Uniwersytet im. A. Mickiewicza w Poznaniu

Wydział Matematyki i Informatyki

kierunek: Informatyka

specjalizacja: Inżynieria Oprogramowania

Praca magisterska

Paweł Gadecki

Adam Kułakowski

Rozpoznawanie obrazów oparte o bazę wzorców - eksperyment

System rozpoznawania znaków drogowych

Praca magisterska

wykonana pod kierunkiem

doktora Krzysztofa Dyczkowskiego

Poznań, 2010

Poznań, 06.10.2010

Oświadczenie

My, niżej podpisani Paweł Gadecki i Adam Kułakowski, studenci Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczamy, że przedkładaną pracę dyplomową pt: "Rozpoznawanie obrazów oparte o bazę wzorców - eksperyment. System rozpoznawania znaków drogowych." napisaliśmy samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystaliśmy z pomocy innych osób, a w szczególności nie zlecaliśmy opracowania rozprawy lub jej części innym osobom, ani nie odpisywaliśmy tej rozprawy lub jej części od innych osób. Oświadczamy również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej. Jednocześnie przyjmujemy do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu nam dyplomu zostanie cofnięta.

.....

.....

Spis treści

Wstęp

Obecnie produkowane samochody bardzo często wyposażone są w tak zwane systemy pokładowego wspomagania kierowcy. Najczęściej są to systemy wielomodułowe, złożone z kilku niezależnych od siebie podsystemów takich jak system nawigacji czy system sygnalizujący odległość od przeszkód. Ich zadaniem jest dostarczanie określonych informacji kierowcy (bez bezpośredniej ingerencji w prowadzenie pojazdu). Systemy te bazują na szeregu dostępnych informacji takich jak:

- położenie geograficzne — system nawigacji satelitarnej GPS,
- odległości od przeszkód i innych pojazdów — czujniki odległości,
- cyfrowa interpretacja otoczenia pojazdu — kamery.

Jednym z istotnych modułów takiego systemu jest moduł przeznaczony do rozpoznawania mijanych znaków drogowych. Taka identyfikacja odbywać się ma na podstawie analizy obrazu z kamery lub kilku kamer rejestrujących otoczenie pojazdu. Działanie tego modułu podzielić można na kilka etapów. Musi on najpierw wykryć na obrazie przechwyconym przez kamerę obszary, w obrębie których występują lub mogą występować znaki drogowe, następnie zidentyfikować te znaki, po czym przekazać zebrane informacje osobie prowadzącej pojazd.

Informacje o ograniczeniach prędkości, drogach jednokierunkowych, przejazdach kolejowych itp. są często dostarczane wraz z mapami, na których bazują systemy nawigacji GPS. Nie można jednak na nich zbytnio polegać ze względu na takie czynniki jak brak aktualności, zbyt mała dokładność czy wreszcie błąd człowieka wprowadzającego te informacje. Tak dla kierowcy, jak i dla systemu mającego go wspomagać, jedynym pewnym źródłem tego typu informacji są znaki drogowe.

Sprawny system detekcji i rozpoznawania znaków działać musi w czasie rzeczywistym. W praktyce oznacza to, że analiza pojedynczej klatki obrazu musi trwać na tyle krótko, by informacja uzyskana na jej podstawie nie przestała być aktualna.

Cel i zakres pracy

Celem pracy jest sprawdzenie jakie rezultaty w rozpoznawaniu obrazów uzyskać można przy zastosowaniu wybranych przez nas metod takich jak: wnioskowanie rozmyte czy poszerzająca się baza wiedzy.

Pośrednim celem naszej pracy jest stworzenie systemu próbującego wykrywać oraz identyfikować mijane znaki drogowe w czasie rzeczywistym.

Zespół znaków drogowych stanowiących pod względem informacyjnym reprezentacje tego samego znaku (np. zakaz wjazdu) w praktyce nie jest grupą identycznych znaków. Poszczególne znaki z tego zespołu różnić się mogą odcieniem oraz, w ograniczonym stopniu, kształtem symbolu. Strumień wideo jest jedynie cyfrową interpretacją rzeczywistości. Na obraz ten istotny wpływ mają także warunki atmosferyczne oraz parametry samego urządzenia rejestrującego. W związku z powyższym informacje, na których bazuje system identyfikacji, uznać możemy za informacje nieprecyzyjne. Dlatego też tworzony przez nas system w procesach detekcji i identyfikacji znaków drogowych

wykorzystywać będzie elementy logiki rozmytej.

Efektem naszej pracy są odpowiedzi na pytania:

- Jakie rezultaty uzyskamy przy zastosowaniu metodyki na jaką się zdecydowaliśmy?
- Czy zastosowanie tego typu rozwiązań w systemach mających analizować (wykrywać pewne obiekty oraz je rozpoznawać) dynamicznie zmieniający się obraz jest uzasadnione?

Rozwiązania, jakie zdecydowaliśmy się zastosować podczas projektowania systemu, to:

- segmentacja obrazu w oparciu o wnioskowanie rozmyte,
- analiza powstałego obrazu binarnego także na bazie reguł rozmytych,
- tworzenie reprezentacji wykrytych znaków drogowych w postaci wektorów liczbowych, z których każdy kolejny ma bardziej szczegółowo opisywać kodowany obraz (realizacja założenia kolokwialnie nazywanego "od ogółu do szczegółu"),
- rozpoznawanie znaków w oparciu o bazę wiedzy,
- stworzenie miary podobieństwa pozwalającej na porównanie pary kodów.

Na stworzony przez nas system składają się:

- baza wiedzy przechowująca reprezentacje znaków drogowych,
- narzędzia pozwalające na powiększanie bazy wiedzy,
- moduł wykrywania i rozpoznawania znaków drogowych.

Przed rozpoczęciem pracy zdefiniowaliśmy podstawowe oczekiwania jakie stawiamy wobec projektowanego systemu:

- działanie w czasie rzeczywistym – oznacza to, że informacja o rozpoznanym znaku powinna zostać dostarczona kierowcy, zanim znak ten zniknie z jego pola widzenia,
- powiększanie bazy wiedzy – możliwość dodawania nowych reprezentacji znaków drogowych z dowolnych nagrań filmowych,
- zastosowanie wnioskowania rozmytego przy podejmowaniu istotnych decyzji – system nie powinien podejmować kluczowych decyzji bazując na klasycznej (zero-jedynkowej) logice. Przez decyzje istotne rozumiemy tu te decyzje, które mają bezpośredni wpływ na uznanie badanego obszaru obrazu za znak drogowy i późniejszą jego identyfikację.

Ponieważ niniejsza praca jest pracą zespołową (dwuosobową) przed przystąpieniem do jej realizacji podzieliliśmy tworzenie systemu oraz samej pracy na zadania. Następnie podzieliliśmy pomiędzy siebie odpowiedzialność za realizację poszczególnych zadań.

Tabela 1 prezentuje sposób w jaki podzieliliśmy pomiędzy siebie pracę przy tworzeniu systemu.

Struktura pracy

"Wstęp" napisaliśmy wspólnie.

W rozdziale pierwszym (*"Podstawy teoretyczne"*) opisane zostały po krótce zagadnienia teoretyczne, na których bazuje stworzony przez nas system. Podrozdział *"Przetwarzanie obrazu"* (autor: Adam Kułakowski) skupia się na metodologii związanej z analizą obrazu. Przybliża takie zagadnienia jak:

Tab. 1. Podział pracy przy tworzeniu systemu na zadania oraz przypisanie tych zadań członkom zespołu

Zadanie	**Osoba odpowiedzialna za realizację**
Zaprojektowanie interface'u	Paweł Gadecki
Zebranie niezbędnego materiału filmowego	zadanie zespołowe
Segmentacja obrazu wejściowego	Adam Kułakowski
Lokalizacja znaków drogowych	Adam Kułakowski
Kodowanie wykrytych znaków drogowych	Adam Kułakowski
Stworzenie bazy danych reprezentacji znaków drogowych	Paweł Gadecki
Stworzenie relacji podbieństwa dla reprezentacji znaków drogowych	Paweł Gadecki

- segmentacja obrazu,
- podstawowe modele przestrzeni barw,
- klasyczne metody kodowania obrazu,
- metody rozpoznawania wzorców (algorytm najbliższego sąsiada, wykorzystanie sieci neuronowych).

Podrozdział *"Modelowanie informacji nieprecyzyjnej"* (autor: Paweł Gadecki) przybliża podstawowe zagadnienia teoretyczne z zakresu logiki rozmytej.

W rozdziale drugim (*"System detekcji i rozpoznawania znaków drogowych"*) opisujemy w jaki sposób działa stworzony przez nas system. Podrozdział *"Architektura systemu"* (autor: Paweł Gadecki) opisuje w bardzo ogólny sposób działanie poszczególnych modułów składających się na nasz system. Kolejne trzy podrozdziały (autor: Adam Kułakowski) opisują szczegółowo w jaki sposób:

- segmentowane są kolejne klatki strumienia wideo,
- wykrywane są znaki drogowe na obrazie binarnym uzyskanym w procesie segmentacji,
- kodowane są fragmenty obrazu uznane za znaki drogowe.

Pozostałe podrozdziały (autor: Paweł Gadecki) przybliżają:

- strukturę bazy danych oraz mechanizmy pozwalające na jej "uczenie się",
- stworzoną miarę podobieństwa pozwalającą na porównanie reprezentacji znaków drogowych,
- proces rozpoznania znaku drogowego wykrytego na analizowanym obrazie.

Ostatni rozdział (*"Rezultaty"*) napisaliśmy wspólnie. Prezentujemy w nim efektywność naszego systemu oraz porównujemy uzyskane rezultaty z wynikami podobnych systemów. W rozdziale tym odpowiadamy na pytania, jakie postawiliśmy sobie przed przystąpieniem do pracy.

Trudności napotkane podczas tworzenia systemu

Najbardziej kłopotliwą częścią systemu okazał się początkowy etap – segmentacja obrazu wejściowego. Początkowo bazowaliśmy tutaj na modelu przestrzeni barw RGB. Rezultaty były jednak niezadowalające. Zdefiniowanie uniwersalnych zasad, które pozwoliłyby wyłonić fragmenty znaków drogowych i jednocześnie odrzucić pozostałe elementy nazywane dalej tłem, okazało się praktycznie niemożliwe. Wartości składowych modelu RGB okazały się zbyt mocno zależne od warunków, w jakich dokonano nagrania.

Dlatego zmieniliśmy koncepcję, postanowiliśmy zastosować model przestrzeni barw HSB. Po wielu analizach i testach udało nam się zdefiniować reguły segmentacji bazujące na wartościach składowych modelu HSB. Opis barw w modelu HSB jest bardziej intuicyjny i naturalny. Ostatecznie uzyskaliśmy wysoce zadowalające wyniki segmentacji.

Bardzo istotnym elementem systemu jest także kodowanie znaków drogowych. Musieliśmy wymyślić sposób kodowania, który jednocześnie:

- wygeneruje kod znaku, który będzie przechowywać informacje na temat kształtów, symboli i ich barw,
- dla analizowanej reprezentacji znaku drogowego pozwoli na bardzo szybkie wyodrębnienie z bazy danych grupy podobnych reprezentacji.

Zakładamy, że baza danych będzie się rozrastać, czas identyfikacji znaku drogowego powinien być niezależny od rozmiaru bazy danych. Dlatego musimy dysponować narzędziami pozwalającymi na wyodrębnienie grupy reprezentacji znaków drogowych zbliżonych do badanego w stałym czasie. Dopiero wzorce z tej grupy szczegółowo porównywać będziemy z wzorcem wygenerowanym dla badanego znaku drogowego.

Rozdział 1

Podstawy teoretyczne

System pracuje na strumieniu wideo. Może to być plik zawierający nagranie wideo lub strumień z kamery cyfrowej podłączonej do komputera. W czasie działania aplikacji strumień ten jest odtwarzany i wyświetlany, a poszczególne klatki obrazu analizowane pod kątem obecności znaków drogowych.

Analiza obrazu, której celem jest wyznaczenie obszarów mogących zawierać znaki drogowe, składa się z kilku etapów. System podejmujący decyzje o istotności poszczególnych pikseli lub obszarów oparty jest o wnioskowanie rozmyte. Na tym etapie analizy piksel uznany jest za istotny, jeśli może stanowić część znaku drogowego, obszar — jeśli znak drogowy lub jego część może znajdować się w jego (tego obszaru) obrębie.

Po wyodrębnieniu obszarów obrazu mogących zawierać znaki drogowe, każdy taki obszar zostanie "zakodowany", będzie reprezentowany przez wektory liczbowe. Wektory te następnie zostaną porównane z informacjami przechowywanymi w bazie danych przy zastosowaniu odpowiedniej miary podobieństwa.

1.1. Przetwarzanie obrazu

1.1.1. Segmentacja obrazu

Celem segmentacji obrazu jest jego podział na obszary (zbiory pikseli). Obszary te są jednorodne pod względem badanej własności (piksele do nich należące mieszczą się w tym samym przedziale).

Do najczęstszych kryteriów segmentacji należą: barwa, poziom szarości, tekstura.

Częstym efektem segmentacji jest obraz binarny, złożony z pikseli o wartościach: 0 – nie spełniających zadanych kryteriów oraz 1 – spełniających te kryteria.

Poniższe informacje bazują na rozdziale "Image Segmentation" ([1], str. 131 – 150), gdzie znaleźć można bardziej wnikliwą analizę omówionych metod.

Metody segmentacji obrazu podzielić możemy na trzy grupy oraz grupę hybrydową:

- segmentacja punktowa – rozważająca każdy piksel obrazu niezależnie od pozostałych, na podstawie branych pod uwagę własności przypisuje go do odpowiedniej klasy,
- segmentacja krawędziowa – bazuje na algorytmach wykrywania krawędzi, brane są pod uwagę kształty, wyszukiwanie zamkniętych obszarów,
- segmentacja obszarowa – dzieli obraz na obszary, które następnie poddaje analizie, w wyniku której obszary te mogą być łączone, dzielone, pomniejszane lub powiększane,

- segmentacja hybrydowa – oparta na wykorzystaniu przynajmniej dwóch z powyższych metod.

Progowanie

Segmentacja przez progowanie jest jedną z metod segmentacji punktowej. Najprostsze zastosowanie ma ona w odniesieniu do obrazów w odcieniach szarości, gdzie każdemu pikselowi przypisana jest jedna wartość – stopień szarości. Przyjmujemy jakąś wartość progową, może ona być sztywno określona, możemy także wyznaczać ją na podstawie histogramu obrazu. Wartość progowa może także być optymalizowana, to znaczy zmieniana na podstawie wyników wcześniejszej segmentacji.

Każdy z pikseli segmentowanego obrazu rozpatrujemy niezależnie, jeżeli jego poziom szarości osiąga (lub przekracza) wartość progową przypisujemy mu wartość 1, w przeciwnym razie wartość 0. W wyniku otrzymujemy obraz binarny, na którym niewygaszone punkty reprezentują części obrazu, które uznaliśmy za istotne. Obraz taki poddać następnie możemy dalszej analizie.

Rysunek 1.1 przedstawia efekt prostego progowania obrazu w odcieniach szarości.

Bardzo często podstawą wyboru wartości progowej jest histogram segmentowanego obrazu. Rozważmy na przykład obraz w odcieniach szarości składający się z ciemnych obiektów na jasnym tle. Jego histogram będzie miał dwa szczyty. Jako wartość progową możemy przyjąć punkt znajdujący się pomiędzy tymi szczytami, dla którego histogram ma najmniejszą wartość.

Opisane wyżej podejście jest oczywiście tylko prostym wariantem segmentacji progowej. Zamiast stopnia szarości piksela możemy brać pod uwagę funkcję na jego składowych w dowolnym modelu barw. Co więcej wyżej

Rys. 1.1. Segmentacja progowa, kolejne obrazy to wyniki segmentacji dla progu o wartości 50, 100 i 125 (w skali 0...255)

mówiliśmy o globalnie wyznaczanej wartości progowej, to znaczy jednej dla całego obrazu. Można także używać dynamicznego wyznaczania tej wartości – zależy wtedy ona od badanego punktu oraz jego otoczenia.

Klasteryzacja

Klasteryzacja jest także metodą segmentacji punktowej. Można powiedzieć, że jest pewnego rodzaju rozszerzeniem progowania. Wynikiem klasteryzacji nie musi być obraz binarny.

Każdy piksel przypisany zostanie do jednej z wielu (co najmniej dwóch) klas. W efekcie obraz podzielony zostanie na obszary punktów należących do danej klasy. Banalnym przykładem może być segmentacja na podstawie barw i przypisywanie każdego piksela do jednej z klas: czerwony, niebieski, czarny, pozostałe.

Metody krawędziowe

Istotą tych metod jest wykrycie na obrazie krawędzi, które dzielą go na obszary. Krawędź utożsamiamy z granicą pomiędzy obszarem ciemnym i jasnym lub na odwrót.

Metody te bazują na wyznaczeniu operatorów gradientowych, ich wartości mówią w jakim stopniu dany punkt należy do krawędzi. Tutaj także musimy dobrać wartość progu detekcji, po osiągnięciu której piksel uznajemy za przynależący do krawędzi.

Istnieje wiele zdefiniowanych operatorów wyznaczających gradienty w wybranych kierunkach.

Segmentacja obszarowa

Jednym z podejść jest metoda podziału obszarów. Zaczynamy od traktowania obrazu jako jednego obszaru. Zdefiniować musimy warunek jednolitości. Warunek ten, badając wszystkie punkty należące do danego obszaru, pozwala stwierdzić, czy obszar ten jest jednolity.

Sprawdzamy czy cały obraz stanowi obszar jednolity. Jeśli tak nie jest – dzielimy go na cztery obszary i dla każdego z nich sprawdzamy czy spełniony jest warunek jednolitości. Postępujemy tak dopóki każdy obszar, na jakie podzielono obraz nie zostanie uznany za jednolity.

Analogicznym podejściem jest łączenie obszarów. Obraz dzielimy na małe obszary, w skrajnym przypadku każdy piksel traktujemy jako osobny obszar. Następnie próbujemy łączyć sąsiadujące ze sobą obszary, to znaczy sprawdzamy, czy obszar powstały z ich połączenia będzie jednolity. Jeśli tak łączymy je, postępujemy tak dopóki możliwe jest połączenie dowolnych dwóch obszarów.

Możemy także połączyć powyższe metody, gdy niemożliwe jest dalsze łączenie obszarów próbujemy je ponownie dzielić i łączyć w inny sposób. Zakładamy wtedy na przykład, że szukamy podziału na jak najmniejszą ilość obszarów.

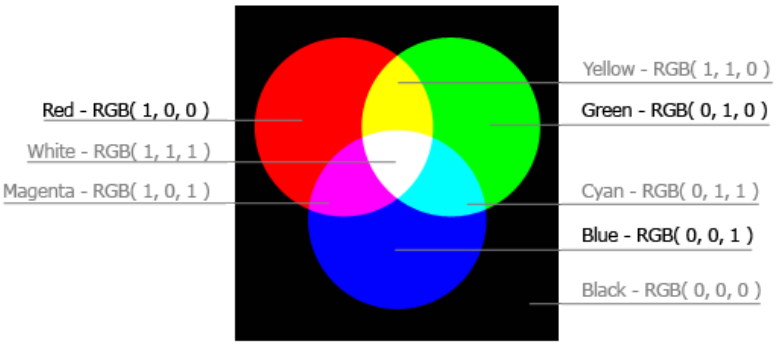
1.1.2. Model barw

Model barw to matematyczny model opisujący barwę poprzez zestaw (najczęściej trzech lub czterech) liczb. Jeden z podziałów modeli barw proponuje grupę modeli orientowanych na urządzenia (takie jak monitory i drukarki) oraz orientowanych na człowieka (bardziej intuicyjny opis barwy).

Źródłem poniższych informacji jest rozdział "Color Space" ([2], str. 3 – 7).

Model RGB

Nazwa powstała ze złożenia pierwszych liter jego barw składowych: Red, Green i Blue (barwa czerwona, zielona i niebieska). Model ten bazuje na syntezie addytywnej, czyli zjawisku mieszania barw przez sumowanie wiązek światła widzialnego.

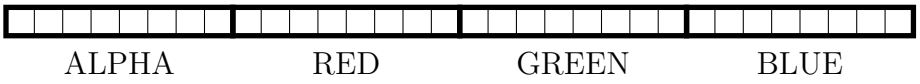


Rys. 1.2. Mieszanie addytywne barw, model RGB

Barwa wyjściowa powstaje więc z połączenia barw RGB w różnych proporcjach. Model przestrzeni barw RGB jest zależny od urządzenia. Poszczególne urządzenia mogą wyznaczać inne składowe dla danej barwy lub wyświetlać niejednakowe barwy dla tych samych składowych.

Reprezentacja binarna barwy w modelu RGB. Obraz, na którym pracuje system, reprezentowany jest ciągiem 32-bitowych liczb. Każdy piksel obrazu opisany jest więc za pomocą 32 bitów.

Każda barwa składowa modelu barw reprezentowana jest polem ośmiobitowym. Może więc przyjmować wartości opisujące jej nasycenie należące do przedziału $[0, 255]$. Pierwsze osiem bitów z 32-bitowego pola odpowiadającego pikselowi opisuje kanał alfa - odpowiednik współczynnika pochłaniania światła. Kanał alfa pozwala na manipulowanie stopniem przezroczystości poszczególnych pikseli, jest zupełnie nieistotny z naszego punktu widzenia. Kolejne trzy ciągi 8-bitowe opisują nasycenie odpowiednio barwą czerwoną, zieloną i niebieską.



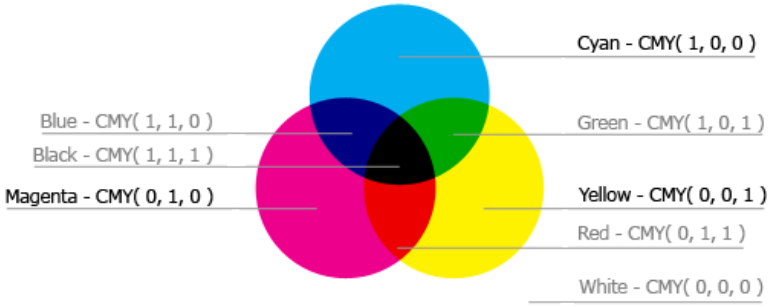
Rys. 1.3. 32-bitowa reprezentacja modelu przestrzeni barw ARGB

Używanie modelu przestrzeni barw ARGB (Alpha Red Green Blue) jest powszechne także w sytuacjach, w których współczynnik alpha nie odgrywa żadnej roli. Obecne procesory bazują najczęściej na słowach długości równej 32 bitom lub jej wielokrotności.

Model CMY(K)

Nazwa tego modelu jest także wynikiem zestawienia pierwszych liter barw składowych: Cyan, Magneta i Yellow. W tym modelu barwa powstaje nie tyle poprzez mieszanie barw składowych, co poprzez ich nakładanie na siebie.

Model wykorzystywany jest w urządzeniach takich jak drukarki oraz plottery. Wyposażone są one w atramenty o barwach składowych (substancje barwiące przepuszczające światło). Barwa wynikowa powstaje poprzez nałożenie na siebie każdej składowej w odpowiednim nasyceniu. Zakładamy, że podłoże jest białe, dlatego brak jakiejkolwiek barwy składowej daje w wyniku barwę białą.



Rys. 1.4. Nakładanie na siebie barw składowych w modelu CMY

Teoretycznie nałożenie na siebie wszystkich składowych w 100 procentowym nasyceniu tworzy barwę czarną. W praktyce jednak barwniki w urządzeniach nie odpowiadają idealnie barwom składowym modelu, dlatego uzyskanie idealnej czerni ze zmieszania barw składowych nie jest możliwe. Poza tym barwa czarna jest najczęściej używana w druku, więc uzyskiwanie jej ze zemiszania trzech barw składowych jest marnotrawstwem.

Dlatego urządzenia takie jak drukarki wyposażone są dodatkowo w barwnik czarny. W takiej sytuacji mówimy o modelu CMYK, gdzie litera K wywodzi się od czarnego barwnika (blacK lub Key black).

Model HSB (HSV)

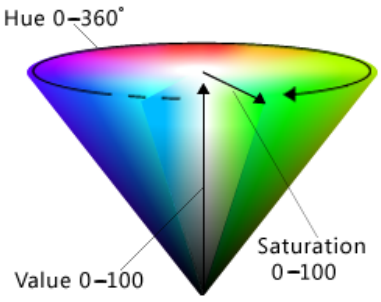
Model ten opisuje barwę w sposób bardziej zrozumiały (intuicyjny) dla człowieka. Jego nazwa wywodzi się od pierwszych liter nazw poszczególnych składowych opisujących barwę:

Hue – odcień barwy (spektrum, barwa spektralna) wyrażony kątem ($[0° \dots 360°]$ lub $[0 \dots 2\pi]$),

Saturation – nasycenie barwy wyrażone liczbą rzeczywistą $\in [0 \dots 1]$ (lub $[0\% \dots 100\%]$),

Brightness – (lub Value) jasność (moc światła białego) wyrażona liczbą rzeczywistą $\in [0 \dots 1]$ (lub $[0\% \dots 100\%]$).

Trójwymiarową reprezentacją tego modelu jest stożek lub ostrosłup o podstawie sześciokąta foremnego.



Rys. 1.5. Graficzna reprezentacja modelu barw HSB

1.1.3. Sposoby kodowania obrazu

Celem kodowania obrazu jest uzyskanie jego opisu. Opis ten powinien być możliwie zwięzły oraz zawierać jak najwięcej informacji. Na ogół chcemy też, by opis ten był niezmienny ze względu na podstawowe transformacje obrazu, przede wszystkim ze względu na skalowanie. Oznacza to, że na przykład opisy zdjęcia tego samego obiektu w różnych rozmiarach powinny być do siebie zbliżone (identyczne w idealnym modelu).

Większość metod opisu obrazu bazuje na analizie jego kształtu lub tekstury (zagęszczenie krawędzi, analiza histogramu).

Źródłem poniższych informacji jest pozycja [3], str. 223 – 233.

Sygnatura

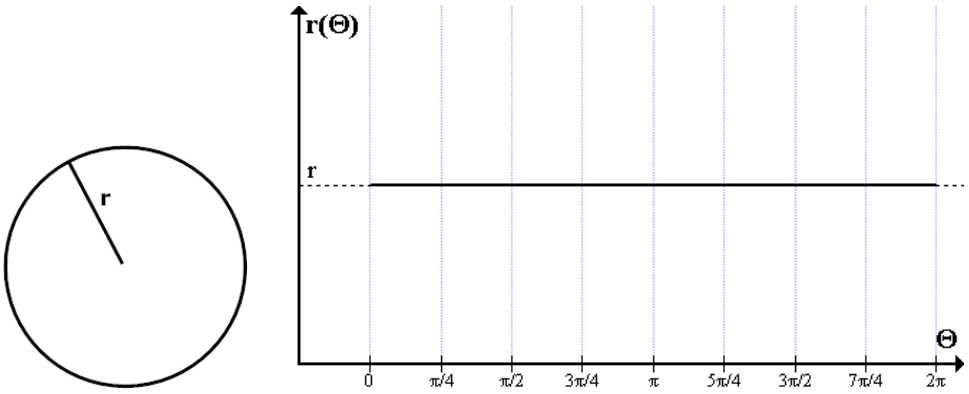
Sygnatura to jednowymiarowa funkcja odwzorowująca brzeg obszaru.

Innymi słwy jest to opis obrazu poprzez opisanie jego konturu jednowymiarową funkcją. Opis ten wyrażany jest jako zależność pewnej miary od ustalonego punktu. Na poniższych przykładach (rys. 1.6 i 1.7) widzimy opisy kształtów kwadratu oraz okręgu jako odległości euklidesowej od centroidu opisywanego obiektu.

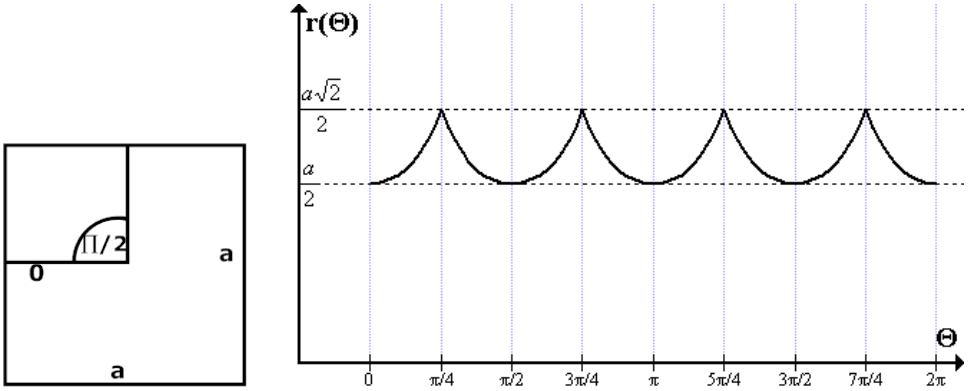
Opis krawędzi

Próbujemy opisać kształt obrazu za pomocą pewnej łamanej. We wstępnej fazie musimy przeprowadzić segmentację wyodrębniającą kontury opisywanego obiektu.

Gdybyśmy następnie próbowali wyznaczyć zespół odcinków opisujący obraz okazałoby się prawdopodobnie, że jest to bardzo duża ilość krótkich odcinków. Dlatego jedynie przybliżamy kształt obiektu za pomocą tych odcinków.

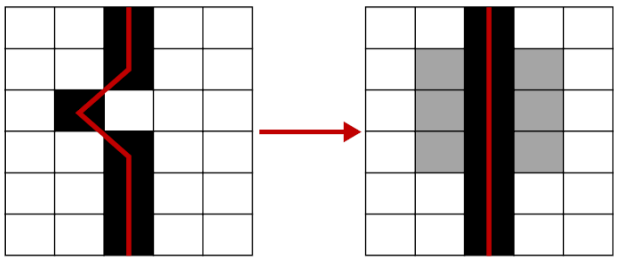


Rys. 1.6. Opis kształtu dla okręgu o promieniu r



Rys. 1.7. Opis kształtu dla kwadratu o boku a

Jednym z podejść jest uśrednianie współrzędnych punktów tworzących kontur w oparciu o współrzędne jego sąsiadów. Takie podejście prezentujemy na przykładzie (rys. 1.8).

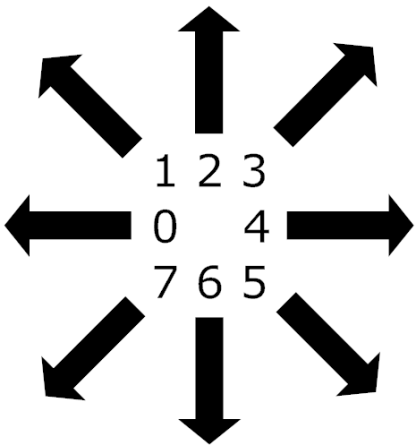


Rys. 1.8. Opis krawędzi obiektu za pomocą przybliżenia ciągiem odcinków (łamaną)

Na rysunku 1.8 po lewej stronie widzimy odcinki wyznaczone dla fragmentu konturu powstałego w wyniku zastosowania algorytmu wykrywania krawędzi. Po prawej stronie każdemu pikselowi tworzącemu ten kontur zmieniliśmy współrzędne na uśrednioną wartość współrzędnych pikseli go otaczających. Dzięki temu zabiegowi otrzymujemy mniej (za to dłuższych) odcinków opisujących kształt analizowanego obiektu. Pozwala to na redukcje zaburzeń wynikających tak z samej jakości analizowanego obrazu, jak i z zastosowanego algorytmu wykrywania krawędzi.

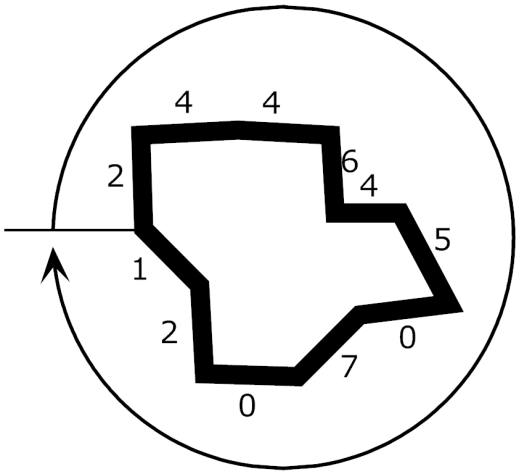
Opis kształtu przy pomocy kodów łańcuchowych

To podejście jest zbliżone do poprzedniego. Chcemy opisać kształt konturu badanego obiektu za pomocą ciągu liczb. Każdej liczbie przypisujemy kierunek skierowania odcinka o znormalizowanej krawędzi. Sami dobieramy szczegółowość opisu, to znaczy definiujemy kierunki, pod jakimi skierowana może być krawędź (przykład na rys. 1.9). Oczywiście skierowanie poszczególnych fragmentów konturu często nie będzie w pełni odpowiadało żadnemu zdefiniowanemu, wybieramy najbliższe skierowanie.



Rys. 1.9. Przypisanie wartości liczbowych kierunkom

Na rysunku 1.10 prezentujemy opisanie skierowania krawędzi tworzących przykładowy kontur. Kod opisujący ten kształt to: 24464507021.



Rys. 1.10. Wyznaczenie kodu łańcuchowego opisującego przykładowy kształt

Taki sposób kodowania nie jest niezmienny ze względu na skalowanie. Oznacza to, że gdybyśmy powiększyli dwukrotnie badany obraz, dostalibyśmy inny (dłuższy) kod. Możemy długość odcinka, dla jakiej przypisujemy pojedynczą liczbę, uzależnić od rozmiaru badanego obiektu. Wtedy kod stanie się niezależny od rozmiarów badanego obiektu.

Aby kod taki był także niezmienny ze względu na obrót musimy ustalić jakąś jednoznaczną metodę doboru punktu startowego oraz stały kierunek kodowania. Możemy na przykład zaczynać w punkcie najbardziej oddalonym od centroidu i poruszać się wzdłuż konturu w kierunku zgodnym ze wskazówkami zegara.

Jest także inny sposób uniezależnienia kodów łancuchowych od punktu, w którym rozpoczęto je tworzyć. Traktujemy stworzony kod jako cykliczny, to znaczy następnikiem jego ostatniego elementu jest pierwszy element. Następnie dobieramy faktyczny pierwszy element kodu tak, by liczba, na jaką składają się wszystkie jego elementy była możliwie największa (lub najmniejsza).

1.1.4. Klasyczne algorytmy rozpoznawania wzorców

Celem rozpoznawania wzorców nie jest jedynie stwierdzenie, czy badany wzorzec jest taki sam jak jakiś ze znanych systemowi. Wzorce tworzą grupy (klasy), efektem rozpoznania ma być przypisanie badanego wzorca do jednej z klas.

System rozpoznający wzorce musi być oparty o pewną bazę wiedzy przechowującą reprezentacje wzorców przypisane do konkretnych klas. Aby zdefiniować jakąkolwiek klasę wzorców musimy wzbogacić wiedzę systemu o przynajmniej jedną reprezentację tej klasy. Załóżmy, że tworzymy system rozpoznający znaki, jedną z klas wzorców będzie więc klasa opisująca cyfrę 1. Istnieje wiele wariantów zapisania tej cyfry. Przykładowy zestaw wzorców definiujący klasę "jedynka" przedstawiamy na rysunku 1.11.

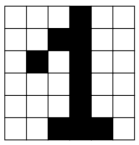
Rys. 1.11. Przykładowy zestaw definiujący klasę wzorców "jedynka"

Algorytm najbliższego sąsiada

Poniższe informacje zaczerpnięte są z pozycji [1], str. 162.

Jest to bardzo proste podejście do problemu. Nasz zbiór wiedzy, zwany tutaj zbiorem uczącym, zawiera pewną ilość obserwacji. Każda obserwacja składa się z ciągu $X_1, X_2, \ldots, X_n$ zmiennych objaśniających oraz Y – zmiennej objaśnianej.

Wartością zmiennej objaśnianej jest klasa, do jakiej wzorzec należy (patrz rys. 1.12).



```
X={0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,1,0,0,
   0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,1,1,0}
Y='1'
```

Rys. 1.12. Zmienne objaśniające i wartość zmiennej objaśnianej dla wzorca jedynki

W metodzie tej porównujemy badany wzorzec ze wszystkimi wzorcami ze zbioru uczącego. Szukamy wzorca najbliższego badanemu. Oznacza to, że szukamy wektora zmiennych jak najbliższego badanemu. Do wyznaczenia odległości pomiędzy wektorami posługujemy się wybraną miarą niepodobieństwa (po więcej informacji można sięgnąć po: [4], str. 136).

Badany wzorzec przypisujemy więc do klasy, do której należy najbliższy mu wzorzec ze zbioru uczącego. Dodatkowo możemy założyć, że jeżeli minimalna różnica przekracza jakąś zdefiniowaną wartość, to badanego wzorca nie przypisujemy do żadnej klasy, aby uniknąć wyników niedorzecznych.

Możemy także wyznaczyć grupę najbliższych sąsiadów i szukać klasy, dla której najwięcej wzorców znajduje się w tej grupie.

Analiza skupień – algorytmy hierarchiczne

Analizę skupień opisujemy w oparciu o pozycję [4], str. 345 - 352.

Celem analizy skupień jest grupowanie n obiektów (każdy z nich opisany jest wektorem p cech) w K skupień. Skupienia to grupy niepuste i rozłączne.

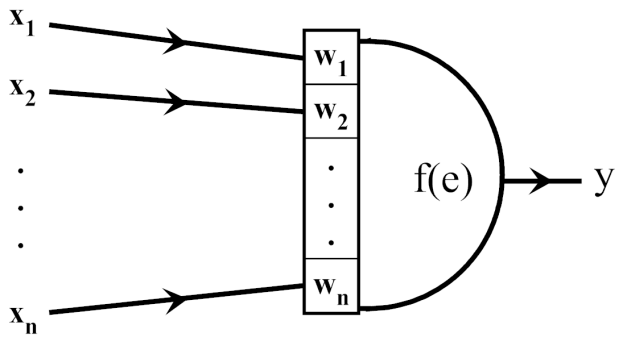
Najpowszechniejszą metodą analizy skupień jest metoda hierarchiczna. Metoda ta polega na łączeniu skupień w kolejnych krokach. Niezbędne jest zdefiniowanie miary niepodobieństwa obiektów ([4], str. 136) oraz miary niepodobieństwa skupień ([4], str. 348).

Po wybraniu stosowanych miar niepodobieństwa przystąpić możemy do realizacji algorytmu. Początkowo każdy obiekt tworzy skupienie, mamy zatem tyle skupień ile obiektów. W każdym kolejnym kroku dwa skupienia złożone z najbardziej podobnych do siebie obiektów zostają połączone. Ponieważ operujemy miarami niepodobieństwa przez najbardziej podobne do siebie obiekty rozumiemy obiekty najmniej różniące się od siebie. Z każdym krokiem liczba skupień maleje o jeden. Postępujemy tak do momentu uzyskania zadeklarowanej liczby skupień.

Zastosowanie sztucznych sieci neuronowych

Nie chcemy opisywać tutaj całej teorii sztucznych sieci neuronowych, a jedynie nieco ją przybliżyć i naszkicować sposób ich wykorzystania w zagadnieniu, jakim jest rozpoznawanie wzorców (bazując na [1], od str. 171).

Sztuczna sieć neuronowa jest zbiorem połączonych ze sobą jednostek, tak zwanych neuronów. Połączenia te są skierowane, jednostki (neurony) mają dowolną liczbę wejść orazy jedno wyjście. Każdemu połączeniu neuronów przypisana jest waga, która może ulegać zmianie w procesie uczenia sztucznej sieci neuronowej.



Rys. 1.13. Budowa sztucznego neuronu o n wejściach

Na sztuczny neuron (patrz rys. 1.13) składają się:

- Wektor wag $w = \{w_1, w_2, \ldots, w_n\}$
- Wektor sygnałów wejściowych $x = \{x_1, x_2, \ldots, x_n\}$
- e – pobudzenie neuronu, jest to suma ważona sygnałów wejściowych pomniejszona o próg Θ:

$$e = \sum_{i=1}^{n} w_i * x_i - \Theta$$

- Funkcja aktywacji $y = f(e)$. Funkcja ta na podstawie wag oraz sygnałów wejściowych określa wartość sygnału wyjściowego neuronu. Do często stosowanych funkcji pobudzenia należą
  funkcja progowa (funkcja skoku jednostkowego):

$$f(e) = \begin{cases} 1 & \text{dla } e \geq \Theta \\ 0 & \text{dla } e < \Theta \end{cases}$$

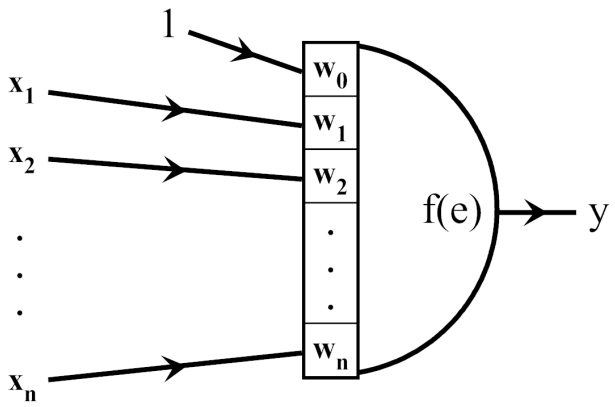
oraz funkcja sigmoidalna:

$$f(e) = \frac{1}{1 + e^{-\beta e}}$$

gdzie β to współczynnik stromości.

Dla uproszczenia zapisu często wprowadza się dodatkowy, stały sygnał $x_0 = 1$ o wadze $w_0 = \Theta$ (rys. 1.14), wówczas pobudzenie neuronu wylicza się z sumy:

$$e = \sum_{i=0}^{n} w_i * x_i$$



Rys. 1.14. Budowa sztucznego neuronu z dodatkowym wejściem o wadze $w_0 = \Theta$

Sieci neuronowe ze względu na budowę podzielić możemy na dwa typy:

- Sieci jednokierunkowe – charakteryzujące się jednym kierunkiem przepływu sygnałów
- Sieci rekurencyjne – sieci, w których sygnał wyjściowy ostatniej warstwy neuronów powraca jako sygnał wejściowy do neuronów wcześniejszych, do sieci takich należy między innymi sieć Hopfielda.

Gdy dysponujemy już siecią neuronową ustalić musimy reguły uczenia sieci. Istnieją dwa sposoby uczenia, uczenie nadzorowane oraz nienadzorowane. Przybliżymy nieco uczenie nadzorowane.

Dysponujemy zbiorem przykładów uczących, są to pary (x_i, z_i), gdzie x to wektory sygnałów wejściowych, a z to oczekiwana odpowiedź. W przypadku uczenia sieci wzorców x będzie wektorem opisującym kolejne piksele obrazu, a z klasą wzorców, do której opisywany należy.

Odpowiedź sieci dla danego wzorca (wektor x) i ustalonej funkcji aktywacji uzależniona jest od wartości wag. Uczenie sieci polega więc na dobieraniu wag tak, by zminimalizować błąd dla danego wzorca. Miarą błędu dla danego wzorca jest odległość pomiędzy odpowiedzią rzeczywistą sieci a odpowiedzią pożądaną.

"Nauczona sieć neuronowa" (pod kątem rozpoznawania wzorców) to sieć neuronowa, której wagi zostały ustalone dla każdej klasy wzorców. Dla nowego wzorca wejściowego sieć taka da odpowiedź mówiącą czy wzorzec ten należy do danej klasy wzorców czy nie.

1.2. Modelowanie informacji nieprecyzyjnej

Klasyczna teoria zbiorów bazuje na założeniu, według którego o dowolnym elemencie powiedzieć możemy, że element ten jednoznacznie należy bądź nie należy do danego zbioru. Innymi słowy w obrębie klasycznej teorii zbiorów istnieją tylko dwie odpowiedzi na pytanie czy pewien element należy do danego zbioru: tak lub nie. Podobnie jest z klasyczną logiką. Każdemu stwierdzeniu przyporządkowuje się jedną z dwu wartości logicznych: prawdę lub fałsz. Takie podejście zmusza nas do określania zjawisk oraz pojęć w sposób precyzyjny i jednoznaczny.

Często takie podejście nie pozwala na dobre modelowanie rzeczywistości. Zaproponowana przez Lofti Zadeha teoria zbiorów nieostrych (rozmytych – ang. *fuzzy sets*)([10]) i zbudowany na jej podstawie system logiki rozmytej pozwala na tworzenie matematycznych modeli rzeczywistości opartych na informacji nieprecyzyjnej. W teorii tej mówi się o częściowej przynależności elementu do zbioru oraz wykorzystuje się zmienne ligwistyczne przyjmujące nieprecyzyjne wartości języka naturalnego.

1.2.1. Zbiory nieostre

Zbiór rozmyty jest strukturą ze zdefiniowaną funkcją przynależności μ definiującą stopień w jakim rozpatrywany element należy do danego zbioru.

Definicja 1 (Zbiór rozmyty) *Zbiorem rozmytym A w uniwersum M nazywać będziemy zbiór par uporządkowanych:*

$$A = \{(x, \mu_A(x)) : x \in M\}$$

gdzie:

$$\mu_A : M \to [0, 1]$$

jest funkcją przynależności zbioru A.

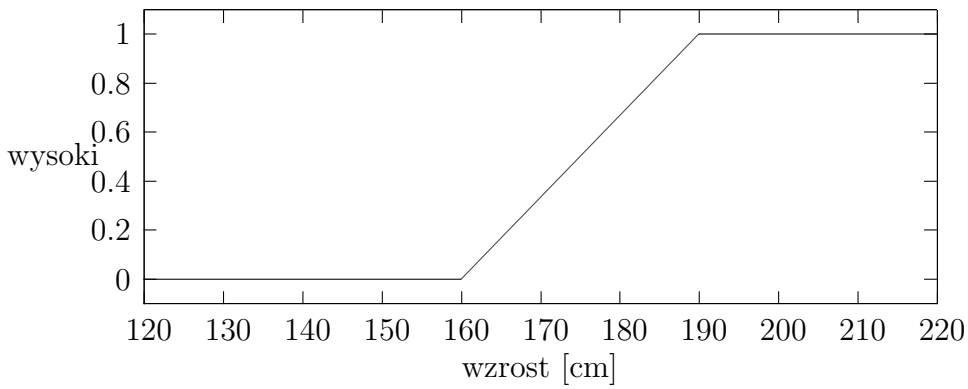
Funkcja μ_A (zwana też funkcją charakterystyczną) każdemu elementowi zbioru A przyporządkowuje stopień w jakim należy on do zbioru A. O elementach, dla których $\mu_A(x) = 0$ mówimy, że nie należą do zbioru A, w przypadku $\mu_A(x) = 1$ mówimy o pełnej przynależności do zbioru rozmytego. Natomiast elementy, dla których funkcja przynależności przyjmuje wartości pośrednie ($0 < \mu_A(x) < 1$) mówimy, że należą do zbioru w pewnym stopniu ([5] str. 23–27).

Przykład

Niech A będzie zbiorem ludzi wysokich. Zakładamy, że ludzie o wzroście do 160 cm nie są wysocy (czyli są wysocy w stopniu zerowym). Za ludzi wysokich w stopniu 1 uznamy wszystkich mających co najmniej 190 cm wzrostu. Natomiast osoby mierzące pomiędzy 160 a 190 cm są wysokie w pewnym stopniu (tym większym im są wyższe).

Przy powyższych założeniach najprostszą funkcją przynależności do zbioru ludzi wysokich będzie funkcja złożona z trzech funkcji liniowych (Rys. 1.15):

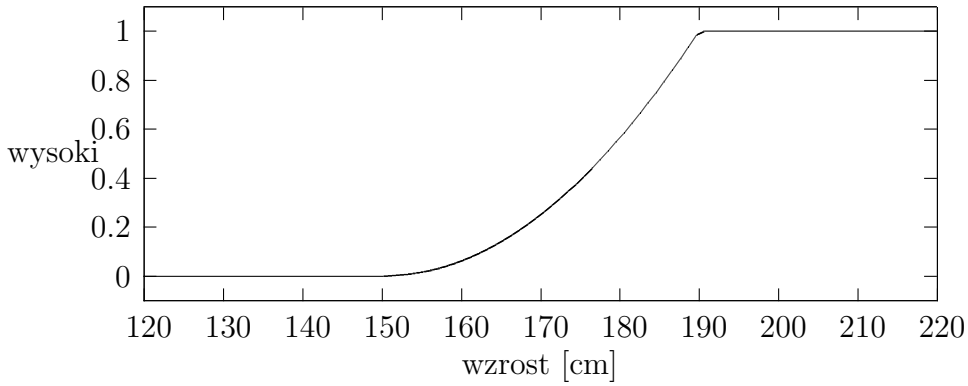
$$\mu_A(x) = \begin{cases} 0 & \text{gdy } x < 160 \\ \frac{x-160}{30} & \text{gdy } 160 \leq x \leq 190 \\ 1 & \text{gdy } x > 190 \end{cases}$$



Rys. 1.15. Przykład funkcji przynależności do zbioru "wysoki człowiek"

To, w jaki sposób definiujemy funkcję przynależności μ_A zależy wyłącznie od naszej intuicji Funkcję przynależności do zbioru ludzi "wysokich" moglibyśmy zdefiniować także następująco (Rys. 1.16):

$$\mu_A(x) = \begin{cases} 0 & \text{gdy } x < 150 \\ (\frac{x-150}{40})^2 & \text{gdy } 150 \leq x \leq 190 \\ 1 & \text{gdy } x > 190 \end{cases}$$



Rys. 1.16. Przykład funkcji przynależności do zbioru "wysoki człowiek" o parabolicznym tempie wzrostu

Własności

Obszar, dla którego funkcja przynależności przyjmuje wartość 1 nazywamy rdzeniem (ang. *core*). Obszar, dla którego funkcja przynależności ma wartość większą od 0 nazywamy nośnikiem (ang. *support*).

$$\begin{aligned} core(A) &= \{x \in M \mid \mu_A(x) = 1\} \ , \\ support(\mu) &= \{x \in M \mid \mu_A(x) > 0\} \ . \end{aligned}$$

Operacje na zbiorach rozmytych

Podobnie jak w klasycznej teorii zbiorów, dla zbiorów rozmytych wprowadzone zostały operacje sumy, przecięcia oraz dopełnienia. Aby je przedstawić niezbędne jest zdefiniowanie pojęć t-normy, t-konormy i negacji:

Definicja 2 (T-norma) *Funkcję dwóch zmiennych t, taką że $t : [0,1] \times [0,1] \to [0,1]$, nazywamy T-normą, jeżeli spełnia poniższe warunki:*

1. $t(a,b) = t(b,a)$ *(warunek przemienności)*
2. $a \leq c \wedge b \leq d \Rightarrow t(a,b) \leq t(c,d)$ *(warunek monotoniczności)*
3. $t(t(a,b),c) = t(a,t(b,c))$ *(warunek łączności)*
4. $t(a,1) = a$, *(element neutralny = 1)*

$\forall a,b,c,d \in [0,1]$

Definicja 3 (T-konorma) *Funkcję dwóch zmiennych s, taką że $s : [0,1] \times [0,1] \to [0,1]$, nazywamy T-konormą, jeżeli spełnia 3 pierwsze warunki definicji T-normy (jest przemienna, monotoniczna i łączna) oraz 0 jest jej elementem neutralnym:*

$$s(a,0) = a,$$

$\forall a \in [0,1]$

Tabela 1.1 definiuje najczęściej używane T-normy i T-konormy.

Definicja 4 (Negacja) *Każdą nierosnącą funkcję $\nu : [0,1] \to [0,1]$, taką że: $\nu(0) = 1$ i $\nu(1) = 0$ nazywamy negacją.*

Przykład:

$$\nu_{Ł}(a) := 1 - a \text{ (negacja Łukasiewicza)}$$

T-normy i T-konormy zostały wyczerpująco opisane w [5] str. 2 – 14.

Przytoczone powyżej T-normy i negacje to narzędzia, dzięki którym można zdefiniować podstawowe operacje na zbiorach rozmytych. Niech s, t i ν

T-norma	T-konorma
minimum $t_{min}(a,b) := min\{a,b\}$	maksimum $s_{max}(a,b) := max\{a,b\}$
algebraiczna $t_a(a,b) := ab$	algebraiczna $s_a(a,b) := a+b-ab$
Łukasiewicza $t(a,b) := max\{0, a+b-1\}$	Łukasiewicza $s(a,b) := min\{a+b, 1\}$
drastyczna $t_D(a,b) := \begin{cases} b & \text{jeżeli } a=1 \\ a & \text{jeżeli } b=1 \\ 0 & \text{w p. p.} \end{cases}$	drastyczna $s_D(a,b) = \begin{cases} b & \text{jeżeli } a=0 \\ a & \text{jeżeli } b=0 \\ 1 & \text{w p. p.} \end{cases}$

Tab. 1.1. Podstawowe T-normy i T-konormy

oznaczają odpowiednio T-normę, T-konormę i negację oraz A,B to dowolne zbiory rozmyte w uniwersum M([5] str. 27).

Definicja 5 (Suma zbiorów rozmytych) *Sumą zbiorów rozmytych A i B indukowaną przez s jest zbiór rozmyty $A \cup_s B$, taki że:*

$$\forall_{x \in M} : (A \cup_s B)(x) := A(x) s B(x)$$

Definicja 6 (Przekrój zbiorów rozmytych) *Przekrojem zbiorów rozmytych A i B indukowanym przez t jest zbiór rozmyty $A \cap_t B$, taki że:*

$$\forall_{x \in M} : (A \cap_t B)(x) := A(x) t B(x)$$

Definicja 7 (Dopełnienie zbioru rozmytego) *Dopełnieniem zbioru rozmytego A indukowanym przez ν jest zbiór rozmyty A^ν, taki że:*

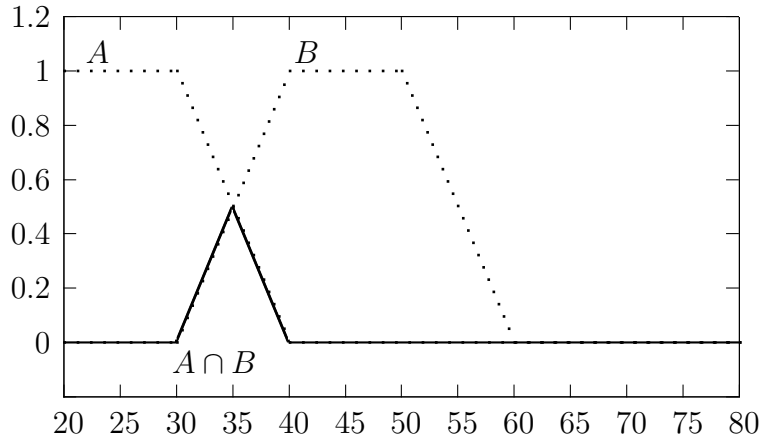
$$\forall_{x \in M} : A^\nu(x) := \nu(A(x))$$

Dodatkowo dla zbiorów rozmytych A_1 i A_2 w uniwersach, odpowiednio M_1 i M_2 oraz t-normy t wprowadźmy definicję iloczynu kartezjańskiego:

Definicja 8 (Iloczyn kartezjański zbiorów rozmytych) *Iloczynem kartezjańskim zbiorów rozmytych A_1 i A_2 indukowanym przez t jest zbiór rozmyty $A_1 \times_t A_2$, taki że:*

$$\forall_{\substack{x \in M_1 \\ y \in M_2}} : A_1 \times_t A_2(x, y) := A_1(x) t A_2(y)$$

Na rys. 1.17 pogrubioną linia oznacznony jest przekrój zbiorów rozmytych A i B.



Rys. 1.17. Przekrój zbiorów A i B przy wykorzystaniu T-normy minimum

Zmienna lingwistyczna

Aby opisać zmienną rzeczywistą przy pomocy nieprecyzyjnych terminów często potrzeba dwóch lub więcej zbiorów rozmytych. Zmienna lingwistyczna (Def. 9) jest strukturą definiującą powiązania zbiorów rozmytych z lingwistycznymi opowiednikami.

Definicja 9 (Zmienna lingwistyczna ([6])) *Zmienną lingwistyczną nazywamy czwórkę* (N, T, X, I)*, gdzie:*

- N — *to nazwa zmiennej lingwistycznej,*
- T — *zbiór wartości lingwistycznych, zwanych terminami,*
- X — *uniwersum, przestrzeń rozważań,*
- I — *zbiór interpretacji terminów.*

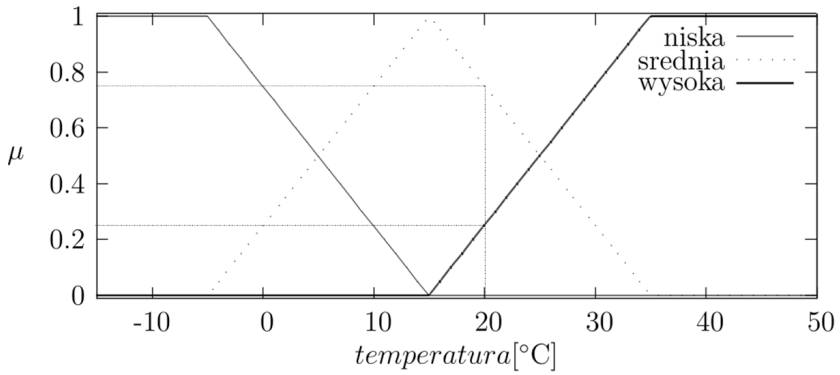
Interpretacje są zbiorami nieostrymi w X. Każdemu terminowi odpowiada dokładnie jedna interpretacja.

Przykład

Rozważmy zmienną lingwistyczną *temperatura* o wartościach ligwistycznych: niska, średnia, wysoka w przestrzeni rozważań $[-14°\mathrm{C}, 50°\mathrm{C}]$ o interpretacji terminów jak na Rys. 1.18. Temperatura 20°C jest w zerowym stopniu niska, w stopniu 0.78 średnia i 0.22 wysoka.

1.2.2. Wnioskowanie rozmyte – systemy regułowe

W realnym świecie procesy wnioskowania często odbiegają od schematu klasycznej logiki. Wynika to z faktu, że przesłanki, którymi się posługujemy nie są do końca pewne, w związku z czym otrzymujemy wnioski uznawane tylko w pewnym stopniu. Sterownik rozmyty to aparat, który bazując na takich przesłankach oraz w oparciu o zestaw reguł rozmytych podejmuje decyzje. W klasycznej teorii logiki wnioskowanie przeprowadzane jest przy wykorzystaniu reguły *modus ponens* (łac. *modus ponendo ponens* – sposób



Rys. 1.18. Przykładowe funkcje przynależności ilustrujące zmienną lingwistyczną *temperatura*

potwierdzający przez potwierdzenie). Schemat takiego wnioskowania można przedstawić w następujący sposób:

Przesłanka:	A
Implikacja:	IF A THEN B
Wniosek:	B

gdzie:

A, B – zdania logiczne.

Natomiast w sterownikach rozmytych stosuje się rozmytą regułę *modus ponens*:

Przesłanka:	x jest A'
Implikacja:	IF x jest A THEN y jest B
Wniosek:	y jest B'

gdzie:

x i y – zmienne lingwistyczne,

A, A', B, B' – zbiory rozmyte opisujące te zmienne.

Warto zauważyć, że zbiory A' i B' stanowią pewne odstępstwo od zbiorów odpowiednio A i B, B' zależy od stopnia spełnienia przesłanek. Daje nam to pewną elastyczność, którą można zobrazować przy pomocy przykładu: Mając regułę *Jeśli prędkość samochodu jest duża, to poziom hałasu jest wysoki* oraz posługując się przesłanką *Prędkość samochodu jest średnia* można wywnioskować, że *Poziom hałasu będzie średni.*

W regułach rozmytych dopuszczalne jest używanie operatorów:

- AND – operator koniunkcji, podczas obliczeń zastępowany wybraną t - normą,
- OR – operator alternatywy, podczas obliczeń stosuje się jedną z t - konorm,
- NOT – operator negacji.

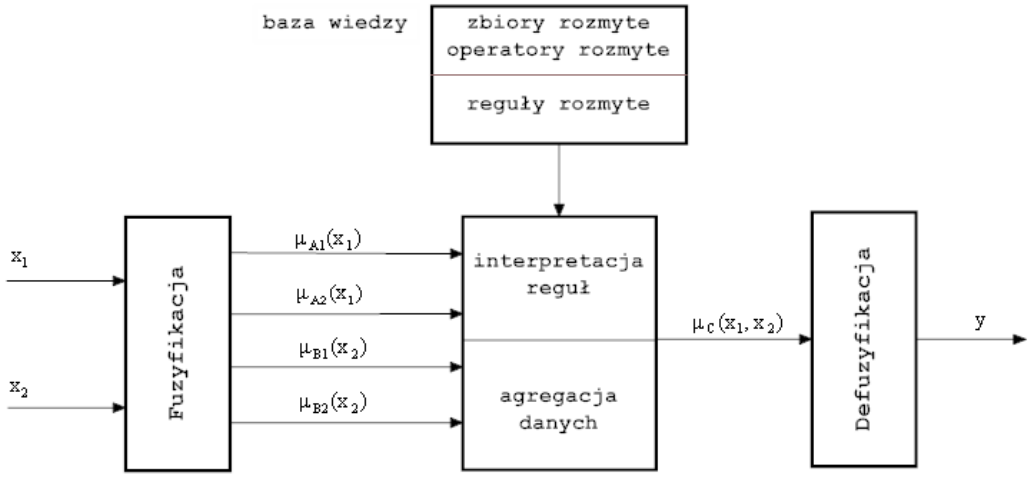
Sterownik rozmyty Mamdaniego

Sterownik Mamdaniego to system regułowy, który dla pewnego wektora danych wejściowych wyznacza wektor danych wyjściowych wykorzystując zdefiniowane wcześniej zmienne lingwistyczne oraz bazę reguł rozmytych. Schemat przykładowego sterownika znajduje się na Rys. 1.19.

Sterownik rozmyty składa się następujących elementów:

Dane wejściowe – wejście sterownika rozmytego stanowi zestaw pewnych zmiennych rzeczywistych (np. wyniki pomiaru parametrów wymaganych do sterowania czyli podjęcia decyzji lub wyznaczenia wartości wyjściowej bazującej na tych danych).

Blok rozmywania – zmienne wektora wejściowego x poddawane są operacji rozmywania (ang. *fuzzyfication*), w wyniku której odwzorowywane



Rys. 1.19. Schemat przykładowego sterownika rozmytego

są one na zbiory rozmyte (poszczególnym zmiennym przypisane zostają wartości funkcji przynależności μ do zbiorów rozmytych zmiennej lingwistycznej opisującej daną zmienną wejściową).

Blok wnioskowania – na podstawie stopni przynależności wektora danych wejściowych oraz w opraciu o bazę wiedzy wyznaczona zostaje wyjściowa funkcja przynależności. Baza wiedzy to zbiór reguł rozmytych opisanych na zbiorach rozmytych, zdefiniowanych w oparciu o wiedzę ekspercką, w postaci:

(1) IF x_1 IS A_1 AND x_2 IS B_1 THEN y IS C_1
(2) IF x_1 IS A_2 OR x_2 IS B_2 THEN y IS C_2
(3) IF x_1 IS A_1 THEN y IS C_3

Dla każdej reguły w bazie wyznaczamy stopień spełnienia jej przesłanek przy użyciu wybranych t-norm oraz negacji. Następuje to poprzez obliczenie wartości przesłanki P_i reguły z uwzględnieniem operatorów OR, AND i NOT. Kolejnym krokiem jest wyznaczenie zbioru rozmytego C_i',

przy pomocy operatora implikacji inżynierskiej $\rightarrow$ ([12]):

$$C_i^{'}(y) = P_i(x) \rightarrow C_i(y) = P_i(x) \times_t C_i(y)$$

gdzie:

- t – wybrana t-norma (w przypadku sterownika Mamdaniego stosuje się zwykle t-normę minimum),
- i – numer reguły, $i = 1, 2, \ldots, n$.

Wyznaczamy wynikowy zbiór rozmyty C sumując $C_1, \ldots, C_n$:

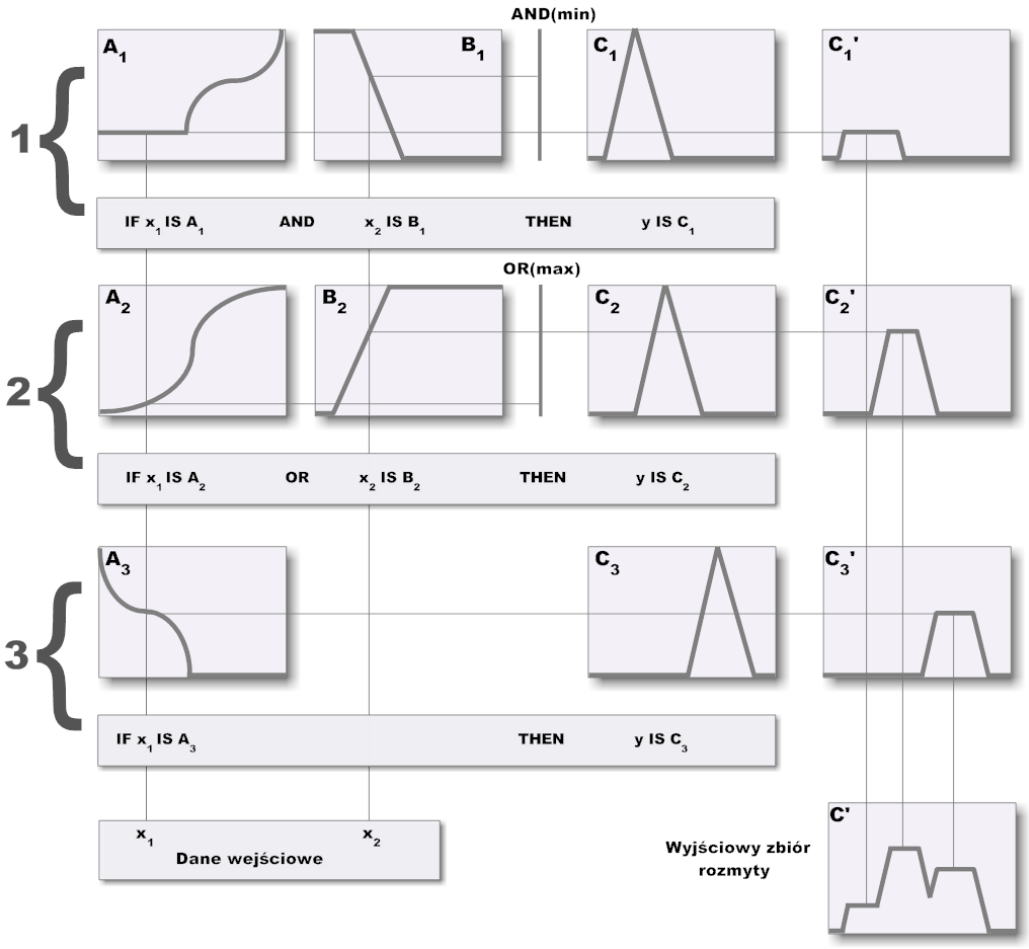
$$C = C_1 \cup_s \ldots \cup_s C_n$$

gdzie:

- s - wybrana t-konorma (w przypadku sterownika Mamdaniego stosuje się zwykle t-konorme maksimum).

Na Rys. 1.20 przedstawiono schemat wnioskowania opartego na trzech przykładowych regułach. Zastosowano operacje minimum i maksimum jako, odpowiednio, t-normę i t-konormę.

Blok wyostrzania – na podstwie zbioru rozmytego będącego wnioskiem z danych wejściowych następuje defuzyfikacja (ang. *defuzzyfication* - wyostrzenie) i wyznaczenie wektora danych wyjściowych. Jest wiele sposobów wyliczania tych wartości, najbardziej popularną jest metoda środka ciężkości (*COG, Center of Gravity*). Jako wynik y przyjmowana jest współrzędna środka ciężkości powierzchni pod krzywą będącą funkcją



Rys. 1.20. Graficzna interpretacja wnioskowania w oparciu o reguły rozmyte ([7]).

przynależności μ_C wynikowego zbioru rozmytego:

$$y = \frac{\sum_{x \in M} x \mu_C(x)}{\sum_{x \in M} \mu_C(x)}$$

Dane wyjściowe – wynikiem działania sterownika rozmytego jest zmienna rzeczywista y – wynik działania reguł rozmytych dla danych wejściowych.

1.2.3. Relacje nieostre

Definicja 10 (Relacja rozmyta) *Relacją rozmytą R między dwoma zbiorami X i Y nazywamy zbiór rozmyty określony na iloczynie kartezjańskim $X \times Y$:*

$$R = \{((x, y), \mu_R(x, y)) : x \in X, y \in Y\}$$

gdzie:

$$\mu_R : X \times Y \to [0, 1]$$

jest funkcją przynależności zbioru A.

Funkcja μ_r przyporządkowuje parze (x, y) $x \in X, y \in Y$ jej stopień przynależności, który można interpretować jako siłę powiązania między elementami x i y.

Przykład relacji rozmytej

Dla danych przestrzeni: $X = \{x_1, x_2, x_3\} = \{1, 2, 3\}$, $Y = \{y_1, y_2, y_3\} = \{2, 3, 4\}$. Relację $R \subset X \times Y$ określoną jako "x jest mniej więcej równe y" można zdefiniować w następujący sposób:

$$\mu_R(x) = \begin{cases} 1 & \text{gdy } |x - y| = 0 \\ 0.6 & \text{gdy } |x - y| = 1 \\ 0.3 & \text{gdy } |x - y| = 2 \\ 0 & \text{gdy } |x - y| = 3 \end{cases}$$

Definicja 11 (Typy relacji nieostrych) *Niech t – t - norma. Mówimy, że relacja nieostra $R : M \times M \to [0,1]$ jest:*

- *zwrotna, gdy:* $\forall_{x \in M} R(x,x) = 1$,
- *antyzwrotna, gdy:* $\forall_{x \in M} R(x,x) = 0$,
- *symetryczna, gdy:* $\forall_{x,y \in M} R(x,y) = R(y,x)$,
- *antysymetryczna, gdy:* $\forall_{x,y \in M}(R(x,y) > 0 \wedge R(y,x) > 0) \Rightarrow x = y$,
- *t–przechodnia, gdy:* $\forall_{x,y,z \in M} R(x,z) \geq R(x,y)tR(y,z)$,
- *spójna, gdy:* $\forall_{x,y \in M} R(x,y) > 0 \vee R(y,x) > 0$.

Definicja 12 (Nieostra relacja podobieństwa) *Niech t – dowolna t - norma, $R : M \times M \to [0,1]$*
Jeżeli R jest zwrotna, symetryczna i t–przechodnia to nazywamy ją nieostrą relacją podobieństwa.

Przykład

Niech $M = \{x_1, x_2, x_3\}$, $R : M \times M \to [0,1]$ jest zdefiniowana w postaci tabeli, gdzie kolumny i wiersze odpowiadają kolejnym elementom M:

$$R = \begin{bmatrix} 1 & 0.2 & 1 \\ 0.2 & 1 & 0.2 \\ 1 & 0.2 & 1 \end{bmatrix}$$

Z łatwością można stwierdzić, że R jest zwrotna, symetryczna i t-przechodnia (dla t-normy minimum). Liczba $R(x_i, x_j)$ określa w jakim stopniu elementy x_i i x_j są podobne. Takie relacje nieostre nazywane są miarami podobieństwa([12]).

1.2.4. Miara podobieństwa

Znalezienie odpowiedzi na pytanie w jakim stopniu dwa elementy są do siebie podobne lub bliskie sobie wymaga pewnego usystematyzowania przestrzeni rozważań. Należy zdefiniować na tej przestrzeni pewną metrykę, która określa odległość pomiędzy parami elementów.

Definicja 13 (Metryka) *Metryką na zbiorze X nazywamy funkcję $d : X \times X \to \boldsymbol{R}$ spełniającą warunki:*

- $\forall_{x,y \in X}\, d(x,y) = 0 \Leftrightarrow x = y$,
- $\forall_{x,y \in X}\, d(x,y) = d(y,x)$,
- $\forall_{x,y,z \in X}\, d(x,z) \leq d(x,y) + d(y,z)$.

Parę (X,d) nazywamy przestrzenią metryczną.

Przykłady

Niech $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n)$ należą do przestrzeni $\boldsymbol{R}^n$

- **Metryka euklidesowa**

$$d_e(x,y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$$

- **Metryka Manhattan** - odległość w tej metryce to suma wartości bezwzględnych różnic poszczególnych współrzędnych:

$$d_m(x,y) = \sum_{k=1}^{n} |x_k - y_k|$$

- **Metryka dyskretna (Hamminga)** - odległość pomiędzy dwoma punktami wynosi 0, gdy są to te same punkty, 1 w przeciwnym przypadku:

$$d_d(x,y) = \begin{cases} 0, & \text{gdy } \forall_{k=1\ldots n}\, x_k = y_k \\ 1, & \text{w przeciwnym przypadku} \end{cases}$$

Rozdział 2

System detekcji i rozpoznawania znaków drogowych

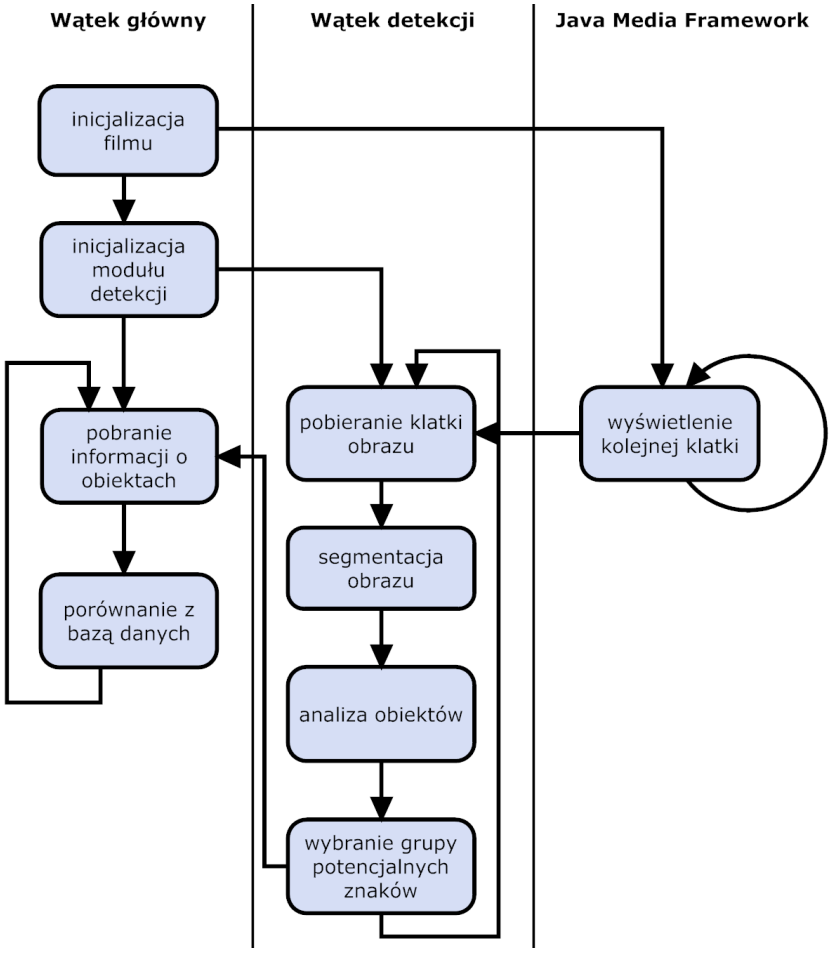
2.1. Architektura systemu

Zaprojektowany i wykonany przez nas *System detekcji i rozpoznawania znaków drogowych* składa się z trzech kluczowych elementów:

Baza wiedzy – przechowuje dane o znakach i skojarzonych z nimi reprezentacjach, pozwala na szybkie znalezienie reprezentacji najbardziej podobnych do otrzymanego na wejściu zakodowanego wzorca (i tym samym znaków do niego najbardziej podobnych).

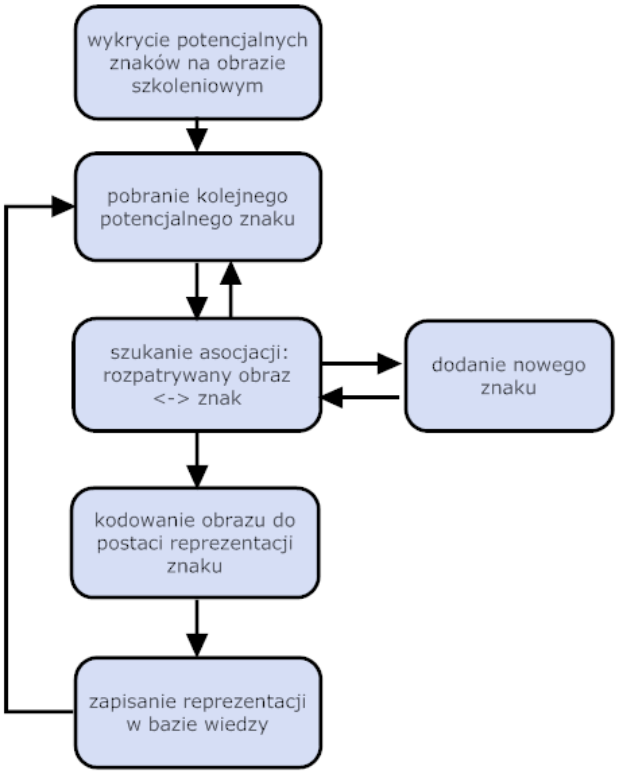
Moduł uczenia – umożliwia stworzenie i rozszerzanie bazy wiedzy systemu poprzez dodawanie powiązań: *fragment obrazu pobrany z materiału szkoleniowego* $\leftrightarrow$ *znak* (Rys. 2.2).

Moduł rozpoznawania znaków – jest odpowiedzialny za przetwarzanie obrazu wejściowego w taki sposób, aby zidentyfikować na nim obszary, które mogą zawierać znaki drogowe, wybrać najlepsze z nich, porównać je z danymi w bazie wiedzy oraz zaprezentować rezultat użytkownikowi (Rys. 2.1).



Rys. 2.1. Uproszczony schemat działania **Modułu rozpoznawania znaków**

Działanie powyższych podsystemów zostanie szczegółowo opisane w kolejnych podrozdziałach.



Rys. 2.2. Uproszczony schemat działania **Modułu uczenia**

2.2. Segmentacja obrazu wejściowego

Wątek zajmujący się detekcją znaków drogowych wysyła do wątku głównego żądanie informujące, że jest gotowy do analizy i oczekuje na kolejną klatkę obrazu.

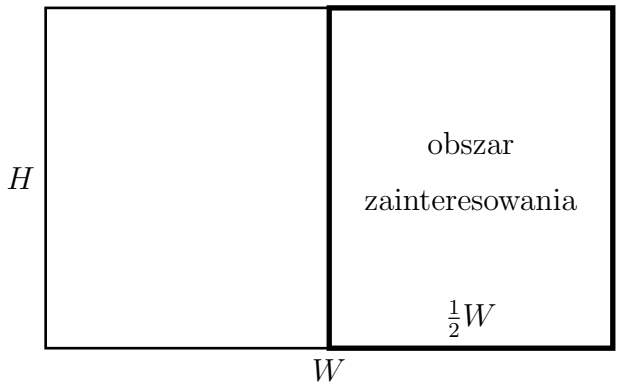
Zadaniem tego wątku jest nie tyle precyzyjne wykrycie znaków na obrazie, co wyznaczenie obszarów, w których mogą one występować, poddanie ich analizie, odrzucanie obszarów nie spełniających wyznaczonych kryteriów, ewentualne grupowanie kilku obszarów w jeden większy, wreszcie zakodowanie informacji o potencjalnych znakach i przekazanie ich do wątku głównego.

Pierwszym etapem jest segmentacja obrazu wejściowego. Wynik segmentacji to obraz binarny, złożony z białych i czarnych pikseli. Piksele czarne

(wygaszone) to te, które w procesie segmentacji uznane zostały za tło. Piksele białe to potencjalne elementy znaków drogowych. Aby piksel uznany został za element znaku drogowego jego barwa należeć musi do jednej z klas barw: czerwonej, żółtej lub niebieskiej.

2.2.1. Obszar segmentacji

Zakładamy, że kamera, z której obraz analizujemy, rejestruje obszar znajdujący się przed pojazdem, zbliżony do obszaru widzianego przez pasażera. Dlatego nie będziemy szukać znaków drogowych w obrębie całego obrazu, interesują nas tylko znaki widniejące po prawej stronie jezdni. Na początku definiujemy więc obszar zainteresowania. Będzie to prawa połowa obrazu wejściowego (rys. 2.3, strona 50).



Rys. 2.3. Obszar obrazu, w którym "szukane są" znaki drogowe

2.2.2. Zastosowany model barw

Początkowo stosowaliśmy model przestrzeni barw RGB (patrz "model RGB" na stronie 17). Wyniki segmentacji w oparciu o ten model barw były zadowalające dla barw czerwonej oraz niebieskiej. Jego stosowanie wiązało się

jednak z dwiema istotnymi wadami:

- Model RGB nie jest intuicyjny, nie opisuje barw w sposób łatwo zrozumiały dla człowieka.
- Wymodelowanie funkcji przynależności definiujących zbiory rozmyte opisujące poszczególne barwy jest bardzo skomplikowane gdy stosuje się model RGB.

Dlatego aby lepiej opisać klasy barw zmieniliśmy model przestrzeni barw stosowany w procesie segmentacji na HSB (patrz "model HSB" na stronie 20).

2.2.3. System regułowy

Badany jest każdy piksel z obszaru zainteresowania (rys. 2.3). Zdefiniowaliśmy funkcje przynależności opisujące zbiory rozmyte "żółty", "niebieski" oraz "czerwony". Argumentem tej funkcji jest para składowych H i S. Jeżeli badany piksel należy w wystarczającym stopniu do jednego z tych zbiorów (jest wystarczająco żółty, niebieski lub czerwony), zostaje uznany za fragment znaku drogowego i przypisuje mu się wartość 1. W przeciwnym razie, jako elementowi tła, przypisujemy mu wartość 0.

Zdefiniowanie barw "żółta", "niebieska" i "czerwona" w modelu HSB

Przedstawiony poniżej model opisu konkretnych barw w przestrzeni barw HSB zaczerpnęliśmy z książki *"Fuzzy Techniques in Image Processing"* ([8], str. 267 – 285).

Barwami współtworzącymi znaki drogowe są także barwy biała i czarna. Nie bierzemy ich jednak pod uwagę na etapie segmentacji, ponieważ zbyt wie-

le elementów tła (nie będących znakami drogowymi) system przypisywałby do tych barw. Segmentacja realizowana jest przez system regułowy, który na podstawie wartości składowych modelu barw HSB wyznacza stopień, w jakim badany piksel przynależy do znaku drogowego. Aby zbudować taki system regułowy przeprowadzić musieliśmy serię testów i analiz. Napisaliśmy w tym celu aplikację testową, która:

- pozwala na przeglądanie zdjęć – kadrów z nagrania wideo,
- pozwala na dynamiczne dobieranie wartości modelujących barwy żółtą, niebieską oraz czerwoną,
- podaje wartości składowe oraz stopień przynależności do zdefiniowanych barw dla wskazanego piksela,
- wyświetla obraz binarny będący efektem segmentacji dla ustalonych wartości,
- wyświetla obraz – efekt segmentacji, na którym każdemu pikselowi przypisana jest jedna z barw:

czarna – piksel uznany za element tła,

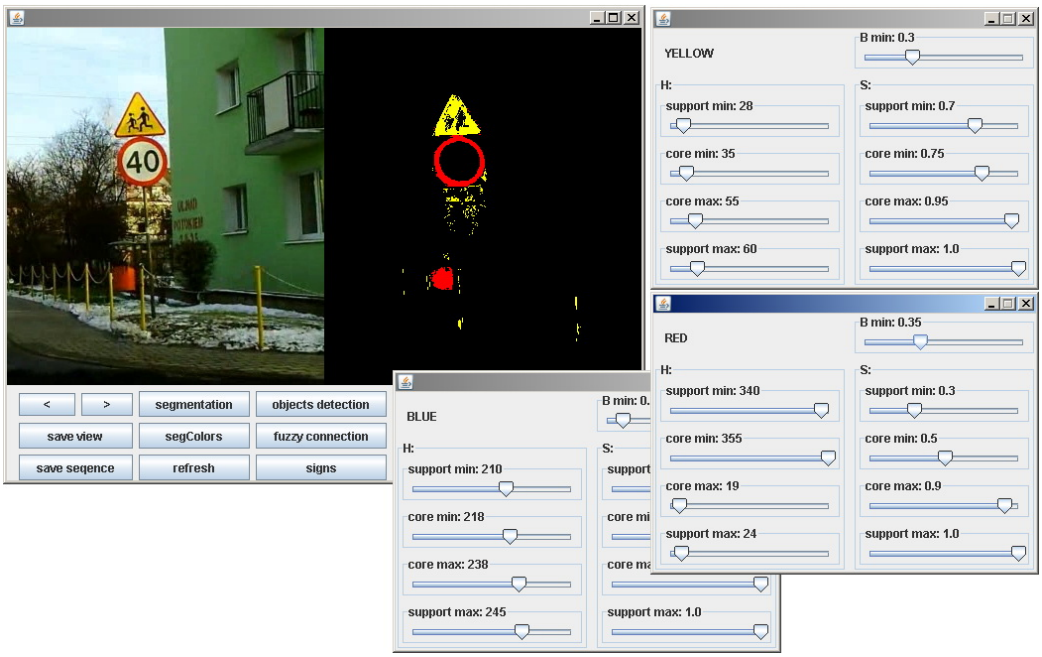
żółta – piksel uznany za fragment znaku o żółtej barwie,

niebieska – piksel uznany za fragment znaku o niebieskiej barwie,

czerowna – piksel uznany za fragment znaku o czerwonej barwie.

Aplikacja ta wraz z przykładowym efektem segmentacji dla podanych parametrów prezentowana jest na rysunku 2.4.

Korzystając z wspomnianej aplikacji przeanalizowaliśmy kilkaset kadrów z nagrań wideo. Pozwoliło nam to dobrać odpowiednie wartości i wymodelować funkcje przynależności do poszczególnych barw.



Rys. 2.4. Aplikacja testująca, dobór parametrów segmentacji

Badanie stopnia przynależności do zbioru "żółty"

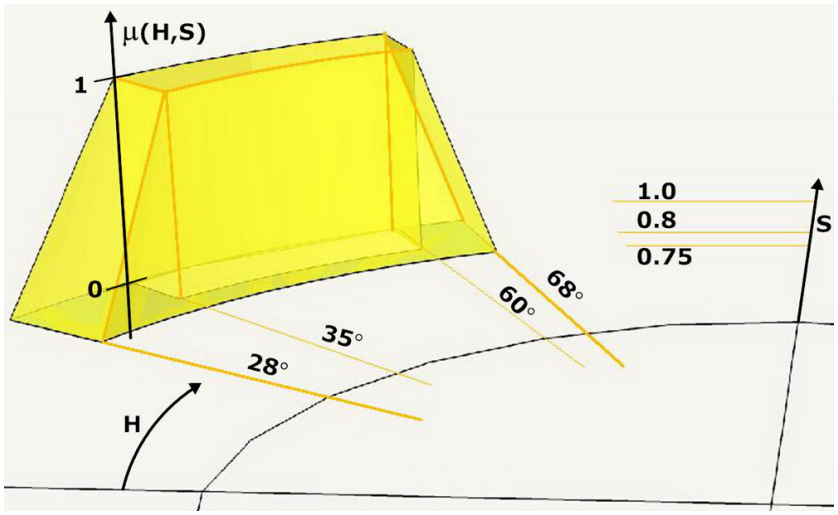
Najpierw badamy wartość składowej B (Brightness):
Jeżeli $B \notin (0.4, 1.0)$ to badany piksel nie należy do zbioru "żółty" (stopień przynależności wynosi 0.0). W przeciwnym razie stopień przynależności do barwy żółtej wyznacza funkcja przynależności (rys. 2.5, str. 54) na podstawie składowych H i S.

Niech U będzie *uniwersum*, zbiorem wszystkich par wartości składowych H i S:

$$U = H \times S = [0° \ldots 360°] \times [0.0 \ldots 1.0]$$

Dla zbioru "żółty" mamy:

$$\begin{aligned} core(\mu_{yellow}) &= \{(h,s) \mid h \in [35°, 60°] \wedge s \in [0.8, 1.0]\} \\ support(\mu_{yellow}) &= \{(h,s) \mid h \in [28°, 68°] \wedge s \in [0.75, 1.0]\} \end{aligned}$$



Rys. 2.5. Funkcja przynależności do zbioru "żółty"

Poniżej wzory pozwalające bezpośrednio wyznaczyć wartość funkcji przynależności do barwy żółtej:

$$\mu_{yellow}(h,s) = 1.0 \quad \text{dla} \quad h \in [35°, 60°] \wedge s \in [0.8, 1.0] \ ,$$

$$\mu_{yellow}(h,s) = \frac{h-28°}{35°-28°} \quad \text{dla} \quad h \in [28°, 35°] \wedge s \geq 0.75 + (0.8-0.75)\frac{h-28°}{35°-28°} \ ,$$

$$\mu_{yellow}(h,s) = \frac{68°-h}{68°-60°} \quad \text{dla} \quad h \in [60°, 68°] \wedge s \geq 0.75 + (0.8-0.75)\frac{68°-h}{68°-60°} \ ,$$

$$\mu_{yellow}(h,s) = \frac{s-0.75}{0.8-0.75} \quad \text{dla} \quad s \in [0.75, 0.8] \wedge h \geq 28° + (35°-28°)\frac{s-0.75}{0.8-0.75} \wedge$$
$$h \leq 68° - (68°-60°)\frac{s-0.75}{0.8-0.75} \ ,$$

$$\mu_{yellow}(h,s) = 0.0 \quad \text{dla} \quad \text{pozostałych.}$$

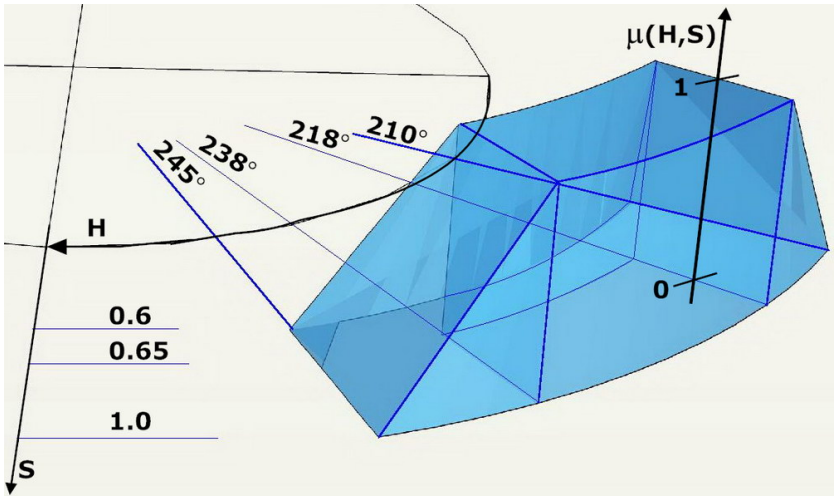
Badanie stopnia przynależności do zbioru "niebieski"

Najpierw badamy wartość składowej B (Brightness):

Jeżeli $B \notin (0.1, 1.0)$ to badany piksel nie należy do zbioru "niebieski" (należy do zbioru niebieski w stopniu 0.0).

Jeżeli jednak składowa $B \in (0.1, 1.0)$, to stopień przynależności do barwy niebieskiej wyznacza funkcja przynależności (rys. 2.6, str. 55) na podstawie składowych H i S.

$$\begin{aligned} core(\mu_{blue}) &= \{(h,s) \mid h \in [218°, 238°] \wedge s \in [0.65, 1.0]\} \\ support(\mu_{blue}) &= \{(h,s) \mid h \in [210°, 245°] \wedge s \in [0.6, 1.0]\} \end{aligned}$$



Rys. 2.6. Funkcja przynależności do zbioru "niebieski"

Stopień przynależności wyznaczamy z wzorów:

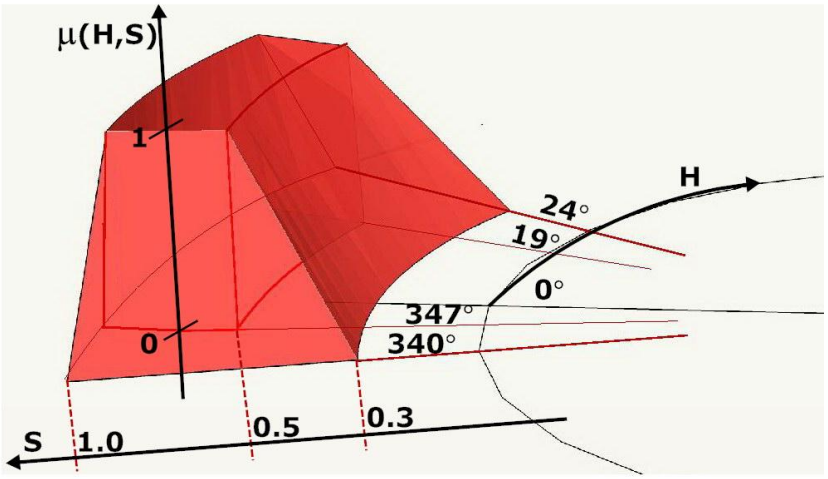
$$\begin{aligned}
&\mu_{blue}(h,s) = 1.0 && \text{dla} \quad h \in [218°, 238°] \wedge s \in [0.65, 1.0] \ , \\
&\mu_{blue}(h,s) = \tfrac{h-210°}{218°-210°} && \text{dla} \quad h \in [210°, 218°] \wedge s \geq 0.6 + (0.65 - 0.6)\tfrac{h-210°}{218°-210°} \ , \\
&\mu_{blue}(h,s) = \tfrac{245°-h}{245°-238°} && \text{dla} \quad h \in [238°, 245°] \wedge s \geq 0.6 + (0.65 - 0.6)\tfrac{245°-h}{245°-238°} \ , \\
&\mu_{blue}(h,s) = \tfrac{s-0.6}{0.65-0.6} && \text{dla} \quad s \in [0.6, 0.65] \wedge h \geq 210° + (218° - 210°)\tfrac{s-0.6}{0.65-0.6} \wedge \\
& && \quad\ h \leq 245° - (245° - 238°)\tfrac{s-0.6}{0.65-0.6} \ , \\
&\mu_{blue}(h,s) = 0.0 && \text{dla} \quad \text{pozostałych.}
\end{aligned}$$

Badanie stopnia przynależności do zbioru "czerwony"

Najpierw badamy wartość składowej B (Brightness):
Jeżeli $B \notin (0.15, 1.0)$ to badany piksel nalezy do zbioru "czerwony" w stopniu 0.0.

Jeśli składowa $B \in (0.15, 1.0)$ – stopień przynależności piksela do barwy czerwonej wyznacza funkcja przynależności (rys. 2.7, str. 56) na podstawie składowych H i S.

$$\begin{aligned} core(\mu_{red}) &= \{(h,s) \mid h \in [-13°, 19°] \wedge s \in [0.5, 1.0]\} \\ support(\mu_{red}) &= \{(h,s) \mid h \in [-20°, 24°] \wedge s \in [0.3, 1.0]\} \end{aligned}$$



Rys. 2.7. Funkcja przynależności do zbioru "czerwony"

Stopień przynależności wyznaczamy z wzorów:

$$
\begin{aligned}
&\mu_{red}(h,s) = 1.0 && \text{dla} \quad (h \geq -347^\circ \vee h \leq 19^\circ) \wedge s \in [0.5, 1.0] \ , \\
&\mu_{red}(h,s) = \tfrac{h-340^\circ}{347^\circ-340^\circ} && \text{dla} \quad h \in [340^\circ, 347^\circ] \wedge s \geq 0.3 + (0.5-0.3)\tfrac{h-340^\circ}{347^\circ-340^\circ} \ , \\
&\mu_{red}(h,s) = \tfrac{24^\circ-h}{24^\circ-19^\circ} && \text{dla} \quad h \in [19^\circ, 24^\circ] \wedge s \geq 0.3 + (0.5-0.3)\tfrac{24^\circ-h}{24^\circ-19^\circ} \ , \\
&\mu_{red}(h,s) = \tfrac{s-0.3}{0.5-0.3} && \text{dla} \quad s \in [0.3, 0.5] \wedge h \geq 340^\circ + (347^\circ-340^\circ)\tfrac{s-0.3}{0.5-0.3} \wedge \\
& && \qquad h \leq 24^\circ - (24^\circ-19^\circ)\tfrac{s-0.3}{0.5-0.3} \ , \\
&\mu_{red}(h,s) = 0.0 && \text{dla} \quad \text{pozostałych.}
\end{aligned}
$$

Podejmowanie decyzji

Mając zdefiniowane funkcje przynależności jesteśmy w stanie ocenić przynależność dowolnego piksela do poszczególnych barw typowych dla znaków drogowych na podstawie jego składowych modelu przestrzeni barw HSB.

O tym czy badany piksel x zostanie uznany za fragment znaku drogowego decyzuje oczywisty warunek (stosujemy T-normę minimum i T-conormę maksimum):

- IF x IS YELLOW OR x IS BLUE OR x IS RED THEN x IS SIGN

Na rysunku 2.8 widzimy przykład działania segmentacji. Pierwszy obraz od lewej przedstawia obszar zainteresowania wycięty z kadru nagrania wideo. Drugi prezentuje wyniki segmentacji, w której każdy piksel został przypisany do jednej z czterech klas: tło, barwa zółta, barwa niebieska oraz barwa czerwona. Trzeci to obraz binarny dzielący piksele na tło i fragmenty znaków drogowych, takim obrazem posługujemy się do dalszej analizy. Jak widzimy także fragmenty tła uznane zostają za fragmenty znaków drogowych. Tutaj są to fragmenty światła samochodu (czerwone) oraz mały fragment elewacji budynku uznany za żółte elementy znaku drogowego.

Rys. 2.8. Przykład działania segmentacji

2.3. Analizowanie obrazu binarnego – lokalizowanie znaków

Na tym etapie dysponujemy już obrazem binarnym. Naszym celem jest teraz wyznaczenie obszarów obrazu w obrębie których z wysokim prawdopodobieństwem znajdują się znaki drogowe. Mówiąc bardziej trywialnie staramy się teraz wykryć znaki drogowe na analizowanym obrazie. W dalszej części posługiwać się będziemy określeniami, których znaczenie w kontekście tej pracy chcielibyśmy doprecyzować:

plama: Przez plamę rozumiemy grupę sąsiadujących ze sobą niewygaszonych (białych) pikseli na obrazie binarnym.

obiekt: Obiekt jest to prostokątny wycinek analizowanego kadru. Obiekt definiowany jest przez prostokąt opisujący plamę lub zespół plam.

Wynikiem działania tego etapu są wycinki obrazu wejściowego (kadru filmu), które z wysokim prawdopodobieństwem zawierają znaki drogowe.

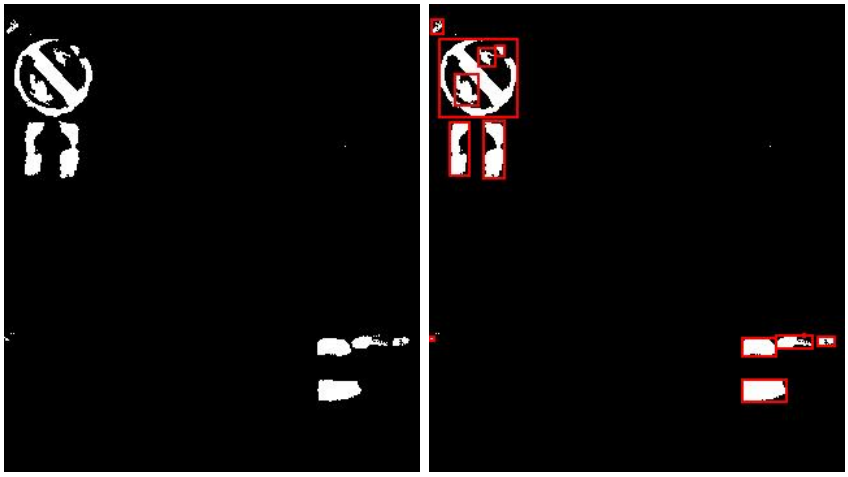
2.3.1. Wykrywanie plam

Pierwszym krokiem jest wyznaczanie plam na utworzonym obrazie binarnym. Początkowo stworzyliśmy w tym celu algorytm rekurencyjny. Okazał się on jednak zbyt wymagający pamięciowo, dlatego zaprojektowaliśmy wersję sekwencyjną. Algorytm ten działa następująco:

1. przebiegamy obszar zainteresowania przeskakując co dwa piksele w poziomie i co dwa piksele w pionie,
2. jeżeli natrafimy na biały piksel definiujemy nową plamę o rozmiarze 1×1 i środku odpowiadającym współrzędnym piksela i dodajemy ją do listy plam,
3. ustawiamy barwę piksela na czarną, ponieważ został już przypisany do definiowanej plamy,
4. dodajemy ten punkt do listy punktów, których sąsiadów należy zbadać,
5. dopóki lista punktów do zbadania nie jest pusta, 'wyjmujemy' z niej pierwszy element, powiedzmy punkt p:
   - przebiegamy wszystkie piksele otaczające p, jeżeli któryś z nich jest biały:
     (a) ustawiamy jego barwę na czarną,
     (b) dodajemy go do listy punktów, których sąsiadów należy zbadać,
     (c) aktualizujemy współrzędne prostokąta opisującego definiowaną plamę o ten punkt.

W efekcie działania tego algorytmu otrzymujemy listę plam mogących być znakami drogowymi lub ich fragmentami. Plamy te zdefiniowane są za pomo-

cą opisujących je prostokątów. Rysunek 2.9 przedstawia plamy wykryte na obrazie binarnym będącym efektem segmentacji (wykryte plamy obrysowane są czerwonymi prostokątami).



Rys. 2.9. Przykład działania algorytmu wykrywania plam na obrazie binarnym

Jak widzimy znaki często dzielone są na kilka mniejszych obiektów. W procesie segmentacji dopuszczaliśmy barwy typowe dla zewnętrznych części znaków drogowych, symbole w obrębie znaków najczęściej są czarne lub białe, gdybyśmy dopuszczali także te barwy podczas segmentacji, do powstałego obrazu binarnego dostawałoby się zbyt wiele elementów otoczenia.

Dlatego kolejnym, bardzo istotnym krokiem będzie próba odnalezienia obiektów opisujących fragmenty tego samego znaku drogowego i połączenia ich w jeden większy obiekt.

2.3.2. Łączenie i odrzucanie wykrytych obiektów

Dysponujemy listą wykrytych plam. Celem tego etapu jest odrzucenie części z nich oraz połączenie grup plam w większe. W wyniku opisanych niżej zabiegów otrzymamy listę fragmentów obrazów uznanych za reprezentacje znaków drogowych.

Usuwanie bardzo małych plam

Najpierw usuniemy z listy plam te, dla których powierzchnia prostokąta je opisującego jest mniejsza lub równa dziesięć. Większość z nich to pojedyncze piksele, które zostały niesłusznie uznane za fragmenty znaków drogowych. Plamy, które pozostaną na liście po tym zabiegu poddamy dalszej analizie.

Łączenie obiektów reprezentujących fragmenty tego samego znaku drogowego

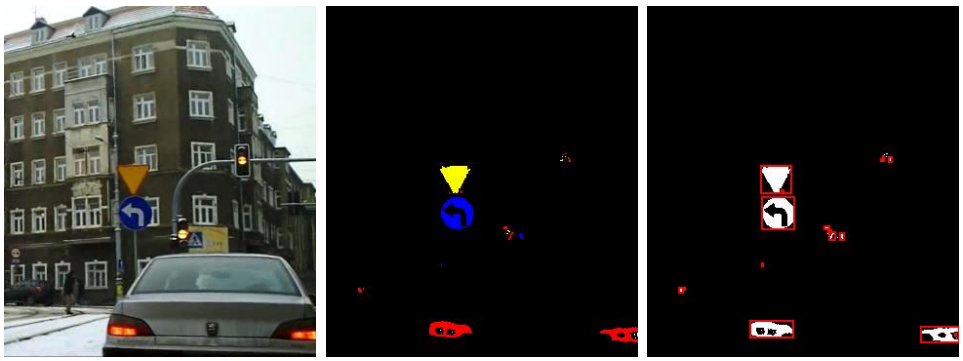
Moduł wnioskowania rozmytego wykorzystany do podejmowania decyzji odnośnie łączenia obiektów zaimplementowany został w języku FCL (Fuzzy Control Language, [9]).

Zanim przystąpimy do łączenia ze sobą obiektów reprezentujących ten sam znak drogowy stworzyć musimy zestaw reguł pozwalających stwierdzić, czy badana para obiektów reprezentuje różne fragmenty tego samego znaku. Przy podejmowaniu decyzji posłużymy się rozmytym systemem regułowym. Dlatego reguły tworzymy w oparciu o język naturalny. Matematyczny model określeń takich jak “względnie blisko siebie w poziomie” omówimy później. Stworzenie modeli tego typu określeń pozwoli nam na zdefiniowanie reguł w języku naturalnym. Reguły te opisywać będą ludzki sposób postrzegania i podejmowania decyzji.

Dwa obiekty mogą być fragmentami tego samego znaku, jeśli:

- jeden z nich zawiera się w znacznej mierze w drugim,
- są względnie blisko siebie w poziomie i jednocześnie położone podobnie w pionie,
- są względnie blisko siebie w pionie i jednocześnie położone podobnie w poziomie.

W wyniku testów rozbudowaliśmy nieco trzecią regułę. Obiekty leżące blisko siebie w pionie oraz położone podobnie w poziomie nie zostaną uznane za fragmenty tego samego znaku, jeśli oba prostokąty je opisujące zbliżone są do kwadratów. Ten dodatkowy warunek zapobiec ma łączeniu ze sobą obiektów reprezentujących różne znaki drogowe umiejscowione fizycznie jeden pod drugim. Sytuacja taka widoczna jest na rysunku 2.10.



Rys. 2.10. Przykład znaków drogowych umieszczonych jeden pod drugim

Algorytm dla każdej pary obiektów sprawdzać będzie, czy spełniają one warunki wystarczające do połączenia ich w jeden większy.
Zdefiniować musimy zmienne wejściowe, których wartości będą podstawą do podejmowania decyzja:

d_h – względna odległość pozioma pomiędzy badaną parą obiektów,

d_v – względna odległość pionowa pomiędzy badaną parą obiektów,

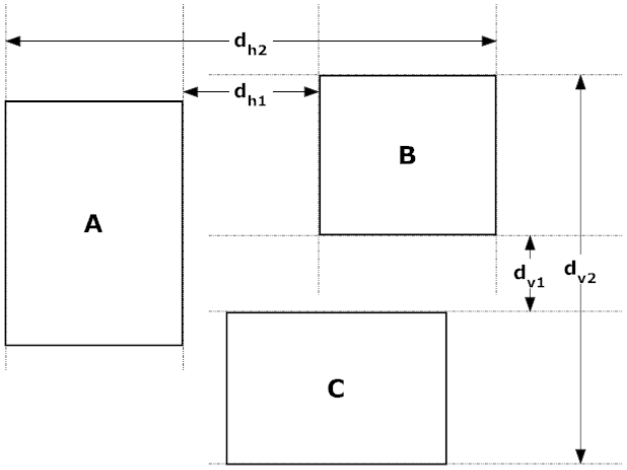
p_h – względna różnica w położeniu rzutów obiektów na oś X,

p_v – względna różnica w położeniu rzutów obiektów na oś Y,

inter – część wspólna badanych obiektów względem powierzchni mniejszego z nich,

ratio – stosunek szerokości do wysokości obiektu.

Jeżeli rzuty obiektów na oś X nachodzą na siebie ich względna odległość pozioma (d_h) równa jest 0. Podobnie jeśli ich rzuty na oś Y na siebie nachodzą, względna odległość pionowa pomiędzy obiektami (d_v) równa jest 0. W przeciwnym wypadku odległości te wyliczamy z wzorów 2.1 i 2.2. Wzory te bazują na odległościach zdefiniowanych na rysunku 2.11.

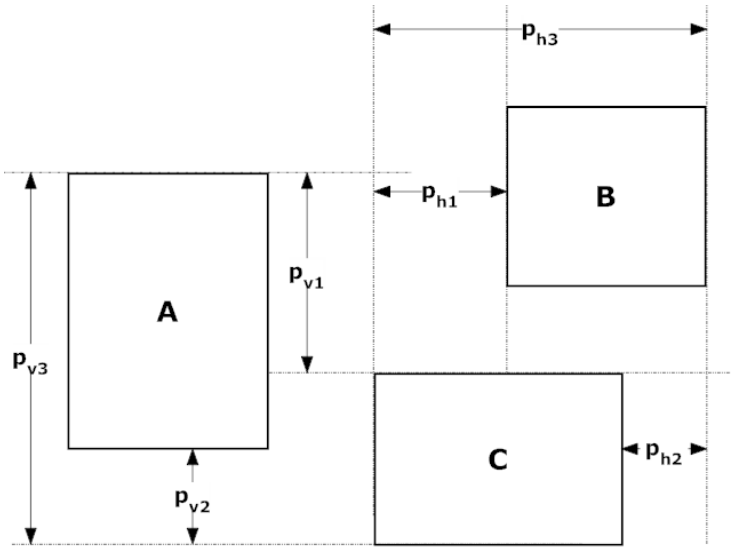


Rys. 2.11. Wyznaczanie odległości względnej pomiędzy obiektami

$$d_h = \frac{d_{h1}}{d_{h2}} \tag{2.1}$$

$$d_v = \frac{d_{v1}}{d_{v2}} \tag{2.2}$$

Względną różnice w poziomym i pionowym położeniu obiektów wyliczamy z wzorów 2.3 i 2.4. Bazują one na wartościach zdefiniowanych na rysunku 2.12.



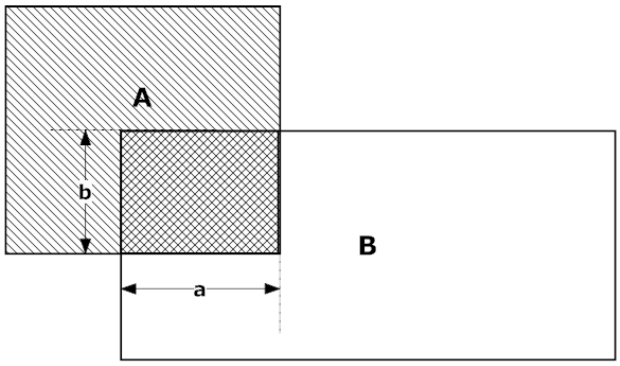
Rys. 2.12. Wyznaczanie względnej różnicy w położeniu obiektów

$$p_h = \frac{p_{h1} + p_{h2}}{p_{h3}} \tag{2.3}$$

$$p_v = \frac{p_{v1} + p_{v2}}{p_{v3}} \tag{2.4}$$

Stosunek powierzchni części wspólnej obiektów do powierzchni mniejszego z nich wyliczamy z wzoru 2.5 (w oparciu o rys. 2.13). *Ratio* to po prostu iloraz szerokości i wysokości obiektu (wzór 2.6).

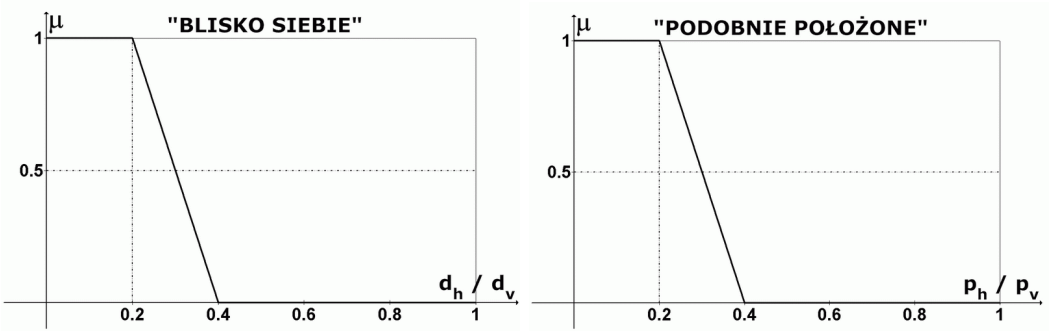
$$section = \frac{ab}{\min(P_A, P_B)} \tag{2.5}$$



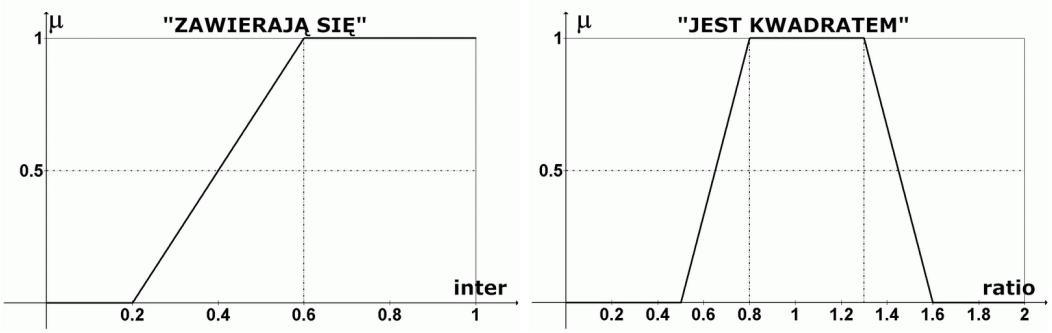
Rys. 2.13. Wyznaczanie części wspólnej obiektów

$$ratio = \frac{width}{height} \tag{2.6}$$

W poglądowym zestawie reguł, jaki stworzyliśmy, wprowadziliśmy kilka pojęć. Nasz system regułowy na wejście dostanie wartości liczbowe. Musimy więc zdefiniować zbiory rozmyte opisujące te pojęcia. Tak jak wcześniej, zbiory rozmyte opisujemy poprzez zdefiniowanie funkcji przynależności. Rysunek 2.14 przedstawia funkcje przynależności do zbiorów "blisko siebie" oraz "podobnie położone". Na rysunku 2.15 widzimy funkcje przynależności do zbiorów "zawierają się" oraz "jest kwadratem".



Rys. 2.14. Funkcje przynależności do zbiorów "blisko siebie" i "podobnie położone"



Rys. 2.15. Funkcje przynależności do zbiorów "zawierają się" i "jest kwadratem"

W oparciu o zdefiniowane zmienne wejściowe oraz zbiory rozmyte możemy sformalizować zestaw reguł decydujących o tym, czy badana para obiektów reprezentuje różne fragmenty tego samego znaku:

DEFAULT: *theSameSign* `IS` *'nie'*

RULE 1: `IF` *inter* `IS` *'zawierają się'* `THEN` *theSameSign* `IS` *'tak'*

RULE 2: `IF` d_h `IS` *'blisko siebie'* `AND` p_v `IS` *'podobnie położone'* `THEN` *theSameSign* IS *'tak'*

RULE 3: `IF` ($ratio_A$ `IS NOT` *'jest kwadratem'* `OR` $ratio_B$ `IS NOT` *'jest kwadratem'*) `AND` d_v `IS` *'blisko siebie'* `AND` p_h `IS` *'podobnie położone'* `THEN` *theSameSign* IS *'tak'*

Przy obliczeniach stosujemy T-normę minimum i T-conormę maksimum.

Powyższe reguły stosujemy do każdej pary plam z listy, którą dysponujemy. Jeśli dla badanej pary wynik jest pozytywny, to znaczy system reguły stwierdził, że są one fragmentami tego samego znaku, łączy je w jedną większą plamę. Połączenie dwóch plam polega na zastąpieniu jej jedną większą, taką, że prostokąt ją opisujący obejmuje oba prostokąty opisujące łączone plamy.

Dalsza analiza obiektów

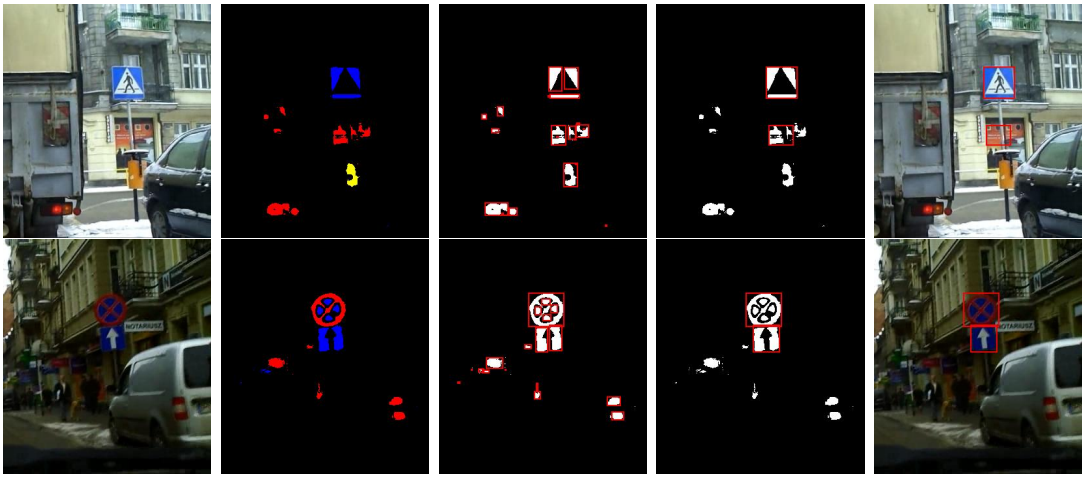
Po etapie łączenia plam opisujących te same znaki drogowe przystępujemy do usuwania z listy plam, które nie spełniają postawionych kryteriów:

- Usuwamy plamy zbyt małe, zakładając, że nawet jeśli reprezentują one znak drogowy, to są zbyt małe, by go rozpoznać. Plamę uznajemy za zbyt małą jeśli wysokość lub szerokość prostokąta ją opisującego jest mniejsza od 32.
- Usuwamy plamy zbyt duże, nie jest możliwe aby znak drogowy wypełniał cały obszar zainteresowania.
- Stosunek szerokości do wysokości większości znaków drogowych mieści się w pewnym przedziale. Wyliczamy ten stosunek dla każdej plamy. Usuwamy z listy plamy, dla których ta proporcja: $ratio \notin [0.3, 3.0]$.

Rysunek 2.16 prezentuje wyniki wszystkich powyższych zabiegów. Widzimy na nim kolejno fragment kadru, wynik segmentacji, wynik działania algorytmu wykrywania plam oraz efekt łączenia i odrzucania plam.

2.4. Reprezentacja znaku drogowego

Etapem następującym po wykryciu znaków jest oczywiście próba ich rozpoznania. W celu rozpoznania wykryty znak drogowy porównywany będzie z informacjami w bazie danych. Oczywiście nie możemy porównywać ze sobą obrazów. Nie dość, że byłoby to zbyt czaso i pamięciochłonne to jeszcze mało efektywne. Dlatego każdy fragment obrazu stanowiący ujęcie znaku drogowego musimy sprowadzać do jakiegoś kodu, ciągu liczb. Chcemy stworzyć taki sposób kodowania, który pozwoli na zachowanie istotnych informacji na



Rys. 2.16. Efekt działania segmentacji, detekcji plam oraz ich łączenia i odrzucania

temat kodowanego znaku drogowego. Jednocześnie znaczące jest stworzenie kodu, który łatwo i szybko porównać będzie można z wielką ilością kodów w bazie danych.

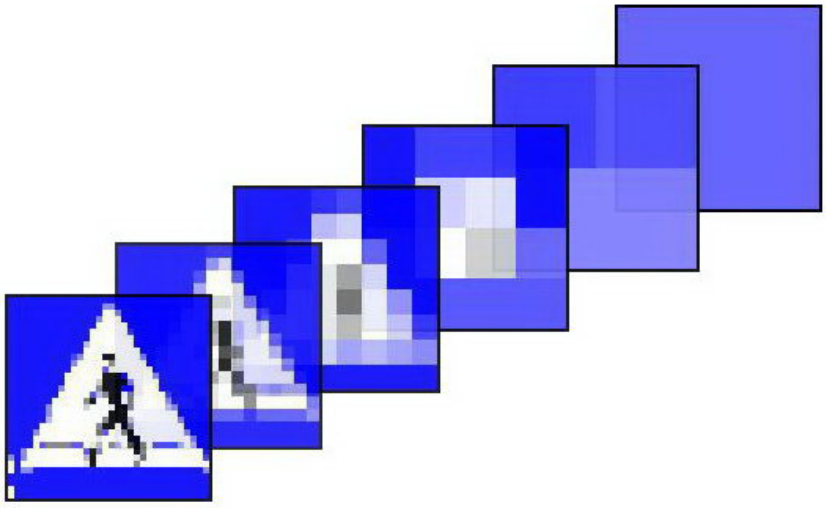
2.4.1. Ogólna idea

Kodować będziemy prostokątny obraz, w obrębie którego znajdować się powinien znak drogowy. Zaczniemy od rozpoznania barwy każdego piksela na tym obrazie i w zależności od niej przypisania mu pewnej liczby. Rozpatrujemy jedynie barwy pojawiające się na znakach drogowych: żółtą, czerwoną, niebieską, czarną i białą oraz barwę nietypową, nie stanowiącą części znaku.

Kod znaku tworzyć będą kolejne wektory liczbowe. Każdy z nich opisywać będzie coraz bardziej szczegółowo rozmieszenie poszczególnych barw w obrębie kodowanego obrazu. Pierwszy wektor to jedna liczba – uśrednienie wartości przypisanych wszystkim pikselom na podstawie ich barwy. W celu wyliczenia wartości następnego wektora liczbowego dzielimy kodowany obraz na cztery części. Wektor składa się z czterech liczb, z których każda to śred-

nia ważona wartości przypisanych pikselom w obrębie pola, które opisuje. Następnie każde z pól, na jakie podzieliliśmy obraz, znów dzielimy na cztery. Kolejny wektor składa się więc z szesnastu liczb. Wyznaczamy w ten sposób sześć wektorów, pierwszy składający się z jednej liczby, a każdy następny czterokrotnie dłuższy od poprzedniego. Ostatni odpowiada podziałowi kodowanego obrazu na 1024 (32×32) części.

Opis taki pozwala realizować ideę coraz bardziej szczegółowego przyglądania się znakowi. Pierwszy wektor to jakby znak widziany z dużej odległości, nie opisuje żadnych kształtów, symboli, jedynie zarys i barwę (w postaci jednej liczby – uśrednienia wartości przypisanych pikselom badanego obrazu). Każdy kolejny wektor to znak oglądany z coraz bliższej odległości, coraz lepiej opisuje kształt znaku oraz zawarte w jego obrębie symbole i ich barwy. Zastosowane podejście ilustruje rysunek 2.17 (str. 69).



Rys. 2.17. Zestawienie graficznych reprezentacji sześciu wektorów opisujących obraz dla przykładowego znaku drogowego

Zakładamy, że system oparty będzie na bazie danej przechowującej bardzo dużą ilość reprezentacji znaków drogowych. Zarysowane powyżej podejście pozwolić ma na szybkie wybranie grupy znaków mogących odpowiadać

szukanemu. Początkowe wektory (bardzo krótkie) stosować będziemy do natychmiastowej eliminacji większości reprezentacji znaków drogowych z bazy danych. Natomiast dłuższe wektory pozwolą na dokładniejsze porównanie znaku drogowego, który system próbuje rozpoznać z tymi, które zostały wybrane z bazy danych jako podobne.

2.4.2. Algorytm kodowania

W ogólnym ujęciu algorytm kodowania obrazu składa się z dwóch kroków:

- segmentacja obrazu przy uwzględnieniu sześciu klas (barw):
  - barwa czarna
  - barwa biała
  - barwa niebieska
  - barwa czerwona
  - barwa żółta
  - barwa inna niż powyższe
- wyznaczenie sześciu wektorów opisujących kodowany obraz

Segmentacja kodowanego obrazu

Zakładamy, że na tym etapie mamy już do czynienia ze znakiem drogowym lub jego fragmentem. Dlatego proces segmentacji będzie mniej skomplikowany, warunki, jakie będzie musiał spełnić piksel by zostać uznany za np. żółty, będą prostsze i mniej restrykcyjne niż te zastosowane przy segmentacji całego obrazu (sekcja 2.2, strona 49).

Każdy piksel przypisany zostanie do jednej z sześciu klas. Pięć z tych klas opisuje konkretne barwy, szósta przeznaczona jest dla punktów obrazu o barwie nie należącej do żadnej z tych pięciu. Każdej klasie przypisana jest

liczba. Wejściem algorytmu segmentacji jest więc obraz – dwuwymiarowa tablica opisująca piksele, jego wyjściem jest dwuwymiarowa tablica zawierająca kody przypisane tym pikselom.

Oto lista klas (barw) oraz przypisanych im kodów:

0 – barwa nietypowa dla znaku drogowego (klasa *'grey'*)
10 – barwa żółta (klasa *'yellow'*)
20 – barwa czerwona (klasa *'red'*)
30 – barwa niebieska (klasa *'blue'*)
40 – barwa biała (klasa *'white'*)
50 – barwa czarna (klasa *'black'*)

W procesie segmentacji znaku drogowego bazujemy na modelu barw RGB (patrz punkt 1.1.2, str. 17). Ponieważ model RGB nie bazuje bezpośrednio na barwie, a jedynie na stopiniach nasycenia barw składowych, przy jego zastosowaniu łatwiej ocenić czy piksel jest barwy czarnej lub białej.

Na podstawie wartości RGB każdemu pikselowi kodowanego obrazu przypisywana jest jedna z powyższych wartości. Oto konstrukcja warunkowa decydująca do jakiej klasy przypisać badany piksel (R, G, B – wartości składowych modelu barw RGB dla badanego piksela, x – badany piksel):

`IF` $(R \geq 40 \wedge G \geq 60 \wedge B \geq 50 \wedge \frac{R}{G} \in [0.55, 1.2] \wedge \frac{R}{B} \in [0.4, 2.3])$ `THEN` $x \in$*'WHITE'*

`ELSE IF` $(R \geq 95 \wedge \frac{G}{R} \in [0.5, 1.2] \wedge \frac{R}{B} \geq 4)$ `THEN` $x \in$*'YELLOW'*

`ELSE IF` $(B \geq 20 \wedge \frac{B}{R} \geq 2 \wedge \frac{B}{G} \geq 1.6)$ `THEN` $x \in$*'BLUE'*

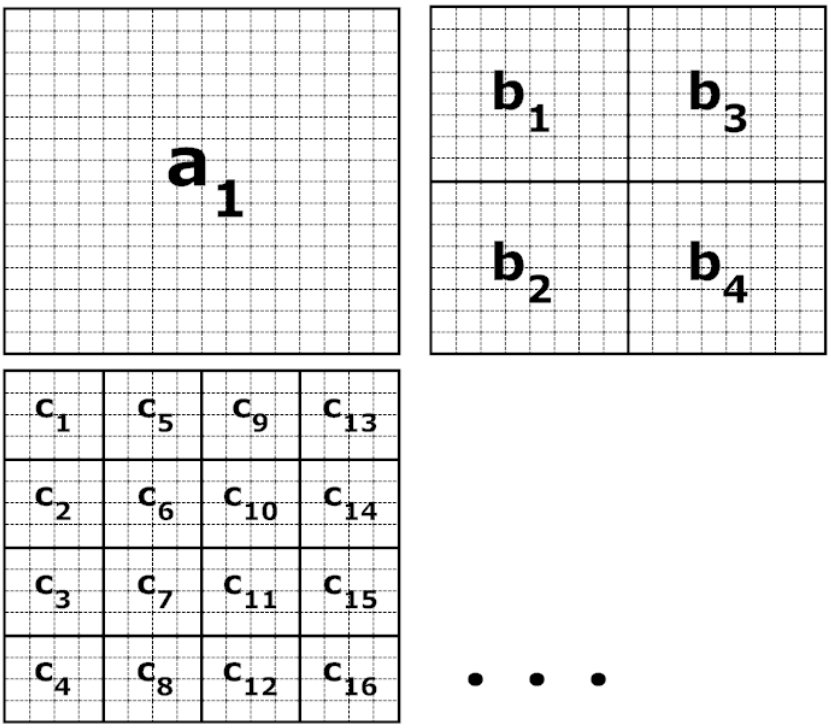
`ELSE IF` $((R \geq 30 \wedge \frac{R}{G} \geq 4.5 \wedge \frac{R}{B} \geq 4.5) \vee (R \geq 60 \wedge \frac{R}{G} \geq 1.6 \wedge \frac{R}{B} \geq 2) \vee (R \geq 150 \wedge \frac{R}{G} \geq 1.2 \wedge \frac{R}{B} \geq 1.8))$ `THEN` $x \in$*'RED'*

`ELSE IF` $(R < 110 \wedge G < 70 \wedge B < 50)$ `THEN` $x \in$*'BLACK'*

`ELSE` $x \in$*'GREY'*

Wyznaczanie wektorów stanowiących kod znaku drogowego

Dysponujemy teraz tablicą, która każdemu pikselowi przypisuje liczbę opisującą jego barwę. Znak drogowy kodujemy do sześciu wektorów, z których każdy kolejny opisuje obraz coraz bardziej szczegółowo. Pierwszy wektor to jedna liczba opisująca cały znak drogowy. Każdy kolejny wektor jest czterokrotnie dłuższy od poprzedniego, ponieważ każde pole z poprzedniego podziału dzielone jest na cztery mniejsze pola (2×2) (rys. 2.18).



Rys. 2.18. Sposób dzielenia znaku drogowego podczas kodowania

Aby zakodować obraz musimy wyznaczyć sześć wektorów:

$$A = [a_1], B = [b_1, b_2, b_3, b_4], \ldots, F = [f_1, f_2, \ldots, f_{1024}]$$

Dla każdego pola x_i wartość liczbowa wyliczana jest następująco:

$$x_i = \frac{suma}{ilosc}, \qquad gdzie : \tag{2.7}$$

$x \in \{a, b, c, d, e, f\}$,

suma to suma wartości przypisanych pikselom należącym do badanego pola,

ilosc to ilość pikseli o rozpoznanej barwie + 0.2 × ilość pikseli nierozpoznanych (należących do klasy 'grey').

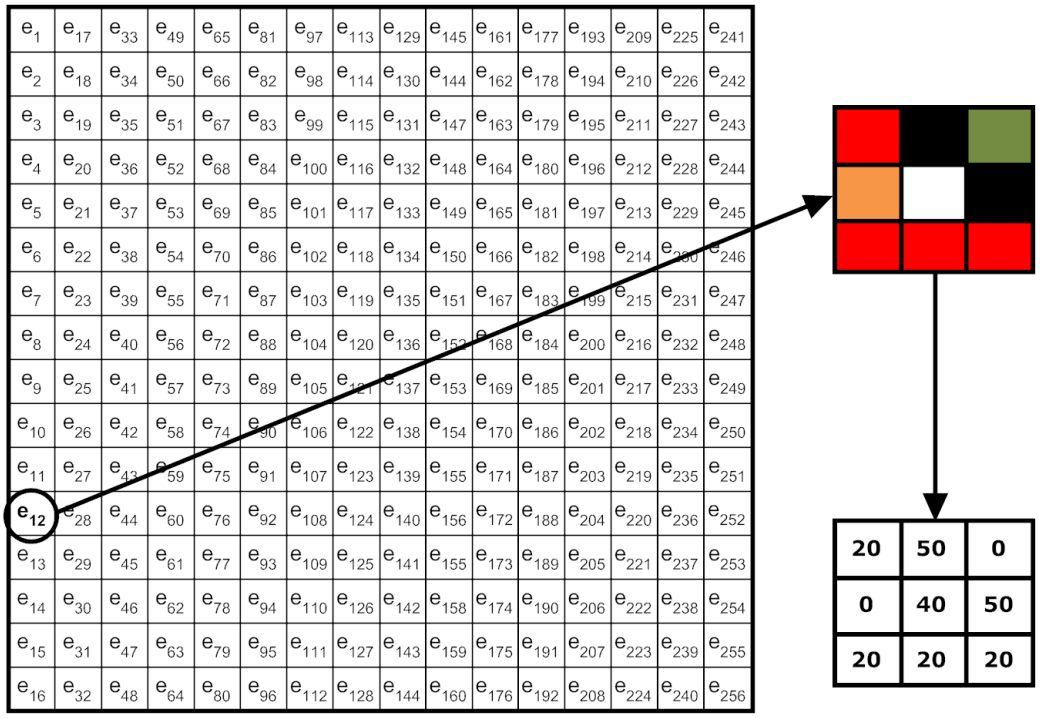
Wartość wyliczona dla dowolnego pola spełnia: $x_i \in [0, 50]$.

Piksele o barwie nierozpoznanej, czyli albo nieczytelne albo nienależące do znaku drogowego, mają przypisaną wartość zero. Moglibyśmy także nie uwzględniać ich naliczając ilość punktów w badanym obszarze. Chcemy jednak by punkty takie jawnie wpływały na wyliczoną wartość, dlatego traktujemy każdy z nich jak 0.2 piksela o rozpoznanej barwie typowej dla znaku. Gdy kodujemy okrągły znak drogowy w wielu przypadkach otoczony jest on punktami o barwie nierozpoznanej (nietypowej dla znaku drogowego). Ilość takich punktów w obrębie badanego pola wpływa na wartość przypisanej mu liczby, więc liczba ta przechowuje także informacje na temat kształtu znaku.

Aby lepiej zobrazować proces kodowania znaku drogowego posłużymy się przykładem. Załóżmy, że wyliczyć mamy wartość dla pola e_{12}, rysunek 2.19.

Pikselom przypisaliśmy wartości zgodnie z wcześniej podanymi zasadami. Wyliczmy zatem:

$$e_{12} = \frac{20 + 50 + 40 + 50 + 20 + 20 + 20}{7 + 0.2 \times 2} = \frac{220}{7.4} \approx 29.73$$



Rys. 2.19. Przykład wyliczania pojedynczej wartości kodu znaku drogowego

Oczywiście pojedyncza, odseparowana wartość nie przechowuje przydatnych informacji.

Pierwszy wektor opisuje cały badany obszar jedną liczbą, zawiera pewne informacje o barwach w obrębie tego obszaru. Następny mówi już coś więcej na temat rozkładu tych barw. Każdy kolejny podział niesie w sobie kolejne informacje. Podejście takie ma pozwolić na szybkie uznanie porównywanej pary znaków za różne od siebie, poprzez porównanie pierwszych dwóch lub trzech wektorów. Zauważmy tutaj, że pierwsze wektory są bardzo krótkie, pozwala to na założenie indeksów w bazie danych na kolumnach, które je przechowują. Dzięki temu możemy bardzo szybko wydobyć z bazy danych jedynie te znaki, które są dość podobne do badanego. Dopiero dla tej grupy znaków porównywać będziemy kolejne (znacznie dłuższe) wektory.

Na ryunkach 2.20 i 2.21 widać przykłady graficznych reprezentacji kodu dwóch reprezentacji znaku ‘przejście dla pieszych’ oraz ‘ograniczenie prędkości do 50 km/h’. Na górze po prawej stronie widzimy kodowany fragment obrazu (uznany za znak drogowy). Niżej efekt segmentacji tego fragmentu (struktura warunkowa na str. 71). Po prawej stronie graficzna reprezentacja kodu – każdy z sześciu fragmentów reprezentuje kolejny, coraz dłuższy i bardziej szczegółowy, wektor opisujący obraz.

Jak widzimy, pomimo tego że znaki drogowe na tych obrazach są nieco inaczej ułożone i innych rozmiarów, pierwsze trzy wektory je opisujące są do siebie bardzo zbliżone.

2.5. Baza wiedzy

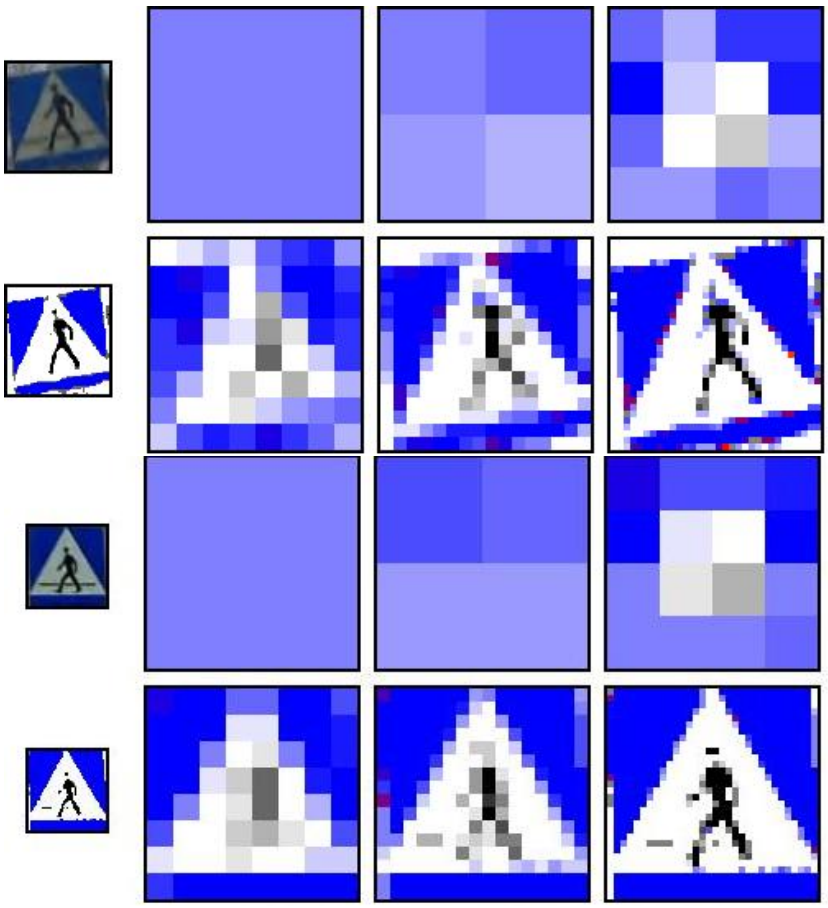
Baza wiedzy to element systemu, gdzie składowane są kody reprezentacji oraz opisy znaków. Struktura ta pozwala na szybkie wyszukiwanie elementów podobnych do zadanego wzorca oraz uzyskanie informacji jaki znak przedstawia dana reprezentacja.

2.5.1. Struktura bazy, indeksy

Baza wiedzy (Rys. 2.22) jest prostą bazą danych składającą sie z trzech tabel:

signs – przechowuje informacje o znakach drogowych dodanych do systemu

- id – klucz główny
- sname – nazwa znaku
- description – opis znaku

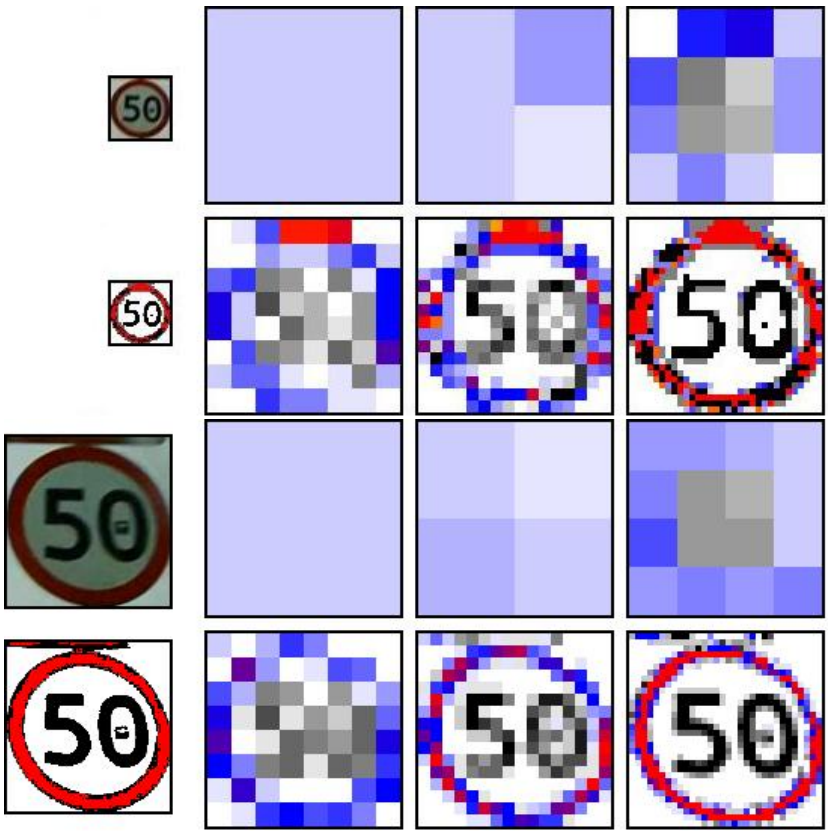


Rys. 2.20. Proces kodowania znaków drogowych na przykładzie dwóch znaków 'przejście dla pieszych'

- image – obrazek wykorzystywany przy wyświetlaniu znaku w interfejsie użytkownika

sign_images – zawiera zakodowane do postaci wektorów liczb reprezentacje znaków (Rozdział 2.4.2)

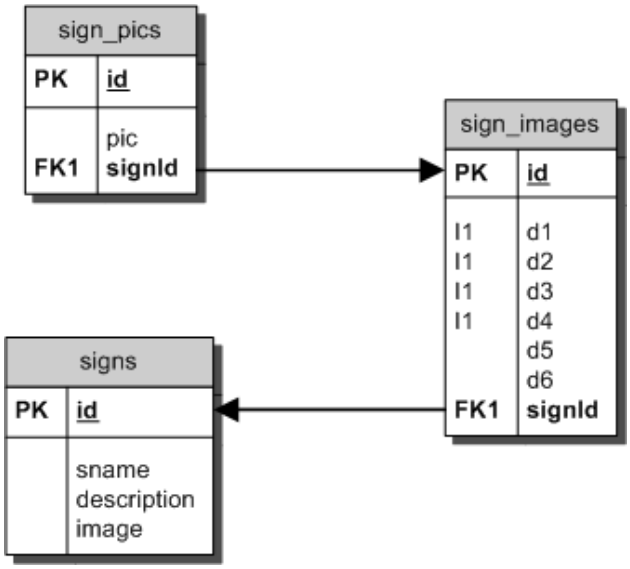
- id – klucz główny
- d1...d6 – kolejne wektory opisujące reprezentację znaku
- signId – referencja do znaku odpowiadającego bieżącej reprezentacji



Rys. 2.21. Proces kodowania znaków drogowych na przykładzie dwóch znaków ‘ograniczenie prędkości do 50 km/h’

sign_pics – dodatkowa tabela przechowująca dla każdej zakodowanej reprezentacji obraz, który jej odpowiada

- id – klucz główny
- pic – obraz
- signId – referencja do reprezentacji znaku



Rys. 2.22. Schemat bazy danych

Dla usprawnienia wyszukiwania podobnych reprezentacji na kolumnach $d1 \ldots d4$ tabeli *sign_images* założony został indeks. Dobór kolumn jest wynikiem przeprowadzonej przez nas analizy. Zależy nam na tym, aby czas wyszukiwania podobnych reprezentacji, jak i wielkość podzbioru spełniającego kryteria, były jak najmniejsze.

Wyszukiwanie w bazie zrealizowane zostało przy pomocy procedury składowanej `getD1234`. Jej argumentami jest zestaw ośmiu wektorów odpowiadających długościami wektorom w kolumnach $d1 \ldots d4$ tabeli *sign_images* oraz stanowiących swoiste, górną i dolną, "granice" wyszukiwania w bazie. Jej działanie polega na wybraniu z tabeli *sign_images* reprezentacji, które mieszczą się we wczesniej wspomnianych granicach.

2.5.2. Uczenie bazy danych

Aby skutecznie rozpozwać znaki należy zbudować bazę wiedzy o jak największej liczbie zakodowanych reprezentacji. Główny problemem stanowi pozy-

skanie jak największej ilości materiału szkoleniowego wysokiej jakości, czyli takiego gdzie średnia częstotliwość występowania znaków jest wysoka (w praktyce zgromadzenie 1000 reprezentacji znaków przekłada się na 2–3 godziny nagrań).

Następnie system analizuje zgromadzony obraz wideo i wykrywa na nim fragmenty, które spełniają opisane w Rozdz. 2.2.3 warunki (na 1h nagrania przypada średnio 10000 fragmentów). Zgromadzone wycinki są prezentowane operatorowi, który stwierdza czy dany fragment reprezentuje znak. Jeśli istotnie tak jest, po wskazaniu przez operatora konkretnego znaku, fragment jest kodowany i zapisywany do bazy jako jego reprezentacja. W przeciwnym wypadku system prezentuje kolejny wycinek.

Po zbudowaniu należy zweryfikować jej zawartość – często okazuje się, że zapisane w niej reprezentacje, zamiast wnosić dodatkową informację, powodują zakłócenia w rozpoznawaniu (np. przypisanie reprezentacji do nieodpowiedniego znaku lub zbyt duża liczba zakłóceń w obrazie). Problemy te mogą być łatwo rozwiązane dzięki wbudowanej przeglądarce bazy wiedzy.

2.5.3. Pozyskiwanie materiału szkoleniowego

Baza wiedzy powstała na podstawie znaków wykrytych na filmach szkoleniowych. Obraz ten został nagrany przy pomocy kamery cyfrowej na ulicach Poznania i okolic. Nagrania miały miejsce w różnych porach roku i przy zróżnicowanych warunkach atmosferycznych i oświetleniowych aby zakres analizy, jak i sama wielkość budowanej na ich podstawie bazy wiedzy, były jak największe. Kamera zainstalowana została przed siedzeniem pasażera w samochodzie osobowym. Skierowana została 5° na prawo od osi pojazdu, tak aby w prawej części rejestrowanego obrazu znajdowało się jak najwięcej znaków.

Udało nam się zgromadzić około 10 godzin nagrań, które poddane zostały obróbce. Polegała ona na połączeniu wszystkich zarejestrowanych fragmentów w jeden, przekonwertowaniu go do rozdzielczości 640x360 (30fps) oraz zakodowaniu go przy użyciu kodeka *XVid MPEG-4*

Tak przygotowany obraz poddany został analizie, dzieki której udało się wyznaczyć przedziały kolorów określające znak. Posłużył on również jako materiał do utworzenia bazy wiedzy.

2.6. Miara podobieństwa reprezentacji znaków

W celu znalezienia najlepiej odpowiadających sobie reprezentacji zaprojektowaliśmy miarę podobieństwa Sim, która dla dwóch reprezentacji R_1 i R_2, opisanych zestawami wektorów $R_1 = (A_1, B_1, C_1, D_1, E_1, F_1)$ i $R_2 = (A_2, B_2, C_2, D_2, E_2, F_2)$, zwraca wartość z przedziału $[0, 1]$.

Wartości bliższe jedynce odpowiadają większemu podobieństwu rozpatrywanych reprezentacji:

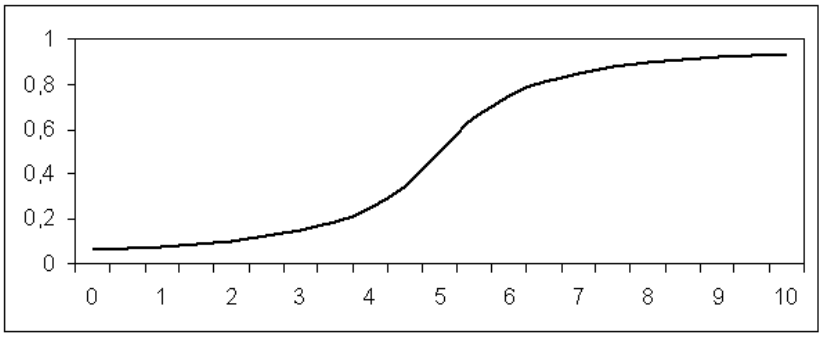
$$\begin{aligned} Sim(R_1, R_2) &= 1 - (0.01 * D_1(R_1, R_2) + 0.02 * D_2(R_1, R_2) + 0.07 * \\ & D_3(R_1, R_2) + 0.1 * D_4(R_1, R_2) + 0.2 * D_5(R_1, R_2) + 0.6 * \\ & D_6(R_1, R_2)) \end{aligned}$$

gdzie:

$$\begin{aligned} D_1(R_1, R_2) &= Dif(|a1_1 - a2_1|) \\ D_2(R_1, R_2) &= \frac{\sum_{j=0}^{4} Dif(|b1_j - b2_j|)}{4} \\ D_3(R_1, R_2) &= \frac{\sum_{j=0}^{16} Dif(|c1_j - c2_j|)}{16} \\ D_4(R_1, R_2) &= \frac{\sum_{j=0}^{64} Dif(|d1_j - d2_j|)}{64} \\ D_5(R_1, R_2) &= \frac{\sum_{j=0}^{256} Dif(|e1_j - e2_j|)}{256} \\ D_6(R_1, R_2) &= \frac{\sum_{j=0}^{1024} Dif(|f1_j - f2_j|)}{1024} \end{aligned}$$

oraz Dif jest fuknją opisującą stopień różnicy argumentów (Rys. 2.23):

$$Dif(x) = \begin{cases} \frac{arctan(x-5)+\frac{\pi}{2}}{\pi} & if\ x > 0 \\ 0 & if\ x = 0 \end{cases} \tag{2.8}$$



Rys. 2.23. Wykres funkcji Dif

2.7. Rozpoznawanie znaków drogowych

Moduł rozpoznawania to element systemu, który na wejściu otrzymuje wycinek obrazu, sprawdza czy jest on wystarczająco podobny do którejkolwiek reprezentacji w bazie i na końcu, w przypadku znalezienia takiej reprezentacji, prezentuje znak przypisany do niej. Poszczególne kroki działania algorytmu rozpoznawania znaku wyglądają następująco:

1. Pobranie fragmentu obrazu do rozpoznania i zakodowanie go
2. Wyszukanie reprezentacji znaków podobnych w bazie danych
3. Wyliczenie stopnia podobieństwa dla każdej reprezentacji
4. Sprawdzenie, czy reprezentacja o najwyższej wartości podobieństwa jest wystarczająco bliska rozpoznawanemu wzorcowi
   - 4a. Tak – wyszukanie w bazie znaku odpowiadającego danej reprezentacji, wyświetlenie go i powrót do pkt. 1
   - 4b. Nie – powrót do pkt. 1

2.7.1. Pobieranie obrazu do rozpoznania

Jeśli moduł wykrywania, analizując klatkę obrazu wideo, zlokalizuje na nim znak, to ten wycinek obrazu jest kodowany do postaci reprezentacji znaku:

$$Inp = (A^{Inp}, B^{Inp}, C^{Inp}, D^{Inp}, E^{Inp}, F^{Inp})$$

Format ten odpowiada temu, w którym przechowywane są dane w bazie wiedzy, dzięki czemu możliwe będzie porównanie go z reprezentacjami w

niej przechowywanymi.Dzięki mierze podobieństwa Sim (Rozdz. 2.6), możnaby łatwo wyznaczyć reprezentację i tym samym znak najbardziej podobny do Inp.

Niestety wyznaczenie miary podobieństwa dla wszystkich reprezentacji w bazie jest operacją zbyt kosztowną obliczeniowo (dodatkowo rośnie liniowo wraz z rozmiarem bazy). W następnym podrozdziale opisany zostanie sposób pozyskiwania z bazy podzbioru reprezentacji najbliższych poszukiwanemu wzorcowi.

2.7.2. Pozyskiwanie grupy znaków podobnych z bazy

Dzięki opisanym wcześniej (Rozdz.2.5) indeksom oraz procecedurze składowanej `getD1234` możliwe jest znaczne ograniczenie wielkości zbioru reprezentacji do porównania.

Dla rozpoznawanego obrazu Inp tworzone są sztuczne reprezentacje ograniczające od góry $H = (A^H, B^H, C^H, D^H)$ i od dołu $L = (A^L, B^L, C^L, D^L)$ obszar poszukiwań w bazie danych. Reprezentacje te składają sie tylko z czterech wektorów liczb, odpowiadających czterem pierwszym (najkrószym) wektorom reprezentacji w bazie. Te czery wektory określają każdą reprezentację w stopniu wystarczającym do stwierdzenia czy jest ona podobna do obrazu wejściowego.Wyznaczane są w następujący sposób:

$$\forall_{i=1}^{1} A_i^L = A_i^{In} - \delta_A,\ A_i^H = A_i^{In} + \delta_A$$
$$\forall_{i=1}^{4} B_i^L = B_i^{In} - \delta_B,\ B_i^H = B_i^{In} + \delta_B$$
$$\forall_{i=1}^{16} C_i^L = C_i^{In} - \delta_C,\ C_i^H = C_i^{In} + \delta_C$$
$$\forall_{i=1}^{64} D_i^L = D_i^{In} - \delta_D,\ D_i^H = D_i^{In} + \delta_D$$

gdzie:
$\delta_A = 1$, $\delta_B = 2$, $\delta_C = 4$ i $\delta_D = 4$ to wyznaczone eksperymentalnie wartości.

Wyznaczone w ten sposób wektory ograniczeń są argumentami procedury składowanej `getD1234`, która zwraca pozdbiór reprezentacji $S = (R_1, ...R_n)$, $R_i = (A^i, B^i, C^i, D^i, E^i, F^i)$ takich, że dla każdej z nich:

$$\begin{cases} \forall_{j=1}^{1} A_j^L \leq A_j^i < A_j^H \\ \forall_{j=1}^{4} B_j^L \leq B_j^i < B_j^H \\ \forall_{j=1}^{16} C_j^L \leq C_j^i < C_j^H \\ \forall_{j=1}^{64} D_j^L \leq D_j^i < D_j^H \end{cases}$$

Wielkość podzbioru S zależy od obrazu wejściowego oraz od doboru parametrów $\delta_A \ldots \delta_B$. Podjęliśmy próbę dynamicznego ustalania owych parametrów, tak aby wielkość zbioru S mieściła się w określonych granicach. Niestety okazało się, że, z powodu dużej liczby dodatkowych zapytań do bazy, średni czas rozpoznania pojedyńczego znaku znacznie się wydłużył. Wpłynęło to na ograniczenie wydajności całego systemu i dlatego zrezygnowaliśmy z tego podejścia.

2.7.3. Wyliczanie stopnia podobieństwa

Dla każdej reprezentacji ze zbioru S wyznaczany jest stopień podobieństwa do obrazu wejściowego:

$$s_i = Sim(Inp, R_i) \text{ dla } i = 1 \ldots n$$

Następnie wybieramy reprezentację R_k o największej wartości s. Aby uznać reprezentacje R_k oraz Inp za określające ten sam znak spełniony musi zostać warunek:

$$s_k \geq 0.8$$

Powyższa granica ustalona została ekperymentalnie. Wartości bliższe jedynce prowadzą do wzrostu odsetka poprawnie rozpoznanych znaków przy spadku ogólnej liczby rozpoznanych znaków (jej zmniejszanie powoduje efekt odwrotny).

Rozdział 3

Rezultaty

3.1. Informacje w bazie danych

Aby testować aplikację zebraliśmy około pięć godzin nagrań filmowych. System automatycznie wyodrębnił z tego materiału fragmenty obrazu mogące być znakami drogowymi. Następnie ręcznie wybraliśmy spośród tych fragmentów obrazu reprezentacje znaków drogowych, których obecność w bazie danych uznaliśmy za istotną.

Sprowadza się to do tego, że staraliśmy się unikać wprowadzania do bazy danych dwóch reprezentacji tego samego znaku drogowego jeśli są one do siebie bardzo zbliżone. Reprezentacje tego samego znaku drogowego uznajemy za różne jeśli spełniają przynajmniej jeden z poniższych czynników:

- różnią się rozmiarem,
- pochodzą z nagrań wykonanych w różnych warunkach atmosferycznych, różnią się odcieniem, jasnością lub nasyceniem barwy,
- ujęte na nich znaki drogowe są pod nieco innymi kątami.

Tabela 3.1 podaje zebraną w bazie ilość reprezentacji dla poszczególnych znaków drogowych wybranych do testów.

Tab. 3.1. Ilość reprezentacji wybranych do testów znaków drogowych w bazie danych

Symbol	Opis znaku	Liczba reprezentacji
A-7	Ustąp pierwszeństwa przejazdu	96
B-21	Zakaz skrętu w lewo	61
B-36	Zakaz zatrzymywania się	187
C-4	Nakaz skrętu w lewo za znakiem	50
D-1	Droga z pierwszeństwem	150
D-6	Przejście dla pieszych	156

3.2. Metodologia testów

W celu uzyskania jak najbardziej zgodnych z rzeczywistością wyników testów przyjęliśmy szereg założeń:

- nagranie testowe bedzie miało długość 2h i będzie się składało z nagrań pobranych przy różnych warunkach atmosferycznych i oświetleniowych,
- badać będziemy ilość znaków wykrytych dla każdego rodzaju znaku,
- wśród wykrytych znaków zliczać będziemy znaki poprawnie rozpoznane,
- w badaniu brane będą pod uwagę jedynie znaki o największej liczbie reprezentacji w bazie wiedzy, stanowiące reprezentantów wszystich grup znaków, jeśli chodzi o kształt i kolor(patrz Tab. 3.1),

- wynik to średnia ważona (wagi proporcjonalne do ilości wykrytych/rozpoznanych znaków).

3.3. Wyniki testów

Wyniki testów przedstawione zostały w Tab. 3.2.

Tab. 3.2. Wyniki testów

Znak	**Moduł wykrywania**	**Moduł rozpoznawania**
	Znaki wykryte	Znaki rozpoznane
A-7	57%	100%
B-21	100%	100%
B-36	88%	68%
C-4	100%	100%
D-1	53%	75%
B-6	88%	95%
Średnia	**83.0%**	**84.4%**

3.4. Porównanie z wynikami podobnych systemów

W tabeli 3.3 zaprezentowana została efektywność naszego systemu na tle czterech innych:

M. Schneier *Road Sign Detection and Recognition* – System wykrywający znaki drogowe w oparciu o reguły określające kolor, kształt oraz domnimane umiejscowienie znaku na obrazie. Wykrywanie znaków przez

porównywanie z szablonami w bazie danych oraz śledzenie poprzez sekwencje obrazów ([13]).

M. Taha Khan *Real–Time Recognition System For Traffic Signs* – Do wykrywania znaków wykorzystano szereg filtrów graficznych w przestrzeni kolorów HSV (segmentacja, wyszukiwanie obszarów homogenicznych, etykietowanie, filtrowanie rozmiaru wykrywanego obiektu). Rozpoznanie znaku nastepuje przez dopasowanie kształtu obrazu jednego ze wzorców ([14]).

M. Fifik, J. Turán, L. Ovseník *Experiments with a Transform based Traffic Sign Recognition System* – System wyszukuje znaki przy pomocy algorytmow segmentacji kolorów i kształtów. Zniekształcenia obrazu niwelowane są przez zastosowanie tranfotmat Hough i Trace. Rozpoznanie znaku nastepuje przez zaklasyfikowanie wykrytego obszaru do jednej z grup na podstawie opisujacych go cech ([15]).

J. Miura, T. Kanda, Y. Shirai *An Active Vision System for Real-Time Traffic Sign Recognition* – System wykorzystujący dwie kamery. Pierwsza, z obiektywem szerokokątnym, służy do wykrywania znaków–kandydatów z wykorzystaniem informacji o kolorze, jasności oraz kształcie. Druga kamera, wyposażona w teleobiektyw, zostaje nakierowana na przewidywaną pozycję wybranego kandydata w celu pobrania dokładniejszego obrazu. Rozpoznanie znaku nastepuje przez dopasowanie go do jednego z szablonów ([16]).

Szczegółowy opis działania tych systemów można znaleźć w źródłach zawartych w bibliografii.

Tab. 3.3. Porównanie efektywności

System	**Moduł wykrywania**	**Moduł rozpoznawania**
	Wykryte	Rozpoznane
M. Schneier [13]	88%	78%
M. Taha Khan [14]	–	95%
M. Fifik, J. Turán, L. Ovseník [15]	–	86%
J. Miura, T. Kanda, Y. Shirai [16]	97%	42%
Nasz system	**83.0%**	**84.4%**

3.5. Podsumowanie

Stworzony przez nas system rozpoznawania znaków drogowych realizuje wszystkie stawiane przed nim oczekiwania: działa w czasie rzeczywistym, pozwala na rozszerzanie bazy wiedzy i wykorzystuje wnioskowanie rozmyte przy podejmowaniu decyzji. Dzięki rozwiązaniom, na które zdecydowaliśmy się w fazie projetkowania i zweryfikowaliśmy podczas implementacji system pozwala na rozpoznawanie znaków wszystkich typów i, jak widać na zestawieniu powyżej (Tab. 3.3), wypada bardzo dobrze na tle innych rozwiązań.

Opisana w niniejszej pracy metodyka, dzięki zastosowaniu reguł rozmytych oraz rozpoznawania obrazu w oparciu o metodę “od ogółu do szczegółu”, może zostać zastosowana do wykrywania i rozpoznawania praktycznie dowolnych obiektów na obrazie wideo (po przeprowadzeniu analizy i modyfikacji reguł sterujących).

Spis rysunków

Spis tabel

Bibliografia

[1] T. Acharya and A. K. Ray, *Image Processing. Principles and Applications*, John Wiley & Sons, Inc., 2005.

[2] H. R. Kang, *Color technology for electronic imaging devices*, SPIE - The International Society for Optical Engineering, 1996.

[3] F. Y. Shih, *Image Processing and Pattern Recognition: Fundamentals and Techniques*, John Wiley & Sons, Inc., 2010.

[4] M. Krzyśko, W. Wołyński, T. Górecki, and M. Skorzybut, *Systemy uczące się*, Wydawnictwo Naukowo-Techniczne Sp. z o.o., 2008.

[5] M. Wygralak, *Cardinalities of Fuzzy Sets*, Springer-Verlag Berlin Heildelberg, 2003.

[6] M. Brown, *An Introduction to Fuzzy and Neurofuzzy Systems*, Prentice Hall International, 1996.

[7] *Fuzzy Logic Toolbox,* `http://www.mathworks.de/access/helpdesk/help/toolbox/fuzzy/fp351dup8.html`.

[8] M. Nachtegael and E. E. Kerre, *Fuzzy techniques in image processing*, Physica-Verlag Heidelberg, cop., 2000.

[9] International Electrotechnical Commision, *Technical Committee No. 65: Industrial Process Measurement and Control*: *Part 7 - Fuzzy Control Programming*, 1997.

[10] L. A. Zadeh, *Fuzzy sets*, Information and Control 8, 1965.

[11] W. Pedrycz and F. Gomide, *Fuzzy Systems Engineering Towards Human-Centric Computing*, John Wiley & Sons, Inc., 2007.

[12] M. Wygralak, *Wykład do przedmiotu "Teoria i Zastosowania Zbiorów Nieostrych"*, UAM, 2008.

[13] M. Shneier, *Road Sign Detection and Recognition*, IEEE Computer Society International Conference on Computer Vision and Pattern Recognition, 2005.

[14] M. Taha Khan, *Real–Time Recognition System For Traffic Signs*, Department of Computer Engineering, Dalarna University, 2008.

[15] J. Turán M. Fifik L. Ovseník, *Experiments with a Transform based Traffic Sign Recognition System*, International Conference on Systems, Signals and Image Processing, 2010.

[16] T. Kanda J. Miura Y. Shirai, *An Active Vision System for Real-Time Traffic Sign Recognition*, Department of Computer-Controlled Mechanical Systems, Osaka University, 2010.