BÁO CÁO DỌo ÁN

Nhaạn Dieẹn Bieẹn Soọ Xe voọi Sobel Filter và Các Kyỵ Thuaạt Song Song Hóa

Sinh viên: [Tên cuụa baạn]

Môn hOọc: Laạp trình Song song

December 30, 2025

Contents

CHỤONG 1. GIOọI THIEẹU

1.1 Ḍaạt vaạn ḍeẹ

1.1.1 Bài toán

Trong boọi caạnh phát trieẹn cuụa công ngheẹ thông minh, vieẹc nhaạn dieẹn bieẹn soọ xe tuụ ḍoọng ḍóng vai trò quan trOọng trong nhieẹu uụng dUụng thuục teẹ:

- **Giám sát giao thông**: Tuụ ḍoọng phát hieẹn và theo dõi phụọng tieẹn, quaạn lý lụu lụoọng giao thông
- **Bãi giuụ xe thông minh**: Tuụ ḍoọng nhaạn dieẹn bieẹn soọ ḍeẹ tính phí, quaạn lý ra vào
- **Nhaạn daạng phụọng tieẹn tuụ ḍoọng**: Hoọ troọ heẹ thoọng an ninh, tìm kieẹm phụọng tieẹn

1.1.2 Khó khăn

Tuy nhiên, bài toán nhaạn dieẹn bieẹn soọ xe gaạp nhieẹu thách thuục:

- **Nhieẹu aạnh**: Aạnh có theẹ biị nhieẹu do ḍieẹu kieẹn chUụp, chaạt lụoọng camera
- **Ánh sáng thay ḍoọi**: Ḍieẹu kieẹn ánh sáng khác nhau (ngày/ḍêm, naạng/bóng râm) aạnh hụoọng ḍeẹn chaạt lụoọng aạnh
- **Aạnh ḍoọ phân giaại thaạp**: Camera ḍaạt xa hoaạc chaạt lụoọng thaạp làm giaạm ḍoọ rõ nét
- **Toọc ḍoọ xuụ lý real-time**: Yêu caạu xuụ lý nhanh ḍeẹ ḍáp uụng uụng dUụng thoọi gian thuục

1.2 MUục tiêu ḍeẹ tài

1.2.1 Áp dUụng boọ lOọc Sobel

- **Làm noọi biên**: Phát hieẹn và tăng cụoọng các caạnh cuụa bieẹn soọ, giúp model deẹ dàng nhaạn dieẹn họn
- **Giaạm nhieẹu không caạn thieẹt**: LOọc boọ các chi tieẹt không quan trOọng, taạp trung vào thông tin biên

1.2.2 Song song hóa Sobel

Ḍeẹ ḍaạt hieẹu năng cao, ḍeẹ tài taạp trung vào song song hóa boọ lOọc Sobel:

- **CPU ḍa luoọng**: Suụ dUụng OpenMP ḍeẹ chia nhoọ công vieẹc cho nhieẹu cores
- **GPU**: Taạn dUụng CUDA ḍeẹ xuụ lý song song hàng nghìn pixels cùng lúc
- **Vector hóa**: Suụ dUụng SIMD (AVX-256) ḍeẹ xuụ lý nhieẹu pixels trong moọt instruction

1.2.3 Keẹt hoọp model hOọc sâu

- **Phát hieẹn bieẹn soọ chính xác họn**: Suụ dUụng YOLOv5 dẹẹ phát hieẹn vùng bieẹn soọ
- **Giaạm thoọi gian suy luaạn**: Toọi ụu pipeline dẹẹ dạạt throughput cao

1.3 Phaạm vi nghiên cuụu

- **Aạnh tĩnh / video**: Heẹ thoọng hoọ troọ xuụ lý caạ aạnh dọn leẹ và video stream
- **OCR bieẹn soọ**: Tích hoọp mô hình OCR dẹẹ nhaạn dieẹn ký tuụ trên bieẹn soọ
- **So sánh có Sobel vs không Sobel**: Dánh giá aạnh hụoọng cuụa tieẹn xuụ lý Sobel dẹẹn dọọ chính xác

1.4 Dóng góp cuụa dẹẹ tài

- **Dẹẹ xuaạt pipeline tieẹn xuụ lý song song**: Keẹt hoọp Sobel filter voọi các kyỵ thuaạt song song hóa
- **Dánh giá aạnh hụoọng Sobel toọi model**: Phân tích tác dọọng cuụa tieẹn xuụ lý dẹẹn hieẹu quaạ detection
- **So sánh hieẹu năng**: Benchmark chi tieẹt giuụa các phụọng pháp (CPU scalar, SIMD, CUDA)

CHỤỌNG 2. CỌ SOọ LÝ THUYEẹT

2.1 Toọng quan xuụ lý aạnh soọ

2.1.1 Khái nieẹm cọ baạn

- **Pixel**: Dọn viị nhoọ nhaạt cuụa aạnh soọ, moọi pixel có giá triị màu (grayscale hoaạc RGB)
- **Kernel**: Ma traạn nhoọ dùng dẹẹ áp dUụng các phép toán lên aạnh
- **Convolution**: Phép toán nhân kernel voọi tuụng vùng aạnh dẹẹ taạo aạnh moọi

2.1.2 LOọc không gian (Spatial Filtering)

LOọc không gian là kyỵ thuaạt xuụ lý aạnh baạng cách áp dUụng moọt kernel lên tuụng pixel và các pixel lân caận:

$$I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} K(i,j) \cdot I(x+i, y+j) \tag{1}$$

Trong dó:

- $I(x,y)$: Giá triị pixel taại viị trí (x,y) cuụa aạnh goọc

- $I'(x, y)$: Giá trị pixel sau khi lọc
- $K(i, j)$: Giá trị kernel tại vị trí (i, j)
- k: Kích thước kernel (thường là 1 cho kernel 3x3)

2.2 Bộ lọc Sobel

2.2.1 Nguyên lý phát hiện biên

Bộ lọc Sobel là một toán tử gradient dùng để phát hiện biên cạnh trong ảnh. Sobel tính gradient theo hai hướng:

- **Gradient theo X (Gx)**: Phát hiện biên dọc
- **Gradient theo Y (Gy)**: Phát hiện biên ngang

2.2.2 Kernel Sobel

Kernel Sobel 3x3 được định nghĩa như sau:

Kernel Gx (phát hiện biên dọc):

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

Kernel Gy (phát hiện biên ngang):

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3}$$

2.2.3 Tính độ lớn gradient

Độ lớn gradient được tính bằng:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{4}$$

Hoặc xấp xỉ:

$$|G| \approx |G_x| + |G_y| \tag{5}$$

2.2.4 Đặc điểm

- **Nhấn mạnh biên**: Làm nổi bật các cạnh và đường viền trong ảnh
- **Nhạy với nhiễu**: Có thể tăng cường nhiễu nếu ảnh gốc có nhiễu nhiều
- **Hướng biên**: Có thể xác định hướng của biên thông qua $\arctan(G_y/G_x)$

2.3 Song song trong xuụ lý aạnh

2.3.1 Tinh chaạt

Xuụ lý aạnh có tính chaạt song song tuụ nhiên:

- **Moọi pixel xuụ lý ḍoọc laạp**: Các pixel có theẹ ḍụoọc xuụ lý song song mà không phUụ thuoọc nhau (truụ border pixels)
- **Duụ lieẹu cUục boọ**: Moọi pixel chIị caạn duụ lieẹu tuụ các pixel lân caạn
- **Tinh toán ḍoọng nhaạt**: Cùng moọt phép toán áp dUụng cho taạt caả pixels

2.3.2 Các mô hình song song

CPU ḍa luoọng (OpenMP, Threading)

- Chia aạnh thành các vùng (theo dòng hoaạc block)
- Moọi thread xuụ lý moọt vùng ḍoọc laạp
- Suụ dUụng `#pragma omp parallel for` ḍeẹ tuụ ḍoọng phân chia

GPU (CUDA / OpenCL)

- Moọi thread GPU xuụ lý moọt hoaạc moọt nhóm pixels
- Hàng nghìn threads chaạy song song
- Taạn dUụng boọ nhoọ shared và texture memory

SIMD / Vectorization

- Xuụ lý nhieẹu pixels cùng lúc trong moọt instruction
- AVX-256: Xuụ lý 8 giá triị float (32-bit) hoaạc 32 giá triị int (8-bit) cùng lúc
- Giaạm soọ lụoọng instructions và tăng throughput

2.4 Bài toán phát hieẹn bieẹn soọ xe

2.4.1 Object Detection

Phát hieẹn bieẹn soọ xe là bài toán Object Detection, caạn:

- Xác ḍiịnh viị trí bieẹn soọ trong aạnh (bounding box)
- Phân loaại ḍoọi tụoọng (có/không có bieẹn soọ)

2.4.2 Các hụoọng tieẹp caạn

Truyeẹn thoọng

- **Haar Cascade**: Suụ dUụng các ḍaạc trụng Haar-like
- **Edge-based**: Duụa trên phát hieẹn biên và hình daạng hình hOọc
- Haạn cheẹ: Ḍoọ chính xác thaạp, nhaạy voọi ḍieẹu kieẹn ánh sáng

HOọc sâu

- **YOLO (You Only Look Once)**: Phát hieẹn real-time, xuụ lý toàn boọ aạnh moọt laạn
- **SSD (Single Shot Detector)**: Tụọng tuụ YOLO, suụ dUụng multi-scale features
- **Faster R-CNN**: Dọọ chính xác cao nhụng chaạm họn

2.5 Mô hình hOọc sâu suụ dUụng

2.5.1 YOLOv5

Kieẹn trúc toọng quan

- Backbone: CSPDarknet53 (Cross Stage Partial Network)
- Neck: PANet (Path Aggregation Network)
- Head: 3 scale detection heads

Dạạu vào / Dạạu ra

- **Input**: Aạnh RGB, kích thụọọc 640x640 pixels
- **Output**: Bounding boxes voọi format $(x_1, y_1, x_2, y_2, confidence, class\_id)$

Loss function

- Classification loss: BCE (Binary Cross Entropy)
- Localization loss: CIoU (Complete IoU)
- Objectness loss: BCE

Ụu dieẹm – Haạn cheẹ Ụu dieẹm:

- Toọc dọọ nhanh, phù hoọp real-time
- Dọọ chính xác cao
- Deẹ tích hoọp và trieẹn khai

Haạn cheẹ:

- Caạn dataset loọn dẹẹ huaạn luyeẹn
- Yêu caạu GPU dẹẹ dạạt hieẹu năng toọt
- Khó phát hieẹn dọọi tụọng nhoọ

CHỤƠNG 3. PHỤƠNG PHÁP DẸẹ XUAạT

3.1 Toọng quan heẹ thoọng

3.1.1 Pipeline xuụ lý

Heẹ thoọng dụoọc thieẹt keẹ theo kieẹn trúc pipeline dạ luoọng:

```
Aạnh goọc
  ↓
[Sobel song song] → Aạnh tăng cụoọng biên
  ↓
[Model phát hieẹn bieẹn soọ] → Bounding boxes
  ↓
[OCR ký tuụ] → Text bieẹn soọ
  ↓
Keẹt quaạ cuoọi cùng
```

3.1.2 Các thành phaạn chính

1. **Capture Stage**: DOọc frame tuụ camera/video
2. **Sobel Filter Stage**: Phát hieẹn biên voọi song song hóa
3. **Detection Stage**: Phát hieẹn vùng bieẹn soọ baạng YOLOv5
4. **OCR Stage**: Nhaạn dieẹn ký tuụ trên bieẹn soọ
5. **Render Stage**: Hieẹn thiị keẹt quaạ

3.2 Sobel song song

3.2.1 Cách chia aạnh

Theo dòng (Row-based)

- Chia aạnh thành các dòng
- Moọi thread xuụ lý moọt hoaạc nhieẹu dòng
- Phù hoọp voọi OpenMP

Theo block (Block-based)

- Chia aạnh thành các block 2D (ví dUụ: 64x64 pixels)
- Moọi thread/block xuụ lý moọt vùng
- Phù hoọp voọi GPU CUDA

3.2.2 Chieẹn lụoọc song song

Multi-thread CPU (OpenMP)

Listing 1: Sobel voọi OpenMP

```
#pragma omp parallel for
for (int y = 1; y < rows - 1; ++y) {
    for (int x = 1; x < cols - 1; ++x) {
        // Tinh Sobel cho pixel (x,y)
    }
}
```

GPU Kernel (CUDA)

Listing 2: Sobel CUDA Kernel

```
__global__ void sobelKernel(uchar* src, uchar* dst,
                            int width, int height) {
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    //  M i  thread  x   l   1 pixel
}
```

SIMD Vectorization (AVX-256)

Listing 3: Sobel voọi SIMD

```
//  X   l   8 pixels  c  ng  l  c
__m256 gx = _mm256_add_ps(...);
__m256 gy = _mm256_add_ps(...);
__m256 mag = _mm256_sqrt_ps(...);
```

3.2.3 Pseudocode Sobel song song

Thuaạt toán: Sobel Filter Song Song

```
Input:  Anh dau vao I (grayscale)
Output: Anh bien E

1. Chia anh thanh N vung
2. PARALLEL for moi vung i tu 1 den N:
   a. FOR moi pixel (x,y) trong vung i:
      - Tinh G_x = sum K_x * I(x+i, y+j)
      - Tinh G_y = sum K_y * I(x+i, y+j)
      - E(x,y) = sqrt(G_x^2 + G_y^2)
3. RETURN E
```

3.3 Tích hoọp Sobel vào pipeline AI

3.3.1 Sobel trụoọc model

Sobel filter dụoọc áp dUụng nhụ buoọc tieẹn xuụ lý trụoọc khi dụa vào model:

- **Làm noọi biên**: Tăng cụoọng các caạnh cuụa bieẹn soọ

- **Giaạm nhieẹu**: LOọc boọ các chi tieẹt không caạn thieẹt
- **Caại thieẹn contrast**: Làm rõ họn suụ khác bieẹt giuụa bieẹn soọ và neẹn

3.3.2 Chuaạn hóa aạnh

- Chuyeẹn aạnh Sobel veẹ grayscale (neẹu caạn)
- Normalize giá triị pixel veẹ khoaạng [0, 1]
- Resize veẹ kích thụoọc input cuụa model (640x640)

3.3.3 Resize phù hoọp input model

Model YOLOv5 yêu caạu input 640x640 pixels:

Listing 4: Pre-processing cho model

```
cv::Mat sobel;  //   nh   sau Sobel
cv::Mat resized;
cv::resize(sobel, resized, cv::Size(640, 640));
resized.convertTo(resized, CV_32F, 1.0/255.0);
```

3.4 Mô hình phát hieẹn bieẹn soọ

3.4.1 Caạu trúc model

- **Model**: YOLOv5 nano (YOLOv5n)
- **Format**: ONNX (dẹẹ deploy trong C++)
- **Input size**: 640x640x3

3.4.2 Input shape

- Kích thụoọc: [1, 3, 640, 640]
- Kieẹu duụ lieẹu: Float32
- Giá triị: Normalized veẹ [0, 1]

3.4.3 Output bounding box

- Format: [batch, num_detections, 6]
- 6 giá triị: $(x_1, y_1, x_2, y_2, confidence, class\_id)$
- TOọa dọọ dụọọc normalize veẹ [0, 1], caạn scale veẹ kích thụoọc aạnh goọc

3.4.4 Tham soọ huaạn luyeẹn

- Dataset: Custom dataset bieẹn soọ Vieẹt Nam
- Epochs: 100
- Batch size: 16
- Learning rate: 0.01
- Optimizer: SGD voọi momentum

3.5 Luoọng xuụ lý toọng theẹ

3.5.1 Soị ḍoọ khoọi heẹ thoọng

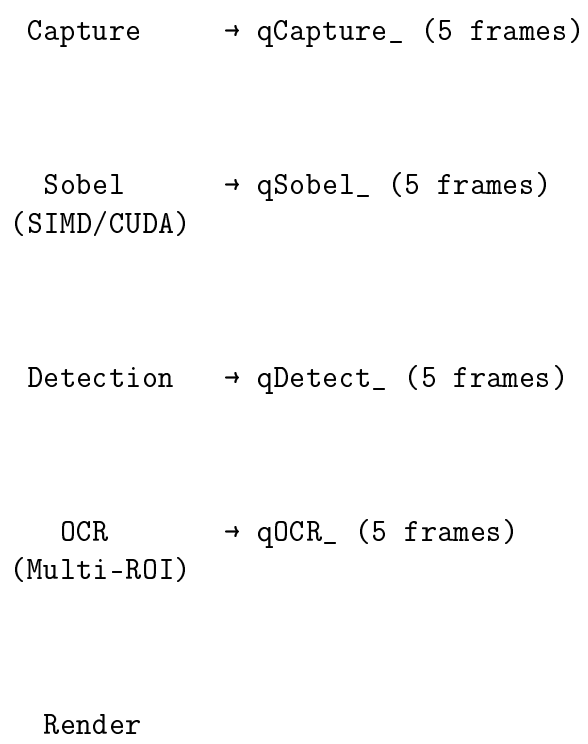

Figure 1: Kieẹn trúc Pipeline ḍa luoọng

3.5.2 Trình tuị xuụ lý duụ lieẹu

1. **Capture**: ḌOọc frame tuị camera/video vào queue
2. **Sobel**: Áp dUụng Sobel filter song song, ḍaạy vào queue tieẹp theo
3. **Detection**: Phát hieẹn bieẹn soọ, traạ veẹ bounding boxes
4. **OCR**: Nhaạn dieẹn ký tuị trên tuụng vùng bieẹn soọ (song song neẹu có nhieẹu ROI)
5. **Render**: Veẹ keẹt quaạ lên frame và hieẹn thiị

CHỤƠNG 4. CÀI ĐẠẶT VÀ THỰC NGHIỆM

4.1 Môi trựờng thực nghiệm

4.1.1 Phần cứng

- **CPU**: [Ghi rõ model CPU, sờ cores]
- **GPU**: [Ghi rõ model GPU nếu có, ví dỤ: NVIDIA GTX/RTX]
- **RAM**: [Sờ GB]
- **OS**: Linux (Ubuntu/Debian)

4.1.2 Phần mềm

- **Language**: C++17
- **OpenCV**: Version 4.x (xử lý ảnh)
- **ONNX Runtime**: Version 1.16.3 (inference model)
- **CUDA**: Version [nếu có GPU]
- **Compiler**: GCC vởi flags `-fopenmp -mavx2`

4.2 Dataset

4.2.1 Nguồn dữ liệu

- Dataset custom: Ảnh biển sờ xe Việt Nam
- Nguồn: Thu thập tử camera giao thông, bãi đỢ xe

4.2.2 Sờ lựợng ảnh

- Training set: [Sờ lựợng]
- Validation set: [Sờ lựợng]
- Test set: [Sờ lựợng]

4.2.3 Điều kiện chỤp

- Đa dạng điều kiện ánh sáng (ngày/đêm)
- Nhiều góc chỤp khác nhau
- Đỡ phân giải: 640x480 đến 1920x1080
- Có nhiễu và blur trong mợt sờ ảnh

4.3 Các kiịch baạn thuục nghieẹm

4.3.1 Không Sobel

- Dụa aạnh goọc truục tieẹp vào model
- Không có bụoọc tieẹn xuụ lý
- Baseline dẹẹ so sánh

4.3.2 Sobel dọn luoọng

- Áp dUụng Sobel filter tuaạn tuụ (không song song)
- Xuụ lý tuụng pixel moọt cách tuaạn tuụ
- Dọ thoọi gian xuụ lý

4.3.3 Sobel song song

- **CPU OpenMP**: Chia aạnh theo dòng, moọi thread xuụ lý moọt phaạn
- **CPU SIMD**: Suụ dUụng AVX-256 dẹẹ xuụ lý 8 pixels cùng lúc
- **GPU CUDA**: Moọi thread GPU xuụ lý moọt pixel

4.4 ChIị soọ dánh giá

4.4.1 Dọọ chính xác (Precision, Recall, mAP)

- **Precision**: Tyỵ leẹ detection dúng trong toọng soọ detection
- **Recall**: Tyỵ leẹ bieẹn soọ dụoọc phát hieẹn trong toọng soọ bieẹn soọ thuục teẹ
- **mAP (mean Average Precision)**: Dọọ chính xác trung bình oọ các muục IoU khác nhau

4.4.2 Thoọi gian xuụ lý

- Thoọi gian Sobel filter (ms/frame)
- Thoọi gian detection (ms/frame)
- Thoọi gian OCR (ms/frame)
- Toọng thoọi gian pipeline (ms/frame)

4.4.3 FPS (Frames Per Second)

Soọ frame xuụ lý dụoọc trong 1 giây:

$$FPS = \frac{1000}{thoọi\_gian\_xuụ\_l\_(ms)} \tag{6}$$

4.4.4 Latency

Thoọi gian tuụ khi nhaạn frame dẹẹn khi có keẹt quaạ (end-to-end latency).

4.5 Keẹt quaạ thuục nghieẹm

4.5.1 Baạng so sánh hieẹu năng Sobel Filter

Benchmark dụoọc chaạy trên aạnh `bienso.jpg` (800x600 pixels, 0.48 MP), 100 runs:

Phụọng pháp	Thoọi gian (ms)	FPS	Speedup
CPU OpenMP (scalar)	7.144	140.0	1.0x (baseline)
CPU SIMD (AVX-256)	1.885	530.4	**3.79x**
GPU CUDA	**0.494**	**2025.5**	**14.47x**

Table 1: Keẹt quaạ benchmark Sobel filter - Data Parallelism

4.5.2 Phân tích keẹt quaạ

- **SIMD vectorization**: Caại thieẹn **3.79x** so voọi scalar code baạng cách xuụ lý 8 pixels cùng lúc
- **CUDA GPU**: Caại thieẹn **14.47x** so voọi CPU scalar và **3.82x** so voọi SIMD
- **GPU là phụọng pháp nhanh nhaạt**: Phù hoọp cho xuụ lý real-time voọi throughput cao

4.5.3 Bieẹu dọọ hieẹu năng

```
Thoọi gian xuụ lý (ms):
CPU Scalar:   7.144
CPU SIMD:     1.885
GPU CUDA:     0.494

Speedup:
GPU CUDA:     14.47x
CPU SIMD:     3.79x
CPU Scalar:   1.0x
```

Figure 2: So sánh hieẹu năng các phụọng pháp Sobel

4.5.4 Keẹt quaạ Detection và OCR

- **Dọọ chính xác detection**: [Ghi keẹt quaạ neẹu có]
- **Dọọ chính xác OCR**: [Ghi keẹt quaạ neẹu có]
- **Aạnh hụọng cuụa Sobel**: [Phân tích xem Sobel có caại thieẹn accuracy không]

CHỤONG 5. ĐÁNH GIÁ VÀ THAạO LUAạN

5.1 Phân tích keẹt quaạ

5.1.1 Sobel aạnh hụoọng theẹ nào toọi model

- **Tăng cụoọng biên**: Sobel làm noọi baạt các caạnh cuụa bieẹn soọ, giúp model deẹ dàng phát hieẹn họn
- **Giaạm nhieẹu**: LOọc boọ các chi tieẹt không caạn thieẹt, taạp trung vào thông tin quan trOọng
- **Caại thieẹn contrast**: Làm rõ họn suụ khác bieẹt giuụa bieẹn soọ và neẹn
- **Aạnh hụoọng tiêu cuục**: Có theẹ tăng cụoọng nhieẹu neẹu aạnh goọc đã có nhieẹu nhieẹu

5.1.2 Loọi ích song song hóa

- **Tăng throughput**: Xuụ lý nhieẹu frame họn trong cùng moọt khoaạng thoọi gian
- **Giaạm latency**: Thoọi gian xuụ lý moọi frame giaạm đáng keẹ
- **Taạn dUụng tài nguyên**: Suụ dUụng hieẹu quaạ CPU cores và GPU
- **Scalability**: Có theẹ moọ roọng voọi nhieẹu cores/GPU họn

5.1.3 So sánh các phụọng pháp song song

- **CPU OpenMP**: Deẹ trieẹn khai, phù hoọp voọi heẹ thoọng không có GPU
- **CPU SIMD**: Caại thieẹn đáng keẹ voọi chi phí thaạp (chỊị caạn compiler flags)
- **GPU CUDA**: Nhanh nhaạt nhụng yêu caạu GPU và phuục taạp họn trong trieẹn khai

5.2 So sánh voọi phụọng pháp khác

5.2.1 Không tieẹn xuụ lý

- **Ụu đieẹm**: Đọn giaạn, không toọn thoọi gian tieẹn xuụ lý
- **Nhụoọc đieẹm**: Model có theẹ khó phát hieẹn trong đieẹu kieẹn ánh sáng xaạu hoaạc nhieẹu

5.2.2 Tieẹn xuụ lý truyeẹn thoọng

- **Gaussian Blur**: Làm miịn aạnh, giaạm nhieẹu nhụng có theẹ làm moọ biên
- **Histogram Equalization**: Caại thieẹn contrast nhụng có theẹ làm tăng nhieẹu
- **Sobel**: Taạp trung vào biên, phù hoọp voọi bài toán phát hieẹn bieẹn soọ

5.3 Haạn cheẹ cuụa dẹẹ tài

5.3.1 Nhaạy nhieẹu

- Sobel filter có theẹ tăng cụoọng nhieẹu trong aạnh
- Caạn keẹt hoọp voọi các boọ lOọc khác (ví dUụ: Gaussian blur trụoọc Sobel)

5.3.2 Dieẹu kieẹn ánh sáng xaạu

- Trong dieẹu kieẹn ánh sáng yeẹu, Sobel có theẹ không hieẹu quaả
- Caạn caại thieẹn pre-processing (normalization, contrast enhancement)

5.3.3 Tài nguyên phaạn cuụng

- GPU CUDA yêu caạu GPU NVIDIA, không phù hoọp voọi heẹ thoọng nhúng
- SIMD yêu caạu CPU hoọ troọ AVX (tuụ Intel Sandy Bridge troọ lên)
- Caạn cân nhaạc trade-off giuụa hieẹu năng và chi phí

CHỤỌNG 6. KEẹT LUAạN VÀ HỤOọNG PHÁT TRIEẹN

6.1 Keẹt luaạn

6.1.1 Toọng keẹt keẹt quaả dạạt dụoọc

Dẹẹ tài dã thành công trieẹn khai heẹ thoọng nhaạn dieẹn bieẹn soọ xe voọi các kyỵ thuaạt song song hóa:

- Pipeline hoàn chỊnh: Capture → Sobel → Detection → OCR → Render
- Trieẹn khai 3 phụọng pháp song song: CPU OpenMP, CPU SIMD, GPU CUDA
- Tích hoọp YOLOv5 cho detection và OCR
- Dánh giá và so sánh hieẹu năng chi tieẹt

6.1.2 Muục caại thieẹn so voọi baseline

- **CPU SIMD**: Caại thieẹn **3.79x** so voọi CPU scalar (tuụ 7.144ms xuoọng 1.885ms)
- **GPU CUDA**: Caại thieẹn **14.47x** so voọi CPU scalar (tuụ 7.144ms xuoọng 0.494ms)
- **Throughput**: Dạạt **2025.5 FPS** voọi CUDA, phù hoọp cho uụng dUụng real-time

6.1.3 Dóng góp chính

- Dẹẹ xuaạt pipeline tieẹn xuụ lý song song hieẹu quaả
- Benchmark chi tieẹt so sánh các phụọng pháp song song hóa
- Tích hoọp thành công Sobel filter voọi model hOọc sâu

6.2 Huoọng phát trieẹn

6.2.1 OCR bieẹn soọ

- Caại thieẹn ọoọ chính xác OCR voọi các kyỵ thuaạt post-processing
- Xuụ lý các trụoọng hoọp ọaạc bieẹt (bieẹn soọ biị moọ, góc nghiêng)
- Tích hoọp dictionary ọeẹ suụa loọi OCR

6.2.2 Real-time video

- Toọi ụu pipeline ọeẹ xuụ lý video stream real-time
- Suụ dUụng frame skipping và temporal smoothing
- Tích hoọp voọi heẹ thoọng streaming (RTSP, WebRTC)

6.2.3 Trieẹn khai Edge AI

- Toọi ụu model cho thieẹt biị nhúng (Jetson Nano, Raspberry Pi)
- Quantization và pruning ọeẹ giaạm kích thụoọc model
- Suụ dUụng TensorRT cho inference nhanh họn

6.2.4 Toọi ụu GPU sâu họn

- Suụ dUụng CUDA streams ọeẹ overlap computation và memory transfer
- Toọi ụu memory access pattern (coalesced access)
- Suụ dUụng shared memory và texture memory hieẹu quaạ họn
- Batch processing ọeẹ tăng throughput

6.2.5 Moọ roọng chuục năng

- Hoọ troọ nhieẹu loaại bieẹn soọ (xe máy, xe taại, v.v.)
- Tích hoọp voọi database ọeẹ tra cuụu thông tin xe
- Phát hieẹn vi phaạm giao thông tuụ ọoọng

CHỤỌNG A. Code Examples

A.1 Sobel SIMD Implementation

Xem file: `Main/SobelSIMD.cpp`

A.2 Sobel CUDA Kernel

Xem file: `Main/SobelCuda.cu`

A.3 Pipeline Multi-threading

Xem file: `Main/Pipeline.cpp`

CHỤỌNG B. Build Instructions

```
# Cài đaạt dependencies
sudo apt-get install libopencv-dev
pip install onnx onnxruntime yolov5

# Export models
cd Main
python3 export_torchscript.py

# Build (CPU only)
g++ -std=c++17 -fopenmp -mavx2 \
    -I/tmp/onnxruntime-linux-x64-1.16.3/include \
    Main/main.cpp Main/Pipeline.cpp Main/LPDetector.cpp \
    Main/LPOCR.cpp Main/SobelSIMD.cpp \
    -o lp_main \
    `pkg-config --cflags --libs opencv4` \
    -L/tmp/onnxruntime-linux-x64-1.16.3/lib -lonnxruntime

# Build voọi CUDA
nvcc -c Main/SobelCuda.cu -o SobelCuda.o \
     -I/usr/local/cuda/include \
     `pkg-config --cflags opencv4` -arch=sm_50
g++ -std=c++17 -fopenmp -mavx2 -DUSE_CUDA_SOBEL \
    Main/main.cpp Main/Pipeline.cpp Main/LPDetector.cpp \
    Main/LPOCR.cpp Main/SobelSIMD.cpp SobelCuda.o \
    -o lp_main \
    `pkg-config --cflags --libs opencv4` \
    -L/tmp/onnxruntime-linux-x64-1.16.3/lib -lonnxruntime \
    -L/usr/local/cuda/lib64 -lcudart

# Chaạy
./lp_main 0  # Camera
./lp_main image.jpg  # Aạnh
```

CHỤỌNG C. Tài lieẹu tham khaạo

- OpenCV Documentation: `https://docs.opencv.org/`
- ONNX Runtime: `https://onnxruntime.ai/`
- YOLOv5: `https://github.com/ultralytics/yolov5`
- OpenMP Specification: `https://www.openmp.org/`

- Intel AVX Intrinsics: `https://www.intel.com/content/www/us/en/docs/intrinsics-guide/`
- CUDA Programming Guide: `https://docs.nvidia.com/cuda/`