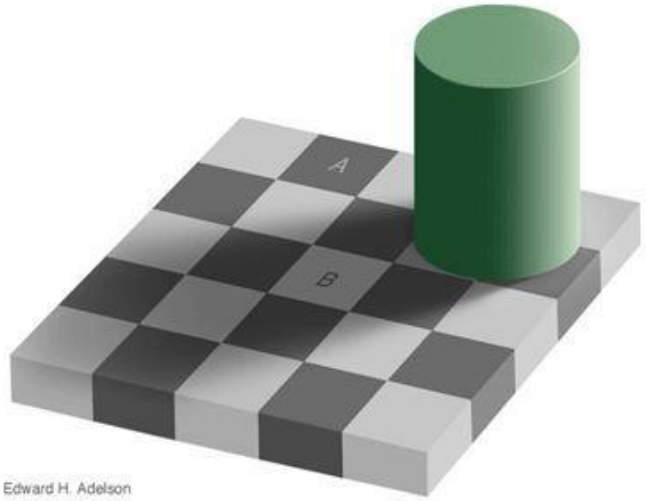
Обработка изобажений

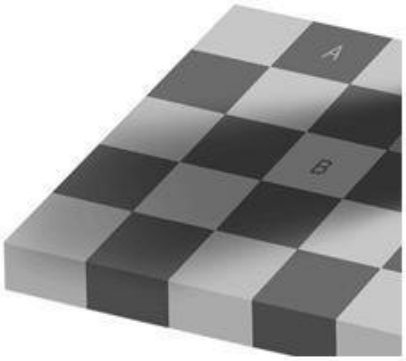
Причины обработки

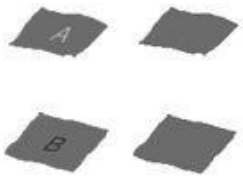
- улучшение восприятия человеком
  - субъективное улучшение
- улучшение “восприятия” компьютером
  - упрощение распознавания
- преобразование для технических нужд
  - изменение разрешения
  - изменение пропорций
  - корректировка яркости, контраста, оттенка и т.п.
- спецэффекты
  - художественные преобразования

Яркость

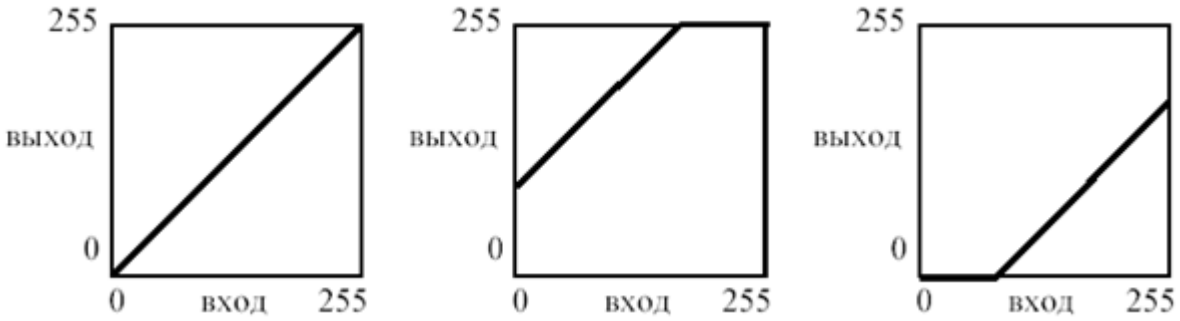


Edward H. Adelson

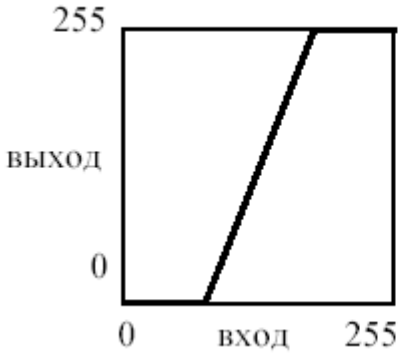


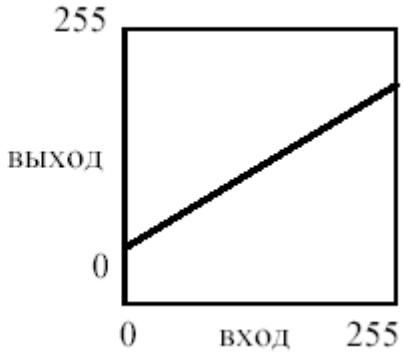


Яркость

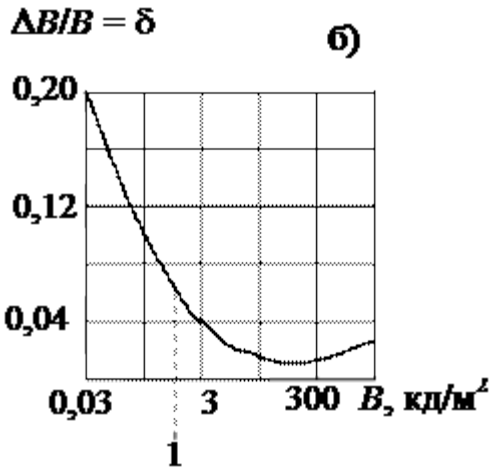

255
ВЫХОД
0
0
ВХОД
255
255
ВЫХОД
0
0
ВХОД
255
255
ВЫХОД
0
0
ВХОД
255

Контраст





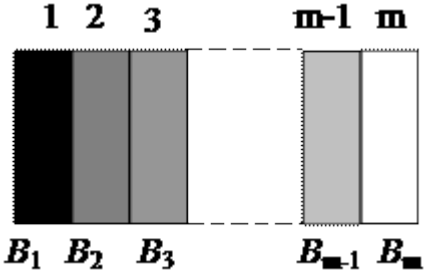
Градации яркости



Закон Фехнера

$$\Delta B/B = \delta \approx const$$

где ΔB – приращение яркости, едва заметное на глаз; B – начальная яркость; δ - контрастная чувствительность глаза.



$$\beta = B_{макс}/B_{мин} = (1+\delta)^{m-1}$$

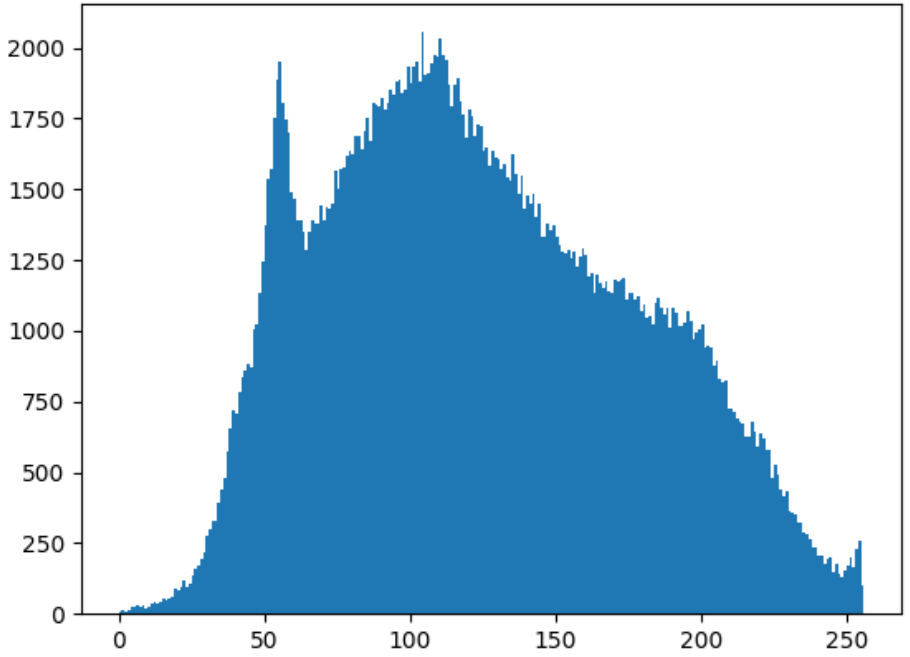
$$m = \frac{\ln \beta}{\ln(1+\delta)} + 1.$$

$$m = \frac{2,3\lg(10 \div 40)}{0,03} = 80 \div 130$$

Гистограмма яркости

Гистограмма - это график статистического распределения элементов цифрового изображения с различной яркостью, в котором по горизонтальной оси представлена яркость, а по вертикали — относительное число пикселей с конкретным значением яркости.

$$M_i = \sum_{j=1}^{i} m_j.$$

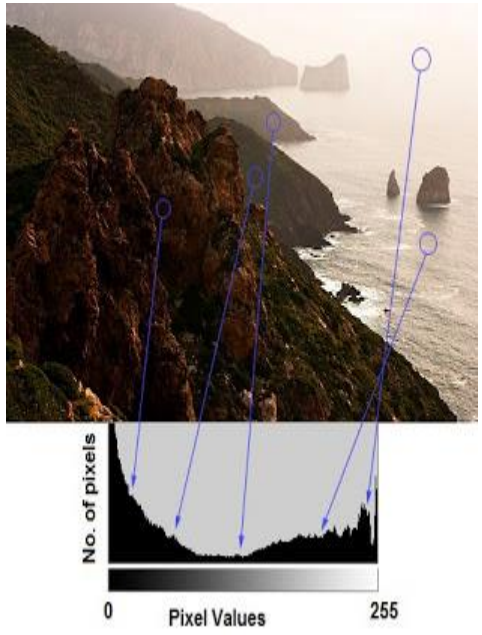


Яркость и контрастность

Гистограмма

cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])

1. images : it is the source image of type uint8 or float32. it should be given in square brackets, ie, "[img]".
2. channels : it is also given in square brackets. It is the index of channel for which we calculate histogram. For example, if input is grayscale image, its value is [0]. For color image, you can pass [0], [1] or [2] to calculate histogram of blue, green or red channel respectively.
3. mask : mask image. To find histogram of full image, it is given as "None". But if you want to find histogram of particular region of image, you have to create a mask image for that and give it as mask. (I will show an example later.)
4. histSize : this represents our BIN count. Need to be given in square brackets. For full scale, we pass [256].
5. ranges : this is our RANGE. Normally, it is [0,256].



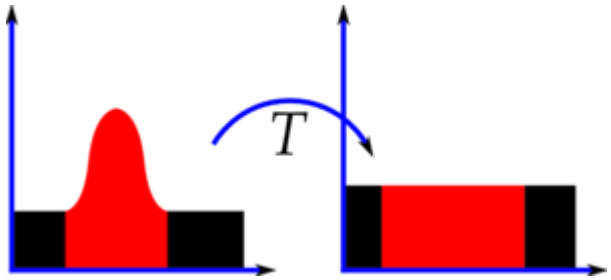
Передача яркости

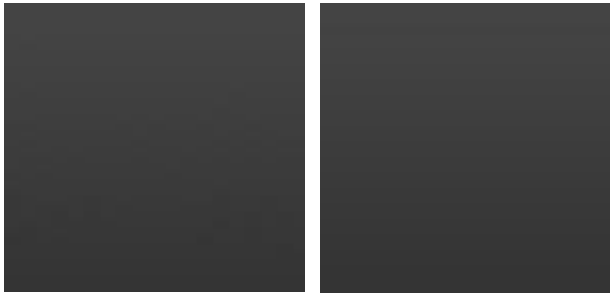
- Диапазон чувствительности датчика
- Функция передачи яркости

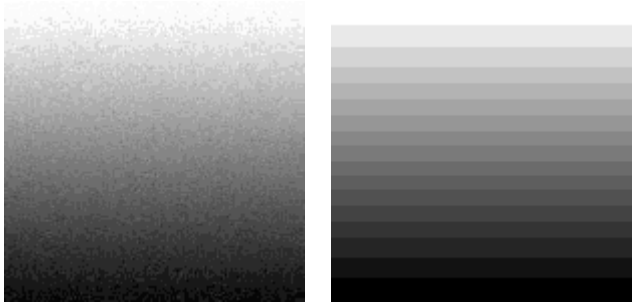
HDR

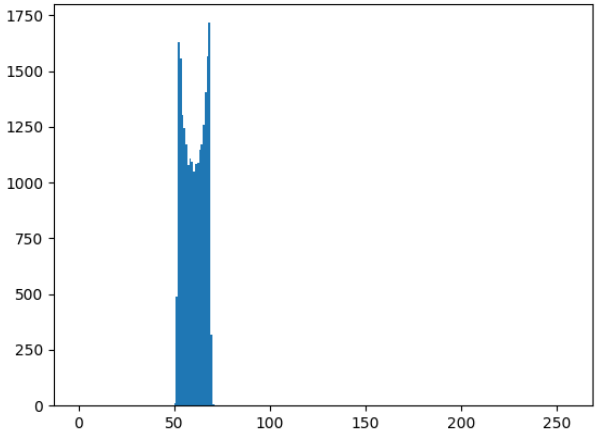
Линейная коррекция

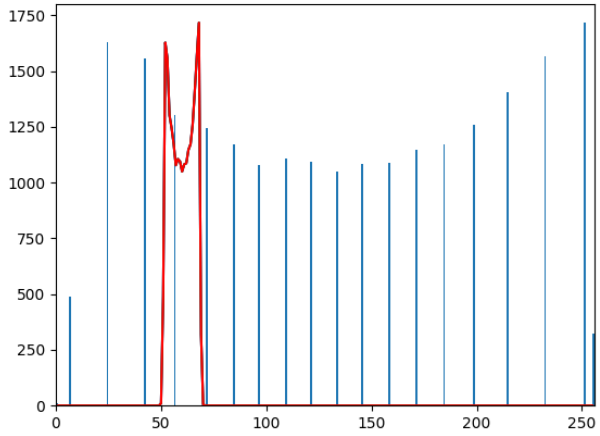
Histogram equalization $f(y)=(y-y_{min})\cdot\frac{255}{y_{max}-y_{min}}$



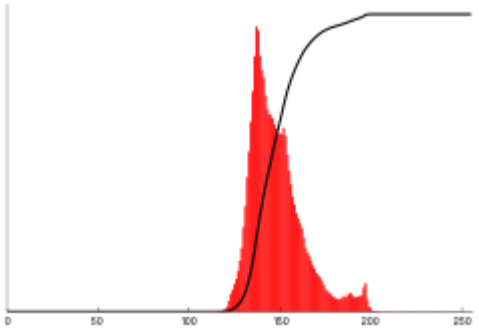


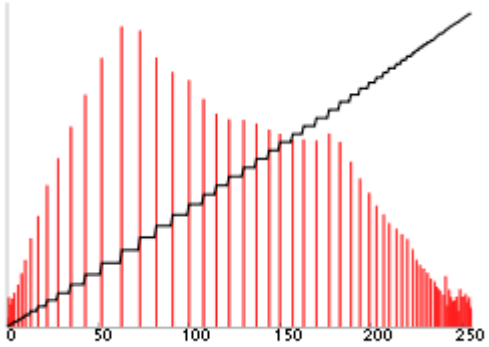






Линейная коррекция

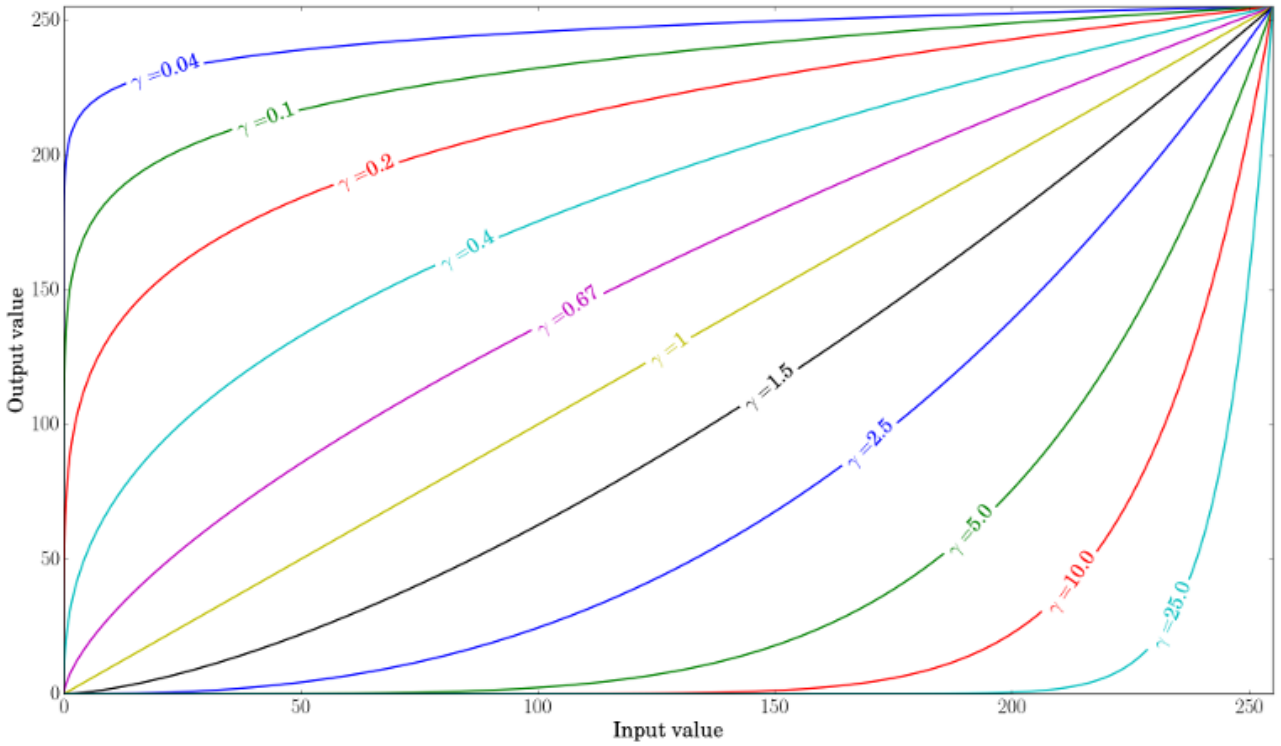




Гамма-коррекция

$$f(y')=\left(\frac{y}{255}\right)^{\gamma}\cdot 255$$

Gamma-correction

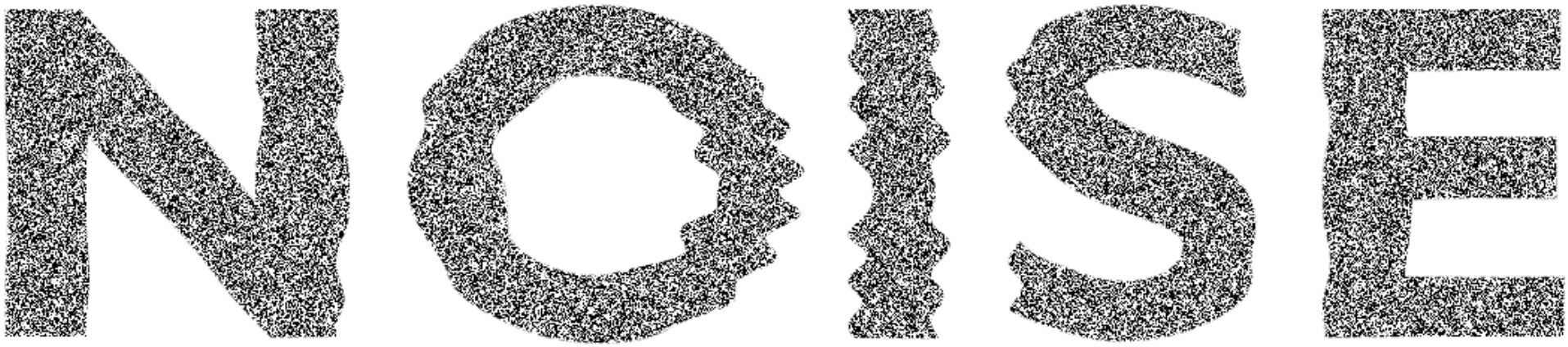


Гамма-коррекция

α=1.3
β=40.

γ=0.4.

Шум



Источники шума

– неидеальное оборудование для захвата изображения (ТВ-тюнер, видеокамера, сканер и т.п.);

– плохие условия съемки (например, сильные шумы, возникающие при ночной фото/видеосъемке);

– помехи, возникающие при передаче по аналоговым каналам, т. е. наводки от источников электромагнитных полей, собственные шумы активных компонентов (усилителей) линии передачи;

– искажение данных при их передаче через цифровые каналы или повреждение информации на носителе;

– неточности при выделении яркостного и цветоразностных сигналов из аналогового композитного сигнала и т. п.;

– спекл-шум, возникающий на радиолокационных изображениях и изображениях ультразвуковой диагностики, вызван энергетическими помехами из-за беспорядочно распределенных отражателей сигнала, слишком мелких для того, чтобы их могла отобразить система.

Классификация шума

По способу искажения

- аддитивный шум,
- мультипликативный шум,
- импульсный шум

С точки зрения визуального восприятия

- белый шум – сигнал, отсчеты которого не коррелируют друг с другом. Его разновидностью является белый гауссовский шум.
- импульсный шум – случайные изолированные точки на изображении, значения которых значительно отличаются от значений окружающих их точек (обычно возникает при передаче по аналоговым каналам)
- цветные пятна – характерны для аналогового сигнала (к примеру, присутствуют в видеоизображении, оцифрованного с носителя стандарта *VHS* или аналогов);
- биение пикселей – области, точки в которых имеют произвольное значение в связи с ошибкой декодирования.

- шум, вызываемый помехами в электросети;
- вертикальные царапины (характерны для старых черно-белых видеозаписей). Вертикальные царапины возникают при механическом повреждении эмульсии на пленке.

Цифровой шум

- Тепловой шум матрицы
- Шум переноса заряда
- Шум квантования АЦП
- Усиление сигналов в цифровом фотоаппарате
- Грязь, пыль на сенсоре
- Компрессия
- Помехи при передаче

Цифровой шум

На левой части изображения приведён фрагмент фотографии снятой при неблагоприятных условиях (длинная выдержка, высокая чувствительность ISO), шум хорошо заметен.
На правой части изображения — фрагмент фотографии снятой при благоприятных условиях. Шум практически незаметен

Артефакты сжатия

Ошибка передающего тракта

Соль и перец

Original

Salt and pepper noise

Impulse noise

Gaussian noise

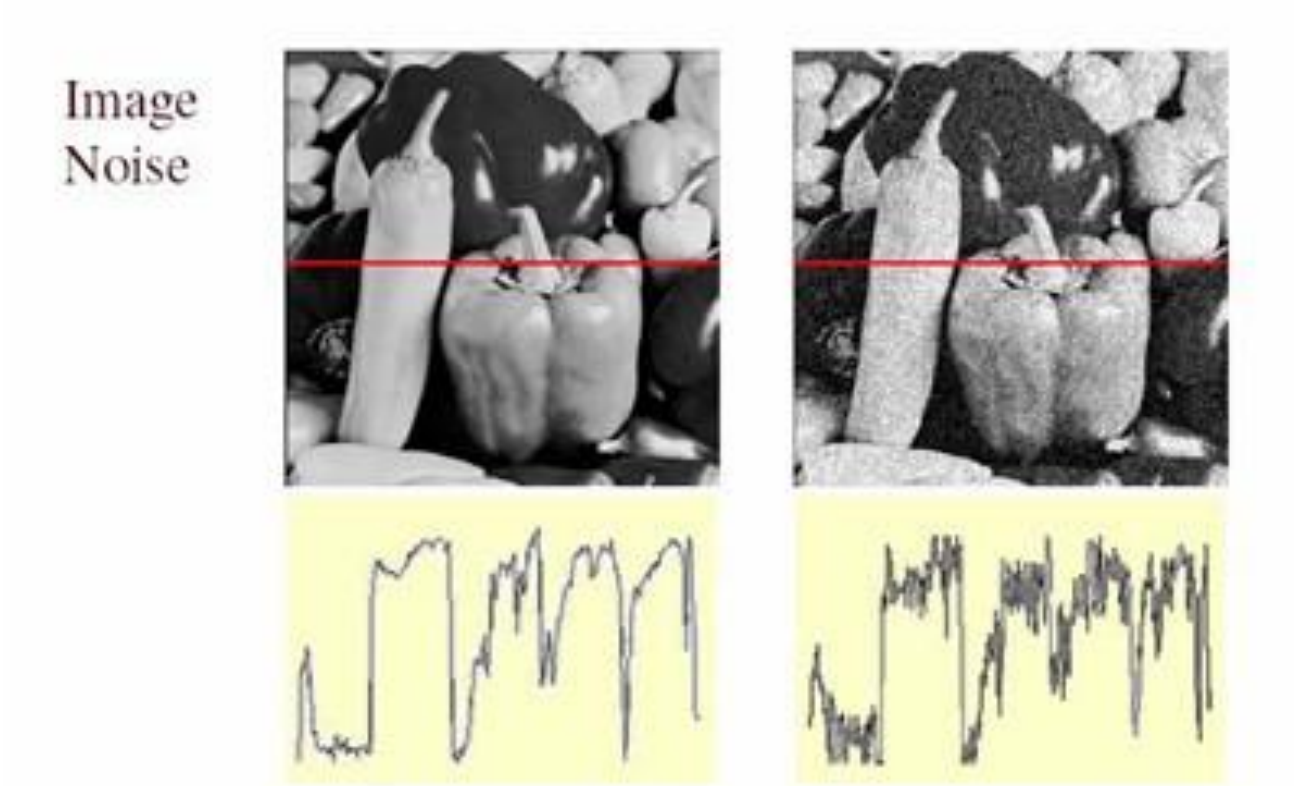
Соль и перец: содержит случайные белые и чёрные пикселы

Импульсный шум: случайные белые пикселы

Гауссов шум: распределяется по нормальному закону распространения Гауссового шума

Source: S. Seitz

Гауссов шум



$$f(x,y) = \overbrace{\bar{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

Source: M. Herbert

Временная фильтрация

Усреднение пикселов (измерений) в разные моменты времени

Серия зашумленных изображений

Результат усреднения по 10 изображениям

$$I(i,j) = g_r(i,j) + Err(i,j);$$

$$\bar{I}(i,j) = \frac{1}{N}\sum_{k=1}^{N} I_k(i,j);$$

$$E(\bar{I}(i,j)) = g_r(i,j);$$

Source: Konushin

Пространственная фильтрация

- Фильтры усреднение пикселей по соседям
  - вычисление среднего арифметического
  - вычисление среднего геометрического
  - вычисление среднего гармонического
  - вычисление среднего контрагорманического

- Фильтры основанные на порядковых статистиках
  - Медианные фильтры
  - фильтр максимума/минимума
  - Фильтр срединной точки
  - Фильтр усеченного среднего

- Линейные фильтры
  - Простые сглаживающие фильтры
  - Фильтр Гаусса

- Нелинейные фильтры
  - kuwahara Filter
  - 2D Cleaner Filter by Jim Casaburi
  - Spatial Smoother by Ioura Batugowski

Пространственная фильтрация

• Усреднение значения пиксела взвешенным средним по окрестности пиксела - свёртка

• Веса обозначаются как ядро фильтра

• Для каждого пикселя веса (ядро) одинаковые!

Пусть f – изображение, g -ядро. Свертка изображения f с помощью g обозначается как f * g:

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]\, g[k,l]$$

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

“box filter”

Source: D. Lowe

Линейным фильтром называется такое преобразование изображения, которое удовлетворяет двум свойствам:

filter(f 1 + f 2) = filter(f 1) + filter(f 2)
filter(a * f 1) = a * filter(f 1)

Инвариантность к сдвигу: фильтрация не зависит от сдвига пиксела:
filter(shift(f)) = shift(filter(f))

Любой линейный оператор (фильтр), инвариантный к сдвигу, может быть записан в виде свертки

Source: S. Lazebnik

Усреднение пикселей по соседям

вычисление среднего геометрического

$$f(x,y) = \left[\prod_{(s,t) \in S_{xy}} g(s,t) \right]^{\frac{1}{m \times n}}$$

S_{xy} - прямоугольную окрестность (множество координат точек изображения) размерами *mxn* с центром в точке *(x,y).*

g (x,y) – точка оригинального изображения (искаженного изображения)

f (x,y) – точка обработанного изображения

Усреднение пикселей по соседям

вычисление среднего гармонического

$$f(x,y) = \frac{m \times n}{\sum_{(s,t) \in S_{xy}} \frac{1}{g(s,t)}}$$

S_{xy} - прямоугольную окрестность (множество координат точек изображения) размерами *mxn* с центром в точке *(x,y).*
g (x,y) – точка оригинального изображения (искаженного изображения)
f (x,y) – точка обработанного изображения

Среднегармонический фильтр хорошо работает в случае униполярного «белого» импульсного шума (т.е. когда значение шума соответствует появлению белых точек на изображении), но не работает в случае униполярного «черного» импульсного шума (когда значение шума соответствует появлению черных точек). Этот фильтр также хорошо работает для других типов шума, таких как гауссов шум.

Усреднение пикселей по соседям

вычисление среднего контрагорманического

$$f(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

S_{xy} – прямоугольную окрестность (множество координат точек изображения) размерами *mxn* с центром в точке *(x,y).*
g (x,y) – точка оригинального изображения (искаженного изображения)
f (x,y) – точка обработанного изображения
Q – порядок фильтра

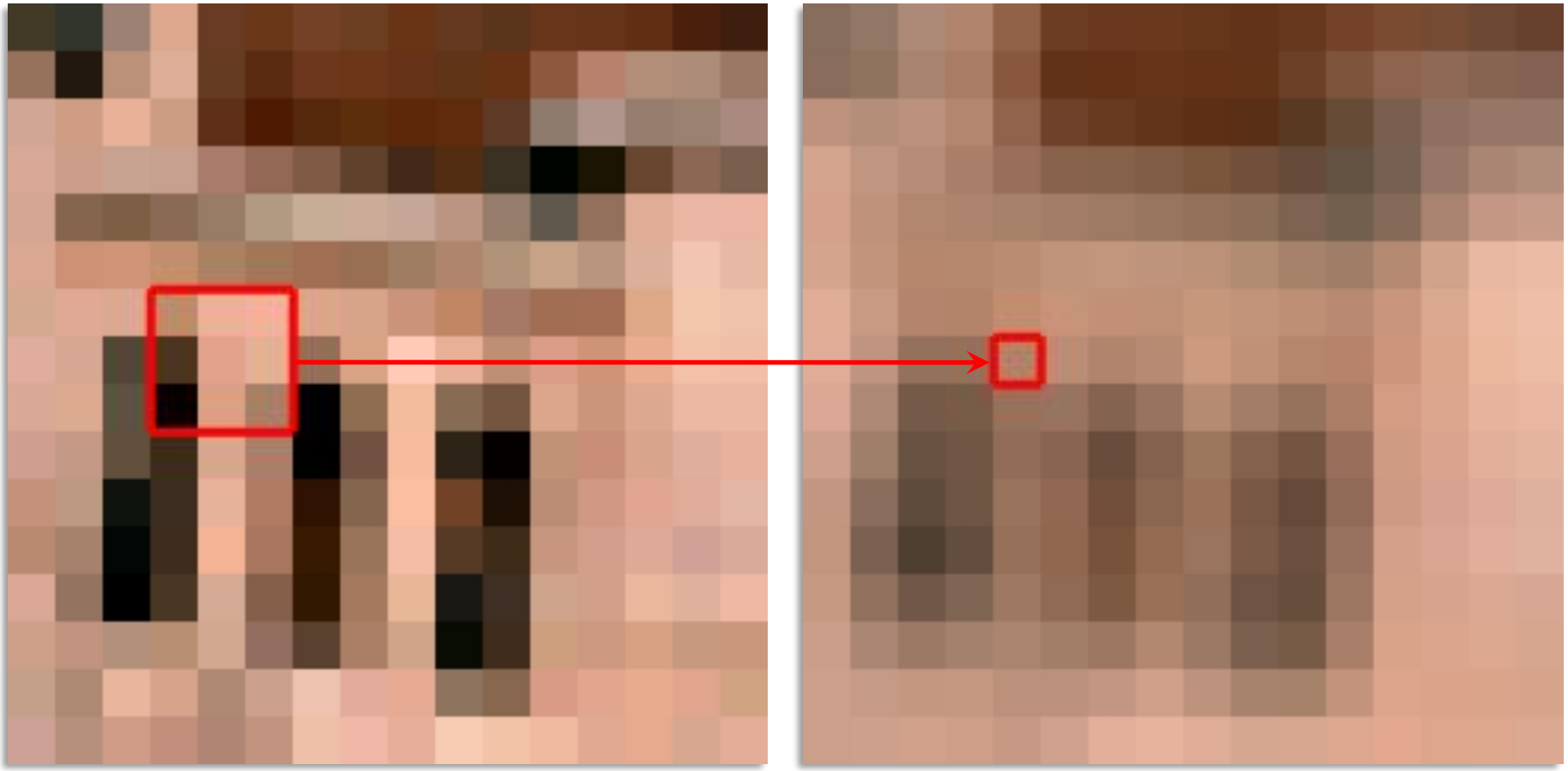
Этот фильтр хорошо приспособлен для уменьшения или почти полного устранения импульсного шума. При положительных значениях *Q* фильтр устраняет «черную» часть импульсного шума. При отрицательных значениях *Q* фильтр устраняет «белую» часть импульсного шума. Обе части шума не могут быть устранены одновременно.

Обработка скользящим окном:

Оригинал

Усреднение 3×3

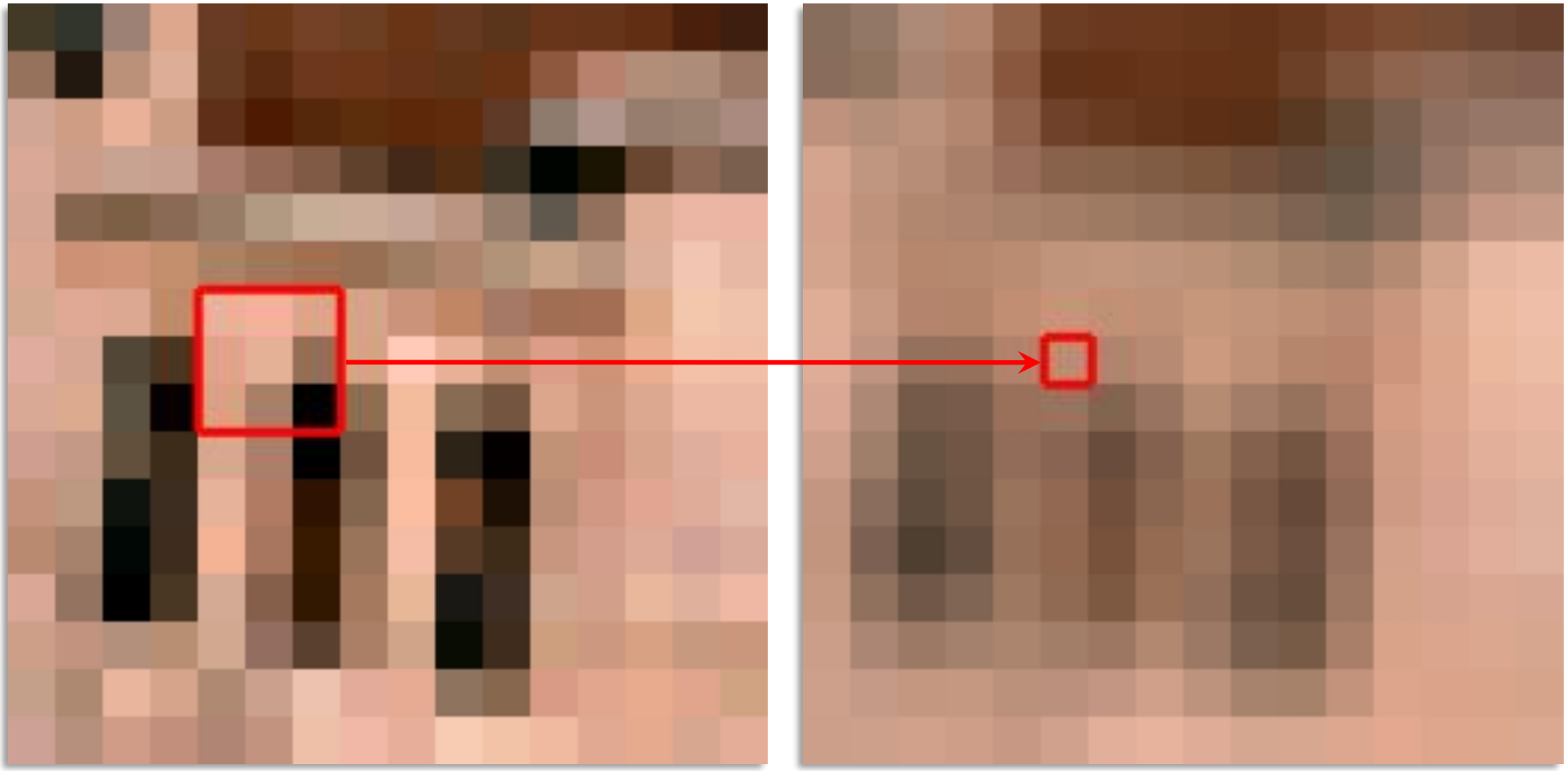
Обработка скользящим окном:



Оригинал

Усреднение 3×3

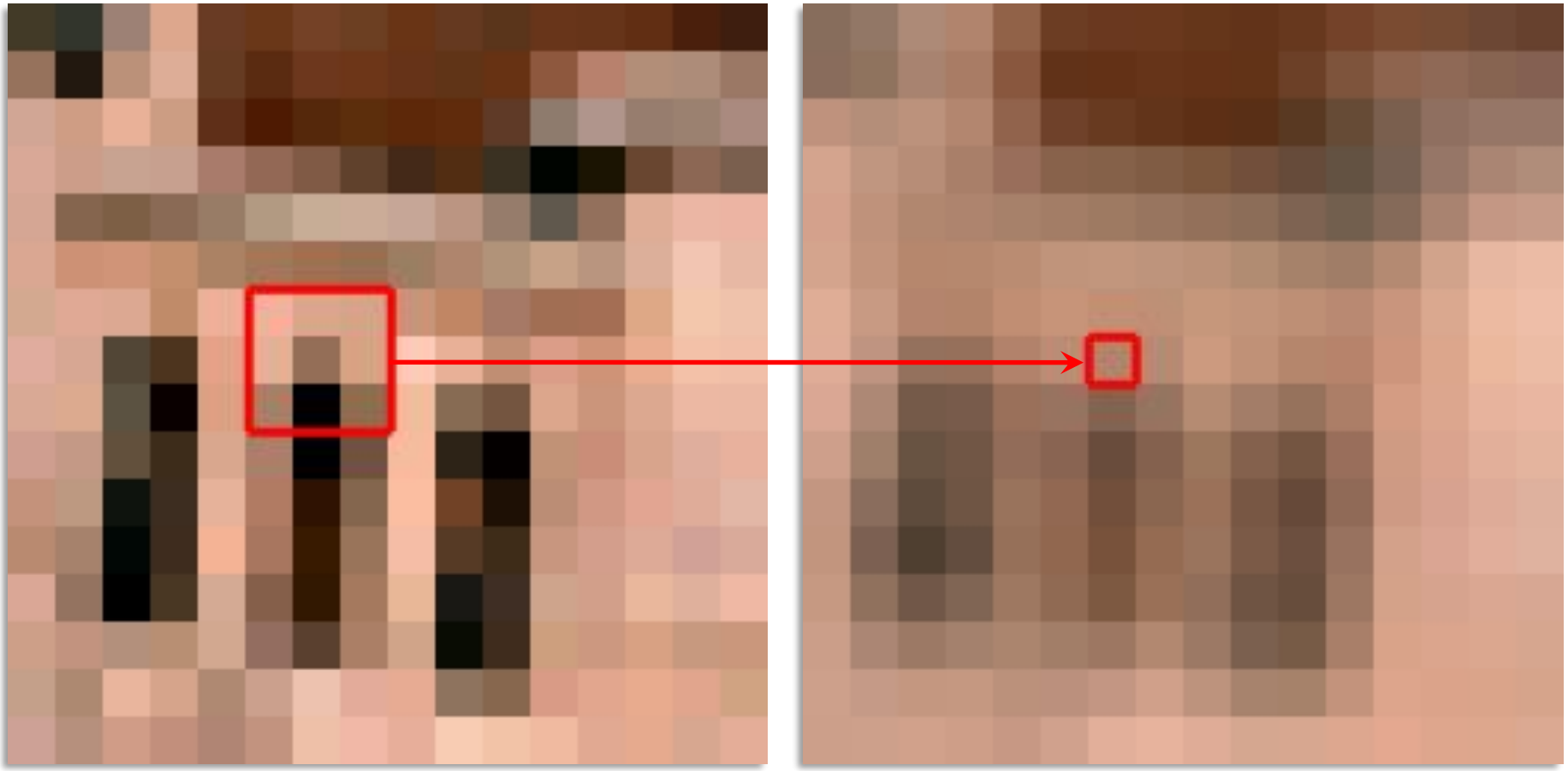
Обработка скользящим окном:



Оригинал

Усреднение 3×3

Обработка скользящим окном:



Оригинал

Усреднение 3×3

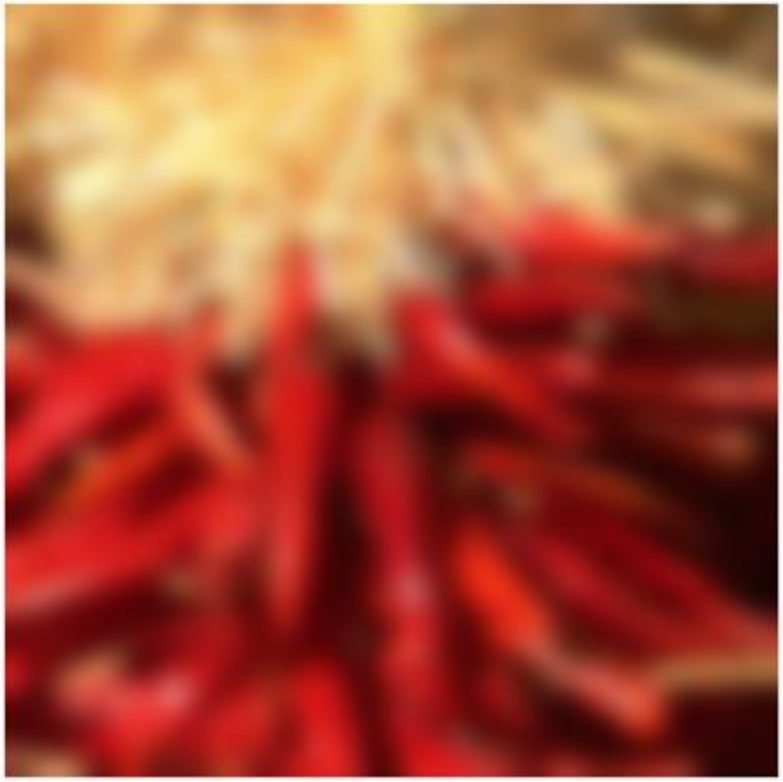
Фильтрация по краям

Окно фильтра выходит за границы изображения.

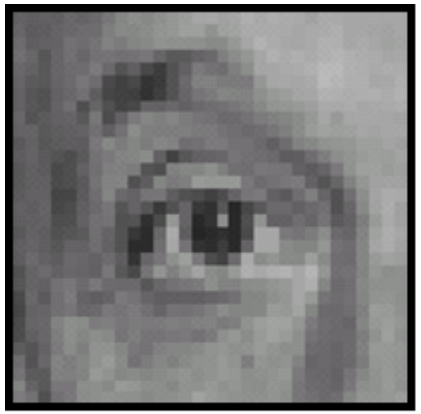
Необходимо экстраполировать изображение.

Варианты:

- clip filter (black)
- wrap around
- copy edge
- reflect across edge



Бокс-фильтр



Original

0	0	0
0	1	0
0	0	0

?

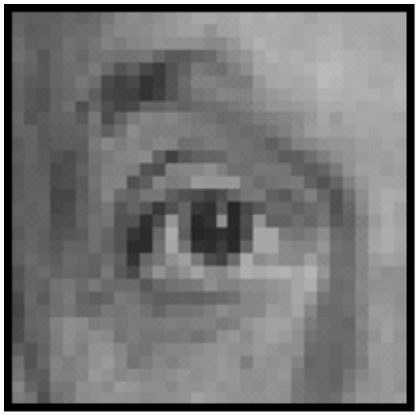
Бокс-фильтр

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Бокс-фильтр



Original

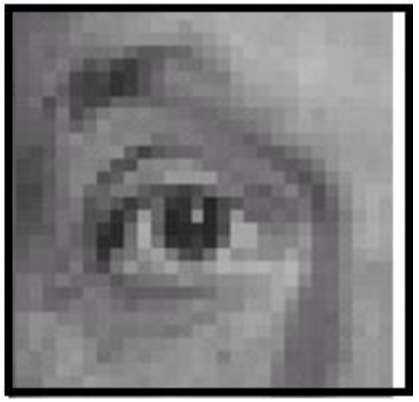
0	0	0
0	0	1
0	0	0

?

Бокс-фильтр

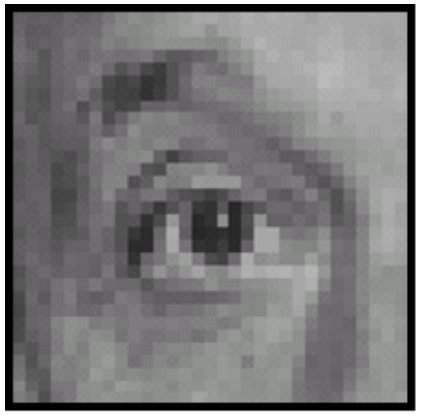
Original

0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Бокс-фильтр



Original

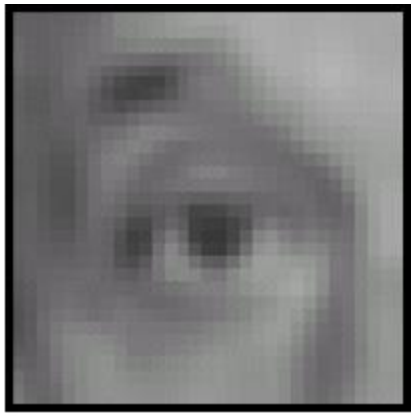
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

Бокс-фильтр

Original

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



Blur (with a box filter)

Функция filter2D

dst=cv.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])

src

input image.

dst

output image of the same size and the same number of channels as src.

ddepth

desired depth of the destination image, see **combinations**

kernel

convolution kernel (or rather a correlation kernel), a single-channel floating point matrix; if you want to apply different kernels to different channels, split the image into separate color planes using split and process them individually.

anchor

anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center.

delta

optional value added to the filtered pixels before storing them in dst.

borderType

pixel extrapolation method, see **cv::BorderTypes**

OpenCV свёртка

blur = cv.blur(img,(5,5))

dst=cv.blur(src, ksize[, dst[, anchor[, borderType]]])

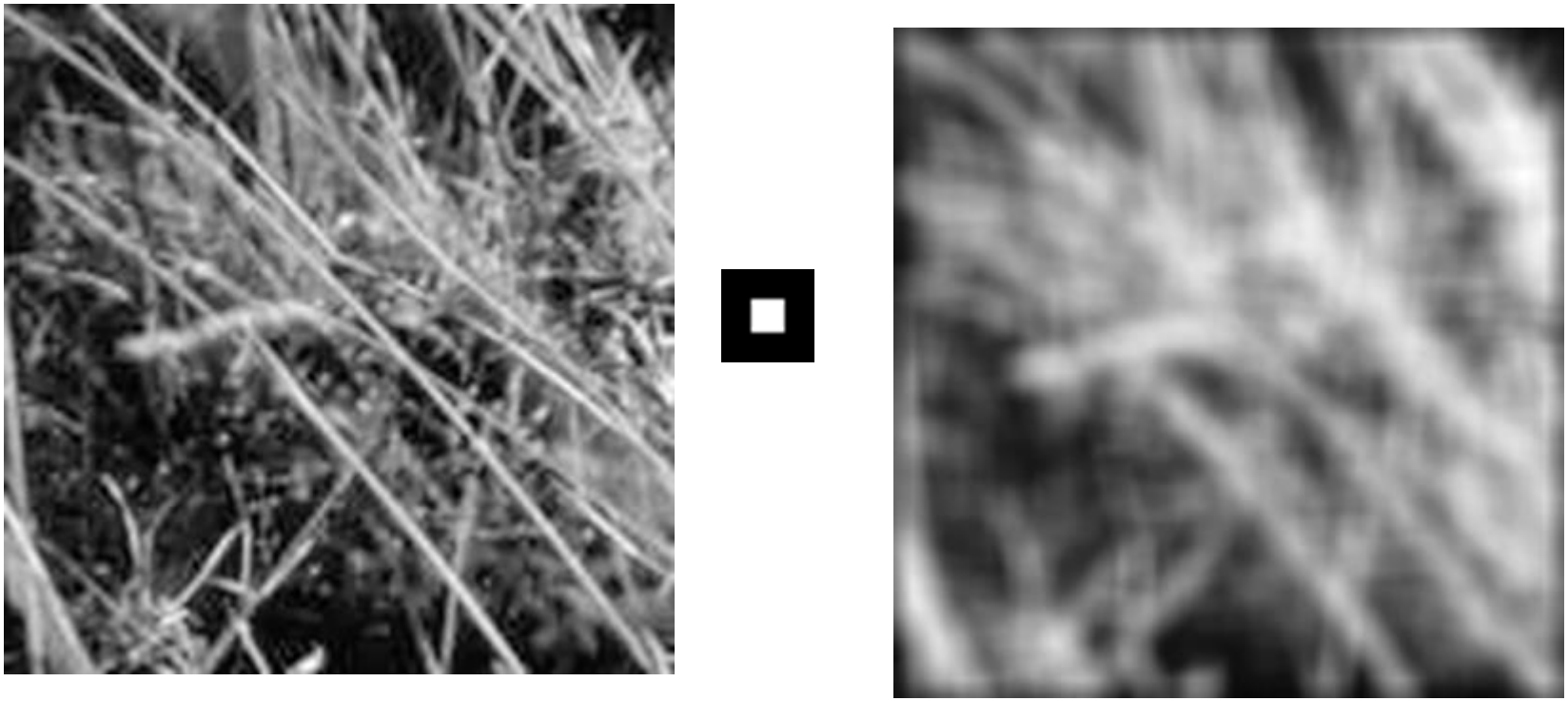
src input image; it can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.

ksize blurring kernel size.

Сглаживание box-фильтром

Что не так с изображением?

При сглаживании box-фильтром образуются паразитные линии

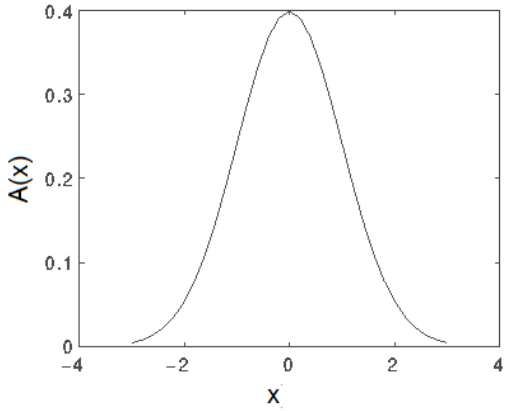


Source: D. Forsyth

Фильтр Гаусса

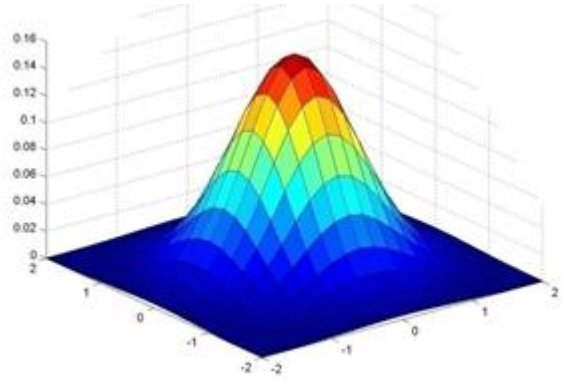
Функция распределения коэффициентов фильтра Гаусса в одномерном пространстве принимает вид:

$$A(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$



а в двумерном пространстве рассчитывается по формуле:

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

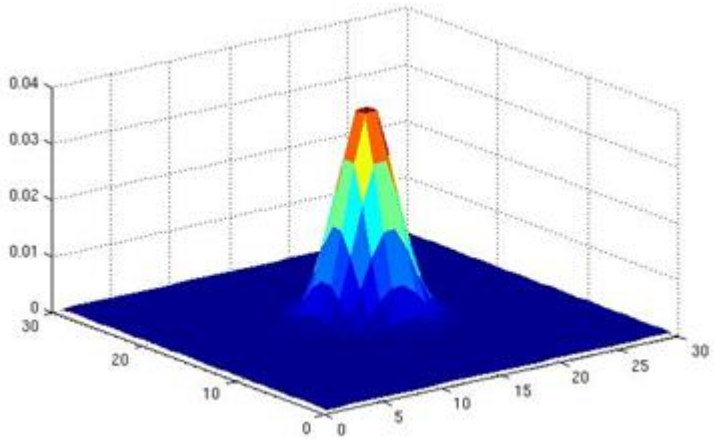


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

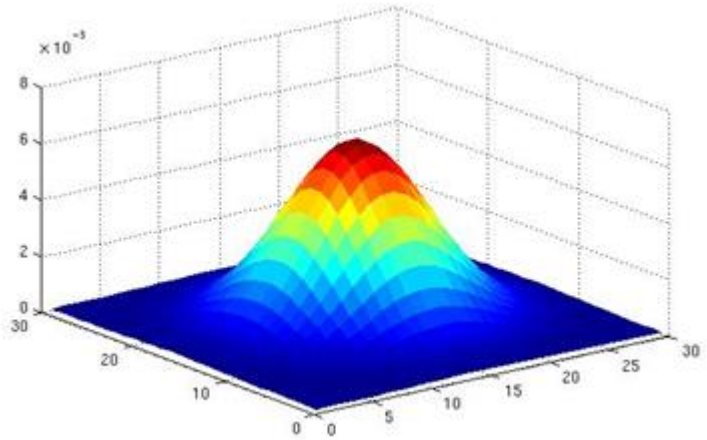
5 x 5, $\sigma = 1$

Ядро фильтра

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



σ = 2 with 30 x 30 kernel

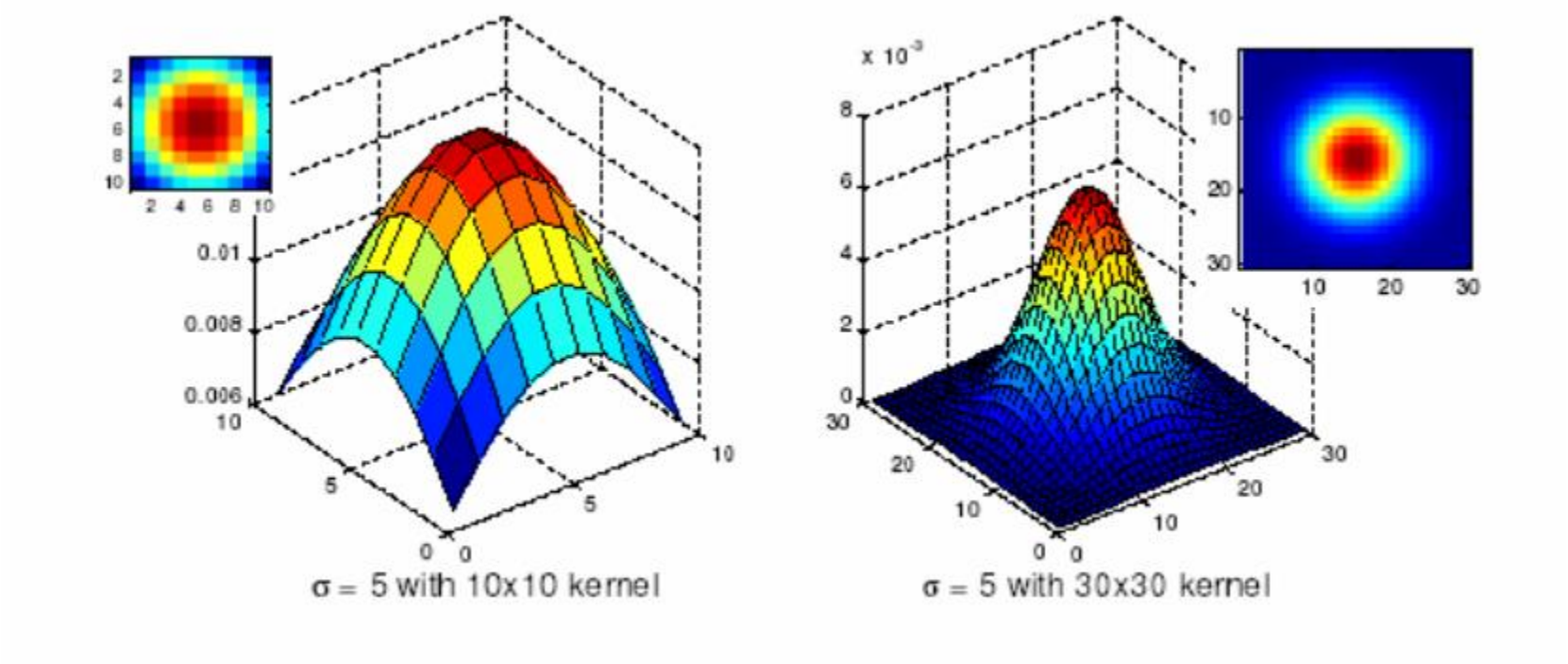


σ = 5 with 30 x 30 kernel

Значение sigma определяет степень размытости

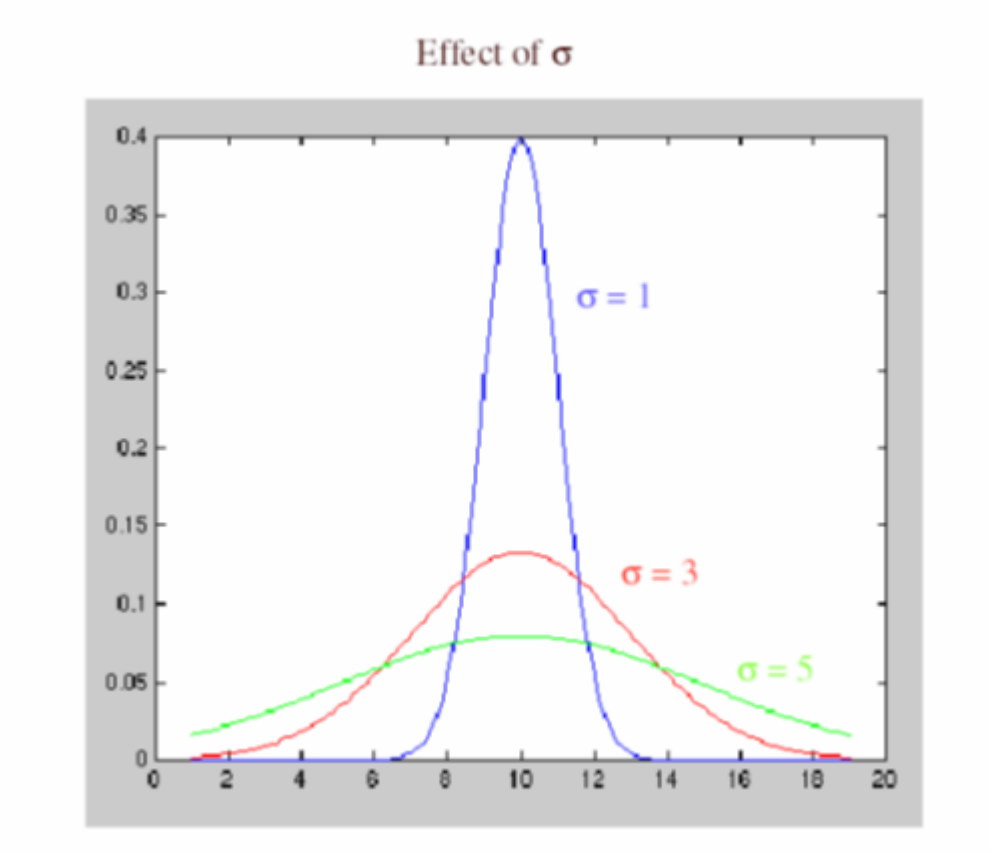
Выбор размера ядра

Размер ядра фильтра ограничен

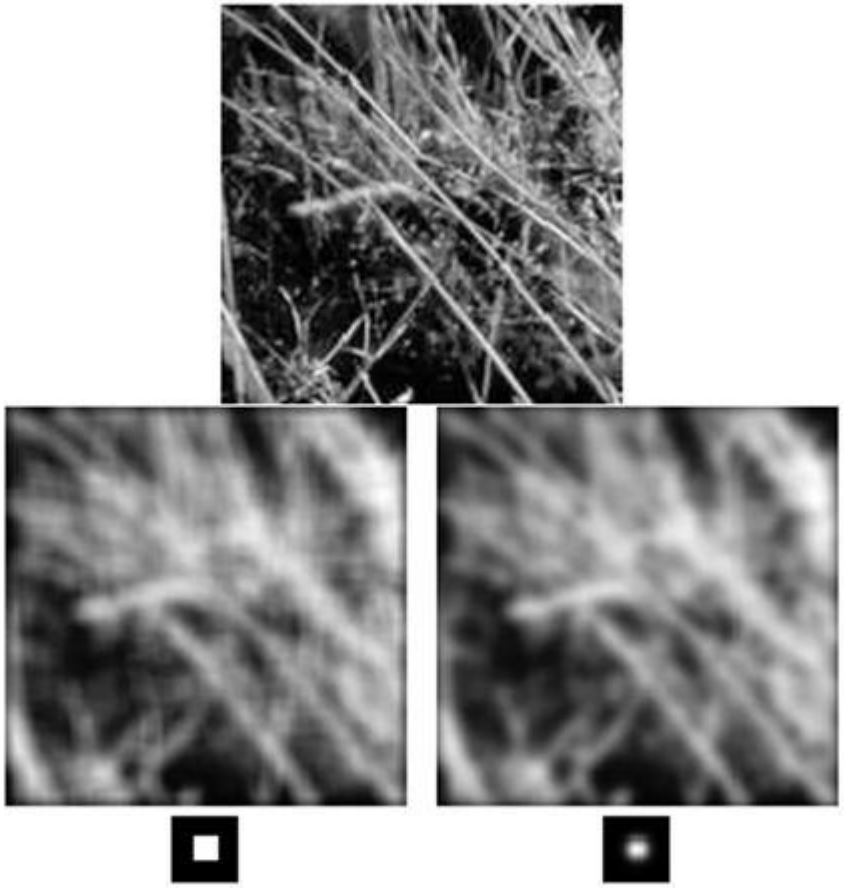


Source: K. Grauman

Эмпирическим путём выявлено, что размер ядра должна быть равной 6 sigma, т.е. веса на границах фильтра должны быть близки к 0.

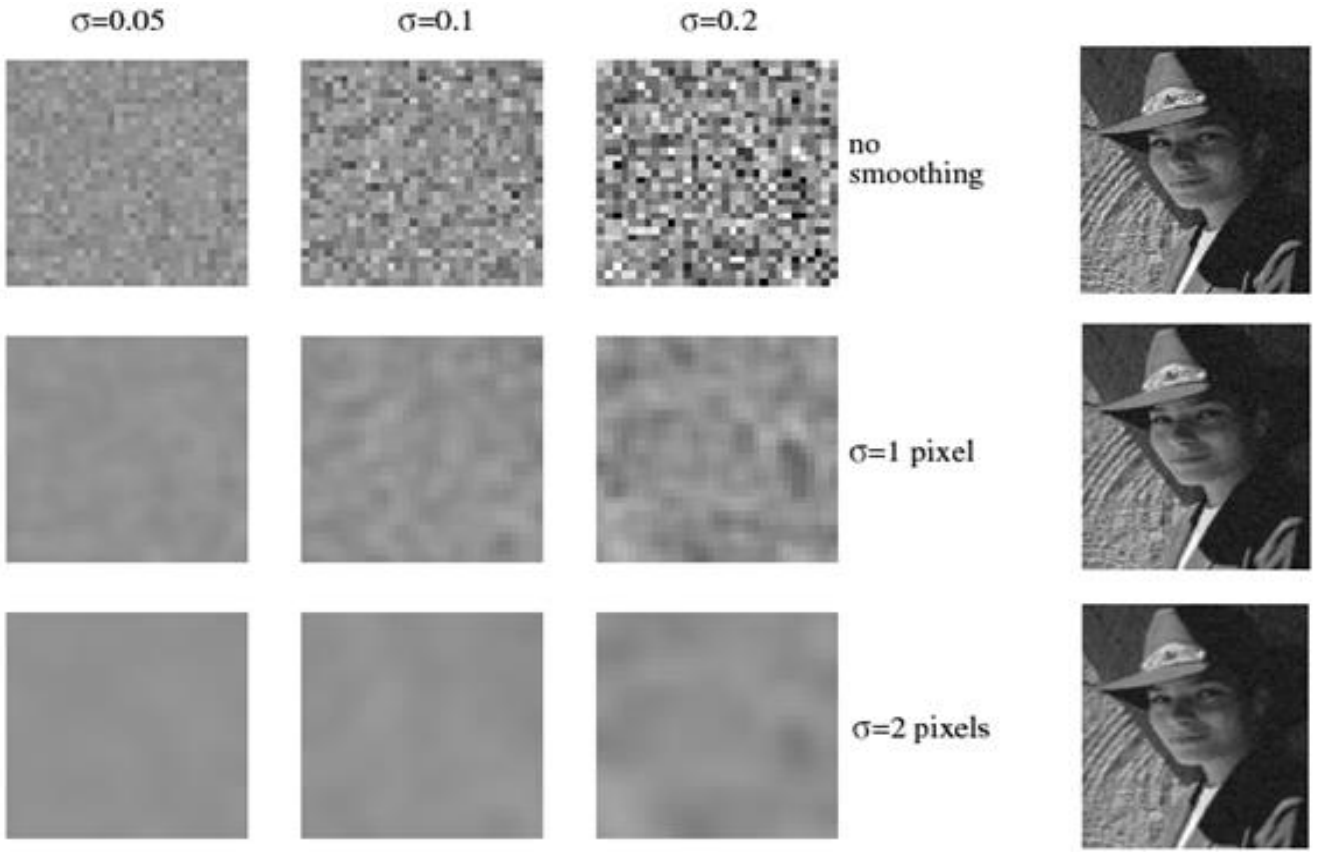


Фильтр Гаусса и box-фильтр



Source S. Lazebnik

Подавление гауссова шума



Slide by S. Lazebnik

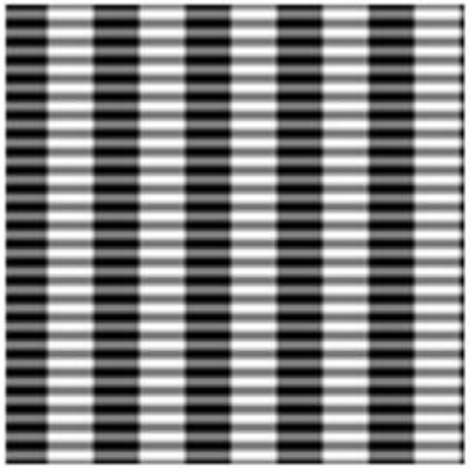
Фильтр Гаусса

- Сглаживание несколько раз фильтром с маленьким ядром дает результат, аналогичный свертке с большим ядром
- Свертка 2 раза с фильтром радиуса σ дает тот же результат, что с фильтром радиуса σ√2
- Фильтр Гаусса можно представить как сумму одномерных фильтров по осям x и y:

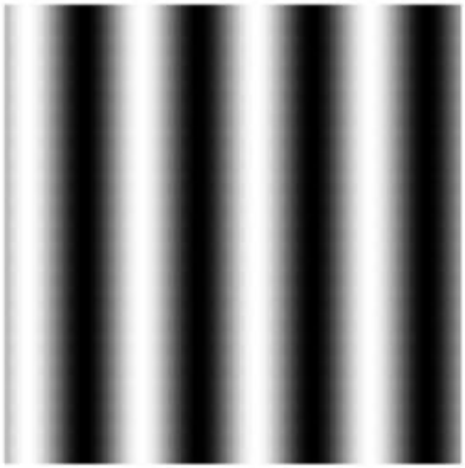
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

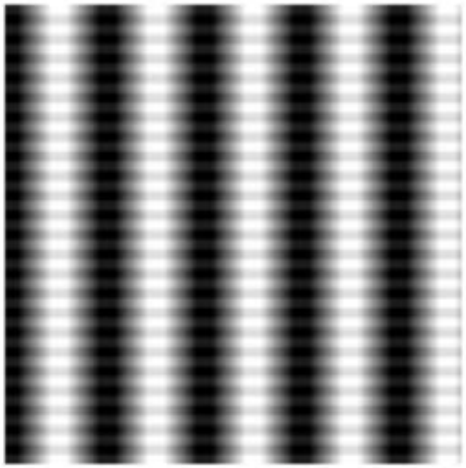
Результат свертки фильтром гаусса и усреднения



Исходное изображение



Фильтр Гаусса с Sigma = 4



Усреднение по 49 пикселям (7x7)

Фильтр Гаусса по сути является фильтром низких частот.

Фильтр Гаусса

blur = cv.GaussianBlur(img,(5,5),0)

src

input image; the image can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.

ksize

Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.

sigmaX

Gaussian kernel standard deviation in X direction.

sigmaY

Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively (see **cv::getGaussianKernel** for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.

Уменьшение шума Salt and Pepper

Фильтры основанные на порядковых статистиках

Медианные фильтры

$$f(x,y) = \underset{(s,t)\in S_{xy}}{\text{med}}\{g(s,t)\}$$

фильтр максимума/минимума

$$f(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\} \qquad f(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

S_{xy} – прямоугольную окрестность (множество координат точек изображения) размерами *mxn* с центром в точке *(x,y).*

g() – точки оригинального изображения (искаженного изображения)

f (x,y) – точка обработанного изображения

Фильтры основанные на порядковых статистиках

Фильтр срединной точки

$$f(x,y)=\frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\}+\min_{(s,t)\in S_{xy}}\{g(s,t)\}\right].$$

Фильтр усеченного среднего

$$f(x,y)=\frac{1}{mn-d}\sum_{(s,t)\in Sxy} g_r(s,t),$$

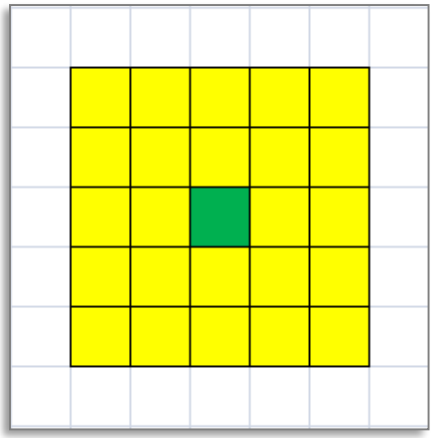
S_{xy} – прямоугольную окрестность (множество координат точек изображения) размерами *mxn* с центром в точке *(x,y).*

g() – точки оригинального изображения (искаженного изображения)

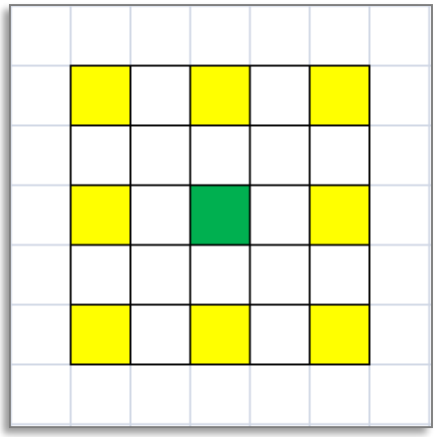
$g_r()$ – оставшиеся точки оригинального изображения после отсечения

f (x,y) – точка обработанного изображения

Вариация ядра фильтров

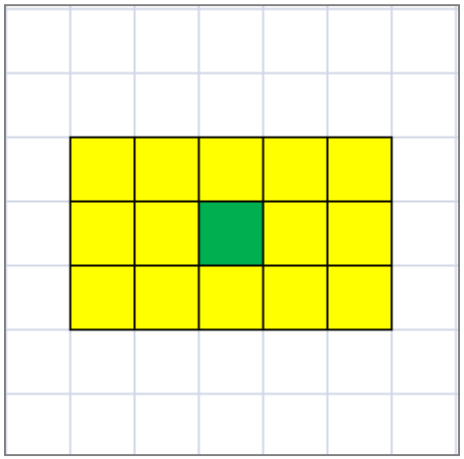


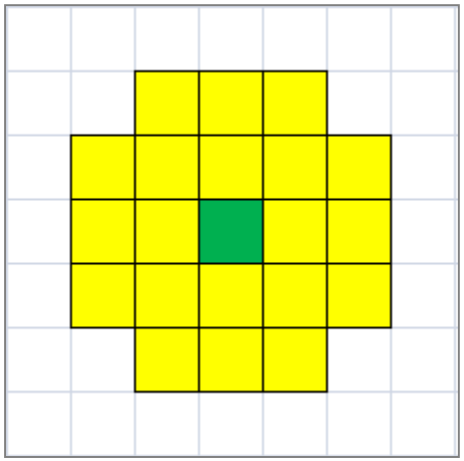
Классическая реализация

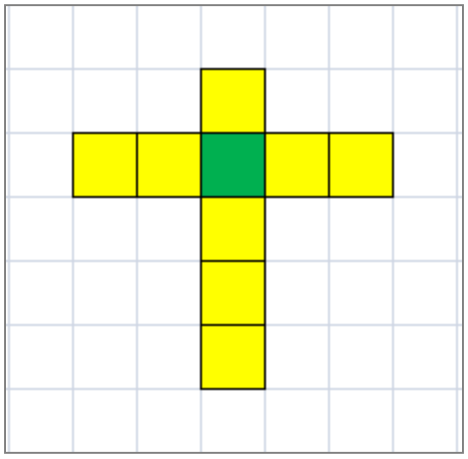


Модификация Де Хааном

Другие возможные виды ядра фильтра







Kuwahara Filter

A	A	AB	B	B
A	A	AB	B	B
AC	AC	ABCD	BD	BD
C	C	CD	D	D
C	C	CD	D	D

ШАГ1 вычисление среднего арифметического значение м дисперсии для областей **a b c d** с учётом расширения размерности

$$As = \frac{1}{m \times n} \sum_{k=0}^{m} \sum_{l=0}^{n} A_{k,l} \qquad \ldots\ldots\ldots\ldots \qquad Ds = \frac{1}{m \times n} \sum_{k=0}^{m} \sum_{l=0}^{n} D_{k,l}$$

ШАГ2 выявление области с наименьшей дисперсией

ШАГ3 назначение отклика как наименьшего среднего значения области с наименьшей дисперсией

Kuwahara Filter

Оригинал

Kuwahara Filter 7×7

Фильтр *2D Cleaner Filter by Jim Casaburi*

для каждого пикселя изображения рассчитывается значение с учетом окрестности по формуле:

$$P_{x,y} = \begin{pmatrix} R = Ch(Ts, r) \\ G = Ch(Ts, r) \\ B = Ch(Ts, r) \end{pmatrix}$$

$P_{x,y}$ – пиксель, с окрестностью которого ведется работа; *RGB* – значения каналов *RGB* спектра; *Ts* – значение порога, характеризующего возможность обработки; *r* – ранг окрестности (вид окрестности квадрат со стороной $2 \cdot r+1$); *Ch*(*Ts*,*r*) – функция расчета значения канала в спектре

$$Ch(Ts, r) = \frac{\sum_{i=-r}^{r} \sum_{j=-r}^{r} Sv(i, j)}{\sum_{i=-r}^{r} \sum_{j=-r}^{r} Cc(i, j)}$$

Фильтр *2D Cleaner Filter by Jim Casaburi*

Sv(*i*,*j*) – функция отсечения значения спектра по порогу; *Cc*(*i*,*j*) – функция указания пригодности значения спектра по порогу. Эти функции принимают следующие значения:

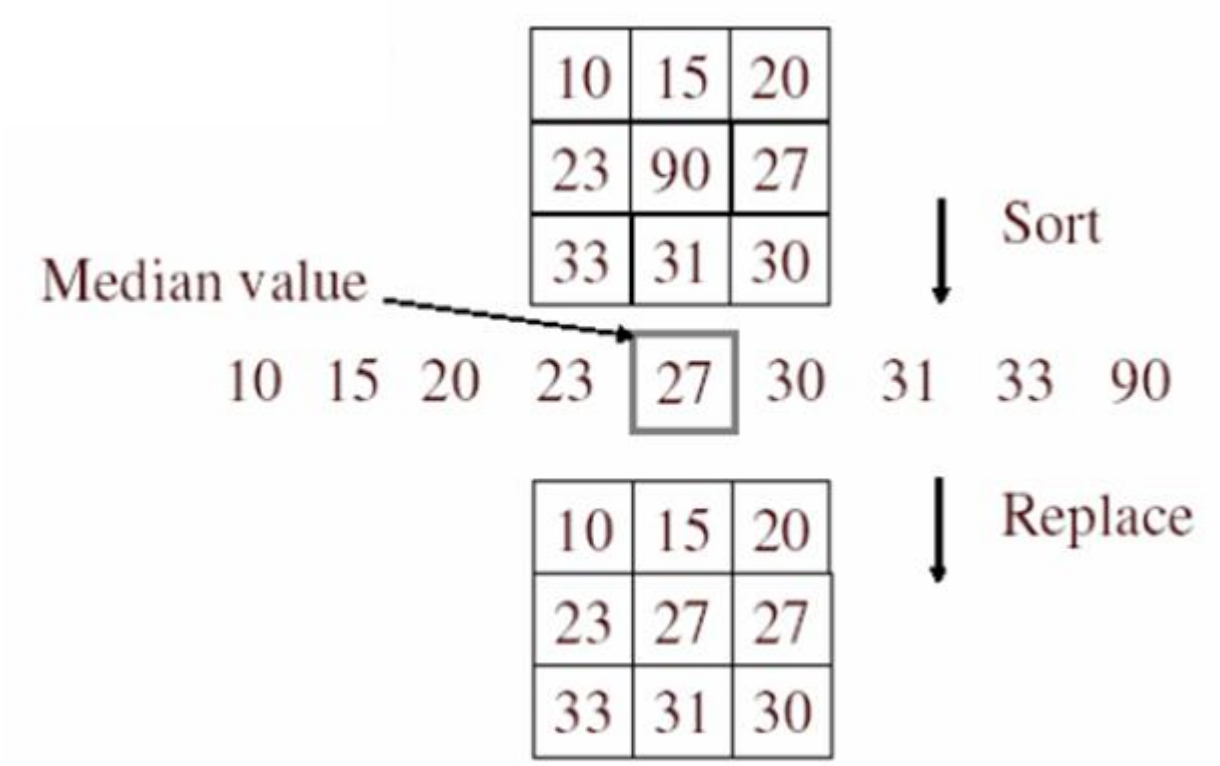
$$Sv(i,j)=\begin{cases} sv_{i,j}, & если \quad \left|sv_{i,j}-sv_{0,0}\right| \le Ts \\ 0, & если \quad \left|sv_{i,j}-sv_{0,0}\right| > Ts \end{cases}$$

$$Cc(i,j)=\begin{cases} 1, & если \quad \left|sv_{i,j}-sv_{0,0}\right| \le Ts \\ 0, & если \quad \left|sv_{i,j}-sv_{0,0}\right| > Ts \end{cases}$$

где $sv_{i,j}$ – значение спектра рассматриваемого цветового канала;
Ts – значение порога.

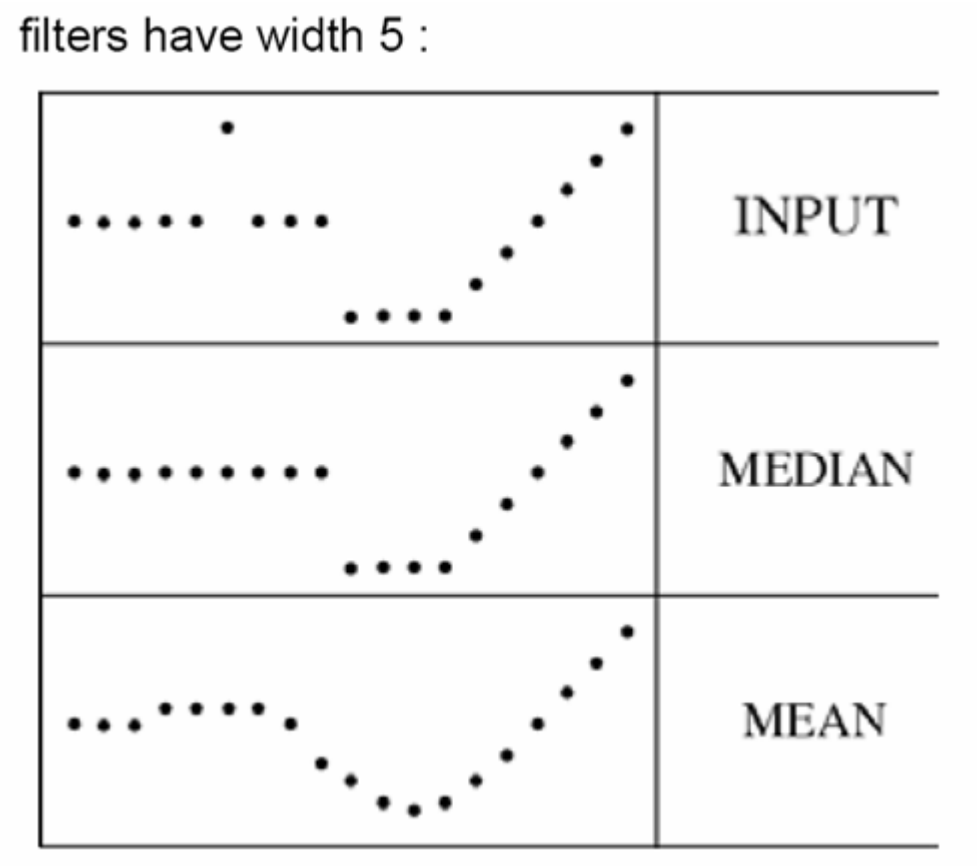
Медианная фильтрация

Выбор медианы из выборки пикселей по окрестности данного



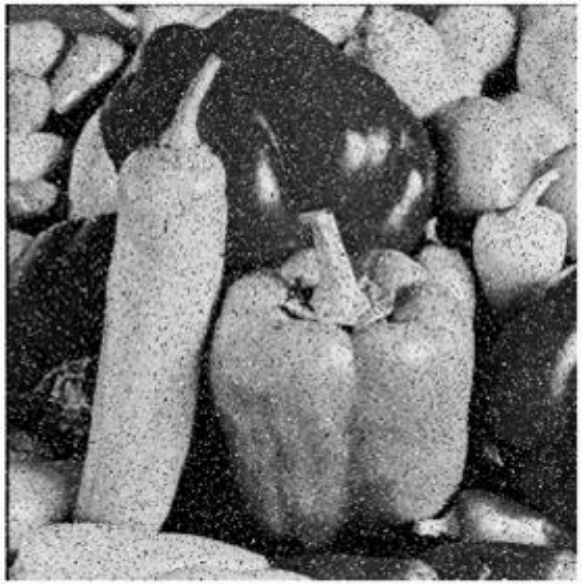
Source: K. Grauman

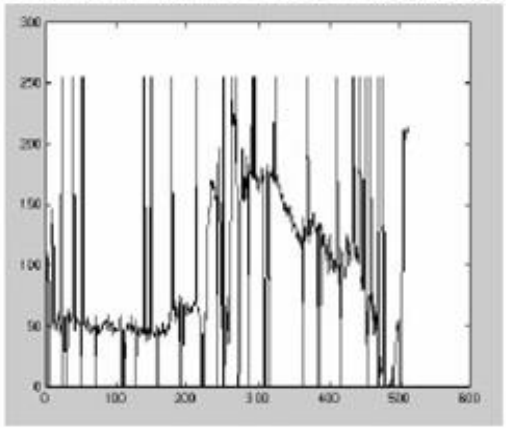
Фильтры с размером ядра 5 пикселов.

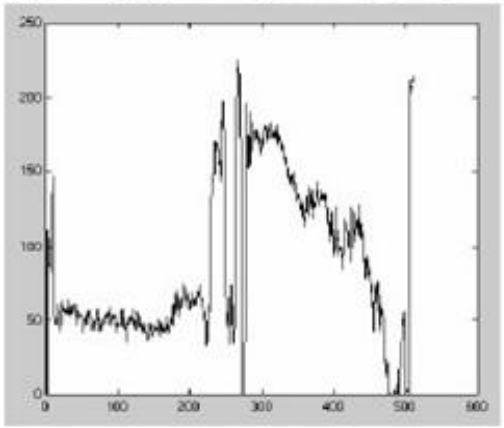


Source: K. Grauman

Медианная фильтрация







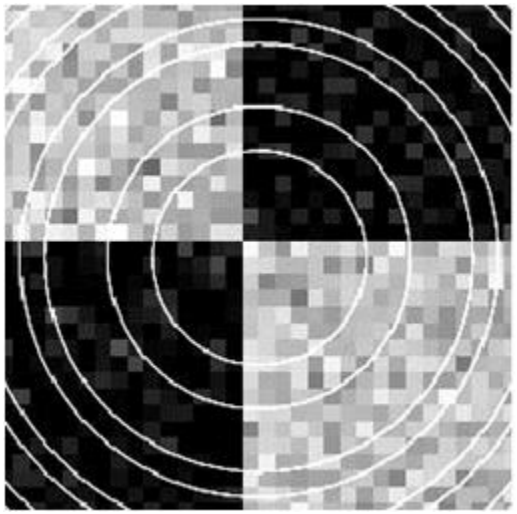
Фильтрация

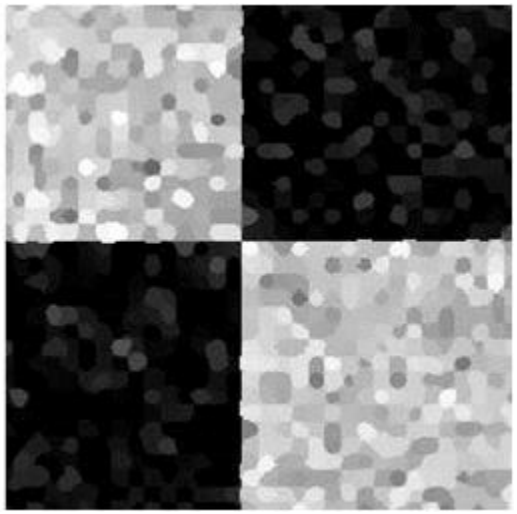
Gaussian

Median

Медианный фильтр

Результат применения медианного фильтра с радиусом в 7 пикселей к изображению с шумом и артефактами в виде тонких светлых окружностей.





Медианный фильтр

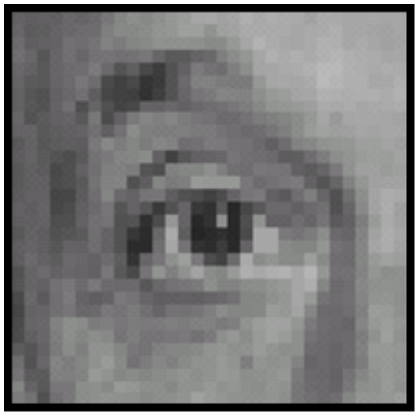
```
median = cv.medianBlur(img,5)
```

src

input 1-, 3-, or 4-channel image; when ksize is 3 or 5, the image depth should be CV_8U, CV_16U, or CV_32F, for larger aperture sizes, it can only be CV_8U.

ksize

aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...



Original

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

(Note that filter sums to 1)

Original

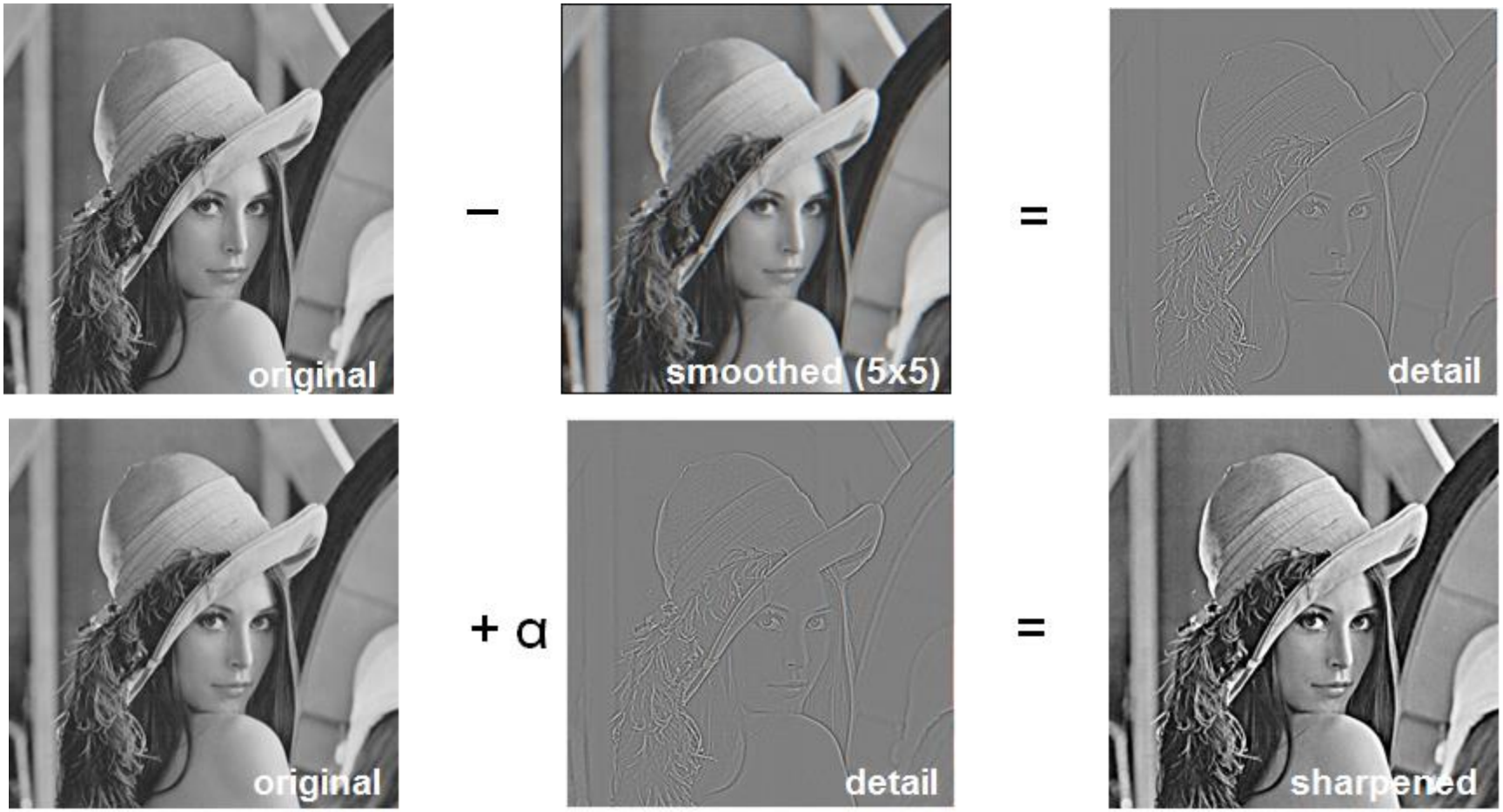
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

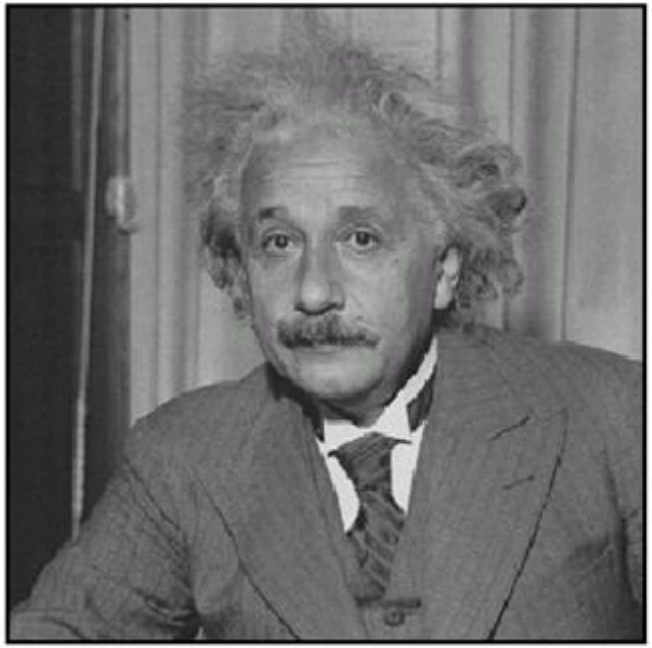
- Accentuates differences with local average

Source: D. Lowe

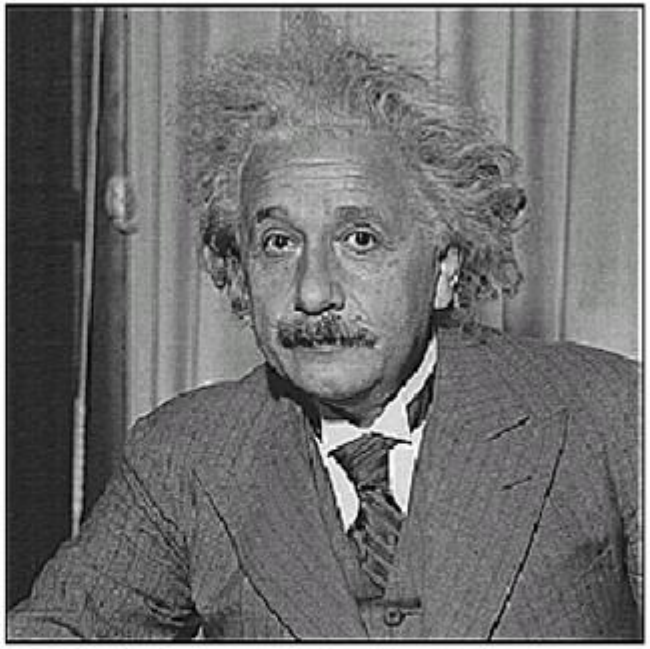
Повышение резкости



Sharpening



before



after

Source: D. Lowe

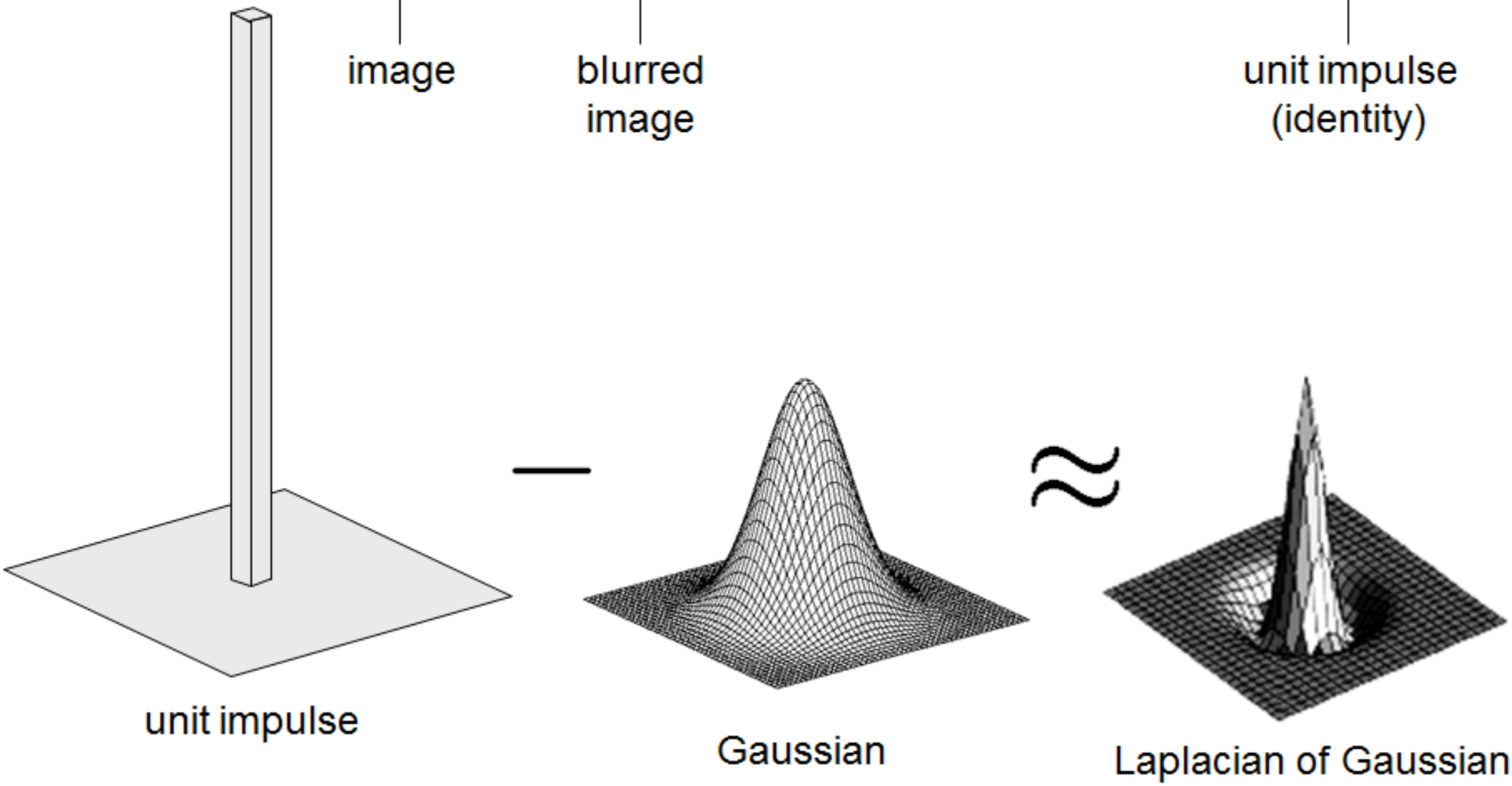
Фильтр unsharp

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

image

blurred image

unit impulse (identity)



unit impulse

Gaussian

Laplacian of Gaussian

Пример повышения резкости

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

```
#The code for an adding the noise to an image
img= cv2.imread('../Dropbox/tesla.jpg',0)
h,w = np.shape(img)
#create a null matrix
noise = np.zeros((h,w),np.uint8)
#add random values to pixe
cv2.randu(noise, 0,255)
#add noise
noisy_img = img + np.array(0.1*noise, dtype=np.int)
noisy_img[noisy_img>255]=255 #restrict the max value
noisy_img= np.array(noisy_img,dtype=np.uint8)
#convert to image type
```

Задание 1:

Внесите шумы в изображение с помощью вышеприведённого кода.

Сравните работу бокс-фильтра, фильтра Гаусса и медианного фильтра.

Задание 2:

Исправьте изображения 1.jpg, 2.jpg, 3.jpg