Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи № 4
з навчальної дисципліни «Computer Vision»

Тема:

ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ПОКРАЩЕННЯ ЯКОСТІ ЦИФРОВИХ ЗОБРАЖЕНЬ ДЛЯ ЗАДАЧ COMPUTER VISION

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-14
Бабіч Д. В.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О. О.

Київ 2024

I. Мета:

Дослідити принципи та особливості практичного застосування технологій покращення якості цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек.

II. Завдання:

Завдання II рівня.

Здійснити виконання завдання лабораторної роботи для відеопотоку за варіантами таблиці додатку.

Таблиця 1.1 – Варіант завдання

Варіант (місяць народження)	**Контент цифрового зображення / джерело**	**Об'єкт ідентифікації**
7	Автомагістраль	Легкові автомобілі

III. Результати виконання лабораторної роботи.

3.1. Синтезована математична модель обробки графічних зображень.

На стадії попередньої обробки зображення перетворюється у відтінки сірого, що означає, що кожен піксель представлений лише одним значенням яскравості замість трьох значень кольору (червоного, зеленого та синього), що робить обробку більш ефективною та зменшує обсяг даних для аналізу.

$$color = \frac{r + g + b}{3}$$

Де $color$ – колір, який буде отримано, r – значення кольору на червоному каналі, g – значення кольору на зеленому каналі, b – значення кольору на синьому каналі.

Використовується метод вирівнювання гістограми, що поліпшує контраст та розподіл яскравості у зображенні.

$$h(v) = \frac{cdf(v) - cdf_{min}}{(M \cdot N) - cdf_{min}} \cdot (L - 1)$$

Де $h(v)$ – нове значення інтенсивності пікселя, $cdf(v)$ – кумулятивна функція розподілу для пікселя з заданою початковою інтенсивністю, cdf_{min} – мінімально допустиме значення інтенсивності, $M \cdot N$ – загальна кількість пікселів у растровому зображені, L – кількість рівнів інтенсивності (256 для 8-бітного зображення).

Застосовується розмивання Гаусса для зменшення впливу випадкових змін в яскравості та кольорі. Використовуються морфологічні операції, такі як закриття, ерозія та дилатація, для видалення непотрібних дрібних деталей у зображенні та з'єднання близьких областей. Застосовується алгоритм Canny для виявлення контурів об'єктів у зображенні на основі попередніх етапі векторизації. Створюється маска, що визначає області, які потрібно розглядати у подальшому аналізі.

Знаходяться контури об'єктів на зображенні. Для кожного контуру визначається прямокутник, що охоплює його. Якщо розміри та співвідношення сторін прямокутника відповідають заданим параметрам, прямокутник додається до списку. Застосовується фільтрація, щоб видалити прямокутники, які повністю містяться в інших прямокутниках. Для кожного прямокутника на зображенні накладається прямокутник з заданим кольором та товщиною. Якщо задано назву області інтересу (ROI), вона виводиться на зображенні вище відповідного прямокутника.

3.2. Результати архітектурного проектування та їх опис.

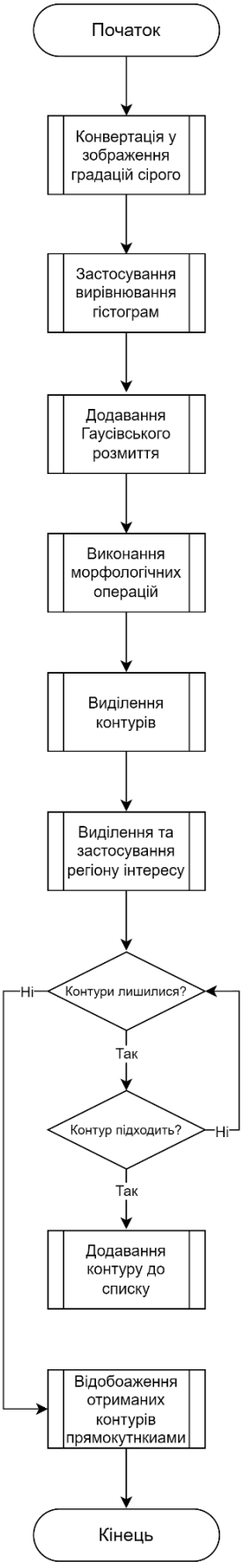

Рисунок 1.1 – Діаграма алгоритму ідентифікації

1. Перетворення зображення у градації сірого з метою покращення обробки та зменшення використання ресурсів;
2. Застосування вирівнювання гістограм з метою поліпшення контрасту та розподілення яскравості;
3. Застосування розмивання Гаусса для зменшення впливу випадкових змін та шуму.
4. Використання морфологічних операцій з метою видалення дрібних деталей та з'єднання контурів;
5. Виявлення контурів об'єктів;
6. Визначення прямокутників об'єктів та чи підходять вони під задані величини;
7. Відображення контурів на зображенні.

3.3. Опис структури проекту програми.

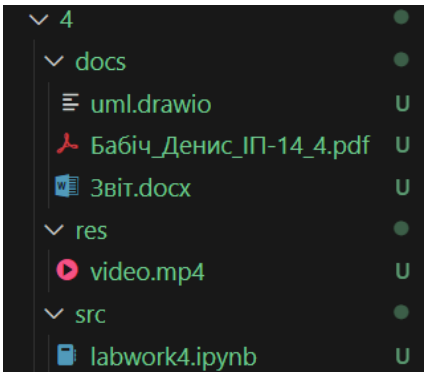

Рисунок 1.2 – Структура проєкту

У директорії src зберігається labwork4.py, який є юпітер-записником з вихідним кодом, директорія docs – зберігає файли звіту у форматі pdf, docx та drawio (файл з діаграмами), директорія res – файл з відеорядом

3.4. Результати роботи програми відповідно до завдання.

Рисунок 1.3 – Приклад роботи алгоритму

3.5. Програмний код, що забезпечує отримання результату.

```
# %% [markdown]
# # Лабораторна робота № 4

# %% [markdown]
# ## ІП-14 Бабіч Денис (09.07.2003)

# %% [markdown]
# ---

# %% [markdown]
# # Підготовчий етап

# %% [markdown]
# ## Імпортування необхідних модулів

# %%
import cv2
import numpy as np

# %% [markdown]
# ## Створення необхідних функцій

# %%
def preprocess_frame(frame: np.ndarray, blur_strength: int, threshold: float, *roi_polygons: list)
-> np.ndarray:
    preprocessed_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    preprocessed_frame = cv2.equalizeHist(preprocessed_frame)
      preprocessed_frame = cv2.GaussianBlur(preprocessed_frame, (blur_strength, blur_strength),
0)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (1, 2))
    preprocessed_frame = cv2.morphologyEx(preprocessed_frame, cv2.MORPH_CLOSE, kernel)
    preprocessed_frame = cv2.erode(preprocessed_frame, kernel, iterations = 1)
    preprocessed_frame = cv2.dilate(preprocessed_frame, kernel, iterations = 1)
    preprocessed_frame = cv2.Canny(preprocessed_frame, threshold / 2, threshold)
    mask = np.zeros_like(preprocessed_frame)

    for roi_polygon_vertices in roi_polygons:
        cv2.fillPoly(mask, [np.array(roi_polygon_vertices, dtype = np.int32)], 255)

    preprocessed_frame = cv2.bitwise_and(preprocessed_frame, mask)

    return preprocessed_frame

def determine_obejcts_rectangles(frame: np.ndarray, width_min: int, width_max: int,
height_min: int, height_max: int, aspect_ratio_min: float, aspect_ratio_max: float) -> list:
    objects_rectangles = []

                contours, _ = cv2.findContours(frame, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
```

```
        x, y, width, height = cv2.boundingRect(contour)
        aspect_ratio = float(width) / height

            if (width_min < width < width_max) and (height_min < height < height_max) and (aspect_ratio_min < aspect_ratio < aspect_ratio_max):
            objects_rectangles.append((x, y, width, height))

     objects_rectangles = [r1 for r1 in objects_rectangles if not any((r1[0] > r2[0] and r1[1] > r2[1] and r1[0] + r1[2] < r2[0] + r2[2] and r1[1] + r1[3] < r2[1] + r2[3] and r1[2] * r1[3] < r2[2] * r2[3]) for r2 in objects_rectangles if r1 != r2)]

    return objects_rectangles

def overlay_objects_rectangles(frame: np.ndarray, rectangles: list, roi_title: str = "", color: tuple = (0, 255, 0), thickness: int = 2) -> np.ndarray:
    ROI_TITLE_OFFSET = 10

    FONT_SCALE = 0.5
    FONT_LINE_TYPE = 2

    for (x, y, width, height) in rectangles:
        cv2.rectangle(frame, (x, y), (x + width, y + height), color, thickness)

        if roi_title:
                                    cv2.putText(frame, roi_title, (x, y - ROI_TITLE_OFFSET), cv2.FONT_HERSHEY_SIMPLEX, FONT_SCALE, color, FONT_LINE_TYPE)

    return frame

# %% [markdown]
# ---

# %% [markdown]
# # Основний етап

# %% [markdown]
# ## Застосування на прикладі відео

# %%
KEY_ESCAPE = 27

CAPTURE = cv2.VideoCapture("../res/video.mp4")
FPS = int(CAPTURE.get(cv2.CAP_PROP_FPS))

THRESHOLD = 250
BLUR_STRENGTH = 5

WIDTH_MIN = 55
WIDTH_MAX = 110
HEIGHT_MIN = 30
HEIGHT_MAX = 55
ASPECT_RATIO_MIN = 1.25
```

```
ASPECT_RATIO_MAX = 3.75
FRAME_HEIGHT, FRAME_WIDTH = CAPTURE.read()[1].shape[:2]
FRAME_HALF_HEIGHT = FRAME_HEIGHT / 2
OFFSET_MARKUP = 45
OFFSET_TOP_BOUNDS = 110
OFFSET_BOTTOM_BOUNDS = 75

ROI_TOP_ROAD = [(0, OFFSET_TOP_BOUNDS),
            (0, FRAME_HALF_HEIGHT - OFFSET_MARKUP),
            (FRAME_WIDTH, FRAME_HALF_HEIGHT - OFFSET_MARKUP),
            (FRAME_WIDTH, OFFSET_TOP_BOUNDS)]

ROI_BOTTOM_ROAD = [(0, FRAME_HEIGHT - OFFSET_BOTTOM_BOUNDS),
               (0, FRAME_HALF_HEIGHT + OFFSET_MARKUP),
               (FRAME_WIDTH, FRAME_HALF_HEIGHT + OFFSET_MARKUP),
               (FRAME_WIDTH, FRAME_HEIGHT - OFFSET_BOTTOM_BOUNDS)]

ROI_TITLE = "Car"

while CAPTURE.isOpened():
   ret, frame = CAPTURE.read()

   if not ret:
      break

        preprocessed_frame = preprocess_frame(frame, BLUR_STRENGTH, THRESHOLD,
ROI_TOP_ROAD, ROI_BOTTOM_ROAD)
            objects_roi = determine_obejcts_rectangles(preprocessed_frame, WIDTH_MIN,
WIDTH_MAX, HEIGHT_MIN, HEIGHT_MAX, ASPECT_RATIO_MIN,
ASPECT_RATIO_MAX)
   processed_frame = overlay_objects_rectangles(frame, objects_roi, ROI_TITLE)

   cv2.imshow("", processed_frame)

   if cv2.waitKey(FPS) & 0xFF == KEY_ESCAPE:
      break

CAPTURE.release()
cv2.destroyAllWindows()
```

IV. Висновки.

У цій роботі було проведено дослідження методів обробки зображень. Починаючи з перетворення зображення у відтінки сірого, для спрощення подальшої обробки та зменшення обсягу даних. Далі було використано метод вирівнювання гістограми для поліпшення контрасту та розподілу яскравості. Для виявлення контурів об'єктів та створення маски використовувалися різні фільтри та алгоритми, такі як розмивання Гаусса, морфологічні операції та алгоритм Canny. Завершальним етапом було визначення контурів об'єктів та створення охоплюючих прямокутників для подальшого аналізу та обробки. Таким чином, за допомогою всіх кроків вдалося створити приклад системи розпізнавання легкових автомобілів у відеопотоці.

Виконав: ІП-14 Бабіч Д. В.