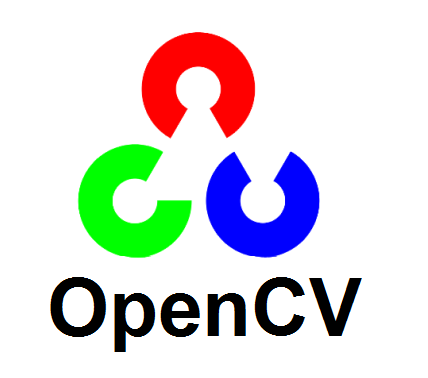
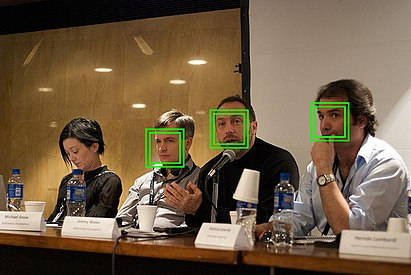
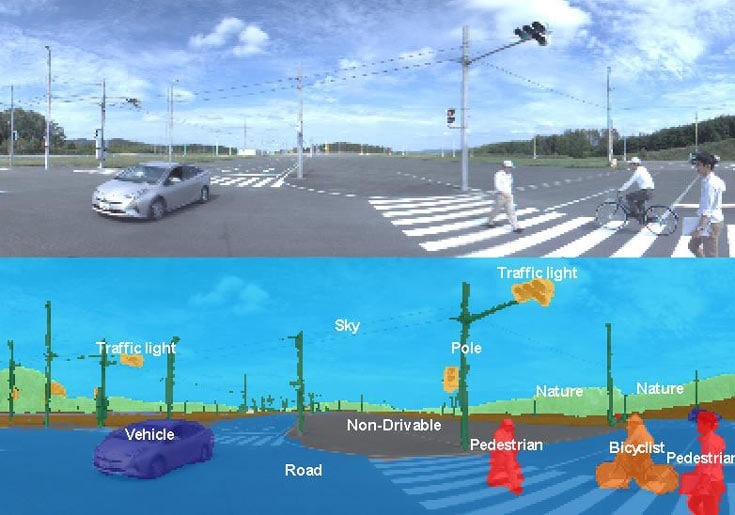
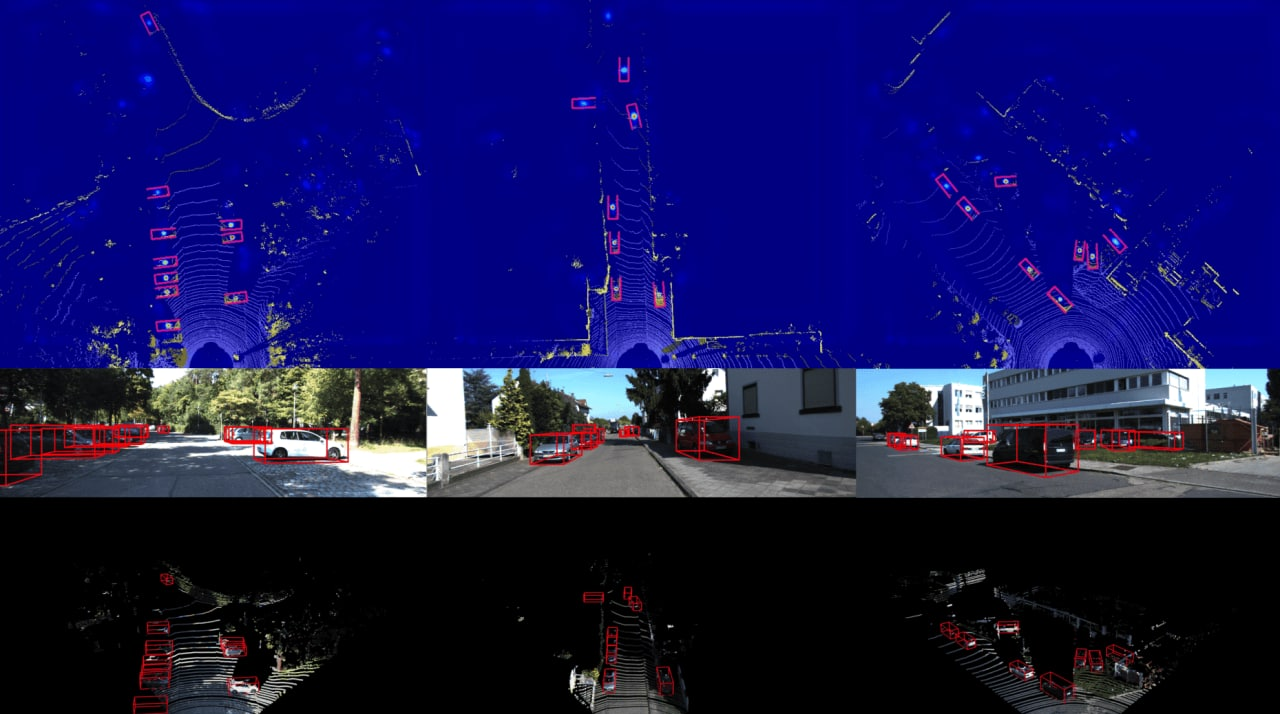
**OpenCV в эксплуатации, общие характеристики сопутствующих программ, а также краткий экскурс по работе с OpenCV.**

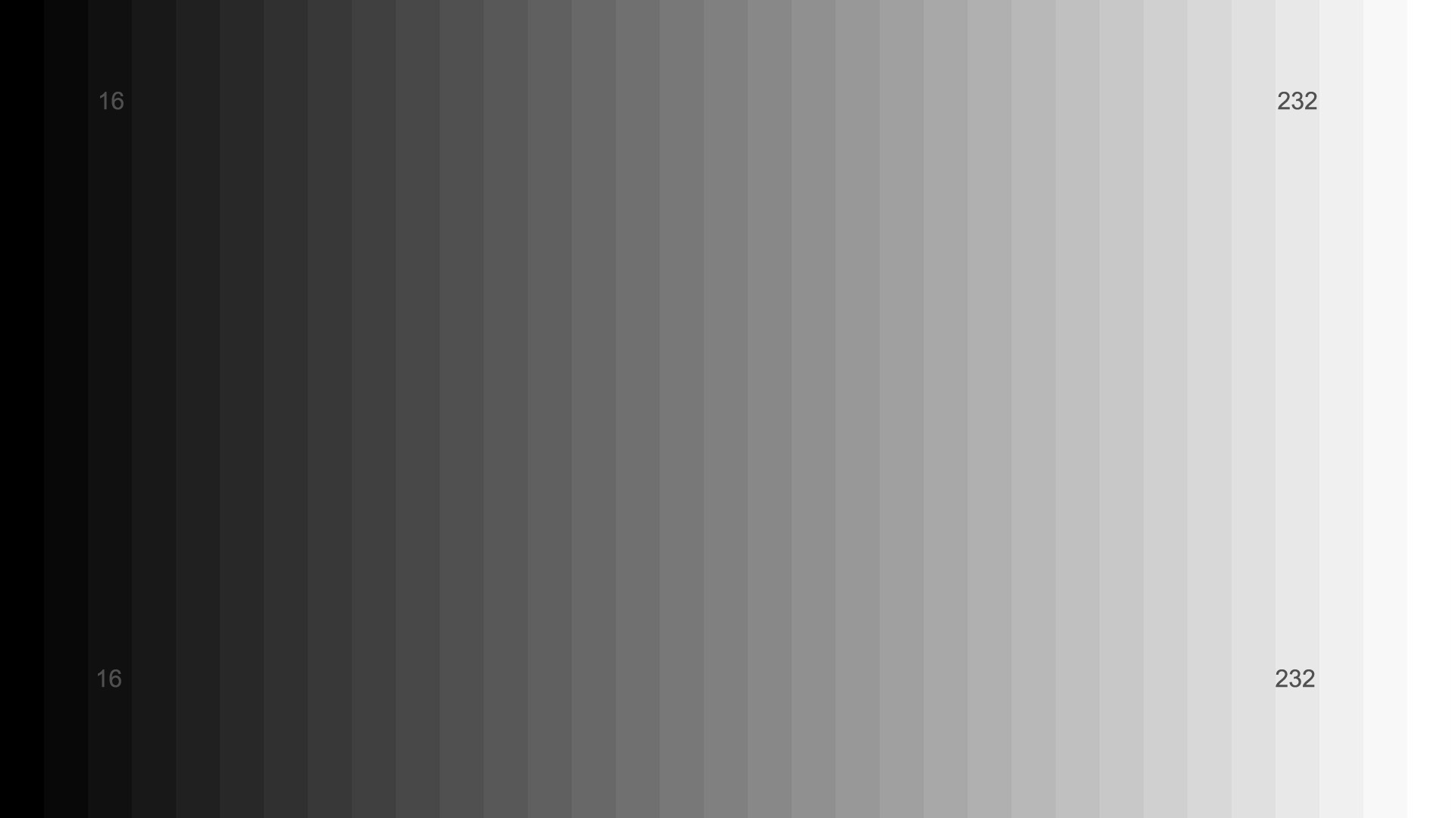


Давайте на конкретном примере разберемся, что такое OpenCV, с чем его едят и как его использовать в собственном проекте. С чем он совместим, а также некоторые сведения о программах, которые берут, прямое участие в самом процессе, считывания и передачи информации. OpenCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других. Тобишь она базируется на роспознавании обектов(обработка изображений). Как это происходит? Важно знать, что именно представляет собой обработка изображений и какова ее роль в общей картине, прежде чем углубляться в ее принципы. Обработка изображений чаще всего называется «Цифровая обработка изображений», а область, в которой она часто используется, - «Компьютерное зрение». Не смущайтесь - мы поговорим об этих терминах и о том, как они связаны. Оба алгоритма обработки изображений и алгоритмы Computer Vision (CV) принимают изображение в качестве входных данных; однако при обработке изображения вывод также является изображением, тогда как в компьютерном зрении выводом могут быть некоторые особенности / информация об изображении. Вот как выглядит считывание изображения с помощью компьютерного зрения:

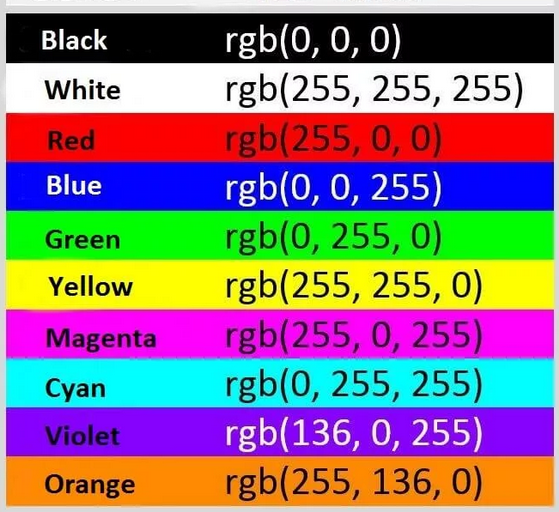
  

Почему именно OpenCV? Есть подводные камни при использовании, которые вы должны знать Прежде чем перейти к использованию обработки изображений в приложении, важно получить представление о том, какие операции попадают в эту категорию и как выполнять эти операции. Эти операции, наряду с другими, будут использоваться позже в наших приложениях. Перед тем как перейти к практике, нам нужно разобраться немного также с теорией о пискселях, ведь они являются фундаментом изображения, а позже перейдём к манипуляций с ними. Каждое изображение состоит из набора пикселей. Пиксель — это строительный блок изображения. Если представить изображение в виде сетки, то каждый квадрат в сетке содержит один пиксель, где точке с координатой ( 0, 0 ) соответствует верхний левый угол изображения. К примеру, представим, что у нас есть изображение с разрешением 400x300 пикселей. Это означает, что наша сетка состоит из 400 строк и 300 столбцов. В совокупности в нашем изображении есть 400\*300 = 120000 пикселей.

В большинстве изображений пиксели представлены двумя способами: в оттенках серого и в цветовом пространстве RGB. В изображениях в оттенках серого каждый пиксель имеет значение между 0 и 255, где 0 соответствует чёрному, а 255 соответствует белому. А значения между 0 и 255 принимают различные оттенки серого, где значения ближе к 0 более тёмные, а значения ближе к 255 более светлые:



Цветные пиксели обычно представлены в цветовом пространстве RGB(red, green, blue — красный, зелёный, синий), где одно значение для красной компоненты, одно для зелёной и одно для синей. Каждая из трёх компонент представлена целым числом в диапазоне от 0 до 255 включительно, которое указывает как «много» цвета содержится. Исходя из того, что каждая компонента представлена в диапазоне [0,255], то для того, чтобы представить насыщенность каждого цвета, нам будет достаточно 8-битного целого беззнакового числа. Затем мы объединяем значения всех трёх компонент в кортеж вида (красный, зеленый, синий). К примеру, чтобы получить белый цвет, каждая из компонент должна равняться 255: (255, 255, 255). Тогда, чтобы получить чёрный цвет, каждая из компонент должна быть равной 0: (0, 0, 0). Ниже приведены распространённые цвета, представленные в виде RGB кортежей:



Теперь немного практики, Первое, что нам необходимо сделать — это импортировать библиотеку. Есть несколько путей импорта, самый распространённый — это использовать выражение. Для загрузки изображения мы используем функцию cv2.imread(), где первым аргументом указывается путь к изображению, а вторым аргументом, который является необязательным, мы указываем, в каком цветовом пространстве мы хотим считать наше изображение. Чтобы считать изображение в RGB — cv2.IMREAD\_COLOR, в оттенках серого — cv2.IMREAD\_GRAYSCALE. По умолчанию данный аргумент принимает значение cv2.IMREAD\_COLOR. Данная функция возвращает 2D (для изображения в оттенках серого) либо 3D (для цветного изображения) массив NumPy. Форма массива для цветного изображения: высота x ширина x 3, где 3 — это байты, по одному байту на каждую из компонент. В изображениях в оттенках серого всё немного проще: высота x ширина. Вот пример изображения в оттенках серого и в цветном форматах:



Важно помнить, что у изображений в оттенках серого img. Share [2] будет недоступно, так как данные изображения представлены в виде 2D массива. Вот как можно манипулировать пикселями, с помощью OpenCV на Phyton.

Чтобы получить доступ к значению пикселя, нам просто нужно указать координаты x и y пикселя, который нас интересует. Также важно помнить, что библиотека OpenCV хранит каналы формата RGB в обратном порядке, в то время как мы думаем в терминах красного, зеленого и синего, то OpenCV хранит их в порядке синего, зеленого и красного цветов. Cначала мы берём пиксель, который расположен в точке (0,0). Данный пиксель, да и любой другой пиксель, представлены в виде кортежа. Заметьте, что название переменных расположены в порядке b, g и r. В следующей строке выводим значение каждого канала на экран. Как можно увидеть, доступ к значениям пикселей довольно прост, также просто можно и манипулировать значениями пикселей. Что ещё возможно изменить с изображением с помощью OpenCV. Полезной будет фича изменения изображения ведь нужно его подстроить под определенную модель. Первый метод, который мы изучим — это как поменять высоту и ширину у изображения. Для этого в OpenCV есть такая функция как resize (). Данная функция первым аргументом принимает изображение, размер которого мы хотим изменить, вторым — кортеж, который должен содержать в себе ширину и высоту для нового изображения, третьим — метод интерполяции(необязательный). Интерполяция — это алгоритм, который находит неизвестные промежуточные значения по имеющемуся набору известных значений. Фактически, это то, как будут заполняться новые пиксели при модификации размера изображения. К примеру, интерполяция методом ближайшего соседа (cv2.INTER\_NEAREST) просто берёт для каждого пикселя итогового изображения один пиксель исходного, который наиболее близкий к его положению — это самый простой и быстрый способ. Наиболее предпочтительным методом интерполяции для сжатия изображения является cv2.INTER\_AREA, для увеличения — cv2.INTER\_LINEAR. От данного метода зависит качество конечного изображения, но как показывает практика, если мы уменьшаем/увеличиваем изображение меньше, чем в 1.5 раза, то не важно каким методом интерполяции мы воспользовались — качество будет схожим.

Важно!

При изменении размера изображения учитывать соотношение сторон. Соотношение сторон — это пропорциональное соотношение ширины и высоты изображения. Если мы забудем о данном понятии, то получим изображение такого плана:



Также, стоит отметить перемещение изображения в любом двухмерном векторном пространстве, с помощью функции cv2.warpAffine(). Возможно вырезать, определенный фрагмент. Для того, чтобы вырезать интересующий вас фрагмент из изображения, достаточно воспользоваться следующим кодом:

def cropping():

crop\_img = img[10:450, 300:750]

Выполнив код выше, мы увидим, что обрезали лицо девочке:

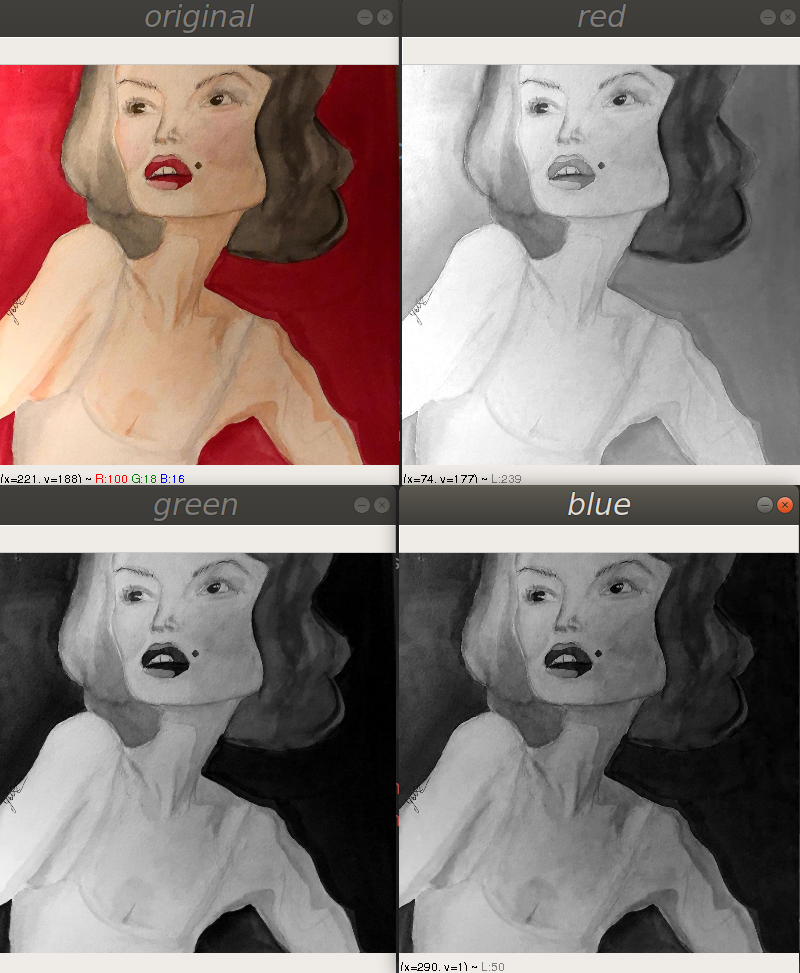


Доступен, поворот изображения. Когда мы поворачиваем изображение, нам нужно указать, вокруг какой точки мы будем вращаться, именно это принимает первым аргументом функция cv2.getRotationMatrix2D (). В данном случае я указал центр изображения, однако opencv позволяет указать любую произвольную точку, вокруг которой вы захотите вращаться. Следующим аргументом данная функция принимает угол, на который мы хотим повернуть наше изображение, а последним аргументом — коэффициент масштабирования. Мы используем 0.6, то есть уменьшаем изображение на 40% для того, чтобы оно поместилось в кадр. Данная функция возвращает массив numpy, который мы передаём вторым аргументом в функцию cv2.warpAffine(). В итоге, у вас на экране должно отобразиться следующее изображение:



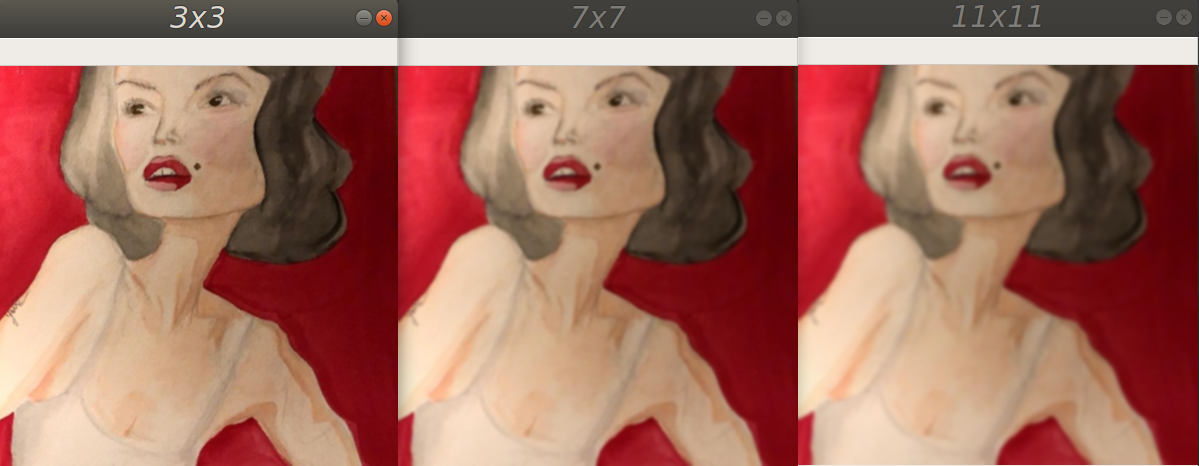
После того как мы освоили базовые вещи по работе с изображениями, можем перейти к более серьезным манипуляциям. К примеру: разбиение и слияние каналов, размытие, Averaging.

Как мы знаем, RGB изображение состоит из красной, зелёной и синих компонент. И что, если мы захотим разделить изображение на соответствующие компоненты? Для этого в OpenCV есть специальная функция — split (). Вот как это выглядит на той же картине с девушкой.

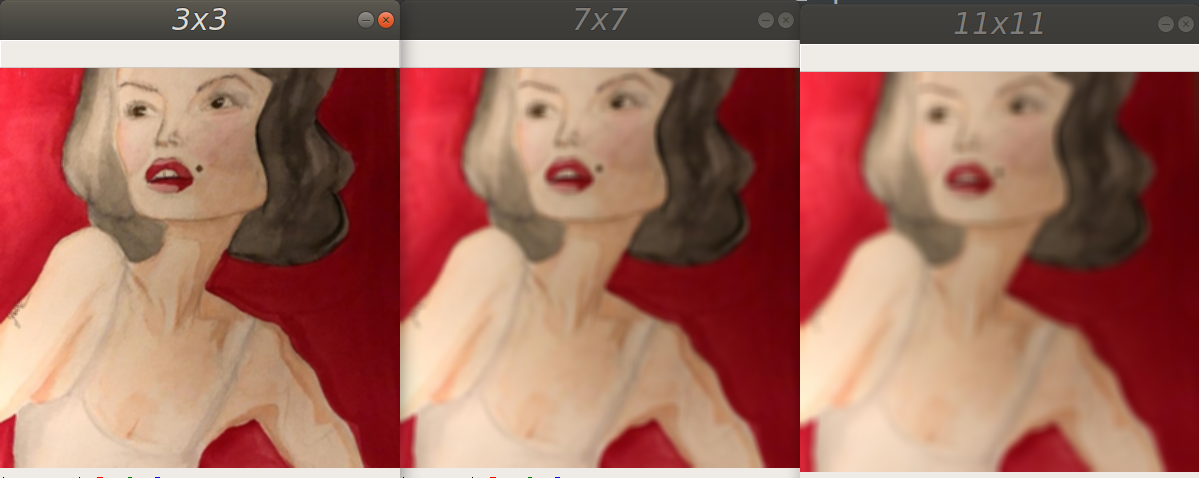


Красный цвет, наиболее яркий в связи с тем, что его количество преобладает, над синим и зелёным, в данном изображении. При более выраженном наличии этих цветов, изображения, были бы более светлого тона, а также уравновешенны.

Размытие — это когда более резкие области на изображении теряют свою детализацию, в результате чего изображение становится менее чётким. В opencv имеются следующие основные методы размытия: averaging(усреднённое), gaussian(гауссово) и median(медианное).

Гауссово − размытие похоже на предыдущее размытие, за исключением того, что вместо простого среднего мы теперь используем взвешенное среднее, где соседние пиксели, которые ближе к центральному пикселю, вносят больший «вклад» в среднее. Конечным результатом является то, что наше изображение размыто более естественно. Это размытие реализуется в opencv с помощью функции GaussianBlur (), которая принимает первые два аргумента такие же, как и предыдущая функция, а третьим аргументом указываем стандартное отклонение ядра Гаусса. Примерно такой результат мы получим: 

Усредненный фильтр, — это вычисление нового значения пикселя, при котором учитываются значения соседних пикселей. Ядро свёртки — это квадратная матрица, где пиксель в центре этой матрицы затем устанавливается как среднее значение всех других пикселей, окружающих его. Для того, чтобы воспользоваться данным размытием достаточно вызвать метод blur (). Пример данного эффекта можем увидеть ниже.

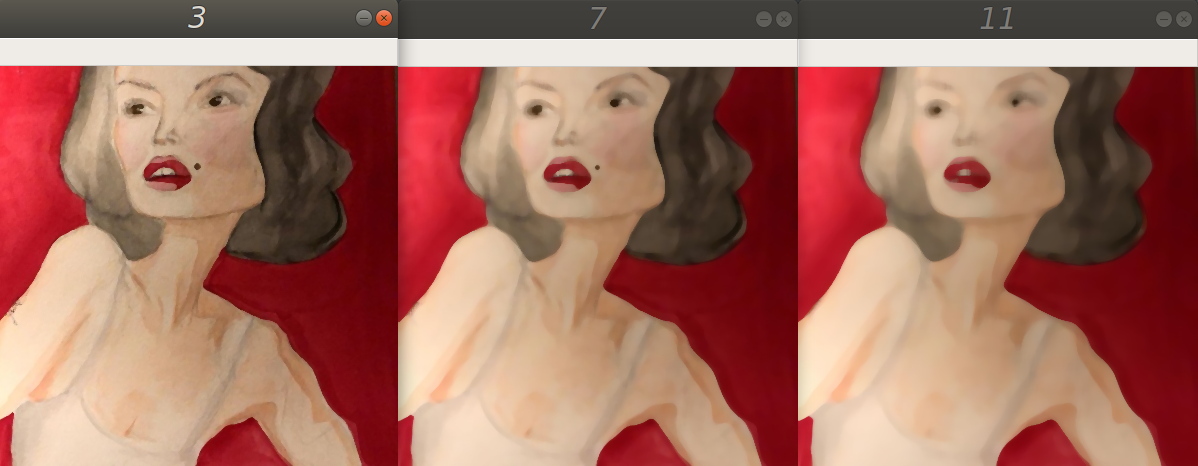


Заметьте!

Чем больше размер ядра, тем более размытым будет становиться изображение.

Медианное размытие, наиболее эффективно удаляет шум «соли». Базируется на замене центрального пикселя изображения на медиану всех пикселей в области ядра.

Вот, что мы получим при использовании данного эффекта.



Данная прога используется в данной игре. Все вышеупомянутые функции мы можем, использовать в нашем проекте. Но вот что, с помощью неё удалось сделать нам. Камера фиксированная жёстко также имеется прицел, который является точкой, которая распознает движущиеся объекты. Управляя моделью, в данном случае роботом, игрок, наводя на него камеру, видео аналитика делит его на две зоны, к примеру туловище и голова, подсвечивая их разными цветами. При этом используется Python+скрипты, а также OpenCV. В чем же заключается суть данной системы? Игрок наводит прицел на наиболее поражаемую точку на противнике, к примеру голова, туловище, ноги и т. п. Нажимает определенную клавишу. И тут вступает в действие наша видео аналитика, которая отслеживает, где именно будет происходить выстрел, та же голова, туловище и т. п. Данные отправляются на сервер, где они обрабатываются, и приходят назад с указанием того, что, выстрел был произведен именно в эту часть, на которую был наведён прицел, курсор, точка и т. д. Это приводит к, тому, что у противника отнимается определённое количество жизненных очков при чём, в зависимости от части тела количество очков разнится. К примеру, попадание в голову может отнять 30 очков, в туловище 15, в ту же ногу или руку 10. Данное действие имеет, является сигналом от вас и для противника. Всё это серверная часть которой нужен Apache HTTP-сервер. Проведём краткий экскурс что такое Apache HTTP-сервер. Apache HTTP-сервер − свободный веб-сервер. Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS. Основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользо Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Ядро (в отличие от модулей) полностью разрабатывается Apache Software Foundation, без участия сторонних программистов. Теоретически ядро apache может функционировать в чистом виде, без использования модулей. Однако функциональность такого решения крайне ограничена. Ядро Apache полностью написано на языке программирования C. вателей, модифицировать сообщения об ошибках и т. д. Поддерживает IPv4. Имеет собственный язык конфигурационных файлов, основанный на блоках директив. Практически все параметры ядра могут быть изменены через конфигурационные файлы, вплоть до управления MPM. Большая часть модулей имеет собственные параметры. Часть модулей использует в своей работе конфигурационные файлы операционной системы (например, /etc/passwd и /etc/hosts). Помимо этого, параметры могут быть заданы через ключи командной строки. С помощью него идёт взаимодействие серверной и клиентской части.

На этом пункте хотелось бы также заострить, внимание, ведь наличие серверной базы. Позволит вашему детищю, превратится с одно пользовательской в мультиплеерную игру. Концепция серверной части, такова, к примеру игрок посылает запрос нашему серверу, который добавляет новую группу в список доступных групп для походов, после чего остальные игроки имеют возможность получить этот список и присоединиться. Группа заполняется, и, нажатием кнопки, создатель отправляет ее в подземелье, новый мир или альтернативную ветку реальности на встречу приключениям. Логично предположить, что правильной реализацией такого взаимодействия был бы серверный «демон», который отвечал за состояние игрового мира, принимал запросы от игроков (в порядке очереди), обрабатывал их и отдавал результат. Пожалуй, самое интересное — это обработчик запросов.

Он содержит в себе основную логику нашего сервера:

— хранение состояния игроков (свободен, находится в подземелье, в группе и т. д.)

— хранение и обработка информации об игровом мире (битвы на арене, походы в подземелья, группы для похода)

Хранить состояние игрока нам необходимо для того, чтобы, например, пока персонаж занимается, майнингом лута, мы не смогли пойти на торговую площадку, на новое задание и т. д. Кроме того, на сервере должна производиться обработка игрового мира.

Обратите внимание!

Сервер не резиновый и большое количество игроков, приводит к его перегруженности. И в итоге к лагам.

В идеале сервер должен хранить текущее состояние игроков, чтобы иметь возможность корректно обрабатывать результаты.

Реализация, происходит на PHP 5.3 с расширением libevent, стоит отметить возможность создания отложенных событий (флаг EV\_TIMEOUT), что в нашем случае решает очень много проблем. Неплохо было бы, иметь ещё один «демон» с необходимым числом нитей, который будет выполнять всю кропотливую работу, да бы. Основной сервер не «слёг» от перегрузки.

А теперь о недостатках. Самый главный недостаток реализации: PHP течет. Даже новый сборщик мусора в 5.3 не решает всех проблем. И мы имеем:

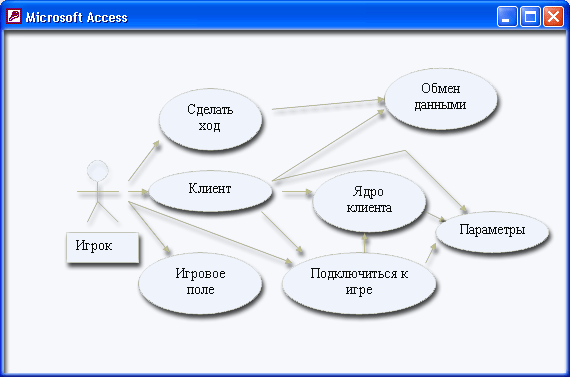
Состояние памяти на каждом worker

Решение: периодически перегружать worker'а, когда объем используемой памяти достигает определенного предела.

Клиентский сервер, это то с чем взаимодействуем мы с вами, делаем те или иные движения, улучшаем уровень, проходим задание и т. д. Все эти действия приходят на серверную часть где они обрабатываются. Взаимодействие клиента с сервером осуществляется методом посылки " серверу заявки.

Как же происходит обратная связь? Обратная связь сервера с клиентом осуществляется со стороны клиента и реализована в виде вызываемой периодически процедуры проверки наличия адресованных игроку сообщений. Сообщения от сервера делятся на две категории: информационные и управляющие. Управляющие сообщения, в отличие от информационных, не отображаются на экране, а приводят к изменению состояния клиента или выполнению клиентом определенных действий.

Схема взаимодействия клиентской и серверной связей, на примере "Игры в доминирование":



## Взаимодействие с HUD. В полном названии Heads-Up Display − часть визуального интерфейса игрока, отображающаяся на фоне виртуального игрового пространства в видеоигре. Название происходит от аналогичного в английском языке обозначения ИЛС. Часто элементы HUD располагаются по периметру экрана и (или) полупрозрачны, чтобы не мешать основному процессу игры. Иногда, в угоду реализму, информация, обычно отображаемая в HUD, маскируется вместо этого как часть окружения. Например, когда игрок управляет автомобилем, способным выдержать определенное количество ударов, может появиться дым или небольшой огонь, когда автомобиль почти разбит, чтобы указать, что следующее повреждение будет смертельным. Раненые персонажи также могут хромать, шататься, сутулиться или тяжело дышать, чтобы показать, что они ранены, ярким примером является Resident Evil 2, Tomb Raider (2013), Mirror's Edge, Dead Space и другие. Пример HUD на [S.T.A.L.K.E.R.: Shadow of Chernobyl](https://ru.wikipedia.org/wiki/S.T.A.L.K.E.R.:_Shadow_of_Chernobyl):

## 

Само поле, где будут происходить, все действия будут интерактивным. На нём будет не малое количество разнообразных точек, которые будут играть роль аптечек, пополнения боеприпасов, защиты и других полезных функций. Пишется скрипт, который дает данным точкам определенные координаты. Все данные точки распознаются с помощью OpenCV. Все они расставляются через серверную часть.

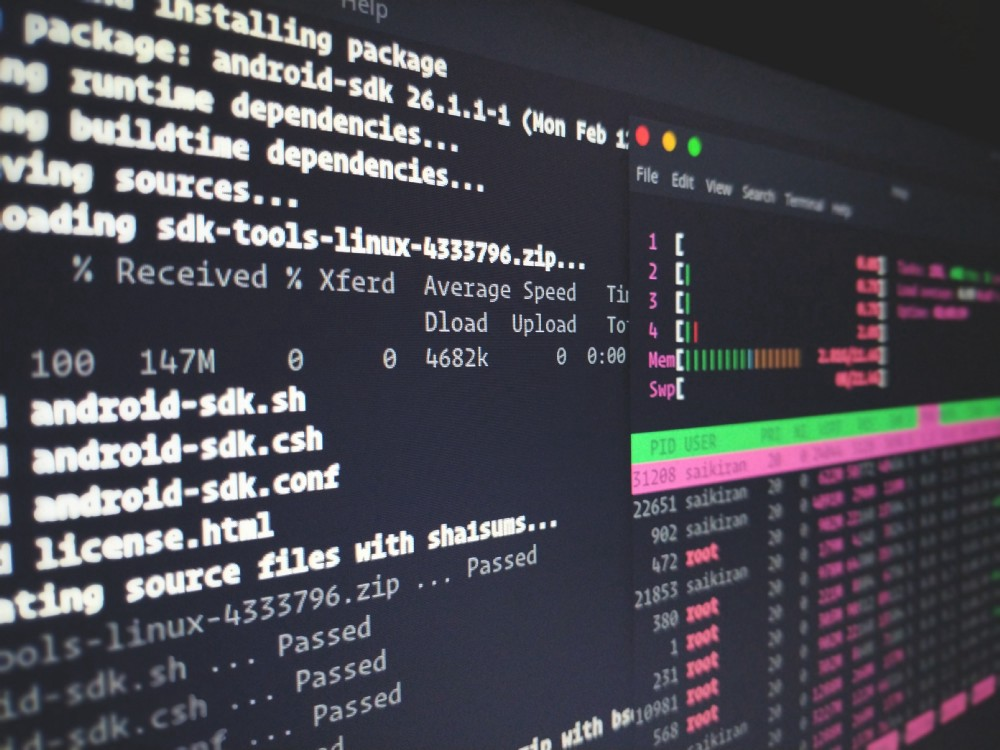


Почему, именно Python используется в данной конфигурации, с OpenCV? Всё достаточно просто он является самым универсальным языком программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью. Необычной особенностью языка является выделение блоков кода пробельными отступами. Что может отпугнуть некоторых из разработчиков. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как Си или C++. Является мультипарадигмальным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, мета программирование и функциональное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения. Поэтому вместо «присваивания значения переменной» лучше говорить о «связывании значения с некоторым именем». Язык программирования Python на своём пути стал наиболее популярным языком программирования. Это в особенности справедливо в области вычислительной обработки изображений, которая в большинстве случаев требует быстрого прототипирования и проектирования, а также значительных возможностей автоматизации. Как мы увидим в своём следующем разделе, цифровые изображения представлены двумерными и трёхмерными матрицами с тем чтобы компьютерам можно было бы их проще обрабатывать. В конечном счёте, большую часть времени цифровая обработка изображений в основном привлекает матричные вычисления. Многие библиотеки и модули Python не только предоставляют эффективные возможности матричных вычислений, но также и бесшовное взаимодействие с прочими библиотеками, которые обрабатывают чтение/ запись изображений.

Как мы уже знаем, и задачи автоматизации, и превращение их в совместно обрабатываемые, и то и другое являются сильными комплектами Python. Это делает Python первейшим кандидатом для реализации ваших приложений обработки изображений.

Технология распознавания видео позволяет локализоваться в пространстве, можно привязывать к этим координатам некий виртуальный объект. То есть использовать ее для конструирования объектов в виртуальной реальности. Технология распознавания лиц позволит сделать мимику персонажей максимально живой, а также позволит не приглашать, звёзд для игры того или иного персонажа. Отслеживание выстрелов связанны с видео аналитикой, которая отслеживает его, в реальном времени, и передает на обработку серверам.

Итак, мы с вами подходим, к логическому концу. И нам нужно запустить наш проект.



Для этого есть, скрипты, которые являются простыми последовательности команд, которые компьютеру нужно выполнить. Часто языки, которые максимально упрощают написание скриптов (как говорят, "скриптование") и их запуск, называют "скриптовыми языками" или же "языками для написания сценариев" (слово "script" на русский переводится как "сценарий").

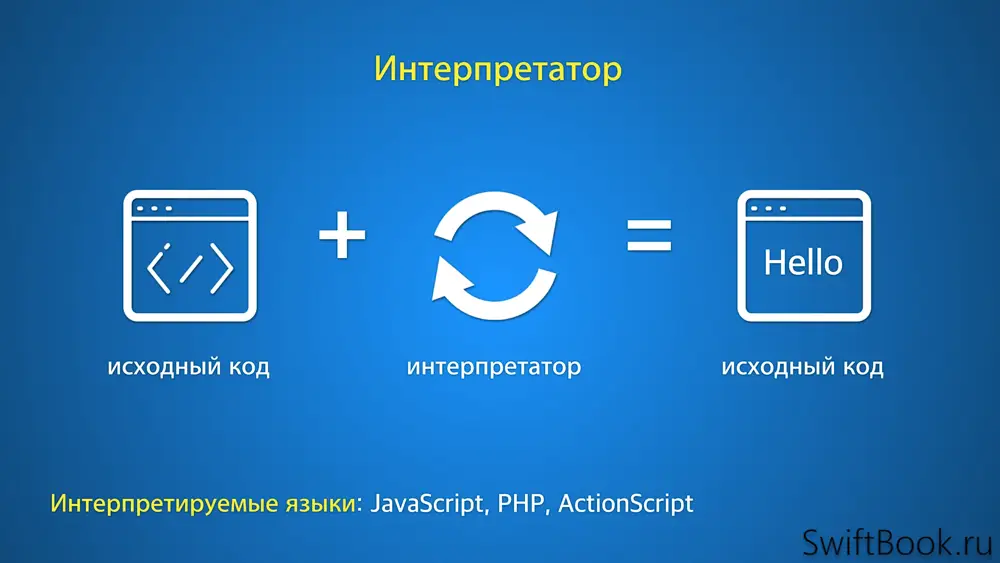
Python отлично подходит на роль скриптового языка: последовательность команд в простых сценариях не нужно никак оформлять и запускать скрипты максимально просто — мы просто пишем команды одну за другой в файл:

# file <script.py>

print('Hello, world!')

print('This is a python-script!')

Также, стоит помнить о хорошем интерпретаторе. Небольшая схема, которая покажет цепочку действий:



Он позволяет не внедрятся в систему вручную и компилировать в машинный код. Интерпретатор указывается специальной строкой в самой первой строчке файла скрипта, которая называется shebang, от названий первых двух символов такой строчки: # называется "sharp", а ! – "bang!".

Эта простота использования вместе с большим количеством полезных модулей и функций, входящих в поставку Python, делают последний хорошим инструментом для автоматизации различных задач, которые не хочется выполнять вручную при работе на компьютере. shebang — штука довольно простая, когда интерпретатор в системе ровно один. Но мы с вами знаем, что версий Python в системе может быть установлено несколько. Эта программа умеет находить и запускать программы с учётом переменных окружения и, т.к. при активации виртуального окружения модифицируется переменная $PATH, то env будет запускать именно ту версию интерпретатора, которая нам нужна (она просто найдётся раньше, т.к. путь до исполняемых файлов окружения добавляется в начало $PATH).