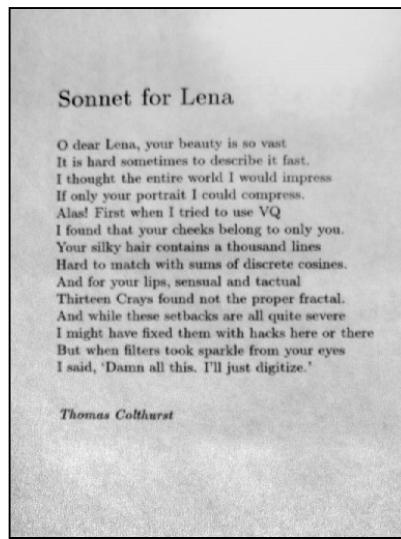
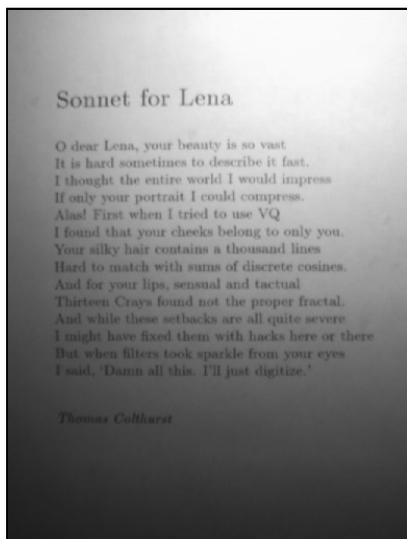
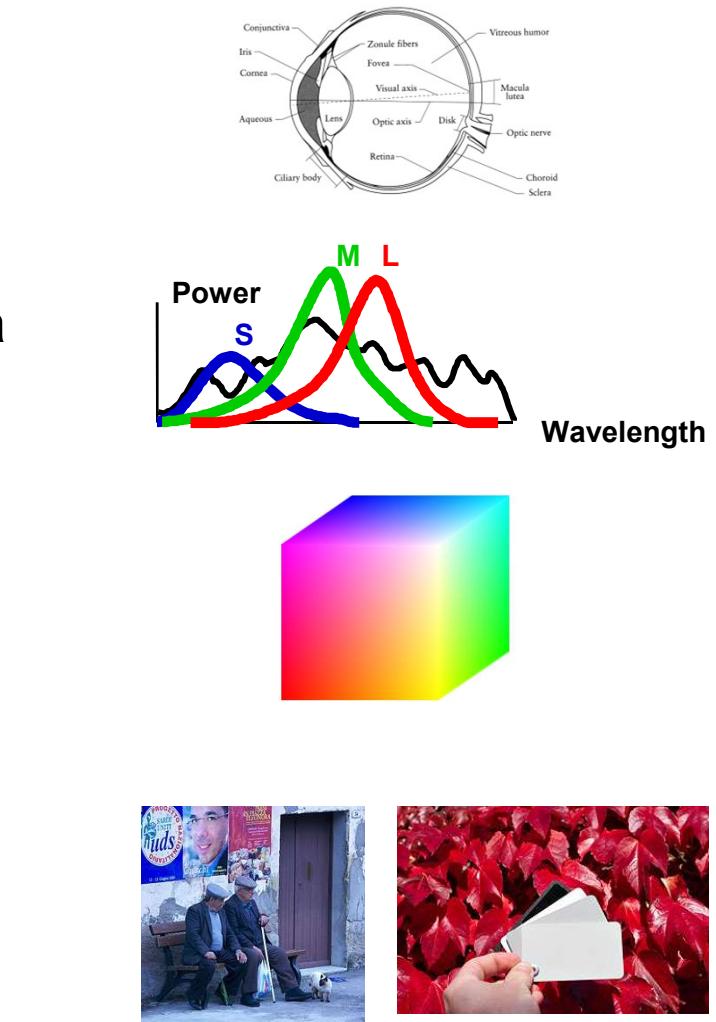


Обработка изображений



На предыдущей лекции

- Камера-обскура
 - Фотоаппарат, глаз
- Цвет
 - Психологическое свойство человека
 - Свет описывается спектром
- Сетчатка глаза
 - Колбочки 3х видов
 - Трихроматическая теория
 - 3 канала для пиксель
- Адаптация зрения
 - Цветовой баланс, «баланс белого»
 - Серые карточки, «серый мир»



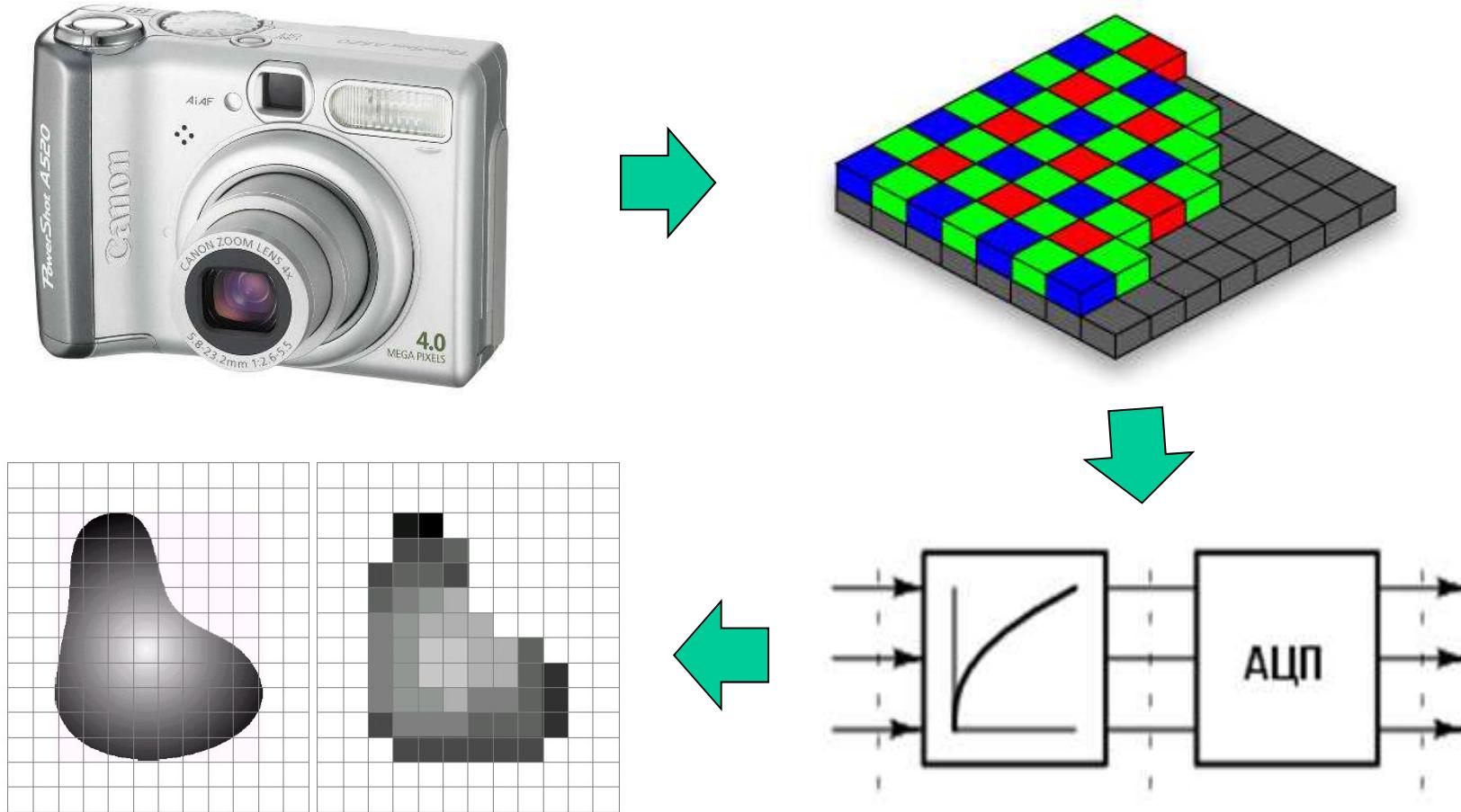
Обработка изображений

- Семейство методов и задач, где входной и выходной информацией являются изображения.
- Примеры :
 - Устранение шума в изображениях
 - Улучшение качества изображения
 - Усиления полезной и подавления нежелательной (в контексте конкретной задачи) информации

Обработка изображений

- Зачем обрабатывать?
 1. *Улучшение изображения для восприятия человеком*
 - цель – чтобы стало «лучше» с субъективной точки зрения человека
 2. *Улучшение изображения для восприятия компьютером*
 - цель – упрощение последующего распознавания
 3. *Развлечение (спецэффекты)*
 - цель – получить эстетическое удовольствие от красивого эффекта

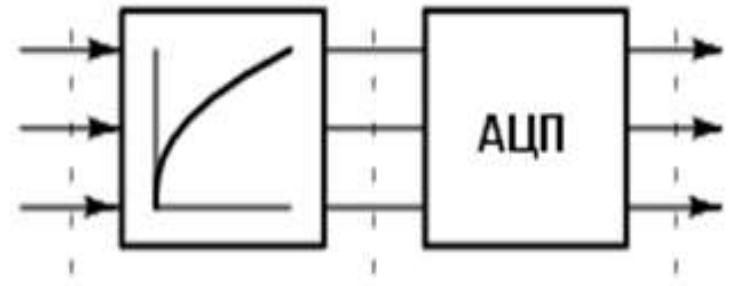
Цифровое изображение



Вспоминаем процесс получения цифрового изображения...

Почему оно может получиться плохо?

- Ограниченный диапазона чувствительности датчика
- “Плохой” функции передачи датчика



Что такое гистограмма?

Гистограмма – это график распределения яркостей на изображении. На горизонтальной оси - шкала яркостей тонов от белого до черного, на вертикальной оси - число пикселей заданной яркости.



0 255

Изменение контраста изображения

Что может не устраивать в полученном изображении:

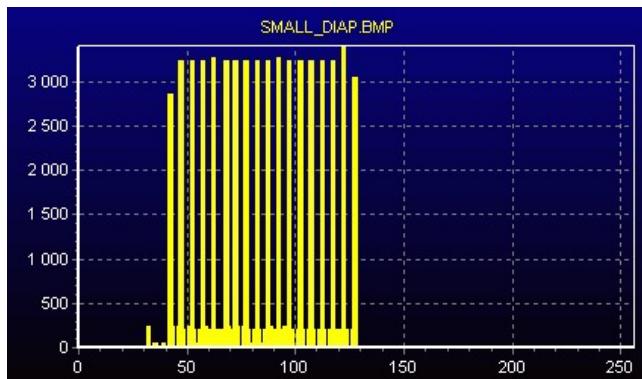
- Узкий или смещенный диапазон яркостей пикселей (тусклое или «пересвеченое» изображение)
- Концентрация яркостей вокруг определенных значений, неравномерное заполнение диапазона яркостей (узкий диапазон - тусклое изображение)

Коррекция - к изображению применяется преобразование яркостей, компенсирующий нежелательный эффект:

$$f^{-1}(y) = x \quad \begin{aligned} y & - \text{яркость пикселя на исходном изображении,} \\ x & - \text{яркость пикселя после коррекции.} \end{aligned}$$

Линейная коррекция

Компенсация узкого диапазона яркостей –
линейное растяжение:



$$f^{-1}(y) = (y - y_{\min}) * \frac{(255 - 0)}{(y_{\max} - y_{\min})}$$

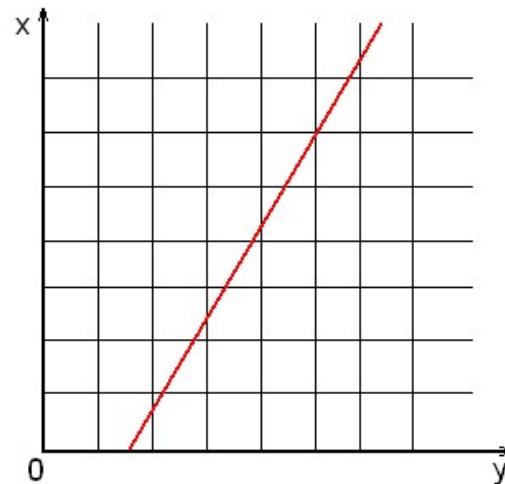
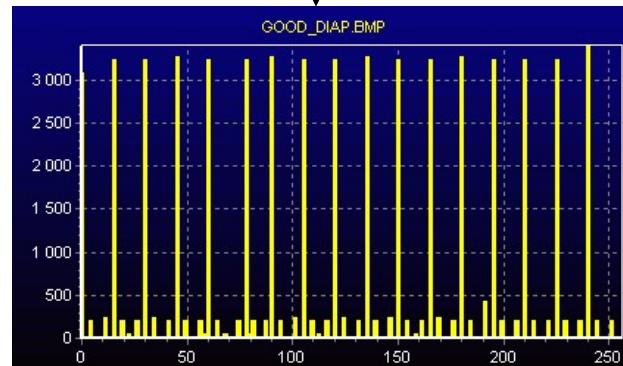
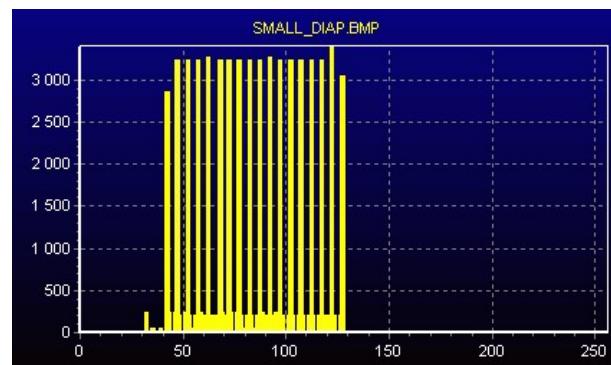
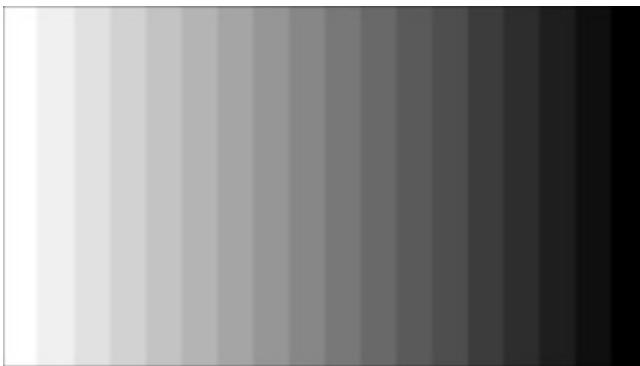


График функции $f^{-1}(y)$

Линейная коррекция

Компенсация узкого диапазона яркостей – линейное растяжение:



Линейная коррекция

Линейное растяжение – «как AutoContrast в Photoshop»



Линейная коррекция

Линейная коррекция помогает не всегда!



Нелинейная коррекция

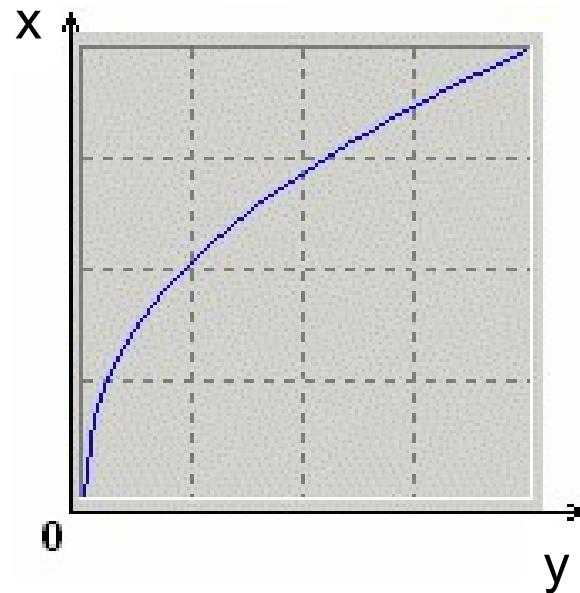


График функции $f^{-1}(y)$

Нелинейная коррекция

Нелинейная компенсация недостаточной контрастности

Часто применяемые функции:

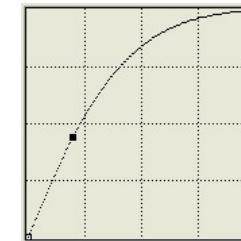
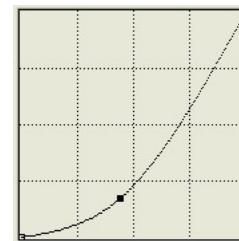
- Гамма-коррекция
 - Изначальная цель – коррекция для правильного отображения на мониторе.

$$y = c \cdot x^\gamma$$

- Логарифмическая
 - Цель – сжатие динамического диапазона при визуализации данных

$$y = c \cdot \log(1 + x)$$

Гамма-коррекция



Графики функции $f^{-1}(y)$

Нелинейная коррекция

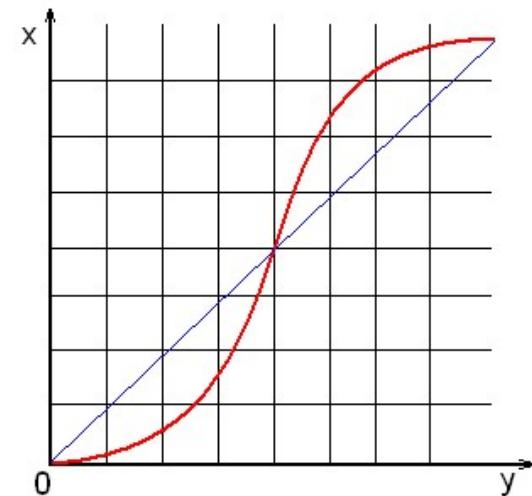
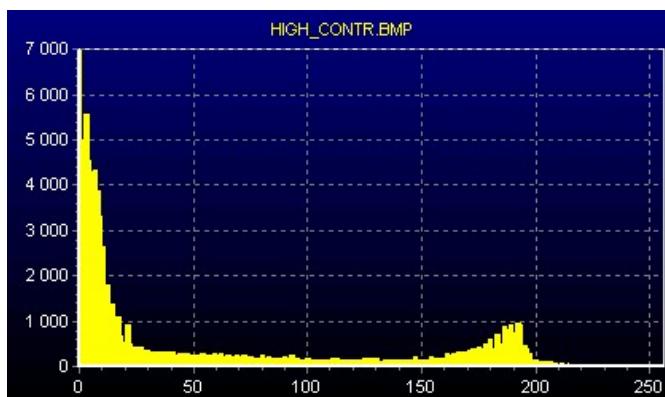
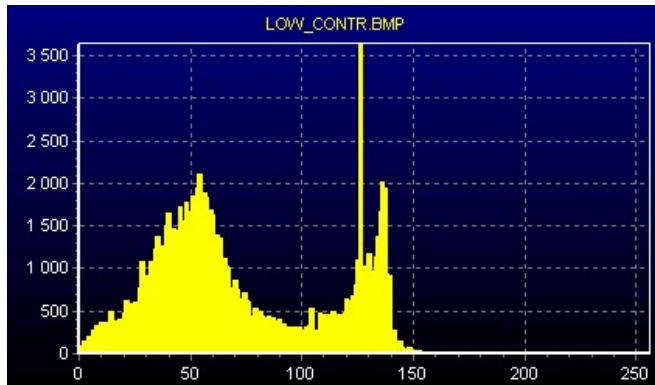
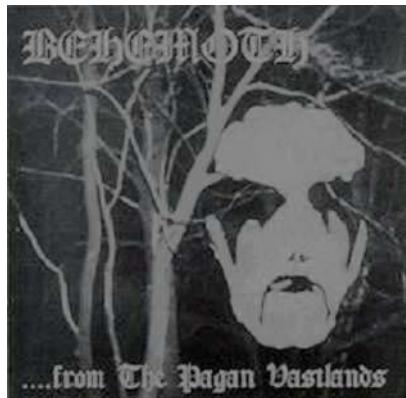


График функции $f^{-1}(y)$

Хроматическая адаптация

Чувствительность зрительной системы меняется в зависимости от доминантной освещенности наблюданной сцены

- Механизм плохо изучен

Адаптация к разным уровням освещенности

- Размер зрачка регулирует объем света, попадающий на сетчатку
- Размер резко меняется при входе в здание с ярко освещенной солнцем улицы

Цветовая адаптация

- Клетки сетчатки меняют свою чувствительность
- Пример: если доля красного в освещении повышается, понижается чувствительность клеток, отвечающих за красный, пока вид сцены не придет к норме
- Мы лучше адаптируемся при яркой освещенности, при освещении свечой все остается в желтых тонах

Баланс белого

Когда мы смотрим на фотографию или монитор, глаза адаптируются к освещению в комнате, а не к освещению сцены на фотографии

Если баланс белого неточен, цвета фотографии кажутся неестественными

incorrect white balance



correct white balance



<http://www.cambridgeincolour.com/tutorials/white-balance.htm>

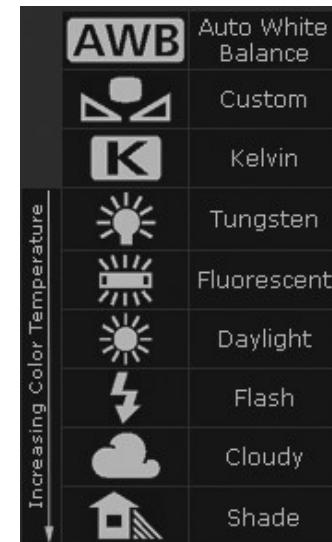
Баланс белого

Пленочные камеры:

- Разные виды пленки и светофильтры применяются для разных сцен

Цифровые камеры:

- Автоматический баланс белого
- Предустановки баланса белого для типичных условий съемки
- Настраиваемый по опорному объекту



<http://www.cambridgeincolour.com/tutorials/white-balance.htm>

Баланс белого

Von Kries adaptation

- Домножаем каждый канал на коэффициент передачи
- В ряде случаев эффект более сложный, соответствующий домножению на матрицу 3x3

Простейший способ: серые (белые) карточки

- Фотографируем нейтральный объект (белый)
- Оцениваем вес каждого канала
 - Если цвет объект записывается как r_w, g_w, b_w тогда веса $1/r_w, 1/g_w, 1/b_w$



Баланс белого

Если нет серых карточек, тогда нам нужно угадать
(или оценить) коэффициенты усиления

Модель «Серого мира» (Grayworld)

- Средний уровень («серый») по каждому каналу должен быть одинаков для всех каналов
- Если цветовой баланс нарушен, тогда «серый» в этом канале больше «серого» других каналов
- Вычислим коэффициенты усиления так, чтобы среднее в каждом канале стало одинаковым:

$$\bar{R} = \frac{1}{N} \sum R(x, y); \quad \bar{G} = \frac{1}{N} \sum G(x, y); \quad \bar{B} = \frac{1}{N} \sum B(x, y); \quad Avg = \frac{\bar{R} + \bar{G} + \bar{B}}{3};$$

$$R' = R \cdot \frac{Avg}{\bar{R}}; \quad G' = G \cdot \frac{Avg}{\bar{G}}; \quad B' = B \cdot \frac{Avg}{\bar{B}}$$

Цветовая коррекция

- Изменение цветового баланса
 - Компенсация:
 - Неверного цветовосприятия камеры
 - Цветного освещения
- Ряд алгоритмов рассмотрели на предыдущей лекции



Цветовая коррекция изображений

- Растворение контрастности (“autolevels”)
 - Идея – растворить интенсивности по каждому из каналов на весь диапазон;
- Метод:
 - Найти минимум, максимум по каждому из каналов:

$$R_{\min}, R_{\max}, G_{\min}, G_{\max}, B_{\min}, B_{\max}$$

- Преобразовать интенсивности:

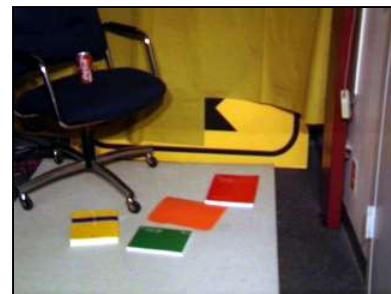
$$(R - R_{\min}) * \frac{(255 - 0)}{(R_{\max} - R_{\min})}; \quad (G - G_{\min}) * \frac{(255 - 0)}{(G_{\max} - G_{\min})};$$

$$(B - B_{\min}) * \frac{(255 - 0)}{(B_{\max} - B_{\min})};$$

Растяжение контрастности



Растяжение контрастности



Шумоподавление

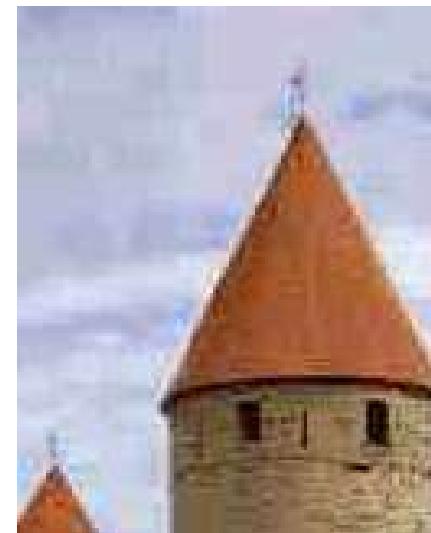
- Причины возникновения шума:
 - Несовершенство измерительных приборов
 - Хранение и передача изображений с потерей данных



Шум фотоаппарата



Сильное сжатие JPEG



Цель: подавление шума

Пусть дана камера и статичная сцена, требуется подавить шум.



Простейший вариант: усреднить несколько кадров

Усреднение

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

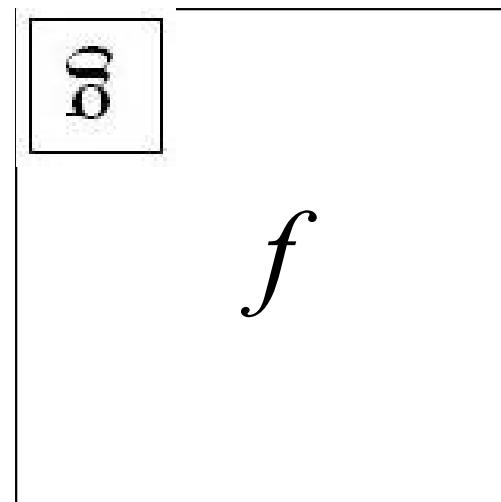
$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

“box filter”

Определение свертки

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

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



- Соглашение: ядро “перевернуто”
- scipy imfilter

Source: F. Durand

Основные свойства

- **Линейность:** $\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$
- **Инвариантность к сдвигу:** не зависит от сдвига пикселя: $\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$
- **Теория:** любой линейный оператор, инвариантный к сдвигу, может быть записан в виде свертки

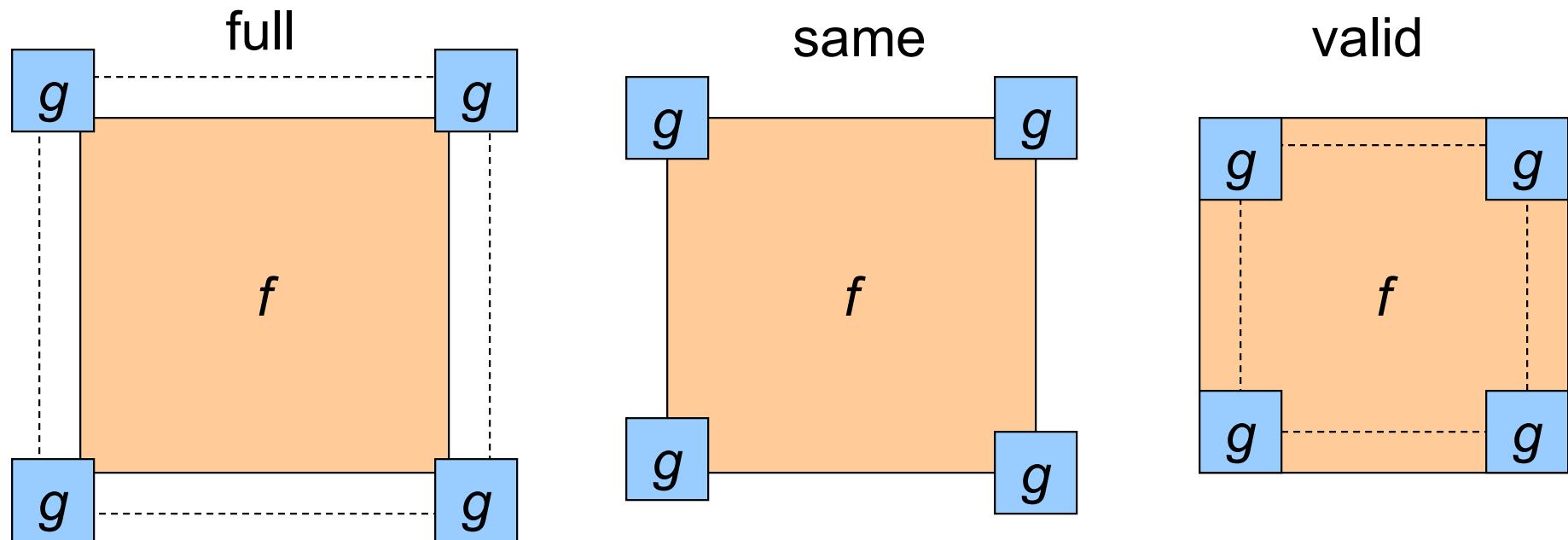
Свойства

- 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: $e = [..., 0, 0, 1, 0, 0, ...]$,
$$a * e = a$$

Детали реализации

Размер результирующего изображения?

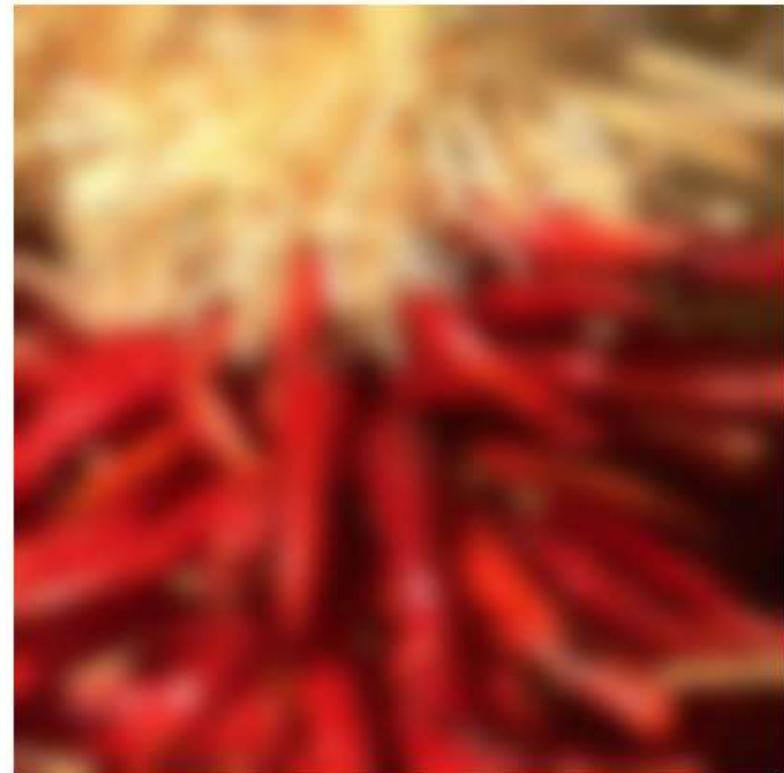
- MATLAB: `filter2(g, f, shape)`
 - `shape = 'full'`: output size is sum of sizes of f and g
 - `shape = 'same'`: output size is same as f
 - `shape = 'valid'`: output size is difference of sizes of f and g



Детали реализации

Как происходит фильтрация по краям?

- Окно фильтра выходит за границы изображения
- Необходимо экстраполировать изображение
- Варианты:
 - clip filter (black)
 - wrap around
 - copy edge
 - reflect across edge



Source: S. Marschner

Простейшие фильтры



0	0	0
0	1	0
0	0	0

?

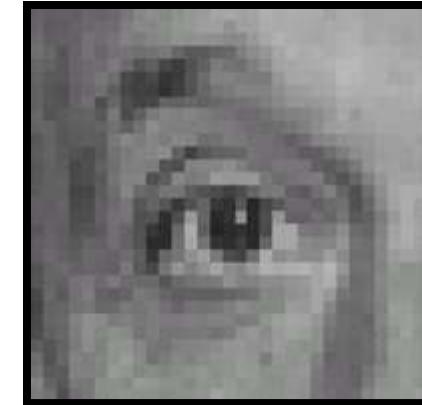
Original

Простейшие фильтры



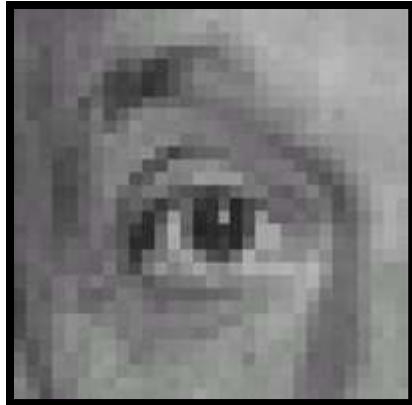
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Простейшие фильтры

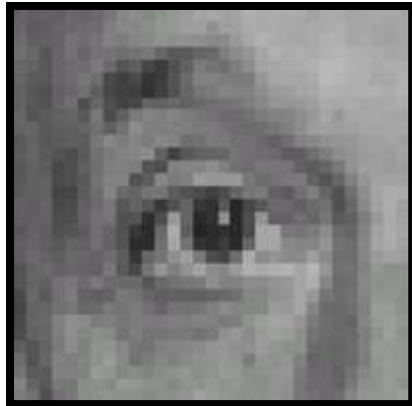


0	0	0
0	0	1
0	0	0

?

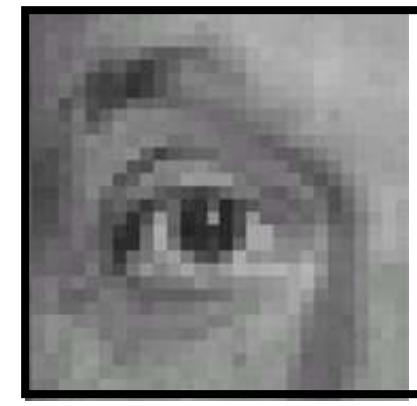
Original

Простейшие фильтры



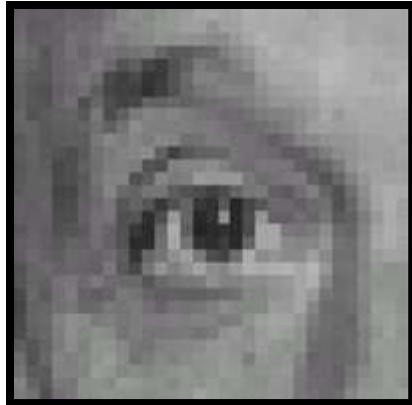
Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Простейшие фильтры



$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

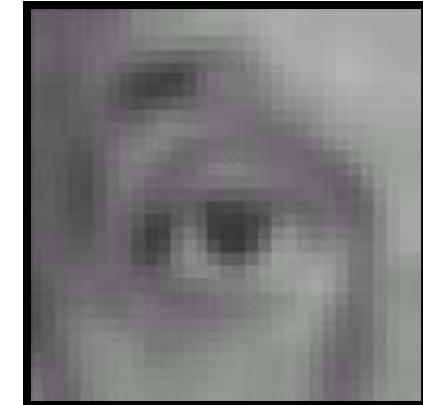
?

Простейшие фильтры



Original

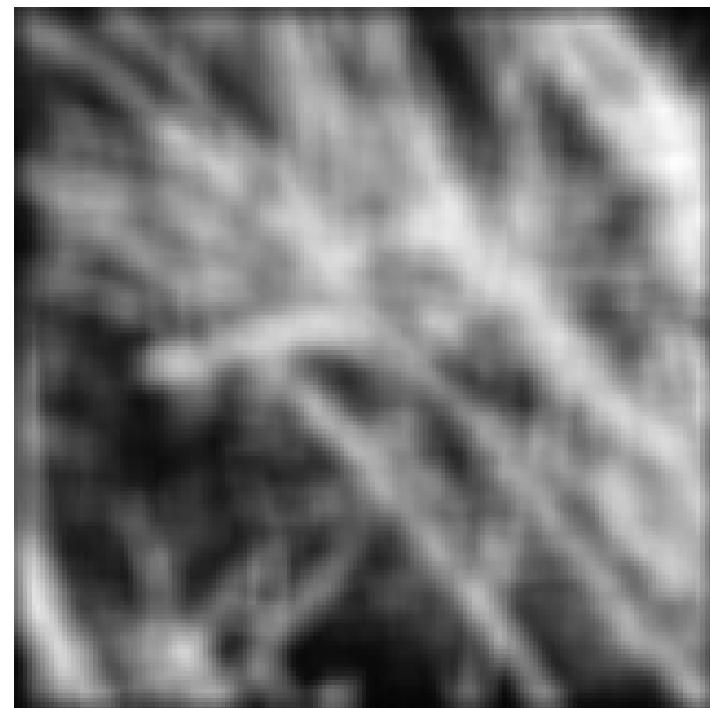
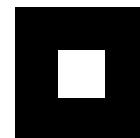
$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



Blur (with a
box filter)

Сглаживание с box-фильтром

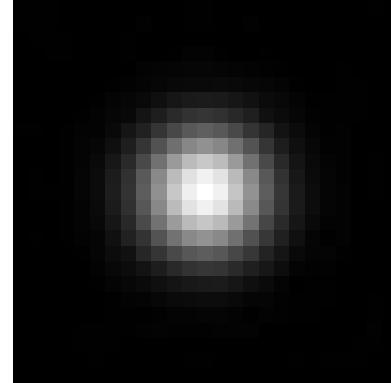
- Результат сглаживания с помощью усреднения отличается от разфокусированного изображения
- Точка света, наблюданная с разфокусированного объектива, выглядит как кружок света, а усреднение дает квадратик



Source: D. Forsyth

Сглаживание

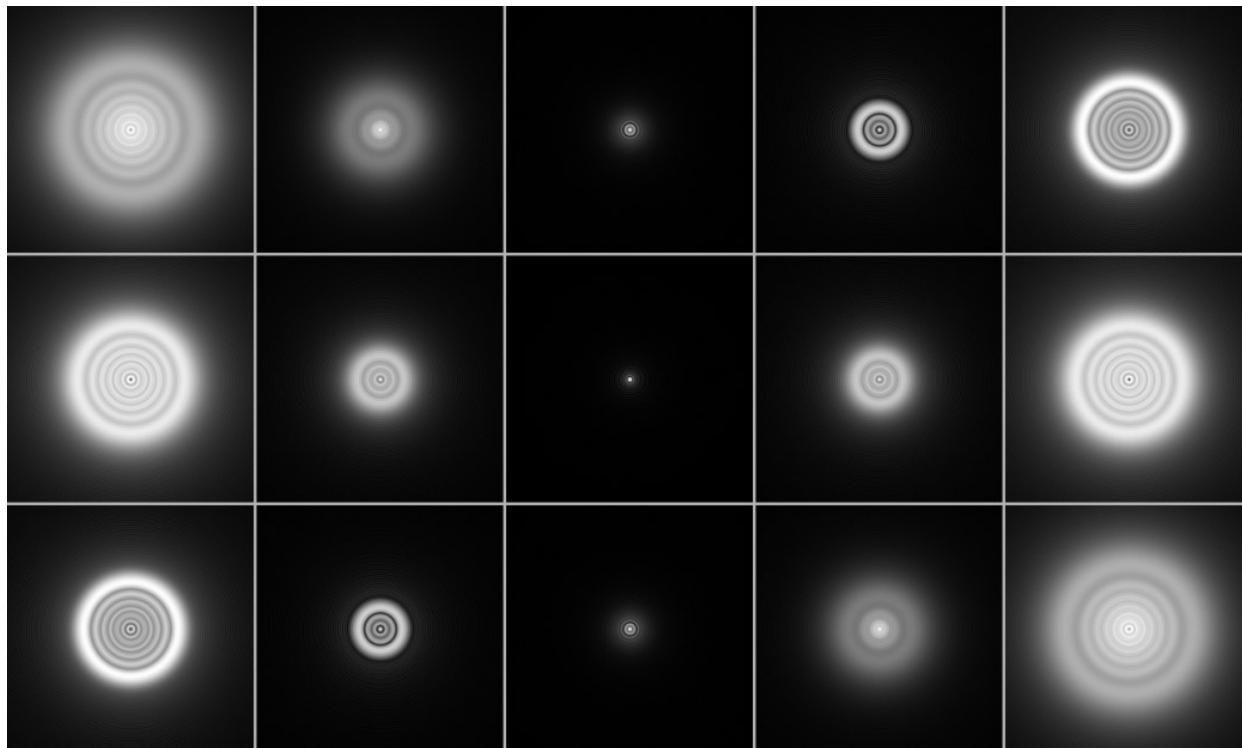
- Точка света, наблюдаемая с расфокусированного объектива, выглядит как кружок света, а усреднение дает квадратик
- Другой способ: взвешивает вклад пикселей по окрестности с учетом близости к центру:



“fuzzy blob”

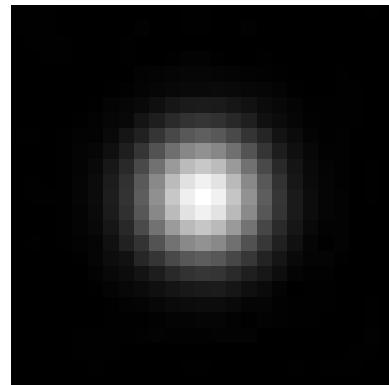
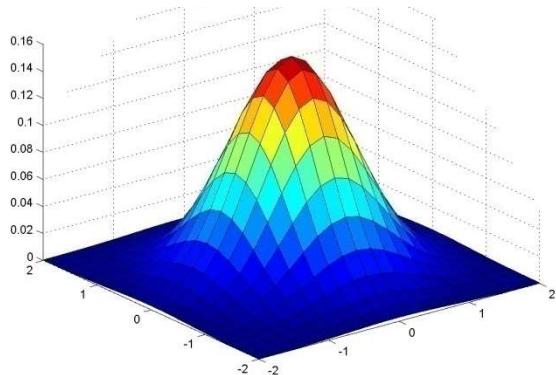
Point Spread Function (PSF)

- Point spread function (PSF) – отклик оптической системы на точечный источник света (объект)



Ядро фильтра гаусса

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



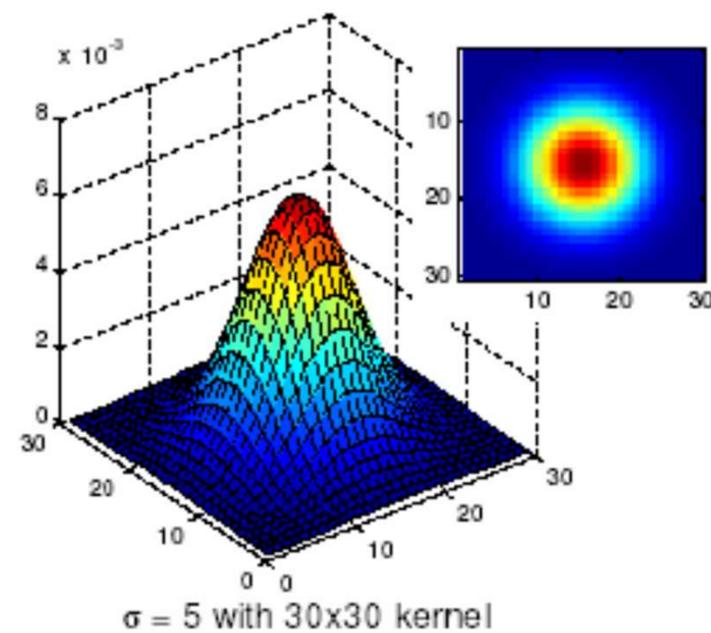
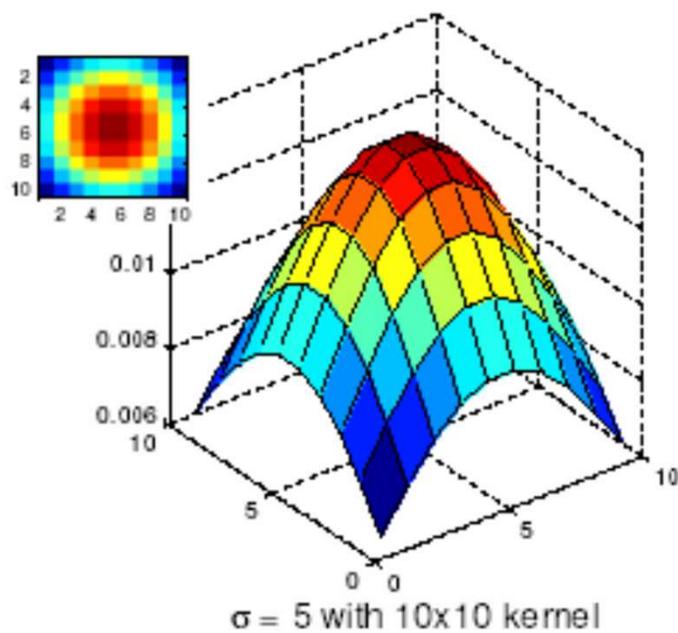
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$5 \times 5, \sigma = 1$

Source: C. Rasmussen

Выбор размера ядра

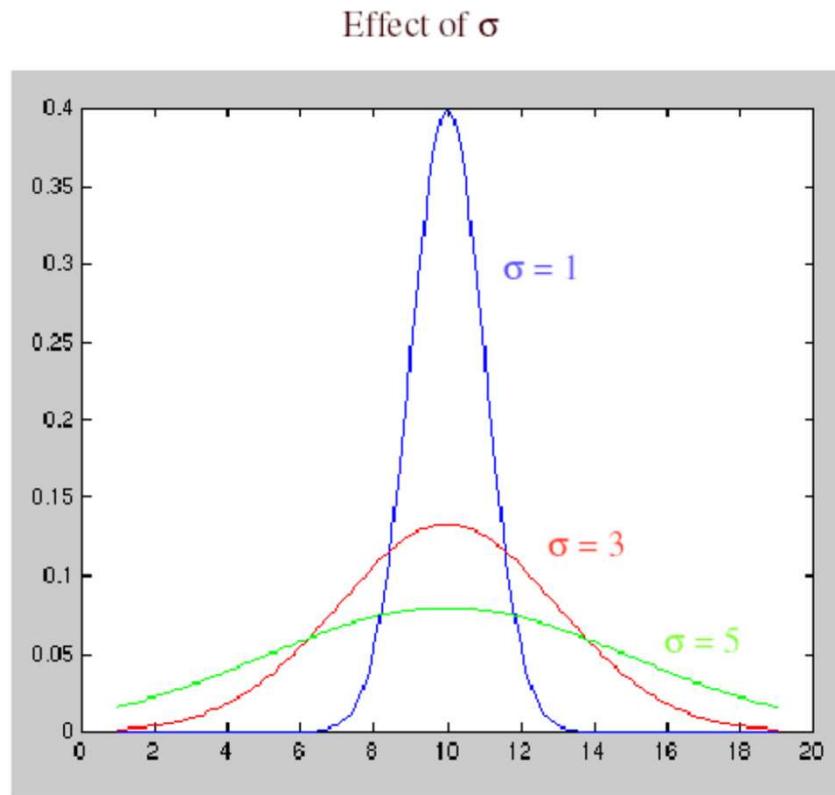
- Размер ядра дискретного фильтра ограничен



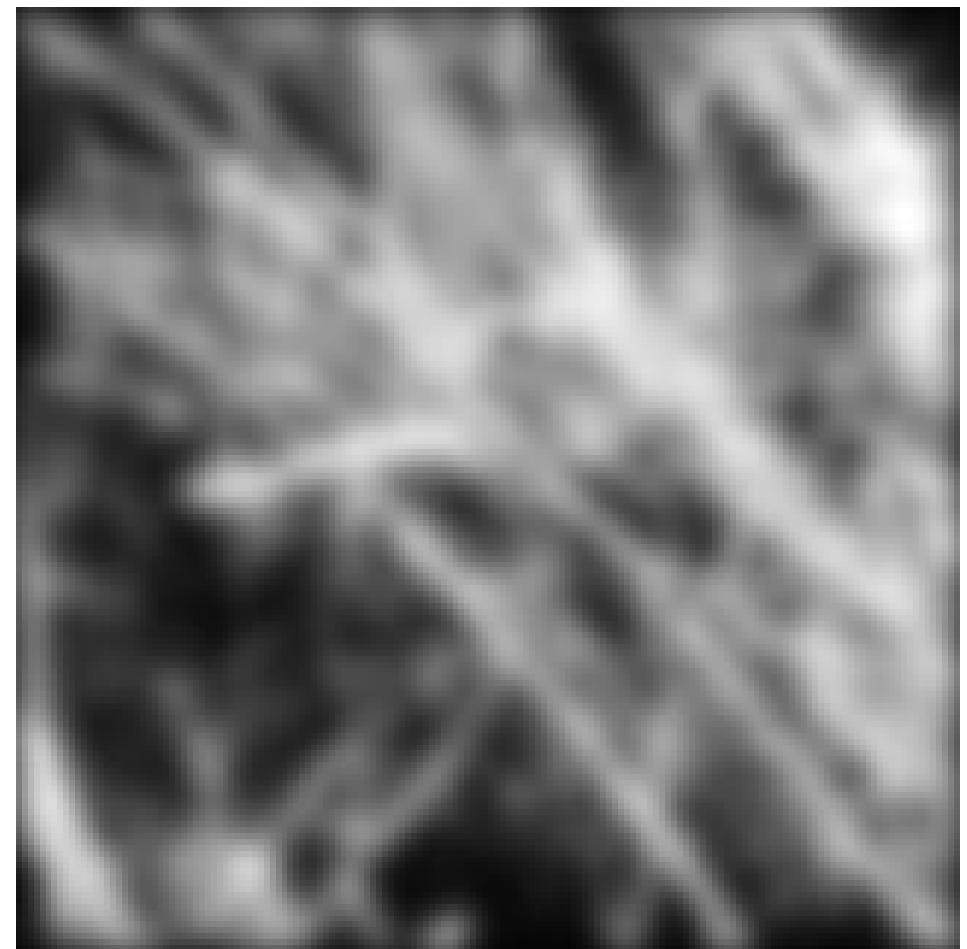
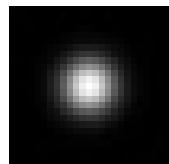
Source: K. Grauman

Выбор размера ядра

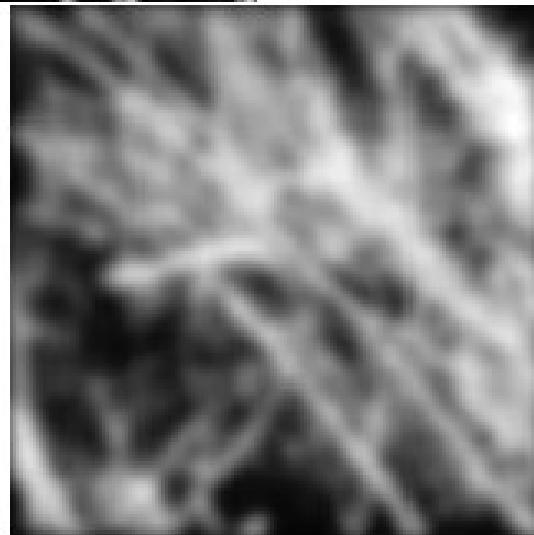
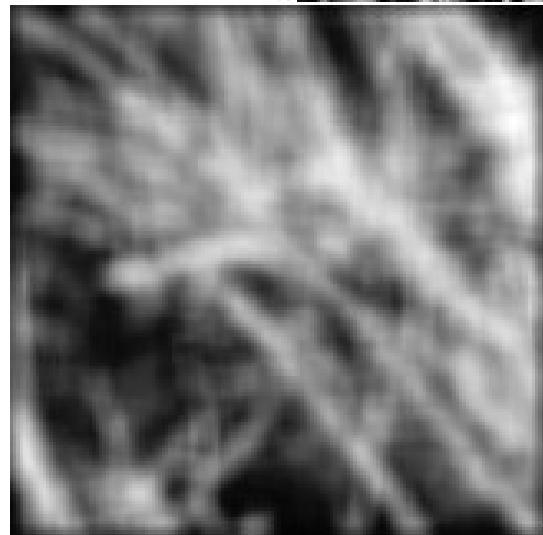
- Эмпирика: полуразмер фильтра равен 3σ



Сглаживание фильтром гаусса



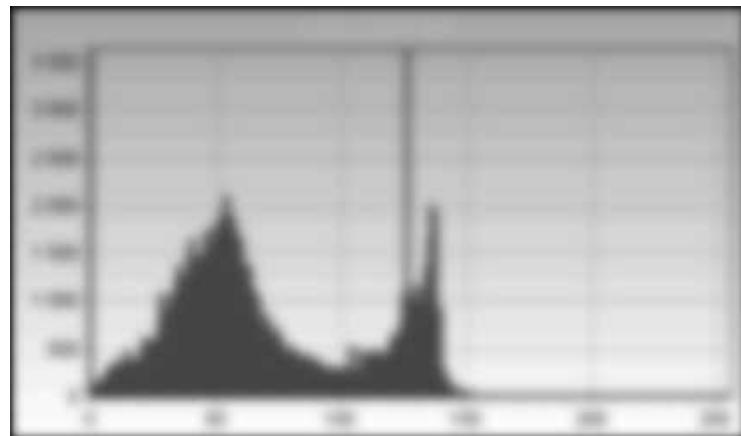
Сравнение



Свойства фильтра Гаусса

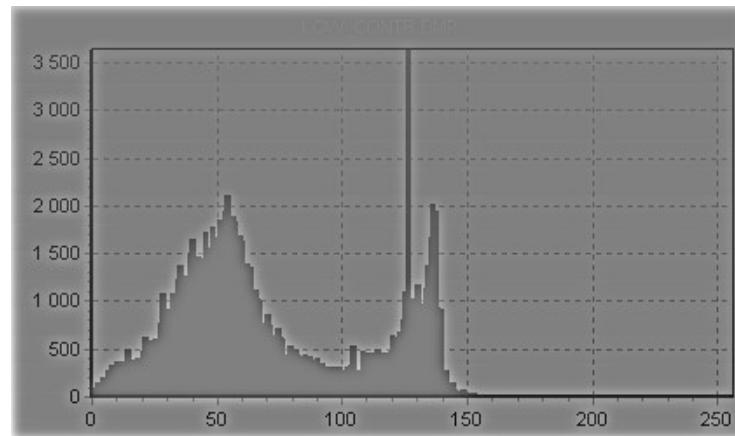
- Свертка с сами собой дает тоже фильтр гаусса
 - Сглаживание несколько раз фильтром с маленьким ядром дает результат, аналогичный свертке с большим ядром
 - Свертка 2 раза с фильтром радиуса σ дает тот же результат, что с фильтром радиуса $\sigma\sqrt{2}$

Маленькая экскурсия к Фурье

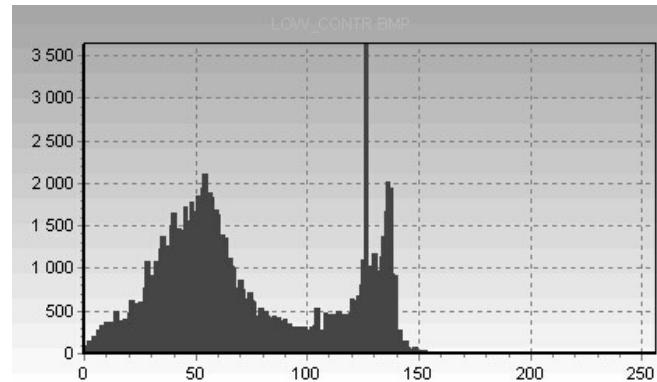


Низкие частоты

+

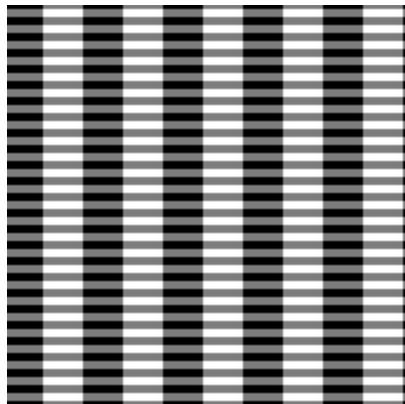


Высокие частоты

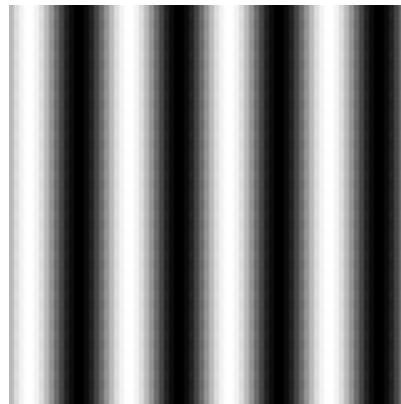


Фильтр Гаусса (gaussian blurring)

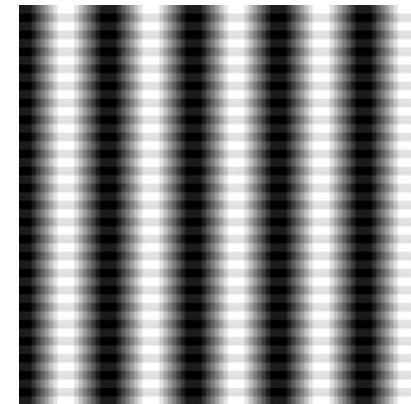
Результат свертки фильтром гаусса и усреднения



Исходное изображение



Фильтр Гаусса с
Sigma = 4



Усреднение по 49
пикселям (7x7)

Важное свойство фильтра Гаусса – он по сути является фильтром низких частот.

Сепарабельность

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

Сепарабельное ядро

Раскладывается в произведение двух одномерных фильтром гаусса

Пример

2D свертка

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} * \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix}$$

Фильтр раскладывается
в произведение двух 1D
фильтров:

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} = \begin{matrix} 1 \\ 2 \\ 1 \end{matrix} \times \begin{matrix} 1 & 2 & 1 \end{matrix}$$

Свертка по строкам:

$$\begin{matrix} 1 & 2 & 1 \end{matrix} * \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix} = \begin{matrix} 11 & & \\ 18 & & \\ 18 & & \end{matrix}$$

Затем свертка по столбцу:

Сепарабельность

- Почему сепарабельность полезна на практике?

Виды шума



Original



Salt and pepper noise



Impulse noise



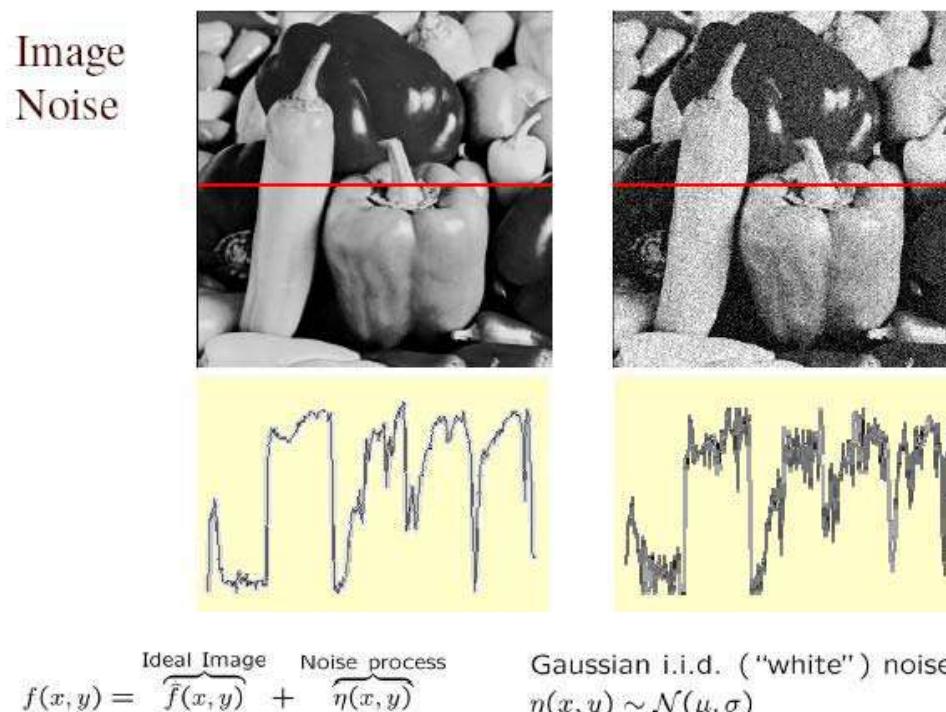
Gaussian noise

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

Source: S. Seitz

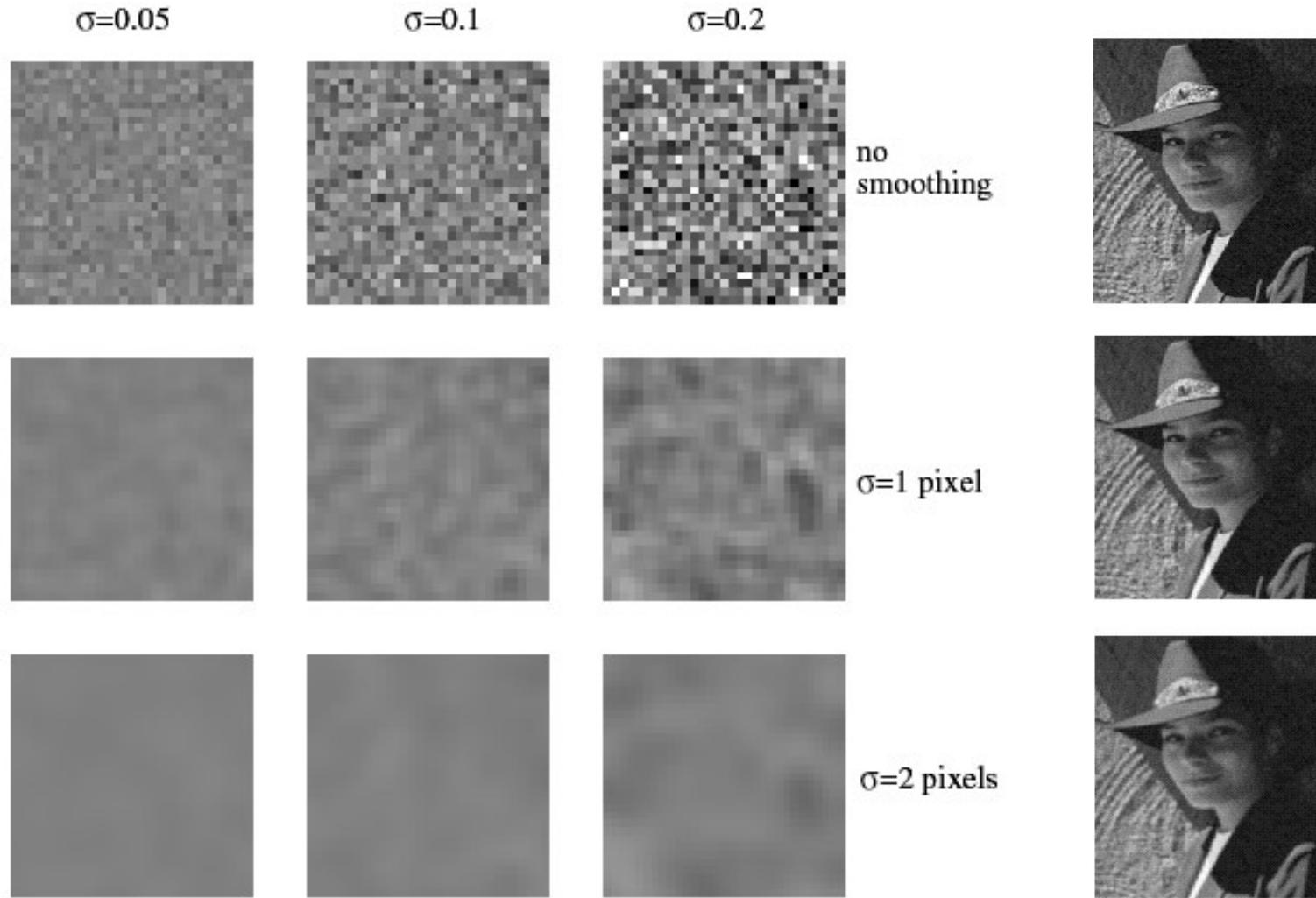
Гауссов шум

- Мат.модель: сумма множества независимых факторов
- Подходит при маленьких дисперсиях
- Предположения: независимость, нулевое матожидание



Source: M. Hebert

Подавление гауссова шума



Сглаживание фильтрами большого радиуса подавляет шум, но размывает изображение

Подавление шума «соль и перец»

3x3



5x5



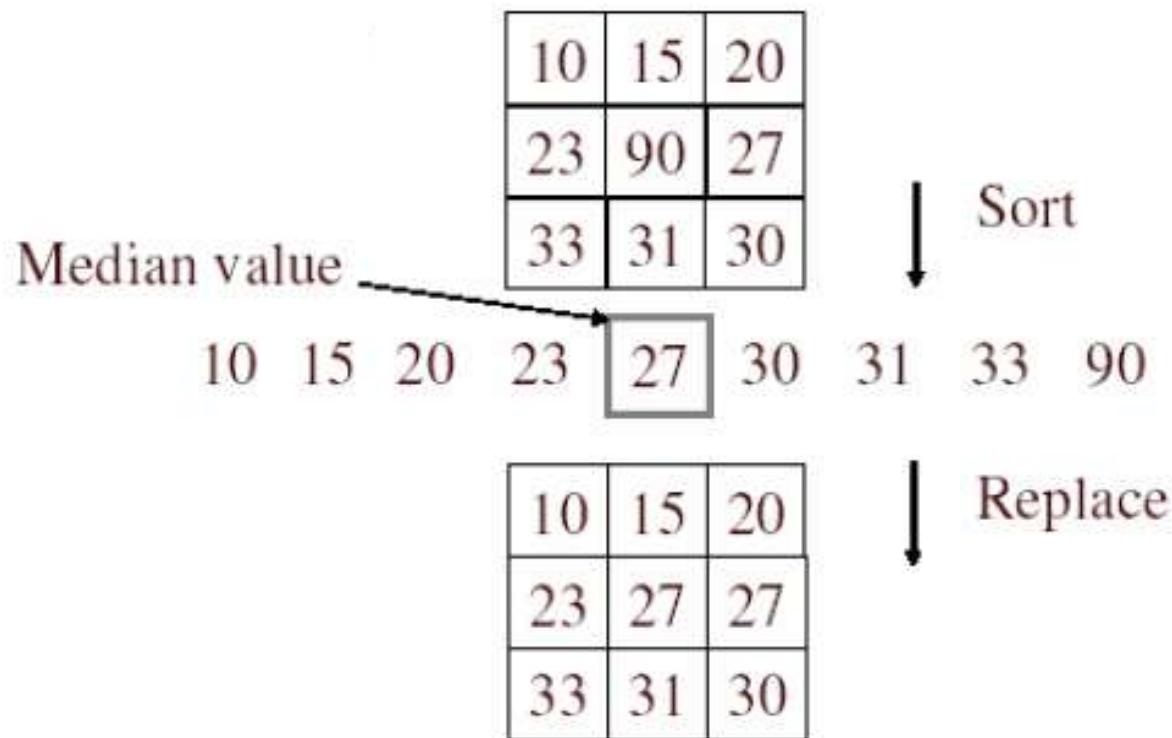
7x7



Чем результат плох?

Медианный фильтр

- Выбор медианы из выборки пикселей по окрестности данного

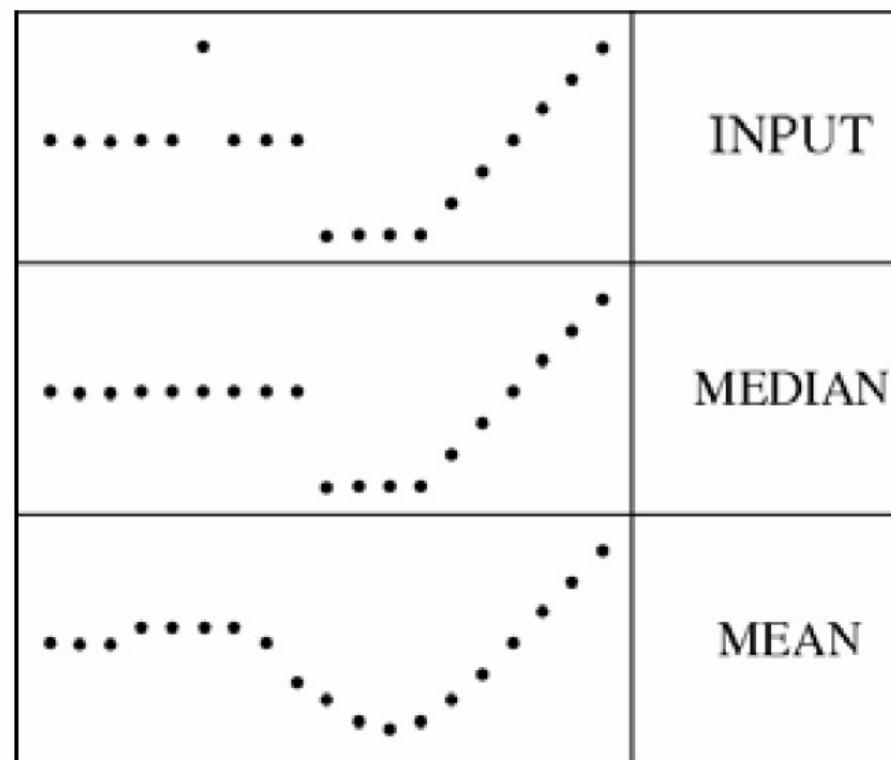


- Является ли фильтр линейным?

Медианный фильтр

- В чем преимущество медианного фильтра перед фильтром гаусса?
 - Устойчивость к выбросам (outliers)

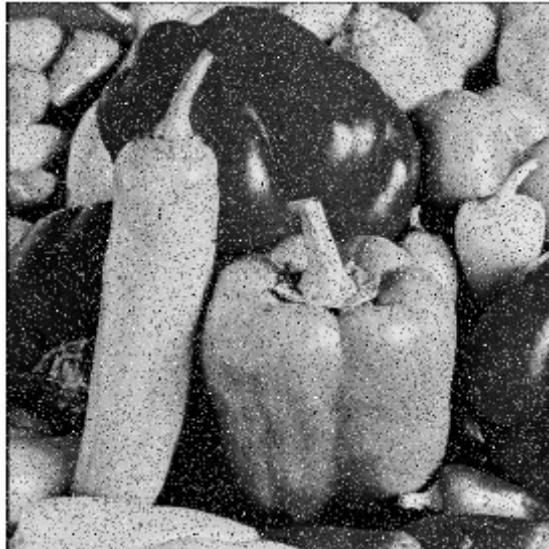
filters have width 5 :



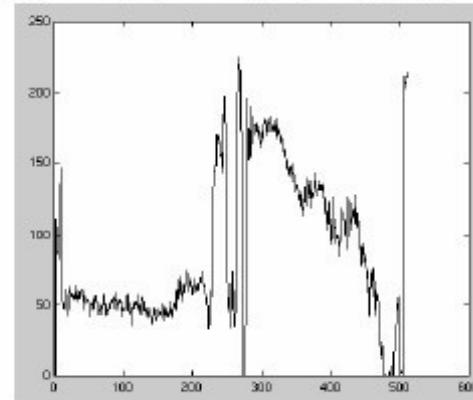
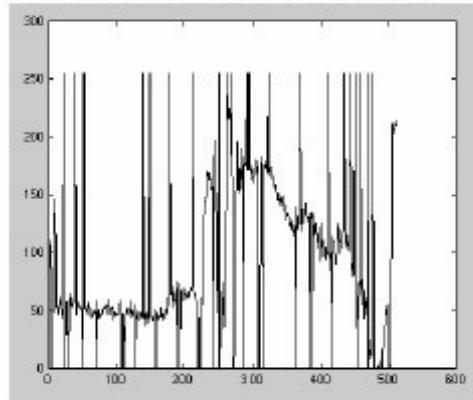
Source: K. Grauman

Медианный фильтр

Salt-and-pepper noise



Median filtered

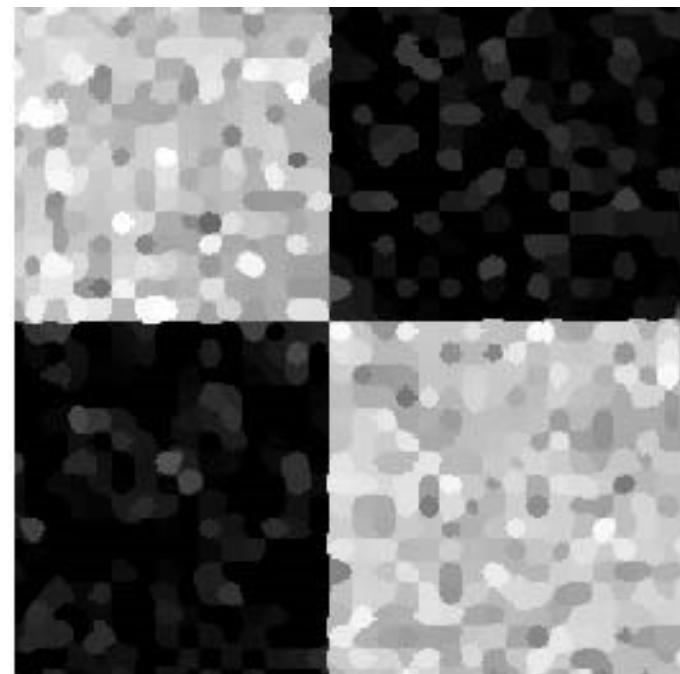
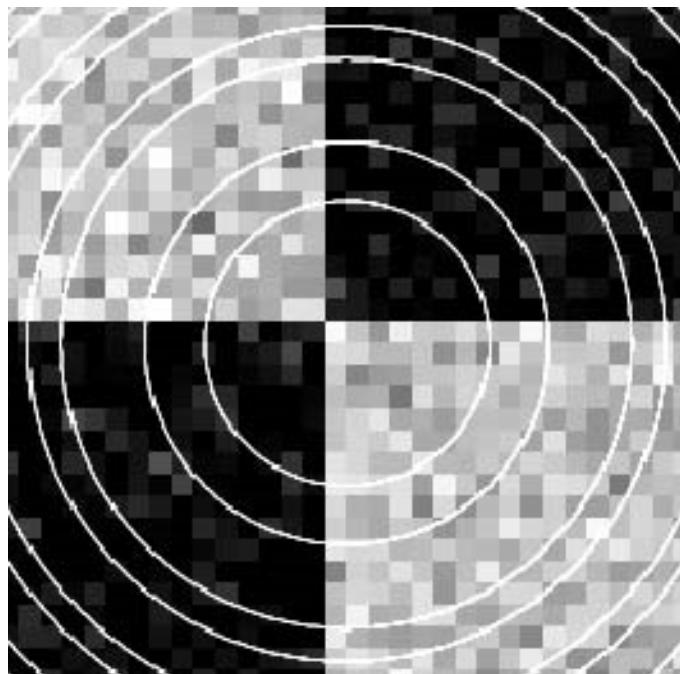


MATLAB: `medfilt2(image, [h w])`

Source: M. Hebert

Медианный фильтр

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



Сравнение фильтров

3x3



5x5



7x7



Гауссов

Медианный



Повышение резкости

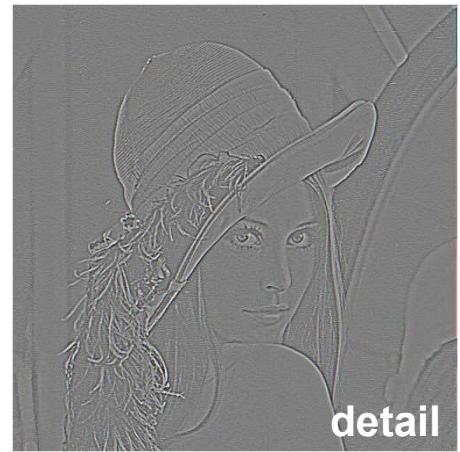
Что теряется при сглаживании?



-



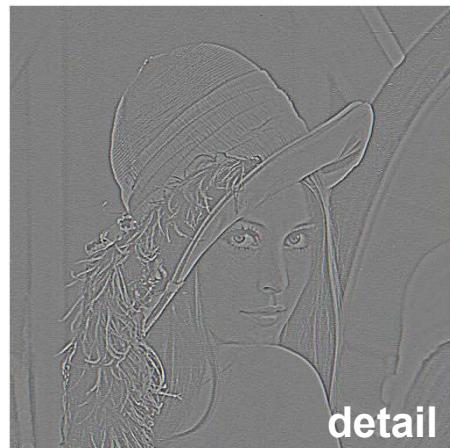
=



Добавим дополнительно высокие частоты:



+ a



=



Фильтр Unsharp



Пример

Ядро
свертки



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

Компенсация разности освещения

Пример

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactful
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactful
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst

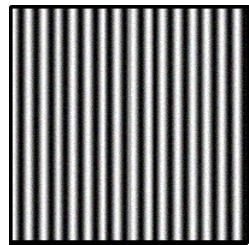
Компенсация разности освещения

Идея:

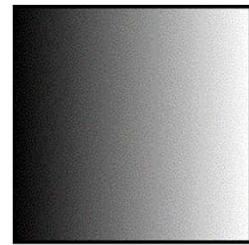
Формирование изображения:

$$I(i, j) = l(i, j) \cdot r(i, j)$$

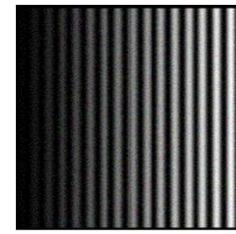
Плавные изменения яркости относятся к освещению,
резкие - к объектам.



объект $r(i, j)$



освещение $l(i, j)$

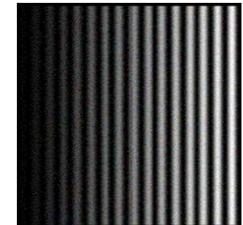


Изображение
освещенного
объекта $I(i, j)$

Выравнивание освещения

- Алгоритм Single scale retinex (SSR)
 - Получить приближенное изображение освещения путем низочастотной фильтрации

$$\hat{l}(i, j) = G * I(i, j)$$

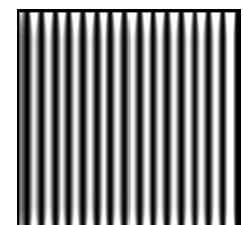


- Восстановить изображение по формуле

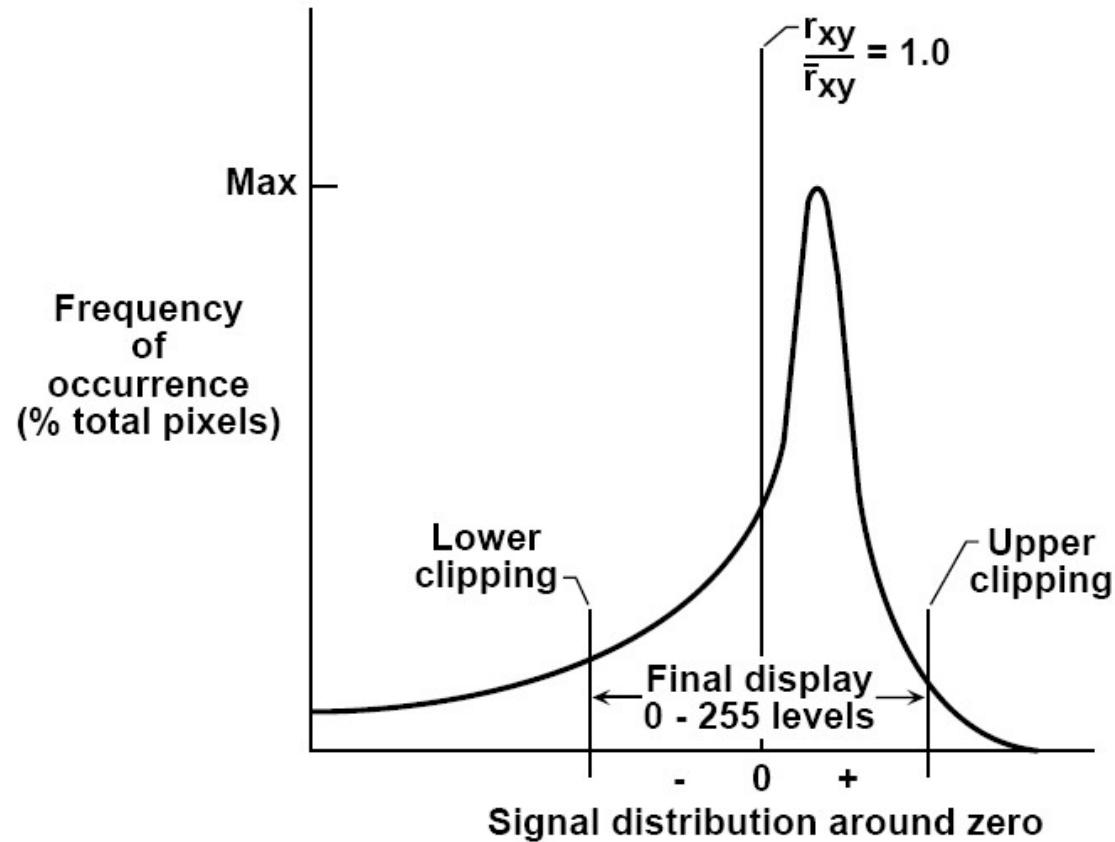
$$\hat{r}(i, j) = \log \frac{I(i, j)}{\hat{l}(i, j)}$$



$$\hat{r}(i, j) = \log I(i, j) - \log \hat{l}(i, j)$$

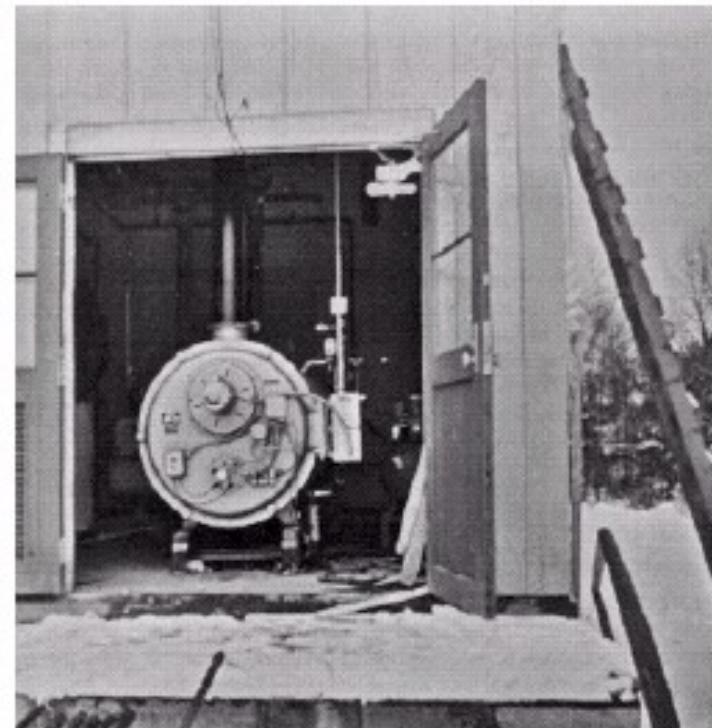


Обрезание по порогу



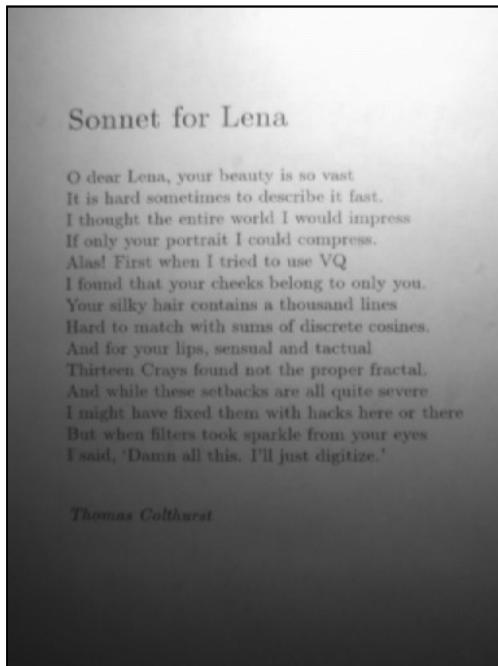
Выравнивание освещения

Пример

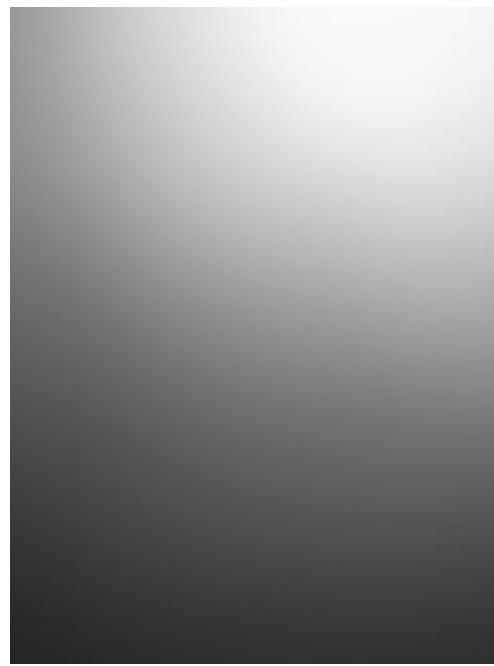


Компенсация разности освещения

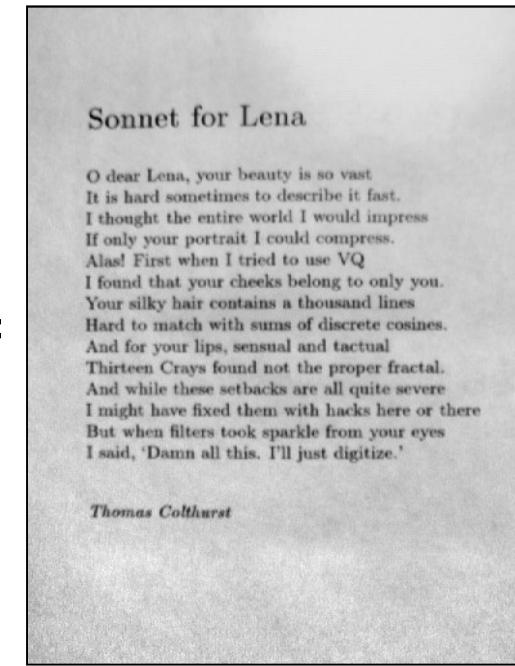
Пример



/



=



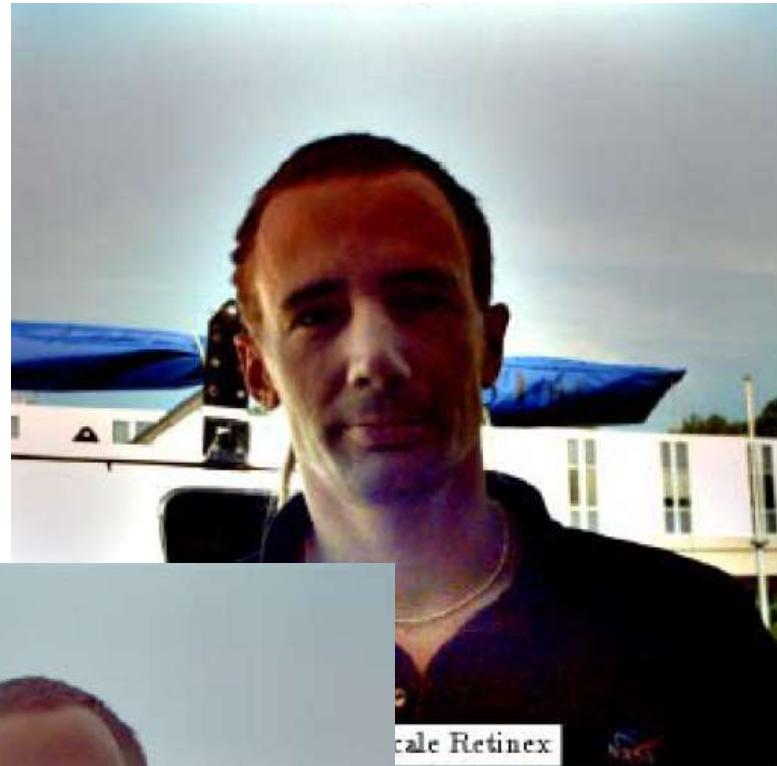
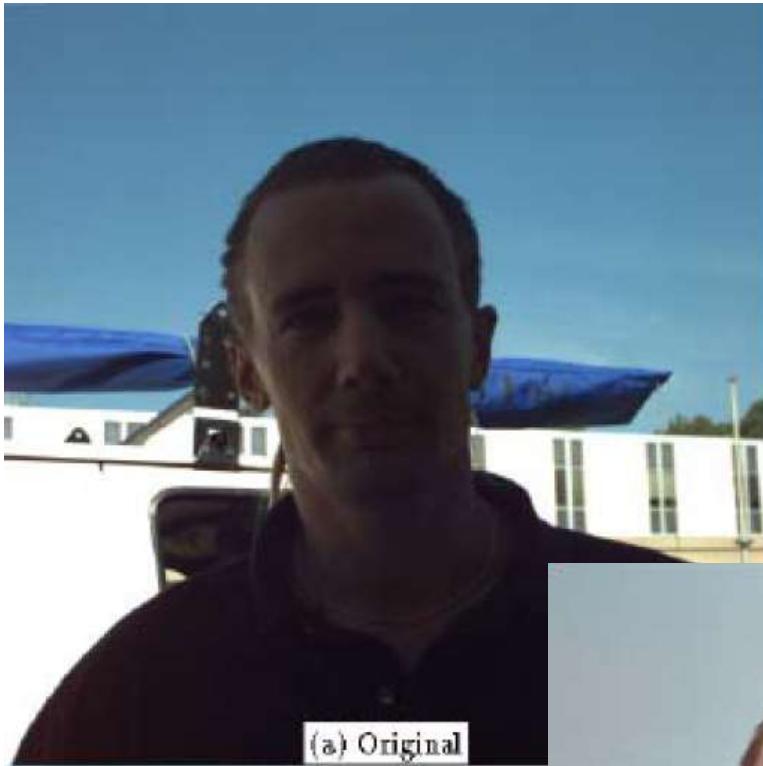
Gauss 14.7 пикселей

Многомасштабный вариант

$$\hat{r}(i, j) = \sum_k w_k (\log I(i, j) - \log g_k(i, j) \cdot I(i, j))$$

- Чаще всего выбирают 3 масштаба
- Веса одинаковые (1/3)

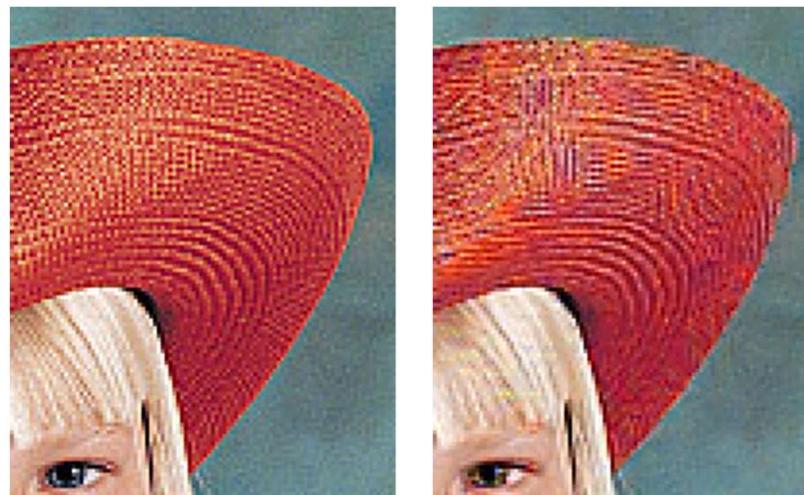
Multi-scale retinex



Source by Z.Rahman et.al.

Метрики качества

- Как измерить похожесть двух изображений?
 - Для оценки качества подавления шума, например



исходное изображение искаженное изображение

Метрики качества

- Среднеквадратичная ошибка (MSE)

$$MSE = \frac{1}{N} \sum_{i=1}^N (x_i - y_i)^2 \quad N - \text{число пикселей}$$

- Пиковое отношение сигнал/шум (PSNR)

$$PSNR_{dB} = 10 \lg \frac{M^2}{MSE} \quad M - \text{максимальное значение пикселя}$$

Метрики качества

- PSNR и MSE не учитывают особенности человеческого восприятия!



Оригинал

Далее будут использованы рисунки из статьи
Wang, Bovik, Lu "WHY IS IMAGE QUALITY ASSESSMENT SO DIFFICULT?"

Метрики качества

- У этих изображений одинаковые PSNR с оригиналом (примерно 25 dB)



Повышена контрастность

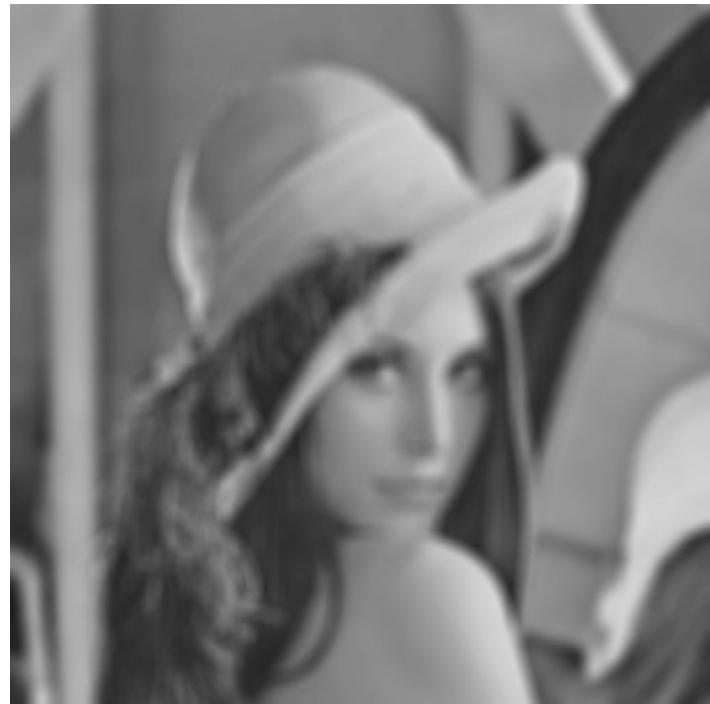
Добавлен белый гауссов шум

Метрики качества

- И у этих – тоже примерно 25 dB!



Добавлен импульсный шум



Размытие

Метрики качества

- И у этого – тоже!



Артефакт блочности после JPEG

Метрики качества

- Вывод: PSNR не всегда отражает реальный видимый уровень искажений.
- Как улучшить?

- HVS models
(human visual system)
- Использовать функцию чувствительности глаза к различным частотам (CSF)
 - Использовать свойство маскировки
 - Использовать равномерные к восприятию цветовые пространства (CIE Lab, CIEDE2000)

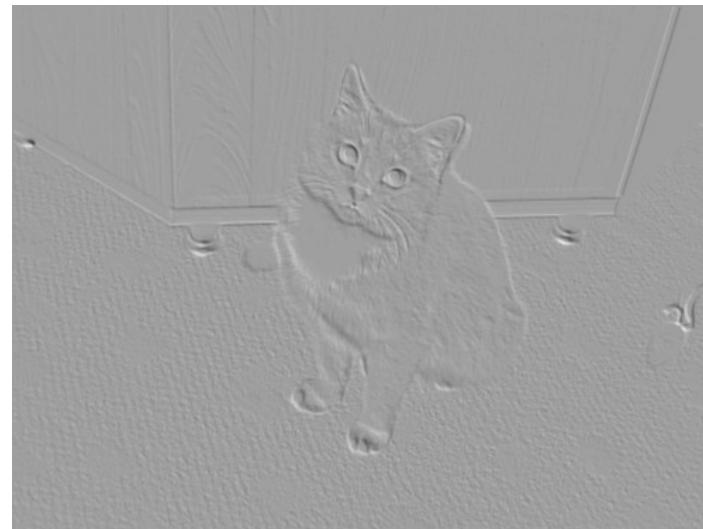
Спецэффекты

- Рассмотрим
 - Тиснение
 - Негатив
 - «Светящиеся» края
 - Геометрические эффекты
 - Перенос/поворот
 - Искажение
 - «Эффект стекла»

Тиснение

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

Фильтр + сдвиг яркости, нормировка...



Цифровой негатив



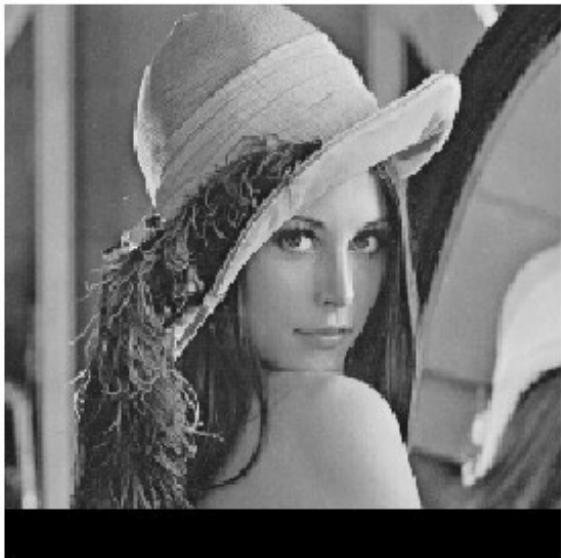
$$R' = 255 - R; \quad G' = 255 - G; \quad B' = 255 - B;$$

Светящиеся края



Медианный фильтра + выделение краев + фильтр
«максимума»

Перенос/поворот



Перенос:

$$x(k; l) = k + 50; y(k; l) = l;$$

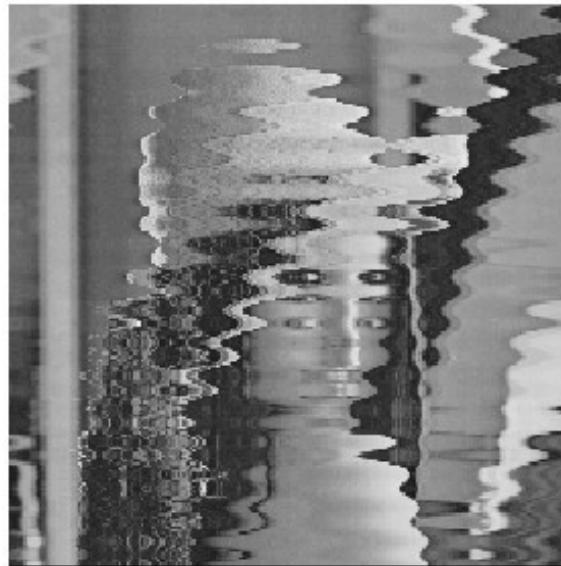
Поворот:

$$x(k; l) = (k \cdot x_0) \cos(\mu) + (l \cdot y_0) \sin(\mu) + x_0;$$

$$y(k; l) = -(k \cdot x_0) \sin(\mu) + (l \cdot y_0) \cos(\mu) + y_0;$$

$$x_0 = y_0 = 256.5 \text{ (центр поворота)}, \mu = \pi/6$$

«Волны»



Волны 1:

$$x(k; l) = k + 20\sin(2\pi l / 128); y(k; l) = l;$$

Волны 2:

$$x(k; l) = k + 20\sin(2\pi k / 30); y(k; l) = l;$$

«Эффект стекла»


$$x(k; l) = k + (\text{rand}(1, 1) - 0.5) * 10;$$
$$y(k; l) = l + (\text{rand}(1, 1) - 0.5) * 10;$$

Filters with `scipy.ndimage`

- `convolve`
- `convolve1d`
- `gaussian_filter`
- `gaussian_filter1d`
- `generic_filter`
- `generic_filter1d`
- `maximum_filter`
- `maximum_filter1d`
- `median_filter`
- `minimum_filter`
- `minimum_filter1d`
- `percentile_filter`

Multidimensional convolution.

```
scipy.ndimage.convolve(input, weights, output=None,  
mode='reflect', cval=0.0, origin=0)
```

mode : {'reflect', 'constant', 'nearest', 'mirror', 'wrap'}, optional

The mode parameter determines how the array borders are handled, where cval is the value when mode is equal to 'constant'. Default is 'reflect'

cval : scalar, optional

Value to fill past edges of input if mode is 'constant'. Default is 0.0

origin : scalar, optional

The origin parameter controls the placement of the filter. Default 0.0.

Each value in result is $C_i = \sum_j I_{i+k-j} W_j$, where W is the *weights* kernel, j is the n-D spatial index over W , I is the *input* and k is the coordinate of the center of W , specified by *origin* in the input parameters.

Calculate a one-dimensional convolution along the given axis.

```
scipy.ndimage.convolve1d(input, weights, axis=-1,  
output=None, mode='reflect', cval=0.0, origin=0)
```

mode : {'reflect', 'constant', 'nearest', 'mirror', 'wrap'}, optional

The mode parameter determines how the array borders are handled, where cval is the value when mode is equal to 'constant'. Default is 'reflect'

cval : scalar, optional

Value to fill past edges of input if mode is 'constant'. Default is 0.0

origin : scalar, optional

The origin parameter controls the placement of the filter. Default 0.0.

Multidimensional Gaussian filter.

```
scipy.ndimage.gaussian_filter(input, sigma, order=0,  
output=None, mode='reflect', cval=0.0, truncate=4.0)
```

The multidimensional filter is implemented as a sequence of one-dimensional convolution filters. The intermediate arrays are stored in the same data type as the output. Therefore, for output types with a limited precision, the results may be imprecise because intermediate results may be stored with insufficient precision.

One-dimensional Gaussian filter.

```
scipy.ndimage.gaussian_filter1d(input, sigma, axis=-1,  
order=0, output=None, mode='reflect', cval=0.0,  
truncate=4.0)
```

order : {0, 1, 2, 3}, optional

An order of 0 corresponds to convolution with a Gaussian kernel.

An order of 1, 2, or 3 corresponds to convolution with the first, second or third derivatives of a Gaussian.

mode : {'reflect', 'constant', 'nearest', 'mirror', 'wrap'}, optional

The mode parameter determines how the array borders are handled, where cval is the value when mode is equal to 'constant'. Default is 'reflect'

cval : scalar, optional

Value to fill past edges of input if mode is 'constant'. Default is 0.0

truncate : float, optional

Truncate the filter at this many standard deviations. Default is 4.0.

scipy.ndimage.generic_filter

```
scipy.ndimage.generic_filter1d(input, function,  
filter_size, axis=-1, output=None, mode='reflect',  
cval=0.0, origin=0, extra_arguments=(),  
extra_keywords=None)
```

generic_filter1d iterates over the lines of the array, calling the given function at each line. The arguments of the line are the input line, and the output line. The input and output lines are 1D double arrays. The input line is extended appropriately according to the filter size and origin. The output line must be modified in-place with the result.

Calculate a one-dimensional maximum filter along the given axis.

This function implements the MAXLIST algorithm [R151], as described by Richard Harter [R152], and has a guaranteed $O(n)$ performance, n being the input length, regardless of filter size.

Calculates a multidimensional median filter.

```
scipy.ndimage.median_filter(input, size=None,  
footprint=None, output=None, mode='reflect', cval=0.0,  
origin=0)
```

footprint : array, optional

Either size or footprint must be defined. size gives the shape that is taken from the input array, at every element position, to define the input to the filter function. footprint is a boolean array that specifies (implicitly) a shape, but also which of the elements within this shape will get passed to the filter function. Thus size=(n,m) is equivalent to footprint=np.ones((n,m)). We adjust size to the number of dimensions of the input array, so that, if the input array is shape (10,10,10), and size is 2, then the actual size used is (2,2,2).

Calculate a one-dimensional
minimum filter along the given axis.

```
scipy.ndimage.minimum_filter1d(input, size, axis=-1,  
output=None, mode='reflect', cval=0.0, origin=0)
```

Calculates a multi-dimensional percentile filter.

```
scipy.ndimage.percentile_filter(input, percentile,  
size=None, footprint=None, output=None,  
mode='reflect', cval=0.0, origin=0)
```

percentile : scalar

The percentile parameter may be less than zero, i.e.,
percentile = -20 equals percentile = 80

Calculate the histogram of the values of an array, optionally at labels.

```
scipy.ndimage.histogram(input, min, max, bins,  
labels=None, index=None)
```

Histogram calculates the frequency of values in an array within bins determined by min, max, and bins. The labels and index keywords can limit the scope of the histogram to specified sub-regions within the array.

Others stats

[maximum](#)(input[, labels, index])

[maximum_position](#)(input[, labels, index])

[mean](#)(input[, labels, index])

[median](#)(input[, labels, index])

[minimum](#)(input[, labels, index])

[minimum_position](#)(input[, labels, index])

[standard_deviation](#)(input[, labels, index])

[sum](#)(input[, labels, index])

[variance](#)(input[, labels, index])

Calculate the maximum of the values of an array over labeled regions.

Find the positions of the maximums of the values of an array at labels.

Calculate the mean of the values of an array at labels.

Calculate the median of the values of an array over labeled regions.

Calculate the minimum of the values of an array over labeled regions.

Find the positions of the minimums of the values of an array at labels.

Calculate the standard deviation of the values of an n-D image array, optionally at specified sub-regions.

Calculate the sum of the values of the array.

Calculate the variance of the values of an n-D image array, optionally at specified sub-regions.

Shift an array.

```
scipy.ndimage.shift(input, shift, output=None, order=3,  
mode='constant', cval=0.0, prefilter=True)
```

The array is shifted using spline interpolation of the requested order. Points outside the boundaries of the input are filled according to the given mode.

Rotate an array.

```
scipy.ndimage.rotate(input, angle, axes=(1, 0),  
reshape=True, output=None, order=3, mode='constant',  
cval=0.0, prefilter=True)[source]
```

The array is rotated in the plane defined by the two axes given by the axes parameter using spline interpolation of the requested order.

На следующей лекции

- Старые-добрьи методы распознавания объектов
- Сопоставление шаблонов
- Выделение краёв
- Выделение контрастных объектов
- Геометрические и фотометрические инварианты