МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



**Лабораторна робота №1**

**З дисципліни «Обробка зображень методами штучного інтелекту»**

***Виконав:*** *студент групи КН-408*

*Бурак Марко*

***Викладач:***

Пелешко Д. Д.

Львів – 2022

**Тема роботи**: Попередня обробка зображень.

**Мета роботи**: Вивчити просторову фільтрацію зображень, методи мінімізації шуму, морфології, виділення країв і границь та елементи бібліотеки OpenCV для розв’язання цих завдань.

**Теоретичні відомості.**

У світі комп'ютерного зору фільтрація зображень використовується для модифікації зображень на етапі поперднього опрацювання. Ці зміни, по суті, дозволяють прояснити зображення, щоб отримати потрібну інформацію. Фільтрація може включати в себе все, що завгодно - видобуток країв з зображення, його розмиття, видалення небажаних об'єктів тощо.

Існує багато причин для використання фільтрації зображень. Наприклад, зйомка при сонячному світлі або в темряві вплине на чіткість зображення, тому можливо необхідновикористовувати фільтри зображень, щоб змінити зображення згідно власних потреб. Аналогічно, зображення може бути розмитим або зашумленим, яке потребувати уточнення і фокусування.

Є такі методи фільтрації зображення – лінійна та нелінійна фільтрація.

До лінійної фільтрації зображення належать:

* 1D лінійна фільтрація зображення.
* 2D лінійна фільтрація зображення.
* Box фільтрація.

До нелінійної фільтрації зображення належать:

* Фільтр Гауса.
* Метод вирівнювання гістограми.
* Медіанна фільтрація зображення.

**Хід роботи**

Варіант 4. Вибрати з інтернету два зображення з різною деталізацією об’єктів та два зображення з різним контрастом. Без використання жодних бібліотек для обробки зображень (наприклад Open CV), виконати відповідне завдання (номер завдання вказано у рейтинговій таблиці).

**Завдання:** Виконати медіанну фільтрацію зображення з різними значеннями ядра. Провести порівняльний аналіз.

Алгоритм дій:

A picture containing text

Description automatically generated

* Створюється ядро матрицею NxN
* Застосувати матрицю для першого пікселя
* Визначити медіану з посортованих значень матриці
* Замінити значення пікселя на значення медіани
* Зсунути ядро на наступний піксель і повторити процес

На ілюстрації показано визначення пікселя з індексом [1,1] та ядром 3х3 та із початковим значенням пікселя – 0.

Формується масив значень [20,20,1,20,0,0,20,0,90].

Сортується масив – [0,0,0,20,20,20,20,90]

Медіана 20 – число з індексом 4, якщо починати відлік з 0.

Замінюємо значення 0 на число 20.

**Код програми:**

import random

import numpy

from PIL import Image

import cv2

import matplotlib.pyplot as plt

from skimage.util import random\_noise

def add\_noise(img):

# Getting the dimensions of the image

row, col = img.shape

# Randomly pick some pixels in the

# image for coloring them white

# Pick a random number between 300 and 10000

number\_of\_pixels = random.randint(300, 10000)

for i in range(number\_of\_pixels):

# Pick a random y coordinate

y\_coord = random.randint(0, row - 1)

# Pick a random x coordinate

x\_coord = random.randint(0, col - 1)

# Color that pixel to white

img[y\_coord][x\_coord] = 255

# Randomly pick some pixels in

# the image for coloring them black

# Pick a random number between 300 and 10000

number\_of\_pixels = random.randint(300, 10000)

for i in range(number\_of\_pixels):

# Pick a random y coordinate

y\_coord = random.randint(0, row - 1)

# Pick a random x coordinate

x\_coord = random.randint(0, col - 1)

# Color that pixel to black

img[y\_coord][x\_coord] = 0

return img

def median\_filter(data, filter\_size):

temp = []

indexer = filter\_size // 2

data\_final = numpy.zeros((len(data), len(data[0])))

for i in range(len(data)):

for j in range(len(data[0])):

for z in range(filter\_size):

if i + z - indexer < 0 or i + z - indexer > len(data) - 1:

for c in range(filter\_size):

temp.append(0)

else:

if j + z - indexer < 0 or j + indexer > len(data[0]) - 1:

temp.append(0)

else:

for k in range(filter\_size):

temp.append(data[i + z - indexer][j + k - indexer])

temp.sort()

data\_final[i][j] = temp[len(temp) // 2]

temp = []

return data\_final

def main():

filename = "apple"

kernel = 3

dir\_salt = "detailed\_salt"

dir\_denoised = "detailed\_denoised"

img = cv2.imread(filename+".jpg", cv2.IMREAD\_GRAYSCALE)

noisy\_gray = add\_noise(img)

cv2.imwrite(f'{dir\_salt}/salt-and-pepper-{filename}-{kernel}.jpg', img)

arr = numpy.array(noisy\_gray)

removed\_noise = median\_filter(arr, kernel)

cv2.imwrite(f"{dir\_denoised}/image-denoised-{filename}-{kernel}.jpg", removed\_noise)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Результат роботи програми:



A picture containing nature, water, waterfall

Description automatically generated



Рис. 1 Контрастні зображення із застосованою медіанною фільтрацією з ядром 2-3-5.







Рис. 2 Меншоконтрастні зображення із застосованою медіанною фільтрацією з ядром 2-3-5.







Рис. 3 Детальне зображення із застосованою медіанною фільтрацією з ядром 2-3-5.







Рис. 4 Менш детальне зображення із застосованою медіанною фільтрацію з ядром 2-3-5.

**Висновок:** після виконання цієї лабораторної роботи я навчився виконувати просторову фільтрацію зображення, освоїв методи мінімізації шуму, морфології, виділення країв і границь та елементи бібліотеки OpenCV для розв’язання цих завдань. За результатами практичної частини лабораторної можна зауважити, що медіанна фільтрація чудово видаляє шуми типу сіль та перець. Також даний алгоритм зберігає границі та чіткі лінії. В усіх випадках фільтрація пройшла досить вдало, проте, на менш контрастному зображенні проявляються часткові шуми, та черезнизький контраст вихідне зоюраження важче сприймається людині. Слід зауважити, що фільтрація з ядром 2 показала невтішні результати, адже багато шумів все ще залишилися на вихідному зображенні. Фільтрація з ядром 5 добре справилась з видаленням шумів, але дуже розмила зображення і знизила чіткість. Найкращим ж було використання ядра 3, яке залишило чіткість та видалило усі можливі шуми.