

---

# Simple object\edge detection

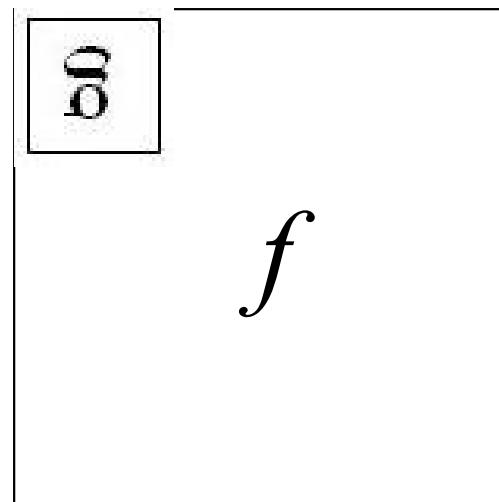
Alexandr Pak

# 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”

# 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

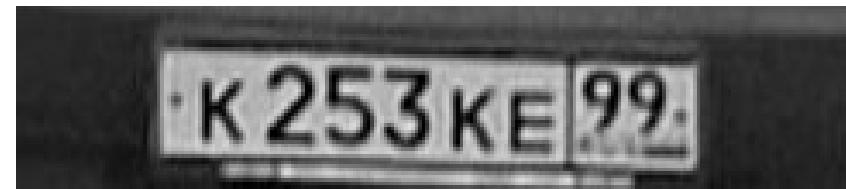
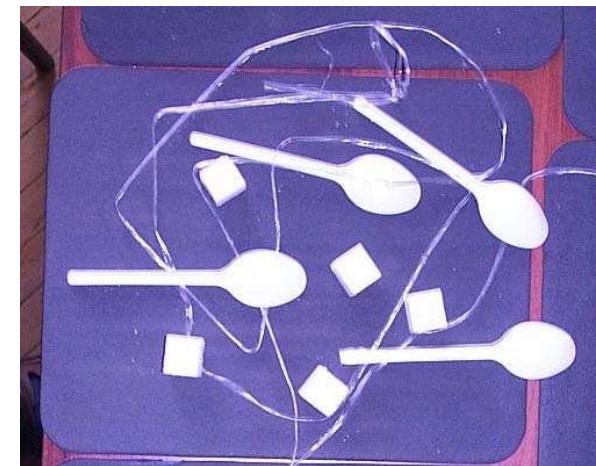
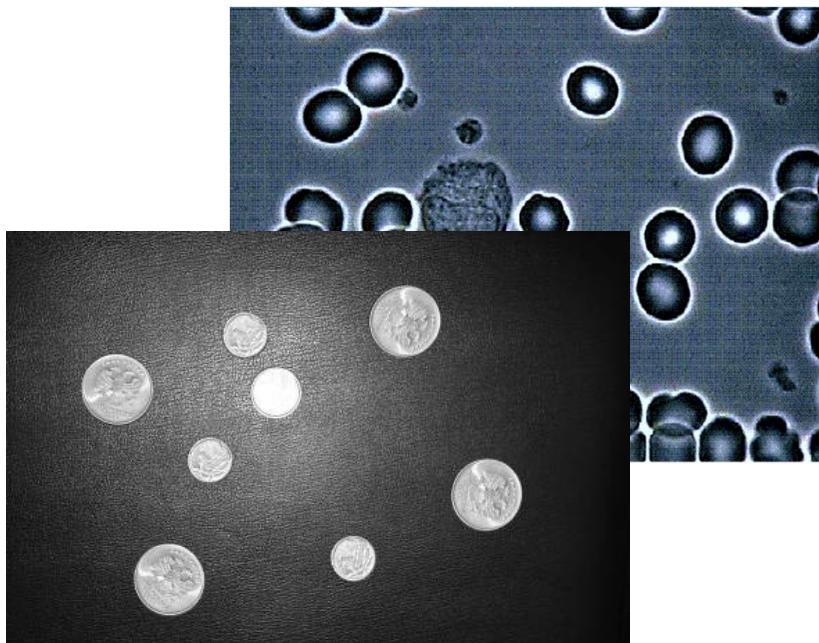
# Свойства

---

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

# Простой анализ изображений

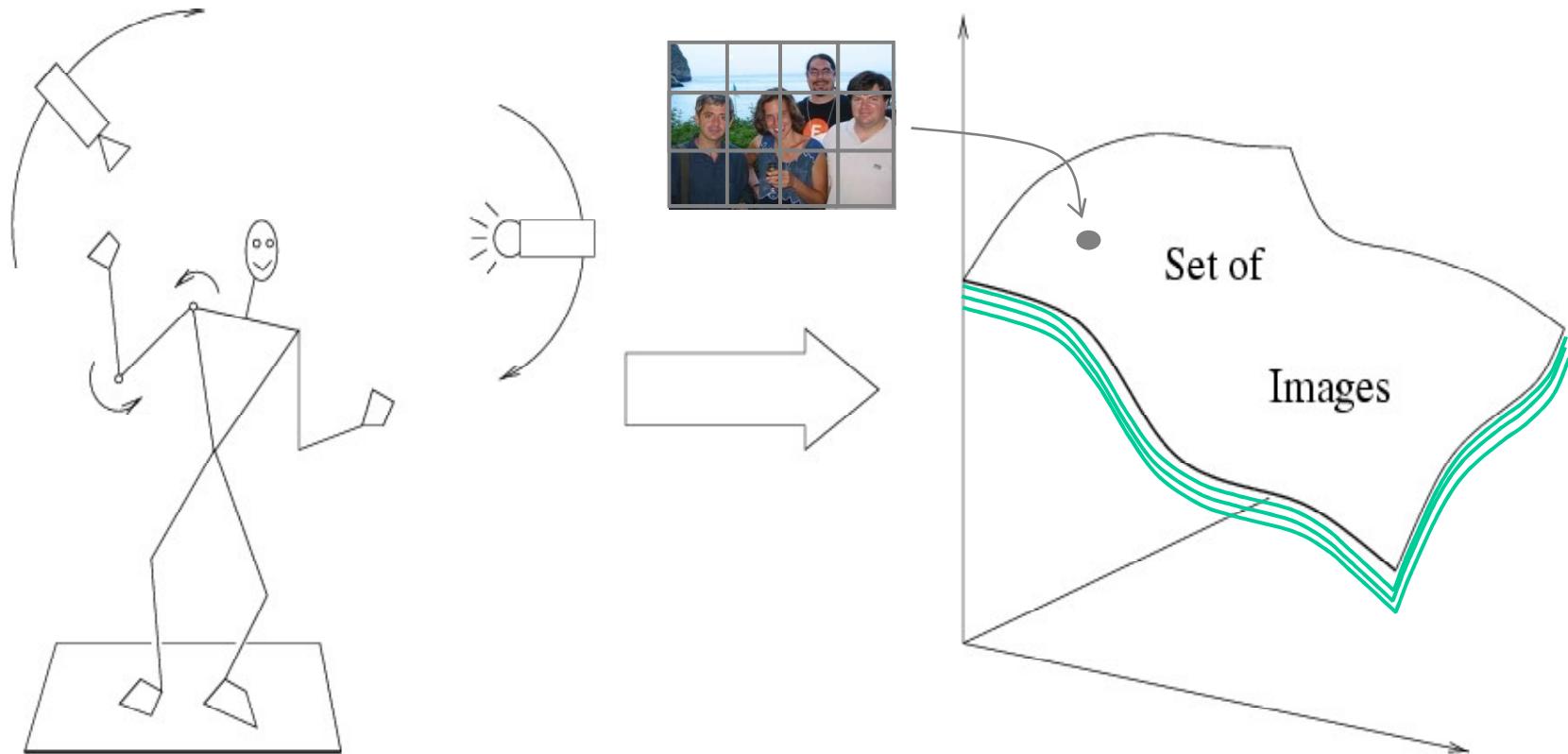
---



Many slides adapted from Fei-Fei Li, Rob Fergus, Antonio Torralba, Jean Ponce and Svetlana Lazebnik

# Изменчивость изображений

---



Внешние факторы:

Положение камеры  
Освещение

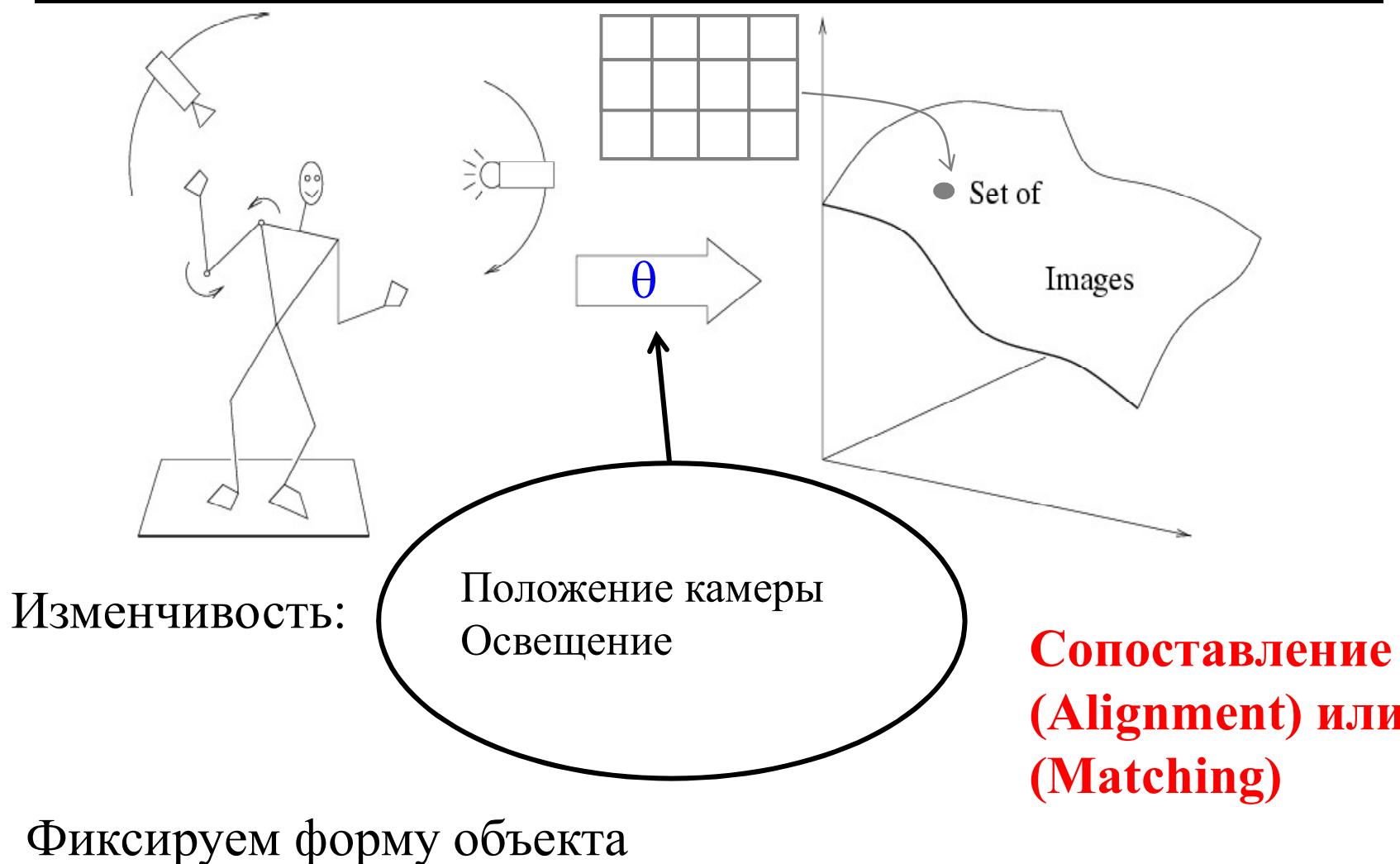
Внутренние факторы: Внутриклассовая изменчивость

# Внутриклассовая изменчивость

---



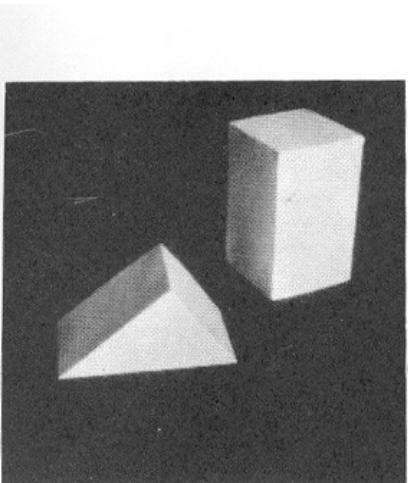
# Сопоставление



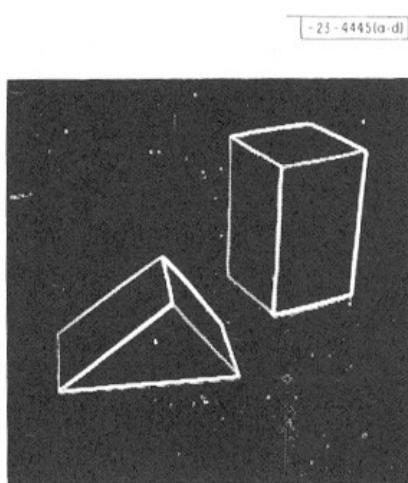
Roberts (1965); Lowe (1987); Faugeras & Hebert (1986); Grimson & Lozano-Perez (1986); Huttenlocher & Ullman (1987)

# Сопоставление

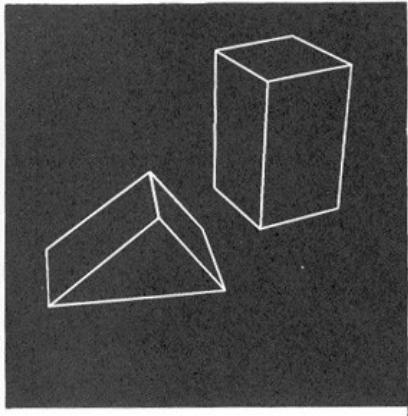
---



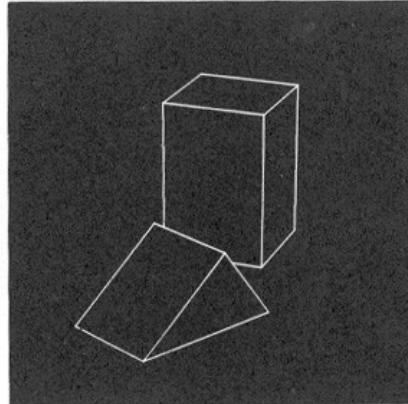
(a) Original picture.



(b) Differentiated picture.



(c) Line drawing.

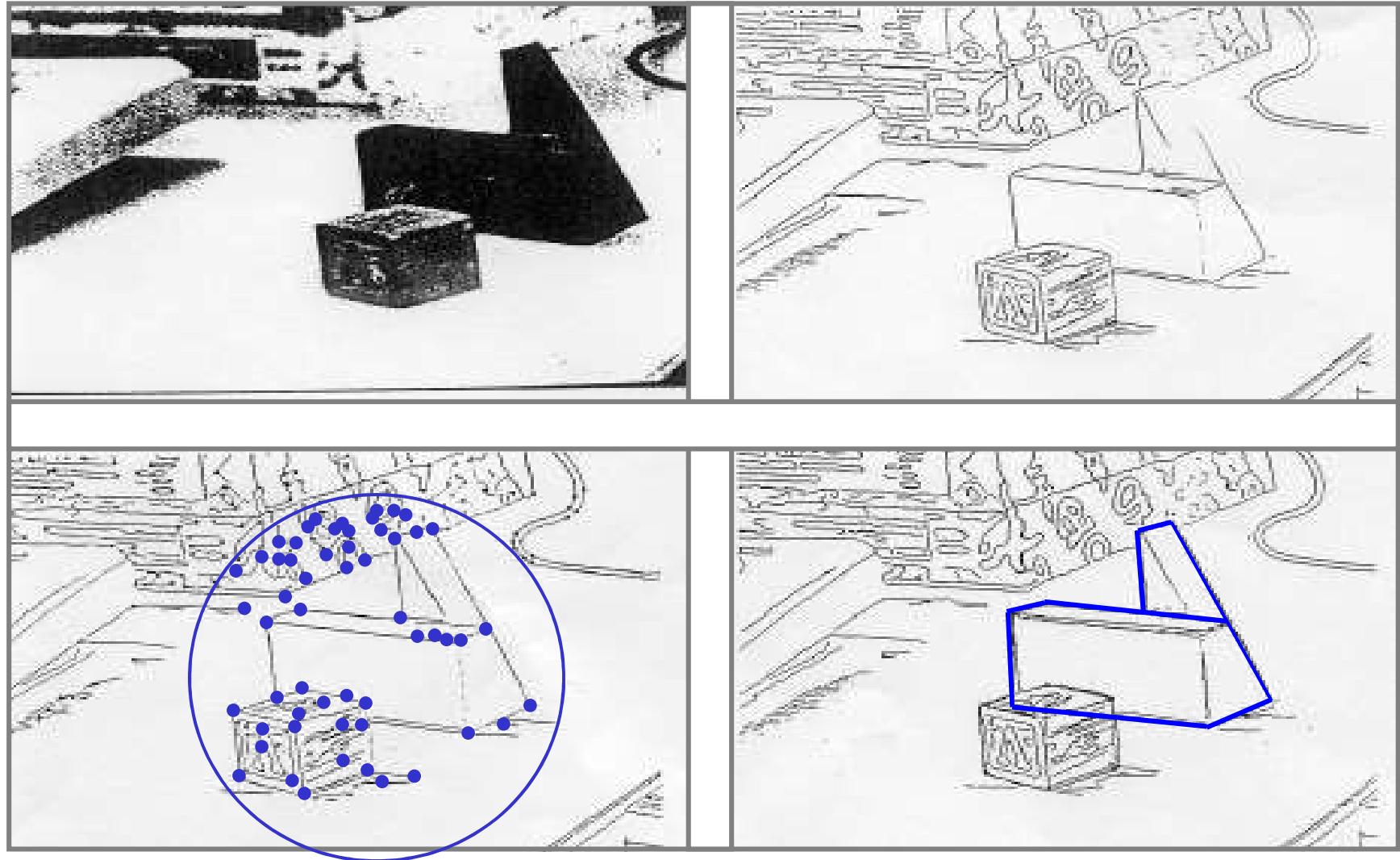


(d) Rotated view.

L. G. Roberts, [\*Machine Perception of Three Dimensional Solids\*](#),  
Ph.D. thesis, MIT Department of Electrical Engineering, 1963.

# Сопоставление

---



Huttenlocher & Ullman (1987)

# Сопоставление шаблонов

---

- Фиксируем объект
- Опишем объект его изображением – шаблоном (pattern)
- Хотим найти объект в изображении
- Ограничим возможные преобразования (внешние факторы)
  - Сдвиг, размер, поворот
  - Освещение?
- Будем искать объект в изображении путём попиксельного сравнения шаблона и всех фрагментов изображения
- «Pattern matching»



# Метрики

---

$$\sum_X \sum_Y |I_1(X, Y) - I_2(X, Y)| \quad (\text{SAD}) \text{ Sum of absolute differences}$$

$$\sum_X \sum_Y (I_1(X, Y) - I_2(X, Y))^2 \quad (\text{SSD}) \text{ Sum of squared differences}$$

$$\sum_X \sum_Y I_1(X, Y) I_2(X, Y) \quad (\text{CC}) \text{ Cross-correlation}$$

- SAD, SSD – минимизируются (0 – точное совпадение)
- CC – максимизируется (1 – точное совпадение)

# Нормализация освещенности

---

- Освещённость может меняться
- Можно нормализовать интенсивности пикселей шаблона и фрагмента изображения

$$\bar{I} = \frac{1}{|W_m(x,y)|} \sum_{(u,v) \in W_m(x,y)} I(u,v)$$

Средняя интенсивность

$$\|I\|_{W_m(x,y)} = \sqrt{\sum_{(u,v) \in W_m(x,y)} [I(u,v)]^2}$$

Норма интенсивности окна

$$\hat{I}(x, y) = \frac{I(x, y) - \bar{I}}{\|I - \bar{I}\|_{W_m(x,y)}}$$

Нормализованный пиксель

# Выравнивание освещенности

---

Исходное изображение



Линейная функция  
освещенности



Скорректированное  
освещенности



Выравнивание  
гистограммы  
(контраста)



# Пример: пульт ТВ

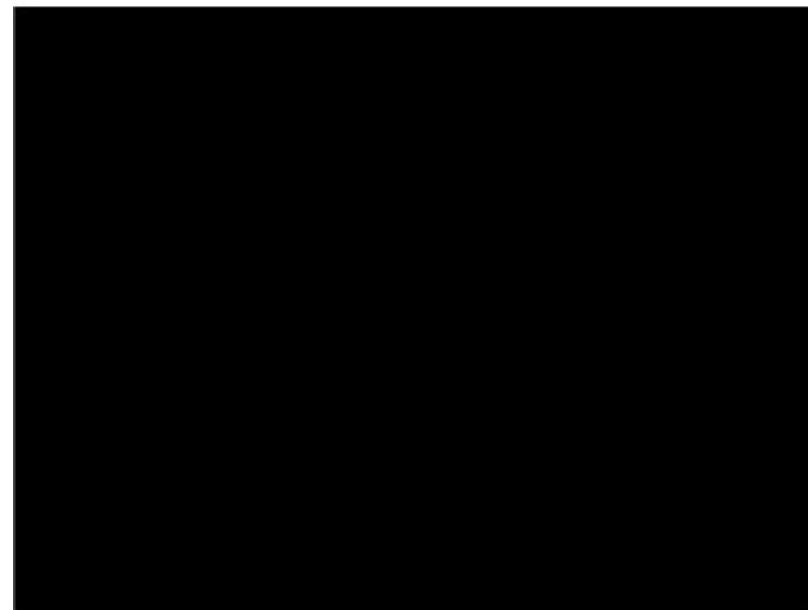
---



- Шаблон (слева), изображение (в центре), карта нормализованной корреляции (справа)
- Пик яркости (максимум корреляции) соответствует положению руки (искомого шаблона)

# Пример: пульт ТВ

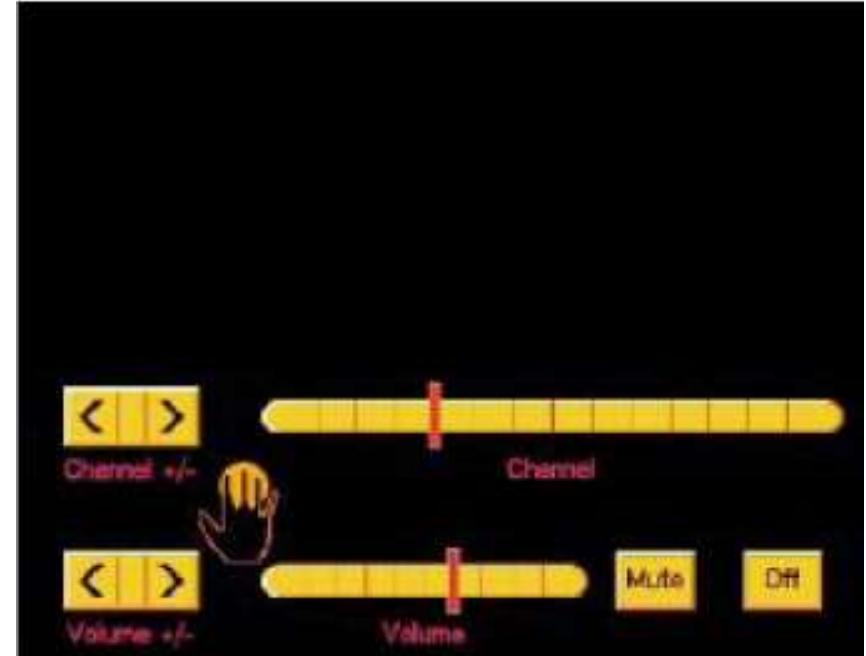
---



Credit: W. Freeman *et al*, “Computer Vision for Interactive Computer Graphics,” *IEEE Computer Graphics and Applications*, 1998

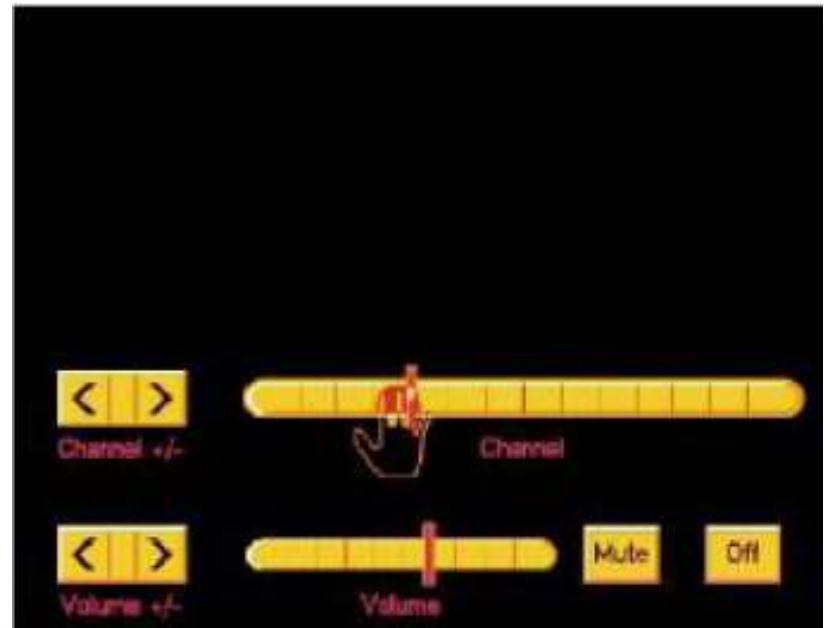
# Пример: пульт ТВ

---



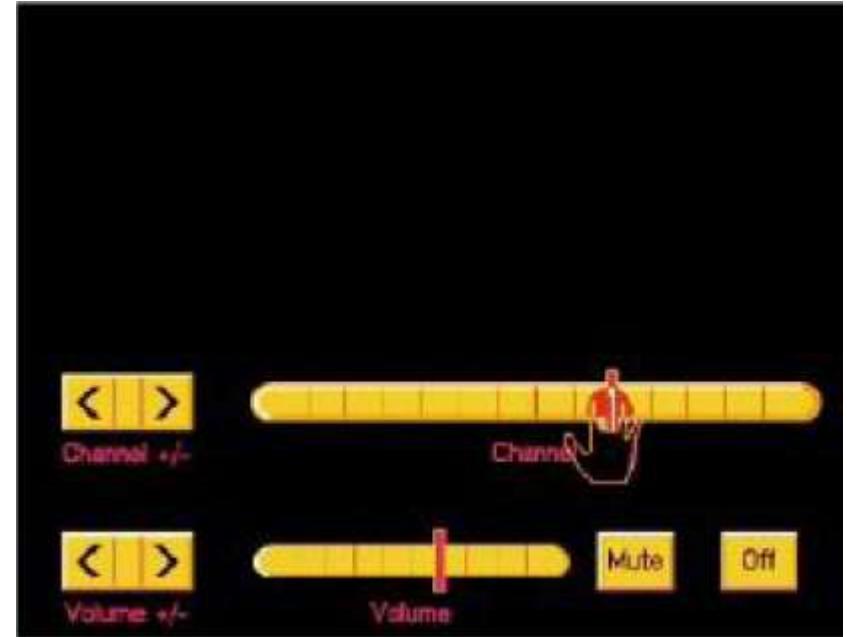
# Пример: пульт ТВ

---



# Пример: пульт ТВ

---



# Ограничения и проблемы

---

- Ищем конкретный объект, а не класс / категорию объектов
  - Не «символ», а конкретную букву в конкретном шрифте
- Трудоёмкость
  - Полный перебор параметров
- Модель преобразования
  - В простом варианте неизвестно только положение, размер и ориентация фиксированы
  - Чтобы учесть поворот и ориентацию придётся перебрать все возможные параметры
- Шаблонов может быть много
  - OCR – распознавание символов
  - По шаблону на каждый символ

# Как улучшить подход?

---

# Поиск краев

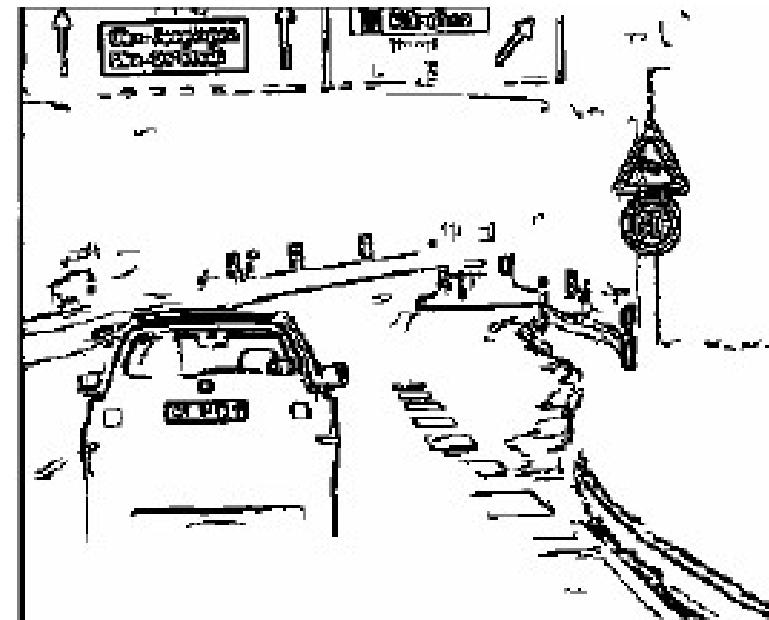
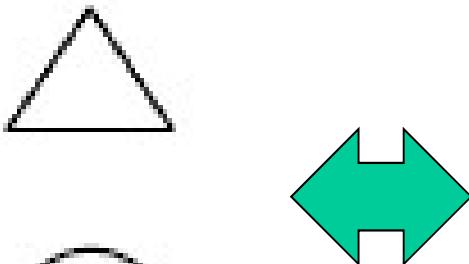
---

- Интуитивно понятно, что основная информация в картинке содержится как раз в границах (краях)
  - Компактное представление
  - Соответствует устройству мозга
- **Задача:** Выделить резкие изменения (разрывы) изображения
- **Идеал:** рисунок художника (но художник уже пользуется своими знаниями об объектах)



# Края для сопоставления шаблонов

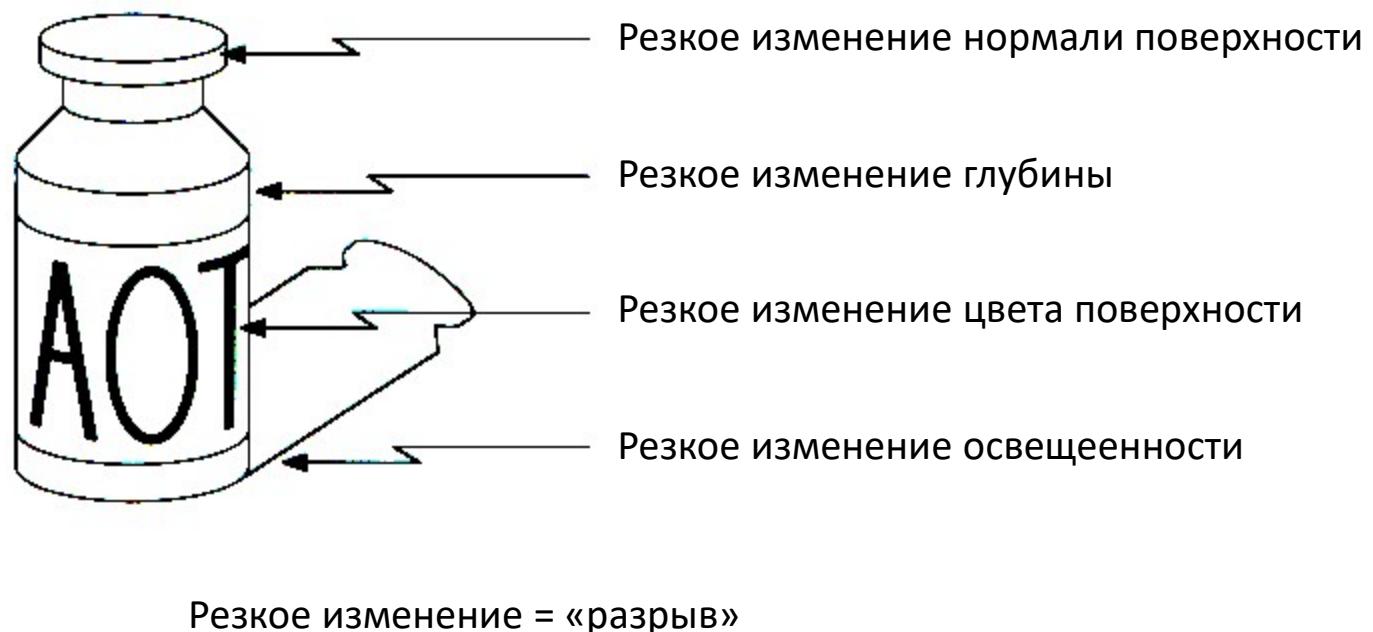
---



- Будем учитывать только часть, но очень важную, для распознавания шаблонов
- Даже улучшим обобщающую способность

# Откуда берутся границы

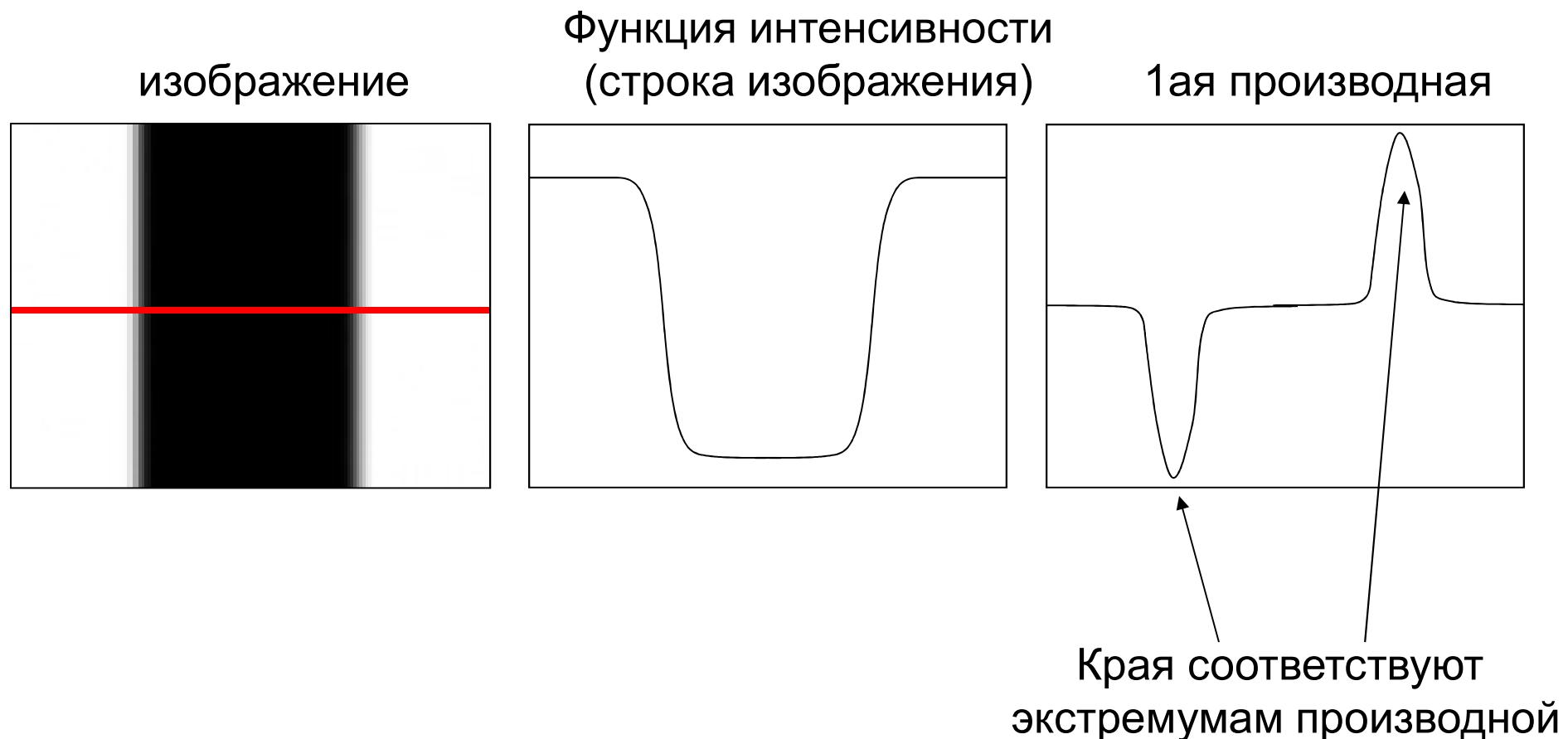
---



- Существует множество причин формирования границ на изображении

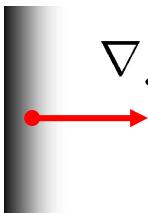
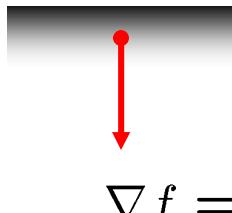
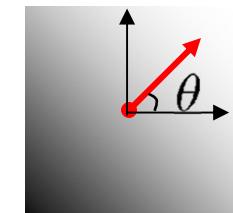
# Описание «края»

- Край – это точка резкого изменения значений функции интенсивности изображения



# Градиент изображения

---

- Градиент изображения:  $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$
- $\nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right]$  
- $\nabla f = \left[ 0, \frac{\partial f}{\partial y} \right]$  
- $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$  

Градиент направлен в сторону наибольшего изменения  
интенсивности

Направления градиента задается как:  $\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$

- Как направление градиента соответствует направлению края?
- Сила края задается величиной (нормой) градиента:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

# Дифференцирование и свёртка

---

- Для функции 2х переменных,  $f(x,y)$ :

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \rightarrow 0} \left( \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)$$

- Разностная производная:

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

- Линейная и инвариантная к переносу, поэтому м.б.  
Результатом свертки

- Свёртка!

|    |   |
|----|---|
| -1 | 1 |
|----|---|

# Вычисление градиента

---

Семейство методов основано на приближенном вычислении градиента, анализе его направления и абсолютной величины. Свертка по функциям:

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

Робертса

Превитт

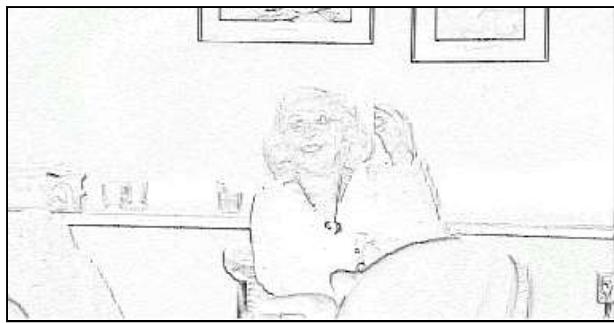
Собеля

Математический смысл – приближенное вычисление производных по направлению

# Примеры карты силы краев

---

Примеры:



Робертса



Превитт

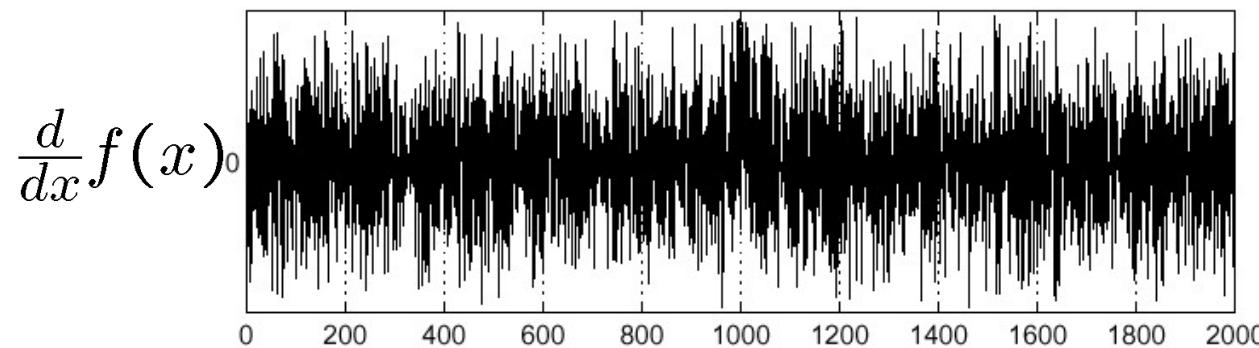
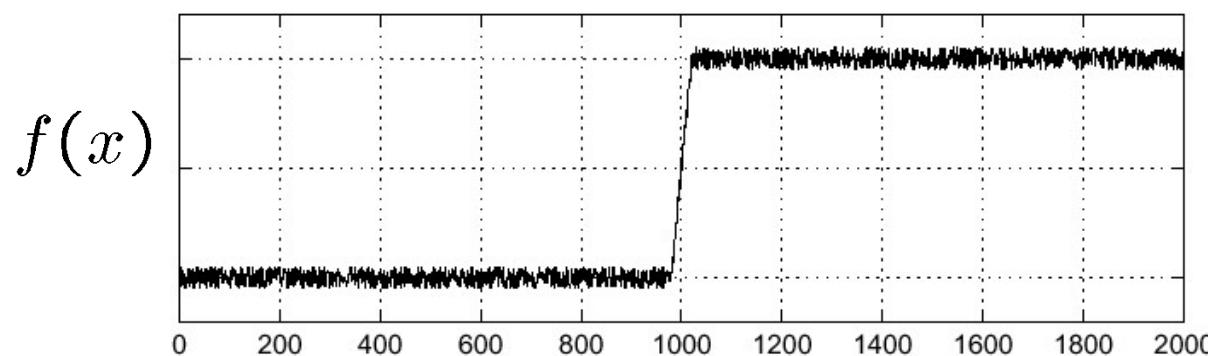


Собеля

# Влияние шума

---

- Рассмотрим строку или столбец изображения
  - Интенсивность от положения можно рассматривать как сигнал



Край исчез

Source: S. Seitz

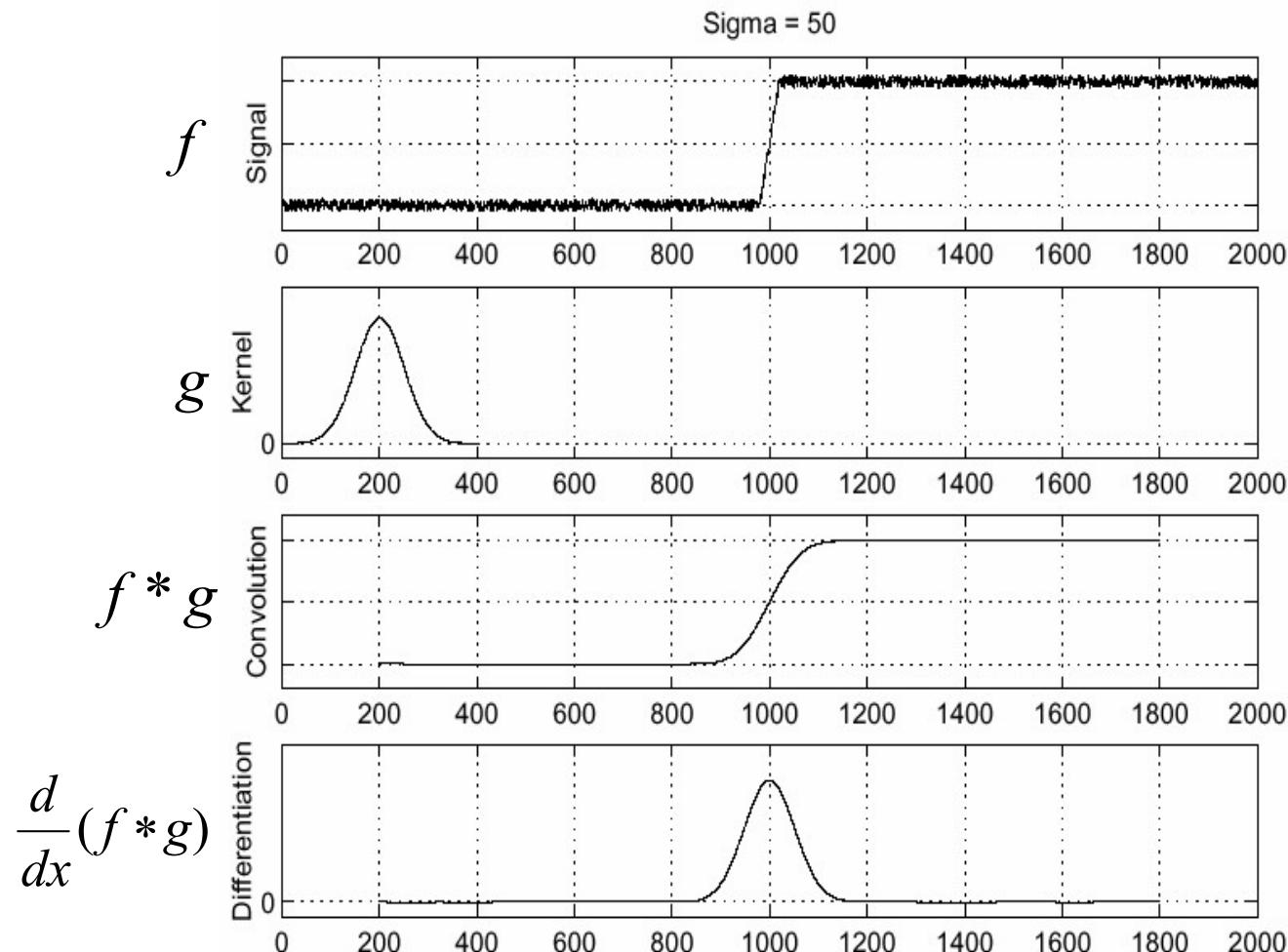
# Влияние шума

---

- Разностные производные очень чувствительны к шуму
  - Зашумленные пиксели отличаются от соседей
  - Чем сильнее шум, тем выше отклик
- Сглаживание
  - Сглаживание делает все пиксели (зашумленные?) чуть более похожими на соседей

# Предобработка (сглаживание)

---

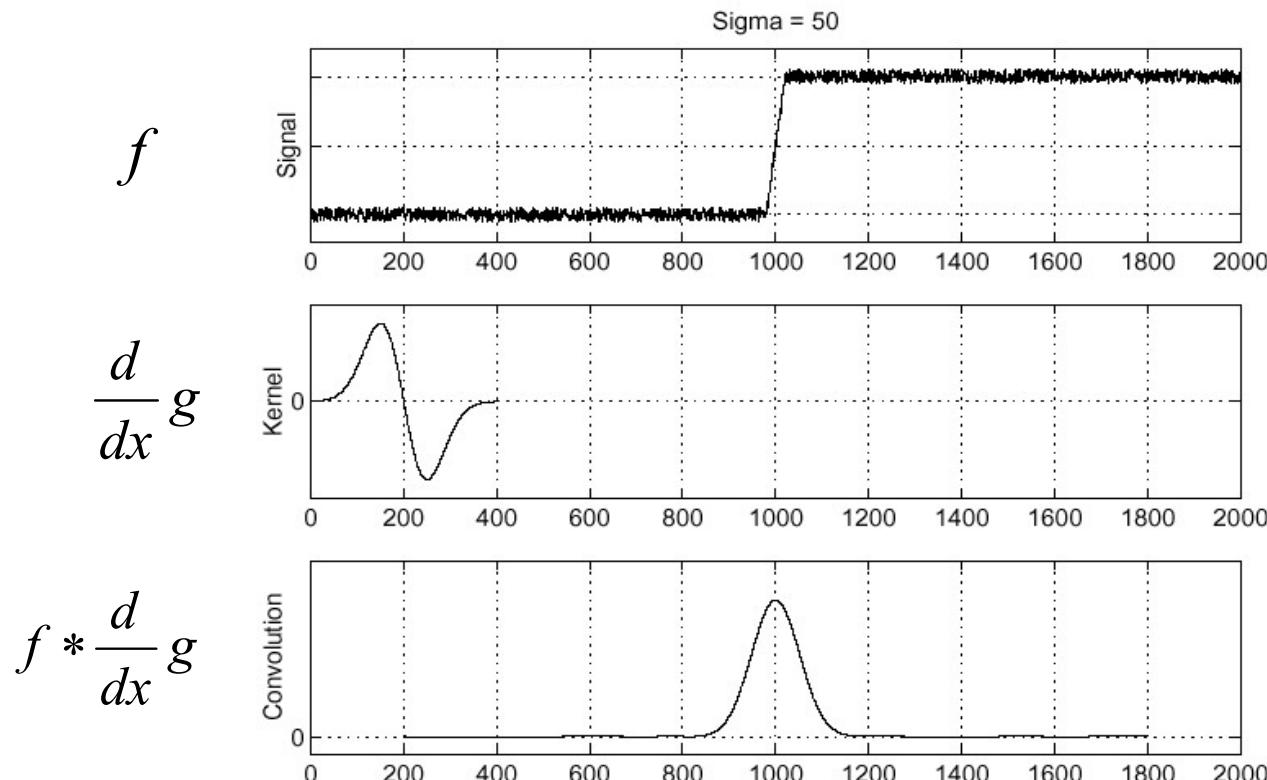


- Для поиска краев ищем пики в:  $\frac{d}{dx}(f * g)$

Source: S. Seitz

# Свойства свертки

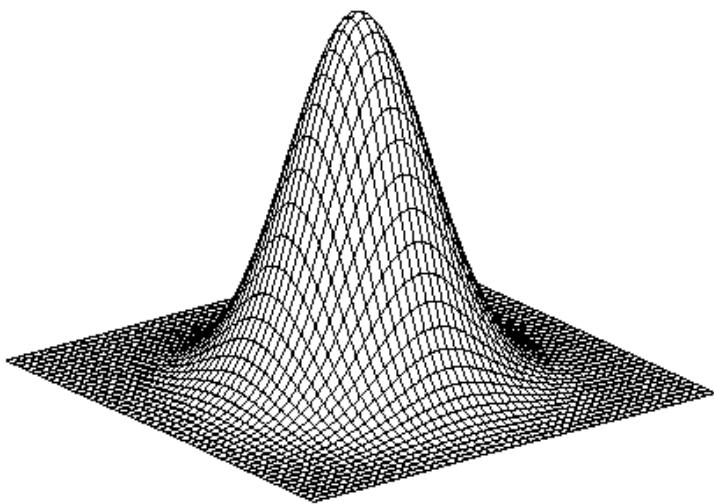
- Операции свертки и дифференцирования ассоциативны:  
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$
- Это экономит 1 операцию:



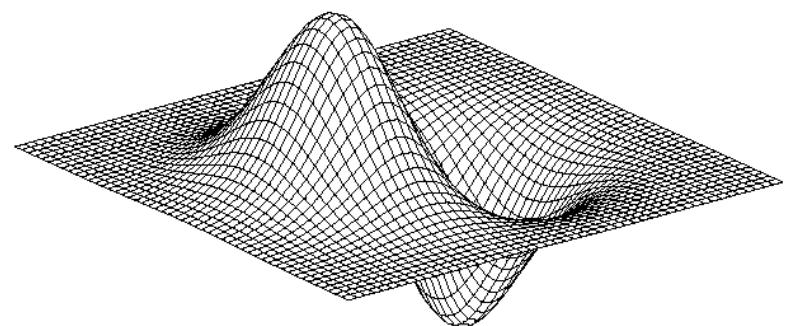
Source: S. Seitz

# Производная фильтра гаусса

---

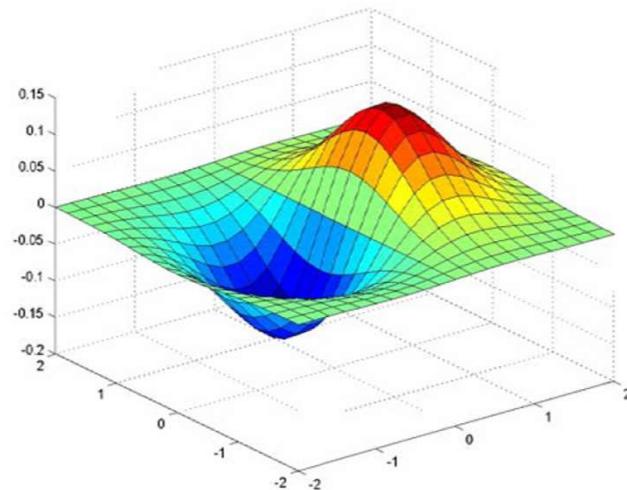


$$* [1 \ -1] =$$

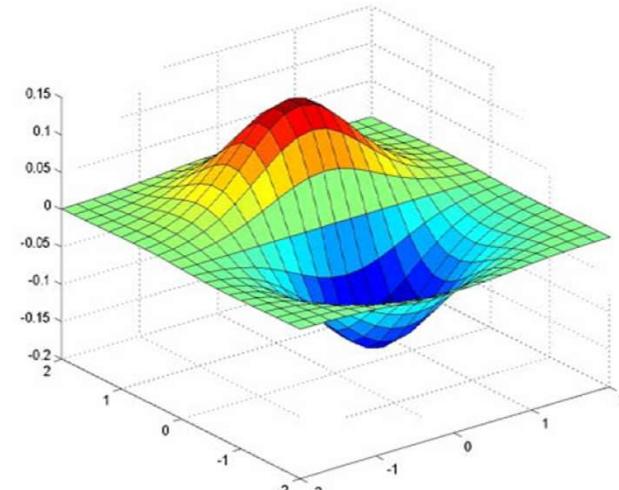


# Производная фильтра гаусса

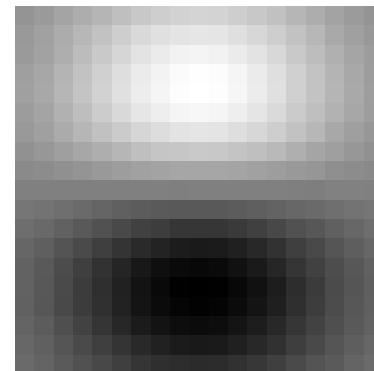
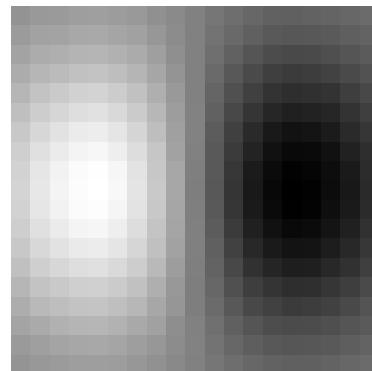
---



x-direction

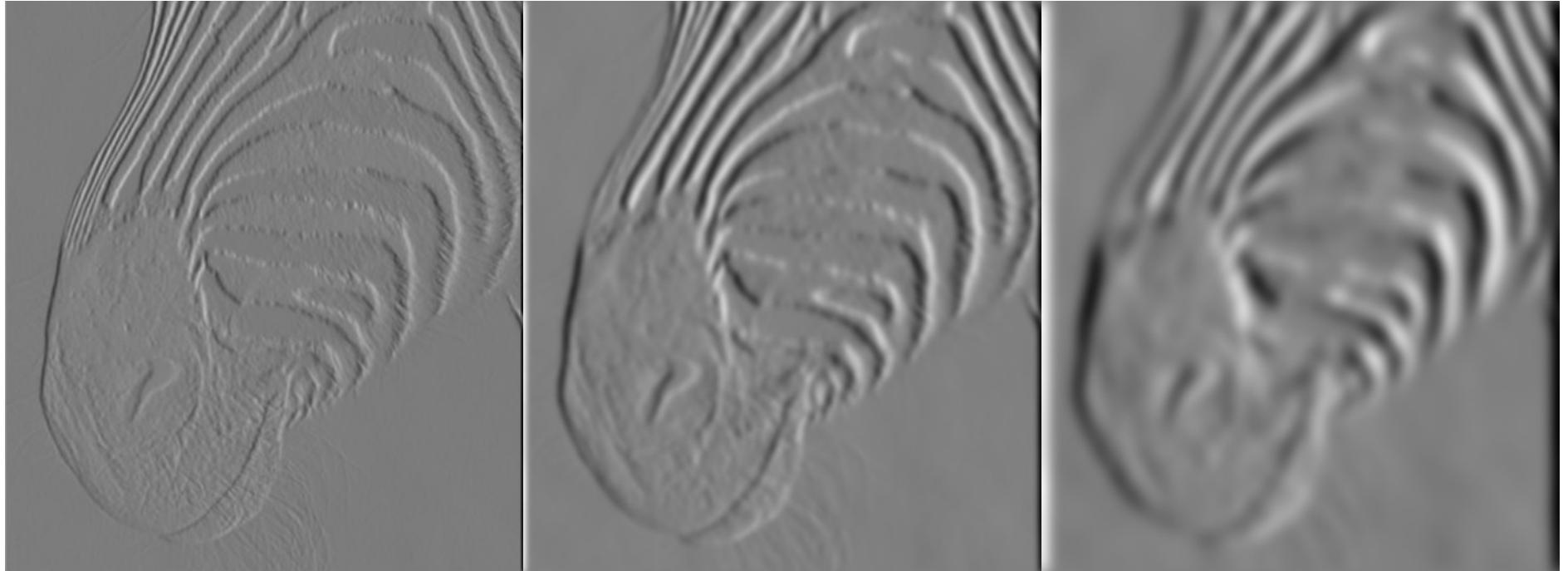


y-direction



# Поиск баланса

---



1 pixel

3 pixels

7 pixels

- Сглаженные производные подавляют шум, но размывают края. Плюс края находится на разных «масштабах»

# Выделение краев

---

- Вычисление градиента – это еще не всё...



Исходное изображение



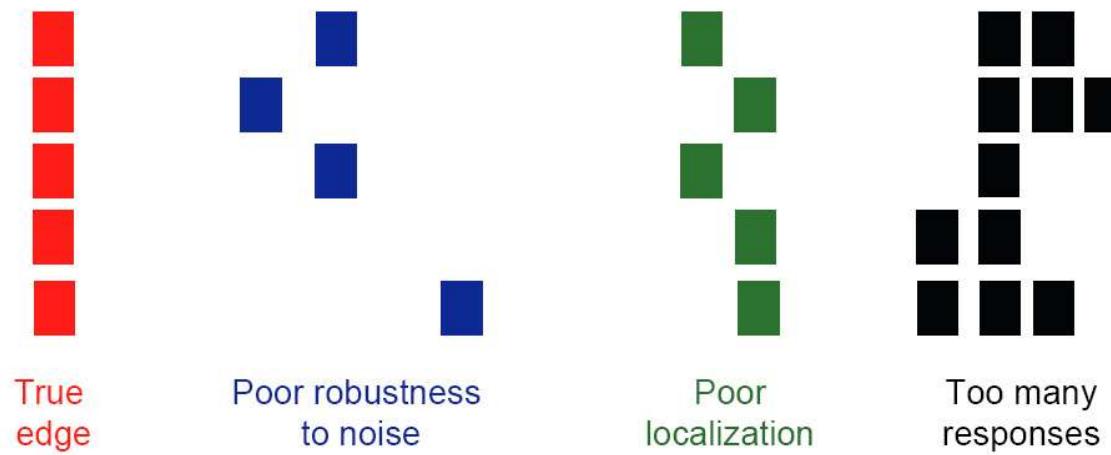
Карта силы краев

- Чего не хватает?
  - Точности – края «толстые» и размытые
  - Информации о связности

# Разработка детектора краев

---

- Критерии качества детектора:
  - **Надежность:** оптимальный детектор должен редко ошибаться (ложные края и пропущенные края)
  - **Точная локализация:** найденный край должен быть как можно ближе к истинному краю
  - **Единственный отклик:** детектор должен выдавать одну точку для одной точки истинного края, т.е. локальных максимум вокруг края должно быть как можно меньше
  - **Связанность:** хотим знать, какие пиксели принадлежат одной линии края



Source: L. Fei-Fei

# Детектор Canny

---

1. Свертка изображения с ядром – производной от фильтра гаусса
2. Поиск нормы и направления градиента
3. Выделение локальных максимумов (Non-maximum suppression)
  - Уточнение полос в несколько пикселей до одного пикселя
4. Связывание краев и обрезание по порогу (гистерезис) :
  - Определяем два порога: нижний и верхний
  - Верхний порог используем для инициализации кривых
  - Нижний порог используем для продолжения кривых

```
from skimage import feature  
edges1 = feature.canny(im)
```

# Пример

---



- Исходное изображение (Lena)

# Пример

---



Норма градиента

# Пример

---



Отсечение по порогу

# Пример

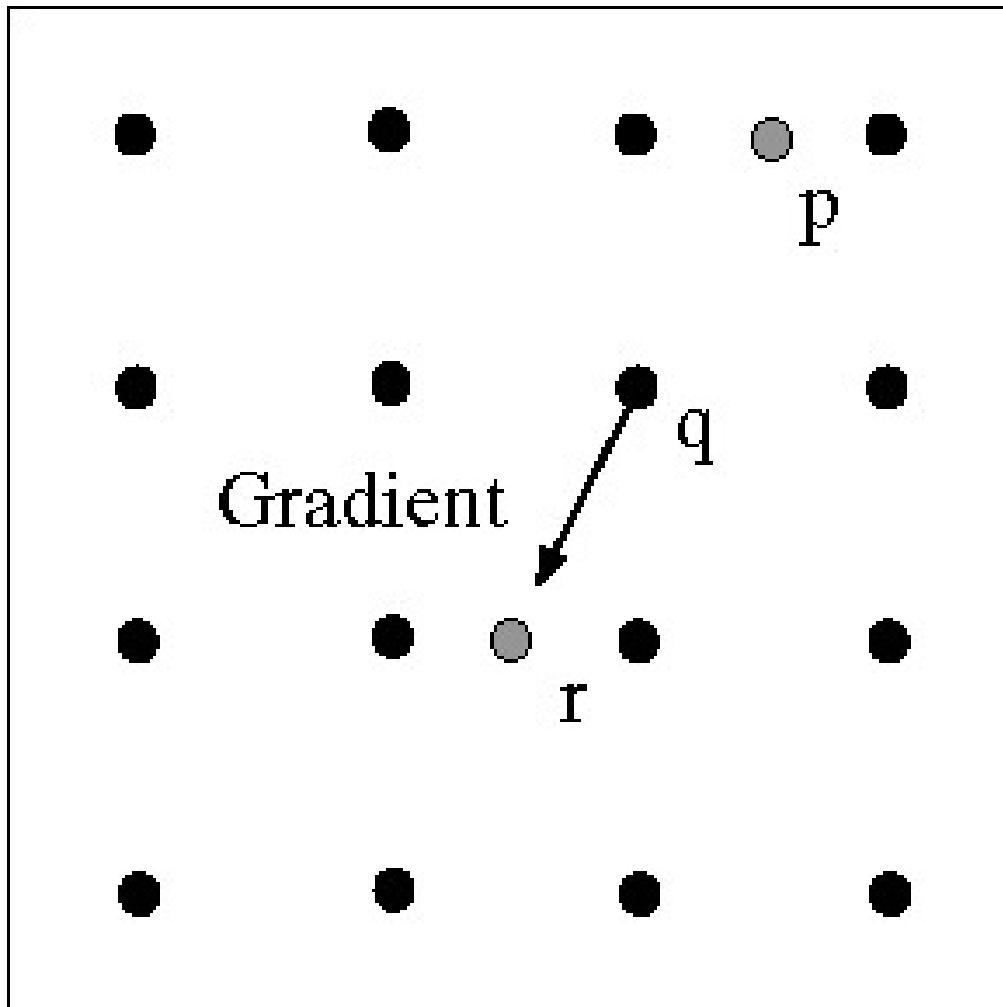
---



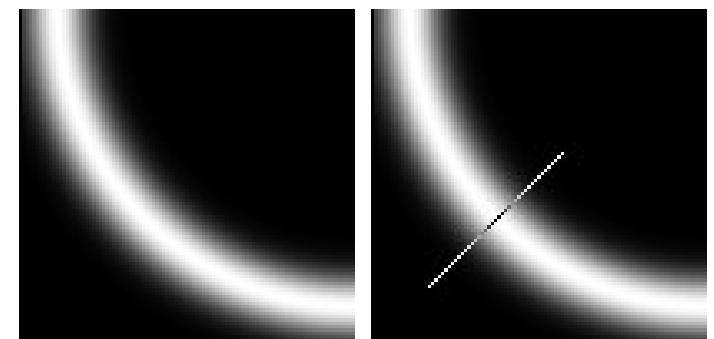
Утоньшение  
(non-maximum suppression)

# Поиск локальных максимумов

---

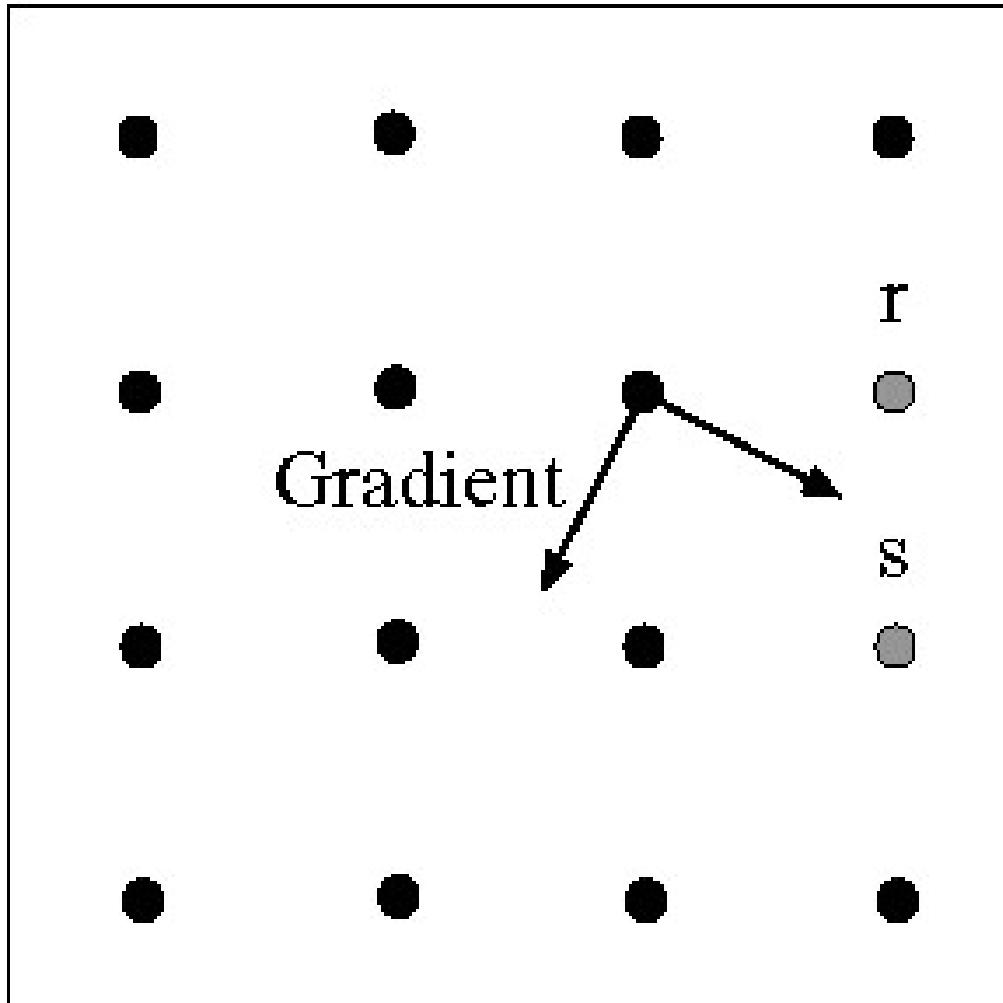


Максимум  
достигается в  $q$ ,  
если значение  
больше  $p$  и  $r$ .  
Значения в  $p$  и  $r$   
интерполируем.

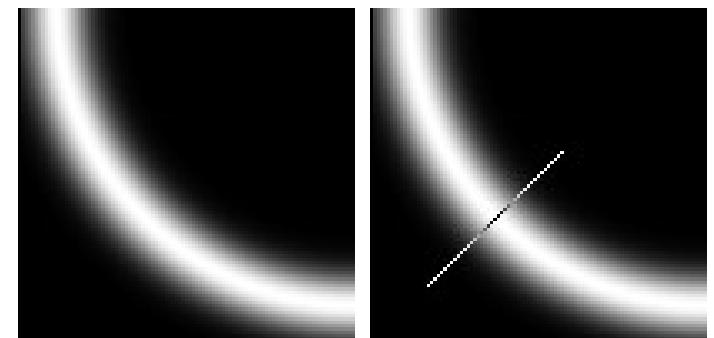


# Связывание точек

---



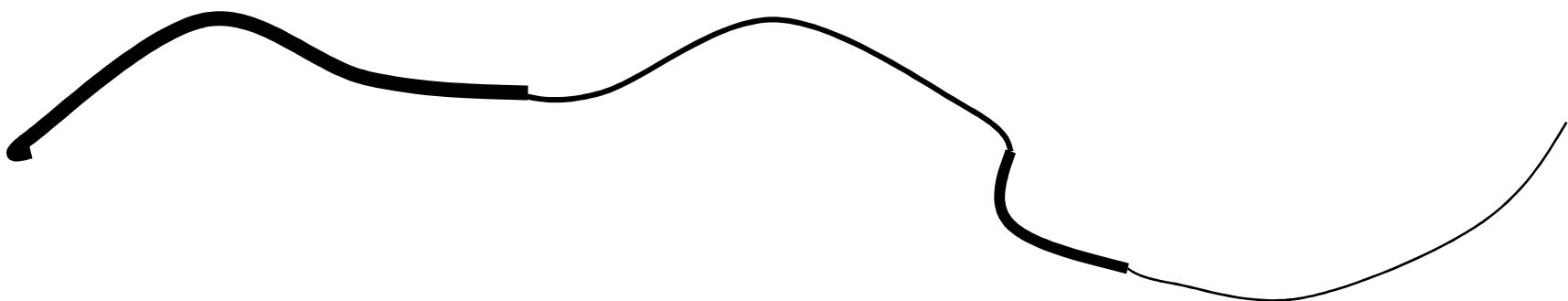
Пусть отмеченная точка – край. Строим касательную к границе (нормаль к направлению градиента) и используем ее для предсказания новой точки (это либо  $s$  либо  $r$ ).



# Отсечение по порогу

---

- Проверяем точку, чтобы значение градиента было выше порога
  - Используем **гистерезис**
    - Большой порог для начала построения кривой и низкий порог для продолжения края (связывания)



# Эффект гистерезиса

---



Исходное изображение



Высокий порог  
(сильные края)



Низкий порог  
(слабые края)

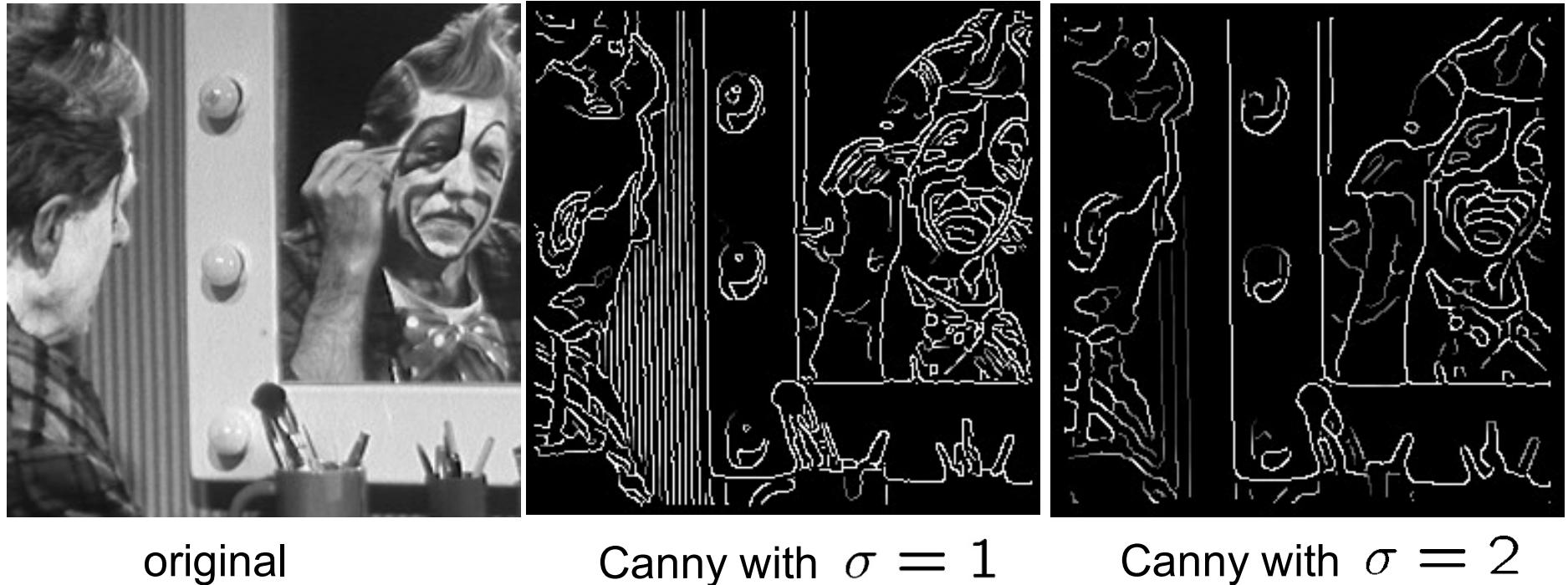


Порог по гистерезису

Source: L. Fei-Fei

# Влияние $\sigma$

---



original

Canny with  $\sigma = 1$

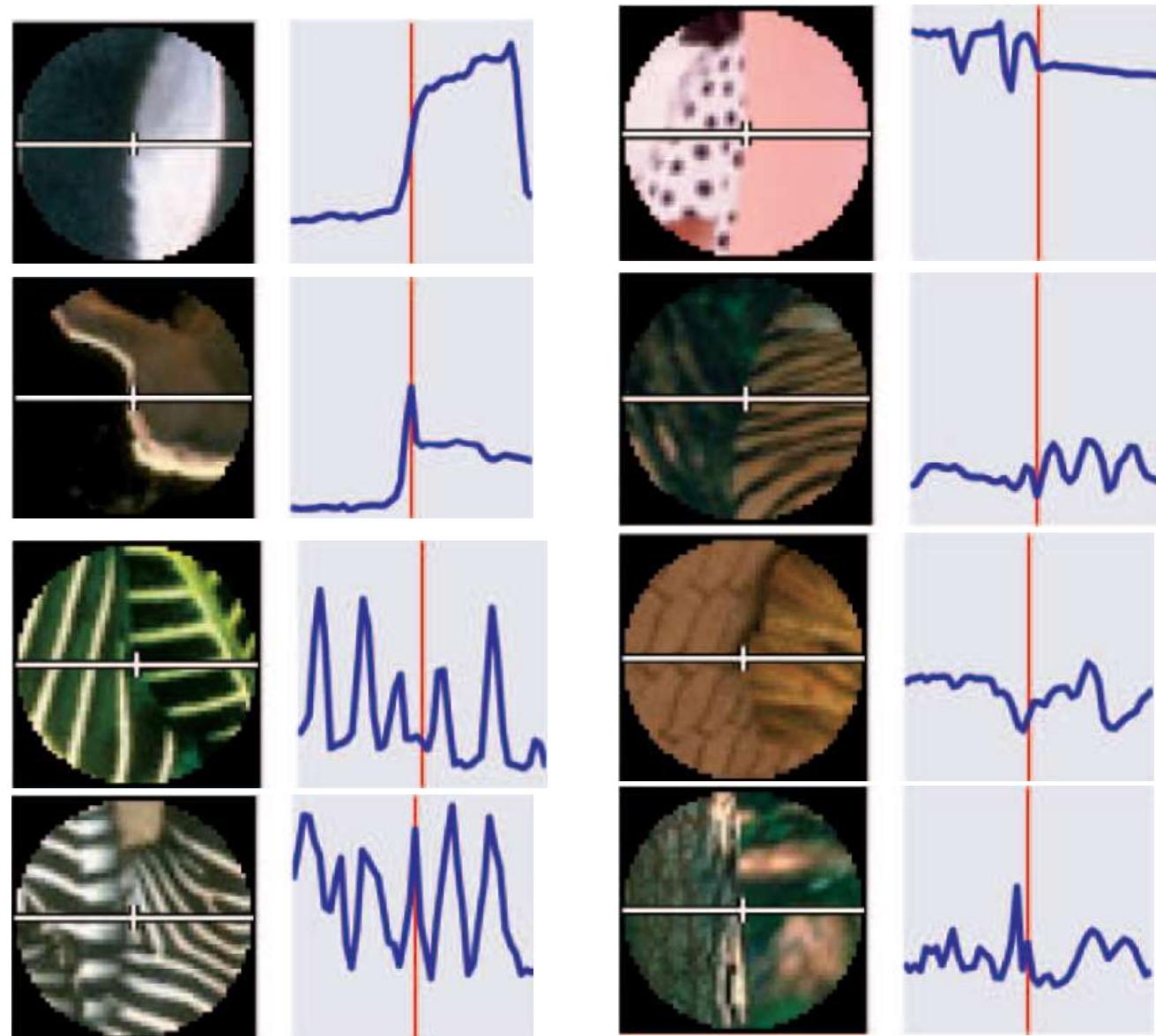
Canny with  $\sigma = 2$

Выбор  $\sigma$  (размера ядра размытия) зависит от задачи

- большое  $\sigma$  - поиск крупных границ
- малая  $\sigma$  - выделение мелких деталей

# Ограничения детектора

---



Source: Martin et al. 2003

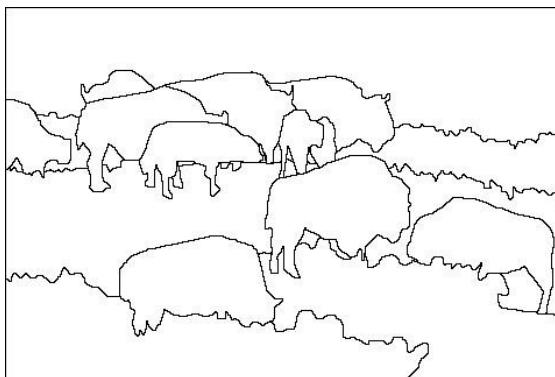
# Поиск краев – это только начало...

---

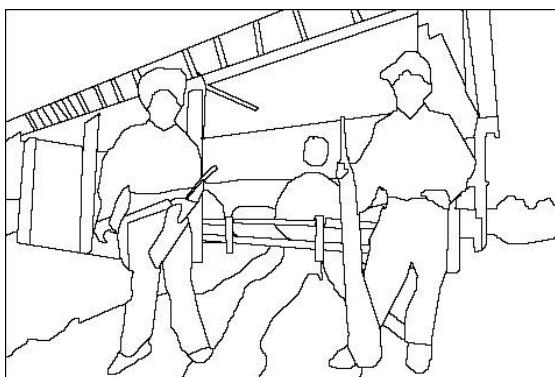
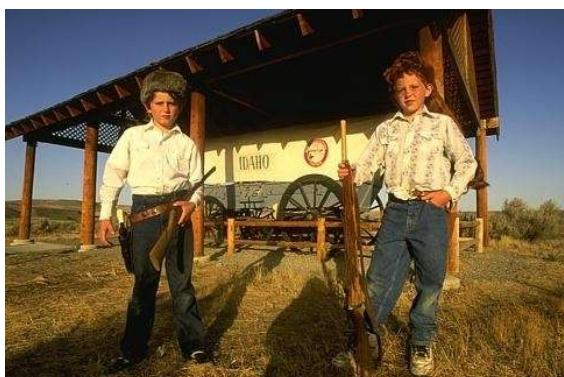
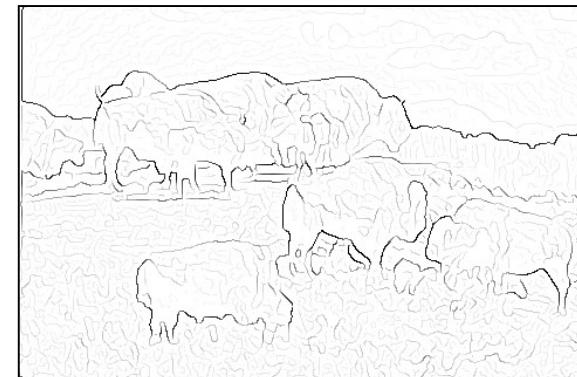
image



human segmentation



gradient magnitude

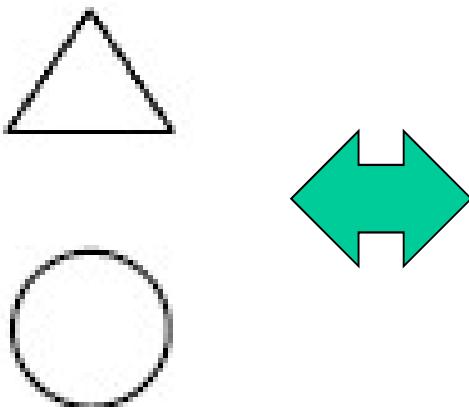


- Berkeley segmentation database:

<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/>

# Края для сопоставления шаблонов

---



- Получили карту краёв шаблона и изображения
- Как их сравнить друг с другом?
  - Просто попиксельно явно не оптимально

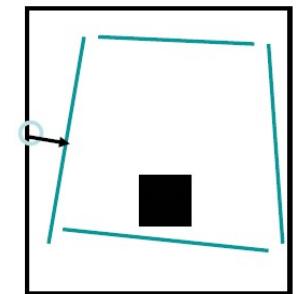
# Метрики

---

- Chamfer Distance

- Для каждого пикселя  $a$  края шаблона  $A$  вычисляем расстояние до ближайшего пикселя  $b$  края изображения  $B$

$$r(a, B) = \min_{b \in B} \|a - b\|$$



- Суммируем все найденные расстояния

$$ChDist(A, B) = \sum_{a \in A} \min_{b \in B} \|a - b\|$$

- Hausdorff Distance

- Почти то же самое, но берём не сумму, а максимальное расстояния

$$HausDist(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

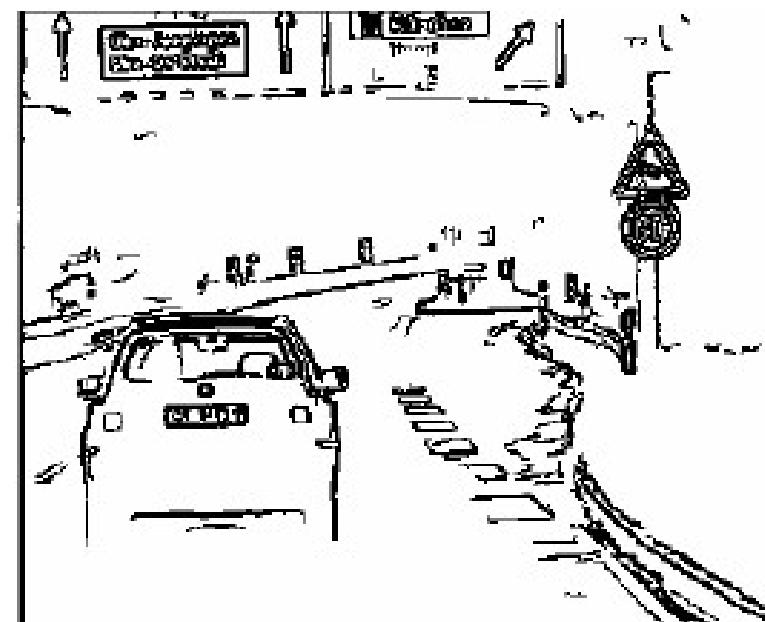
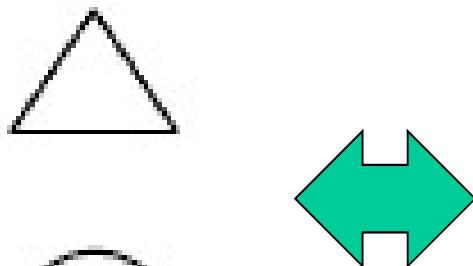
# Метрики

---

- Свойства метрик
  - Chamfer требует нормализации, Hausdorff нет
  - Chamfer симметрична, Hausdorff нет
    - $\text{HausDist}(A,B) \neq \text{HausDist}(B,A)$
  - Можно использовать не max, а медиану (медленнее)
- Какую метрику использовать?
  - Обычно заранее сказать нельзя, нужна экспериментальная проверка

# Поиск ближайших пикселей края

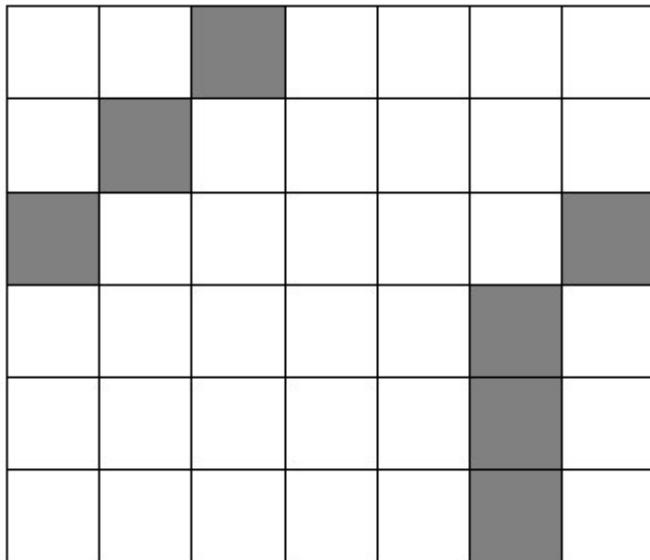
---



Вопрос: как найти ближайший пиксель края на изображении?

# Distance Transform

---



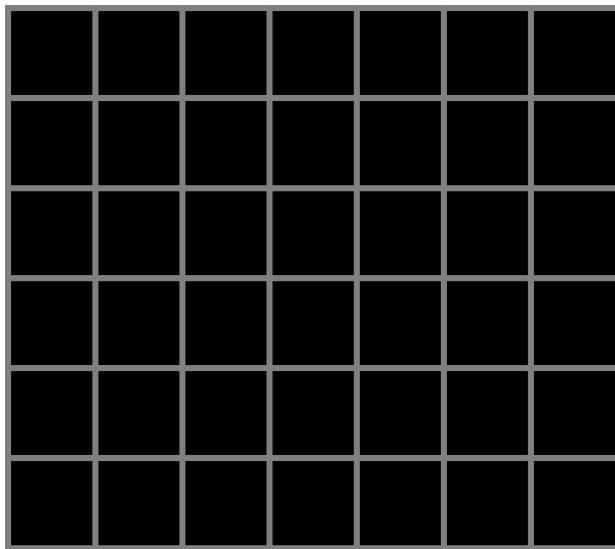
|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 2 | 1 | 0 | 1 | 2 | 3 | 2 |
| 1 | 0 | 1 | 2 | 3 | 2 | 1 |
| 0 | 1 | 2 | 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 2 | 1 | 0 | 1 |
| 2 | 3 | 3 | 2 | 1 | 0 | 1 |
| 3 | 4 | 3 | 2 | 1 | 0 | 1 |

Для каждого пикселя вычисляется расстояние до ближайшего пикселя края

`scipy.ndimage.morphology.distance_transform_edt`

# Применение DT

---



|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 2 | 1 | 0 | 1 | 2 | 3 | 2 |
| 1 | 0 | 1 | 2 | 3 | 2 | 1 |
| 0 | 1 | 2 | 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 2 | 1 | 0 | 1 |
| 2 | 3 | 3 | 2 | 1 | 0 | 1 |
| 3 | 4 | 3 | 2 | 1 | 0 | 1 |

- Совмещаем шаблон и карту DT
- Вычисляем ошибку, суммируя все значения в пикселях краев

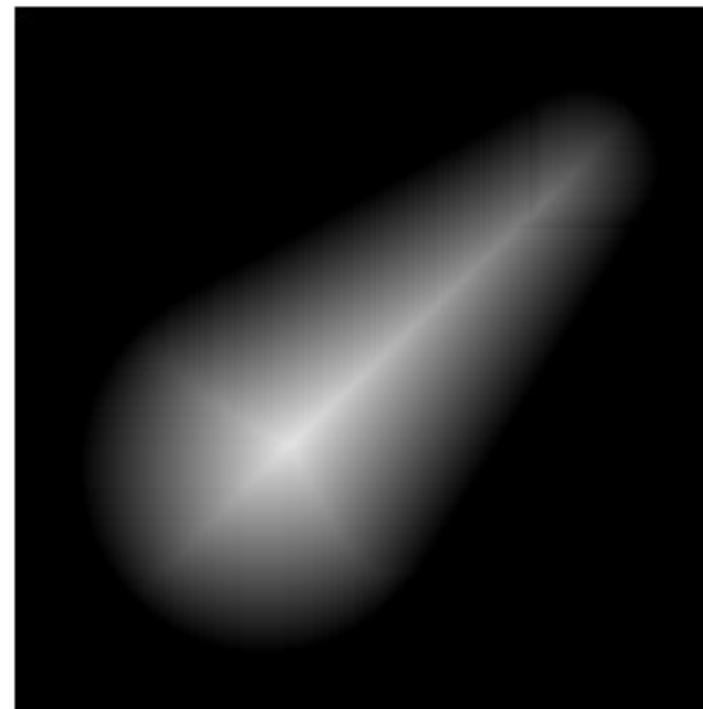
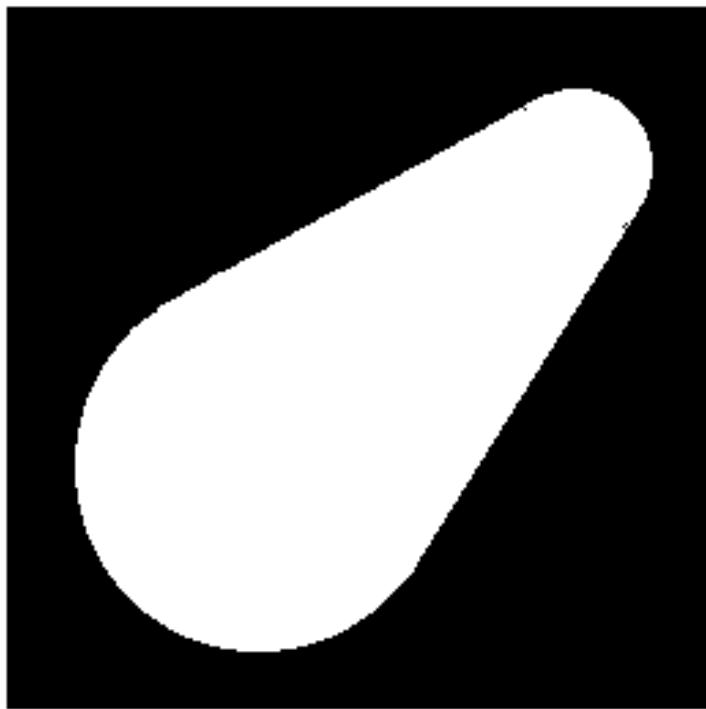
# Вычисление DT

---

- Простейший алгоритм – N проходов
  - Первый проход помечает края 0
  - На втором помечаем все граничащие с 0 пиксели как 1
  - И т.д.
- Существует двухпроходный алгоритм

## Пример DT

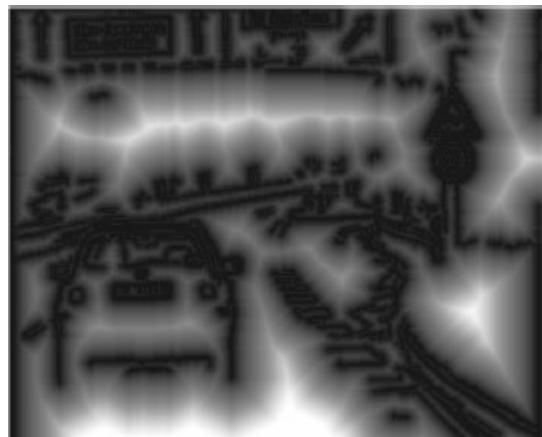
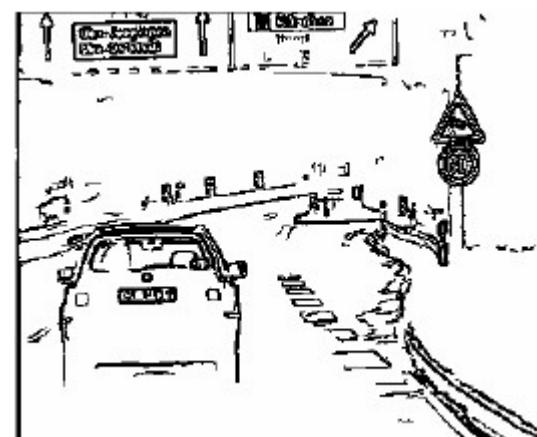
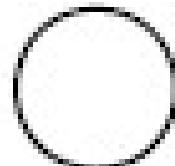
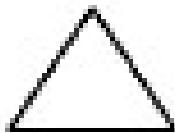
---



DT может использоваться для поиска «скелета»  
– осей объекта

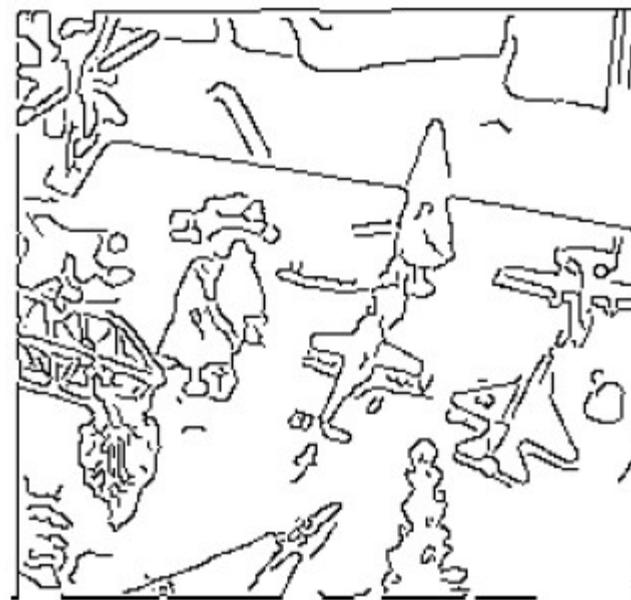
# Пример поиска с помощью DT

---



# Пример

---



# Резюме сопоставления шаблонов

---

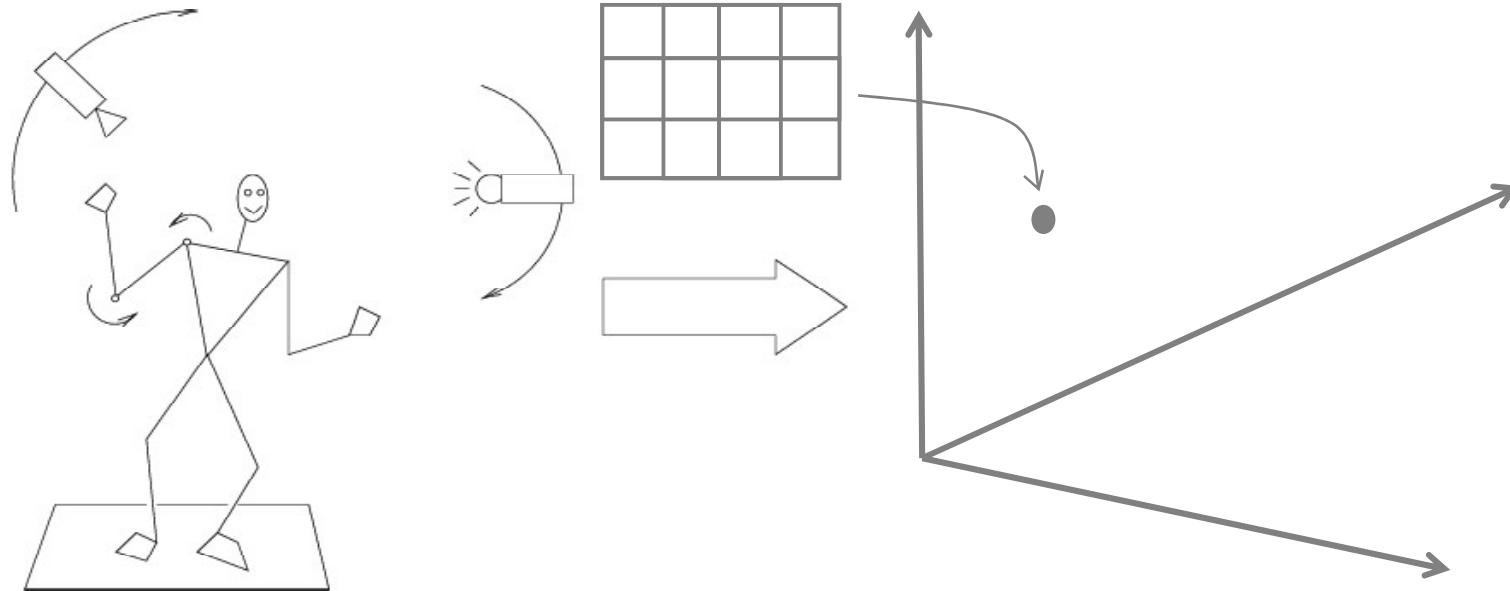
- Подходит в тех случаях, когда объекты фиксированы и модель преобразования не очень сложная
  - Цифры на знаках
  - Цифры на конвертах
  - Аэрофотосъёмка / Космическая съёмка
- Не очень быстрые методы
  - Требуются специальные процедуры для ускорения, пр. отбраковка ложных фрагментов по упрощённым критериям и т.д.



Номера

# Инвариантность

---



~~Изменчивость~~

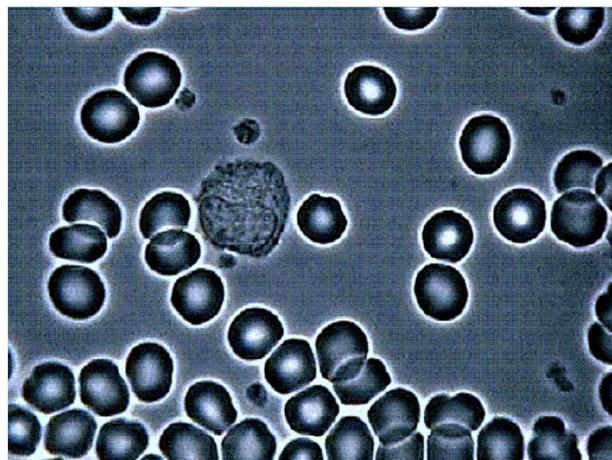
**Инвариантность**  
к:

Положение камеры  
Освещение  
Внутренние параметры

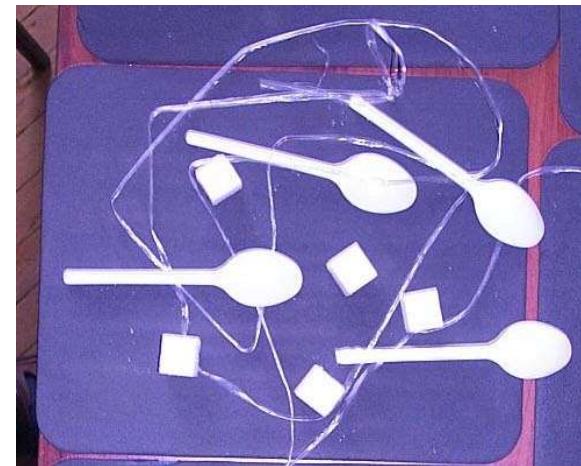
Duda & Hart ( 1972); Weiss (1987); Mundy et al. (1992-94);  
Rothwell et al. (1992); Burns et al. (1993)

# Примеры

---



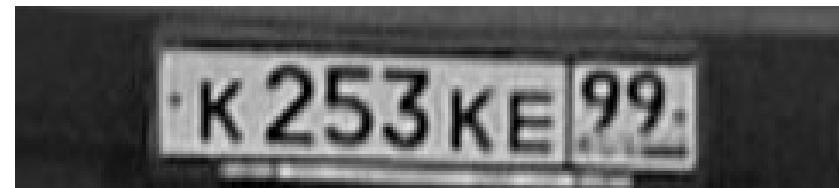
Клетки крови



Ложки и сахар



Монеты и купюры

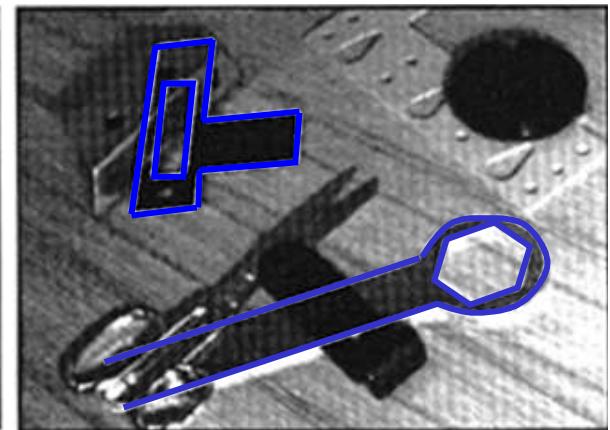
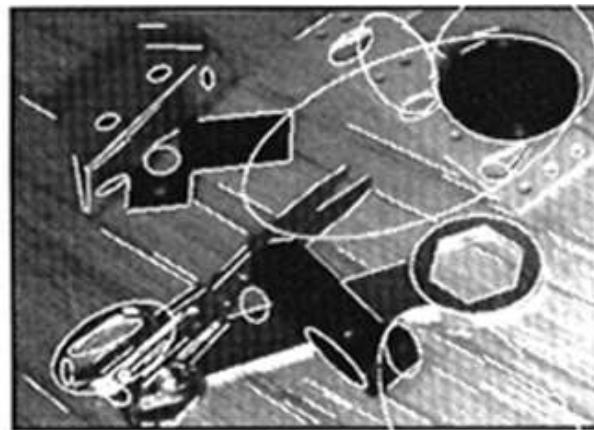
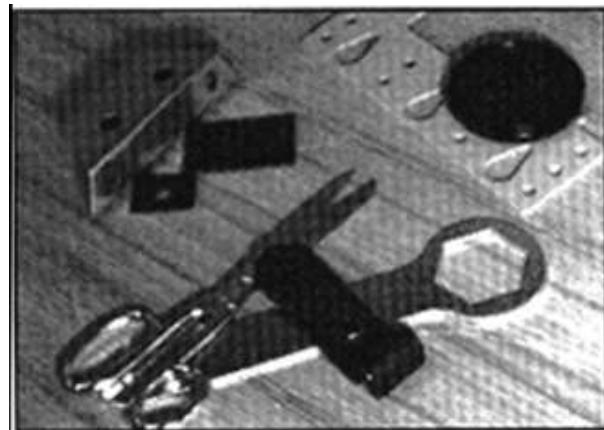
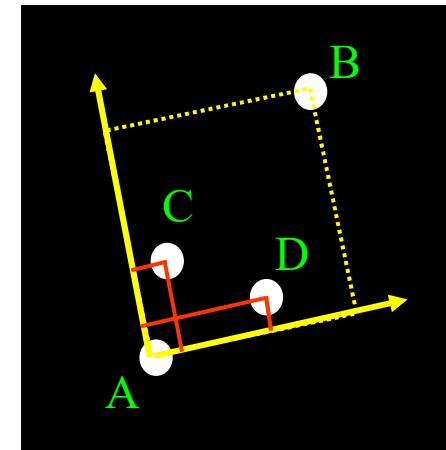


Номера

Контрастные объекты на фоне!

# Более сложные примеры

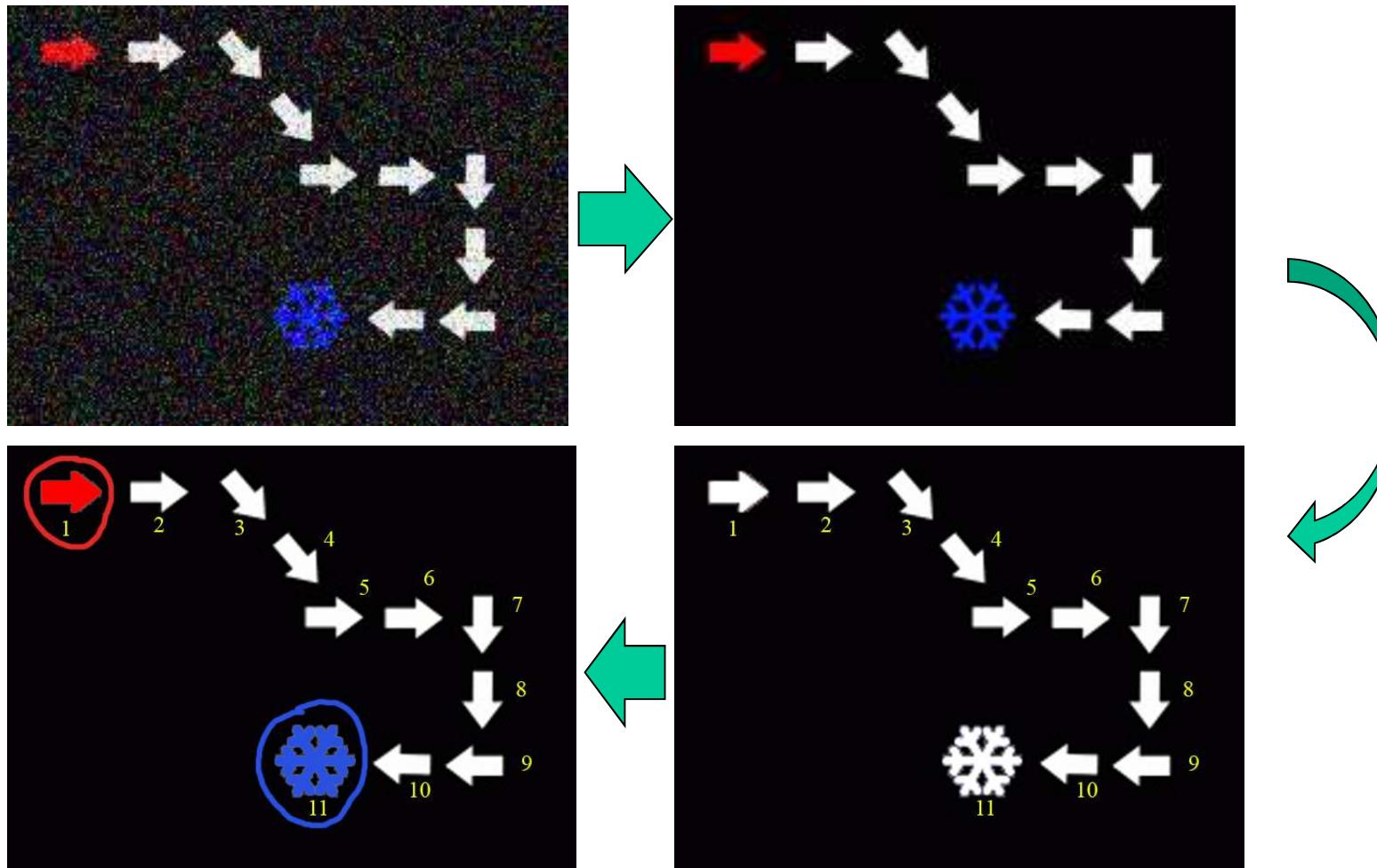
Инвариантность к  
перспективным искажениям –  
проективные инварианты  
(Rothwell et al., 1992)



В общем случае, для 3D объектов не существует проективных  
инвариантов (Burns et al., 1993)

# Схема простого алгоритма

---



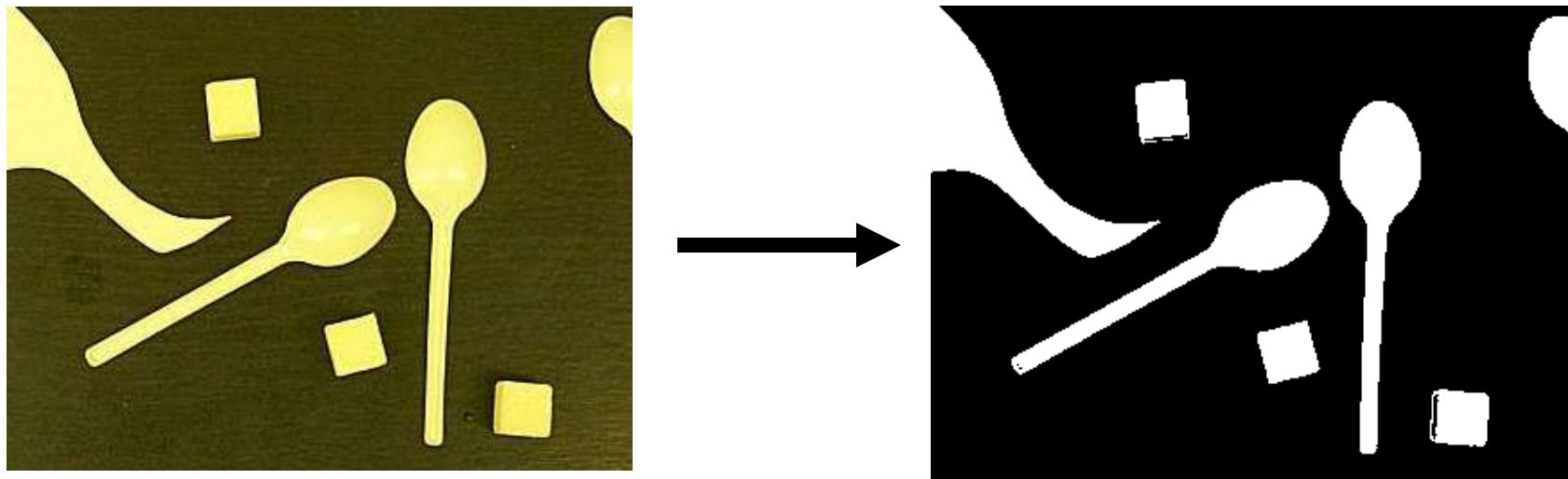
# Схема простого алгоритма

---

- Предобработка изображения для упрощения анализа (например – шумоподавление)
- Выделение на изображении контрастных областей-кандидатов в которых может находится искомый объект
- Вычисление признаков (инвариантов) по выделенным фрагментам
- Проверка – является ли фрагмент изображения изображением нужного нам объекта по измеренным параметрам

# Бинаризация изображений

---

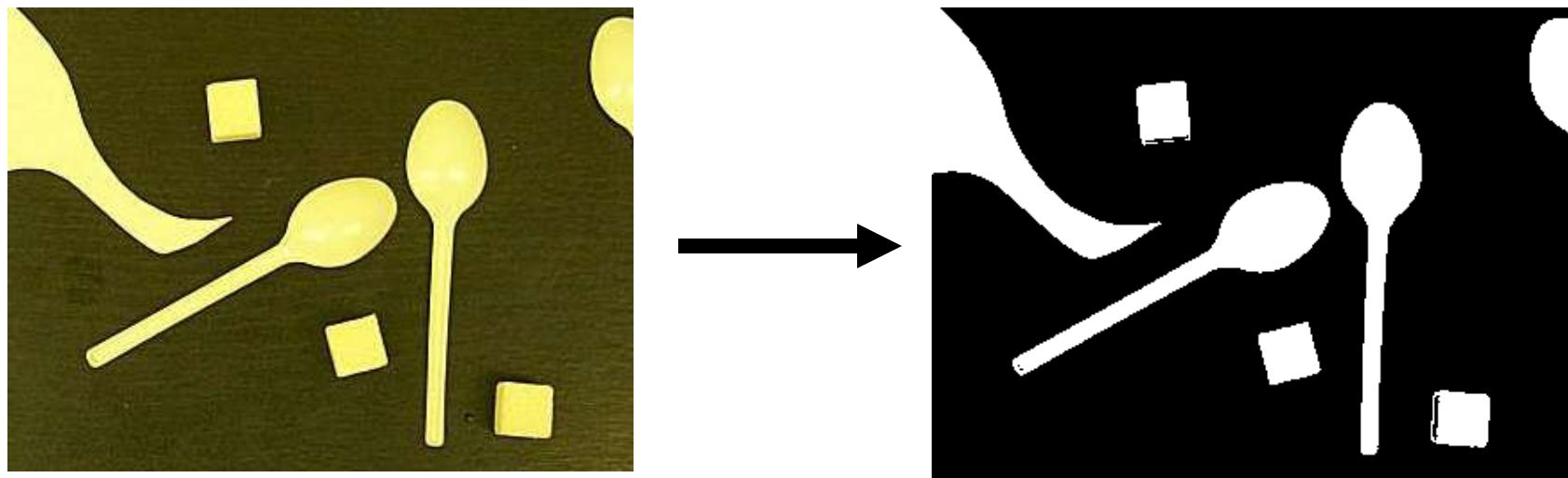


- Пиксель бинарного изображения может принимать только значения 0 и 1
- Бинаризация – построение бинарного изображения по полутоновому / цветному
- Смысл?
  - Разделить изображение на фон и контрастные объекты
  - Объекты помечены 1, фон 0

# Пороговая фильтрация

---

- Простейший вариант - пороговая фильтрация (thresholding)
  - Выделение областей, яркость которых выше/ниже некоторого порога, заданного «извне»

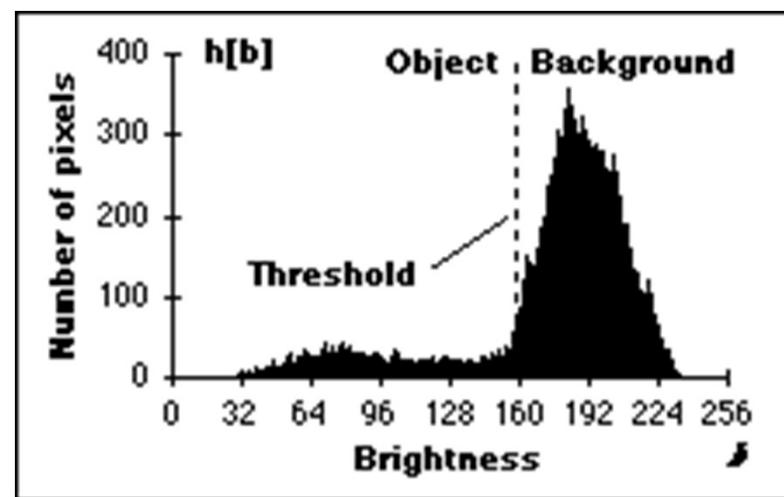
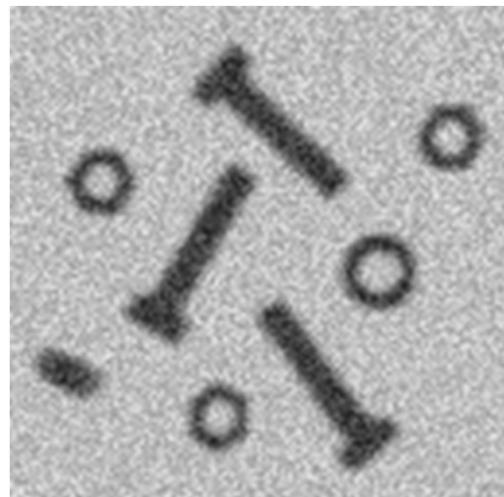


# Пороговая фильтрация

---

Более интересный способ – определение порога автоматически, по характеристикам изображения

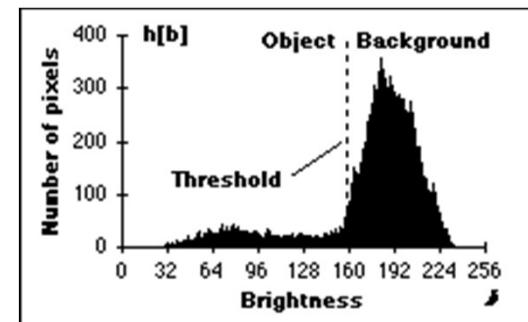
- Анализ гистограммы



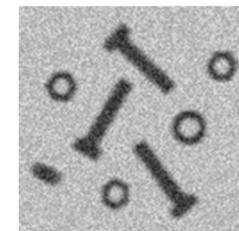
# Анализ гистограммы

---

- Анализ симметричного пика гистограммы
- Применяется когда фон изображения дает отчетливый и доминирующий пик гистограммы, симметричный относительно своего центра.



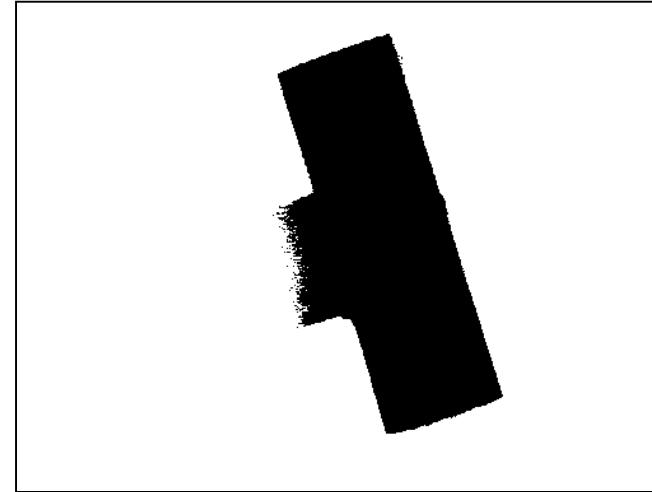
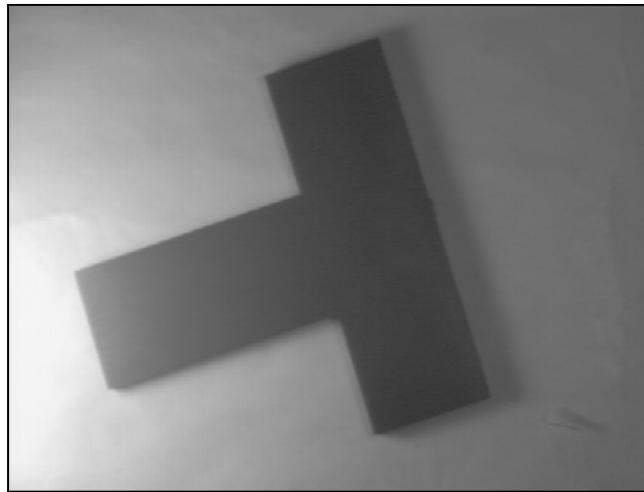
1. Сгладить гистограмму;
2. Найти ячейку гистограммы  $h_{\max}$  с максимальным значением;
3. На стороне гистограммы не относящейся к объекту (на примере – справа от пика фона) найти яркость  $h_p$ , количество пикселей с яркостью  $\geq h_p$  равняется  $p\%$  (например 5%) от пикселей яркости которых  $\geq h_{\max}$ ;
4. Пересчитать порог  $T = h_{\max} - (h_p - h_{\max})$ ;



# Адаптивная бинаризация

---

Необходима в случае неравномерной яркости фона/объекта.



# Адаптивная бинаризация

---

Необходима в случае неравномерной яркости фона/объекта.

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

Варианты выбора  $T$ :

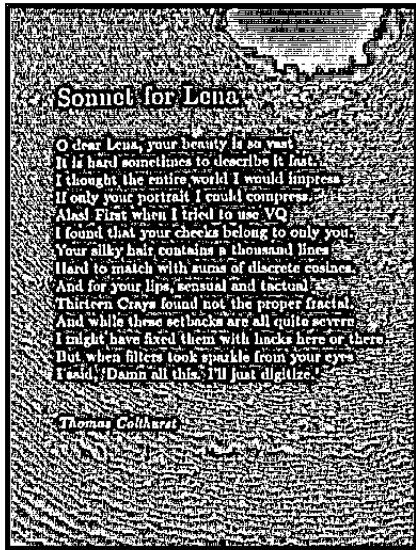
- $T = \text{mean}$
- $T = \text{median}$
- $T = (\min + \max) / 2$

# Адаптивная бинаризация

## 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*



Исходное

r=7, C=0

## 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*

r=7, C=7

## 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.'

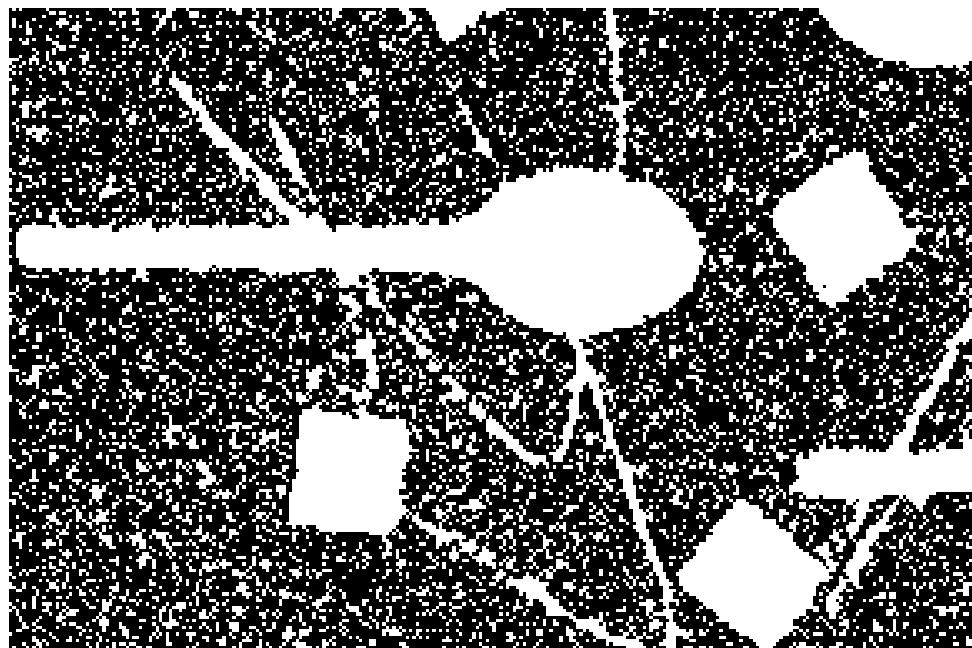
*Thomas Colthurst*

r=75, C=10

# Шум в бинарных изображениях

---

Пример бинарного изображению с сильным шумом

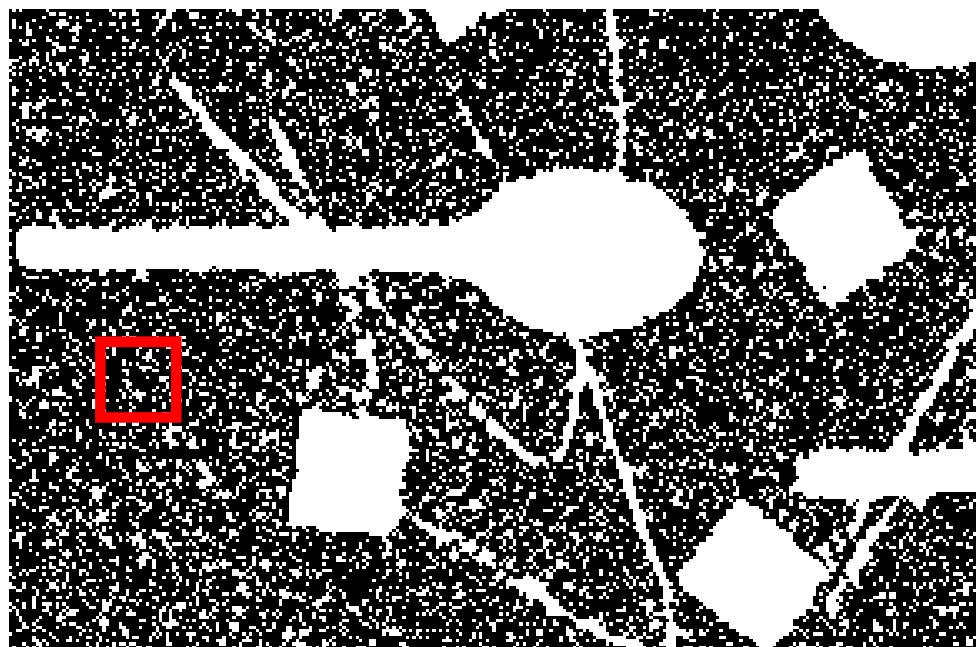


Часто возникает из-за невозможности полностью подавить шум в изображениях, недостаточной контрастности объектов и т.д.

# Шум в бинарных изображениях

---

- ⑩ По одному пикслю невозможно определить – шум или объект?
- ⑩ Нужно рассматривать окрестность пикселя!



# Подавление и устранение шума

---

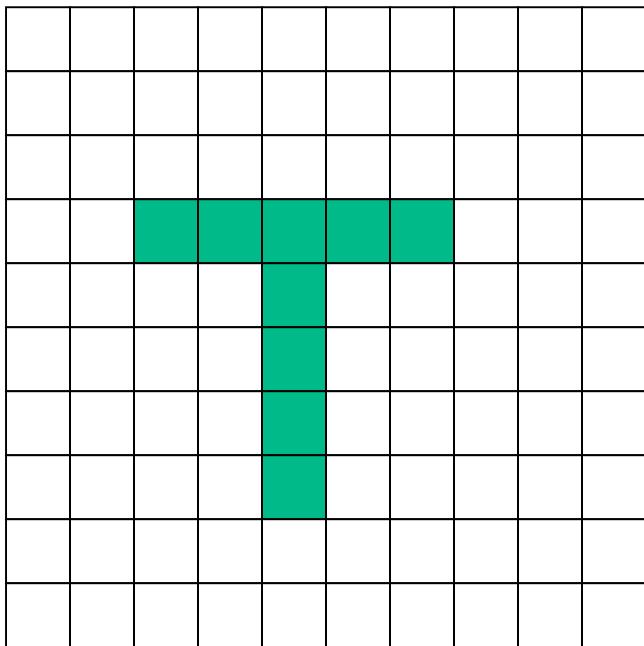
Широко известный способ - устранение шума с помощью операций математической морфологии:

- Сужение (erosion)
- Расширение (dilation)
- Закрытие (closing)
- Раскрытие (opening)

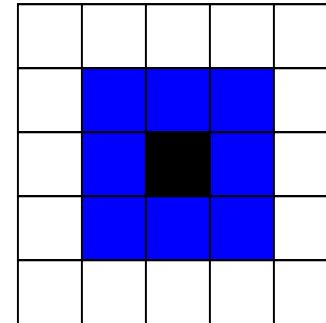
# Математическая морфология

---

A



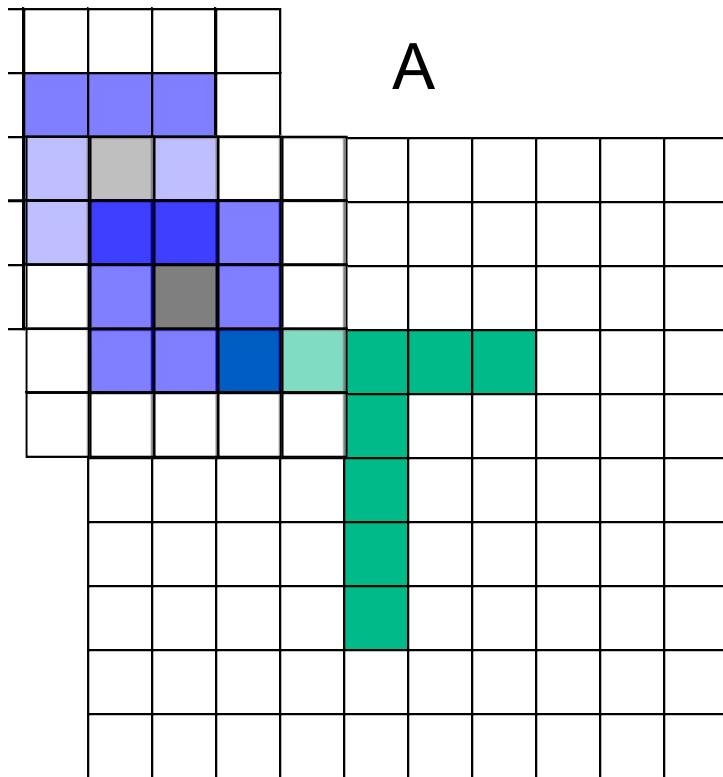
B



Множество А обычно является объектом обработки, а множество В (называемое структурным элементом) – инструментом.

# Расширение в дискретном случае

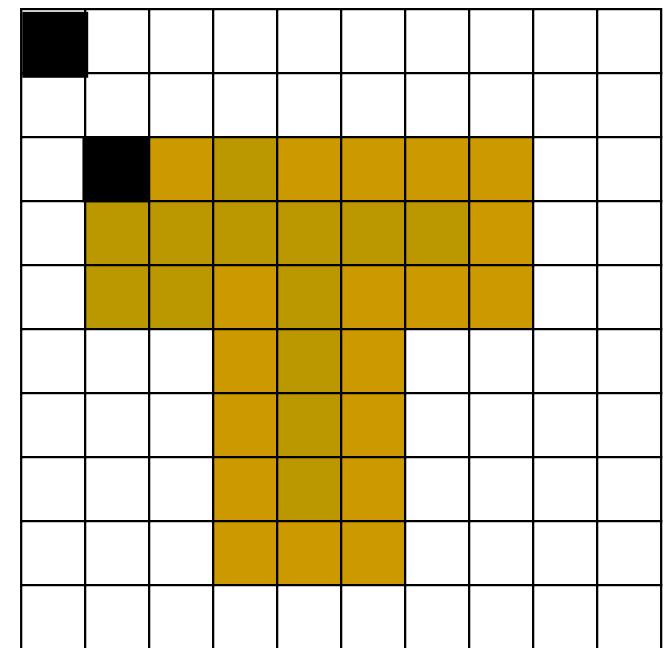
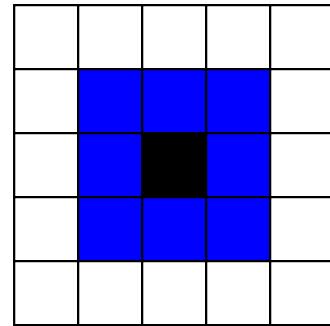
---



A

B

A(+) $B$



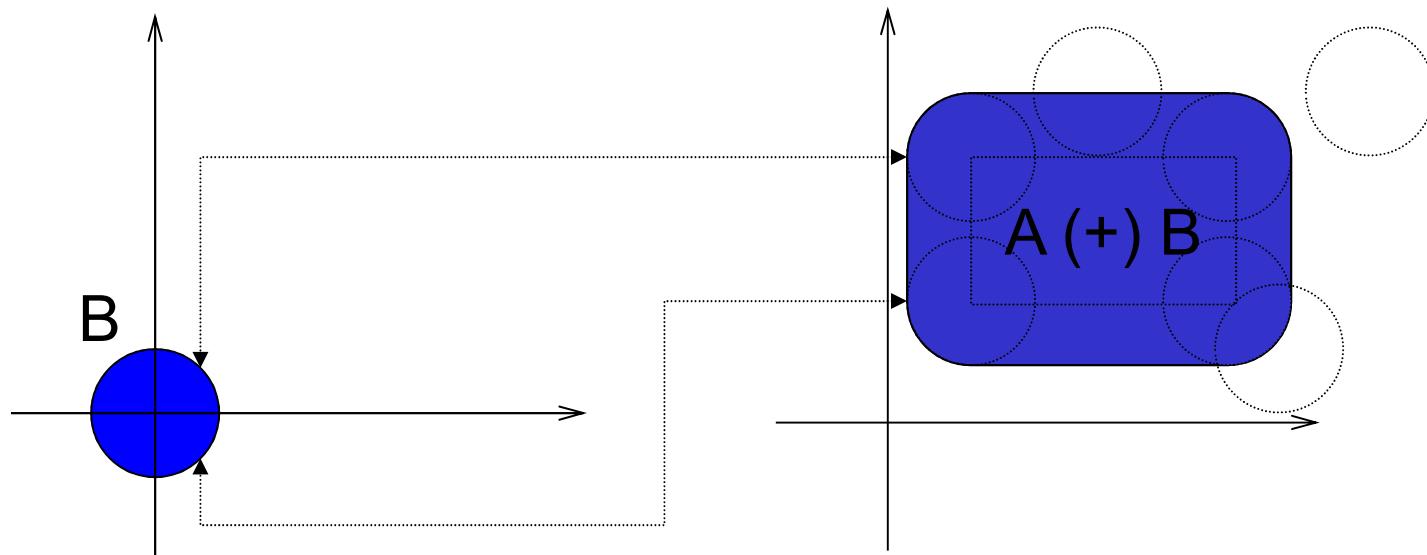
Операция «расширение» - аналог логического «или»

# Расширение

---

Расширение (dilation)

$$A (+) B = \{t \in \mathbb{R}^2: t = a + b, a \in A, b \in B\}$$

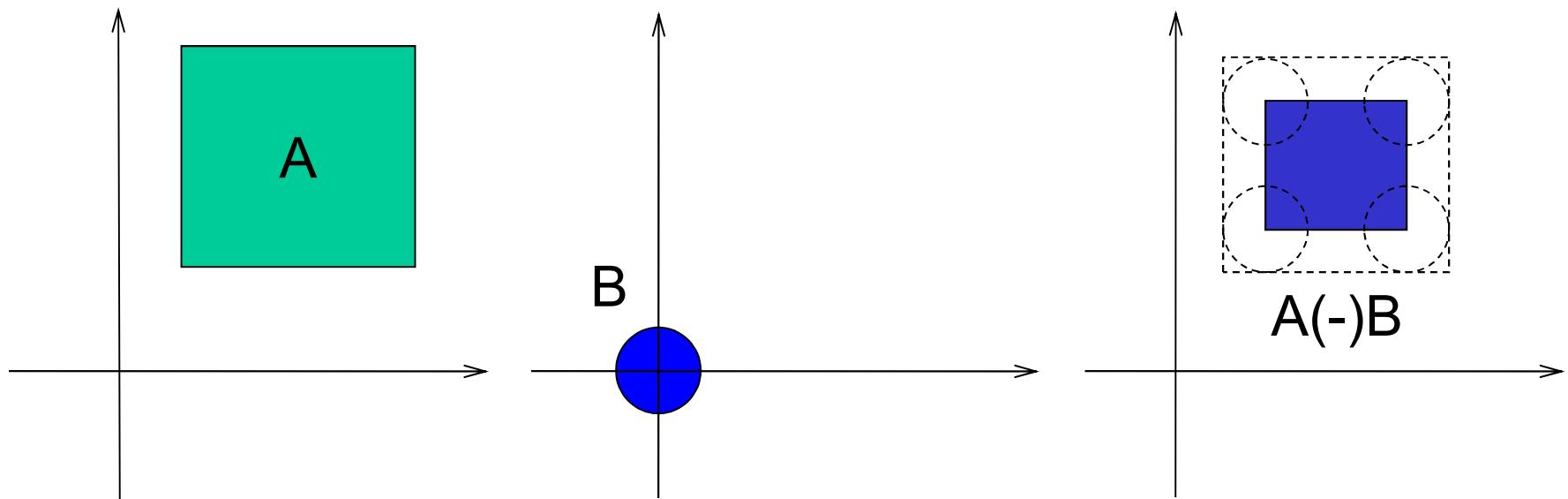


# Сужение

---

Сужение (erosion)

$$A(-)B = (A^C (+) B)^C, \text{ где } A^C - \text{ дополнение } A$$



# Результат операции сужения

---



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



$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & [1] & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



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

# Свойства

---

Коммутативный закон

- $A (+) B = B (+) A$
- $A (-) B < > B (-) A$

Ассоциативный закон

- $A (+) (B (+) C) = (A (+) B) (+) C$
- $A (-) (B (-) C) = (A (-) B) (-) C$

# Важное замечание

---

Результат морфологических операций во многом определяется применяемым структурным элементом. Выбирая различный структурный элемент можно решать разные задачи обработки изображений:

- Шумоподавление
- Выделение границ объекта
- Выделение скелета объекта
- Выделение сломанных зубьев на изображении шестерни

# Расширение

---

```
void Dilation(BIT* src[], bool* mask[], BIT* dst[])
{
    // W, H – размеры исходного и результирующего изображений
    // MW, MH – размеры структурного множества
    for(y = MH/2; y < H - MH/2; y++)
    {
        for(x = MW/2; x < W - MW/2; x++)
        {
            BIT max = 0;
            for(j = -MH/2; j <= MH/2; j++)
            {
                for(i = -MW/2; i <= MW/2; i++)
                    if((mask[i][j]) && (src[x + i][y + j] > max))
                    {
                        max = src[x + i][y + j];
                    }
            }
            dst[x][y] = max;
        }
    }
}
```

# Сужение

---

```
void Erosion(BIT* src[], bool* mask[], BIT* dst[])
{
    // W, H – размеры исходного и результирующего изображений
    // MW, MH – размеры структурного множества
    for(y = MH/2; y < H - MH/2; y++)
    {
        for(x = MW/2; x < W - MW/2; x++)
        {
            BIT min = MAXBIT;
            for(j = -MH/2; j <= MH/2; j++)
            {
                for(i = -MW/2; i <= MW/2; i++)
                    if((mask[i][j]) && (src[x + i][y + j] < min))
                    {
                        min = src[x + i][y + j];
                    }
            }
            dst[x][y] = min;
        }
    }
}
```

# Операции раскрытия и закрытия

---

Морфологическое раскрытие (opening)

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

Морфологическое закрытие (closing)

- **close(A, B) = (A (+) B) (-) B**

# Применение открытия

---

Применим операцию открытия к изображению с сильным шумом:



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

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

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

# Сужение vs Открытие

---



Сужение

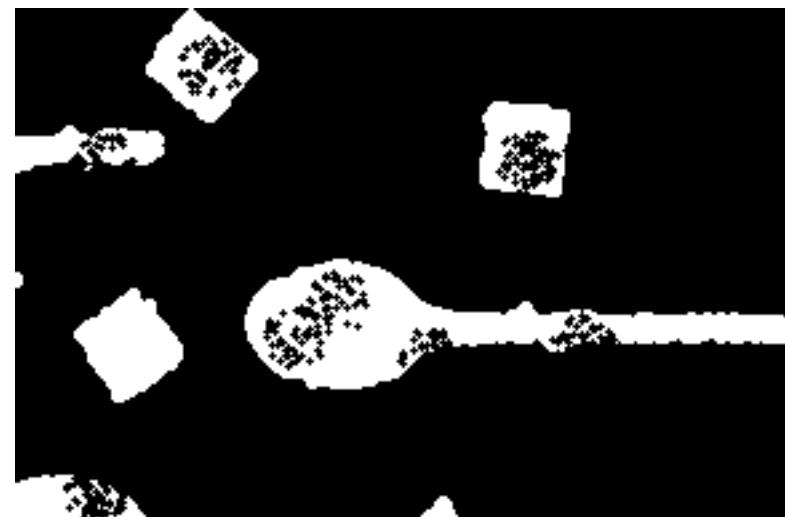


Открытие

# Дефекты бинаризации

---

Пример бинарного изображению с дефектами  
распознаваемых объектов



# Применение закрытия

---

Применим операцию закрытия к изображению с дефекиами объектов:



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



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



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

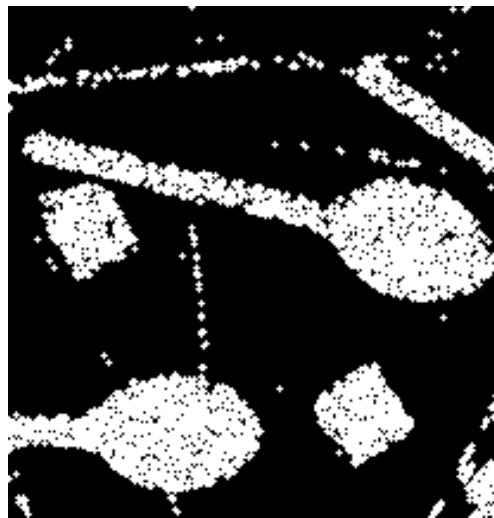
# Не лучший пример для морфологии

---

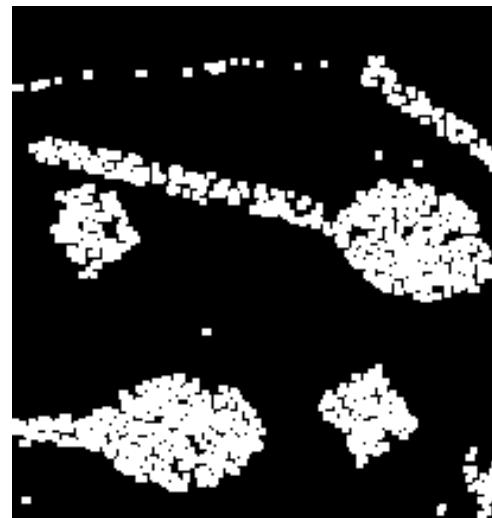
Не во всех случаях математическая  
морфология так легко убирает дефекты,  
как хотелось бы...



# Применения операции открытия



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



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



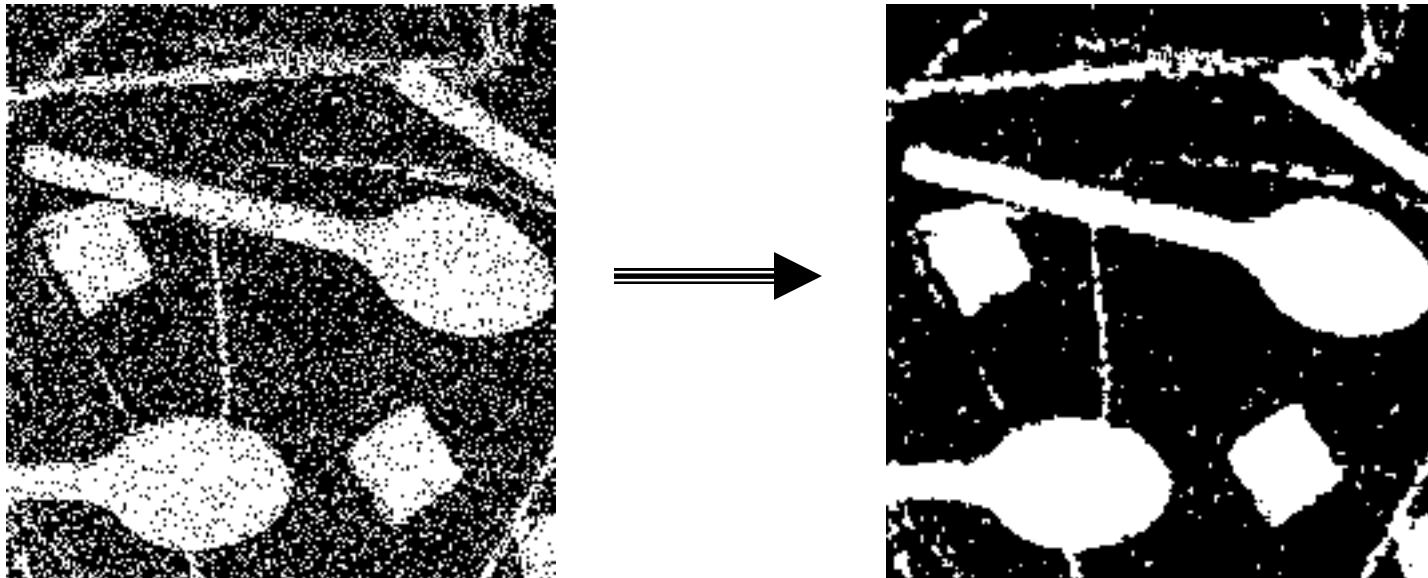
$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Часто помогает медианная фильтрация!

# Медианный фильтр

---

Фильтр с окрестностью 3x3



# Выделение связных областей

- Определение связной области:
  - Множество пикселей, у каждого пикселя которого есть хотя бы один сосед, принадлежащий данному множеству.



Соседи пикселей:

|   |   |   |
|---|---|---|
|   | 1 |   |
| 2 | * | 3 |
|   | 4 |   |

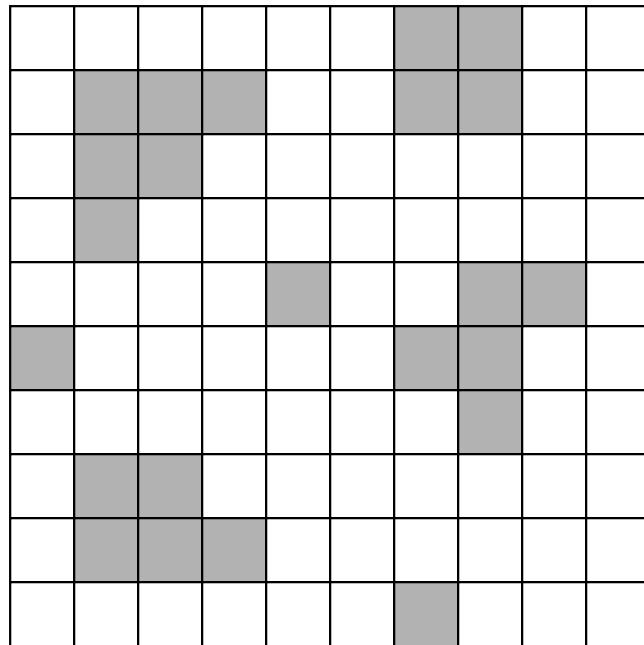
4-связность

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | * | 5 |
| 6 | 7 | 8 |

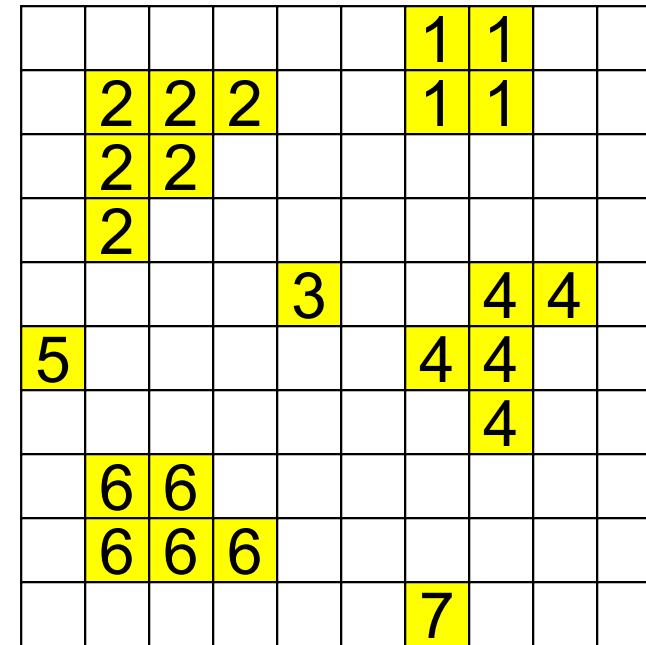
8-связность

# Разметка связных областей

---



Бинарное изображение



Размеченное изображение

# Рекурсивный алгоритм

---

```
void Labeling(BIT* img[], int* labels[])
{
    // labels должна быть обнулена
    L = 1;
    for(y = 0; y < H; y++)
        for(x = 0; x < W; x++)
    {
        Fill(img, labels, x, y, L++);
    }
}
```

# Рекурсивный алгоритм

---

```
void Fill(BIT* img[], int* labels[], int x, int y, int L)
{
    if( (labels[x][y] == 0) && (img[x][y] == 1) )
    {
        labels[x][y] = L;
        if( x > 0 )
            Fill(img, labels, x - 1, y, L);
        if( x < W - 1 )
            Fill(img, labels, x + 1, y, L);
        if( y > 0 )
            Fill(img, labels, x, y - 1, L);
        if( y < H - 1 )
            Fill(img, labels, x, y + 1, L);
    }
}
```

# Последовательное сканирование

---

Последовательно, сканируем бинарное изображение сверху вниз, слева направо:

|   |   |  |
|---|---|--|
|   | C |  |
| B | A |  |
|   |   |  |

```
if A = 0
    do nothing

else if (not B labeled) and (not C labeled)
    increment label numbering and label A

else if B xor C labeled
    copy label to A

else if B and C labeled
    if B label = C label
        copy label to A
    else
        copy either B label or C label to A
        record equivalence of labels
```

# Последовательное сканирование

---

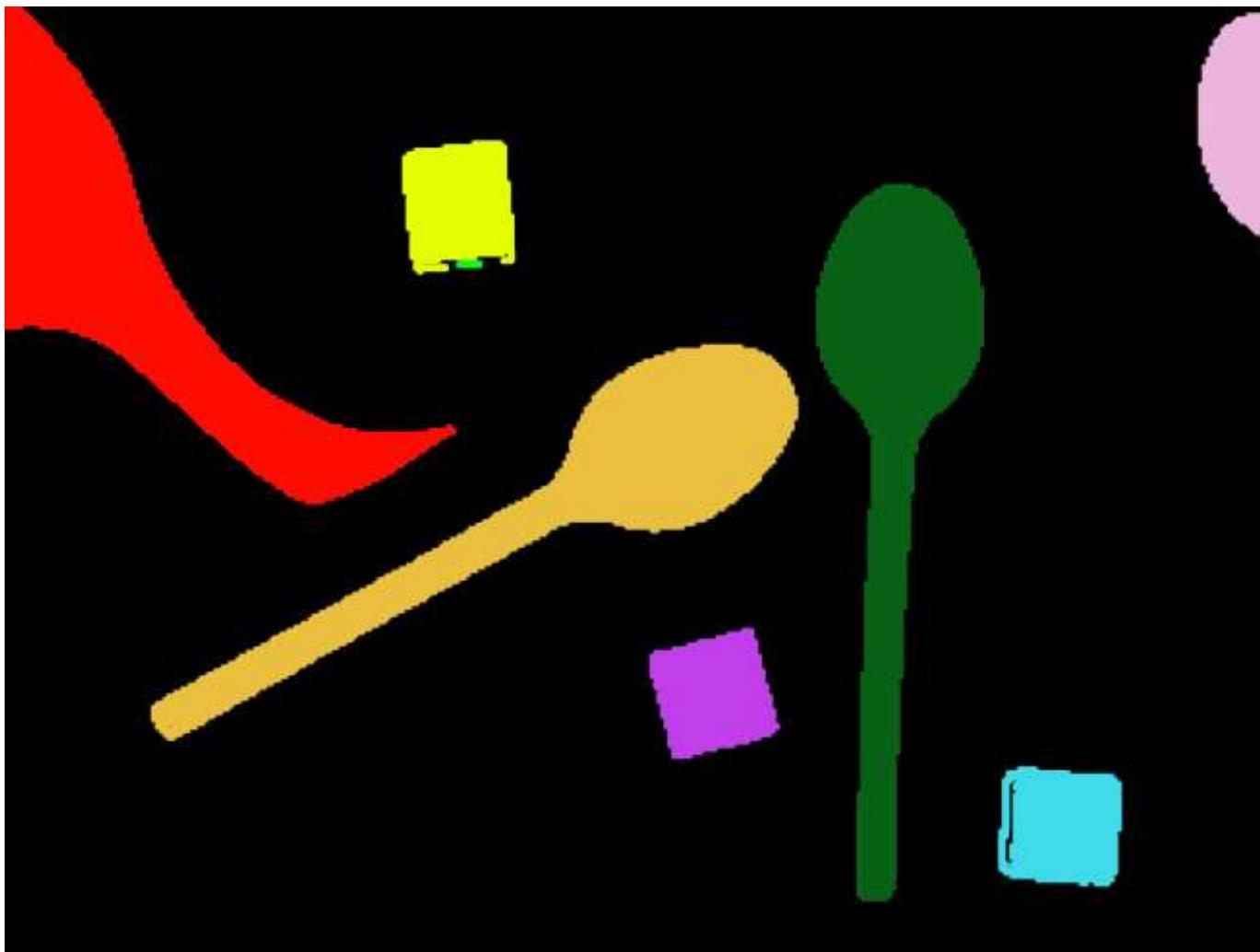
Случай конфликта:

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |
| 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | ? |   |   |   |

Постобработка - переразметка с учетом эквивалентностей областей  
(второй проход в алгоритме)

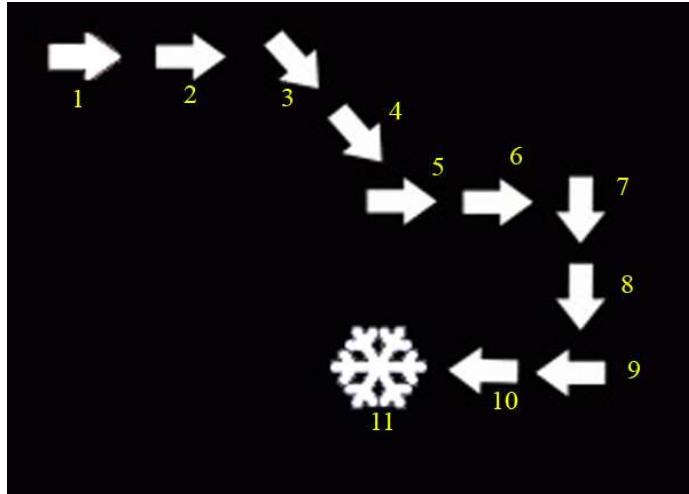
# Выделенные связанные компоненты

---



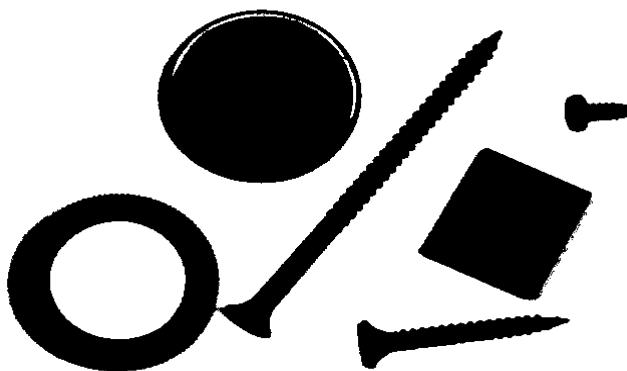
# Анализ выделенных областей

---



Для дальнейшего анализа требуется вычислить некоторые числовые характеристики (признаки) областей:

- геометрические признаки
- фотометрические признаки



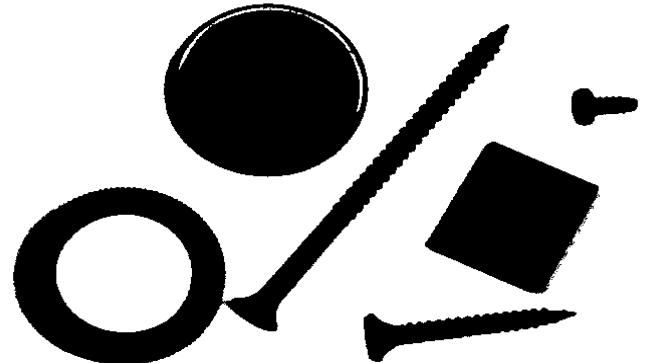
На основе этих характеристик можно классифицировать получаемые области

# Геометрические признаки

---

Для каждой области можно подсчитать некий набор простейших числовых характеристик:

- Площадь
- Центр масс
- Периметр
- Компактность
- Ориентацию главной оси инерц
- Удлиненность (эксцентриситет)

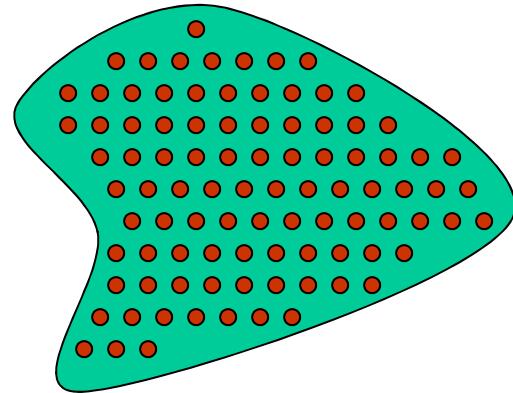


# Площадь и центр масс

---

- Площадь – количество пикселей в области;

$$A = \sum_{x=0}^m \sum_{y=0}^n I(x, y)$$



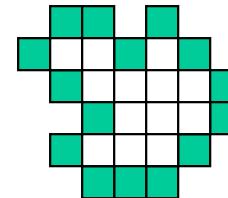
- Центр масс

$$\bar{x} = \frac{\sum_{x=0}^m \sum_{y=0}^n x I(x, y)}{A}; \bar{y} = \frac{\sum_{x=0}^m \sum_{y=0}^n y I(x, y)}{A}$$

# Периметр и компактность

---

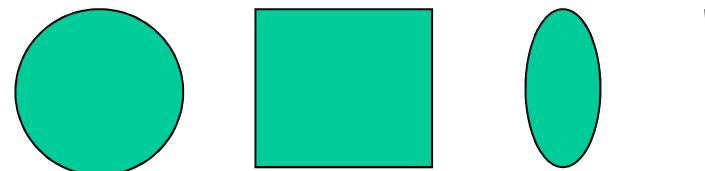
- Периметр – количество пикселей принадлежащих границе области;



- Компактность – отношение квадрата периметра к площади;

$$C = \frac{P^2}{A}$$

Наиболее компактная фигура –  $C = 4\pi$   
круг:



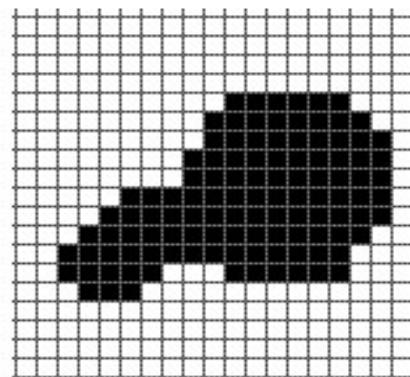
# Подсчет периметра области

1. Пиксель лежит на границе области, если он сам принадлежит области и хотя бы один из его соседей области не принадлежит.  
*(внутренняя граница)*
  
2. Пиксель лежит на границе области, если он сам не принадлежит области и хотя бы один из его соседей области принадлежит.  
*(внешняя граница)*

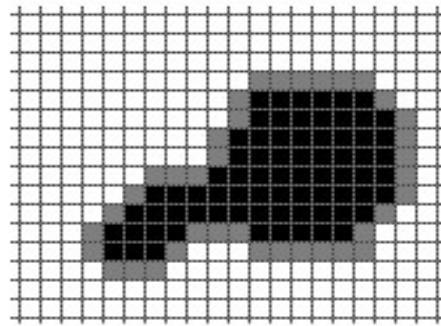
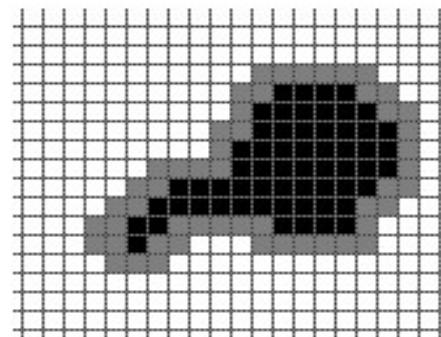
Периметр зависит также от того 4-х или 8-ми связность используется для определения соседей.

# Пример периметров области

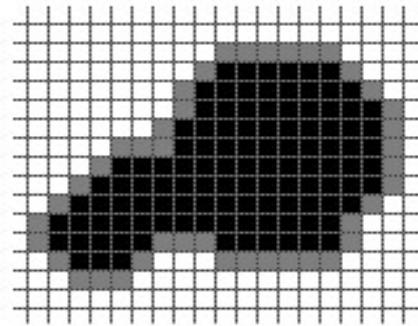
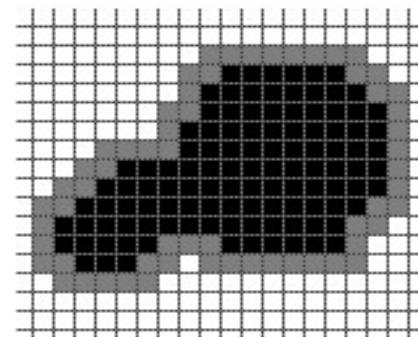
---



Область



Внутренняя граница



Внешняя граница

# Операция оконтуривания объекта

---

При работе с бинарными изображениями контуры объекта можно получить с помощью операций математической морфологии

Внутреннее оконтуривание

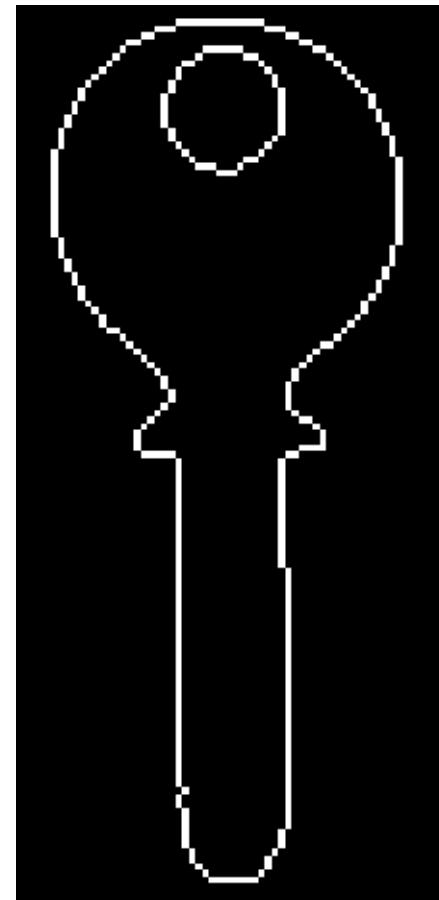
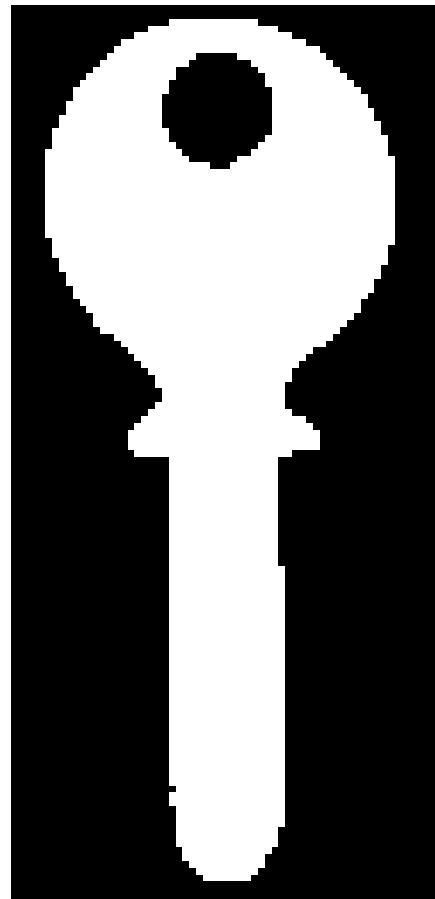
- $C_I = A - (A \cap B)$

Внешнее оконтуривание

- $C_O = (A \cup B) - A$

# Пример оконтуривания объекта

---



# Статистические моменты области

---

Дискретный центральный момент  $m_{ij}$  области определяется следующим образом:

$$m_{ij} = \sum_{x,y \in S}^n (x - \bar{x})^i (y - \bar{y})^j I(x, y)$$

↓  
Центр масс области

# Инвариантные характеристики

---

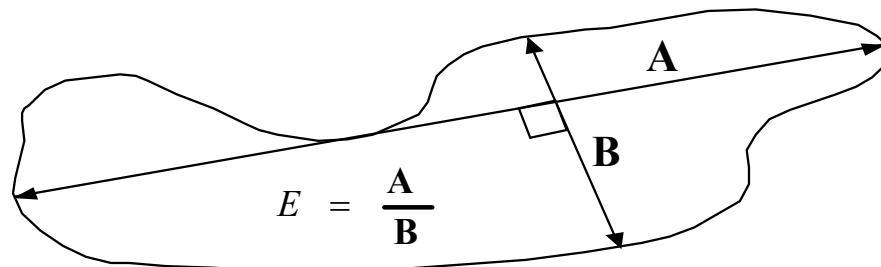
Для распознавания нас интересуют характеристики инвариантные по отношению к масштабированию, переносу, повороту:

- Удлиненность, нецентрированность (эксцентриситет)

$$\text{elongation} = \frac{m_{20} + m_{02} + \sqrt{(m_{20} - m_{02})^2 + 4m_{11}^2}}{m_{20} + m_{02} - \sqrt{(m_{20} - m_{02})^2 + 4m_{11}^2}}$$

- Компактность

$$C = \frac{P^2}{A}$$

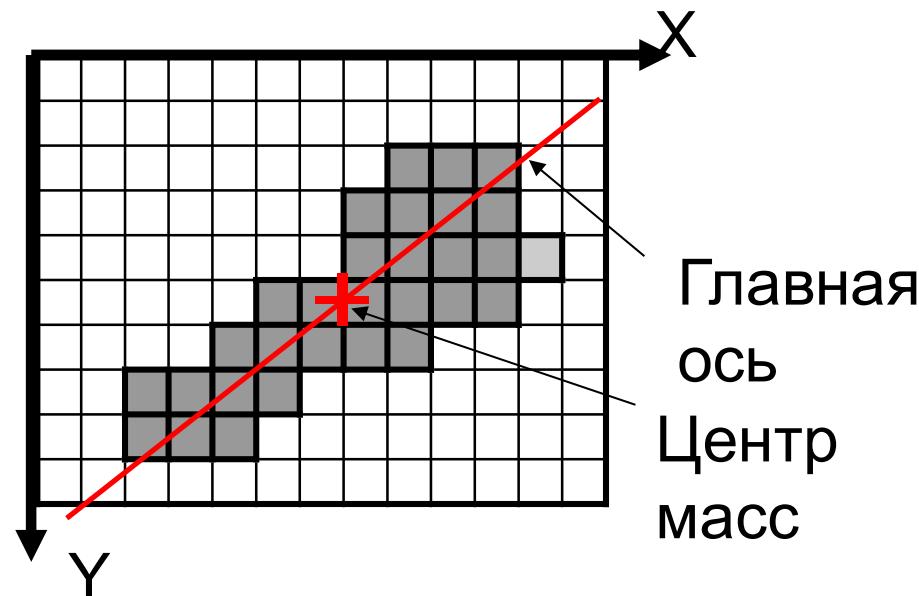


# Ориентация главной оси инерции

---

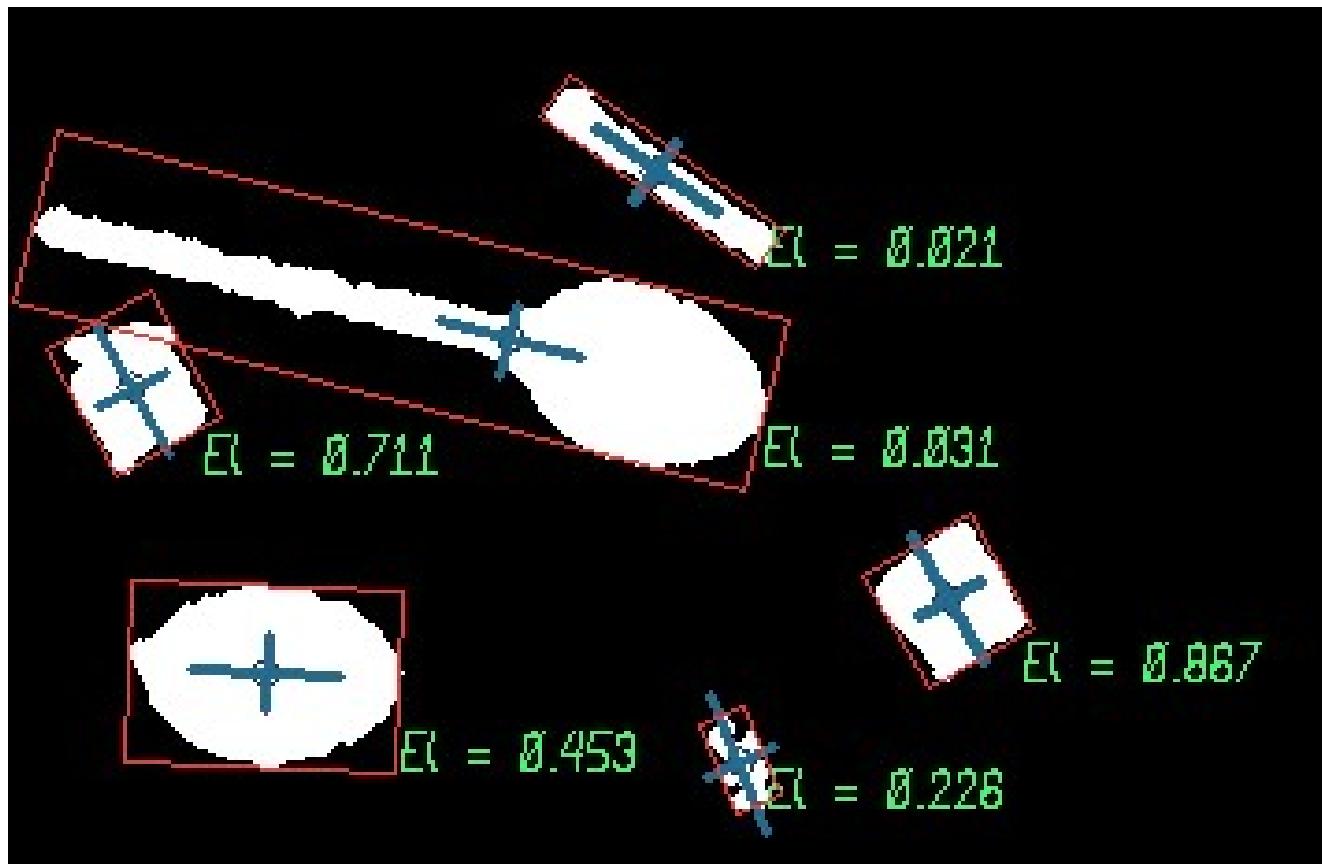
Не является инвариантной к повороту, но в ряде случаев предоставляет полезную информацию об ориентации объекта:

$$\theta = \frac{1}{2} \arctan \left( \frac{2m_{11}}{m_{20} - m_{02}} \right)$$



# Пример

---



Вычисленные значения признаков

## Другие признаки

---

Другие инвариантные характеристики области:

$$M_1 = m_{20} + m_{02}$$

$$M_2 = (m_{20} - m_{02})^2 + 4m_{11}^2$$

$$M_3 = (m_{30} - 3m_{12})^2 + (3m_{21} - m_{03})^2$$

$$M_4 = (m_{30} + m_{12})^2 + (m_{21} + m_{03})^2$$

$$M_5 = (m_{30} - 3m_{12})(m_{30} + m_{12}) [(m_{30} + m_{12})^2 - 3(m_{21} + m_{03})^2] \\ + (3m_{21} - m_{03})(m_{21} + m_{03}) [3(m_{30} + m_{12})^2 - (m_{21} + m_{03})^2]$$

$$M_6 = (m_{20} + m_{02}) [(m_{30} + m_{12})^2 - 3(m_{21} + m_{03})^2] \\ + 4m_{11}(m_{30} + m_{12})(m_{03} + m_{21})$$

$$M_7 = (3m_{21} - m_{03})(m_{12} + m_{30}) [(m_{30} + m_{12})^2 - 3(m_{21} + m_{03})^2] \\ - (m_{30} - 3m_{12})(m_{12} + m_{03}) [3(m_{30} + m_{12})^2 - (m_{21} + m_{03})^2]$$

(54)

# Фотометрические признаки

---

Для каждой области можно подсчитать некий набор простейших числовых характеристик:

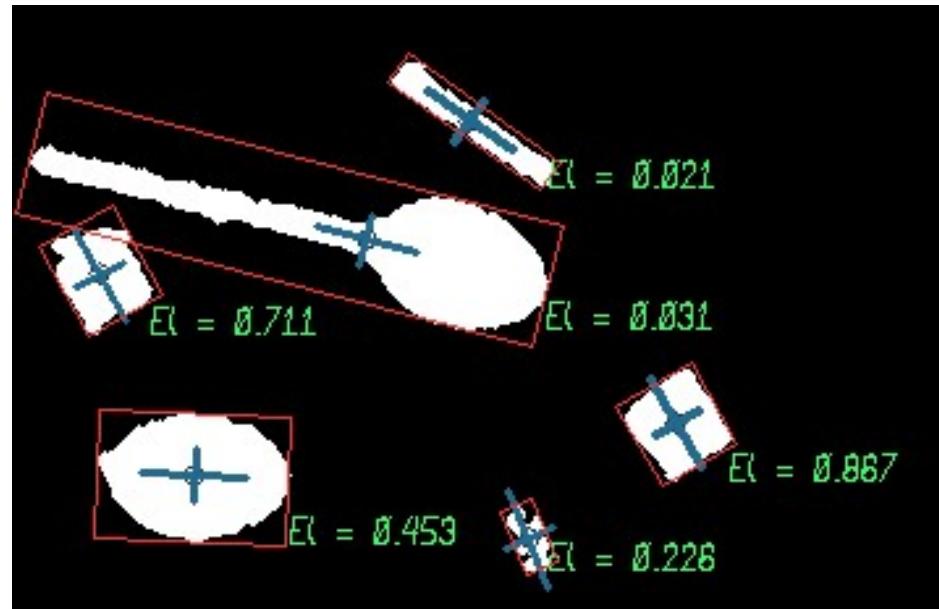
- Средняя яркость
- Средний цвет (если изображение цветное)
- Гистограмма распределения яркостей  
(или три гистограммы распределения R, G, B)
- Дисперсию (разброс) яркостей или цвета

Разумеется, все это считается по исходному, а не бинарному изображению!

# Как анализировать признаки

---

- Пример – ложки и сахар



# Как анализировать признаки

---

- Как воспользоваться признаками для классификации?
  - Подобрать диапазоны значений для разных классов вручную, экспериментально  
(может быть весьма трудоемко)
  - Подобрать диапазоны значений графически  
(нужна база для тренировки, трудно, если признаков много)
  - Обучить классификатор с помощью машинного обучения
    - На будущих лекциях!
    - Второе задание!

# Ручной подбор

---

- Из общих соображений:
  - Ложки более вытянутые, чем сахарные кусочки
  - Ложки больше чем сахарные кусочки
  - Сахарные кусочки квадратные
  - Области появляющиеся из-за шума обычно небольшие и неквадратные
- Пытаемся сконструировать решающее правило, проверяем экспериментально
- Может быть весьма утомительно

# Графический анализ

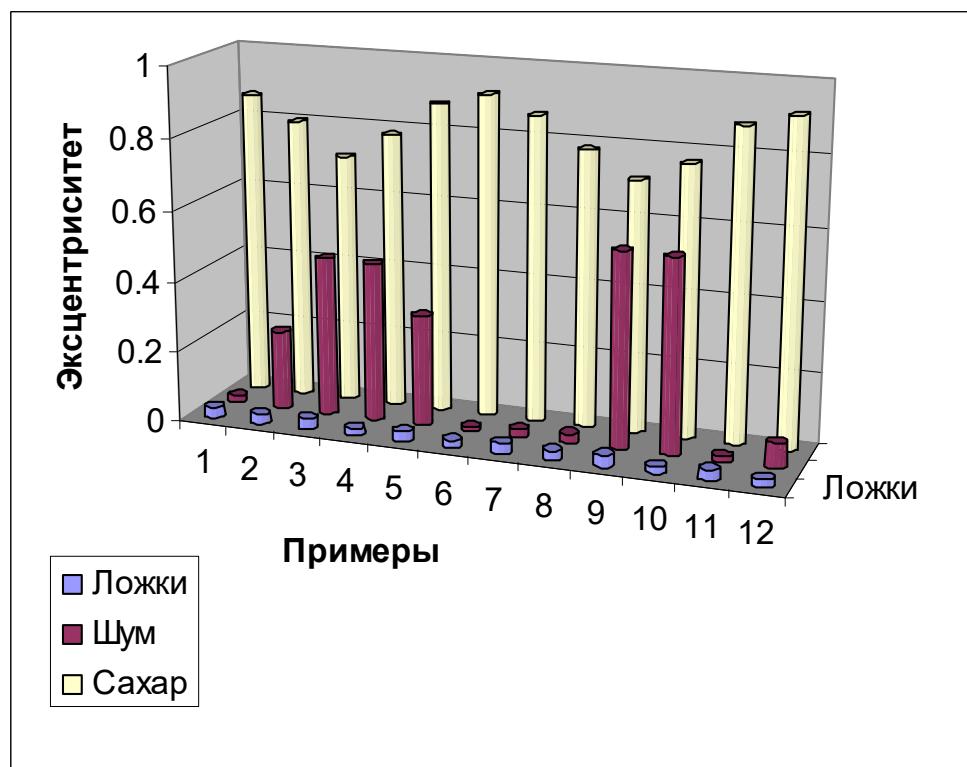
---

- Собрать тренировочную базу изображений
  - Где только ложки
  - Где только сахар
  - Где только шум
- Как получить такие? Да просто закрасить все остальное.
- Брать признаки и строить графики

# Графический анализ

---

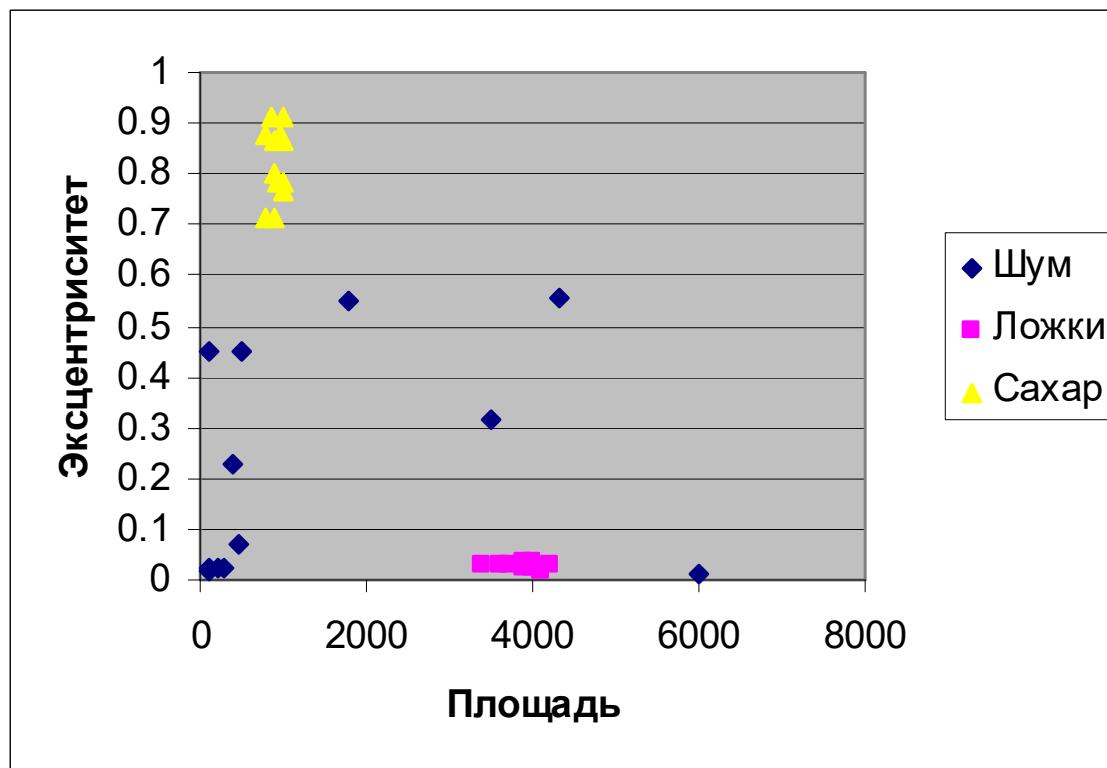
- Диаграмма распределения эксцентрикитета  
(проблема – не получается отличить шум от ложек)



# Графический анализ

---

- График распределения эксцентрикитета и площади (гораздо лучше – можем подобрать значения порогов)



# Машинное обучение

---

- Причина бурного развития компьютерного зрения в последние годы.
- Требуются большие коллекции примеров для обучения.
- Рассмотрим позднее!

# На следующей лекции

---

- Методы представления изображений
- Избыточность данных
- «Компактность» vs «Разреженность»
- Обработка изображений на основе обучаемых словарей