МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

Кафедра автоматизированных систем и технологий

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

Дисциплина Технологии компьютерного зрения

# Лабораторная №9.

# Использование каскадов.

Выполнили студенты Копосов Лев Владимирович

Копосов Владимир Владимирович

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома

Вопросы:

1. **К какому преобразованию каскады Хаара НЕ являются инвариантными: перенос, масштаб, поворот?**

Каскады Хаара не являются инвариантными к преобразованию поворота.

1. **Зачем используется каскад классификаторов (каскад внимания)?**

Каскад классификаторов (каскад внимания) используется для ускорения и улучшения точности распознавания объектов. Он заключается в формировании цепочки из составных классификаторов, чтобы каждый последующий отбрасывал как можно больше отрицательных примеров, но при этом пропускал все или почти все положительные. Это позволяет ограничиться вычислением несложных составных классификаторов для большинства окон на изображении, так как для последующих более сложных, просто дело не дойдет.

1. **В чём заключается бустинг?**

Бустинг - методика, основанная на объединении набора слабых классификаторов в линейную комбинацию, которая работает как сильный классификатор. Каждый слабый классификатор должен быть лучше, чем случайный. Такими классификаторами могут быть как простые деления пространства признаков по осям, так и более сложные, например нейронные сети.

Бустинг - это итеративный процесс. На каждой итерации выбираются такие примеры, которые оказались трудными для предыдущих простых классификаторов, причем степень трудности зависит от величины правильного отклика. То есть выбирается такой простой классификатор, который хорошо распознает именно эти примеры. Итоговый составной классификатор будет являться линейной комбинацией простых классификаторов. Их отклики будут умножаться на коэффициенты и суммироваться

1. **Для чего используется параметр min\_neighbours?**

Параметр min\_neighbours используется для указания сколько соседей должен иметь каждый прямоугольник-кандидат, чтобы сохранить его. Более высокое значение приводит к меньшему количеству обнаружений, но с более высоким качеством. При применении методики скользящего окна происходит много ложных срабатываний, и чтобы исключить их и получить корректный результат, применяется подход соседства.

Вывод в консоль.

##### **Задание 1**

Создайте программу, использующую классификатор Виолы-Джонса для поиска лица на изображении.

Используйте заранее обученный каскад, поставляемый вместе с OpenCV. Получить путь к этому каскаду можно так:

cascade\_file = os.path.join(cv2.haarcascades, 'haarcascade\_frontalface\_default.xml')

Используйте класс **cv2.CascadeClassifier** для загрузки файла каскада и поиска целевого объекта на изображении. Используйте метод **.empty()**, чтобы убедиться что каскад был загружен.

Выведите на экран изображение, и нарисуйте прямоугольники вокруг каждого найденного лица.

По нажатию клавиш "W"/"S" должен изменяться параметр **min\_neighbours** (в диапазоне от 0 до 20, с шагом 1).

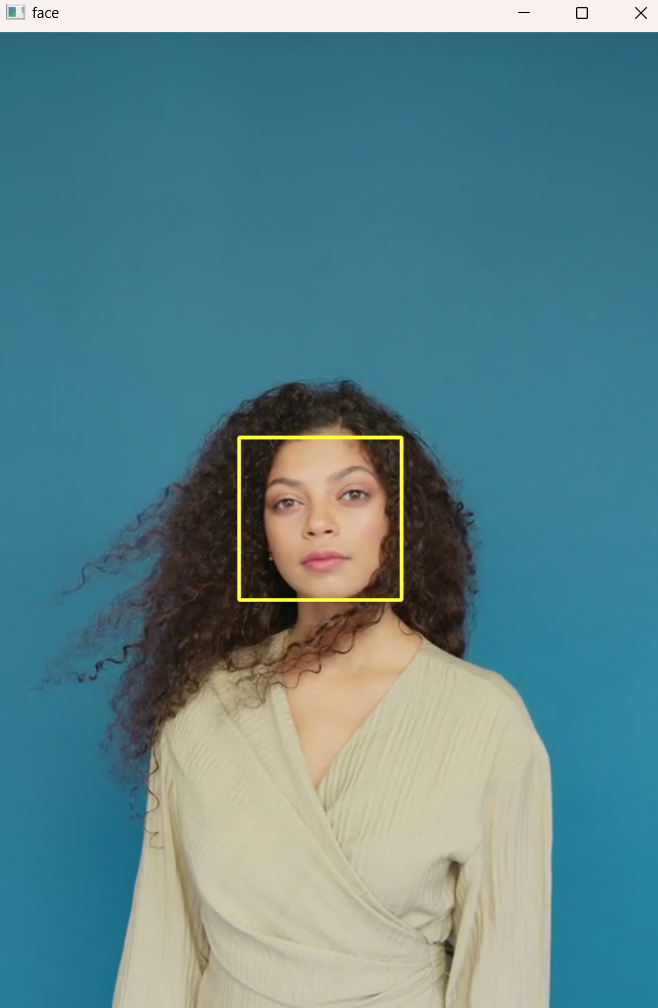
Приведите в отчёте скриншоты для малых и больших значений **min\_neighbours**.



##### **Задание 2**

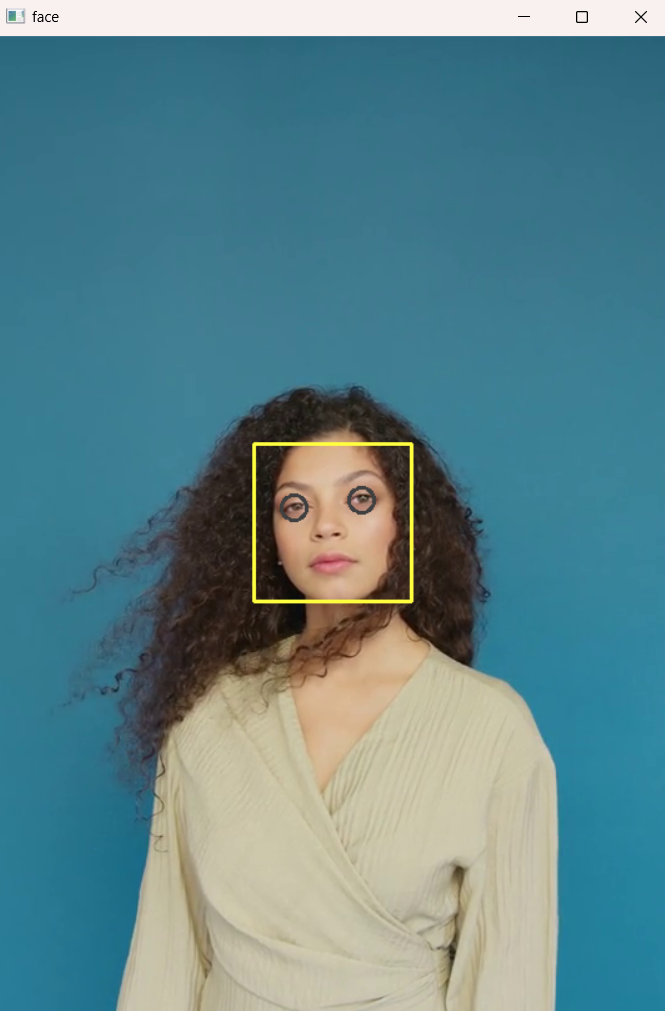
Модифицируйте программу из задания 1 так, чтобы она обрабатывала видеофайл.

В качестве источника видео используйте веб-камеру или приложенный к заданию файл **face.mp4**.



##### **Задание 3**

Модифицируйте программу из задания 3 следующим образом. Используя заранее обученный каскад **haarcascade\_eye.xml** из комплекта OpenCV, найдите глаза в области, отмеченной каскадом поиска лица, и обведите их эллипсами.



##### **Дополнительное задание**

Обучите ваш собственный каскад Хаара поиску объекта по одному изображению. Рекомендации по выполнению:

1. Загрузите набор утилит opencv, в частности **opencv\_createsamples** и **opencv\_traincascade**, а также необходимые им DLL-библиотеки. Распакуйте их в рабочий каталог в корне диска, без русских букв и пробелов в названии. Здесь и далее предполагается, что **C:\MyDirName** - ваш рабочий каталог.
2. Подготовьте фото вашего объекта. Используйте контрастный объект без движущихся частей, и желательно не дающий бликов. Для этой цели хорошо подходят логотипы.
3. Подготовьте отрицательные примеры (не менее 100 изображений). Для этого можно снять короткое видео помещения, затем написать программу, которая разделяет это видео на отдельные кадры. Рекомендуется поместить их в отдельный подкаталог **negatives**.
4. Подготовьте файлы, содержащие список файлов в этом каталоге. Их можно сгенерировать следующей парой команд в терминале:  
    **dir C:\MyDirName\negatives\\*.jpg /B /S >C:\MyDirName\negatives\negatives\_abs.txt** **dir C:\MyDirName\negatives\\*.jpg /B >C:\MyDirName\negatives\negatives\_rel.txt**
5. Создайте пустые подкаталоги **C:\MyDirName\positives** и **C:\MyDirName\training**. Поместите ваше изображение - положительный пример в каталог **C:\MyDirName\** и назовите его **positive.jpg**. В качестве положительного примера желательно разместить ваш объект на светлом фоне, если он тёмный, и наоборот.
6. Сгенерируйте положительные примеры с помощью утилиты **opencv\_createsamples**. Команда будет выглядеть *примерно* следующим образом:  
   **C:\MyDirName\opencv\_createsamples.exe -info positives\info.lst -img positive.jpg -bg negatives\negatives\_rel.txt -maxxangle 0.1 -maxyangle 0.1 -maxzangle 0.1 -bgcolor 0 -bgthresh 0 -w 50 -h 50 -num 100**Параметры:  
   -info - выходной файл.  
   -img - входное изображение - положительный пример  
   -bg - фоновые изображения. Положительный пример будет наложен на них (в оттенках серого).  
   -max?angle - допустимые углы поворота примера. Утилита выполнит перспективное преобразование примера перед наложением.  
   -bgcolor и -bgthresh задают яркость (среднее и диапазон изменения) для "прозрачного цвета". Например, если ваш объект на белом фоне, задайте эти параметры равными 240 и 15 (диапазон яркости 225-255). Задав оба параметра равными 0, вы отключите эту функцию и все цвета будут непрозрачными.  
   -w и -h задают минимальный размер для размещаемой копии вашего образца.  
   -num - количество примеров, которые стоит генерировать. Не должно превышать количество изображений в каталоге.  
   Утилита должна вывести ряд сообщений вида "Open background image", а в конце вывести "Done".
7. Сгенерируйте vec файл следующей командой:  
   **C:\MyDirName\opencv\_createsamples.exe -info positives\info.lst -num 100 -w 20 -h 20 -vec positives\positives.vec**где info.lst был сгенерирован в ходе пункта 6.  
   -w и -h задают минимальный размер для объекта, распознаваемого в ходе работы каскада.  
   -num - количество сгенерированных примеров. Столько же, сколько и в пункте 6.
8. Проведите обучение каскада с помощью утилиты **opencv\_traincascade**. Команда будет иметь примерно следующий вид:  
   **C:\MyDirName\opencv\_traincascade.exe -data training -vec positives\positives.vec -bg negatives\negatives\_abs.txt -numStages 100 -numPos 100 -numNeg 100 -featureType haar -w 20 -h 20 -minHitRate 0.999 -maxFalseAlarmRate 0.4 -precalcValBufSize 4048 -precalcIdxBufSize 4048 -numThreads 24 -acceptanceRatioBreakValue 10e-5**-data - каталог для рабочих данных, который вы создали в пункте 5.  
   -vec - индекс, который был создан в пункте 7.  
   -bg - файл со списком отрицательных примеров (используйте абсолютные пути!)  
   -numPos и -numNeg - количество положительных примеров (пункт 6) и отрицательных примеров (пункт 3). Для большого количества примеров (1000 и более) имеет смысл уменьшить numPos до 80-90% от общего числа примеров.  
   -numStages - максимальное количество этапов каскада. Итоговый каскад может содержать меньшее количество этапов.  
   -featureType - определяет тип признаков. Признаки Хаара обучаются медленнее, но зато более точны.  
   -w и -h задают минимальный размер для объекта, распознаваемого в ходе работы каскада. Должны строго совпадать с заданными в пункте 7.  
   -minHitRate и -maxFalseAlarmRate задают качество работы одного каскада.  
   -acceptanceRatioBreakValue определяет момент, когда каскад перестаёт обучаться.  
   -precalcValBufSize и -precalcIdxBufSize задают потребление памяти процессом.  
   -numThreads определяет число рабочих потоков.
9. По итогам обучения в каталоге **training** должен появиться файл **cascade.xml**, который можно загружать так, как это делалось в задании 2. Обратите внимание, что если вы хотите запустить обучение с начала, нужно очистить содержимое каталога **training**.
10. Используя вебкамеру или записанное видео с объектом, проверьте работу каскада аналогично заданию 2. Возможно, придётся понизить значение **min\_neighbours**.



