

# Сегментация изображений

Владимир Вежневец, Антон Конушин  
Александр Вежневец

Курс – «Введение в компьютерное зрение»  
МГУ ВМК, Graphics & Media Lab,  
Осень 2006

# На прошлой лекции...

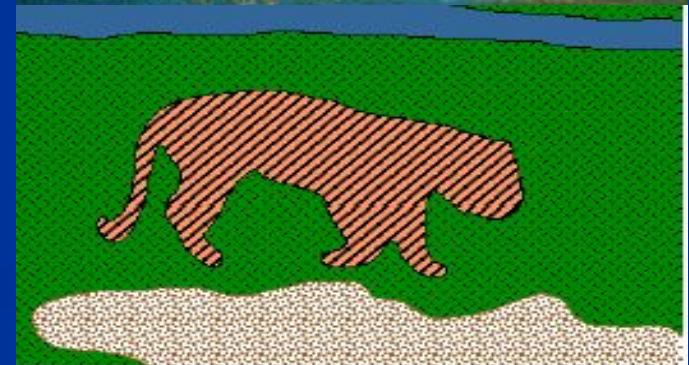


- На прошлой лекции научились работать со структурами «с нулевой площадью» на изображении (с контурами)
  - Выделять
  - Анализировать (отличать)
- На этой лекции – перейдем к «объектам с площадью». Будем:
  - Выделять
  - Анализировать (отличать)



# Что такое сегментация?

- Анализ высокого уровня:
  - отделение находящихся на изображении объектов от фона (и друг от друга)
- Анализ низкого уровня:
  - разбиение на области «похожих» между собой пикселей



# Автоматика и интерактивность



- Подразделяем
  - Автоматическая
    - Сегментация производимая без взаимодействия с пользователем
      - Картинка на входе, регионы на выходе
  - Интерактивная
    - Сегментация, управляемая пользователем, допускающая и/или требующая ввода дополнительной информации
      - Пример – «волшебная палочка» в Photoshop



# Применение сегментации

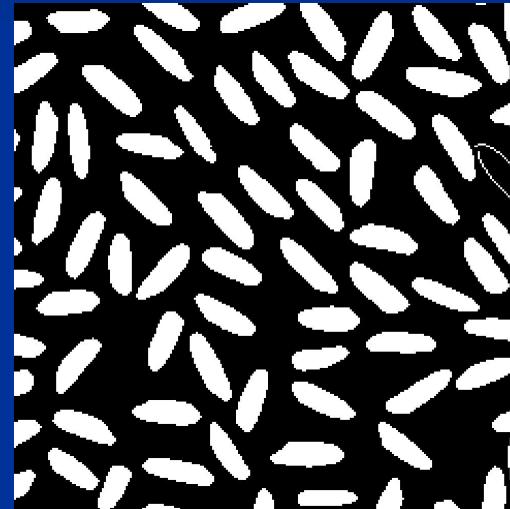
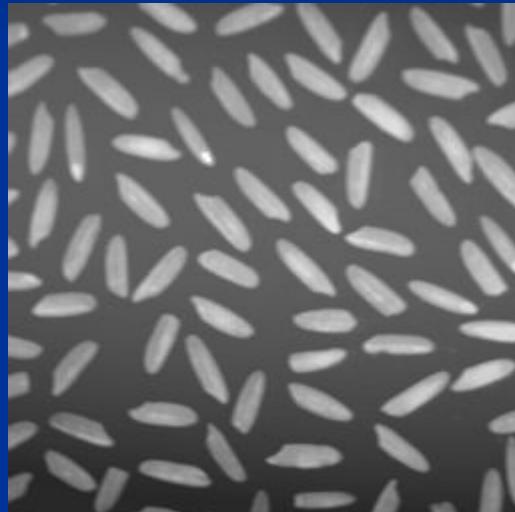
- Фото(видео)монтаж, композиция



# Применение сегментации



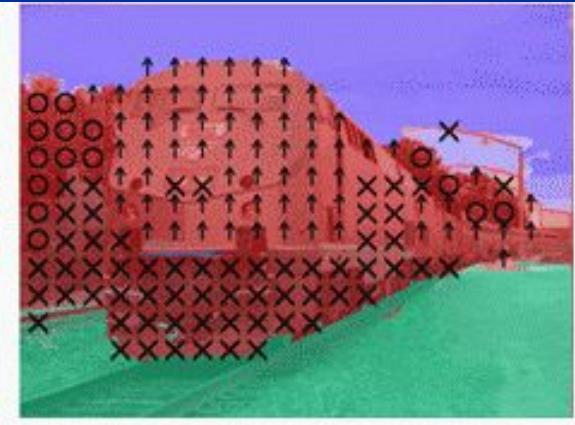
- Измерение параметров объектов





# Применение сегментации

- Предобработка перед высокоуровневым анализом



# Определение сегментации 1



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

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

- $I = \bigcup_{i=1..N} S_i$
- $\forall i, j = 1, N : i \neq j \ S_i \cap S_j = \emptyset$
- $\forall i = 1, N, P(S_i) = \text{истина}$
- $\forall i, j = 1, N : i \neq j \ P(S_i \cap S_j) = \text{ложь}$

# Рассмотрим семейства методов:



- Основанные на поиске краев
- Основанные на формировании однородных областей
- Метод водораздела / tobogganing
- Методы из теории графов

# Автоматическая сегментация



- Как можно сформировать однородные области?
  - Отталкиваясь от неоднородности на границах
    - Пример – ищем резкие переходы яркости, берем их как границы областей
  - Отталкиваясь от однородности внутри областей
    - Пример – объединяем в одну область пиксели, близкие по яркости



# Однородность

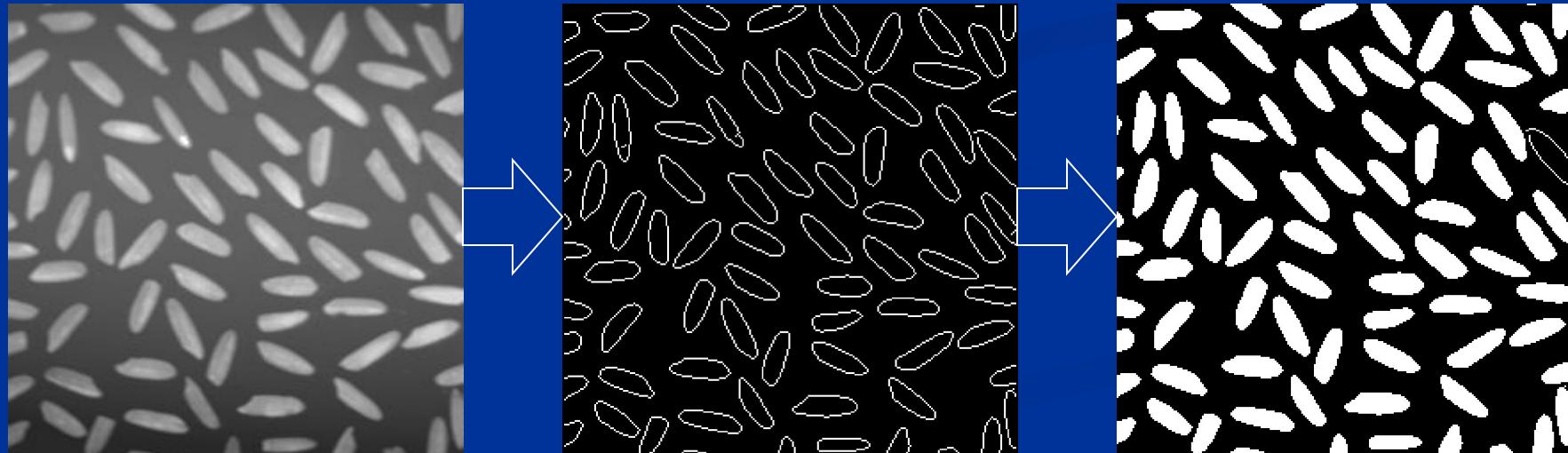
## ■ Варианты однородности:

- По яркости
- По цвету
- По близости на изображении
- По текстуре
- По глубине
  - (Если есть 3D информация)

# Сегментация через поиск неоднородностей



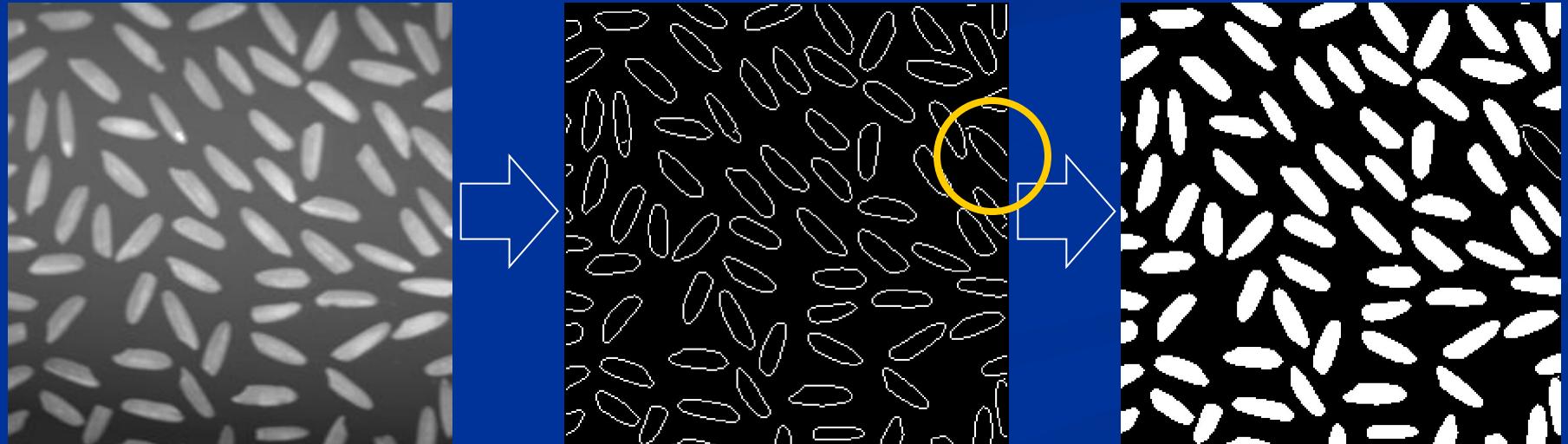
- Наиболее простой и чаще всего используемый вариант:
  - Поиск неоднородностей яркости через выделение краев





# Алгоритм

1. Найдём все контура на изображении алгоритмом Canny;
2. Найдем все замкнутые контура;
3. «Внутренности» замкнутых контуров являются искомыми однородными областями;



# Сегментация через поиск однородных областей



- План
  - Сегментация без учета пространственных связей
    - Пороговая фильтрация
    - Кластеризация по цвету
  - Сегментация с учетом пространственных связей
    - Разрастание областей (region growing)
    - Слияние/разделение областей (region merging/splitting)

# Пороговая фильтрация



- Разделение пикселей на n классов по их яркости
  - Чаще всего используется 2 класса (бинаризация)





# Гистограммы

Гистограмма (одноканального изображения) – график распределения яркостей пикселей:

- На горизонтальной оси - шкала яркостей от черного до белого
- На вертикальной оси - число пикселей заданной яркости



255



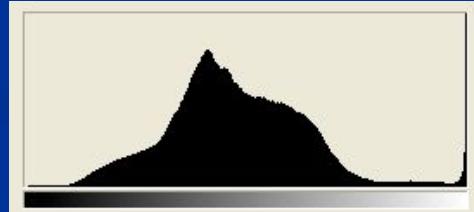
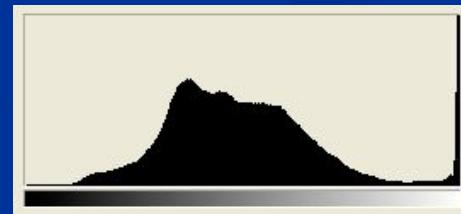
0

255



# Гистограммы

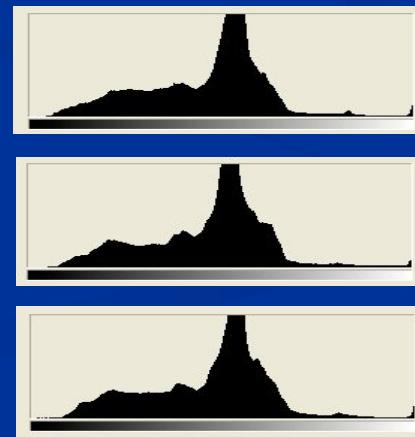
- Свойства:
  - Рассчитываются глобально для всего изображения
  - Пространственная информация (расположение пикселей различной яркости) полностью игнорируется
- Это можно использовать для сравнения изображений:





# Гистограммы

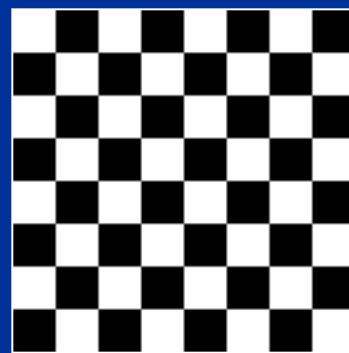
- Свойства:
  - Рассчитываются глобально для всего изображения
  - Пространственная информация (расположение пикселей различной яркости) полностью игнорируется
- Это можно использовать для сравнения изображений:



# Гистограммы



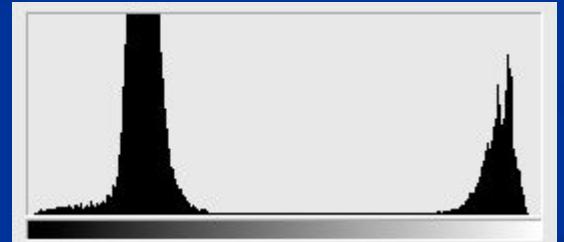
- Свойства:
  - Рассчитываются глобально для всего изображения
  - Пространственная информация (расположение пикселей различной яркости) полностью игнорируется
- Однако при анализе сложных сцен это может мешать
  - Сильно различные «с виду» сцены могут иметь очень похожие гистограммы





# Пороговая фильтрация

- Яркий объект на темном фоне
  - Выбрать величину  $T$  разделяющую яркость объекта и фона
  - Каждый пиксель  $(x,y)$  яркость которого  $I(x,y) > T$  принадлежит объекту



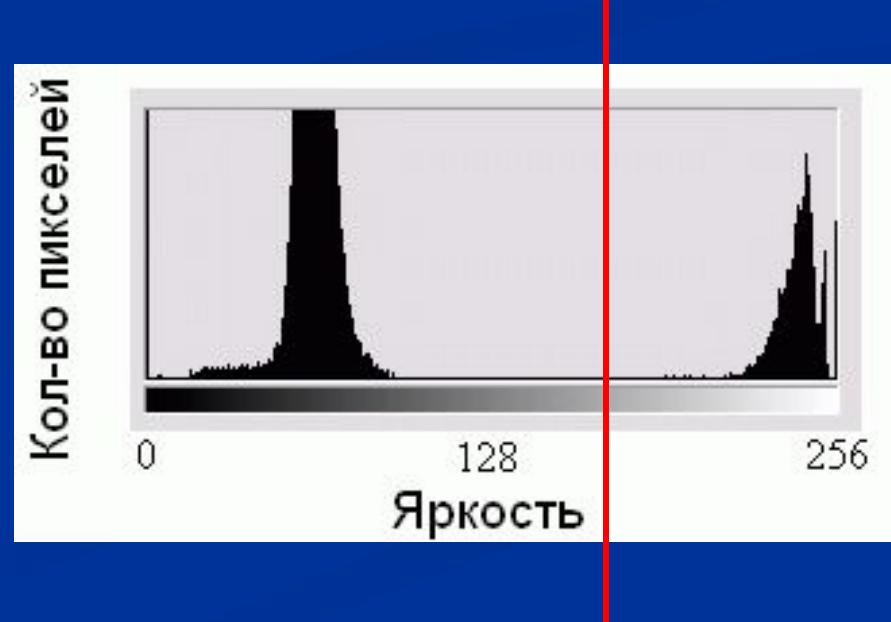
0

255

# Как определить величину Т?



- В каждом конкретном случае хотим уметь рассчитать правильный порог
  - Вариант решения – анализ гистограммы изображения



# Автоопределение величины Т

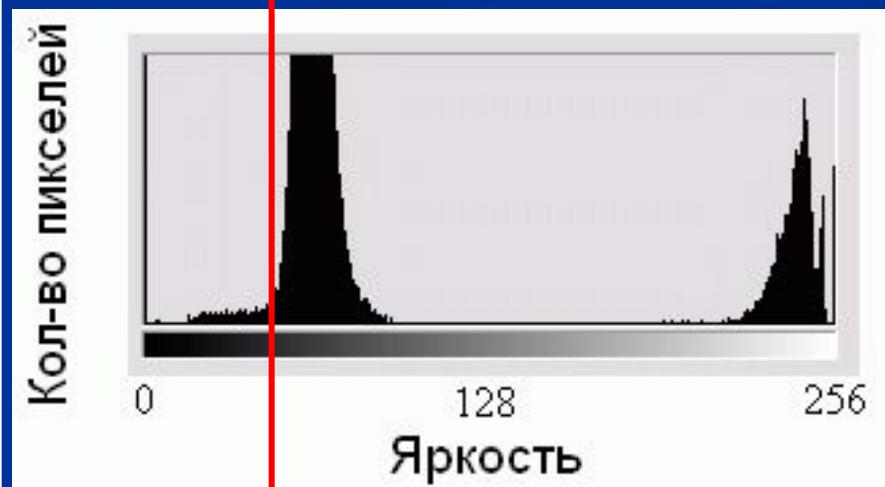
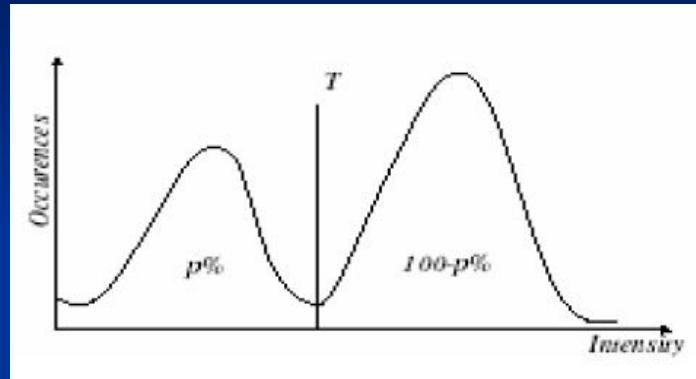


- Можно использовать следующее:
  1. Предположение о яркости объектов
  2. Размеры объектов
  3. Площадь изображения занятого объектом
  4. Количество различных типов объектов
- Вопрос - как?

# Автоопределение величины $T$



- Метод P-tile:
  - Если знаем (предполагаем) что объект занимает  $P\%$  площади
  - $T$  устанавливаем так, чтобы отсечь  $P\%$  пикселей на гистограмме



# Расчет Т путем последовательных приближений



Частный случай алгоритма k-средних

1. Выбрать порог Т равным середине диапазона яркостей;
2. Вычислить среднюю яркость всех пикселей с яркостью < Т  $m_1$ , аналогично  $m_2$  для пикселей с яркостью > Т;
3. Пересчитать порог  $T = (m_1 + m_2) / 2$ ;
4. Повторять шаги 2, 3 порог не перестанет изменяться;



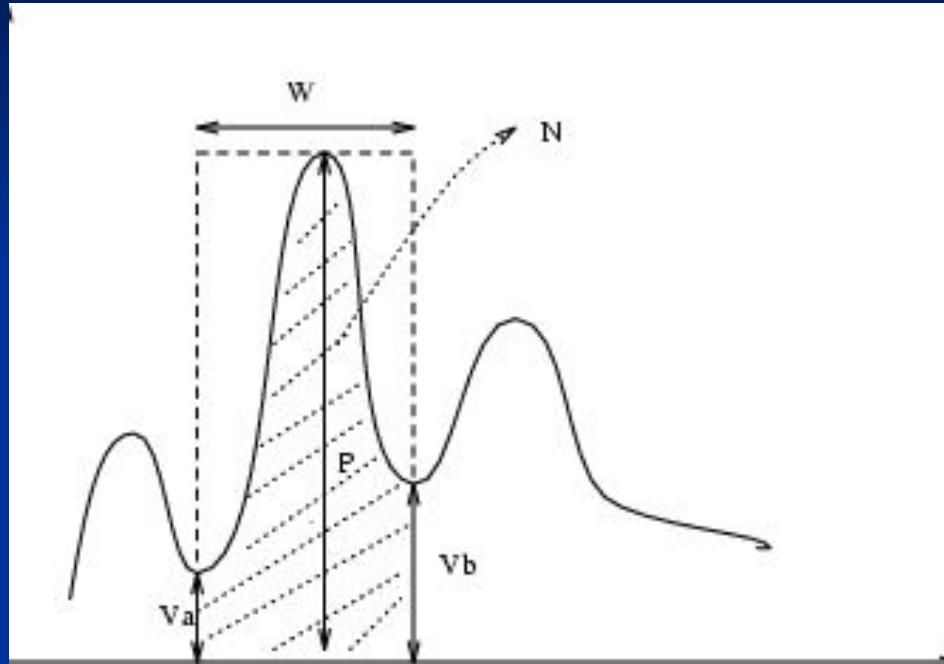


# Поиск пиков в гистограмме

1. Найти соседние локальные максимумы в гистограмме  $g_i$ ,
2. Рассчитать меру «пиковости» для  $g_i$ ,
3. Отфильтровать пики с слишком маленькой «пиковостью».
4. Для оставшихся найти самые «низкие» точки между пиками – это и будут пороги.



# Мера «ПИКОВОСТИ»

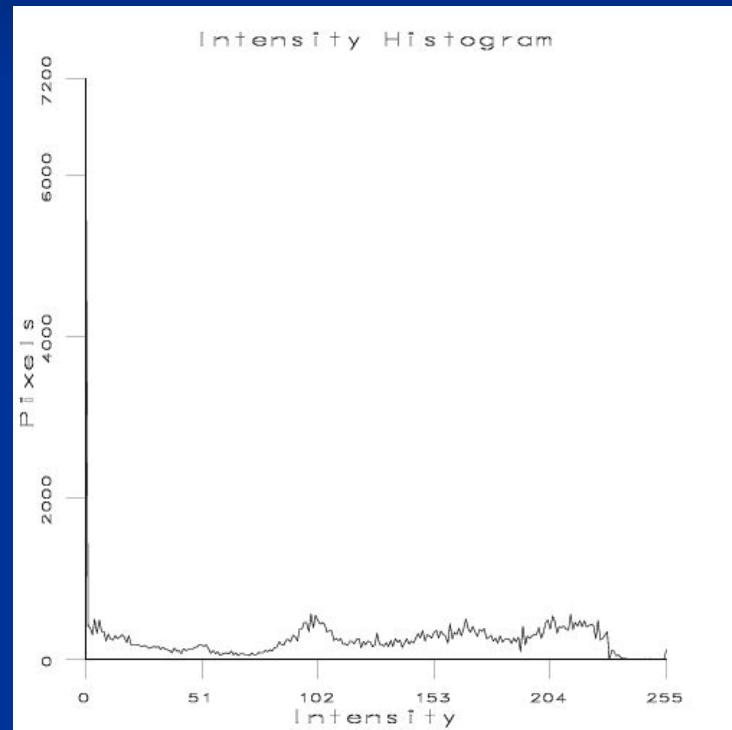
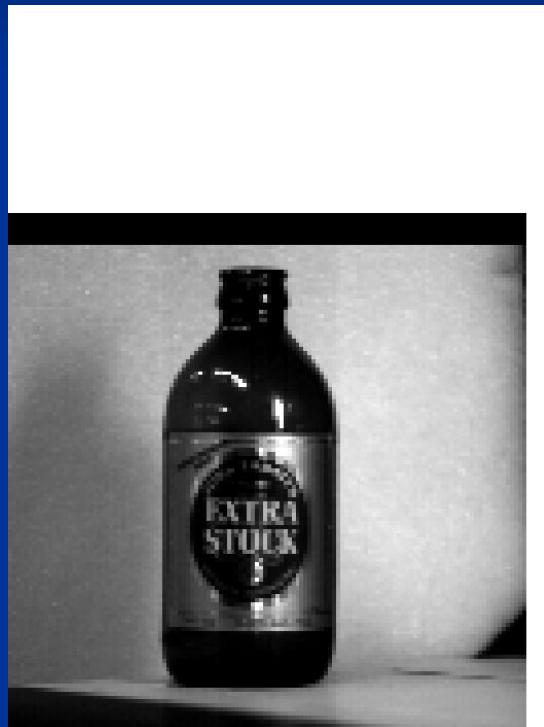


$$Peak = \left(1 - \frac{(V_a + V_b)}{2P}\right) \cdot \left(1 - \frac{N}{(W \cdot P)}\right)$$

# Зашумленность гистограмм



- Это проблема – много «лишних» локальных максимумов

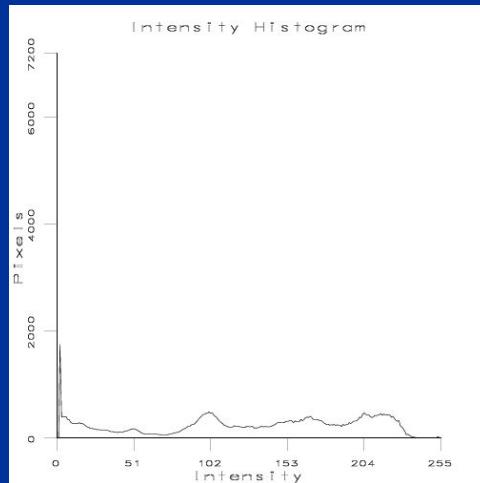


93 пика

# Сглаживание гистограмм

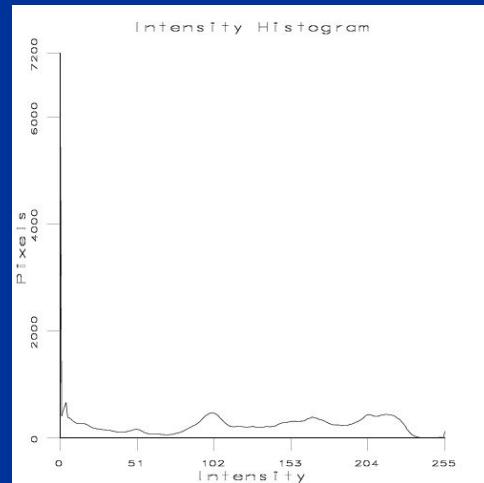


- Сглаживание посредством усреднения соседних значений
  - Свертка одномерным box-фильтром



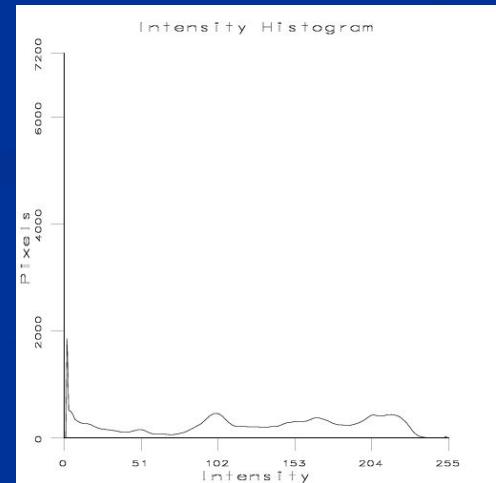
Сглажено 1 раз

54 пика  
«Пиковость» проходят 18



2 раза

21 пика  
«Пиковость» проходят 7

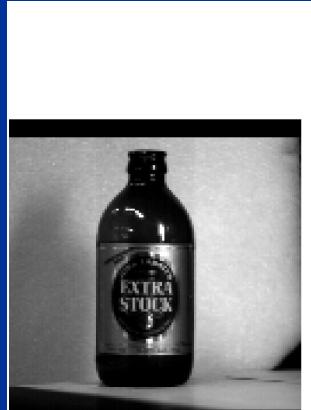


3 раза

11 пиков  
«Пиковость» проходят 4 peaks



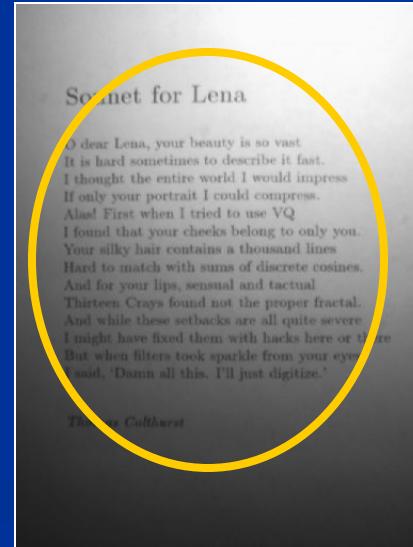
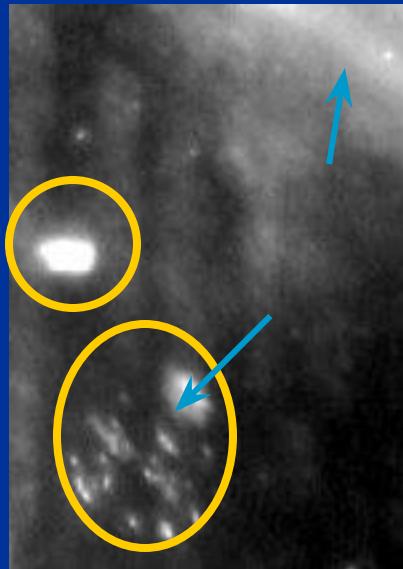
# Области найденные по пикам





# Адаптивный порог

- Проблема:
  - Яркость фона может быть разной в разных частях изображения
  - Единый порог не подойдет





# Адаптивный порог

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

Варианты выбора С по окрестности (x, y):

- $C = \text{среднее}$
- $C = \text{медиана}$
- $C = (\min + \max) / 2$

*Обратите внимание – начинаем учитывать пространственную информацию*

# Адаптивный порог



## Sonnet for Lena

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

*Thomas Colthurst*

## Sonnet for Lena

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

*Thomas Colthurst*

## 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.'

## 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

$r=7, T=0$

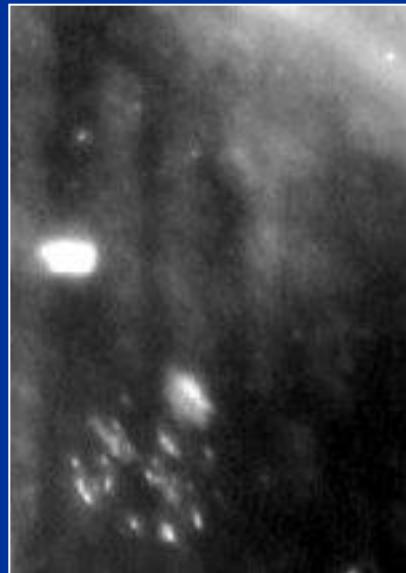
$r=7, T=7$

$r=75, T=10$

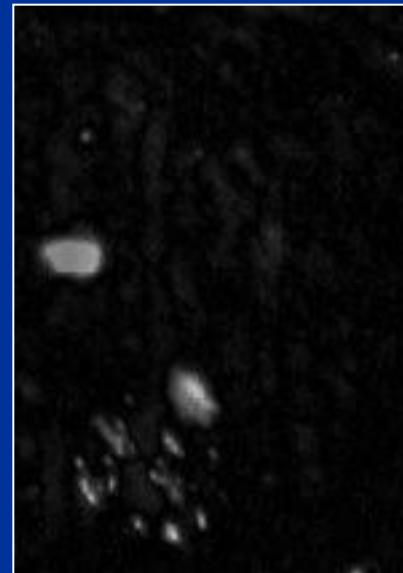


# Адаптивный порог

- Другая формулировка
  - Приближение фона усреднением
  - Вычитание фона -  $I(x, y) - C(x, y) > T$



Исходное



$I(x, y) - C(x, y), r=18$

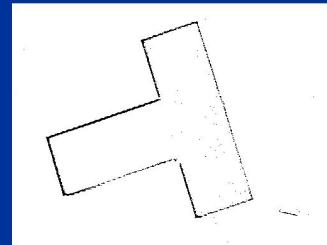


# Адаптивный порог

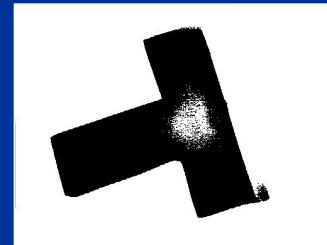
- Хорошо работает
  - Когда размер искомого объекта заметно меньше размера оцениваемой окрестности
- Хуже работает,
  - Когда объект велик по сравнению с самим изображением



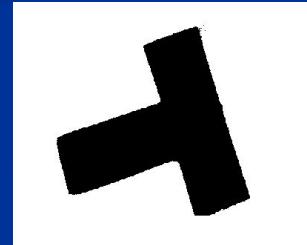
Исходное



$r=7$



$r=140$



$r=300$



# Кластеризация k-средних

- Способ определения нескольких порогов одновременно
- Нужно заранее знать  $k$  - количество диапазонов яркостей
  - В принципе можно  $k$  найти по гистограмме с помощью анализа «пиковости»

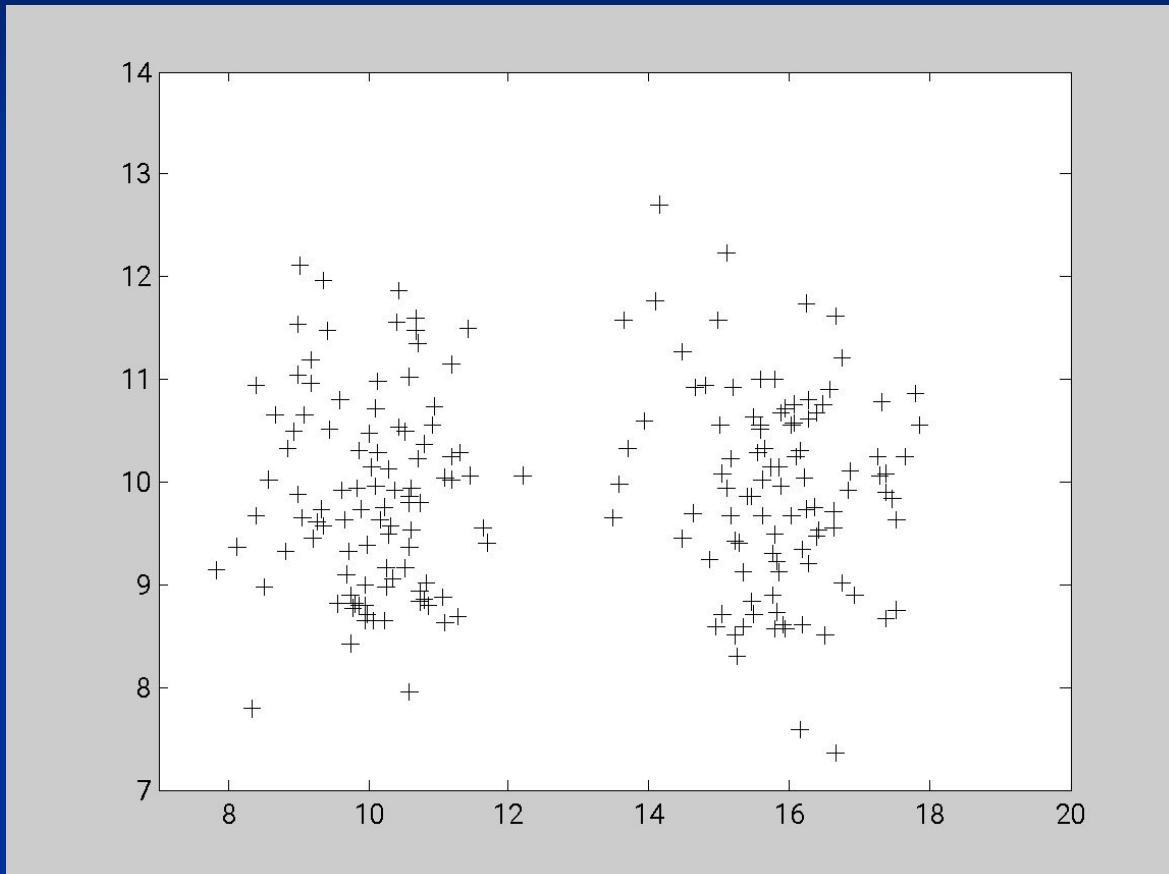


# Алгоритм k-средних

- **Входные данные** – набор векторов  $n$ -мерного пр-ва  $v_i$ ,  $i=1,\dots,p$ .
  - **Выходные данные** – центры кластеров  $m_j$ ,  $j=1,\dots,k$  и принадлежность  $v_i$  к кластерам
- 
1. Случайным образом выбрать  $k$  средних  $m_j$ ,  $j=1,\dots,k$ ;
  2. Для каждого  $v_i$ ,  $i=1,\dots,p$  подсчитать расстояние до каждого из  $m_j$ ,  $j=1,\dots,k$ ,
  3. Отнести (приписать)  $v_i$  к кластеру  $j'$ , расстояние до  $m_{j'}$  минимально;
  4. Пересчитать средние  $m_j$ ,  $j=1,\dots,k$  по всем кластерам;
  5. Повторять шаги 2, 3 пока кластеры не перестанут изменяться;



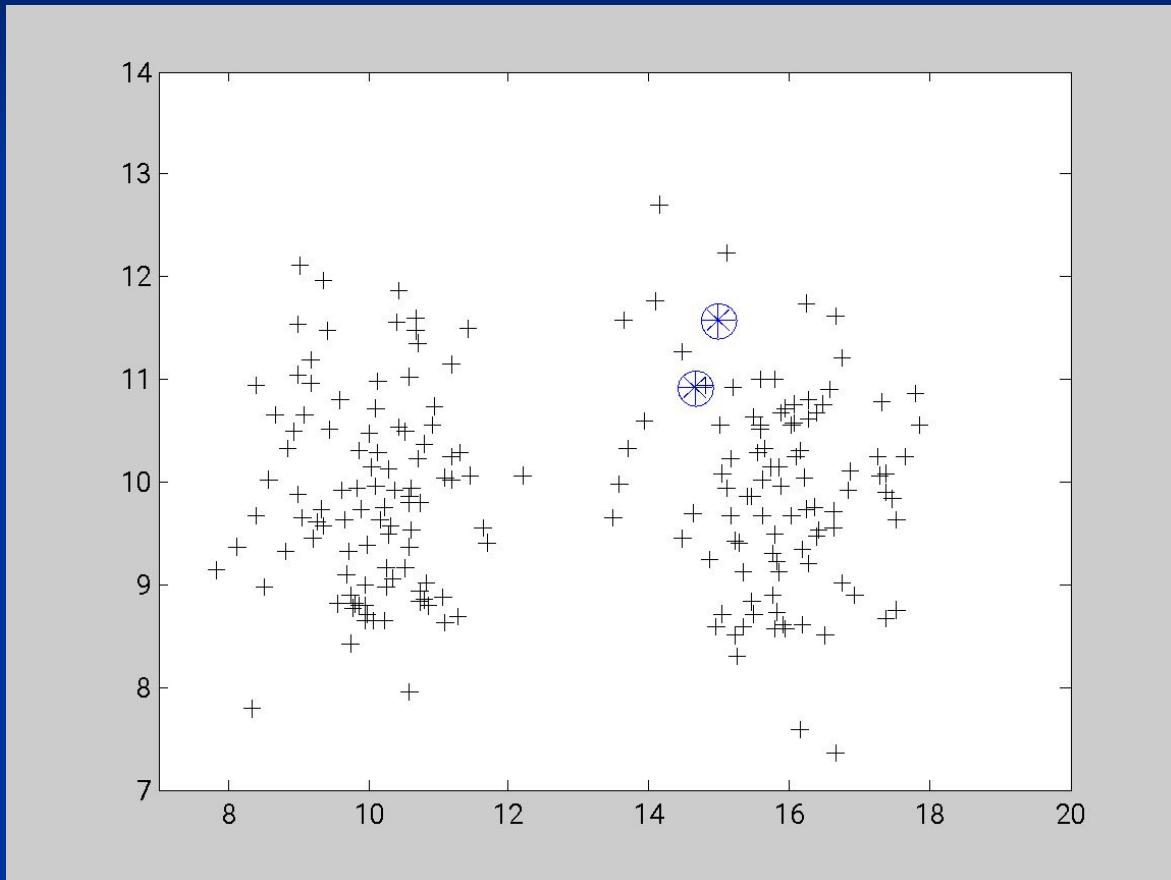
# Пример кластеризации в 2D



Исходные данные



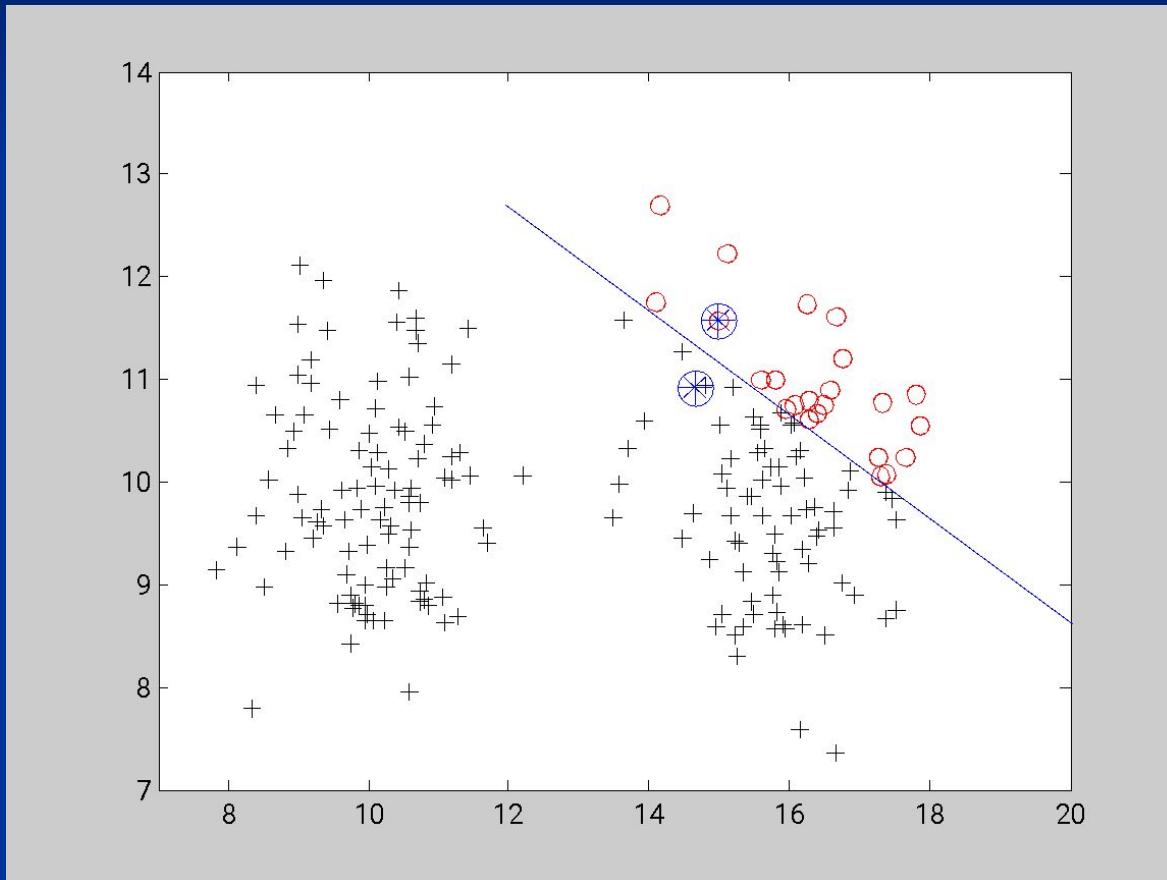
# Пример кластеризации в 2D



Случайная инициализация центров кластеров (шаг 1)



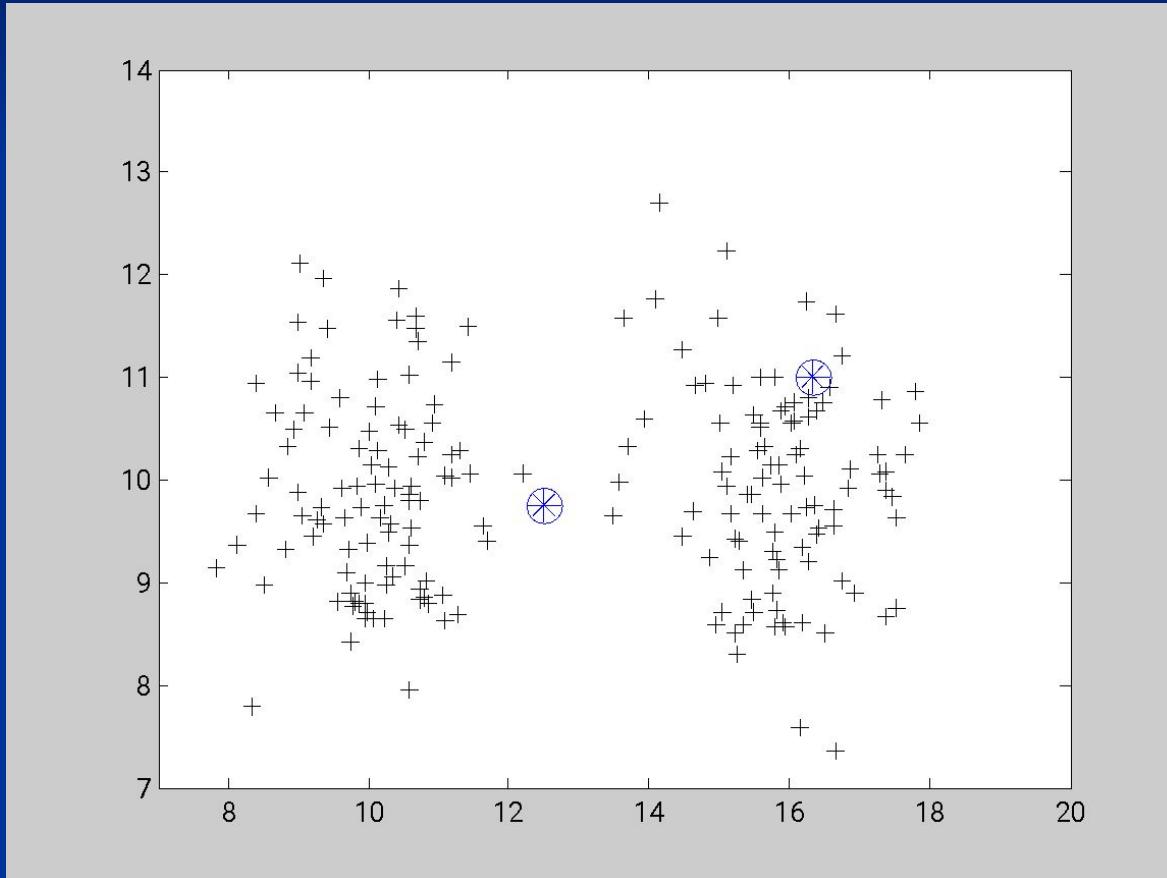
# Пример кластеризации в 2D



Кластеры после первой итерации (шаг 2)



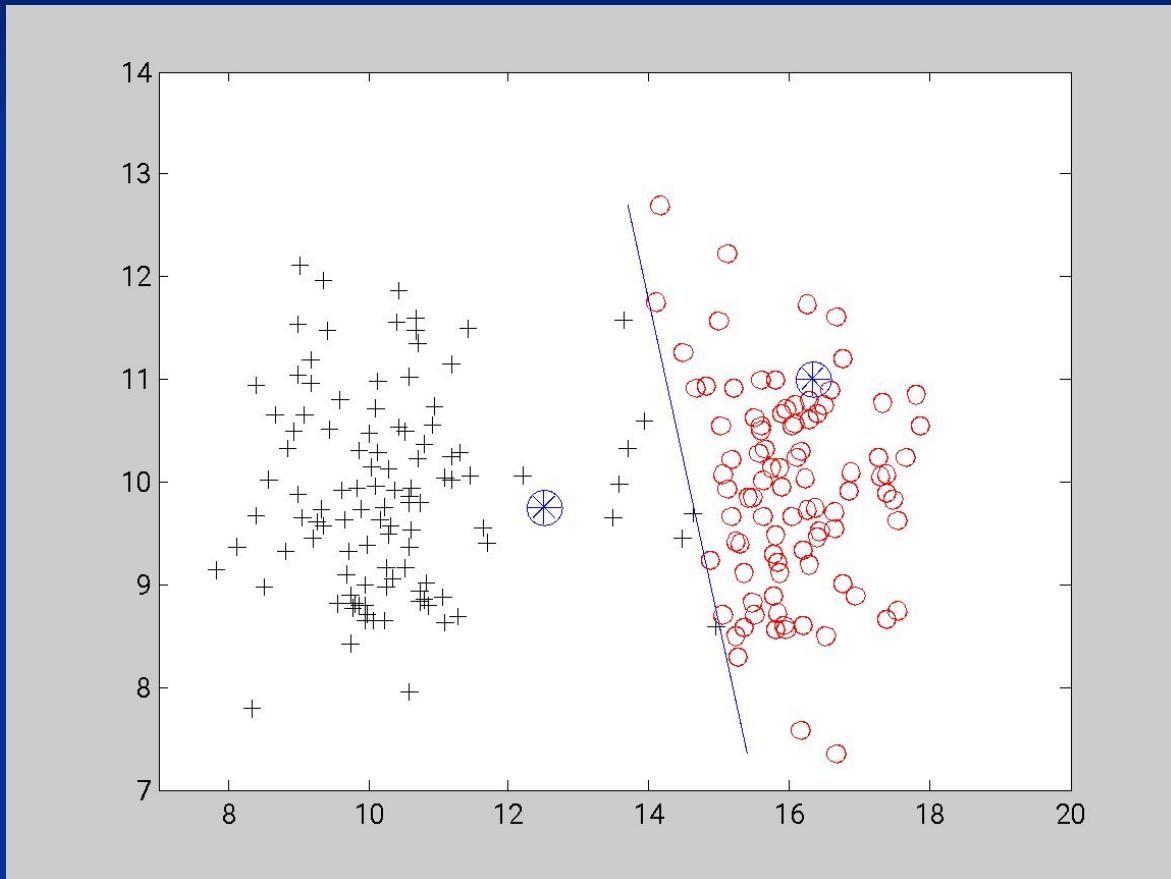
# Пример кластеризации в 2D



Пересчет центров кластеров после первой итерации (шаг 3)



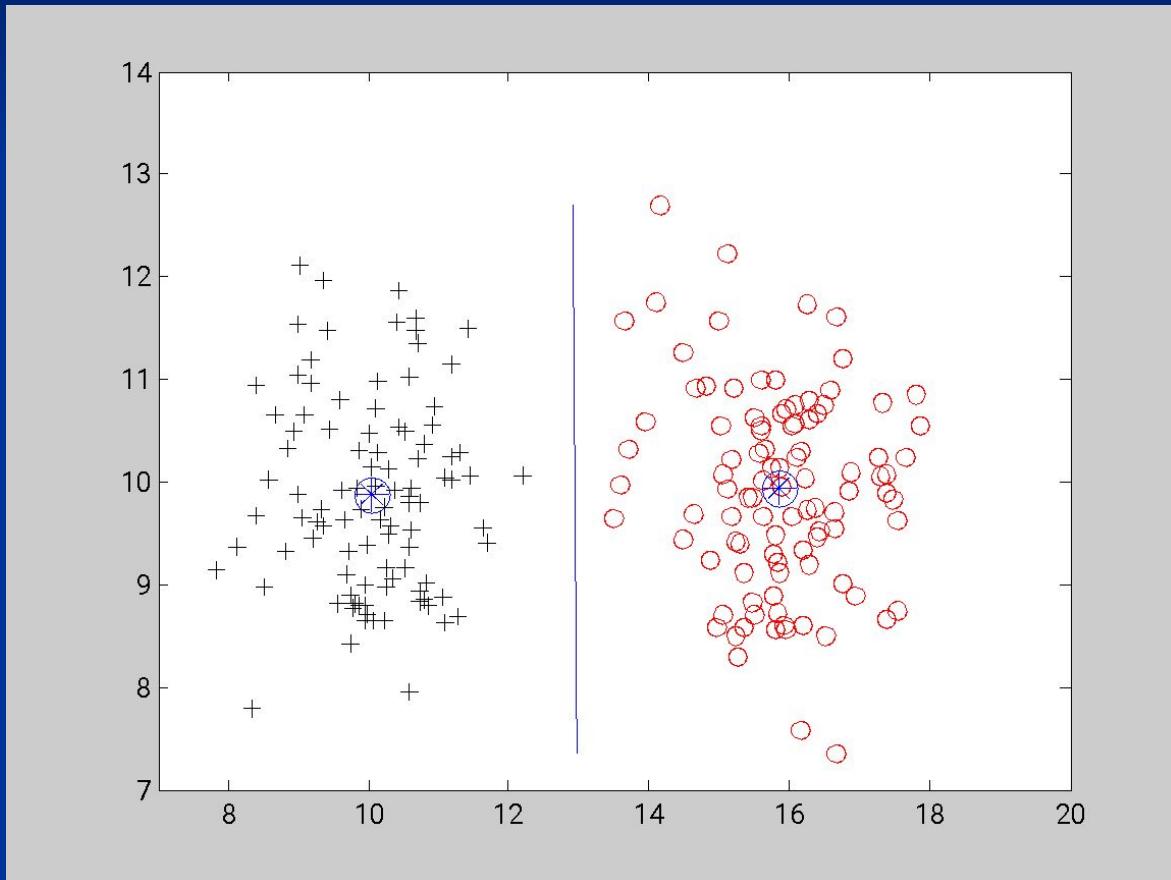
# Пример кластеризации в 2D



Кластеры после второй итерации (шаг 2)



# Пример кластеризации в 2D



Стабильная конфигурация после четвертой итерации



# k-средних для сегментации

- Если изображение одноканальное
  - $v_i = I(x, y)$  – работаем в одномерном пространстве
  - Получается итеративный алгоритм пересчета порога
- Если изображения трехканальное (RGB)
  - $v_i = (R(x, y), G(x, y), B(x, y))$  – работаем в трехмерном пространстве
- Можно работать и с многоканальными изображениями
  - Например – RGB + инфракрасный канал

# Алгоритм k-средних для одноканального изображения



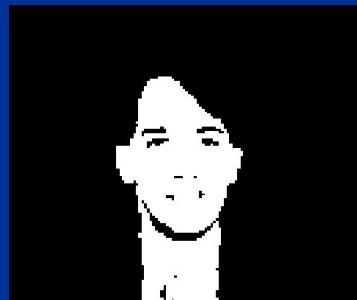
1. Случайным образом выбрать  $k$  средних  $m_j$ ,  $j=1,\dots,k$ ;
2. Для каждого пикселя  $(x,y)$  подсчитать  $D_j = |I(x,y) - m_j|$  для  $j=1,\dots,k$
3. Приписать  $(x,y)$  к кластеру  $j'$ ,  $D_{j'} = \min\{D_j, j=1,\dots,k\}$ ;
4. Пересчитать средние  $m_j$ ,  $j=1,\dots,k$  по всем кластерам;
5. Повторять шаги 2, 3 пока кластеры не перестанут изменяться;

# Сравнение k-средних с порогом по средней яркости

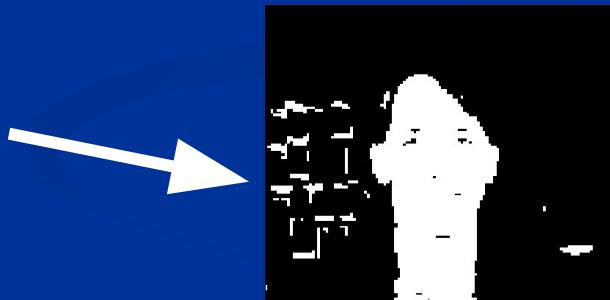


Чем отличается сегментация с помощью k-средних на 2 кластера от простейшей пороговой бинаризации по средней яркости изображения?

Пример:



k-средних



Порог по средней яркости

В причинах предлагается разобраться самостоятельно



# Общие недостатки описанного

- Игнорируется пространственное расположение пикселей
  - За исключением адаптивного порога, но и там соседство не учитывается
- Перейдем к методам, учитывающим взаимное расположение пикселей



# Понятие связности

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

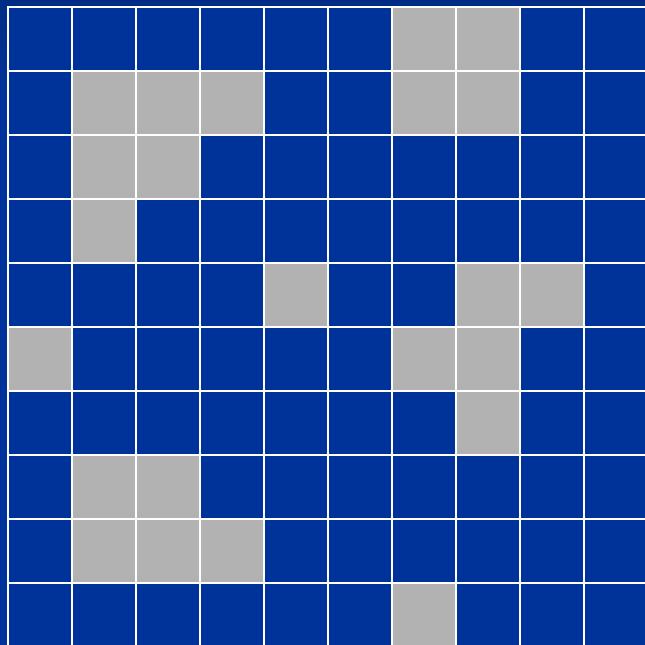
	1	
2	*	3
	4	

4-связность

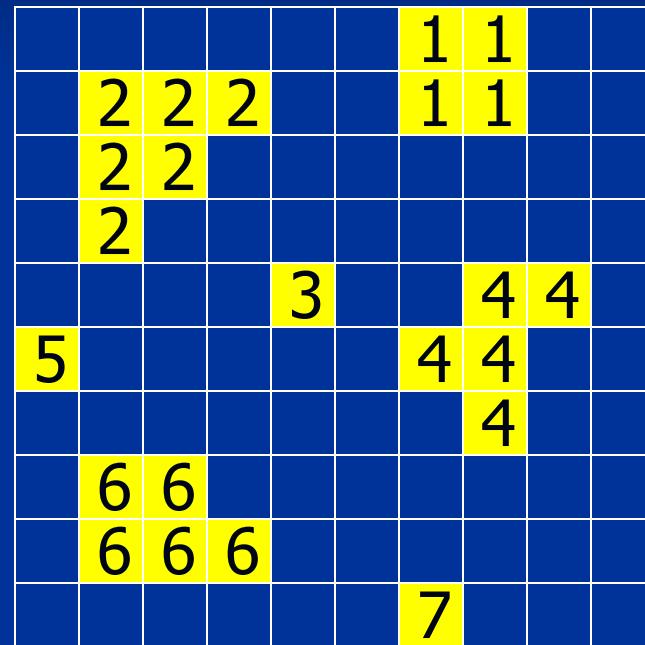
1	2	3
4	*	5
6	7	8

8-связность

# Разметка связных областей



Бинарное изображение



Размеченное изображение

# Разрастание регионов (Region growing)



- Простая идея – начиная с некоторого “семени” обходить пиксели и объединять в области пока выполняется условие однородности



# Что необходимо определить

- Критерий однородности
  - Гистограмма содержит не больше 1 значительного пика



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

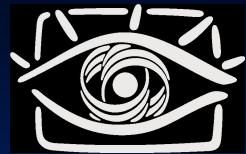
$$\forall p \in S \quad \left| I(p) - \frac{1}{N} \sum_{q \in S} I(q) \right| < T_{avg}$$

- Разница между соседними пикселями

1	2	3
4	*	5
6	7	8

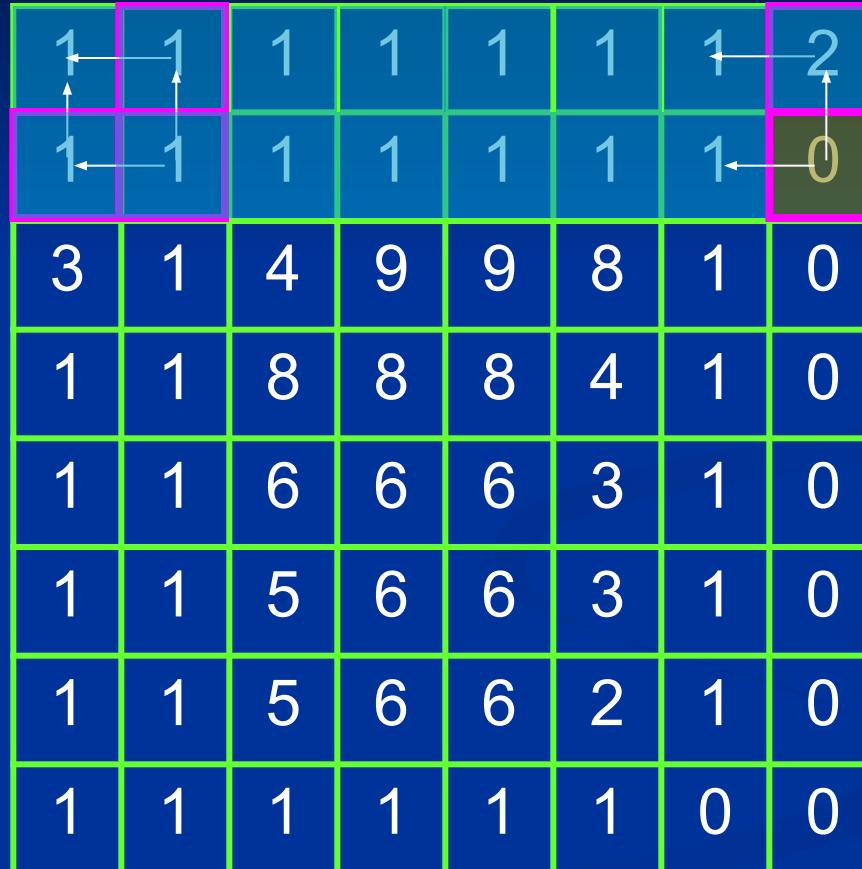
- «Слабая» граница между регионами (только для

# Алгоритм разрастания регионов



Среднее: 1

Среднее: 1.125



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

Пример  $\delta = 1$

# Алгоритм разрастания регионов



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



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

Пример  $\delta = 1$



# Разрастание регионов

Сканируем изображение сверху вниз, слева направо:

	C	
B	A	

1. if  $|I(A) - Cl_{avg}(B)| > \delta$  and  $|I(A) - Cl_{avg}(C)| > \delta$  -  
создаем новую область, присоединяя 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  $|I(A) - Cl_{avg}(C)| \leq \delta$  :
  1.  $|Cl_{avg}(B) - Cl_{avg}(C)| \leq \delta$  –  
сливаем области B и C.
  2.  $|Cl_{avg}(B) - Cl_{avg}(C)| > \delta$  –  
добавляем пикSEL A к тому классу, отклонение от  
которого минимально.

$I(A)$  – яркость пикселя A

$Cl_{avg}(B)$  – средняя яркость области к которой принадлежит B

# Разделение областей

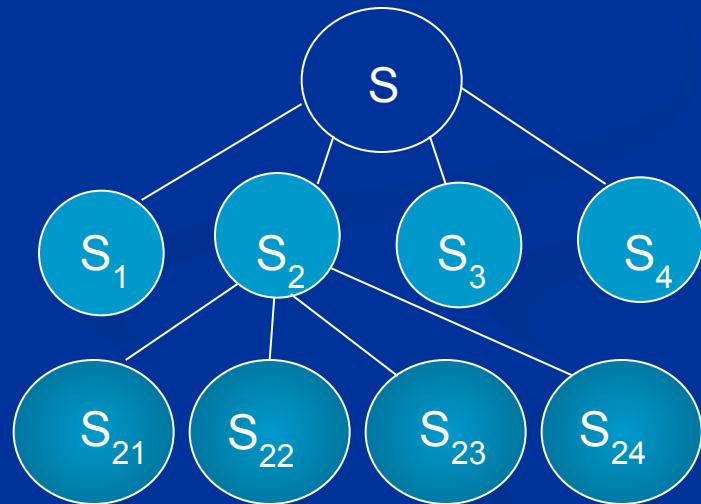
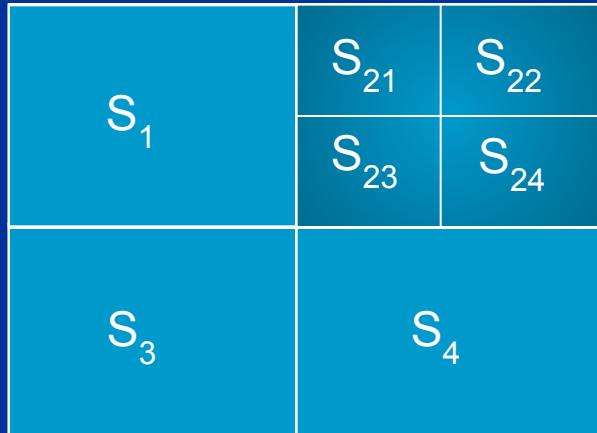


1. Первый шаг – всё изображение это одна область, поместить область в стек
2. Пока стек не пуст
  - Взять область  $S$  из стека
  - Проверить область на однородность
  - Если область неоднородна
    - разделить ее, новые области поместить в стек
  - Если область однородна
    - область больше не трогаем

# Что необходимо определить 2



- Правило разделения областей
  - Распространенный вариант – на 4 части, как квадр дерево



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

# 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

Пример

# Алгоритм разбиения (split)



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

Первое разбиение

# 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

Второе разбиение

# Алгоритм разбиения (split)



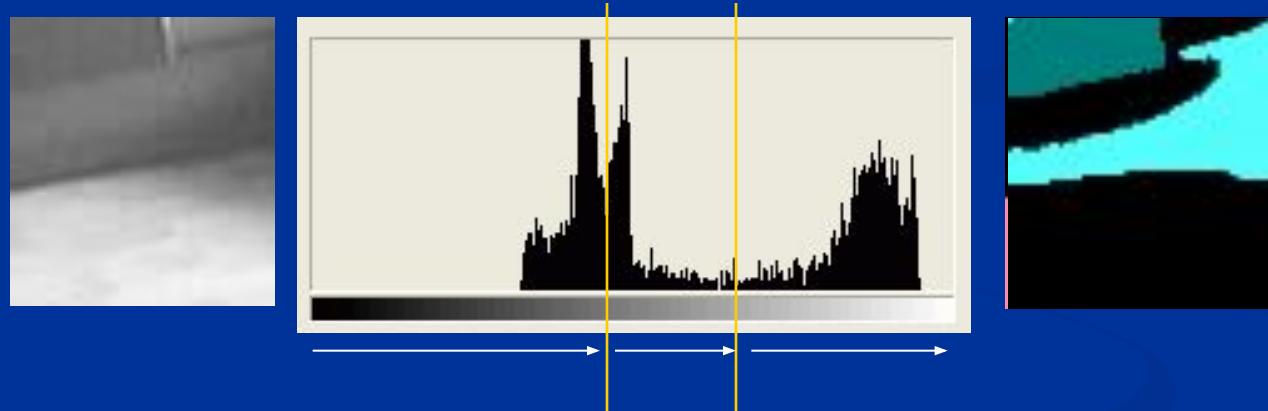
1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

Третье разбиение



# Что необходимо определить 3

- Правило разделения областей – более умно
  - Найти в гистограмме пики, разделить гистограмму по ним
  - Для каждой части гистограммы найти связные компоненты – это будут новые области



Реализовать сложнее, работает дольше



# Слияние областей

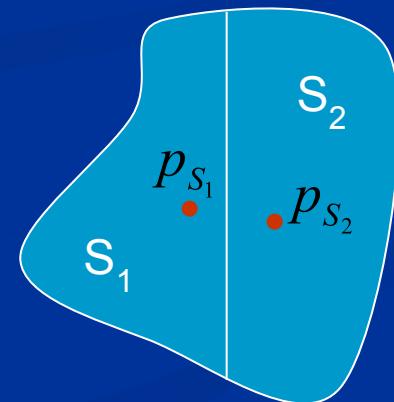
1. Первый шаг – каждый пиксель это отдельная область, поместить все области в стек
2. Пока стек не пуст
  - i. Взять область  $S$  из стека, для всех соседних областей  $S_i$ :
    - Проверить  $S' = S \cup S_i$  на однородность
    - Если  $S'$  однородна -
      - Слити  $S$  и  $S_i$ ,  $S'$  поместить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



# Алгоритм «фагоцита»

- Истонение 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*

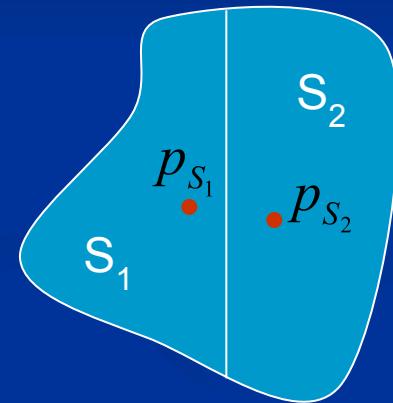


# Алгоритм «фагоцита»

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

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

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





# Алгоритм «фагоцита»

- Слитъ две области если:

$$\frac{W(\text{граница})}{\min(P_1, P_2)} > T_2, \quad 0 \leq T_2 \leq 1$$

где  $P_1$  и  $P_2$  – периметры областей  $S_1$  and  $S_2$

- Слитъ две области если:

$$\frac{W(\text{граница})}{\text{Кол - во тточе на границе}} > T_3, \quad 0 < T_3 \leq 1$$

# Алгоритмы разбиения и слияния



- Недостатки:
  - 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:
  - Сначала провести разбиение на небольшие однородные области
    - Обычно используется принцип квадродерева
  - Затем слить между собой те из них, которые вместе не нарушают требование однородности
    - Продолжать до тех пор, пока остаются регионы которые можно объединить

# Алгоритм разбиения/слияния (split and merge)



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

# Алгоритм разбиения/слияния (split and merge)



1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
	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

Результат

# Сравним с разрастанием регионов



1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
	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

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

Результат



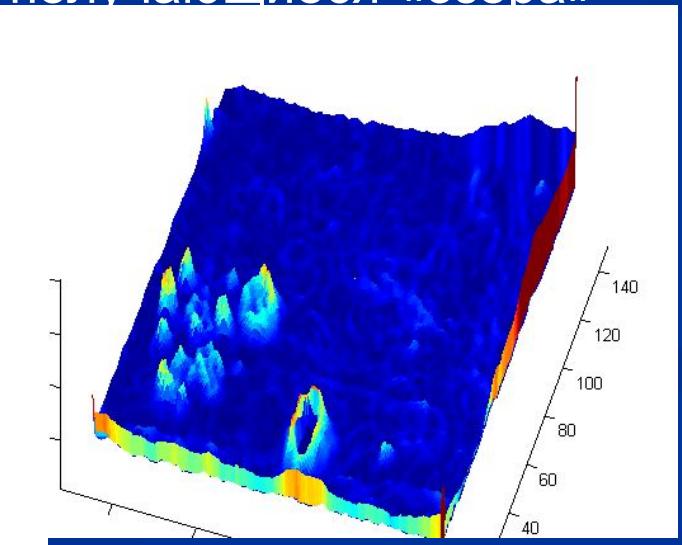
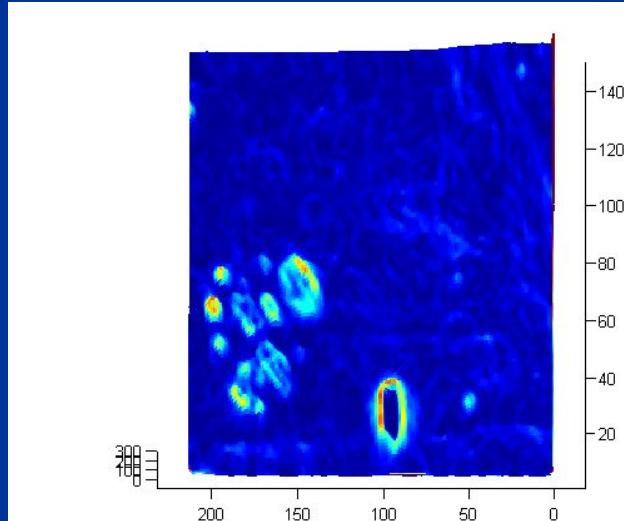
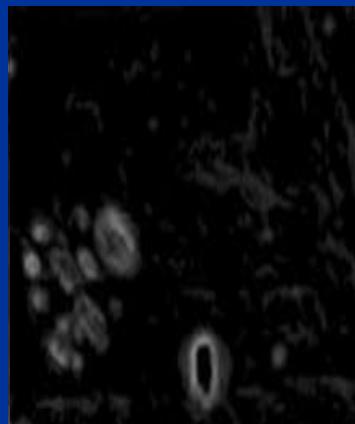
# Сравним подходы

- Сегментация на основе областей
  - В результате всегда замкнутые границы областей
  - Использование многоканальных изображений (RGB, RGB + ИК) обычно улучшает результаты
- Сегментация на основе границ
  - Границы обычно лучше локализованы

# Алгоритм водораздела (watershed)



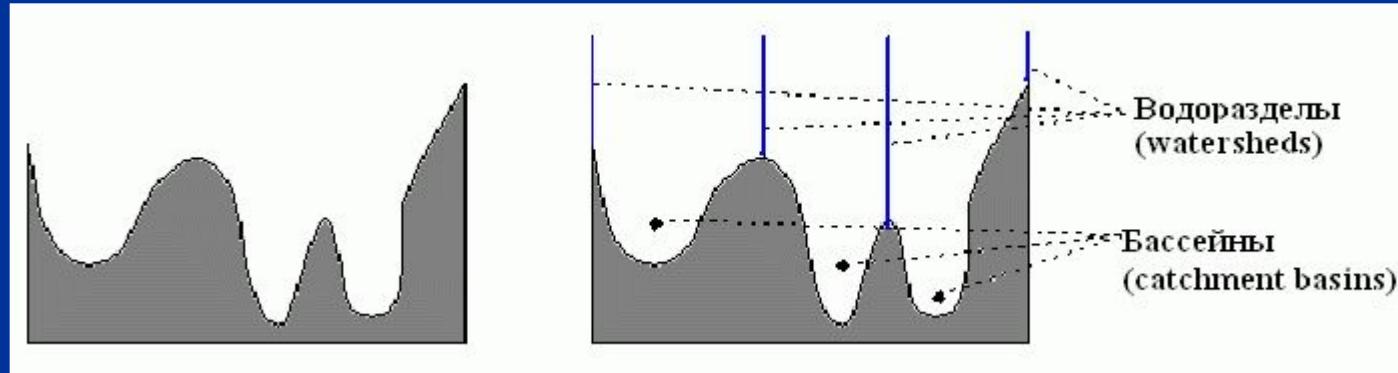
- Идея метода:
  - Вспомним – большие значения градиента соответствуют резким переходам на изображении
  - Рассмотрим абсолютную величину градиента как карту высот ландшафта
  - Там где резкие границы – получатся «стены»
  - Будем «лить воду» в «ямы» и искать получающиеся «озера»





# Алгоритм водораздела

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

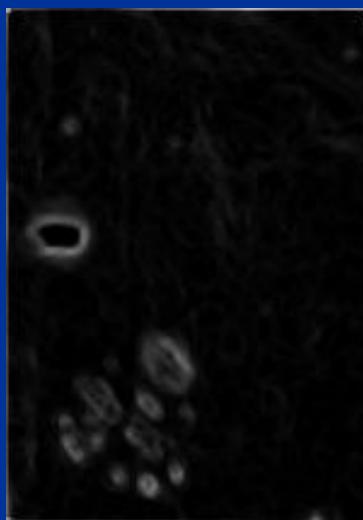


Слева – профиль интенсивностей изображения, справа – локальные минимумы определяют бассейны, локальные максимумы – линии водораздела.

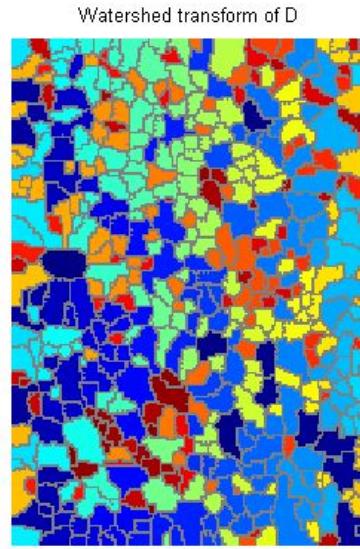
# Алгоритм водораздела



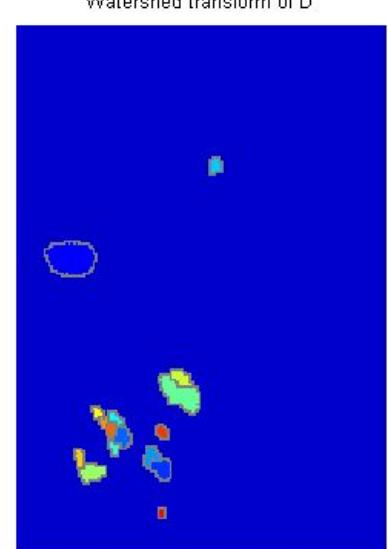
- Алгоритм, как и разбиение дает множество небольших регионов
  - Очень чувствителен к шуму – ищет все локальные минимумы



Абс. величина градиента



Результат по данному градиенту



градиент < 10 обращен в 0



# Алгоритм «погружения»

Алгоритм «погружения» (immersion) :

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

Для каждой яркости  $k$ :

Для каждой связной компоненте пикселей яркости  $k$ :

Если прилежит только к одному существующему  
бассейну

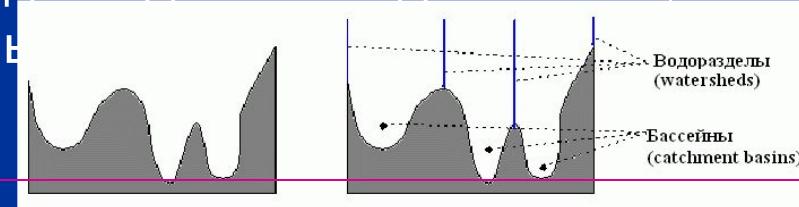
Добавить компоненту к бассейну

Если прилежит более чем к одному существующему  
бассейну

Пометить как границу (водораздел)

Иначе – создать новый бассейн

Аналог – вода медленно поднимается, пока не погружается в нее  
водоразделы



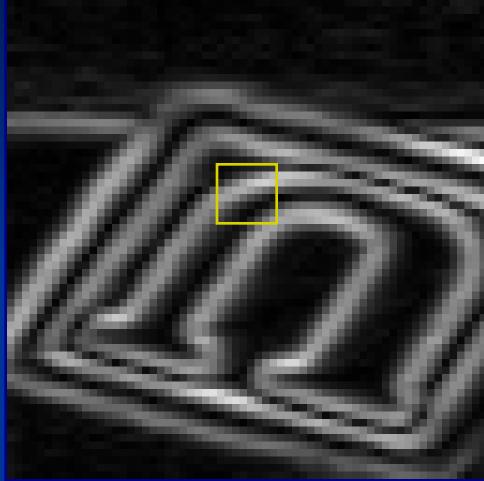
# Алгоритм tobogganing



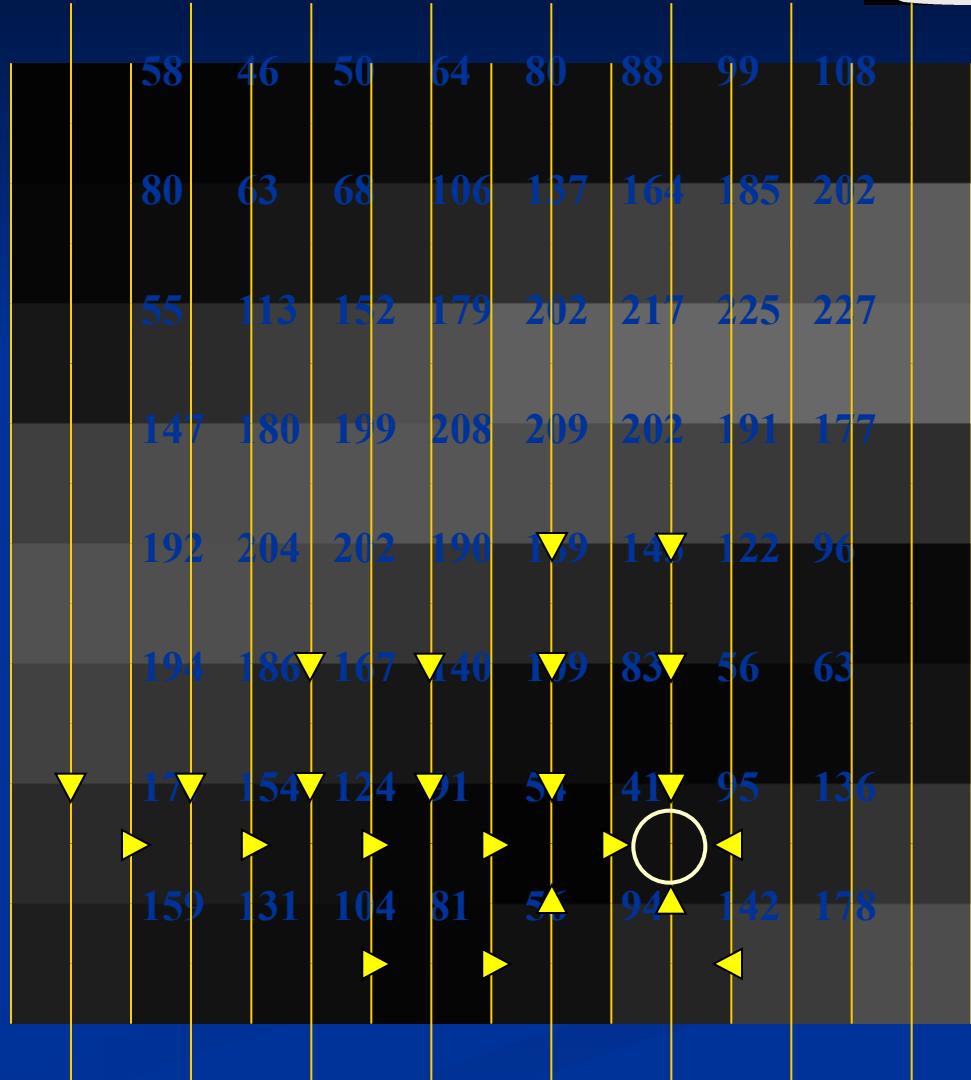
- Идея:
  - Из каждого пикселя «спускаемся» в локальный минимум среди его соседей
  - Спускаемся до тех пор, пока есть куда спускаться
  - Пиксели «спустившиеся» в один минимум – одна область

Как с горы на санках

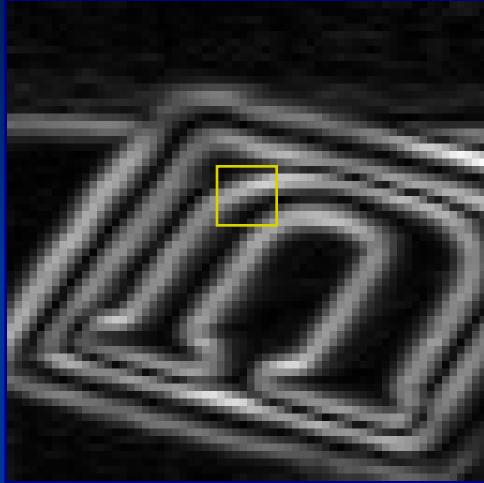
# Алгоритм tobogganing



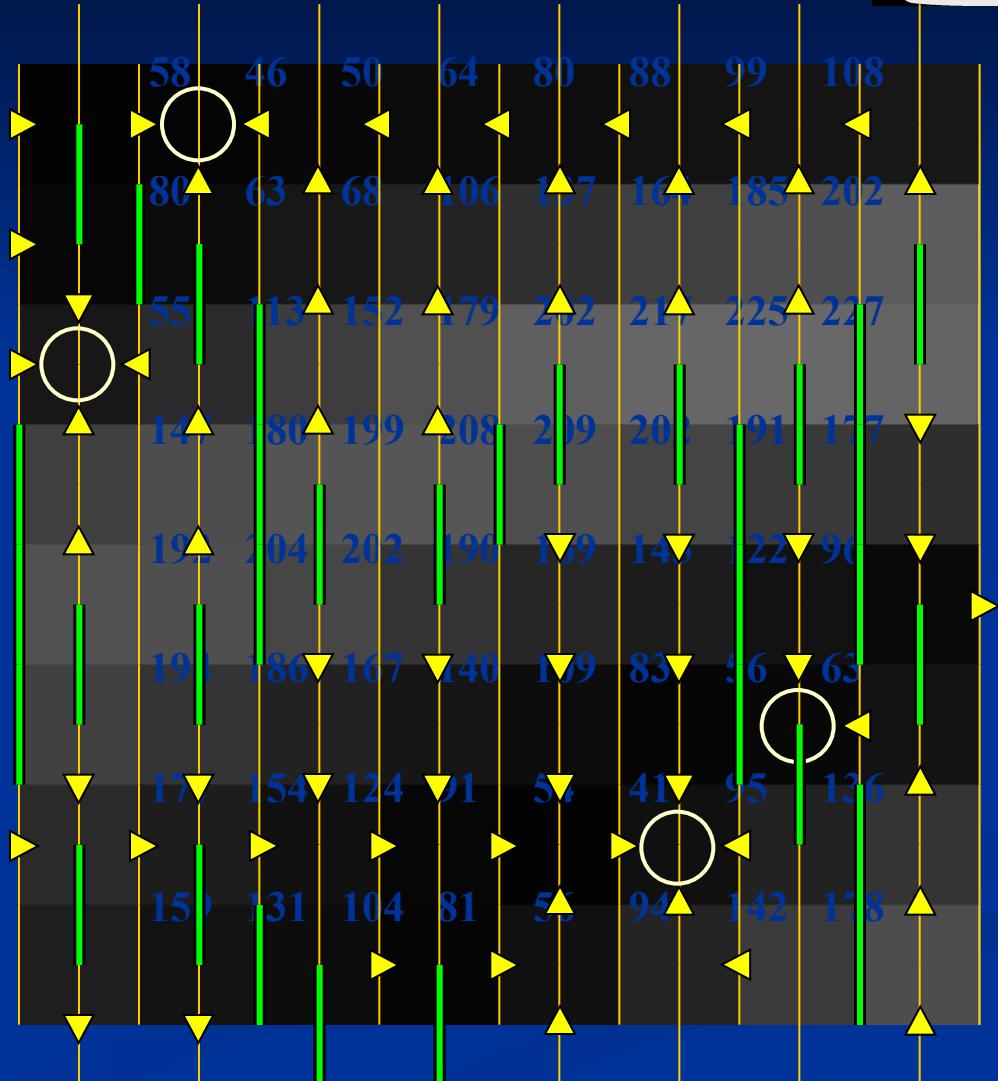
- Из каждого пикселя «спускаемся» в локальный минимум среди его соседей
  - Спускаемся до тех пор, пока есть куда спускаться
  - Пиксели «спустившиеся» в один минимум – одна область



# Алгоритм tobogganing



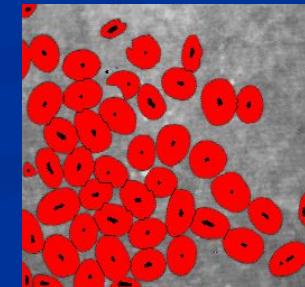
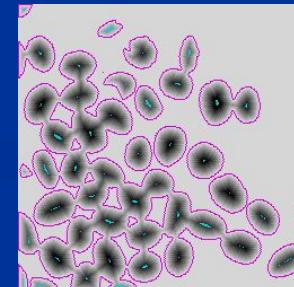
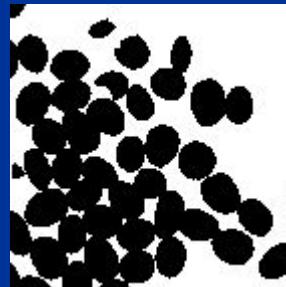
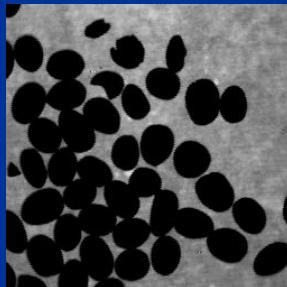
- Из каждого пикселя «спускаемся» в локальный минимум среди его соседей
- Спускаемся до тех пор, пока есть куда спускаться
- Пиксели «спустившиеся» в один минимум – одна область





# Tobogganing и водораздел

- В зависимости от задачи можно анализировать
  - само изображение
  - абсолютную величину его градиента
  - distance transform изображения (в каждой точке хранится расстояние до ближайшей границы)
  - Часто генерируют слишком много регионов, как и разделение
    - Требуется постобработка для слияния
  - В комбинации с distance transform хорошо для перекрывающихся регионов



# Методы теории графов

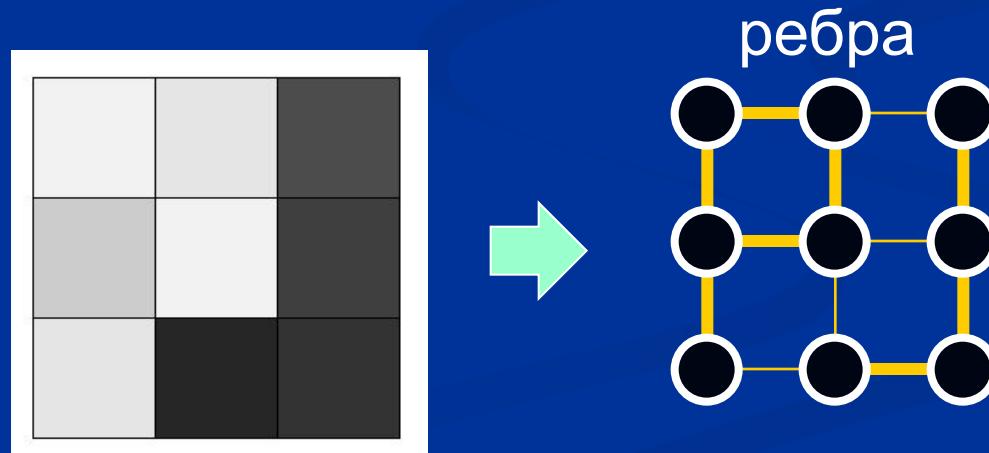


- Теория графов – хороший инструмент для работы с изображениями
  - Хорошая теоретическая база
  - Много проработанных методов
  - Изображение легко «превращается» в граф
- Математические модели теории графов хорошо применимы в частности для сегментации

# Граф и изображение



- Изображение превращается во взвешенный неориентированный граф
  - Пиксели – вершины графа
  - Ребра – связи между соседними пикселями
  - Вес ребер пропорционален «похожести» пикселей



# Критерии «похожести» пикселей



- По расстоянию

$$aff(\mathbf{x}, \mathbf{y}) = \exp\left\{-(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}) / 2\sigma_d^2\right\}$$

- По яркости

$$aff(\mathbf{x}, \mathbf{y}) = \exp\left\{- (I(\mathbf{x}) - I(\mathbf{y}))^2 / 2\sigma_I^2\right\}$$

- По цвету

$$aff(\mathbf{x}, \mathbf{y}) = \exp\left\{- (dist(c(\mathbf{x}) - c(\mathbf{y}))^2 / 2\sigma_c^2\right\}$$

- По текстуре

$$aff(\mathbf{x}, \mathbf{y}) = \exp\left\{- (dist(f(\mathbf{x}) - f(\mathbf{y}))^2 / 2\sigma_f^2\right\}$$

# 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 А и В из V
  - Удаляем все ребра, связывающие А и В

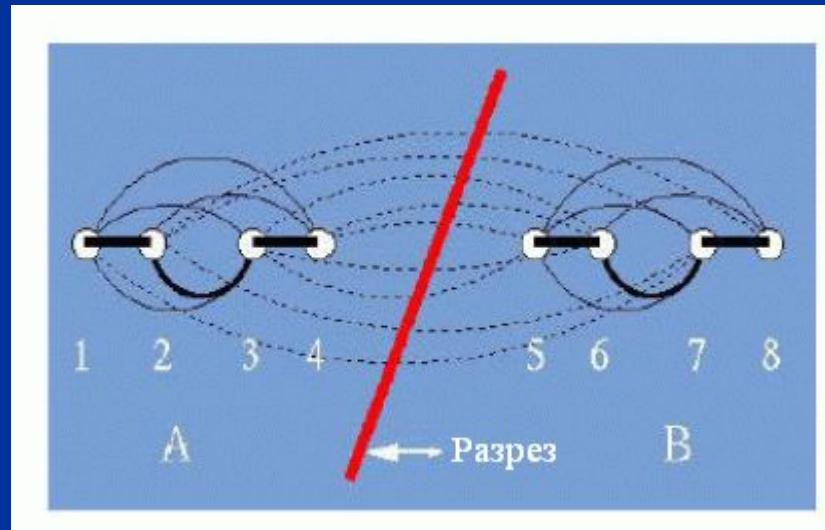
$$Cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

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



# Разрез графа

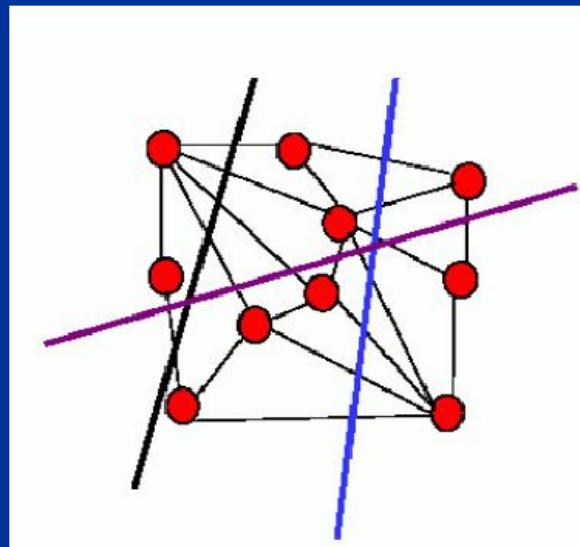
- Разрез графа превращает граф в два несвязанных друг с другом подграфа





# 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

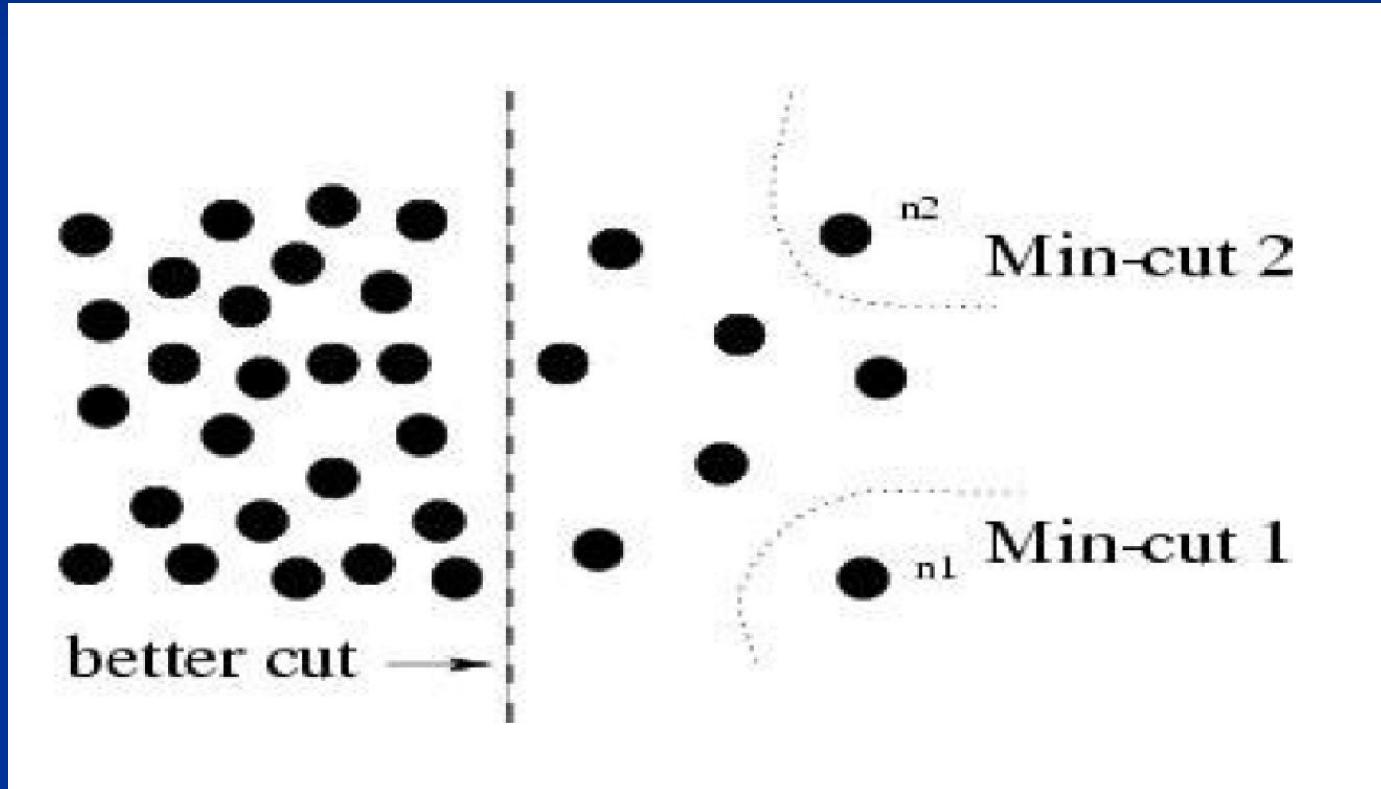


$$Cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

# 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

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

$$Ncut(A, B) = 2 - \left( \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)} \right)$$

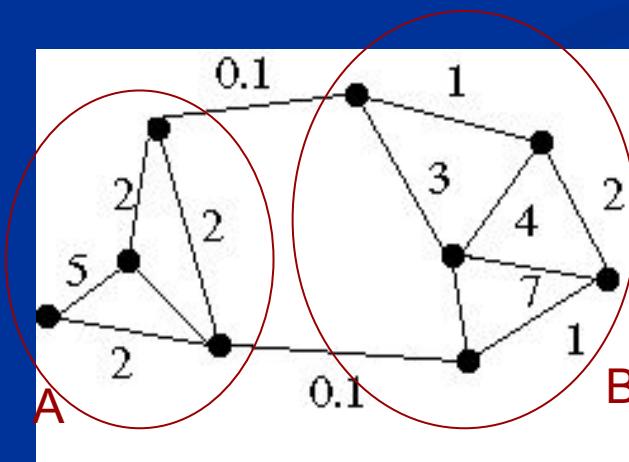
$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

Все ребра графа



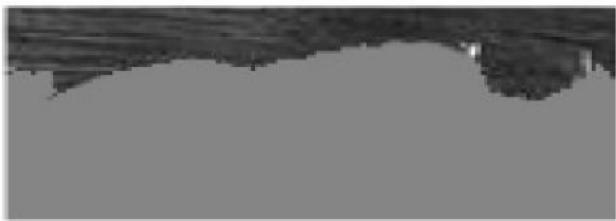
# Минимальный нормализованный разрез

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





# Пример:





# Подытожим:

- Рассмотрели следующие методы
  - Использующие края
    - Edge-based
  - Пороговой фильтрации
    - Thresholding
  - k-средних
    - k-means
  - Разрастания регионов
    - Region growing
  - Разделения / слияния
    - Split and merge
  - Водораздела
    - Watershed, tobogganing
  - Нормализованный разрез графа
    - Normalized cut

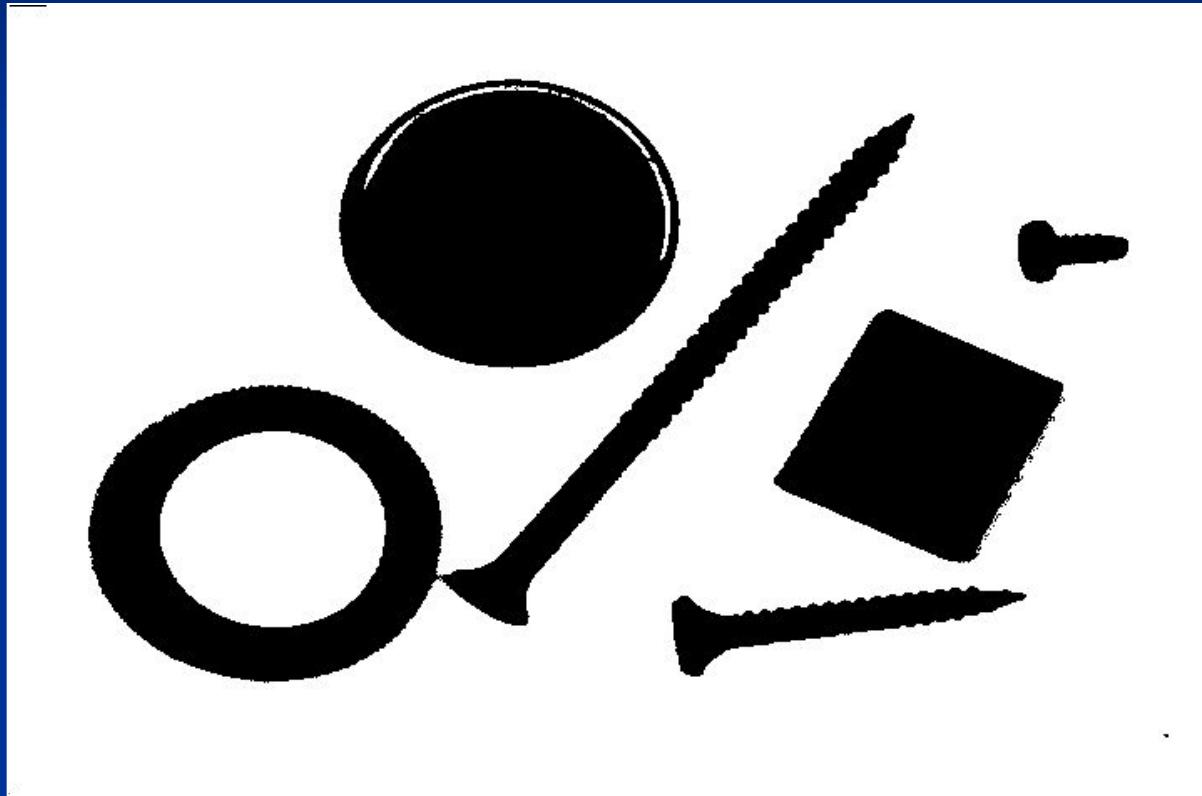
# Анализ областей после сегментации

Владимир Вежневец, Антон Конушин  
Александр Вежневец

Курс – «Введение в компьютерное зрение»  
МГУ ВМК, Graphics & Media Lab,  
Осень 2006



Какие параметры формы областей помогут  
различить объекты на этом примере?





# Свойства области

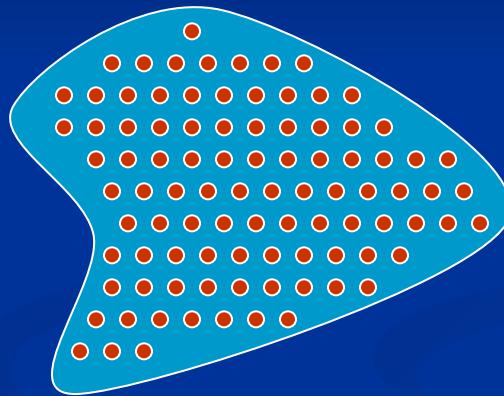
- Характеристики границы области
  - См. предыдущую лекцию
- Площадь
- Кол-во «дырок» внутри
- Центр масс
- Периметр
- Компактность
- Моменты
- Ориентация главной оси
- Цвет/яркость



# Площадь

- Кол-во пикселей в области

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



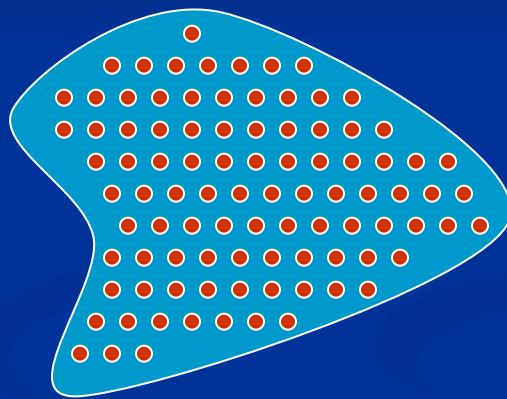


# Центр масс

## ■ Центр масс:

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

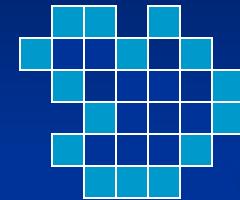
$$\bar{y} = \frac{\sum_{x=0}^m \sum_{y=0}^n y B(x, y)}{A}$$



# Периметр и компактность



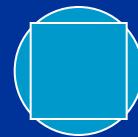
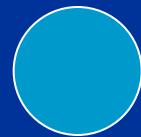
- Периметр - количество пикселей принадлежащих границе области



- Компактность

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

- Наиболее компактная фигура – круг,  $C = 4\pi$





# 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)
1. 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.



# Моменты

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

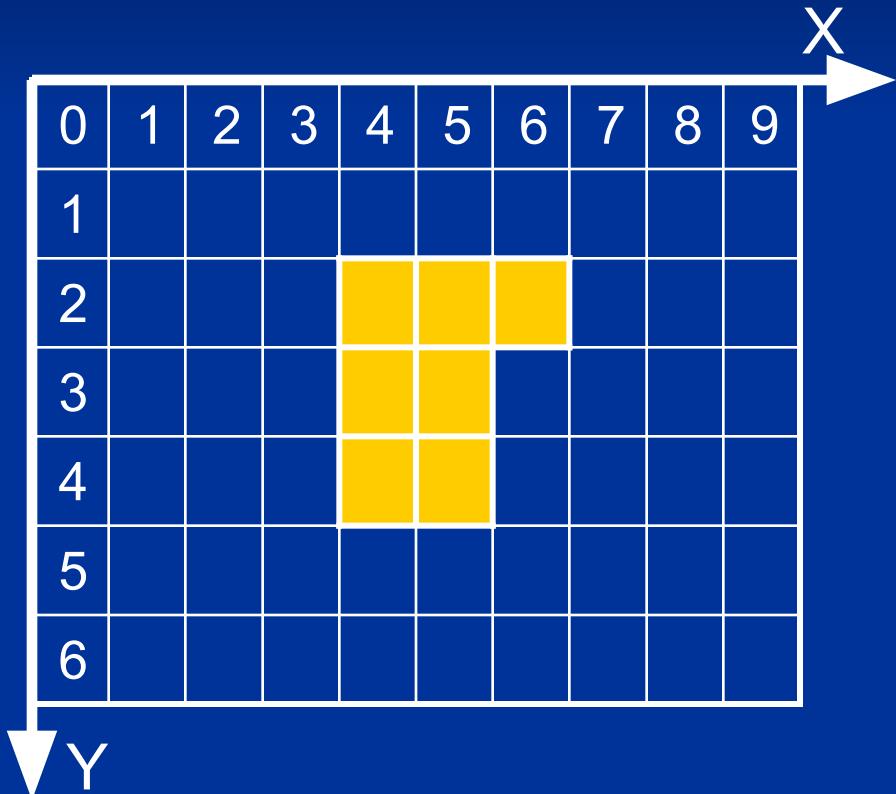
$$m_{ij} = \sum_{x,y \in S}^n x^i y^j B(x, y)$$

$B(x, y)$  - значение пикселя изображения  $(x, y)$



# Моменты

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



i	j	M <sub>ij</sub>
0	0	7
1	0	33
0	1	20
2	0	159
0	2	64
1	1	93

Площадь

Моменты  
инерции

# Центральные моменты



- Инвариантны к переносу

$$\mu_{pq} = \iiint (x - \bar{x})^p (y - \bar{y})^q B(x, y) d(x - \bar{x}) d(y - \bar{y})$$

Центр масс области



# Центральные моменты

$$\mu_{pq} = \iint (x - \bar{x})^p (y - \bar{y})^q B(x, y) d(x - \bar{x}) d(y - \bar{y})$$

$$m_{pq}^i = \int \int x^p y^q B(x, y) dx dy$$

$$\mu_{00} = m_{00} \equiv \mu$$

$$\mu_{01} = 0$$

$$\mu_{10} = 0$$

$$\mu_{20} = m_{20} - \mu \bar{x}^2$$

$$\mu_{11} = m_{11} - \mu \bar{x} \bar{y}$$

$$\mu_{02} = m_{02} - \mu \bar{y}^2$$

$$\mu_{30} = m_{30} - 3m_{20}\bar{x} + 2\mu \bar{x}^3$$

$$\mu_{21} = m_{21} - m_{20}\bar{y} - 2m_{11}\bar{x} + 2\mu \bar{x}^2 y$$

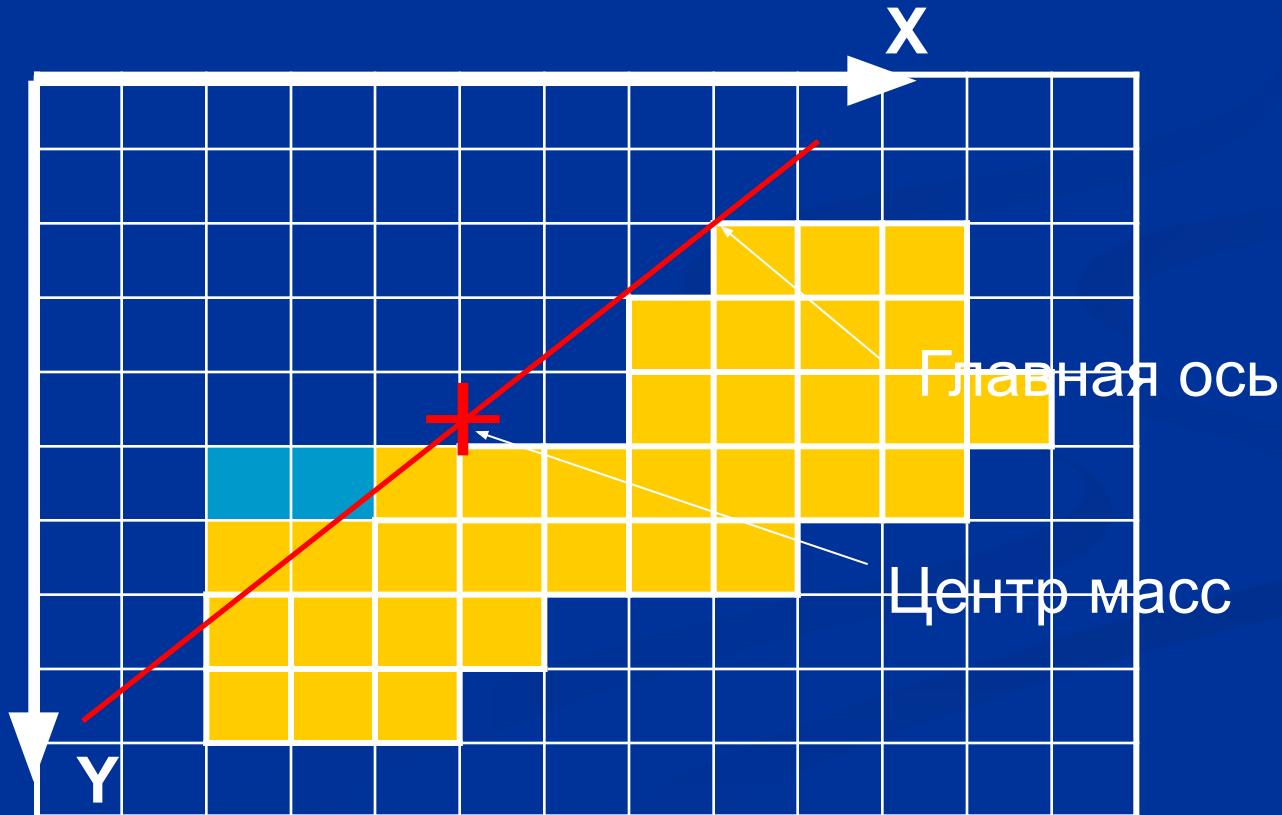
$$\mu_{12} = m_{12} - m_{02}\bar{x} - 2m_{11}\bar{y} + 2\mu \bar{x} y^2$$

$$\mu_{03} = m_{03} - 3m_{02}\bar{y} + 2\mu \bar{y}^3$$



# Ориентация главной оси инерции

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



# Моменты Hu



$$\mu_{pq} = \iint (x - \bar{x})^p (y - \bar{y})^q B(x, y) d(x - \bar{x})d(y - \bar{y})$$

- Инвариантны к повороту, переносу, скалированию

$$\phi_1 = \mu_{20} + \mu_{02}$$

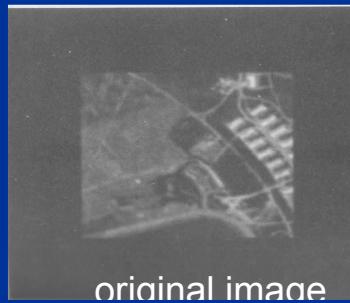
$$\phi_2 = (\mu_{20} - \mu_{02})^2 + \mu_{11}^2$$

$$\phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{12} - \mu_{03})^2$$

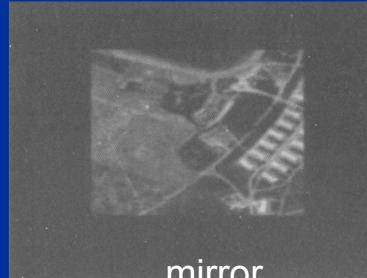
$$\phi_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

⊗

# Пример



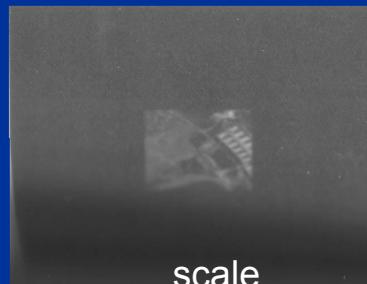
original image



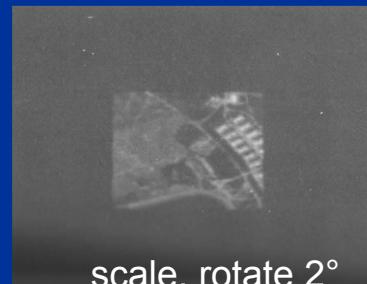
mirror



mirror scale rotate 4



scale



scale, rotate 2°

**Table 8.2 Moment Invariants for the Images in Figs. 8.24(a)–(e)**

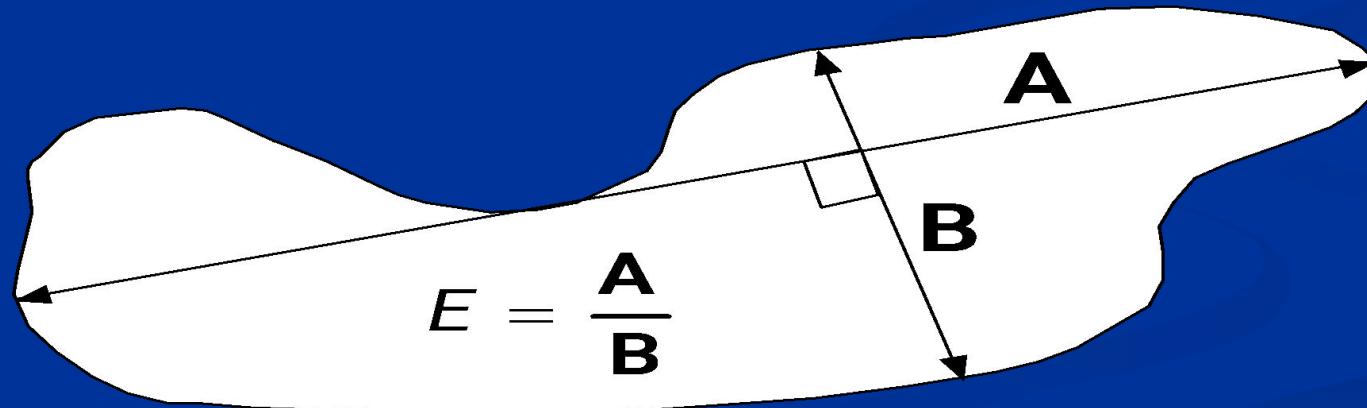
Invariant (Log)	Original	Half Size	Mirrored	Rotated 2°	Rotated 4°
$\phi_1$	6.249	6.226	6.919	6.253	6.318
$\phi_2$	17.180	16.954	19.955	17.270	16.803
$\phi_3$	22.655	23.531	26.689	22.836	19.724
$\phi_4$	22.919	24.236	26.901	23.130	20.437
$\phi_5$	45.749	48.349	53.724	46.136	40.525
$\phi_6$	31.830	32.916	37.134	32.068	29.315
$\phi_7$	45.589	48.343	53.590	46.017	40.470



# Инвариантные характеристики области

- Удлиненность, нецентрированность (эксцентриситет)

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



# Цвет, яркость



- Цвет и яркость области тоже хорошие признаки. Варианты
  - Гистограмма яркости, цветов в данной области
  - Средняя яркость, средний цвет
  - Дисперсия яркости, цветов (R, G, B) внутри области

# Немного о машинном обучении



- Мы рассмотрели сейчас методы «низкого уровня»
  - Они анализируют небольшое кол-во «простой» информации
- При рассказе о машинном обучении будут упомянуты методы производящие более «умный» анализ изображения



# Задание

- Выдадим на следующей лекции
- Выполняться будем на MATLAB
- Всем желающим получить задание  
нужно будет записаться на лекции
  - Будет распределение по вариантам



# Благодарности

- В лекции использовались иллюстрации из курсов:
  - Nick Krouglicof
    - Memorial University of Newfoundland
  - K. K. Biswas
    - University of Central Florida
  - Alper Yilmaz
    - University of Central Florida
  - Dr. Boyer
    - Ohio State University