Цифровая обработка изображений

3. Сегментация и детекция

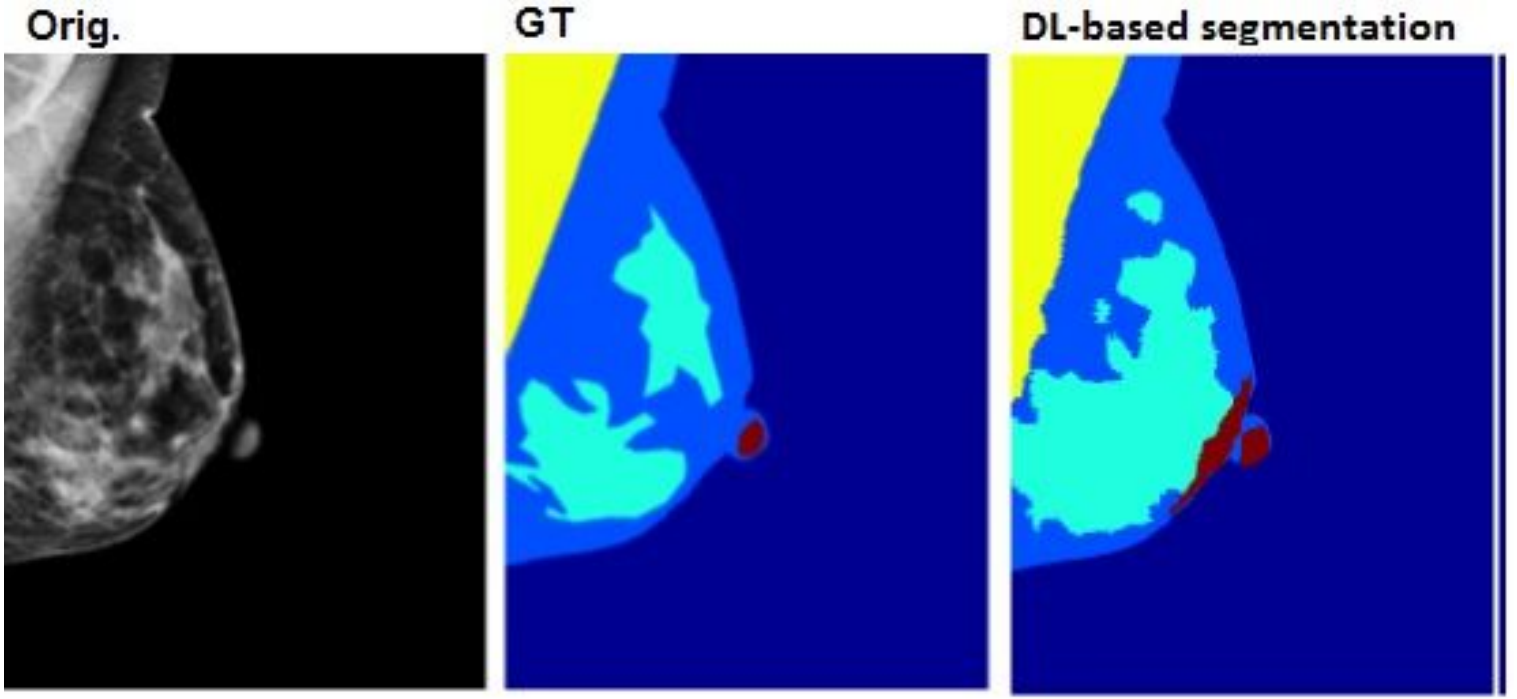
План занятия

3.1 Сегментация
3.2 Детекция объектов
3.3 Трекинг объектов на видео
3.4 Пример. Распознавание жестов

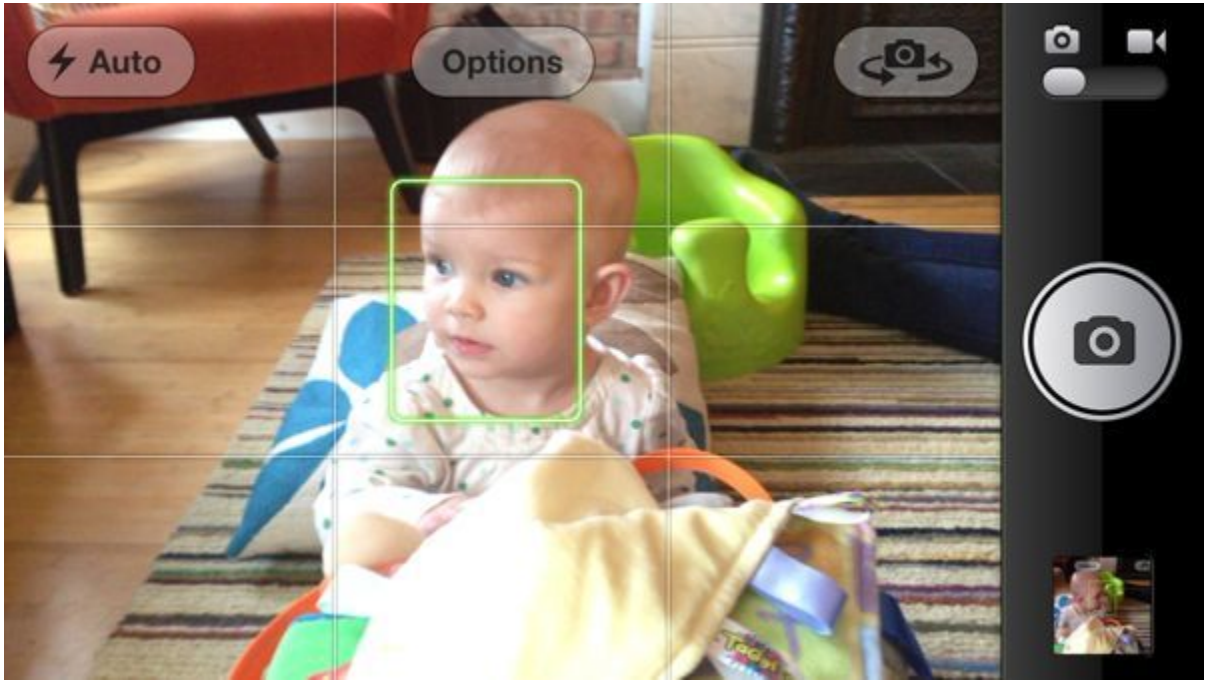
Примеры

Пример. Удаление фона

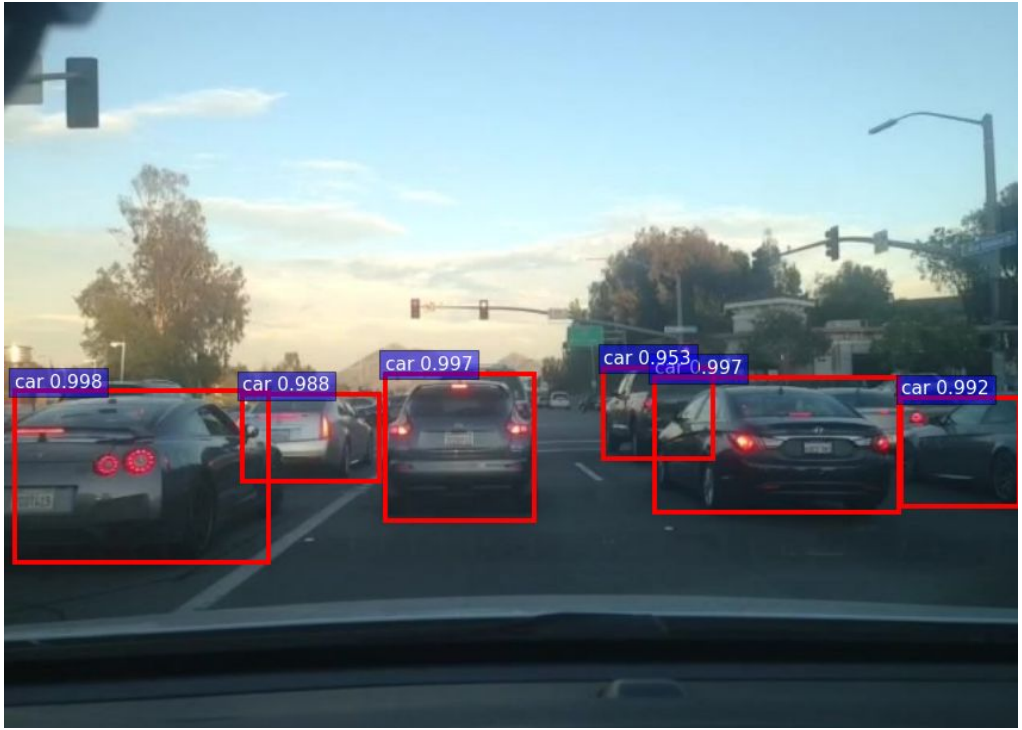
Пример. Сегментация медицинских снимков



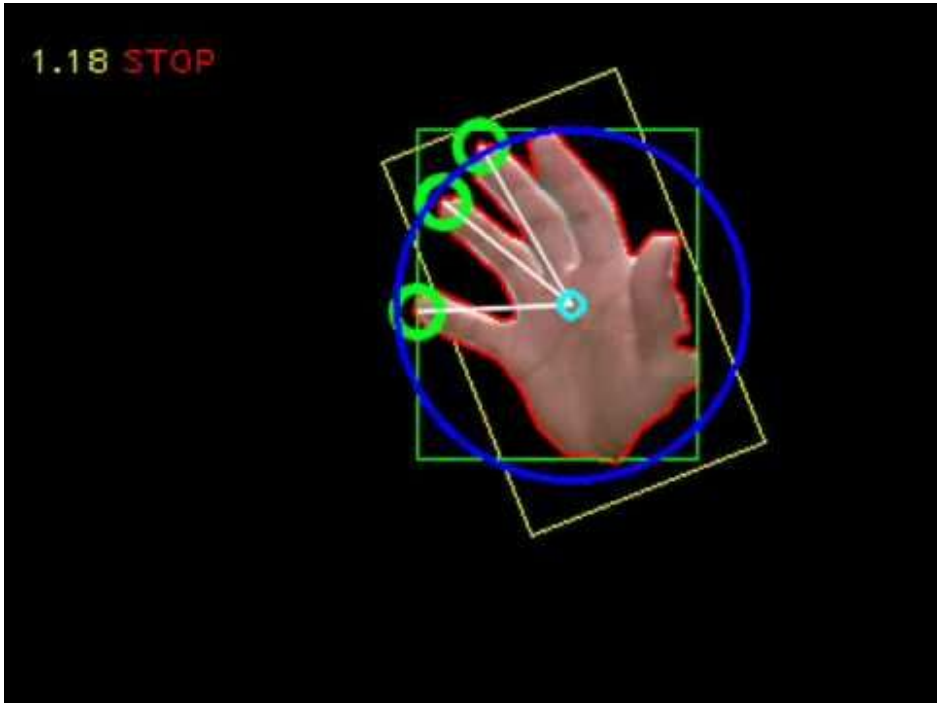
Пример. Детекция лица для автофокусировки камеры



Пример. Детекция и распознавание объектов для оценки дорожной ситуации



Пример. Управление устройством с помощью жестов



https://www.youtube.com/watch?v=mLT4CFLIi8A

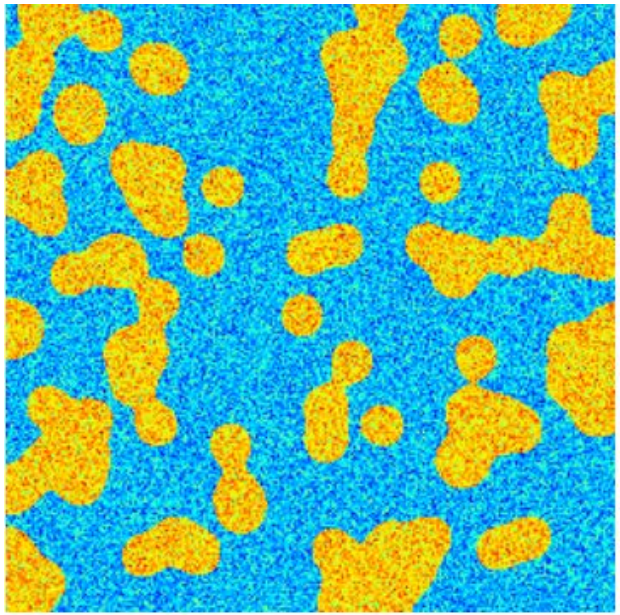
3.1 Сегментация

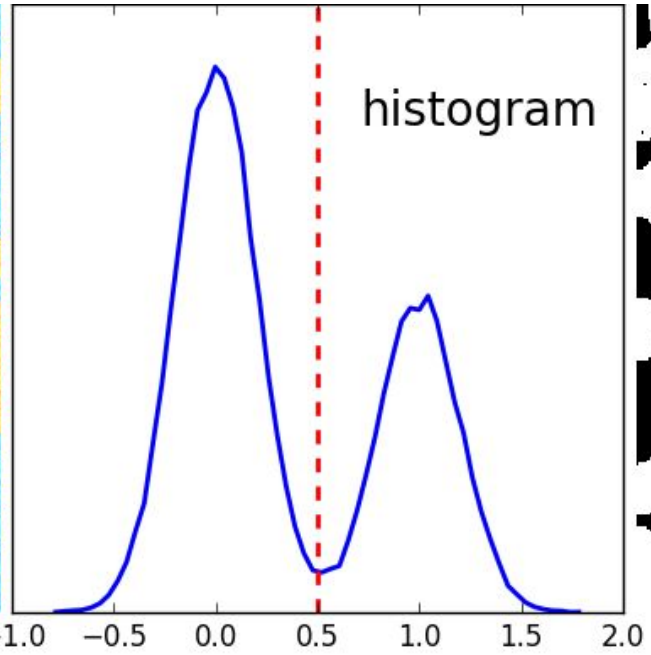
Основные подходы

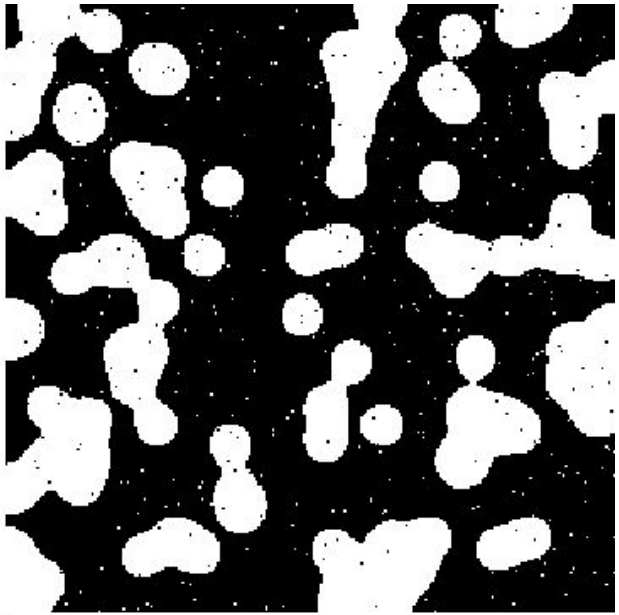
- сегментация по порогу
- выделение контура объекта
- водораздел
- супер-пиксель
- алгоритмы на графах

3.1.1 Сегментация по порогу

Сегментация по порогу







Сегментация по порогу

cv2.**threshold**(src, thresh, maxval, type)

src – исходное изображение

thresh – пороговое значение

maxval – значение записывается, в случае если интенсивность пикселя выше порога

type – cv2.THRESH_BINARY, cv2.THRESH_TRUNC, cv2.THRESH_TOZERO

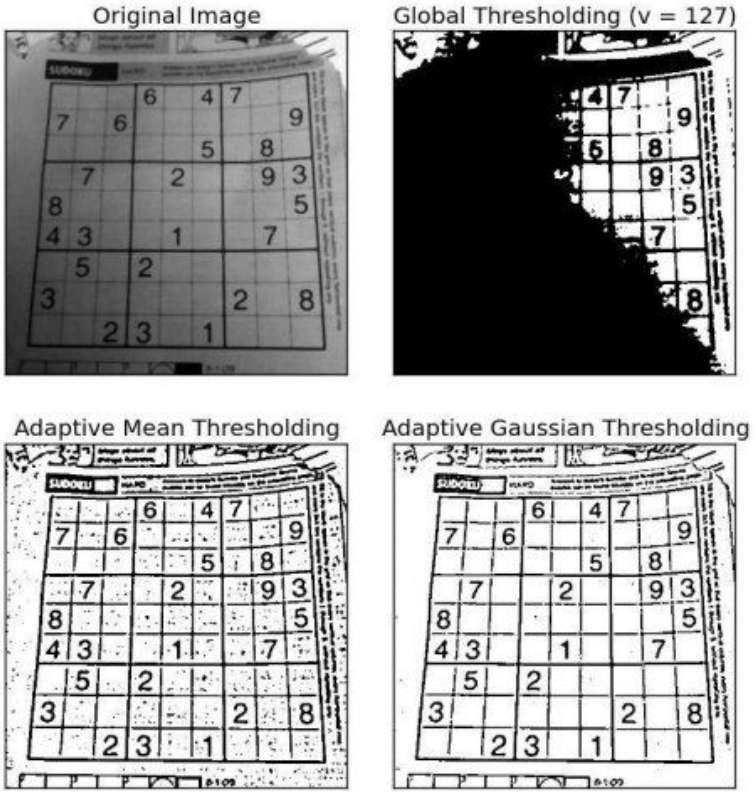
Сегментация по порогу

- прост в реализации
- неустойчив к шумам
- неустойчив к изменению освещенности различных частей изображения

Адаптивный порог

- для каждого пикселя вычисляется усредненное значение интенсивности в окрестности
- в качестве значения порога используют либо само среднее значение
- либо вычисляется взвешенное среднее с гауссовским ядром

Адаптивный порог



Адаптивный порог

cv2.**adaptiveThreshold**(src, maxValue, adaptiveMethod, thresholdType, blockSize, C)

src - исходное одноканальное изображение

maxValue - значение пикселей для которых интенсивность выше порога

adaptiveMethod - метод оценки порога: cv2.ADAPTIVE_THRESH_MEAN_C, cv2.ADAPTIVE_THRESH_GAUSSIAN_C

thresholdType – тип порога: cv2.THRESH_BINARY или cv2.THRESH_BINARY_INV

blockSize – размер блока для вычисления порога: 3, 5, 7 и т.д.

C – константа, вычитается из оценки значения порога

Адаптивный порог

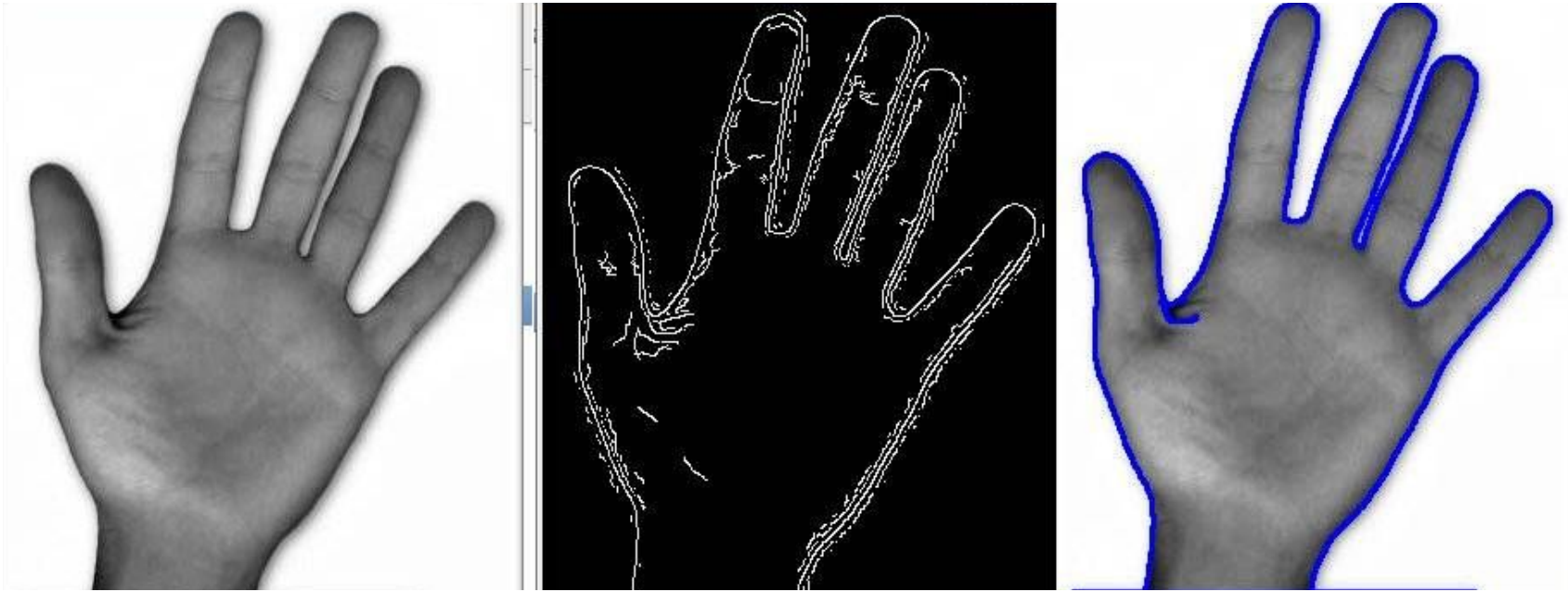
- устойчив к локальным изменениям на изображении
- требует больше вычислительных ресурсов по сравнению с бинарным порогом

3.1.2 Выделение контура объекта

Выделение контура объекта

- находит границы объекта на изображении
- добавляет информацию о форме объекта
- можно использовать для детекции и распознавания объектов

Выделение контура объекта



Выделение контура объекта

- для поиска контура необходимо бинаризовать исходное изображение
- бинаризовать можно по порогу
- либо применить предобработку с помощью детектора граней
- для получения информации о форме объекта полученные контуры необходимо аппроксимировать полигоном

Выделение контура объекта

cv2.**findContours**(image, mode, method[, contours[, hierarchy[, offset]]]) → contours, hierarchy

image - бинаризованное изображение
mode - определяет какие контуры необходимо найти,
cv2.CV_RETR_EXTERNAL, cv2.CV_RETR_TREE
method - метод поиска и фильтрации точек контура
contours – массив контуров, каждый контур - это массив точек
hierarchy - задает иерархию вложенности контуров
offset - задает сдвиг точек контура

Выделение контура объекта

- метод требует предобработки (бинаризации) изображения
- время работы зависит от сложности входного изображения
- позволяет получить информацию о форме сегмента

3.1.3 Водораздел

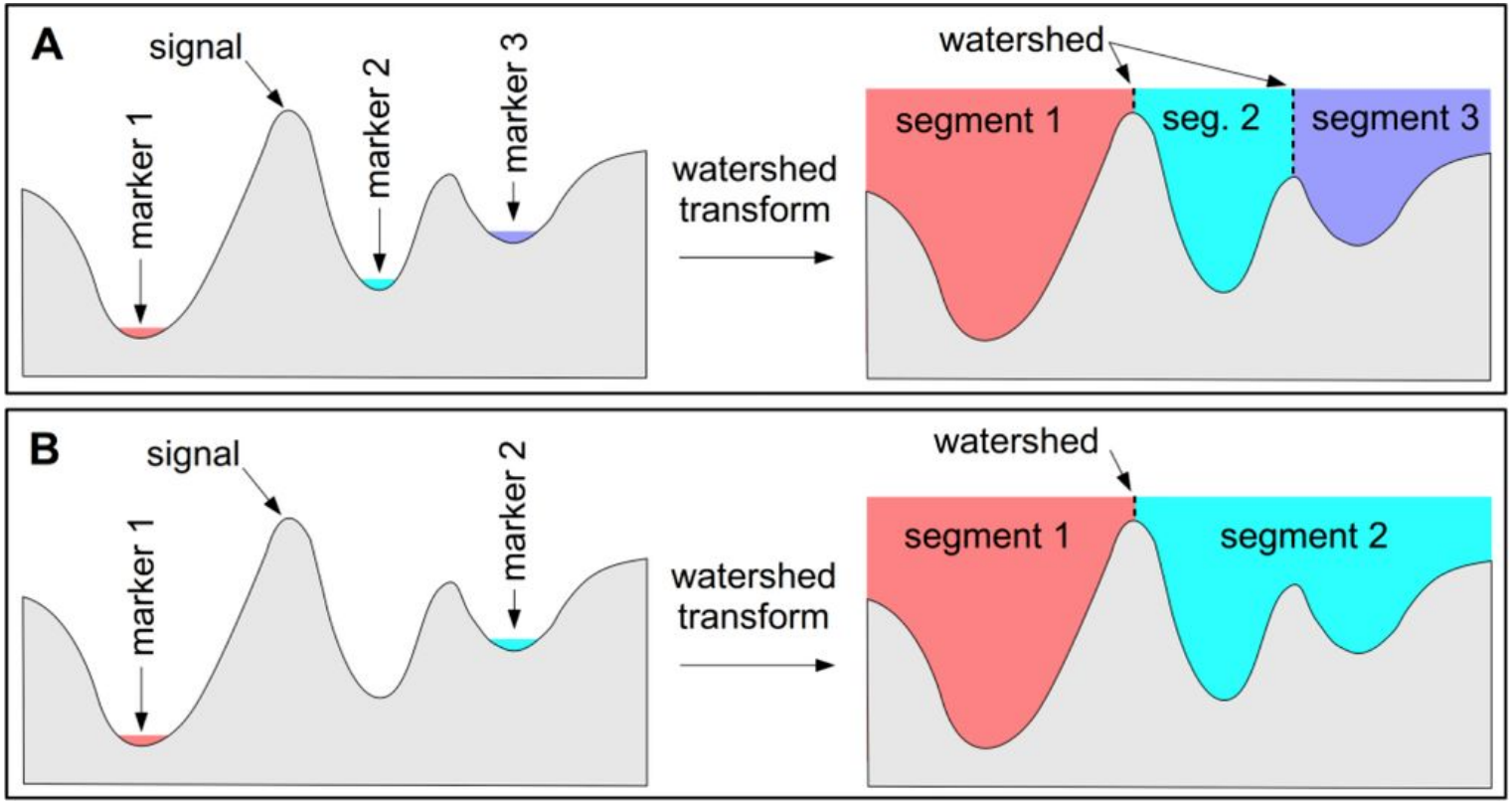
Водораздел (Watershed)

- изображение представляется в виде рельефа
- точки с высокой интенсивностью - возвышенности
- точки с низкой интенсивностью - низины
- помимо интенсивности, используют фактор расстояния пикселя до грани

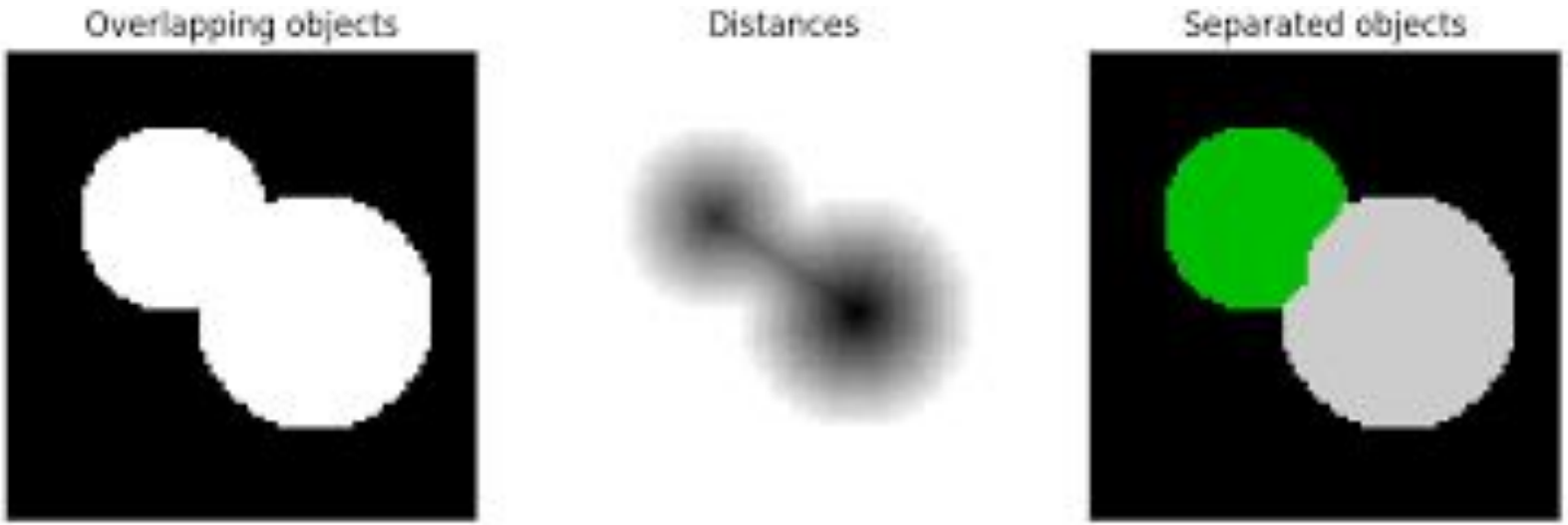
Водораздел (Watershed)

- начинаем заполнять низины
- там где сталкиваются области из разных бассейнов проходит граница объекта
- заполняем до тех пор, пока все вершин не скроются под водой

Водораздел (Watershed)



Водораздел (Watershed)



Водораздел (Watershed)

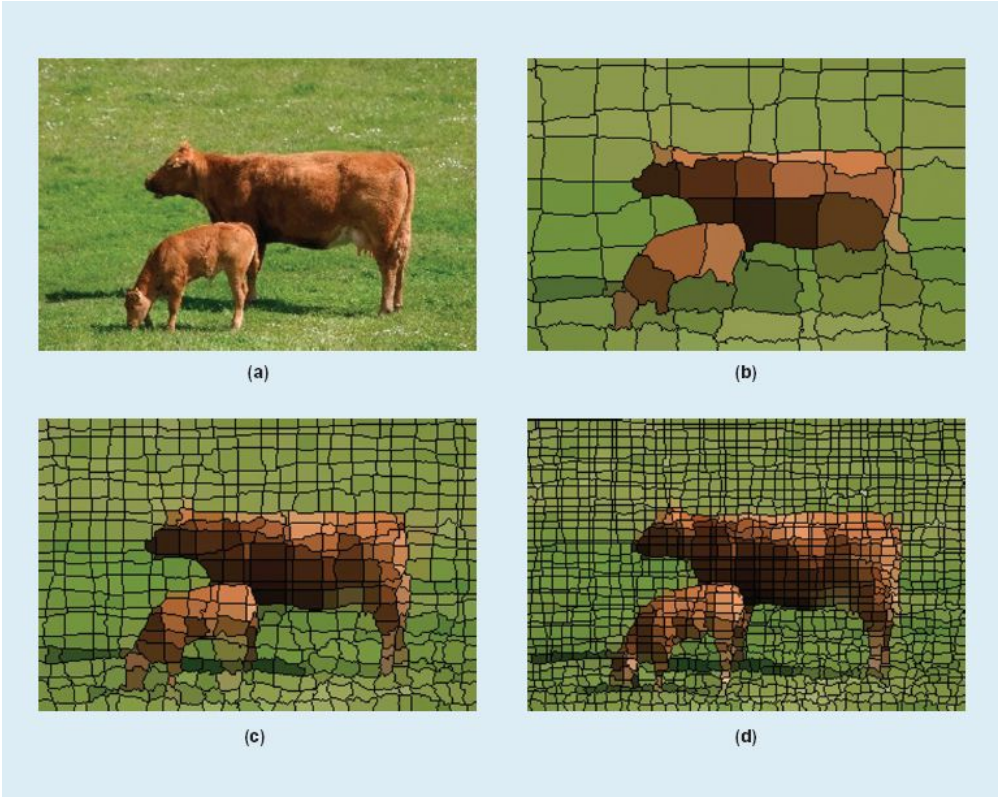
- из-за шума и не идеальной структуры сегменты могут объединяться в один
- необходимо указать начальное приближение (маркеры) для сегментации

Супер-пиксель (super-pixel)

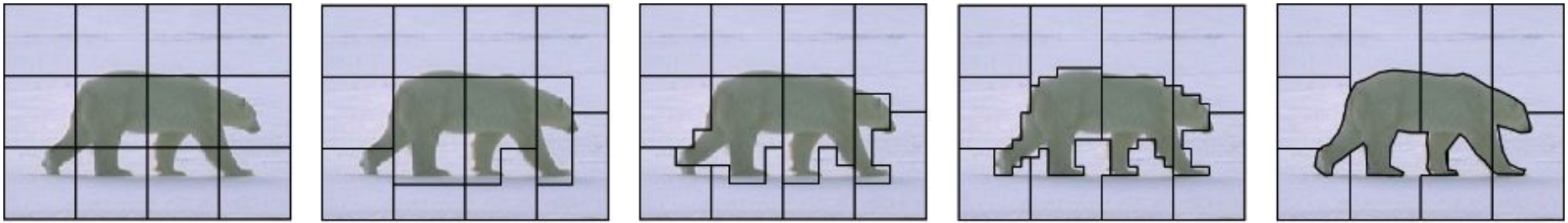
Супер-пиксель

- объединение нескольких соседних пикселей на изображении в единую область
- объединение происходит по таким признакам как цвет или текстура
- в результате объединения пикселей получается упрощенное представление изображения, которое можно использовать в задачах распознавания

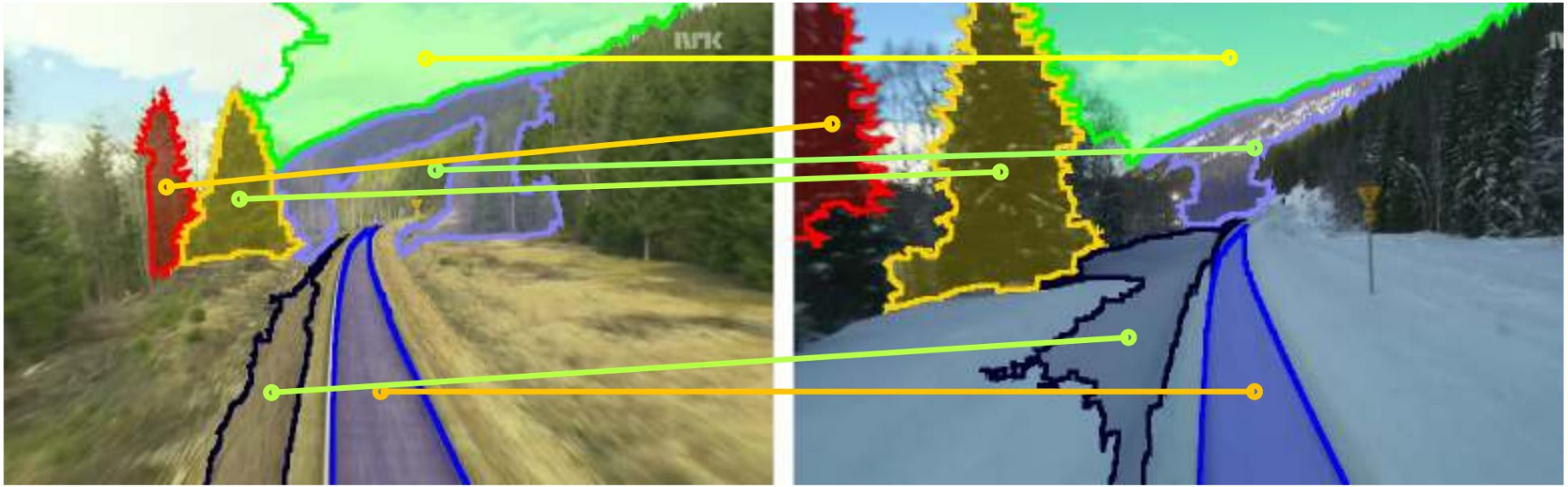
Супер-пиксель



Superpixels Extracted via Energy-Driven Sampling



Распознавание географических мест на фото при изменении окружения



Graph Cut

Graph Cut

- необходимо задать начальные приближения для пикселей фона и объекта
- по этим приближениям оцениваются распределения цвета фона и картинки
- зная эти распределения, можно получить вероятность принадлежности пикселя фону или картинке

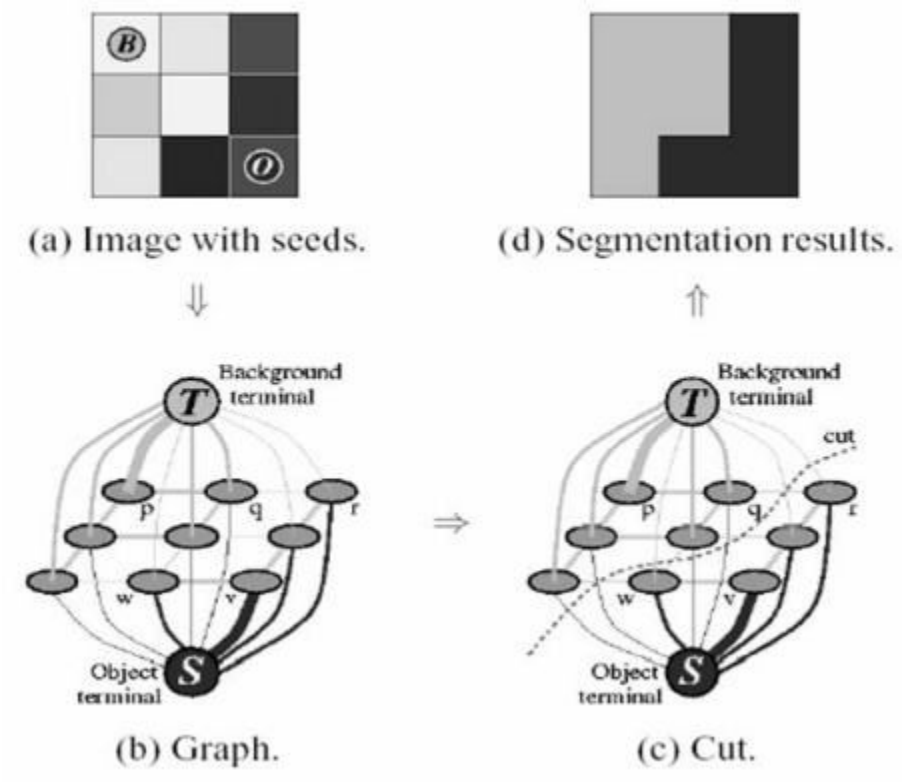
Graph Cut

- строится граф с двумя вершинами: источник (фон) и сток (объект)
- каждый пиксель изображения соединяется с источником и стоком
- вес ребра пропорционален вероятности принадлежности пикселя фону и объекту соответственно

Graph Cut

- пикселы изображения разделяются на части алгоритмом Minmium Cut
- алгоритм разрезает связный граф на две части таким образом, чтобы сумма весов ребер через которые проходит разрез была минимальна

Graph Cut


B
O
(a) Image with seeds.
(d) Segmentation results.
Background terminal
T
p
q
r
w
v
Object terminal
S
cut
(b) Graph.
(c) Cut.

3.2 Детекция объектов

Постановка задачи

- Детекция
  - поиск объекта на изображении
  - в результате получаем область на изображении с соответствующим объектом
  - в некоторых случаях выдается оценка вероятность детекции

3.2 Детектор лиц на изображении

Детектор лиц на изображении

Детектор лиц на изображении

- детектор по деталям
  - отдельно детектируем части лица (глаза, нос, рот)
  - объединяем части в результат детекции
- детектор лица в целом
  - сканирующим окном проходим по изображению
  - оцениваем похожесть на лицо каждую часть изображения
  - есть ограничение на допустимый размер окна

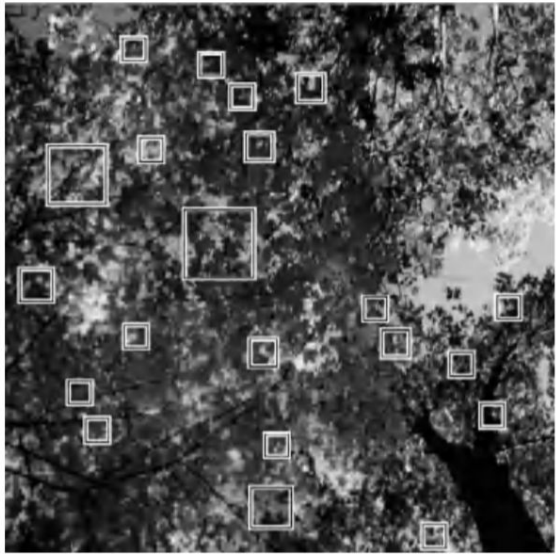
Подготовка данных

- подготавливаем положительные и отрицательные примеры изображений
- как правило положительных примеров больше, чем отрицательных
- для баланса выборки можно применить аугментацию изображений

Аугментация изображения

- масштабирование
- поворот
- сдвиг
- зеркальное отображение
- вычитание градиента (эмуляция освещенности)
- выравнивание гистограммы интенсивности

Подготовка данных



Обучение модели

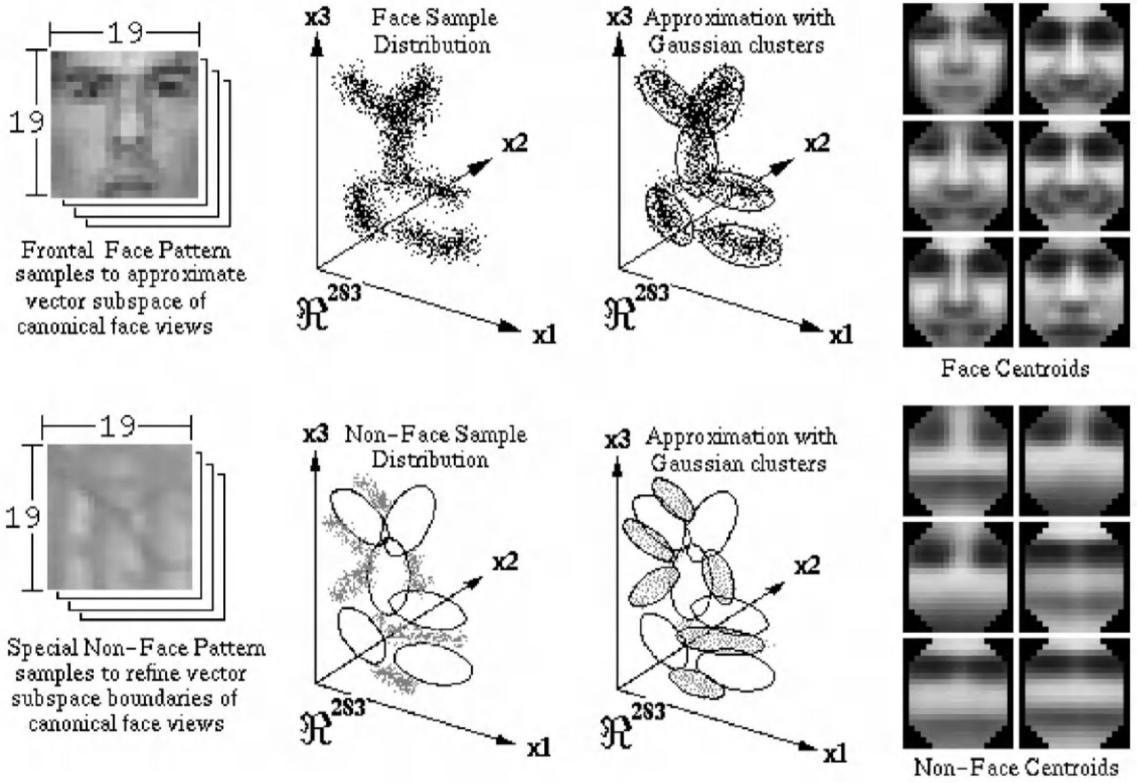
- PCA + Кластеризация
- Нейросеть
- Бустинг (Viola, Jones)

Кластеризация + PCA

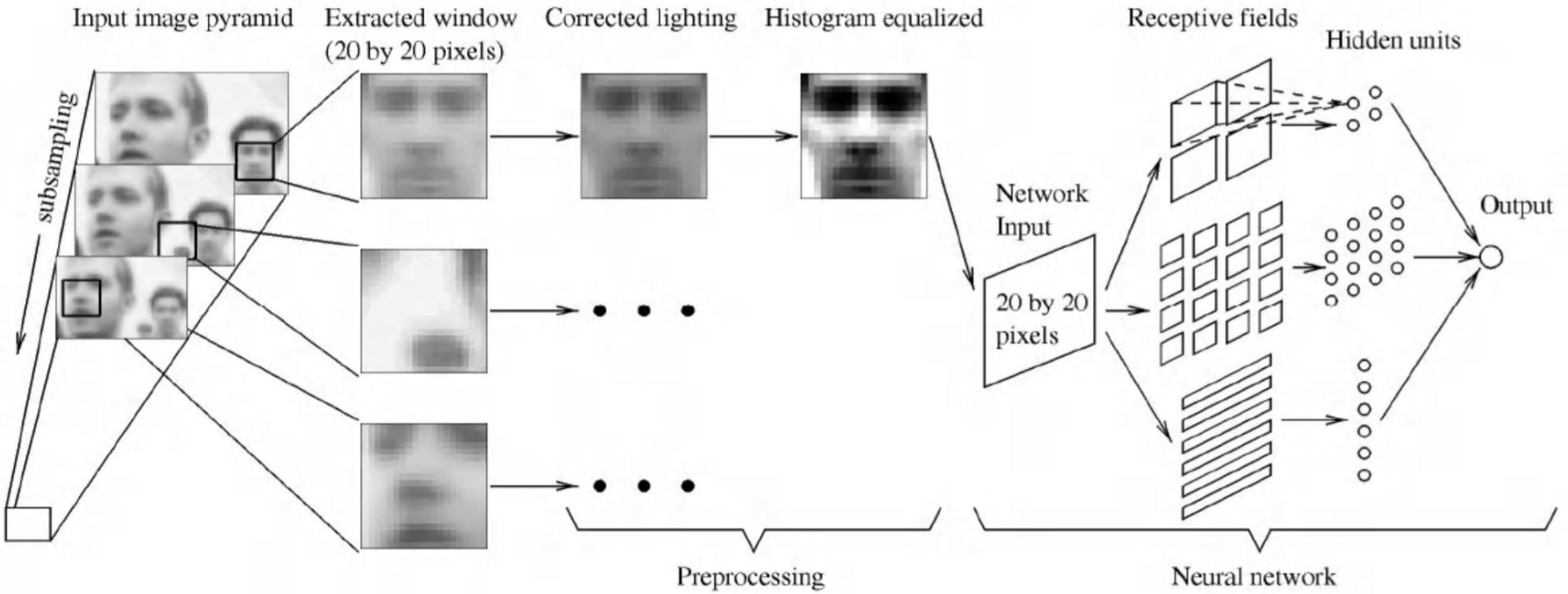
- снижаем размерность изображения с помощью PCA преобразования
- кластеризуем изображения в каждой группе на 6 кластеров
- в результате получается 12 центроидов
- расстояние от изображения до центроида используем в качестве фактора для обучения
- обучаем модель SVM, RandomForest, Neural Network

Probabilistic Visual Learning for Object Detection

Кластеризация + PCA



Нейронная сеть



Бустинг (Viola, Jones)

- детектор Viola и Jones
- предложен подход с использованием бустинга в задачах компьютерного зрения

Rapid Object Detection using a Boosted Cascade of Simple Features

Бустинг (Viola, Jones)

$$h(\boldsymbol{x}) = \operatorname{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right]$$

h(x) - финальный классификатор
m - число классификаторов в ансамбле
h_j(x) - базовый классификатор
alpha_j - вес базового классификатора (зависит от ошибки)

Бустинг (Viola, Jones)

$$h_j(\boldsymbol{x}) = a_j[f_j < \theta_j] + b_j[f_j \geq \theta_j] = \begin{cases} a_j & \text{if } f_j < \theta_j \\ b_j & \text{otherwise,} \end{cases}$$

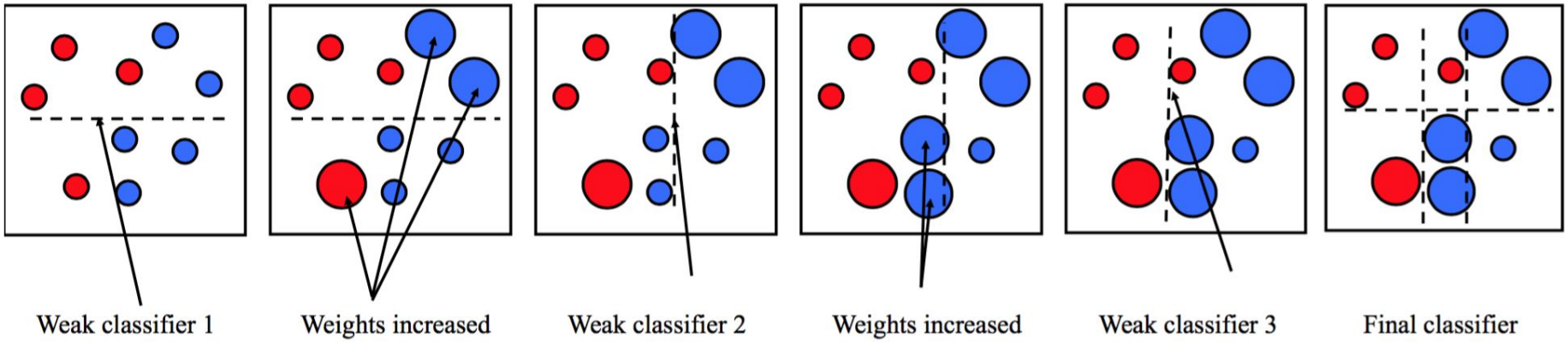
a_j - решение принимается в случае если значение фактора f меньше порога theta

b_j - принимается в противном случае

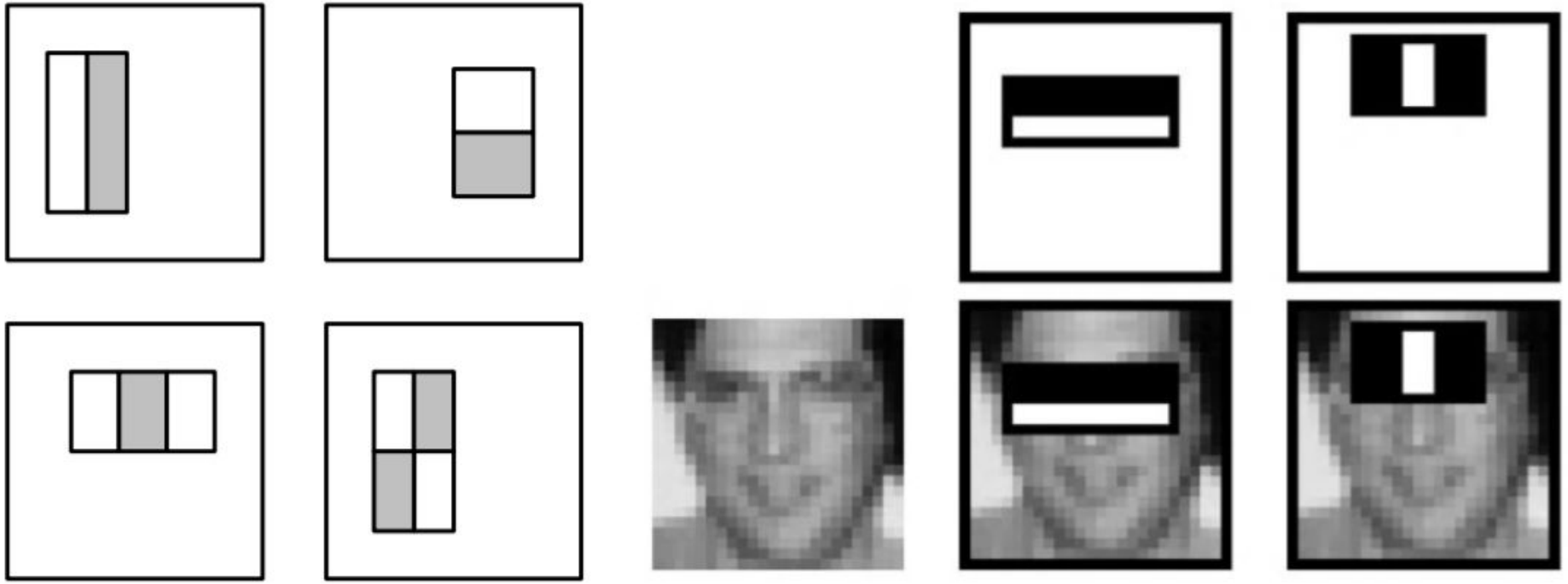
Бустинг (Viola, Jones)

- при инициализации присваиваем всем объектам в выборке одинаковый вес
- на очередном шаге j среди множества простых классификаторов выбираем тот, который дает наибольший прирост качества
- вычисляем коэффициент alpha пропорциональный приросту качества с учетом весов объектов
- увеличиваем вес объектов, которые остаются классифицированы неверно

Бустинг (Viola, Jones)



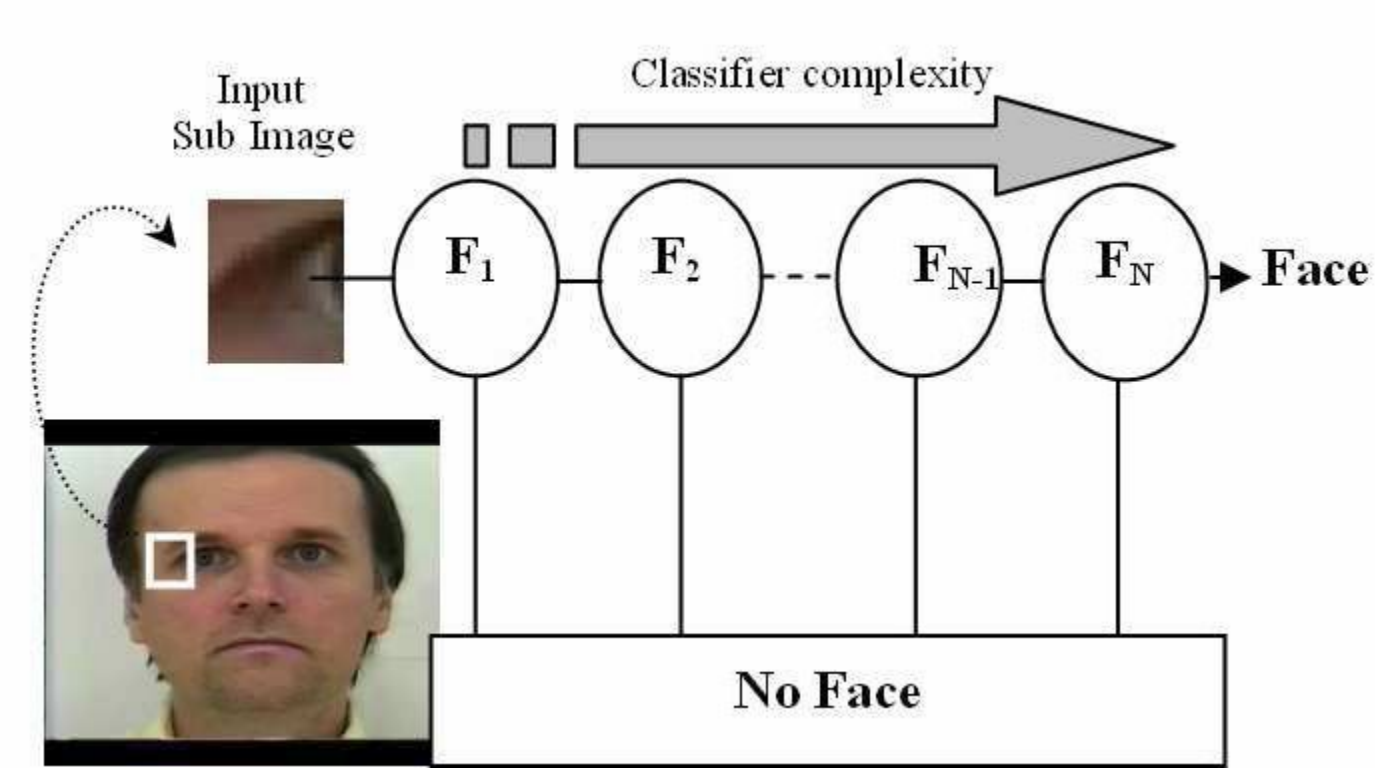
Бустинг (Viola, Jones)



Бустинг (Viola, Jones)

- ориентировочное число простых моделей в ансамбле ~3000
- поиска скользящим окном по изображению большим ансамблем может занимать слишком много времени
- на практике строят каскады ансамблей меньшего размера
- область классифицируется очередным ансамблем в каскаде, только если предыдущий ансамбль классифицировал область как положительную

Бустинг (Viola, Jones)



Бустинг (Viola, Jones)

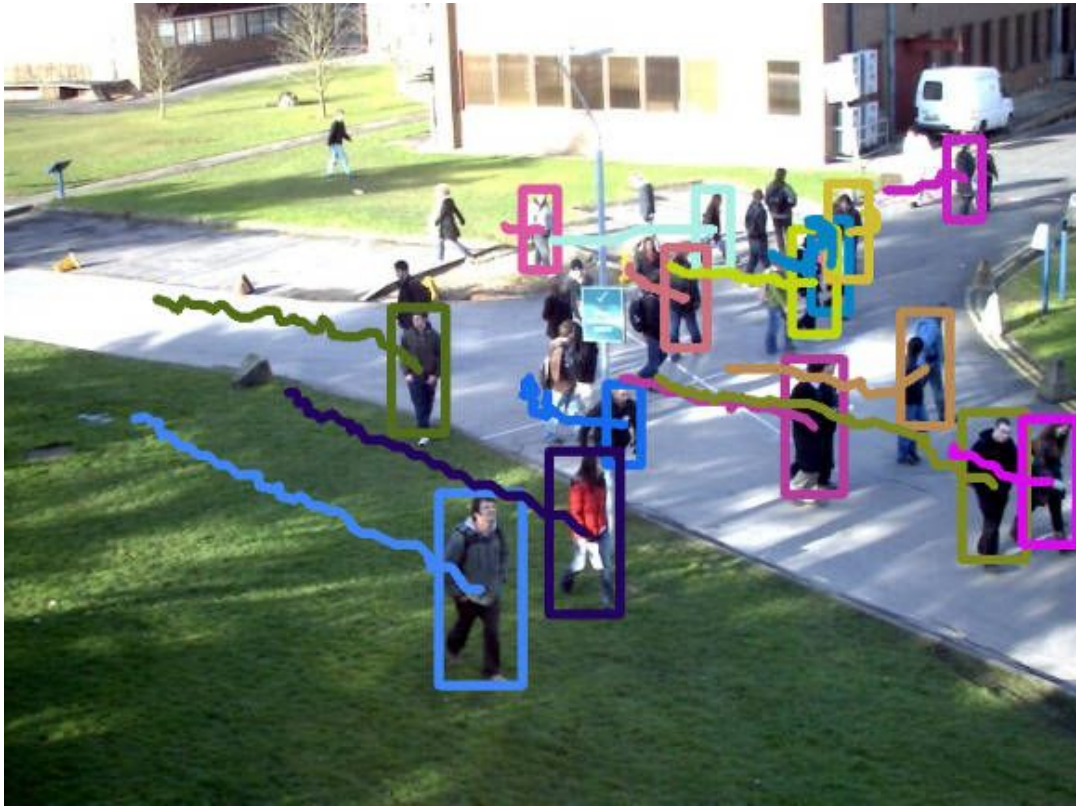
- предобученные модели для opencv доступны по ссылке
- opencv предоставляет утилиты для подготовки данных обучения детектора: инструкция

Бустинг (Viola, Jones)

- алгоритм сравнительно быстро вычисляет детекции
- но медленно обучается, тк на каждом шаге необходимо перебрать большое число вариантов простых классификаторов

3.3 Трекинг объектов на видео

Трекинг



Постановка задачи

- отслеживание перемещения объекта в кадре
- предсказание направления перемещения
- сообщение в случае потери объекта

Сложности

- несколько объектов на кадре
- частичное или полное перекрытие другими объектами
- реидентификация после временной потери из зоны видимости
- объект может изменять внешний вид во время наблюдения (повороты, масштаб)

Детекция и трекинг

- как правило, детекция сложнее чем трекинг
- трекинг использует информацию из предыдущих кадров
- трекер может накапливать ошибку
- детектор ищет объект на изображении без учета предыдущих кадров
- на практике алгоритм детекции запускается реже чем трекинг

Обзор подходов

- оптический поток (Kanade-Lucas-Tomashi)
  - трекинг характерных точек между кадрами
- калмановский фильтр
  - использует априорную информацию о возможных перемещениях объекта для предсказания позиции в будущем

Обзор подходов

- трекинг отдельного объекта
  - детектируем объект на первом кадре, затем отслеживаем его перемещение с помощью трекера
- трекинг нескольких объектов
  - детектор для каждого кадра детектор выдает предсказания, задача трекера - поставить в соответствие детекции между кадрами

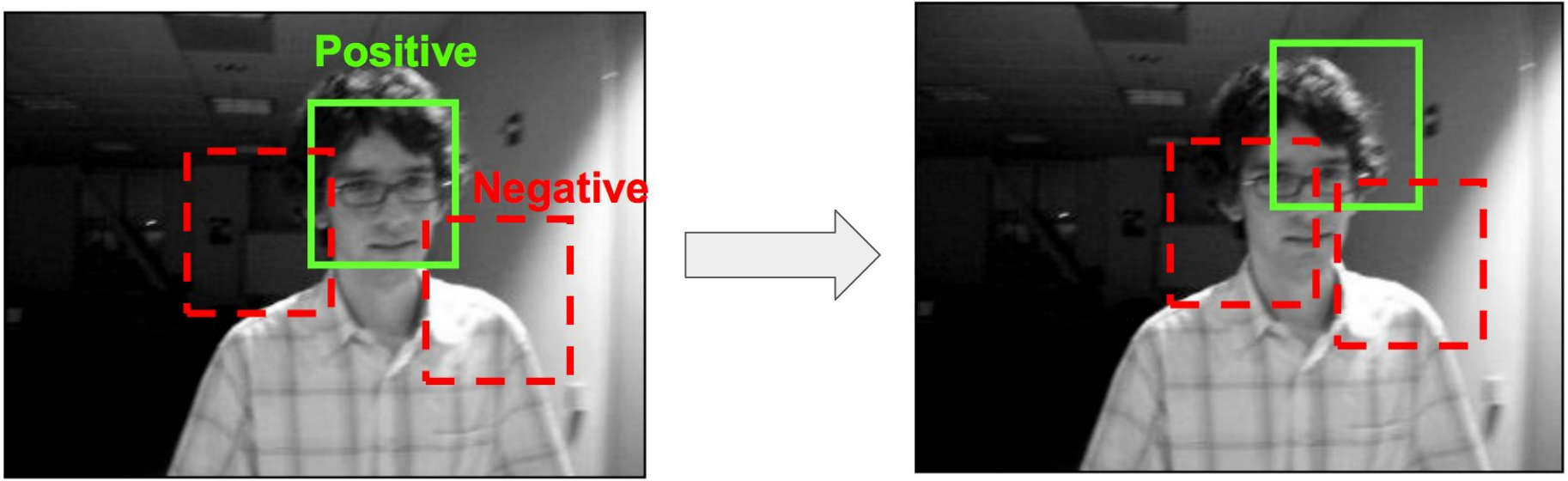
Алгоритмы трекинга в OpenCV

- для использования трекеров в opencv необходимо установить пакет opencv-contrib-python
- трекеру необходимо хранить состояние между вызовами
- состояние хранится в объекте трекера
- создать объект трекера можно с помощью функции *cv2.Tracker<алгоритм_трекинга>_create()*

Boosting Tracker

- основан на онлайн версии алгоритма AdaBoost
- детекция является положительным примером для обучения
- область вокруг детекции - отрицательные примеры
- для определения объекта на очередном фрейме запускается поиск в окрестности предыдущей детекции
- качество работы среднее

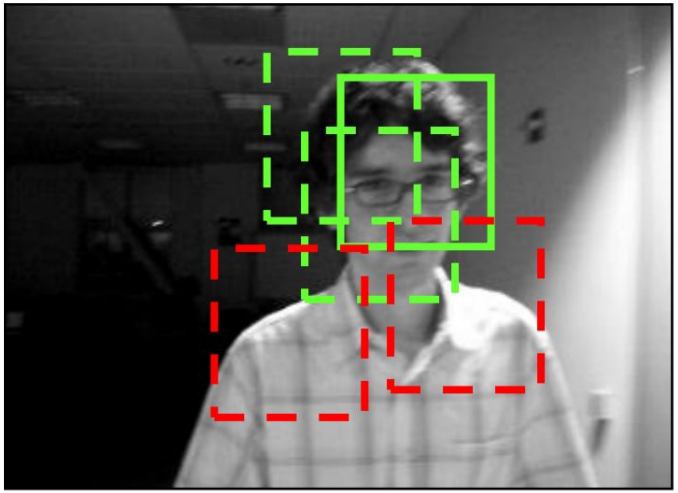
Boosting Tracker

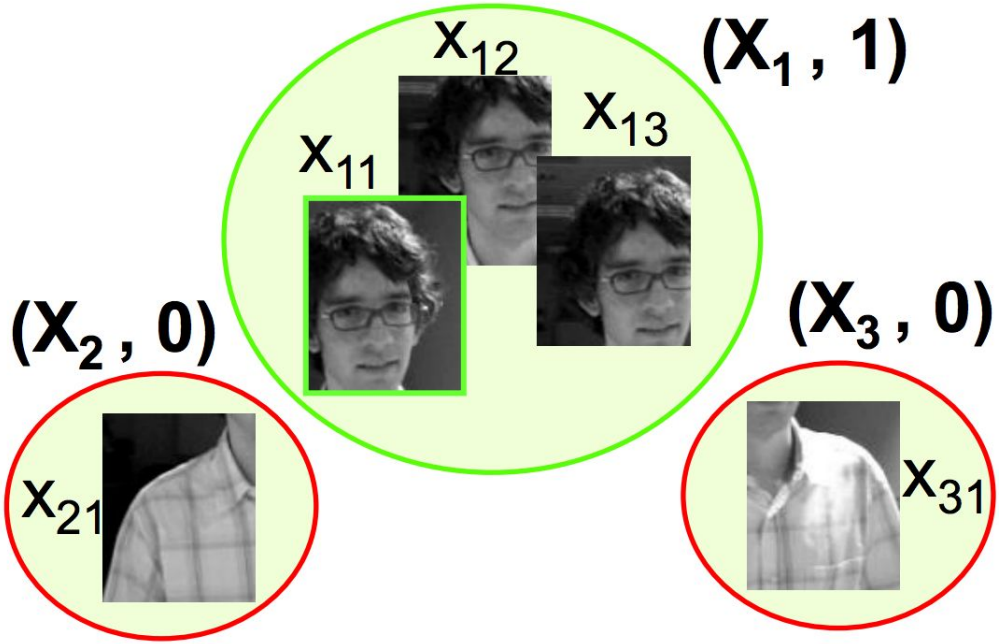


MIL - Multiple Instance Learning

- решение аналогично Boosting Tracker
- в отличие от Boosting Tracker семплируется несколько примеров вокруг вокруг детекции
- полученные семплы объединяются в группу
- задача классификатора предсказать класс группы
- качество детекции выше, чем у Boosting Tracker
- устойчив к частичным перекрытием объектов
- не восстанавливается после полной потери объекта из зоны видимости.

MIL - Multiple Instance Learning





http://vision.stanford.edu/teaching/cs231b_spring1415/slides/MIL_kelsie.pdf

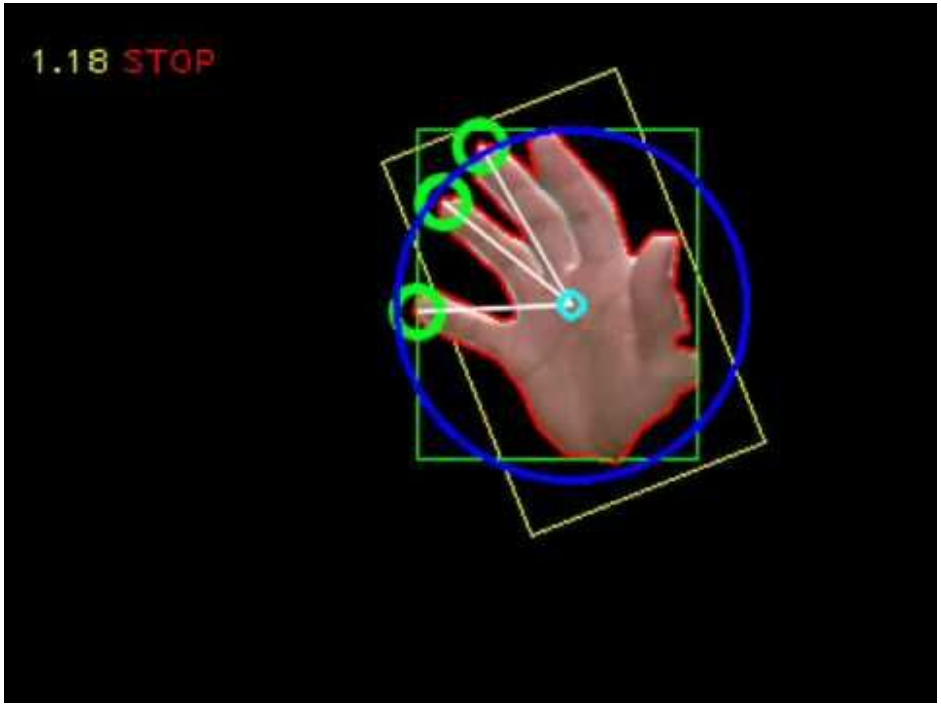
KCF - Kernelized Correlation Filters

- идея семплирования положительных примеров, аналогичная MIL
- используется факт перекрытия положительных семплов
- скорость работы и качество выше, чем у Boosting Tracker и MIL
- не устойчив к потере объекта из зоны видимости

High-Speed Tracking with Kernelized Correlation Filters

3.4 Пример. Распознавание жестов

Распознавание жестов



Постановка задачи

- разработать программируемую систему управления компьютером с помощью жестов
- жесты распознаются в реальном виде
- видеопоток поступает в реальном времени через веб-камеру

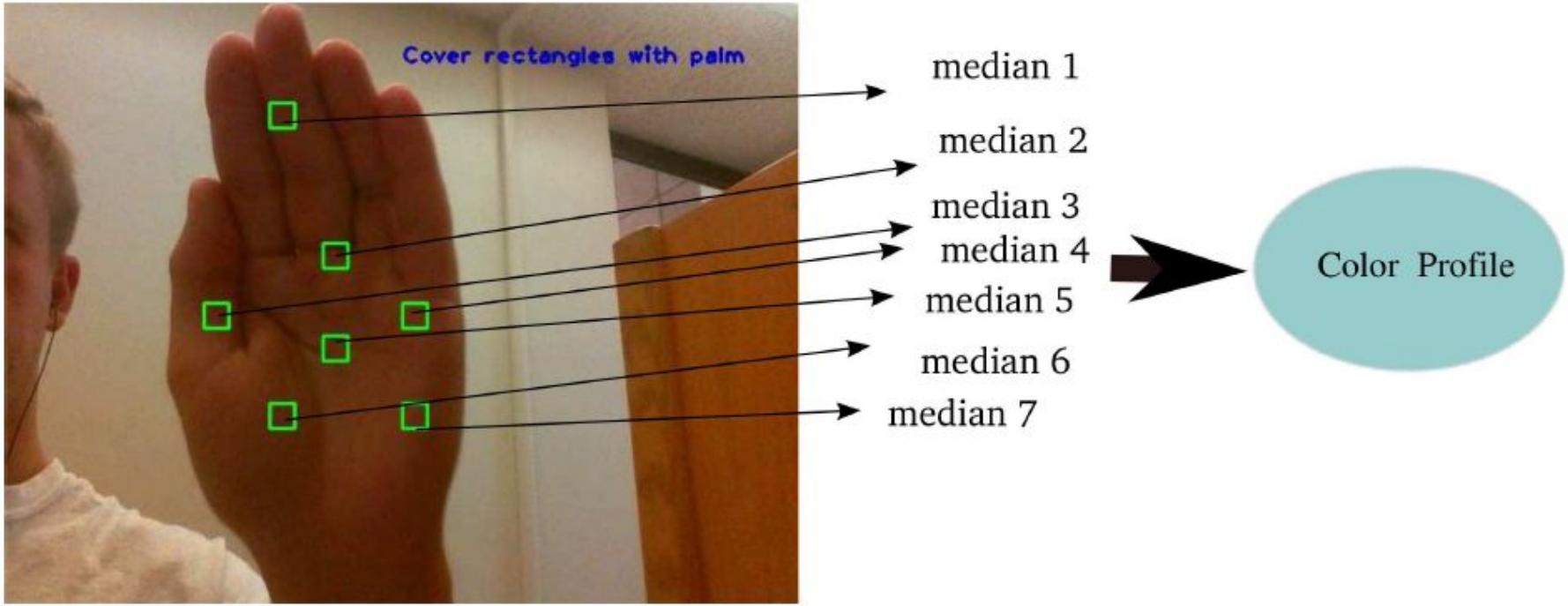
Этапы решения

- сегментация: отделение кисти от фона
- детекция: определяем положение кисти на изображении
- распознавание жеста

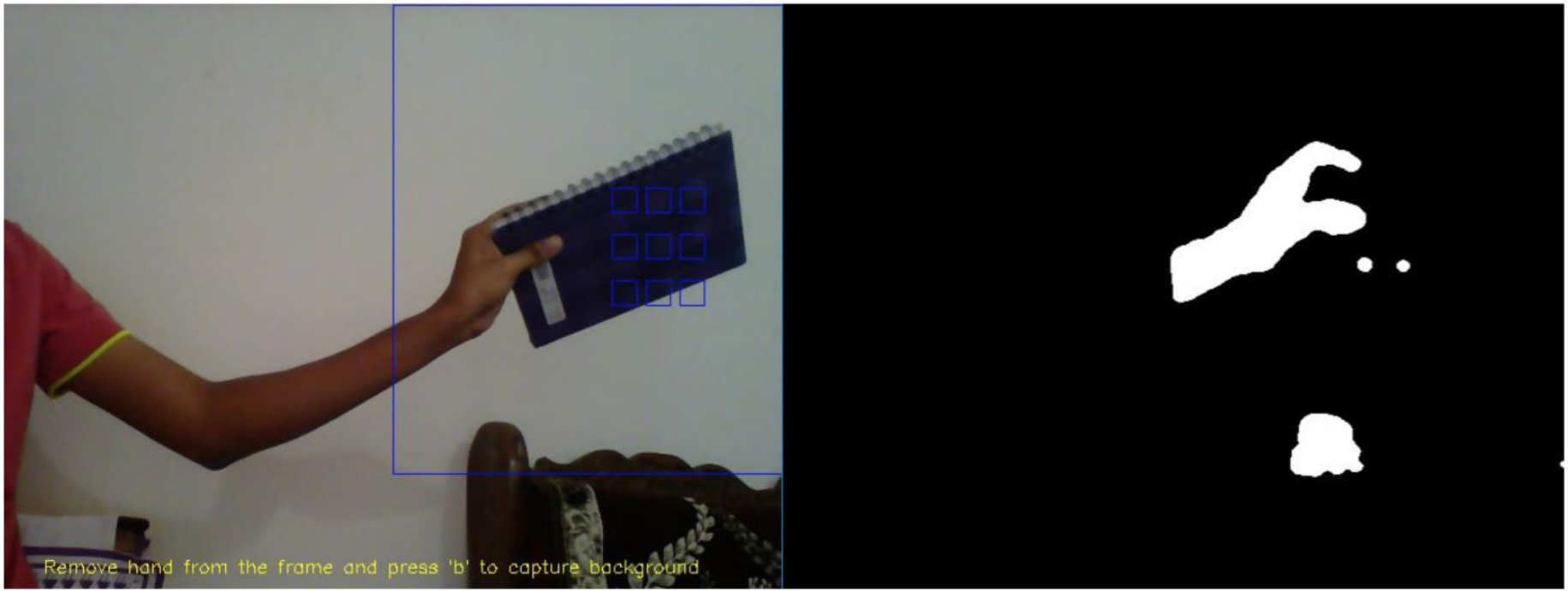
Сегментация кисти

- задаем области на изображении, которые относятся к кисти
- оцениваем гистограмму цветов в пространстве HSV
- используем эту оценку для сегментации
- сегментируем по гистограмме кисть функцией cv2.calcBackProject

Сегментация кисти



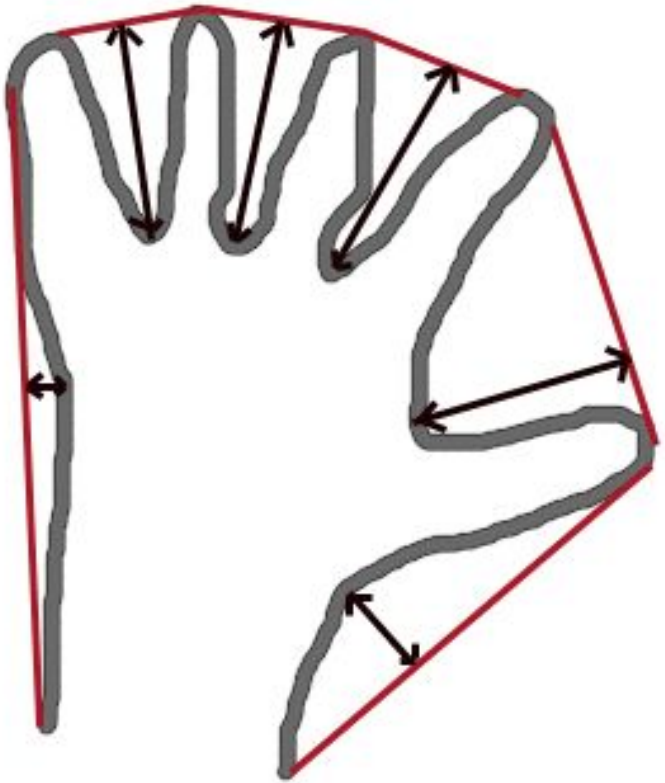
Сегментация кисти



Детекция кисти

- находим все контуры сегментированного изображения
- контур наибольшей площадью считаем кистью
- находим выпуклую оболочку контура с помощью функции cv2.convexHull

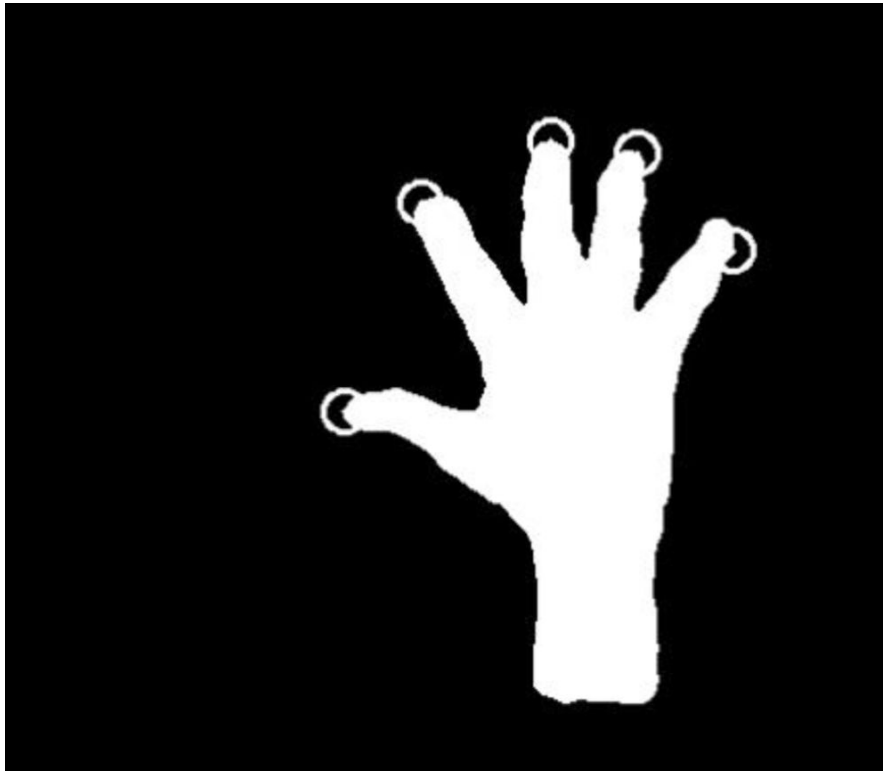
Детекция кисти



Детекция кисти

- оцениваем центр ладони
- оцениваем радиус кисти
- удаляем точки на выпуклой оболочки таким образом, чтобы осталось не более пяти точек (одна точка на палец)

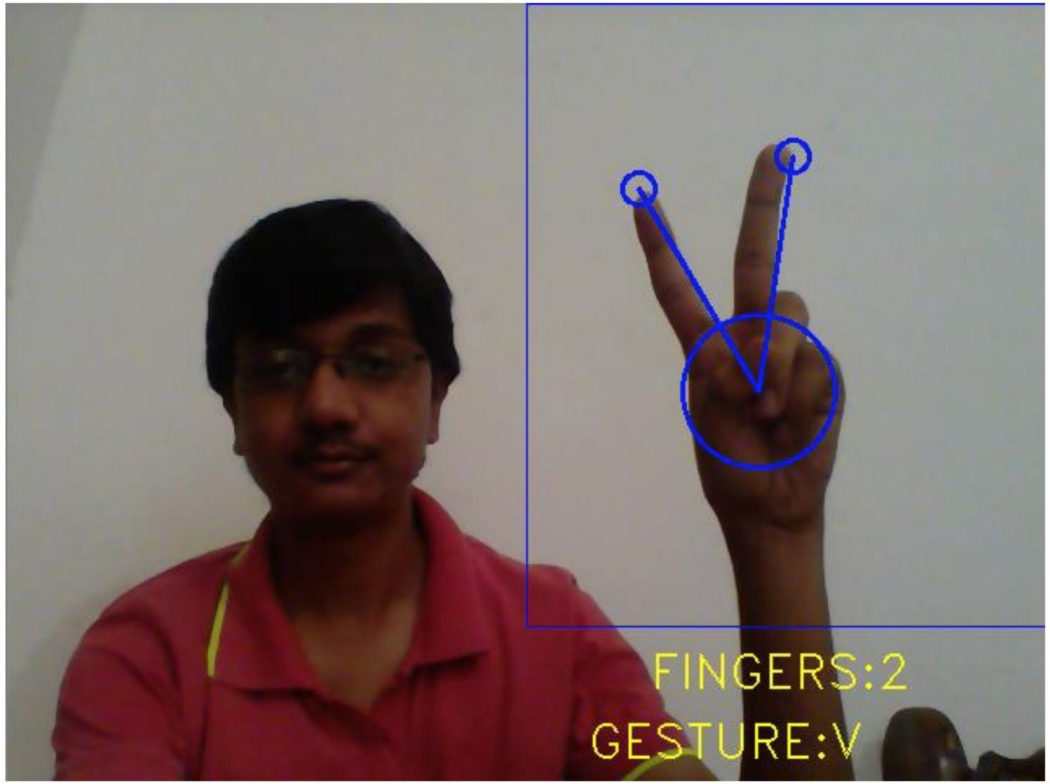
Детекция кисти



Распознавание жеста

- входные данные: координаты центра ладони, радиус и координаты пальцев
- задаем базу размеченных жестов в таком же формате
- ищем в базе наиболее близкий жест к детектированному

Распознавание жеста



Резюме

- сегментация по порогу
- выделение контура объекта
- водораздел
- суперпиксели
- graph cut

Резюме

- эффективным способом детекции объектов является каскад AdaBoost классификаторов
- в качестве признаков для базовых классификаторов используются разностные свертки
- метод быстро работает на этапе применения
- но требует сравнительно много времени и большого числа примеров для обучения

Резюме

- один из способов решения задачи трекинга - трекинг через детекцию (tracking by detection)
- на кадре с детекцией обучаем модель
- обученную модель применяем на следующем шаге

Полезные материалы

Computer Vision: Algorithms and Applications (Chapter 5)
Computer Vision: Algorithms and Applications (Chapter 14)
Image segmentation (Wikipedia)
Visual Tracking: An Experimental Survey