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

Отчет по лабораторной работе №3

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;
- Провести для него сглаживание по Гауссу;
- Определить границы сглаженного изображения с помощью Лапласиана;
- Добиться повышения резкости границ исходного изображения, используя найденные границы из предыдущего шага.

2 Ход работы

Для выполнения работы был разработан класс EdgeStrength с использованием библиотек OpenCV и Matplotlib на языке Python (листинг А.1 на с. 8). Пример его использования приведен в листинге А.2, а использованные зависимости для virtualenv — в листинге А.3.

Исходное изображение представлено на рис. 2.1 на следующей странице.

Рис. 2.1. Черно-белое изображение

2.1 Сглаживание по Гауссу

Проведем сглаживание исходного изображения по Гауссу (рис. 2.2 на следующей странице), применив ядро свёртки $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$. Для этого воспользуемся методом `gaussSmoothing` класса EdgeStrength.

Рис. 2.2. Сглаживание по Гауссу

2.2 Определение границ

Проведем определение границ с помощью Лапласиана (рис. 2.3 на следующей странице), применив ядро свёртки $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$. Для этого воспользуемся методами `_laplacian` и `_lsLaplacian` (для линейного сжатия лапласиана) класса EdgeStrength.

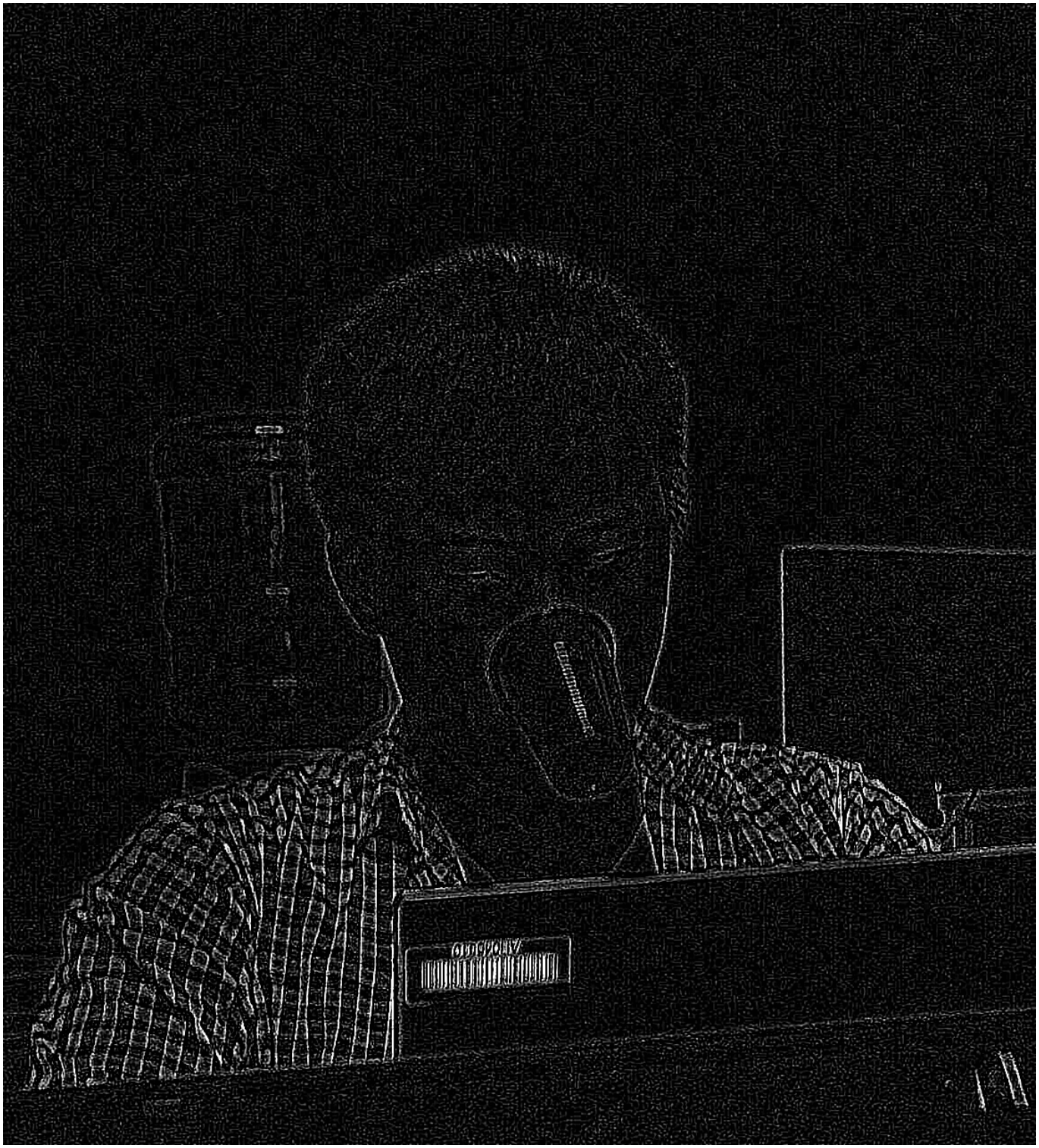

Рис. 2.3. Определение границ Лапласианом

2.3 Повышение резкости границ изображения

Проведем повышение резкости границ изображения (рис. 2.4 на следующей странице) путем умножения пикселей исходного изображения на определенные Лапласианом границы. Для этого воспользуемся методом `strengtheningEdges` класса EdgeStrength.

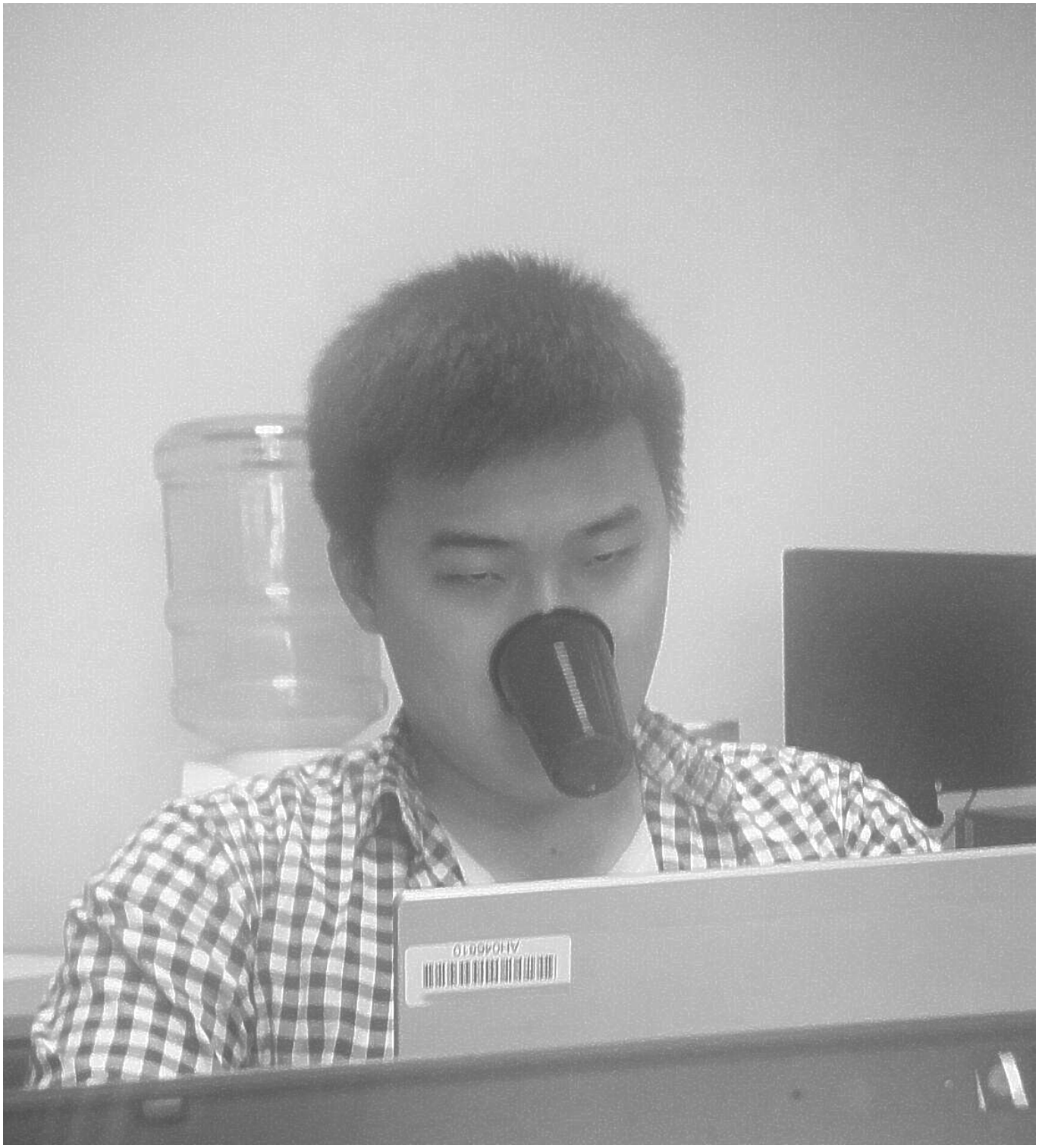

Рис. 2.4. Повышение резкости границ изображения

3 Выводы

В данной работе было проведено сглаживание по Гауссу, определение границ с помощью Лапласиана, а затем совмещение полученного изображения с исходным для повышения резкости границ изображения.

В результате проделанных действий получилось изображение с более четкими границами, но вследствие этого границы стали выглядеть зашумленными.

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 EdgeStrength:
    """
    Class for edge strengthening 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
```

```
    def _expandImage(self, img = None):
        """
        Creating a border around of image with values of nearest pixels

        :return: Image with border around it
        :rtype:  numpy
        """
        if img is None:
            img = self.inImg
        height = img.shape[0]
        width  = img.shape[1]
        expImg = np.zeros((height + 2, width + 2))

        # Fill center of image with original image
        expImg[1:-1,1:-1] = img
        # Fill borders of expanded image with borders of original image
        # (except angular pixels)
        expImg[1:-1,0] = img[...,0]
        expImg[1:-1,-1] = img[...,-1]
        expImg[0,1:-1] = img[0,...]
        expImg[-1,1:-1] = img[-1,...]
        # Fill angular pixels
        expImg[0,0] = img[0,0]
        expImg[0,-1] = img[0,-1]
        expImg[-1,0] = img[-1,0]
        expImg[-1,-1] = img[-1,-1]

        return expImg

    def gaussSmoothing(self):
        """
        Gauss smoothing operator with 3x3 square
        """
        # Expand image with border around it
        expImg = self._expandImage()

        self.inImg = (expImg[0:-2,0:-2] + 2*expImg[0:-2,1:-1] + expImg
    [0:-2,2:] +
            2*expImg[1:-1,0:-2] + 4*expImg[1:-1,1:-1] + 2*expImg
    [1:-1,2:] +
            expImg[2:,0:-2] + 2*expImg[2:,1:-1] + expImg[2:,2:]) / 16

    def gaussSmoothingTwice(self):
        """
        Gauss smoothing operator with 5x5 square
        """
        # Expand image with border around it
        expImg = self._expandImage(self._expandImage())

        self.inImg = (expImg[0:-4,0:-4] + 2*expImg[0:-4,1:-3] + 4*
    expImg[0:-4,2:-2] + 2*expImg[0:-4,3:-1] + expImg[0:-4,4:] +
            2*expImg[1:-3,0:-4] + 4*expImg[1:-3,1:-3] + 8*expImg
    [1:-3,2:-2] + 4*expImg[1:-3,3:-1] + 2*expImg[1:-3,4:] +
```

```
            4*expImg[2:-2,0:-4] + 8*expImg[2:-2,1:-3] + 16*expImg
    [2:-2,2:-2] + 8*expImg[2:-2,3:-1] + 4*expImg[2:-2,4:] +
            2*expImg[3:-1,0:-4] + 4*expImg[3:-1,1:-3] + 8*expImg
    [3:-1,2:-2] + 4*expImg[3:-1,3:-1] + 2*expImg[3:-1,4:] +
            expImg[4:,0:-4] + 2*expImg[4:,1:-3] + 4*expImg[4:,2:-2] +
    2*expImg[4:,3:-1] + expImg[4:,4:]) / 100

     def laplacianEdges(self, coef):
         """
         Use Laplacian operator to find edges of image and lower
    Laplacian shade range to 0 - coef

         :param coef: Maximum value of Laplacian shades
         :type  coef: float
         """
         self._laplacian()
         self._lsLaplacian(coef)

     def _laplacian(self):
         """
         Getting second-order derivative approximations with using of
    Laplacian operator
         """
         # Expand image with border around it
         expImg = self._expandImage()

         self.inImg = - expImg[0:-2,0:-2] - expImg[0:-2,1:-1] - expImg
    [0:-2,2:] - expImg[1:-1,0:-2] + 8*expImg[1:-1,1:-1] - expImg
    [1:-1,2:] - expImg[2:,0:-2] - expImg[2:,1:-1] - expImg[2:,2:]

     # def _lsLaplacian(self, coef = np.finfo(np.float64).max / 32786):
     def _lsLaplacian(self, coef = 1024):
         """
         Lower Laplacian shade range to 0 - coef

         :param coef: Maximum value of Laplacian shades
         :type  coef: float
         """
         self.inImg[self.inImg < 0] = 0
         self._linearStretching(coef)
         self.inImg[...] += 1

     def _linearStretching(self, coef):
         """
         Performing linear stretching on greyscale image
         """
         # Get bounds of shades
         a = np.amin(self.inImg)
         b = np.amax(self.inImg)
         # print(b)

         # Set bounds of new shades
         c = 0
         d = coef
```

```
148
        # Linear stretching
150     resImg = np.zeros((self.height, self.width))
        resImg[...] = (self.inImg[...] - a) * ((d - c) / (b - a)) + c
152
        # Update image
154     self.inImg = resImg

156 def strengtheningEdges(self):
        """
158     Strengthening edges on greyscale image
        """
160     self.inImg[...] = self.inImg[...] * self.origImg[...]

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

166     :param path: Save path
        :type  path: str
168     """
        cv.imwrite(path, self.inImg)
170
    def restoreImage(self):
172     """
        Restoring original greyscale image and shade map from backup
174     """
        self.inImg = self.origImg
```

Листинг А.1. edgestrength.py

```
1  #!/usr/bin/env python
2
   from edgestrength import EdgeStrength
4  import os, sys

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

10 # Load image
   if len(sys.argv) > 1:
12         img = EdgeStrength(sys.argv[1])
   else:
14         img = EdgeStrength("test.jpg")

16 imgName  = [dirname + "/{}.png".format(i) for i in ["1_grey", "2_lap",
      "3_gauss3", "4_lap3", "5_res3", "6_gauss5", "7_lap5", "8_res5"]]

18 # Save greyscale image
   img.saveImage(imgName[0])
20
   # Laplacian without smoothing
22 img._laplacian()
```

```
img._lsLaplacian()
img.saveImage(imgName[1])
img.restoreImage()

# Gauss smoothing 3x3
img.gaussSmoothing()
img.saveImage(imgName[2])

# Laplacian with smoothing 3x3
img._laplacian()
img._lsLaplacian()
img.saveImage(imgName[3])

# Result with 3x3
img._lsLaplacian(0.7)
img.strengtheningEdges()
img.saveImage(imgName[4])
img.restoreImage()

# Gauss smoothing 5x5
img.gaussSmoothingTwice()
img.saveImage(imgName[5])

# Laplacian with smoothing 5x5
img._laplacian()
img._lsLaplacian()
img.saveImage(imgName[6])

# Result with 5x5
img._lsLaplacian(0.7)
img.strengtheningEdges()
img.saveImage(imgName[7])
```

Листинг А.2. main.py

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

Листинг А.3. Файл с использованными зависимостями для virtualenv