AKADEMIA GÓRNICZO – HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI

KATEDRA AUTOMATYKI

Praca magisterska

Imię i nazwisko:	**Marcin Pomarański**
Kierunek studiów:	Automatyka i Robotyka
Temat pracy dyplomowej:	**„System aktywnego śledzenia oczu kierowcy celem uniknięcia kolizji na przypadek zaśnięcia kierowcy samochodu.”**
Opiekun pracy:	dr Adrian Horzyk

Kraków, rok 2007

Chciałbym złożyć
najserdeczniejsze podziękowania
Panu dr Adrianowi Horzykowi
za okazaną życzliwość, zrozumienie,
pomoc oraz bezcenne uwagi przekazywane
w trakcie tworzenia niniejszej pracy.

1. Wstęp

Obecny rozwój cyfrowych technik przetwarzania obrazu oraz miniaturyzacja wydajnych procesorów graficznych pozwalają na szybką oraz stosunkowo niedrogą budowę skomplikowanych urządzeń analizy ruchomego obrazu w czasie rzeczywistym. Jednym z przykładów takiego systemu jest mechanizm wykrywania faktu zasypiania kierowcy samochodu podczas prowadzenia pojazdu.

Niezbędnym elementem takich systemów jest zwykle kamera, która śledzi oczy osoby kierującej samochodem. Obraz pozyskany z kamery poddawany jest obróbce poprzez system procesorów graficznych oraz wnikliwej analizie przy pomocy algorytmów „wykrywania senności". Kierowca informowany jest o swoim stanie psychofizycznym, jeśli jego stan wskazuje na senność niebezpieczną dla niego lub innych uczestników ruchu drogowego. Komputer, zależnie od sytuacji sugeruje spożycie napoju energetycznego lub filiżankę kawy. Jeżeli kondycja kierowcy nie uległa poprawie, system proponuje krótką drzemkę lub (zależnie od sytuacji) dłuższy odpoczynek. Na wypadek zaśnięcia osoby prowadzącej samochód, podnoszony jest alarm, włączają się światła awaryjne, w przypadku braku reakcji uruchamiane jest łagodne hamowanie pojazdu. Takie urządzenia coraz częściej montuje się w samochodach ciężarowych oraz autokarach zwiększając poziom biernego bezpieczeństwa.

Kulminacyjnym punktem pracy algorytmu jest wykrycie faktu zaśnięcia za kierownicą. Osoba prowadząca powinna zostać szybko, a zarazem nie gwałtownie przebudzona, samochód musi zacząć kontrolowane wyhamowywanie prędkości, zostają włączone światła awaryjne. Dalsza jazda powinna być warunkowana zmiana kierowcy lub poprzedzona dłuższym odpoczynkiem. Naturalnie mechanizm nie może powodować zagrożenia na drodze, ewentualne awaryjne hamowanie musi odbywać się z uwzględnieniem rygorystycznych warunków bezpieczeństwa. Auto nie może wpaść w poślizg, trakcja musi być zachowana. Zatrzymanie się pojazdu nie może być zbyt nagłe, nie trudno wyobrazić sobie skutki takiego hamowania na autostradzie. Dodatkowo samochód może przygotować się do ewentualnego zderzenia, pasy powinny być napięte, zagłówki ustawione we właściwej pozycji. System powinien więc być sterowany przez centralny komputer zainstalowany w samochodzie, tylko wtedy istnieje duża szansa prawidłowej współpracy algorytmu z systemami ABS oraz kontroli trakcji. Takie systemy sukcesywnie stają się integralną częścią samochodów osobowych, w Europie badania

„śpiących" kierowców od kilku lat przeprowadza Mercedes, Audi, Volvo, w USA ze swoich „inteligentnych samochodów" znany jest koncern Chrysler (auta potrafią „same się" prowadzić po autostradzie), w Japonii najbardziej zaawansowanymi pracami nad autem przyszłości może pochwalić się Toyota. Niemieccy inżynierowie pracujący w firmie Mercedes już w 2000 roku rozpoczęli prace nad zautomatyzowaniem mechanizmu wykrywanie śpiących kierowców aut ciężarowych oraz autokarów. Duże, mało wydajne obliczeniowo komputery o niemalże nieograniczonym poborze energii można było instalować jedynie w samochodach użytkowych. Kamera analogowa wraz ze specjalną kartą rozszerzeń montowaną w laptopie rejestrowała obraz twarzy kierowcy, specjalne czujniki zapisywały ruchy głowy oraz rąk. Film z kamery uzupełniony o zapis przebiegu fal mózgowych kierowcy testowego stanowiły podstawę badań oraz analiz kondycji psychofizycznej osoby prowadzącej pojazd. Naukowcy doszli do wniosku, że sama analiza ruchu powiek jest wystarczająco skutecznym sygnałem mówiącym o „poziomie" senności kierowcy. W 2002 roku autokary „z górnej półki" posiadały pierwsze systemy wykrywania zmęczenia osób prowadzących pojazd. Prace nad podobnym systemem prowadzone były przez innych Europejskich producentów aut ciężarowych, samochody Scania oraz Volvo również są wyposażane w podobne mechanizmy, jednak wielu klientów ciągle traktuje je jako tzw. „drogą zabawkę". Właściciele firm transportowych nie dostrzegają słuszności inwestowania w opisywany system ze względu na niską stopę zwrotu. Prawo obowiązujące w Unii Europejskiej narzuca na kierowcach oraz na zarządzających flota pojazdów regularne odpoczynki wraz z przerwami na sen. Stosunkowo wysokie kary – zwłaszcza na zachodzie Europy – przyczyniają się do przestrzegania tych przepisów, a co za tym idzie do eliminowania z dróg nie wyspanych kierowców. Systemy analizy kondycji psychofizycznej osób prowadzących pojazdy mechaniczne stały się zbędne. Obecnie niewielu z Europejskich producentów samochodów oferuje opisywany system w wyposażeniu dodatkowym. Koncerny samochodowe z USA oraz Japonii propagują model tzw. inteligentnego samochodu osobowego, szpikując pojazdy elektroniką najwyższej klasy. Auta z coraz to niższych półek posiadają system kontroli stanu kierowcy w standardzie! Już od dziesięciu lat Amerykańska część koncernu Daimler – Chrysler prowadzi badanie nad sennością kierowców aut zarówno ciężarowych jaki i osobowych. Według Generalnej Administracji Autostrad oraz Dróg Szybkiego Ruchu (The National Highway Traffic Safety Administration) w Stanach Zjednoczonych Ameryki Północnej każdego roku zasypianie za kierownicą jest przyczyną 1500 śmiertelnych wypadków, w których około 71000 osób jest rannych. Według ankiety przeprowadzonej przez Amerykańską Narodową Fundację Badania Snu (The National Sleep Foundation, in

Washington, D.C.) 62 procent dorosłych kierowców przyznało się do prowadzenia pojazdu będąc w stanie silnego zmęczenia spowodowanego brakiem snu. Aż 25% badanych twierdzi, że zna przynajmniej jedną osobę która spowodowała wypadek samochodowy poprzez zaśnięcie za kierownicą w ciągu ostatniego roku! Naukowcy z USA rozpoczęli badania nad stworzeniem ogólnego systemu koordynującego bezpieczne prowadzenie. System został podzielony na dwie grupy: pierwsza z nich to wszelkie działania mające na celu zminimalizowanie błędów popełnianych przez kierowcę. Powstały systemy kontroli trakcji, inteligentne światła drogowe, trwają badania nad systemami automatycznego prowadzenia się pojazdów. Przy autostradzie zainstalowane są nadajniki wysyłające cyfrową informację o położeniu samochodu względem drogi (system GPS wykorzystywany przez ludność cywilną ma za małą dokładność). Osobną gałęzią rozwoju mechanizmów zapewniających bezpieczeństwo jest czuwanie pojazdu nad kierowcą. Obecnie w USA można kupić urządzenie mniejsze od telefonu komórkowego, które zainstalowane na kierownicy samochodu monitoruje oczy kierowcy. Cyfrowa kamera przekazuje obraz twarzy kierowcy do niezwykle wydajnego procesora graficznego, który analizuje każdą klatkę filmu w czasie rzeczywistym. Jeżeli urządzenie wykryje stosunkowo duży poziom senności, włącza głośny alarm. Alarm podnoszony jest również na wypadek rozpoznania faktu zaśnięcia za kierownicą. Japońscy inżynierowie od kilku lat montują urządzenia monitorowania kondycji psychofizycznej kierowcy w samochodach osobowych „z górnej półki". Dużą zaletą Japońskich rozwiązań jest ich precyzyjne zestrojenie. Auto opuszczające fabrykę na tle konkurencji z zachodu pracuje zdecydowanie najlepiej. Mechanizmy najdokładniej odzwierciedlają faktyczny stan kierowcy. Lexus dopuszcza opcje monitorowania twarzy kierowcy pod kątem czuwania, czy osoba prowadząca pojazd skupiona jest na drodze przed nią. Można włączyć system, który przy prędkości powyżej 50 mil na godzinę podnosi alarm, jeżeli kierowca poparzy w lewo lub prawo dłużej niż dwie sekundy. W opinii autora pracy istnieje potrzeba rozwijania aktywnych systemów zwiększających bezpieczeństwo na drogach. Mechanizmy powinny być instalowane zarówno w samochodach ciężarowych, jak i osobowych. Komputer nie dopuszcza do sytuacji prowadzenia pojazdu przez nie wypoczętego kierowcę monitorując jego kondycję psychofizyczną.

Autor pracy pragnie wykazać jak stosunkowo nieduży nakład środków może zapobiec kilku tysiącom wypadków samochodowych na świecie.

Celem opisanej pracy jest zaprojektowanie oraz przetestowanie systemu aktywnego śledzenia oczu kierowcy, który na podstawie ruchów powiek kontroluje kierowcę samochodu i wnioskuje o włączeniu alarmu oraz jeśli działanie to jest nieskuteczne o aktywnym włączeniu systemu hamowania, zachowania trakcji samochodu, włączenie świateł awaryjnych itp.

Założeniem projektowanego systemu było, żeby był on:

- możliwie najbardziej odporny na zmienne warunki zewnętrzne, takie jak: pora dnia, przeźroczystość powietrza, ruchy kierowcy przysłaniające oczy,
- mechanizm powinien pracować niezmiennie od budowy anatomicznej różnych kierowców, raz zestrojony mechanizm prawidłowo rozpoznaje stan psychofizyczny kobiety, mężczyzny, osoby o małych oraz dużych oczach,
- posiadać opcję biernego zwiększania bezpieczeństwa (np. w przypadku, kiedy kierowca staje się senny, można zwiększyć głośność radia, zaproponować przerwę w podróży).

W ramach pracy zostały zaprojektowane wszystkie niezbędne algorytmy oraz zaimplementowany mechanizm aktywnego śledzenia oczu kierowcy wraz z modułem decyzyjnym w środowisku Matlab. Autor pracy główny nacisk kładzie na zastosowanie możliwie najmniej skomplikowanych algorytmów realizujących poszczególne etapy analizy ruchomego obrazu oraz podejmowania decyzji. System powinien w łatwy sposób być przenoszony na inne platformy sprzętowe (z innymi systemami instalowanymi w samochodzie). Szczegółowa analiza poprzedzona zebraniem stosunkowo dużej ilości danych pomiarowych przeprowadzana jest dla następujących etapów pracy:

- instalacja właściwej kamery rejestrującej twarz kierowcy samochodu,
- wstępna obróbka próbkowanego obrazu – przetwarzanie pojedynczej klatki filmu, filtracja niwelująca niedoskonałości kamery cyfrowej,
- uniezależnienie się od zakłóceń pochodzących od „środowiska", takich jak pora dnia, zmienne natężenie oraz kąt oświetlenia,
- analiza obrazu poszukiwanie elementów ujęcia pasujących do wzorca oka
- walidacja danych (system w najgorszym przypadku odpowiada za życie kierowcy),

- praca algorytmu analizy czujności kierowcy samochodu, rozpoznawanie stanów zmęczenia, senności oraz faktu zaśnięcia za kierownicą,
- budowa prostego mechanizmu decyzyjnego wspomagającego kierowcę, system podpowiada osobie prowadzącej samochód.

Szczegółowa analiza wyników algorytmu w środowisku laboratoryjnym powinna zostać skonfrontowana z rezultatami pracy mechanizmu „rzeczywistym świecie". System przede wszystkim musi być odporny na trudne do wyeliminowania zakłócenia, takie jak:

- częste, (pseudo) losowe zmiany natężenia oraz kąta padania światła na twarz kierowcy,
- zmienne oraz nieprzewidywalne tło,
- chwilowe, lub ciągłe przesłonięcie jednego z oczu kierowcy,

Z sytuacją zasypiania kierowcy podczas prowadzenia pojazdu najczęściej mamy do czynienia na autostradzie, lub drodze szybkiego ruchu, przeważnie w nocy. Samochody nadjeżdżające z przeciwka oświetlają na krótka chwilę twarz kierowcy stosunkowo mocnym promieniem światła o różnej długości fali (w zależności od żarówek: białe ksenony, żółte halogeny, etc). Mechanizmu musi być odporny na tego rodzaju zakłócenia. Za kierowcą, na tylnym siedzeniu może siedzieć pasażer, jego oczy nie mogą być źródłem analizy opisywanego mechanizmu. Prowadząc samochód, często wykonujemy wiele ruchów manipulatywnych wokół twarzy, nierzadko zdarzają się przesłonięcia oczu, algorytm decyzyjny powinien przewidzieć taka sytuację.
Projektowanie mechanizmu śledzenia oczu kierowcy samochodu jest nieco uproszczone w porównaniu do problematyki rozpoznawania obrazów dla przypadku ogólnego dzięki temu, że:

- komputer szuka dwóch obiektów w stałej, określonej odległości od siebie, znana jest ich przybliżona wielkość oraz dosyć dokładny kształt,
- oczy, klatka po klatce, przemieszczają się po obrazie ze ściśle określoną (stosunkowo małą) „prędkością poruszania" – żaden kierowca (w normalnych warunkach) nie jest w stanie przesunąć głowy o więcej niż kilka centymetrów w 1/25 sekundy!
- każda osoba ma inna wielkość oraz kształt oczu otwartych (zamkniętych), jednak długość trwania pomiarów w zupełności wystarcza, aby szybko nauczyć

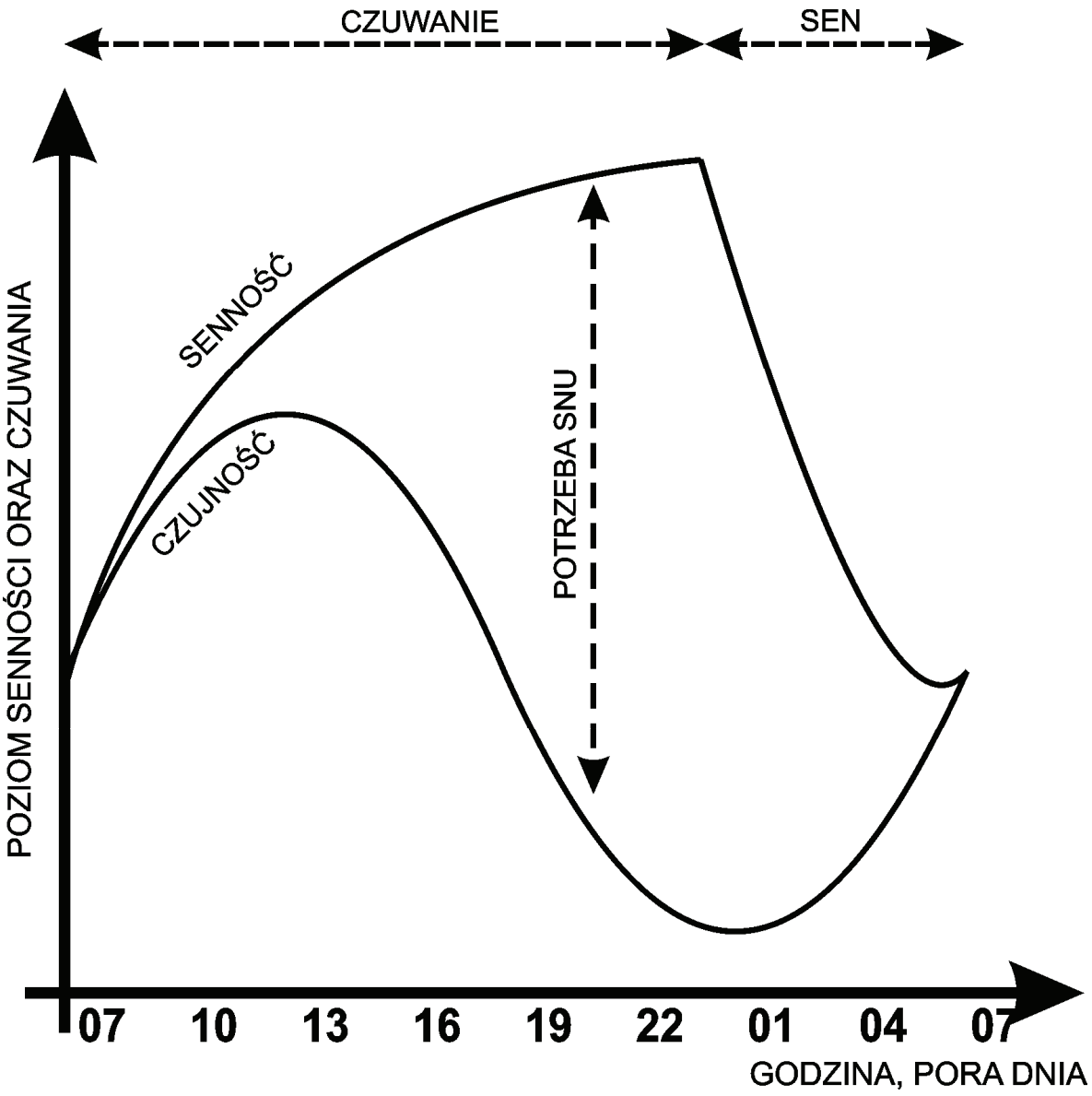
mechanizm, kiedy oczy są otwarte, kiedy zamknięte – ewentualne zaśnięcie za kierownicą następuje po kilkudziesięciu minutach od rozpoczęcia jazdy.

Wszystkie wymienione wyżej restrykcje stały się podstawą do zaimplementowania mechanizmu analizy stanu psychofizycznego kierowcy samochodu w środowisku Matlab. Aplikacja posłużyła głównie w celu przeprowadzanie badań efektywności skonstruowanego algorytmu opisanego w pracy. Analizy przeprowadzano dla bardzo dużej ilości danych. Aplikacja ma możliwość otworzenia wcześniej zarejestrowanego filmu. Po dokonaniu analizy, program zapisuje plik filmowy na dysk. Każdy plik można z łatwością badać „klatka po klatce" i wyciągać wnioski.

2. Opis fizjologii zasypiania kierowcy samochodu oraz działań asekuracyjnych.

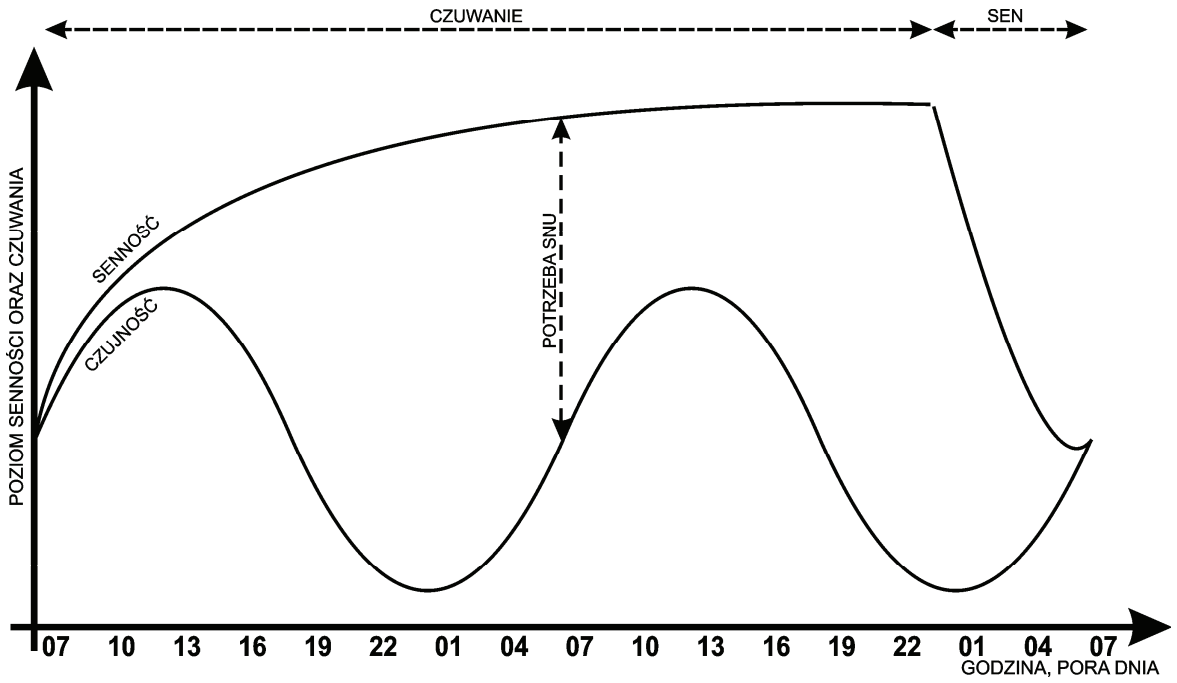
Hipnologia – wiedza o śnie człowieka zaczęła się prężnie rozwijać w latach 50-tych i 60-tych XX wieku. Opisano najistotniejsze fakty dotyczące fizjologii ludzkiego snu. Naukowcy przedstawiają kilka teorii związanych z zasypianiem, jednak większość z nich zgodnie dzieli ten proces na cztery fazy. Etap I i II – sen płytki oraz III i IV – sen głęboki. Zapis zmian w elektroencefalogramie (EEG) dla fazy I i IV jest identyczny i do złudzenia przypomina stan czuwania człowieka. Charakterystyczny objaw tego etapu to szybkie ruchy gałek ocznych (REM – rapid eye movements) widoczne pod zamkniętymi powiekami. W fazie REM człowiek przeżywa marzenia senne, przebudzenie się podczas trwania tej fazy umożliwia zapamiętanie tych marzeń. Radykalne skrócenie czasu snu powoduje ograniczenie trwania faz snu płytkiego (leki nasenne eliminują fazy REM).

Mechanizm regulacji snu człowieka nie jest dokładnie zbadany. Istnieje kilka modeli wzajemnych powiązań ścieżki zasypiania, współczesna Hypnologia niejako „forsuję" mechanizm zaproponowany przez szwajcarskiego neurofizjologa Aleksandra Borbely'ego [2]. Rysunek 1. wyjaśnia sposób analizy snu według naukowca. Idea modelu wyróżnia dwa czynniki regulujące wielkość „potrzeby" zaśnięcia. Po kilkugodzinnym śnie, człowiek budzi się wypoczęty, jest godzina 7:00 rano. Stopień senności jest niski, czujność organizmu stosunkowo wysoka. Wraz z upływem dnia stopień czujności zwiększa się, szczytową wartość osiąga około godziny 12:00 - 13:00 (dla osoby, która wstaje codziennie o 7:00 rano, zasypia o 23:00), a następnie spada, minimalną wartość uzyskując około 1:00 rano kolejnego dnia. Senność organizmu rośnie nie liniowo od momentu przebudzenia, aż do chwili zaśnięcia. Jako wielkość potrzeby snu człowieka w modelu dr Borbely'ego określa się różnicę pomiędzy wartością krzywej senności oraz czujności.



Rysunek 1. Analiza potrzeby snu wg dr Aleksandra Borbely'ego.

Rysunek 2. przedstawia proces tzw. „wymuszonej bezsenności". Przykładowo kierowca pojazdu decydując się na długą podróż , która rozpoczyna się w godzinach wieczornych, największą potrzebę snu odczuwa pomiędzy 24, a 6 rano. Jego organizm przywykł do regeneracyjnego snu. Zapewne wielu z doświadczonych kierowców doświadczyło sytuacji, kiedy na początku podróży rozpoczętej wcześnie rano (np. o 4:00) osoba prowadząca pojazd odczuwa mocną potrzebę snu. Około godziny 6:00 – 8:00 potrzeba snu niemalże znika, umożliwiając wielogodzinną podróż, trwającą do wieczora. Wtedy zmęczony długa trasą organizm ludzki „katowany" jest narastającą potrzebą snu, wynikającą z coraz większej różnicy pomiędzy krzywymi senności oraz czujności. Cykliczność krzywej czujności jest wynikiem pracy zegara biologicznego. Oscylacje charakterystyczne są dla każdego organizmu żywego [8].



Rysunek 2. Analiza potrzeby snu wg dr Aleksandra Borbely'ego – stan wymuszonej bezsenności.

Fizjologia zasypiania człowieka została bardzo dobrze zbadana przez naukowców na całym świecie. Opinie profesorów w większości przypadków pokrywają się ze sobą. Ciało człowieka, który staje się senny wysyła do otoczenia wiele sygnałów:

- coraz wolniejsze, monotonne ruchy ciała,
- długi czas reakcji na bodźce zewnętrzne,
- częste, powtarzające się ziewanie,
- zamykające się oczy, wraz z wzrostem senności, nasila się częstotliwość mrużenia, oraz rośnie czas zamknięcia oczu.

Zasypiający człowiek zaczyna mieć chaotyczne myśli, pojawia się apatia.
Stosunkowo łatwo jest rozpoznać kierowcę, który odczuwa dużą potrzebę snu, wystarczy zauważyć któryś z poniższych sygnałów:

- opadanie głowy,
- niekontrolowana zmiana toru jazdy, zmiana pasa ruchu,
- przeoczenie znaku stop/ustąp pierwszeństwa,
- niezatrzymanie się na czerwonym świetle,

- powolna „bezmyślna” jazda za pojazdem z przodu,
- wzrok wpatrzony w jeden punkt na drodze.

Dogłębna analiza fizjologii snu pozwala na wyciągnięcie wniosków dotyczących warunków pracy urządzenia wykrywającego zbliżającą się drzemkę kierowcy.

- mechanizm będzie pracował przeważnie w nocy,
- kierowca pokonuje długą, monotonną trasę,
- zasypianie najczęściej zdarza się na autostradach, lub drogach szybkiego ruchu, podróż przez teren zabudowany wymusza większą koncentracje uwagi.

Szczegółowe informacje analizujące fizjologie snu oraz modele zasypiania podczas kierowania pojazdami mechanicznymi można znaleźć w [8], [14], [15], [16].
Nocna praca urządzenia wymusza zastosowanie kamery pracującej w podczerwieni. Kierowca oświetlany jest diodami emitującymi światło IR. Element rejestrujący obraz otrzymuję monochromatyczną mapę pixeli, będącą ilustracją twarzą osoby prowadzącej pojazd. Kamera powinna być zainstalowana w taki sposób, aby nie zaburzać pracy kierowcy, oraz móc rejestrować mimikę jego twarzy. Autor pracy doświadczalnie wybrał najbardziej optymalne miejsce instalacji odbiornika IR – deska rozdzielcza po lewej stronie od kierownicy, na wysokości łączenia szyby pojazdu z elementami tapicerki.
Twarz kierowcy oświetlana jest od czasu do czasu przez samochodu nadjeżdżające z przeciwka, mechanizm musi szybko dopasowywać poziom jasności próbkowanego obrazu, lub uniezależnić się od tej wartości. Na jakość pracy algorytmu nie może wpływać pora dnia.
Zasypianie najczęściej zdarza się podczas podróżowania autostradami, twarz kierowcy jest niemalże nieruchoma, co stanowi duże uproszczenie dla algorytmu, dane otrzymywane na wyjściu są obarczone mniejszym błędem, komputer z większym prawdopodobieństwem wie, że obserwuje oczy kierowcy. Mechanizm w szczególnym przypadku odpowiada za ludzkie życie, więc algorytm musi poprawnie pracować w sytuacji, kiedy kierowca wykonuje ruchy głową oraz rękoma.
Zasadniczo autor pracy główny nacisk kładzie na rozpoznawanie dwóch szczególnych sytuacji. Algorytm powinien znać stopień zmęczenia kierowcy (mechanizm wie, kiedy kierowca jest senny), oraz wykrycie faktu zaśnięcia osoby prowadzącej pojazd za kierownicą.

Dla przypadku, kiedy urządzenie rozpozna narastający poziom zmęczenia kierowcy, system mógł by proponować następujące czynności poprawiające stan psychofizyczny osoby prowadzącej samochód:

- otwórz okno, włącz klimatyzację, przewietrz pojazd,
- włącz lub zwiększ głośność radia,
- weź małą przekąskę,
- zacznij rozmawiać, śpiewać,
- zatrzymaj samochód, wybierz się na spacer,
- wypij napój energetyczny, lub filiżankę kawy,
- zatrzymaj się i zdrzemnij przez min. 20 minut.

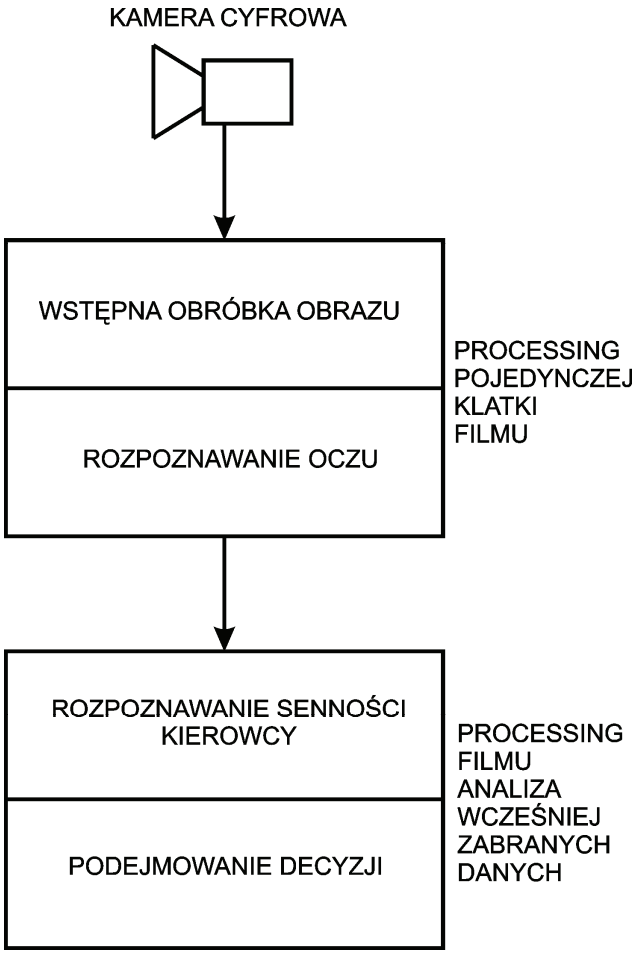
Autor pracy bazując na doświadczeniu zawodowych kierowców wybrał jedynie kilka punktów, które powinny być skutecznymi radami systemu zwiększającego bezpieczeństwo dla zmęczonego kierowcy. Kiedy człowiek zaczyna stawać się senny powinien zacząć rozmawiać z inną osobą, śpiewać, lub aktywnie słuchać radia. Jeżeli stan psychofizyczny kierowcy „pogarsza" się, system powinien proponować spożycie napoju energetycznego lub kawy (Napoje energetyczne potrafią przedłużyć stan czuwania kierowcy nawet do dwóch godzin, kawa w zależności od stosunku ilości filiżanek spożywanych codziennie do ilości kawy spożytej podczas podróży poprawia stan kierowcy na czas od piętnastu minut do dwóch godzin). Zdecydowanie najlepsze efekty „pozbywania" się zmęczenia kierowcy jest drzemka. Nawet 20 minut pozwala na przedłużenie podróży o kilka godzin! System proponuję drzemkę wyjątkowo zmęczonym kierowcom.
W przypadku wykrycia faktu zaśnięcia za kierownicą, system wysyła sygnał do centralnego komputera zainstalowanego w samochodzie, powodując bezpieczne zatrzymanie pojazdu, włączenie świateł awaryjnych oraz obudzenie kierowcy zwiększającą się stopniowo głośnością radia.

3. Opis algorytmu wykrywania oraz reagowania na sen kierowcy.

Autor pracy podzielił proces tworzenia algorytmu na trzy główne etapy, są to:

1. projekt oraz realizacja części hardwarowej urządzenia,
2. implementacja programu obróbki oraz analizy obrazu,
3. ustanowienie zasad pracy części decyzyjnej algorytmu.

Rysunek 3. przedstawia schemat blokowy urządzenia, na którym można wyraźnie wyróżnić wspomniane etapy pracy mechanizmu badającego stan psychofizyczny kierowcy.



Rysunek 3. Schemat blokowy mechanizmu badającego kondycje kierowcy.

3.1 Elementy hardwareowe.

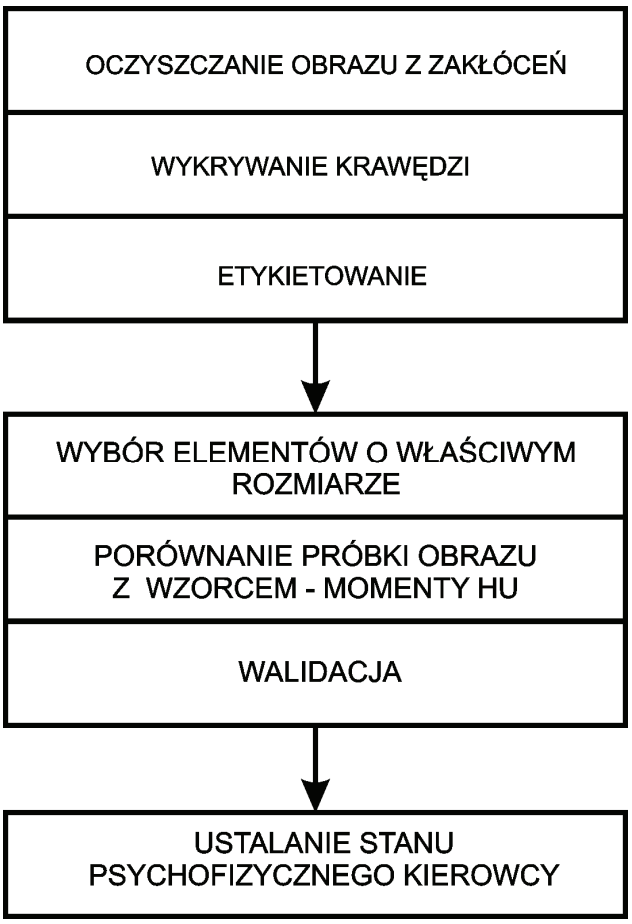
Autor projektu umieścił kamerę cyfrową pracującą w podczerwieni w lewej części deski rozdzielczej, na wysokości łączenia się szyby pojazdu z elementami tapicerki. Rysunek 4 pokazuje przykładowe zdjęcie twarzy kierowcy pochodzące z tak umieszczonej kamery.

Rysunek 4. Przykładowe zdjęcie wykonane kamerą zainstalowaną w samochodzie - twarz kierowcy.

Kamera, która przy minimalnej wymaganej rozdzielczości 640x480 pixeli nie dokonuje stratnej kompresji obrazu wysyła cyfrowy obraz do komputera zamontowanego w samochodzie (do celów badawczych autor pracy wykorzystał komputer klasy PC). Procesor, który realizuje program algorytmu rozpoznawania stanu kierowcy współpracuje z centralnym komputerem pojazdu, wysyła informacje o ewentualnym awaryjnym hamowaniu, zwiększenie głośności radia, etc.

3.2 Software – opis aplikacji realizującej algorytm rozpoznawania kondycji kierowcy.

Program analizy obrazu pochodzącego z kamery cyfrowej wraz z wszystkimi funkcjami sterującymi został przedstawiony na listingu 1. Aplikacja napisana w środowisku Matlab pozwala na szybką oraz wygodną analizę cząstkowych wyników pracy każdego z pod-etapów pracy algorytmu. Język Matlab pozwala na szybkie modyfikowanie kodu programu, co jest niezwykle istotne podczas pisania kodu, w którym autor eksperymentuje z różnymi algorytmami labeling'u, rozpoznawania obiektów, etc. Dogłębną analizę wyników pracy urządzenia gwarantuje specjalny sposób konstrukcji wejścia oraz wyjścia algorytmu, program pobiera zarejestrowany wcześniej film, następnie przelicza każdą klatkę oraz zapisuje na wyjściu film z naniesionymi rezultatami pracy. Rysunek 5. przedstawia ogólny schemat blokowy aplikacji.



Rysunek 5. Schemat blokowy aplikacji.

Trzy główne części programu to:

- blok dopasowywania klatki pojedynczego ujęcia na potrzeby algorytmu wyszukiwania oczu,
- rozpoznawanie oczu w pojedynczej klatce filmu, oraz walidacja wyników,
- wyszukiwanie oznak pogarszającego się stanu psychofizycznego kierowcy wskazującego na senność.

3.2.1 Pre-processing pojedynczej klatki filmu.

Wstępna obróbka pojedynczej klatki filmu składa się z następujących etapów:

1. oczyszczenie obrazu z szumów kamery,
2. mocne wykrywanie obrysów (krawędzi) obiektów,
3. indeksacja obiektów w obrazie.

Etap 1. Oczyszczanie obrazu z szumów kamery

Aplikacja w napisana w języku Matlab pobiera na wejściu kolorowy film. Pierwszych etapem pracy jest zamiana do na obraz w odcieniach skali szarości o wartościach od 0 do 255 (materiał do badań pobierany jest w dzień, aplikacja symuluje próbkowanie obrazu za pomocą monochromatycznej kamery pracującej w podczerwieni). Cyfrowe urządzenia rejestrujące, które pracują w warunkach słabego oświetlenia (pomieszczenia zamknięte, samochód) wprowadzają stosunkowo dużo szumu. Autor pracy eksperymentalnie dobrał najbardziej optymalny filtr typu blur. Maska 5x5 pixeli składająca się z „jedynek" przechodzi przez obraz eliminując wszelkie zanieczyszczenia oraz sprawia, że krawędzie pojedynczych obiektów stają się mniej „postrzępione". Rysunek 6 przedstawia pojedynczą klatkę filmu po filtracji wstępnej. Więcej informacji dotyczących technicznych szczegółów opisywanych przekształceń można znaleźć w pracach [4], [11-13], [17-23].

Rysunek 6. Klatka filmu po wstępnej filtracji.

Etap 2. Wykrywanie krawędzi – obrysów obiektów.

Obraz normalizowany liniowo trafia na filtr wykrywający krawędzie typu outline. Maska 7x7 przedstawiona jest na rysunku 7. Cechą charakterystyczną filtru jest to, że suma elementów macierzy 7x7 wynosi 0. Wszelkie krawędzie statystycznie mają „grubość” 7 pixeli i są w kolorze białym, kolor tła – czarny (wartość 0).

1	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	0	0	0	1
1	0	0	-24	0	0	1
1	0	0	0	0	0	1
1	0	0	0	0	0	1
1	1	1	1	1	1	1

Rysunek 7. Maska konwolucji 7x7 – filtr typu outline.

Rysunek 8. Binarny obraz po filtrze outline 7x7.

Etap 3. Binaryzacja.

Pojedyncza klatka filmu jest binaryzowana z doświadczalnie wybranym progiem 100. Tak przetworzona mapa pixeli trafia do funkcji indeksującej wszystkie pojedyncze obiekty w obrazie. Rysunek 8 prezentuje binarny obraz przefiltrowany maska outline 7x7. Pierwszy napotkany obiekt otrzymuje etykietę 1, kolejny 2 itd. Tło ma zawsze wartość 0. Autor pracy zauważył, że obraz przetworzony filtrem blur oraz outline zawiera znacznie mniejsza liczbę obiektów, niż obraz traktowany jedynie filtrem outline. Program w skrajnych przypadkach szum traktuje jako krawędź nowego obiektu, znacznie zmniejszając wydajność pierwszego etapu rozpoznawania oczu. Kombinacja filtrów 5x5 blur oraz 7x7 outline eliminuje z obrazu oczy pasażerów siedzących na tylnym siedzeniu pojazdu – obiekty są na tyle małe, że po przetworzeniu komputer rejestruje je jako kropki o wymiarze średnio 7x9 pixeli.

3.2.2 Rozpoznawanie oczu.

Na mapie bitowej przetworzonej w taki sposób możliwe jest rozpoznawanie obrazów**.** Odnajdowanie oczu przebiega trzy etapowo (zgodnie ze schematem blokowym – rysunek 5)

1. wybór elementów o właściwym rozmiarze,
2. porównanie próbki obrazu z wzorcem,
3. walidacja wyników.

Etap 1. Wyszukanie elementów o odpowiednim rozmiarze.

Obraz po etykietowaniu kierowany jest na pierwszy etap pracy funkcji rozpoznającej oczy kierowcy. Program wybiera tylko te obiekty, które spełniają warunek „odpowiedniego rozmiaru”. Podczas trwania badań autor pracy zaobserwował, że dla obrazu 640x480 pixeli najmniejsze zarejestrowane oczy mają wymiar 30x10 pixeli. Największe oczy, które można zarejestrować (twarz kierowcy najbliżej kamery) mieszczą się zwykle w prostokącie 70x30. Obiekty zostaną zakwalifikowane do kolejnego etapu wyszukiwania oczu, jeżeli ich szerokość mieści się w przedziale <30,70> i ich wysokość należy do przedziału <10,30>. Obraz testowy składający się z pięciu zmontowanych twarzy wraz z zaznaczonymi obiektami spełniającymi kryterium właściwego rozmiaru przedstawiono na rysunku 7 (do obrazu domontowano jedno oko w prawym dolnym rogu, obiekt jest pomocny przy walidacji wyników pracy algorytmu). Dla tak wybranych elementów obrazu

program oblicza wartości momentów tych obiektów. Dodatkowo tworzone są tablicę z informacją o skrajnych pikselach badanego elementu (wartości pixeli najbardziej wysuniętych na dół/górę oraz prawo/lewo), oraz liczone są wartości momentów centralnych. Znając wartości momentów centralnych program generuje zestawienie funkcji „Hu" [1], [3], [5-7], [9-10].

Rysunek 7. Obiekty po pierwszym etapie klasyfikacji – wybór elementów o właściwym rozmiarze.

Etap 2. Porównanie próbki obrazu z wzorcem.

Kolejnym etapem klasyfikacji obiektów jest filtrowanie pod względem podobieństwa do wcześniej zapisanych wzorców. Autor pracy przygotował wzorce otwartego oka lewego, prawego oraz oka zamkniętego. Program porównuje wartości funkcji Hu wzorów z analizowanym elementem pojedynczej klatki. W tabeli 1 przedstawiono wyniki badań obrazu „Pmix" – rysunek 7. zawierającego zmontowane jedenaście różnych oczu. dziesięć z nich tworzy pary. Obraz „Pmix" analizowano pod względem jakości rozpoznawania oczu w stosunku do różnych nastaw tolerancji podobieństwa wzorca do badanego obiektu. Autor pracy pozostawił sobie dwie zmienne, którymi można zmieniać czułość programu.

Funkcja „Hu” składa się z sześciu stopni (sześć skalarów). Pierwszą zmienną jest liczba z zakresu 0-6 – ile węzłów funkcji „Hu” wzorca jest podobna do węzłów analizowanego obiektu. Przykładowo jeżeli zmienna ma wartość „4” – cztery stopnie „Hu” wzorca muszą pokrywać się z czterema stopniami próbki obrazu. Stopień pierwszy wzoru może pokrywać się wyłącznie ze stopniem pierwszym badanego obrazu, stopień drugi – z drugim, etc.

Drugą zmienną regulacyjną jest wartość tolerancji, z jaką przyrównywane są poszczególne stopnie „Hu”. Przykładowo, jeżeli wartość tolerancji wynosi 10, badany węzeł „Hu” próbki obrazu może różnić się dziesięciokrotnie od wzorca (jeżeli wzór wynosi 1, dopuszczalne wartości obrazu są w przedziale <0.1 , 10>).

Tabela 1. Badanie ilości węzłów „Hu” wzorca, które muszą być podobne do węzłów próbki, wraz z dopuszczalnymi tolerancjami.

lp	Minimalna ilość zgodnych węzłów Hu	Tolerancja zgodności	Ilość rozpoznanych oczu	Faktyczna ilość oczu	Błąd	Uwagi
1	2	1	22	11	11	
2	2	3	24	11	13	
3	2	10	24	11	13	
4	3	1	8	11	-3	nie wszystkie oczy
5	3	3	19	11	8	ok.
6	3	10	25	11	14	
7	4	1	9	11	-2	nie wszystkie oczy
8	4	3	18	11	7	nie wszystkie oczy
9	4	10	21	11	10	

Autor pracy na podstawie doświadczeń, których wyniki pokazane są w tabeli 1, postanowił, że minimalna liczba podobnych stopni „Hu” powinna wynieść 3, natomiast dopuszczalna tolerancja – różnica pomiędzy „Hu” wzorca oraz obiektu nie może być większa niż 3 (pomiar 5). W pomiarze 8. algorytm rozpoznał 18 obiektów, które potencjalnie są oczami, jednak komputer nie wskazał wszystkich oczu prawidłowo, dwa z nich (prawa – górna twarz) nie zostały poprawnie zakwalifikowane.

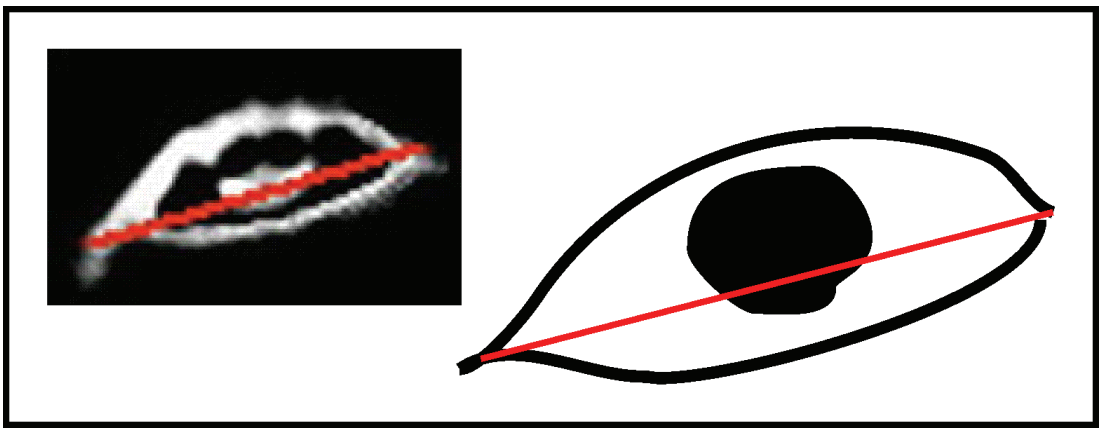
Rozpoznawanie obrazów przy pomocy funkcji „Hu” w opinii autora jest najbardziej optymalnym algorytmem wyszukiwania oczy kierowcy pojazdu. Analiza momentów „Hu” posiada wiele zalet, które znacząco wpływają na prawidłową pracę całego programu:

- wartości „Hu” są niezależne od rozmiarów porównywanych obrazów – skalowalność,
- funkcje „Hu” są niezmienne dla obrazów po translacji o wektor oraz obrocie wokół dowolnej osi płaszczyzny X:Y,
- momenty „Hu” są dosyć podobne dla obiektów obracanych wokół osi Z – transformata 3D!
- aplikacja do każdego badanego obiektu otrzymuje wektor sześciu liczb, które różnią się znacząco (nawet kilkanaście tysięcy razy!) dla różnych próbek obrazu.

Dzięki tym zaletom, algorytm, aby rozpoznać oko lewe, które jest w dowolnym miejscu na ekranie, pod dowolnym kątem, zmienia w czasie swój rozmiar (odległość twarzy od kamery zmianie się nieustannie) oraz jest transformowane w niewielkim stopniu wokół osi Z (kierowca obraca głową w lewo i prawo) - potrzebuje tylko jeden wzorzec oka lewego. Dodatkowo wzorzec ten może być zapisany pod dowolnym kątem i w dowolnym rozmiarze. Na rysunku 8 aplikacja zaznaczyła wszystkie elementy obrazu „Pmix” pasujące do wzorca porównując momenty „Hu”.

Rysunek 8. Elementy obrazu pasujące do wzorca oka – klasyfikacja przy pomocy funkcji „Hu”.

Na tym etapie rozpoznawania oczu, przetestowano kilka algorytmów, na uwagę zasługuje mechanizm szukania oczu w obrazie w którym nie wykryto krawędzi. Monochromatyczna mapa bitowa (po filtrze blur) na początku binaryzowana jest z niskim progiem (dobry kontrast dla ciemnych obrazów, jasne znikają – w samochodzie jest z reguły ciemno), a następnie etykietuje się każdy z dostępnych obiektów. Program wyszukuje tylko te obiekty, które spełniają założony wcześniej rozmiar, a następnie wyznacza skrajne punkty tych próbek (np. lewy dolny róg, prawy dolny, etc.) Aplikacja zakłada, że wszystkie próbki są oczami i rysuje linie poziomą od lewego kącika do prawego. Następnie badana jest sekwencja zmiany kolorów obrazu pod tą prostą. Analizując „idealne oko” prawidłowy wynik dla oka otwartego to czarny-biały-czarny-biały-czarny – rysunek 9.



Rysunek 9. Jeden z przykładów rozpoznawania oczu przy pomocy algorytmu sekwencji zmian.

Każdy kolor powinien wystąpić przynajmniej przez 3 pixele, nie więcej niż 20 pixeli. Jeżeli algorytm nie znajdzie dwóch oczu, powtarza się całą procedurę dla obrazu binaryzowanego z większym progiem. Badania pokazują, że najlepiej binaryzować w zakresie jasności od 30 do 220, z krokiem 10. Większy krok przyśpiesza znacząco pracę mechanizmu, jednak obniża jego skuteczność, w większości przypadków oko ludzkie oświetlone jest jednorodnym, słabym światłem – posiada niski kontrast. Algorytm posiada zalety niezależności od rozmiaru, miejsca na ekranie, poziomu oświetlenia, obrotu względem osi XYZ. Mechanizm posiada jednak bardzo dużą wadę, nie potrafi jednoznacznie odnaleźć zamkniętych oczu, traktuje je jako proste, czarne linie, wybierając wszystkie poziome linie o odpowiednim rozmiarze (brwi, usta, elementy tapicerki, włosów, oraz ubrania kierowcy).

Autor pracy po konsultacji z opiekunem pracy dr Adrianem Horzykiem ostatecznie zdecydowali się wykorzystać algorytm rozpoznawania obrazów przy pomocy momentów „Hu".

Rysunek 8 przedstawia przykładowy obraz analizowany pod kątem wyszukiwania oczu metodą funkcji „Hu". Jak łatwo zauważyć, aplikacja posiadając dosyć „luźne" tolerancje odnalazła kilka obiektów, które potencjalnie są oczami (otwartymi lub zamkniętymi).

Etap 3. Walidacja wyników (aplikacja w szczególnym przypadku odpowiada za życie kierowcy – wobec tego wyniki powinny być obarczone możliwie najmniejszym błędem)

Taka mapa bitowa trafia na kolejny etap analizy – walidację danych. Algorytm mechanizmu sprawdzania poprawności odnalezionych oczu wraz z komentarzem można zapisać w punktach:

1. upewnienie się, że badane obiekty nie są brwiami – czasami komputer rozpozna brew jako zamknięte oko – jeżeli blisko pod obiektem nie występuje żaden inny obiekt (potencjalne oko) to obraz nie jest brwią, jeżeli obraz zostanie zakwalifikowany jako brew – odpada ze zbioru dostępnych potencjalnych oczu,
2. wybranie dowolnej pary obiektów - program bada wszystkie możliwe pary, złożoność obliczeniowa jest niska, ponieważ do dyspozycji jest na ogół od dwóch do sześciu obrazów,
3. Sprawdź, czy obiekty są na podobnej wysokości – kierowca trzyma głowę z reguły poziomo,
4. sprawdzenie, czy zachowany jest właściwy rozstaw oczu – obiekty rozsunięte w odległości od 50 do 109 pixeli w poziomie,
5. jeżeli znaleziono więcej niż jedną parę oczu, program wybierze tę, która jest najbliżej oczu z poprzedniej klatki filmu (dla pierwszej klatki – najbliżej środka ekranu).

Dodatkowo na algorytm narzucone są restrykcje:

- jeżeli nie zostanie znaleziona żadna para oczu, aplikacja szuka obiektów podobnych z luźniejszymi tolerancjami rozmiarów dopuszcza się nieco mniejsze obiekty (czasami oko jest słabo oświetlone – robi się małe po filtrach), tylko dwa momenty „Hu" wzorca musza pasować do badanego oka (jeżeli kierowca siedzi daleko od kamery i patrzy w lewo lub prawo, tylko dwa momenty będą podobne),
- jeżeli oczy znalezione na aktualnej klatce filmu są przesunięte o więcej jak 100 pixeli w stosunku do oczu z poprzedniej klatki, program nie bierze tej klatki pod uwagę w dalszej analizie filmu.

Rysunek 10 oraz rysunek 11 przedstawiają obrazy, które są wynikiem pracy algorytmu wyszukującego oczy.

Rysunek 10, 11. Wyniki pracy algorytmu wyszukującego oczy.

3.3 Wyszukiwanie oznak pogarszającego się stanu psychofizycznego kierowcy.

Kolejnym etapem pracy całego mechanizmu analizy stanu psychofizycznego kierowcy jest badanie stopnia otwarcia oczu. Częste zamykanie oczu jest jedna z oznak senności organizmu. Mruganie jest charakterystyczne dla każdego ssaka, trwa zwykle jedną trzecią część sekundy i występuje średnio raz na piętnaście sekund. Człowiek, który staje się senny, mruga znacznie częściej oraz czas zamknięcia powieki jest znacznie dłuższy (trwa nawet 2 sekundy). Częstość oraz czas trwania zamykania oczu wzrasta wprost proporcjonalnie do poziomu zmęczenia kierowcy. Ustalanie kondycji kierowcy przebiega trzy etapowo:

- mechanizm oblicza „stopień otwarcia oczu" w aktualnie przetwarzanej klatce filmu,
- algorytm wyznacza przeciętne wielkości zamkniętych oraz otwartych oczu kierowcy,
- zaprojektowana metoda decyduje, czy oczy zawarte w aktualnie obrabianym obrazie są otwarte czy zamknięte.

3.3.1 Odnajdowanie poziomu otwarcia oczu w aktualnie przetwarzanej klatce filmu.

Aplikacja analizuje każde oko pod względem poziomu jego otwarcia. Na środku oraz w odległościach równych 1/8 szerokości oczu, rysowane są pionowe linie, które mierzą poziom otwarcia oczu. Rysunek 10 przedstawia oczy wraz z odcinkami pomiarowymi. Im większa jest sumaryczna długość tych linii - tym poziom otwarcia oka jest większy. Punkty odcinka mogą przebiegać tylko po obiekcie, w ten sposób unika się mierzenia wysokości oka – jak pokazują doświadczenia liczenie odległości od dwóch skrajnych punktów (górny i dolny w połowie szerokości) daje gorsze, zafałszowane efekty, zwłaszcza dla oczu częściowo zamkniętych.

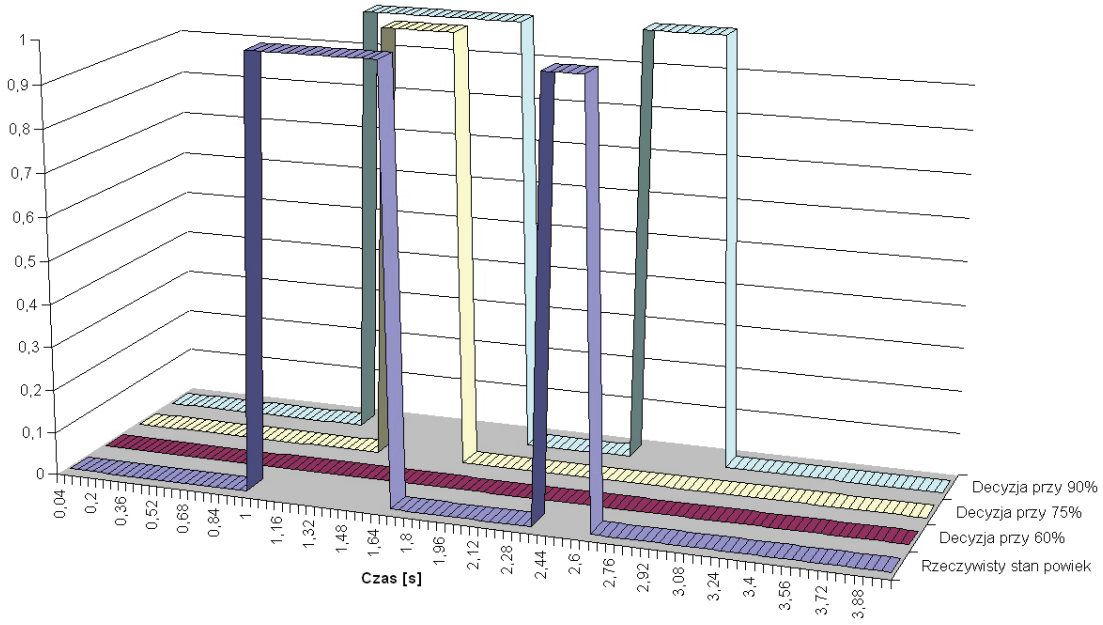
3.3.2 Nauka poziomów zamknięcia oraz otwarcia powiek.

Tak zebrana informacja o stopniu otwarcia oczu zapisywana jest w buforze filtru. Doświadczenia pokazują, że najbardziej optymalny jest filtr, który zapamiętuje dane z ośmiu sekund trwania filmu (200 klatek, przy 25 f/sec). Uśredniony bufor służy jako informacja: jaka jest przeciętna wielkość otwartych oczu kierowcy – informacja wykorzystywana jest jako zmienna referencyjna do wykrywania oczu zamknięcia oczu. Różni kierowcy mają oczy różnej wielkości, aplikacja uczy się, jaka jest wielkość otwartych oczu osoby prowadzącej pojazd. Badania pokazują, że do rozpoznania zamknięcia oczu, nie należy brać „czystej" wielkości oczu z aktualnej klatki filmu, wartość często się zmienia w różnych przedziałach, czasami zdarzają się jedno-klatkowe duże odchylenia od normy.

3.3.3 Podejmowanie decyzji w obrębie jednego obrazu – oczy zamknięte lub otwarte.

Zaproponowana metoda porównuje przeciętną wielkość oczu z filtru ośmiosekundowego z wartością przefiltrowaną w buforze pół-sekundowym (12 klatek – ten sam filtr, aplikacja bierze wartość średnią z ostatnich 12 klatek). Taka filtracja ma jeszcze jedną niezwykle istotną zaletę – eliminuje wpływ codziennego mrugania charakterystycznego dla każdej osoby – niezależnego od stopnia senności. Dwie liczby porównywane są ze sobą, na tej podstawie podejmowana jest decyzja, czy oczy są zamknięte, czy otwarte. Rysunek 13 pokazuje różne nastawy głównej zmiennej mającej wpływ na decyzje. Kolorem fioletowym zaznaczono aktualny stan powiek kierowcy (nie filtrowany), gdzie wartość 0 oznacza, że w danej klatce oczy są otwarte, 1 - oczy zamknięte. W kolejnych seriach przedstawiono wyniki pracy mechanizmu decyzyjnego. Odfiltrowana wartość poziomu otwarcia oczy porównywana jest kolejno z 60%, 75% oraz 90% przeciętnej wartości stopnia otwarcia oczy danego kierowcy. Ostatecznie autor pracy zdecydował się kontynuować badania dla zmiennej równej 75%, jeżeli aktualny poziom otwarcia oczu jest mniejszy bądź równy 75% wartości typowej danego kierowcy, aplikacja uznaje, że oczy są zamknięte. Filtr przepuszcza jedynie takie zamknięcia powiek, których czas trwania jest stosunkowo długi – mechanizm odrzuca krótki mrugnięcia. Niezwykle istotnym warunkiem, jest fakt, iż obydwoje oczy muszą mieć stopień otwarcie mniejszy lub równy 75% przeciętnej wartości, aby zamknięcie powiek traktowane było jako oznaka

zbliżającego się snu kierowcy. Badania pokazują, że zmienna wyjściowa powinna mieć charakter binarny (True/False) nie należy mierzyć stopnia otwarcia oczu w sposób liniowy – ciągły, kierowcy często mrużą lekko oczy zmieniając kierunek jazdy (jazda pod słońce), auta nadjeżdżające z przeciwka oślepiając kierowcę również powodują lekkie mrużenie oczu.



Rysunek 13. Badanie pracy mechanizmu decydującego, czy oczy są otwarte, czy zamknięte – cztery sekundy.

4. Zbieranie wyników pracy systemu, analiza działania.

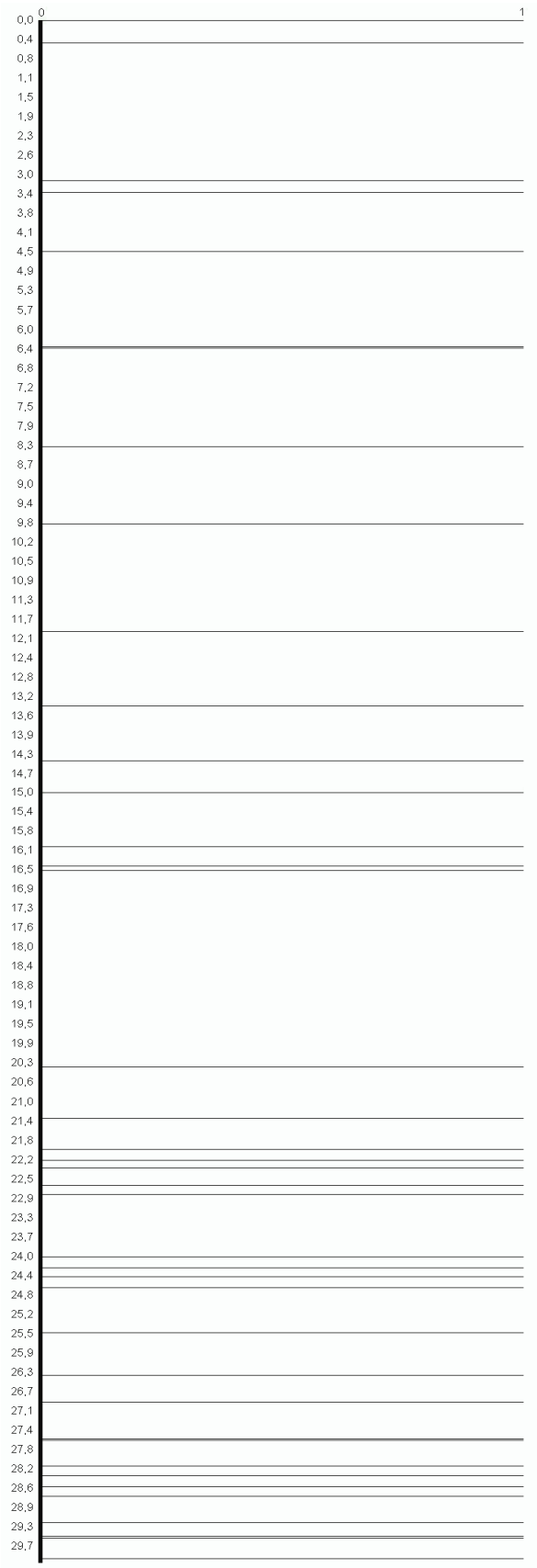
Dysponując mechanizmem, który potrafi dla każdej klatki filmu jednoznacznie wskazać, czy oczy są otwarte, czy zamknięte, przeprowadzono analizę filmu zarejestrowanego w przez autora pracy. Dbając o bezpieczeństwo, kierowca po przejechaniu trasy Kopenhaga – Tarnów (1300 kilometrów, 4 przerwy: prom 45 minut, tankowanie 15 minut, granica z Polska 15 minut oraz tankowanie 45 minut, jazda nocą) usiadł w zatrzymanym samochodzie symulując dalszą jazdę przez 30 minut. Podczas 30 minut nie udało się zarejestrować momentu zaśnięcia kierowcy, wyraźnie widać jednak narastający poziom senności. Rysunek 14 przedstawia wykres częstości zamykania oczu w skali 30 minutowej, poziom 0 oznacza, że oczy są otwarte, poziom 1 – oczy zamknięte. Rysunek 15 przedstawia pierwszą oraz ostatnią minutę filmu, wyraźnie widać dłuższy czas zamknięcia oczu podczas pojedynczego mrużenia. Dzięki zastosowaniu filtracji, każde zamknięcie oczu rejestrowane jest jako prostokątny skok sygnału trwający określony czas, bez przerw oraz zakłóceń. Proces zbierania wyników pracy algorytmu można podzielić na trzy etapy:

- wykreowanie ogólnej idei – koncepcji pracy podprogramu zbierającego oraz agregującego dane pochodzące z poprzedniej warstwy aplikacji – wykrywania zamkniętych oczu,
- tworzenie funkcji opisującej stan psychofizyczny kierowcy,
- doświadczalne dobranie ostatecznych nastaw prawidłowej pracy algorytmu.

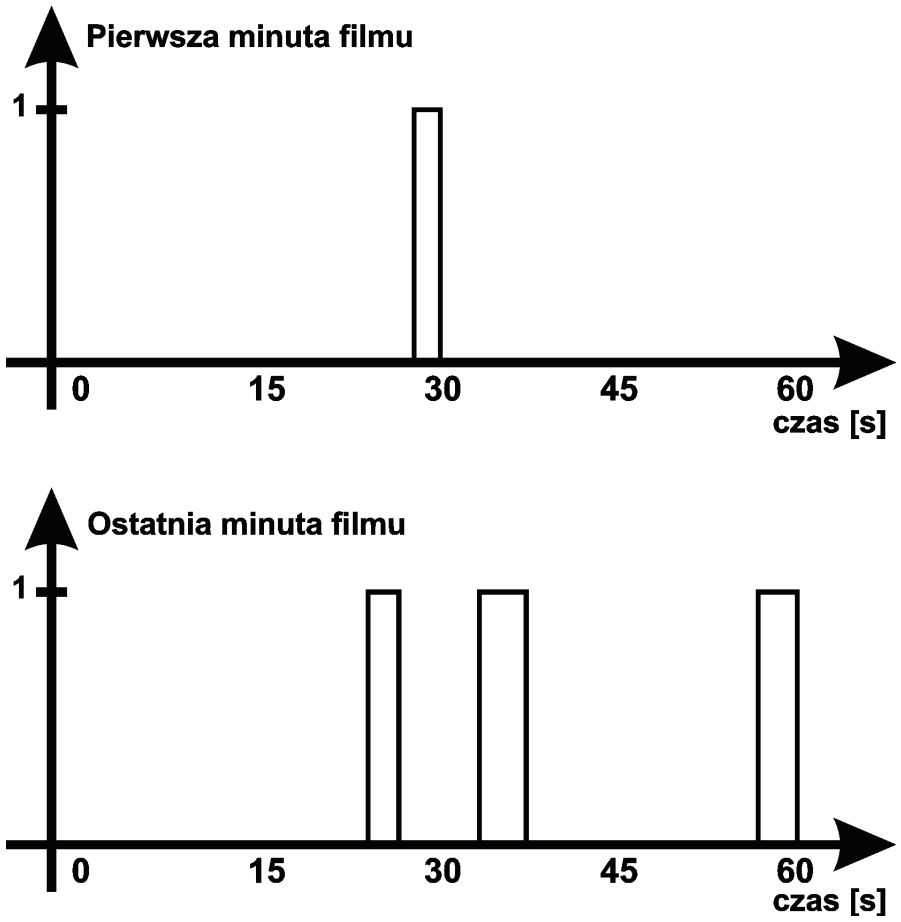
4.1 Idea pracy mechanizmu zbierającego oraz agregującego dane.

Analizując rysunki 14 i 15 można zapisać reguły, którymi powinien kierować się blok ustalania stanu psychofizycznego kierowcy:

- dane wejściowe binarne, 0 – oczy otwarte, 1 – oczy zamknięte,
- dane są odfiltrowane, aplikacja na tym etapie zlicza jedynie zamknięcia powiek towarzyszące zasypianiu, bez szumów, przymykanie jednego oka, etc.
- istotnym parametrem jest czas zamknięcia oczu, jeżeli kierowca jest bardziej senny, czas znacząco się wydłuża,
- algorytm musi dopuszczać poprawę stanu kierowcy, np. po spożyciu napoju energetycznego, etc.,
- wyjście algorytmu powinno możliwie jednoznacznie opisywać kondycje kierowcy.



Rysunek 14. Wykres częstości zamykania oczu - film 30 minut.



Rysunek 15. Pierwsza i ostatnia minuta filmu – porównanie czasu zamknięcia powiek.

W opinii autora pracy konieczne jest stworzenie mechanizmu realizującego powyższe postulaty, wyjście algorytmu powinno być skalarem, który informuje otoczenie o stanie psychofizycznym osoby prowadzącej pojazd.

4.2 Tworzenie funkcji opisującej stan kierowcy.

Podczas przeprowadzania badań, autor w porozumieniu z opiekunem pracy stworzył funkcję, której wartość jest wyznacznikiem kondycji osoby prowadzącej pojazd. Funkcja zależna jest od binarnej zmiennej „o”. Jeżeli oczy są otwarte, zmienna „o” przyjmuje wartość 0, jeżeli są zamknięte, zmienna ma wartość 1.

Wzór 1 przedstawia ogólną postać funkcji:

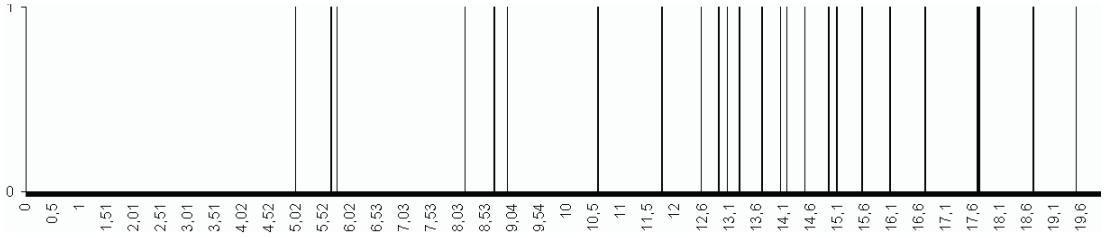
$$\mathbf{F(o,t) = o*T1 - \sim oT2}$$

Wzór 1. Postać ogólna funkcji opisującej kondycje kierowcy.

Parametry T1 oraz T2 to stałe czasowe mające wpływ na dynamikę zmian funkcji F. Parametr T2 przyjmuję stałą, niską wartość i odpowiada za „nadążaniem" funkcji za poprawiającym się stanem kierowcy. Jeżeli kierowca nie ma zamkniętych oczu, funkcja F stale obniża nieznacznie swoją wartość. Wartość T1 zmienia się w czasie, kiedy oczy kierowcy są zamknięte. W pierwszej klatce – kiedy wykryto mrużenie oczu wartość T1 jest stosunkowo niska, jednak znacznie większa niż wartość T2. Jeżeli mrużeniu oczu trwa kilkanaście klatek, wartość T1 narasta do pewnego poziomu.

4.3 Doświadczalne dobranie właściwych nastaw mechanizmu rozpoznającego stan psychofizyczny kierowcy.

Podczas tworzenie mechanizmu analizy kondycji kierowcy przeprowadzono kilkanaście doświadczeń mających na celu odnalezienie właściwych nastaw mechanizmu interpretacji funkcji decyzyjnej. Zdaniem autora pracy konieczne jest opisanie trzech szczególnych przypadków, w których znalazł się algorytm dokonujący analizy filmu „zasypiającego kierowcy". We wszystkich doświadczeniach badano dwudziestominutowy film zarejestrowany w godzinę po rejestracji filmu opisywanego wcześniej (rysunek 14), pomiędzy sesjami kierowca spożył napój energetyczny. Momenty występowania otwartych oraz zamkniętych oczu kierowcy pokazano na rysunku 16. Rysunek 16 uwzględnia czas zamknięcia powiek. Regulując parametry T1 oraz T2 próbowano znaleźć funkcję, która optymalnie odzwierciedla kondycje kierowcy.



Rysunek 16. Analiza filmu 20 minut.

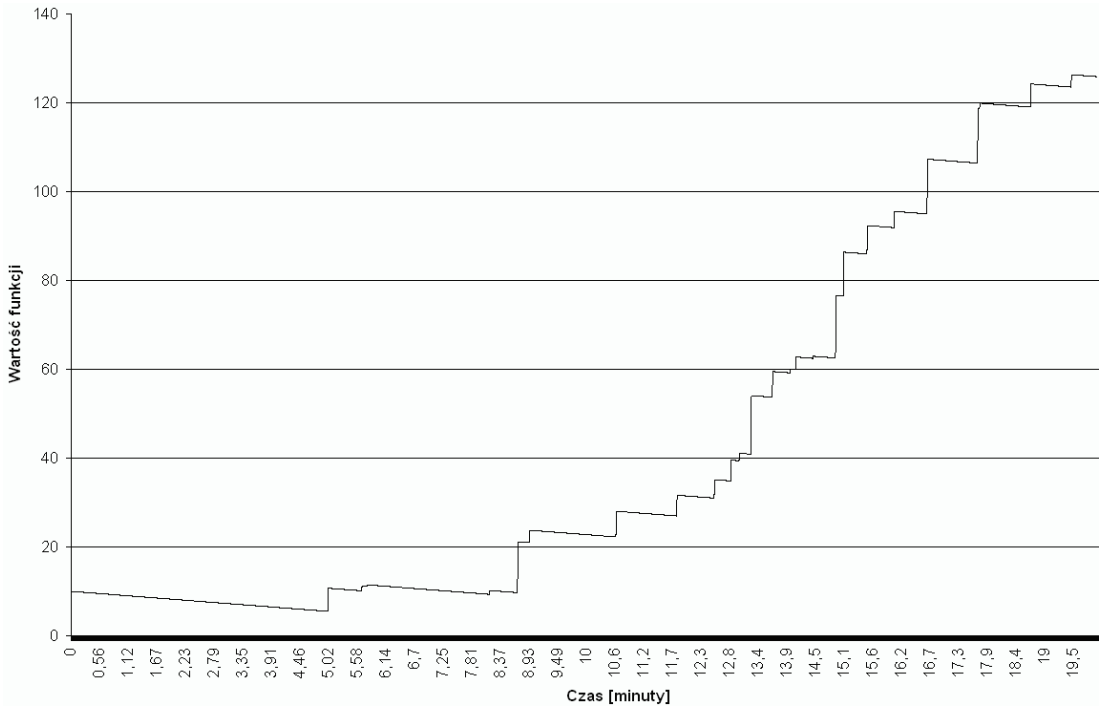
4.3.1 Przeprowadzenie badań.

Doświadczenia mają na celu takie dobranie parametrów funkcji opasującej stan psychofizyczny kierowcy, aby jak najlepiej odzwierciedlała kondycje osoby prowadzącej pojazd. Aplikacja powinna być informowana możliwie z największą dokładnością, jak duże są potrzeby kierowcy, jeżeli chodzi o sen oraz odpoczynek. Przeciwdziedziną funkcji F(o) jest przedział <1,600> wraz z wzrostem wartości funkcji rośnie poziom senności badanej osoby.

Doświadczenie 1.
Próbne nastawy parametrów T1 oraz T2 wynoszą:

- jeżeli wykryto sytuacje zamknięcia oczu, wartość T1 wynosi 0,005 jednostki na jedną klatkę filmu (0,125 j/sekundę),
- jeżeli mrużenie powiek trwa dłużej niż jedną klatkę (z reguły trwa) wartość T1 rośnie liniowo, zwiększając swoją wartość o 0,03 jednostki na klatkę (0,75 j/sekundę),
- wartość parametru polepszania się stanu kierowcy – T2 - wynosi 0,0006 jednostki na klatkę filmu (0,015 j/sekundę).

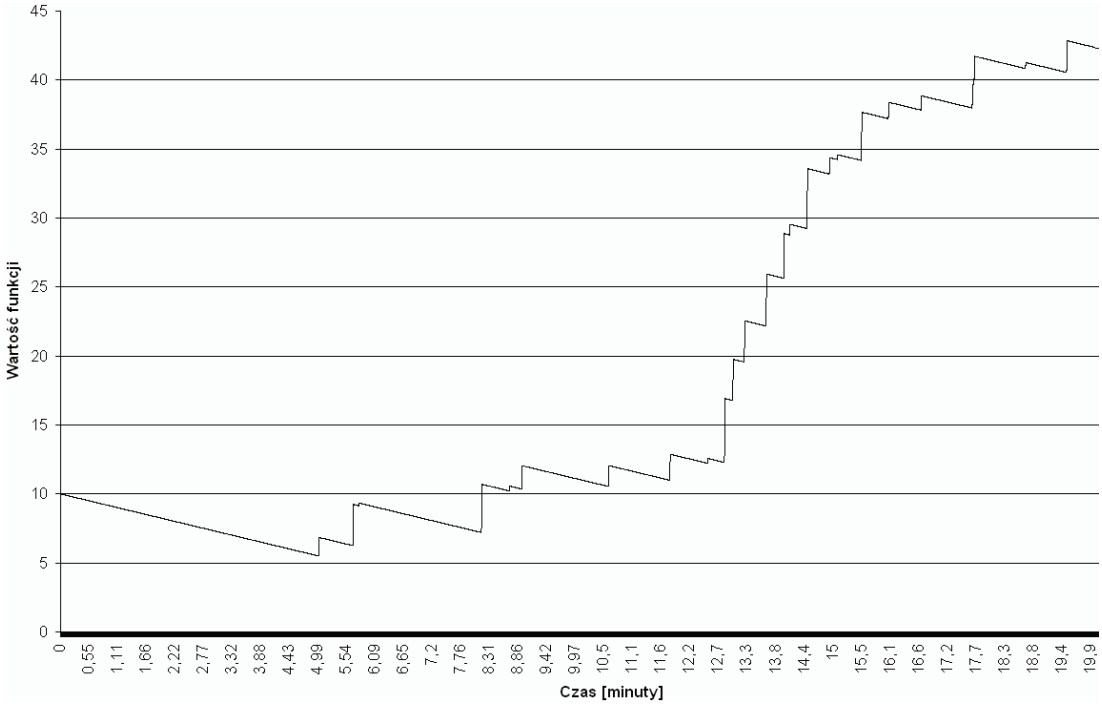
Rysunek 17. przedstawia wartość funkcji senności w zależności od czasu.



Rysunek 17. Wykres funkcji senności – doświadczenie 1.

Doświadczenie 2.

Autor pracy zdecydował się oscylować wokół wartości znalezionych w doświadczeniu 1. Wartość T1 rośnie od poziomu 0,01 jednostki na klatkę (0,25 jednostki na sekundę) z liniowym przyrostem 0,01 jednostki na klatkę. T2 pozostaje bez zmian na poziomie 0,0006 jednostki na jedną klatkę filmu. Funkcje senności doświadczenia 2. przedstawia rysunek 18.

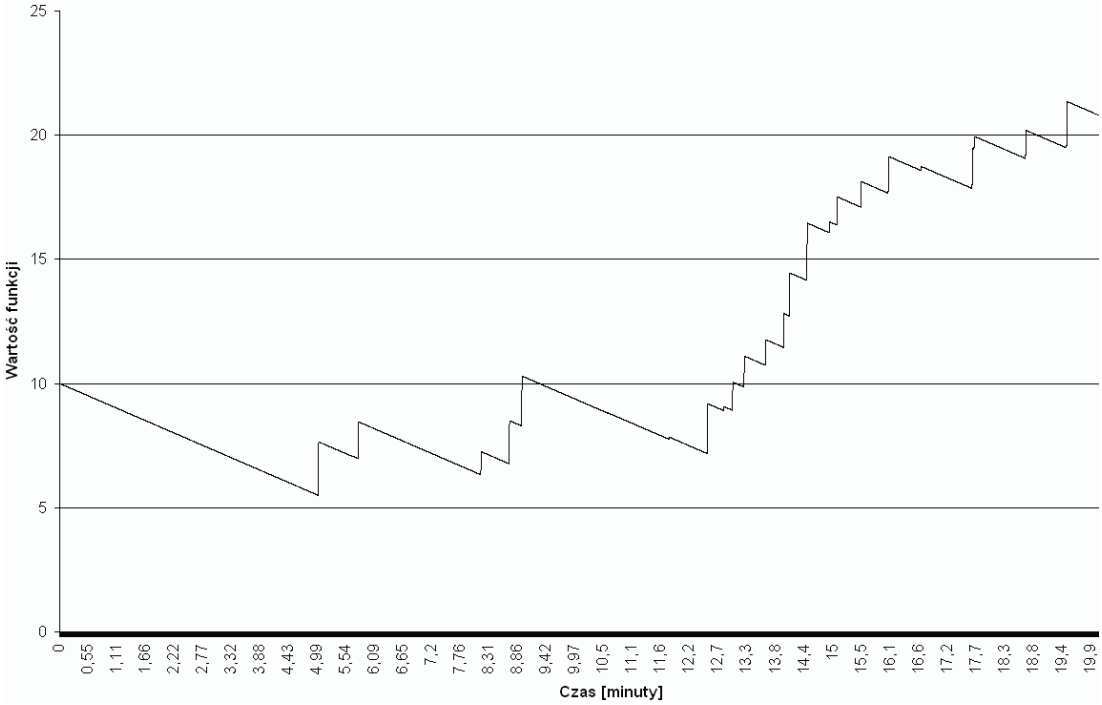


Rysunek 18. Wykres funkcji senności – doświadczenie 2.

Doświadczenie 3.

Wartość T1 w pierwszej klatce, gdzie oczy są zamknięte wynosi 0,01, T1 rośnie z krokiem 0,005 jednostki na klatkę (0,125 jednostki/sekundę).

T2 niezmiennie - 0,0006. Rezultat pracy algorytmu przedstawiono na rysunku 19.



Rysunek 19. Wykres funkcji senności – doświadczenie 3.

4.3.2 Wnioski wyciągnięte z doświadczeń.

Wartość T2 jest stała dla opisywanych doświadczeń,

Nastawy parametrów T1 oraz T2 wykorzystane w doświadczeniach zestawiono w tabeli 2.

Tabela 2. Nastawy T1 oraz T2 wykorzystane w trzech doświadczeniach.

Doświadczenie	T1 startuje od	T1 rośnie z krokiem	T2	T1 startuje od	T1 rośnie z krokiem	T2
Lp.	j/klatkę	j/klatkę	j/klatkę	j/sekundę	j/sekundę	j/sekundę
1	0,005	0,03	0,0006	0,125	0,75	0,015
2	0,01	0,01	0,0006	0,25	0,25	0,015
3	0,01	0,005	0,0006	0,25	0,125	0,015

Rysunki 17, 18, 19 przedstawiają wykresy funkcji dla różnych nastaw parametrów T1 oraz T2 – wyniki doświadczeń. Wyraźnie widać, że skoki wartości funkcji pokrywają się z faktem wykrycia mrużenia oczu pokazanym na rysunku 16. Im większy jest czas zamknięcia powiek, tym większy jest skok funkcji senności. Doświadczenia miały na celu dobranie właściwych proporcji – nastaw zmiennych wpływających na ostateczną wartość funkcji senności w danej chwili.

Analiza wyników badań przebiega w kilku płaszczyznach. Należy mieć na uwadze zbiór wartości funkcji, od 1 – kierowca wypoczęty do 600 kierowca zmęczony. Wartości nie mogą rosnąć zbyt szybko, nie dopuszczalna jest sytuacja kiedy funkcja rośnie powoli nie nadążając za kondycją osoby prowadzącej pojazd. Kierowca po całej nocy spędzonej w samochodzie (opisywana wcześniej trasa z Kopenhagi do Tarnowa), zarejestrował 30 minutowy film, spożył napój energetyczny oraz po godzinie przerwy zarejestrował 20 minutowy film będący przedmiotem badań. Przyjęto założenie, że wartość funkcji przy starcie algorytmu wynosi 10. Dla tego konkretnego przypadku, algorytm powinien „wykryć” szybko pogarszającą się kondycję kierowcy, napój energetyczny w sytuacji dużego zmęczenia „działa” około godziny. Z drugiej strony funkcja nie może rosnąć zbyt

gwałtownie, nie ma potrzeby zmuszać kierowcą do odpoczynku, który nie jest absolutnie niezbędny. Przyjęto założenie, że wartość zmęczenia osoby prowadzącej pojazd dla skrajnie „niewyspanego" kierowcy powinna rosnąc w tempie około 30 jednostek w dziesięć minut. Algorytm uzyska wtedy poziom niebezpiecznego zmęczenia kierowcy – 450 jednostek po 150 minutach. Jeżeli kierowca jest wypoczęty funkcja powinna (zakładając, że startuję od 10) wraca do poziomu 1.

W doświadczeniu 1. funkcja zmęczenia rośnie zdecydowanie za szybko. Od chwili kiedy stan kierowcy zaczął się mocno pogarszać (po dziesiątej minucie), funkcja zwiększyła swoją wartość aż o 120 jednostek! Tempo wzrostu zdecydowanie za duże, 600 jednostek zostanie osiągnięte w 50 minut! Dodatkowo algorytm źle oddaje kondycję kierowcy w pierwszych dziesięciu minutach pracy, średnia wartość funkcji spadła – po całej nocy spędzonej za kierownicą taka sytuacja jest niedopuszczalna.

W doświadczeniu 2. wartość funkcji poprawnie odzwierciedla lekko pogarszający się stan kierowcy w pierwszych minutach pracy algorytmu. Poprawnie zostało zarejestrowane i wykryte długie zamknięcie oczu w 13-tej minucie które zapoczątkowało szybkie pogarszanie się kondycji kierowcy. Wartość algorytmu rośnie maksymalnie w tempie 32 jednostki w 8 minut (40 jednostek w10 minut, 600 jednostek w 150 minut).

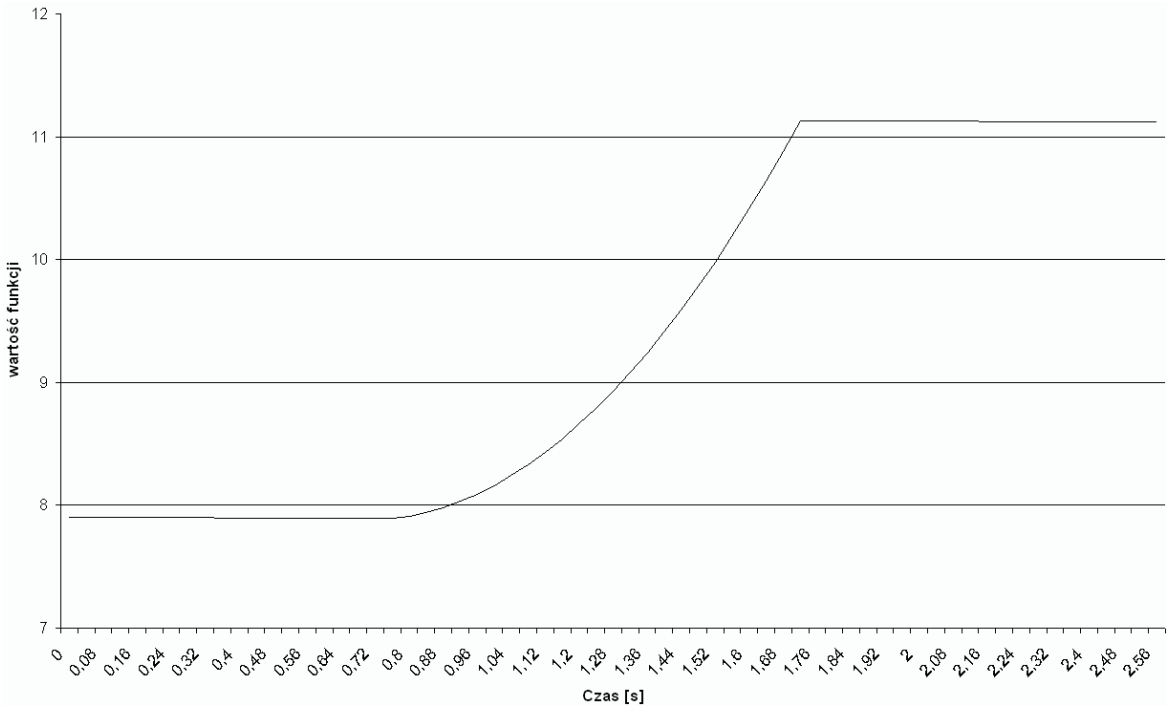
W doświadczeniu 3. wartość funkcji senności rośnie zdecydowanie za wolno. Algorytm wspomagający decyzje kierowcy o spożyciu napoju energetycznego, lub krótkiej drzemce działałby zdecydowanie za późno.

Podczas trwania badań przeprowadzono liczne próby, z różnymi nastawami T1 oraz T2. W porozumieniu z opiekunem pracy ostatecznie zdecydowano się na następujące wartości:

- funkcja F(o) może przyjmować wartości naturalne z przedziału 1 – 600, gdzie 1 – kierowca wypoczęty, 600 – kierowca zasypia,
- parametr T2 jest stały i wynosi 0,015 jednostki na 1 sekundę – w takim tempie spada wartość funkcji F(o), jeżeli oczy są otwarte,
- parametr T1 w pierwszej klatce zamknięcia oczu wynosi 0,01 jednostki na sekundę,
- jeżeli w każdej kolejnej klatce oczy są zamknięte, wartość T1 rośnie liniowo (z krokiem 0,01) aż do poziomu 0,30 jednostek na sekundę,

Dodatkowo aplikacja uznaje, że kierowca usnął, jeżeli jego oczy są nieustannie zamknięte przez 5 sekund.

Przy nastawach dobranych w taki sposób, pojedyncze zwiększenie wartości funkcji, jeżeli oczy kierowca są zamknięte przez 1 sekundę pokazano na rysunku 20.



Rysunek 20. Wzrost funkcji przy jednorazowym mrużeniu.

Program sugeruje kierowcy spożycie napoju regenerującego lub filiżankę kawy, jeżeli wartość funkcji osiągnie 300 jednostek.

Kierowca powinien udać się na minimum 20 minutową drzemkę, jeżeli funkcja osiągnie krytyczną wartość 450 jednostek.

Rysunek 21. przedstawia przykładową klatkę filmu, na którą aplikacja nanosi odpowiednie wyniki swojej pracy.

Rysunek 21. Przykładowa klatka z filmu z naniesionymi markerami.

W lewym górnym rogu aplikacja informuje użytkownika o aktualnym stopniu otwarcie oka lewego oraz prawego. Im prostokąt jest wyższy tym oczy SA bardziej otwarte. Wielkością prostokątów steruje sygnał po filtracji, liczona jest wartość średnia wysokości oka z 12 poprzednich klatek. Program, bazując na funkcji opisującej stan kierowcy doradza osobie aktualnie prowadzącej pojazd:

- dalszą bezpieczną jazdę - brak ikony,
- spożycie napoju energetycznego – ikona widoczna na rysunku 21 w prawym górnym rogu,
- krótką drzemkę, duża ikona w prawym górnym rogu.

Wartość samej funkcji opisującej kondycję kierowcy obrazowana jest przez wielkość prostokąta u dołu ekranu. Jeżeli osoba prowadząca pojazd staje się bardziej senna, prostokąt wydłuża się.

Pojedyncze nie zamalowane w środku prostokąty, które występują na twarzy kierowcy to pary obiektów, które są potencjalnymi oczami. Film służy celom badawczym, informacja,

że są na ekranie obiekty, które mogą być rozpoznane jako oczy jest niezwykle cenna. Im większa jest ilość tych prostokątów na ekranie tym większą trudność miał algorytm przy podejmowaniu decyzji o analizie właściwych oczu. Oczy, które pomyślnie przeszły proces walidacji zaznaczono długim nie zamalowanym prostokątem (jedna para oczu we wspólnym prostokącie).

5. Podsumowanie, wnioski, dalszy rozwój projektu.

Celem opisanej pracy było zaprojektowanie oraz przetestowanie systemu aktywnego śledzenia oczu kierowcy, który na podstawie ruchów powiek kontroluje kierowcę samochodu i wnioskuje o włączeniu alarmu oraz jeśli działanie to jest nieskuteczne o aktywnym włączeniu systemu hamowania. Moc obliczeniowa dzisiejszych procesorów, oraz niezwykle daleko zaawansowana technika wizyjna pozwalają na stosowanie niemalże nieograniczonej kombinacji różnych technik przetwarzania obrazu. Autor pracy testując wiele metod współpracy komputera klasy PC z kamerą cyfrową, oraz analizując wyniki pracy różnego rodzaju algorytmów służących do wstępnej obróbki obrazu, rozpoznawania wzorców w danej mapie bitowej oraz ciągłej pracy programu w czasie rzeczywistym zdecydował się na następujący sposób budowy urządzenia:

1. kamera cyfrowa o rozdzielczości minimum 640x480 pixeli zainstalowana w samochodzie śledzi oczy kierowcy, rejestruje ruch powiek,
2. program komputerowy analizuje film w czasie rzeczywistym klatka po klatce wykorzystując następujące algorytmu:
    a) oczyszczanie obrazu filtrem blur,
    b) wykrywanie krawędzi mocnym filtrem outline,
    c) tradycyjne indeksowanie obiektów
    d) rozpoznawanie oczu (z pośród dostępnych elementów aplikacja wybiera te o właściwym rozmiarze, później te, które pasują do wzorca (momenty „Hu”), walidacja polegająca na badaniu wzajemnych zależności pomiędzy elementami),
    e) rozpoznawanie, czy w danej klatce oczy są zamknięte, czy otwarte (poprzedzone filtracją),
    f) zbieranie i agregacja danych dotyczących stanu oczy kierowcy w poszczególnych klatkach,
    g) podejmowanie decyzji dotyczącej aktualnej kondycji psychofizycznej osoby prowadzącej pojazd,
    h) doradzanie kierowcy – w zależności od jego stanu – różnego rodzaju propozycje „poprawy jego senności”,
    i) wykrywanie momentu zaśnięcia kierowcy,

3. aplikacja komunikuje się z kierowcą przy pomocy okonek wyświetlanych na desce rozdzielczej (propozycja spożycia napoju energetycznego, etc.),
4. program komunikuje się z komputerem pokładowym (ewentualne awaryjne hamowanie),

Autor projektu po konsultacjach z opiekunem pracy dr Adrianem Horzykiem uznał, że taka konstrukcja urządzenia jest najbardziej optymalna na potrzeby monitorowania oraz wnioskowania o kondycji kierowcy samochodu. Opisane, wykorzystane algorytmy wraz z odpowiednimi zależnościami pomiędzy nimi posiadają następujące zalety:

- aplikacja rejestruje twarz kierowcy, która znajduje się w odległości kilkudziesięciu cm od kamery, rozdzielczość stosunkowo nie drogich kamer – 640x480 pixeli jest wystarczająca,
- w samochodzie niezwykle często dochodzi do gwałtownej zmiany oświetlenia twarzy kierowcy, mocny filtr wykrywający oraz wzmacniający krawędzie obiektów niemalże uniezależnia aplikacje od stosunkowo dużych wahań poziomu oświetlenia,
- wstępna klasyfikacja obiektów na ekranie – badanie ich rozmiaru znacznie przyśpiesza działanie algorytmu – program nie musi przetwarzać szumów oraz innych małych obiektów, które bardzo często występują w opisywanych warunkach,
- wykorzystanie momentów „Hu" pozwala na uniezależnienie się od skali, poziomu obrócenia oraz umiejscowienia na ekranie próbki obrazu porównywanej z wzorcem,
- samo badanie zgodności wzorca z analizowanym elementem obrazu (potencjalnym okiem) jest mało skomplikowane obliczeniowo, wystarczy przyrównać do siebie sześć skalarów, co zapewnia dużą wydajność obliczeniową
- wykorzystany sposób walidacji obliczeń znacznie zwiększa pewność otrzymywanych wyników, oczy zawsze są w pewnej, stałej odległości od siebie, występują na podobnym poziomie, aplikacja świetnie korzysta z tych danych,
- każda klatka filmu otrzymuje etykietę, binarną zmienna mówiącą, czy oczy są zamknięte, czy otwarte z uwzględnieniem filtracji codziennego mrugania

powiekami, co znacząco upraszcza analizę filmu pod względem badania senności kierowcy,

- stworzenie dynamicznej funkcji senności, która zależna jest od danych dotyczących stanu powiek kierowcy w danej chwili oraz przeszłości wydaje się idealnym narzędziem analizy kondycji osoby prowadzącej pojazd, w każdej chwili wartość funkcji informuje otoczenie aplikacji o poziomie senności kierowcy, łatwo można określić stan organizmu w przeszłości,
- aplikacja zależnie od kondycji kierowcy proponuje spożycie napoju regenerującego lub krótką drzemkę, nierzadko kierowca skupiony na samym fakcie prowadzenia pojazdu, zapomina o swojej pogarszającej się kondycji i niespostrzeżenie zasypia, opisane w tej pracy mechanizmy zmniejszają ryzyko zaśnięcia za kierownicą,
- decyzja o rozpoczęciu ewentualnego hamowania awaryjnego zwiększa szanse przeżycia kierowcy, który zasnął podczas prowadzenia pojazdu.

Liczne próby, doświadczenia pozwoliły autorowi pracy na precyzyjne zestrojenie mechanizmów decyzyjnych. Aplikacja bardzo dobrze oddaje zmiany poziomu ludzkiej senności.
Urządzenie może być stosowane przez większość kierowców (zarówno kobiety z mocnym makijażem podkreślającym oczy, jak również mężczyźni o różnym kolorze skóry).

Aplikacja, której kod znajduje się na listingu 1. została napisana w środowisku Matlab w taki sposób, aby stosunkowo niewielkim nakładem pracy można ją było przenieść na inne platformy programowe oraz sprzętowe. Program nie wykorzystuje gotowych implementacji algorytmów środowiska Matlab (nawet zamiana obrazu na kolor szary, oraz binaryzacja wykonywane są w pętlach „for"). Autor pracy wykorzystuje aplikacje w celach badawczych, dlatego program wczytuje wcześniej zarejestrowany film, przetwarza go, a następnie zapisuje plik „avi" wraz z markerami – wynikami analizy. Złożoność obliczeniowa jest dosyć duża, jedna klatka filmu obliczana jest w 0,88 sekundy, stanowczo za długo, jeżeli program miałby pracować w urządzeniu zainstalowanym w samochodzie w czasie rzeczywistym – 25 klatek na sekundę. Środowisko Matlab wszelkie obliczenia wykonuje na liczbach typu „double", prawie wszystkie dane w programie są typu „integer", których wielkość w większości przypadków mieści się w 8 bitach – obraz w 8 bitowej skali szarości. Przekonwertowanie aplikacji do języka C++ zwiększa szybkość pracy algorytmów od dziesięciu do pięćdziesięciu razy. Dodatkowo mechanizmy wstępnej

obróbki obrazu można realizować sprzętowo, co sprawia, że mechanizm może działać z prędkościom dochodzącą do 50 klatek na 1 sekundę przy rozdzielczości 640x480 pixeli.
Projekt może być rozwijany poprzez dogłębna analizę potrzeb snu różnych kierowców. Aplikacja może uczyć się zachowań wybranego kierowcy, lepiej oceniając jego kondycje psychofizyczną. Stosunkowo łatwo zwiększyć wachlarz możliwości komunikacji urządzenia z osobą prowadzącą pojazd (włączenia, radia, komunikaty głosowe, wentylacja kabiny pasażerskiej, etc).
Urządzenie może być wykorzystywane w zupełnie innym celu niż zostało zaprojektowane:

- dodając mechanizm rozpoznający twarz (kilka prostych relacji usta, oczy, nos) można sprawić, że żaden inny kierowca (poza zdefiniowanym wcześniej) nie uruchomi samochodu – nikt nie ma takiego immobilizera, który rozróżnia kierowców po twarzy!
- rejestrowanie funkcji senności ułatwia badania nad samym mechanizmem zasypiania, to cenne narzędzie dla osób zawodowo zajmujących się problematyką snu za kierownicą, algorytmu służą jako uzupełnienie oraz automatyzacja rejestrowania danych zbieranych podczas badań,
- urządzenie może służyć jako narządzie analizy reakcji wzorku człowieka na różne bodźce (np. jak różne barwy światła białego meczą wzrok),
- wiele innych …

6. Literatura.

1. Bielecki Z. i Rogalski A., „Detekcja sygnałów optycznych”, WNT, 2001.

2. Borbely Aleksander, „Tajemnice snu”, PWN, Warszawa 1990,

3. Cieslak, M., Smoluk, A. „Zbiory rozmyte. Rozpoznawanie obrazów. Teoria katastrof.”, PWN, 1998.

4. Gonzales R., Woods R., “Digital Image Processing”, ISBN 0-201-50803-6, 1992

5. Hjelmås Erik, “Biometric Systems: A Face Recognition Approach”, University of Oslo, 2006.

6. Hsu R., Abdel-Mottaleb, Jain A., “Face detection in color images”, ICIP, 2001.

7. Kotropoulos, Tefas, Pitas, "Frontal face authentication using morphological elastic graph matching.', ' IEEE Trans. Image Processing, 2000.

8. Kryger M.H., Roth T., Dement W.C., "In Principles and Practice of Sleep Medicine.”, eds. Philadelphia, PA, W.B. Saunders Company, 2004

9. Lindeberg Tony, “Scale-Space Theory in Computer Vision”, ISBN 0-7923-9418-6, 1994

10. Morimoto Carlos, Koons Dave, Amir Arnon, Flickner Myron, “Real-Time Detection of Eyes and Faces”, IBM Almaden Research Center, 2005.

11. Pavlidis, T., „Grafika i przetwarzanie obrazów. Algorytmy”, WNT, 1987.

12. Romeny H., “Front-End Vision and Multi-Scale Image Analysis”, ISBN 1-4020-1507-0, 2003

13. Romeny H., “Geometry-Driven Diffusion in Computer Vision”, ISBN 0792330870, 1994

14. Rosekind Mark R., Dinges David, Curtis Graeber R., Samel Alexander, Wegmann Hans M., “Principles and Guidelines for Duty and Rest Scheduling in Commercial Aviation”, NASA Technical Memorandum, 1995

15. Rosekind Mark R., “Fatigue and Its Safety Effects on the Commercial Motor Vehicle and Railroad Industries”, 1998

16. Rosekind, Mark R., "Managing work schedules: An alertness and safety perspective.”, wersja elektroniczna.

17. Russ J., “The Image Processing Handbook”, ISBN 0849372542, 2006

18. Szafran J., Wiszniewski A., „Algorytmy pomiarowe i decyzyjne cyfrowej automatyki elektroenergetycznej”, WNT, 2001.

19. Tadeusiewicz, R., Flasinski, M., „Rozpoznawanie obrazów”, PWN 1991.

20. 6. Tadeusiewicz Ryszard, „Komputerowa analiza i przetwarzanie obrazów.”, Kraków, Wydawnictwo Fundacji Postępu Telekomunikacji, 1997.

21. Watkins, C.D., Sadun, A., Marenka, S,. „Nowoczesne metody przetwarzania obrazu.”, WNT, 1995.

22. Wojciechowski, K., „Rozpoznawanie obrazów”, Politechnika Śląska, 1997.

23. Young T., Vliet V., “Fundamentals of Image Processing”, ISBN 90-75691-01-7, 1995

Dodatek A. Spis rysunków.

Dodatek B. Spis tabel.

Dodatek C. Kod programu.

Listing 1. Implementacja programu w języku Matlab.

movieproces.m – główny plik wykonywalny

```
clear all;
tic
moviename='db.avi';%ab i ce
readonce=10;   % ile klatek wczytywac jednoczesnie (400 1Gb ram, 50
512MbRam)
readtable=1:readonce;

aviobj=avifile('testdbroc75cze12ikony.avi');
numerator=0;    %ktora klatke wlasnie obrabia
ttlfound=0;     %na ilu klatkach rozpoznal

inform=aviinfo(moviename);

%X = aviread(moviename,readtable);

%X = aviread('finall.avi');

movdx=inform.Height;
movdy=inform.Width;
framescount=inform.NumFrames
aviobj.quality = 100;
aviobj.compression = 'None'

prevframex=round(movdx/2);  %wspolrzedne oczu w poprzedniej klatce x
prevframey=round(movdy/2);  %wspolrzedne oczu w poprzedniej klatce x
prevframes=zeros(movdx,movdy);% poprzednia zaleziona siatka z
zaznaczonymi oczami

histrylen=175;  % jak dlugi jest filtr zamkniecia oczu
graphhistrylen=12;% dlugosc filtru do wizualizacji
leftheighistry=24*ones(histrylen,1);% filtr - kolejne probki
rightheighistry=24*ones(histrylen,1);% filtr - kolejne probki
graphleftheig=0;    %usrednione wielkosci oczu do ryzunku oczu
graphrightheig=0;
drvrsleftheig=0;    %wysokosc otwartych oczu kierowcy -srednia z 7 sekund
drvrsrightheig=0;
sleeply=10;          %1 nie spiacy - 600 max spiacy
sleeplystep=0.03;   %jezeli oczy ciagle zamkniete to wskaznik przyrasta o
tyle szybciej z kazda klatka

prevclosed=0;       %jezeli w poprzedniej klatce bylo zamkniete to 1, jak
otwarte to 0
closedcount=0;      %przez ile klatek pod rzad oko jest zamkniete
closedcountlimit=15;%jezeli przez tyle klatek oczy sa zamkniete to
kierowca spi
closedcountset=0;   %jezeli 1 to znaczy ze zasnal

falasl=imread('zasnal.bmp');falasl=double(falasl);%przygotowanie napisow
do filmu

rest=imread('wypoczety1.bmp');rest=double(rest);
slplbmp=imread('senny.bmp');slplbmp=double(slplbmp);
coffbmp=imread('e.bmp');coffbmp=double(coffbmp);
goslpbmp=imread('drzemka.bmp');goslpbmp=double(goslpbmp);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%analiza filmu klatka po klatce

for aa=1:readonce:(framescount-readonce)
    readtable=aa:1:(aa+readonce-1);
    X = aviread(moviename,readtable);
    for bb=1:1:(readonce)
        numerator=numerator+1
        film=X(bb).cdata;
        film=double(film);

[found,frames,leftcentx,leftcenty,rightcentx,rightcenty,leftheig,righthei
g]=lookforeyes(film,prevframex,prevframey,1);
        found
        if(found==0)    % jezeli nie znalazl to szukaj z mniejsza
precyzja rozpoznawania - wieksza tolerancja
            if(ttlfound>0)

[found,frames,leftcentx,leftcenty,rightcentx,rightcenty,leftheig,righthei
g]=lookforeyes(film,prevframex,prevframey,2);
            end;
        end;

        if(found==1)
            if((((prevframex-
round((rightcentx+leftcentx)/2))^2)+((prevframey-
round((rightcenty+leftcenty)/2))^2))>=10000)
                found=0;    %jezeli oczy w 1 klatce przesunely sie o 100
            else
                ttlfound=ttlfound+1;
                prevframex=round((rightcentx+leftcentx)/2);
                prevframey=round((rightcenty+leftcenty)/2);
            end;
        end;

        for cc=2:1:histrylen    %filtr przes sie o 1 w tyl i bierze
kolejna probke
            leftheighistry(cc-1)=leftheighistry(cc);rightheighistry(cc-
1)=rightheighistry(cc);
        end;

leftheighistry(histrylen)=leftheig;rightheighistry(histrylen)=rightheig;

        graphleftheig=0;graphrightheig=0;
        drvrsleftheig=0;drvrsrightheig=0;
        for cc=1:1:(histrylen-graphhistrylen-1)   %pobierz usrednionedane
do wysokosci oczy kierowcy
            drvrsleftheig=drvrsleftheig+leftheighistry(cc);
            drvrsrightheig=drvrsrightheig+rightheighistry(cc);
        end;
        for cc=(histrylen-graphhistrylen):1:histrylen   %pobierz
usrednionedane do rysunku oczu + kierowcy oczy
            drvrsleftheig=drvrsleftheig+leftheighistry(cc);
            drvrsrightheig=drvrsrightheig+rightheighistry(cc);
            graphleftheig=graphleftheig+leftheighistry(cc);
            graphrightheig=graphrightheig+rightheighistry(cc);
        end;
        drvrsleftheig=round(drvrsleftheig/histrylen);
        drvrsrightheig=round(drvrsrightheig/histrylen);
        graphleftheig=round(graphleftheig/graphhistrylen);
        graphrightheig=round(graphrightheig/graphhistrylen);
```

```
%if((graphleftheig<=(0.75*drvrsleftheig))&&(graphrightheig<=(0.75*drvrsri
ghtheig)))%jezlei aktualnie oko jest zamkniete w stosunku do przeszlosci

            sleeply=sleeply+1+sleeplystep;
            sleeplystep=sleeplystep+sleeplystep;
            if(sleeplystep>0.1) sleeplystep=0.1; end;
            if(prevclosed==1)   % jak w poprzedniej klatce - zamkniete
czy nie
                closedcount=closedcount+1;
            else
                closedcount=0;
            end;
            prevclosed=1;
        else
            sleeplystep=0.03;
            prevclosed=0;
        end;

        if(closedcount>=closedcountlimit)
            closedcountset=1;
        end;

        sleeply=sleeply-0.015;   %jezeli nie zamyka oczu to nie jest
senny - wskaznik wraca powoli do normy-stan czuwania

        if(sleeply<1) sleeply=1; else if(sleeply>600) sleeply=600; end;
end;%zabezpieczenie

        new_image4=(film);

        if(found==0) frames=prevframes; else prevframes=frames; end; %
jezeli nie znajdzie to pozostaw ramke na poprzednich oczach

        if(closedcountset==1)
frames=draw_bmpfile(frames,falasl,140,130,0,255); end;    %czy zasnal

        if(sleeply>=300) if(sleeply>=450)
frames=draw_bmpfile(frames,goslpbmp,20,490,255,0); else
frames=draw_bmpfile(frames,coffbmp,20,580,255,0); end; end;

        frames=draw_filledrect(frames,20,20,20+graphleftheig*2,70);
        frames=draw_filledrect(frames,20,90,20+graphrightheig*2,140);
        frames=draw_filledrect(frames,460,20,470,20+sleeply);

        frames=draw_bmpfile(frames,rest,433,20,255,0);  %napisy
wypoczety, senny
        frames=draw_bmpfile(frames,slplbmp,433,537,255,0);


        for x=1:1:movdx    %szary
           for y=1:1:movdy

temporary=round((new_image4(x,y,1)+new_image4(x,y,2)+new_image4(x,y,3))/3
);
              new_image4(x,y,1)=temporary;
              new_image4(x,y,3)=temporary;
              new_image4(x,y,2)=temporary;
              if(frames(x,y)>0)

new_image4(x,y,1)=255;new_image4(x,y,2)=0;new_image4(x,y,3)=0;
              end;

           end;
```

```
        end;
        aviobj = addframe(aviobj,uint8(new_image4));
        toc
        %if(bb>2) break; end;
        end;

    end;
end;

numerator

new_image4=uint8(new_image4);
image(new_image4);

aviobj=close(aviobj);
```

lookforeyes.m – funkcja wyszukuje oczy w pojedynczej klatce filmu

```
function
[rfound,rframe,rleftcentx,rleftcenty,rrightcentx,rrightcenty,rleftheig,rr
ightheig]=lookforeyes(inputimage,iprevcentx,iprevcenty,precise)

%oblicza pojedyncza klatke z filmu

%[obraz z ramkami z oczami,wspolrzedne oczu,wysokosci oczu] = obraz
%rgb,wspolrzedne oczu w poprzedniej klatce, dokladnosc
szukania=tolerancja
%precise=1 - najdokladniej, 2 - mniej dokladnie

ALLMARKERSON=0;%jezeli 1 to na klatce wyswietla dodatkowe ramki z
potencjalknymi oczami, ktore pozniej odpadly przy klasyfikacji obiekty

my_image=inputimage;
my_image=double(my_image);
dx=length(my_image(:,1,1));
dy=length(my_image(1,:,1));
new_image=zeros(dx,dy);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



sel=1;      %ktora maska wygladzania sel=0 wylanczony blur
if(sel==1)
  conv_mask=[1,1,1,1,1;
            1,1,1,1,1;
            1,1,1,1,1;
            1,1,1,1,1;
            1,1,1,1,1];
  conv_mask_dim=2;  %maska 5x5 ma 2 7x7 ma 3
end;
if(sel==2)
  conv_mask=[1,1,1;
            1,1,1;
            1,1,1];
  conv_mask_dim=1;  %maska 5x5 ma 2 7x7 ma 3
end;
if(sel==3)
 conv_mask=[1,1,1,1,1,1,1;
            1,1,1,1,1,1,1;
            1,1,1,1,1,1,1;
```

```
            1,1,1,1,1,1,1;
            1,1,1,1,1,1,1;
            1,1,1,1,1,1,1;
            1,1,1,1,1,1,1];
  conv_mask_dim=3;  %maska 5x5 ma 2 7x7 ma 3
end;

for x=1:1:dx    %szary
    for y=1:1:dy

new_image(x,y)=round((my_image(x,y,1)+my_image(x,y,2)+my_image(x,y,3))/3)
;
    end;
end;

r1_image=zeros(dx,dy); %//rezultat 1 obrazek
if(sel>0)
for x=1+conv_mask_dim:1:dx-conv_mask_dim-1    %filtr blur
    for y=1+conv_mask_dim:1:dy-conv_mask_dim-1
        for mask_x=-1*conv_mask_dim:1:conv_mask_dim
            for mask_y=-1*conv_mask_dim:1:conv_mask_dim

r1_image(x,y)=r1_image(x,y)+(conv_mask(mask_x+conv_mask_dim+1,mask_y+conv
_mask_dim+1)*new_image(x+mask_x,y+mask_y));
            end;
        end;
    end;
end;


max_value=max(max(r1_image));   %normalizacja liniowa
r1_image=round(255*r1_image/max_value);
new_image=r1_image;
end;



grey_image=round(r1_image);



if(1)% wylancza blok obliczen, laduje zmienna z pliku




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%%%po blurze filtr outline

 conv_mask=[1,1,1,1,1,1,1;
            1,0,0,0,0,0,1;
            1,0,0,0,0,0,1;
            1,0,0,-24,0,0,1;
            1,0,0,0,0,0,1;
            1,0,0,0,0,0,1;
            1,1,1,1,1,1,1];
  conv_mask_dim=3;  %maska 5x5 ma 2 7x7 ma 3
r1_image=zeros(dx,dy); %//rezultat 1 obrazek
for x=1+conv_mask_dim:1:dx-conv_mask_dim-1    %filtr blur
    for y=1+conv_mask_dim:1:dy-conv_mask_dim-1
        for mask_x=-1*conv_mask_dim:1:conv_mask_dim
            for mask_y=-1*conv_mask_dim:1:conv_mask_dim
```

```
r1_image(x,y)=r1_image(x,y)+(conv_mask(mask_x+conv_mask_dim+1,mask_y+conv
_mask_dim+1)*new_image(x+mask_x,y+mask_y));
            end;
        end;
    end;
end;

normali=3;                          % musi byc 3 zeby mozna bylo
etykietowac
if(normali==1)
    min_value=min(min(r1_image));   %normalizacja liniowa
    r1_image=r1_image-min_value;
    max_value=max(max(r1_image));
    r1_image=round(255*r1_image/max_value);
end;

if(normali==2)                      %normalizacja nie liniowa wynik
liniowy 0-255
     for x=1:1:dx
         for y=1:1:dy
             if r1_image(x,y)<0  r1_image(x,y)=0; elseif
r1_image(x,y)>255 r1_image(x,y)=255; end;
         end;
     end;
    r1_image=round(r1_image);
end;

if(normali==3)                      %normalizacja nie liniowa wynik
binarny - binaryzacja
    n_border=100;
     for x=1:1:dx
         for y=1:1:dy
             if r1_image(x,y)<=n_border  r1_image(x,y)=0; elseif
r1_image(x,y)>n_border r1_image(x,y)=1; end;
         end;
     end;
    r1_image=round(r1_image);
end;

                    %labeling - indeksacja obiektow
cur_ind=0;                  %index ktory teraz przyznaje to ..2
alot=333333;

r1_image=label_images(r1_image,2,2,dx-1,dy-1);


%save('r1_imagep26.mat','r1_image');
end;%od wlanczania pobierania i indexacji
%load('r1_imagep26.mat','r1_image');
alot=333333;

%toc
%%%obliczanie momentow
moments00=zeros(1,alot);%momenty 00 dla poszczegolnych obiektow
(geometryczne zwykle)
moments01=zeros(1,alot);
moments10=zeros(1,alot);
cmoments00=zeros(1,alot);%momenty 00 dla poszczegolnych obiektow
(geometryczne centralne)
cmoments02=zeros(1,alot);
cmoments03=zeros(1,alot);
cmoments11=zeros(1,alot);
cmoments12=zeros(1,alot);
```

```
cmoments20=zeros(1,alot);
cmoments21=zeros(1,alot);
cmoments30=zeros(1,alot);
cnmoments02=zeros(1,alot);% momenty centralne znormalizowane
cnmoments03=zeros(1,alot);
cnmoments11=zeros(1,alot);
cnmoments12=zeros(1,alot);
cnmoments20=zeros(1,alot);
cnmoments21=zeros(1,alot);
cnmoments30=zeros(1,alot);
centx=zeros(1,alot);        %srodek obiektu X
centy=zeros(1,alot);        %srodek obiektu X
funhu=zeros(6,alot);        %funkcje momentow - Hu
score=alot*ones(3,alot);        %posrednia punktacja jak probka pasuje do
wzorca(nr wzoru,probka)

items=zeros(1,alot);    %jezeli = 1 to znaczy, ze obiekt istnieje
lastitem=0;             %ostatnia jedynka
itemlist=zeros(1,alot); %lista z numerami obiektow ktore sa klasyfikowane
jako oczy
itemlistlen=0;          %ile obiektow jest zaklasyfikowanych jako oczy
minxval=5000*ones(1,alot);  %min x value to lewy skrajny punkt danego
obiektu
minyval=5000*ones(1,alot);
maxyval=zeros(1,alot);
maxxval=zeros(1,alot);
selecteditems=zeros(dx,dy);
MINPATX=10;           %minimalna szerokosc rozpoznan obiektu
MAXPATX=30;
MINPATY=30;if(precise==2) MINPATY=15; end;
MAXPATY=70;

for x=2:1:(dx-1)        %%%obliczanie momentow
    for y=2:1:(dy-1)    %%%zbierania informacji o obiektach
        if(r1_image(x,y)>0)
            r1_imagetemp=r1_image(x,y);
            items(r1_imagetemp)=1;
            if(r1_imagetemp>lastitem) lastitem=r1_imagetemp; end;
            moments00(r1_imagetemp)=(moments00(r1_imagetemp)+1);
            moments10(r1_imagetemp)=(moments10(r1_imagetemp)+x);
            moments01(r1_imagetemp)=(moments01(r1_imagetemp)+y);
            if(minxval(r1_imagetemp)>x) minxval(r1_imagetemp)=x; end;
            if(minyval(r1_imagetemp)>y) minyval(r1_imagetemp)=y; end;
            if(maxxval(r1_imagetemp)<x) maxxval(r1_imagetemp)=x; end;
            if(maxyval(r1_imagetemp)<y) maxyval(r1_imagetemp)=y; end;
        end;
    end;
end;

sizematchitems=0;       %ile probek pasuje rozmiarem
for a=1:1:lastitem      %%wstepna klasyfikacja obiektow
    if(items(a)==1)
        if((maxxval(a)-minxval(a))>=MINPATX)
            if((maxyval(a)-minyval(a))>=MINPATY)
                if((maxxval(a)-minxval(a))<=MAXPATX)
                    if((maxyval(a)-minyval(a))<=MAXPATY)
                        %selecteditems=draw_rect(selecteditems,minxval(a)-
1,minyval(a)-1,maxxval(a)+1,maxyval(a)+1);
                        %selecteditems=draw_rect(selecteditems,minxval(a)-
0,minyval(a)-0,maxxval(a)+0,maxyval(a)+0);
                        items(a)=2;  %spelnia rozmiar
                        sizematchitems=sizematchitems+1;
                        %a
                    end;
```

```
            end;
        end;
    end;
end;
end;


%%%%%%%%%%%%


for a=1:1:lastitem      %%liczenie momentow centralnych i funkcji HU
    if(items(a)==2)
        centx(a)=round(moments10(a)/moments00(a));
        centy(a)=round(moments01(a)/moments00(a));
        cmoments00(a)=moments00(a);
        for x=minxval(a):1:maxxval(a)
            for y=minyval(a):1:maxyval(a)
                if(r1_image(x,y)>0)
                    cmoments02(a)=(cmoments02(a)+(y-centy(a))^2);
                    cmoments03(a)=(cmoments03(a)+(y-centy(a))^3);
                    cmoments11(a)=(cmoments11(a)+(x-centx(a))*(y-
centy(a)));
                    cmoments12(a)=(cmoments12(a)+(x-centx(a))*((y-
centy(a))^2));
                    cmoments20(a)=(cmoments20(a)+((x-centx(a))^2));
                    cmoments21(a)=(cmoments21(a)+((x-centx(a))^2)*((y-
centy(a))));
                    cmoments30(a)=(cmoments30(a)+((x-centx(a))^3));
                end;
            end;
        end;
        cnmoments02(a)=cmoments02(a)/((cmoments00(a))^(3));
        cnmoments03(a)=cmoments03(a)/((cmoments00(a))^(5/2));
        cnmoments11(a)=cmoments11(a)/((cmoments00(a))^(2));
        cnmoments12(a)=cmoments12(a)/((cmoments00(a))^(5/2));
        cnmoments20(a)=cmoments20(a)/((cmoments00(a))^(2));
        cnmoments21(a)=cmoments21(a)/((cmoments00(a))^(5/2));
        cnmoments30(a)=cmoments30(a)/((cmoments00(a))^(5/2));
        funhu(1,a)=cnmoments20(a)+cnmoments02(a);
        funhu(2,a)=(cnmoments20(a)-
cnmoments02(a))^2+4*(cnmoments11(a)^2);
        funhu(3,a)=(cnmoments30(a)-3*cnmoments12(a))^2+(3*cnmoments21(a)-
cnmoments03(a))^2;

funhu(4,a)=(cnmoments30(a)+cnmoments12(a))^2+(cnmoments21(a)+cnmoments03(
a))^2;
        funhu(5,a)=(cnmoments30(a)-
3*cnmoments12(a))*(cnmoments30(a)+cnmoments12(a))*((cnmoments30(a)+cnmome
nts12(a))^2-3*(cnmoments21(a)+cnmoments03(a))^2)+(3*cnmoments21(a)-
cnmoments03(a))*(cnmoments21(a)+cnmoments03(a))*(3*(cnmoments30(a)+cnmome
nts12(a))^2-(cnmoments21(a)+cnmoments03(a))^2);
        funhu(6,a)=(cnmoments20(a)-
cnmoments02(a))*((cnmoments30(a)+cnmoments12(a))^2-
(cnmoments21(a)+cnmoments03(a))^2)+4*cnmoments11(a)*(cnmoments30(a)+cnmom
ents12(a))*(cnmoments21(a)+cnmoments03(a));
        %a
    end;
end;

tol=03;                     %wspolczynnik tolerancji - im wiekszy tym
wiecej obiektow mozna zakwalfikowac
%pattno=1;                   %ktory obraz jest wzorcem
temp=0;
patterndesc=0;              %zmienna z wzorcem
```

```
lippattdesc=0;
load('pmixpattern1.mat');
load('pmixlippattern.mat');
%funhu(:,1)=patterndesc;


for a=1:1:lastitem      %%rozpoznawanie obrazow prawe oko - wzorzec w
tolerancji - wartosci bezwzgledne
    if(items(a)==2)
        if(sum((abs(funhu(:,a))>=(abs(patterndesc)-
tol*abs(patterndesc)))&(abs(funhu(:,a))<=(abs(patterndesc)+tol*abs(patter
ndesc))))>=3)% standardowo 4
                %selecteditems=draw_rect(selecteditems,minxval(a)-
4,minyval(a)-4,maxxval(a)+4,maxyval(a)+4);
                if(ALLMARKERSON)
selecteditems=draw_rect(selecteditems,minxval(a)-3,minyval(a)-
3,maxxval(a)+3,maxyval(a)+3); end;
                items(a)=3;
                itemlistlen=itemlistlen+1;
                itemlist(itemlistlen)=a;
                %a
         end;
    end;
end;

load('pmixpattern2.mat');

for a=1:1:lastitem      %%rozpoznawanie obrazow lewe oko - wzorzec w
tolerancji - wartosci bezwzgledne
    if(items(a)==2)
        if(sum((abs(funhu(:,a))>=(abs(patterndesc)-
tol*abs(patterndesc)))&(abs(funhu(:,a))<=(abs(patterndesc)+tol*abs(patter
ndesc))))>=3)% standardowo 4
                %selecteditems=draw_rect(selecteditems,minxval(a)-
4,minyval(a)-4,maxxval(a)+4,maxyval(a)+4);
                if(ALLMARKERSON)
selecteditems=draw_rect(selecteditems,minxval(a)-3,minyval(a)-
3,maxxval(a)+3,maxyval(a)+3); end;
                items(a)=3;
                itemlistlen=itemlistlen+1;
                itemlist(itemlistlen)=a;
                %a
           end;

    end;
end;

for a=1:1:lastitem      %%rozpoznawanie obrazow usta - wzorzec w
tolerancji - wartosci bezwzgledne
    if(items(a)==2)
        if(sum((abs(funhu(:,a))>=(abs(lippattdesc)-
tol*abs(lippattdesc)))&(abs(funhu(:,a))<=(abs(lippattdesc)+tol*abs(lippat
tdesc))))>=3)% standardowo 4
                if(ALLMARKERSON)
selecteditems=draw_rect(selecteditems,minxval(a)-3,minyval(a)-
3,maxxval(a)+3,maxyval(a)+3); end;
                %selecteditems=draw_rect(selecteditems,minxval(a)-
4,minyval(a)-4,maxxval(a)+4,maxyval(a)+4);
                items(a)=3;
                itemlistlen=itemlistlen+1;
                itemlist(itemlistlen)=a;
                %a
           end;
```

```
    end;
end;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
5

%klasyfikowanie metoda binaryzacji ze zmniennym progiem i szukanie oka

if(0)

OFFSET=5;   %ile obrazu wiecej brac do testow
bin_imagetmp=zeros(dx,dy);  %pojedyncze obrazy probek
tmpitems=zeros(1,alot);    %jezeli = 1 to znaczy, ze obiekt istnieje
tmplastitem=0;             %ostatnia jedynka
tmpminxval=5000*ones(1,alot);  %min x value to lewy skrajny punkt danego
obiektu
tmpminyval=5000*ones(1,alot);
tmpmaxyval=zeros(1,alot);
tmpmaxxval=zeros(1,alot);
mybreak=0;


for a=1:1:lastitem      %%wstepna klasyfikacja obiektow
    if(items(a)==3)
        a;
        for border=50:20:250    %binaryzacja z zadanym progiem
            for x=minxval(a)-OFFSET:1:maxxval(a)+OFFSET
                for y=minyval(a)-OFFSET:1:maxyval(a)+OFFSET
                    if(grey_image(x,y)<=border) bin_imagetmp(x,y)=0; else
bin_imagetmp(x,y)=1; end;
                end;
            end;

bin_imagetmp=label_images(bin_imagetmp,minxval(a),minyval(a),maxxval(a),m
axyval(a));%labeling

tmpitems=zeros(1,alot);tmplastitem=0;tmpminxval=5000*ones(1,alot);tmpminy
val=5000*ones(1,alot);tmpmaxyval=zeros(1,alot);tmpmaxxval=zeros(1,alot);

[tmpitems,tmplastitem,tmpminxval,tmpminyval,tmpmaxxval,tmpmaxyval]=get_im
ageinfo(bin_imagetmp,minxval(a),minyval(a),maxxval(a),maxyval(a));
            if(tmplastitem>0)   %jezeli rozpoznal cokolwiek na obrazie
                for b=1:1:tmplastitem
                    if(tmpitems(b)==1)  %spelnia rozmiar
                        if(((tmpmaxyval(b)-
tmpminyval(b))>=MINPATY)&((tmpmaxyval(b)-tmpminyval(b))<=MAXPATY))
                            sequ=0; %zmienna biale>czarne>biale= jak = 3
to oko
                            lengt=0;%jak dlugi jest obszar (np kazdy z
odcinkow po 5 pixeli)

x1=tmpminxval(b);x2=tmpmaxxval(b);y1=tmpminyval(b);y2=tmpmaxyval(b);
                            a_val=1*((x2-x1)/(y2-y1));%prosta
przechodzaca od lewego kacika do prawego
                            b_val=x1-a_val*y1;
                            %for y=y1:1:y2%
                            %    x=round(a_val*(y1+y2-y)+b_val);
                            %    if((sequ==0)&(bin_imagetmp(x,y)==0))
lengt=lengt+1; if(lengt>=5) sequ=1; lengt=0; end; end;
                            %    if((sequ==1)&(bin_imagetmp(x,y)==1))
lengt=lengt+1; if(lengt>=5) sequ=2; lengt=0; end; end;
```

```
                            %    if((sequ==2)&(bin_imagetmp(x,y)==0))
lengt=lengt+1; if(lengt>=5) sequ=3; lengt=0; end; end;
                            %    if((sequ==3))
selecteditems=draw_rect(selecteditems,minxval(a)-1,minyval(a)-
1,maxxval(a)+1,maxyval(a)+1); sequ=0; mybreak=1; break; end;
                            %end;
                            for x=x1:1:x2
                                if(mybreak==0) sequ=0; for y=y1:1:y2%
prosta przechodzaca w polowie wysokosci od lewej do prawej
                                    %x=round((x2+x1)/2);
                                    if((sequ==0)&(bin_imagetmp(x,y)==1))
lengt=lengt+1; if(lengt>=2) sequ=1; lengt=0; end; end;
                                    if((sequ==1)&(bin_imagetmp(x,y)==0))
lengt=lengt+1; if(lengt>=5) sequ=2; lengt=0; end; end;
                                    if((sequ==2)&(bin_imagetmp(x,y)==1))
lengt=lengt+1; if(lengt>=5) sequ=3; lengt=0; end; end;
                                    if((sequ==3)&(bin_imagetmp(x,y)==0))
lengt=lengt+1; if(lengt>=5) sequ=4; lengt=0; end; end;
                                    if((sequ==4)&(bin_imagetmp(x,y)==1))
lengt=lengt+1; if(lengt>=2) sequ=5; lengt=0; end; end;
                                    if((sequ==5))
selecteditems=draw_rect(selecteditems,minxval(a)-1,minyval(a)-
1,maxxval(a)+1,maxyval(a)+1); mybreak=1; sequ=0; break; end;
                                    %selecteditems(x,y)=255;
                                end; end;
                                if(mybreak==1) break; end;
                            end;

                        end;
                    end;
                    if(mybreak==1) mybreak=0; break; end;
                end;
            end;




        end;
    end;
end;

end; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%koniec klasyfikacji ze zmniennym
progiem

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%polozenie poszczegolnych obiektow wzgledem siebie

maxheightdif=30;   %jaka jest max dopuszczalna roznica wysokosci pomiedzy
oczami
maxwidthdif=109;   %max dopuszczalna odleglosc pomiedzy oczami
minwidthdif=50;    %min dopuszczalny rozstaw oczu
forbidheigh=40;     %tyle pixeli pod obiektem nie moze wystapic inny
obiekt (czy badany obiekt nie jest brwia)
forbidwidth=30;     % ale tylko pod obiektem nie dalej w bok niz ...
pixeli
okflag=1;           %jezeli 1 to pod obiektem nie ma innego
poteneyeleft=zeros(1,555);%numer obiektu klasyfikowanego jako oko lewe
poteneyeright=zeros(1,555);%numer obiektu klasyfikowanego jako oko prawe,
zawsze para z lewym - ten sam numer jedna para
noofpoteyes=0;              %ile potencjalnych oczu
prevframeeyescordx=iprevcentx;%polozenie oczu w poprzedniej klatce - dla
1 klatki to srodek ekranu
prevframeeyescordy=iprevcenty;%iprevcentx,iprevcenty
```

```
prevframetmplen=9999999999;       %odleglosc badanej pary od tej z poprz
klatki
bestpoteneyes=0;            %ktora para jest najblizej najlepsza


for a=1:1:itemlistlen-1 %badanie polozenia wzgledem siebie
    okflag=1;
    for c=1:1:itemlistlen%czy blisko pod badanym obiektem jest pusto (czy
obiekt nie jest brwiami)

if((centx(itemlist(c))>(centx(itemlist(a))))&(centx(itemlist(c))<=(centx(
itemlist(a))+forbidheigh)))%wysokosc
            if((centy(itemlist(a))>=(centy(itemlist(c))-
forbidwidth))&(centy(itemlist(a))<=(centy(itemlist(c))+forbidwidth)))
%szerokosc
                okflag=0;
            end;
        end;
     end;
         if(okflag==1)
            for b=(a+1):1:itemlistlen
                if((centx(itemlist(a))>=(centx(itemlist(b))-
maxheightdif))&(centx(itemlist(a))<=(centx(itemlist(b))+maxheightdif)))%c
zy sa na podobnej wyskokosci

%selecteditems=draw_rect(selecteditems,minxval(itemlist(a))-
5,minyval(itemlist(a))-5,maxxval(itemlist(a))+5,maxyval(itemlist(a))+5);

%selecteditems=draw_rect(selecteditems,minxval(itemlist(b))-
5,minyval(itemlist(b))-5,maxxval(itemlist(b))+5,maxyval(itemlist(b))+5);
                    if((centy(itemlist(a))>=(centy(itemlist(b))-
maxwidthdif))&(centy(itemlist(a))<=(centy(itemlist(b))+maxwidthdif)))%
rozstaw max szer

%selecteditems=draw_rect(selecteditems,minxval(itemlist(a))-
7,minyval(itemlist(a))-7,maxxval(itemlist(a))+7,maxyval(itemlist(a))+7);

%selecteditems=draw_rect(selecteditems,minxval(itemlist(b))-
7,minyval(itemlist(b))-7,maxxval(itemlist(b))+7,maxyval(itemlist(b))+7);
                        if((centy(itemlist(a))<=(centy(itemlist(b))-
minwidthdif))|(centy(itemlist(a))>=(centy(itemlist(b))+minwidthdif)))%
rozstaw min szer

selecteditems=draw_rect(selecteditems,minxval(itemlist(a))-
5,minyval(itemlist(a))-5,maxxval(itemlist(a))+5,maxyval(itemlist(a))+5);

selecteditems=draw_rect(selecteditems,minxval(itemlist(b))-
5,minyval(itemlist(b))-5,maxxval(itemlist(b))+5,maxyval(itemlist(b))+5);
                            noofpoteyes=noofpoteyes+1;  %zapis numeru
obiektu potencjalnej pary oczu
                            if(centy(itemlist(a))<centy(itemlist(b)))
poteneyeleft(noofpoteyes)=itemlist(a);
poteneyeright(noofpoteyes)=itemlist(b);
                                else
poteneyeleft(noofpoteyes)=itemlist(b);
poteneyeright(noofpoteyes)=itemlist(a);
                            end;
                        end;
                    end;

                end;
            end;

    end;
```

```
end;

if(noofpoteyes>=1)%jezeli wykryto wiecej niz 1 pare oczu na ekranie,
wybieram te najblizej tych z poprzedniej klatki
    for a=1:1:noofpoteyes
        if( ((((centy(poteneyeleft(a))+centy(poteneyeright(a)))/2)-
prevframeeyescordy)^2) +
((((centx(poteneyeleft(a))+centx(poteneyeright(a)))/2)-
prevframeeyescordx)^2)<prevframetmplen  )

prevframetmplen=((((centy(poteneyeleft(a))+centy(poteneyeright(a)))/2)-
prevframeeyescordy)^2) +
((((centx(poteneyeleft(a))+centx(poteneyeright(a)))/2)-
prevframeeyescordx)^2);
            bestpoteneyes=a;
        end;
    end;
else
    %bestpoteneyes=1;
end;

bestpoteneyes

leftheight=0;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% mierzenie wysokosci
oka, czy otwarte czy zamkniete
rightheight=0;

if(bestpoteneyes>0)     % jezeli znalazl oczy

selecteditems=draw_rect(selecteditems,minxval(poteneyeleft(bestpoteneyes)
)-7,minyval(poteneyeleft(bestpoteneyes))-
7,maxxval(poteneyeright(bestpoteneyes))+7,maxyval(poteneyeright(bestpoten
eyes))+7);

    tmp=poteneyeleft(bestpoteneyes);
    for x=minxval(tmp):1:maxxval(tmp)
     %if(r1_image(x,y)==tmp;
     y=round(centy(tmp)-((maxyval(tmp)-minyval(tmp))/8));
        if(r1_image(x,y)>0)
            leftheight=leftheight+1;
            selecteditems(x,y)=128;
        end;
        y=round(centy(tmp));
        if(r1_image(x,y)>0)
            leftheight=leftheight+1;
            selecteditems(x,y)=128;
        end;
        y=round(centy(tmp)+((maxyval(tmp)-minyval(tmp))/8));
        if(r1_image(x,y)>0)
            leftheight=leftheight+1;
            selecteditems(x,y)=128;
        end;
    end;

    tmp=poteneyeright(bestpoteneyes);
    for x=minxval(tmp):1:maxxval(tmp)
        %if(r1_image(x,y)==tmp;
        y=round(centy(tmp)-((maxyval(tmp)-minyval(tmp))/8));
        if(r1_image(x,y)>0)
            rightheight=rightheight+1;
            selecteditems(x,y)=128;
        end;
        y=round(centy(tmp));
        if(r1_image(x,y)>0)
```

```
            rightheight=rightheight+1;
            selecteditems(x,y)=128;
        end;
        y=round(centy(tmp)+((maxyval(tmp)-minyval(tmp))/8));
        if(r1_image(x,y)>0)
            rightheight=rightheight+1;
            selecteditems(x,y)=128;
        end;
    end;
    %leftheight
    %rightheight
end;


if(0)   %pokaz klatke
%%%%%%%%%%tymzasowo - potem usunac
min_value=min(min(r1_image));   %normalizacja liniowa
    r1_image=r1_image-min_value;
    max_value=max(max(r1_image));
    r1_image=round(255*r1_image/max_value);



new_image=r1_image;

new_image3(:,:,1)=selecteditems;
%new_image3(:,:,2)=grey_image;
new_image3(:,:,2)=new_image;
new_image3(:,:,3)=new_image;



new_image=uint8(new_image3);

image(new_image);
end;

rframe=selecteditems;
if(bestpoteneyes>0)
    rleftcentx=centx(poteneyeleft(bestpoteneyes));
    rleftcenty=centy(poteneyeleft(bestpoteneyes));
    rrightcentx=centx(poteneyeright(bestpoteneyes));
    rrightcenty=centy(poteneyeright(bestpoteneyes));
    rleftheig=leftheight;
    rrightheig=rightheight;
    rfound=1;
else
    rleftcentx=round(dx/2);
    rleftcenty=round(dy/2);
    rrightcentx=round(dx/2);
    rrightcenty=round(dy/2);;
    rleftheig=0;
    rrightheig=0;
    rfound=0;
end;
```

drawbmpfile.m – funkcja rysuje na klatce filmu grafikę z monochromatycznej mapy bitowej

```
function [result]=draw_bmpfile(where,what,sx,sy,brush1,brush2)
```

```
%obrazglowny=obrazglowny,co rysowac, pocztek rysowania w
obrazglowny,pedzel jak 0, pedzel jak 1
result=where;

dx=length(what(:,1,1));
dy=length(what(1,:,1));

for a=sx:1:sx+dx-1
    for b=sy:1:sy+dy-1
        if(what(a-sx+1,b-sy+1,1)==0)
            result(a,b)=brush1;
        else
            if(what(a-sx+1,b-sy+1,1)==255)
                result(a,b)=brush2;
            end;
        end;
    end;
end;
```

drawfilledrect.m – funkcja rysuje na klatce filmu zamalowany prostokąt

```
function [result]=draw_filledrect(where,sx,sy,ex,ey)
result=where;

for aa=sx:1:ex
    for bb=sy:1:ey
        result(aa,bb)=255;
    end;
end;
```

label_images.m – funkcja przyjmuje binarną klatkę filmu, procedura etykietuje obiekty.

```
function [result]=label_images(source,sx,sy,ex,ey)
result=source;


                    %labeling - indeksacja obiektow
cur_ind=0;                  %index ktory teraz przyznaje to ..2
alot=333333;
for x=sx:1:ex
    for y=sy:1:ey
        if(result(x,y)==1)
                neig(1)=result(x,y-1);neig(2)=result(x-1,y-1);
                neig(3)=result(x-1,y);neig(4)=result(x-1,y+1);%wartosci
sasiadow
                for a=1:1:4 if(neig(a)==0) neig(a)=alot; end;end;
                neig_min=min(neig);
                if(neig_min==alot)
                    if(cur_ind==0)
                        cur_ind=1;
                        result(x,y)=1;
                    else
                        cur_ind=cur_ind+1;
                        result(x,y)=cur_ind;
                    end;
                else
                    result(x,y)=neig_min;
```

```
                    neig_sort=sort(neig);
                    change2=0;change3=0;change4=0;%ktorych sasiadow
nalezy zmienic
                    if(neig_sort(2)<alot)
                        if(neig_sort(2)~=neig_min)
                            change2=1;
                        end;
                        if(neig_sort(3)<alot)
                            if(neig_sort(3)~=neig_sort(2))
                                change3=1;
                            end;
                            if(neig_sort(4)<alot)
                                if(neig_sort(4)~=neig_sort(3))
                                    change4=1;
                                end;
                            end;

                        end;

                        if((change2==1)|(change3==1)|(change4==1))
                            for xx=sx:1:ex
                                for yy=sy:1:ey
                                    if(result(xx,yy)>1)
                                        if(change2)

if(result(xx,yy)==neig_sort(2))
                                                result(xx,yy)=neig_min;
                                            end;
                                        end;
                                        if(change3)

if(result(xx,yy)==neig_sort(3))
                                                result(xx,yy)=neig_min;
                                            end;
                                        end;
                                        if(change4)

if(result(xx,yy)==neig_sort(4))
                                                result(xx,yy)=neig_min;
                                            end;
                                        end;
                                    end;
                                end;
                            end;
                        end;

                    end;
                end;

        end;
    end;
end;
```

draw_rect.m – funkcja rysuje nie zamalowany prostokąt na klatce filmu.

```
function [result]=draw_rect(where,sx,sy,ex,ey)
result=where;
if(sx<1) sx=1; end;
if(sy<1) sy=1; end;
if(sx>479) sx=479; end;
if(sy>639) sy=639; end;
```

```
for aa=sx:1:ex
    result(aa,sy)=255;
    result(aa,ey)=255;
end;
for aa=sy:1:ey
    result(sx,aa)=255;
    result(ex,aa)=255;
end;
```