Uniwersytet Komisji Edukacji Narodowej w Krakowie
Instytut Bezpieczeństwa i Informatyki

PROJEKT INŻYNIERSKI
DOKUMENTACJA PROJEKTOWA

RADAR ODCINKOWY

wykonany przez:

Tomasz Górski
Nr albumu: 151896
&
Tomasz Joniec
Nr albumu: 151861
&
Patryk Golonka
Nr albumu: 145857

pod opieką:
dr inż. Grzegorz Sokal, mgr Łukasz Przybytek

Kraków 2024
(ostatnia aktualizacja: 23:14:44, 2024-01-14)

Spis treści

1 Szczegółowa dokumentacja projektowa

W zależności od specyfiki projektu! Wymienione niżej podpunkty mają charakter orientacyjny.

1.1 Projekt UML

W szczególności: diagram klas, ew. np. przypadki użycia, diagramy sekwencji, czynności, stanów, obiektów/komponentów/pakietów itp.

1.2 Projekt bazy danych

Baza danych 'radar' składa się z 4 tabel, które służą do zbierania danych wejściowych z kamer oraz do prezentowania wyników za pomocą raportu dla zarejestrowanych użytkowników. Poniżej opis tabel wraz z omówieniem kolumn przechowujących dane.

Tabela cars zawiera 10 pól:

Nazwa	**Opis**
id	-
brand	marka
model	-
year_of_manufacture	rok produkcji
first_registration_date	data pierwszej rejestracji
color	-
plate_number	numer rejestracyjny
owner_name	nazwa właściciela
owner_address	adres właściciela
owner_contact_number	numer kontaktowy właściciela

Tabela 1: Tabela cars

Tabela przechowuje szczegółowe informacje o samochodach, w tym ich markę, model, kolor oraz dane właściciela. W środowisku produkcyjnym tabela ta musiałaby posiadać informacje o wszystkich samochodach zajerejestrowanych w Polsce oraz ich właścicielach, ponieważ bez tych danych byłoby niemożliwe wygenerowanie raportu ze zdarzenia z poziomu aplikacji webowej. Dane w tej tabeli mogłby replikować się z rządowych baz danych.

Tabela plates zawiera 9 pól:

Nazwa	**Opis**
id	-
timestamp_in	czas rozpoczęcia pomiaru
location_in	lokalizacja rozpoczęcia pomiaru
plate_number_in	numer rejestracyjny przy starcie pomiaru
timestamp_out	czas zakończenia pomiaru
location_out	lokalizacja zakończenias pomiaru
plate_number	numer rejestracyjny
plate_number_out	numer tablicy przy zakończeniu pomiaru
isDeleted	flaga usuniętego rekordu

Tabela 2: Tabela plates

Tabela używana do śledzenia wejścia i wyjścia pojazdów z ich numerami tablic w określonych miejscach i czasach. Jest kluczowa z punktu widzenia systemu radaru odcinkowego, ponieważ zbiera informacje o wykroczeniach prędkości i na jej podstawie analizowana jest prędkość średnia na wybranym odcinku drogi.

Tabela raport zawiera 5 pól:

Nazwa	**Opis**
id	-
timestamp_in	czas wygenerowania raportu
speed	prędkość
plates_id	identyfikator rekordu tablicy rejestracyjnej

Tabela 3: Tabela raport

Tabela używana do celów raportowania, zawiera informacje o wygenerowanym raporcie, przechowuje prędkość na zmierzonym odcinku oraz wysokość należnego mandatu wraz z kluczem obcym do tabeli plates.

Tabela users zawiera 3 pola:

Nazwa	**Opis**
id	-
username_in	nazwa użytkownika
password	hasło

Tabela 4: Tabela users

Tabela służy do przechowywania poświadczeń użytkowników z unikanymi nazwami oraz powiązanymi hasłami z procesu rejestracji konta. Na cele projektowe jest wystarczająca natomiast zostawia spore możliwości rozwoju, które powinny zostać wykorzystane podczas rozwiajanie funkcjonalości projektu.

1.3 Szczegółowa dokumentacja kodu

Omówienie katalogu **cut_plate_from_picture**. Katalog cut_plate_from_picture zawiera plik licenseplate.py, którego działanie zostanie opisane poniżej.

1. Biblioteki oraz konfiguracja pytesseract

```
import cv2
import pytesseract
import numpy as np
import os
import re
pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'
```

   Ta sekcja importuje niezbędne biblioteki i konfiguruje Pytesseract, określając ścieżkę do wykonawczego pliku Tesseract. Te biblioteki są niezbędne do przetwarzania obrazów (OpenCV), OCR (Pytesseract), manipulacji tablicami (NumPy) oraz do operacji na plikach (os, re).

2. Ekstrakcja i formatowanie nazw plików

```
filename_parts = os.path.basename(image_filename).split('_')
timestamp = filename_parts[0]
additional_text = filename_parts[1].split('.')[0]
```

   Program ekstrahuje informacje z nazwy pliku obrazu. Ta część kodu jest odpowiedzialna za rozdzielanie nazwy pliku na komponenty, aby uzyskać znacznik czasu i dodatkowy tekst, które mogą być używane do dalszego przetwarzania lub nazywania plików wyjściowych.

3. Funkcja przetwarzania obrazu - process_image(image_filename)

   To serce skryptu, gdzie odbywa się przetwarzanie obrazu i OCR.

```
def process_image(image_filename):
    img = cv2.imread(image_filename)
    imgray1 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    canny = cv2.Canny(imgray1, 313, 480)
    contours, _ = cv2.findContours(canny, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
```

   Skrypt odczytuje obraz, konwertuje go na odcienie szarości i używa detektora krawędzi Canny do znalezienia krawędzi. Następnie pobiera kontury z obrazu, priorytetyzując je według wielkości.

4. Przetwarzanie konturów i izolacja tablic rejestracyjnych

```
for i in contours:
    area = cv2.contourArea(i)
    approx = cv2.approxPolyDP(i, 0.01*cv2.arcLength(i, True), True)
    if len(approx) == 4 and area > 800:
        cv2.drawContours(img, [approx], 0, (0, 255, 0), 2)
        x, y, w, h = cv2.boundingRect(i)
        img4 = imgray1[y:y+h, x:x+w]
```

   Iteruje przez kontury, aby znaleźć czworokąt (prawdopodobnie tablicę rejestracyjną) na podstawie jego kształtu i wielkości. Gdy prawdopodobny kandydat zostanie znaleziony, wyodrębnia tę część obrazu do OCR.

5. Ekstrakcja tekstu OCR

```
try:
    license_plate = pytesseract.
    image_to_string(img4, config='—psm 7')
except:
    print("Nie mozna odczytac
    tablicy rejestracyjnej.")
    return
```

   Używa Pytesseract do wstępnego odczytania tablicy rejestracyjnej. Jest to opcja która może zostać wykorzystana przy ewentualnej rozbudowie programu. Blok 'try-except' zapewnia, że program nie zakończy się awaryjnie, jeśli OCR nie uda się odczytać tekstu.

6. Post-processing i wyjście tekstu

```
plate_number_raw = pytesseract.image_to_string(morph_img, config='—psm 8 —oem 3')
plate_number = plate_number_raw.replace(':', '').strip()
print("Extracted Plate Number:", plate_number)
```

Po OCR skrypt wykonuje pewne czyszczenie wyekstrahowanego tekstu, usuwając niechciane znaki i przycinając białe znaki, a następnie wyświetla oczyszczony numer tablicy.

7. Zapisywanie i obsługa plików

```
cv2.imwrite('workdir/only_plate.jpg', img4)
...
cv2.imwrite(full_save_path, white_canvas)
...
if os.path.exists(work_directory):
    for filename in os.listdir(work_directory):
        os.unlink(work_path)
```

Skrypt zapisuje średnie i końcowe przetworzone obrazy do przeglądu lub dalszego użytku. Docelowe pliki są zostają zapisane w podfolderze images. Skrypt również kod do czyszczenia miejsca pracy przez usunięcie tymczasowych plików po przetwarzaniu w katalogu workdir.

8. Kontrola przetwarzania wsadowego

```
start_files = []
meta_files = []

for filename in os.listdir(directory):
    if filename.endswith(start + '.jpg'):
        start_files.append(filename)
    elif filename.endswith(meta + '.jpg'):
        meta_files.append(filename)

def process_files(file_list):
    for filename in file_list:
        file_path = os.path.join(directory, filename)
        process_image(file_path)
```

Program segreguje pliki na podstawie ich nazw końcówek, tworząc odrębne listy dla różnych kategorii plików (tutaj jako przykład ”balice”i ”chrzanow”). Następnie używa funkcji process_files do przetworzenia każdego zestawu plików. To organizuje przetwarzanie w bardziej zarządzalne i modularne grupy.

9. Wstępna obsługa plików graficznych

Dostarczony skrypt to rozwiązanie do przetwarzania wstępnego obrazów w celu ekstrakcji tablic rejestracyjnych z zidentyfikowanych pojazdów samochodowych. Łączy on techniki przetwarzania obrazów w celu przygotowania ich do OCR, a następnie używa Pytesseract do ekstrakcji tekstu natomiast głównym jego założeniem jest wydobycie rejestracji ze zdjęcia i przedstawienie jej jako grafiki na białym tle w celu dalszego przetwarzania. Skrypt jest zorganizowany tak, aby automatycznie obsługiwać wiele obrazów.

Następnym katalogiem do omówienia jest **find_plate_number**. W tym katalogu znajduje się kod odpowiedzialny za rozpoznanie numerów rejestracyjnych z obrazu przygotowanego przez kod z cut_plate_from_picture.

Katalog find_plate_number zawiera następujące elementy:

- Skrypt główny main.py
- Plik wyników results.json
- Podkatalogi registration_processing i resources

Podkatalog registration_processing zawiera kluczowe skrypty do przetwarzania i rozpoznawania tablic rejestracyjnych. Podkatalog resources przechowuje obrazy liter i cyfr używane w procesie rozpoznawania. Skrypt główny main.py koordynuje cały proces rozpoznawania tablic rejestracyjnych, włączając w to wczytywanie obrazów, ekstrakcję danych, rozpoznawanie numerów rejestracyjnych, oraz interakcje z bazą danych. Wykorzystuje moduły z 'registration_processing' do wykonania wybranych zadań.

Moduł registration_processing zawiera dwa skrypty:

- character_classifier.py odpowiada za trenowanie klasyfikatora znaków i generowanie ich konturów.
- license_plate_recognizer.py zawiera logikę do rozpoznawania tablic rejestracyjnych, w tym przetwarzanie obrazów, wyszukiwanie potencjalnych tablic i znaków, a także końcowe rozpoznawanie i weryfikację tablic rejestracyjnych.

Opis character_classifier.py

```
import numpy as np
import cv2
import glob
import re
import os

MIN_CONTOUR_AREA = 100
RESIZED_IMAGE_WIDTH = 20
RESIZED_IMAGE_HEIGHT = 30
```

Ten fragment kodu rozpoczyna się od importowania potrzebnych bibliotek, takich jak numpy (do pracy z tablicami numerycznymi), cv2 (OpenCV do przetwarzania obrazów), glob (do wyszukiwania plików w określonym katalogu), re (do obsługi

wyrażeń regularnych) i os (do operacji na systemie plików). Następnie definiowane są pewne stałe, takie jak minimalna powierzchnia konturu (MIN_CONTOUR_AREA) oraz szerokość i wysokość obrazu po przeskalowaniu (RESIZED_IMAGE_WIDTH i RESIZED_IMAGE_HEIGHT).

```
def get_chars_contour():
    images_dir = "resources/characters/"
    data_path = os.path.join(images_dir, '*g')
    files = glob.glob(data_path)

    chars_contour = {}

    for f1 in files:
        img_character = cv2.imread(f1, 0)
        letter = re.findall(r"q\w", f1)
        img_letter_edges = cv2.Canny(img_character, 30, 200)
        contours, hierarchy = cv2.findContours(img_letter_edges.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
        chars_contour[str(letter[0][1])] = contours

    return chars_contour
```

Funkcja get_chars_contour służy do wczytywania obrazów znaków, przetwarzania ich w celu uzyskania konturów i zapisywania tych konturów w słowniku chars_contour. Proces ten polega na:

- Określeniu ścieżki do katalogu z obrazami znaków.
- Wyszukaniu wszystkich plików obrazów w katalogu.
- Iterowaniu przez każdy plik obrazu znaku.
- Wczytaniu obrazu, odczytaniu informacji o znaku z nazwy pliku, wygenerowaniu obrazu krawędzi znaku i znalezieniu jego konturu.
- Zapisaniu konturu znaku w słowniku, gdzie kluczem jest znak.

```
def train_classifier(chars{\_}contour):
    img_training_chars = cv2.imread("resources/all{\_}characters.jpg")
    if img_training{\_}chars is None:
        print("blad nie mozna odczytac pliku. Sprawdz sciezke pliku")
        os.system("pause")
        return

    img_gray = cv2.cvtColor(img_training_chars, cv2.COLOR_BGR2GRAY)

    ret, img_thresh = cv2.threshold(img_gray, 150, 255, cv2.THRESH_BINARY_INV)
    img_thresh_copy = img_thresh.copy()

    npa_contours, npa_hierachy = cv2.findContours(img_thresh_copy,
                                                  cv2.RETR_EXTERNAL,
                                                  cv2.CHAIN_APPROX_SIMPLE)

    npa_flattened_images = np.empty((0, RESIZED_IMAGE_WIDTH*RESIZED_IMAGE_HEIGHT))
    int_classifications = []

    for idx, npa_contour in enumerate(npa_contours):
        if cv2.contourArea(npa_contour) > MIN_CONTOUR_AREA:
            [intX, intY, intW, intH] = cv2.boundingRect(npa_contour)

            img_ROI = img_thresh[intY-5:intY+intH+5, intX-5:intX+intW+5]
            img_ROI1 = img_thresh[intY:intY+intH, intX:intX+intW]
            img_ROI_resized = cv2.resize(img_ROI1, (RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))

            img_ROI_edges = cv2.Canny(img_ROI.copy(), 30, 200)

            contours_ROI, hierarchy_ROI = cv2.findContours(img_ROI_edges.copy(),
                                                           cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
            matches = {}
            for letter, letter_cntr in chars_contour.items():
                ret = cv2.matchShapes(letter_cntr[0], contours_ROI[0], 1, 0.0)
                matches[letter] = ret
            best = min(matches, key=matches.get)

            if idx == 0 and best == '6':
                best = '9'
            if idx == 7 and best == 'S':
                best = '2'
            if idx == 10 and best == 'O':
                best = 'M'
            if idx == 11 and best == 'O':
                best = "N"
            if idx == 15 and best == 'O':
                best = 'X'
            if idx == 23 and best == '0':
                best = 'D'
            if idx == 30 and best == 'O':
                best = 'W'
            int_classifications.append(ord(best))
            npa_flattened_image = img_ROI_resized.reshape((1, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT))
            npa_flattened_images = np.append(npa_flattened_images, npa_flattened_image, 0)

    flt_classifications = np.array(int_classifications, np.float32)
    npa_classifications = flt_classifications.reshape((flt_classifications.size, 1))
    return npa_classifications, npa_flattened_images
```

Funkcja train_classifier służy do przetwarzania obrazu treningowego zawierającego wiele znaków, wyodrębniania poszczególnych znaków, dopasowywania ich do wcześniej zdefiniowanych konturów znaków i tworzenia zestawu spłaszczonych ob-

razów znaków wraz z odpowiadającymi im klasyfikacjami. Metoda została dostosowana z repozytorium github.com: OpenCV_3_KNN_Character_Recognition_Python.

Proces ten obejmuje:

- Wczytanie obrazu treningowego zawierającego wszystkie znaki.
- Konwersję obrazu na odcienie szarości.
- Progowanie obrazu.
- Znajdowanie konturów dla wszystkich znaków na obrazie.
- Przygotowanie pustej tablicy numpy (npa_flattened_images) do przechowywania spłaszczonych obrazów znaków.
- Przygotowanie pustej listy (int_classifications) do przechowywania klasyfikacji znaków.

Dodatkowo dla każdego konturu znaku:

- Sprawdzenie, czy kontur jest wystarczająco duży.
- Wyodrębnienie obszaru zainteresowania (ROI) wokół konturu.
- Dopasowanie konturu ROI do znaków treningowych i wybór najlepszego dopasowania.
- Dodanie klasyfikacji znaku do listy int_classifications jako kod Unicode.
- Spłaszczenie obrazu ROI i dodanie go do npa_attened_images.

Na końcu funkcja zwraca tablicę numpy npa_classifications zawierającą klasyfikacje znaków oraz npa_flattened_images zawierającą spłaszczone obrazy znaków.

Opis license_plate_recognizer.py

```
import numpy as np
import cv2
kNearest = cv2.ml.KNearest_create()
PLATE_HEIGHT_TO_WIDTH_RATIO = 90 / 520
CHAR_RATIO_MIN = 0.25
CHAR_RATIO_MAX = 0.85
LICENSE_PLATE_LENGTH = 7
RESIZED_CHAR_IMAGE_WIDTH = 20
RESIZED_CHAR_IMAGE_HEIGHT = 30
SHOW_STEPS = False
```

Główna funkcja recognize_license_plate koordynuje cały proces rozpoznawania tablicy rejestracyjnej, od przetwarzania wstępnego po końcową weryfikację i rozpoznawanie znaków. Stałe zdefiniowane w skrypcie license_plate_recognizer.py są kluczowymi wartościami, które sterują różnymi aspektami procesu rozpoznawania tablic rejestracyjnych. Oto ich szczegółowy opis:

- PLATE_HEIGHT_TO_WIDTH_RATIO - określa stosunek wysokości do szerokości typowej polskiej tablicy rejestracyjnej. W Polsce standardowy wymiar tablicy to 520 mm szerokości na 114 mm wysokości, co daje stosunek około 0,2192 (114/520). Jest to używane do identyfikowania potencjalnych tablic rejestracyjnych na obrazie poprzez porównywanie wymiarów znalezionych konturów z tym stosunkiem.

- CHAR_RATIO_MIN i CHAR_RATIO_MAX - Określają minimalny i maksymalny akceptowalny stosunek wysokości do szerokości pojedynczego znaku na tablicy rejestracyjnej. Te proporcje są wykorzystywane do filtrowania potencjalnych znaków na podstawie ich geometrycznych cech, eliminując te, które nie pasują do typowych rozmiarów liter i cyfr.

- LICENSE_PLATE_LENGTH - Oznacza oczekiwaną liczbę znaków na polskiej tablicy rejestracyjnej, która wynosi 7. Jest to kluczowe dla procesu weryfikacji i finalizacji rozpoznawania tablicy rejestracyjnej, aby upewnić się, że ostateczny wynik zawiera odpowiednią liczbę znaków.

- RESIZED_CHAR_IMAGE_WIDTH i RESIZED_CHAR_IMAGE_HEIGHT - to wymiary (szerokość i wysokość), do których każdy potencjalny znak jest skalowany przed przetwarzaniem przez klasyfikator (np. KNN). Jednolite rozmiary są ważne dla spójności danych wejściowych do modelu klasyfikacji. W przypadku tego skryptu wymiary te wynoszą odpowiednio 20 i 30 pikseli.

- SHOW_STEPS - jest to zmienna logiczna (prawda/fałsz), która kontroluje, czy skrypt powinien wyświetlać poszczególne kroki i wyniki pośrednie procesu rozpoznawania tablic rejestracyjnych (np. wyświetlanie obrazów w trakcie przetwarzania). Ustawienie jej na True umożliwia wizualizację i debugowanie procesu, podczas gdy False prowadzi do cichszego wykonania bez wyświetlania dodatkowych informacji.

```
def train_KNN(classifications, flattened_images):
    npa_classifications = classifications.astype(np.float32)
    npa_flattened_images = flattened_images.astype(np.float32)
    npa_classifications = npa_classifications.reshape((npa_classifications.size, 1))
    kNearest.setDefaultK(1)
    kNearest.train(npa_flattened_images, cv2.ml.ROW_SAMPLE, npa_classifications)
    return True
```

Funkcja ‘train_KNN‘ służy do trenowania klasyfikatora k-Nearest Neighbors (k-NN) używanego do rozpoznawania znaków na tablicach rejestracyjnych. Konwersja typów danych:

- **classifications** i **flattened_images** są konwertowane na typ ‘np.float32‘. Jest to standardowa praktyka w uczeniu maszynowym, ponieważ wiele algorytmów działa lepiej lub wymaga danych zmiennoprzecinkowych.
- **classifications** to tablica zawierająca etykiety klas (np. której litery czy cyfry odpowiada dany obraz).
- **flattened_images** to spłaszczone obrazy znaków, które są używane jako dane wejściowe dla klasyfikatora. Spłaszczenie polega na przekształceniu obrazu z dwuwymiarowej matrycy pikseli na jednowymiarowy wektor.

Przekształcenie klasyfikacji - npa_classifications jest przekształcana tak, aby miała wymiary (‘n‘, 1), gdzie ‘n‘ to liczba próbek. Jest to konieczne, ponieważ OpenCV oczekuje, że etykiety klas będą w tej formie.

Ustawienie parametru ’K’ dla k-NN - kNearest.setDefaultK(1) ustawia parametr K klasyfikatora k-NN na 1. Oznacza to, że klasyfikacja danej próbki będzie oparta na najbliższym sąsiedzie w przestrzeni cech.

Trenowanie klasyfikatora - metoda ‘train‘ klasyfikatora k-NN jest wywoływana z ‘npa_flattened_images‘ jako danymi treningowymi i ‘npa_classifications‘ jako etykietami. Parametr ‘cv2.ml.ROW_SAMPLE‘ informuje klasyfikator, że każdy wiersz ‘npa_flattened_images‘ stanowi osobną próbkę.

Zwracanie wyniku -funkcja zwraca ‘True‘, sygnalizując pomyślne zakończenie procesu trenowania.

Opis pozostałych fukcji z pliku license_plate_recognizer.py:

1. get_potential_chars_ROI: Szuka potencjalnych regionów zawierających znaki na tablicach rejestracyjnych.
2. recognize_chars_in_plate: Rozpoznaje znaki na tablicach rejestracyjnych na podstawie ROI.

3. license_plate_rules: Sprawdza, czy rozpoznane tablice rejestracyjne pasują do polskich standardów tablic rejestracyjnych.

4. preprocess: Przygotowuje obraz do dalszego przetwarzania poprzez skalowanie, rozmycie i wykrywanie krawędzi.

5. find_potential_plates_vertices: Szuka potencjalnych tablic rejestracyjnych na obrazie.

6. get_birds_eye_view: Zmienia perspektywę potencjalnych tablic rejestracyjnych na widok z góry.

7. find_potential_chars_on_plates: Wyszukuje potencjalne znaki na tablicach rejestracyjnych.

8. three_chars_in_first_part: Sprawdza, czy na tablicy rejestracyjnej są trzy znaki w pierwszej części.

9. recognize_license_plate: Główna funkcja do rozpoznawania tablic rejestracyjnych na obrazach.

Opis main.py

Skrypt main.py w projekcie rozpoznawania tablic rejestracyjnych pełni funkcję koordynatora całego procesu rozpoznania numerów rejestracyjnych pojazdów. Przeanalizujmy szczegółowo jego działanie i strukturę:

1. Importowanie Bibliotek

   Skrypt rozpoczyna się od importowania niezbędnych bibliotek i modułów, takich jak argparse, json, pathlib, re, cv2 (dla operacji na obrazach), mysql.connector (do obsługi bazy danych) oraz funkcji z modułu registration_processing.

2. Funkcje Pomocnicze

   extract_timestamp_from_filename: Wyodrębnia datę i czas z nazw plików. insert_into_database_start i update_database_meta: Odpowiadają za komunikację z bazą danych MySQL, wstawiając i aktualizując rekordy dotyczące rozpoznanych tablic rejestracyjnych.

3. Przetwarzanie Obrazów

   process_images: Ta funkcja przetwarza listę obrazów, używając recognize_license_plate do identyfikacji numerów rejestracyjnych. Wyniki są przechowywane w słowniku.

4. Główna funkcja main:

    - Parsowanie Argumentów
      Skrypt akceptuje argumenty dotyczące lokalizacji obrazów i pliku wynikowego.
    - Przygotowanie Danych
      Obrazy są sortowane i przetwarzane w zależności od ich lokalizacji (np. Balice, Chrzanów).
    - Trening Klasyfikatora
      Wywołuje funkcje get_chars_contour, train_classifier, i train_KNN do przygotowania modelu do rozpoznawania znaków.
    - Rozpoznawanie Tablic Rejestracyjnych
      Skrypt przetwarza obrazy, rozpoznaje tablice rejestracyjne i zapisuje wyniki do pliku JSON
    - Interakcja z Bazą Danych
      Wyniki są następnie przekazywane do funkcji zarządzających bazą danych.

5. Wykonywanie Skryptu

   Instrukcja if ___name___ == '___main___': zapewnia, że skrypt main.py działa jako główny program, kiedy jest uruchamiany bezpośrednio, a nie jako moduł importowany do innego skryptu.

Skrypt main.py jest sercem systemu i koordynuje wszystkie kluczowe etapy procesu rozpoznawania tablic rejestracyjnych. Od wczytywania obrazów, poprzez ich analizę, rozpoznawanie znaków, aż po zapis wyników i interakcję z bazą danych. Jest to dobrze zorganizowany i modułowy kod, który efektywnie integruje różne aspekty systemu rozpoznawania tablic rejestracyjnych.

Opis katalogu apache

Katalog apache odpowiedzialny za serwis apache2 składa się z następujących plików i katalogów:

- auth_check.php
  Skrypt odpowiedzialny za weryfikację, czy użytkownik jest zalogowany. Jest wykorzystywany w innych skryptach do sprawdzenia, czy użytkownik ma dostęp do określonych stron lub funkcji.

- delete_record.php
  Skrypt do usuwania rekordów z bazy danych. Zamiast usuwania rekordu całkowicie, zmienia on flagę isDeleted na 1, co oznacza tzw. "miękkie usunięcie". Umożliwia to zachowanie danych dla celów archiwizacyjnych.

- generate_report.php
  Skrypt generuje raport na podstawie wybranego rekordu. Pobiera dane z bazy, oblicza prędkość na podstawie czasu i lokalizacji, a następnie tworzy raport w formacie PDF, który jest następnie przekazywany do użytkownika do pobrania.

- index.php
  Ten plik wyświetla interfejs użytkownika dla strony głównej. Zawiera tabelę z rekordami dotyczącymi przypadków przekroczenia prędkości, włączając informacje takie jak ID, czas wejścia i wyjścia, lokalizację oraz numer rejestracyjny pojazdu. Użytkownicy mogą generować raporty lub usuwać rekordy.

- login.php
  Skrypt logowania do systemu. Użytkownicy wpisują swoje dane (nazwę użytkownika i hasło), które są weryfikowane z bazą danych. W przypadku pomyślnego zalogowania użytkownik jest przekierowywany do strony głównej.

- logut.php
  Ten plik służy do wylogowywania użytkowników. Usuwa wszystkie zmienne sesyjne i niszczy sesję, a następnie przekierowuje użytkownika do strony logowania.

- registration.php
  Skrypt obsługujący proces rejestracji nowych użytkowników. Umożliwia wprowadzenie nazwy użytkownika oraz hasła (i jego potwierdzenia), a następnie zapisuje te dane w bazie danych.

- TCPDF

Katalog tcpdf to zewnętrzna biblioteka PHP służąca do generowania dokumentów PDF w aplikacjach internetowych. Jest to narzędzie open-source, które umożliwia tworzenie zaawansowanych dokumentów PDF z wykorzystaniem kodu HTML i CSS, co ułatwia projektowanie i formatowanie. TCPDF oferuje wsparcie dla różnorodnych

języków, w tym tych pisanych od prawej do lewej, oraz zawiera szeroki zakres wbudowanych fontów. Umożliwia generowanie kodów kreskowych i QR, wstawianie linków, obrazów, grafiki wektorowej oraz znaków wodnych

1.4 Środowisko programistyczne

Do pracy nad projektem można użyć dowolnego systemu operacyjnego. Dla systemu Windows idealnym rozwiązaniem jest Visual Studio Code (VSCode), ponieważ jest doskonałym wyborem zarówno dla programowania w Pythonie 3.7, jak i w PHP. Do poprawnego działania wymagana będzie biblioteka TCPDF w wersji 6.6.5, PHP w wersji 8.2. Dla środowiska napisanego w języku Python wymaganna jest instalacja opencv-python w wersji 4.4.0.44 oraz numpy 1.18.3.

Literatura (jeżeli wymagana)