POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA DYPLOMOWA MAGISTERSKA
na kierunku INFORMATYKA
specjalność: Inżynieria Komputerowa

Sylwester MADEJ

Nr albumu: 196200

Rok akad.: 2009/2010
Warszawa, 11.04.2009

PRZETWARZANIE I ROZPOZNAWANIE OBRAZÓW W SYSTEMACH WBUDOWANYCH

Zakres pracy:

1. *Wprowadzenie i sformułowanie celu pracy*
2. *Analiza zagadnienia rozpoznawania numerów tablic rejestracyjnych*
3. *Projekt układu analizy obrazu, w oparciu o mikrokontroler i kamerę cyfrową*
4. *Implementacja algorytmów akwizycji obrazu oraz rozpoznawania numerów tablic rejestracyjnych na systemie wbudowanym*
5. *Budowa systemu zarządzania urządzeniem z poziomu przeglądarki WWW*
6. *Podsumowanie i wnioski*

Kierujący pracą: dr inż. Witold Czajewski

Konsultant:

Termin złożenia pracy: 15.09.2010
Praca wykonana i obroniona pozostaje własnością Instytutu i nie będzie zwrócona wykonawcy.

Politechnika Warszawska
Wydział Elektryczny

Warszawa, 7 lutego 2012r.

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. PRZETWARZANIE I ROZPOZNAWANIE OBRAZÓW W SYSTEMACH WBUDOWANYCH:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Sylwester Madej Sylwester Madej

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA DYPLOMOWA MAGISTERSKA
na kierunku INFORMATYKA
specjalność: Inżynieria Komputerowa

Sylwester MADEJ
Nr albumu: 196200

Rok akad.: 2009/2010
Warszawa, 11.04.2009

PRZETWARZANIE I ROZPOZNAWANIE OBRAZÓW W SYSTEMACH WBUDOWANYCH

Zakres pracy:

1. Wprowadzenie i sformułowanie celu pracy
2. Analiza zagadnienia rozpoznawania numerów tablic rejestracyjnych
3. Projekt układu analizy obrazu, w oparciu o mikrokontroler i kamerę cyfrową
4. Implementacja algorytmów akwizycji obrazu oraz rozpoznawania numerów tablic rejestracyjnych na systemie wbudowanym
5. Budowa systemu zarządzania urządzeniem z poziomu przeglądarki WWW
6. Podsumowanie i wnioski

Kierujący pracą: dr inż. Witold Czajewski

Konsultant:

Termin złożenia pracy: 15.09.2010

PRZETWARZANIE I ROZPOZNAWANIE OBRAZÓW W SYSTEMACH WBUDOWANYCH

Streszczenie

Celem niniejszej pracy było opracowanie wydajnego i skutecznego algorytmu rozpoznawania numerów tablic rejestracyjnych, przeznaczonego do implementacji na systemach wbudowanych. Pod uwagę wzięto głównie rozpoznawania statyczne, ze szczególnym uwzględnieniem systemów kontroli dostępu do parkingów i posesji.

Pracę podzielono na dwie zasadnicze części: teoretyczną i praktyczną.

W części teoretycznej przedstawiono istniejące algorytmy opisane w literaturze przedmiotu oraz komercyjne systemy wykorzystujące rozpoznawanie tablic rejestracyjnych. Dodatkowo wprowadzono podstawowe pojęcia z dziedziny przetwarzania i rozpoznawania obrazów oraz uczenia maszynowego, w zakresie niezbędnym do zrozumienia zagadnień rozpatrywanych w części praktycznej.

Celem części praktycznej, było przedstawienie wyników badań nad opracowaniem wspomnianego wcześniej algorytmu oraz opis systemu Parking-LPR, stanowiącego praktyczne zastosowanie opracowanego algorytmu na urządzeniu wbudowanym.

W toku przeprowadzonych prac powstały trzy wersje algorytmu statycznego rozpoznawania numerów tablic rejestracyjnych, różniące się szybkością przetwarzania obrazów oraz skutecznością rozpoznawania. W ramach badań przeprowadzono testy i wybrano algorytm o najlepszym stosunku wydajności do szybkości, który nadawał się do stworzenia prototypu systemu kontroli dostępu do parkingu.

Wybrany algorytm posłużył jako baza do stworzenia systemu parkingowego opartego o komputer jednopłytkowy Beagleboard, kamerę oraz ultradźwiękowy czujnik odległości. Opis przygotowania sprzętu oraz szczegóły implementacji oprogramowania dla systemu Parking-LPR, stanowią ostatni fragment części praktycznej pracy.

Wyniki badań potwierdzają, że opracowany algorytm nadaje się do zastosowań w systemach parkingowych oraz wskazują na wydajność i szybkość porównywalną z niektórymi systemami komercyjnymi.

IMAGE PROCESSING AND RECOGNITION ON EMBEDDED SYSTEMS

Abstract

The objective of this thesis was to develop an efficient and effective ANPR (Automatic Number Plate Recognition) algorithm, intended for implementation on embedded systems. Mainly static recognition systems were considered, with particular emphasis on access control systems for properties and parking lots.

The work is divided into two major parts: theoretical and practical.

The theoretical part presents the existing algorithms described in the literature and commercial systems that use number plate recognition. In addition, an explanation of basic concepts in the field of image processing, pattern recognition and machine learning, is provided, at the level necessary to understand the issues under consideration in the practical part.

The practical part shows the results of research focused on the development of the algorithm and a description of Parking-LPR system, which is the practical application of the developed algorithm, on an embedded device.

In the course of the work carried out, three versions of the algorithm were developed. Each of them with different image processing speed and efficiency of recognition. Algorithm, with the best performance to efficiency ratio, was selected, as a result of performed tests.

The selected algorithm was used as a base to create a prototype parking system, running on a Beagleboard single board computer, with PSEye camera and a custom build ultrasonic distance sensor connected via USB. The final piece of practical part, describes hardware used by Parking-LPR system and implementation details of created software.

The results indicated that created algorithm is suitable for usage on parking lots systems and is as effective and fast as some commercial algorithms.

Politechnika Warszawska
Wydział Elektryczny

Warszawa, 15.09.2010r.

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. PRZETWARZANIE I ROZPOZNAWANIE OBRAZÓW W SYSTEMACH WBUDOWANYCH:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Sylwester MADEJ.....

Spis treści

1. Wstęp

Postęp cywilizacyjny i zmiany gospodarcze powodują gwałtowny wzrost liczby zarejestrowanych samochodów. Wg. raportu opublikowanego przez Polski Związek Przemysłu Motoryzacyjnego [23], w 2008 roku liczba, zarejestrowanych w Polsce, pojazdów samochodowych wyniosła ponad 21 mln. Jednocześnie transport drogowy wykonuje ponad 60 procent całej pracy przewozowej. Wyniki te oznaczają, że konieczne jest automatyzowanie jak największej liczby czynności związanych z ruchem drogowym, celem obsłużenia jego rosnącego natężenia. Podstawowe dziedziny, które mogą podlegać automatyzacji to:

- wykrywanie wykroczeń drogowych,
- analiza ruchu drogowego,
- naliczanie opłat i kontrola dostępu.

We wszystkich tych zadaniach konieczna jest identyfikacja pojazdu. Najprostszym sposobem na określenie położenia pojazdu i jego identyfikacji jest umieszczenie w aucie specjalnego urządzenia wykorzystującego np. technologię RFID (*ang. Radio Frequency IDentification*) lub GPS (*ang. Global Positioning System*).

Technologia RFID stosowana jest np. w elektronicznym systemie pobierania opłat viaTOLL [38], obowiązującym w Polsce od 1 lipca 2011 r. Każdy kierowca chcący korzystać z systemu, musi zainstalować w pojeździe urządzenie viaBOX (rysunek 1.1), które komunikuje się z bramownicami wyposażonymi w anteny. „Za każdym razem, gdy pojazd (wyposażony w viaBOX) przejeżdża pod bramownicą, zostaje naliczona opłata za przejazd konkretnym odcinkiem drogi płatnej. Kierowca zostaje o tym powiadomiony pojedynczym sygnałem z viaBOX-a. Proces naliczenia opłaty przebiega w pełni automatycznie bez potrzeby redukowania prędkości pojazdu lub zatrzymywania się." [39]

Wadą powyższego rozwiązania jest konieczność wyposażania każdego pojazdu w specjalizowane urządzenia, co w większości przypadków jest nieopłacalne. Dobrą alternatywą są systemy identyfikujące pojazdy na podstawie tablic rejestracyjnych, które są unikalne dla każdego pojazdu, a ponadto przystosowane do automatycznej detekcji (specjalne czcionki, ograniczenia występowania znaków, materiały poprawiające kontrast). Systemy tego typu znajdują coraz szersze zastosowanie, począwszy od systemów kontroli dostępu do parkingu, poprzez naliczanie opłat za wjazd do określonej strefy, aż po kontrolę prędkości przejeżdżających pojazdów.

Oba typy systemów mają swoje zastosowania. Systemy wykorzystujące urządzenia lokalizacyjne umieszczone w pojazdach, są drogie i kłopotliwe w eksploatacji, ale dają bardzo dobrą skuteczność identyfikacji i lokalizacji po-

Rysunek 1.1: Urządzenie viaBOX montowane w samochodach, na użytek systemu viaTOLL. Źródło: [40]

jazdów. Z drugiej strony systemy wizyjne dzięki zastosowaniu coraz bardziej skutecznych algorytmów, minimalizacji kosztów kamer i komputerów oraz wzrostowi ich wydajności, stanowią coraz lepszą alternatywę, zwłaszcza w przypadku, gdy priorytetem jest łatwość eksploatacji systemu.

1.1. Cel i układ pracy

Celem niniejszej pracy jest przedstawienie istniejących rozwiązań w dziedzinie automatycznego rozpoznawania tablic rejestracyjnych na obrazach statycznych, ze szczególnym naciskiem na ich wydajność, oraz opracowanie i implementacja algorytmu, z myślą o specyfice urządzeń wbudowanych. Urządzenia te, ze względu na ograniczenia wydajnościowe oraz architektoniczne, wymagają stosowania algorytmów o niewielkiej złożoności obliczeniowej, operujących najlepiej na liczbach stałoprzecinkowych. Opracowany algorytm został następnie zastosowany w prostym systemie kontroli dostępu do parkingu, z możliwością monitoringu pojazdów poprzez przeglądarkę WWW.

Praca składa się z sześciu rozdziałów.

Pierwszy rozdział zawiera wstęp do tematu identyfikacji pojazdów oraz cel pracy.

Rozdział drugi poświęcony został wprowadzeniu do zagadnienia rozpoznawania tablic rejestracyjnych, z uwzględnieniem istniejących rozwiązań oraz podstaw prawnych dotyczących tablic rejestracyjnych.

Rozdział trzeci zawiera przegląd literatury oraz prezentację najpopularniejszych wizyjnych algorytmów lokalizacji i segmentacji tablicy rejestracyjnej oraz optycznego rozpoznawania wydzielonych znaków. Dokonano w nim podziału procesu rozpoznawania na poszczególne fazy oraz zdefiniowano pojęcia cechy i klasyfikatora, wykorzystywane przy opisie algorytmów rozpoznawania znaków.

Rozdział czwarty stanowi opis przeprowadzonych badań nad opracowaniem algorytmu rozpoznawania tablic rejestracyjnych na obrazie, działają-

cego na urządzeniach wbudowanych. Zawiera opis trzech algorytmów, które zostały opracowane w toku badań.

Rozdział piąty rozwija zagadnienie opisane w rozdziale czwartym, o testy porównawcze powyższych algorytmów przeprowadzone na trzech platformach wbudowanych, które dodatkowo zostały pokrótce opisane. Rozdział kończy zestawienie wyników porównania oraz wybór algorytmu, do zastosowania w dalszych pracach nad opracowaniem systemu parkingowego.

Rozdział szósty poświęcony został opisowi przykładowej implementacji systemu parkingowego, na platformie Beagleboard, wykorzystującej opracowany algorytm. Opisano w nim zarówno część sprzętową oraz programową rozwiązania, włącznie z panelem administracyjnym dostępnym z poziomu przeglądarki internetowej.

Rozdział siódmy zawiera podsumowanie i wnioski wyciągnięte z pracy.

2. Rozpoznawanie tablic rejestracyjnych

2.1. Wstęp

Systemy automatycznego rozpoznawania tablic rejestracyjnych występują pod wieloma nazwami [36]:

- ANPR - (ang. Automatic Number Plate Recognition),
- ALPR/LPR - (ang. Automatic License Plate Recognition),
- CPR – (ang. Car Plate Recognition),
- AOTR – Automatyczny Odczyt Tablic Rejestracyjnych,
- ARTR – Automatyczne Rozpoznawanie Tablic Rejestracyjnych.

W każdym przypadku mamy do czynienia z systemami tego samego typu. Ich działanie polega na wykrywaniu tablic rejestracyjnych na obrazie, a następnie wyodrębnieniu znaków i ich rozpoznaniu. Umożliwia to prostą identyfikację pojazdu, bez konieczności ich dodatkowego znakowania i instalowania specjalnych urządzeń.

Każdy system ANPR wykonuje szereg następujących czynności:

1. Akwizycja obrazu z kamery.
2. Normalizacja obrazu.
3. Lokalizacja tablicy rejestracyjnej na obrazie.
4. Segmentacja (wydzielenie obszarów zawierających znaki).
5. Rozpoznanie znaków z wykorzystaniem algorytmu OCR (*ang. Optical Character Recognition*).
6. Analiza syntaktyczna (walidacja poprawności) z ew. korektą wyniku.
7. Zapis wyników i wykonanie dodatkowych czynności, w zależności od wyniku rozpoznania.

Uzyskane w ten sposób dane znajdują wiele zastosowań m.in. w:

- automatycznym otwieraniu bram/szlabanów dla uprawnionych pojazdów (parkingi, posesje),
- naliczaniu i kontroli opłat, czasu przebywania i postoju,
- kontroli prędkości (fotoradary) i rejestracji pojazdów,
- gromadzeniu informacji o trasie i średniej prędkości pojazdu, na wybranym odcinku drogi,
- identyfikacji skradzionych pojazdów,
- identyfikacji i lokalizacji (województwo, gmina) w celu uzyskania danych statystycznych potrzebnych do przeprowadzania analiz ruchu drogowego.

Systemy identyfikacji tablic rejestracyjnych podzielić można na dwie podstawowe kategorie:

- dla pojazdów nieruchomych (rozpoznawanie statyczne),
- dla pojazdów ruchomych (rozpoznawanie dynamiczne).

2.1.1. Rozpoznawanie statyczne

Pojazd podlegający rozpoznaniu, w tego typu systemach, nie może się poruszać, a ponadto powinien znajdować się w określonym położeniu względem kamery. Często wymuszane jest to poprzez instalowanie szlabanów oraz bramek wjazdowych. Podstawową wadą tego typu systemów jest możliwość rozpoznawania jednocześnie tylko jednego pojazdu, jednak dzięki temu dokładność rozpoznawania jest wyższa niż w systemach wykorzystujących rozpoznawanie dynamiczne. Pozwala to stosować systemy rozpoznawania statycznego np. na parkingach, gdzie jakość rozpoznawania jest szczególnie ważna.

2.1.2. Rozpoznawanie dynamiczne

Systemy oparte na rozpoznawaniu dynamicznym pozwalają zidentyfikować tablice wielu poruszających się pojazdów, na podstawie obrazu z jednej kamery. Pozwala to, na znaczne zwiększenie przepustowości systemu, kosztem skuteczności rozpoznawania i zwiększonych wymagań, co do mocy obliczeniowej systemu. Coraz częściej systemy takie używane są w Polsce m.in. w fotoradarach.

2.2. Podstawy prawne dotyczące tablic rejestracyjnych

Opracowanie optymalnego i poprawnie działającego algorytmu, dla systemu rozpoznawania polskich tablic rejestracyjnych, wiąże się z koniecznością zapoznania z aktami prawnymi, które są odpowiedzialne za unormowanie wzorów tablic rejestracyjnych, wydawanych na terenie Rzeczypospolitej Polskiej.

2.2.1. Charakterystyka tablic rejestracyjnych na terenie RP

Podstawowym dokumentem, normującym wzory i wygląd tablic rejestracyjnych, jest Rozporządzenie Ministra Infrastruktury[26]. W chwili obecnej, pojazdy zarejestrowane na terenie Rzeczypospolitej Polskiej posiadają jeden z dwóch formatów tablic rejestracyjnych („stare” i „nowe”). Różnorodność ta wynika z faktu, iż 31 marca 2000 r wprowadzony został całkiem nowy format tablic. Stare tablice rejestracyjne, wydawane od roku 1976, o wymiarach 520 x 120 mm charakteryzowały się czarnym tłem oraz białymi znakami (rysunek 2.1a). Ze względu na trudności podczas rozpoznawania numerów rejestracyjnych w nocy, z dniem 1 maja 2000 roku, wprowadzony został nowy wzór tablic rejestracyjnych.

Nowe tablice rejestracyjne o wymiarach 520 x 114 mm (tablice jednorzędowe) oraz 305 x 214 mm (tablice dwurzędowe) posiadają odblaskowe tło barwy białej, żółtej lub niebieskiej, na którym znajdują się wytłoczone w linii

prostej czarne znaki. Dodatkowo w skrajnej lewej części tablicy znajdowała się, na niebieskim tle, flaga Polski (rysunek 2.1b), jednak od 1 maja 2006 r. została zastąpiona przez symbol Unii Europejskiej, składający się z 12 pięcioramiennych gwiazdek barwy żółtej (rysunek 2.1c). Pod wyżej wymienionymi symbolami umieszcza się znak z literami *PL*.

(a) Stara tablica rejestracyjna (1976-2000)

(b) Nowa tablica rejestracyjna z flagą Polski (V 2000 – V 2006)

(c) Nowa tablica rejestracyjna z symbolem Unii Europejskiej (od V 2006)

Rysunek 2.1: Rodzaje polskich tablic rejestracyjnych. Źródło: http://pl.wikipedia.org/wiki/Tablica_rejestracyjna

Wraz z upływem czasu, od wprowadzenia nowego wzoru tablic rejestracyjnych, liczba tablic o starym wzorze jest coraz mniejsza, dlatego też algorytm rozpoznawania tablic, będący przedmiotem niniejszej pracy, został zaprojektowany z myślą o nowych tablicach, pomijając stare.

2.2.2. Numer rejestracyjny

Zbiór możliwych znaków, z których tworzony jest numer rejestracyjny określa rozporządzenie [26]: „§ 21. 1. Numery rejestracyjne są tworzone ze zbioru następujących 25 liter: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R, S, T, U, V, W, X, Y i Z oraz cyfr od 0 do 9." W tablicach samochodowych zwyczajnych pierwsza litera stanowi wyróżnik województwa natomiast druga lub druga i trzecia stanowi wyróżnik powiatu. W zależności od liczby liter w wyróżniku powiatu rozróżnia się tablice: grodzkie z jednoliterowym wyróżnikiem oraz tablice ziemskie z dwuliterowym wyróżnikiem. Koleje cyfry, cyfry i litery lub litery budują wyróżnik pojazdu o podanych niżej wariantach:

1. Dla powiatów z wyróżnikiem jednoliterowym:
   - 5 cyfr (np. XY 12345)
   - 4 cyfry, 1 litera (np. XY 1234A)
   - 3 cyfry, 2 litery np. (np. XY 123AB)
   - 1 cyfra, 1 litera, 3 cyfry (np. XY 1A234)
   - 1 cyfra, 2 litery, 2 cyfry (np. XY 1AC23)
2. Dla powiatów z wyróżnikiem dwuliterowym:
   - 1 litera, 3 cyfry (np. XYZ A123)
   - 2 cyfry, 2 litery (np. XYZ 12AC)
   - 1 cyfra, 1 litera, 2 cyfry (np. XY 1A23)
   - 2 cyfry, 1 litera, 1 cyfra (np. XYZ 12A3)
   - 1 cyfra, 2 litery, 1 cyfra (np. XYZ 1AC2)
   - 2 litery, 2 cyfry (np. XYZ AC12)
   - 5 cyfr (np. XYZ 12345)
   - 4 cyfry, 1 litera (np. XYZ 1234A)
   - 3 cyfry, 2 litery (np. XYZ 123AB)

Budowa wyróżnika pojazdu obwarowana jest przez Rozporządzenie Ministra Infrastruktury[26] dodatkowymi zastrzeżeniami dotyczącymi znaków możliwych do wykorzystania przy jego tworzeniu: „§ 22. 1. Wyróżnik pojazdu, określa organ rejestrujący, posługując się zestawem cyfr i liter stanowiących pojemność rejestracyjna, z wyłączeniem liter B, D, I, O i Z." Brak możliwości stosowania liter wymienionych powyższym paragrafie wynika z ich podobieństwa do cyfr 8,0,1,0,2. Dodatkowym problemem, który nie został objęty normą jest podobieństwo symbolu litery „S" do cyfry „5".

Rozporządzenie Ministra Infrastruktury[26] gwarantuje, iż każdy samochód zarejestrowany w Polsce ma obowiązek posiadać tablice rejestracyjną o indywidualnym numerze, zamontowaną zarówno z przodu jak i z tyłu pojazdu: „§ 25. 1.47) Właściciel pojazdu umieszcza na pojeździe tablicę z przodu i z tyłu w miejscach konstrukcyjnie do tego przeznaczonych, z wyjątkiem przyczep, ciągników rolniczych, motocykli i motorowerów, na których tablice umieszcza się tylko z tyłu, z zastrzeżeniem ust. 1a—2."

Dodatkowym uwarunkowaniem znajdującym się w tym samym paragrafie jest punkt 3 o treści: „Z przodu pojazdu umieszcza się wyłącznie tablicę jednorzędową, z wyjątkiem pojazdu, o którym mowa w ust. 1a, oraz ciągnika rolniczego, o którym mowa w ust. 2". Dzięki temu, można ograniczyć działanie algorytmu do rozpoznawania jednorzędowych tablic, co znacznie ułatwi jego konstrukcję, eliminując tym samym jednak możliwość rozpoznawania pojazdów ciągnikowych, w których montowana jest wyłącznie dwurzędowa tylna tablica rejestracyjna. Założenia projektowe dotyczące wykorzystywania opracowanego algorytmu na parkingach pozwoliły na pominięcie pojazdów ciągnikowych, a co za tym idzie również tablic dwurzędowych.

2.2.3. Położenie tablic rejestracyjnych

Bardzo istotnym elementem w rozpoznawaniu tablic rejestracyjnych jest ich umiejscowienie, jednak w tej kwestii nie ma ściśle określonych reguł mocowania tablic rejestracyjnych. W Ustawie o Warunkach Technicznych Pojazdów [25] możemy jedynie znaleźć uwarunkowania dotyczące położenia tylnych tablic rejestracyjnych natomiast brak jest konkretnych regulacji odnośnie przednich tablic. Dodatkowo w Rozporządzeniu Ministra Infrastruktury [26] znaleźć można podpunkt o treści: „Jeżeli pojazd nie ma specjalnego miejsca do umocowania tablic, to przednią i tylną tablicę umieszcza się w środku szerokości pojazdu, a gdyby to utrudniało jego eksploatowanie lub ograniczało widoczność tablicy — po lewej stronie pojazdu." Niestety ustawodawstwo nie określa wysokości na jakiej ma być umieszczona przednia tablica rejestracyjna, co wymusza jej wyszukiwanie na całym obrysie auta. Paradoksalnie, położenie tylnej tablicy rejestracyjnej jest dokładnie określone przez Ustawę o Warunkach Technicznych Pojazdów „§ 1. 2, 4) Wysokość położenia tablicy nad ziemią: a) wysokość położenia dolnej krawędzi tablicy nad ziemią nie może być mniejsza niż 0,30 m, b) wysokość położenia górnej krawędzi tablicy nad ziemią nie może być większa niż 1,20 m, jeżeli jednak wymóg ten nie może być spełniony w praktyce, wysokość położenia może przekraczać 1,20 m, lecz powinna być tak zbliżona do tego wymagania, jak

to jest możliwe ze względu na konstrukcję pojazdu, i nie może w żadnym przypadku przekraczać 2 m;"[25].

Z powyższych powodów nie można ujednolicić zasad odnoszących się montowania przednich tablic rejestracyjnych.

2.3. Wybrane systemy stosowane komercyjnie

Pierwszy system ANPR został opracowany w 1976 r., przez oddział *Scientific Development Branch*, Policji w Wielkiej Brytanii i wykorzystany po raz pierwszy na autostradzie A1 oraz w tunelu Dartford. Od tamtego czasu powstało wiele komercyjnych systemów wykorzystujących zarówno rozpoznawanie statyczne, jak i dynamiczne. W dalszej części rozdziału opisano pokrótce, najpopularniejsze systemy stosowane przez Policję, służby drogowe oraz służby ochrony. Kryterium wyboru stanowiła dostępność informacji o systemie, oraz ilość instalacji go wykorzystujących.

2.3.1. System Congestion Charging

Najbardziej znanym przykładem wykorzystania systemów ANPR na dużą skalę jest, uruchomiony w 2003 r., system naliczania opłaty za wjazd do centrum Londynu (Congestion Charging)[9]. System ten zbudowany został w oparciu o 1500 kamer, umieszczonych bezpośrednio nad jezdnią (rysunek 2.2), które nieustannie wykonują zdjęcia pojazdów wjeżdżających i wyjeżdżających z centrum miasta. Odczytane numery tablic są porównywane z bazą pojazdów, które uiściły opłatę i w przypadku braku opłaty wystawiany jest wysoki mandat karny. Ocenia się, że system rejestruje około 98% pojazdów poruszających się w strefie objętej opłatą. Zarejestrowane obrazy są przesyłane do dwóch serwerowni, z których jedna odpowiada za analizę danych, a druga stanowi składowisko kopii zapasowych, które mogą służyć jako dowody w przypadku roszczeń.

Rysunek 2.2: Zestaw kamer systemu Congestion Charge, umieszczony przy stacji Pimlico w Londynie. Źródło: http://en.wikipedia.org/wiki/London_congestion_charge

2.3.2. Silnik ANPR Talon

Talon (http://ndi-rs.com/ukrs/software) opracowany został przez firmę NDI Recognition Systems, jako generyczny silnik ANPR. Wykorzystywany jest w produktach Speedplate (instalacje mobilne) i Veriplate (instalacje stacjonarne), tej samej firmy. System wykorzystuje sieci neuronowe do rozpoznawania znaków na tablicach rejestracyjnych i wg producenta osiąga skuteczność rozpoznawania na poziomie 97%.

2.3.3. DTK ANPR SDK

DTK ANPR (http://www.dtksoft.com/dtkanpr.php), to biblioteka opracowana przez firmę DTK Software, która może być zastosowana do szybkiego tworzenia rozwiązań wykorzystujących rozpoznawanie tablic rejestracyjnych. Tablice mogą być wyszukiwane zarówno na obrazach nieruchomych jak i ruchomych, a zbiór obsługiwanych tablic może zostać zawężony jedynie do określonego kraju, wybranego z pośród kilkudziesięciu dostępnych.

2.3.4. SeeCar

SeeCar (http://www.htsol.com/Products/SeeCar.html) to linia produktów ANPR, stworzona przez izraelską firmę Hi-Tech Solutions, w skład której wchodzą systemy:

- SeeWay (wysokie prędkości pojazdów, wyzwalanie sprzętowe),
- SeeRoad (wysokie prędkości pojazdów, wyzwalanie programowe),
- SeeLane (niskie i średnie prędkości pojazdów).

Każdy z produktów dostarczany jest w postaci biblioteki DLL dla systemu Windows, co pozwala na budowanie systemów, dopasowanych do wymagań klientów. Ponadto producent sprzedaje kamery wraz z podświetleniem, zaprojektowane specjalnie z myślą o zastosowaniu z produktami SeeCar.

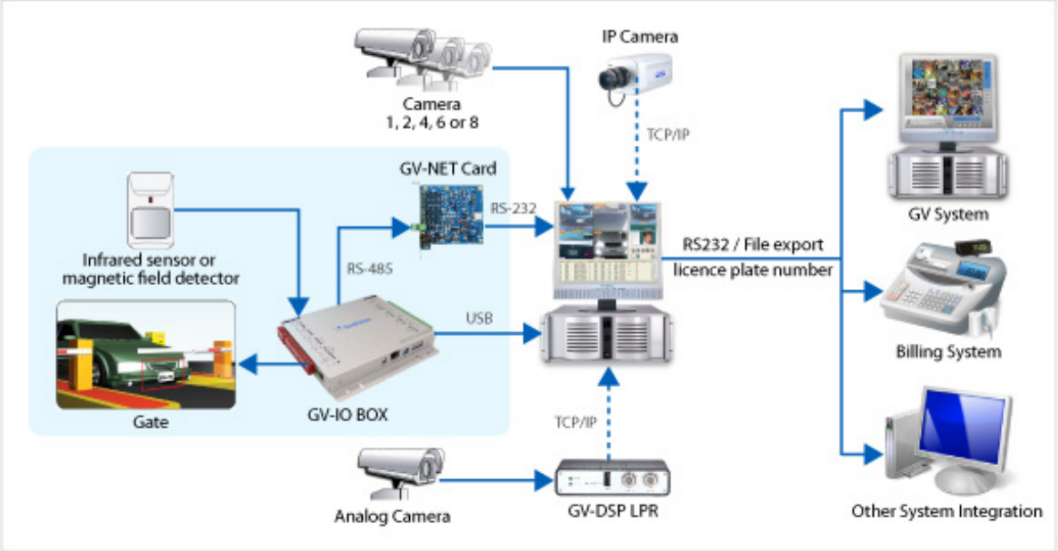
2.3.5. GV-LPR

Na polskim rynku dostępnych jest także wiele komercyjnych systemów ANPR, produkcji zarówno krajowej, jak i zagranicznej. Przykładem zagranicznych produktów jest GV-LPR, stworzony przez tajwańską firmę Geovision Inc.

System składa się z komputera stacjonarnego, z zainstalowanym oprogramowaniem GV-LPR oraz zestawu urządzeń peryferyjnych (kamery, czujniki ruchu, systemy billingowe), pozwalających na budowę złożonych systemów (rysunek 2.3). Przeznaczeniem GV-LPR jest głównie obsługa parkingów, choć producent podaje m.in. przykład wykorzystania systemu, do lokalizacji skradzionych samochodów na Tajwanie [15]. Rozpoznawanie może odbywać się w zarówno w trybie statycznym (z wykorzystaniem bramek oraz czujników zbliżeniowych) oraz dynamicznym (wyszukiwanie tablic na wszystkich klatkach obrazu z kamery).

Główne cechy systemu (wg producenta):

- skuteczność rozpoznawania 99%,



Rysunek 2.3: Infrastruktura systemu Geovison GV-LPR. Źródło: http://www.geovision.com.tw/english/Datasheet/Datasheet_LPR.pdf

- czas rozpoznawania <0,2 s,
- obsługa do 8 wejść wyzwalających,
- obsługiwane rozdzielczości 720 x 480 (NTSC) / 720 x 576 (PAL),
- licznik liczby pojazdów na parkingu,
- możliwość uruchamiania rozpoznania w przypadku wykrycia ruchu w wejściowym obrazie,
- możliwość sygnalizacji rozpoznania na wyjściu lub uruchomienia wybranej aplikacji,
- minimalne wymagania: procesor Core2 Duo E6300 / 1.86GHz, 1GB RAM, 250GB HDD.

2.3.6. ProVida ANPR

Innym popularnym rozwiązaniem jest, stosowany m.in. przez londyńską Metropolital Police, system ProVida ANPR (rysunek 2.4).

Unikalną cechą tego produktu jest możliwość wykorzystywania w samochodzie, przy prędkościach dochodzących do 100 mil na godzinę, ze skutecznością rozpoznawania na poziomie 95% . Pozwala to na stosowanie go m.in. w pojazdach patrolowych na autostradach. Urządzenie może pobierać sygnał wideo z dwóch kamer np. umieszczonych z przodu i tyłu radiowozu. Jednocześnie częstotliwość rozpoznawania może dochodzić nawet do 50 rozpoznań na sekundę (2ms na tablicę).

2.3.7. M3S-ANPR

Ostatni z opisanych systemów, M3S-ANRP [27], to autorski produkt polskiej firmy Polixel S.A.

Struktura systemu składa się z centralnego serwera gromadzącego dane o rozpoznanych tablicach oraz serwerów rozpoznających, które analizują ob-

Rysunek 2.4: System ProVida ANPR w użyciu. 1) Obraz tablicy; 2) Rozpoznany numer; 3) Informacje operacyjne; 4) Podgląd obrazu. Źródło: http://www.petards.com/emergency_services/literature/provida_anpr.pdf

razy z kamer stosując m.in. detekcję ruchu jako metodę wyzwalania procesu rozpoznawania.

Analiza obrazu na serwerze rozpoznającym *PlateFinder*, może być przeprowadzana zarówno dla pojazdów poruszających się jak i stojących. Serwery obsługiwać mogą do czterech kamer, z których obraz jest równocześnie poddawany analizie w trybie on-line. Zadaniem serwera centralnego jest przechowywanie wyników w bazie SQL oraz udostępnianie ich stacjom zarządzania oraz maszynom łączącym się przez interfejs WWW.

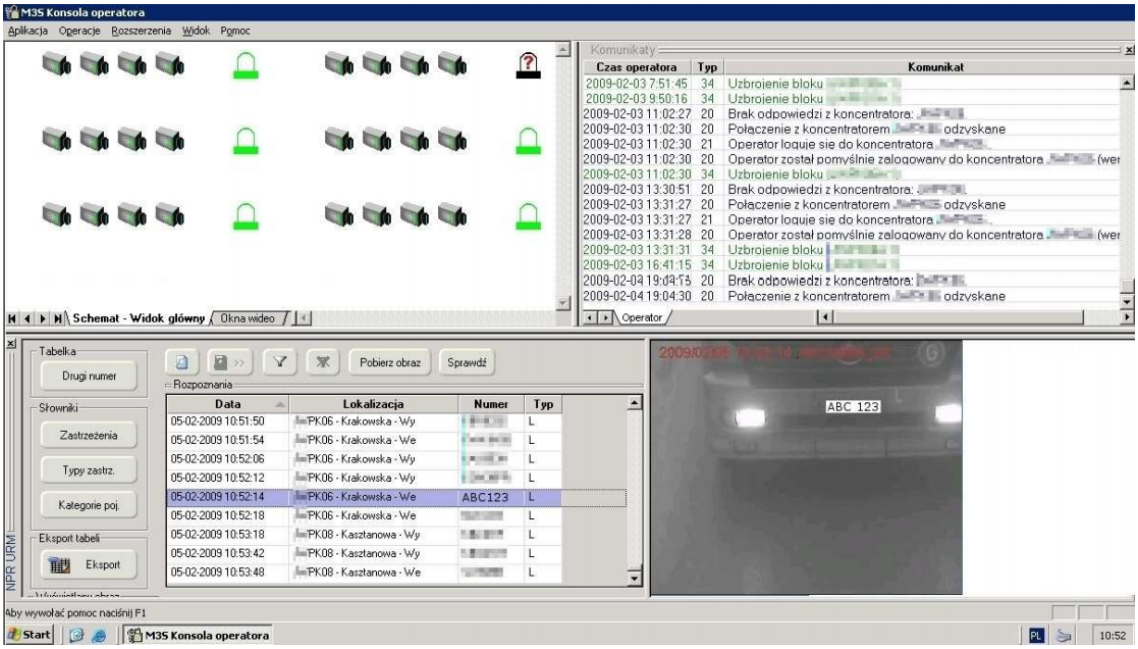
Każda ze stacji zarządzania umożliwia przeglądanie, analizę rozpoznanych tablic oraz zarządzanie bazą pojazdów poszukiwanych (rysunek 2.5). Ponadto stacje zarządzania mają możliwość podglądu na żywo obrazu z kamer oraz ich zdalnej konfiguracji.

”Właściwości systemu:

- Bardzo krótki czas analizy obrazu
- Rozpoznawanie numerów rejestracyjnych pojazdów poruszających się z szybkością do 150 km/h
- Odczytywanie numerów z tablic białych, czarnych oraz kolorowych
- Zapisywanie do bazy danych rozpoznanych numerów rejestracyjnych
- Dostęp do systemu przez przeglądarkę internetową
- Szerokie możliwości filtrowania i wyszukiwania danych w bazie
- Prowadzenie rejestru pojazdów poszukiwanych, uprawnionych lub nieuprawnionych do wjazdu itp.
- Możliwość integracji z zewnętrznymi bazami danych przechowującymi in-

formacje o pojazdach poszukiwanych, uprawnionych lub nieuprawnionych do wjazdu itp.

- Określanie prędkości pojazdów
- Definiowanie szablonów określających kraj pochodzenia i typ tablicy
- Rozpoznawanie tablic rejestracyjnych (transfer do bazy danych Policji)
- Zliczanie pojazdów
- Różnicowanie pojazdów
- Automatyczna detekcja pojazdów jadących pod prąd
- Automatyczne powiadamianie o kolizjach i wypadkach
- Alarmowanie o zapaleniu się pojazdu
- Automatyczne powiadamianie o zablokowaniu pasa ruchu" [27]



Rysunek 2.5: Aplikacja kliencka używana na stacjach zarządzających. Źródło: [27]

3. Algorytmy rozpoznawania tablic rejestracyjnych

3.1. Wstęp

Proces rozpoznawania numeru rejestracyjnego, umieszczonego na tablicy, można sprowadzić do trzech podstawowych kroków:

1. Lokalizacja tablicy na obrazie.
2. Wyodrębnienie znaków na tablicy (segmentacja).
3. Rozpoznanie znaków za pomocą algorytmu OCR i sprawdzenie poprawności składniowej wyniku.

W dalszej części rozdziału opisano najpopularniejsze algorytmy, jakie można spotkać w literaturze. Dokonany został także podziału algorytmów na trzy grupy, odpowiadające poszczególnym krokom procesu rozpoznawania tablicy rejestracyjnej.

3.2. Lokalizacja tablic

Jakość i wydajność całego procesu zależy głównie od jego pierwszego kroku, czyli lokalizacji tablicy na obrazie. Dlatego właśnie ten temat jest podejmowany w wielu pracach naukowych i badawczych. Zestawienie najpopularniejszych algorytmów lokalizacji można znaleźć w pracach [5] i [21].

3.2.1. Wykrywanie krawędzi i rzut jasności

Algorytm wykorzystujący wykrywanie krawędzi i rzut jasności opiera się na założeniu, że tablice rejestracyjne zwykle występują na obrazie jako fragmenty o wysokim kontraście (czarne litery na białym lub żółtym tle, białe litery na czarnym tle). Ponadto znaki na tablicy znajdują się w jednym rzędzie, co powoduje dużą częstotliwość zmian jasności obrazu w linii poziomej. Pozwala to na szukanie tablic rejestracyjnych w wierszach, w których występuje duża liczba wyraźnych zmian jasności. Zgodnie z tym założeniem algorytm wyznacza najpierw liczbę zmian jasności w każdym wierszu, a następnie wyszukuje wiersze o największej liczbie zmian, jako najbardziej prawdopodobne miejsca występowania tablic. Następnie proces powtarza się dla kolumn, w wybranych obszarach poziomych, celem wyznaczenia pionowych granic tablicy.

Pierwszym krokiem procesu jest zastosowanie algorytmu wykrywania krawędzi na obszarze całego obrazu. Jego zadaniem jest uwydatnienie obszaru tablicy rejestracyjnej, która powinna zawierać wyraźnie większą liczbę krawędzi niż reszta obrazu. Dobrze sprawdza się w tym celu operator Sobel'a,

dokonujący dyskretnego różniczkowania i aproksymacji pochodnych kierunkowych intensywności obrazu.

Matematycznie, operator używa dyskretnego splotu dwóch macierzy (jąder) przekształcenia z macierzą obrazu. Jedna z macierzy wykrywa krawędzie poziome, druga pionowe. Jeśli zdefiniujemy $\mathbf{A}$ jako macierz obrazu, to wyznaczenie macierzy pochodnych kierunkowych $\mathbf{G}_x$ i $\mathbf{G}_y$, można uzyskać wg wzorów 3.1 i 3.2 (gdzie * oznacza dwuwymiarowy operator splotu).

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \tag{3.1}$$

$$\mathbf{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \tag{3.2}$$

Przykład obrazu przed i po zastosowaniu wykrywania krawędzi przedstawiają rysunki 3.1a i 3.1b.

(a) Obraz oryginalny

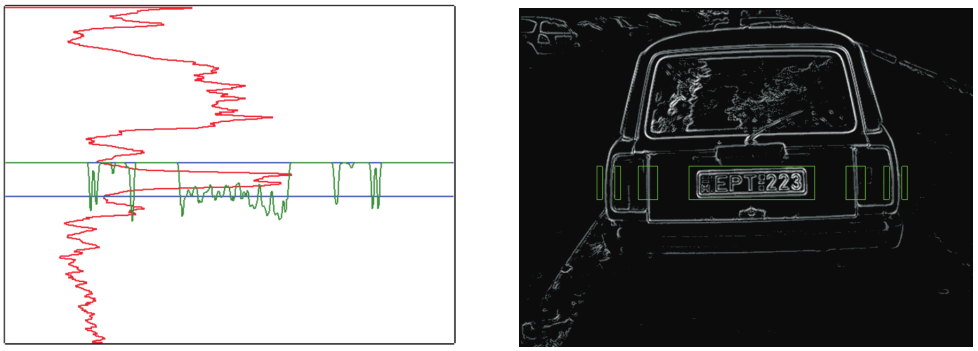
(b) Obraz poddany wykrywaniu krawędzi

Rysunek 3.1: Zastosowanie operatora Sobel'a w celu wykrycia krawędzi znaków na tablicy. Źródło: [5]

Na tak przetworzonym obrazie, pozycja pionowa tablicy rejestracyjnej może zostać określona, poprzez zsumowanie liczby krawędzi w każdym wierszu, a następnie wybranie zakresów zawierających najwyższe wyniki. Wyznaczanie pasma rozpoczyna się od wybrania wiersza o największej amplitudzie, a następnie poszukiwanie wartości granicznych pasma powyżej i poniżej tego wiersza. Granicę znajduje się poprzez poszukiwanie pierwszego lokalnego minimum znajdującego się za wierszem zawierającym połowę wartości maksymalnej. Aby poprawić skuteczności tej metody, wartości w wierszach, po zsumowaniu, poddawane są działaniu filtra dolnoprzepustowego np. poprzez uśrednianie wartości w pewnym przedziale, co zmniejsza liczbę lokalnych minimów.

W kolejnym kroku wyznaczane są poziome granice obszaru tablicy rejestracyjnej. Wykorzystywana w tym celu metoda jest analogiczna do zastosowanej w poprzednim kroku, z tą różnicą, że sumowane są jedynie te wartości

w kolumnach, które wyznaczone zostały w poprzednim kroku procesu (rysunek 3.2a i 3.2b). W pewnych przypadkach możliwe jest rozbicie tablicy na kilka potencjalnych obszarów lub wybór regionów nie będących tablicami (rysunek 3.2b). Oznacza to konieczność wyboru najbardziej prawdopodobnej pozycji z pośród dostępnych, poprzez analizę rozmiarów i proporcji obszarów i ew. łączenie obszarów znajdujących się w niewielkiej odległości.



(a) Uśrednione sumy w wierszach i kolumnach

(b) Prawdopodobne elementy tablicy

Rysunek 3.2: Lokalizacja tablicy dzięki zastosowaniu rzutów jasności dla wykrytych krawędzi. Źródło:[5]

3.2.2. Wykrywanie krawędzi z zastosowaniem funkcji okna

Metoda opisana w poprzednim podrozdziale, nie sprawdza się dla obrazów zawierających w tle dużą liczbę szczegółów i krawędzi. W takim przypadku rzut jasności dla tablicy nie wyróżnia się wystarczająco i może zostać pominięty na rzecz innego fragmentu obrazu. Zjawisko to można zminimalizować poprzez zastosowanie, podczas wyznaczania rzutu jasności, funkcji okna o rozmiarze dopasowanym do rozmiaru tablicy na obrazie. Podczas przesuwania okna wyznaczane są, dla każdego z wierszy, wartości maksymalne sumy w oknie, co minimalizuje wpływ rozproszonych wierszy, zawierających dużą liczbę krawędzi. Jeśli rozmiar okna zostanie zrównany z rozmiarem obrazu, to wynik będzie identyczny jak w poprzedniej metodzie, jeśli jednak wybrane okno będzie zbyt wąskie to wyniki będą niepoprawne. Wpływ zastosowania funkcji okna na wynik wyszukiwania przedstawiają rysunki 3.3a-d.

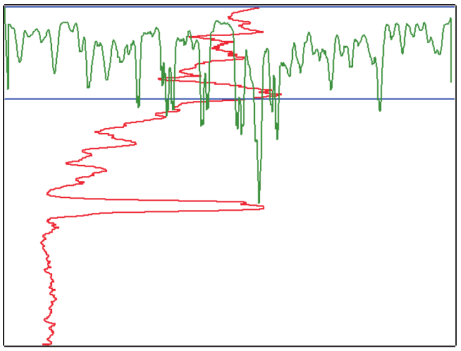
3.2.3. Transformata Hougha

Metoda składa się z pięciu następujących po sobie kroków (kroki 2-4 wykonywane są w dwóch potokach - dla linii poziomych i pionowych).

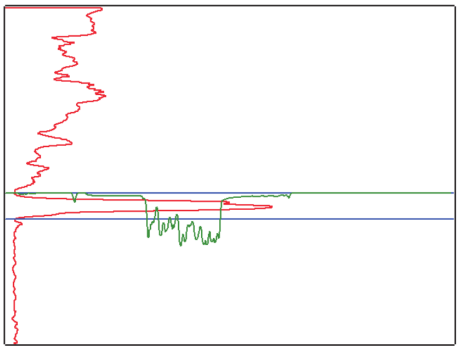
1. Progowanie obrazu wejściowego (przejście ze skali szarości do obrazu czarno-białego).
2. Wykrywanie krawędzi.
3. Wykonanie Transformaty Hougha, w celu wygenerowania listy linii równoległych.
4. Połączenie linii w segmenty.

(a) Obraz o złożonym tle

(b) Obraz po zastosowaniu operatora Sobel'a



(c) Rzuty jasności dla wierszy i kolumn



(d) Rzuty jasności z zastosowaniem funkcji okna

Rysunek 3.3: Zastosowanie funkcji okna w celu zmniejszenia wpływu złożonego tła na proces lokalizacji tablicy. Źródło:[5]

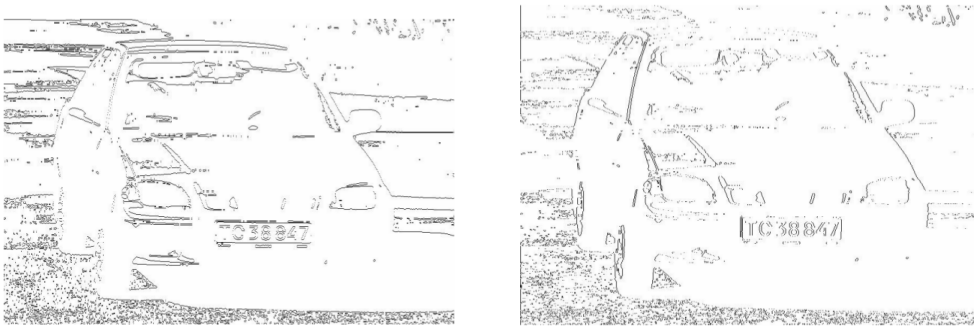
5. Połączenie linii poziomych i pionowych w prostokątne obszary i wybranie tych, które odpowiadają parametrom tablicy rejestracyjnej.

Wynikiem działania algorytmu są obszary obrazu, które są kandydatami na tablicę rejestracyjną. Progowanie może być przeprowadzone z zastosowaniem stałego progu, wyznaczonego na podstawie warunków oświetleniowych obrazu, lub w oparciu o progowanie adaptacyjne. Punkty 2-5 opisane są dokładniej w kolejnych podrozdziałach.

3.2.3.1. Wykrywanie krawędzi

Celem zastosowania wykrywania krawędzi jest zmniejszenie ilości informacji zawartych w obrazie źródłowym, poprzez usunięcie wszystkiego poza krawędziami w kierunku pionowym, bądź poziomym. Krawędzie wykrywane są przy użyciu operacji filtrowania przestrzennego (*ang. Spacial Filtering*), z zastosowaniem macierzy przekształcenia opisanej wzorem 3.3; osobno dla kierunku poziomego i pionowego. Wynikiem działania tego algorytmu są krawędzie o grubości jednego piksela, które doskonale sprawdzają się jako dane wejściowe dla transformaty Hougha (rysunek 3.4).

$$\mathbf{R_{poziome}} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \mathbf{R_{pionowe}} = \begin{bmatrix} -1 & 1 \end{bmatrix} \tag{3.3}$$



(a) Krawędzie poziome (b) Krawędzie pionowe

Rysunek 3.4: Wynik działania wykrywania krawędzi w obu kierunkach, na obrazie wcześniej poddanym progowaniu. Źródło:[16]

3.2.3.2. Wyszukiwanie prostych przy użyciu Transformaty Hougha

Transformata Hougha [16] jest metodą pozwalającą na wykrywanie linii na obrazach binarnych. Została opracowana jako alternatywa dla metody siłowej (*ang. Brute Force*), która jest bardzo złożona obliczeniowo. W przypadku metody siłowej znalezienie wszystkich linii łączących n punktów wymagałoby wyznaczenia wszystkich linii, łączących każde dwa punkty, a następnie sprawdzenia czy jakiekolwiek inne punkty należą do tych linii. Wynika z tego, że jeśli jest w przybliżeniu $n(n-1)/2$ tych linii, a na każdą z linii przypada n porównań, to jest to algorytm o złożoności $O(n^3)$.

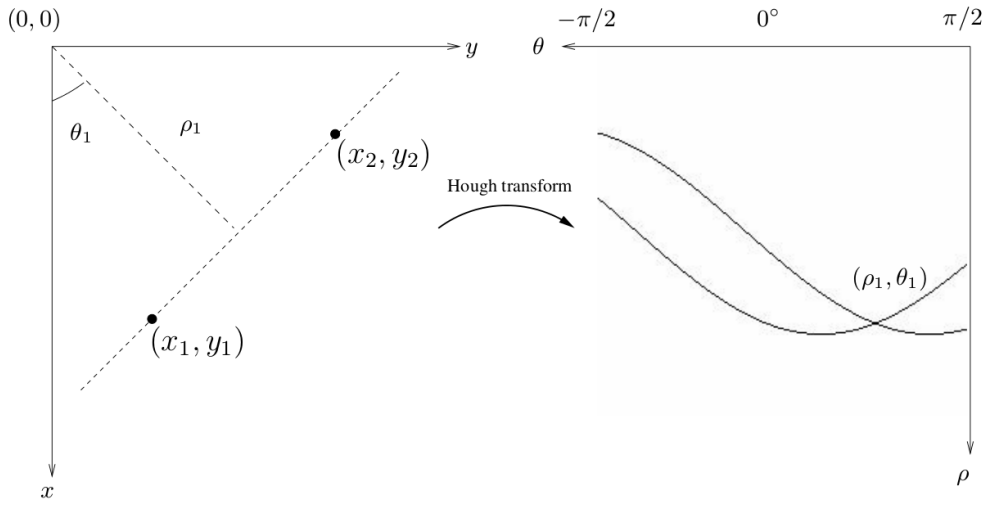
Transformata Hougha działa natomiast w czasie liniowym. Jej działanie opiera się na możliwości zapisu każdej prostej w postaci, jak na równaniu 3.4.

$$x\cos\theta + y\sin\theta = \rho \tag{3.4}$$

W równaniu tym parametr θ oznacza kąt pomiędzy normalną do linii i osią X. Parametr ρ stanowi odległość pomiędzy linią i początkiem układu współrzędnych. Punkty na obrazie odpowiadają krzywym sinusoidalnym w przestrzeni $\rho\theta$. Ilustruje to rysunek 3.5. Zgodnie z tym rysunkiem, jeśli za dziedzinę θ przyjmiemy przedział $\pm\frac{\pi}{2}$, to dla obrazu o wymiarach $w \times h$, przedział ρ wynosi od 0 do $\sqrt{w^2+h^2}$. Ponadto warto zauważyć, że linia przechodząca przez punkty (x_1, y_1) i (x_2, y_2) odpowiada parametrom punktu przecięcia się krzywych reprezentujących te punkty w przestrzeni Hougha. Zgodnie z powyższym, celem algorytmu jest wykrycie punktów w przestrzeni Hougha, w których przecinać będzie się wiele krzywych. Każda z tych krzywych reprezentować będzie dokładnie jeden z punktów tworzących linię, a punkty przecięcia dużej liczby krzywych, będą stanowiły interesujące, z punktu widzenia algorytmu, linie.

3.2.3.3. Łączenie linii w segmenty

Linie wykryte w poprzednim kroku należy połączyć ze sobą, aby uzyskać segmenty, które stanowią grupy równoległych linii znajdujących się w niewielkiej odległości. W tym celu należy posortować wszystkie punkty wg ich pozycji w linii, a następnie zgrupować je w segmenty. Jako, że znajdują się



Rysunek 3.5: Graficzna interpretacja transformaty Hougha. Źródło: [16]

one w kolejności, na skutek sortowania, to wystarczy iterować po wszystkich punktach i sprawdzać, czy kolejny punkt w linii, znajduje się w pewnej maksymalnej odległości, dobranej eksperymentalnie. Jeśli tak, to segment jest rozszerzany o punkt, a jeśli nie to rozpoczynany jest nowy. Rysunek 3.6 przedstawia przypadek idealny, gdzie najdłuższe segmenty zostały wyznaczone, a krótkie linie w literach i nad tablicą zostały usunięte.

Rysunek 3.6: Przykładowa tablica rejestracyjna i wyszukane segmenty. Źródło: [16]

3.2.3.4. Selekcja obszarów-kandydatów

Ostatnim krokiem jest wyszukiwanie obszarów, które mogą zawierać tablice rejestracyjne. Danymi wejściowymi są listy segmentów zbudowanych na postawie linii pionowych i poziomych. Obszarami mogą być pary segmentów, które spełniają następujące warunki:

1. Muszą zaczynać się i kończyć, w przybliżeniu, w tym samym miejscu.
2. Współczynnik długości segmentów, do odległości pomiędzy nimi, powinny odpowiadać tym ustalonym dla tablic rejestracyjnych.

Wynikowe listy par obszarów (jedna dla pionowych, a druga dla poziomych) są ze sobą porównywane. Jeśli jakikolwiek obszar występuje tylko na jednej z nich, to zostaje on pominięty. Pozostałe obszary uznawane są za kandydatów na tablice. Metoda jest niezależna od wielkości regionu, ponieważ wyszukiwane są jedynie obszary o określonych proporcjach, a algorytm wyszukiwania krawędzi i transformata Hougha są niezależne od skali obrazu.

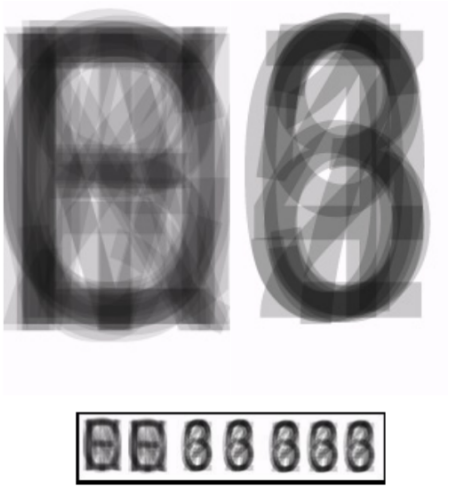
Podstawowymi wadami tej metody są problemy z krawędziami pionowymi (ze względu na mniejszą długość, są bardziej podatne na zakłócenia) oraz fakt wyznaczania obszarów nie zawierających tablic rejestracyjnych (wyszukiwane są wszystkie obszary o określonych proporcjach).

3.2.4. Wyszukiwanie wzorców

Lokalizacja tablic rejestracyjnych z wykorzystaniem wyszukiwania wzorców (*ang. Template Matching*), to algorytm opierający się na przesuwaniu wzorca po całym obszarze obrazu i wyznaczaniu miejsca, w którym współczynnik podobieństwa pomiędzy obrazem i wzorcem jest największy. Wyznaczanie maksimum może być przeprowadzane poprzez proste przesuwanie wzorca piksel po pikselu lub, bardziej wydajnie, z zastosowaniem różnorodnych technik optymalizacyjnych np. algorytmów bezgradientowych.

3.2.4.1. Tworzenie wzorca

Dużą rolę w skuteczności algorytmu odgrywa stworzenie dobrego wzorca tj. takiego, który będzie posiadał jak najwięcej cech charakterystycznych dla tablic pojawiających się na obrazach. Podstawowym założeniem dla tej metody jest fakt, że tablica będzie zawsze miała stałą i znaną wcześniej wielkość. Wynika to z konieczności określenia wielkości wzorca, który musi odpowiadać rozmiarami tablicy na obrazie. Ponadto w celu zmniejszenia wpływu warunków oświetleniowych, zarówno podczas konstrukcji wzorca, jak i podczas jego dopasowywania, stosowane jest progowanie adaptacyjne. Ważne jest także pominięcie obramowania tablicy, które może wprowadzać duże błędy nawet w przypadku niewielkiego obrotu tablicy na obrazie względem wzorca. Wzorzec najlepiej zbudować poprzez nakładanie na siebie warstw zawierających znaki. Każda z nich powinna mieć współczynnik nieprzezroczystości odpowiadający prawdopodobieństwu wystąpienia danego znaku na określonej pozycji (rysunek 3.7).



Rysunek 3.7: Wzorzec powstały przez złączenie warstw. Źródło: [16]

3.2.4.2. Wyznaczanie współczynnika korelacji

Popularną metodą określania stopnia podobieństwa pomiędzy obrazami jest zastosowanie miary zwanej współczynnikiem korelacji. Współczynnik ten oparty jest na odległości Euklidesowej w przestrzeni dwuwymiarowej i opisywany równaniem 3.5. Równianie to wylicza sumę kwadratów odległości pomiędzy pikselami wzorca, a pikselami fragmentu obrazu porównywanego ze wzorcem. W celu dalszych uproszczeń, równanie 3.5 należy rozwinąć do postaci 3.6.

$$d^2_{f,t}(u,v) = \sum_x \sum_y [f(x,y) - t(x-u, y-v)]^2 \tag{3.5}$$

$$d^2_{f,t}(u,v) = \sum_x \sum_y [f^2(x,y) - 2f(x,y)t(x-u,y-v) + t^2(x-u,y-v)] \tag{3.6}$$

Można przyjąć, że zarówno składnik $\sum_x \sum_y f^2(x,y)$, jak i $\sum_x \sum_y t^2(x-u,y-v)$, są stałe, nie wpływają więc na współczynnik korelacji, który w ostatecznej postaci można zapisać równaniem 3.7.

$$d^2_{f,t}(u,v) = \sum_x \sum_y f(x,y)t(x-u,y-v) \tag{3.7}$$

Niestety założenie stałości składnika $\sum_x \sum_y f^2(x,y)$, jest prawdziwe jedynie dla obrazów, na których intensywność poszczególnych partii różni się nieznacznie. W dużej części przypadków, współczynnik korelacji będzie więc wyższy na jaśniejszych obszarach obrazu, niż w poszukiwanych fragmentach, które powinny najlepiej pasować do wzorca. Dlatego właśnie konieczne jest zastosowanie znormalizowanego współczynnika korelacji, opisanego równaniem 3.8.

$$\gamma(u,v) = \frac{\sum_x \sum_y [f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\sqrt{\sum_x \sum_y [f(x,y) - \bar{f}_{u,v}]^2 \sum_x \sum_y [t(x-u,y-v) - \bar{t}]^2}} \tag{3.8}$$

$\bar{f}_{u,v}$ oznacza średnią wartość pikseli obrazu we fragmencie przykrytym przez wzorzec, a $\bar{t}$ oznacza wartość średnią pikseli wzorca. Wartość znormalizowanego współczynnika zawiera się pomiędzy -1 a 1, gdzie -1 oznacza odwrócone dopasowanie (obraz jest negatywem wzorca), a 1 oznacza idealne dopasowanie. Powyższa metoda stanowi dobry i prosty sposób porównywania obrazów. Jednak jej przydatność do lokalizacji tablic rejestracyjnych, ogranicza wiele negatywnych cech zastosowanego algorytmu:

- powolność dla dużych obrazów, ze względu na zwiększoną liczbę operacji koniecznych do wyznaczenia znormalizowanego współczynnika korelacji,
- wrażliwość na obrót i zniekształcenia perspektywiczne,
- wrażliwość na skalowanie,
- konieczność stosowania progowania adaptacyjnego, w celu zmniejszenia wpływu rozkładu jasności,

- brak możliwości stworzenia jednego wzorca dla polskich tablic rejestracyjnych, ze względu na istnienie kilku układów znaków na tablicach.

Metoda ta znalazła jednak zastosowanie jako szybki i prosty algorytm OCR, pozwalający na wybór najbardziej odpowiadającego wzorca dla danego znaku.

3.2.5. Operacje morfologiczne

Zastosowanie wyłącznie operacji morfologicznych zaproponowane zostało w pracy [18]. Idea tej metody opiera się na założeniu, że tablice rejestracyjne posiadają określone cechy charakterystyczne:

- składają się z obiektów ciemnych (lub jasnych dla starych tablic rejestracyjnych) na kontrastowym jasnym (ciemnym) tle,
- ułożone są równolegle do osi poziomej obrazu.

W celu zwiększenia skuteczności, algorytm przyjmuje parametr h, który stanowi wysokość znaku wyrażoną w pikselach. Wielkość tą należy wyznaczyć podczas kalibracji kamery.

3.2.5.1. Poprawa kontrastu

Aby tablica rejestracyjna mogła być poprawnie zlokalizowana niezbędna jest zmiana kontrastowości obrazu wejściowego. Dzięki temu możliwe jest uwydatnienie różnic poszczególnych elementów, szczególnie tych znajdujących się na powierzchni tablicy rejestracyjnej. W tym celu stosowane są dwa operatory typu top-hat. „Operacja ta jest wykonywana z jednostkowym elementem strukturującym ośmiospójnym (typu E), jej rozmiar wynosi h/2, gdzie h jest wysokością znaku.” [18] (równanie 3.9).

$$f_1 = f + WTH_E^{(h/2)}(f) - BTH_E^{(h/2)}(f) = 3f - \gamma_E^{(h/2)}(f) - \varphi_E^{(h/2)}(f) \quad (3.9)$$

Wynik operacji poprawy kontrastu prezentują rysunki 3.8a i 3.8b.

(a) Obraz oryginalny (b) Wynik poprawy kontrastu

Rysunek 3.8: Poprawa kontrastu obrazu. Źródło: [18]

3.2.5.2. Czyszczenie tła

Faza usuwania tła z obrazu stanowi kluczowy element algorytmu. Jej zadaniem jest eliminacja jak największej liczby elementów obrazu nie będących znakami tablicy rejestracyjnej. Metodę komplikuje fakt, że algorytm

ma działać zarówno dla tablic nowego, jak i starego typu, co oznacza, że trzeba zastosować kombinację operacji zorientowanych na każdy z przypadków. Do tego właśnie stosuje się operatory top-hat przez rekonstrukcje. Operator biały wyszukuje znaki białe (stare tablice), a operator czarny - czarne (nowe tablice). Następnie wyniki obu operacji łączone są poprzez zastosowanie operacji *supremum*, wyznaczającej punktowe maksimum dwóch obrazów. Rozmiar filtrów jest dobrany tak, aby nie były one większe niż szerokość linii tworzącej znak, która stanowi 20% wysokości znaku, przyjętej jako parametr h. Równanie 3.10 przedstawia matematycznie całość procesu czyszczenia tła, a rysunek 3.9a ilustruje wynik tej fazy algorytmu.

$$f_2 = WTHR_{E,E}^{(h/5)}(f_1) \vee BTHR_{E,E}^{(h/5)}(f_1) \tag{3.10}$$

(a) Obraz poddany czyszczeniu (b) Zgrubne wyszukiwanie tablicy

Rysunek 3.9: Czyszczenie obrazu i lokalizacja tablicy. Źródło: [18]

3.2.5.3. Wykrycie obszaru tablicy rejestracyjnej

Tablica lokalizowana jest w trzech etapach:

- zgrubne wykrycie obszaru tablicy,
- filtracja, w celu precyzyjnego określenia położenia,
- progowanie.

Przybliżone określenie obszaru tablicy możliwe jest dzięki zastosowaniu operacji zamknięcia z elementem strukturującym (typu H), działającym w kierunku poziomym. Wykorzystanie tego typu elementu wynika z poziomego położenia znaków oraz niewielkich odległości pomiędzy nimi. Rozmiar zamknięcia został ustalony na *4h*, co pozwoliło na połączenie nawet najbardziej odległych znaków na tablicy.

Rysunek 3.9b przedstawia wynik zgrubnego wyszukiwania tablicy. Jak widać obszar tablicy został wyróżniony, choć otoczony jest też zakłóceniami, które są wyraźnie cieńsze.

Kolejny etap, czyli filtracja, polega na usunięciu zakłóceń otaczających obszar tablicy. W tym celu zastosowane zostało złożenie dwóch otwarć z kierunkowymi elementami strukturyzującymi: pionowym (typu V), a następnie poziomym (typu H) (równanie 3.11). Rozmiary elementów zostały dobrane

tak, aby usunąć jak najwięcej elementów nie będących tablicą rejestracyjną i wynosiły:

- dla otwarcia pionowego *h/2*,
- dla otwarcia poziomego *2h*.

$$f_3 = \gamma_H^{(2h)}(\gamma_V^{(h/2)}(\varphi_H^{(4H)}(f_2))) \tag{3.11}$$

3.3. Segmentacja

Po wydzieleniu tablicy z obrazu, konieczny jest podział jej na fragmenty zawierające poszczególne znaki. Proces ten nazywamy segmentacją. Jego wynikiem są obrazy przedstawiające jedynie znaki, które mogą być rozpoznawane algorytmem OCR. W pracach [16] i [24] opisywane są dwa podstawowe algorytmy segmentacji:

- metody wykorzystujące rozrost obszarów RGA (*ang. Region Growing Algorithm*),
- algorytmy opierające się na wykrywaniu krawędzi HEDA (*ang. Horizontal Edge Detection Algorithm*).

Każda z metod posiada swoje pole zastosowań. W pierwszym przypadku znaczną przeszkodą może okazać się np. śruba występująca na tablicy, która spowoduje połączenie dwóch znaków w jeden. W drugim przypadku na jakość segmentacji może mieć wpływ np. przekrzywienie tablicy na obrazie.

3.3.1. Algorytm wykorzystujący rozrost obszarów

Algorytmy tego typu wyszykują obszary składające się z elementów sąsiadujących ze sobą i posiadających jednakowy kolor. W większości przypadków operują one na obrazach 1-bitowych, co wymaga wstępnego przygotowania obrazu np. poprzez progowanie.

Obszary będące wynikiem działania algorytmu muszą być spójne. „Pod pojęciem «spójny obszar» należy rozumieć taki zbiór pikseli obrazu o jednakowej wartości (jasności, kolorze, indeksie – zależnie od interpretacji), w którym dwa dowolne piksele można połączyć linia łamana niewychodzącą poza tenże zbiór." [11] Często wykorzystywanym algorytmem rozrostu obszarów, jest algorytm rozrostu ziarna (*ang. Flood Fill*), który przeszukuje obraz w poszukiwaniu pikseli o określonej wartości. W przypadku gdy piksel taki zostanie znaleziony, zostaje oznaczony, a następnie sprawdzane są piksele sąsiadujące. W najprostszej rekurencyjnej implementacji problemem może okazać się rozmiar stosu, choć istnieją również wydajne implementacje iteracyjne.

3.3.2. Algorytm opierający się o rzut jasności

Algorytm opierający się o rzut jasności jest w gruncie rzeczy identyczny z algorytmem lokalizacji tablic opisanym w rozdziale 3.2.1, z tą różnicą, że jest wykonywany jedynie na obrazie tablicy. Dla każdej z pionowych linii obrazu

wykonywany jest rzut na oś X, będący sumą jasności poszczególnych pikseli, co można opisać wzorem:

$$P[i] = \sum_{j=0}^{h} f(i,j) \; dla \, i = 0, 1, 2..., w \tag{3.12}$$

gdzie: P - rzut jasności (tablica); i - numer kolumny, w - szerokość obszaru; h - wysokość obszaru.

Przerwy pomiędzy znakami stanowią lokalne minima na wykresie. Jeśli więc przyjmie się pewien niewielki próg, to miejsca, gdzie wartość spada poniżej tego progu mogą być uznane za granice znaków. Do minimalizacji liczby wyznaczonych w ten sposób znaków można zastosować dodatkowe kryteria heurystyczne np. stosunek wysokości do szerokości.

3.4. Rozpoznawanie znaków

Kolejny etap w procesie rozpoznawania tablicy, stanowi identyfikacja znaków wyróżnionych w procesie segmentacji. Zadanie to realizowane może być przez jeden z wielu algorytmów OCR (*ang. Optical Character Recognition*), które, z pośród dostępnych wzorców, wybierają najbardziej odpowiadający rozpoznawanemu znakowi.

Opisując algorytmy OCR często stosuje się pojęcia cechy i klasyfikatora. W przypadku rozpoznawania znaków (w szczególności znaków na tablicy rejestracyjnej), wykorzystywane są różnorodne cechy i klasyfikatory, opisane w kolejnych podrozdziałach.

3.4.1. Cechy

Cecha to własność obrazu (np. kształt, kolor, tekstura), która pozwala dopasować go do jednego z istniejących wzorców. W pewnych przypadkach istnieje konieczność zastosowania zestawu cech, w celu rozróżnienia wzorców podobnych pod względem jednej cechy, ale różnych pod względem innej.

3.4.1.1. Rzut jasności

Rzut jasności dla znaków wyznaczany jest w sposób podobny, do stosowanego przy lokalizacji tablicy w rozdziale 3.2.1. Dla każdej linii i wiersza obrazu zliczana jest suma czarnych pikseli, a uzyskane w ten sposób wektory wartości pozwalają łatwo porównywać wzorce z obrazem. Ze względu na fakt, że np. litery W i M mają bardzo podobny rzut w poziomie (rysunek 3.10), stosowane są jednocześnie rzut pionowy jak i poziomy.

Niestety metoda ta jest skuteczna tylko w przypadku gdy znak pokrywa się z wzorcem. Jeśli, na skutek niedokładnej segmentacji znak będzie przesunięty lub obrócony, to zmniejszają się szanse na jego poprawne rozpoznanie.

3.4.1.2. Mapy gęstości

„Cechy gęstościowe wyznacza się poprzez podział mapy bitowej znaku na pewną liczbę obszarów (identycznych dla wszystkich znaków) i zsumowanie wszystkich zapalonych pikseli w każdym z tych obszarów. W ten sposób

(a) Rzuty litery W (b) Rzuty litery M

Rysunek 3.10: Porównanie rzutów jasności liter W i M. Źródło: opracowanie własne.

powstaje wektor o elementach całkowitych o stałej długości." [12] Prostą metodą do uzyskania cechy jest przeskalowanie obrazu do rozdzielczości odpowiadającej wymiarom mapy gęstości. Cechy takie wykazują dużo większa niezmienność przy przesunięciu lub obrocie znaku. Wynika to z faktu, że wektor wynikowy generowany jest z sekcji obrazu, która w wyniku tych przekształceń nie zmienia znacząco liczby czarnych pikseli, a jedynie ich rozmieszczenie.

3.4.1.3. Cechy geometryczne

Powszechnie stosowne są również cechy geometryczne. Liczne przykłady tego typu cech podaje praca [29].

1. Odległość w poziomie, podana w pikselach, pomiędzy lewą krawędzią obrazu, a środkiem najmniejszego prostokąta zawierającego wszystkie „zapalone" piksele (prostokąta otaczającego).
2. Odległość w pionie, podana w pikselach, pomiędzy dolną krawędzią obrazu, a środkiem prostokąta otaczającego.
3. Wysokość prostokąta otaczającego, podana w pikselach.
4. Szerokość prostokąta otaczającego, podana w pikselach.
5. Liczba „zapalonych" pikseli na obrazie.
6. Średnie poziome przesunięcie wszystkich „zapalonych" pikseli względem środka prostokąta otaczającego, podzielone przez szerokość prostokąta otaczającego. Jeśli cecha posiada wartość ujemną, to obraz posiada „ciężar po lewej stronie", co można zaobserwować np. dla litery „L".
7. Średnie pionowe przesunięcie wszystkich „zapalonych" pikseli względem środka prostokąta otaczającego, podzielone przez wysokość prostokąta otaczającego.
8. Wartość średniokwadratowa poziomych odległości, wyznaczanych jak w punkcie 6. Cecha ta będzie przyjmowała większe wartości w przypadku

obrazów, które posiadają piksele rozmieszczone szerzej w poziomie np. w przypadku liter „W” i „M”.

9. Wartość średniokwadratowa pionowych odległości, wyznaczanych jak w punkcie 7.
10. Średni iloczyn odległości poziomej i pionowej, wyznaczanych jak w punktach 6 i 7. Cecha przyjmuje wartość dodatnią dla linii ukośnych przebiegających z dolnego lewego do górnego prawego rogu, a wartość ujemną dla linii przebiegających z górnego lewego do dolnego prawego rogu.
11. Średnia liczba krawędzi („zapalony” piksel bezpośrednio na prawo do „zgaszonego” piksela lub krawędzi obrazu) napotkanych podczas skanowania obrazu z lewej stron na prawą, dla wszystkich pionowych pozycji wewnątrz prostokąta otaczającego. Cecha ta pozwala na rozróżnienie liter „W” i „M” oraz „I” i „L”.
12. Suma pionowych pozycji wszystkich krawędzi napotkanych w punkcie 11. Cecha ta przyjmuje większe wartości, jeśli więcej krawędzi znajduje się w górnej części prostokąta otaczającego np. dla litery „Y”.
13. Średnia liczba krawędzi („zapalony” piksel bezpośrednio powyżej „zgaszonego” piksela lub krawędzi obrazu) napotkanych podczas skanowania obrazu z góry na dół, dla wszystkich poziomych pozycji wewnątrz prostokąta otaczającego.
14. Suma poziomych pozycji wszystkich krawędzi napotkanych w punkcie 13.

3.4.1.4. Momenty geometryczne

Wykorzystanie momentów do analizy kształtów zaproponowane zostało w pracy [31]. Metoda ta stosuje momenty, znane ze statystyki i mechaniki, do operowania na obrazie w skali szarości, przedstawionego jako dwuwymiarowa funkcja gęstości [34].

Definicję momentu $m_{p,q}$ rzędu $m+p$ dla funkcji gęstości $f(x,y)$ przedstawia równanie 3.13.

$$m_{p,q} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x,y) dx dy \tag{3.13}$$

W przypadku obrazu o rozmiarze $M \times N$ pikseli ($g(x,y)$), moment można wyznaczyć w sposób dyskretny zgodnie z równaniem 3.14.

$$m_{p,q} = \sum_{0}^{M-1} \sum_{0}^{N-1} x^p y^q g(x,y) \tag{3.14}$$

Momenty, w zależności od rzędu, wyznaczają różne własności obrazu np. :

- momenty rzędu 0 - całkowity obszar obiektu,
- momenty rzędu 1 - lokalizacja środka masy (równanie 3.15),

$$\bar{x} = \frac{m_{10}}{m_{00}} \, \bar{y} = \frac{m_{01}}{m_{00}} \tag{3.15}$$

- momenty rzędu 2 - momenty bezwładności, które pozwalają na określenie m.in. osi głównych obiektu, elipsy obrazu, promienia bezwładności.

3.4.1.5. Niezmienniki Hu

Podstawowym problemem w przypadku stosowania momentów do opisywania kształtu obiektu, jest ich zmienność pod wpływem przekształceń geometrycznych. W pracy [17], Ming-Kuei Hu wprowadził siedem niezmienników momentowych, niezależnych od translacji, zmian orientacji i skalowania, opisanych równaniami 3.16.

$$
\begin{aligned}
M1 &= \mu_{20} + \mu_{02} \\
M2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \\
M3 &= (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \\
M4 &= (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2 \\
M5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12}) - 3(\mu_{21} + \mu_{03})^2] \\
&\quad +(3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \\
M6 &= (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{03} + \mu_{21})^2] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{03} + \mu_{21}) \\
M7 &= (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] \\
&\quad -(\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]
\end{aligned}
\tag{3.16}
$$

Wektor wartości zbudowanych z niezmienników M1-7 stanowi dobry wyznacznik podobieństwa pomiędzy obrazami, ze względu na niewielką długość oraz niezmienności względem przekształceń geometrycznych.

3.4.2. Klasyfikatory

Wybór wzorca najlepiej pasującego do zadanego obrazu, to zadanie klasyfikatora. Jest to algorytm określający podobieństwo pomiędzy obrazem i wzorcami, w oparciu o wyselekcjonowane cechy i klasy wzorców. Większość klasyfikatorów zaliczyć można do jednej z poniższych grup.

3.4.2.1. Metody oparte o przykłady (pamięciowe)

Metody te uczą się poprzez zapamiętywanie danych treningowych, co często określane jest jako „leniwe uczenie się", ponieważ proces przetwarzania danych odwlekany jest, aż do momentu pojawienia się danych do rozpoznania. Klasyfikacja sprowadza się do przejrzenia danych treningowych i wybrania tych, które najbardziej pasują do danych poddawanych klasyfikacji. Przykładami metod pamięciowych są:

- metoda k-najbliższych sąsiadów, wykorzystująca reprezentację danych treningowych w przestrzeni metrycznej (szerzej opisana w podrozdziale 3.4.2.7),
- metoda modeli lokalnych (*ang. locally weighted regression*), oparta o konstruowanie lokalnych aproksymacji pojęć,
- metoda wnioskowania z przypadków (*ang. Case Based Reasoning*), wykorzystująca reprezentacje symboliczne i metody wnioskowania oparte o bazę wiedzy.

3.4.2.2. Wnioskowanie Bayesowskie

Naiwny klasyfikator bayesowski, stanowi prosty klasyfikator probabilistyczny wykorzystujący prawdopodobieństwa warunkowe, do określenia zależności pomiędzy kategoriami. Zakłada, że każdy z przykładów, składających się na dane treningowe, da się opisać za pomocą zestawu dyskretnych atrybutów. „Na podstawie zbioru trenującego T są szacowane prawdopodobieństwa poszczególnych kategorii pojęcia docelowego oraz prawdopodobieństwa poszczególnych wartości wszystkich atrybutów dla przykładów różnych kategorii."[2] Szacowanie odbywa się poprzez obliczanie częstotliwości występowania przykładów należących do określonych kategorii lub opisanych wybranymi atrybutami. Naiwny klasyfikator bayesowski sprawdza się szczególnie dobrze w zagadnieniach klasyfikacji tekstów.

3.4.2.3. Drzewa decyzyjne

"Przez drzewo decyzyjne rozumiemy strukturę, która ma zwykłe właściwości drzewa w znaczeniu, jaki temu słowu nadaje się w informatyce, jest więc strukturą złożoną z węzłów, z których wychodzą gałęzie prowadzące do innych węzłów lub liści, oraz z liści, z których nie wychodzą żadne gałęzie. Węzły odpowiadają testom przeprowadzanym na wartościach atrybutów przykładów, gałęzie odpowiadają możliwym wynikom tych testów, liście zaś etykietom kategorii."[2] Testami określa się funkcje, które pozwalają na podział zbioru przykładów na podzbiory, w zależności od wartości wybranego atrybutu lub grupy atrybutów. Klasyfikacja, w przypadku drzewa decyzyjnego, sprowadza się do znalezienia ścieżki od korzenia drzewa do jednego z jego liści, odpowiadających konkretnym kategoriom.

3.4.2.4. SVM: Maszyny wektorów nośnych

Metoda SVM służy do klasyfikacji binarnej, co oznacza, że służy do podziału danych wejściowych do jednej z dwóch klas. Celem metody jest wyznaczenie płaszczyzny dzielącej dane, przedstawione w przestrzeni metrycznej, na dwie klasy. Ponadto stara się zachować możliwie największy margines separacji, czyli odstęp pomiędzy płaszczyzną, a najbliższym do niej punktem.

3.4.2.5. Metoda wzmacniania klasyfikatorów (*ang. Boosting*)

Boosting jest metodą klasyfikacji, której zadaniem jest poprawienie działania innego algorytmy uczącego. Polega na stosowaniu zestawu prostych modeli, z których każdy kolejny przykłada coraz większą wagę do danych błędnie zakwalifikowanych przez poprzedników. Podstawowym przykładem Boostingu jest iteracyjny algorytm AdaBoost, którego autorami są Yoav Freund i Robert Schapire.

"W pierwszym kroku algorytm AdaBoost inicjalizuje wagi dla każdej obserwacji, nadając im tę samą wartość $\frac{1}{m}$. Dalej następuje iteracyjne:

1. normalizowanie wag ;
2. wyznaczanie klasyfikatora ;
3. wyznaczanie błędu klasyfikatora ;
4. wyznaczanie nowych wag na podstawie błędu klasyfikatora.

Proces iteracji kończy się, gdy błąd klasyfikatora przekroczy wartość $\frac{1}{2}$ lub, gdy numer iteracji równy jest danej stałej L (czyli ustalonej maksymalnej liczbie iteracji). Ostateczny klasyfikator wyznaczamy na podstawie metody zwanej głosowaniem większościowym."[14]

W dalszej części rozdziału, bardziej szczegółowo, rozpatrywane będą tylko metody, które mogą operować na dwumiarowych mapach bitowych. Wynika to z faktu, że mapy gęstości stanowiły rodzaj cechy lepiej sprawdzający się podczas rozpoznawania znaków z tablic rejestracyjnych.

3.4.2.6. Algorytm dopasowywania wzorca

Algorytm dopasowywania wzorca (*ang. Template Matching*) pozwala na określenie współczynnika podobieństwa obrazu wejściowego i wzorca, przy użyciu jednej z poniższych metod (litera I oznacza obraz wejściowy, T - wzorzec, a R - rezultat porównania).

Square difference matching method Wynikiem tej metody jest suma różnic pomiędzy wzorcem, a obrazem wejściowym. W przypadku idealnego dopasowania, współczynnik podobieństwa będzie równy 0 i będzie rósł wraz z liczbą różnic pomiędzy tymi obrazami.

$$R_{sq\_diff} = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2 \tag{3.17}$$

Correlation matching method Rezultat tej metody jest zależny od iloczynu obrazu i wzorca. W przypadku idealnego dopasowania wynik jest bardzo duży, dla złych dopasowań będzie mały, a w skrajnych przypadkach będzie będzie wynosił 0.

$$R_{ccor} = \sum_{x',y'} [T(x',y') \cdot I(x+x',y+y')]^2 \tag{3.18}$$

Correlation coefficient matching methods Sposób ten jest rozszerzeniem poprzedniej metody. Zamiast obrazu wejściowego i wzorca, porównywane są ich kopie, poddane uśrednieniu. Współczynnik dopasowania zawiera się pomiędzy wartościami:

- 1 - idealne dopasowanie,
- 0 - brak dopasowania, losowe piksele dopasowane,
- -1 - odwrotne dopasowanie (obraz jest negatywem wzorca).

$$R_{ccoeff} = \sum_{x',y'} [T'(x',y') \cdot I'(x+x',y+y')]^2 \tag{3.19}$$

$$T'(x',y') = T(x',y') - \frac{1}{(w \cdot h) \sum_{x'',y''} T(x+x'',y+y'')} \tag{3.20}$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{(w \cdot h) \sum_{x'',y''} I(x+x'',y+y'')} \tag{3.21}$$

3.4.2.7. Algorytm K-najbliższych sąsiadów

Algorytm K-najbliższych sąsiadów (*ang. K-nearest neighbors*), jest przykładem zastosowania regresji nieparametrycznej i stanowi jeden z najprostszych klasyfikatorów. Założeniem algorytmu jest to, że dany jest zbiór uczący składający się z wielu wektorów cech i przypisanych im etykiet oraz wektor cech C dla którego należy wyznaczyć etykietę[1] .

Algorytm polega na:

1. Porównaniu wektora cech C ze wszystkimi wektorami w zbiorze uczącym, wyznaczeniu podobieństwa pomiędzy nimi i zapamiętaniu tych wartości.
2. Posortowaniu zbioru uczącego wg wartości podobieństwa i wybraniu K etykiet najbardziej podobnych zbiorów.
3. Wybraniu najbardziej odpowiedniej wartości poprzez głosowanie np. z zastosowaniem mediany (wartości dyskretne etykiet) lub uśrednienie (wartości ciągłe). Podobieństwo zbiorów najczęściej określone jest poprzez zastosowanie pewnej metryki określającej odległość pomiędzy wektorami cech. Zwykle stosowane miary odległości:
   - Euklidesa (równanie 3.22),

$$D(x,y)^2 = \sum_{i=1}^{n}(x_i - y_i)^2 \tag{3.22}$$

   - Manhattan (równanie 3.22),

$$D(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{3.23}$$

   - Czebyszewa (równanie 3.24).

$$D(x,y) = max_{i=1}^{n}(|x_i - y_i|) \tag{3.24}$$

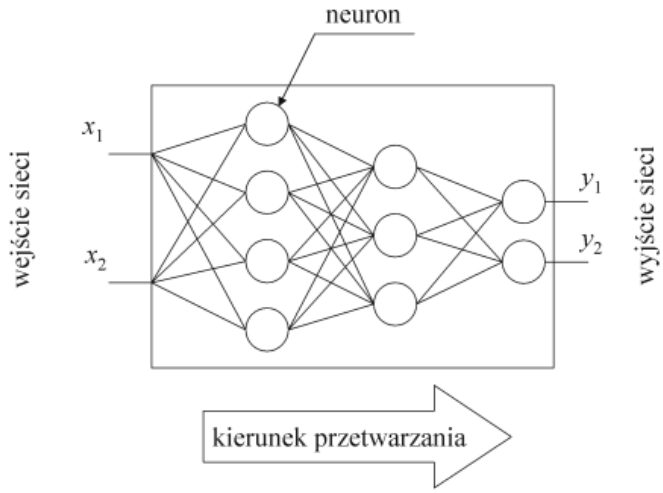
3.4.2.8. Sztuczne sieci neuronowe (SSN)

Sztuczne sieci neuronowe to sposób realizacji nieliniowego aproksymatora, o N wejściach i M wyjściach, który podobnie do biologicznych struktur neuronowych ma zdolność uogólniania informacji, adaptacji i samoorganizacji [37]. Nazwa wywodzi się ze sposobu budowy sieci, która podobnie do mózgu składa się z elementów zwanych neuronami, połączonych ze sobą synapsami. Każda z synaps opisana jest parametrem, zwanym wagą, którego wartość ulega zmianom w procesie uczenia sieci. Neurony organizowane są w grupy, zwane warstwami, które zwykle występują w trzech typach:

- warstwa wejściowa (jedna na sieć),
- warstwy ukryte (opcjonalne, ale może występować nawet kilka),
- warstwa wyjściowa (również jedna na sieć).

Przykładem sztucznej sieci neuronowej, wykorzystywanej często w zagadnieniach rozpoznawania obrazów, jest perceptron wielowarstwowy MLP (*ang. Multi-layer perceptron*) [3][28].

Rysunek 3.11 przedstawia graficzną postać sieci tego typu. Neurony symbolizowane są przez węzły grafu, a synapsy poprzez krawędzie skierowane od jednego węzła do drugiego, co wynika z faktu, że przepływ danych w sieci



Rysunek 3.11: Schemat graficzny sieci typu MLP. Źródło: [3]

odbywa się tylko w jednym kierunku, oznaczonym na rysunku jako kierunek przetwarzania.

Działanie neuronu wymaga wyznaczenia wartości pobudzenia h_i poprzez obliczenie sumy ważonej wejść (równanie 3.25).

$$h_i = \sum_{i=1}^{n} w_{ij} x_i + w_{i0} \tag{3.25}$$

Tak wyznaczona wartość pobudzenia podawana jest jako argument do funkcji aktywacji $y_i = g(h_i)$, którą najczęściej jest funkcja sigmoidalna - monotonicznie rosnąca z asymptotami poziomymi w nieskończonościach np. tangens hiperboliczny (równanie 3.26), przyjmująca wartości w przedziale $[-1; 1]$.

$$g(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.26}$$

Neurony zgrupowane w warstwie nie mają ze sobą żadnych połączeń. Łączą się jedyne z neuronami z warstw sąsiednich w kierunku przepływu danych. W przypadku warstwy wyjściowej, za funkcję aktywacji przyjmuje się funkcję liniową, ponieważ wyjścia neuronów w tej warstwie stanowią jednocześnie wyjście sieci. Ustalenie wag dla poszczególnych synaps jest procesem złożonym i określane jest mianem uczenia sieci neuronowej. W przypadku sieci typu MLP, najczęściej stosowaną metodą jest algorytm wstecznej propagacji błędu, który wykorzystuje metody gradientowe do minimalizacji funkcji błędu średniokwadratowego na zbiorze trenującym.

4. Implementacja wybranych algorytmów

4.1. Wstęp

Celem przeprowadzonych badań było opracowanie algorytmu rozpoznawania tablic rejestracyjnych, który byłby najbardziej odpowiedni do postawionego zadania. Inspiracją były istniejące rozwiązania, opisane w poprzednim rozdziale.

Proces opracowywania algorytmu był procesem iteracyjnym, dlatego też wynikiem badań było opracowanie aż trzech algorytmów. Każdy kolejny, korzystał z obserwacji poczynionych podczas testów poprzedniego. Pozwoliło to na osiągnięcie, w ostatnim algorytmie, kompromisu pomiędzy skutecznością, a szybkością, wymaganego dla urządzeń wbudowanych.

4.2. Algorytm I

Klasyczne podejście do rozpoznawania tablic rejestracyjnych przedstawione w pracach [24], [5], [18], [20], [6] wyróżnia najczęściej trzy podstawowe czynności, tj. lokalizację tablicy na obrazie, wykrycie i wydzielenie znaków z obszaru tablicy oraz ich ostateczne rozpoznanie. Etap pierwszy ma zasadniczo na celu ułatwienie i przyspieszenie późniejszego przetwarzania i rozpoznawania znaków poprzez wskazanie fragmentu obrazu rzeczywiście zawierającego tablicę.

4.2.1. Lokalizacja tablicy rejestracyjnej

Lokalizacja tablicy na obrazie wydaje się być bardzo istotnym elementem całego procesu rozpoznawania - błędne wykrycie obszaru tablicy utrudnia lub wręcz uniemożliwia poprawne rozpoznanie znaków. Jednocześnie ze względu na złożoność obliczeniową wydzielania i rozpoznawania poszczególnych znaków wskazane jest, aby liczba potencjalnych tablic była ograniczona do minimum. Dużym ułatwieniem w procesie projektowania pierwszego algorytmu (zrealizowanego na podstawie [24]) był fakt, że zgodnie z założeniami wstępnymi dla omawianej aplikacji, na obrazie występuje tylko jedna tablica rejestracyjna, a jej pozycja i rozmiary są w przybliżeniu znane. Sam proces lokalizacji tablicy można podzielić na trzy podstawowe fazy.

W pierwszym etapie obraz z kamery (rysunek 4.1a) poddawany jest normalizacji w celu poprawy kontrastu (rysunek 4.1b) oraz odszumieniu za pomocą filtra medianowego, aby pozbyć się niepotrzebnych szczegółów. Tak

przygotowany obraz poddawany jest wykrywaniu krawędzi metodą Cannego. Wynikiem działania algorytmu jest czarno-biały obraz zawierający wykryte krawędzie (rysunek 4.1c).

(a) Obraz oryginalny (b) Po normalizacji histogramu (c) Zlokalizowana tablica

Rysunek 4.1: Kolejne fazy wykrywania tablicy. Źródło: opracowanie własne.

Drugi etap to poszukiwanie poziomego pasa na obrazie, o jak największej gęstości krawędzi. Najprostszą metodą jest zastosowanie dla każdej poziomej linii rzutowania na oś Y, które sprowadza się do zsumowania białych pikseli w danej linii. Otrzymany w ten sposób wektor wartości dla osi Y jest następnie poddawany uśrednieniu w celu eliminacji pojedynczych linii o bardzo dużej wartości, na przykład linii poziomych. W następnym kroku wyznaczana jest wartość maksymalna wektora, a po obu stronach punktu maksimum, poszukiwane są punkty, w których wartość spada poniżej 85% wartości maksymalnej. Punkty te są uznawane za górną i dolną krawędź tablicy rejestracyjnej. Dzięki znajomości przybliżonej wysokości tablicy, możliwe jest odrzucenie pasów za wąskich i za szerokich.

Trzeci etap jest bardzo podobny do drugiego i sprowadza się do wyznaczenia rzutu na oś X dla punktów, które znajdują się wewnątrz pasa wyznaczonego poprzednio. Tak przygotowywany wektor jest uśredniany, a następnie wyznaczany jest punkt o maksymalnej wartości, po bokach którego poszukiwane są punkty o wartości mniejszej od 15% wartości maksymalnej. Współczynnik ten, podobnie jak współczynnik odcięcia dla wartości pionowych, został dobrany eksperymentalnie, jako dający najlepsze rezultaty. Po tych operacjach sprawdzany jest rozmiar i stosunek szerokości do wysokość tablicy. Jeśli obie wartości mieszczą się w określonym zakresie, to tablica nadaje się do dalszej analizy.

4.2.2. Ekstrakcja znaków

Kolejnym etapem identyfikacji tablicy jest wyszukanie w jej obrazie pól, które zawierają znaki do rozpoznania. W celu zmniejszenia złożoności obliczeniowej tego zadania zastosowano detekcję elementów połączonych za pomocą algorytmu rozrostu ziarna (ang. *flood fill*). Pozwala on na wydzielenie z obrazu połączonych obszarów jednego koloru, poczynając od punktu startowego zwanego ziarnem (ang. *seed*). Metoda wyszukiwania ziaren polega, na lokalizacji lokalnych maksimów, w rzucie na oś X (uzyskanego tak jak opisano w sekcji 4.2.1), a następnie dla każdego z takich maksimów, poszukiwaniu

czarnych punktów w górę i dół od środka linii. Punkt taki staje się następnie ziarnem, a region otaczający tak zapełniony element staje się kandydatem do rozpoznania. Ponownie, prostą metodą eliminacji kandydatów, jest sprawdzenie stosunku wysokości do szerokości oraz samej wysokości kandydata, która musi odpowiadać wysokości obserwowanej tablicy.

4.2.3. Rozpoznanie znaków

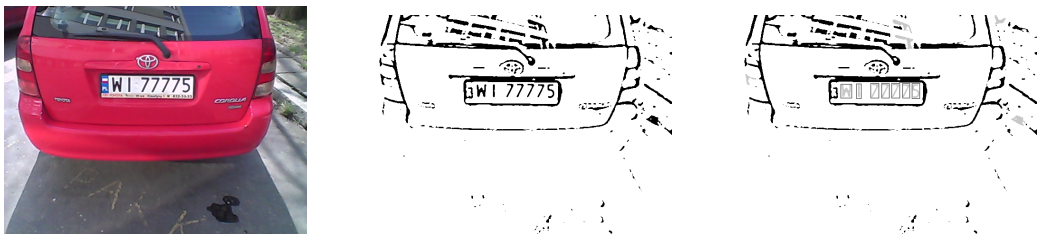
Algorytm rozpoznania znaków jest najprostszą częścią algorytmu. Każdy element połączony z poprzedniego etapu (obraz binarny) skalowany jest do wysokości 80 pikseli i umieszczany po środku białego prostokąta o rozmiarze 80x60 pikseli, a następnie skalowany do rozdzielczości 9x9 pikseli (w skali szarości tym razem), w celu uzyskanie prostej mapy gęstości pikseli [10] reprezentujących dany znak. Mapa ta jest następnie porównywana z każdym ze wzorców za pomocą znormalizowanej korelacji zaimplementowanej w funkcji cvTemplateMatch() z biblioteki OpenCV. Wykorzystywane wzorce, wykonane na podstawie rozporządzenia [26], zapisane zostały jako obrazy binarne o rozdzielczości 80x60 pikseli. Po przeskalowaniu, z obrazów tych powstały mapy gęstości o rozmiarze 9x9, w odcieniach szarości, które mogą być łatwo porównywane ze znakiem, dając w wyniku jedną wartość oznaczającą współczynnik korelacji obu obrazów. Jako najbardziej odpowiadający zostaje wybrany ten wzór, dla którego współczynnik korelacji jest największy. W celu zmniejszenie wpływu położenia tablicy względem kamery, każdy ze znaków wzorcowych został przygotowany w kilku wersjach uwzględniających różne kąty obserwacji tablicy.

4.3. Algorytm II

Metoda lokalizacji tablicy rejestracyjnej w obrazie, oparta na wykrywaniu krawędzi pozwala na uzyskanie skuteczności na poziomie 90%. Próba jej zwiększenia prowadzi często do wykrycia wielu fałszywych tablic, a to z kolei powoduje konieczność wykonywania segmentacji i rozpoznawania znaków dla każdego kandydata na tablicę. W skrajnym przypadku może to oznaczać, że dla jednego obrazu wykonane zostanie nawet 20-krotne rozpoznanie, co znacznie wydłuża czas przetwarzania.

4.3.1. Lokalizacja tablicy i ekstrakcja znaków

Bardzo dobre rezultaty dało zastosowanie połączenia progowania adaptacyjnego oraz algorytmu rozrostu ziarna. Pozwoliło to wyszukać od razu, w jednym kroku, kandydatów na znaki na całym obrazie, a następnie na odsianie tych, które nie spełniają podstawowych ograniczeń stawianych znakom (określone pole powierzchni, proporcje rozmiarów, wysokość i szerokość). Ostatnim krokiem było wyszukanie tych znaków, które znajdowały się w jednej, poziomej linii , a odległość pomiędzy nimi była niewielka. Takie grupowanie pozwoliło wyznaczyć zestaw znaków, które miały zostać rozpoznane jako numer tablicy. Na rysunku 4.2 zilustrowano kolejne etapy procesu selekcji kandydatów.



(a) Obraz oryginalny (b) Progowanie adaptacyjne (c) Usuwanie elementów

Rysunek 4.2: Wykrywanie znaków tablicy rejestracyjnej w oparciu o progowanie adaptacyjne i algorytm rozrostu ziarna - po usunięciu elementów niespełniających eksperymentalnie ustalonych kryteriów, do dalszej analizy pozostają tylko znaki zaznaczone ramkami. Źródło: opracowanie własne.

Opisany powyżej algorytm charakteryzował się zaskakująco krótkim czasem wykonania, pozwalało to na wielokrotne wykonanie progowania adaptacyjnego z różnymi wartościami parametrów i zwiększało szanse na poprawne wydzielenie tablicy w przypadkach szczególnych, co pokazano na rysunku 4.3.

(a) Błędnie wykryta tablica (b) Pierwsze progowanie (c) Drugie progowanie

Rysunek 4.3: Porównanie wykrywania krawędzi i progowania adaptacyjnego - poprawne wydzielenie wszystkich znaków z tablicy nastąpiło dla jednego z kilku parametrów progowania adaptacyjnego, metoda krawędziowa zawiodła. Źródło: opracowanie własne.

4.3.2. Rozpoznanie znaków

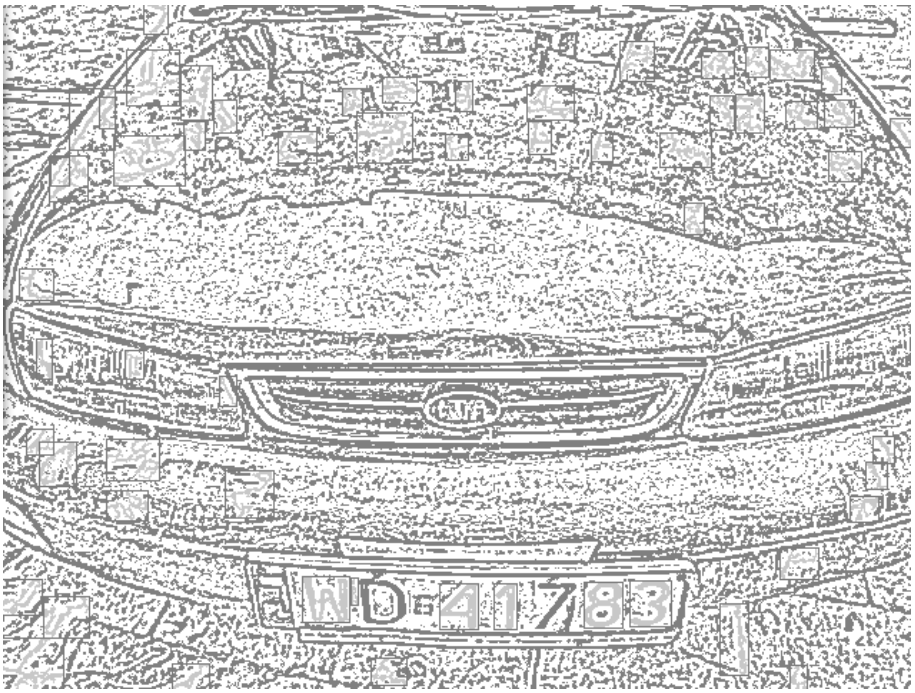
Zastosowany algorytm rozpoznawania znaków był identyczny jak w przypadku pierwszego algorytmu.

4.4. Algorytm III

Pomimo dobrej szybkości i skuteczności, algorytm II nie radził sobie z pewną klasą problemów. Zwłaszcza, w przypadku, gdy choćby jeden ze znaków został pominięty np. poprzez złączenie z obramowaniem tablicy, następowało pominięcie tablicy rejestracyjnej, poprzez odrzucenie na poziomie walidacji. Dodatkowym problemem były np. śruby występujące często na

tablicach pomiędzy znakami, które w trakcie progowania łączyły ze sobą znaki, co powodowało ich odrzucenie. Bardzo dobrze ilustruje to rysunek 4.4, na którym pominięte zostały aż dwa znaki.

(a) Obraz z kamery



(b) Wyszukane segmenty - znaki D i 7 zostały pominięte

Rysunek 4.4: Pominięcie znaków w wyniku scalenia znaków z obramowaniem tablicy. Źródło: opracowanie własne.

Powyższe błędy wynikają z faktu połączenia procesu lokalizacji i segmentacji w jeden etap. Przeprowadzone testy wykazały, że zadania te wyma-

gają nieco innych parametrów, ponieważ na etapie lokalizacji konieczne było połączenie znaków stanowiących tablicę w jeden obszar. Celem segmentacji było wtedy jak najdokładniejsze wydzielenie obszaru poszczególnych znaków i usunięcie istniejących zakłóceń np. śrub i zabrudzeń. Wynikiem tych spostrzeżeń było opracowanie algorytmu III, będącego bezpośrednią modyfikacją algorytmu II.

4.4.1. Lokalizacja tablicy rejestracyjnej

Lokalizacja tablicy jest przeprowadzana w sposób bardzo zbliżony do poprzedniego algorytmu. Różnice mają na celu jedynie zmniejszenie selektywności algorytmu, ponieważ wykryte segmenty nie są poddawane rozpoznawaniu, a tylko służą do wykrycia obszaru tablicy będącego najmniejszym prostokątem otaczającym (*ang. MBR - Minimum Bounding Rectangle*) dla grupy znaków.

Wykryte segmenty są więc poddawane łączeniu w grupy o bardzo zbliżonych wartościach współrzędnej Y (maksymalne odchylenie o 50% wysokości znaku) i znajdujących się w odległości maksymalnie równej trzem szerokościom znaku. Grupowanie takie pozwala na poprawną lokalizację tablicy, nawet gdy dwa znaki zostaną pominięte np. w przypadku połączenia przez śrubę. Następnie, jeśli grupa składała się minimalnie z czterech znaków, to wyznaczany był najmniejszy prostokąt otaczający, który był uznawany za obszar tablicy rejestracyjnej (rysunek 4.5a).

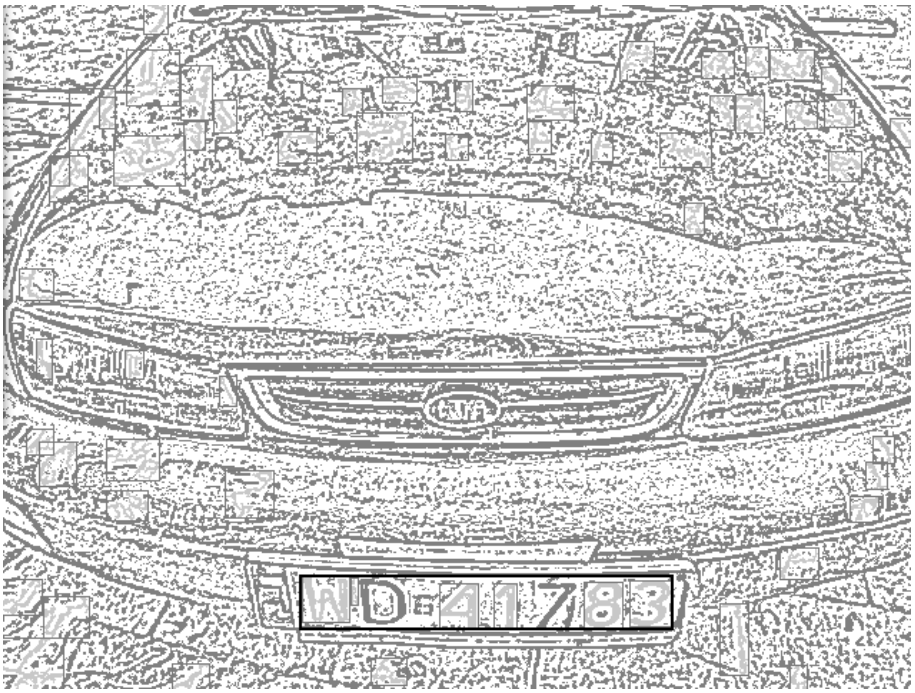
4.4.2. Segmentacja znaków

Etap segmentacji wykonywany był w sposób analogiczny do poprzedniego, z tą różnicą, że progowanie i rozrost ziarna przeprowadzane były tylko na obszarze tablicy, gdzie dużo łatwiej było wyznaczyć parametry progowania adaptacyjnego oraz wielkość znaków, które mają być wynikiem segmentacji. Dużo większa selektywność umożliwiła usunięcie zakłóceń, dzięki temu znaki pominięte na etapie lokalizacji mogą być poprawnie wydzielone (rysunek 4.5b).

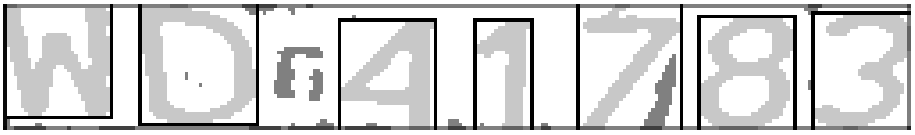
4.4.3. Rozpoznawanie znaków

Bardzo dobre wyniki algorytmów lokalizacji i segmentacji sprawiły, że głównym źródłem zawartych w nich błędów stała się niedokładność algorytmu OCR, bazującego na wyznaczaniu korelacji map gęstości wzorców i znaku, który pozostał niezmieniony względem algorytmu I. Podstawowym problemem tego algorytmu było podobieństwo znaków np. O i 0 (zero), 5 i S oraz Y i V. Podczas prezentacji wstępnych wyników badań na XVII Konferencji „Systemy Czasu Rzeczywistego" w Gdańsku, zaproponowane zostało użycie Sztucznych Sieci Neuronowych, jako klasyfikatora znaków. Metoda ta była więc oczywistym wyborem jako alternatywa dla stosowanego algorytmu.

Wybrana została bardzo popularna, w zastosowaniach rozpoznawania znaków, sieć typu MLP (opisana w rozdziale 3.4.2.8). Dzięki istniejącej implementacji tej sieci w bibliotece OpenCV, możliwe było szybkie stworzenie i przetestowanie nowego algorytmu OCR.



(a) Najmniejszy prostokąt otaczający tablicy



(b) Obszar tablicy z wyznaczonymi segmentami

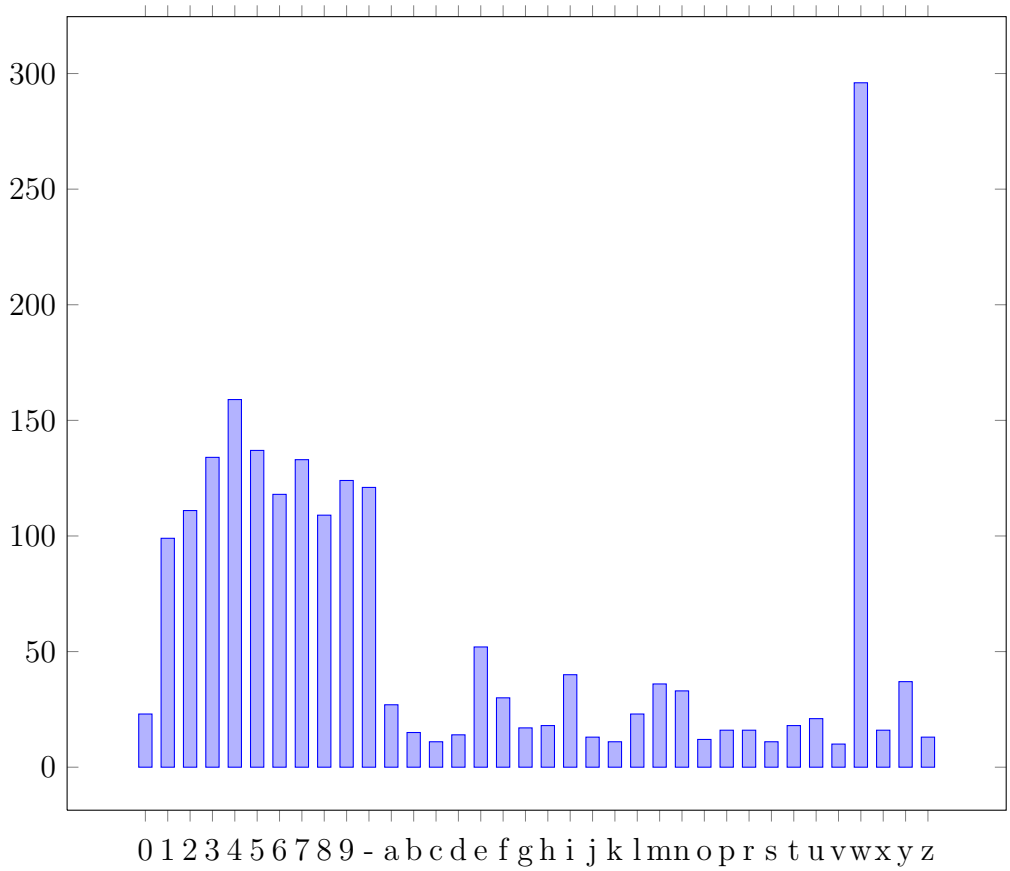
Rysunek 4.5: Lokalizacja tablicy i segmentacja znaków z wykorzystaniem algorytmu III, z wydzielonym etapem segmentacji. Źródło: opracowanie własne.

Jako dane treningowe wykorzystano 2140 znaków wyciętych ze zdjęć tablic rejestracyjnych. Rysunek 4.6 przedstawia wykres rozkładu ilości próbek dla poszczególnych znaków. Widoczna przewaga próbek odpowiadających cyfrom oraz literze „W", wynika z faktu, że zdjęcia tablic wykonane zostały na terenie województwa mazowieckiego, którego wyznacznikiem jest właśnie litera „W".

Przykładowe znaki, wycięte z tablic rejestracyjnych prezentuje rysunek 4.7.

Każdy z nich był poddany skalowaniu do rozmiarów 7x8, a następnie drukowany w postaci pojedynczej linii zawierającej następujące po sobie: etykietę (znak) i kolejne wartości z wektora gęstości. Przykładowa linia wzorca dla litery Y:

```
Y,255,255,255,255,255,255,255,255,30,110,228,255,255,255,255,
255,166,42,28,151,190,190,190,215,255,230,25,4,11,11,11,54,
120,29,59,180,237,237,237,217,43,139,240,255,255,255,255,255,
255,255,255,255,255,255,255,255
```



Rysunek 4.6: Liczba wystąpień poszczególnych znaków w zbiorze uczącym. Źródło: opracowanie własne.

Cały proces został zautomatyzowany dzięki wykorzystaniu skryptu w języku Python, wykorzystującego bibliotekę OpenCV:

```
#!/usr/bin/python
import cv
import os
import sys
import glob

def print_file(filename):
    im = cv.LoadImage(filename, 0)
    #skalowanie wzorca do mapy gęstości 7x8
    dst = cv.CreateImage((7,8), cv.IPL_DEPTH_8U, 1)
    cv.Resize(im, dst, cv.CV_INTER_AREA)
    #etykieta jest pierwszym znakiem nazwy
    #po nazwie katalogu
    sys.stdout.write( filename[8] )
    #drukowanie wartości gęstości
    for i in range( 0, dst.height ):
        for j in range( 0, dst.width):
            sys.stdout.write( ",")
```

Rysunek 4.7: Przykładowe obrazy znaków (po jednym dla każdej klasy znaków), wycięte ze zdjęć tablic rejestracyjnych. Źródło: opracowanie własne.

```
            sys.stdout.write(str(int(dst[i,j])) )
    sys.stdout.write("\n")

if __name__ == "__main__":
    path = 'mlptest/'
    for infile in glob.glob( os.path.join(path, '*.png') ):
        print_file(infile)
```

Zebrane w ten sposób dane zostały podzielone na dwie grupy:

- dane treningowe (80%),
- dane testowe (20%).

Pozwoliło to na natychmiastową weryfikację wpływu struktury sieci i parametrów ją opisujących, na skuteczność wytrenowania sieci.

Wykorzystana sieć miała 56 neuronów w warstwie wejściowej (7 * 8 wartości w mapie gęstości) oraz 36 wyjść w warstwie wyjściowej (25 liter A-Z bez Q + 10 cyfr + znak „_”, odpowiadający błędnie wykrytym segmentom np. fladze występującej na początku tablicy). Dodatkowo sieć zawierała od

jednej do trzech warstw ukrytych o liczbie neuronów wahającej się do 50 do 100. W celu wybrania najlepszej, przeprowadzono serię treningów sieci o różnej strukturze i różnych parametrach. Wyniki badań przedstawiono w tabeli 4.1.

Parametry	**dw=0,001;moment=0**		**dw=0,01;moment=0,1**	
Zbiór	**train**	**test**	**train**	**test**
Warstwy ukryte				
1 x 50n	98,7%	98,1%	99,6%	98,6%
1 x 100n	98,8%	97,7%	99,6%	98,6%
2 x 50n	-	-	99,7%	98,4%
2 x 60n	98,8%	98,1%	99,7%	98,8%
2 x 70n	**-**	**-**	**99,8%**	**98,8%**
2 x 100n	98,9%	98,1%	99,6%	98,6%
3 x 50n	98,0%	98,1%	99,5%	98,6%
3 x 80n	98,0%	98,1%	99,6%	98,1%

Tablica 4.1: Odsetek prawidłowych rozpoznań dla zbioru uczącego (train - 1712 znaków) i testowego (test - 428 znaków), dla różnych parametrów i struktur sieci neuronowej. Źródło: opracowanie własne.

Do obliczania gradientu zastosowano algorytm propagacji wstecznej, a za funkcję aktywacji posłużyła symetryczna funkcja sygmoidalna, o współczynnikach $\alpha = 1$ i $\beta = 1$ opisywana równaniem 4.1.

$$f(x) = \beta \cdot \frac{1 - e^{-\alpha x}}{1 + e^{-\alpha x}} \tag{4.1}$$

Z przeprowadzonych badań wynika, że najlepsze rezultaty otrzymano dla sieci z dwoma warstwami ukrytymi, każda po 70 neuronów. Dodatkowo, w przypadku zmiany domyślnych ustawień propagacji wstecznej (bp_dw_scale = 0,001 i bp_moment_scale = 0) na bp_dw_scale = 0,01 i bp_moment_scale = 0,1, nastąpił zauważalny wzrost wytrenowania sieci zarówno dla danych treningowych jak i testowych (od 0,5% do 1,5%).

Wynikiem procesu uczenia sieci, w przypadku biblioteki OpenCV, był plik XML, który opisywał strukturę wytrenowanej sieci i pozwalał na łatwe wykorzystywanie jej do rozpoznawania znaków, bez konieczności ponownego uczenia.

5. Badania algorytmów na wybranych platformach wbudowanych

5.1. Platformy sprzętowe wybrane do testów

Do testów porównawczych, wybrane zostały trzy popularne obecnie platformy sprzętowe, zdolne do pracy w środowisku Linux. Dodatkowym warunkiem był jak najmniejszy pobór mocy (maksymalnie 15 W). Wybór padł na urządzenia oparte o procesory Intel Atom N270 , AT91SAM9260 (ARM9) i OMAP3530 (ARM Cortex-A8). Każdy z tych procesorów ma swoją domenę zastosowań, jednak wszystkie nadają się do przetwarzania obrazów ze względu na wydajność oraz możliwości: podłączenia kamery przez port USB i przenoszenia bibliotek oraz oprogramowania z systemu Linux. Każdy z procesorów potrzebował do pracy zestawu układów peryferyjnych, takich jak pamięć ROM i RAM, porty USB, RS232 i Ethernet. Najlepszym rozwiązaniem okazało się zastosowanie gotowych urządzeń dostarczanych w formie tak zwanych układów ewaluacyjnych. W niniejszym rozdziale opisane zostaną układy MMnet1002 (ARM926EJ), Beagleboard (OMAP3530) oraz netbook Samsung N130, jako przykład komputera opartego o procesor Atom N270. W tabeli 5.1 przedstawiono zestawienie podstawowych parametrów tych układów.

Nazwa	MMnet1002	Beagleboard	N130
Procesor	AT91SAM9260	OMAP3530	Atom N270
Producent	Atmel	Texas Instruments	Intel
Architektura	ARM926EJ-S	ARM Cortex-A8	x86
Taktowanie	210MHz	600MHz	1600MHz
Pamięć RAM	64MB	256MB	1024MB
Pamięć Flash	1GB	256MB	dysk twardy
Linux	OpenWrt	Ångström	Ubuntu
Moc	1.5W	2W	15W
Moc procesora	<0.5W	<1.5W	2.5W
Cena detaliczna	250 PLN	530 PLN	1000 PLN

Tablica 5.1: Zestawienie platform testowych. Źródło: opracowanie własne.

5.1.1. MMnet1002

MMnet1002 (rysunek 5.1), to moduł Ethernet z systemem Linux, wyprodukowany przez polską firmę Propox. To doskonała platforma do budowania

urządzeń, które mają być podłączone do sieci komputerowej. Dzięki bogatemu zestawowi portów i wyprowadzeń GPIO (ang. *General Purpose Input Output*), możliwe jest wykorzystanie go w bardzo szerokim spektrum zastosowań: od przyrządów pomiarowych po układy przetwarzanie sygnałów. Wraz z modułem producent dostarcza system OpenWrt, będący wersją systemu Linux przygotowaną z myślą o routerach oraz SDK (ang. *Software Development Kit*), czyli zestaw oprogramowania potrzebny do tworzenia własnych programów na ten system, wraz z obszerną dokumentacją.

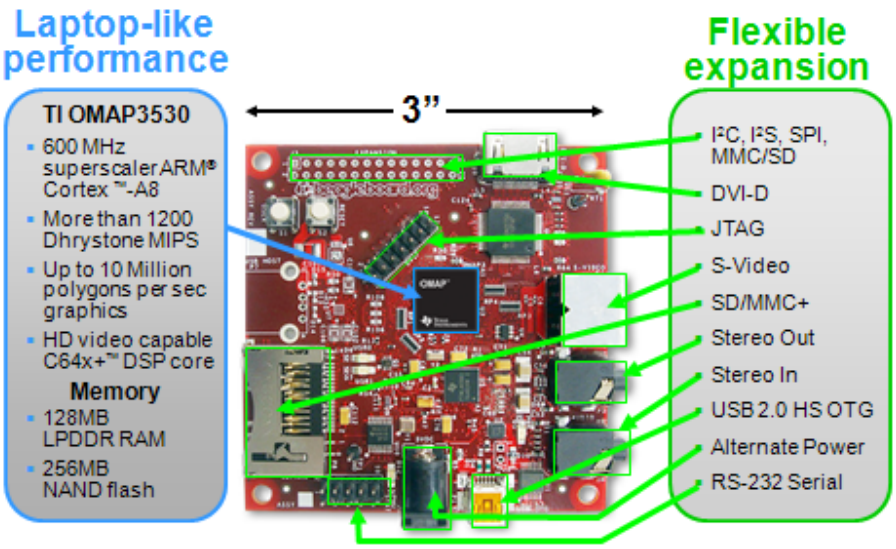
Rysunek 5.1: MMNet1002 - Minimoduł Ethernetowy z procesorem ARM9 400MHz oraz systemem Linux. Źródło: http://www.propox.com/products/t_232.html

5.1.2. Beagleboard

Beagleboard (rysunek 5.2), to zintegrowany komputer jednopłytkowy SBC (ang. *Single Board Computer*), oparty na nowoczesnym procesorze OMAP3530 firmy Texas Instruments, którego podstawowe cechy to:

- superskalarny rdzeń ARM Cortex-A8 taktowany zegarem 600MHz,
- zintegrowany rdzeń DSP TMS320C64x+, zdolny do odtwarzania video w jakości HD;
- oraz akcelerator 2D/3D zgodny z OpenGL ES 2.0.

Dodatkowo urządzenie to oferuje pełen zestaw portów, jakich oczekuje się od komputera, na czele z wyjściem DVI-D, do podłączenia monitorów i telewizorów pracujących w rozdzielczościach HD. Największą zaletą urządzenia jest jednak ogromna społeczność użytkowników, którzy wykorzystują Beagleboard z różnymi systemami (Linux, Windows Embedded, Android). Ponadto bogactwo dokumentacji i przykładowych projektów gwarantuje, że użytkownik nie zostanie pozostawiony z problemami sam.



Rysunek 5.2: Beagleboard. Źródło: http://beagleboard.org/hardware

5.1.3. Samsung N130

Samsung N130 (rysunek 5.3) to typowy przedstawiciel popularnych obecnie netbooków, czyli komputerów przenośnych o przekątnej ekranu nie przekraczającej 10 cali. Pod względem technicznym maszyny tej klasy niewiele się od siebie różnią. Większość netbooków, to maszyny oparte o procesor Intel Atom N270, 1GB pamięci RAM i chipset Intel 945GSE. Zapotrzebowanie na moc dla tych urządzeń waha się w przedziale od 10 do 20 watów, zależnie od zastosowanego profilu zarządzania energią. N130 został wybrany jako przedstawiciel komputerów z rodziny x86 o bardzo małym poborze mocy. Po wyłączeniu ekranu, stanowi on substytut SBC, czyli niewielkiego komputera, którego wszystkie komponenty są zintegrowane na płycie głównej.

5.2. Opis badań

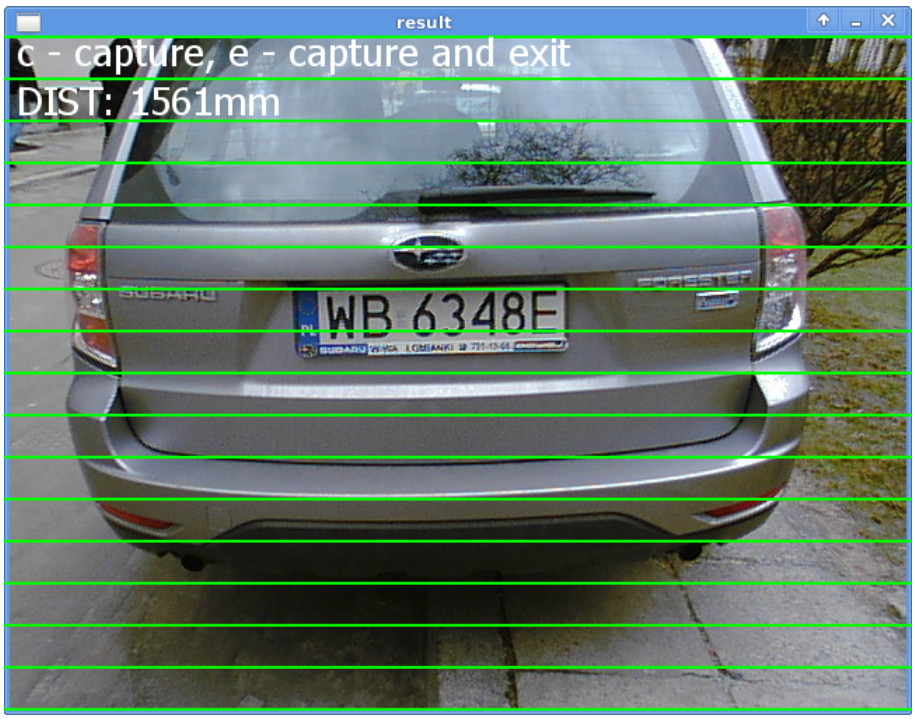
5.2.1. Dane testowe

W celu uzyskania wiarygodnych wyników podczas testów skuteczności i wydajności wszystkich algorytmów, konieczne było zgromadzenie reprezentatywnej bazy zdjęć testowych. Oznaczało to, że zdjęcia te musiały spełniać założenia dotyczące umiejscowienia kamery i pomiaru odległości, która stanowi jeden z parametrów algorytmu. Wykonano 250 zdjęć z wykorzystaniem kamery i czujnika odległości, opisanych w rozdziale 6.2. W celu zapewnienia odpowiedniego kadrowania, poziomowania i zakresu odległości, konieczne okazało się ręczne wyzwalanie zdjęć. Oznaczało to konieczność napisania prostego programu wykorzystującego bibliotekę OpenCV i uruchomienie go na opisywanym w rozdziale 5.1.3 netbook-u N130.

Rysunek 5.4 przedstawia okno podglądu podczas wykonywania zdjęcia tablicy rejestracyjnej. Rozmieszczone co 30 pikseli linie poziome pozwalają na

Rysunek 5.3: Samsung N130. Źródło: http://www.videotesty.pl/m/pliki/2009/12/07/Samsung-NP-N130-KA01PL_1_.jpeg

szybkie ustawienie kamery w pozycji poziomej, a odświeżana co pół sekundy odległość, odczytana z czujnika pozwala ustawić kamerę w odległości 1-2m (zgodnie z założeniami algorytmów).



Rysunek 5.4: Okno podglądu aplikacji do robienia zdjęć testowych z wykorzystaniem kamery PS Eye i czujnika odległości. Źródło: opracowanie własne.

5.2.2. Metodyka testów na wybranych platformach

Na potrzeby badań, wersje algorytmu opisane w podrozdziałach 4.2, 4.3 i 4.4 zostały zaimplementowane w języku C++, z wykorzystaniem biblioteki OpenCV [8]. Biblioteka ta dostarczyła implementacji większości operacji przeprowadzanych na obrazie i stanowiła doskonałe rozwiązanie ułatwiające tworzenie prototypu programu. Dzięki dobrej przenośności można ją wykorzystywać zarówno w środowisku Windows, jak i we wszystkich odmianach systemów Linux, także w wersjach na systemy wbudowane. Pozwoliło to na wykorzystanie tego samego kodu źródłowego do stworzenia trzech wersji programu.

Prace nad przeniesieniem kodu na każdą z platform polegały na rekompilowaniu biblioteki OpenCV oraz wykorzystywanych przez nią bibliotek na maszynę docelową, przy użyciu zestawu narzędzi (ang. *toolchain*), przeznaczonych dla danej architektury, a następnie kompilacji programu z wykorzystaniem tych bibliotek. Proces ten został zautomatyzowany dzięki zastosowaniu skryptów powłoki, które wymagają jedynie ustawienia katalogu zawierającego toolchain (zmienna DEVROOT) oraz prefiksu oznaczającego architekturę. Skrypty te inspirowane były notatką z bloga [35].

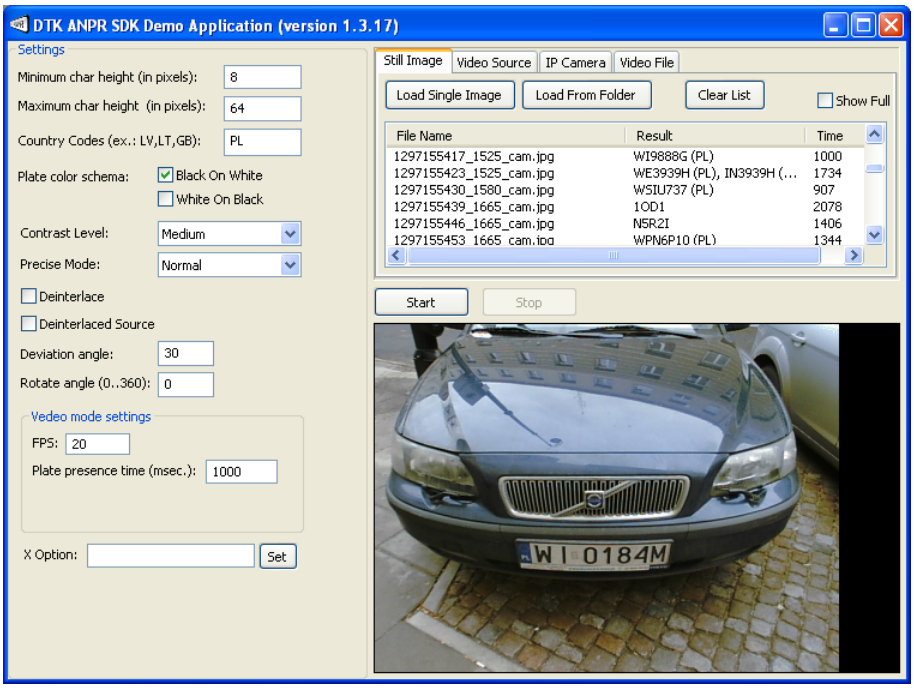
Dodatkowo, dla porównania, przeprowadzono test z wykorzystaniem tych samych zdjęć, na demonstracyjnej aplikacji „DTK ANRP SDK Demo Application" wykorzystującej komercyjną bibliotekę opisaną w rozdziale 2.3. Niestety, ze względu na brak dostępu do kodu źródłowego aplikacji, test przeprowadzony został jedynie z wykorzystaniem netbooka N130, pracującego pod kontrolą systemu Windows XP. Wykorzystany został tryb rozpoznawania obrazów nieruchomych (ang. *Still Image*). Maksymalny czas rozpoznawania i ilość błędnie rozpoznanych tablic, zostały odczytane z listy wyników (rysunek 5.5).

5.2.3. Wyniki badań

Program testowy skompilowano i uruchomiono na wszystkich omówionych platformach sprzętowych. Testy wykorzystywały zawsze ten sam zbiór testowy, składający się z 250 obrazów o rozdzielczości 640x480 pikseli. W celu ułatwienia testów obrazy wejściowe zostały zapisane w plikach, o nazwach w formacie:

```
NRTABLICY_ODLEGLOSC.jpg
```

gdzie:
NRTABLICY - numer rejestracyjny tablicy na obrazie np WW12345
ODLEGLOSC - wyrażona w milimetrach odległość pomiędzy tablicą, a kamerą (czujnikiem) np. 1234.
Pozwoliło to na wykorzystanie skryptu automatyzującego test i zliczającego liczbę błędnych rozpoznań. Ponieważ na wszystkich platformach uruchamiano dokładnie ten sam kod źródłowy, uzyskiwane wyniki były identyczne pod względem jakościowym. Znacząca była za to różnica w czasie wykonania na poszczególnych platformach, co pokazano w Tabeli 5.2.



Rysunek 5.5: Okno aplikacji DTK ANRP SDK Demo Application w trakcie testu. Źródło: opracowanie własne.

Algorytm	I	II	III	DTK
Liczba błędów	23	15	5	32
Skuteczność [%]	91%	94%	98%	87%
Czas maksymalny [s]				
MMnet1002	16.2	13.1	14.5	-
Beagleboard	2.8	2.4	2.9	-
N130	1.7	1.5	1.7	2.1

Tablica 5.2: Zestawienie czasów i skuteczności opisanych algorytmów na poszczególnych platformach sprzętowych. Źródło: opracowanie własne.

5.2.4. Wybór algorytmu

Wyniki badań jednoznacznie pokazują, że Algorytm III lepiej nadaje się do zastosowania w wybranej klasie problemów. Jego dokładność jest zdecydowanie największa, jednocześnie czas przetwarzania jest nieznacznie tylko gorszy niż w przypadku Algorytmu II. Algorytmy II i III nie mają problemów wynikających z podziału tablicy na mniejsze fragmenty, które stanowiły podstawowy problem Algorytmu I, ponieważ fragmenty takie są łączone podczas grupowania elementów. Dodatkowo wykorzystanie sieci neuronowej jako klasyfikatora w module OCR powiększyło jeszcze bardziej przewagę Algorytmu III.

Zaskoczeniem okazał się wynik uzyskany przez bibliotekę DTK, która dla zbioru testowych zdjęć, miała najgorszą skuteczność z pośród badanych algorytmów. Szczególnie źle radziła sobie ze zdjęciami, na których litery po-

łączone są z krawędzią tablicy lub występują elementy łączące znaki (rysunek 5.6). Znaczna część tablic sprawiających problemy bibliotece DTK, została poprawnie rozpoznana przez Algorytm III, co w znaczym stopniu jest zasługą wykorzystania dwupoziomowego progowania oraz segmentacji, które dobrze radzą sobie z pominięciem pojedynczych znaków przy lokalizacji tablicy.

Ponadto, aplikacja demonstracyjna okazała się wolniejsza od wszystkich opracowanych algorytmów, co w pewnym stopniu może wynikać z konieczności obsługi graficznego interfejsu użytkownika, niemniej wskazuje na dobrą wydajność opracowanych algorytmów.

Rysunek 5.6: Przykłady błędnych rozpoznań aplikacji DTK ANRP SDK Demo Application. Każdy z przykładów składa się z wiersza wynikowego oraz obrazu tablicy. Źródło: opracowanie własne.

5.2.5. Analiza przypadków błędnego działania wybranego algorytmu

Wybrany algorytm bardzo dobrze radził sobie w większości przypadków, ale warto omówić klasy błędów jakie wystąpiły podczas testów. Analiza błędnie rozpoznanych tablic wykazała, że istniały dwie główne przyczyny błędów:

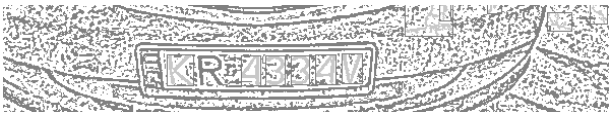
- błędy modułu OCR,

- błędy lokalizacji i segmentacji.

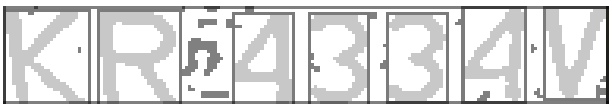
Pierwszy typ błędu ilustruje rysunek 5.7. Pomimo poprawnej lokalizacji i segmentacji, wynik otrzymany podczas walidacji prezentuje się następująco:

```
20:55:15,073 INFO - Walidacja tablicy [KR43347]
20:55:15,075 INFO - Tablica poprawna: TAK
20:55:15,075 INFO - Typ tablicy:
Zwyczajna - 2 znakowy wyznacznik powiatu
20:55:15,075 INFO - Wyznacznik: KR
20:55:15,075 INFO - Info:
Województwo: małopolskie
Powiat: Kraków
```

(a) Obraz oryginalny



(b) Lokalizacja tablicy



(c) Segmentacja

Rysunek 5.7: Błąd na etapie rozpoznawania znaków. Źródło: opracowanie własne.

Znak *V* został rozpoznany jako *7*. Wynika to z błędu sieci neuronowej, który mogłoby zostać usunięty poprzez trening sieci neuronowej z większą liczbą znaków. Błąd tego typu jest szczególnie kłopotliwy, ponieważ tablica przeszła poprawnie proces walidacji.

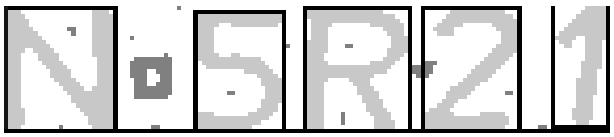
Drugi typ błędu został zilustrowany na rysunku 5.8. W tym przypadku, błąd występuje już na etapie lokalizacji tablicy. Wynika on z faktu, że znaki *W* i *P* zlały się z obramowaniem tablicy. Tablica taka została jednak odrzucona na etapie walidacji:

```
20:53:49,985 INFO - Walidacja tablicy [N5R21]
20:53:49,988 INFO - Tablica poprawna: NIE
20:53:49,988 INFO - Typ tablicy: Indywidualna
20:53:49,988 INFO - Wyznacznik: N5
20:53:49,988 INFO - Info:
Błędny wyznacznik numeru rejestracyjnego!
```

Problemy tego typu wynikają ze sposobu działania algorytmów lokalizacji i segmentacji, a ich eliminacja jest niezwykle trudna. Jednym z rozwiązań,

(a) Obraz oryginalny

(b) Lokalizacja tablicy



(c) Segmentacja

Rysunek 5.8: Błąd lokalizacji tablicy, wynikający ze złączenia znaków W i P z obramowaniem tablicy. Źródło: opracowanie własne.

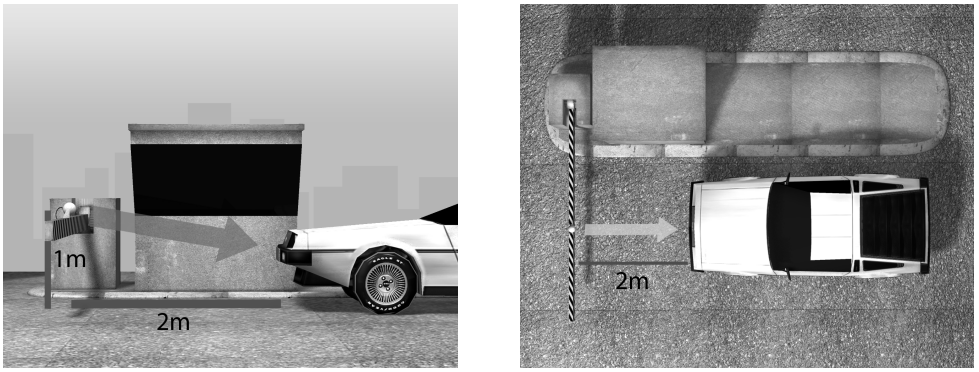
jest umieszczenie kamery, jak najbardziej, na wprost tablicy rejestracyjnej, aby uniknąć zasłaniania znaków przez krawędzie tablicy.

Rozdział 6 przedstawia przykładowe zastosowanie wybranego algorytmu, w automatycznym systemie obsługi parkingu.

6. Realizacja systemu parkingowego na platformie Beagleboard

6.1. Wstęp

Systemy ALPR wykorzystywane na parkingach, w przeciwieństwie np. do fotoradarów, charakteryzują się relatywnie stałymi, a w każdym razie dającymi się przewidzieć, warunkami pracy. Znacznie ułatwia to projektowanie algorytmu i umożliwia implementację prostszych, a zatem szybszych rozwiązań, nadających się dla licznych platform wbudowanych. Umieszczenie kamery centralnie przed pojazdem, nieco powyżej poziomu reflektorów (rysunek 6.1a), powoduje, że zniekształcenia wynikające z wykonywania zdjęć pod kątem są niewielkie, a efekt oślepienia kamery nie występuje. Dużym ułatwieniem jest także fakt, że zdjęcia robione są pojazdom zatrzymującym się w znanej i w przybliżeniu niezmiennej odległości od kamery (przed szlabanem - rysunek 6.1b), co pozwala zachować stały rozmiar tablicy na zdjęciach, upraszczając algorytm lokalizacji.



(a) Widok z boku (b) Widok z góry

Rysunek 6.1: Umiejscowienie kamery na szlabanie wjazdowym. Źródło: opracowanie własne.

Zdjęcia mogą być wykonywane przy użyciu zwykłej kamery internetowej, w rozdzielczości 640x480. Zastosowanie dodatkowych źródeł światła, pozwala na eliminację cieni utrudniających poprawne rozpoznanie znaków na tablicy. Doskonale sprawdza się w tym przypadku światło podczerwone, które nie jest widoczne dla kierowcy, a jest rejestrowane przez stosowane w kamerach matryce CMOS. Ze względu na funkcję jaką ma pełnić omawiany system (na przykład otwieranie szlabanu, wydawanie biletu), konieczne jest aby działał

on w czasie rzeczywistym, to znaczy, aby jego czas działania był akceptowalny dla kierowców. W praktyce oznacza to maksymalny czas oczekiwania nie przekraczający kilku sekund. Ograniczone możliwości sprzętu, na który system ma być uruchomiony sprawiały, że konieczne było zastosowanie prostego i szybkiego algorytmu lokalizacji i rozpoznawania tablic, pozwalającego jednocześnie na osiągnięcie wysokiej skuteczności identyfikacji. Warunki te doskonale spełniał algorytm III opisany w poprzednim rozdziale.

6.2. Zastosowany sprzęt

6.2.1. Beagleboard

Podstawowa specyfikacja tego urządzenia została już podana podczas opisywania platform sprzętowych użytych do testów. Sprzęt ten doskonale sprawdził się, podczas testów wydajnościowych, mieszcząc się w założonych kryteriach czasowych, a ponadto, dzięki bogactwu dokumentacji i łatwości wykorzystania USB, stanowi doskonały element sterujący do złożonych systemów pomiarowych lub automatyki przemysłowej.

6.2.1.1. Przygotowanie urządzenia do pracy

W trakcie prowadzonych badań, poza samą płytą Beagleboard, wykorzystane zostały także następujące akcesoria, które ułatwiały jej używanie.

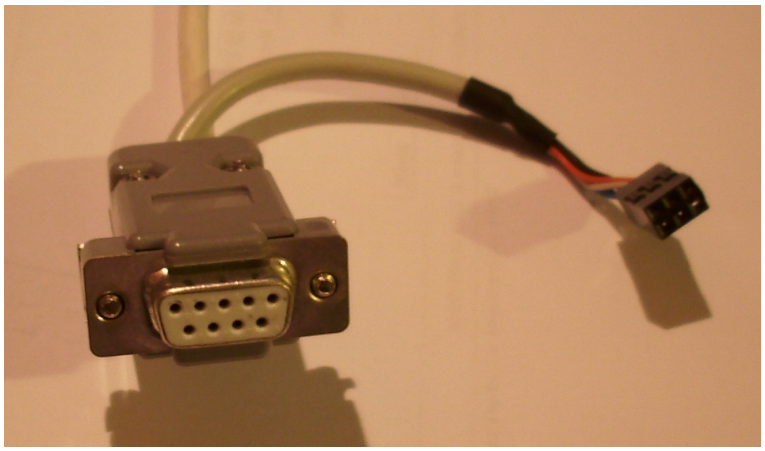
Zasilacz stabilizowany 5V/1A Urządzenie może być zasilane na dwa sposoby:

- przewodem mini-USB (z komputera),
- zasilaczem stabilizowanym.

Wykorzystanie magistrali USB jako źródła zasilania jest rozwiązaniem łatwiejszym (nie ma konieczności dokupowania kolejnych urządzeń), ale podczas testów okazało się, że prąd dostarczany przez port USB w komputerze, nie jest wystarczający, aby obsłużyć np. kamerę internetową podłączoną do Beagleboard. Objawiało się to problemami w komunikacji z kamerą, a czasem nawet błędami typu *kernel panic* systemu Linux.

Problemy te nie występowały, gdy zasilanie odbywało się z wykorzystaniem zewnętrznego zasilacza, za który może służyć dowolny stabilizowany zasilacz o napięciu wyjściowym 5V i maksymalnym obciążeniu 1A, wyposażony we wtyczkę 2.1mm.

Przewód do magistrali RS232 Podstawowym sposobem komunikacji z Beagleboard, jest wykorzystanie konsoli tekstowej, udostępnionej przez port szeregowy. Niestety, ze względu na oszczędność miejsca, producent urządzenia nie umieścił na płytce standardowego złącza DB9, a jedynie złącze IDC (ang. *Insulation-displacement connector*), wyposażone w 10 pinów. Powodowało to konieczność posiadania specjalnego przewodu, zakończonego z jednej strony złączem IDC10F (żeńskim), a z drugiej wtyczką DB9F (żeńską), który pokazano na rysunku 6.2.



Rysunek 6.2: Przewód RS232 łączący komputer i Beagleboard. Źródło: opracowanie własne.

Koncentrator USB Beagleboard wyposażony jest w jedno złącze USB, pracujące w trybie Host, do którego można podłączać praktycznie dowolne urządzenia (ograniczeniem jest niewielka wydajność prądowa tego złącza). Jest to zdecydowanie zbyt mało do poważniejszych systemów. W przypadku omawianego systemu, konieczne okazało się podłączenie aż trzech urządzeń (kamera, czujnik odległości, karta sieciowa). Najprostszym rozwiązaniem tego problemu okazało się wykorzystanie aktywnego koncentratora (HUB) USB. Po podłączeniu do urządzenia udostępnił on aż osiem wejść USB, a dzięki zewnętrznemu zasilaniu zwiększył wydajność prądową każdego z nich.

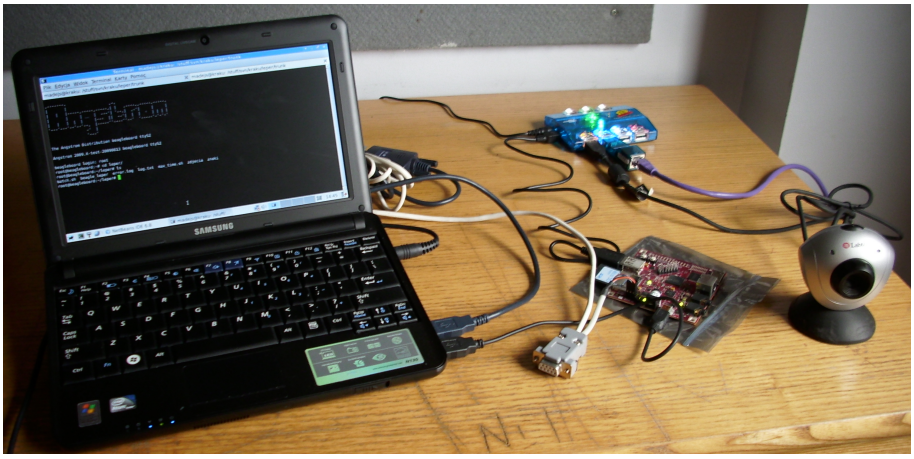
Interfejs USB<−>Ethernet Podłączenie przez Ethernet stanowi doskonałe rozszerzenie możliwości komunikacji przez magistralę szeregową, jednocześnie pozwalając na wykorzystanie komputera w roli serwera (np. HTTP, FTP). Najprostszym sposobem podłączenia płytki do sieci Ethernet, jest wykorzystanie prostej karty sieciowej podłączanej pod port USB, którą można kupić już za kilka złotych. Karta ta, w systemie Linux, widziana jest jako pełnoprawny interfejs sieciowy, co pozwala wykorzystywać ją np. jako sposób dostępu do internetu, w przypadku podpięcia do odpowiednio skonfigurowanego routera.

Karta pamięci SD/SDHC Dostępna na płytce pamięć Flash, stanowiąca podstawową pamięć masową urządzenia nie jest wystarczająca do pracy z systemem Linux. Producent zaleca użycie, jako pamięci masowej, karty pamięci o pojemności minimum 4GB. Karta ta musi być przed użyciem podzielona na partycje, w sposób opisany w rozdziale 6.2.1.2.

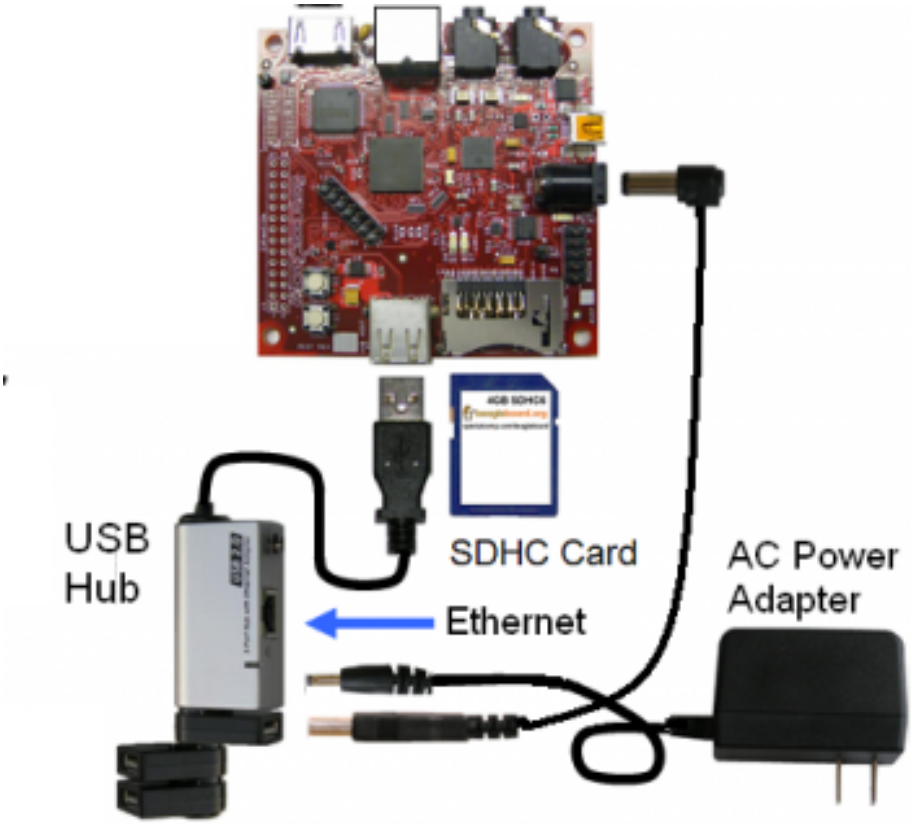
Rysunek 6.4 przedstawia schemat połączeń wykorzystujący powyższe elementy, a rysunek 6.3 prezentuje stanowisko testowe, na którym przeprowadzono testy algorytmu i implementację prototypu systemu parkingowego.

6.2.1.2. Ångström

Ångström jest dystrybucją systemu Linux polecaną przez producenta Beagleboard. Do jej podstawowych zalet należy zaliczyć:



Rysunek 6.3: Środowisko testowe z płytą Beagleboard. Źródło: opracowanie własne.



Rysunek 6.4: Schemat połączenia elementów środowiska testowego. Źródło: http://www.johandekoning.nl/index.php/tag/beagleboard/

- pliki binarne z obrazem systemu, gotowe do wgrania na kartę pamięci,
- repozytorium programów, w postaci binarnych pakietów,
- możliwość zbudowania własnego obrazu dystrybucji, prosto ze źródeł, dzięki zastosowaniu programu BitBake.

Podczas testów wykorzystano wersję systemu przygotowaną w postaci gotowego pliku (obrazu), który można wgrać na kartę SD. Czynność tą należy poprzedzić przygotowaniem karty SD.

Przygotowanie karty SD Karta stosowana w Beagleboard musi być przede wszystkim podzielona na odpowiednie logiczne części (partycje). Operację tą najłatwiej wykonać na komputerze działającym pod kontrolą systemu Linux (lub podobnego) i wyposażonego w czytnik kart SD. Najlepszym narzędziem do tego celu jest program *fdisk*, działający w tzw. *Expert mode* [7].

Pierwszym krokiem, jest wyczyszczenie istniejących partycji:

```
$ sudo fdisk /dev/sdb

Command (m for help): o
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course,
the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4
will be corrected by w(rite)
```

Następnie należy wyświetlić informacje szczegółowe o nośniku, jednocześnie zapisując pojemność karty podaną w bajtach.

```
Command (m for help): p
Disk /dev/sdb: 128 MB, 128450560 bytes
```

Dalsze czynności będą wykonywane w trybie zaawansowanym (Expert mode), do którego można wejść wybierając jako komendę *x*.

```
Command (m for help): x
```

Kolejnym krokiem jest ustawienie geometrii karty na 255 głowic (head) i 63 sektory (sectors) oraz wyznaczyć liczbę cylindrów zależnie od rozmiaru karty, korzystając ze wzoru 6.1.

$$liczbacylindrow = \lfloor \frac{rozmiarkarty(bajty)}{liczbaglowic \cdot liczbasektorow \cdot rozmiarsektora} \rfloor \quad (6.1)$$

$$= \lfloor \frac{128450560}{255 \cdot 63 \cdot 512} \rfloor = \lfloor 15,616557734 \rfloor = 15 \quad (6.2)$$

```
Expert command (m for help): h
Number of heads (1-256, default 4): 255

Expert command (m for help): s
Number of sectors (1-63, default 62): 63
```

```
Warning: setting sector offset for DOS compatiblity

Expert command (m for help): c
Number of cylinders (1-1048576, default 1011): 15
```

Dopiero w tym momencie można rozpocząć faktyczny podział karty na partycje. Potrzebne będą dwie:

- startująca partycja FAT32, z przeznaczeniem na plik wykorzystywany podczas startu systemu (boot image),
- partycja Linux, na której przechowywana będzie struktura katalogów systemu Ångström oraz pliki użytkownika.

```
Expert command (m for help): r
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-245, default 1): (press Enter)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-245, default 245): +50

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): a
Partition number (1-4): 1

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (52-245, default 52): (press Enter)
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (52-245, default 245):
Using default value 245
```

Ostatnim krokiem procesu tworzenia partycji jest wydrukowanie podsumowania i zapisanie nowo powstałej tablicy partycji na nośniku.

```
Command (m for help): p

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1          51      409626    c  W95 FAT32 (LBA)
/dev/sdc2              52         245     1558305   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16:
Device or resource busy. The kernel still uses the old
table. The new table will be used at the next reboot.

WARNING: If you have created or modified any DOS 6.x partitions,
please see the fdisk manual page for additional information.
Syncing disks.
```

Otrzymane w ten sposób partycje należy sformatować. Może wymagać to wyjęcia i włożenia karty z powrotem, aby system mógł odczytać nową strukturę partycji i zamontować nowe urządzenie. Do formatowania w systemie Linux służą programy z pakietu *mkfs*, o sufiksach odpowiadających systemowi plików.

```
$ sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL
mkfs.msdos 2.11 (12 Mar 2005)

$ sudo mkfs.ext3 /dev/sdc2
mke2fs 1.40-WIP (14-Nov-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
195072 inodes, 389576 blocks
19478 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=402653184
12 block groups
32768 blocks per group, 32768 fragments per group
16256 inodes per group
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information:
```

Na tak przygotowaną kartę pamięci można już wgrać system Ångström

w wersji przygotowanej dla Beagleboard. Zawsze aktualny komplet plików dostępny jest na stronie:

`http://www.angstrom-distribution.org/demo/beagleboard/`

Należy pobrać pliki:

- Angstrom-Beagleboard-demo-image*.tar.bz2
- MLO
- u-boot.bin
- uImage
- modules-2.6.X-rX-beagleboard.tgz (w wersji dopasowanej do jądra, które zostało użyte w pierwszym pliku)

Następnie obie partycje trzeba zamontować w systemie plików (dla przykładu będzie to odpowiednio /mnt/sda1 i /mnt/sda2) i skopiować na nie potrzebne pliki:

```
$ cp MLO /mnt/sda1
$ cp u-boot.bin /mnt/sda1
$ cp uImage /mnt/sda1
$ sudo cp Angstrom-Beagleboard-demo-image*.tar.bz2 /mnt/sda2
$ sudo cp modules-2.6.X-rX-beagleboard.tgz /mnt/sda2
```

Teraz można rozpakować system plików oraz moduły:

```
$ cd /mnt/sda2
$ sudo tar -jxvf Angstrom-Beagleboard-demo-image*.tar.bz2
$ sudo rm Angstrom-Beagleboard-demo-image*.tar.bz2
$ sudo tar -xvf modules-2.6.X-rX-beagleboard.tgz
$ sudo rm modules-2.6.X-rX-beagleboard.tgz
```

Na tym etapie kartę można już odmontować i wyjąc z czytnika, a następnie umieścić w Beagleboard. Urządzenie należy podłączyć do portu szeregowego komputera i innych urządzeń opisanych wcześniej, a następnie podłączyć przewód zasilający. Po maksymalnie kilku minutach (tylko przy pierwszym uruchomieniu), powinien pojawić się komunikat powitalny i prośba o login:

```
.-------.
|       |                  .-.
|   |   |-----.-----.-----.| |   .----..-----.-----.
|       |     | __  |  ---'| '--.|  .-'|     |     |
|   |   |  |  |     |---  ||  --'|  |  |  '  | | | |
'---'---'--'--'--.  |-----''----''--'  '-----'-'-'-'
                -'  |
                '---'

The Angstrom Distribution beagleboard ttyS2

Angstrom 2011.1-test-20110918 beagleboard ttyS2

beagleboard login:
```

Wystarczy wpisać *root* i nacisnąć enter. W tym momencie można już w pełni korzystać z systemu Ångström linux na urządzeniu.

6.2.1.3. Środowisko deweloperskie

Proces pisania, kompilacji i testowania programów na Beagleboard można znacząco ułatwić, instalując pełen zestaw narzędzi deweloperskich na urządzeniu. Najłatwiej to zrobić jeśli system posiada skonfigurowane połączenie z internetem, dzięki dostępowi do repozytorium pakietów, którym zarządza się przy użyciu menadżera pakietów *opkg*. Przed instalacją jakiegokolwiek nowego oprogramowania warto upewnić się, że system posiada informację o aktualnie dostępnych pakietach. Można to wymusić poprzez wykonanie poleceń:

```
# opkg update
# opkg upgarde
```

Narzędzia deweloperskie, które warto następnie zainstalować, przy użyciu *opkg install*, to:

- bash (powłoka systemu),
- task-nativ-sdk (kompilator GCC i biblioteki systemowe),
- gdb (debugger),
- subversion (kontrola wersji),
- libopencv-core2.2 (OpenCV - główna biblioteka),
- libopencv-highgui2.2 (OpenCV - dostęp do plików),
- libopencv-ml2.2 (OpenCV - uczenie maszynowe),
- lighttpd (serwer HTTP),
- cmake (zarządzanie kompilacją).

6.2.2. PlayStation Eye

PlayStation Eye (rysunek 6.5) to urządzenie oparte na kamerze cyfrowej, które powstało jako akcesorium do konsoli PlayStation 3. Głównymi zastosowaniami urządzenia były gry wykorzystujące wizję komputerową, rozpoznawanie gestów i rozszerzoną rzeczywistość (*ang. Argumented Reality*).

W skład urządzenia wchodzą kamera cyfrowa i mikrofon kwadrofoniczny, a całość jest podłączana do konsoli przez port USB [30].

Parametry kamery:

- maksymalna rozdzielczość rejestrowanego obrazu 640x480,
- 60 Hz przy rozdzielczości 640x480,
- 120 Hz przy rozdzielczości 320x240,
- wysoka czułość, pozwalająca na pracę przy słabym oświetleniu (wg. producenta wystarczy oświetlenie wytwarzane przez telewizor),
- dwa stałe kąty widzenia kamery: 56° i 75°,
- przesyłane obrazy mogą być nieskompresowane, co poprawia jakość obrazu.

Parametry mikrofonu:

- cztery niezależne mikrofony,

Rysunek 6.5: Urządzenie Playstation Eye; na górze widoczna grupa mikrofonów. Źródło: http://en.wikipedia.org/wiki/PlayStation_Eye

- każdy kanał przetwarza 16-bitowe próbki z częstotliwością 48 kHz i SNR 90dB,
- wsparcie dla śledzenia źródła dźwięku,
- wsparcie dla usuwania echa,
- wsparcie dla usuwania szumu tła.

Urządzenie zostało zaprojektowane do zastosowań związanych z przetwarzaniem obrazów, dzięki temu ostrość obrazu i czułość urządzenia doskonale sprawdzają się podczas rejestrowania tablic rejestracyjnych z niewielkiej odległości (rzędu pojedynczych metrów). Dodatkowo zastosowany przez firmę Sony czujnik OV7725 firmy Omnivision działa sprawnie pod kontrolą sterownika wbudowanego w jądro systemu Linux od wersji 2.6.29. Pozwala to na łatwe podpięcie kamery do dowolnego urządzenia pracującego pod kontrolą systemu Linux, o ile urządzenie to posiada kontroler USB pracujący w trybie Host.

6.2.3. Czujnik odległości

Dużym ułatwieniem w procesie rozpoznawania tablicy rejestracyjnej była znajomość jej rozmiarów, co pozwalało na odrzucenie znaków o niewłaściwych rozmiarach oraz na dobranie współczynników do progowania adaptacyjnego. W przypadku rozpoznawania statycznego, jakie używane jest w systemach parkingowych, możliwe jest zastosowanie czujnika odległości. Czujnik taki pozwala na określenie w jakiej odległości od kamery zatrzymał się samochód lub wyzwoli proces rozpoznawania, gdy samochód znajdzie się w określonej odległości. W automatyce i robotyce najczęściej stosuje się czujniki odległości opierające się o pomiar czasu przelotu impulsu odbitego do przeszkody lub pomiar przesunięcia fazowego pomiędzy sygnałem wysłanym i odebranym. „Jako medium (nośnik energii) najczęściej stosowane są:

- podczerwień (dalmierze IR),

- ultradźwięki (sonary),
- skupiona wiązka światła (dalmierze laserowe),
- fale radiowe (radary)."[19]

W niniejszej pracy zastosowano sonarowy czujnik odległości MOBOT-USv2 (rysunek 6.6), pracujący w paśmie ultradźwięków.

Parametry urządzenia:

- pomiar w zakresie od 5 cm do 3,5 m,
- rozdzielczość 3 mm,
- komunikacja przez interfejs szeregowy I2C,
- możliwość współpracy do 8 urządzeń na jednej magistrali, dzięki konfigurowalnemu adresowi,
- napięcie zasilania 5 V,
- pobór prądu: 25 mA - praca ciągła, 2,5 mA tryb uśpienia.

Rysunek 6.6: Sonarowy czujnik odległości MOBOT-USv2. Źródło: http://www.mobot.pl/download/images/sonar_Mobot-usv2_01_o.jpg

Czujnik ten został wybrany przede wszystkim ze względu na zakres pracy, odpowiadający założonym odległościom przy jakich ma być wykonywane zdjęcie (1-2 m) oraz bardzo dobrą dokumentację i prosty protokół komunikacyjny. Niestety ze względu na problemy z obsługą magistrali I2C na Beagleboard, została podjęta decyzja o zbudowaniu konwertera I2C-USB, który pozwoliłby na łatwą komunikację z czujnikiem, z dowolnego urządzenia obsługującego USB Host.

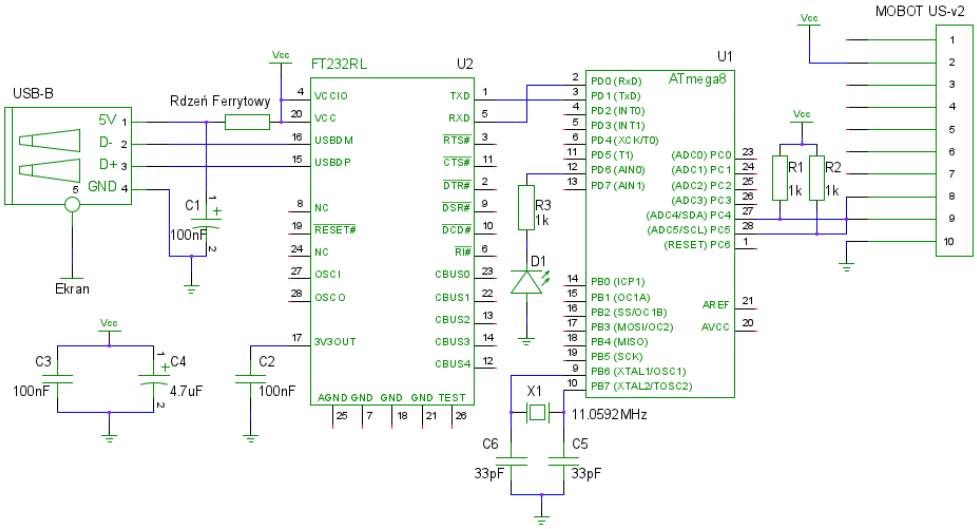
Konwerter został zbudowany w oparciu o dwa układy scalone:

- mikrokontroler ATMEGA8 firmy ATMEL[4],
- konwerter RS232-USB FT232R firmy FTDI Chip[13].

Ze względów konstrukcyjnych (układ FT232R w obudowie SSOP, wymagającej montażu powierzchniowego), podjęto decyzje o wykorzystaniu modułu konwertera AVT MOD09, zawierającego układ FT232R oraz złącze USB-B (rysunek 6.7).

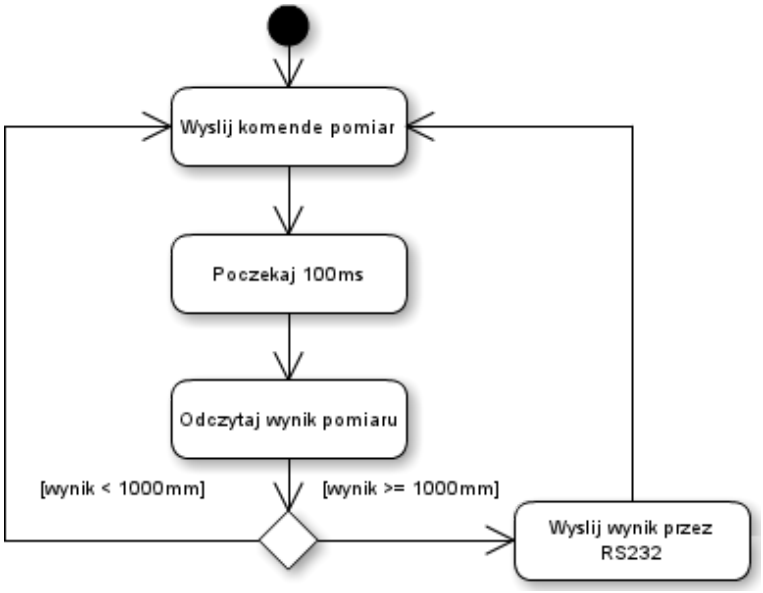
Rysunek 6.8 przedstawia schemat wykonanego konwertera. Oprogramowanie

Rysunek 6.7: Moduł konwertera RS232-USB AVT MOD09. Źródło: http://sklep.avt.pl/photo/product_big/d/7/f/1_d7f027631365.jpg



Rysunek 6.8: Schemat konwertera I2C-USB, stworzonego do współpracy z sonarem MOBOT US-v2. Źródło: opracowanie własne.

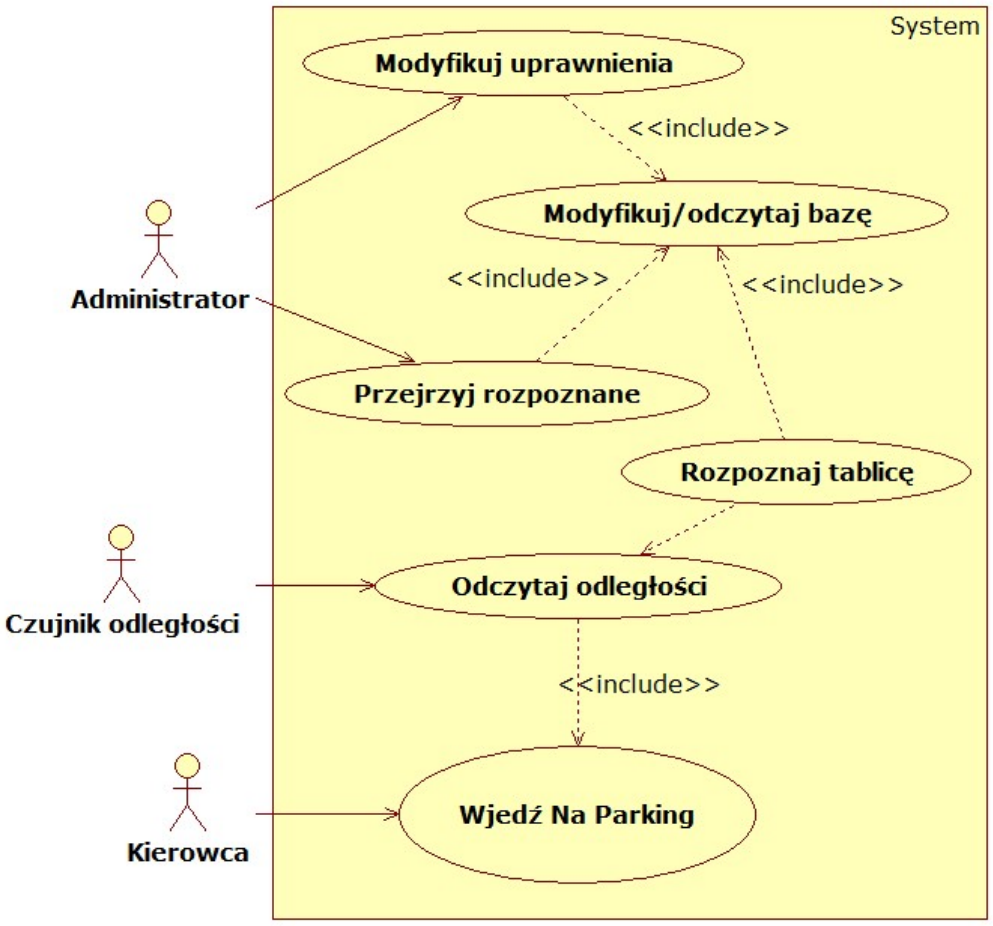
do mikrokontrolera ATMEGA8 napisane zostało w języku C i skompilowane przy użyciu kompilatora GCC AVR. Jego podstawowym zadaniem jest cykliczne wykonywanie pomiaru odległości przy użyciu sonaru. Wymaga to wysłania do czujnika polecenia rozpoczęcia pomiaru, poczekania na zakończenie pomiaru i odczytania wyniku. Rysunek 6.9 przedstawia diagram aktywności dla programu konwertera.



Rysunek 6.9: Diagram aktywności ilustrujący działanie oprogramowania konwertera I2C-USB. Źródło: opracowanie własne.

6.3. Program Parking-LPR

Rysunek 6.10 przedstawia przypadki użycia dla systemu.



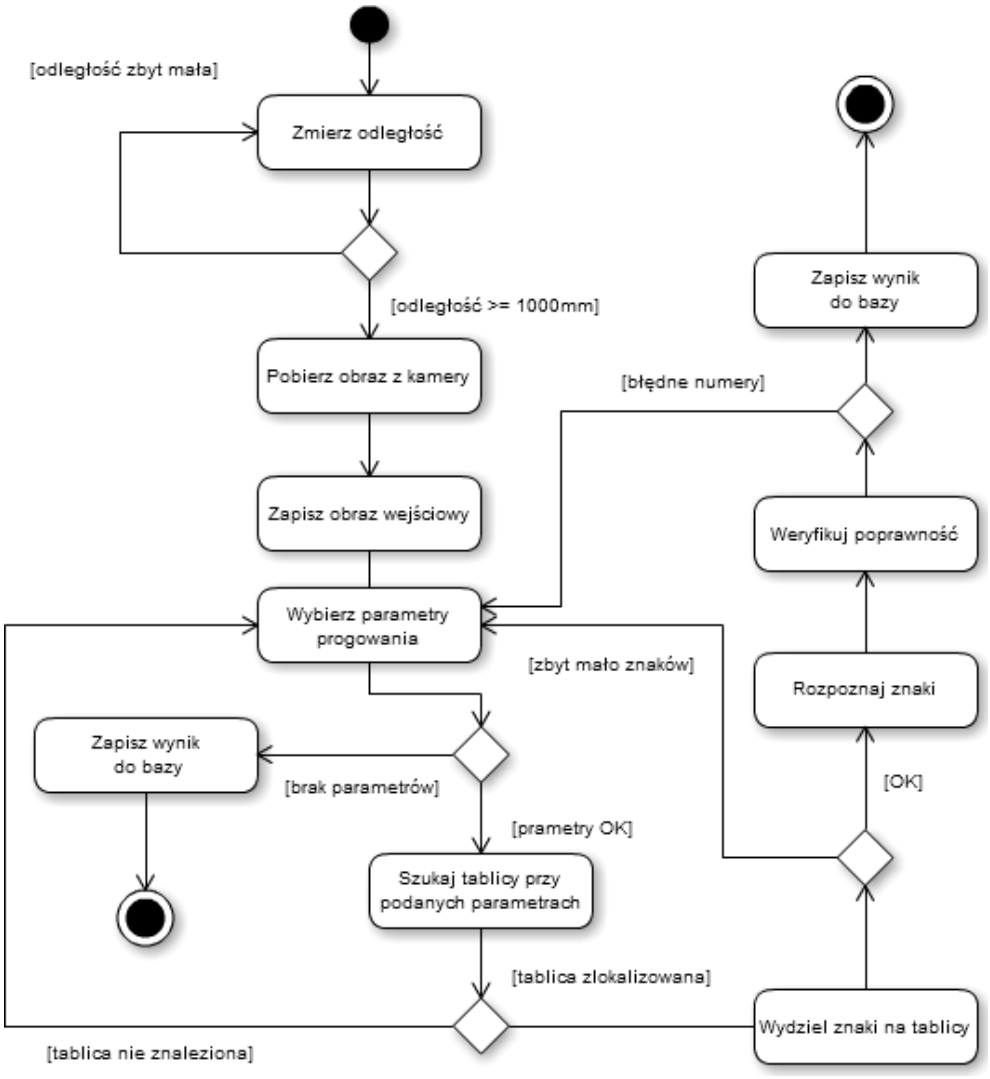
Rysunek 6.10: Przypadki użycia dla systemu parkingowego. Źródło: opracowanie własne.

6.3.1. Specyfikacja funkcjonalna

Program Parking-LPR stanowi główny moduł całego systemu. Jest to program rezydentny, działający w tle i nie wymagający jakiejkolwiek ingerencji użytkownika. Uruchamiany jest podczas startu urządzenia i natychmiast rozpoczyna pracę (rysunek 6.11).

Jego działanie polega na ciągłym nasłuchiwaniu na porcie szeregowym, do którego podpięty jest czujnik i sprawdzaniu czy pojawiła się nowa wartość odczytu i czy wartość ta mieści się w zdefiniowanym przedziale pracy urządzenia. W przypadku spełnienia obu tych warunków, następuje rozpoczęcie akwizycji obrazu i uruchomienia procesu rozpoznawania tablicy, z uwzględnieniem odczytanej odległości kamery (z podpiętym czujnikiem) od tablicy.

Wynik wraz z danymi o czasie rozpoznania jest zapisywany w lokalnej bazie danych, skąd dane te dostępne są dla użytkownika poprzez panel admi-



Rysunek 6.11: Diagram aktywności ilustrujący działanie programu. Źródło: opracowanie własne.

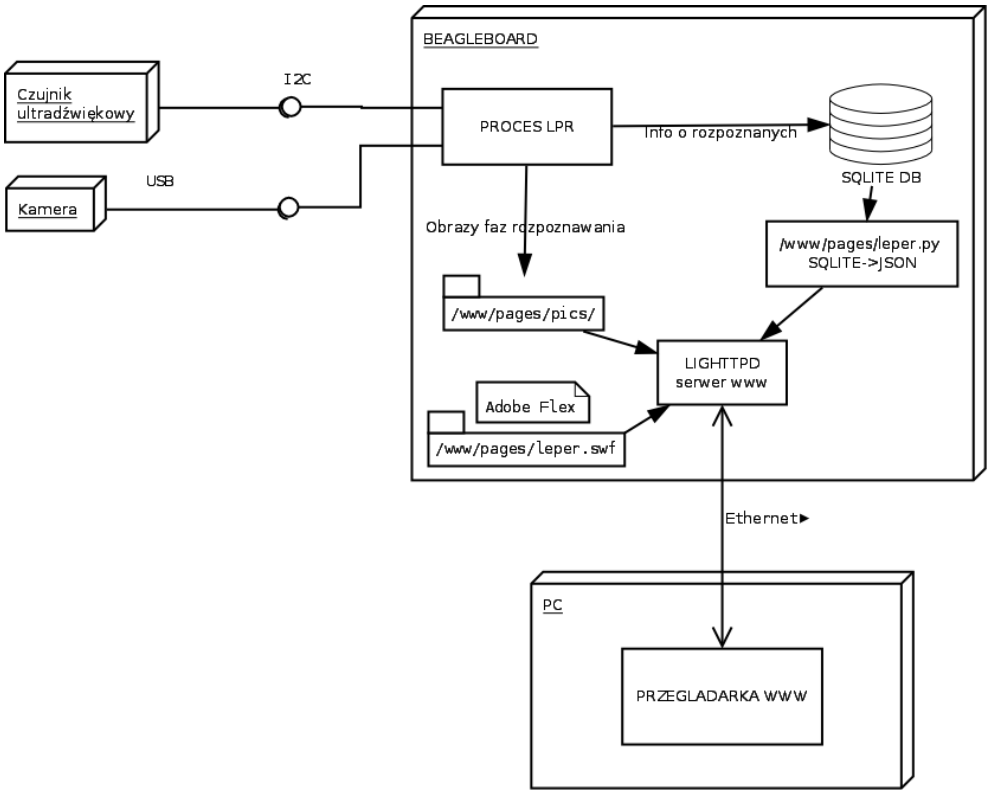
nistracyjny. Dodatkowo aplikacja zapisuje pliki zawierające obrazy pobrane z kamery oraz odpowiadające im obrazy reprezentujące wynik przetwarzania.

Rysunek 6.12 przedstawia diagram wdrożenia systemu na urządzeniu oraz połączenia z zewnętrznymi urządzeniami.

6.3.2. Szczegóły implementacji

Program Parking-LPR napisany został w języku C++ z wykorzystaniem biblioteki OpenCV 2.2. Za proces budowania projektu odpowiedzialny był program CMake. Zastosowanie takie zestawu narzędzi pozwoliło na stworzenie rozwiązania przenośnego, które mogło być kompilowane i uruchamiane zarówno pod systemem Linux (kompilator GCC), jak i Windows (kompilatory GCC, Visual C++). Zgodnie z wynikami przeprowadzonych badań, zastosowany został Algorytm III opisany w rozdziale 4.4.

Kod programu zorganizowany został w postaci trzech modułów:



Rysunek 6.12: Diagram przedstawiający zależności pomiędzy programowymi i sprzętowymi elementami systemu. Źródło: opracowanie własne.

- biblioteki *liblocalsqlite3* obsługującej komunikację z bazą danych,
- biblioteki *liblpr* zawierającej implementację algorytmu rozpoznawania tablic,
- głównego programu, korzystającego z powyższych bibliotek i odpowiedzialnego m.in. za komunikację z czujnikiem.

Wydzielenie biblioteki *liblpr* pozwoliło na łatwe testowanie samego algorytmu oraz tworzenie nowych wersji programu korzystających z zaimplementowanego już algorytmu rozpoznawania tablic rejestracyjnych. Biblioteka ta, rozbita została na moduły, które komunikują się ze sobą dzięki komunikatom przesyłanym pomiędzy nimi. Całość została zaimplementowana w postaci klas C++ , opisanych w rozdziale 6.3.3.

6.3.3. Opis klas z biblioteki liblpr

ImageProcessor Zadaniem tej klasy jest przygotowanie obrazu do segmentacji, poprzez zastosowanie rozmycia filtrem medianowym 3x3, a następnie progowania adaptacyjnego. Tak przygotowany obraz, jest następnie poddawany segmentacji, w metodzie *selectSegments()*, która wyznacza wszystkie spójne obszary, spełniające warunki konieczne bycia znakiem. Metoda ta

jest wykorzystywana po raz drugi, do wyznaczenia segmentów na obszarze tablicy rejestracyjnej, po wyznaczeniu jej obszaru.

SegmentGroup Kolejnym etapem procesu lokalizacji tablicy, jest połączenie uzyskanych w poprzedniej klasie segmentów w grupy i wydzielenie tylko tych które mają przynajmniej cztery segmenty. Wartość ta, wynika z faktu, że minimalna liczba znaków na tablicy wynosi pięć oraz, że należy uwzględnić możliwość pominięcia pojedynczych znaków, na tym etapie. Wynikiem działania głównej metody tej klasy *createSegmentsGroups(segments)*, jest kolekcja grup segmentów, które mogą stanowić tablicę rejestracyjną.

OCRProcessor Każda z grup segmentów wykorzystywana jest w klasie *OCRProcessor*, do wyznaczenia obszaru tablicy w metodzie *getOnlyPlate()*. Uzyskany w ten sposób obraz tablicy, jest przekazywany znów do klas *ImageProcessor* i *SegmentGroup*, w celu dokładniejszego wyznaczenia segmentów i zapisania ich w formie kolekcji obiektów klasy *CharSegment*. Wektory uzyskanych w ten sposób segmentów, posortowane wg kolejności, od lewej do prawej, stanowią dane wejściowe dla algorytmu OCR. Implementacja tego algorytmu znajduje się w metodzie *recognize(charSegments)*, która to, wykorzystuje, wczytaną w metodzie *loadMLPNetwork()*, sieć neuronową typu MLP. Każdy z segmentów, jest podawany na wejście sieci, a następnie wybierany jest znak odpowiadający wyjściu o największej wartości. Znaki te są kolejno dopisywane do łańcucha wynikowego, który po zakończeniu rozpoznawania jest poddawany sprawdzeniu pod względem poprawności.

CharSegment Pojedyncze segmenty zapisywane są jako obiekty klasy *CharSegment*, która zapisuje m.in. znormalizowany obrazu segmentu i jego mapę gęstości, wykorzystywaną przez algorytm OCR.

NumberValidator *NumberValidator* to ostatnia z kluczowych klas w bibliotece *liblpr*. Jej zadaniem jest sprawdzenie, czy ciąg znaków wyznaczony przez algorytm OCR, jest poprawnym numerem rejestracyjnym. Ponadto, dla poprawnych numerów, wyznaczany jest typ tablicy oraz powiat, z którego pochodzi.

6.3.4. Opis elementów programu Parking-LPR

main.cpp Plik ten zawiera metodę *main()* i stanowi punkt startowy programu. Metoda ta pobiera argumenty wywołania programu, analizuje je i w zależności od ustawionych parametrów, uruchamia program w jednym z trzech trybów pracy:

- ciągły odczyt wartości z czujnika i rozpoznawanie w przypadku przekroczenia wartości progowej,
- tryb rozpoznawania obrazów z pliku (wykorzystywany podczas testów),
- tryb przechwytywania obrazów z kamery, z jednoczesnym podglądem i odczytem wartości z czujnika (wykorzystywany przy zbieraniu obrazów testowych).

W każdym z przypadków, proces działania nadzoruje klasa *LPRManager*.

LPRManager Klasa ta stanowi główny moduł sterujący, który zarządza działaniem algorytmu rozpoznawania, odczytem z czujnika oraz komunikacją z bazą danych. Główny proces rozpoznawania, zarówno z kamery jak i z pliku, jest obsługiwany w metodzie *findAndRecognizePlateInImage(distance)*. Metoda ta, korzystając z informacji o odległości, pobiera obraz z kamery lub pliku, a następnie dokonuje rozpoznania, w sposób opisany w rozdziale 6.3.3 i zapisuje wyniki do bazy danych oraz do plików graficznych, w systemie plików urządzenia.

Database Zadaniem tej klasy jest udostępnienie łatwego w użyciu interfejsu do bazy danych, w której przechowywane są wyniki rozpoznawania. Głównymi metodami tej klasy, są metody *insert()*, *open()* i *close()*, które pozwalają odpowiednio: zapisywać wyniki, otwierać i zamykać połączenie z bazą.

Sensor Ostatnia z opisywanych klas odpowiada za komunikację z czujnikiem odległości, po magistrali USB. Jej głównym elementem jest blokująca metoda *waitForRightDistance(barrier)*, która oczekuje, aż czujnik zwróci wartość powyżej wartości progowej, a potem wartość tą zwraca do programu.

6.4. Zarządzanie z poziomu przeglądarki WWW

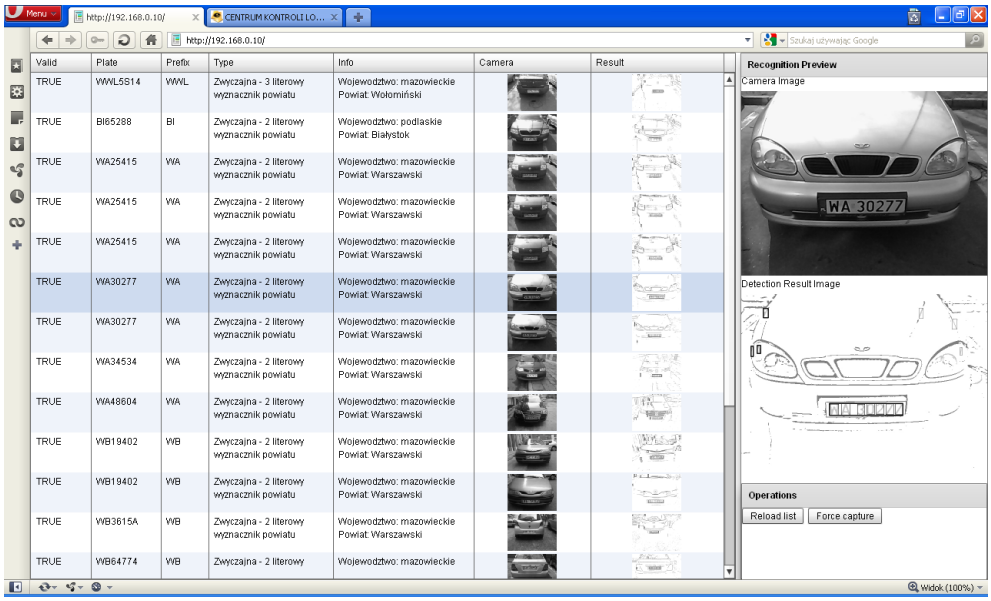
W przypadku systemu parkingowego, ważna jest możliwość modyfikacji listy pojazdów uprawnionych i bieżącej kontroli informacji o pojazdach, które poddane zostały procesowi rozpoznawania. W przypadku systemu Parking-LPR, możliwości takie daje panel administracyjny dostępny dla użytkownika z poziomu przeglądarki internetowej wyposażonej w wtyczkę Adobe Flash Player. Panel składa się z dwóch widoków:

- listy pojazdów poddanych rozpoznawaniu (Recognition results),
- panelu zarządzania uprawnieniami (Access management).

Oba widoki dostępne są bezpośrednio po wpisaniu w pasek adresu przeglądarki internetowej adresu IP urządzenia. Domyślnie otwierany jest widok listy pojazdów, jednak zmiana widoku sprawdza się jedynie do kliknięcia na belce z nazwą tego widoku.

6.4.1. Lista pojazdów poddanych rozpoznawaniu

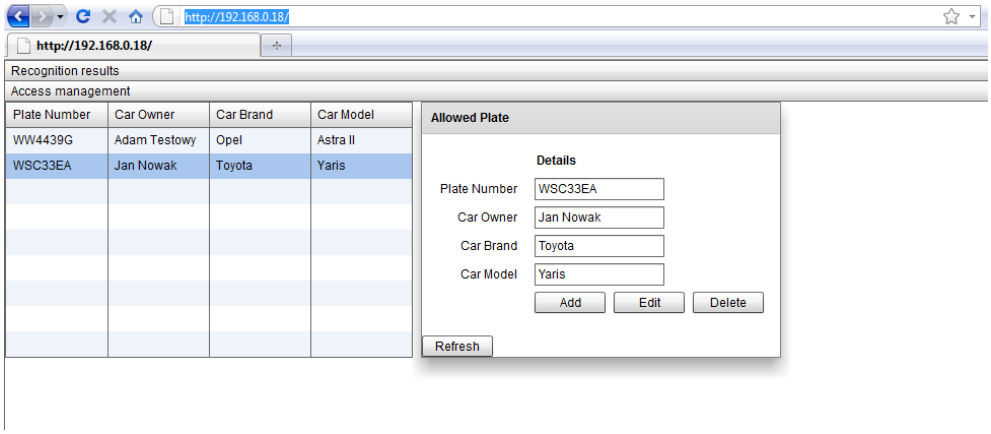
Widok listy pojazdów (rysunek 6.13), składa się z dwóch części, działających na zasadzie lista-szczegół (*ang. Master-Detail*). Po lewej stronie znajduje się lista pojazdów poddanych rozpoznaniu, wraz z wynikami rozpoznania. Prawa strona zawiera widok szczegółowy, z podglądem obrazu wejściowego (Camera Image) i obrazu będącego wynikiem progowania wraz z zaznaczeniem zlokalizowanej tablicy i wyznaczonych segmentów. Dodatkowo użytkownik może odświeżyć listę pojazdów (Reload list) lub ręcznie wymusić pobranie obrazu z kamery i rozpoznanie (Force capture).



Rysunek 6.13: Panel administracyjny z lista pojazdów poddanych rozpoznawaniu, wraz z podglądem uchwyconego obrazu i wyniku rozpoznawania. Źródło: opracowanie własne.

6.4.2. Ekran zarządzania uprawnieniami

Drugi z widoków, widoczny na rysunku 6.14, służy do przeglądania i modyfikacji listy pojazdów dopuszczonych do wjazdu na parking. Lista ta jest przeglądana przez program, w celu ustalenia czy pojazd, którego tablica została rozpoznana, ma zezwolenia na wjazd i czy należy uruchomić ewentualny moduł wykonawczy np. otworzyć szlaban.



Rysunek 6.14: Panel administracyjny służący do zarządzania listą pojazdów uprawnionych do wjazdu. Źródło: opracowanie własne.

7. Podsumowanie i wnioski

W pracy zestawione zostały najpopularniejsze algorytmy rozpoznawania tablic rejestracyjnych, które mogą być wykorzystywane w systemach wbudowanych. Przedstawiono także autorskie rozwiązanie, które zostało zaprojektowane z myślą o tego typu zastosowaniach.

Przeprowadzone badania wykazały, że połączenie progowania adaptacyjnego i algorytmu rozrostu ziarna, pozwala na szybkie i efektywne lokalizowanie tablicy rejestracyjnej na obrazie, przy założeniu, że jej rozmiar jest w przybliżeniu znany (np. dzięki pomiarowi odległości tablicy od kamery). Ponadto wykazano, że połączenie lokalizacji i segmentacji znaków, pomimo niewątpliwej szybkości, wymaga zastosowania dodatkowego etapu dokładnej lokalizacji segmentów, na wstępnie wyznaczonym obszarze tablicy rejestracyjnej, poprzez wybranie lepszych parametrów progowania oraz bardziej rygorystycznych kryteriów odrzucenia segmentów.

Z przeprowadzonych badań wynika również, że zastosowanie sztucznej inteligencji, a w szczególności sieci neuronowej typu MLP, widocznie zmniejsza liczbę pomyłek pomiędzy podobnymi znakami, co było problemem w przypadku algorytmu OCR wykorzystującego korelację pomiędzy wzorcami.

Algorytm będący wynikiem badań został wykorzystany w prostym systemie parkingowym, z możliwością zarządzania z poziomu przeglądarki WWW. W skład systemu wchodziły:

- komputer jednoukładowy Beagleboard,
- kamera PS Eye,
- opracowany na potrzeby systemu parkingowego czujnik odległości, podłączany do portu USB,
- aktywny koncentrator USB,
- program Parking-LPR wraz z modułem dostępu przez WWW.

W procesie opracowywania algorytmu i tworzenia systemu parkingowego napotkano problemy, wynikające z faktu, że opracowany algorytm słabo sobie radził z zakłóceniami w postaci grubych linii poziomych lub krawędzi, łączących ze sobą więcej niż dwa znaki oraz śrubami łączącymi obrysy znaków. Problem ten jednak widoczny jest także w przypadku biblioteki DTK ANPR SDK, która wykorzystana została do testów porównawczych skuteczności rozpoznawania. Zjawisko to można ograniczać poprzez umieszczenie kamery wykonującej zdjęcia na wysokości tablicy rejestracyjnej, jednak w przypadku wystąpienia śruby łączącej znaki, rozwiązaniem może okazać się zastosowanie podziału zauważalnie szerszych segmentów na tablicy rejestracyjnej np. z zastosowaniem rzutu jasności.

Wyniki i wnioski z badań, dla pierwszych dwóch algorytmów, zostały wykorzystane w artykule [22] oraz zaprezentowane na konferencji Systemy Czasu

Rzeczywistego, która odbyła się we wrześniu 2010 roku. Ponadto, w trakcie implementacji systemu parkingowego, przeprowadzono, z powodzeniem, próby uruchomienia algorytmu, na urządzeniu opartym o system Android. Świadczy to, że algorytm może być stosowany na szerokiej gamie urządzeń, a jego konwersja z jednej platformy na drugą nie sprawiała większego problemu. Bardzo przyjemnym zaskoczeniem okazało się porównanie szybkości i wydajności opracowanego algorymtu, z jedną z dostępnych na rynku komercyjnych bibliotek ANPR, opracowaną przez firmę DTK Software. Algorytm będący wynikiem pracy, w trakcie testów, uzyskał o 10% większą skuteczność rozpoznawania, będąc jednocześnie nieznacznie szybszym niż aplikacja demonstracyjna dostarczana przez producenta biblioteki DTK ANPR.

Bibliografia

[1] Algorytm k-najbliższych sąsiadów, http://www.naukowy.pl/encyklopedia/Algorytm_k-najbliższych_sąsiadów

[2] Arabas J., Cichosz P., Sztuczna inteligencja Moduł 10 - Zadanie i metody klasyfikacji, http://wazniak.mimuw.edu.pl/index.php?title=Sztuczna_inteligencja/SI_Modu\OT4\l_10

[3] Arabas J., Cichosz P., Sztuczna inteligencja Moduł 12, http://wazniak.mimuw.edu.pl/index.php?title=Sztuczna_inteligencja/SI_Modu\OT4\l_12

[4] ATMEGA8 Datasheet, Atmel Corporation, 2011, http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

[5] Balázs E., Lajos K., Ká lmán F. : Real Time Number Plate Localization Algorithms. Journal of ELECTRICAL ENGINEERING, VOL. 57, NO. 2, 2006, 69–77

[6] Barroso, J.; Dagless, E.L.; Rafael, A.; Bulas-Cruz, J., Number plate reading using computer vision, Proceedings of the IEEE International Symposium, Industrial Electronics, 1997, ISIE '97. s. 761-766.

[7] BeagleBoardBeginners, eLinux.org - Embedded Linux Wiki, http://elinux.org/BeagleBoardBeginners

[8] Bradski, G., Kaehler, A. : Learning OpenCV, Computer Vision with OpenCV Library, O'REILLY (2008)

[9] Congestion Charging, Transport for London, http://www.tfl.gov.uk/roadusers/congestioncharging

[10] Czajewski, W.: Rough Sets in Optical Character Recognition. In Rough Sets and Current Trends in Computing(1998), s. 601-604

[11] Czajewski, W.; Staniak, M.: Biblioteka do akwizycji i rozpoznawania obrazów, Warszawa: Centrum Automatyki i Technik Informacyjno-Decyzyjnych, 2003.

[12] Czajewski, W.; Żuraw, J.: Zastosowanie teorii zbiorów przybliżonych do rozpoznawania znaków, Warszawa: Politechnika Warszawska, Instytut Sterowania i Elektroniki Przemysłowej, 1997.

[13] FT232R Datasheet, Future Technology Devices International Ltd., 2010, http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[14] Głowacka I., Rodziny klasyfikatorów na przykładzie algorytmów wzmacniających. BOOSTING, 19 listopada 2003 roku http://multifraktal.net/seminarium/boosting.pdf

[15] GV-LPR Licence Plate Recognition, GeoVison, 2011, http://www.geovision.com.tw/english/Datasheet/Datasheet_LPR.pdf

[16] Hansen H., Kristensen A. W., Køhler M. P., Mikkelsen A. W., Pedersen J. M., Trangeled M.: Automatic recognition of license plates, Aalborg Universitet, 26.05.2002

[17] Hu M.: Visual pattern recognition by moment invariants, IRE Transactions on Information Theory, Luty 1962, str. 179 - 187

[18] Iwanowski, M.: Automatic car number plates detection using morphological image processing *Przegląd Elektrotechniczny*, 2005, 81, pp.58-61

[19] Kajner K., Czulak B., Bugiera Ł., Grabas P., Struzik Z., Mój pierwszy robot, 2005,
http://rab.ict.pwr.wroc.pl/~arent/rr/mpr/czujniki3.html

[20] Kong, J.; Liu, X.; Lu, Y.; Zhou, Y.: A Novel License Plate Localization Method Based On Textural Feature Analysis, IEEE International Symposium on Signal Processing and Information Technology, 2005.

[21] Lewkowicz M., Identyfikacja i rozpoznawanie numerów tablic rejestracyjnych pojazdów samochodowych na sekwencjach obrazów wraz z analizą parametrów ruchu, Instytut Sterowania i Elektroniki Przemysłowej PW, 2006

[22] Madej S., Czajewski W.: Rozpoznawanie numerów tablic rejestracyjnych w czasie rzeczywistym, w systemach wbudowanych, Metody wytwarzania i zastosowania systemów czasu rzeczywistego, Rzeszów: Politechnika Rzeszowska, 2010, str. 399-409

[23] Martini J., Karpiesiuk Ł.: Raport branży motoryzacyjnej 2010, Polski Związek Przemysłu Motoryzacyjnego, 2011,
http://www.pzpm.org.pl/media/File/Raport%20branzy%20motoryzacyjnej%202010.pdf

[24] Martinsky O.: Algorithmic And Mathematical Principles Of Automatic Number Plate Recognition Systems. Brno University Of Technology, Brno 2007.

[25] Ministerstwo Infrastruktury: Rozporządzenie Ministra Infrastruktury z dnia 31 grudnia 2002 r. w sprawie warunków technicznych pojazdów oraz zakresu ich niezbędnego wyposażenia, Dz. U. z dnia 26 lutego 2003 r.

[26] Ministerstwo Transportu I Gospodarki Morskiej : Rozporządzenie Ministra Transportu i Gospodarki Morskiej, z dnia 22 lipca 2002 r., w sprawie rejestracji i oznaczania pojazdów. Dz. U. Nr 133, poz. 1123

[27] M3S-ANPR System automatycznego rozpoznawania numerów tablic rejestracyjnych, Polixel S.A., Warszawa
http://www.polixel.com.pl/doc/M3S_ANPR.pdf

[28] Neural Networks, OpenCV v2.3 documentation,
http://opencv.itseez.com/modules/ml/doc/neural_networks.html

[29] Frey P. W., Slate D. J.: Letter recognition using Holland-Style Adaptive Classifiers, Machine Lerning vol. 6, Kluwer Academic Publishers, 1991, str. 161-182

[30] PLAYSTATION®Eye Brings Next-Generation Communication to PLAYSTATION®3, 26.04.2007,
http://us.playstation.com/corporate/about/press-release/396.html

[31] Prokop R. J., Reeves A.: A survey of moment-based techniques for unoccluded object representation and recognition, CVGIP: Graphics Models and Image Processing, vol. 54 1992, str. 438-460

[32] Shapiro, V., Dimov, D., Bonchev, S., Velichkov, V., Gluhchev, G.: Adaptive License Plate Image Extraction, Proc. of CompSysTech'2003

[33] Shapiro, V., Gluchev, G., Dimov, D.:Towards a multinational car license plate recognition system Machine Vision and Applications, vol.17, 2006, pp.173-183

[34] Stach M.: Wyszukiwanie obrazów na podstawie zawartości, Akademia Górniczo-Hutnicza, Katedra Automatyki, Kraków 2010

[35] Stepura I. : ARM-wrestling with OpenCV,
http://www.computer-vision-software.com/blog/2009/03/arm-wrestling-with-opencv/

[36] Stroński M., Systemy ARTR (Automatycznego Rozpoznawania Tablic Rejestracyjnych) możliwości i wykorzystanie w systemach sterowania ruchem, Biuletyn KLIR nr 72, Styczeń 2010, str. 61-66
http://edroga.pl/inzynieria-ruchu/its/
[37] Tadeusiewicz R., Sieci neuronowe, Akademicka Oficyna Wydawnicza RM, Warszawa, 1993.
[38] viaTOLL: Elektroniczny system poboru opłat viaTOLL,
http://www.viatoll.pl
[39] viaTOLL: System viaTOLL - Jak działa?,
http://www.viatoll.pl/pl/pojazdy-ciezkie/system-viatoll
[40] viaTOLL: viaBox - Co to takiego?,
http://www.viatoll.pl/pl/pojazdy-ciezkie/viabox