МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики  
Кафедра технической кибернетики

**Отчет по лабораторной работе №3**

Дисциплина: «Системы обработки изображений»

Тема: **«Обработка бинарных изображений»**

Выполнил: Пшенин В.И.

Группа: 6133-010402D

Самара 2020

**Задание на лабораторную работу**

1. Фильтрация бинарных изображений
   1. Создать или выбрать бинарное изображение с любым простым рисунком, состоящим из примитивных фигур (круги, квадраты, полигоны).
   2. На полученное изображение наложить шум (1%, 2%, 5%, 10%, 20%, 50%).
   3. Выполнить операции эрозии и дилатации для различных структурных элементов (крест, квадрат). Визуально оценить результаты эрозии и дилатации.
   4. Провести фильтрацию шумов на изображении при помощи морфологических операций вскрытия и закрытия с различными размерами (от 3 до 5) и формами структурного элемента (крест, квадрат), а также алгоритма фильтрации шумов, которому соответствует логическая таблица истинности, представленная в табл. 1 со крестообразным структурным элементом размером 3.
   5. Рассчитать коэффициент снижения шума для различных методов фильтрации, для различной интенсивности шума и различных размеров и форм структурного элемента (окна).
   6. Сравнить результаты фильтрации.
2. Выделение контуров на бинарных изображениях
   1. Создать или выбрать бинарное изображение, содержащее простой рисунок, состоящий из примитивных фигур (полосы, круги, квадраты).
   2. С помощью морфологических операций выделить контур объекта. Выяснить, когда контур получается внешним, внутренним, четырёхсвязным, восьмисвязным.
   3. Выделить горизонтальные контуры на изображении.
   4. Выделить вертикальные контуры на изображении.
3. Прокомментировать полученные результаты.

**Задание 1 - Фильтрация бинарных изображений**

Создать или выбрать бинарное изображение с любым простым рисунком, состоящим из примитивных фигур (круги, квадраты, полигоны).

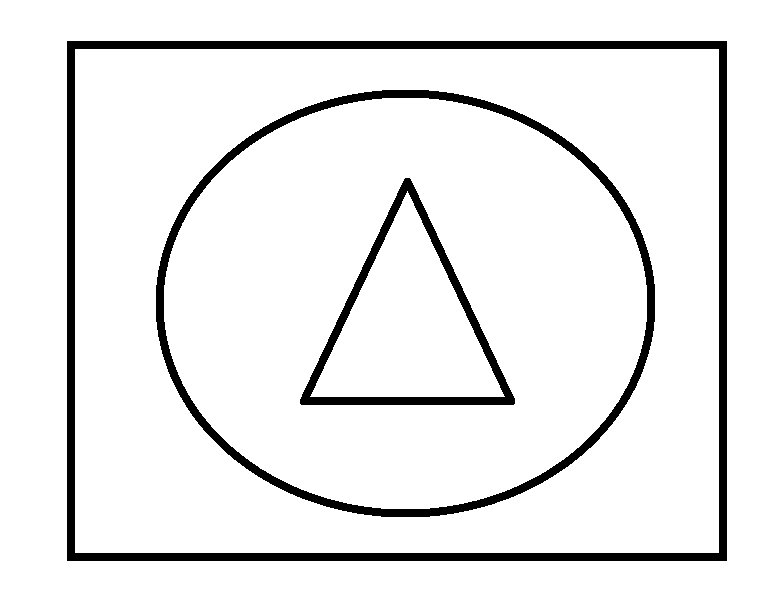


Рисунок 1 – Входное изображение

На полученное изображение наложить шум (1%, 2%, 5%, 10%, 20%, 50%).

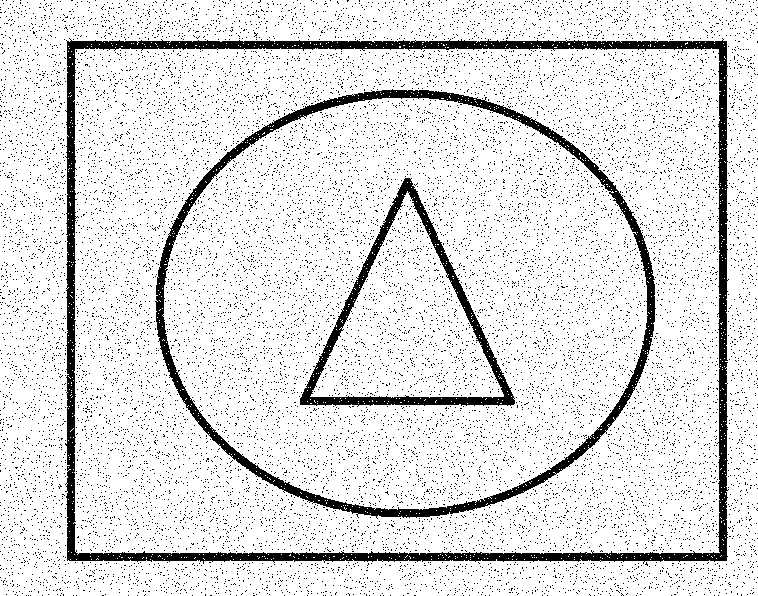


Рисунок 2 – Входное изображение с шумом 5%

Выполнить операции эрозии и дилатации для различных структурных элементов (крест, квадрат). Визуально оценить результаты эрозии и дилатации.

|  |  |
| --- | --- |
| **C:\Users\PVI\Desktop\3 лаба\эрод крест.png** | C:\Users\PVI\Desktop\3 лаба\эрод квадрат.png |
| Рисунок 3 – Операция эрозии структурными элементами крест (слева), квадрат (справа) | |
|  | |
| C:\Users\PVI\Desktop\3 лаба\дилат крест.png | C:\Users\PVI\Desktop\3 лаба\дилат квадрат.png |
| Рисунок 4 – Операция дилатации структурными элементами крест (слева), квадрат (справа) | |

Как видно из рисунков, применение эрозии привело к сужению белых областей, дилатации – расширению. Операция дилатации удалила все шумы с избражения, но вместе с частями исходного.

Провести фильтрацию шумов на изображении при помощи морфологических операций вскрытия и закрытия с различными размерами (от 3 до 5) и формами структурного элемента (крест, квадрат) (рисунок 5 и 6), а также алгоритма фильтрации шумов, которому соответствует логическая таблица истинности (рисунок 7).

|  |  |
| --- | --- |
| C:\Users\PVI\Desktop\3 лаба\3 на 3 крест.png | C:\Users\PVI\Desktop\3 лаба\3 на 3 квадрат.png |
| Рисунок 5 – Операция вскрытия и закрытия структурными элементами (3x3) крест (слева), квадрат (справа) | |
|  | |
| C:\Users\PVI\Desktop\3 лаба\5 на 5 крест.png | C:\Users\PVI\Desktop\3 лаба\5 на 5 квадрат.png |
| Рисунок 6 – Операция вскрытия и закрытия элементами (5x5) крест (слева), квадрат (справа) | |

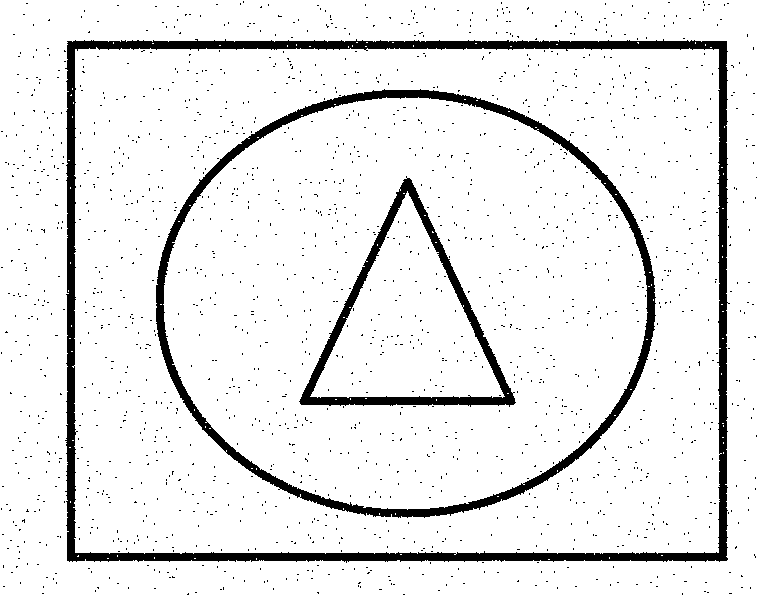


Рисунок 7 – Фильтрация алгоритмом с таблицей истинности структурным элементом (3x3) крест

Рассчитать коэффициент снижения шума для различных методов фильтрации, для различной интенсивности шума и различных размеров и форм структурного элемента (окна).

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Размер окна | Метод фильтрации | Структурный элемент | Шум, % |
| 3x3 | Вскрытие + закрытие | Крест | 0,57 |
| Таблица истинности | 1,07 |
| Вскрытие + закрытие | Квадрат | 0,94 |
| 5x5 | Вскрытие + закрытие | Крест | 1,11 |
| Вскрытие + закрытие | Квадрат | 7,88 |

Для данного изображения увелечение размера окна не приводит к снижению шума, а в качестве структурного элемента лучше выбирать крест. Наибольшее снижение шума удалось достигнуть при фильтрации методом выскрытия и закрытия окном 3x3 структурным элементом крест, процент шума на изображении до фильтрации – 5%, после фильтрации – 0,57%.

**Задание 2 - Выделение контуров на бинарных изображениях**

Создать или выбрать бинарное изображение, содержащее простой рисунок, состоящий из примитивных фигур (полосы, круги, квадраты).

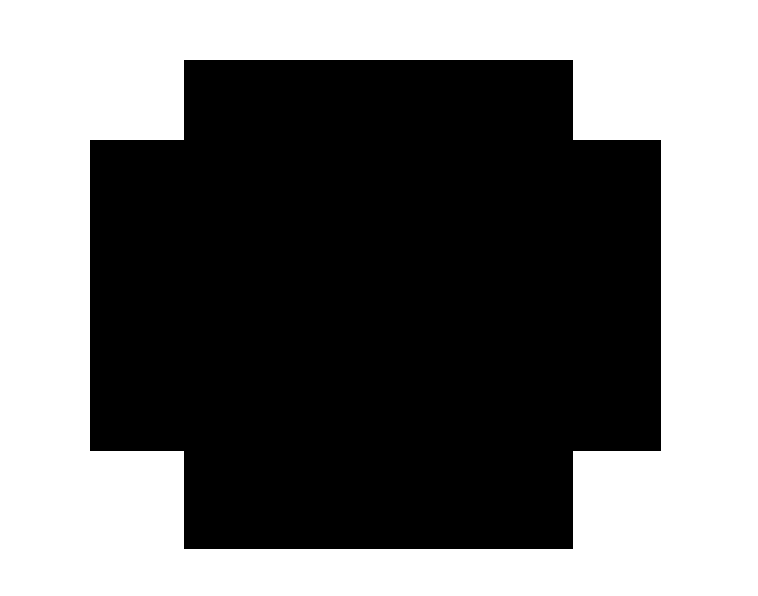


Рисунок 8 – Входное изображение

Выделить контур объекта можно, взяв сумму по модулю 2 результирующего изображения после операции дилатации или эрозии исходного изображения с исходным изображением, результат на рисунке 9.

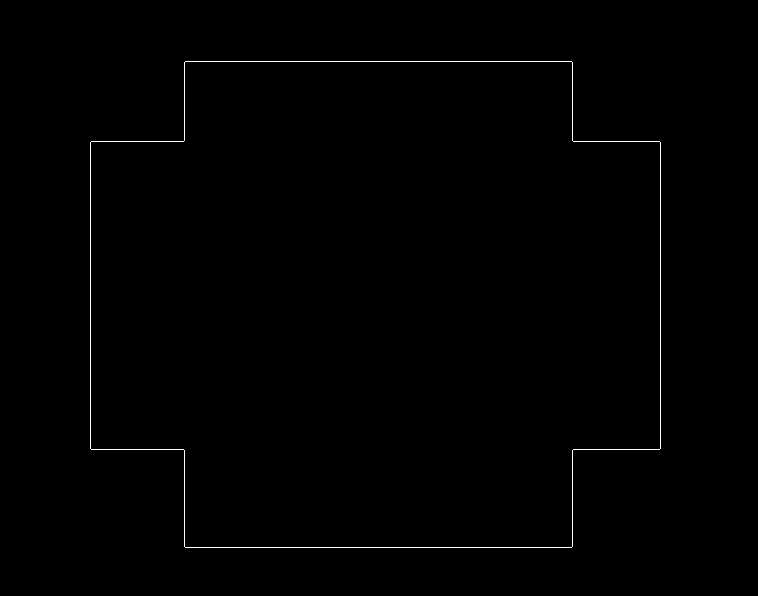


Рисунок 9 – Контур объекта

Четырехсвязным называется фильтр, в которой от каждого пикселя можно «добраться» до любого другого, переходя от пикселя к пикселю в двух горизонтальных и двух вертикальных направлениях, т.е. вправо, влево, вверх и вниз. В восьмисвязном фильтре это же можно сделать, добавив к указанным направлениям переходов еще четыре диагональных: вправо и вверх, вправо и вниз, влево и вверх, влево и вниз. Отметим также то, что граница четырехсвязного фильтра является восьмисвязной, а граница восьмисвязного фильтра – четырехсвязной.

Выделить внутренний контур объекта можно, взяв сумму по модулю 2 результирующего изображения после операции эрозии исходного изображения с исходным изображением.

Выделить внешний контур объекта можно, взяв сумму по модулю 2 результирующего изображения после операции дилатции исходного изображения с исходным изображением.

Выделить вертикальные контуры на изображении.

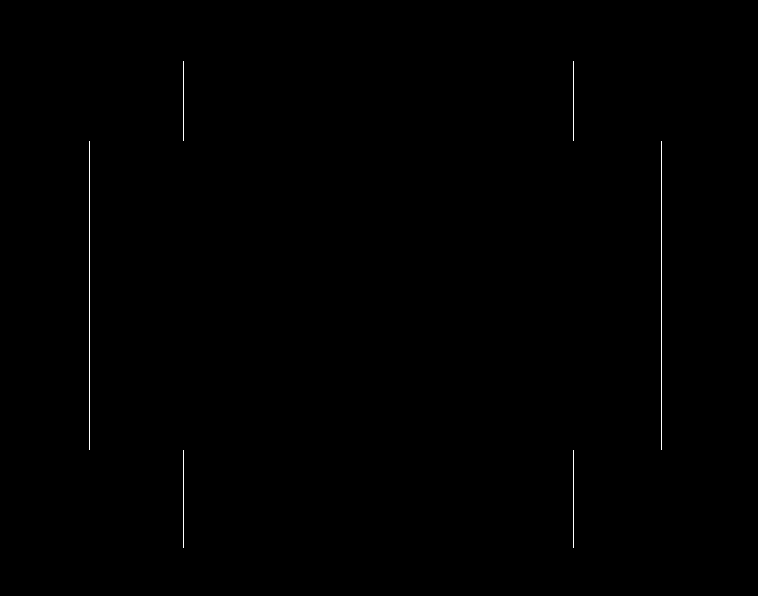


Рисунок 10 – Вертикальные контуры объекта

Выделить горизонтальные контуры на изображении.

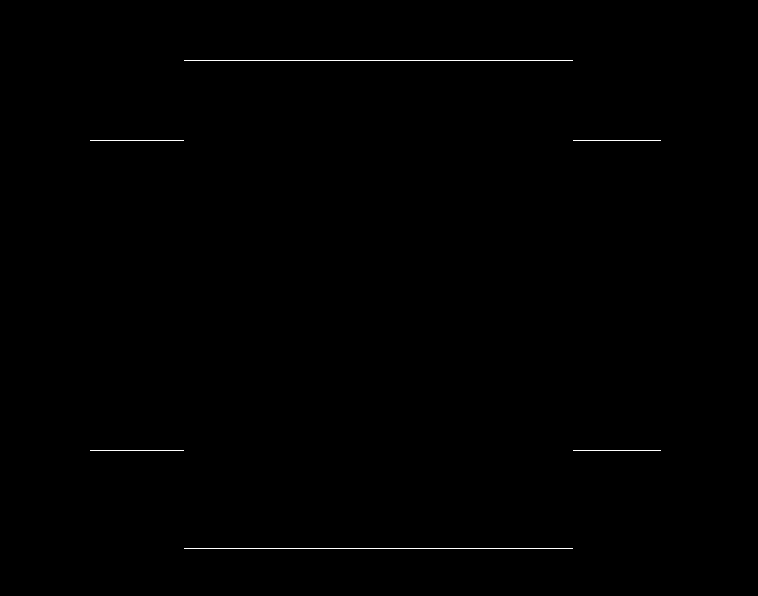


Рисунок 10 – Горизонтальные контуры объекта

**Задание 3**

Для фильтрации шумов на изображении нужно выбирать метод с таблицей истинности, если важна скорость работы, по эффективности наилучшим образом себя показал метод выскрытия и закрытия окном 3x3 структурным элементом крест. Для выделения контура объекта можно, взять сумму по модулю 2 результирующего изображения после операции дилатации или эрозии исходного изображения.

**ПРИЛОЖЕНИЕ А**

**Код программы**

from PIL import Image, ImageChops

import random

import numpy as np

# In[68]:

inputImg = Image.open("test1.bmp").convert('1')

img = Image.open("test1.bmp").convert('1')

# In[69]:

img

# In[40]:

def perOfNoise(img1,img2):

pixels1 = img1.load()

pixels2 = img2.load()

diffPix = 0

for i in range(img1.size[0]):

for j in range(img1.size[1]):

if (pixels1[i,j]!=pixels2[i,j]):

diffPix += 1

return diffPix/(img1.size[0]\*img1.size[1])\*100

# In[41]:

def myNoise(pixels,prob):

for i in range(img.size[0]):

for j in range(img.size[1]):

rdn = random.random()

if rdn < prob:

pixels[i,j] = 0

elif rdn > (1 - prob):

pixels[i,j] = 255

# In[42]:

def myNoisePer(pixels,per):

countOfPixels = int(img.size[0]\*img.size[1]/100\*per)

for \_ in range(countOfPixels):

i = random.randint(0,img.size[0]-1)

j = random.randint(0,img.size[1]-1)

if pixels[i,j] == 255:

pixels[i,j] = 0

else:

pixels[i,j] = 255

# In[43]:

myNoise(img.load(),0.05)

# In[44]:

img

# In[45]:

perOfNoise(img,inputImg)

# In[46]:

def erode(mask,frame):

frameFit = set()

for i in range(mask.size):

if (mask[i]==1):

frameFit.add(frame[i])

frameFit = list(frameFit)

if len(frameFit) == 1 and frameFit[0] == 255:

return 255

else:

return 0

# In[47]:

def dilate(mask,frame):

frameFit = set()

for i in range(mask.size):

if (mask[i]==1):

frameFit.add(frame[i])

frameFit = list(frameFit)

if len(frameFit) == 1 and frameFit[0] == 0:

return 0

else:

return 255

# In[48]:

def table(mask,frame,centre):

frameFit = set()

for i in range(mask.size):

if mask[i]==1 and i != centre:

frameFit.add(frame[i])

frameFit = list(frameFit)

if len(frameFit) == 1:

if frameFit[0] == 255:

return 255

else:

return 0

else:

return frame[centre]

# In[49]:

def myFilter(img,mask,mode):

pixels = img.load()

newImage = Image.new('1', (img.size[0],img.size[1]), color=255)

newImagePixels = newImage.load()

ijStart = mask.shape[0] // 2

iEnd = img.size[0]-ijStart

jEnd = img.size[1]-ijStart

mask = mask.flatten()

centre = mask.size // 2

if (mode == 'erode'):

for j in range(ijStart,jEnd):

for i in range(ijStart,iEnd):

frame = []

for jFrame in range(j-ijStart,j+ijStart+1):

for iFrame in range(i-ijStart,i+ijStart+1):

frame.append(pixels[iFrame,jFrame])

newImagePixels[i,j] = erode(mask,frame)

elif (mode == 'dilate'):

for j in range(ijStart,jEnd):

for i in range(ijStart,iEnd):

frame = []

for jFrame in range(j-ijStart,j+ijStart+1):

for iFrame in range(i-ijStart,i+ijStart+1):

frame.append(pixels[iFrame,jFrame])

newImagePixels[i,j] = dilate(mask,frame)

else:

for j in range(ijStart,jEnd):

for i in range(ijStart,iEnd):

frame = []

for jFrame in range(j-ijStart,j+ijStart+1):

for iFrame in range(i-ijStart,i+ijStart+1):

frame.append(pixels[iFrame,jFrame])

newImagePixels[i,j] = table(mask,frame,centre)

return newImage

# In[50]:

def opening(img, kernel):

return myFilter(myFilter(img,kernel,'erode'),kernel,'dilate')

def closing(img, kernel):

return myFilter(myFilter(img,kernel,'dilate'),kernel,'erode')

# In[51]:

def contour(img, mask, inner=True):

if inner:

return ImageChops.logical\_xor(img, myFilter(img,kernel,'erode'))

else:

return ImageChops.logical\_xor(img, myFilter(img,kernel,'dilate'))

# In[72]:

#kernel = np.array([[0,0,1,0,0],[0,0,1,0,0],[1,1,1,1,1],[0,0,1,0,0],[0,0,1,0,0]])

#kernel = np.array([[0,1,0],[1,1,1],[0,1,0]])

kernel = np.array([[0,0,0],[1,1,1],[0,0,0]])

#kernel = np.array([[0,1,0],[1,1,1],[0,1,0]])

# In[53]:

imgTable = myFilter(img,kernel,'ero12de')

# In[54]:

perOfNoise(imgTable,inputImg)

# In[55]:

imgOpening = opening(img,kernel)

imgOpening

perOfNoise(imgOpening,inputImg)

# In[57]:

closing(img,kernel)

# In[414]:

myFilter(closing(img,kernel),kernel,'erodef')

# In[411]:

closing(opening(closing(img,kernel),kernel),kernel)

# In[35]:

circle = Image.open("круг.bmp").convert('1')

circle2 = Image.open("круг2.bmp").convert('1')

fig = Image.open("фигура.bmp").convert('1')

fig

# In[73]:

contour(fig,kernel)

# In[64]:

myFilter(fig,kernel,'erode')

# In[67]:

contour(fig,kernel,inner=True)

# In[82]:

contour(circle2,kernel)