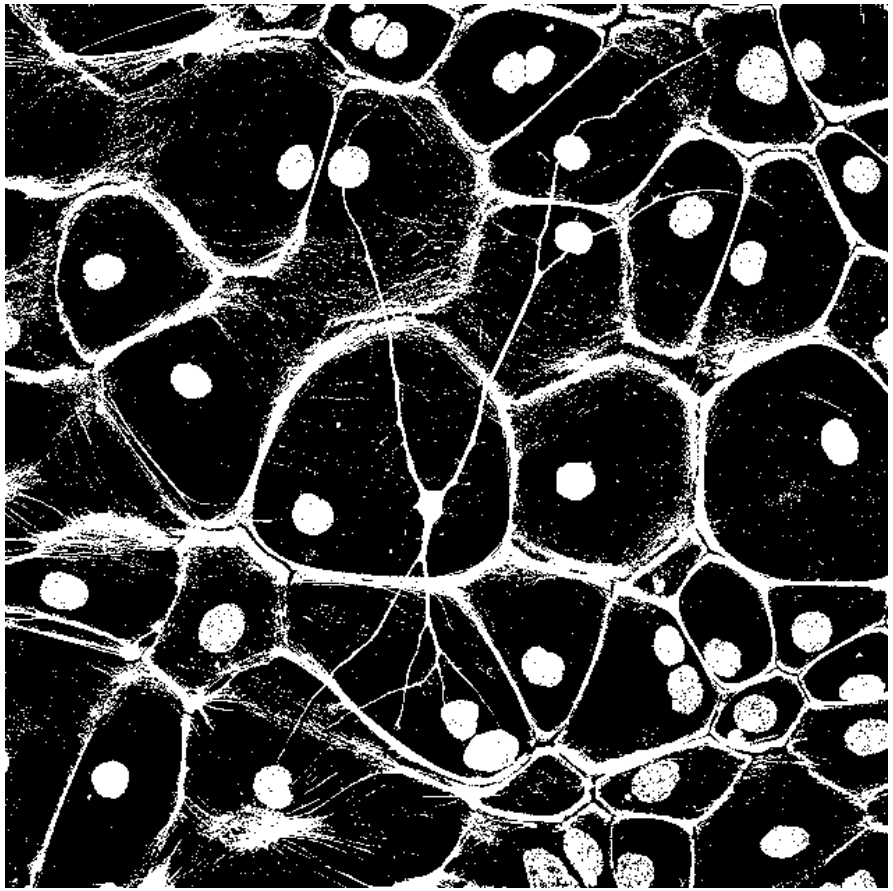


Лаба 2. Капорцев Олег

простая
бинаризация

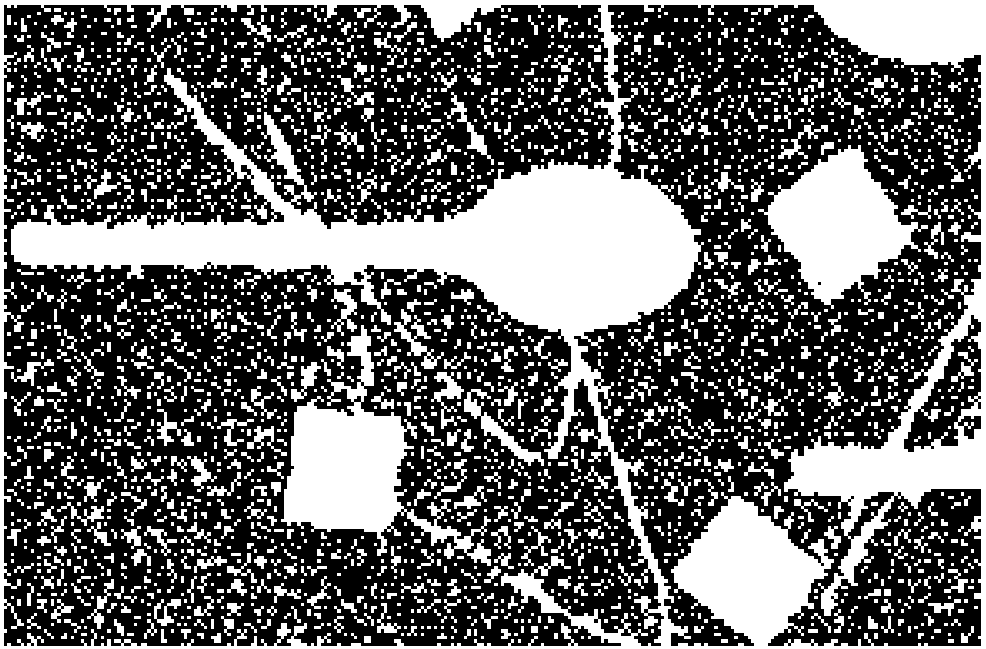


адаптивная
бинаризация



Шум в бинарных изображениях

Пример бинарного изображению с сильным шумом

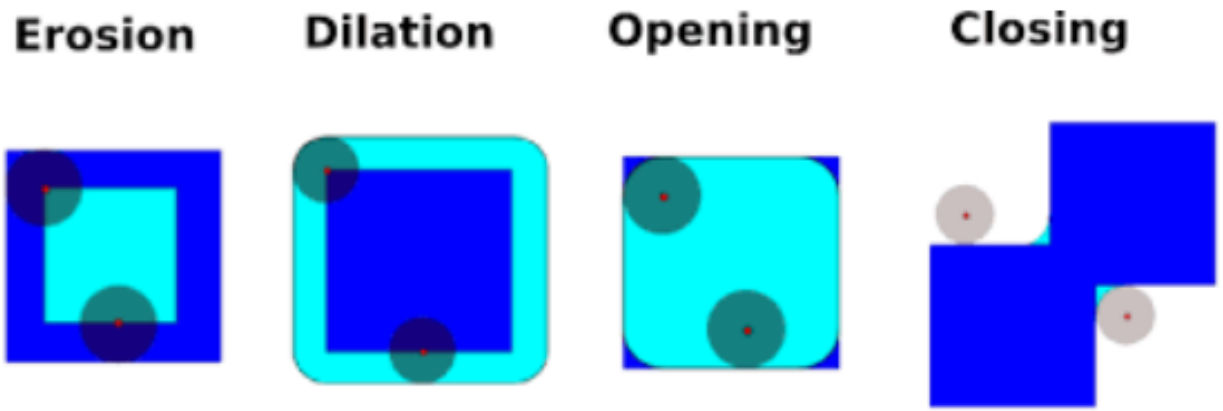


Часто возникает из-за невозможности полностью подавить шум в изображениях, недостаточной контрастности объектов и т.д.

Операции математической морфологии

Широко известный способ - устранение шума с помощью операций математической морфологии:

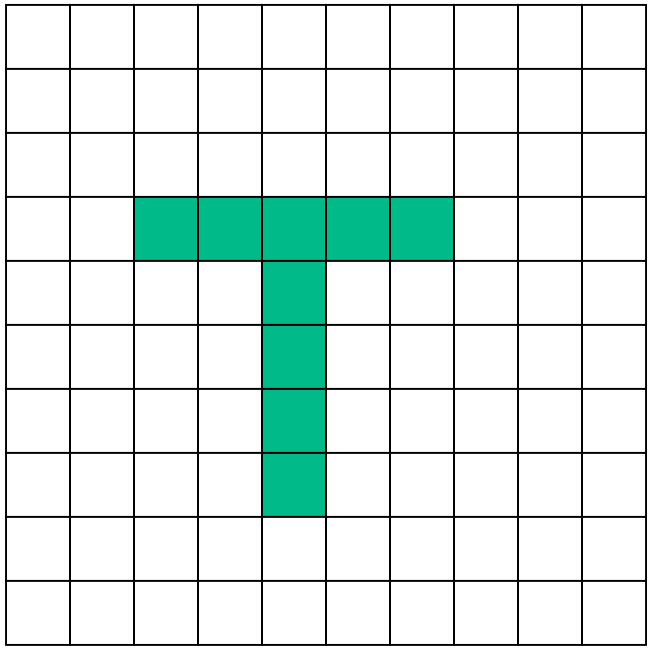
- Сужение (**erosion**) *увеличиваются тёмные области*
- Расширение (**dilation**) *увеличиваться светлые области*
- Закрытие (closing)
- Раскрытие (opening)



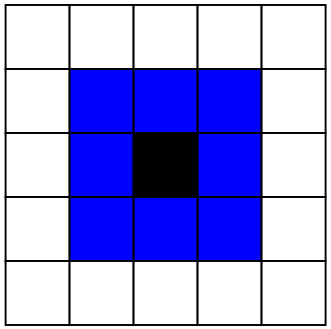
Математическая морфология – это теория и техника анализа и обработки геометрических структур

Математическая морфология

A

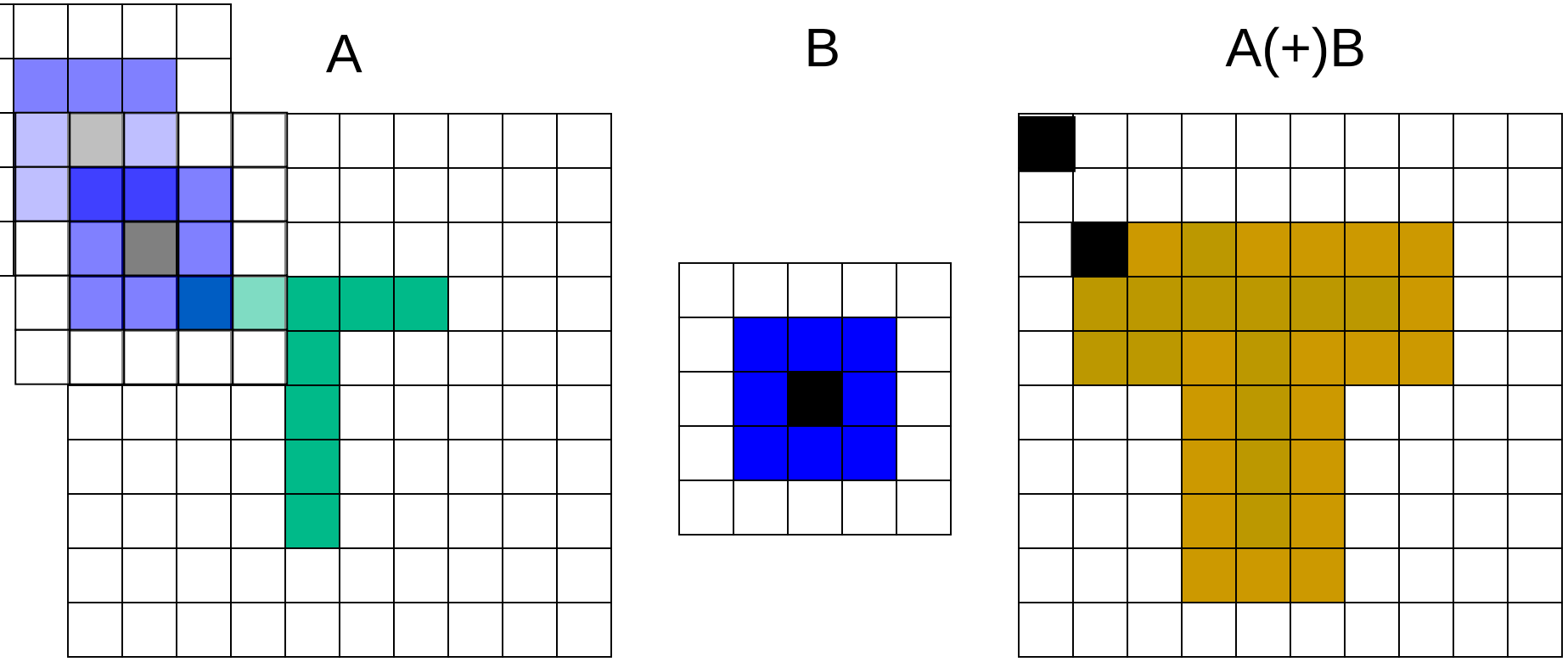


B

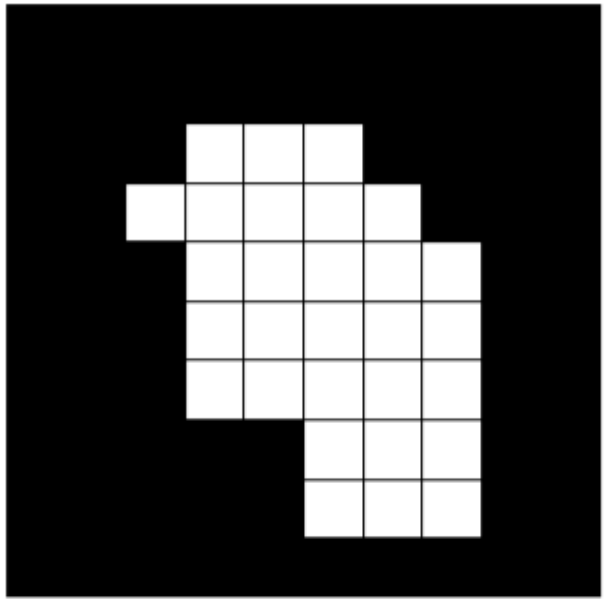


Множество A обычно является объектом обработки, а множество B (называемое структурным элементом) – аналог фильтра.

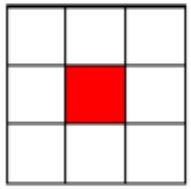
Расширение (dilation) в дискретном случае



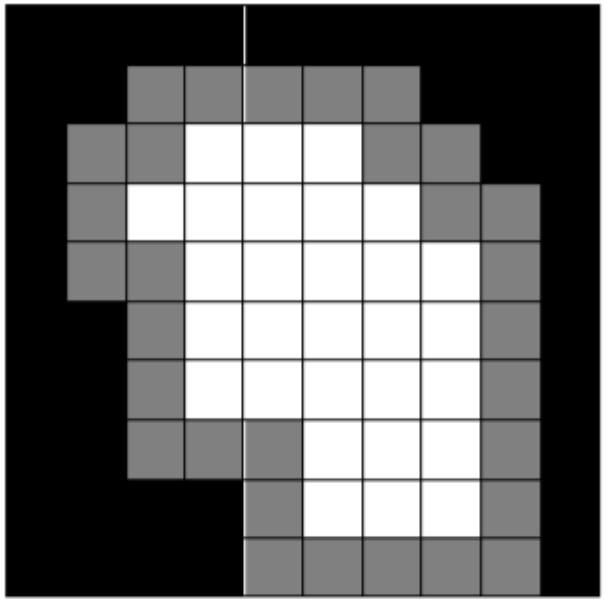
Операция «расширение» - аналог логического «или»



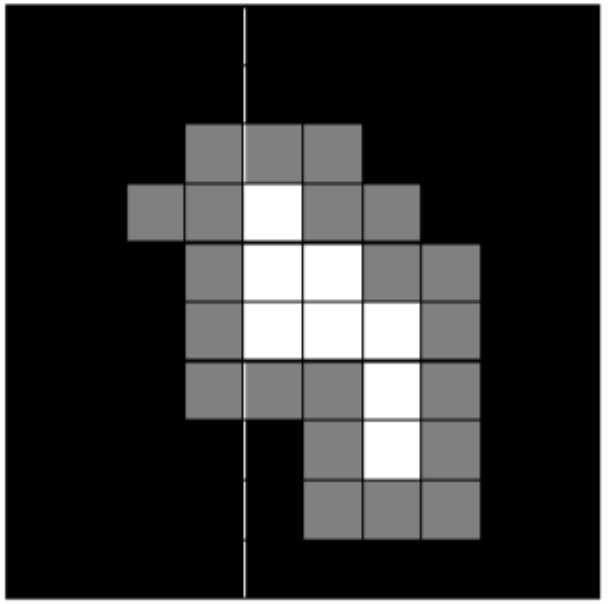
a) исходное изображение



b) шаблон (центр – ведущий элемент)



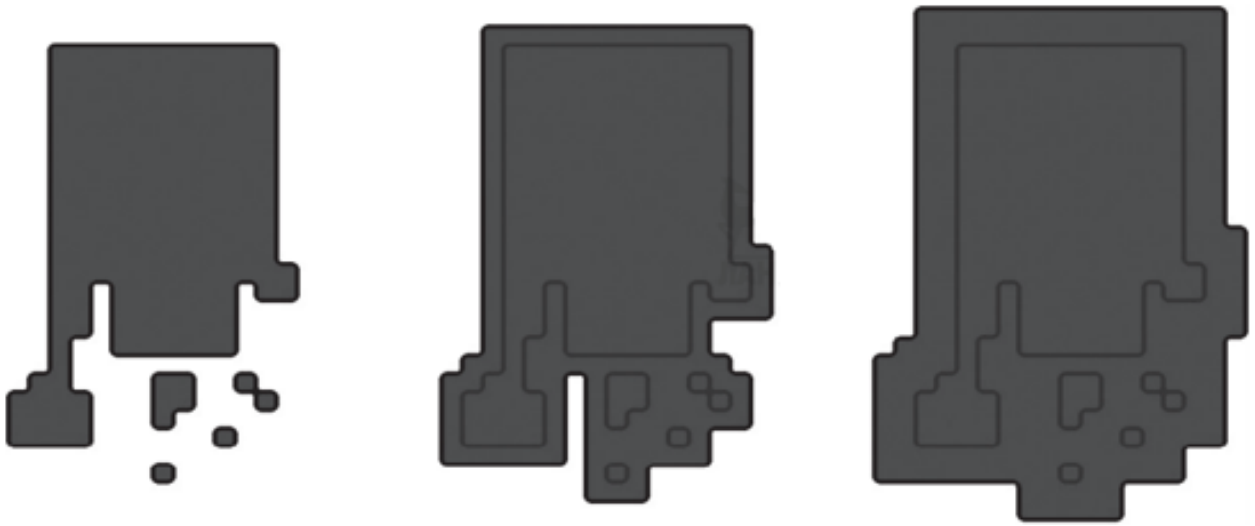
c) результат дилатации



d) результат эрозии

Дилатация

Растягивание (расширение), по идее, должно устранять шум и **способствовать объединению областей изображения, которые были разделены шумом, тенями, etc**.
Применение небольшого растягивания должно сплавить эти области в одну.



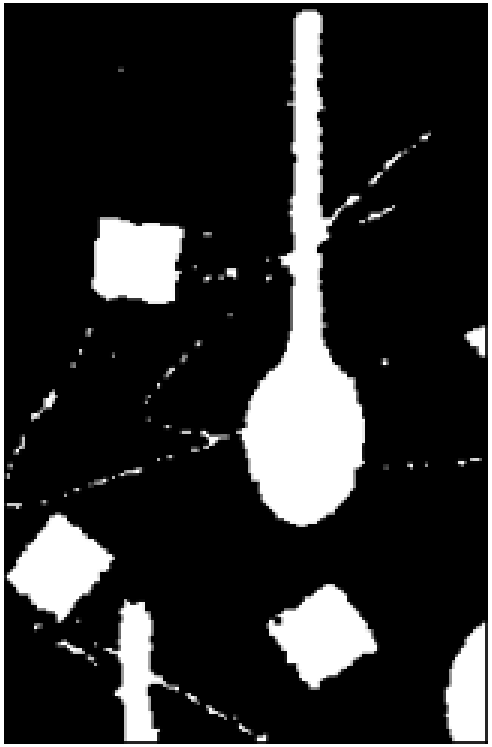
Результат применения дилатации: а – оригинал; б – дилатация; в – после второго применения

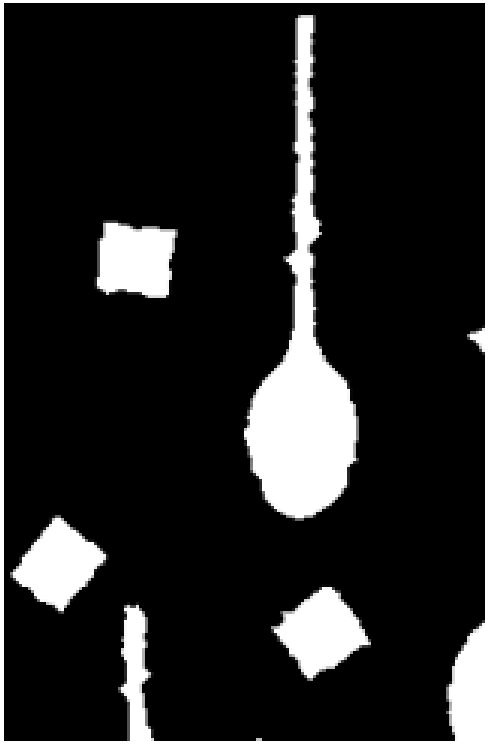
Эрозия

- **Эрозия** (сужение) изображения обычно используется для избавления от случайных вкраплений на изображении. **Идея состоит в том, что вкрапления при сужении устранятся, тогда как крупные и соответсвенно более визуально-значимые регионы остаются**.

Результат применения эрозии: а – оригинал; б – эрозия; в – после второго применения

Эрозия





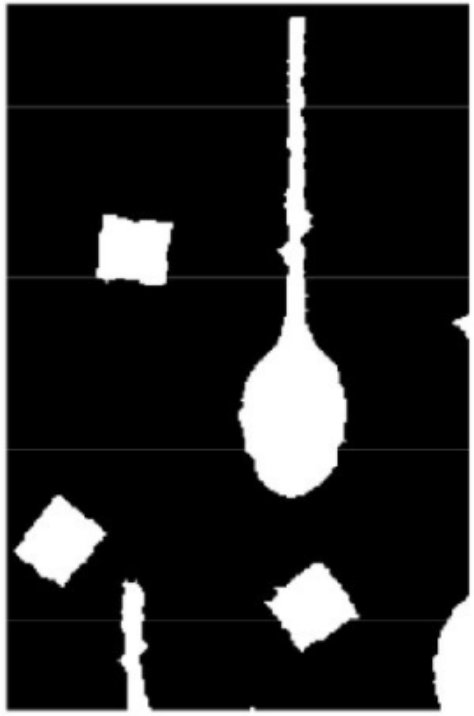
Операции раскрытия и закрытия

- Морфологическое раскрытие (opening)

$$open(A,B)=(A(-)B)(+)B$$

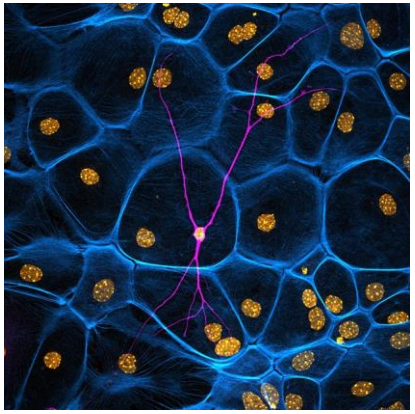
A - объект обработки, B - структурный элемент

Сильный шум



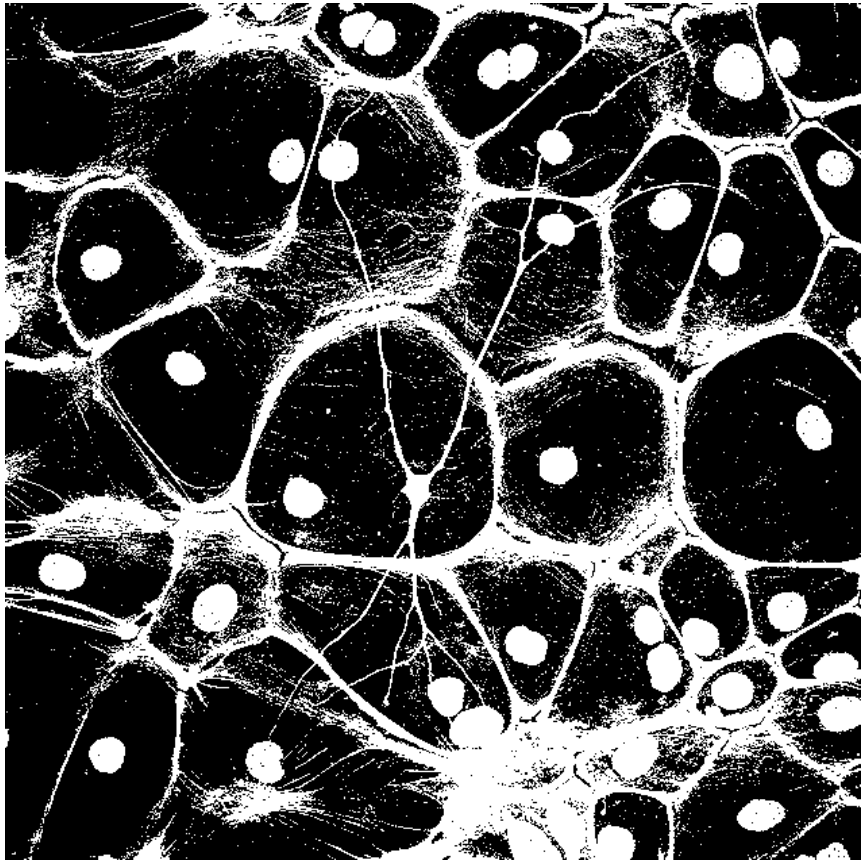
Сужение

Раскрытие

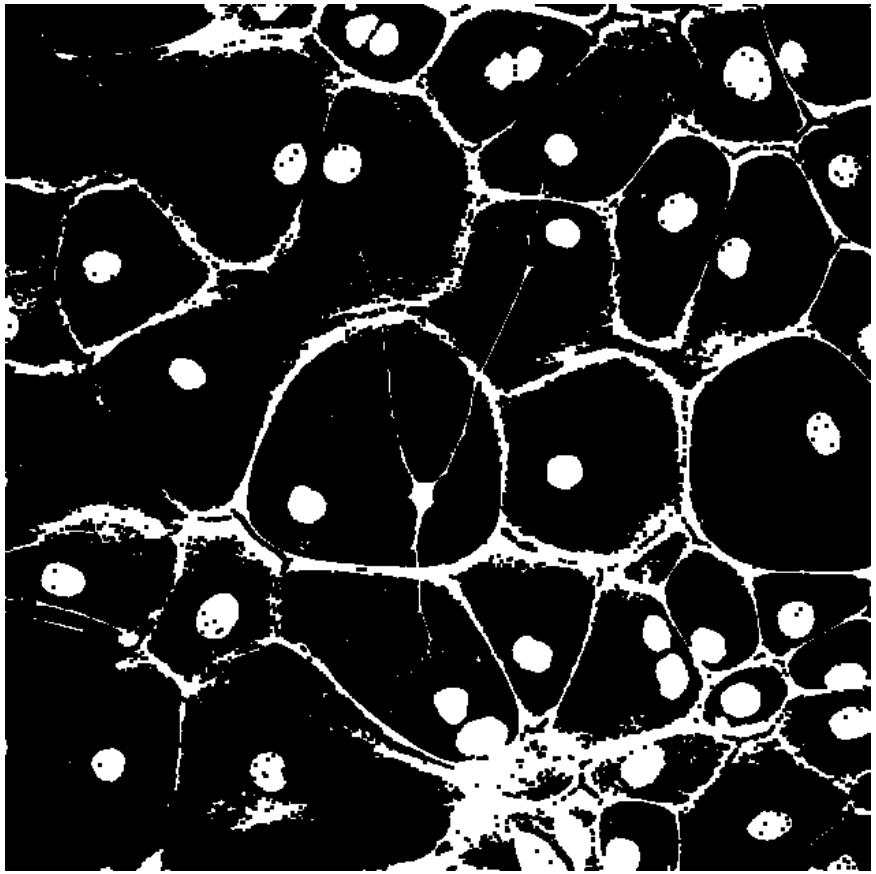


Лаба 2. Капорцев Олег

простая
бинаризация



сужение (erode)
позволило убрать шум



Операции раскрытия и закрытия

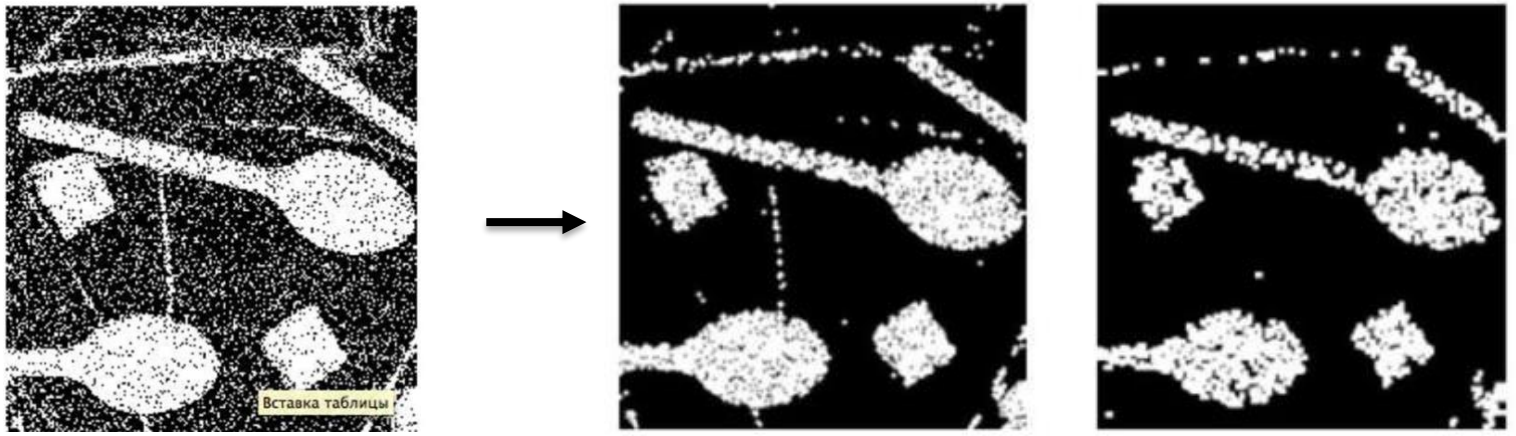
- Применим операцию закрытия к изображению с дефектами объектов:

- Морфологическое закрытие (closing)

$$close(A, B) = (A \oplus B) \ominus B$$

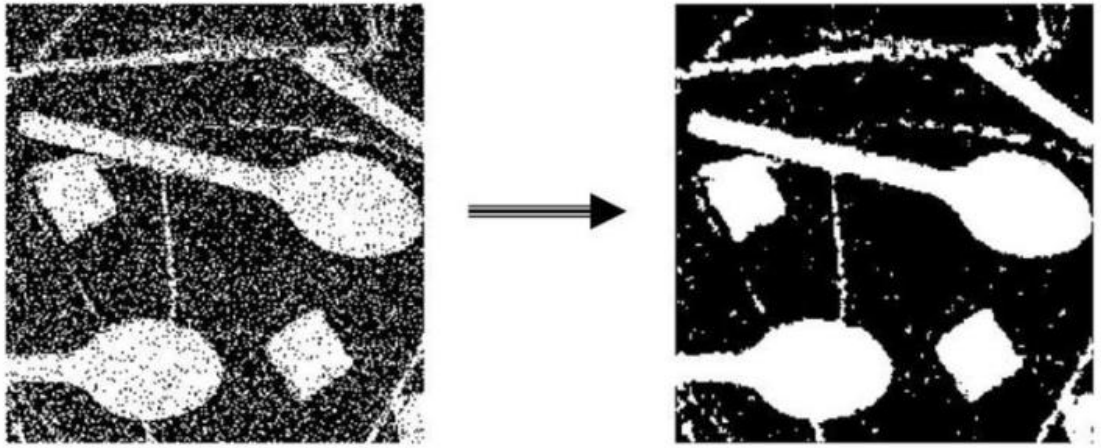
Не лучший пример для морфологии

В данном примере применение морфологических операций приведет к таким результатам



Что можно сделать?

До морфологических операций применить медианный фильтр



В *OpenCV* эти преобразования реализуются функциями ***erode***() и ***dilate***():

```
void erode(
  const Mat& src,                  // входное изображение
  const Mat& dst,                  // выходное изображение
  const Mat& element,              // ядро, объект типа Mat()
  Point anchor = Point(-1,-1),     // позиция якоря
  int iterations = 1,              // сколько раз применять
  int borderType = BORDER_CONSTANT,   // экстраполяция границы
  const Scalar& borderValue = morphologyDefaultBorderValue()
  );
```

Запишем фрагмент кода, применяющий операции эрозии и дилатации к изображению *img*:

```
element = Mat();                        // ядро по умолчанию
erode(img, erodeImg, element);          // вычисление эрозии
dilate(img, dilateImg, element);        // вычисление дилатации
```

Операции размыкания и замыкания можно реализовать с помощью многоцелевой функции ***morphologyEx***(), параметры которой аналогичны параметрам функций ***erode***() и ***dilate***()

```
void morphologyEx(
    const Mat& src,                 // входное изображение
    const Mat& dst,                 // выходное изображение
    int op,                         // оператор
    const Mat& element,             // ядро, объект типа Mat()
    Point anchor = Point(-1,-1),       // позиция якоря
    int iterations = 1,                // сколько раз применять
    int borderType = BORDER_CONSTANT,  // экстраполяция границы
    const Scalar& borderValue = morphologyDefaultBorderValue()
    );
```

Оператор *op* определяет выбранную операцию: *MOP_OPEN* – размыкание; *MOP_CLOSE* – замыкание; *MOP_GRADIENT* – морфологический градиент; *MOP_TOPHAT* – «Верх шляпы»; *MOP_BLACKHAT* – «Черная шляпа».

Пример использования процедуры дилатации для устранения разрывов

0	1	0
1	1	1
0	1	0

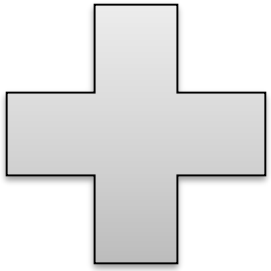
B

До дилатации

После дилатации

0	1	0
1	1	1
0	1	0

B



- Эрозия: $dst(x,y)=\min_{((x',y') \text{ in } [-3..3])} src(x+x',y+y')$

- Наращивание: $dst(x,y)=\max_{((x',y') \text{ in } [-3..3])} src(x+x',y+y')$

- Размыкание: $dst=open(src)=dilate(erode(src))$

- Замыкание: $dst=close(src)=erode(dilate(src))$