Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Programowo-sprzętowa realizacja algorytmów

Implementacja asystenta pasa ruchu na Xilinx Zynq

Prowadzący:
dr inż. Tomasz Kryjak

Autorzy:
Konrad Adasiewicz
Anna Musiał
Filip Kubicz

1 Wstęp

Jazda samochodem jest niebezpieczna. W Polsce w 2015 roku miało miejsce 32 967 wypadków drogowych, w których zginęło 2 938 osób [1]. 82,8% wypadków było spowodowanych przez kierowców. Jednym z systemów, który może poprawić bezpieczeństwo na drogach jest ostrzeżenie o zmianie pasa ruchu. Ostrzegawczy sygnał dźwiękowy lub nawet aktywna zmiana toru jazdy samochodu może powstrzymać zasypiającego lub nieuważnego kierowcę przed zjechaniem do rowu, potrąceniem pieszego na poboczu czy spowodowaniem kolizji z innymi samochodami.

W celu rozpoznania pasów ruchu i położenia samochodu względem jezdni wykorzystaliśmy wykrywanie krawędzi metodą Canny, a następnie transformatę Hougha na otrzymanym obrazie binarnym w celu znalezienia linii malowań zbliżonych do prostych. W dalszej części opisano model programowy w OpenCV, model stałoprzecinkowy C++ oraz opis sprzętu na platformę Xilinx Zynq w języku Verilog.

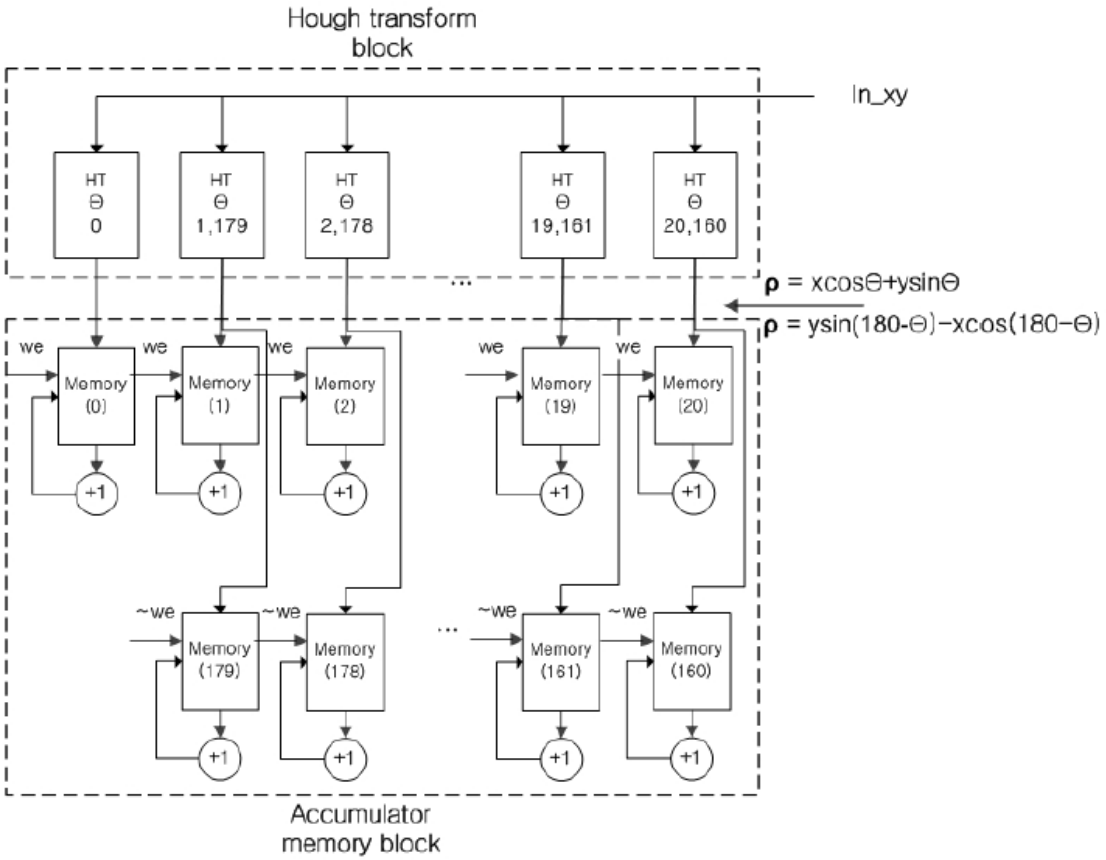
2 Wykrywanie pasów ruchu - przegląd artykułów naukowych

Projekt miał charakter badawczo-implementacyjny. Do uzyskania obrazu binarnego krawędzi wykorzystaliśmy algorytm Canny opracowany w pracy inżynierskiej [2]. Następnie należało zdecydować w jaki sposób rozpoznawać pasy ruchu i jak wybrany algorytm zrealizować w torze wizyjnym na platformie Xilinx Zynq-7000.

Do wykrywania pasów ruchu wybraliśmy transformatę Hougha, jako że pasy drogowe widziane z przedniej kamery w aucie najczęściej są zbliżone do linii prostych. Przykład analizowanego obrazu przedstawia rysunek 1.

Rysunek 1: Widok z przedniej kamery w samochodzie

Transformata Hougha jest złożoną operacją, którą trudno zrealizować w czasie rzeczywistym na systemie procesorowym. Wpływa na to konieczność wykonania obliczeń dla każdego kąta θ dla każdego piksela obrazu (lub przynajmniej pikseli należących do krawędzi). Przy obliczaniu transformaty Hougha dla kątów co 1°dla obrazu HD 1280x720 pikseli, trzeba wykonać 331 776 000 mnożeń dla każdej ramki. Praca [3] podaje sposób zrównoleglenia obliczeń arytmetycznych w transformacie Hougha na układzie FPGA.



Rysunek 2: Architektura obliczeniowa transformaty Hougha w pracy [3]

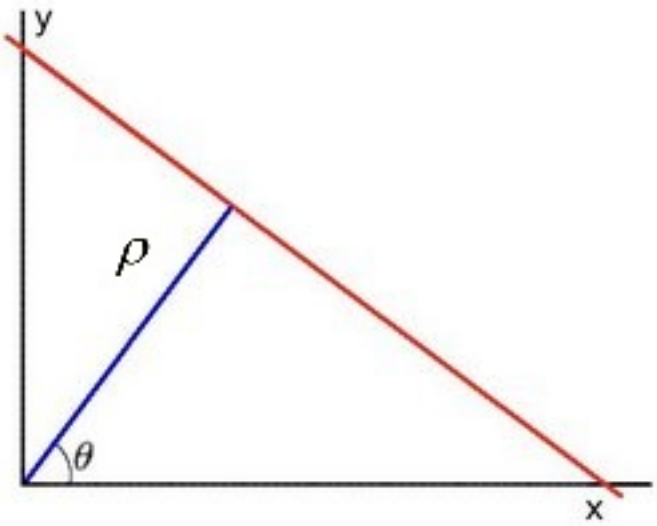
W pracy [4] opisano

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3 Rozpoznawanie linii - transformata Hougha

Transformata Hougha służy do znajdywania na obrazie linii prostych. Jest to przekształcenie z przestrzenii kartezjańskiej (x, y) w przestrzeń Hougha (ρ, θ) dane wzorem

$$\rho = x\,cos\,\theta + y\,sin\,\theta \tag{1}$$



Rysunek 3: Transformata Hougha w przestrzeni kartezjańskiej

Transformata pozwala wykryć linie proste korzystając z faktu, że punkt w przestrzeni (x, y) odpowiada sinusoidzie w przestrzeni Hougha. Stosując transformatę Hougha dla linii prostej na płaszczyźnie kartezjańskiej, w przestrzeni (ρ, θ) zaobserwujemy rodzinę sinusoid, które przecinają się we wspólnym punkcie. To przecięcie stanowi maksimum w obrazie wynikowym i reprezentuje wykrytą prostą.

4 Model programowy transformaty Hougha

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4.1 Model funkcjonalny z użyciem OpenCV

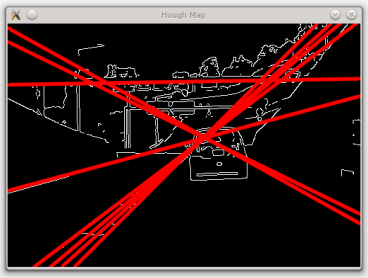
Do budowy wysokopoziomowego modelu funkcjonalnego użyte zostały funkcje biblioteki OpenCV:
`void Canny(InputArray image, OutputArray edges, double threshold1, double threshold2, int apertureSize=3)`

gdzie
`image` – obraz wejściowy
`edges` – wyjściowy obraz do detekcji krawędzi,
`threshold1` – pierwszy próg binaryzacji,
`threshold2` – drugi próg binaryzacji,
`apertureSize` – rozmiar maski użytej w funkcji `Sobel()`.

`void HoughLines(InputArray image, OutputArray lines, double rho, double theta, int threshold, double srn=0, double stn=0 )`
`image` – binarny obraz wejściowy,
`lines` – wyjściowy wektor linii; każda reprezentowana jest przez dwa elementy – ρ i θ
`rho` – krok dla odległości ρ w pikselach
`theta` – krok dla kąta θ w radianach
`threshold` – próg głosów, które musi osiągnąć linia aby została zwrócona.

Rysunek 4: Obraz z kamery

Rysunek 5: Krawędzie znalezione metodą Canny



Rysunek 6: Linie wybrane przez transformatę Hougha z głosowaniem

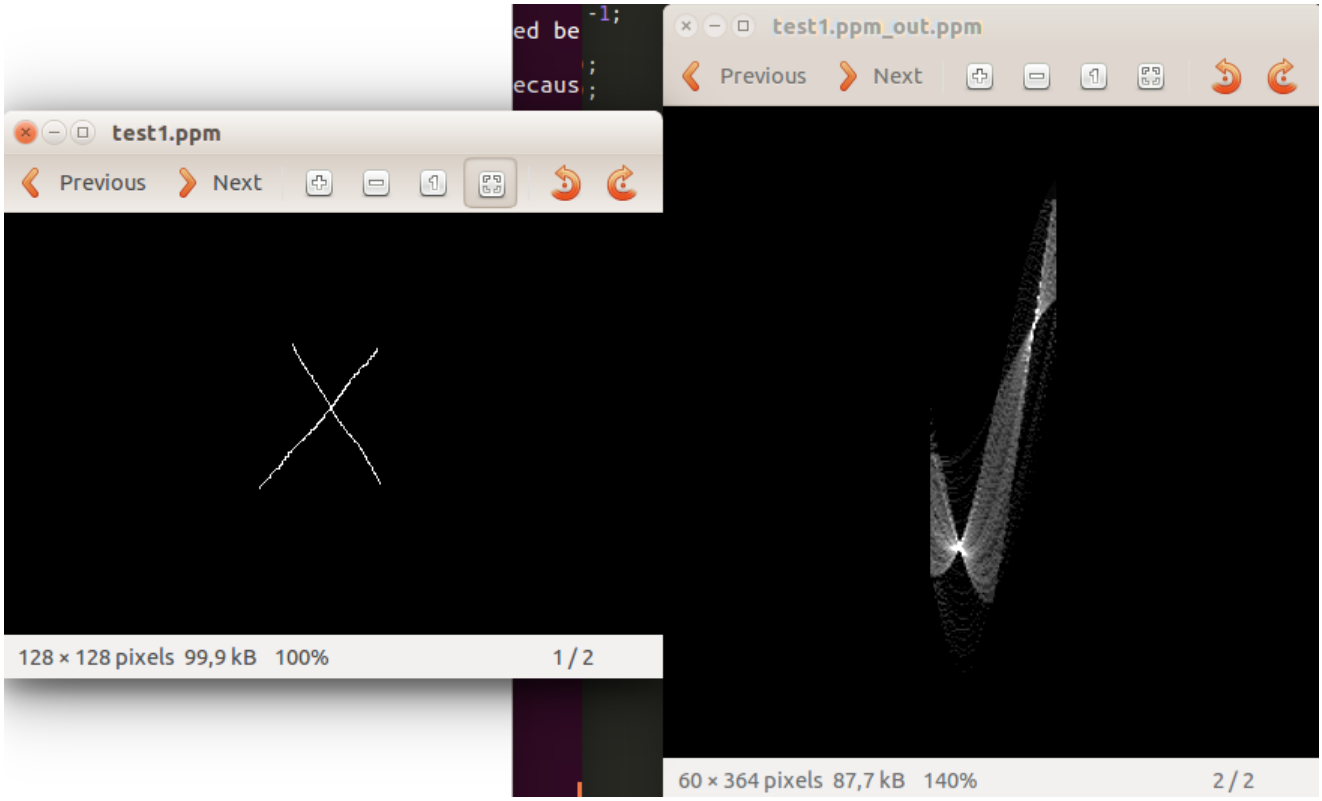
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis

non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4.2 Model stałoprzecinkowy w języku C++

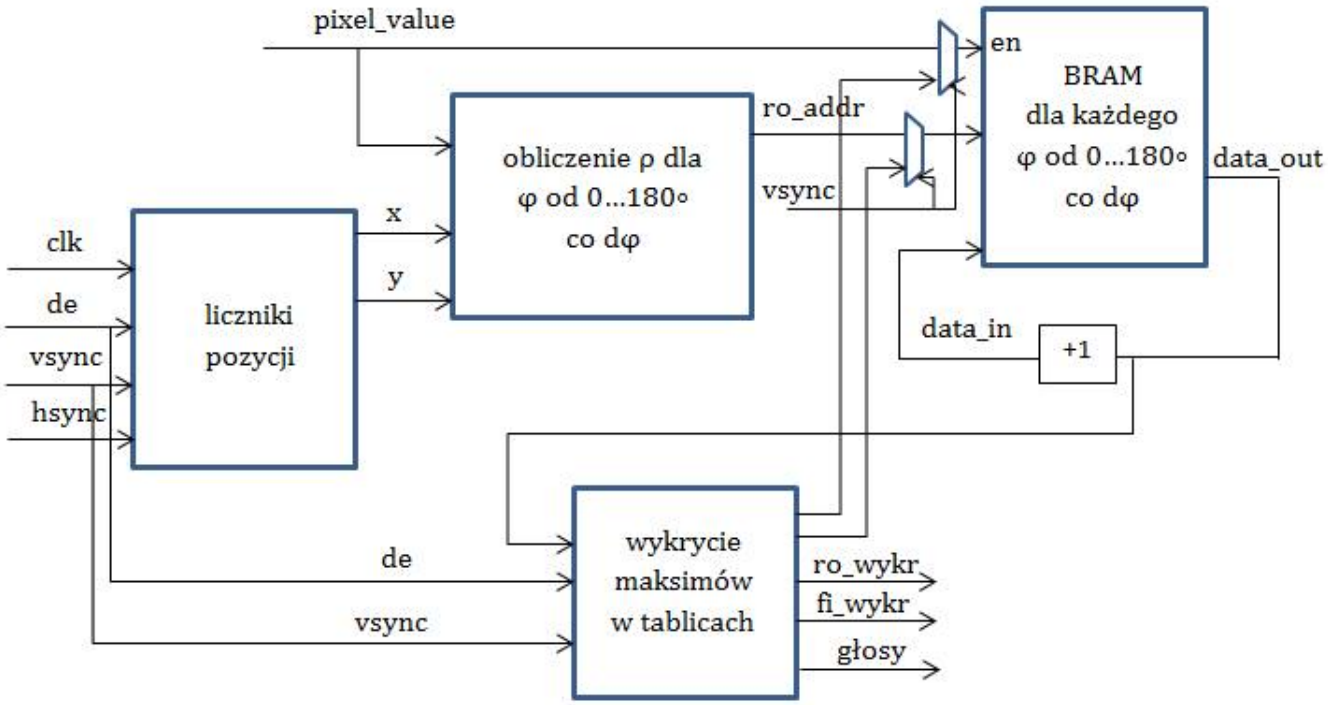
Opis programu

Klasa `fp.h`



Rysunek 7: Wynik działania funkcji `Hough()` dla liczb stałoprzecinkowych

5 Realizacja modułu Hough w sprzęcie



Rysunek 8: Struktura modułu realizującego transformatę Hougha z wygaszaniem minimów na FPGA

6 Wnioski

Zybo ...

Zabrakło czasu, aby poszczególne moduły implementacji sprzętowej połączyć ze sobą i wykorzystać do wykrywania linii.

6.1 Możliwe ulepszenia

Literatura

[1] Wypadki drogowe - raporty roczne, dostęp 01.06.2016, 2016. http://statystyka.policja.pl/st/ruch-drogowy/76562,Wypadki-drogowe-raporty-roczne.html.

[2] Adam Żelazowski. Implementacja sprzętowa wybranych operacji przetwarzania wstępnego obrazu: wyrównywania histogramu, filtracji medianowej oraz wykrywania krawędzi metodą Canny, 2015.

[3] Yong-Jin Jeong Hyo-Kyun Jeong. Design of Hough transform hardware accelerator for lane detection. 2013.

[4] Mariusz Kapruziak. Zmodyfikowana zrandomizowana transformata Hougha w strukturze FPGA. 2011.