Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №2

Дисциплина: Методы и средства цифровой обработки информации

Выполнил
студент гр. 13541/2 _____ Зобков Д. А.
(подпись)

Преподаватель _____ Абрамов Н. А.
(подпись)

“___”_____ 2017 г.

Санкт-Петербург
2017 г.

СОДЕРЖАНИЕ

1 Техническое задание

– Подобрать черно-белое изображение;
– Применить операторы определения границ на изображении.

2 Ход работы

Для выполнения работы был разработан класс EdgeFinder с использованием библиотек OpenCV и Matplotlib на языке Python (листинг А.1 на с. 7). Пример его использования приведен в листинге А.2, а использованные зависимости для virtualenv — в листинге А.3. Все три оператора (Превитта, Собеля, Робертса) вызываются с помощью метода `findEdges`.

Исходное изображение представлено на рис. 2.1.

Рис. 2.1. Черно-белое изображение

2.1 Оператор Собеля

Применим к изображению оператор Собеля (рис. 2.2).

Рис. 2.2. Оператор свертки Собеля

2.2 Оператор Превитта

Применим к изображению оператор Превитта (рис. 2.3 на следующей странице).

Рис. 2.3. Оператор свертки Превитта

2.3 Оператор Робертса

Применим к изображению оператор Робертса (рис. 2.4 на следующей странице).

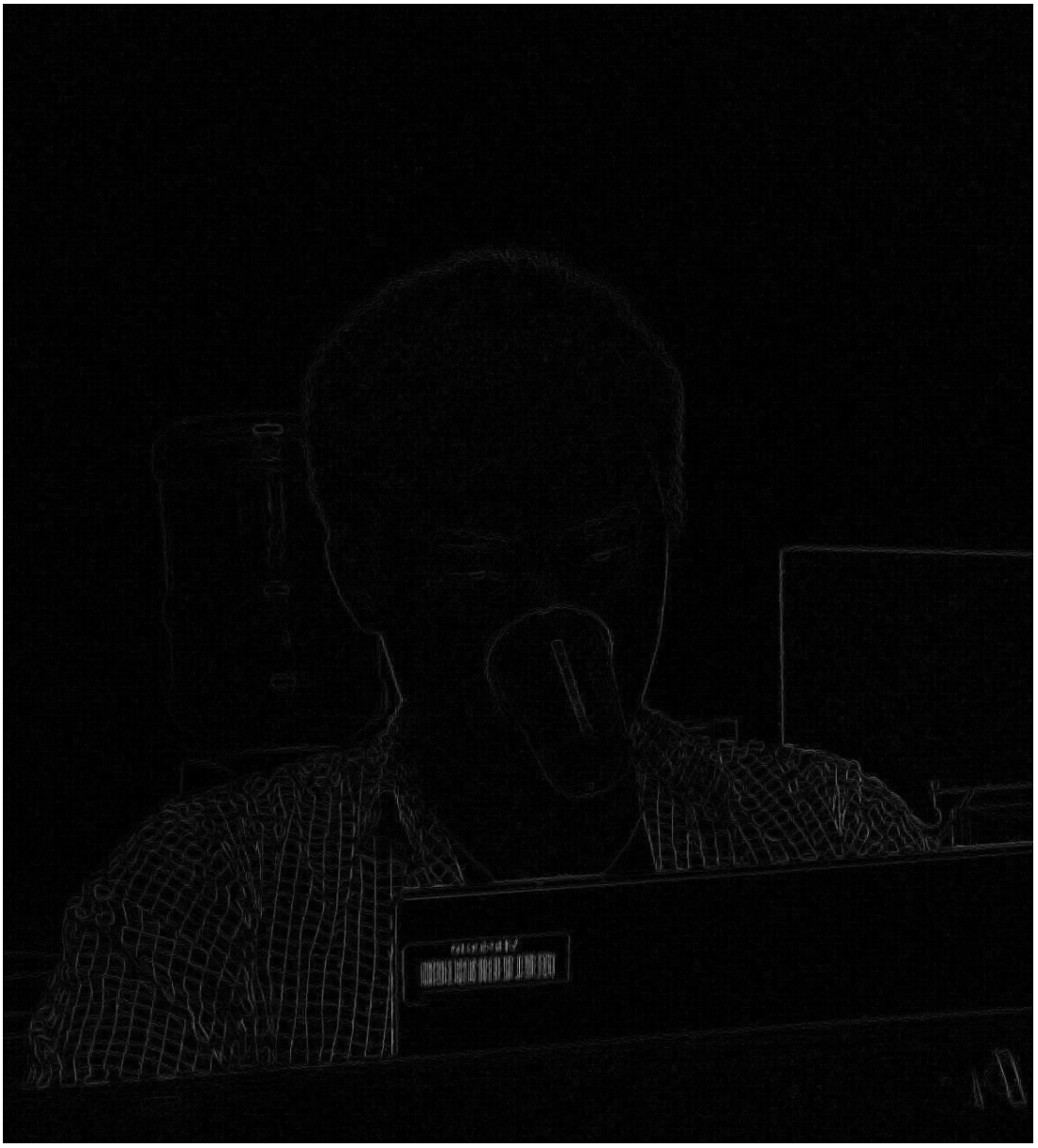

Рис. 2.4. Оператор свертки Робертса

3 Выводы

В ходе выполнения данной лабораторной работы к изображению были применены три оператора выделения контуров.

Оператор Собеля отличается самыми толстыми контурами, позволяя очень ярко и контрастно выделять линии. При этом выделение мелких объектов в данном операторе работает очень плохо. Лучше всего данный оператор проявляет себя при выделении вертикальных и диагональных линий.

Оператор Превитта также хорошо справляется с вертикальными и диагональными линиями. При обработке вертикальных линий данный оператор позволяет увидеть их даже в местах с большим количеством шумов. Однако мелкие детали и объекты выделяются также не очень хорошо, из-за они превращаются в шум.

Оператор Робертса, по сравнению с операторами Превитта и Собеля, отличается наименее тонким выделением границ, но в то же время, он более четко выделяет мелкие, сгруппированные объекты и диагональные линии.

Приложение А Исходный код программы

```
#!/usr/bin/env python

import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
from math import sqrt

class EdgeFinder:
    """
    Class for edge detection on greyscale image

    :ivar    inImg:    Loaded image, converted to greyscale
    :vartype inImg:    numpy
    :ivar    height:   Height of image
    :vartype height:   int
    :ivar    width:    Width of image
    :vartype width:    int
    :ivar    origImg:  Backup of original greyscale image
    :vartype origImg:  numpy
    """
    def __init__(self, inImg):
        # Load image
        self.inImg = cv.imread(inImg)
        # Rows
        self.height = self.inImg.shape[0]
        # Columns
        self.width = self.inImg.shape[1]
        # RGB to Greyscale image
        if len(self.inImg.shape) == 3: self.toGreyscale()
        # Greyscale image backup
        self.origImg = self.inImg

    def toGreyscale(self):
        """
        Converting RGB image to greyscale
        """
        # Get color arrays
        red   = self.inImg[...,2]
        green = self.inImg[...,1]
        blue  = self.inImg[...,0]

        # Fill array with shades of grey
        outImg = np.zeros((self.height, self.width))
        outImg[...] = 0.299 * red + 0.587 * green + 0.114 * blue

        # Round result shades
        outImg = np.round(outImg)

        # Update image
        self.inImg = outImg
```

```
52      def _expandImage(self):
            """
54          Creating a border around of image with values of nearest pixels

56          :return: Image with border around it
            :rtype:  numpy
58          """
            expImg = np.zeros((self.height + 2, self.width + 2))
60          # Fill center of image with original image
            expImg[1:-1,1:-1] = self.inImg
62          # Fill borders of expanded image with borders of original image
            # (except angular pixels)
64          expImg[1:-1,0] = self.inImg[...,0]
            expImg[1:-1,-1] = self.inImg[...,-1]
66          expImg[0,1:-1] = self.inImg[0,...]
            expImg[-1,1:-1] = self.inImg[-1,...]
68          # Fill angular pixels
            expImg[0,0] = self.inImg[0,0]
70          expImg[0,-1] = self.inImg[0,-1]
            expImg[-1,0] = self.inImg[-1,0]
72          expImg[-1,-1] = self.inImg[-1,-1]

74          return expImg

76      def findEdges(self, operator):
            """
78          Create image with detected edges using Sobel, Prewitt or
    Roberts operator

80          :param operator: Operator for edge detection ("sobel", "prewitt
    " or "roberts")
            :type  operator: str
82          """
            # Create arrays for derivative approximations
84          Gx = np.zeros((self.height, self.width))
            Gy = np.zeros((self.height, self.width))
86
            # Expand image with border around it
88          expImg = self._expandImage()

90          # Fill Gx and Gy
            if operator == "sobel":
92              Gx, Gy = self._sobelEdges(expImg, Gx, Gy)
            elif operator == "prewitt":
94              Gx, Gy = self._prewittEdges(expImg, Gx, Gy)
            elif operator == "roberts":
96              Gx, Gy = self._robertsEdges(expImg, Gx, Gy)
            else:
98              raise ValueError('Undefined operator: {}. Available
    operators: "sobel", "prewitt" or "roberts"'.format(type))

100         # Create image with detected edges
            self._findG(Gx, Gy)
102
```

```
    @staticmethod
104 def _sobelEdges(expImg, Gx, Gy):
        """
106     Getting horizontal and vertical derivative approximations with
    using of Sobel operator

108     :param expImg: Image expanded with borders
        :type  expImg: numpy
110     :param Gx:     Horizontal derivative approximation array/image
        :type  Gx:     numpy
112     :param Gy:     Vertical derivative approximation array/image
        :type  Gy:     numpy
114     :return:       Gx, Gy
        :rtype:        numpy, numpy
116     """
        Gx[...] = expImg[0:-2,0:-2] - expImg[0:-2,2:] + 2*(expImg
    [1:-1,0:-2] - expImg[1:-1,2:]) + expImg[2:,0:-2] - expImg[2:,2:]
118     Gy[...] = expImg[2:,0:-2] - expImg[0:-2,0:-2] + 2*(expImg
    [2:,1:-1] - expImg[0:-2,1:-1]) + expImg[2:,2:] - expImg[0:-2,2:]

120     return Gx, Gy

122  @staticmethod
     def _prewittEdges(expImg, Gx, Gy):
124     """
        Getting horizontal and vertical derivative approximations with
    using of Prewitt operator
126
        :param expImg: Image expanded with borders
128     :type  expImg: numpy
        :param Gx:     Horizontal derivative approximation array/image
130     :type  Gx:     numpy
        :param Gy:     Vertical derivative approximation array/image
132     :type  Gy:     numpy
        :return:       Gx, Gy
134     :rtype:        numpy, numpy
        """
136     Gx[...] = expImg[0:-2,0:-2] - expImg[0:-2,2:] + expImg
    [1:-1,0:-2] - expImg[1:-1,2:] + expImg[2:,0:-2] - expImg[2:,2:]
        Gy[...] = expImg[2:,0:-2] - expImg[0:-2,0:-2] + expImg[2:,1:-1]
     - expImg[0:-2,1:-1] + expImg[2:,2:] - expImg[0:-2,2:]
138
        return Gx, Gy
140
     @staticmethod
142  def _robertsEdges(expImg, Gx, Gy):
        """
144     Getting horizontal and vertical derivative approximations with
    using of Roberts operator

146     :param expImg: Image expanded with borders
        :type  expImg: numpy
148     :param Gx:     Horizontal derivative approximation array/image
        :type  Gx:     numpy
```

```
        :param Gy:      Vertical derivative approximation array/image
        :type  Gy:      numpy
        :return:        Gx, Gy
        :rtype:         numpy, numpy
        """
        Gx[...] = expImg[1:-1,1:-1] - expImg[2:,2:]
        Gy[...] = expImg[1:-1,2:] - expImg[2:,1:-1]

        return Gx, Gy

    def _findG(self, Gx, Gy):
        """
        Create result image using filled horizontal and vertical
    derivative approximations
        """
        # Get absolute values
        Gx = np.absolute(Gx)
        Gy = np.absolute(Gy)

        self.inImg = Gx + Gy

    def saveImage(self, path):
        """
        Saving image to file

        :param path: Save path
        :type  path: str
        """
        cv.imwrite(path, self.inImg)

    def restoreImage(self):
        """
        Restoring original greyscale image and shade map from backup
        """
        self.inImg = self.origImg
```

Листинг А.1. edgefinder.py

```
#!/usr/bin/env python

from edgefinder import EdgeFinder
import os, sys

# Create output directory
dirname = "out"
if not os.path.exists(dirname): os.mkdir(dirname)

# Load image
if len(sys.argv) > 1:
        img = EdgeFinder(sys.argv[1])
else:
        img = EdgeFinder("test.jpg")
```

```
16 imgName  = [dirname + "/{}.png".format(i) for i in ["grey", "sobel", "
      prewitt", "roberts"]]

18 # Save greyscale image
   img.saveImage(imgName[0])
20
   # Use operators and save results to files
22 img.findEdges("sobel")
   img.saveImage(imgName[1])
24
   img.restoreImage()
26 img.findEdges("prewitt")
   img.saveImage(imgName[2])
28
   img.restoreImage()
30 img.findEdges("roberts")
   img.saveImage(imgName[3])
```

Листинг А.2. main.py

```
1 cycler==0.10.0
2 matplotlib==2.0.2
  numpy==1.12.1
4 opencv-python==3.3.0.10
  pyparsing==2.2.0
6 python-dateutil==2.6.1
  pytz==2017.2
8 six==1.10.0
```

Листинг А.3. Файл с использованными зависимостями для virtualenv