Оглавление

[Введение 5](#_Toc495350137)

[1 Обзор предметной области 7](#_Toc495350138)

[1.1 Алгоритмы распознавания, основанные на дескрипторах ключевых точек 7](#_Toc495350139)

[1.2 Детектор Харриса 9](#_Toc495350140)

[1.3 Детектор Ши-Томаси 11](#_Toc495350141)

[1.4 Детектор SIFT 11](#_Toc495350142)

[2 Разработка системы 13](#_Toc495350143)

[2.1 Подготовка рабочей среды 13](#_Toc495350144)

[2.1.1 Установка операционной системы 13](#_Toc495350145)

[2.1.2 Настройка удаленного доступа 16](#_Toc495350146)

[2.1.3 Установка OpenCV 19](#_Toc495350147)

[2.2 Подготовка данных для тренировки 28](#_Toc495350148)

[2.3 Тренировка каскада 32](#_Toc495350149)

[3 Выводы 34](#_Toc495350150)

[Заключение 34](#_Toc495350151)

[Список литературы 36](#_Toc495350152)

[Приложение А 37](#_Toc495350153)

[Приложение Б 40](#_Toc495350154)

# Введение

Зрение является самым мощным из наших органов чувств. Оно снабжает нас поразительно большим объемом информации о тот, что нас окружает, и дает возможность свободно взаимодействовать с внешним миром, причем все это без непосредственного физического контакта. Благодаря ему мы узнаем расположение объектов, идентифицируем их, соотносим их друг с другом. Неудивительно, что, как только цифровые вычислительные машины оказались достаточно доступными, начали предприниматься попытки наделит их способностью «видеть» [1].

Обработка изображений является одним из важнейших направлений применения современной вычислительной техники. Одной из распространенных задач обработки изображений являются фильтрация и восстановление изображений, сегментация, а также средства сжатия информации. На место классических задач по распознаванию фигур заданной формы на изображении приходят новые задачи распознавания линий и углов на изображении, распознавания краев изображений. Сегодня, с каждым днем появляется всё больше данных, которые уже некогда нельзя будет обработать человеческим трудом, поэтому темы обработки данных, распознавание образов и машинное обучение являются актуальными темами.

По мнению авторов [2], не существует общепринятой точки зрения, где заканчивается обработка изображений и начинаются другие смежные области, такие как анализ изображений и машинное зрение. Естественным этапом перехода от обработки изображений к их анализу выступает автоматическое или интерактивное распознавание отдельных областей или объектов на предварительно улучшенном изображении.

Целью данной работы является разработка системы анализа и классификации объектов на изображении при помощи алгоритмов, основанных на нахождении особых точек, уникальных характеристик объектов.

В ходе выполнения работы необходимо выполнить ряд задач:

* изучение методов нахождения ключевых точек;
* анализ алгоритмов, основанных на ключевых точках;
* построение информационной модели;
* реализация программы.

Объектом исследования является система анализа, обработки и нахождения ключевых точек на изображении.

Предметом исследования является система классификация образов на исходном изображении.

# 1 Обзор предметной области

## 1.1 Алгоритмы распознавания, основанные на дескрипторах ключевых точек

Распознавание образов в полном объеме машинным способом – это задача, которая до сих пор не решена, однако в рамках существенных ограничений, есть методы, позволяющие приблизится к ее решению. Проецируя задачу на человека, можно представить игру-головоломку «Пазл», в которой из мелких деталей нужно собрать цельную картину. Основной алгоритм заключается в том, чтобы найти конкретные шаблоны или уникальные особенности, которые легко отыскать и просто сравнить с эталонным изображением, и далее по собранным частям составить итоговую картину. Мы ищем уникальные части объекта на пазлах, и до завершения работы уже способны составить цельное представление о конечном результате. Что бы понять, как мы легко находим особые детали, достаточно взглянуть на следующее изображение (рисунок ):



Рисунок 1 – Характеристические признаки

В верхней части изображения даны шесть небольших клочков, задача –найти точное местоположение каждой части в исходном изображении.

A и B – плоские поверхности, распределенные во многих местах. Трудно найти точное местоположение каждого из этих обрывков.

C и D – задача становится намного проще. Части являются краями здания и можно найти приблизительное местоположение, но точное местоположение узнать все еще сложно. Таким образом, край – намного лучший характерный признак по сравнению с равномерной областью, но все равно недостаточно хорош.

И наконец, E и F – некоторые углы здания. Их можно легко узнать, т.к. в углах, как бы не перемещать и прикладывать к изображению этот кусочек, он все равно будет выглядеть иначе. Поэтому углы можно считать хорошим характеристическим признаком.

Для выделения из изображения некоторой информации необходимо привязаться к локальным особенностям изображения. На изображении возможно выделить особые (характеристические) точки. Характеристическая точка (точка интереса) – точка изображения, обладающая высокой локальной информативностью. В качестве численной меры информативности предлагаются различные формальные критерии, называемые операторами интереса. Оператор интереса должен обеспечивать достаточно точное позиционирование точки в плоскости снимка. Необходимо также, чтобы положение точки интереса обладало достаточной устойчивостью к фотометрическим и геометрическим искажениям изображения, включающим неравномерные изменения яркости, сдвиг, поворот, изменение масштаба, ракурсные искажения. Процесс определения особых точек достигается путем использования детектора и дескриптора.

Детектор – метод извлечения особых точек из изображения.

Дескриптор – идентификатор особой точки, выделяющий ее из остального множества особых точек. Дескрипторы должны обеспечивать инвариантность нахождения соответствия между особыми точками относительно аффинных преобразований изображений [3].

В 1992 Haralick и Shapir [4] выделили следующие требования к особым точкам в виде следующих свойств:

* Отличимость (distinctness) – особая точка должна явно выделяться на фоне и быть отличимой (уникальной) в своей окрестности.
* Инвариантность (invariance) – определение особой точки должно быть независимо к аффинным преобразованиям.
* Стабильность (stability) – определение особой точки должно быть устойчиво к шумам и ошибкам.
* Уникальность (uniqueness) – кроме локальной отличимости, особая точка должна обладать глобальной уникальностью для улучшения различимости повторяющихся паттернов.
* Интерпретируемость (interpretability) – особые точки должны определяться так, чтобы их можно было использовать для анализа соответствий и выявления интерпретируемой информации из изображения.

Подходы к определению особых точек могут быть разделены на 3 категории [4]:

1. Основанные на интенсивности изображения.
2. Использующие контуры изображения.
3. На основе использования модели с интенсивность в качестве параметров, которые подстраиваются к изображениям-шаблонам.

На практике для широкого применения наиболее распространены методы, основанные на интенсивности изображения.

# 1.2 Детектор Харриса

Можно заметить, что углы являются областями с большим изменением интенсивности во всех направлениях. Одна из первых попыток найти эти углы была сделана Крисом Харрисом и Майком Стивенсом в их статье «A Combined corner and edge detector» [5] в 1988 году. В основе лежит нахождение разности интенсивности для смещения по всем направлениям. Это выражается следующим образом:

*–* весовая функция (обычно используется функция Гаусса или бинарное окно).

Для максимизации функции для определения угла:

где:

*–* производные своих направлений.

Затем идет основная часть. Создается оценка, в которой определяется, может ли окно содержать угол или нет.

где:

– эмпирическая константа

­– собственные вектора.

Таким образом, если *R* мала, т.е. когда малы, то область однородная. Если *R* < 0, что происходит при или наоборот, то область является краем. Если *R* велика, что происходит, когда велики и , то область является углом (рисунок).

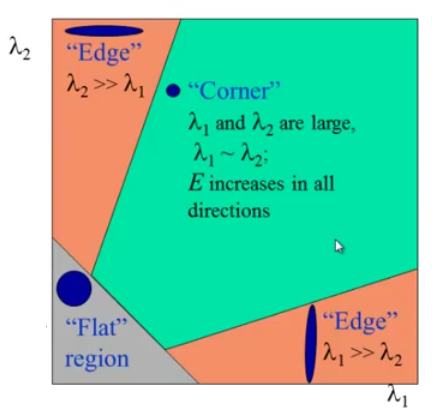


Рисунок 2 – Значения собственных векторов

# 1.3 Детектор Ши-Томаси

Детектор Ши-Томаси, на ряду с детектором Харриса, отличается тем, то мера отклика значения напрямую вычисляется следующей формулой:

В своей работе [6] авторы делают предположение, что поиск углов становится более стабильным.

# 1.4 Детектор SIFT

SIFT (Scale-Invariant Feature Transform) – алгоритм масштаба-независимого преобразования ключевых точке. Предыдущие алгоритмы инвариантны к любым вращениям изображения, т.к. те же углы так же остаются на повернутой картинке, однако при ее масштабировании, угол в маленьком окне может превратится в край (рисунок).

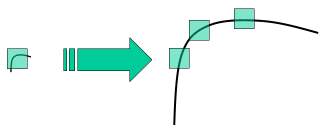


Рисунок 3 – Наложение фрейма на отмасштабированное изображение

В 2004 году, Давидом Ловем из университета British Columbia, в своей статье «Отличительное представление характеристик из масштаба-инвариантных ключевых точек» [7], был представлен новый алгоритм SIFT (Инвариантное масштабирование признаков). Алгоритм SIFT состоит из четырех основных шагов.

1. Обнаружение точек при масштабировании

Из вышеприведенного изображения очевидно, что мы не можем использовать одно и то же окно для обнаружения ключевых точек с разным масштабом. Все в порядке с маленькими углами, но для обнаружения больших углов нужны окна больших размеров. Для этого используется построение пирамиды гауссианов и разностей гауссианов:

где:

– значение гауссиана в точке с координатами (x, y)

– радиус размытия

– гауссово ядро

– значение исходного изображения

– операция свертки.

Difference of Gaussian (DoG):

(7)

Инвариантность достигается благодаря нахождению характеристических точек для первоначального изображения, взятого в разных масштабах. Строится пирамида гауссианов, разбивая все пространство на октавы (рисунок):

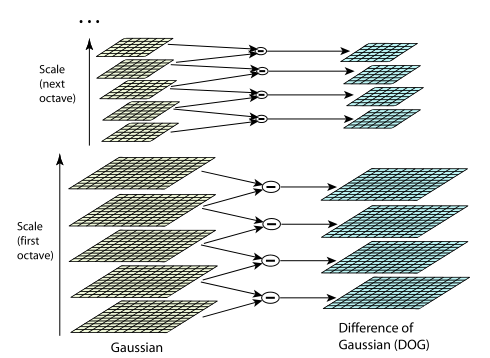


Рисунок 4 – Пирамиды преобразования гауссиана

Как только найдены разности гауссианов, ищутся локальные экстремумы в каждом из DoG изображений. Каждая точка, сравнивается с восемью соседними точками и девятью, расположенными выше и ниже уровнем. Наибольшая или наименьшая по значению точка принимается за экстремум (рисунок).

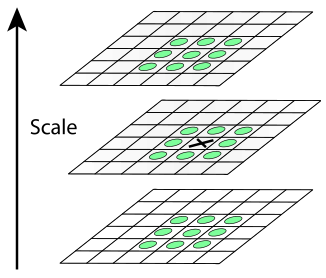


Рисунок 5 – DoG октавы

2. Уточнение ключевых точек

После обнаружения местоположений потенциальных ключевых точек их нужно уточнить, используя ряд Тейлора второго порядка. Если интенсивность на текущем экстремуме меньше порогового значения, то идет сдвижение к соседней точки. Если расчет оказался неудачным и произошел выход за пределы октавы, то точка отвергается. Также идет проверка, лежит ли точка на границе какого-то объекта или плохо освещена. Таким образом устраняются любые низко контрастные точки, оставляя только характеристические точки.

3. Выяснение ориентации ключевой точки

Далее необходимо определить ориентацию каждой ключевой точки для достижения инвариантности к вращению изображения. Берутся соседние с особой точи и вычисляется значение градиента (8) и его направление (9).

4. Формирование дескрипторов

Создается вектор направленных градиентов в окрестности ключевой точки. Берется матрица размером 16х16, окружающая ключевую точку. Далее блок разделяется на шестнадцать подблоков, каждый из которых состоит из восьми компонент гистограммы, покрывающей участок в 360 градусов. Таким образом образуется дескриптор размерности 128 компонент (рисунок).

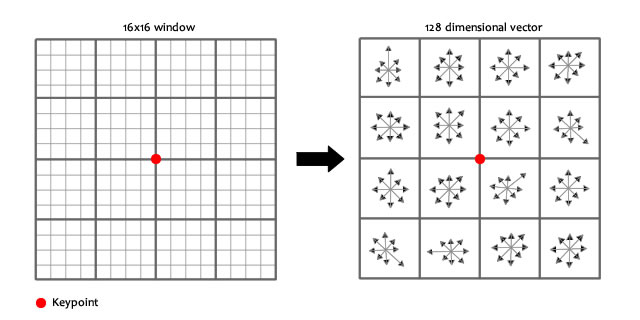


Рисунок 6 – SIFT Дескриптор

# 1.5 Детектор SURF

SURF (Speeded up Robust Features) – алгоритм, разработанный с целью ускорить поиск ключевых точек [9]. Обнаружение характеристических точек основано на вычислении детерминанта матрицы Гессе (гессиана):

где:

– значение текущего пикселя

После чего вычисляется градиент для каждой особой точки применяя фильтры Хаара [10] (рисунок).

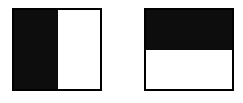


Рисунок 7 – Фильтры Хаара

# 2 Разработка системы

## 2.1 Подготовка рабочей среды

## 2.1.1 Установка операционной системы

Raspberry Pi 3 – одноплатный компьютер со следующими характеристикам (рисунок 13):

* 4х ядерный ARM процессор с частотой 1200 МГц,
* 1 Гб оперативной памяти,
* порты USB, HDMI, LAN, Jack 3.5 mm,
* поддерживается Bluetooth, WiFi;



Рисунок 8 –­ Raspberry Pi 3

Для установки операционной системы скачал образ операционной системы с официального сайта Raspberry Pi. Далее для прошивки операционной системы на SD – карту использовал программу SDFormatter V4.0 со следующими настройками (рисунок 14).

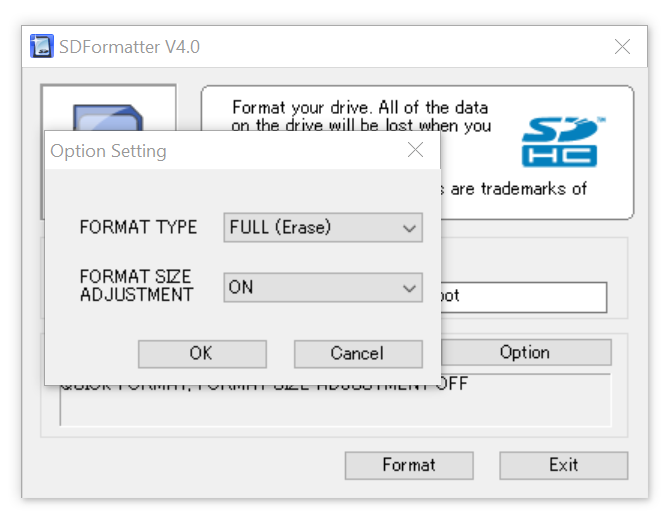


Рисунок 9 – SDFormater

После завершения передачи операционной системы на SD карту, ее следует вставить в задний порт Raspberry Pi, далее запустить, подсоединив устройство питания. При запуске, установил предложенную Raspbian OS (рисуноки 15 – 17).

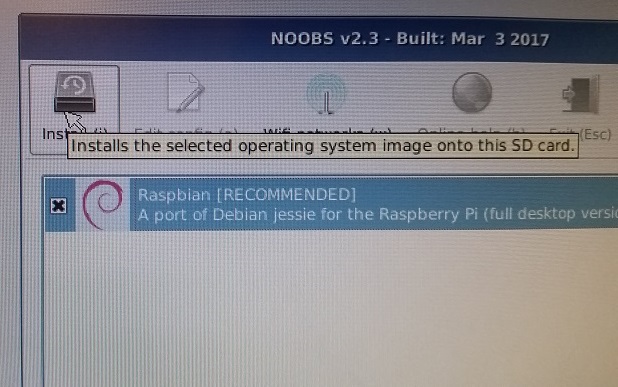


Рисунок 10 – Raspbian OS

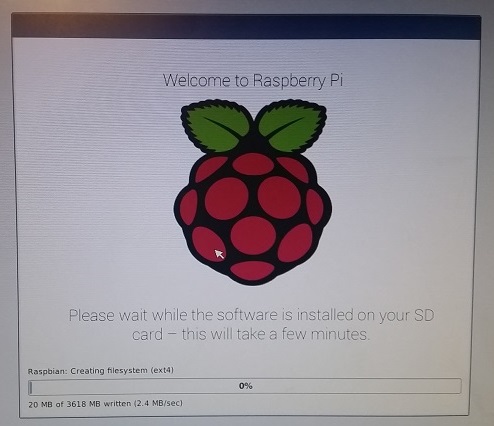


Рисунок 11 – Установка Raspbian OS

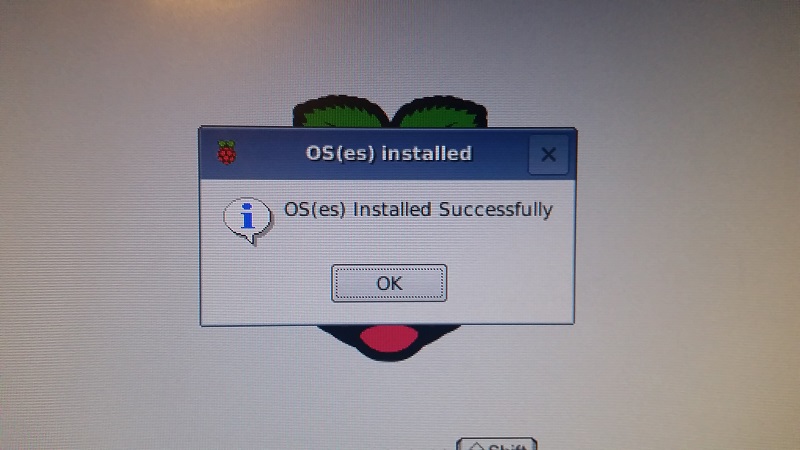


Рисунок 12 – Завершение установки

После перезагрузки компьютера, сперва подключился к домашней WiFi сети, далее сочетанием клавиш ctrl+alt+t запустил командную строку и ввел команды для обновления системы:

* sudo apt-get update
* sudo apt-get upgrade

## 2.1.2 Настройка удаленного доступа

Сперва запустил SSH сервера на Raspberry Pi. Для этого в командной строке ввел sudo raspi-config, после чего открылось меню настоек (рисунок).

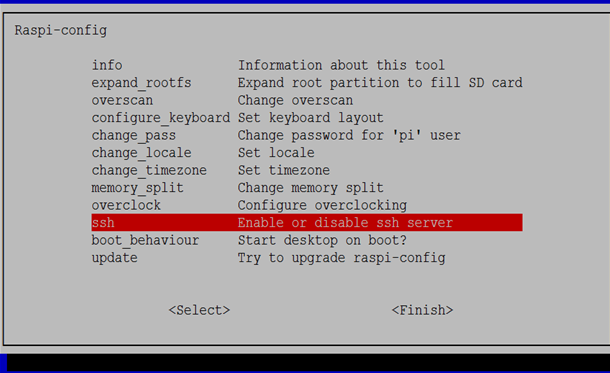


Рисунок 13 –Raspberry Pi

После чего перевел состояние ssh в enable (рисунок).



Рисунок 14 – SHH Ebable

Далее настроил программу PuTTY для SSH доступа к консоли Raspberry Pi из Windows. Сперва скачал дистрибутив программы с официального сайта. После запуска программы перешел в раздел Translation и выбрал кодировку UTF–8 (рисунок).

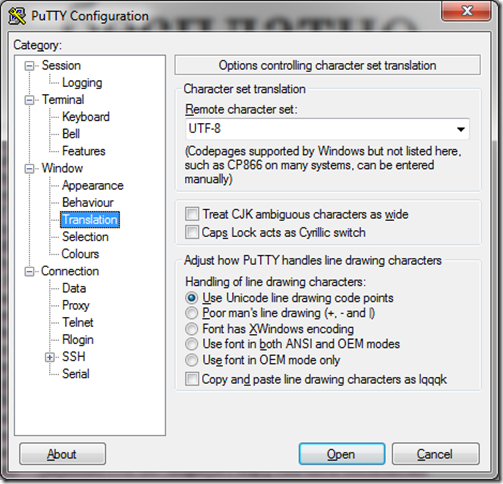


Рисунок 15 – PuTTY настройка кодировки

Используя программу Free IP Scanner, посмотрел список доступных устройств и их IP (рисунок).

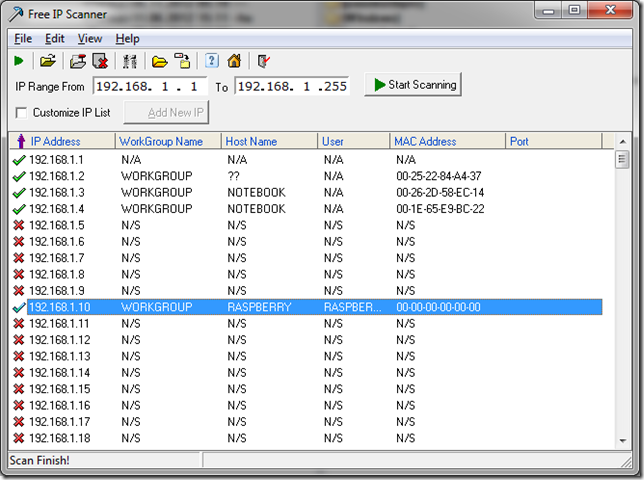


Рисунок 16 – Free IP Scanner

После чего в разделе Session ввел IP, Port сохранил настройки (рисунок).

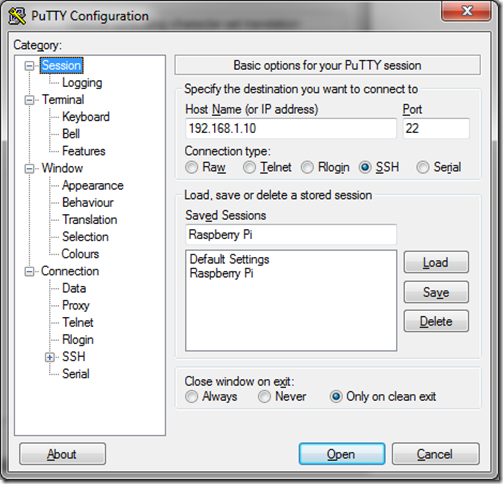


Рисунок 17 – PuTTY конфигурация сессии

Далее нажал кнопку Open и получил удаленный доступ к устройству (рисунок).

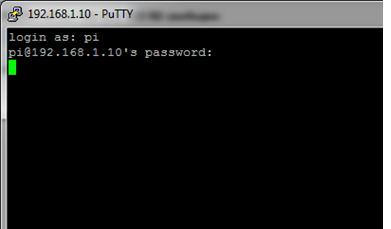


Рисунок 18 – Удаленный доступ

## 2.1.3 Установка OpenCV

OpenCV – это библиотека функций машинного зрения, позволяющая распознавать объекты окружающего мира.

Для полноценной работы с OpenCV обновил и установил новые пакеты. Сперва обновил систему:

* sudo apt-get update
* sudo apt-get upgrade

Далее установил в систему cmake, несколько пакетов для работы с изображениями и видео:

* sudo apt-get install build-essential cmake pkg-config
* sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
* sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv41-dev
* sudo apt-get install libxvidcore-dev libx264-dev

Установил пакеты для создания экранных форм, а также ускоренные операции над матрицами и заголовочные файлы языка python 2.7 и python 3:

* sudo apt-get install libgtk2.0-dev
* sudo apt-get install libatlas-base-dev gfortran
* sudo apt-get install python2.7-ev python3-dev
* После чего скачал архив с opencv в папку Download и распаковал его:
* cd ~/Downloads
* wget -O opencv.zip https://github.com/opencv/opencv/archive/master.zip
* unzip onpecv.zip

Далее скачал пакет с дополнительными функциями, не вошедшими в основной релиз:

* wget -O opencv\_contrib.zip https://github.com/opencv/opencv\_contrib/archive/master.zip
* unzip opencv\_contrib.zip

Перед установкой OpenCV, создал виртуальное окружение для изоляции проекта от системной версии и установленных пакетов python. Сперва установил менеджер пакетов pip:

* wget https://bootstrap.pypa.io/get-pip.py
* sudo python get-pip.py

Затем установил менеджер виртуального окружения:

* sudo pip install virtualenv virtualenvwrapper
* sudo rm -rf ~/.cache/pip

Далее добавил несколько строк в профиль пользователя:

* echo -e “\n# virtualenv and virtualenvwrapper” >> ~/.profile
* echo “export WORKON\_HOME=$HOME/.virtualenvs” >> ~/.profile
* echo “source /usr/local/bin/virtualenvwrapper.sh” >> ~/.profile

После чего перезагрузил Raspberry Pi, в командной строке ввел команду:

* source ~/.profile

Далее создал виртуальное окружение с именем «cv», запустил его и тут же установил в него математический пакет NumPy:

* mkvirtualenv cv -p python3
* workon cv
* pip install numpy

Находясь в виртуальном окружении, скомпилировал и установил OpenCV. Для этого зашёл в папку с исходным кодом библиотеки и выполнил команды:

* cd ~/Downloads/opencv-master
* mkdir build
* cd build
* cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

-D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D INSTALL\_PYTHON\_EXAMPLES=ON \

-D OPENCV\_EXTRA\_MODULES\_PATH=~ \

/Downloads/opencv\_contrib-master/modules \

-D BUILD\_EXAMPLES=ON ..

После подготовки компонентов собрал бинарные файлы командой:

* make -j2, где ключ -j2 обозначает количество ядер использующихся для сборки библиотеки

После нескольких часов сборки установил бинарный файл командой:

* sudo make install
* sudo ldconfig

Далее переименовал появившейся в папке python файл cv2.cpython-34m.so:

* cd /usr/local/lib/python3.4/site-packages/
* sudo mv cv2.cpython-34m.so cv2.so

Затем, чтобы пользоваться OpenCV, находясь в виртуальном окружении, сделал символьную ссылку на получившийся cv2.so файл:

* cd ~/.virtaulev/cv/lib/python3.4/site-packages/
* ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so

Для проверки, запустил виртуальное окружение, интерпретатор, импортировал библиотеку и проверил текущую версию (рисунок):

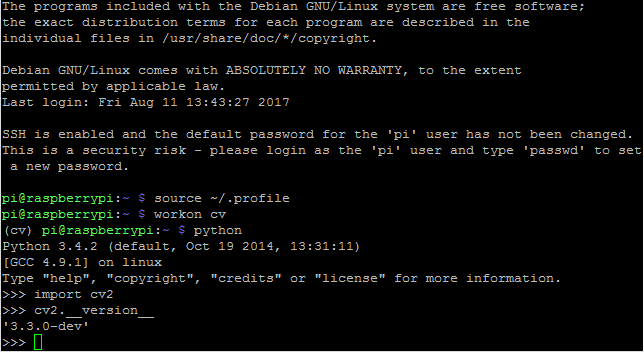


Рисунок 19 – PuTTY connection

## 2.2 Подготовка данных для тренировки

Для построения каскада необходима выборка «позитивных» изображений, на которых есть искомый объект и «негативных» на которых этого объекта нет. После чего необходимо из «позитивных» изображений сделать вектор поместив их в один файл. Обучение требует большой выборки изображений с объектом, но можно поступить следующим образом: взять несколько изображений объектов и поместить их поверх «негативных» изображений.

Для создания выборки «негативных» - фоновых изображений использовал сервис image-net.org позволяющий скачать множество тематических изображений. Т.к. вручную скачивание изображений порядка двух тысяч штук нецелесообразно, написал скрипт, позволяющий это сделать за меня, а также привести изображения в градации серого и уменьшив размер (рисунок 27).

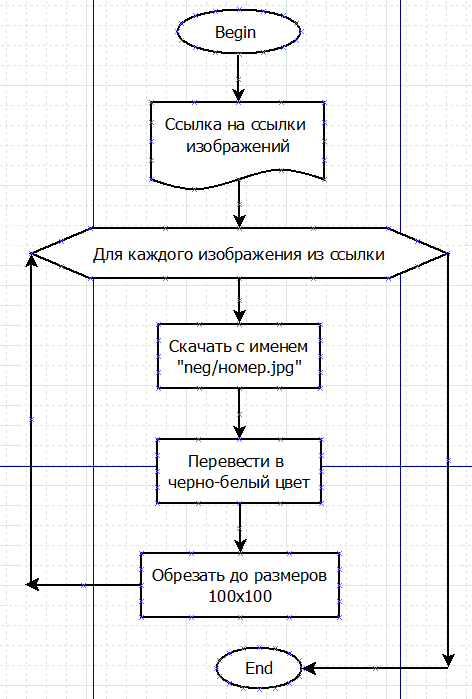


Рисунок 20- Блок-схема алгоритма скачивания изображений

В результате удалось скачать и обработать две тысячи семьсот тридцать пять изображений (рисунок 28).

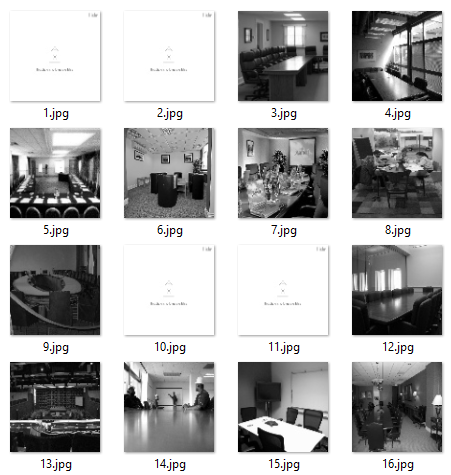


Рисунок 21 - Набор фоновых изображений

В получившуюся выборку также попали изображения с ошибкой доступа. Такие изображения могут плохо повлиять на тренировку каскада, поэтому они были удалены с помощью скрипта (рисунок 29).

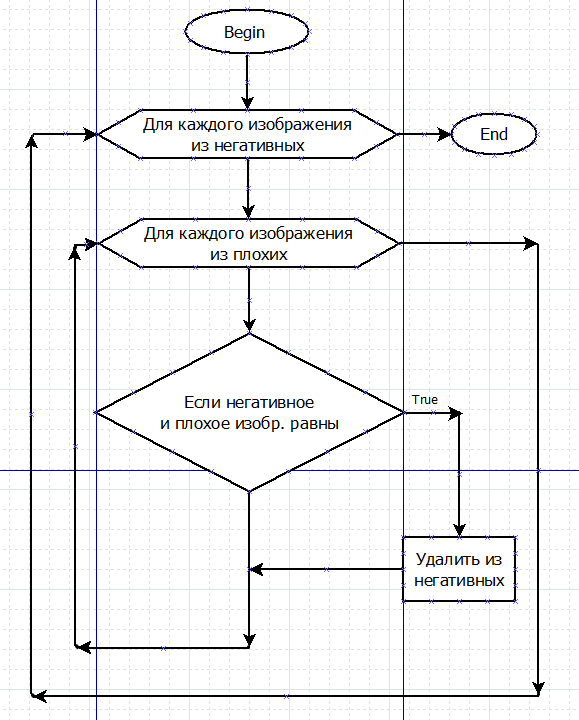


Рисунок 22 - Блок-схема скрипта удаляющего ошибочные изображения

В результате автоматически было отсеяно двести двенадцать изображений.

Далее подготовил набор изображений пяти различных гранат, сделал прозрачный фон и привел к размеру 30 на 50 пикселов (рисунок 30).

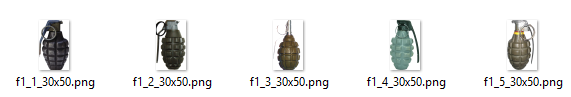


Рисунок 23 - Подготовленные изображения

Затем создал файл дескриптор. Для фоновых изображений файл представляет собой текстовый документ, в котором хранится список этих изображений.

Далее выполнив команду: opencv\_createsamples –img gr/f1\_1\_30x50.png -bg bg.txt -info info/info.lst -pngoutput info -maxxangle 0.5 -maxyangle 0.5 -maxzangle 0.5 -num 2537 создал позитивные изображения путём добавления позитивного изображения на фоновые (рисунок 31).



Рисунок 24 - Получившееся изображение

Также к этим изображениям сгенерировался файл дескриптор, где первое число – количество объектов на изображении, два вторых – левый угол начала координат, а остальные два – его ширина и высота (рисунок 32).

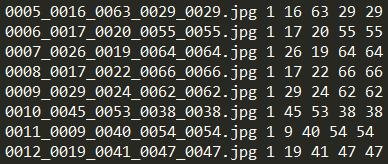


Рисунок 25 - Дескриптор получившихся изображений

## 2.3 Тренировка каскада

Подготовив данные командой: opencv\_ctraincascade –data data –vec positives.vec –bg bg.txt –numPos 2000 numNeg 1000 –numStages 11 –w 15 –h 25 –minhirate 0.98 запустил процесс тренировки каскада.

Далее, после прогонки 11 эпох, собрал получившиеся файлы в один каскад (рисунок 33).

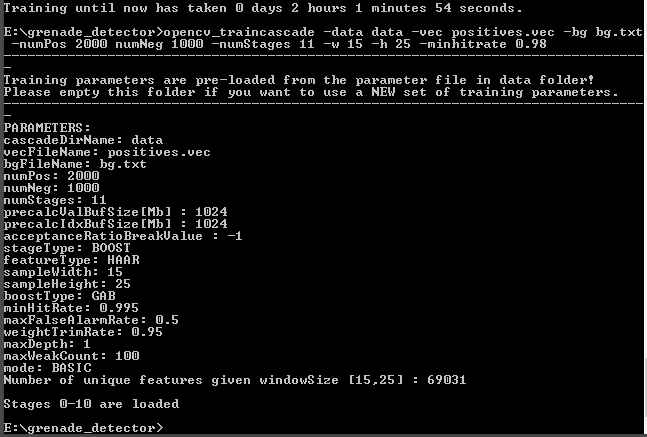


Рисунок 26 - Сборка итогового каскада

# 3 Выводы

Ознакомился и опробовал механизм распознавания образов методом Виолы – Джонса, использующий признаки Хаара. Воспользовавшись библиотекой OpenCV написал скрипты, позволяющие получить и обработать изображения, на основании которых в дальнейшем обучил каскад. Также ознакомился с механизмами распознавания образов с помощью нейронных сетей.

В результате использования натренированного мною каскада, на тестовом изображении было выделено 5 областей из которых одна область определила классифицирующийся объект, остальные выделенные области подходили по определенным в тренировочной выборке признакам (рисунок 34).

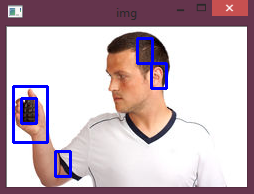


Рисунок 27 - Результат обработки

# Заключение

В ходе выполнения исследования были решены следующие задачи:

1) проанализированы алгоритмы классификации;

2) реализована система распознавания образа.

Таким образом, все поставленные задачи были решены и цель работы, заключающаяся в разработки системы по распознаванию образа, была достигнута.

В следующей работе планируется продолжить работу над улучшением точности и качества распознавания (рисунок35).

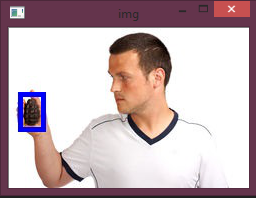


Рисунок 28 - Планируемый результат

# Список литературы

1. Хорн Б. К. П. Зрение роботов: Пер. с англ. – М.: Мир, 1989. – 487 с., ил. ISBN 5-03-000570-6.
2. Фисенко В. Т., Фисенко Т. Ю. Компьютерная обработка и распознавание изображений: учеб. пособие / В. Т. Фисенко. – СПб: СПбГУ ИТМО, 2008. – 195 с.
3. Конушин А. Слежение за точечными особенностями сцены (Point feature tracking). Компьютерная графика и мультимедиа. Выпуск №1(5)/2003.
4. V. Rodehorst, A. Koschan. Comparison and evaluation of feature point detectors, 2006.
5. Chris Harris, Mike Stephens. A Combined Corner and Edge Detector. Plessey Research Roke Manor, United Kingdom the Plessey Company plc. 1988.
6. Jianbo Shi, Carlo Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994.
7. Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision (2004) 60: 91.
8. Дж. Стокман, Л. Шапиро Компьютерное зрение. Пер. с анг. – М.: БИНОМ. Лаборатория знаний, 2006.
9. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gook. Speeded Up Robust Features. ETH Zurich, Katholieke Universiteit Leuven.
10. Дружков П. Н., Золотых Н. Ю., Половинкин А. Н. Параллельная реализация алгоритма предсказания с помощью модели градиентного бустинга деревьев решений. Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2011. No. 37 (254). С. 82-89.

# Приложение А

**(обязательное)**

**Исходный код обработки изображений на Python**

Листинг А.1 – Загрузка и обработка тестовых изображений

import urllib.request

import cv2

import numpy as np

import os

import sys

def store\_raw\_images():

try:

with open ('neg\_links.txt', 'r') as url\_file:

neg\_url\_list = url\_file.read().split()

url\_file.close()

except Exception as e:

print(str(e))

sys.exit(0)

pic\_num = 1

if not os.path.exists('neg'):

os.makedirs('neg')

for link in neg\_url\_list:

try:

neg\_image\_urls = urllib.request.urlopen(link).read().decode()

for i in neg\_image\_urls.split('\n'):

try:

print(i)

urllib.request.urlretrieve(

i,

"neg/"+str(pic\_num)+".jpg"

)

img = cv2.imread(

"neg/"+str(pic\_num)+".jpg",

cv2.IMREAD\_GRAYSCALE

)

resize\_image = cv2.resize(img, (100, 100))

cv2.imwrite(

"neg/"+str(pic\_num)+".jpg",

resize\_image

)

pic\_num += 1

if pic\_num == 2500:

print('2500 images was downloaded!')

sys.exit(0)

except Exception as e:

print(str(e))

except Exception as e:

print(str(e))

store\_raw\_images()

Листинг А.2 – Удаление неподходящих для обучения изображений

import urllib.request

import cv2

import numpy as np

import os

def find\_uglies():

match = False

for file\_type in ['neg']:

for img in os.listdir(file\_type):

for ugly in os.listdir('uglies'):

try:

current\_image\_path = str(file\_type)+'/'+str(img)

ugly = cv2.imread('uglies/'+str(ugly))

question = cv2.imread(current\_image\_path)

if ugly.shape == question.shape and not(np.bitwise\_xor(ugly,question).any()):

print('That is one ugly pic! Deleting!')

print(current\_image\_path)

os.remove(current\_image\_path)

except Exception as e:

print(str(e))

find\_uglies()

А.3 – Создание дескрипторов для позитивных и негативных изображений

import os

def create\_pos\_n\_neg():

for file\_type in ['neg']:

for img in os.listdir(file\_type):

if file\_type == 'pos':

line = file\_type+'/'+img+' 1 0 0 50 50\n'

with open('info.dat','a') as f:

f.write(line)

elif file\_type == 'neg':

line = file\_type+'/'+img+'\n'

with open('bg.txt','a') as f:

f.write(line)

create\_pos\_n\_neg()

# Приложение Б

**(обязательное)**

**Исходный код распознавания образа**

Листинг Б1.1 – Исходный код распознавания образа

import numpy as np

import cv2

import sys

grenade\_cascade = cv2.CascadeClassifier('cascade\_11.xml')

img = cv2.imread(sys.argv[1], 1)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

grenades = grenade\_cascade.detectMultiScale(gray, 1.5, 3)

for (x,y,w,h) in grenades:

cv2.rectangle(img,(x,y), (x+w,y+h), (255, 0, 0), 2)

cv2.imshow('img', img)

cv2.waitKey()