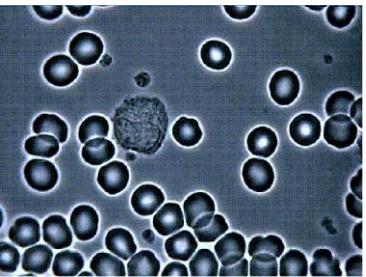


# Сегментация изображений

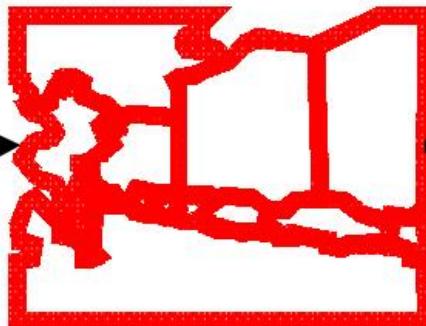
## Лекция 6



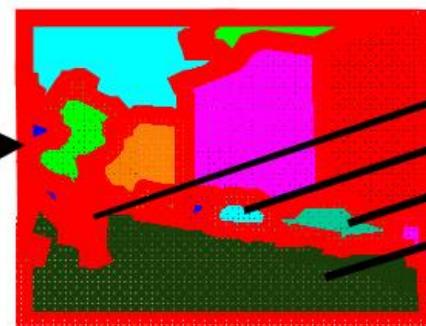
Изображение



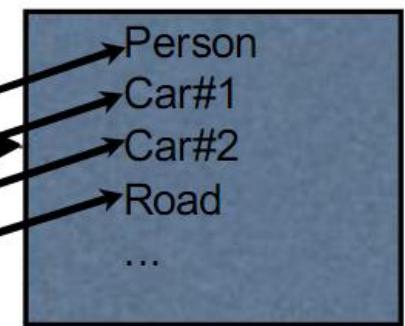
Границы



Сегментация



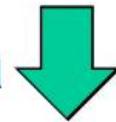
Распознавание



Теория



Практика



Изображение



Края



Сегментация



Распознавание



# Сколько сегментов и какие?



# Или такие? Грубая сегментация

Небо

Строения

Автомобиль

Автомобиль

# Определение сегментации

- «Жесткая» сегментация
  - Разбиение изображения на неперекрывающиеся области, однородные по некоторому признаку и покрывающие всё изображение
- Формально:
  - Разбиение изображения на набор областей

$$S = \{S_i\}, i = \overline{1, N}$$

$$I = \bigcup_{i=1..N} S_i$$

$$\forall i, j = \overline{1, N} : i \neq j \quad S_i \cap S_j = \emptyset$$

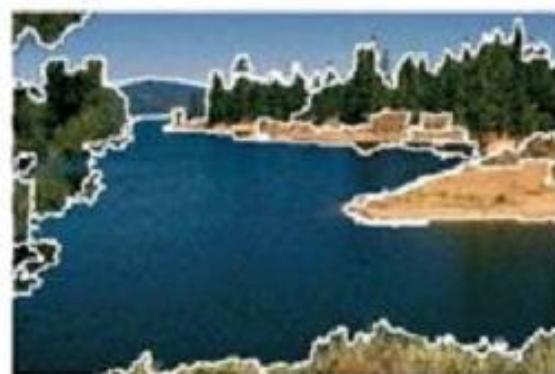
$$\forall i = \overline{1, N}, P(S_i) = \text{истина}$$

$$\forall i, j = \overline{1, N} : i \neq j \quad P(S_i \cup S_j) = \text{ложь}$$

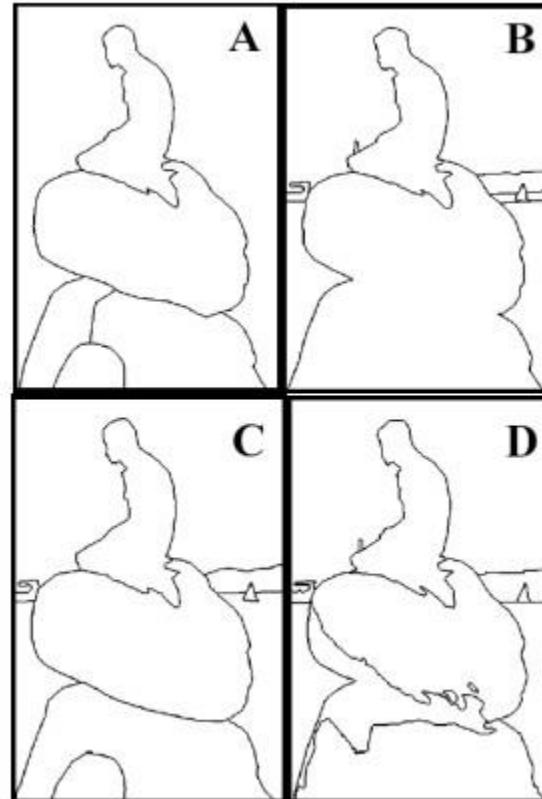
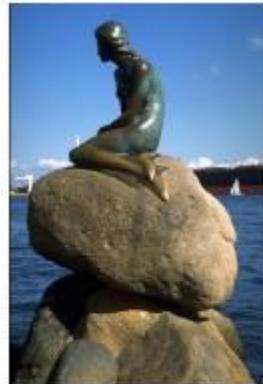
# Критерии «похожести» пикселей

- По расстоянию
- По яркости
- По цвету
- По текстуре

# Сегментация по цвету



# Варианты сегментации изображения. Какой верный?



Berkeley Segmentation Dataset

# Оценка качества работы методов сегментации

- Целостность и однородность по некоторому признаку
- Отличие признака для смежных областей
- Отсутствие мелких отверстий внутри
- Гладкие границы

# Критерий однородности

- Гистограмма содержит не больше 1 значительного пика



- Отклонение любого пикселя от средней яркости  $< T_{avg}$

$$\forall p \in S \quad \left| I(p) - \frac{1}{N} \sum_{q \in S} I(q) \right| < T_{avg}$$

- Разница между соседними пикселями  $< T_{diff}$

$$\forall p \in S, \forall q \in N(p) \quad |I(p) - I(q)| < T_{diff}$$

- «Слабая» граница между регионами (только для слияния)

# Задачи автоматической сегментации делятся на два класса

- Выделение областей изображения с известными свойствами
- Разбиение изображения на однородные области

# Семейства классических методов

- **Основанные на формировании однородных областей**
  - без пространственных связей
  - с учетом пространственных связей
- Суперпиксели
- Основанные на поиске краев (края и области)
  - Canny
  - Pb-детектор краёв
- Методы на графах
  - Normalized cut
  - «Эффективный метод» Felzenszwalb & Huttenlocher
- Энергетические методы
  - Snakes
  - Методы уровня

# Сегментация через поиск однородных областей

- **Сегментация без учёта пространственных связей**
  - Пороговая фильтрация
  - Кластеризация по цвету
  - К-средних
  - сдвиг среднего (Mean shift) и развитие
- Сегментация с учётом пространственных связей
  - Разрастание областей (region growing)
  - Слияние/разделение областей (region merging/splitting)
  - Методы водораздела

# Этапы кластеризации цветового пространства

- Чтобы свести задачу сегментации к задаче кластеризации, достаточно
  - задать отображение точек изображения в некоторое пространство признаков
  - и ввести метрику (меру близости) на этом пространстве признаков
- Применение методов кластерного анализа
- Обычно после кластеризации точек - выделение связных компонент

# Кластеризация цветового пространства: к-средних

Наиболее популярный метод кластеризации, используемый для сегментации изображений к-средних



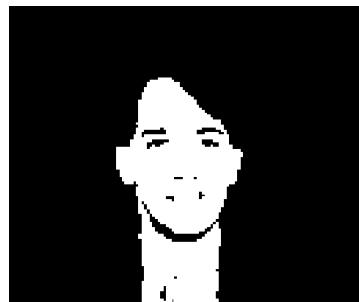
6 кластеров

Методы кластеризации плохо работают на зашумленных изображениях

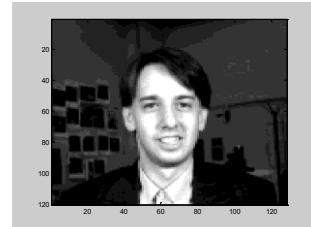
# Сравнение k-средних с порогом по средней яркости

Чем отличается сегментация с помощью k-средних на 2 кластера от простейшей пороговой бинаризации по средней яркости изображения?

Пример:



k-средних



Порог по средней яркости

В причинах предлагается разобраться самостоятельно

# Семейства методов

- **Основанные на формировании однородных областей**
  - без пространственных связей
  - **с учетом пространственных связей**
- Суперпиксели
- Основанные на поиске краев (края и области)
  - Canny
  - Pb-детектор краёв
- Методы на графах
  - Normalized cut
  - «Эффективный метод» Felzenszwalb & Huttenlocher
- Энергетические методы
  - Snakes
  - Методы уровня

# Сегментация через поиск однородных областей

- Сегментация без учёта пространственных связей
  - Пороговая фильтрация
  - Кластеризация по цвету
  - К-средних
  - сдвиг среднего (Mean shift) и развитие
- **Сегментация с учётом пространственных связей**
  - Разрастание областей (region growing)
  - Слияние/разделение областей (region merging/splitting)
  - Методы водораздела

# Разрастание регионов (Region growing)

- Простая идея – начиная с некоторого «семени» обходить пиксели и присоединять к области пока выполняется условие однородности

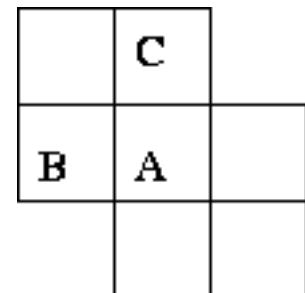
# Критерии присоединения к региону

- Близость точки к центру региона
- Близость к соседней точке, присоединенной к региону на предыдущем шаге
- Близость по некоторой статистике региона
- Стоимость кратчайшего пути от точки до центра региона

Существуют различные стратегии выбора зерен и выращивания регионов [14, 15, 16, 17]

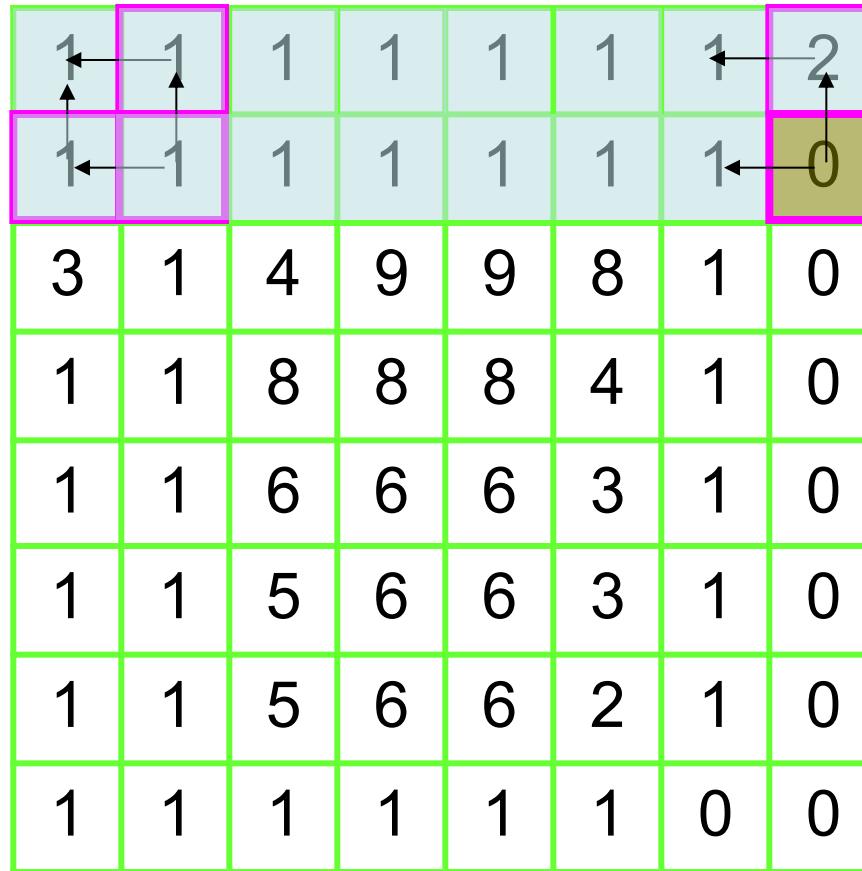
# Разрастание регионов

1. if  $|I(A) - Cl_{avg}(B)| > \delta$  and  $|I(A) - Cl_{avg}(C)| > \delta$  -  
создаем новую область, присоединяем к ней пиксел A
2. if  $|I(A) - Cl_{avg}(B)| \leq \delta$  xor  $|I(A) - Cl_{avg}(C)| \leq \delta$  –  
добавить A к одной из областей
3. if  $|I(A) - Cl_{avg}(B)| \leq \delta$  and  $|I(A) - Cl_{avg}(C)| \leq \delta$  :
  - a)  $|Cl_{avg}(B) - Cl_{avg}(C)| \leq \delta$  –  
сливаем области B и C.
  - b)  $|Cl_{avg}(B) - Cl_{avg}(C)| > \delta$  –  
добавляем пиксел A к тому классу, отклонение от  
которого минимально.



# Алгоритм разрастания регионов 1

Среднее: 1



Пример  $\delta = 1$

$$\forall p \in S \quad \left| I(p) - \frac{1}{N} \sum_{q \in S} I(q) \right| < \delta$$

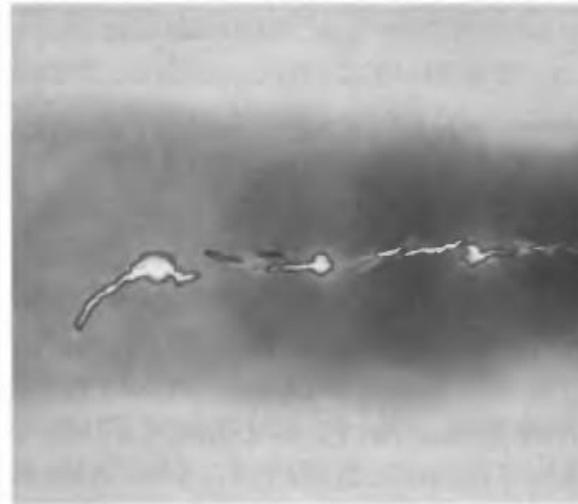
## Алгоритм разрастания регионов 2

$$\forall p \in S \quad \left| I(p) - \frac{1}{N} \sum_{q \in S} I(q) \right| \leq \delta$$

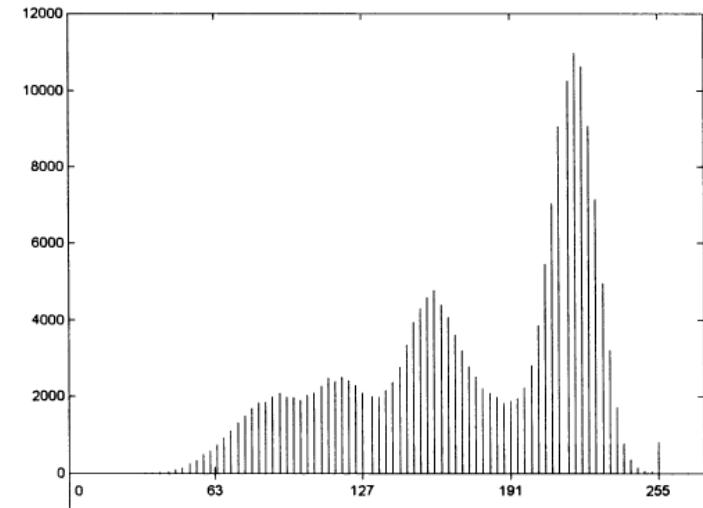
Пример  $\delta = 1$



# Сварочный шов с дефектами Центры кристаллизации



Гистограмма для сварного шва с дефектами



Результаты выращивания областей

Границы дефектов, выявленных при сегментации

# Слияние областей

1. Первый шаг – каждый пиксель это отдельная область, поместить все области в стек
2. Пока стек не пуст
  - Взять область  $S$  из стека, для всех соседних областей  $S_i$ :
    - Проверить  $S' = S \cup S_i$  на однородность
    - Если  $S'$  однородна –
      - Слить  $S$  и  $S_i$ ,  $S'$  поместить в стек,  $S_i$  из стека удалить, перейти на 2
    - Если область не однородна
      - Пробуем другого соседа

# Разделение областей

1. Первый шаг – всё изображение это одна область, поместить область в стек
2. Пока стек не пуст
  - Взять область  $S$  из стека
  - Проверить область на однородность
  - Если область неоднородна
    - разделить ее, новые области поместить в стек
  - Если область однородна
    - область больше не трогаем

# Алгоритм разбиения (split)

Первое разбиение

1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0	
1	1	8	8	8	4	1	0	
1	1	6	6	6	3	1	0	
1	1	5	6	6	3	1	0	
1	1	5	6	6	2	1	0	
1	1	1	1	1	1	1	0	0

# Алгоритм разбиения (split)

## Второе разбиение

1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

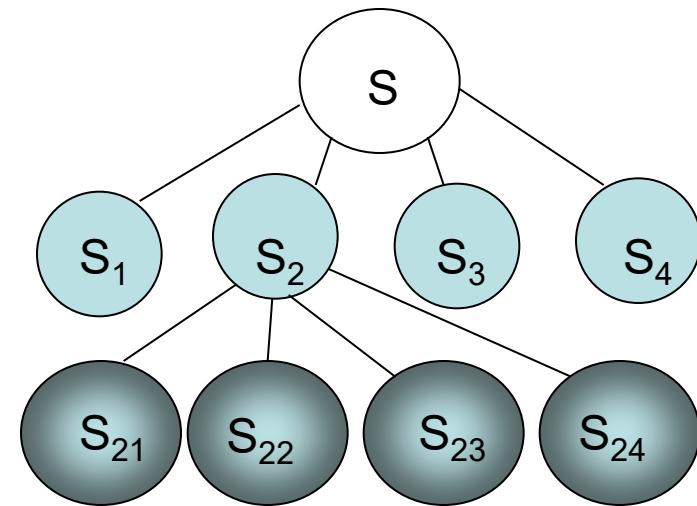
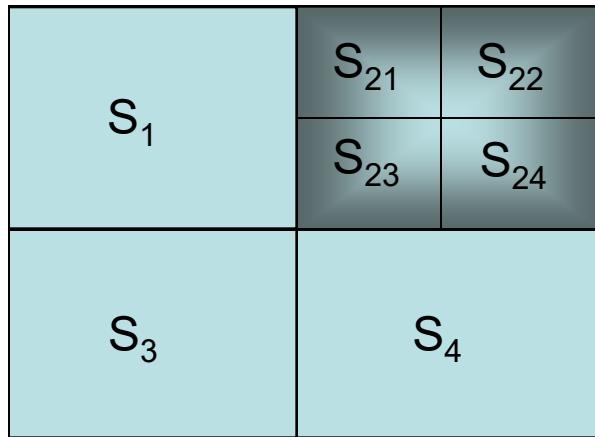
# Алгоритм разбиения (split)

## Третье разбиение

1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

# Правило разделения областей 1

- Распространенный вариант – на 4 части, как квадродерево



Просто реализовать, но границы получившихся областей вряд ли будут соответствовать границам объектов

# Недостатки алгоритмов разбиения и слияния

- Разбиение
  - Может дать слишком много регионов
  - Если использовать квадродерево, границы скорее всего будут неверны
- Слияние
  - Долго работает, если начинать с индивидуальных пикселей
- **Вывод – нужен комбинированный метод**

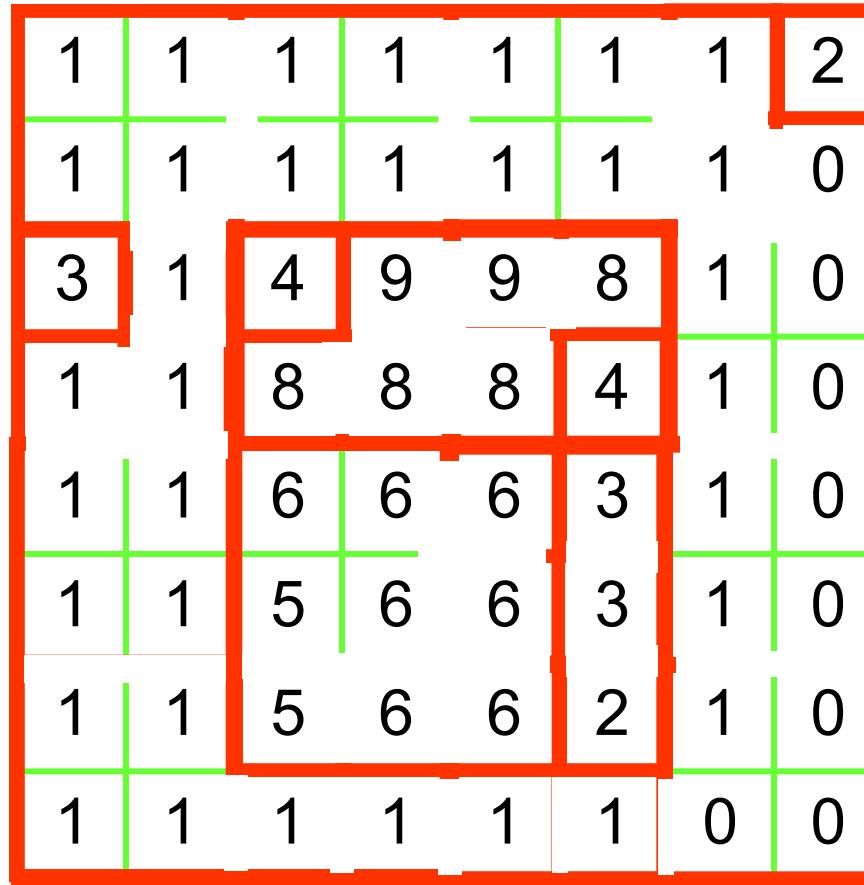
# Алгоритм разбиения/слияния (split and merge)

- Идея [4, 6] :
  - Сначала провести разбиение (дробление, пересегментация) на небольшие однородные области
    - Обычно используется принцип квадротрека
  - Затем слить между собой те из них, которые вместе не нарушают требование однородности
    - Продолжать до тех пор, пока остаются регионы которые можно объединить

# Третье разбиение из алгоритм разбиения (split)

1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

# Слияние в алгоритме разбиения/слияния (split and merge)



# Сравним результаты

1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

разбиение/слияние

1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

разрастание регионов

# Сравнение результатов разделения-слияния и пороговой обработки



Исходное



Разделение - слияние



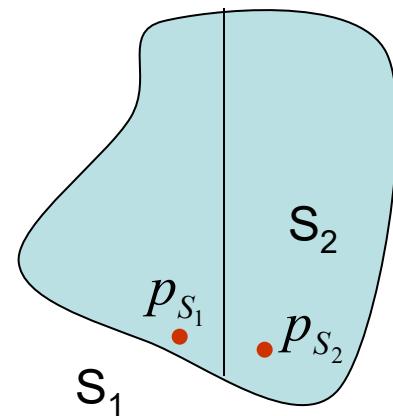
Пороговая обработка

# Алгоритм «фагоцита»

- Истивание границ
  - Убирает слабые границы
- «Слабость границ» определяется по разности яркостей граничных пикселей

$$S(p_{S_1}, p_{S_2}) = |I(p_{S_1}) - I(p_{S_2})|$$

*клетка способная захватывать и переваривать посторонние тела*

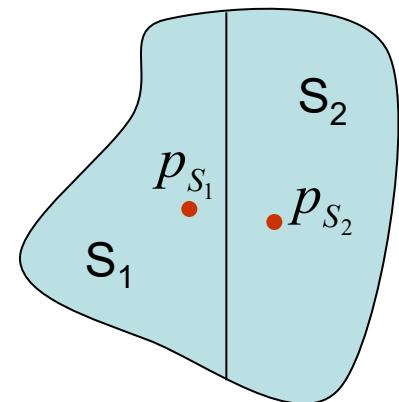


# Алгоритм «фагоцита»

$$S(p_{S_1}, p_{S_2}) = |I(p_{S_1}) - I(p_{S_2})|$$

$$W(p_{S_1}, p_{S_2}) = \begin{cases} 1 & S(p_{S_1}, p_{S_2}) > T \\ 0 & \text{иначе} \end{cases}$$

$$W(S_1, S_2) = \sum_{p_{S_1} \in R_1 \wedge p_{S_2} \in R_2} W(p_{S_1}, p_{S_2})$$



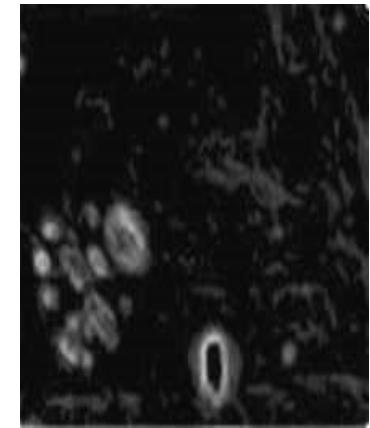
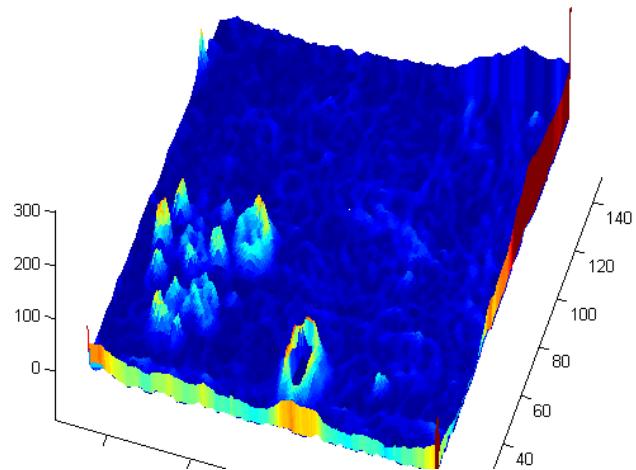
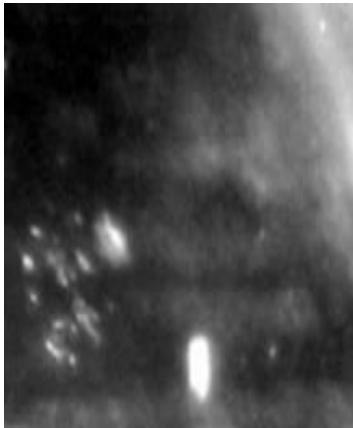
Слитъ две области если:

$$\frac{W(\text{граница})}{\text{кол - во Точек границы}} < T_3, \quad 0 < T_3 \leq 1$$

# Алгоритмы водораздела (watershed)

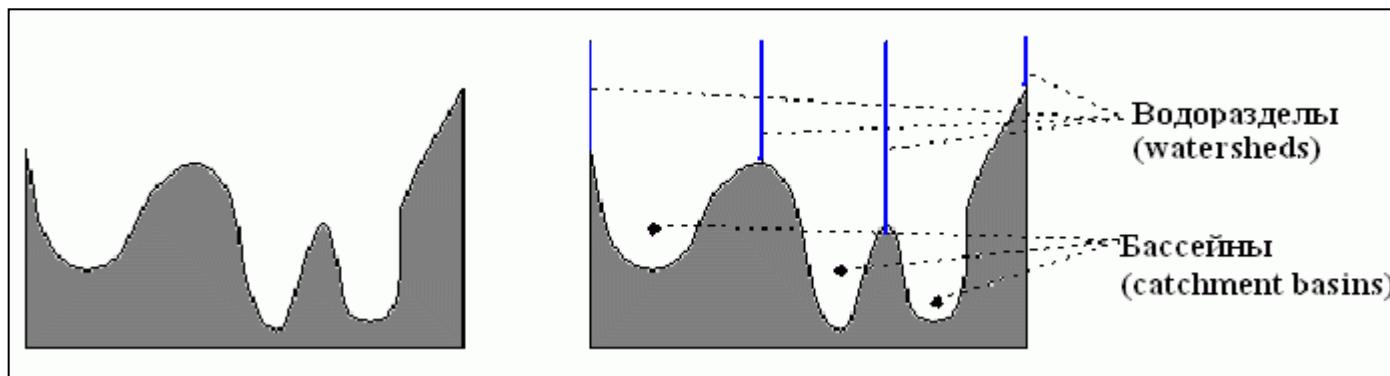
Рассмотрим изображение как карту высот ландшафта

- Значением функции может быть интенсивность или модуль градиента.
- Для наибольшего контраста можно взять градиент от изображения.



# На чём основаны алгоритмы водораздела

**Область водораздела, бассейн (*catchment basin*):** область, в которой поток из всех точек «стекает» к одной общей точке



# Основные понятия алгоритмов водораздела

1. точки локального минимума
2. точки на склоне, с которых вода скатывается в один и тот же локальный минимум
3. точки на гребне, с которой вода с более-менее равной вероятностью скатывается в более чем один минимум

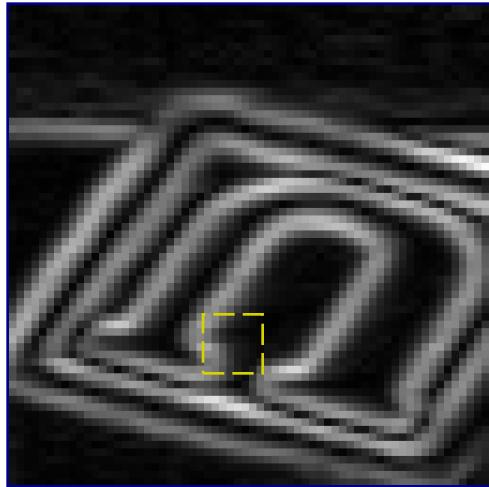
Бассейн (водосбор) локального минимума

Линии водораздела

# Алгоритмы водораздела (watershed)

- Водораздел в результате наводнения
- Водораздел по топографическому расстоянию
- Водораздел по принципу капли воды
- Межпиксельный водораздел
- Топологический водораздел
- Алгоритм затопления Мейера
- Оптимальные алгоритмы охватывающего леса (разрезы водораздела)
- И другие

# Алгоритм tobogganing

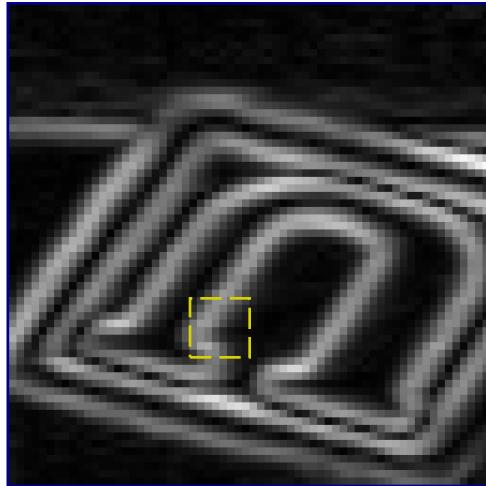


Идея

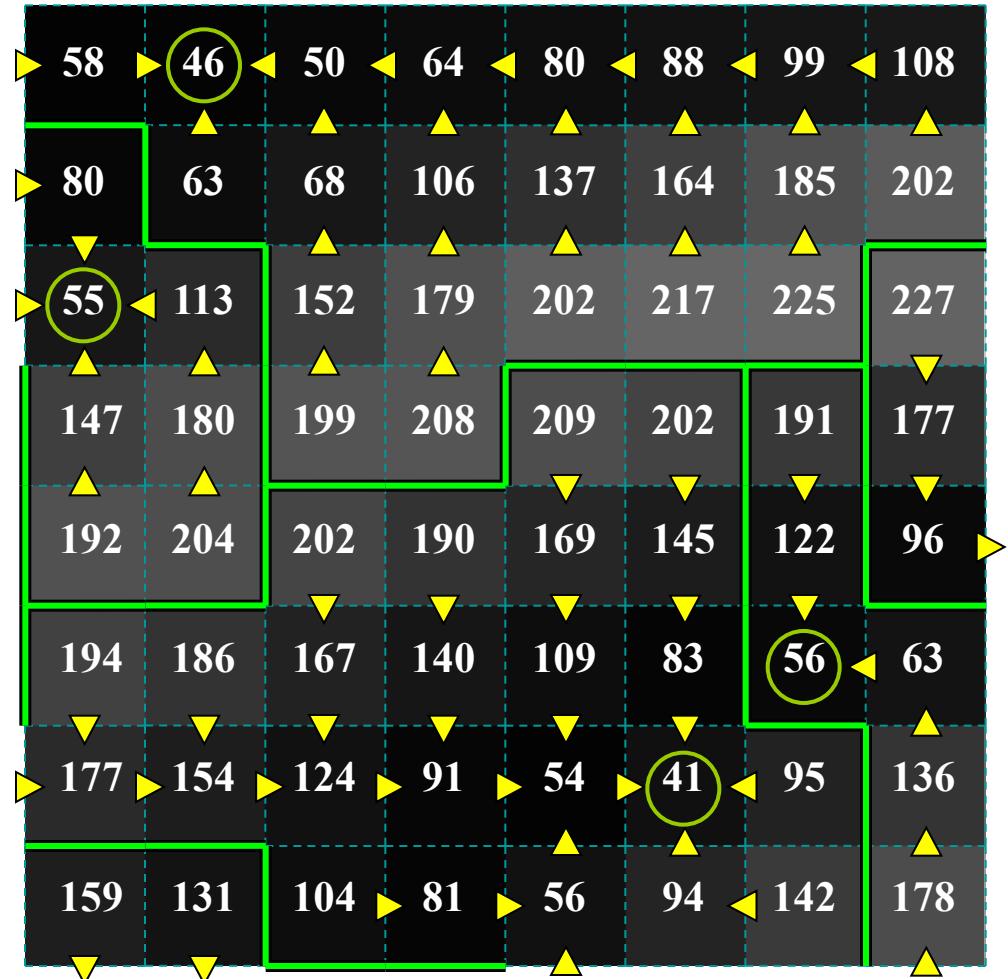
- Из каждого пикселя «спускаемся» в локальный минимум среди его соседей
- Спускаемся до тех пор, пока есть куда спускаться
- Пиксели «спустившиеся» в один минимум – одна область

58	46	50	64	80	88	99	108
80	63	68	106	137	164	185	202
55	113	152	179	202	217	225	227
147	180	199	208	209	202	191	177
192	204	202	190	169	145	122	96
194	186	167	140	109	83	56	63
177	154	124	91	54	41	95	136
159	131	104	81	56	94	142	178

# Сегментация алгоритмом tobogganing



- Из каждого пикселя «спускаемся» в локальный минимум среди его соседей
- Спускаемся до тех пор, пока есть куда спускаться
- Пиксели «спустившиеся» в один минимум – одна область



# Алгоритм «погружения» (immersion)

Начнем с самых «глубоких» (темных) пикселей  
**(они определят начальные бассейны)**

Для каждой яркости  $k$ :

Для каждой связной компоненты пикселей яркости  $k$ :

Если прилежит только к одному существующему бассейну

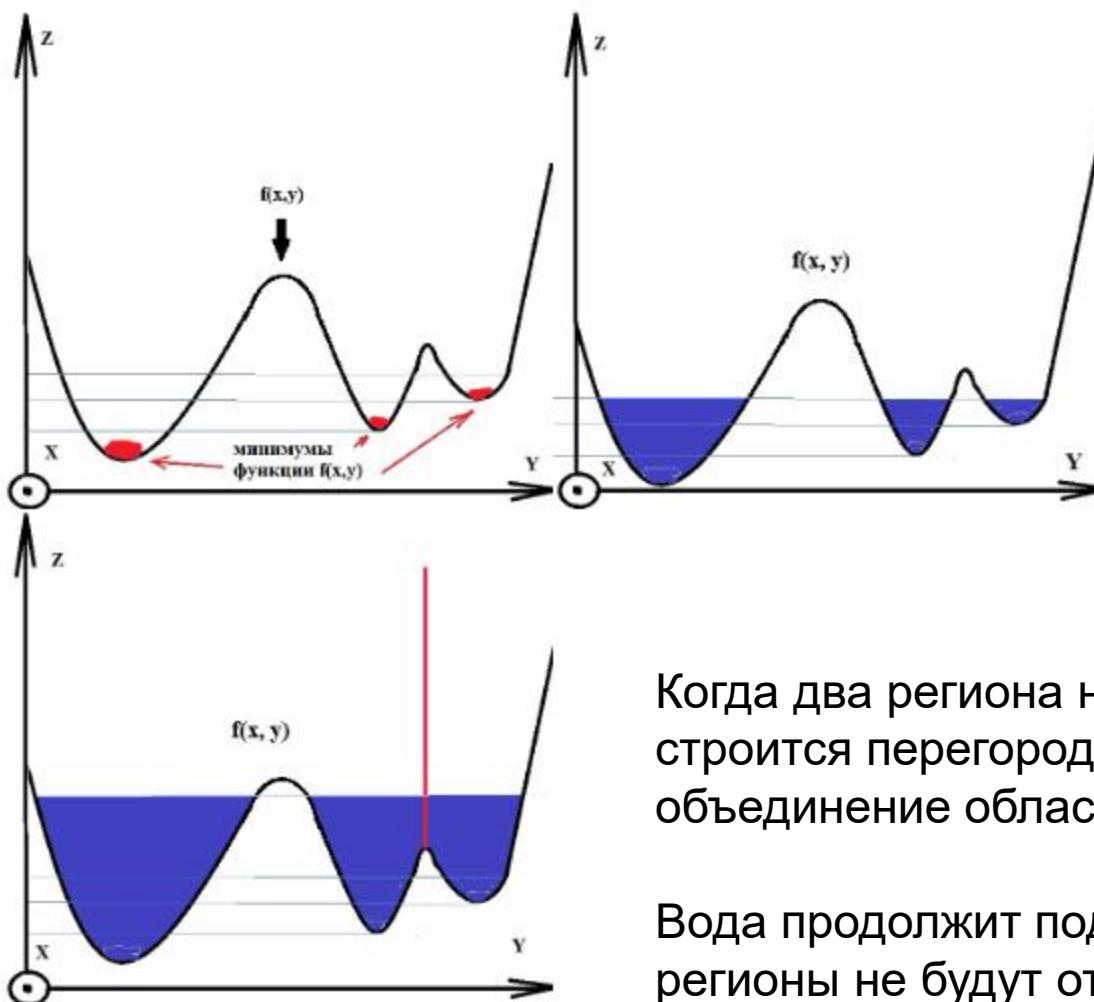
Добавить компоненту к бассейну

Иначе если прилежит более чем к одному существующему бассейну

Пометить как границу (водораздел)

Иначе – создать новый бассейн

# Алгоритм с построением перегородок



После нахождения минимумов функции  $f$ , идет процесс заполнения водой, который начинается с глобального минимума.

Как только уровень воды достигает значения очередного локального минимума, начинается его заполнение водой.

Когда два региона начинают сливаться, строятся перегородки, чтобы предотвратить объединение областей.

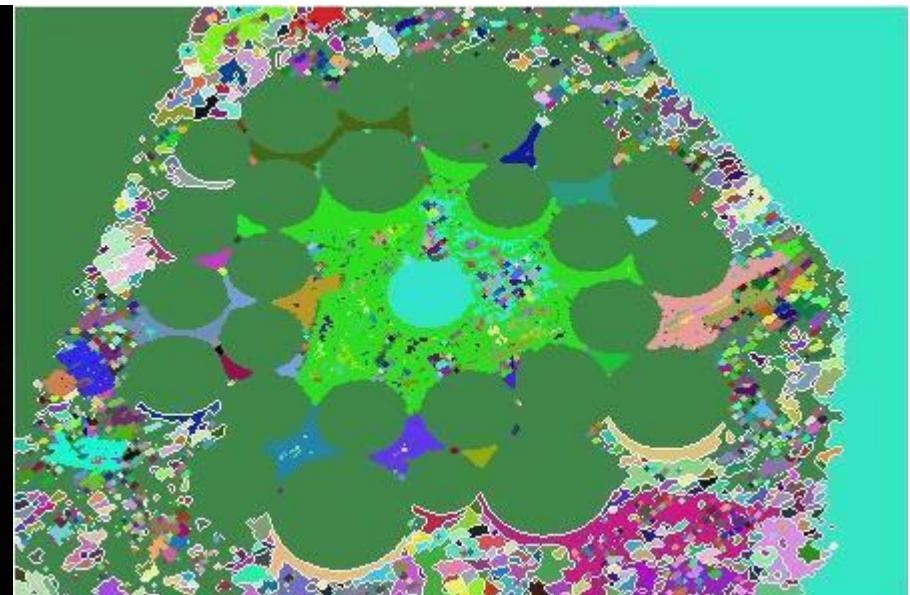
Вода продолжит подниматься до тех пор, пока регионы не будут отделяться только искусственно построенными перегородками.

# Избыточное разбиение на сегменты

Исходное изображение



Изображение после сегментации  
алгоритмом WaterShed



# Как справиться с мелкими деталями?

Чтобы избавиться от избытка мелких деталей, можно задать области, которые будут привязаны к ближайшим минимумам.

Перегородка будет строиться только в том случае, если происходит объединение двух регионов с маркерами, в противном случае будет происходить слияние этих сегментов.

Такой подход убирает эффект избыточной сегментации, но требует предварительной обработки изображения для выделения маркеров,

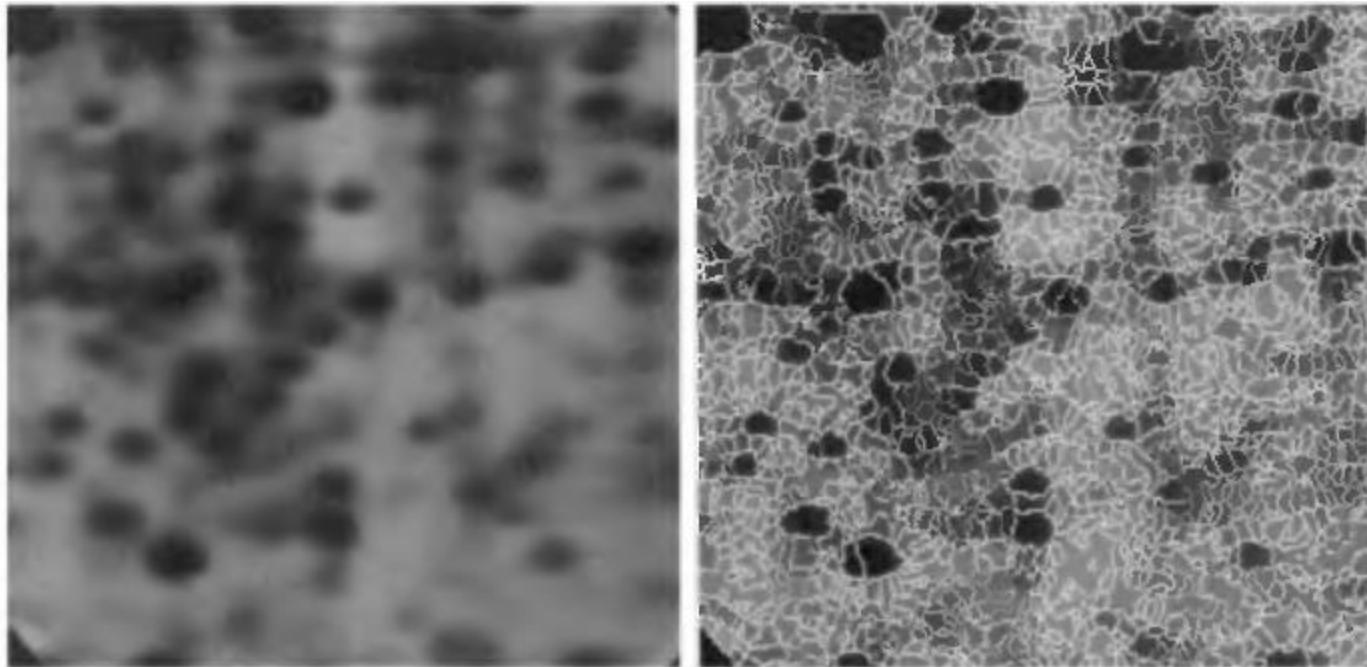
Изображение с маркерами



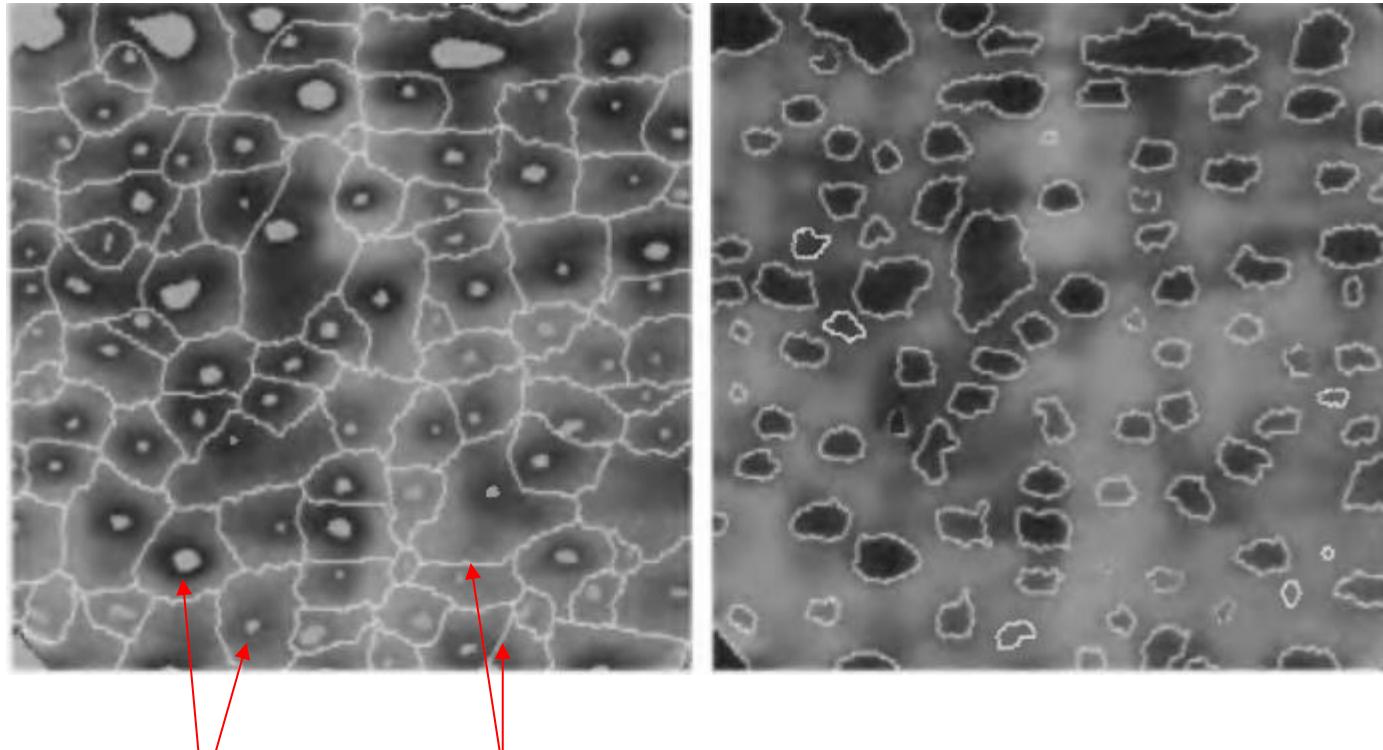
Изображение после сегментации  
алгоритмом WaterShed с  
использованием маркеров



# Избыточная сегментация



# Использование маркеров



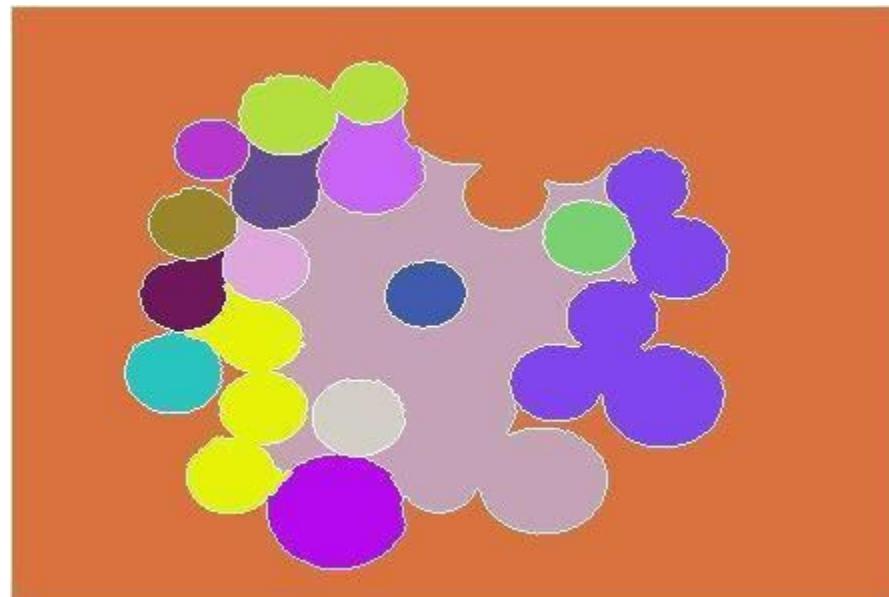
Внутренние маркеры

Внешние маркеры

Если требуется действовать автоматически без вмешательства пользователя, то можно использовать, например, функцию `findContours()` для выделения маркеров.

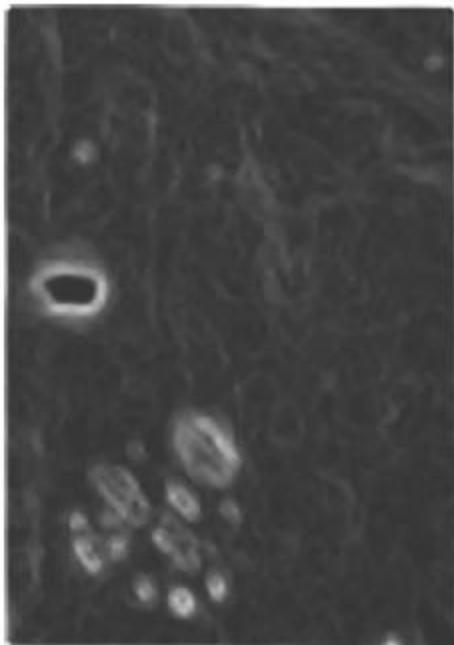
Но тут тоже для лучшей сегментации мелкие контуры следует исключить, например, убирая их по порогу по длине контура.

Или перед выделением контуров использовать эрозию с дилатацией, чтобы убрать мелкие детали.

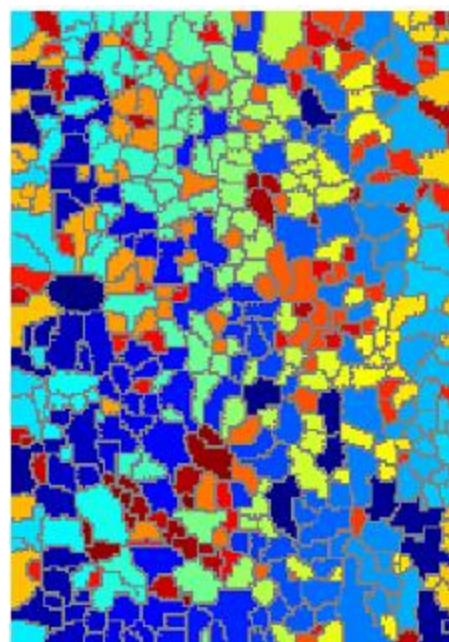


# Проблемы всех алгоритмов водораздела

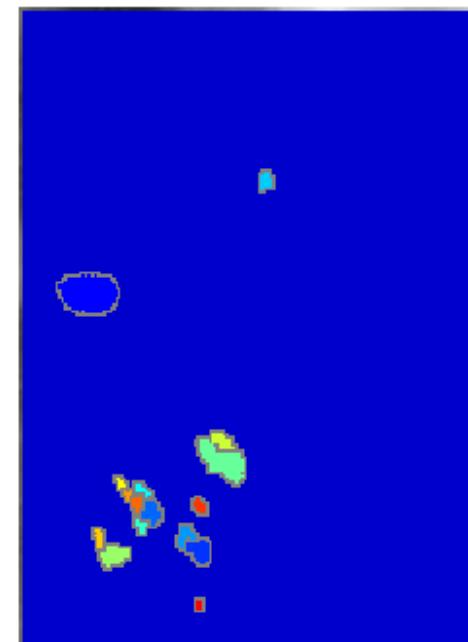
- Алгоритм дает множество небольших регионов
- Очень чувствителен к шуму – ищет все локальные минимумы
- Результат – избыточная сегментация



Абсолютная  
величина  
градиента



Сегментация по  
данному  
градиенту

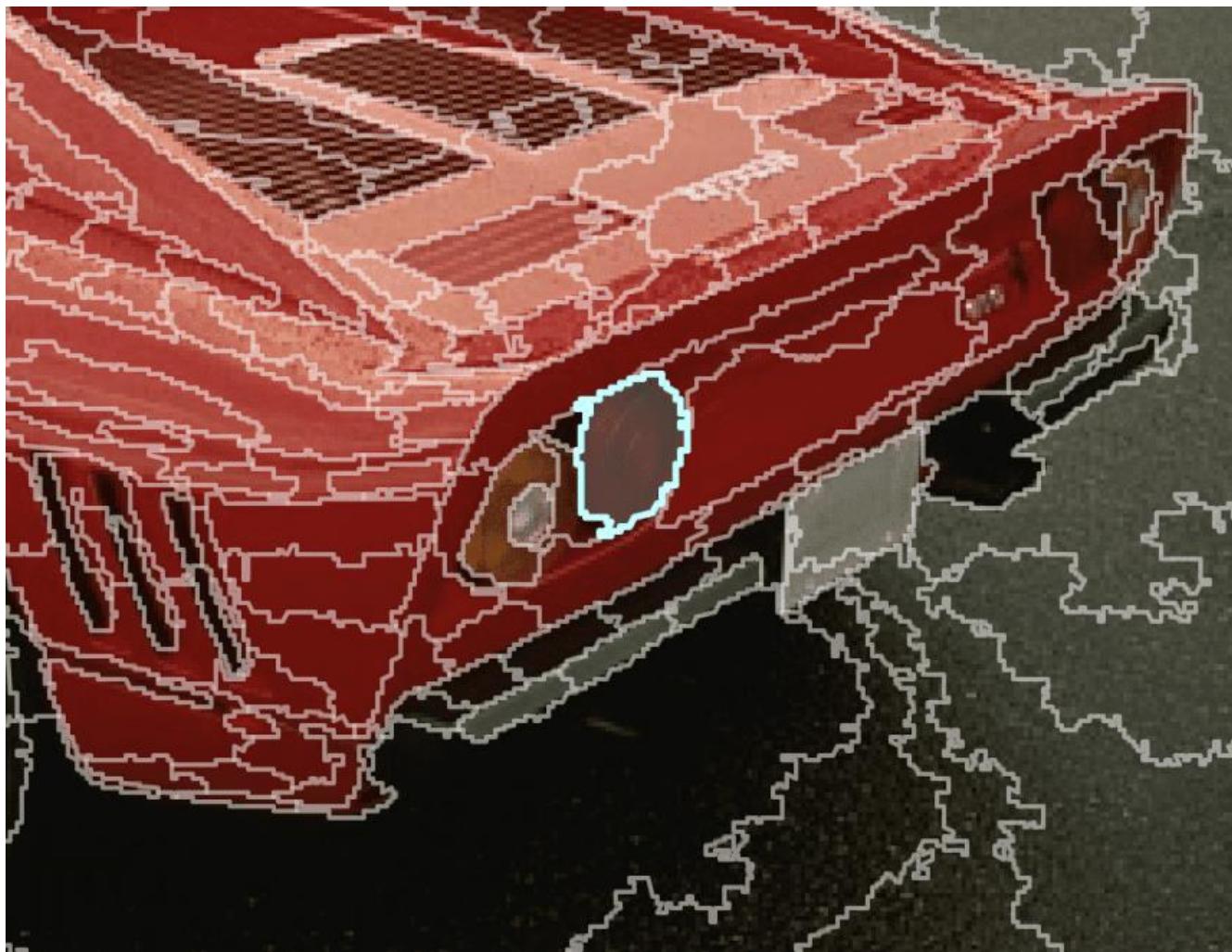


Градиент  $< 10$   
обращен в 0

# Семейства классических методов

- Основанные на формировании однородных областей
  - без пространственных связей
  - с учетом пространственных связей
- **Суперпиксели**
- Основанные на поиске краев (края и области)
  - Canny
  - Pb-детектор краёв
- Методы на графах
  - Normalized cut
  - «Эффективный метод» Felzenszwalb & Huttenlocher
- Энергетические методы
  - Snakes
  - Методы уровня

# Суперпиксели



# Суперпиксели

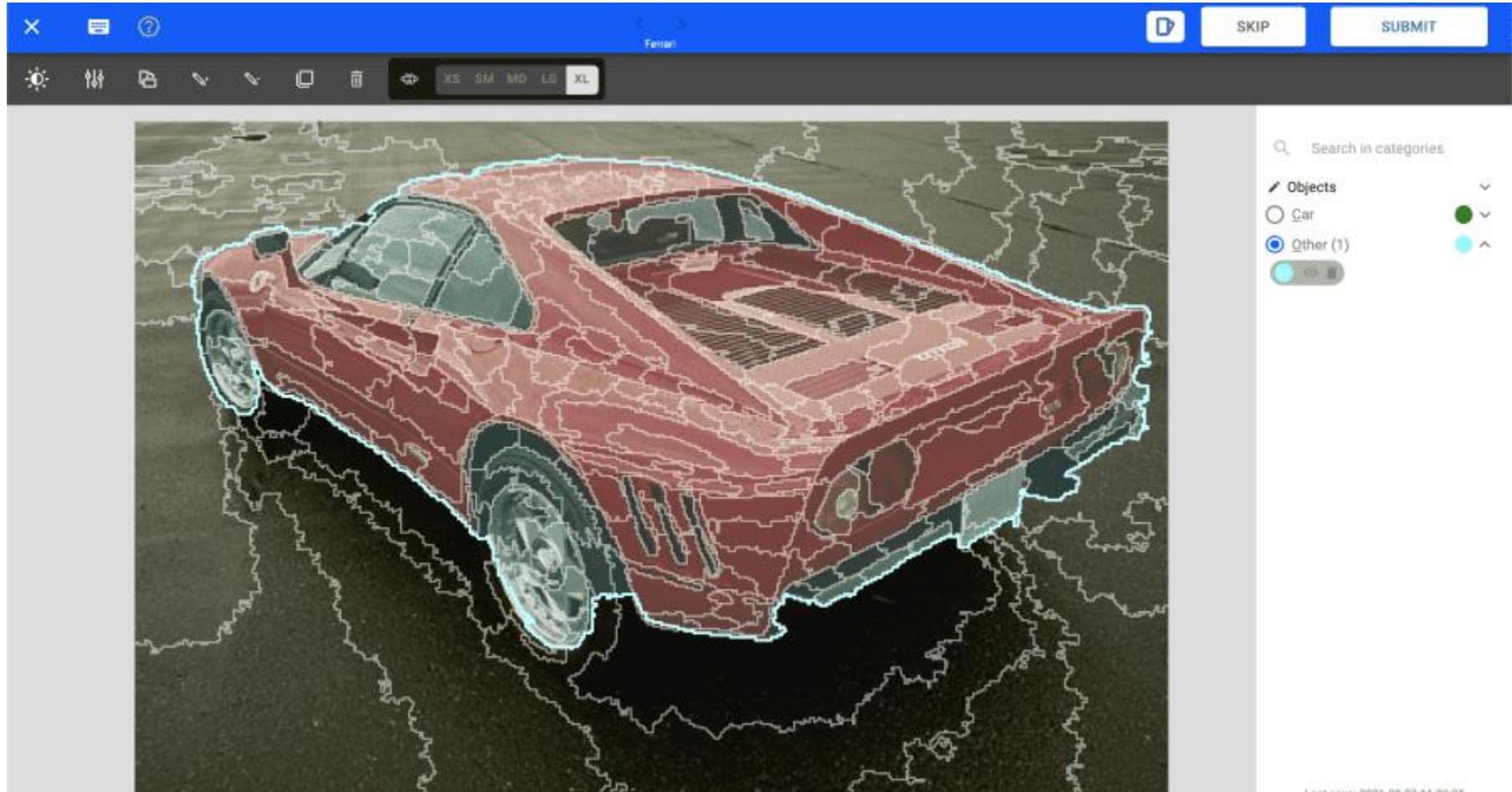
Суперпиксели, представленные Реном и Маликом в 2003 году, группируют пиксели, похожие по цвету и другим низкоуровневым свойствам.

В этом отношении суперпиксели решают две проблемы, присущие обработке цифровых изображений:

- 1) пиксели являются всего лишь результатом дискретизации;
- 2) большое количество пикселей в больших изображениях значительно увеличивают время работы алгоритмов.

Рен и Малик представляют суперпиксели как более естественные объекты – группирующие пиксели, которые воспринимаются как принадлежащие друг другу, при этом значительно сокращая количество примитивов для последующих алгоритмов

Модели ML на входе лучше получать сегментацию на уровне пикселей, а не ограничительные рамки.



**Семантика.** В суперпикселях должно быть смысловое содержание, которое выражается в соблюдении последовательности форм, цветовой однородности и т. д. Должна быть возможность легко определять границы, идеально выбирая суперпиксели.

**Целесообразность форм (компактность).** В суперпикселях должны быть похожие формы. Это не значит, что у всех суперпикселей должно быть одинаковое количество пикселей, но они не должны быть слишком маленькими или иметь очень сложные формы с тонкими областями.

**Градиенты цвета.** Суперпиксели должны быть разделены идеальными цветовыми градиентами, определяющими границы, которые будут использоваться моделями машинного обучения наилучшим образом. Этот аспект особенно интересен, ведь такие градиенты компьютеры определяют лучше людей, поэтому границы суперпикселей эффективнее и точнее.

**Скорость.** При расчёте суперпикселей может требоваться большой объём вычислений. Важно проводить их в разумные сроки, избегая пустой траты времени: чтобы они оказывались быстрее аннотаций, выполняемых человеком.

**Различные разрешения.** Должна быть возможность менять размер суперпикселей. Определить размер бывает сложно, но можно прикинуть средний размер или общее количество суперпикселей. Меняя разрешение суперпикселя, можно сделать аннотации точнее.

**Соответствие.** Когда при изменении разрешения количество суперпикселей увеличивается, границы не должны удаляться никогда, вместо этого должны только добавляться новые, а иначе часть работы потерянется при переключении разрешений, что чревато проблемами: не все алгоритмы могут предотвратить потери.

# Классы алгоритмов

Есть два главных класса алгоритмов вычисления суперпикселей:

- графовые
- кластерные

Графовые методы интерпретируют каждый пиксель как узел графа, а его рёбра — как привязки. Затем пиксели объединяются в суперпиксели.

Кластерные методы постепенно детализируют кластеры пикселей, пока не достигается соответствие заданному критерию.

# TurboPixel

- Алгоритм, специально нацеленный на сегментацию изображения на суперпиксели
- Даёт суперпиксели примерно одного размера, равномерно распределенные по изображению
- Использует подход линий уровня для сегментации
- Сложность  $O(N)$ , где  $N$  – пиксели
- При увеличении количества суперпикселей даже ускоряется

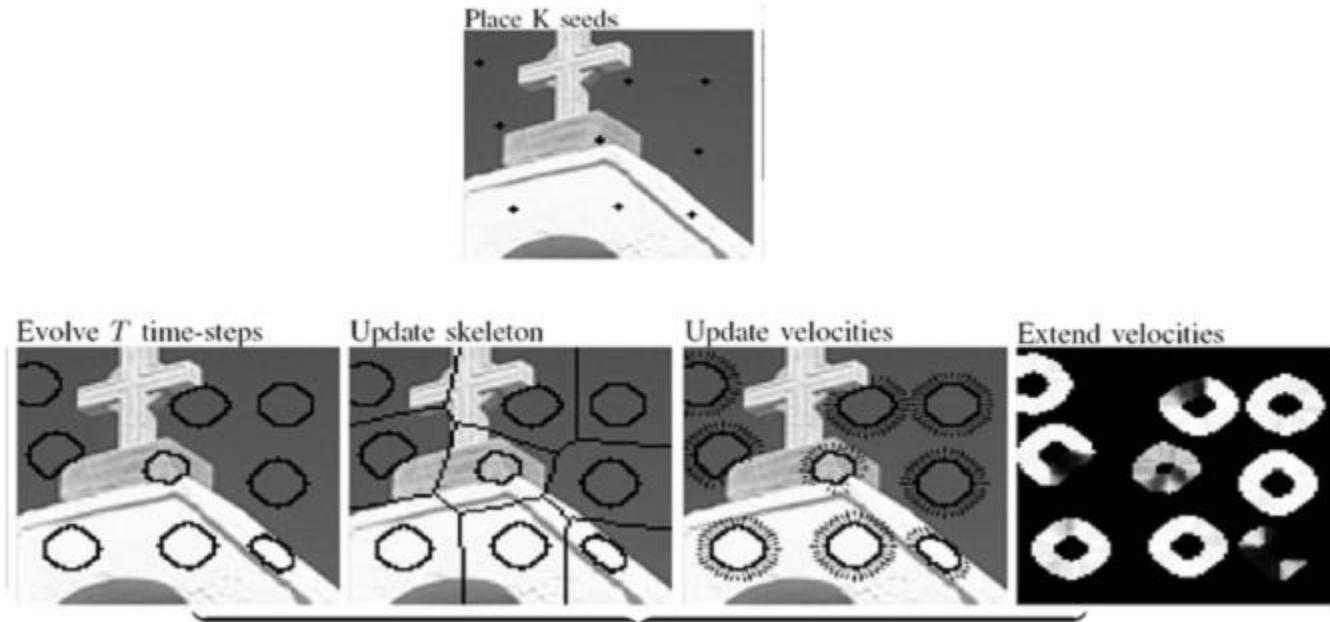


Alex Levinstein, et. al, TurboPixels: Fast Superpixels Using Geometric Flows, PAMI 2009

<https://www.cim.mcgill.ca/~shape/publications/pami09.pdf>

# Идея алгоритма

- Скорость движения «контура» зависит от градиента, близости к предполагаемой границе региона и т.д.
- Благодаря этому суперпиксели «тормозятся» у краёв изображения и делят его на фрагменты похожего размера



# Схема алгоритма

## 1. Инициализация «зародышами» (Seeding)

- Изображение сначала разбивается на регулярную сетку.
- Центры каждого квадрата сетки становятся начальными точками («зародышами») для будущих суперпикселей.
- Вокруг каждого зародыши создается малый начальный контур (например, круг небольшого радиуса).

## 2. Геометрический поток и уровневые множества

Каждый суперпиксель представлен в виде функции уровня (level set function)  $\phi(x, y)$

$\phi(x, y) = 0$  определяет границу (контур) суперпикселя

$\phi(x, y) > 0$  — это внутренность суперпикселя

$\phi(x, y) < 0$  — это внешняя область

Чтобы заставить контур «расти», применяется уравнение эволюции, известное как Уравнение в частных производных (УрЧП):

$\partial\phi/\partial t = -V |\nabla\phi|$ , где:

$\partial\phi/\partial t$  — скорость изменения функции уровня во времени

$V$  — скорость расширения

$|\nabla\phi|$  — градиент функции уровня (обеспечивает корректное распространение контура)

### 3. Управление скоростью расширения (Velocity Term V)

Скорость  $V$  — не постоянная. Она умно адаптируется, чтобы рост останавливался на реальных границах объектов.

Она состоит из двух основных компонентов:

- Сглаживающее слагаемое (Smoothing Term):  
Заставляет контур оставаться гладким и геометрически "приятным", противодействуя образованию шумных, изрезанных границ.
- Останавливающее слагаемое (Stopping Term):  
Замедляет или полностью останавливает рост контура, когда он встречает резкий перепад цветов (градиент) на изображении. Если пиксель на пути роста сильно отличается от пикселей внутри текущего суперпикселя, скорость  $V$  в этой точке падает до нуля.

#### 4. Процесс одновременного роста

- Все суперпиксели начинают расти одновременно из своих зародышей с одинаковой начальной скоростью.
- По мере роста их контуры сталкиваются друг с другом. В точках столкновения рост прекращается, и формируется четкая граница между суперпикселями.
- В областях с однородным цветом (где нет границ) суперпиксели будут расти, пока не встретят соседа. В результате они заполняют всё пространство, образуя структуру, похожую на пенообразную или пчелиную соту.
- В областях с сильными градиентами (краями объектов) рост останавливается раньше, что позволяет границам суперпикселей точно следовать за границами объектов.



# Преимущества TurboPixels

Геометрическая регулярность: Создает одни из самых визуально приятных и компактных суперпикселей.

Гарантированная связность: Критически важно для многих последующих алгоритмов анализа.

Хорошее сохранение границ: Эффективно останавливается на контрастных краях.

Равномерное покрытие: Изображение делится на регионы примерно одинакового размера, что удобно для многих задач.

# Недостатки TurboPixels

Вычислительная сложность: Значительно медленнее, чем более современные методы, такие как SLIC или SEEDS.

Протекание на слабых границах: Если градиент недостаточно силен, контур может "протечь" через границу объекта.

Меньшая адаптивность: Менее эффективен на текстурированных областях по сравнению со спектральными методами (вроде LSC), так как *relies* в основном на цветовой градиент.

# SLIC

Простой алгоритм суперпикселей с линейной итеративной кластеризацией (SLIC), который использует метод кластеризации k-средних для эффективного генерирования суперпикселей.

Несмотря на свою простоту, SLIC может получить хорошие границы. В то же время он имеет более высокую скорость, более высокую эффективность памяти и может улучшить производительность сегментации.

В настоящее время наиболее распространенными алгоритмами сегментации суперпикселей являются SLIC, SEEDS и LSC.

# Схема алгоритма SLIC

## 1. Инициализация:

- Изображение разбивается на сетку из  $K$  одинаковых прямоугольников. Центры этих прямоугольников становятся начальными центрами кластеров (суперпикселей).

## 2. Создание нового пространства:

- Для учета и цвета, и координат создается 5-мерное пространство  $[l, a, b, x, y]$ .
- $[l, a, b]$  – цвет пикселя в LAB-пространстве (воспринимается более равномерным, чем RGB).
- $[x, y]$  – пространственные координаты пикселя.

# Итеративное обновление

Каждому пикслю присваивается метка ближайшего центра в пределах его окна поиска.

После присвоения всех меток, центры кластеров пересчитываются как средние значения всех пикселей в кластере.

Шаги присвоения и обновления повторяются несколько раз (обычно 10) до сходжения.

# Расстояние с ограничением

Вместо того чтобы сравнивать каждый пиксель со всеми центрами, SLIC использует «окно поиска» размером  $2S \times 2S$  (где  $S = \sqrt{N/K}$  – примерный размер суперпикселя).

Вычисляется комбинированное расстояние  $D$  между пикселем и центром кластера:

$d_{color}$  – евклидово расстояние в LAB-пространстве.

$d_{xy}$  – евклидово расстояние в координатном пространстве.

$$D = \sqrt{ (d_{color}/m)^2 + (d_{xy}/S)^2 },$$

где  $m$  – параметр, регулирующий баланс между важностью цвета и пространственной близости. Чем больше  $m$ , тем более пространственно-компактные суперпиксели получаются.

---

**Algorithm 1** SLIC superpixel segmentation

---

*/\* Initialization \*/*

Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .

Move cluster centers to the lowest gradient position in a  $3 \times 3$  neighborhood.

Set label  $l(i) = -1$  for each pixel  $i$ .

Set distance  $d(i) = \infty$  for each pixel  $i$ .

**repeat**

*/\* Assignment \*/*

**for** each cluster center  $C_k$  **do**

**for** each pixel  $i$  in a  $2S \times 2S$  region around  $C_k$  **do**

        Compute the distance  $D$  between  $C_k$  and  $i$ .

**if**  $D < d(i)$  **then**

            set  $d(i) = D$

            set  $l(i) = k$

**end if**

**end for**

**end for**

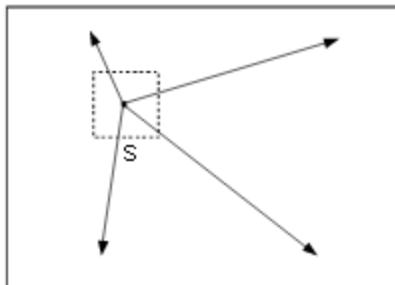
*/\* Update \*/*

Compute new cluster centers.

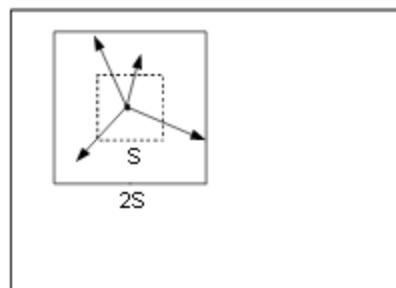
Compute residual error  $E$ .

**until**  $E \leq \text{threshold}$

---



(a) standard k-means searches  
the entire image



(b) SLIC searches  
a limited region

# Преимущества и недостатки SLIC

## Преимущества:

- Простота и эффективность.
- Высокая скорость.
- Простота использования (легко контролировать размер и компактность).
- Хорошо сохраняет границы.

## Недостатки:

- Чувствителен к начальной инициализации.
- Может плохо работать на градиентах и сложных текстурах.

# Результат SLIC Superpixel



Images segmented using SLIC into superpixels of size 64, 256, 1024 pixels (approximately).

# SEEDS (Superpixels Extracted via Energy-Driven Sampling)

## Ключевая идея:

- Это энергетический метод, который **не использует итеративное** перераспределение, как K-means.
- Вместо этого он начинает с блочного разбиения и постепенно «перекраивает» границы суперпикселей, чтобы максимизировать цветовую однородность.

# Схема алгоритма SEEDS

Инициализация блоками:

Изображение делится на крупные регулярные блоки (например, 16x16 пикселей). Это начальные суперпиксели.

Определение энергии:

Вводится энергетическая функция, которая оценивает «качество» разбиения.

Основная цель — максимизировать цветовую однородность внутри каждого суперпикселя. Однородность часто измеряется с помощью гистограммы в цветовом пространстве (например, LAB).

# Итеративное обновление границ

На каждом шаге алгоритм рассматривает пиксели, находящиеся на границе между двумя суперпикселями.

Для каждого такого граничного пикселя проверяется гипотетический «обмен»: что будет, если переместить этот пиксель к соседнему суперпикслю?

Если этот «обмен» увеличивает общую однородность (уменьшает энергию), то пиксель перемещается.

Процесс начинается с грубого уровня (перемещение крупных блоков пикселей) и постепенно переходит к более тонкому (перемещение отдельных пикселей).

# Преимущества и недостатки SEEDS

## Преимущества:

- Очень быстрый. Часто работает быстрее SLIC, так как проверяет только граничные пиксели.
- Высокое качество границ.
- Менее чувствителен к шуму, так метод основан на статистике (гистограммах), а не на отдельных пикселях.

## Недостатки:

- Суперпиксели могут быть не такой строгой формы, как в SLIC (менее компактны).
- Сложнее контролировать точный размер и количество суперпикселей.

# LSC (Linear Spectral Clustering)

Ключевая идея:

Это более математически сложный метод, который преобразует задачу в пространство спектральных признаков, где данные легче разделить на кластеры.

Похож по идее на алгоритм Normalized Cuts, но значительно ускоренный.

# Схема алгоритма LSC

Нелинейное вложение:

Каждый пиксель  $i$  с координатами  $(x_i, y_i)$  и цветом  $(l_i, a_i, b_i)$  представляется в виде точки в 10-мерном пространстве.

Это пространство строится так, чтобы евклидово расстояние в нем отражало не только разницу в цвете и координатах, но и степень сходства в контексте всего изображения. Это «масштабируемое» представление.

# Схема алгоритма LSC

Линейное разделение:

Основная «магия» LSC в том, что он использует математический трюк (аппроксимация ядра) чтобы эффективно спроектировать данные в это новое пространство, где они становятся линейно разделимыми.

В этом новом пространстве простой и быстрый алгоритм, подобный K-means, может легко разделить пиксели на компактные кластеры.

Кластеризация:

В преобразованном пространстве запускается эффективная кластеризация, результатом которой и являются суперпиксели.

# Преимущества и недостатки LSC

## Преимущества:

- Высокое качество сегментации. LSC часто превосходит SLIC и SEEDS по точности соблюдения границ объектов, особенно на текстурированных областях.
- Вычислительная эффективность (несмотря на сложную математику, реализация оптимизирована).
- Супераксели получаются регулярными и хорошо облегают границы.

## Недостатки:

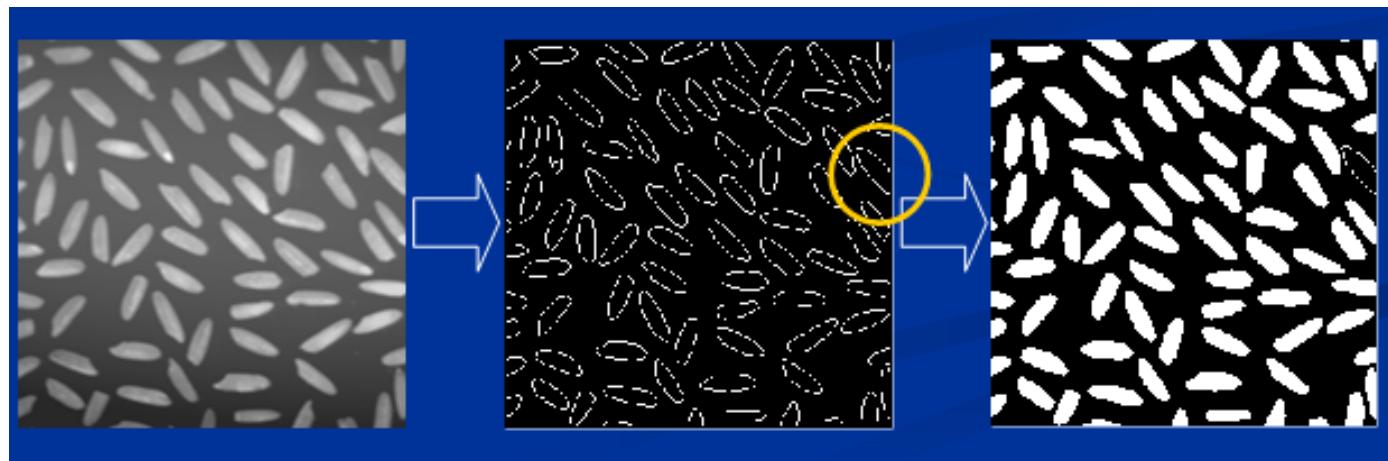
- Сложность реализации. Алгоритм сложнее для понимания и самостоятельной реализации.
- Больше параметров для настройки по сравнению с SLIC.

# Семейства методов

- Основанные на формировании однородных областей
  - без пространственных связей
  - с учетом пространственных связей
- Суперпиксели
- **Основанные на поиске краев (края и области)**
  - Canny
  - Pb-детектор краёв
- Методы на графах
  - Normalized cut
  - «Эффективный метод» Felzenszwalb & Huttenlocher
- Энергетические методы
  - Snakes
  - Методы уровня

# Сегментация посредством выделения контуров

1. Найдём все контуры на изображении алгоритмом Canny или РВ-детектором.
2. Найдем все замкнутые контуры.
3. «Внутренности» замкнутых контуров являются искомыми однородными областями.



# Методы, основанные на операторах выделения краёв

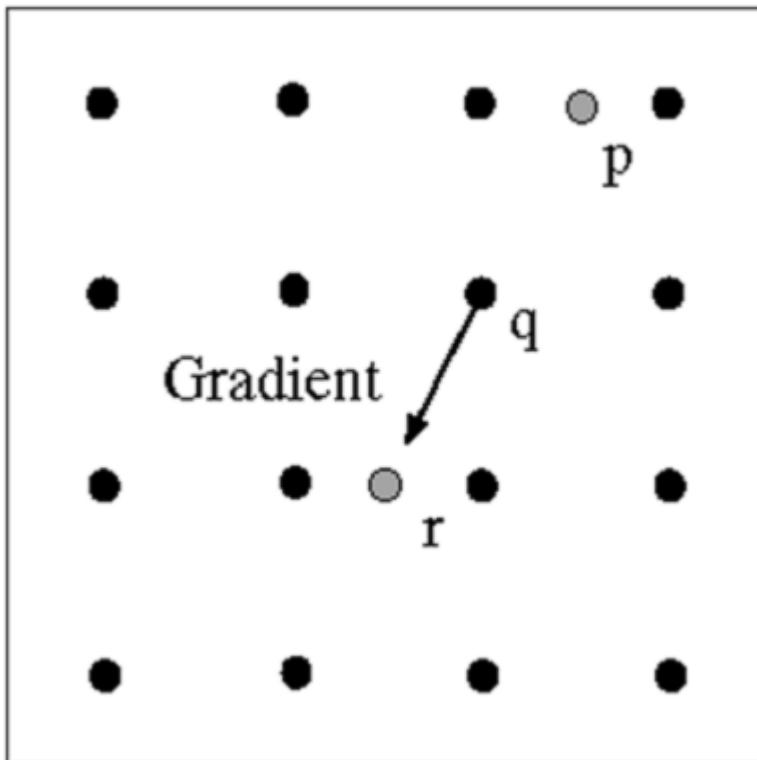


# Детектор Canny

- Свертка изображения с ядром – размытие изображения фильтром Гаусса
- Поиск значения и направления градиента
- Подавление немаксимумов (Non-maximum suppression)
  - выделение локальных максимумов
  - уточнение полос в несколько пикселей до одного пикселя
- Связывание краев и обрезание по порогу (гистерезис)
  - Определяем пороги: нижний и верхний
  - Верхний порог используем для инициализации кривых
  - Нижний порог используем для продолжения кривых

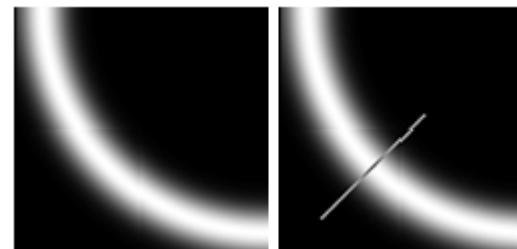
J.Canny, A Computational Approach To Edge Detection

# Поиск локальных максимумов



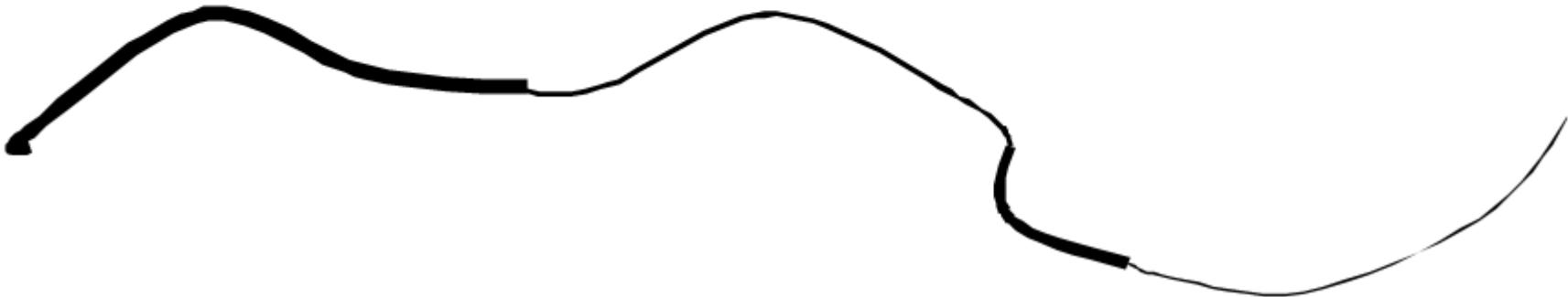
Source: D. Forsyth

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

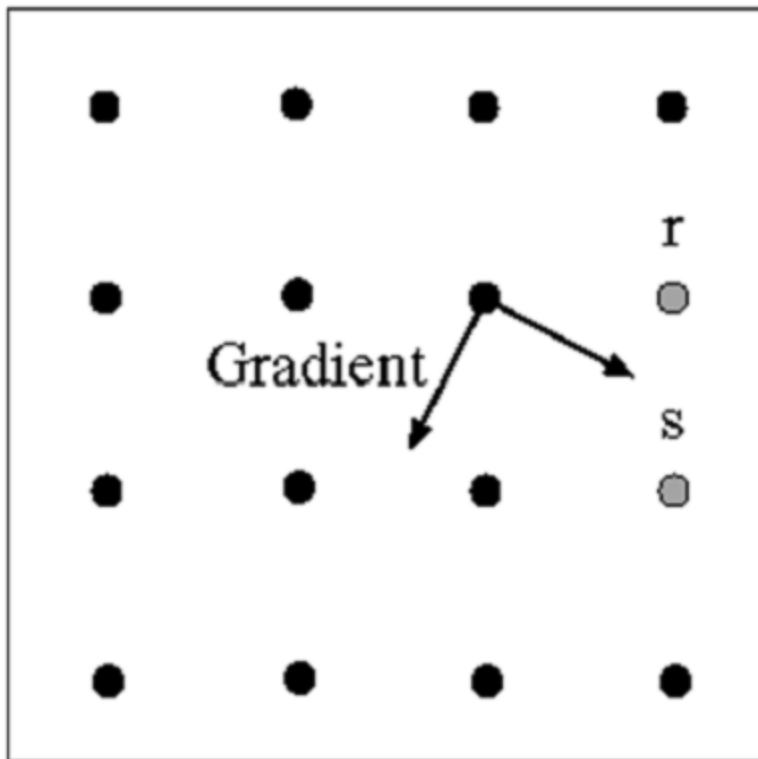


# Отсечение по порогу

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

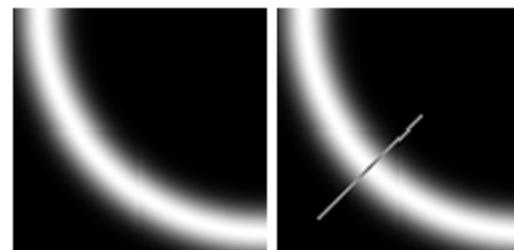


# Связывание точек



Source: D. Forsyth

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



# Эффект гистерезиса



Исходное изображение



Высокий порог  
(сильные края)



Низкий порог  
(слабые края)



Порог по гистерезису

# Пример

Карта силы краев



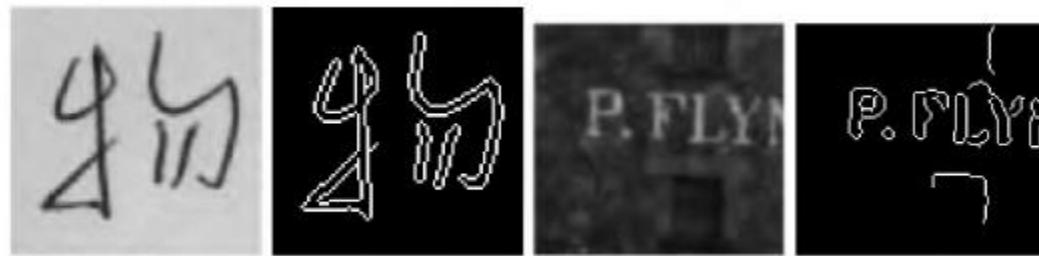
Утончение краев



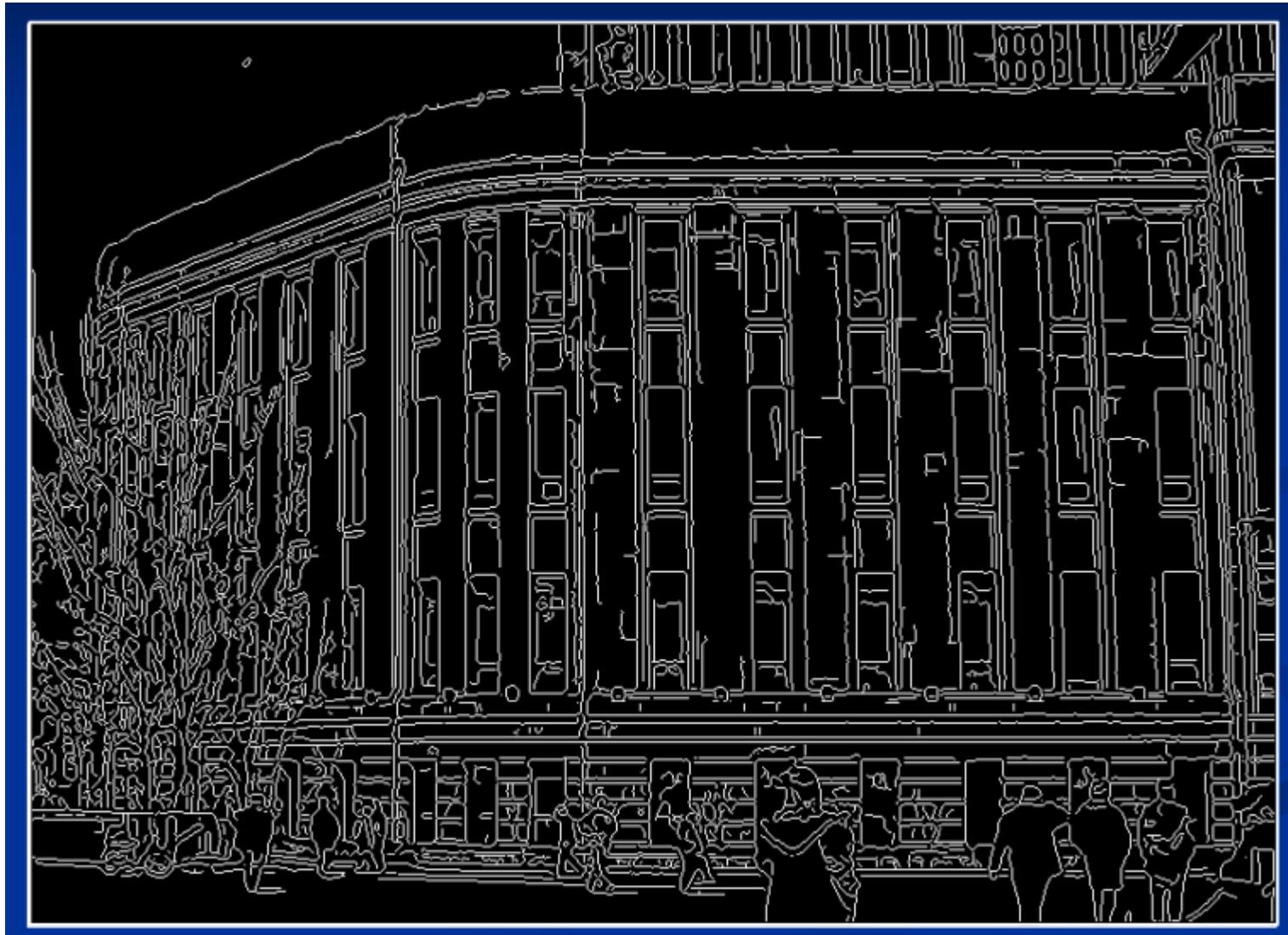
Обрезание по порогу



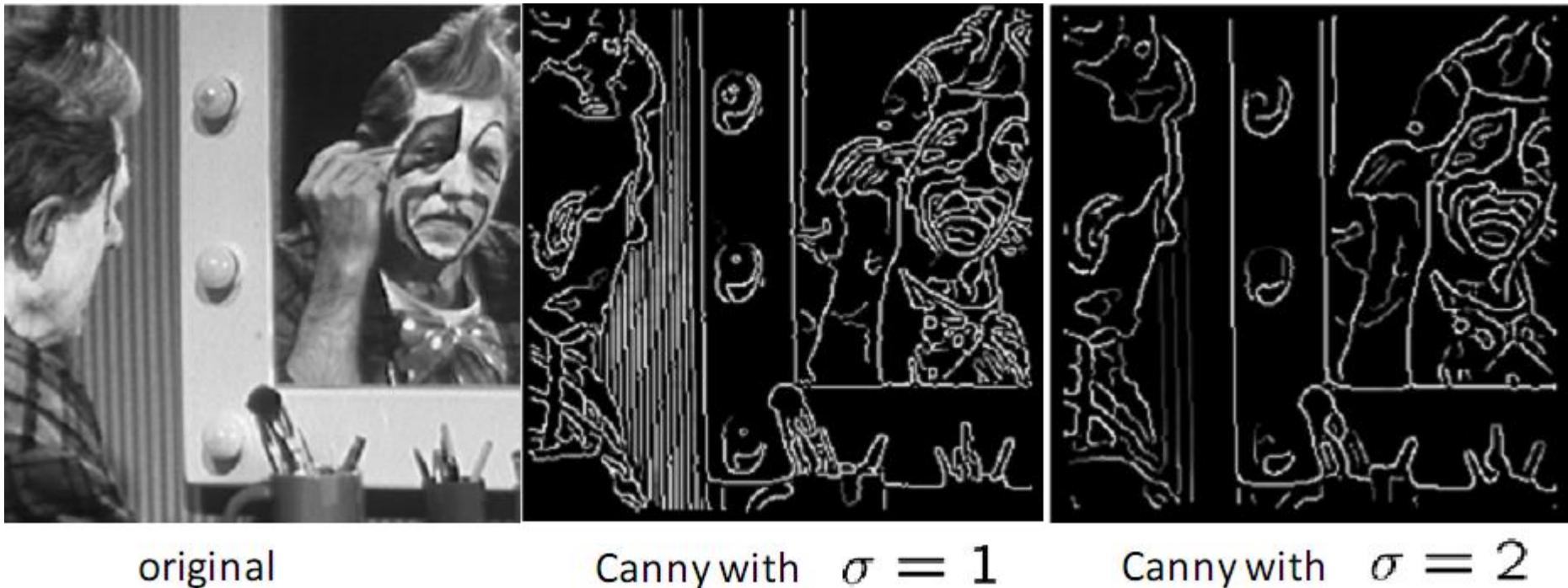
# Удачный пример применения Canny



# Canny



# Влияние $\sigma$ (размер ядра размытия)



original

Canny with  $\sigma = 1$

Canny with  $\sigma = 2$

Выбор  $\sigma$  зависит от задачи

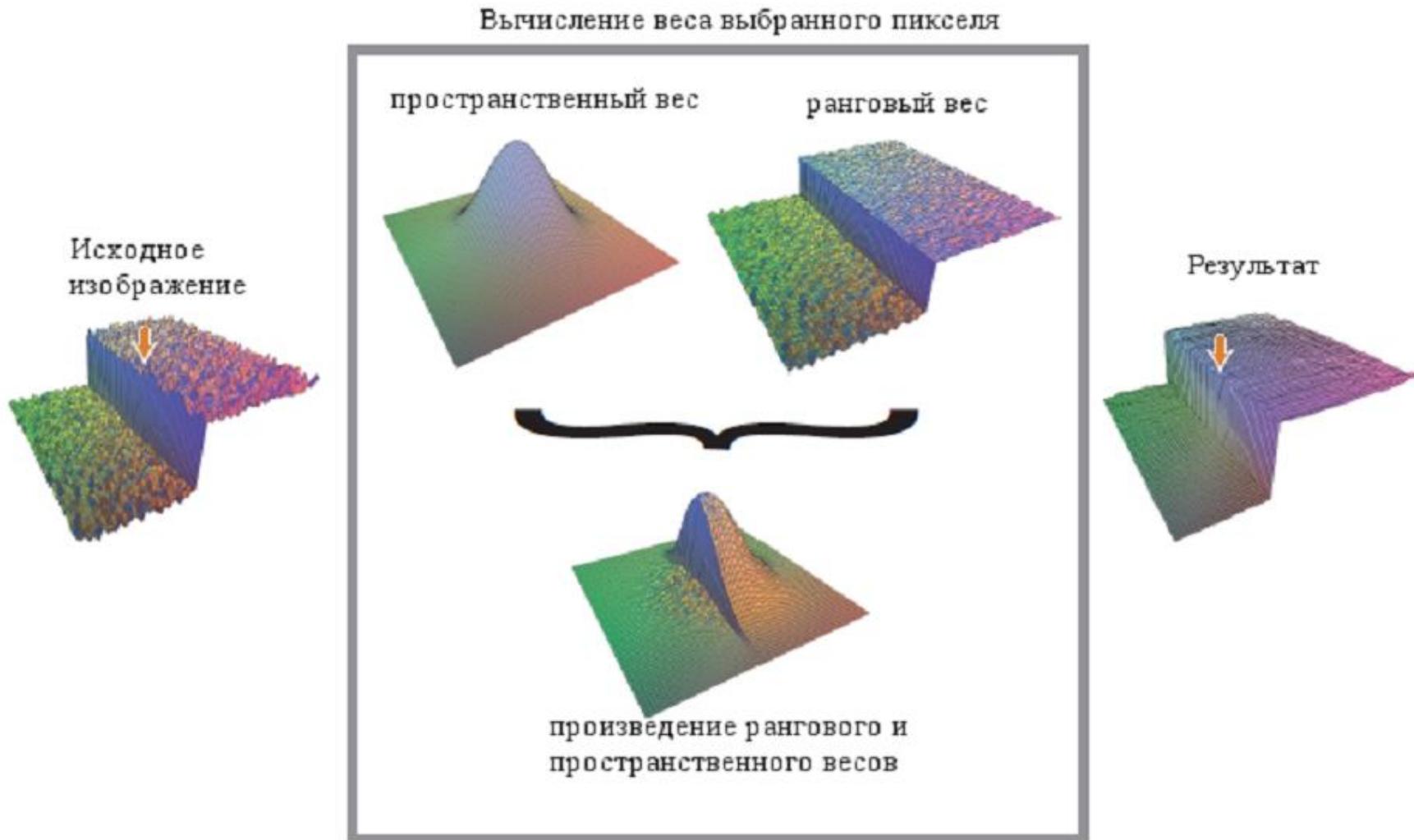
- большое  $\sigma$  - поиск крупных границ
- маленькое  $\sigma$  - выделение мелких деталей

Source: S. Seitz

# Сглаживание билатеральным фильтром с сохранением границ



# Визуализация билатерального фильтра с сохранением границ



# Билатеральный фильтр с сохранением границ

$$h(a_0) = k^{-1} \sum_{i=0}^{n-1} f(a_i) \times g(a_i) \times r(a_i)$$

$$r(a_i) = e^{\frac{(f(a_i) - f(a_0))^2}{2\sigma^2}}$$
$$g(x, y) = e^{-\frac{x^2 - y^2}{2t^2}}$$

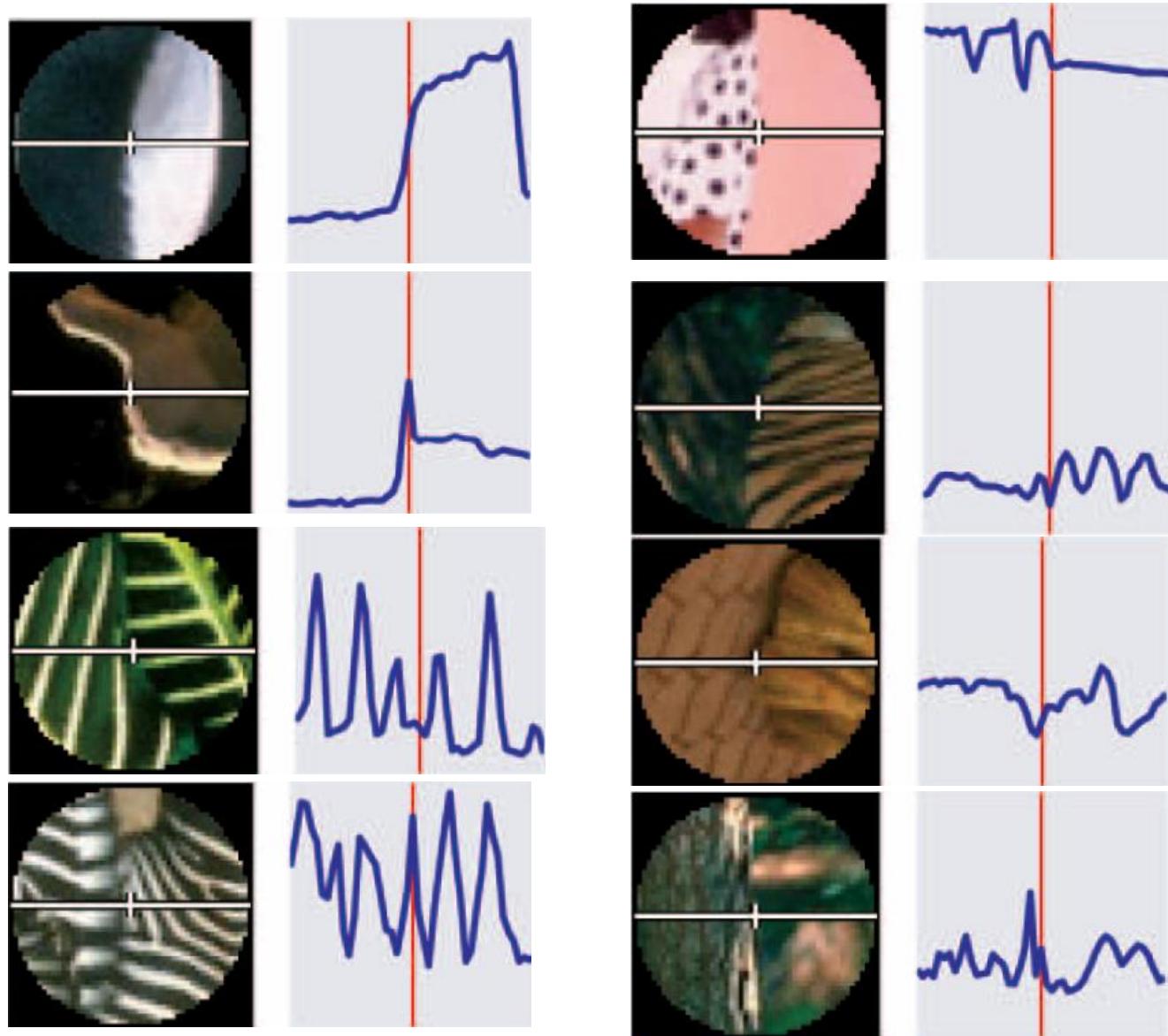
$$k = \sum_{i=0}^{n-1} g(a_i) * r(a_i)$$

# Критерии качества детектора

- **Good detection:** оптимальный детектор должен минимизировать ошибки 1 и 2го родов (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

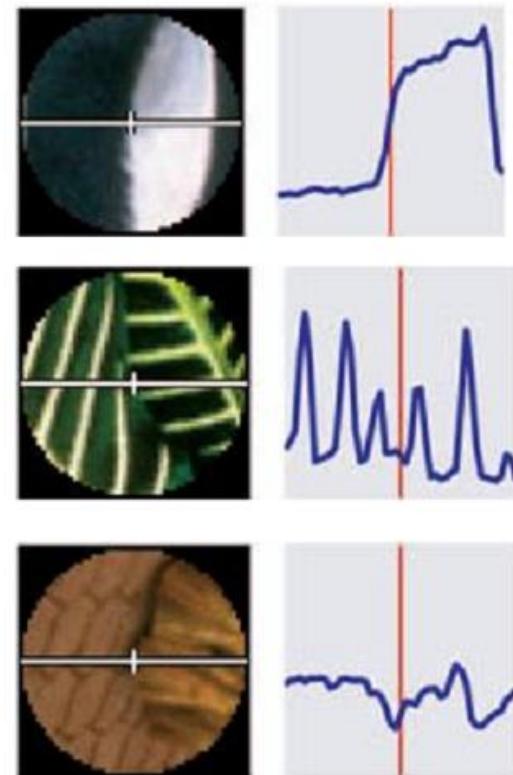
Source: L. Fei-Fei

# Ограничения детектора



# Probability boundary (Pb) -детектор

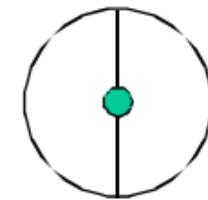
- Недостаточно находить градиент изображения
- Идея – обучить классификатор граница / не граница
- Будем использовать разные признаки:
  - Яркость
  - Цвет
  - Текстура
- «Probability boundary» (Pb) детектор



D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI 2004.

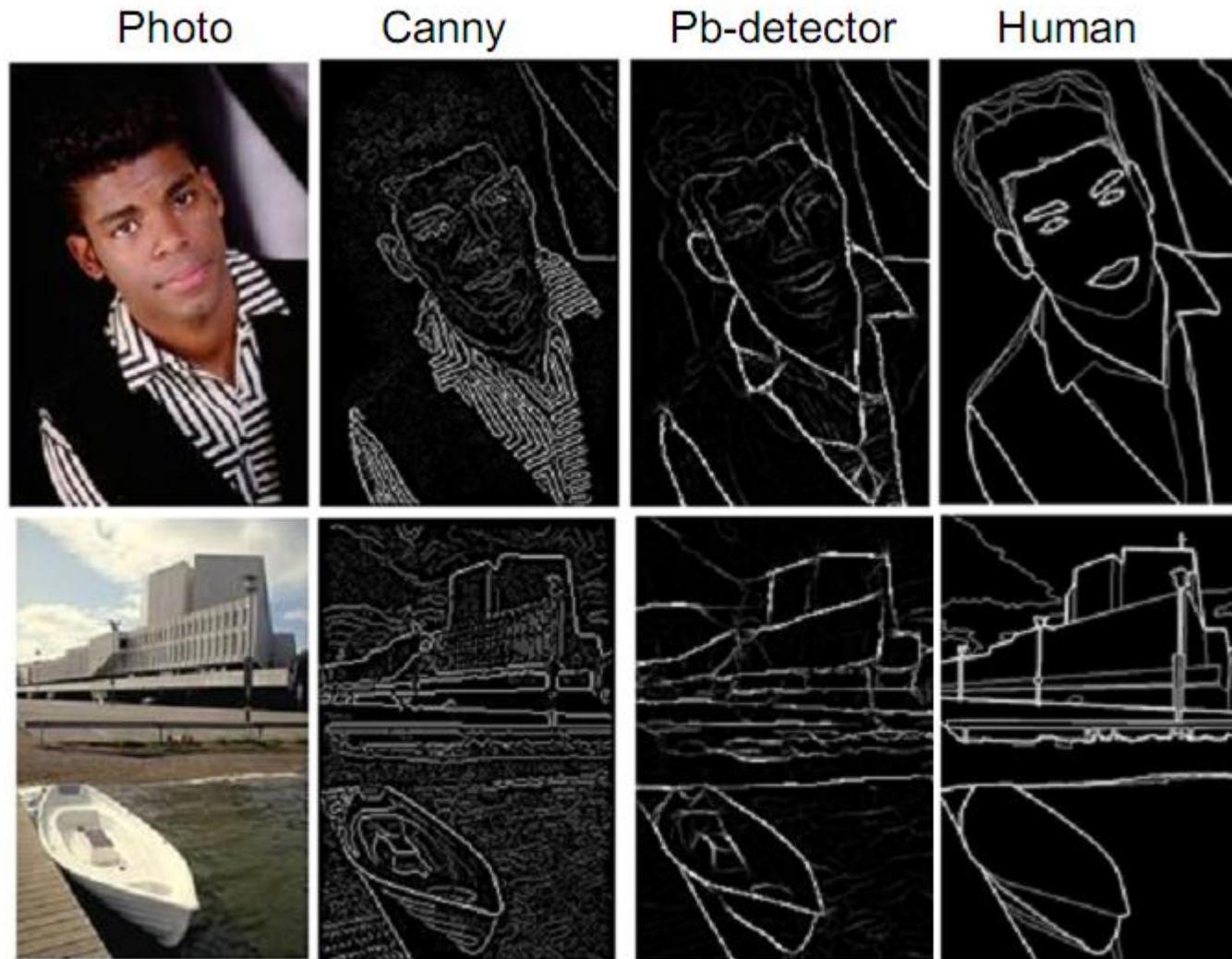
# Схема Pb-детектора

- Берём круг с центром в исследуемом пикселе
- Выбираем ориентацию края
- Считаем градиент между признаками в левой и правой половине круга
- Обучаем классификатор края
- Требуется размеченная коллекция изображений



D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI 2004.

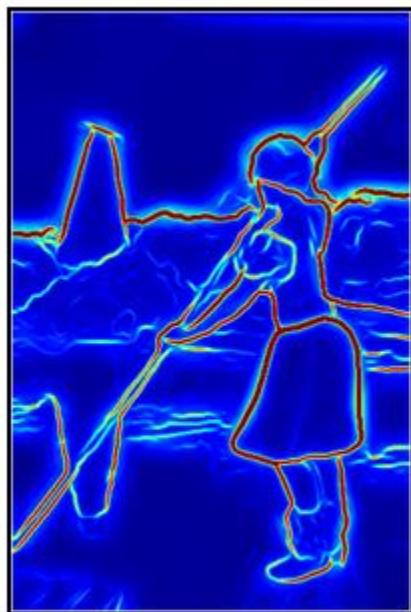
# Результат Pb-детектора



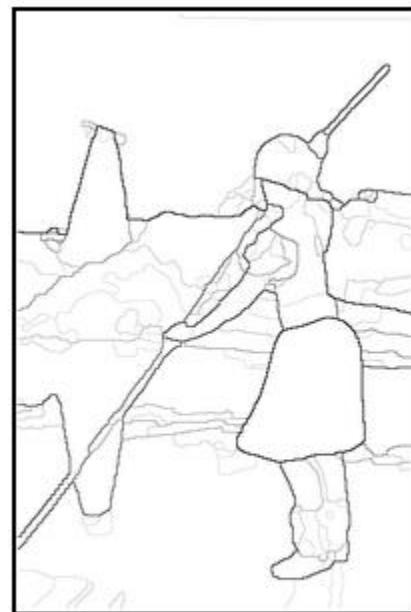
# Автоматическая сегментация



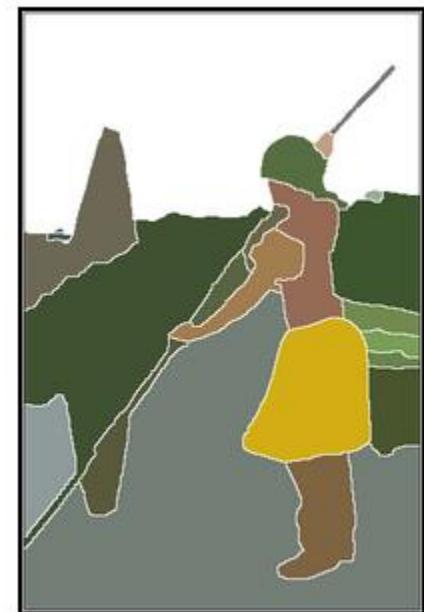
Original Image



globalPb



Ultrametric Contour Map

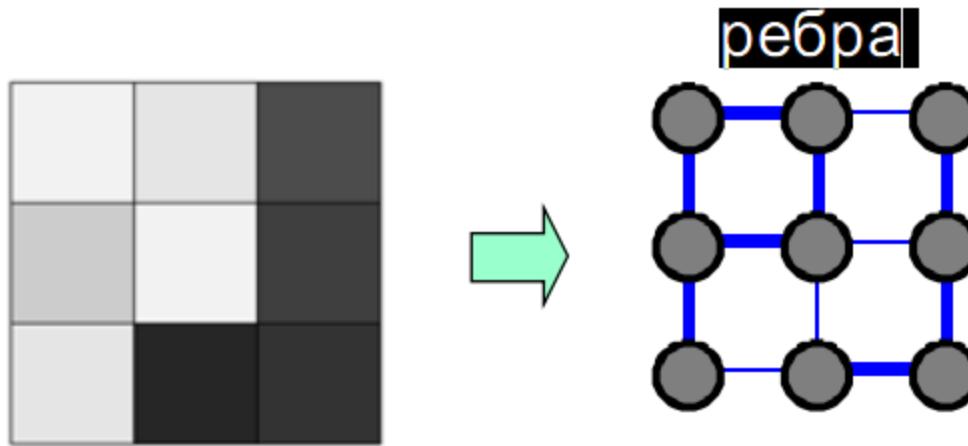


Automatic Segmentation

# Семейства методов

- Основанные на формировании однородных областей
  - без пространственных связей
  - с учетом пространственных связей
- Суперпиксели
- Основанные на поиске краев (края и области)
  - Canny
  - Pb-детектор краёв
- **Методы на графах**
  - Normalized cut
  - «Эффективный метод» Felzenszwalb & Huttenlocher
- Энергетические методы
  - Snakes
  - Методы уровня

# Представление в виде графа

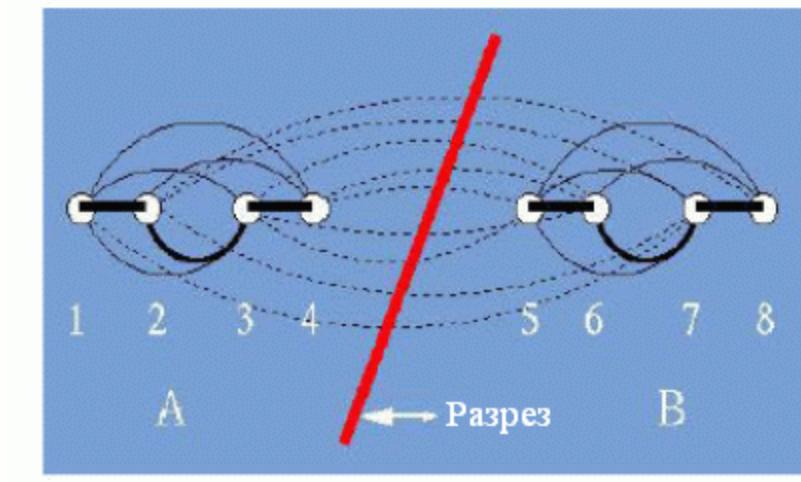


Изображение превращается во взвешенный неориентированный граф  $G = (V, E)$

- Вершины графа  $V$  – пиксели изображения
- Ребра  $E$  – связи между соседними пикселями
- Вес ребер пропорционален «похожести» пикселей

# Сегментация с помощью разрезов графа

- Создать граф
- Разрезать граф
- Каждую связную компоненту после разреза рассматривать как отдельную область

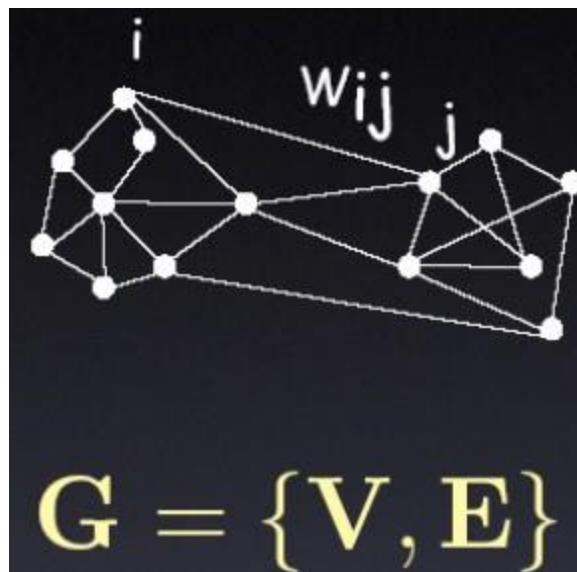


# Разрез графа

- $G=(V,E)$ 
  - Непересекающиеся подмножества вершин  $A$  и  $B$  из  $V$
  - Удаляем все ребра, связывающие  $A$  и  $B$

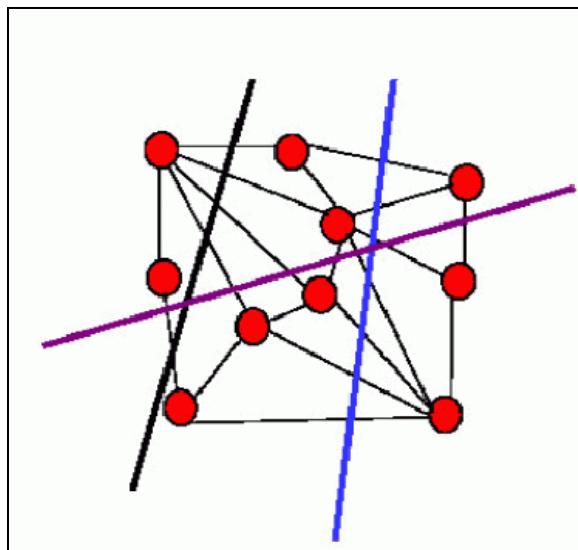
$$Cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

- $Cut(A, B)$  – мера «силы связности» множеств  $A$  и  $B$



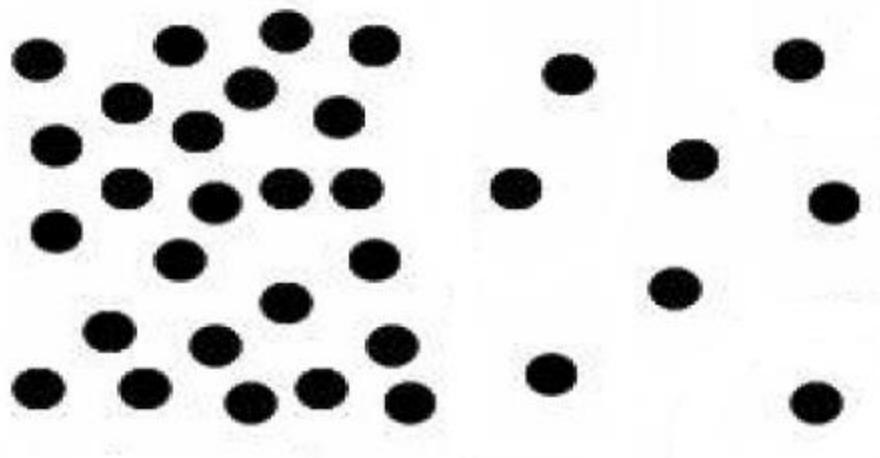
# Минимальный разрез графа

- Если множества А и В не заданы заранее – разрезать граф можно по-разному:
  - **Минимальный разрез** – разрез, превращающий граф в несвязный, с минимальной суммой весов удаленных ребер



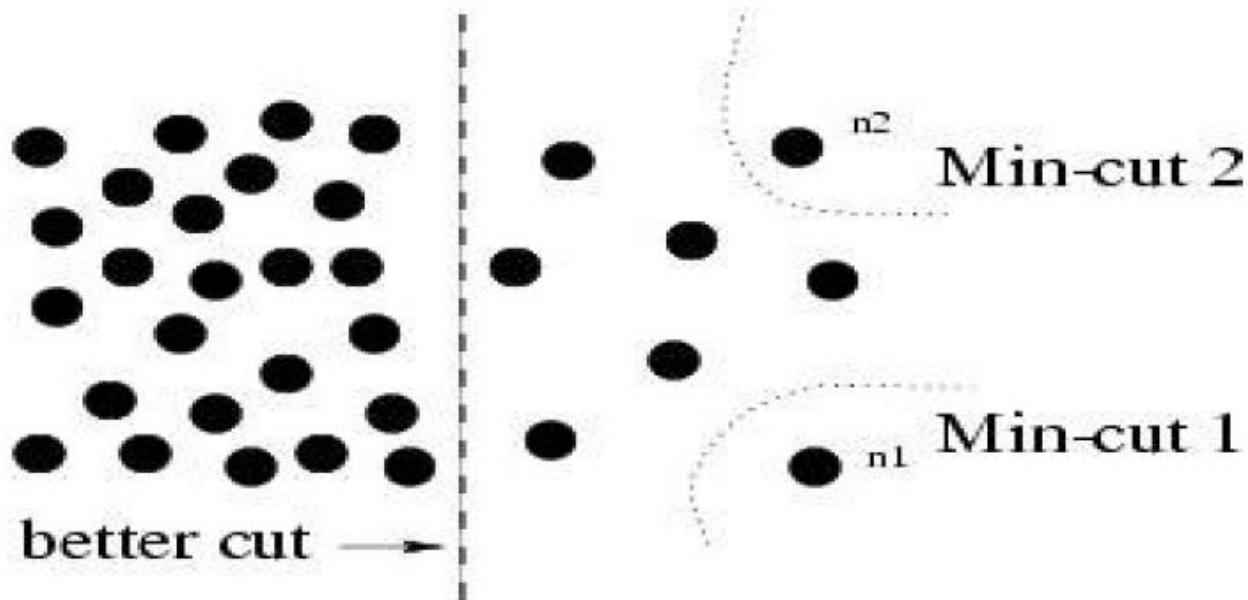
$$Cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

# Как бы разбили вы?



На данном рисунке вес ребер графа показан расстоянием между вершинами

# Минимальный разрез хорош не всегда



На данном рисунке вес ребер графа показан расстоянием между вершинами

# Нормализованный разрез графа (Normalized cut)

Другая мера разреза – измеряет «похожесть» двух групп вершин, нормированную на «объем», занимаемый ими в графе

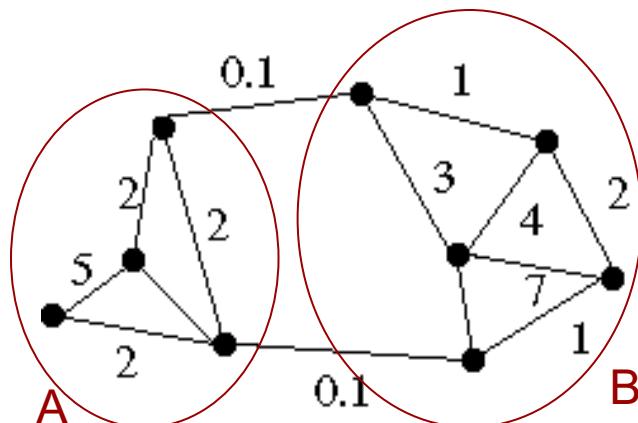
$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

$$Cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

# Минимальный нормализованный разрез

- **Минимальный нормализованный разрез** – разрез, превращающий граф в несвязный, с минимальной величиной  $NCut$
- *Как его найти?*



$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

$$Cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

# Алгоритм сегментации с помощью normalized cuts (по Ши)

1. Задать граф на изображении

Вес ребра (по Ши)

$$w(i, j) = e^{\frac{-\|F(i) - F(j)\|_2}{\sigma_I}} * \begin{cases} e^{\frac{-\|X(i) - X(j)\|_2}{\sigma_X}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases}$$

–  $F(i) = I(i)$

–  $F(i) = [v, v \cdot s \cdot \sin(h), v \cdot s \cdot \cos(h)](i)$

where  $h$ ,  $s$ , and  $v$  are the HSV values

# Алгоритм сегментации с помощью normalized cuts (по Ши)

2. Рассчитать матрицы W и D

$$D = \begin{bmatrix} \sum_j w(1, j) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_j w(N, j) \end{bmatrix}, \quad W = \begin{bmatrix} w(1,1) & \cdots & w(1,N) \\ \vdots & \ddots & \vdots \\ w(N,1) & \cdots & w(N,N) \end{bmatrix}$$

Можно вывести что:

$$\text{MinNcut}(G) = \min_y \frac{y^t(D - W)y}{y^t D y}$$



NP-трудная задача

Если разрешить  $y \in \mathbb{R}$  задача сводится к задаче на собственные значения:

$$(D - W)y = \lambda Dy$$

# Алгоритм сегментации с помощью normalized cuts

3. Решить задачу  $(D-W)y = \lambda Dy$ , найти вектора с наименьшими собственными значениями
4. По вектору со вторым наименьшим с.з. разрезать граф на две части
5. Рекурсивно разбить получившиеся области, если требуется

Jianbo Shi and Jitendra Malik "Normalized Cuts and Image Segmentation",  
IEEE PAMI, 2000

[https://repository.upenn.edu/cgi/viewcontent.cgi?article=1101&context=cis\\_papers](https://repository.upenn.edu/cgi/viewcontent.cgi?article=1101&context=cis_papers)

<https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf>

# Результат сегментации Ши



(a)



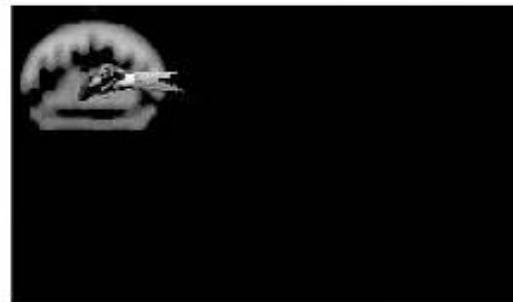
(b)



(c)



(d)



(e)



(f)

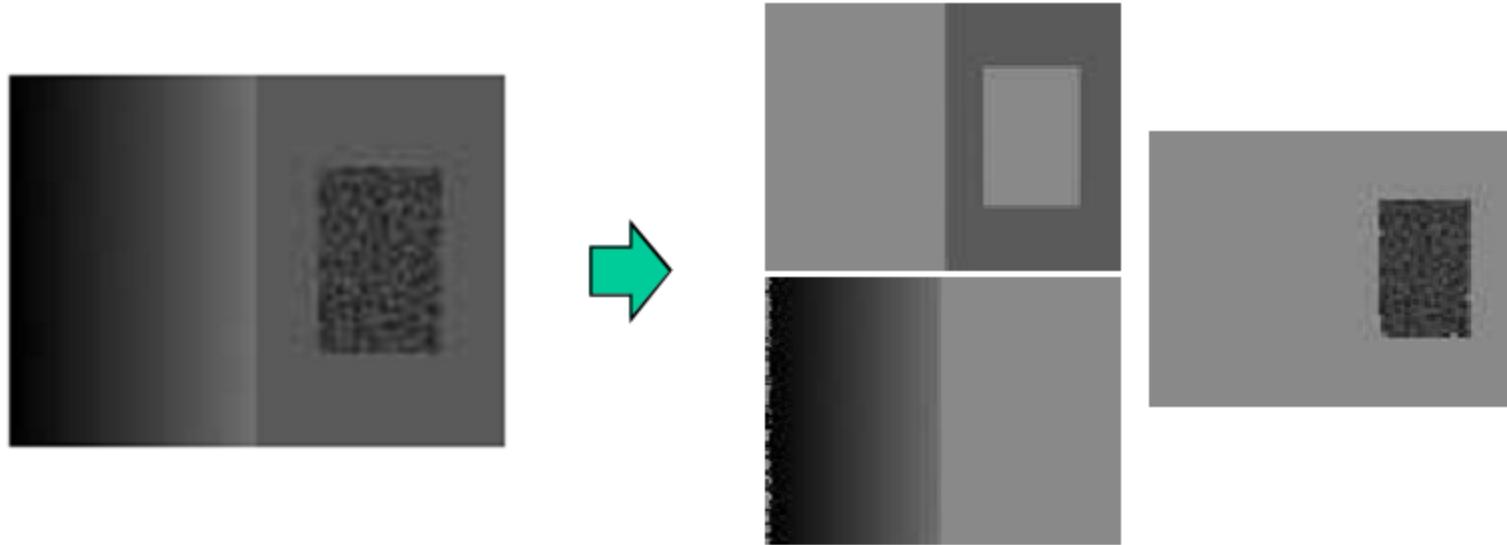
# Результат сегментации Ши



# Анализ алгоритма

- Метод требует хранения матрицы размером  $n^*n$ , где  $n$  – число точек изображения, и потому в исходном виде неприменим к большим изображениям.
- Для данного метода предложены модификации [24, 25], позволяющие сократить сложность алгоритма и требования по памяти за счет аппроксимации матрицы расстояний. Такой подход дает выигрыш в скорости работы в 10-20 раз по сравнению с исходным методом.

# «Эффективный метод»



- Идея: разница в интенсивности вдоль границы между областями должна быть существенной по сравнению с колебаниями интенсивности внутри одной из областей

P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. IJCV, 59(2):167–181, 2004.

# Формализация

- «Внутренняя разница» в компоненте С:

$$Int(C) = \max_{e \in MST(C, E)} w(e) \quad \text{где } w(e) \text{ - вес ребра (мера различия двух пикселов)}$$

- Разница между областями:

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j))$$

- Предикат присутствия границы между областями:

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

где  $MInt$  – минимальные колебания интенсивности по областям

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$$

и регуляризационный параметр  $\tau(C) = k/|C|$

где  $|C|$  - размер области,  $k$  – коэффициент регуляризации

# Алгоритм

- **Схема «слияния регионов», но с учётом выбранных функций слияния:**
  - Сортируем все ребра по возрастанию веса
  - Инициализируем сегментацию максимальным разбиением (у каждого пикселя своя компонента)
  - Проходим по списку всех ребер  $(i,j)$ 
    - Пусть  $C_i, C_j$  – компоненты, которым принадлежат вершины  $i, j$
    - Тогда если не выполняется  $D(C_i, C_j)$ , тогда объединяем  $C_i$  и  $C_j$

Сложность метода  $O(N \log N)$

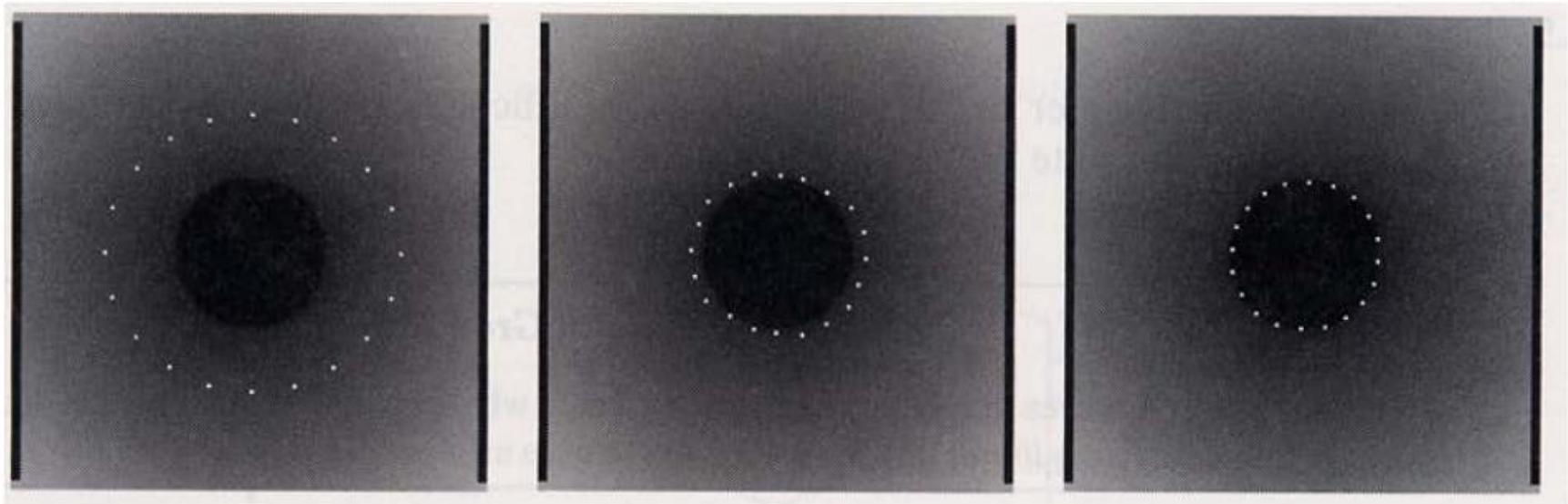
# Результат



# Семейства методов

- Основанные на формировании однородных областей
  - без пространственных связей
  - с учетом пространственных связей
- Суперпиксели
- Основанные на поиске краев (края и области)
  - Canny
  - Pb-детектор краёв
- Методы на графах
  - Normalized cut
  - «Эффективный метод» Felzenszwalb & Huttenlocher
- **Энергетические методы**
  - Snakes (активные контуры)
  - Методы уровня

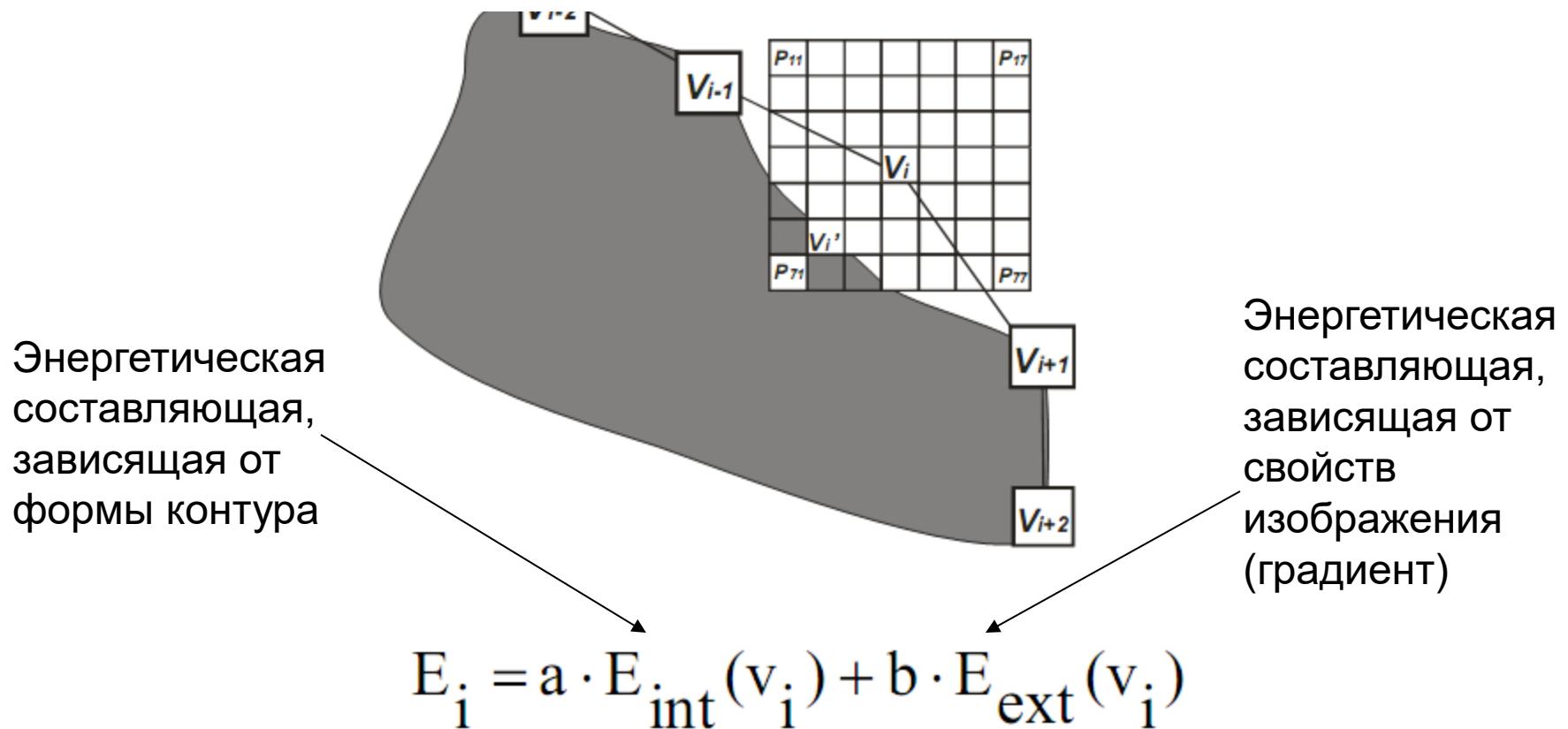
# Активный контур (Snakes)



M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. IJCV, 1988

# Активный контур

$$V = \{v_1, \dots, v_n\}, \text{ где } v_i = (x_i, y_i), i = \{1, \dots, n\}$$



# Внутренняя энергия

Внутренняя энергия = сглаживающая + распирающая

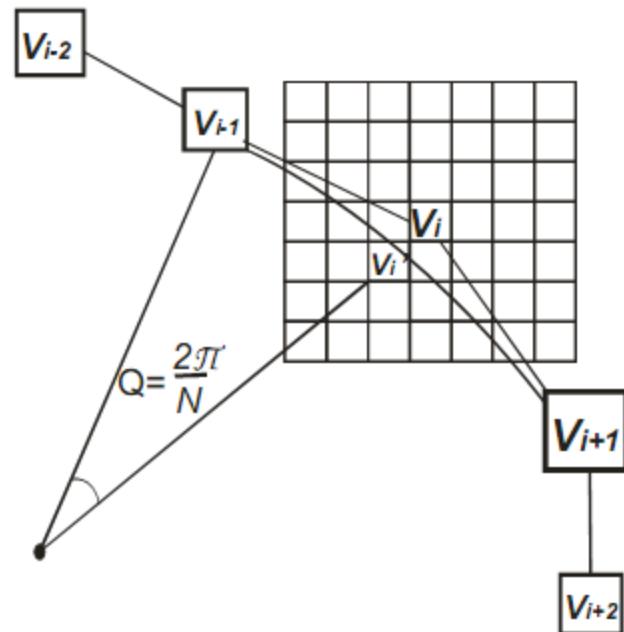
$$a \cdot E_{\text{int}}(v_i) = c \cdot E_{\text{con}}(v_i) + d \cdot E_{\text{bal}}(v_i)$$

# Сглаживающая составляющая

$$e_{jk}(v_i) = \frac{1}{l(V)} \left\| p_{jk}(v_i) - \gamma(v_{i-1} + v_{i+1}) \right\|^2$$

$$\gamma = \frac{1}{2 \cos\left(\frac{2\pi}{n}\right)}$$

$$l(V) = \frac{1}{n} \sum_{i=1}^n \|v_{i+1} - v_i\|^2$$

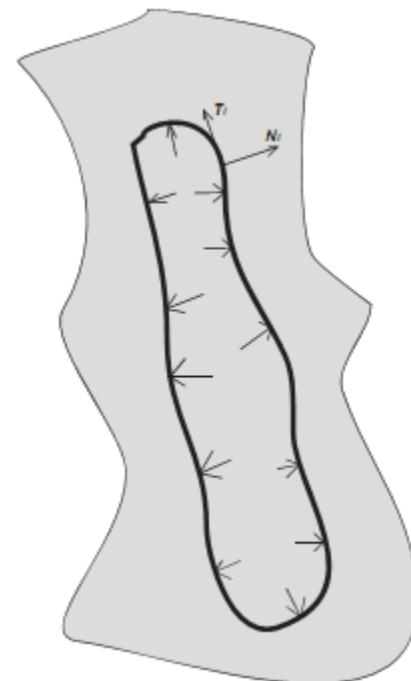


# Распирающая составляющая

$$e_{jk}(v_i) = n_i \cdot (v_i - p_{jk}(v_i))$$

$n_i$  – перпендикуляр к  $t_i$

$$t_i = \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{\|\mathbf{v}_i - \mathbf{v}_{i-1}\|} + \frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{\|\mathbf{v}_{i+1} - \mathbf{v}_i\|}$$



# Внешняя энергия

Внешняя энергия = энергия изображения + энергия градиента

$$\mathbf{b} \cdot E_{\text{ext}}(v_i) = m \cdot E_{\text{mag}}(v_i) + g \cdot E_{\text{grad}}(v_i)$$

Энергия изображения

$$e_{jk} = I(p_{jk}(v_i))$$

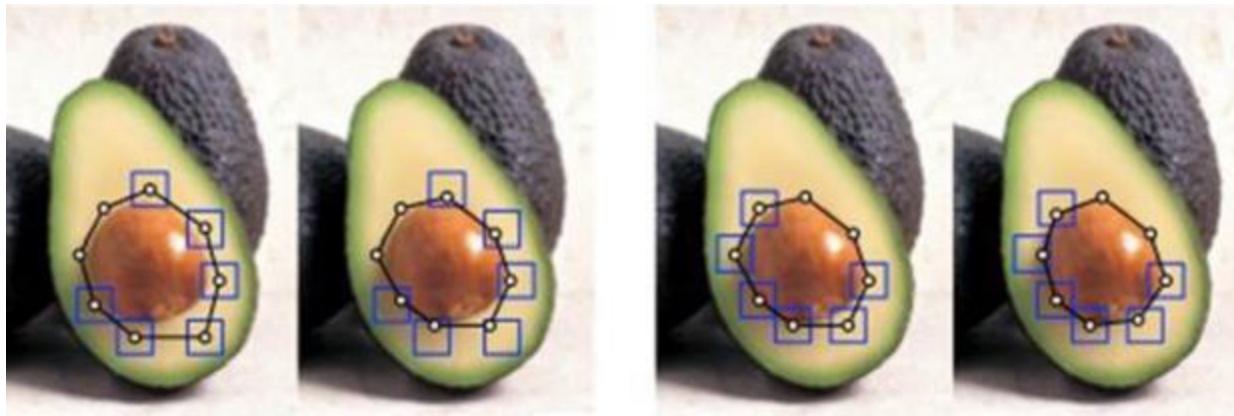
Энергия градиента

$$e_{jk}(v_i) = -|\nabla I(p_{jk}(v_i))|$$

# Дополнительные энергии

- Энергия схожести по яркости
- Энергия, соответствующая расстоянию от центра масс фигуры

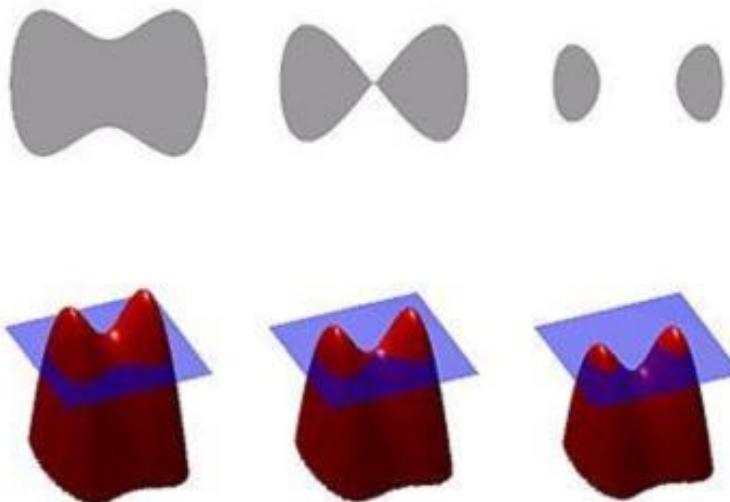
# Вариант алгоритма Snakes



# Метод фиксации уровня (Level set method)

- введён американскими математиками Стэнли Ошером и Джеймсом Сетьяном в 1980-е годы.
- Он стал популярным во многих дисциплинах, таких как
  - компьютерная графика,
  - обработка изображений,
  - вычислительная геометрия,
  - оптимизация,
  - вычислительная гидродинамика и
  - вычислительная биофизика

# Level Sets (Линии уровня)



Визуализация представления контура с помощью функции вложения

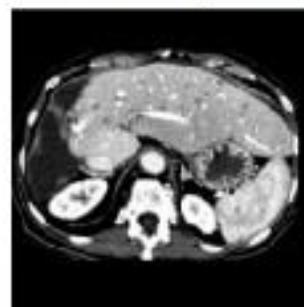
- Вместо явного представления контура зададим функцию вложения (embedding function)  $f$ 
  - $f > 0$  в области,  $f < 0$  вне области
  - Можем переформулировать задачу обновления контура как задачу обновления функции вложения

- Функция вложения позволяет моделировать изменения топологии области
- Работа с ней сводится к решению систем дифференциальных уравнений

# Level set method

- Начальная маска. Это объект любой формы, находящийся внутри интересующего нас объекта. Чаще всего это прямоугольник, потому что его довольно просто задать.
- После определенного количества итераций функция модификации уровня меняется таким образом, чтобы ее пересечение с нулевой плоскостью давало интересующую нас кривую

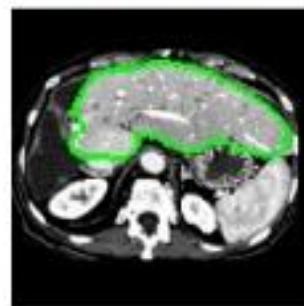
Входное изображение



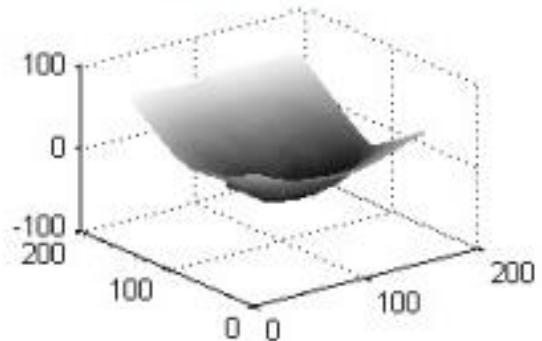
Маска



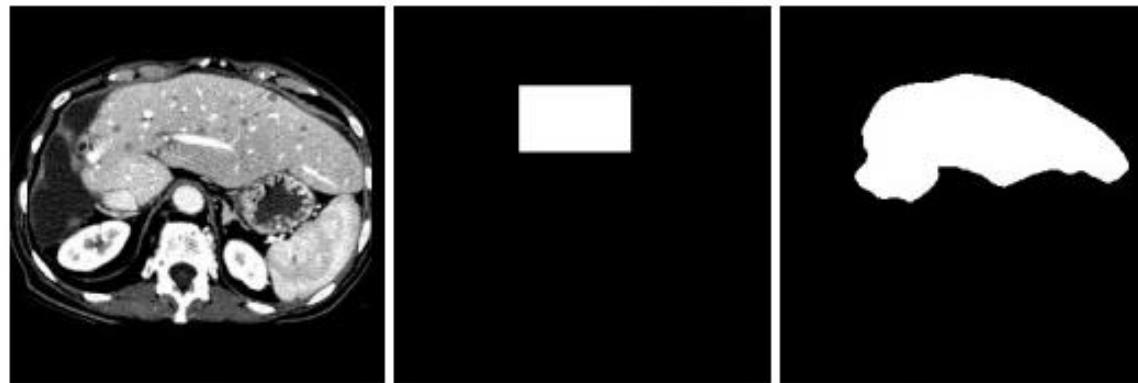
500 итераций



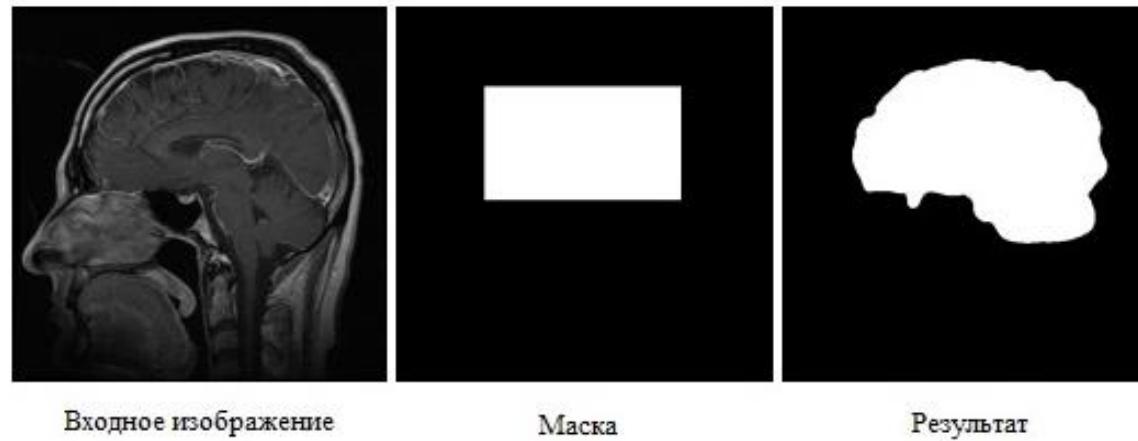
Функция фиксации уровня



Для медицинских изображений это работает вот так:



Сегментация печени с параметрами  $T = 180, \epsilon = 45, \alpha = 0.003$

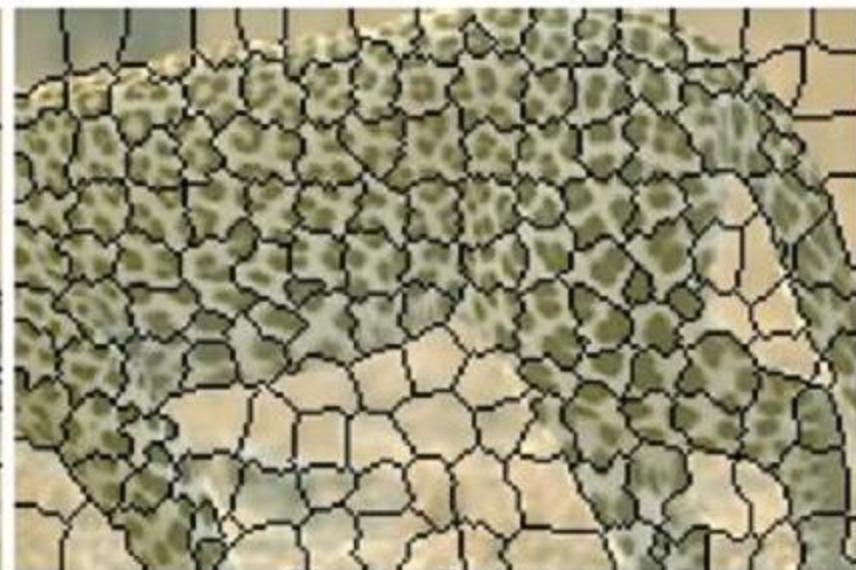


Сегментация мозга с параметрами  $T = 45, \epsilon = 30, \alpha = 0.003$

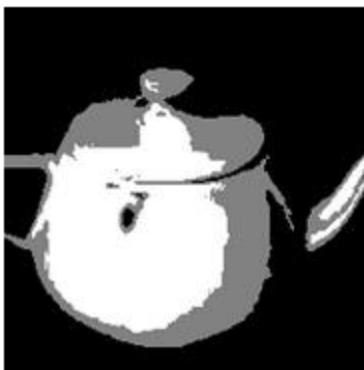
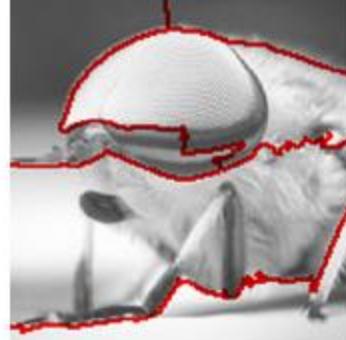
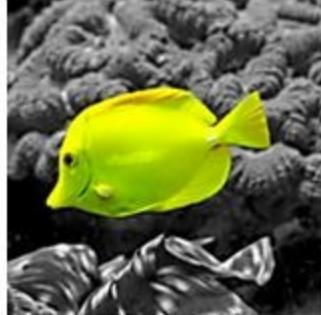
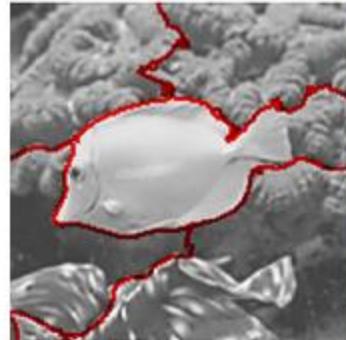
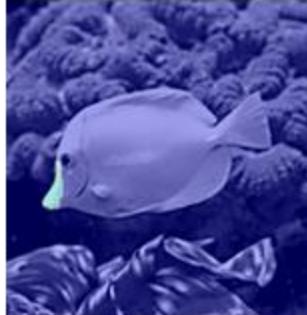
# Сравнение

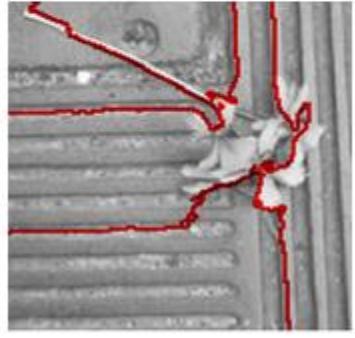


Ncuts

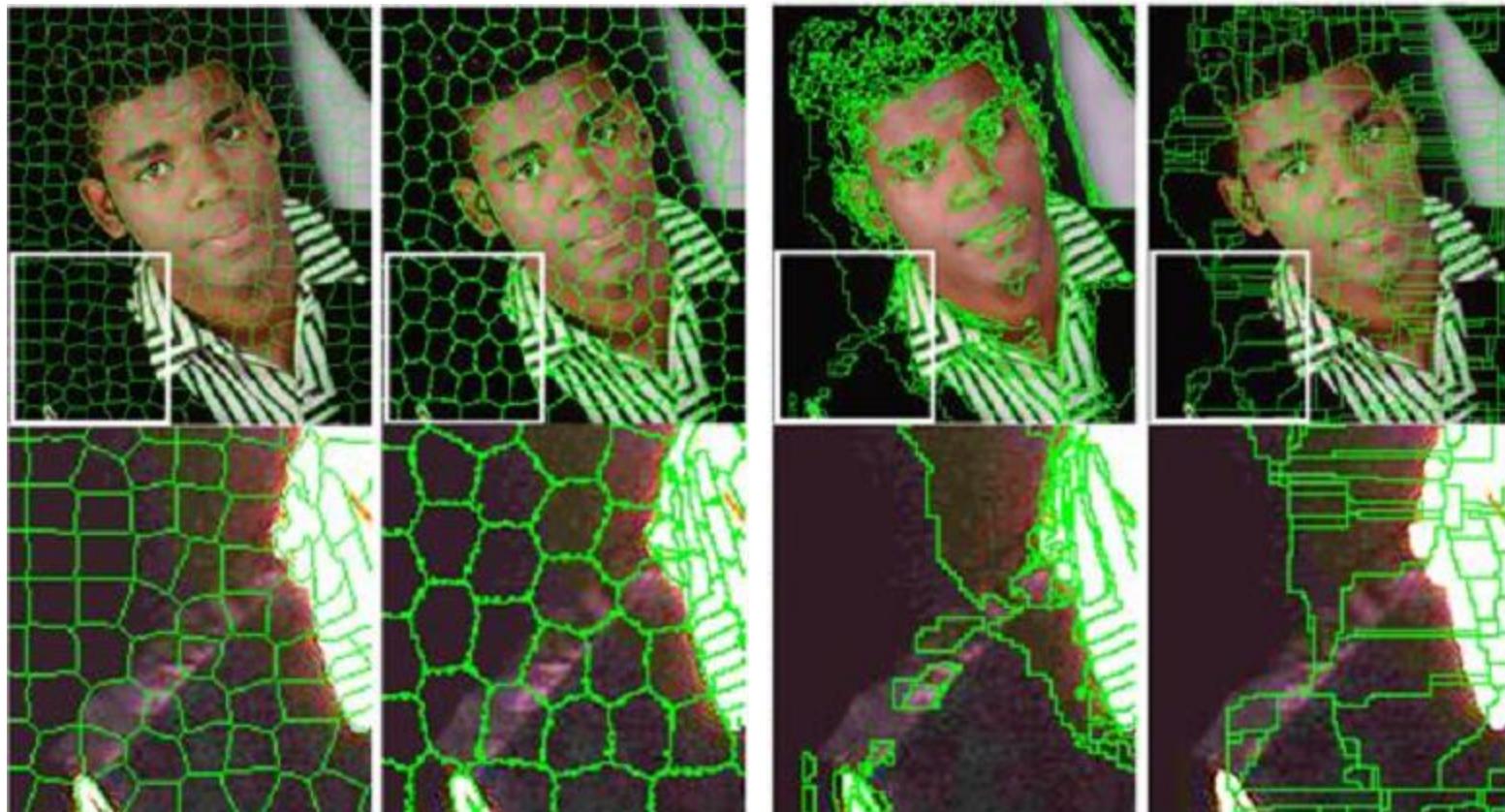


TurboPixels

Исходное изображение	Метод выращивания регионов	Метод нормальных разрезов	Метод водораздела
			
			
			

Исходное изображение	Метод выращивания регионов	Метод нормальных разрезов	Метод водораздела
			
			

# Сравнение



TurboPixel

NCuts

Mean-shift

Watershed

# Резюме

- Задача сегментации – «разбора изображения» в общем случае является целью распознавания изображений
- Сегментация изображения на области по набору признаков – эффективная предобработка для решения других основных задач
- Понижение размерности задач (работа с областями, а не отдельными пикселями)
- Хорошая сегментация должна учитывать несколько признаков в совокупности
- Используются в зависимости от задачи

# УРОВНИ «ПОНИМАНИЯ» ИЗОБРАЖЕНИЯ АЛГОРИТМОМ:

## 1. Семантическая сегментация (Semantic Segmentation)

Что делает: Присваивает каждому пикселью изображения класс (например: «человек», «автомобиль», «дерево», «дорога»).

Главная особенность: Не различает отдельные экземпляры объектов. Два автомобиля на изображении будут просто помечены как «автомобиль», без указания, где заканчивается один и начинается другой.

Пример архитектур: U-Net, FCN (Fully Convolutional Network), DeepLab, SegNet.

## 2. Сегментация экземпляров (Instance Segmentation)

Что делает: Определяет границы каждого отдельного объекта на изображении. Каждый экземпляр объекта получает свою уникальную метку.

Главная особенность: Различает объекты одного класса. На изображении с тремя кошками алгоритм точно выделит каждую кошку отдельно.

Пример архитектур: Mask R-CNN (золотой стандарт), YOLACT, SOLO.

# Уровни «понимания» изображения алгоритмом:

## 3. Панорамная сегментация (Panoptic Segmentation)

Что делает: Объединяет в себе лучшие черты семантической и инстанс-сегментации.

Она присваивает каждому пикселю две метки: 1) семантический класс и 2) идентификатор экземпляра.

Главная особенность: Сегментирует всё на изображении: и «вещи» (счётные объекты, как люди, машины), и «stuff» (бесформенные области, как небо, трава, дорога).

**Пример архитектур: Модификации Mask R-CNN и Panoptic FPN.**

# Ключевые подходы и архитектуры на основе сверточных нейронных сетей (CNN)

## Encoder-Decoder Архитектура (например, U-Net):

Encoder («кодировщик»): Похож на классический классификатор (например, ResNet).

Он извлекает features (признаки) из изображения, но при этом постепенно уменьшает пространственное разрешение (width, height), увеличивая «глубину» (количество каналов).

Это позволяет уловить контекст и смысл сцены.

Decoder («декодировщик»): Постепенно увеличивает разрешение feature maps, чтобы восстановить исходный размер изображения и получить плотное пиксельное предсказание.

Skip-connections (пропускные соединения) напрямую связывают encoder и decoder, чтобы передавать информацию о мелких деталях и границах, которые были потеряны при сжатии.

# Ключевые подходы и архитектуры на основе сверточных нейронных сетей (CNN)

## Архитектуры на основе DETR (Detection Transformer):

Это более современный подход, который использует трансформеры (как в NLP) для задач компьютерного зрения.

Модели типа Mask2Former показывают результаты сразу для всех трех типов сегментации, используя архитектуру «transformer decoder».

## Беспроводные методы (Zero-Shot Segmentation):

Модели вроде Segment Anything Model (SAM) могут выполнять автоматическую сегментацию, генерируя маски для всех объектов на изображении без явного указания, что это за классы. Это мощный инструмент для «просто выделить всё, что видно».

# Для чего это используется? Применения

Автономные транспортные средства: Понимание сцены вокруг автомобиля — где дорога, пешеходы, другие машины, знаки.

Медицинская диагностика: Автоматическое выделение опухолей на МРТ/КТ, сосудистых структур, анализ клеток в микроскопии.

Дистанционное зондирование: Анализ спутниковых снимков для классификации типов землепользования (лес, город, водоем), мониторинг сельского хозяйства.

Робототехника: Помощь роботам в навигации и взаимодействии с объектами.

Фото- и видеоредактирование: Автоматическое размытие фона (portrait mode), стилизация, ретушь.

Дополненная реальность: Точное наложение виртуальных объектов на реальные.

# Плюсы автоматической сегментации

## **Скорость:**

Обрабатывает изображения мгновенно, не требуя времени пользователя.

## **Масштабируемость:**

Идеально для обработки больших объемов данных (например, видеопоток или тысячи медицинских снимков).

## **Консистентность:**

Алгоритм всегда работает одинаково, без человеческой усталости или субъективности.

# Минусы автоматической сегментации

## **Неточность:**

Может ошибаться на сложных границах, с мелкими объектами или в нестандартных условиях освещения/ракурса.

## **Отсутствие контроля:**

Если алгоритм ошибся, вы не можете его легко поправить «на лету». Нужно либо дообучать модель, либо вручную править результат.

## **Зависимость от данных:**

Качество сильно зависит от данных, на которых модель была обучена. Модель, обученная на городских сценах, будет плохо работать на медицинских снимках.