

Лаборатория компьютерной  
графики и мультимедиа  
ВМК МГУ имени М.В.  
Ломоносова

Курс «Введение в компьютерное зрение  
и глубокое обучение»

# Лекция №4 «Классификация изображений и выделение объектов»

Антон Конушин

Заведующий лабораторией компьютерной графики и мультимедиа  
ВМК МГУ

11 марта 2019 года



# Классификация изображений

---



## Бинарная классификация

- Пешеход ли это?
- Бинарный ответ  
 $y \in [0,1]$ , 1 — да, 0 — нет



Car

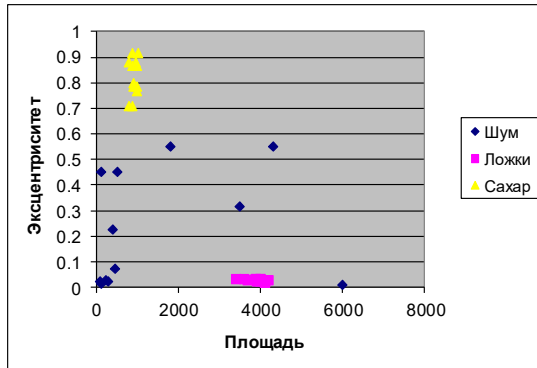
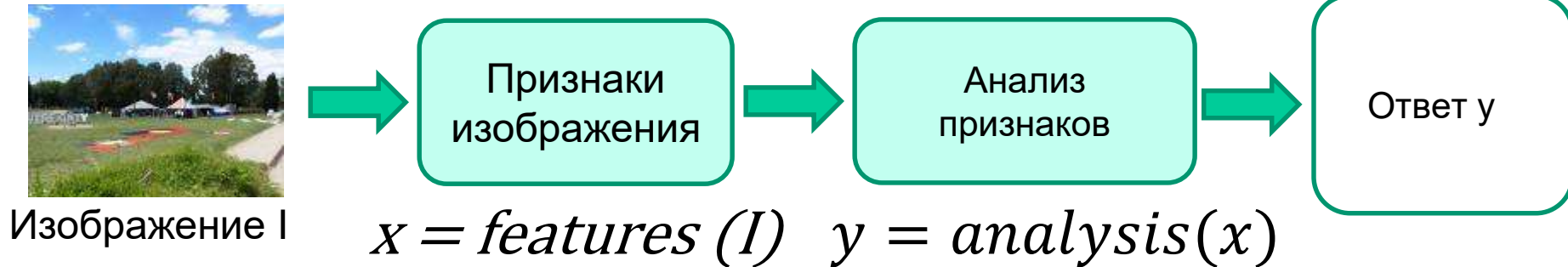
## Многоклассовая классификация

- К какому из заданных  $s$  классов относится данное изображение?
  - Ответ — метка класса  $s \in [1, S]$
- 
- Другое название — категоризация изображений
  - Определить, есть ли на изображении объект (сцена) заданной класса (категории)



# Классификация изображений

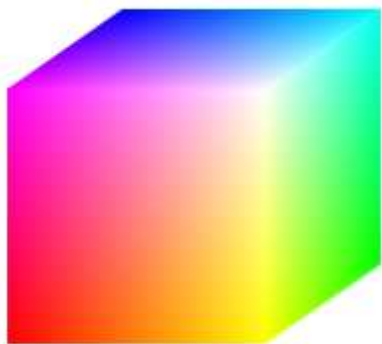
## Декомпозиция задачи



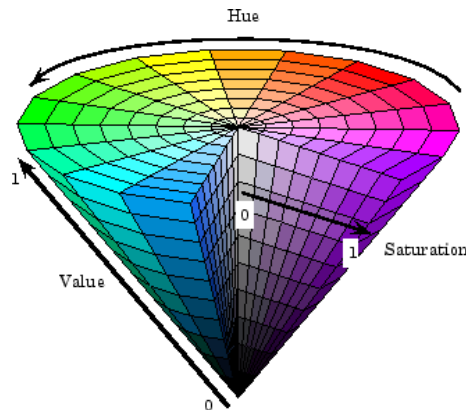
- Умеем строить классификаторы с помощью машинного обучения
- Какие признаки можно использовать для классификации?



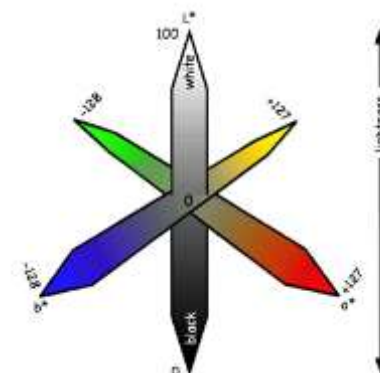
# Признаки пикселей



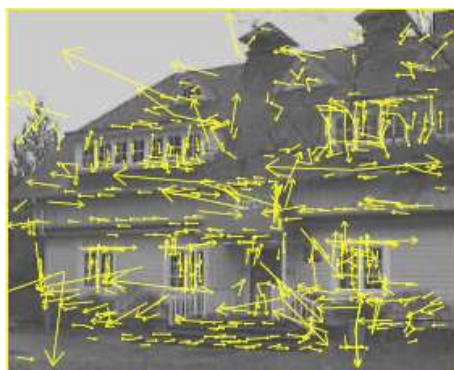
Пространство  
цветов RGB



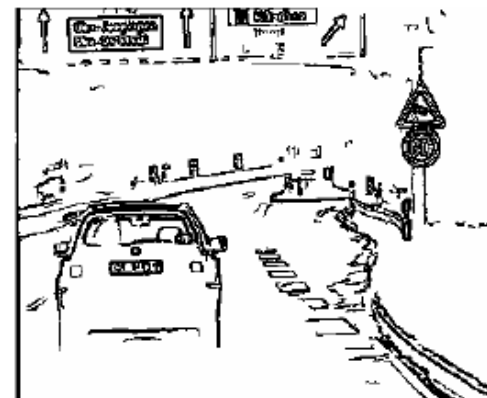
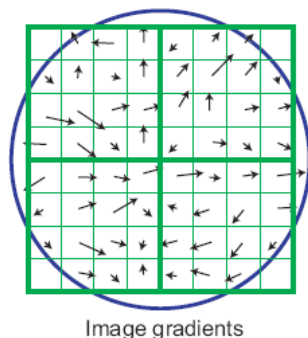
Пространство  
цветов HSV



Пространство  
цветов  $L^*a^*b^*$



Градиенты в каждом  
пикселе



Наличие и ориентация  
края в каждом пикселе



# Использование напрямую

---



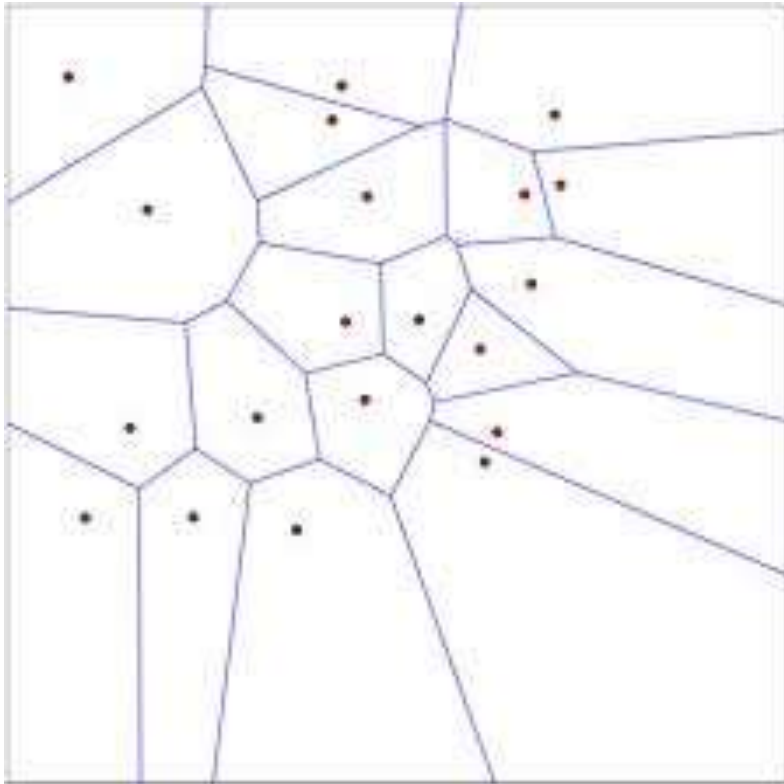
- Можно ли использовать признаки всех пикселей напрямую?
- Можно, если все изображения для классификации будут одинакового размера
- Нормализуем изображения, т.е. приведем их к одному размеру
- Вытянем изображение в вектор, признаки пикселей будут элементами вектора

Для распознавания, изображение должно описываться вектор-признаком фиксированной длины!



# Применение NN-классификатора

## Ближайший сосед (Nearest Neighbor)



- Будем сравнивать изображения попиксельно (L2-метрика)

$$D_{SSD}^2 = \sum_{x,y,c} (I_1(x, y, c) - I_2(x, y, c))^2$$

- Будем искать наиболее похожие по метрике L2 по всей или части коллекции

- Запоминаем M элементов
- Для нового вектора x ищем ближайший и берём метку его





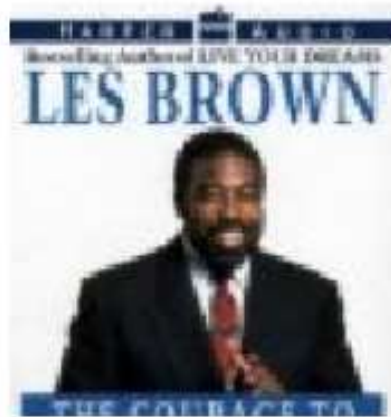
# Применение NN-классификатора



- Проведём 3 эксперимента с коллекциями в 7900, 790000, 79000000 изображений
- Похожие визуально изображения содержат те же самые объекты



# Выделение лиц через NN



a)



25



27



20





# Примеры работы

---

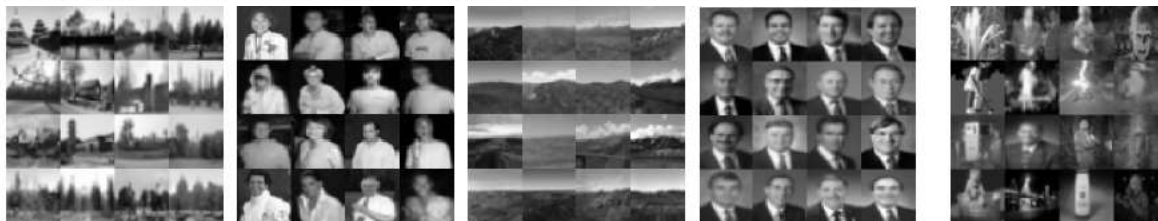




Gray scale  
input



Gray level  
32x32 siblings



High resolution  
color siblings



Average color



Average  
colorization



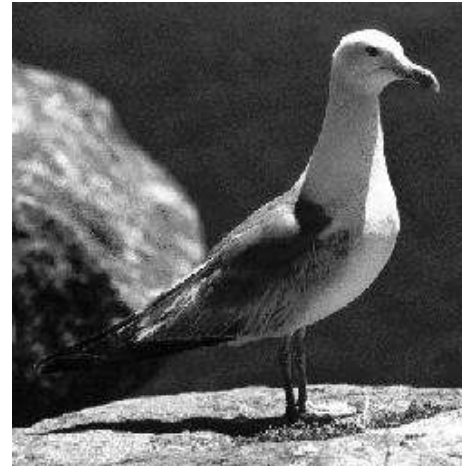
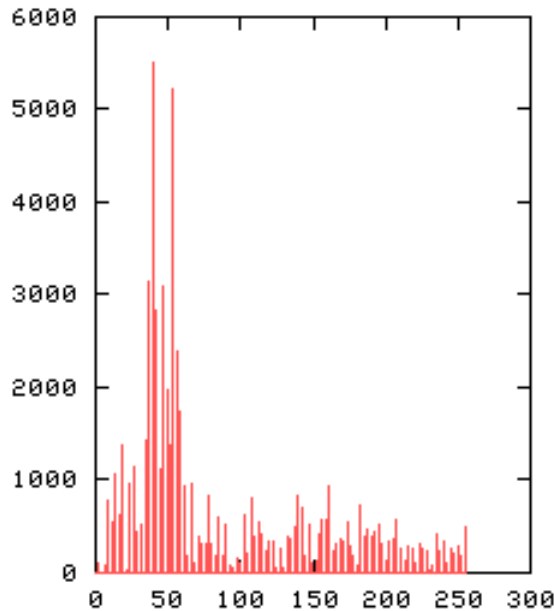
Proposed  
colorizations





# Гистограммы

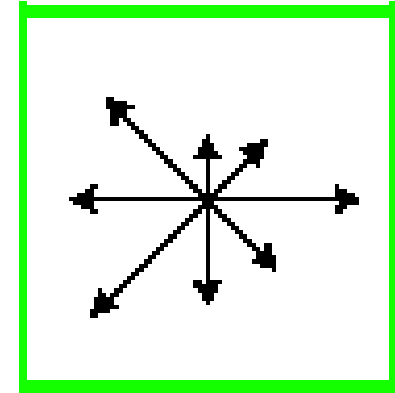
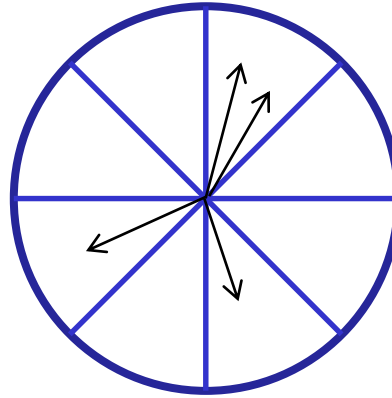
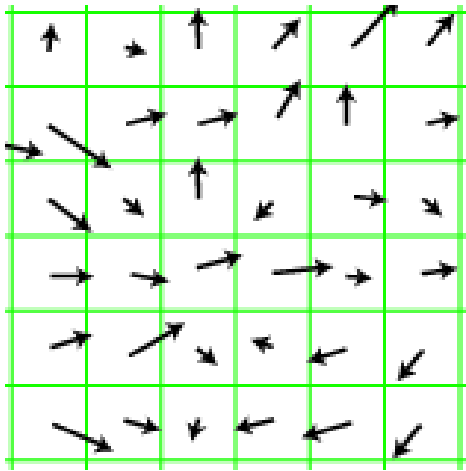
---



- Вместо использования пикселей напрямую, можем считать разные статистики распределения характеристик
  - Цвет, края, градиенты, текстуру и т.д.
- Гистограммы – стандартный способ непараметрического описания распределения признаков



# Квантование признаков

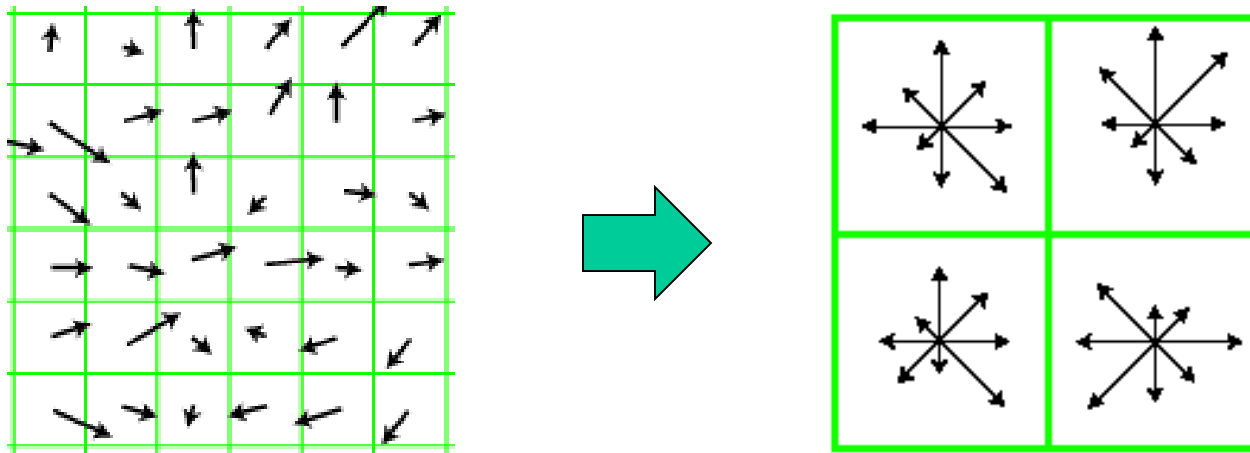


- Для некоторых признаков потребуется дискретизация (квантование)
  - Если признаки вещественные
  - Если слишком большое разрешение
- Пример для направления градиента
  - Разбиваем 360 градусов на 8 секторов
  - Каждый сектор нумеруем от 1 до 8
  - Считаем число пикселов, градиенты которых попадают в соответствующие сектора





# Гистограммы градиентов



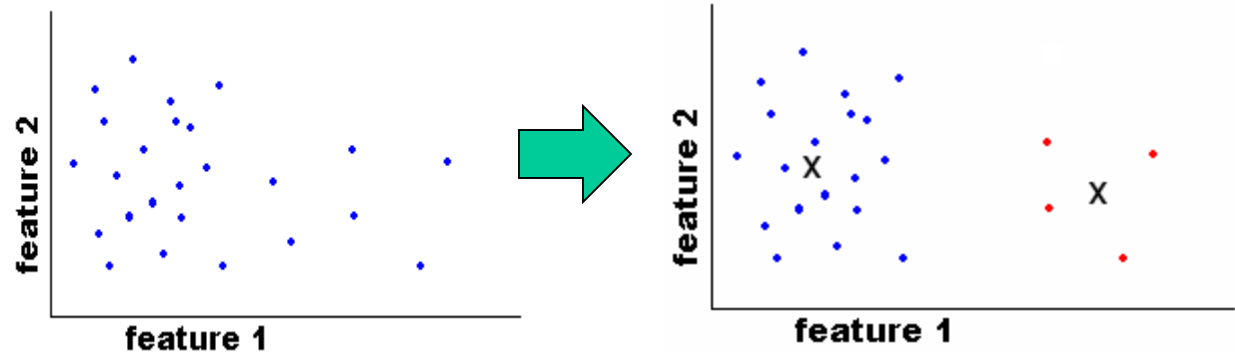
- Много названий – HOG (Histogram of Oriented Gradients), дескриптор SIFT
- Построение
  - Считаем градиент в каждом пикселе
  - Разбиваем изображение на блоки сеткой
  - В каждом блоке считаем гистограмму распределения пикселей по ориентации градиентов
  - Обычно 6-8 корзин в каждой гистограмме
- Какие достоинства у таких признаков?



# Квантование признаков



Не все цвета  
встречаются!



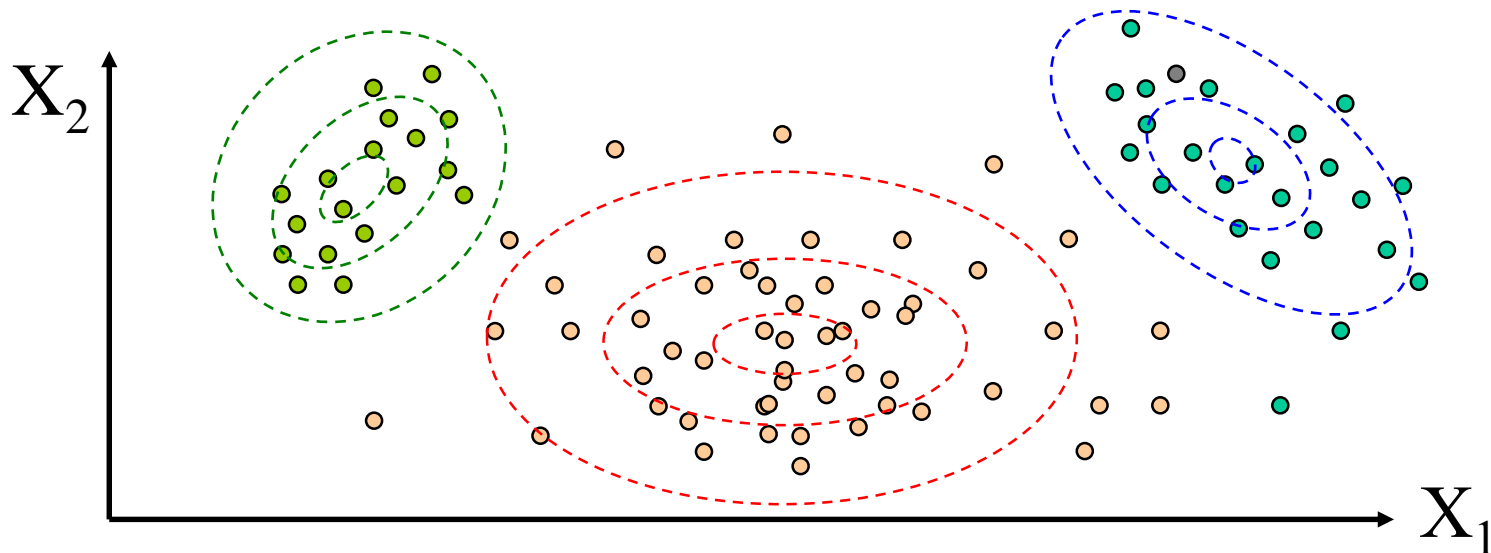
- Во многих случаях равномерное квантование по сетке не подходит
  - Высокая размерность
  - Неравномерные распределения
- Хотим разбить элементы (пр. пиксели) на часто встречающиеся группы
- Затем можем каждый элемент сопоставить своей группе (дать ему номер группы)
- Кластеризация!





# Кластеризация

- **Кластерный анализ** (Data clustering) — задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.
- Одна из задач «обучения без учителя»





# Кластеризация К-средними

---

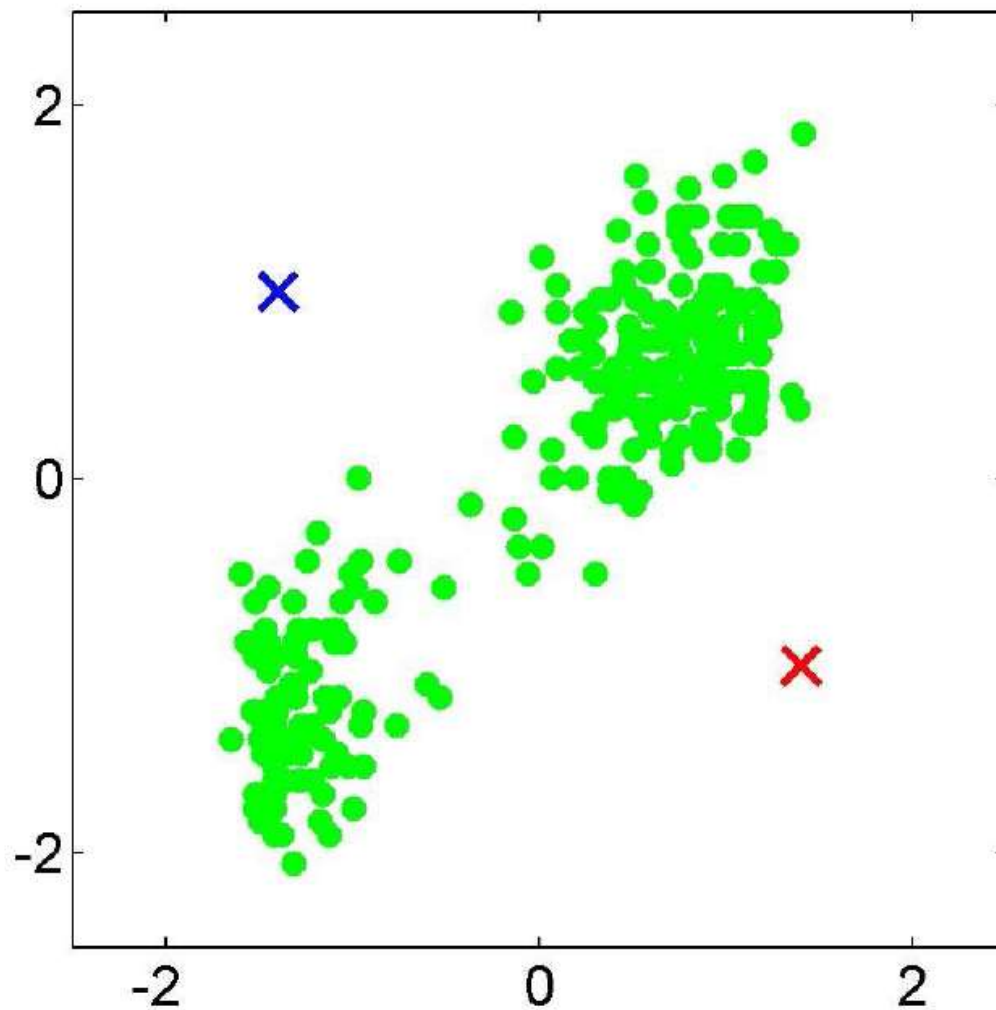
- Минимизируем сумму квадратов Евклидовых расстояний между точками  $x_i$  и ближайшими центрами кластеров  $m_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

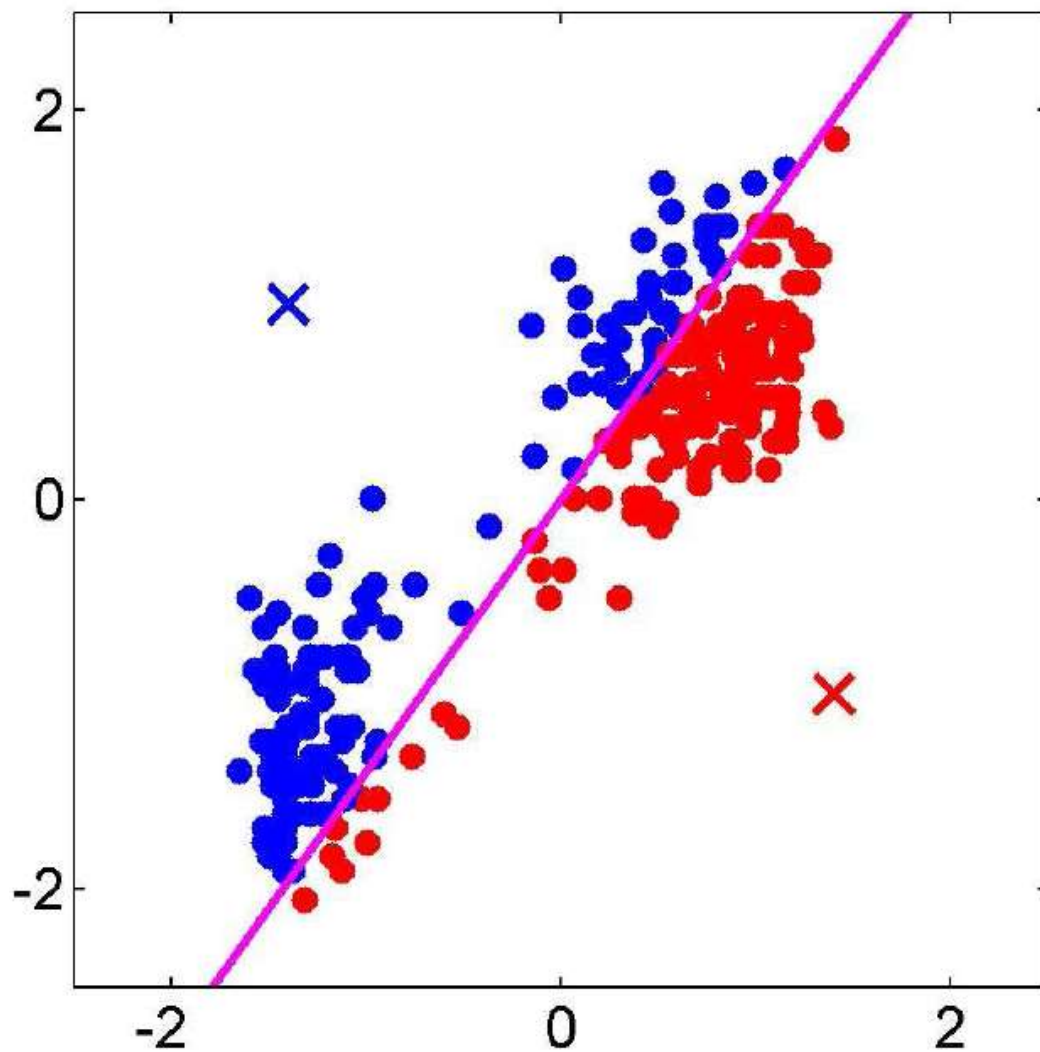
- Алгоритм:
- Случайно инициализируем К центров кластеров
- Повторяем до сходимости:
  - Назначаем каждую точку ближайшему центру
  - Пересчитываем центр каждого кластера как среднее всех назначенных точек

# Иллюстрация

---

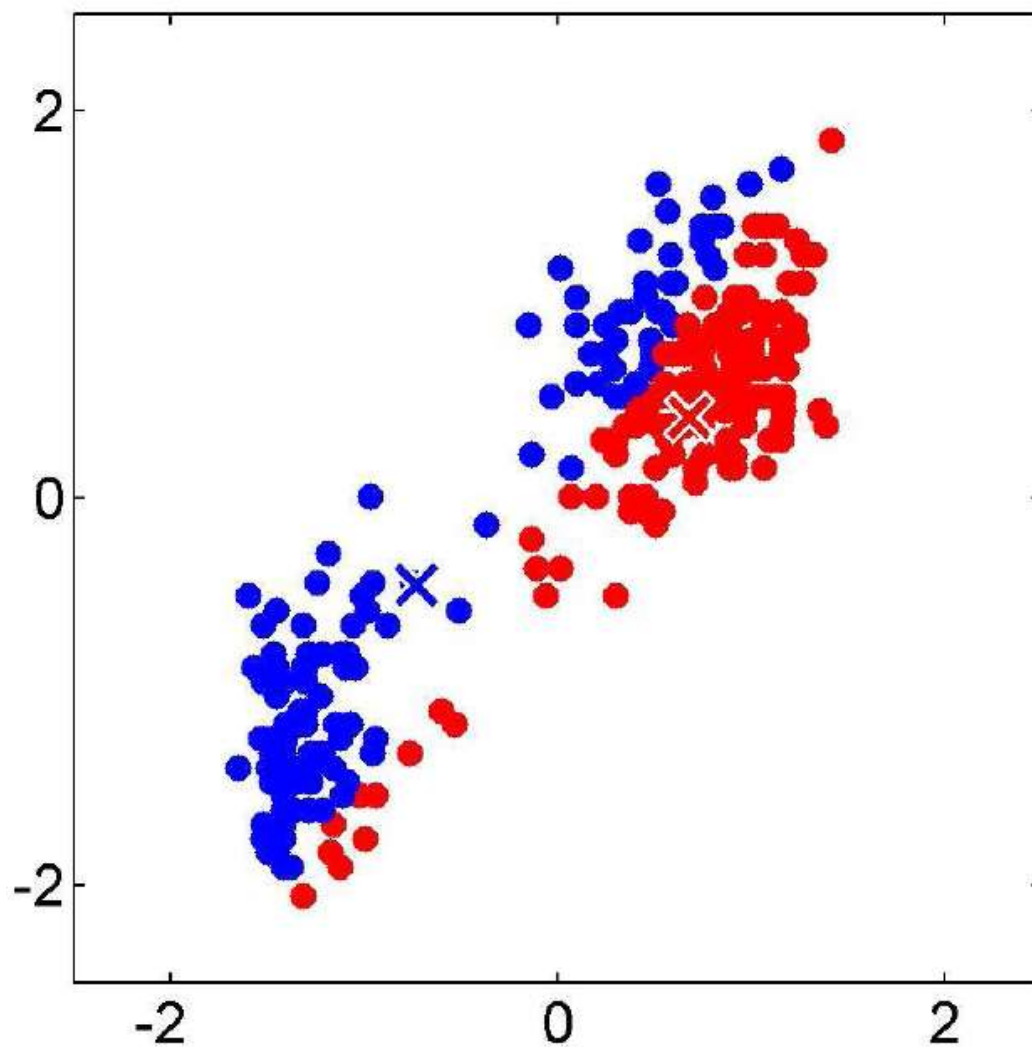


# Иллюстрация

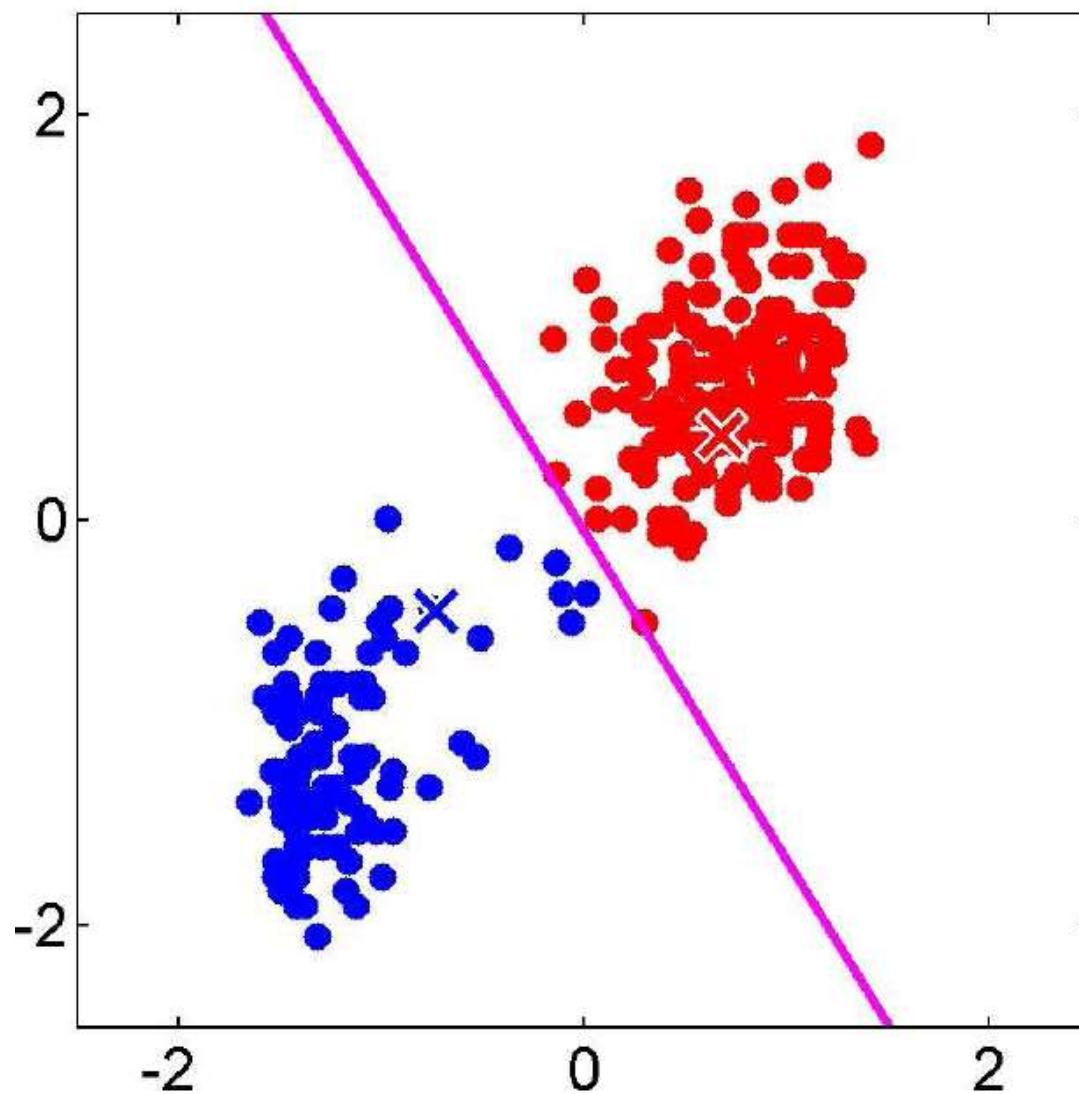


# Иллюстрация

---



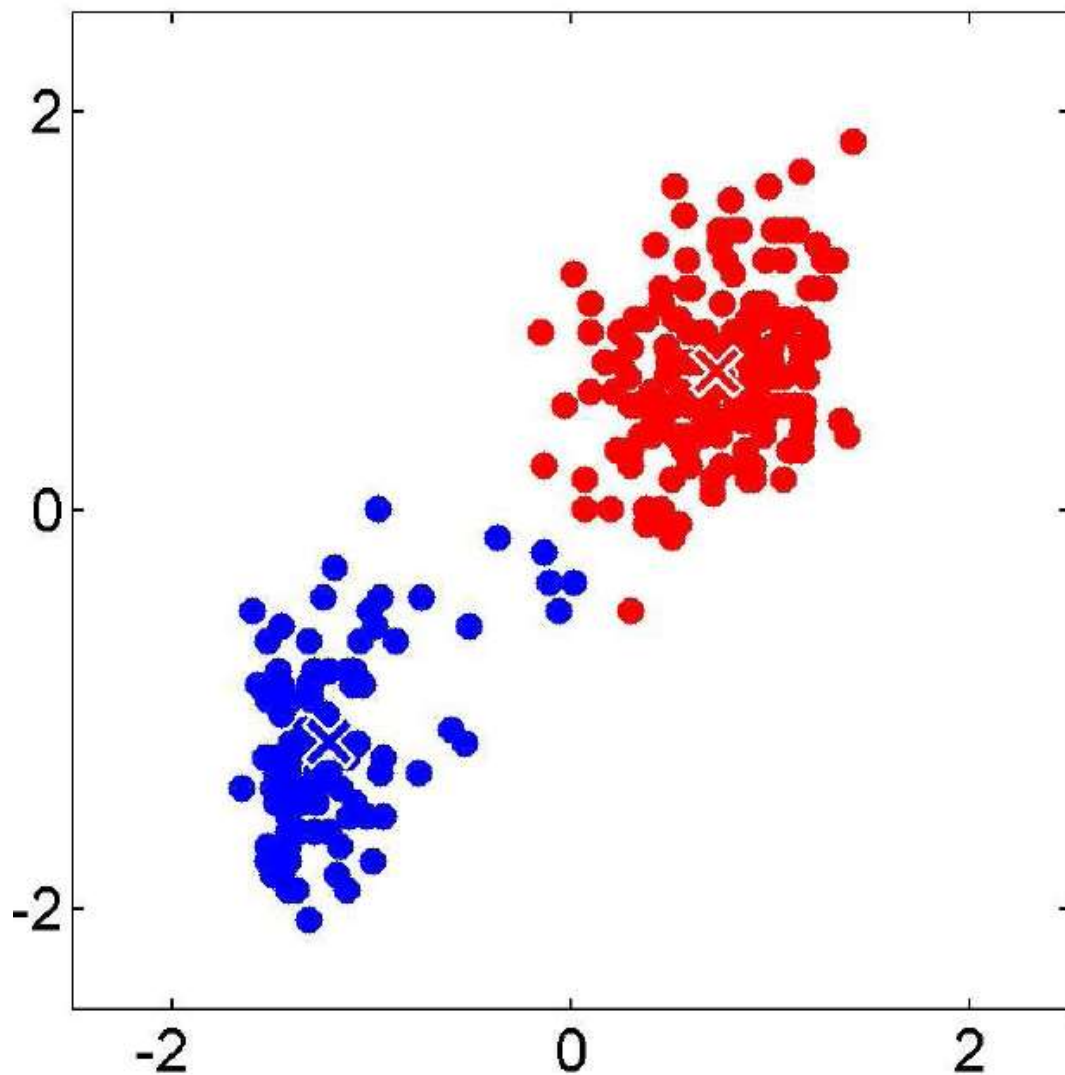
# Иллюстрация





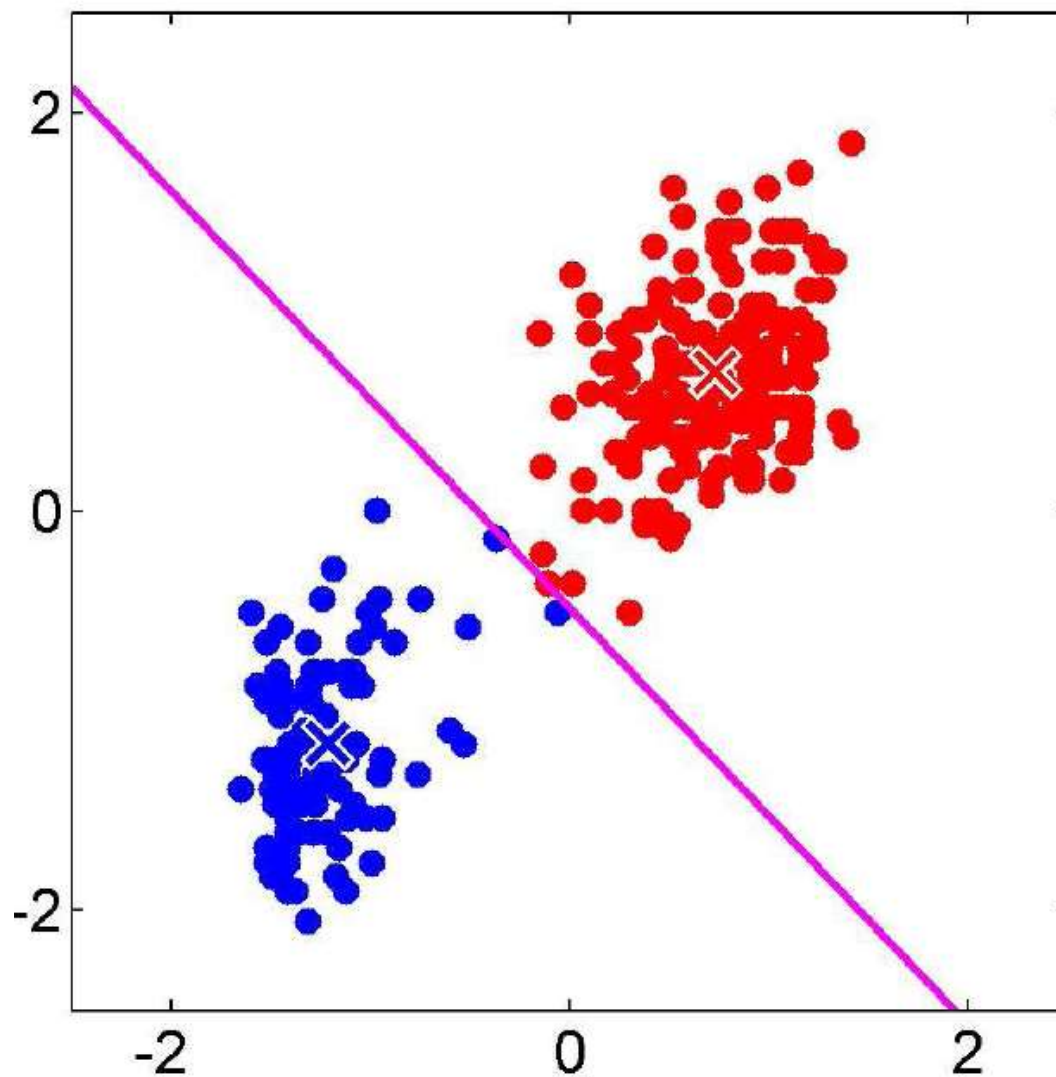
# Иллюстрация

---



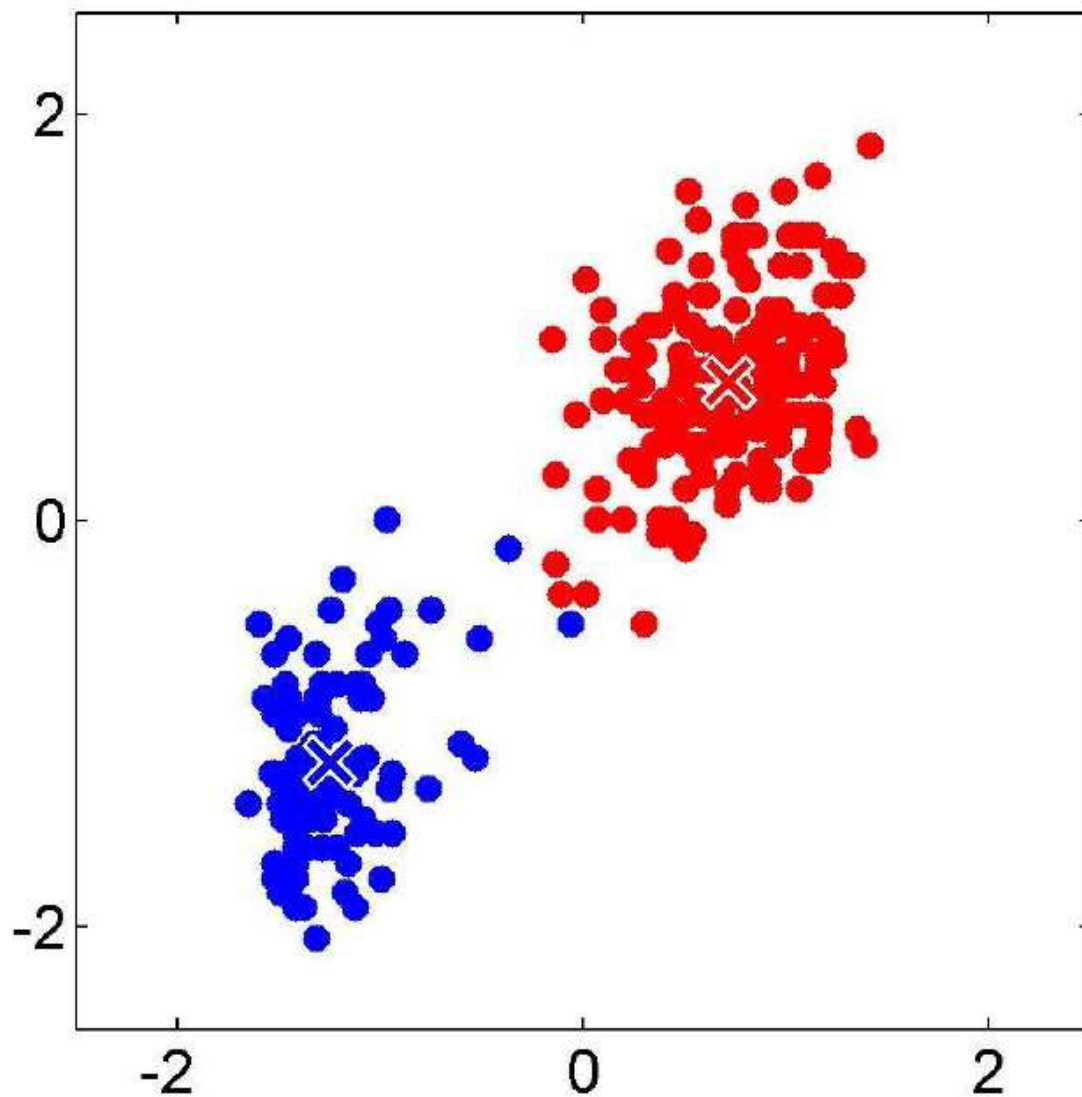
# Иллюстрация

---



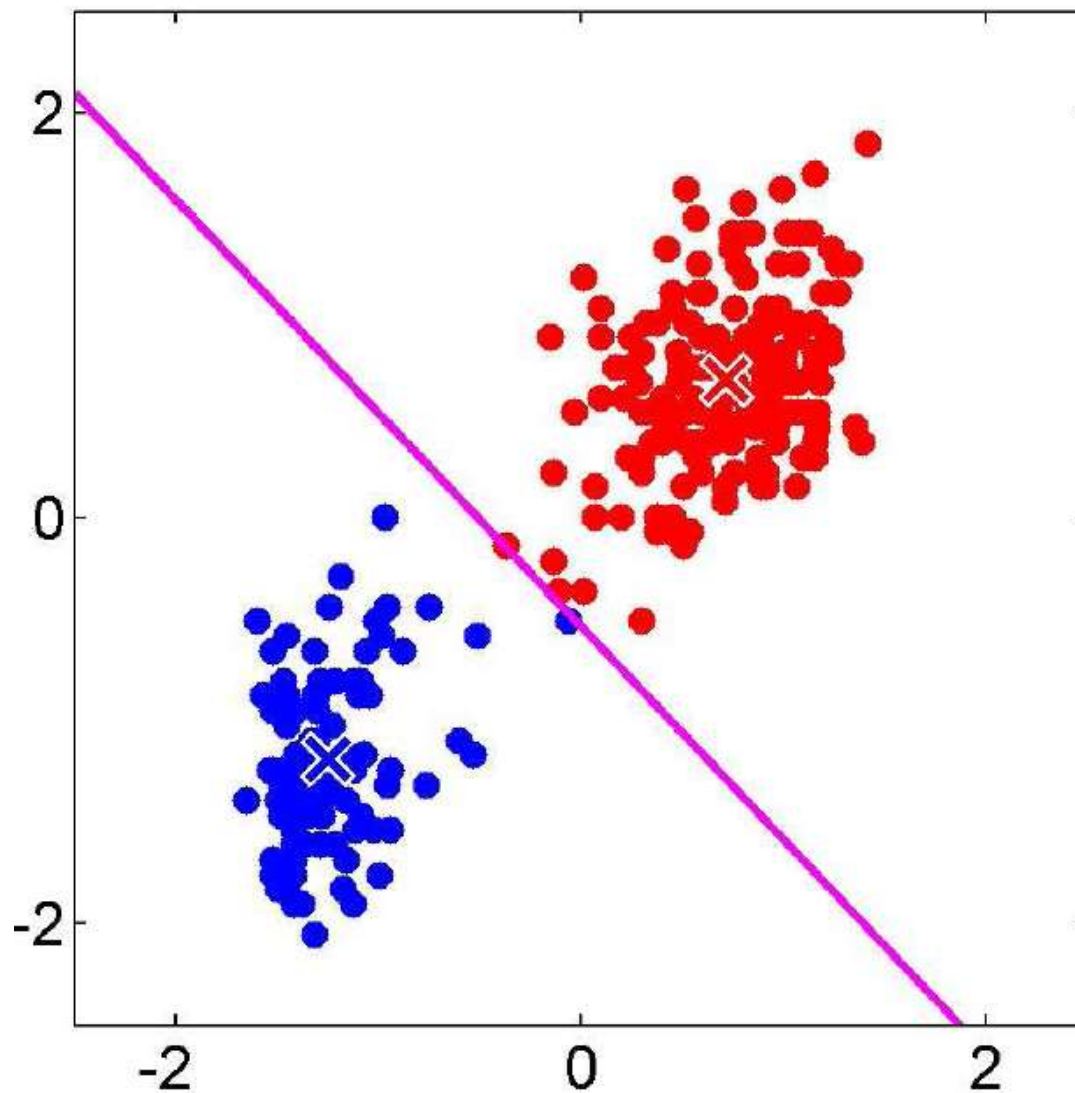
# Иллюстрация

---



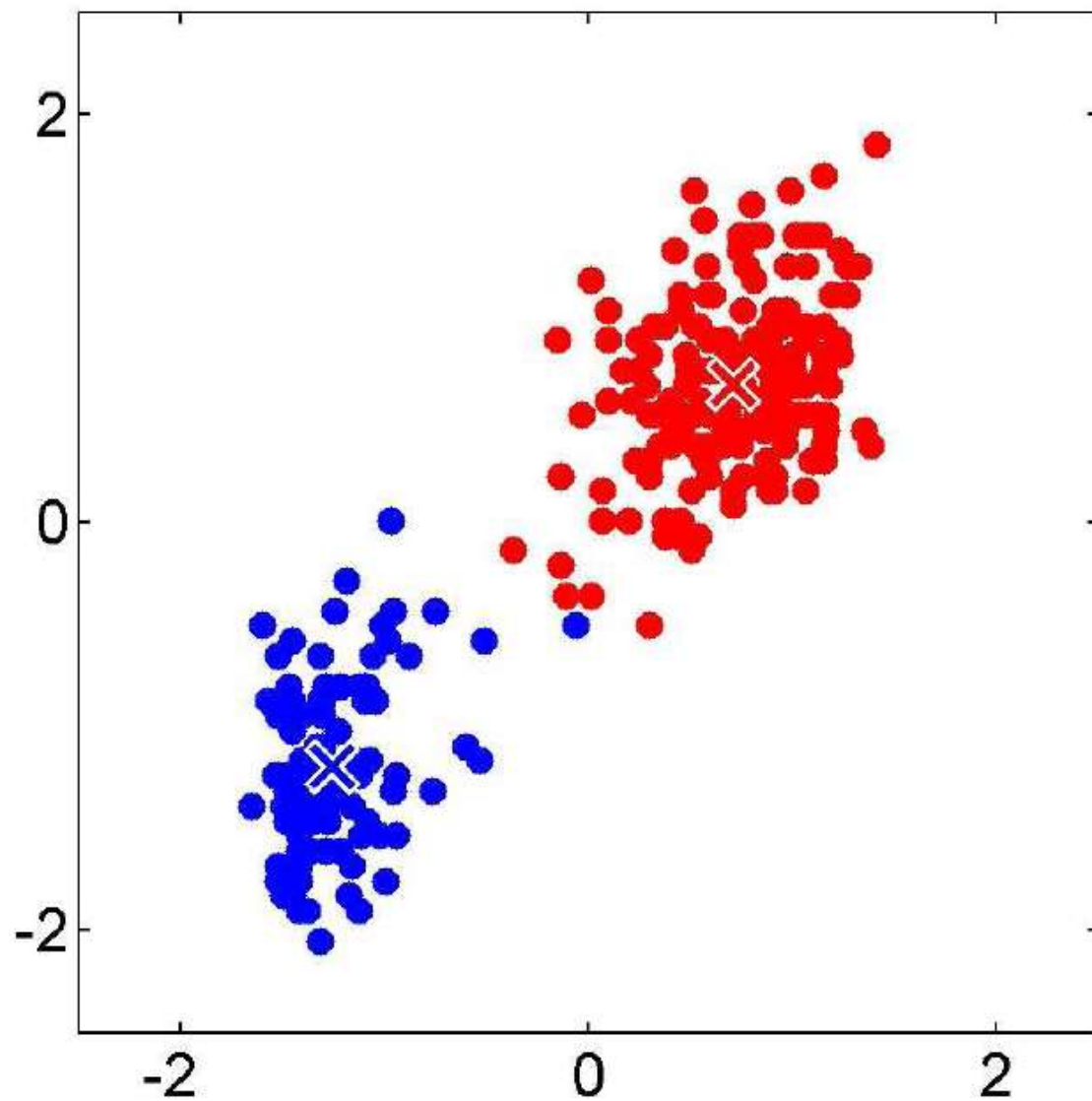
# Иллюстрация

---



# Иллюстрация

---

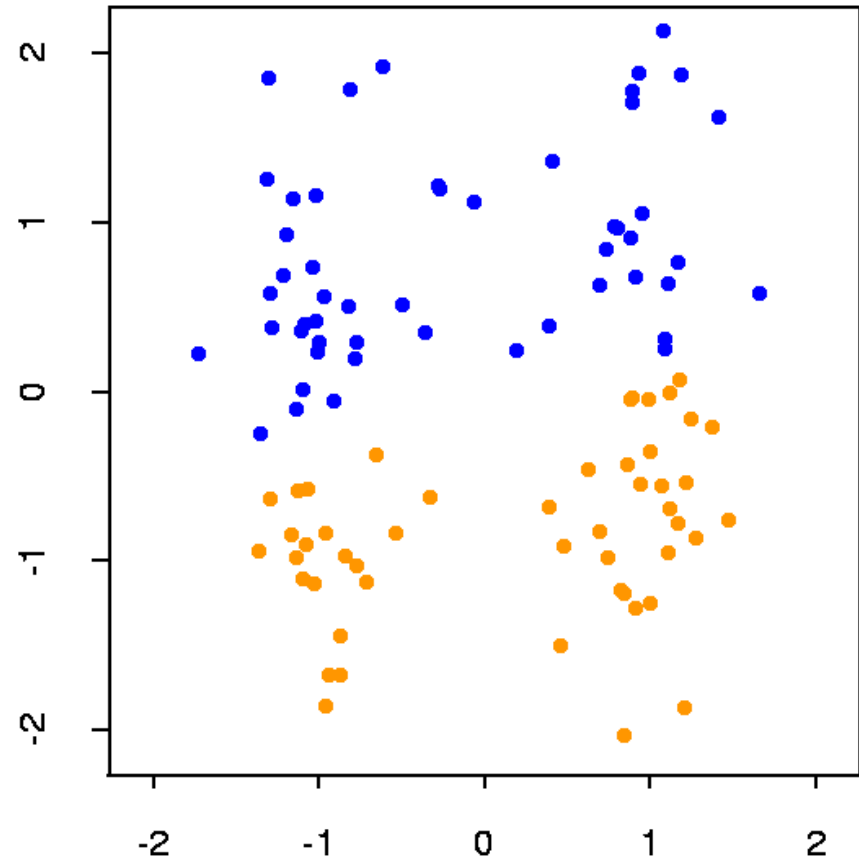




# Алгоритм К-средних

---

- Однопараметрический
  - Требуется знание только о количестве кластеров
- Рандомизирован
  - Зависит от начального приближения
- Не учитывает строение самих кластеров
- Часто применяется



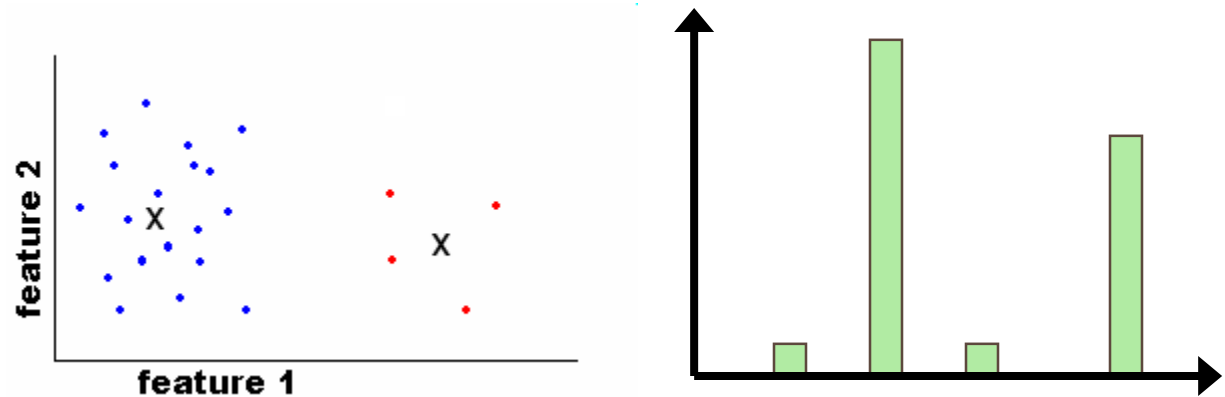




# Адаптивное квантование



Не все цвета встречаются!

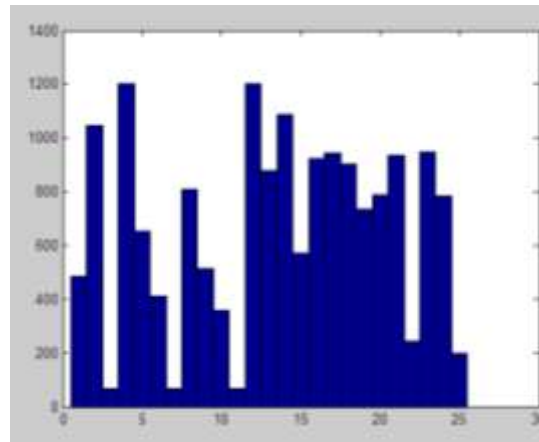


- Применим кластеризацию пикселей по цветам методом К-средних
- Теперь можем для каждого изображения посчитать гистограмму распределения пикселей изображения по найденным кластерам
- Эти гистограммы и будем далее анализировать



# Пространственное распределение

---

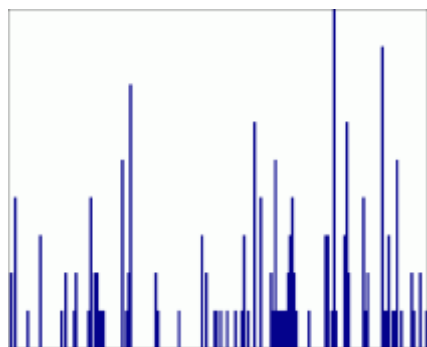


У всех этих трех изображений похожие гистограммы цветов

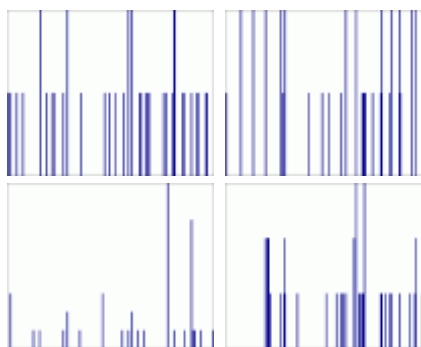


# Пространственная пирамида

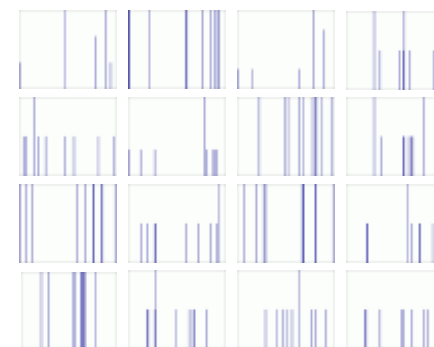
Вычислим гистограмму в каждом блоке и объединим все гистограммы в один вектор-признак



Уровень 0



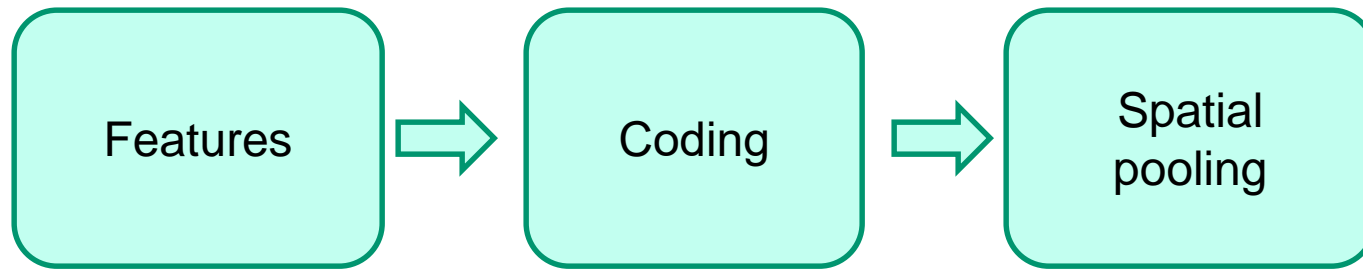
Уровень 1



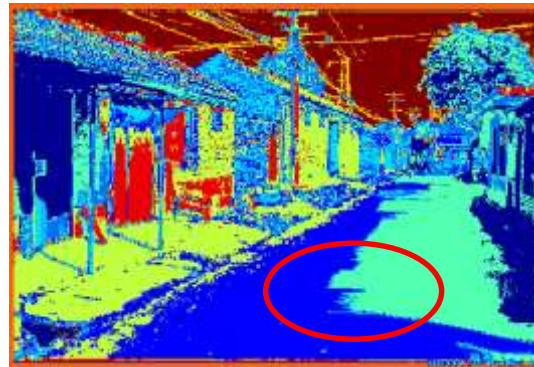
Уровень 2



# Общая схема для признаков



RGB



Квантование на 10 уровней

71%



29%



Посчитанная гистограмма

Spatial pooling (пространственная агрегация) = выбор областей для агрегации и метод агрегации



# Мешок слов

---

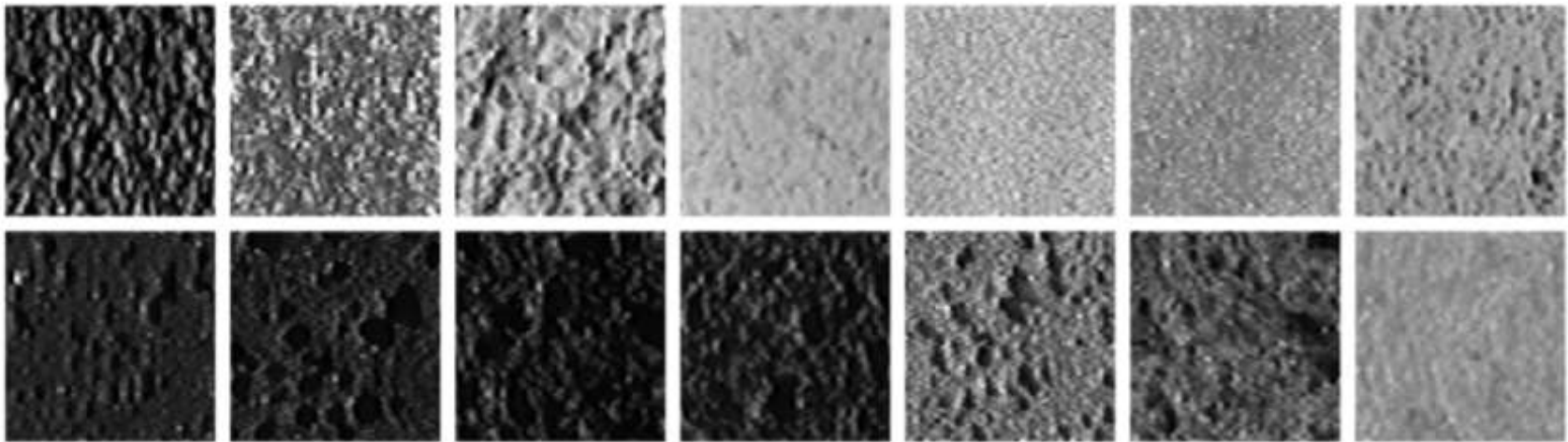




# Классификация текстур

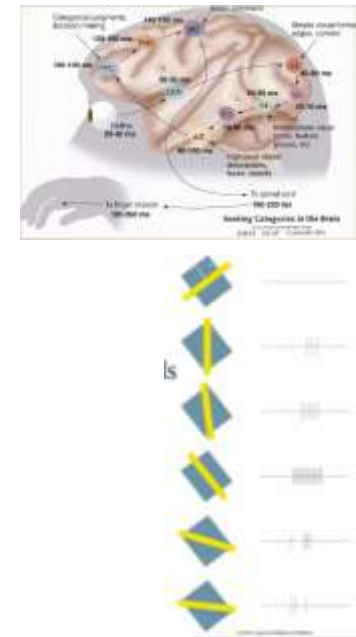
---

**Тексту́ра** — преимущественная ориентация элементов, составляющих материал



*Figure 2.* Small inter class variations between textures can make the problem harder still. In the top row, the first and the fourth image are of the same texture while all the other images, even though they look similar, belong to different classes. Similarly, in the bottom row, the images appear similar and yet there are three different texture classes present.





- Выберем фильтр, чувствительный к краю определенной ориентации
- Результат фильтрации сгладим
- Будут «подсвечены» области, содержащие текстуру с краями заданной ориентации

Pietro Perona and Jitendra Malik «Detecting and Localizing edges composed of steps, peaks and roofs», ICCV 1990



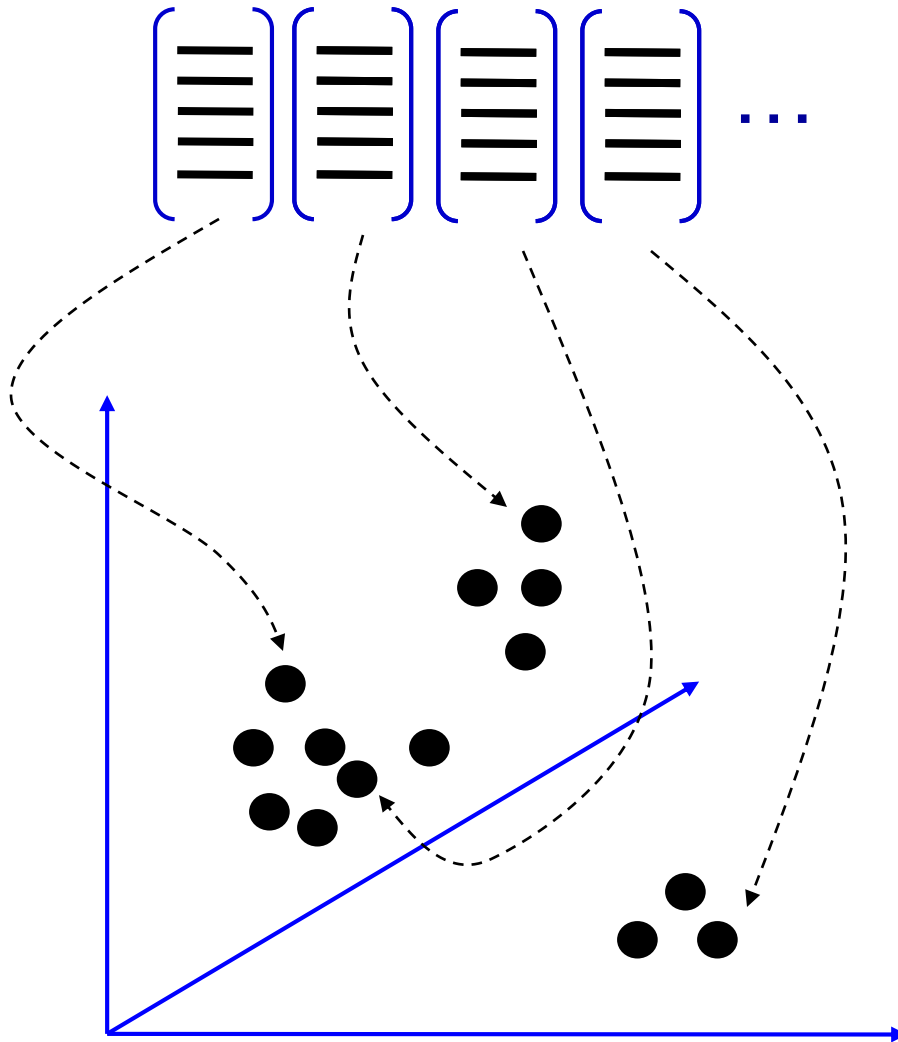
# Банки фильтров и признаки текстуры

- Возьмём теперь несколько фильтров разного масштаба и ориентации
- Такой набор называют «банк фильтров»
- Каждый пиксель изображения после обработки банком фильтров даёт вектор признаков
- Этот вектор признаков эффективно описывает локальную текстуру окрестности пикселя



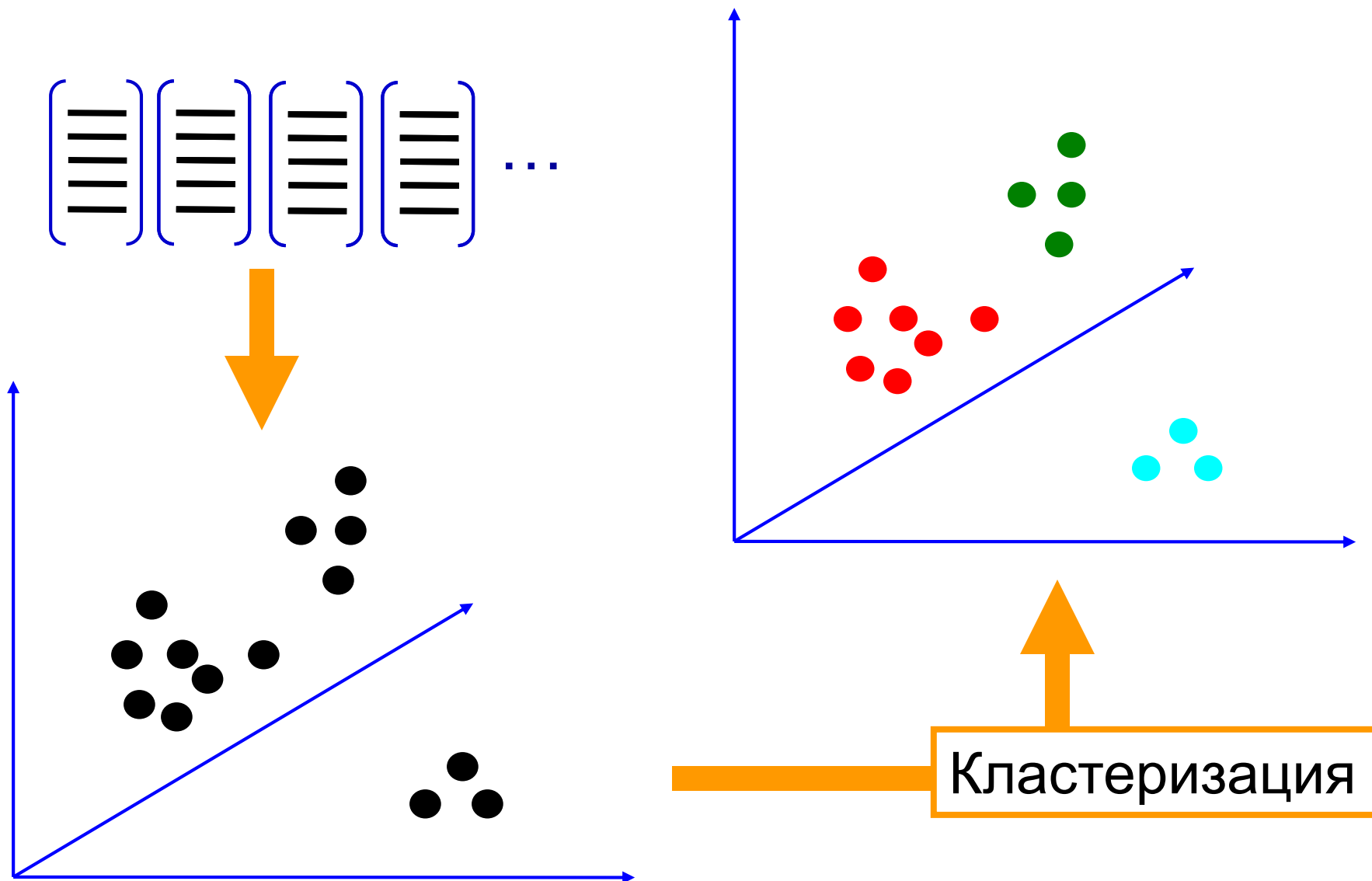
- Как нам построить «словарь» элементов текстуры?

# Как построить словарь?

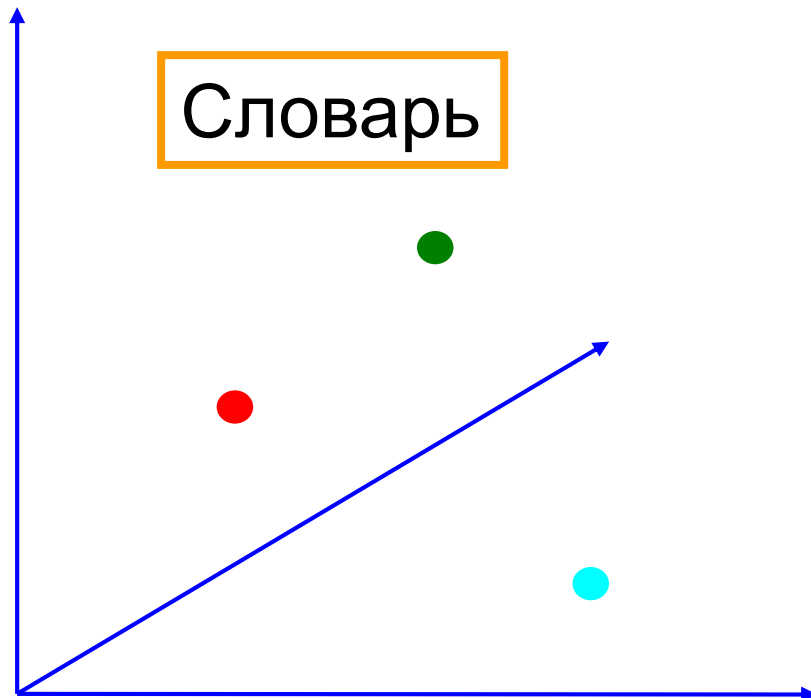
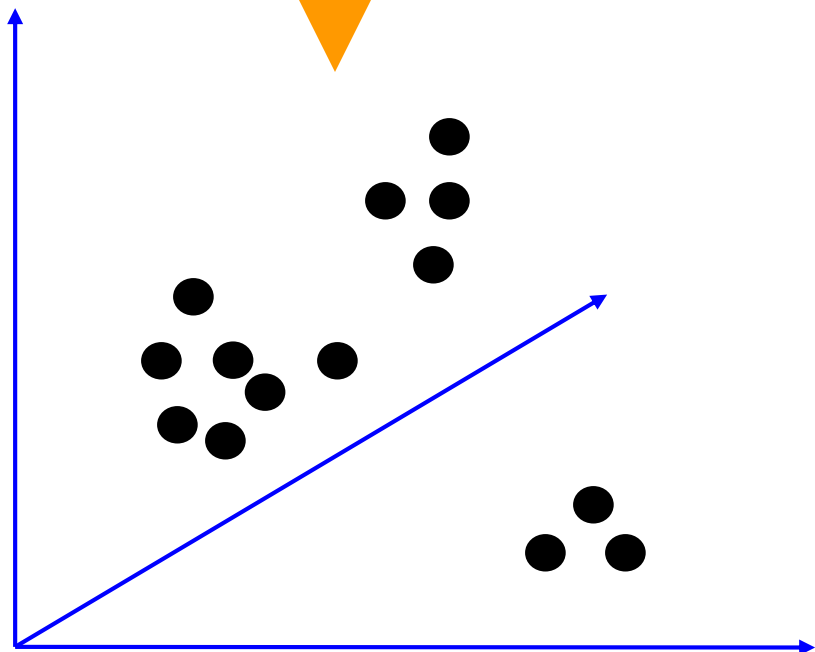
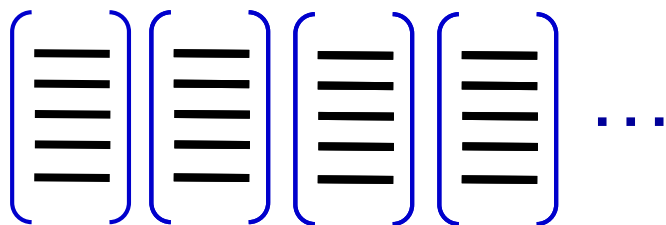




# Обучение словаря



# Обучение словаря



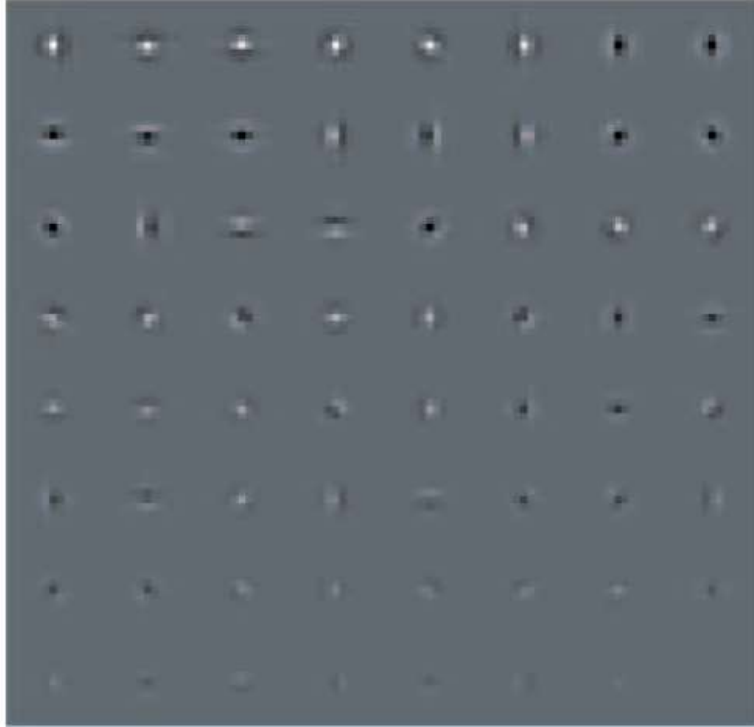
Кластеризация



# Текстоны

---

«Текстон» = характерный фрагмент текстуры изображения



Часть текстонов из словаря



Изображение



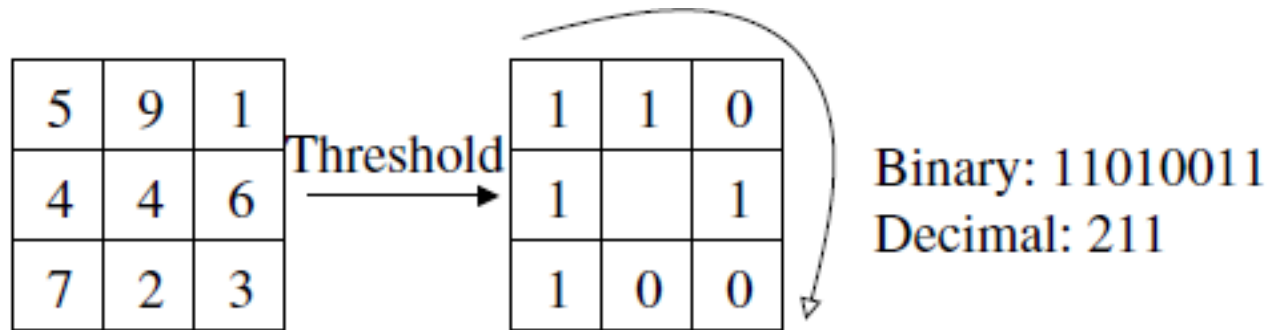
Карта текстонов

Текстуру фрагмента изображения можно описать гистограммой частот текстонов



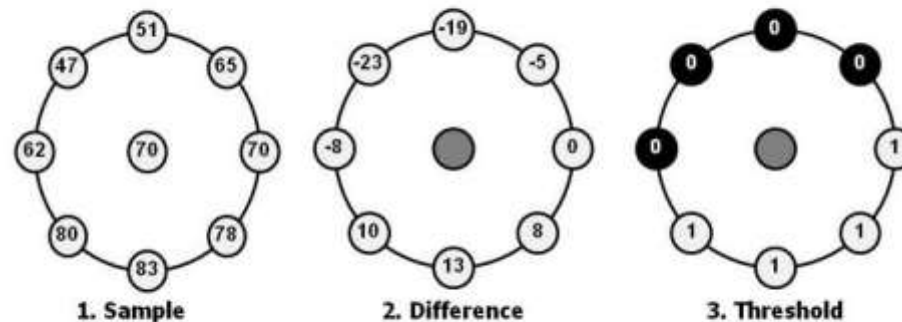


# Local Binary Patterns



The value of the LBP code of a pixel  $(x_c, y_c)$  is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad s(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$



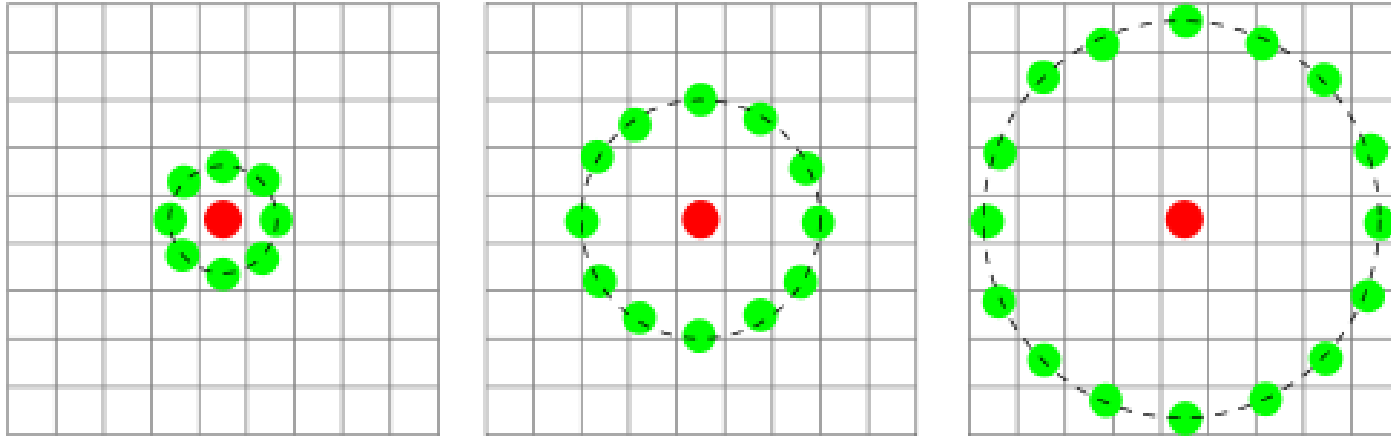
$$1*1 + 1*2 + 1*4 + 1*8 + 0*16 + 0*32 + 0*64 + 0*128 = 15$$

4. Multiply by powers of two and sum



# Расчёт LBP

---

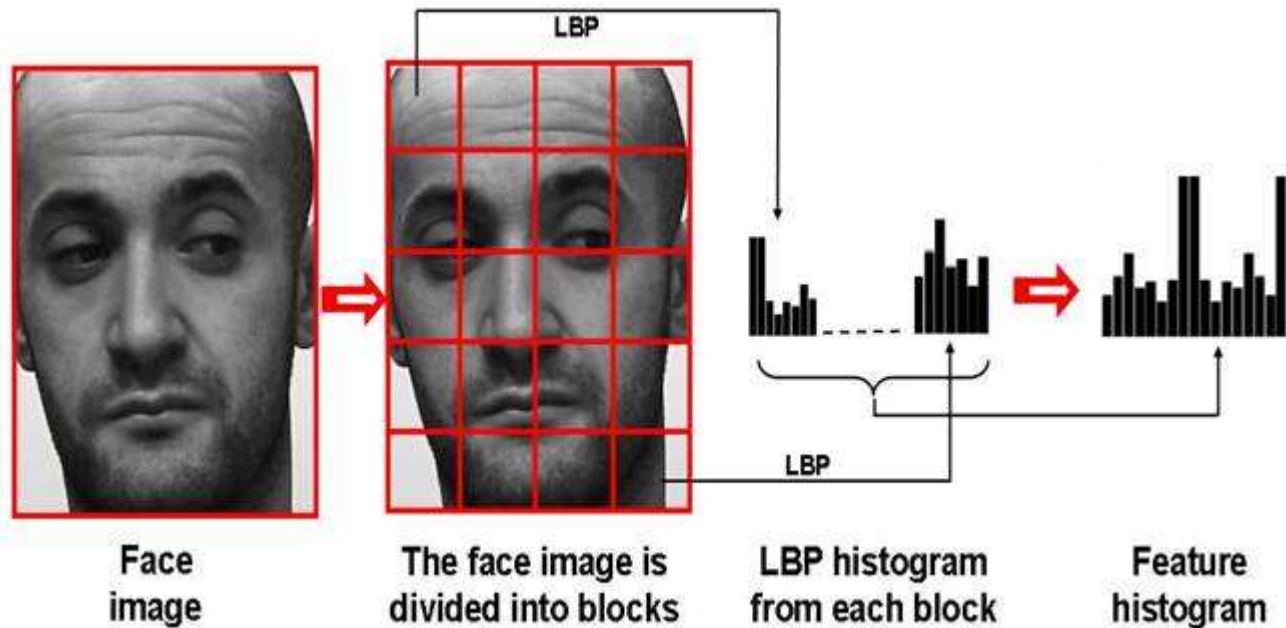


Может потребоваться интерполировать значения в точках, необходимых для расчёта LBP

Мы можем использовать LBP код как номер визуального слова в фиксированном словаре



# Применение LBP

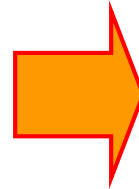


- Изображение разбивается на области. В каждой области применяются LBP операторы к каждому пикселю. Строится гистограмма.
- Объединение гистограмм – LBP дескриптор для изображения.
- Для пары изображений считается разность дескрипторов по какой-нибудь метрике (например, Хи-квадрат)

Ahonen, T., Hadid, A. and Pietikäinen, M. (2006), Face Description with Local Binary Patterns: Application to Face Recognition. IEEE PAMI 28(12):2037-2041.



# «Визуальные слова»



- «Визуальное слово» – часто повторяющийся фрагмент изображения («visual word»)
- В изображении визуальное слово может встречаться только один раз, может ни разу, может много раз
- Чем отличается от текстов?



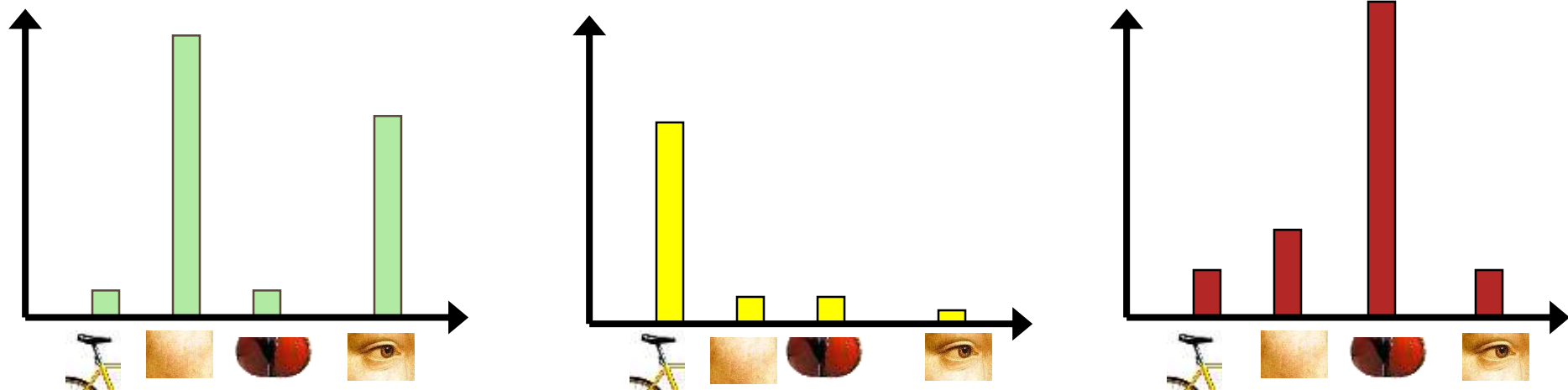
# Схема метода «мешок слов»

## 1. Построение словаря

1. Извлечение фрагментов из обучающей выборки
2. Кластеризация и построение «визуального» словаря

## 2. Построение дескрипторов

1. Извлечение фрагментов из изображения
2. Квантование фрагментов по словарю
3. Построение гистограммы частот визуальных слов

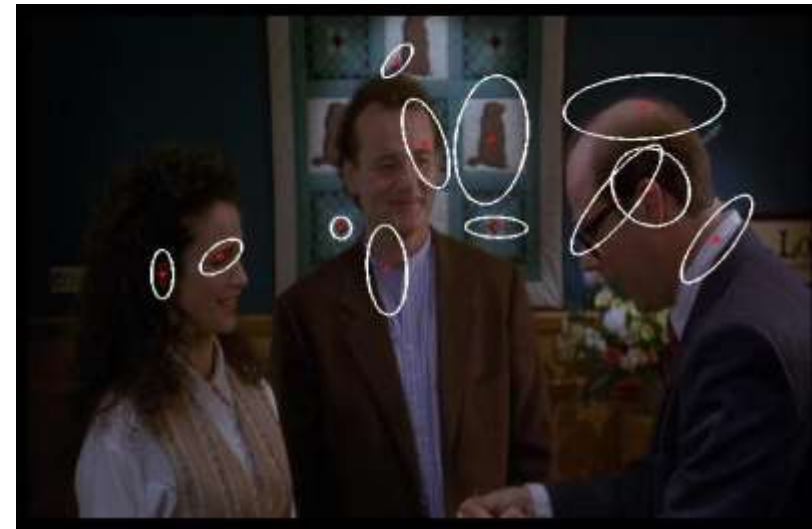
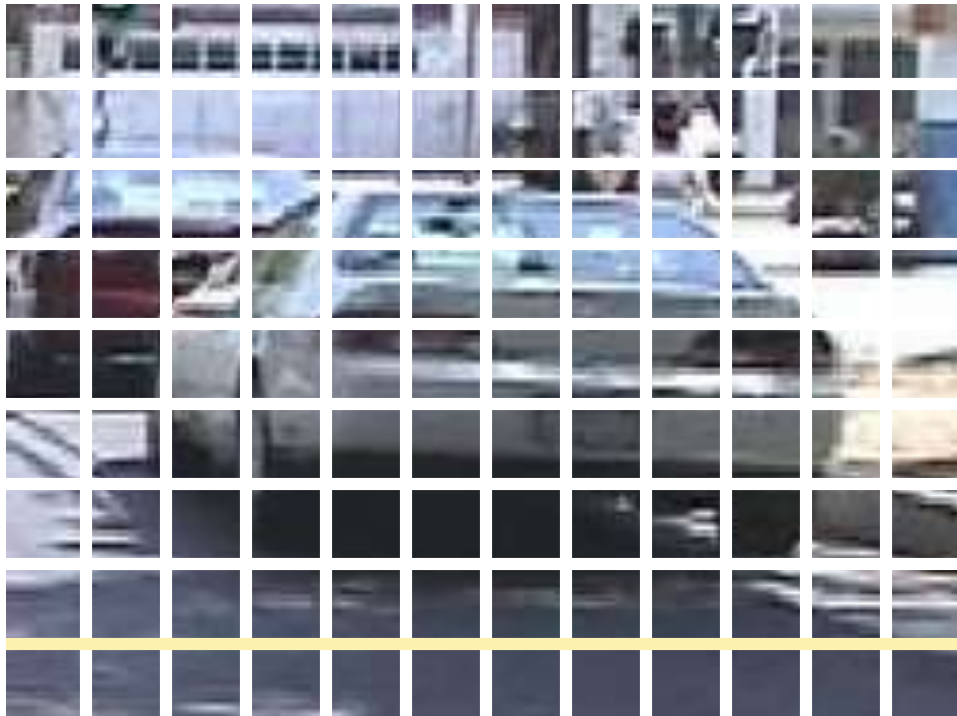




# 1. Извлечение фрагментов

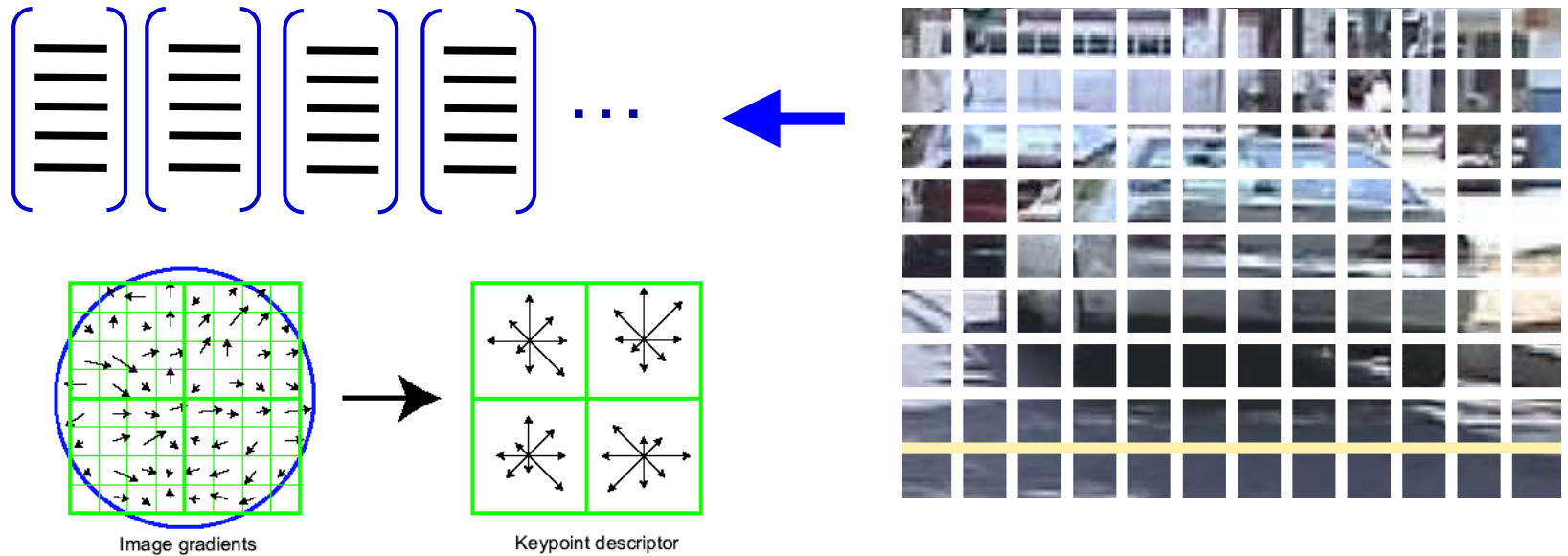
---

- По регулярной сетке
  - Можем без перекрытия, можем с перекрытием
- Вокруг характерных точек
  - Точки, которые можно устойчиво идентифицировать на изображении после деформации (рассматриваем отдельно)





# 1. Вычисление признаков фрагментов

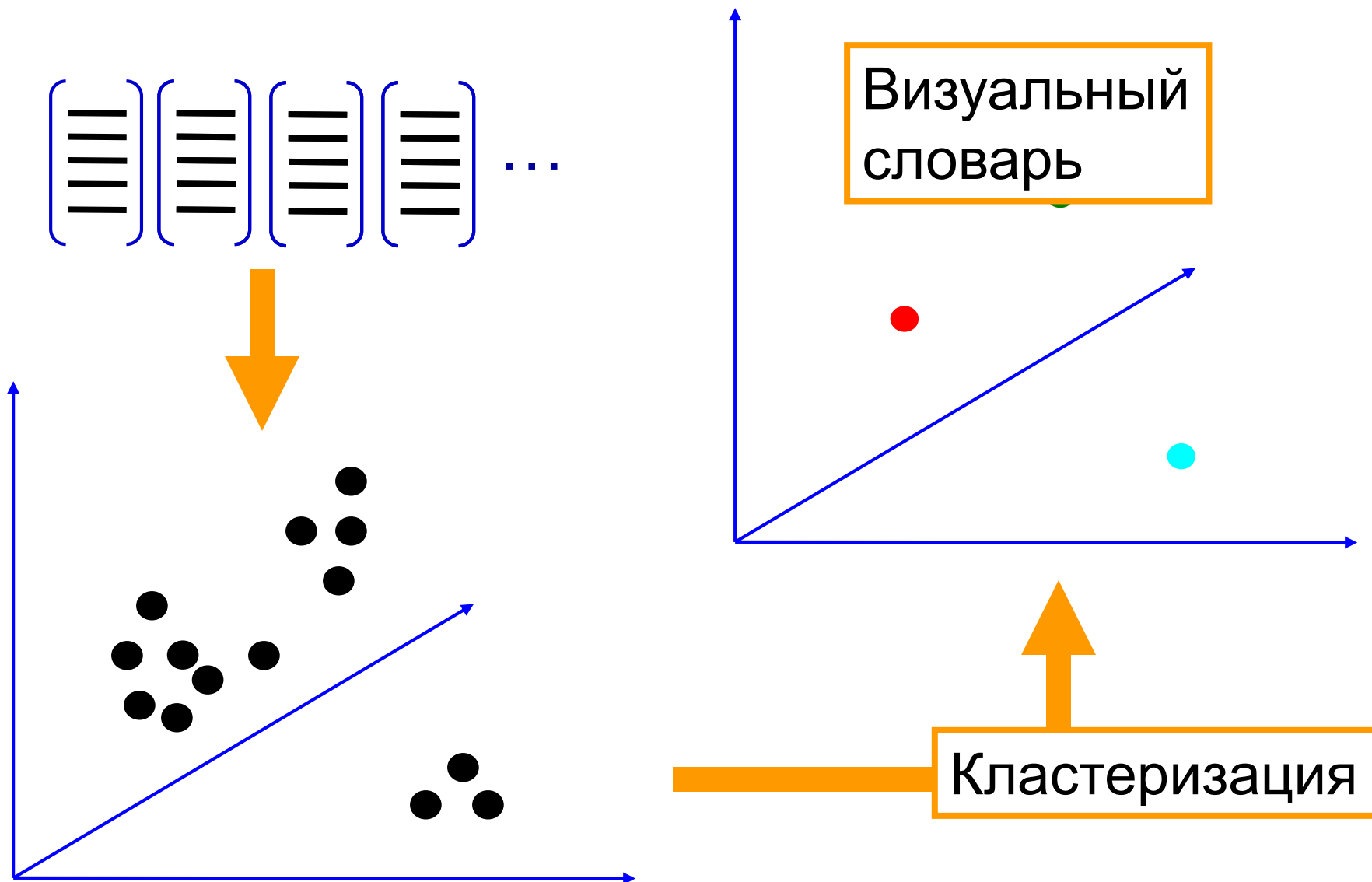


- Неупорядоченный список фрагментов из множества изображений
- Нужно описать каждый фрагмент вектор-признаком
- Например, с помощью SIFT





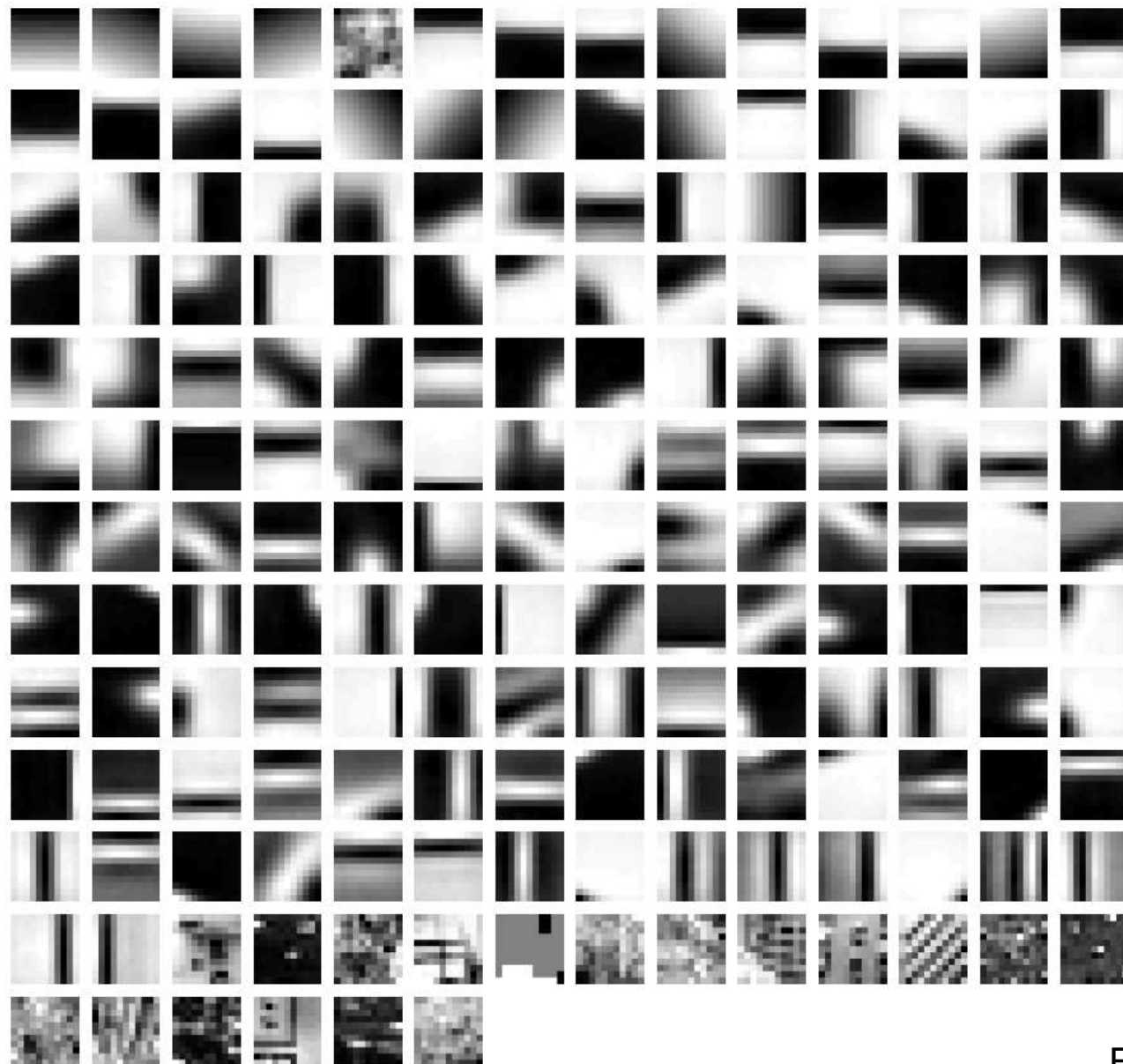
## 2. Обучение словаря через кластеризацию





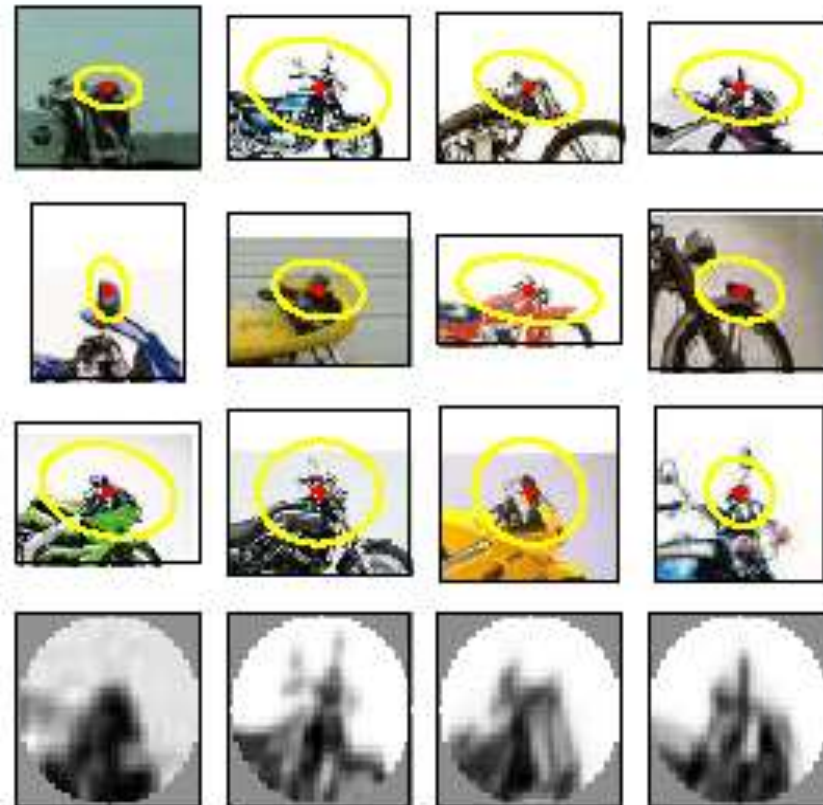
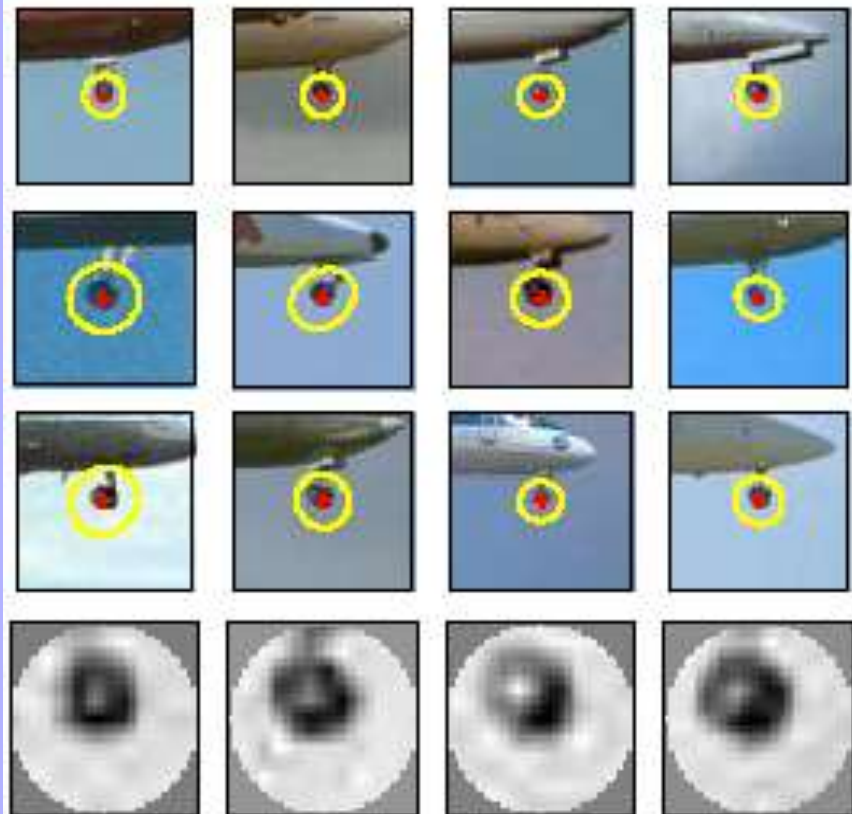
# Пример словаря

---





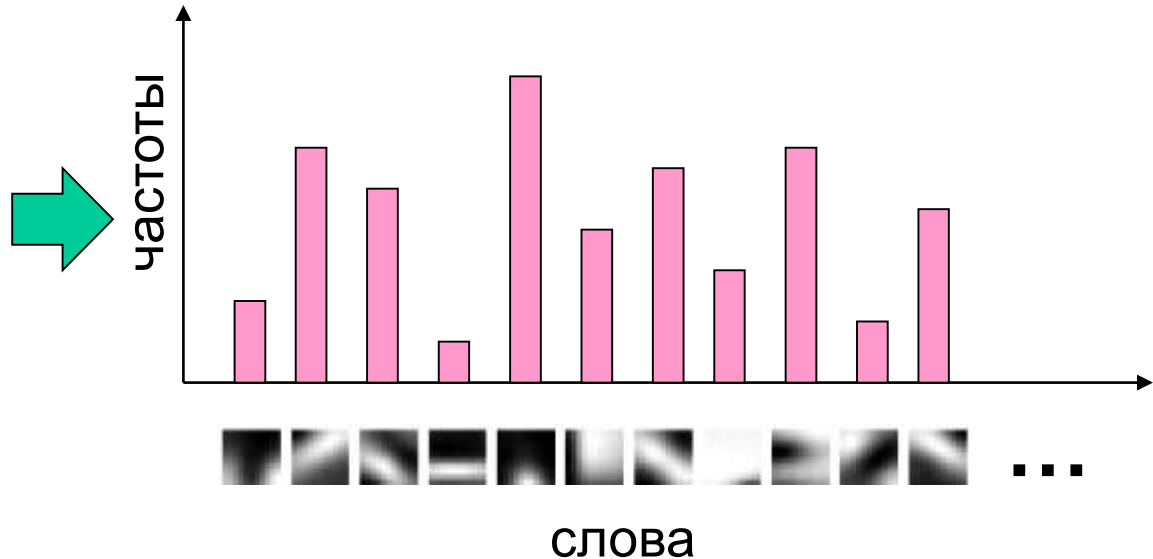
# Примеры визуальных слов



Построены по характерным точкам



### 3. Построение «мешка слов»



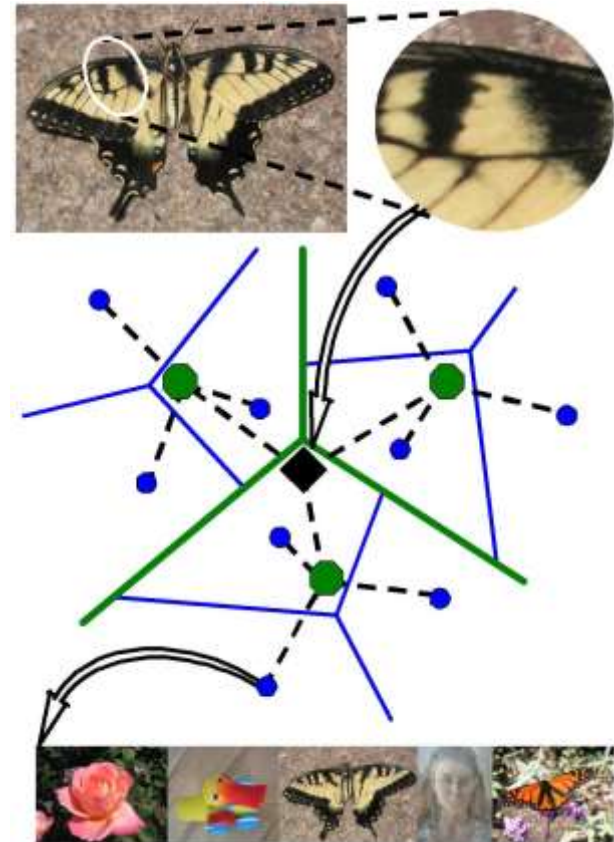
- Соберём тем же способом фрагменты для выбранного изображения
- «Квантуем» каждый фрагмент
  - Определим, какому слову из словаря он соответствует (фактически, сопоставим номер от 1 до  $N$ , где  $N$  – размер словаря)
- Построим гистограмму частот фрагментов для изображения



# Визуальные словари

---

- Как выбрать размер словаря?
  - Маленький: слова не могут описать все особенности
  - Большой: переобучение
- Вычислительная сложность
  - Деревья словарей (Nister & Stewenius, 2006)
  - Приближенные методы
  - Хеширование





# Резюме мешка слов

---

- Идея «мешка слов» оказалась очень интересной и эффективной в применении к картинкам
- Началось с описания текстуры, и затем перешли к «визуальным» словам
- Мешок слов используется как сам по себе, так и совместно с другими признаками изображения
- Часто используются совместно и разреженная и плотная версии:
  - Мешок слов по характеристическим точкам
    - Bags of visual words
  - Плотные слова
    - Dense words (PhowGray, PhowColor)
- Реализация – библиотека VLFeat <http://www.vlfeat.org/>

# Эталонные коллекции

---



[http://www.vision.caltech.edu/Image\\_Datasets/Caltech101](http://www.vision.caltech.edu/Image_Datasets/Caltech101)

<http://pascallin.ecs.soton.ac.uk/challenges/VOC/>

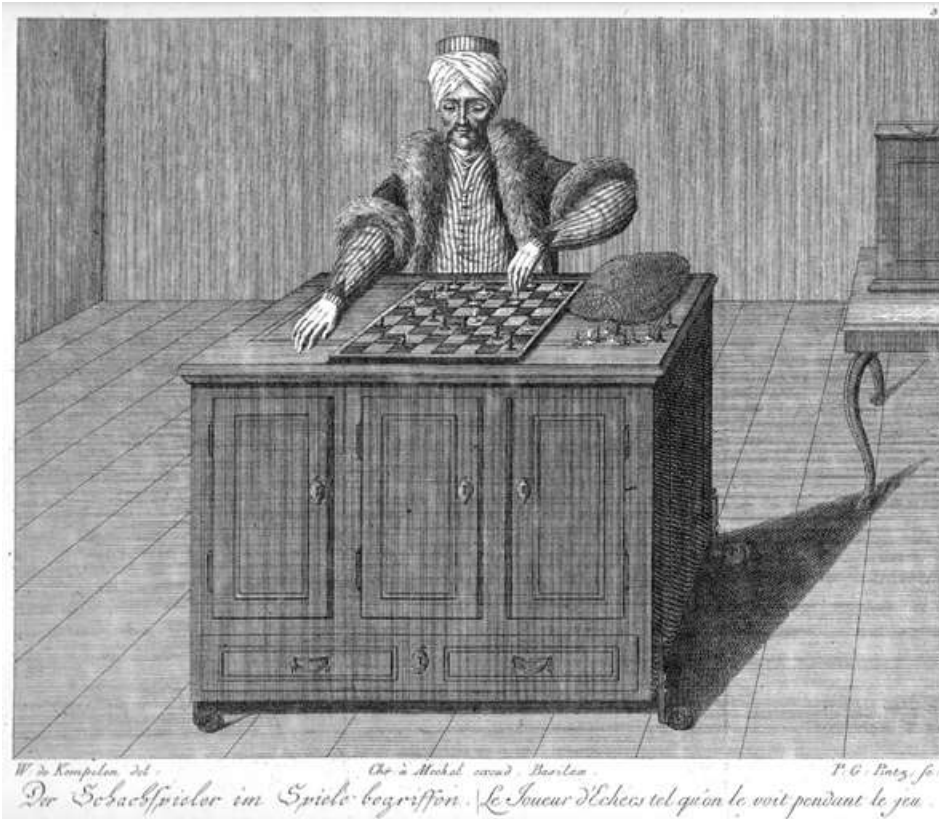
Как сейчас получают эталонные коллекции?

1. Берут готовые
2. Размечают вручную
3. Сопоставление имеющихся коллекций для получения доп. разметки





# Mechanical Turk (1770)



- Automaton Chess Player – робот, игравший в шахматы
  - Автоматон двигает фигуры, говорит «Чек» и обыгрывает всех!
- С 1770 по 1854 развлекал публику, только в 1820 году раскрыли обман



# Galaxy Zoo

---

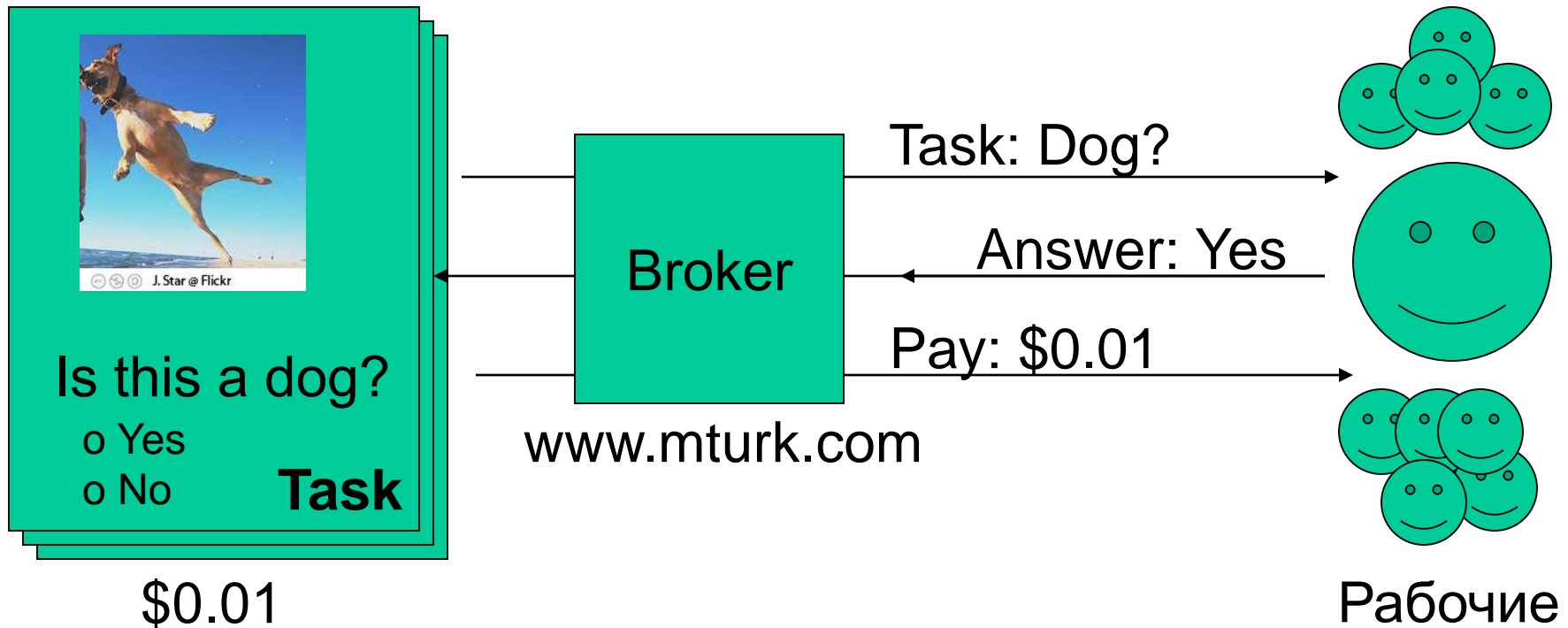


<http://www.galaxyzoo.org/>

- Классификация изображений галактик
- Первый масштабный проект такого рода
- Более 150000 волонтеров за первый год бесплатно сделали более 60 млн. меток



# Фрилансеры



- Возникли интернет-брокеры для сведения заказчиков и работников
- Пример: Amazon Mechanical Turk, Яндекс Толока
- Разметчики – просто подработка (студенты, домохозяйки), люди из более бедных стран















# Tiny images (80 млн.)

6919) Mohammed ali

Hide definition

Click on the image if you think it is correct (a green frame will appear), and twice if it is wrong. For images that you are not sure leave the black frame around the image.



Click to submit selection



Next ▶

**Definition** (from Wordnet)

Mohammed Ali, Mehemet Ali, Muhammad Ali -- (Albanian soldier in the service of Turkey who was made viceroy of Egypt and took control away from the Ottoman Empire and established Egypt as a modern state (1769-1849))

**Wikipedia:** open wikipedia page

**Average Image**



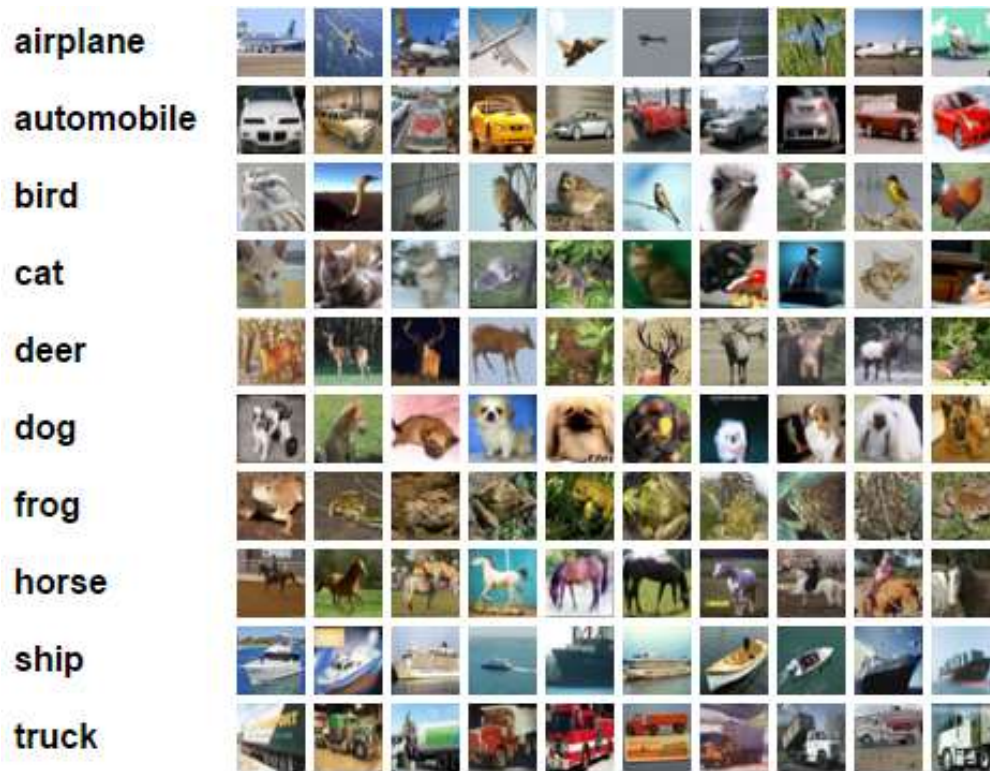
<http://people.csail.mit.edu/torralba/tinyimages/>





# CIFAR-10 и CIFAR-100

## Выборки из TinyPictures



- **CIFAR-10**
  - 10 классов
  - 60000 изображений
  - 5000 обучающих и 1000 тестовых на класс
- **CIFAR-100**
  - 100 классов
  - 60000 изображений
  - 500 обучающих и 100 тестовых на класс

<http://www.cs.toronto.edu/~kriz/cifar.htm>

|



Цель: собрать коллекцию с 1000 изображений на каждый из 117 тыс. synsets.

Текущая статистика:

- Total number of non-empty synsets: 21841
- Total number of images: 14,197,122
- Number of images with bounding box annotations: 1,034,908
- Number of synsets with SIFT features: 1000
- Number of images with SIFT features: 1.2 million

Конкурс large-scale image recognition

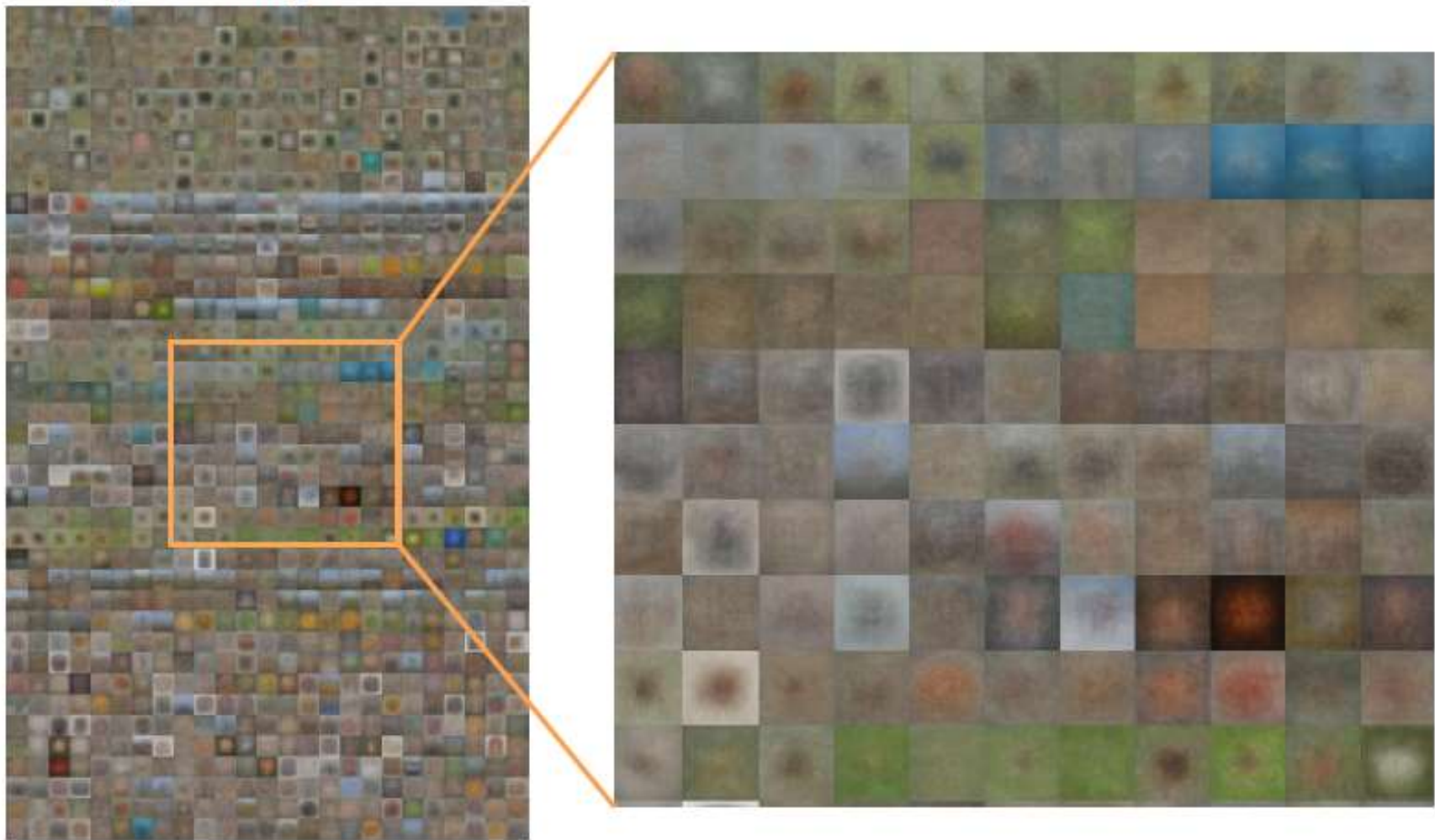
<http://www.image-net.org>



# Изображения в среднем

---

Average Test images







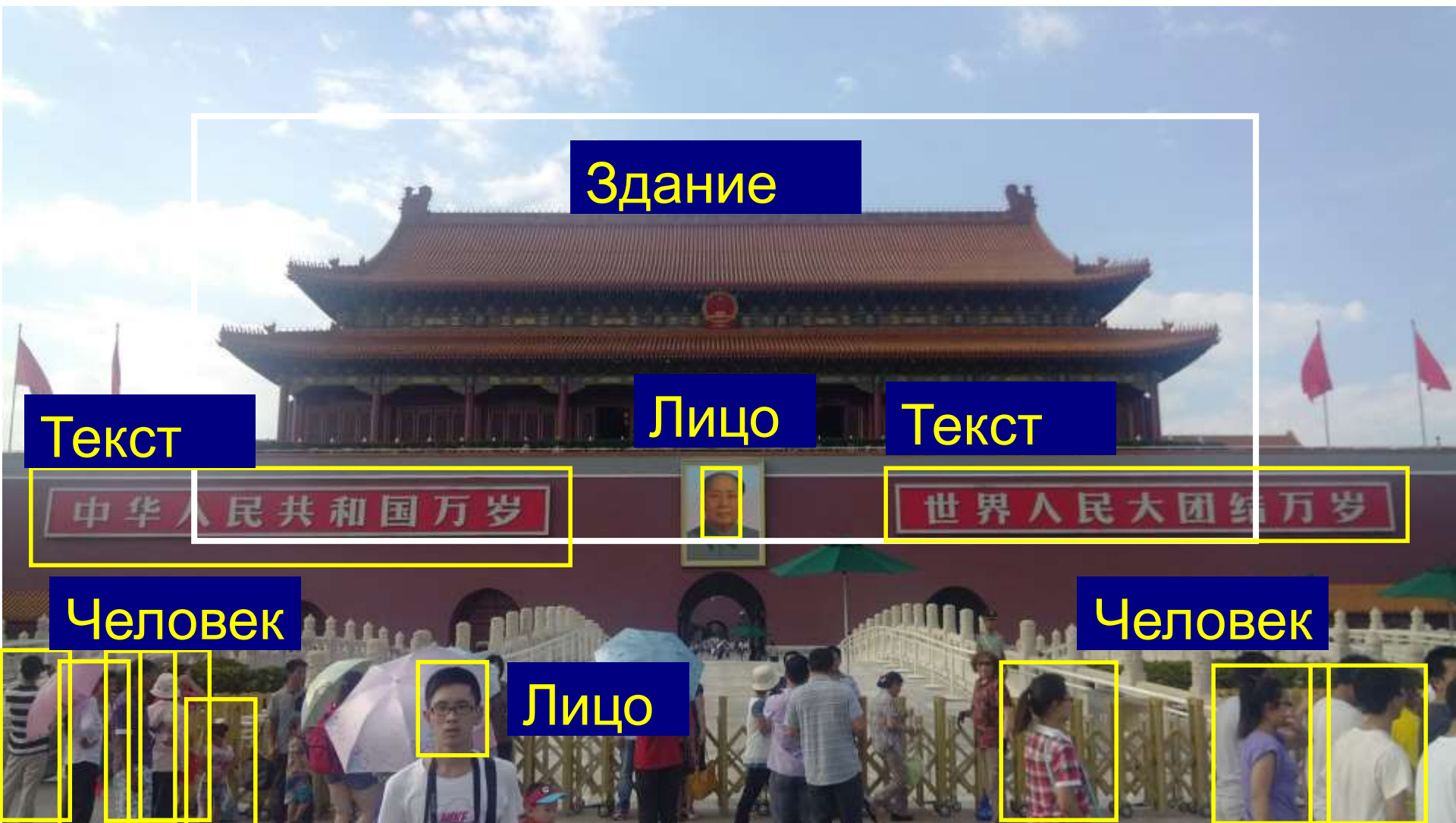
# Резюме классификации изображений

---

- Построение вектор-признака изображение обычно разделяют на 3 этапа
  - Вычисление признаков пикселей (features)
  - Кодирование признаков (coding)
  - Пространственная агрегация (spatial pooling)
- Для кодирования и агрегации часто используют гистограммы
- Визуальные слова можно рассматривать как расширение обычной схемы, в которой мы рассматриваем окрестности пикселей (фрагменты, а не отдельные пиксели)
- Основные – гистограммы цветов, HOG, BoW признаки
- Применяем любой классификатор поверх вектор-признаков



# Выделение объектов



Необходимо определить, есть ли на изображении объекты заданного типа и если да, то определить их положение



# Формализуем задачу

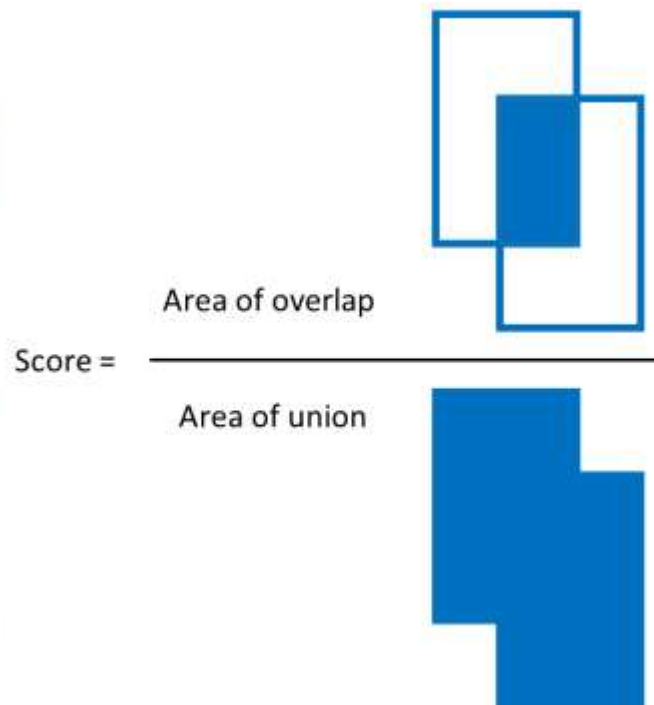
---

- Пусть задан набор из  $s$  интересующих нас классов объектов, пронумерованных от 1 до  $S$
- Вход:
  - Цветное изображение  $I \in R^{N \times M \times C}$
- Выход:
  - Набор найденных объектов  $Obj = \{Obj_i | i = 1, p\}$ , где  $Obj_i = (x_i, y_i, w_i, h_i, c_i)$ .  $x_i, y_i, w_i, h_i$  - координаты левого верхнего угла, ширина и высота  $i$ -го ограничивающего прямоугольника, содержащего объекта класса  $c_i \in [1, S]$

Алгоритмы выделения объектов на изображениях – **детекторы** объектов (object detection)

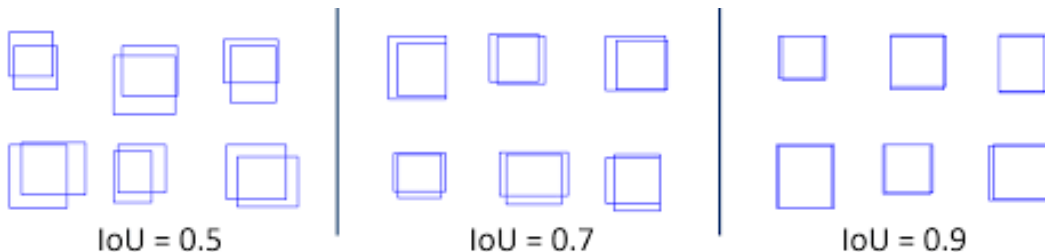


# Критерий обнаружения (IoU)



IoU = Intersection over Union

Обнаружение, если  
 $\text{IoU} > p$  (пр.: 0.5)

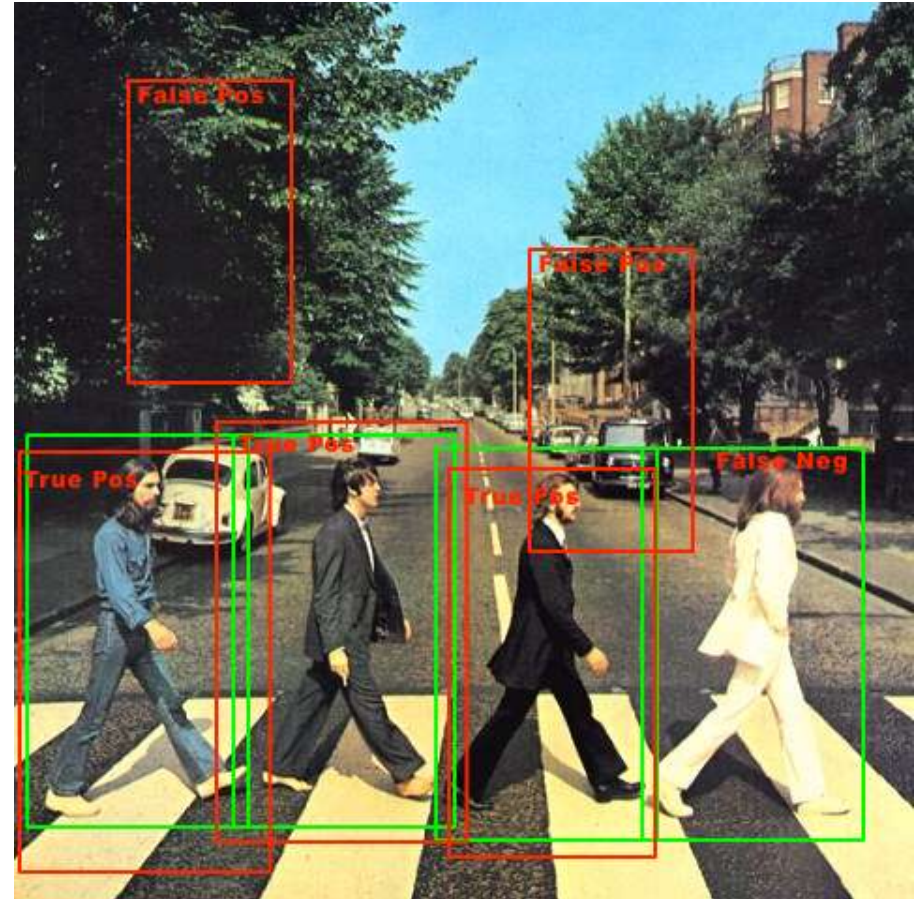






# Оценка качества детектора

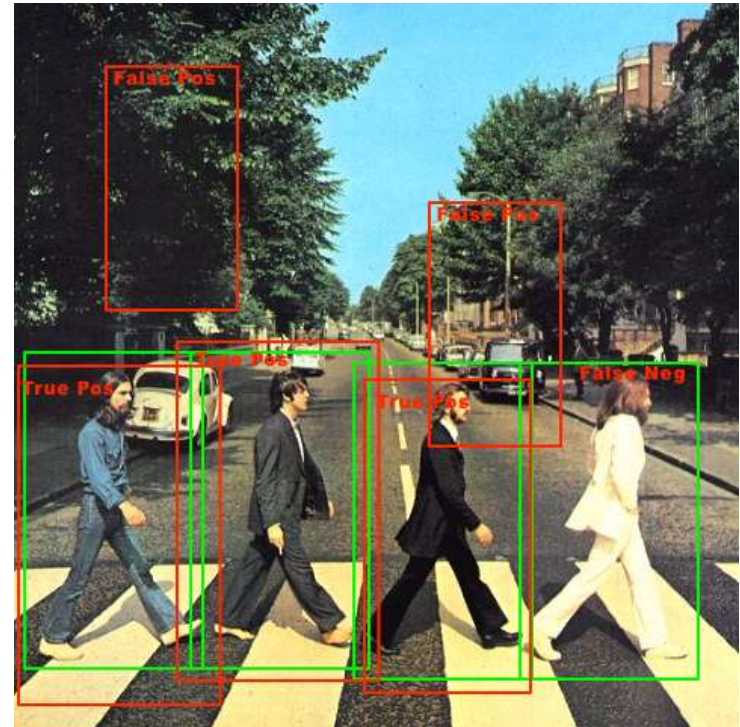
- Выход алгоритма:  
отсортированные по  
качеству обнаружения
- Все обнаружения  
оцениваются:
  - $IoU > p \Rightarrow$  true  
positive
  - $IoU < p \Rightarrow$  false  
positive
- Пропущенный пример  
 $\Rightarrow$  false negatives





# Precision & recall

- Точность (Precision)
  - Доля истинных объектов основного класса среди всех классифицированных, как основной класс
- Полнота (Recall)
  - Доля правильно распознанных объектов основного класса среди всех объектов основного класса из тестовой выборки



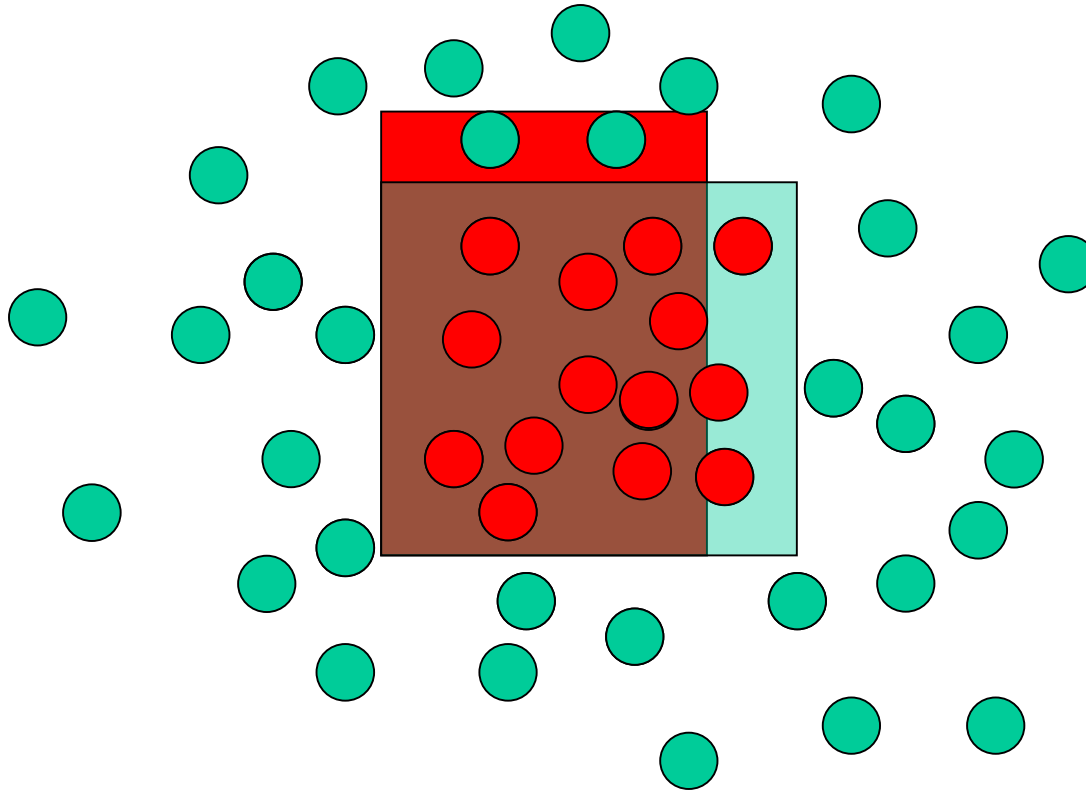
$$\text{Precision} = \frac{\text{Number of true detections}}{\text{Number of detections}}$$

$$\text{Recall} = \frac{\text{Number of true detections}}{\text{Number of gt objects}}$$



# Точность и полнота (пример)

---



- Полнота – 10 из 13, т.е. 77%
- Точность – 10 из 12, т.е. 83%

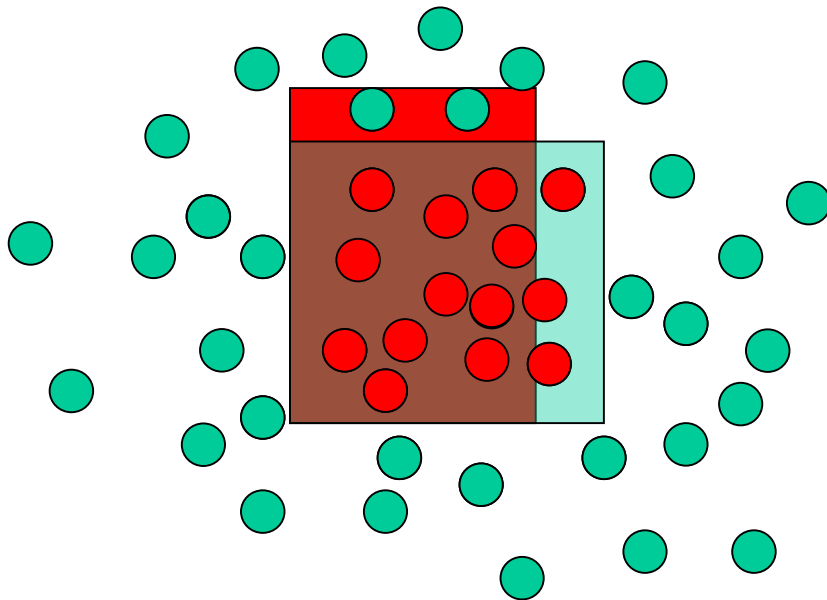




# Интегральный показатель

- Интегральный показатель F (F-measure)

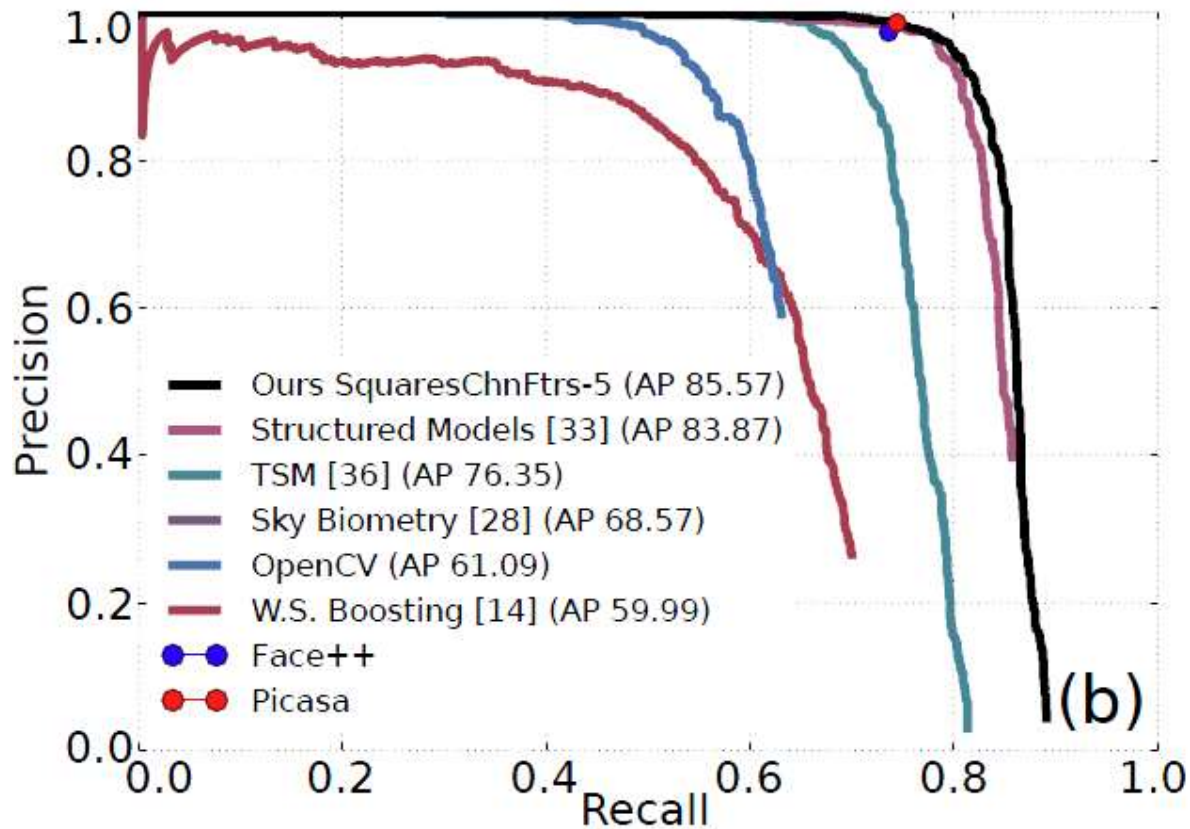
- $$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$



- Полнота – 10 из 13, т.е. 77%
- Точность – 10 из 12, т.е. 83%
- F-мера =  $2 * 0.77 * 0.83 / (0.77 + 0.83) = 0.8$



# Кривая точности-полноты

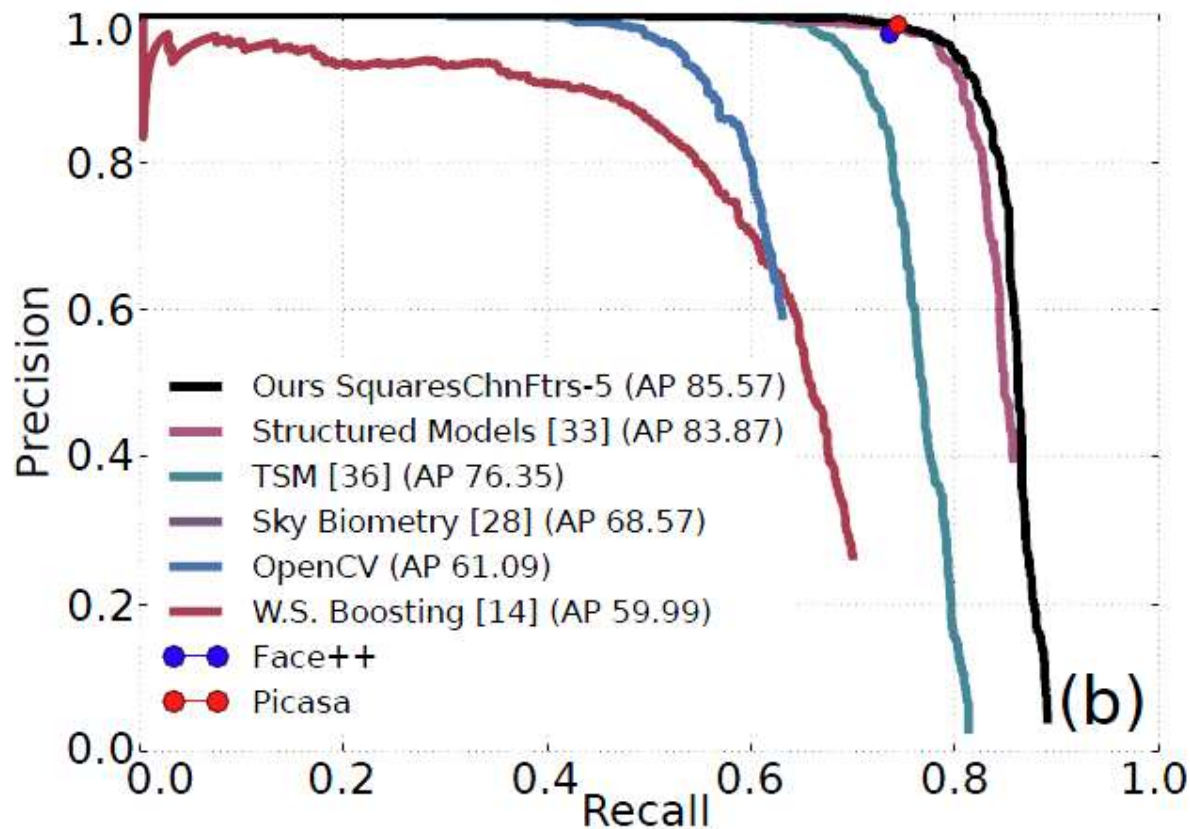


M. Mathias. et. al.  
Face detection  
without bells and  
whistles. ECCV2014

Зависимость между точностью и полнотой в зависимости от параметров



# Пример

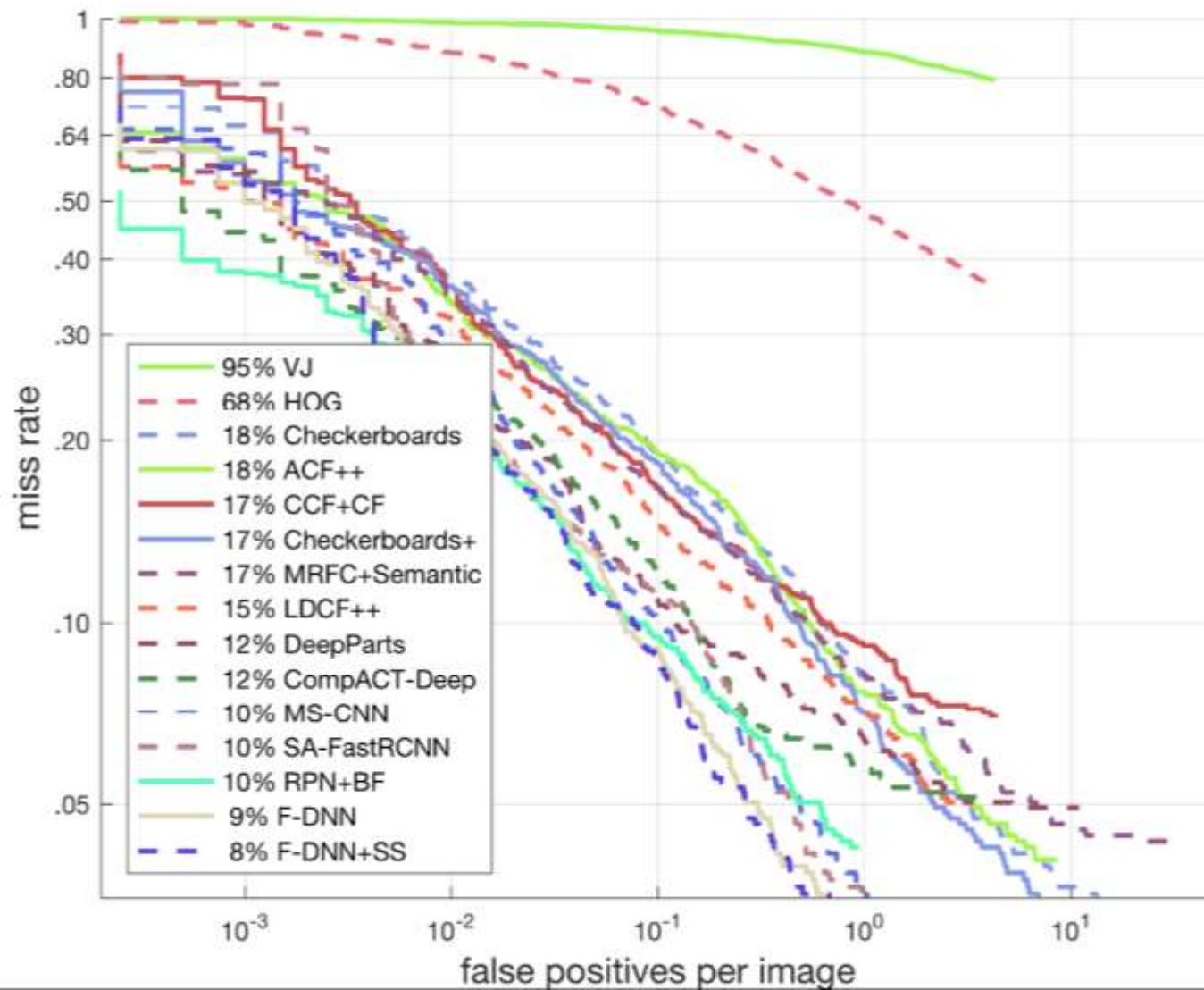


M. Mathias. et. al.  
Face detection  
without bells and  
whistles. ECCV2014

По кривой точности-полноты мы можем выбрать параметр с оптимальным для наших целей соотношением ошибок



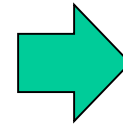
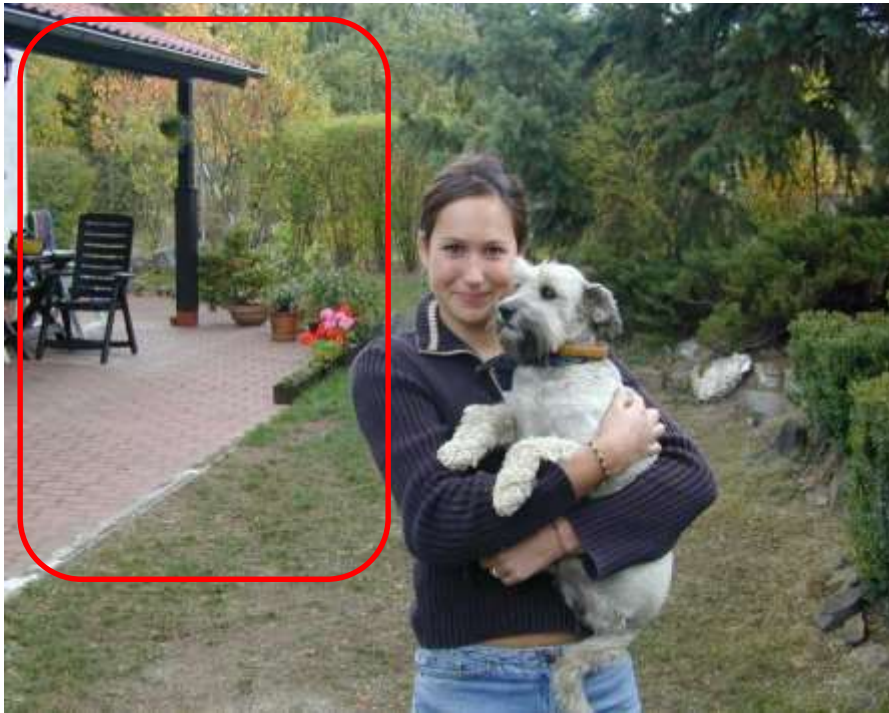
# Miss rate vs FPPI



Source: [http://www.vision.caltech.edu/Image\\_Datasets/CaltechPedestrians/](http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/)



# Сведение выделения к классификации



- Мы умеем обучать бинарные классификаторы «объект»/ «не объект». Как нам решить задачу локализации?
- Скользящее окно – просмотр всех фрагментов изображения и применения к ним классификатора





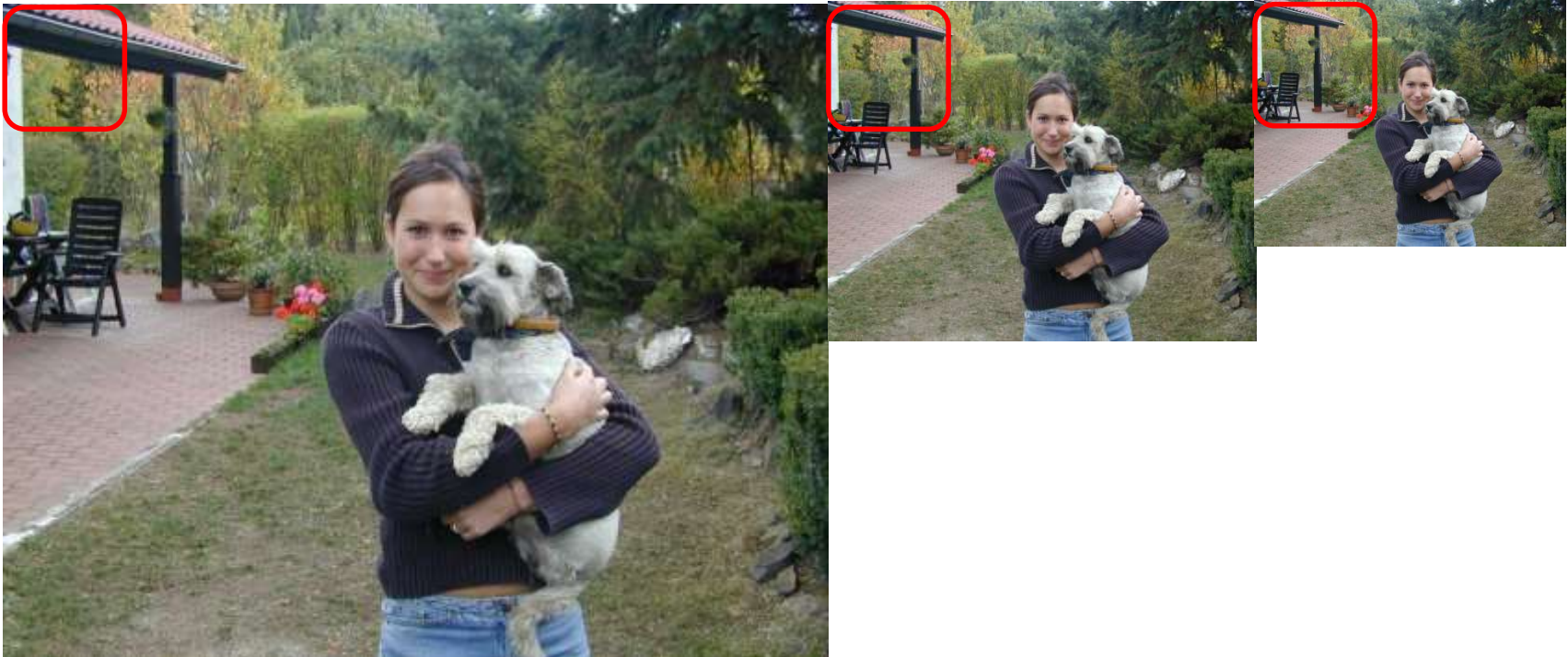
# Разные размеры окна



Сканируем изображение окнами разного размера



# Разные размеры объектов



- Строим пирамиду масштабов и сканируем окном одного размера
- В зависимости от качества признаков и классификатора может потребоваться разное число масштабов



# Разные пропорции объектов

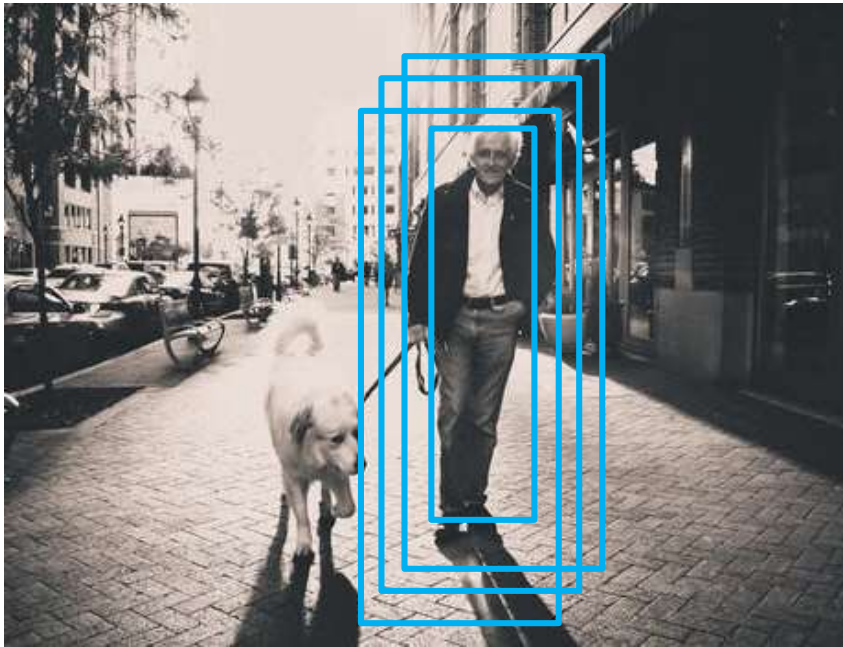


- Используем набор окон с разными пропорциями
- “Number of images scanned” =  
“Number of scales” x “Number of aspect ratios”



# Множественные отклики

---



Множественные отклики  
вокруг объекта

Как выбрать  
наилучшие?

- Алгоритм жадного подавления
  - На вход список окон, отсортированных по убыванию score (выхода классификатора)
  - Выбираем окно с наибольшим score
  - Находим и убираем все окна, для которых  $\text{IoU} > 0.5$
  - Повторяем до сходимости



# Сопоставление шаблонов

- Рассмотрим простейший алгоритм
- Пусть изображение объекта «фиксировано», т.е. мало меняется
- Назовём изображение объекта шаблоном *template*
- В качестве  $x = \text{features}(I, r)$  будем использовать векторизацию фрагмента изображения  $r(I)$ 
  - Длина вектора признака – число пикселей фрагмента
  - Можем брать цветной фрагмент
- В качестве  $y = \text{analysis}(x)$  воспользуемся функцией сравнения изображений, например, MSE
  - $d = \text{Distance}(x, \text{template})$
  - $y = \begin{cases} 0, & d \geq t \\ 1, & d < t \end{cases}$
  - где  $t$  – порог на сходство изображений

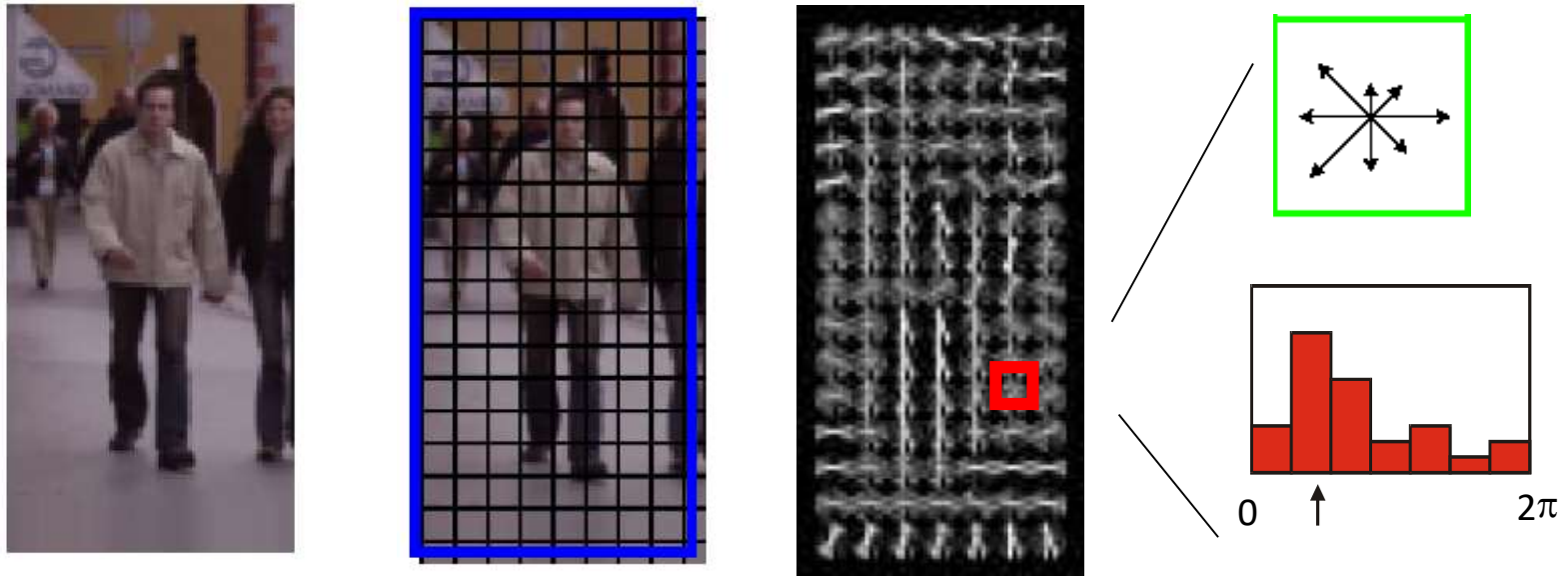


*template*





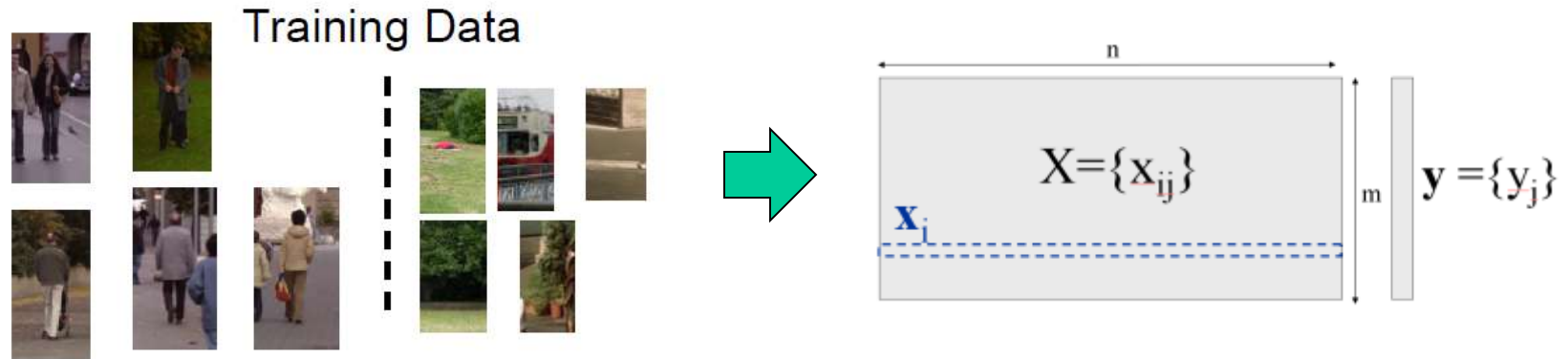
# HOG+SVM для поиска пешеходов



- Возьмём окно 64 x 128 пикселей
- Разобъём его на блоки 8 x 8 пикселей
- Всего будет  $8 * 16 = 128$  блоков
- В каждом блоке посчитаем гистограмму ориентаций градиентов с 8 ячейками (8 параметров)
- Всего у нас получится  $128 * 8 = 1024$  признака



# Обучение классификатора



- Соберём обучающую выборку фрагментом изображения с пешеходами и без
- Для каждого фрагмента посчитаем вектор признаков  $x$  и метку  $y$
- На полученной обучающей выборке обучим линейный классификатор SVM (метод опорных векторов)

# SVM

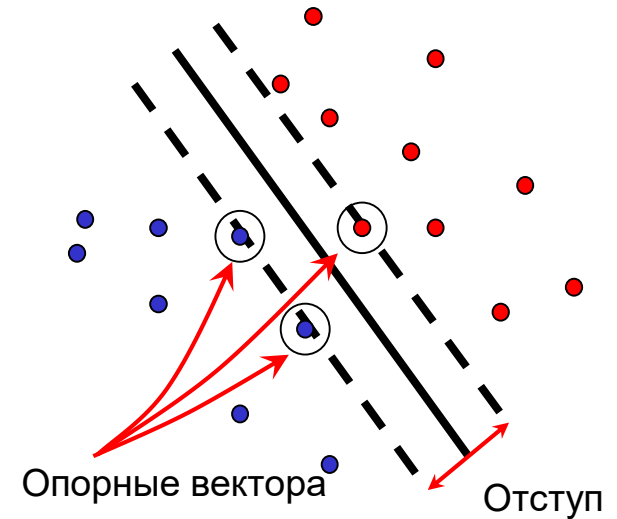


- Обучаем линейную SVM:

$$f(x) = w^T x + b$$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

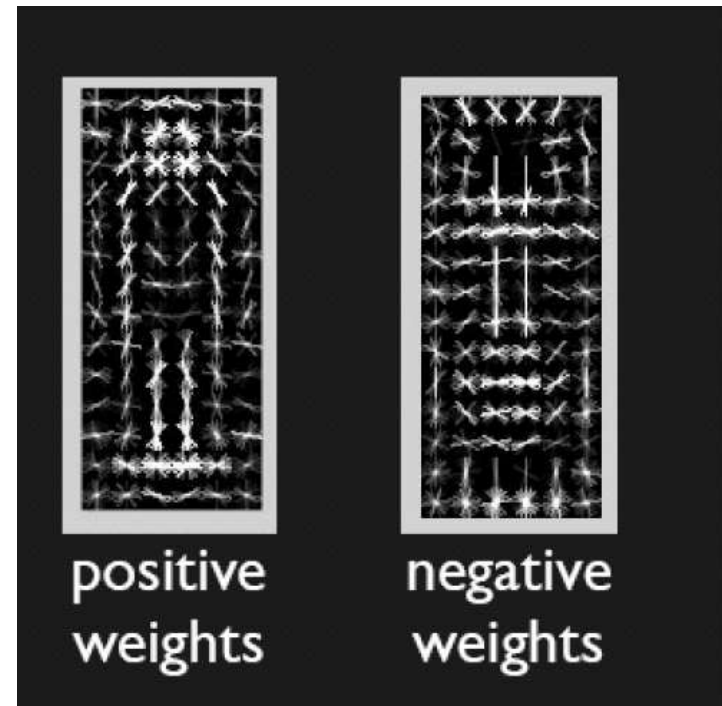
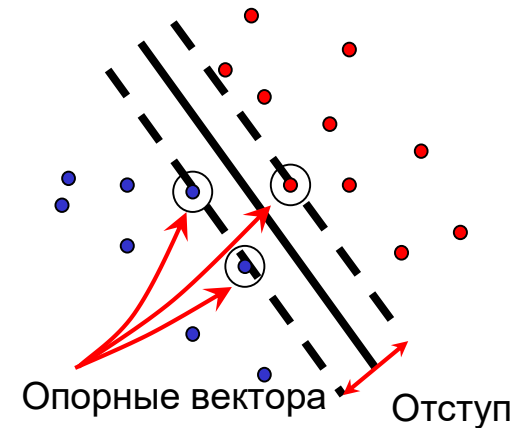
- Опорные вектора с положительными и отрицательными весами
- Чем фактически в нашем случае являются  $\mathbf{x}$  ?



# SVM

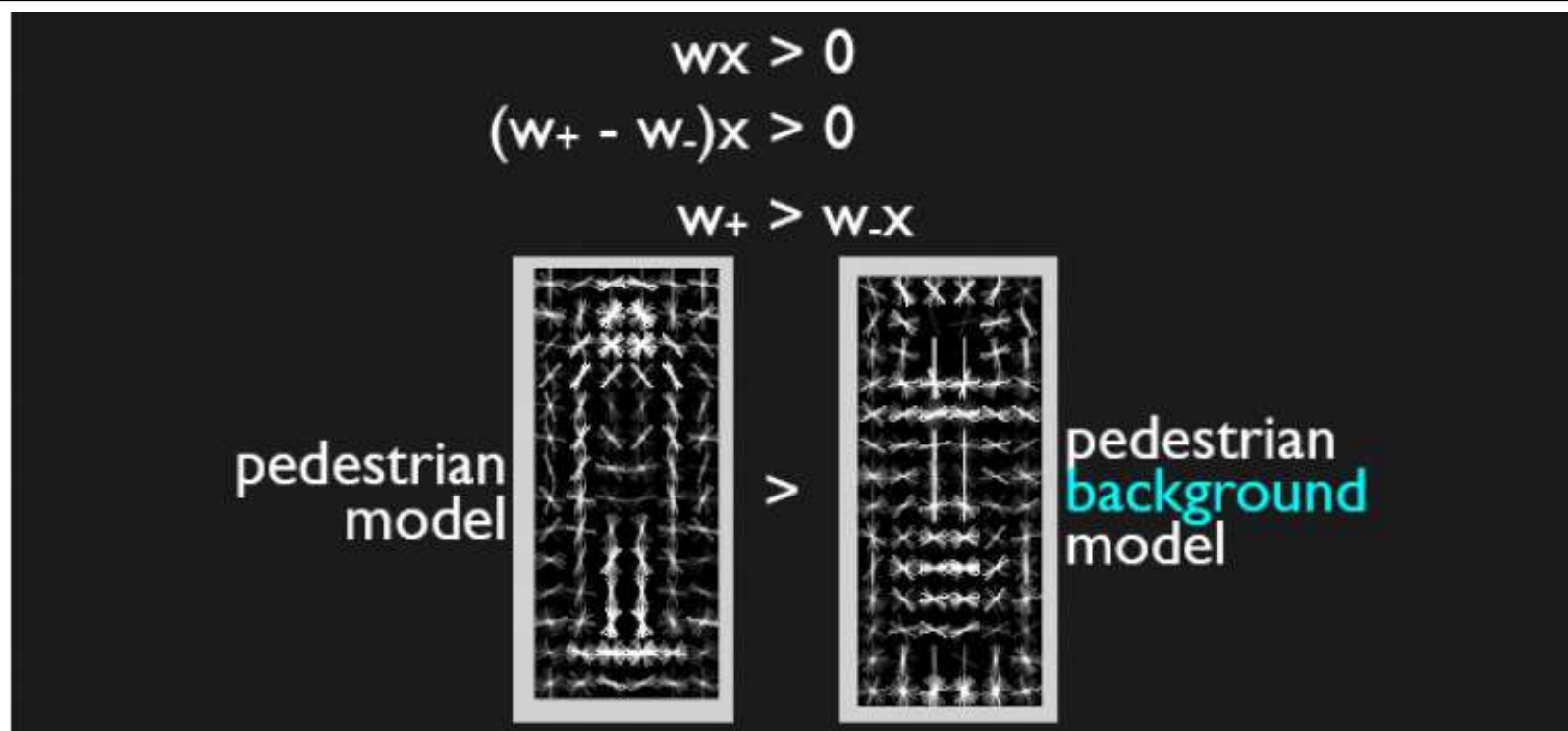


- Каждый опорный вектор – это вектор-признак одного из примеров
- Опорные вектора с положительными и отрицательными весами
- Положительный вес – пример фрагмента, содержащего пешехода
- Отрицательный вес – пример фрагмента, содержащего фон





# SVM



- Разделяющая гиперплоскость задается как  $wx$  – линейная комбинация положительных и отрицательных примеров
- Каждый признак – «мягкий шаблон» краёв, мы берём не чёткие края, а распределение ориентаций краёв в небольших областях
-



# «Детектор пешеходов»

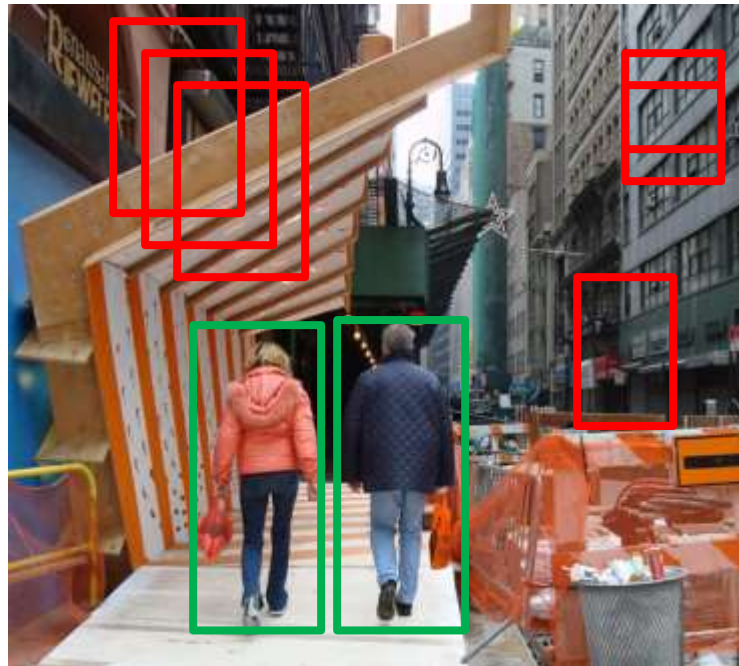
---

- Получили работающий «детектор пешеходов»
- Базовый алгоритм:
  - Скользящее окно
  - Вычисление признаков (гистограмм ориентаций градиентов)
  - Классификация с помощью SVM
- Базовый алгоритм можно существенно улучшить простыми способами



# Обучение детектора

- Сколько на изображении объектов «пешеход» и сколько фрагментов фона?

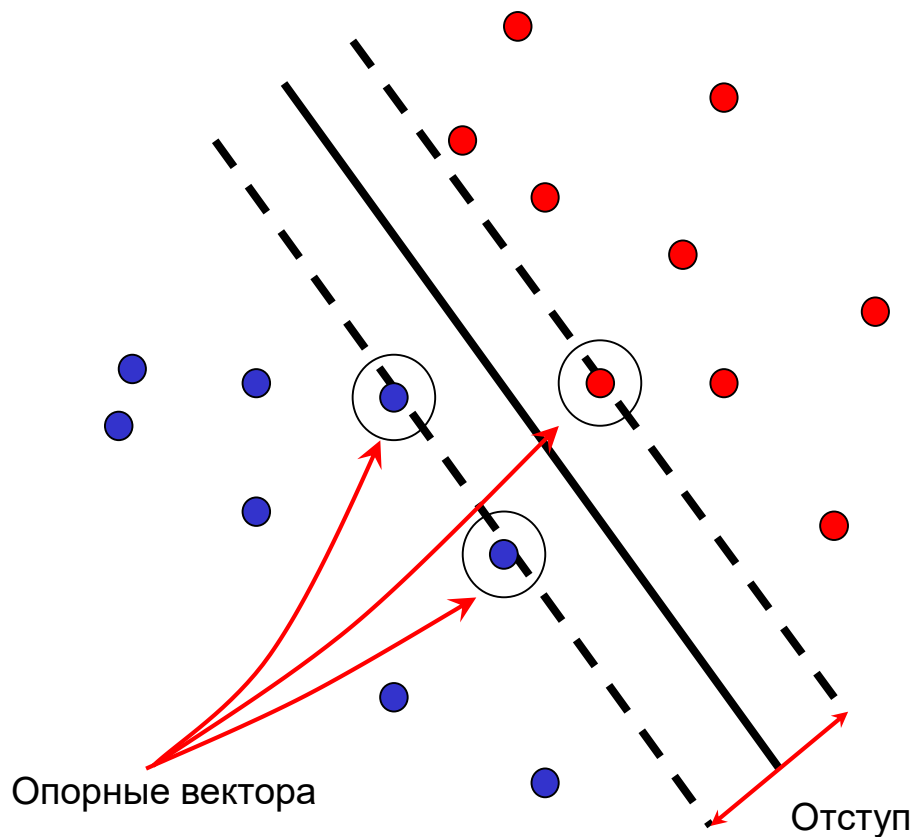


- Выделение объектов ассиметричная задача: объектов гораздо меньше, чем «не-объектов»
- Вдобавок, класс «не объект» очень сложный – нужно много разных данных для обучения
- Для SVM желательно одинаковое количество и фона, и объекта



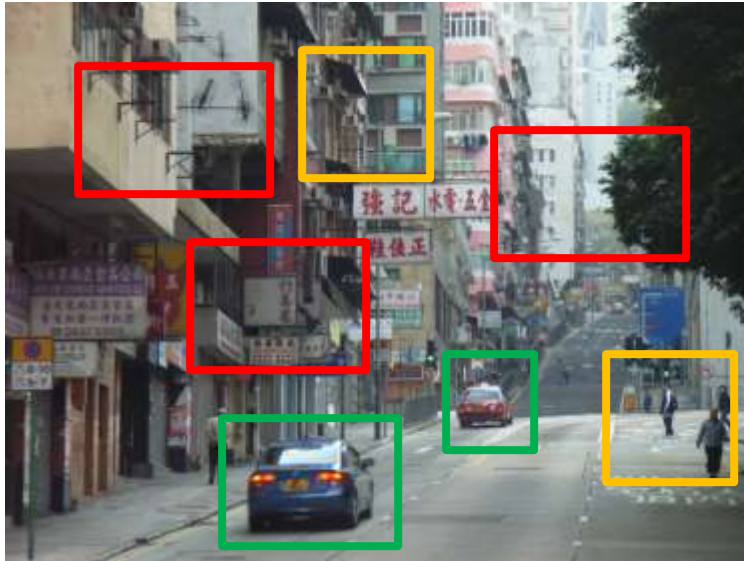
# Обучение детектора

Какие нам нужны примеры фона для получения наилучшего детектора?





# Hard-negative mining



- Выбираем отрицательные примеры случайным образом
- Обучаем классификатор
- Применяем к данным
- Добавляем ложные обнаружение к выборке
- Повторяем

- Смысл:

- Ложные обнаружения для первого детектора – сложные (**hard negative**)
- Пусть наша выборка фона будет маленькой, но сложной и представительной



# Пример – поиск «торса»

---

- Хотим построить детектор «верхней части тела и головы»
- Воспользуемся схемой HOG + линейный SVM с бустраппингом
- Данные
  - 33 фрагмента фильмов из базы Hollywood2
  - 1122 кадров с размеченными объектами
- На каждом кадре отмечены 1-3 человека, всего 1607 людей



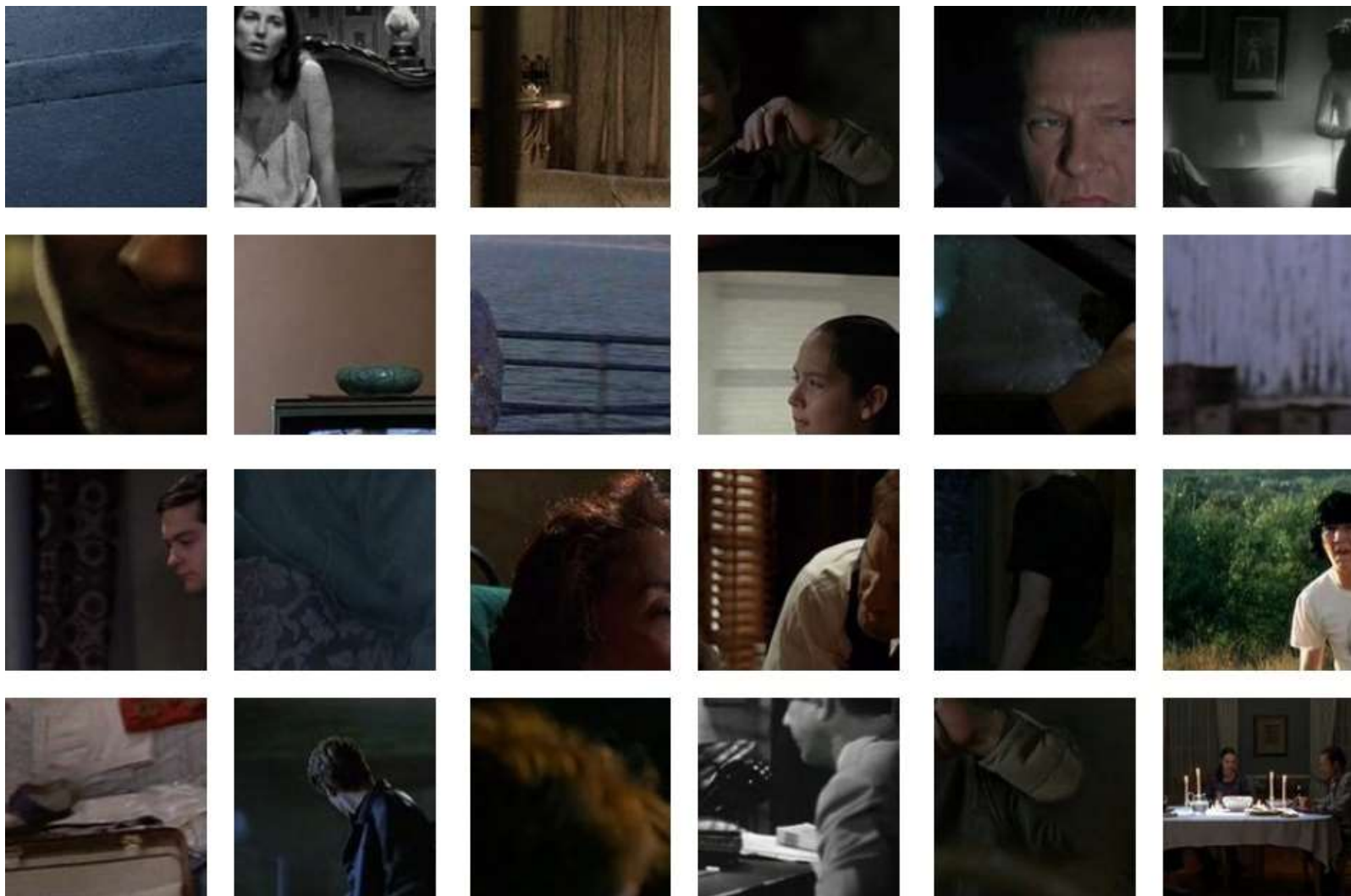




# Первая итерация

---

Соберём случайные фрагменты фона:





# Первая стадия

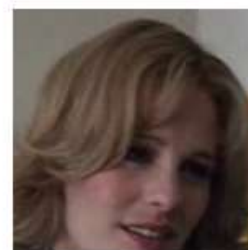
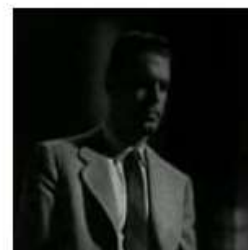
---

Трудный  
отрицательный  
пример



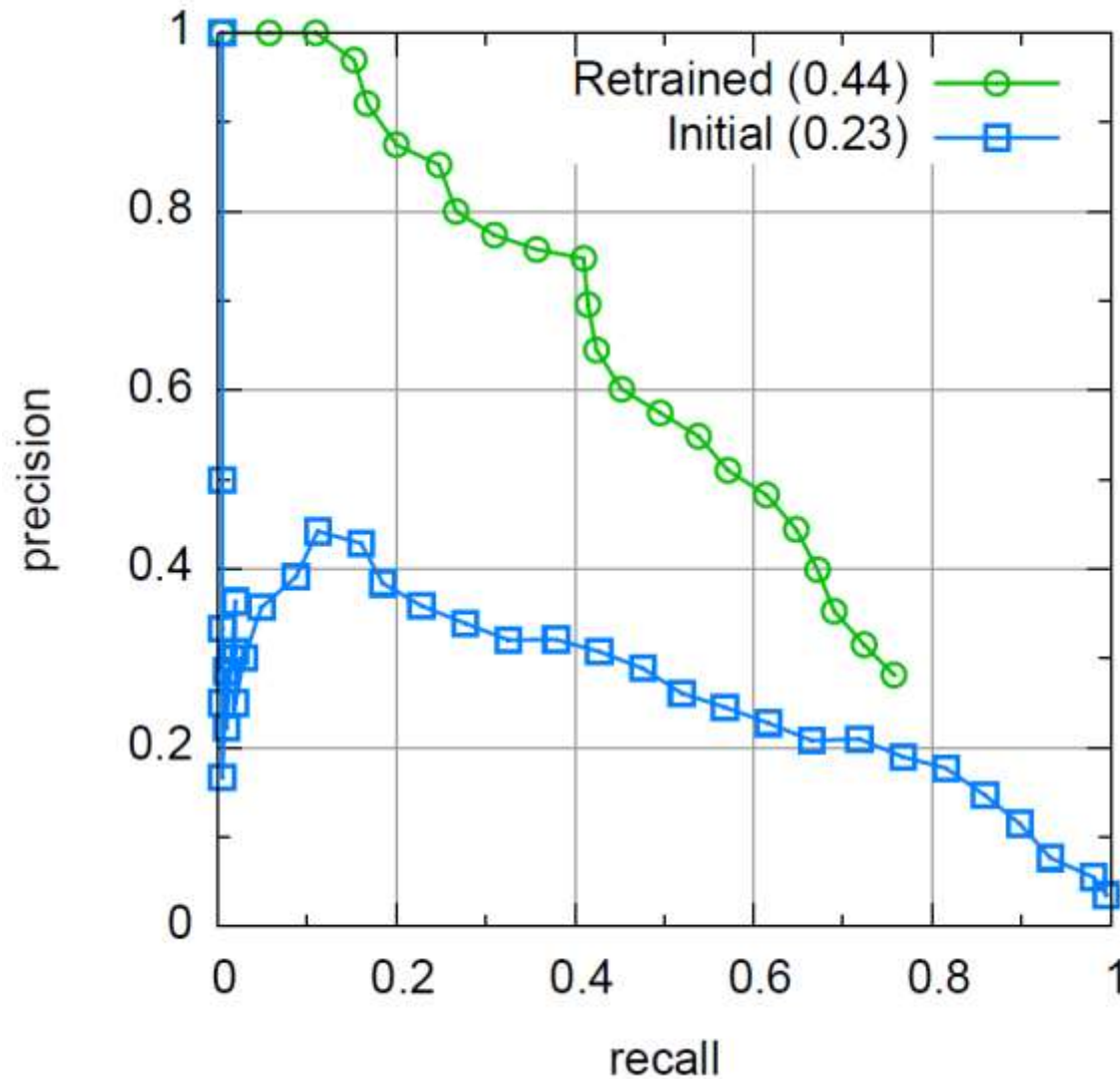
- Обучим детектор по случайной выборке фона
- Применим его к исходным изображениям
- Ищем ложные обнаружения с высоким рейтингом
- Используем их как трудные отрицательные примеры
- Затраты:  $\#$  количество изображений  $\times$  поиск в каждом

# Трудные примеры



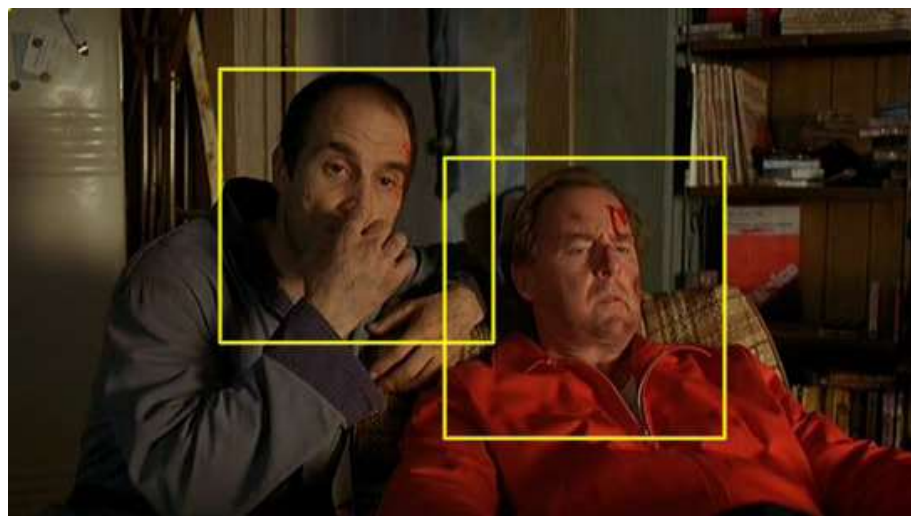
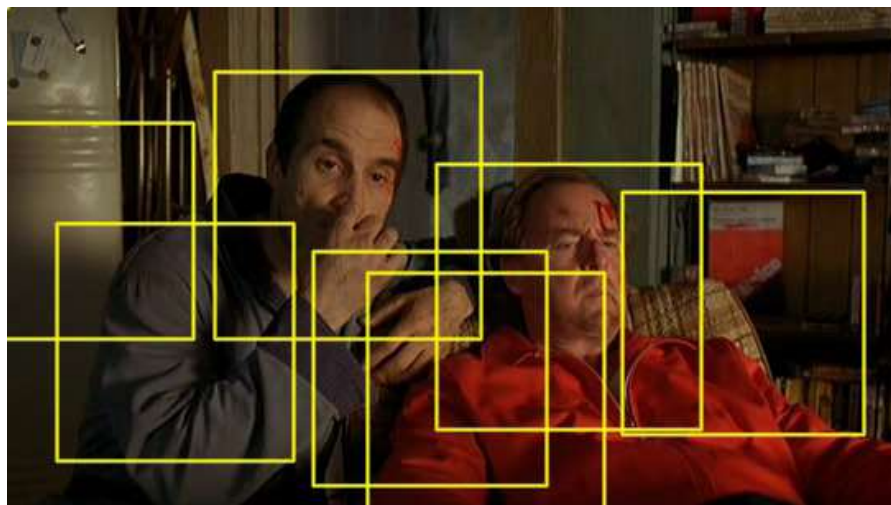
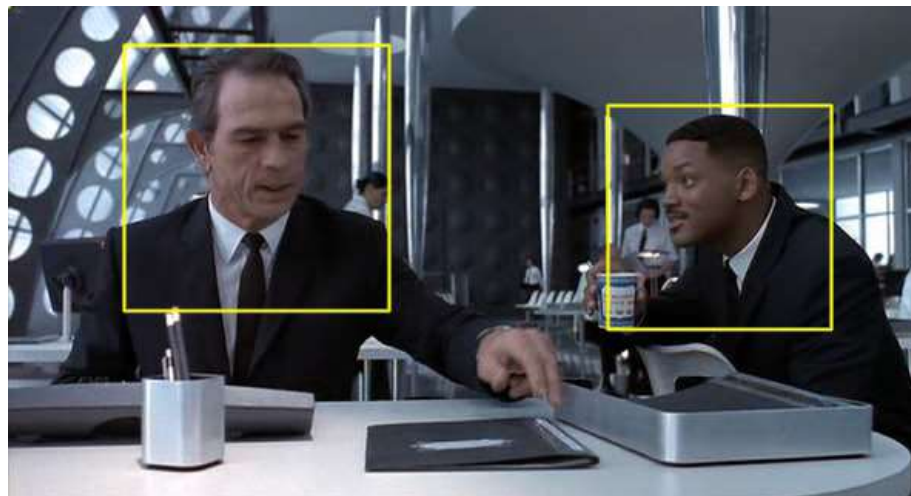
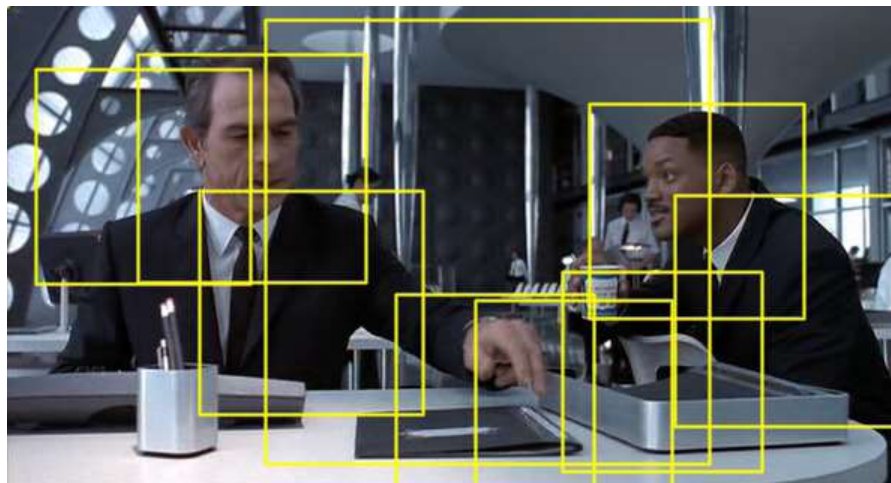


# До и после hard-negative mining

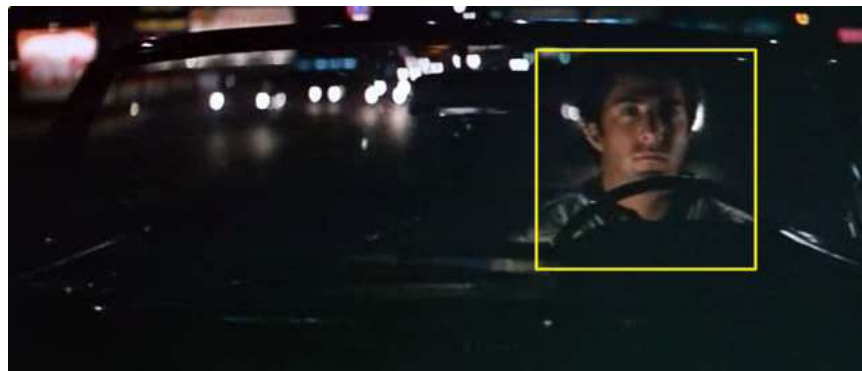
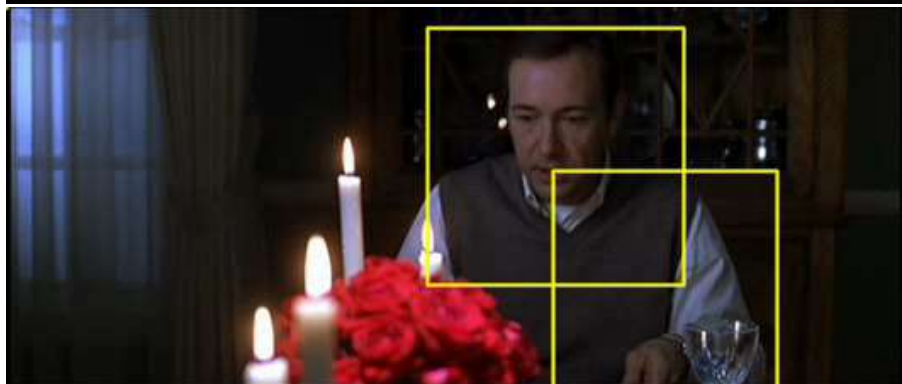
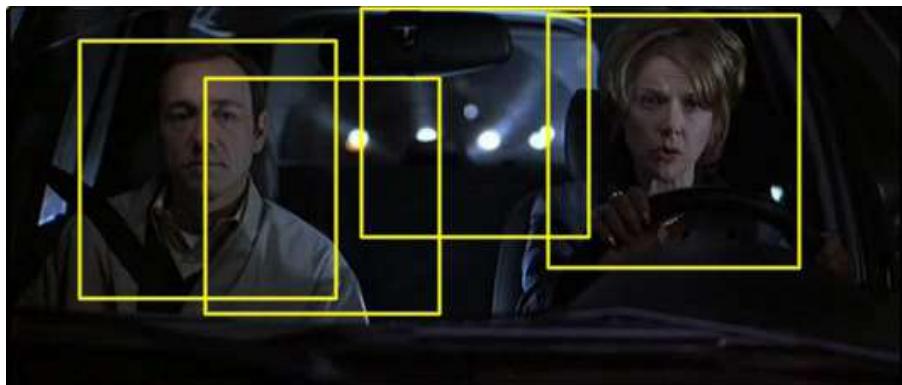




# Сравнение



# Сравнение







# Резюме алгоритма HOG + SVM

---

- Используем скользящее окно
- Вычисляем вектор-признак на основе HOG
  - Разбиваем окно на ячейки
  - В каждой ячейке считаем гистограмму ориентации градиентов
- Обучаемый линейный SVM
- Для обучения:
  - Размножаем (шевелим) эталонные примеры объектов
  - Используем схему bootstrapping для выбора примеров фона
    - На первой стадии берём случайные окна для фона
    - На следующих стадиях выбираем ложные срабатывания детектора как «трудные» примеры



# Резюме лекции

---

- Мы рассмотрели 2 задачи – классификация изображений и выделение объектов
- Задачу выделения объектов можно свести к классификации изображений с помощью метода скользящего окна
- Стандартная схема построения признаков состоит из features, coding, spatial pooling
- В качестве признаков чаще всего используют гистограммы распределений цвета, градиентов, краёв, визуальных слов
- Любой метод классификации на основе машинного обучения можно использовать
- Есть ряд хороших эталонных коллекций, но для прикладных задач приходится размечать вручную