УДК 004.89

Разработка скрипта поиска изображения лица с использованием библиотеки OpenCV

Черкашин Александр Михайлович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Научный руководитель:
Баженов Руслан Иванович
Приамурский государственный университет имени Шолом-Алейхема
к.п.н., доцент, зав. кафедрой информационных систем, математики и правовой информатики

Аннотация
Целью исследования является написание скрипта на языке Python по поиску изображения лица с использование библиотеки OpenCV. В статье представлен скрипт по чтению серии изображений, поиска лиц и вывода изображения содержащийся лица. В результате разработана программа по поиску лиц в изображениях.
Ключевые слова: распознавания лица, поиск лица в изображения, OpenCV.

Development of a face image search script using the OpenCV library

Cherkashin Alexander Mikhailovich
Sholom-Aleichem Priamursky State University
Student

Scientific adviser:
Bazhenov Ruslan Ivanovich
Sholom-Aleichem Priamursky State University
candidate of pedagogical sciences, associate professor, Head of the Department of Information Systems, Mathematics and Legal Informatics

Abstract
The purpose of the study is to write a script in Python to search for a face image using the OpenCV library. The article presents a script for reading a series of images, searching for faces, and displaying the image of the contained face. As a result, a program was developed to search for faces in images.
Keywords: face recognition, face search in images, OpenCV.

1 Введение

1.1 Актуальность исследования

Данная статья описывает возможность написание скрипта по поиску изображения лица.

1.2 Цель исследования

Целью исследования является написание скрипта на языке Python по поиску изображения лица с использование библиотеки OpenCV.

1.3 Обзор исследований

Авторы статьи «Краткое введение в библиотеку OpenCV» И. А. Цуляк рассматривает компьютерное зрение и библиотеку OpenCV [1]. В статье «Компьютерное зрение в реальном времени с OpenCV» К. Пулли и др. представили использование компьютерного зрения и сравнение оценки производительности использования CPU и GPU [2]. Г. Хие, Щ. Лу показали возможность использования библиотеки OpenCV для обнаружения краев изображений [3]. Г. Агам описывает справочник по использованию библиотеки OpenCV [4]. А. Собрал описывает библиотеку BGSLibrary для захвата движения и распознавания изображений с использованием OpenCV [5]. Х. Фан рассматривает технологию обнаружения лиц с применением алгоритма AdaBoost на основе библиотеки OpenCV[6]. М. Маренгони, Д. Стрингхини представили теоретические части и практические примеры по обработке изображений и применению компьютерного зрения с использованием библиотеки OpenCV. Описывают технологию графического ускорения CUDA[7]. М. Маренгони, С. Стрингхини описывают учебное руководство по использованию библиотеки OpenCV [8]. Ц. Домингуез, Й. Херас, В. Пасцуал показали возможности библиотеки OpenCV в анализе изображений [9]. П. Н. Дружков и др. рассматривает проблемы создания алгоритмов обнаружения объектов на основе градиентного спуска деревьев [10]. К. Гоял, К. Агарщал, Р. Кумар показали реализацию распознавания лиц и захвата движения с камеры с использованием библиотеки OpenCV [11]. Г.Чандан, описывает алгоритмы глубокого обучения Single Shot Detector (SSD), Region-based Convolutional Neural Networks (RCNN) по скорости распознавания объектов [12]. К. М. Сайяд, применяет библиотеку OpenCV для автоматического распознавания номерных знаков (ALPR)[13]. Н. Боико, О. Басыстиук, Н. Шакховска описывают сравнение библиотек по компьютерному зрениею OpenCV и dlib [14]. А. Ануар и др. рассматривает возможность по экономии энергопотребления с использованием библиотеки OpenCV для мобильных устройств [15].

2. Результаты и обсуждение

2.1 Исходные данные

Набор данных содержится в виде изображений (рис 1), данные загружались из интернета.

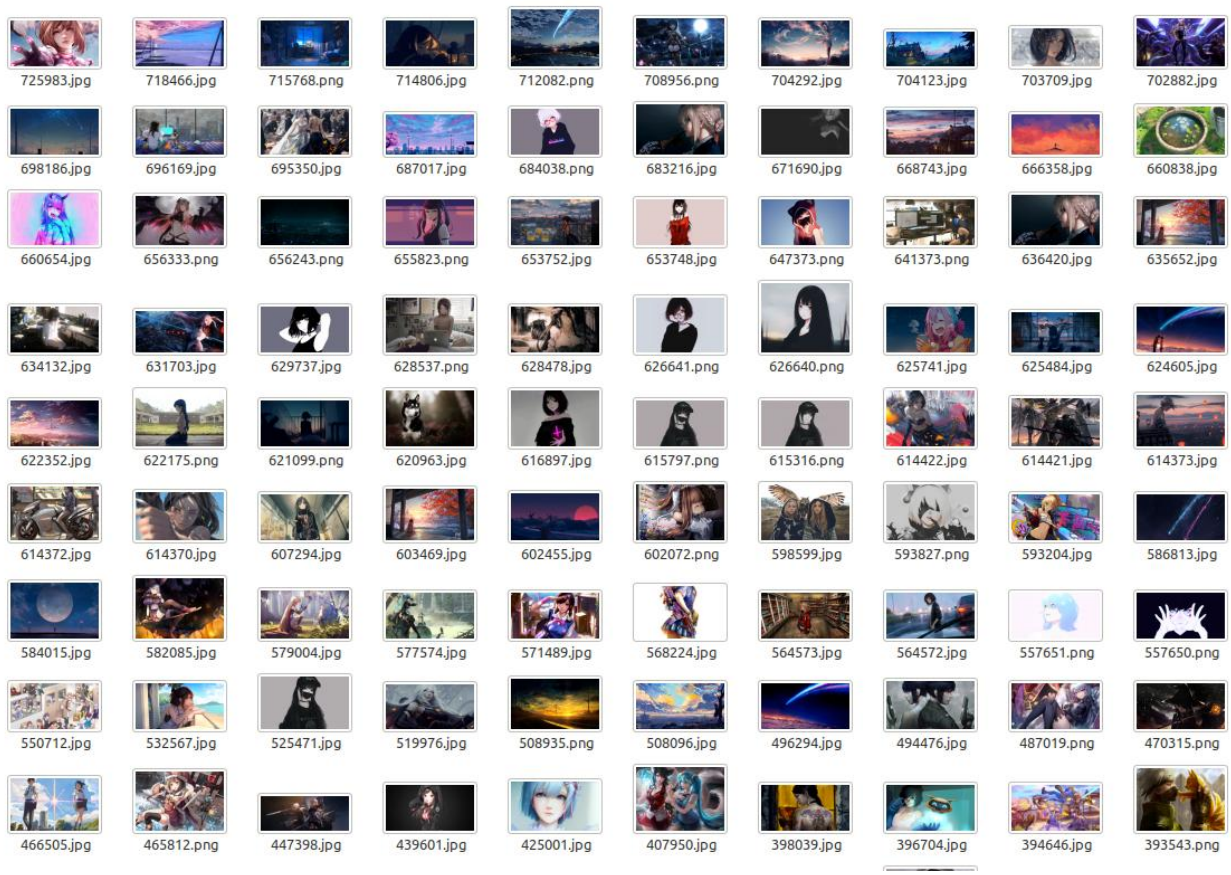

Рисунок 1. Исходные изображения

2.2. Скрипт

Данный скрипт написан на языке Python и использует библиотеки OpenCV, numpy, multiprocessing, argparse, tqdm, os, sys. Использует два наборы готовых обученных нейронныхсетей по распознавания лиц:

1. Встроенный в OpenCV haarcascade_frontalface_alt.xml каскадные распознавания лица.

2. Данные готовой нейронной сети для распознавания лиц взяты с https://github.com/kb22/Create-Face-Data-from-Images . Она использует модуль DNN OpenCV. Хранится в директории /model_data

Скрипт управляется через терминал (листинг 1).

```
usage: get_faces [-i [URL...]] [-o [URL...]] [-t [Number...]] [-v [VERBOSE]]
                 [-p [PROGRESS]] [--help]

Face detection

optional arguments:
  -i [ URL... ], --input [ URL... ]
                        Source dir images
  -o [ URL... ], --output [ URL... ]
                        Output dir images
  -t [ Number... ], --thread [ Number... ]
```

```
                Run simultaneously to N tasks
 -v [VERBOSE], --verbose [VERBOSE]
                Explain what will be done
 -p [PROGRESS], --progress [PROGRESS]
                Only progress
 --help, -h          Print this help text and exit.
```

Листинг 1. Скрипт введенная команда «./get_faces.py —help»

В скрипте содержится аргументы в консоли, -i, --input — исходная директория изображений. -o, --output — директория вывода для изображения. -t, --thread — число одновременно выполняемых процессов (многопоточность). -v, --verbose — выводить подробные действия в консоль. -p, --progress — выводить только индикатор выполнения в консоль. -h, --help — показать справку и выйти.

Таблица 1. Файловая структура проекта

Название	Описание
/	Корень проекта
/get_faces.py	Исходное скрипт по поиску лиц в изображения
/model_data	Директория содержащая готовые нейронные сети для распознавания лиц
/image	Исходные изображения
/out_image	Изображения после обработки

При запуске скрипт с заданным значением —thread создается множество процессов (многопоточность), читается директория с изображениями, указанная аргументом —input. Программа читает каждый файл и выполняет поиск лиц в изображении. Вначале используется метод стандартного набора обученной нейронной сети в библиотеке OpenCV, а затем готовый набор обученной нейронной сети в директории /model_data. Из изображения вырезается найденное лицо и результат выводится в директорию указанную в аргументе —output (рис 2). Если файл существует то распознавание лиц не выполняется (Таблица 2).

Таблица 2. Методы распознавания лиц

Название метода	Имя файла для вывода
Стандартный набор обученной нейронной сети в библиотеке OpenCV	Имя файла_alt_Номер.Расширение

/model_data готовый набор данных обученной нейронной сети DNN	Имя файла_dnn_Номер.Расширение

Скрипт использует расширение файлов .png, .jpg, .jpeg. В таблице 2 указан «номер» — количество найденных лиц в одном изображении, «Имя файла» - исходное название файла, «Расширение» - исходное расширение.

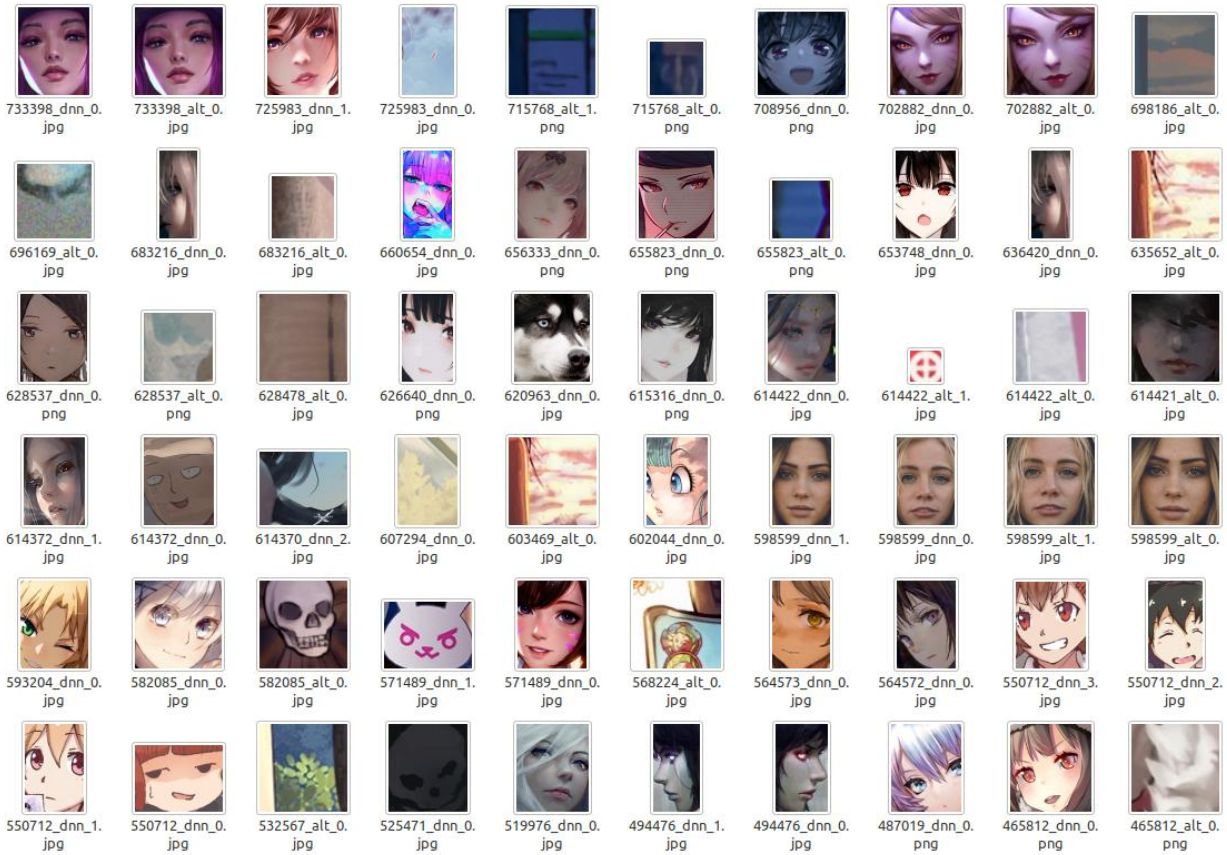

Рисунок 2. Набор изображения как результат найденных лиц

2.3. Исходный код

```
1   #!/usr/bin/python3
2   # -*- coding: utf-8 -*-
3   import os, sys
4   import cv2 as cv
5   import numpy as np
6   import multiprocessing
7   import argparse
8   from tqdm import tqdm
9   parser = argparse.ArgumentParser(
10              description = '''
11  Face detection
12  ''',
13              prog = "get_faces",
14              add_help = False
15  )
16  class readable_dir(argparse.Action):
17              def __call__(self, parser, namespace, values, option_string=None):
18                      prospective_dir=values
19                      if not os.path.isdir(prospective_dir):
20                              raise argparse.ArgumentTypeError("readable_dir:{0} is not a valid path".format(prospective_dir))
21                      if os.access(prospective_dir, os.R_OK):
22                              setattr(namespace,self.dest,prospective_dir)
23                      else:
24                              raise argparse.ArgumentTypeError("readable_dir:{0} is not a readable dir".format(prospective_dir))
25
26  class arg_verbose(argparse.Action):
27              def __call__(self, parser, namespace, values, option_string=None):
28                      setattr(namespace,self.dest, 1)
```

```
29
30  parser.add_argument('-i', "--input", metavar="[ URL... ]", action=readable_dir, default="./image", type=str, help="Source dir images")
31  parser.add_argument('-o', "--output", metavar="[ URL... ]", action=readable_dir, default="./out_image", type=str, help="Output dir images")
32  parser.add_argument('-t', "--thread", metavar="[ Number... ]", default="2", type=int, help="Run simultaneously to N tasks")
33  parser.add_argument('-v', "--verbose", nargs="?", action=arg_verbose, help="Explain what will be done")
34  parser.add_argument('-p', "--progress", nargs="?", action=arg_verbose, help="Only progress")
35  parser.add_argument('--help', '-h', action="help", help="Print this help text and exit.")
36  cc_face_alt = cv.CascadeClassifier(cv.data.haarcascades + 'haarcascade_frontalface_alt.xml')
37  prototxt_path = os.path.join('./model_data/deploy.prototxt')
38  caffemodel_path = os.path.join('./model_data/weights.caffemodel')
39  model_dnnCaff = cv.dnn.readNetFromCaffe(prototxt_path, caffemodel_path)
40  def write_image(image_out, full_url, i, out_image, verbose):
41      file_name, file_ext = os.path.splitext(os.path.basename(full_url))
42      file_full = file_name + file_ext
43      url_out = out_image + "/" + file_name + "_" + str(i) + file_ext
44      if image_out is None:
45          print("Write image Empty: ", url_out)
46          return
47      try:
48          if (verbose == 1):
49              print("write image: ", url_out)
50          cv.imwrite(url_out, image_out)
51      except cv.error:
52          print("Error: image write: ", url_out)
53
54  def worker_detect(file, out_image, verbose):
55      file_name, file_ext = os.path.splitext(os.path.basename(file))
56      if (os.path.isfile(out_image + "/" + file_name + "_alt" +  "_" + str(0) + file_ext) and
57          os.path.isfile(out_image + "/" + file_name + "_dnn" +  "_" + str(0) + file_ext)):
58          if (verbose != -1):
59              print("Skipping: File exists: " + out_image + "/" + file_name + "_alt" +  "_" + str(0) + file_ext)
60              print("Skipping: File exists: " + out_image + "/" + file_name + "_dnn" +  "_" + str(0) + file_ext)
61          return
62
63      if (not file_ext in [".png", ".jpg", ".jpeg"]):
64          return
65
66      if (verbose == 1):
67          print("read file: ", file)
68      if file in ["..", "."]:
69          return
70      image = cv.imread(file)
71      if image is None:
72          print("Error read file: ", file)
73          return
74      if (not os.path.isfile(out_image + "/" + file_name + "_alt" +  "_" + str(0) + file_ext)):
75          if (verbose == 1):
76              print("Face Detect default")
77          faces = cc_face_alt.detectMultiScale(image, 1.1, 4)
78          i = 0
79          for (x,y,w,h) in faces:
80              frame = image[y:(y+h), x:(x+w)]
81              write_image(frame, file_name + "_alt" + file_ext, i, out_image, verbose)
82              i += 1
83      else:
84          if (verbose != -1):
85              print("Skipping: File exists: " + out_image + "/" + file_name + "_alt" +  "_" + str(0) + file_ext)
86
87      if (not os.path.isfile(out_image + "/" + file_name + "_dnn" +  "_" + str(0) + file_ext)):
88          if (verbose == 1):
89              print("Face Detect DNN")
90          blob = cv.dnn.blobFromImage(cv.resize(image, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0))
91          model_dnnCaff.setInput(blob)
92          detections = model_dnnCaff.forward()
93          (h, w) = image.shape[:2]
94          for i in range(0, detections.shape[2]):
95              box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
96              (x0, y0, x1, y1) = box.astype("int")
97              confidence = detections[0, 0, i, 2]
98              if (confidence > 0.5):
99                  frame = image[y0:y1, x0:x1]
100                 write_image(frame, file_name + "_dnn" + file_ext, i, out_image, verbose)
101     else:
102         if (verbose != -1):
103             print("Skipping: File exists: " + out_image + "/" + file_name + "_dnn" +  "_" + str(0) + file_ext)
104
105 if __name__ == '__main__':
106     num_proc = 2
107     url_image = "./image"
108     url_image_out = "./out_image"
109     verbose = 0
110     arg_data = parser.parse_args(sys.argv[1:])
111     num_proc = arg_data.thread
112     url_image = arg_data.input
113     url_image_out = arg_data.output
114     if (arg_data.progress == 1):
115         verbose = -1
116     verbose = arg_data.verbose
117     if (verbose == 1):
```

```
                print("verbose: on")
                print("thread: ", num_proc)
                print("input dir image: ", url_image)
                print("output dir image: ", url_image_out)

        dir_ls = os.listdir(url_image)
        i = 0
        tqbar = tqdm(total=len(dir_ls))
        while (i < len(dir_ls)):
                for j in range(min(num_proc, len(dir_ls)-i)):
                        if (verbose == 1):
                                print("process start: dir_ls[i]: ", dir_ls[i])
                        p = multiprocessing.Process(target=worker_detect, args=(url_image + "/" + dir_ls[i], url_image_out, verbose,))
                        p.start()
                        i += 1
                        tqbar.update(1)
                if (verbose == 1):
                        print("process join...")
                p.join()
```

3 Выводы

В данной статье был разработан скрипт по поиску изображения лица с использованием библиотеки OpenCV, которая содержит две готовых обученных нейронных сети для распознавания лиц в изображениях. Скрипт позволяет искать лица в изображениях и выводить лица в виде отдельных файлов.

Библиографический список

1. Culjak I. et al. A brief introduction to OpenCV //2012 proceedings of the 35th international convention MIPRO. IEEE, 2012. С. 1725-1730.
2. Pulli K. et al. Real-time computer vision with OpenCV //Communications of the ACM. 2012. Т. 55. №. 6. С. 61-69.
3. Xie G., Lu W. Image edge detection based on opencv //International Journal of Electronics and Electrical Engineering. 2013. Т. 1. №. 2. С. 104-106.
4. Agam G. Introduction to programming with OpenCV //Online Document. 2006. Т. 27.
5. Sobral A. et al. BGSLibrary: An opencv c++ background subtraction library //IX Workshop de Visao Computacional. 2013. Т. 27.
6. Fan X. et al. The system of face detection based on OpenCV //2012 24th Chinese Control and Decision Conference
7. Marengoni M., Stringhini D. High level computer vision using opencv //2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials. – IEEE, 2011. С. 11-24.
8. Marengoni M., Stringhini S. Tutorial: Introdução à visão computacional usando opencv //Revista de Informática Teórica e Aplicada. 2009. Т. 16. №. 1. С. 125-160.
9. Domínguez C., Heras J., Pascual V. IJ-OpenCV: combining ImageJ and OpenCV for processing images in biomedicine //Computers in biology and medicine. 2017. Т. 84. С. 189-194.
10. Druzhkov P. N. et al. New object detection features in the OpenCV library //Pattern Recognition and Image Analysis. 2011. Т. 21. №. 3. С. 384-386.
11. Goyal K., Agarwal K., Kumar R. Face detection and tracking: Using OpenCV //2017 International conference of Electronics, Communication and Aerospace

Technology (ICECA). IEEE, 2017. Т. 1. С. 474-478.
12. Chandan G. et al. Real time object detection and tracking using Deep Learning and OpenCV //2018 International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, 2018. С. 1305-1308.
13. Sajjad K. M. Automatic license plate recognition using python and opencv //Department of Computer Science and Engineering MES College of Engineering. 2010.
14. Boyko N., Basystiuk O., Shakhovska N. Performance evaluation and comparison of software for face recognition, based on dlib and opencv library //2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP). IEEE, 2018. С. 478-482.
15. Anuar A. et al. OpenCV based real-time video processing using android smartphone //International Journal of Computer Technology and Electronics Engineering (IJCTEE). 2011. Т. 1. №. 3. С. 1-6.