AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I ROBOTYKI

Projekt dyplomowy

Sprzętowo-programowy system wizyjny do detekcji pasa ruchu
Hardware-software vision system for lane detection

Autor: *Bartosz Gil*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr inż. Tomasz Kryjak*

Kraków, 2019

Składam serdeczne podziękowania Panu Tomaszowi Kryjakowi za okazywaną cierpliwość i pomoc

Spis treści

1. Wprowadzenie 7

1.1. Cele pracy 8

1.2. Układ pracy 9

2. Przegląd algorytmów z literatury 11

2.1. Wstępne przetwarzanie obrazu 11

2.2. Wyznaczanie obszaru zainteresowania 12

2.3. Wyznaczanie linii drogowych 13

2.4. Fuzja obrazu oraz danych z mapy 14

3. Opis platformy sprzętowej 15

3.1. Specyfikacja sprzętowa platformy Zybo 15

3.1.1. Logika reprogramowalna 15

3.1.2. System procesorowy 16

4. Implementacja modelu programowego 19

4.1. Binaryzacja 19

4.2. ROI 20

4.3. LMPS 22

4.4. Podział obrazu i wyznaczenie wykrytych linii 23

5. Implementacja sprzętowa 25

5.1. Binaryzacja 25

5.2. Wyznaczanie ROI 26

5.3. LMPS 27

5.4. Podział obrazu 29

6. Testy 33

7. Podsumowanie 35

7.1. Wnioski 35

1. Wprowadzenie

Samochód autonomiczny jest to pojazd sterowany przez system komputerowy, w oparciu o dane z wielu różnorodnych czujników. Według klasyfikacji SAE (ang. *Society of Automotive Engineers* – związek inżynierów zajmujących się motoryzacją) rozróżnia się pięć poziomów autonomizacji. Ostatni z nich zakłada, że w sterowanie samochodu nie ma żadnej ingerencji ze strony człowieka. Pozostałe cztery odnoszą się do różnego stopnia zaawansowania systemu ADAS (ang.*Advanced Driving Assistance System* – zaawansowany system wspomagania kierowcy).

Zadaniem systemów ADAS jest wspomaganie człowieka w czasie jazdy tj. wykrywanie i ostrzeganie o sytuacjach niebezpiecznych dla kierowcy i otoczenia. W sytuacjach krytycznych komputer pokładowy jest w stanie przejąć kontrolę nad autem i wykonać manewr zapobiegający wypadkowi. Do postrzegania otoczenia wykorzystuje się czujniki takie jak radar, lidar, GPS, kamery, IMU (ang. *inertial measurement unit*) oraz czujniki ultradźwiękowe.

Do podstawowych systemów ADAS zalicza się:

- asystent parkowania - czujniki ultradźwiękowe rozmieszczone wokół samochodu umożliwiają mierzenie odległości pojazdu od innych przeszkód ułatwiając manewrowanie przy parkowaniu,
- monitorowanie martwego pola - kamery rozmieszczone bo bokach samochodu zbierają informacje z przestrzeni, których kierowca nie jest w stanie kontrolować wykorzystując z pomocy lusterek,
- ostrzeganie o kolizji przedniej, tylnej - tempomaty znajdujące się z przodu i z tyłu auta mierzą prędkość z jaką pojazd zbliża się do przeszkody. Gdy jest zbyt duża i istnieje ryzyko kolizji, system uruchomia alarm w postaci wizualnej lub dźwiękowej, a nawet dostosuje prędkość w celu uniknięcia kolizji,
- detekcja i rozpoznawanie znaków drogowych - system wykrywa i informuje kierowcę o znakach i uruchamia sygnał alarmowy w przypadku niedostosowania się do nich,
- detekcja samochodów i pieszych - system przy pomocy kamer wykrywa ludzi i pojazdy znajdujące się w najbliższym otoczeniu oraz wyznacza ich przewidywaną ścieżkę ruchu. Jeśli istnieje ryzyko kolizji zostanie uruchomiony alarm. A w sytuacji krytycznej auto zahamuje,
- system ostrzegania przed opuszczeniem pasa ruchu (ang. *Lane Departure Warning System*) – jest to mechanizm opierający się na detekcji linii drogowych i rozpoznawaniu jezdni. W sytuacji,

w której samochód zaczyna zmieniać pas ruchu bez uprzedno włączonego odpowiedniego kierunkowskazu system wysyła ostrzeżenie do kierowcy. Może to być sygnalizacja w postaci wizualnej, dźwiękowej lub wibracji. Wyróżnia się też bardziej zaawansowane wersje oprogramowania, w których system przejmuje kontrolę nad układem kierowniczym i nie pozwala na zmianę pasa ruchu przez pojazd lub kieruje nim na środek pasa. Tego typu systemy zostały zaprojektowane w celu zminimalizowania liczby wypadków drogowych wynikających z błędów kierowców powodowanych między innymi przez zmęczenie bądź utrate koncentracji.

W ostanim czasie można zauważyć spory rozwój w dziedzinie pojazdów bezzałogowych. Jest to spowodowane widocznym rozwojem w dziedzinie sensoryki. oraz jednostek obliczeniowych w dziedzinie CPU (ang. *Central Processing Units*) i GPGPU (ang. *General Purpose computing on Graphics Processing Units*). Dodatkowo kolejny firmy motoryzacyjne starają się osiagac coraz to lepsze rozwiązania w celu wyróznienia się na tle konkurancji i zdobycia większego rozgłosu. Na horyzoncie już widać pierwsze przymiarki do oficialnego startu sezonu wyścigów samochodów autonomicznych Roborace [**roborace**]. Przez długi czas wyścigi samochodowe były poligonem doświadczalnym dla nowych technologii, które następnie mogły trafić do samochodów drogowych. Aktualnie większość elektronicznych wspomagaczy kierowcy zostało zakazanych w Formule 1 i wpływ wyscigów uległ osłabieniu. Ale skoro w przyszłości mamy poruszać się pojazdami autonomicznymi, Roborace może pomóc w opracowywaniu nowej technoligii takich pojazdów.

Jednym z możliwych rozwiązań efektywnego rozwoju oprogramowania są układy Zynq SoC. Dzięki możliwości ich reprogramowania są wygodnym zamiennikiem układów ASIC (ang. *Application-Specific Integrated Circuit* - układ scalony specyficzny dla aplikacji) [**ASIC_zynq**]. Rekonfigurowalność układu zapewnia szybszy i tańszy rozwój oprogramowania oraz pozwala na naprawę błędów w oprogramowaniu bez konieczności tworzenia nowego układu. Firma Xilinx oferuje specialistyczne układy z myślą o systemach ADAS. Urządzenia takie jak XA Zynq^{TM}-7000 SoCs idealnie nadają się do wysokich wymagań obliczeniowych zaawansowanych systemów wspomagania kierowcy (ADAS) [**xilinx**].

1.1. Cele pracy

W niniejszej pracy podjęto temat detekcji jezdni dla potrzeb pojazdów autonomicznych. Celem pracy jest napisanie modelu programowego w języku Matlab algorytmów przetwarzających nagrany obraz. Rezultatem powinien być film, na który naniesione są krzywe reprezentujące detekcje linii drogowych. Kolejną częścią pracy jest przeprowadzenie implementacji sprzętowej algorytmów na układzie Zynq SoC(ang. *System On Chip*) przy użyciu języka opisu sprzętu Verilog.

1.2. Układ pracy

W rozdziale drugim przedstawiono przegląd metod przetwarzania obrazu. W kolejnym rozdziale omówiono platformę Zybo Zynq SoC. Rozdział czwarty odnosi się do implementacji programowej, a piąty do sprzętowej. Szósty rozdział zawiera w sobie testy. Siódmy podsumowanie.

2. Przegląd algorytmów z literatury

2.1. Wstępne przetwarzanie obrazu

Pierwszym etapem detekcji pasów ruchu drogowego jest konwersja obrazu kolorowego w formacie RGB (ang. *Red, Green, Blue* – czerwony, zielony, niebieski) na obraz binarny. Proces ten ma na celu aby wstępnie otrzymać cechy określające dany obraz oraz ułatwić dalsze jego przetwarzanie. W celu ułatwienia konwersji z kolorowego na binarny, możemy przekonwertować wstępnie obraz do formatu w skali szarości.

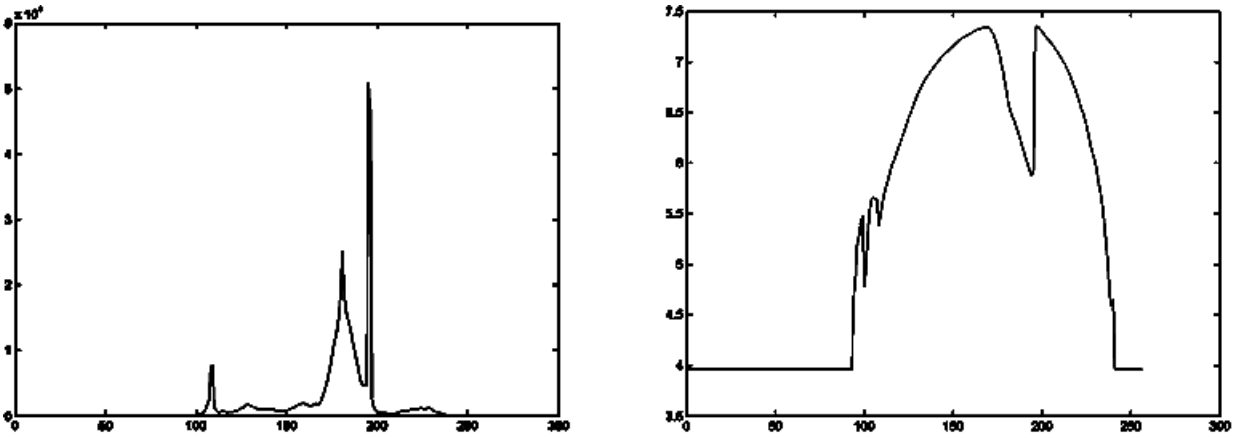
Każdy piksel w formacie RGB składa się z trzech kanałów, każdy przyjmuje wartośći z przedziału od 0 do 255, gdzie 0 oznacza brak nasycenia danego koloru, a 255 maksymalne nasycenie. W skali szarości na jeden piksel przypada tylko jeden kanał przyjmujący wartości od 0 do 255. Dla obrazu binarnego na jeden piksel również przypada tylko jeden kanał, ale może przyjmowac wartości o lub 1, gdzie 0 to brak nasycenia kolorów, a 1 to pełne nasycenie kolorów. Obraz w skali szarości z obrazu w formacie RGB można otrzymać, np. wyznaczając średnią wartość trzech kanałów dla każdego z pikseli [**4**], lub poprzez wykorzystanie składowej L z formatu HSL (ang. *Hue, Saturation, Lightness* – odcień, saturacja, jasność) [**reichenbach_comparison**].

Jednym z metod konwersji obrazu ze skali szarości do binarnego, który zaproponowano w pracy [**4**], jest zwiększenie kontrastu dzięki wykorzystaniu rozciągania histogramu. Następnie korzystając z filtru Sobel'a uwydatnia się krawędzie obiektów, które ma na celu ułatwienie późniejszej detekcji linii pasa ruchu drogowego. Kolejno wykorzystuje się progowanie polegające na przypisywaniu wartości 0 lub 1 do piksela w zależności od tego czy jego wartość jest mniejsza od wartości progu, czy nie. Operator Sobel'a jest zasadniczo operatorem różniczkowania dyskretnego, zwraca pochodne pierunkowe obrazu w ośmiu kierunkach co 45 stopni [**3**], [**sobel**].

Innym podejściem [**reichenbach_comparison**] jest wykorzystanie algorytmu Canny Edge Detector w miejscu filtra Sobel'a. Canny Edge łączy w sobie filtr sobela i zdefiniowaną histerezę [**cany**]. Jeśli wartośc gradientu G (2.1) piksela jest powyżej ustalonego progu górnego, zaliczany jest do zbioru krawędzi. Gdy jest poniżej ustalonego progu dolnego piksel jest odrzucany. Gdy znajduje się pomiędzy oboma wspomnianymi progami, piksel zostanie zaliczony do zbioru krawędzi jeśli sąsiaduje z pikselem zaklasyfikowanym jako krawędź.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}, G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

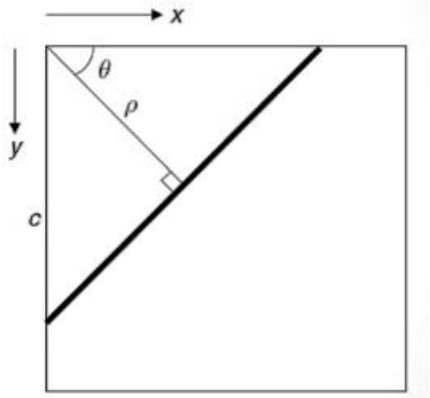
Próg binaryzacji można również dobrać samodzielnie, np. przyjmując, że jego wartość jest równa połowie wartości maksymalnej jaką może mieć piksel [**1**]. Innym podejściem, zaprezentowanym w artykule [**2**], jest wykorzystanie funkcji entropii lub histogramu. Histogram obrazu jest sposobem reprezentacji rozkładu wartości piksleli, z których składa się obraz. Może przyjmować formę interpolowanego wykresu, gdzie pozioma oś układu współrzędnych odpowiada za wartość piksela. A pionowa oś reprezentuje liczebność punktów o wartości równej wartości argumentu znajdującego się na poziomej osi [**histogram**]. Na rysunku 2.1 przedstawiono zestawienie wykresu danego histogramu oraz wykresu funkcji entropii. Funkcje te mają charakterystyczną cechę wspólną. Argument dla drugiej największej wartości funkcji entropii jest równy argumentowi dla jakiego histogram przyjmuje wartość maksymalną. Próg binaryzujący znajduje się pomiędzy dwoma największymi wierzchołkami funkcji entropii.



Rys. 2.1. Porównanie histogramu i funkcji entropii. Źródło: [**2**]

2.2. Wyznaczanie obszaru zainteresowania

Pomimo występowania rożnorodnych pasów ruchu w rzeczywistości, część ich cech charakterystycznych pozostaje bez zmian. W wyniku zjawiska perspektywy, czym obiekt jest dalej od kamery tym staje się mniejszy, linie drogowe bliżej kamery, w dolnej części obrazka, są grubsze od tych znajdujących się dalej, w wyższej części obrazu. To samo dotyczy całego pasa ruchu. W dolnym obszarze obrazu występuje większy stosunek obiektu pierwszoplanowego, którym jest pas ruchu w stosunku do tła, reprezentowanego przez pobocze i obszar poza drogą. Dodatkowo linie znajdujące się bliżej kamery wakazują na obrazie mniejszą tendencję do nagłej zmiany swojego położenia. Natomiast nawet lekkie zakręty powodują nagłe przesunięcia się obiektów w górnej części obrazu. Podane zależności są charakterystyczne dla obrazów przedstawiających fragment ulicy wzdłuż której porusza się samochód. Przyczyniło się to



Rys. 2.2. Graficzne przedstawienie zależności współrzędnych x, y i θ, ρ. Źródło: [**hough_rotheta**]

powstanie różnych konceptów wyznaczania ROI (ang. *Region Of Interest* – obszar zainteresowania). W pracach [**2**] i [**4**] wykorzystano określenie statycznego obszaru zainteresowania. ROI jest w kształcie trapezu zwężającego się w kierunku górnej części obrazu [**4**].

Wyznaczanie obszaru zainteresowania na obrazie może odbywać w sposób statyczny albo dynamiczny. Wyznaczanie statyczne polega na predefiniowanych ograniczeniach, na podstawie których otrzymywany jest obraz przetworzony. Dynamiczne odnosi się do sytuacji, w której wyznaczony obszar zainteresowania zależny jest od danych wejściowych. W artykule [**vanishing_point**] przedstawiono podejście polegające na wyznaczaniu obszaru zainteresowania na podstawie określonego punktu na horyzoncie. Jest to punkt przecięcia się dwóch prostych, powstałych w skutek przedłużenia linii określających krawędzie drogi. Celem wyznaczenia ROI jest pozbycie się jak największej części obrazu, na której nie znajduje się pas ruchu. Ma to na celu zmniejszenie wpływu tła na działanie algorytmów do detekcji linii.

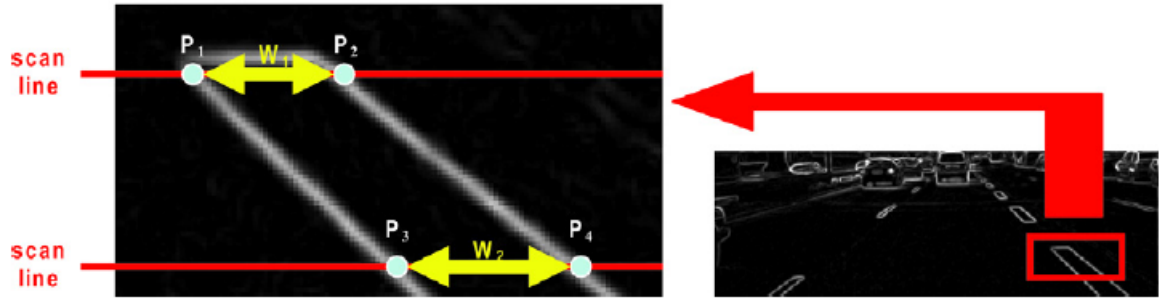
2.3. Wyznaczanie linii drogowych

W tym etapie skupimy się na detekcji pasów ruchu poprzez wyznaczenie linii na podstawie obrazu binarnego. W artykule [**reichenbach_comparison**] wykorzystano w tym celu transformatę Hough'a. Jest to metoda wykrywania prostych w widzeniu komputerowym [**hough**].

Prostą znajdującą się na obrazie o współrzędnych kartezjańskich x, y można zapisać jako punkt w układzie o współrzędnych θ, ρ 2.2 (przestrzeń Hough'a) spełniający zależność (2.2), gdzie θ - kąt nachylenia, ρ - odległość od początku układu współrzędnych.

$$x\cos(\theta) + y\sin(\theta) = \rho \tag{2.2}$$

W celu wyznaczenia pełnego obrazu w przestrzeni Hough'a należy przeiterować po całym przetwarzanym obrazie i dla każdego piksela o wartości 1 (białego) zaznaczyć w układzie θ, ρ wszystkie punkty odpowiadające prostym we współrzędnych x, y jakie mogą przechodzić przez ten piksel.



Rys. 2.3. Zdjęcie przedstawiająca sprawdzanie odległości w poziomie pomiędzy dwoma białymi pikselami. Źródło: [**4**]

Zakres grubości pasów znajdujących się przykładowo na autostradzie jest ściśle określony przepisami ruchu drogowego. Ta zależność została wykorzystana w pracy [**4**], gdzie użyto filtru wykrywającego linie ruchu drogowego. Działa on na zasadzie sprawdzania odległości w poziomie pomiędzy dwoma białymi pikselami. Jeśli mieści się on w ustalonej normie oznacza to, że punkty leżą na linii. Zakres dobierany jest w zależności od badanej części obrazu, z zachowaniem właściwości perspektywy 2.3.

2.4. Fuzja obrazu oraz danych z mapy

Postrzeganie otoczenia jest jednym z głównych wyzwań dla systemów w pojazdach autonomicznych.

Przednia kamera w pojeździe dostarcza informacji o kluczowych elementach drogi takich jak oznaczenia pasa ruchu i jego granice. Poprawna detekcja przestrzeni, po której auto może się poruszać jest poważnym wyzwaniem ze względu na występowanie wielu pasów ruchu, przecinania się linii.

W artykule [**hdmap**] podjęto się przeprowadzenia fuzji detekcji pasa ruchu drogowego oraz informacji z HD (ang. *High Definition* – wysoka rozdzielczość) mapy. Informacja o obecnym położeniu auta zapewniła możliwość wyekstrahowania z mapy danych o kształcie drogi na jakiej się znajdował.

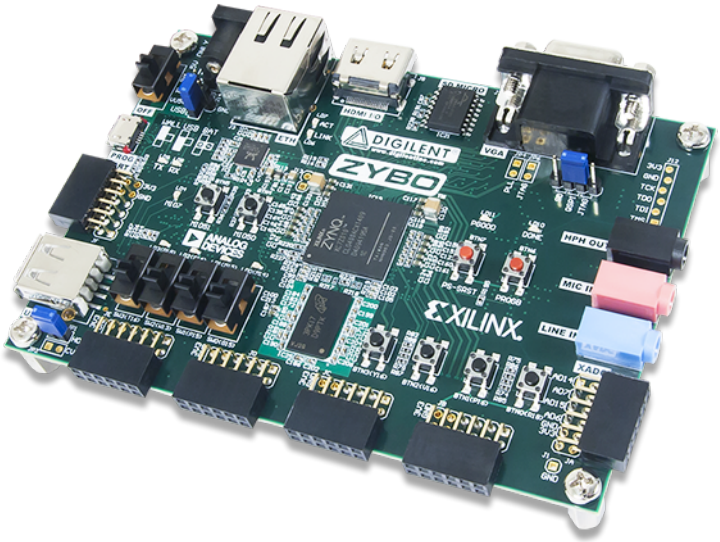
Zaproponowane podejście ma znaczącą przewagę nad korzystaniem z samej kamery, ponieważ jest w stanie rekompensować skutki takich błędów jak:

1. zmienne warunki oświetlenia, obraz może zawierać nieregularne cienie lub przebłyski,
2. brak widoczności linii drogowych, naruszona tekstura krawędzi,
3. nieregularna kolorystycznie nawierzchnia drogowa lub obiekty ją przysłaniające,
4. kamera może nie uchwycić całego obszaru, który powinien zostać wykryty, ze względu na krzywiznę drogi.

W sytuacji, gdy z obrazu nie da się uzyskać wystarczających informacji przykładowo o nagłym zakręcie na drodze, dane z mapy mogą okazać się nieocenione.

3. Opis platformy sprzętowej

3.1. Specyfikacja sprzętowa platformy Zybo



Rys. 3.1. Platforma Zybo z układem Zynq SoC. Źródło: [**zybo_img**]

Do implementacji sprzętowej została wykorzystana karta Zybo firmy Digilent 3.1 wyposażona w układ Zynq SoC (ang. *System on Chip*). W dokumencie [**zybo_description**] znajduje się opis budowy układu. Jest on określany mianem "heterogeniczny", ponieważ zawiera dwa rodzaje zasobów sprzętowych: dwurdzeniowy procesor ARM Cortex-A9 oraz układ FPGA(ang.*Field Programmable Gate Array* – bezpośrednio programowalna macierz bramek), czyli logikę reprogramowalną.

3.1.1. Logika reprogramowalna

Część reprogramowalna układu Zynq na karcie Zybo jest oparta o logikę serii Artix-7 firmy Xilinx. Podstawowym elementem z którego zbudowane jest FPGA, to blok CLB (ang. *Configurable Logic Block*). Składa się on z dwóch Slice'ów połączonych z matrycą przełączeń (ang. *Switch Matrix*). W układach Zynq występują dwa rodzaje elementów Slice, są to SliceL i SliceM. Slice typu M składa się z:

- generatora funkcyjnego (4 sztuki) – został on zrealizowany przy pomocy układów LUT (ang. *Look-Up Table*). Posiada 6 wejść i 2 wyjścia. Może także zostać skonfigurowany jako synchroniczna pamięć RAM lub 32 bitowy rejestr przesuwny wykorzystywany w liniach opóźniających,
- przerzutnika typu D (FF – ang. *Flip-Flop*) – slice zawiera 8 sztuk, przy czym 4 mogą zostać skonfigurowane jako zatrzask (ang. *latch*),
- szybkiej logiki przeniesienia,
- multiplekserów.

Do pozostałych zasobów dostępnych w układach FPGA serii Artix-7 należą:

- CMT (ang. *Clock Managment Tiles*) – bloki umożliwiające zarządzanie sygnałem zegarowym, generowanie różnych częstotliwości zegara, równomierną propagację sygnału, tłumienie zjawiska zakłócenia fazy zegara,
- Block RAM (BRAM) – blokowa dwuportowa pamięć RAM o rozmiarze 36Kb (na blok). Może zostać skonfigurowana jako moduł FIFO (ang. *First In First Out*),
- GTP/GTX Transceivers – moduły umożliwiające transmisję szeregową z prędkością do 12,5 Gb/s (GTX) i 6,25 (GTP),
- DSP48A1 – moduł zawierający mnożarkę 25x18 bitów oraz akumulator 48 bitowy. Liczba modułów zależy od rozmiaru układu i zawiera się w przedziale od 66 do 2020,
- Select I/O – banki zasobów wejścia/wyjścia, których liczba zawiera się w przedziale od 100 do 400 końcówek podłączonych do części FPGA.

3.1.2. System procesorowy

ARM Cortex-A9 MPCore jest 32 bitowym procesorem firmy ARM Holdings z zaimplementowaną archtekturą ARMv7-A. Zawiera od 1 do 4 rdzeni. Płytka Zybo Zynq SoC jest wyposażona w wersje procesora dwurdzeniowego. Rozkazy procesorów ARM są tak skonstruowane, aby wykonywały jedną określoną operację w jednym cyklu maszynowym. Kluczowymi cechami rdzenia Cortex-A9 są [**armCortex**]:

- NEON SIMD (ang. *single instruction, multiple data* – pojedyncza instrukcja, wiele danych) opcjonalne rozszerzenie zestawu instrukcji do 16 operacji na instrukcję,
- rozszerzenia zabezpieczeń TrustZone,
- jednostka zmiennoprzecinkowa VFPv3 dwukrotnie przewyższająca wydajność swojego poprzednika ARM FPUs,
- kodowanie zestawu instrukcji Thumb-2 zmniejsza rozmiar programów, co poprawia wydajność,

- program Trace Macrocell i CoreSight Design Kit do nieinwazyjnego śledzenia wykonywania instrukcji,
- przetwarzanie wielordzeniowe,
- kontroler pamięci podręcznej L2 (0-4 MB),
- kontroler pamięci statycznej, dynamicznej i bezpośredniej.

4. Implementacja modelu programowego

W celu przetestowania wybranych algorytmów zdecydowano się wykorzystać tzw. model programowy tworzonej aplikacji. Do zaimplementowania prototypu wybrano komputer z procesorem Intel Core i7 6700HQ. Posłużono się językiem programowania Matlab.

Z racji tego, że algorytmy miały zostać zaimplementowane z myślą o późniejszych przeniesieniu ich na część reprogramowalną platformy Zynq SoC. Na jeden takt zegara otrzymuje się jeden piksel (co wynika ze sposobu obsługi sygnału wideo). Z każdym kolejnym taktem otrzymuje się kolejne piksele. W celu przeprowadzenie operacji kontekstowych należy korzystając z linii opóźniających zrobić tak, aby w danym takcie zegara mieć dostęp do wszystkich pikseli tworzących kontekst. Takie podejście powoduje, że niektóre algorytmy nie są możliwe do zaimplementowania. Należy zrezygnować ze wszystkich algorytmów, w których każdy kolejny krok algorytmu jest zależny od danych wejściowych, np. segmentacja przez rozrost. Z myślą o oszczędzaniu zasobów, w działaniach dzielenia starano się sprowadzać mianownik do potęgi liczby 2. Taką operację można wtedy wykonać przy wykorzystaniu dużo mniejszej ilości zasobów układu FPGA.

Aplikacja ma na celu detekcję punktów reprezentujących prawą i lewą linie drogową.

4.1. Binaryzacja

W celu otrzymania obrazu zbinaryzowanego z obrazu w formacie RGB zdecydowano się wykorzystać binaryzację stałym progiem [**1**]. Obraz wejściowy najpierw konwertowany jest w obraz w skali szarości. Otrzymano wykorzystując składową L z formatu HSL. Wzór na przekształcenie składowych RGB w składową L (4.1).

$$\begin{aligned} C_{max} &= max(R,\, G,\, B) \\ C_{min} &= min(R,\, G,\, B) \\ L &= (C_{max} + C_{min}) \,/\, 2 \end{aligned} \tag{4.1}$$

Po otrzymaniu obrazu w skali szarości jest on binaryzowany progiem o wartości 192, jest to około 3/4 maksymalnej wartości jaką może posiadać piksel. Podany próbsprawdzał się wystarczająco w celu

oddzielenia pasów od jezdni. Rysunek 4.1 przedstawia obraz wejściowy w kolorze. Rysunek 4.2 obraz w skali szarości. Obraz 4.3 prezentuje obszar wyjsciowy, zbinaryzowany podanym progiem.

Rys. 4.1. Obraz wejsciowy przed przekształceniami

Rys. 4.2. Obraz w skali szarości

4.2. ROI

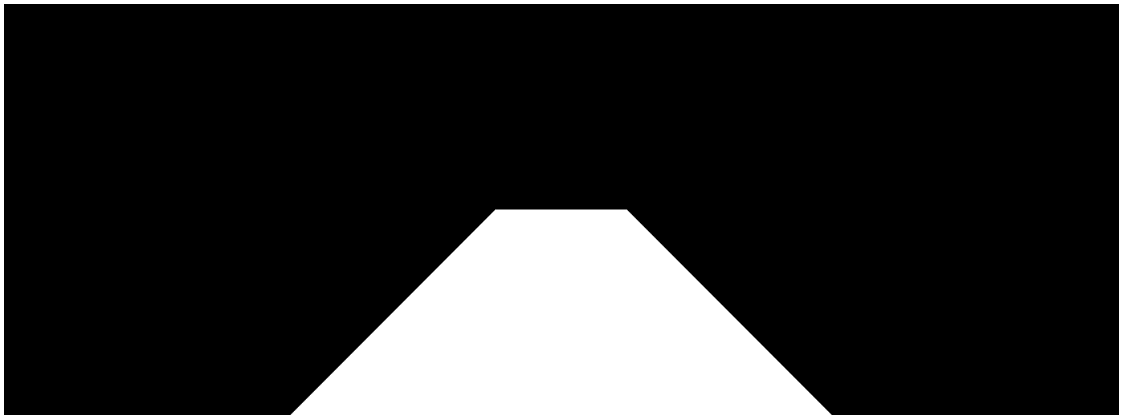
Celem wyznaczenia ROI jest pozbycie się części danych obrazu, które mogą utrudniać dalszą detekcję pasów ruchu drogowego. Obszar zainteresowanie został wyznaczony statycznie przy pomocy dobranych ograniczeń. Aby piksel nie został odfiltrowany, czyli jego wartość ustawniona na 0, jego współrzędne muszą spełniać warunki opisane wzorami (4.2). Przy czym x oznacza numer kolumny, w której znajduje się piksel, a y numer wiersza. $Width$ oznacza szerokość obrazu, a $High$ wysokość obrazka. Ograńiczenia sotały dobrane w taki sposób, aby odfiltrować jak najwięcej tła obrazu i pozostawić praktycznie tylko obiekt pierwszoplanowy, którym jest jezdnia. Ze względu na znane standardy szerokości pasów ruchu drogowego, zdecydowano się statycznie dobierać ograniczenia.

Rys. 4.3. Obraz zbinaryzowany przed wyznaczeniem ROI w modelu programowym

$$\begin{aligned} x - 3/8\,Width &< y \\ -x + 1/2\,Width &< y \\ y &> 5/8\,High \end{aligned} \tag{4.2}$$

Algorytm wyznaczania ROI należy przeprowadzić po wcześniejszym zbinaryzowaniu obrazu. Przy poprawnej binaryzacji linie drogowe zostają zaliczane jako obiekt pierwszoplanowy (białe piksele). Za to jezdnia jest zaliczana jako obiekt drugoplanowy (czarne piksele). Przeprowadzenie kolejno metody wyznaczania ROI powoduje usunięcie obszaru znajdującego się poza pasem ruchu. Rysunek 4.3 przedstawia obraz wejściowy, zbinaryzowane zdjęcie drogi. Rysunek 4.4 obraz wyjściowy, obraz po wyznaczeniu ROI. Rysunek 4.5 prezentuje obszar (czarny), który zostaje usunięty z obrazu w wyniku wyznaczania ROI.

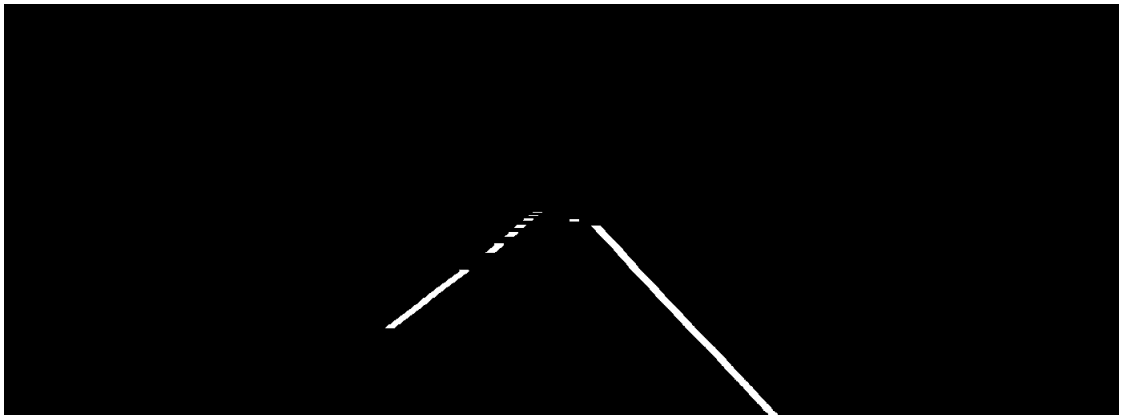
Rys. 4.4. Obraz wyjściowy wyznaczania ROI w modelu programowym



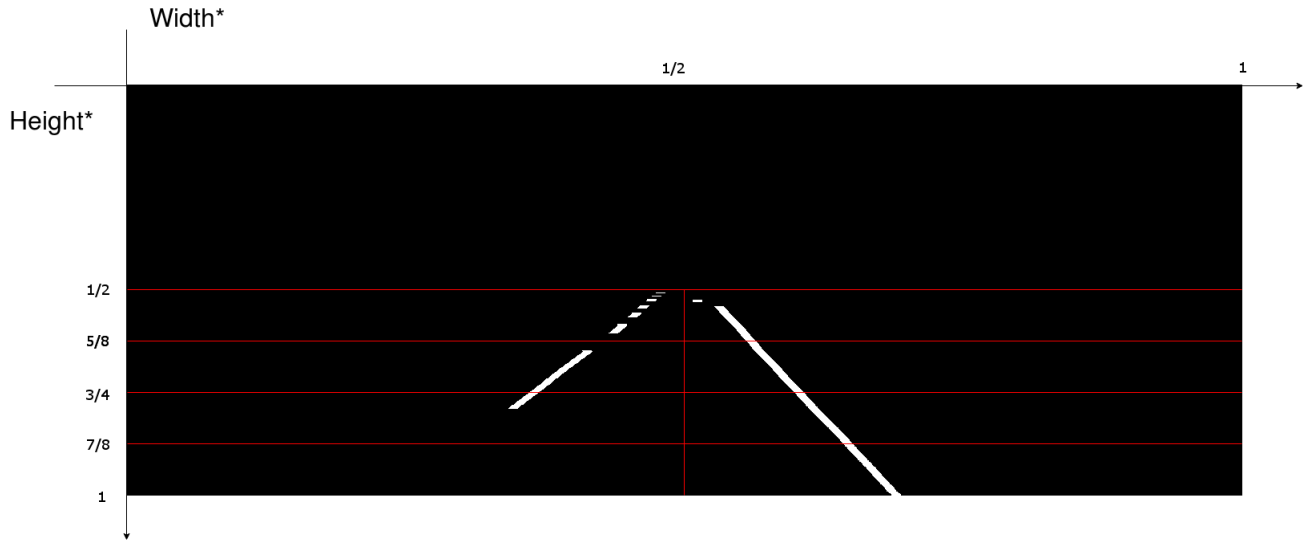
Rys. 4.5. Obraz przedstawiający, która cześć obrazu nie została usunięta (białe piksele) podczas wyznaczania ROI.

4.3. LMPS

LMPS (ang. *Lane Marker Pattern Search* – wyszukiwanie wzorców pasów ruchu) zaimplementowano tak, aby obiekty nienależące do zbioru linii drogowych zostały odfiltrowane. Takimi elementami są między innymi pasy przejścia dla pieszych oraz przedmioty znajdujące się na ulicy. Przed użyciem filtru LMPS, na obrazku przprowadzamy operację zamknięcia w celu uniknięcia błędnego nie rozpoznania pasów wyniku ich zużycia bądź zabrudzenia. Pozbywamy się czarnych pikseli z wnętrza obiektów. Na rysunkach 4.4 i 4.6 został przedstawiony efekt odfiltrowania pasów. Na obrazie wynikowym znajdują białe piksele odnoszą się do wykrytych linii drogowych. Algorytm działa na zasadzie sprawdzania grubości w poziomie białych części obrazu wejściowego. Przyjęto, że linie drogowe mają konkretną grubość oraz wzięto pod uwagę wpływ perspektywy na zniekształcenie obiektów. Dzięki temu progi określające szerokość linii drogowej przedstawionej na obrazie zmieniają się liniowo w zależności od obecnego indeksu wiersza piksela.



Rys. 4.6. Obraz 4.3 przetworzony przez filtr LMPS



Rys. 4.7. Zdjęcie przedstawiające wydzielone obszary, w których szukane są środki ciężkości.

4.4. Podział obrazu i wyznaczenie wykrytych linii

Kolejnym etapem detekcji pasa ruchu Obraz został podzielony na osiem części. Podział przedstawiony jest na grafice 4.7. W każdym z obszarów został wyznaczony środek ciężkości obliczony przy pomocą wzorów (4.3) - (4.7). Bazują one na momentach geometrycznych figur. Z czego figura jest rozumiana jako zbiór pikseli o takiej sam wartości. Uzyskane punkty stanowią podstawę do wyznaczenia krzywych reprezentujących wykryte linie drogowe. Punkty wykryte w obszarach z prawej strony zostały wykorzystane do reprezentacji prawej linii drogowej. Analogicznie postąpiono z punktami wykrytymi po lewej stronie obrazka, które wykorzystamy do otrzymania krzywej reprezentującą lewą linię drogową/ Gdzie $Width$ – szerokość obrazu w pikselach, $Height$ – Wysokość obrazu

$$m_{00} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{ij} \tag{4.3}$$

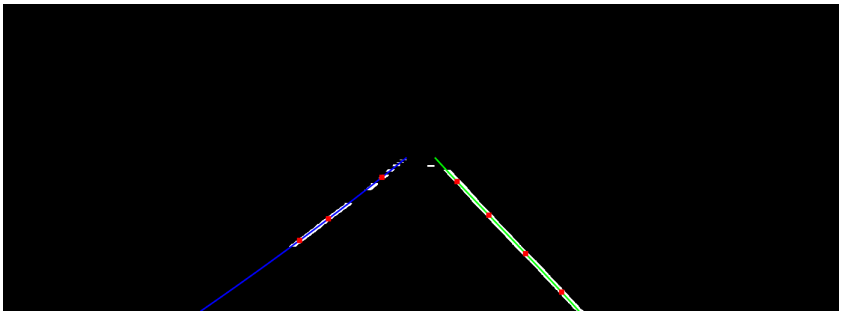
$$m_{10} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} i \cdot x_{ij} \tag{4.4}$$

$$m_{01} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} j \cdot x_{ij} \tag{4.5}$$

$$x_{sc} = \frac{m_{10}}{m_{00}} \tag{4.6}$$

$$y_{sc} = \frac{m_{01}}{m_{00}} \tag{4.7}$$

Gdzie: m_{00}, m_{10}, m_{01} – zmienne pomocnicze, i – indeks wiersza piksela, j – indeks kolumny obecnego piksela, i – indeks kolumny piksela, x_{ij} – piksel o współrzędnych i, j, N – szerokość obrazu, M – wysokość obrazu, x_{sc} – współrzędna wierszowa punktu reprezentującego środek ciężkości, y_{sc} - współrzędna kolumnowa punktu reprezentującego środek ciężkości.



Rys. 4.8. Obraz przedstawiający wyznaczone środki ciężkości, reprezentowane są przez czerwone punkty. Oraz łączące je proste

Na rysunku 4.8 przedstawiono punkty oznaczające wyznaczone środki ciężkości, oraz łączącą ją krzywą. Znalezione czerwone punkty leżą na liniach otrzymanych z filtru LMPS. Dopasowane krzywe pokrywają się z liniami.

5. Implementacja sprzętowa

5.1. Binaryzacja

W celu ekstrakcji obrazu zbinaryzowanego z obrazu w formacie RGB zdecydowano się wykorzystać binaryzację ze stałym progiem. Obraz wejściowy najpierw konwertowany jest do obrazu w odcieniach szarości. W formacie RGB na każdy piksel składają się trzy kanały, każdy o wartości z przedziału 0-255 zapisane na 8 bitach. Największe i najmniejsze wartości z kanałów są dodawane i dzielone przez 2. Operacja ta zostaje przeprowadzona dla każdego piksela. Zdecydowano się na taką metodę, ponieważ dzielenie przez liczbę, która jest wielokrotnością 2 może zostać zrealizowane poprzez przesunięcie bitowe. Jest ono znacznie szybsze niż użycie dzielarki. W porównaniu do wyliczania średniej z trzech kanałów, zachowujemy pełny zakres od 0 do 255 i nie musimy wykorzystywać dzielarki. Obrazem wynikowym jest obraz w odcieniach szarości, który kolejno jest poddawany binaryzacji.
Rysunki 5.1, 5.2 przedstawiają efekt symulacji tego algorytmu dla obrazu o wymiarach 348x128.

Rys. 5.1. Obraz wejściowy do symulacji sprzętowej binaryzacji

Rys. 5.2. Obraz wyjściowy z symulacji sprzętowej binaryzacji

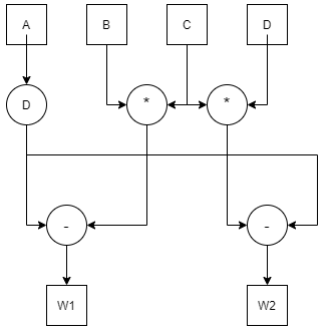
5.2. Wyznaczanie ROI

Efekty symulacji implementacji sprzętowej wyznaczania obszaru zainteresowania zostały przedstawione na rysunku 5.4.

Opis schematu 5.3 implementacji sprzętowej algorytmu do wyznaczenia współczynników wykorzystywanych w 5.5(w nawiasach podano liczbę bitów potrzebną do zapisania danej wartości albo liczbę taktów zegara potrzebnych do przeprowadzania danej operacji):

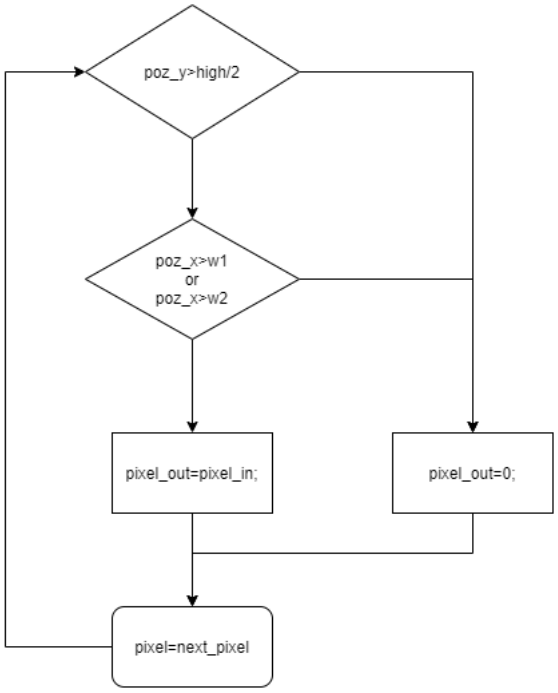
- A - współrzędna x piksela, poz_x (12 bity),
- B - stała wartość równa 3 (4 bity),
- C - jedna ósma szerokości obrazu (12 bitów),
- D - stała wartość równa 5 (4 bity),
- W1 - współczynnik 1 (16 bitów),
- W2 - współczynnik 2 (16 bitów),
- (-) - odejmowanie (1 takt zegara),

Rezultaty odejmowania W1 i W2 są kolejno wykorzystywane w logice asynchronicznej 5.5



Rys. 5.3. Schemat algorytmu do wyznaczania współczynników dla ROI.

Rys. 5.4. Obraz wyjściowy z symulacji sprzętowej algorytmu do wyznaczania ROI.

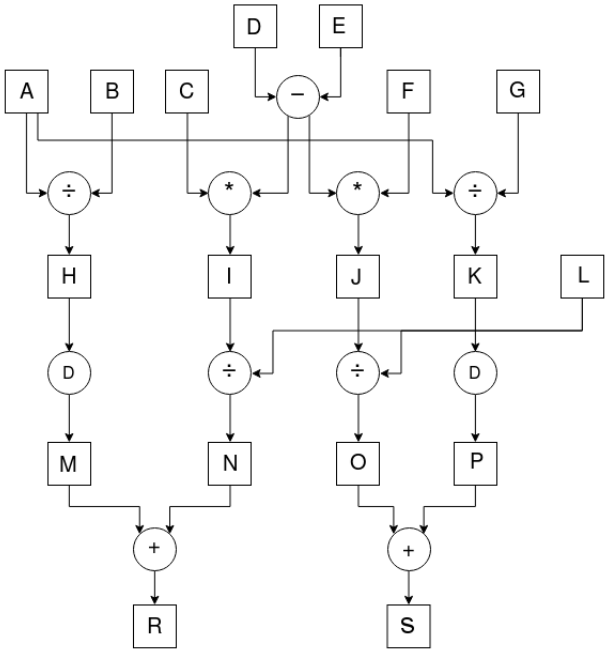


Rys. 5.5. Schemat algorytmu do wyznaczania ROI. Wartości współczynnik1 i współczynnik2 są rezultatami 5.3

5.3. LMPS

Opis schematu implementacji sprzętowej algorytmu do wyznaczenia ograniczeń górnych i dolnych 5.6:

- A - szerokość obrazu (32 bity),
- B - stała zależna od grubości linii drogowych, wynosi 400 (20 bitów),
- C - stała zależna od grubości linii drogowych, wynosi 20 (8 bitów),
- D - indeks wiersza obecnego piksela (24 bity),
- E - połowa szerokości obrazu (24 bity),
- F - stała zależna od grubości linii drogowych, wynosi 62 (8bitów),
- G - stała zależna od grubości linii drogowych, wynosi 90 (20 bitów),
- L - wysokość obrazu (20 bitów),
- H, K, N, O - rezultaty dzielenia (32 bity),
- I, J - rezultaty mnożenia (32 bity),
- M, P - sygnały wyjściowe z linii opóźniających (32 bity),



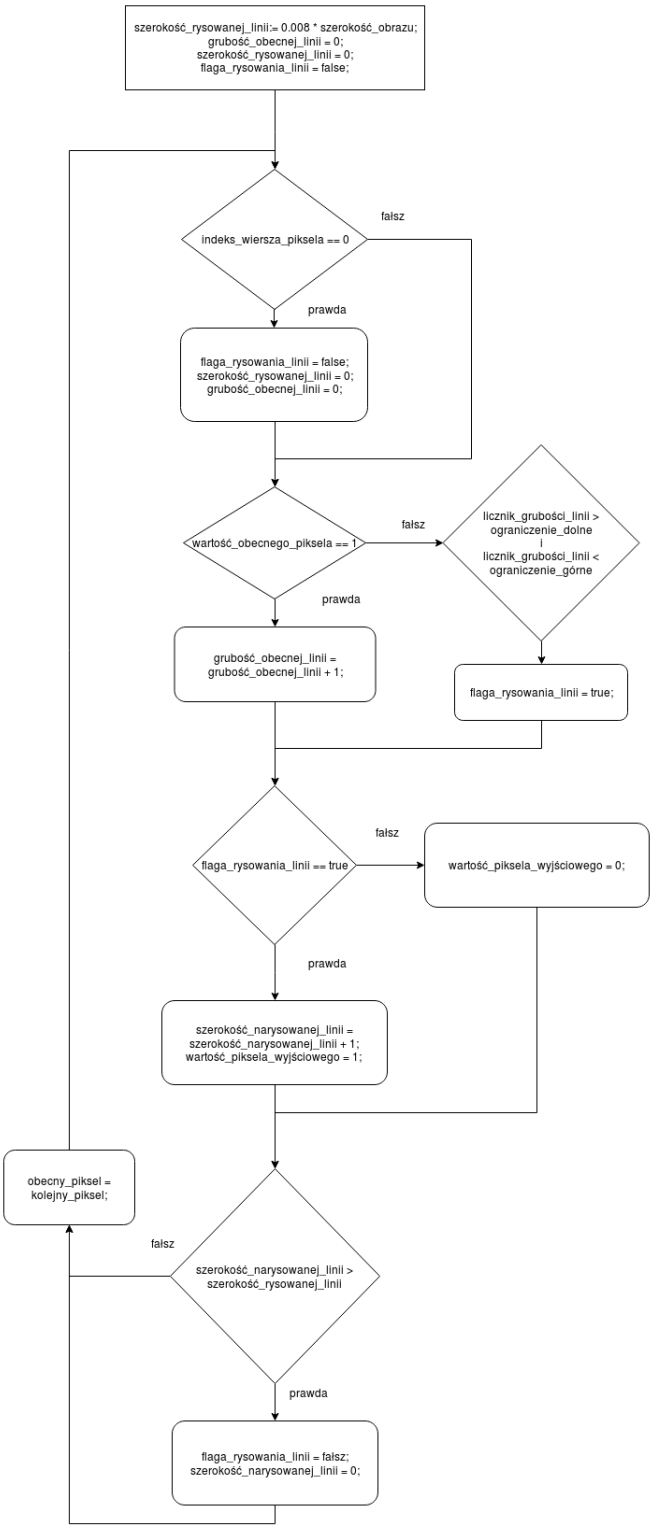
Rys. 5.6. Schemat blokowy implementacji sprzętowej algorytmu do wyznaczenia ograniczeń górnych i dolnych dla filtru LMPS

- R - próg ograniczający dolny (32 bity),
- S - próg ograniczający górny (32 bity),
- (÷) - dzielenie (4 takty zegara),
- (*) - mnożenie (1 takt zegara),
- (D) - linia opóźniająca (1 takt zegara),
- (+) - dodawanie (1 takt zegara).
- (-) - odejmowanie (1 takt zegara).

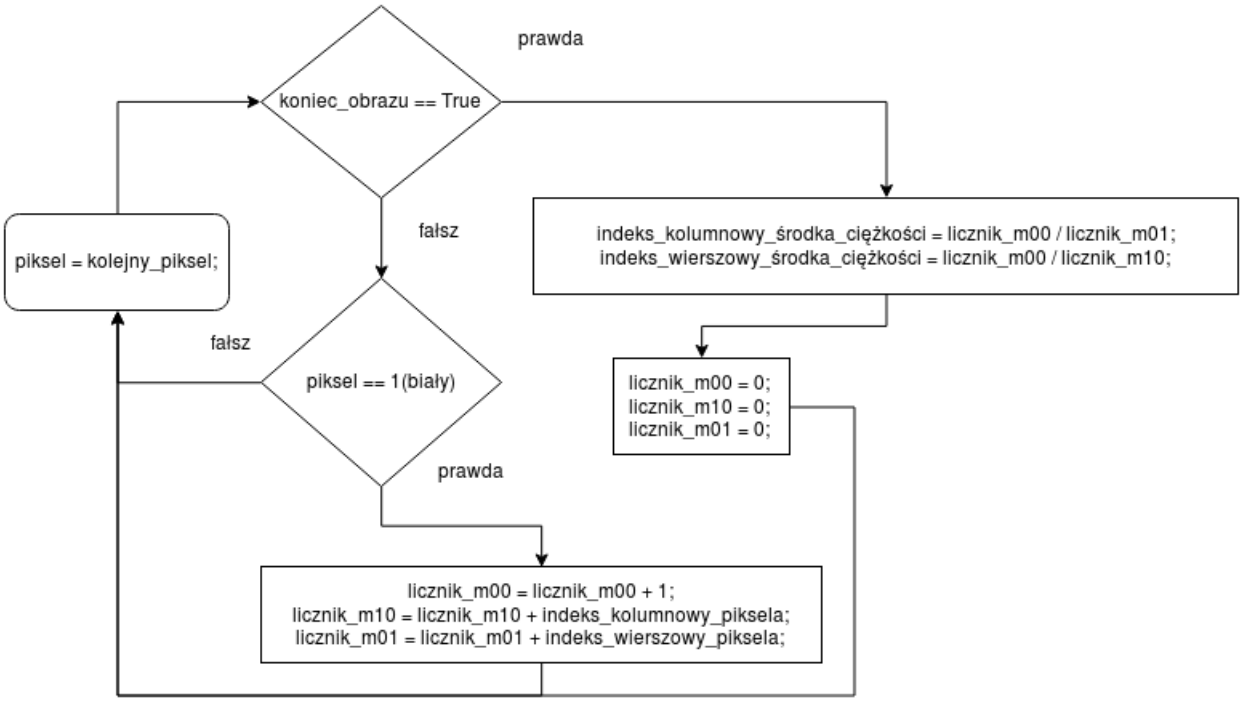
Progi ograniczające dolny i górny wykorzystywane są kolejno w logice asynchronicznej przedstawionej na schemacie 5.7. Nie zamieszczono rysunków z symulacji implementacji sprzętowej, ponieważ dla obrazów o rozmiarze tak małym jak 348x128, filtr nie daje rzetelnych rezultatów. W modelu programowym wykorzystano obraz o rozmiarze 1392x512. Tak znaczne zmniejszenie go oraz zniekształcenie powoduje zaburzenie działania filtru. Wynika to z tego, że działanie filtru jest zależne od wysokości i szerokości obrazu.

5.4. Podział obrazu

W celu wyznaczenia środków ciężkości w określonych strefach na obrazie, wykorzystuje się implementacje sprzętową(dla każdego obszaru oddzielnie) przedstawioną na schemacie 5.8. Jeśli, któryś z liczników po przeiterowaniu całego obrazu jest równy 0, środek nie jest wyznaczany.



Rys. 5.7. Schemat blokowy implementacji sprzętowej algorytmu do odfiltrowania elementów nie będących liniami ruchu drogowego.



Rys. 5.8. Schemat blokowy implementacji sprzętowej algorytmu do wyznaczenia środka ciężkości.

6. Testy

Zaimplementowane algorytmy przetestowano na materiałach pobranych z internetu [**Geiger2013IJRR**].

7. Podsumowanie

7.1. Wnioski

W ramach pracy inżynierksiej udało się zaimplementować model programowy w języku Matlab. Algorytm charakteryzował się jednak zauważalnym czasem obliczeń. Implementacje sprzętowe binaryzacji oraz wyznaczania obszaru zainteresowanie ROI udało się przetestować symulacyjnie. Porównanie wyników z implementacją programową świadczy o tym, że algorytmy zostały poprawnie zaimplementowane. Symulacja możliwa była do przeprowadzanie dla obrazów o wymiarach 348x128. Dlatego też nie udało się przeprowadzić symulacji filtru LMPS. Znaczne zmniejsze i zniekształcenie względem orginalnego obrazu powoduje, że filtr nie daje rzetelnych rezultatów. Zamieszczone testy przeprowadzono na modelu programowym. Przedstawionego podejscia nie udało się zaimplementować na platformie Zybo Zynq SoC ze względu na zdarzenia losowe.