การพัฒนาระบบตรวจนับปริมาณรถด้วยการประมวลผลภาพจากกล้องวิดีโอบน FPGA

Development of a Vehicle Counting System by Image Processing from Video Camera on an FPGA

ชลธิศา เวทโอสถ

Chonthisa Wateosot

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering

Prince of Songkla University

2552

ชื่อวิทยานิพนธ์ การพัฒนาระบบตรวจนับปริมาณรถด้วยการประมวลผลภาพจากกล้องวิดีโอบน FPGA

ผู้เขียน นางสาวชลธิศา เวทโอสถ

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	**คณะกรรมการสอบ**
.....	..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)	(รองศาสตราจารย์ ดร.มนตรี กาญจนะเดชะ)
	..... กรรมการ
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	(ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)
.....	..... กรรมการ
(ดร.นิคม สุวรรณวร)	(ดร.นิคม สุวรรณวร)
.....	..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.พรชัย พฤกษ์ภัทรานนต์)	(รองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....

(รองศาสตราจารย์ ดร.เกริกชัย ทองหนู)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์	การพัฒนาระบบตรวจนับปริมาณรถด้วยการประมวลผลภาพจากกล้องวิดีโอบน FPGA
ผู้เขียน	นางสาวชลธิศา เวทโอสถ
สาขาวิชา	วิศวกรรมไฟฟ้า
ปีการศึกษา	2552

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอการออกแบบระบบการนับปริมาณรถอย่างง่ายโดยใช้กระบวนการประมวลผลภาพด้วยภาษาระดับสูง ImpulseC เพื่อช่วยในการออกแบบและพัฒนาร่วมกันระหว่างซอฟท์แวร์และฮาร์ดแวร์ ทำให้ช่วยลดระยะเวลาและความยุ่งยากในการสร้างวงจรและวงจรที่ได้สามารถถูกพัฒนาและจำลองแบบได้บนเทคโนโลยี FPGA (Field Programmable Gate Array) เพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้เร็วยิ่งขึ้น อัลกอริธึมการประมวลผลภาพเริ่มจากการแยกภาพรถจากถนนด้วยวิธีการหาผลต่างเฟรม การแก้ไขมุมมองภาพแบบ Perspective เพื่อขยายภาพของรถให้มีสัดส่วนเดียวกันเพื่อแก้ปัญหาในบริเวณที่ภาพรถอยู่ไกลจากกล้องซึ่งรถมีขนาดเล็กลง และทำการนับจำนวนรถด้วยการหาพื้นที่จุดสีทั้งหมดหารด้วยค่าเฉลี่ยของรถ 1 คัน อัลกอริธึมดังกล่าวถูกพัฒนาให้มีความเร็วเพิ่มขึ้นโดยใช้เทคนิคไปป์ไลน์และการทำงานแบบขนานบน FPGA ระบบถูกทดสอบด้วยการใช้ลำดับภาพวิดีโอในสกุล pgm ขนาด 320×240 พิกเซล และให้ระบบแสดงผลลัพธ์เป็นจำนวนคันของรถ ระบบสามารถประมวลผลได้เสร็จภายในเวลา 0.37 วินาทีต่อเฟรม และใช้ทรัพยากรบนชิพ Vertex II Pro XC2VP30 เท่ากับ 22% ใช้ Slices จำนวน 3077 และหน่วยความจำขนาด 18 กิโลไบต์จำนวน 3 บล็อกแรม และจากการทดสอบภาพวิดีโอจำนวน 400 ภาพ ได้ผลความผิดพลาดในการนับรถไม่เกิน 2 คัน

Thesis Title	Development of a Vehicle Counting System by Image Processing from Video Camera on an FPGA
Author	Miss. Chonthisa Wateosot
Major Program	Electrical Engineering
Academic Year	2009

ABSTRACT

This thesis presents a design methodology of a simple vehicle counting circuit using a system-level language named ImpulseC. Using this , it can reduce the hardware/software co-design development time and allows completely simulating on FPGA (Field Programmable Gate Array). The proposed algorithm uses the frame different based background subtraction to separate the vehicles from the road, applies perspective view correction for vehicles far from the camera. The vehicle count is calculated from the area of normalized pixels. The algorithm was implemented to achieve higher speed by pipelining to execute multiple iteration loop in parallel on an FPGA. The system was tested by using the video streaming in PGM format and the resolution of 320x240 pixels. The system result shows the vehicle count. The system processing time was 0.37 second per frame. The utilize resources were 22% of Vertex II Pro XC2VP30 including 3077 slices and 2 blocks of 18 kB RAM. From the results of 400 video frames, the error of vehicle counting was no more than two cars.

กิตติกรรมประกาศ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์ ประธานกรรมการที่ปรึกษางานวิจัย ที่ได้เสียสละเวลาในการให้คำปรึกษา แนวคิดในการทำวิจัย รวมถึงการช่วยเหลือแก้ไขปัญหาที่เกี่ยวกับการวิจัย ตลอดจนตรวจสอบและแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างลุล่วงสมบูรณ์

ขอขอบพระคุณ ดร.นิคม สุวรรณวร ที่ได้เสียสละเวลาในการให้คำปรึกษา แนวคิดในการทำวิจัย คำแนะนำ และให้ความช่วยเหลือในงานวิจัย ตลอดจนช่วยตรวจทานแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างลุล่วงสมบูรณ์

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.พรชัย พฤกษ์ภัทรานนต์ ที่ได้กรุณาให้คำปรึกษา คำแนะนำ และให้ความช่วยเหลือในงานวิจัย

ขอขอบพระคุณ รองศาสตราจารย์ ดร.มนตรี กาญจนะเดชะ ที่ได้กรุณาเสียสละเวลาเป็นประธานกรรมการสอบวิทยานิพนธ์และตรวจทานแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ขอขอบพระคุณ รองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี ที่กรุณาเสียสละเวลาเป็นกรรมการสอบวิทยานิพนธ์ อีกทั้งตรวจทานและแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ที่ให้การสนับสนุนทุนในการทำวิจัยและให้ความช่วยเหลือด้านการประสานงานต่างๆ

ขอขอบพระคุณ เทศบาลนครหาดใหญ่ในความเอื้อเฟื้อภาพวิดีโอสำหรับการทดสอบการออกแบบระบบนับรถบนเอฟพีจีเอ

ขอขอบพระคุณ คณาจารย์ บุคลากร และนักศึกษาปริญญาโทภาควิชาวิศวกรรมไฟฟ้าทุกคนที่ได้ให้คำปรึกษา และกำลังใจในการทำงานเป็นอย่างดีเสมอมา

และสุดท้าย ข้าพเจ้าขอน้อมรำลึกถึงพระคุณของ บิดามารดา และครอบครัว ที่ส่งเสริมและสนับสนุนข้าพเจ้าในทุกๆ เรื่องตลอดมาจนสำเร็จการศึกษา

ชลธิศา เวทโอสถ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

รายการตาราง

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

รายการภาพประกอบ (ต่อ)

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของหัวข้อวิจัย

ปัญหาการจราจรที่ติดขัดเป็นปัญหาสำคัญ โดยเฉพาะในช่วงเวลารีบเร่งที่มีปริมาณรถยนต์ในการจราจรมากกว่าช่วงปกติ ทำให้เกิดปัญหามากมาย เช่น ไม่มีความคล่องตัวในการจราจร สิ้นเปลืองพลังงาน เกิดอุบัติเหตุได้ง่าย เกิดปัญหามลภาวะ เป็นต้น ซึ่งการจัดการกับการปล่อยรถในแยกที่สำคัญๆ หากระบบที่ใช้อยู่เดิมเป็นการตั้งเวลาเอาไว้เท่าๆ กันทุกแยก แล้วทำการปล่อยรถตามลำดับ อาจเป็นวิธีที่ดีและเสมอภาคกันในการใช้ถนน แต่หากเป็นแยกที่มีช่องการจราจรมีรถยนต์เข้าแยกหนาแน่นไม่เท่ากัน แต่กลับใช้การหน่วงเวลาเช่นเดิม จะส่งผลให้แยกที่มีปริมาณรถยนต์หนาแน่น ไม่สามารถระบายรถออกจากแยกได้ แล้วจึงส่งผลกระทบต่อๆไปยังแยกอื่นๆ

หากมีระบบที่สามารถจัดการกับการปล่อยรถออกจากแยกได้อย่างชาญฉลาด ปัญหาเหล่านี้ก็จะได้รับการแก้ไขไปในทางที่ดีขึ้น เป็นเหตุให้เกิดแนวความคิดในการจัดการการจราจรอัจฉริยะ

หลักการทำงานของระบบการจราจรอัจฉริยะสามารถคำนวณความหนาแน่นของปริมาณรถในสี่แยกจราจร เพื่อทำการปล่อยรถยนต์ให้มีประสิทธิภาพมากกว่าในปัจจุบัน ที่ติดกล้องเอาไว้ตามมุมสูงในแยกต่างๆ จะทำหน้าที่จับภาพรถบนท้องถนนแล้วส่งสัญญาณภาพไปประมวลผลภาพบนเซิร์ฟเวอร์ที่ศูนย์ควบคุมกลาง [3]-[7] โดยหากแยกใดมีปริมาณรถหนาแน่น ระบบจะทำการปล่อยสัญญาณไฟเขียวเพื่อให้รถออกไปมากกว่าแยกที่มีปริมาณรถน้อย แต่พบว่าระบบดังกล่าว เซิร์ฟเวอร์ต้องรับภาระงานหนักมากจึงไม่เหมาะกับถนนที่มีทางแยกจำนวนมาก

ในงานวิจัยนี้ เป็นส่วนหนึ่งของระบบสัญญาณไฟจราจรอัจฉริยะ คือ ในส่วนของการตรวจนับรถยนต์เพื่อนำไปคำนวณหาความเหมาะสมในการจัดการระบบการจราจร ขั้นตอนการทำงานของระบบเริ่มจาก สัญญาณภาพแต่ละแยกที่ได้มาจะถูกประมวลผลโดยใช้กระบวนการอิมเมจโปรเซสซิง บนบอร์ดเอฟพีจีเอ ที่ช่วยประมวลผลแทนเซิร์ฟเวอร์ จากข้อมูลปริมาณรถยนต์ในแต่ละแยกนี้เซิร์ฟเวอร์ควรจะสามารถประมวลผลในภาพรวมทั้งระบบเพื่อจัดการจราจรที่มีประสิทธิภาพต่อไป

1.2 การตรวจเอกสาร

1.2.1 Background subtraction techniques : a review [5]

บทความนี้ได้แสดงถึงการวิจารณ์การลบภาพพื้นหลังเพื่อตรวจหาวัตถุที่เคลื่อนที่จากกล้องวิดีโอที่มีสภาพนิ่ง ด้วยเทคนิควิธีต่างๆ รวมทั้งวิเคราะห์ข้อดีข้อเสีย ข้อจำกัดในด้านความเร็ว หน่วยความจำ และความแม่นยำในการประมวลผลในแต่ละเทคนิค ได้แก่ Running Gaussian Average , Temporal Median filter , Mixture of Gaussians , KDE และ Eigenbackgrounds

1.2.2 Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance[6]

บทความนี้แสดงถึงการใช้เทคนิคการลบภาพพื้นหลัง โดยใช้ผลต่างของเฟรมเนื่องจากภาพพื้นหลังค่อนข้างคงที่ นอกจากนี้ งานวิจัยดังกล่าวได้มีการวิเคราะห์บล็อบ และทำการนับรถเมื่อรถผ่านเส้นอ้างอิง แต่มีการนับผิดพลาดประมาณ 10% เนื่องจากการที่สีของภาพพื้นหลังใกล้เคียงกัน

1.2.3 Automatic Vehicle Counting from Video for Traffic Flow Analysis [7]

บทความนี้เป็นการตรวจหารถเคลื่อนที่โดยใช้โมเดลของภาพพื้นหลัง (Gaussian Mixture Modeling) รวมถึงการใช้เทคนิคการวิเคราะห์บล็อบ และคาลมานฟิลเตอร์ ในการติดตามรถที่เคลื่อนที่ ซึ่งเป็นกระบวนการที่ค่อนข้างซับซ้อน

1.2.4 Implementation of Traffic Flow Measuring Algorithm Using Real-Time Dynamic Image Processing [8]

บทความนี้เป็นการตรวจหารถเคลื่อนที่โดยเปรียบเทียบจุดสีของรถกับถนน และใช้บริเวณของวัตถุที่เคลื่อนไหวมานับจำนวนจุดสีเพื่อหาพื้นที่ของรถ พบว่ามีความผิดพลาดในการนับรถประมาณ 2% ซึ่งน้อยมาก และมีการหาความกว้างและความยาวของรถ เพื่อจำแนกประเภทรถ

1.2.5 A Codesign Approach for a High Performance Vehicle Detector [9]

บทความนี้เป็นการออกแบบระบบร่วมฮาร์ดแวร์/ซอฟต์แวร์ ในการนับจำนวนรถโดยใช้วิธีการ 2D FIR ฟิลเตอร์ และฮิสโตแกรมเทรสโฮลด์ โดยทำการนับรถเมื่อรถมีการเคลื่อนที่ผ่านบริเวณสี่เหลี่ยมที่กำหนดขึ้น ซึ่งออกแบบในแพลตฟอร์มของ DSP บนเอฟพีจีเอ ที่เพิ่มประสิทธิภาพในการประมวลผล

1.2.6 ระบบควบคุมสัญญาณไฟจราจรระบบ RONDO (ROLLING-HORIZON DYNAMIC OPTIMAZATION OF SIGNAL CONTROL) : กรณีศึกษาการติดตั้งโครงการนำร่องจังหวัดภูเก็ต[10]

บทความนี้กล่าวถึงการติดตั้งสัญญาณไฟจราจรที่ทันสมัยที่สามารถต่อเชื่อมเป็นโครงข่ายและขยายรูปแบบการควบคุมได้หลายขั้นซึ่งเป็นระบบที่พัฒนาและใช้งานในประเทศญี่ปุ่น โดยอธิบายในแง่การตรวจวัดและการเก็บข้อมูลการจราจรด้วยเครื่องตรวจนับแบบกล่องอิมเมจโพรเซสซิ่ง (Image Processing) และอัลตราโซนิก(Ultrasonic) แนวคิดในการควบคุมสัญญาณไฟจราจร (Traffic Control) การปรับตัวของรอบสัญญาณไฟตามปริมาณจราจรบทความยังแสดงถึงการแก้ปัญหาการจราจร ณ บริเวณทางแยกในโครงการนำร่องในจังหวัดภูเก็ตโดยอาศัยการปรับเปลี่ยนลักษณะทางกายภาพของถนนและการจัดจังหวะสัญญาณไฟให้สอดคล้องกับปริมาณจราจรรวมถึงการติดตั้งสัญญาณไฟระบบ RONDO ผลการใช้งานระบบพบว่าทางแยกที่ติดตั้งนั้นความยาวแถวคอยลดลงแต่ในแยกที่ติดกันถัดไป มีความยาวแถวคอยเพิ่มขึ้นเนื่องจากระบบสัญญาณไฟทั้ง 2 ทางแยกไม่มีความสัมพันธ์กัน

1.2.7 FPGA Architecture for Object Segmentation in Real Time [11]

บทความนี้เป็นการแยกภาพวัตถุในลำดับภาพวิดีโอโดยใช้ FPGA เป็นกระบวนการที่จำเป็นสำหรับหลายๆแอปพลิเคชันของระบบการมองเห็นแบบเทียม เช่น ระบบเฝ้าระวังในภาพจากกล้องวิดีโอ การควบคุมการจราจร การตรวจจับและดึงภาพวัตถุเคลื่อนไหวจากกล้องวิดีโอทางไกล เป็นต้น งานวิจัยนี้ เป็นการนำเสนอการออกแบบสถาปัตยกรรมสำหรับการแยกภาพวัตถุ ประโยชน์ของข้อมูลและการใช้ลอจิกแบบขนานบน FPGA ที่อัตราสัญญาณนาฬิกา 40 MHz ที่ 30 เฟรม/วินาที และภาพที่มีความละเอียด 240 x 120

1.3 วัตถุประสงค์

1.3.1 เพื่อศึกษาและพัฒนาระเบียบวิธีตรวจนับปริมาณรถบนถนนด้วยการประมวลผลภาพจากกล้องวิดีโอ

1.3.2 เพื่อพัฒนาและออกแบบวงจรตรวจนับปริมาณรถบนถนนด้วยการประมวลผลภาพบน FPGA

1.4 ขอบเขตการวิจัย

1.4.1 การนับปริมาณรถจะนับเฉพาะจำนวนรถที่ผ่านกล้องที่มีตำแหน่งคงที่และไม่มีการหมุนของกล้องเท่านั้น

1.4.2 เนื่องจากไม่สามารถติดตั้งกล้องบนถนนทางแยกจริงได้จึงต้องใช้กล้องจากสำนักงานเทศบาลอำเภอหาดใหญ่ ซึ่งมีข้อจำกัดของระยะการมองเห็นภาพ และ คุณสมบัติของกล้องซึ่งมีดังนี้

- กล้อง AXIS 210 network camera
- 30 เฟรมต่อวินาทีที่ความละเอียด 640x480

1.4.3 เอาท์พุตที่ได้จากการประมวลผลเป็นจำนวนรถยนต์ที่นับได้ในแต่ละเฟรมภาพ

1.5 ขั้นตอนและวิธีการดำเนินการวิจัย

1.5.1 ศึกษาและรวมรวมข้อมูลเกี่ยวกับการจราจรในอำเภอหาดใหญ่

1.5.2 ศึกษาเกี่ยวกับเทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม

1.5.3 ออกแบบการหาภาพรถเคลื่อนที่ (Background Subtraction) การแก้ไขมุมมองภาพแบบ Perspective สำหรับใช้ในระบบประมวลผลภาพการหาปริมาณรถ

1.5.4 ทำการสร้างและพัฒนาวงจรสำหรับการตรวจนับปริมาณรถบนเทคโนโลยี FPGA

1.5.5 ทดสอบขั้นตอนการประมวลผลภาพ และแก้ไขเพื่อให้สามารถทำงานได้ถูกต้องตรงตามฟังก์ชันที่ต้องการ

1.5.6 ทำการทดสอบเพื่อวิเคราะห์ และเก็บผล

1.5.7 จัดทำวิทยานิพนธ์

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 ได้วงจรตรวจนับปริมาณรถบนถนนด้วยการประมวลผลภาพจากกล้องวิดีโอบน FPGA ซึ่งเป็นระบบสมองกลฝังตัวสามารถตรวจนับปริมาณรถในแต่ละแยกได้อย่างรวดเร็ว

1.6.2 ได้วงจรตรวจนับปริมาณรถที่สามารถนำไปประยุกต์ใช้ในการจัดการระบบจราจรอย่างอัตโนมัติและปรับตัวได้ สามารถ “ลดไฟเขียวที่ไร้ประโยชน์” ณ บริเวณที่เกิดปัญหา

ด้านการจราจรและความหนาแน่น ส่งผลให้ลดระยะเวลาในการเดินทาง ประหยัดพลังงานและลดมลภาวะสิ่งแวดล้อม

บทที่ 2

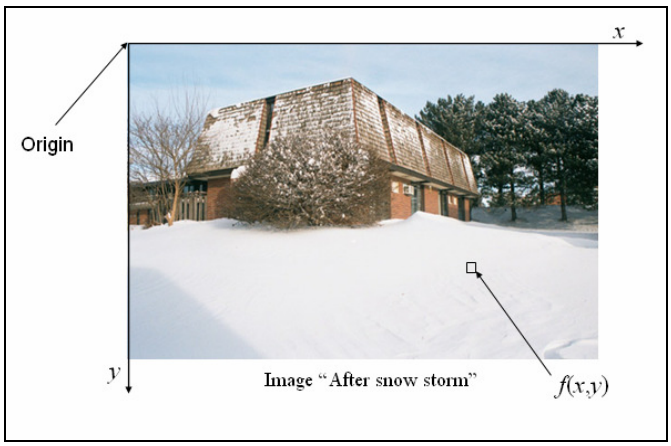
การวิเคราะห์ภาพวิดีโอ

ในบทนี้อธิบายถึงการวิเคราะห์ภาพวิดีโอ ที่แนะนำการประมวลผลภาพวิดีโอโดยทั่วไป อธิบายการได้มาของภาพดิจิตอลตามโมเดลกล้อง ความรู้พื้นฐานของภาพดิจิตอล ชนิดภาพเคลื่อนไหวของกล้องวงจรปิด จากนั้นอธิบายการได้มาของภาพนิ่งที่มาจากอุปกรณ์รับภาพมาแปลงเป็นรูปแบบของคอมพิวเตอร์ ซึ่งเป็นการแปลงภาพในพิกัดสามมิติเป็นภาพในพิกัดสองมิติแบบ Perspective โดยใช้โมเดลแบบกล้องรูเข็ม เพื่อนำภาพสองมิติแบบ Perspective ที่เป็นลำดับภาพวิดีโอที่ได้ มาเข้าสู่กระบวนการนับปริมาณรถที่ต้องตรวจหาการเคลื่อนที่ของวัตถุโดยใช้เทคนิคการลบภาพพื้นหลัง จึงอธิบายพื้นฐานของเทคนิควิธีการลบภาพพื้นหลัง ได้แก่ วิธีการหาผลต่างของเฟรม วิธี Running Average วิธี Running Gaussian วิธี Mixture of Gaussians และวิธี Eigenbackgrounds การลดสัญญาณรบกวนด้วยวิธี Morphological ด้วยโอเปอร์เรชันพื้นฐาน ได้แก่ Dilation และ Erosion

2.1 แนะนำการประมวลผลภาพวิดีโอ (Video Processing)

การประมวลผลภาพ คือ การวิเคราะห์สารสนเทศของภาพโดยใช้คอมพิวเตอร์ในการประมวลผล โดยวิธีการในการประมวลผลขึ้นอยู่กับผลลัพธ์ที่ต้องการ เช่น การแปลงภาพ (Image Transformation) การนิยามภาพ (Image Description) การกรองภาพ (Image Filters) การค้นคืนภาพ (Image Restoration) การปรับปรุงคุณภาพของภาพ (Image Enhancement) การแบ่งภาพและการหาขอบวัตถุในภาพ (Image Segmentation and Edge Detection) ระบบสี RGB (RGB Color) การลบพื้นหลัง (Background Subtraction) ฮิสโตแกรม (Histogram)และ การบีบอัดข้อมูลภาพ (Image Compression) เป็นต้น

ภาพเชิงดิจิตอล คือ ฟังก์ชัน 2 มิติ f(x, y) ของความเข้มของแสง (Intensity) โดยที่ x และ y คือค่าแสดงตำแหน่งในระบบพิกัดฉาก และค่าของฟังก์ชัน f ณ ตำแหน่ง (x, y) ใด ๆ จะเป็นสัดส่วนกับความสว่างของแสง ณ ตำแหน่งนั้น ดังภาพประกอบ 2-1



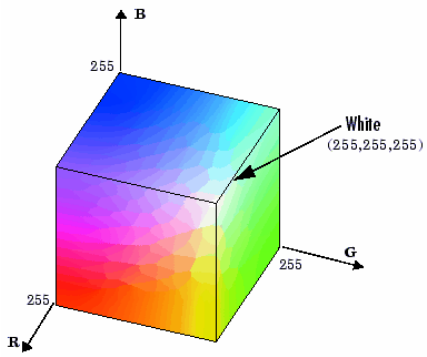
ภาพประกอบ 2-1 ภาพเชิงดิจิตอล [3]

พื้นที่เล็กๆจุดหนึ่งในภาพ โดยในแต่ละจุดนั้นจะมีค่าตัวเลขกำกับ ซึ่งตัวเลขเหล่านี้จะมาจากค่าของแม่สีสามสี R (สีแดง) G (สีเขียว) B (สีฟ้า) ใช้บอกระดับความเข้มของแต่ละเฉดสี หากมี Pixel หลายๆจุดมาต่อกันจะกลายเป็นภาพซึ่งมีขนาด จำนวนจุดภาพด้านกว้างคูณจำนวนจุดภาพด้านยาว ยกตัวอย่าง เช่นรูปภาพขนาด 800 x 600 pixels หมายความว่า รูปภาพนี้มีความกว้าง 800 pixels และมีความยาว 600 pixels เป็นต้น

สี คือ ลักษณะความเข้มของแสงที่ปรากฏแก่สายตาให้เห็นเป็นสี โดยผ่านกระบวนการรับรู้ด้วยตา โดยที่ตาได้ผ่านกระบวนการวิเคราะห์ข้อมูลพลังงานแสงมาแล้ว ผ่านประสาทสัมผัสการมองเห็น ผ่านศูนย์สับเปลี่ยนในสมองไปสู่ศูนย์การเห็นภาพ การสร้างภาพหรือการมองเห็นก็คือการที่ข้อมูลได้ผ่านการวิเคราะห์ แยกแยะให้เรารับรู้ถึงสรรพสิ่งรอบตัว โดยการมองเห็นภาพจากคอมพิวเตอร์นั้นเป็นการประกอบขึ้นของจุดภาพ ที่มีค่าของสีอยู่ในจุดภาพนั้น จำนวนสีสูงสุดที่เป็นไปได้ของแต่ละจุดภาพขึ้นอยู่กับจำนวนบิตที่ใช้เมื่อมีการกำหนดให้ขนาดของบิตต่อจุดมากขึ้นจะทำให้จำนวนของสีมากขึ้นด้วย ตัวอย่างเช่น 8 บิตจะแทนได้ 256 สี (2^8) เป็นต้น

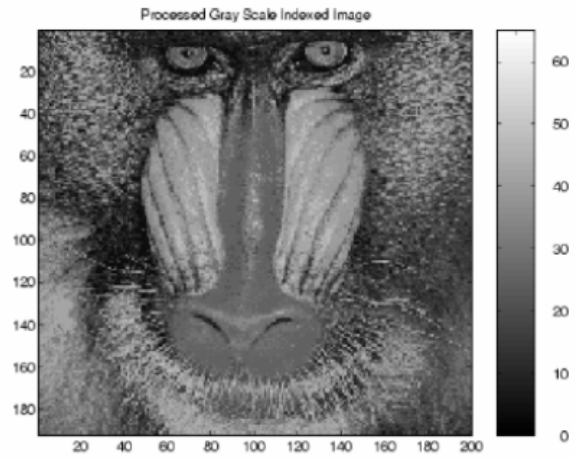
มาตรฐานของสีที่ใช้อยู่ในปัจจุบันมีอยู่หลายระบบด้วยกัน ทั้งนี้จะขึ้นอยู่กับการนำไปใช้ แต่โดยทั่วไปแล้วทุกมาตรฐานจะมีแนวคิดเดียวกันคือ การแทนจุดสีด้วยจุดที่อยู่ภายในพิกัด 3 มิติ โดยจะมีแกนอ้างอิงสำหรับจุดสีนั้นในระนาบซึ่งแต่ละแกนจะมีความเป็นอิสระต่อกัน

ระบบสี RGB นั้นย่อมาจาก RED , GREEN , BLUE เป็นระบบสีของแสง ซึ่งเกิดจากการหักเหของแสงผ่านแท่งแก้วปริซึมจะเกิดแถบสี และระบบแสงสีมีพื้นฐานจากหลักการของการมองเห็น เป็นระบบสีที่ใช้ในงานแสดงผลด้านโทรทัศน์ ซึ่งสีต่างๆเกิดจากการผสมผสานระหว่าง 3 แม่สีในอัตราส่วนที่ต่างกันไป จะทำให้เกิดสีต่างๆได้อีกมากมาย โดยอัตราส่วนของสี R:G:B จะมีค่าตั้งแต่ 0-225 งานระบบสี RGB ยังมีการสร้างมาตรฐานที่แตกต่างกันออกไปที่นิยมใช้งานได้แต่ RGBCIE และ RGBNTSC [16] ดังภาพประกอบที่ 2-2



ภาพประกอบ 2-2 แสดงโมเดลในระบบพิกัด Color Space [19]

ระบบสีเทา[11] เป็นค่าซึ่งระบุความสว่างหรือความเข้ม ซึ่งมีค่าตั้งแต่ 0-255 (0 คือระดับเข้ม และ 255 คือระดับสว่าง) รวมทั้งพิกัดแนวนอนและแนวตั้ง ซึ่งใช้ระบุตำแหน่งในแถวลำดับภาพ (Image Array) ดังตัวอย่างภาพประกอบ 2-3



ภาพประกอบ 2-3 ตัวอย่างภาพโทนสีขาวดำแสดงค่าระดับความเข้มแสง [10]

การแปลงภาพสีให้เป็นภาพขาว-ดำ (Thresholding) [17]เป็นกระบวนการแปลงภาพสีให้มีการ แสดงผลได้แค่ 2 ระดับ คือ ขาว และดำ โดยจะแปลงข้อมูลภาพให้เป็นภาพขาวดำ (Binary Image) มีกระบวนการแปลงภาพที่มีความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ หรือ 1 บิต (bit) คือ 0 และ 1 โดย 0 แทนด้วยจุดที่มีภาพสีขาว และ 1 แทนด้วยจุดที่มีภาพสี ดำ Thresholding Technique คือการพิจารณาจุดภาพในภาพว่าจุดใดควรจะเป็นจุดขาว หรือจุดใด ควรจะเป็นจุดที่มีค่าเท่ากับ 1 (จุดดำ) โดยจะทำการเปรียบเทียบค่าของแต่ละจุดภาพ (f(x,y)) กับค่าคงที่ ที่เรียกว่า Threshold (Threshold Value) เทคนิคนี้นิยมใช้กันมากในกรณีที่ความ

แตกต่างระหว่าง วัตถุ (Object) และพื้นหลัง (Background) ค่าจุดภาพในภาพที่มีค่าน้อยกว่าค่า Threshold จะถูก กำหนดเป็น 1 (จุดดำ) และถ้าค่าของจุดภาพใด ๆ ในภาพมีค่ามากกว่าหรือเท่ากับค่า Threshold จะถูก กำหนดให้เป็น 0 (จุดขาว) ในการทำภาพขาวดำ โดยการทำ Thresholding ให้ได้ภาพดีและคมชัด ต้องเกิดจากการเลือกค่า Threshold ที่ถูกต้องและเหมาะสม ถ้าเลือกค่า Threshold ไม่เหมาะสม เช่น ค่า Threshold ที่มากหรือน้อยจนเกินไป ภาพที่ได้จะขาดความคมชัด หรืออาจทำให้รายละเอียดของ ภาพขาดหายไป หรือภาพที่ได้อาจจะมืดเกินไป หรือสว่างเกินไป หรืออาจจะเป็นภาพที่มีสิ่งรบกวน (Noise) เกิดขึ้น ทำให้ภาพผลลัพธ์ที่ได้ไม่ชัดเจน

2.2 การได้มาของภาพ (Image Acquisition)

2.2.1 การได้มาของภาพนิ่ง

การได้มาของภาพแต่ละภาพจะแตกต่างกันขึ้นอยู่กับอุปกรณ์การรับภาพ วิธีการที่ใช้ในการรับภาพนอกจากนี้ยังขึ้นอยู่กับการนำเอาภาพนั้นๆไปใช้งานเมื่อจะนำภาพต่าง ๆ เข้ามาสู่คอมพิวเตอร์จำเป็นที่จะทำให้คอมพิวเตอร์รู้จักโดยการแปลงรูปแบบให้ตรงกับระบบคอมพิวเตอร์ต้องการ

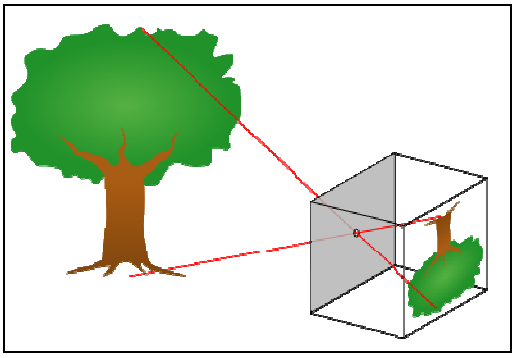
การสร้างภาพจากคอมพิวเตอร์นั้นจะทำให้เกิดภาพ ที่แตกต่างกันออกไปจากที่เกิดกับตามนุษย์และกล้องถ่ายภาพ เนื่องจากการสร้างภาพด้วยคอมพิวเตอร์นั้นเป็นการสร้างภาพจากวัตถุที่ไม่มีอยู่จริงและกล้องที่ใช้บันทึกภาพก็ไม่มีอยู่จริงเช่นเดียวกัน

การที่จะสร้างภาพจากคอมพิวเตอร์ที่เป็น Virtual World ซึ่งเป็นโลกที่เราทำการสมมุติขึ้นเพื่อจำลอง Actual World หรือโลกที่มีอยู่จริงโดยอาศัยหลักการของธรรมชาติดังกล่าวนั้นเป็นเรื่องที่ยากหรือแทบจะเป็นไปไม่ได้สำหรับบางปรากฏการณ์จึงได้มีการพัฒนาวิธีอื่นๆที่ดีกว่าและมีความเหมาะสมกับประสิทธิภาพของเครื่องมากกว่า โดยวิธีการต่างๆซึ่งจะให้ผลของภาพที่แตกต่างกันออกไป

Virtual World นี้ถูกพัฒนาขึ้นเพื่อที่จะช่วยให้เราทำการสร้าง ดัดแปลง แก้ไข และการดำเนินการอื่น ๆ กับข้อมูลหรือพารามิเตอร์ในระดับต่ำได้ง่ายขึ้น ซึ่งการมองเห็นนั้นเริ่มจากแหล่งกำเนิดแสง วัตถุ และสิ้นสุดที่อุปกรณ์รับภาพ ดังนั้น ส่วนประกอบหลักของ Virtual world นี้จึงได้แก่ Virtual Camera ,Virtual Light Source และVirtual Object

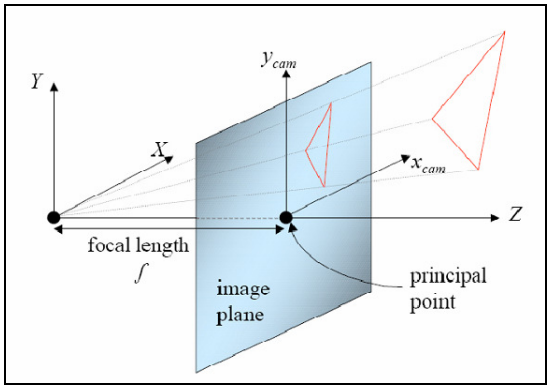
โมเดลกล้องรูเข็ม [2] เป็น Virtual Camera เป็นสิ่งที่แทนกล้องหรือตามนุษย์ ที่มีช่องรับภาพ ไม่มีเลนส์ เพื่อทำการจัดระเบียบแสง ไม่มีความยาวโฟกัส ไม่มีระยะโฟกัสทำให้เกิดภาพของวัตถุที่คมชัดไม่ว่าจะอยู่ในตำแหน่งใด ไม่มีความไวชัตเตอร์จึงทำให้เกิดภาพของวัตถุที่

หยุดนิ่งไม่ว่าจะกำลังเคลื่อนที่ด้วยความเร็วสูงเท่าใด ข้อดีคือจะไม่มีการเกิดภาพหลุดโฟกัสขึ้น หลักการทำงานไม่มีความซับซ้อนเพราะแสงเดินทางเป็นเส้นตรง รังสีส่วนบนของภาพกลางแจ้งสามารถผ่านเข้าไปสัมผัสกับฉากรับภาพในส่วนล่างผ่านทางช่องรับภาพซึ่งก่อให้เกิดภาพกลับหัวจากแสงสะท้อน ดังภาพประกอบ 2-4



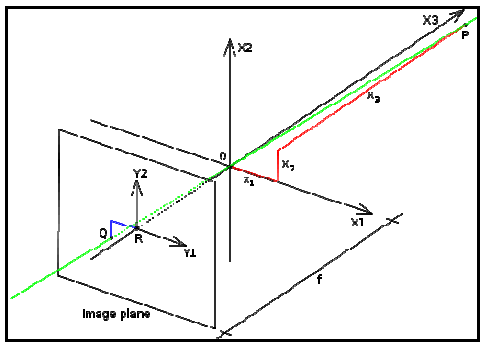
ภาพประกอบ 2-4 หลักการทำงานของกล้องรูเข็ม [2]

ดังนั้น สามารถใช้ค่าประมาณของค่าอนุพันธ์อันดับที่หนึ่ง(first order) ของโมเดลกล้องรูเข็มในการแปลงภาพในฉากสามมิติ เป็นภาพสองมิติได้ ดังภาพประกอบ 2-5



ภาพประกอบ 2-5 โมเดลกล้องรูเข็มในสามมิติ [12]

รูปเรขาคณิตของกล้องรูเข็ม ที่เกี่ยวข้องในการแปลงภาพในฉากสามมิติเป็นภาพสองมิติ ดังรูปประกอบ 2-6

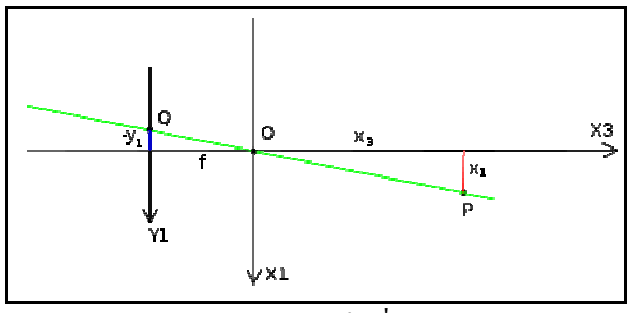


ภาพประกอบ 2-6 รูปเรขาคณิตของกล้องรูเข็ม [12]

จากภาพประกอบ 2-6 สามารถอธิบายได้ดังนี้

- ระบบพิกัดฉากสามมิติมีจุดกำเนิดที่จุด O ซึ่งเป็นตำแหน่งช่องรับภาพของกล้อง ซึ่งระบบพิกัดฉากประกอบด้วยสามแกน คือ X_1, X_2 และ X_3 สามารถอธิบายได้ว่า แกน X_3 ชี้ไปยังทิศทางการมองของกล้อง หมายถึง แกนเชิงแสง (optical axis) แกนมุขสำคัญ (principal axis) หรือ รังสีมุขสำคัญ (principal ray) ที่มีแกน X_1 และ X_2 ตัดกันที่หน้ากล้อง หรือหน้าระนาบของแกนมุขสำคัญ
- ระนาบของภาพสามมิติของโลกจะถูกฉายผ่านช่องรับภาพของกล้องและขนานไปกับแกน X_1 และ X_2 ซึ่งแทนด้วยระยะ f ที่มีระยะมาจากจุดกำเนิดในทิศของแกน X_3 ในด้านแกนลบ ทำให้ระนาบของภาพตัดกับแกน X_3 ที่พิกัด $-f$ เมื่อ f มีค่ามากกว่า 0 จะได้เป็นจุดรวมของแสงจากกล้องรูเข็ม
- ที่จุด R เป็นจุดตัดของแกนเชิงแสงและระนาบของภาพ ซึ่งเป็นจุดมุขสำคัญ (principal point) หรือจุดศูนย์กลางภาพ
- จุด P เป็นพิกัดของโลกที่จุด (x_1,x_2,x_3) ที่เกี่ยวข้องกับแกน X_1, X_2, X_3
- เส้นการฉายภาพ(projection line) ของจุด P ไปยังกล้อง คือ เส้นที่ลากผ่านจุด P และจุด O
- การฉายภาพ(projection) ของจุด P ไปยังระนาบของภาพ ซึ่งหมายถึง Q ซึ่งได้มาจากเส้นการฉายภาพและระนาบของภาพ
- นอกจากนี้ยังมีพิกัดของระบบสองมิติอยู่ด้วยในระนาบของภาพ ที่มีจุดกำเนิดเป็น R และมีแกน Y_1 และ Y_2 ที่ขนานไปกับ X_1 และ X_2 ตามลำดับ ซึ่งพิกัดของจุด Q จะเกี่ยวข้องกับพิกัด(y_1,y_2) ของระบบ

จากนั้นจะต้องหาความสัมพันธ์ของจุด Q ที่พิกัด (y_1,y_2) ที่ขึ้นอยู่กับจุด P ที่พิกัด (x_1, x_2, x_3) ดังภาพประกอบ 2-7 แสดงฉากที่มองจากแนวแกน X_2 ด้านลบ



ภาพประกอบ 2-7 ฉากกล้องรูเข็มที่มองจากแกน X2 [12]

จากภาพประกอบ 2-7 จะเห็นได้ว่าเส้นการฉายภาพ จะมีลักษณะเป็นรูปสามเหลี่ยมสองรูป ซึ่งความกว้างของสามเหลี่ยมด้านซ้ายเป็น $-y_1$ และ ความกว้างของสามเหลี่ยมด้านขวาเป็น x_1 และ x_3 สามารถเขียนเป็นสมการ (2-1) หรือ(2-2)

$$\frac{-y_1}{f} = \frac{x_1}{x_3} \tag{2-1}$$

หรือ
$$y_1 = -\frac{f\,x_1}{x_3} \tag{2-2}$$

จากแกนลบของ X_1 จะได้เป็นสมการ (2-3) หรือ (2-4)

$$\frac{-y_2}{f} = \frac{x_2}{x_3} \tag{2-3}$$

หรือ
$$y_2 = -\frac{f\,x_2}{x_3} \tag{2-4}$$

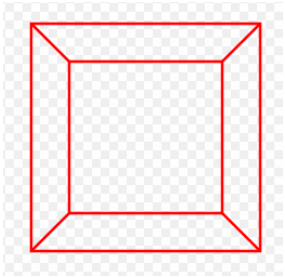
สามารถนำสมการ (2-3) และ (2-4) มารวมกันได้เป็นสมการ (2-5)

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{f}{x_3}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{2-5}$$

จากสมการที่ (2-33) เป็นความสัมพันธ์ระหว่างพิกัด (x_1, x_2, x_3) ในระนาบสามมิติที่จุด P และภาพที่พิกัด (y_1, y_2) จะได้เป็นจุด Q ในระนาบของภาพ

มุมมองภาพแบบ Perspective เป็นภาพที่ปรากฏออกมาในลักษณะที่เหมือนการมองเห็นจริง บนพื้นผิวที่แบนราบ ซึ่งประกอบด้วย คุณสมบัติ คือ วัตถุที่อยู่ด้านหน้ากล้องจะมีขนาดใหญ่ และจะมีขนาดเล็กลงเรื่อยๆเมื่อภาพวัตถุมีระยะทางไกลจากกล้องออกไป

ภาพ Perspective แบบจุดเดียว เป็นภาพที่มีจุดลับตา หรือจุดอันตธาน (Vanishing point) คือ จุดรวมของเส้นฉายของภาพมีจุดเดียว อาจอยู่ด้านซ้าย หรือขวา บนหรือล่าง หรืออยู่กึ่งกลางของภาพก็ได้ ดังภาพประกอบ 2-8



ภาพประกอบ 2-8 ภาพ Perspective แบบจุดเดียว [2]

จากภาพประกอบ 2-8 ด้านหน้าของภาพแบบ Perspective จะตรงกับมุมมองของผู้มองวัตถุ ซึ่งวัตถุต่างๆจะเป็นเส้นตรง ซึ่งตรงไปในแนวขนานกับแนวสายตาของผู้มอง หรือเป็นแนวเดียวกับเส้นตั้งฉาก

การแปลงภาพที่มีพิกัดอยู่ในระนาบสามมิติให้เป็นภาพระนาบสองมิติหรือระนาบของภาพแบบ Perspective [12] มีขั้นตอนดังนี้

ขั้นตอน 1) การแปลงภาพวัตถุจากระบบพิกัดของโลก (x_w, y_w, z_w) เป็นระบบพิกัดสามมิติ ของกล้อง (x, y, z) สามารถทำได้ ดังสมการ (2-6)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \qquad (2\text{-}6)$$

เมื่อ R คือ 3x3 เมตริกซ์ rotation ดังสมการ (2-7)

$$R = \begin{bmatrix} r_1 r_2 r_3 \\ r_4 r_5 r_6 \\ r_7 r_8 r_9 \end{bmatrix} \qquad (2\text{-}7)$$

และ T คือ เวกเตอร์ของการแปลงภาพ ดังสมการ (2-8)

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{2-8}$$

ตัวแปรที่ถูกคาลิเบรท คือ R และ T ซึ่งการเปลี่ยนพิกัดจากระบบพิกัดคาร์ทีเซียน (x_w, y_w, z_w) เป็นระบบพิกัด (x, y, z) ทำได้โดยเทคนิคการคาลิเบรทกล้องโดยใช้การหมุน

ขั้นตอน 2) แปลงพิกัดสามมิติของกล้อง (x, y, z) เป็นพิกัดของภาพ(ideal) (X_u, Y_u) โดยใช้การฉายภาพแบบ Perspective ของโมเดลกล้องรูเข็ม ดังสมการ (2-9) และ(2-10)

$$X_u = f\frac{x}{z} \tag{2-9}$$

$$Y_u = f\frac{y}{z} \tag{2-10}$$

เมื่อ f คือ ค่า focal length ที่ถูกคาลิเบรท

ขั้นตอน 3) Radial lens distortion ตามสมการ (2-11) และ (2-12)

$$X_d + D_x = X_u \tag{2-11}$$

$$Y_d + D_x = Y_u \tag{2-12}$$

เมื่อ (X_d, Y_d) เป็นพิกัดภาพจริงของระนาบของภาพ ซึ่งค่า D_x D_y และ r มีค่าตามสมการ (2-13) (2-14) และ(2-15)

$$D_x = X_d(K_1 r^2 + k_2 r^4 + \ldots) \tag{2-13}$$

$$D_y = Y_d(K_1 r^2 + k_2 r^4 + \ldots) \tag{2-14}$$

และ $$r = \sqrt{X_d^2 + Y_d^2} \tag{2-15}$$

ตัวแปรที่ถูกคาลิเบรท คือ K_i

ขั้นตอน 4) พิกัดของภาพจริง (X_d, Y_d) เป็นพิกัดของเฟรมภาพ (X_f, Y_f) หาค่าของ X_f และ Y_f ได้จากสมการ (2-16) และ (2-17)

$$X_f = s_x d_x^{'-1} X_d + C_x \tag{2-16}$$

$$Y_f = d_y^{-1} Y_d + C_y \tag{2-17}$$

เมื่อ (X_f, Y_f) คือ ค่าจุดสีที่ตำแหน่งแถวและคอลัมน์ของเฟรมภาพ

(C_x, C_y) คือ จุดศูนย์กลางที่ตำแหน่งแถวและคอลัมน์ของเฟรมภาพ

2.2.2 ภาพเคลื่อนไหวจากกล้องวงจรปิด

ระบบกล้องวงจรปิด คือ ระบบการบันทึกภาพเคลื่อนไหว เพื่อการรักษาความปลอดภัย หรือใช้เพื่อการสอดส่องดูแลเหตุการณ์หรือสถานะการณ์ต่างๆ ที่นอกเหนือจากการรักษาความปลอดภัย มีหลักการทำงาน คือ ตัวกล้องที่เป็นตัวรับสัญญาณภาพ จะรับภาพได้นั้นจะต้องมีแสงส่องไปยังที่วัตถุที่ต้องการและแสงนั้นจะตกกระทบ วัตถุ แล้วจึงสะท้อนกลับออกมา (ประสิทธิภาพกล้องนั้น ขึ้นอยู่กับความไวแสง ซึ่งจะส่งผลให้คุณภาพของการทำงานแตกต่างกันออกไป)

บันทึกภาพส่วนมากเป็นสัญญาณในระบบ RGB ซึ่งถ้านำอุปกรณ์ดังกล่าวมาต่อกับคอมพิวเตอร์จำเป็นต้องใช้อุปกรณ์เพิ่มเติมในการแปลงภาพดังกล่าวเข้าสู่คอมพิวเตอร์โดยที่อุปกรณ์นั้นจะทำหน้าที่รับภาพเป็นเฟรม

ตัวอย่างภาพเคลื่อนไหวที่ได้จากกล้องวงจรปิด คือ MPEG4 The Moving Pictures Expert Group Layer 4 อยู่ในตระกูลของ MPEG (Moving Pictures Expert Group) คือ ไฟล์วีดีโอประเภทหนึ่งที่มีการบีบอัดข้อมูลวีดีโอหรือรูปภาพให้มีขนาดเหมาะสมต่อการส่งผ่านไปยังการสื่อสารต่างๆทุกประเภท โดยการบีบอัดใช้หลักการเข้ารหัสกราฟฟิกและวีดีโอในแบบอัลกอริทึมที่ได้รับ การพัฒนามาจาก MPEG 1 MPEG 2 โดยไฟล์ที่ได้รับการบีบอัดในรูปแบบ Wavelet - based MPEG 4 จะมีขนาดเล็กกว่า JPEG ซึ่ง เป็นผลมาจากการลดขนาดช่วงกว้างของแบนวิท และใช้หลักการบีบอีด ใช้หลักการบีบอัดเป็นเฟรมๆ แทนที่จะบันทึกภาพในทุกเฟรมซึ่งต้องใช่เนื้อที่ในการเก็บมาก

2.3 การตรวจหาการเคลื่อนที่ของวัตถุจากลำดับภาพวิดีโอ (Motion Detection)

มีหลากหลายวิธีในการตรวจหาการเคลื่อนที่ของวัตถุของลำดับภาพวิดีโอที่อยู่ในตำแหน่งคงที่ในหลายแอพพลิเคชัน เช่น ระบบควบคุมการจราจร ระบบรักษาความปลอดภัย เป็นต้น ซึ่งวิธีทั้งหลายเหล่านี้มีแนวคิดพื้นฐาน คือ การเปรียบเทียบเฟรมภาพปัจจุบัน ด้วยเฟรมภาพก่อนหน้าหรือที่เรียกว่าภาพพื้นหลังหรือโมเดลภาพพื้นหลัง ซึ่งภาพพื้นหลังหมายถึง ภาพที่ไม่มีวัตถุที่เคลื่อนที่ ส่วนโมเดลภาพพื้นหลังจะมีการเก็บรายละเอียดการเปลี่ยนแปลงของภาพพื้นหลัง ซึ่งเป็นกระบวนการซับซ้อนมากยิ่งขึ้นอาจจะใช้ทรัพยากรหน่วยความจำเพิ่มขึ้นในการประมวลผล แต่ก็ทำให้ผลลัพธ์ที่ได้มีความถูกต้องมากขึ้นเช่นกัน ซึ่งเทคนิคการตรวจหาการเคลื่อนที่ของวัตถุโดยทั่วไป [5] มีดังนี้

2.3.1 การหาผลต่างของเฟรม (Frame Difference)

การแยกความแตกต่างของจุดสี ระหว่างรถที่เคลื่อนที่และภาพพื้นหลังวิธีนี้จะใช้ภาพพื้นหลังที่ประมาณค่าได้เป็นเฟรมก่อนหน้าเพื่อง่ายต่อการประมวลผล สำหรับขั้นตอนการประมวลผลจำเป็นจะต้องทำให้ภาพเป็นขาวดำ เพื่อลดหน่วยความจำที่ใช้ ซึ่งใช้ 8 bit เหมาะกับแอพพลิเคชันที่ต้องการความเร็วในการประมวลผล ใช้ทรัพยากรหน่วยความจำน้อย และกล้องวิดีโออยู่ในตำแหน่งที่คงที่ตามสมการที่ (2-18)

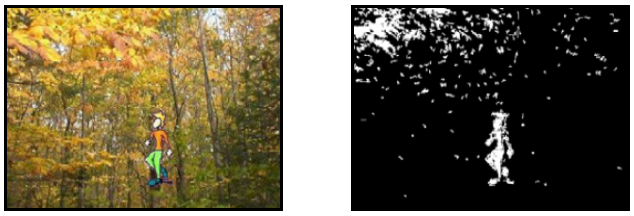
$$\left| frame_i - frame_{i-1} \right| > T \tag{2-18}$$

เมื่อ $frame_i$ คือ เฟรมปัจจุบัน

$frame_{i-1}$ คือ เฟรมก่อนหน้า และ

T คือ ค่า threshold

ข้อเสียของวิธีการหาผลต่างของเฟรม [13] ในกรณีที่ภาพพื้นหลังมีความซับซ้อนไม่อยู่นิ่ง จะทำให้เกิดสัญญาณภาพรบกวนจำนวนมากในส่วนของภาพพื้นหลัง ดังภาพประกอบ 2-10 ที่เป็นภาพวิดีโอตัดต่อ ภาพพื้นหลังเป็นต้นไม้ที่มีใบไม้เคลื่อนที่ตลอดเวลา ดังภาพประกอบ 2-9



(ก) (ข)

ภาพประกอบ 2-9 ผลจากการลบภาพพื้นหลังด้วยวิธีหาผลต่างของเฟรม [13]
(ก) ภาพวิดีโอตัดต่อเฟรมปัจจุบัน (ข) ภาพวัตถุเคลื่อนที่

2.3.2 วิธี Running Average

เทคนิคนี้เป็นวิธีการหาโมเดลภาพพื้นหลังอย่างง่าย ไม่มีความซับซ้อน ที่ไม่จำเป็นต้องใช้ทรัพยากรหน่วยความจำ ประมวลผลได้รวดเร็ว เนื่องจากการคำนวณของภาพพื้นหลังในเฟรมปัจจุบัน สามารถให้ค่าของการหาภาพพื้นหลังในเฟรมถัดไปได้ ตามสมการ (2-19)

$$B_{i+1} = \alpha * F_i + (1-\alpha) * B_i \tag{2-19}$$

เมื่อ B_{i+1} คือ ภาพพื้นหลังของเฟรมถัดไป

B_i คือ ภาพพื้นหลังของเฟรมปัจจุบัน

α คือ ค่าสัมประสิทธิ์ โดยทั่วไปมีค่า 0.5 และ

F_i คือ ภาพวัตถุที่เคลื่อนที่

ตัวอย่างการใช้การลบภาพพื้นหลังด้วยวิธี Running Average ดังภาพประกอบ (2-10) ซึ่งภาพดังภาพประกอบ (2-10) ก เป็นภาพเฟรมปัจจุบันจากการจราจร ภาพประกอบ (2-10) ข ภาพพื้นหลังปัจจุบัน จากนั้นหาผลต่างของเฟรมปัจจุบันและภาพพื้นหลังปัจจุบันได้เป็น object mask ดังภาพประกอบ (2-10) ค แล้วใช้ Object Mask กับภาพเฟรมปัจจุบัน motion mask image ดังภาพประกอบ (2-10) ง และใช้ Object Mask กับภาพพื้นหลังปัจจุบันได้เป็น motion mask background ดังภาพประกอบ (2-10) จ เพื่อหาภาพที่นำไปเปลี่ยนภาพพื้นหลังให้เป็นปัจจุบันโดยนำ motion mask image และ motion mask background มาบวกกัน ดังภาพประกอบ (2-10) ฉ

(ก) (ข) (ค)

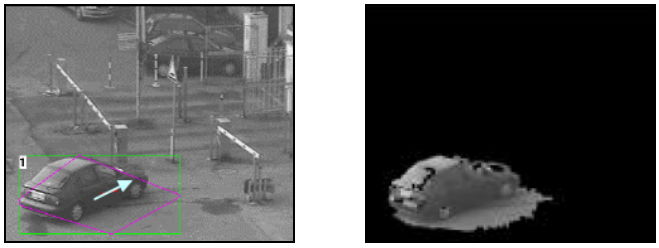
(ง) (จ) (ฉ)

ภาพประกอบ 2-10 ผลจากการลบภาพพื้นหลังด้วยวิธี Running Average [2] (ก) ภาพเฟรมปัจจุบัน (ข) ภาพพื้นหลังปัจจุบัน (ค) object mask (ง) motion mask image (จ) motion mask background (ฉ) ภาพที่นำไปเปลี่ยนภาพพื้นหลังให้เป็นปัจจุบัน

จากภาพประกอบ 2-10 จะเห็นได้ว่าแต่ละเฟรมที่เข้ามาไม่มีการแยกจุดสีว่าเป็นภาพพื้นหลัง หรือวัตถุเคลื่อนที่ ไม่มีการสร้างเป็นโมเดลภาพพื้นหลัง มีข้อเสียในกรณีที่แสงมีการเปลี่ยนแปลงไม่คงที่ทำให้ภาพพื้นหลังเกิดเป็นสัญญาณรบกวนได้ เนื่องจากภาพพื้นหลังของแต่ละเฟรมจะต้องเปลี่ยนให้เป็นปัจจุบันทุกเฟรม

การใช้เทคนิค Selectivity [14] เป็นการสร้างโมเดลของภาพพื้นหลังจากลำดับเฟรมภาพวิดีโอ โดยเฟรมภาพที่เข้ามาใหม่ จะทำการพิจารณาจุดสีของเฟรมภาพนั้นๆว่าเป็นภาพพื้นหลัง หรือเป็นวัตถุเคลื่อนที่ ซึ่งถ้าจุดสีใดถูกพิจารณาว่าเป็นวัตถุเคลื่อนที่แล้ว จะไม่นำจุดสีนั้นมาคิดเป็นโมเดลของภาพพื้นหลัง ซึ่งจุดสีนั้นจะถูกตัดทิ้งในโมเดลภาพพื้นหลัง ซึ่งสมการสำหรับการคำนวณ ดังสมการที่ (2-20) และตัวอย่างการการใช้เทคนิค Selectivity ดังภาพประกอบ 2-11

$$B_{t+1}(x,y)=\begin{cases}\alpha * F_t(x,y)+(1-\beta)*B_t(x,y) & \text{เมื่อ } F_t(x,y) \text{ เป็นจุดสีของภาพพื้นหลัง}\\ B_t(x,y) & \text{เมื่อ } F_t(x,y) \text{ เป็นจุดสีวัตถุเคลื่อนที่}\end{cases} \quad (2\text{-}20)$$



(ก) (ข)

ภาพประกอบ 2-11 ผลจากการลบภาพพื้นหลังด้วยวิธี Selectivity [14] (ก) เฟรมปัจจุบัน (ข) ภาพวัตถุเคลื่อนที่

เทคนิคนี้ใช้ทรัพยากรหน่วยความจำน้อย ไม่มีความซับซ้อน แต่มีความถูกต้องแม่นยำน้อย

2.3.3 วิธี Running Gaussian

เทคนิคนี้เป็นการสร้างโมเดลของภาพพื้นหลังจากลำดับเฟรมภาพวิดีโอ ของตำแหน่งของแต่ละจุดสีที่เป็นอิสระต่อกัน ซึ่งเป็นโมเดลทางสถิติด้วยวิธีเกาส์เซียนดิสทริบิวชัน (μ,σ) ซึ่งให้ภาพพื้นหลังแบบ PDF (Probability Density Function) ของค่าจุดสีทั้งหมด n จุดสี [1] ซึ่งสมการสำหรับการคำนวณ ดังสมการที่ (2-21) (2-22) และ (2-23)

$$\mu_{t+1}=\alpha F_t+(1-\alpha)\mu_t \quad (2\text{-}21)$$

$$\sigma^2_{t+1}=\alpha(F_t-\mu_t)^2+(1-\alpha)\sigma^2_t \quad (2\text{-}22)$$

เมื่อ μ_t คือ ค่าเฉลี่ยของเฟรมปัจจุบัน

μ_{t+1} คือ ค่าเฉลี่ยของเฟรมถัดไป

σ คือ ค่าความแปรปรวนของเกาส์เซียน

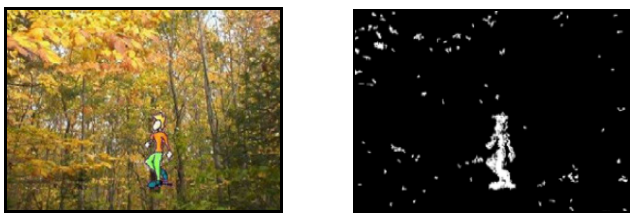
α คือ ค่าน้ำหนัก และมักจะเป็นค่าน้อยๆ เช่น 0.05

F คือ ค่าจุดสีของภาพอินพุตที่เข้ามาใหม่ในแต่ละเฟรม ที่เวลา t

ซึ่งค่า α ที่เลือกจะมีผลต่อการปรับค่าให้เป็นปัจจุบัน ของโมเดลให้เป็นไปอย่างช้าๆ หรืออย่างรวดเร็ว แต่ก็เป็นค่าที่ไม่เฉพาะเจาะจง และพารามิเตอร์อื่นของเกาส์เซียน PDF เช่นค่าเบี่ยงเบนมาตรฐานก็สามารถคำนวณได้คล้ายๆกัน ส่วนการเพิ่มความเร็ว เนื่องจากเทคนิค Running average ใช้ทรัพยากรหน่วยความจำน้อยมาก ซึ่งแต่ละจุดสีจะเก็บพารามิเตอร์สองตัว คือ (μ_t, σ_t) ในบัฟเฟอร์ ซึ่งค่าของจุดสีที่มีทั้งหมด n ค่า และแต่ละเฟรมที่เข้ามาในเวลา t ค่าของ F_t จะถูกพิจารณาเป็นประเภทของวัตถุได้ก็ต่อเมื่อมีเงื่อนไข ตามสมการ (2-7)

$$| F_t - \mu_t | > k\sigma_t \tag{2-23}$$

ถ้าในกรณีที่นอกเหนือจากสมการที่ (2-22) นั้น F_t จะถูกจัดเป็นประเภทของภาพพื้นหลัง จากโมเดลภาพพื้นหลังตามสมการที่ (2-20) และ (2-21) นั้น ในระบบเวลาจริงนั้น อาจทำให้มีการประมวลผลที่ช้าลง ทำให้อัตราการปรับค่าให้เป็นปัจจุบันของค่า μ หรือ σ ช้ากว่าอัตราการเข้ามาของแต่ละเฟรมได้ อย่างไรก็ตาม ในความเป็นจริงแล้วเทคนิคนี้ก็ยังถือว่ามีการประมวลผลได้เร็วกับภาพพื้นหลังที่เป็นแบบไดนามิก ตัวอย่างการลบภาพพื้นหลังด้วยวิธี Running Gaussian [13] ดังภาพประกอบ 2-12



(ก) (ข)

ภาพประกอบ 2-12 ผลจากการลบภาพพื้นหลังด้วยวิธี Running Gaussian [14]
(ก) ภาพวิดีโอตัดต่อเฟรมปัจจุบัน (ข) ภาพวัตถุเคลื่อนที่

2.3.4 วิธี Mixture of Gaussians

เทคนิคนี้เป็นการสร้างโมเดลของภาพพื้นหลังที่สามารถปรับค่าให้เป็นปัจจุบันได้ตลอดเวลาให้เหมือนกับพื้นหลังของทุกๆเฟรมปัจจุบัน ที่มีวัตถุเคลื่อนที่เกิดขึ้นภายในเฟรมนั้นๆ แต่ละจุดสีของเฟรมภาพจะถูกแยกประเภทเป็นจุดสีของภาพพื้นหลัง หรือจุดสีของวัตถุด้วยวิธีเกาส์เซียนดิสทริบิวชันที่มากกว่า 1 ที่มีประสิทธิภาพในการหาโมเดลของภาพพื้นหลังที่มีความซับซ้อน

ในสภาพแวดล้อมจริงที่มีการเปลี่ยนแปลงที่ไม่แน่นอนของสภาพแวดล้อม เช่น ความเข้มของแสง เงาจากต้นไม้ หรือเงาจากอาคารต่างๆ ทำให้ค่าของจุดสีที่ตำแหน่งเดิมของแต่ละลำดับเฟรมภาพเปลี่ยนแปลงตามสภาพอากาศ และสภาพแวดล้อมได้ ทำให้การสร้างโมเดลของภาพพื้นหลังเพียงหนึ่งโมเดลอาจจะไม่เพียงพอกับภาพวิดีโอที่มีความซับซ้อน

จากงานวิจัยของ Stauffer และ Grimson[11] สร้างโมเดลภาพพื้นหลังที่มีหลายค่าเพื่อแก้ปัญหาการของจุดสีของภาพพื้นหลังที่มีหลายค่า ทำให้การแบ่งประเภทจุดสีถูกต้องมากยิ่งขึ้น ซึ่งความน่าจะเป็นของการพิจารณาค่าจุดสี สามารถอธิบายได้ดังสมการ (2-24)

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t - \mu_{i,t} \sum_{i,t}) \qquad (2\text{-}24)$$

เมื่อ K คือ จำนวนของเกาส์เซียนดิสทริบิวชัน

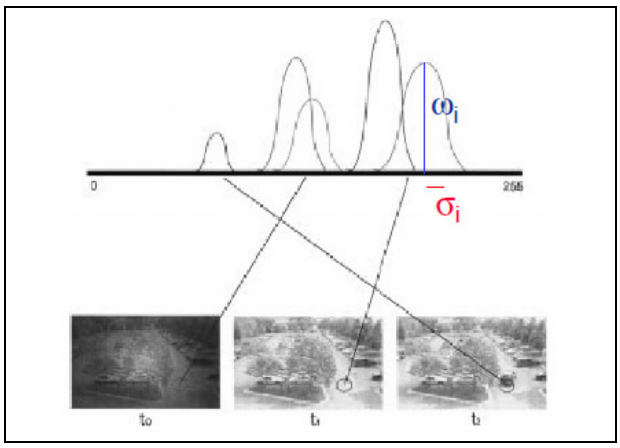
$\omega_{i,t}$ คือ ค่าน้ำหนักของเกาส์เซียนตัวที่ i^{th} ที่เวลา t

$\mu_{i,t}$ คือ ค่าเฉลี่ยของเกาส์เซียนตัวที่ i^{th} ที่เวลา t

$\sum_{i,t}$ คือ ค่าความแปรปรวนร่วมของเกาส์เซียนตัวที่ i^{th} ที่เวลา t

η คือ Gaussian Probability Density Function

เกณฑ์ในการแยกประเภทจุดสี ว่าเป็นจุดสีของภาพพื้นหลังหรือจุดสีของวัตถุเคลื่อนที่ หาได้จากเมื่อพิจารณาจุดสีค่าแอมพลิจูดสูงสุด จะมีค่าเป็น ω_i และมีค่าของส่วนเบี่ยงเบนมาตรฐานเป็น σ_i [15] ดังภาพประกอบ 2-13 ถ้าดิสทริบิวชันที่มีความหนาแน่นมากจะเป็นส่วนของภาพพื้นหลัง ดังสมการ(2-25)



ภาพประกอบ 2-13 การแยกประเภทจุดสี ของวิธี Mixture of Gaussians [15]

$$\sum_{i=1}^{B} \omega_i > t \tag{2-25}$$

เมื่อ T คือ Threshold

จากสมการ(2-25) ถ้าแอมพลิจูดของดิสทริบิวชันมีค่ามากกว่า Threshold ดิสทริบิวชันนั้นจะเป็นส่วนของภาพพื้นหลังและทำการปรับค่าพารามิเตอร์ $(\omega_{i,t}, \mu_{i,t}, \sigma_{i,t})$ ให้เป็นปัจจุบัน แต่ถ้านอกเหนือจากนี้ดิสทริบิวชันนั้นจะกลายเป็นส่วนของวัตถุที่เคลื่อนที่ในเฟรมภาพ ตัวอย่างผลจากวิธี Mixture of Gaussians ดังภาพประกอบ 2-14

(ก) (ข)

ภาพประกอบ 2-14 ผลจากวิธี Mixture of Gaussians [14]

(ก) ภาพวิดีโอตัดต่อเฟรมปัจจุบัน (ข) ภาพวัตถุเคลื่อนที่

2.3.5 วิธี Eigenbackgrounds

การลบภาพพื้นหลังโดยใช้ไอเกน (Eigen-background Subtraction) วิธีการนี้มิติของพื้นที่ที่สร้างจากภาพตัวอย่างจะถูกลดจำนวนมิติลง โดยการใช้ Principle Component Analysis (PCA) ดังนั้น หลังจากทำการลดมิติแล้ว ภาพจะเหลือแต่ส่วนที่ไม่เคลื่อนไหว ซึ่งสามารถนำมาลบ

ออกจากภาพเพื่อให้ได้มาซึ่งวัตถุที่เคลื่อนที่ได้ดังภาพประกอบที่ 2-15 วิธีการนี้สามารถสรุปขั้นตอนได้คือ

ขั้นตอน 1) เก็บตัวอย่างจำนวน N ภาพ จากนั้นคำนวณหาค่าเฉลี่ยของพื้นหลัง ทำการ Normalize และจัดให้อยู่ในเมตริกซ์แบบคอลัมน์ *(A)*

ขั้นตอน 2) คำนวณค่าความแปรปวน

ขั้นตอน 3) คำนวณหา eigen-vector

กำหนดให้ A เป็นค่าเมตริกซ์จัตุรัส

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

และ v เป็นเวกเตอร์หลัก (Column Vector) และ λ เป็นค่าคงที่ใดๆ โดยที่

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_3 \end{bmatrix}$$

ที่ทำให้ $$Av = \lambda v \qquad (2\text{-}26)$$

เมื่อ A คือ ค่าเมตทริกซ์

λ คือ เป็นค่าคงที่ใดๆ เป็นสเกลาร์

v คือ ค่าไอเกนเวกเตอร์

จากสมการจะเห็นว่า $v = 0$ ที่ทำให้สมการ เป็นจริงทุกๆ ค่า λ ของสมการ (2-26) อาจเขียนให้อยู่ในรูปอีกรูปหนึ่งคือ

$$(\lambda I - A)v = \bar{0} \qquad (2\text{-}27)$$

เมื่อ A คือ ค่าเมตทริกซ์

I คือ เมตทริกซ์เอกลักษณ์

λ คือ เป็นค่าคงที่ใดๆ เป็นสเกลาร์

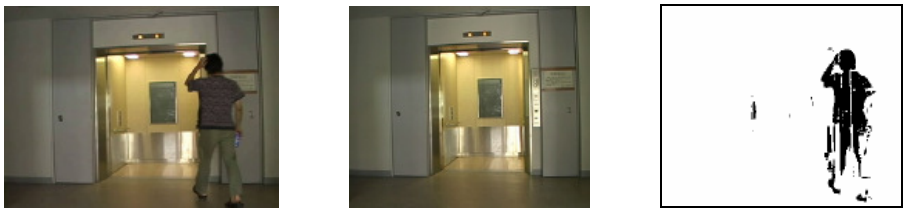
v คือ ค่าไอเกนเวกเตอร์

จากสมการ (2-27) จะมีคำตอบที่ไม่เป็นศูนย์ ก็ต่อเมื่อ $\det(\lambda I) - A = 0$ เรียกสมการนี้ว่าสมการแคแรกเทอริสติก ของ A

ขั้นตอน 4) eigen-vector จำนวน *M* ตัวแรกที่มีค่ามากที่สุด จะถูกใช้เป็นค่าของพื้นหลัง

ขั้นตอน 5) ภาพใหม่ *I* ที่เข้ามาจะถูกเทียบกับ eigen-vector จำนวน *M* ตัวที่เก็บไว้และสร้างภาพใหม่ *(I')*

ขั้นตอน 6) ผลต่างระหว่างภาพที่เข้ามา *(I)* กับภาพที่สร้างใหม่ *(I')* จะถูกคำนวณเพื่อนำมาหาวัตถุที่เคลื่อนที่



(ก) (ข) (ค)

ภาพประกอบ 2-15 ผลจากวิธี Eigenbackgrounds [5] (ก) ภาพเฟรมปัจจุบัน
(ข) ภาพลบพื้นหลัง (ค) ภาพวัตถุเคลื่อนที่

Principal Component Analysis (PCA) เป็นวิธีการวิเคราะห์องค์ประกอบหลักเป็นวิธีการทางสถิติ ซึ่งถูกนำไปประยุกต์ใช้งานต่างๆ เช่น การบีบอัดข้อมูล , การสร้างภาพใบหน้าไอเกนเพื่อใช้ในระบบจดจำ และ การลบภาพพื้นหลังโดยใช้ไอเกน เป็นต้น

วิธีการวิเคราะห์องค์ประกอบหลักสามารถนำมาใช้ในการลดมิติของข้อมูลโดยการวิเคราะห์ข้อมูลและเลือกเฉพาะข้อมูลที่มีความสำคัญเท่านั้น ส่วนข้อมูลที่ไม่สำคัญจะถูกตัดทิ้งไป ดังนั้นเมื่อภาพผ่านกระบวนการ PCA แล้ว จะได้ผลลัพธ์เป็นไอเกนเวกเตอร์และค่าไอเกน ซึ่งไอเกนเวกเตอร์ที่มีค่าสมนัยกับค่าไอเกนที่มีค่าสูงๆ จะเป็นการดึงข้อมูลที่มีความถี่ต่ำ (เป็นข้อมูลของภาพพื้นหลัง) ส่วนไอเกนเวกเตอร์ที่สมนัยกับค่าไอเกนที่ต่ำๆ จะเป็นการดึงข้อมูลที่มีความถี่สูง (ไม่ใช่ภาพของพื้นหลัง)

2.4 การลบสัญญาณภาพรบกวนด้วยวิธี Morphological Image Processing

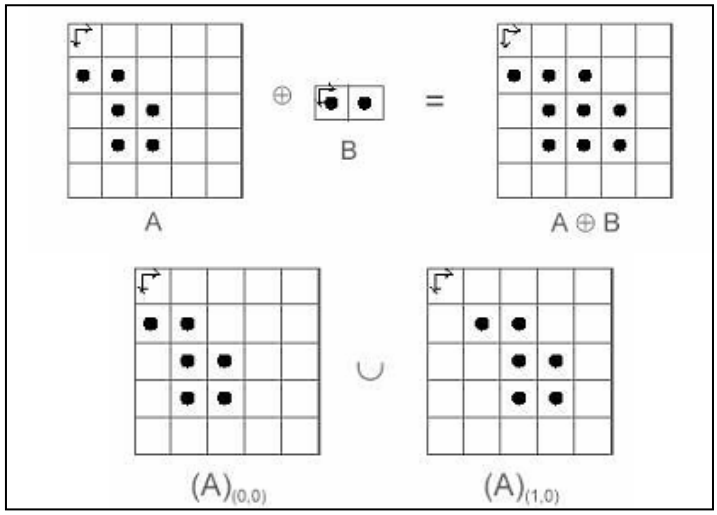
Morphological Image Processing เป็นการประมวลผลภาพโดยการเปลี่ยนแปลงลักษณะรูปร่างหรือโครงสร้างของภาพ โอเปอเรชั่นพื้นฐานโดยทั่วไปได้แก่ การ Dilation และErosion โดยการ Dilation คือการขยายภาพโดยมีสัดส่วนเท่ากันทั่วทั้งภาพ(Uniform) การ Erosion คือการย่อภาพ

2.4.1 การขยายภาพ (Dilation)

การขยายภาพจะทำได้โดยกำหนดเทมเพลต ซึ่งสามารถสร้างได้จาก * และ 1 โดยมีจุดเริ่มต้นที่กำหนดโดยวงกลมและนำ Template นี้สแกนไปบนข้อมูลภาพตามลำดับตลอดทั้งภาพ ซึ่งในขณะที่จุดเริ่ม(Origin)ของ Template ตรงกับตำแหน่งข้อมูลภาพที่พิกเซลมีค่าเท่ากับ 1 นั้นก็จะทำการยูเนียน Template นี้เข้ากับข้อมูลภาพตามสมการ (2-28) และ (2-29) ดังภาพประกอบ 2-16

$$A \oplus B = \{x \mid x = a + b, a \in A, b \in B\} \tag{2-28}$$

$$A \oplus B = \bigcup_{b \in B} (A)_b \tag{2-29}$$



ภาพประกอบ 2-16 กระบวนการขยายภาพด้วยเทมเพลตเมตริก(1x2) [3]

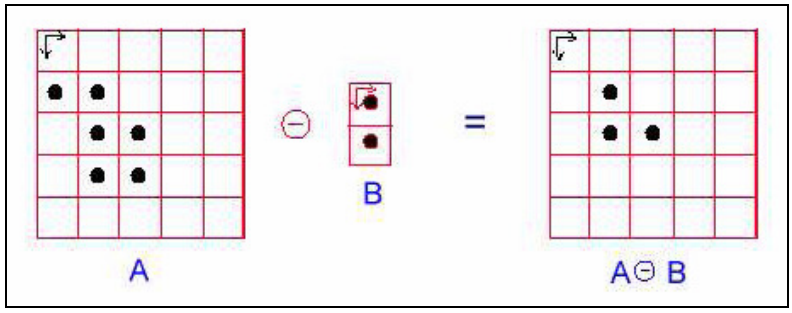
2.4.2 การย่อภาพ (Erosion)

การย่อภาพเป็นลักษณะของการลบข้อมูลภาพบริเวณขอบของภาพ การย่อภาพสามารถทำได้มีลักษณะคล้ายกับการขยายภาพโดยการสร้าง Template ขึ้นแล้วนำ Template ไปสแกนตามข้อมูลภาพ สำหรับทุกตำแหน่งที่เลื่อน Template ไปบนภาพก็จะมีการเปรียบเทียบกับข้อมูลภาพ ถ้าข้อมูลภาพมีค่าเหมือนกับ Template จะทำการกำหนดค่าข้อมูลภาพในตำแหน่งที่ตรงกับจุดเริ่มต้น(Origin)ของ Template ถูกกำหนดให้มีค่าเท่ากับ 1 ตามสมการ (2-30) (2-31) และ (2-32) และภาพประกอบ 2-17

$$A \Theta B = \{x \mid x + b \in A, \forall b \in B\} \tag{2-30}$$

$$A\Theta B = \{x \mid (B)_x \subseteq A\} \tag{2-31}$$

$$A\Theta B = \bigcup_{b \in B} (A)_{-b} \tag{2-32}$$



ภาพประกอบ 2-17 กระบวนการย่อภาพด้วยเทมเพลตเมตริก(2x1) [3]

2.4.3 โอเปอเรชั่น Opening

กำหนดให้ OPEN (I, T) เป็นการกระทำ Opening ของภาพ I โดยใช้ Template T ซึ่งมีลักษณะดังสมการ (2-33)

$$A \circ B = (A\Theta B) \oplus B \tag{2-33}$$

จากสมการ(2-27) จะได้ว่าการทำโอเปอเรชั่น OPEN คือการนำข้อมูลภาพ I ผ่านการทำขยายภาพ(Erosion) แล้วตามด้วยการย่อภาพ(Dilation) โดยใช้ Template ชุดเดียวกัน

2.4.4 โอเปอเรชั่นการ Close

กำหนดให้ CLOSE (I, T) เป็นการกระทำแบบ Closing ของภาพ I โดยใช้ Template T ซึ่งมีลักษณะดังสมการ(2-34)

$$A \bullet B = (A \oplus B)\Theta B \tag{2-34}$$

จากสมการ(2-28) จะได้ว่าการทำโอเปอเรชั่น CLOSE คือการนำข้อมูลภาพ I ผ่านการทำการย่อภาพ(Dilation)แล้วตามด้วยการขยายภาพ(Erosion)โดยใช้ Template ชุดเดียวกัน

2.5 สรุป

จากการศึกษาเทคนิคการลบภาพพื้นหลังด้วยวิธีต่างๆที่มีข้อดีและข้อเสีย ดังนี้ วิธีการหาผลต่างของเฟรมที่มีวิธีการไม่ซับซ้อน ประมวลผลได้รวดเร็ว การใช้ทรัพยากรหน่วยความจำขึ้นอยู่กับขนาดของเฟรมภาพ ข้อเสียคือความถูกต้องขึ้นอยู่กับการเลือกค่า Threshold วิธี Running Average และวิธี Selectivity ที่มีวิธีการไม่ซับซ้อน การใช้ทรัพยากรหน่วยความจำน้อย ความเที่ยงตรงน้อย วิธี Running Gaussian ที่มีวิธีการมีความซับซ้อนปานกลาง การใช้ทรัพยากรหน่วยความจำน้อย ความเที่ยงตรงน้อย วิธี Mixture of Gaussians ที่มีวิธีการที่มีความซับซ้อน การใช้ทรัพยากรหน่วยความจำที่ใช้การเก็บแบบสถิติ ความเที่ยงตรงปานกลาง และวิธี Eigenbackground ที่มีวิธีการที่มีความซับซ้อนมาก การใช้ทรัพยากรหน่วยความจำที่ใช้เฉพาะการสร้างเมตริกซ์ ความเที่ยงตรงสูง

การเลือกวิธีการที่เหมาะสมกับระบบการนับปริมาณรถที่ต้องการพัฒนาบนเทคโนโลยีเอฟพีจีเอ ที่มีภาพพื้นหลังของภาพการจราจรค่อนข้างคงที่ วิธีการจะต้องไม่ซับซ้อน ใช้ทรัพยากรหน่วยความจำน้อยเนื่องจากทรัพยากรที่มีจำกัด สามารถประมวลผลได้อย่างรวดเร็ว ซึ่งวิธีการลบภาพพื้นหลังโดยใช้เทคนิคการหาผลต่างเฟรมเป็นวิธีที่เหมาะสมกับระบบ

บทที่ 3

โมเดลการโปรแกรมแบบขนานบนเอฟพีจีเอ

ในบทนี้อธิบายถึงโมเดลการโปรแกรมแบบขนานบนเอฟพีจีเอ เริ่มต้นอธิบายถึงโมเดลแบบต่างๆสำหรับงานประยุกต์บนเอฟพีจีเอตั้งแต่โมเดลการโปรแกรมแบบพื้นฐานและการพัฒนาโมเดลการโปรแกรมมาจนถึงปัจจุบันที่มีการประมวลผลแบบขนาน รวมทั้งอธิบายการใช้ FPGAs เป็นตัวประมวลผลแบบขนานโดยใช้คอมไพเลอร์ทูลที่มีอยู่ใน ImpulseC อธิบายโมเดลการโปรแกรมด้วย ImpulseC และวิธีการปรับปรุงการโปรแกรมให้มีความเหมาะสม (Optimize) ให้เป็นการโปรแกรมแบบขนานภายใต้ทรัพยากรที่มีอยู่อย่างจำกัดบนเอฟพีจีเอ

3.1 โมเดลการประมวลผลแบบขนาน (Parallel Processing Model)

การเลือกโมเดลการโปรแกรมที่เหมาะสมเป็นสิ่งที่สำคัญ ในการสร้างแอปพลิเคชันและองค์ประกอบของขั้นตอนวิธีต่างๆ ในวิธีการทางทฤษฎี ภายใต้สิ่งที่ควรตระหนัก และทำให้โปรแกรมมีประสิทธิภาพ รายละเอียดของโมเดลการโปรแกรมในรูปแบบต่างๆ [4][18] เป็นดังนี้

สถาปัตยกรรมโพรเซสเซอร์โดยทั่วไปในปัจจุบันนี้ส่วนใหญ่มีพื้นฐานมาจากสถาปัตยกรรมที่ถูกนำเสนอโดย John von Neumann ในทศวรรษ 1940 สถาปัตยกรรม von Neumann ตั้งเดิมประกอบด้วยหน่วยประมวลผลกลางหรือ CPU (Central Processing Unit) เชื่อมต่อกับหน่วยความจำซึ่งเก็บทั้งคำสั่งและข้อมูล โดย CPU ทำการประมวลผลคำสั่งตามลำดับของคำสั่งที่ถูกเก็บไว้

- **SISD (Single Instruction, Simple Data machine)**

SISD เป็นโมเดลที่ง่ายที่สุด มีตัวประมวลผลเดียว (single-processor) ที่ถูกกำหนดให้มีชุดคำสั่งเดียว (single Instruction) และข้อมูลชุดเดียว (single data machine) ณ เวลาหนึ่งๆ สามารถประมวลผลได้เพียงโอเปอเรชันเดียวเท่านั้น เนื่องด้วยเป็นโมเดลที่ง่ายจึงมีภาษาโปรแกรมที่ใช้ในการพัฒนางานประยุกต์ถูกพัฒนาขึ้นจำนวนมาก

การเขียนโปรแกรมสำหรับ SISD เป็นกระบวนการซอฟท์แวร์ (software process) ที่มีการใช้ชุดคำสั่งแบบตามลำดับ โดยแต่ละกระบวนการอาจจะมีโอเปอเรชันเดียวหรือมากกว่าหนึ่งที่แตกต่างกัน ซึ่งอาจจะถูกขัดจังหวะไปทำงานกระบวนอื่นด้วยกระบวนการกระโดดแบบมี

เงื่อนไข การวนลูป และการเรียกฟังก์ชันย่อย ซึ่งทำให้ดูเสมือนทำงานหลายๆอย่างได้ในเวลาเดียวกัน วิธีการโปรแกรมหรือโมเดลการโปรแกรมในปัจจุบันล้วนถูกพัฒนามาเพื่อรองรับสถาปัตยกรรมแบบ SISD ถึงแม้จะมีการเพิ่มระดับการทำงานแบบขนานในรูปแบบของไปป์ไลน์

การที่โพรเซสเซอร์เอนกประสงค์มีสถาปัตยกรรมพื้นฐานแบบ von Neumann ร่วมกันนั้นเป็นประโยชน์สำหรับนักพัฒนาซอฟท์แวร์ในการพัฒนาเป็นแอพพลิเคชัน และระบบปฏิบัติการ ที่สามารถนำไปพัฒนาระบบที่ซับซ้อนและมีประสิทธิภาพมากขึ้น สามารถรองรับระบบการทำงานแบบหลายงาน (Multitasking system) ที่ทำให้เสมือนว่าโพรเซสเซอร์ทำงานได้หลายๆอย่างในเวลาเดียวกัน ภาษาที่ใช้เขียนก็สามารถถูกปรับเปลี่ยนได้ง่ายและสามารถพัฒนาเป็นภาษาใหม่ได้โดยไม่เปลี่ยนพื้นฐานเดิมมากนัก

- **SIMD (Single Instruction, Multiple Data)**

SIMD เป็นสถาปัตยกรรมในโดเมนของซุปเปอร์คอมพิวเตอร์ ที่สามารถรองรับการทำงานแบบขนานที่มากขึ้น มีตัวประมวลผลควบคุมโดยตรงไปยังตัวดำเนินการ เพื่อดำเนินการในแบบขนานกับส่วนอื่นๆ ได้ ยกตัวอย่างเช่นคำสั่งในการคูณเมตริกซ์ ที่ต้องใช้วงจรคูณหลายๆตัวทำงานพร้อมกันโดยที่แต่ละตัวมีข้อมูลเข้า-ออกเป็นของตัวเอง

- **MIMD (Multiple Instructions on Multiple Data)**

MIMD เป็นสถาปัตยกรรมที่สามารถดำเนินการได้หลายคำสั่งบนหลายชุดข้อมูลขนานกัน มีตัวประมวลผลหลายตัวสามารถรองรับการดำเนินการคำสั่งที่แตกต่างกัน โดยแต่ละคำสั่งมีข้อมูลที่แยกกัน ทำงานขนานไปพร้อมๆกันได้ ซึ่งดูเหมือนว่าจะเป็นสถาปัตยกรรมที่ดีที่สุด แต่การโปรแกรมระบบนี้มีสิ่งที่จำเป็นในกระบวนการจำนวนมากสำหรับการประสานงานตัวประมวลผลหลายตัวทำงานสัมพันธ์กัน ส่งผลให้เกิดความยุ่งยากและอาจเกิดปัญหาในการโปรแกรมได้ ซึ่งผู้เขียนโปรแกรมต้องระมัดระวังในเรื่องลำดับการควบคุมระบบ

มีงานวิจัยเป็นจำนวนมากที่พัฒนาระบบมัลติคอมพิวเตอร์(Multicomputer) และกระบวนวิธีที่จะทำการโปรแกรม ตัวอย่างหนึ่งคือ ทรานสพิวเตอร์ (Transputer) ซึ่งถูกพัฒนาขึ้นมาโดย INMOS ในช่วงกลางทศวรรษที่ 1980 ทรานสพิวเตอร์เป็นอุปกรณ์แผ่นปรินท์สำหรับสร้างส่วนประกอบของระบบคอมพิวเตอร์ที่มีส่วนย่อยเป็นอาร์เรย์ของชิพเดี่ยว รองรับการซิงโครไนซ์ (Synchronize) ของชิพแบบเป็นอิสระต่อกัน มาเชื่อมต่อโดยไม่มีการจำกัดขนาด และความซับซ้อน

การเชื่อมต่อระหว่างชิพของระบบทรานสพิวเตอร์เป็นแบบอนุกรม ดังนั้นปัญหาคอขวด(Bottleneck) ที่เกิดขึ้นเป็นการเคลื่อนย้ายข้อมูลมากกว่าความสามารถในการประมวลผลในแต่ละชิพ แต่อย่างไรก็ตามถือว่า ระบบทรานสพิวเตอร์เป็นโมเดลที่มีประสิทธิภาพสูงมากเมื่อ

เทียบกับราคาของอุปกรณ์ที่มีราคาต่ำ และแนวคิดที่นำไปสู่การพัฒนาระบบประมวลผลสมรรถนะสูงบนแพลตฟอร์มเอฟพีจีเอ

เนื่องจากระดับการทำงานแบบขนานที่สูงของระบบทรานสพิวเตอร์ ทำให้ต้องใช้ภาษาโปรแกรมที่จำเพาะคือ Occam ซึ่งสามารถรองรับการทำงานแบบขนานได้หลายระดับ เช่นการขนานระดับกระบวนการ (process) การขนานระดับคำสั่ง และบล็อกของคำสั่ง

- **สถาปัตยกรรม MIMD แบบหน่วยความจำร่วม**

เนื่องจากการสื่อสารของระบบทรานสพิวเตอร์เป็นแบบอนุกรม ระบบนี้อาจถือเป็น สถาปัตยกรรมส่งผ่านข้อมูล (message-passing architecture) ซึ่งไม่มีหน่วยความจำร่วมกันหรือไม่มีการใช้ทรัพยากรในระบบร่วมกัน ข้อมูลจะเคลื่อนที่จากกระบวนการหนึ่งไปยังอีกกระบวนการหนึ่งเป็นแพ็กเก็ตขนาดเล็กบนช่องทางสื่อสารที่มีบัฟเฟอร์หรือไม่มีบัฟเฟอร์พักข้อมูล ด้วยการเชื่อมต่อกันอย่างง่ายนี้ทำให้เป็นไปได้ที่จะทำให้ตัวประมวลผลต่างๆสามารถทำงานเป็นอิสระต่อกันโดยที่การขัดแย้งด้านการใช้ทรัพยากรน้อยลง

ยังมี MIMD ประเภทอื่นที่มีการรวมเอาการใช้หน่วยความจำร่วมกัน ซึ่งอาจจะถูกจัดลำดับชั้นเป็นแคช (cache) ของเฉพาะตัวประมวลผลหนึ่งสามารถเข้าถึงได้รวดเร็ว และหน่วยความจำทั่วไปที่ใช้ร่วมกันในระบบ โดยหน่วยความจำเหล่านี้ถูกใช้ร่วมหรือเป็นตัวเลือกหนึ่งในกระบวนวิธีส่งผ่านข้อมูลให้มีประสิทธิภาพ

3.2 การใช้ FPGAs เป็นตัวประมวลผลแบบขนาน

เป็นที่รู้กันโดยทั่วไปว่าเอฟพีจีเอสามารถรองรับการคำนวณแบบขนานได้อย่างเหมาะสม และเพิ่มความเร็วให้กับอัลกอริทึมที่ซับซ้อน แต่เอฟพีจีเอมีข้อจำกัดในเรื่องทรัพยากรที่มีจำนวนจำกัด จึงเป็นสิ่งท้าทายในการพัฒนาวิธีจัดการกับอัลกอริธึมขนาดใหญ่ให้สามารถทำงานได้อย่างมีประสิทธิภาพภายใต้ข้อจำกัดของทรัพยากร

ความสำเร็จในการใช้งานเอฟพีจีเอเป็นตัวประมวลผลแบบขนานนั้น ได้มาจากการใช้คอมไพเลอร์ทูล (Compiler tool) และภาษาโปรแกรมที่สามารถบรรยายการประมวลผลแบบขนานได้ ถึงแม้ว่าคอมไพเลอร์ทูลที่ใช้จะสามารถคัดแยกการประมวลผลที่ขนานกันในโปรแกรมงานประยุกต์ขนาดใหญ่ได้ แต่ถ้าโปรแกรมงานประยุกต์ถูกพัฒนาขึ้นโดยไม่ได้พิจารณาการทำงานแบบขนานตั้งแต่แรก คอมไพเลอร์ทูลก็ไม่สามารถคัดแยกการประมวลผลที่ขนานกันออกมาได้อย่างมีประสิทธิภาพ นั่นคือไม่สามารถทำการตัดสินใจและปรับปรุงที่ระดับระบบ (system-level optimization) ให้สามารถใช้ประโยชน์ของโครงสร้างแบบขนานที่มีอยู่ได้ เป็นเหตุให้นักออกแบบ

ฮาร์ดแวร์จำเป็นต้องทำการปรับปรุงในระดับล่างเพิ่มอีก เพื่อที่จะได้วงจรที่มีประสิทธิภาพสูงสุด ดังนั้นจึงเป็นสิ่งที่สำคัญที่จะต้องพัฒนาการโปรแกรมและวิธีการแบ่งส่วนย่อย (Partitioning) ที่สามารถทำให้คอมโพเนนท์ที่ถูกแบ่งเป็นอิสระต่อกัน เพื่อให้สามารถปรับปรุงเฉพาะคอมโพเนนท์ใดๆ โดยไม่ต้องออกแบบใหม่ทั้งระบบ

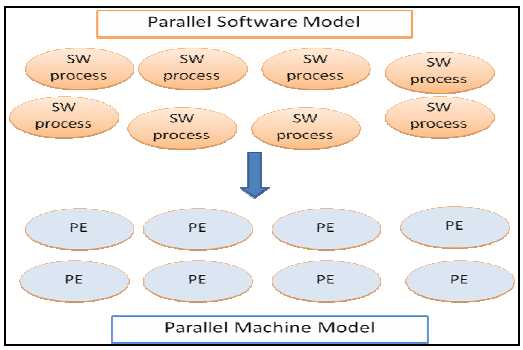
การแบ่งส่วนย่อยแบบขนานของแอพพลิเคชันสำหรับ System level จำเป็นสำหรับโมเดลการโปรแกรมที่มีแนวคิดต่างจากการโปรแกรมซอฟท์แวร์ทั่วๆไป ซึ่งเป็นโมเดลที่มีการแบ่งเป็นฟังก์ชันย่อยที่เป็นอิสระต่อกัน แล้วคอมไพล์เป็นบล็อกของฮาร์ดแวร์ ต่างกับตัวประมวลผลที่แปลงเป็นภาษาแอสแซมบลี ไม่มีไซเคิลของการเฟทช์และเอ็กซิคิวท์แบบตัวประมวลผล แต่คอมโพเนนท์ทั้งหมดของโปรแกรมสามารถทำงานเป็นแบบขนานได้

การพัฒนาโปรแกรมเพื่อรองรับการทำงานแบบขนาน กล่าวคือบล็อกของโปรแกรมหลายๆส่วนจะทำงานไปพร้อมๆกันบนข้อมูลหลายๆสตรีม และในแต่ละบล็อกจะเป็นโครงสร้างแบบขนาน จำเป็นต้องอาศัยหลายๆโมเดลผสมกัน ต้องสามารถรองรับทั้งคำสั่งทำงานแบบขนาน (Parallel statement) และในขณะเดียวกันต้องมีคำสั่งทำงานแบบตามลำดับ (Procedural statement) ได้ด้วย ซึ่งภาษา ImpulseC ที่ใช้ในงานวิจัยนี้สามารถรองรับการทำงานที่กล่าวมาข้างต้นได้เป็นอย่างดี

3.2.1 การโปรแกรมสำหรับการประมวลผลแบบขนาน

การโปรแกรมสำหรับการประมวลผลแบบขนานจำเป็นต้องสามารถรองรับการทำงานแบบแข่งขนาน (Concurrency) สามารถประสานการทำงานของตัวประมวลผลหลายๆตัวที่ทำงานอิสระต่อกัน ภาษาซีโดยทั่วไปไม่สามารถบรรยายการประมวลผลแบบขนานนี้ได้ ส่วนภาษาบรรยายฮาร์ดแวร์ VHDL หรือ Verilog นั้นสามารถบรรยายการประมวลผลแบบขนานได้ แต่เป็นภาษาที่ระดับต่ำกว่า ผู้เขียนต้องมีความเข้าใจการเชื่อมต่อของฮาร์ดแวร์เป็นอย่างดี

โมเดลภาษาที่เหมาะสมสำหรับการออกแบบระบบที่ทำงานแบบขนานควรผสมผสานระหว่างภาษาสำหรับส่วนซอฟท์แวร์และภาษาสำหรับส่วนของฮาร์ดแวร์ ในส่วนซอฟท์แวร์ควรมีลักษณะที่ดีของภาษาซีที่รองรับการทำงานแบบขนานคล้ายกับ Thread ในระบบปฏิบัติการ และมีไลบารีไว้สำหรับดึงมาประกอบกันได้เพื่อรองรับออกแบบงานประยุกต์ขนาดใหญ่ ในส่วนฮาร์ดแวร์ควรมีการจัดเตรียมฮาร์ดแวร์อิลิเมนต์ (ประกอบด้วยเกทและฟลิบฟลอปที่โปรแกรมได้) ให้เป็นโครงสร้างที่เหมาะสมสำหรับการเขียนโปรแกรมที่ระดับสูง และง่ายต่อคอมไพเลอร์ในการแปลงมาเป็นวงจรระดับเกทบนเอฟพีจีเอ ดังภาพประกอบ 3-1 บล็อกของโปรเซสซอฟท์แวร์ที่ขนานกันถูกแปลงเป็นโปรเซสอิลิเมนต์อย่างตรงไปตรงมา



ภาพประกอบ 3-1 โมเดลการโปรแกรมของการประมวลผลแบบขนาน

3.2.2 กระบวนการเชื่อมต่อกันของโมเดลการโปรแกรม

โมเดล CSP (Communicating Sequential Programming) เป็นส่วนของโมเดลการโปรแกรมและภาษาที่เป็นรูปแบบที่สัมพันธ์กันระหว่างส่วนของกระบวนการทำงานหลายส่วนที่เป็นอิสระต่อกัน แต่ละกระบวนการทำงานในระบบที่เป็นส่วนฮาร์ดแวร์ที่มีการโปรแกรมเป็นแบบตามลำดับ จึงมีข้อจำกัดในเรื่องการเชื่อมต่อผ่านช่องทางข้อมูล(Channel) ไปยังกระบวนการอื่น และแต่ละกระบวนการมีสถาปัตยกรรมพื้นฐานแบบ Von noumann ที่มีหน่วยควบคุมหลักเรียกใช้กระบวนการทำงานอื่นๆได้ที่เป็นชุดคำสั่งย่อย (Subroutines)

ข้อมูลของแอพพลิเคชันจะถูกออกแบบเป็นบัฟเฟอร์ส่งผ่านไปยังระบบผ่านช่องทางข้อมูล ในขณะที่แต่ละกระบวนการกำลังดำเนินการ แต่ละกระบวนการจะเข้าไปจัดการทรัพยากรหน่วยความจำภายในโดยตรง(สำหรับการคำนวณและเก็บผลลัพธ์) ทำให้ไม่มีการเชื่อมต่อกันหรือมีการเชื่อมต่อกันเล็กน้อยระหว่างกระบวนการแบบอิสระผ่านทางช่องทางข้อมูลหรือสตรีม

โมเดล CSP ไม่มีการหน่วงเวลา(Time delay) ในการส่งข้อมูลระหว่างสองโหนด ไม่มีการคำนึงถึงตำแหน่ง โดยเฉพาะอย่างยิ่งถ้าแอพพลิเคชันมีส่วนของกระบวนการทำงานที่แตกต่างกัน เช่น ส่วนฮาร์ดแวร์ในเอฟพีจีเอ และตัวประมวลผลแบบฝังตัว การอิมพลิเมนท์แบบโมเดล CSP จะไม่มีการเชื่อมต่อภายนอกผ่านช่องทางข้อมูลหรือสตรีม

สิ่งที่สำคัญของโมเดลแบบ CSP เป็นการเข้าถึงหน่วยความจำภายในซึ่งใช้หน่วยความจำน้อยกว่าการเข้าถึงหน่วยความจำจากภายนอก ซึ่งการเข้าถึงข้อมูลภายในจะเป็นสิ่งที่สำคัญมากแตกต่างกับการเข้าถึงข้อมูลจากภายนอก ดังนั้น เพื่อให้ได้ประสิทธิภาพสูงสุดแนวคิดการโปรแกรมแบบขนานจะถูกเรียกว่า Locality

การใช้โมเดลการโปรแกรมที่อยู่บนพื้นฐานของโมเดล CSP เป็นสิ่งที่ดีสำหรับการคอมไพล์แอพพลิเคชันแบบขนานในภาษาซี ส่วนของกระบวนการทำงานแบบโมเดล CSP แอพ

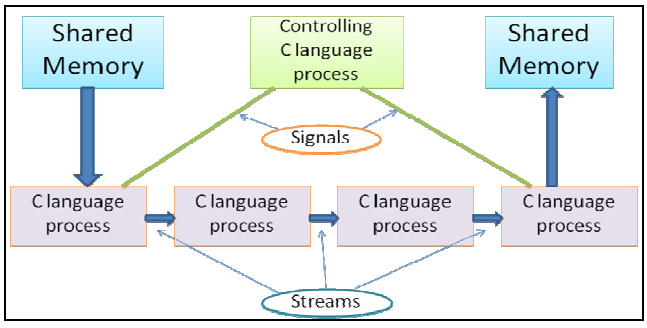
พลิเคชันจะถูกล้อมรอบด้วยบล็อกของการคำนวณ ซึ่งตัวคอมไพเลอร์จะสร้าง(Generate)ส่วนของฮาร์ดแวร์ หรือซอฟท์แวร์ ได้โดยตรงตามมาตรฐานของภาษาซีในระดับ System Level มีการจัดเตรียมรูปแบบของฟังก์ชันไลบรารีที่กระบวนการทำงานต้องการ รวมทั้งช่องทางสื่อสารในการส่งสตรีมในการโปรแกรมแบบขนาน

การโมเดลแบบผสมนี้อาจจะไม่ได้ใช้กับทุกแอพพลิเคชัน แต่จะเป็นประโยชน์อย่างมากกับแพลทฟอร์มแบบขนานที่ช่วยเร่งความเร็วในการประมวลผลบนเอฟพีจีเอ

3.3 โมเดลโปรแกรมของ ImpulseC

การโปรแกรมด้วยภาษา ImpulseC เป็นโมเดลแบบ CSP ใช้สำหรับสตรีม และเป็นแอพพลิเคชันที่เป็นแบบผสมกันของทั้งส่วนของฮาร์ดแวร์และซอฟท์แวร์ ช่วยในการคำนวณในส่วนของฮาร์ดแวร์เพื่อเร่งความเร็วของการทำงาน เช่น บางแอพพลิเคชันในหลายๆโดเมนที่เกี่ยวกับการประมวลผลภาพ การส่งผ่านข้อมูล การส่งผ่านข้อมูลแบบไร้สาย และอื่นๆ

หัวใจของการโมเดลการโปรแกรมภาษา ImpulseC เป็นส่วนของกระบวนการทำงานและสตรีม กระบวนการทำงานต่างๆ ถูกซิงโครไนซ์อย่างเป็นอิสระต่อกัน คอมโพเนนท์ต่างๆของแอพพลิเคชันจะทำงานไปพร้อมๆกัน ดังภาพประกอบ 3-2 เป็นประโยชน์ต่อกระบวนการทำงานเหมือนเป็นโปรแกรมย่อยๆที่จัดการกับข้อมูลจำนวนมากมีการคำนวณและสร้างข้อมูลเอาท์พุตผ่านทางสตรีมข้อมูล หรือเขียนลงบนหน่วยความจำ กระบวนการทำงานจะไม่เหมือนรูทีนส์ย่อยของซอฟท์แวร์พื้นฐานโดยทั่วไป เนื่องจากแต่ละกระบวนการทำงานไม่มีการถูกเรียกใช้ข้อมูลที่เข้ามาจะเป็นข้อมูลอินพุตของแต่ละกระบวนการที่เชื่อมต่อกัน เมื่อรับข้อมูลเข้ามาแต่ละกระบวนการทำการคำนวณแล้วสร้างเป็นเอาท์พุต



ภาพประกอบ 3-2 การจัดการข้อมูลโดยกระบวนการทำงานต่างๆผ่านช่องทางสตรีมโดยใช้หน่วยความจำร่วมกัน

โมเดลการโปรแกรมแบบ CSP ที่ถูกออกแบบเป็นแอพพลิเคชันแบบขนานอย่างง่ายที่ใช้การจัดการสตรีมในการส่งข้อมูล จัดการข้อมูล และซิงโครไนซ์ ในแอพพลิเคชันที่ไม่จำกัดจำนวน

คุณลักษณะที่เหมาะสมกับการใช้โมเดล CSP ดังนี้

- แอพพลิเคชันที่มีอัตราข้อมูลสูงระหว่างแต่ละกระบวนการทำงาน
- ข้อมูลที่มีขนาดที่แน่นอน
- ข้อมูลที่เกี่ยวข้องกันแต่การคำนวณเป็นอิสระต่อกันที่ส่งผ่านสตรีมเดียวกัน
- ข้อมูลที่ประกอบด้วยค่าจุดทศนิยมต่ำหรือคงที่ เป็นตัวเลขจำนวนเต็มขนาดคงที่
- ข้อมูลที่มีการใช้หน่วยความจำร่วมกันหรือหน่วยความจำภายใน ที่ใช้สำหรับเก็บข้อมูลอะเรย์ ค่าสัมประสิทธิ์ค่าคงที่อื่นๆ และผลลัพธ์จากการคำนวณ
- กระบวนการทำงานจำนวนมากที่เป็นอิสระต่อกันและมีการส่งผ่านข้อมูลสูง

ภาษา ImpulseC เป็นการออกแบบที่เฉพาะเจาะจงสำหรับการส่งข้อมูลจำนวนมากผ่านสตรีมแอพพลิเคชันหรือส่งผ่านไปยังแอพพลิเคชันอื่นๆ เนื่องจากภาษา ImpulseC เป็นโมเดลแบบ CSP ที่มีดาต้าไทป์ขนาดเล็ก ทำให้ภาษา ImpulseC สามารถเรียกใช้ไลบรารีฟังก์ชันจากการโปรแกรมภาษาซีและกำหนดแอพพลิเคชัน สตรีม สัญญาณต่างๆ และการใช้หน่วยความจำร่วมกัน ฟังก์ชันไลบรารีที่กำหนดในภาษา ImpulseC ใช้ส่งสตรีมและข้อมูลสัญญาระหว่างกระบวนการทำงานต่างๆ เพื่อส่งข้อมูลเข้าและออกจากหน่วยความจำร่วมหรือหน่วยความจำภายในและต้องอยู่ภายใต้ทรัพยากรที่มีอยู่จำกัดของเอฟพีจีเอ

3.4 โมเดลการคำนวณบนเอฟพีจีเอ

การคำนวณบนเอฟพีจีเอดำเนินการโดยหน่วยคำนวณทางคณิตศาสตร์ เช่น ตัวบวก (adder) ตัวลบ (subtractor) และตัวคูณ (multipliers) เสมือนหน่วยประมวลผลกลางที่ตัวคำนวณเหล่านี้สามารถดำเนินการแบบขนาน แต่ในเอฟพีจีเอระดับของการดำเนินการแบบขนานถูกจำกัดด้วยขนาดของฮาร์ดแวร์ที่มีจำกัด และโครงสร้างการเชื่อมต่อภายในเอฟพีจีเอ

หน่วยประมวลผลกลางและเอฟพีจีเอ มีตัวควบคุมสัญญาณนาฬิกาที่ควบคุมความเร็วของตัวดำเนินการภายในวงจร ตัวดำเนินการต่างๆมีการทำงานเฉพาะในไซเคิล ความเร็วสูงสุดของสัญญาณนาฬิกาได้ถูกจำกัดด้วยความเร็วเฉพาะของตัวดำเนินการทางคณิตศาสตร์ หน่วยประมวลผลกลางโดยทั่วไปจะมีเพียงหนึ่งสัญญาณนาฬิกาที่ควบคุมการทำงานของตัวดำเนินการ

ทางคณิศาสตร์ ในขณะที่เอฟพีจีเอใช้หลายสัญญาณนาฬิกาควบคุมตัวดำเนินการทางคณิตศาสตร์ต่างๆ

โมเดลของเอฟพีจีเอจะต้องมีตัวควบคุมลำดับขั้นตอนของการเอ็กซิคิวท์ การจัดการข้อมูลสำหรับตัวดำเนินการและจัดการผลลัพธ์ แต่ตัวควบคุมโมเดลของหน่วยประมวลผลกลางใช้เฉพาะการทำงานในอัลกอริทึม ที่ทำให้การประมวลผลช้า ในระยะต่อมาจึงได้พัฒนาหน่วยประมวลผลกลางดังนี้ หลีกเลี่ยงการควบคุมและการคำนวณที่เวลาเดียวกัน หลีกเลี่ยงการสร้างตัวดำเนินการที่มีจำนวนมาก หลีกเลี่ยงการใช้เพียงหนึ่งไซเคิลในการทำงาน และหลีกเลี่ยงการเข้าถึงหน่วยความจำครั้งเดียวทั้งระบบ ซึ่งการพัฒนานี้ทำให้ฮาร์ดแวร์ที่ถูกสร้างสำหรับการอิมพลีเมนท์บนเอฟพีจีเอหน่วยความจำแรมได้หลายบล็อกและใช้หน่วยความจำจากภายนอกได้ เพื่อจัดเตรียมให้กับกระบวนการทำงานแบบขนาน

3.5 ภาษาซีและการโปรแกรมแบบขนาน

การสร้างส่วนของฮาร์ดแวร์ที่มีประสิทธิภาพจากภาษาซี ผู้เขียนโปรแกรมจะต้องเข้าใจกระบวนการแบบขนานที่จะนำมาเพิ่มเติมในส่วนของการโปรแกรมแบบดั้งเดิม วิธีการปรับปรุงการโปรแกรมให้มีความเหมาะสม (optimization) คือ ต้องเขียนโปรแกรมอย่างง่ายที่สุดและไม่ซับซ้อนเท่าที่จะเป็นไปได้ เพื่อให้มีประสิทธิภาพตามที่ต้องการ

3.6 การปรับปรุงการโปรแกรมให้มีความเหมาะสม

ภาษา ImpulseC ได้มีการรวมเอาภาษาซีและ RTL ที่มีตัวปรับปรุงการโปรแกรมให้มีความเหมาะสม(optimizer) เป็นการโปรแกรมแบบขนานเฉพาะสำหรับเอฟพีจีเอ เมื่อทำการโปรแกรมกระบวนการทำงานของ ImpulseC เป็นสิ่งที่สำคัญที่จะต้องเข้าใจว่าจะทำให้โปรแกรมภาษาซีเป็นแบบขนานได้อย่างไร ต้องทำการปรับปรุงการโปรแกรมให้มีความเหมาะสมเพื่อให้ได้ผลลัพธ์ตามต้องการ การโปรแกรมในส่วนของขนาดและความเร็วของลอจิกให้มีความเหมาะสมนั้น จะมีบล็อกส่วนเฉพาะของแต่ละตัวปรับปรุงการโปรแกรมให้มีความเหมาะสม ที่พยายามหาสเตจของคำสั่งให้มีจำนวนน้อยที่สุดโดยใช้กระบวนการไปป์ไลน์หรือคำสั่งแบบ scheduling ที่ไม่ทำให้ผลลัพธ์ผิดพลาด

3.6.1 คำสั่งแบบ Scheduling

คำสั่งแบบ Scheduling ประกอบด้วย การดึงส่วนการโปรแกรมแบบขนาน (parallel extraction) การอนุมานตัวควบคุม(control inference) และ การแทรกตัวรีจิสเตอร์(register insertion) การดึงส่วนการโปรแกรมแบบขนานมีความจำเป็นสำหรับทำให้ผลลัพธ์ที่มีประสิทธิภาพที่สุดของอัลกอริทึมแบบตามลำดับ ในขณะที่ coarse-grained ของการโปรแกรมแบบขนานจะดีขึ้นอย่างรวดเร็วที่ระดับ system ถึงแม้จะเป็นระบบที่แบ่งส่วนการทำงาน ตัวคอมไพเลอร์แบบ C-to-RTL จะกระทำการหาข้อมูลที่เป็นอิสระต่อกัน เพื่อนำมาใช้ในการออกแบบโปรแกรมแบบขนานและออกแบบคำสั่งแบบ Scheduling การโปรแกรมแบบขนานอาจจะมีการเพิ่มการโปรแกรมแบบไปป์ไลน์ และ Loop unroll

3.6.2 กระบวนการทำงานแบบไปป์ไลน์

ไปป์ไลน์ คือ เทคนิคทำให้คำสั่งหลายๆ คำสั่งทำงานพร้อมๆ กัน แต่ละส่วนจะทำงานให้เสร็จในส่วนของมัน และทำงานต่างกัน การทำงานแต่ละส่วนเรียกว่าสเตจซึ่งแต่ละสเตจทำงานไปพร้อมๆกัน เปรียบเสมือนสายพานเครื่องจักรในโรงงานที่แบ่งส่วนกันทำงานและทำงานไปพร้อมๆกันได้

สเตจที่เกิดขึ้นการทำงานแบบขนาน ถ้าไม่มีการทำงานแบบไปป์ไลน์ สเตจที่เกิดขึ้นจะเป็นแบบตามลำดับ ทุกๆคำสั่งภายในสเตจจะถูกอิมพลีเมนท์บนลอจิกจำนวนมากโดยใช้เพียงหนึ่งสัญญาณนาฬิกา ตัวอย่างการเขียนด้วยภาษาซีเพื่อให้คำสั่งมีระดับการทำงานแบบขนานดังนี้

แบ่งการทำงานที่ซับซ้อนเพื่อเตรียมสำหรับการคำนวณแบบขนาน เช่น x = (a+b)*(c+d)

ตัวดำเนินการประกอบด้วยตัวดำเนินการบวกสองตัว ที่ต้องการสองไซเคิลสำหรับสองคำสั่งที่แตกต่างกันในการหาผลรวมก่อนที่จะนำผลรวมนั้นมาคูณกัน ส่วนการโปรแกรมแบบขนาน ตัวดำเนินการบวกทั้งสองตัวจะเกิดขึ้นในสองไซเคิลของสองคำสั่งคำนวณแบบขนาน เนื่องจาก การดำเนินการของ a+b และ c+d เป็นอิสระต่อกัน

การโปรแกรมที่แบ่งย่อยที่สามารถนำมาโปรแกรมแบบขนานได้นั้น ลำดับของคำสั่งจะต้องเป็นอิสระต่อกัน เช่น x = a+b และ y = a+c จากทั้งสองคำสั่งหรือจำนวนคำสั่งที่มากกว่านี้ สามารถโปรแกรมเป็นโครงสร้างของฮาร์ดแวร์แบบขนานได้ที่ต้องการเพียงสัญญาณนาฬิกาเพียงหนึ่งสัญญาณสำหรับกระบวนการนี้

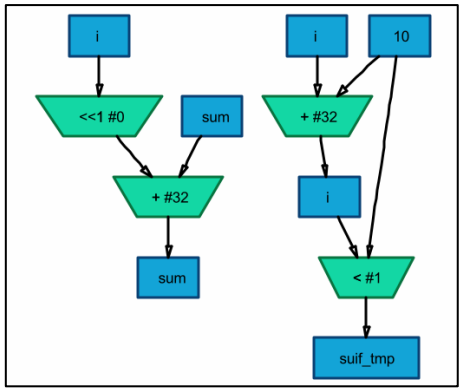
การทำซ้ำหลายๆครั้งภายในคำสั่งวนรอบสามารถโปรแกรมแบบขนานได้โดยใช้เทคนิคไปป์ไลน์ ที่ทำให้การทำซ้ำในรอบถัดไปของการวนรอบเริ่มเอ็กซิคิวท์ก่อนที่การทำซ้ำในใน

รอบปัจจุบันและเสร็จสมบูรณ์ในแบบขนานด้วยการเอ็กซิคิวท์ในการทำซ้ำในรอบปัจจุบัน ดังตัวอย่าง คำสั่งการวนรอบ

```
for(i=0 ; i<10 ; i++)
    sum +=i<<1;
```

ผลที่ได้จากการวนรอบในหนึ่งสเตจที่ไม่มีการใช้เทคนิคไปป์ไลน์ ด้วยตัวบวกหนึ่งตัวและตัวชิฟหนึ่งตัวที่ทำงานตามลำดับดังภาพประกอบ 3-3 และทำการเอ็กซิคิวท์ 10 ครั้ง สามารถคำนวณเวลาในการคำนวณการทำงานได้ดังสมการ (3.1) โดยที่ i คือ จำนวนการวนรอบ delay (shift) คือ เวลาทำงานของวงจร shifter (<<) และ delay (adder) คือ เวลาการทำงานของวงจร adder (+)

$$\text{Computation time} = (i) \times (delay(shifter) + delay(adder)) \qquad (3.1)$$



ภาพประกอบ 3-3 สเตจการทำงานของคำสั่งวนรอบแบบตามลำดับที่ไม่ใช้เทคนิคไปป์ไลน์

การใช้เทคนิคไปป์ไลน์ในคำสั่งวนรอบที่ทำซ้ำหลายครั้งในตัวอย่างที่แล้ว สามารถทำได้ดังนี้

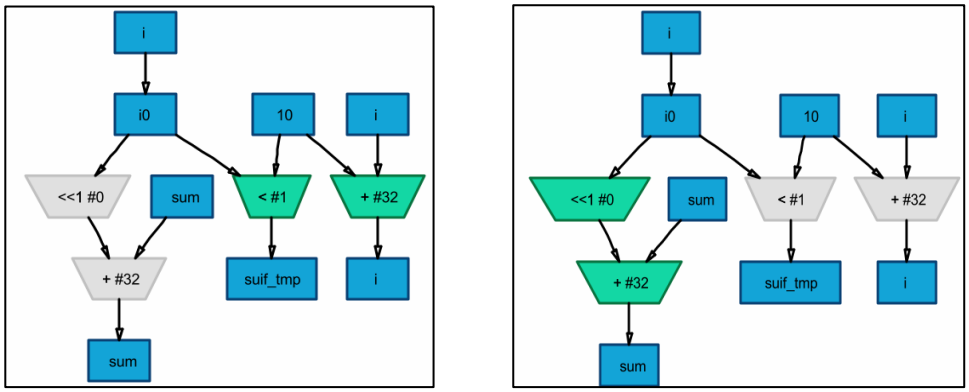
```
for(i=0 ; i<10 ; i++){
    #pragma CO PIPELINE
    sum +=i<<1;
    }
```

ผลลัพธ์ที่ได้เหมือนกับตัวอย่างที่แล้วที่ไม่มีการใช้เทคนิคไปป์ไลน์ แต่การทำงานแบ่งเป็นสองสเตจ ดังภาพประกอบ 3-4 ที่มีสเตจของการบวกค่าดังภาพประกอบ 3-4 (ก) และสเตจ

การชิฟ ดังภาพประกอบ 3-4 (ข) และทั้งสองสเตจทำงานพร้อมๆกัน ดังนั้น สามารถคำนวณเวลาในการคำนวณการทำงานได้ดังสมการ (3.2)

$$\text{Computation time} = (i) \times \max(delay(shifter), delay(adder)) \tag{3.2}$$

จากสมการที่ (3.2) การคำนวณเวลาการทำงานที่ใช้เทคนิคไปป์ไลน์ เนื่องจากการใช้เทคนิคไปป์ไลน์แต่ละสเตจจะทำงานพร้อมๆกัน ในการทำซ้ำแต่ละครั้งเวลาการทำงานจะเท่ากับค่าดีเลย์ของตัวดำเนินการที่มากที่สุด ทำให้เวลาในการทำงานเร็วขึ้นเมื่อใช้เทคนิคไปป์ไลน์



(ก) (ข)

ภาพประกอบ 3-4 สเตจการทำงานของคำสั่งวนรอบที่ใช้เทคนิคไปป์ไลน์

เมื่อทำการสังเคราะห์วงจรด้วยโปรแกรม Xilinx ISE เพื่อวิเคราะห์ทรัพยากรที่ใช้ในการทำคำสั่งวนรอบจากตัวอย่าง การทำงานตามคำสั่งวนรอบแบบตามลำดับมีการใช้จำนวน Slices ที่เป็นพื้นที่โดยรวมในการใช้งานเท่ากับ 156 ดังภาพประกอบที่ 3-5 ซึ่งใช้ทรัพยากรโดยรวมที่น้อยกว่าการทำคำสั่งวนรอบที่ใช้เทคนิคไปป์ไลน์ที่มีการใช้จำนวน Slices เท่ากับ 214 ดังภาพประกอบ 3-6 เนื่องจากการทำงานของทั้งสองสเตจที่ต้องทำงานพร้อมๆกันจึงต้องใช้ทรัพยากรที่มากขึ้นแต่ได้ความเร็วในการทำงานมากขึ้นด้วย

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slices	156	13696	1%
Number of Slice Flip Flops	144	27392	0%
Number of 4 input LUTs	235	27392	0%
Number of bonded IOBs	40	416	9%
Number of GCLKs	1	16	6%

ภาพประกอบ 3-5 ผลการใช้ทรัพยากรของคำสั่งวนรอบแบบตามลำดับ

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	214	13696	1%
Number of Slice Flip Flops	214	27392	0%
Number of 4 input LUTs	340	27392	1%
Number of bonded IOBs	40	416	9%
Number of GCLKs	1	16	6%

ภาพประกอบ 3-6 ผลการใช้ทรัพยากรของคำสั่งวนรอบที่ใช้เทคนิคไปป์ไลน์

3.6.3 กระบวนการปรับปรุงการโปรแกรมให้มีความเหมาะสม

การดำเนินการพื้นฐานของตัวปรับปรุงการโปรแกรมให้มีความเหมาะสม ทำได้โดยการแบ่งการโปรแกรมให้เป็นบล็อคย่อยๆ และจัดการแต่ละบล็อกให้ดำเนินการในหนึ่งสเตจหรือมากกว่า เพื่อทำให้เป็นกระบวนการแบบขนาน

โดยทั่วไปแต่ละสเตจ ส่วนของฮาร์ดแวร์จะทำงานโดยใช้ไซเคิลสัญญาณนาฬิกาเพียงหนึ่งสัญญาณ ทำได้ก็ต่อเมื่อสเตจที่มีการรอข้อมูลจากแหล่งข้อมูลภายนอก เช่น สตรีมหรือหน่วยความจำ ซึ่งสเตจนั้นสามารถชะลอความเร็วสำหรับหนึ่งไซเคิลสัญญาณนาฬิกาหรือมากกว่า จนกระทั่งข้อมูลมาถึง สเตจ การที่สเตจถูกดำเนินการแต่ละครั้งจะสร้างตัวควบคุมฮาร์ดแวร์เพื่อให้ทราบว่าสเตจใดกำลังดำเนินการ และสเตจใดที่จะต้องดำเนินการในลำดับถัดไป

การปรับปรุงการโปรแกรมให้มีความเหมาะสม ประกอบด้วย 4 ส่วนตามลำดับดังนี้ ส่วนแรก คือ unroll inner loops เป็นการนำบล็อกของการโปรแกรมแบบดั้งเดิมมาโปรแกรมใหม่เป็นโครงสร้างของการโปรแกรมแบบขนาน ส่วนที่สองกำหนดสเตจของการทำงาน เป็นการนำส่วนของโปรแกรมมาสร้างเป็นบล็อก แต่ละการดำเนินการภายในบล็อคจะถูกจัดการเป็นสเตจที่คำนวณแบบขนาน ส่วนที่สามทำให้การหน่วงในสเตจมีความสมดุลกันสำหรับสเตจที่ใช้เทคนิคไปป์ไลน์ และส่วนสุดท้ายพิจารณาข้อจำกัดของทรัพยากรและหาการทำซ้ำร่วมกัน เพื่อให้ทราบว่าการทำซ้ำในรอบถัดไปจะสามารถทำขนานกับรอบปัจจุบันได้หรือไม่

3.6.4 การปรับปรุงการโปรแกรมให้มีความเหมาะสมในระดับนิพจน์

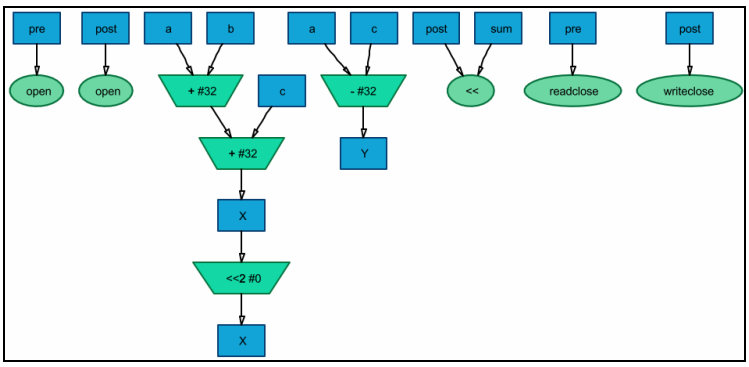
ตัวปรับปรุงการโปรแกรมให้มีความเหมาะสมจะคำนวณการโปรแกรมแบบขนานได้อย่างอัตโนมัติในหลายๆนิพจน์รวมทั้งสมการ ดังตัวอย่าง

$$X = a+b+c;$$

$$X = X<<2;$$

$$Y = a - c;$$

จากสมการแรก จะต้องใช้ตัวดำเนินการบวกสองตัว สมการที่สองใช้ตัวดำเนินการเลื่อน(shift operation) และตัวดำเนินการลบในสมการที่สาม สำหรับหนึ่งสเตจ ดังภาพประกอบ 3-7



ภาพประกอบ 3-7 สเตจการทำงานแบบตามลำดับ

แต่ถ้ามีตัวดำเนินการมากมายในหนึ่งสเตจ การลดค่าความเร็วสูงสุดของสัญญาณนาฬิกาเป็นสิ่งที่ควรตระหนัก เนื่องจากแต่ละสเตจมีเพียงหนึ่งไซเคิล โดยไม่คำนึงถึงจำนวนลอจิก จึงควรใช้อีกทางเลือก คือ การใช้การหน่วงของสเตจ(StageDelay) เพื่อหาค่าการหน่วงสูงสุด หรือหาค่าประมาณของการหน่วงของเกทซึ่งกระทำได้สำหรับหนึ่งสเตท

3.6.5 การปรับปรุงการโปรแกรมให้มีความเหมาะสมของบล็อกพื้นฐาน

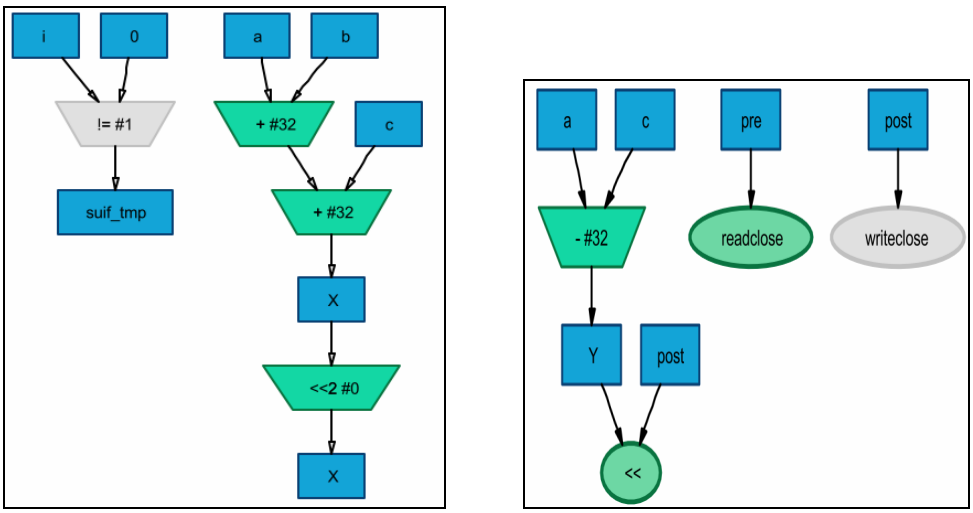
บล็อกพื้นฐาน (basic block) คือ ส่วนของบล็อกที่ติดกัน หรือในแต่ละบรรทัดที่ติดกันของการโปรแกรม โดยไม่มีส่วนการควบคุมเข้ามาเกี่ยวข้อง เช่น คำสั่งการวนรอบ if หรือ while จากตัวอย่างเดิม

```
X = a+b+c;
X = X<<2;
Y = a – c;
```

สามารถเขียนได้ใหม่เป็น

```
while(i) {
X= a+b+c ;
X= X<<2;
}
Y = a-c;
```

การปรับปรุงโปรแกรมให้เหมาะสม จะสร้าง (a+b+c) <<2 เป็นสเตจที่หนึ่ง ดังภาพประกอบ 3-8 (ก) จากนั้นจะเป็นสเตจที่สองของการคำนวณ a – c ดังภาพประกอบ 3-8 (ข)



(ก) (ข)

ภาพประกอบ 3-8 สเตจการทำงานเมื่อปรับปรุงโปรแกรมให้มีความเหมาะสม

3.6.6 การลดหน่วยความจำเพื่อระบบที่มีประสิทธิภาพสูง (Reduce Memory Accesses for Higher Performance)

สิ่งสำคัญที่ควรพิจารณาเมื่อมีการเขียนโปรแกรมแบบการวนซ้ำสำหรับการโปรแกรมแบบขนานนั้น คือ การพิจารณาข้อมูล ซึ่งโดยทั่วไปแล้วตัวปรับปรุงการโปรแกรมให้มีความเหมาะสมไม่สามารถสร้างสเตจการทำงานแบบขนานได้โดยตรง ดังนั้น จะต้องลดขนาดของอะเรย์ขนาดใหญ่ ไปเก็บในหน่วยความจำภายในภายในเอฟพีจีเอ (local storage) ที่มีขนาดอะเรย์ที่เล็กลงเสียก่อน แล้วคำนวณข้อมูลภายในเอฟพีจีเอที่เป็นแบบขนาน ซึ่งจะทำให้การโปรแกรมแบบขนานมีประสิทธิภาพมากยิ่งขึ้น

3.6.7 วิธีการแยกอะเรย์ (Array Splitting)

เมื่อมีการจัดการกับหน่วยความจำ เช่น อะเรย์ที่อยู่ภายในเอฟพีจีเอ สามารถทำให้มีเกิดผลกระทบเกิดขึ้นมากมาย การปรับปรุงการโปรแกรมให้มีความเหมาะสม เพื่อเป็นการจำกัดสเตจของการโปรแกรมแบบขนานสามารถกระทำได้ตัวอย่างเช่น X= A[0]+A[1]+A[2]; X= X<<2; จากตัวอย่างอะเรย์ A ที่เก็บไว้ในหน่วยความจำภายในบล็อคแรม(block ram)ของเอฟพีจีเอ การอ่านข้อมูลจากหน่วยความจำใช้หนึ่งไซเคิล ส่วนของการคำนวณแบ่งเป็น 4 สเตจ โดยสเตจที่หนึ่งทำการอ่านค่า A[0] สเตจที่สองทำการอ่านค่า A[1] สเตจที่สามอ่านค่า A[2] และทำการบวก A[0] กับ A[1] และสเตจสุดท้ายบวกค่า A[2] เข้ากับผลบวกในสเตจที่สามแล้วทำการเลื่อน

เพื่อหลีกเลี่ยงปัญหาที่จะเกิดกับหน่วยความจำโดยเพิ่มจำนวนมิติอะเรย์จะได้เป็น

```
int a[4][10];
```

```
for(i=o;i<10;i++)
a[3][i] = a[0][i] + a[1][i] + a[2][i];
```

จากนั้น นำอะเรย์ a[4][10] มาแยกบล็อกเพื่อแยกเป็นบล็อกแรม จะได้เป็น

```
int a0[10] ,a1[10] ,a2[10] ,a3[10];
for(i=o;i<10;i++)
    a3[i] = a0[i] + a1[i] + a2[i];
```

เมื่อทำการแยกบล็อกแรม ทำให้สามารถอ่านและเขียนข้อมูลได้พร้อมๆกัน โดยใช้เพียง สเตจเดียวแทนที่จะใช้สี่สเตจที่ไม่มีการแยกส่วนของอะเรย์ ซึ่งวิธีการนี้เป็นที่นิยมในการเพิ่มการโปรแกรมแบบขนาน แต่ผู้เขียนโปรแกรมก็ต้องเข้าใจในการใช้เทคนิคนี้เป็นอย่างดีเนื่องจากไม่มีตัวปรับปรุงการโปรแกรมของเทคนิคนี้อย่างอัตโมมัติ

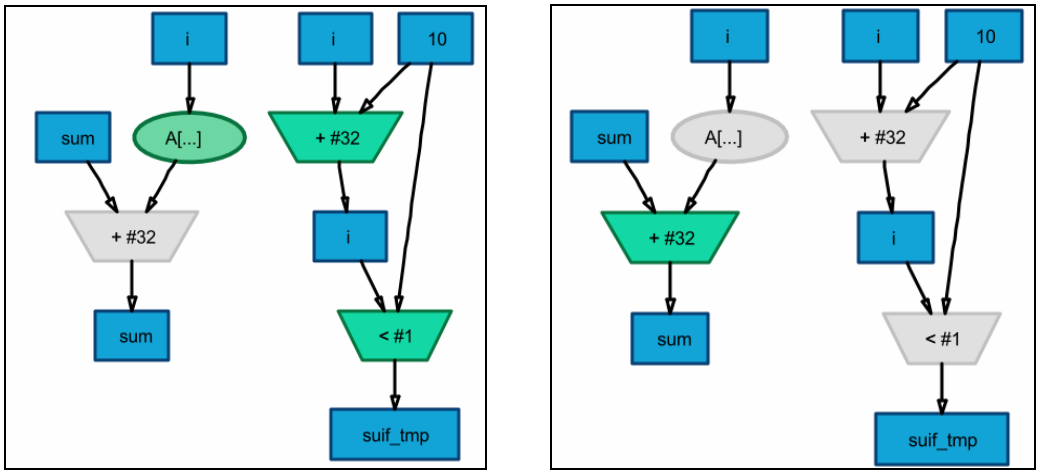
3.6.8 Unrolling Loops

ตัวปรับปรุงการโปรแกรมให้มีความเหมาะสม สามารถโปรแกรมแบบขนานสำหรับการทำซ้ำภายในคำสั่งวนซ้ำได้อย่างอัตโนมัติ Loop unrolling จึงเป็นอีกหนึ่งเทคนิคในการโปรแกรมแบบขนาน โดยแปลงการวนรอบการทำซ้ำเป็น บล็อกพื้นฐานหนึ่งบล็อค

โดยถ้ารู้จำนวนในการวนรอบ ซึ่งไม่ต้องมีการวนซ้ำ และทำการสร้างลอจิกเฉพาะสำหรับแต่ละการทำซ้ำ โดยจำลองส่วนของคำสั่งภายในคำสั่งการวนรอบได้หลายๆครั้ง ในการทำซ้ำของการวนรอบ ดังตัวอย่าง

```
for(i=0 ; i<10 ; i++)
    sum + = A[i];
```

จากตัวอย่างการวนซ้ำนี้ จะสร้างลอจิกสำหรับแต่ละรอบของการทำซ้ำ เป็นสองไซเคิล ซึ่งไซเคิลแรกจะทำการอ่านข้อมูลจากหน่วยความจำ ดังภาพประกอบ 3-9 (ก) และไซเคิลที่สองทำการคำนวณส่วนของการบวก ดังภาพประกอบ 3-9 (ข) ตัวดำเนินการบวกหนึ่งตัวจะถูกใช้ทั้งหมด สิบครั้งระหว่างการดำเนินการภายในการวนรอบ



(ก) (ข)

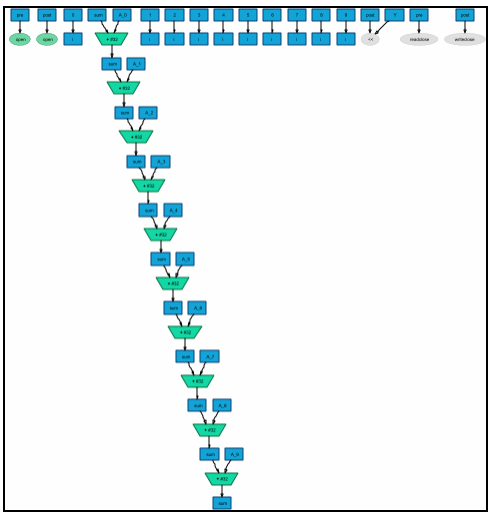
ภาพประกอบ 3-9 สเตจการทำงานของคำสั่งการวนรอบแบบตามลำดับ

ซึ่งการใช้เทคนิคนี้ ทำได้โดยการใช้คำสั่ง #pragma CO UNROLL ในส่วนคำสั่งภายในการวนรอบ ดังตัวอย่าง

```
for(i=0 ; i<10 ; i++){
    #pragma CO UNROLL
    sum + = A[i];
    }
```

ทำให้การทำงานเปลี่ยนไปโดยที่จะจำลองคำสั่งทุกค่าที่มีอยู่ในการวนรอบ ดังตัวอย่างและภาพประกอบ 3-10

```
sum + = A[0];
sum + = A[1];
sum + = A[2];
sum + = A[3];
sum + = A[4];
sum + = A[5];
sum + = A[6];
sum + = A[7];
sum + = A[8];
sum + = A[9];
```



ภาพประกอบ 3-10 สเตจการทำงานเมื่อใช้เทคนิค Unrolling Loops

เมื่อใช้เทคนิค Unrolling Loops ทำให้มีการใช้ตัวดำเนินการบวกจำนวนสิบตัว ที่ทำงานไปพร้อมๆกัน ซึ่งเมื่อพิจารณาเวลาในการคำนวณ เมื่อการทำคำสั่งวนรอบแบบตามลำดับใช้เวลาในการคำนวณ 5.835 ns. และใช้พื้นที่ทรัพยากรโดยรวม 140 ผลจากการสังเคราะห์วงจรจากโปรแกรม Xilinx ISE ดังภาพประกอบ 3-11 ที่ใช้เวลาคำนวณนานกว่าการทำคำสั่งวนรอบที่ใช้เทคนิค Unrolling Loops ซึ่งใช้เวลาในการคำนวณ 2.829 ns. และพื้นที่ทรัพยากรโดยรวม 98 ภาพประกอบ 3-12

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	140	13696	1%
Number of Slice Flip Flops	114	27392	0%
Number of 4 input LUTs	204	27392	0%
Number of bonded IOBs	40	416	9%
Number of GCLKs	1	16	6%

```
Minimum period: 5.835ns (Maximum Frequency: 171.380MHz)
Minimum input arrival time before clock: 2.952ns
Maximum output required time after clock: 4.424ns
```

ภาพประกอบ 3-11 ผลการใช้ทรัพยากรของคำสั่งวนรอบแบบตามลำดับ

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	98	13696	0%
Number of Slice Flip Flops	84	27392	0%
Number of 4 input LUTs	125	27392	0%
Number of bonded IOBs	40	416	9%
Number of GCLKs	1	16	6%

```
Minimum period: 2.829ns (Maximum Frequency: 353.444MHz)
Minimum input arrival time before clock: 3.062ns
Maximum output required time after clock: 4.424ns
```

ภาพประกอบ 3-12 ผลการใช้ทรัพยากรของคำสั่งวนรอบเมื่อใช้เทคนิค Unrolling Loops

ซึ่งจากตัวอย่างเมื่อใช้เทคนิคนี้ จะทำให้มีการใช้ตัวดำเนินการบวกจำนวนสิบตัว แต่ละตัวใช้เพียงครั้งเดียว ซึ่งทำให้มีลอจิกจำนวนมากและควรปรับเปลี่ยนข้อมูลที่เป็นอะเรย์ให้เป็นข้อมูลแบบสเกลาร์ ที่สามารถอ่านข้อมูลได้พร้อมๆกัน แต่ก็มีสิ่งที่ควรระมัดระวัง ในกรณีที่ข้อมูลของการวนรอบแต่ละครั้งจะต้องรอข้อมูลก่อนหน้า ดังนั้น จะต้องมีการใช้การหน่วงของเวลาร่วมด้วย เนื่องจากต้องการใช้เพียงหนึ่งไซเคิล

3.7 สรุป

จากการศึกษาโมเดลการโปรแกรมแบบต่างๆบนเอฟพีจีเอ จะต้องสามารถนำมาประยุกต์ใช้กับเอฟพีจีเอเพื่อให้เอฟพีจีเอเป็นตัวประมวลผลแบบขนานตามโมเดลการโปรแกรมแบบขนานได้ ที่สามารถใช้คอมไพเลอร์ทูลที่มีอยู่ใน ImpulseC โดยต้องเข้าใจโมเดลการโปรแกรมของ ImpulseC เพื่อนำพัฒนาการโปรแกรมให้เป็นการโปรแกรมแบบขนาน โดยใช้วิธีการปรับปรุงการโปรแกรมให้มีความเหมาะสมภายใต้ทรัพยากรที่มีอยู่อย่างจำกัดบนเอฟพีจีเอ เพื่อให้ระบบประมวลผลการทำงานได้อย่างมีประสิทธิภาพมากขึ้น

บทที่ 4

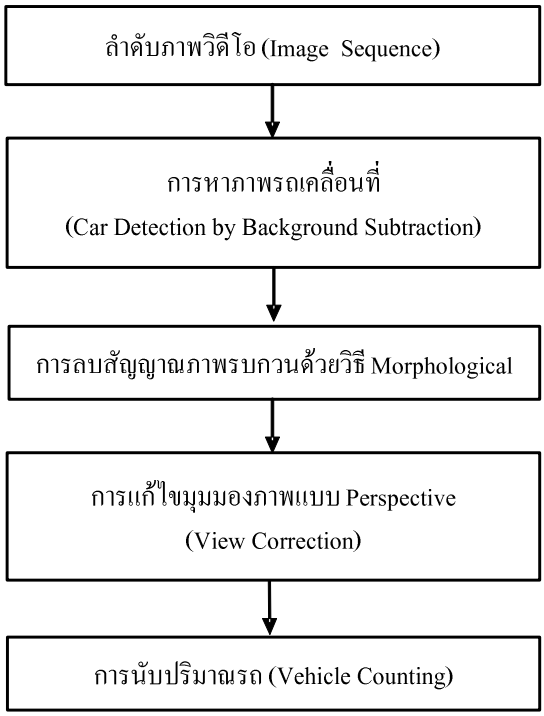
ระเบียบวิธีวิจัยการออกแบบวงจรสำหรับการนับปริมาณรถ

ในบทนี้อธิบายการออกแบบอัลกอริทึมการนับปริมาณรถ เพื่อพัฒนาวงจรด้วยภาษาระดับสูงที่ชื่อว่า ImpulseC ให้ทำการนับปริมาณรถที่สามารถนำไปใช้งานร่วมกับเอฟพีจีเอ ที่ประกอบด้วยกระบวนการลบภาพพื้นหลัง การลดสัญญาณภาพรบกวนด้วยวิธี Morphological การแก้ไขมุมมองภาพแบบ Perspective และการนับปริมาณรถ จากนั้นเป็นการพัฒนาอัลกอริทึมการนับปริมาณรถบนเอฟพีจีเอที่เป็นแบบตามลำดับ การใช้เทคนิคไปป์ไลน์ และการใช้เทคนิคการโปรแกรมแบบขนาน

4.1 การออกแบบอัลกอริทึมของการนับปริมาณรถ

การพัฒนาวงจรการนับปริมาณรถโดยใช้กระบวนการอิมเมจโปรเซสซิง ด้วยภาษาระดับสูง ImpulseC เพื่อช่วยในการออกแบบและพัฒนาร่วมกันระหว่างซอฟท์แวร์และฮาร์ดแวร์ที่สามารถนำมาใช้งานร่วมกับเอฟพีจีเอ ดังนั้น การออกแบบระบบแบ่งเป็นสองส่วน คือ การออกแบบอัลกอริทึมการนับปริมาณรถ และการพัฒนาอัลกอริทึมการนับรถบนเอฟพีจีเอ

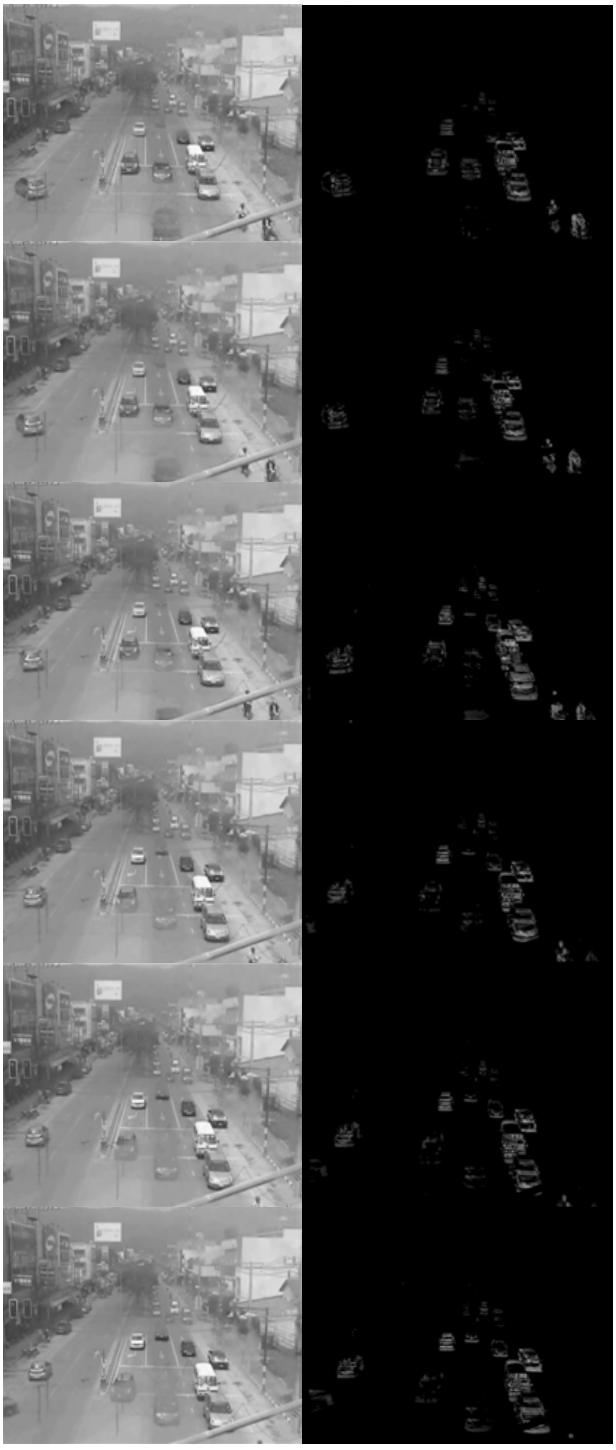
อัลกอริทึมการนับปริมาณรถโดยภาพรวม แสดงดังภาพประกอบ 4-1 เริ่มต้นแยกรถที่เคลื่อนที่ด้วยการลบภาพพื้นหลังโดยการหาผลต่างระหว่างเฟรม ผลที่ได้นำมาทำการลบสัญญาณรบกวนด้วยวิธี Morphological โดยใช้โอเปอร์เรชัน close จากนั้น การแก้ไขมุมมองภาพแบบ Perspective ของจุดสีในแต่ละแถว เพื่อขยายภาพรถในเลนถนนให้มีสัดส่วนเท่ากันทั่วทั้งภาพ และหาพื้นที่จำนวนจุดสีของรถทั้งหมดเพื่อทำการนับจำนวนรถ โดยเปรียบเทียบพื้นที่จำนวนจุดสีของรถทั้งหมดกับพื้นที่จำนวนจุดสีของรถ 1 คัน ซึ่งจะได้ปริมาณรถทั้งหมดใน 1 เฟรม



ภาพประกอบที่ 4-1 แผนภาพอัลกอริทึมการนับรถ

4.1.1 การหาภาพรถที่เคลื่อนที่

จากการศึกษาการหาภาพรถเคลื่อนที่ด้วยวิธีลบภาพพื้นหลังมีหลากหลายวิธีให้เลือกใช้ ซึ่งในงานวิจัยนี้ เลือกใช้เทคนิคการลบภาพพื้นหลัง ด้วยวิธีหาผลต่างของเฟรม [2] เนื่องจากภาพพื้นหลังค่อนข้างคงที่และประมวลผลได้อย่างรวดเร็ว ซึ่งวิธีนี้เป็นการแยกความแตกต่างของจุดสี ระหว่างรถที่เคลื่อนที่และภาพพื้นหลัง วิธีนี้จะใช้ภาพพื้นหลังที่ประมาณค่าได้เป็นเฟรมก่อนหน้าเพื่อง่ายต่อการประมวลผล สำหรับขั้นตอนการประมวลผลจำเป็นจะต้องทำให้ภาพเป็นเกรสเกลเพื่อลดหน่วยความจำที่ใช้ ซึ่งใช้ 8 บิต ที่มีค่าของจุดสีในช่วง 0-255 และนำค่าจุดสีในแต่ละลำดับเฟรมภาพมาคำนวณตามสมการ (2-1) ที่กล่าวมาแล้วในบทที่ 2 การหาผลต่างของเฟรมปัจจุบันกับเฟรมก่อนหน้าของลำดับภาพวิดีโอเพื่อให้ได้ภาพรถที่เคลื่อนที่ในแต่ละเฟรมที่เข้ามายังระบบ ตัวอย่างดังภาพประกอบ 4-2



ภาพประกอบ 4-2 ตัวอย่างการหาผลต่างของเฟรมจากลำดับภาพวิดีโอ

จากภาพประกอบ 4-2 เป็นภาพแสดงตัวอย่างผลการหาผลต่างของเฟรมจากลำดับภาพประกอบ 4-2 (ซ้าย) เป็นตัวอย่างภาพอินพุตของระบบซึ่งเป็นภาพจากการจราจรจริง ทำให้เกิดปัญหาบางประการของภาพที่ได้มา เช่น การสั่นไหวของกล้องในบางครั้งเนื่องจากขนาดของกล้อง

ที่มีขนาดเล็ก และสภาพกล้องที่ขุ่นมัวทำให้ภาพที่ได้จากกล้องในบางบริเวณไม่ถูกต้องจนรายละเอียดของรถบางส่วนหายไป สังเกตได้ว่าในภาพอินพุตบริเวณด้านล่างของภาพมีลักษณะขุ่นมัวจนเห็นรายละเอียดของรถไม่ชัดเจนจนบางส่วนของรถหายไป หรือในกรณีที่รถเป็นสีเข้ม เช่น สีเทาหรือสีดำ จะยิ่งทำให้รายละเอียดของรถน้อยลงมาก

ส่วนภาพประกอบ 4-2 (ขวา) เป็นผลจากการหาผลต่างของเฟรมที่นำค่าจุดสีของเฟรมปัจจุบันกับเฟรมก่อนหน้าในตำแหน่งเดียวกันมาทำการลบกันดังที่กล่าวมาแล้วนั้น ผลที่ได้จะเป็นภาพรถเคลื่อนที่ของเฟรมปัจจุบัน สำหรับในกรณีที่รถมีสีอ่อนตัดกับพื้นถนนจะทำให้มีรายละเอียดของผลลัพธ์ของภาพรถเคลื่อนที่ชัดเจน แต่ในกรณีที่รถมีสีเข้มใกล้เคียงกับพื้นถนนทำให้มีรายละเอียดของผลลัพธ์ของภาพรถเคลื่อนที่น้อยลงหรือหายไปบางส่วน และบริเวณที่ขุ่นมัวของกล้องดังที่กล่าวมารายละเอียดของผลลัพธ์ของภาพรถเคลื่อนที่น้อยลงจนบางส่วนของรถหายไปเช่นกัน ต้องมีการปรับปรุงภาพเพื่อเพิ่มรายละเอียดของภาพรถร่วมด้วย

การทำ Threshold เป็นสิ่งที่สำคัญใช้เพื่อแยกวัตถุเคลื่อนที่กับภาพพื้นหลัง และรายละเอียดของภาพวัตถุเคลื่อนที่ส่วนหนึ่งก็ขึ้นอยู่กับค่า Threshold ที่เลือก เช่น ถ้าเลือกค่า Threshold ที่มากเกินไปทำให้มีรายละเอียดของวัตถุน้อยลงจนไม่ชัดเจน แต่ถ้าเลือกค่า Threshold ที่น้อยเกินไปทำให้มีรายละเอียดของวัตถุมากขึ้นพร้อมๆ กับสัญญาณรบกวนที่มากขึ้นด้วย อย่างไรก็ตามการเลือก Threshold ก็มักจะมีสัญญาณภาพรบกวนเกิดขึ้น ต้องมีกระบวนการลดสัญญาณภาพรบกวนร่วมด้วย

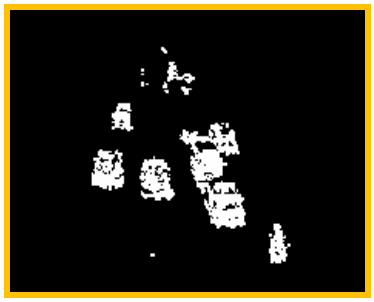
การทำ Threshold ทำได้โดยการแปลงภาพผลลัพธ์จากขั้นตอนที่ 1 ที่เป็นภาพแบบเกรสเกล เป็นแบบภาพไบนารีที่ทุกๆจุดสีในเฟรมภาพมีค่าเฉพาะ 0 กับ 1 โดยที่ค่าจุดสีที่เป็น 0 เป็นจุดที่ภาพจะแสดงผลเป็นสีดำซึ่งเป็นภาพพื้นหลังและค่าจุดสีที่เป็น 1 เป็นจุดที่ภาพจะแสดงผลเป็นสีขาวซึ่งเป็นภาพวัตถุเคลื่อนที่ ซึ่งการพิจารณาค่าจุดสีที่เป็นเกรสเกลเป็นค่า 0 หรือ 1 นั้น จะต้องนำค่าเกรสเกลจากทุกจุดสีนั้นมาเทียบกับค่า Threshold ค่าหนึ่ง เพื่อเลือกจุดสีนั้นว่าเป็นวัตถุเคลื่อนที่หรือไม่ ซึ่งถ้าค่าจุดสีเกรสเกลนั้นมีค่ามากกว่าค่า Threshold ก็แสดงว่าจุดสีนั้นเป็นวัตถุเคลื่อนที่ และถ้าจุดสีนั้นน้อยกว่าค่า Threshold ก็แสดงว่าจุดสีนั้นเป็นภาพพื้นหลัง ตัวอย่างผลที่ได้จากการทำ Threshold ดังภาพประกอบ 4-3

ภาพประกอบ 4-3 บริเวณที่เลือกนับปริมาณรถและเทียบกับค่า Threshold

จากภาพประกอบที่ 4-3 เป็นผลจากการหาผลต่างของเฟรมภาพปัจจุบันกับภาพก่อนหน้า โดยค่า Threshold ที่เลือกเป็นค่าเดียว คือ 25 เนื่องจากการจราจรในไฟล์ภาพวิดีโอเป็นช่วงเวลาเพียงสั้นๆประมาณ 10 นาที ที่ไม่มีความแตกต่างของแสงมากนัก ที่ทำให้มีรายละเอียดของภาพชัดเจนแต่เกิดสัญญาณภาพรบกวนเล็กน้อย และการลบกันทั้งภาพทุกตำแหน่งจะทำให้สิ้นเปลืองทรัพยากรหน่วยความจำ เนื่องจากบริเวณที่ต้องการนับปริมาณรถอยู่ในบริเวณเลนถนนทางด้านขวาเท่านั้น ดังนั้น การคำนวณของภาพจะเลือกเฉพาะบริเวณที่ต้องการนับปริมาณของรถเท่านั้น

4.1.2 การลดสัญญาณภาพรบกวนด้วยวิธี Morphological

การลดสัญญาณภาพรบกวนด้วยวิธี Morphological โดยใช้โอเปอร์เรชัน Dilation Erosion Close และ Open สำหรับลดสัญญาณภาพรบกวนของภาพที่ไม่ต้องการออก และปิดช่องว่างของรถที่มีรายละเอียดหายไป ซึ่งโอเปอร์เรชัน Dilationใช้สำหรับขยายภาพให้มีขนาดใหญ่ขึ้น ตามลักษณะของเทมเพลตเมตริกซ์ที่เรากำหนด ตรงกันข้ามกับโอเปอร์เรชัน Erosionใช้สำหรับทำให้วัตถุในภาพมีขนาดลดลงตามลักษณะของเทมเพลตเมตริกซ์ที่เรากำหนด ส่วนโอเปอร์เรชัน Openเป็นการนำเทมเพลตเมตริกซ์มาทำโอเปอร์เรชัน Erosionก่อน แล้วจึงตามด้วยโอเปอร์เรชัน Dilation ซึ่งจะทำให้วัตถุบริเวณมุม จุด หรือเส้นที่มีขนาดเล็กกว่าเทมเพลตเมตริกซ์หายไป และโอเปอร์เรชัน Closeเป็นการนำเทมเพลตมาทำโอเปอร์เรชัน Dilation ก่อน แล้วจึงตามด้วยโอเปอร์เรชัน Erosion ซึ่งจะเติมเต็มวัตถุส่วนที่เว้าแหว่งที่มีขนาดเล็กกว่าเทมเพลตเมตริกซ์ ตัวอย่างการลดสัญญาณภาพรบกวนด้วยวิธี Morphological ดังภาพประกอบ 4-4



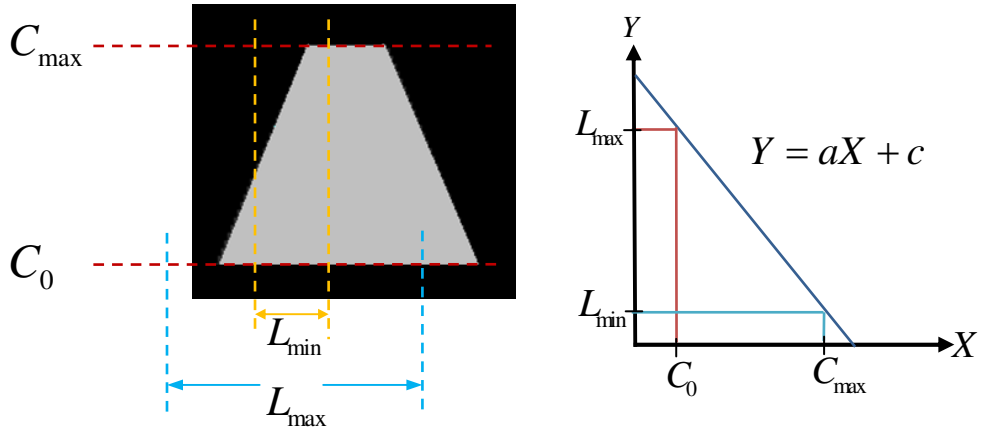
ภาพประกอบ 4-4 การลดสัญญาณภาพรบกวนด้วยวิธี Morphological

จากภาพประกอบ4-4 ที่มีช่องว่างของรถจะต้องนำมาทำกระบวนการ Morphological โดยใช้โอเปอร์เรชันCloseและโอเปอร์เรชันOpen สำหรับการเลือกใช้เทมเพลตเมตริกซ์สำหรับแต่ละโอเปอร์เรชันเป็นสิ่งที่สำคัญที่จะต้องพิจารณาพื้นที่จุดสีของรถเคลื่อนที่แต่ละ

แถวด้วย เนื่องจากถ้าพื้นที่ของจุดสีมีขนาดเล็กกว่าเทมเพลตเมตริกซ์จุดสีนั้นจะหายไปได้ เมื่อพิจารณาจำนวนพื้นที่จุดสีที่น้อยที่สุดของรถเคลื่อนที่มีขนาด 3 จุดสี จึงต้องเลือกเทมเพลตเมตริกซ์ขนาด 2x2 สำหรับแต่ละโอเปอร์เรชันเพื่อลดสัญญาณรบกวนและปิดช่องว่างรถ

4.1.3 การแก้ไขมุมมองภาพแบบ Perspective

เนื่องจากมุมมองของภาพจากกล้องวิดีโอที่เป็นแบบ Perspective ดังภาพประกอบ 4-5(ก) รถที่อยู่บริเวณด้านหน้ากล้องจะมีขนาดใหญ่ และจะมีขนาดเล็กลงเรื่อยๆ เมื่อรถไกลจากกล้องวิดีโอ ซึ่งจำนวนจุดสีของรถ 1 คัน จะมีขนาดลงลงด้วยเมื่อรถอยู่ไกลจากกล้องออกไป ที่มีความสัมพันธ์เป็นไปในลักษณะเชิงเส้น ดังนั้นจึงต้องหานอร์มอลไลซ์แฟคเตอร์ที่จะต้องนำมาคูณกับจำนวนพิกเซลในแต่ละแถวของบริเวณที่ต้องการนับรถ เพื่อเป็นการขยายภาพของรถในบริเวณที่อยู่ไกลกล้องออกไปให้เป็นสัดส่วนเดียวกันทั้งหมด



(ก) (ข)

ภาพประกอบ 4-5 บริเวณที่ต้องการนับปริมาณรถนำค่านำค่าจุดสีพล็อตกราฟเส้นตรง (ก) กำหนดตัวแปรในบริเวณที่ต้องการนับ และ (ข) นำค่าจุดสีมาพล็อตกราฟเส้นตรง

จากภาพประกอบ 4-5(ก) คือ มุมมองของภาพจากกล้องในบริเวณที่ต้องการนับปริมาณรถที่เป็นภาพแบบ Perspective ค่า $C_0 - C_{max}$ คือ ค่าของจำนวนแถวจากบริเวณภาพที่ใกล้กล้องที่สุดจนถึงบริเวณที่ไกลกล้องที่สุด และ $L_{min} - L_{max}$ คือ ค่าจำนวนจุดสีของรถเคลื่อนที่ในแต่ละแถวของพื้นที่บริเวณที่ต้องการ และสามารถหาความสัมพันธ์สมการเส้นตรง

$$Y = aX + c \tag{3-2}$$

เมื่อนำบริเวณของพื้นที่ที่เลือกมาพล็อตกราฟได้ดังภาพประกอบ 4-5 (ข) ซึ่งค่าจำนวนจุดสีในแต่ละแถวนำมาพล็อตในแกน Y ส่วนจำนวนแถวของบริเวณที่เลือกในภาพจะนำมาพล็อตในแกน X จะได้เส้นตรงดังรูป ซึ่งสามารถนำมาเขียนสมการได้ดังสมการ (3-3)

$$Y = (\frac{L_{\min} - L_{\max}}{C_{\max}})X + L_{\max} \tag{3-3}$$

ดังนั้น จะได้นอร์มอลไลซ์แฟคเตอร์ ดังสมการ (3-4)

$$F_x = \frac{L_{\max}}{(\frac{L_{\min} - L_{\max}}{C_{\max}})X + L_{\max}} \tag{3-4}$$

เมื่อ $X = 0$ ถึง $C_{\max}$

จากนั้นนำค่า F_x ไปคูณกับจำนวนจุดสีของรถในแต่ละแถว แล้วนำค่าจุดสีใหม่มารวมกันทุกแถวจะได้บริเวณพื้นที่ของรถทั้งหมดที่มีการขยายพื้นที่ของรถในบริเวณที่ไกลจากกล้องให้มีขนาดเดียวกันเป็นสัดส่วนเดียวกันทั้งหมด

4.1.4 การนับปริมาณรถ

เมื่อได้พื้นที่โดยรวมของรถทั้งหมด ใน แต่ละเฟรมภาพ จะต้องนำมาเปรียบเทียบด้วยพื้นที่ของรถ 1 คัน ตามสมการ (3-5) และ (3-6)

$$Count = \frac{\sum_{i=C_0}^{C_{\max}} \sum_{j=L_0}^{L_{\max}} F_i * f(i,j)}{C_{area}} \tag{3-5}$$

เมื่อ C_{area} คือ พื้นที่เฉลี่ยของพื้นที่รถ 1 คันที่ถูกขยายให้มีสัดส่วนเดียวกันทั้งภาพแล้ว

และ $$f(i,j) = \begin{cases} 1 & \text{ถ้า } |F - B| > T \\ 0 & \text{กรณีอื่นๆ} \end{cases} \tag{3-6}$$

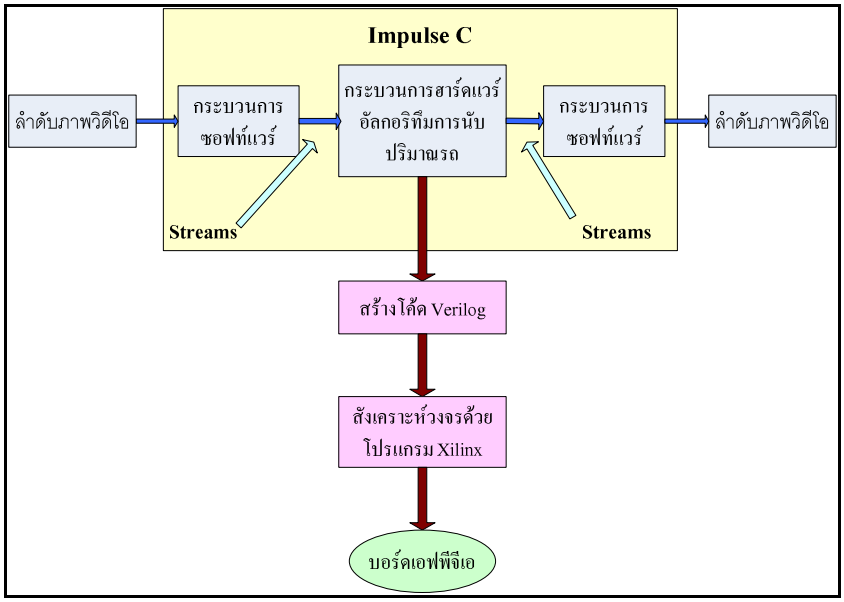
เมื่อ $Count$ คือ จำนวนรถในแต่ละเฟรมภาพ

F_i คือ นอร์มอลไลซ์แฟคเตอร์

$f(i,j)$ คือ จุดสีของภาพที่ตำแหน่ง (i,j)

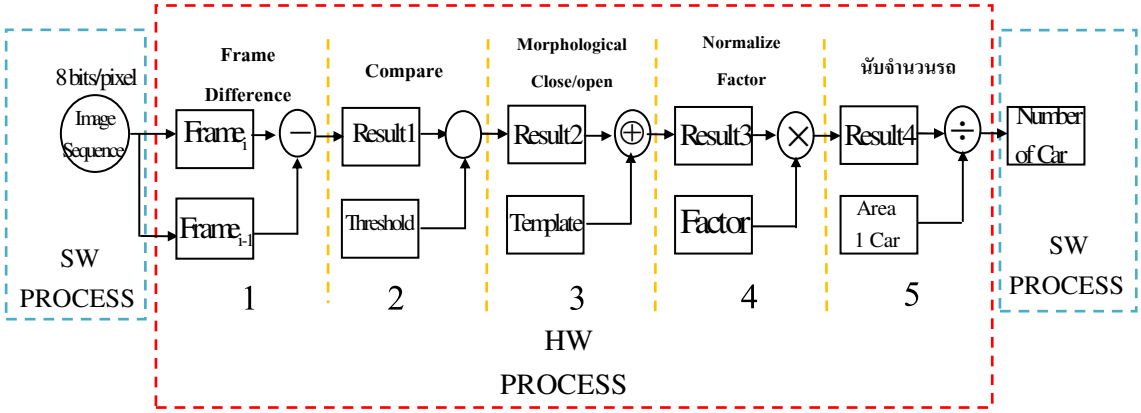
4.2 การพัฒนาอัลกอริทึมนับปริมาณรถบนเอฟพีจีเอ

การออกแบบและพัฒนาระบบร่วมฮาร์ดแวร์/ซอฟต์แวร์ [1] ของระบบนับปริมาณรถใช้ภาษา ImpulseC [18] โดยแบ่งออกเป็นสองส่วนหลักคือ ระบบซอฟท์แวร์ และระบบฮาร์ดแวร์ โดยมีพื้นฐานมาจากการเขียนด้วยภาษา C ซึ่งกระบวนการทำงานของ CoDeveloper [18] สามารถนำอัลกอริทึมในส่วนของระบบฮาร์ดแวร์มาสร้างในรูปของภาษาทางฮาร์ดแวร์ เป็นภาษา VHDL หรือ Verilog ได้ และสามารถทำการจำลองแบบการทำงานระดับระบบ (System – Level Simulation) ได้ดังภาพประกอบ 4-6



ภาพประกอบ 4-6 การออกแบบระบบ(ภาพรวม)

เมื่อพัฒนาการนับปริมาณรถด้วยภาษา ImpulseC แล้ว จากนั้นใช้โปรแกรม Impulse CoDeveloper แปลภาษาเพื่อให้ได้วงจรที่อธิบายด้วยภาษาทางฮาร์ดแวร์ ในที่นี้เลือกใช้เป็นภาษา Verilog จากนั้นใช้ซอฟต์แวร์ที่มีชื่อเรียกว่า Xilinx ISE [19] ช่วยในการสังเคราะห์วงจรและจำลองการทำงาน ซึ่งขั้นตอนการออกแบบวงจรบนเทคโนโลยีเอฟพีจีเอ สามารถอธิบายได้ตามไดอะแกรม ดังภาพประกอบ 4-7 เป็นภาพโดยรวมของการออกแบบวงจรนับปริมาณรถแบบตามลำดับ ที่มี 5 บล็อกเพื่อให้สามารถรองรับการทำงานแบบไปป์ไลน์ ส่วนในกรณีที่ไม่ใช้ไปป์ไลน์ ผลลัพธ์ Result1-5 ในภาพประกอบ 4-7 สามารถใช้หน่วยความจำตัวเดียวกันได้ เพื่อเป็นการประหยัดหน่วยความจำ



ภาพประกอบ 4-7 ไดอะแกรมอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับ

4.2.1 ออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอแบบตามลำดับ

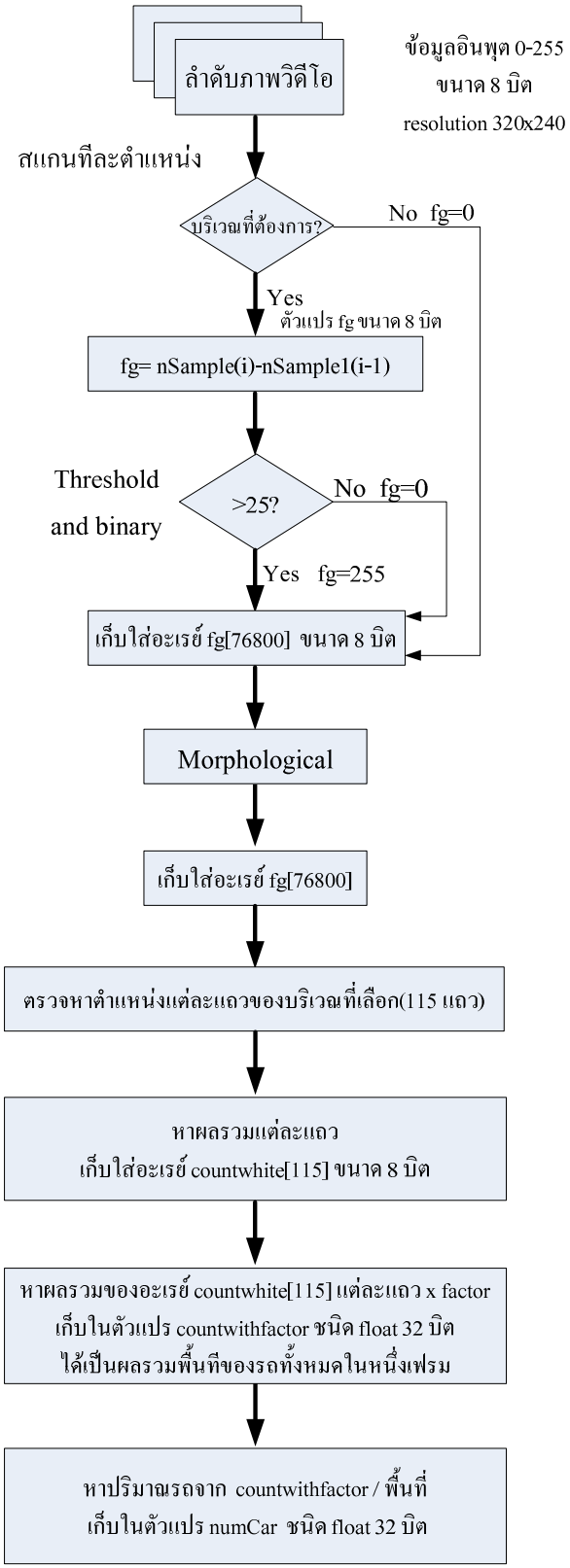
จากภาพประกอบ 4-7 ที่มีการใช้ทรัพยากรหน่วยความจำเดิมได้เพื่อไม่เป็นการสิ้นเปลืองทรัพยากร สามารถออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอแบบตามลำดับ ได้ดังนี้

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 1

การออกแบบวงจรนับปริมาณรถแบบตามลำดับ ดังภาพประกอบที่ 4-8 แสดงไดอะแกรมของอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิม ซึ่งเริ่มต้นลำดับภาพวิดีโอที่นำมาประมวลผลจะมีข้อมูลของแต่ละจุดสีเป็น 0-255 ขนาด 8 บิต ความละเอียด 320x240 เริ่มต้นกระบวนการแรกเป็นการหาค่าผลต่างของเฟรม จากนั้น นำผลต่างของเฟรมที่เก็บในตัวแปร fg ขนาด 8 บิตที่ได้มาเทียบกับค่า Threshold เพื่อทำภาพเกรสเกลเป็นภาพขาวดำ จากนั้นนำค่าที่ได้ไปเก็บไว้ในบัฟเฟอร์อะเรย์ fg หนึ่งมิติที่มีความกว้างเท่ากับจำนวนจุดสีทั้งหมด คือ 76800 ขนาด 8 บิต ซึ่งเป็นการเก็บทั้งภาพทั้งส่วนของบริเวณที่ต้องการนับรถจะแสดงเป็นภาพขาวดำ ส่วนบริเวณที่เหลือจะเป็นสีดำคือ ค่า 0 ทั้งหมด หลังจากนั้นอ่านภาพทั้งเฟรมเพื่อให้ทุกจุดสีเข้าสู่กระบวนการ Morphological ผลที่ได้ก็นำมาเก็บในบัฟเฟอร์อะเรย์ fg ตัวเดิม

จากนั้นทำการอ่านภาพทั้งเฟรมอีกครั้ง ทำการตรวจหาตำแหน่งของบริเวณที่ต้องการนับปริมาณรถพร้อมกับหาผลรวมของพื้นที่จุดสีของรถในแต่ละแถวที่มีทั้งหมด 115 แถว เก็บในบัฟเฟอร์อะเรย์อีกหนึ่งตัวที่มีความกว้าง 115 เท่ากับจำนวนแถวและมีขนาด 8 บิต เมื่อหาผลรวมของพื้นที่จุดสีของรถในแต่ละแถวแล้ว นำค่าพื้นที่จุดสีทุกๆแถวมาคูณด้วยค่านอมอลไลซ์แฟคเตอร์ ผลที่ได้เป็นพื้นที่ของจุดสีใหม่ของรถเคลื่อนที่ ที่มีสัดส่วนเดียวกันทั้งภาพ แล้วนำมาพื้นที่จุดสีทุกๆแถวมารวมกันทั้งหมด จะได้พื้นที่ทั้งหมดของปริมาณรถในเฟรมภาพ เก็บในตัวแปร

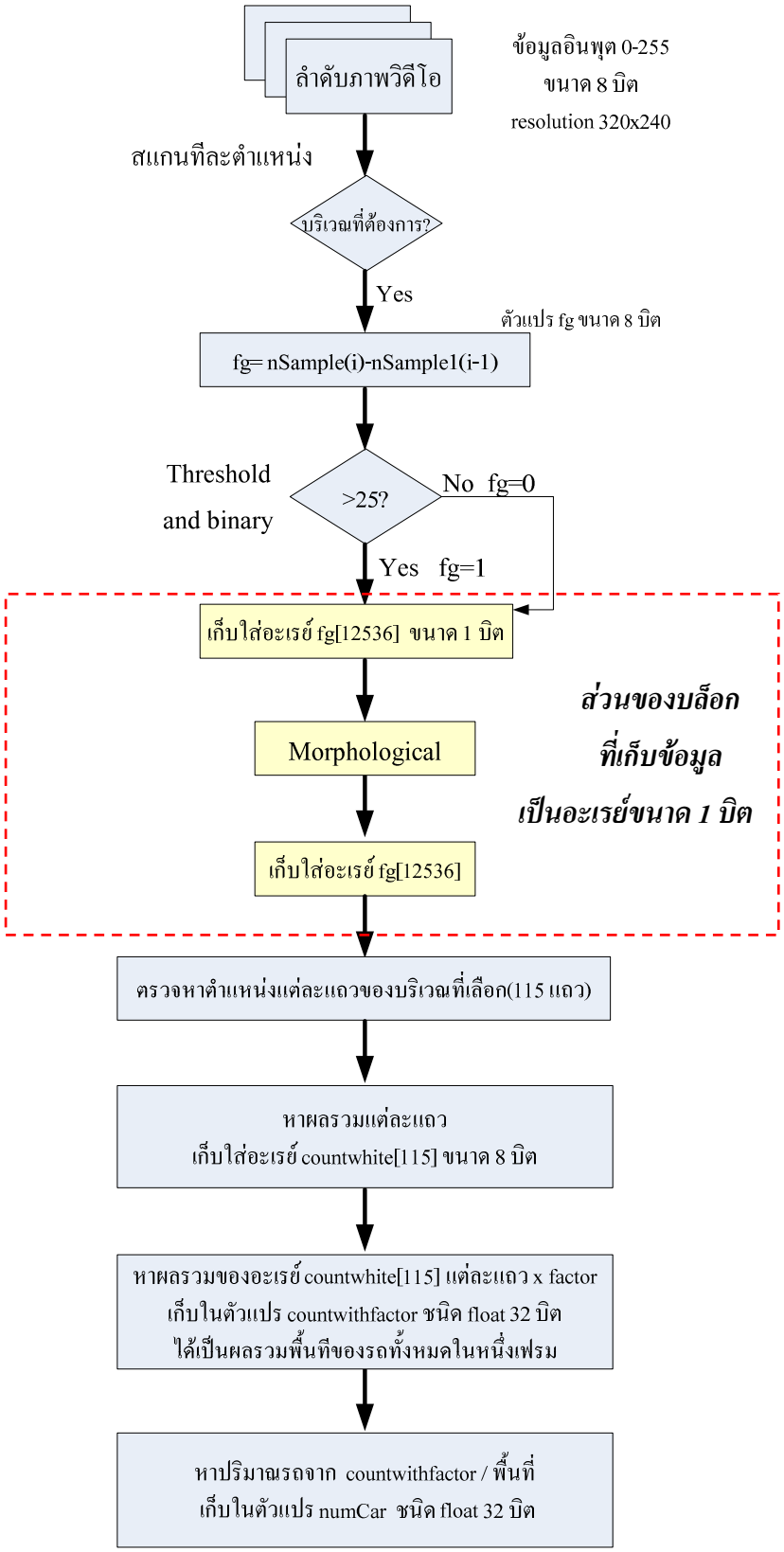
countwithfactor ชนิด float ขนาด 32 บิต แล้วนำมาหารด้วยพื้นที่ ผลที่ได้เป็นปริมาณรถที่อยู่ในเฟรม และเก็บในตัวแปร numCar ชนิด float ขนาด 32 บิต ซึ่งเป็นผลลัพธ์ของระบบ การทำงานของแต่ละบล็อกจะส่งผลลัพธ์ไปยังตัวถัดไปตามลำดับของไดอะแกรม



ภาพประกอบ 4-8 ไดอะแกรมของอัลกอริทึมการนับรถบนเอฟพีจีเอ
แบบตามลำดับและใช้หน่วยความจำเดิม Solution ที่ 1

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 2

การเก็บภาพทั้งภาพมาคำนวณนั้นทำให้สิ้นเปลืองทรัพยากรหน่วยความจำ จึงควรปรับปรุงกระบวนวิธีการเก็บข้อมูลในอะเรย์ จึงได้ลดขนาดของอะเรย์เป็นเฉพาะบริเวณที่ต้องการนับรถเท่านั้น เพื่อเป็นการลดขนาดบล็อกแรมของเอฟพีจีเอดังที่กล่าวมาในบทที่ 3 และข้อมูลภาพที่เข้าสู่ระบบมีขนาดเดิมที่กล่าวมาแล้วในการออกแบบ Solution ที่ 1 เริ่มต้นเมื่อวนลูปโปรแกรมทำการอ่านภาพทีละจุดสีของเฟรมภาพที่เข้ามาตามลำดับของตำแหน่งจุดสี และเข้าสู่กระบวนการหาผลต่างของเฟรมในบล็อกที่ 1 ผลที่ได้ในแต่ละรอบที่อ่านจุดสีเป็นค่าตัวเลข 8 บิต จะถูกเก็บในตัวแปร fg ขนาด 8 บิต แล้วนำค่าของตัวแปร fg มาเข้าสู่กระบวนการ Threshold ในบล็อกที่ 2 ผลลัพธ์จากการทำกระบวนการนี้ เป็นค่าจุดสีที่มีค่า 0 กับ 1 ที่นำมาเก็บในบัฟเฟอร์อะเรย์ fg ที่ลดขนาดลงได้เป็น 1 บิต และเก็บเฉพาะบริเวณพื้นที่ที่ต้องการนับรถเท่านั้น ระบบจะตรวจสอบตำแหน่งของจุดสีว่าเป็นบริเวณที่ต้องการนับปริมาณรถหรือไม่ ถ้าใช่บริเวณที่ต้องการจะทำการเก็บค่าจุดสีนั้นไว้ แต่ถ้าไม่ใช่บริเวณที่ต้องการก็จะไม่ทำการเก็บค่าจุดสีนั้น ซึ่งพื้นที่จุดสีเฉพาะในบริเวณที่ต้องการทั้งหมดมีจำนวน 12536 ซึ่งกระบวนการอื่นๆ ในลำดับถัดไปจะเป็นในบล็อกที่ 3-5 จะเป็นกระบวนการที่เหมือนกับ Solution ที่ 1 แต่การวนรอบครั้งถัดไปเพื่อทำกระบวนการ Morphological และหาพื้นที่ของรถในแต่ละแถวนั้น จะมีรอบการทำงานที่น้อยลงตามขนาดของอะเรย์ที่ลดลง ทำให้ใช้เวลาในการประมวลผลเร็วขึ้น ไดอะแกรมแสดงลำดับการทำงานแสดงดังภาพประกอบที่ 4-9



ภาพประกอบ 4-9 ไดอะแกรมอัลกอริทึมการนับรถบนเอฟพีจีเอ
แบบตามลำดับและใช้หน่วยความจำเดิมแต่ลดขนาดบัฟเฟอร์ที่เก็บภาพ Solution ที่ 2

4.2.2 ออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอแบบใช้เทคนิคไปป์ไลน์

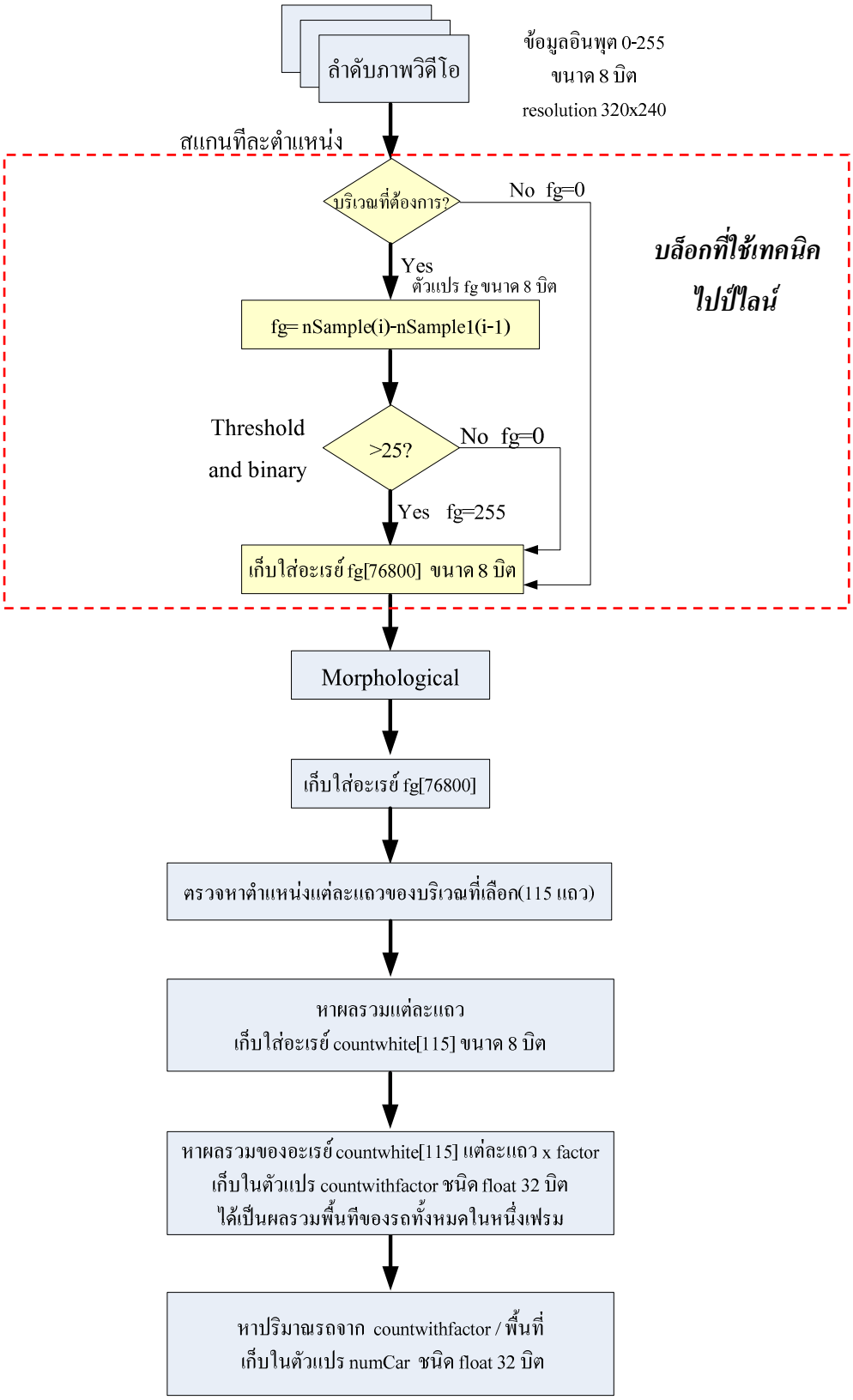
ไปป์ไลน์ คือ เทคนิคทำให้คำสั่งหลายๆ คำสั่งทำงานพร้อมๆ กัน แต่ละส่วนจะทำงานให้เสร็จในส่วนของมัน และทำงานต่างกัน การทำงานแต่ละส่วนเรียกว่า Stage ซึ่งแต่ละ Stage ทำงานไปพร้อมๆกัน

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 3

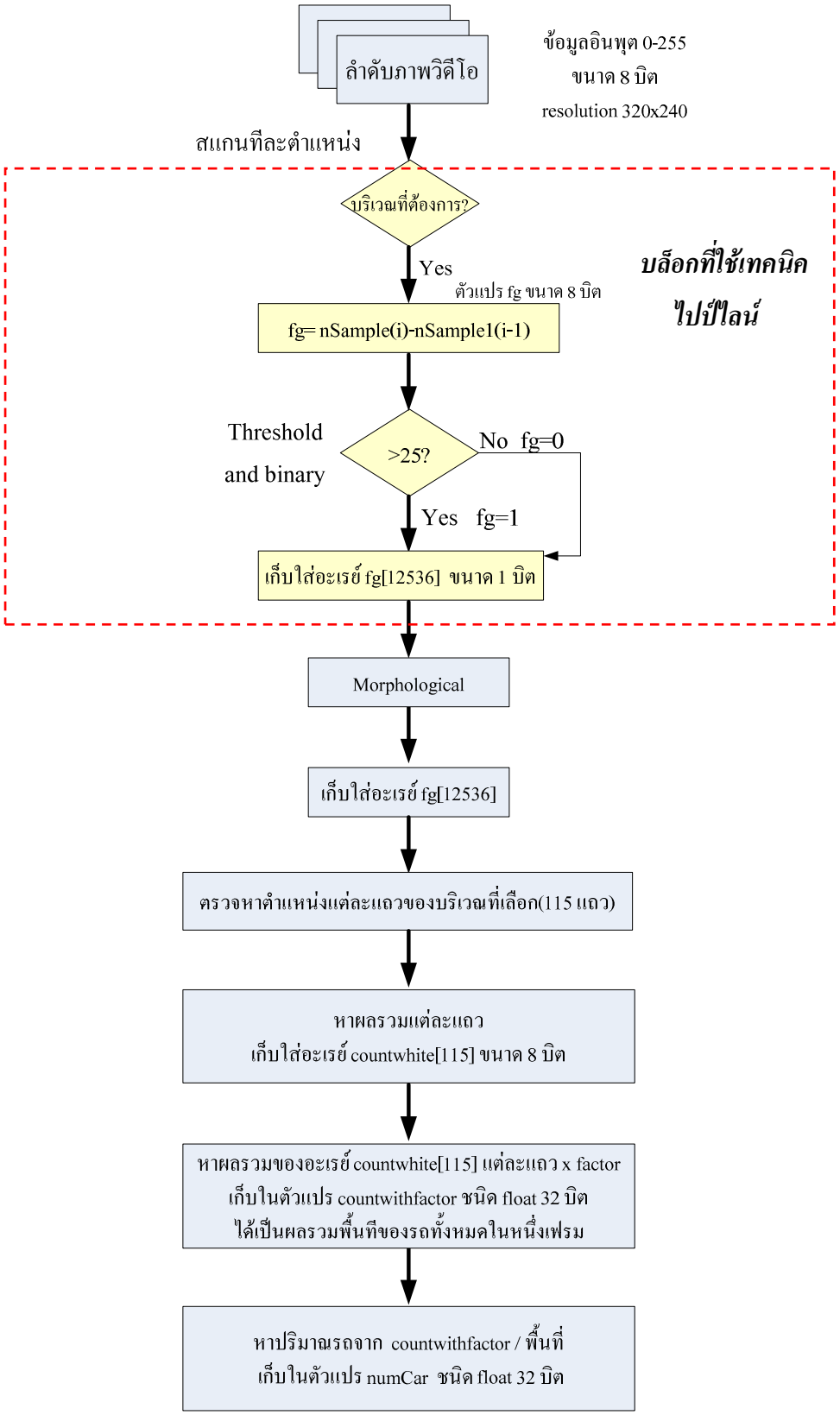
เป็นการนำเอาการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution 1 ที่เก็บข้อมูลด้วยอะเรย์ขนาด 8 บิต มาพัฒนาโดยใช้เทคนิคไปป์ไลน์ร่วมด้วยในส่วนบล็อกที่ 1 และบล็อกที่ 2 เนื่องจากในแต่ละรอบที่รับค่าจุดสีของเฟรมมาคำนวณจะเป็นการทำงานที่ซ้ำเหมือนเดิมในทุกๆรอบ และเป็นการคำนวณจากค่าจุดสีที่เป็นอิสระต่อกัน เหมาะสมที่จะนำมาใช้กับเทคนิคไปป์ไลน์ การทำงานในบล็อกที่ 1 เป็นการหาผลต่างของเฟรมและการทำงานในบล็อกที่ 2 เป็นทำกระบวนการ Threshold เพื่อเป็นการลดสเตจการทำงานจากออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution 1 กระบวนการทำงานแสดงดังภาพประกอบ 4-10

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 4

เป็นการนำเอาการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution 2 ที่เก็บข้อมูลด้วยอะเรย์ขนาด 1 บิต มาพัฒนาโดยใช้เทคนิคไปป์ไลน์ร่วมด้วยในส่วนบล็อกที่ 1 และบล็อกที่ 2 เนื่องจากในแต่ละรอบที่รับค่าจุดสีของเฟรมมาคำนวณจะเป็นการทำงานที่ซ้ำเหมือนเดิมในทุกๆรอบ และเป็นการคำนวณจากค่าจุดสีที่เป็นอิสระต่อกัน เหมาะสมที่จะนำมาใช้กับเทคนิคไปป์ไลน์ การทำงานในบล็อกที่ 1 เป็นการหาผลต่างของเฟรมและการทำงานในบล็อกที่ 2 เป็นทำกระบวนการ Threshold เพื่อเป็นการลดสเตจการทำงานจากออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution 2 กระบวนการทำงานแสดงดังภาพประกอบ 4-11



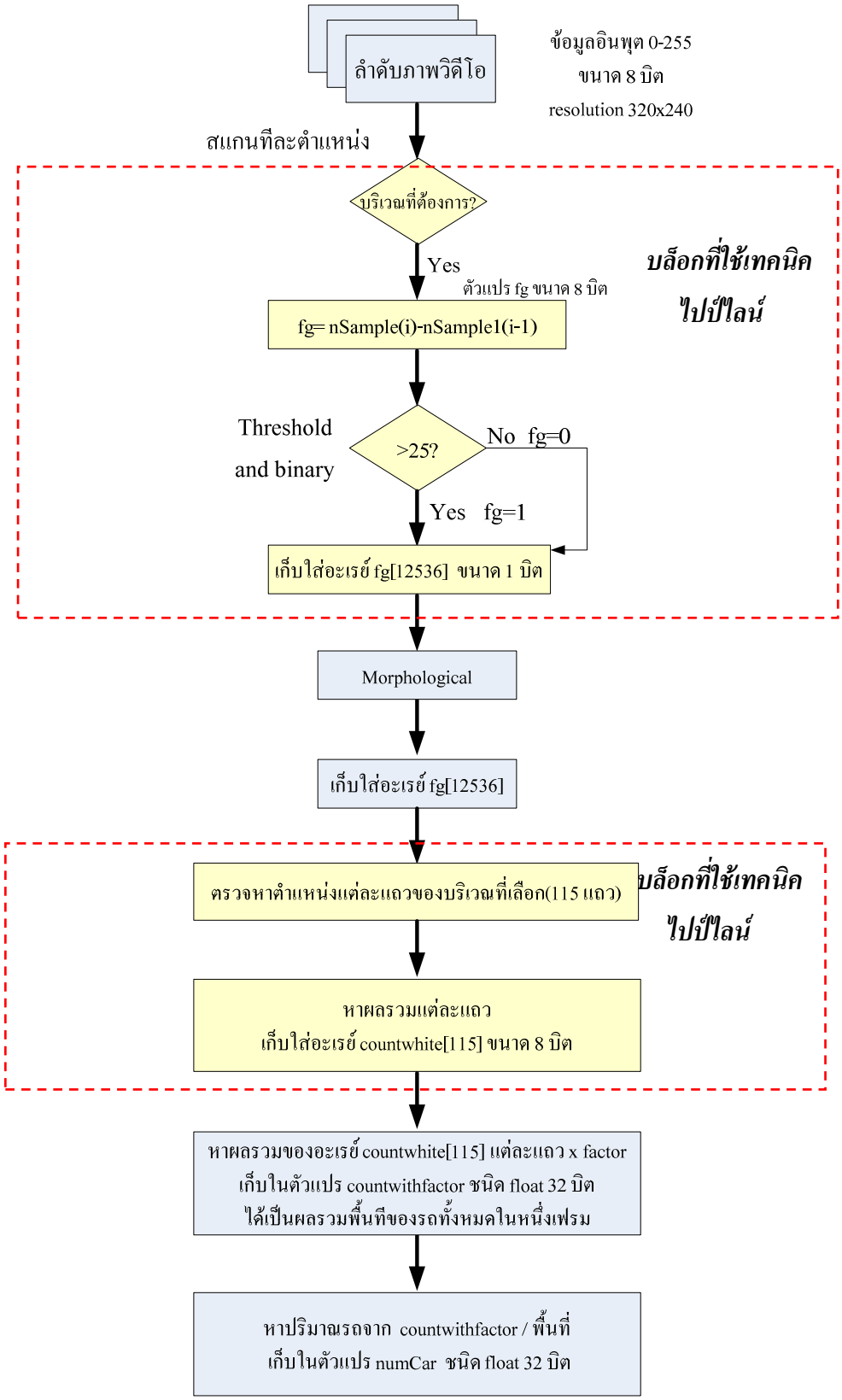
ภาพประกอบ 4-10 ไดอะแกรมอัลกอริทึมการนับรถบนเอฟพีจีเอ
ของการออกแบบ Solution ที่ 3 มีการใช้เทคนิคไปป์ไลน์ร่วมด้วย



ภาพประกอบ 4-11 ไดอะแกรมอัลกอริทึมการนับรถบนเอฟพีจีเอ
ของการออกแบบ Solution ที่ 4 มีการใช้เทคนิคไปป์ไลน์ร่วมด้วย

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 5

เป็นการนำเอาการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution 4 ที่เก็บข้อมูลด้วยอะเรย์ขนาด 1 บิตและใช้เทคนิคไปป์ไลน์ร่วมด้วยในส่วนบล็อกที่ 1 และบล็อกที่ 2 มาพัฒนาต่อโดยเพิ่มกระบวนการไปป์ไลน์ในลูปของบล็อกที่ 4 เพิ่มขึ้น เนื่องจากในแต่ละรอบที่รับค่าจุดสีของเฟรมมาคำนวณจะเป็นการทำงานที่ซ้ำเหมือนเดิมในทุกๆรอบ และเป็นการคำนวณจากค่าจุดสีที่เป็นอิสระต่อกัน เหมาะสมที่จะนำมาใช้กับเทคนิคไปป์ไลน์ การทำงานในบล็อกที่ 1 เป็นการหาผลต่างของเฟรมและการทำงานในบล็อกที่ 2 เป็นทำกระบวนการ Threshold และการทำงานของบล็อกที่ 4 เป็นการวนลูปของอะเรย์ fg เพื่อหาตำแหน่งแถวของบริเวณที่ต้องการนับปริมาณรถและหาผลรวมของพื้นที่จุดสีแต่ละแถวด้วย เพื่อเป็นการลดสเตจการทำงานจากออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 4 กระบวนการทำงานแสดงดังภาพประกอบ 4-12



ภาพประกอบ 4-12 ไดอะแกรมอัลกอริทึมการนับรถบนเอฟพีจีเอ ของการออกแบบ Solution ที่ 5 มีการใช้เทคนิคไปป์ไลน์ร่วมด้วย

4.2.3 ออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอแบบใช้เทคนิคการโปรแกรมแบบขนาน

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 6

จากการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 5 ใช้เทคนิคไปป์ไลน์ร่วมด้วยในส่วนบล็อกที่ 1 บล็อกที่ 2 และบล็อกที่ 4 ที่เก็บข้อมูลด้วยอะเรย์ขนาด 1 บิต การคำนวณค่าจุดสีจากข้อมูลอินพุตที่เข้ามาที่เป็น 1 แถวทำให้การนำข้อมูลอินพุตมาประมวลผลได้ช้ากว่าการแบ่งข้อมูลเป็นหลายส่วนแล้วนำมาประมวลผล จึงได้ใช้เทคนิคแบ่งส่วนอะเรย์ (Array Splitting) ข้อมูลภาพที่เข้ามาเป็น 2 ส่วนเพื่อแบ่งการประมวลผลแบบขนานในระบบที่ช่วยให้ประมวลผลได้เร็วขึ้น ส่วนกระบวนการอื่นๆใช้เทคนิคที่เหมือนกับการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 5

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 7

ใช้กระบวนการเดียวกันกับการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 6 แต่ใช้เทคนิคแบ่งส่วนอะเรย์ (Array Splitting) ข้อมูลภาพที่เข้ามาเป็น 3 ส่วนเพื่อแบ่งการประมวลผลแบบขนานในระบบที่ช่วยให้ประมวลผลได้เร็วขึ้น

4.3 สรุป

ในบทนี้ได้แสดงถึงแนวคิดและแสดงขั้นตอนในการออกแบบระบบให้สามารถประมวลผลภาพวิดีโอด้วยภาษา“ImpulseC” ซึ่งในส่วนนี้ก็จะมีอัลกอริทึมที่ใช้นับปริมาณรถ ส่วนที่สองเป็นการออกแบบวงจรการนับปริมาณรถบนเอฟพีจีเอ และนำผลลัพธ์วงจร VHDLมาทำการจำลองการทำงานเพื่อทดสอบฟังก์ชันของระบบ แล้วสังเคราะห์วงจรบนเทคโนโลยีเอฟพีจีเอ สุดท้ายนำวงจรจริงที่ได้ไปทำการจำลองผลอีกครั้งเพื่อยืนยันว่าระบบที่ออกแบบและพัฒนาสามารถทำงานได้ถูกต้อง

บทที่ 5

ผลการวิจัย

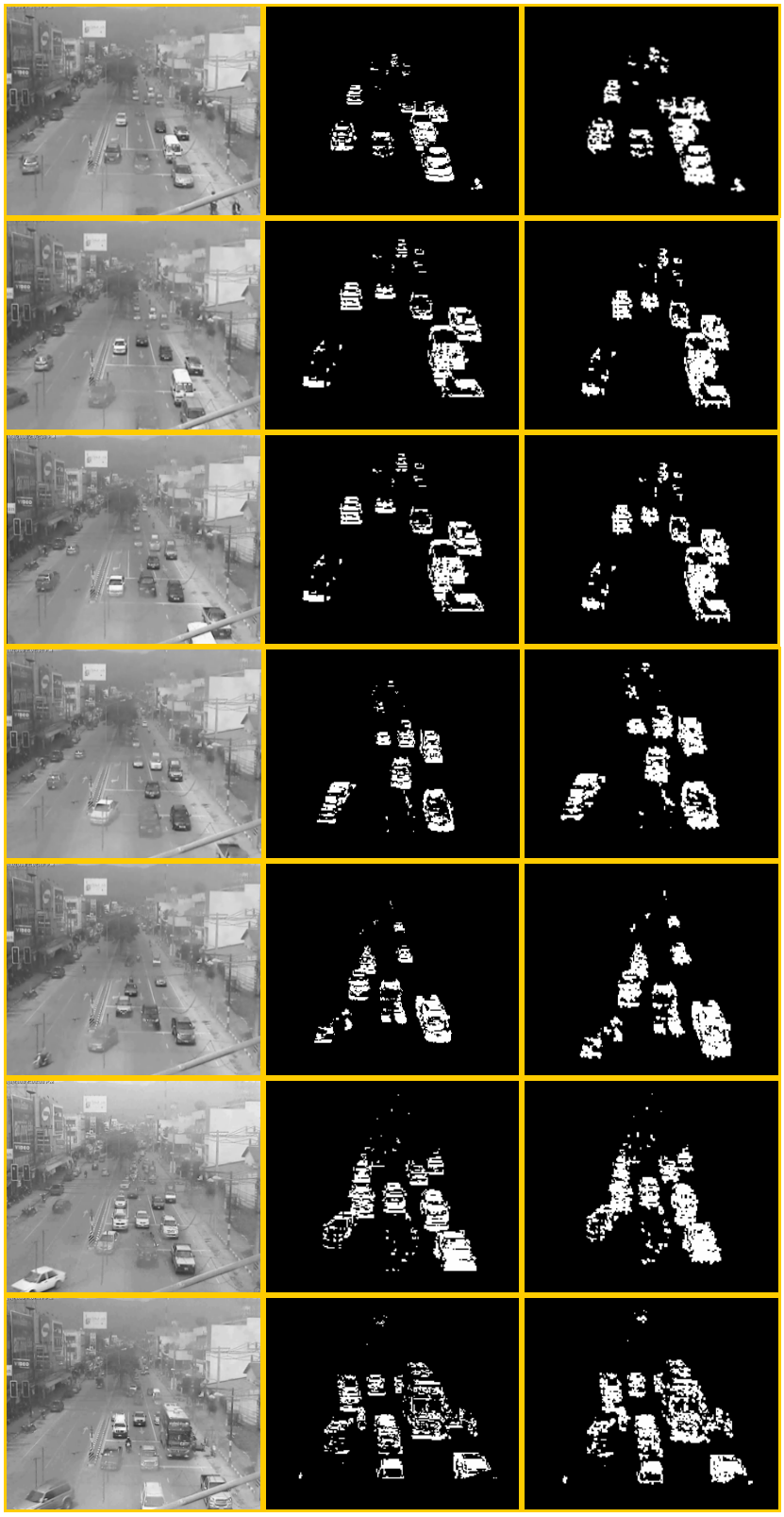
ในบทนี้อธิบายผลการทดสอบอัลกอริทึมของการนับปริมาณรถที่ใช้ภาษา ImpulseC ที่นำเฟรมภาพวิดีโอจากการจราจรจริงมาทดสอบแล้วพิจารณาความถูกต้องของผลลัพธ์ของการนับปริมาณรถจากโปรแกรม และปริมาณรถจริง และผลจากการออกแบบวงจรที่พัฒนาบนเทคโนโลยีเอฟพีจีเอในเรื่องความเร็วของการประมวลผล และการใช้ทรัพยากรเอฟพีจีเอ ตามขั้นตอนการออกแบบที่ได้กล่าวไว้ในบทที่ 4

5.1 ผลการหาภาพรถเคลื่อนที่

จากการศึกษาอัลกอริทึมการลบภาพพื้นหลังในการหาภาพรถเคลื่อนที่ในบทที่ 2 นั้น และได้ทำการทดสอบอัลกอริทึมการใช้เทคนิคการลบภาพพื้นหลัง ด้วยวิธีต่างๆดังที่กล่าวมาแล้วในบทที่ 2 ด้วยโปรแกรม Microsoft Visual C++ ที่มี Library ของ OpenCV สำหรับการเรียกใช้ฟังก์ชันการทำงานต่างๆ เพื่อศึกษาทำความเข้าใจวิธีการของอัลกอริทึม ข้อดีและข้อเสียของวิธีการลบภาพพื้นหลังแบบต่างๆ แล้วนำวิธีที่เหมาะสมที่สุดมาใช้กับระบบที่ต้องนำมาพัฒนาบนเทคโนโลยีเอฟพีจีเอ โดยใช้ภาษา Impulse C ที่เป็นการออกแบบพัฒนาระบบร่วมระหว่างฮาร์ดแวร์และซอฟต์แวร์ จากนั้นทำการออกแบบวงจรของระบบโดยใช้เทคนิคต่างๆเพื่อพัฒนาอัลกอริทึมให้มีการประมวลผลได้เร็วขึ้นในการหาภาพรถเคลื่อนที่ดังที่กล่าวไว้ในบทที่ 4

จากการออกแบบอัลกอริทึมในบทที่ 4 ขั้นตอนแรกได้เลือกใช้เทคนิคการลบภาพพื้นหลัง ด้วยวิธีหาผลต่างของเฟรม การทดสอบใช้ลำดับเฟรมภาพวิดีโอที่เป็นการจราจรจริงบริเวณแยกห้างสรรพสินค้าคาร์ฟูร์ อำเภอหาดใหญ่เป็นภาพอินพุต ที่มีข้อจำกัดของภาพที่ได้มา ได้แก่ ตำแหน่งกล้องที่อยู่คงที่ไม่มีการหมุนเพื่อเปลี่ยนทิศทางของภาพ ภาพที่ได้จากกล้องวงจรปิดเป็นมุมมองของภาพที่เป็นแบบ Perspective ที่ทำให้วัตถุที่อยู่หน้ากล้องมีขนาดใหญ่และจะมีขนาดเล็กลงเมื่อวัตถุอยู่ไกลออกไป จึงต้องทำให้สัดส่วนของภาพมีขนาดเท่ากันทั่วทั้งภาพ และปัญหาบางประการ ได้แก่การสั่นไหวของกล้องในบางครั้งเนื่องจากขนาดของกล้องที่มีขนาดเล็ก และสภาพกล้องที่ขุ่นมัวทำให้ภาพที่ได้จากกล้องในบางบริเวณไม่ถูกต้องจนรายละเอียดของรถบางส่วนหายไป เมื่อหาผลต่างของเฟรมแล้วลำดับถัดไปจะต้องนำค่าผลต่างเฟรมมาทำกระบวนการ

Threshold สำหรับการเลือกค่า Threshold นั้นเนื่องจากไฟล์วิดีโอตัวอย่างเป็นไฟล์ในระยะเวลาสั้นๆ ที่ไม่มีความแตกต่างของแสงมากนัก จึงใช้ค่า Threshold เพียงค่าเดียว คือ 25 เพื่อทำภาพไบนารีจากผลต่างของเฟรมมาเทียบกับค่า Threshold จะได้เป็นภาพขาวดำ ซึ่งส่วนสีขาวของภาพเป็นภาพของรถเคลื่อนที่ แต่มีช่องว่างของรถเนื่องจากสีของรถใกล้เคียงกับสีของพื้นถนน หรือสภาพของกล้องที่ขุ่นมัวทำให้ในบางตำแหน่งสีของรถไม่ชัดเจน จึงต้องปิดช่องว่างของรถด้วยกระบวนการ Morphological ดังภาพประกอบ 5-1 ซึ่งภาพประกอบ 5-1(ก) เป็นลำดับภาพวิดีโอที่เวลาต่างๆ ภาพประกอบ 5-1(ข) ผลจากการหาผลต่างของเฟรมและเทียบกับค่า Threshold และภาพประกอบ 5-1(ค) ผลจากการทำกระบวนการ Morphological โดยใช้โอเปอร์เรชัน close ด้วยเทมเพลตเมตริกซ์ขนาด 2x2 เพื่อปิดช่องว่างรถ



(ก) (ข) (ค)

ภาพประกอบ 5-1 ผลจากการหาภาพรถเคลื่อนที่

5.2 ผลการแก้ไขมุมมองภาพแบบ Perspective

การแก้ไขมุมมองภาพจากกล้องวิดีโอที่เป็นมุมมองของภาพแบบ Perspective ซึ่งจำนวนพิกเซลของรถหนึ่งคันจะมีขนาดลดลงเมื่อรถอยู่ไกลจากกล้องออกไป ต้องหาค่านอมอลไลซ์แฟคเตอร์นำมาคูณกับจำนวนพิกเซลของรถในแต่ละแถว เพื่อเป็นการขยายภาพของรถในบริเวณที่อยู่ไกลกล้องออกไปให้เป็นสัดส่วนเดียวกันทั้งหมด เมื่อนับจำนวนจุดสีของทุกๆแถวในบริเวณที่ต้องการนับปริมาณรถ ดังภาพประกอบ 3-6(ก) ที่กล่าวมาแล้วในบทที่ 3 ในตำแหน่งของกล้องซึ่งเป็นตำแหน่งเดิมนั้น จากนั้น นำค่าจำนวนแถวทั้งหมดของบริเวณที่ต้องการนับปริมาณรถมาพล็อตกราฟในแกน X และจำนวนของรถในแต่ละแถวมาพล็อตกราฟในแกน Y ดังภาพประกอบ 3-6(ข) ที่กล่าวมาแล้วในบทที่ 3

ในตำแหน่งกล้องดังภาพประกอบ 3-6 (ก) ซึ่งมีคุณสมบัติ คือเมื่อนับจำนวนจุดสีของบริเวณที่ต้องการนับปริมาณรถ มีจำนวนแถวของภาพ 115 แถว และจำนวนจุดสีในแต่ละแถวมีจำนวนตั้งแต่ 58-158 เมื่อนำค่าจำนวนแถวทั้งหมดของบริเวณที่ต้องการนับปริมาณรถมาพล็อตกราฟในแกน X และจำนวนของรถในแต่ละแถวมาพล็อตกราฟในแกน Y ดังภาพประกอบ 3-6(ข) จะได้ว่า $L_{\max} = 158,\ L_{\min} = 58$ และ $C_{\max} = 115$ จากสมการนอมอลไลซ์แฟคเตอร์ ตามสมการ (3-4) ที่กล่าวมาแล้วในบทที่ 3

เมื่อแทนค่าในสมการ (3-4) จะได้เป็นสมการ (4-1)

$$F_x = \frac{158}{(\frac{-100}{115})X + 158} \tag{4-1}$$

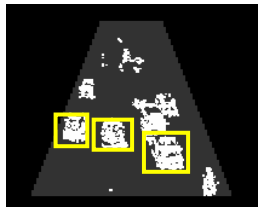
เมื่อ X = 0 – 115

จากสมการ (4-1) ค่า F_x ที่ได้เป็นจุดทศนิยมที่เป็น Floating point จากการที่มีตัวหาร ที่ทำให้การประมวลผลของระบบทำงานได้ช้าลงและใช้ทรัพยากรของเอฟพีจีเอที่มากขึ้นจึงออกแบบให้ ค่า F_x เป็นตัวแปรแบบจำนวนเต็มที่เป็น Fixed point โดยปรับสมการที่ (4-1) เป็น

$$F_x = \frac{18170}{18170 - 100X} \tag{4-2}$$

จากสมการที่ (4-2) เป็นการปรับสมการเพื่อหลีกเลี่ยงการใช้ตัวหารด้วยการ shift ขวาและหลีกเลี่ยงการใช้ตัวคูณด้วยการ shift ซ้าย ค่า F_x ที่ได้จะเป็นค่า Fixed point และกระบวนการอื่น ได้แก่ การหาภาพพื้นหลัง กระบวนการ morphological และหาพื้นที่ของจุดสีภาพยังคงเป็นกระบวนการเดียวกับอัลกอริทึมแบบ Floating point ที่เป็นตัวแปรแบบจำนวนเต็ม (integer)

ซึ่งต้องนำค่า F_x ไปคูณกับจำนวนจุดสีของรถในแต่ละแถว จะได้บริเวณพื้นที่ของรถทั้งหมด ที่มีการขยายพื้นที่ของรถในบริเวณที่ไกลจากกล้องให้มีขนาดเดียวกัน เมื่อได้พื้นที่โดยรวมของรถทั้งหมด ในแต่ละเฟรมภาพ จะต้องนำมาเปรียบเทียบด้วยพื้นที่ของรถ 1 คัน ผลลัพธ์ที่ได้ก็จะเป็นจำนวนรถในแต่ละเฟรมภาพ โดยค่าเฉลี่ยของรถ 1 คัน มาจากพื้นที่ที่ทำการแก้ไขสัดส่วนของภาพแล้ว คือ ค่าความกว้างของภาพคูณด้วยความยาวของภาพ มีพื้นที่เท่ากับค่าเฉลี่ยพื้นที่ของรถ 1 คัน ที่สามารถหาได้ดังภาพประกอบ 5-2 (ก)-(ง)



(ก)

(ข)

(ค)

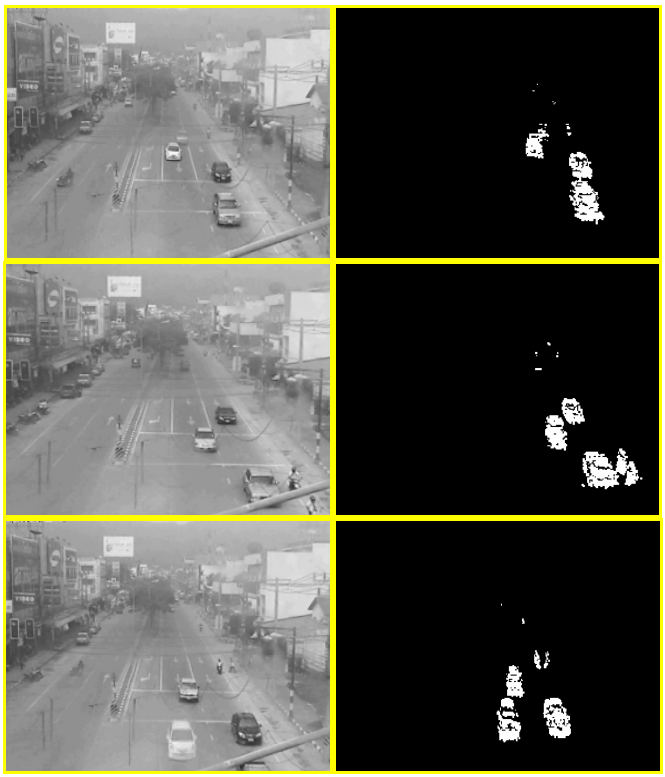
(ง)

ภาพประกอบที่ 5-2 การเลือกบริเวณพื้นที่เฉลี่ยของรถ 1 คัน (ก) คือ บริเวณที่ทำการเลือกพื้นที่รถแต่ละคัน (ข) คือ รถที่เลือกคันที่ 1 (ค) คือ รถที่เลือกคันที่ 2 (ง) คือ รถที่เลือกคันที่ 3

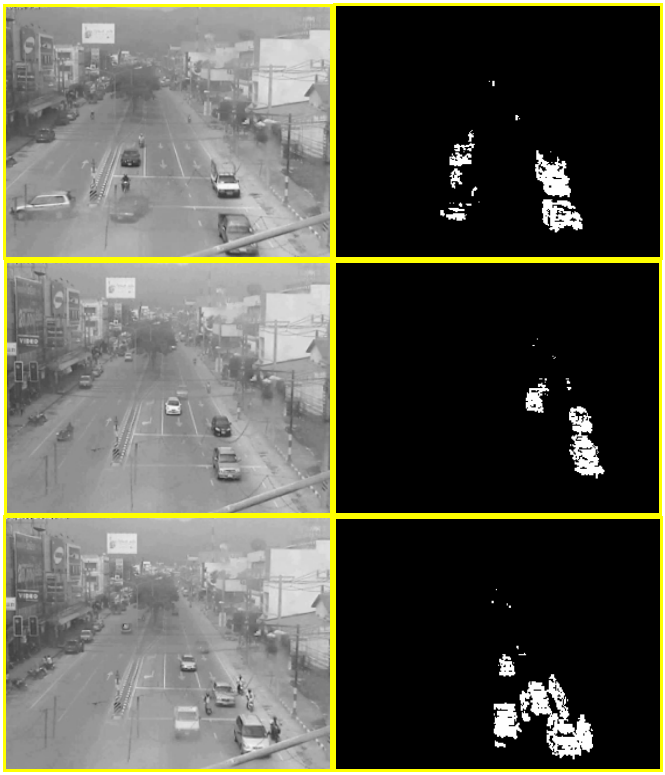
เมื่อเลือกเฟรมที่มีจำนวนรถ ตั้งแต่ 3 – 10 คัน ที่แตกต่างกัน 10 กรณี จำนวน 400 เฟรม ผลลัพธ์ที่ได้แสดงดังตารางที่ 5-1 และภาพประกอบที่ 5-3 ถึงภาพประกอบที่ 5-8

ตารางที่ 5-1 ผลการนับปริมาณรถตามจำนวน 3 – 10 คัน ที่แตกต่างกัน 10 กรณี
จำนวน 400 เฟรม จาก CoDeveloper

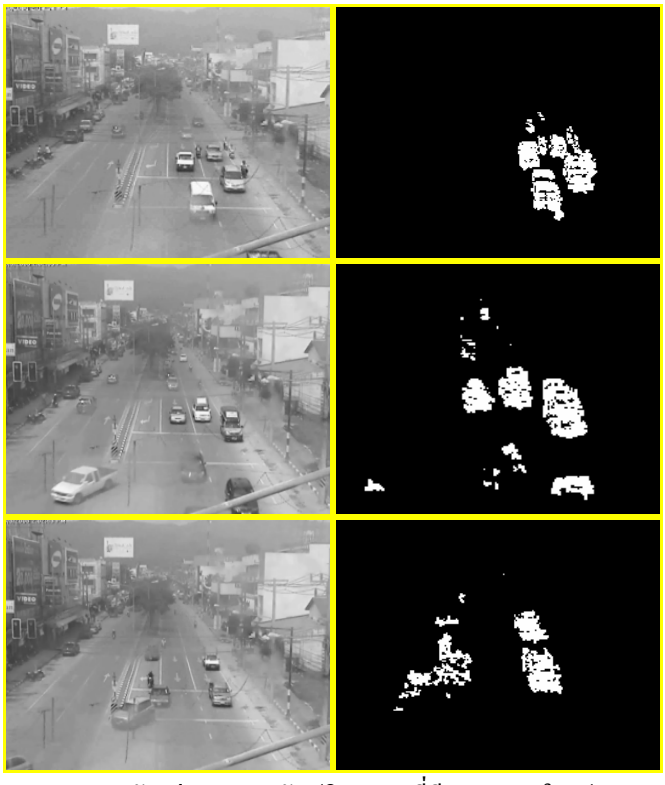
จำนวนรถภายใน เฟรมภาพ (คัน)	จำนวนรถที่นับได้โดยเฉลี่ย 10 กรณี (คัน) (Floating Point)	จำนวนรถที่นับได้โดยเฉลี่ย 10 กรณี (คัน) (Fixed Point)
3	3.46	4
4	4.05	4
5	5.35	5
6	5.21	6
7	6.19	6
8	6.11	6
9	7.37	7
10	8.41	8



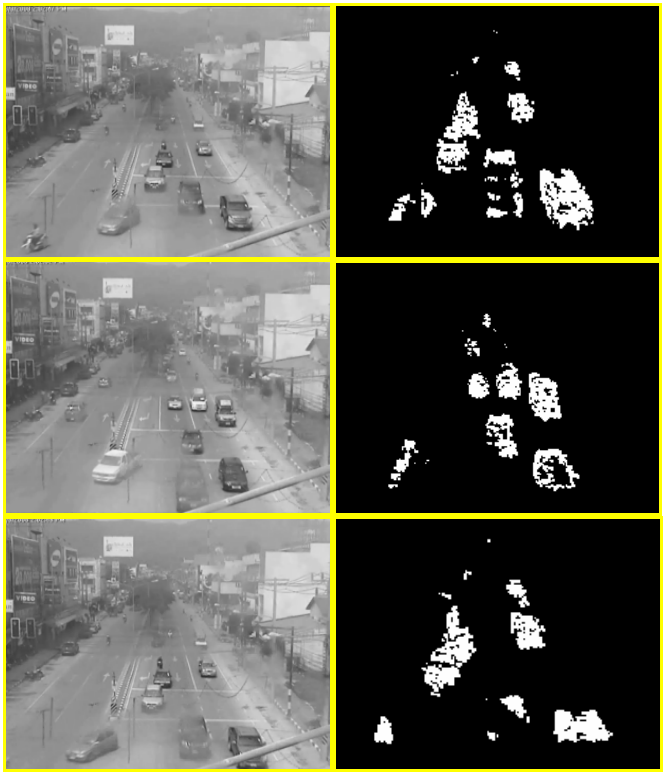
ภาพประกอบ 5-3 ตัวอย่างผลการนับปริมาณรถที่มีจำนวนรถในเฟรมภาพ 3 คัน



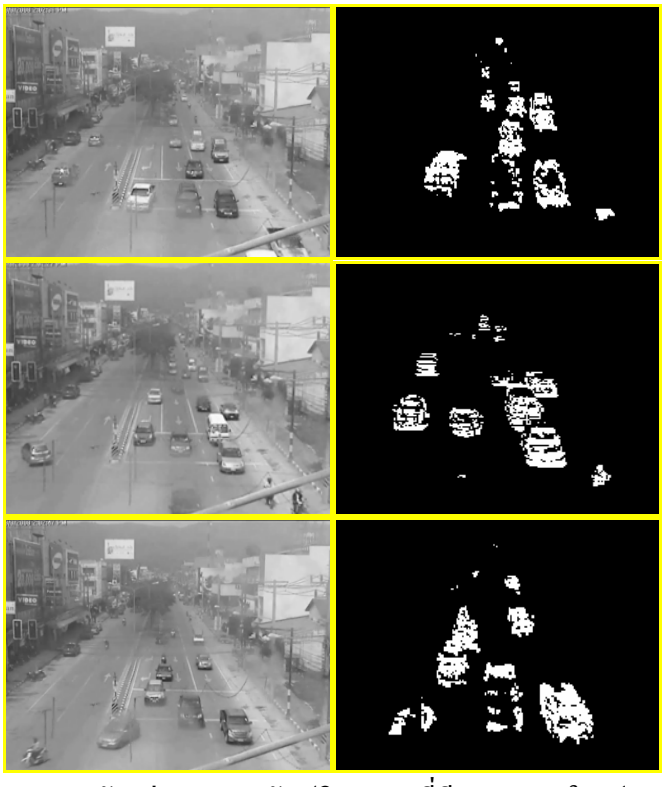
ภาพประกอบ 5-4 ตัวอย่างผลการนับปริมาณรถที่มีจำนวนรถในเฟรมภาพ 4 คัน



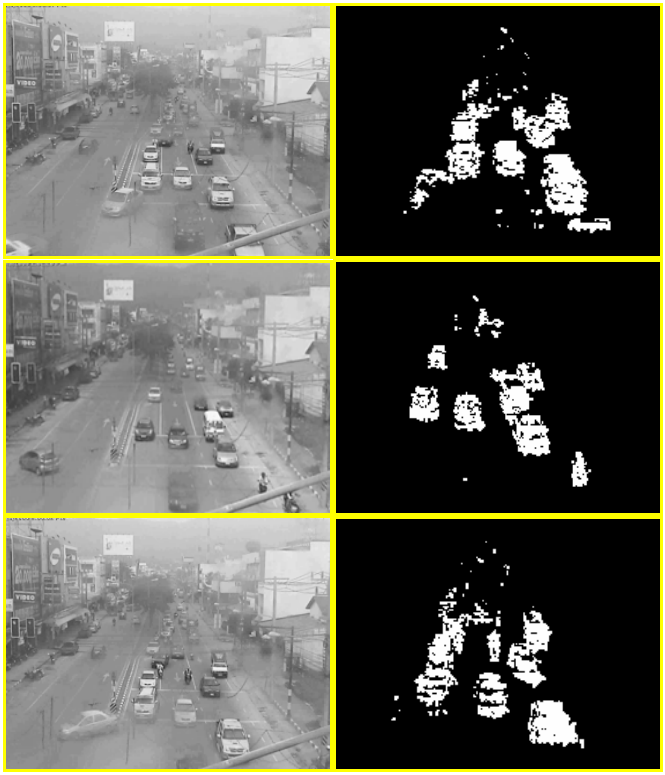
ภาพประกอบ 5-5 ตัวอย่างผลการนับปริมาณรถที่มีจำนวนรถในเฟรมภาพ 5 คัน



ภาพประกอบ 5-6 ตัวอย่างผลการนับปริมาณรถที่มีจำนวนรถในเฟรมภาพ 6 คัน



ภาพประกอบ 5-7 ตัวอย่างผลการนับปริมาณรถที่มีจำนวนรถในเฟรมภาพ 7 คัน



ภาพประกอบ 5-8 ตัวอย่างผลการนับปริมาณรถที่มีจำนวนรถในเฟรมภาพ 8 คัน

จากตารางที่ 5-1 พบว่า การนับปริมาณรถมีการผิดพลาดไม่เกิน 2 คัน ทั้งนี้เนื่องมาจากสภาพกล้องที่ขุ่นมัวทำให้ภายในภาพเห็นรถไม่ชัดเจน รถที่มีสีใกล้เคียงกับพื้นถนนเมื่อหาผลต่างของเฟรมแล้วทำให้บางส่วนของรถหายไป และไม่ได้แยกประเภทรถมอเตอร์ไซค์ รถบัส และรถบรรทุก

จากนั้นได้นำวงจร HDL ที่สร้างได้จากซอฟต์แวร์ Impulse CoDeveloper ไปจำลองการทำงานบนเอฟพีจีเอด้วยซอฟต์แวร์ Xilinx ISE เพื่อวิเคราะห์ผลลัพธ์ที่ได้ ดังตาราง 5-2 โดยในการทดสอบได้ใช้ลำดับเฟรมภาพเกรสเกล ในสกุล pgm ขนาด 320x240 พิกเซล เป็นภาพอินพุต

5.3 ผลการสังเคราะห์วงจรบนเอฟพีจีเอ

ตารางที่ 5-2 ผลการสังเคราะห์วงจรแบบ Floating point บนเอฟพีจีเอ Virtex II Pro XC2VP30

Solution	Latency (s)	Throughput (Frames/sec)	Slices/Flip Flops	BRAMs (18 KB)	MULT18×18s (18x18 bit)
1	0.81	1.23	4940(36%) / 2332(8%)	41(30%)	8(5%)
2	0.792	1.26	4943(36%) / 2321(8%)	2(1%)	8(5%)
3	0.349	2.538	4476(32%) / 2146(7%)	41(30%)	8(5%)
4	0.392	2.551	4480(32%) / 2113(7%)	2(1%)	8(5%)
5	0.18	5.56	18221(133%) / 26380(96%)	2(1%)	8(5%)
6	0.22	4.55	12486(91%) / 15299(55%)	2(1%)	16(11%)
7	0.251	3.984	12142(88%) / 13117(47%)	3(2%)	24(27%)

ตารางที่ 5-3 ผลการสังเคราะห์วงจรแบบ Fixed point บนเอฟพีจีเอ Virtex II Pro XC2VP30

Solution	Latency (s)	Throughput (Frames/sec)	Slices/Flip Flops	BRAMs (18 KB)	MULT18×18s (18x18 bit)
1	0.70	1.43	3574(26%) / 899(3%)	42(30%)	-
2	0.8	1.25	3596(26%) / 892(3%)	3(2%)	-
3	0.38	2.63	3119(22%) / 702(2%)	42(30%)	-
4	0.37	2.70	3077(22%) / 644(2%)	3(2%)	-
5	0.16	6.25	16837(122%) / 24928(91%)	3(2%)	-
6	0.18	5.55	9531(69%) / 12469(45%)	3(2%)	-
7	0.19	5.26	9063(66%) / 10888(39%)	3(2%)	-

จากการออกแบบวงจรได้แบ่งการทำงานเป็น 5 บล็อกการทำงาน ที่ประกอบด้วย บล็อกที่ 1 การหาผลต่างเฟรม บล็อกที่ 2 การทำ Threshold บล็อกที่ 3 การทำ morphological บล็อกที่ 4 การคูณด้วยค่านอมอลไลซ์แฟคเตอร์ และบล็อกที่ 5 การหาปริมาณรถ

จากตารางที่ 5-2 และ 5-3 แสดงผลการใช้จำนวนทรัพยากรที่จะใช้งานบนบอร์ด VirtexII Pro XC2VP30 และแสดงเวลาที่ประมวลผลต่อหนึ่งเฟรม ตารางที่ 5-2 เป็นการทดสอบโดยใช้อัลกอริทึมแบบ Floating point และ ตารางที่ 5-3 เป็นการใช้อัลกอริทึมแบบ Fixed pointโดย จำนวน Slices เป็นตัวบอกถึงพื้นที่โดยรวมที่จะใช้งาน บอร์ดเอฟพีจีเอรุ่นนี้มีจำนวนพื้นที่ Slices 13696 Slices ประกอบด้วย จำนวนการใช้ลอจิก Flip Flops บอร์ดเอฟพีจีเอรุ่นนี้มีจำนวน Flip Flops 27392 ตัว, แสดงจำนวนการใช้บล็อกแรม (BRAMs) แต่ละบล็อกแรมมีขนาด 18 กิโลไบต์ บอร์ดเอฟพีจีเอรุ่นนี้มีจำนวนบล็อกแรม 136 บล็อก , แสดงการใช้บล็อกตัวคูณ (MULT18×18) แต่ละบล็อกตัวคูณมีขนาด 18×18บิต บอร์ดเอฟพีจีเอรุ่นนี้มีจำนวนบล็อกตัวคูณ 136 บล็อก และ Latency(s)เป็นตัวบอกความเร็วของวงจรในการประมวลผล ภาพต่อหนึ่งเฟรมซึ่งในการประมวลผลหนึ่งเฟรม(Latency) คำนวณจาก (stage)x(clock Period)x (จำนวนจุดสี) เมื่อประมวลผลทั้งระบบโดยใช้อัลกอริทึมแบบ Floating point และ Fixed point ได้ทดลองจาก 7 วิธี ตามที่ออกแบบในบทที่ 4 ดังนี้

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 1 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 8 บิตทั้งเฟรม
- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 2 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ
- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 3 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 8 บิตทั้งเฟรม ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 และบล็อกที่ 2
- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 4 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 และบล็อกที่ 2
- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 5 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้

หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 บล็อกที่ 2 และบล็อกที่ 4

- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 6 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบขนานและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 บล็อกที่ 2 และบล็อกที่ 4 และเทคนิค Array Splitting แบ่งส่วนอะเรย์บัฟเฟอร์เป็น 2 ส่วนเพื่อแบ่งการประมวลผล
- การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 7 เป็นผลการสังเคราะห์วงจร อัลกอริทึมการนับรถบนเอฟพีจีเอแบบขนานและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 บล็อกที่ 2 และบล็อกที่ 4 และเทคนิค Array Splitting แบ่งส่วนอะเรย์บัฟเฟอร์เป็น 3ส่วนเพื่อแบ่งการประมวลผล

5.4 วิเคราะห์ผลการทดลอง

จากผลการทดลองในตารางที่ 5-2 เป็นผลจากการออกแบบวงจรเพื่อพัฒนาบนเอฟพีจีเอแบบต่างๆที่นำเอาเทคนิคการปรับปรุงการโปรแกรมให้มีความเหมาะสมในจากบทที่ 3 และบทที่ 4 มาประยุกต์ใช้กับระบบเพื่อเพิ่มประสิทธิภาพในการประมวลผลที่คำนึงถึงความเร็วในการประมวลผลภาพต่อหนึ่งเฟรม ที่สัมพันธ์กับการใช้ทรัพยากรที่มีอยู่อย่างจำกัดในเอฟพีจีเอ

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 1 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 8 บิตทั้งเฟรม ที่ไม่มีการใช้เทคนิคใดๆเพื่อเพิ่มความเร็วในการประมวลผล ผลจากอัลกอริทึมแบบ Floating point ใช้ทรัพยากรหน่วยความจำสำหรับเก็บข้อมูลอะเรย์ 8 บิต และผลลัพธ์ที่เป็นจำนวนคัน ชนิด float 32 บิตใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอถึง 41 บล็อกแรม ใช้บล็อกตัวคูณ 8 บล็อก จำนวน Slices เท่ากับ 4940 และใช้ความเร็วในการประมวลผล 0.81 วินาทีต่อหนึ่งเฟรม ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอถึง 42 บล็อกแรม จำนวน Slices เท่ากับ 3574 และใช้ความเร็วในการประมวลผล 0.7 วินาทีต่อหนึ่งเฟรม ซึ่งไม่มีการ

ใช้ตัวคูณ MULT 18x18 บิต เนื่องจากการไม่มีการคูณและการหาร เป็นการใช้การ shift ขวาแทนการหารและการ shift ซ้ายแทนการคูณ

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 2 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิต เฉพาะบริเวณที่ต้องการนับรถและผลลัพธ์ที่เป็นจำนวนคัน ชนิด float 32 บิต ที่ไม่มีการใช้เทคนิคใดๆเพื่อเพิ่มความเร็วในการประมวลผล ผลจากอัลกอริทึมแบบ Floating point จะเห็นได้ว่าสามารถลดการใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอลงมาได้เป็น 2 บล็อกแรม ใช้บล็อกตัวคูณ 8 บล็อก จำนวน Slices เท่ากับ 4943 ส่วนความเร็วในการประมวลผล 0.792 วินาทีต่อหนึ่งเฟรม ในเรื่องของความเร็วจะใกล้เคียงกับ การออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 1 ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอ 3 บล็อกแรม จำนวน Slices เท่ากับ 3596 และใช้ความเร็วในการประมวลผล 0.8 วินาทีต่อหนึ่งเฟรม

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 3 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 8 บิตทั้งเฟรมและผลลัพธ์ที่เป็นจำนวนคัน ชนิด float 32 บิต ที่เป็น Solution ที่ 1 ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 และบล็อกที่ 2 ผลจากอัลกอริทึมแบบ Floating point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอถึง 41 บล็อกแรมและใช้บล็อกตัวคูณ 8 บล็อก เท่ากับ Solution ที่ 1 จำนวน Slices เท่ากับ 4476 แต่ใช้ความเร็วในการประมวลผลลดลงถึง 0.349 วินาทีต่อหนึ่งเฟรม เนื่องจากสเตจของการทำงานลดลงทำให้ความเร็วในการประมวลผลเพิ่มขึ้นด้วย ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอถึง 42 บล็อกแรม จำนวน Slices เท่ากับ 3119 และใช้ความเร็วในการประมวลผล 0.38 วินาทีต่อหนึ่งเฟรม

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 4 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิต เฉพาะบริเวณที่ต้องการนับรถและผลลัพธ์ที่เป็นจำนวนคัน ชนิด float 32 บิตเช่นเดียวกับ Solution ที่ 2 แต่มีการใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 และบล็อกที่ 2 ร่วมด้วย ผลจากอัลกอริทึมแบบ Floating point ใช้ความเร็วในการประมวลผลลดลงถึง 0.392 วินาทีต่อหนึ่งเฟรม เนื่องจากสเตจของการทำงานลดลงทำให้ความเร็วในการประมวลผลเพิ่มขึ้นด้วย เช่นเดียวกับ Solution ที่ 3 ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอเพียง 2 บล็อกแรมและใช้บล็อกตัวคูณ 8 บล็อก เท่ากับ Solution ที่ 2 และ จำนวน Slices เท่ากับ 4480 เป็น Solution ที่ให้ผลลัพธ์ที่ดีในด้านความเร็วและการใช้ทรัพยากรหน่วยความจำ ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็น

หน่วยความจำในเอฟพีจีเอ 3 บล็อกแรม จำนวน Slices เท่ากับ 3077 และใช้ความเร็วในการประมวลผล 0.37 วินาทีต่อหนึ่งเฟรม

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 5 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิต เฉพาะบริเวณที่ต้องการนับรถและผลลัพธ์ที่เป็นจำนวนคัน ชนิด float 32 บิตเช่นเดียวกับ Solution ที่ 4 ที่ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 บล็อกที่ 2 และเพิ่มการใช้เทคนิคไปป์ไลน์บล็อกที่ 4 ผลจากอัลกอริทึมแบบ Floating point ความเร็วของการประมวลผลเป็น 0.18 วินาทีต่อหนึ่งเฟรมใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอเพียง 2 บล็อกแรมและใช้บล็อกตัวคูณ 8 บล็อกเท่ากับ Solution ที่ 4 แต่ใช้ Slices จำนวนมากถึง 18221 ซึ่งใช้เกินทรัพยากรหน่วยความจำที่มีจำกัดของเอฟพีจีเอ ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอ 3 บล็อกแรม จำนวน Slices เท่ากับ 16737 ซึ่งใช้เกินทรัพยากรหน่วยความจำที่มีจำกัดของเอฟพีจีเอ และใช้ความเร็วในการประมวลผล 0.16 วินาทีต่อหนึ่งเฟรม

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 6 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบขนานและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 บล็อกที่ 2 และบล็อกที่ 4 จาก Solution ที่ 5 และเพิ่มเทคนิค Array Splitting แบ่งส่วนอะเรย์บัฟเฟอร์เป็น 2 ส่วนเพื่อแบ่งการประมวลผล ผลจากอัลกอริทึมแบบ Floating point ความเร็วของการประมวลผลเป็น 0.22 วินาทีต่อหนึ่งเฟรมใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอเพียง 2 บล็อกแรม เท่ากับ Solution ที่ 5 แต่ใช้บล็อกตัวคูณ 16 บล็อกเพิ่มขึ้นจาก Solution ที่ 5 เป็นสองเท่าและใช้ Slices ลดลงจาก Solution ที่ 5 เป็น 12486 ซึ่งใช้ไม่เกินทรัพยากรหน่วยความจำที่มีจำกัดแต่ใกล้เคียงค่าสูงสุดของทรัพยากรหน่วยความจำที่มีของเอฟพีจีเอ ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอ 3 บล็อกแรม จำนวน Slices เท่ากับ 9531 และใช้ความเร็วในการประมวลผล 0.18 วินาทีต่อหนึ่งเฟรม

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 7 เป็นอัลกอริทึมการนับรถบนเอฟพีจีเอแบบขนานและใช้หน่วยความจำเดิมที่เก็บภาพในอะเรย์ขนาด 1 บิตเฉพาะบริเวณที่ต้องการนับรถ ใช้เทคนิคไปป์ไลน์ในบล็อกที่ 1 บล็อกที่ 2 และบล็อกที่ 4 และเทคนิค Array Splitting เช่นเดียวกับ Solution ที่ 6 แต่เพิ่มการแบ่งส่วนอะเรย์บัฟเฟอร์เป็น 3 ส่วนเพื่อแบ่งการประมวลผล ผลจากอัลกอริทึมแบบ Floating point ความเร็วของการประมวลผลเป็น 0.251 วินาทีต่อหนึ่งเฟรมใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอเพิ่มขึ้นเป็น 3 บล็อกแรม แต่ใช้บล็อกตัวคูณ 24 บล็อกเพิ่มขึ้นจาก Solution ที่ 5 เป็นสามเท่าและใช้ Slices ใกล้เคียงกับ Solution ที่ 6 เป็น 12142 ซึ่งใช้ไม่เกินทรัพยากรหน่วยความจำที่มีจำกัดแต่ใกล้เคียงค่าสูงสุดของทรัพยากร

หน่วยความจำที่มีของเอฟพีจีเอ ส่วนผลจากอัลกอริทึมแบบ Fixed point ใช้บล็อกแรมที่เป็นหน่วยความจำในเอฟพีจีเอ 3 บล็อกแรม จำนวน Slices เท่ากับ 9063 และใช้ความเร็วในการประมวลผล 0.19 วินาทีต่อหนึ่งเฟรม

5.5 สรุป

จากผลการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอแบบ Solution ต่างๆ สรุปได้ว่า เมื่อออกแบบทำให้อัลริทึมให้มีความเร็วที่เพิ่มขึ้น โดยเทคนิคไปป์ไลน์ หรือการแบ่งส่วนของอะเรย์เพื่อประมวลผลแบบขนานนั้นได้ความเร็วที่เพิ่มขึ้น แต่การใช้ทรัพยากรของเอฟพีจีเอก็มากขึ้นเช่นกัน ดังนั้น ในการออกแบบวงจรนับปริมาณรถบนเอฟพีจีเอ Solution ที่ 4 จึงเป็นการออกแบบวงจรที่เหมาะสมกับระบบ การประมวลผลจากอัลกอริทึมแบบ Fixed point ทำให้ระบบสามารถประมวลผลได้เร็วกว่าการประมวลผลจากอัลกอริทึมแบบ Floating point และการประมวลผลจากอัลกอริทึมแบบ Fixed point ทำให้การใช้ทรัพยากรของเอฟพีจีเอน้อยลงกว่าการประมวลผลจากอัลกอริทึมแบบ Floating point เนื่องจากเป็นการหลีกเลี่ยงการใช้การคูณและการหาร โดยใช้การ Shift ขวาแทนการหารและการ shift ซ้ายแทนการคูณ

บทที่ 6

บทสรุปและข้อเสนอแนะ

ในบทนี้กล่าวถึงสรุปผลการวิจัยที่ได้ดำเนินการสำหรับวิทยานิพนธ์นี้ รวมทั้งข้อเสนอแนะต่างๆ ที่จะเป็นประโยชน์ต่อการทำวิจัยด้านการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์และเอฟพีจีเอ

6.1 บทสรุป

วิทยานิพนธ์ฉบับนี้ได้นำเสนอการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลภาพวิดีโอเพื่อการนับปริมาณรถจากการจราจรโดยใช้เทคโนโลยีเอฟพีจีเอเข้ามาเป็นตัวช่วยในการประมวลผลจากผลการทดสอบอัลกอริทึมกระบวนการนับปริมาณรถโดยใช้เทคนิคการลบภาพพื้นหลังด้วยวิธีการหาผลต่างของเฟรมที่ต้องเปรียบเทียบผลต่างของเฟรมด้วยค่า Threshold ผลที่ได้เกิดสัญญาณภาพรบกวนที่เกิดจากปัญหาการสั่นของกล้อง การขุ่นมัวของกล้อง และสีรถที่ใกล้เคียงกับถนน จึงต้องทำการลดสัญญาณภาพรบกวนและปิดช่องว่างของรถด้วยวิธี Morphological โดยใช้โอเปอร์เรเตอร์ Close และ Open

ภาพที่ได้จากกล้องวงจรปิดเป็นภาพแบบ Perspective ที่มีสัดส่วนของภาพไม่เท่ากัน ภาพวัตถุที่อยู่หน้ากล้องจะมีขนาดใหญ่และวัตถุจะมีขนาดเล็กลงเมื่อไกลกล้องออกไป จึงต้องทำการแก้ไขมุมมองของภาพให้มีสัดส่วนเดียวกันทั้งหมดโดยหาค่านอมอลไลซแฟคเตอร์ที่จะต้องนำมาคูณกับพื้นที่ของวัตถุในแต่ละแถวของพื้นที่ที่ต้องการนับปริมาณรถ แล้วหาผลรวมของพื้นที่วัตถุทุกๆแถวจะได้พื้นที่ของวัตถุเคลื่อนที่ทั้งหมด และการหาจำนวนรถโดยนำพื้นที่ของรถทั้งหมดมาหารด้วยค่าเฉลี่ยของรถหนึ่งคัน

เมื่อทดสอบอัลกอริทึมการนับปริมาณรถดังที่กล่าวมา ซึ่งเป็นอัลกอริทึมแบบ Fixed point และ Floating point ผลจากการโปรแกรมอัลกอริทึมที่ใช้ ImpulseC ที่นำลำดับภาพวิดีโอที่เลือกเฟรมที่มีจำนวนรถ ตั้งแต่ 3 – 10 คัน ที่แตกต่างกัน 10 กรณี มาทดสอบการนับปริมาณรถมีการผิดพลาดไม่เกิน 2 คัน ทั้งนี้เนื่องมาจากสภาพกล้องที่ขุ่นมัวทำให้ภายในภาพเห็นรถไม่ชัดเจน รถที่มีสีใกล้เคียงกับพื้นถนนเมื่อหาผลต่างของเฟรมแล้วทำให้บางส่วนของรถหายไป และไม่ได้แยกประเภทรถจักรยานยนต์ รถบัส และรถบรรทุก

ขั้นตอนสุดท้าย เป็นการพัฒนาอัลกอริทึมนับปริมาณรถบนเทคโนโลยีเอฟพีจีเอ นำวงจร HDL ที่สร้างขึ้นมาจากซอฟต์แวร์ ImpulseC ไปทดสอบด้วยขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอ เพื่อทำการจำลองผลการทำงานบนบอร์ดเอฟพีจีเอ VirtexII Pro XC2VP30 โดยออกแบบวงจรของอัลกอริทึมแบบตามลำดับ และนำเทคนิคไปป์ไลน์ และการโปรแกรมแบบขนาน โดยออกแบบเป็น Solution แบบ Fixed point และ Floating point เพื่อเปรียบเทียบผลการประมวลผลในการใช้ทรัพยากร และความเร็วของการประมวลผลต่อหนึ่งเฟรม เมื่อออกแบบทำให้อัลริทึมให้มีความเร็วที่เพิ่มขึ้น โดยเทคนิคไปป์ไลน์ หรือการแบ่งส่วนของอะเรย์เพื่อประมวลผลแบบขนานนั้นทำให้มีการประมวลผลด้วยความเร็วที่เพิ่มขึ้น แต่การใช้ทรัพยากรของเอฟพีจีเอก็มากขึ้นเช่นกัน และการประมวลผลจากอัลกอริทึมแบบ Fixed point ทำให้ระบบสามารถประมวลผลได้เร็วกว่าการประมวลผลจากอัลกอริทึมแบบ Floating point และการประมวลผลจากอัลกอริทึมแบบ Fixed point ทำให้การใช้ทรัพยากรของเอฟพีจีเอน้อยลงกว่าการประมวลผลจากอัลกอริทึมแบบ Floating point

6.2 ข้อเสนอแนะ

1. ในงานวิจัยนี้ ช่วยเพิ่มประสิทธิภาพการประมวลผลภาพด้วยการนำเทคนิคการออกแบบร่วมฮาร์ดแวร์และซอฟต์แวร์ ทำให้ใช้เวลาในการพัฒนาระบบการประมวลผลภาพรวดเร็วขึ้น

2. ในงานวิจัยนี้ ใช้ซอฟต์แวร์ที่ชื่อว่า “ Impulse CoDeveloper “ ในการพัฒนาโปรแกรมเพราะซอฟต์แวร์ตัวนี้มีคุณสมบัติเด่นคือสามารถแปลงภาษาซีเป็นภาษา VHDL หรือ Verilogเพื่อให้สามารถนำไปใช้งานบนบอร์ดเอฟพีจีเอได้จริง แต่ทั้งนี้ขึ้นกับขนาดของวงจรที่ได้จะต้องไม่มีขนาดใหญ่เกินความสามารถของเอฟพีจีเอ

3. ในงานวิจัยนี้ ไม่มีการแยกขนาดของรถชนิดอื่นๆ ดังนั้น แนวทางการพัฒนาต่อไปควรมีการแยกขนาดของรถชนิดอื่นๆที่มีขนาดแตกต่างกันเพิ่มเติม เช่น รถประจำทาง รถจักรยานยนต์ เป็นต้น และหาค่าเฉลี่ยของขนาดของรถ 1 คัน จากขนาดของรถที่สุ่มเลือกในจำนวนที่มากขึ้น เพื่อความถูกต้อง แม่นยำและมีประสิทธิภาพของระบบการนับปริมาณรถที่จะพัฒนาต่อไป

4. ในงานวิจัยนี้ใช้ข้อมูลการจราจรที่เป็นช่วงเวลากลางวันสภาพอากาศปกติจึงใช้ค่า Threshold ค่าเดียว แนวทางการพัฒนาต่อไปควรเพิ่มข้อมูลการจราจรที่สภาพอากาศเงื่อนไข

อื่นๆ เช่น กรณีฝนตก หรือช่วงเวลากลางคืน เป็นต้น เพื่อให้ระบบสามารถทำงานได้ทุกสภาพอากาศ

5. ในงานวิจัยนี้ใช้ การลบภาพพื้นหลังโดยใช้เทคนิคการหาผลต่างของเฟรม ทำให้สามารถนับปริมาณรถได้เฉพาะที่รถเคลื่อนที่เท่านั้น แนวทางการพัฒนาต่อไปควรใช้การลบภาพพื้นหลังโดยเพิ่มเทคนิค Selectivity เพื่อให้ระบบสามารถนับปริมาณรถที่หยุดนิ่งได้

6. ควรออกแบบพัฒนาอัลกอริทึมบนเอฟพีจีเอที่ใช้เทคนิคไปป์ไลน์ในทุกบล็อกย่อยๆของระบบ เนื่องจากข้อมูลในการคำนวณแต่ละบล็อกเป็นอิสระต่อกัน เพื่อให้ระบบสามารถประมวลผลได้เร็วขึ้น

บรรณานุกรม

[1] ธีรยศ เวียงทอง, "รู้จักกับการออกแบบร่วมกันระหว่างฮาร์ดแวร์-ซอฟตแวร์เบื้องต้น", ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร, 2547.

[2] Richard Hartley and Andrew Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2000.

[3] Castleman, K.R., "Digital Image Processing", N.J.: Prentice-Hall, 1996.

[4] David Pellerin and Scott Thibault , "Practical FPGA Programming in C", Prentice Hall, 2005.

[5] M. Piccard , "Background subtraction techniques: a review", In Proc. of IEEE SMC 2004 International Conference on Systems, Man and Cybernetics, Vol. 4, pp. 3099-3104 , 2004.

[6] Thou-Ho Chen, Yu-Feng Lin, and Tsong-Yi Chen, "Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance", Proceedings of the Second International Conference on Innovative Computing, Information and Control, Kaohsiung, Taiwan, pp. 238 – 238, 5-7 Sept 2007.

[7] Erhan Ba, A. Murat Tekalp, and F. Sibel Salman, "Automatic Vehicle Counting from Video for Traffic Flow Analysis", Proceedings of the 2007 IEEE Intelligent Vehicles Symposium Istanbul, Turkey, June 13-15 2007.

[8] Tae-Seung Lee, Eung-Min Lee, Hyeong-Taek Park, Young-Kil Kwag, Sang-Seok Lim, Joong-Hwan Baek, and Byong-Won Hwang, "Implementation of Traffic Flow Measuring Algorithm Using Real-Time Dynamic Image Processing", Springer-Verlag Berlin Heidelberg, Korea, pp. 78–87, 2003.

[9] Manoel E.de Lima, Pablo Viana da Silva, Alejandra C Frery, Edna Barros, "A Codesign Approach for a High Performance Vehicle Detector", High Perfomance Computing HPC2003-ASTC2003, San Diego, USA, January 2003.

[10] สราวุธ จันทร์สุวรรณ, Mr.Hajime Sakakibara และ ดร.เกษม ชูจารุกุล, "ระบบควบคุมสัญญาณไฟจราจรระบบ RONDO (ROLLING-HORIZON DYNAMICOPTIMAZATION OF SIGNAL CONTROL) : กรณีศึกษาการติดตั้งโครงการนำร่อง จังหวัดภูเก็ต (RONDO (ROLLING-HORIZON DYNAMIC OPTIMAZTION OF SIGNAL CONTROL): A CASE STUDY OF PHUKET PILOT POOJECT)" , โครงการบริษัท ซูมิโตโม อีเล็กตริก (ประเทศไทย), 2547.

[11] OLIVEIRA, J.P. PRINTES, A.L. FREIRE, R. C. S. MELCHER, E.U.K. SILVA, I.S.S. "FPGA Architecture for Object Segmentation in Real Time" ,In 14th European Signal Processing Conference - EUSIPCO, 2006, Florença. Proceedings of the 14th European Signal Processing Conference - EUSIPCO, pp. 1-4, Brasil , 2006.

[12] R. Tsai, "A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses", IEEE Journal of Robotics and Automation, pp. 323-344, 1987.

[13] Y. Benezeth, P.M. Jodoin, B. Emilel, H. Laurent and C. Rosenberger, "Review and Evaluation of Commonly-Implemented Background Subtraction Algorithms", Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, pp.1-4, 8-11 Dec 2008.

[14] R.Cucchiara, C.Grana, M.Piccardi, A.Prati, "Statistic and Knowledge-based Moving Object Detection in Traffic Scenes", Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE, pp.27-32, 2000.

[15] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in Proc. 1999 IEEE Conf. Computer Vision and Pattern Recognition, vol. 2, Fort Collins, pp. 246–252, June 23–25 1999.

[16] http://dx.sheridan.com/advisor/cmyk_color.html

[17] http://www.bobpowell.net/grayscale.htm

[18] http://www.impulsec.com

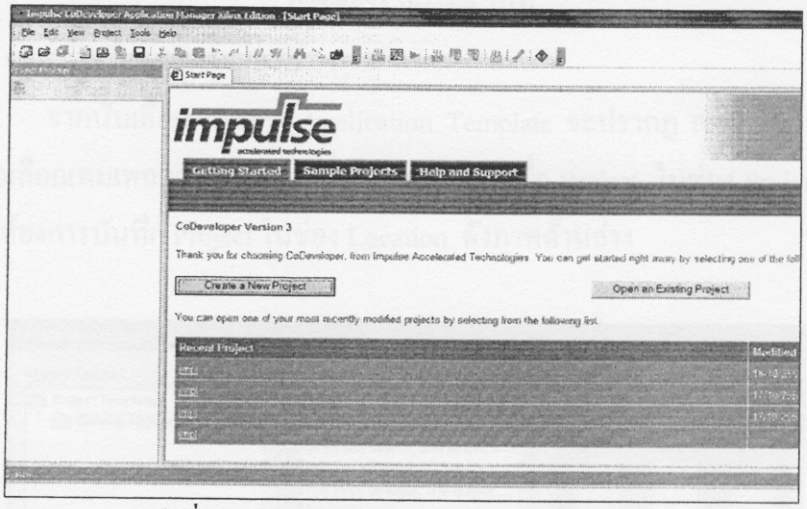
[19] http://www.xilinx.com

ภาคผนวก ก
รายละเอียดของโปรแกรมและซอฟต์แวร์ช่วยออกแบบ

การอ่านภาพและแสดงภาพ

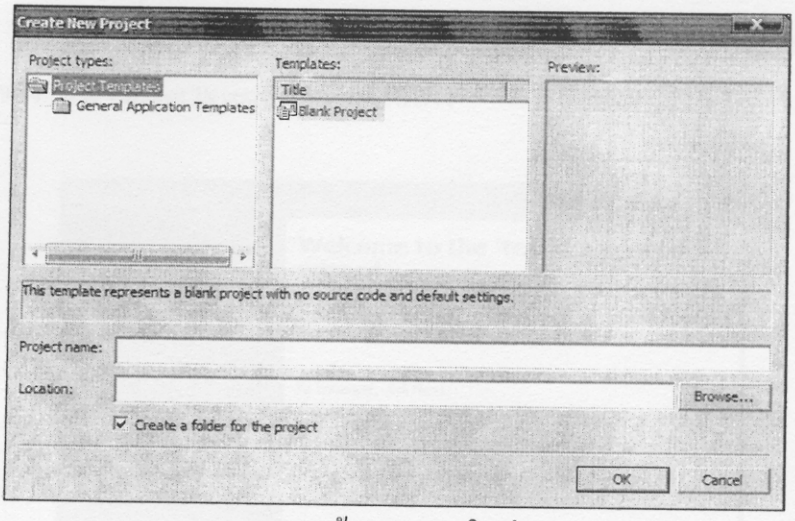
การอ่านหรือการเขียนภาพ หรือข้อมูลที่เรียกว่าสตรีมของแอพพลิเคชัน เมื่อทำการรันระบบเกิดขึ้นจะมีการส่งผ่านข้อมูลสตรีมโดยเป็นการวนซ้ำเพื่อทำการรับข้อมูลหรืออ่านข้อมูลภาพทีละจุดภาพ ที่สามารถอ่านภาพจากภาพหนึ่งภาพ หรือมากกว่าหนึ่งภาพก็ได้ เพื่อนำค่าแต่ละจุดภาพมาประมวลผลในระบบตามที่ต้องการ

ตัวอย่างการอ่านภาพและเขียนหรือแสดงภาพด้วยโปรแกรม ImpulseCนั้นสามารถทำได้โดย เปิดโปรแกรม Impulse CoDeveloper ที่มีหน้าเริ่มต้นดังภาพด้านล่าง



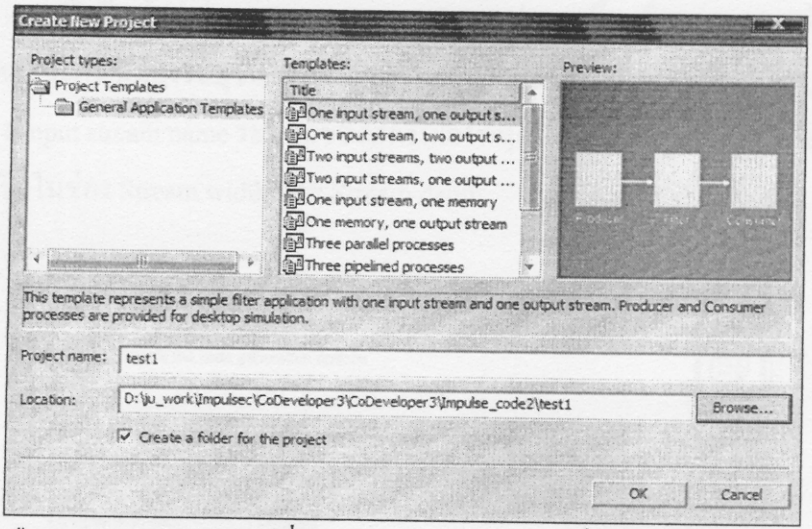
หน้าเริ่มต้นของโปรแกรม Impulse CoDeveloper

จากนั้นทำการสร้าง Project Template ด้วยโปรแกรม Impulse CoDeveloper โดยเลือก Create a New Project ในหน้าเริ่มต้นของโปรแกรมแล้วจะขึ้นเป็นหน้าที่ให้เลือกชนิดของ Project ที่จะสร้างดังภาพด้านล่าง



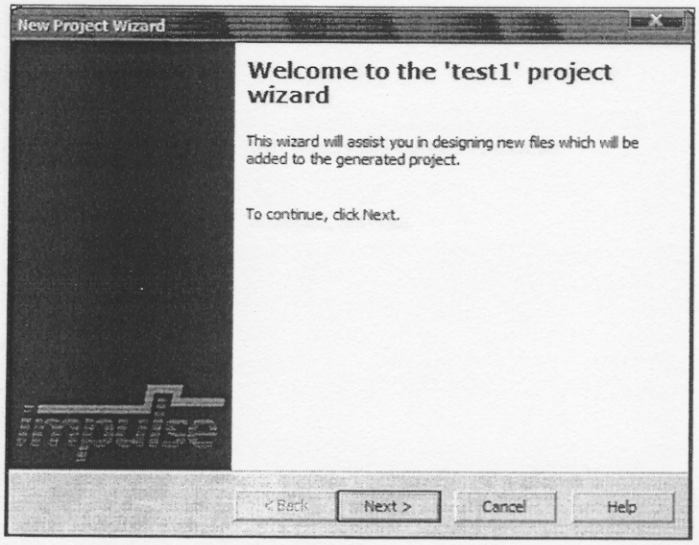
การสร้าง Project ใหม่

จากนั้นเลือก General Application Template จะปรากฏ Template รูปแบบที่มีอยู่อัตโนมัติ เมื่อเลือกเทมเพลตตามที่ต้องการแล้วจะต้องระบุชื่อ Project ในช่อง Project name และเลือกไดเรคที่ต้องการบันทึก Project ในช่อง Location ดังภาพด้านล่าง



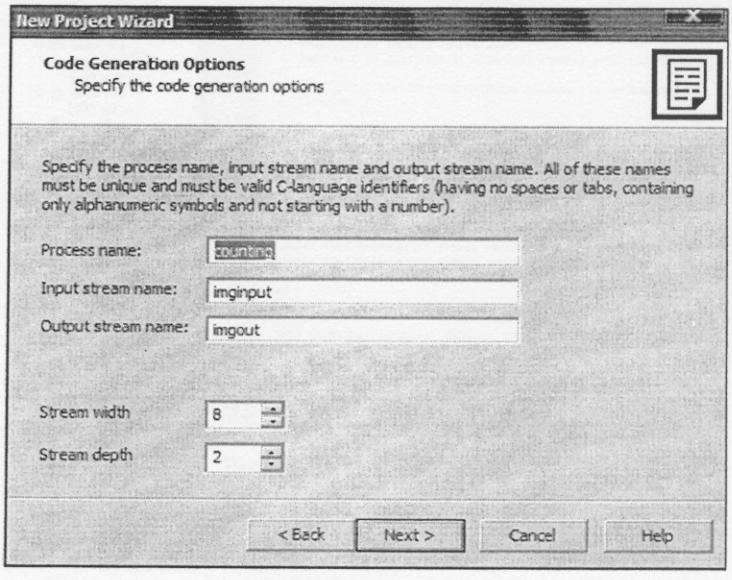
การเลือก Template และระบุชื่อProject และไดเรคทอรีที่ต้องการบันทึก Project

เมื่อทำการเลือกชนิด Template ของ Project ระบุชื่อและการบันทึก Project แล้ว จะมีหน้าต่างของ New Project Wizard ดังภาพด้านล่าง



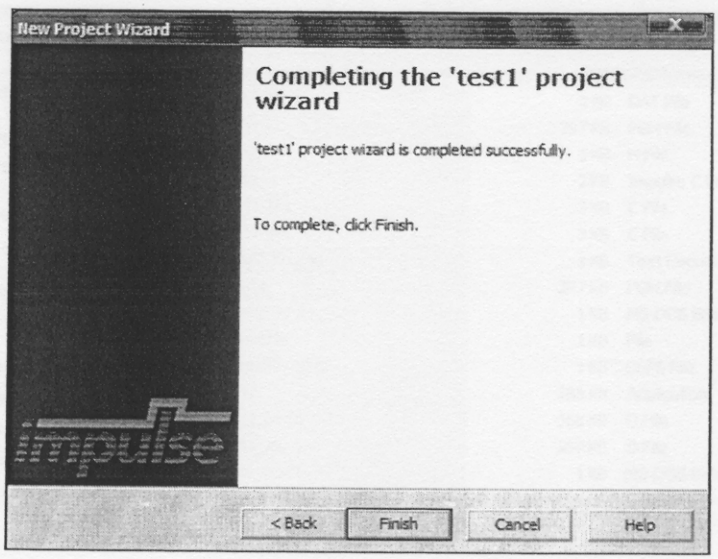
หน้าต่าง New Project Wizard

เมื่อมีหน้าต่างของ New Project Wizard ปรากฏขึ้น เลือก Next จะปรากฏหน้าต่าง แสดงดังภาพด้านล่าง ที่ต้องระบุชื่อฟังก์ชันการทำงาน Process Name จากนั้นระบุชื่อของสตรีมอินพุตในช่อง Input stream name ระบุชื่อของสตรีมเอาท์พุตในช่อง Output stream name ระบุความกว้างของสตรีมในช่อง Stream width และ Stream depth



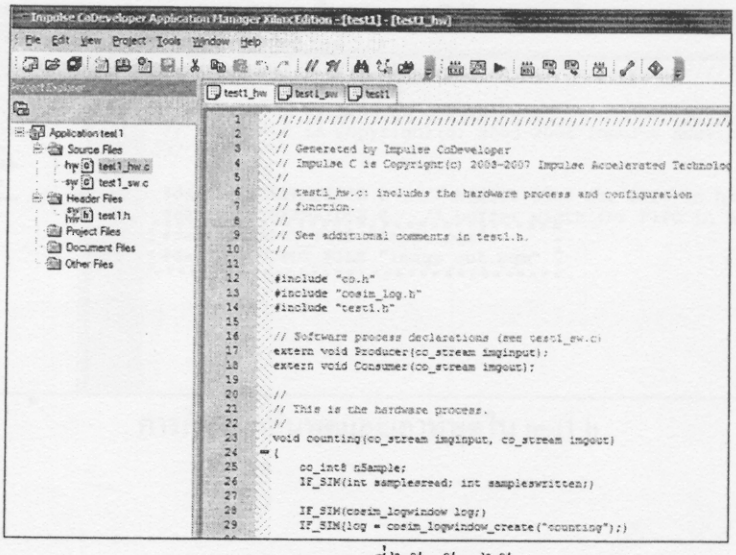
New Project Wizard

เมื่อระบุข้อมูลใน New Project Wizard ทุกช่องแล้วเลือก Next จะปรากฏหน้าต่างการสร้าง Project ที่เสร็จสมบูรณ์ดังภาพด้านล่าง



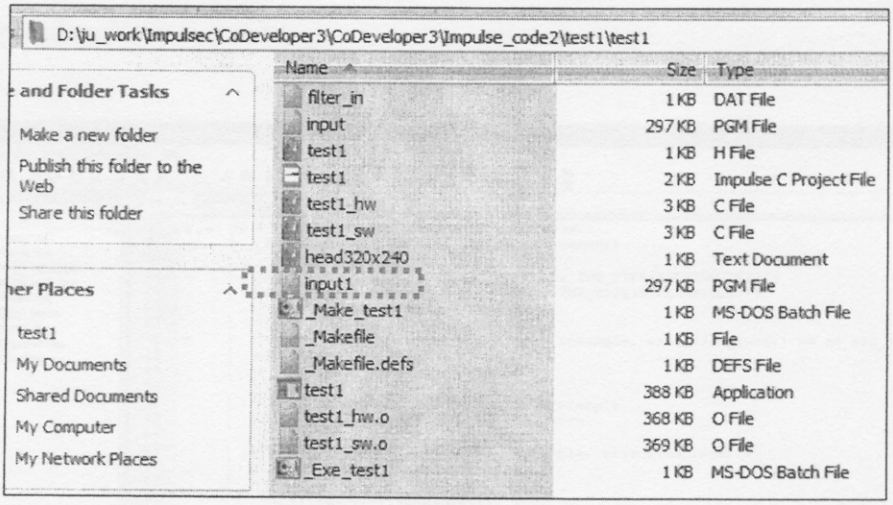
การสร้าง Project เสร็จสมบูรณ์

เมื่อเปิด Project ในไดเรคทอรีที่ได้บันทึกไว้ โปรแกรมจะมีการสร้าง Source File ให้อย่างอัตโนมัติที่ประกอบด้วย ไฟล์ test1_hw.c ไฟล์ test1_sw.c และไฟล์ test1.h



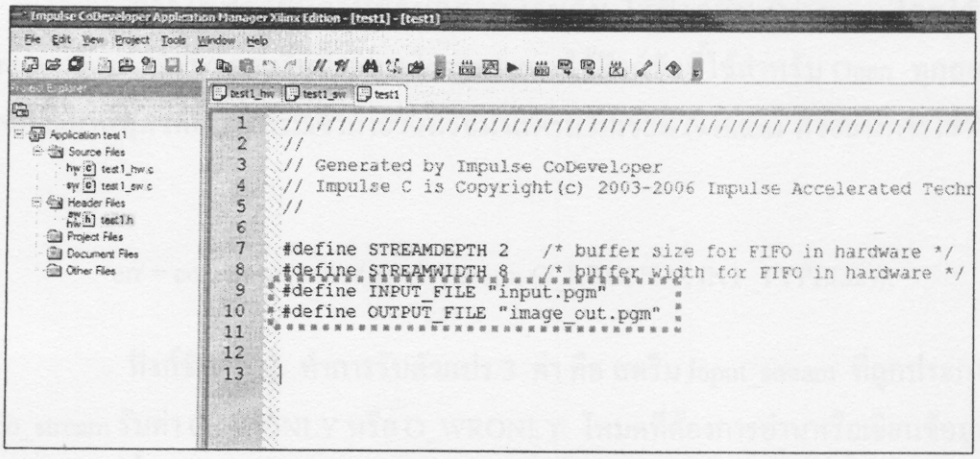
Project test1 ที่ได้สร้างไว้

การที่จะนำภาพมาประมวลผลนั้น จะต้องบันทึกภาพนั้นในไดเรคทอรีของ Project ด้วย ดังภาพประกอบด้านล่าง ตัวอย่างการนำภาพที่ชื่อ input1.pgm มาประมวลผล



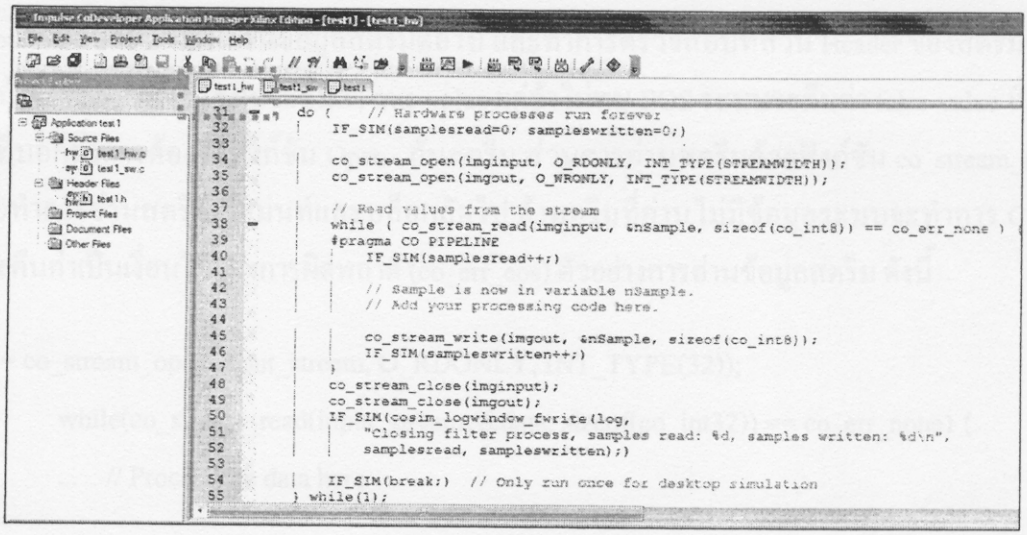
ไดเรคทอรีของ Project

จากนั้นจะต้องกำหนดชื่อสตรีมอินพุต ในไฟล์ test1.h ให้ตรงกับภาพอินพุตด้วย และกำหนดชื่อของไฟล์เอาท์พุตเช่นกัน ในตัวอย่างนี้กำหนดเป็น image_out.pgm



การกำหนดอินพุตและเอาท์พุตใน test1.h

ส่วนไฟล์ test1_hw.c เป็นส่วนของโค้ดของฟังก์ชัน ซึ่งเป็นการนำภาพอินพุตมาประมวลผลตามอัลกอริทึมที่ได้ออกแบบไว้ ซึ่งจะต้องมีการอ่านและเขียนสตรีมอินพุตเข้ามาก่อน ที่จะเข้าสู่ฟังก์ชัน และทำการเพิ่มส่วนของโค้ด ในส่วนข้อความ Add your processing code here ดังภาพประกอบด้านล่าง



test1_hw.c

การใช้ฟังก์ชันอ่านข้อมูลภาพ เริ่มต้น ใช้ฟังก์ชัน Open โดยใช้คำสั่ง co_stream_open เพื่อรีเซตสตรีมที่อยู่ภายในสเตท การใช้ฟังก์ชันนี้ใช้สำหรับ Open ทุกสตรีมทั้งอินพุตและเอาท์พุต เพื่อนำสตรีมเข้ามาอ่านหรือนำสตรีมไปเขียน (written) ตัวอย่างการใช้ฟังก์ชัน Open

err = co_stream_open(input_stream, O_RDONLY, INT_TYPE(32));

ฟังก์ชัน Open ทำการรับตัวแปร 3 ค่า คือ สตรีม input_stream ที่ถูกประกาศเป็นชนิด co_stream รับค่า O_RDONLY หรือ O_WRONLY โหมดที่ต้องการอ่านหรือเขียนข้อมูล และรับค่าชนิดของตัวแปรข้อมูล เช่น INT_TYPE(n), UINT_TYPE(n), CHAR_TYPE,FLOAT_TYPE, หรือ DOUBLE_TYPE ถ้าสตรีมถูก Open ไว้แล้วโปรแกรมจะคืนค่าเป็น co_err_already_open และเมื่อไม่ต้องการอ่านสตรีมนั้นแล้วจะต้องใช้ฟังก์ชัน Close โดยใช้คำสั่ง co_stream_close ดังตัวอย่าง

err = co_stream_close(input_stream);

ฟังก์ชัน Close จะเป็นการแสดงถึงการสิ้นสุดของสตรีม "end-of-stream" (EOS) ไปยังเอาท์พุตสตรีม ถ้าสตรีมถูก Close ไว้แล้วโปรแกรมจะคืนค่าเป็น co_err_already_open

อินพุตสตรีม

เมื่อมีอินพุตสตรีมเข้ามายังระบบจะมีการดำเนินการสองอย่าง คือ มีการตรวจสอบ EOS และการอ่านสตรีม ซึ่งการตรวจสอบ EOS เพื่อจะได้ทราบว่าเมื่อใดที่จะต้องมีการใช้ฟังก์ชัน Close เพื่อที่จะทำการเขียนข้อมูลสตรีมต่อไป และทำการตรวจสอบที่ส่วน Header ของสตรีมด้วย ซึ่งถ้าตรวจพบ EOS ระบบจะคืนค่า true value แต่ถ้าไม่พบ EOS ระบบจะคืนค่า false value ที่เป็นค่าที่บอกถึงการต้องใช้ฟังก์ชัน Open กับสตรีม ส่วนการอ่านสตรีมด้วยฟังก์ชัน co_stream_read เพื่อทำการอ่านสตรีมอีลีเมนท์และบล็อกถัดไป ถ้าสตรีมที่อ่านไม่มีข้อมูลระบบจะทำการ Close และคืนค่าเป็นเงื่อนไขของการผิดพลาด (co_err_eos) ตัวอย่างการอ่านข้อมูลสตรีม ดังนี้

```
err = co_stream_open(input_stream, O_RDONLY, INT_TYPE(32));
        while(co_stream_read(input_stream, &data, sizeof(co_int32)) == co_err_none) {
        ...  // Process the data here
}
co_stream_close(input_stream);
```

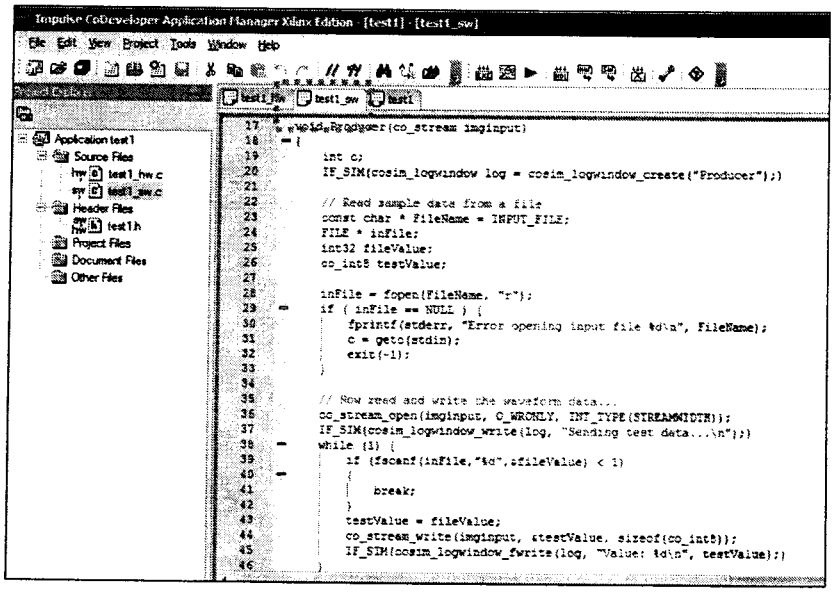
เอาท์พุตสตรีม

การแสดงภาพหรือเอาท์พุตสตรีมได้โดยใช้ฟังก์ชันwrite ที่มีคำสั่ง co_stream_write และตัวอย่างการแสดงข้อมูลภาพหรือสตรีมดังตัวอย่าง

```
co_stream_open(output_stream, O_WRONLY, INT_TYPE(32));
        for (i=0; i < ARRAYSIZE; i++) {
        co_stream_write(output_stream, &data[i], sizeof(co_int32));
        }
co_stream_close(output_stream);
```

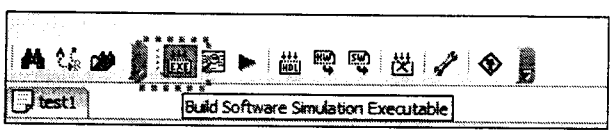
สตรีมที่เป็นข้อมูลภาพจะถูกเขียนค่าโดยมีชนิดตามที่ได้กำหนดไว้ในที่นี้ได้กำหนดตัวแปรแบบ int ขนาด 32 บิต

ในส่วนของไฟล์ test1_sw.c เป็นส่วนของโค้ด Producer และ Consumer ที่จัดการและตรวจสอบไฟล์อินพุตและไฟล์เอาท์พุตสามารถเปลี่ยนแปลงตัวแปรของข้อมูลภาพที่ต้องการนำมาประมวลผลได้ ดังภาพประกอบด้านล่าง



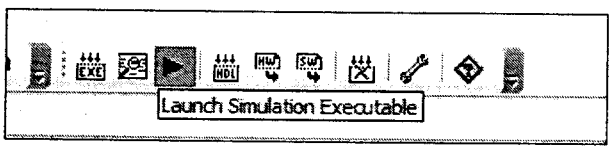
test1_sw.c

เมื่อเขียนโปรแกรมในส่วนของ test1_sw.c test1_hw.c และ test1.h เสร็จแล้วทำการคอมไพล์โค้ด โดยเลือก Build Software Simulation Executable ดังภาพด้านล่าง



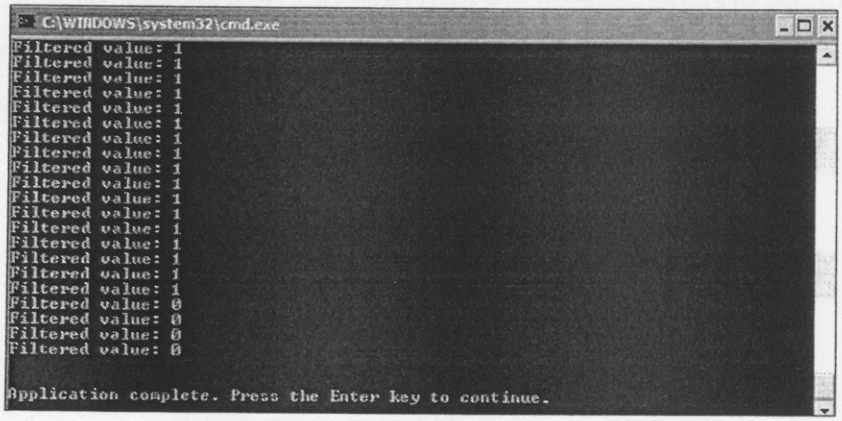
Build Software Simulation Executable

จากนั้นทำการรันโปรแกรมโดยเลือก Launch Simulation Executable ดังภาพประกอบด้านล่าง



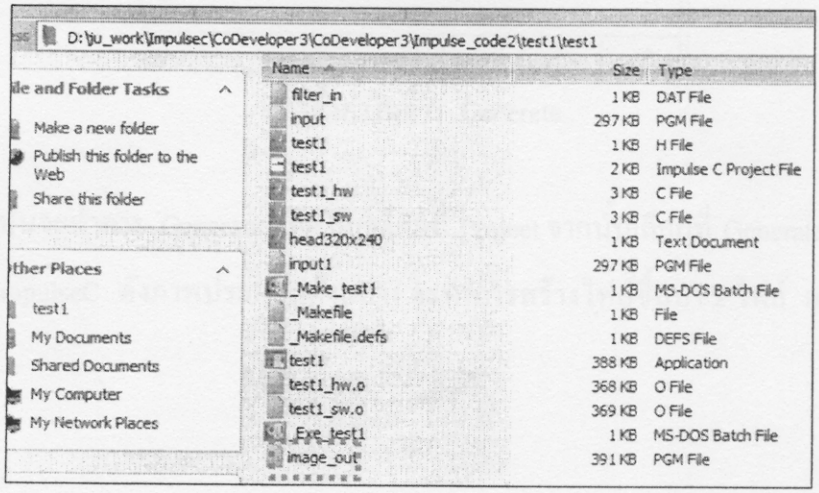
Launch Simulation Executable

ผลจากการรันโปรแกรมจะแสดงหน้าต่าง ดังภาพด้านล่าง จากตัวอย่างเป็นข้อมูลของภาพแต่ละจุดของภาพ



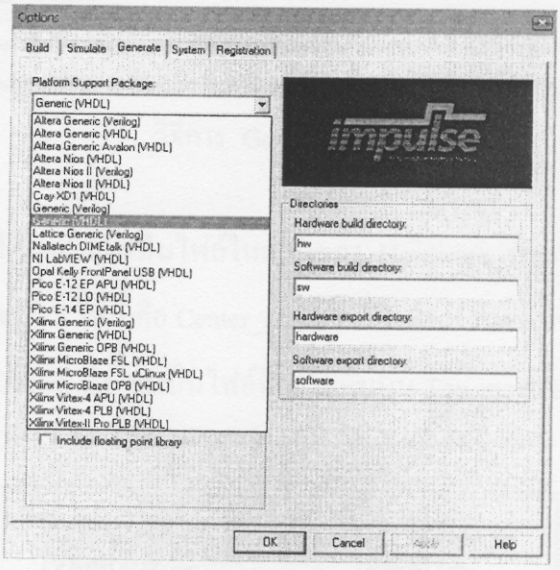
ผลการรันโปรแกรม

และไฟล์ของภาพเอาท์พุตจะถูกเขียนไว้ในไดเรคทอรีของ Project ดังภาพประกอบด้านล่าง ตัวอย่างเอาท์พุตของภาพ image_out.pgm



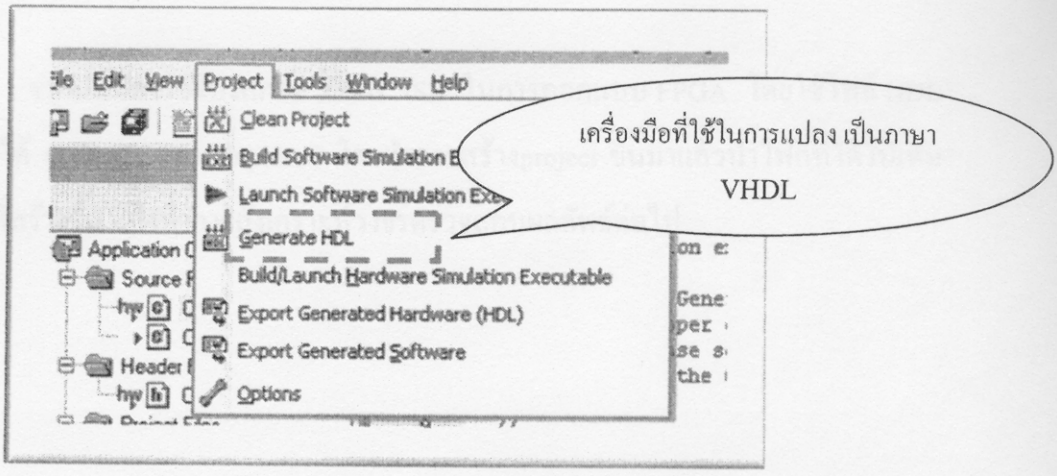
Generate HDL ด้วย ImpulseC

การ Generate ภาษาซีเป็นภาษา HDL ด้วย ImpulseC เนื่องจาก มีส่วนที่ช่วยในการแปลงเป็นภาษาอื่นที่ Hardware สามารถอ่านแล้วเข้าใจได้ โดยสามารถเลือกภาษาได้ใน Project>>Options>>Generate ดังภาพประกอบด้านล่าง



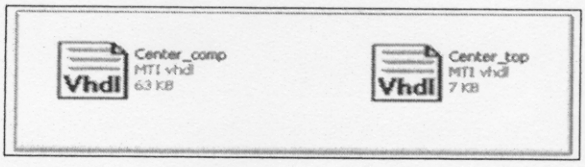
การตั้งค่าในการ Generate

จากนั้นจะทำการ Generate โค้ดโดยเลือกที่ Project จากนั้นเลือกที่ Generate HDL ในตัวโปรแกรม ImpulseC ดังภาพประกอบด้านล่าง จะทำการสร้างไฟล์ขึ้นมา 2 ไฟล์ และสร้างไฟล์ไลบรารี



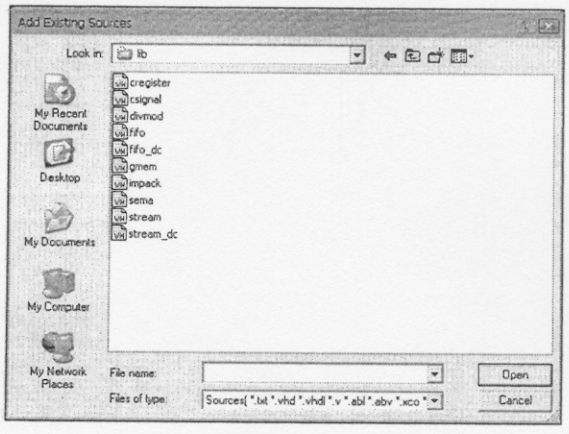
วิธีการ Generate HDL

ไฟล์ที่ได้มานั้นจะเป็นไฟล์ในส่วนของ Hardware ที่จะทำการโหลดลงบอร์ดในขั้นต่อไป ซึ่งไฟล์ที่นั้นจะมี 2 ไฟล์ คือ Center_comp จะเป็นในส่วนของเฉพาะที่เขียนขึ้นมาและมีผลกับการทำงาน ส่วน Center_top จะเป็นไฟล์ที่อยู่ระดับบน Top module โดยทำการรวมทุกส่วนเข้าด้วยกัน



ไฟล์ที่ได้จากการ Generate

และ ImpulseC จะ generate ไฟล์ไลบรารีในไดเรคทอรีของ Project ดังภาพประกอบด้านล่าง



ไฟล์ Library

จากนั้นต้องใช้โปรแกรม Xilinx ISE ในการออกแบบ FPGA โดยใช้ไฟล์ HDL และไลบรารีที่ได้ Generate จาก ImpulseC โดยทำการสร้างproject ขึ้นมาแล้วนำไฟล์ที่ได้ไปเพิ่มลงใน project ที่สร้างขึ้น แล้วทำการสังเคราะห์วงจรตรวจสอบผลลัพธ์ต่อไป

ภาคผนวก ข
งานวิจัยที่ได้รับการตีพิมพ์

การออกแบบการประมวลผลภาพวิดีโอด้วยภาษาระดับสูงสำหรับระบบนับปริมาณรถแบบฝังตัว

A System-Level Design of Video Processing for an Embedded Vehicle Counting System

ชลธิศา เวทโอสถ[1] ณัฏฐา จินดาเพ็ชร์[1] และ นิคม สุวรรณวร[2]
[1]ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์
ต.คอหงส์ อ.หาดใหญ่ จ.สงขลา 90112 โทรศัพท์: 0-7428-7045
[2]ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์
ต.คอหงส์ อ.หาดใหญ่ จ.สงขลา 90112 โทรศัพท์: 0-7428-7076
Email : i_juju10@hotmail.com snattha@eng.psu.ac.th kom@coe.psu.ac.th

บทคัดย่อ

บทความนี้นำเสนอการออกแบบระบบการนับปริมาณรถอย่างง่ายโดยใช้กระบวนการอิมเมจโปรเซสซิง ด้วยภาษาระดับสูง ImpulseC เพื่อช่วยในการออกแบบและพัฒนาร่วมกันระหว่างซอฟท์แวร์และฮาร์ดแวร์ทำให้ช่วยลดระยะเวลาและความยุ่งยากในการสร้างวงจร วงจรที่ได้สามารถพัฒนาบนเทคโนโลยีเอฟพีจีเอเพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น โดยบทความนี้ใช้เทคนิคการลบภาพพื้นหลังด้วยวิธีหาผลต่างเฟรม การแก้ไขมุมมองภาพแบบ Perspective เพื่อให้รถในบริเวณที่ภาพรถอยู่ไกลจากกล้องซึ่งรถมีขนาดลดลง และทำการนับด้วยการหาพื้นที่จุดสีของรถ ซึ่งสามารถจำลองการทำงานได้จริงบนเทคโนโลยีเอฟพีจีเอ ก่อนนำไปใช้งานบนบอร์ด Virtex II Pro

คำสำคัญ: ImpulseC, การนับปริมาณรถ, การลบภาพพื้นหลัง, เอฟพีจีเอ

Abstract

This paper presents a design methodology of a simple vehicle counting circuit on a Field Programmable Gate Array (FPGA) using system-level language named ImpulseC. By using this, it could accelerate image processing system performance and also reduce the design development time. We use the frame different based background subtraction to separate the vehicles from the road, apply perspective view corrected for vehicles far from the camera. The vehicle count is calculated from the area of normalized pixels. This design flow allows completely simulating on FPGA technology before applying on the Virtex II Pro board.

Keywords: ImpulseC, Vehicle Counting, Background Subtraction, FPGA

1. บทนำ

ปัญหาการจราจรที่ติดขัดเป็นปัญหาสำคัญ โดยเฉพาะในช่วงเวลารีบเร่งที่มีปริมาณรถยนต์ในการจราจรมากกว่าช่วงปกติ ทำให้เกิดปัญหามากมาย เช่น ไม่มีความคล่องตัวในการจราจร สิ้นเปลืองพลังงาน เกิดอุบัติเหตุได้ง่าย เกิดปัญหามลภาวะ เป็นต้น หากมีระบบที่สามารถจัดการกับการปล่อยรถออกจากแยกได้อย่างชาญฉลาด ปัญหาเหล่านี้ก็จะได้รับการแก้ไขไปในทางที่ดีขึ้น เป็นเหตุให้เกิดแนวความคิดในการจัดการการจราจรอัจฉริยะ

หลักการทำงานของระบบนี้สามารถคำนวณความหนาแน่นของปริมาณรถในสี่แยกจราจร เพื่อทำการปล่อยรถยนต์ให้มีประสิทธิภาพมากกว่าในปัจจุบัน ที่ติดกล้องเอาไว้ตามมุมสูงในแยกต่างๆ จะทำหน้าที่จับภาพรถบนท้องถนนแล้วส่งสัญญาณภาพไปประมวลผลภาพบนเซิร์ฟเวอร์ที่ศูนย์ควบคุมกลาง [2]-[4] โดยหากแยกใดมีปริมาณรถหนาแน่น ระบบจะทำการปล่อยสัญญาณไฟเขียวเพื่อให้รถออกไปมากกว่าแยกที่มีปริมาณรถน้อย แต่พบว่าระบบดังกล่าวเซิร์ฟเวอร์ต้องรับภาระงานหนักมากจึงไม่เหมาะกับถนนที่มีทางแยกจำนวนมาก

ในงานวิจัยนี้ เป็นส่วนหนึ่งของระบบสัญญาณไฟจราจรอัจฉริยะ คือ ในส่วนของการตรวจนับรถยนต์เพื่อนำไปคำนวณหาความเหมาะสมในการจัดการระบบการจราจร ขั้นตอนการทำงานของระบบเริ่มจาก สัญญาณภาพแต่ละแยกที่ได้มาจะถูกประมวลผลโดยใช้กระบวนการอิมเมจโปรเซสซิง บนบอร์ดเอฟพีจีเอ ที่ช่วยประมวลผลแทนเซิร์ฟเวอร์ จากข้อมูลปริมาณรถยนต์ในแต่ละแยกนี้เซิร์ฟเวอร์ควรจะสามารถประมวลผลในภาพรวมทั้งระบบเพื่อจัดการจราจรที่มีประสิทธิภาพต่อไป

2. งานและทฤษฎีที่เกี่ยวข้อง

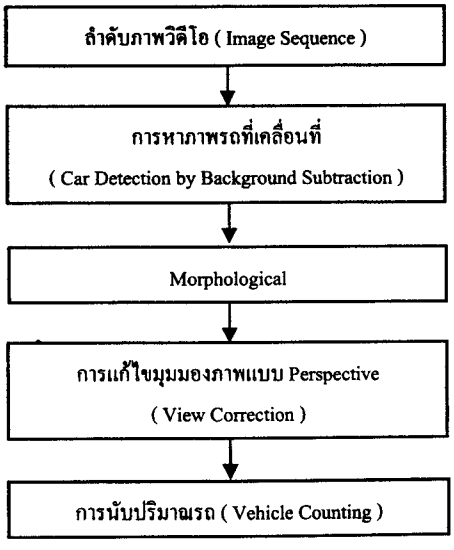
งานวิจัยที่เกี่ยวข้องกับการนับปริมาณรถด้วยกระบวนการอิมเมจโปรเซสซิง ดังในงานวิจัย[2] ใช้เทคนิคการลบภาพพื้นหลังโดยใช้ผลต่างของเฟรม เนื่องจากภาพพื้นหลังค่อนข้างคงที่ นอกจากนี้

DS 026

งานวิจัยดังกล่าวได้มีการวิเคราะห์บล็อบ และทำการนับรถเมื่อรถผ่านเส้นอ้างอิง แต่มีการนับผิดพลาดประมาณ 10% เนื่องจากการที่สีของภาพพื้นหลังใกล้เคียงกัน งานวิจัย [3] เป็นการตรวจหารถเคลื่อนที่โดยใช้โมเดลของภาพพื้นหลัง (Gaussian Mixture Modeling) รวมถึงการใช้เทคนิคการวิเคราะห์บล็อบ และคาลมานฟิลเตอร์ ในการติดตามรถที่เคลื่อนที่ ซึ่งเป็นกระบวนการที่ค่อนข้างซับซ้อน งานวิจัย [4] เป็นการตรวจหารถเคลื่อนที่โดยเปรียบเทียบจุดสีของรถกับถนน และใช้บริเวณของวัตถุที่เคลื่อนไหวมานับจำนวนจุดสีเพื่อหาพื้นที่ของรถ พบว่ามีความผิดพลาดในการนับรถประมาณ 2% ซึ่งน้อยมาก และมีการหาความกว้างและความยาวของรถเพื่อจำแนกประเภทรถ งานวิจัย [5] เป็นการออกแบบระบบร่วมฮาร์ดแวร์/ซอฟต์แวร์ ในการนับจำนวนรถ โดยใช้วิธีการ 2D FIR ฟิลเตอร์ และฮิสโตแกรมเทรสโฮลด์ โดยทำการนับรถเมื่อรถมีการเคลื่อนที่ผ่านบริเวณสี่เหลี่ยมที่กำหนดขึ้น ซึ่งออกแบบในแพลตฟอร์มของ DSP บนเอฟพีจีเอ ที่เพิ่มประสิทธิภาพในการประมวลผล ซึ่งผู้เขียนได้เลือกใช้เทคนิคการลบภาพพื้นหลัง ด้วยวิธีหาผลต่างของเฟรม [2] เลือกเทคนิคการนับปริมาณรถด้วยวิธีการนับจำนวนจุดสีเพื่อหาพื้นที่ของรถ [4] และพัฒนาอัลกอริทึมการนับปริมาณรถบนเอฟพีจีเอ ที่เพิ่มประสิทธิภาพในการประมวลผล [5]

3. การออกแบบอัลกอริทึมของการนับปริมาณรถ

อัลกอริทึมการนับปริมาณรถโดยภาพรวมแสดงดังรูปที่ 1 เริ่มต้นแยกรถที่เคลื่อนที่ด้วยการลบภาพพื้นหลังโดยการหาผลต่างระหว่างเฟรม ผลที่ได้นำมาทำกระบวนการ Morphological เพื่อปรับปรุงรูปร่างของวัตถุ จากนั้น การแก้ไขมุมมองภาพแบบ Perspective ของจุดสีในแต่ละแถว เพื่อขยายภาพรถในเลนถนนให้มีสัดส่วนเท่ากันทั่วทั้งภาพ และหาพื้นที่จำนวนจุดสีของรถทั้งหมดเพื่อทำการนับจำนวนรถ โดยเปรียบเทียบพื้นที่จำนวนจุดสีของรถทั้งหมดกับพื้นที่จำนวนจุดสีของรถ 1 คัน ซึ่งจะได้ปริมาณรถทั้งหมดใน 1 เฟรม



รูปที่ 1 แผนภาพอัลกอริทึมการนับปริมาณรถ

3.1 การหาภาพรถที่เคลื่อนที่

เพื่อแยกความแตกต่างของจุดสี ระหว่างรถที่เคลื่อนที่และ background วิธีนี้จะใช้ background ที่ประมาณค่าได้เป็นเฟรมก่อนหน้าเพื่อง่ายต่อการประมวลผล สำหรับขั้นตอนการประมวลผลจำเป็นจะต้องทำให้ภาพเป็นเกรสเกล เพื่อลดหน่วยความจำที่ใช้ ซึ่งใช้ 8 บิต ซึ่งมีสมการดังนี้

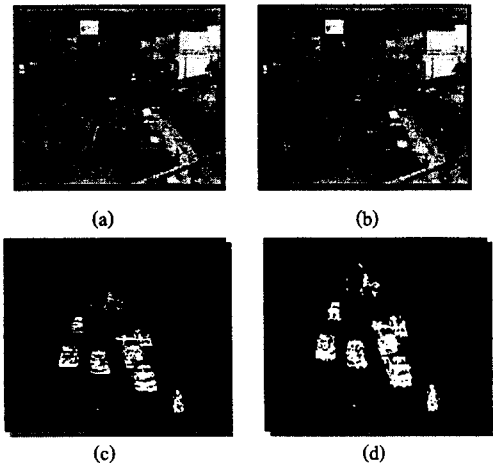
$$\left|frame_i - frame_{i-1}\right| > T \qquad (1)$$

เมื่อ $frame_i$ คือ เฟรมปัจจุบัน

$frame_{i-1}$ คือ เฟรมก่อนหน้า และ

T คือ ค่า threshold

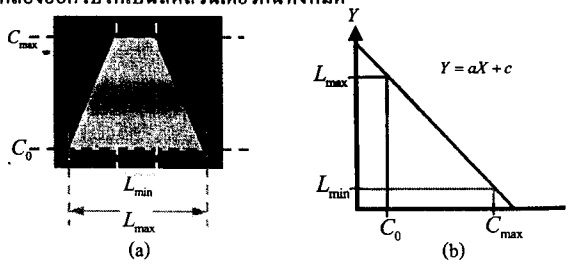
โดยรูปที่ 2 คือ กระบวนการหาผลต่างของเฟรมแล้วนำมาเทียบกับค่า Threshold ที่มีค่าเท่ากับ 25 ซึ่งได้ภาพรถที่เคลื่อนที่ดังรูปที่ 2 (c) ที่มีช่องว่างของรถจะต้องนำมาทำกระบวนการ Morphological โดยใช้โอเปอร์เรชัน Close ด้วยเทมเพลตเมตริกซ์ขนาด 2×2 เพื่อปิดช่องว่างรถดังรูป 2 (d)



รูปที่ 2 ผลจากการหาผลต่างของเฟรม (a) คือ เป็นภาพเฟรมก่อนหน้า (b) เป็นภาพของเฟรมปัจจุบัน (c) เป็นผลต่างของเฟรมปัจจุบันที่ลบด้วยเฟรมก่อนหน้าแล้วเปรียบเทียบด้วยค่า Threshold และ (d) การใช้โอเปอร์เรชัน Close เพื่อปิดช่องว่างในรถ

3.2 การแก้ไขมุมมองภาพแบบ Perspective

เนื่องจากภาพจากกล้องวิดีโอ ดังรูปที่ 3 รถที่อยู่บริเวณด้านหน้ากล้องจะมีขนาดใหญ่ และจะมีขนาดเล็กลงเรื่อยๆ เมื่อรถไกลจากกล้องวิดีโอ ซึ่งจำนวนพิกเซลของรถ 1 คัน จะมีขนาดลงลงเมื่อรถอยู่ไกลจากกล้องออกไป ซึ่งมีความสัมพันธ์เป็นไปในลักษณะเชิงเส้น ดังนั้นจึงต้องหานอร์มอลไลซ์แฟคเตอร์ที่จะต้องนำมาคูณกับจำนวนพิกเซลในแต่ละแถว เพื่อเป็นการขยายภาพของรถในบริเวณที่อยู่ไกลกล้องออกไปให้เป็นสัดส่วนเดียวกันทั้งหมด



รูปที่ 3 บริเวณที่ต้องการนับปริมาณรถนำค่านำค่าจุดสีพล็อตกราฟเส้นตรง (a) กำหนดตัวแปรในบริเวณที่ต้องการนับ และ (b) นำค่าจุดสีมาพล็อตกราฟเส้นตรง

จากรูปที่ 3 สามารถหาความสัมพันธ์สมการเส้นตรง

$$Y = aX + c \quad (2)$$

เมื่อนำบริเวณของพื้นที่ที่เลือกมาพล็อตกราฟได้ดังรูปที่ 3 (b) ซึ่งค่าจำนวนจุดสีในแต่ละแถวนำมาพล็อตในแกน Y ส่วนจำนวนแถวของบริเวณที่เลือกในภาพจะนำมาพล็อตในแกน X จะได้เส้นตรงดังรูป ซึ่งสามารถนำมาเขียนสมการได้ดังสมการ (3)

$$Y = (\frac{L_{\min} - L_{\max}}{C_{\max}})X + L_{\max} \quad (3)$$

ดังนั้น จะได้นอร์มอลไลซ์แฟคเตอร์ ดังสมการ (4)

$$F_x = \frac{L_{\max}}{(\frac{L_{\min} - L_{\max}}{C_{\max}})X + L_{\max}} \quad (4)$$

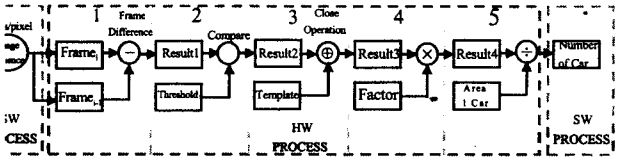
เมื่อ $X = 0$ ถึง $C_{\max}$

4. การพัฒนาอัลกอริทึมนับปริมาณรถบนเอฟพีจีเอ

การออกแบบและพัฒนาระบบร่วมฮาร์ดแวร์/ซอฟต์แวร์ [1] ในงานวิจัยนี้ใช้ภาษา ImpulseC [6] โดยแบ่งออกเป็นสองส่วนหลักคือ ระบบซอฟท์แวร์ และระบบฮาร์ดแวร์ โดยมีพื้นฐานมาจากการเขียนด้วยภาษา C ซึ่งกระบวนการทำงานของ CoDeveloper [7] สามารถนำอัลกอริทึมในส่วนของระบบฮาร์ดแวร์มาสร้างในรูปของภาษาทางฮาร์ดแวร์ เป็นภาษา VHDL หรือ Verilog ได้ และสามารถทำการจำลองแบบการทำงานระดับระบบ (System – Level Simulation) ได้

4.1 ออกแบบและพัฒนาวงจรการนับจำนวนรถ

เมื่อพัฒนาการนับปริมาณรถด้วยภาษา ImpulseC แล้ว จากนั้นใช้โปรแกรม Impulse CoDeveloper แปลภาษาเพื่อให้ได้วงจรที่อธิบายด้วยภาษาทางฮาร์ดแวร์ ในที่นี้เลือกใช้เป็นภาษา Verilog จากนั้นใช้ซอฟต์แวร์ที่มีชื่อเรียกว่า Xilinx ISE [8] ช่วยในการสังเคราะห์วงจรและจำลองการทำงาน ซึ่งขั้นตอนการออกแบบวงจรบนเทคโนโลยีเอฟพีจีเอสามารถอธิบายได้ตามไดอะแกรม ดังรูปที่ 4 เป็นการออกแบบวงจรนับปริมาณรถแบบตามลำดับ ที่มี 5 โอเปอร์เรเตอร์ เพื่อให้สามารถรองรับการทำงานแบบไปป์ไลน์ ส่วนในกรณีที่ไม่ใช้ไปป์ไลน์ ผลลัพธ์ Result1-5 ในรูปที่ 4 สามารถใช้หน่วยความจำตัวเดียวกันได้ เพื่อเป็นการประหยัดหน่วยความจำ



รูปที่ 4 ไดอะแกรมอัลกอริทึมการนับรถบนเอฟพีจีเอแบบตามลำดับ

5. ผลการทดลอง

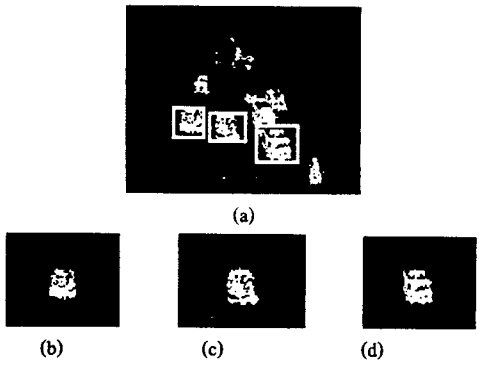
เมื่อทดสอบอัลกอริทึมการนับปริมาณรถกับเฟรมภาพวิดีโอขนาด 320×240 ซึ่งเป็นบริเวณแยกห้างสรรพสินค้าคาร์ฟูร์ หาดใหญ่ โดยนับจำนวนจุดสีในทุกๆแถวของภาพ ในบริเวณที่ต้องการนับปริมาณรถ ในตำแหน่งกล้องคงที่ ดังรูปที่ 3 (a) ซึ่งมีคุณสมบัติคือ บริเวณที่ต้องการนับปริมาณมี 145 แถว และจำนวนจุดสีมีจำนวนตั้งแต่ 53 – 165 จากนั้น นำค่าจำนวนแถวทั้งหมดของบริเวณที่ต้องการนับปริมาณรถมาพล็อตกราฟในแกน X และจำนวนของรถในแต่ละแถวมาพล็อตกราฟในแกน Y ดังรูปที่ 3 (b)

จะได้ว่า $L_{\max} = 165$, $L_{\min} = 53$ และ $C_{\max} = 144$

เมื่อแทนค่าในสมการ (4) จะได้

$$F_x = \frac{165}{(\frac{-112}{144})X + 165} \quad \text{เมื่อ X = 0 – 144}$$

ซึ่งต้องนำค่า F_x ไปคูณกับจำนวนจุดสีของรถในแต่ละแถว จะได้บริเวณพื้นที่ของรถทั้งหมด ที่มีการขยายพื้นที่ของรถในบริเวณที่ไกลจากกล้องให้มีขนาดเดียวกัน เมื่อได้พื้นที่โดยรวมของรถทั้งหมด ในแต่ละเฟรมภาพ จะต้องนำมาเปรียบเทียบด้วยพื้นที่ของรถ 1 คัน ผลลัพธ์ที่ได้ก็จะเป็นจำนวนรถในแต่ละเฟรมภาพ โดยค่าเฉลี่ยของรถ 1 คัน มาจากพื้นที่เฉลี่ยของรถ 3 คัน ดังรูปที่ 5 (a) - (d)

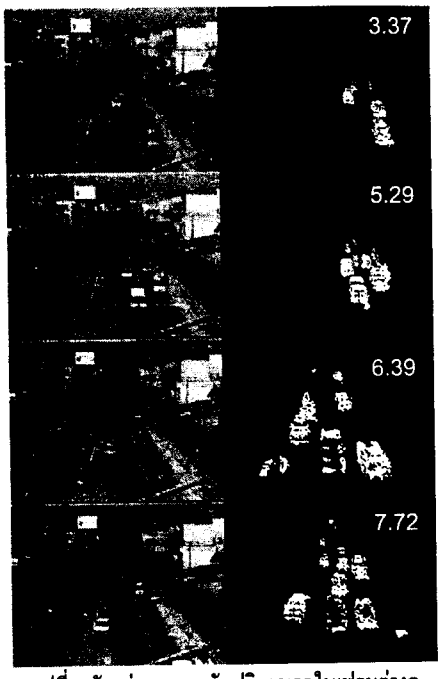


รูปที่ 5 การเลือกบริเวณพื้นที่เฉลี่ยของรถ 1 คัน (a) คือ บริเวณที่ทำการเลือกพื้นที่รถแต่ละคัน (b) คือ รถที่เลือกคันที่ 1 (c) คือ รถที่เลือกคันที่ 2 (d) คือ รถที่เลือกคันที่ 3

เมื่อเลือกเฟรมที่มีจำนวนรถ ตั้งแต่ 3 – 10 คัน ที่แตกต่างกัน 10 กรณี ผลลัพธ์ที่ได้แสดงดังตารางที่ 1 และรูปที่ 6

ตารางที่ 1 ผลการนับปริมาณรถตามจำนวน 3 – 10 คัน ที่แตกต่างกัน 10 กรณี จาก CoDeveloper

จำนวนรถภายในเฟรมภาพ (คัน)	จำนวนรถที่นับได้โดยเฉลี่ย 10 กรณี (คัน)
3	3.46
4	4.05
5	5.35
6	5.21
7	6.19
8	6.11
9	7.37
10	8.41



รูปที่ 6 ตัวอย่างผลการนับปริมาณรถในเฟรมต่างๆ

ตารางที่ 2 ผลการสังเคราะห์วงจรบนเอฟพีจีเอ Virtex II Pro XC2VP30

Latency (s)	Throughput (frame/s)	Slices/ Flip Flops	BRAMs	MULT 18×18s
0.81	1.23	4940/2332	41	8

จากตารางที่ 1 พบว่า การนับปริมาณรถมีการผิดพลาดไม่เกิน 2 คัน ทั้งนี้เนื่องมาจากสภาพกล้องที่ขุ่นมัวทำให้ภายในภาพเห็นรถไม่ชัดเจน รถที่มีสีใกล้เคียงกับพื้นถนนเมื่อหาผลต่างของเฟรมแล้วทำให้บางส่วนของรถหายไป และไม่ได้แยกประเภทรถจักรยานยนต์ รถบัส และรถบรรทุก

จากตารางที่ 2 แสดงจำนวนทรัพยากรที่จะใช้งานบนบอร์ด VirtexII Pro XC2VP30 โดยจำนวน Slices เป็นตัวบอกถึงพื้นที่โดยรวมที่จะใช้งานทรัพยากร ประกอบด้วย จำนวนการใช้ลอจิก Flip Flops , แสดงจำนวนการใช้บล็อกแรม (BRAMs) แต่ละบล็อกแรมมีขนาด 18 กิโลไบต์ , แสดงการใช้บล็อกตัวคูณ (MULT18×18) แต่ละบล็อกตัวคูณมีขนาด 18×18บิต และLatency(s)เป็นตัวบอกความเร็วของวงจรในการประมวลผลภาพต่อหนึ่งเฟรม

เมื่อตรวจสอบความถูกต้องของผลลัพธ์แล้ว จะเข้าสู่ขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอตามรูปที่ 4 บนโปรแกรม Xilinx ISE เพื่อทำการ Simulation ก่อนนำไปใช้งานจริงบนบอร์ดเอฟพีจีเอ

6. บทสรุป

บทความนี้นำเสนอการออกแบบระบบการนับปริมาณรถอย่างง่ายโดยใช้กระบวนการอิมเมจโปรเซสซิง ด้วยภาษาระดับสูง ImpulseC เพื่อช่วยในการออกแบบและพัฒนาร่วมกันระหว่างซอฟท์แวร์และฮาร์ดแวร์ทำให้ช่วยลดระยะเวลาและความยุ่งยากในการสร้างวงจร วงจรที่ได้สามารถพัฒนาบนเทคโนโลยีเอฟพีจีเอเพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น และผลที่ได้มีความผิดพลาดในการนับปริมาณรถเล็กน้อยไม่เกิน 2 คันในสภาพกล้องที่ขุ่นมัว

7. แนวทางการพัฒนาต่อ

งานวิจัยนี้นำเสนอแนวความคิดเบื้องต้นของการนับปริมาณรถอย่างง่าย ผลที่ได้จากการนับปริมาณรถที่ระบุเป็นจำนวนคันที่กำหนดเป็นขนาดของรถยนต์ส่วนบุคคลเท่านั้น ที่มาจากค่าเฉลี่ยของขนาดของรถยนต์ 1 คัน ที่สุ่มเลือกเป็นจำนวน 3 คัน ไม่มีการแยกขนาดของรถชนิดอื่นๆ ดังนั้น แนวทางการพัฒนาต่อไปควรมีการแยกขนาดของรถชนิดอื่นๆที่มีขนาดแตกต่างกันเพิ่มเติม เช่น รถประจำทาง รถจักรยานยนต์ เป็นต้น และหาค่าเฉลี่ยของขนาดของรถ 1 คัน จากขนาดของรถที่สุ่มเลือกในจำนวนที่มากขึ้น เพื่อความถูกต้อง แม่นยำ และมีประสิทธิภาพของระบบการนับปริมาณรถที่จะพัฒนาต่อไป

8. กิตติกรรมประกาศ

ขอขอบคุณเทศบาลนครหาดใหญ่ในความเอื้อเฟื้อภาพวิดีโอสำหรับการทดสอบการออกแบบระบบนับรถบนเอฟพีจีเอ

9. เอกสารอ้างอิง

[1] ธีรยศ เวียงทอง, "รู้จักกับการออกแบบร่วมกันระหว่างฮาร์ดแวร์-ซอฟตแวร์เบื้องต้น", ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร, 2547.

[2] Thou-Ho Chen, Yu-Feng Lin, and Tsong-Yi Chen, "Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance", Proceedings of the Second International Conference on Innovative Computing, Information and Control, Kaohsiung, Taiwan, 5-7 Sept. 2007, pp. 238 – 238.

[3] Erhan Ba, A. Murat Tekalp, and F. Sibel Salman, "Automatic Vehicle Counting from Video for Traffic Flow Analysis", Proceedings of the 2007 IEEE Intelligent Vehicles Symposium Istanbul, Turkey, June 13-15, 2007.

[4] Tae-Seung Lee, Eung-Min Lee, Hyeong-Taek Park, Young-Kil Kwag, Sang-Seok Lim, Joong-Hwan Baek, and Byong-Won Hwang, "Implementation of Traffic Flow Measuring Algorithm Using Real-Time Dynamic Image Processing", Springer-Verlag Berlin Heidelberg, Korea, 2003, pp. 78–87.

[5] Manoel E.de Lima, Pablo Viana da Silva, Alejandra C Frery, Edna Barros, "A Codesign Approach for a High Performance Vehicle Detector", High Perfomance Computing HPC2003-ASTC2003, San Diego, USA, January 2003.

[6] David Pellerin and Scott Thibault , "Practical FPGA Programming in C", Prentice Hall, 2005.

[7] http://www.impulsec.com

[8] http://www.xilinx.com

ประวัติผู้เขียน

ชื่อ สกุล นางสาวชลธิศา เวทโอสถ

รหัสประจำตัวนักศึกษา 5010120132

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2550

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

ทุนค่าเล่าเรียน คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ประจำปีการศึกษา 2550

การตีพิมพ์เผยแพร่ผลงาน

ชลธิศา เวทโอสถ, ณัฏฐา จินดาเพ็ชร์, และ นิคม สุวรรณวร, (2552), การออกแบบการประมวลผลภาพวิดีโอด้วยภาษาระดับสูงสำหรับระบบนับปริมาณรถแบบฝังตัว,งานประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 32 (EECON-32), ปราจีนบุรี, pp.1091-1094 , 28-30 ตุลาคม 2552.