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

Miejsce z podziękowaniami.

Spis treści

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 [1]. 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) [2]. 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) [3].

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

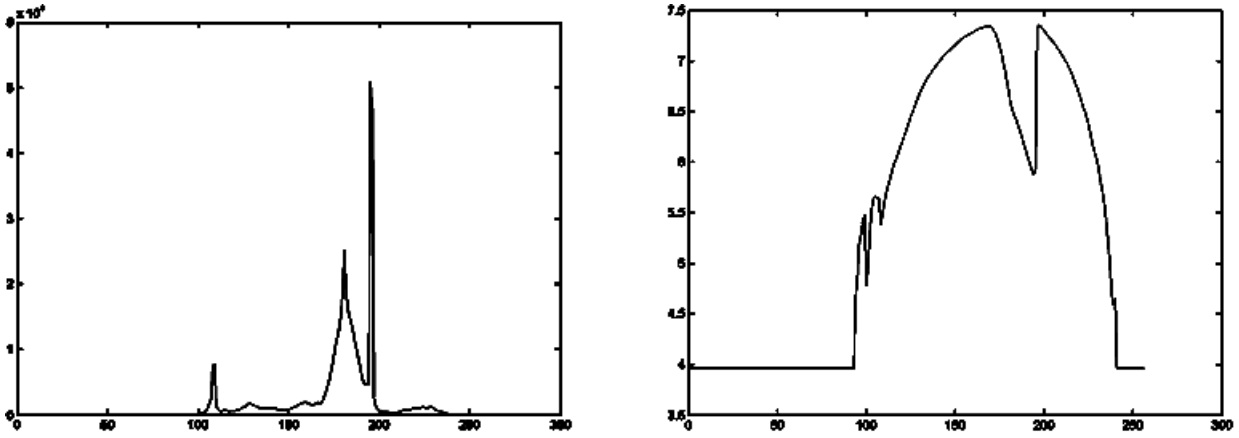
Pierwszym etapem detekcji pasów ruchu drogowego jest konwersja obrazu kolorowego w formacie RGB (ang. *Red, Green, Blue* – czerwony, zielony, niebieski) na obraz binarny. Proces ten ma na celu aby wstępnie otrzymać cechy określające dany obraz oraz ułatwić dalsze jego przetwarzanie. W celu ułatwienia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

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 [5], [6].

Innym podejściem [7] jest wykorzystanie algorytmu Canny Edge Detector w miejscu filtra Sobel'a. Canny Edge łączy w sobie filtr sobela i zdefiniowaną histerezę [8]. 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.

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



Rys. 2.1. Porównanie histogramu i funkcji entropii. Źródło: [10]

Próg binaryzacji można również dobrać samodzielnie, np. przyjmując, że jej wartość jest równa połowie wartości maksymalnej jaką może mieć piksel [9]. Innym podejściem, zaprezentowanym w artykule [10], jest wykorzystanie funkcji entropii lub histogramu. Histogram obrazu jest to sposób reprezentacji rozkładu wartości piksleli, z których składa się obraz. Może przyjmować formę interpolowanego wykresu, gdzie pozioma oś układu współrzędnych odpowiada za wartość piksela. Pionowa oś reprezentuje liczebność punktów o wartości równej wartości argumentu znajdującego się na poziomej osi [11]. 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.

2.2. Wyznaczanie linii drogowych

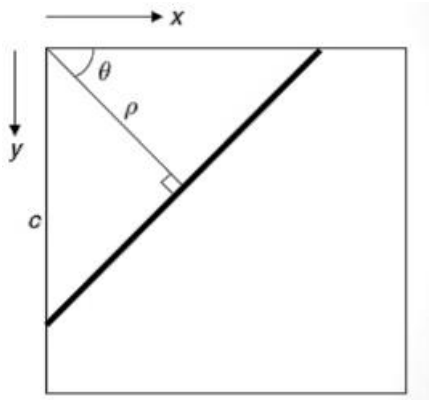
W tym etapie skupimy się na detekcji pasów ruchu poprzez wyznaczenie linii na podstawie obrazu binarnego. W artykule [7] wykorzystano w tym celu transformatę Hough'a. Jest to metoda wykrywania prostych w widzeniu komputerowym [12].

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.

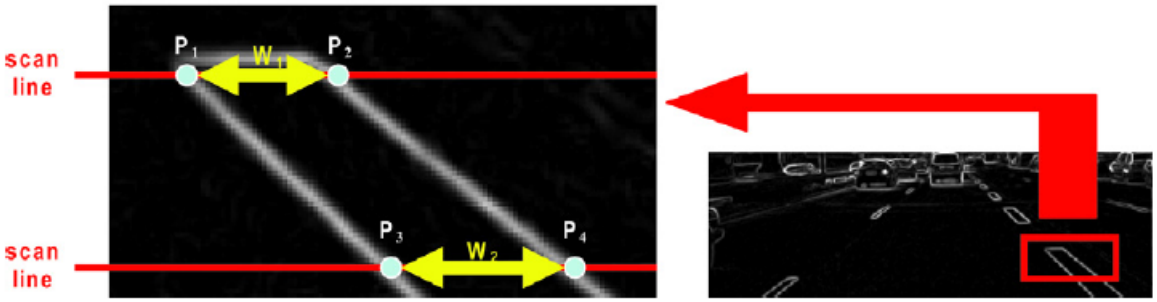
$$x\cos(\theta) + y\sin(\theta) = \rho \tag{2.2}$$

W celu wyznaczenia pełnego obrazu w przestrzeni Hough'a należy przeiterować po całym obrazie przetwarzanym 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.

Zakres grubości linii znajdujących się przykładowo na autostradzie jest ściśle określony. Ta zależność została wykorzystana w pracy [4], gdzie użyto filtru wykrywającego linie ruchu drogowego. Działa on



Rys. 2.2. Graficzne przedstawienie zależności współrzędnych x, y i θ, ρ. Źródło: [13]

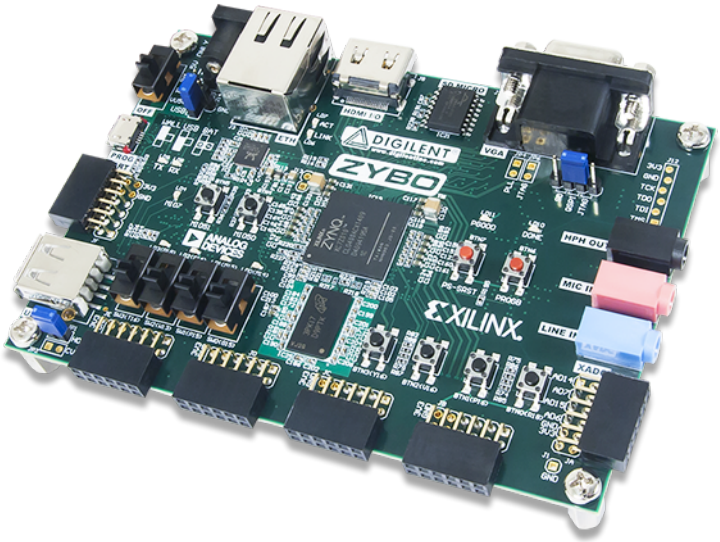


Rys. 2.3. Zdjęcie przedstawiająca sprawdzanie odległości w poziomie pomiędzy dwoma białymi pikselami. Źródło: [4]

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 2.3.

3. Opis platformy sprzętowej

3.1. Specyfikacja sprzętowa platformy Zybo



Rys. 3.1. Platforma Zybo z układem Zynq SoC. Źródło: [14]

Do implementacji sprzętowej została wykorzystana karta Zybo firmy Digilent 3.1 wyposażona w układ Zynq SoC (ang. *System on Chip*). W dokumencie [15] 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*),
- multiplekserów,
- szybkiej logiki przeniesienia.

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*),
- 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,
- GTP/GTX Transceivers – moduły umożliwiające transmisję szeregową z prędkością do 12,5 Gb/s (GTX) i 6,25 (GTP).

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 [16]:

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

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

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

5. Implementacja sprzętowa

6. Testy

6.1. Wnioski

7. Podsumowanie