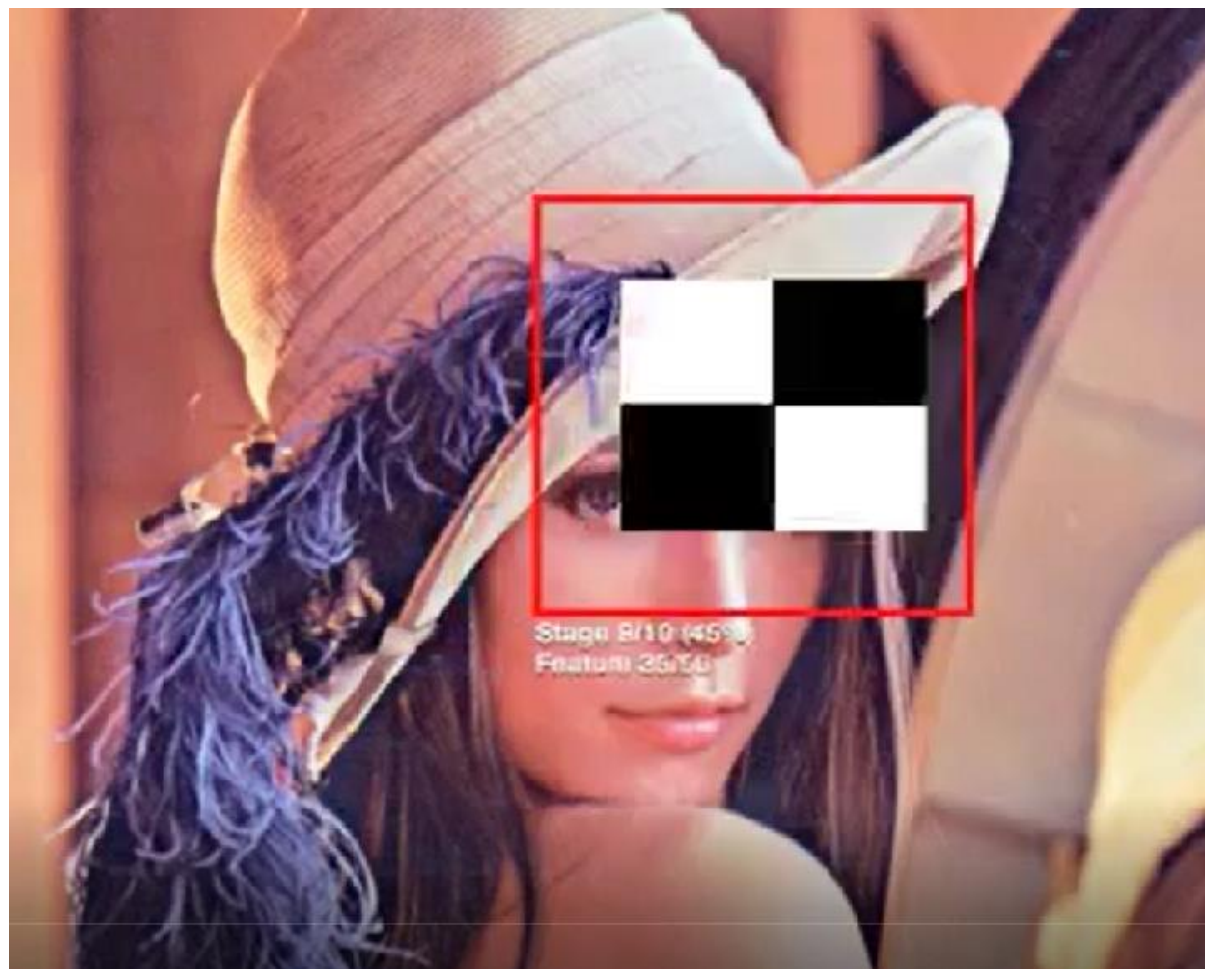


# Детектирование лиц



# Метод Виолы-Джонса (Viola-Jones)

---

**Метод Виолы—Джонса** (Viola–Jones object detection) — алгоритм, позволяющий обнаруживать объекты на изображениях в реальном времени.

- Разработан в 2001 году Полом Виолой и Майклом Джонсом.
- Используется в бюджетных цифровых фотоаппаратах.
- Является одним из лучших по соотношению показателей эффективности распознавания/скорость работы.
- Алгоритм находит лица с высокой точностью и низким количеством ложных срабатываний.
- Алгоритм хорошо работает и распознает черты лица под небольшим углом, примерно до 30 градусов.
- Реализован в составе библиотеки компьютерного зрения OpenCV.

# Метод Виолы-Джонса. Основные принципы.

---

Основные принципы, на которых основан метод, таковы:

- используются **изображения в интегральном представлении**, что позволяет вычислять быстро необходимые объекты;
- используются признаки Хаара, с помощью которых происходит поиск нужного объекта;
- 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.

# Интегральное представление изображения

---

**Интегральное представление изображения** — это матрица, размерность которой совпадает с размерностью исходного изображения.

Каждый элемент интегрального изображения содержит в себе сумму значений всех пикселей левее и выше данного пикселя  $(x,y)$ .

Original

5	2	3	4	1
1	5	4	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Integral

5	7	10	14	15
6	13	<b>20</b>	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

$$5 + 2 + 3 + 1 + 5 + 4 = 20$$

# Интегральное представление изображения

---

Интегральное представление позволяет **быстро вычислить сумму пикселей произвольного прямоугольника** с помощью четырех ссылок на массив.

Original

5	2	3	4	1
1	5	4	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

$$5 + 4 + 2 + 2 + 1 + 3 = 17$$

Integral

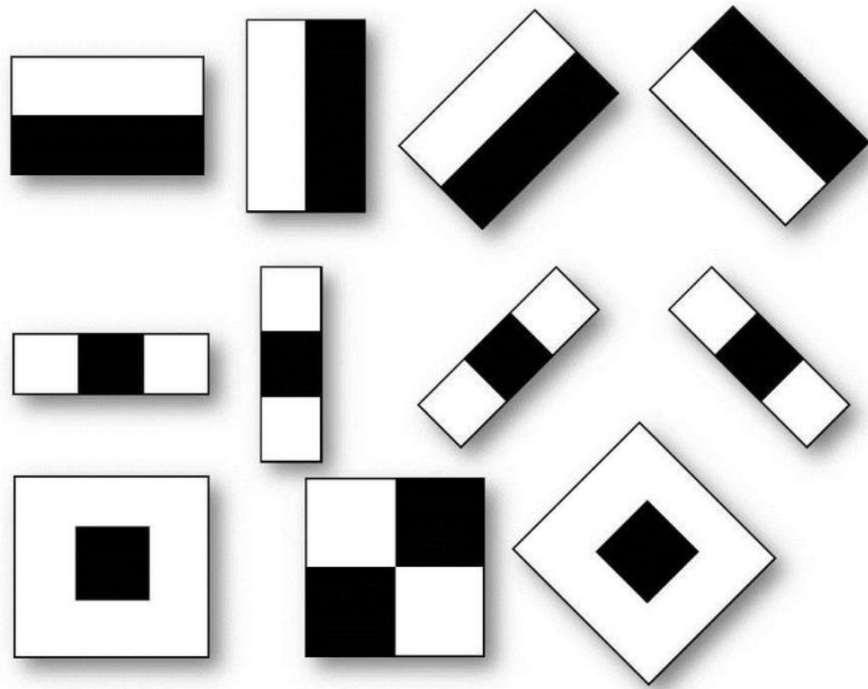
5	7	10	14	15
6	13	20	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

$$34 - 14 - 8 + 5 = 17$$

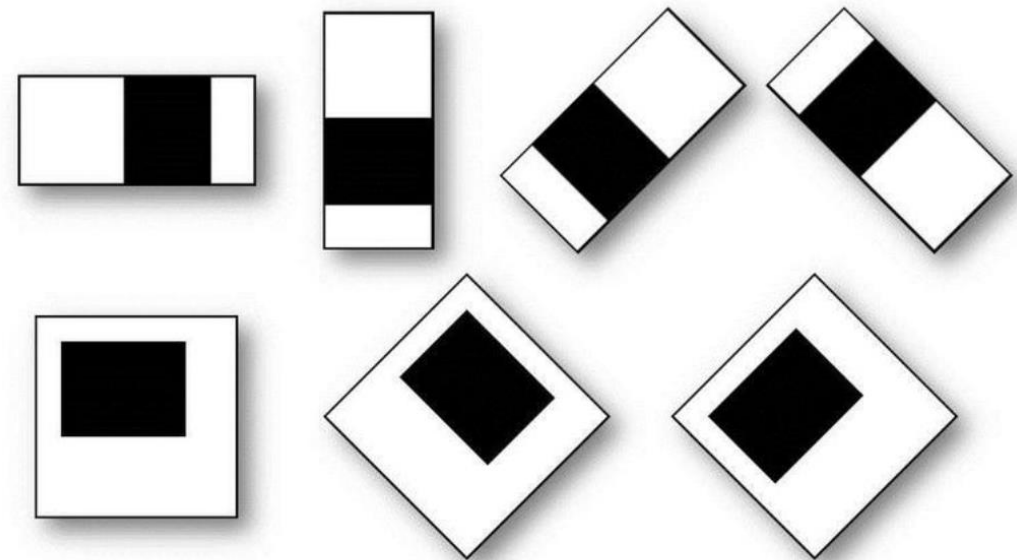
# Признаки Хаара

---

В методе Виолы-Джонса основу составляют примитивы Хаара, представляющие собой разбивку заданной прямоугольной области на наборы разнотипных прямоугольных подобластей:



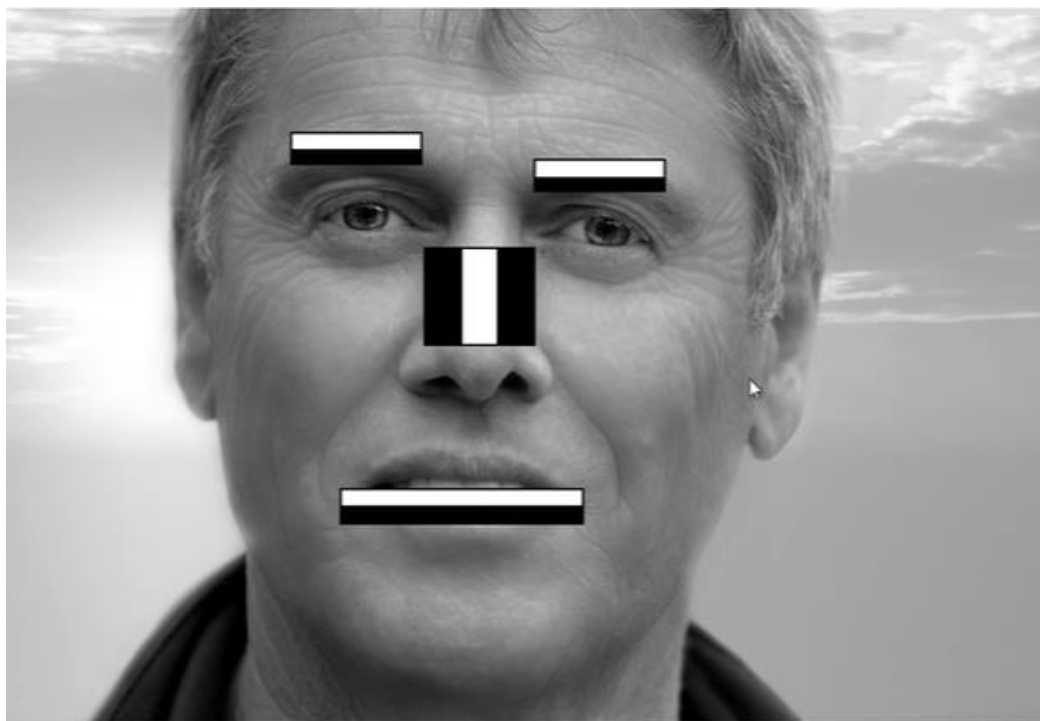
Стандартный метод



Расширенный метод

# Вычисление признаков Хаара

---

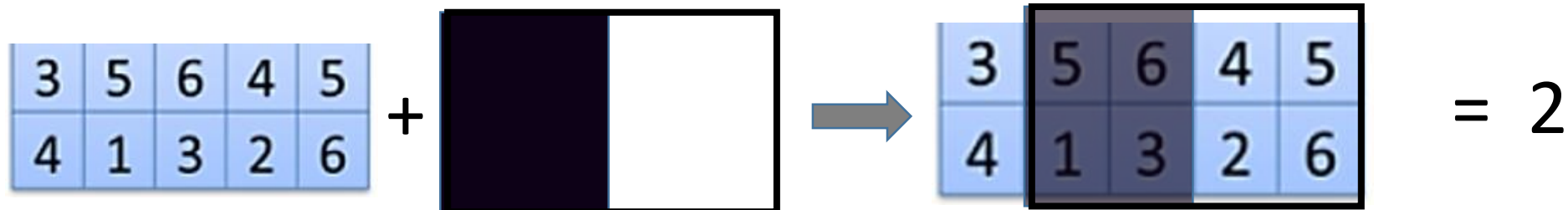


Вычисляемым значением такого признака будет

$$F = X - Y$$

где  $X$  – сумма значений яркостей точек закрываемых светлой частью признака, а  $Y$  – сумма значений яркостей точек закрываемых темной частью признака.

Для их вычисления используется понятие интегрального изображения, рассмотренное выше.



# Обучение классификатора в методе Виолы-Джонса

---

**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.

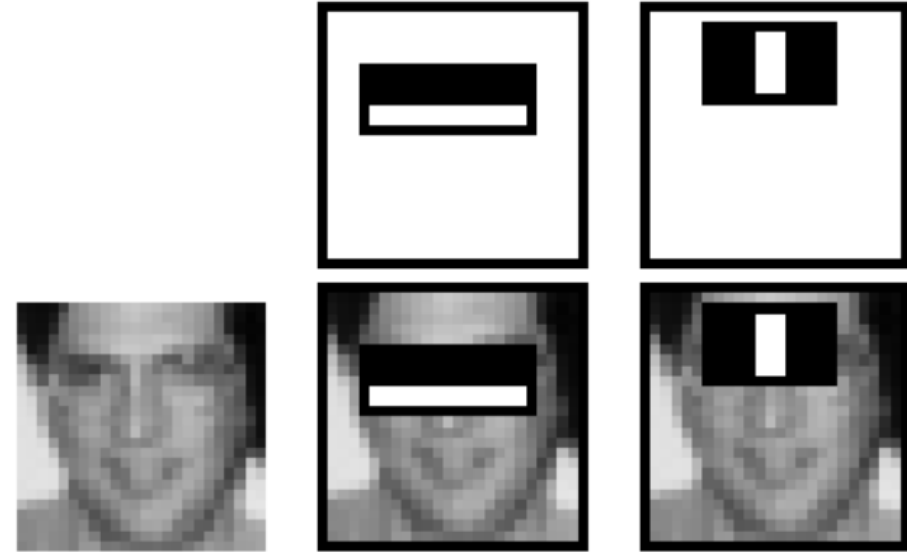
*Предложена Робертом Шапиром (Scharire) в конце 90-х годов, когда надо было найти решение вопроса о том, чтобы имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить один хороший. В основе такой идеи лежит построение *цепочки (ансамбля) классификаторов*, который называется **каскадом**, каждый из которых (кроме первого) *обучается на ошибках предыдущего*.*

**AdaBoost** (*adaptive boosting*) (1999) может использовать произвольное число классификаторов и производить обучение на одном наборе примеров, поочередно применяя их на различных шагах.



# Каскад признаков

Из двух признаков Хаара строится первый каскад системы по распознаванию лиц, который имеет вполне осмысленную интерпретацию.



Для определения принадлежности к классу в каждом каскаде, находится сумма значений слабых классификаторов этого каскада. Каждый слабый классификатор выдает два значения в зависимости от того больше или меньше заданного порога значение признака, принадлежащего этому классификатору. В конце сумма значений слабых классификаторов сравнивается с порогом каскада и выносятся решения найден объект или нет данным каскадом.

# Алгоритм сканирования окна с признаками



[Демо](#)

# Обучение классификатора в методе Виолы-Джонса

---

В процессе поиска вычислять все признаки затратно. Следовательно, **классификатор** должен реагировать *только на определенное, нужное подмножество всех признаков*. Значит классификатор надо обучить нахождению лиц по данному определенному подмножеству.

**Классификатор(classifier)** — в задачах классификации это аппроксимирующая функция, выносящая решение, к какому именно классу данный объект принадлежит.

В случае алгоритма Виолы-Джонса для идентификации и распознавания лица классификация является *двухклассовой*.

# Обучение OpenCV каскада Хаара

---

Весь процесс обучения выборки не требует навыков программирования. Для этого имеются уже готовые консольные программы, присутствующие в основной сборке OpenCV.

- **Фотографии предмета в реальной среде обитания.** Чем более похожа выборка будет на то, что мы будем распознавать, тем лучше будут результаты. Если обучать распознаватель лица по фотографиям людей из студии, то на улице уровень распознавания будет ниже, чем в студии. На это влияют как тени, одежда, так и выражение лица.
- **Выборка отрицательных фотографий, на которых нет объекта распознавания.** Фотографии должны быть сделаны в той же среде где будет распознавание. Если выборка контрпримеров будет сделана по фотографиям на северном полюсе, а распознавать будете в тропических джунглях, то ничего не заработает.

# Обучение OpenCV каскада Хаара

---

Для того, чтобы начать обучение, нам нужно иметь **2 папки с примерами**. **«Good»** — папка с **позитивными изображениями**, **«Bad»** — с **отрицательными**.

Для каждой папки нужно иметь текстовый файл, в котором описаны используемые изображения. Назовём их «Good.dat» и «Bad.dat». **ВАЖНО!** Этот файл должен лежать на том же уровне файловой системы, на котором лежит папка.

```
\Good
    \1. bmp
    \2. bmp
    \... bmp
    \N. bmp

\Bad
    \1. bmp
    \2. bmp
    \... bmp
    \N. bmp

Good.dat
Bad.dat
```

# Обучение OpenCV каскада Хаара

---

Файлы описания для отрицательных и положительных объектов имеют разную структуру.

```
Bad\1. bmp  
Bad\2. bmp  
Bad\.... bmp  
Bad\N. bmp
```

```
Good \0.bmp 1 0 0 414 148  
Good \1.bmp 1 0 0 568 164  
Good \....bmp 1 0 0 440 144  
Good \N.bmp 1 0 0 590 182
```

Само обучение происходит в два этапа:

- создается коллекция [\*opencv\\_createsamples.exe\*](#)
- расчет итогового каскада. Работает долго. Обучение каскада на 500-1000 объектов займёт почти целый день [\*opencv\\_traincascade.exe\*](#)

Branch: master ▼

[opencv](#) / [data](#) / [haarcascades](#) /

Create new file

Find file










History



**StevenPuttemans** fixing models to resolve XML violation issue

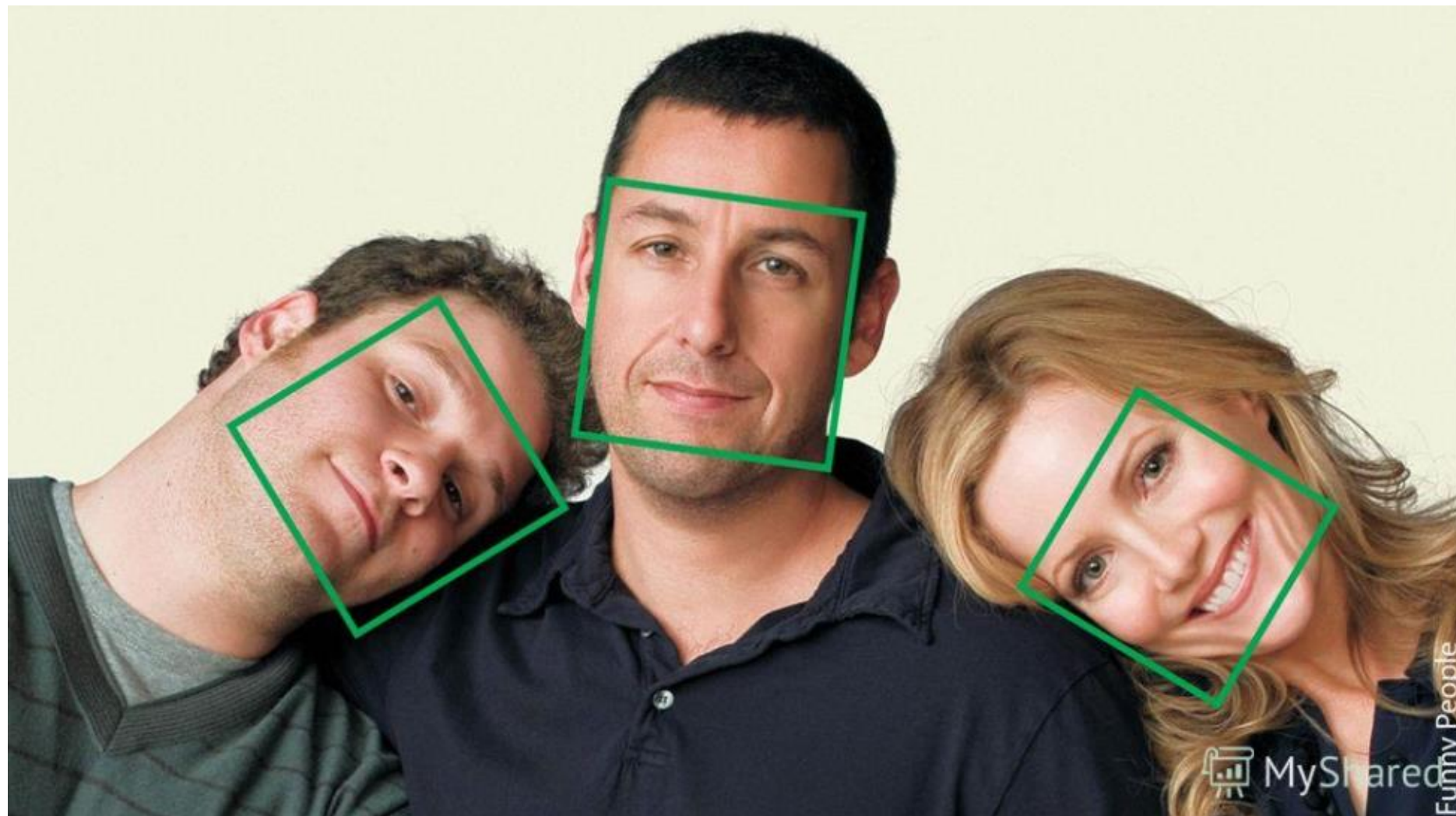
Latest commit 2ddbcc3 on 13 Jun

..

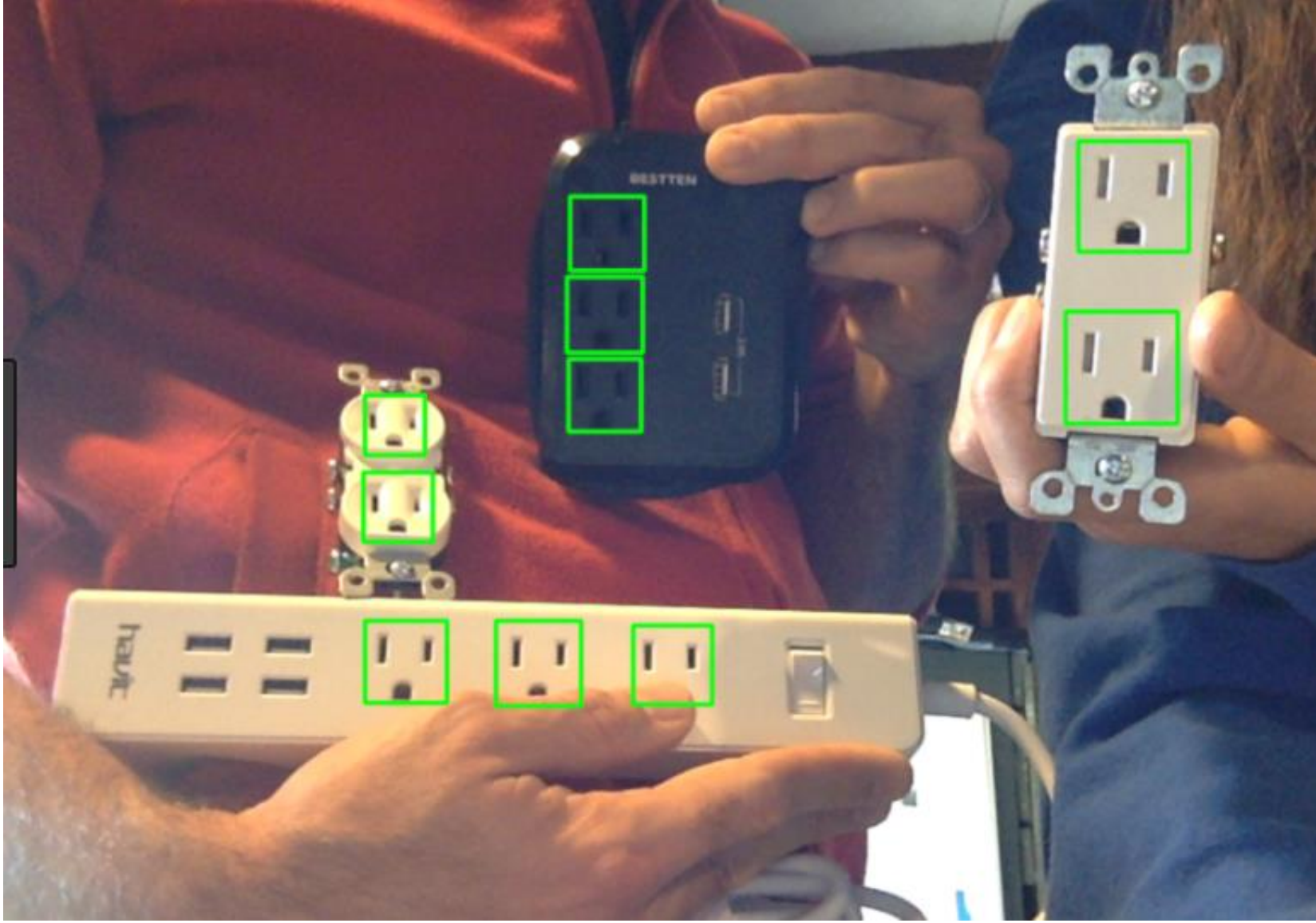
 <a href="#">haarcascade_eye.xml</a>	some attempts to tune the performance	4 years ago
 <a href="#">haarcascade_eye_tree_eyeglasses.xml</a>	some attempts to tune the performance	4 years ago
 <a href="#">haarcascade_frontalcatface.xml</a>	fixing models to resolve XML violation issue	6 months ago
 <a href="#">haarcascade_frontalcatface_extende...</a>	fixing models to resolve XML violation issue	6 months ago
 <a href="#">haarcascade_frontalface_alt.xml</a>	some attempts to tune the performance	4 years ago
 <a href="#">haarcascade_frontalface_alt2.xml</a>	some attempts to tune the performance	4 years ago
 <a href="#">haarcascade_frontalface_alt_tree.xml</a>	some attempts to tune the performance	4 years ago
 <a href="#">haarcascade_frontalface_default.xml</a>	some attempts to tune the performance	4 years ago
 <a href="#">haarcascade_fullbody.xml</a>	fixing wrong model sizes	3 years ago



Почему эти лица не могут быть определены методом  
Виолы-Джонса?







## Результат выполнения лабораторной работы



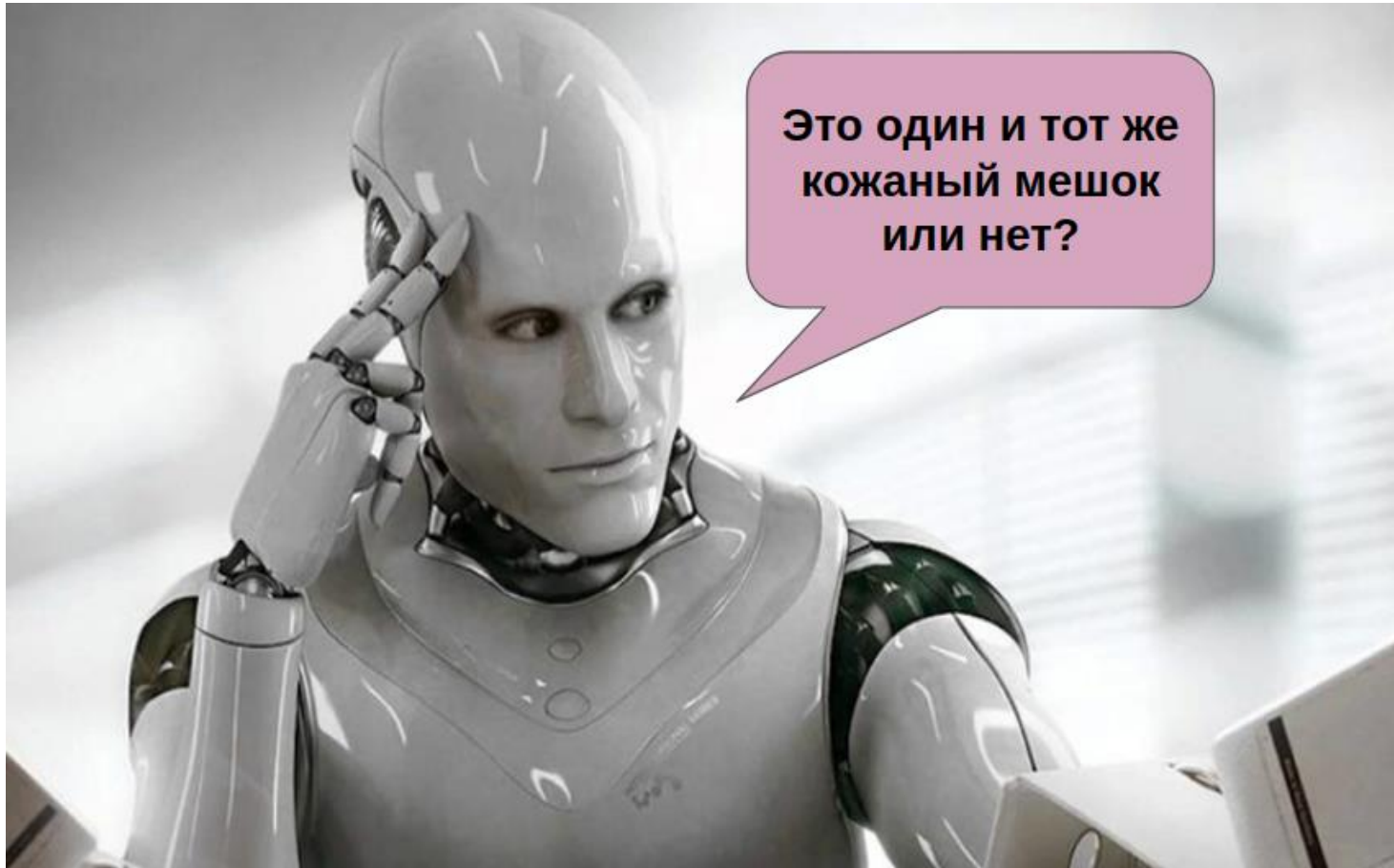


# Распознавание лиц

## Анализ существующих подходов

# Распознавание лиц

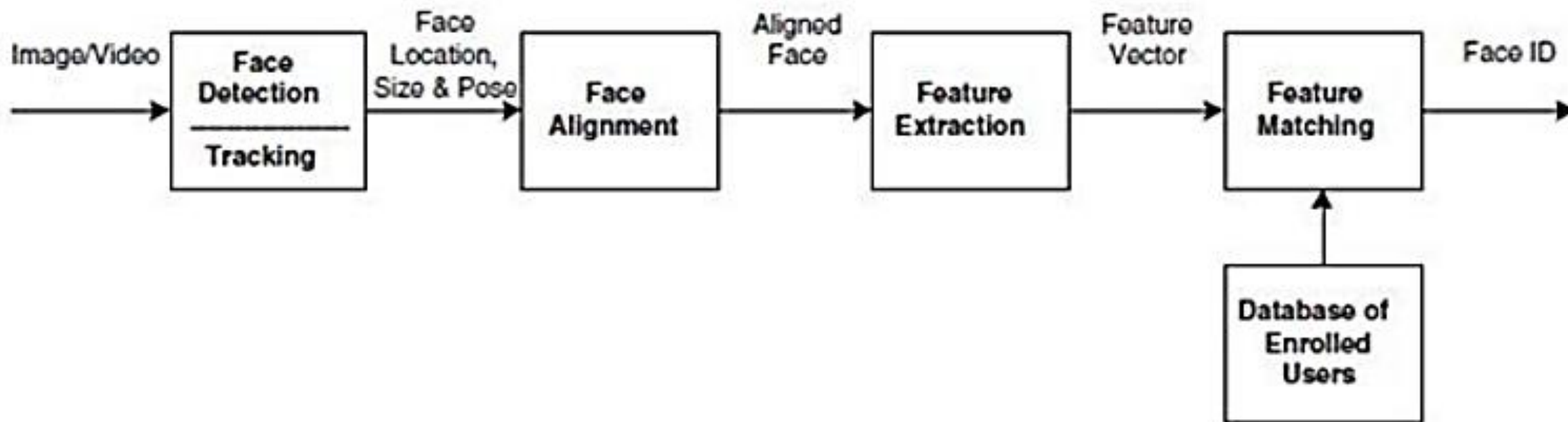
---



# Распознавание лиц

---

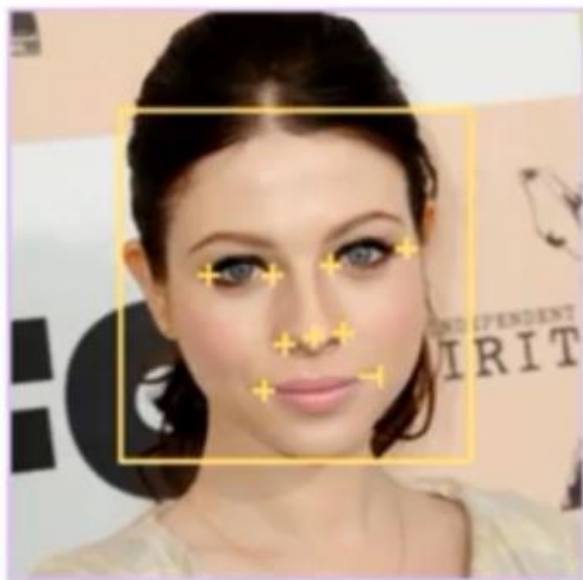
Несмотря на большое разнообразие представленных алгоритмов, можно выделить общую структуру процесса распознавания лиц:



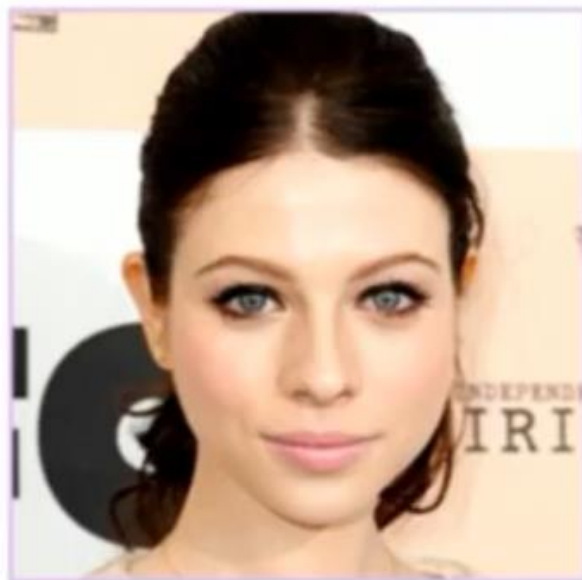


# Этапы распознавания лица

1. Детектирование лица



2. Выравнивание лица



3. Извлечение дескриптора

$$f(I_i) = d_i$$


4. Вычисление схожести

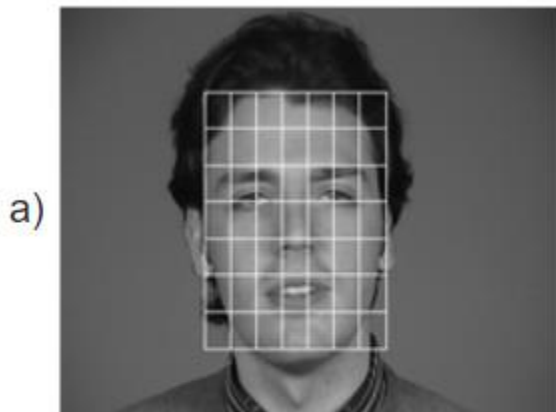
$$S_{ij} = k(d_i, d_j)$$

$$S(\text{Image 1}, \text{Image 2}) = 0$$

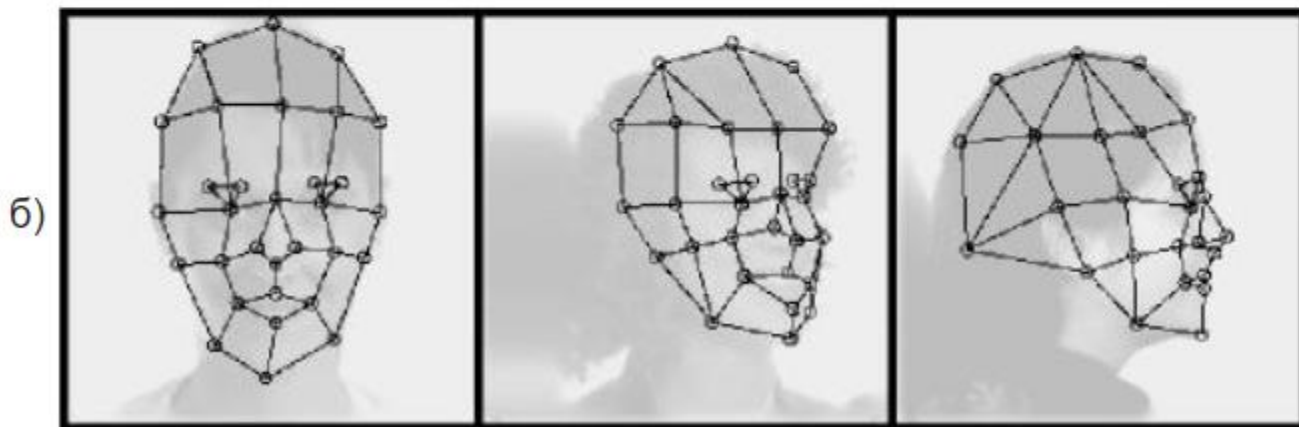

$$S(\text{Image 3}, \text{Image 4}) = 6$$


# 1. Метод гибкого сравнения на графах (Elastic graph matching)

---



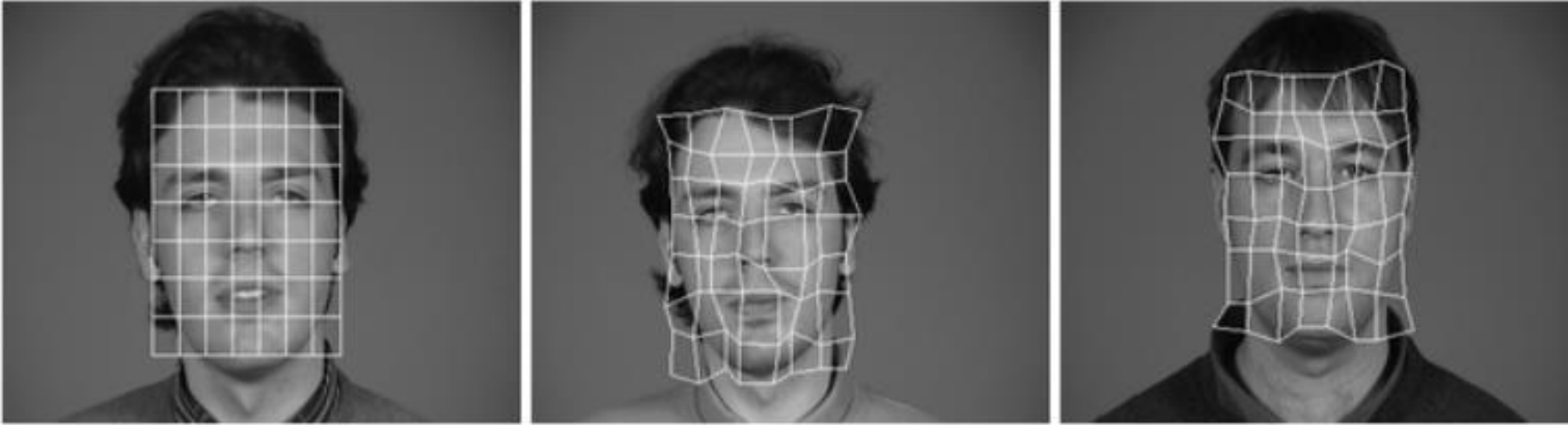
Суть метода сводится к эластичному сопоставлению графов, описывающих изображения лиц.



Графы могут представлять собой как прямоугольную решетку (а), так и структуру, образованную характерными (антропометрическими) точками лица (б).

# 1. Метод гибкого сравнения на графах

---



*Пример деформации графа в виде регулярной решетки*

На этапе распознавания один из графов – эталонный – остается неизменным, в то время как другой деформируется с целью наилучшей подгонки к первому.

Различие между двумя графами вычисляется при помощи некоторой ценовой функции деформации, учитывающей как различие между значениями признаков, вычисленными в вершинах, так и степень деформации ребер графа.



# 1. Метод гибкого сравнения на графах

---

В отдельных публикациях указывается **95-97%-ая эффективность** распознавания даже при наличии различных эмоциональных выражениях и изменении ракурса лица до 15 градусов. Однако разработчики систем эластичного сравнения на графах ссылаются на высокую вычислительную стоимость данного подхода.

## Недостатки:

- высокая вычислительная сложность процедуры распознавания
- низкая технологичность при запоминании новых эталонов
- линейная зависимость времени работы от размера базы данных лиц.

## 2. Метод главных компонент или principal component analysis (PCA).

---

В задаче распознавания лиц его применяют главным образом для представления изображения лица вектором малой размерности (главных компонент), который сравнивается затем с эталонными векторами, заложенными в базу данных.

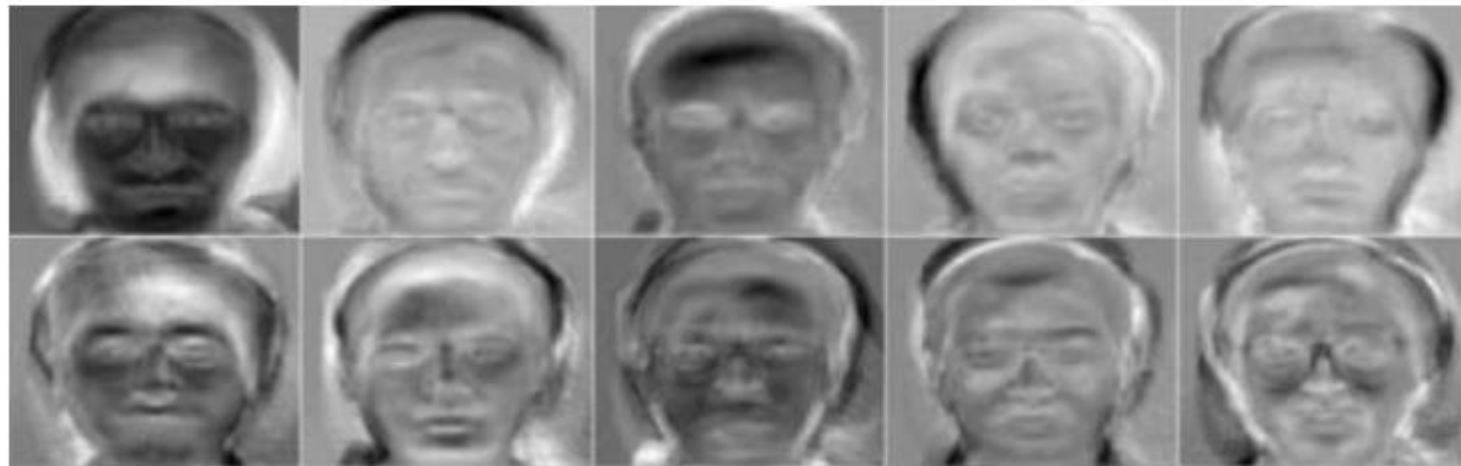
**Главной целью метода главных компонент является значительное уменьшение размерности пространства признаков** таким образом, чтобы оно как можно лучше описывало «типичные» образы, принадлежащие множеству лиц.

Полученный один раз на обучающей выборке изображений лиц набор собственных векторов используется для кодирования всех остальных изображений лиц, которые представляются взвешенной комбинацией этих собственных векторов.

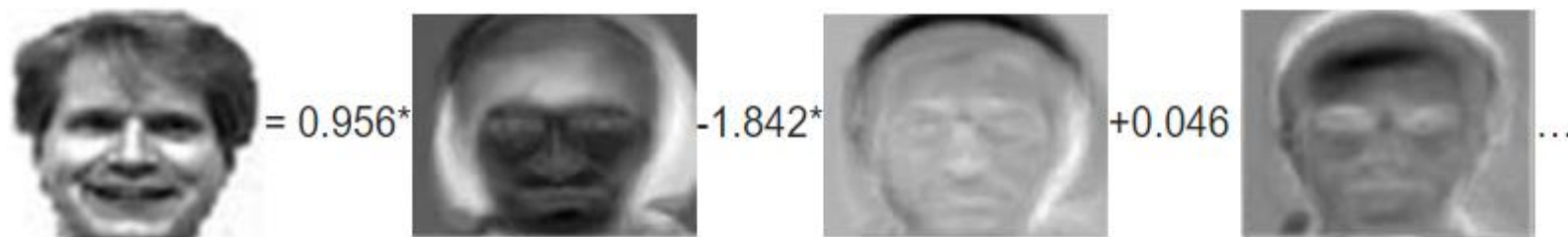
Используя ограниченное количество собственных векторов можно получить сжатую аппроксимацию входному изображению лица, которую затем можно хранить в базе данных в виде вектора коэффициентов, служащего одновременно ключом поиска в базе данных лиц.

# Метод главных компонент или principal component analysis (PCA).

---



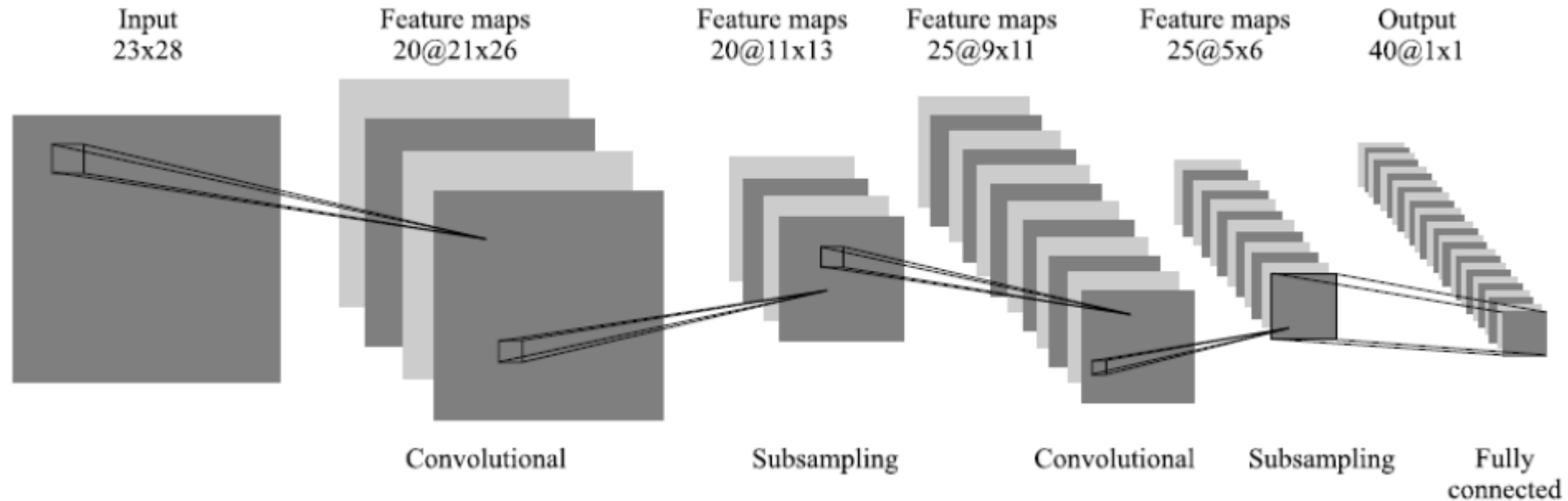
*Пример первых десяти собственных векторов (собственных лиц), полученных на обучаемом наборе лиц*



*Пример построения (синтеза) человеческого лица с помощью комбинации собственных лиц и главных компонент*

### 3. Нейронные сети

---

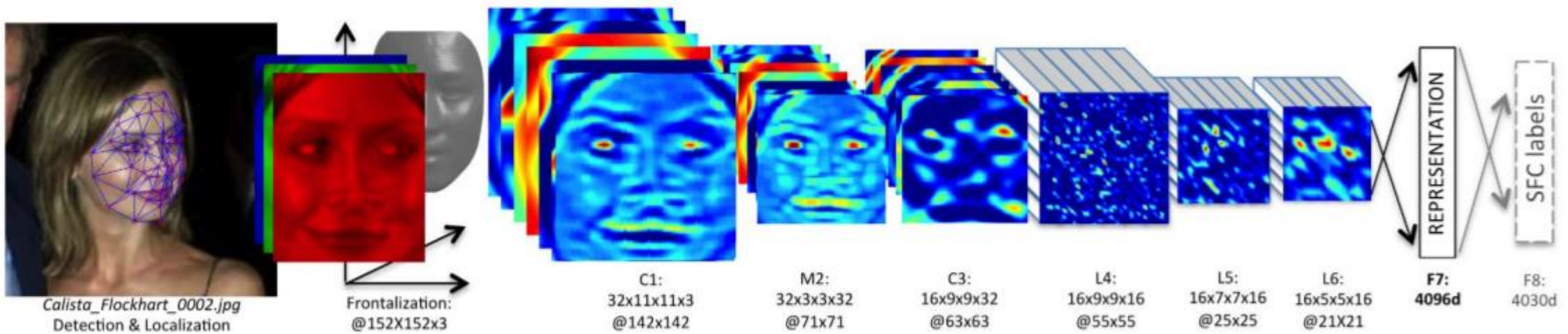


Наилучшие результаты в области распознавания лиц (по результатам анализа публикаций) показала Convolutional Neural Network или сверточная нейронная сеть (СНС).

СНС обеспечивает частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям.

# Нейронные сети. DeepFace

Тестирование СНС показало **96% точность** распознавания.



Свое развитие СНС получили в разработке DeepFace, которую приобрел Facebook для распознавания лиц пользователей своей соцсети.

# DeerFace. Выравнивание лица

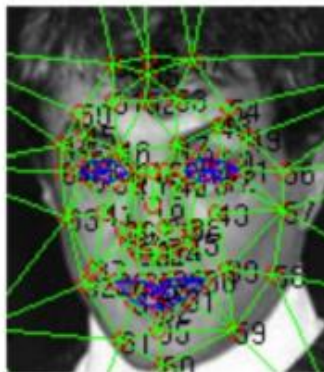
---



(a)



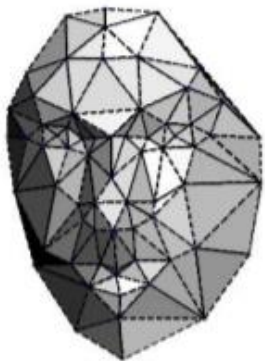
(b)



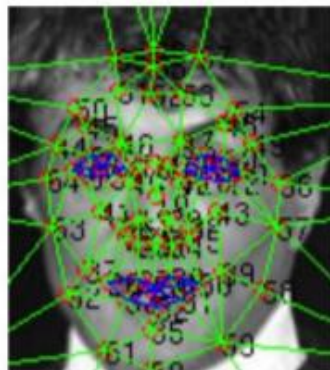
(c)



(d)



(e)



(f)



(g)



(h)

Detect 67 “fiducial points” (using SVR).

Compare them with a fixed 3D model of a generic face.

Reconstruct the 3D model of the detected face.

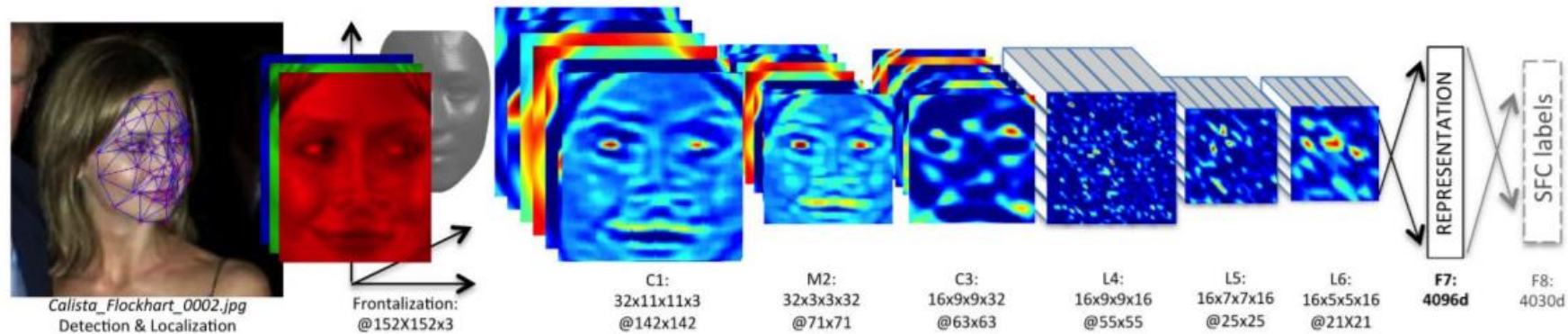
Generate the 3D-aligned version of the crop.

The final image is given to the DNN.



# Нейронные сети.

---



**Недостатки нейронных сетей:** добавление нового эталонного лица в базу данных требует полного переобучения сети на всем имеющемся наборе (достаточно длительная процедура, в зависимости от размера выборки от 1 часа до нескольких дней). Проблемы математического характера, связанные с обучением: попадание в локальный оптимум, выбор оптимального шага оптимизации, переобучение и т. д. Трудно формализуемый этап выбора архитектуры сети (количество нейронов, слоев, характер связей).

Обобщая все вышесказанное, можно заключить, что НС – «черный ящик» с трудно интерпретируемыми результатами работы.



Размерность этого вектора у каждой системы может быть своя, обычно это некоторая степень двойки: 128, 256 или 512. Какой бы ни была размерность, норма вектора равна единице:

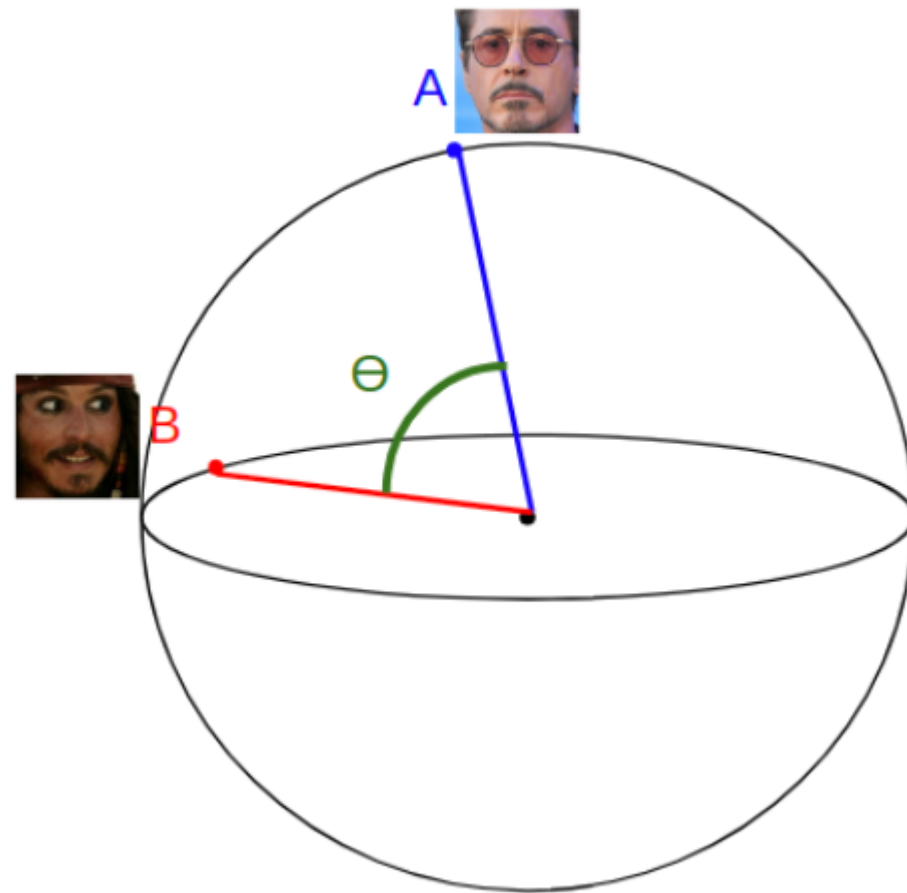
для вектора  $a = (a_1, a_2 \dots a_N)$  будет справедливо  $\sqrt{a_1^2 + a_2^2 + \dots + a_N^2} = 1$



Это означает, что **все возвращаемые системой векторы лежат на N-мерной гиперсфере**, где  $N$  — размерность вектора. Представить себе такую гиперсферу довольно сложно, поэтому — опять-таки, для простоты — прибегнем к известному совету Джеффри Хинтона о визуализации многомерного пространства:

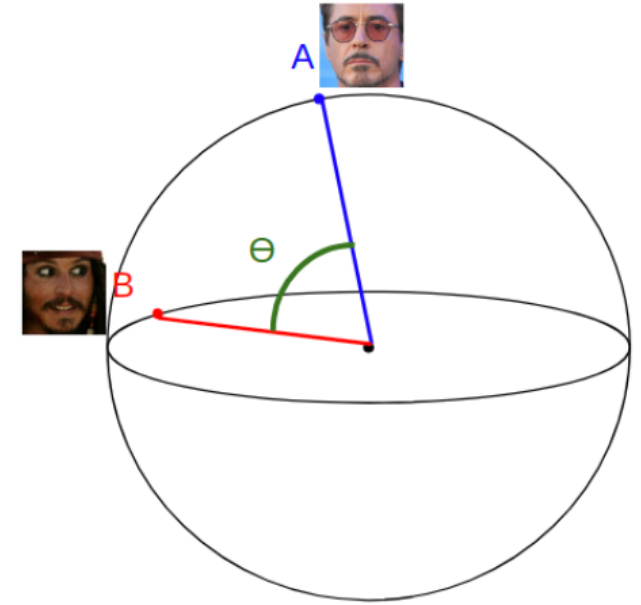
*To deal with hyper-planes in a 14-dimensional space, visualize a 3-D space and say “fourteen” to yourself very loudly.*

*Everyone does it. (Когда вам предстоит иметь дело с гиперплоскостями в 14-мерном пространстве, представьте его себе трёхмерным и громко скажите: «Четырнадцать!» Все так поступают.)*



Эти векторы обладают следующим свойством: если мы попробуем дважды закодировать одно и то же изображение лица, мы получим два одинаковых вектора — угол между ними будет равен нулю, а чем сильнее будут различаться лица, тем дальше друг от друга они будут лежать на сфере и тем больше будет угол между ними. Это означает, что для определения «похожести» двух лиц нам достаточно измерить угол между их векторами; удобнее всего в качестве меры схожести использовать косинус угла, а не сам угол (не забываем, что все вектора имеют норму, равную 1):

$$similarity = \cos \Theta = a_1 b_1 + a_2 b_2 + \dots + a_N b_N$$

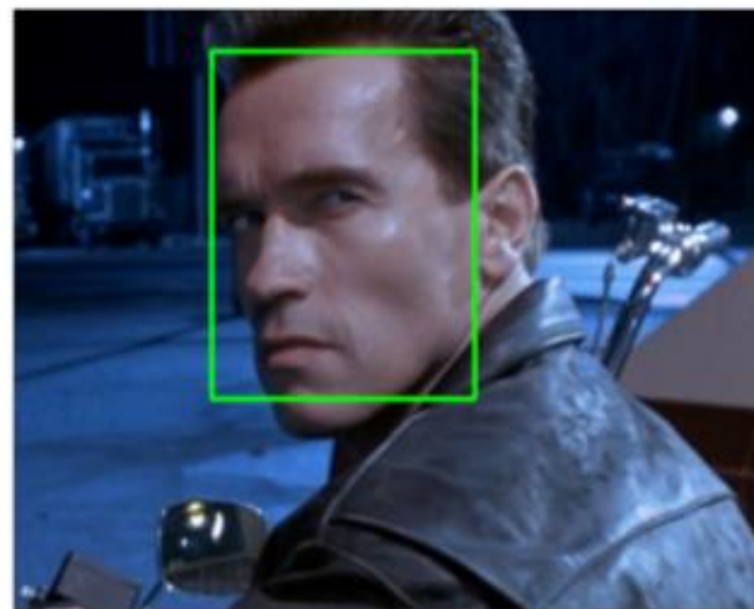
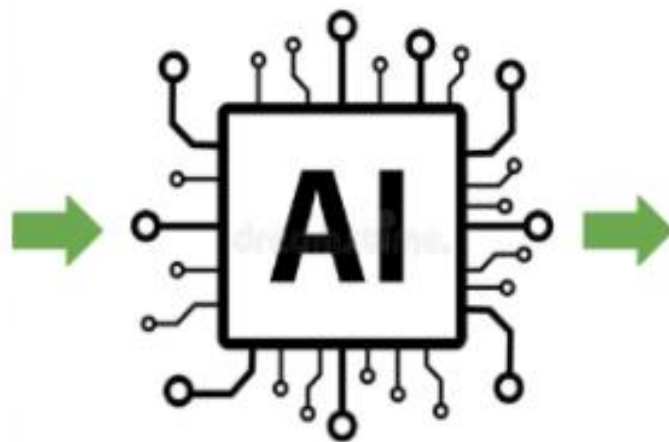


Система распознавания лиц не может нам сказать, что на некоторой фотографии изображён условный Иванов И. И. (или наоборот, что на фото совсем не Иванов) — она работает по-другому.

Мы можем взять реальное фото Иванова и при помощи системы построить для него вектор признаков. В дальнейшем этот вектор можно будет сопоставить с вектором исследуемого изображения и узнать меру их схожести.

## Детектор

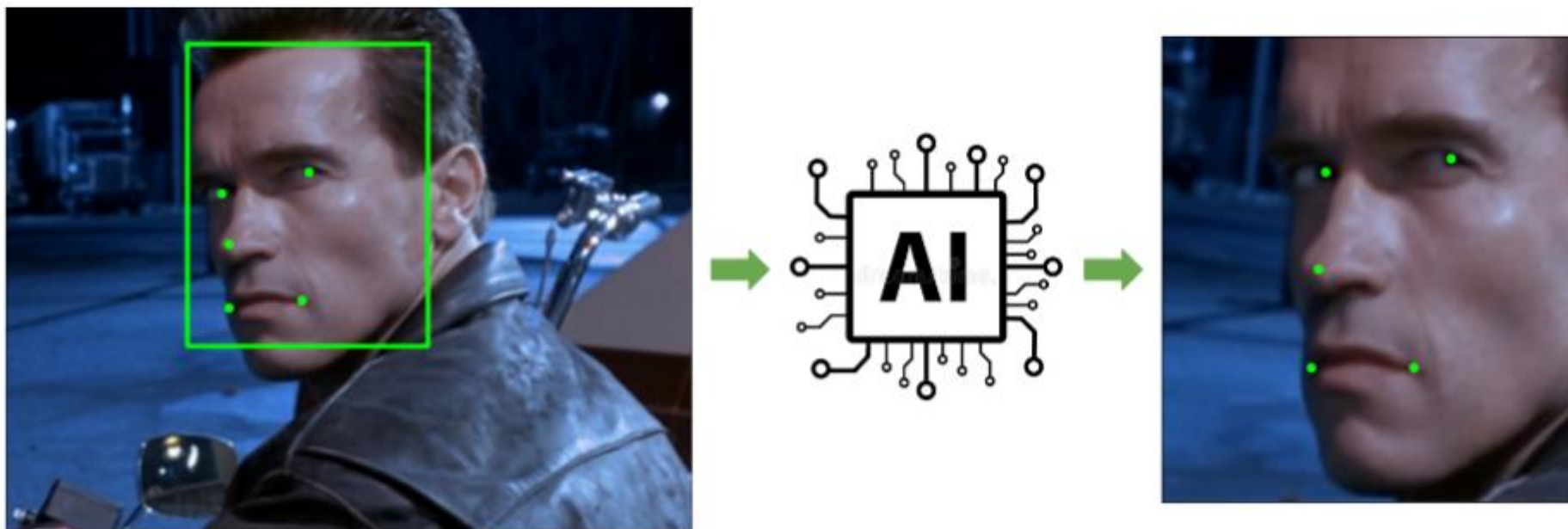
Теперь будем постепенно открывать чёрный ящик. Первым делом, получив на вход картинку, алгоритму нужно отыскать на ней лица людей. За это отвечает компонент, называемый детектором, и его задача — выделить области, в которых содержится нечто, напоминающее лицо.



# Нормализатор

- **scale:** мы можем «приблизить» или «отдалить» лицо;
- **rotation:** мы можем повернуть лицо на любой угол в плоскости изображения;
- **shift:** мы можем сместить лицо на несколько пикселей влево или вправо, вверх или вниз.

Каждое из этих преобразований описывается матрицей  $3 \times 3$ . Перемножив все три матрицы, мы также получим матрицу  $3 \times 3$  для суммарного преобразования — его необходимо применить к лицу для приведения к нужному нам виду:

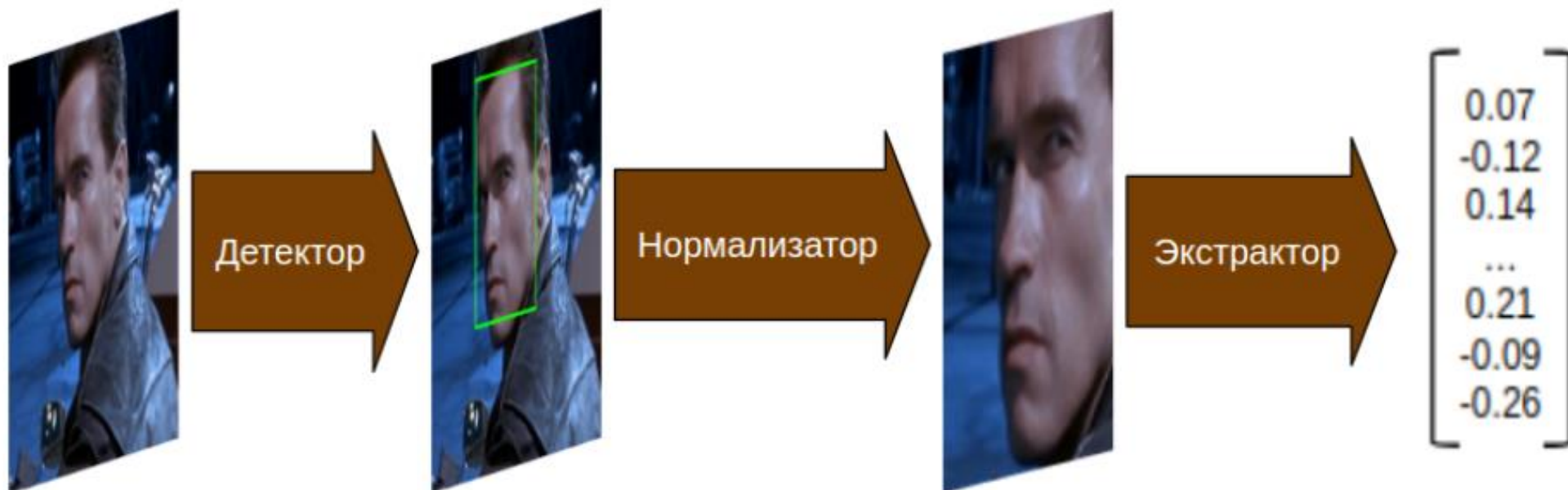




## Экстрактор

Теперь, когда у нас есть нормализованное лицо, настало время строить вектор — этим занимается компонент, называемый экстрактором, основной элемент всей системы. Он принимает на вход картинки фиксированного разрешения — обычно 90–130 пикселей, такой размер позволяет соблюсти баланс между точностью работы алгоритма и его скоростью (картинка большего разрешения могла бы содержать больше полезной для распознавания информации, но и обработка её выполнялась бы дольше).

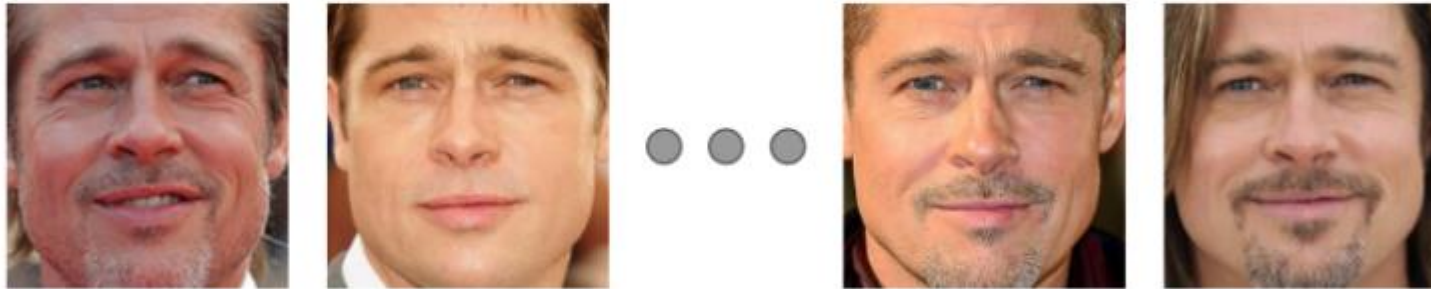
Экстракция вектора — завершающий этап пайплайна обработки лица, который можно схематически изобразить следующим образом:



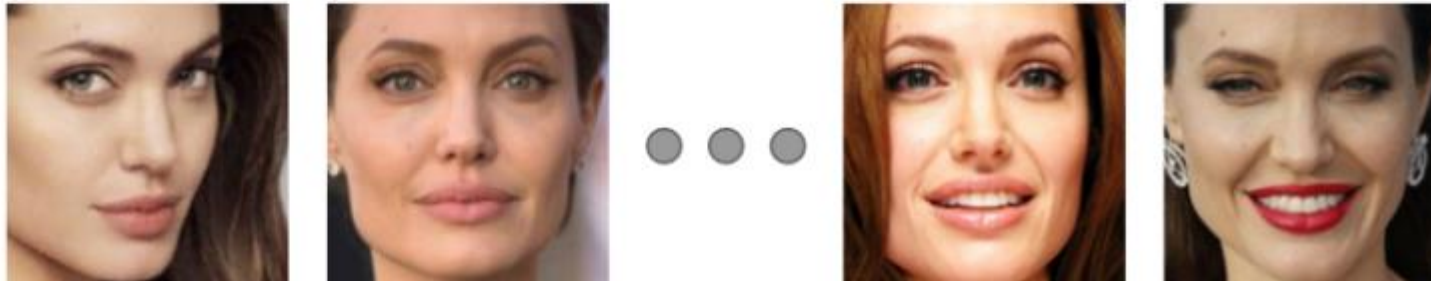
## Обучение экстрактора

Главное, чего мы ждём от хорошего экстрактора — чтобы он строил как можно более «близкие» векторы для схожих лиц и как можно более «далёкие» — для непохожих. Для этого экстрактор нужно обучить, а для обучения первым делом нам понадобится датасет — набор размеченных данных. Выглядеть он может примерно так:

Персона 1



Персона 2



Сколько персон нам нужно для обучения? И сколько фотографий для каждой персоны? Напрашивается очевидный ответ: чем больше, тем лучше. Самые крутые системы обучаются на датасетах в миллионы, а то и в десятки миллионов персон, а вот фотографий на каждую из них нам будет достаточно пяти–десяти

## ВОПРОСЫ

- На что похожи признаки, которые определяет нейросеть?
- Чем руководствуется алгоритм при построении вектора?
- Может ли вектор сам по себе что-то сказать нам о внешности человека?
- На какие именно части лица обращает внимание алгоритм?

**это всё очень хорошие вопросы, и они сильно интересуют не только обывателей, но и самих исследователей — разработчиков нейросетей.**

Если попросить простого человека описать приметы некоторого лица, он наверняка назовёт разрез и цвет глаз, особенности причёски и растительности на лице, длину носа, изгиб бровей... Тренированный физиогномист (например, пограничник, который проверяет ваш паспорт в аэропорту, или криминалист, специализирующийся на портретной экспертизе) оценит расположение антропометрических точек и ключевые расстояния между ними. Так же и нейросеть: она, безусловно, «обращает внимание» на характерные особенности лица, однако нужно понимать, что **каждое из чисел, составляющих вектор признаков, не отвечает за какую-то конкретную точку или черту лица. Мы не можем, взглянув на числовое представление вектора, указать, что вот этот его участок описывает глаза, а вот тот — форму носа.**



Итак, для ручного анализа вектор непригоден. В то же время, существует немалое количество научных исследований, авторы которых пытаются восстановить внешность человека, располагая вектором признаков. Среди них упомянем работу учёных из Канады и США [Vec2Face](#). Суть предлагаемого ими решения, если излагать её максимально упрощённо, такая: для генерации изображения из вектора признаков будем использовать специальным образом устроенную нейросеть, а датасет для её обучения получим, прогнав большое количество фотографий через экстрактор и сохранив полученные векторы.

Авторам удалось получить весьма приемлемый результат:



# Как всё работает на практике





# Показатели качества



0.92



0.68



0.54



0.45



0.54



0.54



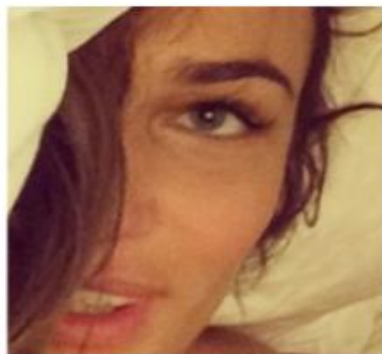
0.64



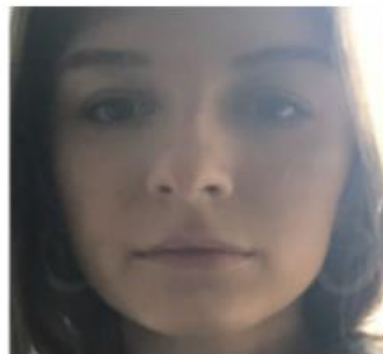
0.82



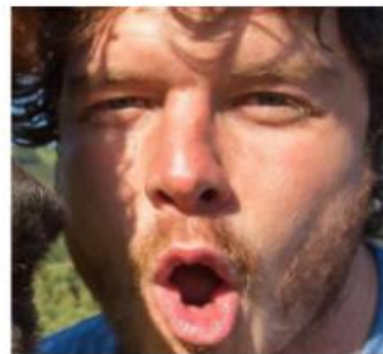
0.73



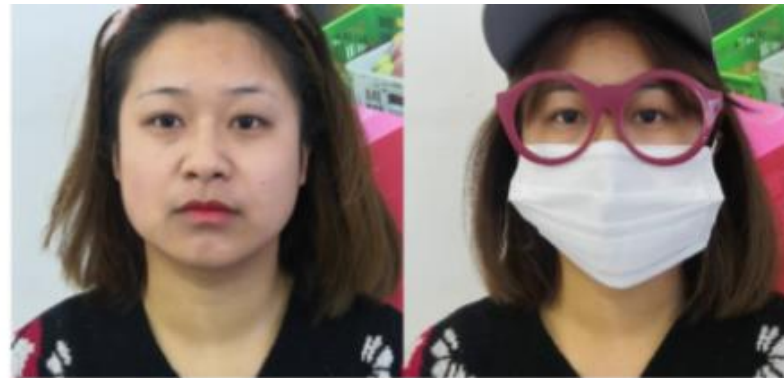
0.53



0.89



0.65



Similarities (masks, glasses, hats)

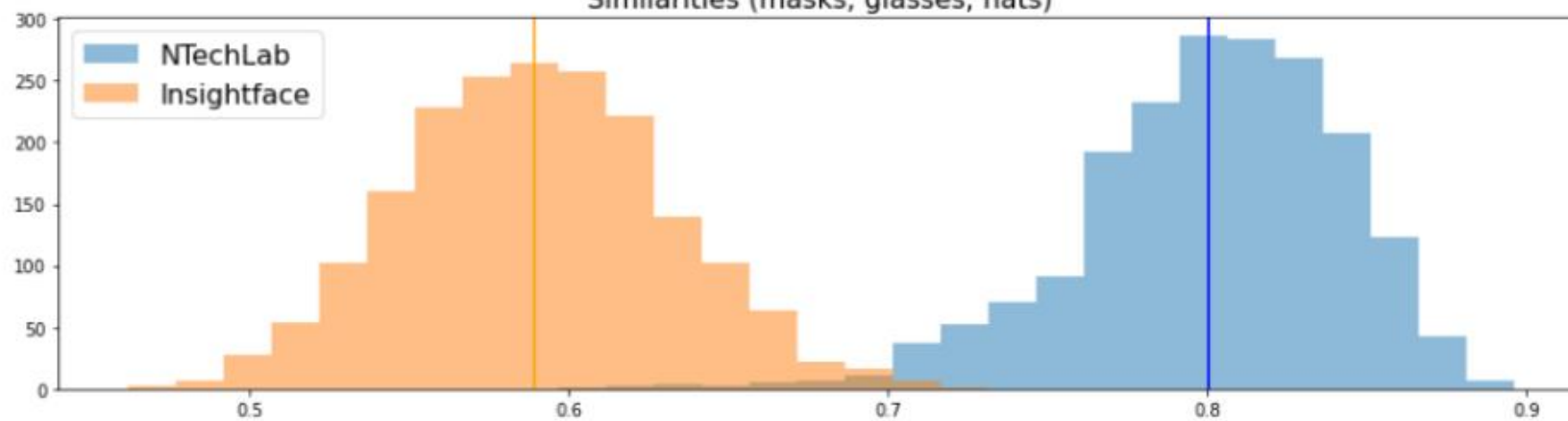
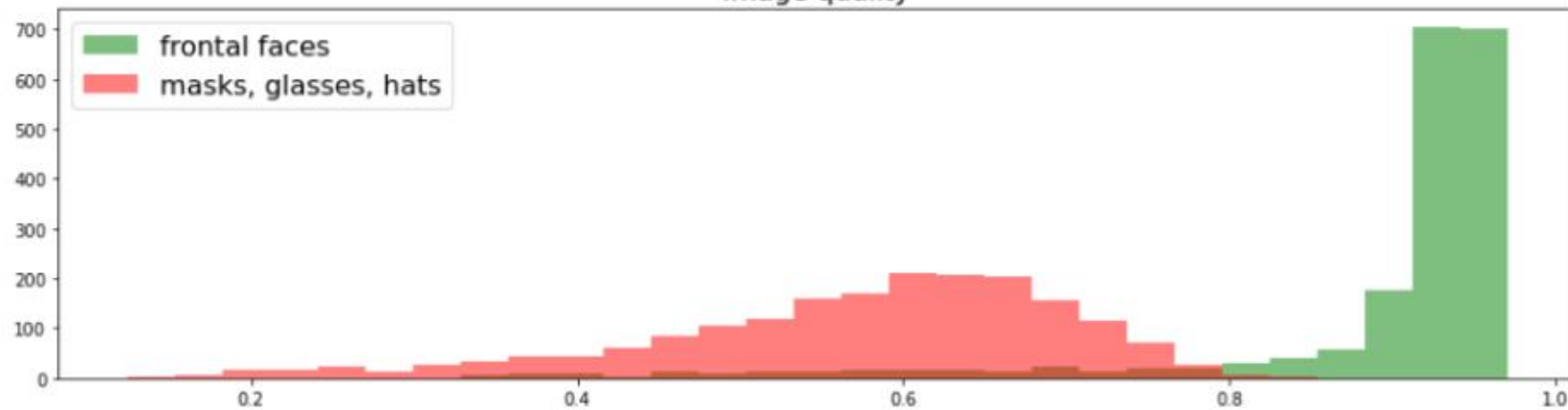
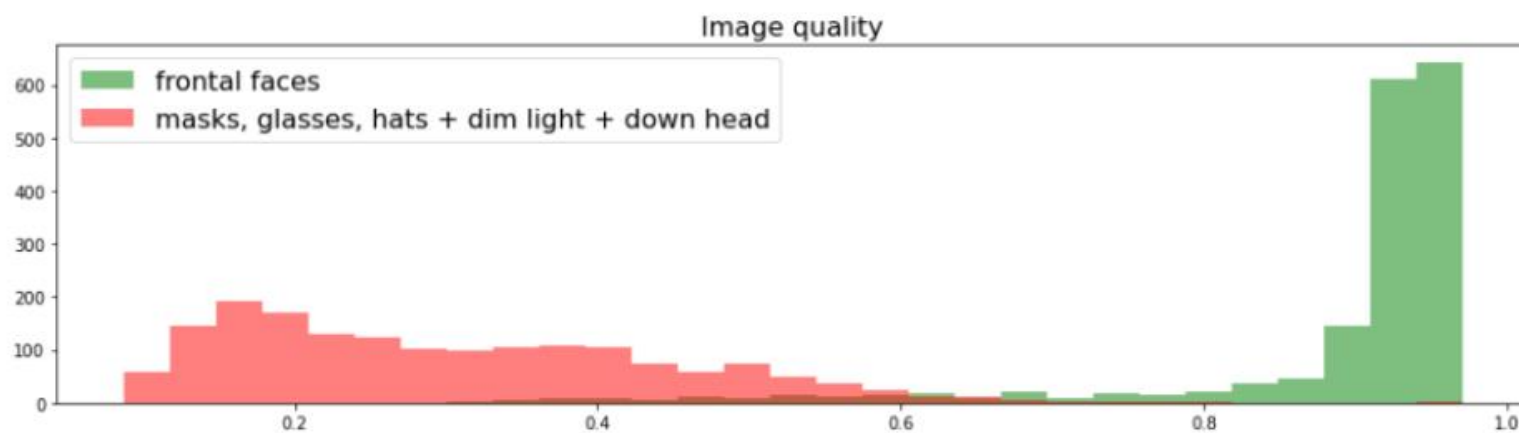
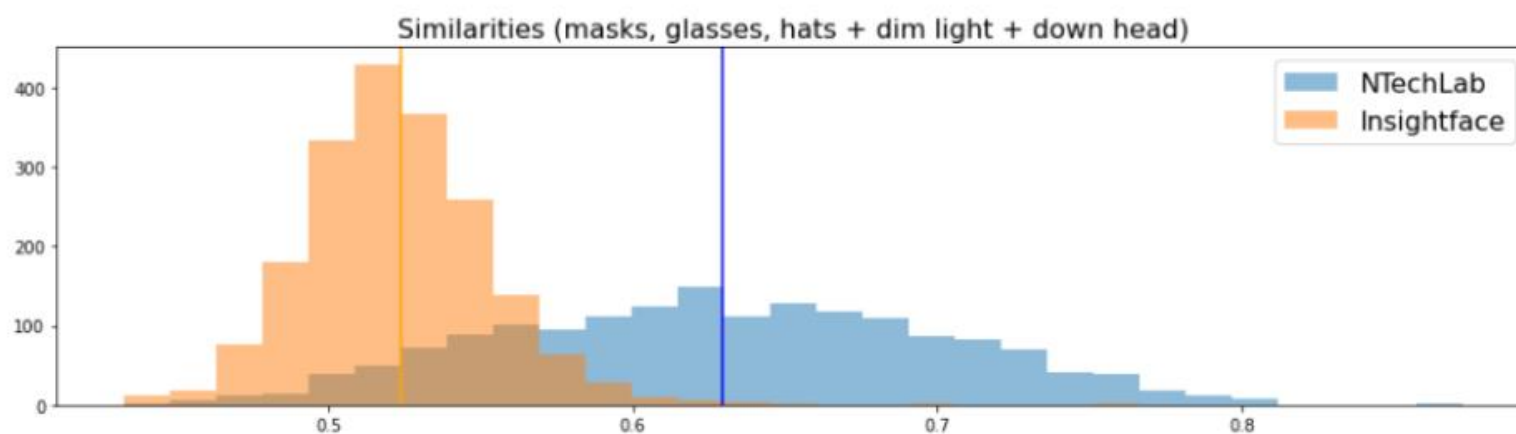
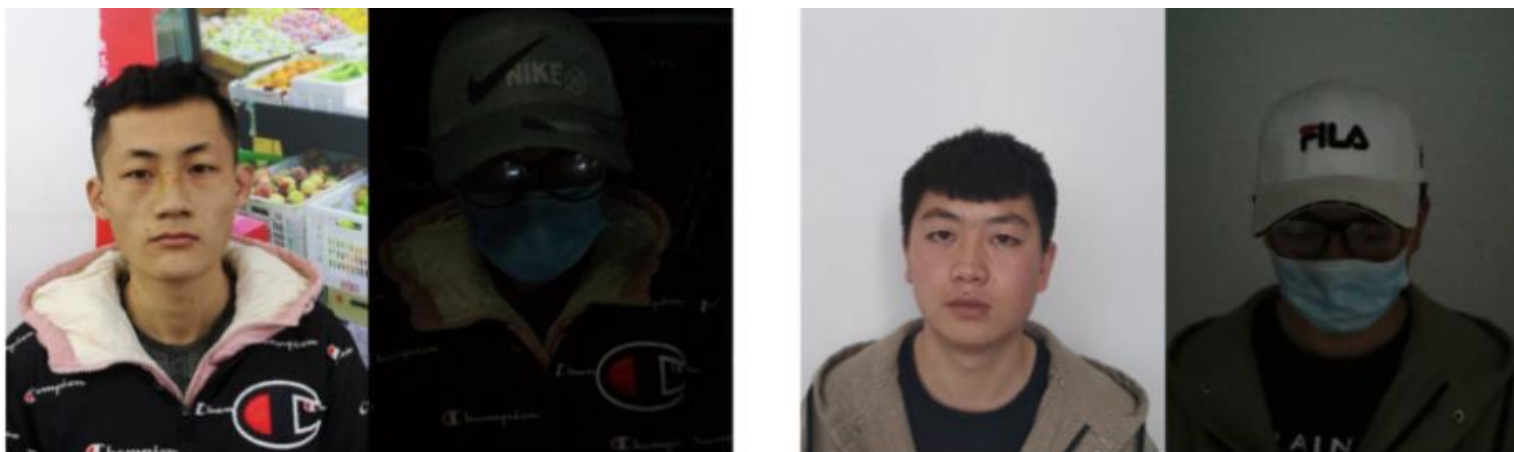


Image quality





# Оценка качества распознавания

возможны только четыре исхода сравнения:

- ✓ **истинно-положительный** (true match): оба сравниваемых образца фактически принадлежат одному и тому же человеку;
- ✓ **истинно-отрицательный** (true non-match): оба сравниваемых образца фактически принадлежат разным людям;
- ✓ **ложноотрицательный (ошибка первого рода, false non-match)**: не опознал, в то время как оба сравниваемых образца фактически принадлежат одному и тому же человеку;
- ✓ **ложноположительный (ошибка второго рода, false match)**: опознал (принял одного человека за другого), в то время как оба сравниваемых образца фактически принадлежат разным людям.

## False Rejection Rate (FRR)

ложный отказ (ошибка первого рода)

## False Acceptance Rate (FAR)

ложный доступ (ошибка второго рода)

**0,01% FAR - 5% FRR**

Из 10 000 сравнений  
1 раз пропустит чужака  
каждый раз из 20 не узнает своего



## **Качество распознавания зависит от:**

- освещения
- ориентации камеры
- размера лица в кадре
- качества снимков в БД

# Распознавание лиц

---

## плюсы:

- удобство

## минусы:

- наличие ошибок
- меньшая защита от подделки идентификатора\*
- требуется более мощное вычислительное оборудование\*
- высокая стоимость\*

## Устойчивость к маскировке



99,3%



99%



56%



51%



99,4%



97%



99%



83%



99%



91%



-



99%

## Условия освещения

30 люкс



96%

5 люкс



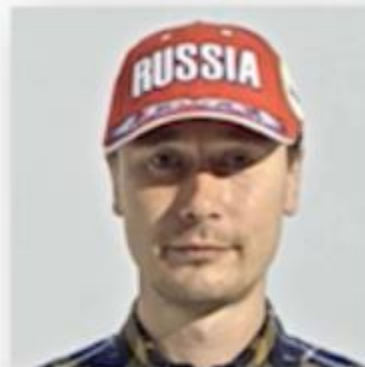
90%

Освещение  
сбоку



95%

Освещение  
сверху



77%

ИК подсветка



99%



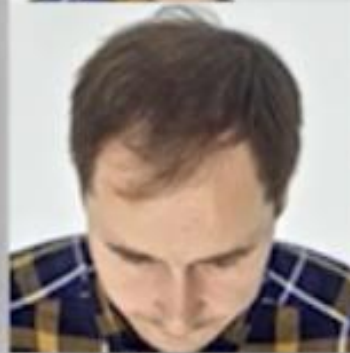


## Угол наклона к плоскости лица

15 град

30 град

45 град



99%

96%

79%



# Источники

---

<https://habrahabr.ru/post/133826/>

<https://habr.com/ru/company/ntechlab/blog/586770/>

<https://habr.com/ru/company/recognitor/blog/418127/>