МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра систем штучного інтелекту

Лабораторна робота №1

З курсу “Обробка зображень методами штучного інтелекту”

Виконала:
студентка групи КН-408
Жук Анастасія

Викладач:
Пелешко Д. Д.

Львів – 2022

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.

Теоретичні відомості

У світі комп'ютерного зору фільтрація зображень використовується для модифікації зображень на етапі попереднього 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 необхідно використовувати фільтри зображень, щоб змінити зображення згідно власних потреб. Аналогічно, зображення може бути розмитим або зашумленим, яке потребувати уточнення і фокусування.

Варіант 13

1. 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).

а) Виконати детекцію границь на зображеннях за допомогою операторів Sobel, Prewitt. Провести порівняльний аналіз.

Хід роботи

1**.** Обрали дві фотографії із різною контрастністю

Рис. 1 Знимок із високою контрасністю

Рис. 2 Знимок із низькою контрасністю

2. Обрали дві фотографії із різною деталізованістю

Рис. 3 Низько деталізована картинка

Рис. 4 Високо деталізована картинка

Код:

```
# -*- coding: utf-8 -*-
```

"""Untitled1.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1lB3QmRGs6OhreZHDugIF-Ywi6U-W1myy
"""

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from math import sqrt
import time
import math

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline

!wget -c https://image.shutterstock.com/image-photo/blurry-zebra-crossing-silhouettes-shadows-600w-734948344.jpg -O high_contrast.jpg

!wget -c https://miro.medium.com/max/1400/1*IqNKPg7wUktQDDTbk80H3Q.jpeg -O another_contrast.jpeg

!wget -c https://www.cyfrovychok.ua/UserFiles/Photo/43/landscape_2.jpg -O detailized.jpg

!wget -c https://cdn.xxl.thumbs.canstockphoto.com/password-in-my-mind-drawing_csp17412096.jpg -O low_detailized.jpg

from PIL import Image
```

```
from IPython.display import display
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow

high_contrast = 'high_contrast.jpg'
low_detailized = 'low_detailized.jpg'
detailized = 'detailized.jpg'
another_contrast = 'another_contrast.jpeg'

img1_high_contrast = cv2.imread(high_contrast)
img2_another_contrast = cv2.imread(another_contrast)
img3_low_detailized = cv2.imread(low_detailized)
img4_detailized = cv2.imread(detailized)

cv2_imshow(img4_detailized)

img1_high_contrast = cv2.imread(high_contrast, cv2.IMREAD_GRAYSCALE)
img2_another_contrast = cv2.imread(another_contrast, cv2.IMREAD_GRAYSCALE)
img3_low_detailized = cv2.imread(low_detailized, cv2.IMREAD_GRAYSCALE)
img4_detailized = cv2.imread(detailized, cv2.IMREAD_GRAYSCALE)



kernel_Sobel_operator_horizontal = np.array(
    [[-1, 0, 1],
     [-2, 0, 2],
     [-1, 0, 1]]
)

kernel_Sobel_operator_vertical = np.array(
    [[1, 2, 1],
```

```
    [0, 0, 0],
    [-1, -2, -1]]
)

kernel_Prewitt_operator_horizontal = np.array(
   [
   [-1, 0, 1],
   [-1, 0, 1],
   [-1, 0, 1]
])
kernel_Prewitt_operator_vertical = np.array([
   [1, 1, 1],
   [0, 0, 0],
   [-1, -1, -1]
])

def apply_filter(image, x_filter, y_filter):
  result = np.zeros(shape=(image.shape[0], image.shape[1]), dtype=np.uint8)
  h, v = image.shape
  for row in range(1, h - 2) :
    for column in range(1, v - 2) :
      result[row, column] = np.sqrt(
          np.sum(x_filter * image[row-1: row+2, column-1: column+2])**2 +
          np.sum(y_filter * image[row-1: row+2, column-1: column+2])**2
      )
  return result

def plot_images_orig_and_detected(images):
  fig, axs = plt. subplots(1,3, figsize=(25, 5))
  fig.suptitle('Original - Detected Sobel - Detected Prewitt')
  axs[0].imshow(images[0], cmap='gray')
```

```
  axs[0].set_title('Orig')
  axs[1].imshow(images[1], cmap='gray')
  axs[1].set_title('Sobel')
  axs[2].imshow(images[2], cmap='gray')
  axs[2].set_title('Prewitt')
  plt.show()


detected_sobel_1 = apply_filter(img2_another_contrast,
kernel_Sobel_operator_horizontal, kernel_Sobel_operator_vertical)
detected_prewitt_1 = apply_filter(img2_another_contrast,
kernel_Prewitt_operator_horizontal, kernel_Prewitt_operator_vertical)
plot_images_orig_and_detected([img2_another_contrast, detected_sobel_1,
detected_prewitt_1])


detected_sobel_2 = apply_filter(img1_high_contrast, kernel_Sobel_operator_horizontal,
kernel_Sobel_operator_vertical)
detected_prewitt_2 = apply_filter(img1_high_contrast,
kernel_Prewitt_operator_horizontal, kernel_Prewitt_operator_vertical)
plot_images_orig_and_detected([img1_high_contrast, detected_sobel_2,
detected_prewitt_2])


detected_sobel_3 = apply_filter(img3_low_detailized,
kernel_Sobel_operator_horizontal, kernel_Sobel_operator_vertical)
detected_prewitt_3 = apply_filter(img3_low_detailized,
kernel_Prewitt_operator_horizontal, kernel_Prewitt_operator_vertical)
plot_images_orig_and_detected([img3_low_detailized, detected_sobel_3,
detected_prewitt_3])


detected_sobel_4 = apply_filter(img4_detailized, kernel_Sobel_operator_horizontal,
kernel_Sobel_operator_vertical)
detected_prewitt_4 = apply_filter(img4_detailized, kernel_Prewitt_operator_horizontal,
kernel_Prewitt_operator_vertical)
plot_images_orig_and_detected([img4_detailized, detected_sobel_4,
detected_prewitt_4])
```

3. Оператори Sobol, Prewitt для поданих фотографій

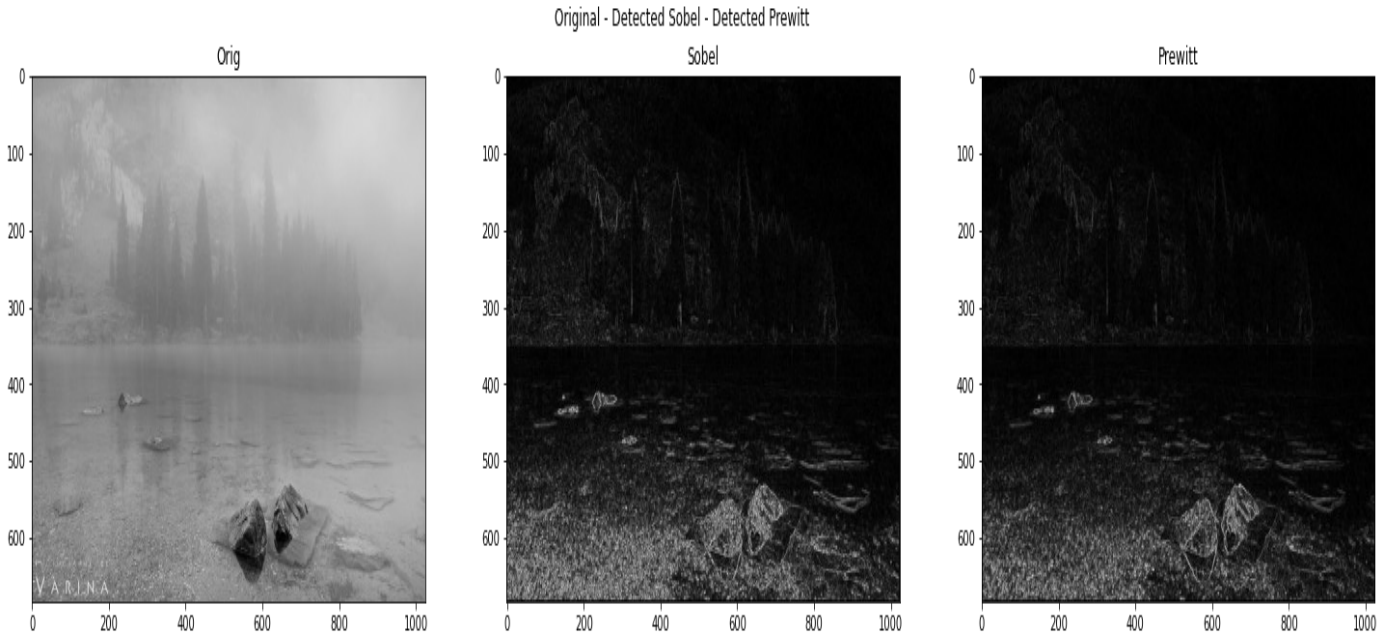

рис. 5 Фільтри Sobel і Prewitt для фотографії з низьким контрастом

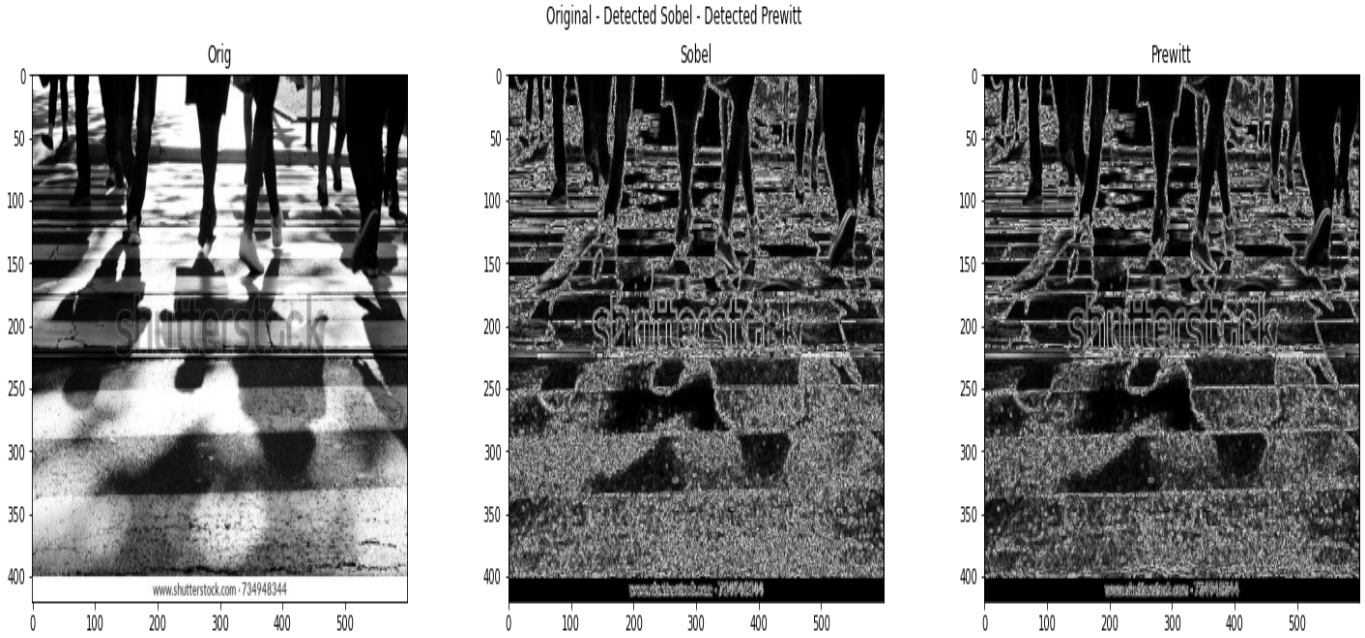

рис. 6 Фільтри Sobel і Prewitt для фотографії з високим контрастом

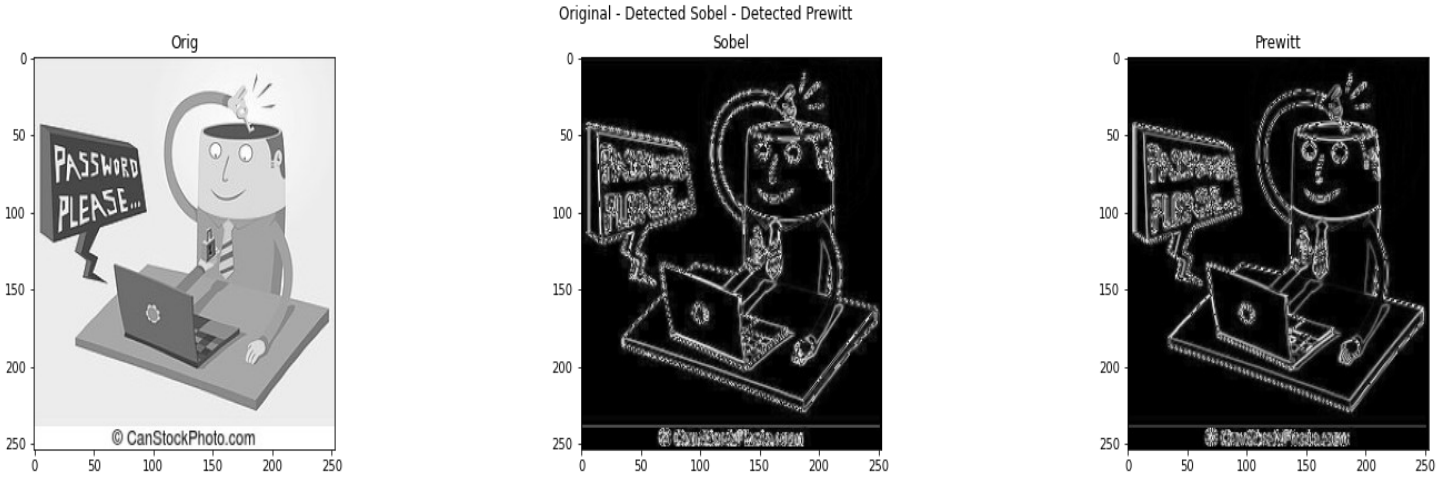

рис. 7 Фільтри Sobel і Prewitt для фотографії з низькою деталізацією

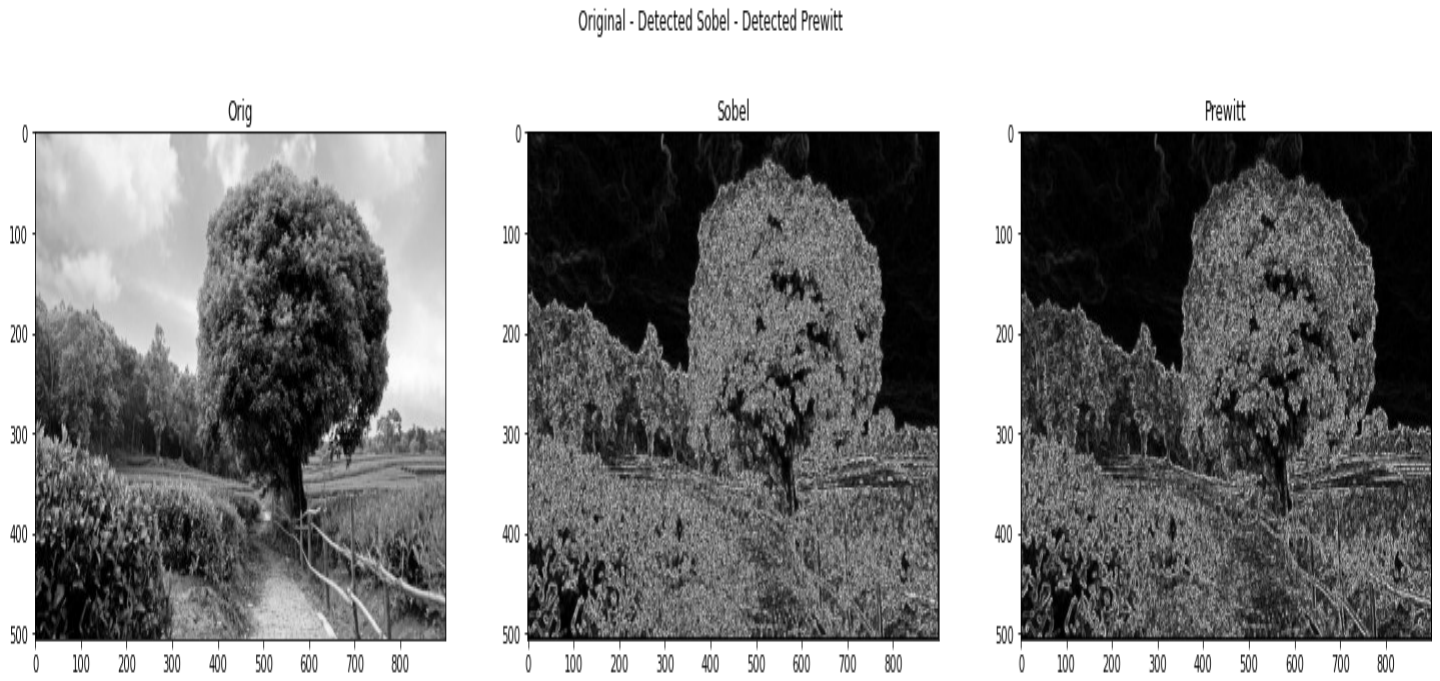

рис. 8 Фільтри Sobel і Prewitt для фотографії з високою деталізацією

Висновок: на даній лабораторій було досліджено два оператори для знаходження країв обєктів на зображеннях. Наглядну різницію важко розібрати неозброєним оком. Про те варто підмітити:

1. оператор Sobel робить кордони світлішими через наявність 2 в середині ядра

2. оператор Sobel також через збільшені ваги посередині ядра детальніше описує границі, що в деяких випадках є перевагою адже допомагає при пошуку незначних деталей на фото (Див. Рис. 8), але при тому є і недоліком адже підкреслює шуми відповідно (Див. Рис. 2)

3. оператор Prewitt у свою чергу, роблячи висновок із поданих вверху фотографій гарно згладжує контур і виглядає більш гладким порівнянно до відповідної фотографії із фільтром Sobel (Див. Рис. 7)