Akademia Łomżyńska

Wydział nauk Informatyczno-Technologicznych





**Detekcja jadących aut**

**Wydziałowy projekt zespołowy**

**Zespół autorski**

Sebastian Kadłubowski

Damian Rutkowski

Rafał Żebrowski

**Prowadzący:**

dr inż Janusz Rafałko

Informatyka

Studia stacjonarne I stopnia, rok IV, semestr VII

Rok akademicki: 2024/2025

**1. Wprowadzenie**

Aplikacja umożliwia analizę nagrań wideo w celu wykrycia jadących pojazdów. Proces obejmuje identyfikację ruchu na filmie, zaznaczanie wykrytych pojazdów prostokątami i generowanie nowego nagrania z nałożonymi oznaczeniami. Wykorzystane technologie to Python, OpenCV, NumPy oraz wytrenowany model YOLO v4.

Model YOLO v4 (You Only Look Once, wersja 4) jest jednym z najpopularniejszych i wydajnych algorytmów do detekcji obiektów. Jego kluczowe zalety to:

* **Szybkość:** YOLO analizuje całą klatkę wideo jednocześnie, co pozwala na detekcję obiektów w czasie rzeczywistym.
* **Precyzja:** Model zapewnia wysoką dokładność w wykrywaniu i lokalizowaniu obiektów dzięki zaawansowanej architekturze sieci neuronowej.
* **Wszechstronność:** YOLO v4 może wykrywać różne typy obiektów, ale w tej aplikacji skonfigurowano go do detekcji pojazdów.

**2. Funkcjonalności**

1. **Wczytywanie nagrania wideo:** Aplikacja obsługuje nagrania w popularnych formatach wideo.
2. **Wykrywanie ruchu:** Identyfikacja obiektów w ruchu za pomocą analizy klatek wideo i modelu YOLO v4.
3. **Zaznaczanie pojazdów:** Wykryte obiekty są obrysowywane prostokątami, aby uwidocznić ich pozycję na nagraniu.
4. **Tworzenie nowego nagrania:** Aplikacja zapisuje zmodyfikowany film z naniesionymi oznaczeniami.

**3. Algorytm działania**

1. **Wczytanie wideo:**Aplikacja otwiera plik wideo i przygotowuje dane do analizy.
2. **Przetwarzanie klatek:**Każda klatka wideo jest analizowana w celu wykrycia ruchu.
3. **Detekcja pojazdów:**Model YOLO v4 identyfikuje obiekty w ruchu, odrzucając małe lub nieistotne elementy.
4. **Zaznaczanie pojazdów:**Na wykrytych pojazdach rysowane są prostokąty w celu wizualizacji wyników.
5. **Renderowanie wideo:**Zmienione klatki są zapisywane jako nowe nagranie wideo z nałożonymi oznaczeniami.
6. **Zakończenie przetwarzania:**Po przeanalizowaniu całego filmu plik wyjściowy jest zapisywany w katalogu użytkownika.

**4. Instrukcja obsługi**

1. **Przygotowanie pliku wideo:**Upewnij się, że nagranie, które chcesz przeanalizować, jest dostępne i kompatybilne z aplikacją.
2. **Uruchomienie aplikacji:**Otwórz skrypt w środowisku Python i uruchom go, wskazując plik wejściowy.
3. **Odbiór wyników:**Po zakończeniu analizy wideo zostanie wygenerowany nowy plik z wykrytymi pojazdami oznaczonymi prostokątami.

**5. Przypadki użycia**

* **Monitorowanie ruchu drogowego:**Aplikacja może być używana do analizy nagrań z monitoringu w celu monitorowania natężenia ruchu.
* **Bezpieczeństwo obszarów prywatnych:**Wykrywanie ruchu pojazdów w określonych strefach, takich jak parkingi czy drogi dojazdowe.

**6. Ograniczenia**

1. **Jakość nagrania:**Aplikacja najlepiej działa na nagraniach o dobrej rozdzielczości i w odpowiednim oświetleniu.
2. **Brak zaawansowanej klasyfikacji:**Obecna wersja skupia się wyłącznie na wykrywaniu ruchu, bez szczegółowej klasyfikacji pojazdów.
3. **Stabilność kamery:**Skuteczność algorytmu może być obniżona w przypadku drgających nagrań.
4. **Ilość klatek w nagraniu:**W naszej aplikacji użyliśmy wartości 30 klatek na sekundę, ponieważ zazwyczaj tyle mają nagrania z kamer. Wraz ze wzrostem klatek na sekundę prędkość renderowania drastycznie spada i potrzebny jest dobry sprzęt by móc to robić w czasie rzeczywistym.

Dodanie modelu YOLO v4 znacząco poprawia wydajność aplikacji, umożliwiając bardziej precyzyjne wykrywanie pojazdów w różnych warunkach nagrywania.

**7.Podział zadań na członków zespołu**

**Rafał Żebrowski** :

* Przetwarzanie danych i przygotowanie zbioru danych:
* Zgromadzenie i przygotowanie obrazów (zbiorów danych) do trenowania i testowania modelu.
* Oznaczanie samochodów na obrazach (tworzenie etykiet/bounding boxes)
* Wstępne przetwarzanie obrazów (zmiana rozdzielczości, konwersja kolorów, normalizacja itp.).

**Damian Rutkowski**:

* Implementacja detekcji w Pythonie przy użyciu OpenCV:
* Implementacja algorytmu detekcji samochodów z użyciem klasyfikatora kaskadowego Haar w OpenCV.
* Optymalizacja parametrów klasyfikatora w celu poprawy dokładności wykrywania.
* Tworzenie skryptu Python do wczytywania obrazów, uruchamiania detekcji i wyświetlania wyników.

**Sebastian Kadłubowski**:

* Testowanie, optymalizacja i dokumentacja techniczna:
* Testowanie dokładności detekcji na różnych obrazach (walidacja wyników).
* Optymalizacja czasu działania i wydajności
* Przygotowanie dokumentacji technicznej projektu, w tym opis działania kodu i analizy wyników detekcji.