

Инженерная и компьютерная графика

6 семестр (диф.зачет)

Лектор:

Таранцев Игорь Геннадьевич
Доцент ФИТ НГУ, ИАиЭ, «СофтЛаб-НСК»

Создатели курса:

Дебелов Виктор Алексеевич

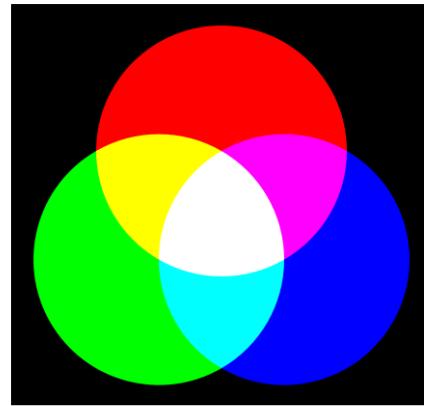
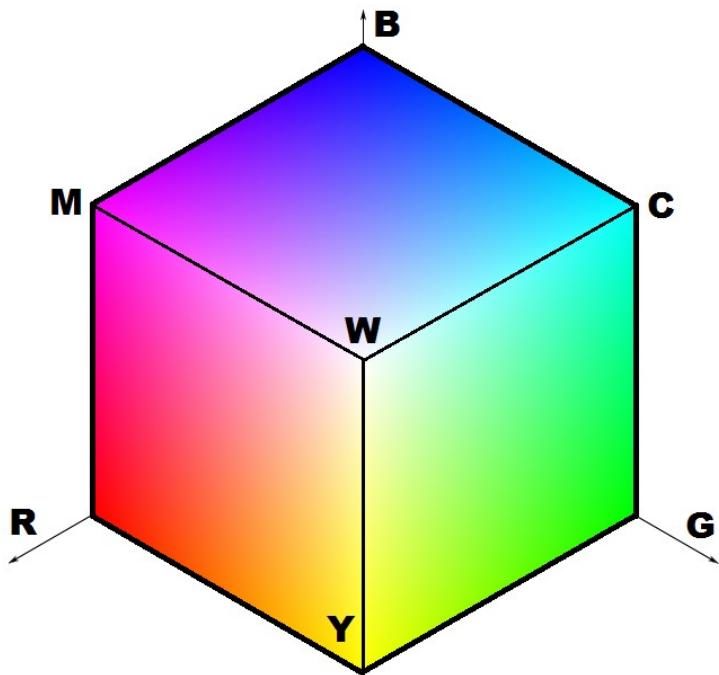
Валеев Тагир Фаридович

Козлов Дмитрий Сергеевич

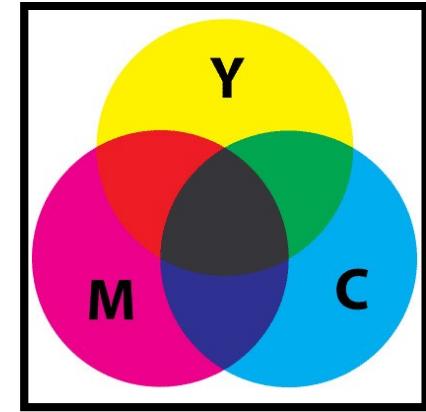
Лекция №2

Дизеринг.
Фильтрация изображений.

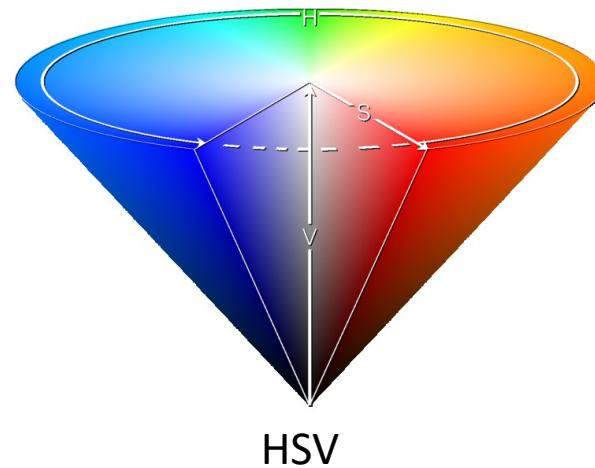
Цветовые модели



RGB



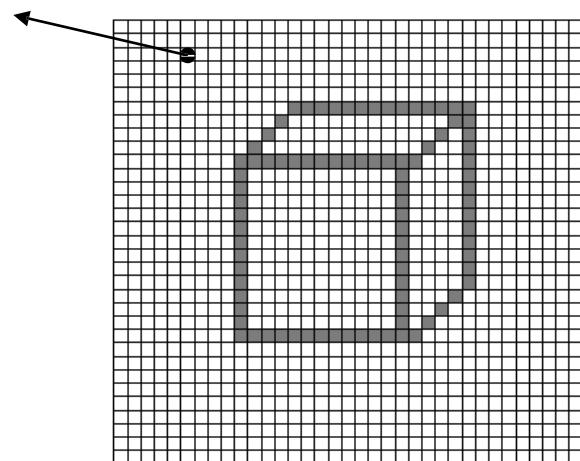
CMY



Дисплей и буфер кадра

Изображение – это растр = 2D массив пикселей

int FB[n][m] – frame buffer FB[5][2]



Типы пикселей:

- Черно-белый – 1 бит/пиксель
- Монохромные – 2, 4, 8, 12 бит
- Цветные – 2, 3, 4, 8, 15, 16, 18, 24, 32, 48, 96 б/п
- Супер – 96 б/п

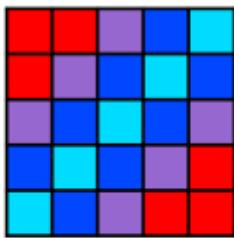
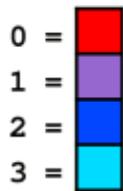
Индексный цвет

```
const int n=1<<depth;  
RGB palette[n]; // LUT
```

Цвет точки определяется как palette[FB[x][y]];

В ходу были 1, 2, 4, 6, 8, 12-битные палитры

0	0	1	2	3
0	1	2	3	2
1	2	3	2	1
2	3	2	1	0
3	2	1	0	0



1 бит: 2 цвета (чёрно/белый, чёрно/зелёный и т. д.)

2 бита: 4 цвета адаптер CGA(IBM, 1981, 16 Kb Video RAM)

4 бита: EGA(IBM, 1984, 64-256 Kb)/VGA (IBM, 1987, 256 Kb)

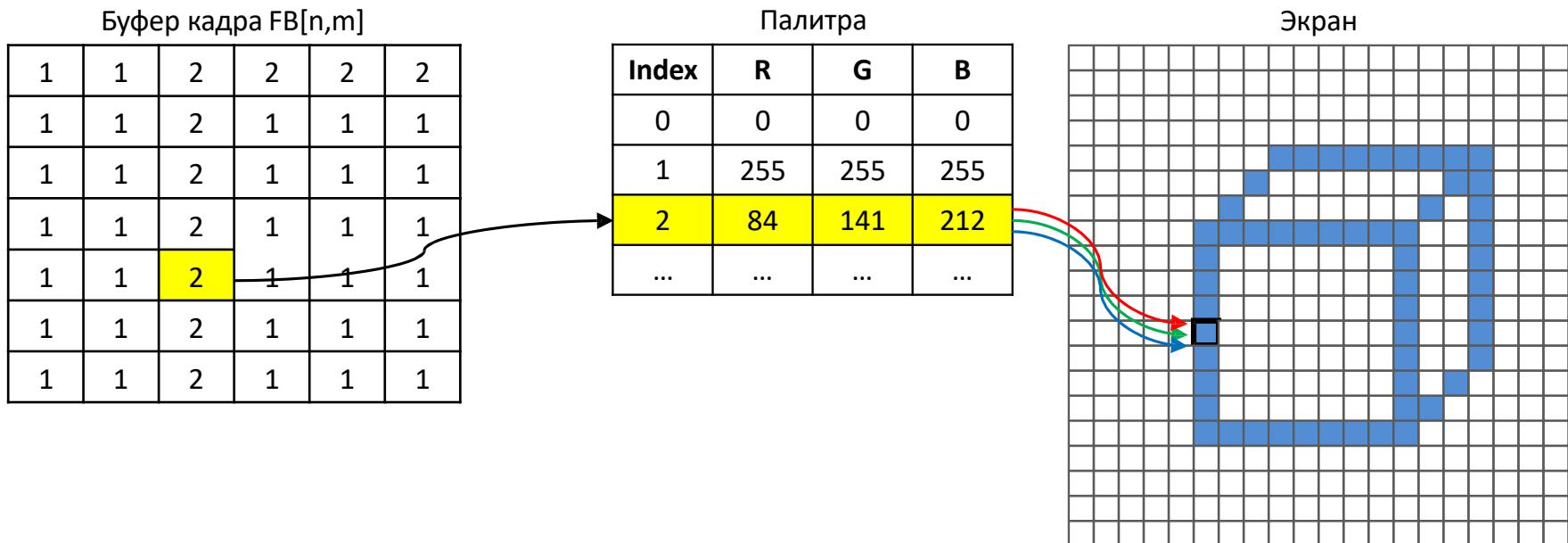
6 бит: Original Amiga chipset (Commodore Amiga, палитра 32 цвета)

8 бит: VGA low res, Super VGA(1989), AGA

12 бит: некоторые станции Silicon Graphics



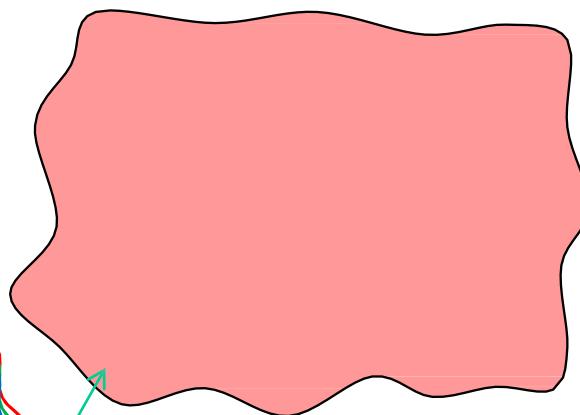
Палитра (LUT – LookUp Table)



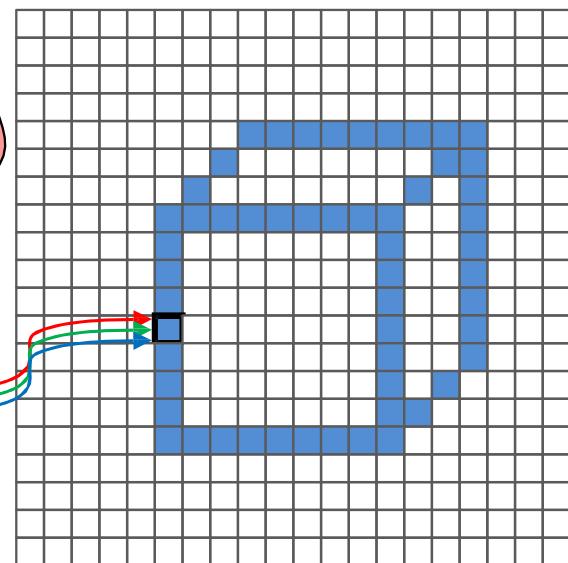
Полноцветные RGB дисплеи

Буфер кадра FB[n,m]

255,255,255	84,141,212	255,255,255
255,255,255	84,141,212	255,255,255
255,255,255	84,141,212	255,255,255
255,255,255	84,141,212	255,255,255
255,255,255	84,141,212	255,255,255
255,255,255	84,141,212	255,255,255
255,255,255	84,141,212	255,255,255



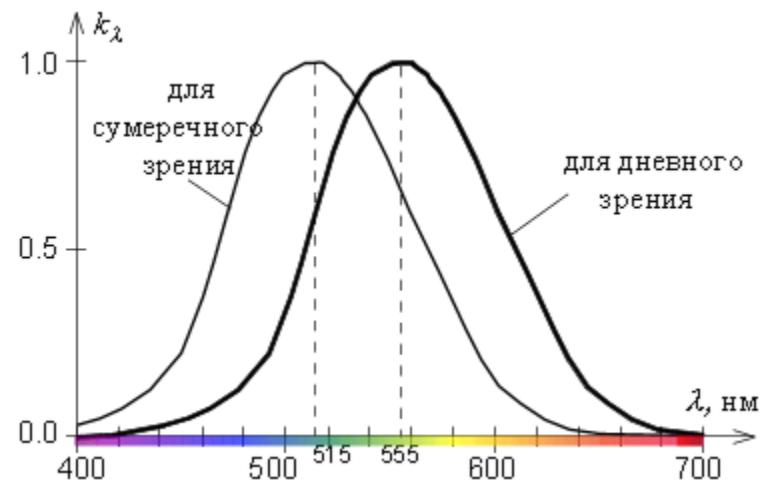
Экран



Но алгоритмы и
модели остались

Полноцветные RGB дисплеи

- 12 бит ($4R+4G+4B$) = 4096 цветов, некоторые модели сотовых, карманных плееров, КПК
- 15 бит HighColor ($5R+5G+5B$) = 32768 цветов
- 16 бит HighColor ($5R+6G+5B$) = 65536 цветов
- 24 бит TrueColor ($8R+8G+8B$) = 16777216 цветов
- 32 бит TrueColor ($8R+8G+8B+8Alpha/empty$)
- 48 бит ($12R+12G+12B+12Alpha$), 96 бит ($32 FB + 16 Z-buffer$) * 2, SGI
- 40-64 бита BrilliantColor
(пример:
 $8R+8G+8B+8C+8M+8Y$),
TI 2005



Halftoning (dithering)

Аппроксимация полутонов

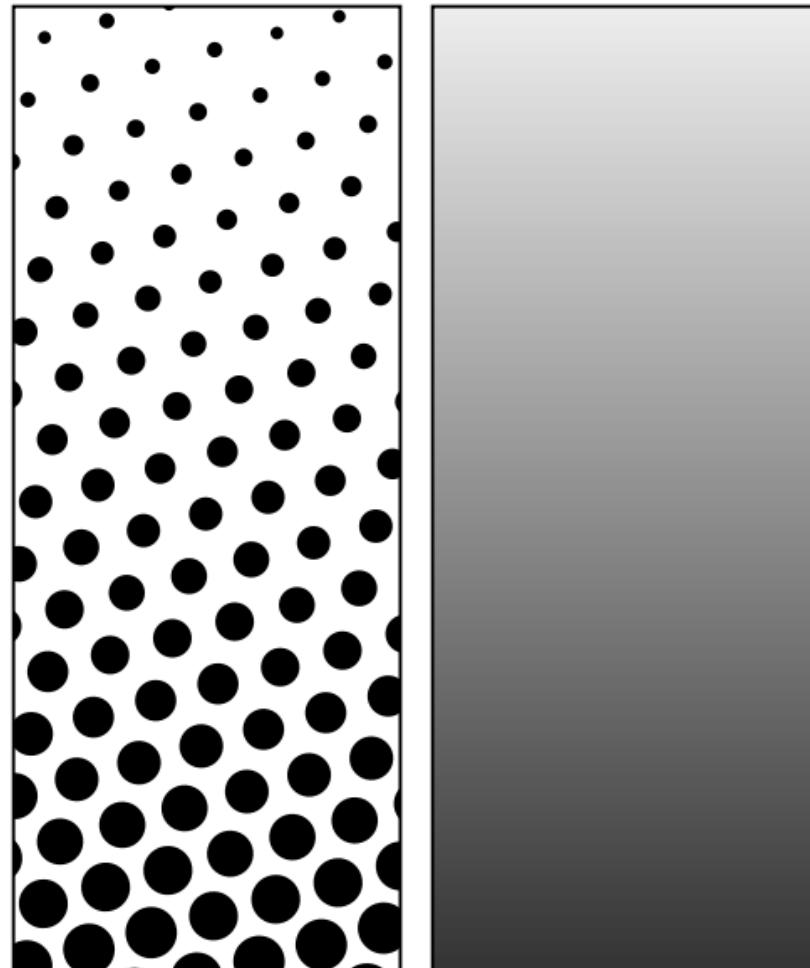
Газетная печать

Достигается за счет варьирования пространственной (площадной) яркости (черноты).

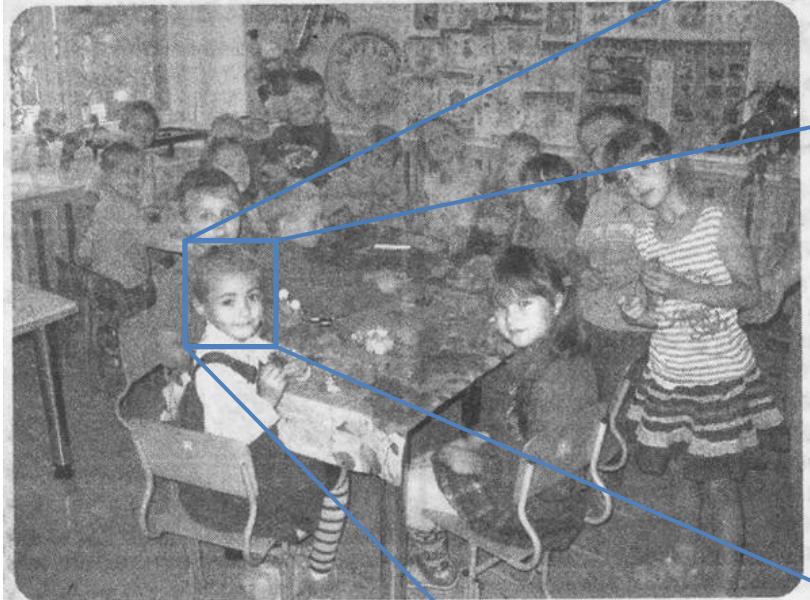
- Газета (оффсет): ~85 lpi
- Лазерный принтер (600 dpi): ~100 lpi
- Журнал (оффсет): 100–200 lpi

Точки разного диаметра.

Повышение цветового разрешения достигается за счет высокого пространственного разрешения.

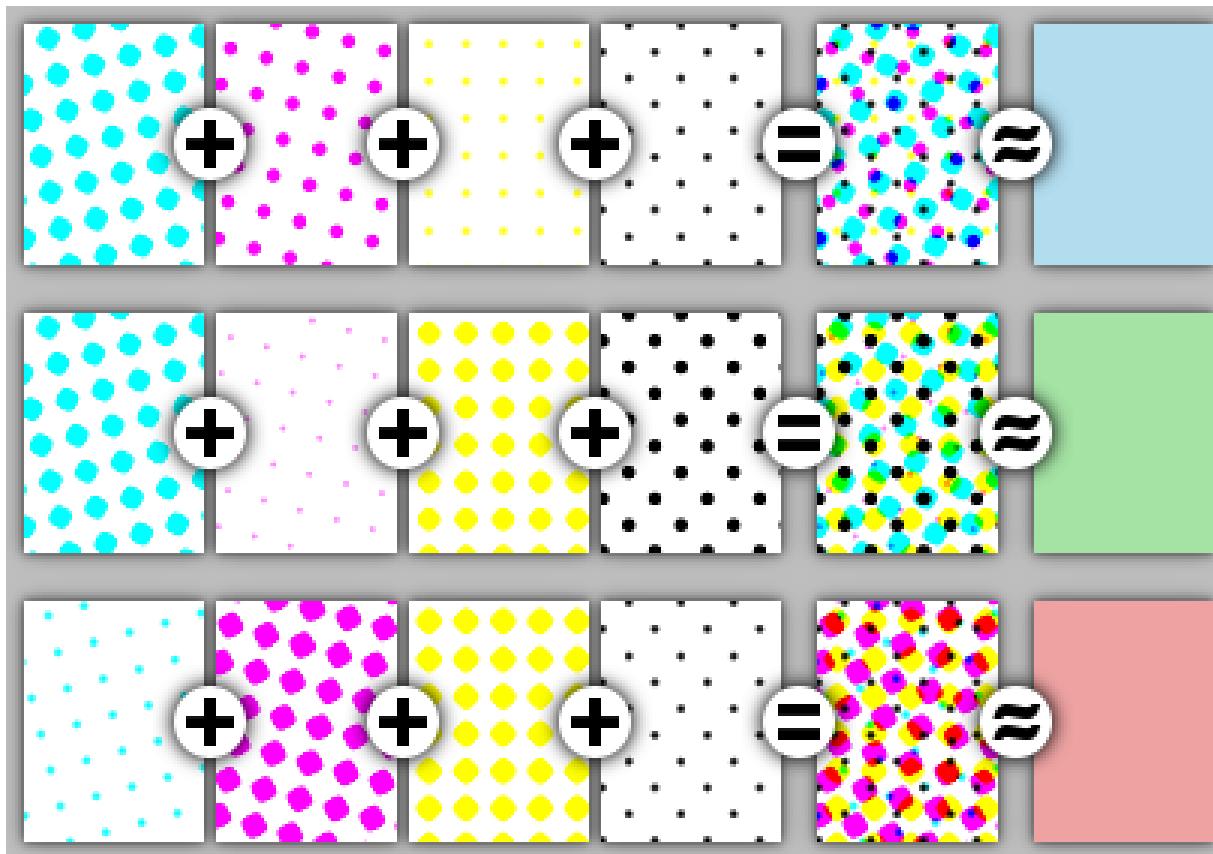


Halftoning: пример



«Навигатор», 21 сентября 2012 г.

В цвете



Зачем нужен дизеринг?

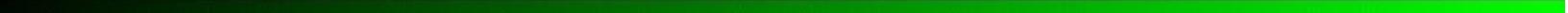
- Печать на принтерах
- Ограничения, накладываемые устройством вывода (дисплей, видеокарта)
- Ограничения, накладываемые форматом файла
- Ограничение памяти, используемой для хранения изображения

Зачем нужен дизеринг?

256 градаций



32 градации



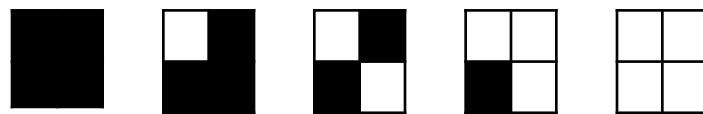
32 градации с дизерингом



Дизеринг с увеличением разрешения

Глаз осуществляет усреднение по пространству, т.е. надо создать шаблоны, которые дают требуемые усредненные яркости.

Например, 5 разных уровней яркости:

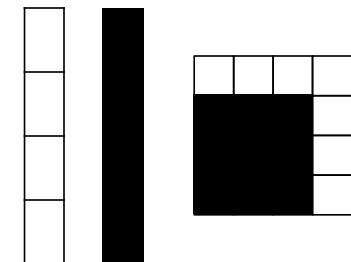


$$2 \times 2 + 1 = 5$$

Дизеринг с увеличением разрешения

Правила построения шаблонов:

- Пиксель, появившийся на одной яркости, должен присутствовать и на более высоких яркостях
- Надо избегать надоедающих артефактов, проявляющихся в области постоянной яркости (см. пример 3x3 – наклонные или вертикальные штрихи)
- Удаление изолированных пикселей для ряда устройств типа лазерных принтеров

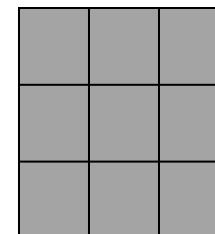
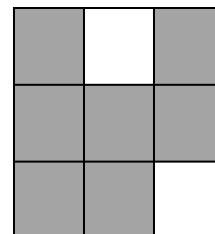
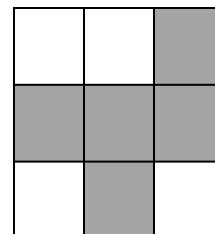
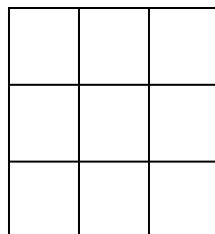


Дизеринг с увеличением разрешения

- Матрица дизеринга при использовании повышенного пространственного разрешения

6 8 4
1 0 3
5 2 7

- В целом используя шаблон $K \times K$ можно получить $K \times K + 1$ уровень.



Дизеринг без увеличения разрешения

Пересчет изображения с N интенсивностями без изменения разрешения в палитру, состоящую из K интенсивностей:

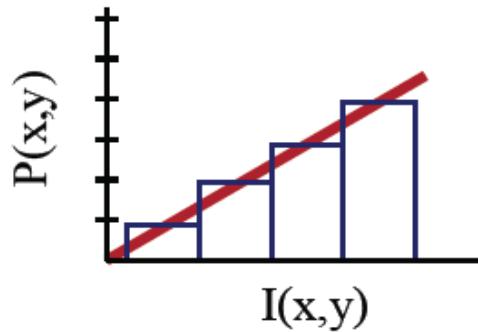
- $K = 2$, только 1 и 0
- $K < N$

Идея дизеринга – сохранение «энергетически» исходного изображения (отклонения от истинных значений в одном пикселе компенсируются коррекцией значений соседних пикселей)

Равномерное квантование



$I(x,y)$



$P(x,y)$



8 бит



4 бита

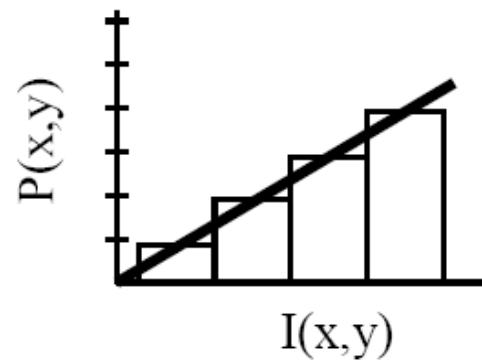
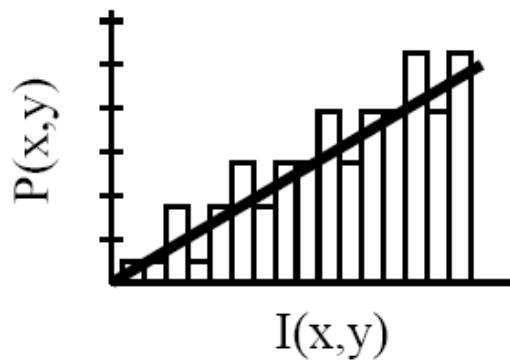


2 бита



1 бита

Дизеринг с зашумлением



$$P(x, y) = \text{trunc}(I(x, y) + \text{noise}(x, y) + 0.5)$$



Оригинал



Uniform quantization



Rand(80)

Дизеринг с зашумлением



Uniform quantization



Rand(10)



Rand(20)



Rand(40)



Rand(80)

Rand(160)

Упорядоченный дизеринг

Дизеринг с использованием матрицы, но без увеличения разрешения
Обычно используют матрицы 2×2 , 4×4 , 8×8 , 16×16

$$\begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}$$

0	32	8	40	2	34	10	42
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

Ordered dither

$$D_{2n} = \begin{bmatrix} 4D_n & 4D_n + 2U_n \\ 4D_n + 3U_n & 4D_n + U_n \end{bmatrix}$$

$$\begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix}$$

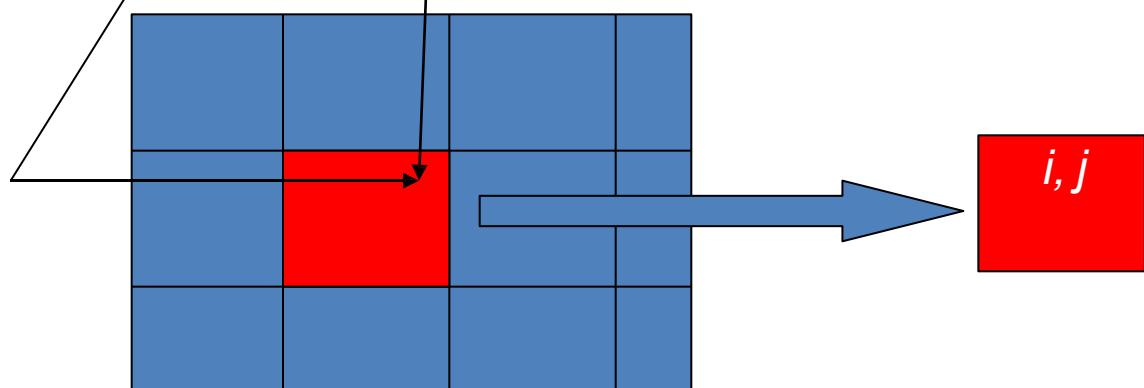
$$\begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 32 & 8 & 40 & 2 & 34 & 10 & 42 \\ 48 & 16 & 56 & 24 & 50 & 18 & 58 & 26 \\ 12 & 44 & 4 & 36 & 14 & 46 & 6 & 38 \\ 60 & 28 & 52 & 20 & 62 & 30 & 54 & 22 \\ 3 & 35 & 11 & 43 & 1 & 33 & 9 & 41 \\ 51 & 19 & 59 & 27 & 49 & 17 & 57 & 25 \\ 15 & 47 & 7 & 39 & 13 & 45 & 5 & 37 \\ 63 & 31 & 55 & 23 & 61 & 29 & 53 & 21 \end{pmatrix}$$

Ordered dither

$$\begin{aligned} i &= x \pmod{n} \\ j &= y \pmod{n} \end{aligned}$$

$$I_{\text{out}}(x,y) = \begin{cases} 1, & I_{\text{in}}(x,y) > D_n(i,j) \\ 0, & I_{\text{in}}(x,y) \leq D_n(i,j) \end{cases}$$



$$\left(\begin{array}{cccc} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{array} \right)$$

Ordered dither — результаты



Оригинал



Rand(80)



OD-2



OD-4



OD-8



OD-16

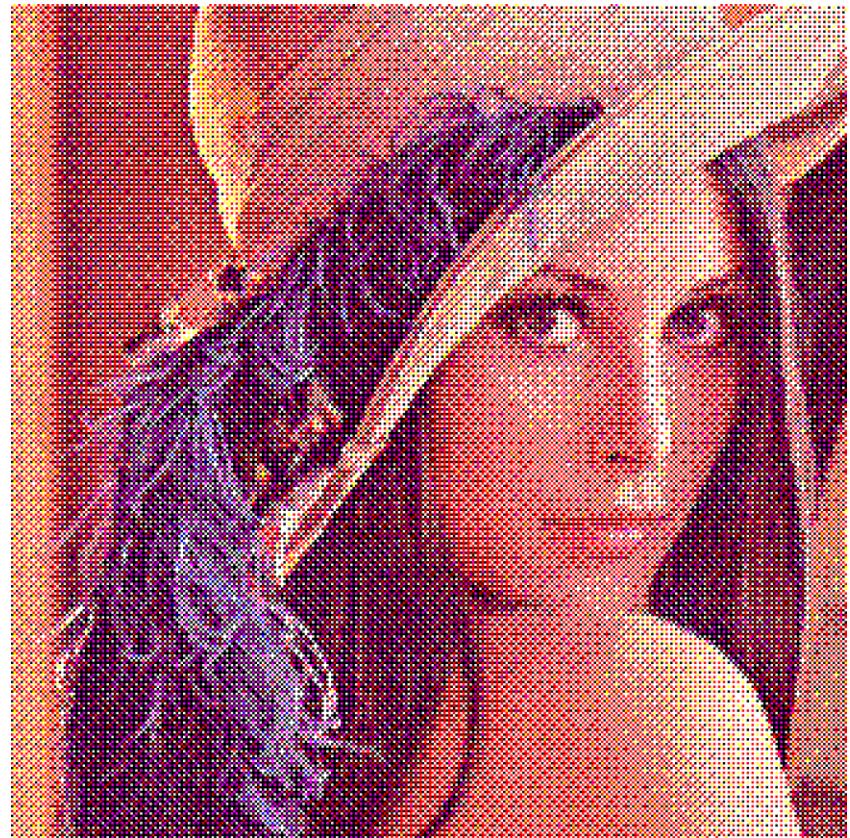
Крупным планом



OD-4



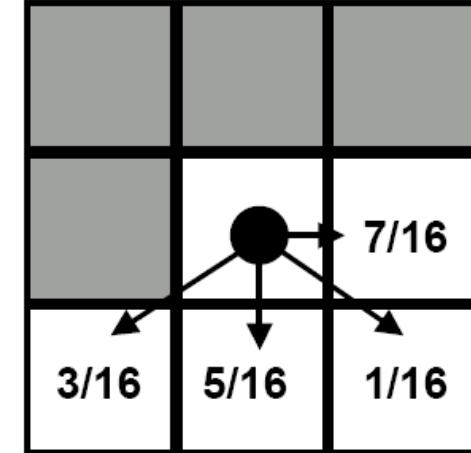
По R, G, B



- Красный – 3 бита,
- Зеленый – 3 бита,
- Синий – 2 бита, т.к. он наименее информативен

Floyd-Steinberg Error Diffusion, 1975

- Простая аппроксимация
- Из N градаций в M градаций
- Перебор по строкам растра
- В каждом пикселе рисовать значение, дающее минимальную ошибку (округление)
- Делить ошибку на 4 неравных порции
- Порции ошибки «возвращать» во входное изображение



```
for (y=0; y<height; y++) {  
    for (x=0; x<width; x++) {  
        P(x, y)=trunc(I(x, y)+0.5);  
        e=I(x, y)-P(x, y); // ошибка  
        I(x+1, y) +=7*e/16;  
        I(x-1, y+1) +=3*e/16;  
        I(x, y+1) +=5*e/16;  
        I(x+1, y+1) +=1*e/16;  
    }  
}
```

Floyd-Steinberg — результаты



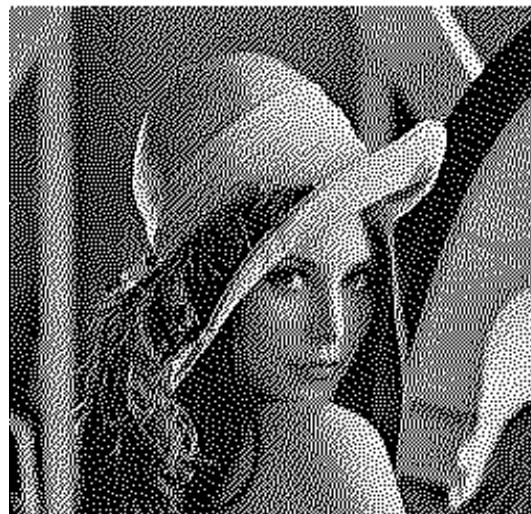
Оригинал



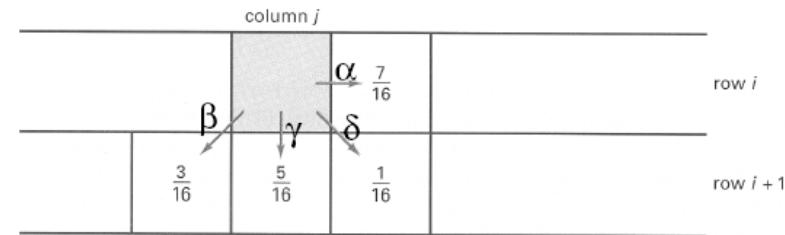
Rand(80)



OD-16



Floyd-Steinberg

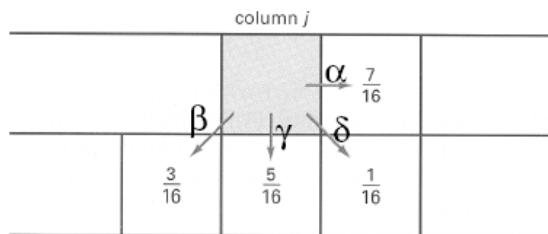


$$\alpha + \beta + \gamma + \delta = 1.0$$

Крупным планом



FS



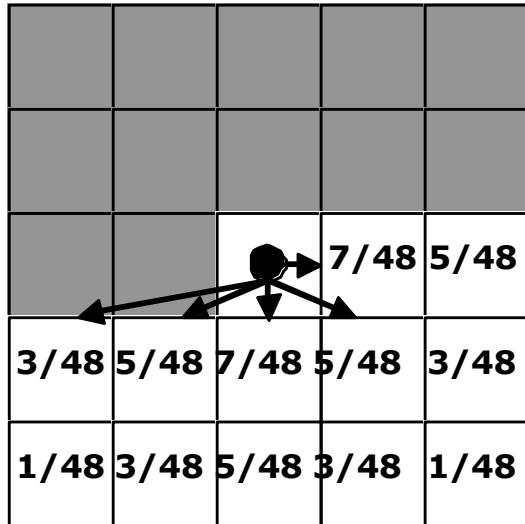
$$\alpha + \beta + \gamma + \delta = 1.0$$

Что делать с границей?

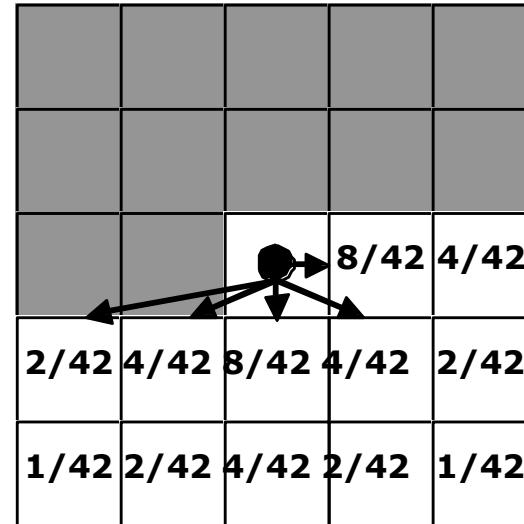


Разброс ошибки по-другому

Jarvis dithering, 1976
(Jarvis, Judice, Ninke)



Stucki dithering, 1981



Burkes Dithering (1988), Sierra Dithering (1989, несколько) и т. д.

Сравним



Оригинал



По порогу



Ordered



Floyd-Steinberg



Jarvis



Stucki

Тестовые изображения



Портрет
шведской модели Лены
Сёдерберг, 1972

Стандартное тестовое
изображение для сравнение
алгоритмов обработки
изображений

Лена, Lenna

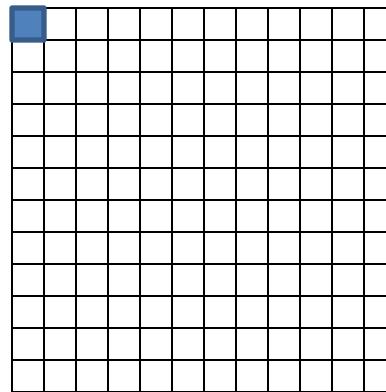
Широкий диапазон цветов,
Наличие резких и плавных границ

Задачи обработки изображений

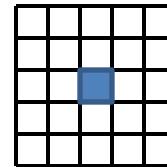
- Выделение контурной информации
- Перевод в черно-белое изображение
- Художественные фильтры

Фильтрация изображений

$A[x, y]$ – входное изображение

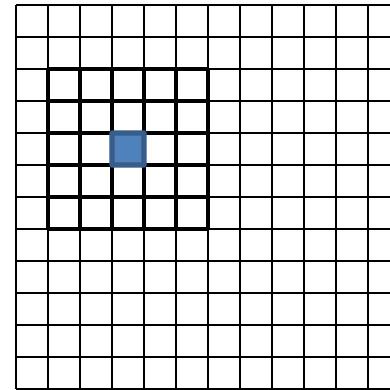


$H[x, y]$ – фильтр $(2n + 1) * (2n + 1)$



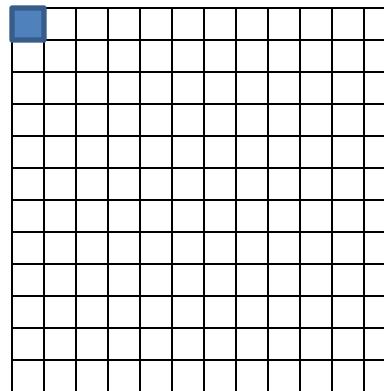
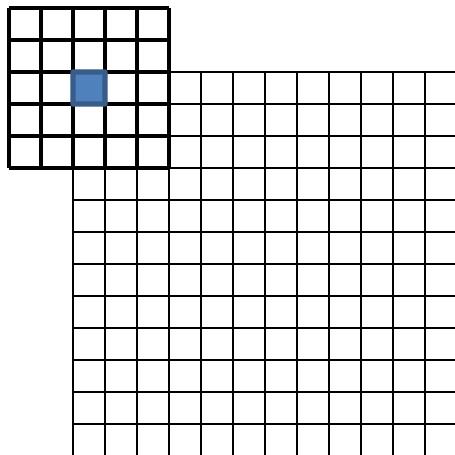
Фильтрация изображений

$B[x, y]$ – выходное изображение

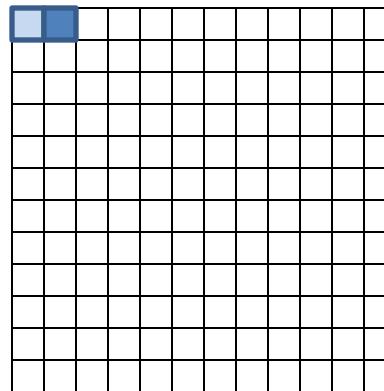
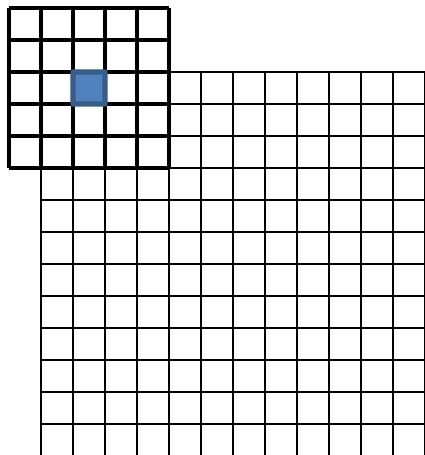


$$B[x, y] = \sum_{u=-n}^{+n} \sum_{v=-n}^{+n} H[u, v] * A[x + u, y + v]$$

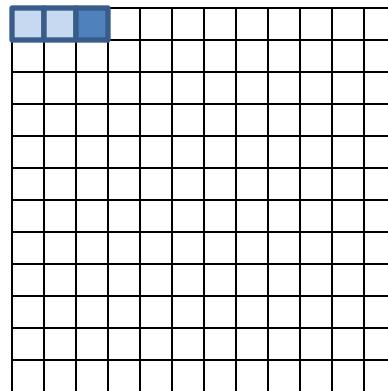
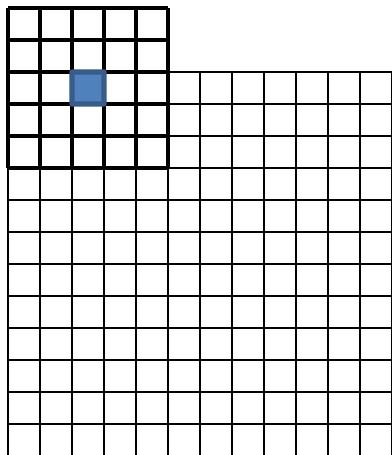
Фильтрация изображений



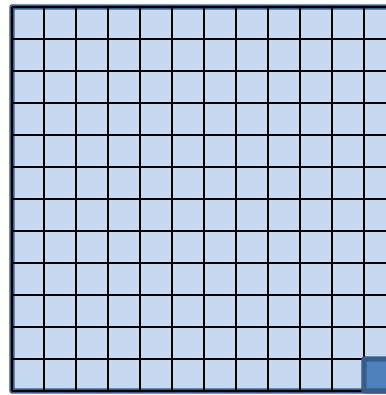
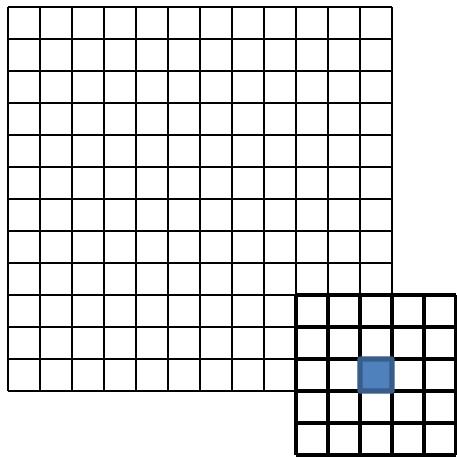
Фильтрация изображений



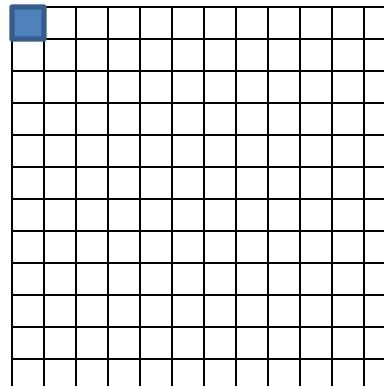
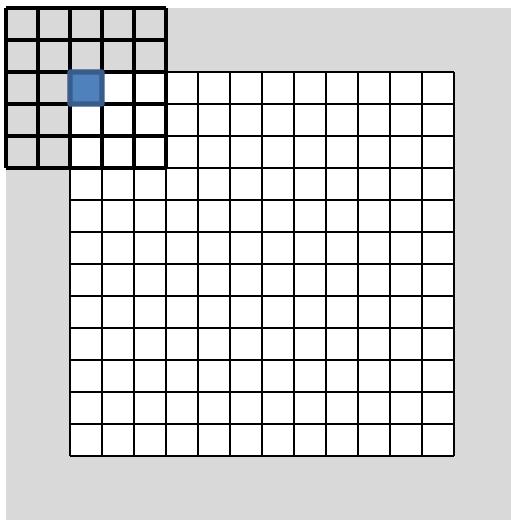
Фильтрация изображений



Фильтрация изображений



Фильтрация изображений



Сглаживатели / усреднители:

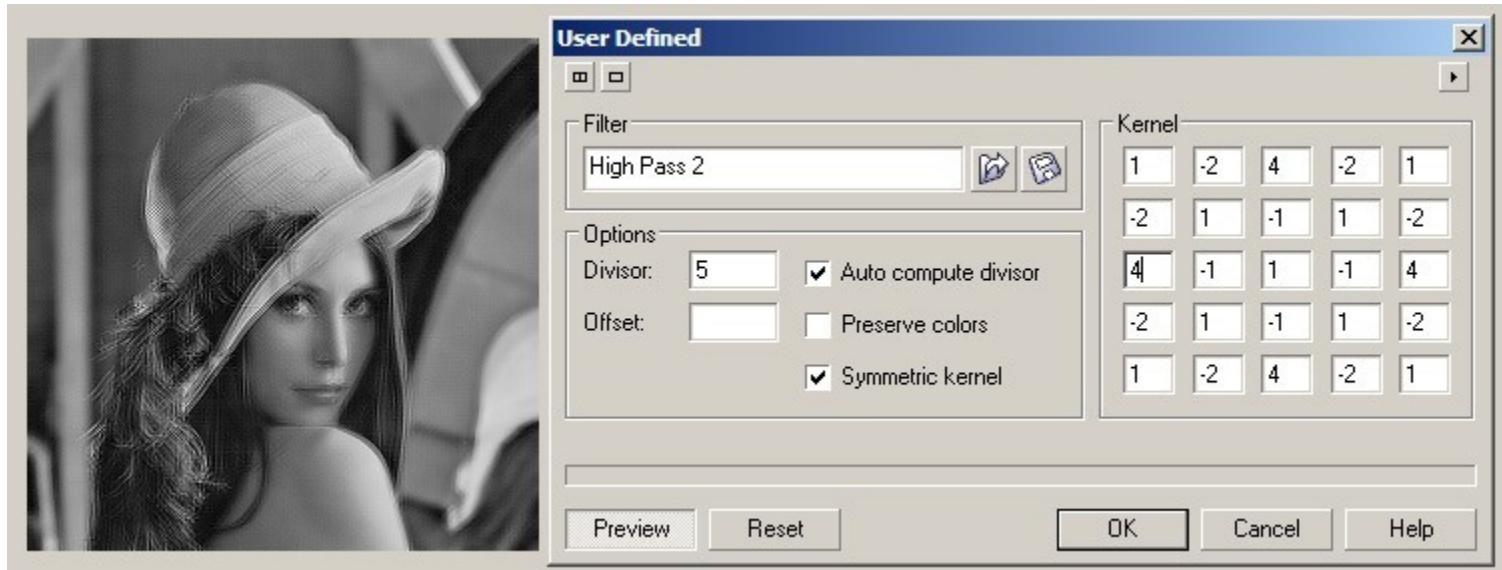
$$H = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$H = \frac{1}{10} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$H = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

$$\frac{1}{6} \begin{bmatrix} 1 & & \\ & 1 & \\ 1 & 2 & 1 \\ & 1 & \end{bmatrix} \text{blur}$$

$$\frac{1}{74} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 5 & 4 & 2 \\ 3 & 5 & 6 & 5 & 3 \\ 2 & 4 & 5 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

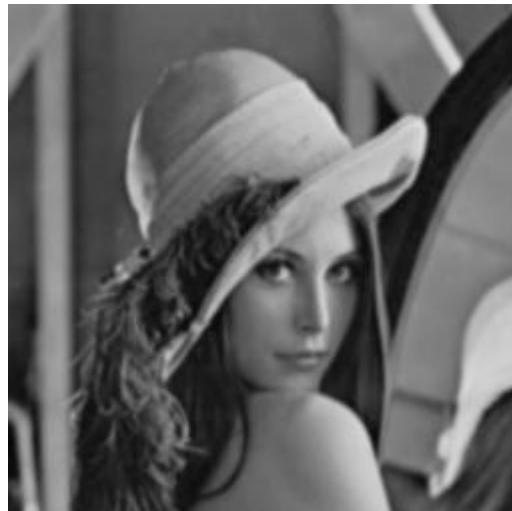


Гауссово размытие

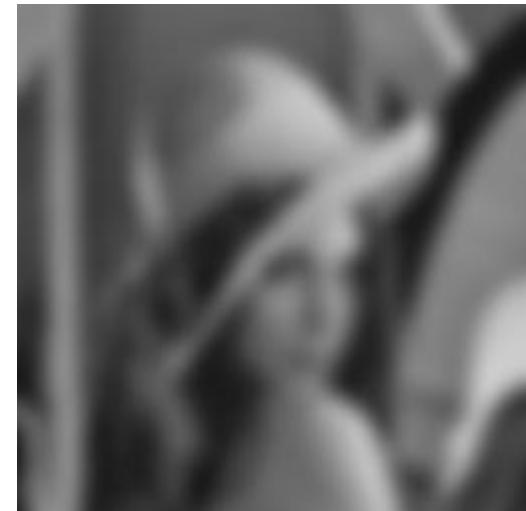
Гауссово размытие (Gaussian blur): ядро — двумерная нормированная гауссиана; обычно можно управлять параметром σ (в пикселях)



$$\sigma = 0.1$$



$$\sigma = 1.0$$



$$\sigma = 5.0$$

Размытие (blur)



17.10.2000

Компьютерная графика. Лекция

3

$$\frac{1}{6} \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

25

Размытие (прод)

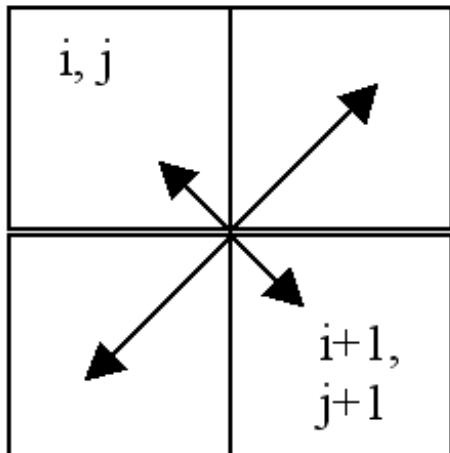


$$\frac{1}{74} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 5 & 4 & 2 \\ 3 & 5 & 6 & 5 & 3 \\ 2 & 4 & 5 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

Дифференцирование – выделение границ

$$g(x, y) \longrightarrow g_{ij} = g(i, j)$$

$$\|\nabla g(x, y)\| \approx R(i, j) = \sqrt{[g_{i,j} - g_{i+1,j+1}]^2 + [g_{i,j+1} - g_{i+1,j}]^2}$$



Однородная интенсивность:
 $R(i, j) = 0$.

Упрощение: оператор Робертса:

$$R(i, j) \leq F(i, j) = |g_{i,j} - g_{i+1,j+1}| + |g_{i,j+1} - g_{i+1,j}| \leq \sqrt{2}R(i, j)$$

Этот оператор известен как:

- Градиентное изображение
- Пространственное дифференцирование
- Выделение контуров
- Повышение резкости
- Взятие градиента

Он применяется следующим образом:

```
if (F(i, j) > C)
    (i, j) := 1;
else
    (i, j) := 0;
```

Другая оценка градиента (оператор Собеля)

a	b	c
d	e	f
g	h	i

$$S_x = (c + 2f + i) - (a + 2d + g)$$

$$S_y = (g + 2h + i) - (a + 2b + c)$$

$$S = \sqrt{S_x^2 + S_y^2} \longrightarrow S = |S_x| + |S_y|$$

Выделение контуров

$$\frac{1}{4} \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

h

0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25

a

0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0
0	0	0	0	0	25	25	0	0	0	0

b

Выделение контура



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

+ порог

Выделение контуров (отдельно по компонентам цвета)



Увеличение резкости



17.10.2000

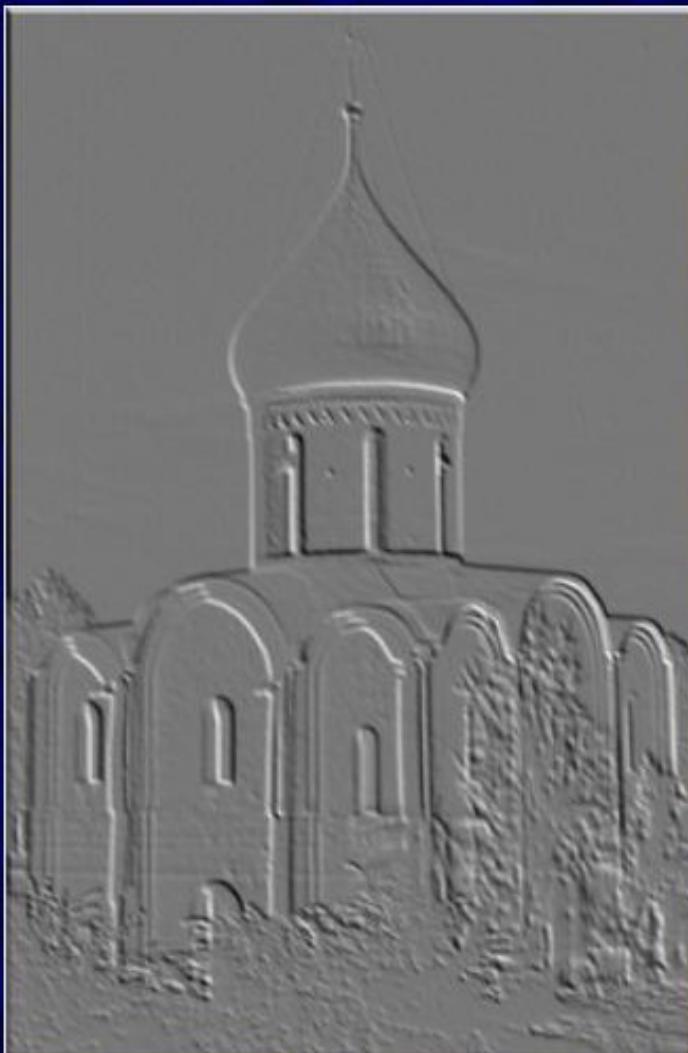
Компьютерная графика. Лекция

3

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

27

Тиснение



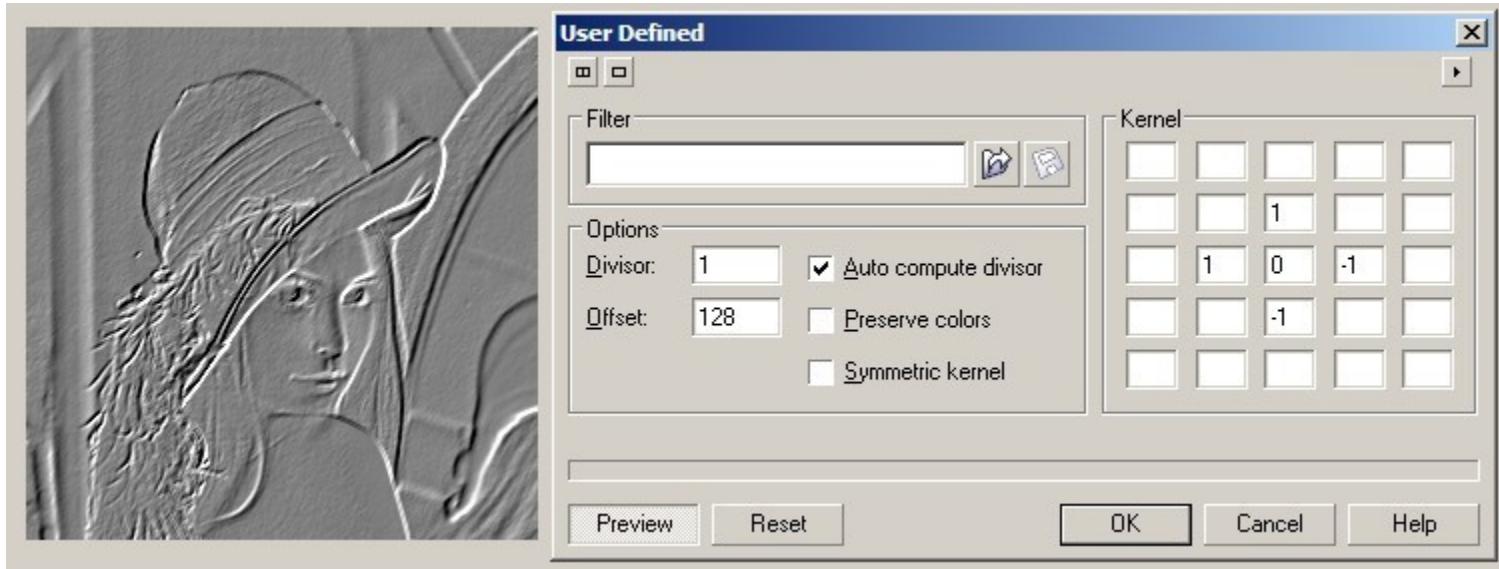
$$\begin{bmatrix} 1 \\ -1 & 0 & 1 \\ -1 \end{bmatrix}$$

+ сдвиг
яркости

Тиснение

1. Тиснение делается почти также как размывание и увеличение резкости.
Процесс начинается с обычным цветным изображением.
2. Каждый пиксель в изображении обрабатывается ядром тиснения размером 3×3 . В отличие от ядер размывания и резкости, в которых сумма коэффициентов равна 1, сумма весов в ядре тиснения равна 0. Это означает, что "фоновым" пикселям (пикселям, которые не находятся на границах перехода от одного цвета к другому) присваиваются нулевые значения, а не фоновым пикселям – значения, отличные от нуля.
3. После того, как значение пикселя обработано ядром тиснения, к нему прибавляется 128. Таким образом значением фоновых пикселей станет средний серый цвет (красный = 128, зеленый = 128, синий = 128). Суммы, превышающие 255, можно округлить до 255.
4. В тисненом варианте изображения, контуры кажутся выдавленными над поверхностью. Направление подсветки изображения можно изменять, меняя позиции 1 и -1 в ядре. Если, например, поменять местами значения 1 и -1, то реверсируется направление подсветки.

Тиснение



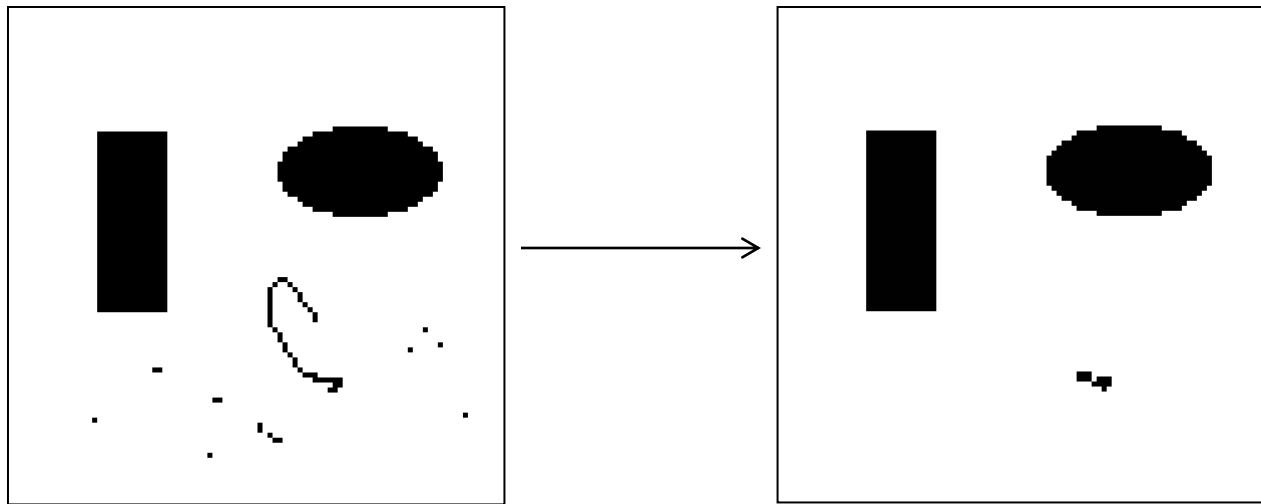
Черно-белое изображение



$$Y = 0.299R + 0.587G + 0.114B$$

Фильтрация

Медианный фильтр – все пиксели в окрестности текущего пикселя упорядочиваются по возрастанию и текущему пикселю присваивается центральное значение последовательности



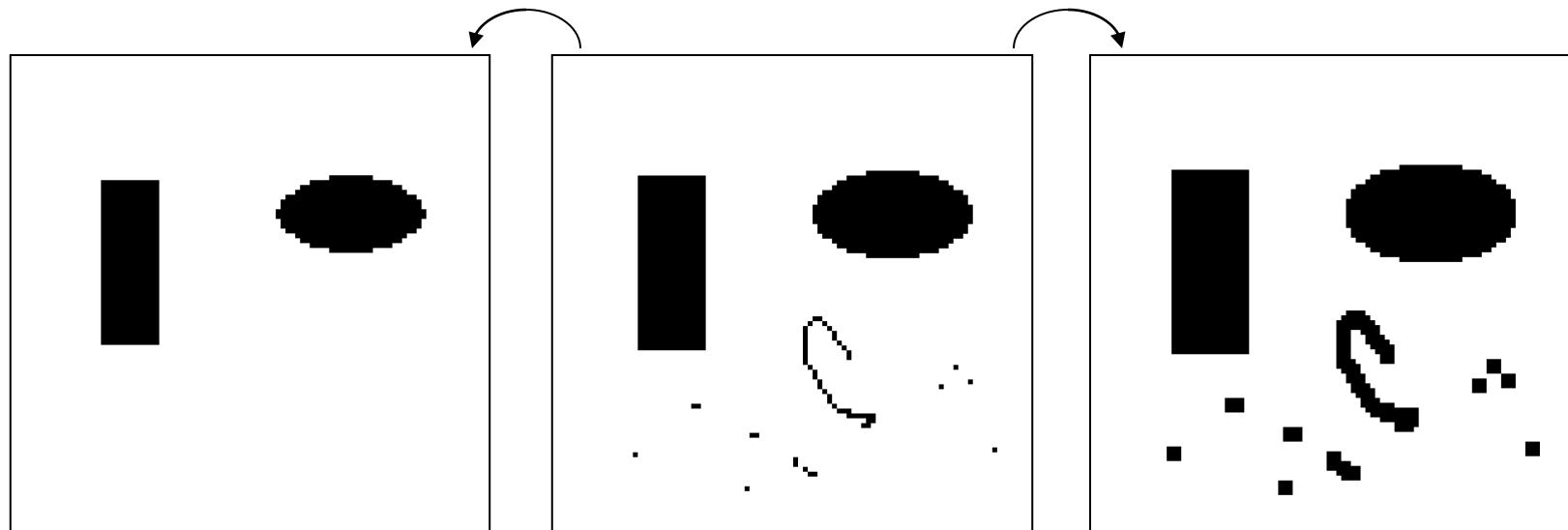
Фильтрация

Сужение (Erode) – если все точки окрестности равны 1, то значение в точке равно 1, иначе равно 0

Расширение (Dilate) – если хотя бы одна точка в окрестности равна 1, то значение в точке равно 1, иначе равно 0

Закрытие (Closing) – последовательное выполнение расширения и сужения

Раскрытие (Opening) – последовательное выполнение сужения и расширения



Акварелизация

1. Акварельный фильтр преобразует изображение, и после обработки оно выглядит так, как будто написано акварелью. Например, берем цифровое изображение, просканированное с фотографии.
2. Первый шаг в применении акварельного фильтра - сглаживание цветов в изображении. Одним из способов сглаживания является процесс медианного усреднения цвета в каждой точке (*медианный фильтр*). Значение цвета каждого пикселя и его 24 соседей помещаются в список и сортируются от меньшего к большему. Медианное (тринадцатое) значение цвета в списке присваивается центральному пикселу.
3. После сглаживания цветов, компьютер обрабатывает каждый пикセル в изображении ядром резкости, чтобы выделить границы переходов цветов.
4. Результатирующее изображение напоминает акварельную живопись. Это лишь один пример, который показывает, как можно объединять различные методы обработки изображений и добиваться необычных визуальных эффектов.



Акварелизация

