ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ
КАФЕДРА ЕОМ

КВАЛІФІКАЦІЙНА РОБОТА
ПЕРШИЙ РІВЕНЬ (БАКАЛАВР)

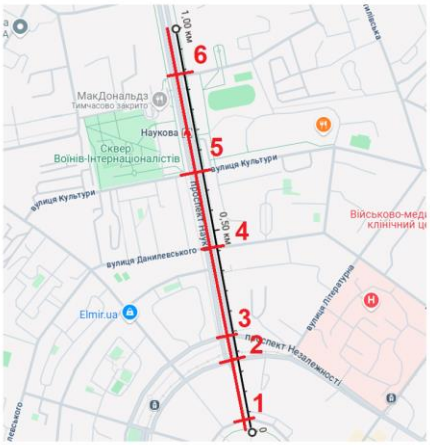
ІНТЕЛЕКТУАЛЬНИЙ ПРОГРАМНО-АПАРАТНИЙ МОДУЛЬ СУПРОВОДУ СЛАБОЗОРИХ НА ЗУПИНКАХ ГРОМАДСЬКОГО ТРАНСПОРТУ

АВТОР
НЕЧІТАЙЛО О. В.
СТ. ГР. КІУКІ-21-6

КЕРІВНИК
ГАВРАШЕНКО А. О.
АС. КАФ. ЕОМ

СЛАБОЗОРІСТЬ ТА ВІЙНА

2021 рік – 17 000
2023 рік – 19 000



Позначення кількості перехресть на 1 км на проспекті Науки

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В ДАНІЙ ОБЛАСТІ

WeWALK Smart Cane 2

Sonic Pathfinder

BlindSquare

OrCam MyEye 3 Pro

6

МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ

Мета кваліфікаційної роботи 7

Метою кваліфікаційної роботи є розробка програмно-апаратного модулю для допомоги особам з вадами зору у самостійному орієнтуванні на зупинках громадського транспорту, який забезпечує автоматичне розпізнавання типу транспорту та маршрутного номеру, а також генерацію голосових підказок для користувача.

8

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

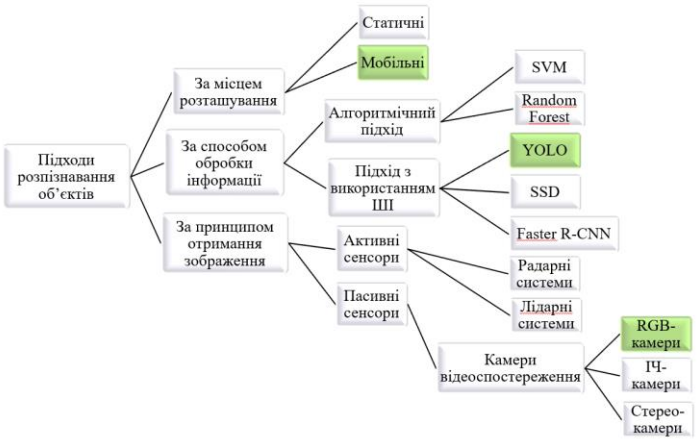
Задачі кваліфікаційної роботи 9

- збір та розмітка даних для глибинного навчання моделі YOLO розпізнаванню різних типів транспортних засобів (автобус, тролейбус, трамвай) та їх маршрутні номери;
- аугментація датасету шляхом моделювання умов низької видимості, створюючи штучні погодні умов (туман, дощ, ніч тощо) в різний час доби (денне світло, сутінки, темрява);
- розробка програмного модуля ідентифікації транспорту із застосуванням методів комп'ютерного зору;
- розробка програмного модуля голосового супроводу на основі методів синтезування мовлення для відтворюння в режимі реального часу через аудіоінтерфейс розробленого пристрою;
- побудова апаратного модуля для збору, аналізу та постпроцесінгу даних;
- тестування та адаптація системи до реальних умов експлуатації.

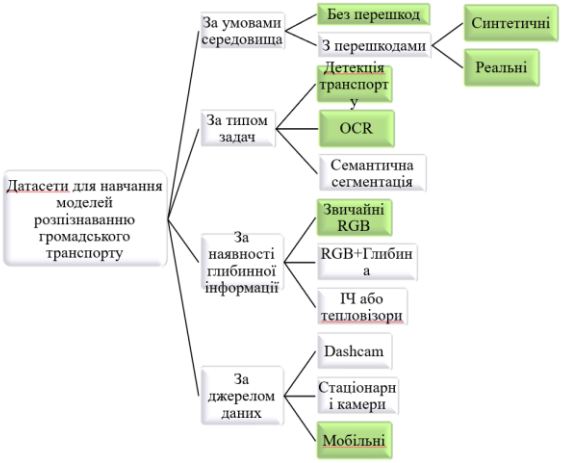
10

ЗБІР ТА РОЗМІТКА ДАНИХ

Збір та розмітка даних 11



Збір та розмітка даних 12



Збір та розмітка даних 13

(239)

Приклад розмітки для автобуса

(428)

Приклад розмітки для тролейбуса

(234)

Приклад розмітки для трамвая

Усього 756 зображень:

Датасети з сайту Roboflow (326)

Instagram - “gor_transport” (430)

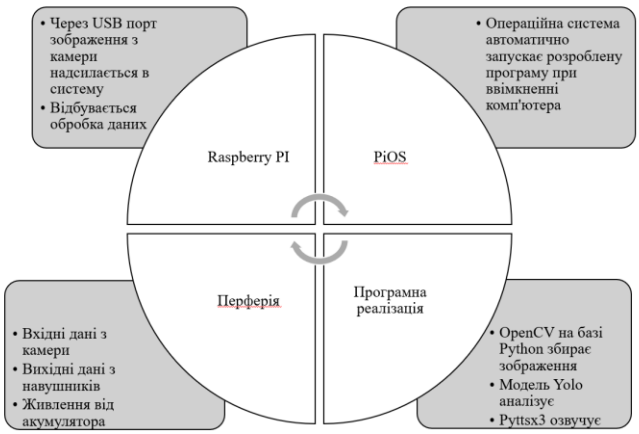
Збір та розмітка даних 14

Приклад розмітки з декількома об’єктами які перекривають один одого

16

АУГМЕНТАЦІЯ ДАТАСЕТУ

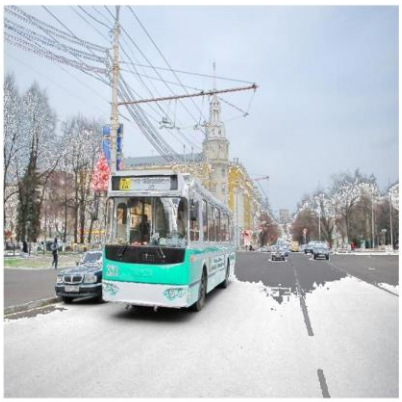
Розробка програмного модуля ідентифікації транспорту 15



Програмно-апаратний стек

Аугментація датасету 17

Умови для аугментації датасету: дощ



Умови для аугментації датасету: сніг

Аугментація датасету 18

Усього 756 + 300 = 1056 зображень

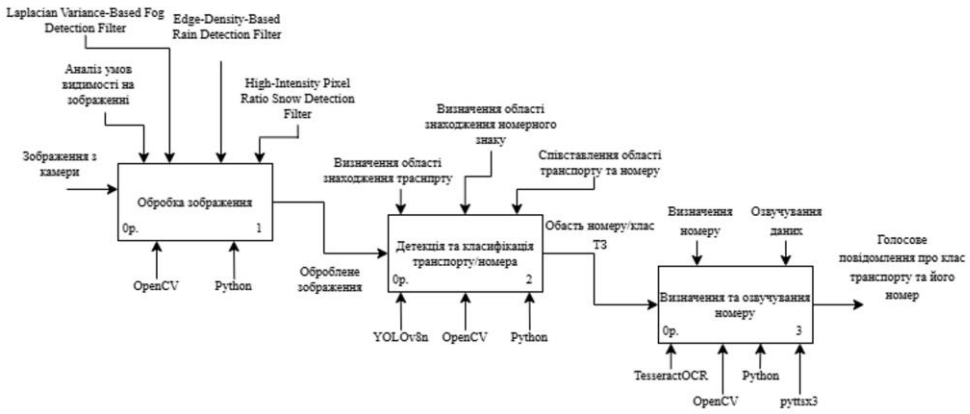
Умови для аугментації датасету: сутінки

Умови для аугментації датасету: замилення

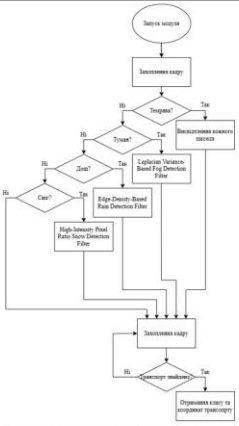
19

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ІДЕНТИФІКАЦІЇ ТРАНСПОРТУ

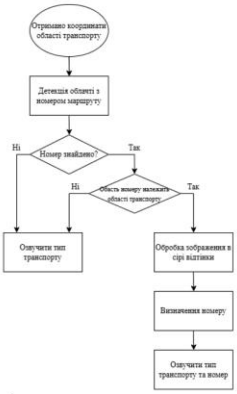
Розробка програмного модуля ідентифікації транспорту 20



Функціональна діаграма запропонованої системи

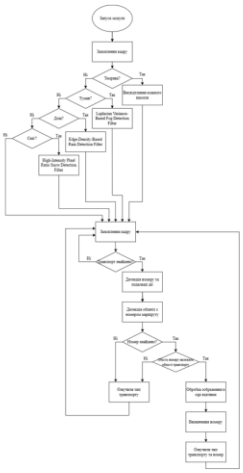


Блок-схема роботи модуля виявлення транспорту



Блок-схема роботи модуля виявлення номеру маршруту

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ГОЛОСОВОГО СУПРОВОДУ



Загальна блок-схема роботи рішення

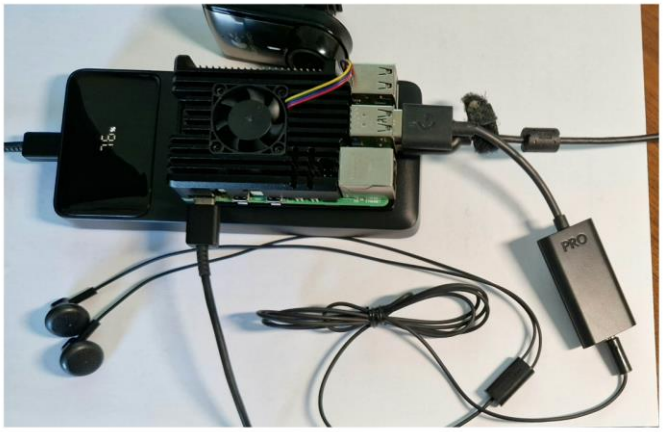
ПОБУДОВА АПАРАТНОГО МОДУЛЯ

Raspberry Pi 5 з встановленою системою охолодження

USB-камера Logitech Web Camera

Звукова карта Logitech G Pro X з навушником

Джерело живлення



Загальний вигляд модуля

27

ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Умови тестування	Кількість тестів, шт.	Точність детектування ТЗ (точність), %	Точність класифікації типу ТЗ, %	OCR правильний, %	Повне озвучення (успіх), шт.
Реальні, денне світло, (100-10000 lx)	10	100%	90%	70%	7
Реальні, сутінки, (50-100 lx)	10	80%	70%	50%	5
Імітація, темрява, (10-50 lx)	10	90%	70%	70%	7
Імітація, туман	10	100%	90%	70%	7
Імітація, дощ	10	100%	100%	80%	6
Імітація, сніг	10	100%	100%	90%	9
Імітація, замилення	10	100%	80%	60%	6
Усього / Середнє	70	95.7%	85.7%	70%	47

Таблиця результатів тестування в реальних та імітованих умовах

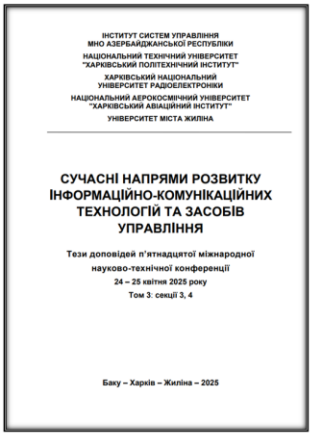
Тестування та аналіз результатів 30

31

ВИСНОВКИ

Висновки 32

- зібрано та розмічено дані для глибинного навчання моделі YOLO розпізнаванню різних типів транспортних засобів (автобус, тролейбус, трамвай) та їх маршрутних номерів;
- аугментовано датасет шляхом моделювання умов низької видимості, створюючи штучні погодні умов (туман, дощ, ніч тощо) в різний час доби (денне світло, сутінки, темрява);
- розроблено програмний модуль ідентифікації транспорту із застосуванням методів комп'ютерного зору;
- розроблено програмний модуль голосового супроводу на основі методів синтезування мовлення для відтворюння в режимі реального часу через аудіоінтерфейс розробленого пристрою;
- побудовано апаратний модуль для збору, аналізу та постпроцесінгу даних;
- протестовно та адаптовано системи до реальних умов експлуатації.



ІНСТИТУТ СИСТЕМ УПРАВЛІННЯ
МНО АЗЕРБАЙДЖАНСЬКОЇ РЕСПУБЛІКИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
"ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ"
УНІВЕРСИТЕТ МІСТА ЖИЛІНА

СУЧАСНІ НАПРЯМИ РОЗВИТКУ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЗАСОБІВ УПРАВЛІННЯ

Тези доповідей п'ятнадцятої міжнародної науково-технічної конференції
24 – 25 квітня 2025 року
Том 3: секції 3, 4

Баку – Харків – Жиліна – 2025

Тези доповіді:
Нечітайло О. В., Гаврашенко А. О. Інтелектуальний модуль навігації для реабілітації осіб з вадами зору на зупинках громадського транспорту. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доп. 15-ї міжнар. наук.-техн. конф. (24–25 квіт. 2025 р., Баку – Харків – Жиліна). – Т. 3. – Харків ; Баку ; Жиліна, 2025. – С. 123.

ДОДАТОК Б

Фрагменти програмного коду системи

Б.1 Словник співставлення ідентифікатору та назви типу визначеного об’єкта (class_names.py)

```
class_names = {
    0: "bus",
    1: "tram",
    2: "trolleybus",
    3: "number"
}
```

Б.2 Модуль детекції транспорту (detection.py)

```
from ultralytics import YOLO

class ObjectDetector:
    def __init__(self, model_path):
        self.model = YOLO(model_path)

    def detect(self, frame):
        results = self.model(frame)
        return results
```

Б.3 Модуль захоплення кадрів (frame_grab.py)

```
import cv2
import time

class FrameGrabber:
    def __init__(self, source, interval_sec=1):
        self.cap = cv2.VideoCapture(source)
        self.interval = interval_sec
        self.last_time = time.time()
        self.fps = self.cap.get(cv2.CAP_PROP_FPS)

    def read(self):
        ret, frame = self.cap.read()
        if not ret:
            return None, None
```

```
        snapshot = None
        current_time = time.time()
        if current_time - self.last_time >= self.interval:
            self.last_time = current_time
            snapshot = frame.copy()

        return frame, snapshot

    def get_fps(self):
        return self.fps

    def release(self):
        self.cap.release()
```

Б.4 Модуль пошуку об’єктів на кадрі (analyzation.py)

```
from detection import ObjectDetector
from class_names import class_names
import pytesseract
import cv2
import os

class Analizer:
    def __init__(self, model):
        self.detector = ObjectDetector(model)

    def analize(self, snapshot):
        if snapshot is not None:
            results = self.detector.detect(snapshot)

            transports = []
            numbers = []

            for result in results:
                for box in result.boxes.data:
                    x1, y1, x2, y2, conf, cls = box.tolist()
                    if conf < 0.65:
                        continue
                    class_id = int(cls)
                    box_data = (int(x1),int(y1),int(x2),int(y2))

                    if class_id in [0, 1, 2]:
                        transports.append((class_id, box_data))
                    elif class_id == 3:
                        numbers.append(box_data)

            if numbers:
                for class_id, (tx1, ty1, tx2, ty2) in
transports:
                    for nx1, ny1, nx2, ny2 in numbers:
                        if nx1 > tx1 and ny1 > ty1 and nx2 < tx2
```

```
and ny2 < ty2:
                                cropped =
snapshot[int(ny1):int(ny2), int(nx1):int(nx2)]
                                gray = cv2.cvtColor(cropped,
cv2.COLOR_BGR2GRAY)
                                text = pytesseract.image_to_string(
                                    gray, config='--psm 7 -c
tessedit_char_whitelist=0123456789'
                                )
                                number = ''.join(filter(str.isdigit,
text))
                                if number:
                                    label =
class_names.get(class_id, "unknown")
                                    return f"{label} {number} is
coming"

            # якщо номерів не було або не співпали — просто
озвучити транспорт
            if transports:
                class_id, _ = transports[0]
                label = class_names.get(class_id, "unknown")
                return f"{label} is coming"

            return None
```

Б.5 Глобальні конфігураційні змінні (config.py)

```
VIDEO_PATH="test_inputs/testvid2.mp4"
MODEL_PATH="models/v1/best.pt"
CHECK_RATE=3
```

Б.6 Модуль озвучування (speech.py)

```
import pyttsx3
import threading
import queue

class Speaker:
    def __init__(self):
        self.engine = pyttsx3.init()
        self.queue = queue.Queue()
        self.thread = threading.Thread(target=self._run,
daemon=True)
        self.thread.start()

    def say(self, text):
        self.queue.put(text)
```

```
    def _run(self):
        while True:
            text = self.queue.get()
            if text is None:
                break
            self.engine.say(text)
            self.engine.runAndWait()
```

Б.7 Модуль визначення погодних умов та накладання фільтрів (weather_detection.py)

```
import numpy as np
import cv2

class WeatherConditionDetector:
    def detect(self, frame):
        hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
        brightness = np.mean(hsv[..., 2])

        if brightness < 50:
            return 'night'
        elif self.is_foggy(frame):
            return 'fog'
        elif self.is_rainy(frame):
            return 'rain'
        elif self.is_snowy(frame):
            return 'snow'
        else:
            return 'clear'

    def is_foggy(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
        laplacian_var = cv2.Laplacian(gray, cv2.CV_64F).var()
        return laplacian_var < 100

    def is_rainy(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
        edges = cv2.Canny(gray, 100, 200)
        return np.sum(edges > 0) > 15000

    def is_snowy(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
        white_pixels = np.sum(gray > 220)
        return white_pixels > 10000

    def apply_filter(self, frame, condition):
        if condition == 'fog':
            return cv2.detailEnhance(frame, sigma_s=10,
sigma_r=0.15)
```

```
        elif condition == 'rain' or condition == 'snow':
            return cv2.fastNlMeansDenoisingColored(frame, None,
10, 10, 7, 21)
        elif condition == 'night':
            return cv2.convertScaleAbs(frame, alpha=1.5,
beta=30)
        return frame
```

Б.8 Головний модуль запуску програмного рішення (main.py)

```
import config
import cv2
import time

from frame_grab import FrameGrabber
from speech import Speaker
from analyzation import Analizer
from weather_detection import WeatherConditionDetector

def main():

    video_path = config.VIDEO_PATH
    model_path = config.MODEL_PATH
    check_rate = config.CHECK_RATE

    frame_grabber = FrameGrabber(0, check_rate)
    fps = frame_grabber.get_fps()
    delay = 1.0 / fps if fps > 0 else 0.033
    speaker = Speaker()
    analyzer = Analizer(model_path)
    weather_detector = WeatherConditionDetector()

    frame, snapshot = frame_grabber.read()
    condition = weather_detector.detect(frame)


    while True:
        start = time.time()

        frame, snapshot = frame_grabber.read()
        if frame is None:
            break

        # показуємо відео завжди для відлагодження
        cv2.imshow("Detection", frame)

        mod_snapshot = weather_detector.apply_filter(snapshot,
condition)
        result = analyzer.analize(mod_snapshot)
        if result:
            speaker.say(result)
```

```
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

            elapsed = time.time() - start
            if delay > elapsed:
                time.sleep(delay - elapsed)

    frame_grabber.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```