Tomasz Kaczor, Małgorzata Kuchta

Państwowa Wyższa Szkoła Zawodowa w Raciborzu, Instytut Techniki

SYSTEM ROZPOZNAWANIA ZNAKÓW DROGOWYCH NA ZDJĘCIACH CYFROWYCH

Streszczenie (abstrakt): W artykule przedstawiono system rozpoznawania wybranych znaków drogowych, budowę aplikacji, sposób działania poszczególnych składowych systemu oraz wykorzystane algorytmy. Przeprowadzone zostały testy aplikacji na specjalnie do tego celu wykonanych zdjęciach cyfrowych oraz zostały zaprezentowane uzyskane wyniki.

Słowa kluczowe: systemy wizyjne, rozpoznawanie znaków drogowych

RECOGNITION SYSTEM OF ROAD SIGNS IN DIGITAL PHOTOS

Abstract: The article presents a system for recognizing road signs, building applications, operating methods of individual components of the system and applied algorithms. Application tests were dene on specially made digital photos. The results of recognition (efficiency) are presented.

Keywords: vision systems, recognition of road signs

1. Wstęp

Coraz częściej rozwiązania techniczne wkraczają w nasze życie, ułatwiając nam wykonywane pracy zawodowej, dostarczając rozrywki lub wspomagając w codziennych obowiązkach. Jedną z bardzo prężnie rozwijających się dziedzin przemysłu jest motoryzacja, która wprowadza z każdym nowym modelem samochodu zaawansowane systemy wspomagające kierowcę, dążące do poprawy komfortu jazdy, wspomagania decyzji oraz do poprawy bezpieczeństwa. Na chwilę obecną testowane są rozwiązania autonomicznych samochodów. Nowoczesny samochód to coraz więcej elektroniki wspomagającej pracę kierowcy i dbającej o jego bezpieczeństwo.

W przeciągu ostatnich kilku lat systemy inteligentne w samochodach stopniowo stają się coraz bardziej dostępne i rozbudowane. Szczególnie dużym zainteresowaniem cieszą się systemy wspomagające kierowcę. Możemy tu wyróżnić systemy automatycznego hamowania, kontrolujące sposób utrzymania kierunku jazdy, systemy statycznego doświetlania zakrętów oraz wzywające służby ratownicze w przypadku zdarzenia drogowego. Kilka rozwiązań wykorzystywanych w dzisiejszych samochodach bazuje na systemach wizyjnych. Są to miedzy innymi np. System Night Vision (asystent wspomagania jazdy nocnej)

zastosowany przez takie marki jak Audi, BMW i Mercedes-Benz, system adaptacyjnej regulacji prędkości jazdy (Mercedes-Benz, Audi), systemy wspomagające parkowanie i manewrowanie pojazdem, układ utrzymania pasa ruchu, system rozpoznający znaki drogowe czy układ HUD (Head-Up Display) umożlwiający wyświetlanie przydatnych informacji na szybie.

Znaki drogowe dostarczają kierowcom wielu informacji dotyczących dopuszczalnej prędkości i kierunku jazdy oraz ostrzeżeń pozwalających wcześniej przygotować się do sytuacji które mogą mieć miejsce na drodze. Mając na uwadze jak ważne jest bezpieczeństwo na drogach, opracowano systemy wspomagające kierowcę, których celem jest ułatwienie prowadzenia samochodu. Jednym z takich systemów jest system rozpoznawania znaków drogowych.

Znaki drogowe mają określone kształty i kolory a wiele z nich posiada również symbol, który może służyć do ich identyfikacji. Znaki zazwyczaj stanowią silny kontrast z tłem, na którym się znajdują, natomiast kształty i kolory są tak dobrane, żeby kierowcy mogli bez trudu odróżnić je od otoczenia. Niektóre znaki są do siebie bardzo podobne np. znaki ograniczenia prędkości. Duży wpływ na identyfikację znaków może mieć pogoda i zmienne oświetlenie, w którym znaki stają się mało czytelne lub niewidoczne.

W artykule przedstawiona zostanie aplikacja wykrywająca i rozpoznawająca wybrane znaki drogowe na zdjęciach, wykonana w środowisku MATLAB. Do detekcji i rozpoznania została wykorzystana funkcja SURF.

2. SURF (Speeded-Up Robust Features)

Duża grupa komputerowych systemów wizyjnych opiera się na znalezieniu punktów charakterystycznych pomiędzy dwoma obrazami. Analizę obrazów w oparciu o punkty charakterystyczne można podzielić na trzy główne etapy:

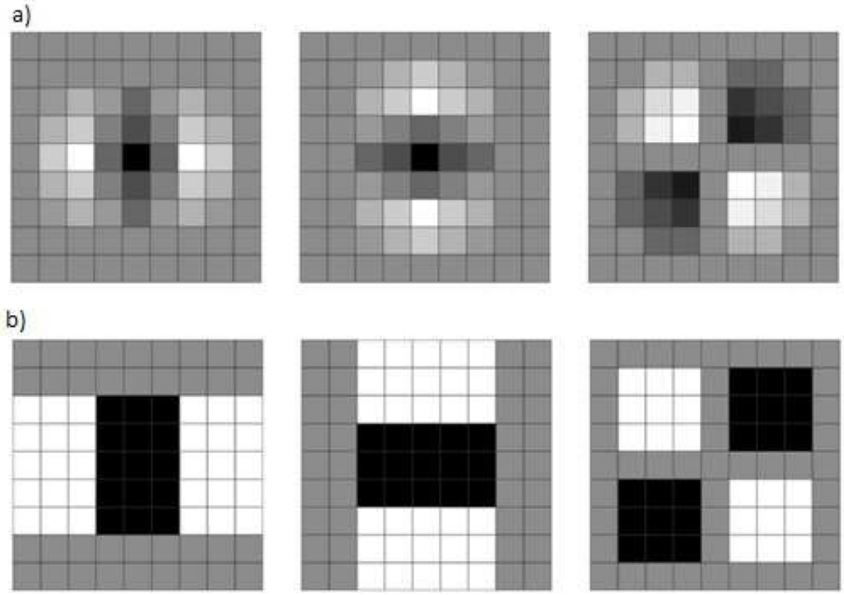
- detekcja punktów charakterystycznych,
- wydobycie cech (deskryptorów) tych punktów,
- znalezienie odpowiadających sobie punktów charakterystycznych na obrazach.

Porównywanie badanych obrazów z obrazami wzorcowymi (wzorcami znaków) wymaga algorytmów przetwarzania punktów charakterystycznych cechujących się niezmiennością od rotacji, przesunięcia oraz skali. Deskryptor SURF zaprezentowany przez Herberta Bay'a, Tinne Tuytelaars'a i Luca Van Gool [4] opisuje rozkład odpowiedzi falki Haar'a (Haar Wavelet) w sąsiedztwie punktów charakterystycznych. Wykorzystywane są tylko 64 wymiary, pozwalając na redukcję czasu obliczeń cech. Deskryptor SURF jest inwariantny względem obrotu i zmian skali, jednocześnie zachowując wysoką powtarzalność detekcji punktów obrazu oraz odporność na zakłócenia [4].

2.1. Detekcja punktów charakterystycznych

Detektor SURF bazuje na strukturach przypominających krople na obrazie. Punkty charakterystyczne znajdują się w miejscach takich jak narożniki, krawędzie czy skrzyżowania

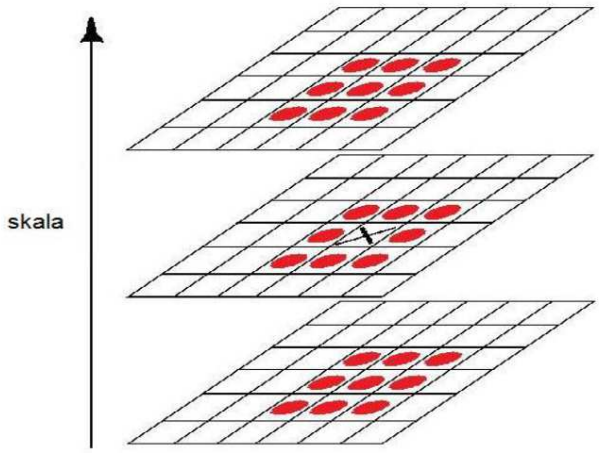
typu „T”. W celu znalezienia punktów charakterystycznych muszą zostać nałożone filtry na obraz źródłowy, których przykłady dla rozmiaru 9x9 przedstawiono na Rys. 1.



Rys. 1. Gausowska pochodna drugiego rzędu z a) x- (L_{xx}), y- (L_{yy}) oraz xy- (L_{xy}), b) x- (D_{xx}), y- (D_{yy}) oraz xy- (D_{xy}) [3]

Punkty charakterystyczne muszą być wyszukiwane w różnych skalach, ponieważ wyszukiwane są wspólne cechy na obrazach, które mogą różnić się skalą.

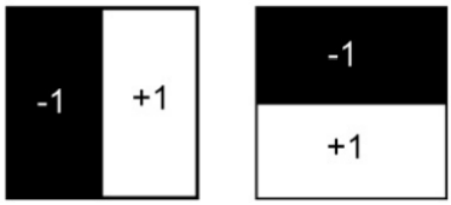
W kolejnym etapie należy znaleźć kandydatów na punkty charakterystyczne poprzez wyznaczenie ekstremów w różnicach rozmyć gaussowskich. W tym celu każdy piksel dla każdej warstwy i skali jest porównywany ze swoimi 26-cioma sąsiadami. Osiem punktów znajduje się w skali rodzimej, a po dziewięć występuje na skalach poniżej i powyżej (Rys. 2). W przypadku, kiedy rozpatrywany piksel ma wartość większą od sąsiednich pikseli lub mniejszą od pozostałych, zostaje „kandydatem” na punkt charakterystyczny [4].



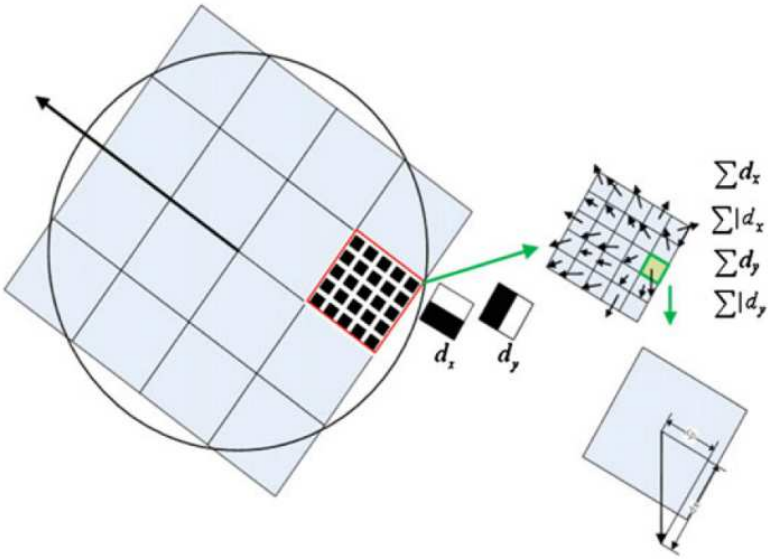
Rys. 2. Wykrywanie ekstremów funkcji poprzez porównanie piksela [4]

2.2. Deskryptor

Punktom charakterystycznym przypisuje się orientację poprzez połączenie pikseli (iloczyn pikseli obszaru i falki) w ich sąsiedztwie z poziomymi i pionowymi filtrami falkowymi Haar'a (Rys. 3). Następnie dookoła tego punktu budowany jest kwadratowy obszar. Wszystkie regiony kwadratowe mają wielkość 20-krotności skali, na której wykryto punkt charakterystyczny. Następnie region ten zostaje podzielony na mniejsze regiony o wymiarach 4x4. Dla każdego z podregionów obliczamy pionową i poziomą falkę Haar'a na regularnie rozłożonych wierzchołkach siatki 5x5 (Rys. 4). Następnie sumujemy wartości dx i dy w każdym podregionie i tworzymy pierwszy wektor cech. Wyodrębniamy również sumy wartości bezwzględnych |dx| i |dy|, które razem z wektorem tworzą czterowymiarowy deskryptor. Po połączeniu wszystkich podregionów 4x4 powstaje wektor deskryptora o długości 64. Wynikowy deskryptor SURF jest niezmienny względem rotacji, skali oraz jasności [3].



Rys. 3. Poziome i pionowe filtry falkowe Haar'a [3]



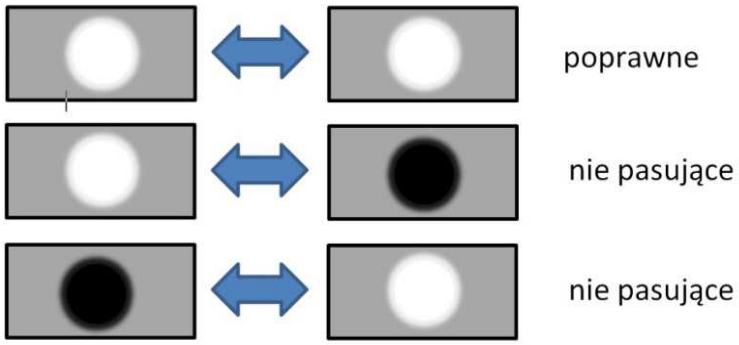
Rys. 4. Podział regionu zainteresowania na podregiony 4x4 w celu obliczenia deskryptora SURF [3].

W kolejnym kroku należy do każdego punktu przypisać orientację na podstawie lokalnych cech jego otoczenia tak, aby uzyskać niezmienniczość względem obrotu obrazu [4].

Wynikiem działania algorytmu SURF jest deskryptor o długości 64, lokalizacja (x,y), orientacja, wartość wyznacznika Hessian oraz znak Laplace'a dla każdego punktu charakterystycznego.

2.3 Dopasowanie punktów charakterystycznych

W celu szybkiego indeksowania podczas etapu dopasowywania dołączany jest znak Laplacian dla punktu charakterystycznego. Na etapie łączenia punktów porównujemy tylko te punkty charakterystyczne, które mają ten sam typ kontrastu. Operacja ta umożliwia odróżnienie jasnych obszarów od ciemnego tła, co przedstawiono na Rys. 5. Informacje te pozwalają na szybsze dopasowanie, bez zmniejszenia wydajności descryptora.



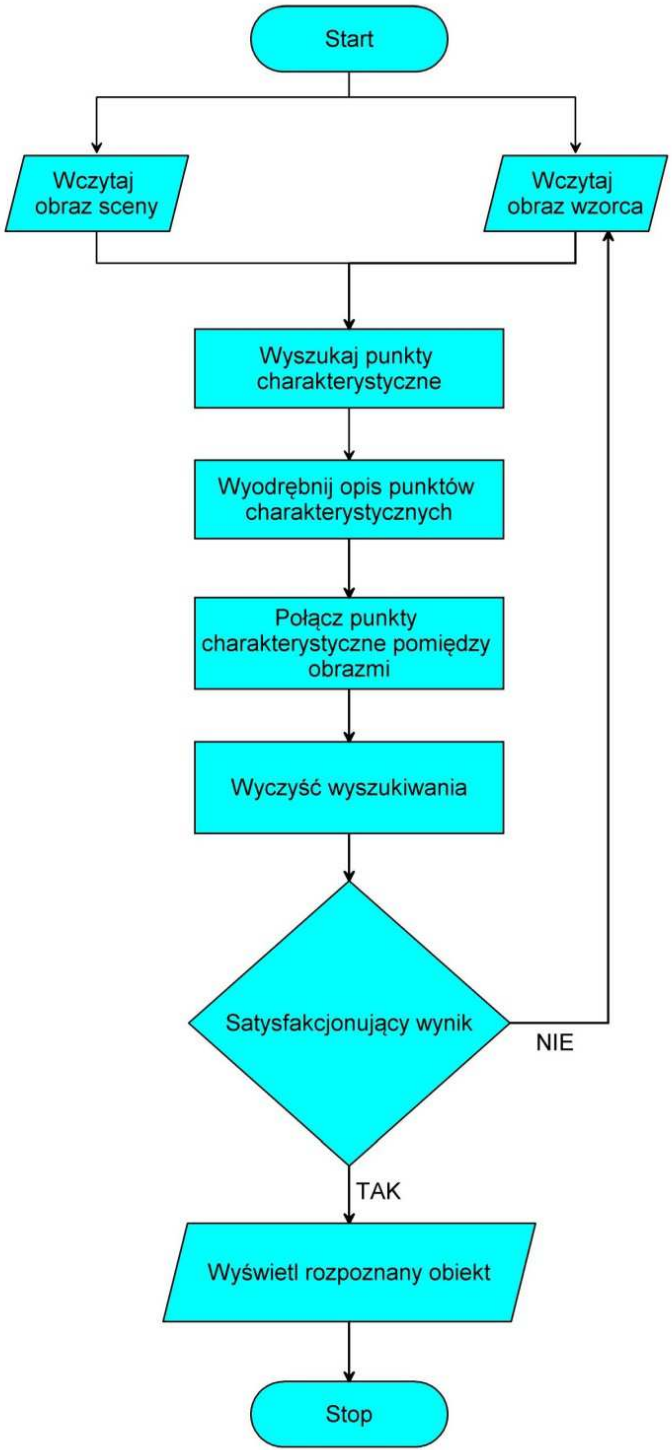
Rys. 5. Wpływ znaku Laplacian na możliwość łączenia cech obrazu [3]

3. System detekcji i rozpoznawania znaków drogowych

Schemat blokowy algorytm rozpoznawania znaków drogowych zaimplementowanego w środowisku Matlab przedstawiono na Rys.6.

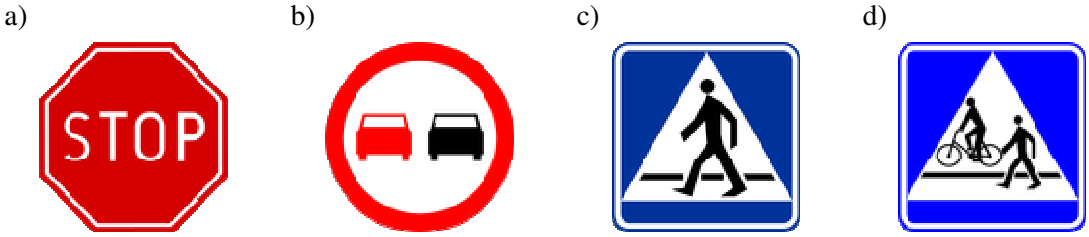
Główne etapy działania programu możemy podzielić na:

- wczytanie obrazu (zdjęcia cyfrowego) sceny oraz inicjalizację wyszukiwania,
- detekcję punktów charakterystycznych,
- wydobycie cech (deskryptorów) tych punktów,
- znalezienie odpowiadających sobie punktów charakterystycznych,
- czyszczenie dopasowań,
- wyświetlenie wyników identyfikacji.



Rys. 6. Schemat blokowy proponowanego rozwiązania

W pierwszym etapie zostały przygotowane wzorce znaków drogowych o rozdzielczości 72x72 dpi. Na potrzeby testu wybrano znaki (Rys. 7) stop (B-20), zakaz wyprzedzania (B-25), przejście dla pieszych (D-6), przejście dla pieszych i przejazd dla rowerzystów (D-6b).



Rys.7. Wzorce: a) B-20, b) B-25, c) D-6, d) D-6b [6]

Przygotowano 50 zdjęć testowych zawierających znaki drogowe. Zdjęcia zostały wykonane za pomocą aparatu, umieszczonego na kokpicie samochodu, tak aby warunki były jak najbardziej zbliżone do rzeczywistych. W celu sprawdzenia wpływu czynników zewnętrznych, takich jak oświetlenie czy warunki pogodowe zdjęcia zostały wykonane w świetle dziennym, jak również po zmroku. Przykładowe zdjęcia przedstawia Rys. 8.

Rys. 8. Zdjęcia testowe

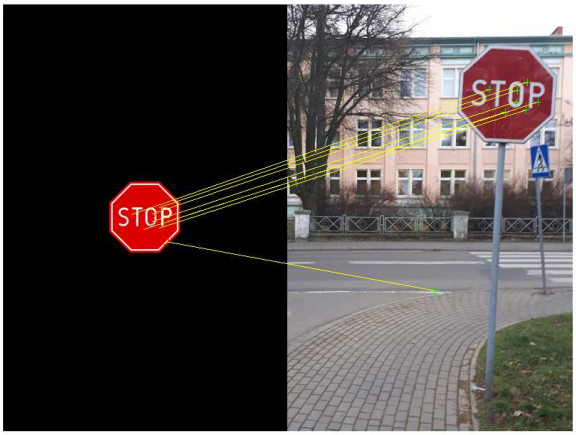
Program w pierwszej kolejności wyszukuje punkty charakterystyczne obrazy (Rys. 9), a następnie po kolei punkty dla każdego z wzorców (Rys. 10).

Kolejnym etapem jest wyodrębnienie wektorów cech: lokalizacji (x,y), orientacji, wartości wyznacznika Hessian oraz znaku Laplace'a dla każdego punktu charakterystycznego.

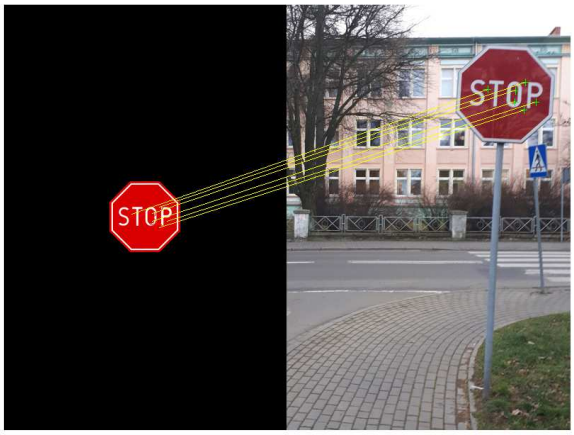
Rys. 9. Punkty charakterystyczne sceny

Rys. 10. Punkty charakterystyczne wzorca

Następnie łączone są pasujące punkty charakterystyczne pomiędzy wzorcem a sceną. Efekt takiego połączenia przedstawia Rys.11, który przedstawia zarówno poprawnie dopasowane punkty jak i jedno błędne dopasowanie.



Rys. 11. Dopasowanie punktów pomiędzy wzorcem a sceną



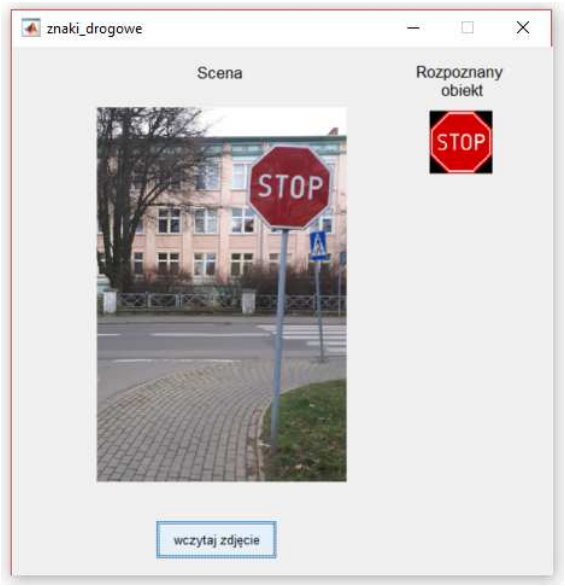
Rys. 12. Poprawne dopasowania po usunięciu błędnych trafień

W celu wyeliminowania błędnych dopasowań został zastosowany algorytm oparty na iteracyjnej metodzie szacowania parametrów modelu matematycznego na podstawie zbioru obserwowanych danych zawierających wartości znacząco różniące się od pozostałych.

W każdej iteracji algorytm zlicza liczbę próbek zgodnych z modelem i wprowadza wektor wag, który odpowiada modelowi o najwyższym rankingu. Podstawowym założeniem jest eliminacja danych nie pasujących do modelu.

Efekt działania programu jest widoczny na Rys. 12, gdzie znajdują się jedno połącznie mniej w stosunku do połączeń przedstawionych na Rys. 11.

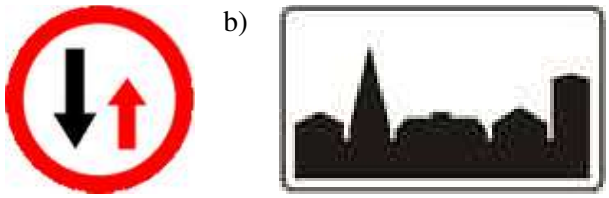
W efekcie działania programu w oknie zostaje wyświetlony rozpoznany znak obok wczytanego zdjęcia na którym poszukiwany jest znak (Rys. 13). W przypadku znalezienia na zdjęciu większej ilości znaków, zostają one wyświetlone jedno pod drugim.



Rys. 13. Interfejs programu z wyświetlonym rozpoznaniem

W kolejnym teście sprawdzimy działanie programu, dla dwóch znaków. Pierwszy znak B-31 („pierwszeństwo dla nadjeżdżających z przeciwka"), wybrany ze względu na podobieństwo do wcześniej testowanych znaków. Drugi znak to D-42 („obszar zabudowany"), o zupełnie innym kształcie i wyglądzie. Nowe wzorce znaków przedstawia Rys. 14. Wzorce te zostały dodane do istniejącej bazy zawierającej wcześniej testowane znaki.

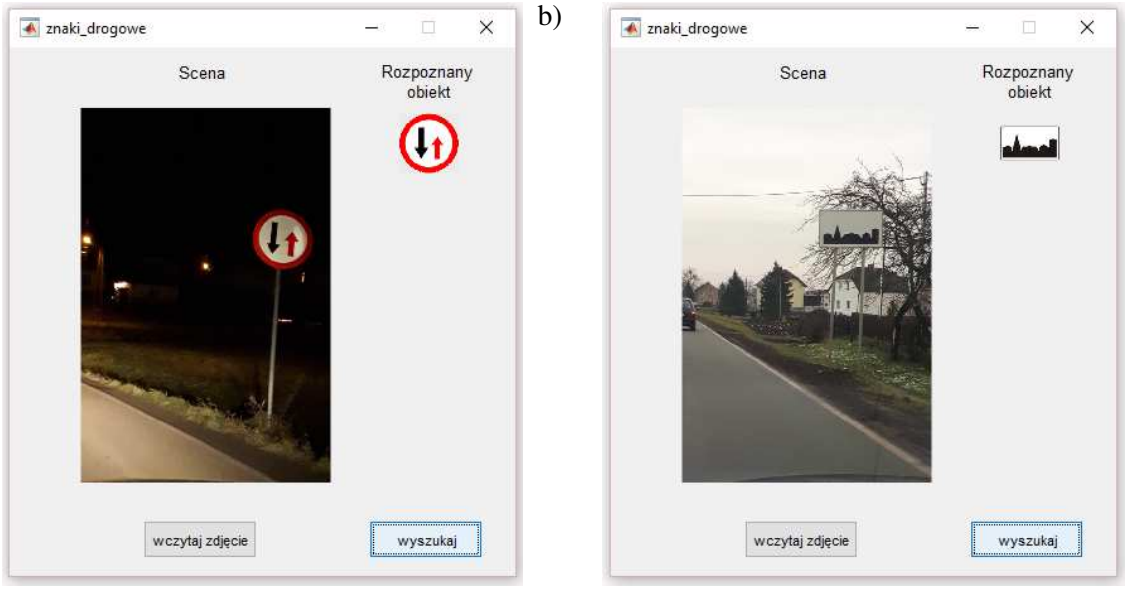
a) b)



Rys. 14. Wzorce znaków a) B-31 b) D-42 [6]

Po wykonaniu testów możemy stwierdzić, że program prawidłowo rozpoznał obydwa testowane znaki. Rezultaty testu dla znaku B-31 i znaku D-42 przedstawiono na Rys. 15.

a) b)



Rys. 15. Wynik wyszukiwania dla znaku a) B31; b) D42

Wyniki testów zostały przedstawione w Tab. 1**.**

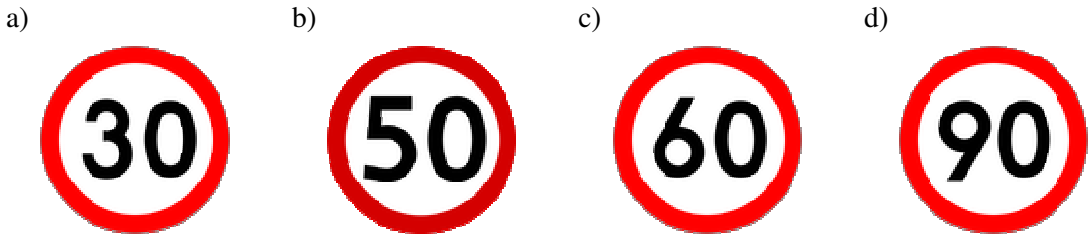
Tab. 1. Wyniki testów

Symbol znaku	Nazwa znaku	Liczba zdjęć	Poprawne rozpoznanie	Błędne rozpoznanie	Brak rozpoznania	Sprawność rozpoznania
B-20	stop	20	18	0	2	90%
B-25	zakaz wyprzedzania	5	5	0	0	100%
D-6	przejście dla pieszych	20	17	1	2	85%
D-6a	przejście dla pieszych i przejazd dla rowerzystów	5	4	0	1	80%
B-20 + D-6	stop + przejście dla pieszych	3	3	0	0	100%
B-31	pierwszeństwo dla nadjeżdżających z przeciwka	3	3	0	0	100%
D-42	obszar zabudowany	6	5	0	1	84%

Źródło: Opracowanie własne

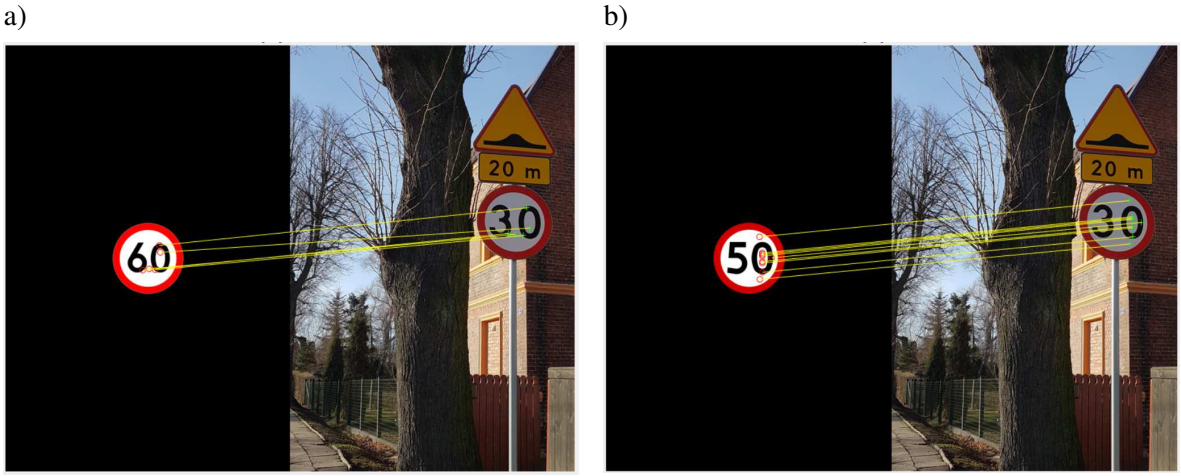
4. Rozszerzenie działania aplikacji

Znaki drogowe obejmują w swoim zakresie wiele grup, a w każdej z nich charakteryzują się sporą różnorodnością. Prezentowana aplikacja dedykowana jest do rozpoznawania określonych znaków (B-20, B-25, D-6, D-6b, B-31 oraz D-42), zakładając możliwość rozszerzenia aplikacji na kolejne znaki drogowe. Przeprowadzone zostały testy aplikacji dla znaków ograniczenia prędkości. Do testów przyjęto znak B-33 z ograniczeniem prędkości do np. 30, 50, 60, 90 km/h (Rys. 16).



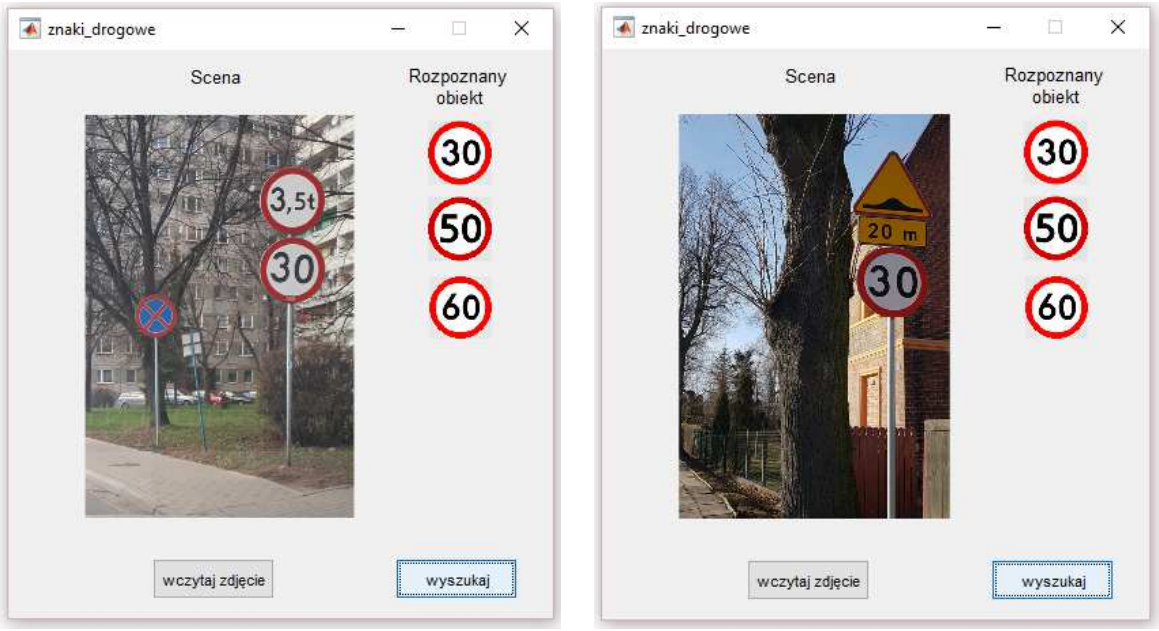
Rys. 16. Wzorce znaków B-33 [6]

Program prawidłowo wyszukuje znaki z ograniczeniem prędkości. Niestety, ze względu na duże podobieństwo, a szczególnie występujące na każdym znaku „0" występuje problem z prawidłowym rozpoznaniem wartości ograniczenia prędkości znajdującego się na znaku (Rys. 17). I tak w przypadku znaku z ograniczeniem do 30 km/h, w wynikach możemy zaobserwować przypisanie również znaków z ograniczeniem do 50 i 60 km/h (Rys. 18).



Rys. 17. Błędnie przypisany wzorzec znaku ograniczenia prędkości do a) 60 km/h b) 50 km/h

a) b)



Rys. 18. Testy znaków z ograniczeniem prędkości

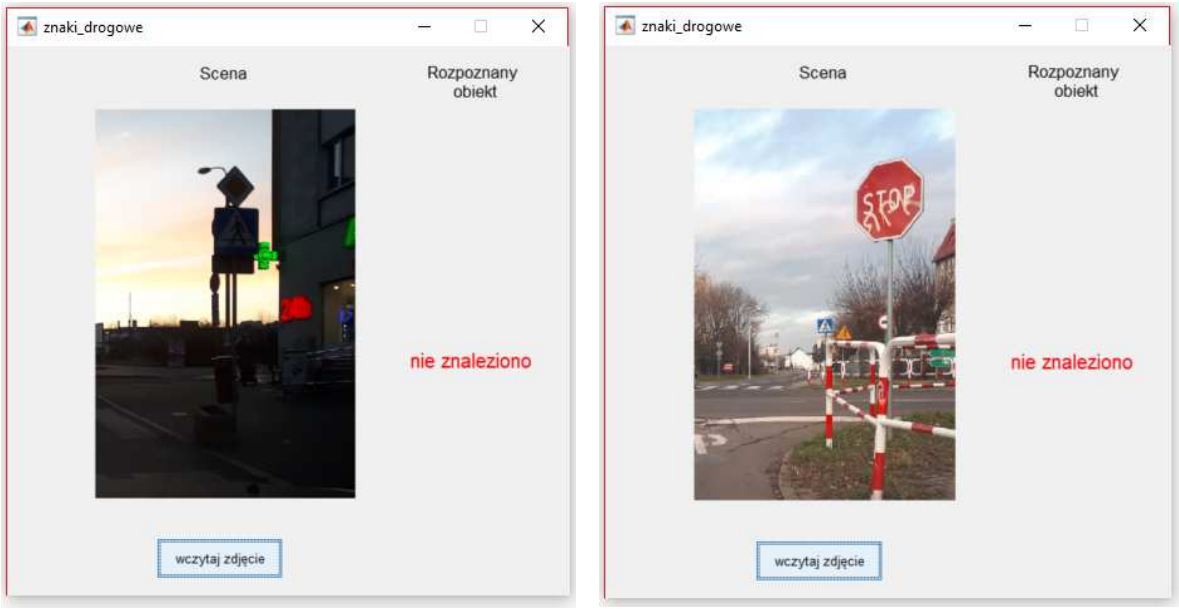
5. Wnioski

Opracowany system rozpoznawania znaków drogowych realizuje wszystkie stawiane przed nim oczekiwania, a mianowicie rozpoznanie znaków drogowych: B-20, B-25, D-6, D-6b, B-31 oraz D-42. Znakomicie sprawdza się dla znaków, które zostały umieszczone w katalogu wzorców i przetestowane, ze sprawnością co najmniej 80%

Analizując zdjęcia, na których znaki drogowe nie zostały rozpoznane, można określić parametry wpływają na skuteczność działania programu. Głównym czynnikiem jest odpowiednie oświetlenie znaku podczas wykonywania zdjęcia. Przykładem może tu być światło słoneczne świecące z naprzeciwka (Rys. 19a), co znacząco wpływa na kontrast na znakach. Kolejnym problemem są znaki zniszczone lub zawierające dodatkowe elementy. Powoduje to niezgodność z elementem wzorcowym. Przykład takiego znaku przedstawia Rys. 19b, gdzie znak został popisany białą farbą.

W przypadku rozbudowy aplikacji należy wziąć pod uwagę to, że różne rodzaje znaków drogowych mogą wymagać innego podejścia. Zostało to zademonstrowane na przykładzie znaków z ograniczeniem prędkości. Znaki te ze względu na swoje podobieństwo wymagają innej koncepcji. Prawdopodobnie zastosowane wzorce nie są wystarczająco dokładne. Innym rozwiązaniem mógłby być dodatkowy moduł, który po wyszukaniu znaku mógłby np. za pomocą systemu do rozpoznawania tekstu identyfikować liczbę zapisaną na znaku.

a) b)



Rys. 19. Błędne rozpoznanie a) zdjęcie wykonane pod światło słoneczne b) ze względu na uszkodzenie znaku

Docelowo aplikacja może zostać rozbudowana i nieznacznie zmodyfikowana aby móc ją używać na smartfonie w celu bieżącego informowania kierowcy o mijanych znakach drogowych nie tylko w postaci wyświetlanych znaków ale również poprzez komunikat głosowy. Aktualnie identyfikacja znaku drogowego realizowana jest w czasie poniżej jednej sekundy.

Bibliografia:

1. Bay H., Ess A., Tuytelaars T., Van Gool L., *Speeded-Up Robust Features (SURF)*, Computer Vision and Image Understanding, vol.110, 2008.
2. Evans Ch., *Notes on the OpenSURF Library*, University of Bristol, 2009.
3. http://kodeks-drogowy.org z dnia 1.04.2019
4. Malina W., Smiatacz M., *Cyfrowe przetwarzanie obrazów*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2008.
5. Sayago Benito M., *Speeded up robust features (SURF): performance test*, University of Madrid, 2013
6. Tadeusiewicz R., Korohoda P., *Komputerowa analiza i przetwarzanie obrazów*, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1997.

Dane kontaktowe
malgorzata.kuchta@pwsz.raciborz.edu.pl