**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ВТ**

отчет

**по лабораторной работе №2**

**по дисциплине «Компьютерное зрение»**

Тема: Базовые операции рисования фигур на изображении

|  |  |  |
| --- | --- | --- |
| Студент гр. 5307 |  | Пешков М.А.  Пешков Д.А. |
| Преподаватель |  | Костичев С.В. |

Санкт-Петербург

2020

**Цель работы.**

Изучить применение базовых операций рисования на изображении

**Основные теоретические положения.**

В OpenCV любое изображение представляет собой матрицу интенсивностей, а именно объект класса **Mat**. Один из способов создания объекта данного типа – использование конструктора класса **Mat**:

Mat(int \_rows, int \_cols, int \_type);

// \_rows – количество строк

// \_cols – количество столбцов

// \_type – тип матрицы (CV\_8UC1, CV\_64FC3 и другие)

После обработки изображения необходимо посмотреть результат. Для корректного отображения изображения необходимо последовательно выполнить три действия:

1. Создать окно для отображения с помощью функции **namedWindow**. Функция принимает на вход название окна, которое служит идентификатором окна, и флаги. Отметим, что если окно с аналогичным идентификатором уже существует, то функция не выполняет никаких действий.

void namedWindow(const string& winname, int flags)

Параметр **flags** может принимать следующие значения или их комбинации, полученные в результате применения оператора ‘**OR**’:

* + **CV\_WINDOW\_NORMAL** или **CV\_WINDOW\_AUTOSIZE**, первое значение позволяет пользователю вручную изменять размеры окна в процессе работы приложения, в то время, как второе автоматически подгоняет размеры окна под размеры изображения, запрещая пользователю изменять размер вручную;
  + **CV\_WINDOW\_FREERATIO** или **CV\_WINDOW\_KEEPRATIO**, **CV\_WINDOW\_FREERATIO** подгоняет изображение под окно без сохранения пропорций, **CV\_WINDOW\_KEEPRATIO** позволяет сохранять пропорции;
  + **CV\_GUI\_NORMAL** или **CV\_GUI\_EXPANDED**, первое значение обеспечивает отрисовку окна без дополнительных компонент таких, например, как строка состояний и панель инструментов, второе же, напротив, предоставляет более новые возможности по созданию графических интерфейсов пользователя наряду с отображением изображений.

По умолчанию для окна устанавливаются свойства **CV\_WINDOW\_AUTOSIZE**, **CV\_WINDOW\_KEEPRATIO**, и **CV\_GUI\_EXPANDED**.

2. Отобразить изображение посредством вызова функции **imshow**. Для отображения необходим идентификатор окна **winname**, в которое будет помещено изображение, а также само изображение **image**.

void imshow(const string& winname, const Mat& image)

3. Дождаться нажатия какой-либо клавиши, чтобы закрыть окно. Ожидание реализуется посредством вызова функции **waitKey**, которая принимает на вход время ожидания **delay** в миллисекундах. По умолчанию **delay=0**, что означает ожидание в течение бесконечного промежутка времени, т.е. исполнение приложения будет продолжено после нажатия какой-либо клавиши. Отметим, что отсутствие вызова указанной функции приведет к тому, что по завершении программы окно будет автоматически закрыто.

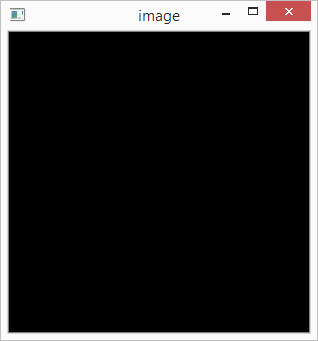
int waitKey(int delay=0)

Рисование на изображение осуществляется с помощью функций

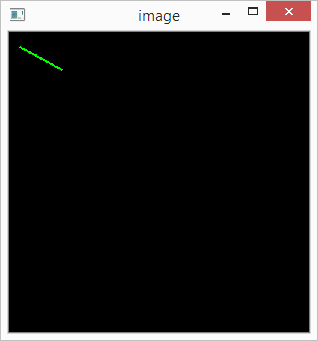
line, rectangle, circle, ellipse, fillPoly, polylines, putText.

**Экспериментальные результаты.**

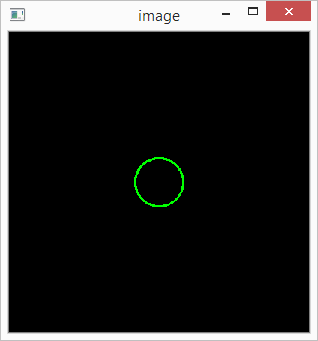
1. Создали в качестве изображения черный квадрат размером 400\*400 пикселей.



2. Нарисовали в нем отрезок любого цвета между точками с координатами (15, 20) и (70, 50).



3. Нарисовали в нем окружность любого цвета с центром в точке с координатами (200, 200) и радиусом 32



4. Нарисовали в нем эллипс любого цвета со следующими параметрами

- центр в точке с координатами (200, 200),

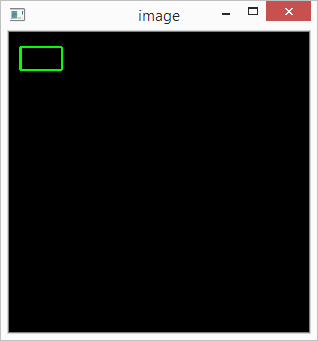
- оси 100 и 160

- угол поворота эллипса в градусах 45

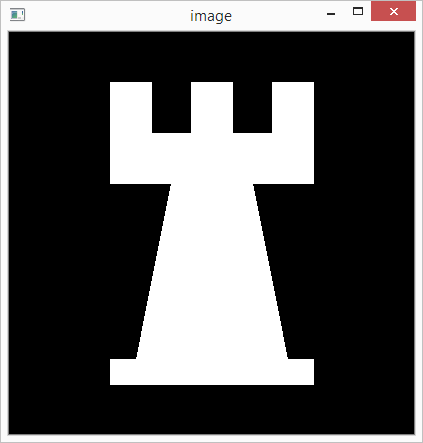
- начальный и конечный углы эллиптической дуги в градусах 0 и 360

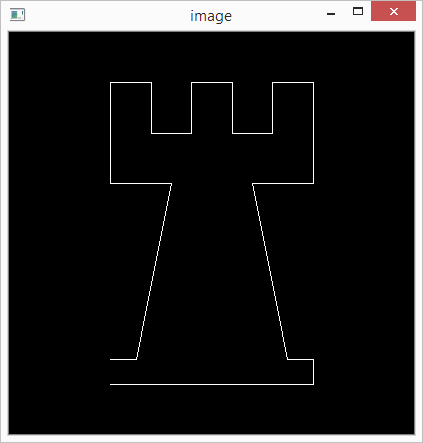


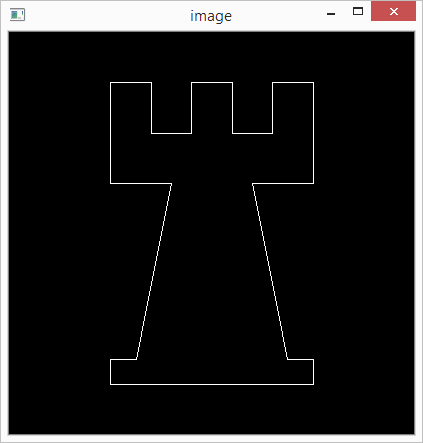
5. Нарисовали в нем прямоугольник любого цвета с координатами противоположных вершин (15, 20) и (70, 50)



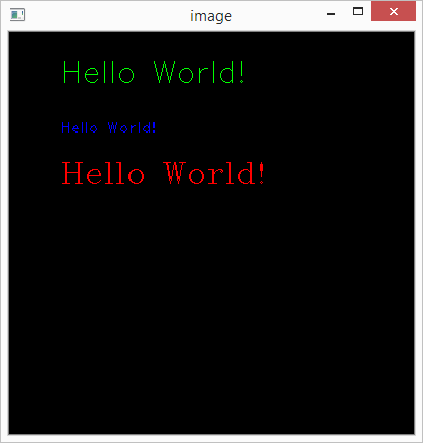
6. Нарисовали в нем полигоны следующих видов:







7. Нарисовать в нем текстовую строку “Hello, world!” различными шрифтами и различным цветом



**Выводы.**

Изучили применение базовых операций рисования на изображении

Приложение А

1. Создали в качестве изображения черный квадрат размером 400\*400 пикселей.

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

2. Нарисовали в нем отрезок любого цвета между точками с координатами (15, 20) и (70, 50).

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
start = (15, 20)  
end = (70, 50)  
  
line = cv2.line(img, start, end, (0, 255, 0))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', line)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

3. Нарисовали в нем окружность любого цвета с центром в точке с координатами (200, 200) и радиусом 32

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
center = (200, 200)  
radius = 32  
  
line = cv2.circle(img, center, radius, (0, 255, 0))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', line)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

4. Нарисовали в нем эллипс любого цвета со следующими параметрами

- центр в точке с координатами (200, 200),

- оси 100 и 160

- угол поворота эллипса в градусах 45

- начальный и конечный углы эллиптической дуги в градусах 0 и 360

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
center = (200, 200)  
axis = (100, 160)  
angel = 45  
startAngle = 0  
endAngle = 360  
  
line = cv2.ellipse(img, center, axis, angel, startAngle, endAngle, (0, 255, 0))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', line)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

5. Нарисовали в нем прямоугольник любого цвета с координатами противоположных вершин (15, 20) и (70, 50)

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
pt1 = (15, 20)  
pt2 = (70, 50)  
  
line = cv2.rectangle(img, pt1, pt2, (0, 255, 0))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', line)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

6. Нарисовали в нем полигоны следующих видов:

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
pts = numpy.array([  
 [100, 50],  
 [140, 50],  
 [140, 100],  
 [180, 100],  
 [180, 50],  
 [220, 50],  
 [220, 100],  
 [260, 100],  
 [260, 50],  
 [300, 50],  
 [300, 150],  
 [240, 150],  
 [275, 325],  
 [300, 325],  
 [300, 350],  
 [100, 350],  
 [100, 325],  
 [125, 325],  
 [160, 150],  
 [100, 150],  
], numpy.int32)  
# poly = cv2.polylines(img, [pts], True, (255, 255, 255))  
poly = cv2.fillPoly(img, [pts], (255, 255, 255))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', poly)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
pts = numpy.array([  
 [100, 325],  
 [125, 325],  
 [160, 150],  
 [100, 150],  
 [100, 50],  
 [140, 50],  
 [140, 100],  
 [180, 100],  
 [180, 50],  
 [220, 50],  
 [220, 100],  
 [260, 100],  
 [260, 50],  
 [300, 50],  
 [300, 150],  
 [240, 150],  
 [275, 325],  
 [300, 325],  
 [300, 350],  
 [100, 350],  
], numpy.int32)  
poly = cv2.polylines(img, [pts], False, (255, 255, 255))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', poly)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
pts = numpy.array([  
 [100, 50],  
 [140, 50],  
 [140, 100],  
 [180, 100],  
 [180, 50],  
 [220, 50],  
 [220, 100],  
 [260, 100],  
 [260, 50],  
 [300, 50],  
 [300, 150],  
 [240, 150],  
 [275, 325],  
 [300, 325],  
 [300, 350],  
 [100, 350],  
 [100, 325],  
 [125, 325],  
 [160, 150],  
 [100, 150],  
], numpy.int32)  
poly = cv2.polylines(img, [pts], True, (255, 255, 255))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', poly)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

7. Нарисовать в нем текстовую строку “Hello, world!” различными шрифтами и различным цветом

import cv2  
import numpy  
  
img = numpy.zeros((400, 400, 3), numpy.uint8)  
window\_name = 'Image'  
text = 'Hello World!'  
org1 = (50, 50)  
org2 = (50, 100)  
org3 = (50, 150)  
fontFace1 = cv2.FONT\_ITALIC  
fontFace2 = cv2.FONT\_HERSHEY\_PLAIN  
fontFace3 = cv2.FONT\_HERSHEY\_COMPLEX  
fontScale = 1  
  
text1 = cv2.putText(img, text, org1, fontFace1, fontScale, (0, 255, 0))  
text2 = cv2.putText(img, text, org2, fontFace2, fontScale, (255, 0, 0))  
text3 = cv2.putText(img, text, org3, fontFace3, fontScale, (0, 0, 255))  
  
cv2.namedWindow('image', cv2.WINDOW\_NORMAL)  
cv2.imshow('image', text1)  
cv2.imshow('image', text2)  
cv2.imshow('image', text3)  
cv2.waitKey(0)  
cv2.destroyAllWindows()