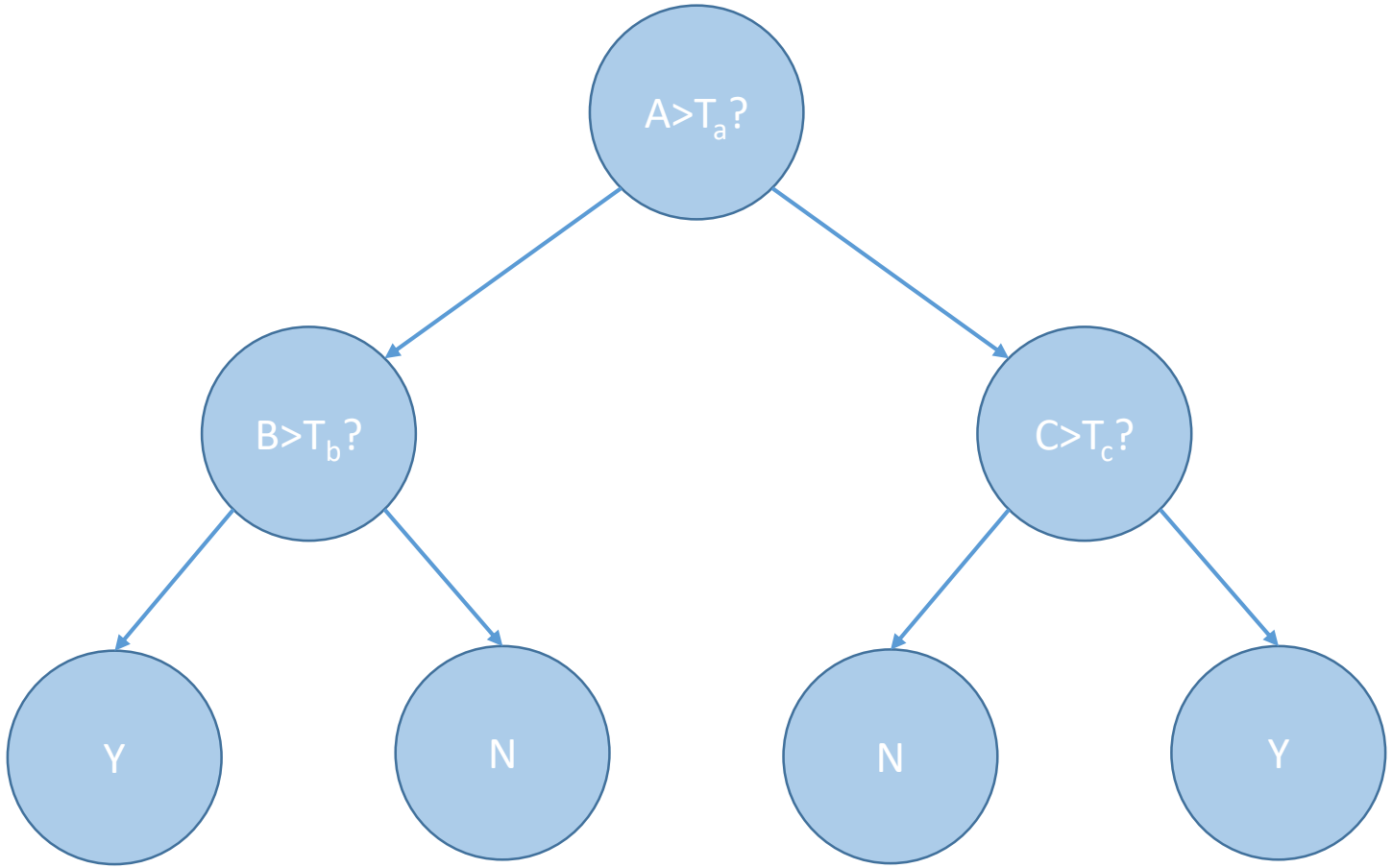
Курс “Анализ изображений”

Лекция#9.
Градиентный бустинг. Обнаружение объектов. Гистограммы направленных градиентов. Алгоритм Виолы-Джонса.

ФИВТ МФТИ

2017

Деревья принятия решений



Градиентный бустинг

- Цель: собрать из нескольких одинаковых классификаторов один, который бы работал лучше всех по-отдельности
- Как: давайте жадно выбирать классификаторы, так, чтобы каждый новый добавленный классификатор уменьшал бы финальную ошибку
- Итоговый классификатор имеет форму линейной комбинации:

$$F(x) = \sum_{i=1}^{T} \alpha_i h_i(x)$$

Градиентный бустинг

- Обучающий набор: $\{x_i, y_i\}_{i=1}^n$; функция потерь: $L\big(y, F(x)\big)$
- $F_0(x) = C$
- $\text{FOR } t = 1..T$
  - Вычисляем псевдо-невязки: $r_t = \frac{\partial L}{\partial F_{t-1}}$
  - Обучаем h_t на $\{x_i, r_{it}\}_{i=1}^n$
  - $F_t(x) = F_{t-1}(x) + \gamma_t h_t(x)$, где $\gamma_t = \underset{\gamma}{\text{argmin}}\, L\big(y, F_t(x)\big)$
- Реализации: sklearn, XGBoost, LightGBM, CatBoost

AdaBoost

- Обучающий набор: $\{x_i, y_i\}_{i=1}^{n}$; $L\big(y, F(x)\big) = \sum_i e^{-y_i F(x_i)}$
- Равномерно инициализируем веса w_i^0
- $\mathrm{FOR}\ t = 1..T$
  - Нормализуем веса: $\sum_i w_i^t = 1$
  - Обучаем слабые классификаторы: $\varepsilon_t = \min_h \sum_i w_i^t |h(x_i) - y_i|$
  - Обновляем веса: $w_i^{t+1} = w_i^t \beta_t^{1-e_i}$

    (где $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ и $e_i = I[h(x_i) \neq y_i]$)
  - Итоговый классификатор: $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$

    (где $\alpha_i = \log\frac{1}{\beta_t}$)

Задача детектирования

Задача детектирования:

Дано: набор графических образов

Задача: определить присутствие, положение и размер всех объектов

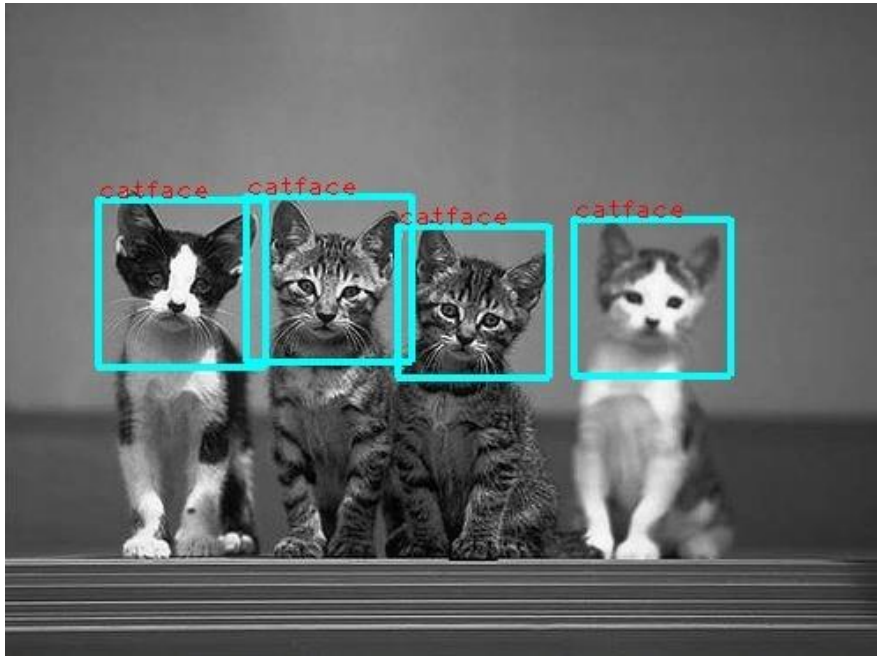
Метод скользящего окна сводит задачу детектирования к задаче классификации каждого окна изображения

Задача классификации:

X – множество объектов, Y – множество классов

Дано: отображение a^*: $X \to Y$, заданное на конечном подмножестве X

Задача: построить отображение a: $X \to Y$ на всем множестве X



История развития методов

2001: Оригинальный алгоритм Viola-Jones (VJ)

2003: Апробация VJ в задаче обнаружения пешеходов

2005: метод опорных векторов на основе гистограмм ориентированных градиентов (HOG+SVM), стенд INRIA (Dalal, Triggs)

2008: Детектор деформируемых частей (DPM) (Felzenszwalb)

2009: Признаки интегрированных каналов (ICF, Dollar)

2009: Публикация стенда Caltech (Dollar)

2010: FPDW: ускоренная версия ChnFtrs (Dollar)

2010: Улучшенная версия DPM, DPM на основе VJ (Felzenszwalb)

2012: Улучшенная методология оценки (Dollar)

2012: VeryFast – ускоренный ChnFtrs (Benenson)

2013: Roerei – совокупность всевозможных улучшений на основе ICF (Benenson)

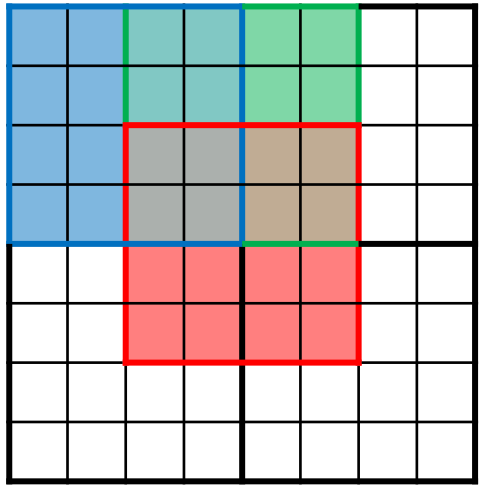
2014: Улучшенная версия ICF, признаки аггрегированных каналов – ACF (Dollar)

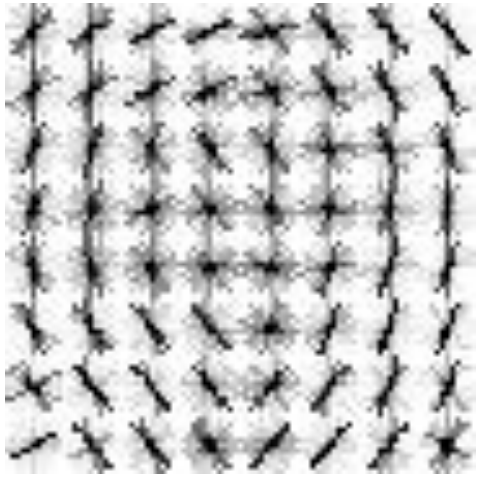
Гистограммы направленных градиентов (Dalal, Triggs, 2005)

Классический подход к обнаружению пешеходов на основе SVM

Устройство HOG:

1. Cell – участок изображения, пиксели которого составляющих гистограмму по дискретизированным направлениям градиента
2. Block – участок изображения, содержащий несколько ячеек и пиксели которого участвуют в нормализации гистограмм





Support Vector Machine

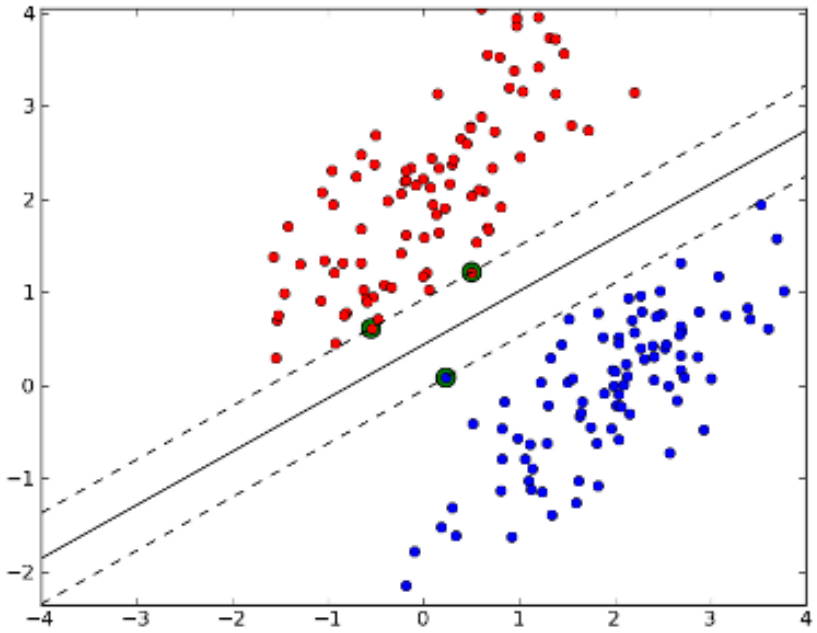
- Метод опорных векторов, классификатор максимального зазора

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

- Обучение:

$$\min_w \|w\|^2$$
$$y_i(w^T x_i + b) \geq 1$$

- Задача квадратичного программирования, единственный глобальный минимум
- Модель можно рассматривать как линейный фильтр изображения



Bootstrapping

Проблема: несбалансированность классов: объектов гораздо меньше чем фона

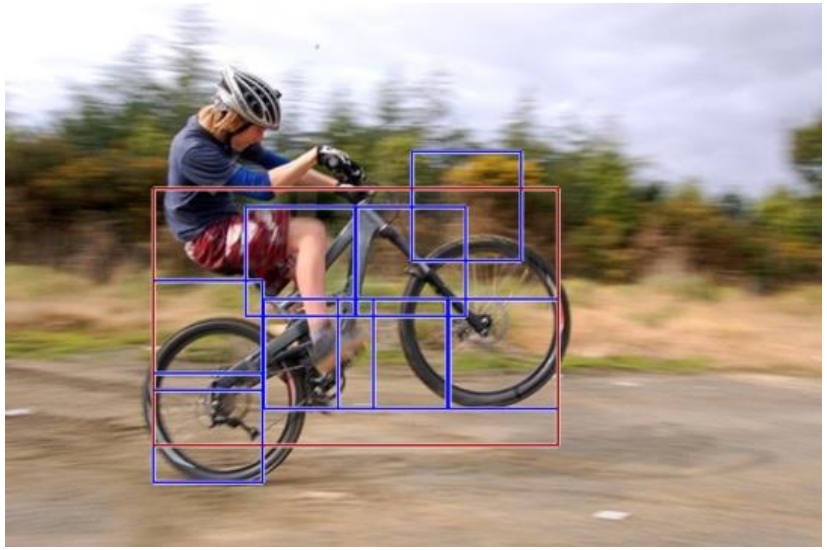
Решение – bootstrapping:

1. Берем положительную выборку, рандомно генерим отрицательную
2. Обучаем классификатор
3. Добавляем ложные срабатывания детектора к отрицательной выборке
4. Переходим обратно к п. 2

- Цель: формирование репрезентативной негативной выборки из “сложных” примеров
- Для адекватных результатов необходимо минимум 3 итерации bootstrapping’а

Детектор деформируемых частей (DPM, Felzenszwalb, 2008)

Первый из семейства детекторов, использующих не жесткий шаблон, а описывающих объект как набор гибко расположенных частей. Детектор хорошо зарекомендовал себя в задаче детектирования произвольных объектов. По сути – набор классификаторов HOG+SVM для самого объекта и его частей. Каждый классификатор обучается отдельно и итеративно, каждую итерацию выбирая наилучшее положение на изображении предыдущего классификатора.



Детектор деформируемых частей (DPM, Felzenszwalb, 2008)

Обучение частей объекта:

- количество частей фиксировано (6)
- размер – a: $6 * a = 0.8 * s$, где s – размер классификатора объекта
- начальное положение выбирается жадно (по максимуму суммы модуля градиента в области) и так чтобы пересечение с классификатором объекта было не менее 50%
- начальные веса инициализируются весами классификатора объекта
- Обучается на большем масштабе изображения

При детектировании оценка расположения производится исходя из суммы оценок обнаружения всех частей и штрафа за отклонение от исходного положения частей:

$$s = \sum_{i=0}^{n} F_i * \phi(H, p_i) + \sum_{i=1}^{n} a_i * (\tilde{x}_i, \tilde{y}_i) + b_i * \left(\tilde{x}_i^2, \tilde{y}_i^2\right)$$

где $(\tilde{x}_i, \tilde{y}_i) = ((x_i, y_i) - 2 * (x, y) + v_i)/s_i$ – положение *i*-й части относительно положения центра классификатора объекта, $-1 < \tilde{x}_i, \tilde{y}_i < 1$; v_iи s_i – центр и размер прямоугольника возможных положений части *i;* a_i и b_i – обучаемые коэффициенты

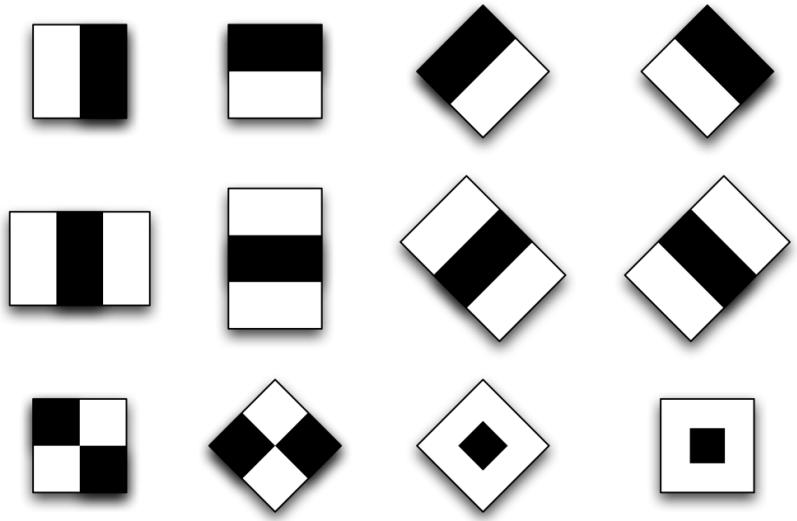
Алгоритм VJ (Viola, Jones, 2001)

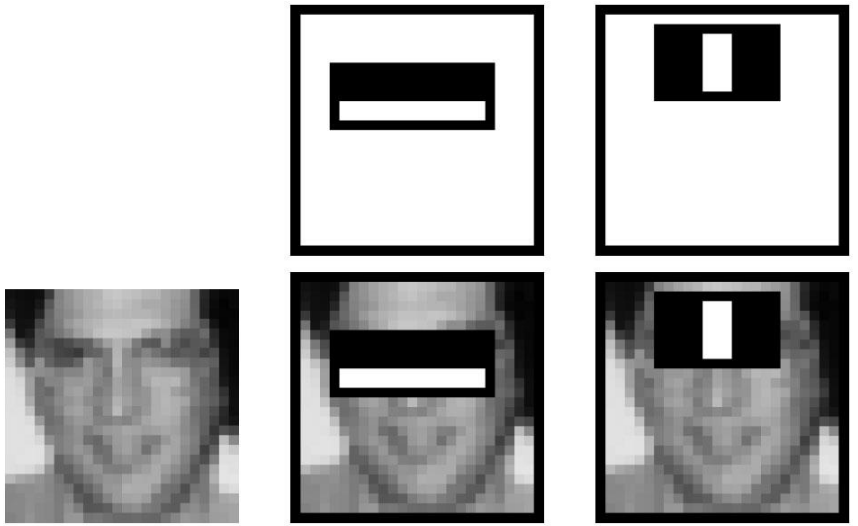
Классический алгоритм детектирования объектов с жесткой структурой, представленный в 2001 году.

Основные составляющие:

- Хаар-подобные признаки
- Интегральное изображение
- AdaBoost для выбора признаков
- Каскад классификаторов для быстрой фильтрации негативных примеров

Хаар-подобные признаки





$$Value = \sum_{(x,y)\in white\ rect} I(x,y) - \sum_{(x,y)\in black\ rect} I(x,y)$$

В окне поиска 24x24 пикселя число возможных прямоугольных признаков достигает ~160К

Интегральное изображение

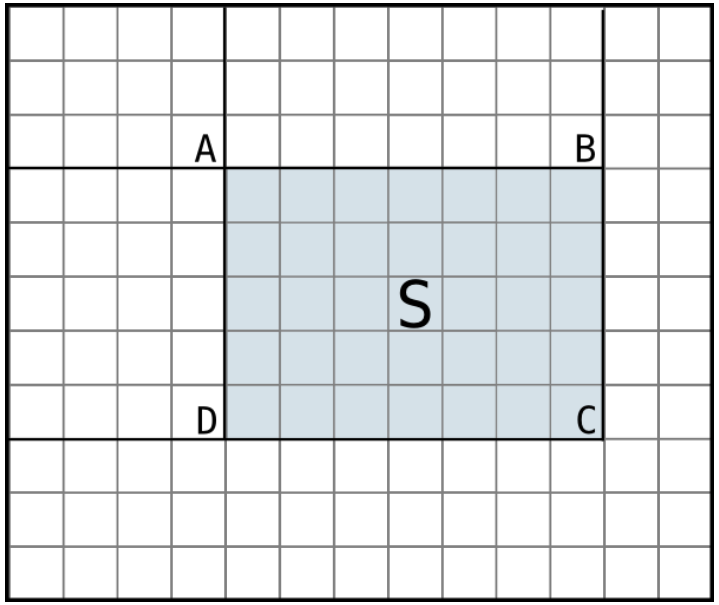
- Значение пикселя – сумма значений всех пикселей выше и левее
- Считается за 1 проход
- MATLAB:

$$II = cumsum\big(cumsum(I, 2)\big)$$

- Сумма пикселей в прямоугольнике:

$$S = A + C - B - D$$

- Для подсчета суммы в 1-м прямоугольнике необходимо 4 взятия элементов массива, для 2-х смежных: 6 взятий



Attentional Cascade

Предпосылки:

- классификатор, обученный AdaBoost должен быть достаточно большим, чтобы адекватно работать. Но это сильно увеличивает время обнаружения.
- Всего нескольких признаков достаточно для фильтрации огромного числа окон фона.

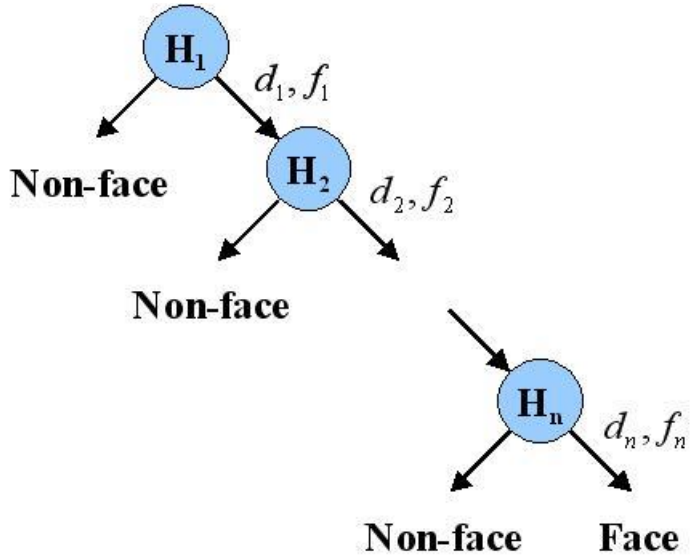
Обучение:

Повторяем:

- Выбираем негативную обучающую выборку как ошибки предыдущей части каскада
- Обучаем новый уровень каскада

Использование:

Если на одном из уровней классификатор сказал “нет” – далее не рассматриваем это окно, иначе запускаем следующий классификатор



Attentional Cascade

- FPR и TPR можно оценить как произведения соответствующих значений всех уровней каскада
- Первые уровни состоят из 2-3 признаков, потом планомерно увеличиваются
- Для достижения нужных значений ошибок у каждого уровня уменьшаем порог классификатора AdaBoost для максимизации обнаружения

Резюме: Виола-Джонс

- Хаар-подобные признаки
- Быстрое вычисление, используя интегральное изображение
- AdaBoost для выбора признаков, настроенный на фильтрацию отрицательных примеров
- Каскад классификаторов для быстрой фильтрации фона

Признаки интегрированных каналов (ChnFtrs, Dollar, 2009)

Метод, вобравший в себя лучшие улучшения алгоритма VJ

- Совместно используются несколько каналов (или статистик) изображения
- Используются только признаки 1-го порядка (только суммы по прямоугольникам)
- Полноценные деревья принятия решений вместо ступенек
- Использование только небольшого подмножества от всех признаков (значительно ускоряет процесс обучения, незначительно влияет на итоговое качество)
- Каскад -> Soft Cascade

Каналы

- каналы изображения в цветовом пространстве LUV (3 канала)
- модуль градиента изображения (1 канал)
- модуль градиента изображения, дискретизированный по направлению (6 каналов)

Soft Cascade

- Обучается большой классификатор AdaBoost (~2-4К деревьев):

$$F(x) = \sum_{t=0}^{T} \alpha_t h_t(x)$$

- Каждой частной сумме назначается свой порог по обучающей выборке, так чтобы все (или почти все) результат на положительных примерах был больше
- Обучается bootstrapping’ом; вначале обучаются классификаторы меньшего размера (например, 32->128->512->2048), которые потом отбрасываются

Статистики естественных изображений

Обозначения: I – исходное изображение; I_s – изображение, имеющее масштаб s; $\phi(I)$ – произвольная скалярная статистика изображения $M[\cdot]$ – матожидание по ансамблю естественных изображений

Утверждение (Ruderman, Bialek, 1994) Отношение $M\left[\phi\left(I_{s_1}\right)\right]/M\left[\phi\left(I_{s_2}\right)\right]$, посчитанное по 2-м ансамблям изображений масштабов s_1 и s_2 соответственно зависит только от отношения масштабов s_1/s_2 и не зависит от абсолютных значений s_1 и s_2.

Следствие: Матожидание скалярных статистик изображений взятое по ансамблю естественных изображений подчиняется степенному закону:

$$\frac{M\left[\phi\left(I_{s_1}\right)\right]}{M\left[\phi\left(I_{s_2}\right)\right]} = \left(\frac{s_1}{s_2}\right)^{-\lambda_\phi}$$

Детектор ICF (Dollar, 2010)

Ускорение детектора ChnFtrs за счет использования вычисления разреженной пирамиды масштабов изображений и интерполяции каналов на основе статистики естественных изображений

Обозначения: $R(I,s)$ – изображение I, смаштабированное в s раз; $\Omega(I)$ – канал, посчитанный на изображении I

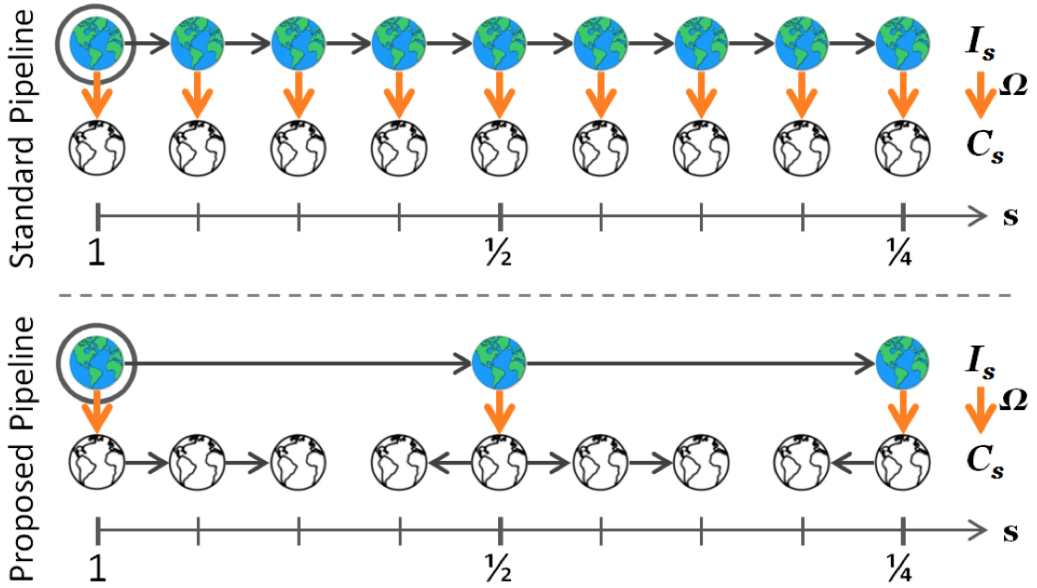
Построение пирамиды признаков:

Классический подход:

$$C_s = \Omega(R(I,s))$$

Предлагаемый подход:

$$C_s = R(C,s) * s^{-\lambda_\Omega}$$



Детектор VeryFast (Benenson, 2012)

В детектор вошли всевозможные ускорение детектора ChnFtrs (реализация CPU+GPU, 1.38 fps без использования Soft Cascade):

- Полное отсутствие масштабирования изображения, использование предобученных классификаторов для каждого масштаба (2.68 fps)
- Использование Soft Cascade для построения модели (50 fps vs. 10 fps для ChnFtrs)
- Для стерео: использование оценки положения земли (100 fps)
- Для стерео: использование оценки положения объекта фиксированной высоты (135 fps)

Детектор Roerei (Benenson, 2013)

Совокупность всевозможных улучшений детектора ChnsFtrs:

- Использование всех возможных хаар-подобных признаков (718К) вместо 30К произвольно выбранных
- Использование многомасштабной модели (аналогично VeryFast)
- Использование глобальной нормализации яркости изображения

Детектор ACF (Dollar, 2014)

Логическое продолжение детектора ICF. Использует те же каналы и подход к построению пирамиды признаков разных масштабов.

Отличия:

- Вместо использования хаар-подобных признаков используются множество lookup-таблиц, построенных по каждому каналу сглаживанием и масштабированием в 2 раза
- Используются локально нормализованные значения модуля градиента

Сравнение качества работы

1. Критерий обнаружения объекта:

$$a_0 = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} > Tr,$$

где BB_{dt} – ограничивающий прямоугольник объекта, полученный от детектора; BB_{gt} – идеальный ограничивающий прямоугольник объекта; Tr – пороговое значение (в эксперименте принято за 0.5)

2. Ключевые характеристики:

    1. Доля найденных объектов:$TP = \frac{\sum_{i=1}^{K}\{1|a_i=1\}}{N}$, где N – кол-во объектов, *K* – кол-во срабатываний детектора

    2. Кол-во ложных срабатываний на изображение: FPPI = $TP = \frac{\sum_{i=1}^{K}\{1|a_i\neq 1\}}{M}$, где *M* – кол-во изображений

Измерение качества работы

1. Сравнение TP при фиксированном FPPI
2. Сравнение ROC-кривых
3. Сравнение средних TP по логарифмическому диапазону FPPI $10^{-2}..10^{0}$ (~устойчивое значение TP при FPPI = 10^{-1}) (Log-average miss rate):

$$MR = \frac{\sum_{i=0}^{N}\{TP \mid FPPI = 10^{f_i}\}}{N+1}, f_i = \{\frac{2(N-i)}{N}\}_{i=0}^{N}$$

Сравнение рассмотренных алгоритмов

Алгоритм	MR, INRIA	MR, Caltech	FPS
VJ	72	94,7	0,45
HOG+SVM	46	68,5	0,24
DPM	20	63	0,63
ChnFtrs	22	56,3	1,2
ICF	21	57,4	2,7
VeryFast	15,4	–	50
Roerei	13	48,35	5
ACF	17	44,2	12
ACF-exact	17	43	32