

**POLITECHNIKA WROCLAWSKA**

**WYDZIAŁ ELEKTRONIKI**

---

KIERUNEK: Automatyka i Robotyka

SPECJALNOŚĆ: Technologie informacyjne w systemach automatyki

# **PRACA DYPLOMOWA MAGISTERSKA**

Badanie metod rozpoznawania znaków  
drogowych z wykorzystaniem wielu źródeł  
obrazu

Research on traffic sign recognition methods  
using multiple image source

autor : BARTOSZ LENARTOWICZ

Opiekun pracy:  
Dr inż. Bartosz Jabłoński W04/K8

**OCENA PRACY:**

---

WROCLAW 2019

## **Streszczenie**

Niniejsza praca zatytułowana „Badanie metod rozpoznawania znaków drogowych z wykorzystaniem wielu źródeł obrazu” jest pracą dyplomową zrealizowaną z myślą o stworzeniu systemu potrafiącego rozpoznawać znaki drogowe oraz sprawdzeniu jaki wpływ na skuteczność systemu ma zastosowanie wielu, kamer. System przedstawiony w pracy jest w stanie rozpoznawać i klasyfikować 33 Polskie znaki drogowe. W pracy zbadano zalety i wady wykorzystania większej ilości kamer. Praca została zrealizowana na potrzeby studiów II stopnia Politechniki Wrocławskiej.

## Spis treści

Streszczenie.....	2
Analiza problemu, cel i zakres pracy .....	4
Metody detekcji obiektów na obrazach .....	6
Zastosowania różnych typów rejestratorów obrazu .....	8
Tworzenie panoramy na potrzeby systemu TSR.....	9
Metody bazujące na punktach charakterystycznych .....	11
Segmentacja obrazów .....	13
Wykrywanie znaków na podstawie koloru .....	14
Wykrywanie na podstawie kształtu .....	17
Wykrywanie na podstawie tekstury .....	21
Podejście hybrydowe oraz inne podejścia .....	22
Klasyfikacja znaków drogowych .....	29
Wzorcowe dopasowanie .....	29
Sztuczne sieci neuronowe .....	29
Maszyna wektorów nośnych do klasyfikacji .....	33
Drzewo k-wymiarowe.....	34
Lasy losowe .....	34
Podsumowanie.....	34
Opis proponowanej metody do rozwiązania problemu TSR.....	36
Propozycja wykorzystania większej ilości kamer .....	36
Segmentacja .....	37
Klasyfikacja .....	43
Przygotowanie środowiska badawczego i plan badań.....	46
Wyniki badań .....	48
Podsumowanie.....	60
Bibliografia .....	61

## Analiza problemu, cel i zakres pracy

Automatyczne rozpoznanie znaków drogowych jest aspektem niosącym ze sobą wiele korzyści, szczególnie jeżeli ich analiza następuje w czasie rzeczywistym. Korzyścią może być wspomaganie kierowców podczas jazdy. Taki system, nazywany ADAS (ang. Advanced Driver Assistance System) zostaje implementowany w wielu współczesnych samochodach i poprawia komfort i bezpieczeństwo prowadzenia samochodu. Po nieco dłuższym zastanowieniu można wywnioskować, że rozpoznawanie znaków również jest niezbędne dla autonomicznych pojazdów. Muszą one mieć możliwość adaptacji do zmieniającej się infrastruktury drogi oraz zasad na niej panującej np. w trakcie robót budowlanych. Oprócz dwóch powyższych aspektów rozpoznanie znaków drogowych może realnie przyczynić się do polepszania bezpieczeństwa na drodze. Badając w jakim stopniu znak jest czytelny np. czy nie został zamazany, zakrzywiony system może informować służbę drogową. Zastosowanie automatycznego rozpoznawania znaków znajduje zastosowanie w coraz to nowych dziedzinach z całą pewnością można rzec, że taki aspekt będzie poruszany w wielu pracach badawczych. Warto wspomnieć, że dziedzina jest rozwijana od ponad dwóch dekad. Jedną z pierwszych prac nad systemem została przedstawiona w roku 1996 w pracy [1].

Starając się wyjaśnić zakres pracy magisterskiej należy rozszerzyć kilka terminów, które zostały w niej poruszone. W literaturze problem rozpoznawania znaków drogowych przedstawia się jako problem TSR (ang. Traffic Sign Recognition). Jest on składową większego zagadnienia jakim jest analiza sytuacji w oku samochodu podczas jazdy. W takim przypadku obszar wokół samochodu traktowany jest jako scena, w której należy rozpoznawać i analizować obiekty na niej umieszczone. Zostaje to wspomniane, ponieważ niektóre systemy prócz rozpoznania znaków drogowych są w stanie rozpoznawać inne obiekty takie jak samochody, rowerzystów czy pieszych [2].



Rysunek 1 Rodzaje obiektów na i wokół drogi

Rozpoznanie większej ilości obiektów rozszerza możliwości systemu i umożliwia śledzenie obiektów ruchomych np. w celu poinformowania kierowców o kursie kolizyjnym. Śledzenie pozwala również zapobiec wykrywaniu kilkakrotnie tego samego obiektu. Według autorów

pracy [3] aspekt śledzenia jest szczególnie ważny dla systemów, w których pojazd jest kierowany przez kierowcę (a nie przez algorytm). Wielokrotne informowanie kierowcy o znaku może rozproszyć jego uwagę powodując tym samym zagrożenie. Autorzy w swojej pracy również proponują traktowanie kierowcy jako integralną część środowiska i śledzenie jego ruchów dla lepszej integracji systemu z kierowcą.

Niniejsza praca została stworzona z myślą o wzbogaceniu systemu wykrywania znaków drogowych przez podłączenie do systemu kolejnych kamer. W celu realizacji pracy dyplomowej należało wykonać kilka zadań. Pierwszym z nich był przegląd literaturowy metod lokalizacji obiektów na obrazie oraz metod rozpoznawania znaków drogowych. Następnie należało przeanalizować możliwości wykorzystania wielu źródeł obrazu. Kluczowym etapem było opracowanie algorytmu detekcji znaków na podstawie więcej niż jednego źródła obrazu. Po jego opracowaniu należało zaimplementować algorytm w wybranym środowisku oraz przygotować dane, które mogłyby posłużyć do testów. Najważniejszym etapem dla pracy było wykonanie eksperymentów i przeanalizowanie ich wyników. Na sam koniec należało przedstawić uzyskane wyniki czytelnikowi i zredagować całą pracę w taki sposób by była spójna i czytelna.

Opis pracy dyplomowej został stworzony w następujący sposób. Wpierw zostało przedstawione streszczenie zawierające krótki opis pracy. Na następnej stronie został umieszczony spis treści. Następnie we wstępie przedstawiono tematykę problemu, zadania do wykonania i strukturę pracy. W części literaturowej zostały opisane przykłady rozwiązania problemu wykrywania znaków przez autorów prac naukowych. Następnie przedstawiono autorskie rozwiązanie problemu wyszukiwania znaków drogowych oraz parametry sieci neuronowej stworzonej do klasyfikacji znaków. W kolejnym akapicie opisano w jaki sposób system był testowany oraz przedstawiono wyniki tych testów. Pracę wieńczy akapit podsumowujący zrealizowane zadania.

## Metody detekcji obiektów na obrazach

W niniejszym rozdziale zostały przedstawione metody, które starają się sprostać zadaniu rozpoznawania obiektów na obrazie. Większość z przedstawionych metod skupia się na wykrywaniu i rozpoznaniu pionowych znaków drogowych. Zazwyczaj przedstawiane metody są podzielone na dwa etapy. Pierwszym z nich jest wyszukanie na obrazie regionów mogących zawierać znak drogowy lub obiektów przypominający taki znak. Taki etap nazwany jest segmentacją obrazu. Ponieważ znaki drogowe mają konkretnie określone kolory i kształty, w przypadku problemu TSR etap segmentacji prawie zawsze poprawia szybkość algorytmu. Metody segmentacji zostały przedstawione w podrozdziale Segmentacja obrazów. Dla rejonów wyselekcjonowanych przez algorytmy segmentacji należy sprawdzić czy znak drogowy faktycznie na nich istnieje i jeżeli istnieje rozpoznać jakie ma on znaczenie. Etap ten nazywany jest klasyfikacją, a metody odpowiadające za klasyfikacje zostały przedstawione w rozdziale Klasyfikacja znaków drogowych.

### *Rodzaje znaków drogowych*

Przedstawiając metody odpowiedzialne za segmentację i klasyfikację należy zaznaczyć, że znaki drogowe nie zostały ujednolicone na całym świecie. Istnieją dwie główne konwencje starające się je ujednolicić. Pierwszą z nich jest grupa znaków opartych według zasad MUTCD (ang. Manual on Uniform Traffic Control Devices). Tworzenie znaków według tych restrykcji obowiązuje głównie w krajach położonych w Ameryce Północnej i Południowej, Australii i krajach azjatyckich położnych blisko Australii. Kraje położone w Europie, Azji oraz Afryce używają znaków drogowych zaprojektowanych według Konwencji Wiedeńskiej. Oczywiście istnieją kraje np. Argentyna, Brazylia, Chiny, które przyjmują część zasad z jednej, a część z drugiej konwencji lub projektują je według własnych zasad. Znaki każdej konwencji mają ujednolicone wzory tworzenia i konkretnie przypisane kolory i kształty dla każdego typu znaku. Konwencja również wprowadziła zasady obowiązujące dla znaków poziomych i sygnalizacji świetlnej. Niestety dopuszcza ona rozbieżności według projektowania znaków. W większości krajów znaki ostrzegawcze Konwencji Wiedeńskiej są przedstawiane jako biały równoboczny trójkąt z czarnym symbolem pośrodku czerwoną obwódką, lecz w kilku krajach w tym w Polsce białe tło zastąpiono kolorem żółtym. Problem różnorodności znaków został to wspomniany, ponieważ wszystkie rozwiązania problemu TSR, w tym te przedstawione w niniejszej pracy, bazują na wykrywaniu znaków za pomocą algorytmów bazujących na kolorze czy kształcie. Opisywane prace skupiać się będą wokół wykrywania znaków tworzonych według Konwencji Wiedeńskiej.

### *Bazy znaków drogowych*

Ważnym aspektem każdej pracy baza obrazów ze znakami drogowymi. Niekiedy w starszych algorytmach autorzy samodzielnie selekcjonowali klatki filmu, zaś większość nowych publikacji zostało przetestowanych na istniejących bazach danych. Powszechnie wykorzystywane publiczne bazy danych to baza:

- [German Traffic Sign Recognition Benchmark](#),
- [BelgiumTS Dataset](#),
- [Chinese Traffic Sign Database](#)
- [Linkopings Universitet Traffic Signs Dataset](#),
- [RUG Traffic Sign Image Database](#).



Zbiorem wyjściowym dla większości publikacji jest pierwszy przedstawiony zbiór danych drogowych GTSRB, w tłumaczeniu Niemiecki Benchmark Znaków Drogowych. Baza ta złożona jest z 43 klas znaków pochodzących z Niemiec. Każda klasa posiada od kilku do kilkunastu osobnych znaków drogowych. Każdy znak został przedstawiony 30 razy obrazach powstałych poprzez wyselekcjonowanie z sekwencji wideo następujących po sobie klatek filmu. Tym sposobem każda klasa posiada od 200 do 2000 obrazów. Obraz jest złożony głównie ze znaku, który wypełnia ok 80 % obrazu. Rysunek 2 przedstawia 30 obrazów jednego znaku z tego zbioru. Drugi ze zbiorów BelgiumTS posiada 62 klasy znaków. Znaki zostały sfotografowane pojedynczo i również zajmują około 80 % obrazu. Każda klasa posiada od 5 do 300 obrazów. Zbiór trzeci CTSD jest to zbiór posiadający jak poprzednio znaki zajmujące 80 % obrazu posiadający 58 klas znaków pochodzących z Chin. Większość znaków chińskich projektowane jest według zasad Konwencji Wiedeńskiej, lecz niektóre z nich tworzone bez konkretnych zasad oraz zamiast symboli posiadają Chińskie znaki. Niemniej jednak część prac zwłaszcza azjatyckich bazuje na tym zbiorze. Zbiór czwarty LUTSD jest Szwedzkim zbiorem posiadający inaczej niż w przypadku poprzednich zbiorów całe obrazy pochodzące z ulicy, gdzie znaki posiada niewielką część obrazów oraz część obrazów bez znaków. Uniwersytet Linkopings dodał adnotacje do 1940 takich obrazów, a około 4 razy tyle udostępnił bez adnotacji. Ostatnim zbiorem jest Rosyjski zbiór danych zawierający 48 obrazów podpisanych w tak jak nazywają się znaki znajdujące się na nich. Gdyby nie różnica w kolorze znaków ostrzegawczych w dwóch ostatnich zbiorach mógłby być przydatne do testowania budowanego algorytmu.



*Rysunek 2 Znak Ustąp pierwszeństwa ze zbioru GTSRB*

Głównym zagadnieniem pracy jest wpływ zastosowania większej ilości kamer dla wykrywania znaków drogowych. Taki problem może być poruszany na kilka sposobów. Pierwszym z nich jest zastosowanie różnych typów rejestratorów obrazu. Skierowane na ten sam obszar w różny sposób odwzorowują otoczenie na obrazie co może powodować polepszenie się skuteczności wyszukiwania zwłaszcza w trudnych warunkach oświetleniowych. Takie zagadnienie zostało przedstawione w akapicie: Zastosowania różnych typów . Inną możliwością jest objęcie przez kilka kamer jednego typu obszaru, który byłby niemożliwy do objęcia przez jedną kamerę. Obrazy uzyskane w ten sposób mogą być analizowane osobno lub można połączyć je ze sobą tworząc obraz panoramiczny, jeżeli w jakimś stopniu się. Algorytmy tworzenia obrazu

panoramicznego i wpływ na zagadnienie TSR został przedstawione w akapicie Zastosowania różnych typów rejestratorów obrazu

Oprócz klasycznej kamery dostarczającej obraz widzialny przez człowieka istnieje wiele typów rejestratorów obrazu mogących polepszyć wykrywanie znaków w trudnych warunkach. Z wszystkich najbardziej pomocny dla rozpoznawania sytuacji na drodze wydaje się kamera z oświetlaczem podczerwieni. Przetwarzanie obrazu dostarczonego przez obie kamery może odbywać się równolegle lub możliwa jest połączenie obrazów. W tym miejscu można przytoczyć pracę [4], w której został zaproponowany algorytm do łączenia klatek filmowych tego samego obszaru nagrywanych różnymi kamerami. Autorzy testowali algorytm na dwóch kamerach – TV i IR. Ponieważ zazwyczaj kamery rejestrują obraz o różnych rozdzielczościach, algorytm przeskalowuje obrazy do jednej wielkości, a następnie zostają wyszukane krawędzie na obrazie. Autorzy pracy nie podają jaką metodą wyszukują krawędzie. W większości przypadków prac taką metodą jest metoda Canny opisana w akapicie Detektor Canny. Następnie jeden z obrazów zostaje podzielony pionowo na równe części. Każdy z wycinków algorytm stara się dopasować do obrazu drugiego. Najlepsze dopasowanie każdego odcinka wyznacza się za pomocą sumy najmniejszych kwadratów lub sumy modułów różnic. Następnie za pomocą jednej z metod: średniej, mediany lub dominanty wybiera się najlepsze dopasowanie ogólne. Na podstawie dopasowania ogólnego zostaje wyznaczona translacja pozioma obrazów. Warto zaznaczyć, że dominanta uzyskała najlepsze wyniki dopasowania. Gdy obrazy zostają przesunięte o wyznaczoną wartość i proceder powtarza się dla translacji pionowej. Algorytm jest słabo odporny na rotację, jego największą zaletą jest szybkość. Opisując źródła, z których można czerpać inspirację dla fuzji obrazów nie sposób nie wspomnieć o medycynie. Przykładem, który można podać jest tomografia lub rezonans magnetyczny, gdzie nie dość, że pojedyncze klatki obrazów są łączone w taki sposób by tworzyć obraz 3-D to jeszcze często na potrzeby dokładniejszej rozeznania łączy się obrazy 3-D z różnych źródeł. Artykuł przedstawia metodę Powella z użyciem metody Brenta na potrzeby łączenia obrazów medycznych. Chcąc podsumować rozwiązanie zastosowania różnych typów rejestratorów dla problemu TSR można stwierdzić, że jest to dział niosący wiele możliwości, lecz dotychczas niebadany.

Tworzenie panoramy na potrzeby systemu TSR. Niestety żadna z znalezionych artykułów dla pracy dyplomowej nie rozwija tematu zastosowania większej ilości kamer, dlatego posłużono się publikacjami, które zostały stworzone na potrzeby systemów przeznaczonych do innych zadań.

### **Zastosowania różnych typów rejestratorów obrazu**

Oprócz klasycznej kamery dostarczającej obraz widzialny przez człowieka istnieje wiele typów rejestratorów obrazu mogących polepszyć wykrywanie znaków w trudnych warunkach. Z wszystkich najbardziej pomocny dla rozpoznawania sytuacji na drodze wydaje się kamera z oświetlaczem podczerwieni. Przetwarzanie obrazu dostarczonego przez obie kamery może odbywać się równolegle lub możliwa jest połączenie obrazów. W tym miejscu można przytoczyć pracę [4], w której został zaproponowany algorytm do łączenia klatek filmowych tego samego obszaru nagrywanych różnymi kamerami. Autorzy testowali algorytm na dwóch kamerach – TV i IR. Ponieważ zazwyczaj kamery rejestrują obraz o różnych rozdzielczościach, algorytm przeskalowuje obrazy do jednej wielkości, a następnie zostają wyszukane krawędzie na obrazie. Autorzy pracy nie podają jaką metodą wyszukują krawędzie. W większości przypadków prac taką metodą jest metoda Canny opisana w akapicie Detektor Canny. Następnie jeden z obrazów zostaje podzielony pionowo na równe części. Każdy z wycinków



algorytm stara się dopasować do obrazu drugiego. Najlepsze dopasowanie każdego odcinka wyznacza się za pomocą sumy najmniejszych kwadratów lub sumy modułów różnic. Następnie za pomocą jednej z metod: średniej, mediany lub dominanty wybiera się najlepsze dopasowanie ogólne. Na podstawie dopasowania ogólnego zostaje wyznaczona translacja pozioma obrazów. Warto zaznaczyć, że dominanta uzyskała najlepsze wyniki dopasowania. Gdy obrazy zostają przesunięte o wyznaczoną wartość i proceder powtarza się dla translacji pionowej. Algorytm jest słabo odporny na rotację, jego największą zaletą jest szybkość. Opisując źródła, z których można czerpać inspirację dla fuzji obrazów nie sposób nie wspomnieć o medycynie. Przykładem, który można podać jest tomografia lub rezonans magnetyczny, gdzie nie dość, że pojedyncze klatki obrazów są łączone w taki sposób by tworzyć obraz 3-D to jeszcze często na potrzeby dokładniejszej rozeznania łączy się obrazy 3-D z różnych źródeł. Artykuł [5] przedstawia metodę Powella z użyciem metody Brenta na potrzeby łączenia obrazów medycznych. Chcąc podsumować rozwiązanie zastosowania różnych typów rejestratorów dla problemu TSR można stwierdzić, że jest to dział niosący wiele możliwości, lecz dotychczas niebadany.

### **Tworzenie panoramy na potrzeby systemu TSR**

W przypadku zarejestrowania otoczenia wokół samochodu przez wiele kamer, gdzie każda kamera obejmuje jego wycinek, jedną z pierwszych rzeczy jaka się nasuwa jest stworzenie obrazu panoramicznego. Taka innowacja może posiadać kilka zalet. Pierwszą z nich jest posiadanie ciągłości obrazu otoczenia wokół samochodu. Rozpoznanie znaków na takim obrazie nie powoduje sytuacji konfliktowych w przypadku rozpoznania jednego znaku inaczej na dwóch obrazach. Przykładowo, gdy mamy dwa obrazy niepołączone ze sobą, na jednym obrazie możemy rozpoznać inny znak niż na drugim co wprowadza problem rozstrzygnięcia, który ze znaków jest bardziej prawdopodobny. Jedną z bardziej istotnych zalet systemu z panoramą jest możliwość wychwycenia raz obiektu i jego śledzenie do czasu, kiedy nie zniknie z pola widzenia wszystkich kamer. Jak zostało wspomniane we wstępie jest to bardzo istotne dla systemów ADAS. Dodatkowo przy segmentacji obrazu otrzymanego poprzez panoramę przetwarzane jest mniej jest analiza mniejszej ilości obrazów na etapie segmentacji.

Istnieje szereg algorytmów starających się sprostać problemowi dopasowania kilku obrazów do siebie. Istotną cechą poszukiwanego algorytmu powinna być odporność na różnicę w jasności w obrazach wejściowych oraz szumy spowodowane drganiem i rotacją oraz zabrudzeniami typu: błoto, kurz, krople wody. W wielu pracach np. [6] łączenie obrazów w panoramę zostaje podzielone na kilka etapów. W artykule starano się uniwersalną metodę mogącą łączyć obrazy tej samej sceny uzyskane w różnych czasach, z różnych punktów widzenia lub zarejestrowane różnymi czujnikami. Według pracy proponuje się trzy etapy łączenia obrazów. Etapy graficznie zostały przedstawione na Rysunek 3. Wpierw następuje wykrywanie cech kluczowych, gdzie wykrywa się np. kontury, narożniki, centra grawitacji, punkty kluczowe. Następnie dopasowuje się cechy, na obu obrazach i ustala zgodność między cechami wykrytymi na dwóch obrazach. Następnie jeden z obrazów zostaje przekształcony i dopasowany do drugiego tworząc panoramę. Efekt został przedstawiony na obrazie (c).



(a) - osobne obrazy



(b) – znalezione i dopasowane punkty wspólne



(c) – połączone obrazy w panoramę

*Rysunek 3 Przedstawienie kroków obrazu panoramicznego*

W każdym etapie należy zdecydować, które funkcje będą najskuteczniejsze. Warto pamiętać, by wybrany algorytm wykrywał te same cechy kluczowe na obu obrazach, przy czym każdy z obrazów może być odmiennie zaszumiony. W tej fazie poprzez ekstrakcję cech szczególnych klatki wyłania się punkty kluczowe, niezmiennicze bądź krawędzie obiektów służące do dopasowania obrazów. Po dopasowaniu obrazów następuje fuzja właściwa powodująca scalenie obrazu w panoramę. Warto zaznaczyć, że wyekstraktowane cechy mogą posłużyć w kolejnych etapach do identyfikacji obiektów.

Poniżej zostały omówione różne algorytmy tworzenia panoramy bazujące na punktach kluczowych, charakterystycznych. Są to metody najpopularniejsze mające największe

znaczenie w rzeczywistości. Można również wspomnieć o metodzie przedstawionej w artykule [7]. W skrócie metoda polega na podzieleniu obu obrazów na niewielkie obszary i dopasowaniu obrazów do siebie za pomocą szybkich, mało dokładnych metod z wykorzystaniem miary statystycznej korelacji. Następnie używając metod bardziej złożonych obliczeniowo, ale dokładniejszych łączy się segmenty w całość i znajduje najlepsze globalne dopasowanie. Autorzy algorytmu w opisie przedstawiają, że algorytm wykorzystuje metodę quasi-Newtona, z piramidą obszarów. Takie łączenie obrazów pozwala na większą eliminację błędów.

### Metody bazujące na punktach charakterystycznych

Metody łączenia obrazów bazujące na wspólnych punktach kluczowych są najbardziej popularnymi metodami znajdującymi zastosowanie w tworzeniu panoramy np. z obrazów zrobionych smartphonem, ponieważ posiadają dużą odporność na zakłócenia. Wyłanianie takich punktów bazuje na ekstrakcji cech obrazu. W literaturze punkty charakterystyczne nazywane są również punktami, kluczowych bądź kontrolnymi CP (ang. Control Point). Cechy takie rozumiane są jako specyficzne konfiguracje pikseli układające się w struktury. Przykładami takich konfiguracji mogą być: zakończenia linii, krawędzie lub kąty. Wykrywanie struktur zazwyczaj następuje za pomocą filtracji obrazu. Fragmenty obrazu, które nie zmieniają się podczas przekształceń obrazu nazywane są punktami kluczowymi. Ważnym elementem punktów kluczowych jest to ich skalo-niezmienniczość.

#### Detektor Harris'a

Jako punkty charakterystyczne można wykorzystać narożniki wykryte na obrazach. Detektor Harris'a, który jest ulepszoną wersją detektora Moraveca z powodzeniem wykrywa narożniki. Koncepcja detektora Moraveca polega na przeszukiwaniu obrazu z wykorzystaniem okna przeszukiwania ROI (ang. Region of Interest). Gdy podczas przeszukiwania w jednym kierunku zostanie zauważona duża zmiana w jasności pikseli obszar zostaje zakwalifikowany jako krawędź. Jeżeli zmiana zostanie również zauważona w kierunku prostopadłym to obszar kwalifikuje się jako narożnik. Zmiany intensywności sprawdza się co  $45^\circ$  co jest istotną wadą. W detektorze Moraveca wadami są również zaszumiona odpowiedź z uwagi na binarną funkcję okna oraz minimum jako kryterium. Chcąc poprawić algorytm Chris Harris w 1988 roku zaproponował ulepszoną wersję algorytmu z powodzeniem stosowaną do dnia dzisiejszego. W detektorze Harris'a wykorzystywana jest macierz autokorelacji w postaci:

$$M = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 e^{\frac{-(x^2+y^2)}{2\sigma^2}} & \left(\frac{\partial I}{\partial x}\right)^2 \left(\frac{\partial I}{\partial y}\right)^2 e^{\frac{-(x^2+y^2)}{2\sigma^2}} \\ \left(\frac{\partial I}{\partial x}\right)^2 \left(\frac{\partial I}{\partial y}\right)^2 e^{\frac{-(x^2+y^2)}{2\sigma^2}} & \left(\frac{\partial I}{\partial y}\right)^2 e^{\frac{-(x^2+y^2)}{2\sigma^2}} \end{bmatrix}$$

Równanie 1 Macierz autokorelacji detektora Harris'a

gdzie  $I$  jest funkcją intensywności, a  $\sigma$  jest odchyleniem standardowym funkcji Gaussa. Jeżeli różnica między wyznacznikiem macierzy  $M$ , a kwadratem jej śladu przemnożonym przez stałą  $k$  ( $P(x, y) = \det(M) - k(\text{trace}(M))^2$ ) będzie mniejsza od zera  $P(x, y) < 0$  to punkt  $P(x, y)$  jest uważany za krawędź. Jeżeli zaś  $P(x, y) > 0$  to punkt  $P(x, y)$  uważany jest za narożnik. Jeżeli punkt jest bliski zera  $P(x, y) \sim 0$  to uważa się, że obszar nie posiada znaczących zmian. Dodatkowo dla większej dokładności algorytmu wykrywania narożników wprowadza się próg  $P(x, y) > T$ . Zaleca się również wybrania punktów z lokalnym maksimum. Detektor Harris'a wykrywa narożniki z wiele większą dokładnością niż detektor Moraveca oraz sprawdza



narożniki pod każdym kątem. Algorytm może z powodzeniem zostać wykorzystywany do tworzenia panoram.

### *GFTT*

Algorytm GFTT (ang. - Good Features To Track) jest kolejnym ulepszeniem jakie wprowadzili Jianbo Shi oraz Carlo Tomasi w 1994 roku dla detektora Harrisa. Na potrzeby algorytmu wymaga się by obraz wejściowy był czarno biały. Zmianie w odniesieniu do oryginalnego algorytmu polegała na zastosowaniu innego wzoru wyliczającego funkcje oceniającą. Zamiast jak w oryginale  $P(x, y) = \det(M) - k(\text{trace}(M))^2$  w przypadku GFTT funkcja ta ma postać

$$P(x, y) = \min\left(\left(\frac{\partial I}{\partial x}\right)^2 e^{-\frac{(x^2+y^2)}{2\sigma^2}}, \left(\frac{\partial I}{\partial y}\right)^2 e^{-\frac{(x^2+y^2)}{2\sigma^2}}\right)$$

*Równanie 2 Funkcja oceniająca dla algorytmu GFTT*

Dzięki swojej modyfikacji algorytm znajduje najbardziej użyteczne wierzchołki obrazu. Każdy wykryty wierzchołek sprawdzana się czy przekroczył pewien próg zmiany jasności i jeżeli tego nie zrobił zostaje automatycznie odrzucony. Dodatkowo określa się minimalny dystans pomiędzy wierzchołkami. Kryteria pozwalają wyłonić najlepiej procentujące wierzchołki, czyli interesujące punkty kluczowe.

### *FAST*

Detektor FAST (ang. Features from Accelerated Segment Test) w swoim kryterium porównuje jasność piksela z jasnością pikseli oddalonych o ustalony promień. W przypadku gdy jasność większości pikseli będzie się różnić od piksela centralnego o określoną wartość to punkt kwalifikowany jest jako narożnik. Przykładowo dla promienia równego 3 piksele, jeżeli 12 z 16 pikseli będą jaśniejsze niż piksel centralny pomniejszony o pewną wartość to algorytm wykrył narożnik. Na potrzeby algorytmu promień powinien być dostatecznie mały (od 2 do 6 pikseli) oraz wartość progu kwalifikacji i ilości pikseli odpowiednio przeskalowana do promienia. Algorytm FAST osiąga przeciętne wyniki w wykrywaniu punktów kluczowych, lecz jego zaletą jest szybkość.

### *SIFT*

Algorytmem, który wykorzystuje punkty charakterystyczne na którego należało by zwrócić szczególną uwagę jest zaproponowany przez David Lowe algorytm SIFT (ang. Scale Invariant Feature Transform) [8]. Jest to jeden z bardziej popularnych algorytmów, dlatego jego działanie zostało przybliżone w tym akapicie. Algorytm SIFT realizowany jest w czterech krokach. Kroki zostały wpięrow przedstawione, a następnie szczegółowo opisane. Pierwszy krok nazwany „scale space extrema detection” wykrywa punkty ekstremalne na dwóch obrazach. W kroku drugim „accurate keypoint location” następuje dokładna lokalizacja punktów charakterystycznych. Następnie przypisuje się orientację wykrytym punktom w przestrzeni. Krok nazwany jest „keypoint orientation assignment”. Czwarty etap zawiera tworzenie deskryptorów dla punktów charakterystycznych. Szczegółowe omawianie algorytmu należy zacząć od procesu skalowania obrazu wejściowego do różnych wielkości. Dla każdego obrazu w skali zostaje użyty filtr Laplace’a, który pozwala uzyskać kontury obiektów  $D_n$ , obliczany za pomocą odjęcia dwóch obrazów rozmytych filtrem Gaussa z różnymi parametrem  $\sigma$ . Zazwyczaj większe rozmycie obrazu tworzy się poprzez zwiększenie potęgi, do której zostanie podniesiona stała  $k$ . Dla ułatwienia zrozumienia poniżej opisano wzór

$$D_n = \frac{1}{2\pi(\sigma k^{n+1})^2} e^{\frac{-(x^2+y^2)}{2k^2\sigma^2}} \cdot P(x, y) - \frac{1}{2\pi(\sigma k^n)^2} e^{\frac{-(x^2+y^2)}{2k^2\sigma^2}} \cdot P(x, y)$$

Zazwyczaj w za parametr odchylenia standardowego przyjmuje się  $\sigma = 1,6$ . Stała  $k = \sqrt{2}$  podnoszona do kolejnych potęg. Następnie przez binaryzację wykrywa się lokalne maksima i minima. Taka operacja powoduje wykrycie ogromnej ilości punktów ekstremalnych, dlatego należy zastosować dwa kryterium, które odfiltrują najlepsze możliwe punkty charakterystyczne. Pierwsze kryterium odrzuca płytkie minima bądź niewielkie maksima za pomocą rozwinięcia funkcji w szereg Taylora. Dzięki temu odrzucone punkty, które powstały na obszarze np. nieba. Drugie kryterium sprawdza czy punkt nie leży na odcinku za pomocą detektora Harris'a. Po tym etapie wyłonięone zostają punkty niezmiennicze. Algorytm przypisuje punktom orientację co powoduje niezmienniczość również względem orientacji. Można więc ustawić obrazy tak by odpowiadające sobie punkty charakterystyczne na każdym obrazie wskazywały jednakowy kierunek. Realizowane jest to poprzez wyznaczenie gradientu w punkcie charakterystycznym. Następnie wyznacza się gradienty w małym otoczeniu punktu charakterystycznego i z odpowiednio wyważonych gradientów tworzy się histogram. Z histogramu powstaje deskryptor, który pozwala wyznaczyć orientację obszaru przetwarzanego w końcowym etapie. Etap końcowy polega na tworzeniu ostatecznych deskryptorów. Najczęściej przetwarzaniu podlega rozmyty wejściowy obraz. Etap przypomina ten poprzedni, lecz otoczenie punktu charakterystycznego jest dużo większe i podzielone na cztery obszary. W każdym obszarze wyznacza się osobne deskryptory co kończy algorytm. Uzyskuje się w ten sposób punkty charakterystyczne wraz z dokładną orientacją.

Chcąc zakończyć tematykę tworzenia panoramy należało by wspomnieć w jaki sposób za pomocą metod klasycznych oraz tych bazujących na punktach kluczowych zostaje utworzony jednolity obraz panoramiczny. Nietrudno zauważyć, że w każdej z metod wykryte cechy pozwalają dopasować obrazy do siebie. Następnie nakłada się jeden obraz na drugi w taki sposób by występowała największa liczba pasujących elementów np. wykrytych linii, krawędzi, narożników czy deskryptorów. Obrazy zostają scalone w jeden tzn. dokonuje się ich fuzji. Zazwyczaj scalenie polega na przysłonięciu części obrazu drugim obrazem. Istnieją również prace, które starają się wspólny obszar zapełnić informacjami z obu obrazów, lecz takie rozwiązanie może spowodować pogorszenie się jakości obrazu. Tworzenie panoramy na potrzebę systemu wykrywania znaków zostało zaimplementowane i przedstawione w akapicie Propozycja wykorzystania większej ilości kamer.

## Segmentacja obrazów

Niezależnie od tego w jaki sposób będzie rejestrowany obraz pierwszym etapem rozpoznania na nim znaków będzie znalezienie ich położenia na obrazie. Segmentacja obrazów polega na znalezieniu obiektów na obrazie i podzieleniu go według nich. Zazwyczaj na obrazie wyszukuje się tylko obiekty najistotniejsze ze względu na badany problem. W przypadku systemu TSR segmentacja obrazów polega na znalezieniu na obrazie znaku drogowego. Wykrywanie obiektów na obrazach może się odbywać na wiele sposobów. Kilka metod wykrywania i rozpoznawania obiektu zostały przedstawione poniżej. Wpierw zostało omówione Wykrywanie znaków na podstawie koloru. W kolejnym akapicie omówiono Wykrywanie na podstawie kształtu. Następnie przedstawiono Wykrywanie na podstawie tekstury. Jedne z bardziej skutecznych jest Podejście hybrydowe próbujące łączyć ze sobą podejścia na podstawie koloru, kształtu i tekstury. Na sam koniec zostały Podejście hybrydowe oraz inne podejścia stosowane do problemu rozpoznawania znaków drogowych.

### **Wykrywanie znaków na podstawie koloru**

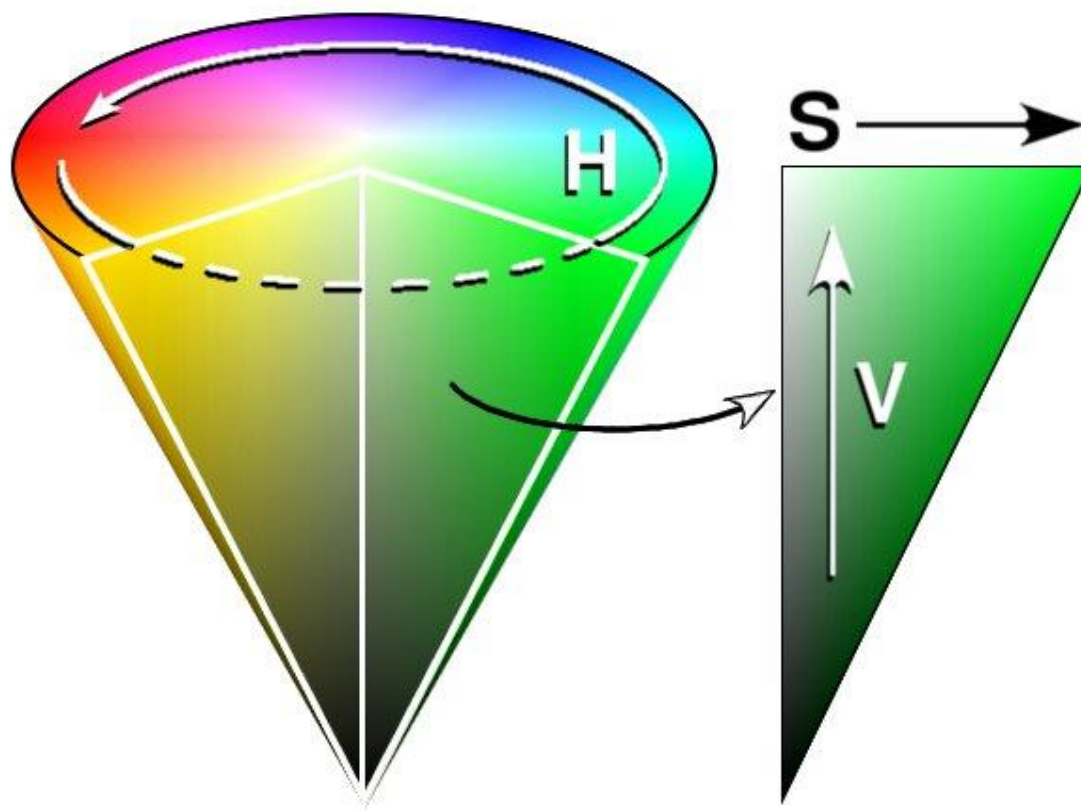
Wykrywanie obiektów na podstawie koloru najczęściej stosuje się w przypadku kolor obiektu jest nieskomplikowany i dobrze określony. By wykrywanie na podstawie koloru mogło się odbyć, obiekt, który chcemy wykryć powinien posiadać z góry określone kolory. Znaki drogowe posiadają taką własność oraz projektowane są w taki sposób by ich kolor znacznie odróżniał się od tła. Najpopularniejszą metodą wykrywania obiektów na podstawie koloru jest progowanie obrazu w określonej barwie. Zaletą progowania jest jego szybkość. Ważnym jest by kolor znacząco odróżniał się od tła. Znaczna część przedstawionych prac np. [9] [10] wykorzystuje progowanie obrazu w kolorze czerwonym by wstępnie wyszukać regiony, gdzie mogą znajdować się znaki. Określenie koloru znaku może być zrealizowane za pomocą kilku sposobów. Zazwyczaj wykonuje się to ręcznie przez pobranie próbek z kilkunastu wzorcowych obiektów. Następnie wyznaczony zostaje średni kolor i możliwy zakres odchylenia od wzorca. Warto zauważyć, że progowanie jest mało odporne na wszelkiego typu zakłócenia, ponieważ kolor obiektu w dużej mierze zależy od oświetlenia. By zaradzić problemowi zmiany oświetlenia w zależności od warunków część algorytmów wykorzystuje możliwość zmiany przestrzeni barw.

#### *Zmiana przestrzeni barw*

Przestrzeń barw RGB (ang. Red, Green, Blue) jest bardzo zmienna w zależności od warunków panujących na drodze. Pogoda, pora dnia lub mnogość różnokolorowych źródeł światła mogą spowodować zmianę odbieranych przez kamerę kolorów. Część algorytmów przed progowaniem wstępnie przetwarza obraz stosując techniki tj. rozciągnięcie histogramu. Metody te są jednak mało skuteczne, dlatego część naukowców badających temat TSR postanowiło przejść z przestrzeni barw RGB na HSI (ang. Hue, Saturation, Intensity) lub HSV (ang. Hue Saturation Value) [11], [12], [13]. W przestrzeni barw HSI każda z trzech wartości odpowiada za inną składową koloru. Hue opisywany jest za pomocą koła, gdzie jego wycinek odpowiada za dany kolor. Saturation wyznacza nasycenie barwy. Czym wartość saturacji jest mniejsza tym



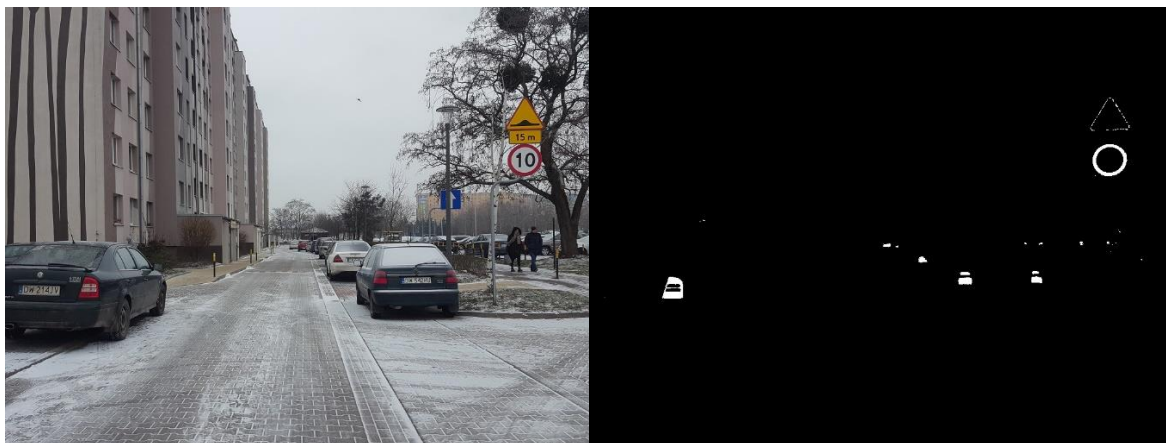
kolor jest bledszy. Ostatnia wartość Value odpowiedzialna jest za jasność koloru. Rysunek 4 graficznie przedstawia trzy składowe przestrzeni HSI.



*Rysunek 4 Składowe przestrzeni barw HSV*

Modele barw HSI i HSV pozwalają w pewnym stopniu zniwelować wahania spowodowane zaburzeniami oświetleniowymi. Niestety nie są one wolne od wad. Nie uwzględniają temperatury barwy. Przestrzeń barw LCH (ang. Lightness, Chroma, Hue) uzyskana za pomocą modelu CIECAM97 pozwala uwzględnić temperaturę barw. Warto wspomnieć również o przestrzeni YUV, która posiada trzy kanały. Jeden kanał Y dostarczający luminację, czyli jasność obrazu i dwa kanały U i V kodującą chrominancję, czyli barwę. Ta przestrzeń zdobyła największą popularność, gdy wprowadzano telewizory kolorowe. Telewizory czarno-białe odbierały jeden kanał, którym przesyłano wartość chrominancji, a nowsze odbiorniki odbierały wszystkie trzy składowe mogąc tym samym odtworzyć obraz kolorowy. Algorytm z pracy [14] bazujący na splotowej sieci neuronowej zaprzęgniętej do problemu lokalizacji znaków wykorzystuje właśnie przejście na taką przestrzeń barw. By opis był pełny należy zaznaczyć, że część autorów w swoich pracach stwierdza, że zmiana przestrzeni barw jest zbędna,

ponieważ unormowana przestrzeń RGB w przeciętnych warunkach jest wystarczająca, a koszt obliczeniowy związany ze zmianą przestrzeni barw jest zbyt duży.



Rysunek 5 Progowanie w przestrzeni barw HSV dla koloru czerwonego

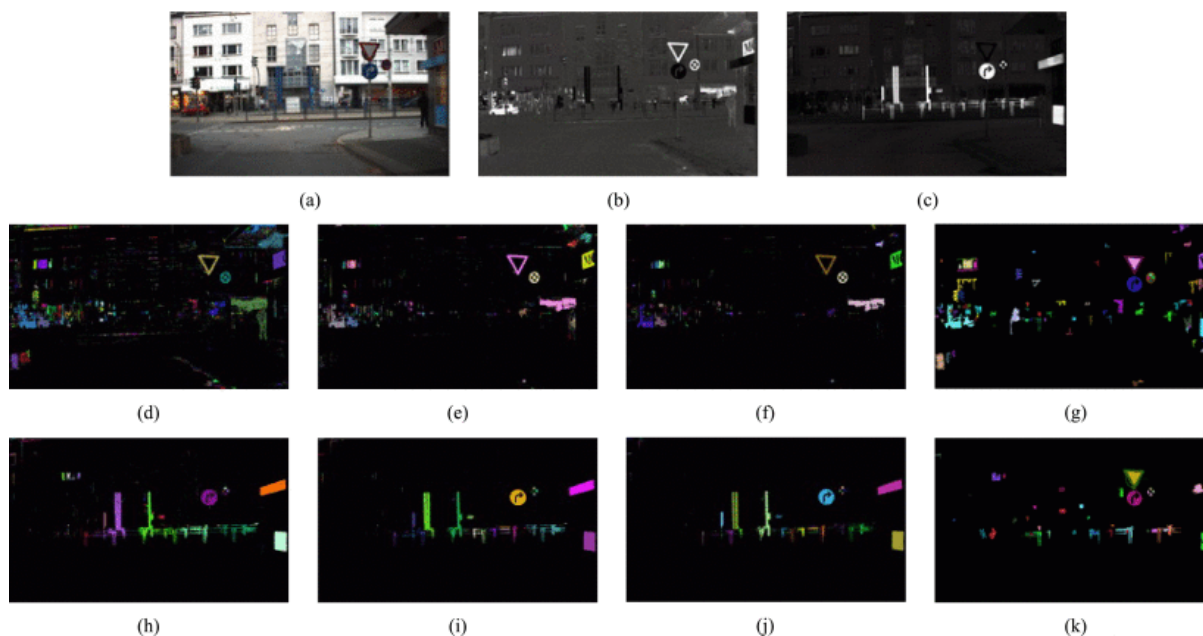
Rysunek 5 przedstawia zdjęcie poddane progowaniu w kolorze czerwonym. Obraz oraz parametry progowania zostały dobrane doświadczalnie. Progowanie uwidocznili znak drogowy, lecz również inne elementy. Na takim obrazie albo wyszukuje się figur, które najbardziej są zbliżone do znaków drogowych np. za pomocą metody nazwanej Transformata Hougha opisanej w znajdującym się nieco niżej akapicie lub można starać się odfiltrować elementy niewłaściwe. Drugie rozwiązanie wprowadzili autorzy pracy [15].

#### *AdaBoost*

AdaBoost to oparty na boostingu algorytm zaprezentowany 1996 roku przez Yoav Freund i Robert Schapire [16]. Główna idea polega na stworzeniu z wielu słabych klasyfikatorów jednego silnego klasyfikatora. Ciekawe rozwiązanie przedstawiono w pracy [10], w której wyszukanie znaków opiera się na kolorach, ale nie na samym progowaniu. Algorytm używa zagnieżdżonej kaskady klasyfikatora Real AdaBoost, którą szkoli się za pomocą cech LRP (ang. Local Rank Pattern). Do obliczenia cech LRP wykorzystuje się siedem typów obrazów, gdzie każdy obraz tworzy się przez odjęcie jednego typu koloru (np. czerwonego, niebieskiego zielonego itp.) W ogólności algorytm AdaBoost jest bardzo popularny i został wykorzystany w większej liczbie prac.

#### *Model prawdopodobieństwa barwy*

Algorytm korzystający z modelu prawdopodobieństwa barw do wykrywania znaków drogowych został przedstawiony w pracy [17]. Algorytm tworzy mapę prawdopodobieństwa występowania znaku drogowego na podstawie poszukiwania kolorów znaków drogowych w obrazie. Następnie w regionach, gdzie jest najwięcej poszukiwanych kolorów algorytm stara się wykryć znak. Dodatkowo poszukiwanie koloru następuje w przestrzeni kolorów Otha, ponieważ taka przestrzeń lepiej sprawdziła się w eksperymentach prowadzonych dla pracy. Skanowanie w przestrzeni Otha zostało przedstawione na Rysunek 6.



Rysunek 6 Tworzenie mapy prawdopodobieństwa w przestrzeni Otha

Na wyżej pokazanym Rysunek 6 obraz (a) jest wejściowym obrazem w kolorze. Obrazy Rysunek 6(b) i (c) są mapami prawdopodobieństwa uzyskanymi odpowiednio dla kolorów czerwonego i niebieskiego. Jak można zauważyć, czerwone i niebieskie piksele na oryginalnym występują w dużej ilości i mają spore natężenie. Zastosowanie map prawdopodobieństwa ułatwia wykrywanie znaków przez zwiększenie kontrastu między znakami drogowymi, a tłem. Aby algorytm pracował w czasie rzeczywistym, obliczenia zostały przyspieszone za pomocą wprowadzenia tabeli wyszukiwania LUT (ang. Look Up Table). Podczas wykrywania obliczony zostaje indeks każdego piksela według wartości RGB, a następnie znajduje odpowiadające mu prawdopodobieństwo w tabeli LUT. Autorzy algorytmu twierdzą, że po wprowadzeniu tabeli czas obliczenia map prawdopodobieństwa dla obrazu w rozdzielczości  $1360 \times 800$  może zostać zredukowany z kilku minut do około 30 ms komputerze czterordzeniowym procesorem Intel 3,4 GHz, 4 G RAM. Dodatkowo autorzy wykorzystali detektor regionów MSER (ang. Maximally Stable Extremal Regions) aby na mapach prawdopodobieństwa zlokalizować miejsca występowania znaków. Działanie detektora MSER przedstawione zostało na obrazach od (d) do (f) dla progowania w kolorze czerwonym i od (h) do (j), dla koloru niebieskiego. Obrazy (g) i (k) sumę obrazów dostarczonych przez detektor. Na obrazach (g) i (k) następuje wykrycie obszarów ze znakami przez ztworzenie map prawdopodobieństwa. Po wykryciu regionów ze znakami następuje klasyfikacja regionów do poszczególnych podklas poprzez maszynę wektorów nośnych przeszkolonych dla cech HOG z uwzględnieniem koloru. Algorytm HOG został przedstawiony w podrozdziale opisującym wykrywanie znaków na podstawie tekstury, dlatego w tym miejscu nie został opisany.

### Wykrywanie na podstawie kształtu

Powszechnym podejściem w wykrywaniu obiektów w tym znaków drogowych na obrazach jest wykrywanie poprzez kształt. Podobnie jak w przypadku koloru tak i w przypadku wykrywania na podstawie kształtu ważnym atutem obiektu, który należy zlokalizować musi być konkretnie określony kształt. Jak dla znaków drogowych taki kryterium jest spełnione. Znaki informacyjne są kwadratowe, nakazu lub zakazu okrągłe, a ostrzegawcze trójkątne. Wykrywanie na podstawie kształtu nie jest również wolne od wad. Percepcja kształtów różni się od kąta

obserwowania. Innym problemem jest fakt, iż podczas obserwacji znaki mogą być częściowo przysłonięte lub na ich część może padać cień, co zmienia ich wygląd dla większości algorytmów. Ważnym jest by zastosowany detektor radził sobie z takimi problemami. Wykrywanie figur geometrycznych na obrazie poprzez kształt może być zrealizowane na kilka sposobów. Te ważniejsze dla pracy zostały przedstawione poniżej.

### *Wykrywanie krawędzi*

Wykrywanie krawędzi jest popularną metodą uwidaczniania obiektów na obrazie. W tym miejscu należy wyjaśnić, że krawędź identyfikowana jest jako przejście z obszaru ciemniejszego do jaśniejszego bądź na odwrót, jest to granica pomiędzy dwoma obszarami o różnych jasnościach. Takie podejście wymaga ustalenia konkretnego progu zmiany jasności powodującego wykrycie. Część metod bazuje na operatorach gradientowych ustalając próg lokalny, zamiast globalnego. Takie metody wykorzystują zmiany pierwszej lub drugiej pochodnej obrazu w skali szarości. Powszechnym rozwiązaniem wykrywania krawędzi jest Detektor Canny, który został omówiony poniżej. Istnieją również inne metody. Przykładami takich metod jest Krzyż Robertsa oraz Operatory Sobela. Oba algorytmy sprawdzają różnicę między sąsiednimi pikselami w obrazie przez splot pewnej macierzy z obrazem. Algorytmy bazujące na takich operatorach przeznaczone do problemu TSR przedstawione zostały np. w pracy [18].

### *Detektor Canny*

Jak wcześniej zostało wspomniane jedną z popularniejszych metod wykrywania krawędzi jest metoda Canny zaprezentowana w Johna F. Canny w 1986. Metoda przedstawia się następująco. Wpierw następuje filtracja obrazu za pomocą filtru Gaussa. Powoduje to wstępnie rozmycie obrazu i odfiltrowanie małych zakłóceń. Następnie za pomocą np. operatora Sobela wykryty zostaje gradient zmian poziomych  $G_x$  i pionowych  $G_y$  każdego punktu na obrazie. W kolejnym kroku wyznacza się na podstawie wykrytych gradientów dwa parametry:

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = \arctan\left(\frac{G_y}{G_x}\right).$$

*Rysunek 7 Parametry krawędzi dla algorytmu Canny*

Wartość  $G$  oznacza długość, a  $\theta$  kąt detekcji krawędzi. Niekiedy kąt zaokrągla się do wartości liczonych co  $45^\circ$ . Piksele, które nie łączą się z żadną pobliską krawędzią zostają odrzucone. Piksele umieszczone blisko zakończeń wykrytych krawędzi zostają połączone w jedną krawędź.



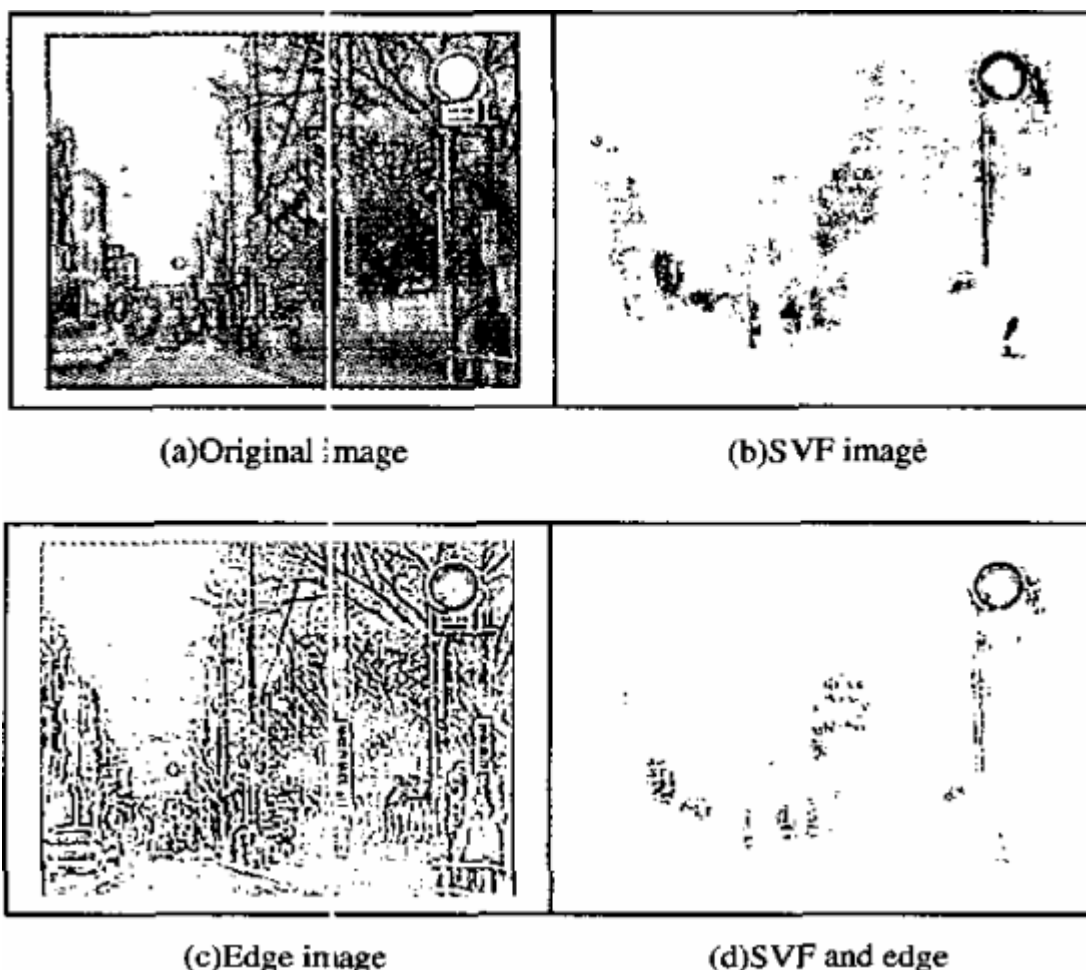
*Rysunek 8 Detekcja krawędzi poprzez algorytm Canny*



Rysunek 8 przedstawia obraz uzyskany za pomocą detekcji krawędzi Canny. Na tak przetworzonym obrazie należy odszukać obiekty, które nas interesują. Dla problemu TSR są to: trójkąty, kwadraty i okręgi. Detektor Canny dzięki swoim właściwościom doskonale nadaje się do uwidaczniania kształtów, dlatego jest to algorytm, po który najczęściej się sięga, gdy wymagane jest wykrycie konturów.

#### *SVF*

SVF (ang. Simple Vector Filter) jest algorytmem pozwalającym w pewnym stopniu odfiltrować z obrazu, na którym wykryto krawędzie nieistotne elementy. Metoda została zaproponowana w artykule [12]. W podanej pracy w pierwszej kolejności następuje wyodrębnienie kształtów przez zastosowanie filtra krawędziowego bazującego na zmianie jasności. Niestety takie działanie wyodrębnia dużo niepożądanych elementów sceny. Należało więc z wszystkich znalezionych elementów zostawić te najbardziej podobne do znaków, a resztę usunąć. Największym problemem są obiekty, których kształt zbliżony jest do kształtu znaków. By poradzić sobie z tą przeszkodą autorzy zaproponowali autorski filtr SVF uwzględniający również kolor znalezionej obiekty. Zaletą proponowanego filtra jest jego szybkość. Dokładny opis przebiegu algorytmu jest bezcelowy, ponieważ można go przeczytać w podanej pracy. Skuteczność metody przedstawiono na Rysunek 9. Na obrazie (a) pokazano obraz oryginalny przekonwertowany do skali szarości i zbinaryzowany. Obraz (b) przedstawia działanie filtra SVF nałożonym na obraz (a). Na obrazie (c) uwidoczniono krawędzie z obrazu (a). Ostatni obraz (d) przedstawia zastosowanie filtra SVF na obrazie (c). Jak można zauważyć metoda dobrze sprawdza się do filtracji i może poprawić wykrywanie znaków przy wykrywaniu na podstawie kształtu.



Rysunek 9 Wykrywanie znaków z zastosowaniem filtra SVF

#### *Wykrywanie rogów znaków jako punktów charakterystycznych*

Wykrywanie rogów może być przydatne zwłaszcza dla znaków ostrzegawczych (trójkąty) i informacyjnych (kwadraty). Wykrywanie narożników może odbywać się np. za pomocą detektora Harris'a opisanego w wcześniejszym akapicie zatytułowanym Detektor Harris'a. Jeżeli nastąpi wykrycie rogów w konkretnej konfiguracji adekwatnej dla znaku można domniemywać, że w tym miejscu wystąpił znak i przystąpić do klasyfikacji znaku.

#### *Transformata Hougha*

Algorytmem, na który należało by również zwrócić szczególną uwagę jest transformata Hougha. Pozwala ona odnajdywać regularne kształty na obrazie. Została zaprezentowana w 1962 roku jako metoda wykrywania linii prostych, lecz udało się go zastosować dla wykrywania innych kształtów takich jak okręgi, kwadraty czy trójkąty [19]. W 1993 roku Anagnou przedstawił udoskonaloną metodę pozwalającą uzyskanie większej rozdzielczości kątowej oraz radialnej znajdowanych linii. Omówienie metody zostało zrealizowane na przykładzie wykrywania kształtu kwadratowego. W tej metodzie każda prosta opisana jest równaniem  $\rho = x \cos \theta + y \sin \theta$ . Początkowo zostaje zaimplementowana tablica (komórki nazywa się akumulatorami) z wartościami zerowymi. Następnie każdy punkt obrazu przekształcany jest w dyskretną krzywą sinusoidalną w przestrzeni  $\rho \times \theta$ . Oblicza się wartości parametru  $\rho$  dla kątów w zakresie  $\Theta = (0, 360^\circ)$ . Obliczenie następuje przez dodanie wartości



jeden do liczby oznaczającej wartość odpowiedniego akumulatora. Po wykonaniu inkrementacji dla wszystkich punktów obrazu, wybierane zostają akumulatory z największą wartością  $\rho$ . Położenie komórki określa położenie i kąt odcinka zaś wielkość  $\rho$  jego długość. Według podobnej zasady można również wykryć okręgi na obrazie. Transformacja Hougha odnosi sukcesy w wykrywaniu linii czy okręgów, lecz do wykrycia innych kształtów jak np. trójkąty wymaga skomplikowanych obliczeń i jest obciążająca pamięciowo. Metody transformaty Hougha użyto np. w pracy [15] do rozróżnienia typu znaku. Praca ta została bliżej przedstawiona w rozdziale klasyfikacji znaków i metodzie Extreme Learning Machine.

### **Wykrywanie na podstawie tekstury**

Podczas wykrywania obiektów na podstawie koloru i kształtu uzyskanie dobrych wyników może być problematyczne, dlatego zaczęto szukać innych rozwiązań mogących poprawić skuteczność wykrywania. Rozwiązaniem mające duży potencjał było wykrywanie na podstawie tekstury. Omawiając je należało wyjaśnić, że tekstura to graficzny odpowiednik faktury. Faktura obiektów w rzeczywistości bardzo wpływa na postrzeganie ich przez człowieka, dlatego część grup zajmujących się odnajdywaniem obiektów na obrazie podejmowało próby konstruowania algorytmu bazujących na teksturze. Kilka takich metod zostało przedstawione poniżej.

#### *Histogram gradientów zorientowanych*

HOG (ang. Histograms of Oriented Gradients) - histogram gradientów zorientowanych jest deskryptorem obrazu pozwalającym na znalezienie kształtu i wyłonienie obiektu [20]. Idea algorytmu polega na podzieleniu obrazu przekonwertowanego do skali szarości na małe fragmenty (np. 8x8 pikseli) i obliczeniu dla każdego piksela w takim fragmencie różnicy jasności (gradientu) pomiędzy pikselami sąsiadującymi. Następnie dla każdego z fragmentu wyznacza się histogram gradientu. W przeciwieństwie do wcześniej przedstawionego algorytmu SIFT, HOG oblicza deskryptory w równomiernie oddalonych fragmentach obrazu. Zastosowanie lokalnej normalizacji kontrastu w nakładających się na siebie regionach poprawia skuteczność wykrywania obiektów [21]. Zaproponowany w pracy [22] wariant HOG (HOGv) posiada dwie modyfikacje poprawiające wykrywanie. Pierwszą z nich jest uwzględnienie zarówno wrażliwych jak i niewrażliwych na kontrast orientacji gradientów, w taki sposób, że bardziej szczegółowe lokalne informacje o znakach mogą być włączone do zgromadzonych histogramów. Drugą modyfikacją jest to, że każdą komórkę normalizuje się odpowiednio z czterema sąsiednimi blokami. Znormalizowane histogramy komórki są wymiarowo zmniejszane za pomocą strategii podstawowej analizy komponentów (PCA). Takie działanie ma na celu usunięcie nadmiarowych informacji.

#### *Lokalne wzorce binarne*

LPB (ang. local binary patterns) zostało opisane w 1994 roku jako algorytm wykrywania obiektów na obrazie [23]. Algorytm odniósł duży sukces w wykrywaniu twarzy, dlatego starano się go zaimplementować do wykrywania innych obiektów. Jego idea jest podobna do algorytmu HOG, ponieważ na początku obraz zostaje podzielony na mniejsze obszary. Następnie porównuje się, każdy piksel z wszystkimi pikselami sąsiadującymi w ustalonej kolejności. W zależności czy wartość piksela porównywanego jest większa lub równa czy mniejsza od piksela centralnego w osobnej tablicy zostaje wpisana odpowiednio wartość 1 lub 0. Takie rozwiązanie niesie ze sobą zaletę, że gdy zostaje zmieniona globalna jasność obrazu, wartości tablic [0,1, (...)] się nie zmieniają i skuteczność algorytmu nie spadnie. Następnie podobnie jak w algorytmie HOG w fragmentach zostaje utworzony histogram, a następnie zostaje ustalony gradient całego

okna. Po wyznaczeniu gradientów w każdym oknie następuje połączenie podobnych gradientów mogących tworzyć jeden obiekt. Jeżeli odnalezione pole wyznaczone przez gradienty jest podobne do jednego z obiektów przechowywanych w bazie algorytm kończy działanie. LBP dla wykrywania znaków drogowych zaproponowano w pracy [15]. Rozwiązanie wzbogacone jest o ustalenie na początku działania potencjalnych obszarów poszukiwań znaków za pomocą okna przesuwanego o niewielkich rozmiarach. Etap nazywa się filtrowaniem zgrubnym i używa algorytmu LDA. Następnie podczas dokładnego filtrowania weryfikuje się okna odnalezione w etapie poprzednim za pomocą metody NMS (ang. Non-Maximal Suppression) wykonywane jest filtrowanie okien mogących odnosić się do tego samego obszaru. Po odnalezieniu obszarów, gdzie możliwe jest wystąpienie znaków drogowych ich klasyfikacja wykonywana jest za pomocą maszyny wektorów nośnych. Maszyna ta została opisana w następnym rozdziale.

### **Podejście hybrydowe oraz inne podejścia**

Podejście hybrydowe w ogólności polega na połączeniu metod opartych na kolorze, kształcie i fakturze. W większości prac na początkowym etapie następuje progowanie w określonych kolorach co zawęża obszar poszukiwań, a następnie wykorzystuje się podejścia oparte na kształcie i fakturze w celu poprawy wydajności wykrywania.

#### *Detekcja krawędzi wzbogacona o Prosty Filtr Wektorowy*

Przykładem metody hybrydowej stworzonej do segmentacji obiektów na obrazie jest metoda bazująca na wykryciu krawędzi oraz późniejszym zastosowaniu filtru z SVF (ang. Simple Vector Filter). Metoda ta została zaproponowana przez autorów artykułu [12]. W podanej pracy w pierw obraz kolorowy dostarczony przez kamerę konwertowano się do skali szarości. Następnie starano się wyodrębnić kształty obiektów poprzez progowanie obrazu albo detekcję krawędzi poprzez metodę bazującą na zmianie jasności. Niestety, takie działanie wyodrębnia wszystkie obiekty sceny, dlatego należało odróżnić znaki drogowe od tła. Największym problemem były obiekty, których kształt zbliżony był do kształtu znaków. By poradzić sobie z tą przeszkodą autorzy zaproponowali autorski filtr SVF uwzględniający kolor znalezionej obiektu. Zaletą proponowanego filtru jest jego szybkość. Dokładny opis przebiegu algorytmu można go doczytać w podanej pracy. Skuteczność SVF w odfiltrowaniu znaków drogowych od reszty obiektów przedstawiono na Rysunek 10.



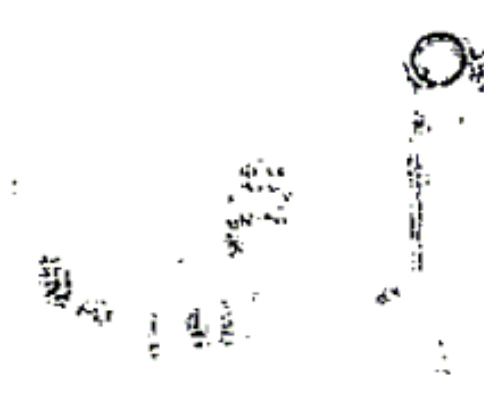
(a) - Zbinaryzowany



(b) - (a) z SVF



(c) – po wykryciu krawędzi



(d) - (c) z SVF

Rysunek 10 Wykrywanie z zastosowaniem filtra SVF

Obraz (a) przedstawia obraz, który został zbinaryzowany, (b) przedstawia obraz (a) po nałożeniu filtra SVF. (c) oryginalny z wykrytymi krawędziami, a obraz (d) obraz analogicznie jak wyżej obraz (c) po nałożeniu filtra SVF. By sprawdzić czy w wykrytym regionie istnieje znak zastosowano algorytm genetyczny opisany w akapicie jako jeden z przykładów w akapicie Algorytmy Genetyczne.

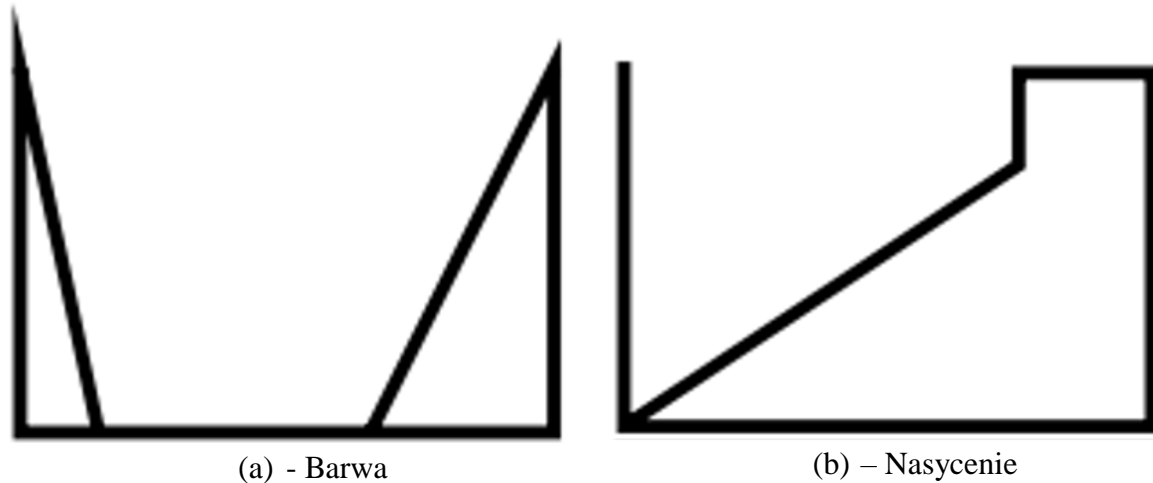
#### *Algorytmy Genetyczne na potrzeby segmentacji*

Algorytmy genetyczne są popularnym podejściem w segmentacji i klasyfikacji znaków drogowych. Działanie GA (ang. Genetic Algorithm) jest iteracyjne iteracyjnie. W każdej iteracji stara się znaleźć lepsze rozwiązanie poprzez krzyżowanie i mutacje znalezionych rozwiązań. Początkowe wartości rozwiązań są losowane ze zbioru możliwych rozwiązań. Algorytm kończy działanie, gdy wyszukane rozwiązanie jest wystarczająco skuteczne.

Rozwiązanie problemu klasyfikacji znaków drogowych z pomocą algorytmu genetycznego przedstawione zostało w wcześniej opisywanej pracy [13] Algorytm skupia się na wgrzaniu znaków okrągłych na obrazach dostarczonych przez filtr krawędziowy i SVF. Oczywiście znaki okrągłe można wyszukać na obrazach przez znalezienie okręgów. W TSR dużym problemem jest dynamicznie zmieniający się promień okręgu i właśnie do znalezienia najlepszego promienia użyto algorytmów genetycznych. W metodzie prezentowanej w artykule każdy osobnik złożony jest z wektora cech. Cechy są liczbami rzeczywistymi o odpowiadającym współrzędnym wyszukiwania i promieniowi znaku. Dodatkowo algorytm zawsze replikuje najlepsze rozwiązanie. W artykule również proponują modyfikacje algorytmu genetycznego nazwaną Step-GA starającą się testować kolejne pokolenia na obrazach wybranych sekwencji filmowej z ustaloną częstotliwością. By poprawić zdolności wykrywania znaków przez algorytm autorzy zaproponowali prostą metodę śledzenia znaku. W większości przypadków podczas jazdy samochodem znak drogowy wykryty na środku ekranu będzie się powiększał i przesunął w okolice prawego górnego rogu. Jeżeli algorytm wykryje obszar, który będzie zachowywał się w taki sposób dopiero wtedy może być on kwalifikowany jako znak drogowy. Takie działanie pozwala odfiltrować dużą część regionów, w których wystąpiły błędy drugiego rodzaju. Jednak mimo dużej ilości zalet metody w trakcie jej publikacji, czyli roku 2002 była dość niedoskonała. Autorzy nie testowali swojego algorytmu na żadnym międzynarodowym benchmarku, dlatego ciężko ocenić skuteczność przedstawionej metody.

Algorytm genetyczny do znajdowania znaków został zaproponowany w pracy [24]. Zaproponowana metoda działa w następujący sposób. Wpierw obraz konwertowany jest z

przestrzeni RGB do przestrzeni HSV. Następnie następuje progowanie w kolorze czerwonym. Autorzy w przeciwieństwie do klasycznej metody progowania nie używają konkretnie określonych wartości progów W celu wybrania obszarów czerwonych zostały przygotowane tablice LUT (Look Up Table). W ten sposób błędy klasyfikacji pikseli spowodowane progowaniem powinny zostać zrekompensowane. Przygotowano dwie takie tablice, dla barwy oraz nasycenia zaprezentowane na Rysunek 11.



Rysunek 11 Tablice LUT

Po splocie każdego piksela z wartościami tablicy obraz zostaje znormalizowany do wartości 255. Dokładna klasyfikacja nie jest konieczna, ponieważ tylko kształty obiektów zamkniętych będą brane pod uwagę. Na tak przygotowanym obrazie zaczyna pracę algorytm genetyczny. Metoda GA musiała zostać zbilansowana pomiędzy przeszukiwaniem całej przestrzeni obrazu, a miejscem, w którym najbardziej prawdopodobne było wystąpienie znaku. Inaczej mówiąc ryzykiem, z którym musi mierzyć się każdy algorytm genetyczny jest zatrzymanie się w lokalnym ekstremum. Należy więc unikać przedwczesnej zbieżności. Jest to odpowiednik utraty genetycznego bogactwa gatunku. Zapis kodu genetycznego przedstawiony został za pomocą Równanie 3 Model znaku w zadanej odległości i prostopadłego do osi optycznych.

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} E_x \cos \theta & E_x \sin \theta & T_x \\ -E_y \sin \theta & E_y \cos \theta & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix}$$

Gdzie:

$T_x$  – przemieszczenie poziome,

$T_y$  – przemieszczenie pionowe,

$E_x$  – skala pozioma,

$E_y$  – skala pionowa,

$\theta$  – rotacja pozioma.

Równanie 3 Model znaku

Rozważanymi modyfikacjami była zmiana pozycji oraz skali znaku spowodowana dystansem, oraz kątem w przypadku, gdy znak nie byłby prostopadły do osi. W klasycznym algorytmie genetycznym, populacja początkowa jest generowana losowo, lecz w tym przypadku możemy przyspieszyć działanie ustawiając populacje początkowe z większym prawdopodobieństwem w miejscach, gdzie zostały odnalezione uwidocznione obiekty na obrazie dostarczonej przez progowanie. Stała liczba osobników jest przyporządkowywana do każdego obiektu. W ten sposób obecność wystarczającej liczby osobników może być zagwarantowana pomimo



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

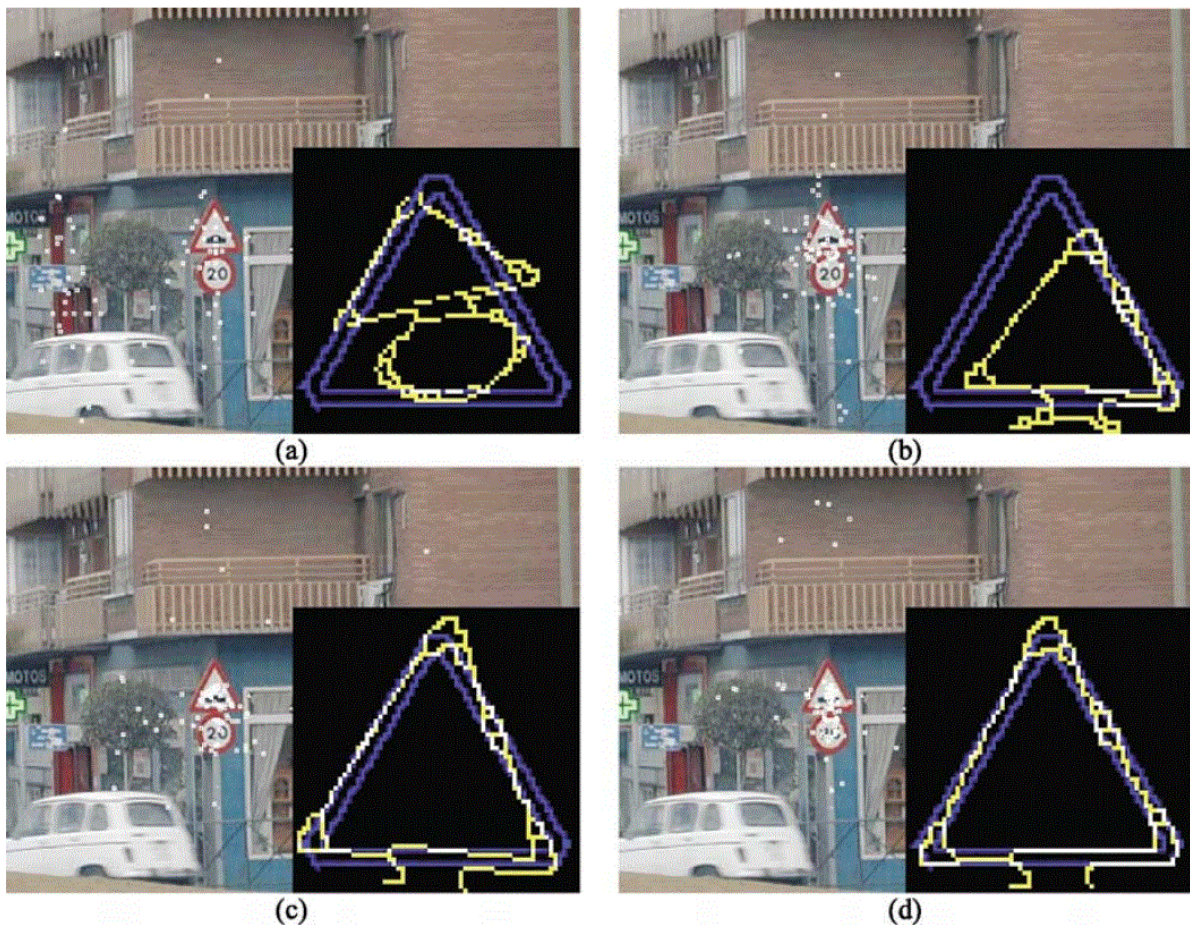
$$H(A, B) = \max(h(A, B), h(B, A))$$

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

$$h_k(A, B) = K_{a \in A}^{th} \min_{b \in B} (a - b)$$

*Równanie 4 Odległość Hausdorffa*

Funkcja  $h(A, B)$  jest to bezpośrednia odległość Hausdorffa. Definiuje ona punkt należący do zbioru  $A$ , który jest najdalej, względem wybranej normy, od każdego z punktów zbioru  $B$ . Wariacja  $A$  jest częściową odległością Hausdorffa, w której tylko  $k$  odległości jest brane pod uwagę. Miara  $h_k$  jest nieczuła na zakłócenia. W opisanym przypadku oba zbiory zawierają krawędzie obiektów wykrytych w poprzednim kroku oraz transformaty zakodowane w chromosomie każdego osobnika. Następnie obliczana jest transformata odległości oraz liczba punktów, których wartość jest mniejsza od progu. Dopasowanie jest relacją pomiędzy liczbą punktów, a całkowitą liczbą punktów modelu. Kolejną zaletą tej funkcji dopasowania jest możliwość zatrzymania tworzenia pokoleń, jeśli wartość jest wystarczająco wysoka.



*Rysunek 12 Cztery wyselekcjonowane pokolenia GA*

Rysunek 12 przedstawia cztery pokolenia algorytmu genetycznego. Obraz (a) przedstawia pierwsze pokolenie. Jak można zauważyć część osobników została zainicjalizowana w miejscu

występowania znaku, lecz część z nich została rozrzucona losowo. Na czarnym tle został na fioletowo przedstawiony wzorec znaku z obrazu, a na żółto najlepsze dotychczas znalezione rozwiązanie. Obrazy (b) i (c) są iteracjami w połowie działania algorytmu, a obraz (d) przedstawia ostatnią iterację. Można zauważyć, że algorytm dobrze dopasował obrys znaku do jego wzorca co pokazane jest na czarnym wycinku na obrazie (d). W każdym pokoleniu wybierani są rodzice, na podstawie których tworzone było kolejne pokolenie. Proces selekcji określony był poprzez wartość dopasowania rozwiązania. W tym procesie geny zawierające dobre rozwiązania były rozszerzane przez populację. Selekcja odbywała się przy pomocy metody ruletki: prawdopodobieństwo wylosowania było proporcjonalne do znormalizowanego dopasowania wszystkich osobników. Dzięki takiemu podejściu najlepsze osobniki miały więcej potomstwa od innych. Czasami metoda ruletki miała problemy z trójkątnymi znakami. Jak wspomniano wcześniej, algorytmy genetyczne są lepsze w wyznaczaniu niż poszukiwaniu, ponieważ mogą zostać uwiecznione w lokalnym ekstremum, z powodu tego, że większość osobników jest skoncentrowana w tej strefie. Trójkątne znaki często doprowadzają do tego zjawiska. Powodem jest to, że kilka osobników może nakładać dwa boki trójkąta na jeden bok modelu. Wygodnie było opóźnić zbieżność, pomimo ryzyka jej spowolnienia, aby zapewnić znalezienie globalnego ekstremum, dlatego właśnie zastosowano metodę rankingową. Jeśli powstały złe osobniki, zostały wyeliminowane w następnym doborze. Jeśli okazały się dobre, ich informacja była propagowana w populacji. Zadowalające wyniki otrzymano dla następujących parametrów:

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

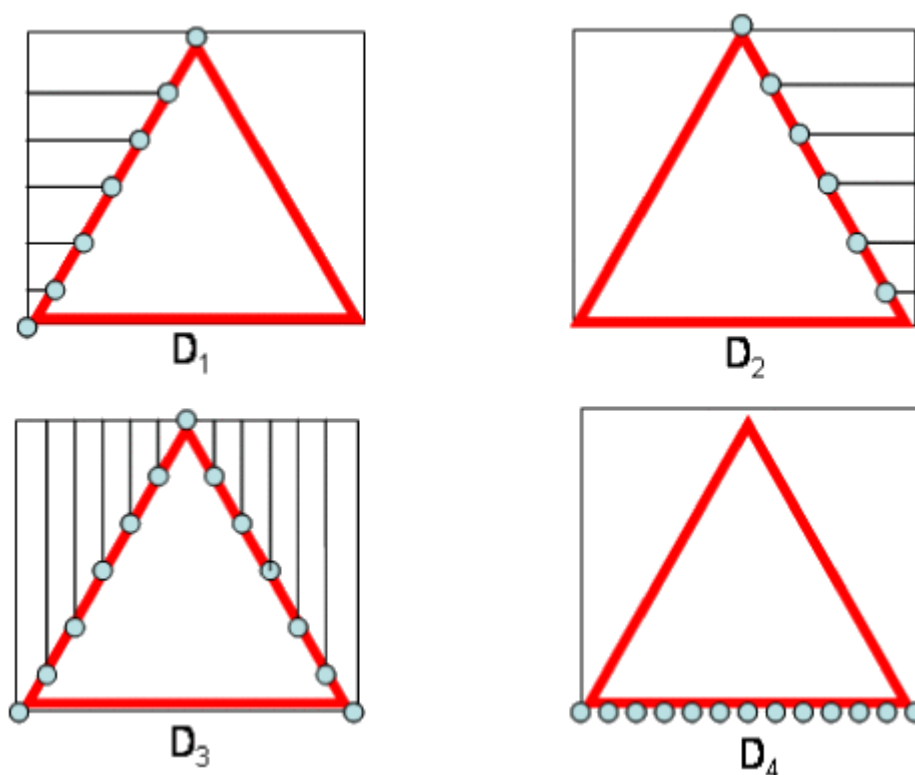
Czas pracy generowania każdego pokolenia wynosi 8.8 ms pracując na procesorze AMD Duron 1 GHz. Po znalezieniu regionów występowania znaku rozpoznanie znaku odbywa się za pomocą sieci neuronowej nauczonej przez obrazy dostarczone w przestrzeni HSI. Warto wspomnieć, że prace dotyczące algorytmów genetycznych w połączeniu z sieciami neuronowymi stosowane były od początku rozważań nad tematem TSR. Przykładem może być przytoczenie pracy [1] z 1996 roku która proponowała takie rozwiązanie.

#### *Maszyna wektorów nośnych*

Maszyna wektorów nośnych SVM (ang. Support Vector Machine) jest klasyfikatorem umożliwiającym określenie do jakiego zbioru należy obiekt, za pomocą jego cech. Do poprawnego działania potrzebuje zbioru danych, zawierającego poprawnie sklasyfikowane



obiekty, po ich cechach. Algorytm działa na zasadzie znalezienia hiperpłaszczyzny w przestrzeni zawierającej cechy obiektów, która oddziela dwie klasy obiektów. Jeżeli taka funkcja nie istnieje w przestrzeni zadania należy wprowadzić funkcję jądra, która powiększy przestrzeń. System, który korzysta z maszyny wektorów nośnych dla problemu TSR został przedstawiony w pracy [11]. Działanie systemu polega na wykrywaniu obszary występowania znaków za pomocą progowania w przestrzeni HSI. Znaki białe wykrywane są za pomocą rozkładu achromatycznego. Każdy obszar prawdopodobnego wystąpienia znaku wpisywany jest w kwadrat. Następnie wyznacza się 20 odległości liczonych od ramki kwadratu do znajdującego koloru w środku ramki mający być obwiednią obiektu pokazanych na Rysunek 13 Odległości tworzące wektor DtB. Z tych 20 odległości tworzy się wektor DtB dostarczany jest na wejście SVM.



Rysunek 13 Odległości tworzące wektor DtBs

Warto wspomnieć, że wyznaczony w ten sposób wektor znaków ośmiokątnych niezbyt różni się od znaków okrągłych, dlatego znaki ośmiokątne klasyfikowane są dopiero na etapie rozpoznawania znaku. Następnie przy użyciu kilku wytrenowanych maszyn SVM następuje klasyfikacja obszaru. Każda maszyna potrafi rozpoznać jeden typ obrazu. W zależności od wykrytego koloru (czerwonego, niebieskiego, żółtego lub białego) algorytm wyznacza inne maszyny do klasyfikacji. Proces rozpoznania opiera się na maszynie SVN z ziarnami Gaussa. Dużą zaletą metody jest jej odporność na rotację.

#### *Histogram gradientów zorientowanych w połączeniu z maszyną wektorów nośnych*

Maszyna wektorów nośnych może być również łączona z algorytmem tworzącym histogram gradientów zorientowanych. Takie podejście zaprezentowane zostało w pracy [17]. W tej pracy wykrywanie obiektów za pomocą metody HOG wspomagane jest informacją o kolorze, co jest ogromnie istotne dla wykrywania znaków drogowych. Oryginalnie algorytm HOG oblicza gradienty dla każdego kanału kolorów i przyjmuje gradient o największej normie, zaś algorytm

w opisywanej pracy [17] oblicza funkcje HOG dla każdego kanału kolorów po czym łączy je ze sobą tworząc histogram. W odróżnieniu od podobnych metod, obliczenie funkcji HOG następuje na podstawie mapy prawdopodobieństwa, ponieważ mapa prawdopodobieństwa może kodować informacje o kolorze i kształcie znaku drogowego, jednocześnie tłumiąc wpływ tła. Jednak na podstawie samej mapy prawdopodobieństwa nie ma możliwości zidentyfikowania znaku. Aby rozwiązać ten problem, dodano dodatkową funkcję w postaci klasycznego algorytmu HOG. Do identyfikacji przeszkolony został wielopoziomowy klasyfikator SVM. Autorzy algorytmu przetestowali swoją pracę na zbiorze danych GTSDb, gdzie istnieją trzy kategorie znaków drogowych, dlatego wyszkolony został 4-klasowy klasyfikator SVM z dodatkową klasą tła. Za jądro klasyfikatora SVM została wybrana funkcja radialna, ponieważ najlepiej sprawdziła się w eksperymentach. Zaproponowany algorytm na bazie danych GTSDb wykrywa znaki zakazu i nakazu z 100 % poprawnością, znaki ostrzegawcze z 94.29 % poprawnością. Czas wykrycia znaków w ramce to 0,067 sekundy co jest bardzo zadowalającym wynikiem.

#### *Algorytm Viola i Jonesa*

Algorytm Viola–Jones zaproponowany w 2001 przez Paul Viola i Michael Jones wprawdzie został stworzony do wykrywania twarzy [25], lecz znakomicie nadaje się do wykrywania obiektów każdego typu na obrazie. Podejście dla problemu TSR zostało przedstawione w pracy [2]. W pracy jako klasyfikatora używa się algorytmu opartego na boostingu nazwanego AdaBoost (ang. Adaptive Boosting). Tworzy on z kilku mniejszych klasyfikatorów jeden silny klasyfikator. Dla ww. pracy jako słabe klasyfikatory przyjmuje się drzewa decyzyjne, w których każdy węzeł jest jedną z cech haara-podobnych. Cechy Haara swoją nazwę zawdzięczają falką Haara zaproponowanym przez Alfréda Haara w 1909 lub 1910 roku. Cechy Haara to najprościej mówiąc pewna maska, która posiada dwa typy pikseli. Po jej nałożeniu na część obrazu sumuje się jasności pikseli należących do danego typu. Jeżeli różnica między dwoma typami jest wystarczająca cecha Haara daje pozytywną odpowiedź. Algorytm używa niewielkiej ramki przesuwanej po obrazie ROI. W każdym położeniu ramki za pomocą cech Haara stara się odnaleźć miejsce występowania znaku. Cechy Haara sprawdzane są kaskadowo co pozwala od razu odrzucić rejony, gdzie znak nie występuje. By przyspieszyć obraz zostaje scałkowany. W wcześniej już wspomnianej pracy [2] metoda pozwoliła na wykrywanie nie tylko znaków drogowych, lecz również na wykrycie samochodów a nawet rowerzystów. Ponieważ algorytm oprócz wykrywania znaków wykrywa dwa inne typy obiektów, wprowadzono dodatkowo funkcje subkategoryzacji. Wielką zaletą metody jest wykrywanie obiektów w czasie rzeczywistym. Praca, która korzysta z metody opartej na AdaBoost i cechach Haara została zaprezentowana w [26]. Dzieło pozwala na wykrywanie i śledzenie obiektów. Cechy Haara zostały również wsparte informacją o kolorach co pozwoliło na redukcję błędów pierwszej klasy z 1.6 % do 1.4 % oraz błędów drugiej klasy z 0.3 % do 0.03 %, co jest spektakularnym wynikiem. Praca [10] pozwala na śledzenie obiektów i również wykorzystuje kaskadę AdaBoost. Co ciekawe algorytm przedstawiony w pracy jest w stanie sam się doszkalać przez zbieranie pozytywnych i negatywnych próbek. Algorytm do doszkalania wykorzystuje śledzenie wsteczne. Oczywiście jest, że znaki które są bliżej są również lepiej rozróżnialne, dlatego po wykryciu znaku algorytm sprawdza poprzednie klatki filmu w poszukiwaniu wykrytego znaku w mniejszej skali i stara się samoczynnie doszkolić.

### *Symetria promieniowa*

Niektóre ostatnie prace [27] i [28] wdrożyły szybki algorytm oparty na symetrii promieniowej, która dostosowany do kształtów trójkątnych, kwadratowych, diamentowych, ośmiokątnych i okrągłych. Algorytm uwzględnia gradient obrazu dostarczonego skali szarości i wykorzystuje naturę kształtów, które głośnią w punkcie środkowym dla okrągłych znaków i linii głośów w przypadku regularnych wielokątów. Główną zaletą tej metody jest to, że jest ona w stanie działać w czasie rzeczywistym. Ponieważ wykrywa kształty oparte na krawędziach, algorytm jest odporny na zmiany oświetlenia.

### **Klasyfikacja znaków drogowych**

Klasyfikowanie znaków, wyłączając etap śledzenia, jest ostatnim etapem wykrywania znaków drogowych. Omawiana poprzednio segmentacja zawęża się obszar występowania znaku co w rezultacie prowadzi do faktu, że znak zajmuje około 70 % fragmentu odnalezionego przez segmentację. Zadaniem klasyfikacji jest odróżnienie czy w miejscu wskazanym przez segmentację znak rzeczywiście istnieje i jeżeli tak jest odczytanie znaku. Obecnie najpowszechniej stosowanym klasyfikatorem obrazów są sieci neuronowe. Warto jednak wspomnieć, że prócz sieci istnieją inne metody rozpoznawania typu znaku. W tej części zostały opisane metody klasyfikacji na potrzeby systemu TSR.

### **Wzorcowe dopasowanie**

Template matching jest klasyczną metodą bazującą na informacji wzajemnej pozwalającą odnaleźć obiekt na obrazie. Na potrzeby metody TM wzorce obiektów, które należy rozpoznać gromadzone są w bazie wzorców. Następnie zazwyczaj prostą metodą polegającą na dopasowaniu dwóch obrazów do siebie np. metodą najmniejszych kwadratów. Metoda TM jest prosta w zaimplementowaniu, lecz przez czas pracy dla kilkudziesięciu typów znaków drogowych jest nieefektywna. Szczegółowy opis metody można znaleźć w publikacji [29]

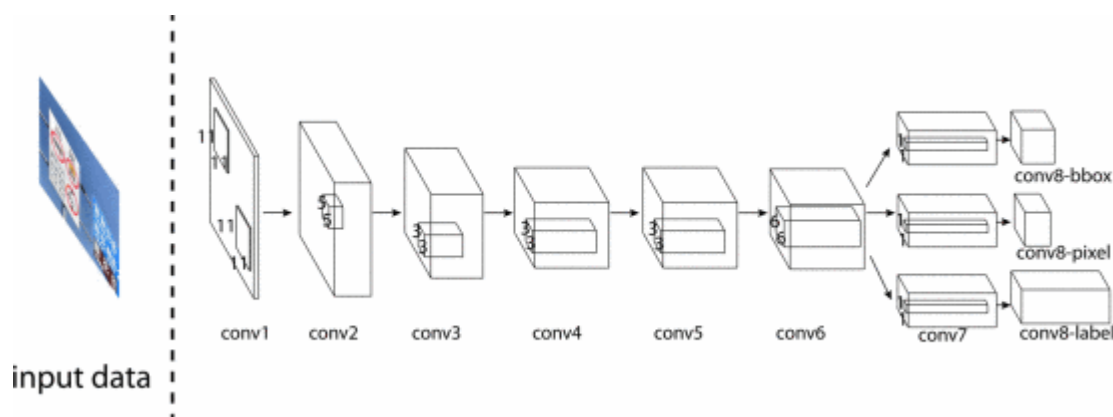
### **Sztuczne sieci neuronowe**

Sztuczne sieci neuronowe są zaprogramowanymi strukturami, które symulują model zdolny do wykonywania obliczeń matematycznych. Idea sztucznych sieci neuronowych inspirowana była budową ludzkich neuronów. Każda sieć zbudowana jest z warstw węzłów zwanych neuronami. Węzły połączone są ze sobą synapsami. Zazwyczaj do jednego węzła wchodzi wiele synaps. Każda synapsa posiada swoją wagę, a węzeł funkcję aktywacji. Gdy na wejściu synapsy pojawia się sygnał mnożony on jest przez wagę synapsy i podawany na węzeł. Gdy łączna wartość sygnału podanego na węzeł jest wystarczająca sygnał podawany jest dalej na kolejną warstwę. Sieci neuronowe stosowane są do wszelakich zadań, lecz na potrzeby pracy zostanie przedstawione zastosowanie w rozpoznawaniu obrazów. W takim zastosowaniu na wejściu sieci podawane są piksele obrazu, który należy rozpoznać. Zazwyczaj sieć ma określoną liczbę pikseli wejściowych, dlatego wykryty obszar przez segmentację należy przeskalować i znormalizować do tej liczby. Do normalizacji obrazu można stosować wiele metod. Najprostszą z nich jest odrzucenie co pikseli, co konkretną wartość liczbową. Rozwiązaniem, dzięki któremu traci się mniejszą ilość informacji, lecz jest bardziej kosztowny obliczeniowo jest interpolacja dwuliniowa. Już w artykule z 1996 roku [18] zostały użyte sieci neuronowe do klasyfikacji typu znaku drogowego. Istnieje ogromna liczba typów sieci neuronowych i metod usprawniających ich działanie. Sieć składająca się z więcej niż jedna warstwa ukryta nazywana jest siecią wielowarstwową. Przykładem takiej sieci jest perceptron wielowarstwowy.

### *Wielowymiarowy perceptron*

MLP (ang. Multilayer perceptron) - perceptron wielowarstwowy jest najpopularniejszy rodzajem sieci neuronowych. Taka sieć składa się z jednej warstwy wejściowej i kilku warstw ukrytych. Na potrzeby klasyfikacji w węzłach sieci MLP znajduje się funkcja nieliniowa. Typem sieci MLP służącym do rozpoznawania obrazów jest spłotowa sieć neuronowa inaczej znana jako sieć konwolucyjna. Tego typu sieć została przedstawiona w wcześniej wspomnianej pracy [18]. Autorzy użyli dwóch sieci typu MLP. Jednej dla znaków trójkątnych, a drugiej dla okrągłych. Wielkość wejścia sieci odpowiadała obrazowi o rozmiarach 30x30 piksel. Warstwa wyjściowa sieci posiadała 10 wyjść z czego 9 wyjść odpowiadało znakom. Ostatnie wyjście informowało, że znak nie został rozpoznany. Autorzy eksperymentalnie dobrali wielkość sieci z pośród 3 typów sieci. Najlepsze wyniki uzyskała 3 konfiguracja 30x30/15/5/10 z 98% skutecznością dla najlepiej rozpoznawalnych znaków. Rozpoznawanie na jednostce obliczeniowej PC486 33 MHz zajmuje 220 ms dla obrazu w rozdzielczości 256x256 co jest znakomitym wynikiem jak na rok 1996. Należy jednak zaznaczyć, że segmentacja w tej pracy była bardzo kosztowna czasowo, co nie pozwoliło algorytmowi działać w czasie rzeczywistym.

W innej pracy przedstawionej w 2003 roku [13] również użyto spłotowych sieci neuronowych do rozpoznawania znaków drogowych. W tej pracy segmentacja obszaru występowania znaku przeprowadzona była za pomocą progowania oraz algorytmów genetycznych. Sieć była uczona za pomocą algorytmu wstecznej propagacji błędów (ang. Backpropagation Neural Network), według paradygmatu ART1 (ang. Adaptive Resonance Theory). Sieć posiada dwie warstwy ukryte. Dla całego algorytmu przedstawionego w pracy rozpoznanie obrazu wraz z klasyfikacją znaków zajmuje mniej niż 8.8 ms na procesorze AMD Duron 1 GHz. Biorąc jednak pod uwagę, że dla systemów TSR rozpoznanie w czasie rzeczywistym liczone jest od prędkości 0.2 s dla jednego obrazu wynik pracy jest bardzo dobry. W pracy [30] opracowano algorytm bazujący na sieci neuronowej propagacji wstecznej pozwalający na śledzenie obiektów. Na potrzeby algorytmu stworzono dwie sieci działające równolegle ze sobą. Jedna z nich stara się zidentyfikować obiekt na bazie koloru, a druga na bazie wykrytych krawędzi. Obiekt jest rozpoznawany i klasyfikowany a następnie śledzony. Czas rozpoznania obrazu wynosi 0.3 s. Sieć neuronowa uczona algorytmem wstecznej propagacji została również zaimplementowana w 2005 r. przez autorów artykułu [31]. Autorzy wspomnieli, że proces uczenia sieci algorytmem wstecznej propagacji jest kosztowny pod względem obliczeniowym oraz łatwo podczas szkolenia sieci wpaść w lokalne minimum co przysparza problemów. By zapobiec ostatniemu problemowi część autorów algorytmów starała się uczyć sieć metodami głębokiego uczenia DL (ang. Deep Learning). Warto zauważyć, że przy uczeniu sieci jakkolwiek metodą ważne jest by niektóre obrazy uczące były w pewien sposób zaszumione oraz obrócone pod małym kontem. Taki zabieg powoduje większą odporność sieci. W pracy [32] szkolono dwie sieci wielowarstwowe metodą głębokiego uczenia. Jedna sieć była przeznaczona do segmentacji obrazów, a druga do klasyfikacji. Obie sieci posiadały podobną strukturę z wyjątkiem ostatniej warstwy. Przy testowaniu sieci przeznaczonej do segmentacji obrazu zauważono, że sieci spłotowe są znacznie wydajniejsze, gdy zostają użyte w trybie przesuwanego okna. Poza tym część informacji z nakładających się regionów można użyć wielokrotnie co przyspiesza algorytm. Architektura sieci przedstawiona jest na Rysunek 14 Architektura sieci spłotowej z rozgałęzieniem po 6 warstwie

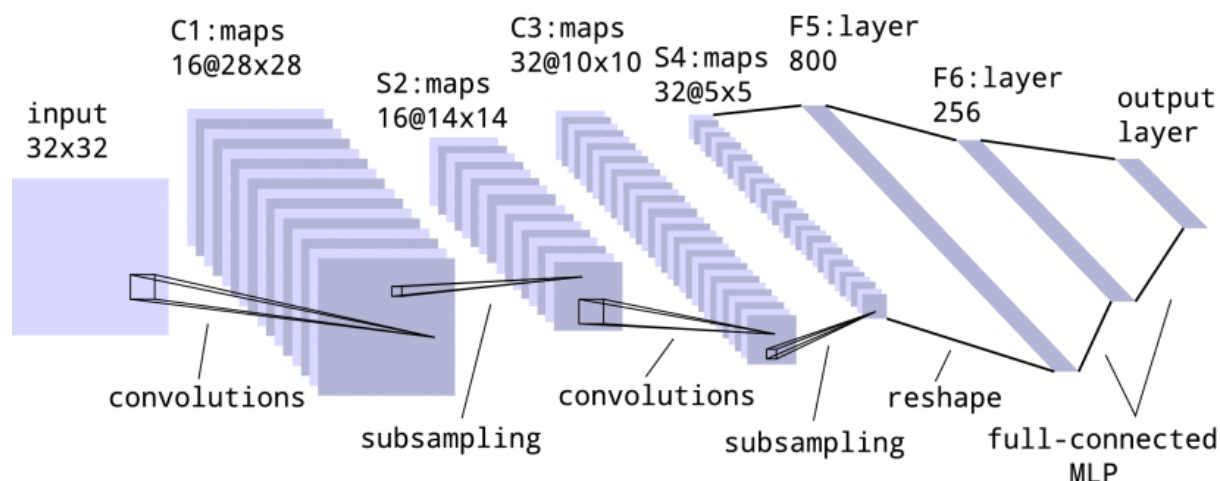


Rysunek 14 Architektura sieci spłotowej z rozgałęzieniem po 6 warstwie

Przy szkoleniu sieci niektóre próbki szkoleniowe były zniekształcone o  $\pm 20^\circ$  oraz dodano obrazy nie zawierające znaków drogowych. Łączenie użyto około 6600 obrazów do szkolenia i 3400 do testowania sieci. Do testów została użyta jednostka Intel Xeon E5-1620, dwoma kartami NVIDIA Tesla K20 G-PU i 32 GB pamięci. Sieć wykrywa znaki drogowe w ciągu 0.8 s z 88 % skutecznością.

#### Konwolucyjne sieci neuronowe

CNN (ang. Convolutional Neural Network) czyli spłotowe sieci neuronowe, nazywa się takie sieci, których pierwsze warstwy dokonują spłotu danych wejściowych. Takie sieci zostały wprowadzone by zapobiec problemowi przeuczenia się sieci. Spłot może polegać na łączeniu informacji z kilku pikseli i przekazywaniu jej jako ujednoliconą wartość. W pracy [17] zastosowano spłotową sieć neuronową dla by sklasyfikować znaki drogowe. W tym celu wyszkolono trzy sieci, każda dla innego typu znaku. Na wejście dostarczano obrazy w skali szarości i rozmiarze  $32 \times 32$  pikseli. Architektura sieci przedstawiona jest na Rysunek 15



Rysunek 15 Struktura sieci spłotowej opisywanej pracy

Ponieważ obrazy są przechwytywane w różnych warunkach oświetleniowych i pogodowych, znaki tej samej klasy mogą znacząco różnić się od siebie. Aby zmniejszyć różnicę jasności użyto metody korelacji adaptacyjnego histogramu ograniczonego kontrastem (CLAHE), pozwalającej dostosować kontrast obrazów. Z powodu tego, że wykryte regiony mogą zawierać błędy pierwszego i drugiego rodzaju dodano klasę tła w trakcie uczenia. Każda z sieci zawierała dwie warstwy spłotowe i dwie warstwy posiadające architekturę sieci MLP. Rozmiar obrazu

wejściowego wynosi  $32 \times 32$ . Rozmiar filtru w obu warstwach splotowych wynosił  $5 \times 5$ . Po pierwszej warstwie splotu, zostało dodanych się 16 map powodując zmniejszenie rozmiaru wejścia drugiej warstwy splotowej najpierw do rozdzielczości  $28 \times 28$ , a następnie do  $14 \times 14$ . Algorytm po transformacjach obrazu i zmapowaniu dostarcza wektor o długości 800 znaków. Sieć jest zaprogramowana w języku C++ z wykorzystaniem biblioteki OPENMP. Wykorzystywane są również implementacje MSER, HOG i SVM z biblioteki OpenCV. Sieć neuronową wyszkolono za pomocą Torch7 i przepisano wyszkolone wartości do kodu C++. Algorytm został uruchomiony na komputerze z czterordzeniowym procesorem 3,7 GHz, poprawnie wykrywając z dokładnością 99.29 % znaków zakazu, 96.74 % znaków nakazu i 97.13 % znaków ostrzegawczych przy czasie wykrywania wynoszącym 0.162 s. Znaki do testowania wzięto z niemieckiego benchmarku GRSDb.

Architektura sieci splotowej przedstawionej w artykule [14] różni się od tradycyjnych sieci splotowych typem nieliniowości, ponieważ w węzłach sieci użyto funkcji przypominającą funkcję tangensa hiperbolicznego. Dodatkowo wykorzystano połączenia między niesąsiadującymi warstwami sieci. Tradycyjne sieci splotowe nie posiadają połączeń między warstwami niesąsiadującymi. W warstwie pooling-u użyto różnych parametrów odpytywania dla danych dostarczanych z różnych warstw. Sieć została zaimplementowana z użyciem biblioteki EBLearn, oraz była również testowana na znakach pochodzących ze zbioru GTSRB. W przeciwieństwie do poprzedniej sieci ta sieć była uczona dla wszystkich zbiorów znaków jednocześnie. Co ciekawe sieć osiągnęła wynik 98.97 % dokładności wykrywania dla architektury, w której zostały wykorzystane informacje o kolorach i 99.17 % dokładności dla obrazów w skali szarości. Jest to sprzeczne z ogólnym poglądem twierdzącym, że wykorzystanie kolorów powinno poprawić skuteczność algorytmu.

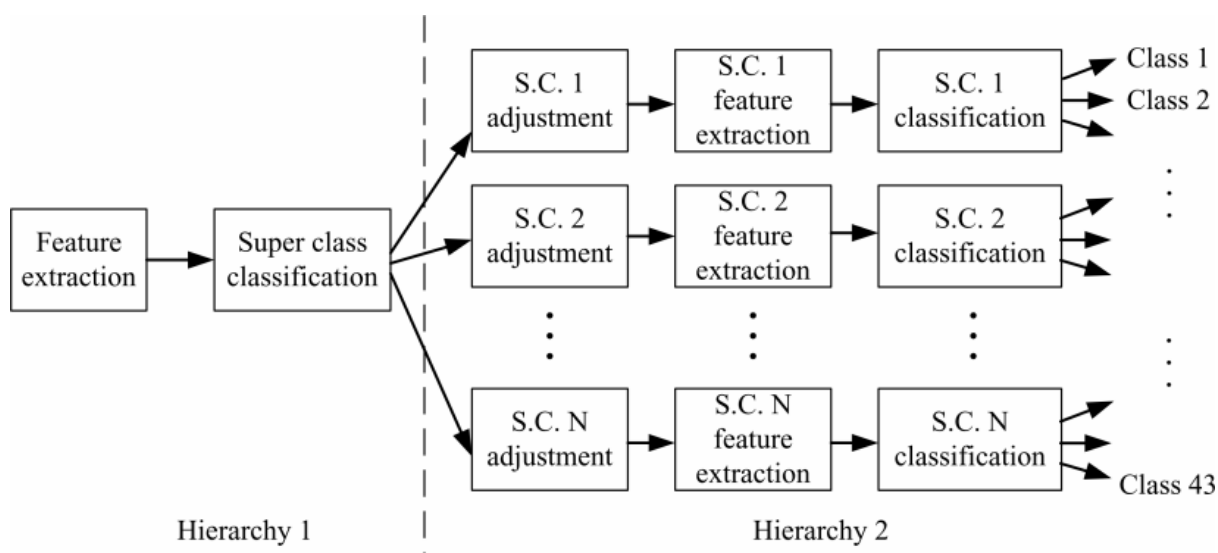
#### *Extreme Learning Machine*

ELM (ang. Extreme Learning Machine) jest algorytmem uczenia się pojedynczych sieci ukrytych sieci neuronowej (SFNN). Pierwszą zaletą algorytmu ELM jest to, że wartości wejściowe między wejściem i warstwami ukrytymi są losowo przypisywane, co pozwala losowe mapowanie cech. Ponieważ wyszkolone są tylko warstwy wyjściowe między warstwami ukrytymi i wyjściowymi, nie jest wymagane strojenie za pomocą wstecznej propagacji warstwa po warstwie. Algorytm ELM może uzyskać optymalne i uogólnione rozwiązanie do rozpoznawania wieloklasowego. Dodatkowo, łatwo można rozszerzyć ELM na sieć wielowarstwową lub głęboką za pomocą techniki autoencoder. Metoda ELM została również wykorzystana do modelowania lokalnych pól recepcyjnych i wykorzystywana do uczenia na dużych zbiorach danych. W związku z tym zastosowanie ELM dla TSR może dać lepsze rozwiązanie. Co więcej, z powodu losowego przypisania wag wejściowych, algorytm ELM może zmniejszyć koszt obliczeniowy szkolenia. Ponieważ istnieje tylko jedna ukryta warstwa, szybkość obliczeniowa procesu rozpoznawania jest również szybka. Praca [22] stosuje jako klasyfikator pojedynczą sieć z ukrytą warstwą. W oparciu o algorytm ELM, połączenie pomiędzy warstwami wejściową i ukrytą realizuje mapowanie cech losowych, podczas gdy tylko wagi pomiędzy warstwami ukrytymi i wyjściowymi są wyszkolone. W rezultacie strojenie warstwa po warstwie nie jest wymagane. Tymczasem norma wag wyjściowych jest zawarta w funkcji kosztów. W związku z tym, Klasyfikator oparty na ELM może osiągnąć optymalne i uogólnione rozwiązanie dla wieloklatkowych TSR. Ponadto może zrównoważyć dokładność rozpoznawania i koszty obliczeniowe.



## Maszyna wektorów nośnych do klasyfikacji

W artykule [15] zaproponowano innowacyjną metodę rozpoznawania znaków nazwaną hierarchiczną metodą wektorów nośnych. Algorytm zajmuje się wyłącznie klasyfikacją znaków drogowych bez etapu segmentacji i był uczony na zbiorze GTSRB. Algorytm działa w następujący sposób. Wpierw znak drogowy klasyfikowany jest przez SVM do jednej z trzech superklas. Następnie w zależności do jakiej klasy znak drogowy został zakwalifikowany zostaje użyta funkcja dostosowania perspektywy z różnymi parametrami. Po dostosowaniu perspektywy ponownie zostają użyte maszyny wektorów nośnych zdolne rozpoznać dokładnie co jest przedstawione na znaku. W roli wyjaśnienia jedna maszyna klasyfikuje znaki do podklas, a dla każdej podklasy wytrenowane są inne maszyny zdolne do odczytania znaku. Pierwsza maszyna SVM przydziela klasę znaku z pomocą algorytmu HOG. Dzięki rozpoznaniu superklasy znaku można wykorzystać informację o kolorze. Skorygowanie perspektywy jest kluczowym elementem pomagającym lepiej rozpoznać znak drogowy. Taka czynność eliminuje wszystkie translacje i rotacje znaku co pozwala lepiej odczytać znak w następnym etapie. Schemat blokowy proponowanej metody znajduje się Rysunek 16.



Rysunek 16 Schemat blokowy metody opartej na SVM

Algorytm dokonuje wstępnej ekstrakcji cech obrazu za pomocą wyszukania czerwonych pikseli oraz sprawdzeniu pikseli sąsiadujących z czerwonymi pikselami. Po wstępnym przetworzeniu obrazu następuje znalezienie okręgów na obrazie za pomocą radialnego detektora symetrii. Radialny detektor symetrii jest wariantem kołowej transformaty Hougha. Po wykryciu okręgów następuje wykrycie trójkątów, czyli znaków ostrzegawczych. Do wykrywania trójkątów na obrazie również użyta zostaje transformata Hougha wykrywająca linie. Następnie algorytm wykrywa linie przecinające się mogące stanowić kontury znaku. Na chwilę obecną algorytm zaproponowany w pracy z 43 klas znaków dostępnych w GTSRB rozróżnia znaki ostrzegawcze i zakazu będące okręgami i trójkątami z dokładnością 99,03 %. Znaki należące do innych klas są klasyfikowane za pomocą algorytmu SVM i HOG bez informacji o kolorze z dokładnością 99,94%, 98,89% i 99,90% dla znaków kolejno nakazu, ograniczeń prędkości i znaków pozostałych, co daje globalny wynik dokładności algorytmu na poziomie 99,52% z średnią szybkością 40 ms. Implementacja została wykonana za pomocą programu Matlab 2011b i na jednostce Core i3 3.3 GHz.

Praca [11] proponuje kompletny algorytm bazujący na SVM pozwalający zlokalizować i rozpoznać znak drogowy. Wcześniej zaznaczono, że segmentacja dokonywana jest w przestrzeni HSI, a regiony wykryte przez segmentację wstępnie są przydzielane do podklas za pomocą linowych maszyn SVM. Pozwala to wstępnie rozróżnić typ znaku. Rozpoznawanie realizowane jest przez maszyny SVM z jądrami Gaussa. Praca do treningu wykorzystuje bibliotekę LIBSVM. Starano się użyć funkcji liniowej jako hiperpłaszczyzny, lecz nie zawsze było to możliwe. W trudnych przypadkach wprowadzono jądro Gaussa. Na wejście SVM podawano 31x31 pikseli, dlatego każdy obszar, gdzie zostały wykryte potencjalne znaki musiał być przeskalowany do tej rozdzielczości. Dla każdego koloru i kształtu podano od 20 do 100 próbek uczących. Przetwarzanie pojedynczej ramki o wielkości 720×576 pikseli za pomocą proponowanej metody zajmuje 1.77 s na procesorze Pentium 4-M 2,2 GHz. Testy prowadzone były na autorskim benchmarku znaków obowiązujących w Hiszpanii.

### **Drzewo k-wymiarowe**

Praca [33] jest przykładem artykułu starającym się rozważyć przydatność drzew wielowymiarowych K-d (ang. K-Dimensional Tree) do rozróżniania znaków drogowych. Drzewo K-d jest wariantem drzewa binarnego opartego o wyszukiwanie najbliższego sąsiada. Zaletą drzew K-d jest łatwość budowy i aktualizacji oraz szybkość ich przeszukiwania. Dodatkowo drzewa dobrze radzą sobie z niezerównoważonymi zbiorami danych w przeciwieństwie do sieci neuronowych, które wymagają dobrego dostrojenia parametrów. W przypadku pracy drzewo K-d składa się z informacji dostarczonych od czterech transformat HOG na obrazie, każdej z innymi parametrami. Do wyszukiwania najbliższego sąsiada użyto algorytmu Best Bin First. Na znakach dostarczonych przez zbiór danych GTSRB drzewo K-d bazujące na deskrytorze HOG przedstawione w pracy uzyskało poprawność rozpoznania na poziomie 92,9%.

### **Lasy losowe**

W tej samej pracy co drzewa K-d [33] testowano również lasy losowe RM (ang. Random Forests) do problemu klasyfikacji znaków drogowych. Klasyfikacja obrazu przez las losowy polega na zebraniu odpowiedzi z kilku drzew decyzyjnych przez głosowanie. Lasy losowe również jak drzewa D-d dobrze radzą sobie z klasyfikacją na zbiorze niesymetrycznym niekiedy przewyższając niekiedy algorytmy SVM. Jak i w drzewie K-d opisywanym powyżej obraz wejściowy poddawany był 4 transformata HOG. Dla lasu losowego testowanego na tym samym zbiorze danych GTSRB poprawność klasyfikacji została poprawiona do 97,2%.

### **Podsumowanie**

Praca miała na celu zbadanie w jaki sposób zastosowanie wielu kamer może polepszyć wykrywanie znaków drogowych. Pierwszym etapem przeglądu literaturowego było opisanie możliwych konfiguracji kamer np. poprzez zastosowanie kamer z IR lub łączenia klatek filmu w panoramę. Do łączenia obrazów w panoramę zostało wyszukanych kilka metod, z czego najskuteczniejsze okazały się metody bazujące na punktach charakterystycznych, które zostały opisane. Są one szybkie w działaniu oraz odporne na małe rotacje dwóch obrazów. Należy również zaznaczyć, że problem tworzenia panoramy na potrzeby pracy niesie ze sobą wiele udogodnień. Z dość dużą dokładnością znana jest translacja pozioma obu obrazów. Translacja pionowa praktycznie nie występuje, co prowadzi do wniosku znany jest obszar, w którym obrazy z kamer się nakładają. Umożliwia to wykonywanie operacji dopasowania tylko na wspólnym obszarze rejestrowanym przez obie kamery co pozwoli zmniejszyć ilość obliczeń. Przez wprowadzone ograniczenia metody klasyczne nie są od skazane na niepowodzenie.

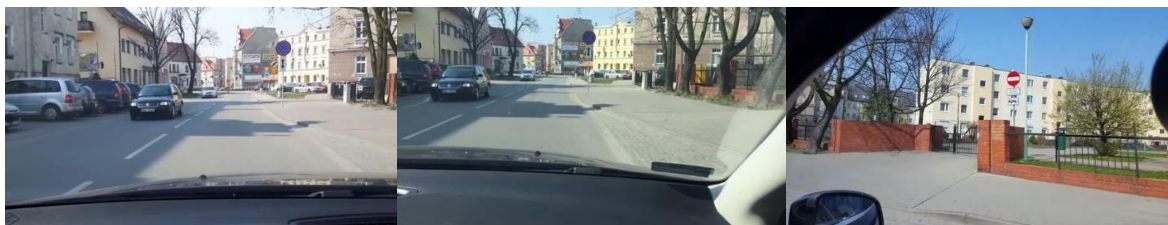
Metody bazujące na punktach charakterystycznych wydają się przynosić najbardziej obiecujące wyniki.

Podsumowując zagadnienie segmentacji należy stwierdzić, że podejścia oparte na kolorze są nadal stosowane ze względu na ich szybkość. Ich zawodność jest natomiast poprawiana dodatkowymi algorytmami działającymi poprzez wykrywanie kształtu. Podczas progowania najczęściej zostają odnalezione grupy pikseli nie powiązane ze sobą. By zlokalizować obiekty odnalezione przez progowanie należy w jakiś sposób połączyć piksele ze sobą i je oznaczyć. Dość prostą i skuteczną metodę przedstawiono w pracy [9] gdzie do każdego odnalezionego piksela zostaje przypisana etykieta. Algorytm etykietowania wprowadza osobny bufor umożliwiający łączenie regionów z różnymi etykietami. Podejścia oparte na teksturze (HOG, LPB, ACF) z dużą dokładnością potrafią wyłonić obrys znaku drogowego, co na etapie identyfikowania może posłużyć do szkolenia klasyfikatora. Niestety takie przetwarzanie na całym obrazie jest zbyt kosztowne obliczeniowo, dlatego zawęża się obszar stosując ROI. Pozwala to przyspieszyć obliczenia i umożliwia przetwarzanie obrazu w czasie rzeczywistym. Obszar ROI dotyczy głównie miejsc środkowych i po prawej stronie. Niektóre algorytmy jak ten zaproponowany w pracy [34] używają funkcji ACF z klasyfikatorem AdaBoost. Algorytm ACF został również użyty w pracy [35] w połączeniu z informacją o kolorach i gradientach oraz algorytmem HOG i uzyskał dobre wyniki. Oprócz powyższych podejść, spłotowa sieć neuronowa (CNN) jest stosowana do wykrywania znaków drogowych i osiąga doskonałe wyniki [14]. Warto również wspomnieć, że dla lepszej wstępnej filtracji obrazu można zastosować również Filtry Gabora, lecz obecnie używanie takiego algorytmu jest nieefektywne ze względu na korzyści i czas obliczeń. Wartym wspomnienia podejściem detekcji znaków jest tworzenie mapy cieplnej obszarów występowania znaków [36]. Taka mapa działa na zasadzie algorytmów biologicznych. Warto również dodać, że część obliczeń wykorzystanych w etapie tworzenia panoramy może posłużyć w etapie segmentacji obiektów.

Najpowszechniej stosowanym podejściem do klasyfikacji obiektów są spłotowe sieci neuronowe. Czas klasyfikacji odnajdywania obiektów przez sztuczne sieci neuronowe balansuje na granicy pracy w czasie rzeczywistym, dlatego niektórzy autorzy starają się przyspieszyć wykrywanie znaków drogowych łącząc sieci CNN z innymi algorytmami np. w pracy [17] algorytmem SVM. Drzewa wielowymiarowe oraz lasy losowe chodź osiągają znakomite wyniki niestety okazuje się, że są zbyt wolne by pracować w czasie rzeczywistym.

## Opis proponowanej metody do rozwiązania problemu TSR

Podczas projektowania programu musiano dokonać wielu wyborów. Pierwszym z nich był wybór w jaki sposób należy rozszerzyć system o zastosowanie większej ilości kamer. Ze względu na koszty kamery podczerwonej zrezygnowano z jej zastosowania. Zdecydowano rozszerzyć system o zastosowanie trzech jednakowych kamer pochodzących obejmujących obraz wokół samochodu. Prototyp systemu był zbudowany na trzech kamerach. Dwie z nich były umiejscowione na daszku przeciwsłonecznym kierowcy i pasażera i obejmowały obszar bezpośrednio przed pojazdem. Trzecia była skierowana w taki sposób by symulowała widok pasażera patrzącego w prawą stronę. Rysunek 17 przedstawia widoki dla każdej z kamer



*Rysunek 17 Widok z trzech kamer wybrany dla systemu*

W dalszej części kamerą lewą nazywano kamerę, której przykład obrazu pokazany jest jako pierwszy z lewej, obraz drugi przedstawia kamerę centralną, a obraz trzeci przedstawia obraz uzyskany z kamery prawej.

## Propozycja wykorzystania większej ilości kamer

Konfiguracja kamer pozwalała zarówno na stworzenie z nich panoramy jak i zastosowanie przetwarzania równoległego. Wpierw podjęto próbę stworzenia obrazu panoramicznego. Zdecydowano, że najlepszym rozwiązaniem będzie stworzenie takiego obrazu z pomocą wcześniej opisywanych metod bazujących na punktach kluczowych. Wybrano algorytm SIFT do uzyskania punktów kluczowych. Minimalna ilość punktów kluczowych do połączenia dwóch obrazów to dwa punkty, lecz wraz z wzrostem ilości wyznaczonych punktów rośnie dokładność dopasowania. Po stworzeniu algorytmu testów okazało się, że obraz z prawej kamery nie zawsze można było dopasować do obrazu centralnego. Skuteczność znajdowania tych punktów dla algorytmu SIFT wynosiła 39,50 %, dla algorytmu SURF 17,28 a algorytm OBR nie znalazł tych samych punktów ani razu. Uznano, że najlepszym algorytmem do wyłaniania punktów kluczowych będzie algorytm SIFT. Rysunek 18 obraz panoramiczny stworzony z obrazów uzyskanych z trzech kamer.



*Rysunek 18 Obraz panoramiczny stworzony z obrazów z trzech kamer*

Z powodu niskiej skuteczności łączenia obrazu uzyskanego z prawej kamery uznano, że połączone zostaną tylko obrazy z kamery centralnej i lewej. Skuteczność łączenia takich obrazów za pomocą algorytmu SIFT i SURT wynosiła 100 %, a za pomocą algorytmu ORB wynosiła 10 %. Rysunek 18 przedstawia obraz panoramiczny uzyskany z połączenia obrazów z kamery lewej i centralnej.



*Rysunek 19 Obraz panoramiczny stworzony z obrazów z kamery lewej i centralnej*

Na Rysunek 18 można dostrzec, że znak został zdeformowany przez algorytm. Była to niedopuszczalna wada takiego rozwiązania, dlatego uznano się odrzucić pomysł tworzenia panoramy z dostarczonych zdjęć. Rozwiązaniem, które zostało wybrane dla tej pracy jest przetwarzanie równoległe każdego z obrazów osobno. Dzięki takiemu rozwiązaniu żaden znak nie został zdeformowany oraz zwiększono możliwość wykrycia znaków. Poniżej znajduje się pełen opis programu odpowiedzialnego za przetwarzanie każdego z obrazów. Opis podzielono na kilka etapów. Wpierw w akapicie Segmentacja przedstawione zostały metody znajdowania znaków, a w akapicie Klasyfikacja przedstawia się w jaki sposób były rozpoznawane.

### **Segmentacja**

Segmentacja obrazu w postaci wyszukiwania obszarów mogących zawierać znaki była jednym z ważniejszych elementów całej pracy. Etapy przedstawiono na przykładzie obrazu pochodzącym z kamery po lewej stronie. Identyczny proces przeprowadzany był dla obrazu dostarczonego z kamery środkowej i po prawej stronie. Wszystkie operacje na obrazie przeprowadzane były z udziałem biblioteki OpenCV.

Pierwszą operacją wykonywaną na obrazie była jego konwertowanie do rozdzielczości 640 x 360 pikseli. Taki obraz przedstawia Rysunek 20.





*Rysunek 20 Oryginalny obraz wejściowy*

Następnie z obrazu zostawiono rejon, gdzie znaki mogły się znaleźć z największym prawdopodobieństwem, a resztę ucięto. Taki obszar nazwany jest regionem zainteresowań ROI. Powoduje to mniejszą ilość obliczeń, a co za tym idzie przyspiesza działanie algorytmu. Powoduje też wykrycie mniejszej ilości fałszywych regionów. Obszar ROI przedstawia Rysunek 21.



*Rysunek 21 Obraz po wyznaczeniu ROI*

Następnym krokiem było przetworzenie obrazu poprzez wyrównanie histogramu osobno dla każdej składowej piksela. Osobno dla czerwonej, zielonej i niebieskiej składowej. Efekt takiego działania został zaprezentowany na Rysunek 22.



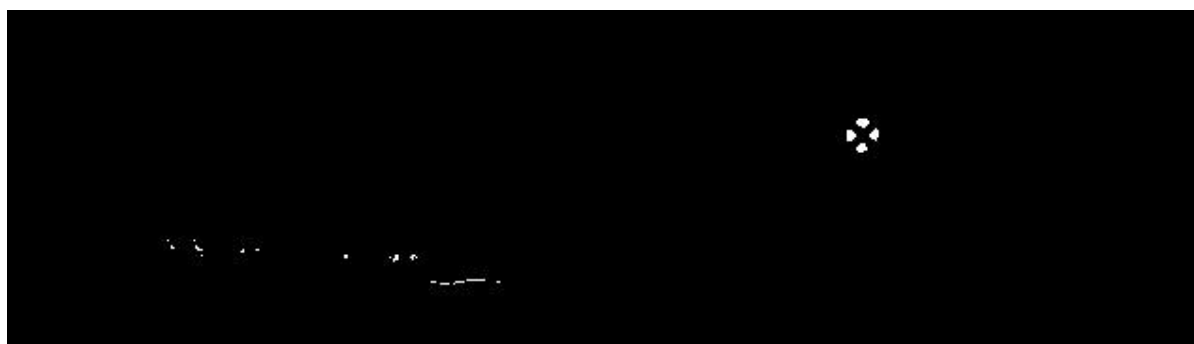
Rysunek 22 Obraz po wyrównaniu histogramów

Następnie obraz konwertowano do przestrzeni HSV. Przestrzeń została wybrana ze względu na swoją odporność na różnicę w jasności obrazu. W tym miejscu obraz wejściowy został skopiowany i przetwarzany przez trzy niezależne detektory. Detektory podzielone były ze względu na kolor znaków. Znajdowano znaki koloru niebieskiego, czerwonego i żółtego. Pierwszym krokiem każdego z detektorów było progowanie. Wartości progów dobrano eksperymentalnie po przeprowadzeniu parudziesięciu testów znaków z pomiędzy dla koloru od 0 do 180. Dla nasycenia i wartości barwy od 0 do 255. Przedstawia je Tabela 1.

Kolor	Niebieski	Czerwony	Żółty
Minimalny kąt koloru	92	129	14
Maksymalny kąt koloru	126	180	31
Minimalne nasycenie	197	101	116
Maksymalne nasycenie	255	255	255
Minimalna wartość barwy	41	30	112
Maksymalna wartość barwy	255	255	255

Tabela 1 Wartość parametrów progowania dla trzech kolorów

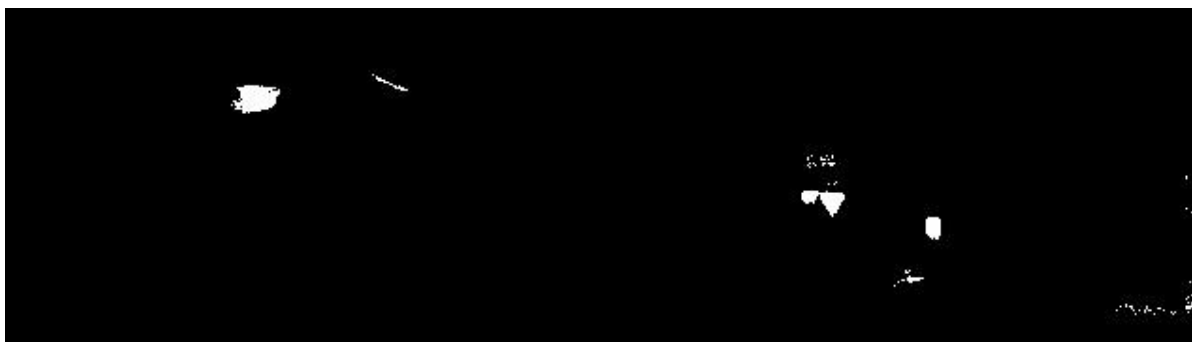
Rysunek 23 przedstawia efekt progowania dla każdego z kolorów.



(a) – Progowanie w kolorze niebieskim



(b) – Progowanie w kolorze czerwonym



(c) – Progowanie w kolorze żółtym

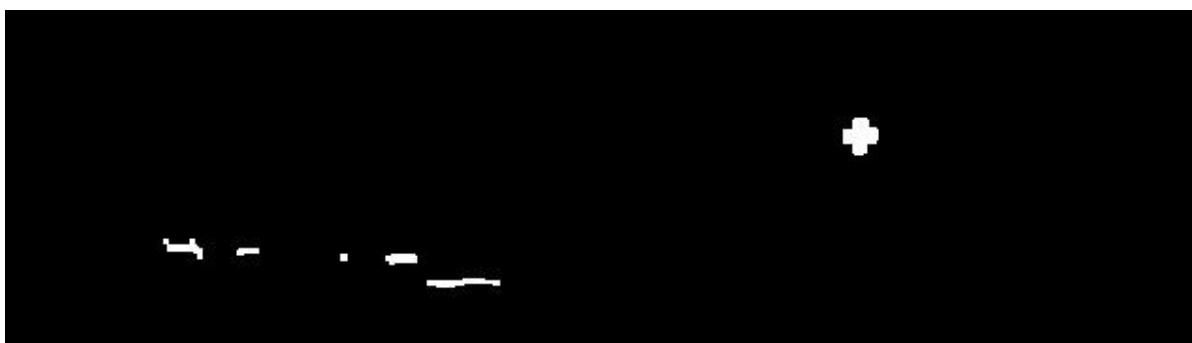
*Rysunek 23 Progowanie wykonane dla każdego z kolorów*

Można zauważyć, że progowanie wyłoniło obrys znaków drogowych, lecz również nieco obiektów, które nimi nie są. Problemem, z którym należało się zmierzyć było odfiltrowanie obszarów najmniej prawdopodobnych. Do tego zadania zdecydowano się zastosować operacje morfologiczne na obrazie. Inne operacje były stosowane dla koloru niebieskiego, czerwonego i żółtego. Filtry zostały przedstawia Tabela 2.

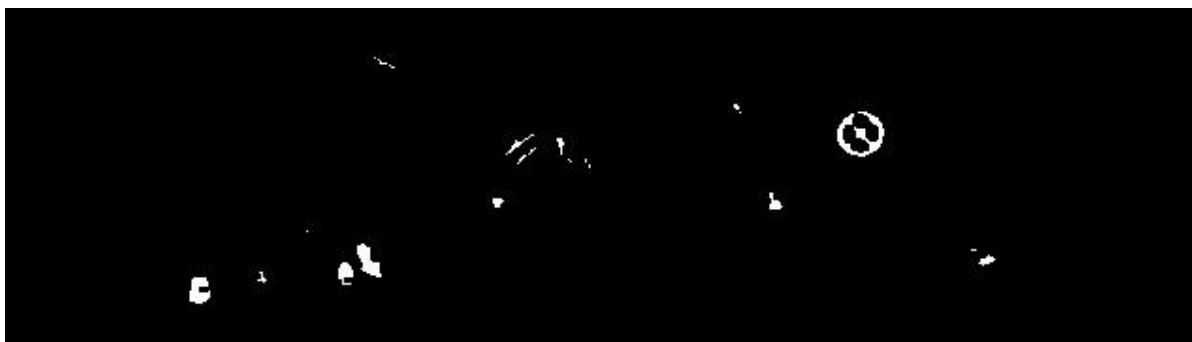
Kolor	Niebieski		Czerwony		Żółty	
Operacje	Dylatacja	1	Domknięcie	1	Otwarcie	1
	Domknięcie	5			Domknięcie	2

*Tabela 2 Operacje morfologiczne użyte na obrazie binarnym*

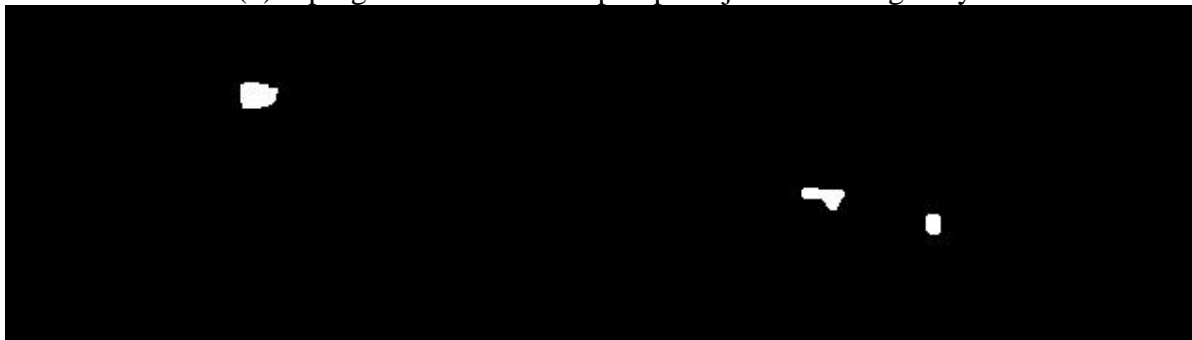
Jądnem filtru była siatka 3x3 piksele. Wybór parametrów został dobrany tak samo jak parametry progowania na podstawie badań i doświadczeń. Zauważono, że znaki niebieskie posiadają duży wycinek obszaru białego, czerwonego bądź czarnego, który nie zostaje uwidoczniony podczas progowania. Powoduje to, że znak na obrazie podzielony jest na fragmenty. By połączyć elementy znaku ze sobą zastosowano filtry przedstawione w Tabela 2. Znaki koloru czerwonego wykrywane były za pomocą detekcji okręgu na obrazie. Jak można dostrzec na Rysunek 23 obrazie (b) okrąg jest dobrze widoczny po samym progowaniu oraz znikoma część innych obiektów posiadała takie właściwości, dlatego zdecydowano się tylko użyć filtra pojedynczego domknięcia. Dla znaków drogowych w kolorze żółtym zdecydowano się zastosować operacje otwarcia i podwójnego domknięcia. Za takim wyborem przemawiały dwa argumenty. Pierwszym z nich jest, że znaki informacyjne nie posiadają obszaru białego na którym umiejscawia się symbol, a jedynie sam symbol znaku. Pozwala to usunąć pojedyncze piksele nie tracąc kształtu znalezionej znaku. Operacje domknięcia pogrubiają obrys znaku by był bardziej widoczny. Efekty nałożonych filtrów zaprezentowane są na Rysunek 24.



(a) – progowanie niebieskie po operacjach morfologicznych



(b) – progowanie czerwone po operacjach morfologicznych



(c) – progowanie żółte po operacjach morfologicznych

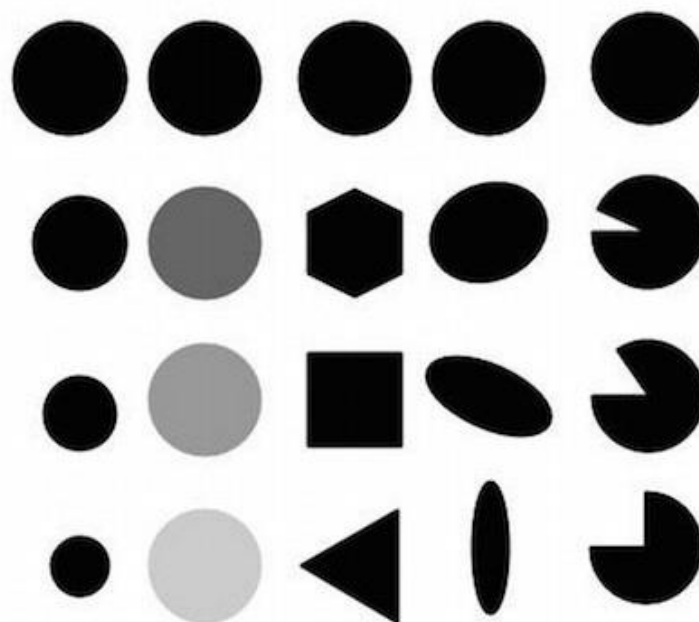
Rysunek 24 Obrazy po zastosowaniu operacji morfologicznych z Tabela 2

Na obrazach binarnych poddanych operacjami morfologicznymi wyszukiwano obszary mogące zawierać znak. Dla obrazów progowanych w kolorze niebieskim i żółtym zastosowano wbudowaną funkcję Blob Detector, czyli detektor plam. Detektor ten przyjmował zestaw danych, które pozwalały wyznaczać parametry wyszukiwanych obiektów. Parametry te przedstawia Tabela 3. Tak jak parametry progowania zostały one dobrane eksperymentalnie.

Kolor	Niebieski	Żółty
Minimalna wielkość	68	40
Maksymalna wielkość	897	1000
Minimalny próg jasności	0	0
Maksymalny próg jasności	255	255
Minimalna kolistość	0,38	0,45
Maksymalna kolistość	1	1
Minimalne wydłużenie	0,2	0,6
Maksymalne wydłużenie	1	1
Minimalna wklęsłość	0,61	0,73
Maksymalna wklęsłość	1	1

Tabela 3 Parametry Blob Detector'a

Wielkość opisywana jest jako ilość pikseli przynależąca do danego obiektu. Minimalny i maksymalny próg jasności brany był z całego dostępnego zakresu od 0 do 255. Kolistość obiektu wskazuje jakie wartości kątów wklęsłych posiada. Czym większa wartość kątów tym obiekt jest bardziej kolisty. Ostatnimi parametrami wejściowymi było wydłużenie i wklęsłość obiektu. Prócz dwóch pierwszych parametrów były one brane z zakresu od 0 do 1. Przykłady obiektów z różnymi parametrami przedstawia Rysunek 25



Wielkość    Jasność    Kolistość    Wydłużenie    Wypukłość

*Rysunek 25 Przykłady obiektów z różnymi parametrami*

Jak wcześniej wspomniano elementy czerwone w znakach zakazu są okręgami. Zdecydowano więc by wyszukiwać je na obrazie nie poprzez funkcję Blob Detector lecz za pomocą transformaty Hougha przystosowanej do wyszukania okręgów. Parametry wejściowe przedstawione są w Tabela 4.

Kolor	Czerwony
Metoda	Gradientowa
Wymiar akumulatora (dp)	1
Minimalny dystans między okręgami	15
Próg krawędzi	1
Próg dla centrów okręgu	15
Minimalny promień	6
Maksymalny promień	18

*Tabela 4 Parametry transformaty Hougha dla okręgów*

Metoda jaką była liczona transformata to jedyna zaimplementowana metoda w OpecCV – gradientowa (Hough Gradient). Następnym parametrem był wymiar akumulatora liczony jako odwrotny stosunek rozdzielczości akumulatora do rozdzielczości obrazu. Na przykład, jeśli  $dp = 1$ , akumulator miał taką samą rozdzielczość jak obraz wejściowy. Jeśli  $dp = 2$ , akumulator miał połowę szerokości i wysokości. Wybrano by akumulator miał taki sam wymiar jak obraz wejściowy. Minimalna wartość pomiędzy dwoma wykrytymi okręgami wynosiła 15 pikseli. Spowodowało to, że wykryte okręgi mogące symbolizować znaki nie nakładały się na siebie. Następnie należało wybrać najwyższy próg przekazanych do detektora krawędzi. Dolny próg był dwa razy mniejszy od górnego. Następnie należało zadać próg akumulatora dla centrów okręgu na etapie wykrywania. Im próg był mniejszy, tym więcej fałszywych okręgów mogło być wykrytych. Koła, z większymi wartościami akumulatora zostały zwracane jako pierwsze, dlatego uznano, że próg ten powinien mieć dużą wartość. Ostatnimi parametrami był zakres długości promienia dla wykrytych okręgów.



Dla obiektów wykrytych przez metodę wykrycia plam jak i przez transformatę Hougha kopiowano wycinek nieprzetworzonego obrazu odpowiadający miejscu wykrycia. W ten sposób tworzono w obrazy mogące być przetwarzane w części klasyfikacji. Przykłady tych obrazów przedstawia Rysunek 26.



Rysunek 26 Przykłady wykrytych obiektów podczas segmentacji.

Na Rysunek 26 można dostrzec, że proponowany algorytm segmentacji jest w stanie wykryć znaki drogowe. Dokładne parametry skuteczności wykrytych regionów przedstawione zostały w akapicie Przygotowanie środowiska badawczego i plan badań. Tak przygotowane obrazy zostały przekazywane do algorytmu odpowiedzialnego za klasyfikację.

### Klasyfikacja

Rozpoznanie wycinka obrazu, na którym powinien znajdować się znak zrealizowane jest za pomocą sieci neuronowej. Ponieważ znaki drogowe zostały rozpoznane na podstawie koloru, zdecydowano się stworzyć trzy niezależne sieci dla każdego koloru osobno. Takie działanie zmniejszyło ilość błędnych rozpoznań. Obraz dostarczony przez segmentację należało przeskalować w taki sposób by jego wielkość odpowiadała wymiarom wejściowym sieci. Tak

zmodyfikowany obraz podawany był na wejście. Sieć została zaprogramowana za pomocą biblioteki TensorFlow. Uznano, że na potrzeby systemu klasyfikacji znaków rozwiązaniem wystarczająco skutecznym będzie zastosowanie sieci sekwencyjnych. Są to sieci, których każdy neuron w warstwie połączony jest z wszystkimi neuronami w warstwie poprzedniej i następnej. W takich sieciach nie występują połączenia między warstwowymi oraz tworzące sprzężenia zwrotne.

#### Parametry sieci

Pierwsza warstwa sieci była warstwą splotową. Następnie dodano warstwy głębokie. Parametry sieci przedstawia Tabela 5 Parametry warstw opisywanej sieci Tabela 5. Zostały one dobrane eksperymentalnie, na podstawie przeczytanych artykułów i badań. W części Przygotowanie środowiska badawczego i plan badań zostały opisane badania mające na celu poprawę właściwości prezentowanej sieci.

Nr.	Rodzaj warstwy	Ilość neuronów	Funkcja aktywacji	Dodatkowy opis
1	Conv2D	64	Relu	Rozmiar obrazu wejściowego 50 x 50 Rozmiar splotu = 3 x 3
2	Flatten	-	-	Warstwa rozplaszczająca
3	Dense	64	Relu	-
4	Dense	64	Relu	-
5	Dense	32	Relu	-
6	Dense	Zmienny	Softmax	Ilość neuronów = ilość znaków w klasie

Tabela 5 Parametry warstw opisywanej sieci

Warstwa *Conv2D* oznacza warstwę konwolucyjną bazującą na dwuwymiarowym obrazie. Warstwa *Flatten* to rozplaszczająca warstwa, która z obrazu 2D tworzy jednowymiarowy wektor o wielkości równej ilości pól w obrazie. Warstwa *Dense* oznacza klasyczną warstwę głęboką. Funkcja aktywacji *Relu* jest funkcją przedstawioną przez Równanie 5 Funkcja aktywacji Relu.

$$f(x) = x \text{ dla } x > 0$$

$$f(x) = 0 \text{ dla } x \leq 0$$

Równanie 5 Funkcja aktywacji Relu

Funkcja *Softmax* jest funkcją regresji wielorakiej. Z powodu tego, że szkolone były trzy sieci dla każdego koloru, a ilość znaków każdego koloru się różniła, ostatnia warstwa sieci musiała mieć różny wymiar. Pojedynczy neuron odpowiadał jednemu znakowi. Sieć była szkolona z pomocą optyimizatora *Adam* oraz funkcji celu (wyniku optymalizacji) *Sparse Categorical Crossentropy*. Szkolenie sieci przez 10 epok, gdzie po każdej epoce dane wejściowe ulegały przemieszaniu.

#### Baza ucząca.

Dużym problemem było stworzenie bazy uczącej dla sieci. Tworzenie bazy zaczęto od pobrania baz znaków przedstawionych w akapicie Bazy znaków drogowych. Niestety jak wspomniano w akapicie Rodzaje znaków drogowych znaki drogowe w Polsce różnią się od większości znaków na świecie. Skutkiem tego było stworzenie własnej bazy danych. Baza ta składała się początkowo z 40 rodzajów znaków. Niestety poszczególne klasy tych znaków posiadały różną

ilość przykładów znaków. Znaki pochodzące z bazy GTSRB posiadały przykłady bardzo podobne do siebie, dlatego należało wybrać tylko część dostępnych tam obrazów jednego znaku i uzupełnić to znakami z bazy belgijskiej i chińskiej. Największym problemem były znaki ostrzegawcze, ponieważ w żadnej ze znalezionych baz nie odpowiadały one znakom polskim. Ten problem również opisany został w akapicie Rodzaje znaków drogowych. Jedynym wyjątkiem jest znak *droga z pierwszeństwem*, który we wszystkich krajach akceptujących Konwencję Wiedeńską jest jednakowy. Najważniejszym znakiem ostrzegawczym oprócz znaku *droga z pierwszeństwem* jest znak *ustąp pierwszeństwa*. Wygląd tego znaku w wszystkich krajach został pokazany na Rysunek 2. Zdecydowano się stworzyć bazę znaków ustąp pierwszeństwa na podstawie zdjęć robionych samodzielnie. Ostatnim problemem z bazą znaków było bardzo duże różnice w ilości znaków w klasach. Przykładem może być klasa znaku *droga z pierwszeństwem* która posiada ponad 2,5 tysiąca przykładów znaków oraz klasa *nakaz jazdy w prawo lub lewo*, która posiadała tylko 16 przykładów. Należało więc ujednolicić bazę tak by każdy znak posiadał podobną ich ilość. Ilość znaków, która została wybrana to ok 300 znaków każdej klasy. Z 40 klas tylko 32 posiadało taką ilość. Reszta została odrzucona. Na samym końcu dodano bazę dla znaku *ustąp pierwszeństwa* stworzoną własnoręcznie i podzielono znaki według koloru. Rodzaje wykrywanych znaków przedstawia Tabela 6.

Niebieskie	Czerwone	Żółte
parking	zakaz skrętu w prawo lub lewo	ustąp pierwszeństwa
zakaz postoju	nakaz skrętu w lewo	pierwszeństwo przejazdu
zakaz zatrzymywania się	ograniczenie do 20	
przejście dla pieszych	ograniczenie do 30	
droga rowerowa	ograniczenie do 40	
rondo	ograniczenie do 50	
nakaz jazdy z prawej strony znaku	ograniczenie do 60	
nakaz jazdy z lewej strony znaku	ograniczenie do 70	
nakaz jazdy prosto	ograniczenie do 80	
nakaz jazdy prosto lub w prawo	zakaz wyprzedzania	
nakaz jazdy prosto lub w lewo	zakaz wjazdu	
nakaz skrętu w prawo	zakaz ruchu	
nakaz skrętu w lewo	STOP	
droga jednokierunkowa	zakaz postoju	
	zakaz zatrzymywania się	
	zakaz skrętu w prawo	
	zakaz jazdy prosto	

Tabela 6 Rodzaje wykrywanych znaków

W Tabeli 6 można dostrzec, że znaki *zakaz postoju* oraz *zakaz zatrzymywania się* zostały dodane do znaków wykrywanych przez detektor znaków niebieskich oraz czerwonych. Takie działanie miało spowodować większą wykrywalność tych znaków.

## Przygotowanie środowiska badawczego i plan badań

By przetestować zaprezentowany system wykrywania znaków stworzono własny benchmark. Test składał się z obrazów uzyskanych z 3 kamer zamocowanych na samochodzie. Dwie kamery obejmowały obszar przed samochodem, trzecia kamera obejmowała obszar po prawej stronie samochodu. Wszystkie obrazy zostały zarejestrowane podczas jednego przejazdu w dobrych warunkach pogodowych. Przykład takich obrazów został przedstawiony na Rysunek 17. Czasy wybranych obrazów były jednakowe. Następnie na każdym obrazie zaznaczono znaki oraz opisano za pomocą nazwy znaku oraz koloru. Przykład takiego obrazu przedstawia. Rysunek 27.



Rysunek 27 Obraz testowy

Z każdej kamery wyselekcjonowano 81 klatek filmu, czyli łączna ilość przetworzonych obrazów dla trzech kamer wynosiła 243 obrazy. Łączna ilość znaków zaznaczonych na wszystkich obrazach wynosiła 322. Część znaków na obrazach w oryginalnym rozmiarze prezentuje Tabela 7. Trzeba przyznać, że przed systemem stawiano bardzo trudne zadanie m. z powodu znacznej dysproporcji znaków. Znaki o rozmiarze mniejszym niż 20 x 20 pikseli stanowiły 45 % wszystkich znaków, a czym mniejszy znak tym gorsza jego rozdzielczość. Do pogorszenia wyników mogło przyczynić się nagrywanie obrazu w mieście, gdzie była duża ilość różnych obiektów, oraz że trasa, na której był nagrywany były filmy stanowiła co powodowało silne i dynamiczne zmiany oświetleniowe. Znaki podzielone były też ze względu na kolor. System natomiast nie rozpoznawał znaków czarnych, ponieważ stanowią one mniejszość na drodze. Najważniejszą informacją dla pracy było sprawdzenie czy zastosowanie większej ilości kamer będzie miało pozytywny wpływ na system rozpoznawania znaków. Test polegał na zliczeniu wykrytych znaków na każdej z kamer oraz na sprawdzeniu, jakie połączenie kamer jest najbardziej efektywne. Ważnym aspektem z punktu widzenia badanego problemu było zrozumienie, że znaki, których nie obejmuje kamera w dalszym ciągu istnieją w rzeczywistości, dlatego zostały one wliczane nawet do testów badających obraz z jednej czy dwóch kamer. Z powodu tego, że system dzielił się na dwa etapy – segmentacje i klasyfikacje osobnej ocenie podlegał każdy etap. Test przeprowadzano w następujący sposób. Wpierw



zliczono znaki znajdujące się na wszystkich 243 obrazach. Liczba ta była podstawą do wszystkich badań. Następnie z wszystkich znaków zliczono te, które były możliwe do rozpoznania przez prezentowany algorytm. Kolejnym krokiem było przebadanie jak radzi sobie system z odnalezieniem znaków na obrazie. Z uwagi na fakt, że wyszukiwano znaki każdego koloru osobno, również w taki sposób podzielono wyniki. Dla klasyfikacji obrazu były kluczowe trzy aspekty. Pierwszym, a zarazem najważniejszym była ilość poprawnie rozpoznanych znaków. Drugą informacją było rozpoznanie znaku jako inny znak. Ostatnim wynikiem było rozpoznanie znaku w miejscu, który nie zawierał żadnego znaku. Podczas testowania systemu warto zwrócić uwagę na fakt, że system rozpoznawania znaków ma informować o wystąpieniu znaku i w większości przypadków jesteśmy zainteresowani, ile razy znak został rozpoznany, dlatego przebadano również taką skuteczność.

Niebieskie	Czerwone	Żółte	Czarne
			




			
---	--	--	--

Tabela 7 Znaki zaznaczone na benchmarku

## Wyniki badań

Skuteczność rozpoznawania znaków drogowych za pomocą zaproponowanego systemu wyszukiwania znaków drogowych zaprezentowano w poniższych tabelach. Tabela 8 przedstawia ilość wykrywanych znaków dla systemu z parametrami przedstawionymi w akapicie **Błąd! Nie można odnaleźć źródła odwołania.**, w którym aktywne są trzy kamery. Tabela 9 przedstawia procentową skuteczność algorytmu. Omawianie należało by zacząć od poprawności wykrywania obszarów ze znakami. Na łącznie 322 zaznaczonych znaków system znalazł 160 znaków. Jak prezentuje Tabela 9 był to 49,68 procent wszystkich znaków. W trakcie wyszukania znaków wytypował 163 obszary, które nie należały do znaków. Największym problemem było znalezienie balansu między dobrze, a błędnie wykrytymi znakami. Zamysłem było poddanie klasyfikacji większej ilości obszarów, które następnie były by odrzucone podczas klasyfikacji.

	Niebieskie	Czerwone	Żółte	Łącznie
Regionów ze znakami na obrazie	209	39	74	322
Regionów ze znakami w bazie na obrazach	138	26	44	208
Właściwie znalezionych regionów	108	12	40	160
Błędnie znalezionych regionów	88	5	70	163
Właściwie rozpoznanych znaków	23	2	18	43
Rozpoznanie znaku jako innego znaku	22	4	13	39
Rozpoznanie znaku w regionie bez znaku	12	0	41	53
Ile znaków powinno być wychwyconych	77	16	24	117
Ile znaków było wychwyconych	17	2	10	29

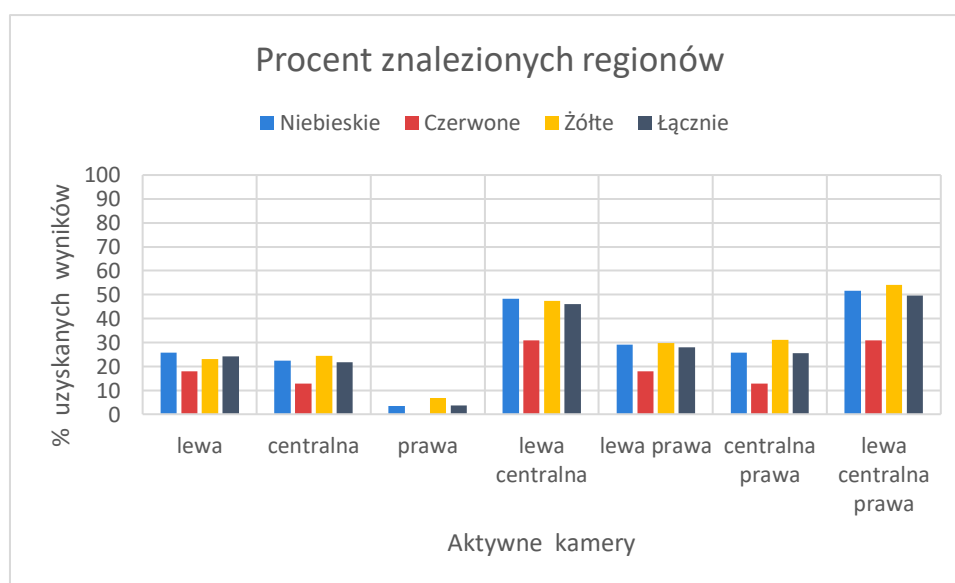
Tabela 8 Ilość znajdowanych znaków

Algorytm klasyfikacji również nie uzyskał ekscytujących wyników. Z 160 dobrze znalezionych znaków rozpoznano 43 znaków, co wyznacza skuteczność 31,25 %. Żle rozpoznano 43 znaki zostały rozpoznane jako nie te znaki jakie powinny być rozpoznane (50,46 %). Warto również dodać, że algorytm klasyfikacji nie odrzucił 53 znaków z 163 (32,51 %). Warto wspomnieć również o wynikach dla pojedynczego typu znaku, który powinien być wychwycony. Na 117 typów znaków obecnych na 240 obrazach wychwycono 29 takich pojedynczych typów znaków co daje algorytmowi skuteczność na poziomie 24,78 %.

	Niebieskie	Czerwone	Żółte	Łącznie
Regionów ze znakami w bazie na obrazach	66.02	66.66	59.45	64.59
Właściwie znalezionych regionów	51.67	30.76	54.05	49.68
Błędnie znalezionych regionów	44.89	29.41	63.63	50.46
Właściwie rozpoznanych znaków	16.66	7.69	40.9	20.67
Błędnie rozpoznanych znaków	20.37	33.33	32.5	24.37
Rozpoznanych znaków w regionie bez znaku	13.63	0	58.57	32.51
Dobrze wychwyconych znaków	22.07	12.5	41.66	24.78

Tabela 9 Parametry systemu w procentach przy użyciu trzech kamer

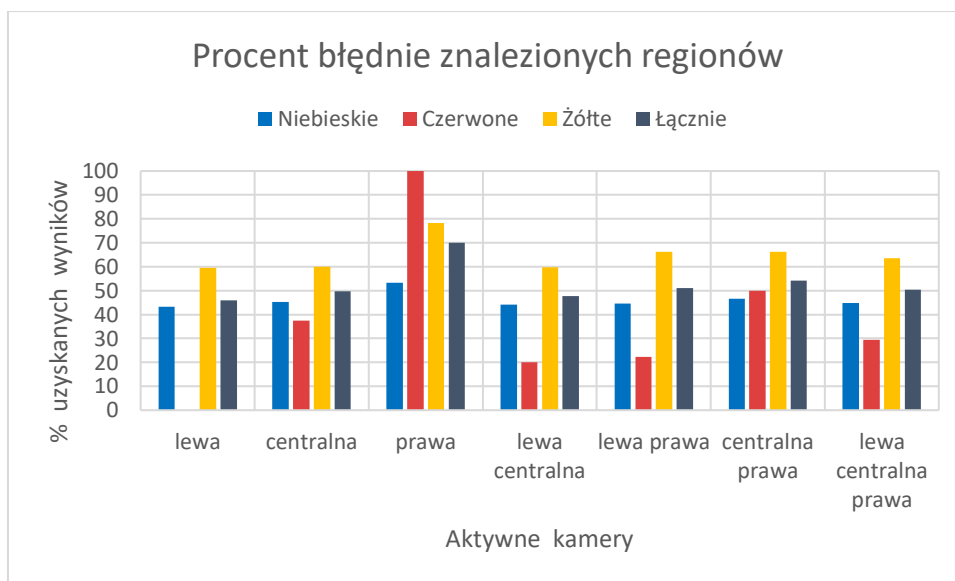
Głównym badaniem pracy magisterskiej było sprawdzenie w jaki sposób wykorzystanie wielu źródeł obrazu może wpłynąć na skuteczność rozpoznania znaków drogowych. Niżej umieszczone wykresy przedstawiają różnice w wykrywaniu znaków za pomocą obrazów uzyskanych z trzech kamer z osobna, obrazów uzyskanych z dwóch wybranych kamer i z wszystkich kamer jednocześnie. Algorytm użyty do znalezienia i rozpoznania znaków jednego koloru posiadał jednakowo dostrojone parametry. Na wykresach znaki również zostały podzielone ze względu na kolor. Zbadano jaki procent wszystkich znaków rozpoznaje każda z kamer oraz jaka była łączna skuteczność wykrywania podczas pracy dwóch i trzech kamer. Wpierw zostały omówione wyniki algorytmu zdolnego do wykrywania regionów ze znakami, a następnie wyniki rozpoznania tych regionów.



Wykres 1 Procent znalezionych regionów

Wykres 1 prezentuje jaki procent znaków zostało znalezionych. Skuteczność wykrywania regionów ze znakami przez pojedynczą kamerę okazała się najlepsza dla kamer lewej i centralnej, które były umiejscowione tak by obejmować obszar przed samochodem. Kamera prawa, która miała kąt widzenia skierowany w prawą stronę przy samodzielnej pracy nie jest w stanie zarejestrować większości znaków drogowych co pokrywa się z intuicją. Brak odnalezienia znaków czerwonych przez kamerę prawą świadczy o tym, że znak widziany pod kątem z jakiego rejestruje go kamera nie odpowiada parametrom detektora Hougha. Można było by poprawić detekcję stosując osobne parametry detekcji dla tej kamery. Połączenie kamery lewej i centralnej dwukrotnie podniosło skuteczność detekcji. Obrazy dostarczane

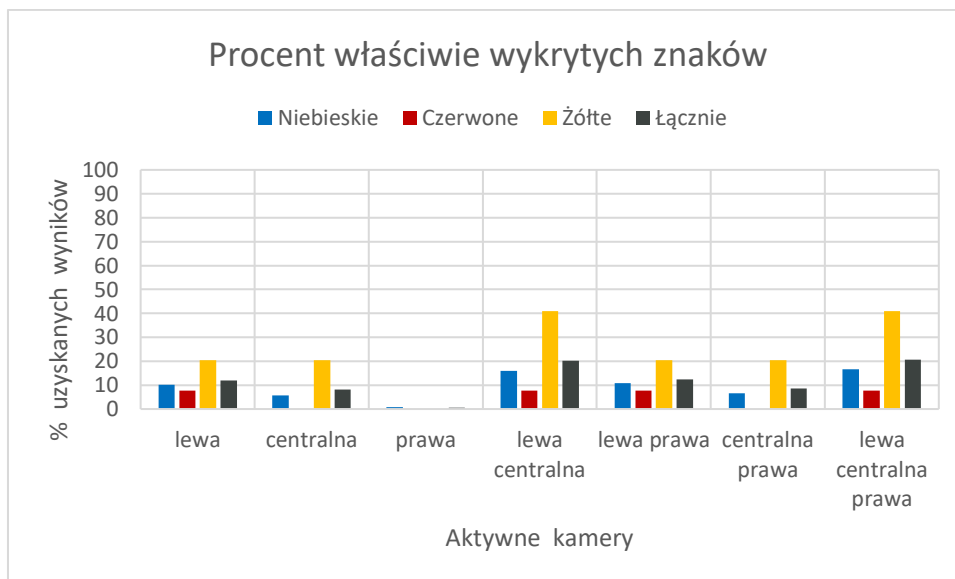
przez prawą kamerę najmniej poprawiły skuteczność wykrywania za pomocą dwóch kamer. Najlepszy wynik został uzyskany dla obrazów pochodzących z wszystkich trzech kamer. Oceniając skuteczność detekcji ze względu na kolor można stwierdzić, że znajdowanie znaków za pomocą wykrywania plam dla znaków niebieskich i żółtych było na podobnym poziomie. Detektor Hougha odnalazł mniejszy procent wszystkich znaków czerwonych.



Wykres 2 Procent nieznaalezionych regionów

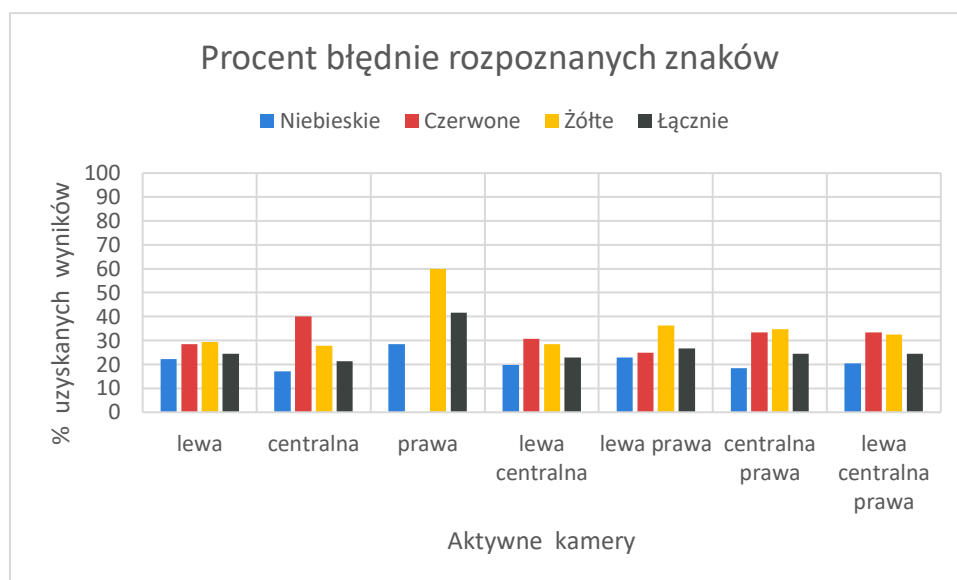
Wykres 2 przedstawia jaki procent wszystkich znalezionych regionów stanowią regiony, gdzie nie ma znaku. Można zauważyć, że detektory w dużej części znajdowały regiony, gdzie znak nie występował. Obniżenie progu detekcji powodowało mniejszą ilość błędnie wykrytych regionów, ale również obniżało ich wykrycie ogólnie. Przebadano różne metody pozwalające odrzucić region błędnie znaleziony. Takimi metodami było ponowne zbinaryzowanie obrazu za pomocą metody Otsu, użycie histogramu progowego oraz wiele indywidualnych pomysłów, lecz każdy z nich wykluczał w około 50 % regiony błędne jak i właściwe. Uznano, że regiony, gdzie znak nie istnieje będą odrzucane przez sieć neuronową na etapie klasyfikacji. Wyniki dla wykrycia regionów są następujące. Najlepiej z detekcją regionów poradziła sobie kamera po lewej stronie, gdzie łączny procent wykrycia błędnych regionów wyniósł 45,82 %. Nieco gorzej, bo około 49,64 % wszystkich regionów zostało błędnie zidentyfikowanych przez kamerę centralną. Najgorszy wynik przypadł dla kamery umiejscowionej po prawej stronie, której regiony błędnie stanowiły 70,00 % wszystkich regionów. Przy łączeniu błędów wykrywania z dwóch i trzech kamer nie zauważono znacznej różnicy. Każde połączenie osiągało wynik błędnej detekcji na podobnym poziomie około 50% lecz najmniejszym błędem była obarczona detekcja obrazów z kamery lewej i centralnej 47,70%. Określając skuteczność detektora niebieskiego należy powiedzieć, że jego błąd wynosił mniej niż połowę zidentyfikowanych obszarów. Jedynie dla kamery prawej jest nieznacznie większy. Jest to dość zadowalający wynik, ponieważ duża ilość obiektów (kosze na śmieci, auta) w mieście jest podobnego koloru niebieskiego co znaki. Detektor żółtych znaków osiągnął dużo gorszą dokładność. Ilość błędnie wykrytych regionów stanowiła dla kamery lewej i centralnej około 60 % wszystkich regionów, a dla kamery umiejscowionej po lewej stronie, aż 70 %. Najlepiej poradził sobie z odrzuceniem błędnych regionów detektor Hougha. Był skuteczny do tego stopnia, dla kamery lewej wszystkie regiony które zidentyfikował były właściwe. Niestety regiony znalezione przez kamerę prawą w 100 % okazały się błędne. Skala błędu wykrycia

mogła być szokująca, lecz warto sobie uświadomić, że liczba ta bierze się z wyniku dwóch błędnych wykryć na zero właściwych liczonych na 81 obrazach.



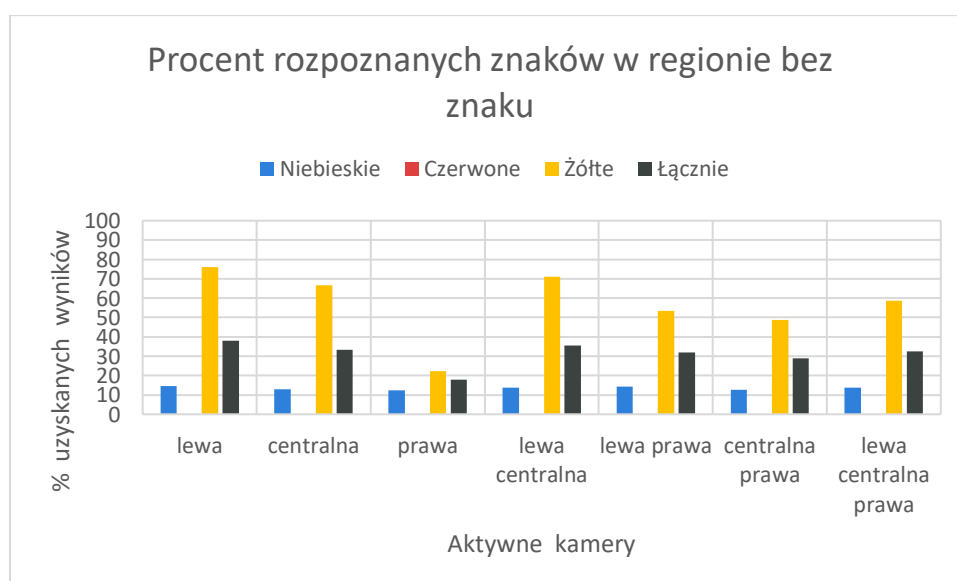
Wykres 3 Procent rozpoznanych znaków

Wykres 3 prezentuje jaki procent wszystkich znaków na obrazach możliwych do rozpoznania został rozpoznany. Do tego testu liczone były tylko znaki, które się mogła rozpoznać, czyli 64,59 % wszystkich znaków w teście. Brane były natomiast również znaki, w regionach nie znalezionych przez detektory. By znak został rozpoznany pewność sieci musiała być większa niż 99 %. Wykres prezentuje skuteczność całego systemu z jaką rozpoznaje znaki, które jest w stanie rozpoznać. Najlepszy wynik dla pojedynczej kamery uzyskała kamera po lewej stronie znajdując i rozpoznając 12,01 % znaków. Nieco gorzej prezentuje się kamera centralna, która znalazła tylko 8,17 % znaków. Kamera po prawej stronie była w stanie znaleźć tylko 0,48 % znaków. Najlepsze połączenie dwóch kamer uzyskano dla kamery lewej i centralnej, który wynosił 20,19 % wszystkich znaków. Dla kamery lewej i prawej wynik wynosił 12,50 %. Najgorsza konfiguracja dla dwóch kamer była uzyskana dla systemu zbudowanego z kamery centralnej i prawej. Najskuteczniejszy okazał się system zbudowany na trzech kamerach uzyskując wynik 20,67 % wszystkich znaków. Dzielic wykrywanie znaków na kolory należy powiedzieć, że największy procent poprawnie rozpoznanych znaków należał do znaków, żółtych. Ma to duży związek z tym, że system potrafił rozpoznać tylko dwa typy znaków. Znaki niebieskie system rozpoznawał znacznie gorzej. Największe problemy system miał z poprawnym rozpoznaniem znaków czerwonych. Wiązało się to z podobnym wyglądem znaków ograniczenia prędkości.



Wykres 4 Procent źle rozpoznanych znaków

Wykres 4 przedstawia jaki procent znaków, zostały błędnie rozpoznane z wszystkich poprawnie wyznaczonych regionów. Inaczej mówiąc jaki jest procent omyłności systemu. Jak zostało wcześniej wspomniane by znak został poprawnie sklasyfikowany pewność klasyfikacji musiała być większa niż 99 %. Dla pojedynczej kamery najmniej myliła się kamera centralna, gdzie poziom błędu wynosił 21,42 %. Trochę gorzej znaki rozróżniała kamera lewa, gdzie osiągnięto 24,35 procent błędnego rozpoznania. Najbardziej omylna była kamera po prawej stronie, gdzie 41,66 % znaków było błędnie sklasyfikowanych. Przy połączeniu dwóch i trzech kamer nie było znacznych różnic w skuteczności klasyfikacji. Połączenie kamery lewej i centralnej uzyskało najlepszy wynik na poziomie 22,97 %, a najwięcej błędów, bo 26,66 % wynikało z połączenia kamery lewej i prawej. Wyniki były spowodowane tym, że najwięcej regionów było znajdowanych na obrazach z kamery lewej i centralnej.

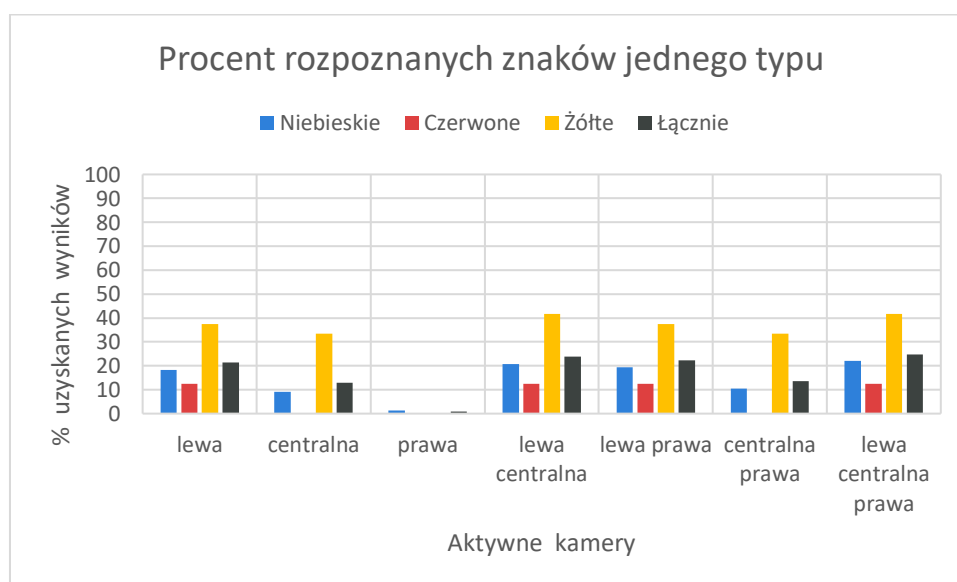


Wykres 5 Procent rozpoznanych znaków w regionie bez znaku

Dość istotny czynnik ze względu na niezawodność systemu przedstawia Wykres 5. Jest to procent znaków, które zostały rozpoznane w błędnie znalezionym regionie. Starano się głównie



by przedstawiona metoda klasyfikacji nie rozpoznawała tych regionów. Niestety jak zostało opisane przy okazji błędnie znajdowanych regionów nie udało się tego uniknąć. Była to największa wada systemu, z którą obniżała jego skuteczność. Zwiększenie skuteczności klasyfikacji obrazów wiązało się z tym, że znaczna liczba regionów bez znaku zostanie wykrytych. Mimo, że sieć powinna mieć co najmniej 99 % pewności co do rozpoznanego znaku nie spowodowało to odrzucenia błędnych regionów. Najmniej, bo 17,87 % błędnych znaków zostało rozpoznanych na obrazach dostarczonych przez kamerę centralną. Błędne rozpoznanie przez kamerę centralną wynosiło 33,33 %. Najgorszy wynik dla pojedynczej kamery uzyskała kamera po lewej stronie - 41,66 %. Połączenie kamery centralnej i prawej dało najlepszy ogólny wynik w postaci 28,86 % regionów bez znaku błędnie sklasyfikowanych przez sieć. Wyniki dla kamery lewej i centralnej, lewej i prawej oraz wszystkich trzech osiągnęły wynik na podobnym poziomie odpowiednio 35,55 %, 31,91 %, 32,51 %. Warto również wspomnieć o skuteczności rozpoznawania obrazów w różnych kolorach. Dla obrazów błędnie znalezionych przez detektor Hougha żaden nie został rozpoznany przez sieć co jest idealnym wynikiem. Mało regionów dostarczonych przez detektor niebieski zostało też sklasyfikowanych przez sieć nauczoną do rozpoznawania znaków. Największym problemem była klasyfikacja regionów przez sieć wyszkoloną do rozpoznawania znaków żółtych. Powodem mogło być nauczanie sieci rozpoznawania tylko dwóch typów znaków drogowych.

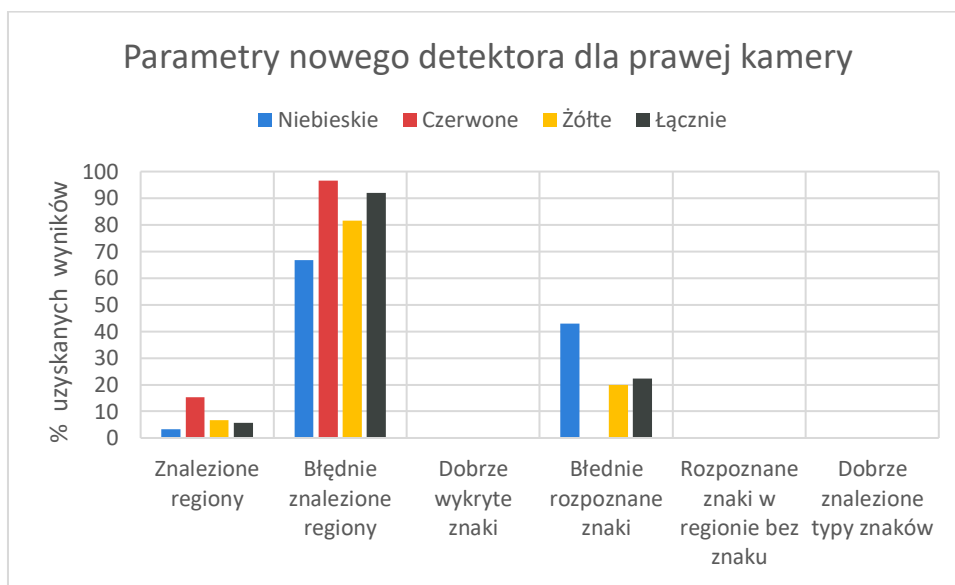


*Wykres 6 procent rozpoznanych znaków jednego typu*

Zdecydowano się uwzględnić jeszcze jeden czynnik określający przydatność algorytmu wykrywania znaków. Czynnikiem tym było wykrycie pojedynczego typu znaku. Nie uwzględniano ilości wystąpienia znaku, lecz sam fakt jego wystąpienia na którymkolwiek z trzech obrazów. Dodatkowo liczone były tylko znaki, które system był nauczony rozpoznawać. Z pojedynczych kamer najbardziej skuteczna okazała się kamera po lewej stronie, której skuteczność wynosiła 21,36 %. Kamera środkowa była w stanie rozpoznać tylko 12,82 % znaków. Ta po prawej stronie wykryła znikomy procent znaków 0,85 %. Informacja uzyskana z kamery lewej również wniosła najwięcej informacji do systemu, w którym użyto dwie lub trzy kamery. System złożony z trzech kamer uzyskał najwyższą skuteczność wykrycia. Biorąc pod uwagę kolory rozpoznawanych znaków to najskuteczniejszy okazał się detektor znaków żółtych. System znajdujący znaki niebieskie zajmował drugie miejsce i był połowę gorszy. System rozpoznający znaki czerwone okazał się najmniej skuteczny. Należy jednak zaznaczyć,

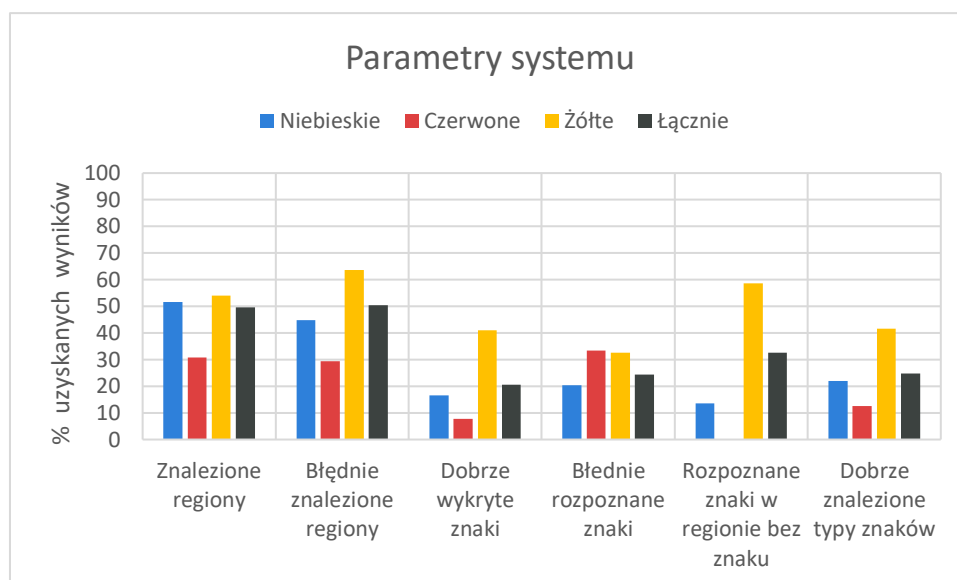
że rozpoznanie znaków niebieskich i czerwonych było o wiele trudniejsze od rozpoznania znaków koloru żółtego, ze względu na ilość uwzględnianych typów znaków. Dla koloru niebieskiego wynosiła ona 13 typów, dla koloru czerwonego 17 typów a dla koloru, żółtego tylko 2 typy. Dodatkowo znaki ograniczenia prędkości, które stanowiły znaczącą część zbioru są bardzo podobne do siebie, co powodowało dużą pomyłkę.

By zaradzić problemowi wyszukiwania znaków przez prawą kamerę zdecydowano się stworzyć i dostroić osobny detektor dla tej kamery. Po stworzeniu takiego detektora przetestowano jego działanie. Wyniki zostały zaprezentowane Wykres 7.



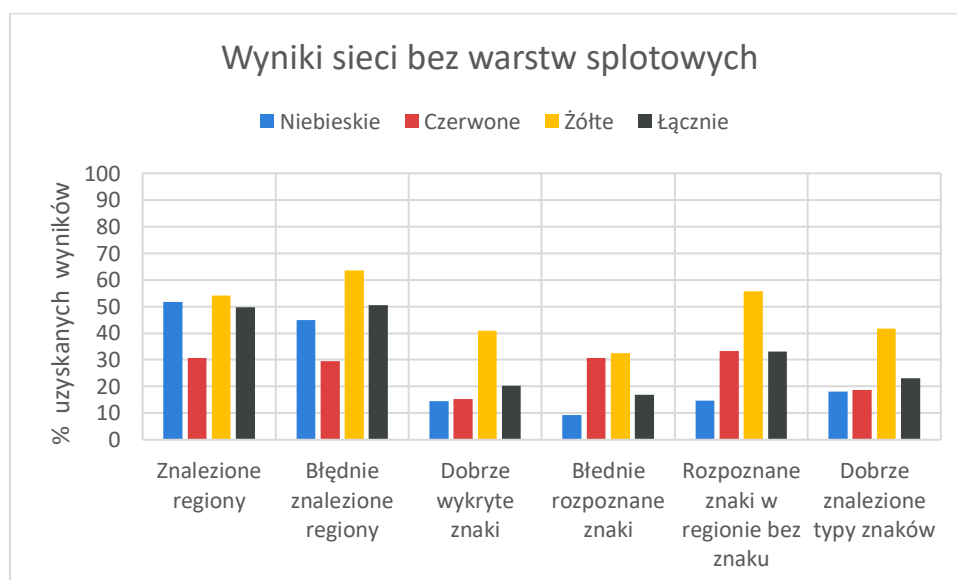
*Wykres 7 Parametry nowego detektora dla prawej kamery*

Po przeanalizowaniu wykresu można stwierdzić, że stworzenie nowego detektora nie poprawiło parametrów systemu. Kilka regionów faktycznie zostało znalezionych poprawnie, lecz w dalszym ciągu jest to niewielki odsetek całości. Sieć neuronowa nie jest w stanie dobrze rozpoznać znaków pod kątem jakim są one uchwycone przez prawą kamerę. Błędnie znalezione regiony stanowią zbyt dużą część wszystkich znalezionych regionów by można było powiedzieć, że zamontowanie kamery z prawej strony miało by sens.



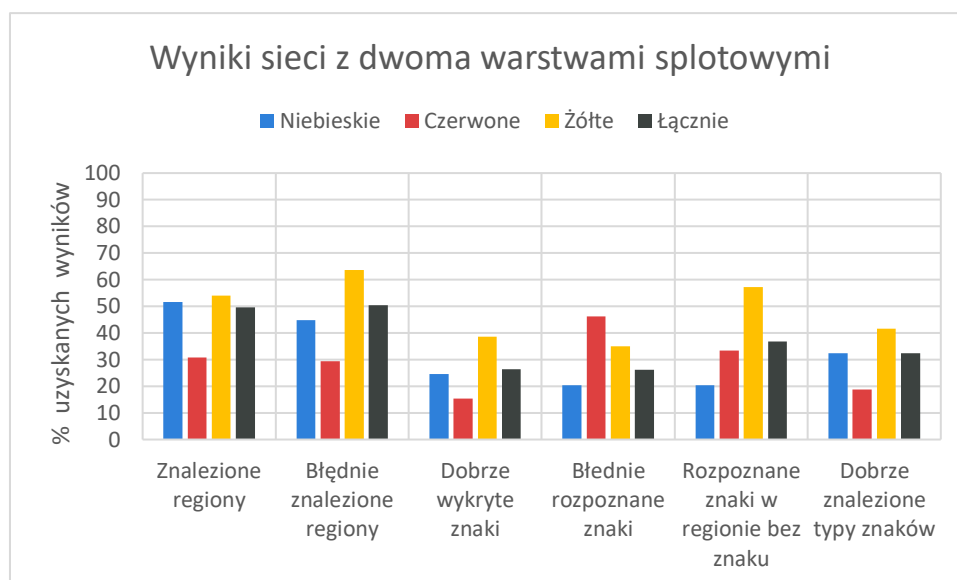
Wykres 8 Parametry systemu w procentach przy użyciu trzech kamer

Wykres 8 prezentuje razem wszystkie parametry systemu dla wykrywania znaków drogowych za pomocą trzech kamer. Jest to graficzne przedstawienie danych z Tabela 9. Wykres służył do porównania skuteczności systemu innych konfiguracji przedstawionych poniżej.



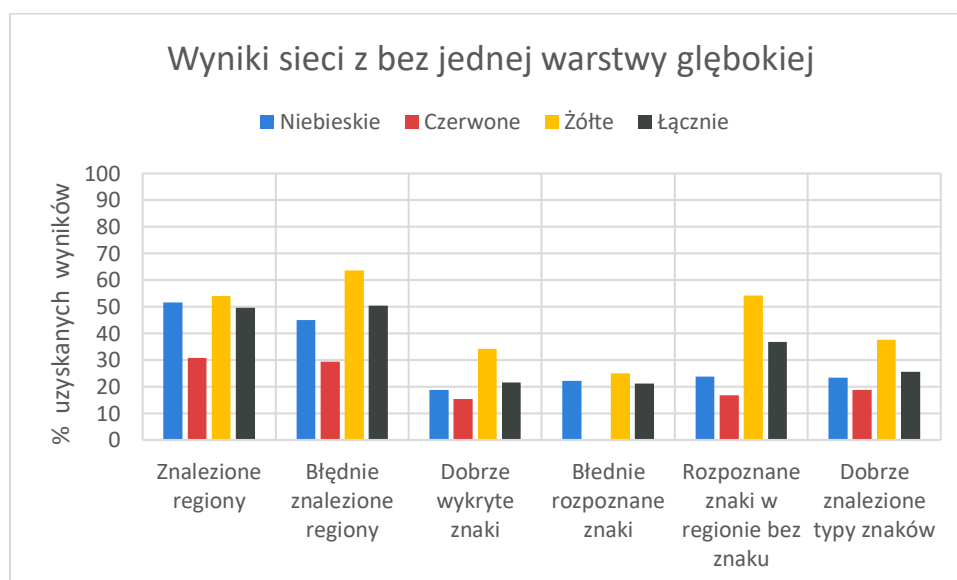
Wykres 9 - Zastosowano sieć bez pierwszej warstwy splotowej

Wykres 9 prezentuje parametry systemu dla sieci bez pierwszej warstwy splotowej. Można zauważyć, że dla takiej sieci wzrosła rozpoznawalność znaków czerwonych i lecz również wzrosła ilość rozpoznanych znaków w regionach bez znaku. Zastosowanie takiej konfiguracji sieci pogorszyło jej działanie, dlatego została odrzucona. Warto dodać, że prędkość szkolenia sieci skróciła się o połowę. Następnie wyciągnięto wniosek, by zastosować sieć, która posiadała dwie warstwy splotowe. Wynik takiej sieci przedstawia Wykres 10.



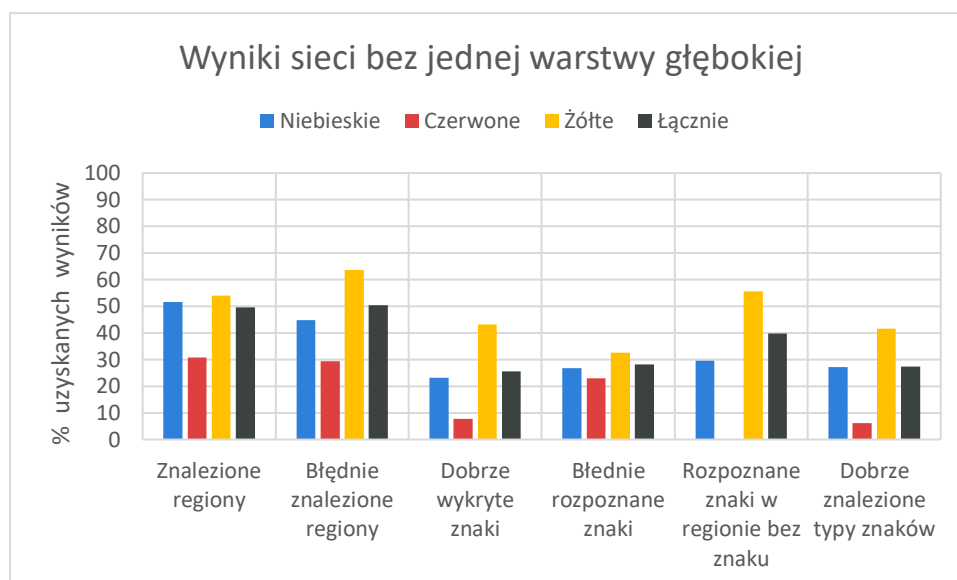
*Wykres 10 parametry sieci z dwoma warstwami spłotowymi*

Rezultat zastosowania takiej sieci nie sprostął oczekiwaniom. Faktycznie ilość rozpoznawanych znaków wzrosła jednak podwyższyła się również ilość błędnie sklasyfikowanych znaków oraz znaków znalezionych w regionie bez znaku. Dołożenie kolejnej warstwy spłotowej nie poprawiło wyników klasyfikacji. Uznano, że skoro najlepszy rezultat osiąga jedna warstwa spłotowa warto sprawdzić różne modyfikacje warstw głębokich. Zaczęto od usunięcia jednej warstwy głębokiej.



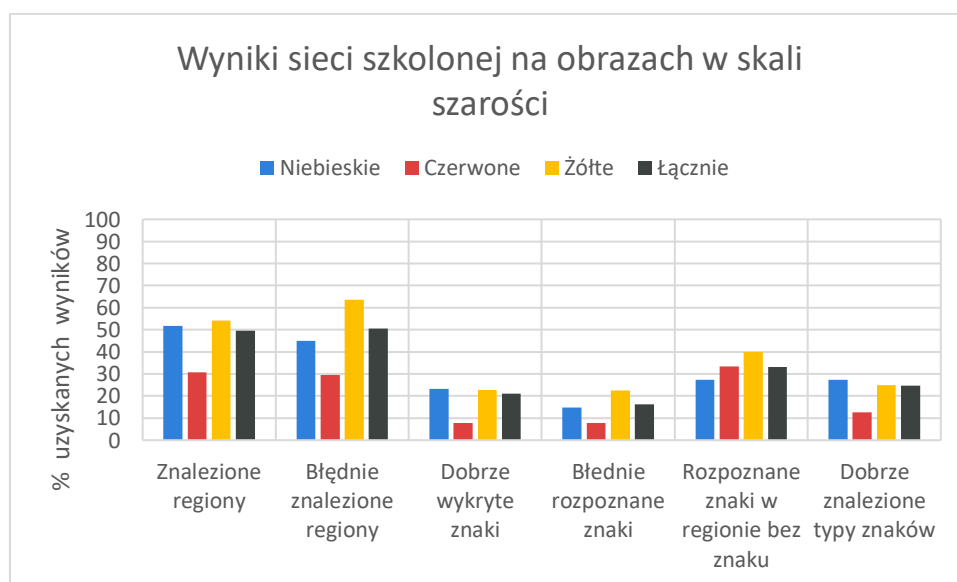
*Wykres 11 Wyniki sieci z bez jednej warstwy głębokiej*

Wykres 11 prezentuje osiągnięte wyniki sieci pozbawionej jednej warstwy głębokiej. W rezultacie zmalała ilość znaków dobrze i błędnie znajdowanych. Następną zmianą konfiguracji polegała na dołożeniu jednej warstwy głębokiej.



*Wykres 12 Wyniki sieci bez jednej warstwy głębokiej*

Wykres 12 prezentuje wynik sieci, która posiadała o jedną warstwę głęboką mniej. Podobnie jak w poprzednich przypadkach sieć rozpoznała więcej znaków przy okazji zwiększając ilość znaków błędnie rozpoznanych. Zmniejszenie głębokości sieci nie poprawiło wyników. Uznano, że mądrym pomysłem może być nauczanie sieci rozpoznawania obrazów w skali szarości. Dodatkowo by poprawić wygląd obrazów zdecydowano się wyrównać ich histogram.

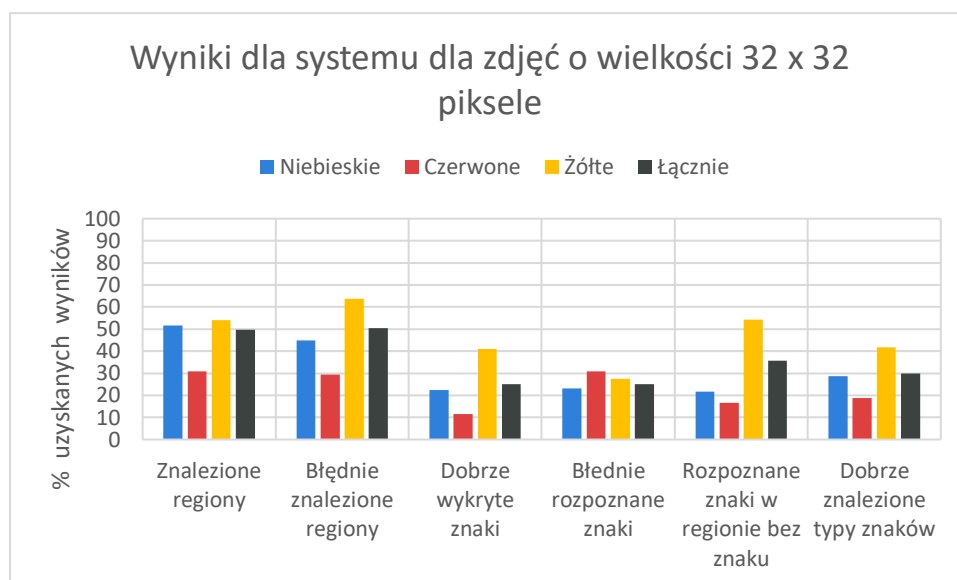


*Wykres 13 Wyniki sieci szkolonej na obrazach w skali szarości*

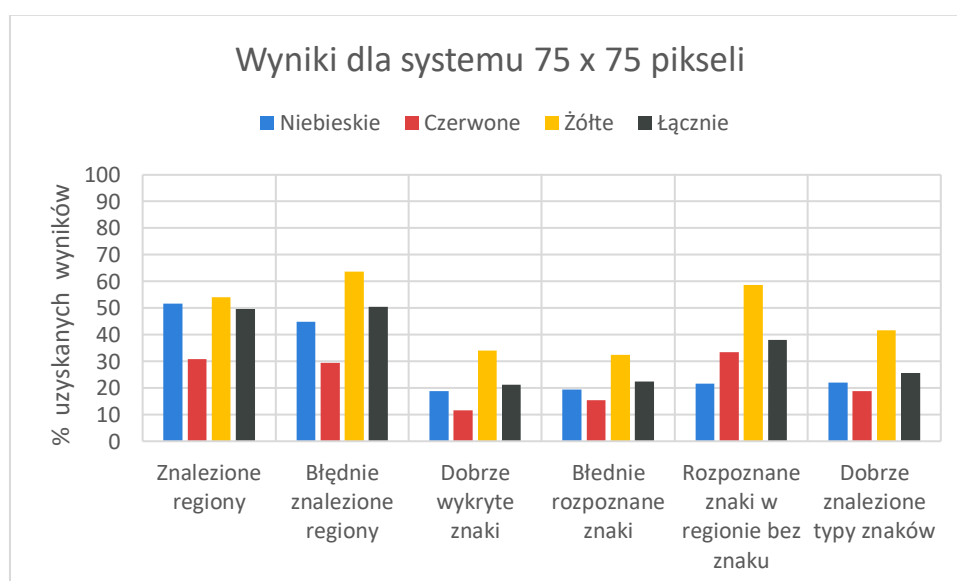
Okazało się, że przy szkoleniu sieci dla obrazów w skali szarości jej wyniki się pogorszyły. Wzrosła ilość błędnie rozpoznanych znaków w regionie bez znaku. Zmalała jednak ilość błędnie rozróżnionych znaków oraz ogólna liczba rozróżnianych znaków. Dowodzi użyteczności rozróżniania barw dla problemu rozpoznawania znaków.

Kolejnym testem było sprawdzenie czy zmiana skali obrazu wejściowego będzie miała pozytywny wpływ na rozpoznawalność. W tym celu wytrenowano sieć dla obrazów o wielkości 32x32 oraz 75x75 piksele. Wyniki prezentują odpowiednio Wykres 12 oraz Wykres 13.





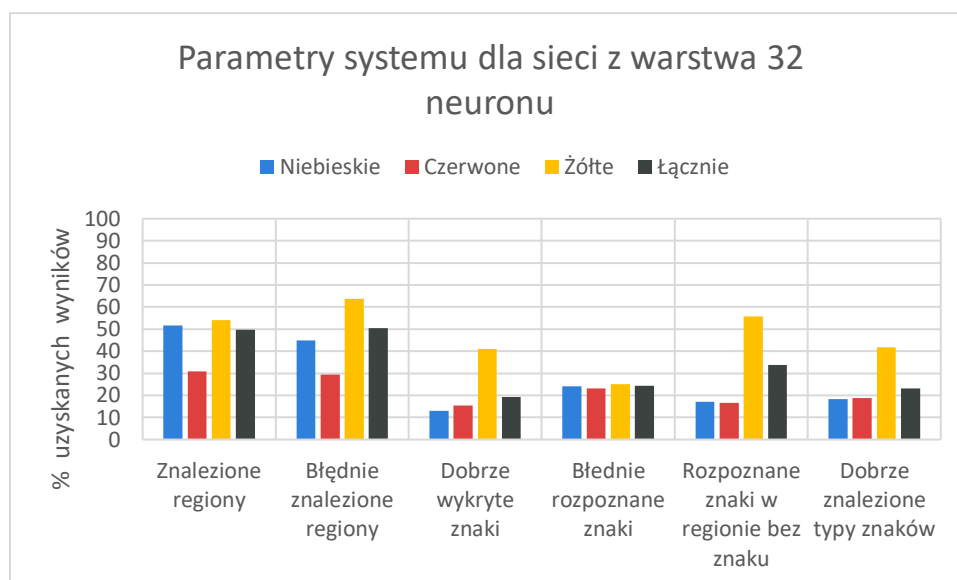
Rysunek 28 Wyniki dla systemu dla zdjęć o wielkości 32 x 32 piksele



Rysunek 29 Wyniki dla systemu dla zdjęć o wielkości 75 x 75 piksele

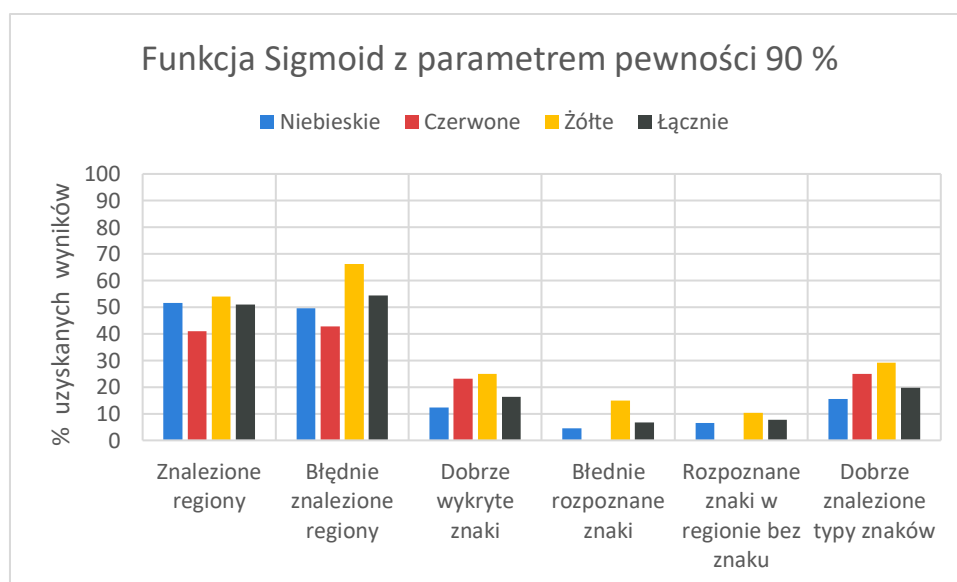
Opisując wyniki sieci szkolonej na innych rozmiarach obrazu można rzec, że zmiana wielkości nie poprawia parametrów rozpoznawalności. Rozmiar obrazu 50 x 50 pikseli był w pełni odpowiedni dla badanego problemu.

Należało również rozpatrzyć zmianę ilości neuronów w warstwach. Zdecydowano się stworzyć sieć, która złożono by była z warstwy konwolucyjnej oraz trzech warstw głębokich posiadających nie 64, ale 32 neurony. Taką sieć poddano testom, a jego wyniki przedstawia Wykres 14 Parametry systemu dla sieci z warstwa 32 neuronu



Wykres 14 Parametry systemu dla sieci z warstwa 32 neuronu

Zmiana ilości neuronów sieci również nie miała pozytywnego wpływu na rozpoznawalność znaków. Uznano, że poprawy parametrów rozpoznawania należy szukać zmieniając inny parametr. Pomysłem było zastosowanie innej funkcji aktywacji ostatniej warstwy neuronów. Zamiast funkcji *Softmax* użyto funkcji *Sigmoid*. Okazało się, że taka zmiana przyniosła korzyść w postaci odrzucenia znacznej liczby błędnych regionów. Uznano by zmniejszyć minimalny procent jaki musi osiągnąć sieć by znak został uznany za dobrze sklasyfikowany. Nowa wartość minimalnej pewności dla każdego z kolorów wynosiła 90 %. Wyniki Takiego rozpoznania przedstawi Wykres 15 Funkcja Sigmoid z parametrem pewności 90 %.



Wykres 15 Funkcja Sigmoid z parametrem pewności 90 %

Można zauważyć, że rozpoznanie obrazu dla sieci z funkcją aktywacji *Sigmoid* zmalało z 20.67 % do 16.34 %. Była to, rzecz jak najbardziej dopuszczalna mając na względzie, że błędne rozpoznanie i rozpoznanie znaku w miejscu bez znaku spadło odpowiednio z 24.37 % do 6.7 % i z 32.51 % do 7.65 %. Był to wynik bardzo satysfakcjonujący. Należy również dodać, że rozpoznanie dobrego typu znaku wzrosło przy tym do 19.65 %.

## Podsumowanie

Praca dyplomowa miała na celu badanie metod rozpoznawania znaków drogowych z wykorzystaniem wielu źródeł obrazu. Z całą pewnością można stwierdzić, że zadanie zostało wykonane. W pierwszej części obszernie opisano różne metody, które zostały używane do rozwiązywania problemu rozpoznawania znaków. Opisano metody segmentacji obrazu oraz jego klasyfikacji. Dodatkowo przedstawiono możliwości kilka możliwości jakie można uzyskać większej ilości kamer. Przywołano artykuł, który badał skuteczność fuzji obrazów dostarczonych przez kamery różnego typu oraz kilka sposobów tworzenia obrazu panoramicznego dostarczonego przez kamery jednego typu. W części praktycznej zaprojektowano i zaprogramowano system zdolny do rozpoznania znaków drogowych oraz przebadano jego możliwości. Sprawdzono również jakie są wady i zalety łączenia obrazów w panoramę. Stworzono autorskie metody znajdowania znaków drogowych niebieskich, czerwonych i żółtych. By sklasyfikować obrazy dostarczone przez segmentację stworzono trzy konwolucyjne sieci neuronowe. Architekturę sieci dobrano poprzez eksperymenty. By nauczyć sieć została stworzona specjalna baza danych ze zdjęć z pobranych z istniejących baz oraz z własnych zdjęć znaków drogowych. Ważnym aspektem było przebadanie skuteczności wykrywania znaków drogowych. W tym celu stworzono autorski test, który umożliwił ocenę systemu. Następnie sprawdzono skuteczności wykrywania znaków za pomocą różnych permutacji kamer i różnych parametrów sieci neuronowej. Na samym końcu podsumowano wyniki pracy.

System przedstawiony w pracy nie dorównuje skutecznością systemów przedstawionych w pracach naukowych. Obarczony jest dużym błędem, który uniemożliwia jego przydatność w realnych sytuacjach. Najwięcej błędów do systemu zostało wprowadzonych przez kamerę umiejscowioną z prawej strony samochodu. Jej przydatność nie została zwiększona poprzez zbudowanie osobnego detektora. Najefektywniejsze były kamery umiejscowione na wprost samochodu. Ważnym zadaniem, w którym częściowo zostało osiągnięte było odrzucenie regionów błędnie znalezionych przez algorytm segmentacji przez sieć neuronową. Sieć przez zastosowanie innej funkcji aktywacji neuronów znacznie poprawiła swoje działanie. Najlepsza popraw parametrów nastąpiła dla znaków niebieskich i czerwonych. Dla znaków żółtych wynik był nieco gorszy. Można domniemywać, że powodem tego były tylko dwie klasy znaków jakie sieć rozpoznawała. Niestety Polska stosuje znaki ostrzegawcze w kolorze innym niż pozostałe kraje, dlatego by nauczyć sieć rozpoznawać takie znaki należało zbudować własną bazę znaków. Tworzenie obrazów ze zdjęć wokół samochodu jest bezcelowe, ponieważ mogą one przyczyniać się do deformacji znaków. Tworzenie panoramy również zmniejsza również możliwość znalezienia znaku przez przysłonięcie części wspólnej obrazu. Należy jednak pamiętać, że prezentowany test był bardzo trudny. Znaki były bardzo zróżnicowane pod względem wielkości. Część z nich była również pokazana tylko częściowo na obrazie. Dodatkowo samochód poruszał się pod wieloma kątami względem słońca co czasami wprowadzało zakłócenia na obrazach. Test również był robiony w mieście, gdzie wiele obiektów jest w podobnej kolorystyce co znaki. Z powyższych powodów nie należy martwić się złymi wynikami systemu.

## Bibliografia

- [1] Y. Aoyagi i T. Asakura, „A study on traffic sign recognition in scene image using genetic algorithms and neural networks,” w *International Conference on Industrial Electronics, Control, and Instrumentation*, Taipei, Taiwan, Taiwan, 9 sierpień 1996.
- [2] Q. Hu, S. Paisitkriangkrai, C. Shen, A. v. d. Hengel i F. Porikli, „Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework,” *IEEE Transactions on Intelligent Transportation Systems*, tom 17, nr 7, pp. 1002-1024, kwiecień 2016.
- [3] A. Mogelmose, M. M. Trivedi i T. B. Moeslund, „Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey,” *Transactions on Intelligent Transportation Systems*, tom 13, nr 4, pp. 1484-1497, grudzień 2012.
- [4] M. Kondej, B. Putz i M. Bartyś, „Szybki algorytm dopasowania obrazów dla potrzeb fuzji w czasie rzeczywistym,” 25 listopad 2010. [Online]. Available: <https://automatykaonline.pl/Artykuly/Sterowanie/szybki-algorytm-dopasowania-obrazow-dla-potrzeb-fuzji-w-czasie-rzeczywistym>. [Data uzyskania dostępu: 11 03 2019].
- [5] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal i P. Suetens, „Multimodality image registration by maximization of mutual information,” *Transactions on Medical Imaging*, tom 16, nr 2, p. 187–198, kwiecień 1997.
- [6] B. Zitová i J. Flusser, „Image registration methods, a survey. Image and Vision Computing,” *Image and Vision Computing*, tom 21, nr 11, p. 977–1000, październik 2003.
- [7] J. Heather i M. Smith, „New adaptive algorithms for real-time registration and fusion of multimodal imagery,” 5 maj 2010. [Online]. Available: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/7689/768908/New-adaptive-algorithms-for-real-time-registration-and-fusion-of/10.1117/12.850029.short?SSO=1>. [Data uzyskania dostępu: 15 marzec 2019].
- [8] D. G. Lowe, „Object Recognition from Local Scale-Invariant Features,” University of British Columbia, Vancouver, B.C., Canada, 1999.
- [9] E. Lee, S.-S. Lee, Y. Hwang i S.-J. Jang, „Hardware implementation of fast traffic sign recognition for intelligent vehicle system,” w *2016 International SoC Design Conference (ISOCC)*, Jeju, South Korea, 23-26 październik 2016.
- [10] D. Deguchi, M. Shirasuna, K. Doman, I. Ide i H. Murase, „Intelligent traffic sign detector: Adaptive learning based on online gathering of training samples,” w *2011 IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, 05 lipiec 2011.
- [11] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno i F. Lopez-Ferreras, „Road-Sign Detection and Recognition Based on Support Vector

- Machines,” *IEEE Transactions on Intelligent Transportation Systems*, tom 8, nr 2, pp. 264 - 278, 04 czerwiec 2007.
- [12] H. Liu, D. Liu i J. Xin, „Real-time recognition of road traffic sign in motion image based on genetic algorithm,” w *Proceedings. International Conference on Machine Learning and Cybernetics*, Beijing, China, 4-5 grudzień 2002.
- [13] A. d. l. Escalera, J. M. Armingol i M. Mata, „Traffic sign recognition and analysis for intelligent vehicles,” *Image and Vision Computing*, tom 21, nr 3, pp. 247-258, marzec 2003.
- [14] P. Sermanet i Y. LeCun, „Traffic sign recognition with multi-scale Convolutional Networks,” w *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, USA, 21 lipiec 2011.
- [15] G. Wang, G. Ren, Z. Wu, Y. Zhao i L. Jiang, „A hierarchical method for traffic sign classification with support vector machines,” w *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, 4-9 sierpień 2013.
- [16] Y. Freund i R. E. Schapire, „Experiments with a New Boosting Algorithm,” w *Proceedings of the Thirteenth International Conference*, New Jersey, 1996.
- [17] Y. Yang, H. Luo, H. Xu i F. Wu, „Towards Real-Time Traffic Sign Detection and Classification,” *IEEE Transactions on Intelligent Transportation Systems*, tom 17, nr 7, pp. 2022 - 2031, czerwiec 2016.
- [18] A. d. l. Escalera, L. E. Moreno, M. A. Salichs i M. Armingol, „Road Traffic Sign Detection and Classification,” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, tom 44, nr 6, pp. 848 - 859, grudzień 1997.
- [19] R. Duda i P. Hart, „Use of the Hough Transformation To Detect Lines and Pictures,” Stanford Research Institute, Menlo Park, California, 1972.
- [20] I. M. Creusen, R. G. Wijnhoven, E. Herbschleb i P. D. With, „Color exploitation in hog-based traffic sign detection,” w *International Conference on Image Processing*, Hong Kong, 26-29 wrzesień 2010.
- [21] P. Sermanet, D. Eigen, Z. Xiang, M. Michael, F. Rob i L. Yann, „Integrated recognition localization and detection using convolutional networks,” Courant Institute of Mathematical Sciences, New York, NY, USA, 21 grudzień 2013.
- [22] Z. Huang, Y. Yu, J. Gu i H. Liu, „An Efficient Method for Traffic Sign Recognition Based on Extreme Learning Machine,” *IEEE Transactions on Cybernetics*, tom 47, nr 4, pp. 920 - 933, 14 marzec 2016.
- [23] T. Ojala, M. Pietikäinen i D. Harwood, „Performance evaluation of texture measures with classification based on Kullback discrimination of distributions,” w *Proceedings of 12th International Conference on Pattern Recognition*, Jerusalem, Israel, Israel, 9-13 październik 1994.



- [24] A. Escalera, J. M. Armingol i M. A. Salichs, „Traffic Sign Detection for Driver Support Systems,” Systems Engineering and Automation Division, Universidad Carlos III de Madrid, Leganés, Madrid, Spain, czerwiec 2001.
- [25] P. Viola i M. Jones, „Robust Real-time Object Detection,” w *Second International Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, 13 sierpień 2001.
- [26] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer i T. Koehler, „A system for traffic sign detection, tracking, and recognition using color, shape, and motion information,” w *IEEE Proceedings. Intelligent Vehicles Symposium*, Las Vegas, NV, USA, USA, 6-8 czerwiec 2005.
- [27] N. Barnes i A. Zelinsky, „Real-time radial symmetry for speed sign detection,” w *IEEE Intelligent Vehicles Symposium*, Parma, Italy, Italy, 14-17 czerwiec 2004.
- [28] G. Loy i N. Barnes, „Fast shape-based road sign detection for a driver assistance system,” w *International Conference on Intelligent Robots and Systems*, Sendai, Japan, 28 wrzesień 2004.
- [29] T. Szymczyk, „Metoda dopasowania wzorców w rozpoznawaniu obrazów – ograniczenia, problemy i modyfikacje,” w *Automatyka*, Lublin, Wydział Elektrotechniki i Informatyki, Politechnika Lubelska, 2008, pp. 449 - 462.
- [30] C.-Y. Fang, S.-W. Chen i C.-S. Fuh, „Road sign detection and tracking,” *Transactions on Vehicular Technology*, tom 52, nr 5, pp. 1329 - 1341, 23 wrzesień 2003.
- [31] P. S. Miguel i A. R. Alastair, „Using self-organising maps in the detection and recognition of road signs,” *Image and Vision Computing*, tom 6, nr 27, pp. 673-683, maj 2009.
- [32] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li i S. Hu, „Traffic-Sign Detection and Classification in the Wild,” w *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27-30 czerwiec 2016.
- [33] F. Zaklouta, B. Stanculescu i O. Hamdoun, „Traffic sign classification using K-d trees and random forests,” w *Proceedings of International Joint Conference of Neural Networks*, San Jose, California, USA, 31 lipiec 2011.
- [34] M. Mathias, R. Timofte, R. Benenson i L. V. Gool, „Traffic sign recognition-how far are we from the solution?,” w *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, 4-9 sierpień 2013.
- [35] P. Dollár, Z. Tu, P. Perona i S. Belongie, „Integral Channel Features,” Dept. of Electrical Engineering, Los Angeles, Ca, USA, 2009.
- [36] R. Kastner, T. Michalke, T. Burbach, J. Fritsch i C. Goerick, „Attention-based traffic sign recognition with an array of weak classifiers,” w *2010 IEEE Intelligent Vehicles Symposium*, San Diego, CA, USA, 21-24 czerwiec 2010.

