ГУАП

КАФЕДРА № 14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Ассистент		А.С. Костин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3
Введение в OpenCV, поиск образов

по курсу: ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 1042		Н. В. Корзун
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

1. Цель работы

В рамках данной лабораторной работы вам необходимо выполнить следующее:

1) Сформировать изображение в котором:

1.1) Разные фигуры(круги, квадраты и тд)

1.2) Фигуры должны быть разных цветов (цвета и фигуры могут повторяться)

В рамках задачи по поиску контуров вам нужно:

1) В начале изображения вывести № вашей группы, ФИО, количество найденных контуров. В терминал вывести ту же информацию.

2) Обвести все ваши объекты в контур

3) В нижнем левом углу вашего изображения отобразите точку, по центру изображения квадрат

Цветовую гамму и размеры шрифтов можете использовать по вашему усмотрению.

2. Краткое изложение основных теоретических понятий

Python – интерпретируемый высокоуровневый язык программирования общего назначения. Он используется в различных качествах: как основной язык программирования
или для создания расширений и интеграции приложений. В лабораторной работе используется версия 3.10.6. Выбор версии не оказывает влияния из-за обратной совместимости, поэтому в отчете именуем как python 3.X.

Python хорошо подходит, как для сложных проектов, так и для простых консольных
приложений.

OpenCV (Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом)— 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 языков[3]. Может и используется свободно в академических и коммерческих целях.

Pillow – форк известной библиотеки PIL поддерживаемый современные версии python. Библиотека предназначена для работы с растровой графикой.

3. Постановка задачи с кратким описанием порядка выполнения

Задача сформулирована четко в пункте 1.

Формировать изображение будем с помощью библиотеки pillow. На формируемое изображение нанесем произвольное число прямоугольников и эллипсов различных форм и цветов. Действия по поиску контуров и их классификации просты ввиду понятной презентации, отражающей суть поставленной задачи.

4. Результаты работы программы

Пример ввода для генерации и вывода изображения представлен на рисунке 1.

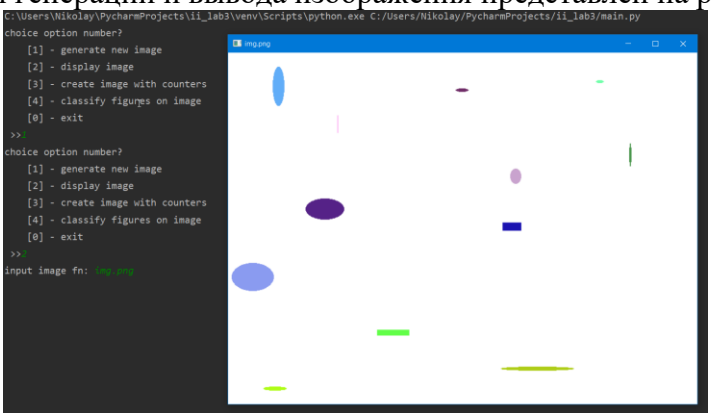

Рисунок 1. Генерация изображения

Нахождение контуров показано на рисунке 2.

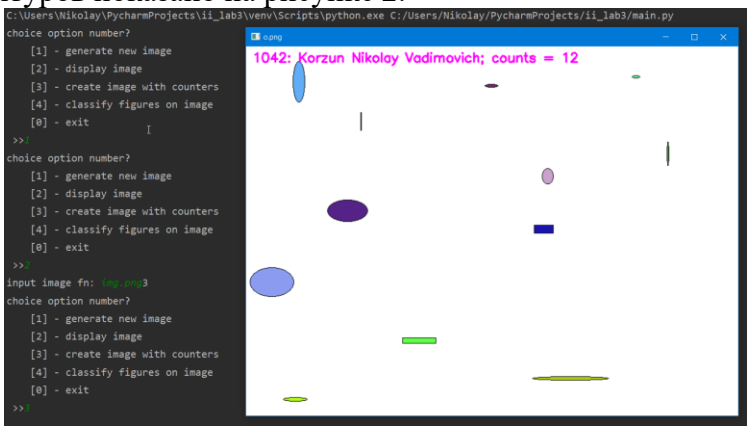

Рисунок 2. Нахождение контуров

5. Общий вывод по проделанной работе

В ходе выполнения лабораторной работы были получены навыки по нахождению контуров на изображении с помощью библиотеки машинного зрения OpenCV

Приложение 1. Исходный код.

```
import math

import cv2
import imutils
from PIL import Image, ImageDraw
import os
from random import randint
from math import pi, sqrt

class App:
    __info_for_lab3_str = "1042: Korzun Nikolay Vadimovich; counts = {}"
    __min_count_rect = 2
    __max_count_rect = 10
    __min_count_ellipse = 2
    __max_count_ellipse = 10
    __min_color_param = 10
    __input_image_filename = "img.png"
    __output_image_filename = "o.png"
    __counters_color = (0, 0, 0)
    __text_color = (255, 0, 255)
    def __get_random_color(self):
        return (randint(self.__min_color_param, 255), randint(self.__min_color_param, 255),
randint(self.__min_color_param, 255))
    def __get_random_xy_in_area(self, maxw, maxh):
        def get_min_in_n_repeat(min_, max_, n):
            a=[]
            for i in range(n):
                a.append(randint(min_, max_))
            return min(a)
        w = get_min_in_n_repeat(1, maxw-10, 1)
        h = get_min_in_n_repeat(1, maxh-10, 1)
        ww = get_min_in_n_repeat(w+1, maxw - 10, 10)
        hh = get_min_in_n_repeat(h + 1, maxh - 10, 10)
        return (w, h, ww, hh)
    def __init__(self, file_name="img.jpg"):
        pass
    def print_creator(self):
        print("this simply app developed by Korzun Nikolay.")
    def __show_image(self, fn, additionaly_info=""):
        if os.path.isfile(fn):
            image = cv2.imread(fn)
            if len(additionaly_info) != 0:
                cv2.putText(image, additionaly_info, (10, 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, self.__text_color, 2)
            cv2.imshow(fn, image)
            cv2.waitKey(0)
        else:
            print("unavailable filename {}".format(fn))
    def __lab3(self, fn_input, fn_output):
        image = cv2.imread(fn_input)
```

```
        gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        ret, thresh = cv2.threshold(gray_img, 255 - self.__min_color_param, 255,
cv2.THRESH_BINARY_INV)
        cnts = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        output = image.copy()
        for c in cnts:
            cv2.drawContours(output, [c], -1, self.__counters_color, 1)
        cv2.imwrite(fn_output, output)
        return len(cnts)
    def __lab4(self, fn_input, fn_output):
        def classify(c):
            def comp(a, b, eps=0.05):
                return (a >= b*(1-eps) and a <= b*(1+eps))
            #approximation? But why? It is performed during the search.
            shape = "unidentified" #this value not using in original code. wtf?
            peri = cv2.arcLength(c, True)
            area = cv2.contourArea(c)
            approx = cv2.approxPolyDP(c, 0.01 * peri, True)
            #count_vertexes = len(c)
            count_vertexes = len(approx)
            if count_vertexes == 2:
                shape = "line"
            elif count_vertexes == 3:
                shape = "triangle"
            elif count_vertexes == 4:
                (x, y, w, h) = cv2.boundingRect(c)
                ar = w / float(h)
                shape = "square" if comp(ar, 1) else "rectangle"
            elif count_vertexes == 5:
                shape = "pentagon"
            else:
                if comp(peri/(2*pi), sqrt(area/pi), 0.06):
                    shape = "circle"
                else:
                    shape = "ellipse"
            return shape
        image = cv2.imread(fn_input)
        gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        ret, thresh = cv2.threshold(gray_img, 255 - self.__min_color_param, 255,
cv2.THRESH_BINARY_INV)
        cnts = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        output = image.copy()
        shapes = {}
        for c in cnts:
            M = cv2.moments(c)
            cX = int((M["m10"] / M["m00"]))
            cY = int((M["m01"] / M["m00"]))
            shape = classify(c)
```

```
            if shape not in shapes:
                shapes[shape] = 1
            else:
                shapes[shape] += 1
            cv2.drawContours(output, [c], -1, self.__counters_color, 1)
            #cv2.circle(output, (cX, cY), 7, (255, 255, 255), -1)
            cv2.putText(output, shape, (cX, cY), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
self.__text_color, 2)
        cv2.imwrite(fn_output, output)
        return shapes
    def __print_info(self, info):
        print(info)
    def menu(self):
        c = input("choice option number?\n\t[1] - generate new image\n\t[2] - display
image\n\t[3] - create image with"
                  " counters\n\t[4] - classify figures on image\n\t[0] - exit\n >>")
        if c == "1":
            self.__generate_new_image(self.__input_image_filename, 800, 600)
        elif c == "2":
            fn = input("input image fn: ")
            self.__show_image(fn)
        elif c == "3":
            count = self.__lab3(self.__input_image_filename, self.__output_image_filename)
            self.__show_image(self.__output_image_filename,
self.__info_for_lab3_str.format(count))
        elif c == "4":
            info = self.__lab4(self.__input_image_filename, self.__output_image_filename)
            self.__print_info(info)
        else:
            return False
        return True
    def __generate_new_image(self, filename:str, width, height):
        img = Image.new('RGBA', (width, height), 'white')
        idraw = ImageDraw.Draw(img)
        count_rect = randint(self.__min_count_rect, self.__max_count_rect)
        count_ellipse = randint(self.__min_count_ellipse, self.__max_count_ellipse)
        for i in range(count_rect):
            idraw.rectangle(self.__get_random_xy_in_area(width, height), fill =
self.__get_random_color())
        for i in range(count_ellipse):
            idraw.ellipse(self.__get_random_xy_in_area(width, height), fill =
self.__get_random_color())
        img.save(filename)
if __name__ == "__main__":
    app = App()
    while(app.menu()):
        pass
```