Оглавление

[Введение 5](#_Toc492847390)

[1 Обзор предметной области 7](#_Toc492847391)

[1.1 Алгоритмы распознавания, основанные на дескрипторах ключевых точе 7](#_Toc492847392)

[2 Разработка системы 8](#_Toc492847393)

[2.1 Подготовка рабочей среды 8](#_Toc492847394)

[2.1.1 Установка операционной системы 8](#_Toc492847395)

[2.1.2 Настройка удаленного доступа 11](#_Toc492847396)

[2.1.3 Установка OpenCV 14](#_Toc492847397)

[2.2 Подготовка данных для тренировки 23](#_Toc492847398)

[2.3 Тренировка каскада 27](#_Toc492847399)

[3 Выводы 29](#_Toc492847400)

[Заключение 29](#_Toc492847401)

[Список литературы 31](#_Toc492847402)

[Приложение А 32](#_Toc492847403)

[Приложение Б 35](#_Toc492847404)

# 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

# 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 1 –­ 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).

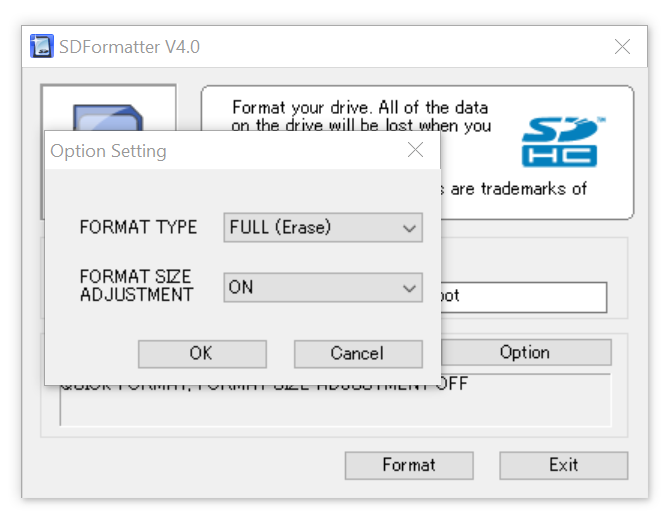


Рисунок 2 – SDFormater

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).

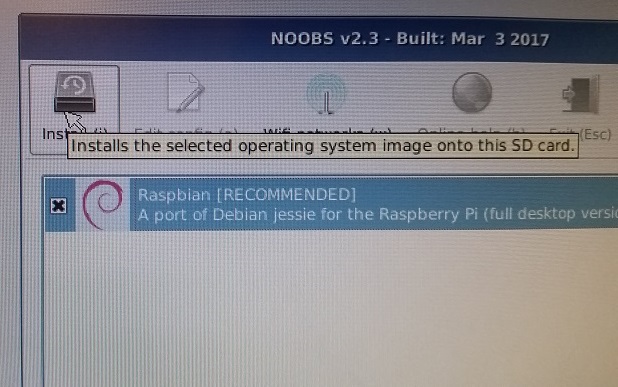


Рисунок 3 – Raspbian OS

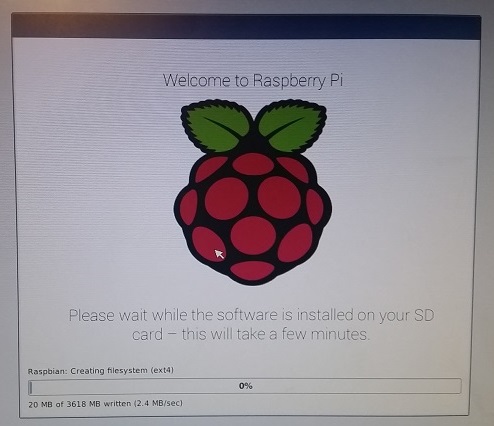


Рисунок 4 – Установка Raspbian OS

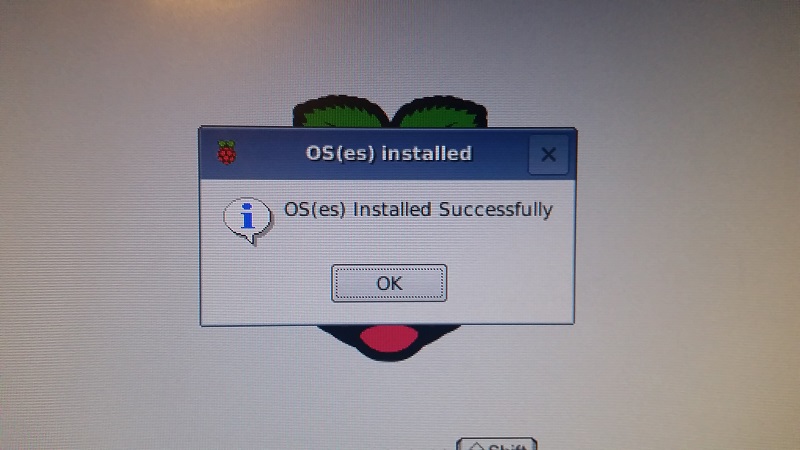


Рисунок 5 – Завершение установки

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).

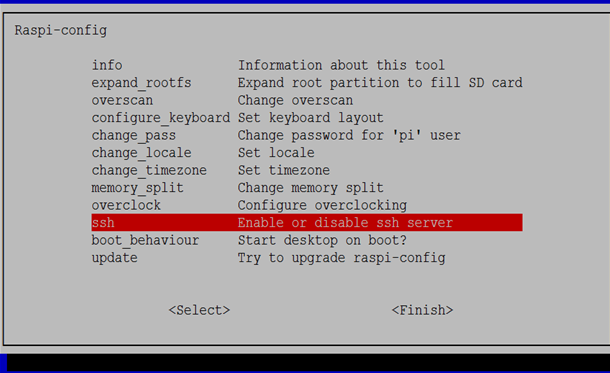


Рисунок 6 –Raspberry Pi

После чего перевел состояние ssh в enable (рисунок).



Рисунок 7 – 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).

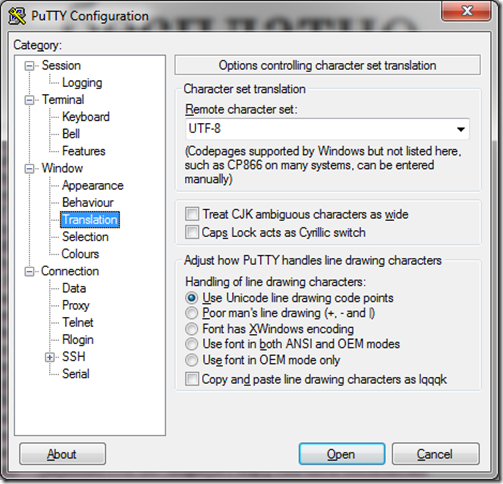


Рисунок 8 – PuTTY настройка кодировки

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

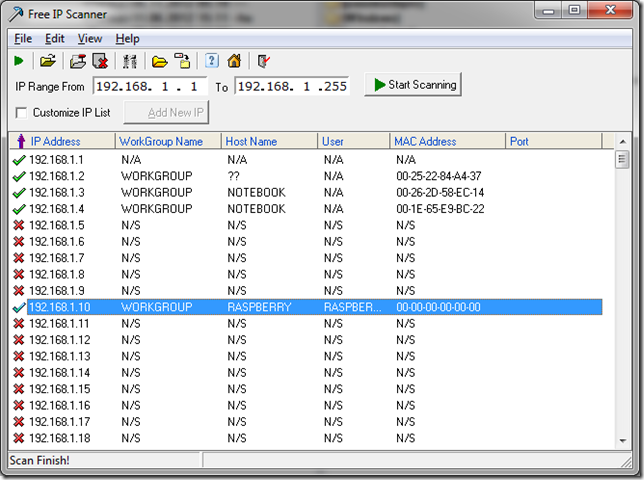


Рисунок 9 – Free IP Scanner

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

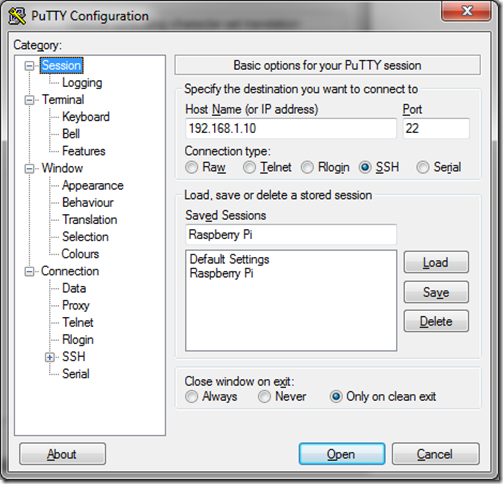


Рисунок 10 – PuTTY конфигурация сессии

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

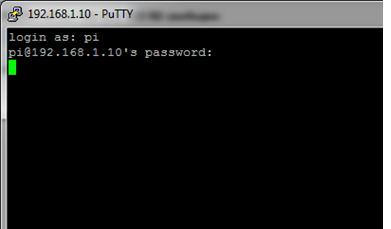


Рисунок 11 – Удаленный доступ

## 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):

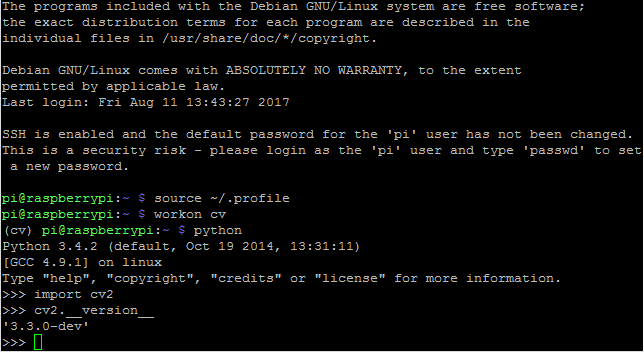


Рисунок 12 – PuTTY connection

Для установки облегченной версии Anaconda – Miniconda зашёл на сайт conda.pydata.org, далее в разделе скачивания выбрал версию Python 3.5 и в соответствии с разрядность моего компьютера выбрал 64-bit (рисунок 13).

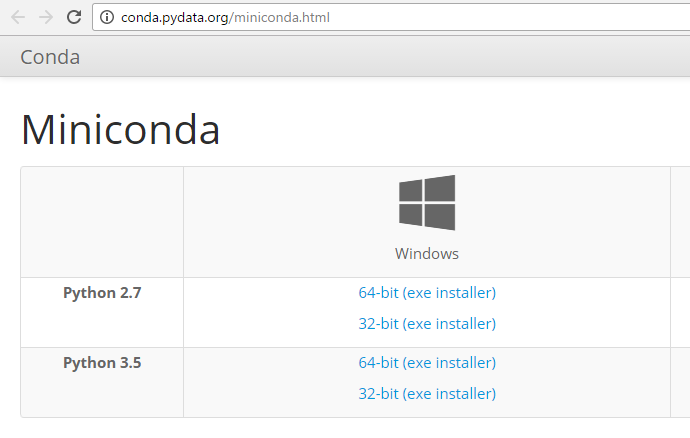


Рисунок 13 - Дистрибутив Miniconda

Далее при установке в разделе Advanced Options, проставил галочки (рисунок 14).

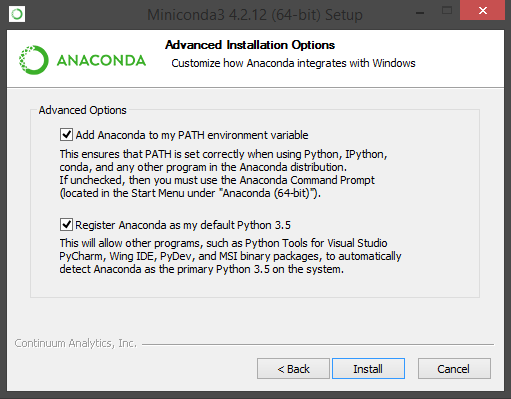


Рисунок 14 - Установк Miniconda

После чего для проверки успешности установки в командной строке для запуска интерпретатора ввёл python, в ответ должна запустится интерактивная командная оболочка Python (рисунок 15). Для выхода из оболочки достаточно ввести exit().

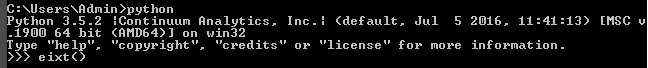


Рисунок 15 - Запуск интерактивной оболочки Python

Затем для Python установил библиотеку OpenCV со своими зависимостями (рисунок 16). После, для проверки успешности установки нужно импортировать библиотеку (рисунок 17).

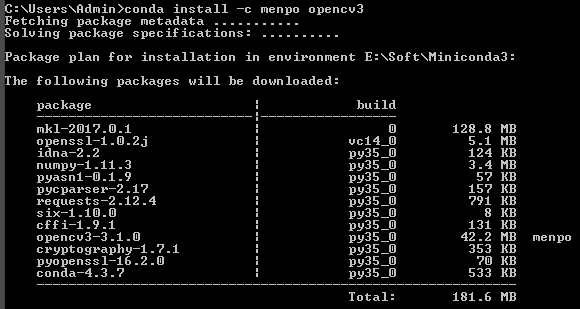


Рисунок 16 - Установка OpenCV для Python

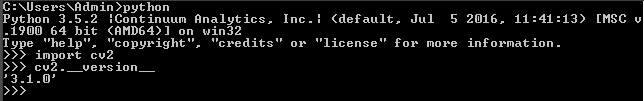


Рисунок 17 - Проверка установки OpenCV

Схожие действия нужно провести для установки библиотеки по работе с сетью (рисунок 18).

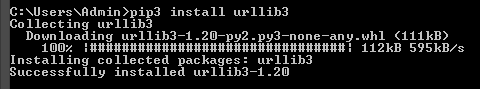


Рисунок 18 - Установка библиотеки urllib

Далее установил библиотеку theano и keras, после чего проверил на работоспособность (рисунки 19, 20, 21). Затем в корневом каталоге появился json файл, который позволяет менять настройки keras для выбора вычислительного backend’а (рисунок 22).

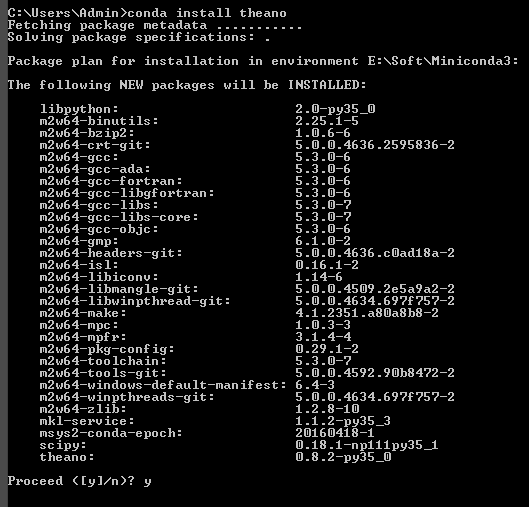


Рисунок 19 - Установка Theano

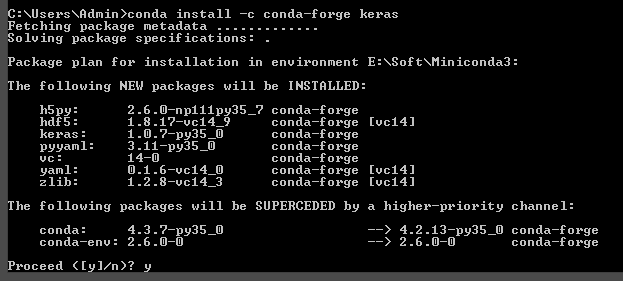


Рисунок 20 - Установка Keras



Рисунок 21 - Проверка Keras

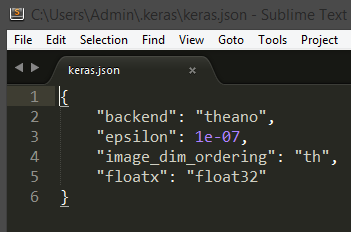


Рисунок 22 - JSON файл наcтройки Keras

Далее перешёл на сайт opencv.org, скачал последнюю версию скомпилированной библиотеки OpenCV (рисунок 23). Содержимое архива распаковал в папку (рисунок 24).

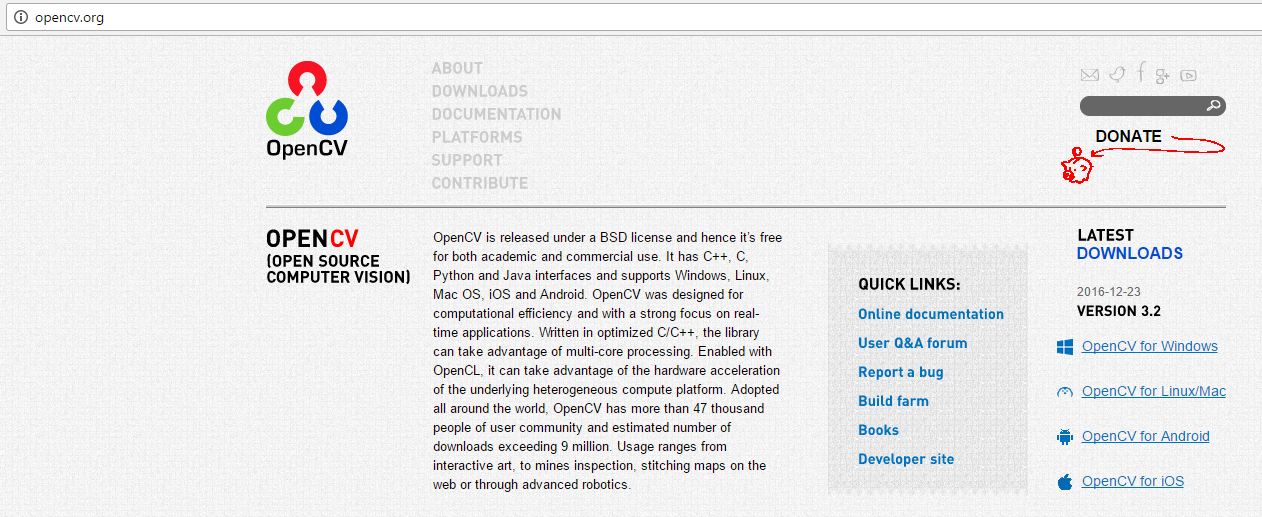


Рисунок 23 - Главный сайт библиотеки OpenCV

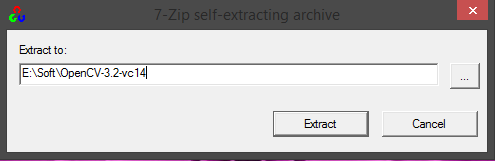


Рисунок 24 - Распаковка содержимого архива

Затем для удобства работы с исполняемыми файлами библиотеки в командной строке - добавил путь до bin папки в переменные среды (рисунки 25, 26).

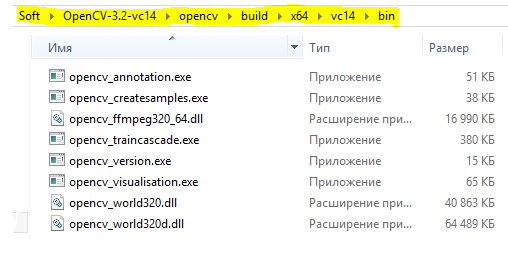


Рисунок 25 - Путь до папки bin

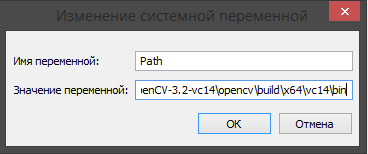


Рисунок 26 - Переменная среды

## 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).

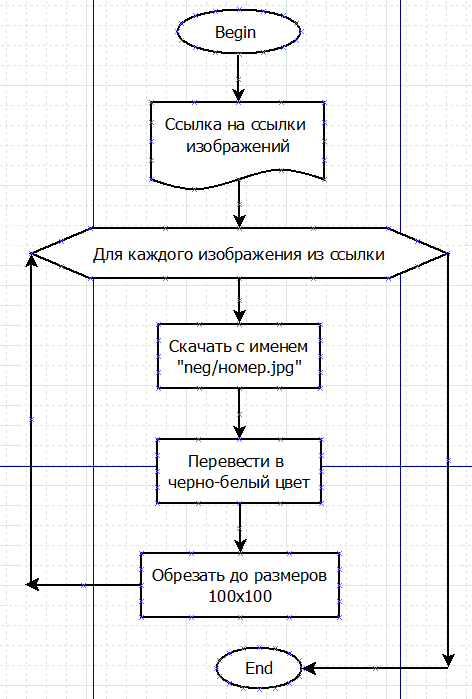


Рисунок 27- Блок-схема алгоритма скачивания изображений

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

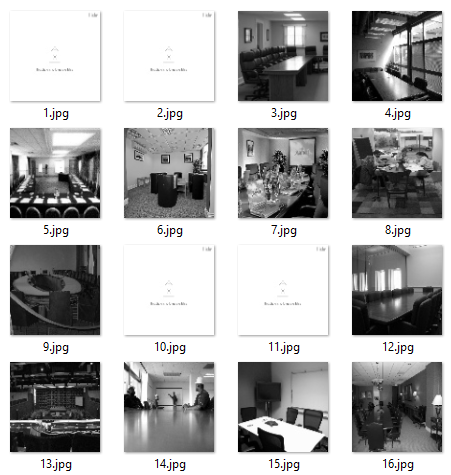


Рисунок 28 - Набор фоновых изображений

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).

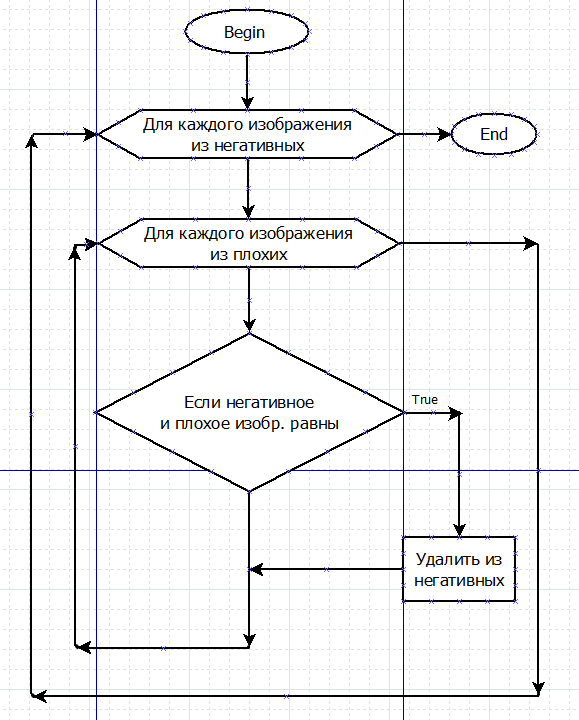


Рисунок 29 - Блок-схема скрипта удаляющего ошибочные изображения

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).

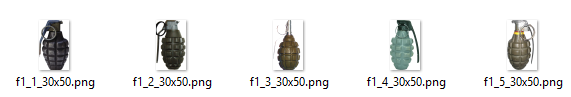


Рисунок 30 - Подготовленные изображения

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 31 - 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).

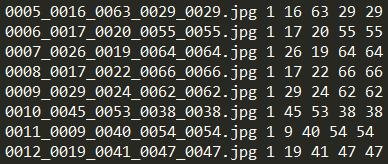


Рисунок 32 - Дескриптор получившихся изображений

## 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).

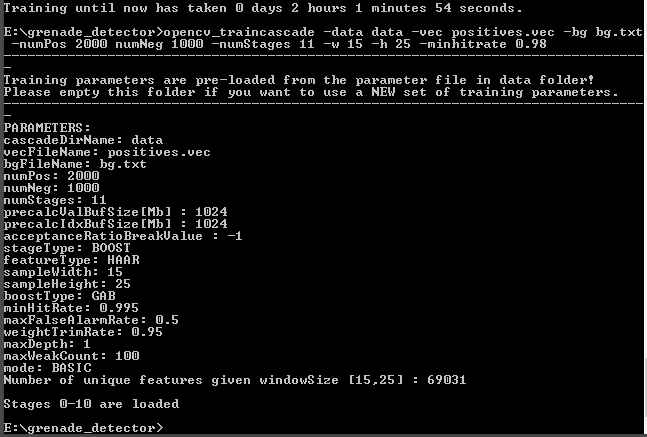


Рисунок 33 - Сборка итогового каскада

# 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).

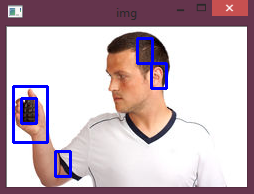


Рисунок 34 - Результат обработки

# 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).

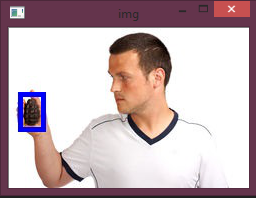


Рисунок 35 - Планируемый результат

# Список литературы

1. Хорн Б. К. П. Зрение роботов: Пер. с англ. – М.: Мир, 1989. – 487 с., ил. ISBN 5-03-000570-6

2. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis / Patrice Y.Simard; Dave Steinkraus; John C. Platt 2003

3. Back-Propagation Applied to Handwritten Zip Code Recognition / Y.LeCun, B.Boser, J.S.Denker et al. 1989

# 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()