Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Praca dyplomowa inżynierska

Wideo detekcja pojazdów samochodowych w ruchu drogowym

Video-based vehicle detection for road traffic applications

Autor: Rafał Prusak
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: dr inż. Zbigniew Marszałek

Kraków, 4 stycznia 2016

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

.....

Spis treści

1. Wstęp

Wraz z rozwojem motoryzacji i infrastruktury drogowej pojawiła się konieczność badania parametrów ruchu drogowego. Badania te mają na celu pozyskanie informacji o statystykach pojazdów, poziomie dostosowania stanu dróg do potrzeb użytkowników ruchu drogowego, wpływie ruchu drogowego na środowisko naturalne czy przyczynach wypadków samochodowych. Rezultaty badań są również ważnym czynnikiem, branym pod uwagę przy projektowaniu długofalowych planów rozbudowy infrastruktury, który pozwala określić najważniejsze cele i potrzeby rozwoju sieci dróg.

Naukowcy dysponują obecnie szeroką gamą narzędzi umożliwiających przeprowadzanie badań. Klasyczne metody, nazwane inwazyjnymi, pozwalają dokonać z dużą dokładnością pomiarów takich parametrów jak prędkość, masa czy ilość osi . Niestety, jak wskazuje ich nazwa, wymagają one często kosztownej i czasochłonnej ingerencji w nawierzchnię drogi. Z drugiej strony, w wyniku szybkiego wzrostu mocy obliczeniowej komputerów, pojawiły się metody bezinwazyjne, bazujące na wiązkach laserowych czy ultradźwiękach.

Ważne miejsce wśród metod bezinwazyjnych zajmują systemy wizyjne. Znalazły one wykorzystanie w wielu aspektach, takich jak: sprawdzanie obecności pieszych i pojazdów na pasach/skrzyżowaniach, detekcja przeciążonych pojazdów, punktowe i odcinkowe pomiary prędkości czy systemy poboru opłat.

Cel i zakres pracy

W pracy przedstawiono zagadnienia związane z wykrawaniem i klasyfikacją pojazdów na podstawie obrazu wideo. Na potrzeby pracy napisano program, który z próbek wideo zawierających nagrania ruchu pojazdów na odcinku drogi jednokierunkowej uzyskuje informacje o przejeżdżających pojazdach.

W rozdziale drugim omówiono sposób przeprowadzania pomiarów ruchu drogowego w Polsce. Przedstawiono najpopularniejsze detektory ruchu pojazdów. Przeanalizowano zalety i wady ich użycia.

Rozdział trzeci zawiera opis kilku popularnych aplikacji komercyjnych, w których zastosowanie znalazło przetwarzanie obrazu.

Rozdział czwarty to prezentacja użytych przez autora narzędzi: jezyka programowania, zewnętrznych bibliotek oraz programów użytych w czasi e pisania programu.

W piątym rozdziale znajdują się informacje o interfejsie i możliwościach programu.

Rozdział szósty opisuje zaimplementowany algorytm, prezentując dokładnie jego etapy na przykładach.

Rozdział siódmy zawiera analizę wyników działania programu wraz z porównaniem z pomiarem przeprowadzonym przy pomocy pętli indukcyjnych.

W rozdziale 8 znalazło się podsumowanie pracy.

2. Metody pomiarów parametrów ruchu drogowego

W wyniku rozwoju motoryzacji oraz rozbudowy infrastruktury drogowej pojawiła się konieczność zbierania i analizy informacji o ruchu pojazdów.

Generalny Pomiar Ruchu Drogowego

W Polsce od roku 1965, co 5 lat, dokonywany jest Generalny Pomiar Ruchu Drogowego(GPRD)[5][6]. Ma on na celu pozyskanie charakterystyk ruchu drogowego na wszystkich odcinakach dróg krajowych i wojewódzkich. Uzyskiwane w wyniku GPRD pomiary są wykorzystywane przy planowaniu rozbudowy infrastruktury drogowej, organizacji ruchu, utrzymaniu dróg, określaniu przyczyn wypadkowości czy badaniu wpływu ruchu na komfort życia mieszkańców i środowisko naturalne.

W czasie pomiarów pojazdy są klasyfikowane do jednej spośród 8 kategorii:

1. motocykle,
2. samochody osobowe,
3. lekkie samochody ciężarowe,
4. samochody ciężarowe bez przyczepy,
5. samochody ciężarowe z przyczepami/ naczepami,
6. autobusy,
7. ciągniki rolnicze,
8. rowery.

Metodologię dokonywania pomiarów, terminy pomiarów, zadania obserwatorów, etc reguluje zarządzenie nr 38 Generalnego Dyrektora Dróg Krajowych i Autostrad z dnia 1 września 2014 r.

Parametry ruchu drogowego

Podczas dokonywania pomiarów dąży się do pozyskania zarówno indywidualnych informacji o pojazdach jak również ogólnej charakterystyki ruchu. Wyróżnia się w tym celu szereg parametrów:

- Prędkość - chwilowa, jazdy, podróży, miarodajna.
- Czas podróży - czas pokonania odcinka drogi.
- Natężenie ruchu - liczba pojazdów na jednostkę czasu.
- Gęstość ruchu - liczba pojazdów znajdujących się na odcinku drogi na jednostkę czasu.
- Zajętość drogi - procent drogi zajęty przez pojazdy.
- Bezpieczeństwo ruchu.
- Wpływ na środowisko.
- Zużycie paliwa.

Pomiarów ruchu drogowego dokonuję się w punkcie, na krótki odcinku drogi, w zdefiniowanym obszarze, na całej trasie lub za pomocą pojazdu/pojazdów próbnych poruszających się po drodze. Rozróżnia się 4 typy ruchu pojazdów: jazda swobodna, jazda przy maksymalnym przepływie, jazda w warunkach dużej gęstości oraz zator.

Indywidualny pojazd można opisać szeregiem wartości: klasą pojazdu, numerem rejestracyjnym, liczbą osi, odległościami między osiami, długością, szerokością , wysokością, prędkością czy naciskiem na jezdnię. Stosuję się także opis strefowy, środowiskowy, pogodowy i opis jezdni.

Detektory

Detektor jest to urządzenie składające się z czujnika, układu kondycjonowania, dyskryminatora oraz elementu wykonawczego. Jego celem jest wykrycie obecności pojazdu w punkcie bądź strefie pomiaru. Detektor powinien być urządzenie prostym w obsłudze i montażu, bezawaryjnym, pozwalającym na łatwy odczyt informacji. Stosowane są dwa rodzaje detektorów: bezinwazyjne i wbudowane(instalowane w nawierzchni)[4].

Klasyfikacja detektorów wbudowanych:

- Linowy - drut stalowy lub linka nylonowa zamontowana jest na wysokości kilku minimetrów na jezdnią. Nacisk przejeżdżającego pojazdu powoduje naciąg linki i uruchomienie zestyku, który generuje impuls. Ten rodzaj detektory może generować fałszywe impulsy, wymaga regulacji i poprawy naciągu linki, ma ograniczoną maksymalna prędkość pomiarów. Jego zaletą jest prosta budowa.

- Taśmowy - dwie taśmy z gumowymi wkładkami, wypełnione gazem elektrycznie obojętnym. Nacisk pojazdu powoduje zetknięcie się ze sobą taśm. Do działania wymaga specjalnego osprzętu. Występują niekorzystne drgania styków. Montaż wymaga interwencji w nawierzchnię.

- Hydrauliczny - nacisk pojazdu na wąż wypełniony olejem powoduje podniesienie się poziomu w kapilarze i uruchomienie zestyku. Pojazdy o dużej masie mogą uszkodzić ten typ detektora.

- Pneumatyczny - działa na podobnej zasadzie co hydrauliczny.

- Reluktancyjny - nacisk samochodu powoduje zwierzenie się szczeliny. Następuje zmiana reluktancji obwodu magnetycznego w wyniku czego zmienia się częstotliwość sygnału wyjściowego.

- Rezystancyjny - nacisk pojazdu powoduje nieliniową zmianę rezystancji czujnika. Detektor ma ograniczoną prędkość pomiaru do max 80 km/h, więc jest rzadko wykorzystywany.

- Pojemnościowy - dwie elektrody oddzielone dielektrykiem. Nacisk pojazdu powoduje zmniejszenie szerokości kondensatora i zmianę jego pojemności. Czujniki te mają kiepską odporność na uszkodzenia mechaniczne i zasolenie. Ich montaż wymaga kosztownej interwencji w nawierzchnię.

- Piezoelektryczny - jest to taśma wykonana z materiału piezoelektrycznego lub kabel koncentryczny z umieszczoną w środku elektrodą. Czujniki są wrażliwe na zmiany temperatury, której wzrost powoduje pogorszenie jakości pomiaru. Mierzony może być jedynie nacisk zmienny w czasie, co ogranicza prędkość pomiaru do zakresu 20-110 km/h. Ten sposób pomiaru jest szeroko stosowany z powodu dużej czułości, szerokiemu zakresowi temperatury oraz niskiej cenie wykonania.

- Kwarcowe - działa na podobnej zasadzie co piezoelektryczny: nacisk samochodu na aluminiowy profil powoduje nacisk na element kwarcowy. Jest często stosowany w systemach ważenia pojazdów. W przeciwieństwie do poprzedniego typu, cechuje się odpornością na oddziaływanie sił podłużnych i większą wrażliwością na oddziaływanie sił poprzecznych. Czujnik może pracować w szerokim zakresie temperatur i prędkości. Jest odporny na zakłócenia temperaturowe i elektromagnetyczne. Jego trwałość osiąga nawet 10 lat.

- Tensometryczny - używany jest zarówno w pomiarach nacisków osi pojazdów statycznych i dynamicznych. Ma postać platformy na której zamontowywane są tensometry, najczęściej w układzie pełnego mostka.Charakteryzuje się dużą dokładnością pomiaru(błędy rzędu 2% przy pomiarze statycznym). Może mierzyć pojazdy o prędkości do 200 km/h. Jego trwałość sięga 10 lat.

- Indukcyjny pętlowy - jest najbardziej rozpowszechnionym typem detektora. Może być stosowany w pomiarach prędkości, wysokości zawieszenia, ilości osi, detekcji incydentów. Uzyskiwany sygnał jest nazwany profilem magnetycznym pojazdu. Czujnik jest odporny na czynniki zewnętrzne, ma dużą trwałość, jego instalacja ma jezdni jest prosta i mało kosztowna.

- Magnetyczne - reaguje na zmiany pola magnetycznego. Jest bardziej czuły na zakłócenia niż pętlowy, ale jego instalacja w nawierzchni jest znacznie prostsza.

Drugim rodzajem detektorów są detektory nieinwazyjne. Spośród nich wyróżniamy:

- Laserowe aktywne - zbudowane są z nadajnika promieniowania, układu optycznego oraz odbiornika promieniowania. Mogą być wykorzystane do pomiaru prędkości, wysokości czy, przy odpowiedniej częstotliwości próbkowania, budowaniu profilu samochodu. Czujnik tej klasy mogą pracować w szerokim zakresie temperatur, wykazują odporność na warunki atmosferyczne oraz są niezależne od zewnętrznego oświetlenia.

- Laserowa pasywne - reagują na zmiany w promieniowaniu podczerwonym wywołane ruchem obiektów w obszarze detekcji. Detektor ren jest bardzo łatwy w instalacji - nie wymaga żadnej interwencji w ruch drogowy. Detektory tego typu nie oddziałują na siebie nawzajem.

- Akustyczne - składają się z matrycy mikrofonów, które zbierają dźwięki przejeżdżających samochodów: opony, silnik, ruch powietrza.

- Ultradźwiękowe - generują falę dźwiękową niesłyszalną dla człowieka, która służy wykrywaniu obecności obiektów w sferze detekcji.

- Mikrofalowe - działają na zasadzie radaru, generują fale wysokiej częstotliwości. Mogą pracować praktycznie w każdych warunkach.

- Kamery z systemami przetwarzania obrazu.

3. Systemy wizyjne w zastosowaniach komercyjnych

Rozdział ten stanowi przegląd dostępnych na rynku systemów pomiaru ruchu drogowego, w których zastosowanie znalazły algorytmy wizyjne.

Charakterystyka systemów wizyjnych

Systemy VIP(Video Image Processing) są coraz częściej stosowane przy pomiarach parametrów ruchu drogowego. Początkowo służyły one jedynie wizualizacji pomiarów bądź jako pomoc dla operatora w procesie weryfikacji poprawności pomiarów dokonywanych „standardowymi" metodami. Rozwój mocy obliczeniowych komputerów sprawił, że pojawiła się możliwość automatycznej analizy obrazu i wydobywania z niego użytecznych informacji, takich jak:

- obecność pojazdów na skrzyżowaniach, drogach, tunelach, etc,
- kierunek ruchu,
- klasyfikacja pojazdów,
- pomiar natężenia ruchu,
- informacje o obecności pieszych i rowerzystów,
- detekcja incydentów drogowych.

Obecnie wykorzystywane aplikacje wizyjne mogą działać zarówno jako samodzielne metody pomiaru lub stanowić część systemów kontroli wykroczeń, pomiaru prędkości czy ważenia samochodów.

Efektywność systemów wizyjnych szacuje się na około 90%. Posiadają one wiele zalet:

- są całkowicie bezinwazyjne,
- mimo wysokiej ceny budowy i instalacji systemu, jego dalsza eksploatacja odbywa się niskim kosztem,
- mają szerokie możliwości konfiguracji i dostosowywania strefy detekcji,

- mogą być zintegrowane z systemami pomiaru wbudowanymi w jezdnię,
- w zaawansowanych aplikacjach kamera może dokonywać auto-pozycjonowania w celu kompensacji zakłóceń wynikających ze zmiany położenia słońca czy spadku jakości oświetlenia,
- istnieje możliwość streamowania obrazu przez sieć Internet.

Największymi wadami metod wizyjnych są:

- niekorzystny wpływ warunków atmosferycznych: deszcz, śnieg, mgła, zachmurzenie,
- zjawiska związane ze wzajemnym mijaniem i zasłanianiem się pojazdów,
- zabrudzenia odkładające się na układzie optycznym.

Ważnym czynnikiem mającym wpływ na działanie systemu jest sposób montażu kamery względem nadzorowanego pasa ruchu. Najczęściej umieszcza się kamerę wysoko nad pasem ruchu. Umieszczenie punktu pomiaru z boku drogi zmniejsza skuteczność pomiaru o około 10%.

NeuroSoft

Rozwiązania wrocławskiej firmy NeuroSoft są jednymi z najpopularniejszych w kategorii systemów pomiaru i zarządzania ruchem drogowym[9]. Firma ta oferuje system o nazwie „Neuro Car" oraz bardziej rozbudowany „Neuro Car 2.0". Programy te dokonują wideo rejestracji pojazdów, odczytów tablic rejestracyjnych i wstępnej selekcji danych. System jest dostarczany w postaci terminala zbudowanego z kamery przemysłowej, komputera, karty WLAN, modemu GPRS i zalicza. Instalacja systemu jest bardzo prosta: polega na ustawieniu kamery, podpięciu zasilania i konfiguracji parametrów.

Zaletami systemu są:

- niska cena,
- możliwość integracji z innymi systemami
- wysoka skuteczność rozpoznawania,
- praca w trybie rzeczywistym,
- odporność na warunki zewnętrzne,
- rozpoznawanie marki, modelu, koloru,
- zastosowanie kamer z dużą dynamiką jasności,
- zdalny dostępu do danych.

W systemie Neuro Car zastosowano procesory wielordzeniowe, DSP oraz system operacyjny Linux, umożliwiając w ten sposób pomiar w stałej ilości 25 klatek na sekundę. Rozpoznanie, klasyfikacja i generacja opisu trwa nie mniej niż 100 ms. Dodatkowo program generuje od 3 do 8 zdjęć dla zidentyfikowanego pojazdu. Producent określa skuteczność detekcji na 99%.

W skład systemu wchodzi wiele dodatkowych aplikacji:

- Velocity - pomiar prędkości chwilowej na podstawie serii zdjęć.
- Dangerous goods - identyfikacja samochodów przewożących materiały niebezpieczne.
- Classification - klasyfikacja pojazdów wg marki i typu(jednośladowy, osobowy, ciężarowy).
- PDA - aplikacja na platformy mobilne do rozpoznawania numerów tablic rejestracyjnych.
- SectionControl - pomiar prędkości średniej na odcinku trasy.
- RedLight - detekcja przejazdu na czerwonym świetle.
- City - tworzenie płatnych stref w miastach i pobieranie opłat za przebywanie w nich.
- Parking - monitorowanie parkingu, rozliczenie opłat za parkowanie.
- Monitoring - uniwersalna aplikacja do obserwacji przepływu pojazdów.
- Base - itegracja z basami mySQL lub postgrSQl.
- Monitor - oprogramowanie do wizualizacji wyników pomiaru.
- ScaleGate - system rozpoznawania tablic i ważenia pojazdów, stosowany w hurtowniach i magazynach w celu pozyskiwania statystyk o tonażu materiałów przychodzący i opuszczających teren zakładu.
- Neurocar WIM&MIM - automatyczna detekcja pojazdów ponadgabarytowych(przeciążonych oraz za wysokich), których obecność powoduje pogorszenie właściwości nawierzchni oraz stwarza zagrożenie na mostach, pod wiaduktami i w tunelach.

Cat Traffic

Poznańska firma Cat Traffic zajmuje się systemami stanowiącymi połączenie metod wizyjnych z wykorzystaniem płyt warzących[3]. Firma wdrożyła kilkanaście stacji warzących na terenie Niemiec, Polski, Austrii i Szwajcarii. W Polsce CatTraffic wdrożyło na autostradzie A4 systemy pomiaru wagi i badania statystki ruchu pojazdów, oraz system czasu przejazdu na „Zakopiance".

Rozwiązania oferowane przez firmę:

- System pomiaru prędkości EasyCount oprócz pomiaru prędkości może badać również kierunek jazdy, odległości między pojazdami, i długość pojazdu. Przy użyciu dodatkowego osprzętu , EasyCount umożliwia pomiar na dwóch pasach ruchu. System ma do dyspozycji pamięć, która oferuje miejsce na dane o 800 000 przejazdach. Można dokonywać jego konfiguracji za pomocą modułu GSM. Zasilanie zapewniają akumulatory lub baterie słoneczne.
- Easy Count-Bike- oferuje podobne funkcje co Easy Count, ale w odniesieniu do rowerzystów.
- Licznik AVC - to pętlowy system pomiaru ruchu, umożliwiający zapis, transfer i analizę danych.
- V-REX 1000 - mobilny radar do odczytu tablic i badania struktury ruchu.
- VIM-VIVER - automatyczna stacja do ważenia pojazdów w ruchu, wraz z pozyskiwanie profilu pojazdu i numeru tablic rejestracyjnych.
- SAW - przenośna waga, umożliwiająca pomiar pojazdów o nacisku do 15 ton. Wykorzystywana przez Państwową Inspekcję Transportu Drogowego.
- ANPR - automatyczne rozpoznawanie tablic, z maksymalnie 8 kamer jednocześnie.

4. Zastosowane narzędzia.

Poniższy rozdział zawiera krótki opis języka programowania, w którym została zaimplementowana praca inżynierska, zastosowanych zewnętrznych bibliotek oraz innych narzędzi użytych przy wykonaniu programu.

Python

Algorytm wideo detekcji pojazdów został zaimplementowany w języku programowania Python, w wersji 3.4[11][10]. Język ten posiada wiele korzystnych cech, które miały istotny wypływ na proces powstawania programu:

- Jest językiem obiektowym wysokiego poziomu. Obiektowość kodu, w przeciwieństwie do takich popularnych języków jak Java czy C#, nie jest jednak odgórnie wymuszana przez składnię języka. Powoduje to, że w Pythonie można pisać proste liniowe skrypty, służące np. do testowania programu.
- Gramatyka języka jest bardzo prosta, podobna do składni języka angielskiego, przez co kod jest jasny i zrozumiały. Dodatkowo język ten wymusza na programiście stosowanie standardu PEP8, co powoduje jednolity i przejrzysty styl kodu w powstającym programie.
- Posiada bogatą bibliotekę standardową zawierającą moduły do programowania sieciowego, pracy z plikami html, xml, json, tworzenia interfejsów gui czy przetwarzania wielowątkowego.
- Jest językiem interpretowanym, więc programy wykonane w tym języku są niezależne od platformy i mogą być łatwo przenoszone na inne systemy operacyjne bądź platformy sprzętowe(komputer PC, systemy embeded).
- Ogromna popularność języka Python i szeroka społeczność skupiona wokół niego, skutkuje w ogramnej bazie gotowych rozwiązań, porad i tutoriali dostępnych w sieci.

OpenCV

OpenCV jest biblioteką „open source” służącą do cyfrowego przetwarzania obrazu wizyjnego. Jest ona bezpośrednio zaimplementowana w języku C/C++. Dostępna jest na wszyst-

kich systemach operacyjnych stacjonarnych i mobilnych: Windows, Linux, MacOS, Android, iOS. Biblioteka zapewnia API dla wielu języków, takich jak: Python, Ruby, Java, Matlab. Została napisana z myślą o programach potrzebujących najwyższej wydajności i wykorzystaniu w aplikacjach czasu rzeczywistego. Najnowsze wydanie, w wersji 3.0, przynosi optymalizację biblioteki na wielu poziomach: algorytmów, wykorzystania wątków oraz rdzeni czy rozkazów dla CPU. Najwięksi producenci sprzętu komputerowego, tacy jak Intel czy Nvidia, także wzięli udział w tworzeniu biblioteki, dostarczając własne niskopoziomowe programy do wykonywania operacji na GPU.

Biblioteka jest podzielona na kilkanaście modułów, dostępnych w postaci bibliotek statycznych lub dynamicznych. Poniżej zostały omówione najważniejsze moduły, które znalazły zastosowanie w implementacji pracy inżynierskiej:

- core - podstawowe funkcje do działań na tablicach wielowymiarowych,
- highgui - obsługa graficznego interfejsu użytkownika, wyświetlanie zdjęć i wideo ,
- imgproc - funkcje do przetwarzania obrazu: filtracja, transformacje geometryczne, konwersje między przestrzeniami barw,
- imgcodecs - interfejs do odczytu i zapisu zdjęć,
- videoio - przechwytywanie obrazu wideo,
- video - analiza obrazu wideo(śledzenie ruchu, wyodrębnianie tła),
- features2D - znajdywanie krawędzi, dopasowywanie szablonów,
- objectdect - dopasowywanie i wykrywanie obiektów.

OpenCv znalazło wiele zastosowań w aplikacjach przemysłowych, takich jak: inspekcja gotowych produktów w fabrykach, medycyna, bezpieczeństwo, systemy wizyjne „wielowymiarowe", robotyka.

NumPy

NumPy jest zewnętrzną biblioteką „open source", służącą do przeprowadzania obliczeń na n-wymiarowych tablicach. Jest ona podobna do środowiska Matlab. Podstawowa typ tablicy, używany przez bibliotekę znacząco różni się od tablicy(listy) samego Pythona: lista jest kontenerem heterogonicznym, mogącym dynamicznie zmieniać swój rozmiar, natomiast „ndarray" jest tablicą homogeniczną o stałym rozmiarze.Rozwiązanie takie zapewnia wysoką wydajność obliczeń i możliwość bezpośredniej integracji z plikami binarnymi biblioteki napisanymi w języku C/C++.

SciPy

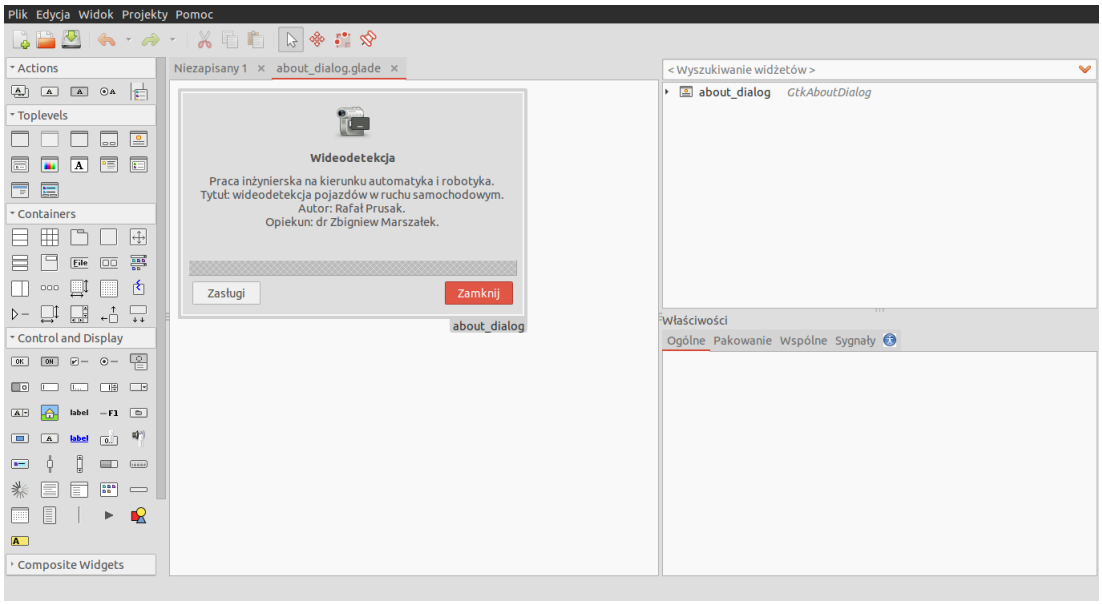
Biblioteka SciPy(skrót od „Scientific Python”) jest biblioteką „open-source” zawierającą funkcje do rozwiązywania zadań optymalizacji, algebry liniowej, przetwarzania sygnałów czy rozwiązywania równań różniczkowych. Biblioteka pozwala zrealizować większość operacji dostępnych w środowisku Matlab.

PyGtk + Glade

W celu zaprojektowania i stworzenia interfejsu graficznego użytkownika zostało wykorzystane PyGtk oraz program Glade.

PyGtk jest nakładką na bibliotekę Gtk+, pozwalającą na tworzenie graficznego interfejsu użytkownika przy użyciu języka Python. Znalazła ona zastosowanie w wielu popularnych aplikacjach: BitTorrent, Gedit, GIMP. Jest częścią linuksowego środowiska graficznego GNOME.

Glade jest programem to projektowania interfejsu graficznego użytkownika z wykorzystaniem „widgetów” z biblioteki Gtk. Stworzony projekt okna interfejsu jest zapisywany do pliku xml. Spreparowany w ten sposób plik, może być wykorzystany do zbudowania okna za pomocą klasy GtkBuilder.

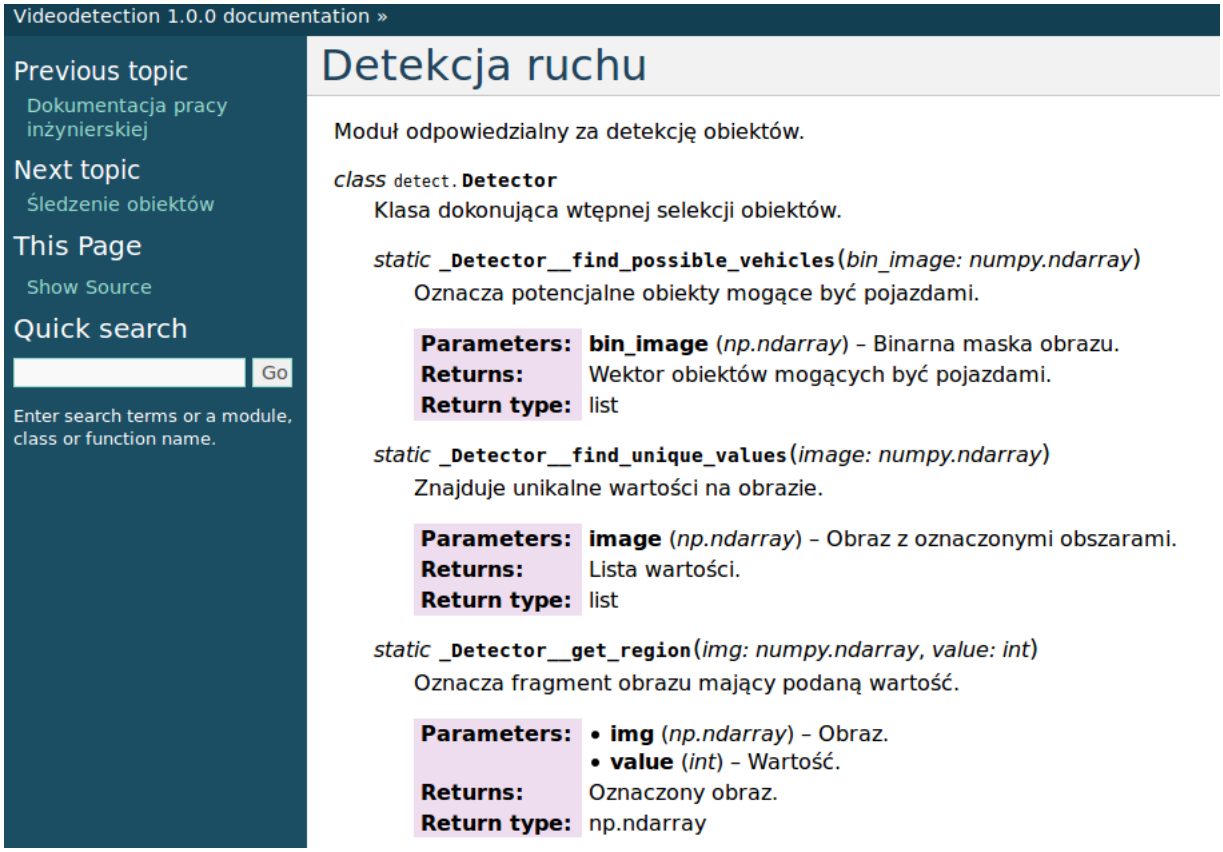


Rysunek 4.1. Widok okna programu Glade

Sphinx

Sphinx jest to narzędzie służące do automatycznej generacji dokumentacji programu na podstawie komentarzy zawartych w kodzie. Program ten pozwala na tworzenie dokumentacji programów napisanych w Pythonie bądź C++. Aby dokumentacja mogła zostać wyge-

nerowana, komentarze muszą być zgodne ze standardem „reStructuredText”. Dokumentacja może być wygenerowana w postaci tekstu, htmla, w formacie pdf czy Latex.



Rysunek 4.2. Wygenerowana dokumentacja.

Pycharm

Pycharm jest środowiskiem deweloperskim dla języka Python, tworzonym przez firmę JetBrains. Zapewnia ono szereg przydatnych funkcji, takich jak: podpowiadanie składni, detekcję błędów, proponowanie poprawek i rozbudowany debuger. Działa zgodnie, ze wszystkimi popularnymi implementacjami języka Python: CPython, IronPython, Jython. Środowisko oferuje wsparcie dla innych technologi, np. Html5, Css3, JavaScript, Angular.js. Za pomocą Pycharma programista może korzystać z frameworków webowych takich jak Django czy Flask. Środowisko wspiera również systemy kontroli wersji(Git, SVN), posiada wsparcie dla narzędzi do pracy z bazami SQL oraz umożliwia zdalne edytowanie kodu w oparciu o protokół SSH. Firma JetBrains udostępnia 3 wersje IDE: profesjonalną(płatną), edukacyjną(dla studentów) oraz „społecznościową”.

5. Opis programu

Poniższy rozdział opisuje interfejs i sposób działania zaimplementowanego programu.

Instalacja

Poniżej został opisany sposób instalacji programu oraz wymaganych bibliotek na systemie Linux.

1. Wchodzimy na stronę, i pobieramy najnowszą wersję programu:

```
https://github.com/artven/videodetection/releases/latest
```

2. Przechodzimy do folderu zawierające pobrany plik:

```
cd ~/Pobrane
```

3. Rozpakowujemy i kopiujemy program do katalogu domowego:

```
unzip videodetection.v1.0.zip
mkdir ~/videodetection
cp -r 'videodetection v1.0'/* ~/videodetection/
cd ~/videodetection
```

4. Instalujemy zależności dla biblioteki OpenCV:

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-upgrade && sudo
    apt-get autoremove
sudo apt-get install build-essential cmake qt5-default libvtk6-dev zlib1g-dev
    libjpeg-dev libwebp-dev libpng-dev libtiff5-dev libjasper-dev libopenexr-dev
    libgdal-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev
    libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev yasm libopencore-
    amrnb-dev libopencore-amrwb-dev libv4l-dev libxine2-dev libtbb-dev libeigen3
    -dev python-dev python-tk python-numpy python3-dev python3-tk python3-numpy
    ant default-jdk doxygen
```

5. Pobieramy bibliotekę z oficjalnej strony:

```
cd /tmp
wget https://github.com/Itseez/opencv/archive/3.1.0.zip
unzip 3.1.0.zip
cd opencv-3.1.0/
```

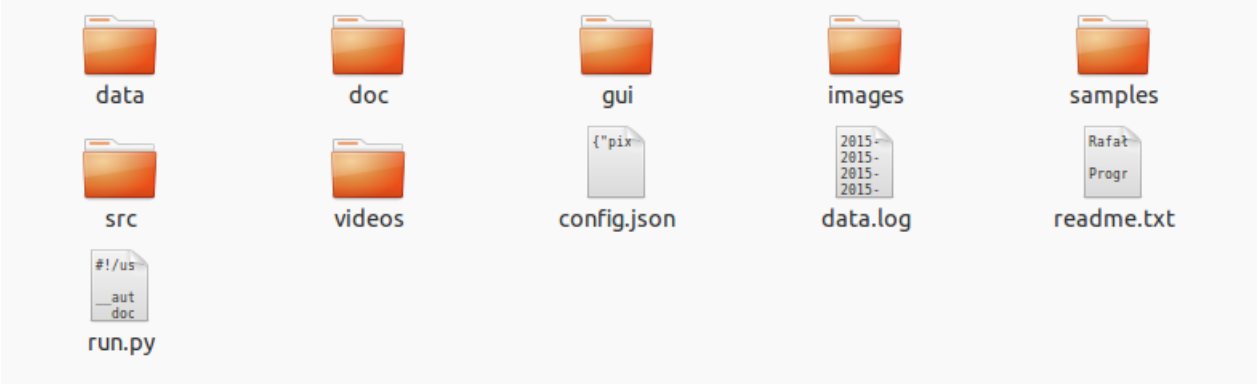
6. Kompilujemy i instalujemy OpenCV:

```
mkdir build
cd build
cmake -DWITH_QT=ON -DWITH_OPENGL=ON -DFORCE_VTK=ON -DWITH_TBB=ON -DWITH_GDAL=ON
    -DWITH_XINE=ON -DBUILD_EXAMPLES=ON ..
make -j4
sudo make install
sudo ldconfig
```

7. Instalujemy sklearn:

```
pip3 install -U scikit-learn
```

Struktura programu



Rysunek 5.1. Widok struktury programu

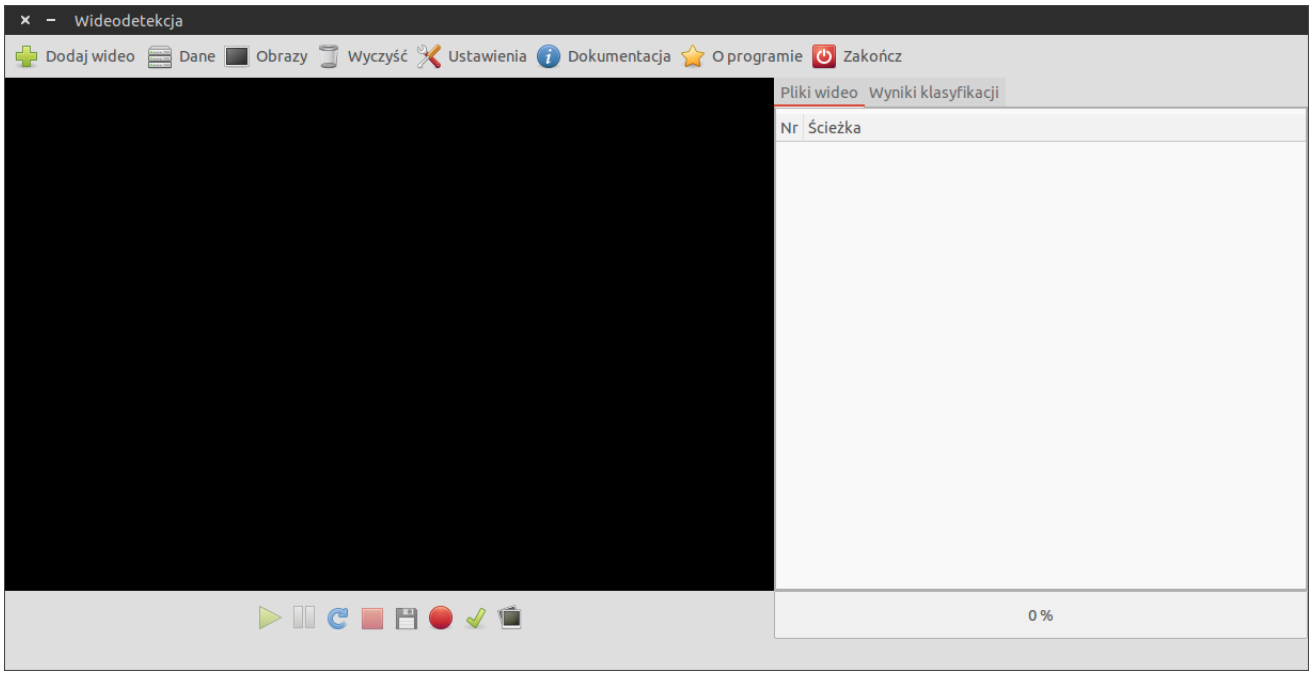
Program został podzielony na następujące pliki oraz foldery:

- data - folder do którego program zapisuje dane o wykrytych pojazdach do baz danych SQLite,
- doc - dokumentacja programu w formacie .html wygenerowana przy użyciu narzędzia Sphinx,
- gui - pliki w formacie .glade używane do wyświetlania interfejsu graficznego użytkownika,
- images - zdjęcia wykrytych samochodów,
- samples - przykładowe próbki nagrań ruchu samochodów,
- src - pliki źródłowe programu,
- videos - zapisane przez program wyniki analizy w postaci plików wideo,

- config.json - plik konfiguracyjny programu, zawiera parametry algorytmów oraz opcje dotyczące prezentacji wyników programu,
- data.log - plik gdzie są odkładane komunikaty diagnostyczne programu,
- readme.txt - plik tekstowy zawierający informacje o programie,
- run.py - skrypt uruchamiający program.

Interfejs

Po uruchomieniu programu skryptem z pliku run.py pojawia się główne okno programu.

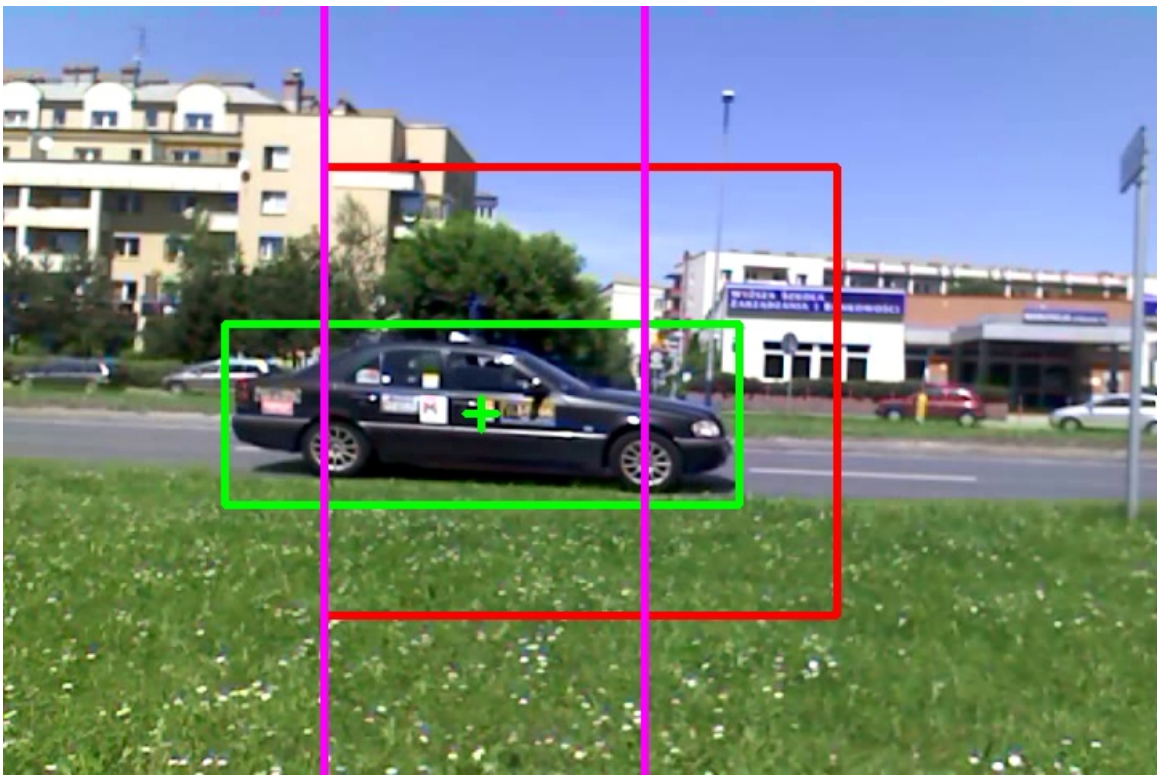


Rysunek 5.2. Główne okno programu

W skład głównego okna wchodzi górne menu, obszar odtwarzania pliku wideo, lista przetwarzanych plików, lista wyników, menu odtwarzania i pasek informujący o postępie przetwarzania.

Opis oznaczeń na obrazie

Podczas przetwarzania plików wideo odbywa się równoczesne ich wyświetlanie z naniesionymi przez program oznaczeniami.



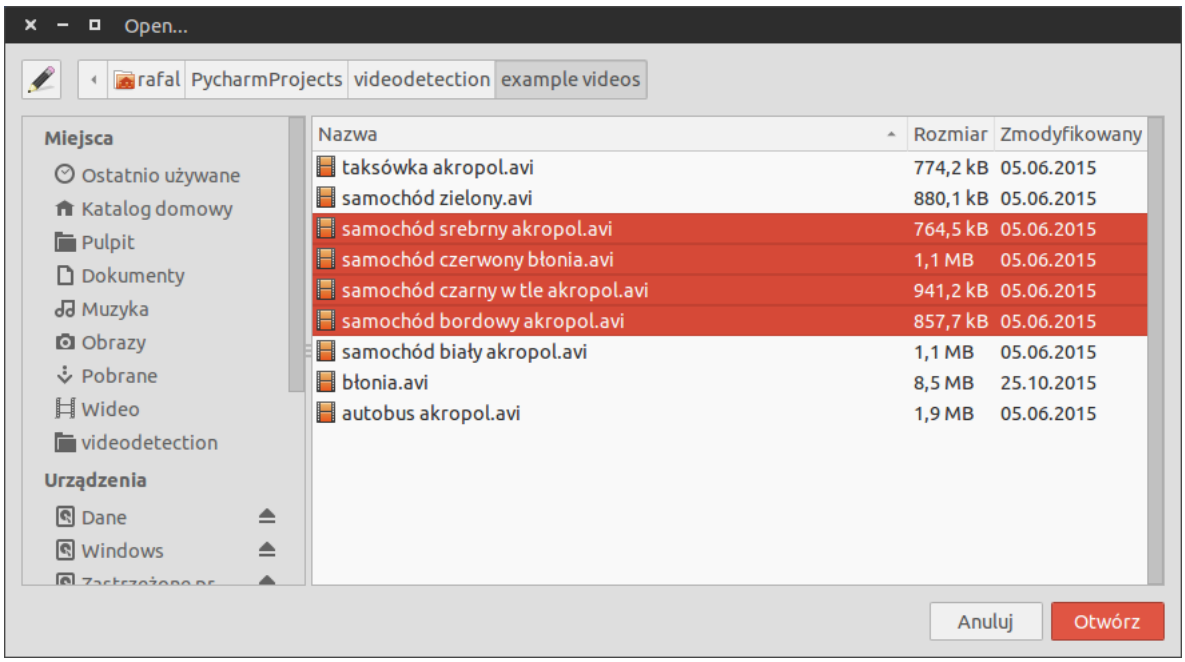
Rysunek 5.3. Oznaczenia na obrazie

- Pojazdy są oznaczane zielonym prostokątem. Prostokąt ten otacza rejon obiektu wykrytego po wyodrębnieniu tła. Zielony krzyżyk jest umiejscowiony w środku ciężkości obiektu.
- Czerwony prostokąt stanowi obszar „czułości" kamery. Wszystkie obiekty, których środki ciężkości znajdują się po za tym obszarem są ignorowane przez program.
- Dwie różowe pionowe linie służą do oznaczenia odcinka, którego długość stanowi podstawę do wyliczenia przelicznika pikseli na obrazie na metry. Pozwala to określać prędkość i wymiary samochodów.

Wyświetlanie oznaczeń można włączać/wyłączać w opcjach programu, które zostały opisane w dalszej części pracy.

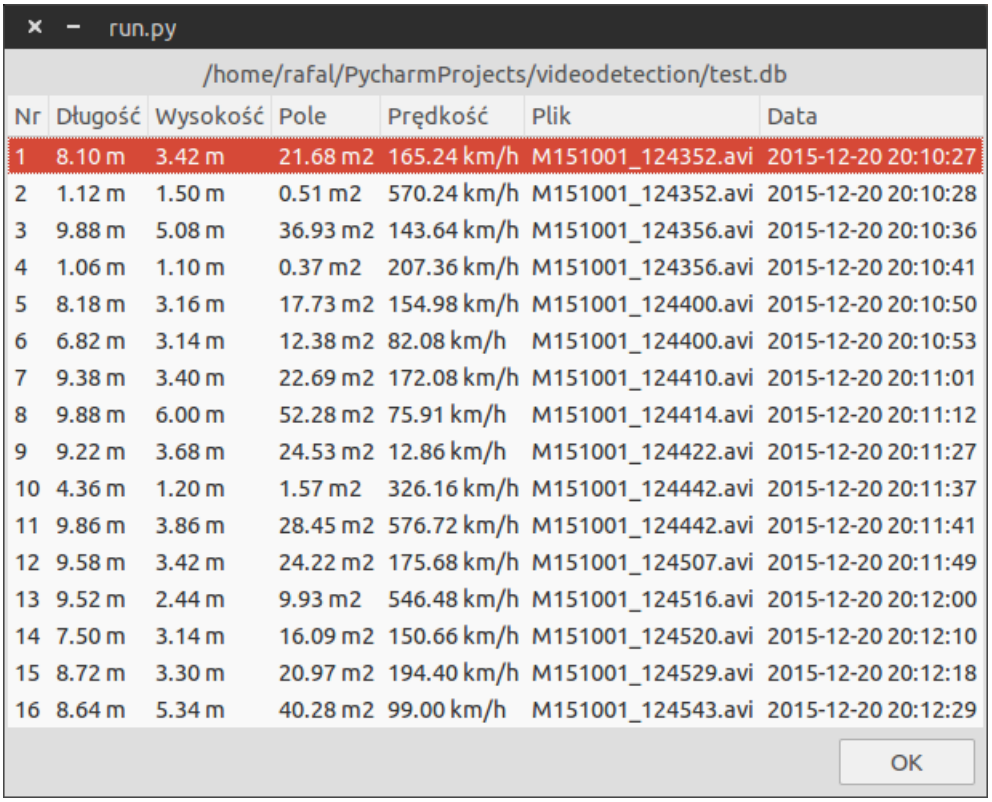
Funkcje elementów głównego menu:

- „Dodaj wideo" - otwiera poniższe okno dialogowe, umożliwiające wybór jednego bądź większej ilości plików wideo do analizy. Wybrane pliki są dodawane do listy po prawej stronie. Użytkownik może usuwać pliki z listy po przez kliknięcie prawym klawiszem myszy na nazwie pliku. Skrót klawiszowy Ctrl+q.



Rysunek 5.4. Okno wyboru plików

- „Dane” - umożliwia wybranie bazy danych i podejrzenie jej zawartości. W bazie danych wygenerowanej przez program znajduje się jedna tabela „cars”, w której znajdują się pola „id”, „width”, „height”, „area”, „speed”, „file” i „date”. Pole „file” umożliwia identyfikację przetwarzanego pliku w celu weryfikacji wyników. Pole „date” została w umieszczone w celu przyszłej implementacji przetwarzania obrazu w czasie rzeczywistym. Skrót klawiszowy Ctrl+w.



Rysunek 5.5. Podgląd bazy danych

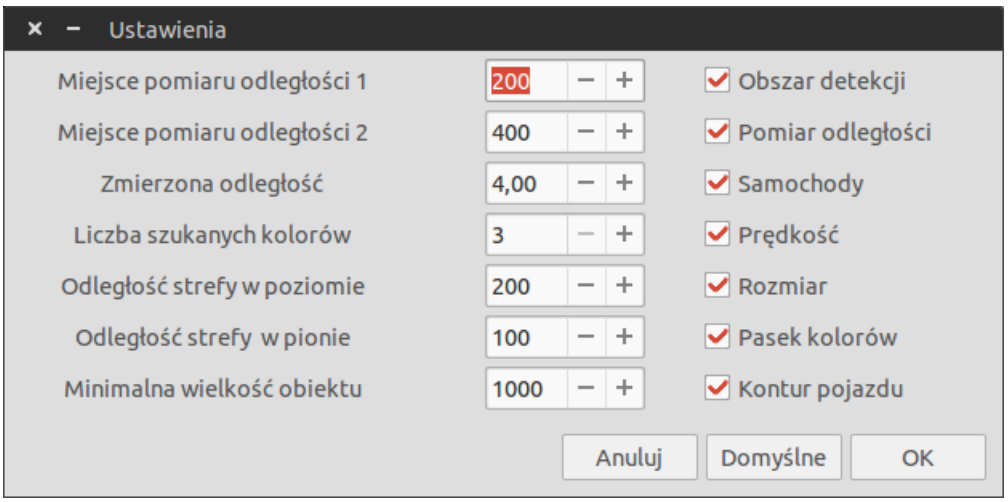
- „Obrazy” - włącza przeglądarkę zapisanych przez program zdjęć pojazdów. Skrót klawiszowy Ctrl+e.

Rysunek 5.6. Widok przeglądania obrazów.

Przeglądanie obrazów odbywa się po przez wywołanie domyślnego programu do otwierania obrazów na systemie Linux(na dystrybucji Ubuntu jest to program „eog”). Na wygenerowanym przez program obrazie wynikowym został umieszczony pojazd wjeżdżający w obszar detekcji(w górnej części) oraz pojazd opuszczający ten obszar(na dole obrazu). Dodatkowo program wypisuje na obrazie informację, w postaci paska kolorów w lewym górnym rogu oraz podpisu u dołu zdjęcia.

- „Wyczyść” - usuwa zdjęcia, filmy, bazy danych oraz wyniki klasyfikacji. Przed wykonaniem operacji pojawia się okno dialogowe z prośbą o potwierdzenie usunięcia danych. Pozwala to uniknąć przypadkowego usunięcia danych. Skrót klawiszowy Ctrl+r.

- „Ustawienia” -pokazuje okno ustawień programu. Opcje programu zostały opisane w następnym podrozdziale programu. Skrót klawiszowy Ctrl+t.

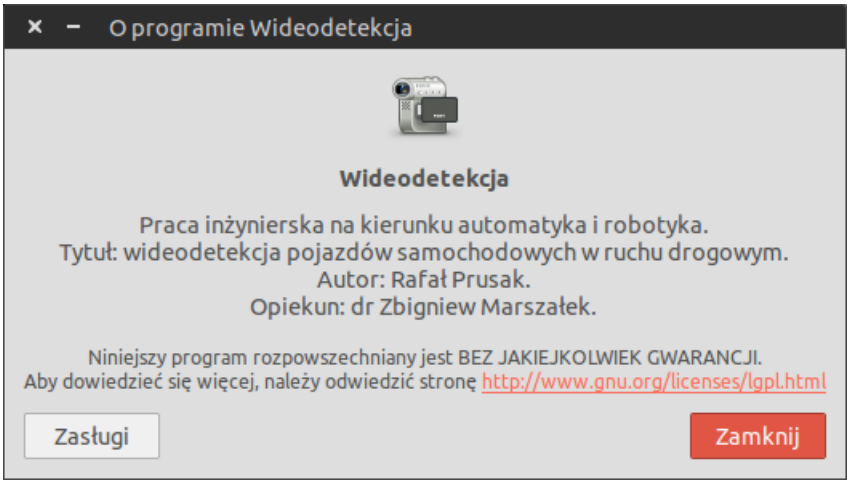


Rysunek 5.7. Okno ustawień programu.

- Miejsce pomiaru prędkości 1/2 - wartość tego pola służy do oznaczenia zmierzonego w rzeczywistości odcinka.
- Zmierzona odległość - długość zmierzonego odcinka.
- Liczba szukanych kolorów - określa liczbę kolorów jaką program ma rozpoznać na obiekcie.
- Odległość strefy od krawędzi w pionie/poziomie- określa ona odległość(wyrażoną w pikselach) o jaką obszar wykrywania jest odsunięty od krawędzi obrazu.
- Minimalna wielkość obiektu - obiekt o mniejszej wartości są ignorowane na etapie detekcji.
- Obszar detekcji - flaga rysowania czerwonego prostokątem ekran będący strefą detekcji kamery.
- Pomiar prędkości- flaga rysowania pionowych linii do pomiaru odległości.
- Samochody - flaga obrysowywania zielonym prostokątem pojazdów.
- Prędkość - podpisywanie obrazu zidentyfikowanego pojazdu informacją o prędkości.
- Rozmiar - podpisuje obraz zidentyfikowanego pojazdu informacją rozmiarze: długości, wysokości, polu powierzchni bocznej.
- Pasek kolorów - rysuje na obrazie zidentyfikowanego pojazdu pasek informujący o kolorze tego pojazdu.
- Kontur pojazdu - rysuje niebieski kontur pojazdu na obrazie wynikowym.

- „Dokumentacja” - uruchamia w domyślnej przeglądarce systemowej stronę główną dokumentacji. Skrót klawiszowy Ctrl+y.

- „O programie” - uruchamia okienko dialogowe pokazujące krótką informację o programie. Skrót klawiszowym Ctrl+u.



Rysunek 5.8. Informacja o programie

- „Zakończ” - wyłącza program. Skrót klawiszowy Ctrl+z.

Funkcje menu odtwarzania:

- Rozpoczęcie przetwarzania. Skrót klawiszowy a.
- Pauza. Skrót klawiszowy s.
- Rozpoczęcie odtwarzania od nowa. Skrót klawiszowy d.
- Stop. Skrót klawiszowy f.
- Zapisz - pozwala zapisać obecną klatę obrazu do pliku. Skrót klawiszowy g.
- Włączenie/wyłączenie nagrywania rezultatu przetwarzania do pliku video.
- Włączenie/wyłączenie algorytmu - gdy przycisk nie jest wciśnięty program wykonuje jedynie detekcję obiektów, pomija zaś śledzenie, analizę i zapis rezultatów.
- Włączenie/wyłącznie pokazywania maski - binarnego obrazu powstałego wy wyniku wycięcia tła.

6. Algorytm detekcji i identyfikacji cech pojazdów.

W tym rozdziale zostały opisane kolejne etapy zaimplementowanego algorytmu. Składa się on z trzech etapów: detekcji obiektów, śledzenia pojazdów i klasyfikacji.

Kalibracja programu

Przed rozpoczęciem analizy próbek wideo należy dobrać odpowiednie ustawienia. W tym celu istnieje możliwość wyłączenia przetwarzania obrazu przez algorytm, co znacząco polepsza wydajność programu na tym etapie. Pierwszym krokiem konfiguracji programu, powinno być dostosowanie rozmiaru strefy czułości pojazdu. Należy przy tym stosować się do następujących zasad:

1. Stefa czułości powinna być tak dobrana aby obejmowała obszar obrazu, będący faktycznym rejonem przejazdu pojazdów. Wybranie zbyt dużej strefy, obejmującej np. pobocze, tło, etc, może powodować pojawienie się błędnych odczytów spowodowanych rejestracją odbić światła i cieni jako poruszających się pojazdów.

2. Strefa powinna być na tyle wąska aby pojazdy były wykrywane dopiero gdy znajdą się całkowicie w kadrze. W przeciwnym razie tracona jest dokładność pomiaru prędkości.

Kolejnym etapem jest wprowadzenie granic pomiaru prędkości. Przed nagraniem danych z ruchu drogowego operator powinien dokonać ręcznego pomiaru odległości, następnie w trakcie konfiguracji programu oznaczyć zmierzony odcinek granicami pomiaru i podać zmierzoną wartość. Wartość ta stanowi podstawę przelicznika pixeli na obrazie na rzeczywistą odległość.

Ostatnim etapem powinno być dobranie minimalnego rozmiaru obiektu. Jest to ważna opcja, kluczowa podczas selekcji obiektów na obrazie. Umożliwia odrzucenie obiektów bodących zakłóceniami. Minimalna wielkość obiektu powinna być dobrana z uwzględnieniem odległości urządzenia nagrywającego od jezdni - im urządzenie było bliżej szosy, dolna granica powinna być większa.

W programie, na chwilę obecną, nie ma możliwości ustawienia przez użytkownika górnej granicy obiektu. Została ona ustalona jako stała wartość równa połowie liczby pikseli klatki obrazu.

Detekcja obiektów

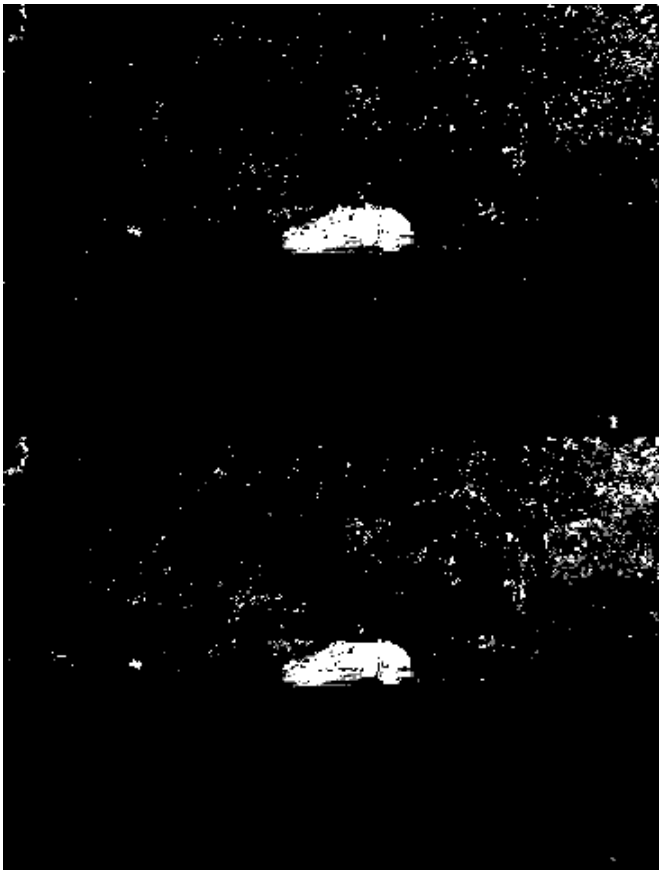
Pierwszym właściwym krokiem algorytmu jest analiza pojedynczej klatki obrazu, znalezienie grup pikseli mogących być obiektami i i ich selekcja.

Pobierane z pliku wideo klatki obrazu są poddawane operacji wyodrębniania tła, która oddziela niezmieniające się, nieruchome elementy tła od poruszających się elementów na pierwszym planie. Operacja ta została zaimplementowana w bibliotece OpenCV pod postacią klas BackgroundSubtractorMOG, BackgroundSubtractorMOG2 oraz BackgroundSubtractorGMG.

BackgroundSubtractorMOG[8] - dokonuje dla każdego piksela obrazu oszacowania czasu przez jaki jego kolor pozostaje niezmienny. Jako elementy tła wybierane są te punkty, których wartości nie zmieniają się wcale lub zmieniają się w niewielkim stopniu. Klasa ta została określona w bibliotece jako przestarzała, nie zaleca się jej stosowania.

BackgroundSubtractorMOG2[13][14] - działa na podobnej zasadzie co powyższa klasa. Dodatkowo dla każdego piksela obrazu wybierany jest inny model przybliżenia czasu niezmienności.

BackgroundSubtractorGMG[7] - na podstawie pierwszych 120 klatek obrazu dokonuje oceny prawdopodobieństwa, który region obrazu może zostać uznany jako tło. Ocena ta ulega zmianie, pod wpływem następnych klatek obrazu.



Rysunek 6.1. Porównanie metod wyznaczania tła.

Na powyższym obrazie zostało zaprezentowanie porównanie metody MOG2(u góry) z metodą GMG(na dole) na przykładowym nagraniu przejazdu samochodu. Rezultaty nie odbiegają od siebie znacząco - można zauważyć, że GMG generuje odrobinę więcej zakłóceń. Metoda ta ma jednak ogromną zaletę - potrafi wyodrębniać cienie obiektów, co pozwala na ich usuwanie na etapie filtracji. W programie zdecydowano się na użycie tej metody.

Następnym etapem jest filtracja uzyskanego obrazu binarnego. Odbywa się ona po przez wykonanie na obrazie kolejno: otwarcia, mediany oraz dylatacji[12].

Binaryzacja to jedno z podstawowych przekształceń obrazu. Polega na zamianie wartości widocznych na obrazie przedstawionym w skali szarości na poziomy 0 i 1(ewentualnie 0 i 255). Kluczowym elementem tego algorytmu jest dobór progu. Stosuje się binaryzację z pojedynczym progiem, odwrotną, z progiem podwójnym i wielokryteriową. Uzyskanie odpowiedniego obrazu binarnego stanowi pierwszy etap algorytmów detekcji obiektów i rozpoznawania kształtów.

Mediana jest operacją kontekstową - dla przeprowadzenia wyliczenia wartości transformacji dla konkretnego piksela obrazu potrzebna jest informacja o otoczeniu, nazwanym również oknem. Charakterystyczną cechą mediany jest to, że równocześnie usuwając zakłócenia nie powoduje uszkodzenia krawędzi obrazu. Działanie metody oparte jest, jak wskazuje nazwa, na matematycznej operacji mediany: z uporządkowanego ciągu wartości pikseli w oknie wybierana jest wartość środkowa, która jest wynikiem transformacji dla elementu centralnego okna. Daje to szereg korzyści: wartości ekstremalne są odrzucane przez filtr, a operacja nie wprowadza do obrazu nowych wartości, ponieważ operuje wyłącznie na wartościach już istniejących w obrazie. Negatywnym skutkiem zastosowania mediany jest pogorszenie jakości zaokrąglonych obszarów. Zjawisko to jest coraz bardziej widoczne wraz ze wzrostem rozmiaru obszaru przetwarzania. Wyliczenie wartości mediany jest kosztowne obliczeniowo, gdyż wymaga dla każdego punktu obrazu posortowania wartości elementów okna.

Erozja należy do klasy operacji przetwarzania obrazu nazwanej przekształceniami morfologicznymi. Opierają się one na zdefiniowaniu jądra transformacji(w postaci siatki eliptycznej, kwadratowej, etc), które zawiera warunki dotyczące wartości pikseli obrazu. Operacja morfologiczna polega na przyłożeniu elementu strukturalnego do każdego punktu obrazu i transformacji tych pikseli, które spełniają warunek. Własność ta powoduje, że w przeciwieństwie do innych metod(punktowych, kontekstowych) przekształceniu ulega tylko fragment obrazu. Sama erozja jest zdefiniowana jako filtr minimalny: każdemu elementowi przypisywana jest minimalna wartość z pośród sąsiednich pikseli. Na obrazie binarnym jej działanie objawia się pomniejszeniem pola powierzchni „skupionych" obszarów oraz usunięciem zakłóceń w postaci cienkich „włosów". Może zostać wykorzystana do odseparowania od siebie obszarów.

Dylatacja jest również operacją morfologiczną, przeciwną do erozji, gdyż działa jak filtr maksymalny. Powoduje usunięcie zakłóceń w postaci „dziur" i zagłębień wewnątrz obszarów. Małe, położone blisko siebie obszary zostają połączone w jeden większy rejon.

Otwarcie jest kombinacją powyższych dwóch metod: najpierw na obrazie wykonuję się erozje, a potem dylatację. W ten sposób zapobiega się zmianom pola powierzchni obiektów. Na obrazie binarnym powoduje usunięcie drobnych wklęsłości i rozłączenie niektórych obiektów. Negatywną cechą tej operacji jest zamazywanie i zniekształcanie obrazu.

Zastosowana metodologia filtracji i selekcji zapewnia że program skutecznie wykrywa obiekty będące pojazdami oraz odrzuca większość zakłóceń. Na poniższym obrazie zaprezentowano rezultat - kolejno od góry znajduje się obraz oryginalny, obraz po wykryciu tła oraz obraz po filtracji.

Rysunek 6.2. Rezultat filtracji.

Na przefiltrowanym obrazie jest oznaczane są unikalnymi wartościami rozłączne elementy pierwszego planu. Każdy znaleziony obiekt jest poddawany ocenie względem kryte-

rium rozmiaru. Jeżeli ocena jest pozytywna program dokonuje opisu rejonu elementu prostokątem i informację o położeniu lewego-górnego wierzchołka prostokąta, wysokości, długości i środku ciężkości obiektu zapisuje do listy rezultatów detekcji.

Śledzenie pojazdów

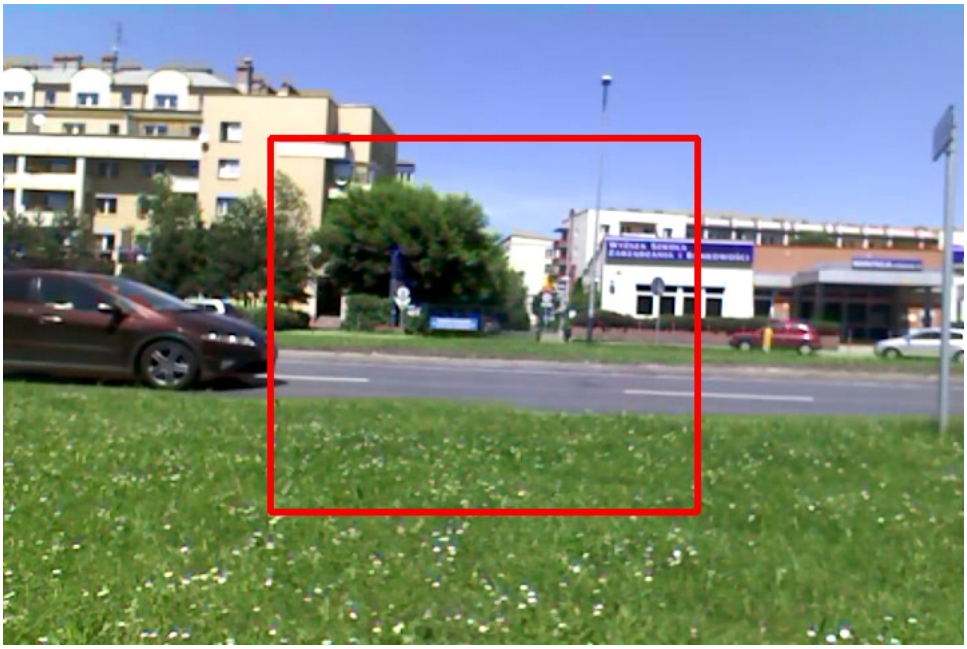
Śledzenie pojazdów polega na analizie informacji o obiektach znajdujących się na kolejnych klatkach obrazu. Klasa odpowiedzialna za śledzenie dokonuje przeglądu wszystkich obiektów, pod kątem położenia w rejonie czułości kamery według poniższego algorytmu:

1. Sprawdź czy obiekt znajduje się w rejonie czułości kamery. Jeżeli nie idź do punktu 5.
2. Sprawdź czy obiekt znajduje się blisko lewej bądź prawej krawędzi obrazu. Jeśli nie, idź do punktu 5.
3. Jeżeli lista obiektów przy przeciwnej krawędzi jest pusta, dodaj obiekt z aktualną klatką obrazu do listy tej strony. Sytuacja ta oznacza, że pojazd pojawił się w obszarze czułości kamery.
4. Jeżeli lista obiektów przy przeciwnej krawędzi rejonu detekcji nie jest pusta, pobierz pierwszą informację z tej listy i przekaż informacje o obiektach z obu stron dalej. Sytuacja ta oznacza, że pojazd opuszcza rejon wykrywania.
5. Weź następny obiekt do analizy.

W celu zabezpieczenia przed kilkukrotnym wykryciem i zapisaniem tego samego przejazdu, po wykryciu pojawienia się pojazdu lub opuszczenia przez niego obszaru, wykrywanie zostaje zablokowane aż do momentu opuszczenia otoczenia krawędzi rejonu detekcji.

Ocena obecności w otoczeniu krawędzi odbywa się po przez obliczenie odległości środka ciężkości obiektu od krawędzi rejonu. Jeżeli ta odległość jest mniejsza niż 50 pikseli, obiekt jest klasyfikowany jako znajdujący się w otoczeniu krawędzi.

Śledzenie przykładowego przejazdu pojazdu

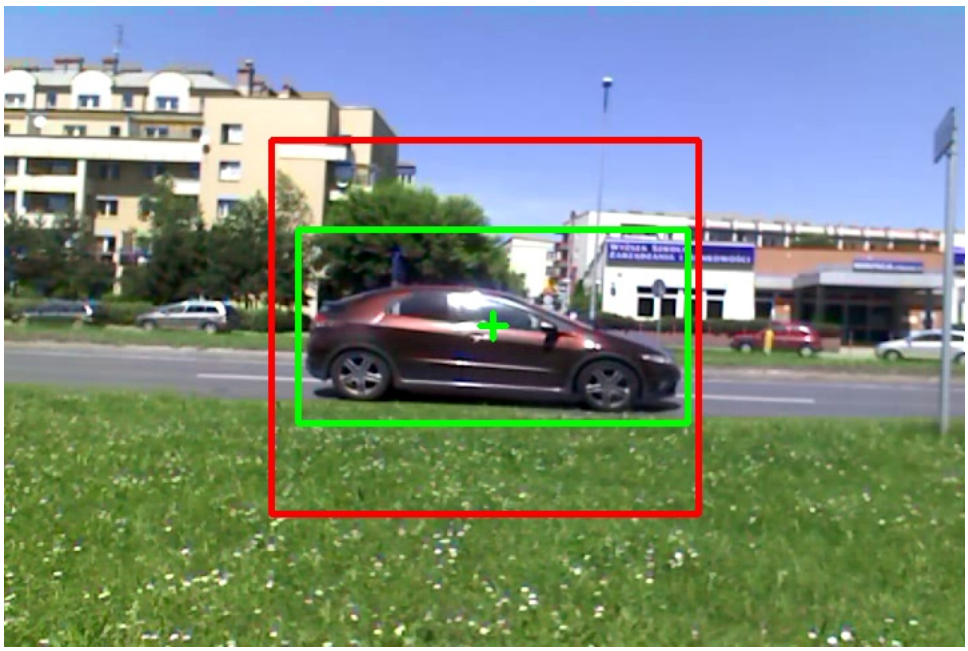


Rysunek 6.3. Nowy pojazd w kadrze.

W kadrze kamery pojawia się pojazd, ale jest po za rejonem detekcji, więc jest ignorowany przez program. Odpowiada to krokowi 1 i 2 przedstawionego wyżej algorytmu.

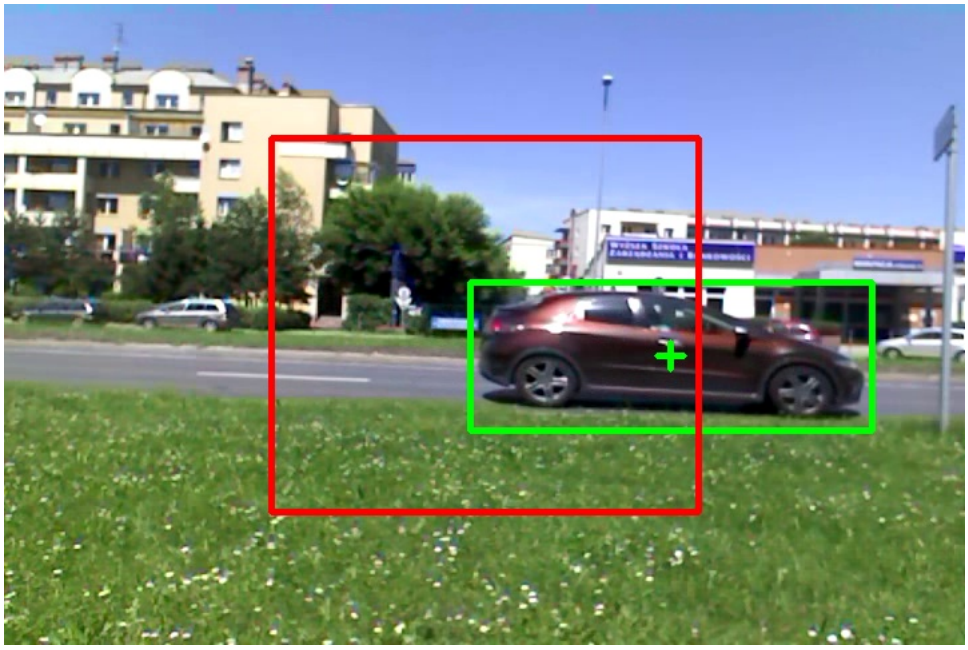
Rysunek 6.4. Wykrycie pojazdu.

Pojazd wchodzi w obszar czułości kamery. Program zapisuje aktualną klatkę obrazu, jej numer oraz współrzędne środka ciężkości pojazdu. Odpowiada to krokowi 3 algorytmu.



Rysunek 6.5. Pojazd w obszarze detekcji.

Środek ciężkości samochodu znajduje się w obszarze wykrywania. Program śledzi jego ruch, aż zbliży się do przeciwnego boku prostokąta.



Rysunek 6.6. Opuszczenie rejonu detekcji.

Samochód jest w otoczeniu drugiej krawędzi prostokąta. Program, podobnie jak na drugim obrazie, pobiera klatkę obrazu, jej numer i współrzędne środka ciężkości i przesyła dane to klasy zajmującej się obliczaniem parametrów. Jest to krok 4 algorytmu.

Rysunek 6.7. Opuszczenie kadru.

Przejazd samochodu został sklasyfikowany. Pojazd opuścił rejon wykrywania i nie jest już interesujący dla programu.

Identyfikacja

Oprogramowanie ma dostarczyć danych o rozmiarze pojazdu, kolorze i prędkości.

Na wyznaczenie rozmiaru składa się obliczenie jego długości, wysokości i pola powierzchni bocznej karoserii. Wysokość i długość są wyliczane jako długość i wysokość zielonego prostokąta otaczającego obiekt razy przelicznik metrów na piksele, natomiast pole powierzchni bocznej to pole konturu samochodu razy przelicznik do kwadratu.

W celu wykrycia koloru, program wyszukuje w prostokącie otaczającym pojazd K dominujących kolorów i wybiera ten największy. Funkcja wyboru kolorów została zaimplementowana z wykorzystaniem klasy KMeans[1] z pakietu scikit-learn[2]. Zasada działania tego algorytmu jest następująca: podziel zbiór N próbek ze zbioru X na K grup, z których każda jest charakteryzowane przez wartość u, tak aby zminimalizować sumę kwadratów różnic w każdej grupie. Poniższy obraz przedstawia rezultat użycia tego algorytmu: po lewej znajduje się obraz oryginalny, po prawej stronie obraz zredukowany do 4 kolorów.

Rysunek 6.8. Przykład działania algorytmu K-means.

Aby zidentyfikować prędkość pojazdu program pobiera różnicę klatek między pojawieniem się a opuszczeniem pola detekcji przez samochód, następnie znając liczbę fps na sekundę określa czas między zajściem tych zdarzeń. Kolejnym krokiem jest wyliczenie różnicy między środkami ciężkości pojazdu w obu miejscach. Znając różnicę w czasie, odległość w pikselach oraz przelicznik metrów na piksele program może wyliczyć prędkość.

Wszystkie wyniki klasyfikacji są pakowane do struktury danych, zwanej w języku Python słownikiem, i przesyłane do modułu zajmującego się zapisywaniem wyników do obrazów i baz danych.

7. Analiza wyników

Wyniki działania programu zostały porównane z danymi pochodzącymi z pomiaru za pomocą pętli indukcyjnych. Dane te miały postać plików *.avi oraz plików tekstowych w formacie „kolumnowym”:

Pole	Opis	Postać lub jednostka
1	Data	RRRR-MM-DD
2	Godzina	HH:MM:SS
3	Temperatura	st. C.
4	Prędkość	km/h
5	Długość	m
6	Liczba osi	
7-12	Odległość między osiami	
13-19	Naciski osi z czujnika nr 1	
20-26	Naciski osi z czujnika nr 2	

Tablica 7.1. Format pliku wejściowego.

Plik avi był powiązany z wpisem w pliku tekstowym(nazwą pliku była godzina pomiaru, z dokładnością do kilku sekund).

Skrypt przeprowadzający testy

W celu automatyzacji testowania został napisany skrypt w języku Python.

```
from os import listdir
from datetime import datetime
from numpy import mean, median
from src.video import VideoReader, Frame
from src.alg import Algorithm
from src.config import Configuration
from src.logs import Database, ImageSaver, Logger
```

Pierwszy etapem w skrypcie jest zaimportowanie potrzebnych bibliotek i klas:

os - moduł biblioteki standardowej, zapewniający podstawowe funkcje zależne od architektury systemu operacyjnego(otwieranie/czytanie plików, poruszanie się po hierarchii katalogów, manipulowanie zmiennymi systemowi)

datetime - moduł biblioteki standardowej, służący do manipulacji datą, czasem, strefą czasową, umożliwiający także odmierzanie czasu.

numpy - zewnętrzna biblioteka do operacji numerycznych

VideoReader - klasą do odczytu danych z pliku wideo, zapewniająca dostęp do właściwości pliku.

Frame - klasa opakowująca klatki obrazu pobrane z pliku.

Algorithm - klasa reprezentująca algorytm programu.

Configuration - klasa pozwalająca zmieniać, przywracać i zapisywać ustawienia programu.

Database - klasa zapisująca wyniki do bazy danych.

ImageSaver - reprezentacja rezultatów w postaci obrazów.

Logger - zapisywanie komunikatów diagnostycznych do pliku tekstowego.

```
input_file = open("input.txt")
output_file = open("output.txt", "w")
folder = "/media/Dane/Dropbox/Studia/IV␣rok/inżynierka/Dane_AVI"
avi_files = listdir("/media/Dane/Dropbox/Studia/IV␣rok/inżynierka/Dane_AVI")
db = Database()
img_saver = ImageSaver()
Logger.start()
Configuration.load_config()
```

Następnie skrypt otwiera plik z wynikami pomiarów, pobiera listę nagrań, tworzył obiekty do zapisu danych i ładuje ustawienia.

```
def test_sigle_file(file):
    Algorithm.reset()
    input_video = VideoReader(file)
    result = []
    while 1:
        frame = Frame(input_video)
        if not input_video.is_good():
            break
        frame = Algorithm.resize(frame)
        frame, mask, records = Algorithm.perform(frame, db, img_saver, True)
        if records is not None and len(records) > 0:
            for rec in records:
                r1 = round(rec['width'], 2)
                r2 = round(rec["speed"], 2)
                result.append((r1, r2))
    return result
```

Na listingu powyżej znajduje się funkcja, wykonywana w głównej pętli programu. Kolejne klatki obrazu są czytane z pliku wideo, konwertowane na rozmiar 480x720 pikseli i przetwarzane przez algorytm. Funkcja wyciąga z rezultatów przetwarzania potrzebne do późniejszego porównania prędkość i długość pojazdu. Pełny rezultat jest zapisywany w plikach jpg i bazie danych.

```
for i, line in enumerate(input_file.readlines()):
    data = line.split(sep="\t")
    lp = str(i)
    date, time, speed, lenght, axes = , data[0], data[1], data[3], data[4], data[5]
    HH, MM, SS = time.split(sep=":")[0], time.split(sep=":")[1], time.split(sep=":")[2]
    orginal_file = "M151001_"+HH+MM+SS+".avi"
```

Kolejnym etapem jest przegląd całego pliku wejściowego. Z każdej linii tekstu wyciągana jest data, prędkość, długość samochodu oraz liczba osi. Na podstawie czasu pomiaru jest rozpoznawana nazwa pliku wideo.

```
    if float(speed) > 0 and float(lenght) > 0 and int(axes) in (1, 2):
        found_file = None
        for SS2 in range(int(SS)-2, int(SS)+3):
            s = str(SS2) if len(str(SS2)) > 1 else "0"+str(SS2)
            f = "M151001_"+HH+MM+s+".avi"
            if f in avi_files:
                found_file = f
                used_files_count += 1
                results = test_sigle_file(folder+"/"+f)
```

Skrypt sprawdza poprawność pomiaru: czy prędkość i długość jest dodatnia oraz czy liczba osi jest równa 1 bądź 2. Drugie kryterium jest wynikiem ustawienia kamery na stanowisku pomiarowym. Jest ona ustawiona zbyt blisko jezdni, przez co dłuższe pojazdy, posiadające więcej osi, nie mieszczą się w kadrze kamery co uniemożliwia ich detekcję w programie. Jeżeli dane są poprawne, wyszukiwany jest plik wideo odpowiadający tym danym. Znaleziony plik poddawany jest analizie.

```
                for r in results:
                    found_length, found_speed = r[0], r[1]
                    # błąd długości
                    length_error = round(abs(float(lenght) - found_length), 2)
                    length_error_pr = round((length_error / float(lenght)), 2) * 100
                    length_errors.append(length_error_pr)
                    # błąd prędkości
                    speed_error = round(abs(float(speed) - found_speed), 2)
                    speed_error_pr = round((speed_error / float(speed)), 2) * 100
                    speed_errors.append(speed_error_pr)
                    if speed_error_pr < 25 and length_error_pr < 25:
                        status = "OK"
                        valid_detection_count += 1
                    else:
                        status = "BAD"
                        bad_detection_count += 1
                output_file.write(form % (lp, lenght, speed, orginal_file, found_file,
                    str(found_length), str(length_error), str(length_error_pr),
                    str(found_speed), str(speed_error), str(speed_error_pr), status))
                break
```

Następuje porównanie danych otrzymanych z przetwarzania obrazu z danymi wejściowymi. Jeśli dane te nie różnią się o więcej niż 25%, rezultat uznawany jest za prawidłowy, a odpowiednia informacja jest zapisywana w pliku wyjściowym.

Skrypt testujący zlicza prawidłowe oraz nieprawidłowe wpisy w plikach i rezultaty. Rekord w wejściowym pliku tekstowym, jest uznany za prawidłowy, wtedy gdy prędkość i długość ma dodatnią wartość a liczba osi jest równa jeden bądź dwa. Rezultat analizy jest uznawany za prawidłowy, jeżeli obliczona prędkość i długości nie różni się o więcej niż o 25% od danych wejściowych.

```
        if found_file is None:
            not_found_files_count += 1
            status = "nie znaleziono pliku"
            output_file.write("%5s %10s %10s %20s %20s \n"
            % (lp, lenght, speed, orginal_file, status))

    else:
        unused_files_count += 1
        status = "niepoprawny pomiar"
        output_file.write("%5s %10s %10s %20s %20s \n"
        % (lp, lenght, speed, orginal_file, status))
```

Na końcu do pliku zapisywane są komunikaty o nieznalezieniu pliku wideo i błędnym formacie danych wejściowych. Zwrócony rezultat jest zapisywany do pliku tekstowego, którego format opisuje poniższa tabela.

Nazwa pola	Opis
lp	numer wpisu z pliku wejściowego
długość	długość pojazdu z pliku wejściowego
prędkość	prędkość pojazdu z pliku wejściowego
plik oryginalny	nazwa pliku avi na podstawie czasu pomiaru
plik znaleziony	dopasowany plik avi
długość	długość wykryta na podstawie obrazu wideo
błąd	różnica pomiarów długości
błąd %	stosunek błędu długości do wartości z pliku wejściowego
prędkość	prędkość wykryta na podstawie obrazu wideo
błąd	różnica pomiarów prędkość
błąd %	stosunek błędu prędkości do wartości z pliku wejściowego
status	„OK” jeżeli pomiar został uznany za prawidłowy, „BAD” w p. p.

Tablica 7.2. Format pliku wynikowego.

Wynik testów

W poniższej tabeli przedstawiono wynik działania skryptu.

Nazwa	Wartość
Wpisy zweryfikowane jako prawidłowe	2111
Wpisy zweryfikowane jako nieprawidłowe	330
Wpisy bez pliku wideo	181
Obiekty zidentyfikowane prawidłowo	1657
Obiekty zidentyfikowane błędnie	666
Średni błąd wyznaczenia długości	16 %
Mediana błędu wyznaczenia długości	12 %
Błąd średni wyznaczenia prędkości	22 %
Mediana błędu wyznaczenia prędkości	15 %
Czas wykonania skryptu	1:33:23

Tablica 7.3. Rezultat testu.

Skrypt testujący odrzucił około 1/5 danych wejściowych. Pojazdy posiadające większą liczbę osi, zostały na wstępie odrzuce, gdyż z powodu pozycji kamery na stanowisku pomiarowym, program nie był w stanie dokonać poprawnej identyfikacji.

Algorytm osiągnął poprawność sięgającą 72 procent. Główną przyczyną takiego wyniku testu jest niezgodność zarejestrowanych przejazdów z założeniami dotyczącymi działania programu. Błędne przypadki testowe zostały opisane w następnym podrozdziale.

Użycie skryptu pozwoliło przyśpieszyć testowanie, gdyż wydajność głównego programu jest ograniczona możliwościami widgetu z biblioteki PyGtk służącego do wyświetlania obrazu. Interfejs graficzny, aby program działał stabilnie, może wyświetlać nową klatkę obrazu co 100 ms, natomiast skrypt działał bez wyświetlania obrazu.

Nieudane przypadki działania

Podstawową przyczyną błędnej identyfikacji było założenie przyjęte przy pisaniu programu, mówiące o tym, że program ma służyć do identyfikacji pojazdów na drodze jednokierunkowej. W trakcie pracy nad algorytmem i późniejszych testów okazało się, że detekcja pojazdów w ruchu na drodze dwupasmowej jest zadaniem bardzo trudnym, jeżeli program ma do dyspozycji jedynie obraz wideo z pobocza drogi. Zachodzi wtedy wiele niekorzystnych zjawisk:

- Mijanie się samochodów - w tej sytuacji dochodzi do zlewania się obiektów w jeden na obrazie binarnym a pojazdy mogą dotrzeć do przeciwnych końców drogi w odwrotnej kolejności. W celu zabezpieczenia programu przed wynikającymi z tej sytuacji przekłamaniami, należałoby dokonać poważnych zmian w algorytmie: zamiast pobierać informację o pojeździe na końcach obszaru „czułości” kamery, program musiałby śledzić obiekt w całym polu, wykryć kierunek ruchu i nadać mu szacowaną prędkość(przyrost pozycji w pikselach na kratkę obrazu).

Rysunek 7.1. Wymijanie.

- Wyprzedzanie się pojazdów - ta sytuacja może powodować, że dwa obiekty staną się dla programu jednym, jeżeli manewr zakończy się po za polem widzenia kamery. Skutkiem tej sytuacji będzie całkowita utrata wiadomości o obiekcie.

Rysunek 7.2. Wyprzedzanie.

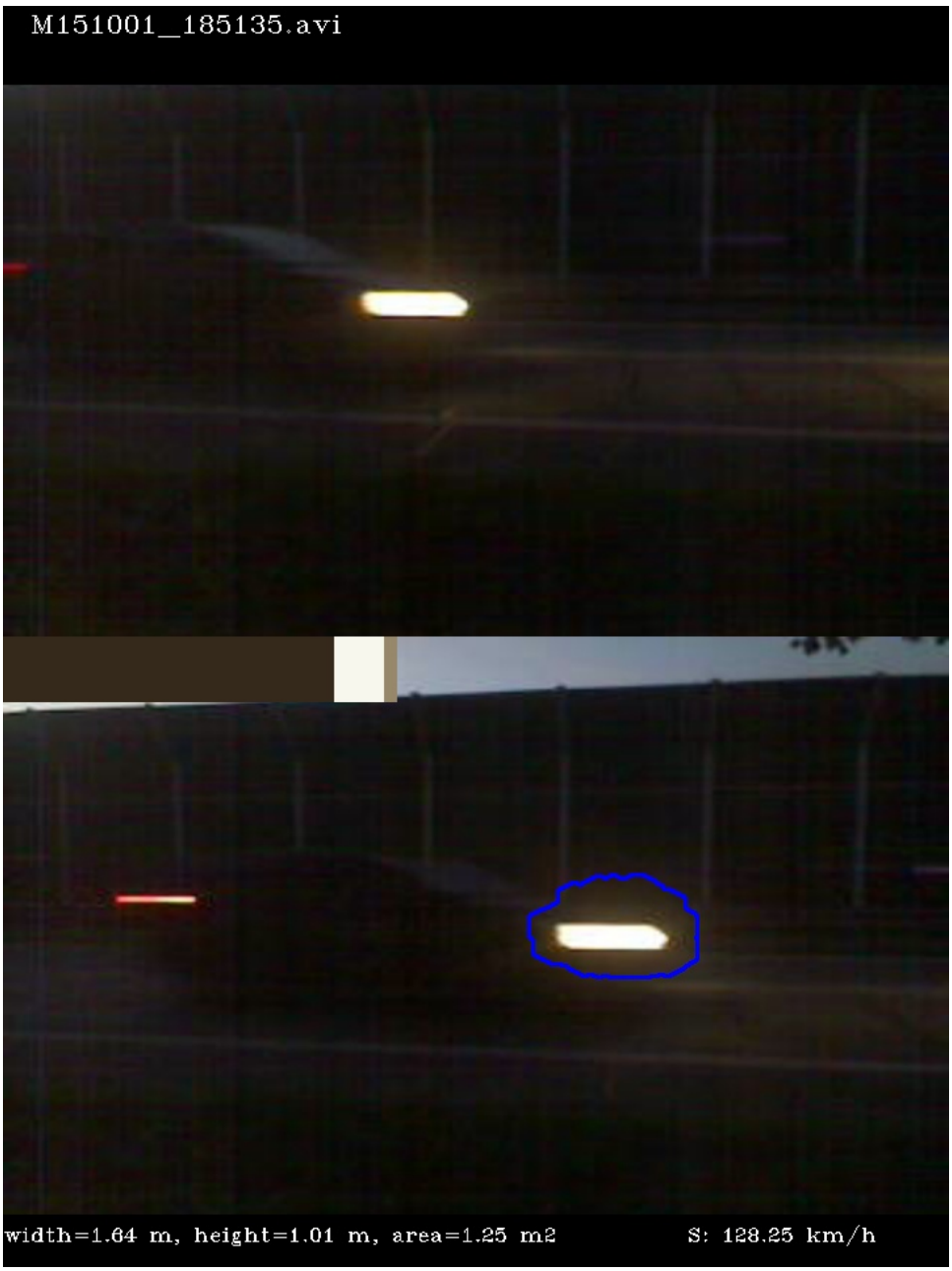
- Wzajemne rzucanie cienia i odbicia światła od samochodów - Powoduje to zniekształcenia na obrazie po wyodrębnianiu tła, takie jak połączenia się dwóch pojazdów w jeden w skutek odbicia światła, albo podział pojazdu na dwa niezwiązane ze sobą obiekty.

Rysunek 7.3. Pojazd na innym pasie.

Stanowisko pomiarowe było umieszczone przy drodze dwukierunkowej, posiadającej po dwa pasy ruchu w każdym kierunku. Pomiar za pomocą pętli indukcyjnych był przeprowadzany tylko na jednym pasie, przez co zidentyfikowane na podstawie obrazu parametry pojazdów na innych pasach były całkowicie nieadekwatne do danych wejściowych.

Rysunek 7.4. Pojazd niemieszczący się kadrze.

Rysunek 7.4 prezentuje sytuację, gdy pojazd nie mieści się kadrze. W wyniku tego nie można poprawnie określić długości pojazdu. Obliczenie prędkości auta wymaga znajomości położenia środka ciężkości i indeksu ramki obrazu. Jeżeli pojazd nie jest w całości w polu widzenia kamery, środek znajduje się w niewłaściwym miejscu, co obniża prędkość wynikową.



Rysunek 7.5. Pomiar w godzinach wieczornych.

Na pomiar miały wpływ także warunki oświetleniowe. Na rysunku 7.5 pokazany jest pomiar w godzinach wieczornych. Mniejsza ilość światła słonecznego pogarszała widoczność konturu karoserii, co skutkowała błędnym oszacowaniem rozmiaru obiektu.

8. Podsumowanie

Celem niniejsze pracy było stworzenie oprogramowania umożliwiającego przeprowadzenie detekcji pojazdów samochodowych na podstawie obrazu wideo.

Systemy wizyjne znajdują coraz szersze wykorzystanie w aplikacjach związanych z ruchem drogowym. Jest to spowodowane kilkoma czynnikami, takimi jak: prosta, bezinwazyjna instalacja urządzeń monitorujących, możliwość transmisji danych przewodowo lub bezprzewodowo, zdalna obsługa stanowiska pomiarowego, łatwość integracji kamer z innymi elementami systemów ruchu drogowego.

Napisane oprogramowanie spełniało swoje zadanie. Program przeprowadzał detekcję pojazdów oraz dokonywał oceny wykrytych pojazdów pod względem rozmiaru, koloru i prędkości. Umożliwiał użytkownikowi dostosowywanie ustawień i parametrów algorytmu, zapis danych w kilku formatach(jpg, avi, sqlite3) oraz możliwość sterowania przebiegiem przetwarzania danych wejściowych.

Niestety nie we wszystkich przypadkach program działał prawidłowo. Pewne specyficzne sytuacje pojawiające się w wejściowych nagraniach powodowały błędy detekcji i identyfikacji pojazdów lub całkowicie uniemożliwiały wykrycie pojazdu. Powodem tych sytuacji były ograniczenia algorytmu oraz uproszczenia przyjęte w czasie pisania programu.

Realizacja pracy inżynierskiej pozwoliła autorowi utrwalić i wykorzystać wiedzę z przedmiotu „systemy wizyjne” w zagadanieniu praktycznym. W programie wykorzystano wiele algorytmów i operacji wizyjnych omawianych na zajęciach. Podczas tworzenia oprogramowania autor dogłębnie zapoznał się z bardzo popularną biblioteką OpenCV. Dodatkowo nauczył się nowego języka programowania i zaznajomił się z najważniejszymi możliwościami jego wykorzystania, takimi jak budowa interfejsów graficznych, praca z plikami(tekstowymi, graficznymi, json), obsługa baz danych czy przetwarzanie wielowątkowe.

Dalszy rozwój oprogramowania

Autor oprogramowania planuje jego dalszy rozwój. W przyszłości możliwe byłoby wprowadzenie następujących zmian:

- Lepszy podział kodu - przez znaczną część okresu powstawania programu, proces ten odbywał się bez dokładnej wiedzy o końcowym kształcie projektu. Spowodowało to, że kod programu miejscami jest niezrozumiały, w niektórych miejscach pojawiają się pozostałości po niewykorzystanych modułach i rozwiązaniach, które porzucono. Kilka

metod jest zdecydowanie za długie(funkcja odpowiadająca za śledzenie pojazdu). W kodzie mieszają się operacje o różnym poziomie abstrakcji, na przykład każda z klas dokonujących detekcji, śledzenie i oceny dorysowuje własne oznaczenia do obrazu wideo. Poprawa tych mankamentów będzie kluczowa i pozwoli na sprawne dodawanie nowych możliwości do programu.

- Identyfikacja pojazdów on-line - aby dodanie tej funkcji do programu było możliwe, musiałby on znacząco podnieść swoją wydajność. Rozwiązaniem tego problemu byłoby przemodelowanie algorytmu, w taki sposób, aby każdy etap przetwarzania(odczyt obrazu, detekcja, śledzenie, identyfikacja i zapis wyników) działał niezależnie w osobnym wątku. Konieczna byłaby również rezygnacja z wyświetlania obrazu, gdyż pochłania ona sporo czasu i mocy obliczeniowej.

- Strona WWW - program mógłby w łatwy sposób wysyłać dane z pomiarów(np. zapisywać je w bazie danych mysql), które następnie byłyby wizualizowane na stronie www.

Spis rysunków

Bibliografia

[1] K-means user guide. Dostęp: 28 XII 2015. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[2] Machine learning in python. Dostęp: 28 XII 2015. [Online]. Available: http://scikit-learn.org/stable/index.html

[3] CatTraffic. Oferta produktów firmy. [Online]. Available: http://www.cat-traffic.pl/produkty/

[4] J. Gajda, inni, *Pomiary parametrów ruchu drogowego*. Wydawnictwa AGH, Kraków, 2012.

[5] GDDKiA. Generalny pomiar ruchu w 2015. [Online]. Available: http://www.gddkia.gov.pl/pl/2551/GPR-2015

[6] GDDKiA. Instrukcja o sposobie przeprowadzenia generalnego pomiaru ruchu w roku 2015 (załącznik d wytycznych gpr 2015). [Online]. Available: https://www.gddkia.gov.pl/userfiles/articles

[7] A. B. Godbehere, A. Matsukawa, K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," 2012.

[8] P. KaewTraKulPong, R. Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," *Vision and Virtual Reality group, Department of Systems Engineering, Brunel University*, 2001.

[9] Nuerosoft. Strona internetowa firmy. [Online]. Available: https://neurosoft.pl/

[10] The python language reference. The Python Software Foundation. [Online]. Available: https://docs.python.org/3.4/reference/index.html

[11] The python standard library. The Python Software Foundation. [Online]. Available: https://docs.python.org/3.4/library/index.html

[12] R. Tadeusiewicz, P. Korohoda, *Komputerowa analiza i przetwarzania obrazów*. Wydawnictwo Fundacji Postępu Telekomunikacji, 1997.

[13] Z.Zivkovic, “Improved adaptive gausian mixture model for background subtraction,” *International Conference Pattern Recognition*, 2004.

[14] Z.Zivkovic, F. van der Heijden, “Efficient adaptive density estimapion per image pixel for the task of background subtraction,” *Pattern Recognition Letters, vol. 27*, 2006.