

Выделение краёв



Выделение краёв

Задача: выделить границы объектов на изображении.

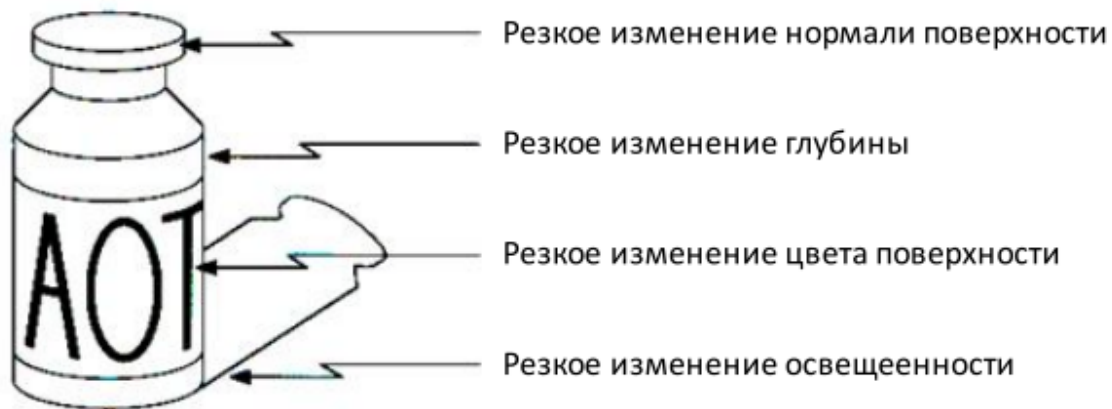
Решаемая задача: Выделить резкие изменения (разрывы) изображения



Выделение краёв



Причины возникновения краёв

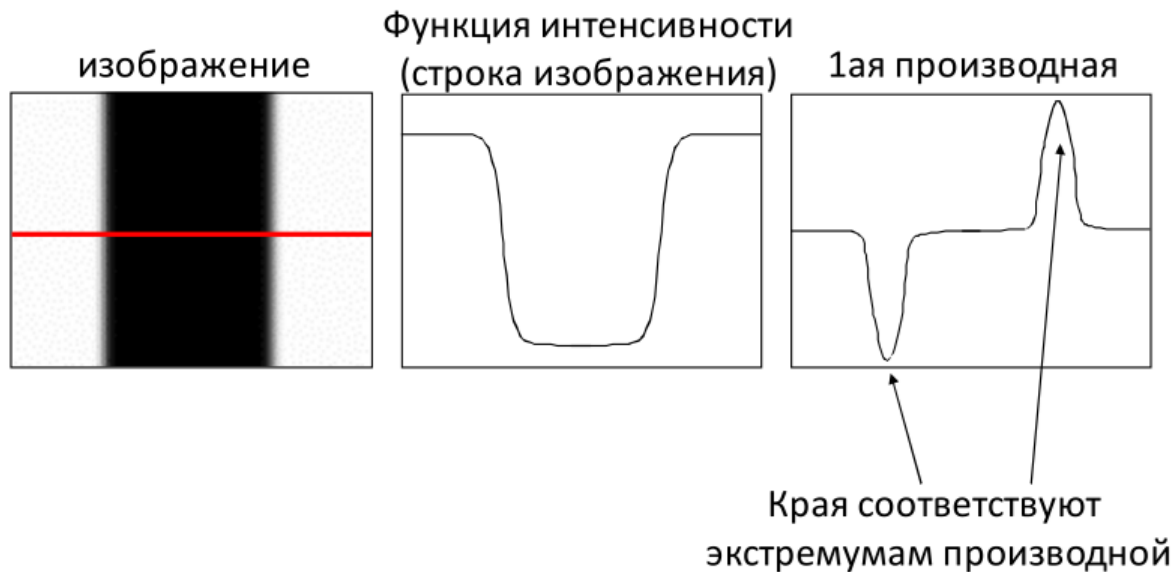


Резкое изменение = «разрыв»

Source: Steve Seitz

Описание «края»

Край – это точка резкого изменения значений функции интенсивности изображения

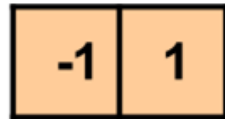


Дифференцирование и свёртка

Производная по оси X

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \rightarrow 0} \left(\frac{f(x + \varepsilon, y)}{\varepsilon} - \frac{f(x, y)}{\varepsilon} \right) \quad \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

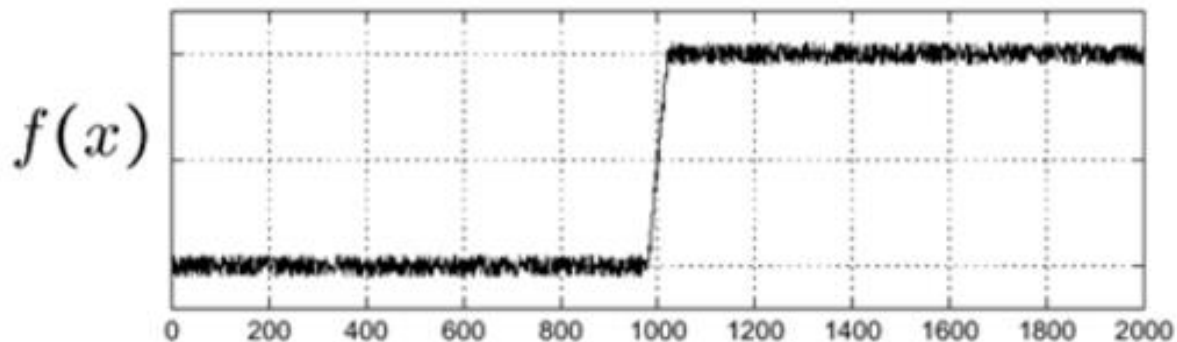
- Разностная производная - линейная и инвариантная к переносу
- Можно записать как свёртку



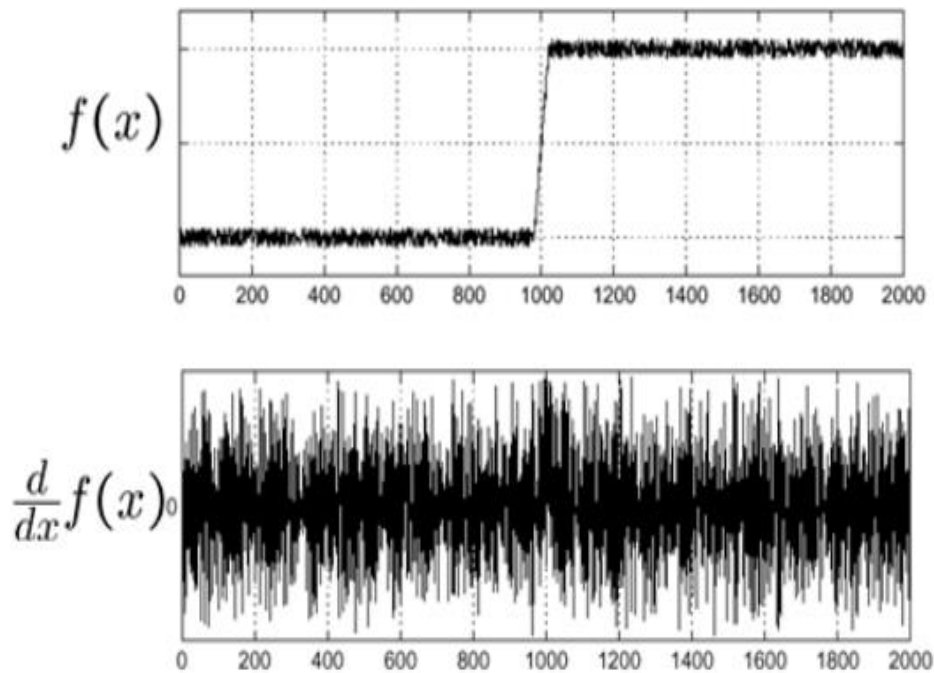
Простейший фильтр

Влияние шума

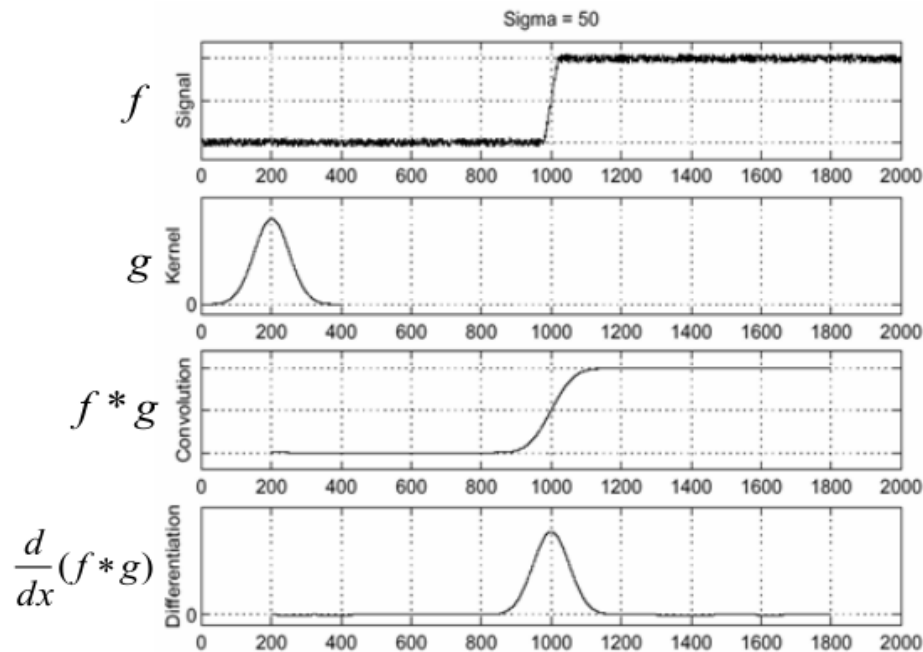
Строка изображения, представленная
в виде сигнала



Влияние шума



Свёртка

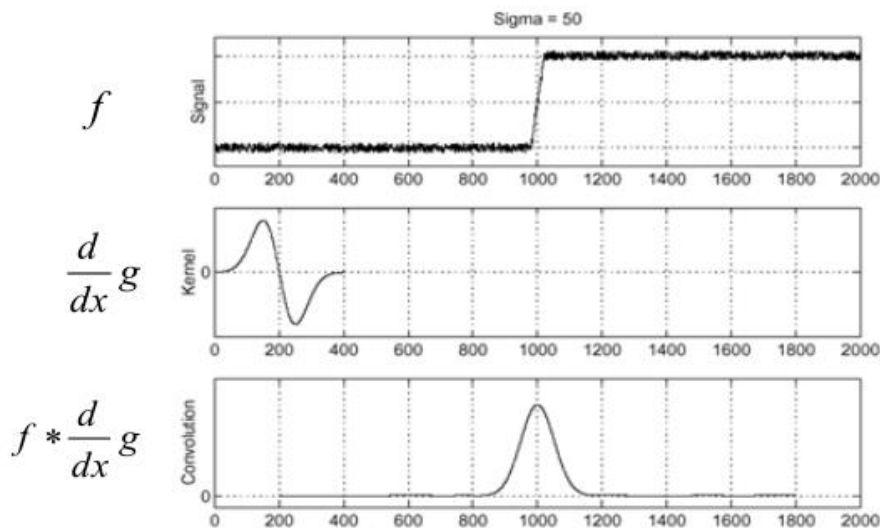


Для поиска краев ищем пики в: $\frac{d}{dx}(f * g)$

Свёртка

- Операции свертки и дифференцирования ассоциативны:
- Это экономит 1 операцию:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$



Фильтры градиентов

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

Робертса



Превитт

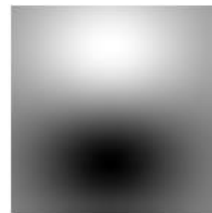
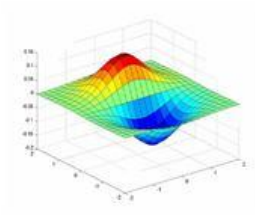
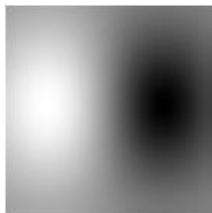
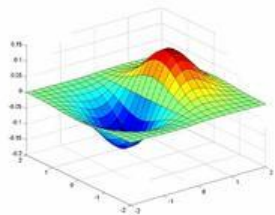
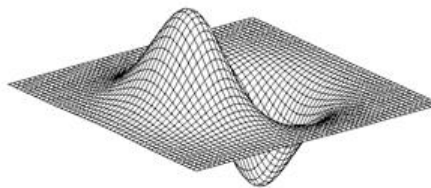
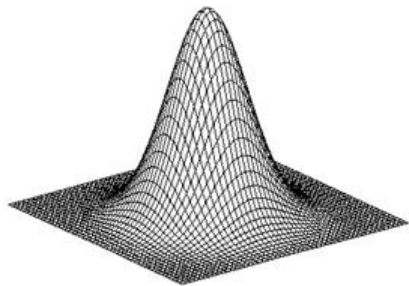


Собеля



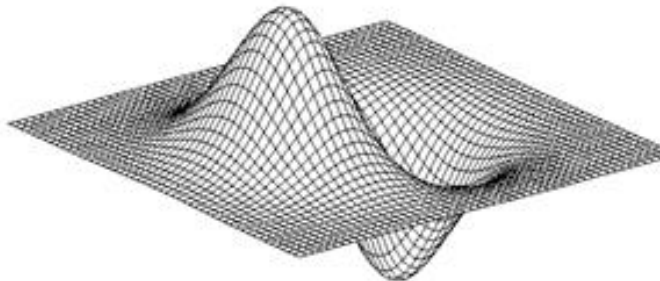
Математический смысл – приближённое вычисление производных по направлению.

Производная фильтра Гаусса



Лапласиана

```
laplacian = cv2.Laplacian(image, cv2.CV_64F)  
cv2.imshow('Laplacian', laplacian)  
cv2.waitKey(0)
```



Детектор Canny

1. Свертка изображения с ядром – производной от фильтра гаусса
2. Поиск силы и направления градиента
3. Выделение локальных максимумов (Non-maximum suppression)
 - Утоньшение 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

Отсечение по порогу



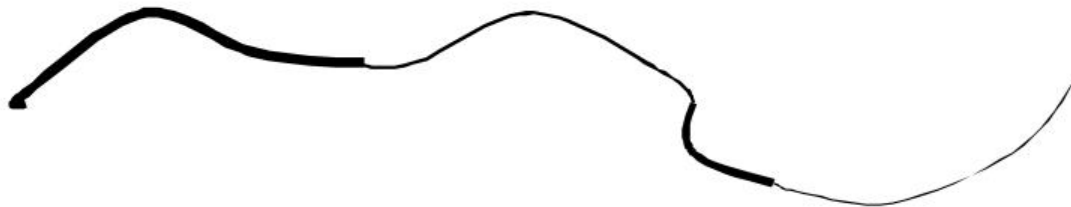
Норма градиента



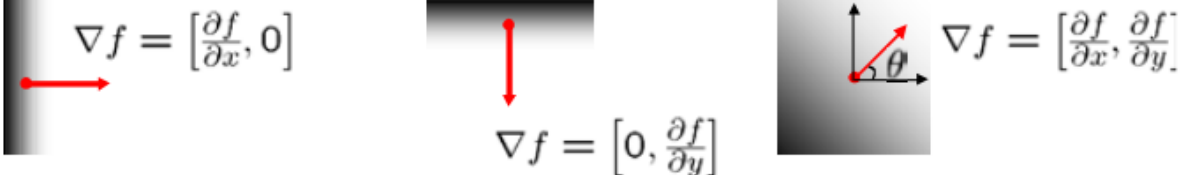
Отсечение по порогу

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)



Градиент изображения

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

Градиент направлен в сторону наибольшего изменения интенсивности

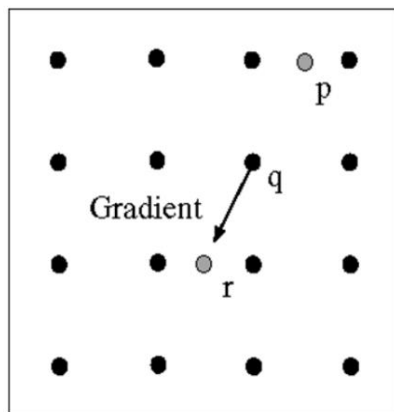
Направления градиента задается как: $\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$

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

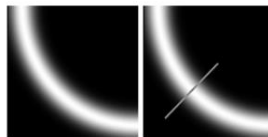
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Гистерезис

Поиск локальных максимумов

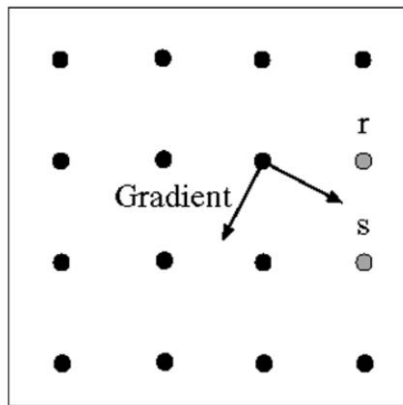


Максимум достигается в q , если значение больше p и r . Значения в p и r интерполируем.

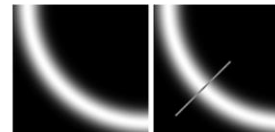


Source: D. Forsyth

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).



Source: D. Forsyth

Пример работы



Исходное изображение



Высокий порог
(сильные края)

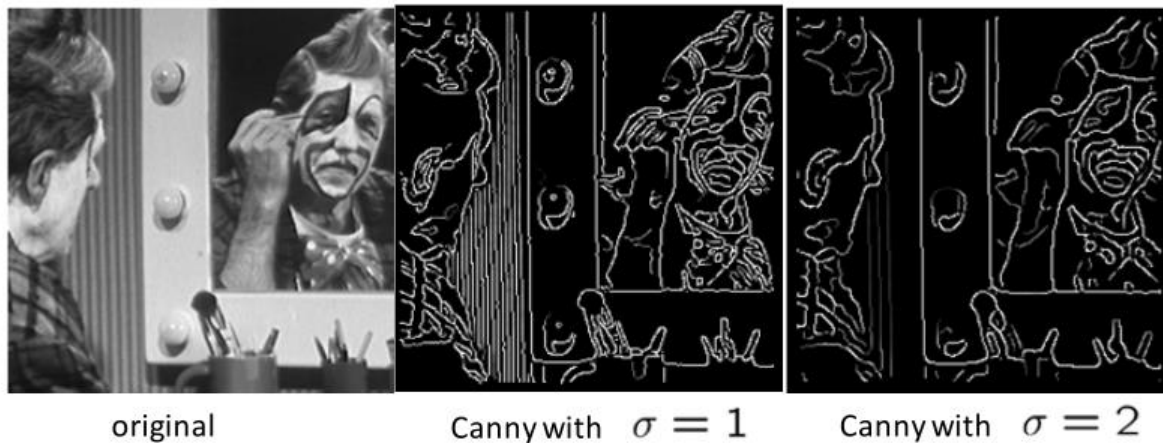


Низкий порог
(слабые края)



Порог по гистерезису

Изменение параметра



Выбор σ зависит от задачи

- большое σ - поиск крупных границ
- маленькое σ - выделение мелких деталей

Детектор Кенни

```
img_canny = cv2.Canny(img,100,200)
```

OpenCV puts all the above in single function, [cv.Canny\(\)](#). We will see how to use it. First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Third argument is aperture_size. It is the size of Sobel kernel used for find image gradients. By default it is 3

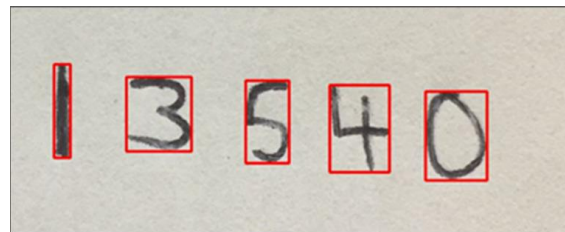
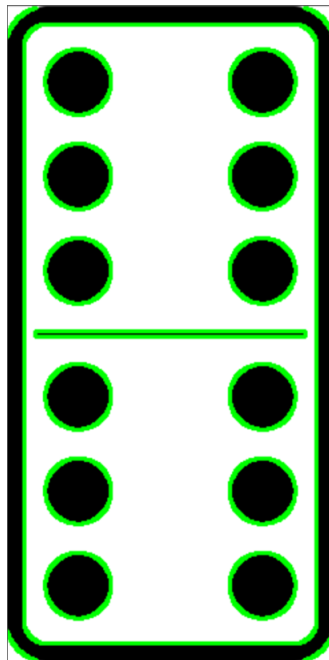
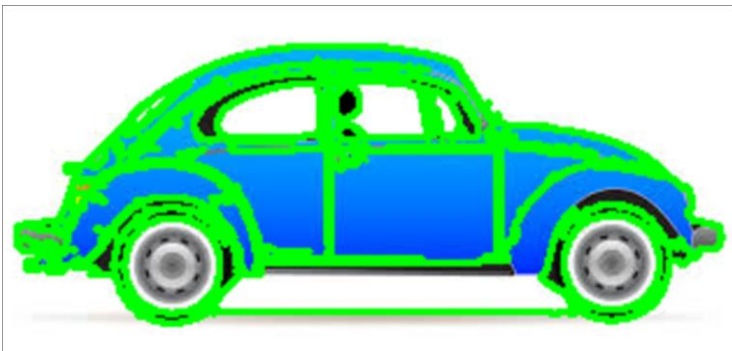
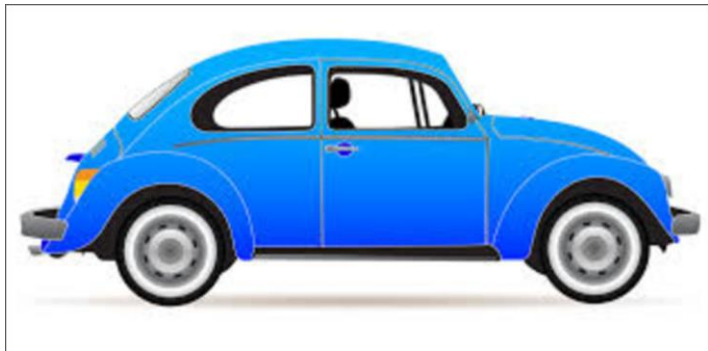
Сегментация



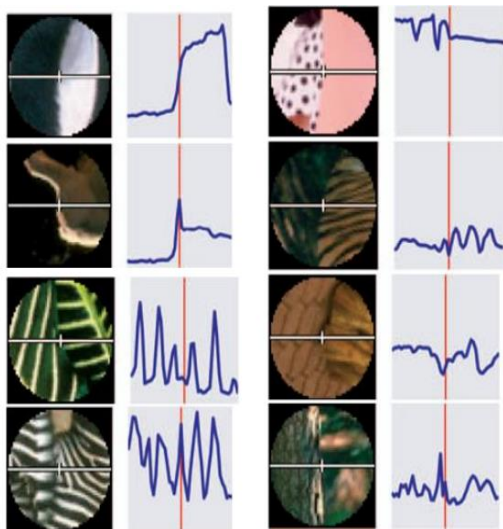
Сегментация

- Понятие контура
- Сортировка контуров (размер, положение слева направо)
- Аппроксимация контуров, вычисление оболочки (convex hull)
- Определение формы
- Детектор линии
- Детектор окружности
- Детектор блоков

Сегментация



Сложные текстуры

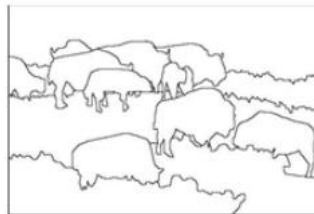


Source: Martin et al. 2003

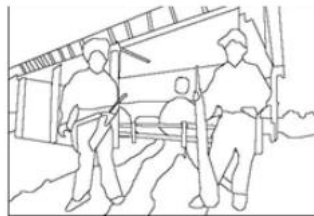
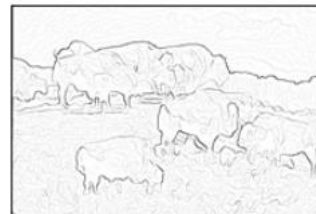
изображение



разметка вручную



норма градиента



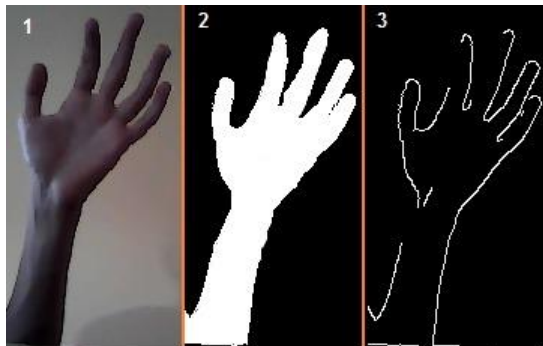
- Berkeley segmentation database:

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

Контур

Контур - кривая, содержащая точки вдоль границы.

- для лучшей точности использовать бинарные изображения (после threshold или Canny)
- в OpenCV поиск контуров происходит как поиск белых объектов на чёрном фоне



findContours

```
cv2.findContours(image, -входное изображение  
    int      retrieval mode, #метод иерархии  
    int      approximation method, #метод упрощения  
)
```

На выходе:

contours - выходной массив точек (x,y) контуров

Hierarchy – описание отношений между контурами (child-parent)

mode - RETR_EXTERNAL

retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.

RETR_LIST

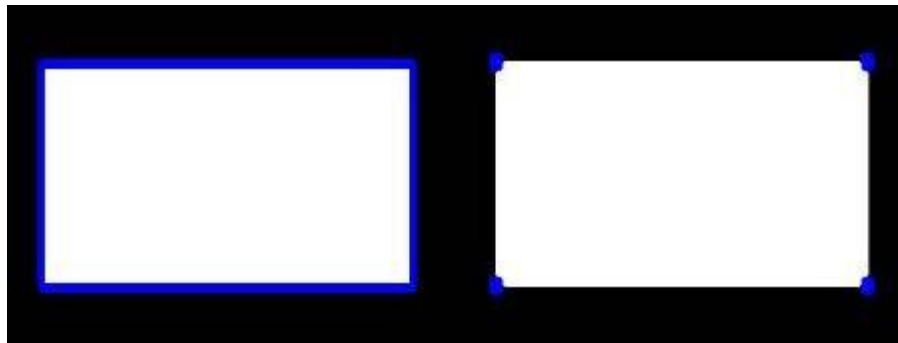
retrieves all of the contours without establishing any hierarchical relationships.

RETR_CCOMP

retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.

RETR_TREE

cv2.findContours



Contour Approximation Method

cv2.CHAIN_APPROX_NONE – возвращает все точки линии контура

cv2.CHAIN_APPROX_SIMPLE – возвращает начальную и конечную координаты прямой

Hierarchy Types

cv2.RETR_LIST – возвращает все контуры

cv2.RETR_EXTERNAL – возвращает только внешние контуры

cv2.RETR_COMP – возвращает всё в диапазоне двух уровней иерархии

cv2.RETR_TREE – возвращает всё в иерархическом порядке

Иерархия хранится в формате: [Next, Previous, First Child, Parent]

Подробнее: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html