Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

PRACA MAGISTERSKA

MATEUSZ MAZUR

SYSTEM DETEKCJI I ROZPOZNAWANIA ZNAKÓW DROGOWYCH

PROMOTOR:
dr Adrian Horzyk

Kraków 2009

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

.....

PODPIS

AGH
University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

MASTER OF SCIENCE THESIS

MATEUSZ MAZUR

ROAD SIGN DETECTION AND RECOGNITION SYSTEM

SUPERVISOR:
Adrian Horzyk Ph.D

Krakow 2009

Serdecznie dziękuję
doktorowi Adrianowi Horzykowi za umożliwienie mi podjęcia interesującej mnie tematyki pracy oraz liczne wskazówki podczas jej realizacji.

Spis treści

Spis rysunków

Spis tablic

1. Wstęp

Stworzenie pojazdu poruszającego się samodzielnie po ulicach jest celem wielu naukowców i konsorcjów motoryzacyjnych. Zagadnienie to stwarza wiele problemów. Jak taka maszyna ma przewidywać i reagować na zachowania ludzi za kierownicą? Jak wiemy z własnego doświadczenia wielu z nich zachowuje się za kierownicą nieprzewidywalnie. Innym problemem jest analiza otoczenia. Człowiek postrzega swoje otoczenie za pomocą kilku doskonałych zmysłów. Czy jest możliwe ich odwzorowanie? Jednym z niezastąpionych zmysłów jest wzrok. Człowiek nie tylko dostrzega swoje otoczenie, potrafi je analizować, rozpoznawać przeróżne przedmioty, nawet takie, które swoim kształtem znacznie odbiegają od poznanych pierwowzorów. Czy tak doskonałą zdolność jesteśmy w stanie powielić w maszynie? Jednym z ciekawszych badań jest projekt armii amerykańskiej o nazwie AVIP (*The U.S. Army's Vehicle Intelligence Program*). Są to badania nad zwiększeniem bezpieczeństwa pojazdów poprzez systemy ostrzegające kierowcę przed niebezpieczeństwami, pomagające w bezpiecznej i ekonomicznej jeździe, następnie częściowo kontrolujące pojazd, aż do autonomicznych funkcji kontroli pojazdu.

Badania w tej dziedzinie są prowadzone od dawna. Jak widać nie tylko na uniwersytetach, lecz również w laboratoriach wojskowych i prywatnych. Zagadnienie to jest bardzo interesujące i może w znacznym stopniu zwiększyć bezpieczeństwo na drogach. Przykładów już działających systemów nie trzeba szukać w armii. Najnowszy model Opla o nazwie Insignia został wyposażony w kamerę wideo. Przy jej pomocy samochód rozpoznaje ograniczenia prędkości, ostrzega również przy niezamierzonych zmianach pasa ruchu. Podobną funkcjonalność możemy znaleźć w nawigacji Blaupunkta, która również jest w stanie rozpoznawać znaki drogowe (Blaupunkt TravelPilot 500/700).

1.1. Postawienie problemu

Celem pracy magisterskiej jest opracowanie i implementacja systemu detekcji i rozpoznawania znaków drogowych. Systemy takie stają się coraz popularniejsze na świecie i istnieje już wiele komercyjnych rozwiązań.

1.2. Cele

Najważniejszym założeniem pracy jest próba opracowania nowego podejścia. Nie chodzi tutaj o odtworzenie już opracowanego systemu, lecz o próbę opracowania czegoś nowego. System powinien być w stanie radzić sobie nawet w trudnych warunkach i przy częściowo przysłoniętych znakach.

Bardzo ważnym założeniem jest również próba osiągnięcia tego celu jak najprostszymi metodami. Analiza obrazu powinna się opierać o jak najmniej złożone obliczeniowo algorytmy. Dzięki takiemu podejściu opracowany system powinien pracować w czasie rzeczywistym na współczesnych komputerach klasy PC. Jest to również ważne założenie. System niewymagający specjalistycznego sprzętu jest tańszy w produkcji i może być przez to szerzej spopularyzowany.

Tak postawiony problem musi napotkać na wiele trudności. Najważniejszymi jest zmiana oświetlenia. Każdy kolor, reprezentowany cyfrowo, w zależności od oświetlenia jest reprezentowany poprzez inny zestaw wartości. Jest to poważny problem, biorąc pod uwagę zmianę pór dnia, czy szybkie zmiany pogody.

Innym problemem jest fakt, że znaki nie są zawsze umiejscowione idealnie prostopadle do drogi. Ich rotacja w innych kierunkach też jest możliwa. Sprawia to dodatkową trudność w procesie rozpoznawania, gdyż piktogram znaku może być spłaszczony bądź przekrzywiony.

Kolejny problem wiąże się z możliwością przysłaniania znaku, na przykład przez gałąź drzewa, czy śnieg. W takim wypadku kształt znaku może ulec zmianie. Te zmiany utrudniają zadanie lokalizacji znaku.

System powinien zmierzyć się ze wszystkimi wymienionymi tu problemami i spróbować przynajmniej w pewnym stopniu je rekompensować i działać poprawnie.

2. Podstawy teoretyczne

2.1. Rys historyczny

2.1.1. Teoria rozpoznawania obrazów

Teoria rozpoznawania obrazów jest ściśle powiązana z pracami i badaniami nad sztuczną inteligencją. Dziedzina ta zajmuje się wykorzystaniem maszyn (głównie komputerów) do zadań, które wymagają ludzkiej inteligencji. Zakres tej dziedziny nieustannie zmienia się z powodu szybkiego rozwoju informatyki. Wiele zagadnień uważanych za wymagające inteligencji, zostało zdewaluowanych na skutek rozwoju techniki. Zmiany te są spowodowane nieprecyzyjnym zdefiniowanie pojęcia *inteligencja*. Nie da się tak zdefiniować tego pojęcia, żeby jednocześnie nadawało się do opisu zachowania człowieka i maszyny. Wiele zagadnień uważanych za klasyczne problemy sztucznej inteligencji są obecnie zbliżone poziomem do zadań szkolnych. Jednym z koronnych przykładów, często wymienianych w pierwszych pracach z tej dziedziny, może być zagadnienie "Wież z Hanoi". Sztuczna inteligencja niezmiennie zajmuje się zagadnieniem rozpoznawania.

Polska nazwa *rozpoznawanie obrazów (ang. pattern recognition)* nie oddaje całego zakresu zagadnienia. Kojarzy się ona z analizą dwuwymiarowych fotografii lub trójwymiarowych scen. Aby sprecyzować pojęcie posłużę się definicją ze skryptu Ryszarda Tadeusiewicza i Mariusza Flasińskiego [32]:

> "Ogólnie w zadaniu rozpoznawania obrazów chodzi o rozpoznawanie przynależności rozmaitego typu obiektów (lub zjawisk) do pewnych klas. Rozpoznawanie to ma być prowadzone w sytuacji braku apriorycznej informacji na temat reguł przynależności obiektów do poszczególnych klas, a jedyna informacja możliwa do wykorzystania przez algorytm lub maszynę rozpoznającą jest zawarta w ciągu uczącym, złożonym z obiektów, dla których znana jest prawidłowa klasyfikacja."

Rozpoznawanie obrazów jest o tyle ciekawe, że maszyna musi klasyfikować obiekty do klas na podstawie wiedzy o kilku reprezentantach. Wiedza jej przekazywana nie jest w postaci algorytmicznej, ponieważ człowiek sam nie wie do końca, w jaki sposób jest w stanie rozpoznać różne przedmioty, kształty, itp. Maszyna musi więc uogólniać wzorce otrzymane podczas procesu uczenia, aby mogła poprawnie funkcjonować. Właśnie z tych powodów dziedzina rozpoznawania obrazów jest pewnego rodzaju poligonem, na którym próbuje się przekazać umiejętności człowieka w klasyfikacji, czy rozpoznaniu. Jest to bardzo atrakcyjny dział nauki. Niestety wszystkie te komplikacje powodują, że zastosowanie już rozwiniętych teorii są znikome, a skuteczność metod wciąż zbyt ograniczona.

Bez względu na obiekty, które chcemy rozpoznawać rozpoczynamy zawsze od tych samych podstaw. Każdy obiekt jest opisany pewnym zestawem wartości, cech. Na nich możemy dokonywać obliczeń w celu zaklasyfikowania obiektu do jednej z klas. W teorii rozpoznawania istnieje wiele metod podejmowania decyzji w oparciu o pewne zestawy cech. Dobór owych zestawów nie został jednak sprecyzowany i zależy od intuicji konstruktora. Wytypowane cechy mogą być oceniane pod względem przydatności do rozpoznania. Można wyróżnić klika podejść do tego zagadnienia.

Podejście całościowe, polegające na wzięciu pod uwagę wszystkich cech opisujących cały obiekt badań i zaklasyfikowaniu go w jednym kroku.

Podejście strukturalne, wyróżniające najpierw określone elementy obiektu i ustalające ich wzajemne relacje.

2.1.2. Wykorzystanie rozpoznawania obrazów dla znaków drogowych

Jedne z pierwszych publikacji związanych z wykorzystaniem rozpoznawania obrazów do klasyfikacji znaków powstały w Japonii w 1984 roku. Ich celem było przetestowanie kilku metod do wykrywania obiektów w zewnętrznym środowisku. W latach dziewięćdziesiątych pojawiło się na świecie kilka grup zainteresowanych tym tematem. Stworzono wiele rozwiązań skupiających się na rozpoznawaniu konkretnych typów znaków, jak i bardziej ogólnych. Detekcja znaków w tych systemach opierała się na różnorodnych rozwiązaniach. Etap klasyfikacji, w większości przypadków, był rozwiązany poprzez wykorzystanie sieci neuronowych.

W badaniach sprzed 1995 roku, zebranych prze Lalonde'a i Li, można zauważyć ogromy wkład europejskich badaczy [22]:

Kanada

Praca Ghica i jego zespołu [16] jest czysto akademicka i prawdopodobnie została opracowana od podstaw. Jest oparta praktycznie tylko na sieciach neuronowych, wykorzystanych do filtracji obrazu (sieć Hopfield'a) oraz rozpoznawania znaków. Wykorzystana sieć jest zdegenerowaną wersją ART, której wyniki są bliższe 'nauczonemu' klasyfikatorowi najbliższego sąsiada z kryterium odrzucenia niż prawdziwej sieci. Cały algorytm działa off-line i składa się z etapu detekcji i progowania opartego na kolorach oraz etapu rozpoznania. W pracy niestety nie zawarto wyników eksperymentów.

Projekt ten był z pewnością innowacyjny. Niestety z powodu małej ilości przeprowadzonych testów, nie da się potwierdzić jego skuteczności.

USA

Bardzo specjalizowane podejście zaprezentował w swojej publikacji [18] Kehtarnavaz. Ograniczało się ono jedynie do rozpoznawania znaków stopu. Tutaj przetwarzanie również odbywało się off-line. Znaki były wykrywane poprzez wyznaczanie czerwonych obszarów oraz ich geometryczną analizę. Przeprowadzono eksperyment z jedenastoma znakami, spośród których pięć było znakami stopu. System rozpoznał wszystkie znaki stopu i odrzucił wszystkie pozostałe. Wyniki te są zachęcające, niestety nie podano czasu przetwarzania.

Kellmeyer i Zwahlen zaprojektowali system detekcji i rozpoznania znaków ostrzegawczych [19]. Wykorzystali sieć neuronową do segmentacji barwnej i detekcji. Tak wybrane regiony, o znanych barwach, są poddane analizie przez kolejną sieć neuronową badającą kształty. Czas przetwarzania dla jednego obrazu to 2 minuty 30 sekund na komputerze PC z procesorem 486MHz, co jest raczej słabym wynikiem. Podano, że skuteczność algorytmu to 75% detekcji przy 40 znakach występujących na 55 obrazach oraz trzy fałszywe alarmy.

Francja

Ciekawe podejście zaproponował Saint-Blancard z firmy Peugeot [10]. Co prawda system rozpoznawał jedynie znaki czerwone, ale uzyskano przetwarzanie w czasie rzeczywistym. Obrazy są pozyskiwane z kamery wyposażonej w czerwony filtr odcinający (ang. red cut-off filter). Dzięki takiemu podejściu nie potrzeba już przeprowadzać segmentacji. Kolejnym krokiem jest wykrywanie krawędzi oraz zamkniętych konturów. Rozpoznanie jest przeprowadzane poprzez wydobycie cech i klasyfikację za pomocą systemu ekspertowego i sieci neuronowej. Cały projekt działa na komputerze AT z kartą graficzną. Cechuje się 94.9% identyfikacją znaków, przy 0.8% błędnej klasyfikacji i 2.6% niepewnych rozpoznań. Średni czas wykonania to około 0.7 sekundy dla wykrywania konturów i klasyfikacji.

Japonia

Jedno z pierwszych podejść do detekcji i rozpoznawania znaków drogowych w czasie rzeczywistym należy do Akatsuka'i i Imai [4]. Projekt powstał już w 1987 roku. Jego założenia skupiają się na rozpoznawaniu ograniczeń prędkości. Są one czerwono-białe z niebieskimi cyframi. Zastosowano podstawowe techniki, jak progowanie kolorów, dopasowywanie wzorców dla wykrycia kształtów czy proste rozpoznawanie cyfr wykorzystujące korelację ze znanymi wzorcami. Dodanie kolejnej klasy znaków

do rozpoznania wymagałoby przeprojektowania całego systemu. Wkład naukowy w tym projekcie jest znikomy. Interesujące jest jedynie wykorzystywanie dedykowanego sprzętu do progowania barwnego i dopasowywania wzorca, umożliwiające przetworzenie klatki filmu w jedną sześćdziesiątą sekundy. Zadanie rozpoznania jest przeprowadzane przez komputer PC-AT w czasie pół sekundy. Brak informacji o skuteczności systemu uniemożliwia ocenę tego rozwiązania.

Włochy

Istotne prace w badaniach nad rozpoznawaniem znaków drogowych zostały przeprowadzone na Uniwersytecie Genuii. Piccioli wraz ze swoim zespołem [27, 28] zaproponował strategię rozpoznawania znaków opartą na rozumowaniu geometrycznym. Z każdego monochromatycznego obrazu drogi wydobywane są linie, umożliwiające detekcję znaków trójkątnych. W celu detekcji znaków kolistych obraz jest przeglądany w poszukiwaniu pierścieni. Etap rozpoznawania jest oparty na korelacji wykrytych znaków ze znormalizowanymi obrazami zapisanymi w bazie danych. Najlepsze wyniki detekcji to 92% z 11 fałszywymi alarmami na ponad 600 obrazach przy czasie przetwarzania równym 6 sekund na obraz, z wykorzystaniem stacji ELC Sparc. Wyniki detekcji dla znaków kolistych są w granicach 93% – 96% w zależności od różnorodności sceny, z większą ilością fałszywych alarmów i czasem przetwarzania jednego obrazu równym 15 sekundom. Rozpoznanie zdaje się być bardzo wydajne z 98% poprawnością klasyfikacji i czasem przetwarznia równym 500 ms, przy bazie danych zawierającej 60 kolistych i 47 trójkątnych znaków.

Niemcy

Niemieckie zespoły naukowców wiodą prym w detekcji i rozpoznawaniu znaków. Grupa badaczy z Daimlera-Benza opisała kompletny system detekcji i rozpoznawania znaków w czasie rzeczywistym [11]. Jest to jeden z najbardziej zaawansowanych prototypów. Moc obliczeniową zapewniają cztery transputery oraz cztery PowerPC. Specjaliści od segmentacji barwnej (Bartneck i Ritter[5]), analizy kształtów (Besserer i jego zespół[6]) oraz rozpoznawania piktogramów (zespół Zhenga[34]) dostarczyli niezbędnych informacji dla detekcji i śledzenia znaków. Jeśli chodzi o wydajności, liczba klas znaków jest rozpoznawana z 90% powodzeniem i 5% generowanych fałszywych alarmów, w czasie poniżej 200ms.

Istotny jest wkład Ritter'a [30] właśnie z tego zespołu, który przedstawia analizę architektury systemu rozpoznawania znaków. Dyskusja skupia się na wiedzy o reprezentacji znaków, o tym jak jest ona uzyskiwana z modelu, czy jak jest ona wykorzystywana w procesie detekcji i rozpoznania.

Inna grupa z Uniwersytetu w Koblenz-Landau, również współpracuje z Daimlerem-Benzem. System opisany przez zespół Priesea [29] wykonuje typowe zadanie segmentacji barwnej, tworzenia list regionów w danych kolorach oraz analizę kształtów. Pomimo typowej tematyki, zaproponowano kilka interesujących rozwiązań. Wydajność systemu nie została podana, a czasy przetwarzania dla najlepszych implementacji to 3.3 sekundy na Sparc'u 10 dla całej klatki oraz 0.77 sekundy dla przetworzenia obszaru, gdzie najprawdopodobniej znajdują się znaki.

Z kolei wstępne badania przedstawione przez zespół Krumbiegela [21] dotyczą wykorzystania wielowarstwowych sieci neuronowych jako ekspertów dla detekcji i rozpoznania znaków. Ponieważ projekt był ciągle w fazie rozwoju nie jest znana jego wydajność.

Podsumowanie

Powyższy przegląd pokazuje jak wiele różnych podejść zastosowano do zagadnienia rozpoznawania i detekcji znaków drogowych. Głównymi problemami, na które napotykały poszczególne zespoły był czas przetwarzania, czy wysoka liczba fałszywych alarmów. Każde z podejść próbowało w inny sposób pokonać wszystkie z trudności i prezentują one powszechne zainteresowanie tą tematyką na świecie. Można również zauważyć dynamiczny rozwój różnych metodyk analizy obrazów.

2.2. Podstawy przetwarzania obrazów

Ważnym aspektem komputerowego przetwarzania obrazu jest jego cyfrowa reprezentacja. Wszystkie cyfrowe przetworniki posiadają skończoną rozdzielczość. Skutkiem tego jest fakt, że obraz cyfrowy jest dyskretyzacją rzeczywistości. Obecne systemy przetwarzania obrazu wykorzystują dwie metody rozmieszczania elementów na obrazach cyfrowych. Jedna z nich to siatka heksagonalna, druga – siatka kwadratowa.

Siatka heksagonalna jest zbliżona do rozmieszczenia receptorów w ludzkim oku. Natomiast Drugi z wymienionych sposobów jest o wiele prostszy i wygodniejszy w obsłudze. Dlatego też jest on najczęściej wykorzystywany w analizie obrazów.

W związku z dyskretyzacją obrazu pojawiają się kilka ważnych pojęć, między innymi **rozdzielczość obrazu**. Cytując za [33]:

> Wyraża się ona ilością elementów podstawowych składających się na obraz. Najczęściej przy płaskich obrazach o kwadratowej siatce zapisywana jest ona jako iloczyn ilości elementów w poziomie i pionie obrazu.

Wybór rozdzielczości jest ważny z punktu widzenia zadania rozpoznania. To ona decyduje o ilości szczegółów, które będą widoczne na obrazie cyfrowym. Decyzja o wyborze rozdzielczości musi być zawsze kompromisem. Z jednej strony chcemy znać jak najwięcej szczegółów. Z drugiej jednak wzrost rozdzielczości powoduje wzrost czasu przetwarzania obrazu - mamy więcej elementów do analizy.

Każdy element tak skonstruowanego obrazu może przyjmować jedną z ustalonych wartości. Ich liczba zależy od ilości bitów, jaką przeznaczymy na zapis informacji o jednostce obrazu. Powszechnie nazywamy ją liczbą kolorów. Tutaj również występuje kompromis doboru odpowiedniej liczby barw. W analizie występuje kilka najpopularniejszych wartości:

binarny – zapis informacji na jednym bicie. To najprostszy zapis, zajmujący najmniej pamięci. Wiele algorytmów operuje właśnie na takich obrazach.

monochromatyczny – zapis informacji na ośmiu bitach. Przy pomocy tego formatu zakodować można 256 odcieni szarości.

kolorowy – zapis na dwudziestu czterech bitach. Najczęściej po osiem bitów zapisuje informacje o kolejnych składowych barwy w przestrzeni RGB. Format ten umożliwia zapis około 17 milionów różnych odcieni kolorów

Istnieje również wiele innych formatów zapisu informacji barwnych. Warto choćby wspomnieć o innych przestrzeniach barwnych, również wykorzystujących trzy składowe, np. HSL.

Można przyjąć, że każdy obraz jest funkcją dwóch zmiennych. W najprostszym przypadku zwracającą pojedynczą wartość:

$$L(m,n) : \mathbb{N}^2 \mapsto \mathbb{N} \tag{2.1}$$

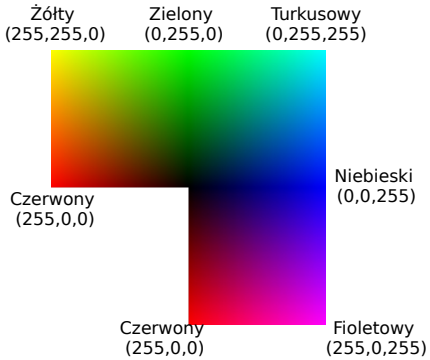
Równanie 2.1 przypomina nam również, że obraz reprezentuje funkcję dyskretną – jest to związane z jego skończoną rozdzielczością.

2.2.1. Przestrzenie barw

Ważnym elementem analizy obrazów kolorowych jest dobór odpowiedniej przestrzeni barw. Wyróżnić możemy wiele sposobów i podejść do tego zagadnienia. Omówimy teraz pokrótce kilka przestrzeni.

RGB

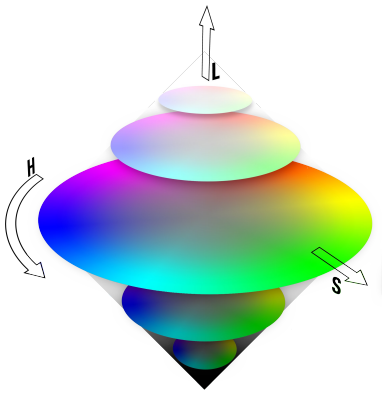
Model barwny RGB (*ang. Red, Green, Blue*), był wzorowany na na właściwościach odbiorczych ludzkiego oka. Był pierwotnie wykorzystywany w technice analogowej. Obecnie został przeniesiony do techniki cyfrowej, gdzie jest powszechnie stosowany. Rysunek 2.1 przedstawia przestrzeń RGB.



Rysunek 2.1: Przestrzeń barw RGB

HSL

Ta przestrzeń była próbą dokładniejszego odwzorowania funkcjonowania ludzkiego oka. Głównym założeniem, oprócz precyzyjniejszego opisu, było zachowanie prostoty modelu RGB. Skrót ten oznacza: odcień (*ang. Hue*), nasycenie (*ang. Saturation*) oraz jasność (*ang. Lightness*). Z doświadczeń przeprowadzonych w tej pracy wynika, że część jej składowych zmienia się w niewielkim stopniu, w zależności od oświetlenia.



Rysunek 2.2: Przestrzeń barw HSL

CIECAM02

Jest to model odwzorowania kolorów oparty na sześciu składowych: jaskrawości (*ang. brightness/luminance*), jasności (*ang. lightness*), barwistości (*ang. colorfulness*), chrominancji (*ang. chroma*), nasyceniu (*ang. saturation*) oraz odcieniowi (*ang. hue*). Jest to dość skomplikowany model barwny. Do obliczania jego wartości brane są pod uwagę parametry takie jak luminancja otoczenia.

2.2.2. Algorytmy przetwarzania obrazu

Wiedza z przetwarzania obrazów jest obecnie dość rozległą nauką. Wykorzystuje ona proste algorytmy analizy, ale też wspomaga się bardziej zaawansowanymi technikami, jak algorytmy genetyczne, czy sieci neuronowe. Przekształceń obrazu można stworzyć nieskończenie wiele. Niestety nie wszystkie mają sens. Ogólnie możemy je podzielić na sześć grup:

- przekształcenia geometryczne
- przekształcenia bezkontekstowe
- przekształcenia kontekstowe

- przekształcenia widmowe
- przekształcenia morfologiczne

Nie będziemy tutaj opisywać wszystkich grup przekształceń. Ich szczegółowy opis można znaleźć w książce Ryszarda Tadeusiewicza i Przemysława Koeohody [33]. Zajmiemy się jedynie opisem binaryzacji oraz kilku przekształceń morfologicznych, które zostały wykorzystane w pracy.

Binaryzacja

Binaryzacja to przekształcenie należące do grupy bez kontekstowych, charakteryzującej się tym, że na wartość elementu nie ma wpływu jego otoczenie. Ważną cechą przekształceń morfologicznych jest to, że dany element obrazu jest zmieniany jedynie, gdy spełniony zostanie zadany warunek logiczny.

Jest to jedna z ważniejszych operacji przeprowadzonych na obrazach cyfrowych. Jest często wykorzystywana w procesie rozpoznawania obrazu. Jej celem jest radykalna redukcja informacji zawartej w obrazie. Binaryzacja polega na zamianie wartości pikseli z odcieni szarości bądź kolorów na wartości 0 lub 1. Sama binaryzacja może być przeprowadzana na wiele sposobów:

binaryzacja z dolnym progiem

$$L'(m,n) = \begin{cases} 0 & dla \quad L(m,n) \leq a \\ 1 & dla \quad L(m,n) > a \end{cases} \tag{2.2}$$

binaryzacja z górnym progiem

$$L'(m,n) = \begin{cases} 0 & dla \quad L(m,n) \geq a \\ 1 & dla \quad L(m,n) < a \end{cases} \tag{2.3}$$

binaryzacja z podwójnym ograniczeniem

$$L'(m,n) = \begin{cases} 0 & dla \quad L(m,n) \leq a_1 \\ 1 & dla \quad a_1 < L(m,n) \leq a_2 \\ 0 & dla \quad L(m,n) > a_2 \end{cases} \tag{2.4}$$

binaryzacja warunkowa

$$L'(m,n) = \begin{cases} 0 & dla \quad L(m,n) \leq a_1 \\ s & dla \quad a_1 < L(m,n) \leq a_2 \\ 1 & dla \quad L(m,n) > a_2 \end{cases} \tag{2.5}$$

binaryzacja wielokryterialna – Operacja przeprowadzania niezależnie na obszarach znacznie różniących się jasnością.

Oznaczenia stosowane we wzorach to:

$L(m,n)$ – wartość punktu w obrazie źródłowym

$L'(m,n)$ – wartość odpowiedniego punktu w obrazie wynikowym

a – próg binaryzacji

a_1, a_2 – progi binaryzacji, takie że $a_1 < a_2$

s – wartość sąsiadujących punktów, $s \in \{0,1\}$

W tej pracy proces binaryzacji odbywa się przy pomocy tablicy LUT:

$$L'(m,n) = LUT(L_r(m,n), L_g(m,n), L_b(m,n)) \tag{2.6}$$

gdzie LUT ma postać:

$$\forall_{r\in[0,256]}\forall_{g\in[0,256]}\forall_{b\in[0,256]} LUT(r,g,b) \in \{0,1\},\ r,g,b \in \mathbb{N} \tag{2.7}$$

natomiast L_r, L_g, L_b to kolorowy obraz źródłowy rozbity na na składowe w przestrzeni RGB.

Erozja

Pierwszym z używanych przekształceń morfologicznych jest erozja. W celu definicji erozji, zakłada się, że istnieje pewien nieregularny obszar X oraz koło B o promieniu r. Erozję można wtedy zdefiniować następująco [33]:

Definicja 1 *Figura zerodowana to zbiór środków wszystkich kół o promieniu r, które w całości są zawarte we wnętrzu obszaru X.*

lub

Definicja 2 *Koło B przetacza się po **wewnętrznej** stronie brzegu figury X. Kolejne położenia środka koła wyznaczają brzeg figury zerodowanej.*

W implementacji na komputerze erozja jednostkowa polega na wyzerowaniu wszystkich punktów obrazu o wartości 1, które graniczą choćby z jednym punktem o wartości 0.

Dylatacja

Jest to przekształcenie odwrotne do erozji. Tutaj również zakładamy istnienie nieregularnego obszaru X oraz koła B o promieniu r, wtedy dylatacje możemy zdefiniować za [33] następująco:

Definicja 3 *Figura po dylatacji jest zbiorem środków wszystkich kół B, dla których choć jeden punkt pokrywa się z jakimkolwiek punktem figury X.*

lub

Definicja 4 *Koło B przetacza się po zewnętrznej stronie brzegu figury X. Kolejne położenia środka koła B wyznaczają brzeg figury po dylatacji.*

Implementacja polega na zamianie wartości wszystkich punktów o wartości 0, które sąsiadują chociaż z jednym punktem o wartości 1, na tę właśnie wartość.

Otwarcie

Otwarcie jest operacją morfologiczną polegającą na przeprowadzeniu erozji, a następnie dylatacji obrazu. Jest to ciekawa operacja przydatna np. przy usuwaniu drobnych zakłóceń z obrazu.

2.3. Transformata Hough

Klasyczna transformata Hough służyła do identyfikacji linii na obrazach cyfrowych. Później jednak została ona rozszerzona o możliwości lokalizowania innych kształtów, najczęściej okręgów i elips. Została ona wynaleziona przez Richarda Dudę i Petera Harta w 1972 roku i nazwana "Uogólnioną Transformatą Hough" po związanym z nią patencie firmy IBM z 1962 roku, opracowanym przez Paula Hough. Została ona spopularyzowana w cyfrowym przetwarzaniu obrazów przez Dana H. Ballarda poprzez artykuł pod tytułem "Uogólnienie transformaty Hough do wykrywania dowolnych kształtów" (*ang. "Generalizing the Hough transform to detect arbitrary shapes"*) w 1981 roku.

W zautomatyzowanym procesie analizy obrazu często zachodzi potrzeba wykrywania kształtów czy linii, w celu identyfikacji interesujących obiektów. W wielu przypadkach na początku wykorzystuje się detektor krawędzi jako pierwszy krok takiego procesu. Dzięki temu otrzymujemy obraz zawierający tylko interesujące nas elementy. Niestety, ze względu na zakłócenia, jak i niedoskonałości detektorów, często otrzymane w ten sposób linie są niekompletne, bądź są one zakłócone i mniej regularne niż faktyczne kształty. Transformata Hough pomaga uporządkować tak uzyskane piksele w linie, elipsy, czy koła. Jest to uzyskiwane przy pomocy jednoznacznej procedury głosującej, działającej na zbiorze sparametryzowanych obiektów obrazu.

2.3.1. Wykrywanie linii

Najprostszym przypadkiem transformaty Hough jest transformata liniowa, wykrywająca proste linie. W przestrzeni obrazu możemy ją opisać równaniem:

$$y = mx + b \tag{2.8}$$

Można ją zobrazować dla pary punktów (x, y). Główny pomysł transformaty opiera się na innym zapisie linii – nie jako par (x, y), lecz za pomocą innych jej parametrów. W przypadku równania 2.8 będą to parametry m oraz b. Przy takim podejściu linia 2.8 może być zapisana w przestrzeni parametrów jako punkt (b, m). Jednakże taki zapis linii pionowych prowadzi do nieograniczonego wzrostu parametrów b, m. Ze względów obliczeniowych, lepszym rozwiązaniem jest sparametryzować je innymi parametrami, powszechnie oznaczanymi jako r i θ. Parametr r oznacza odległość pomiędzy linią, a środkiem układu współrzędnych, natomiast θ, to kąt wektora reprezentującego tą odległość. Po zastosowaniu tych parametrów, równanie prostej przyjmuje postać:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \left(\frac{r}{\sin\theta}\right), \tag{2.9}$$

można je również przekształcić do postaci:

$$r = x\cos\theta + y\sin\theta. \tag{2.10}$$

W ten sposób każdej linii obrazu można przypisać unikalną parę (r, θ), pod warunkiem, że $\theta \in [0, \pi]$ oraz $r \in \mathbb{R}$ albo $\theta \in [0, 2\pi]$ a $r \geq 0$. Przestrzeń par (r, θ) jest często nazywana przestrzenią Hough dla zbioru prostych linii w dwóch wymiarach.

Nieskończona liczba linii może przechodzić przez jeden punkt przestrzeni. Jeśli punkt ten ma na obrazie współrzędne (x_0, y_0), wszystkie linie przechodzące przez ten punkt spełniają równanie:

$$r(\theta) = x_0\cos\theta + y_0\sin\theta \tag{2.11}$$

Jest to odpowiednik sinusoidalnej krzywej w przestrzeni (r, θ), która jest unikalna dla tego punktu. Jeśli krzywe odpowiadające dwóm punktom przecinają się (w przestrzeni Hough), przecięcie to odpowiada linii (w oryginalnym obrazie) przechodzącej przez oba te punkty. Ogólniej mówiąc zbiór punktów stanowiących prostą linię, będzie przedstawiony w przestrzeni Hough jako przecięcie wielu sinusoid, a punkt ich przecięcia będzie wyznaczał parametry tej prostej.

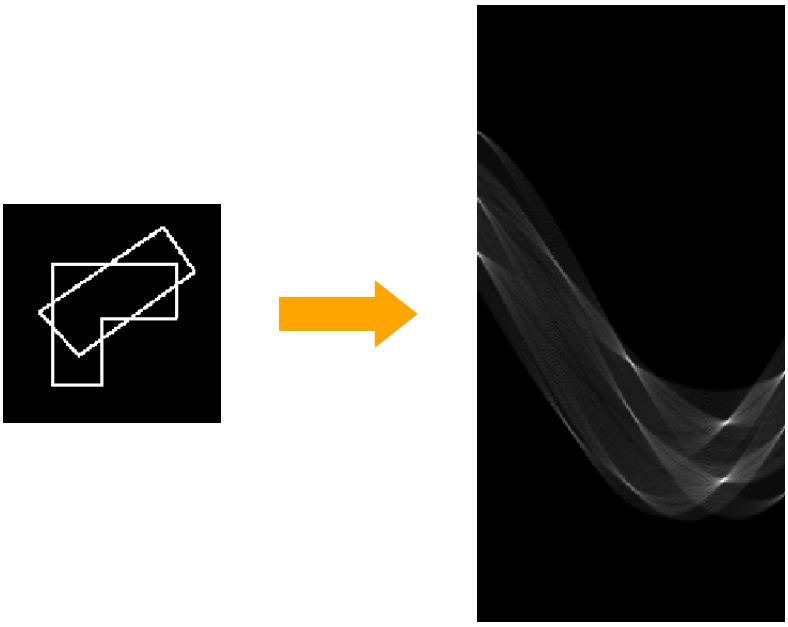
Aby wygenerować transformatę Hough dla linii na naszym obrazie, wybieramy konkretny krok kąta pochylenia linii (np. 10°), a następnie iterujemy przez wszystkie kąty wyznaczone przez ten krok. Dla każdego θ rozwiązujemy równanie 2.10 i powiększamy o jeden wartość punktu (r, θ) w przestrzeni Hough, dla którego jest ono spełnione. Proces ten jest nazywany "głosowaniem" punktów (x, y) na linię (r, θ). Rezultat takiego postępowania przedstawiono poniżej:

2.3.2. Wykrywanie okręgów

Transformata Hough może być również wykorzystana do wykrywania okręgów. Koło można sparametryzować i zapisać przy pomocy równania:

$$(x - a)^2 + (y - b)^2 = r^2 \tag{2.12}$$

W tym wypadku para (a, b) to współrzędne środka okręgu, r to jego promień. Parametr r można przyjąć za stały, co uprości obliczenia i ich czas. Transformata będzie wtedy wykrywać okręgi o zadanym promieniu.



Rysunek 2.3: Liniowa Transformata Hough

2.4. CiSS (Kolista Przestrzeń Próbkowania)

Aby z powodzeniem rozpoznawać piktogramy, należy opisać je w taki sposób, żeby ich cechy nie zależały od rotacji. CiSS, czyli Circular Sampling Space, jest to trójwymiarowa przestrzeń budowana poprzez rzutowanie badanego obrazu na okręgi o danych promieniach. Została opisana przez Kima i Arujo [20]. Niech poszukiwany kształt, oznaczony literą Q, będzie funkcją $Q : D \mapsto 0, 1$, gdzie domena D jest okręgiem. Kształt może nie być ciągły, może też posiadać otwory, co pokazano na rysunku 2.4. Celem algorytmu jest przeanalizowanie obrazu binarnego A w poszukiwaniu kształtów Q,

Rysunek 2.4: Przykłady domen w CiSS

które mogą mieć dowolną rotację. Środek i promień domeny może być zmieniony w celu określenia części kształtu, której będziemy szukać.

Kolista przestrzeń próbkowania dwuwymiarowego obrazu wejściowego $A : \mathbb{R}^2 \mapsto \mathbb{R}$ jest funkcją $C_A : \mathbb{R}^2 \times \mathbb{R}^+ \mapsto \mathbb{R}$ zdefiniowaną następująco:

$$C_A(x, y, r) = \int_0^{2\pi} A(x + r\cos\theta + r\sin\theta)d\theta \tag{2.13}$$

Widać, że $C_A(x, y, r)$ jest sumą wszystkich pikseli o wartości jeden leżących w odległości r od punktu (x, y). Do wydajnego wyznaczenia okręgu na obrazie cyfrowym, można wykorzystać algorytm Bresenhama. W celu przedstawienia poszukiwanego symbolu w przestrzeni, zostają wyznaczone wartości okręgów o różnych promieniach, zmienianych co ustalony krok (np. co 2). Rysunek 2.5 przedstawia przykład przestrzeni na piktogramie znaku. Aby porównać zapisany w ten sposób piktogram z badanym obiektem, najpierw wyznaczamy dla danego obiektu przestrzeń CiSS, najlepiej o takich samych parametrach (krok, maksymalny promień). Następnie wykorzystujemy funkcję odległości w celu oceny

Rysunek 2.5: Przykład przestrzeni CiSS dla piktogramu “Uwaga na spadające odłamki skalne”

dopasowania:

$$CD_{A,Q}(x,y) = \frac{1}{K} \min_{x,y} \sum_{k=0}^{K-1} |C_A(x,y,r_k) - C_Q(\bar{x},\bar{y},r_k)| \tag{2.14}$$

Ponieważ badane obiekty posiadają zakłócenia, akceptowalny próg został ustawiony na 11%. Może się zdarzyć, że dwa obrazy będą miały identyczny obraz w CiSS. Można wtedy taki obraz dodatkowo sprawdzić w przestrzeni RaSS również opisanej w artykule [20]. Tutaj niestety złożoność obliczeniowa jest nieco większa.

2.5. Wady i zalety różnych podejść

Podczas doboru algorytmów dla Systemu Detekcji i Rozpoznawania Znaków Drogowych głównym kryterium była możliwie najmniejsza złożoność obliczeniowa. Wiele interesujących podejść, które zapowiadały się naprawdę ciekawie okazywało się nie spełniać tego kryterium. Inne okazywały się proste, ale nie dość skuteczne.

Pierwszy etap, czyli progowanie barwne, można przeprowadzać w różnych przestrzeniach kolorów. Najczęściej wykorzystywanymi przestrzeniami są RGB i HSI. Na obu tych przestrzeniach zaproponowano działające systemy detekcji znaków ([31, 9]). Wykorzystanie przestrzeni HSI wiąże się z konwersją, ponieważ większość filmów jest kodowana w standardzie RGB. Po przeprowadzeniu testów okazało się, że bardziej opłacalnym jest jednak wykorzystanie progowania w przestrzeni HSI, gdyż umożliwiało to lepsze zawężenie obszarów zainteresowań.

Najlepszym przykładem algorytmu, który okazał się nieefektywny, może być SIFT. Idea tego podejścia oraz prezentowane rezultaty uzyskiwane w publikacjach [14, 12] były zachęcające. Niestety przy próbie wykorzystania tej transformacji okazało się, że jedna klatka filmu o rozdzielczości 240x320, przy wyborze tylko pewnych obszarów do analizy, jest przetwarzana zbyt długo. Dodatkową wadą było podejścia było wydobywanie zbyt małej liczby cech dla znaków których rozmiary nie przekraczały 40 pikseli.

O wiele szybszym sposobem okazało się wykorzystanie najpierw poszukiwania poszczególnych kształtów (trójkątów, okręgów, prostokątów) przy pomocy transformaty Hough, a następnie analiza ich zawartości przy użyciu przestrzeni CiSS. Szczegóły opracowanej metody zostaną przedstawione w dalszej części pracy.

3. Istniejące rozwiązania

Zagadnienie detekcji i rozpoznawania znaków drogowych staje się obecnie bardzo popularne. Wiele uniwersytetów zajmuje się znalezieniem skutecznej metody analizy obrazu pod tym kątem, celem zwiększenia bezpieczeństwa na drogach.

Badania można podzielić na dwa najczęstsze podejścia:

1. Segmentację poprzez progi kolorów, wykrywanie obszarów oraz analizę kształtów.
2. Segmentację poprzez wykrywanie krawędzi na czarno-białym obrazie i ich analizę.

Pierwsze podejście zawiera dwie grupy algorytmów. Jedne opierają się na standardowych przestrzeniach kolorów, takich jak RGB. Inne wykorzystują bardziej wysublimowane, nie wrażliwe na zmiany oświetlenia przestrzenie, jak HSI. Pomimo różnych podejść, żadna z powyższych metod nie daje idealnych rezultatów w każdych warunkach.

W rozdziale tym opiszemy kilka istniejących podejść. Można zauważyć, że każde z nich inaczej podchodzi do problemu. Wskazuje to na dużą gamę możliwości podejścia do tego zagadnienia.

3.1. Road Sign Detection and Recognition

Jeden z zaproponowanych systemów, opierających się na przestrzeni barw RGB, został zaproponowany przez Michaela Shneiera[31].

Aplikacja przetwarza strumienie wideo pobrane z kamery umieszczonej na dachu samochodu testowego. Ponieważ film zawiera przeplot, a pojazd porusza się, można zaobserwować znaczne zakłócenia pomiędzy polami na obrazie. W celu zminimalizowania tego zjawiska, z obrazu są pobierane jedynie co drugie wiersze i kolumny. Znaki są wykrywane w wielostopniowym procesie, rozpoczynającym się od segmentacji opartej o kolory. Z powodu zmian otaczającego oświetlenia, nie jest możliwe poszukiwanie konkretnych wartości czerwonego, zielonego i niebieskiego. Aby rozwiązać ten problem, zostały wykorzystane stosunki składowych RGB. Na przykład dla znaków ostrzegawczych, wartości czerwonego (r), zielonego (g) oraz niebieskiego (b), dla każdego piksela, muszą spełniać równanie:

$$r/g > \alpha_{warn} \;\&\; r/b > \beta_{warn} \;\&\; g/b > \gamma_{warn} \tag{3.1}$$

Analogicznie dla znaków czerwonych (np. stop, zakaz wjazdu), wymagamy:

$$r/g > \alpha_{red} \;\&\; r/b > \beta_{red} \;\&\; g/b > \gamma_{red} \tag{3.2}$$

gdzie α, β i γ to stałe wyznaczone eksperymentalnie. Są one określane poprzez próbkowanie wartości czerwonego, zielonego i niebieskiego na obrazach typowych znaków. Dokładne wartości stałych nie są istotne dla wykrywania znaków. Dzięki takiemu podejściu jest stosunkowo łatwo zwiększać zakres znaków rozpoznawanych przez algorytm.

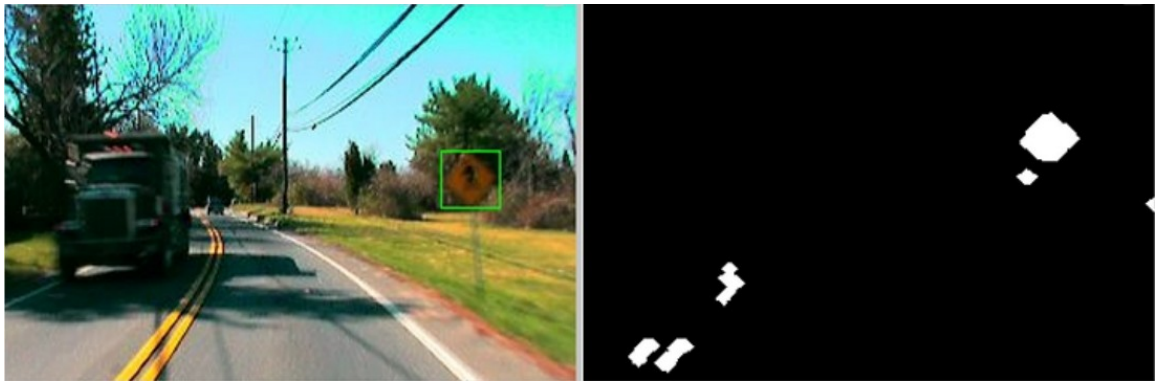
Powyższe ograniczenia są stosowane piksel po pikselu dla całego obrazu. Wynikiem tej operacji jest obraz binarny, w którym piksele będące kandydatami na przynależność do znaku, są oznaczane przez 1. Różne klasy znaków mogą być połączone w jeden wynikowy obraz binarny, bądź też każda klasa może

mieć swój własny obraz binarny. W tym opracowaniu ([31]) wszystkie obrazy binarne zostały połączone w jeden.

Po uzyskaniu obrazu binarnego, zostaje przeprowadzona erozja, w celu pozbycia się pojedynczych pikseli oraz dwie-trzy dylatacje - w celu połączenia części znaków, które mogły zostać rozdzielone przez obecność napisów czy piktogramów na znaku. Uzyskany w ten sposób obrys znaku jest wykorzystywany jako maska na etapie rozpoznania.

Kolejnym krokiem jest znalezienie obiektów na obrazie binarnym oraz zidentyfikowanie tych, które prawdopodobnie są znakami. Dla każdego obiektu są obliczane: środek masy, powierzchnia oraz najmniejszy prostokąt, w którym mieści się dany obiekt *(ang. bounding box)*. Dane te są wykorzystywane w zestawie reguł które klasyfikują każdy obiekt. Reguły wymagają aby obszar znaku był większy niż ustalone minimum i mniejszy od maksimum, stosunek wysokości do szerokości również musi się mieścić w zadanym przedziale. środek masy również musi znaleźć się w ograniczonym obszarze, w którym można oczekiwać pojawienia się znaków. Stosunek powierzchni i prostokąta otaczającego pozwala odrzucić zbyt chude obiekty. Obiekty spełniające wszystkie powyższe kryteria stają się kandydatami na znaki i są śledzone z klatki na klatkę. Jeśli obiekt wystąpi na pięciu kolejnych klatkach, jest wysyłany do części algorytmu odpowiedzialnej za rozpoznanie.

Rozpoznawanie znaków jest realizowane poprzez dopasowanie wzorca. Krok poprzedzający rozpoznanie usuwa z wyciętych obiektów tło, które mogłoby przeszkadzać w rozpoznaniu. Aby tego dokonać, wykorzystywana maska uzyskana w części odpowiedzialnej za detekcję. Takie podejście daje dobre oddzielenie znaku od otoczenia. Obrazy po takiej obróbce są skalowane do stałego rozmiaru (48x48 pikseli) i są porównywane z zapisanymi znakami o tym samym rozmiarze. Zapisane wzorce pochodzą z sekwencji wideo, więc są podobne do rozpoznawanych znaków. Ponieważ znaki mogą się nieznacznie różnić między sobą, może być potrzebne zapisanie kilku wzorców w bazie dla każdego znaku. Wynik działania algorytmu przedstawia rysunek 3.1.



Rysunek 3.1: Oryginalny obraz ze zlokalizowanym znakiem i obraz binarny

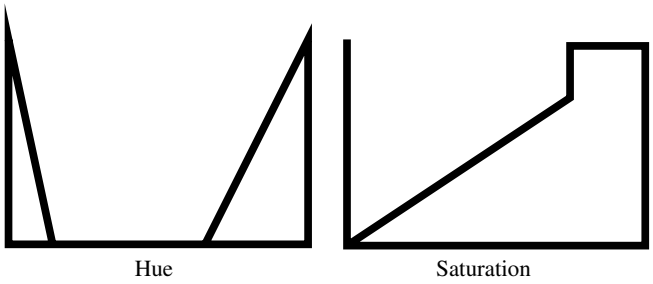
Przeprowadzone testy systemu na 92 znakach (23 637 klatek filmu) pokazują, że system wykrył 88% znaków, rozpoznał 78%. Błędnie wykrył 58%, natomiast błędnych rozpoznań było 6,5%. Wyniki pokazują dobrą skuteczność zarówno detekcji jak i rozpoznania znaków. Rezultaty nie są jednak wystarczająco dobre, by móc zastosować system w praktyce.

3.2. Traffic Sign Detection For Driver Support Systems

Inne podejście, oparte na przestrzeni barw HSI zostało opisane przez Escalera'e, Armingol'a oraz Salichs'a [9].

Tutaj również klasyfikacja barwna jest wyjściowym etapem detekcji. Do tej segmentacji została wybrana przestrzeń barw HSI, ponieważ zawiera różne informacje w każdym ze swoich komponentów. W celu wybrania obszarów zostały przygotowane tablice LUT. W ten sposób błędy klasyfikacji pikseli spowodowane progowaniem powinny zostać zrekompensowane. Przygotowano dwie takie tablice,

dla komponentu H oraz S. Po ich zastosowaniu obrazy są mnożone i normalizowane do maksymalnej



Rysunek 3.2: Tablice LUT

wartości 255. Warto podkreślić, że poprawna klasyfikacja nie będzie konieczna ponieważ tylko kształty zamknięte będą brane pod uwagę. Następnie zostają wyznaczone obiekty i jeśli są one odpowiednio duże, poszukuje się ich krawędzi i ostatecznie szuka znaku.

Po odnalezieniu krawędzi, algorytm musi odnaleźć znaki obecne na obrazie. Do tego celu zostały wykorzystane algorytmy genetyczne. Muszą one zbilansować dwa przeciwstawne zadania: przeszukanie całej przestrzeni oraz przeszukanie konkretnych stref. Najpierw szukamy stref, które mogą posiadać znaki i następnie w ich obszarze szukamy znaków. Ryzykiem tej metody jest zatrzymanie się w lokalnym ekstremum. Chociaż algorytmy genetyczne znajdują równowagę pomiędzy oboma zadaniami, są lepsze w znajdowaniu znaków niż szukaniu obszarów. Dlatego też należy unikać niebezpieczeństwa przedwczesnej zbieżności. Jest to odpowiednik utraty genetycznego bogactwa w gatunku. Krokami algorytmu genetycznego (ang. GA) są:

1. Zapoczątkowanie populacji.
2. Ocena dopasowania każdego osobnika.
3. Wybór dobrych rozwiązań dla kolejnego pokolenia.
4. Generacja nowego pokolenia.
5. Ocena nowych wyników.
6. Zastąpienie starej populacji nową.

Kroki 3 do 6 są wykonywane stałą liczbę razy (pokolenia) lub do czasu uzyskania konkretnej wartości rozwiązania.

Zapis kodu genetycznego rozpoczynamy od modelu znaku w zadanej odległości i prostopadłego do osi optycznych. Rozważane modyfikacje to zmiana pozycji i skal, spowodowane większym lub mniejszym dystansem do znaku, lub tym, że nie jest on prostopadły do osi. Model można zapisać jako:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix} \tag{3.3}$$

Parametry transformaty to:

$$a_{00} = E_x cos\Theta \tag{3.4}$$

$$a_{01} = E_x sin\Theta \tag{3.5}$$

$$a_{02} = T_x \tag{3.6}$$

$$a_{10} = -E_y sin\Theta \tag{3.7}$$

$$a_{11} = E_y cos\Theta \tag{3.8}$$

$$a_{02} = T_y \tag{3.9}$$

gdzie:

T_x – przemieszczenie poziome

T_y – przemieszczenie pionowe

E_x – skala pozioma

E_y – skala pionowa

Θ – rotacja pozioma

Są dwa powody dla których kodujemy powyższe parametry, zamiast sześciu składowych transformaty: jest ich mnie o jeden parametr i ponadto rozwiązania bez fizycznego znaczenia mogą być łatwo odrzucone.

W klasycznym algorytmie genetycznym, populacja początkowa jest generowana losowo, lecz w tym przypadku możemy "pomóc" niektórym wartościom znaleźć się bliżej ostatecznego wyniku. W tym celu binaryzujemy odpowiednim progiem obraz kolorowy, w celu uzyskania liczby i pozycji obiektów. Stała liczba osobników jest przyporządkowywana do każdego obiektu. W ten sposób obecność wystarczającej liczby osobników może być zagwarantowana pomimo obecności większych obiektów.

Reguła dopasowania została oparta na odległości Hausdorffa. Odległość ta wskazuje jak dwa kształty różnią się od siebie. Jeśli mamy dwa zbiory punktów $A = \{a_1, ..., a_m\}$ i $B = \{b_1, ...b_n\}$ to odległość Hausdorffa zdefiniowana jest następująco:

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{3.10}$$

$$h(A, B) = \max_{a \in A}(\min_{b \in B}(a - b)) \tag{3.11}$$

Funkcja $h(A, B)$ jest to bezpośrednia odległość Hausdorffa. Definiuje ona punkt należący do zbioru A, który jest najdalej, względem wybranej normy, od każdego z punktów zbioru B. Wariacja A jest częściową odległością Hausdorffa, w której tylko k odległości jest brane pod uwagę:

$$h_k(A, B) = K^{th}_{a \in A} \min_{b \in B}(a - b) \tag{3.12}$$

Miara h_k jest nieczuła na zakłócenia. W opisanym przypadku oba zbiory zawierają krawędzie obiektów wykrytych w poprzednim kroku oraz transformaty zakodowane w chromosomie każdego osobnika. Następnie obliczana jest transformata odległości oraz liczba punktów, których wartość jest mniejsza od progu. Dopasowanie jest relacją pomiędzy liczbą punktów a całkowitą liczbą punktów modelu. Kolejną zaletą tej funkcji dopasowania jest możliwość zatrzymania tworzenia pokoleń, jeśli wartość jest wystarczająco wysoka. Pokolenie początkowe wyznaczone przy pomocy algorytmu przedstawia rysunek 3.3.

Rysunek 3.3: Oryginalny obraz oraz pokolenie początkowe

W każdym pokoleniu wybierani są rodzice, na podstawie których będzie tworzone kolejne pokolenie. Proces selekcji jest określony poprzez wartość dopasowania rozwiązania. W tym procesie geny zawierające dobre rozwiązania są rozszerzane przez populację. Selekcja odbywa się zwykle przy pomocy metody ruletki: prawdopodobieństwo wylosowania jest proporcjonalne do znormalizowanego dopasowania

wszystkich osobników. Dzięki takiemu podejściu najlepsze osobniki mają więcej potomstwa. Czasami metoda ruletki ma problemy z trójkątnymi znakami. Jak wspomniano wcześniej, algorytmy genetyczne są lepsze w wyznaczaniu niż poszukiwaniu, ponieważ mogą zostać uwięzione w lokalnym ekstremum, z powodu tego, że większość osobników jest skoncentrowana w tej strefie. Trójkątne znaki często doprowadzają do tego zjawiska. Powodem jest to, że kilka osobników może nakładać dwa boki trójkąta na jeden bok modelu. Wygodnie jest opóźnić zbieżność, pomimo ryzyka jej spowolnienia aby zapewnić znalezienie globalnego ekstremum. Aby to uzyskać zastosowano metodę rankingową.

Etap krzyżowania szuka dobrej kombinacji genów, a nie osobników. Dla każdej pary osobników wybierane jest losowo kilka punktów krzyżowań. Potem dokonywane są drobne zmiany potomków (mutacja). Jeśli powstały złe osobniki, zostaną one wyeliminowane w następnym doborze. Jeśli okażą się dobre, ich informacja będzie propagowana w populacji. Wreszcie, najlepsze osobniki poprzedniego pokolenia będą zachowane (elitarność). Zadowalające wyniki otrzymano dla następujących parametrów:

- Poziome przemieszczenie: zakres 0/384 pikseli.
- Pionowe przemieszczenie: zakres 0/286 pikseli.
- Pozioma skala: zakres 0.25/1.3 (30/157 pikseli).
- Pionowa skala: zakres 0.25/1.3 (30/157 pikseli).
- Pozioma rotacja: zakres -15/15 stopni.
- Populacja: 101 osobników.
- Prawdopodobieństwo krzyżowania: 60%.
- Prawdopodobieństwo mutacji: 3%.
- Maksymalna liczba pokoleń: 51.
- Wartość graniczna (ang. escape value): 80%.

Zaprezentowany algorytm może być zastosowany również do innych obiektów, nie tylko znaków. Algorytm razi sobie ze znanymi trudnościami istniejącymi w rozpoznawaniu znaków w środowiskach otwartych. System jest nieczuły na zmiany oświetlenia, czy deformacje znaków. Algorytm został zaimplementowany z pomocą poleceń MMX. Dzięki temu mógł działać w czasie rzeczywistym. Średni czas dla jednej generacji to 17ms dla komputera z procesorem Pentium III 359MHz.

3.3. Detection, Categorization and Recognition of Road Signs for Autonomous Navigation

Na uniwersytecie w Louisville został opracowany system oparty o klasyfikator Bayesa oraz transformatę SIFT *(ang. Scale Invariant Feature Transform)* [12, 14]. Głównym celem projektu jest stworzenie inteligentnego, autonomicznego pojazdu, który potrafiłby poruszać się w środowisku. Mógłby dzięki czujnikom, zbierać dane o tym środowisku, w celu jego zrozumienia i przeprowadzenia konkretnych zadań. Jednym z celów systemu jest zapewnienie Systemu Wspierania Kierowcy *(ang. Driver Suport System - DDS)*. Detekcja i rozpoznawanie znaków drogowych *(ang. RSR)* jest bardzo ważnym zadaniem, ponieważ znaki zawierają wiele informacji koniecznych do udanego, bezpiecznego i łatwego nawigowania. Opisywane podejście RSR wykorzystuje klasyfikator Bayesa do wykrywania znaków na obrazach, oparte o ich barwną zawartość. Kategoria barwna znaku, jest bardzo ważna w procesie jego rozpoznawania. Na przykład, dwa identyczne znaki w innych kolorach mogą mieć zupełnie inne znaczenie. Klasyfikator Bayesa nie tylko etykietuje obraz, ale również dzieli etykiety na kategorie znaków. W oparciu o wyniki uzyskane przez klasyfikator Bayesa, zostaje użyta transformata SIFT, w celu dopasowania etykiet z odpowiadającymi im znakami. Takie podejście posiada kilka zalet odróżniających go od

poprzednich prac o RSR. Jedną z nich jest przezwyciężenie części trudności poprzednich algorytmów, jak powolność dopasowywania wzorców, potrzeby dużej ilości różnych rzeczywistych obrazów znaków do nauki (podejścia oparte o sieci neuronowe), czy potrzeba znajomości fizycznej charakterystyki oświetlenia znaków. Kolejną zaletą wykorzystania klasyfikatora Bayesa jest akceleracja ekstrakcji cech i operacji dopasowania transformaty SIFT, poprzez zmniejszanie obszaru dopasowania tylko to etykiet. Również ograniczenie przeszukiwanej podprzestrzeni, poprzez barwne kategorie przyspiesza działanie algorytmu.

Zostanie teraz przedstawiony klasyfikator Bayesa i jego wykorzystanie do wykrywania znaków. Zakładamy, że obraz będzie etykietowany przy pomocy pewniej liczby klas C. Dla każdego piksela x obrazu, prawdopodobieństwo warunkowe $P(w_i/x)$, $i \in [1, c]$, jest obliczane dla w_i będącego i'tą klasą. Oznacza ono stopień przynależności danego piksela do tej klasy. Klasyfikator Bayesa wybiera klasę, w oparciu o maksymalne prawdopodobieństwo warunkowe:

$$x \in w_i \text{ if } P(w_i/x) > p(w_j/x), \forall j \neq i \tag{3.13}$$

Prawdopodobieństwo wystąpienia każdej z klas $P(w_i)$ może być użyte do obliczenia prawdopodobieństwa warunkowego w następujący sposób:

$$P(w_i/x) = \frac{P(x/w_i)P(w_i)}{P(x)} \tag{3.14}$$

Prawdopodobieństwo każdej z klas może być w łatwy sposób oszacowane/założone. W naszym problemie RSR, rozważamy pięć klas barwnych (czerwoną, żółtą, zieloną, niebieską i białą). Zakładamy równość prawdopodobieństwa wystąpienia każdej z grup. Klasa tła jest wykrywana, jeśli maksymalne obliczone prawdopodobieństwo jest mniejsze od konkretnej wartości. Można użyć parametrycznej lub nieparametrycznej estymacji w celu oszacowania warunkowych gęstości klas $P(w_i/x)$. Rozważono tutaj parametryczne oszacowanie. W tym podejściu, do oszacowania gęstości są użyte dane projektowe *(ang. design data)*. Zakładamy, że $P(w_i/x)$ posiada rozkład Gaussa, ponieważ jedynymi potrzebnymi parametrami, są macierz kowariancji Σ oraz wektor własny μ. W literaturze jest wiele podejść do oszacowania tych wartości. Jednym z najbardziej powszechnych jest estymator maksymalnego prawdopodobieństwa *(ang. maximum likehood estimator - MLE)*, który został wykorzystany w tym podejściu. W MLE, oszacowany wektor własny ($\hat{\mu}$) jest uzyskiwany w następujący sposób:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \tag{3.15}$$

natomiast estymata macierzy kowariancji, jest obliczana przy pomocy wzoru:

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^t \tag{3.16}$$

Gdzie n jest liczbą pikseli w zbiorze projektowym *(ang. design set)*, użytym do oszacowania. Rozważamy problem jako jednowymiarowy rozkład Gaussa. Jednowymiarowe dane wejściowe przedstawiają element *Hue* z modelu barwnego HSI. Powodem takiej reprezentacji, jest to, że ten element jest niewrażliwy na zmianę oświetlenia i cienie, co jest korzystną cechą w problemie RSR. Ponieważ x_k jest składową *Hue* dla k'tego piksela. Jako wejście można również wykorzystać składowe RGB, ale w tym przypadku błąd klasyfikacji będzie większy. Lokalne deskryptory cech niezmiennych są wektorami opisu, które zawierają klucze opisujące region obrazu w sposób nieczuły na transformacje przestrzenne i inne czynniki zaburzające. Aby być wydajnym w dopasowywaniu cech, deskryptory powinny być rozróżnialne i jednocześnie nieczułe na zmiany warunków oświetlenia, czy błędy w wykryciu punktów. Zmiany, występujące w obrazach drogowych dotyczą oświetlenia, skali, rotacji i innych zmian. Wiele ostatnich prac skupiało się na tym, jak zapewnić, by deskryptory były nie czułe na zmianę skali, czy rotację, jak również na zmianę poziomu szarości, iluminacje, czy jasność. Najpowszechniejsze podejście oparte na deskryptorach polega na zbudowaniu niezmiennych "obszarów obrazu", które są wykorzystywane jako regiony

wspierające do obliczania deskryptorów. Tego tematu dotyczy wiele prac. Na przykład Mikołajczyk i Schmid stworzyli afiniczne, niezmiennicze punkty zainteresowania ze skojarzonymi afinicznymi i inwariantnymi regionami[23]. Zrobili oni również porównanie wydajnościowe różnych deskryptorów [24], z którego wynika, że deskryptory SIFT są najlepsze przy zmianach skali, rotacji czy oświetlenia.

Transformata SIFT została opracowana przez Davida Lowe [14]. Jest ona głównie wykorzystywana do tworzenia niezmiennych cech na obrazie, nadających się do detekcji, rozpoznania, czy śledzenia. Jej potencjał leży w niewrażliwości wygenerowanych cech na translacje, skalowanie, rotację i częściowo na zmiany oświetlenia obrazu. Z tego względu została ona zaadaptowana do problemu RSR.

Pierwszym krokiem transformaty jest określenie kluczowych pozycji lub punktów zainteresowania. Jest to realizowane poprzez stworzenie piramidy Gaussa, przy użyciu gaussowskiego wygładzania i subsampling. Każdy poziom piramidy jest budowany z użyciem obrazu różnicowego Gaussa *(ang. diffrence-of-Gaussian image)*, uzyskanego poprzez odjęcie obrazu od jego wersji wygładzonej. Następnie obniżana jest rozdzielczość obrazu, w celu zbudowania kolejnego poziomu piramidy. Lokalizacje cech są identyfikowane jako ekstrema, w zależności od otaczających pikseli i sąsiednich wag. W ten sposób SIFT gwarantuje, że w obszarach są lokalizowane kluczowe punkty oraz wagi o wysokich wahaniach, które zapewniają stabilność tych lokacji, dla scharakteryzowania obrazu. Później kluczowe punkty stają się danymi wejściowymi dla metody indeksowania (nearest-neighbor) identyfikującej kandydatów. Ostateczna weryfikacja każdego z kandydatów odbywa się poprzez znalezienie rozwiązania dla nieznanego modelu parametrów. Lowe nazwał tą metodę generowania cech Ścale Invariant Feature Transform"(SIFT).

Znaki drogowe są projektowane przy użyciu wybranych kształtów i kontrastowych kolorów, w taki sposób, aby łatwo można je było odróżnić od otoczenia. Opierając się na tym fakcie, proponowane podejście wykorzystuje informacje o zarówno kolorze, jak i kształcie. Jak wspomniano wcześniej, klasyfikator Bayesa jest wykorzystywany do etykietowania kolorów obrazu, na kategorie odpowiadające kolorom znaków. Deskryptory SIFT dla standardowych znaków są budowane i zapisywane przed uruchomieniem aplikacji. Zgodnie z klasą przypisaną przez klasyfikator Bayesa, zostaje uruchomiony odpowiadający jej proces pomiędzy deskryptorem SIFT przypisanym do etykiety i tymi zapisanymi dla standardowych znaków. Szczegółowy opis procesu dopasowywania w SIFT jest wyjaśniony prze Lowego w [14]. Liczba niezmiennych cech punktów zainteresowania jest różna dla każdego znaku. Ponieważ dopasowywanie oparte o absolutną liczbę odpowiadających cech może prowadzić do błędnych rezultatów, proces dopasowywania odbywa się przy użyciu następującej funkcji celu:

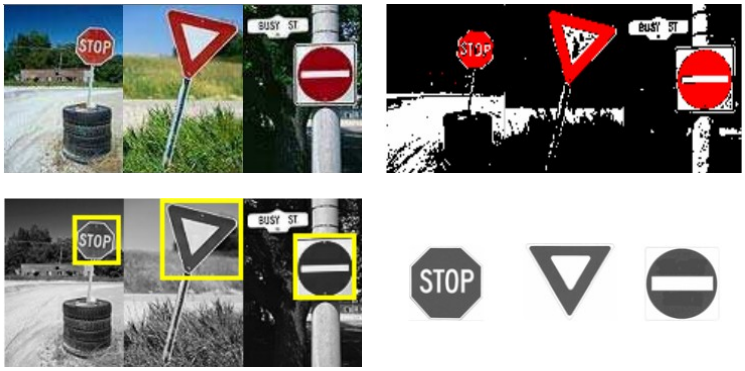
$$f_i = \frac{n}{\sqrt{n_b n_i}} \tag{3.17}$$

gdzie f_i jest to wartość funkcji celu dla i-tego standardowego znaku, n to liczba dopasowań, n_i - całkowita liczba niezmiennych cech w i-tym standardowym znaku, a n_b to całkowita liczba niezmiennych cech w badanym obiekcie. Maksymalna wartość f to 1.0, w wypadku pełnego dopasowania, a minimalna to 0, kiedy brak jest jakiegokolwiek dopasowania. Decyzja o dopasowaniu jest podejmowana dla znaku standardowego o największej wartości f. Jeśli największa wartość funkcji celu jest mniejsza od pewnej zadanej wartości, to decyzja jest odrzucana. Algorytm można przedstawić następująco:

1. Zbudować deskryptory SIFT dla znanych znaków (offline).
2. Pobrać obraz.
3. Wykonać etykietowanie obrazu za pomocą klasyfikatora Bayesa, według barwnej zawartości.
4. Zbudować deskryptory SIFT dla każdej z etykiet.
5. Dla każdej etykiety dopasować jej deskryptory, do deskryptorów znaków standardowych z danej kategorii barwnej.
6. Dla każdej etykiety znaleźć znak standardowy z największą wartością funkcji celu ($f_{i\ max}$).
7. Dla każdej etykiety, jeżeli $f_{i\ max}$ jest mniejsze od pewnej zadanej liczby, to dla tej etykiety nie rozpoznano żadnego znaku.

8. W innym przypadku znak z $f_{i\ max}$ zostaje rozpoznany na obrazie dla tej etykiety.

Wyniki działania algorytmu przedstawia rysunek 3.4.



Rysunek 3.4: Wynik działania algorytmu. Obraz oryginalny, działanie klasyfikatora Bayesa, lokalizacja znaków, wyniki rozpoznania.

3.4. Real-Time Detection of the Triangular and Rectangular Shape Road Signs

Doktor Bogusław Cyganek przedstawił algorytm detekcji i rozpoznawania znaków trójkątnych oraz kwadratowych[8]. Cały system jest oparty o trójfazowe podejście:

1. Pobranie obrazu i filtrowanie
2. Detekcja kształtu
3. Klasyfikacja

Cały proces rozpoczyna się od pobrania obrazu za pomocą kamery Marlin F-033C, która jest również zaprogramowana do przeprowadzenia filtracji dolnoprzepustowej. Następnie jest przeprowadzana segmentacja barwna, w celu uzyskania obrazu binarnego wyznaczającego obszary występowania potencjalnych znaków. Segmentacja odbywa się w przestrzeni barw HSI prostego sprawdzania wartości składowych. Przedziały składowych występujące w znakach drogowych są następujące (wartości podane dla znormalizowanego [0-255] HSI):

- Niebieski
  - Hue: [120-165]
  - Saturation: [80-175]
- Żółty
  - Hue: [15-43]
  - Saturation: [95-255]

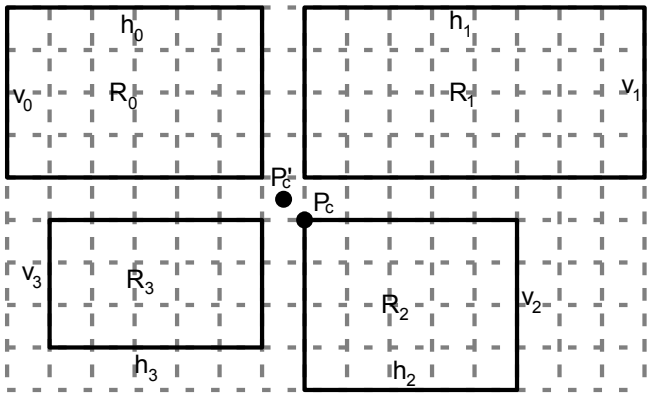
Detektor kształtów jest podzielony na 4 moduły. Cały proces zaczyna się od wykrycia punktów będących wierzchołkami figur, które następnie muszą zostać pogrupowane. W kolejnym kroku dochodzi do detekcji figur i ich weryfikacji. Powyższy detektor jest w stanie rozpoznać trójkąty i kwadraty w różnych pozycjach, skalach, czy rotacjach. Weryfikacja figur musi zapewnić,ze tylko kształty spełniające specyfikację znaków drogowych są przekazywane do klasyfikatorów.

Kolejne kroki przetwarzania obrazu, w celu wydobycia ważnych cech znaków i ich klasyfikacji, rozpoczynają się od ekstrakcji obszarów z wykrytymi figurami. Obszary te są pobierane z obrazu monochromatycznego, ponieważ klasyfikatory nie biorą pod uwagę informacji barwnej. Obraz monochromatyczny jest po prostu pobierany z kanału czerwonego, zamiast zostać wyliczony jako średnia wszystkich trzech kanałów. Takie podejście ma pewne korzystne cechy dla ekstrakcji piktogramów. Celem następującej po konwersji na obraz monochromatyczny rejestracji kształtu, jest jego normalizacja. Normalizacji podlega rozmiar i orientacja, która dostosowuje piktogram do wymogów klasyfikatora. Jest to realizowane poprzez rozwiązania prostego równanie liniowego dla transformacji afinicznej. Zakłada się, że taka transformacja jest wystarczająca, ponieważ znaki drogowe są sztywnymi, płaskimi przedmiotami. Dobre funkcjonowanie takiego podejścia, zostało zweryfikowane eksperymentalnie w rzeczywistych warunkach. Wreszcie, potencjalny znak jest binaryzowany i próbkowany. Następnie binarny wektor cech jest przekazywany do modułu klasyfikującego, który w opisywanym systemie jest grupą współpracujących sieci neuronowych.

Przyjrzyjmy się dokładnie wykrywaniu punktów charakterystycznych znaków - rogom. Znajomość położenia trzech rogów znaku jest zwykle wystarczająca dla jednoznacznego zlokalizowania całego znaku. Jednakże, jest to czasem kłopotliwe, ze względu na przysłonięcia lub niedoskonałości fazy segmentacji. Technika ta może być wykorzystana do wykrywania wszystkich kształtów, które mogą być scharakteryzowane przez wierzchołki. Dla innych kształtów, na przykład okręgów, mogą zostać zastosowane inne techniki[7], które zostaną opisane w dalszej części rozdziału.

Wierzchołki są wykrywane na obrazach binarnych, uzyskanych z modułu segmentującego. Technika ta jest bardzo szybka. Aby sprawdzić czy dany punkt jest wierzchołkiem, należy przeanalizować jego otoczenie. Jest to realizowane poprzez detektor lokalnych cech binarnych (DLBF). W ogólnym przypadku, jest on zbudowany z czterech obszarów, w których centrum znajduje się punkt zainteresowania P'_c. Dla dyskretnej siatki pikseli, punkt centralny leży w wirtualnej pozycji, która nie pokrywa się z siatką. Z tego powodu DLBF jest zakotwiczone w rzeczywistym punkcie P_c, który leży na dyskretnej siatce.

Detektor cech binarnych działa na czterech obszarach R_0, R_1, R_2, R_3 - każdy z nich jest rozmiaru $h_i x v_i$. Wykrywanie punktów, przy pomocy DLBF, odbywa się poprzez liczenie ustawionych pikseli w każdym z obszarów. Z tego powodu, dla każdego punktu uzyskujemy zbiór czterech liczników $c_0, c_1, c_2 i c_3$. Liczniki te są porównywane ze zdefiniowanymi wzorcami dla wierzchołków. Jeśli natrafimy na zgodność, to punkt P_c jest klasyfikowany jako wierzchołek. Dla znaków drogowych DLBF jest



Rysunek 3.5: Lokalny detektor cech binarnych

uproszczony do symetrycznego lokalnego detektora cech binarnych (SDLBF), w którym wszystkie obszary są kwadratami tego samego rozmiaru. Każdy obszar R_i jest dodatkowo podzielony przekątną od punktu P'_c. W takim podejściu SDLBF składa się z ośmiu segmentów. Analiza ich liczników pozwala na klasyfikację punktu centralnego do jednej z grup punktów wierzchołkowych. Podając dozwolone wartości liczników, definiuje się typ punktu wierzchołkowego. Dla każdej możliwości wierzchołka został zdefiniowany odpowiedni zestaw liczników. Podczas eksperymentów okazało się, że dobre wyniki

można uzyskać uznając obszar za pusty, jeśli wartość współczynnika jego wypełnienia jest mniejsza, bądź równa 5% liczby pikseli w małym obszarze (co stanowi 36/45 dla obszarów 9x9). Za pełny, natomiast uznano obszar o wypełnieniu $\geq 95\%$. Inne parametry sterujące to rozmiar i liczba obszarów SDLBF, która oczywiście zależy od rozdzielczości obrazu.

W wyniku poprzedniego kroku otrzymujemy zbiór punktów wierzchołkowych. Punkty te mają tendencję do tworzenia lokalnych zgrupowań, na przykład zamiast pojedynczego punktu będącego wierzchołkiem znaku otrzymujemy zbiór punktów, w którym odległości pomiędzy nimi nie przekraczają kilku pikseli. Z tego też powodu kolejny krok składa się z wyszukania każdego z takich zbiorów i zastąpienia go pojedynczym punktem, umiejscowionym w środku ciężkości zbioru. Zbiór S_p wszystkich punktów wykrytych przy pomocy SDLBF jest opisany w następujący sposób:

$$S_p = \{P_0, P_1, ..., P_n\} = \{(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)\} \tag{3.18}$$

W S_p określane są na podstawie odległości, podzbiory K_i mające być zastąpione przez jeden punkt. Zbiór wszystkich podzbiorów $C(S_p)$ jest przedstawiony następująco:

$$C(S_p) = \{K_0, K_1, ..., K_m\} = \{\{..., x_{i_1}, ...\}, \{..., x_{i_2}, ...\}, ..., \{..., x_{i_m}, ...\}\} \tag{3.19}$$

Dla każdego podzbioru K_i wyznaczany jest środek ciężkości, który reprezentuje cały podzbiór. W wyniku tego procesu powstaje zbiór $M(C(S_p))$:

$$M(C(S_P)) = \{\bar{K_1}, \bar{K_2}, ..., \bar{K_m}\} = \{(\bar{x_0}, \bar{y_0}), (\bar{x_1}, \bar{y_1}), ..., (\bar{x_m}, \bar{y_m})\} \tag{3.20}$$

gdzie

$$\bar{x_p} = \frac{1}{\#K_p} \sum_{x_{pi} \in K_p} x_{p_i} \tag{3.21}$$

$$\bar{y_p} = \frac{1}{\#K_p} \sum_{y_{pi} \in K_p} y_{p_i} \tag{3.22}$$

Tworzenie podzbiorów jest zarządzane przez jeden parametr - maksymalną odległość d_τ pomiędzy dowolnymi dwoma punktami, powyżej której zaliczamy punkty do różnych podzbiorów. Oznacza to, że jeśli dla dwóch punktów P_i oraz P_j zachodzi:

$$d(P_i, P_j) \leq d_\tau \tag{3.23}$$

gdzie $d(.,.)$ oznacza metrykę (np. Euklidesową), wtedy te dwa punkty należą do jednego podzbioru. Dla zbioru S_p zawierającego n punktów, proces wyznaczania podzbiorów rozpoczyna się od zbudowania macierzy odległości D, która zawiera odległości dla każdej pary ze zbioru S_p. Istnieje $(n(n-1)/2$ takich par. Macierz D jest trójkątna i posiada zera na diagonali. Przykład takiej macierzy dla pięciu elementów wygląda następująco (zawiera $5 * 4/2 = 10$ różnych odległości): Testy podejścia przeprowadzono na

$$\begin{bmatrix} 0 & d_{01} & d_{02} & d_{03} & d_{04} \\ & 0 & d_{12} & d_{13} & d_{14} \\ & & 0 & d_{23} & d_{24} \\ & & & 0 & d_{34} \\ & & & & 0 \end{bmatrix} \tag{3.24}$$

Rysunek 3.6: Macierz dystansów D

komputerze PC, z procesorem Pentium IV 3.4GHz oraz 2GB RAM. Algorytm został zaimplementowany w języku C++ przy pomocy Microsoft Visual 6.0 IDE. Przeprowadzone próby wskazują bardzo wysoką dokładność detekcji, wynoszącą 97% dla wszystkich grup znaków. Pojawiają się pewne problemy w przypadku częściowego przysłonięcia znaku, szczególnie gdy przysłonięty jest któryś z wierzchołków. Wyniki działania algorytmu przedstawia rysunek 3.7.

Rysunek 3.7: Rezultaty działania algorytmu

3.5. Circular road signs recognition with soft classifiers

Inna publikacja dr Bogusława Cyganka traktuje o detekcji znaków kolistych[7]. System klasyfikuje kolory bezpośrednio w przestrzeni RGB. Do klasyfikacji została użyta Maszyna Wektorów Wspierających *(ang. Support Vector Machine)*. Jest ona uczona na danych z rzeczywistych obrazów. Klasyfikator SVM działa w trybie jednoklasowym.

3.5.1. SVM

Załóżmy, że dla zbioru danych o kolorach, chcemy uzyskać parametry najmniejszej hiperboli je zawierającej. Problem ten możemy sformułować następująco:

$$\Theta(a, r) = r^2 \tag{3.25}$$

Przy ograniczeniach:

$$\forall_i : ||x_i - a|| \leq r^2 \tag{3.26}$$

W celu uzyskania elastyczności, umożliwiamy na wystąpienie w zbiorze uczącym pewniej liczby danych poza hiperbolą. Z tego powodu powinniśmy zezwalać na większe odległości niż *r*, ale z pewnymi dodatkowymi karami. Aby to uzyskać to równania (3.25) dodajemy zapas:

$$\Theta(a, r) = r^2 + C \sum_i \xi_i \tag{3.27}$$

przy założeniu, że prawie wszystkie dane są wewnątrz hiperboli:

$$\forall_i : ||x_i - a|| \leq r^2 + \xi_i, \xi_i \geq 0 \tag{3.28}$$

gdzie ξ_i jest zapasem dla każdej danej wejściowej x_i oraz C jest parametrem kontrolującym proces optymalizacji. Dla większego C dopuszcza się mniej danych poza hiperbolą. Prowadzi to do jej powiększania.

Dany zbiór uczący $\{x_i\}$, rozwiązanie równania (3.27), które równocześnie spełnia (3.28), mogą być uzyskane dzięki mnożnikom Lagrange'a[25]:

$$Q_L(r, a, \alpha_i, \beta_i, \xi_i) = r^2 + C\sum_i \xi_i - \sum_i \alpha_i \left[r^2 + \xi_i - ||x_i - a||^2\right] - \sum_i \beta_i \xi_i \tag{3.29}$$

gdzie

$$\alpha_i \geq 0, \beta_i \geq 0 \tag{3.30}$$

Wielomian Q_L musi zostać zminimalizowany ze względu na r, a i ξ_i oraz maksymalizowana po α_i i β_i.

Rozwiązanie równania (3.29) może zostać znalezione przy pomocy tzw. dualnej postaci Wolfe'a Q_W, która jest uzyskiwana z Q_L po wstawieniu wyrażeń na warunki Lagrange'a [13].

Q_W przyjmuje postać:

$$Q_W = \sum_i \alpha_i K(x_i, x_i) - \sum_j \alpha_j \sum_i \alpha_i K(x_i, x_j) \tag{3.31}$$

Gdzie $K(x_i, x_j)$ oznacza funkcje jądra *(ang. kernel functions)* [3, 17]. Z równania (3.29), w świetle warunków optymalności Khuna-Tuckera, wynika:

$$\forall_i : \alpha_i \left[r^2 + \xi_i - ||x_i - a||^2\right] = 0 \tag{3.32}$$

Z powyższego równania widać, że punkty x_i, które są wewnątrz hipersfery muszą mieć związane ze sobą wartości α_i równe zero. Pozostałe punkty, dla których $\alpha_i > 0$ są albo na granicy, albo daleko od hipersfery. Punkty wewnątrz hipersfery, są nazywane wektorami podpierającymi *(ang. support vectors)*. Więcej szczegółów dotyczących powyższego klasyfikatora SVM można znaleźć w publikacji [7].

Wyniki eksperymentów pokazują, że taka metoda segmentacji pozwala na lepszy opis przestrzeni niż na przykład progowanie stałymi wartościami dla kanałów H i S w przestrzeni HSI. Dla detekcji znaków drogowych oznacza to mniej błędnie zlokalizowanych obszarów.

3.5.2. Adaptacyjne wykrywanie i weryfikacja kształtu

Po uzyskaniu binarnego obrazu w wyniku segmentacji, nadszedł czas na kolejny krok algorytmu. Składa się on z wykrycia kształtów kolistych na binarnym obrazie. Metoda adaptacyjna wykrywania kształtów została opracowana przez dr Cyganka. Jest to proces iteracyjny, zaczynający w pewnym punkcie z tagiem 's'. Zaczynając od tej pozycji, budowane jest pierwsze kwadratowe okno. Następnie jest ono sekwencyjnie rozszerzane we wszystkich czterech kierunkach (poziomo i pionowo), jak i wzdłuż przekątnych (od rogów). Dla każdego kroku liczba Δs pikseli 's', dodanych w tym kroku jest liczona.

Proces rozszerzania okna jest kontynuowany tylko w kierunkach, w których $\Delta s > 0$. Proces ten zostaje zakończony jeśli wszystkie $\Delta s_i, i = 1, 2, ..., 7$ są równe zero, bądź osiągnięto krawędź obrazu. Własności topologiczne tak znalezionego obrazu są sterowane poprzez rozmiar zbioru, o który powiększa się okno w każdym kroku. Jeśli zbiór powiększa się tylko o jeden piksel, oznacza to wykrycie sąsiadująco-połączonych figur.

Kiedy kształt zostaje wykryty, zostaje on wycięty z obrazu poprzez zmianę wszystkich etykiet 's' na etykiety 'b'. Następnie algorytm kontynuuje przeszukiwanie obrazu dopóki nie przebada wszystkich pikseli obrazu. W ten sposób odnalezione obszary są sprawdzane, czy odpowiadają definicji kolistego znaku typu "Błub Ć". Na początku odrzuca się najmniejsze obszary, ponieważ zawierają zbyt mało informacji do ich klasyfikacji. W tym systemie wszystkie obszary, których pole jest mniejsze niż 10% średniego rozmiaru (N + M)/2, są odrzucane. Przez N i M rozumiemy wysokość i szerokość badanego obrazu. Następnie analizowany jest kształt. Dla okręgu są sprawdzane wszystkie cztery kwadraty zakotwiczone w rogach obszaru i opisane na tym okręgu.

Klasyfikatory mogą tolerować tylko niewielkie odchylenia obrazu, spowodowane niedokładnością detektorów. Niestety nie tolerują różnic spowodowanych zniekształceniem perspektywy. Z tego powodu konieczna jest transformacja wykrytych kształtów do ustalonych wymiarów, takich jak w bazie znaków.

Proces rejestracji znaków opiera się na prostym zbiorze równań wynikających z przekształceń afinicznych. Po transformacji współrzędnych, nowe wartości pikseli muszą być interpolowane. Następnie obszary są przekształcane do przestrzeni log-polar. Transformacja log-polar punktu o współrzędnych (α, r) na współrzędne (x, y) można opisać następująco:

$$x = B^r \cos\alpha + x_0, y = B^r \sin\alpha + y_0 \tag{3.33}$$

gdzie $O = (x_o, y_0)$ jest środkiem transformacji, B oznacza stałą, której wartość musi być większa od 1. Stałą tą zazwyczaj wyliczamy ze wzoru:

$$B = \sqrt[r_{max}]{d_{max}}, d_{max} > 1, r_{max} > 1 \tag{3.34}$$

$$d^2_{max} = (x_{max} - x_0)^2 + (y_{max} - y_0)^2 \tag{3.35}$$

Zmienna d^2_{max} jest maksymalną odległością punktu danego obrazu, od jego środka O. Dla obrazów cyfrowych, wartości r i α powinny być odpowiednio zkwantyfikowane. W tej przestrzeni dobór środka O jest bardzo istotny z powodu niejednorodnego próbkowania obrazu. Im bliżej środka, tym punkty są pobierane z większą gęstością. Ma to kolosalny wpływ na rozpoznanie znaku, w którego centrum znajduje się piktogram. W doborze środka mamy dwa wyjścia: wybrać środek obrazu, bądź środek jego masy. Wybór środka obrazu powoduje przyspieszenie obliczeń, gdyż wtedy nie musimy dla każdego obszaru wyliczać środka masy. Zauważono jednak, że wybór środka masy powoduje wzrost rozpoznawania znaków o 2-5%.

Obrazy referencyjne, znajdujące się w bazie danych, jak również obrazy wejściowe muszą być zbinaryzowane. Obrazy są binaryzowane do dwóch grup, dla koloru niebieskiego i czerwonego. Została opracowana specjalna metoda binaryzacji, oparta na empirycznych obserwacjach histogramów prawdziwych znaków. Dzięki heurystycznej metodzie próg binaryzacji jest ustawiany jako średnia pomiędzy pewną stałą (w projekcie przyjęto 128), a pozycją maksimum na histogramie. Metoda ta opiera się na fakcie, że maksimum histogramu odpowiada tłu znaku.

Ostatnim krokiem jest wydzielenie cech. Przetestowana kilka różnych metod. Dla większości najskuteczniejsze okazało się próbkowanie spiralne. Wektor cech jest budowany od środka piktogramu, w kierunku jego obwódki. Proces ten przedstawia rysunek 3.8.

Rysunek 3.8: Rozszerzanie okna

Wszystkie grupy znaków składają się z ponad dwustu różnych znaków. Jest to niemały problem przy budowanie odpowiedniego klasyfikatora. Często nawet te same znaki mogą się miedzy sobą nieznacznie różnić piktogramem, chociażby przez niedoskonałości procesu detekcji. Dodatkowym problem występującym przy znakach okrągłych, jest niemożliwość ustalenia ich rotacji. Z tych względów wybrano rozpoznawanie z grupy deformowalnych modeli. Polega ono na deformowaniu wzorców i używaniu ich wszystkich w procesie klasyfikacji. W podejściu klasyfikacja w stałej liczbie deformacji jest przeprowadzana przez sieć neuronową Hamminga. Sieci te są organizowane w grupę współpracujących klasyfikatorów ekspertów, nazwaną maszyną komisyjną. Każdy z klasyfikatorów głosuje na swój wzór. Następnie najsilniejsze głosy są wzmacnianie przez mechanizm wsparcia jednomyślnych ekspertów. Ostatecznie pojedynczy głos zostaje wybrany przez arbitra i zwrócony przez maszynę komisyjną. Szczegóły tego rozwiązania znajdują się w publikacji[7].

4. Projekt systemu

4.1. Wymagania

Głównym założeniem projektu jest wykorzystanie możliwie najprostszych metod analizy obrazu do osiągnięcia celu. Oczywiście wybrane metody muszą być w stanie wykrywać i rozpoznawać znaki. Cały system musi działać w czasie rzeczywistym. Jest to ważne założenie, ponieważ ogranicza w znacznym stopniu metody i algorytmy, które można zastosować.

Ważną cechą sytemu powinna być również możliwie jak największa niewrażliwość na zakłócenia. Chodzi tutaj o wszelkie rotacje znaków, ich przekrzywienia czy przysłonięcia. System powinien radzić sobie z tymi trudnościami w celu zapewnienia jak najlepszej jakości rozpoznania.

Ostatnim założeniem jest wykorzystanie do realizacji algorytmu współczesnego komputera PC. Chodzi tutaj o użycie powszechnie dostępnych i relatywnie tanich maszyn. Mają one być wystarczające dla działania programu.

4.2. Projekt rozwiązania

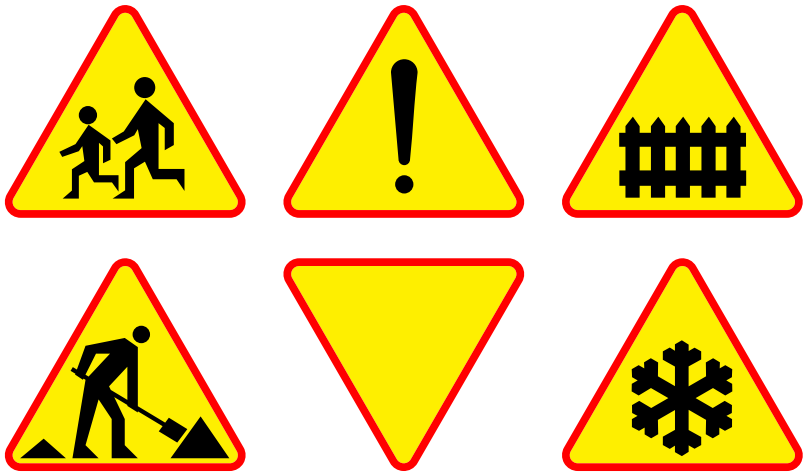
4.2.1. Analiza problemu

Znaki drogowe charakteryzują się dwoma cechami: posiadają typowe dla siebie kolory oraz maja geometryczne kształty. Cechy te należałoby wykorzystać jako najbardziej oczywiste.

Znaki drogowe pionowe dzielą się na:

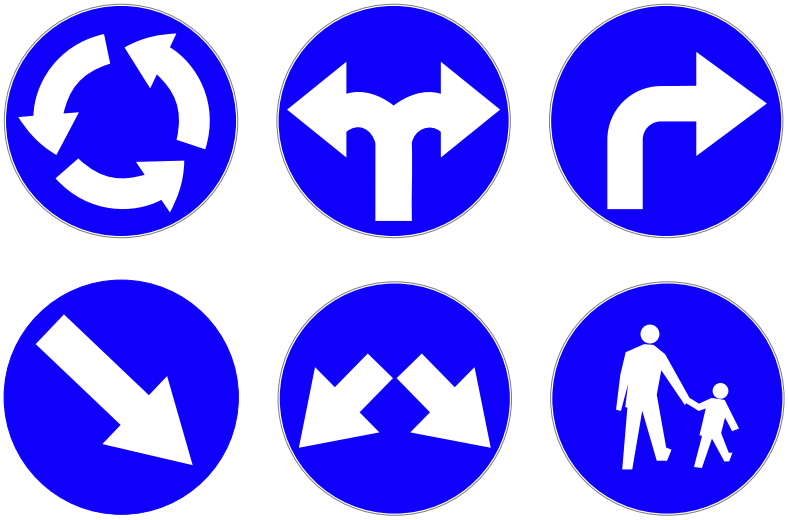
- znaki ostrzegawcze (typ A) - trójkątne w czerwonym obramowaniu, czarny (w niektórych znakach – kolorowy) symbol na żółtym tle
- znaki zakazu (typ B) - okrągłe w czerwonym obramowaniu, czarny (w niektórych znakach – kolorowy) symbol na białym tle
- znaki nakazu (typ C) - okrągłe, niebieskie, z białym symbolem na niebieskim tle
- znaki informacyjne (typ D) - prostokątne, niebieskie (ew. z białym polem na tle niebieskim), z białym lub czarnym symbolem na niebieskim lub białym tle
- znaki kierunku i miejscowości (typ E) - zwane popularnie drogowskazami o różnym kształcie i kolorze zależnym od sytuacji
- znaki uzupełniające (typ F) - duże prostokąty lub kwadraty kolorów: niebieski i żółty, znaki uprzedzające przed ruchem drogowym i znakami
- tabliczki do znaków drogowych (typ T) - małe prostokątne, białe lub żółte tabliczki z czarnym napisem lub symbolem, umieszczone pod znakiem

W pracy zajmiemy się detekcją i rozpoznaniem pierwszych czterech grup znaków. Rysunki 4.1, 4.2, 4.3 oraz 4.4 zawierają przedstawicieli każdej z grup znaków.

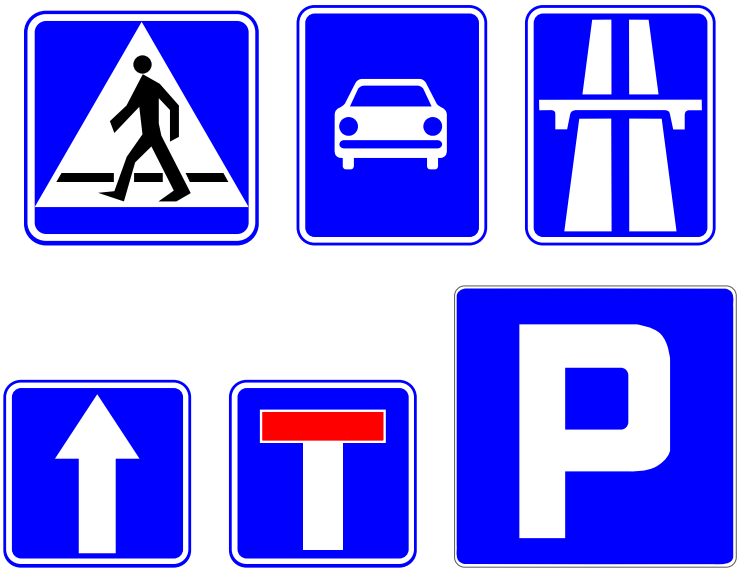


Rysunek 4.1: Znaki typu A (ostrzegawcze)

Rysunek 4.2: Znaki typu B (zakazu)



Rysunek 4.3: Znaki typu C (nakazu)



Rysunek 4.4: Znaki typu D (informacyjne)

Jak widać szukając żółtych obszarów wystarczy skupić się w następnym kroku na poszukiwaniu trójkątów. Przeglądając obszary czerwone – okręgów. Natomiast przeszukując obszary niebieskie trzeba szukać zarówno prostokątów, jak i okręgów.

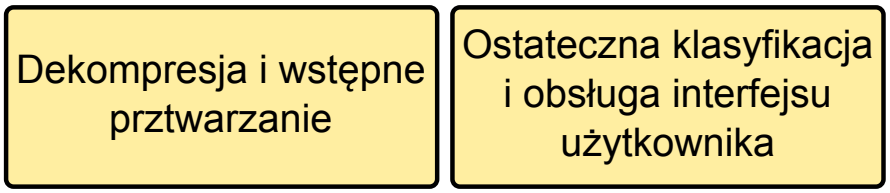
Kolejną cechą każdego znaku jest piktogram. Po odnalezieniu znaku, należy przeanalizować jego piktogram w celu rozpoznania. Nasuwającym się od razu rozwiązaniem jest podzielenie bazy na fragmenty. Stworzenie oddzielnej bazy dla każdego z kolorów, a nawet figur, powinno znacznie przyspieszyć przeszukiwanie i porównywanie zapisanych wzorców, gdyż będzie ich po prostu mniej.

Algorytm

Problem został podzielony na dwa moduły:

1. Moduł odpowiedzialny za dekompresję strumienia wideo oraz jego wstępną obróbkę i lokalizację kształtów,
2. Moduł będący interfejsem użytkownika, zajmujący się również końcową fazą procesu – próbą rozpoznania znaków.

Każdy z modułów działa w osobnym wątku, co umożliwia podzielenie obliczeń na dwa rdzenie procesora. Takie rozwiązanie nasuwa współczesna technika oraz popularność w dzisiejszych komputerach osobistych procesorów wielordzeniowych.



Rysunek 4.5: Główne moduły aplikacji

4.2.2. Wstępne przetwarzanie

W pierwszym etapie, po dekompresji ramki filmu, dochodzi do binaryzacji obrazu. Przetestowano dwa sposoby tej operacji:

1. W przestrzeni barw RGB,
2. W przestrzeni barw HSI.

Pierwsze z podejść okazało się zbyt niedokładne. Wyznaczone w ten sposób obszary zajmowały zbyt dużo niepożądanych fragmentów obrazu. Testowane podejście obejmowało również stosunki poszczególnych składowych jak w [31]. Po szerszej analizie zdecydowano się na przestrzeń barw HSI. Ze względu na konieczność konwersji (filmy są zapisywane za pomocą przestrzeni RGB), do progowania użyto tablic LUT (*ang. Look-Up Table*). Dzięki ich zastosowaniu progowanie sprowadza się do podstawienia odpowiedniej wartości (zera lub jedynki) z tablicy. Cały proces wyznaczania, które kolory z przestrzeni RGB odpowiadają zakresom wyznaczonym w przestrzeni HSI, odbywa się offline.

Binaryzacja

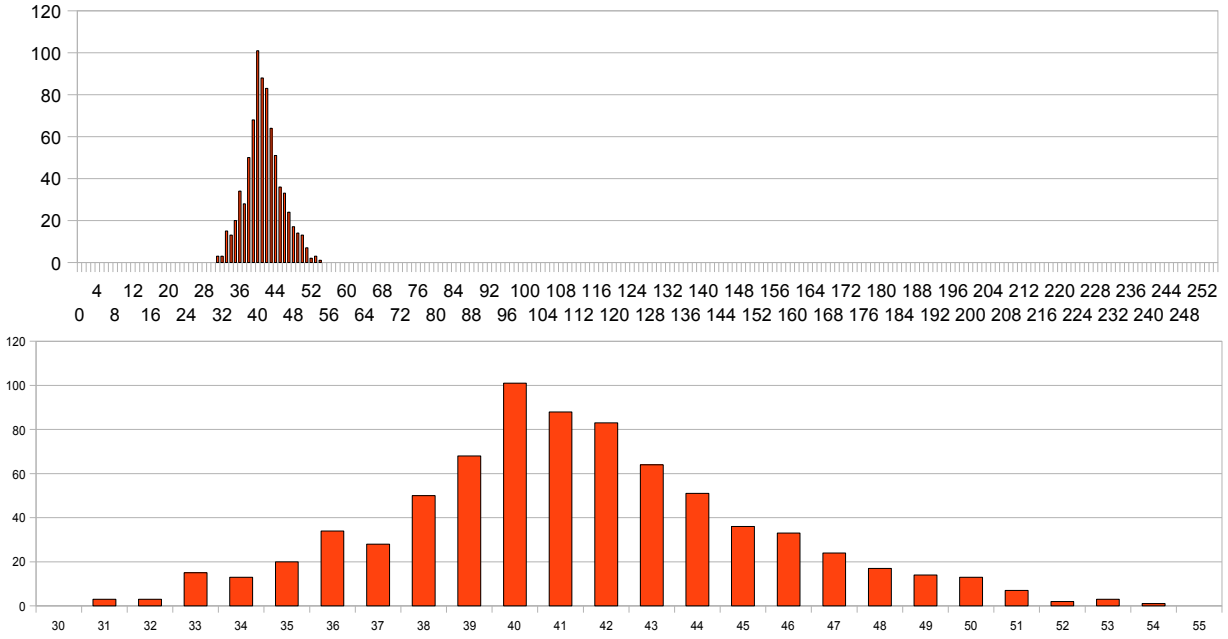
W celu wyznaczenia zakresów dla poszczególnych składowych przebadano rzeczywiste obrazy z występującymi znakami drogowymi i pobrano interesujące nas barwy znaków. Na ich podstawie wyznaczono histogramy z każdą ze składowych, a następnie zakresy. Analizie poddano trzy podstawowe kolory znaków: żółty – znaków ostrzegawczych, niebieski – znaków informacyjnych i nakazu, czerwony – znaków zakazu. Każda z barw jest analizowana z osobna, tworząc odzielny obraz binarny. Takie podejście zaproponował dr Adrian Horzyk – promotor pracy. Widać, że część ze składowych mieści się w niewielkich przedziałach. Pozostałe są rozproszone po całym zakresie. Na podstawie histogramów wyznaczono zakresy poszczególnych składowych dla każdego koloru. Nie wszystkie kolory będą miały zakresy dla wszystkich składowych. Wyznaczone zakresy przedstawia tabela 4.1. Bardzo ważne jest aby tak dobrać przedziały, by uzyskane obszary nie pomijały żadnego ze znaków. Dodatkowo bardzo istotne jest to, żeby wyznaczone obszary zajmowały jak najmniej klatki filmu, dzięki czemu skracamy czas przetwarzania dalszych kroków. Jak widać te dwa kryteria są przeciwstawne. Należy tak dobrać te przedziały, aby były swego rodzaju "złotym środkiem" pomiędzy oboma kryteriami.

Algorytm działania części binaryzacyjnej wygląda następująco:

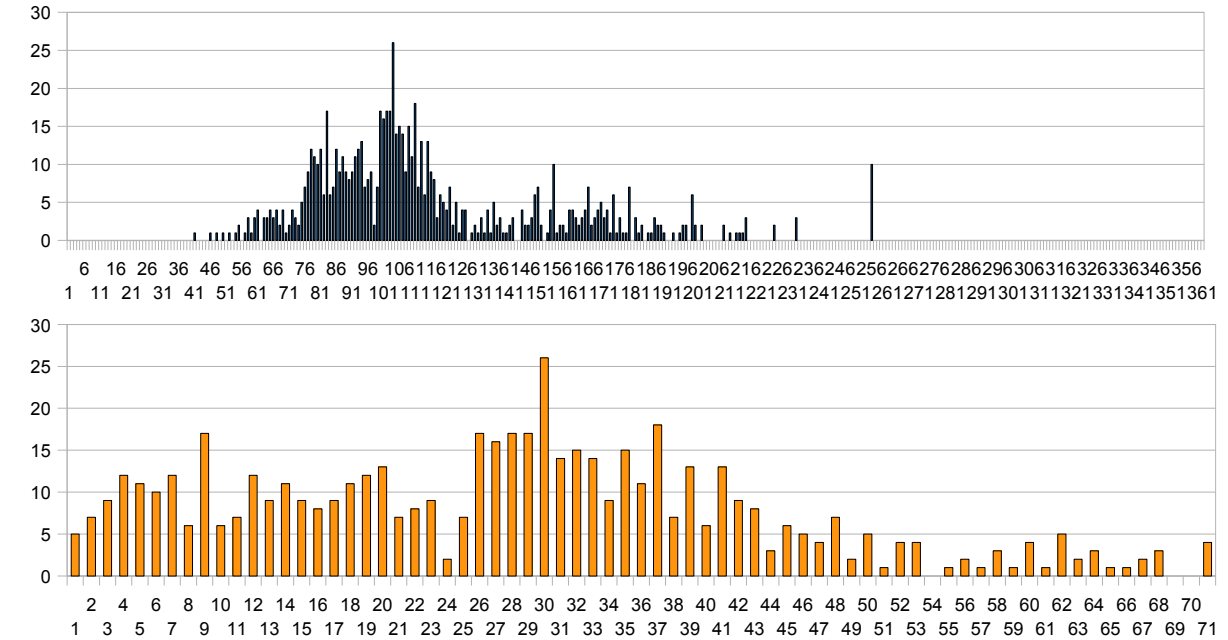
1. Określenie zakresów,
2. Wygenerowanie tablicy LUT (realizowane offline),
3. Dla każdego piksela obrazu dokonywane jest podstawienie wartości z tablic LUT,
4. Zwracane są trzy zbinaryzowane obrazy.

Kolor	Hue	Saturation	Intensity
żółty	$[31, 54]$	—	$[72, 151]$
niebieski	$[195, 205]$	—	$[55, 79] \cup [96, 116]$
czerwony	$[0, 24] \cup [60, 66]$	$[11, 42]$	$[67, 97]$

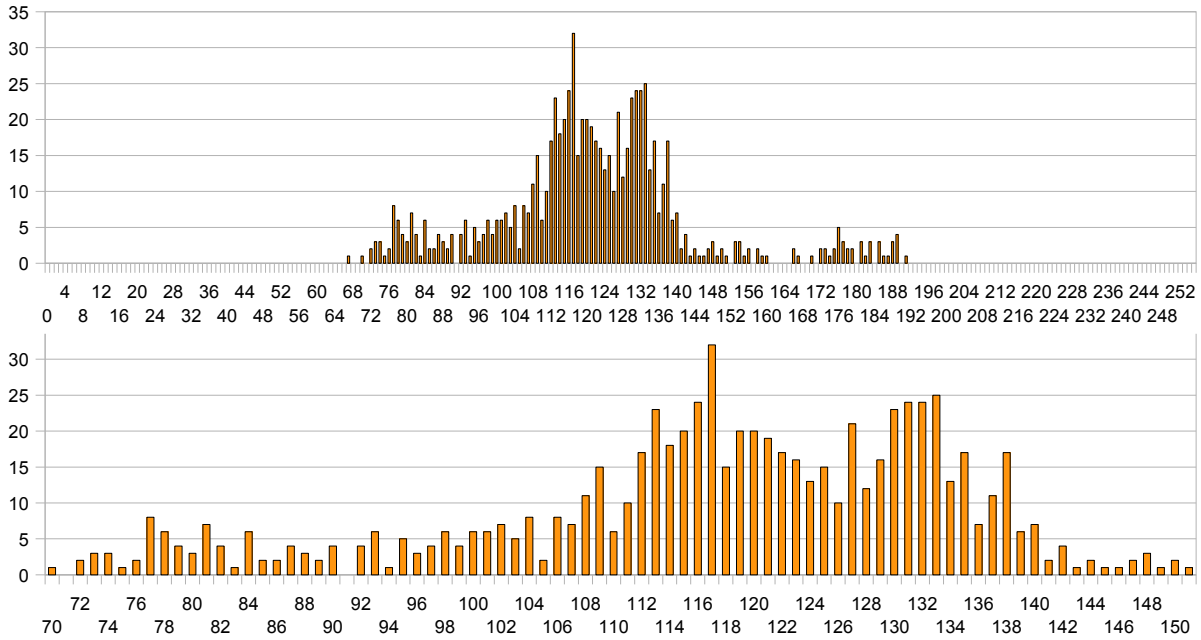
Tablica 4.1: Zakresy składowych HSI



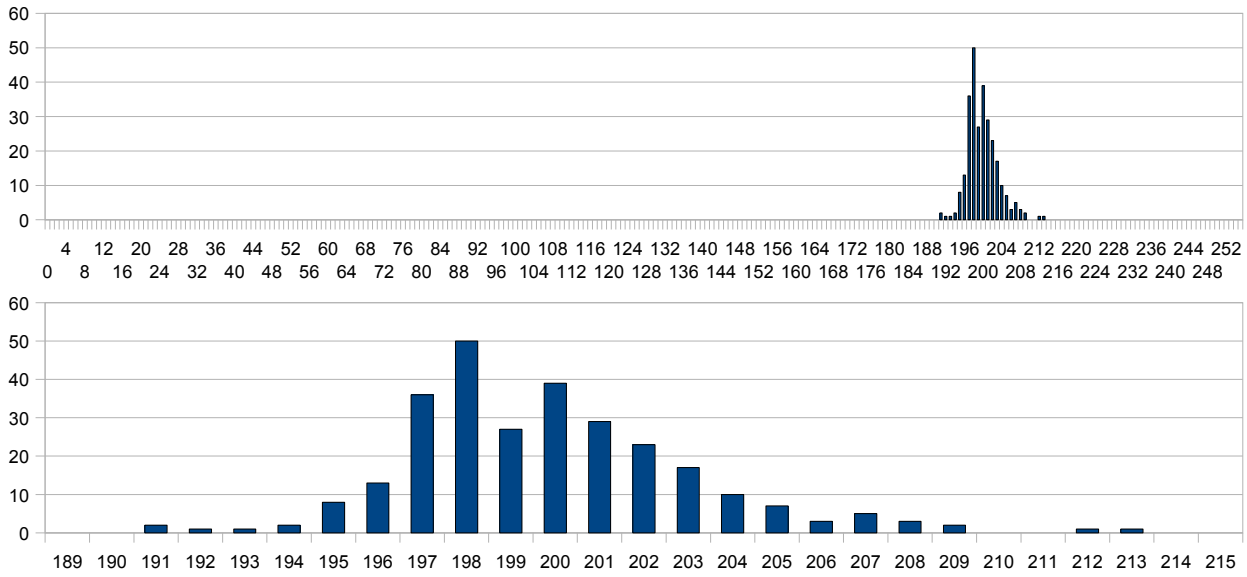
Rysunek 4.6: Częstość występowania poszczególnych wartości składowej **Hue** w kolorze żółtym



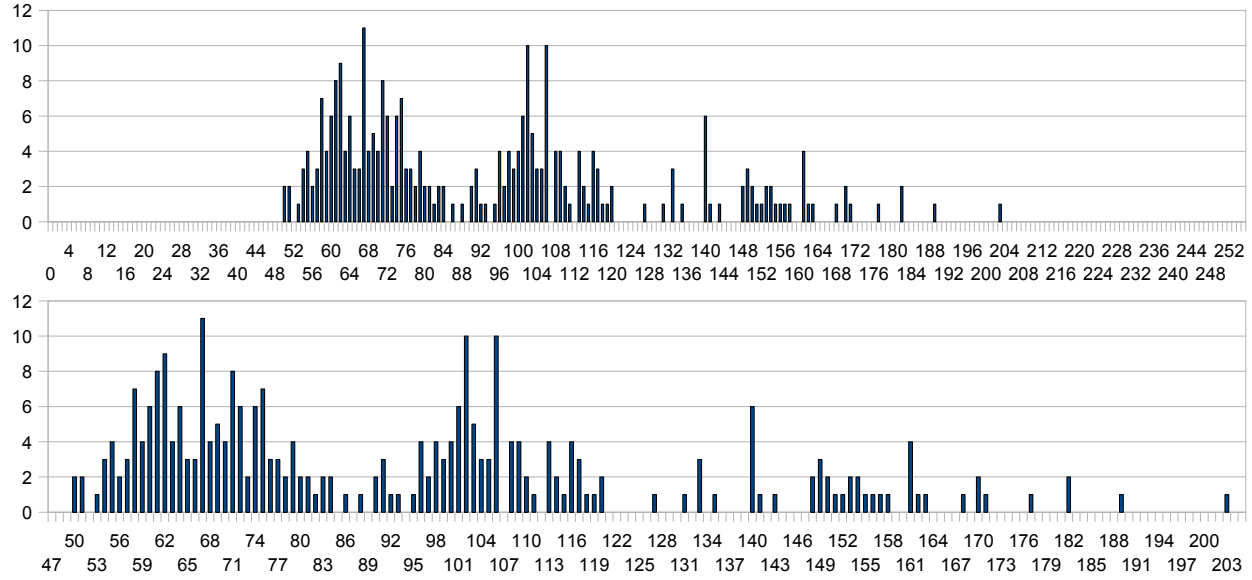
Rysunek 4.7: Częstość występowania poszczególnych wartości składowej **Saturation** w kolorze żółtym



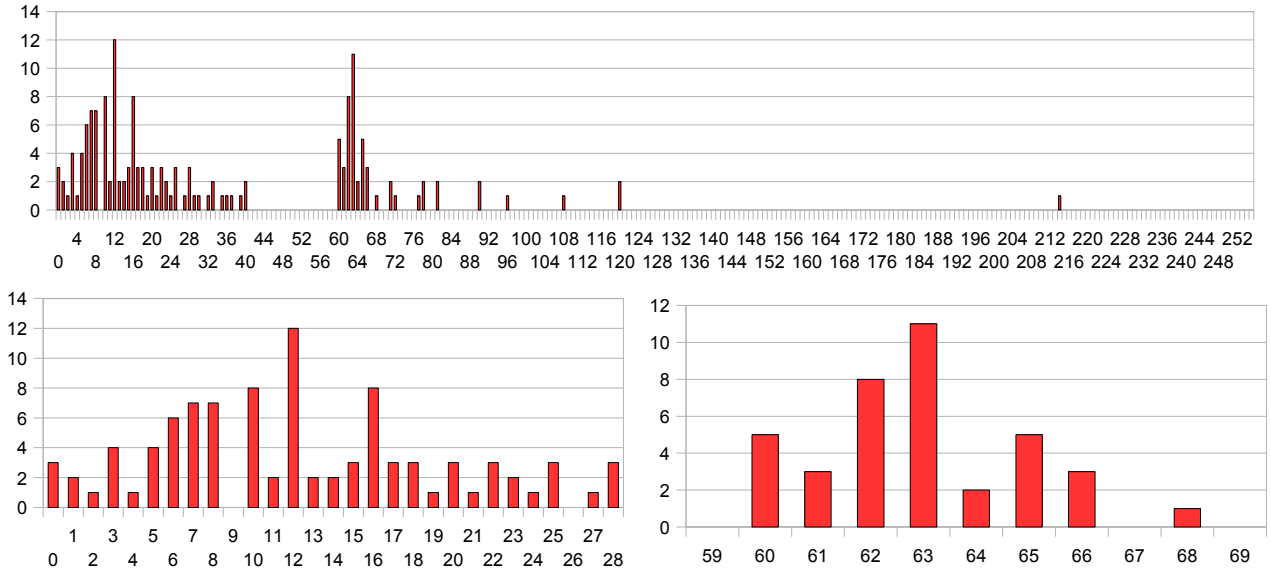
Rysunek 4.8: Częstość występowania poszczególnych wartości składowej **Intensity** w kolorze żółtym



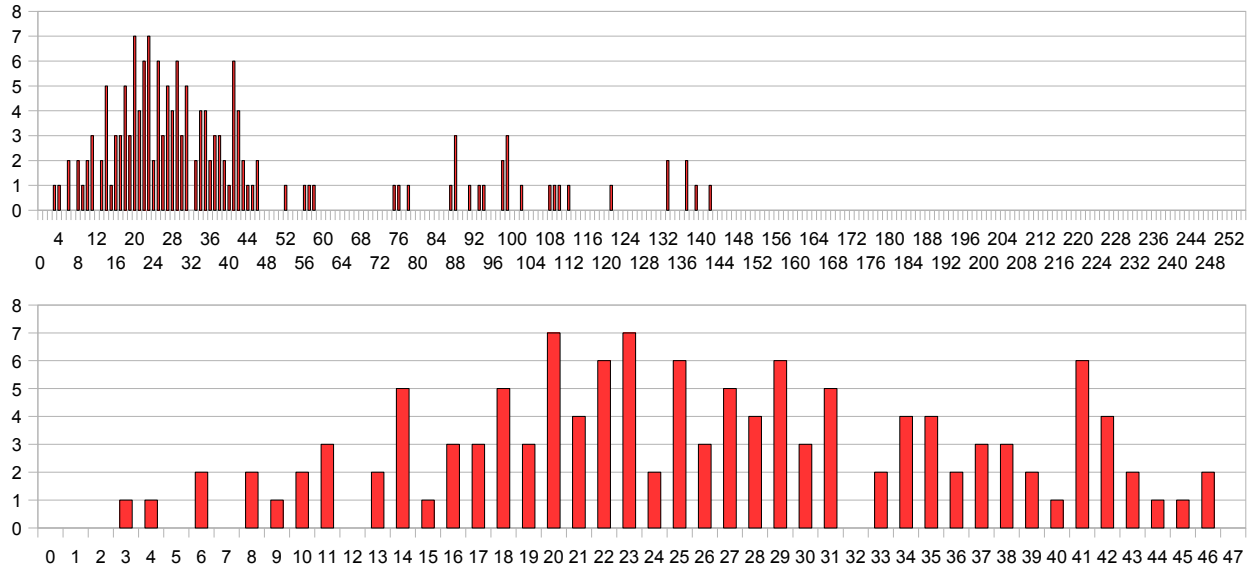
Rysunek 4.9: Częstość występowania poszczególnych wartości składowej **Hue** w kolorze niebieskim



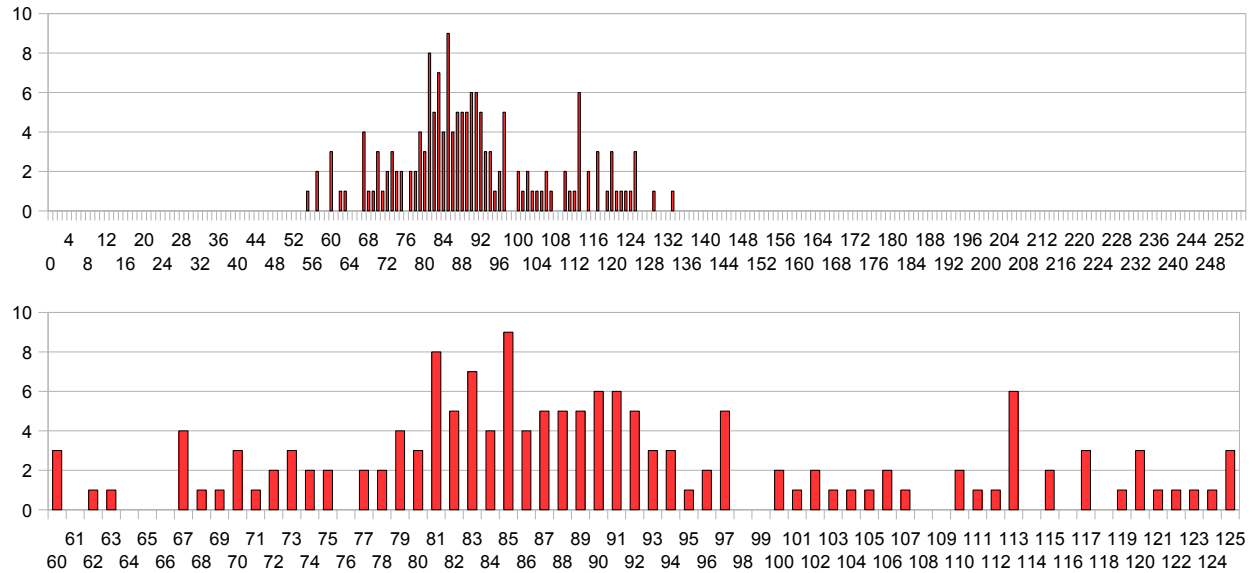
Rysunek 4.10: Częstość występowania poszczególnych wartości składowej **Intensity** w kolorze niebieskim



Rysunek 4.11: Częstość występowania poszczególnych wartości składowej **Hue** w kolorze czerwonym

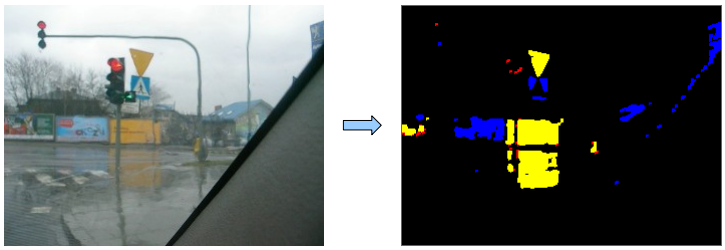


Rysunek 4.12: Częstość występowania poszczególnych wartości składowej **Saturation** w kolorze czerwonym



Rysunek 4.13: Częstość występowania poszczególnych wartości składowej **Intensity** w kolorze czerwonym

Po testach okazało się, że najlepszym sposobem binaryzacji jest analizowanie każdego koloru z osobna. wynikiem tego procesu są trzy obrazy przedstawiające obszary w poszukiwanych barwach. Efekt ich złączenia przedstawia rysunek 4.14. Jak widać jest to kompromis pomiędzy wcześniej wspo-



Rysunek 4.14: Efekt binaryzacji (połączenie trzech obrazów wynikowych)

mnianymi celami.

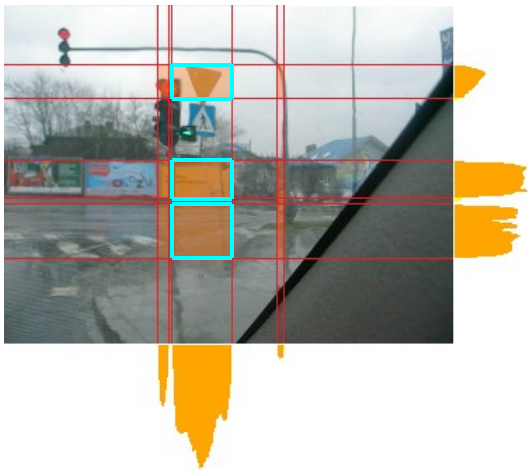
Wyznaczenie obszarów

Cały obraz został już podzielony na tło oraz miejsca, w których mogą znajdować się znaki drogowe. Kolejnym krokiem jest odsianie drobnych, kilku pikselowych zakłóceń. Filtracja ta jest realizowana przy pomocy operacji morfologicznej otwarcia. Wyjątkiem jest tutaj warstwa zawierająca kolor czerwony, która jest przetwarzana w inny sposób.

Po operacji otwarcia kolejnym krokiem jest wyznaczenie na obrazie prostokątów, w których mogą znajdować się znaki. W tym celu, dla każdego koloru liczona jest suma pikseli mogących należeć do znaku, w każdym wierszu i kolumnie. Tak wyliczone histogramy są poddawane analizie. Zgodnie z propozycją dr Adriana Horzyka, przeszukujemy oddzielnie oba histogramy, wyznaczając przedziały spełniające zadane kryteria. Podstawowym kryterium jest liczba pikseli. Jeśli jest ona większa bądź równa pewnemu progowi, to zaliczamy dany wiersz/kolumnę do tych przedziałów. Tak wyznaczone odcinki, są dalej weryfikowane. Każdy z nich musi spełnić warunki:

- Długość odcinka musi być większa od $1.2 \cdot minVal$, gdzie $minVal$ to zadany próg,
- Pole obszaru musi być większe od 250 pikseli,
- Każdy z wymiarów obszaru musi być większy od 15 pikseli.

Tak przeprowadzoną klasyfikację prezentuje rysunek 4.15. Tak wyznaczony obszar zostaje jeszcze po-



Rysunek 4.15: Lokalizacja obszarów zainteresowania

większony w każdą stronę o 4 piksele (jeśli oczywiście jest to możliwe). Celem powiększenia jest umożliwienie wykrycia krawędzi będących bardzo blisko, bądź pokrywających się z granicą obszaru. Metoda

ta skutecznie zakreśla nam obszary, w których należy poszukiwać znaków. Jej najważniejszą zaletą jest niewielka złożoność obliczeniowa – każdy z obrazów jest przeglądany w całości tylko jeden raz.

Analiza obszarów w poszukiwaniu figur

Po określeniu potencjalnych miejsc, w których mogą występować znaki drogowe, obszary te są poddawane dokładniejszej analizie. Algorytm tego etapu opiera się o wykrycie w każdym obszarze krawędzi. Do tego celu wykorzystywany jest algorytm Cannego. Następnie obraz z krawędziami jest analizowany poprzez wykorzystanie transformacji Hough w poszukiwaniu figur. W zależności od koloru danego obszaru poszukujemy innych figur. Dla obszarów żółtych będą to trójkąty, dla obszarów niebieskich – kwadraty i okręgi, a dla czerwonych tylko okręgi. Zostanie teraz pokrótce przedstawiony algorytm wyszukiwania i weryfikacji poszczególnych figur.

Transformacja Hough zwraca nam linie występujące w danym obszarze. Bierzemy dwadzieścia najbardziej intensywnych linii wyznaczonych przy pomocy transformacji. Następnie odsiewamy linie równoległe znajdujące się od siebie w odległości do 2 pikseli. Za linie równoległe uważamy linie, pomiędzy którymi kąt jest nie większy niż $15°$. Znając ich parametry wyznaczamy wszystkie ich przecięcia, występujące w obrębie danego obszaru. Tak wyznaczone przecięcia są poddawane analizie, tak aby wyszukać wśród nich trójkąty. Tak wyszukane figury muszą spełniać kilka warunków:

- Trójkąt nie może się pokrywać z żadnym już znalezionym trójkątem,
- Jego pole, wyliczone przy pomocy długości boków, musi być większe od 280,
- Stosunek boków najmniejszego prostokąta opisanego na tym trójkącie może się różnić od jedynki co najwyżej o 0.15,
- Musi to być trójkąt ostrokątny,
- Największy stosunek boków trójkąta może się odchylać od jedynki co najwyżej o 0.45.

Aby wyliczyć pole trójkąta wykorzystany jest wzór:

$$p = \frac{a+b+c}{2} \tag{4.1}$$

$$s = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)} \tag{4.2}$$

gdzie:

a,b,c – długości boków trójkąta

p – połowa obwodu trójkąta

s – pola trójkąta

Stosunek boków najmniejszego prostokąta opisanego na tym trójkącie liczymy zgodnie ze wzorem:

$$ratio = \frac{width}{height} \tag{4.3}$$

Natomiast największy stosunek boków trójkąta:

$$edgeRatio = \max\left\{\frac{a}{b}, \frac{b}{a}, \frac{a}{c}, \frac{c}{a}, \frac{b}{c}, \frac{c}{b}\right\} \tag{4.4}$$

Jeżeli trójkąt przejdzie pomyślnie przez tą selekcję, zostaje przekazany do dalszej obróbki.

Kwadraty są poszukiwane w podobny sposób. Różnica polega na warunkach, które musi spełniać:

- Prostokąt nie może się pokrywać z żadnym już znalezionym prostokątem,

- Jego pole musi być większe od 600,
- Stosunek boków najmniejszego prostokąta opisanego na tym prostokącie może się różnić od jedynki co najwyżej o 0.6,
- Największy stosunek boków trójkąta może się odchylać od jedynki co najwyżej o 0.5,
- Przeciwległe boki prostokąta muszą być równoległe (akceptowana różnica kąta to 5°).

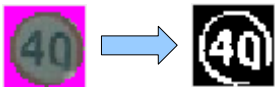
Pole prostokąta jest liczone po podzieleniu go na dwa trójkąty. Pozostałe wartości są liczone w podobny sposób jak miało to miejsce w wypadku trójkąta.

Do wyszukiwania okręgów zastosowano transformację Hough. Wyszukujemy okręgi o promieniu od 11 pikseli do połowy mniejszego wymiaru obszaru, jednak nie więcej niż 30 pikseli. Następnie bierzemy pięć najintensywniejszych okręgów, których intensywność jest większa od 35%. Okręgi przecinające się są eliminowane, ponieważ jest to mało prawdopodobne sytuacja, aby znaki tak intensywnie zachodziły na siebie.

Wyjątkiem od przedstawionego tu postępowania są obszary czerwone. Ze względu na próbę jak najściślejszego określenia koloru, jego występowanie jest najczęściej znikome na obrazie. Dlatego też w tym wypadku poszukujemy okręgów na całej ramce, bezpośrednio na obrazie binarnym.

Identyfikacja znaków

Ostatnim etapem jest dopasowanie tak znalezionej figury do któregoś ze znaków. Tutaj procedura dla wszystkich znaków jest podobna. Na podstawie fragmentu obrazu i odnalezionej figury zostaje ona wycięta, w ten sposób, żeby pozbyć się wszelkich niepotrzebnych informacji. Obraz jest skalowany do standardowego wymiaru (40x40 pikseli). Następnie jest on konwertowany na odcienie szarości i liczony jest histogram. Proces wydobycia piktogramu odbywa się poprzez obliczenie średniej histogramu, która jest później progiem podczas binaryzacji. W ten sposób z obrazu znaku otrzymujemy binarny piktogram, wraz z pewnymi zakłóceniami (rysunek 4.16). Ostatnim procesem jest przeszukanie bazy znaków i od-



Rysunek 4.16: Uzyskanie piktogramu z obrazu znaku

nalezienie piktogramu, którego dystans w przestrzeni CiSS od badanego obrazu jest mniejsza od 11%. Na tej podstawie znak jest identyfikowany.

W wypadku znaków trójkątnych, na początku sprawdzana jest orientacja znaku. Jeśli znak przedstawia odwrócony trójkąt, a jego współczynnik wypełnienia kolorem żółtym jest większy od 70%, to jest on uznawany za znak "Ustąp pierszeństwa przejazdu", gdyż ten znak nie zawiera żadnego piktogramu. Algorytm sprawdzenia orientacji jest następujący:

1. Wyznaczamy dwa wierzchołki, które są najbliżej siebie na osi Y, ich odległość oznaczamy przez $minDiff$,
2. Jeżeli $maxDiff \geq 1.5 \cdot minDiff$ oraz trzeci wierzchołek jest poniżej dwóch wyznaczonych, to znak może być znakiem "Ustąp pierwszeństwa przejazdu"
3. W przeciwnym przypadku to nie jest ten znak.

Podejście zaprezentowane powyżej stara się jak najlepiej spełniać założenia projektu. Przedstawione metody są stosunkowo mało złożone obliczeniowo. Ich cechy umożliwiają kompensację częściowych przysłonięć znaków (wykorzystanie transformaty Hough) znaków, pod warunkiem, że piktogram jest widoczny.

4.2.3. Projekt Aplikacji

Zgodnie z powyższą analizą i zaproponowanym algorytmem, została zaprojektowana aplikacja. Do dekompresji filmów wideo użyto biblioteki DirectShow, za część operacji graficznych jest odpowiedzialna biblioteka AForge.NET. Aplikację podzielono na dwa główne moduły:

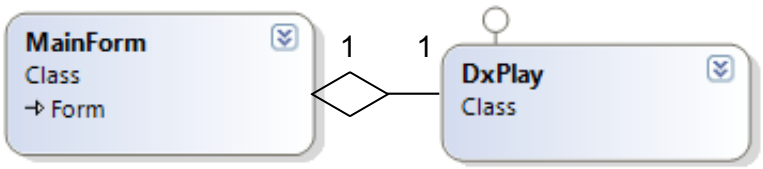
Capture – moduł służący do dekompresji filmu i wstępnej analizy, odpowiedzialny za:

- dekompresje filmu,
- binaryzacje kolejnych klatek filmu,
- wyznaczanie obszarów zainteresowania,
- identyfikacje kształtów w powyższych obszarach.

GUI – moduł do komunikacji z użytkownikiem i prezentujący wyniki:

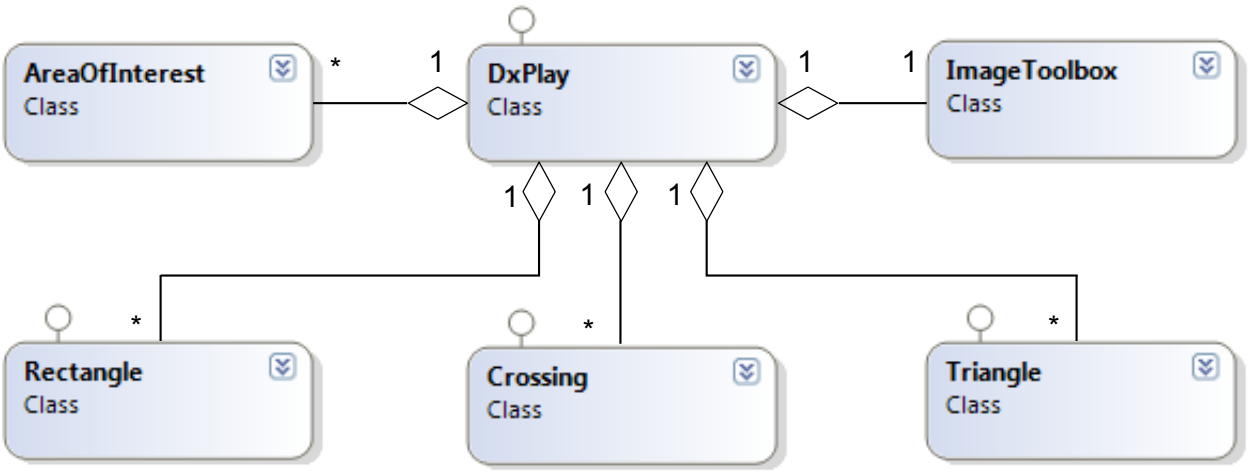
- umożliwia obsługę aplikacji,
- daje możliwość ingerencji w zakresy składowych barw dla podstawowych kolorów,
- umożliwia edycję baz piktogramów dla poszczególnych klas znaków,
- weryfikuje przekazane mu przez moduł **Capture** obszary i figury oraz rozpoznaje znaki.

Na podstawie tego podziału powstały dwie główne klasy: *MainForm* i *DxPlay*. System został zaprojektowany w taki sposób, żeby można było go podzielić na dwa komunikujące się ze sobą wątki. Nacisk położono również na zamknięcie podobnych funkcjonalności w klasach. Poniżej został zamieszczony



Rysunek 4.17: Schemat głównych klas

schemat zależności ważniejszych klas wchodzących w skład systemu:



Rysunek 4.18: Schemat zależności klas (DxPlay)

Opis funkcji ważnych klas oraz ich właściwości i metod został zebrany w tabelach 4.2 i 4.4. W tabeli 4.4 zostały podane jedynie nazwy właściwości i metod, w celu uproszczenia zapisu.

Rysunek 4.19: Schemat zależności klas elementów CiSS

Klasa	Opis
AreaOfInterest	Klasa zawierająca opis obszaru zainteresowania, wyznaczonego przez algorytm.
CiSS	Zawiera opis danego obiektu w przestrzeni CiSS.
CiSSCircle	Podstawowy element składowy przestrzeni CiSS – okrąg.
CiSSDB	Statyczna klasa odpowiedzialna za zapisywanie i odczytywanie obiektów klasy CiSS z plików.
CiSSSpace	Reprezentacja przestrzeni CiSS dla badanego obrazu. Zawiera wiele obiektów klasy CiSS.
Crossing	Klasa reprezentująca przecięcie dwóch linii. Wykorzystywana przy analizie wyników transformacji Hough.
DBEditForm	Okno pozwalające edytować bazy znaków.
DxPlay	Klasa odpowiedzialna za dekompresję i wstępne przetwarzanie obrazu.
Framer	Kontrola wyświetlająca pośrednie wyniki działania algorytmu.
ImageToolbox	Klasa zawierająca funkcja operujące na obrazach, np. progowanie, czy operacja otwarcia.
MainForm	Główne okno aplikacji, odpowiedzialne też za rozpoznawanie znaków.
Rectangle	Klasa opisująca wyszukane prostokąty.
Sign	Klasa reprezentująca znak.
SignInfo	Kontrolka wyświetlająca informacje o rozpoznanych znakach.
Triangle	Klasa opisująca wyszukane trójkąty

Tablica 4.2: Opis ważniejszych klas systemu

Metoda	Opis
CiSS	
Id	Identyfikator znaku.
Space	Lista obiektów CiSSCircle, reprezentująca dziedzinę.
Sum	Suma wszystkich pikseli obiektu w dziedzinie.
CiSSCircle	
Radius	Promień okręgu.
Value	Liczba pikseli obiektu na danym okręgu.
CiSSSpace	
Space	Tablica zawierająca całą przestrzeń.
LocatePattern()	Funkcja szukająca obiektu w przestrzeni, zwraca koordynaty jego środka, oraz różnicę w pikselach.
Crossing	
X, Y	Współrzędne przecięcia prostych.
Lines	Numery linii, które się przecinają
getAllCrossingsWithLine()	Statyczna funkcja wyszukująca wszystkie linie przecinające się z zadaną linią.
getAllCrossingsWithLines()	Statyczna funkcja sprawdzająca czy dwie linie się przecinają.
ImageToolbox	
ApplyTripleLutTable	Funkcja binaryzująca.
Dilation8bits()	Funkcja dokonująca operacji dylatacji na obrazie, w którym na każdy piksel przypada osiem bitów.
Erosion8bits()	Funkcja dokonująca operacji erozji na obrazie, w którym na każdy piksel przypada osiem bitów.
GenerateSeparateLutTables()	Funkcja generująca tablicę LUT.
Histogram()	Funkcja analizuje podane obrazy i zwraca dwa histogramy – w osi x i y.
LoadTripleLutTable()	Funkcja ładująca do pamięci tablicę LUT.
Rectangle	
EdgeRatio	Wartość stosunku boków (por. równanie 4.4).
MinimumRatio	Wartość stosunku przekątnych.
S	Wartość pola prostokąta.
X, Y	Współrzędne środka.
Sign	
BinBmp	Obraz binarny piktogramu.
Bmp	Wycięty obraz znaku.
fillFactor	Stosunek pola figury do jej wypełnienia poszukiwanym kolorem.
ustap	Wskazuje czy znak jest znakiem “Ustąp pierwszeństwa przejazdu”.
Triangle	
EdgeRatio	Wartość stosunku boków (por. równanie 4.4).
Ratio	Wartość stosunku boków prostokąta opisanego na danym trójkącie.
S	Wartość pola trójkąta.
X, Y	Współrzędne środka.

Tablica 4.4: Opis ważniejszych metod i właściwości klas systemu

5. Realizacja i badanie efektywności systemu

5.1. Implementacja

System został zaimplementowany zgodnie z projektem. Implementacji dokonano w środowisku MS Visual Studio 2008, w języku C# przy pomocy .NET Framework oraz dodatkowych bibliotek: DirectShowNet i AForge.NET. Największy nacisk został położony na wydajność systemu. Jedyną przeszkodą w implementacji algorytmu na platformie .NET może być stosunkowo mała wydajność kodu zarządzanego. Problem ten został rozwiązany poprzez zaimplementowanie wszystkich funkcji graficznych w blokach kodu niezarządzanego, co znacznie zwiększyło wydajność i prędkość działania.

5.1.1. Opis dodatkowych bibliotek

DirectShowNet

Biblioteka DirectShowNet[2] umożliwia dostęp do systemowej biblioteki Microsoft DirectShow z poziomu aplikacji napisanej przy pomocy platformy .NET. Zarządzana biblioteka Microsoftu umożliwiająca taki dostęp, jest o wiele mniej kompletna. Zawiera ona prawie wszystkie interfejsy z oryginalnej biblioteki i umożliwia łatwy do nich dostęp. Dzięki tym zaletom, jak i dobrej dokumentacji zdecydowałem się na wykorzystanie jej w systemie do dekompresji filmu i udostępniania algorytmowi kolejnych ramek.

AForge.NET

AForge.NET[1] to biblioteka napisana w języku C#, zaprojektowana z myślą o programistach i badaczach zajmujących się analizą i przetwarzaniem obrazów, czy zagadnieniami sztucznej inteligencji. Biblioteka składa się z kilku podstawowych modułów:

AForge.Imaging – zbioru procedur i filtrów operujących na obrazach

AForge.Vision – biblioteki przetwarzania obrazów

AForge.Neuro – biblioteki związanej z sieciami neuronowymi

AForge.Genetic – biblioteki związanej z algorytmami genetycznymi

AForge.MachineLearning – zbioru związanego z maszynami uczącymi

AForge.Robotics – biblioteki zapewniającej dostęp do funkcji niektórych zestawów robotów (np. Lego Mindstorms)

AForge.Video – zestawu funkcji obsługi sekwencji wideo

Ze względu na bardziej skomplikowany i czasochłonny sposób dostępu do sekwencji wideo w bibliotece AForge, został on zrealizowany przy pomocy wcześniej wspomnianej biblioteki [2].

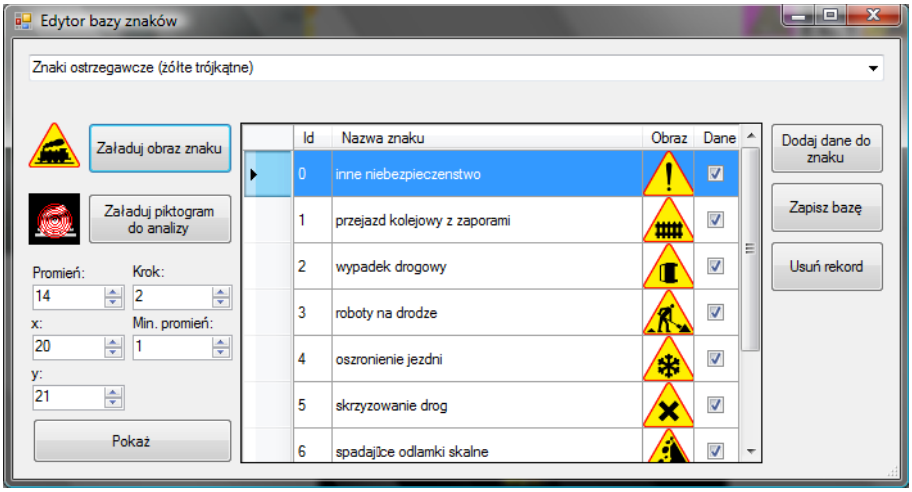
Przy pomocy tej biblioteki są realizowane takie funkcje jak wykrywanie krawędzi, czy transformacja Hough.

5.2. Zrealizowana aplikacja

Na podstawie powyższego projektu i z wykorzystaniem wcześniej wspomnianych bibliotek, została wykonana aplikacja: Po lewej stronie na górze jest wyświetlany oryginalny obraz. Pod nim oraz po jego

Rysunek 5.1: Główne okno aplikacji

prawej stronie rysowane są histogramy dla poszczególnych składowych. Wiersze/kolumny, które spełniają kryteria ilości pikseli są rysowane ciemniejszym kolorem. W prawym dolnym rogu Znajduje się obraz zbinaryzowany, który powstał w wyniku sklejenia trzech składowych obrazów. Są na niego nanoszone znalezione obszary, wyznaczone linie oraz rozpoznane figury. Po prawej stronie są wyświetlane aktualne rozpoznania. Każdy znak jest wyświetlany przez około jedną sekundę. Aplikacja zawiera również wbudowany edytor bazy znaków: Dzięki niemu w prosty sposób możemy dodawać nowe znaki, jak



Rysunek 5.2: Edytor bazy znaków

i edytować już istniejące. Umożliwia on również bardzo proste generowanie zapisu piktogramu w przestrzeni CiSS.

5.3. Testowanie aplikacji i algorytmu (skuteczność rozpoznania)

Testy przeprowadzono na laptopie z Procesorem Intel Core 2 Duo 2x2.4GHz, z 4Gb pamięci RAM oraz systemem operacyjnym Microsoft Windows Vista Home Premium 32bit. Algorytm został zbadany zarówno pod względem czasu przetwarzania jednej ramki, jak i jego skuteczności. Do testów posłużyło pięć filmów wideo prezentujących podróż samochdem. Zawierały one różne znaki drogowe. Łączna długość filmów to 54 sekundy. Na Wszystkich sekwencjach testowych pojawia się łącznie dziewięć znaków drogowych. Wszystkie zostały nakręcone w dzień, przy trudnych warunkach pogodowych. Przykładową ramkę filmu testowego przedstawia rysunek 5.3.

Rysunek 5.3: Przykładowa klatka z jednego filmów testowych

5.3.1. Testy wydajności

Czas obliczeń został wyznaczony osobno dla etapu wyznaczania obszarów i osobno dla rozpoznawania znaków. Czasy które były mniejsze od 1ms zostały uznane za bliskie zeru i za takie liczone. Tabela 5.1 przedstawia wyniki testów dla detekcji, rozpoznawania oraz wynik sumaryczny dla znaków drogowych: Z przeprowadzonych badań widać, że algorytm jest w stanie przeanalizować średnio 10.25

	Czas [ms]		
	Detekcja	Rozpoznanie	Razem
minimalny	43.00	0.00	43.00
maksymalny	314.00	63.00	324.00
średni	95.35	2.17	97.52

Tablica 5.1: Czasy działania algorytmu

ramek na sekundę, co powinno być wystarczającą ilością do sprawnego działania w samochodzie. Wynik taki jest spowodowany po części jakością i umiejscowieniem kamery w sekwencjach testowych.

Materiał był zbierany poprzez filmowanie drogi z miejsca pasażera, przez co często w kadr dostaję się fragment słupka przedniej szyby, lub przyciemnienia szyby, tworzącego kolory zbliżone do niebieskich znaków. Skutkowało to w wyznaczaniu dodatkowych, nieistniejących w rzeczywistości obszarów, które były analizowane. Wynik z pewnością można by jeszcze poprawić dopracowując progowanie barwne poprzez zastosowanie bardziej wyrafinowanej przestrzeni barwnej, niewrażliwej na zmiany otoczenia, czy też lepszej kamery, umiejscowionej poza kabiną pojazdu. Zwiększenie wydajności można by również uzyskać poprzez lepszą optymalizację kodu algorytmu.

5.3.2. Testy skuteczności

Algorytm został poddany również testom skuteczności rozpoznawania i detekcji znaków. Za skuteczną detekcję uznajemy wykrycie znaku chociażby na jednej z klatek, na której występuje. Udane rozpoznanie to prawidłowe podanie przez system znaczenia znaku. Fałszywe rozpoznanie to te, gdzie rozpoznano znak w miejscu, gdzie na prawdę on nie występuje.

Główną wadą algorytmu w obecnej postaci jest praktyczny brak wykrywania znaków prostokątnych. Jest to spowodowane ich stosunkowo małą różnicą kolorystyczną od tła (głównie nieba). Obszary występowań znaków często są dobrze odnajdowane. Niestety filtr krawędziowy Cannego, nie wykrywa zbyt wyraźnie krawędzi tych znaków, co prowadzi do nie odnalezienia prostokąta i niemożliwości detekcji znaku.

Tabela 5.2 przedstawia statystyki detekcji i rozpoznania znaków.

	Skuteczność
Wykryto	90%
Rozpoznano	88%

Tablica 5.2: Skuteczność systemu

Biorąc pod uwagę prostotę metod zastosowanych w systemie oraz czas przetwarzania, wyniki są zachęcające do dalszego rozwijania systemu. Szansa na pojawienie się fałszywego rozpoznania na pojedyńczej klatce wynosi około 1%. Część błędnych rozpoznań znaków była spowodowana trudnymi warunkami panującymi na filmach. System nie został przestowany na filmach pokazujących piękną słoneczną pogodę. Były one kręcone w deszczu, a na szybie samochodu zbierała się woda, co było przyczyną znacznych deformacji znaków. Pokazuje to, że założenie działania w ciężkich warunkach zostało spełnione.

5.3.3. Porównanie z innymi algorytmami

Poniższe tabele przedstawiają wyniki działania innych algorytmów w porównaniu do przedstawionego w tej pracy. Jeśli podano parametry procesora, na którym przeprowadzono testy, wyniki takiego algorytmu zostały przeskalowane. Przeliczamy najpierw czas i taktowanie na liczbę instrukcji, a następnie dzielimy przez częstotliwość 2GHz, zgodnie ze wzorem:

$$t_n = \frac{t \cdot freq_{orig}}{freq_{scale}}, \tag{5.1}$$

gdzie:

t_n – przeskalowany czas

t – czas oryginalny

$freq_{orig}$ – oryginalne taktowanie procesora

$freq_{scale}$ – nowa częstotliwość

Niestety nie wszystkie wyniki udało się w ten sposób przeliczyć. Te, których nie udało się przeliczyć zostały oznaczone znakiem "*". Tabela 5.3 zawiera porównanie czasów wykoania. Skuteczność detekcji i rozpoznania została najpierw zestawiona z algorytmami przetestowanymi w podobnych deszczowych warunkach (tabela 5.4). Nie wszystkie opracowania posiadały wyniki zarówno dla detekcji i rozpoznania. Jeśli brakowało któregoś wyniku, w tabeli oznaczono to znakiem "**?**". W celu jeszcze lepszego zobrazowania wyników, w tabeli 5.5 zestawiono wyniki opracowanej metody z algorytmami testowanymi w lepszych warunkach. Jak widać pomimo prostoty, skuteczność detekcji nie odbiega od pozostałych

Opracowanie	Czas wykonania dla klatki [ms]
Opisywany algorytm	97.52
[10]	*700.00
[29]	*3300.0
[27, 28]	*6000.00
[19]	36450.00

Tablica 5.3: Porównanie czasów działań dla różnych podejść

Opracowanie	Detekcja	Rozpoznanie
Opisywany algorytm	90%	88%
[26]	29%	68%
[15] (CIECAM97)	85%	?
[15] (RGB)	82%	?
[15] (CIELUV)	76%	?
[19]	75%	?
[15] (HSI)	73%	?

Tablica 5.4: Porównanie jakości detekcji i rozpoznawania znaków drogowych dla różnych podejść w ciężkich warunkach

Opracowanie	Detekcja	Rozpoznanie
Opisywany algorytm	90%	88%
[31]	88%	78%
[27, 28]	92%	?
[10]	?	94%

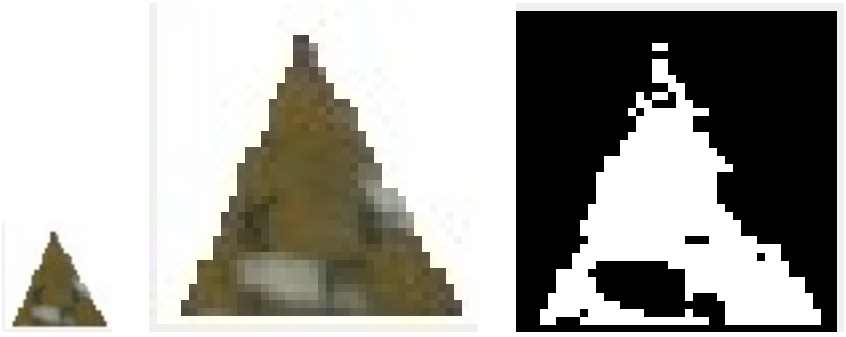
Tablica 5.5: Porównanie jakości detekcji i rozpoznawania znaków drogowych dla różnych podejść w dobrych warunkach

rozwiązań. Skuteczność jest porównywalna z innymi algorytmami, testowanymi w podobnych warunkach. Czasy wykonań są zachęcające, gdyż są zauważalnie mniejsze od podanych w innych publikacjach. Zestawienie porównujące metodę z innymi testowanymi w lepszych warunkach, również wypada pozytywnie. Wyniki porównania dowodzą skuteczności proponowanej metody i zachęcają do dalszego jej dopracowywania.

5.3.4. Analiza błędnych rozpoznań

Przyglądnijmy się przypadkom, w których znak został poprawnie zlokalizowany, natomiast nie udało się go zidentyfikować. Pierwszy z nich to znak ostrzegawczy "skrzyżowanie o ruchu okrężnym" (rysunek

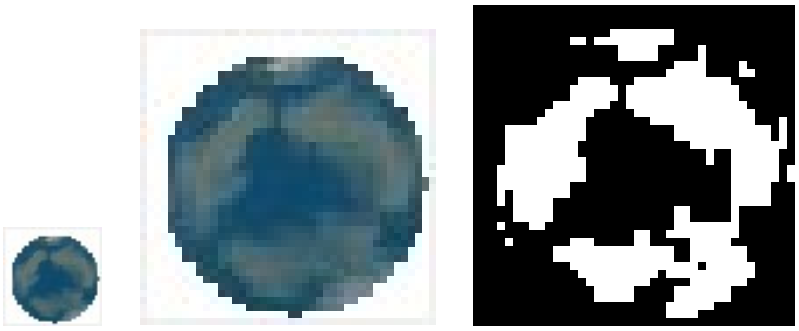
5.4), drugi to znak “ruch okrężny” (rysunek 5.5).



Rysunek 5.4: Wykryty znak “skrzyżowanie o ruchu okrężnym” (rozmiar oryginalny, powiększenie, wydobyty piktogram)

Na pierwszym widzimy, że krople deszczu zbierające się na szybie samochodu zupełnie zniekształciły zawartość znaku. Doszło również do znacznego osłabienia kontrastu pomiędzy żółtym tłem, a czarnym piktogramem. Było to przyczyną błędnego rozpoznania piktogramu, którego po prostu nie udało się odseparować od tła.

Drugi przypadek, to również zamazanie piktogramu przez deszcz. Poszczególne elementy zlewają się z sobą powodując błędne rozpoznanie. Co prawda piktogram zachował swój zarys, ale okazało się to niewystarczające do jednoznacznej identyfikacji.



Rysunek 5.5: Wykryty znak “ruch okrężny” (rozmiar oryginalny, powiększenie, wydobyty piktogram)

Jedyną metodą przeciwdziałania takim przypadkom byłoby zamontowanie kamery poza samochodem. Konstrukcja obudowy takiej kamery musiałaby być tak pomyślana, żeby krople deszczu, śniegu czy błota, nie osadzały się na obiektywie. Zapewniłoby to dobrą jakość materiału wideo i znacznie ułatwiło rozpoznawanie znaków.

5.4. Możliwości dalszej pracy nad algorytmem i aplikacją

Prezentowany algorytm jest bazą wyjściową do dalszych prac. W celu zwiększenia skuteczności można pokusić się o implementację lepszego filtra krawędziowego. Powinno to zwiększyć skuteczność detekcji znaków oraz umożliwić rozpoznawanie znaków informacyjnych. Również fazę rozpoznawania można poddać dalszym rozwinięciom. Rozpoznania otrzymane w wyniku działania algorytmu CiSS można dodatkowo weryfikować za pomocą przestrzeni RaSS, opisanej w [20].

Cała aplikacja może zostać lepiej zoptymalizowana, co może znacznie wpłynąć na czas działania algorytmu. Kolejnym krokiem byłoby zaimplementowanie części obliczeń na układzie GPU komputera. Większość współczesnych komputerów posiada karty graficzne zdolne do akceleracji grafiki 3D. Moc obliczeniową drzemiącą w tych układach można wykorzystać do przyspieszenia działania takich aplikacji, jak przedstawiony tu System Detekcji i Rozpoznawania Znaków Drogowych.

6. Podsumowanie

Opracowany algorytm pokazuje możliwości drzemiące w prostych algorytmach obróbki obrazów cyfrowych. Zamiast od razu implementować sieci neuronowe, czy algorytmy genetyczne, czasem warto zastanowić się, czy przypadkiem nie dałoby się użyć do nowych zastosowań, któregoś z klasycznych algorytmów. Okazuje się, że takie działanie często przynosi lepsze rezultaty niż można by się tego spodziewać. Wyniki testów zdają się potwierdzać, skuteczność takich rozwiązań i zachęcają do podejmowania kolejnych prób realizacji algorytmów przetwarzania obrazów w oparciu o takie techniki.

Powyższa praca prezentuje kolejne podejście do zagadnienia detekcji i rozpoznawania znaków drogowych. Dziedzina ta jest rozwijana na wielu uniwersytetach świata. Również firmy zaczynają się interesować takimi systemami. Praca pokazała, że stosując nawet bardzo proste techniki przetwarzania obrazu, możemy tworzyć skuteczne algorytmy detekcji i rozpoznawania obiektów na obrazach cyfrowych. Warto więc przeanalizować dane zagadnienie i zastanowić się nad jego rozwiązaniem, zamiast od razu decydować się na rozwiązania bardziej złożone obliczeniowo.

Zrealizowana aplikacja działa zaskakująco szybko biorąc pod uwagę implementację w języku wysokopoziomowym, który jest raczej uważany za dość wolny oraz fakt, że kod nie był w żaden sposób optymalizowany. Współczesne komputery mogą zaoferować nam spore moce obliczeniowe oraz duże możliwości multimedialne, co sprzyja implementacji różnych algorytmów analizy obrazu. Skuteczność algorytmu również zachęca do stosowania podobnych metod. Opisana technika analizy nie odbiega w swojej skuteczności od innych opisanych w publikacjach.

Myślę, że dziedzina ta będzie się dalej dynamicznie rozwijać i przyczyni się do wzrostu bezpieczeństwa na drogach. Bardzo możliwe, że w niedługim czasie zobaczymy pojazdy będące w stanie analizować swoje otoczenie i pomagające kierowcy w bezpiecznym dotarciu do celu. Ważne jest żeby kierowca zawsze miał świadomość, że pomimo całej techniki, która znajduje się we współczesnych pojazdach, to zawsze on i jego decyzje będą miały kluczowe znaczenie dla bezpieczeństwa.

Zaprezentowane tu metody obróbki obrazu oraz opisane rozwiązania pokazują nam prawdziwy geniusz ludzkiego ciała. Pomimo wielu lat badań, niezależnych i zupełnie różnych podejść, komputery w dalszym ciągu nie są w stanie dorównać człowiekowi w postrzeganiu i analizie otaczającego środowiska. Człowiek pomimo swojej wiedzy nie potrafi dokładnie przeanalizować procesów zachodzących w mózgu, a co za tym, idzie odwzorować ich w maszynie. Kolejne próby są jednak coraz skuteczniejsze.

W chwili obecnej najpopularniejsze stają się techniki skupiające swój potencjał na jednej wąskiej grupie znaków. Takie metody stają się już na tyle efektywne, że są stosowane komercyjnie w pojazdach. Głównym celem staje się stworzenie kompleksowego systemu będącego w stanie rozpoznawać wszystkie rodzaje znaków. Zadanie to jest mniej trywialne ze względu na różnorodność znaków. Zaprezentowane tutaj prace naukowe oraz opracowany przeze mnie algorytm dowodzą jednak, że postęp w tej dziedzinie jest systematyczny i w niedługiej przyszłości możemy się spodziewać właśnie takiego systemu.

A. Opis bazy znaków w przestrzeni CiSS

Baza znaków w projekcie składa się z dwóch podstawowych plików:

- nazwa_bazy.db
- nazwa_bazy.visual

Pierwszy z nich zawiera informację o samym piltogramie i jego wartościach. Drugi posiada informację o wyświetlaniu. Jest tam nazwa znaku, czy też nazwa pliku z jego obrazem. Plik *nazwa_bazy.db* posiada następującą strukturę:

```
id|pixel_sum|radius:value[|radius:value]\n
```

Gdzie:

id – identyfikator znaku

pixel_sum – liczba wszystkich pikseli w danym piktogramie

radius – promień danego okręgu

value – liczba pikseli na tym okręgu

Plik *nazwa_bazy.visual* ma podobną strukturę:

```
id|nazwa|nazwa_pliku\n
```

Gdzie:

id – identyfikator znaku

nazwa – nazwa znaku drogowego

nazwa_pliku – nazwa pliku obrazu znajdującego się w podkatalogu *Images*

Jeśli posiadamy tak zdefiniowaną bazę, możemy ją wczytać w następujący sposób:

```
using CiSS;

List<CiSS.CiSS> wzory;
Dictionary<int, KeyValuePair<string, Bitmap>> nazwy;
Dictionary<int, string> nazwyPlikow;

wzory = new List<CiSS.CiSS>(CiSSDB.Read("nazwa_bazy.db"));
nazwy = new Dictionary<int,
                       KeyValuePair<string,
                                    Bitmap>>(
                                    CiSSDB.ReadNames(
                                    "nazwa_bazy.visual"
```

```
                                    ));
nazwyPlikow = new Dictionary<int,
                             string>(
                             CiSSDB.ReadFileNames(
                             "nazwa_bazy.visual"
                             ));
```

Aby przeszukać obszar wykorzystujemy funkcję *MatchPattern*:

```
CiSSSpace space = new CiSSSpace(step, min, max, srcBtmap);

double[] wynik = CiSSDB.MatchPattern(wzory, space);

string nazwa wyniku = nazwy[(int)wynik[2]];
```

B. Zawartość płyty CDROM

Dołączona do pracy płyta CDROM zawiera następujące katalogi:

- **Praca magisterska - LaTeX** – katalog zawierający tekst pracy magisterskiej w formacie LaTeX
  - **obrazy** – wykorzystane w pracy obrazy w formacie PDF
- **Praca magisterska - PDF** – katalog zawierający tekst pracy magisterskiej w formacie PDF
- **Testowe filmy** – katalog zawierający sekwencje wideo wykorzystane do testowania algorytmu
- **Zrealizowany projekt** – katalog zawierający stworzoną aplikację
  - **Aplikacja** – katalog zawirający skompilowaną wersję projektu wraz ze wszystkimi koniecznymi do uruchomienia plikami
    - **Images** – obrazy znaków drogowych wyświetlane w aplikacji
  - **Źródła** – kod źródłowy w postaci projektu dla Visual Studio 2008

Bibliografia

[1] Aforge.net framework. `http://www.aforgenet.com/framework/`.

[2] Directshownet library. `http://directshownet.sourceforge.net`.

[3] S. Abe. *Support Vector Machines for Pattern Classification*. Springer, 2005.

[4] H. Akatsuka and S. Imai. Road signposts recognition system. In *SAE vehicle highway infrastructure: safety compatibility*, page 189–196, 1987.

[5] N. Bartneck and W. Ritter. Colour segmentation with polynomial classification. In *11th Int. Conf. on Pattern Recognition, volume 2*, page 635–638, 1992.

[6] B. Besserer, S. Estable, and B. Ulmer. Multiple knowledge sources and evidential reasoning for shape recognition. In *IEEE 4th Conference on Computer Vision*, page 624–631, 1993.

[7] B. Cyganek. Circular road signs recognition with soft classifiers. *Integr. Comput.-Aided Eng.*, 14(4):323–343, 2007.

[8] B. Cyganek. Real-time detection of the triangular and rectangular shape road signs. In *Advanced Concepts for Intelligent Vision Systems*, pages 744–755. AGH - University Of Science and Technology, Kraków, Poland, 2007.

[9] A. de la Escalera, J. Armingol, and M. Salichs. Traffic sign detection for driver support systems. In *3rd International Conference on Field and Service Robotics, Espoo, Finlandia*, Leganes, Madrid, Spain, June 2001. Universidad Carlos III de Madrid.

[10] M. de Saint Blancard. *Road sign recognition: A study of vision-based decision making for road environment recognition*. Springer-Verlag, 1992.

[11] S. Estable, J. Schick, F. Stein, R. Janssen, R. Ott, W. Ritter, and Y.-J. Zheng. A real-time traffic sign recognition system. In *Intelligent Vehicles'94*, page 213–218, 1994.

[12] A. A. Farag and A. E. Abdel-Hakim. Detection, categorization and recognition of road signs for autonomous navigation. In *Advanced Concepts for Intelligent Vision Systems*. University of Louisville, September 2004. http://www.cvip.uofl.edu.

[13] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2003.

[14] L. D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[15] X. Gao, K. Hong, P. Passmore, L. Podladchikova, and D. Shaposhnikov. Colour visionmodel-based approach for segmentation of traffic signs. December 2007.

[16] R. Ghica, S. Lu, and X. Yuan. Recognition of traffic signs using a multilayer neural network. In *Can Conf. on Electrical and Computer Engineering*, 1994.

[17] S. Haykin. *Neutral Networks*. Prent-Hall, 1999.

[18] N. Kehtarnavaz, N. Griswold, and D. Kang. Stop-sign recognition based on color-shape processing. In *Machine Vision and Applications, volume 6*, page 206–208, 1993.

[19] D. Kellmeyer and H. Zwahlen. Detection of highway warning signs in natural video images using color image processing and neural networks. In *IEEE Conf. Neural Networks, volume 7*, page 4226–4231, 1994.

[20] H. Y. Kim and S. A. de Araujo. Rotation, scale and translation-invariant segmentation-free shape recognition. April 2007.

[21] D. Krumbiegel, K.-F. Kraiss, and S. Schreiber. A connectionist traffic sign recognition system for onboard driver information. In *5th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems 1992*, page 201–206, 1993.

[22] M. Lalonde and Y. Li. Survey of the state of the art for sub-project 2.4 - roadsign recognition. Technical report, Centre de recherche informatique de Montreal, August 1995.

[23] K. Mikołajczyk and C. Shmid. An affine invariant interest point detector. In *ECCV*, pages 128–142, 2002.

[24] K. Mikołajczyk and C. Shmid. A performance evaluation of local descriptors. In *CVPR*, pages 257–263, 2003.

[25] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[26] C. F. M. Paulo. Detection and recognition of traffic signs. September 2007.

[27] G. Piccioli, E. D. Michelli, and M. Campani. A robust method for road sign detection and recognition. In *European Conference on Computer Vision*, page 495–500, 1994.

[28] G. Piccioli, E. D. Michelli, P. Parodi, and M. Campani. Robust road sign detection and recognition from image sequences. In *Intelligent Vehicles'94*, page 278–283, 1994.

[29] L. Priese, J. Klieber, R. Lakmann, V. Rehrmann, and R. Schian. New results on traffic sign recognition. In *Intelligent Vehicles'94*, page 249–253, 1994.

[30] W. Ritter. Traffic sign recognition in color image sequences. In *Intelligent Vehicles'92 Symposium*, page 12–17, 1992.

[31] M. Shneier. Road sign detection and recognition. In *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*, 100 Bureau Drive, Stop 8230, Gaithersburg, June 2005. National Institute of Standards and Tehnology.

[32] R. Tadeusiewicz and M. Flasiński. *Rozpoznawanie obrazów [dokument elektroniczny]*. Biblioteka Gółwna AGH, 2000.

[33] R. Tadeusiewicz and P. Korohoda. *Komputerowa analiza i przetwarzanie obrazów*. Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków, 1997.

[34] Y. Zheng, W. Ritter, and R. Janssen. An adaptive system for traffic sign recognition. In *Intelligent Vechicles'94*, page 165–170, 1994.