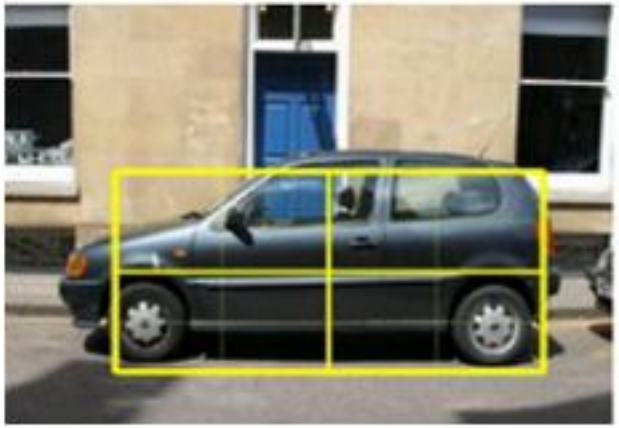
Выделение объектов



Антон Конушин

Many slides adopted from Svetlana Lazebnik, Ondra Chum, Alyosha Efros, Mark Everingham, Pedro Felzenszwalb, Rob Fergus, Kristen Grauman, Bastian Leibe, Ivan Laptev, Fei-Fei Li, Marcin Marszalek, Pietro Perona, Deva Ramanan, Bernt Schiele, Jamie Shotton, Josef Sivic and Andrea Vedaldi

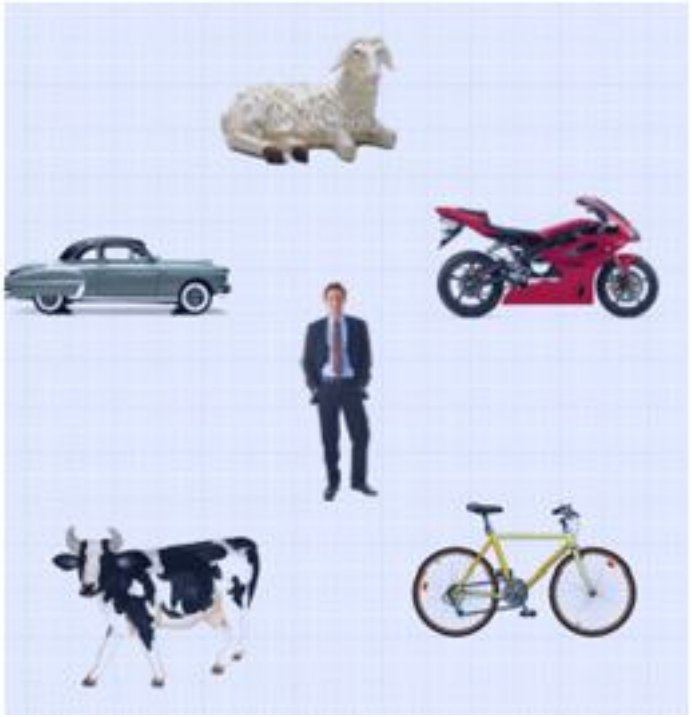
Выделение объектов

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

Things vs. Stuff

Объекты (Things)

- Имеют определенные размер и формы



Найти «человека»

Материалы (Stuff)

- Однородный или повторяющийся шаблон мелких деталей
- Нет определенного размера и формы

Найти «небо»

Ted Adelson, Forsyth et al. 1996.

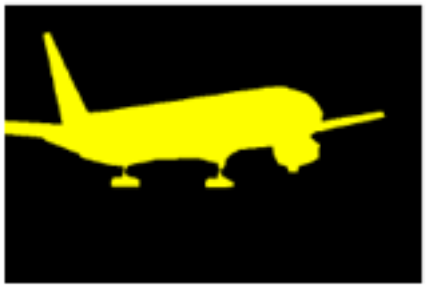
Slide: Geremy Heitz

Задачи распознавания

- Категоризация изображений
  - Image categorization
  - Содержит ли изображение самолёт?

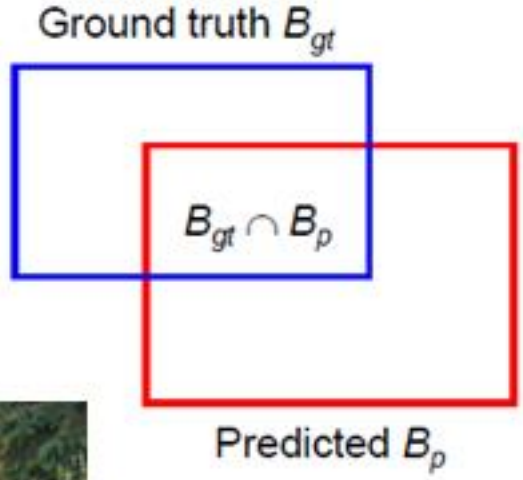
- Выделение объектов
  - Object class detection and localization
  - Есть ли на изображении самолёт, и если да, то где и сколько их?
  - Каждый объект выделяем рамкой (bounding box)

- Семантическая сегментация
  - Object class pixel-level segmentation
  - Какие пиксели изображения принадлежат самолёту?

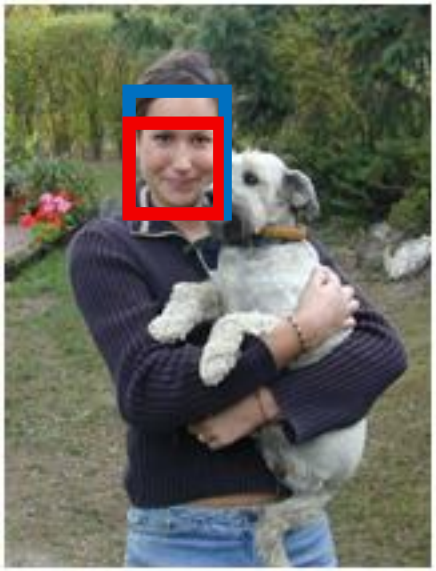


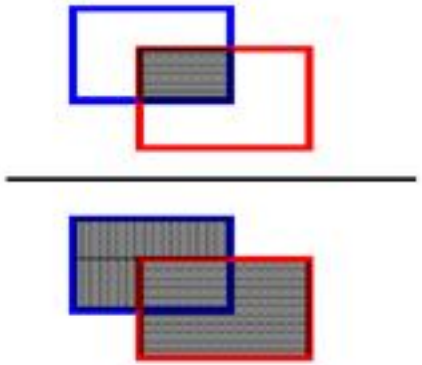
Критерий обнаружения

- Area of Overlap (AO) Measure



$$AO(B_{gt}, B_p) = \frac{|B_{gt} \bigcap B_p|}{|B_{gt} \bigcup B_p|}$$



Detection if  > Threshold

50%

Пример - поиск пешеходов

- Нужно найти «пешеходов» (если есть) и выделить их прямоугольной рамкой
- Пешеход (pedestrian) – это «категория», а не «объект»
  - Пешеходы все похожи, но все разные
  - Известным нам способом (сегментация по контрасту + анализ признаков сегмента) выделять пешеходов получается плохо

Классификация изображения

- Простая бинарная классификация может ответить на вопрос «да» / «нет»
- Например: «есть» пешеход на изображении или «нет» пешехода на изображении

Обучающая выборка

«Картинка ли это пешехода»?

- А нам нужно знать, где ещё на картинке пешеход!

Скользящее окно (*sliding window*)

- Возьмём «окно», пусть размером 64*128 пикселов
- Сканируем изображение «окном»
- Применяем классификатор «пешеход?» к каждому окну
- Скользящее окно – форма сегментации изображения!

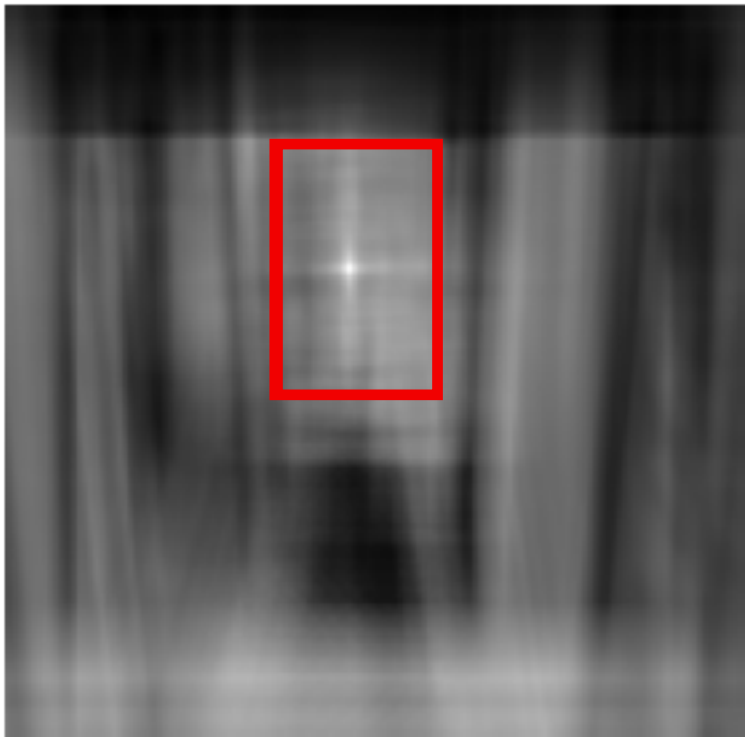
Разделили изображение на фрагменты, которые классифицируем независимо друг от друга

Простейший классификатор

Найдём стул в изображении

Выход корреляции

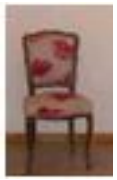
Это стул



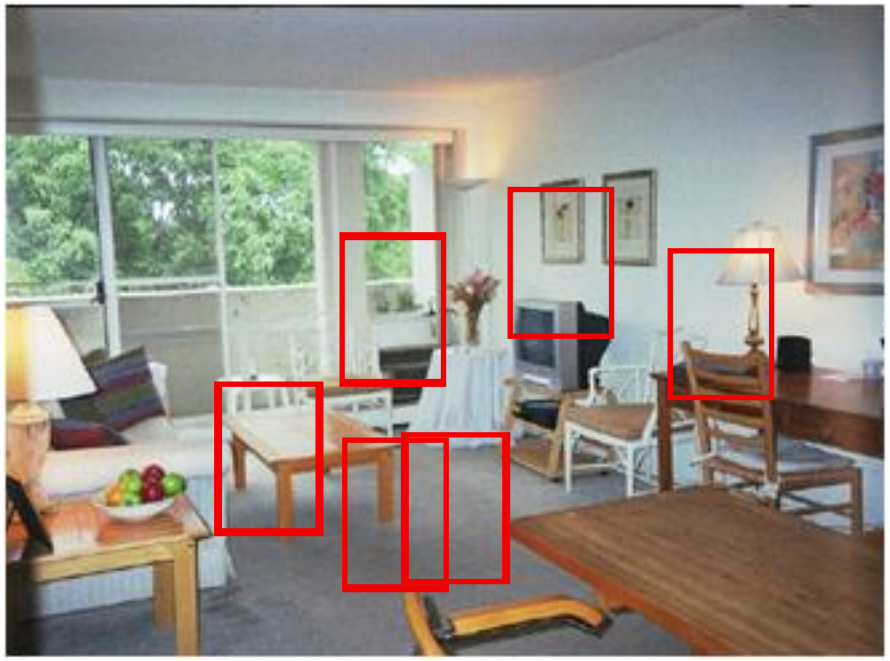
Сопоставление шаблонов («Pattern matching»):

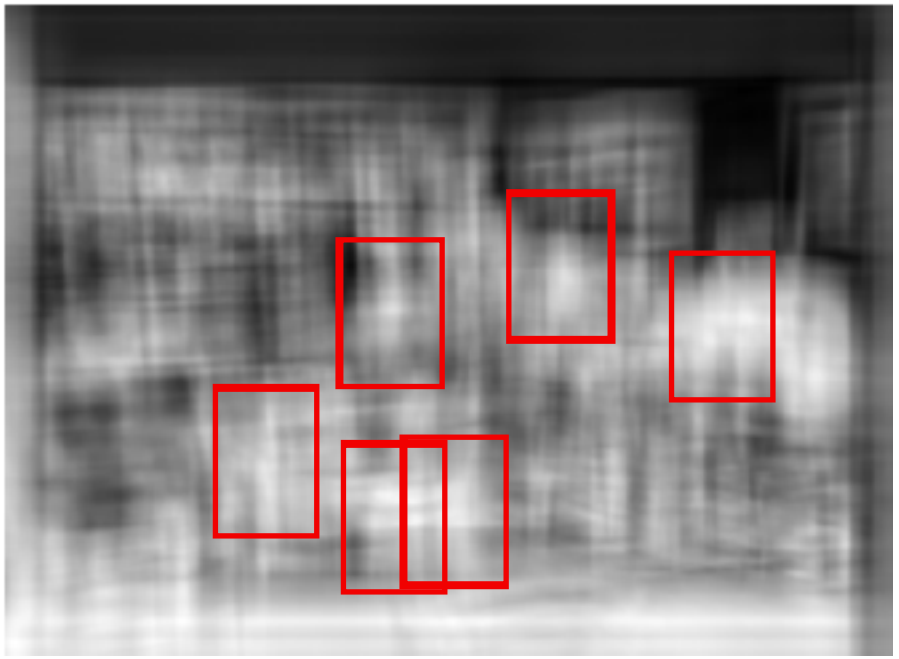
- Фиксируем изображение объекта («шаблон»)
- Классификатор – сравнение шаблона и текущего окна (фрагмента)
- Итогом скользящего окна будет карта откликов (score)

Ограничения подхода



Найдём стул в этом изображени





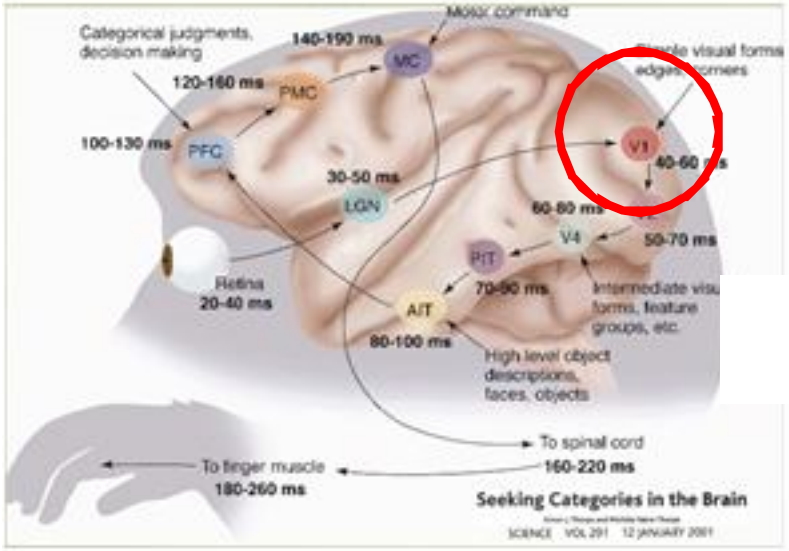
Мусор

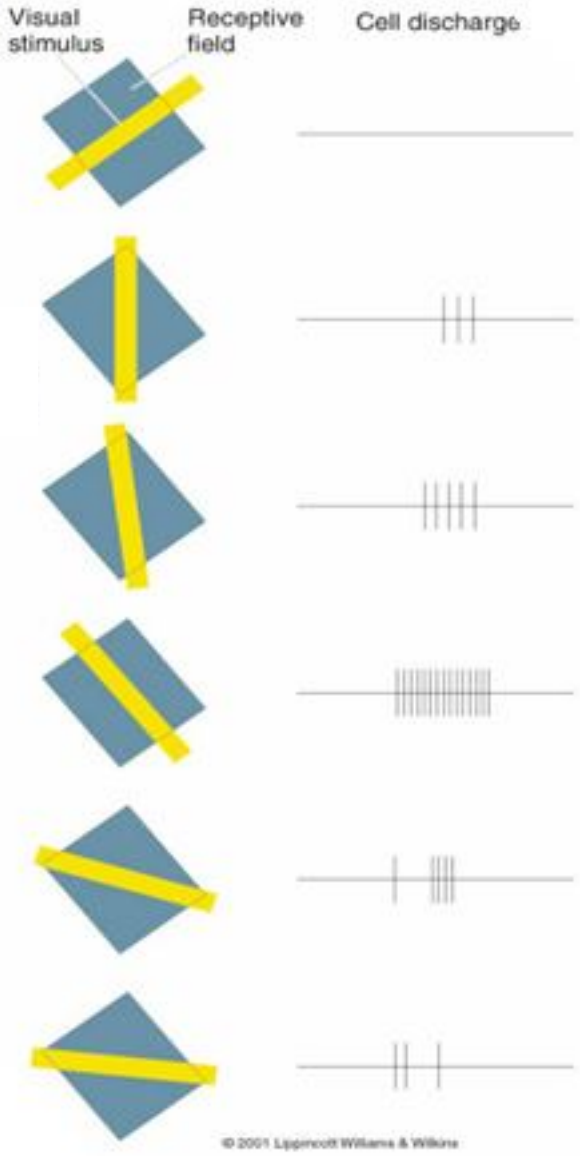
«Классификатор» в виде сравнения с шаблоном не может учесть всё множество факторов изменчивости

Построение классификатора

- Нужно выбрать признаки и классификатор
- Мы рассмотрим схему HOG + линейный SVM

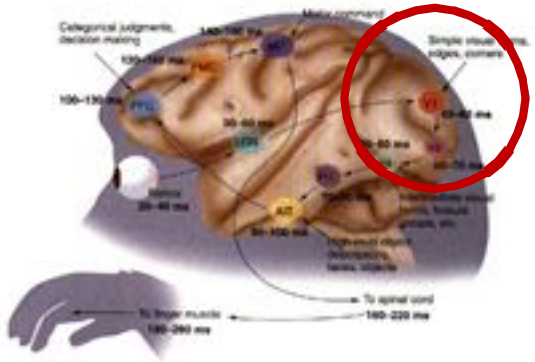
«Простые клетки» V1

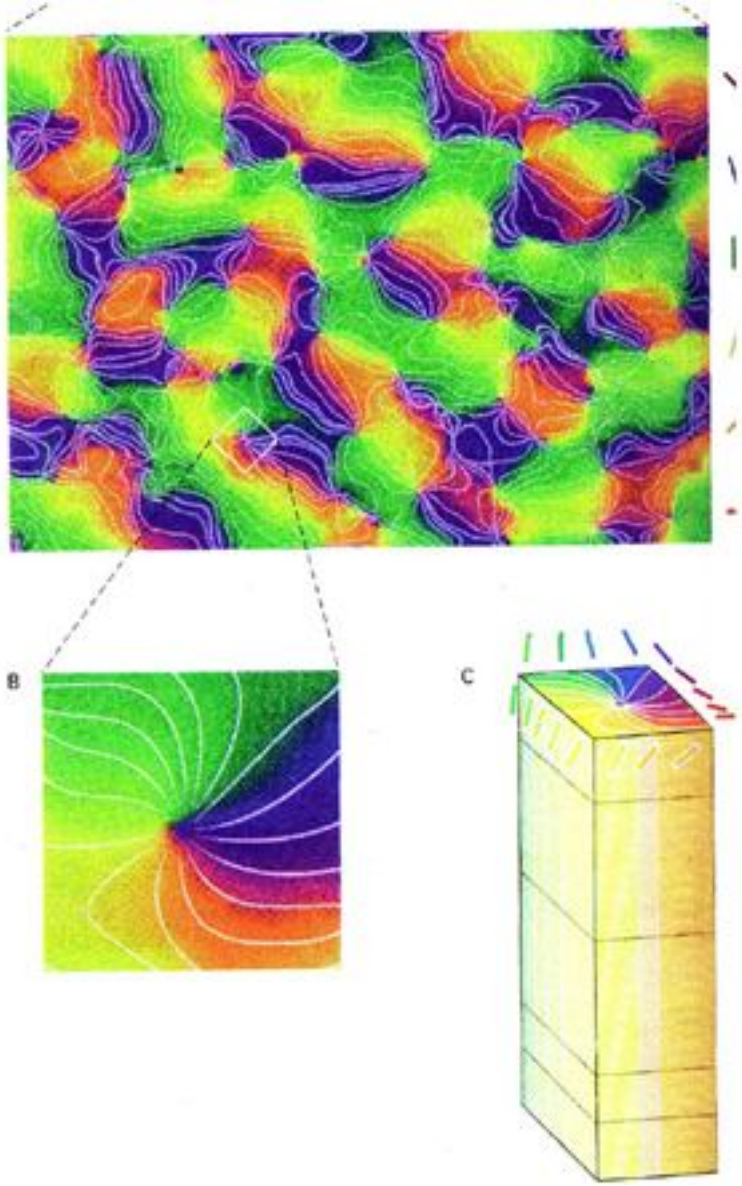




Вспомним, что первичная зрительная кора в основном занимается анализом краёв и градиентов в изображении

«Ретинотопическая» организация



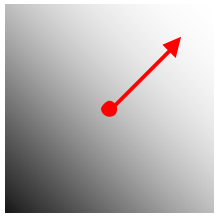


Для каждой области визуального поля есть клетки, чувствительные к краям разной ориентации (отображены на рисунке цветом)

Построим признаки, организованные по похожей схеме!

Оценка градиентов

- Градиент изображения – направление максимального изменения яркости изображения



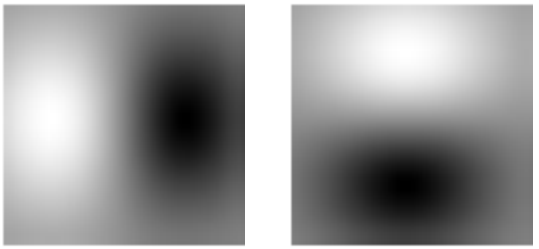
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

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

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

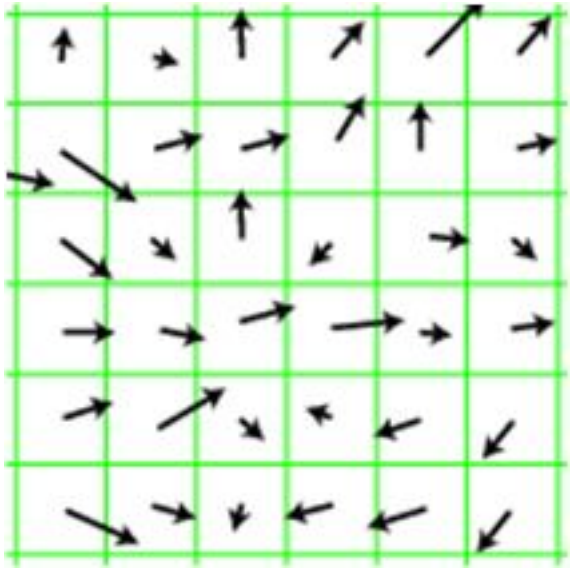
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$ Собеля

Вычисление разностной производной через свёртку



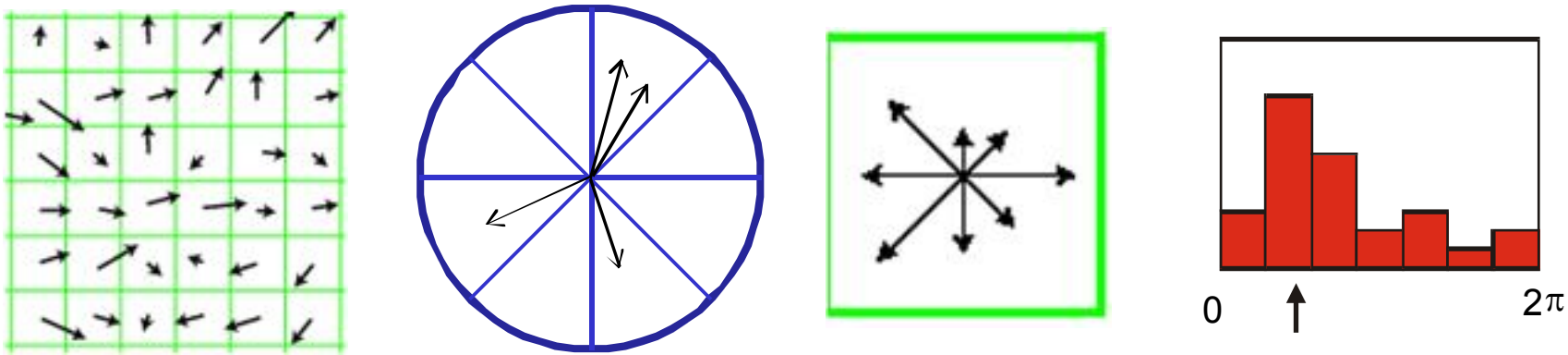
Вычисление градиента со сглаживанием

Признаки через градиенты



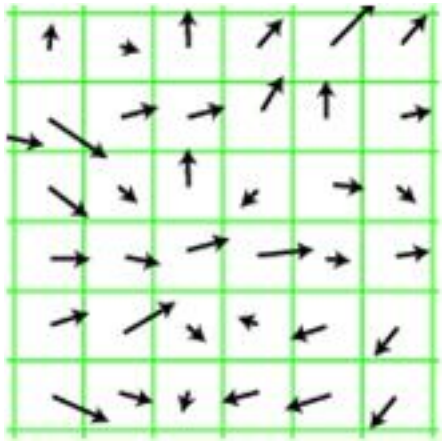
- Вычислим градиенты в каждом пикселе
  - Направление градиента
  - Силу градиента
- Как использовать градиенты в качестве признаков?
  - Можем использовать напрямую (слишком много, слишком шумные и изменчивые)
  - Статистика распределения по области

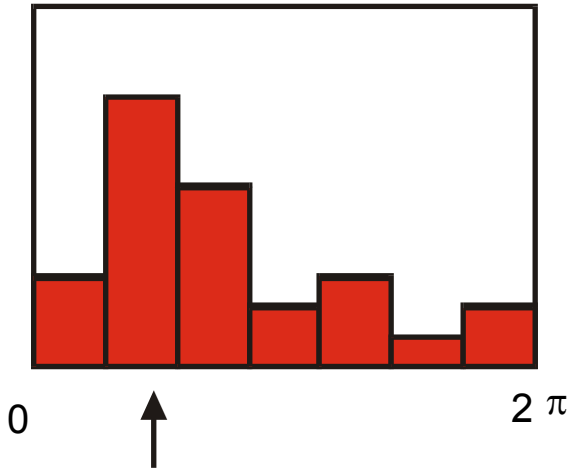
Гистограмма ориентаций градиентов



- Вычислим направление градиента в каждом пикселе
- Квантуем ориентации градиентов на 8 ячеек (направлений)
  - Пометим каждый пиксель номером ячейки
- Посчитаем гистограмму направлений градиентов
  - Для каждой ячейки посчитаем количество пикселов с номером этой ячейки
- Можем нормализовать гистограмму

Гистограмма ориентаций градиентов





- Histogram of oriented gradients (HOG)
- Мощный класс признаков
- Очень широко используется при анализе изображений
- Устойчив к изменениям освещенности изображения
  - Т.к. считаем только направления градиентов

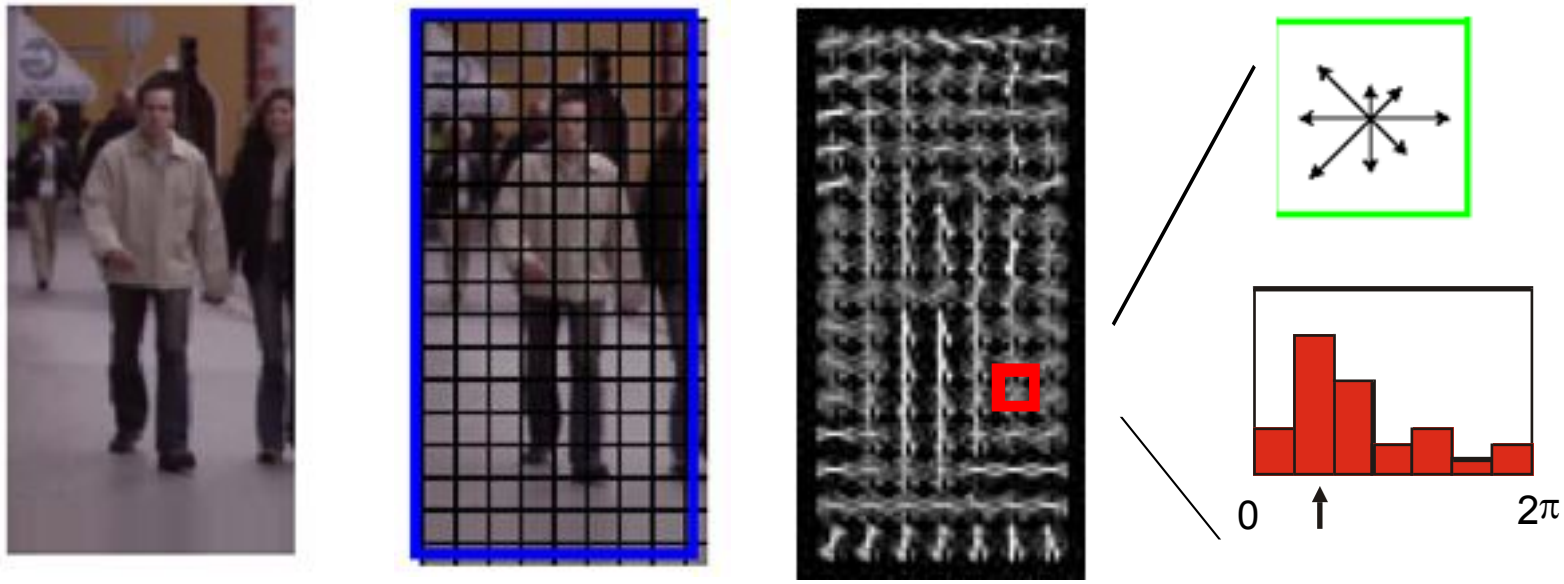
Алгоритм поиска пешеходов

- Алгоритм поиска:
  - Скользящее окно поиска
  - Признаки «гистограммы ориентаций градиентов»
  - Бинарный классификатор «пешеход?» SVM

- Хотя схема HOG + SVM изначально была предложена для пешеходов, она успешно применялась в дальнейшем к разным категориями объектов

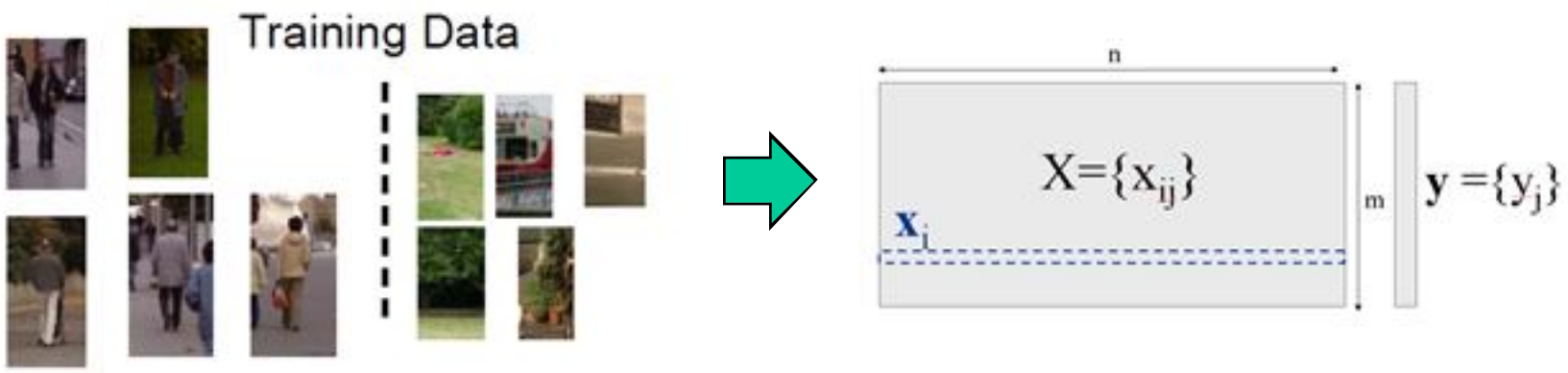
Navneet Dalal, Bill Triggs,Histograms of Oriented Gradients for Human Detection, CVPR 2005

Вычисление признаков



- 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 x 8 = 1024 признака

Обучение классификатора



- Соберём обучающую выборку фрагментом изображения с пешеходами и без
- Для каждого фрагмента посчитаем вектор признак x и метку y
- На полученной обучающей выборке обучим линейный классификатор SVM

SVM

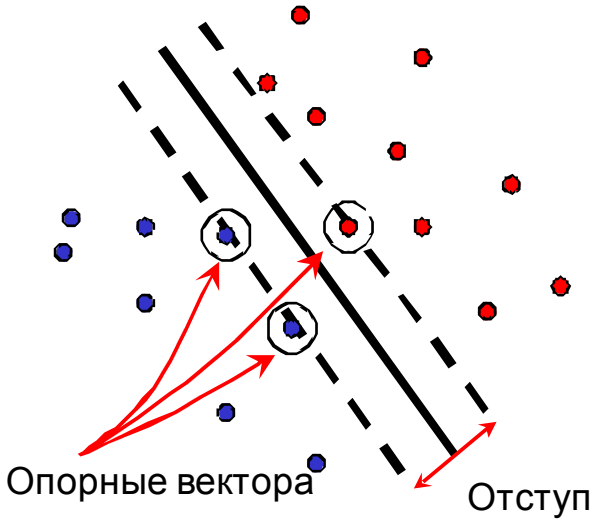
• Обучаем линейную SVM:

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

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

• Опорные вектора с положительными и отрицательными весами

• Чем фактически в нашем случае являются ***x*** ?



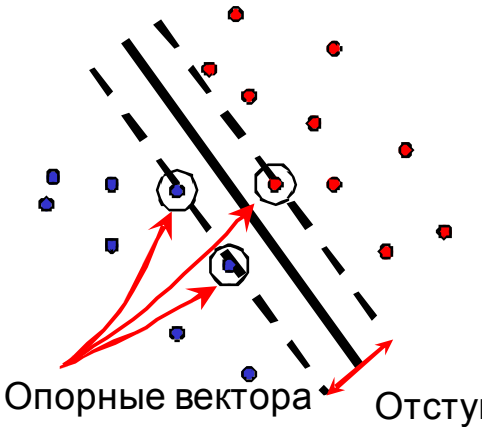
SVM

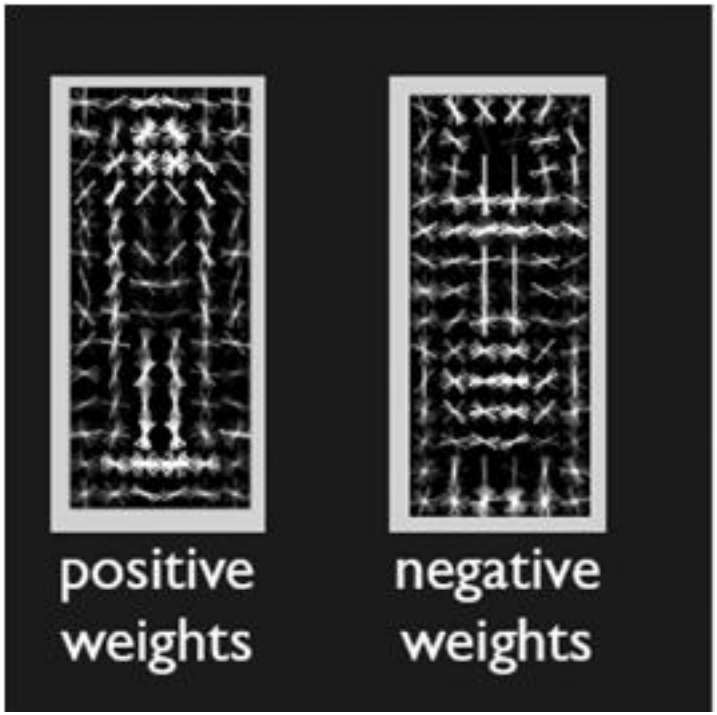
• Каждый опорный вектор – это вектор-признак одного из примеров

• Опорные вектора с положительными и отрицательными весами

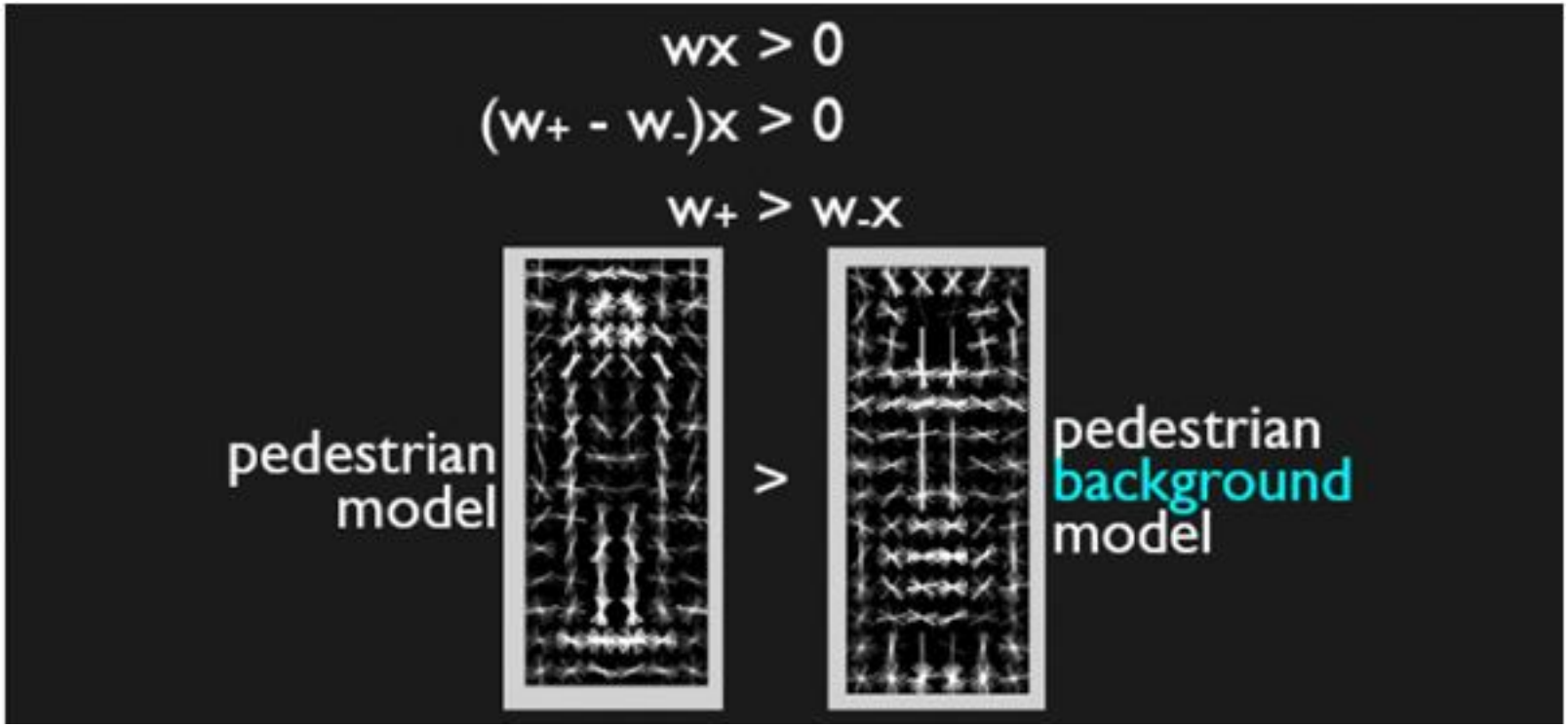
• Положительный вес – пример фрагмента, содержащего пешехода

• 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
- Похоже на многократную линейную фильтрацию изображений

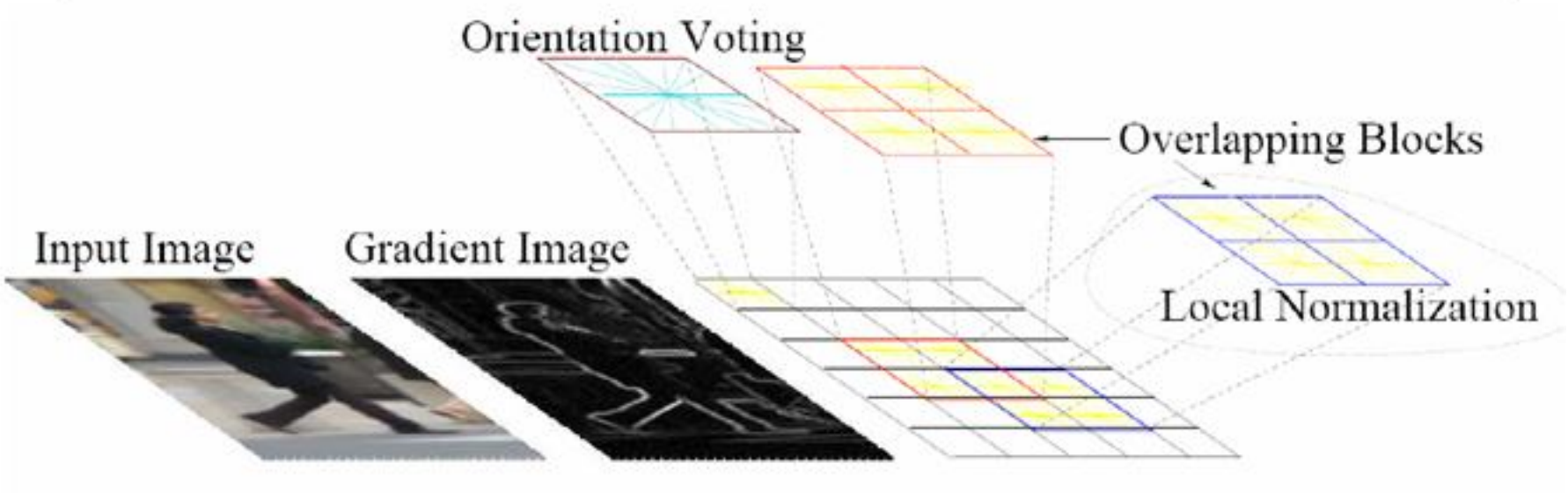
Source: Deva Ramanan

«Детектор пешеходов»

- Получили работающий «детектор пешеходов»
- Базовый алгоритм:
  - Сканирующее окно
  - Вычисление признаков (гистограмм ориентаций градиентов)
  - Классификация с помощью SVM

- Базовый алгоритм можно существенно улучшить простыми способами

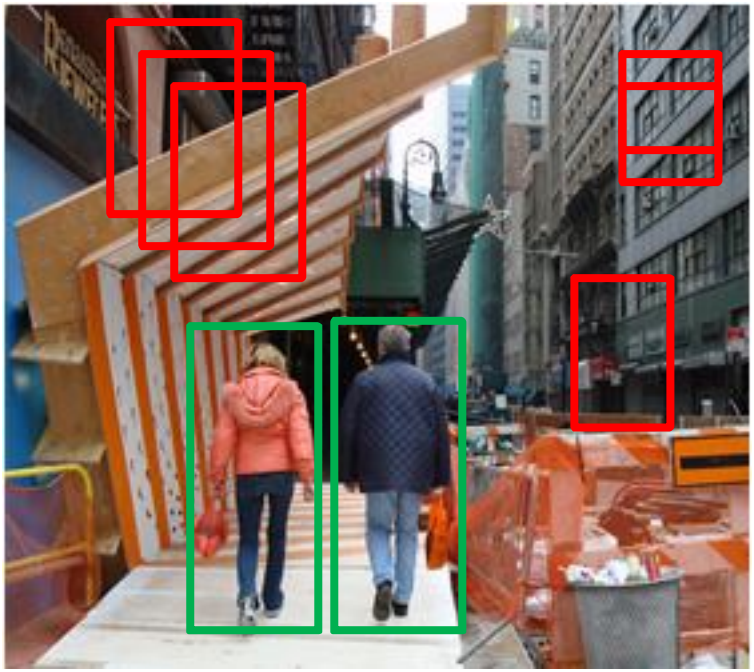
Усложнение схемы



- Добавляем второй уровень:
  - Выделяем блоки 2x2 с перекрытием
  - Нормализуем гистограммы в каждом блоке
  - Это дает большую пространственную поддержку
- Размер вектора = 16 x 8 (сетка) x 8 (ориентаций) x 4 (блоки с перекрытием) = 4096

Обучение детектора

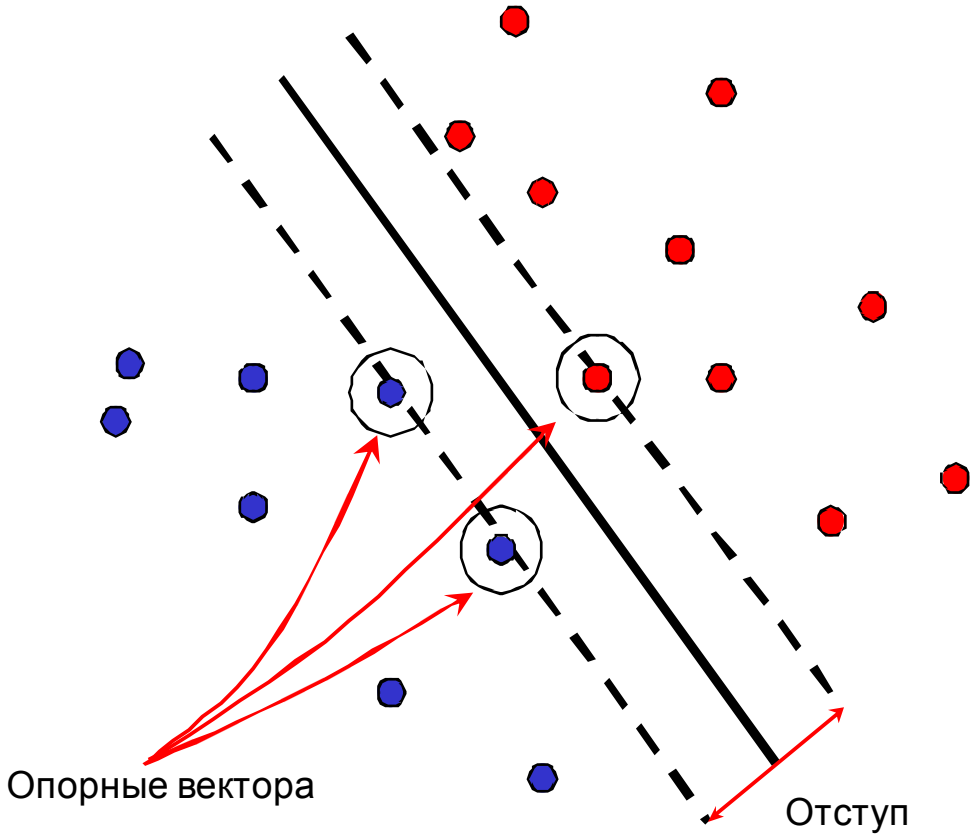
- Сколько на изображении объектов «пешеход» и сколько фрагментов фона?



- 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

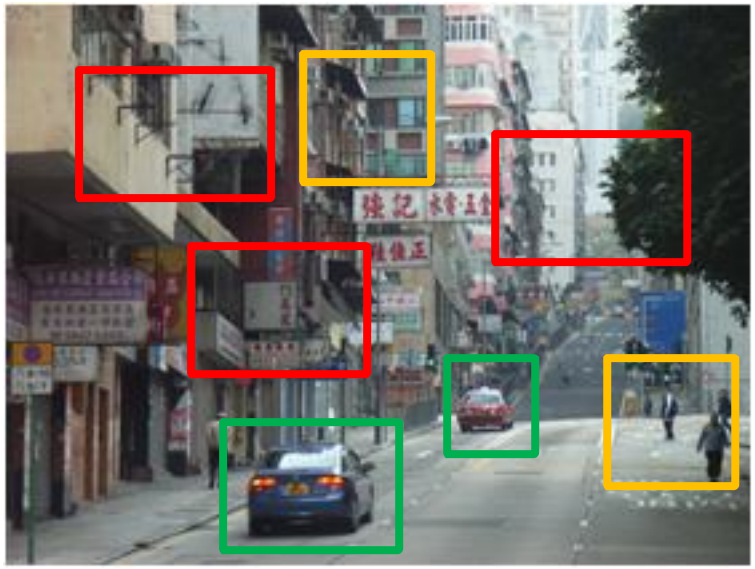
Какие нам нужны примеры фона для получения наилучшего детектора?



Самые трудные!

Как их найти?

Бутстраппинг (Bootstrapping)



- 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

I. Laptev "Improvements of Object Detection Using Boosted Histograms« BMVC 2006

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 размеченным объектами
- На каждом кадре отмечены 1-3 человека, всего 1607 людей, это маловато

Положительные окна

Посмотрим, что отметили люди при разметке:

Внимание: похожие положение и ориентация!

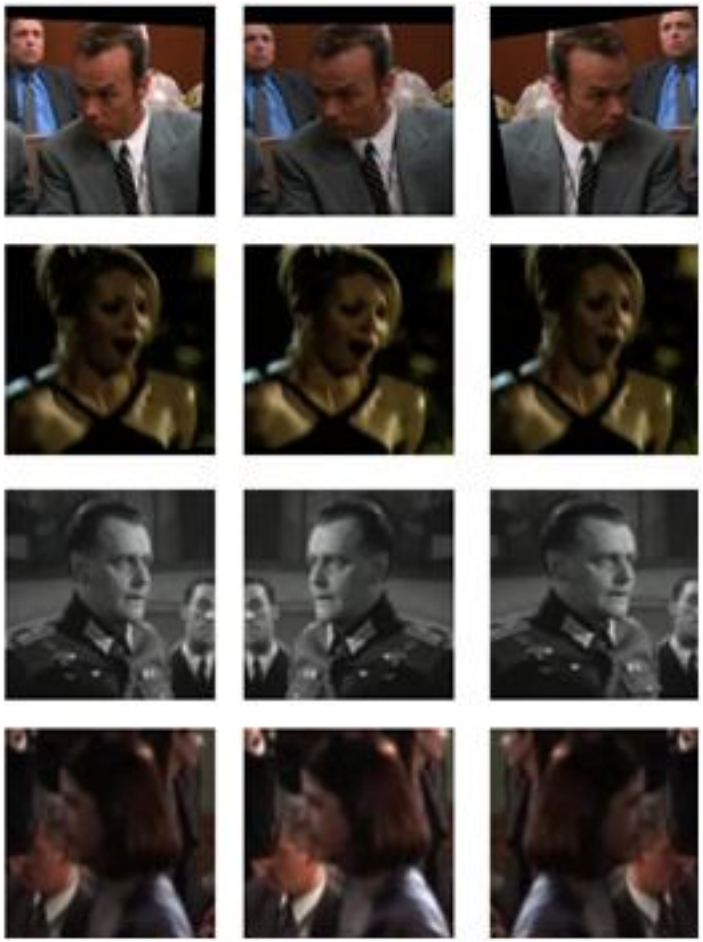
Искаженные примеры

Давайте «размножим» данные, «пошевелив» их:

Небольшие сдвиги, отображения, повороты, изменения масштаба

Искаженные примеры

Из 1607 эталонных примеров получили ~32000 искаженных (jittered) примеров



Сколько отрицательных примеров можно набрать из 1100 кадров?

- Гораздо больше 32k.

Воспользуемся схемой бутстраппинга для поиска «трудных примеров» фона

Первая итерация

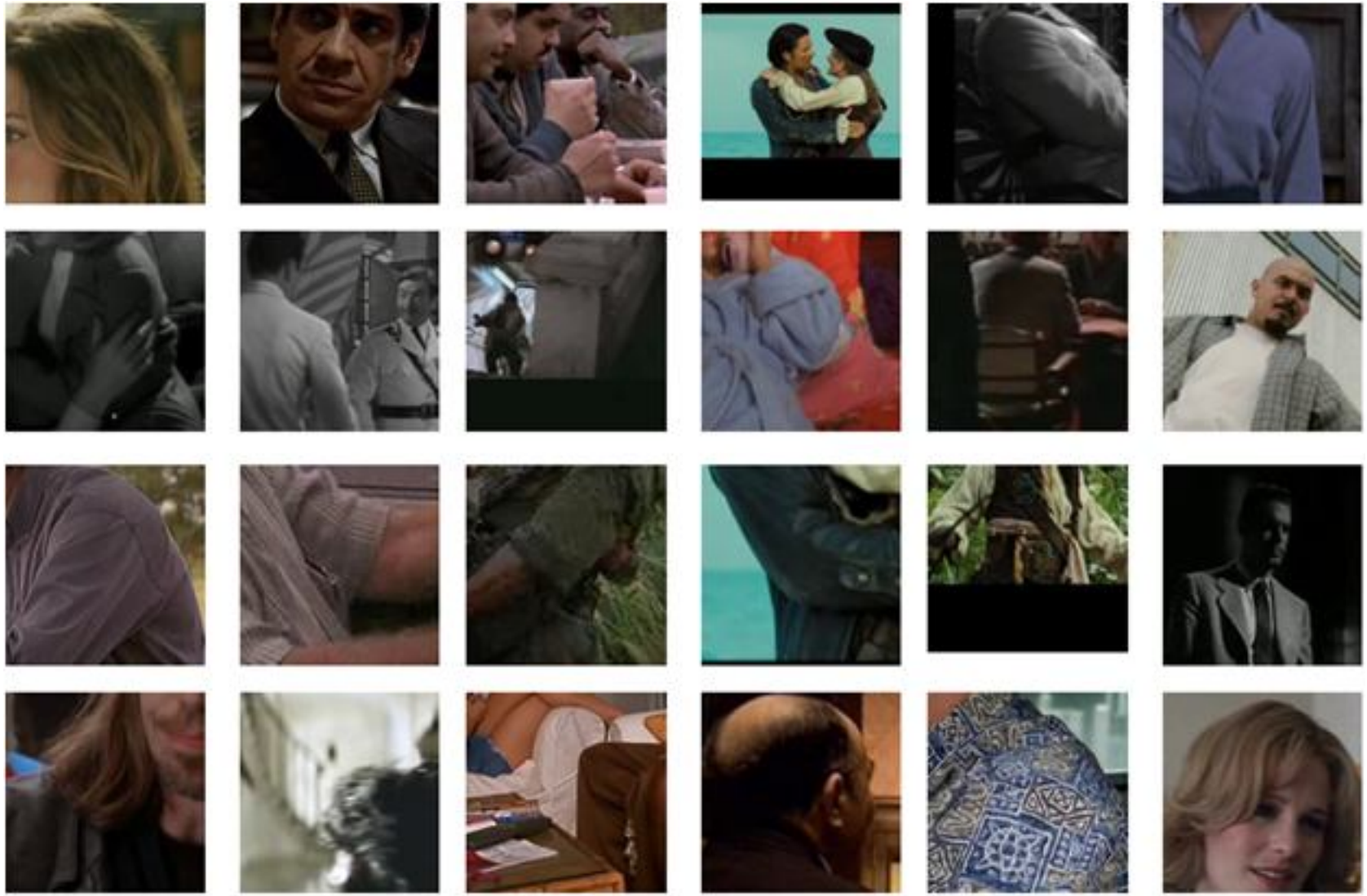
Соберём случайные фрагменты фона:

Первая стадия

Трудный отрицательный пример

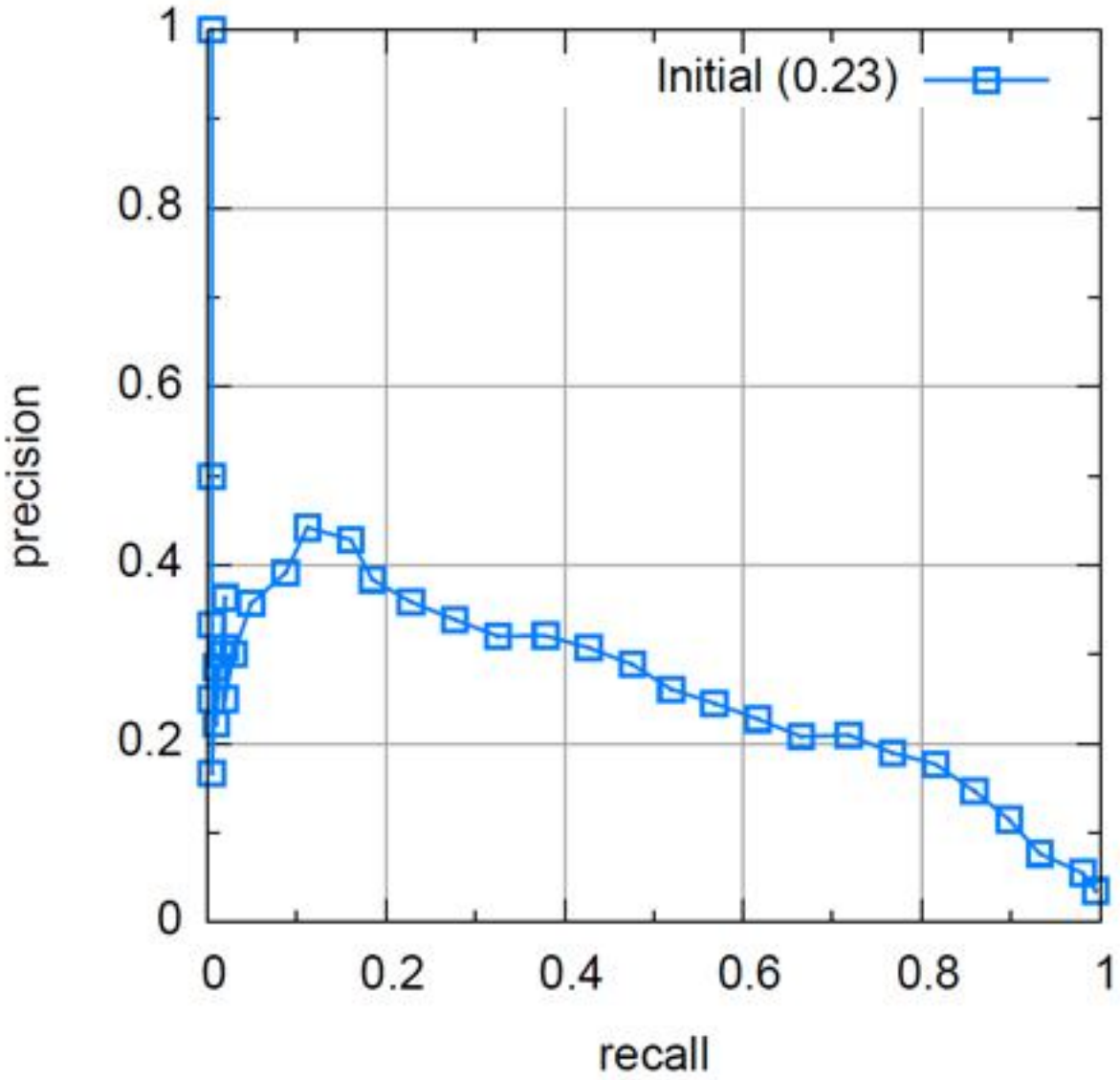
- Обучим детектор по случайной выборке фона
- Применим его к исходным изображениям
- Ищем ложные обнаружения с высоким рейтингом
- Используем их как трудные отрицательные примеры
- Затраты: # количество изображений x поиск в каждом

Трудные примеры

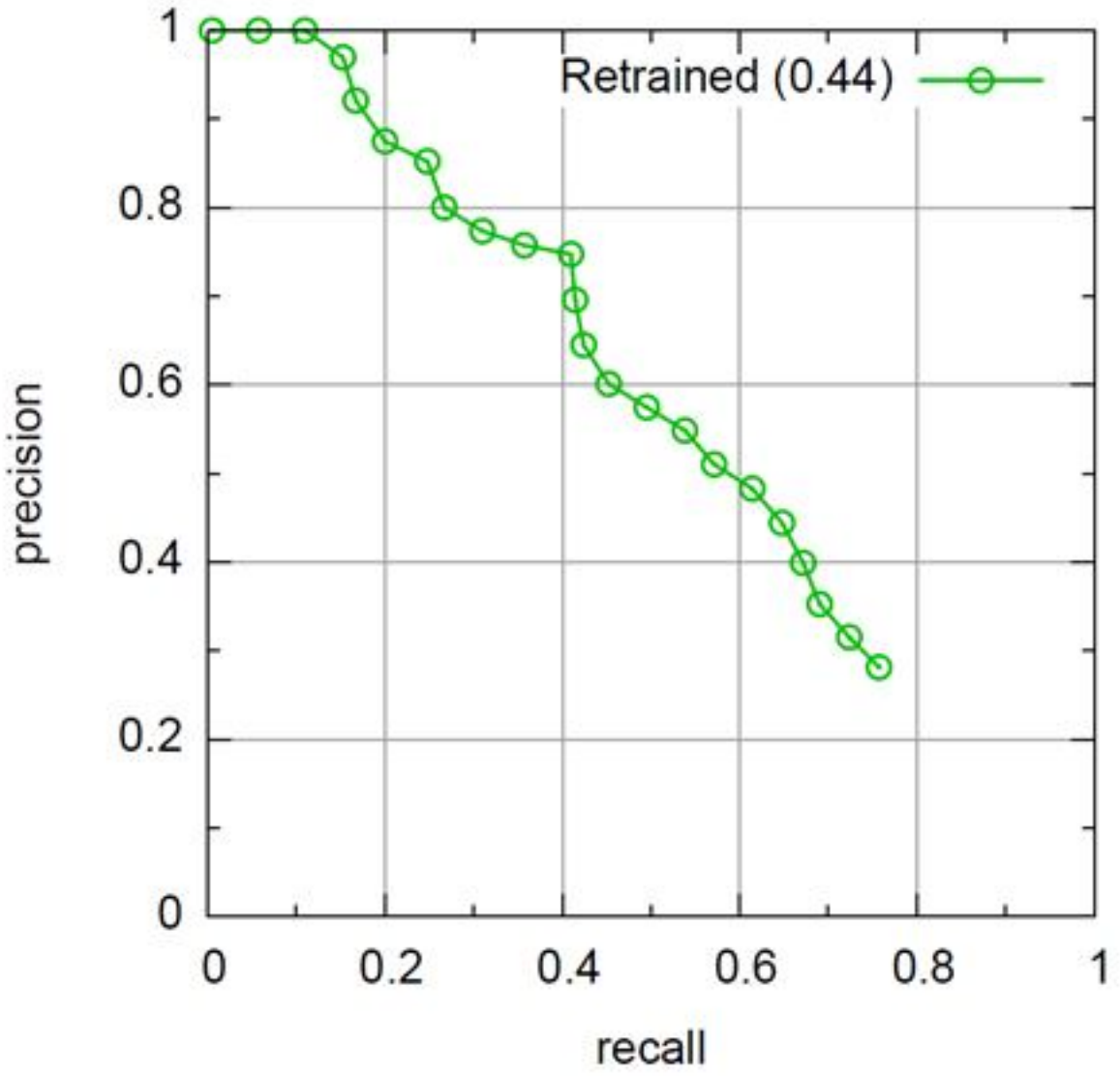


Трудные примеры

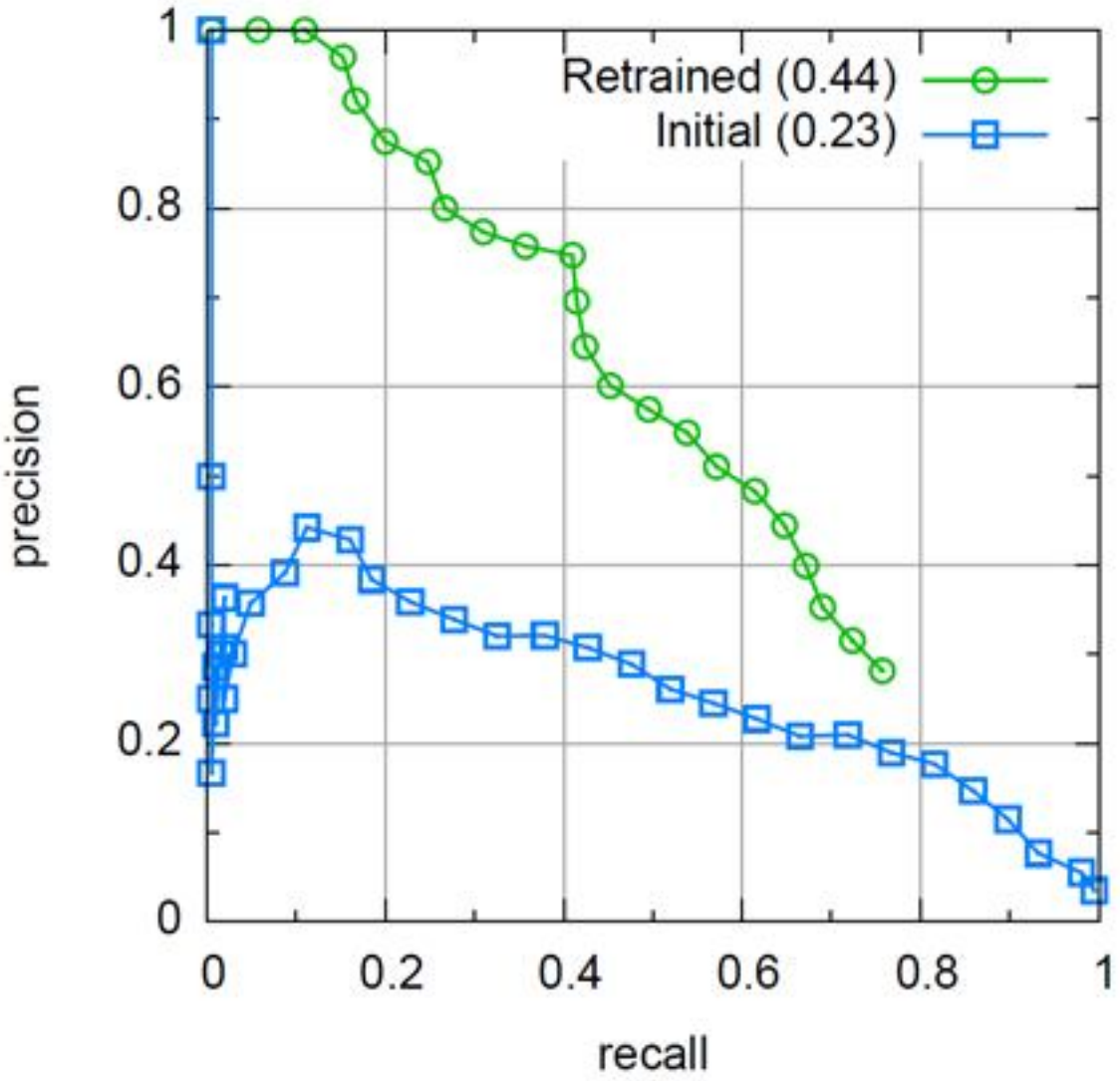
Измерение качества



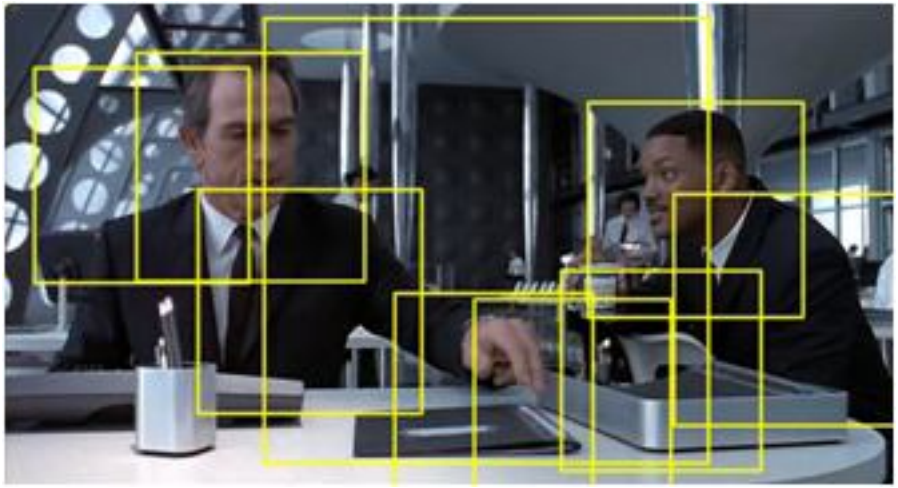
После перетренировки

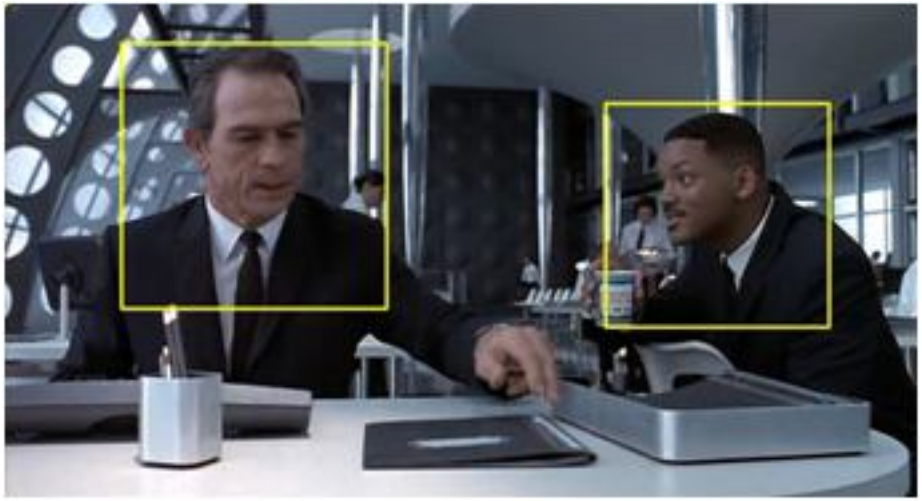


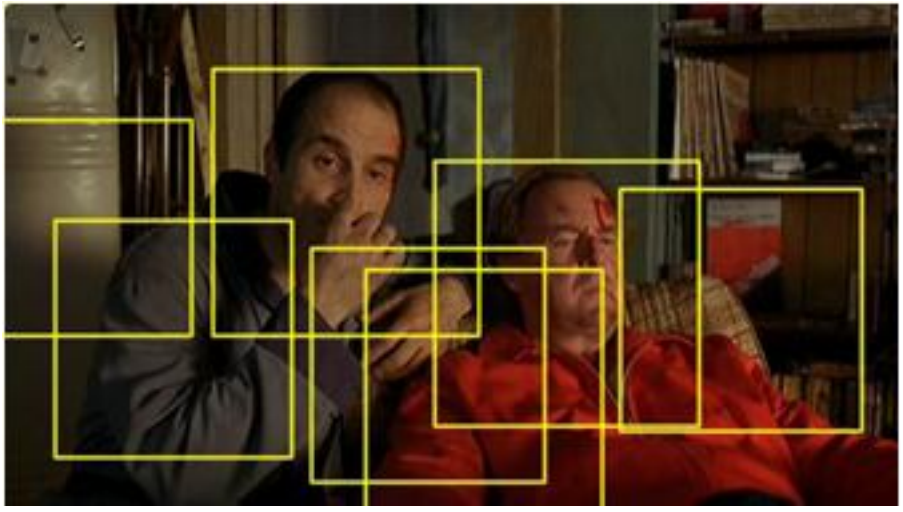
Сравнение

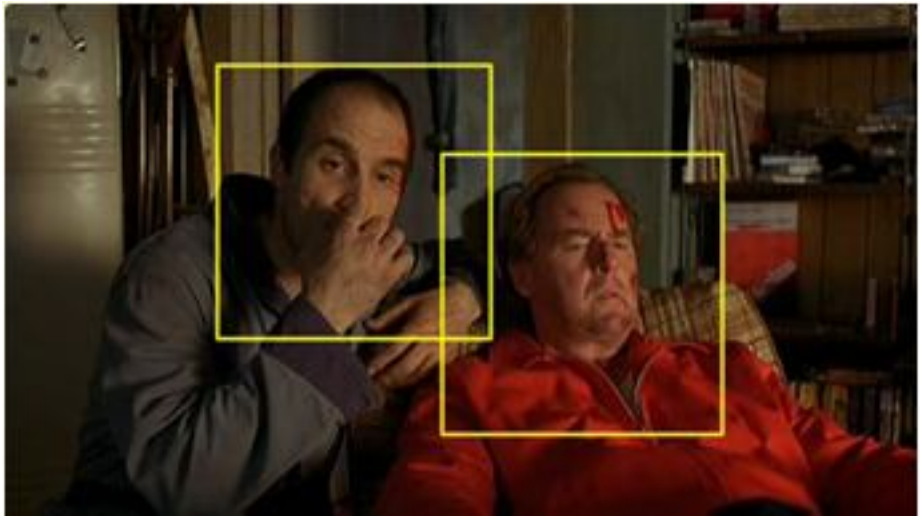


Сравнение

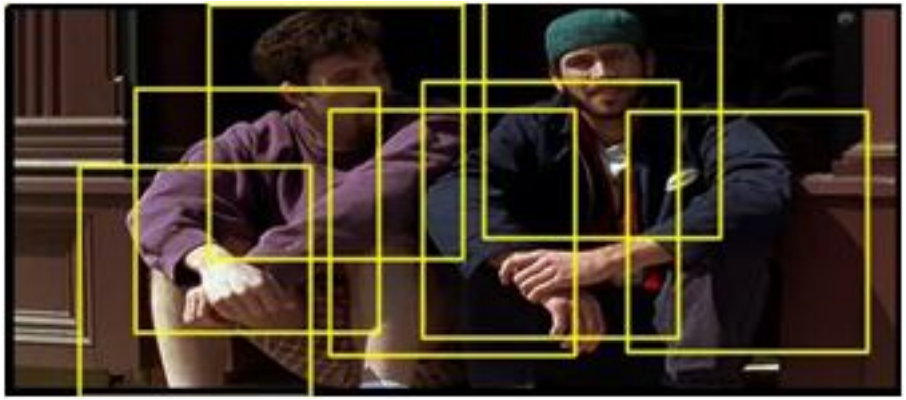


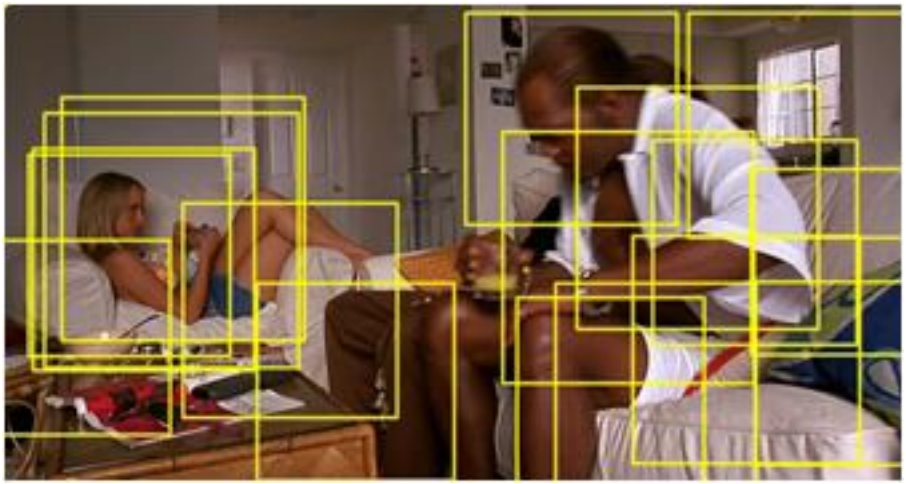




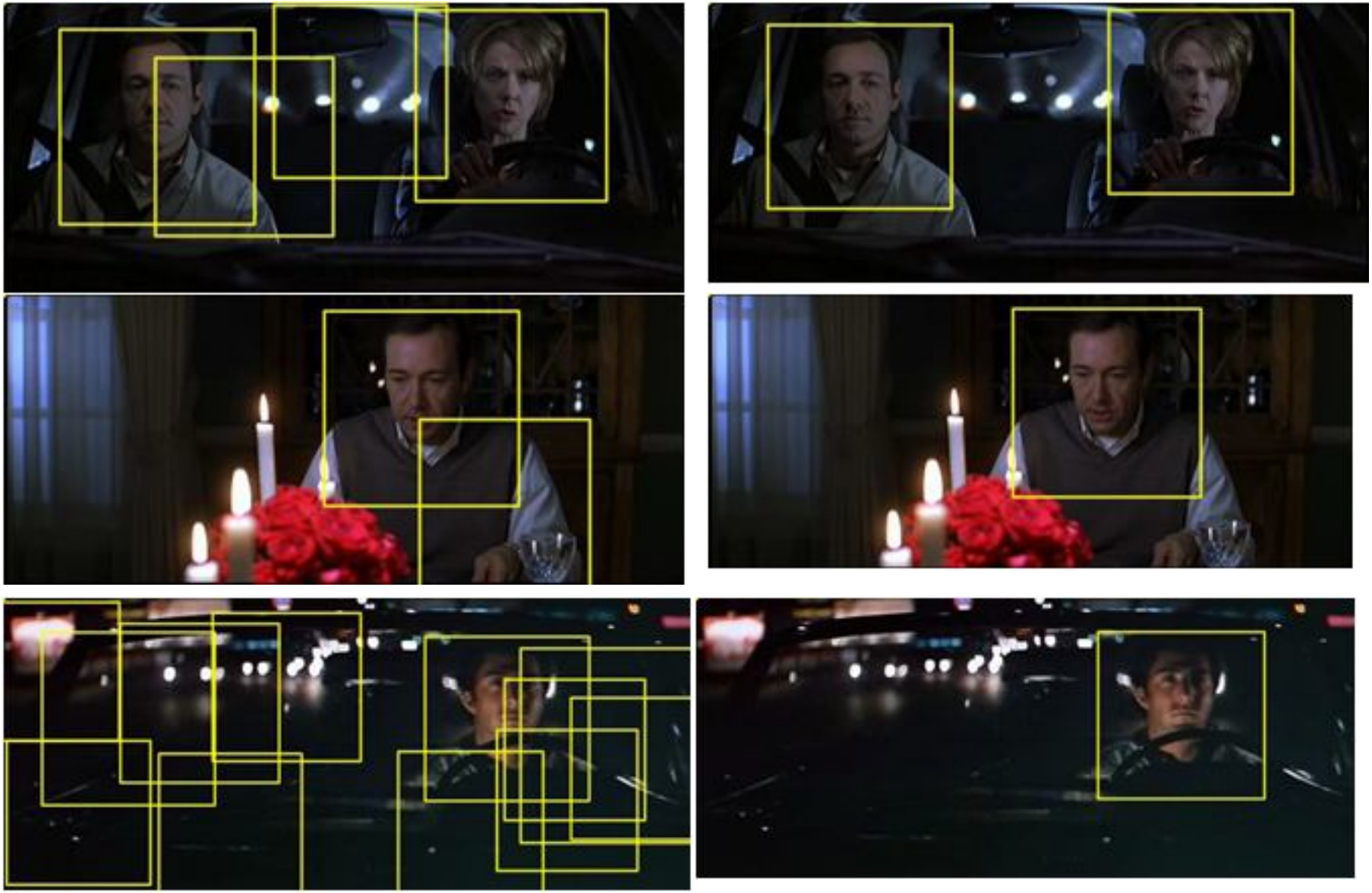


Сравнение





Сравнение

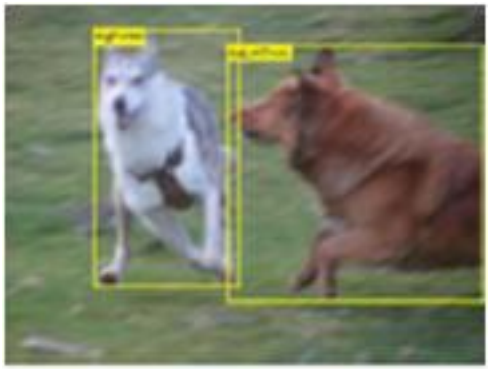


Резюме алгоритма

- 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

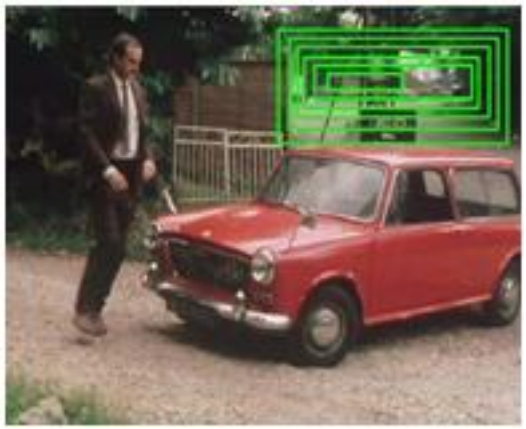
Проблемы скользящего окна

Разный размер объектов



Соотношение размеров окна

Перекрытие и плохое соответствие формы объекта окну

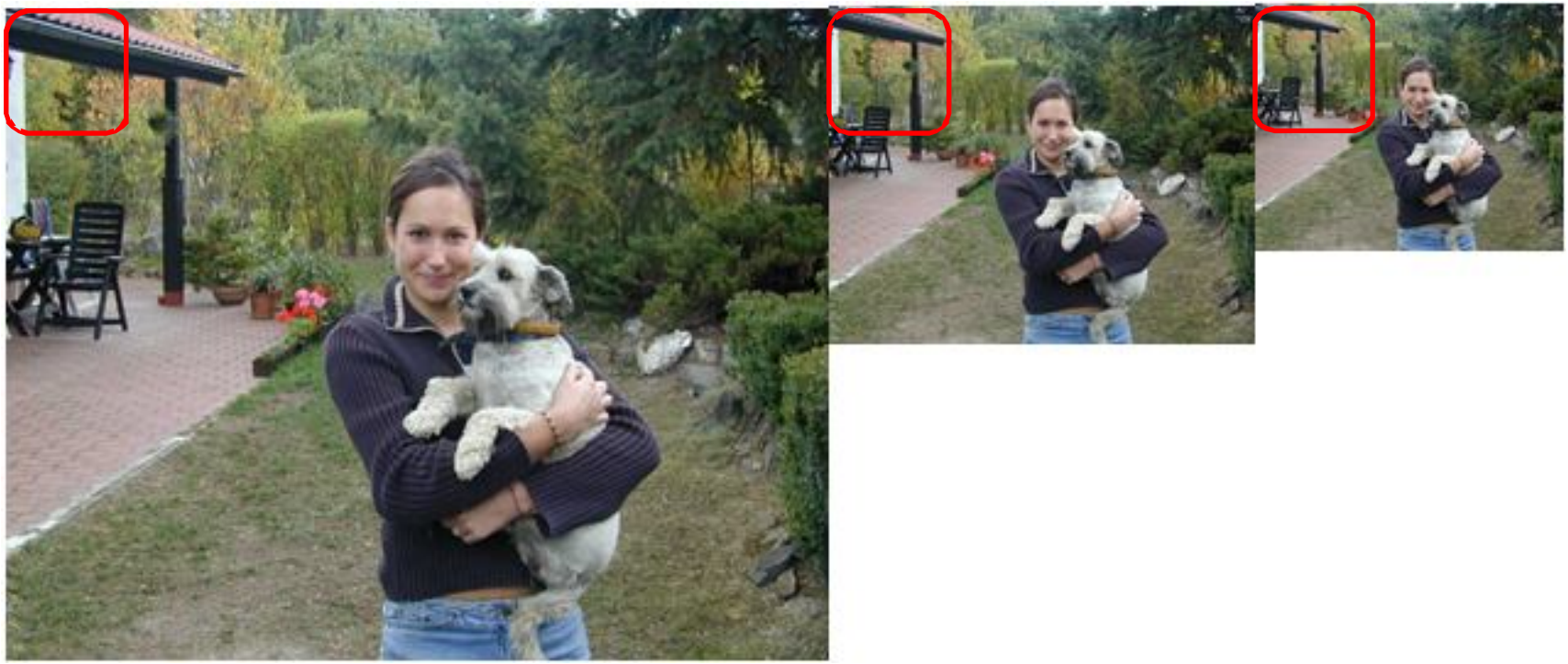


Множественные отклики

Разный размер объектов

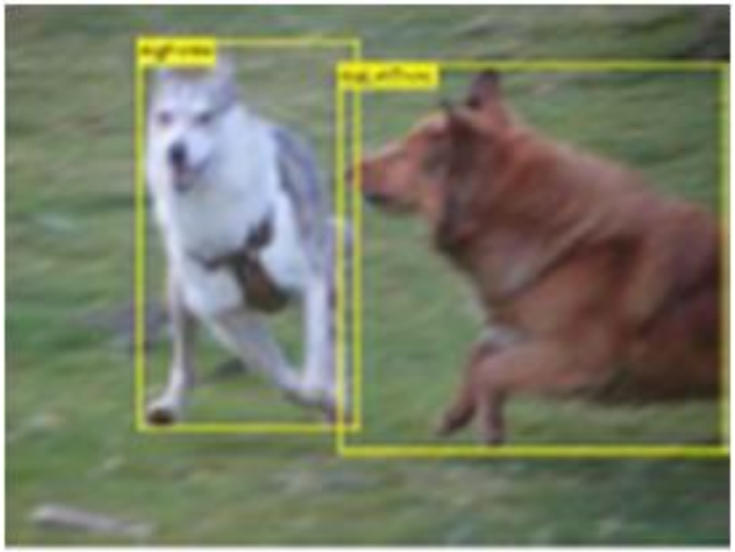
Сканируем изображение рамками разного размера и разных пропорций

Разный размер объектов



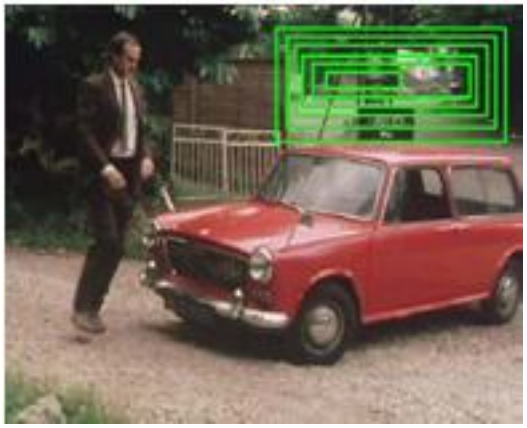
- Вместо изменения размеров рамки можем построить «пирамиду» изображений разных размеров и сканировать одной рамкой
- Сколько потребуется масштабов?
  - На практике до 50-70

Пропорции объектов

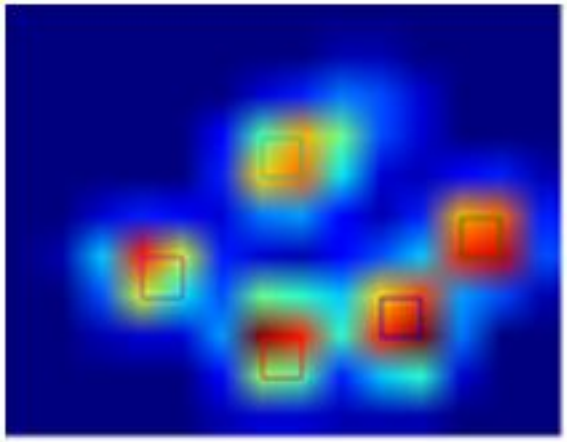


- Будем сканировать рамками разных пропорций
- С учетом масштабов нам придётся сканировать $N_{рамок}$ x $N_{масштабов}$ раз
- Поэтому общее количество сканирований может достигать 150-200

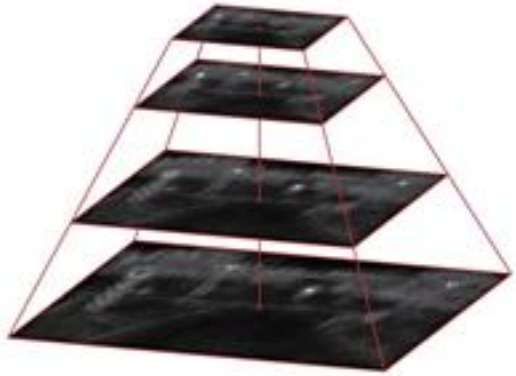
Множественные отклики



Множественные отклики



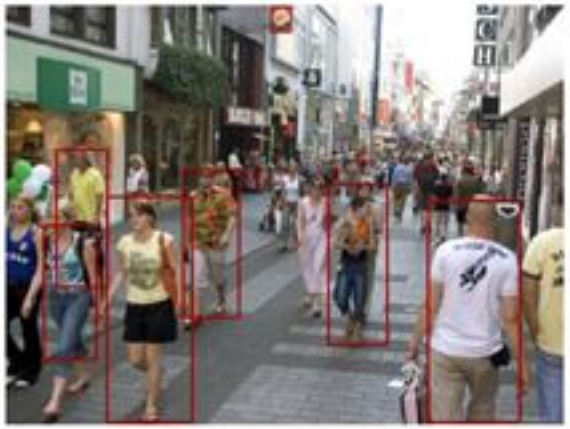
Карта откликов



Пирамида изображений

- В одном масштабе может быть множество откликов разной силы
- Мы выберем из них точки, в которых значение достигает локального максимума («Подавление не-максимумов»)
- При наличии нескольких масштабов нужно выбирать максимальные значения по 3х мерной области
- Параметры выделения максимумов могут значительно влиять на результат!

Перекрытия



Самая сложная проблема!

Требования к детектору

- Для изображения в 1МП нужно просмотреть порядка 1М окон (например, для случая лиц)
- На одном изображении обычно 0-10 лиц

- Чтобы избежать ложных обнаружений (false positives) ошибка 2го рода должна быть ниже 10^{-6}
- Нужно быстро отбрасывать ложные окна

Детектор Viola-Jones

- Основополагающий метод для поиска объектов на изображении в реальном времени
- Обучение очень медленное, но поиск очень быстр
- Основные идеи:
  - *Признаки Хоара в качестве слабых классификаторов*
  - *Интегральные изображения* для быстрого вычисления признаков
  - *Бустинг* для выбора признаков
  - *Каскад (Attentional cascade)* для быстрой отбраковки окон без лица

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

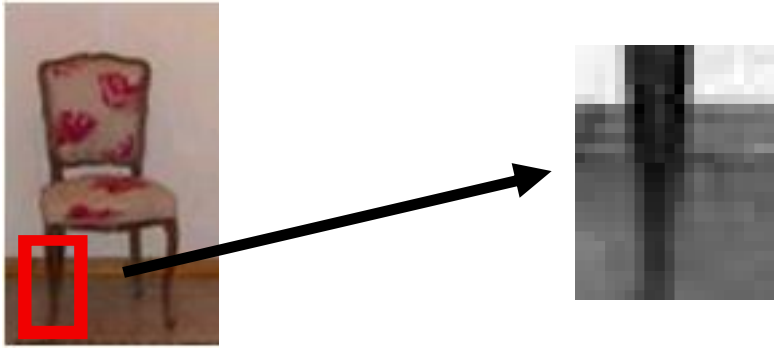
Вспомним сопоставление шаблонов

Найти стул в изображении

Что если воспользоваться маленькими фрагментами?
Искать не стул, а часть стула?

Части объектов

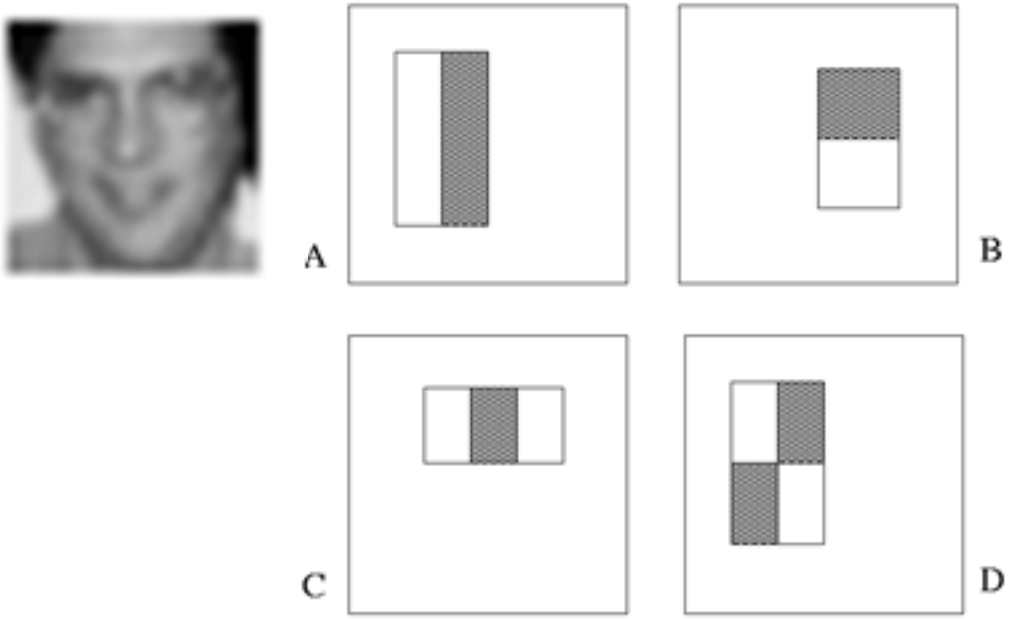
Что если воспользоваться маленькими фрагментами? Искать не стул, а часть стула?



Не так плохо, срабатывает на ножках

Признаки Хоара

“Прямоугольные фильтры”



Value =

∑ *(pixels in white area)* –
∑ *(pixels in black area)*

Слабые детекторы/классификаторы

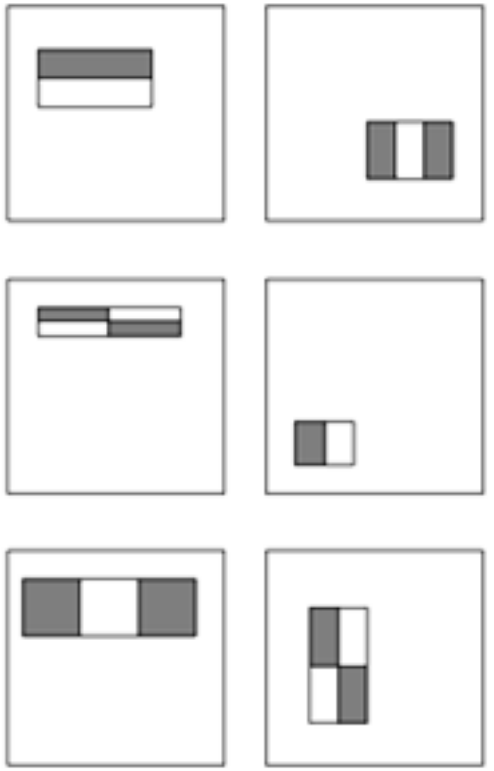
- Определяем слабые классификаторы на основе прямоугольных признаков

Значение признака

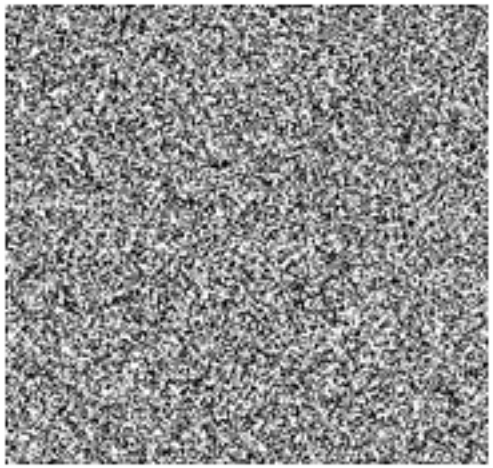
$$h_t(x) = \begin{cases} 1 & \text{if } f_t(x) > \theta_t \\ 0 & \end{cases}$$

окно

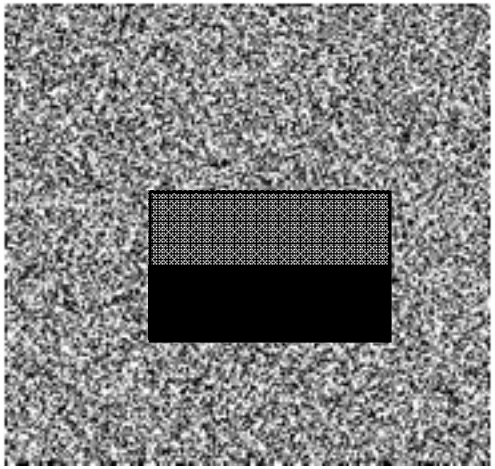
порог



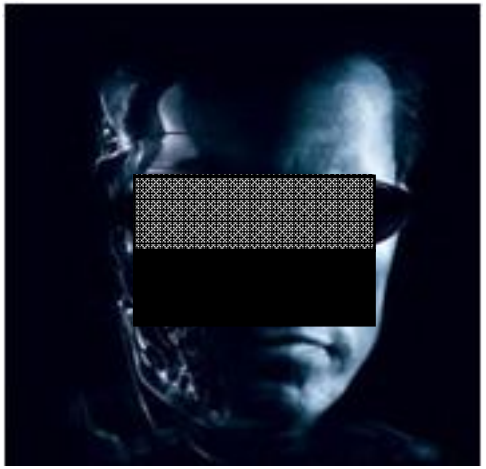
Пример



Source

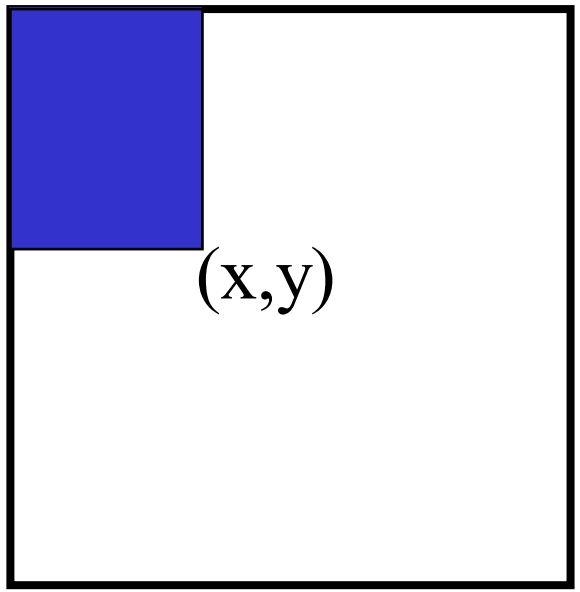


Result

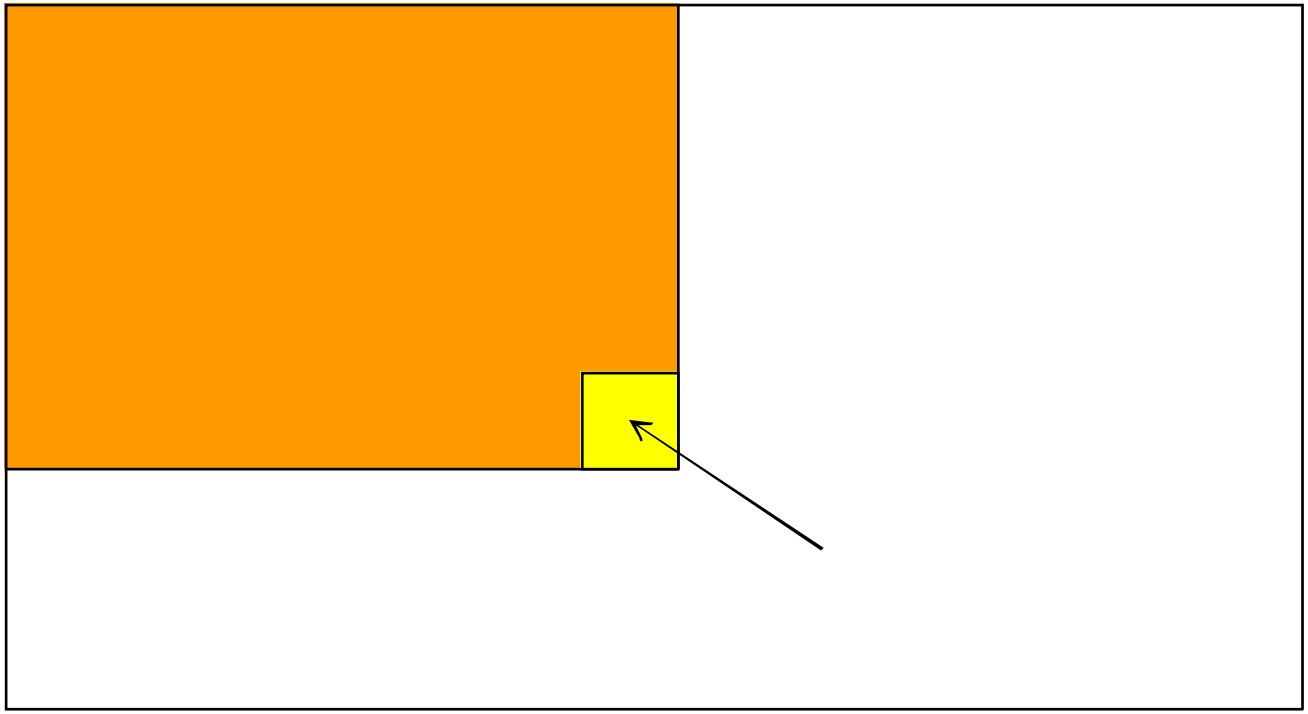


Интегральные изображения

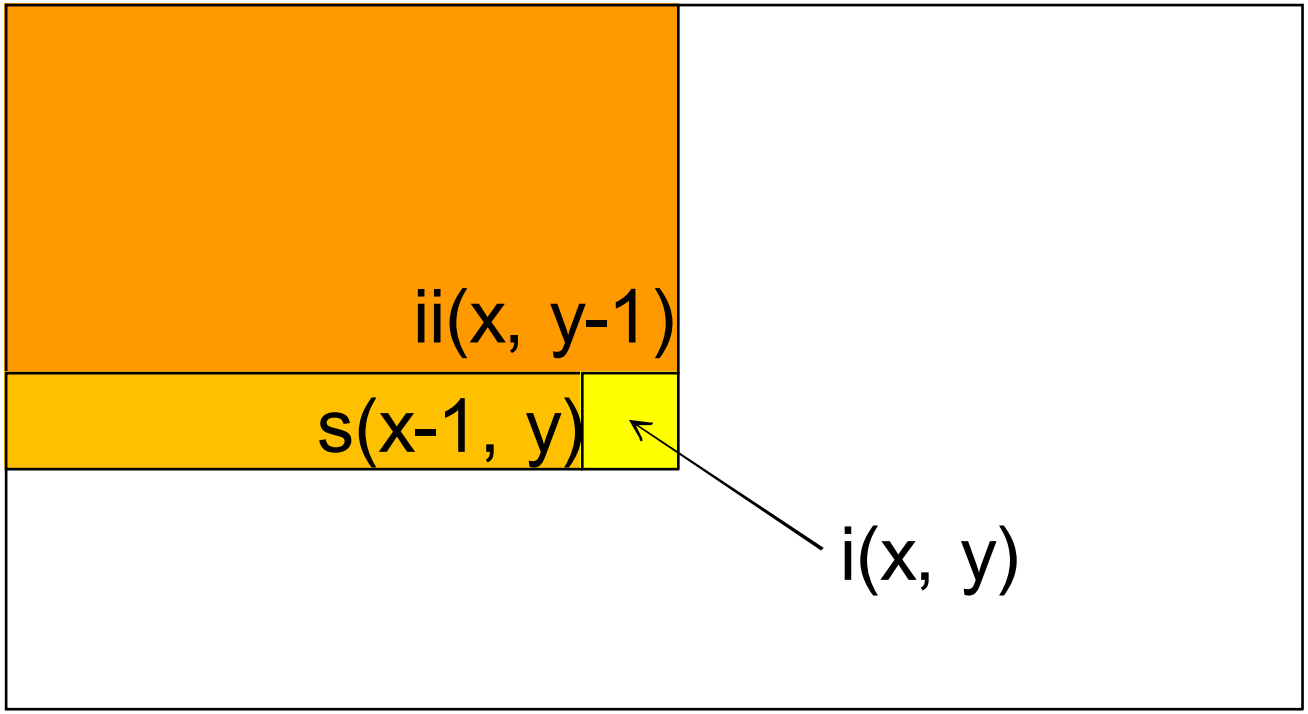
- Значение каждого пиксела (x,y) равно сумме значений всех пикселов левее и выше пикселя (x,y) включительно
- Интегральное изображение рассчитывается за один проход



Вычисление интегрального изображения



Вычисление интегрального изображения



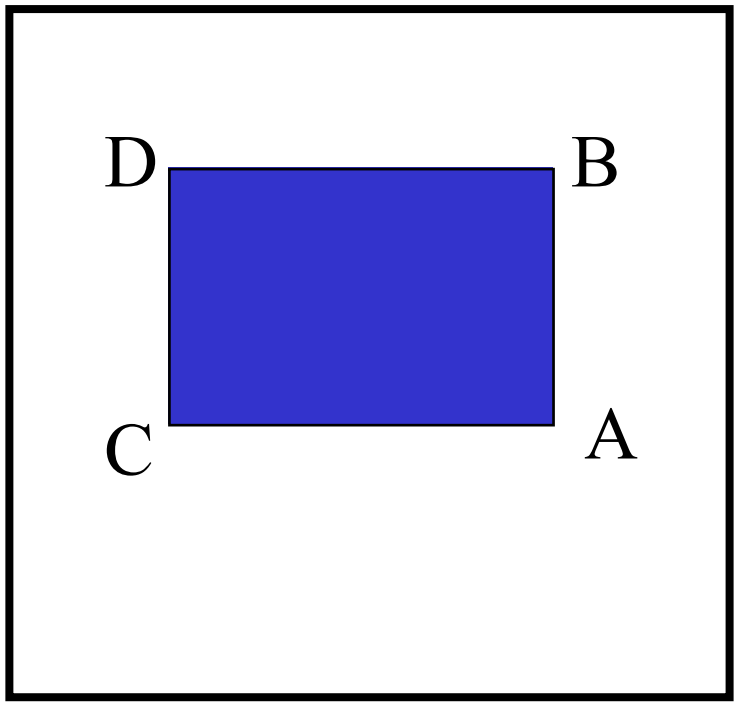
- Сумма по строке: $s(x, y) = s(x–1, y) + i(x, y)$
- Интегральное изображение: $ii(x, y) = ii(x, y−1) + s(x, y)$

MATLAB: ii = cumsum(cumsum(double(i)), 2);

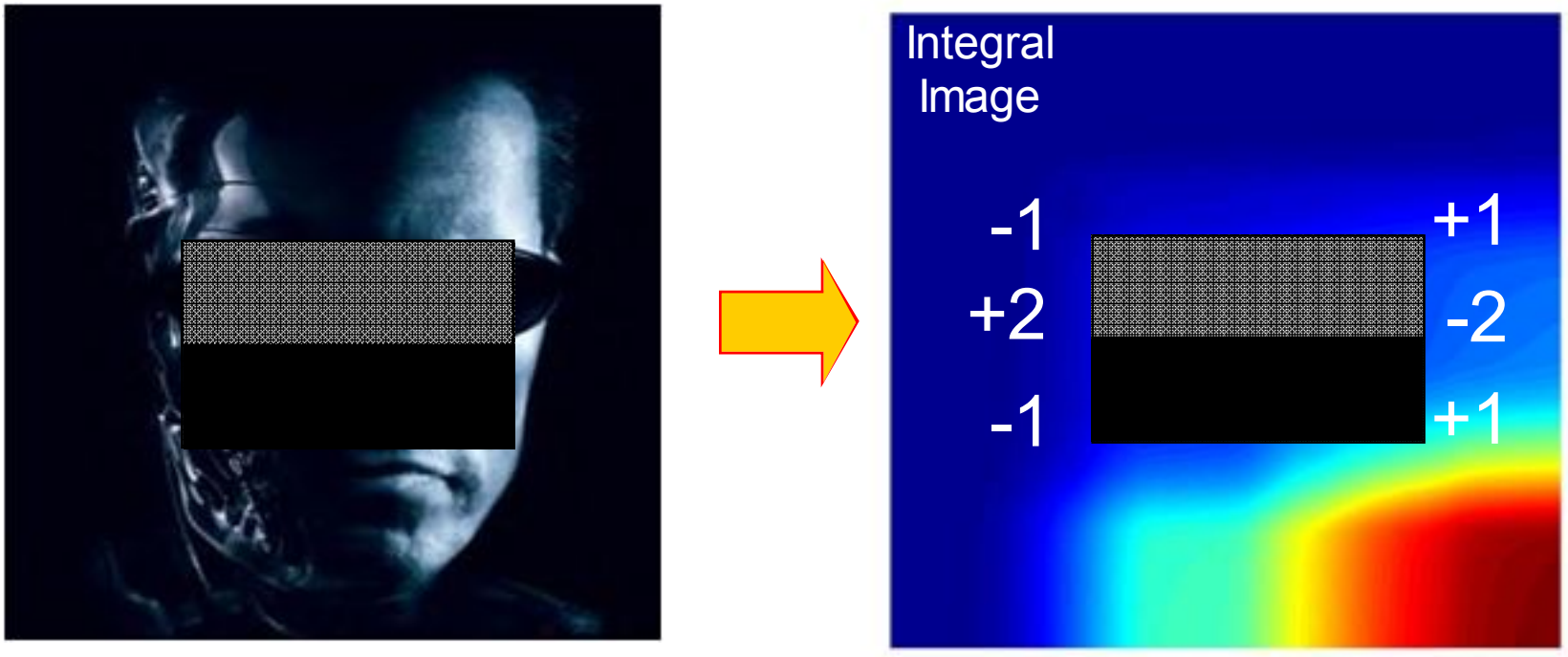
Вычисление суммы в прямоугольнике

- Пусть A,B,C,D – значения интегрального изображения в углах прямоугольника
- Тогда сумма значений пикселов в исходном изображении вычисляется по формуле:

  sum = A – B – C + D
- 3 операции сложения для любого прямоугольника

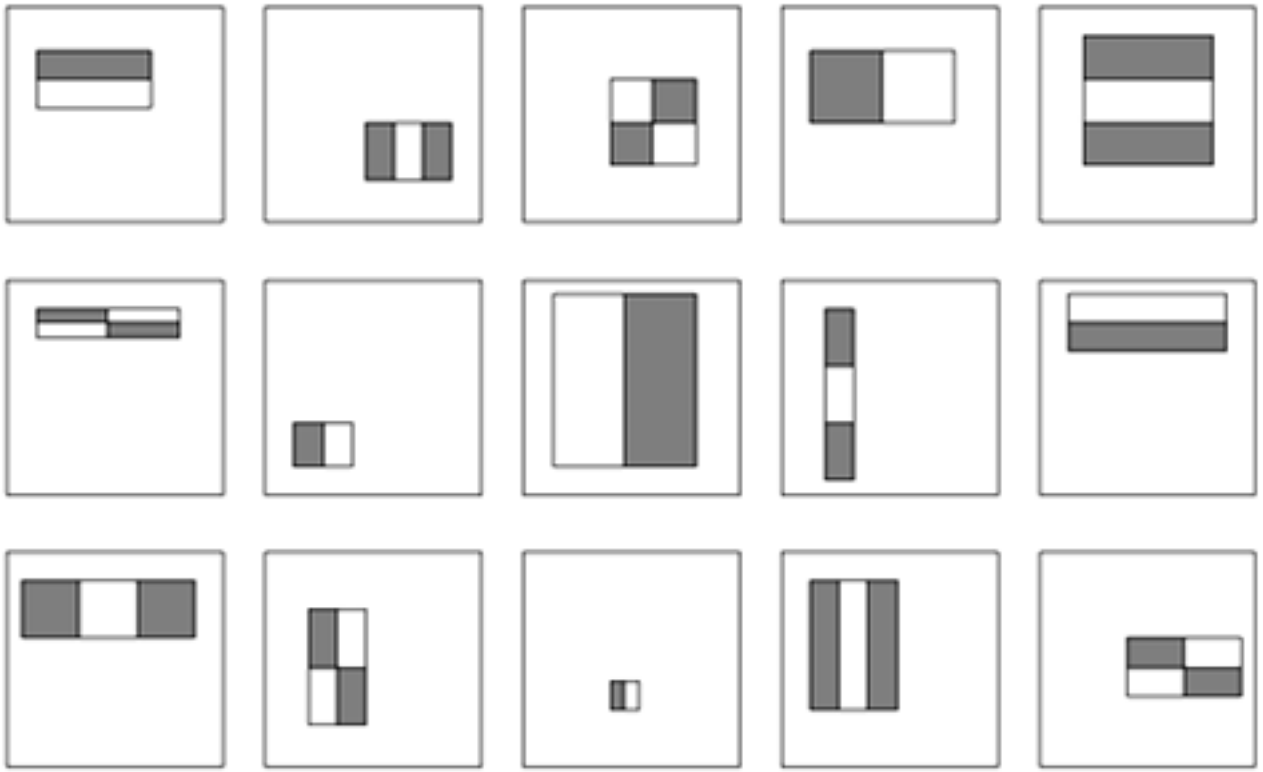


Пример



Количество признаков

Для окна поиска 24x24 пиксела, число возможных прямоугольных признаков достигает ~160,000!



Выбор признаков

- Для окна поиска 24x24 пиксела, число возможных прямоугольных признаков (слабых классификаторов) достигает ~160,000!
- Каждый по отдельности очень «слабый»
- В процессе поиска вычислять все признаки нереально
- Хороший классификатор должен использовать небольшое подмножество всевозможных признаков
  - *«Комитетные классификаторы»*
- Будем выбирать нужные признаки и строить из них линейные комбинации с помощью бустинга

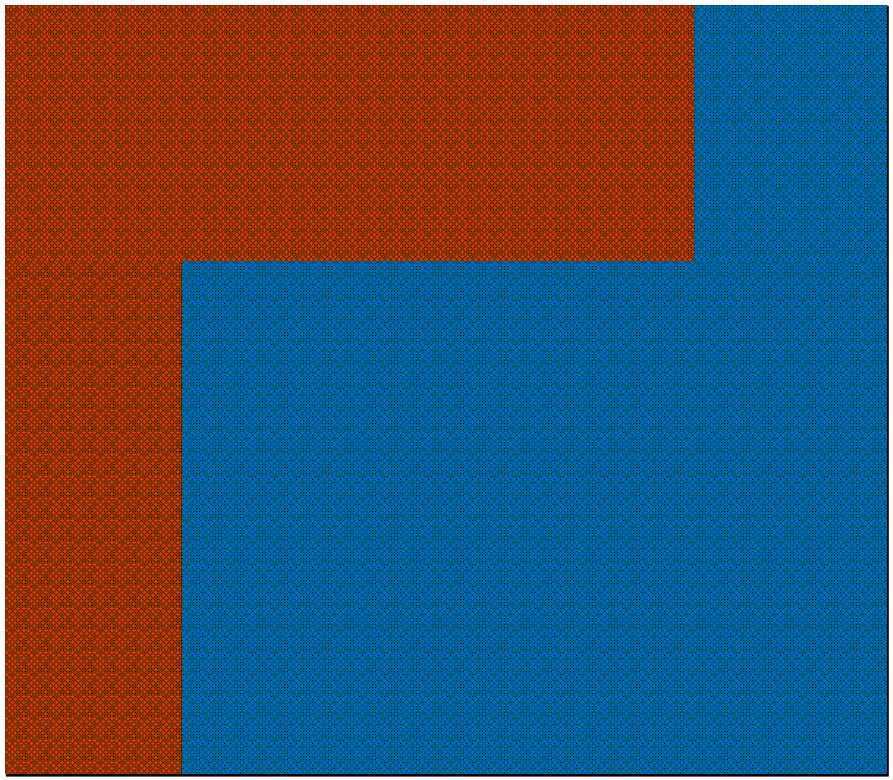
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Бустинг

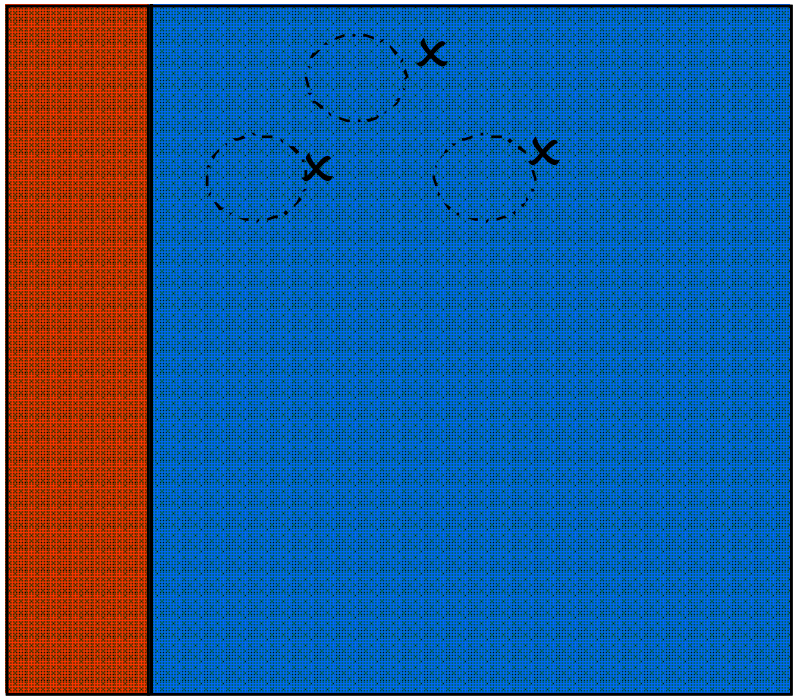
- Бустинг – схема построения сильного классификатора (комитета) за счёт комбинирования слабых классификаторов
  - Слабый классификатор должен быть лучше монетки
- Обучение состоит из нескольких этапов усиления (*boosting rounds)*
  - На каждом этапе выбираем слабый классификатор, который лучше всех сработал на примерах, оказавшихся трудными для предыдущих классификаторов
  - «Трудность» записывается с помощью весов, приписанных примерам из обучающей выборки
  - Составляем общий классификатор как взвешенную линейную комбинацию слабых классификаторов

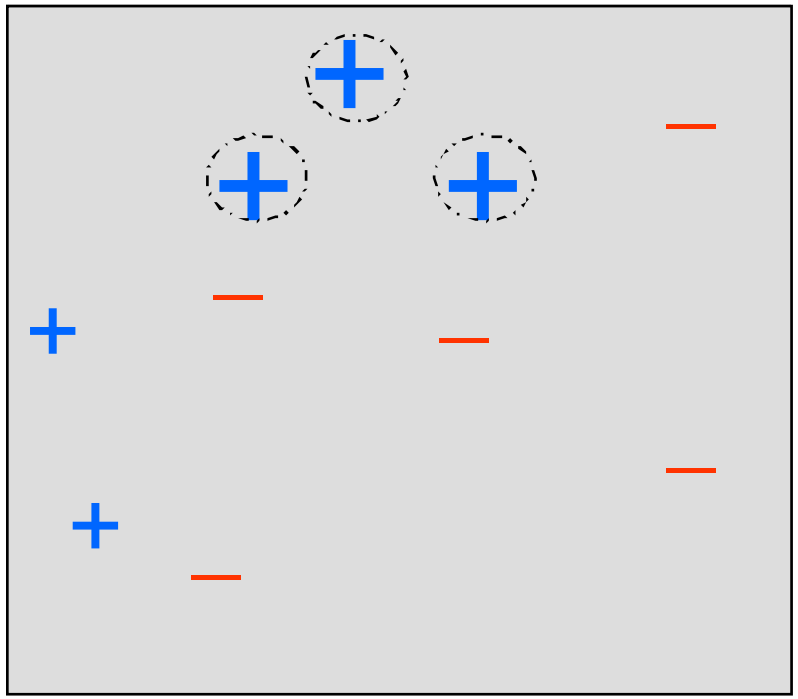
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Пример хорошего классификатора



Итерация 1 из 3





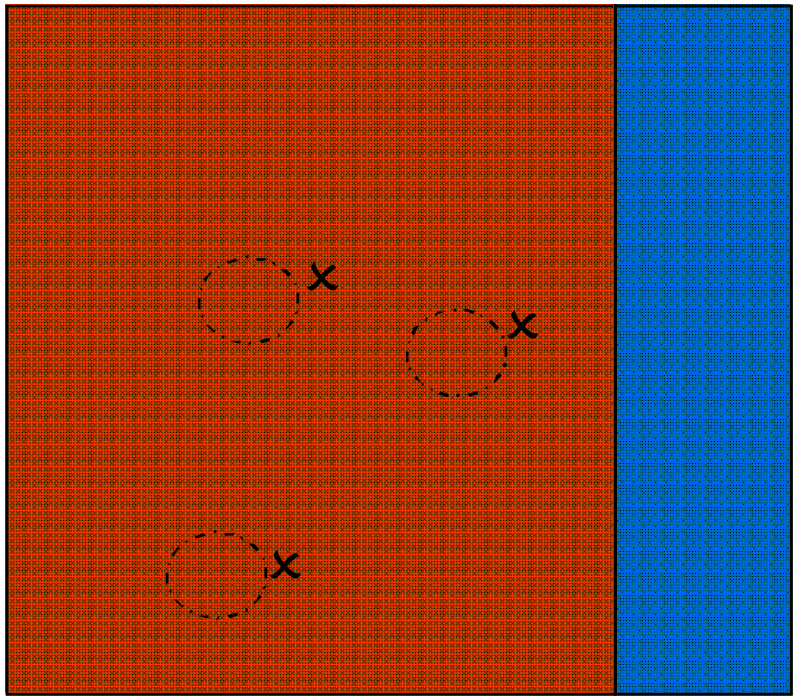
h_1

$\varepsilon_1 = 0.300$

$\alpha_1 = 0.424$

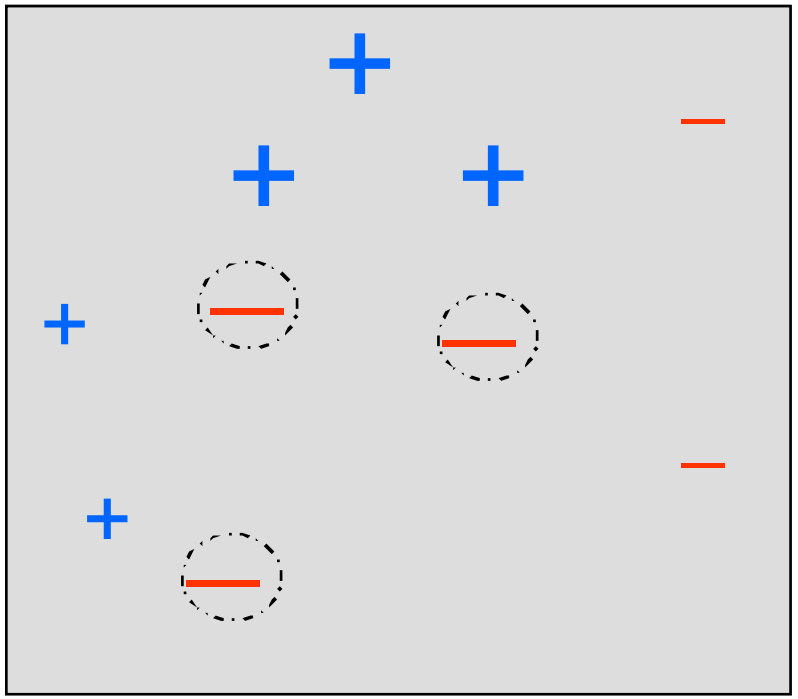
D_2

Итерация 2 из 3



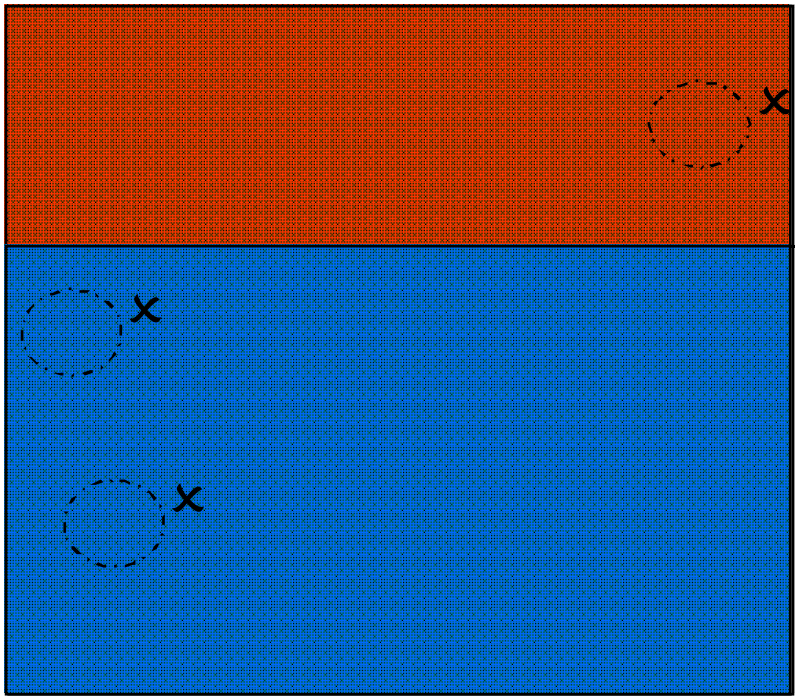
$\varepsilon_2 = 0.196$ h_2

$\alpha_2=0.704$



D_2

Итерация 3 из 3



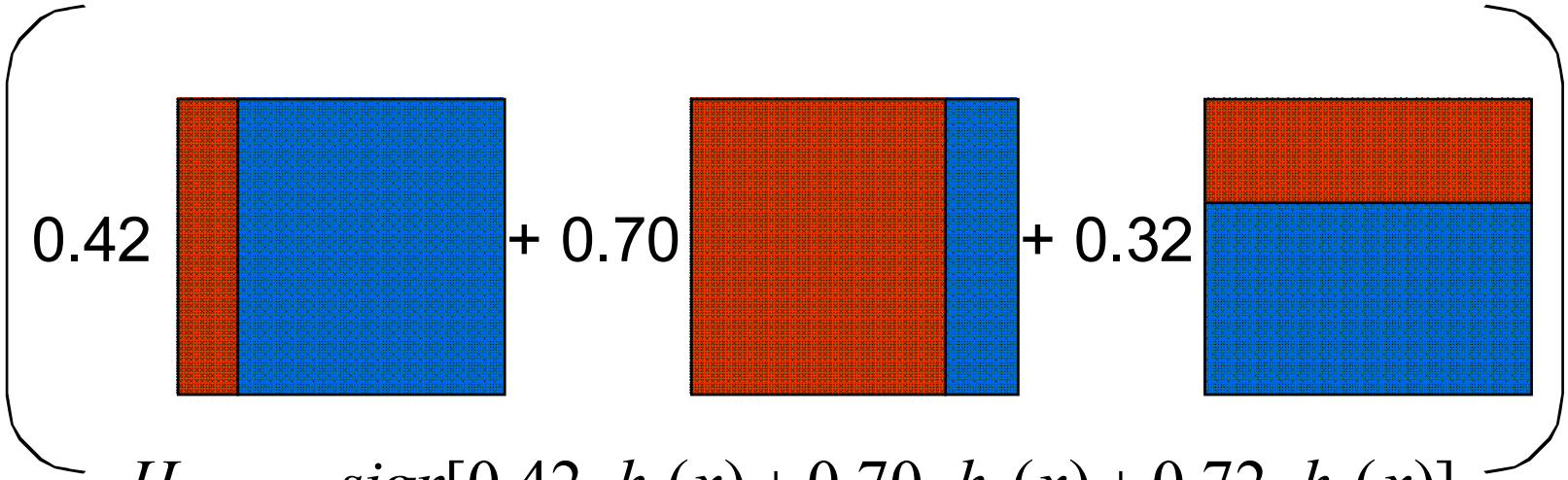
h_3

СТОП

$\varepsilon_3 = 0.344$

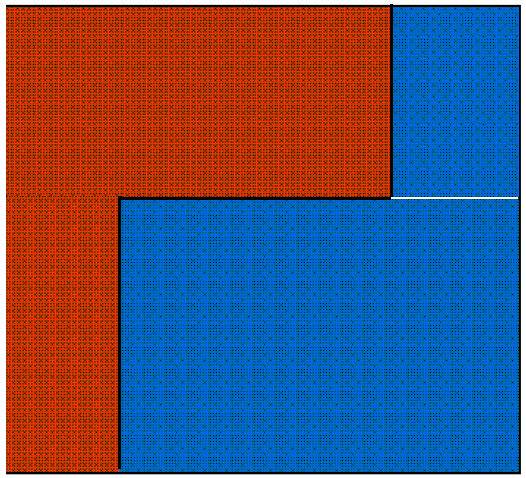
$\alpha_2 = 0.323$

Итоговый классификатор



$$H_{final} = sign[0.42 \cdot h_1(x) + 0.70 \cdot h_2(x) + 0.72 \cdot h_3(x)]$$

$$h_i(x) = \begin{cases} -1, x \to \Theta_0 \\ +1, x \to \Theta_1 \end{cases}$$



AdaBoost (Discreet)

Пусть есть набор *{h}* – слабых классификаторов

Пусть *T*: $(x_1, y_1), \ldots, (x_m, y_m)$ где $x_i \in X, y_i \in \Theta = \{-1, +1\}$

Инициализируем $D_1(i) = 1/m$

Для $k = \overline{1, K}$

1. Обучим h_k с минимальной ошибкой

$$\varepsilon_k = \Pr_{i \sim D_k}[h_k(x_i) \neq y_i]$$

Ошибка h_t рассчитывается с учётом D_t

2. Рассчитаем вес гипотезы

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$$

Вес Адаптируется. Чем больше ε_k тем меньше α_k

3. Для всех i = 1 to m

$$D_{k+1}(i) = \frac{D_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(x_i) = y_i \\ e^{\alpha_k} & \text{if } h_k(x_i) \neq y_i \end{cases}$$

Увеличиваем вес примера, если на нём алгоритм ошибается

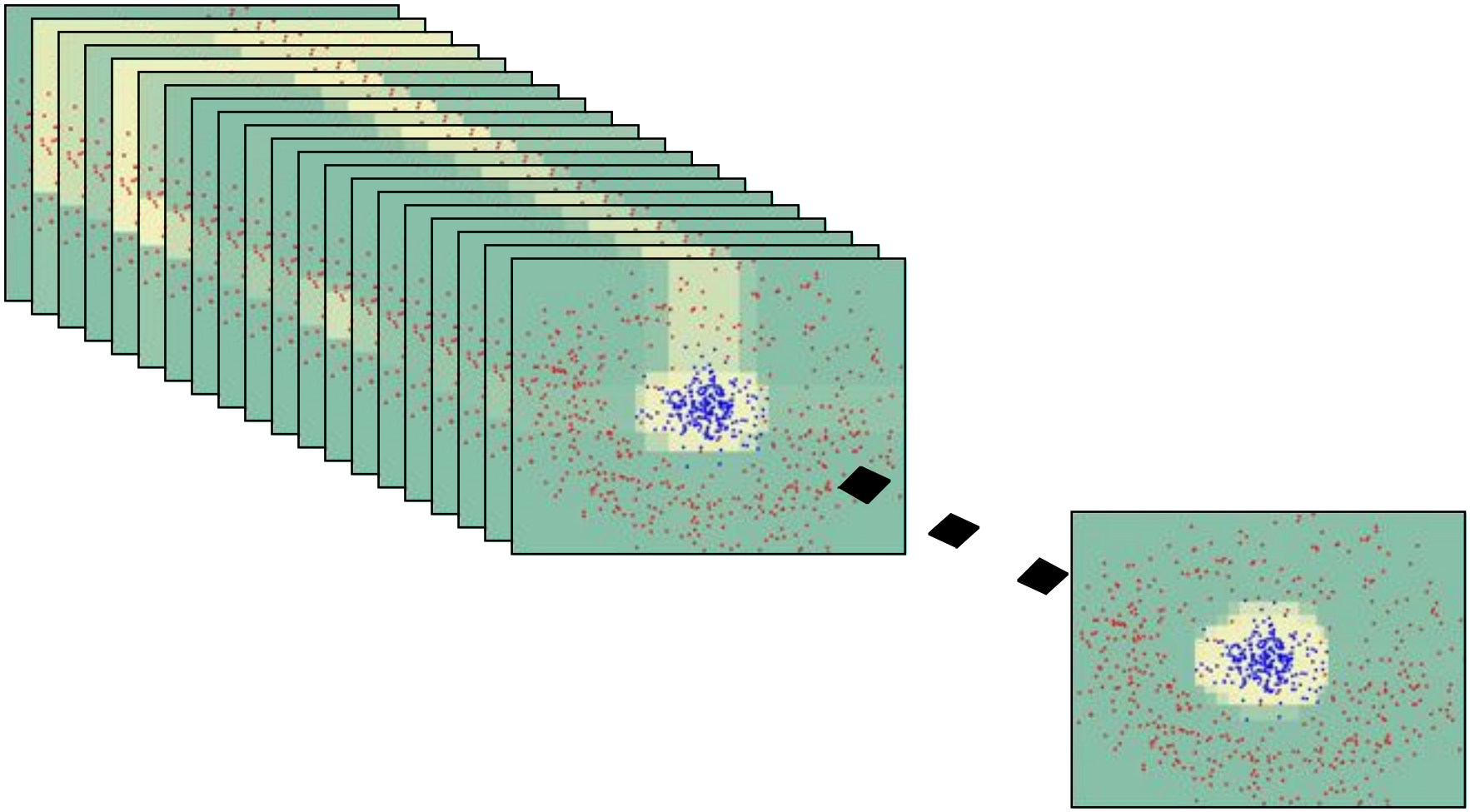
Z_t нормализующий коэф.

Результат:

$$H(x) = \operatorname{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(x)\right)$$

Линейная комбинация

AdaBoost (пороги)



Эмпирический риск

- Как показали Freund и Schapire

$$\gamma_k = \frac{1}{2} - \varepsilon_k$$

$$R_{emp} \le \exp\left(-2\sum_k \gamma_k^2 \right)$$

- Эмпирический риск падает по экспоненте – высокая скорость обучения

Бустинг

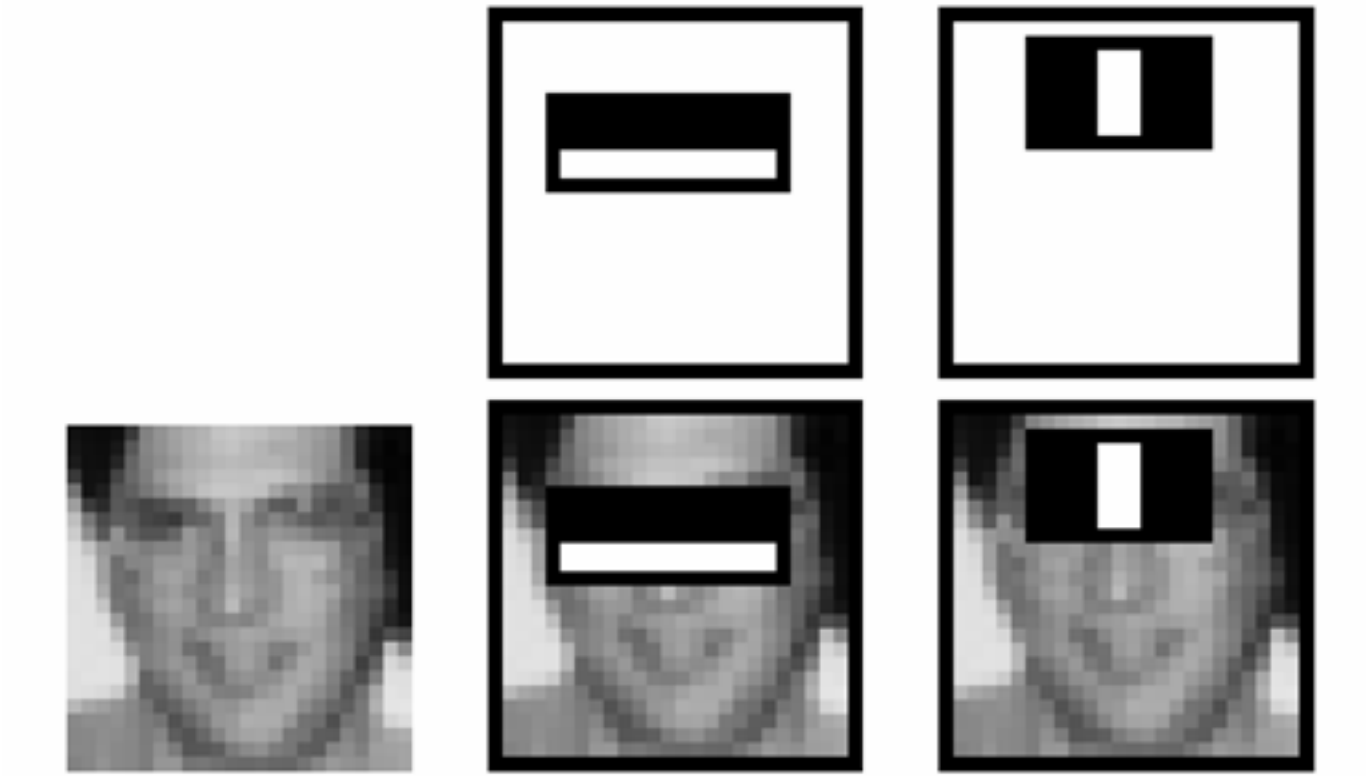
- Плюсы
  - \+ Универсальность
  - \+ Высокая скорость сходимости
  - \+ Высокая обобщающая способность
  - \+ Возможность очень эффективной программной реализации и распараллеливания
  - \+ Простота метода и отсутствия параметров
- Минусы
  - – Трудность определения нужного числа итераций обучения (зачастую, ошибка на контроле продолжает падать и после нуля эмпирического риска)

Бустинг для поиска лиц

- Определяем слабые классификаторы на основе прямоугольных признаков
- Для каждого этапа бустинга:
  - Вычисляем каждый прямоугольный признак на каждом примере
  - Выбираем наилучший порог для каждого признака (обучаем слабый классификатор)
  - Выбираем наилучший слабый классификатор и добавляем его в линейную комбинацию
  - Перевзвешиваем выборку
- Вычислительная сложность обучения: $O(MNK)$
  - M этапов, N примеров, K признаков

Бустинг для поиска лиц

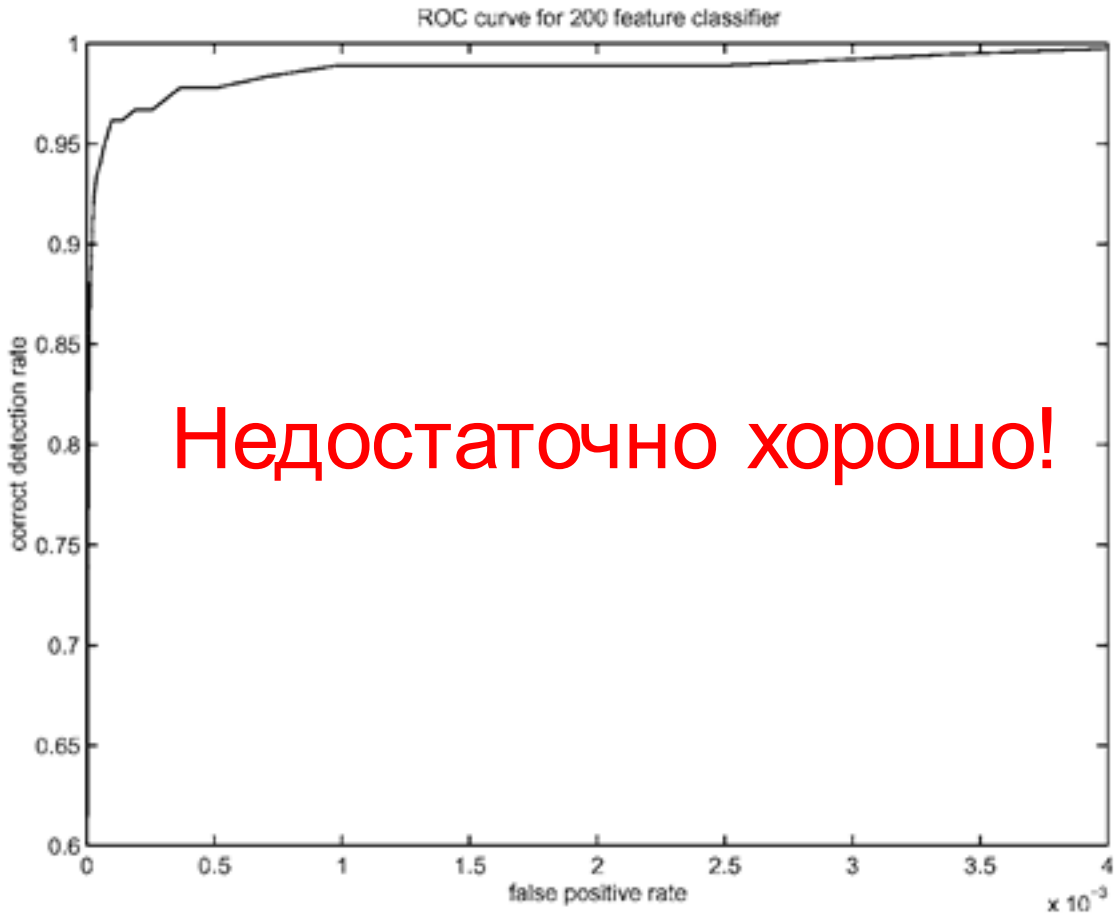
- Первые два классификатора, выбранные бустингом:



Эта комбинация признаков дает 100% detection rate и 50% false positive rate

Бустинг для поиска лиц

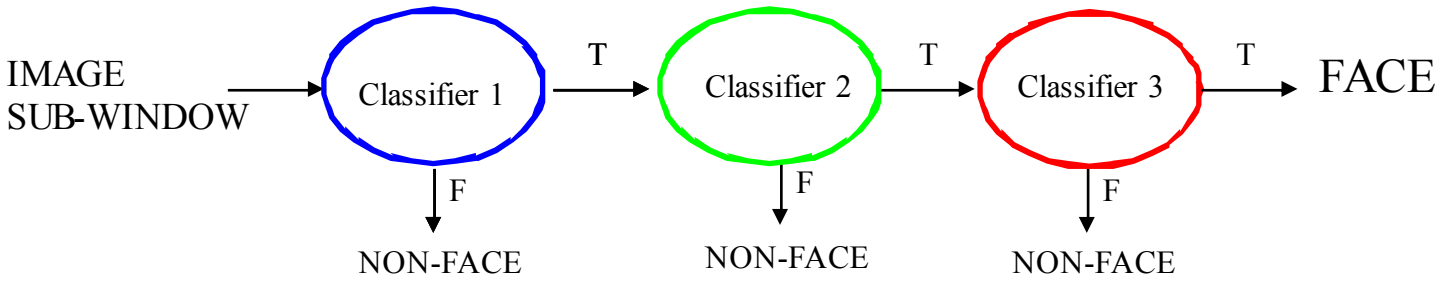
- Классификатор из 200 признаков дает 95% точность и долю ложноположительных срабатываний 1 из 14084



ROC-кривая

Каскад (Attentional cascade)

- Начинаем с простых классификаторов, которые отбрасывают часть отрицательных окон, при этом принимая почти все положительные окна
- Положительный отклик первого классификатора запускает вычисление второго, более сложного, классификатора, и т.д.
- Отрицательный отклик на любом этапе приводит к немедленной отбраковке окна



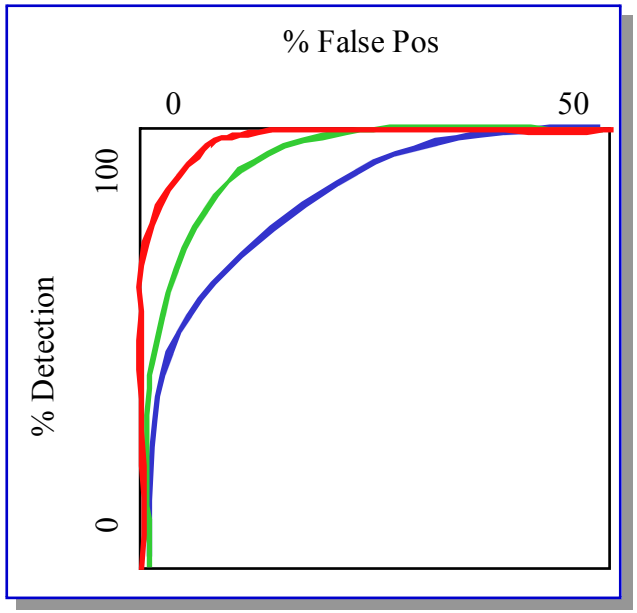
Каскад

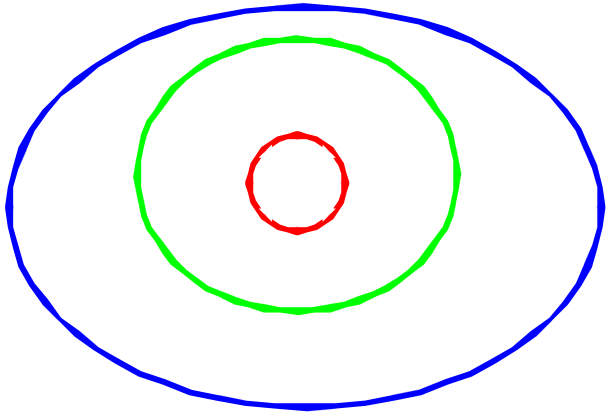
- Медленные классификаторы применяются только к некоторым окнам ⇒ существенное ускорение
- Управляем сложностью/скоростью классификатора:
  - Количество опорных векторов [Romdhani et al, 2001]
  - Количество признаков [Viola & Jones, 2001]
  - Видом ядра SVM [Vedaldi et al, 2009]

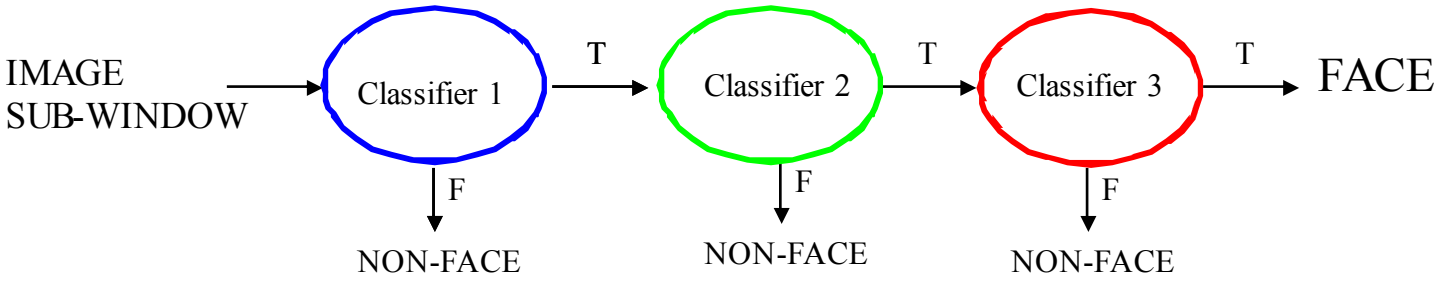
Каскад

Цепочка классификаторов с каждым уровнем становится более сложной, ошибка второго рода постоянно снижается

ROC-кривая

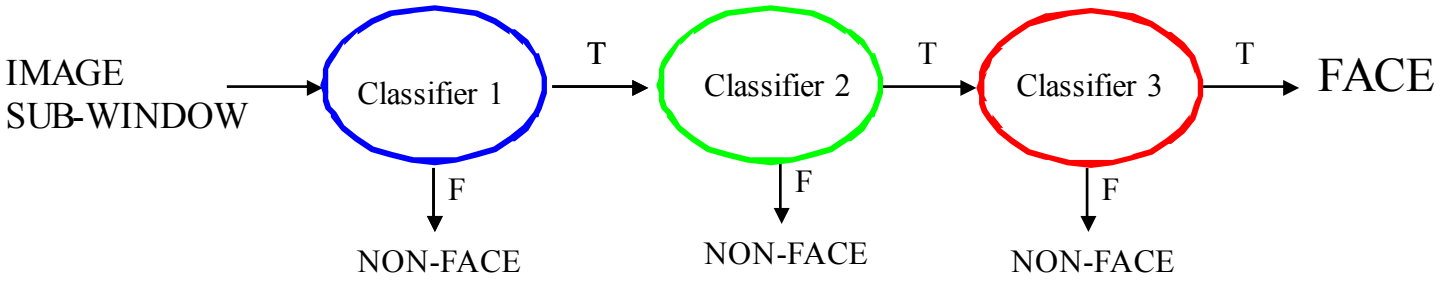






Каскад

- Уровень обнаружения и доля ложноположительных срабатываний каскада можно оценить как произведение соответствующих уровней ошибок каждого этапа
- Уровень обнаружения 0.9 и доля ложноположительных срабатываний порядка 10^{-6} достигается с помощью каскада из 10 этапов, если на каждом этапе уровень обнаружения примерно равен 0.99 ($0.99^{10} \approx 0.9$) и доля ложноположительных примерно 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$)



Обучение каскада

- Задаем требуемые значения уровня обнаружения и доли ложноположительных срабатываний для каждого этапа
- Добавляем признаки до тех пор, пока параметры текущего этапа не достигнут заданного уровня
  - Приходится понижать порог AdaBoost для максимизации обнаружения (в противоположенность минимизации общей ошибки классификации)
  - Тестирование на отдельном наборе (*validation set)*
- Если общий уровень ложноположительных срабатываний недостаточно низок, добавляем очередной этап
- Ложные обнаружения на текущем этапе используются как отрицательные примеры на следующем этапе

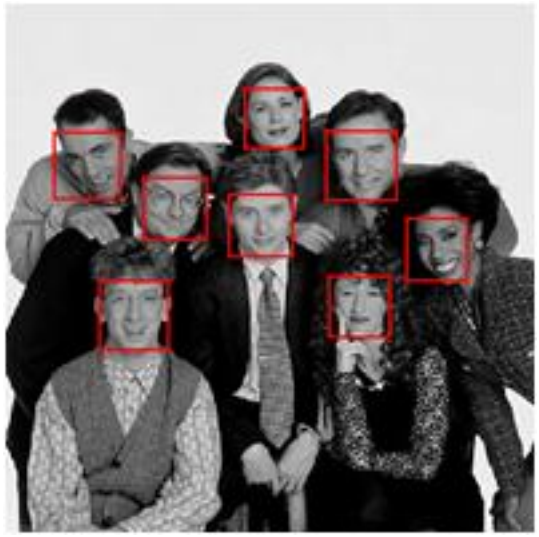
Тренировочная выборка

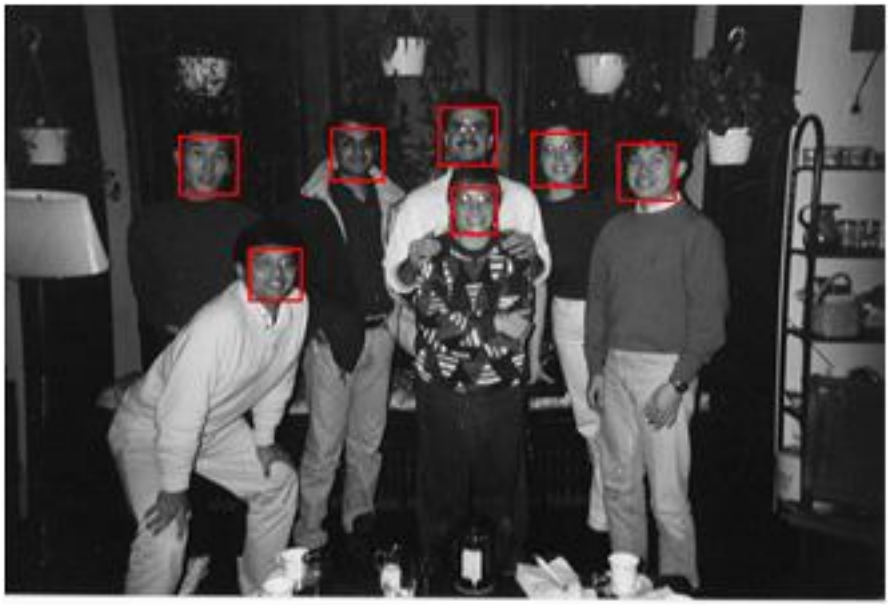
- 5000 лиц
  - Все фронтальные, уменьшенные до 24x24 пикселов
  - Все нормированы
- 300M отрицательных примеров
  - 9500 изображений без лиц
- Большая изменчивость
  - Разные люди
  - Освещение
  - Поза лица

Производительность системы

- Обучение: “недели” на 466 MHz Sun рабочей станции
- 38 этапов, всего 6061 признаков
- В среднем 10 признаков оцениваются для каждого окна на тестовой выборке
- “На 700 Mhz Pentium III, детектор лиц обрабатывает одно изображение 384x288 пикселов за 0.067 секунды”
  - 15 Hz
  - В 15 раз быстрее сравнимого по точности предшествующего метода (Rowley et al., 1998)

Пример работы





Резюме: детектор Viola-Jones

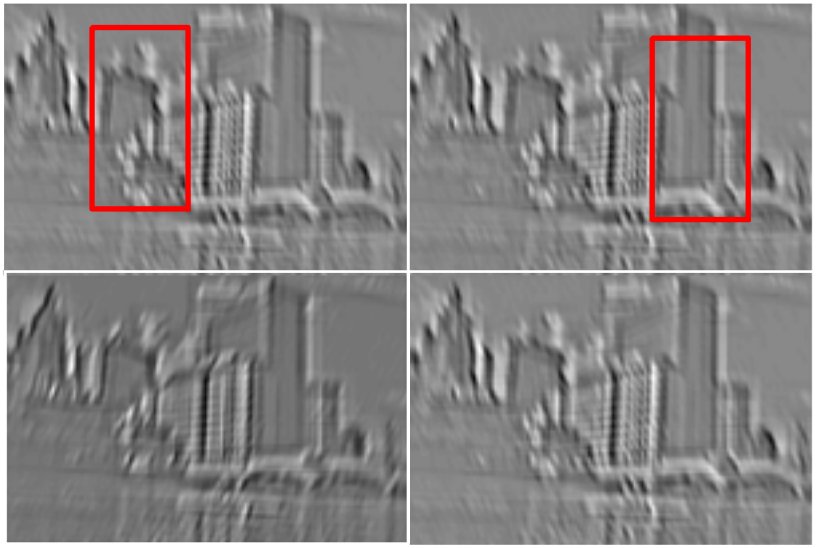
- Прямоугольные признаки как слабые классификаторы
- Интегральные изображения для быстрого вычисления признаков
- Бустинг для выбора признаков
- Каскад классификаторов для быстрого выбраковки отрицательных окон

Недостатки Viola-Jones

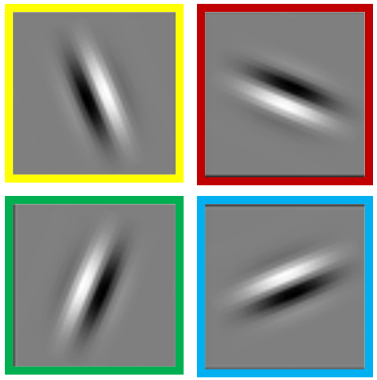
- Малое количество используемых каналов изображения и признаков
  - Серые изображения, фильтры Хоара
- Недостаточная точность
  - 10^{-6} – слишком много ложных срабатываний
- Множественные отклики у правильных обнаружений
- «Дискретность» - возможность завершения только в конце этапа
- Очень долгое время обучения

Интегральные канальные признаки

- Цвет и градиенты очень показательны для многих классов
- Построим много (10 каналов)
  - Яркость + цвета (LUV, HSV) – 3
  - Норма градиента + отклики по направлениям – 1+6 = 7
- Признаки – сумма значений в прямоугольной области в заданном канале
  - Быстро считаем через интегральные изображения



P. Dollár, Z. Tu, P. Perona and S. Belongie Integral Channel Features BMVC 2009,



Учёт масштабов

N models, 1 image scale

(a) Naive approach

1 model, N image scales

(b) Traditional approach

1 model, N/K image scales

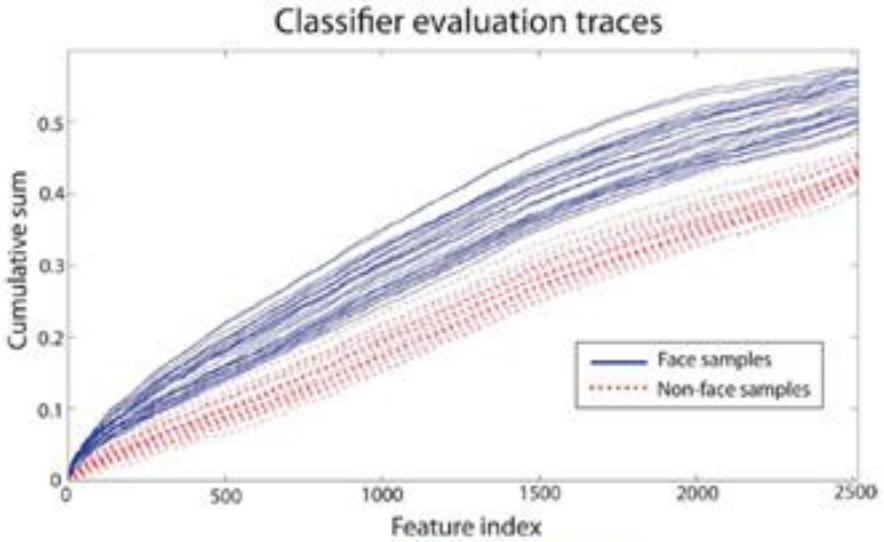
(c) FPDW approach

N/K models, 1 image scale

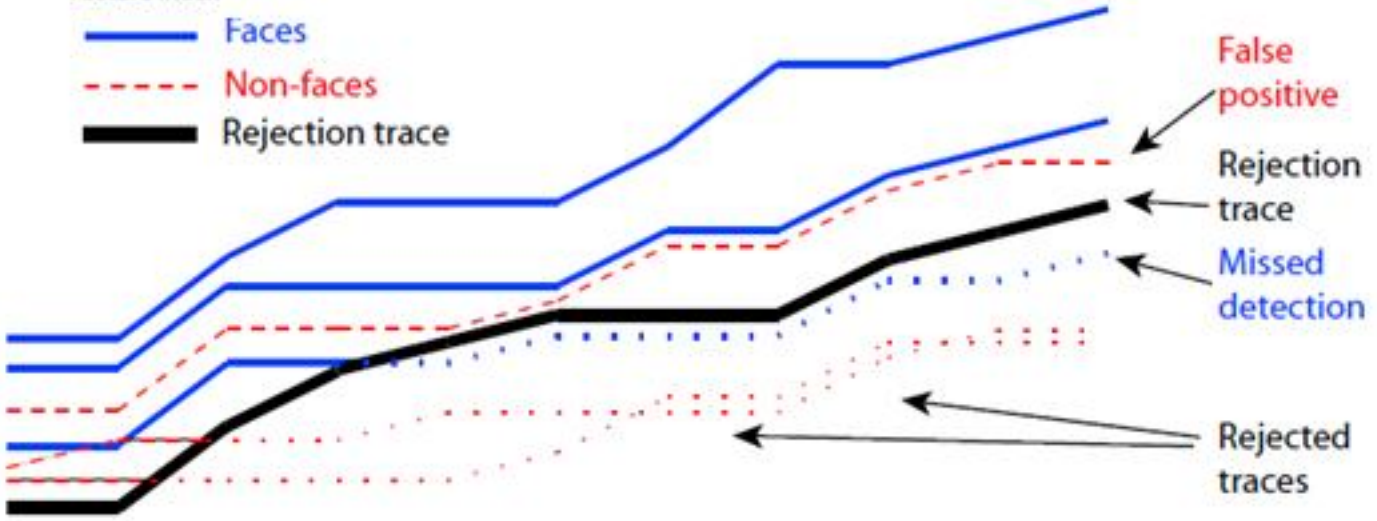
(d) Our approach

P. Dollár, S. Belongie and P. Perona The Fastest Pedestrian Detector in the West BMVC 2010, R. Benenson,M. Mathias, R. Timofte, L. Van Gool Pedestrian detection at 100 frames per second, CVPR 2013

Мягкий каскад

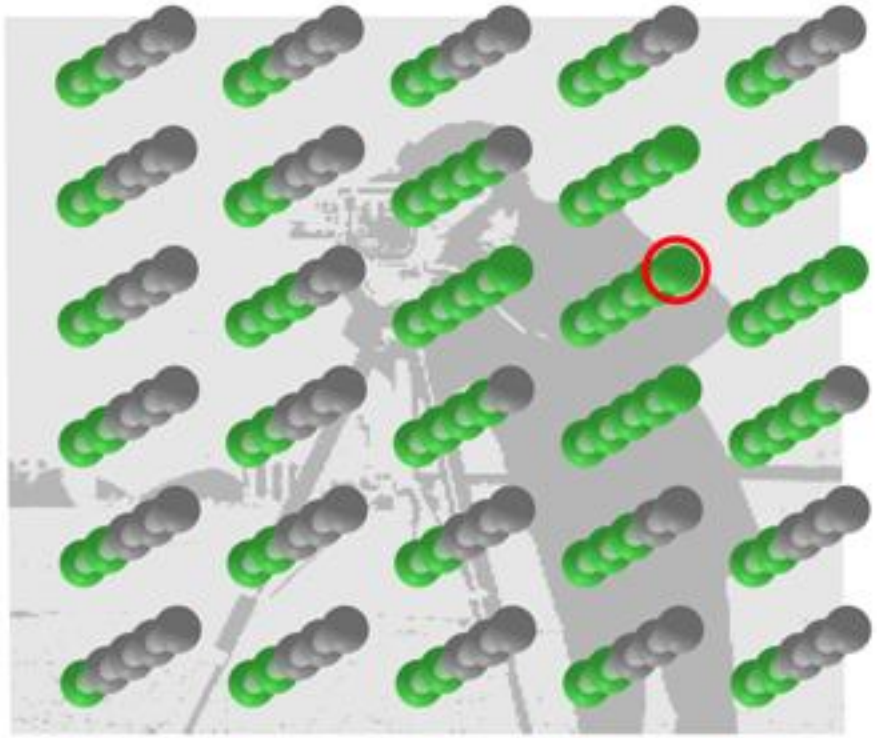


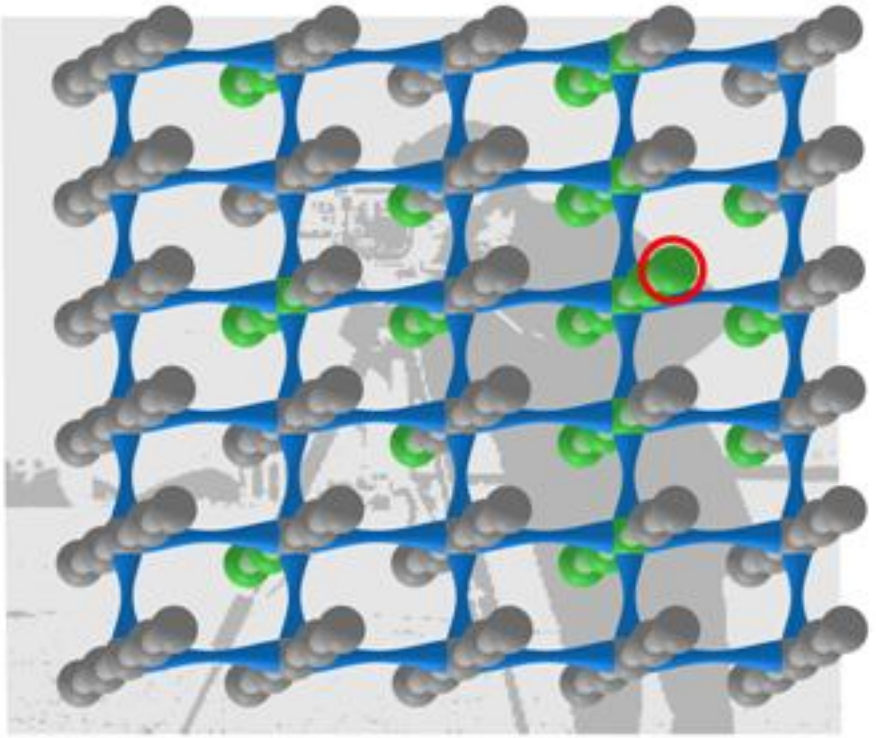
Теперь можем остановиться на любом слабом классификаторв



Bourdev, L., Brandt,J. Robust object detection via soft cascade. CVPR 2005

«Crosstalk cascades»





Идея – нужно «подавлять» каскады в тех областях, где скорее всего не будет локального максимума

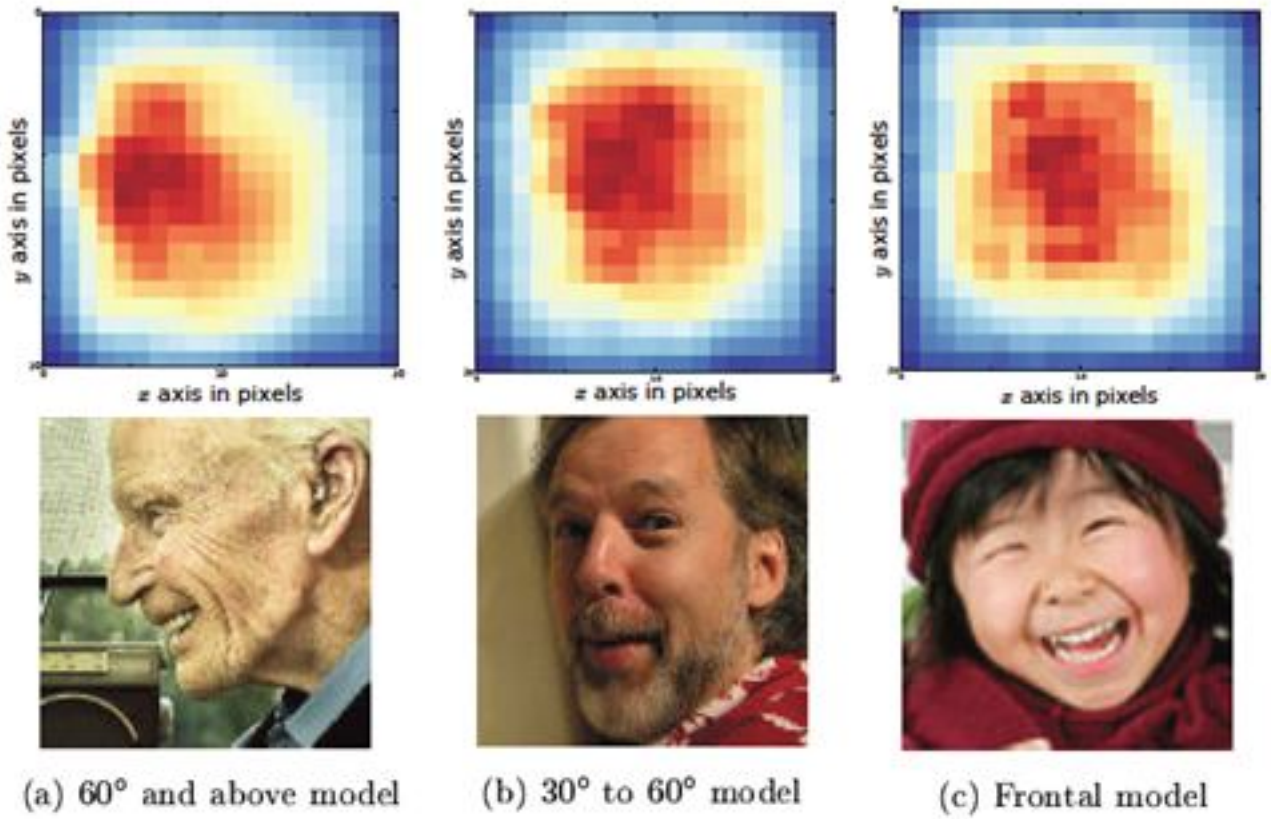
P. Dollár, R. Appel and W. Kienzle Crosstalk Cascades for Frame-Rate Pedestrian Detection. ECCV 2012

Работа с ракурсами

- Попытка делать детектор одновременно для всех ракурсов делает его менее «жёстким», ухудшает специфичность
- Обучаем набор детекторов для разных ракурсов («компоненты»)
  - Проблема сбора данных такого времени

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

Важность «компонент»

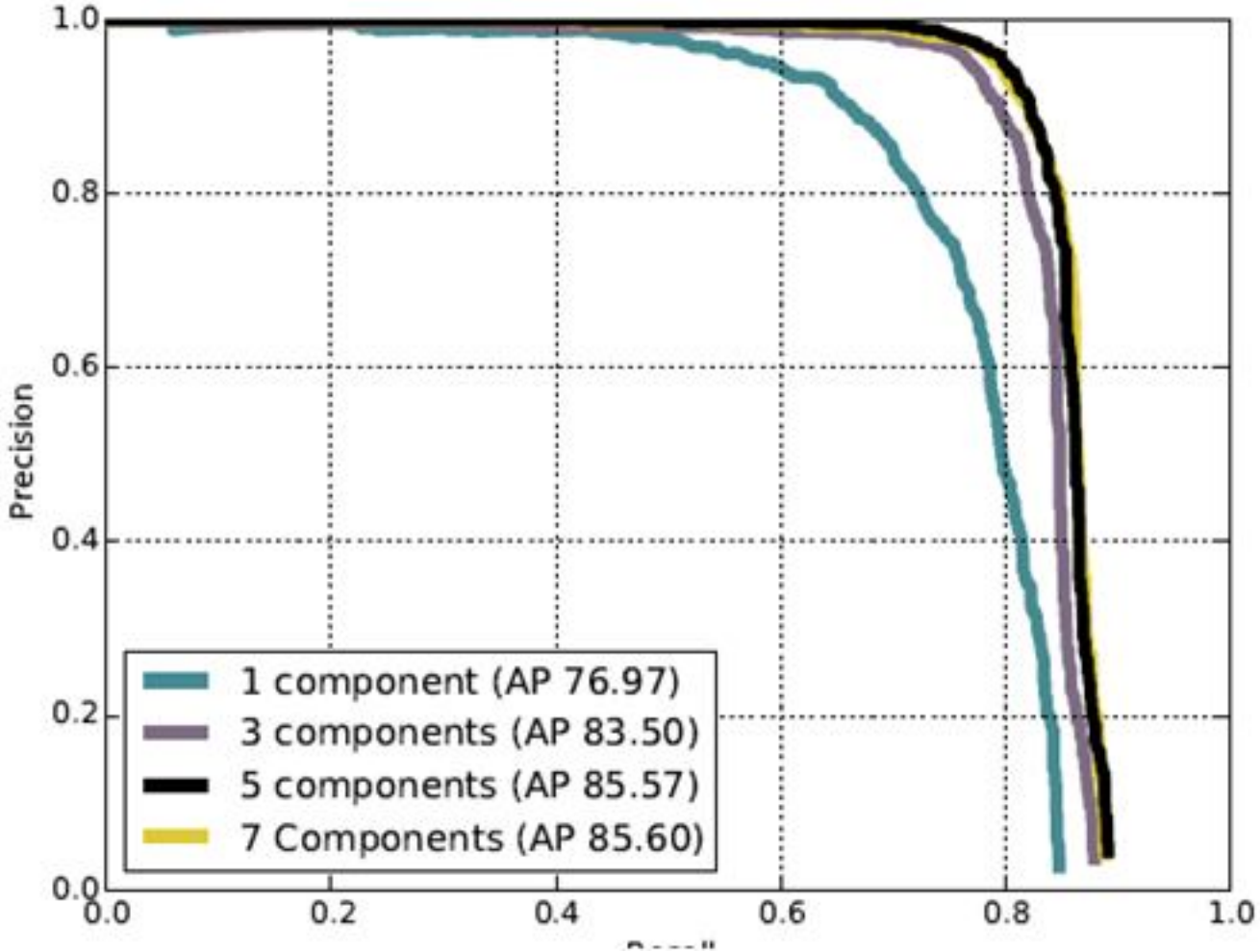


(a) 60° and above model (b) 30° to 60° model (c) Frontal model

Figure 6: Components of the learned model (and example training sample). Red indicates areas with higher influence for the score decision.

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

Влияние «компонент»



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

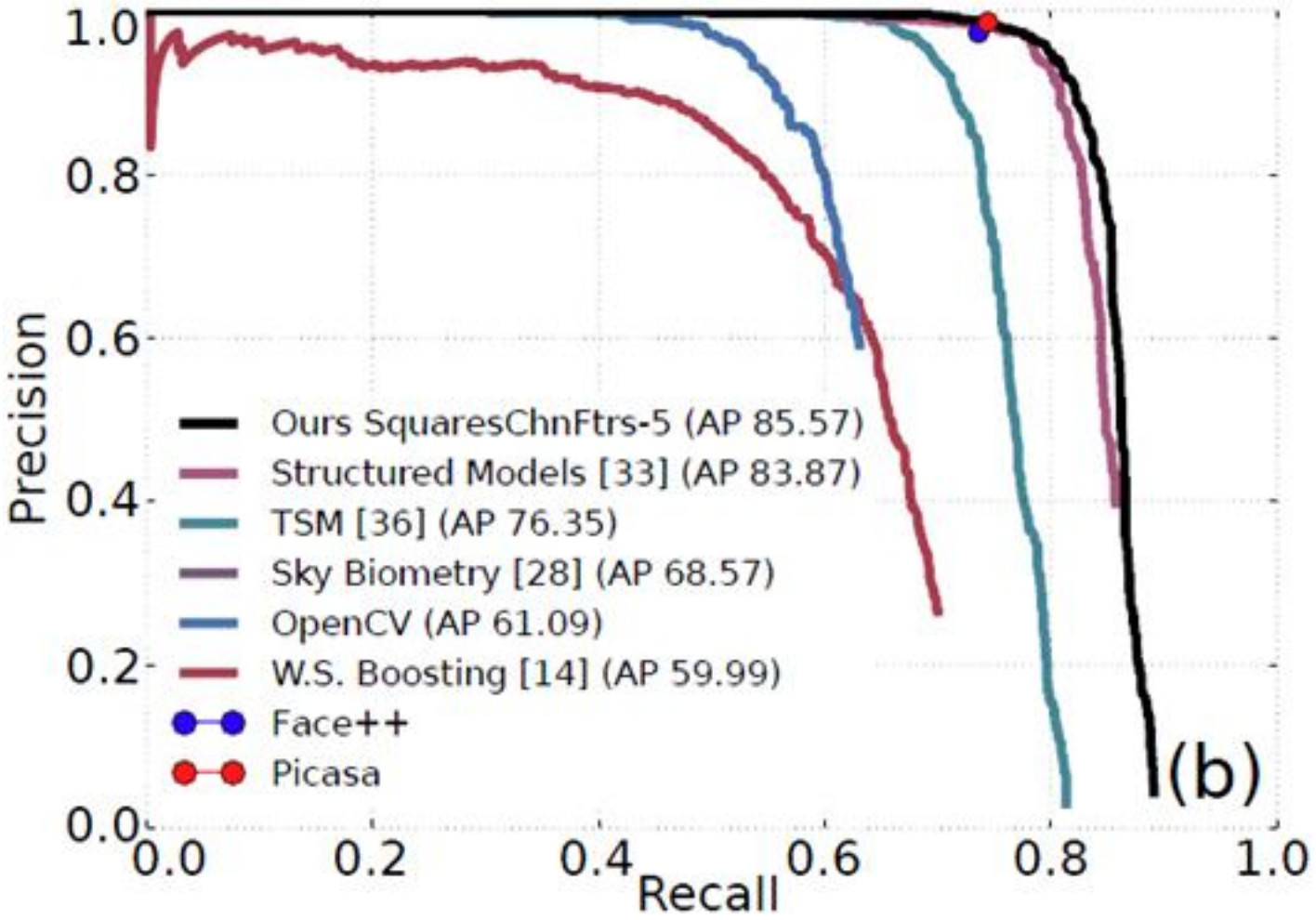
Вопросы разметки

(a) Original annotations

(b) Updated annotations

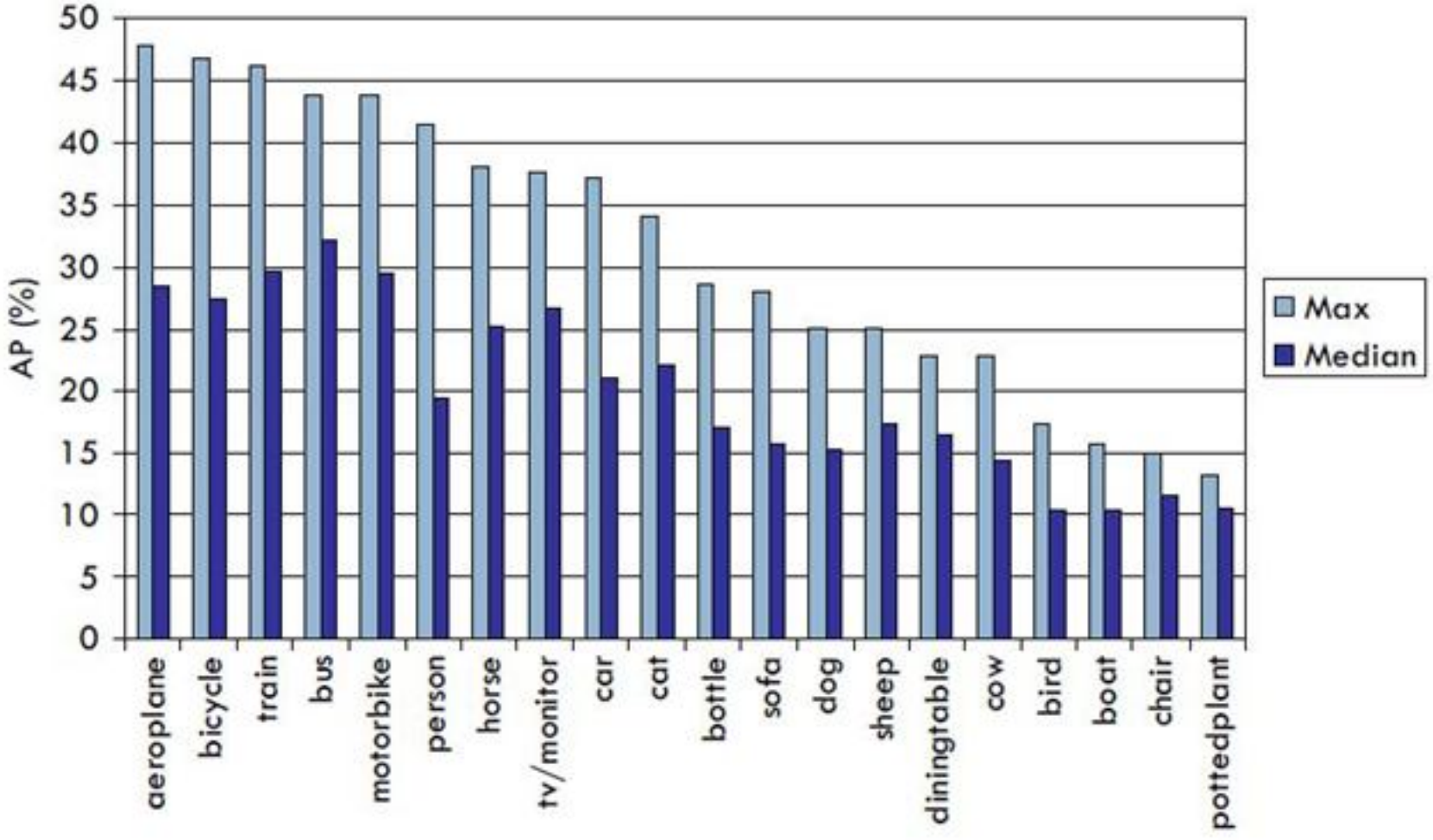
- Разные люди и разные базы размечены немного по разному
- Есть лица, которые лучше игнорировать, но за ошибку не считать
- Проблема «маленьких» лиц

Выделение лиц (2014)

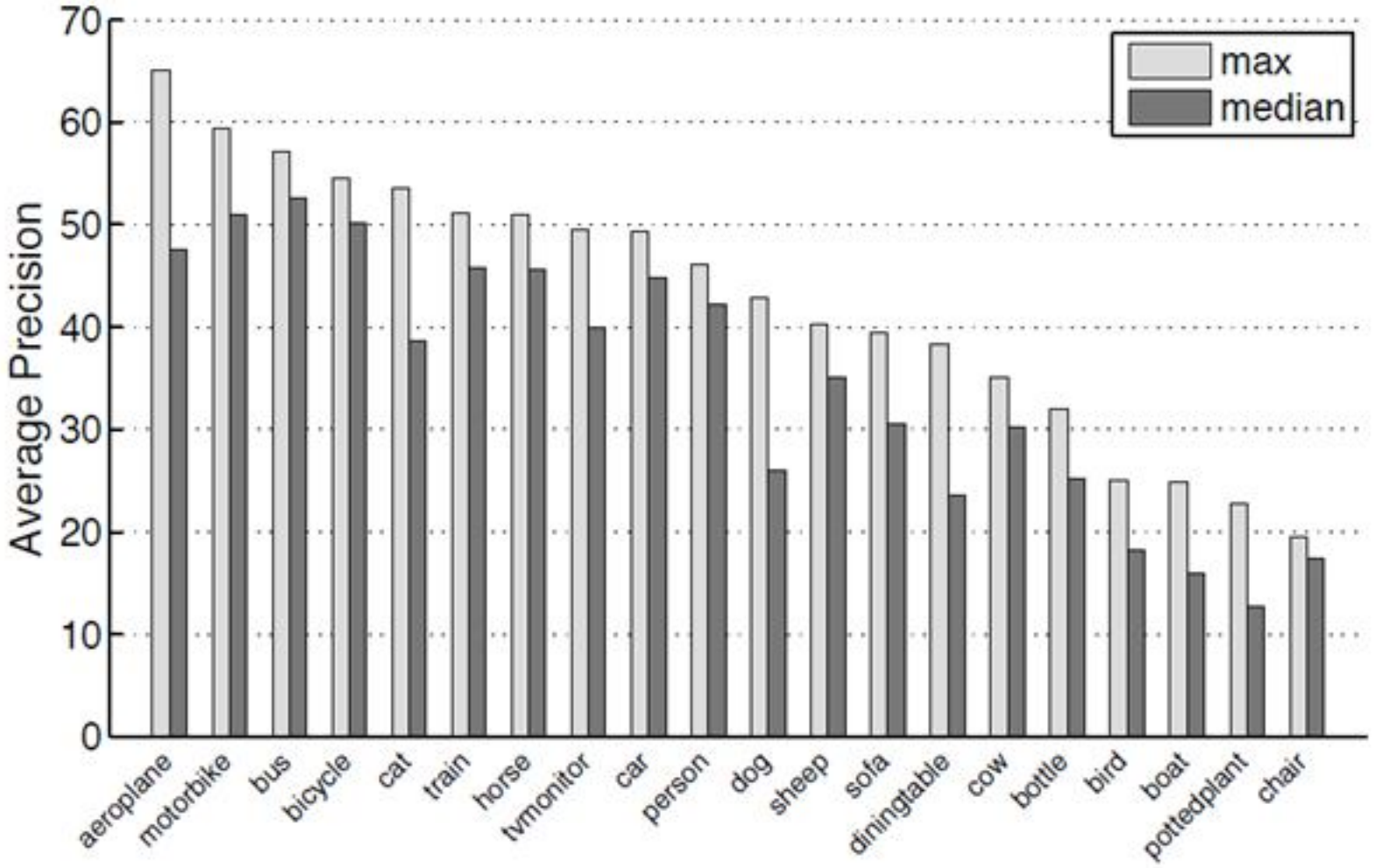


M. Mathias. et. al. Face detection without bells and whistles. ECCV2014

Точность PASCAL VOC (2009 год)



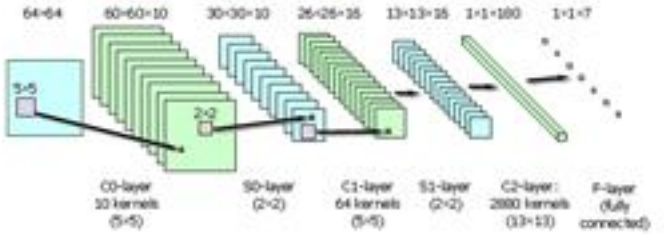
Точность (2012 год)



За пределами лекции

Многослойные нейросети

- На следующей лекции



Модели из набора частей и учет их взаимного расположения

- В курсе «Доп. главы компьютерного зрения»

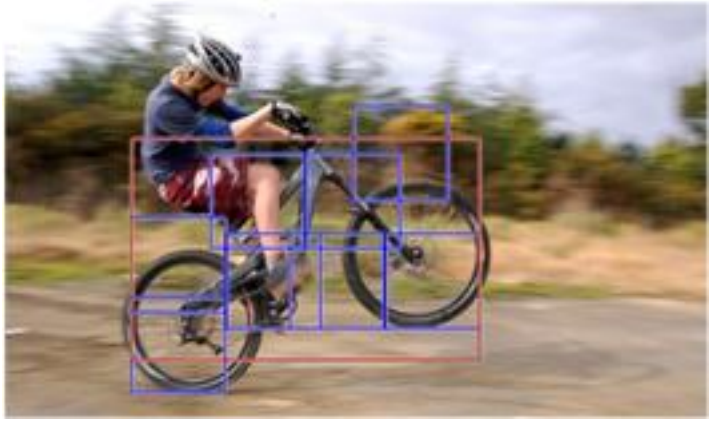
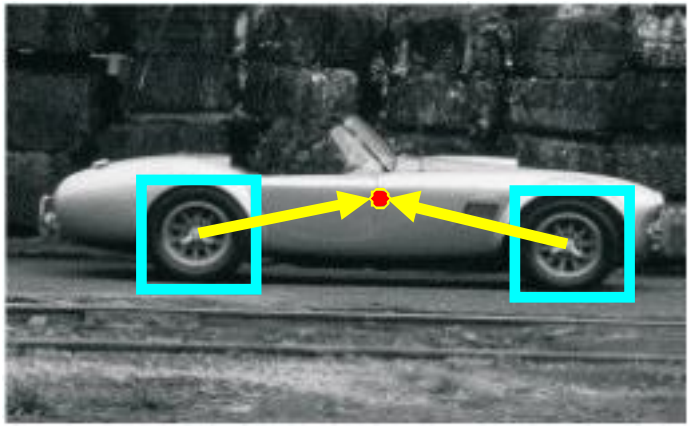


Схема голосования как альтернатива скользящему окну

- В курсе «Доп. главы компьютерного зрения»



Резюме лекции

- Скользящее окно – основной способ сведения задачи выделения объектов к задаче категоризации изображений
- Каскад классификаторов – основной способ повышения скорости и точности работы
- Схема VJ развивается и сейчас является основным методом для real-time выделения объектов
- Нужно уделить большое внимание построению хорошей выборки для обучения (jittering, bootstrapping)