Биометрия

02

Метод Viola Jones

- используются изображения в интегральном представлении, что позволяет вычислять быстро необходимые объекты;
- используются признаки Хаара, с помощью которых происходит поиск нужного объекта (в данном контексте, лица и его черт);
- используется бустинг (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения;
- все признаки поступают на вход классификатора, который даёт результат «верно» либо «ложь»;
- используются каскады признаков для быстрого отбрасывания окон, где не найдено лицо.

- https://habrahabr.ru/post/135244/
- https://habrahabr.ru/post/134857/
- https://habrahabr.ru/post/133909/
- https://habrahabr.ru/post/133826/

Метод Viola Jones. Основные принципы

- используются изображения в интегральном представлении, что позволяет вычислять быстро необходимые объекты;
- используются признаки Хаара, с помощью которых происходит поиск нужного объекта (в данном контексте, лица и его черт);
- используется бустинг (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения;
- все признаки поступают на вход классификатора, который даёт результат «верно» либо «ложь»;
- используются каскады признаков для быстрого отбрасывания окон, где не найдено лицо.

Краткий алгоритм

- имеется *изображение*, на котором *есть искомые объекты*. Оно представлено *двумерной матрицей пикселей размером* ***w*h***, в которой каждый пиксель имеет значение:
  — от 0 до 255, если это черно-белое изображение;
  — от 0 до 255^3, если это цветное изображение (компоненты R, G, B).
- в результате своей работы, алгоритм должен определить лица и их черты и *пометить их* – поиск осуществляется в *активной области* изображения *прямоугольными признаками*, с помощью которых и описывается найденное лицо и его черты:
  $\mathbf{rectangle}_i = \mathbf{\{x,y,w,h,a\}}$,

где x, y – координаты центра i-го прямоугольника, w – ширина, h – высота, a – угол наклона прямоугольника к вертикальной оси изображения.

Интегральное представление изображений

- вейвлет-преобразования
- SURF
- SIFT

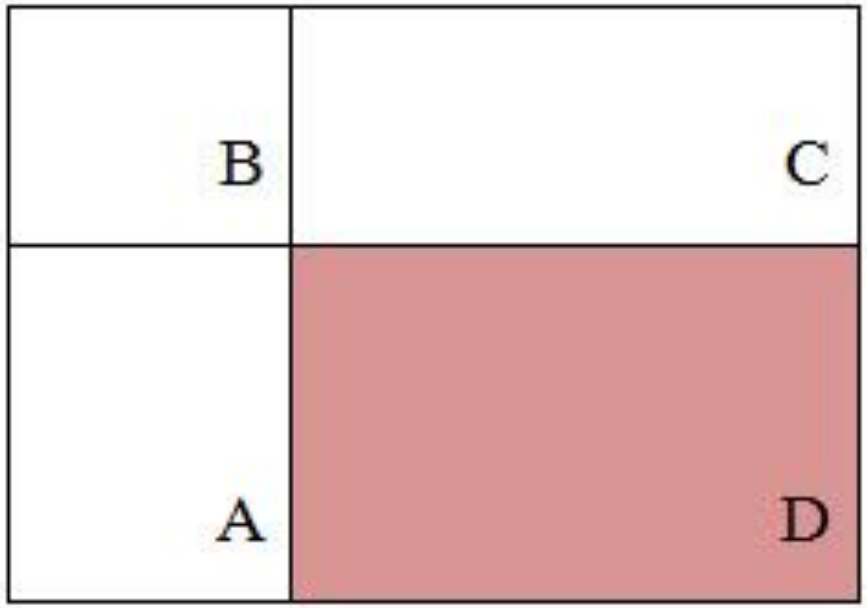
- рассчитывать ***суммарную яркость*** произвольного прямоугольника на изображении, причем какой бы прямоугольник не был, время расчета неизменно.
  Интегральное представление изображения – это ***матрица***, *совпадающая по размерам с исходным изображением*. В каждом элементе ее хранится *сумма интенсивностей всех пикселей, находящихся* ***левее*** *и* ***выше*** *данного элемента*.

$$L(x,y) = \sum_{i=0,j=0}^{i \le x, j \le y} * I(i,j).$$

- где I(i,j) — яркость пикселя исходного изображения. Каждый элемент матрицы L[x,y] представляет собой сумму пикселей в прямоугольнике от (0,0) до (x,y), т.е. значение каждого пикселя (x,y) равно сумме значений всех пикселов левее и выше данного пикселя (x,y). Расчет матрицы занимает линейное время, пропорциональное числу пикселей в изображении, поэтому интегральное изображение просчитывается за один проход.

L(x,y) = I(x,y) – L(x-1,y-1) + L(x,y-1) + L(x-1,y)

- S(ABCD) = L(A) + L(C) — L(B) — L(D)



Пример расчета

30	24	5	20	8	52
17	2	152	0	77	33
5	18	59	89	0	17
34	15	90	104	20	3
9	13	22	44	55	51
72	201	185	104	35	21

A

30	24
17	2

s = 30 + 24 + 17 +2 = 73

B

30	24	5	20
17	2	152	0

s = 73 + 5 + 20 + 152 + 0 = 250

D

30	24
17	2
5	18
34	15

s = 73 + 5 + 18 + 34 + 15 = 145

C

30	24	5	20
17	2	152	0
5	18	59	89
34	15	90	104

s = 250 + 72 + 59 + 89 + 90 + 104 = 664

искомый объект

59	89
90	104

s = 59 + 89 + 90 + 104 = 342

A + C - B -D = 73 + 664 - 250 - 145 = 342

Признаки Хаара

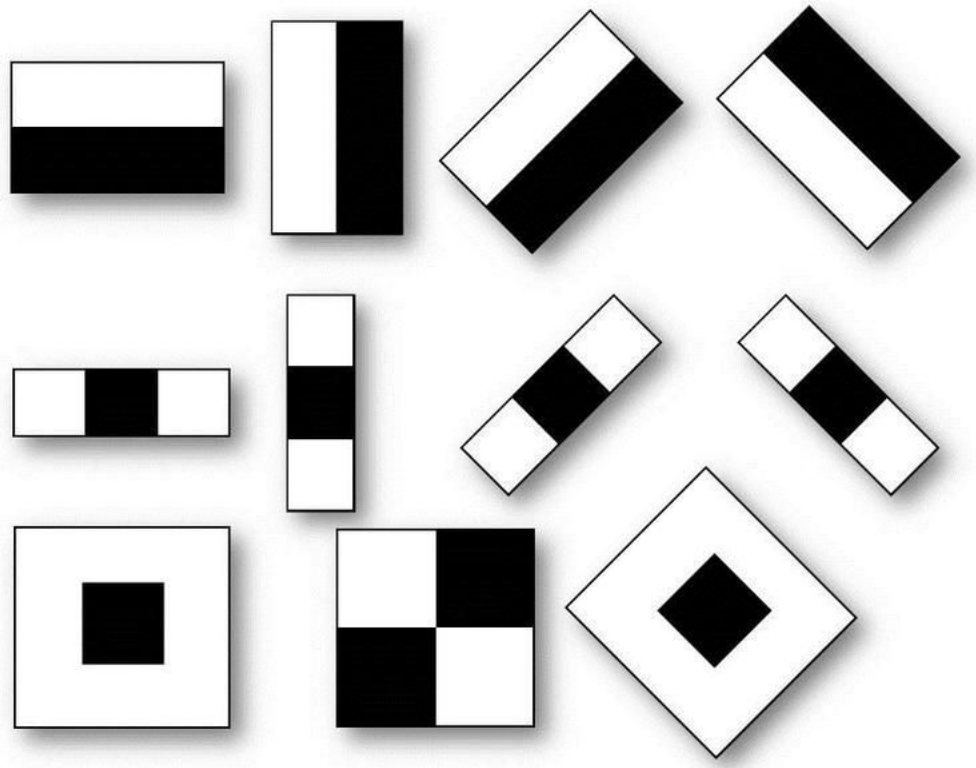
- Признак— отображение$f: X => D_f$, где D_f— множество допустимых значений признака.
- Если заданы признаки$f_1,\ldots,f_n$, то вектор признаков$x = (f_1(x),\ldots,f_n(x))$, называется признаковым описанием объекта $x \in X$.
- Признаковые описания допустимо отождествлять с самими объектами. При этом множество$X = D_{f1} * \ldots * D_{fn}$называют признаковым пространством.

типы в зависимости

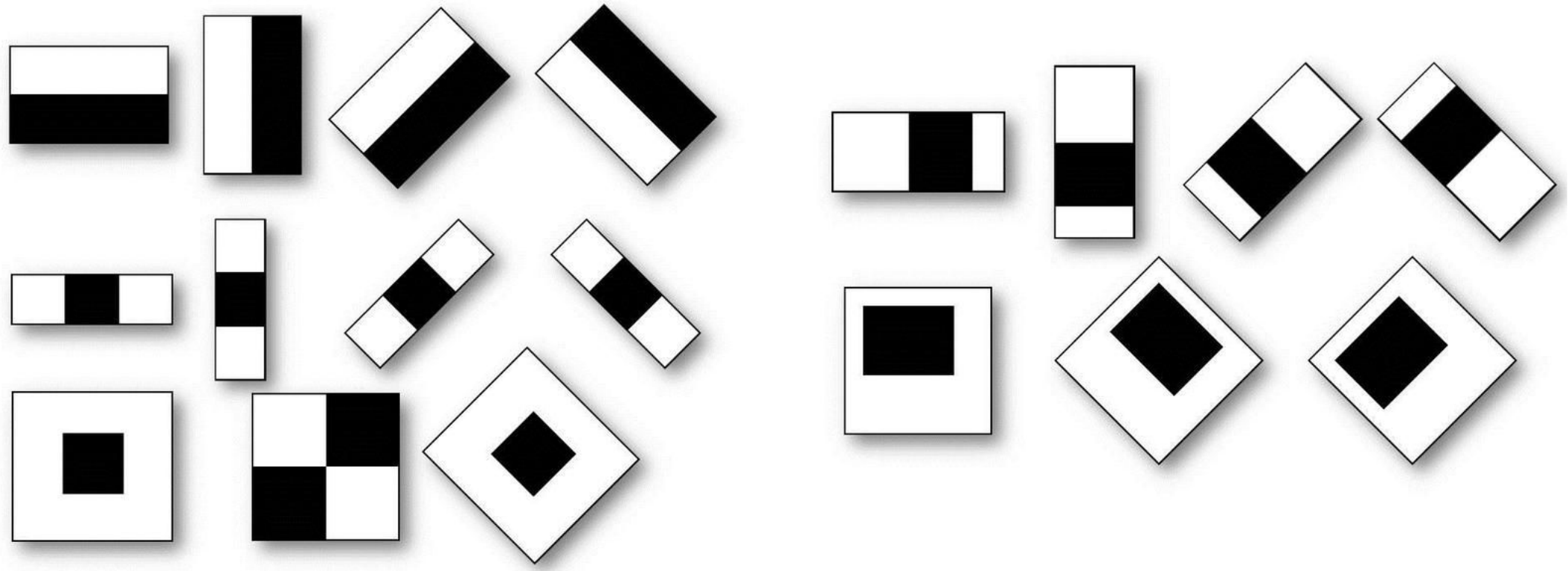
- Признаки делятся на следующие типы в зависимости от множества D_f:
- бинарный признак, $D_f = \{0,1\}$;
- номинальный признак: D_f— конечное множество;
- порядковый признак: D_f— конечное упорядоченное множество;
- количественный признак: D_f— множество действительных чисел.

В стандартном методе Виолы – Джонса

Вычисляемым значением такого признака будет
F = X-Y,
где X – сумма значений яркостей точек закрываемых *светлой частью признака*, а Y – сумма значений яркостей точек закрываемых *темной частью признака*.

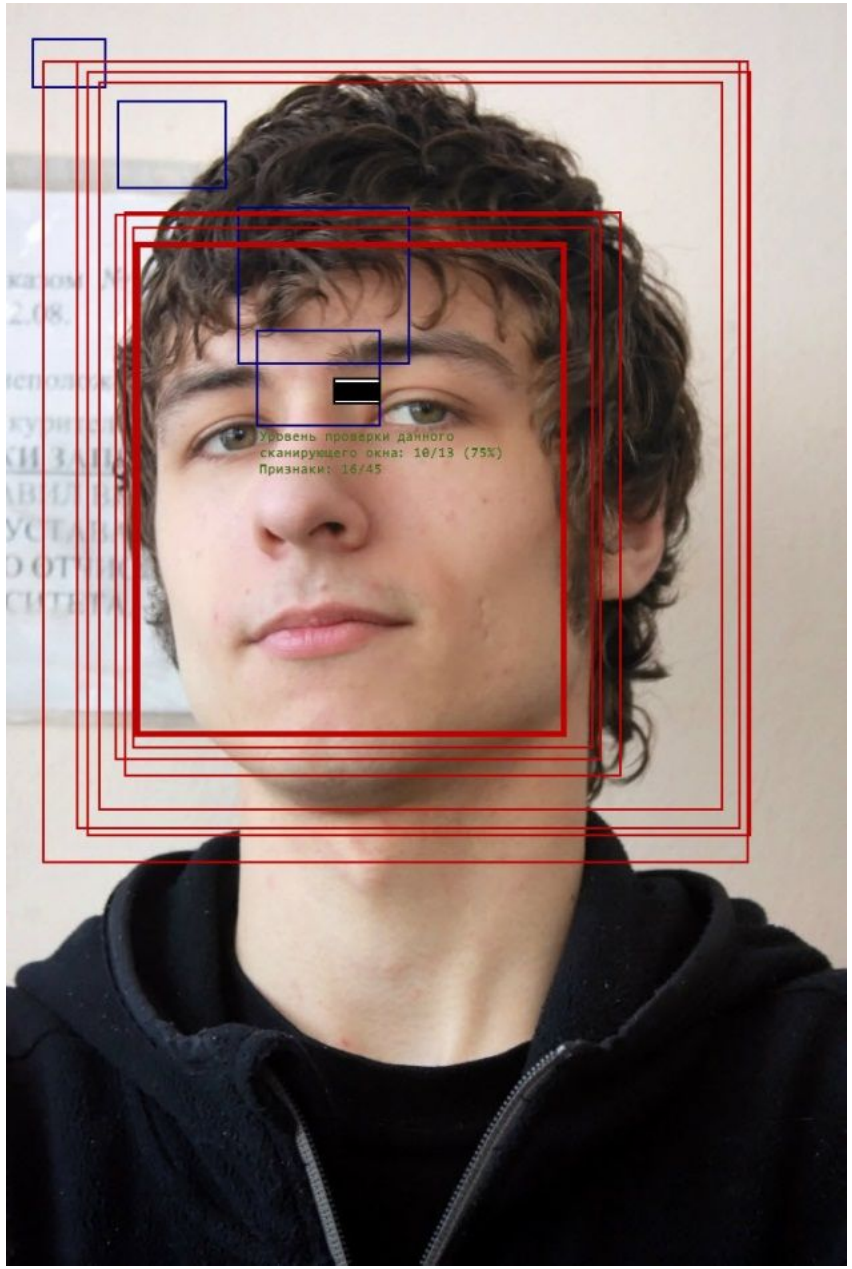


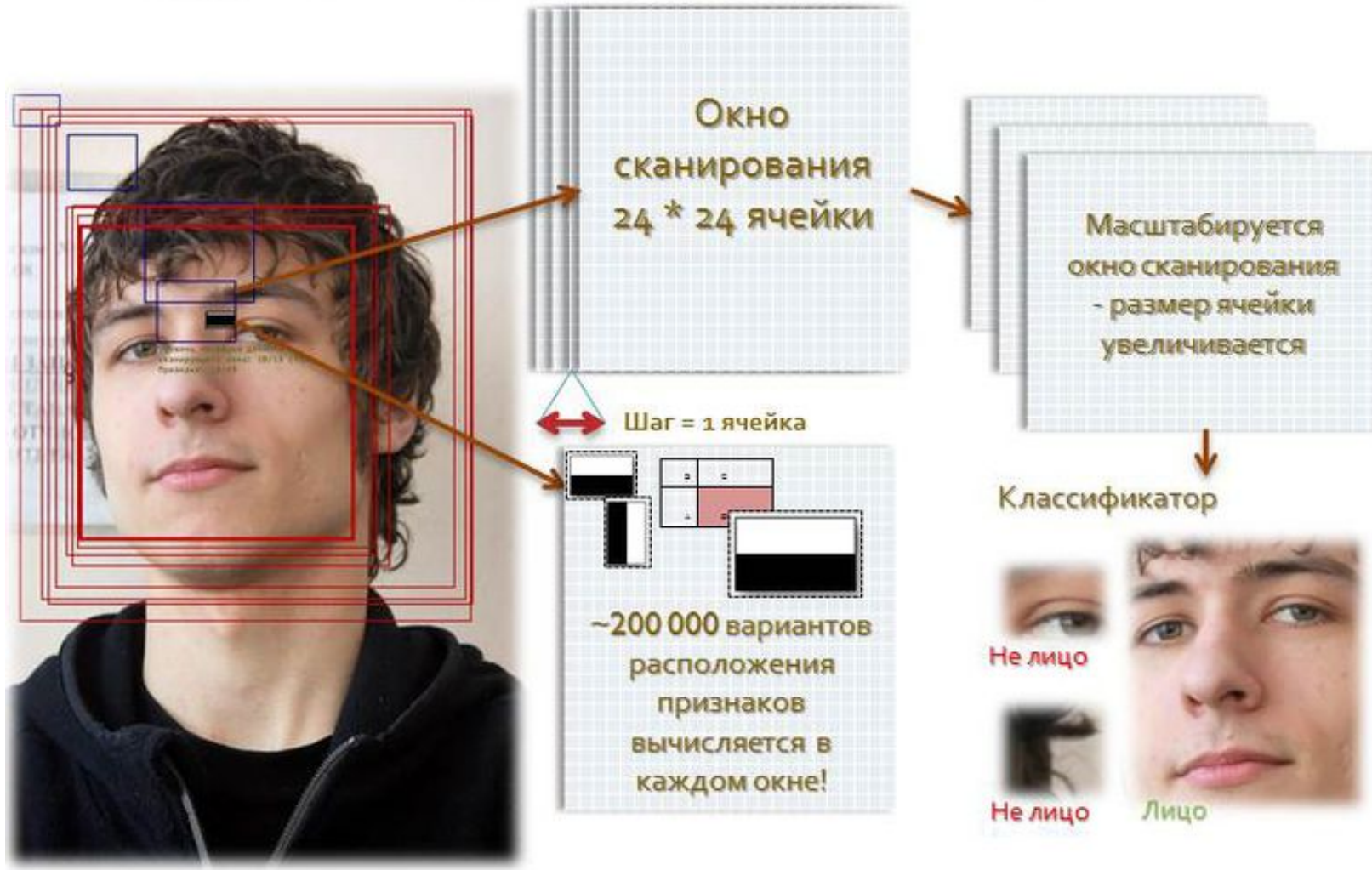
В расширенном методе Виолы – Джонса



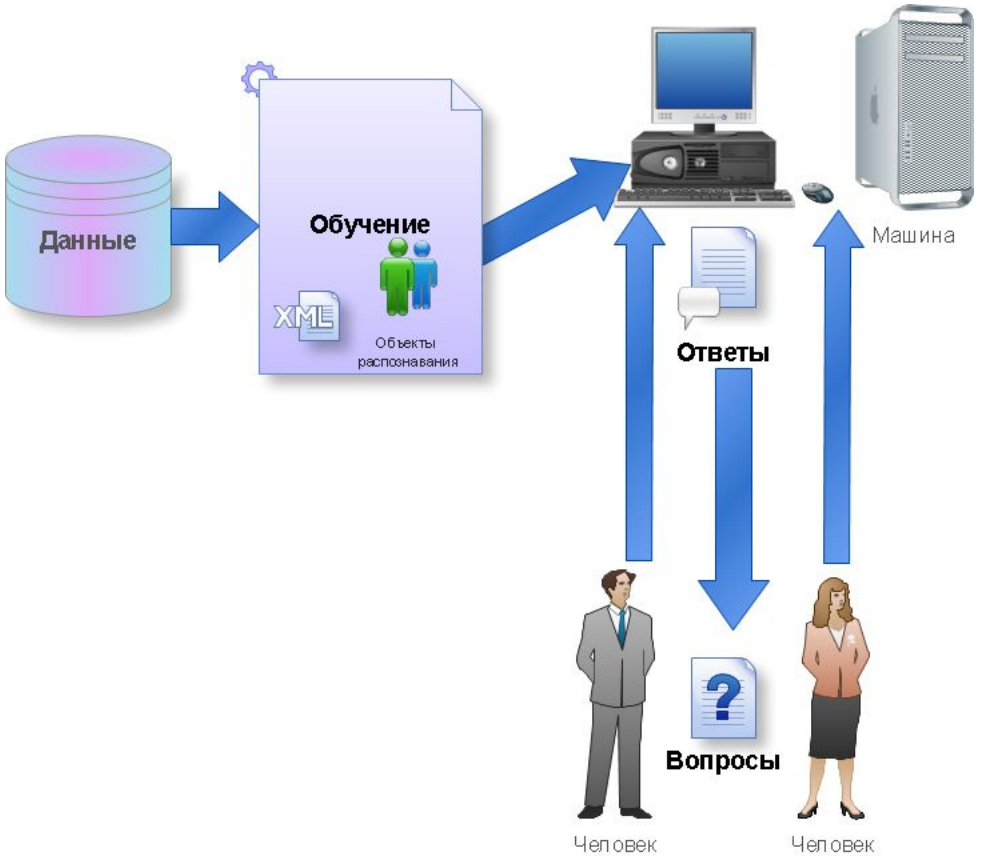
Сканирование окна

- есть исследуемое изображение, выбрано окно сканирования, выбраны используемые признаки;
- далее окно сканирования начинает последовательно двигаться по изображению с шагом в 1 ячейку окна (например, 24*24 ячейки);
- при сканировании изображения в каждом окне вычисляется приблизительно 200 000 вариантов расположения признаков, за счет изменения масштаба признаков и их положения в окне сканирования;
- сканирование производится последовательно для различных масштабов;
- масштабируется не само изображение, а сканирующее окно (изменяется размер ячейки);
- все найденные признаки попадают к классификатору, который «выносит вердикт».




Окно сканирования 24 * 24 ячейки
Масштабируется окно сканирования - размер ячейки увеличивается
Шаг = 1 ячейка
~200 000 вариантов расположения признаков вычисляется в каждом окне!
Классификатор
Не лицо
Не лицо
Лицо

Используемая в алгоритме модель машинного обучения



«Машинное обучение — это наука, изучающая компьютерные алгоритмы, автоматически улучшающиеся во время работы» (Michel, 1996)

Обучение машины — это процесс получения модулем новых знаний.

Обучение классификатора в методе Виолы-Джонса

- **Классифицировать объект** — значит, указать номер (или наименование класса), к которому относится данный объект.
  Классификация объекта — номер или наименование класса, выдаваемые алгоритмом классификации в результате его применения к данному конкретному объекту.
  Классификатор(classifier) — в задачах классификации это аппроксимирующая функция, выносящая решение, к какому именно классу данный объект принадлежит.
  Обучающая выборка – конечное число данных.

Постановка классификации

- Есть X – множество, в котором хранится описание объектов,
- Y – конечное множество номеров, принадлежащих классам.
- зависимость – отображение Y*: X => Y.
- Обучающая выборка $X_m = \{(x_1,y_1), \ldots, (x_m,y_m)\}$.
- Конструируется функция f от вектора признаков X, которая выдает ответ для любого возможного наблюдения X и способна классифицировать объект $x \in X$.

Бустинг и разработка AdaBoost

- Бустинг — комплекс методов, способствующих повышению точности аналитических моделей.
- *Эффективная модель, допускающая мало ошибок классификации*, называется *«сильной»*.
- *«Слабая» не позволяет надежно* разделять классы или давать точные предсказания, делает в работе большое количество ошибок. Поэтому **бустинг** означает дословно **«усиление» «слабых» моделей** – это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.

Идея бустинга

- *Роберт Шапир (Schapire)* в конце 90-х годов
- построение *цепочки (ансамбля) классификаторов*, который называется **каскадом**, каждый из которых (кроме первого) *обучается на ошибках предыдущего*.
- Boost1 использовал каскад из 3-х моделей,
- первая из которых обучалась на всем наборе данных,
- вторая – на выборке примеров, в половине из которых первая дала правильные ответы,
- третья — на примерах, где «ответы» первых двух разошлись.
- имеет место последовательная обработка примеров каскадом классификаторов,
- задача для каждого последующего становится труднее.
- Результат определяется путем голосования: пример относится к тому классу, который выдан большинством моделей каскада.

Математическое объяснение

- Наряду с множествами
- X и Y
- вводится вспомогательное множество R, называемое *пространством оценок*.
- Рассматриваются алгоритмы, имеющие вид суперпозиции
- a(x) = C(b(x)),
- где функция b: X → R называется *алгоритмическим оператором,*
- функция C: R → Y –*решающим правилом*.

Структура алгоритмов классификации

- вычисляются оценки принадлежности объекта классам,
- решающее правило переводит эти оценки в номер класса.
- Значение оценки, как правило, характеризует степень уверенности классификации.

Алгоритмическая композиция

- алгоритм $a: X \to Y$ вида
  $a(x) = C(F(b_1(x), \ldots, b_T(x)), x \in X$,
- составленный из алгоритмических операторов $b_t : X \to R$, $t=1,\ldots, T$,
- корректирующей операции $F: R^T \to R$
- решающего правила $C: R \to Y$.
  Базовыми алгоритмами обозначаются функции $a_t(x) = C(b_t(x))$, а при фиксированном решающем правиле C — и сами операторы $b_t(x)$.
- Суперпозиции вида $F(b_1,\ldots, b_T)$ являются отображениями из X в R, то есть, опять же, алгоритмическими операторами.

совместное применение нескольких критериев

- построено заданное количество базовых алгоритмов T;
- достигнута заданная точность на обучающей выборке;
- достигнутую точность на контрольной выборке не удаётся улучшить на протяжении последних нескольких шагов при определенном параметре алгоритма.

AdaBoost (*adaptive boosting* – адаптированное улучшение)

- Йоав Фройнд (Freund) и Роберт Шапир (Schapire) в 1999,
- может использовать произвольное число классификаторов
- производить обучение на одном наборе примеров, поочередно применяя их на различных шагах.

- задача классификации на два класса, $Y = \{-1,+1\}$. К примеру, базовые алгоритмы также возвращают только два ответа −1 и +1, и решающее правило фиксировано: $C(b) = \text{sign}(b)$. Искомая алгоритмическая композиция имеет вид:

$$a(x) = C(F(b_1(x), \ldots, b_T(x)) = \text{sign}(\sum_{t=1}^{T} a_t b_t(x)), \quad x \in X$$

- Функционал качества композиции Q_t определяется как число ошибок, допускаемых ею на обучающей выборке:

$$Q(b, W^l) = Q_T = \sum_{t=1}^{l} [\, y_i \sum_{t=1}^{T} a_t b_t(x_i)] < 0,$$

- Для решения задачи AdaBoosting’а нужна экспоненциальная аппроксимация пороговой функции потерь [z<0], причем экспонента $E^z = e^{-z}$

- Дано:
  $Y = \{-1,+1\}$,
- $b_1(x), \ldots, b_T(x)$ возвращают -1 и $+1$,
- X^l – обучающая выборка.
-

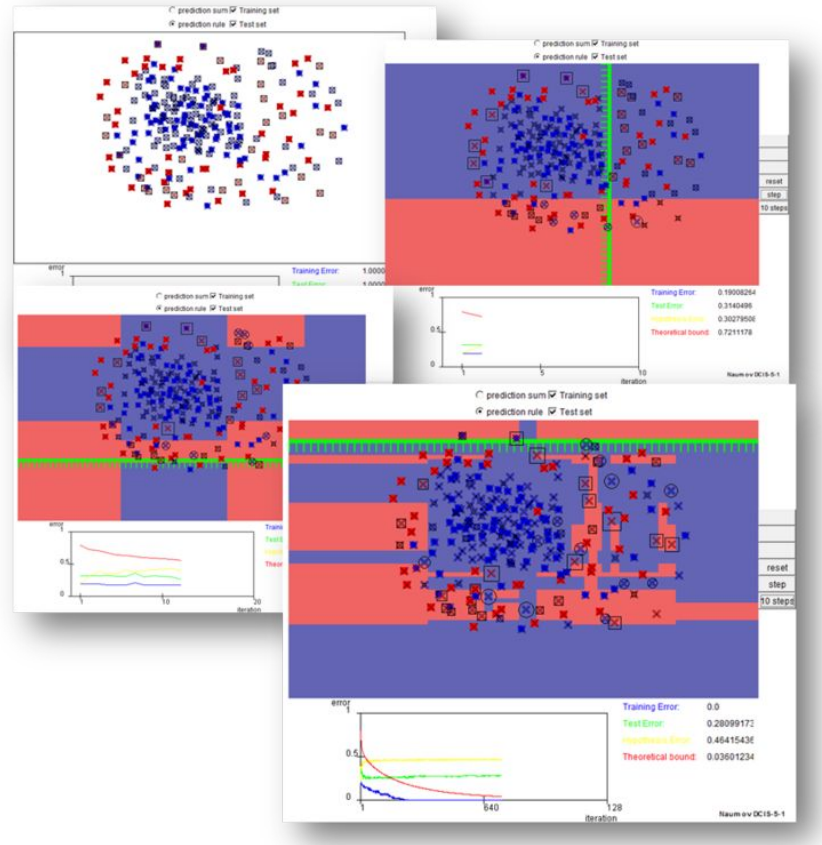
- Решение:
  1. Инициализация весов объектов:
  $w_i := 1/\ell$, $i = 1, \ldots, \ell$; (1.9)
  для всех $t = 1, \ldots, T$, пока не выполнится критерий останова:
  2 а. $b_t := \arg\min_b Q(b; W^l)$;
  $\alpha_t := 1/2 \ln 1 - \frac{1 - Q(b_t; W^l)}{Q(b_t; W^l)}$;
- 2 б.
- 3. Пересчёт весов объектов. Правило мультипликативного пересчёта весов.
- Вес объекта увеличивается в $e^{\alpha t}$ раз, когда b_t допускает на нём ошибку, и во столько же раз уменьшается, когда b_t правильно классифицирует x_i.

- непосредственно перед настройкой базового алгоритма наибольший вес накапливается у тех объектов, которые чаще оказывались трудными для предыдущих алгоритмов:

$$w_i := w_i \exp(-\alpha_t y_i b_t(x_i)),\ i = 1, \ldots, \ell;$$

- 4. Нормировка весов объектов

$$w_0 := \sum_{j=1}^{l} w_j;\ w_i := w_i / w_0,\ i = 1, \ldots, \ell.$$


prediction sum
Training set
prediction rule
Test set
reset
step
10 steps
error
iteration
Training Error:
Test Error:
Theoretical bound:
0.0
0.28099173
0.46415436
0.03601234
Naum-ov DCIS-S-1

Плюсы AdaBoost

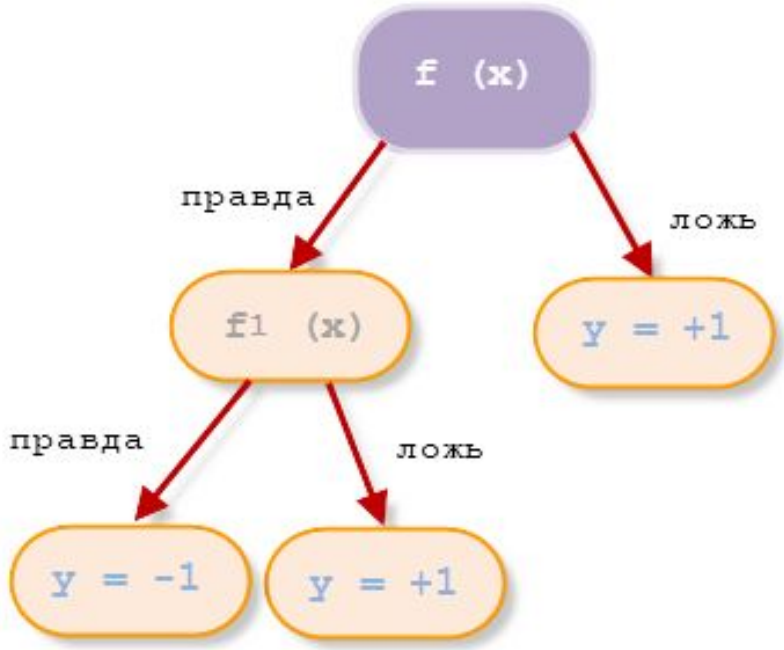
- хорошая обобщающая способность. В реальных задачах практически всегда строятся композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться по мере увеличения числа базовых алгоритмов;
- простота реализации;
- собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов;
- возможность идентифицировать объекты, являющиеся шумовыми выбросами. Это наиболее «трудные» объекты x_i, для которых в процессе наращивания композиции веса w_i принимают наибольшие значения.

Минусы AdaBoost:

- Бывает переобучение при наличии значительного уровня шума в данных. Экспоненциальная функция потерь слишком сильно увеличивает веса «наиболее трудных» объектов, на которых ошибаются многие базовые алгоритмы. Однако именно эти объекты чаще всего оказываются шумовыми выбросами. В результате AdaBoost начинает настраиваться на шум, что ведёт к переобучению. Проблема решается путём удаления выбросов или применения менее «агрессивных» функций потерь. В частности, применяется алгоритм GentleBoost;
- AdaBoost требует достаточно длинных обучающих выборок. Другие методы линейной коррекции, в частности, бэггинг, способны строить алгоритмы сопоставимого качества по меньшим выборкам данных;
- Бывает построение неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново.
- Бустинг может приводить к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных временных затрат на вычисление классификаций.

Принципы решающего дерева в алгоритме

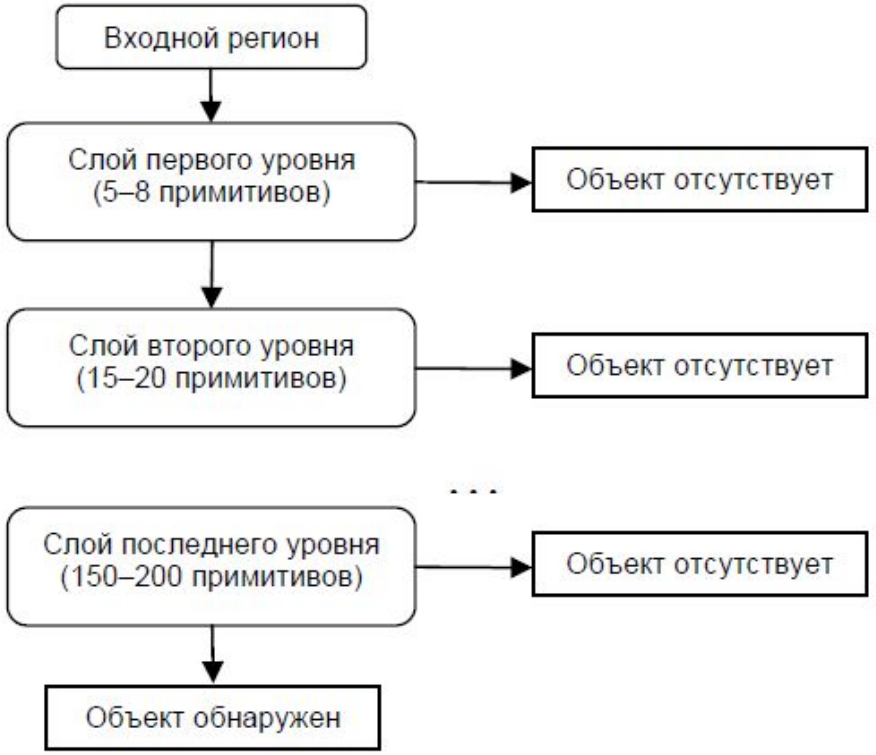
```
function Node = Обучение_Вершины( {(x,y)} ) {
if {y} одинаковые
return Создать_Лист(y);
test = Выбрать_лучшее_разбиение( {(x,y)} );
{(x0,y0)} = {(x,y) | test(x) = 0};
{(x1,y1)} = {(x,y) | test(x) = 1};
LeftChild = Обучение_Вершины( {(x0,y0)} );
RightChild = Обучение_Вершины( {(x1,y1)} );
return Создать_Вершину(test, LeftChild, RightChild);
}
//Обучение дерева
function main() {
{(X,Y)} = Прочитать_Обучающие_Данные();
TreeRoot = Обучение_Вершины( {(X,Y)} );
}
```



Каскадная модель алгоритма

- Алгоритм бустинга для поиска лиц с моей точки зрения таков:
  1. Определение слабых классификаторов по прямоугольным признакам;
  2. Для каждого перемещения сканирующего окна вычисляется прямоугольный признак на каждом примере;
  3. Выбирается наиболее подходящий порог для каждого признака;
  4. Отбираются лучшие признаки и лучший подходящий порог;
  5. Перевзвешивается выборка.

Сложность обучения таких каскадов равна O(xyz), где применяется x этапов, y примеров и z признаков.
Далее, каскад применяется к изображению:
1. Работа с «простыми» классификаторами – при этом отбрасывается часть «отрицательных» окон;
2. Положительное значение первого классификатора запускает второй, более приспособленный и так далее;
3. Отрицательное значение классификатора на любом этапе приводит к немедленному переходу к следующему сканирующему окну, старое окно отбрасывается;
4. Цепочка классификаторов становится более сложной, поэтому ошибок становится намного меньше.



- Для тренировки такого каскада потребуются следующие действия:
  1. Задаются значения уровня ошибок для каждого этапа (предварительно их надо количественно просмотреть при применении к изображению из обучающего набора) – они называются *detection* и *false positive rates* – надо чтобы уровень detection был высок, а уровень false positive rates низок;
  2. Добавляются признаки до тех пор, пока параметры вычисляемого этапа не достигнут поставленного уровня, тут возможны такие вспомогательные этапы, как:
  а. Тестирование дополнительного маленького тренировочного набора;
  б. Порог AdaBoost умышленно понижается с целью найти больше объектов, но в связи с этим возможно большее число неточных определений объектов;
  3. Если false positive rates остается высоким, то добавляется следующий этап или слой;
  4. Ложные обнаружения в текущем этапе используются как отрицательные уже на следующем слое или этапе.

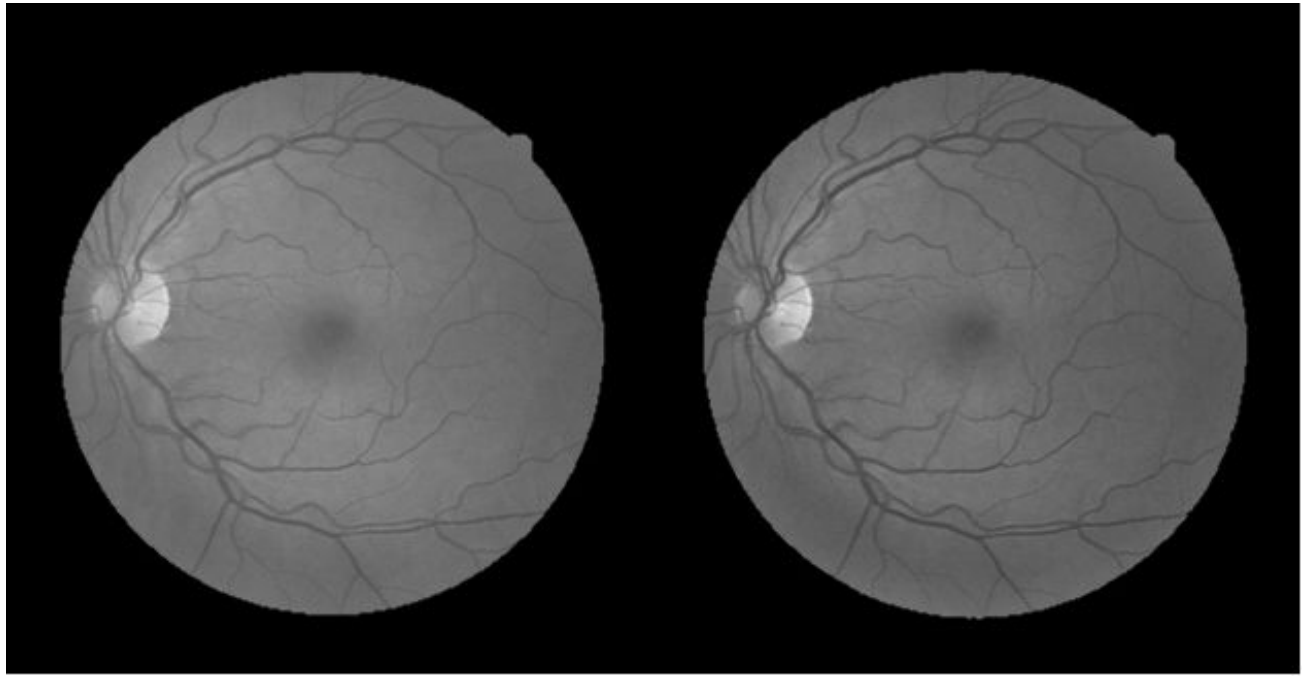
- В *более формальном виде* алгоритм тренировки каскада:
  a) Пользователь задает значения f (максимально допустимый уровень ложных срабатываний на слой) и d (минимально допустимый уровень обнаружений на слой)
  b) Пользователь задает целевой общий уровень ложных срабатываний F_{target}
  c) P – набор положительных примеров
  d) N – набор отрицательных примеров
  e) $F_0 = 1{,}0$; $D_0 = 1{,}0$; $i = 0$
  f) while ($F_i > F_{target}$)
  $i = i+1$; $n_i = 0$; $F_i = F_{i-1}$
  while ($F_i = f * F_{i-1}$)
  $n_i = n_i + 1$
  AdaBoost(P, N, n_i)
  Оценить полученный каскад на тестовом наборе для определения F_i и D_i ;
  Уменьшать порог для i-того классификатора, пока текущий каскад будет иметь уровень обнаружения по крайней мере $d*D_i$-1 (это же касается F_i) ;
  g) N = Ø;
  h) Если $F_i > F_{target}$, то оценить текущий каскад на наборе изображений, не содержащих лиц, и добавить все неправильно классифицированные в N.

- 1. P. Viola and M.J. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features», proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001
  2. P. Viola and M.J. Jones, «Robust real-time face detection», International Journal of Computer Vision, vol. 57, no. 2, 2004., pp.137–154
  3. Р.Гонсалес, Р.Вудс, «Цифровая обработка изображений», ISBN 5-94836-028-8, изд-во: Техносфера, Москва, 2005. – 1072 с.
  4. Местецкий Л. М., «Математические методы распознавания образов», МГУ, ВМиК, Москва, 2002–2004., с. 42 – 44
  5. Jan ˇSochman, Ji ˇr´ı Matas, «AdaBoost», Center for Machine Perception, Czech Technical University, Prague, 2010
  6. Yoav Freund, Robert E. Schapire, «A Short Introduction to Boosting», Shannon Laboratory, USA, 1999., pp. 771-780

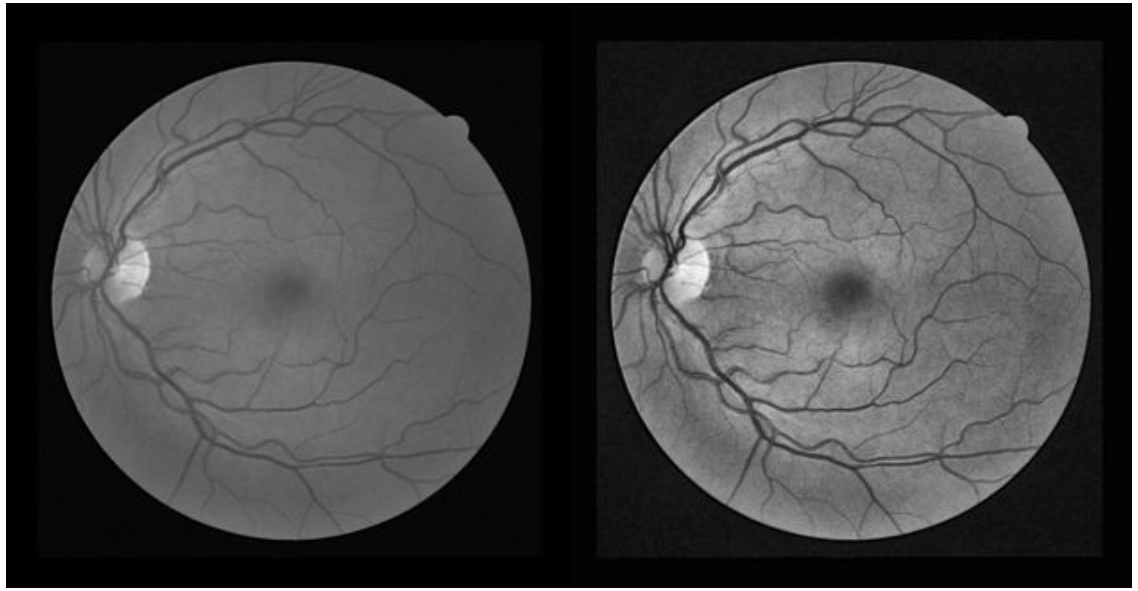
- https://habrahabr.ru/post/133909/

Улучшение контрастности между фоном и кровеносными сосудами

G

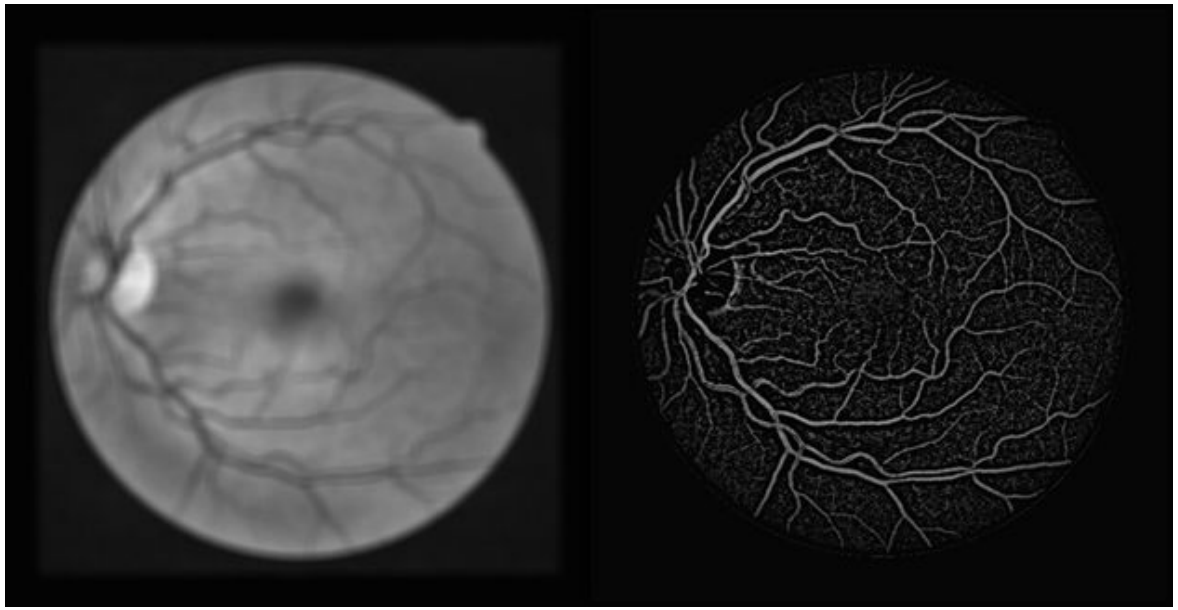


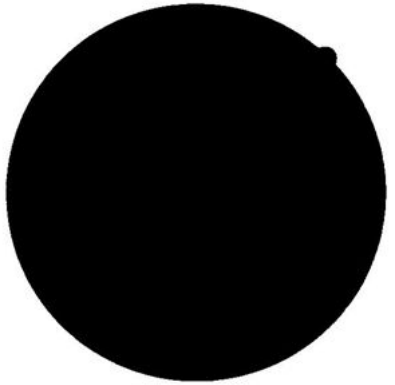
Выбор цветового
канала



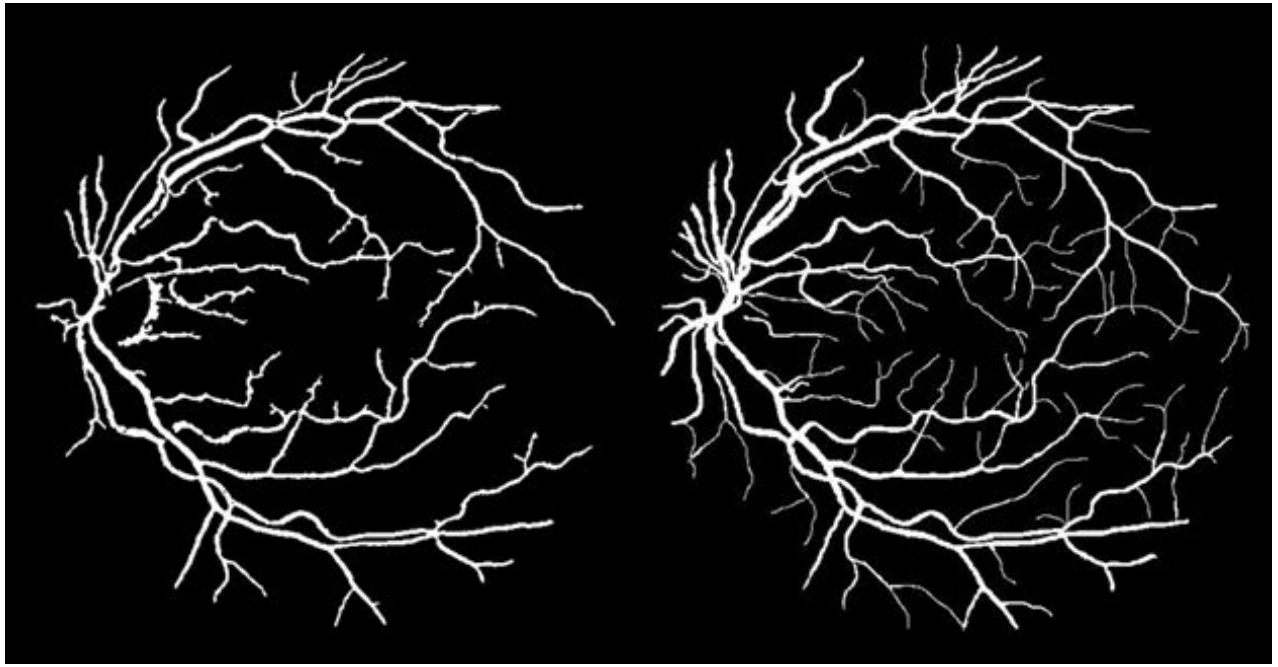
контрастно-ограниченное адаптивное выравнивание гистограммы
(contrast limited adaptive histogram equalization – clahe)

Удаление фона при помощи average фильтра





Маска сетчатки



автоматическое пороговое преобразование методом Otsu, медианный фильтр и фильтр по длине

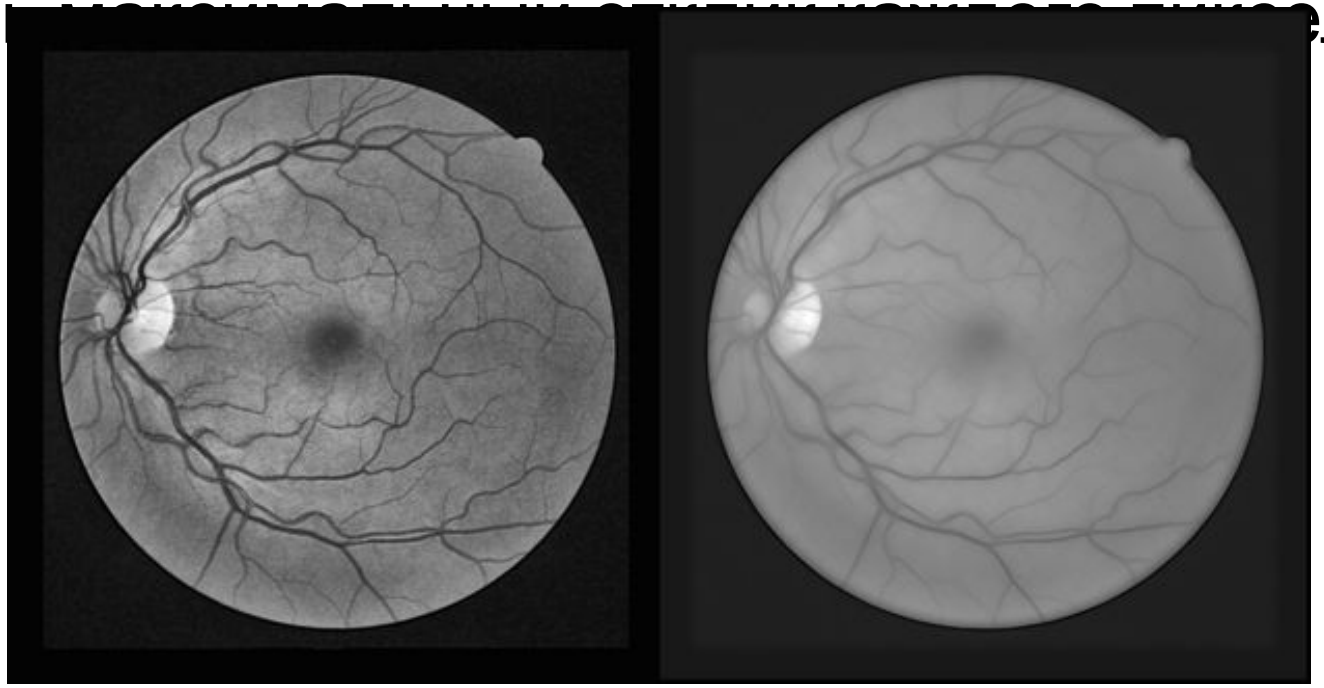
Фильтр Габора

- Способен выделять прямые линии определённого размера и под определённым углом

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x}{\lambda} + \psi\right),$$

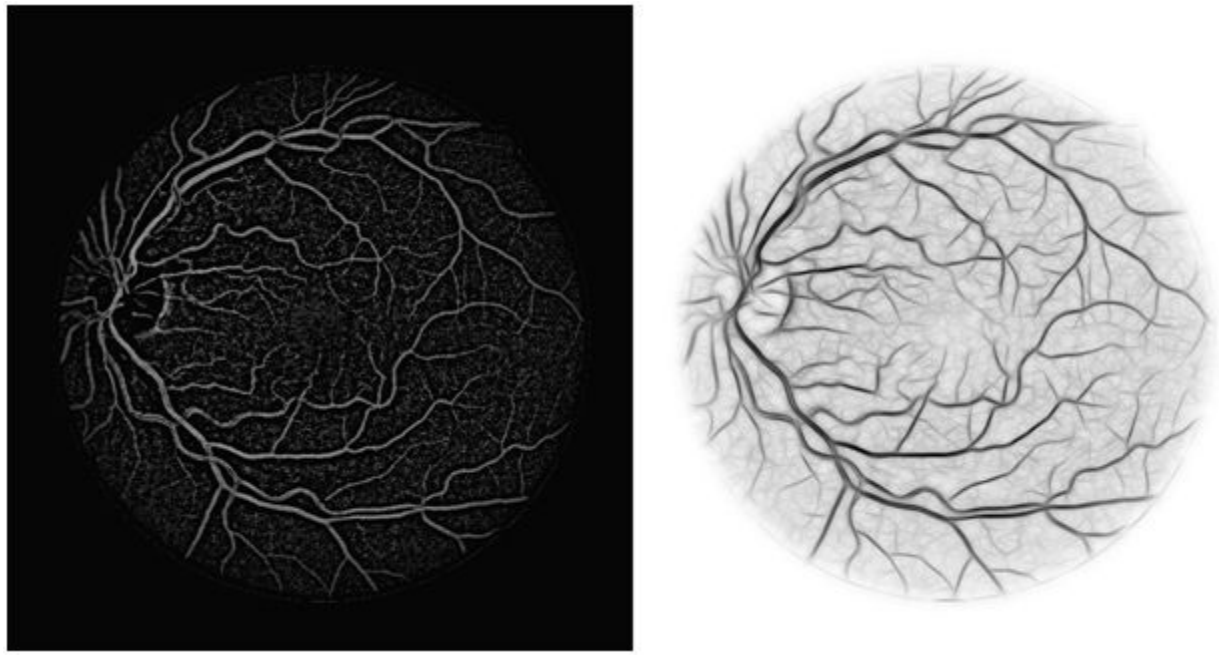
- $x' = x\cos\theta + y\sin\theta$;
- $y' = -x\sin\theta + y\cos\theta$;
- x, y – координаты ядра в заранее заданных пределах;
- λ – период ядра в пикселях;
- θ – наклон ядра;
- σ – дисперсия Гауссиана;
- ψ – смещение фазы ядра;
- γ – сжатие Гауссиана.

- применить фильтр Габора с различными углами наклона ядра
- рассчитать максимальный отклик каждого пикселя на серию фильтров



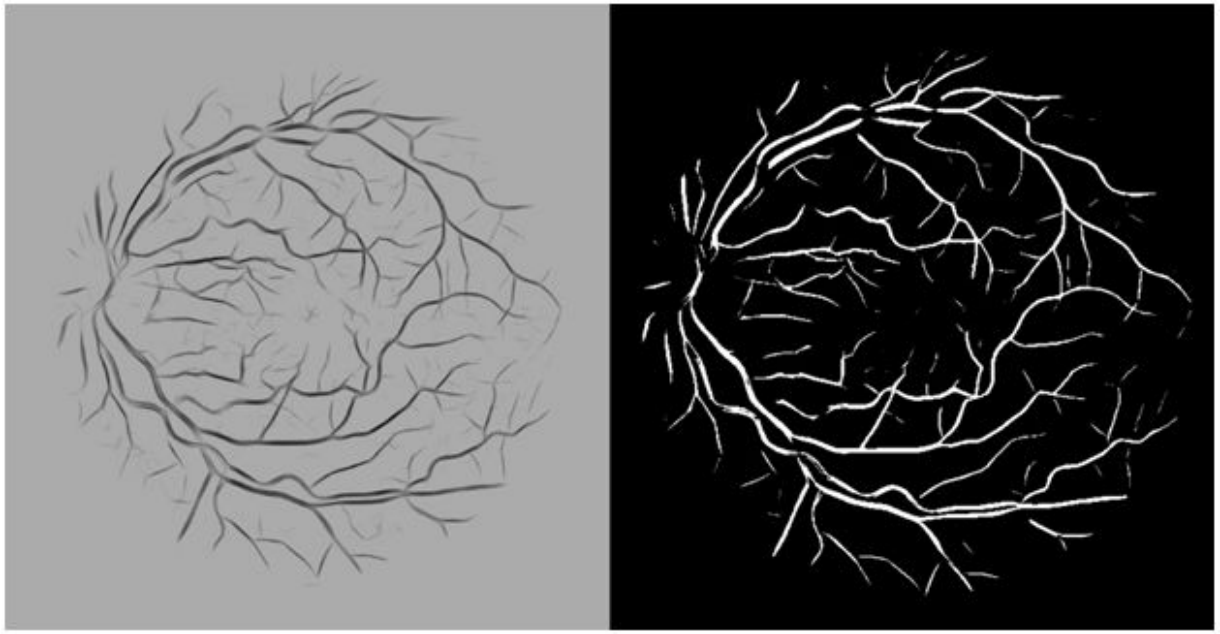
слева – исходное изображение после clahe, справа – результат применения серии габоровских фильтров

Удаление фона



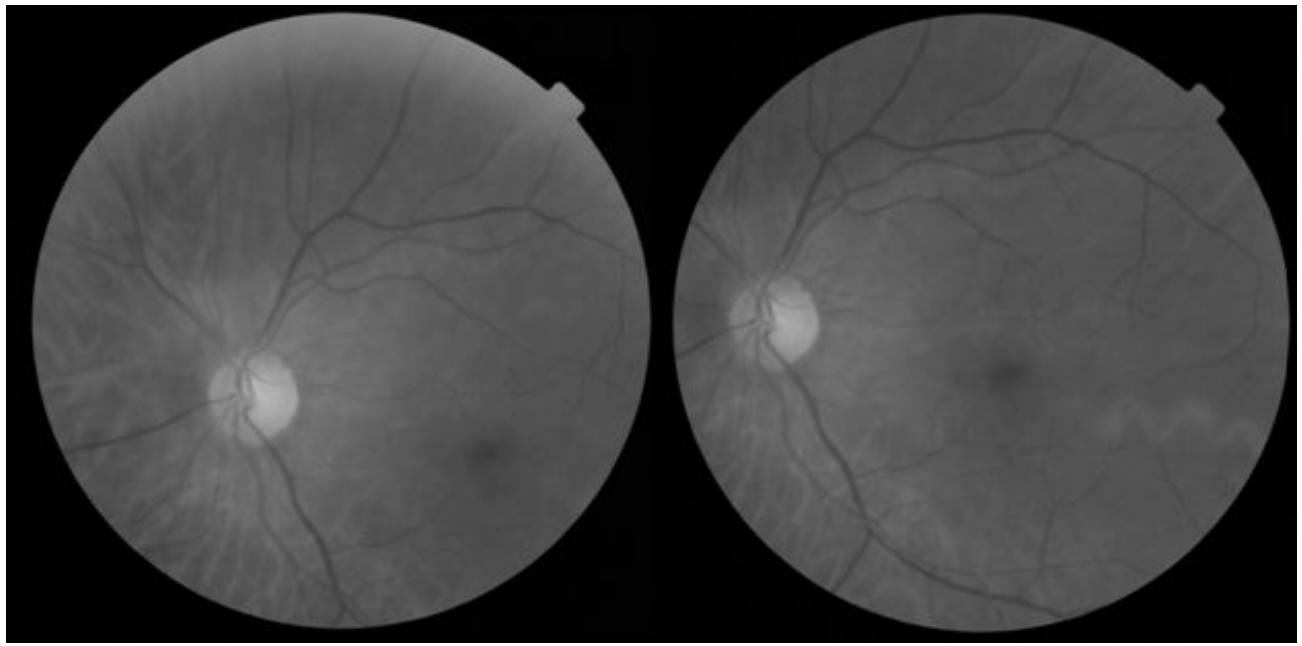
слева – исходное изображение, полученное при помощи алгоритма background exclusion, справа – результат применения серии габоровских фильтров

Пороговое преобразование интенсивности изображения



слева – исходное изображение, полученное после перекрашивания пикселей в соответствии с параметром чувствительности, справа – результат метода Otsu

- Marwan D. Saleh, C. Eswaran, and Ahmed Mueen. An Automated Blood Vessel Segmentation Algorithm Using Histogram Equalization and Automatic Threshold Selection // Journal of Digital Imaging, Vol 24, No 4 (August), 2011, pp 564-572
- P. C. Siddalingaswamy, K. Gopalakrishna Prabhu. Automatic detection of multiple oriented blood vessels in retinal images // J. Biomedical Science and Engineering, 2010, 3, pp 101-107
- www.isi.uu.nl/Research/Databases/DRIVE
- www.ces.clemson.edu/~ahoover/stare



Результат движения головы и глаза при сканировании сетчатки

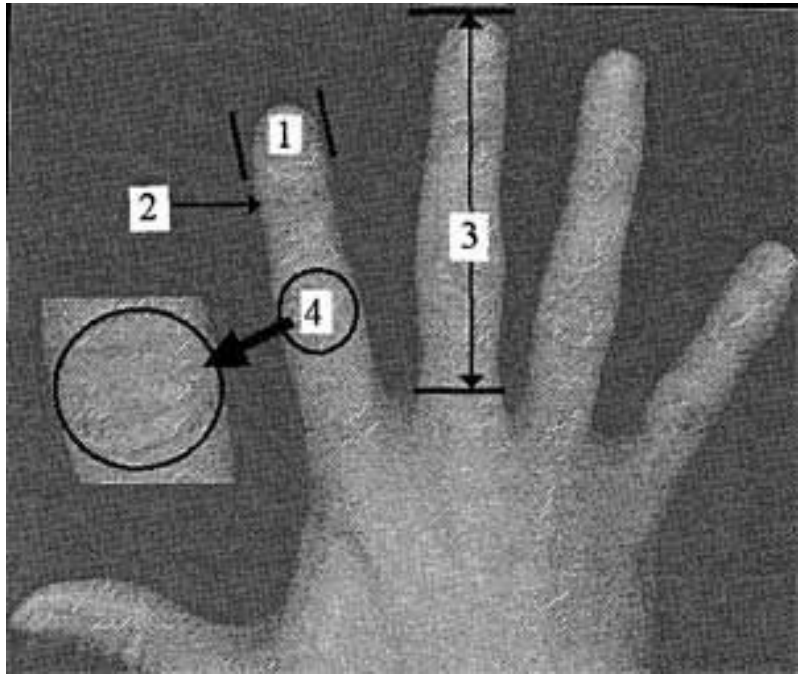
Алгоритм, основанный на методе фазовой корреляции

Алгоритм, использующий углы Харриса

Алгоритм, основанный на поиске точек разветвления

- Reddy B.S. and Chatterji B.N. An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration // IEEE Transactions on Image Processing. 1996. Vol. 5. No. 8. pp. 1266-1271.
- Human recognition based on retinal images and using new similarity function / A. Dehghani [et al.] // EURASIP Journal on Image and Video Processing. 2013.
- Hortas M.O. Automatic system for personal authentication using the retinal vessel tree as biometric pattern. PhD Thesis. Universidade da Coruña. La Coruña. 2009.
- VARIA database
- MESSIDOR database

Геометрия рук



Движения глаз

- фиксация глаза на определенной точке дисплея
- момент движения яблока при перемещении взгляда с одной точки на другую

Neurotechnology

- http://www.neurotechnology.com/

BIOMETRICS
Large-scale systems
Fingerprint SDKs
Face SDKs
Eye iris SDK
Voice SDK
Supported devices (152)
Deduplication service

COMPUTER VISION & A.I.
Video surveillance
Robotics kit
3D face tracking SDK
Object recognition SDKs
3D reconstruction SDK
Gaze tracking SDK

END USER PRODUCTS
SentiVeillance Server
Attendance management
Face hiding in videos
Gestural/voice control
Check My Age

CUSTOM PROJECTS

Company | Biometrics | Surveillance | Computer vision and A.I.

FINGERPRINT, FACE, IRIS, SPEAKER AND PALM PRINT RECOGNITION.
FROM SMALL APPS TO NATIONAL-SCALE SYSTEMS.

MEGAMATCHER SDK
Fingerprints, faces, irises, voiceprints and palmprints.
More info...

MEGAMATCHER ACCELERATOR
The fastest biometric engine in the world.
More info...

MEGAMATCHER ABIS
Integrated system for national-scale multi-biometric projects.
More info...

Subsets of MegaMatcher SDK for single biometric modalities:

VeriFinger SDK, VeriLook SDK, VeriEye SDK, VeriSpeak SDK.

Fingerprint identification for embedded platforms:

FingerCell SDK

Поведенческая биометрия

Биометрия по электрокардиограмме

Биометрия по почерку

Биометрия по походке

Биометрия по особенностям чтения

Биометрия по особенностям набора текста

Идентификация личности на основе данных о перемещениях (трекинга)

Отпечатки пальцев 3D