Лабораторная работа №2

«Основы обработки цифровых изображений/сжатие изображений»

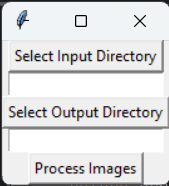
Козлов Юрий, 3 курс 2 группа

12 номер в списке

Номер в списке 12 - выполнял реализацию сглаживающих низкочастотных фильтров и локальную пороговую обработку.

По локальной пороговой обработке выполнил 2 метода: метод Бернсена и метод Ниблацка

Ввод, вывод и обработку изображений выполнял следующим образом:



По нажатию на кнопку «Select Input Directory» открывается file chooser и выбирается папка с изображениями, которые будут обрабатываться фильтрами.

По нажатию на кнопку «Select Output Directory» открывается file chooser и выбирается папка, куда будут записываться обработанные изображения.

Ниже представлен листинг программы:

# Local  
import cv2  
import numpy as np  
  
  
def bernsen\_thresholding(image, e=15, r=15):  
 # Копирование изображения для обработки  
 processed\_image = np.copy(image)  
  
 # Итерация по каждому квадрату  
 half\_r = r // 2  
 for x in range(half\_r, image.shape[1] - half\_r):  
 for y in range(half\_r, image.shape[0] - half\_r):  
 # Получение яркостей пикселей в пределах квадрата  
 pixel\_values = []  
 for i in range(x - half\_r, x + half\_r + 1):  
 for j in range(y - half\_r, y + half\_r + 1):  
 pixel\_values.append(image[j, i])  
  
 # Вычисление наименьшего и наибольшего уровня яркости  
 jlow = np.min(pixel\_values)  
 jhigh = np.max(pixel\_values)  
  
 # Вычисление порога  
 threshold = (jhigh - jlow) / 2  
  
 # Пороговая обработка пикселя  
 if threshold <= e:  
 processed\_image[y, x] = 0 # Замена пикселя на черный  
 else:  
 processed\_image[y, x] = 255 # Замена пикселя на белый  
  
 return processed\_image  
  
  
def niblack\_thresholding(image, r=15, k=-0.2):  
 # Копирование изображения для обработки  
 processed\_image = np.copy(image)  
  
 # Итерация по каждому пикселю  
 half\_r = r // 2  
 for x in range(half\_r, image.shape[1] - half\_r):  
 for y in range(half\_r, image.shape[0] - half\_r):  
 # Получение локальной окрестности  
 neighborhood = image[y - half\_r:y + half\_r + 1, x - half\_r:x + half\_r + 1]  
  
 # Вычисление среднего и среднеквадратического отклонения  
 mean\_value = np.mean(neighborhood)  
 std\_deviation = np.std(neighborhood)  
  
 # Вычисление порога  
 threshold = mean\_value + k \* std\_deviation  
  
 # Пороговая обработка пикселя  
 if image[y, x] <= threshold:  
 processed\_image[y, x] = 0 # Замена пикселя на черный  
 else:  
 processed\_image[y, x] = 255 # Замена пикселя на белый  
  
 return processed\_image  
  
def gaussian\_filter(image):  
 # Применение фильтра Гаусса с ядром размера 5х5 со стандартным отклонением 0  
 filtered\_image = cv2.GaussianBlur(image, (5, 5), 0)  
  
 return filtered\_image  
  
  
# # Загрузка изображения с помощью OpenCV  
# image = cv2.imread('img.png', cv2.IMREAD\_GRAYSCALE)  
#  
# # Применение адаптивной пороговой обработки  
# processed\_image = laplacian\_filter(image)  
#  
# # Сохранение обработанного изображения с помощью OpenCV  
# cv2.imwrite('processed\_image\_laplacian.png', processed\_image)  
import tkinter as tk  
from tkinter import filedialog  
import os  
  
def select\_input\_directory():  
 input\_directory = filedialog.askdirectory(title="Select Input Directory")  
 input\_entry.delete(0, tk.END)  
 input\_entry.insert(0, input\_directory)  
  
def select\_output\_directory():  
 output\_directory = filedialog.askdirectory(title="Select Output Directory")  
 output\_entry.delete(0, tk.END)  
 output\_entry.insert(0, output\_directory)  
  
def process\_images():  
 input\_directory = input\_entry.get().replace('/', '\\')  
 output\_directory = output\_entry.get().replace('/', '\\')  
  
 methods = [  
 bernsen\_thresholding,  
 niblack\_thresholding,  
 gaussian\_filter  
 ]  
  
 for method in methods:  
 method\_output\_directory = os.path.join(output\_directory, method.\_\_name\_\_)  
 os.makedirs(method\_output\_directory, exist\_ok=True)  
  
 image\_files = os.listdir(input\_directory)  
 for image\_file in image\_files:  
 print(image\_file)  
 if image\_file.endswith('.png') or image\_file.endswith('.jpg'):  
 image\_path = os.path.join(input\_directory, image\_file)  
 print(image\_path)  
 image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  
  
 # Процесс обработки изображения с использованием выбранного метода  
 processed\_image = method(image)  
  
 output\_image\_path = os.path.join(method\_output\_directory, image\_file)  
 cv2.imwrite(output\_image\_path, processed\_image)  
 print(method.\_\_name\_\_, " Completed\n")  
  
 print("Processing complete!")  
  
# Создание графического интерфейса  
root = tk.Tk()  
  
# Кнопка 1 - выбор директории с исходными изображениями в формате PNG  
button1 = tk.Button(root, text="Select Input Directory", command=select\_input\_directory)  
button1.pack()  
  
# Поле ввода для отображения выбранной директории  
input\_entry = tk.Entry(root)  
input\_entry.pack()  
  
# Кнопка 2 - выбор директории для сохранения обработанных изображений  
button2 = tk.Button(root, text="Select Output Directory", command=select\_output\_directory)  
button2.pack()  
  
# Поле ввода для отображения выбранной директории  
output\_entry = tk.Entry(root)  
output\_entry.pack()  
  
# Кнопка 3 - обработка изображений  
button3 = tk.Button(root, text="Process Images", command=process\_images)  
button3.pack()  
  
root.mainloop()

РЕЗУЛЬТАТЫ

1. Низкочастотные сглаживающие фильтры

Применил фильтр Гаусса с помощью функции GaussianBlur к изображению:



Результат:



1. Локальная пороговая обработка
2. Метод Бернсена

Применил к изображению:



Результат:



1. Метод Ниблацка

Применил к изображению:



Результат:

