System rozpoznawania lokalizacji na podstawie nagrań z kamerek samochodowych

Analiza i Przetwarzanie Obrazów

1. Skład osobowy zespołu i podział obowiązków.

- Kamil Jaśkiewicz – detekcja i rozpoznawanie znaków pionowych, integracja aplikacji
- Jakub Pietrzyk – przygotowanie zbioru danych
- Michał Piwowarczyk – detekcja i rozpoznawanie cech ludzkich
- Bartosz Sękowski – detekcja i analiza rejestracji samochodowych, interfejs aplikacji
- Jarosław Skwarczek – detekcja napisów i analiza języka
- Michał Stasiak – analiza krajobrazu, redakcja dokumentów
- Mikołaj Wnęk – analiza znaków poziomych oraz kierunku jazdy

2. Działanie systemu.

Klasyfikację krajów oparto na osobistych doświadczeniach w grze GeoGuesser, opierając wybór kraju na detalach i szczególnych cechach różniących poszczególne kraje.

Z uwagi na niską jakość lub niewystarczające pokrycie krajów przez zbiory danych nagrań z kamer samochodowych dostępnych w Internecie, zdecydowaliśmy się na własnoręczne przygotowanie zbioru danych bazując na skrypcie pobierającym nagrana z serwisu YouTube. Z wejściowych nagrań kamery samochodu wybrane zostały klatki z częstotliwością jednej klatki na sekundę filmu. Niestety, nie udało nam się zrealizować pomysłu wyboru klatek bazując na ich różnorodności.

Klatki trafiają do niezależnych modułów odpowiedzialnych za detekcję wybranej cechy i klasyfikację kraju pochodzenia nagrania na jej podstawie. Każdy z modułów zwraca wektor prawdopodobieństw klasyfikacji krajów. Prawdopodobieństwa zawarte w wektorach wynikać mogą z działania modelu AI lub nadane zostaną empirycznie – np. w przypadku detekcji kierunku jazdy po prawej stronie ulicy, kraje lewostronne otrzymają prawdopodobieństwo 0, kraje lewostronne prawdopodobieństwo $1/n$, gdzie n to liczba wykrytych krajów o ruchu prawostronnym.

Na końcu, następuje obliczenie średniej ważonej wektorów prawdopodobieństw. Wagi modułu sumującego zostały wybrane empirycznie. Interfejs przedstawia wybrane klatki z zaznaczonymi na nim zidentyfikowanymi znakami, napisami, rejestracjami i ludźmi, oraz przedstawia ranking klasyfikowanych krajów. W razie braku detekcji pewnych elementów, moduły im dedykowane zwracają wyzerowany wektor prawdopodobieństw.

3. Zbiór danych.

Kraje w każdym z wektorów są identyczne i wybrane zostały na podstawie zebranego zbioru danych, bazując na jak największym zróżnicowaniu krajobrazu, alfabetu, znaków drogowych i typowej karnacji ludzi oraz ich liczności w zbiorze. Materiały wideo pobierane są z serwisu YouTube za pomocą skryptu. Wybrano następujące kraje :

a. Ghana
b. Kenia
c. Republika Południowej Afryki
d. Japonia
e. Chiny
f. Tajlandia
g. Iran
h. Szwecja
i. Czechy
j. Austria
k. Francja
l. Stany Zjednoczone
m. Kanada
n. Meksyk
o. Chile
p. Peru
q. Argentyna
r. Australia
s. Nowa Zelandia
t. Fidżi

Część z pobranych materiałów wideo nie są faktycznymi filmami z kamer samochodowych, jednak zostały pozostawione ponieważ wykazują znacznie lepszą jakość.

4. Działanie poszczególnych modułów.

4.1. Analiza krajobrazu (waga 3%).

Moduł analizy krajobrazu, korzystający z biblioteki OpenCV2, opiera działanie na analizie dominujących kolorów każdej z klatek przy wykorzystaniu algorytmu KMeans. Następnie, obliczana jest odległość w przestrzeni HSV pomiędzy dominującymi kolorami a kolorami utożsamianymi z jednym z trzech klimatów : europejskim, tropikalnym i pustynnym. Bazując na odległościach obliczane jest prawdopodobieństwo przynależności klatki do każdego z klimatów. Następnie, prawdopodobieństwa te są konwertowane do postaci wektora krajów bazując na wagach wybranych empirycznie. Niestety, bazując na naszym zbiorze danych, model w większości przypadków zwraca wektor krajów o niewielkiej wariancji. Model jest w stanie rozpoznać docelowy klimat (przypisać mu prawdopodobieństwo > 50%), jednak ze względu na wiele krajów o zbliżonych klimacie (np. Czechy i Austria) oraz krajach o wielu klimatach (np. Argentyna i Chiny), jego ostateczna skuteczność jest niewielka.

4.2. Detekcja i rozpoznawanie cech ludzkich (waga 15%).

Model opiera swoje działanie na detekcji sylwetek ludzi na zdjęciu, a następnie próbie predykcji ich narodowości. Na model składają się trzy podmodele. Pierwszy wynajduje na obrazie człowieka i wycina go. Wycięty obraz wysyłany jest do modelu wykrywającego twarz człowieka. Jest to niezbędne ponieważ ostatni model, który ma za zadanie określić narodowość człowieka wymaga zdjęcia samej twarzy. Ostatni model dokonuje podziału na jedną z grup – białoskóry, czarnoskóry, azjatycki, hinduski lub inny. Wszystkie sklasyfikowane narodowości na obrazie są zbierane i na ich podstawie tworzony jest jeden wspólny wektor prawdopodobieństw. Wagi każdej z narodowości oparte są na procentowym udziale każdej z grup w populacji kraju. Ostatecznie ze względu na bardzo niską jakość nagrań, model odnajdujący twarze nie radzi sobie zbyt dobrze i bardzo często wynajduje twarze z bardzo niską pewnością (rzędu 20-30%). Na lepszej jakości nagraniach model radził sobie znacznie lepiej, natomiast określenie rasy nie jest wystarczające do dokładnego określenia kraju, może bardziej służyć do wykluczania niektórych państw.

4.3. Detekcja i analiza rejestracji samochodowych (waga 20%).

Do detekcji rejestracji samochodowych zostały użyte dwa pre-trenowane modele YoloV8 wzorowane na **https://github.com/Muhammad-Zeerak-Khan/Automatic-License-Plate-Recognition-using-YOLOv8**. Jeden model wykrywa na zdjęciu różnego typu pojazdy. Wykryte obiekty są przekazywane do drugiego modelu który wyodrębnia z nich tylko tablice rejestracyjne. Model do wykrywanie samych tablic czasami błędnie interpretuje część zdjęcia jako tablice, ponieważ czasami na pojeździe można znaleźć inne napisy. Do oceny kraju do jakiego należy tablica rejestracyjna został stworzony prosty klasyfikator, trenowany na podstawie tablic rejestracyjnych ze strony **https://geomastr.com/license-plates/**. Tablice są tutaj w „idealnym stanie" więc dane zostały poddane augmentacji, aby wzbogacić zbór treningowy. Do stworzenia tego klasyfikatora wykorzystano klasyczne zestawy narzędzi pochodzące z bibliotek *tensorflow*, *keras* oraz *numpy*. Zdjęcia tablic pochodzące z nagrać wideo niestety nie są w dobrej jakości co znacznie utrudniało ocenę klasyfikatora. Próby użycia filtrów np. wyostrzającego nie przynosiła zadowalających efektów, które mogło podnieść prawdopodobieństwo poprawnej klasyfikacji. Także różne rozmiary tablic występujące na nagraniach oraz podobieństwo między krajami (np. tablice z USA oraz Kanady) wpływały na błędną ocenę klasyfikatora.

4.4. Detekcja napisów i analiza języka (waga 20%).

Model opiera swoje działanie na narzędziu tesseract-ocr. Dokonuje detekcji napisów oraz klasyfikuje ich język przy pomocy *landdetect*. Ponieważ wiele z wybranych krajów współdzieli język angielski, model sprawdza się najlepiej dla charakterystycznych języków (np. chiński, japoński). Niestety, narzędzia te mają trudności z odczytem języka o nieregularnych czcionkach, często spotykanych na banerach reklamowych. Model jest również podatny na napisy na rejestracjach, które nie wykazują żadnego z języków.

4.5. Detekcja i analiza znaków pionowych (waga 30%).

Model opiera się na klasyfikacji znaków pionowych bazując na https://geomastr.com/street-signs. Działanie tego modułu podzielono na dwa etapy. Pierwszy z nich to wykrycie znaku na zdjęciu, a drugi to sklasyfikowanie typu znaku. Do wykrywania znaku na zdjęciu użyto wytrenowanego modelu Yolov8 (https://github.com/Mkoek213/road_detection_model). Zaletą tego konkretnego modelu, była dodatkowa klasa "other sign", która umożliwiła wykrywanie dowolnego znaku. Na podstawie wyciętego z pierwotnego obrazu znaku, dokonywana jest analiza jego koloru oraz kształtu. Analiza koloru została wykonana za pomocą konwersji do HSV. Dla każdego koloru (żółtego, niebieskiego i czerwonego) tworzona jest odpowiednia maska, maski są nakładane na wyjściowe zdjęcia. Następnie wykonywana są operacje otwarcia i zamknięcia. Kolejno wydobywane są kontury przy użyciu funkcji cv2.findContours. Dla każdego konturu wykonywana jest analiza kształtu za pomocą funkcji detect_shape. Funkcja ta oblicza podstawowe cechy geometryczne konturu: pole powierzchni, obwód, liczbę wierzchołków po aproksymacji konturu, kołowość (circularity), współczynnik proporcji (aspect ratio). Na podstawie tych cech kontur klasyfikowany jest jako jeden z podstawowych kształtów: koło, trójkąt, romb lub prostokąt. Kryteria klasyfikacji bazują m.in. na liczbie wierzchołków oraz kołowości (np. wysoka kołowość i liczba wierzchołków ≥ 6 sugerują koło). Po rozpoznaniu kształtów kod zlicza wystąpienia poszczególnych figur w kontekście ich koloru. Na tej podstawie przyznawane są punkty dla krajów z dwóch predefiniowanych grup (np. wykrycie żółtego rombu zwiększa prawdopodobieństwo pochodzenia znaku z pewnej grupy krajów). System punktowy oparty jest na prostych regułach heurystycznych, które bazują na często występujących typach znaków drogowych w danym regionie geograficznym. Naturalnie, model działa najlepiej w dobrym oświetleniu, najlepiej rozpoznaje żółte romby – znaki charakterystyczne dla krajów spoza Europy. Problem sprawiają mu zdjęcia wykonane w nocy oraz kraje o zróżnicowanych i ciężkich do zgeneralizowania kształtach i kolorach znaków.

4.6. Rozpoznawanie kierunku jazdy (waga 6%).

Model rozpoznaje kierunek jazdy bazując na wykrytych znakach, zakładając że znaki znajdą się po tej stronie ulicy po której jedzie pojazd. Niestety, model jest podatny na elementy obrazu rozpoznane jako znaki niebędące nimi oraz przypadki kiedy znaki nie odpowiadają kierunkowi jazdy (np. skrzyżowania, drogi wielopasmowe). Pomimo względnie zadowalającej umiejętności wykrycia kierunku jazdy, ta nie przekłada się na jednoznaczną klasyfikację kraju, przez co sam model niewiele wnosi do końcowej klasyfikacji.

4.7. Detekcja i analiza znaków poziomych (waga 11%).

Model rozpoznaje znaki poziome korzystając z modelu YOLO oraz analizy obrazów z wykorzystaniem biblioteki OpenCV2. Model bazuje na kolorystyce i formacie linii na drodze wzorując się na https://geomastr.com/road-lines/. Niestety, model jest bardzo podatny na zmienne oświetlenie na zdjęciu oraz generalną niską jakość znaków poziomych na wielu zdjęciach, którym nie pomaga próba korekcji kolorów. Pomimo względnie zadowalającej umiejętności wykrycia linii, nie zawsze możliwa jest jednoznaczna klasyfikacja kraju, przez co model jedynie nieznacznie wpływa na końcowy wynik.

4.8. Detekcja i analiza pojazdów (waga 0%).

Pomimo że przygotowany prototyp detekcji i analizy pojazdów otrzymywał zadowalające wyniki, nie udało nam się znaleźć wiarygodnej statystki dotyczącej udziału typów pojazdów w danych krajach, co uniemożliwia konwersję otrzymanych wyników modelu na wektor prawdopodobieństw. Zrezygnowaliśmy z tworzenia tego modelu.

5. Interfejs aplikacji.

Interfejs aplikacji został zbudowany z pomocą narzędzi oraz funkcji pochodzących z biblioteki *tkinter*. Aplikacja składa się z 4 paneli. Pierwszy panel zawiera elementy pozwalające wczytać, odtworzyć i zatrzymać wideo, włączyć analizę wideo oraz ustawić parametry dotyczące analizy - zakres filmu do analizy oraz ilość klatek do przeanalizowania na każda sekundę wideo. Drugi panel zawiera miejsce w który może obejrzeć wczytany film. Trzeci panel po zakończonej analizie pokazuje wydobyte elementy z wideo w postaci kilku podpaneli. Ostatni panel pokazuje końcowy wynik analizy jaki został obliczony podczas całego procesowania wideo. Po wczytaniu nagrania, wyświetlane są tam też jego parametry.

6. Uruchomienie.

Naturalnie, z powodu wielu wykorzystanych narzędzi, projekt zgromadził wiele zależności oraz dużych objętościowo plików zawierających wytrenowane modele lub porównawcze dane. Uruchomienie projektu wymaga instalacji zależności z pliku *requirements.txt*. Ponadto, w celu użycia analizy tekstu, wymagana jest obecność pakietu tesseract-ocr.

7. Podsumowanie.

Pomimo tego że końcowy wynik aplikacji nie jest idealny, udało się w jej ramach stworzyć kilka niezależnych modułów z wykorzystaniem różnorodnych narzędzi. Udało nam się odwzorować wiele technik używanych podczas realnej gry w GeoGuesser. Udało się również wykorzystać wiedzę nabytą w ramach zajęć laboratoryjnych, projekt nie został w pełni „zdominowany" przez sieci neuronowe. Każdy z wykorzystanych modeli mógłby przejść proces dalszego trenowania, szczególnie na lepiej wyselekcjonowanych danych. Ponadto, w modułach w których pewne parametry zostały wybrane empirycznie, na pewno możliwe są pewne optymalizacje. Tworząc projekt, nie zważaliśmy na jego prędkość działania (obecnie, analiza pojedynczej klatki może zająć nawet 10s) oraz jego rozmiar (wraz ze wszystkimi zależnościami, projekt ma dużą objętość). Niemniej, znacząca większość z naszych założeń popełnionych podczas planowania architektury całej aplikacji jak i pojedynczych modułów została spełniona, co uznajemy na zdecydowany sukces.