МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Изображение выглядит как текст

Автоматически созданное описание

**Лабораторна робота №1**

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

***Виконав:*** *студент групи КН-408*

*Левицького Богдана*

***Викладач:***

Пелешко Д. Д.

Львів – 2022

**Варіант 4**

**Тема**: Попередня обробка зображень.

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

**Хід роботи**

Логіка медіанного фільру є простою – створюємо кернел розміром NxN, застосовуємо фільтр до зображення та визначаємо медіану значень зображення, що наклалися на кернел. Саме таке значення застосовується до пікселя, який знаходться в центрі кернела.

Імплементуємо програму для нанесення Salt&Pepper шуму, а також медіанний фільтр та продемонструємо роботу програми на мові програмування Python без використання засобів OpenCV.

Изображение выглядит как дерево, природа, вода, водопад

Автоматически созданное описание

А)

Изображение выглядит как дерево, внешний, природа, водопад

Автоматически созданное описание

Б)

Изображение выглядит как природа, вода, водопад, внешний

Автоматически созданное описание

В)

Изображение выглядит как природа, водопад, вода

Автоматически созданное описание

Г)

Изображение выглядит как природа, вода, внутренний, водопад

Автоматически созданное описание

Д)

Изображение выглядит как природа, водопад, вода

Автоматически созданное описание

Е)

Рис. 1 Високо-деталізоване зображення: А) Оригінал зображення; Б) Зашумлене зображення з середньою кількістю шуму; В) Зображення з застосованим медіанним фільтром розміром 3; Г) Зображення з застосованим медіанним фільтром розміром 5; Д) Зображення з застосованим медіанним фільтром розміром 7; Е) Зображення з застосованим медіанним фільтром розміром 9.

Изображение выглядит как цветок, группа, несколько, ткань

Автоматически созданное описание

А)

Изображение выглядит как земля

Автоматически созданное описание

Б)

Изображение выглядит как дым, пар, внешний, облака

Автоматически созданное описание

В)

Изображение выглядит как внешний, дым, пар, облака

Автоматически созданное описание

Г)

Изображение выглядит как внешний, закрыть

Автоматически созданное описание

Д)

Изображение выглядит как в позе, закрыть

Автоматически созданное описание

Е)

Рис. 1 Низько-деталізоване зображення: А) Оригінал зображення; Б) Зашумлене зображення з середньою кількістю шуму; В) Зображення з застосованим медіанним фільтром розміром 3; Г) Зображення з застосованим медіанним фільтром розміром 5; Д) Зображення з застосованим медіанним фільтром розміром 7; Е) Зображення з застосованим медіанним фільтром розміром 9.

Порівнюючи вищенаведені результати можна зробити кілька висновків. Застосування медіанного фільтру дозволяє ефективно боротися з шумом, особливо з так званим Salt&Pepper noise. Як бачимо, оригінальне чорно-біле зображення з доволі великим нанесеним шумом запросто було відфільтроване медіанним фільтром. Окрім видалення шуму, можемо спостерігати незначне спотворення деталей на зображенні – розмиття. Застосування медіанного фільтру із розміром ядра дозволяє ефективніше боротися з великою кількістю шуму, проте плата за це – якість зобрження, воно все більш буде ставати розмитим.

Зображення з низькою кількістю деталей показує такі ж результати. Боротьба з шумом відбувається ефективно. Оскільки зображення містить мало деталей, вплив медіанного фільтру, а саме розмитість, не є такою помітною як на високо-деталізованому зобреженні. Як наслідок – ми можемо застосовувати медіанний фільтр з більшим кернелом, щоб ефективно боротися зі значною кількістю шуму з мінімально замітними змінами для людського ока оригінального зображення.

**Висновок:** виконавши дану лабораторну роботу, я ознайомився з застосування фільтрів різних типів до зображення, освоїв методи мінімазації шуму, дослідивши та імплементувавши медіанний фільтр, методи морфології та виділення границь. Медіанний фільтр чудово справляється з задачою видалення шуму типу Salt&Pepper. Основна задача – підібрати ядро мінімального розміру, яке ефективно видалить шум та внесе мінімальні зміни в оригінальне зображення. Якщо ядро взяти завеликим – зображення буде доволі розмитим. Якщо замале – певні шуми залишаться видимими. Застосування цього фільтру є більш ефективним на зображеннях з малою деталізацією, оскільки розмитість на такому зображенні є менш помітною. Застосування на слабо контрастних зображення привело до результуючого зображення, яке є слабо контрастним та важко сприймається людиною.

Додаток 1. Код Програми

import cv2

import numpy as np

import random as rand

def get\_number\_of\_noised\_pixels(rows,cols, intensity):

    number\_of\_pixels = rows\*cols

    if intensity is "low":

      number\_of\_pixels //= rand.randint(50, 100)

    elif intensity is "medium":

      number\_of\_pixels //= rand.randint(15,50)

    elif intensity is "high":

      number\_of\_pixels //= rand.randint(5,15)

    return number\_of\_pixels

def add\_noise(img, intensity="medium"):

    rows , cols = img.shape

    number\_of\_pixels = get\_number\_of\_noised\_pixels(rows,cols,intensity)

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

        y\_coord=rand.randint(0, rows - 1)

        x\_coord=rand.randint(0, cols - 1)

        img[y\_coord][x\_coord] = 255

    number\_of\_pixels = get\_number\_of\_noised\_pixels(rows,cols,intensity)

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

        y\_coord=rand.randint(0, rows - 1)

        x\_coord=rand.randint(0, cols - 1)

        img[y\_coord][x\_coord] = 0

    return img

def median\_filter(img, kernel\_size):

    rows, cols = img.shape

    kernel\_elements = np.zeros(kernel\_size\*\*2)

    indexer = kernel\_size // 2

    img\_filtered = np.zeros((rows,cols))

    for row in range(rows):

        for col in range(cols):

            for i in range(kernel\_size):

                curr\_process\_row = row - indexer + i

                if curr\_process\_row < 0 or curr\_process\_row > rows - 1:

                    for j in range(kernel\_size):

                        kernel\_elements[i \* kernel\_size + j] = 0

                else:

                    for j in range(kernel\_size):

                      curr\_process\_col = col - indexer + j

                      if curr\_process\_col < 0 or curr\_process\_col > cols-1:

                          kernel\_elements[i \* kernel\_size + j] = 0

                      else:

                          kernel\_elements[i \* kernel\_size + j] = img[curr\_process\_row][curr\_process\_col]

            img\_filtered[row][col] = np.sort(kernel\_elements)[(kernel\_size\*\*2) // 2]

    return img\_filtered

img\_names = ['High-detailized-image', 'Low-detailized-image', 'Cat-high-contrast', 'Cat-low-contrast']

kernel\_sizes = [3,5,7,9]

for img\_name in img\_names:

    img = cv2.imread(f'{img\_name}.jpg',cv2.IMREAD\_GRAYSCALE)

    if img is None:

        print("Error happed while reading the image")

        continue

    noised\_img = add\_noise(img, intensity="medium")

    cv2.imwrite(f'{img\_name}-noised.jpg', noised\_img)

    for kernel\_size in kernel\_sizes:

        cv2.imwrite(f'{img\_name}-median-filtered-with-kernel-{kernel\_size}.jpg', median\_filter(noised\_img, kernel\_size))