Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

(позначення документа)

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДУ ДЕТЕКТУВАННЯ ДОРОЖНЬОЇ СМУГИ З ВИКОРИСТАННЯМ FPGA

(тема)

Виконав:
студент 2 курсу, групи ІНФм-19-2
Селевко О.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Селевку Олександру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та реалізація методу детектування дорожньої смуги з використанням FPGA*

затверджена наказом по університету від « 23 » жовтня 2020 року № 1428Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 20 листопада 2020 р.

3. Вихідні дані до роботи *Теоретичні відомості про методи розпізнавання відеозображень за допомогою програмованих логічних інтегральних схем (FPGA)*

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд методів розпізнавання відеозображень за допомогою програмованих логічних інтегральних схем (FPGA).
2. Математичні моделі детектування дорожньої смуги.
3. Вибір оптимального методу детектування дорожньої смуги.
4. Реалізація методу детектування дорожньої смуги з використанням FPGA.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Актуальність задачі детектування дорожньої смуги, постановка задачі детектування дорожньої смуги з використанням FPGA, плата FPGA DE10-Nano, графічні результати виділення границь дорожньої смуги

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	23.10.2020	**виконано**
2	Аналіз завдання, підбір літератури	24.10.20-25.10.20	**виконано**
3	Аналіз літератури з досліджуваної проблеми	26.10.20-01.11.20	**виконано**
4	Аналіз технічних засобів детектування відеозображень з використанням FPGA	02.11.20-05.11.20	**виконано**
5	Розробка методу детектування дорожньої смуги з використанням FPGA	06.11.20-12.11.20	**виконано**
6	Програмна реалізація	13.11.20-15.11.20	**виконано**
7	Оформлення пояснювальної записки	16.11.20-17.11.20	**виконано**
8	Перевірка на плагіат	20.11.2020	
9	Рецензування	23.11.2020	
10	Підготовка презентації та доповіді	23.11.2020	
11	Занесення роботи в електронний архів	24.11.2020	
12	Попередній захист атестаційної роботи	30.11.2020	

Дата видачі завдання _23_ __жовтня_2020 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Кобилін О.А.__
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка атестаційної роботи: 84 с., 7 табл., 30 рис., 80 джерел.

ПРОСУНУТА СИСТЕМА ДОПОМОГИ ВОДІЄВІ, ДЕТЕКТОР СОБЕЛЯ, ВИЯВЛЕННЯ СМУГИ, ВІДСТЕЖЕННЯ СМУГИ, FPGA.

Об’єктом дослідження є розробка програмно-апаратного засобу знаходження дорожньої смуги з відеоряду, захопленого камерою.

Метою є розгляд існуючих методів обробки зображення в просунутих системах допомоги водієві. Вирішується задача реалізації на FPGA знаходження границь дорожньої смуги.

Виявлення дорожньої смуги відбувається шляхом знаходження білих маркувальних смуг на темній дорозі. Відстеження дорожньої смуги використовує виявлені маркери дорожньої смуги і адаптується до зміни дорожніх умов. Функція відстеження є популярною задачею в обробці зображень і звичайно включає один або кілька методів попередньої обробки, які можуть бути реалізовані на FPGA для роботи в режимі реального часу.

Розглянуто ряд алгоритмів виявлення й відстеження дорожньої смуги, розроблених протягом останніх десятиліть. У рамках атестаційної роботи був розроблений і реалізований апаратно-програмний засіб реалізації методу детектування дорожньої смуги з використанням FPGA.

ADVANCED DRIVER ASSISTANCE SYSTEM, SOBEL DETECTOR, LANE DETECTION, LANE TRACKING, FPGA.

The object of the study is the development of software and hardware for finding a lane from the video captured by the camera.

The aim is to review existing image processing methods in advanced driver assistance systems. The realization problem of lane borders finding on FPGA is solved.

Lane detection is finding the white markings on a dark road. Lane tracking use the detected lane markers and adjusts itself according to changing road conditions. The tracking function is a popular task in image processing and usually includes one or more pre-processing methods that can be implemented on an FPGA for real-time operation.

A number of lane detection and tracking algorithms developed in recent decades are considered. As part of the certification work, a hardware-software tool for implementing the method of lane detection using FPGA was developed and implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACO – Ant Colony Optimization, поліноміальний алгоритм для знаходження наближених розв'язків

ADAS – Advanced Driver Assistance System, просунута система допомоги водієві

ALiS – Alternate Lighting of Surfaces, альтернативне висвітлення поверхонь

ALM – Adaptive Logic Module, адаптивний логічний модуль

ALU – Arithmetic Logic Unit, арифметико-логічне обладнання

ALUT – Adaptive Look-Up Table, адаптивна таблиця відповідності

ANN – Artificial Neural Networks, штучні нейронні мережі

ASIC – Application Specific Integrated Circuit, спеціалізована інтегральна мікросхема

CLB – Configurable Logic Block, логічний блок, що настроюється

CPU – Central Processing Unit, центральний процесор

CRT – The Cathode Ray Tube, електронно-променева трубка

DROI – Dynamic Region of Interest, динамічна видима область

DSC – Dice similarity coefficient, коефіцієнт подоби гри в кості

DSP – Digital Signal Processing, блок обробки цифрових сигналів

EDS – Embedded Development Suite, вбудований комплект розробки

FPC – Flexible PIC Concentrators, гнучкі концентратори PIC

FPGA – Field-Programmable Gate Array, програмована користувачем вентильна матриця або програмована логічна інтегральна схема

FPGASOC – Field-Programmable Gate Array System on a Chip, система програмованої користувачем вентильної матриці на мікросхемі

GPU – Graphic Processing Unit, графічний процесор

HDL – Hardware Description Language, мова опису апаратних засобів

HDMI – High-Definition Multimedia Interface, мультимедійний інтерфейс високої чіткості

HLS – High Level Synthesis, синтез високого рівня

HPS – Hardware Processing System, апаратна система обробки

IP – Intellectual Property, інтелектуальна власність

IPM – Inverse Perspective Mapping, зворотнє перспективне відображення

LDW – Lane Departure Warning, попередження про схід зі смуги

LUT – Look-Up Table, таблиця відповідності

NCC – Normalized Cross Correlation, нормалізована взаємна кореляція

NMS – Non-maximum Suppression, подавлення не максимумів, пошук локальних максимумів

OpenCL – Open Computing Language, відкрита компьютерна мова програмування

OpenCV – Open Source Computer Vision Library, бібліотека машинного зору з відкритим вихідним кодом

PAL – Phase Alternation Line, лінія зміни фаз

PIC – Physical Interface Cards, плата фізичних інтерфейсів

RANSAC – RANdom SAmple Consensus, метод оцінки параметрів моделі на основі випадкових вибірок

RHT – Polar randomized Hough Transform, полярне рандомізоване перетворення Хафа

ROI – Region of Interest, видима область

RTL – Register Transfer Logic, логіка межрегістрових пересилань

SHT – Statistical Hough Transform, статистичне перетворення Хафа

SLT – Symmetrical Local Threshold, симетричний локальний поріг

SOPC – System On a Programmable-Chip, система на програмувальній мікросхемі

UVC – USB Video device Class, відеоприлад USB

VIP – Video Image Processing, обробка відеозображення

ВСТУП

Просунуті системи допомоги водієві (ADAS) забезпечують безпечне й краще керування транспортним засобом. Це допомагає автоматизувати, адаптувати й поліпшити досвід водія. Більшість дорожніх подій відбувається через недбалість водія. Просунута система допомоги водієві забезпечує безпеку й зменшує робоче навантаження водія. Щораз, коли зустрічаються небезпечна ситуація, система або попереджає водія, або бере активну роль шляхом виконання необхідного коригувального заходу для запобігання нещасного випадку.

Попередження про схід зі смуги (LDW) є важливим модулем у просунутій системі допомоги водієві. У заснованій на відео системі попередження про схід зі смуги камера для одержання зображення дороги міститься за лобовим склом транспортного засобу. Смуги на дорозі інтерпретуються, і маршрути визначаються. Щораз, коли переміщення транспортного засобу з дорожньої смуги є некерованим, водієві дається попередження. Модуль відключається сигналом повороту. Виявлення маркера дорожньої смуги є початковим кроком системи попередження про схід зі смуги.

FPGA (Field Programmable Gate Array), програмована користувачем вентильна матриця або програмована логічна інтегральна схема є цифрова інтегральна мікросхема, яка складається з програмуємих логічних блоків та програмуємих з’єднань між цими блоками.

Можливість програмно конфігурувати ці пристрої, їх мала потужність споживання і висока швидкодія роблять актуальною темою використання FPGA для вирішення задач, які ставляться перед системами машинного зору, таких як детектування дорожньої смуги в системах допомоги водієві.

1 ОГЛЯД ОСНОВНИХ АЛГОРИТМІВ ВИЯВЛЕННЯ Й ВІДСТЕЖЕННЯ ДОРОЖНЬОЇ СМУГИ В СИСТЕМАХ ДОПОМОГИ ВОДІЄВІ

1.1 Машинний зір і обробка зображень

У наш час машинний зір використовується далеко поза просто обробкою зображень. Він інтегрує комунікацію й графічні методи, процесор і автоматизоване проектування, а також обробку інформації й керування.

Основні стадії обробки зображень [1] показані на рисунку 1.1.

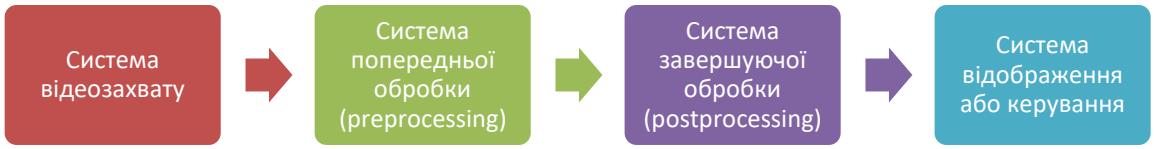


Рисунок 1.1 – Оперативна система обробки зображень

У першому кроці дані зображення отримані від цифрової/аналогової камери або інших обладнань уведення відеосигналу. Тоді система обробки може підрозділяти на систему попередньої обробки й виконуючу наступну обробку систему.

Система попередньої обробки включає перетворення формату відео, попередньо обробляючи алгоритми, такі як граничне виявлення або сегментація зображення [1 – 4]. Потім проводиться завершуюча [5 – 7] обробка, і результат виводиться на пристрій візуалізації, або він може безпосередньо генерувати сигнал керування до інших систем [3].

Одним з важливих застосувань обробки зображень є використання обробки відеопотоку в системах допомоги водієві, де вона здійснює завдання виявлення й відстеження дорожньої смуги [7, 8].

1.2 Обробка відеозображень для виявлення й відстеження дорожньої смуги

Існують два методи для виявлення дорожньої смуги: заснований на функції і заснований на моделі. Заснований на функції підхід використовує низькорівневі функції, такі як крайові функції [9 – 11], тоді як підхід, заснований на моделі, використовує для виявлення дорожніх смуг геометричні параметри [12, 13].

Рисунок 1.2 представляє загальну блок-схему системи попередження про схід зі смуги.

Рисунок 1.2 – Блок-схема простої системи попередження про схід зі смуги

Система попередження про схід зі смуги повинна вирішувати багато проблем, які включають різноманітність появи дорожньої смуги, зміни в ясності зображення, зміни в умовах видимості [8]. У різних країнах буде відмінність у використовуваних маркерах дорожньої смуги. Необхідно досліджувати тип поведінки дорожньої смуги й супутні проблеми у відстеженні.

Звичайно для дорожньої розмітки використовуються жовті й білі кольори, відбивачі смуги з особливими кольорами. Число й ширина дорожніх

смуг міняються залежно від країни. Можуть бути зв’язані проблеми ясності в зображеннях через присутність тіней. Маркери дорожньої смуги можуть бути закриті сусідніми транспортного засобами. Коли транспортний засіб виходить із тунелю, у освітленні відбувається різка зміна. Таким чином, ясність зображення залежить від освітлення. Видимість маркерів дорожньої смуги зменшується через різні погодні умови, таких як дощ, туман, сніг. Видимість може бути менше в нічних умовах.

Рисунок 1.3 показує проблеми у виявленні дорожньої смуги і її відстеженні [8]:

- транспортний засіб, що закриває сусідній маршрут;
- присутність тіні;
- дощова дорога;
- екстремальне освітлення на лівій стороні зображення.

Рисунок 1.3 – Різні проблеми у виявленні дорожньої смуги

Виявлення дорожньої смуги є непростою задачею, для рішення якої розроблено кілька десятків алгоритмів.

1.3 Методи обробки зображення в алгоритмах виявлення й відстеження дорожньої смуги

Як правило, додатки для обробки зображень застосовують декілька методів обробки зображень низького рівня. Наприклад, зображення може бути Гаусово фільтрованим для скорочення шуму в зображенні й потім оброблено граничним алгоритмом виявлення, застосованим для виділення країв функцій у зображенні. Тоді високорівневі алгоритми використовують інформацію від інформації низького рівня для прийняття розв’язків щодо вмісту об’єкта.

Наприклад, в аналізі дорожнього знаку, краї будуть проаналізовані для визначення форм знаків, тобто чи є вони круговими, квадратними, прямокутними тощо для допомоги в їхній класифікації.

Ці ж методи обробки зображень використовуються при виявленні і відстеженні дорожньої смуги.

1.3.1 Граничний метод виявлення Кенні

Граничний метод виявлення є популярним алгоритмом, який використовується на етапах попередньої обробки додатків машинного зору [14], зокрема для реалізації на платформі FPGA. Цей алгоритм використовує методи згортки, які є математичним способом об'єднати дві функції для формування третьої функції.

Існує кілька видів граничних методів виявлення, в тому числі з використанням детектору Собеля, детектору Лапласа з Гаусовим фільтром, детектору Кірша, детектору Роберта та детектора Прюітта [14]. Вони представляють різні дії на основі складності граничного обчислення й здатності алгоритму до виявлення границь, коли зображення страждає від шумового забруднення.

Граничний алгоритм виявлення розглядається як один із самих надійних алгоритмів, тому що показує гарну продуктивність, може досягати низького коефіцієнта помилок і поліпшує локалізацію певних країв [15].

Можливі етапи повного процесу граничного методу виявлення показані на рисунку 1.4.

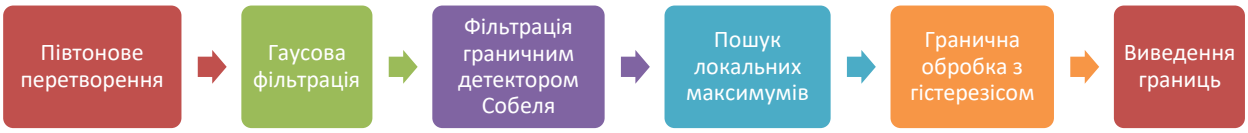


Рисунок 1.4 – Етапи граничного процесу виявлення

Бажаний результат повного процесу граничного методу виявлення показаний на рисунку 1.5.

Півтонове зображення Повна гранична обробка

Рисунок 1.5 – Приклад результату повної граничної обробки методом Кенні

1.3.2 Півтонове перетворення

Для реалізації граничного методу в першу чергу необхідно перетворити вихідне зображення RGB в півтонове зображення. Півтонове зображення (greyscale) є зображенням, де кожний піксель зображення містить тільки інформацію про інтенсивність Y'. Для пікселя RGB півтонове значення пікселя:

$$Y' = 0{,}299R + 0{,}587G + 0{,}114B. \tag{1.1}$$

Для зменшення затримки обчислень в режимі реального часу замість арифметики із плаваючою крапкою використовується арифметика обчислень за допомогою операцій зсуву. Зсув біта ліворуч дорівнює множенню на 2, вправо дорівнює діленню на 2. Можливо спростити вихідне рівняння до приблизного результату:

$$\begin{gathered} Y' = 0{,}299R + 0{,}587G + 0{,}114B \approx \\ \approx 0{,}297R + 0{,}586G + 0{,}113B \approx \frac{76 \cdot R + 150 \cdot G + 29 \cdot B}{256} = \\ = \frac{(2^6R + 2^3R + 2R) + (2^7G + 2^4G + 2G) + (2^5B - 2^3B + B)}{2^8}. \end{gathered} \tag{1.2}$$

При використанні для обчислень операцій зсуву приблизне півтонове перетворення може бути реалізоване без використання операцій множення або ділення.

Крім того, якби був потрібний більш точний результат, то він міг би бути досягнутий шляхом збільшення величини дільника. Наприклад, як варіант:

$$Y' = 0{,}299R + 0{,}587G + 0{,}114B \approx \frac{306 \cdot R + 601 \cdot G + 117 \cdot B}{1024}. \quad (1.3)$$

Однак, оскільки обчислення стає більш точним, система стає більш складною, що приводить до більшої кількості необхідних ресурсів і більш довгої затримки.

1.3.3 Гаусова фільтрація

Гаусова фільтрація використовується для відфільтрації шуму у вихідному зображенні. Зручним є те, що Гаусова фільтрація може бути реалізована методом згортки. 2-мірна Гаусова функція G (Gaussian) описується рівнянням:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}. \quad (1.4)$$

У рівнянні (1.4) x є відстань від джерела по горизонтальній осі, y є відстань від джерела по вертикальній осі й σ є стандартне відхилення.

Для реалізації Гаусової функції за допомогою методу згортки необхідно обчислити маску згортки для Гаусової функції. Це називається фільтрацією вікна. Згортка (ядро) маски звичайно набагато менше, ніж всього зображення.

Маску рухають по зображенню й застосовують згортку у кожнім місці розташування в зображенні. Чим більше розмір ядра, тем нижче чутливість детектора до шуму. Чим більше значення σ, результат стає більш згладженим з меншою кількістю шуму. Однак з більшим згладжуванням зображення, менше країв буде виявлено детектором.

Рисунок 1.6 показує результат застосування Гаусова фільтра з різними значеннями σ.

Півтонове зображення $\sigma = 1{,}4$ $\sigma = 2{,}0$

Рисунок 1.6 – Півтонове зображення після Гаусової фільтрації при різних значеннях σ

Наприклад, для ядра розміром 3 × 3 фільтр Гауса:

$$G = \begin{bmatrix} G_1 & G_2 & G_3 \\ G_4 & G_5 & G_6 \\ G_7 & G_8 & G_9 \end{bmatrix}. \quad (1.5)$$

Для реалізації його на кожному пікселі відеопотоку потрібні буфери на 3 рядка, щоб буферизувати 3 рядка за один раз і оброблювати 3 рядки з трьома пікселями P_i одночасно, як показано на рисунку 1.7.

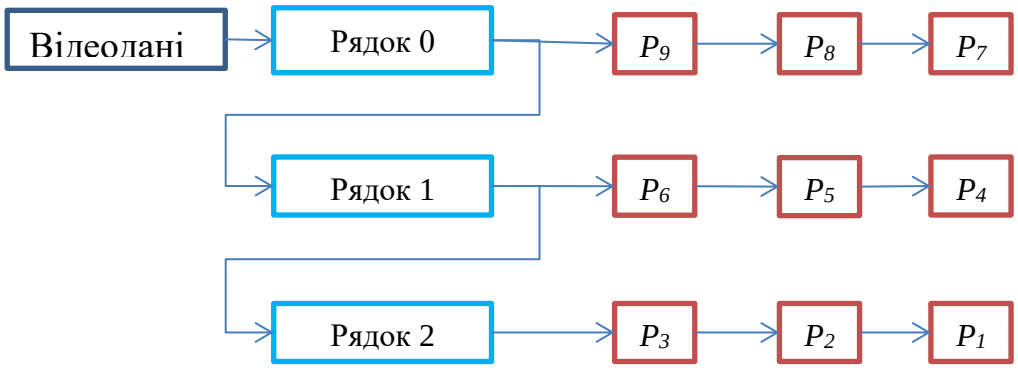


Рисунок 1.7 – Блок-схема обробки даних з буфером на 3 рядка

Тоді кожний піксель із його навколишніми 8 пікселями може сформувати ядро вихідного зображення 3 × 3, як показано на рисунку 1.8.

P_1	P_2	P_3
P_4	P_5	P_6
P_7	P_8	P_9

Рисунок 1.8 – Ядро вихідного зображення

Використовуючи це ядро, результат фільтрації поточного пікселя становить:

$$P' = G \cdot P = G_1 \cdot P_1 + G_2 \cdot P_2 + G_3 \cdot P_3 + G_4 \cdot P_4 + G_5 \cdot P_5 + \\ + G_6 \cdot P_6 + G_7 \cdot P_7 + G_8 \cdot P_8 + G_9 \cdot P_9. \quad (1.6)$$

Алгоритм послідовно реалізується для кожного пікселя.

Звичайно використовується ядро розміром 5 × 5 з фільтром Гауса з стандартним відхиленням $\sigma = 1{,}4$, який обчислюється як показано на рисунку 1.9 [14]. Це потрібно, щоб буферизувати 5 рядків відеопотоку.

$$\frac{1}{159}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Рисунок 1.9 – Значення фільтра Гауса для ядра 5 × 5 з $\sigma = 1{,}4$

Графічна інтерпретація коефіцієнтів фільтра Гауса для ядра 5 × 5 зображена на рисунку 1.10:

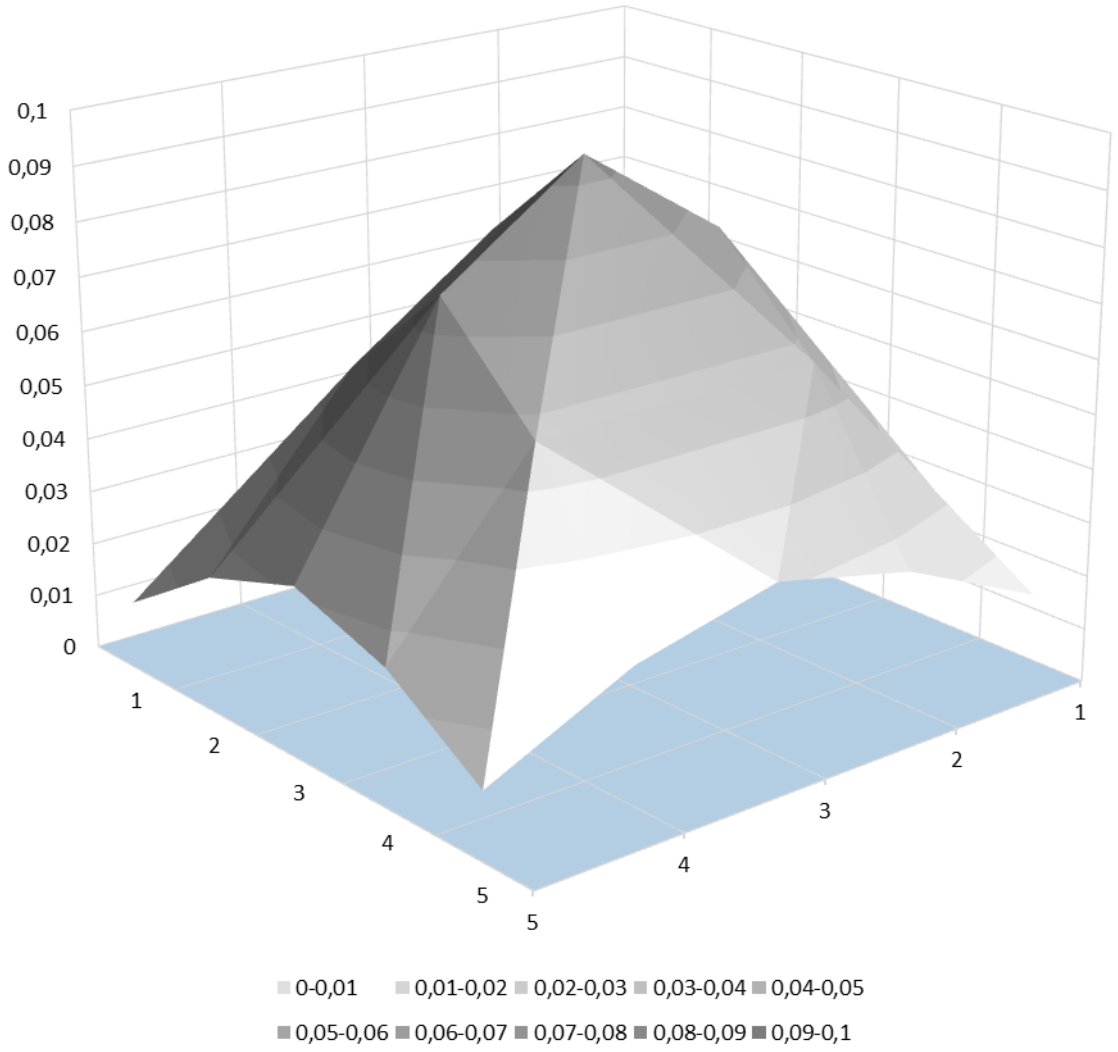


Рисунок 1.10 – Фільтр Гауса для ядра 5 × 5

Інакше можна записати Гаусів фільтр для ядра 5 × 5 з $\sigma = 1{,}4$

$$G = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}. \quad (1.7)$$

Тоді кожний піксель після фільтрації обчислюється:

$$P' = G \cdot P = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \cdot P. \quad (1.8)$$

P в даному випадку складає вікно 5 × 5 з поточного пікселя і 24 оточуючих пікселів.

Для скорочення споживання системних ресурсів і зменшення затримки системи з використанням фіксованої крапки може бути реалізований приблизний алгоритм

$$P' \approx \frac{1}{128} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \cdot P. \quad (1.9)$$

1.3.4 Граничний детектор Собеля з пошуком локальних максимумів

Після відфільтрування шуму й згладжування зображення наступним кроком є знаходження градієнту інтенсивності зображення шляхом виконання стандартного граничного виявлення Собеля. Оператор Собеля (Sobel) використовує маску згортки 3 × 3 для оцінки градієнта у вертикальному напрямку G'_x і оцінки градієнта в горизонтальному напрямку G'_y:

$$G'_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad G'_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \quad (1.10)$$

Після цього можна обчислити приблизну абсолютну величину градієнта (контрастність краю) $|G'|$ на рівні кожного пікселя:

$$|G'| = \sqrt{{G'_x}^2 + {G'_y}^2}. \quad (1.11)$$

Для спрощення обчислення абсолютного градієнта кожного пікселя часто використовується приблизне рівняння:

$$|G'| = \sqrt{{G'_x}^2 + {G'_y}^2} \approx |G'_x| + |G'_y|. \quad (1.12)$$

Кут градієнтної заливки θ кожного пікселя може бути обчислений як показано в рівнянні:

$$\theta = \text{arctg}\left(\frac{G'_y}{G'_x}\right). \quad (1.13)$$

Після обчислення величини й кутів градієнтної заливки кожного пікселя результат внаслідок побічних впливів може містити надто товсті краї. Для збільшення різкості країв для кожного пікселя використовується пошук локальних максимумів або подавлення не максимумів (NMS).

Для області зображення 3 × 3 навколо кожного пікселя може бути 4 напрямки градієнту крайових пікселів, які визначаються кутом градієнтної заливки цього пікселя:

– якщо кут градієнтної заливки становить $0° - 22{,}5°$ або $157{,}5° - 180°$, напрямок пікселя горизонталь;

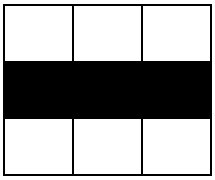
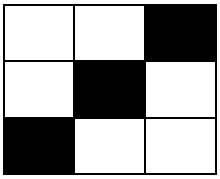
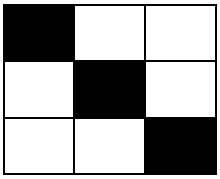
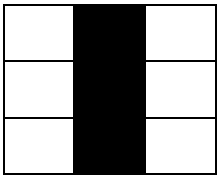
– якщо кут градієнтної заливки від $22{,}5°$ до $67{,}5°$, напрямок пікселя є позитивною діагоналлю;

– якщо кут градієнтної заливки від 112,5° до 157,5°, напрямок пікселя є негативною діагоналлю;

– якщо кут градієнтної заливки від 67,5° до 112,5°, напрямок пікселя є вертикаль.

Це показано в таблиці 1.1, де на зображенні кожна темна область представляє крайовий піксель:

Таблиця 1.1 – Чотири можливі напрямки градієнтів країв

Кут градієнтної заливки	**Назва напрямку градієнту**	**Зображення напрямку градієнту**
$\theta \in (0°; 22{,}5°] \cup (157{,}5°; 180°]$	горизонталь	
$\theta \in (22{,}5°; 67{,}5°]$	позитивна діагональ	
$\theta \in (112{,}5°; 157{,}5°]$	негативна діагональ	
$\theta \in (67{,}5°; 112{,}5°]$	вертикаль	

Як тільки орієнтація кожного пікселя обчислена, необхідно зрівняти значення величини кожного пікселя із двома пікселями поруч із ним, але в

різних напрямках. Якщо величина поточного пікселя буде найбільшою у порівнянні з іншими 2 пікселями, то цей піксель буде збережений як тонкий край. Інакше значення буде подавлене.

Рисунок 1.11 показує пікселі (позначені помаранчевим кольором), які потрібно буде використати для порівняння для кожного напрямку.

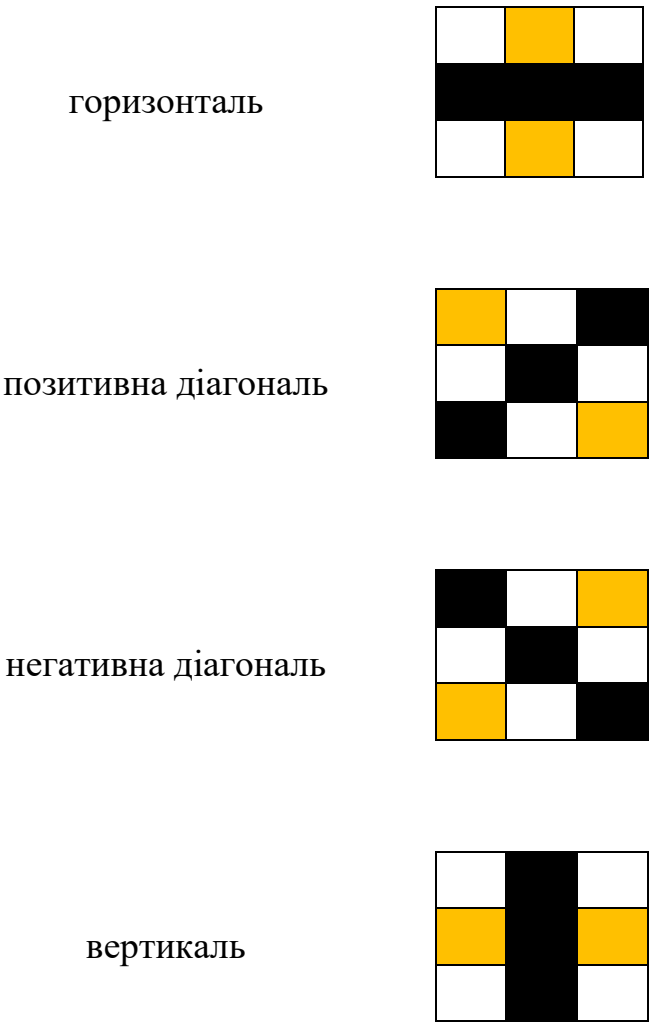


Рисунок 1.11 – Пікселі для порівняння при детектуванні Собеля

Для зменшення використання ресурсів і скорочення затримки використовується швидкий метод обчислення кутів градієнтної заливки θ без використання знаходження тригонометричних функцій (arctg) і ділення.

При цьому замість кутів θ використовується $\operatorname{tg}\theta$:

$$\operatorname{tg}\theta = \frac{G'_y}{G'_x}. \tag{1.14}$$

Отриманий тангенс кута можна округлити для запобігання використання арифметики з плаваючою комою. Тоді, наприклад, для умови $\theta > 22{,}5°$ можна використати

$$\theta > 22{,}5° \Rightarrow \operatorname{tg}\theta = \frac{G'_y}{G'_x} > \operatorname{tg}22{,}5° = 0{,}414. \tag{1.15}$$

$$\frac{G'_y}{G'_x} > 0{,}414 \approx 0{,}5. \tag{1.16}$$

З цього випливає умова для приблизного швидкого методу, який не буде потребувати порівняння десяткових чисел:

$$2G'_y > G'_x. \tag{1.17}$$

Можна використати приближення для тангенсів критичних кутів градієнтної заливки:

$$\operatorname{tg}22{,}5° = 0{,}414 \approx 0{,}5. \tag{1.18}$$

$$\operatorname{tg}157{,}5° = -0{,}414 \approx -0{,}5. \tag{1.19}$$

$$\operatorname{tg}67{,}5° = 2{,}414 \approx 2{,}5. \tag{1.20}$$

$$\operatorname{tg}112{,}5° = -2{,}414 \approx -2{,}5. \tag{1.21}$$

Таким чином, можна апроксимувати всі чотири умови визначення напрямку градієнтів країв, як вказано в таблиці 1.2.

Таблиця 1.2 – Чотири умови визначення напрямку градієнтів країв

Кут градієнтної заливки θ	**Тангенс кута градієнтної заливки** $\mathrm{tg}\,\theta$	**Умови приблизного швидкого методу**
$(0°; 22{,}5°]$ $\cup\,(157{,}5°; 180°]$	$(-0{,}414; 0{,}414]$	$0 < \lvert 2G'_y \rvert \le \lvert G'_x \rvert$
$(22{,}5°; 67{,}5°]$	$(-0{,}414; 2{,}414]$	$\begin{cases} \lvert G'_x \rvert < \lvert 2G'_y \rvert \le \lvert 5G'_x \rvert \\ \dfrac{G'_y}{G'_x} > 0 \end{cases}$
$(112{,}5°; 157{,}5°]$	$(-2{,}414; 0{,}414]$	$\begin{cases} \lvert G'_x \rvert < \lvert 2G'_y \rvert \le \lvert 5G'_x \rvert \\ \dfrac{G'_y}{G'_x} < 0 \end{cases}$
$(67{,}5°; 112{,}5°]$	$(-\infty; -2{,}414]$ $\cup\,(2{,}414; +\infty]$	$\lvert 2G'_y \rvert > \lvert 5G'_x \rvert$

Таблиця 1.3 показує кутові погрішності в обчисленні при приблизному швидкому методі визначення напрямку градієнтів.

Таблиця 1.3 – Кутова помилка приблизного швидкого методу

Кут градієнтної заливки θ	**Наближений тангенс кута градієнтної заливки** $\approx \mathrm{tg}\,\theta$	**Наближений кут градієнтної заливки** $\approx \theta$	**Кутова помилка** $\Delta\theta$
22,5°	0,5	26,5	4°
67,5°	2,5	68,2°	0,7°
112,5°	-2,5	111,8°	-0,7°
157,5°	-0,5	153,5°	-4°

Використання приблизного швидкого методу визначення напрямку градієнтів країв при детектуванні Собеля дозволяє ефективно використовувати можливості FPGA. Однак, оскільки цей метод використовує приблизний результат, деяка інформація під час обробки губиться, зокрема точне значення кута.

Цей метод може бути поліпшений шляхом збільшення точності умов швидкого методу, як показано в таблиці 1.4.

Таблиця 1.4 – Поліпшені умови визначення напрямку градієнтів країв

Кут градієнтної заливки θ	**Тангенс кута градієнтної заливки** $\mathrm{tg}\,\theta$	**Поліпшені умови приблизного швидкого методу**
$(0°; 22{,}5°]$ $\cup\ (157{,}5°; 180°]$	$(-0{,}414; 0{,}414]$	$0 < \lvert 100G'_y \rvert \leq \lvert 41G'_x \rvert$
$(22{,}5°; 67{,}5°]$	$(-0{,}414; 2{,}414]$	$\begin{cases} \lvert 41G'_x \rvert < \lvert 100G'_y \rvert \leq \lvert 241G'_x \rvert \\ \dfrac{G'_y}{G'_x} > 0 \end{cases}$
$(112{,}5°; 157{,}5°]$	$(-2{,}414; 0{,}414]$	$\begin{cases} \lvert 41G'_x \rvert < \lvert 100G'_y \rvert \leq \lvert 241G'_x \rvert \\ \dfrac{G'_y}{G'_x} < 0 \end{cases}$
$(67{,}5°; 112{,}5°]$	$(-\infty; -2{,}414]$ $\cup\ (2{,}414; +\infty]$	$\lvert 100G'_y \rvert > \lvert 241G'_x \rvert$

Поліпшення умов приблизного швидкого методу визначення напрямку градієнтів країв до трьох знаків дає можливість визначати кути градієнтної заливки з точністю до 0,01° [14].

Граничний детектор Собеля є ефективним методом при виявленні границь на зображенні, що може бути продуктивно використане при виявленні границь дорожньої смуги.

Рисунок 1.12 показує результат застосування детектора Собеля до вхідного півтонового зображення.

Півтонове зображення Застосування детектора Собеля

Рисунок 1.12 – Приклад результату застосування детектора Собеля

1.3.5 Гранична обробка з гістерезісом

Виявлені після пошуку локальних максимумів крайові пікселі забезпечують достатньо точне визначення реальних країв у зображенні. Однак залишаються деякі крайові пікселі, які були результатом впливу шуму або змін кольору.

Для усунення цих побічних впливів важливо відфільтрувати крайові пікселі зі значенням малого градієнта й зберегти крайові пікселі з високим значенням градієнта. Це виконується шляхом вибору високих і низьких граничних значень – порогів.

Якщо значення градієнта крайового пікселя вище, ніж високе граничне значення, воно позначається як сильний крайовий піксель. Якщо значення крайового пікселя буде меншим, ніж низьке граничне значення, воно буде подавлене.

Рекомендується [16], на основі аналізу відношення сигнал/шум, щоб відношення верхнього порога до нижнього було в діапазоні два або три до

одного. Пікселі, які є між цими двома порогами, прийняті, якщо вони межують з сильним крайовим пікселем.

При швидкому методі, для скорочення використовуваних ресурсів і зменшення затримки, для фільтрації результату використовуються просто два фіксовані пороги. Тоді кожний тонкий крайовий піксель безпосередньо розглядається як крайовий піксель.

При точному методі поріг визначається найбільшим значенням величини від попереднього результату. Верхній поріг установлюється як 1/6 найбільшої величини попереднього результату. Нижній поріг установлюється як 1/3 або 1/2 від верхнього порогу [17].

Якщо який-небудь піксель буде між цими двома порогами, він буде перевірений, чи оточується він якими-небудь тонкими або товстими крайовими пікселями. Якщо так, його буде відзначено як крайовий піксель. Якщо це не так, його буде подавлено.

Рисунок 1.13 показує архітектуру алгоритму швидкого граничного виявлення Кенні з використанням усіх розглянутих методів.

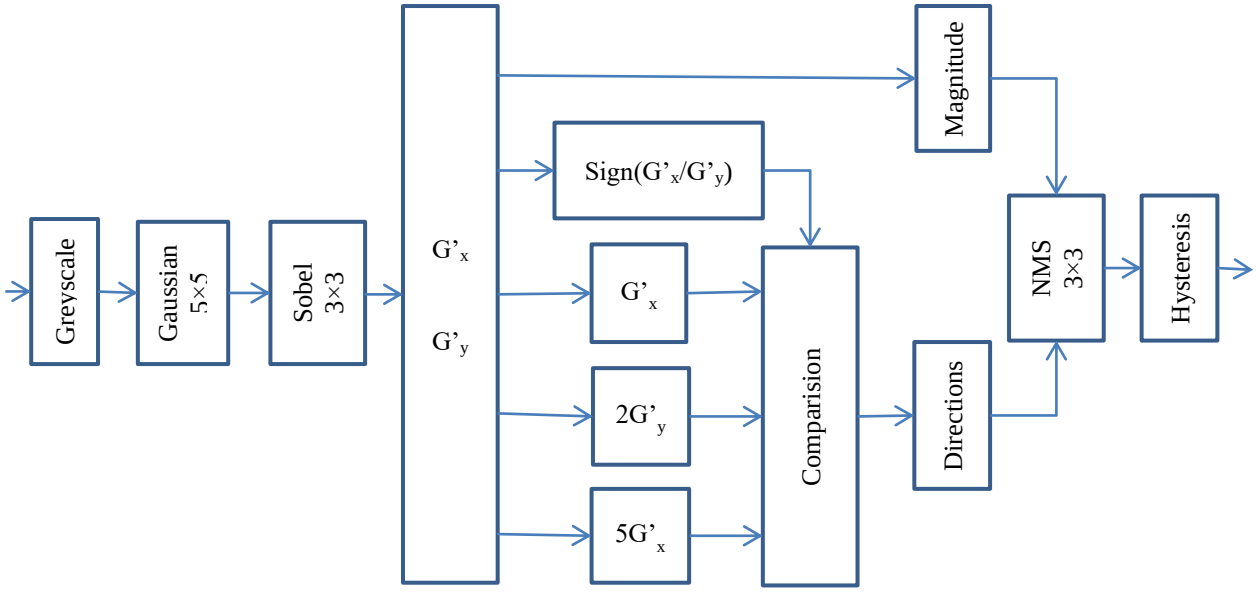


Рисунок 1.13 – Алгоритм швидкого граничного виявлення

1.4 Постановка задачі дослідження

Виявлення дорожньої смуги і її відстеження є однієї з основних характеристик просунутої системи допомоги водієві. Виявлення дорожньої смуги відбувається шляхом знаходження білих маркувальних смуг на темній дорозі. Відстеження дорожньої смуги використовує раніше виявлені маркери дорожньої смуги, і коректує себе згідно з моделлю руху.

З поширенням систем допомоги водієві (ADAS) в сучасних автомобілях світових автовиробників детектування дорожньої смуги стало актуальною задачею обробки відеозображень. Об’єктом цього дослідження є розробка програмно-апаратного засобу знаходження дорожньої смуги з відеоряду, захопленого камерою транспортного засобу. Метою дослідження є реалізація детектування краю фільтром Собеля з подальшим знаходженням границь дорожньої смуги.

Для цього необхідно вирішити такі завдання:

– провести аналіз існуючих методів виявлення дорожньої смуги;

– розробити апаратно-програмну реалізацію детектування країв на відеозображенні дороги на базі алгоритму Собеля;

– реалізувати алгоритм знаходження границь дорожньої смуги на відфільтрованому зображенні.

Апаратним середовищем для реалізації такої задачі може бути система на базі програмованих логічних інтегральних схем FPGA (Field-Programmable Gate Array – програмована користувачем вентильна матриця).

Відстеження смуги є популярною темою в обробці зображень і звичайно включає один або кілька методів попередньої обробки, таких як кутове виявлення, кольорова сегментація, тощо які можуть бути реалізовані на FPGA.

2 МЕТОД ВИЯВЛЕННЯ ДОРОЖНЬОЇ СМУГИ У РЕАЛЬНОМУ ЧАСІ

Обробка зображень і машинний зір стали популярною областю дослідження й в електротехніці й в інформатиці в останні десятиліття. Хоча це було широко застосоване в багатьох полях, промисловість виявила більше інтересу в режимі реального часу обробка зображень в останні роки. Використовуючи класичні послідовні архітектури ЕОМ, це часто забирає час для обробки даних. Щоб досягти продуктивності в реальному часі з низькою затримкою, розподілені системи, такі як багатоядерні центральні процесори GPU, FPGA або ASIC використовуються для поліпшення продуктивності.

За допомогою FPGA можна досягти продуктивності в реальному часі в багатьох додатках, що загальні послідовні процесори забезпечити при використанні таких же методів не можуть [18]. FPGA звичайно мають більш низькі ціни системної інтеграції й обслуговування в порівнянні із центральними процесорами й GPU. Характеристики реконфігурування FPGA забезпечують більше гнучкості, ніж ASIC, які не можуть бути повторно запрограмованими після того, як вони були виготовлені.

В 2010 році Altera, що тепер належить Intel, представила їхню технологію на 28 нм FPGASOC. У цьому поколінні Altera інтегрувала апаратні системи процесора (HPS), які включають двоядерний процесор ARM Cortex-A9 у їхню недорогу номенклатуру виробів FPGA Cyclone V [19]. Це тверде ядро процесора ARM на 925 Мгц є гарною заміною для синтезованого програмно ядра процесора NIOS II на Cyclone IV з максимальною частотою 190 Мгц [19]. FPGASOC комбінує переваги апаратних засобів FPGA для швидкої обробки даних і відносної високоефективної гнучкості вбудованого програмного забезпечення. Шляхом інтеграції обох систем у єдиний компонент досягається зменшення апаратної складності, фізичного розміру, споживаної потужності і вартості усієї системи.

Багато складних додатків для обробки зображень, наприклад, об’єктне відстеження й виявлення поверхні, не тільки засновані на трудомістких алгоритмах згортки, таких як граничне виявлення й кутове виявлення, але також і вимагають, щоб до вихідних неопрацьованих даних були застосовані розрахунки високого рівня. Таким чином, використання FPGA є гарним рішенням для паралельної обробки, яка може зберегти дані вихідного зображення при виконанні низькорівневих алгоритмів згортки, наприклад, граничне виявлення з низькою затримкою. Тому комбінація FPGA і процесора ARM FPGASOC Intel має потенціал для реалізації складних додатків для обробки зображень, у режимі реального часу, на недорогій платі, що робить її придатною для вбудованого приладу обробки зображення.

Розглянемо використання недорогих обладнань FPGASOC для оперативної обробки зображень шляхом розробки системи обробки зображень у реальному часі з декількома методами реалізації алгоритмів попередньої обробки в FPGA. Результати демонструють поліпшену продуктивність, отриману за допомогою об’єднаного FPGA у комбінації з ARM у порівнянні тільки з використанням процесора ARM.

2.1 Метрики продуктивності, які використовуються для оцінки алгоритмів

Для оцінки алгоритмів виявлення й відстеження дорожньої смуги результат виконання алгоритму рівняється з базовим (наземним) дійсним набором даних. Результатом порівняння може бути:

- вірний позитивний результат (TP);
- неправильний позитивний результат (FP);
- неправильний негативний результат (FN);
- вірний негативний результат (TN).

Вірний позитивний результат відбувається, коли існують наземна істина (дорожнє полотно), і вона виявляється алгоритмом. Неправильний позитивний результат відбувається, коли алгоритм виявляє маркер дорожньої смуги, коли насправді його на землі немає. Неправильне заперечення відбувається, коли наземна істина існує в зображенні, але алгоритм пропускає його. Вірне заперечення відбувається, коли немає ніякої наземної істини в зображенні, і алгоритм нічого не виявляє.

Найпоширенішими метриками, використовуваними для оцінки продуктивності алгоритмів виявлення дорожньої смуги, є точність ($accuracy$), повнота або відклик ($recall$), збалансована *F-міра* ($F_{measure}$), точність або правильність ($precision$) [20], криві робочої характеристики приймача (ROC) і коефіцієнт подоби гри в кості (DSC) [21].

Правильність ($precision$) є частина виявлених маркерів дорожніх смуг, які є фактичними маркерами дорожньої смуги:

$$precision = \frac{TP}{TP + FP}. \tag{2.1}$$

Повнота або відклик ($recall$) є частина фактичних маркерів дорожньої смуги, які виявляються:

$$recall = \frac{TP}{TP + FN}. \tag{2.2}$$

F-мірою є міра, яка комбінує правильність і повноту і є середнім гармонійним правильності й повноти:

$$F_{measure} = 2 \cdot \frac{precision \cdot recall}{recision + recall}. \tag{2.3}$$

Точність (accuracy) є мірою того, як добре фактичні маркери дорожньої смуги визначаються правильно:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (2.4)$$

Коефіцієнт подоби гри в кості (DSC):

$$DSC = 2 \cdot \frac{TP}{TP + FP + P}. \quad (2.5)$$

В (2.1) – (2.5) TP є кількістю вірних позитивних результатів, FP є кількістю неправильних позитивних результатів, TN є кількістю вірних негативних результатів, FN є кількістю неправильних негативних результатів, і P є кількістю виявлених смуг.

Крива ROC (рис. 2.1) утворюється шляхом графічного зображення дійсного позитивного рівня (TPR) у порівнянні з неправильним позитивним рівнем (FPR) для різних значень порога добування. Чим більше область під кривою, тем виявлення краще.

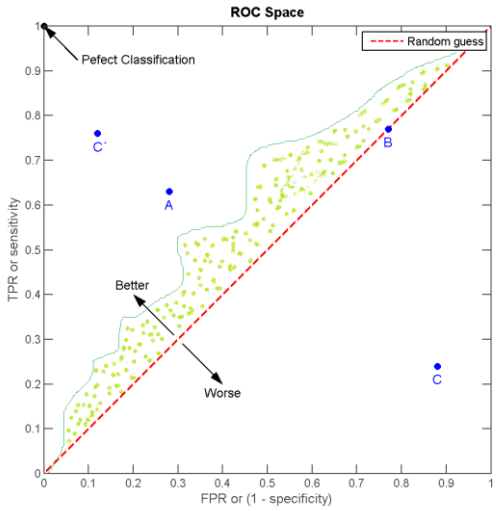


Рисунок 2.1 – Робоча характеристика приймача (ROC)

2.2 Виявлення дорожньої смуги й алгоритми її відстеження

Розглянемо кращі алгоритми виявлення й відстеження дорожньої смуги, які можуть бути обрані для певних дорожніх умов.

Aly [9] представив оперативний і стійкий підхід для виявлення маркерів дорожньої смуги на міських дорогах. Спочатку генерується вид зверху дорожнього зображення за допомогою зворотної перспективи, що відображається для запобігання перспективного ефекту. Вид зверху фільтрований за допомогою вибіркового орієнтованого двовимірного Гаусова фільтра. Фільтр настроюється спеціально для яскравих рядків у темнім тлі з певною шириною. Таким чином, це дає добре реагування на маркери рядка, притому зберігаються тільки найвищі значення шляхом вибору порогового значення квантиля *q%* яркостей фільтрованого зображення й видалення всіх значень нижче порога.

Прямі лінії виявляються за допомогою спрощеного перетворення Хафа, що супроводжується установкою рядка RANSAC, який дає вихідну вірогідність кроку установки сплайну RANSAC. Тоді на черговому кроці робиться локалізація сплайну як маркера й ідентифікація його в зображенні. Алгоритм не виконує відстеження. Метод дозволяє виявити будь-яку кількість границь дорожньої смуги в зображенні, а не тільки поточний маршрут.

Алгоритм виявлення дорожньої смуги, запропонований Zhou і ін. [13], має три модулі: модуль моделі дорожньої смуги, модуль оцінки параметра й модуль оцінки відповідності. Маршрут моделюється за допомогою середнього рядка й трьох діапазонів: дорожньої смуги, вихідної орієнтації й дорожнього скривлення. Параметри в моделі зменшуються за допомогою середньої моделі рядка, який забезпечує стартове місце розташування, орієнтацію й скривлення. Заключні моделі дорожньої смуги одержують, використовуючи ширину дорожньої смуги. Алгоритм фокусується на близькім полі зору. Для оцінки межі параметрів моделі дорожньої смуги

проводиться аналіз структури Габора, який допомагає оцінити орієнтацію кожного пікселя в ROI. Нарешті всі ці піксельні орієнтації голосують за визначення місця розташування межі. Кожний піксель голосує більше ніж за одну межу через присутність артефактів.

Щоб максимізувати ймовірність і одержати єдину межу, використовується Гаусова модель. Після виявлення межі оцінюються ширина й орієнтація дорожньої смуги. Дорожні границі виявляються за допомогою граничного детектора Кенні й перетворення Хафа. У кожного кандидата моделі дорожньої смуги є різне скривлення. Для оцінки скривлення використовується модель, відповідна до алгоритму. Уздовж кожного кандидата використовується фільтр Габора, і кращою відповідністю є кандидат з максимальними голосами.

Borkar і ін. [22] запропонували виявлення дорожньої смуги на основі перетворення Хафа й виконання ітерації підібраних фільтрів. Алгоритм RANSAC використовується для запобігання викидів, спричинених шумом в відеосигналі і артефактами на дорозі. Для відстеження дорожніх смуг використовується фільтр Калмана. На першому кроці алгоритму необхідно перетворити кольорове зображення в шкалу півтонів і застосовати тимчасове розмивання на зворотнім перспективнім відображенні (inverse perspective mapping – IPM) зображення. Для генерації дворівневого зображення IPM застосовується адаптивний поріг.

Кожне дворівневе зображення розділяється на дві половини, кожна з яких містить маркер однієї смуги. До дворівневого зображення застосовується перетворення Хафа з низьким порогом. Для кожного зразка зображення для знаходження приблизного центру кожного рядка застосовуються 1-мірні підібрані фільтри уздовж рядка. Після оцінки центру для виявлення дорожньої смуги до точок даних застосовується алгоритм RANSAC.

Алгоритм, запропонований Liu і ін. [23], має два кроки, початкове виявлення дорожніх смуг і їх наступне відстеження. Ефект перспективи

вилучений із зображення за допомогою зворотного перспективного відображення. Потім на зображення IPM для виявлення дорожньої смуги застосовується статистичне перетворення Хафа (statistical Hough transform – SHT). SHT працює над зображеннями інтенсивності з використанням декількох щільностей ядра для опису змінних Хафа (*p, 0*).

Оскільки SHT оброрблює кожен піксель у зображенні, це в обчислювальнім відношенні дорого й не підходить для виконання його на кожному кадрі. Після цього використовується фільтр часток, щоб відстежити виявлені маршрути й обновити параметри моделі дорожньої смуги. Параметри моделі дорожньої смуги утворюються і обновляються кадр за кадром. SHT є в обчислювальнім відношенні дорогим, він об'єднаний з використанням фільтра часток. Для демонстрації алгоритму використовується пряма модель дорожньої смуги.

Liu і ін. [24] запропонували поліпшення традиційного фільтра часток [23]. Воно підходить для лінійно-параболічної моделі дорожньої смуги. Спочатку утворюється зворотне перспективне відображення (IPM) зображень. Використовуються певні дескриптори, такі як колір, положення, градієнт для виявлення маркерів дорожньої смуги в зображенні IPM. Імовірнісний розподіл параметрів прямої лінії моделюється як щільність мультиядра, і із використанням статистичного перетворення Хафа (statistical Hough transform – SHT) знаходяться три кандидати на маркери дорожньої смуги. SHT може бути розширений для виявлення параболічних структур. Вертикальні краї визначаються за допомогою граничного добування Собеля (рис. 2.2). Кількість часток залежить від розміру частки. Алгоритм є ієрархічним.

Спочатку оцінюється лінійна частина зображення, потім параболічна частина. Для лінійної частини приділяється більше часток, ніж для параболічної частини. Для створення алгоритму в кожнім повторенні створюється стійкий постійний відсоток зразків ініціалізації. Оцінка кожної частини робиться за два кроки. Спочатку стан передвіщається за допомогою

імовірнісної моделі випадкового блукання. На другому кроці здійснюється перемасштабування зображення. Зберігаються тільки частки з високими значеннями ваги. Якщо маршрути в полі бачення мають високе скривлення, то помилка оцінки збільшується.

Рисунок 2.2 – Результати обробки зображення крайовим детектором Собеля

Guo і ін. [25] пропонують метод, де вхідне зображення спочатку перетворюється у зворотне перспективне зображення. Потім паралельно по обидва боки зображення проводиться багатошкальне виявлення дорожньої смуги. Знаходиться подоба відповідних пікселів дорожньої смуги за допомогою нормалізованої взаємної кореляції (normalized cross correlation – NCC). Щоб перевірити, чи пофарбовані виявлені маркування дорожньої смуги чи ні, використовується алгоритм машинного навчання. Класифікатор ANN із двома шарами й сьома схованими вузлами застосовується на маленький патч вікна зображення 9 × 3 пікселя навколо кожної збереженої дорожньої розмітки. При інтеграції інтенсивності й сигналів геометрії створюється зважений графік з вагою, відповідним до ймовірності пікселя

бути точкою дорожньої смуги. Границя дорожньої смуги оцінюється за допомогою фільтра часток й моделі сплайнів Catmull Rom. Це підходить для кривих дорожніх смуг, змін дорожньої смуги, поділів і об'єднань маршрутів.

Borkar і ін. [26] розробили метод на основі паралельної природи маркерів дорожньої смуги. Спочатку виконується перше зворотне перспективне відображення. Потім зображення IPM перетворюється в півтонове зображення. Зображення фільтрується за допомогою нормалізованої взаємної кореляції. Визначається набір прямих ліній за допомогою полярного рандомізованого перетворення Хафа (polar randomized Hough transform – RHT). Кожне оптимальне припасування рядка має високі координати виграшу або піки в просторі ($p, 0$). Визначається, чи є два рядки паралельними піками рядків з ідентичними нульовими значеннями.

Звичайно маркери дорожньої смуги паралельні, але через недоліки в оптиці, зміни зображення в розміщеннях маркера дорожньої смуги тощо зображення не можуть завжди бути паралельними. Таким чином, необхідно ослабити вплив ідентичних нульових значень в зображенні. Цього можна досягти шляхом застосування вікна допуску. Відео тестується в режимі реального часу. При такому підході значно зменшуються труднощі у виявленні дорожньої смуги, такі як присутність тіней, що граничать із транспортними засобами й дефектами поверхні дороги. Також в методі існують труднощі при виявленні зношених маркерів дорожньої смуги.

Tran і ін. [27] використовують горизонтальне виявлення рядка підобласті присутності дорожньої смуги в зображенні, який перебуває під поточним горизонтальним рядком. Спочатку використовується вертикальний метод розподілу середніх для визначення першого мінімуму у верхній кривій. Це мінімальне положення розглядають як горизонтальне положення рядка. У денний час шукають мінімум, у нічний максимум уздовж вертикальної середньої кривої. Потім на нижньому краї зображення застосовується детектор Кенні.

На основі зафіксованого порога можуть бути визначені маршрути й шумовий кандидат у граничнім зображенні. Після цього для виявлення дорожніх смуг використовуються кластеризація методом k-середніх і алгоритм RANSAC. Після того, як отримані ліві й праві границі дорожньої смуги, точка перетинання між ними використовується в якості горизонтального положення рядка для наступного кадру послідовності. Алгоритм орієнтований тільки на поточний маршрут. Коли існує перетинання смуг, виявлення смуг не коректне.

Y.-C. Leng, і ін. [28] пропонують систему виявлення дорожньої смуги для міських доріг. Краї виявляються за допомогою оператора Собеля. Для виявлення прямих ліній використовується перетворення Хафа. У дорожньому зображенні маршрути перетинаються. На різній висоті ширини зображення дорожніх смуг відрізняються. Мінімальна ширина дорожньої смуги визначається як w_{min} і максимальна ширина дорожньої смуги як w_{max}. Ширина дорожньої смуги в кожному регіоні w_i $(i = 0,1 \ldots 4)$ повинна завжди бути між w_{min} і w_{max}. Для кожного лівого й правого кандидата дорожньої смуги відповідність робиться на основі ширини кожної пари кандидата. Якщо ширина пари в різній висоті не задовольняє критеріям, то пари усувається. Після цього виявлення лівого й правого граничного сходів з дорожньої смуги може бути визначене положенням границі дорожньої смуги.

Схожий підхід виявлення дорожньої смуги для міських доріг продемонстрували R. Gopalan і ін. [29] та G. Liu і ін. [30].

Yoo і ін. [31] генерують максимізоване зображення градієнта дорожньої смуги з кольорового зображення на основі лінійного дискримінантного аналізу. Це утворює сильні краї для границі дорожньої смуги в різнім освітленні. Потім застосовується граничний детектор Кенні для одержання граничного зображення. Після граничного виявлення застосовується перетворення Хафа для одержання початкового виявлення дорожньої смуги. Перетворення Хафа не може представити вигнуті маршрути, тому для виявлення кривих дорожніх смуг використовується

добір кривих. Три параметри квадратичної кривої використовуються для оцінки моделі дорожньої смуги. Для першого кадру навчальні дані задаються вручну, а для кадрів, що залишилися, навчальні дані обновляються для адаптації змін освітлення.

Daigavane і ін. [32] розробили метод виявлення дорожньої смуги на основі поліноміального алгоритму для знаходження наближених розв'язків, який має назву «оптимізації колонії мурах» (ant colony optimization – ACO). Спочатку, вхідне зображення стискається до 255 × 255 пікселів для скорочення часу обчислення. Три зображення RGB каналу перетворюються в єдине півтонове зображення каналу. Потім використовується фільтр усереднення, щоб відфільтрувати шум і зберегти краї. Потім краї виявляються за допомогою граничного детектора Кенні.

Вихідне дворівневе зображення дане як вхідне до ACO. Це генерує додаткову інформацію про край, яка загублена в граничнім виявленні Кенні. ACO використовує багато «мурах», які переміщаються на основі зміни інтенсивності в зображення. Використовуючи імовірнісний підхід, переміщається кожна «мураха», поки вона не досягає іншого лінійного сегмента. Після того, як краї виявляються правильно з’єднаними, для виявлення прямих ліній використовується перетворення Хафа. Рядки, що відповідають найвищим пікам у просторі Хафа, позначаються як маршрути.

Yim [11] розробив три засновані на функції алгоритми виявлення дорожньої смуги. Використаними функціями є стартова позиція, орієнтація й значення інтенсивності. На початковому кроці оператор Собеля застосовується для одержання граничної інформації. Границя дорожньої смуги представлена як включення вектора із цих трьох функцій. Поточний вектор дорожньої смуги обчислюється на основі вхідного зображення й попереднього вектора моделі дорожньої смуги. Два вікна, один для кожного, використовуються для лівих і правих границь. Ухвалюється N пікселів у кожному горизонтальному рядку, потім генерується N кандидатів вектора дорожньої смуги. Кращий кандидат вибирається на основі мінімальної

відстані від попереднього вектора дорожньої смуги з використанням зваженої метрики відстані. Для корекції кожній функції привласнюють різну вагу. Тоді система виводу дорожньої смуги використовується для передбачення нового вектора дорожньої смуги. Якщо дорожня ширина змінюється поточний різко обчислений вектор відкидається, а попередній береться за поточний вектор.

Lin і ін. [33] запропонували метод на основі видимої області (ROI). ROI, спочатку ініціалізований оператором Собеля, результат застосування якого зображений на рисунку 2.2, поряд з знаходженням локальних максимумів, використовується для знаходження крайових пікселів. Після одержання граничного зображення створюється розширений край, що зв'язується на основі спрямованого граничного закриття розриву, для з'єднання тих пунктирних ліній у далекім полі бачення, які ушкоджуються через субдискретизацію. Розгорнення растра виконується від нижньої частини зображення до початкової точки нового краю. Виконується граничне трасування і піксель, що утворюється, додається до краю шириною у вісім пікселів. Для даної початкової точки трасування робиться в одній орієнтації.

Наступний крок спрямований на граничне закриття розриву. Граничні посилання розширюються шляхом додавання користувацької конкретної кількості пікселів уздовж орієнтації для заповнення розривів. Нові пікселі вибираються з оточення початкових й кінцевих точок. Після цього усуваються краї з довжиною менше ніж 15 пікселів. Після цього розглядають дві граничні пари посилань, і якщо відстань між ними задовольняє ширину мітки дорожньої смуги, те це розглядається як регіон мітки дорожньої смуги. Наступний крок є перевіркою гіпотези дорожньої смуги, для цього перевіряється колір міток дорожньої смуги. Після того, як кандидати визначені, використовується перетворення Хафа для визначення значень параметрів p і 0.

Ozgunalp і ін. [34] використовують алгоритм використання симетричного локального порога (symmetrical local threshold – SLT). Він

заснований на властивості переходу темрява-світло-темрява дорожніх смуг. Для кожної точки введення обчислюється середнє значення інтенсивності всіх пікселів ліворуч (у діапазоні) у тому ж рядку, потім середня інтенсивність обчислюється для правої сторони. Якщо інтенсивність пікселя більше, ніж ліві й праві середні, тоді це розглядають як характерну точку дорожньої смуги й маркірують у карті характерних точок. Потім застосовується зв'язаний факторний аналіз. Сторони дорожньої смуги шукаються по характерних точках границі. Якщо орієнтації обох сторін близькі одне до одного, то орієнтацію виявленої границі дорожньої смуги обчислюють як середнє кутів сторін. Інакше точка усувається з карти характерних точок.

Для зменшення шуму використовується перетворення відстані. Відстань кожної характерної точки від найближчої не характерної точки в карті функції обчислюється з ваговими коефіцієнтами. У такий спосіб маркери дорожньої смуги мають більш високу вагу, ніж ізольовані пікселі. Також центр маркера дорожньої смуги має високе значення. Для збільшення швидкості обчислення застосовується перетворення Хафа. У кадрі виявляються й прослідковуються й ліва, і права смуги.

Zhao і ін. [12] пропонують метод, який використовує фільтр часток з імітацією відпалу для виявлення дорожньої смуги й відстеження. В такому фільтрі процедура пошуку наближеного глобального оптимуму імітує фізичний тепловий процес відпалу. Він часто використовується при наявності великої кількості локальних оптимумів, коли пошук наближеного глобального оптимуму важливіший, ніж пошук точного локального оптимуму за встановлений проміжок часу. Фільтр використовується, оскільки маркер дорожньої смуги може бути описаний як об'єкт чорна-біла-чорна смуга. Маркери дорожньої смуги звичайно мають білий або жовтий колір, тому колірний сигнал також використовується. Для кращого розрізнення зображень RGB кольори перетворюються в простір HSV.

Ці сигнали обробляються окремо перед використанням фільтра часток з імітацією відпалу. Замість єдиного врахованого кроку в стандартному фільтрі часток цей фільтр фільтрує зроблений крок у кожному кадрі у відео послідовності. Використовується стаціонарна модель руху. Частки переміщаються в регіон високої ймовірності. На кроці виправлення інформація про кут граничної карти використовуються ваги часток. Фільтрація часток має три основні кроки, включаючи перемасштабування, виправлення й нормалізацію ваги. Для виявлення дорожньої смуги й відстеження використовується стійка модель дорожньої смуги. Може бути застосований і до прямих, і до кривих доріг.

Lin і ін. [35] виконують виявлення дорожньої смуги на основі імітації інгібування (властивостей затримки) бічного бачення людини. Це дозволяє алгоритму бути стійким у різних погодних умовах. Алгоритм не заснований на граничній обробці. Він використовує 2D і 3D геометричні дані маркерів дорожньої смуги. Для виявлення дорожньої смуги використовується позитивна й негативна карти відмінності у зображеннях. Це забезпечує сильний контраст між маркером дорожньої смуги й дорогою. Подвійна Гаусова модель використовується, щоб визначити два піки, відповідних до глобального максимуму на правій стороні лівого маркера дорожньої смуги й глобальному мінімуму на лівій стороні лівого маркера дорожньої смуги. Після цього виконується перевірка для оцінки правильності виявленого маркера дорожньої смуги. Перевірка робиться для перевірки нахилів і відносин переривання, фактичної дорожньої ширини й положення межі.

В [35] для виявлення країв в ROI (region of interest) використовується граничний детектор Кенні. Прямі лінії виявляються від двійкового виходу граничного детектора Кенні за допомогою перетворення Хафа. Для усунення шумових ефектів локальні максимуми функцій шукаються уздовж передбачуваної границі дорожньої смуги. Потім для усунення викидів застосовується алгоритм RANSAC. Заключні локальні максимуми функцій

уписуються в пряму лінію. У кадрах, що залишилися, для відстеження дорожніх смуг використовується фільтр Калмана.

Bottazzi і ін., [36] пропонують метод виявлення дорожньої смуги, заснований на гістограмі інваріанта освітлення. Динамічна видима область (dynamic region of interest – DROI) визначається за допомогою попередньої трикутної моделі. Спочатку обчислюється гістограма цілого зображення й дорожнього кадра. Відмінність між ними використовується для виявлення змін освітлення. Від ROI виділяються маркери дорожньої смуги. Алгоритм використовує фільтр Lucas Kanade для відстеження дорожніх смуг.

Wang і ін. [37] використовують повний оптимальний поріг, що перетворює вхідні дані у двійковий файл. Для запобігання ефекту перспективи робиться зворотне перспективне відображення. Потім проводиться кластеризація N зразків у файлі методом k-середніх [5]. Для одержання маркера дорожньої смуги використовується B-сплайн, застосований до центрів кластерів.

ROI можна розділити на близьке поле бачення й далеке поле [38] бачення. Вхідне зображення перетворюється в зображення IPM. Локальне перетворення Хафа застосовується на близьке поле зору для виявлення прямих ліній. У далекім полі бачення використовується поліпшений метод потоку для розширення точок, виявлених у близькім полі бачення або рядку кривої, виявленої в попередньому кадрі. Поточна точка є головною точкою у виявленій прямій лінії. Виявлені характерні точки вписуються в модель гіперболічної інтерполяції за допомогою RANSAC.

Kim [39] розробив алгоритм виявлення й відстеження дорожньої смуги, який може обробити складні сценарії, такі як зниклі маркери дорожньої смуги, скривлення дорожньої смуги й маршрути поділу. На початковому кроці використовуються детектор градієнта й детектор різкої зміни інтенсивності для усунення не маркерів дорожньої смуги. На зразки, що залишаються, для виявлення дорожньої смуги застосовуються штучні нейронні мережі (Artificial Neural Networks – ANN). Виявлені пікселі

маркерів дорожньої смуги згруповані за допомогою кубічних сплайнів. Гіпотези генеруються від випадкового набору лінійних сегментів. Алгоритм RANSAC допомагає в перевірці гіпотез. Для відстеження дорожньої смуги використовується фільтрація часток.

Jung і ін. [40] пропонують алгоритм виявлення дорожньої смуги, який використовує функції Хаара для одержання кандидатів точок дорожньої смуги. Зображення розділене на два прямокутні регіони. Використовуються спрямовані по діагоналі керовані фільтри, оскільки маркер дорожньої смуги здається діагональним через ефект перспективи. Наближений керований фільтр використовує функції Хаара, і утворюється максимальний відгук. Ліві й праві маршрути обчислюються. Оскільки маршрути паралельні, вони будуть сходитися на межі. Ця гіпотеза перевіряється для перевірки правильності виявлених дорожніх смуг. Після цього може бути визначений схід з дорожньої смуги на основі відстані між межею й горизонтальним центральним рядком. Збільшення відстані показує схід з дорожньої смуги.

Tan та ін. [41] розробили метод для виявлення й класифікації дорожньої смуги. Маркери дорожніх смуг виявляються на основі лінійної параболічної моделі. Пікселі маркера дорожньої смуги, як передбачається, мають більш високу інтенсивність, ніж пікселі на тротуарі. Маленький прямокутний патч використовується для добування статистичних залежностей між пікселями маркера дорожньої смуги й пов’язаними із тротуаром пікселями. У кожному кадрові несуперечність кожного пікселя в патчі, відповідному до маркера дорожньої смуги з розподілом пікселів тротуару, перевіряється для розрізнення тротуару й пікселів маркера дорожньої смуги. Після цього використовується каскадний класифікатор для класифікації маркера дорожньої смуги. Чотири бінарні класифікатори використовуються для класифікації виявленого маркування дорожньої смуги в п’ять класів: штрихова смуга, штрихова-суцільна смуга, підкреслена штриховою лінією одиночна смуга, одиночна смуга, подвійна смуга.

Tsai, і ін. [42] пропонують новий граничний алгоритм для виявлення дорожньої смуги. Він заснований на локальному напрямку градієнта. Граничний детектор Собеля використовується для виявлення величини градієнта й напрямку. Потім дві кругові маски, а саме, відстежувальна маска й зондувальна маска, використовуються для збору обмежених зразків орієнтації градієнта. Початковий градієнт визначається на основі найбільшої гістограми для орієнтації. Зондувальна кругова маска використовується для запобігання сходу зі смуги. Також використовується фільтр на основі багаточлена третього порядку.

Метод виявлення дорожньої смуги, що підходить для використання вночі, запропонований Borkar і ін. [43]. Алгоритм спочатку ініціалізує видиму область (region of interest – ROI), що усуває небо й інші невідповідні об’єкти. На наступному кроці півтонове зображення отримане шляхом усереднення цих трьох колірних каналів. Тимчасове розмивання використовується для подовження штрихових дорожніх смуг. Для добування об’єктів високої яскравості робиться адаптивна гранична обробка. Результуюче дворівневе зображення розділене на ліві й праві половини. Потім на кожну половину для виявлення прямих ліній застосовується перетворення Хафа з низьким дозволом. Для добування маркерів дорожньої смуги використовується Гаусове ядро з підібраним за допомогою ітерацій фільтром.

Підхід виявлення дорожньої смуги для міського середовища запропонований Sehestedt і ін. [44]. Тому що маркери дорожньої смуги не явно видимі через зношування, поглинання газів і через складну дорожню геометрію, для виявлення маркерів дорожньої смуги використовується слабка модель. У відображенім зображенні зворотної перспективи застосовується фільтр часток від нижнього ряду до вершини. Фільтр настроюється таким способом, щоб відстежити кілька дорожніх смуг.

Спочатку кольорове зображення перетворюється в шкалу півтонів [45], а потім виконується видалення тіней. Змінений граничний детектор Кенні

використовується для граничного виявлення, і потім застосовується перетворення Хафа. Периферійне сканування дорожньої смуги рядка обрію реалізоване за допомогою граничного результату виявлення перетворення Хафа. Сканування починає із частини, де рядок Хафа зіставляється з нижньою границею кольору зображення. На зібраних точках даних проводиться гіперболічна інтерполяція.

Cheng і ін. [46] представили ієрархічний алгоритм для виявлення дорожньої смуги. Високі розмірні характерні точки видобуваються на основі добування функції кольору. Це використовується для розрізнення структурованих доріг від неструктурованих доріг. Тоді зв'язані компоненти застосовуються на характерні точки. Для характерних точок, що перевищують поріг створюється характеристичний вектор. Власне розкладання значення впорядкованого дискримінантного аналізу зроблене для скорочення моделі сходу з маршруту. Тоді найбільша правдоподібність оцінюється по Гаусовим параметрам. Витягнуті характерні точки використовуються в якості виявлених дорожніх смуг на структурованих дорогах. Для неструктурованих доріг уся сцена розділена на основі сегментації середнього зрушення. Кожний регіон уважається гомогенним, тому за допомогою правила Байеса виявляються маркери дорожньої смуги.

Таблиця 2.1 підсумовує й представляє докладний аналіз [47] алгоритмів виявлення дорожньої смуги й відстеження дорожньої смуги.

Таблиця 2.1 –Алгоритми виявлення й відстеження дорожньої смуги

Автор	**На основі якої моделі**	**Виявлення**	**Відстеження**	**Оцінка**	**Коментарі**
Y. U. Yim і S.-Y. ПРО (2003р.) [11]	Оператор Собеля	Перетворення Хафа й три характеристичні вектори	Тимчасовий предиктор для передбачення поточного вектора дорожньої смуги	Добре працює для дощової й тіньової дороги	Інформація про дорогу не потрібна
S. Sehestedt, і ін. (2007р.) [44]	Зворотне перспективне відображення	Слабкий заснований на моделі метод	Кластеризований фільтр часток	Стійкий у важких умовах освітлення	Підходящий і для прямих й для незначно вигнутих міських доріг

Продовження таблиці 2.1

Автор	На основі якої моделі	Виявлення	Відстеження	Оцінка	Коментарі
Z. Kim (2008р.) [39]	Граничний детектор, детектор перепаду інтенсивності	Штучна нейронна мережа (ANN)	Фільтр часток	Стійкий у присутності провідних механізмів і низького освітлення	Підходящий і для прямих й для вигнутих доріг
M. Aly (2008р.) [9]	Зворотне перспективне відображення, вибіркові орієнтовані Гаусові фільтри	Перетворення Хафа й установка сплайна RANSAC		Порівнянні результати до алгоритмів виявлення й відстеження	У присутності рядків зупинки в перехресних обходах, сусіднє виявлення механізмів
H. Y. Cheng і ін. (2008р.) [46]	На основі кольорового добування	Власне значення декомпозиції n упорядкований дискримінантний аналіз		Стійкий у різних погодних умовах	Підходящий для структурованих і неструктурованих доріг
A. Assidiq і ін. (2008р.) [45]	Граничний детектор Кенні	Перетворення Хафа й установка гіперболи		Стійкий у тінях	Підходящий для пофарбованих і непофарбованих кривих і прямих доріг
C.W. Lin і ін. (2009р.) [10]	Вертикальний граничний детектор	Позитивні й негативні другі карти відмінності		Стійкий у різних погодних умовах і присутності тіней	Крок перевірки зменшує неправильні позитивні сторони
A. Borkar і ін. (2009р.) [43]	Адаптивна гранична обробка	Низький дозвіл перетворення Хафа й підібраний фільтр		Стійкий у нічний час	Не підходящий для операції денного часу
A. Borkar і ін. (2009р.) [22]	Зворотне перспективне відображення, адаптивна гранична обробка	Перетворення Хафа й виконання ітерації підібраного фільтра	Фільтр Калмана	Стійкий у шумі й артефактах на дорозі	Відсутність маркерів дорожньої смуги приводить до неправильного виявлення й відстеження
G. Liu і ін. (2010р.) [23]	Зворотне перспективне відображення, адаптивна гранична обробка	Статистичне перетворення Хафа	Фільтр часток	Витратний по обчисленнях	Використовується пряма дорожня модель
C. Guo і ін. (2010р.) [25]	Каскадний аналізатор дорожньої смуги	Сплайни Catmull Rom	Фільтр часток на основі зваженого графіка	Стійкий у різнім висвітленні й погодних умовах.	100 зразків генеруються для обох лівих і правих дорожніх смуг відповідно
Lin і ін. (2010р.) [33]	Оператор Собеля з не максимальним придушенням	Спрямоване граничне закриття розриву й перетворення Хафа		Адаптивний до різних дорожніх умов	Попередження про схід зі смуги включене
S. Zhou і ін. (2010р.) [13]	Модель смуги отримана на основі параметрів камери	Заснований на фільтрі Габора алгоритм відповідності дорожньої смуги		Стійкий у шумі й тінях	На основі плоского дорожнього припущення

Продовження таблиці 2.1

Автор	На основі якої моделі	Виявлення	Відстеження	Оцінка	Коментарі
P. Daigavane і P. Bajaj (2010р.) [32]	Граничний детектор Кенні і оптимізація «Колонії мурах	Перетворення Хафа		Не стійкий у тінях	Підходящий для намальованих і прямих доріг
Y.-C. Leng і C.-L. Chen (2010р.) [28]	Оператор Собеля	Перетворення Хафа		Успішне виявлення в старім дорожньому покритті, знаках, попереджаючих рядках й при струсі зображення	Підходящий для міських доріг
A. Borkar і ін. (2011р.) [26]	Порівняння із шаблонами	Полярне рандомізоване перетворення Хафа		Не стійкий у зношеному маркері дорожньої смуги	Підходящий для прямих доріг
T. T. Tran і ін. (2011р.) [27]	Граничний детектор Кенні	Кластеризація *k*-середніх і RANSAC		Увага на поточний маршрут	У пересічних маршрутах неточне виявлення
R. Gopalan і ін. (2011р.) [29]	Піксельний дескриптор функції ієрархії	Вивчення заснованого виявлення	Фільтр часток	Стійкий у тінях і поглинанні світла	На основі статичної моделі руху
G. Liu і ін. (2011р.) [24]	Колір, положення й дескриптори градієнта й оператор Собеля	Статистичне перетворення Хафа	Розділений фільтр часток	Витратний по обчисленнях	Підходящий і для прямих, і для вигнутих доріг
M.Tan і ін. (2013р.) [41]	Засноване на патчі виділення ознак	На основі лінійних параболічних зразків і геометричних обмежень	Фільтр часток	Стійкий у тінях і зміні освітлення	Класифікація маркерів дорожньої смуги включена
S. C. Tsai і ін. (2013р.) [42] G. Liu і ін. (2013р.) [30]	Оператор Собеля з фільтром	Трасування кругової маски й статистичне перетворення Хафа	Зондування кругової маски й багаточлен, відповідний до розділеного фільтра часток	Працює на розмитих і ушкоджених дорогах, стійкий у тінях і поглинанні світла	Підходящий і для прямих, і для вигнутих доріг
H. Zhao і ін. (2013р.) [12]	Фільтрація панелі й заснована на кольорі сегментація	На основі стійкої моделі дорожньої смуги	Фільтр часток з імітацією відпалу	Подібний результат як фільтр часток, але час обчислення зменшується	Підходящий і для прямих, і для вигнутих доріг
H. Yoo і ін. (2013р.) [31]	Адаптивний граничний детектор Кенні	Перетворення Хафа й вигин зразкової установки		Стійкий у змінах освітлення	Підходящий і для прямих, і для вигнутих доріг
H.Jung і ін. (2013р.) [40]	Керований фільтр	Функції Хаара		Стійкий у змінах освітлення	Попередження про схід зі смуги включене
J.Wang і ін. (2014р.) [37]	Сегментація	Кластеризація *k*-середніх і застосування *B*-сплайна		Не сприйнятливий до ефекту взаємодії	Міське виявлення дорожньої смуги

Кінець таблиці 2.1

Автор	На основі якої моделі	Виявлення	Відстеження	Оцінка	Коментарі
H. Tan й ін. (2014р.) [38]	Поліпшений річковий потік	Перетворення Хафа		Стійкий у поглинаннях світла транспортного засобу	Підходящий і для прямих, і для вигнутих доріг
U. Ozgunalp і N. Dahnoun, (2014р.) [34]	Симетричний локальний поріг	Перетворення Хафа	Фільтр Калмана	Стійкий у тінях уночі	Підходящий і для прямих, і для вигнутих доріг
Y.Li і ін. (2014р.) [35]	Граничний детектор Кенні	Перетворення Хафа	Фільтр Калмана	Низька продуктивність в інтенсивному русі дорожніх структурах, що заплутують, і нерівнім висвітленні	Підходящий для прямих доріг
V. S., Bottazzi і ін. (2014р.) [36]	Гістограма	Сегментація	Фільтр Lucas-Kanade	Стійкий у змінах освітлення	На основі трикутної моделі

Проведений аналіз різних алгоритмів виявлення й відстеження дорожньої смуги показує, що вибір конкретного алгоритму залежить від особливостей умов, в яких передбачається використання просунутої системи допомоги водієві.

2.3 Огляд і порівняння існуючих апаратно-програмних технологій

2.3.1 Центральний процесор, графічний процесор і FPGA

Різні типи архітектури використовуються для платформ обробки зображень у реальному часі. Одна з популярної архітектури використовує ПК (персональний компьютер) у якості хост-системи. Цей підхід був дуже популярною архітектурою для оперативної обробки зображень протягом декількох десятиліть. Кожний функціональний апаратний модуль може бути реалізований окремо через інтерфейс розширення, наприклад, камера USB або відеокарта.

Оскільки ця технологія була розроблена за десятиліття, існує багато додатків, які використовуються на ПК. Наприклад, Kang і Doraiswami розробили систему, яка використовувала інтерфейсну плату USB і веб-

камеру, дозволяючи ПК записати відео для додатків [48]. Система реалізувала кілька модулів обробки зображень включаючи контрастне поліпшення, граничне виявлення Кенні й перетворення Хафа на ПК за допомогою MATLAB для оперативної обробки зображень.

Багато комп'ютерних систем бачення тепер використовують підхід програмного забезпечення, за допомогою універсального центрального процесору (ЦП) і графічного процесору (GPU) для виконання всіх задач обробки зображень. Зі швидким збільшенням можливостей обробки GPU підхід програмного забезпечення тепер пропонує більшу можливість обробки впоратися з більш складними задачами обробки зображень у режимі реального часу.

Balfour і ін. [49] представили засновану на відео систему керування замкненого циклу для контролювання зварних швів на промисловому підприємстві, де ПК, пов'язаний з відеокамерою й відеокартою, використовувався для виконання захоплення відео в реальному часі, аналізу зображення, відображення й керування процесом. Однак вони не ідеальні для компактних вбудованих систем бачення, оскільки це вимагає головного комп'ютера й відеокарти й тому приводить до потужного енергоспоживання.

2.3.2 FPGA у порівнянні з центральним процесором і графічним процесором

FPGA звичайно розглядають як паралельне обладнання низького рівня, яке забезпечує гнучкість так, щоб вона могла бути запрограмована для реалізації будь-якого логічного каналу. Це також іноді застосовується до проектів прототипів мікросхем ASIC.

Розробка систем на FPGA вважається важчою при порівнянні з розробкою на GPU і центральних процесорах, оскільки системи FPGA повинні бути розроблені за допомогою апаратних мов [50], що є проблемою для розробників, які володіють тільки високорівневими мовами

програмування загального використання. Найпоширенішими апаратними мовами для FPGA є Verilog [51] і VHDL [52], які називають мовами опису апаратних засобів (HDL). Основна відмінність між цими мовами й традиційними мовами програмного забезпечення – те, що HDL описують апаратні засоби, тобто регістри й функції Булевої логіки, тоді як мови програмного забезпечення, такі як C, описують послідовні інструкції, не знаючи про точні деталі апаратної реалізації. У результаті дослідники схильні вибирати розробку програмного забезпечення через відносну простоту розробки й чистої кількості доступних абстракцій і класифікацій, які значно підвищують продуктивність.

Центральні процесори й GPU більш ранньої розробки спільно використовували велику кількість установлених бібліотек, які мають багато ресурсів для розроблювачів для продуктивної реалізації різних задач. Недоліком FPGA є обмежена кількість бібліотек IP і функцій компонента проектів користувачів.

Значним елементом, який впливає на вибір FPGA, центральних процесорів і GPU, є структура проекту. Підходяща архітектура для цільового додатка може значно поліпшити продуктивність і зменшити системну вартість. І ЦП, і GPU, рисунок 2.3, мають блоки керування, блоки ALU (Арифметико-логічне обладнання), DRAM (Динамічна оперативна пам’ять) і кеш. Але відмінністю між ЦП і GPU є значна частка мікросхеми GPU, яка відводиться під блоки ALU.

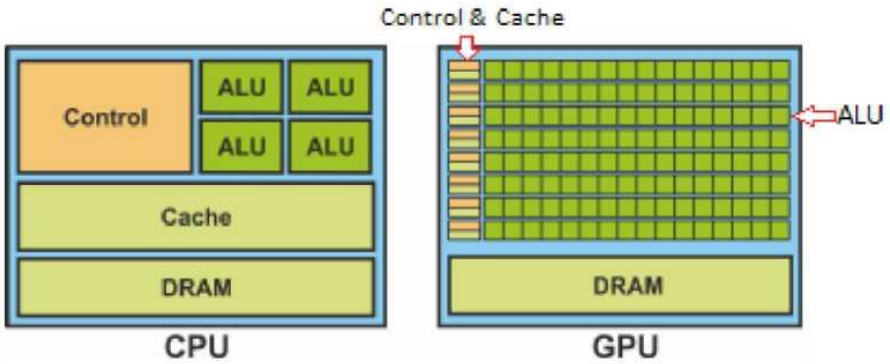


Рисунок 2.3 – Порівняння структур ЦП і GPU

Для обробки зображень, яку можна розглядати як 2D обробку сигналів, потрібна пропускна здатність пам’яті для операцій, що інтенсивно використовують пам’ять [50].

FPGA показують гарну продуктивність при розпаралелюванні процесів. Тому можна реалізувати багато-паралельну архітектуру [53], її розрахункові результати можуть використовуватися безпосередньо до наступного етапу обробки без тимчасового зберігання в оперативній пам’яті.

Отже, вимога для пропускної здатності пам’яті набагато нижче ніж тоді, коли реалізоване з GPU або ЦП. Тому така система має кращу продуктивність у додатках для обробки зображень, оскільки її алгоритми можуть використовувати більший ступінь паралелізму [53].

Відмінності між центральними процесорами й GPU викликаються їхніми різними цілями дизайну, оскільки вони призначені для двох різних сценаріїв. Центральні процесори є універсальними для обробки різних типів даних. Це робить внутрішню структуру ЦП дуже складною.

Відповідаючи ядрам обробки і ЦП, і GPU, програмувальний блок FPGA більш простий. Які апаратні засоби вибрати залежить від складності вирішуваної задачі. Наприклад, якщо оперативна система обробки зображень вимагає високого дозволу й складних обчислень, GPU більш підходить для роботи. Але якщо система тільки вимагає заснованих на згортках низькорівневих алгоритмів, FPGA стають більш підходящими.

Існують різні елементи, які потрібно розглянути при оцінюванні, як працює система. Наскільки продуктивна обробка зображень у реальному часі, значно залежить від таких факторів, як швидкість і енергоефективність.

Як вказував Asano, і ін. [53], те, як швидко ЦП, GPU або FPGA можуть працювати, змінюється для кожного додатка. Загалом кажучи, FPGA демонструють надзвичайно високий паралелізм на більш низькій тактовій частоті у порівнянні з GPU.

GPU також може забезпечити високий паралелізм на високій тактовій частоті, але ЦП, при гіршому паралелізмі, краще керує пам’яттю. Для

системи обробки зображень, яка містить численний паралелізм, переваги ЦП немає. Це залишає порівняння між GPU і FPGA.

Наприклад, GPU краще використовувати при реалізації обчисленні нормалізованої взаємній кореляції [54] і двовимірних фільтрів [53], у той час як FPGA продуктивні в обчисленнях сполучених фільтрів [55], задачах кластеризації *k*-середніх і стереобачення [53].

Для оцінки енергоефективності Fowers і ін. [56] реалізували операцію розсувного вікна та обчислення допусків і показали, що FPGA використовував менше енергії, ніж інше устаткування в багатьох ситуаціях.

Дослідники для порівняння використовують ЦП Intel Core2 Duo E6600, 2,4 Ггц, Geforce GTX 580 GPU, 1,54 Ггц і FPGA Arria V 5AGXFB3 на техпроцесі 28 нм [57]. Споживана потужність FPGA залишається майже на рівні 1,5 Вт, оскільки ефект зміни системи незначний.

Тим часом споживана потужність ЦП і GPU залишається приблизно 150 Вт і 240 Вт. У такий спосіб енергетичне споживання ЦП більш ніж в одну тисячу раз вище, ніж FPGA. З перевагами в архітектурі енергоефективність GPU краще, ніж ЦП, але проте це незрівнянно з FPGA.

Таблиця 2.2 показує порівняння споживаної потужності й енергетичне використання FPGA, ЦП і GPU, при реалізації граничного детектору Кенні у різних роздільностях.

Таблиця 2.2 – Споживання й енергетика реалізацій детектора

Роздільна здатність	**ЦП**		**GPU**		**FPGA**	
	Споживання (Вт)	**Енергія (Дж)**	**Споживання (Вт)**	**Енергія (Дж)**	**Споживання (Вт)**	**Енергія (мДж)**
512 × 512	141	2,8	231	0,5	1,5	1,7
1024 × 1024	147	8,8	244	6,08	1,5	6,7
3936 × 3936	153	213,1	251	15,0	1,5	98,2

2.3.3 Вбудовані системи з FPGA

З удосконаленнями напівпровідникової технології стало можливо інтегрувати всі вбудовані процесори, пам’ять і інші системні одиниці в єдину програмувальну мікросхему – FPGA. Цю технологію називають «SYSTEM-ON-A-PROGRAMMABLE-CHIP» (SOPC) [58]. Як новий підхід, це забезпечило іншу архітектуру, здатну до реалізації автономної вбудованої системи бачення.

Маючи компактність і можливість реконфігурації, SOPC забезпечують високу гнучкість і продуктивність з низькими ризиками, оскільки їх дизайн може легко бути адаптований у різні типи FPGA.

Оскільки усі компоненти SOPC, включали процесори, можуть бути реалізовані як окремі не специфічні ядра, то побоювання із приводу проблем зміни архітектури є безпідставними.

Крім того, використання м’якого IP ядра процесора, наприклад, Nios II від Altera або Microblaze від Xilinx [59], дає можливість компромісу між продуктивністю системи її площею на кристалі FPGA [60]. Крім того, SOPC мають інші переваги, такі як недорогий і короткий час розробки в порівнянні з підходом ASIC [61]. Наприклад, Hau і ін. [62] розробили швидке прототипуваннння автоматизованої системи відеоспостереження. Система була розроблена з використанням IP ядра обробки відеоданих Altera і вбудованої платформи розробки для вбудованих систем (VIP, UIP) та реалізована на платі Altera DE2-70 і FPGA Altera Cyclone II. За допомогою високопродуктивних і паралельних апаратних акселераторів система продемонструвала алгоритм виявлення об’єктів у реальному часі.

Крім того, завдяки використанню м’якого процесора NIOS II, при застосуванні високорівневої мови програмування (C++) були досягнуті гнучкість додатка й зручність користувацького інтерфейсу. Wu і ін. [63] розробили систему обробки зображень у реальному часі. Система була інтегрована в недорогу програмовану мікросхему, і продуктивність системи

максимізувалася за допомогою кеша й потокової передачі у системі. Була також продемонстрована ефективна схема шинного керування. Система була реалізована на обладнанні Altera APEX 20K, SDRAM, камера Cameralink CMOS з виготовленою на замовлення інтерфейсною платою. Обидві системи були реалізовані з підходом SOPC і використовували шинний інтерфейс Avalon [64], щоб зменшити складність розробки й поліпшити продуктивність систем.

Однією із ключових проблем, які повинні були подолати обидві системи, була пропускна здатністю зовнішньої пам’яті, яка зберігала зображення. Ця пам’ять, діючи як кадровий буфер, повинна була забезпечити роботу блоку відеозахоплення, дісплею та ЦП. Крім того, ця пам’ять також використовується ЦП для програми й обладнання зберігання даних.

Інший тип архітектури є вбудованими системами, які складаються з одного або декількох мікропроцесорів/мікроконтролерів, щоб керувати цілою системою й виконати одержання зображень і їх обробку. Ці мікропроцесори могли бути мікропроцесорами загального призначення або мікропроцесорами DSP [65]. В останні роки процесор ARM став самим популярним мікропроцесором у вбудованих системах.

З доступністю процесорів ARM для систем на мікросхемах (SOC) для додатків можна створити систему в єдиному модулі. Для розробки програмного забезпечення можна використовувати Linux або Android як операційну систему, де для спрощення розробки додатків можуть бути встановлені кілька бібліотек з відкритим вихідним кодом, такі як бібліотеки OpenCV і Python. Однак більш значна складність апаратної архітектури мікропроцесорів дає відносно меншу швидкість обробки даних [66].

Guennouni і ін. розробили додаток для виявлення об’єктів на основі бібліотек OpenCV [67]. Система заснована на каскадному алгоритмі виявлення кількох об’єктів. Варфоломєєв і ін. розробили поліпшений алгоритм потоку середніх для відстеження візуального об’єкта і його реалізації на платформі ARM [68]. Алгоритм був реалізований з

використанням бібліотеки OpenCV і протестований на платі Beagleboard-xm на основі процесора ARM. Алгоритм використовує спеціальну функцію для алгоритму відстеження й граничний алгоритму виявлення. Однак функції, залучені в ці проекти, більш підходять для реалізації FPGA замість процесора ARM.

Нове покоління FPGASOC забезпечує оновлену версію SOPC і більш високу продуктивність ЦП. Russell і Fishchaber [69] розробили систему розпізнавання знака на основі OpenCV за допомогою плати Zynq SOC. Система використовує мікросхему Xilinx Zynq-7020 для одержання відео роздільною здатністю 1920 × 1080 пікселей з датчиком, приєднаним до слота інтерфейсу Flexible Physical Interface Cards (PIC) Concentrators (FPC). Система могла обробити кадр за 5 секунд.

2.4 Програмовані логічні інтегральні схеми FPGA

Інтегральні схеми FPGA (Field-Programmable Gate Array – програмована користувачем вентильна матриця або програмована логічна інтегральна схема) розроблені як реконфігуруємі напівпровідникові інтегральні схеми. Вони створюються з матрицею логічних блоків або адаптивних логічних модулів (ALM) різних типів, таких як блоки множника, пам’ять і загальна логіка, оточена й з’єднана програмувальною матрицею маршрутизації, масив оточується блоками введення-виведення, які з’єднують інтерфейсом з зовнішнім обладнанням [70]. Базова структура FPGA показана на рисунку 2.4.

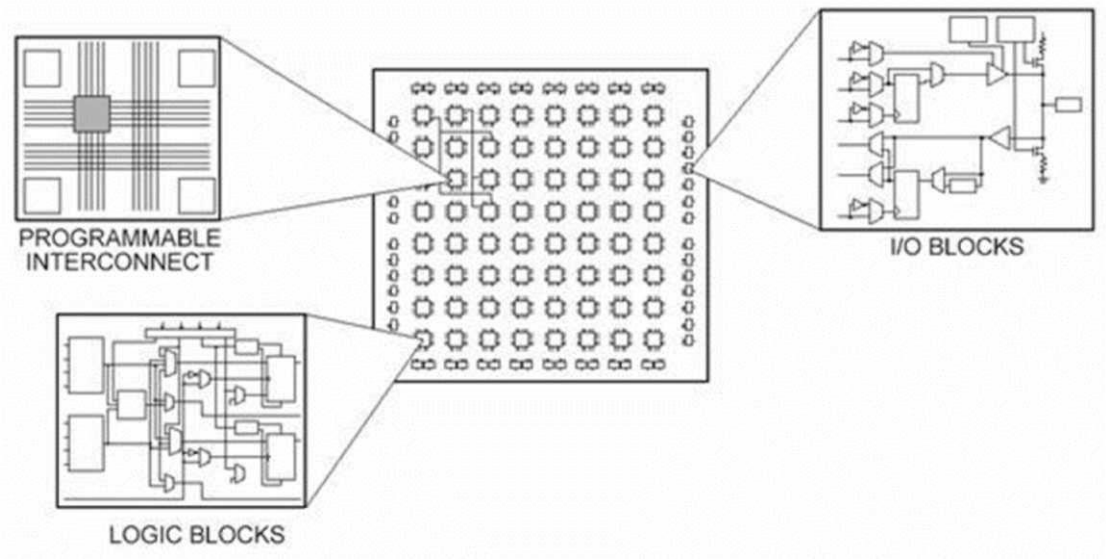


Рисунок 2.4 – Базова структура FPGA

Кожний модуль ALM, як показано на рисунку 2.5, складений із регістрів та таблиць пошуку (LUT) [71].

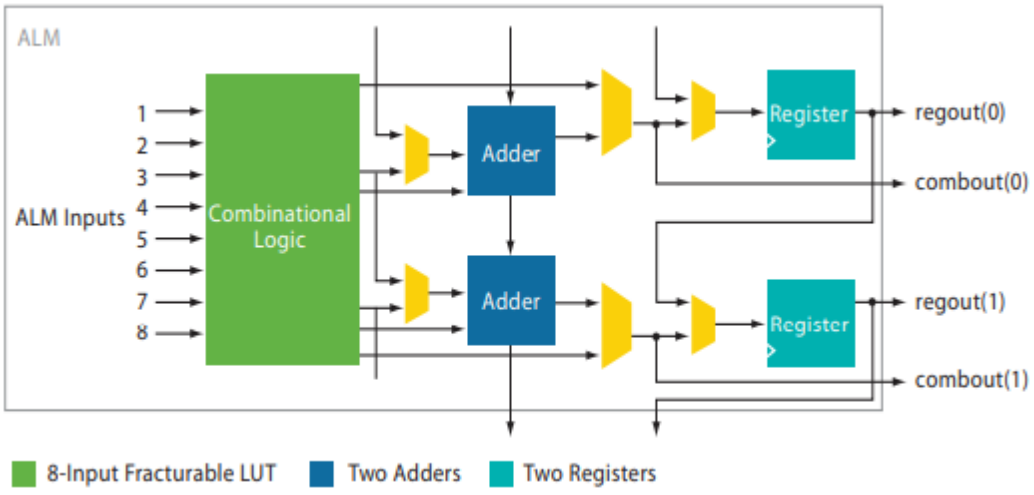


Рисунок 2.5 – Блок-схема адаптивного логічного модуля (ALM)

Коли FPGA настроєний для певних цифрових схем, кожному об’єкту CLB привласнюють просту незалежну логічну функцію. CLB використовують LUT для реалізації функції Булевої логіки, і потім це з’єднується матрицею маршрутизації для створення структури цілої цифрової схеми. Блоки введення-виводу зв’язують логічну матрицю із зовнішнім

обладнанням. Зі збільшенням кількості логічних функцій обладнання FPGA теоретично у стані бути запрограмованим у будь-який вид логічних схем.

Однак така система ніколи не буде працювати з такою швидкістю, як спеціалізована інтегральна мікросхема, при використанні тієї ж напівпровідникової технології.

Існує тенденція [58] розробки крупномодульної архітектури SOPC, синтезувавши міжз'єднання традиційних FPGA і логічних блоків разом із вбудованими мікропроцесорами і їх зв'язаним периферійним обладнанням. Цей вид архітектури використовує програмовані ядра процесора, розроблені постачальником логіки FPGA, наприклад, Microblaze (Xilinx) і Nios II (Intel/Altera).

Також представлений альтернативний підхід FPGASOC на прикладі смстем Xilinx Zynq-7000 SOC [72] і Altera Cyclone V [73], які інтегрують апаратні засоби FPGA із двоядерним твердим процесором ARM-A9. Іншим прикладом є система Smartfusion [74], яка інтегрує тверде ядро процесора Core-M3 ARM, а також аналогове периферійне обладнання, таке як багатоканальні ЦАП і АЦП, із заснованою на флеш-пам'яті матрицею FPGA.

Альтернативною системою є Microblaze [75] з IP ядром мікропроцесора, розробленим Xilinx. Цей м'який процесор реалізований повністю в пам'яті загального призначення й логічній матриці FPGA Xilinx. З точки зору архітектури системи команд Microblaze подібний заснованій на RISC архітектурі DLX [16]. Microblaze надає нову інструкцію кожний цикл, підтримуючи пропускну здатність єдиного циклу при більшості обставин [75].

З іншого боку, тверді процесори звичайно розробляються спеціалізованими компаніями. Постачальники PLD купують свої ліцензії IP, виготовляючи й вбудовуючи процесори в їхню певну родину FPGA, такі як Cyclone Altera V і Xilinx Zynq-7000, які включають двоядерний ARM на 800 Мгц і Cortex-A9 на 1,0 Ггц [76] відповідно. Блок-схема такого обладнання показана на рисунку 2.6.

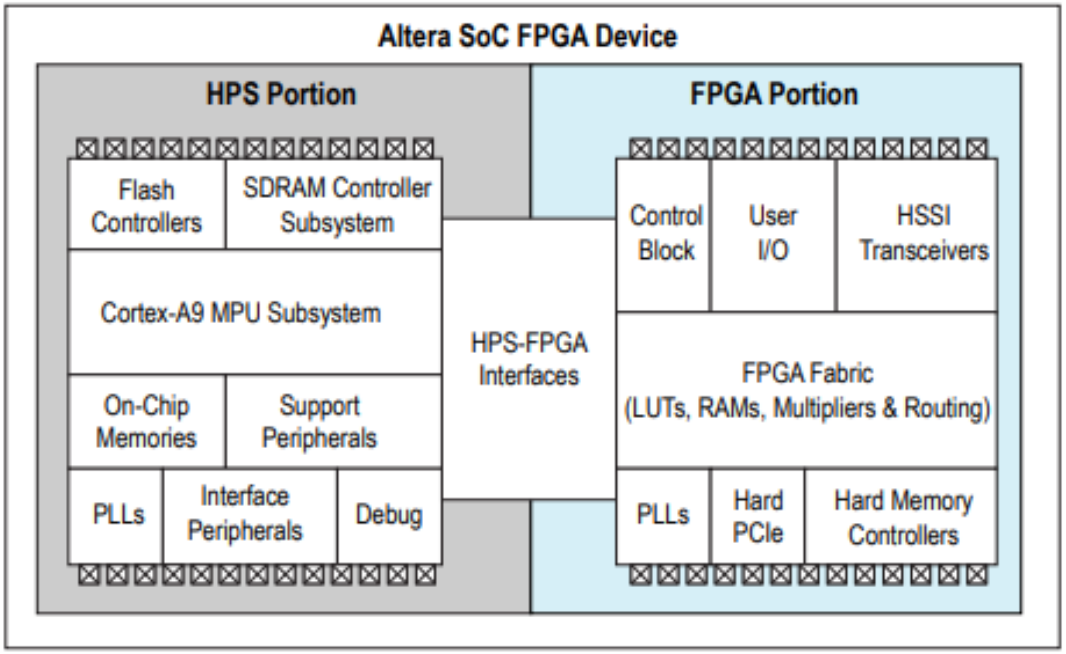


Рисунок 2.6 – Блок-схема обладнання FPGA Altera/Intel SOC

2.5 Апаратне оточення FPGA

Хоча FPGA розглядають як базову частину в цій системі реального часу, інше обладнання також важливе для реалізації обробки зображень. Ціла система (апаратно-програмний комплекс) включає 3 підсистеми: відеосистема захоплення, система обробки відеоданих і відеодисплей / система керування. Це звичайно включає три типи обладнання: обладнання введення відеосигналу, макетна плата (DE10-Nano) і обладнання відеодисплея.

Існує два набори систем, об’єднаних на платі, FPGA і апаратна система процесора (HPS). Кожна із цих систем підключена до різного набору обладнання.

Ядра обох систем (Cyclone V FPGA і процесор ARM Cortex-A9) інтегруються в одну однокристальну схему і спільно використовують сполучні шини між FPGA і процесором ARM.

Рисунок 2.7 поясняє архітектуру обладнання FPGA DE10-Nano [78].

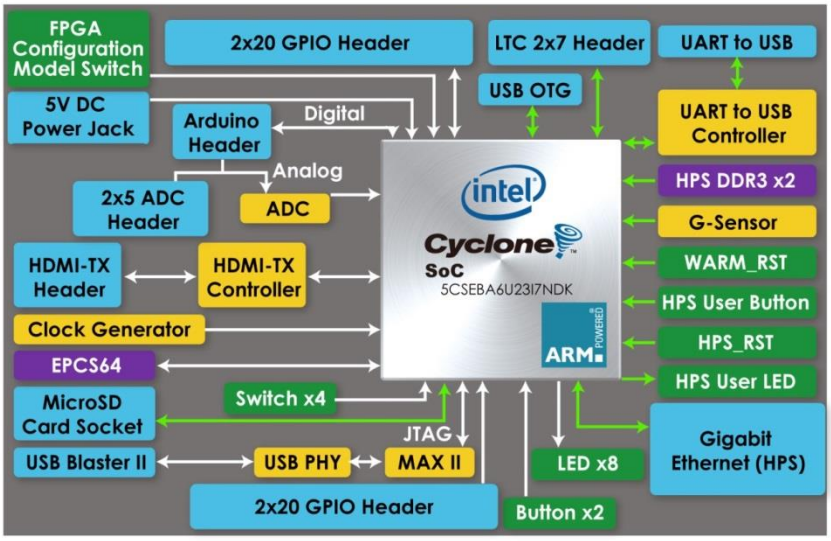


Рисунок 2.7 – Архітектура обладнання DE10-Nano

Ядро процесору ARM Cortex-A9 взаємодіє з мікросхемою DDR3 1 ГБ із 32–розрядною шириною шини, що працює на частоті 400 Мгц. Теоретично можна надати пропускної здатності пам’яті до 400 МГц × 32 біт = 1,6 ГБ/с, що досить для відеопотоку 1024 × 768@30 кадрів/с, який потребує приблизно 1024 × 768 × 24 Б × 30 кадрів/с ≈ 67,7 МБ/с.

Рисунок 2.8 показує схему розміщення зовнішніх інтерфейсів та основних компонентів на платі FPGA DE10-Nano.

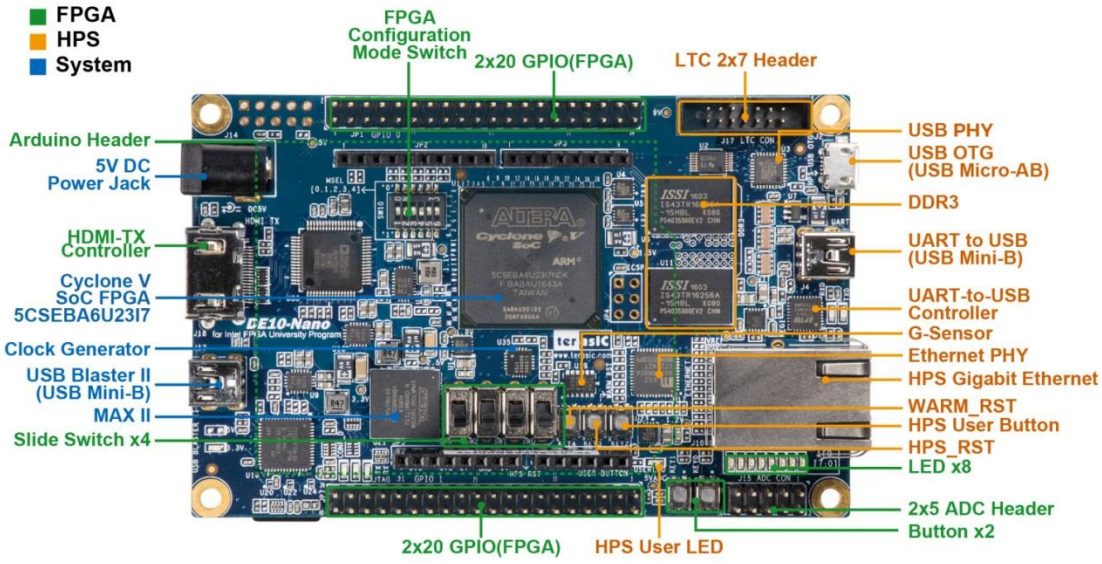


Рисунок 2.8 – Компоненти та інтерфейси плати FPGA DE10-Nano

2.6 Програмування FPGA

2.6.1 Мови опису апаратних засобів

Мови опису апаратних засобів (HDL) є мовами, використовуваними для опису апаратної поведінки електронної системи, структури й потоку даних.

Недороге обладнання FPGASOC використовується для оперативної обробки зображень шляхом розробки системи обробки зображень у реальному часі з декількома підходами для алгоритмів попередньої обробки, за допомогою FPGA, для скорочення часу обробки. Крім того, це синхронізує вихідні дані паралельно з попередньо обробленими даними в пам’яті для наступної обробки, тобто попередньо оброблене зображення зберігається як 64-розрядне слово з 8 бітами кожний для значень RGB і 32-розрядне для результатів попередньої обробки. Одночасно, це забезпечує інфраструктуру для реалізації складних додатків для обробки зображень в інтегрованій системі ARM з підтримкою бібліотеки OpenCV.

Дизайн FPGA може бути розроблений на платформі розробки Quartus II з використанням IP ядра обробки відеозображення (VIP), що забезпечує кілька підсистем, таких як кадровий буфер, синхронізоване відео тощо.

Розроблена Intel (Altera) платформа Qsys є інструментом для розробки систем SOPC. Тому програмувальний апаратний дизайн, необхідний до розробки алгоритмів, щоб бути сумісним із заснованим на Intel VIP IP, використовується так, щоб це могло бути сумісним з підсистемами Intel VIP.

Можливо застосувати два підходи, один для високорівневого синтезу (HLS) мов опису апаратних засобів (HDL), інший для розробки IP алгоритму для FPGA. Ці два методи використовуються для реалізації граничного алгоритму виявлення Кенні з підходом HDL. Обидва методи показують кореляцію використання ресурсів і кореляцію затримок.

Підхід HLS використовується для розробки реалізації граничного алгоритму детектора Кенні й кутового алгоритму виявлення Харріса. Із цим

підходом це поліпшує продуктивність і зменшує труднощі програмування FPGA.

Деталі обробки вбудованого Linux є в основі дизайну FPGASOC, де використовуються зв’язані бібліотеки. IP розробляється за допомогою методу HDL, який містить вихідні дані RGB, кутовий результат виявлення Харріса, граничний результат виявлення Кенні й півтоновий результат, які зводяться попіксельно, як 64-розрядне слово для кожного пікселя. Одночасно, цей дизайн заснований на спеціалізованому додатку OpenCV для наступної обробки реалізацій.

Поведінковий HDL, реалізований за допомогою основних мов або HDL Verilog або VHDL або комбінації обох, використовується як у середніх, так і великих проектах.

2.6.2 Високорівневий синтез на OpenCL

OpenCL є платформою розробки програми для паралельного програмування на гетерогенних платформах. Це середовище було спочатку розроблене для GPU, але тепер вона розвинулася до платформу, яка може працювати з GPU, центральними процесорами, FPGA і іншими. В платформі використовуються мови (на основі C і C++) для запису ядер (робота функцій обладнань OpenCL) і ряд інтерфейсів прикладного програмування (API) для управління платформою. OpenCL забезпечує стандартні паралельні обчислення на основі сегментації даних і задач. При реалізації цього підходу ядра OpenCL виконуються в структурі FPGA. Цей підхід вимагає явного визначення паралелізмів замість автоматичного розпаралелювання в HLS.

Іноді є схильність зменшувати великий паралелізм на FPGA до рівня GPU, ця операція спрощує доступ програміста GPU до програмування FPGA, але викликає втрату продуктивності.

Було проведене порівняльне випробування [77] між OpenCL і іншими мовами HLS, включаючи системи Bluespec Verilog, Legup тощо щодо роботи

кожної архітектури на FPGA. Воно показало, що OpenCL і інші вдосконалені платформи, які застосовували GPU-подібний підхід до програмування, мали погану й нестабільну продуктивність, у той час як підходи з архітектурою низького рівня були реалізовані швидко й ефективно на FPGA.

2.6.3 Високорівневий синтез в системі FPGASOC

У порівнянні із проектами тільки на FPGA, проекти FPGASOC включають систему HPS. HPS, яка використовує процесор ARM, може виконувати користувацькі додатки, включаючи керування поведінкою IP ядер на стороні FPGA, обробку даних або доступ до обладнання на HPS.

Вбудована системна інтеграція Linux включає кілька частин, які разом з апаратний дизайном FPGA становлять архітектуру FPGASOC, приклад реалізації якої показано на рисунку 2.9.

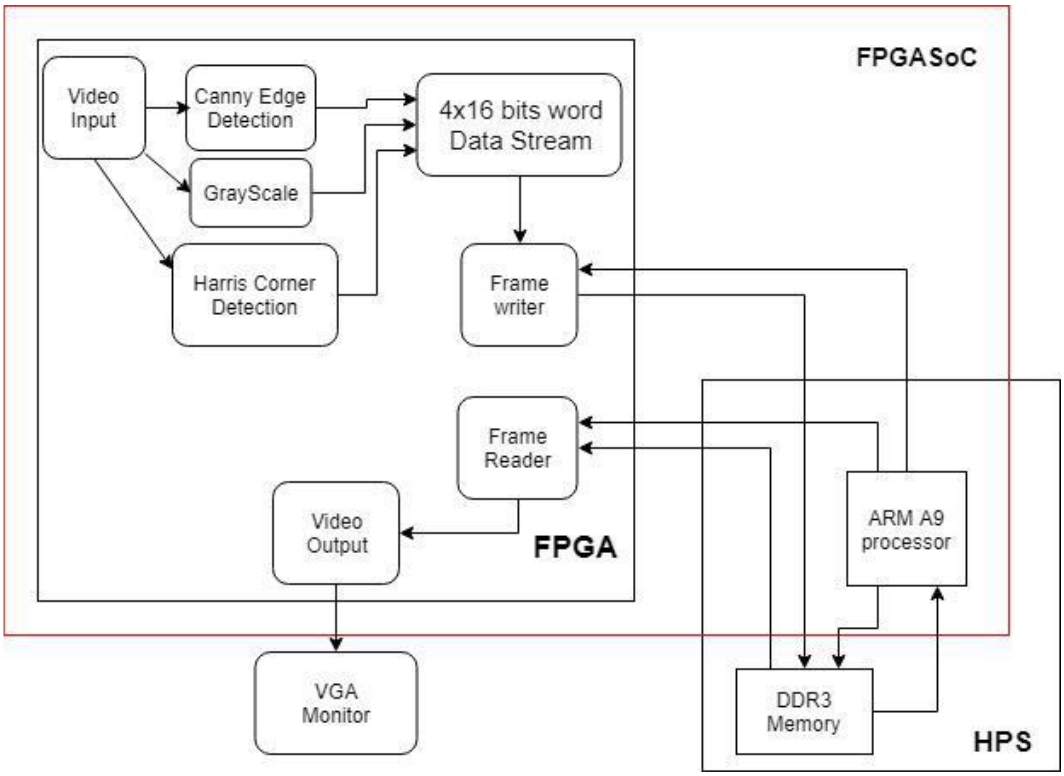


Рисунок 2.9 – Система на архітектурі FPGASOC

HPS може працювати у двох режимах: низькорівневий режим або режим операційної системи.

Низькорівневий режим може працювати з інструментом ARM DS-5, який забезпечує здатність до налагодження коду. Однак, якщо додаток включає API обладнання або інші сторонні бібліотеки, то потрібно працювати на настроєній операційній системі, наприклад, Linux.

Рисунок 2.10 показує етапи процесу розробки в FPGASOC.

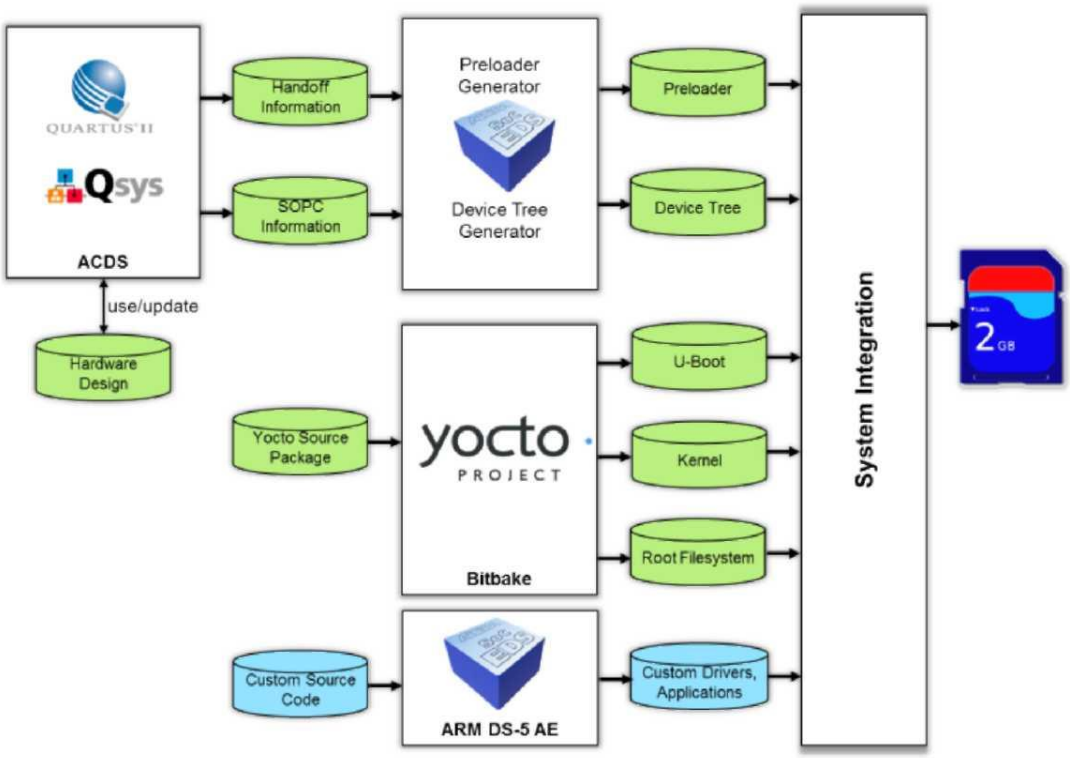


Рисунок 2.10 – Представлення процесу розробки в FPGASOC

Попередній завантажник (U-boot) і дерево обладнань створюються за допомогою платформи Quartus II, ядра Linux й кореневої файлової системи. Ядро Linux може бути створеним за допомогою проекту [79] Yocto та утиліти Openbox.

Після цього використовується платформа розробки, наприклад, ARM DS-5, щоб завантажити й налагодити користувацькі додатки. Настроєна система може бути записана на SD-карту.

Попередній завантажник конфігурує компонент HPS на базі інформації, згенерованої Quartus, і завантажує наступний етап процесу початкового завантаження в SDRAM та передає керування до нього.

Дерево обладнань є структурою даних, яка описує використовуване устаткування до операційної системи - насамперед, Linux. Шляхом передачі цієї структури даних ядру ОС можна підтримувати багато змін апаратних засобів. Ця гнучкість особливо важлива, коли апаратні засоби включають FPGA.

Завантаження ядра Linux починається з виконання архітектурно-залежних послідовностей ініціалізації низького рівня.

Після цього починається ініціалізація усіх підсистем ядра й драйверів, які були скомпільовані в ядро. Нарешті, монтується коренева файлова система, яка містить оболонку користувацьких програм.

Використання Linux на системі HPS виправдано, якщо високорівневий додаток для обробки зображень може використовувати існуючі бібліотеки обробки зображень [67, 69]. OpenCV є однією з найбільш поширених таких бібліотек.

OpenCV є міжплатформовою бібліотекою функцій програмування, що реалізує багато алгоритмів машинного зору, від основної фільтрації до вдосконаленого виявлення об’єктів.

OpenCV відіграє головну роль стандартизації API для машинного зору в реальному часі, оскільки забезпечує інтерфейсну прив’язку для різних мов програмування, таких як Python, Ruby, MATLAB та Java і може використовуватися в різних операційних системах, включаючи Linux, Windows, Android і Mac OS [80].

Завдяки багатьом алгоритмам і функціям, реалізованим в OpenCV, і відкритому коду, OpenCV широко використовується для реалізації систем машинного зору. Ця бібліотека прискорює швидкість розробки систем.

2.7 Характеристика вхідних даних

Вхідні дані задані у вигляді відеопотоку, що є результатом захоплення зображення реальної дорожньої смуги відеокамерою.

Відеопотік роздільної здатності 640 × 480 пікселів 30 кадрів/с, тривалість не обмежена.

У якості тестового відеопотоку може використовуватись відеозапис реальної дороги, створений відеорегістратором транспортного засобу.

Обмежень на національні особливості, стан дороги та погодні умови не накладається.

3 РЕАЛІЗАЦІЯ МЕТОДУ ДЕТЕКТУВАННЯ ДОРОЖНЬОЇ СМУГИ

3.1 Обґрунтування вибору апаратного середовища

Для реалізації методу детектування дорожньої смуги була вибрана плата DE10-Nano фірми Terasic, яка була розглянута в розділі 2.

Ця плата конструктивно складається з двох компонентів: FPGA і HPS.

FPGA складається з мікросхеми Altera Cyclone V SE 5CSEBA6U23I7NDK, мікросхеми конфігуратора EPCS64, інтерфейсу USB-Blaster II для програмування в режимі JTAG, 2 кнопок і 4 перемикачів для зміни режимів роботи, 8 світлодіодних індикаторів, 3 джерела тактових сигналів 50 МГц, 2 40-контактних порта розширення, 1 порта для плат Arduino, 1 10-контактного аналогового порта, мікросхеми АЦП, інтерфейсу SPI і виходу HDMI.

Плата DE10-Nano (рис. 3.1) при низькій вартості забезпечує достатню для реалізації методу продуктивність.

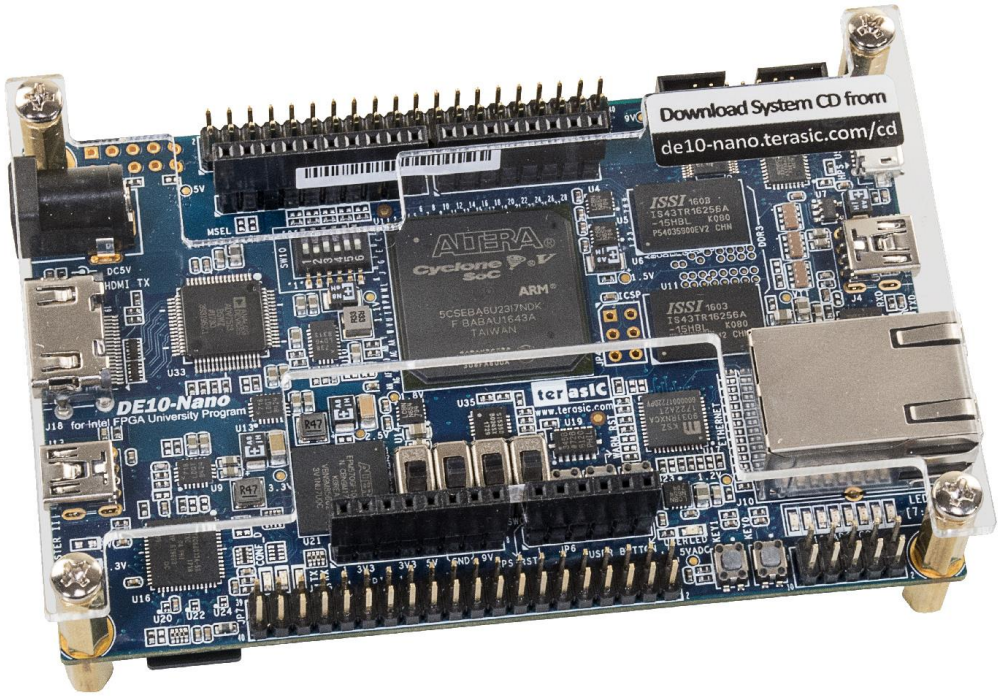


Рисунок 3.1 – Плата FPGA DE10-Nano

На стороні HPS плати DE10-Nano знаходяться процесор 800MHz Dual-core ARM Cortex-A9, пам’ять 1GB DDR3 SDRAM, інтерфейс 1 Gigabit Ethernet PHY, інтерфейс USB OTG, інтерфейс UART to USB, акселерометр, слот під карту пам’яті Micro SD, 3 кнопки для перезапуску і зміни режимів роботи, 1 світлодіодний індикатор і 1 14-контактний порт розширення LTC.

Для захоплення відеопотоку використана цифрова камера PlayStation Eye (рис. 3.2), яка побудована на базі контролеру OV534 USB компанії OmniVision Technologies.

Рисунок 3.2 – Цифрова камера PlayStation Eye

PlayStation Eye здатний знімати стандартне відео із частотою кадрів 60 Гц при роздільній здатності 640 × 480 пікселів та 120 Гц при роздільній здатності 320 × 240 пікселів. Завдяки використанню сенсорних мікросхем OmniVision Technologies забезпечується ефективна робота при слабкому освітленні. Камера оснащена двостворчатим регульованим об'єктивом із фіксованим фокусуванням.

Для виведення обробленого зображення використовується HDMI-монітор через відповідний інтерфейс плати DE10-Nano.

3.2 Обґрунтування вибору алгоритму і середовища програмної реалізації

Як популярний алгоритм, що використовується на етапах попередньої обробки додатків машинного зору, граничний алгоритм виявлення є ідеальним розв'язком для реалізації на платформі FPGA. Із кількох видів граничних методів виявлення був обраний детектор Собеля, оскільки він включає методи згортки, які є математичним способом об’єднати дві функції для формування третьої функції.

Існує кілька видів граничних методів виявлення, які могли бути обрані, включаючи детектор Собеля, детектор Лапласа з Гаусовим фільтром, детектор Кенні, детектор Кірша, детектор Роберта і детектор Прюітта. Вони представляють альтернативні підходи до детектування зображення на основі складності граничного обчислення й здатності до граничного виявлення в умовах, коли зображення страждає від забруднення шумовими перешкодами.

Для конфігурування FPGA на платі DE10-Nano був обраний високорівневий синтез на OpenCL. OpenCL забезпечує стандартні паралельні обчислення на основі сегментації даних і хоча є більш ресурснозатратним, ніж інші методи, забезпечує більшу швидкість розробки.

OpenCL, як високорівневий підхід до програмування, дозволяє використовувати стандартні бібліотеки мов C та C++. На відміну від прямого програмування FPGASOC на HDL мовах, OpenCL не потребує застосування низькорівневих драйверів системи.

Обрана архітектура системи детектування дорожньої смуги показана на рисунку 3.3. Головною відміною від звичайного підходу до розробки на OpenCL є те, що в архітектуру було додано ядро VIP від Altera для організації виводу зображення на HDMI дисплей.

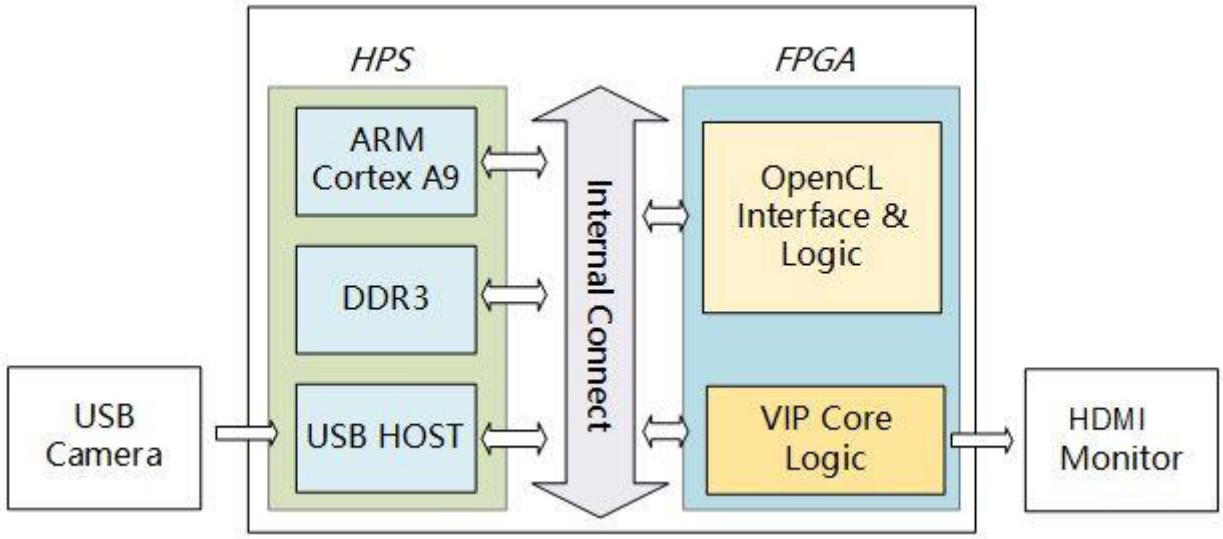


Рисунок 3.3 – Архітектура системи детектування дорожньої смуги

3.3 Програмна реалізація

Програмний засіб складається з двох головних модулів. Перший модуль, який виконується за допомогою HPS процесору, відповідальний за зчитування відеопотоку з зовнішньої USB камери та виводу зображення на HDMI дісплей. Модуль на ядрі OpenCL відповідальний за безпосередню обробку зображення на FPGA.

Програмний засіб реалізований у вигляді застосунку Linux, який збережений під назвою **lane_detection.run**.

Застосунок не приймає вхідних параметрів.

При запуску застосунку у середовищі Linux здійснюється ініціалізація відеокамери, вхідного та вихідного кадрового буферу, конфігурація FPGA та визначення ядра **camera_sobel**, що написаний як застосунок OpenCL.

Після запуску зображення з відеокамери подається на вхідний кадровий буфер, з якого ядро OpenCL зчитує відеодані для обробки фільтром Гауса та подальшою обробкою детектором Собеля і визначення границь дорожньої смуги.

Для детектора Собеля використане порогове значення яскравості граничної обробки 55 при середній кадровій пропускній здатності детектора 400 кадрів/с.

Контроль роботи фільтра спостерігається в відлагоджувальній консолі (рис. 3.4).

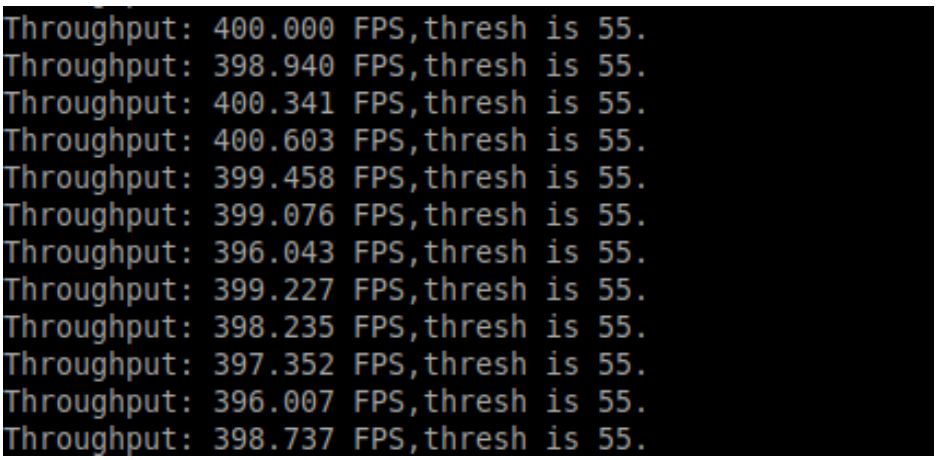


Рисунок 3.4 – Контроль роботи детектора Собеля в консолі

Розроблена система детектування дорожньої смуги використовує апаратні ресурси FPGA плати DE10-Nano як показано в таблиці 3.1.

Таблиця 3.1 – Підсумок використання ресурсів FPGA плати DE10-Nano

Назва ресурсу	**Використання**
Блоки логіки, з них:	19%
– Адаптивні таблиці відповідності (ALUT)	11%
– Виділені логічні регістри	8%
Блоки пам’яті	13%
Блоки обробки цифрових сигналів (DSP)	0%

3.4 Інструкція користувача

Для виконання детектування дорожньої смуги необхідно підготовити карту формату Micro SD з кореневою файловою системою та завантажником і необхідним файлом конфігурації FPGA. Після включення плати та завантаженням операційної системи необхідно запустити розроблений застосунок **lane_detection.run** який знаходиться у домашньому каталозі користувача **root**.

Результати обробки файлу візуалізуються у вигляді виводу зображення з відеопотоку з виділенням границь дорожньої смуги.

3.5 Тестування розробленого програмного засобу

Апаратно-програмний комплекс тестувався на тестовому стенді з відеопотоком роздільної здатності 640 × 480 пікселів 30 кадрів/с.

На рисунку 3.5 відображений кадр вхідного тестового відеопотоку з дорожньою смугою.

Рисунок 3.5 – Кадр тестового відеопотоку з дорожньою смугою

Проміжний етап роботи апаратно-програмного комплексу, який полягає в створенні півтонового зображення, яке потім обробляється фільтром Гауса, показаний на рисунку 3.6.

Рисунок 3.6 – Півтонове зображення після обробки фільтром Гауса

Результати детектування дорожньої смуги відображений на рисунку 3.7.

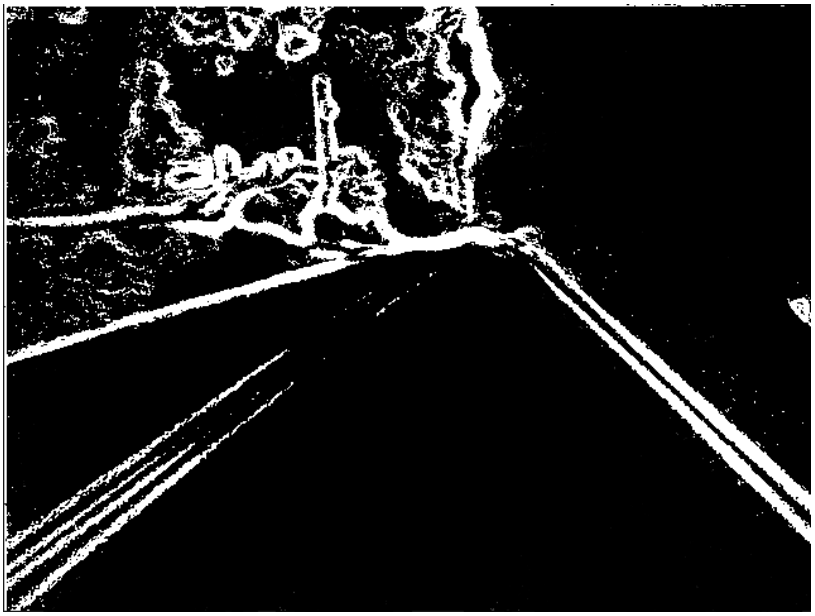


Рисунок 3.7 – Виділення границь дорожньої смуги детектором Собеля

ВИСНОВКИ

Виявлення дорожньої смуги і її відстеження є однією з основних характеристик просунутої системи допомоги водієві.

Заснований на відео підхід є дуже зручною модальністю для виявлення дорожніх смуг. При тому, що в останні роки досягнуто великий прогрес в області виявлення й відстеження дорожньої смуги, усе ще існує можливість для поліпшення методів виявлення й відстеження дорожньої смуги з урахуванням широкого спектру мінливості в середовищах дорожньої смуги.

У рамках атестаційної роботи був розроблений і реалізований апаратно-програмний засіб реалізації методу детектування дорожньої смуги з використанням FPGA. Розглянуто ряд алгоритмів виявлення й відстеження дорожньої смуги, розроблених протягом останніх десятиліть.

В роботі використані проведені раніше з іншими авторами дослідження [4] в напрямку обробки зображень.

Дослідження, описане в атестаційній роботі, показало використання недорогого обладнання FPGA для розробки системи обробки зображень у реальному часі з декількома методами для реалізації алгоритмів попередньої обробки.

Робота демонструє використання спільної обробки бібліотеки OpenCL з FPGA. Протестована система використовувала недорогу плату FPGASOC, DE10-NANO, яка вироблена Terasic Inc. Розроблений додаток, заснований на спеціалізованій бібліотеці OpenCL, що працює на процесорах ARM, і одночасно одержує результат попередньої обробки зображення, виконаний FPGA.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A. Rosenfeld, «Computer vision: basic principles», Proceedings of the IEEE, pp. 863-868, 1988.

2. Вискребенцева С.О., Кобилін О.А. (2019) Методи сегментації зображень. Матеріали XXIII міжнародного молодіжного форуму. Радіоелектроніка та молодь у XXI столітті, 19-20.

3. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2017). Video shot boundary detection via sequential clustering. International Journal Information Theories and Applications, 24(1), pp. 50-59.

4. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – pp. 4701-4706.

5. Mashtalir, S. V., Stolbovyi, M. I., & Yakovlev, S. V. (2019). Clustering Video Sequences by the Method of Harmonic k-Means. Cybernetics and Systems Analysis, 55(2), pp. 200-206.

6. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

7. A.Bar Hillel, R. Lerner, D. Levi, and G. Raz, «Recent progress in road and lane detection: a survey», Machine Vision and Applications , Feb. 2012.

8. J.McCall and M. Trivedi, «Video-based lane estimation and tracking for driver assistance: Survey, System, and Evaluation», IEEE Trans. On Intelligent Transportation Systems , vol. 7, pp. 20–37, Mar. 2006.

9. M.Aly, «Real time detection of lane markers in urban streets», in Proc.IEEE Intell. Veh. Symp., Jun. 4–6, 2008, pp. 7-12.

10. C.W.Lin, H.Y.Wang, D. C. Tseng, «A robust lane detection and verification method for intelligent vehicles», In Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on vol. 1, pp. 521-524.

11. Y.U. Yim and S.- Y. Oh, «Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving», IEEE Trans. Intell. Transp. Syst. , vol. 4, no. 4, pp. 219-225, Dec. 2003.

12. Zhao, H., Teng, Z., Kim, H.H., and Kang, D. J. , «Annealed Particle Filter Algorithm Used for Lane Detection and Tracking» Journal of Automation and Control Engineering 1.1 , 2013.

13. S.Zhou, Y.Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, «A novel lane detection based on geometrical model and gabor filter», in Intelligent Vehicles Symposium (IV), 2010 IEEE , June 2010, pp. 59-64.

14. S. Zhang, «Real Time Image Processing on FPGAs», University of Liverpool, Department of Electrical Engineering and Electronics, January 2018, pp. 60-75.

15. J. Canny, «A computation approach to edge detection», IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. vol. 8, no. no 6, pp. pp. 769-798, November 1986.

16. D. A. Patterson and J. L. Hennessy, Computer Organization & Design, the hardware/software interface, Morgan Kaufmann Publishers, 1998.

17. International Telecommunication Union , «BT.656 : Interface for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601», International Telecommunication Union , 12 2007. [Online]. Available: http://www.itu.int/rec/R-REC-BT.656/en.

18. C. T. Johnston, K. T. Gribbon and D. G. Bailey, «Implementing image processing algorithms on FPGAs», in In Proceedings of the Eleventh Electronics New Zealana Conference, New Zealand, November 2004, ENZCon’04, pp. 118-123.

19. Intel Croporation, Intel Croporation, [Online]. Available: http://www.altera.com.

20. J.Fritsch, T.Kuehnl, and A. Geiger, «A new performance measure and evaluation benchmark for road detection algorithms», in Intelligent Transportation

Systems-(ITSC), 2013 16th International IEEE Conference on, pp. 1693-1700. IEEE, 2013.

21. T.Veit, J.-P.Tarel, P. Nicolle, and P. Charbonnier, «Evaluation of road marking feature extraction», in Proc. 11th Int. IEEE Conf. ITSC, Oct. 2008, pp. 174–181.

22. A.Borkar, M. Hayes, and M.T.Smith, «Robust lane detection and tracking with ransac and Kalman Filter», Proceedings of the IEEE International Conference on Image Processing, pp. 3261-3264, 2009.

23. G.Liu, F.Wörgötter, and I.Markeli´c, «Combining statistical hough transform and particle filter for robust lane detection and tracking», in Proc. IEEE IV, Jun. 2010, pp. 993 – 997.

24. G.Liu, F.Wörgötter and I.Markeli´c, «Lane shape estimation using a´partitioned particle filter for autonomous driving, in Proc. IEEE ICRA, 2011, pp. 1627-1633.

25. C.Guo, S.Mita, and D.McAllester, «Lane detection and tracking in challenging environments based on a weighted graph and integrated cues», in Proc. Int. Conf. on IEEE/RSJ Intelligent Robots and Systems,Taipei, Taiwan, Oct. 2010, pp. 6643-6650.

26. A.Borkar, M.Hayes, and M.T.Smith, «Polar randomized hough transform for lane detection using loose constraints of parallel lines» Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011.

27. T.T. Tran, H.M.Cho, S.B.Cho, «A robust method for detecting lane boundary in challenging scenes» Information Technology Journal. Image Process., vol. 10, no.12, pp. 2300-2307, 2011.

28. Y.-C. Leng and C.-L. Chen, «Vision-based lane departure detection system in urban traffic scenes», in Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on , 2010, pp. 1875- 1880.

29. R.Gopalan, T.Hong, M.Shneier, and R. Chellappa, «A Learning Approach Towards Detection and Tracking of Lane Markings», Intelligent

Transportation Systems, IEEE Transactions on, vol. 13, no. 3, pp. 1088- 1098, 2012.

30. G.Liu, F.Wörgötter, and I.Markeli´c, «Stochastic Lane Shape Estimation Using Local Image Descriptors», IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 1, pp. 13-21, Mar. 2013.

31. H.Yoo, U.Yang, and K. Sohn, «Gradient-enhancing conversion for illumination-robust lane detection», IEEE Transactions on Intelligent Transportation Systems , vol. 14, pp. 1083 -1094, September 2013.

32. P.Daigavane and P. Bajaj, «Road Lane Detection with Improved Canny Edges Using Ant Colony Optimization», in 3rd International Conference on Emerging Trends in Engineering and Technology International Journal of Computer Science & Information Technology (IJCSIT) Vol 7, No 4, August 2015, (ICETET) , pp. 76-80, 2010.

33. Q.Lin, Y.Han, and H. Hahn, «Real-Time Lane Detection Based on Extended Edge-Linking Algorithm», Proceedings of 2nd International Conference on Computer Research and Development, Seoul, South Korea, pp.725-730, Jun. 2010.

34. U.Ozgunalp, and N.Dahnoun, «Robust lane detection and tracking based on novel feature extraction and lane categorization», In Acoustics, Speech and Signal Processing (ICASSP) International Conference on, pp. 8129-8133. IEEE, May.2014.

35. Y.Li, A.Iqbal, and N.R.Gans, «Multiple lane boundary detection using a combination of low-level image features». In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, pp. 1682-1687. IEEE, 2014.

36. V.S.Bottazzi, P.V.Borges, B.Stantic, «Adaptive regions of interest based on HSV histograms for lane marks detection». In Robot Intelligence Technology and Applications 2, pp. 677-687. Springer International Publishing, 2014.

37. J.Wang, T.Mei, B. Kong, and H.Wei, «An approach of lane detection based on Inverse Perspective Mapping», In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on pp. 35-38. IEEE.

38. H.Tan, Y. Zhou, Y.Zhu, D.Yao, and K. Li, «A novel curve lane detection based on Improved River Flow and RANSAC», In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, pp.133-138. IEEE, 2014.

39. Z.Kim, «Robust lane detection and tracking in challenging scenarios», IEEE Trans. Intell. Transp. Syst. , vol. 9, no. 1, pp. 16-26, Mar. 2008.

40. H.Jung, J. Min, and J.Kim, «An efficient lane detection algorithm for lane departure detection», In Intelligent Vehicles Symposium (IV), 2013 IEEE, pp. 976-981.

41. M.Tan, B. Paula, and C.R.Jung, «Real-time detection and classification of road lane markings», In Graphics, Patterns and Images (SIBGRAPI), 2013 IEEE 26th SIBGRAPI-Conference on, pp. 83-90. IEEE, 2013.

42. S.C.Tsai, B.Y.Huang, Y.H.Lin, C. W. Lin, C. S. Tseng, and J. H. Wang, «Novel boundary determination algorithm for lane detection», In Connected Vehicles and Expo (ICCVE), 2013 IEEE International Conference on pp. 598-603. IEEE, 2013.

43. A.Borkar, M. Hayes, M. Smith, and S. Pankanti, «A layered approach to robust lane detection at night», in Proc. IEEE Workshop Comput. Intell. Vehicles Veh. Syst., 2009, pp. 51-57.

44. S.Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake, «Robust lane detection in urban environments», in Proc. IEEE Intell. Robots Syst., Oct. 2007, pp. 123-128.

45. A.Assidiq, O.Khalifa, R. Islam, and S. Khan, «Real time lane detection for autonomous vehicles», in Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on. IEEE,2008, pp. 82-88.

46. H.Y. Cheng, C.C. Yu, , C. C. Tseng, K. C. Fan,J. N. Hwang, and B. S. Jeng, «Hierarchical lane detection for different types of roads», In Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on pp. 1349-1352. IEEE, 2008.

47. A.M. Kumar, P. Simon, «Review of Lane Detection and Tracking Algorithms in Advanced Driver Assistance System», In International Journal of Computer Science & Information Technology (IJCSIT) Vol 7, No 4, August 2015

48. J. Kang and R. Doraiswami, «Real-time image processing system for endoscopic applications», in Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference, May 2003.

49. C. Balfour, J. S. Smith and S. Amin-Nejad, «Feature correlation for weld imageprocessing applications», International Journal of Production Research, pp. Volume 42, pp 975-995, March 2004.

50. S. A. Clukey, «Architecture for Real-Time, Low-SWaP Embedded Vision Using FPGAs», Master's Thesis, University of Tennessee, 2016.

51. S. Palnitkar, Verilog HDL: a guide to digital design and synthesis, Prentice Hall Professional, 2003.

52. P. J. Ashenden, The designer's guide to VHDL. Vol. 3. Morgan Kaufmann, Morgan Kaufmann, 2010.

53. S. Asano, T. Maruyama and Y. Yamaguchi, «Performance comparison of FPGA, GPU and CPU in image processing», in 2009 International Conference on Field, 2009.

54. E. Fykse, «Performance Comparison of GPU , DSP and FPGA implementations of image processing and computer vision algorithms in embedded systems», Norwegian University of Science and Technology, 2013.

55. Z. K. Baker, M. B. Gokhale and J. L. Tripp, «Matched Filter Computation on FPGA, Cell and GPU», 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007), pp. 207-218, 2007.

56. J. Fowers, G. Brown, P. Cooke and G. Stitt, «A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications», in Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, 2012.

57. P. R. Possa, S. A. Mahmoudi, N. Harb, C. Valderrama and P. Manneback, «A Multi-Resolution FPGA-Based Architecture for A Multi-

Resolution FPGA-Based Architecture for», IEEE TRANSACTIONS ON COMPUTERS VOL. 63, no. 10, pp. pp.2376-2388, 2014.

58. Intel Croporation, «SOPC Builder User Guide» December 2010.

59. Xilinx, Inc., [Online]. Available: www.xilinx.com.

60. J. Rose, «Hard vs. soft: the central question of pre-fabricated silicon», in In Multiple-Valued Logic, 2004. Proceedings. 34th International Symposium on, 19-22 May 2004.

61. P. Mead, «Systems on Programmable Chips -Will SOPC Eclipse SoC» in Designing Systems on Silicon /EE Cambridge Seminar, December 2001.

62. H. T. Ngo, R. W. Ives, R. N. Rakvic and R. P. Broussard, «Real-time video surveillance on an embedded, programmable platform», Microprocessors and Microsystems, Vols. 37(6-7), pp. pp.562-571, 2013.

63. F. Wu, J. S. Smith, A. J. Tickle and Q. Huang, «System of a programmable-chip-based image-processing system», Journal of Electronic Imaging, 22(2), 023026, 2013.

64. I ntel Corporation, «Avalon Interface Specifications», 2017.

65. M. B. Sandler, L. Hayat, L. Costa and A. Naqvi, «A comparative evaluation of DSPs,microprocessors and the transputer for image processing», in Acoustics, Speechi, and Signal Processing 1989 International Conference, 23-26 May 1989.

66. E. Jamro and K. Wiatr, «Implementation of convolution operation on general purpose processors», in Euromicro Conference, Proceedings 27thi, 2001.

67. S. Guennouni, A. Ahaitouf and A. Mansouri, «Multiple object detection using OpenCV on an embedded platform», in Information Science and Technology (CIST), 2014.

68. A. Varfolomieiev and O. Lysenko, «An improved algorithm of median flow for visual object tracking and its implementation on ARM platform», Journal of Real-Time Image Processing, vol. 3, no. 11, pp. pp.527-534., 2016.

69. R. Matthew and S. Fischaber, «OpenCV based road sign recognition on Zynq», in Industrial Informatics (INDIN), 2013 11th IEEE International Conference, July, 2013.

70. S. Trimberger, «Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology», Proceedings of the IEEE, 103(3), pp. pp.318-331., 2015.

71. I ntel Corporation, «FPGA Architecture», Intel Corporation, 2006.

72. XILINX INC., «Zynq-7000 All Programmable SoC Data Sheet», 2017.

73. I ntel Croporation, «Cyclone V Device Overview», 2016.

74. Microsemi Corporation, «SmartFusion Customizable System-on-Chip», 2015.

75. XILINX INC., «MicroBlaze Processor Reference Guide», 2009.

76. ARM Limited, «Cortex-A9 Technical Reference Manual», ARM Limited, 2010.

77. Intel Croporation, «Cyclone V Hard Processor System Technical Reference Manual», Intel Croporation, 2018.

78. Terasic Technology Inc., «DE10-NANO User Manual», Terasic Technology Inc., 2016.

79. Yocto Project, «Yocto Project», [Online]. Available: https://www.yoctoproject.org/.

80. OpenCV team, «About OpenCV», OpenCV team, [Online]. Available: https://OpenCV.org/about.html.