Wstęp

Niniejszy raport opisuje proces realizacji projektu obejmującego implementację systemu rozpoznawania tablic rejestracyjnych z użyciem technologii OCR oraz Machine Learnig. Raport obejmuje zastosowane rozwiązania technologiczne, napotkane wyzwania oraz pełną listę wykorzystanych źródeł i materiałów.

Proces Realizacji Projektu

Projekt składał się z dwóch głównych etapów, w których stworzyliśmy dwa fragmenty kodu. Poniżej znajdują się szczegółowe opisy każdego z etapów.

Etap 1: detect.py

Zastosowane Rozwiązania

- Kod wykorzystuje wiele bibliotek takich jak *os*, *sys*, *numpy, torch, cv2, easyocr i csv* do realizacji różnych funkcji w projekcie.
- Inicjalizacja różnych komponentów, takich jak katalogi robocze i modele detekcji, jest kluczowa do zapewnienia poprawnego działania całego systemu.
- Funkcje takie jak *deskew_image, sharpen_image i smooth_edges* odpowiadają za przetwarzanie obrazów w celu polepszenia wyników OCR.
- EasyOCR jest używane do odczytu tablic rejestracyjnych ze zdjęć. Kod wykorzystuje funkcję *reader.readtext* do ekstrakcji tekstu.
- Wyniki oraz odrzucenia są zapisywane w plikach CSV, co umożliwia późniejszą analizę danych.

Napotkane Wyzwania

- Trenowanie modelu, oraz doprowadzenia go do uruchomienia
- Wyzwanie stanowiła poprawna detekcja i rozpoznanie tablic rejestracyjnych z uwzględnieniem różnych warunków oświetleniowych i jakości obrazów.
- Zapewnienie odpowiedniej szybkości przetwarzania przy jednoczesnym zachowaniu dokładności było istotnym aspektem przy wyborze modelu oraz metody OCR.
- Integracja różnych bibliotek i narzędzi wymagała dokładnego zarządzania zależnościami i środowiskiem uruchomieniowym.

Etap 2: gui.py

Zastosowane Rozwiązania

- Zastosowano bibliotekę *tkinter* do stworzenia graficznego interfejsu użytkownika, umożliwiającego łatwe korzystanie z funkcji detekcji.
- Aplikacja umożliwia użytkownikowi wybranie pliku wideo lub obrazu, zaznaczenie opcji zapisywania wyników, trybu debugowania i uruchomienie procesu detekcji.
- Kod integruje się z wcześniej napisanym modułem *detect*, umożliwiając wykonywanie detekcji tablic rejestracyjnych z poziomu GUI.
- Aplikacja posiada obsługę błędów i wyświetla odpowiednie komunikaty w przypadku wystąpienia problemów.

Napotkane Wyzwania

- Skuteczna komunikacja między interfejsem użytkownika a funkcjami detekcji.
- Identyfikacja i obsługa potencjalnych problemów podczas procesu detekcji i zapisywania wyników.

Zastosowane Technologie i Narzędzia

- Python: Język programowania używany do napisania kodu.
- tkinter: Biblioteka używana do tworzenia graficznego interfejsu użytkownika.
- OpenCV: Biblioteka używana do przetwarzania obrazów.
- PyTorch: Biblioteka używana do realizacji sieci neuronowych i modeli detekcji.
- EasyOCR: Narzędzie do optycznego rozpoznawania znaków (OCR).
- CSV: Format plików do zapisywania wyników.
- YOLOv9: Model wyszkolony i wykorzystany do rozpoznawania obiektów (tablic rejestracyjnych)

Wykorzystane Źródła i Materiały

- Dokumentacja Python: Oficjalna dokumentacja języka Python dostępna na python.org.
- Dokumentacja tkinter: Oficjalna dokumentacja biblioteki tkinter dostępna na tkdocs.com.
- Dokumentacja OpenCV: Oficjalna dokumentacja biblioteki OpenCV dostępna na docs.opencv.org.
- Dokumentacja PyTorch: Oficjalna dokumentacja biblioteki PyTorch dostępna na pytorch.org.
- EasyOCR GitHub Repository: Dokumentacja i przykłady dostępne na GitHub.
- YOLOv9 model: YOLOv9
- Trenowanie modelu: openalpr
- Stack Overflow: Forum dyskusyjne