Задачу распознавания лица можно разделить на 4 части:

* Обнаружение лица. Посмотрите на картинку и найдите в ней лицо.
* Сбор данных: Извлеките уникальные характеристики лица, которые можно использовать, чтобы отличить его от другого человека, например, глаз, рта, носа и т. д.
* Сравнение данных: несмотря на изменения в свете или выражении, он будет сравнивать эти уникальные функции со всеми функциями всех людей, которых вы знаете.
* Распознавание лица (конкретного)

OpenCV предоставляет нам два предварительно подготовленных и готовых к использованию классификатора для обнаружения лиц:

* Классификатор Хаара
* Классификатор LBP (Локальные двоичные шаблоны)

Оба эти классификатора обрабатывают изображения в серых тонах, потому, что нам не нужна информация о цвете, чтобы определить, имеется ли лицо на изображении.

Исторически сложилось так, что алгоритмы, работающие только с интенсивностью изображения (например значение RGB в каждом пикселе), имеют большую вычислительную сложность.

**Классификатор Хаара** - это алгоритм, созданный Полом Виолой и Майклом Джонсом, который обучается из положительных образов (с лицами) и отрицательных (без лиц). Он начинается с извлечения признаков Хаара из каждого изображения, как показано ниже (рис 1).

Признак Хаара состоит из смежных прямоугольных областей. Они позиционируются на изображении, далее суммируются интенсивности пикселей в областях, после чего вычисляется разность между суммами. Эта разность и будет значением определенного признака, определенного размера, определенным образом спозиционированного на изображении.

Теперь все возможные размеры каждого окна помещаются во все возможные местоположения каждого изображения, чтобы вычислить множество функций. (рис 2)

Например, в приведенном выше изображении мы извлекаем два признака. Первый фокусируется на том, что область глаз часто темнее области носа и щеки. Следовательно общим признаком Хаара для лиц является 2 смежных прямоугольных региона, лежащих на глазах и щеках. Второй признак зависит от свойства, что глаза темнее, чем переносица.

Но среди всех вычислений большинство признаков не имеют значения. Например, при рассмотрении щек окна становятся неуместными, потому что ни одна из этих областей темнее или светлее других областей на щеках, все области здесь одинаковы. Поэтому мы отбрасываем нерелевантные признаки и сохраняем только те, которые имеют отношение к технике Adaboost. AdaBoost - это учебный процесс для обнаружения лиц, который выбирает только те функции, которые улучшают точность классификации (лицо / не-лицо) нашего классификатора.

В конечном итоге алгоритм считает, что в целом большая часть области изображения является областью без лица. Учитывая это, лучше всего иметь простой способ проверить, является ли окно областью без лица, и если это не так, немедленно отбросить его и не обрабатывать снова. Поэтому мы можем сосредоточиться в основном на области, где есть лицо.

**LBP классификатор.** Он представляет собой эффективный оператор, который представляет каждый пиксель изображения в виде бинарного числа, зависящего от интенсивностей соседних пикселей изображения.

Этот оператор является эффективным в вычислительном плане, так как работает только с целочисленной арифметикой (это позволяет достигать real-time производительности в некоторых задачах), а также он инвариантен к изменениям яркости на изображении, вызванным съемкой в различных условиях освещения.

Каждое обучающее изображение делится на несколько блоков. Для каждого блока LBP смотрит на 9 пикселей (3 × 3 окна) за раз и с особым интересом к пикселю, расположенному в центре окна. Затем он сравнивает значение центрального пикселя со значением пикселя каждого соседа под окном 3 × 3. Для каждого соседнего пикселя, который больше или равен центральному пикселю, он устанавливает его значение равным 1, а для остальных - 0. После этого он считывает обновленные значения пикселей (которые могут быть либо 0 или 1) по часовой стрелке, и формирует двоичное число. Затем он преобразует двоичное число в десятичное число, и это десятичное число является новым значением центрального пикселя. Мы делаем это для каждого пикселя в блоке. (рис 4) Затем он преобразует значения каждого блока в гистограмму, поэтому теперь мы получили одну гистограмму для каждого блока изображения, например: (рис 5)

Наконец, он объединяет эти блок-гистограммы, чтобы сформировать один вектор признаков для одного изображения, который содержит все интересующие нас признаки. Таким образом, мы извлекаем LBP- признаки из изображения.

**Какой способ лучше?**

Каждый классификатор обнаружения лица OpenCV имеет свои плюсы и минусы, но основные отличия в точности и скорости.

Таким образом, если нужны более точные обнаружения, нужно исп классификатор. Он подходит в таких технологиях, как системы безопасности. Но классификатор LBP быстрее, поэтому он должен использоваться в мобильных приложениях или встраиваемых системах.

OpenCV предоставляет нам класс cv2.CascadeClassifier, который принимает в качестве входного файла обучающий файл классификатора (Haar / LBP), который мы хотим загрузить.

CascadedClassifier поставляется с функцией detectMultiScale, которая точно определяет лица.

detectMultiScale (image, scaleFactor, minNeighbors): Это общая функция для обнаружения объектов, в этом случае она будет обнаруживать лица, так как мы вызывали в каскаде лица. Если он находит лицо, он возвращает список позиций указанного лица в форме «Rect (x, y, w, h)», если нет, то возвращает «None».

* image: первый параметр - изображение в оттенках серого.
* scaleFactor: Эта функция компенсирует ложное восприятие в размере, которое возникает, когда одно лицо больше, чем другое, просто потому, что оно ближе к камере.
* minNeighbors: Это алгоритм обнаружения, который использует движущееся окно для обнаружения объектов, он делает это, определяя, сколько объектов найдено возле текущего, прежде чем оно сможет объявить найденное лицо.

**Распознавание**

* **EigenFaces** – cv2.face.createEigenFaceRecognizer()
* **FisherFaces** – cv2.face.createFisherFaceRecognizer()
* **Local Binary Patterns Histograms (LBPH)** – cv2.face.createLBPHFaceRecognizer()

**EigenFaces** (собственные лица)

Этот алгоритм считает, что не все части лица одинаково важны или полезны для распознавания лиц. В самом деле, когда вы смотрите на кого-то, вы узнаете этого человека своими особыми чертами, такими как глаза, нос, щеки или лоб; и как они меняют свое отношение друг к другу.

В этом смысле вы фокусируетесь на областях максимального изменения. Например, от глаз до носа происходит значительное изменение, и то же самое происходит от носа до рта. Когда вы смотрите на несколько лиц, вы сравниваете их, глядя на эти области, потому что, улавливая максимальную вариацию среди лиц, они помогают различать одно лицо от другого.

Таким образом работает распознаватель EigenFaces. Он рассматривает все образы людей в целом и пытается извлечь компоненты, которые являются релевантными и полезными, и отбрасывает остальное. Эти важные признаки называются главными компонентами.

Ниже приведено изображение, показывающее дисперсию (главные компоненты), извлеченную из списка лиц. Таким образом, распознаватель EigenFaces обучает себя извлечением основных компонентов, но также ведет учет того, какие из них принадлежат кому-либо. Таким образом, всякий раз, когда вы вводите новое изображение в алгоритм, он повторяет такой процесс:

* Извлеките основные компоненты из нового изображения.
* Сравнить эти признаки со списком элементов, хранящихся во время обучения.
* Найдите тех, у кого лучшее совпадение.
* Верните метку «person», связанную с компонентой наилучшего соответствия.

Однако следует отметить, что алгоритм EigenFaces также рассматривает освещение как важную функцию. Вследствие этого свет и тени захватываются EigenFaces, который классифицирует их как «лицо».

**FisherFaces**

Этот алгоритм является улучшенной версией предыдущего. Как мы только что видели, EigenFaces просматривает все лица из обучающей выборки сразу и находит главные компоненты из всех из них в совокупности. Делая это, он не фокусируется на признаках, которые различают одного человека от другого. Вместо этого он концентрируется на тех, которые представляют все лица всех людей в данных обучения, в целом.

Но вот *загвоздка*. Рассмотрим изменения освещения следующих изображений: (рис).

Поскольку EigenFaces также считает освещенность полезной компонентой, он найдет эту вариацию очень актуальной для распознавания лиц и может отказаться от особенностей лиц других людей, считая их менее полезными. В итоге главные компоненты, которую EigenFaces извлекла, представляет только черты лица одного человека.

**Как исправить этот вопрос?**

Мы можем это сделать, настроив EigenFaces так, чтобы он извлекал полезные функции из лиц каждого человека отдельно, а не извлекал их из всех совмещенных лиц. Таким образом, даже если один человек имеет большие изменения освещения, это не повлияет на процесс извлечения компонентов других людей.

Точно так же алгоритм распознавания лица FisherFaces извлекает основные компоненты, которые отличают одного человека от других. В этом смысле компоненты индивида не доминируют (становятся более полезными) над другими.

Ниже приведено изображение главных компонент с использованием алгоритма FisherFaces.(рис)

Здесь следует отметить, что FisherFaces только препятствует тому, чтобы один человек стал доминирующим, но по-прежнему считает изменения освещения полезными. Мы знаем, что вариация света не является полезной функцией для извлечения, поскольку она не является частью фактического лица.

**Тогда, как избавиться от этой проблемы?**

**Local Binary Patterns Histograms (LBPH)** (Локальные бинарные гистограммы шаблонов)

Итак, мы знаем, что на Eigenfaces и Fisherfaces влияют свет, и в реальной жизни мы не можем гарантировать идеальные условия освещения. LBPH распознаватель является оптимизацией для преодоления этого недостатка.

Идея с LBPH заключается не в том, чтобы смотреть на изображение в целом, а вместо этого пытаться найти его локальную структуру, сравнивая каждый пиксель с соседними пикселями.

В конце концов, у вас будет одна гистограмма для каждого лица в наборе данных обучения. Это означает, что если в наборе данных обучения было 100 изображений, тогда LBPH будет извлекать 100 гистограмм после тренировки и хранить их для последующего распознавания. Помните, что алгоритм также отслеживает, какая гистограмма принадлежит какому человеку.

Позже в процессе распознавания процесс выглядит следующим образом:

* Подайте новое изображение распознавателю для распознавания лиц.
* Распознаватель генерирует гистограмму для этого нового изображения.
* Затем он сравнивает эту гистограмму с уже имеющимися гистограммами.
* Наконец, он находит лучшее совпадение и возвращает метку пользователя, связанную с этим лучшим совпадением.

Ниже представлена группа лиц и их локальных образов двоичных образов. Вы можете видеть, что на поверхности LBP не влияют изменения условий освещения:(рис)

**Dlib**

Библиотека написана на C++, но у нее есть Python API, который мы и будем использовать.

Для поиска лиц используется обученный HOG-каскад.

Основной идее такого алгоритма является допущение, что внешний вид и форма объекта на участке изображения могут быть описаны распределением градиентов интенсивности или направлением краев. Реализация этих дескрипторов может быть произведена путём разделения изображения на маленькие связные области, именуемые ячейками, и расчетом для каждой ячейки гистограммы направлений градиентов или направлений краев для пикселей, находящихся внутри ячейки. Комбинация этих гистограмм и является дескриптором. Для увеличения точности локальные гистограммы подвергаются нормализации по контрасту. С этой целью вычисляется мера интенсивности на большем фрагменте изображения, который называется блоком, и полученное значение используется для нормализации. Нормализованные дескрипторы обладают лучшей инвариантностью по отношению к освещению.

Итак, мы выделили лица на нашем изображении. Но теперь появляется проблема: одно и то же лицо, рассматриваемое с разных направлений, выглядит для компьютера совершенно по-разному. Чтобы учесть это, попробуем преобразовывать каждое изображение так, чтобы глаза и губы всегда находились на одном и том же месте изображения. Для этого используем алгоритм, называемый **«оценка антропометрических точек»**.  
Основная идея в том, что выделяется 68 специфических точек (меток), имеющихся на каждом лице, — выступающая часть подбородка, внешний край каждого глаза, внутренний край каждой брови и т.п. Затем происходит настройка алгоритма обучения машины на поиск этих 68 специфических точек на каждом лице.

Теперь, когда мы знаем, где находятся глаза и рот, мы будем просто вращать, масштабировать и [сдвигать](https://en.wikipedia.org/wiki/Shear_mapping#/media/File:VerticalShear_m%3D1.25.svg) изображение так, чтобы глаза и рот оказались отцентрованы как можно лучше. Мы не будем вводить какие-либо необычные 3D-деформации, поскольку они могут исказить изображение. Мы будет делать только базовые преобразования изображения, такие как вращение и масштабирование, которые сохраняют параллельность линий (т.н. [аффинные преобразования](https://en.wikipedia.org/wiki/Affine_transformation)):

Для решения задачи распознавания в dlib используется [сверточная нейронная сеть](https://youtu.be/52U4BG0ENiM), а именно предварительно обученная нейроннуя сеть ResNet. От сети отрезаются слои, отвечающие за классификацию, и остаются только сверточные слои, которые извлекают ключевые признаки из изображения. Результат работы - набор чисел, который называется дескриптором. dlib использует модифицированный вариант сети ResNet34. Эта сеть выдает дескрипторы из 128 чисел. Сеть обучена специальным образом так, чтобы дескрипторы фотографий одного человека находились рядом друг с другом, а дескрипторы фотографий разных людей - далеко друг от друга.

Чтобы оценить близость дескрипторов в dlib используется [Евклидово расстояние](https://ru.wikipedia.org/wiki/%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%BE%D0%B2%D0%B0_%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0). Если значение Евклидова расстояния между дескрипторами меньше 0.6, то считается, что на фотографиях один и тот же человек. С использованием такой метрики dlib обеспечивает точность 99.38% на тесте распознавания лиц [Labeled Faces in the Wild](http://vis-www.cs.umass.edu/lfw/).