Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота № 3

з дисципліни
«Штучний інтелект в задачах обробки зображень»

Виконав:
студент групи ІМ-11
Царик Микола Миколайович

Перевірив:
Нікітін Валерій Андрійович

Київ 2023

Мета: навчитися виконувати розмітку дорожних ліній на відео засобами OpenCV в режимі реального часу

Завдання:

1. Проробити з будь-якою фотографією процедури, які описані в теоретичних відомостях;
2. Зробити розпізнавання розмітки з будь-якого відеофайлу.

Хід виконання:

Імпортуємо необхідні бібліотеки:

```
import cv2
import numpy as np
```

Створюємо функцію draw_lines, яка буде використовуватися для малювання ліній на зображенні:

```
def draw_lines(frame, lines, color=[0, 165, 255], thickness=7):
    if lines is None:
        return

    # Ініціалізація списків для координат лівих та правих ліній
    left_line_x = []
    left_line_y = []
    right_line_x = []
    right_line_y = []

    # Перебір усіх ліній для визначення їх нахилу та класифікації
    for line in lines:
        for x1, y1, x2, y2 in line:
            if x2 - x1 == 0:  # Перевірка на вертикальність лінії
                continue
            slope = (y2 - y1) / (x2 - x1)  # Обчислення нахилу

            # Відкидання ліній, які мають надто малий нахил
            if abs(slope) < 0.5:
                continue

            # Класифікація ліній на ліві та праві
            if slope <= 0:
                left_line_x.extend([x1, x2])
                left_line_y.extend([y1, y2])
            else:
                right_line_x.extend([x1, x2])
                right_line_y.extend([y1, y2])
```

```
    # Визначення мінімальної та максимальної координати y для малювання ліній
    min_y = frame.shape[0] * (2.8 / 5)
    max_y = frame.shape[0]

    # Малювання лівої лінії, якщо вона була знайдена
    if len(left_line_x) > 0 and len(left_line_y) > 0:
        poly_left = np.poly1d(np.polyfit(left_line_y, left_line_x, deg=1))
        left_x_start = int(poly_left(max_y))
        left_x_end = int(poly_left(min_y))
        cv2.line(frame, (left_x_start, int(max_y)), (left_x_end, int(min_y)),
color, thickness)

    # Малювання правої лінії, якщо вона була знайдена
    if len(right_line_x) > 0 and len(right_line_y) > 0:
        poly_right = np.poly1d(np.polyfit(right_line_y, right_line_x, deg=1))
        right_x_start = int(poly_right(max_y))
        right_x_end = int(poly_right(min_y))
        cv2.line(frame, (right_x_start, int(max_y)), (right_x_end, int(min_y)),
color, thickness)
```

Створюємо функцію process_image, в якій ми обробляємо кожен кадр відео, виконуючи наступні кроки:

1. Змінюємо розмір кадру.
2. Перетворюємо зображення у градації сірого.
3. Застосовуємо Гауссівське розмиття для зниження шуму.
4. Виконуємо виявлення країв за допомогою алгоритму Кенні.
5. Створюємо маску для виділення регіону інтересу (ROI).
6. Застосовуємо маску до результатів детектора Кенні.
7. Використовуємо перетворення Хафа для знаходження ліній.
8. Викликаємо draw_lines для малювання ліній на кадрі.

```
def process_image(frame):
    # Зміна розміру кадру
    frame = cv2.resize(frame, (400, 508))

    # Перетворення кадру в градації сірого
    grayscale = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Застосування Гауссівського розмиття
    kernel_size = 5
    blur = cv2.GaussianBlur(grayscale, (kernel_size, kernel_size), 0)

    # Визначення країв за допомогою алгоритму Кенні
    low_t = 50
    high_t = 150
    edges = cv2.Canny(blur, low_t, high_t)
```

```
    # Створення маски для виділення регіону інтересу
    vertices = np.array(
        [
            [
                (0, frame.shape[0]),
                (0, 350),
                (100, 300),
                (frame.shape[1], 160),
                (frame.shape[1], frame.shape[0])
            ]
        ], dtype=np.int32
    )

    # Застосування маски до кадру
    mask = np.zeros_like(edges)
    ignore_mask_color = 255
    cv2.fillPoly(mask, vertices, ignore_mask_color)
    masked_edges = cv2.bitwise_and(edges, mask)

    # Використання перетворення Хафа для пошуку ліній
    rho = 3
    theta = np.pi / 180
    threshold = 100
    min_line_len = 75
    max_line_gap = 60
    lines = cv2.HoughLinesP(masked_edges, rho, theta, threshold, np.array([]),
minLineLength=min_line_len, maxLineGap=max_line_gap)

    # Малювання ліній на кадрі
    draw_lines(frame, lines)

    return frame
```

Виконуємо завантаження тестового зображення та його обробку за допомогою process_image:

```
# Отримання початкового зображення та його обробка
img = cv2.imread("Lab3/data/road.jpg")
img = cv2.resize(img, (400, 508))
processed_img = process_image(img)
```

Додаємо текст до оригінального та обробленого зображень для їх порівняння:

```
# Додавання тексту до зображень
cv2.putText(img, "Original", (125,100), cv2.FONT_ITALIC, 1, (16,74,9), 4,
cv2.LINE_AA)
cv2.putText(processed_img, "Processed", (125,100), cv2.FONT_ITALIC, 1, (16,74,9),
4, cv2.LINE_AA)
```

Виводимо на екран оригінальне та оброблене зображення:

```
# Показ зображень
cv2.imshow('Image. Original', img)
cv2.waitKey()
cv2.destroyAllWindows()
cv2.imshow('Image. Processed', processed_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Відкриваємо відеофайл за допомогою cv2.VideoCapture та обробляємо кожен кадр у циклі, викликаючи для цього process_image:

```
# Цикл для обробки відео
cap = cv2.VideoCapture("Lab3/data/Video.mp4")
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        processed_frame = process_image(frame)
        cv2.imshow('Video', processed_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
```

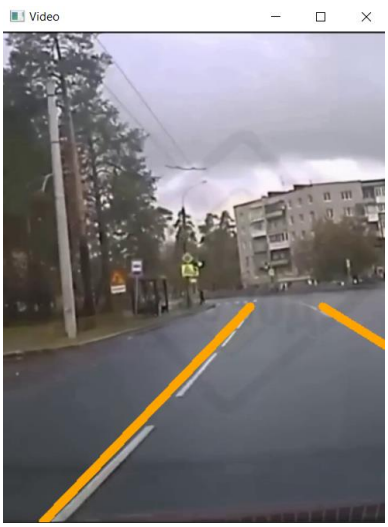
Після завершення роботи з відео закриваємо всі вікна:

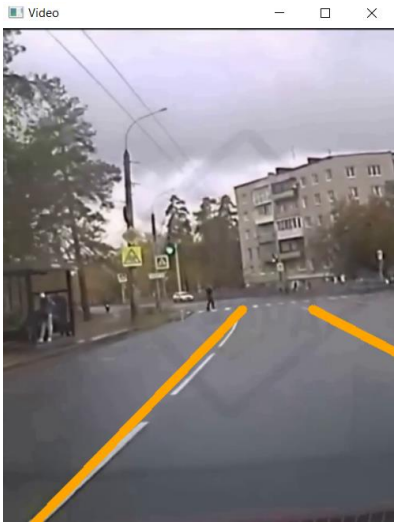
```
# Закриття вікон і звільнення ресурсів
cap.release()
cv2.destroyAllWindows()
```

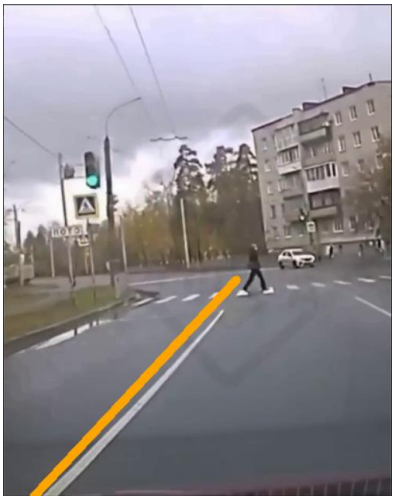
Результат

Фото:

Відео:







Висновки: В ході цієї лабораторної роботи, окрім вдосконалення навичок роботи з cv2, також було зроблено кілька висновків

1. Розглянута робота з масками та застосування алгоритма Хафа. Він використовується для виявлення ліній на зображенні, і нами були налаштовувані його параметри для покращення результатів.
2. Було помічено, що якість результатів на відео та фото однакова, через те, що відео являє собою послідовний набір з фактично тих самих фото (кадрів).
3. Код можна було б покращити, якщо у функцію обробки зображення ми додали б параметр для передачі масиву вершин маски, так маски будуть індивідуальні для кожного фото та відео і якість результата підвиться. Ми не стали цього робити бо лабораторна робота не планує розширення у майбутньому, та обидва медіа-ресурси , які ми використали, мають однакові вершини маски.

Загалом, робота надала практичні навички в галузі комп'ютерного зору та глибше розуміння того, як можуть бути застосовані алгоритми обробки зображень для розв'язання реальних задач, таких як розпізнавання та візуалізація дорожньої розмітки,

Контрольні запитання:

1. Чому краще обробляти сіре зображення?

Сірі зображення мають лише один канал інтенсивності порівняно з трьома в кольорових зображеннях (червоний, зелений, блакитний), що зменшує обсяг обчислень. Коли зображення перетворене в градації сірого, відпадає потреба аналізувати кольорову інформацію, що дозволяє алгоритмам зосередитися на інтенсивності та контрастності. Для багатьох задач обробки зображень, включаючи виявлення країв, кольорова інформація не несе корисної інформації і може навіть ускладнити виявлення важливих особливостей.

2. Навіщо робити розмиття зображення перед розпізнаванням?

Розмиття допомагає зменшити шум та непотрібні деталі на зображенні, що може поліпшити точність виявлення країв. Гауссівське розмиття та інші методи згладжування допомагають об'єднати сусідні пікселі для створення більш однорідного зображення.

3. Що таке алгоритм Кенні?

Алгоритм Кенні - це багатоетапний алгоритм виявлення країв, який використовує градієнти зображення для виявлення областей з високою інтенсивністю змін. Перш ніж виконувати пошук країв, алгоритм зазвичай застосовує Гауссівське розмиття.

4. Що таке перетворення Хафа?

Перетворення Хафа використовується для виявлення параметричних форм, таких як прямі лінії, кола тощо на зображенні. У просторі параметрів ведеться підрахунок "голосів" для потенційних ліній, і ті, що отримують найбільшу кількість голосів, вважаються реальними лініями на зображенні. Існують різні версії перетворення Хафа, адаптовані для різних типів форм.

5. Яка мета маски?

Маска використовується для фокусування на певній частині зображення, наприклад, на дорозі для розпізнавання ліній. Вона дозволяє ігнорувати ті частини зображення, які не несуть інформацію для поточної задачі, зменшуючи можливість хибних виявлень. Обробляючи лише область інтересу, ми зменшуємо кількість обчислень та збільшуємо швидкість виконання алгоритмів.