อัลกอริทึมเกาซ์-เซเดลที่มีประสิทธิภาพบนหน่วยประมวลผลกราฟิกส์และแอปพลิเคชันที่เกี่ยวข้อง

โดย

นายบรรพต ดลวิทยากุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

ภาควิชาคอมพิวเตอร์

บัณฑิตวิทยาลัย มหาวิทยาลัยศิลปากร

ปีการศึกษา 2556

อัลกอริทึมเกาซ์-เซเดลที่มีประสิทธิภาพบนหน่วยประมวลผลกราฟิกส์และแอปพลิเคชันที่เกี่ยวข้อง

โดย

นายบรรพต ดลวิทยากุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

ภาควิชาคอมพิวเตอร์

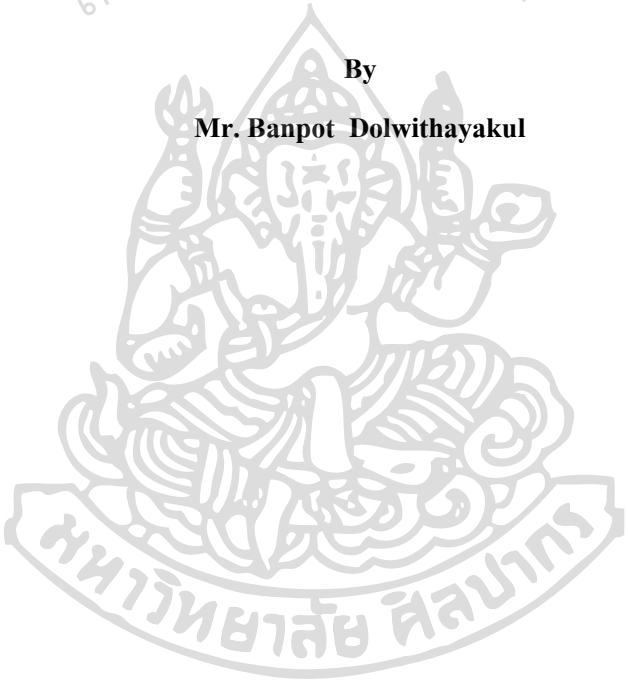
บัณฑิตวิทยาลัย มหาวิทยาลัยศิลปากร

ปีการศึกษา 2556

AN EFFICIENT GAUSS-SEIDEL ALGORITHM ON GRAPHIC PROCESSING UNITS AND APPLICATIONS

By

Mr. Banpot Dolwithayakul



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

Doctor of Philosophy Program in Computer and Information Science

Department of Computing

Graduate School, Silpakorn University

Academic Year 2013

52307803 : สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
คำสำคัญ : เกาซ์-เซเดล / การประมวลผลแบบขนาน / การประมวลผลประสิทธิภาพสูง / หน่วยประมวลผลกราฟิกส์

บรรพต ดลวิทยากุล : อัลกอริทึมเกาซ์-เซเดลที่มีประสิทธิภาพบนหน่วยประมวลผลกราฟิกส์และแอปพลิเคชันที่เกี่ยวข้อง. อาจารย์ที่ปรึกษาวิทยานิพนธ์ : รศ.ดร.จันทนา จันทราพรชัย และ อ.ดร.นพดล ชุมชอบ. 146 หน้า.

เกาซ์-เซเดล เป็นวิธีการแก้ระบบสมการเชิงเส้นวิธีหนึ่งที่ใช้งานอย่างแพร่หลายในสาขาต่างๆ เช่น คณิตศาสตร์ ฟิสิกส์ และการประมวลผลภาพของวิทยาการคอมพิวเตอร์ สำหรับงานประยุกต์เหล่านี้ มักจะต้องใช้ชุดของสมการจำนวนมาก ซึ่งจะใช้เวลานานในการแก้สมการ ดังนั้นในปัญหาขนาดใหญ่ การประมวลผลแบบขนานบนหลายหน่วยประมวลผลเป็นวิธีหนึ่งที่นำมาใช้เพื่อลดเวลาในการทำงาน หน่วยประมวลผลกราฟิกส์นั้นเป็นหน่วยประมวลผลในลักษณะนี้ที่มีราคาไม่แพงนัก และมักถูกนำมาใช้ในการประมวลผลกราฟิกส์เพียงอย่างเดียว ในงานวิจัยนี้จึงมีแนวคิดจะนำหน่วยประมวลกราฟิกส์มาใช้ช่วยคำนวณปัญหาขนาดใหญ่เหล่านี้ด้วย

ในวิทยานิพนธ์เล่มนี้ผู้วิจัยได้ออกแบบและพัฒนาเกาซ์-เซเดลเชิงขนานที่เหมาะสำหรับทำงานบนหน่วยประมวลผลกราฟิกส์สำหรับสมการผลต่างอันตะ โดยผู้วิจัยมุ่งเน้นการลดเวลาผสานการทำงานระหว่างเธรดและออกแบบอัลกอริทึมในหลายรูปแบบให้เธรดต่างๆ เข้าทำงานคำนวณได้รวดเร็วขึ้น ซึ่งผู้วิจัยได้ทดลองวิธีการต่างๆ ที่พัฒนาขึ้นได้กับสมการปัวซง พบว่าสามารถเร่งความเร็วได้ถึง 26.88 เท่าเมื่อเทียบกับการใช้หน่วยประมวลผลหลักเพียงแกนเดียว

นอกจากนี้ผู้วิจัยได้นำเกาซ์-เซเดลเชิงขนานที่พัฒนาได้ไปประยุกต์ใช้กับงานอื่นๆ ในหลายๆ แขนงรวมทั้งออกแบบขั้นตอนการประมวลผลให้เหมาะสมและมีประสิทธิภาพ เช่น การตัดสัญญาณรบกวนบนภาพนิ่งและภาพเคลื่อนไหว การตัดสัญญาณรบกวนบนภาพที่ได้จากคลื่นเสียงความถี่สูง (อัลตราซาวน์) การตัดสัญญาณรบกวนบนภาพที่ได้จากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด ซึ่งปัญหาแต่ละประเภทมีลักษณะของปัญหาต่างกันออกไป วิธีการเกาซ์-เซเดลแบบขนานและขั้นตอนการทำงานที่ผู้วิจัยพัฒนาได้สามารถทำให้ใช้งานหน่วยประมวลผลประเภทต่างๆ และทรัพยากรที่มีอยู่ได้อย่างมีประสิทธิภาพและเหมาะสมกับปัญหาแต่ละประเภท

ภาควิชาคอมพิวเตอร์ บัณฑิตวิทยาลัย มหาวิทยาลัยศิลปากร

ลายมือชื่อนักศึกษา..... ปีการศึกษา 2556

ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์ 1. ..... 2. .....

52307803 : MAJOR : COMPUTER AND INFORMATION SCIENCE
KEY WORD : GAUSS-SEIDEL, GRAPHIC PROCESSING UNIT, PARALLEL PROCESSING, HIGH PERFORMANCE COMPUTING
BANPOT DOLWITHAYAKUL : AN EFFICIENT GAUSS-SEIDEL ALGORITHM ON GRAPHIC PROCESSING UNITS AND APPLICATIONS. THESIS ADVISORS : ASSOC.PROF. CHANTANA CHANTRAPORNCHAI, Ph.D. AND NOPPADOL CHUMCHOB, Ph.D. 146 pp.

Gauss-Seidel is an iterative solver for the system of equations. The Gauss-Seidel is widely used in many scientific fields e.g. mathematics, physics, and image processing in computer science. These applications usually have very large number of equations which requires computational and memory cost. The parallel processing is a technique for reducing overall computational time. The graphic processing units (GPU) are the devices that can be used for the parallel computation. With affordable price and widely used in generic computing, this research applies the GPUs to accelerate the computation.

In this dissertation, the author designs and develops the parallel Gauss-Seidel algorithm for GPUs for solving PDE equations. The proposed algorithms aim to reduce the synchronization cost within the iteration and the waiting time for each thread. The author tested the proposed algorithm with Poisson's equation and found that the proposed algorithms can achieve the speedup up to 26.88 times faster than the sequential computation on the central processing unit (CPU).

Additionally, the author also applies the proposed Gauss-Seidel to other applications: the image and image sequence denoising, ultrasound image denoising, microorganism image from scanning electron microscope. Each application has its specific characteristics and requirements. The proposed methods and strategies implement on the different type of processors, utilize the computation resources efficiently, and are suitable with each application.

Department of Computing Graduate School, Silpakorn University
Student's signature ..... Academic Year 2013
Thesis Advisors' signature 1. ..... 2. .....

กิตติกรรมประกาศ

งานวิจัยที่ผู้วิจัยได้ทำและวิทยานิพนธ์เล่มนี้สำเร็จลุล่วงไปได้ด้วยดี โดยการดูแลและให้คำปรึกษาเป็นอย่างดีจาก รศ. ดร. จันทนา จันทราพรชัยที่ปรึกษาวิทยานิพนธ์หลัก และ ดร. นพดล ชุมชอบที่ปรึกษาร่วมจากภาควิชาคณิตศาสตร์ มหาวิทยาลัยศิลปากร อาจารย์ทั้งสองท่านได้ให้คำเสนอแนะ แนวคิด รวมทั้งชี้ให้เห็นถึงข้อบกพร่องต่างๆ จนกระทั่งสามารถปรับปรุงงานวิจัยให้อยู่ในระดับที่ดีจนสามารถตีพิมพ์เผยแพร่ได้ในวารสารวิชาการ

ทางผู้วิจัยขอขอบคุณโครงการปริญญาเอกกาญนาภิเศกโดยสำนักงานกองทุนสนับสนุนการวิจัย ที่สนับสนุนเงินทุนในการทำวิจัยทั้งในและต่างประเทศ รวมทั้งค่าเล่าเรียนและค่าใช้จ่ายเบ็ดเตล็ดต่างๆ ที่ต้องใช้ในการศึกษาระดับปริญญาเอก

นอกจากนี้ผู้วิจัยขอขอบคุณ รศ. ดร. โชคชัย เลี้ยงสุขสันต์จากมหาวิทยาลัย Louisiana Tech รัฐหลุยเซียน่า ประเทศสหรัฐอเมริกา ที่ให้ใช้เครื่อง Little FE ซึ่งเป็นคลัสเตอร์ขนาด 6 โหนด และคลัสเตอร์ AZUL ขนาด 32 โหนดในการทำวิจัย รวมทั้งให้การดูแลและให้คำแนะนำเป็นอย่างดีตลอดช่วงที่ทำวิจัยในต่างประเทศ

ทางผู้วิจัยขอขอบคุณบิดามารดา และคนในครอบครัวที่สนับสนุนการเรียน และการทำวิจัยในทุกๆ ด้านตลอดการศึกษา รวมทั้งเพื่อนๆ ทุกคนที่ให้กำลังใจเสมอมา

เหนือทุกสิ่งนั้น ผู้วิจัยขอขอบคุณพระเจ้าหนึ่งเดียวผู้สร้างทุกสรรพสิ่ง ผู้ทำสิ่งที่เป็นไปไม่ได้ให้เป็นไปได้

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

ภาพที่ หน้า

ภาพที่ หน้า

คำอธิบายสัญลักษณ์และคำย่อ

4SGS-L	Four-Stated Gauss-Seidel (List-Based)
4SGS-Q	Four-Stated Gauss-Seidel (Queue-Based)
A	เมทริกซ์สัมประสิทธิ์ของระบบสมการเชิงเส้น
AWGN	Additive White Gaussian Noise
BFD	Between-frame Delay
FPS	Frame-per-second
GS	Gauss-Seidel
I/O	Input/Output
IKONOS	ชื่อดาวเทียมที่บริหารโดยบริษัท Space Imaging
l_2	Euclidean Norm
Many-core	หน่วยประมวลผลที่มีจำนวนแกนประมวลผลมากกว่า 8 แกนขึ้นไป
MIMD	Multiple-Instruction, Multiple-Data
Multi-core	หน่วยประมวลผลที่มีจำนวนแกนประมวลผลตั้งแต่ 2 แกนถึง 8 แกน
OpenCV	Open Source Computer Vision (ไลบรารี่)
OpenCL	Open Computing Language (ไลบรารี่)
OpenMP	Open Multi-Processing (ไลบรารี่)
QL-SWGS	Queueless Sliding Window Gauss-Seidel
RGB	Red-Green-Blue

SAR	Synthetic Aperture Radar
SEM	Scanning Electron Microscope
SIMD	Single-Instruction, Multiple-Data
SN-SOR	Single Node Successive Over-Relaxation
SOR	Successive Over-Relaxation
Spatial Noise	ในวิทยานิพนธ์เล่มนี้หมายถึงสัญญาณรบกวนที่ไม่ขึ้นหรือมีผลต่อกับเฟรมอื่น
Speedup	ความเร็วที่เพิ่มขึ้นในการประมวลผลแบบขนาน โดยคิดจากเวลาประมวลผลแบบตามลำดับเทียบกับเวลาในการประมวลผลแบบขนาน
SWGS	Sliding Window Gauss-Seidel
Temporal Noise	ในวิทยานิพนธ์เล่มนี้หมายถึงสัญญาณรบกวนที่ขึ้นต่อหรือมีผลกับเฟรมอื่นๆ บนภาพเคลื่อนไหว
TSWGS	Two-Way Sliding Window Gauss-Seidel
TV	การแปรผันรวม (Total Variation)
$\boldsymbol{u}$	ภาพหรือชุดของตัวแปรไม่ทราบค่าที่ต้องการหา
$\boldsymbol{z}$	ภาพที่มีสัญญาณรบกวน
η	สัญญาณรบกวนที่มีการแจกแจงแบบปกติ (แบบเกาส์)
ξ	สัญญาณรบกวนที่มีการแจกแจงแบบแกมม่า
υ	ดัชนีรอบการทำซ้ำ

บทที่ 1

บทนำ

ที่มาและความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยีฮาร์ดแวร์ในส่วนหน่วยประมวลผลนั้นได้พัฒนาไปอย่างมาก นอกจากหน่วยประมวลผลกลาง (Central Processing Unit: CPU) แล้ว หน่วยประมวลผลอีกประเภทที่ได้รับความนิยมเพิ่มขึ้นอย่างมากในวงการวิทยาศาสตร์คือหน่วยประมวลผลกราฟิกส์ (Graphic Processing Units: GPU) ซึ่งในปัจจุบันนั้นมีการนำหน่วยประมวลผลกราฟิกส์มาใช้อย่างแพร่หลาย ทั้งในด้านกราฟิกส์ และการคำนวณทั่วไป หรือที่เรียกว่าหน่วยประมวลผลกราฟิกส์หลากประโยชน์ (General Propose Graphic Processing Units: GPGPU) ซึ่งเพิ่มประสิทธิภาพในการประมวลผลแบบขนานที่สูงกว่าการใช้หน่วยประมวลผลกลางเพียงอย่างเดียว

แต่อย่างไรก็ดี ลักษณะโครงสร้างของหน่วยประมวลผลกราฟิกส์นั้นแตกต่างจากหน่วยประมวลผลกลางเป็นอย่างมาก คือมีหน่วยประมวลผลย่อยที่เรียกว่าสตรีมโปรเซสเซอร์ (Stream Processor) อยู่ภายในเป็นจำนวนมาก รวมทั้งต้องมีการสั่งงานและส่งผ่านข้อมูลจากหน่วยประมวลผลกลาง การที่จะนำหน่วยประมวลผลกราฟิกส์มาใช้ในการแก้ปัญหาแต่ละปัญหาในวงการวิทยาศาสตร์ได้อย่างมีประสิทธิภาพนั้น จำเป็นต้องใช้อัลกอริทึมที่เหมาะสม ซึ่งออกแบบเฉพาะสำหรับการใช้งานกับหน่วยประมวลผลกราฟิกส์

หนึ่งในปัญหาที่สำคัญในวงการวิทยาศาสตร์ในสาขาวิธีการเชิงตัวเลขนั้นก็คือการแก้ระบบสมการเชิงเส้นที่มีตัวแปรไม่ทราบค่าจำนวนมาก ปัจจุบันได้มีวิธีในการแก้ปัญหานี้มากมาย เช่น วิธีการแบบหยุดนิ่ง (Stationary Method) เช่น ยาโคบิ (Jacobi) เกาซ์-เซเดล (Gauss-Seidel) และการผ่อนปรนสืบเนื่อง (Successive Over-Relaxation) [1] เป็นต้น ส่วนวิธีการแบบไม่หยุดนิ่ง (Non

Stationary Method) เช่นวิธีการเกรเดียนสังยุค (Conjugate Gradient: CG) และเกรเดียนสังยุคแบบปรับสภาวะ (Preconditioned Conjugate Gradient: PCG) วิธีการเหล่านี้หากนำมาใช้บนหน่วยประมวลผลกราฟิกส์นั้นจะต้องใช้อัลกอริทึมที่เหมาะสม ซึ่งในงานวิจัยฉบับนี้ผู้วิจัยได้มุ่งเน้นในการพัฒนาอัลกอริทึมแบบขนานสำหรับเกาซ์-เซเดลและการผ่อนปรนสืบเนื่องโดยพัฒนาต่อยอดจากเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง (Sliding Window Gauss-Seidel: SWGS) [2] ซึ่งนำเสนอโดย H. Moghnieg ในปี ค.ศ. 2010 โดยทำการพัฒนาให้รองรับการทำงานแบบหลายเธรด เพื่อให้สามารถแก้ระบบสมการขนาดใหญ่ได้เร็วและรองรับสถาปัตยกรรมสมัยใหม่แบบหลายแกน (Multi core) หรือ การประมวลผลโดยใช้หน่วยประมวลผลกราฟิกส์แบบมากแกน (Many core) ในการออกแบบอัลกอริทึมนั้น จำเป็นต้องคำนึงถึงปัจจัยสำคัญได้แก่ การกระจายงานที่เหมาะสมกับสถาปัตยกรรม การจัดการหน่วยความจำ การลดเวลาผสานการทำงาน เป็นต้น

นอกจากการพัฒนาอัลกอริทึมที่เหมาะสมแล้ว ผู้วิจัยได้นำอัลกอริทึมไปประยุกต์ใช้กับงานทางวิทยาศาสตร์ด้านต่างๆ ซึ่งในที่นี้ เช่น ด้านคณิตศาสตร์และในด้านการประมวลผลภาพ เป็นต้น

ในด้านคณิตศาสตร์ สมการปัวซงเป็นสมการประเภทหนึ่งที่มีงานประยุกต์มาก เช่น ด้านการนำความร้อน และปรากฏการณ์ทางไฟฟ้า สมการรูปแบบนี้สามารถหาคำตอบได้โดยใช้วิธีการเชิงตัวเลข เช่น วิธีการผลต่างอันตะ (Finite Difference Method) และวิธีการไฟไนต์เอลิเมนต์ (Finite Element Method) ผลที่ได้จากวิธีการเชิงตัวเลขเหล่านี้คือระบบสมการเชิงเส้นขนาดใหญ่ [1, 3] ซึ่งต้องใช้วิธีการเชิงตัวเลขแบบเร็วและมีประสิทธิภาพที่จะถูกพัฒนาในวิทยานิพนธ์ฉบับนี้

ในด้านการประมวลผลภาพ การบันทึกภาพและการนำส่งสัญญาณภาพที่ได้มาจากแหล่งต่างๆ ไม่ว่าจะเป็นภาพนิ่งที่ได้จากการบันทึกด้วยกล้องดิจิตัล การบันทึกภาพเคลื่อนไหวจากกล้องบันทึกภาพ รวมทั้งการส่งภาพในแบบอนาล็อก ล้วนแล้วแต่มีสัญญาณรบกวนทั้งสิ้น ในปัจจุบันนี้ได้มีวิธีการตัดสัญญาณรบกวนออกจากภาพหลายวิธี แต่วิธีหนึ่งที่เป็นที่นิยมและได้ผลดีคือวิธีการแปรผันซึ่งใช้การแปรผันรวม (Total Variation: TV) โดย Rudin, Osher และ Fatami [4] โดยวิธีนี้มีข้อดีในการรักษาเส้นขอบของวัตถุบนภาพ และสามารถตัดสัญญาณรบกวนที่มีปริมาณสูงมากๆ ได้ [5] ซึ่งในการแก้สมการออยเลอร์-ลากรานจ์ (Euler-Lagrange) นี้สามารถทำได้หลายวิธี แต่วิธีการหนึ่งที่เหมาะสมและให้ผลลัพธ์ดีคือการใช้วิธีการทำซ้ำจุดตรึง (Fixed-Point Iterative Method) โดย

Vogel และ Oman [6] ซึ่งเป็นการเปลี่ยนปัญหาไปในรูปแบบของระบบสมการเชิงเส้นและสามารถใช้ตัวแก้ระบบสมการเชิงเส้นที่มีอยู่เข้ามาแก้ปัญหาได้ [6] ในภาพขนาดใหญ่จะทำให้ใช้เวลาในการแก้ระบบสมการมาก ดังนั้นวิธีการของผู้วิจัยจึงเป็นส่วนหนึ่งของการอัลกอริทึมในการตัดสัญญาณรบกวนแบบการแปรผันรวมเช่นกัน

ในการทดลองขั้นต้นของผู้วิจัยพบว่าขั้นตอนที่ใช้เวลานานคือขั้นตอนการคำนวณด้วยวิธีการเกาซ์-เซเดล ซึ่งในแต่ละรอบของการทำงาน จะมีการเรียกการทำงานของเกาซ์-เซเดลจะใช้เวลาถึง 34.21 – 83.87% ของเวลาประมวลผลทั้งหมด ดังนั้นการนำหน่วยประมวลผลกราฟิกส์มาใช้งานกับส่วนนี้จะลดเวลาในการประมวลผลได้มากเช่นกัน

นอกจากการตัดสัญญาณรบกวนบนภาพนิ่งแล้ว การตัดสัญญาณรบกวนบนภาพเคลื่อนไหว (วีดีโอ) นั้นก็เช่นกัน เนื่องจากภาพวิดีโอประกอบด้วยภาพนิ่งหลายๆ เฟรมที่ต่อเนื่องกัน ดังนั้นการตัดสัญญาณรบกวนของภาพเคลื่อนไหวที่มีจำนวนเฟรมมาก ควรใช้อัลกอริทึมที่นำเสนอนี้เพื่อการประมวลผลที่รวดเร็ว รวมทั้งยังต้องปรับปรุงการกระจายงานในระดับของแต่ละเฟรมอีกด้วย ตัวอย่างเหล่านี้จะเป็นตัวอย่างการประยุกต์ใช้โดยตรงจากวิทยานิพนธ์ในเล่มนี้

วัตถุประสงค์ของงานวิจัย

วัตถุประสงค์ของงานวิจัยมีทั้งหมด 3 ประการ ดังนี้

1. เพื่อพัฒนาอัลกอริทึมที่เหมาะสมและมีประสิทธิภาพสูงในการแก้ระบบสมการเชิงเส้นด้วยวิธีการเกาซ์-เซเดล และวิธีการผ่อนปรนเกินสืบเนื่องที่ทำงานได้ดีบนหน่วยประมวลผลกราฟิกส์แบบหลายแกนและแบบมากแกน

2. เพื่อศึกษาและวิเคราะห์ความซับซ้อนด้านเวลาและด้านการการใช้หน่วยความจำในการนำอัลกอริทึมนี้เข้าแก้ปัญหาขนาดใหญ่ซึ่งเกิดจากสมการป๋วซง รวมทั้งเวลาที่ใช้ในการขนส่งข้อมูลระหว่างหน่วยประมวลผลกลางและหน่วยประมวลผลกราฟิกส์ในงานประยุกต์

3. เพื่อปรับปรุงอัลกอริทึมในข้อที่มาและความสำคัญของปัญหาไปใช้ในการนำการตัดสัญญาณรบกวนบนภาพแบบการแปรผันรวมไปใช้กับภาพเคลื่อนไหวแบบเรียลไทม์

สมมุติฐานของงานวิจัย

ในการทำงานของวิธีการเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง [2] นั้น คณะผู้วิจัยของอัลกอริทึมนี้ได้ออกแบบสำหรับหน่วยประมวลผลที่มีจำนวนไม่มาก (ประมาณ 3-4 แกน) และมีการทำงานในลักษณะของโพรดิวเซอร์-คอนซูเมอร์ (Producer-Consumer) ซึ่งใช้เวลาในการผสานการทำงาน (Synchronization) ระหว่างเธรดมาก ดังนั้นหากใช้อัลกอริทึมนี้กับหน่วยประมวลผลกราฟิกส์ที่มีจำนวนแกนมากจะทำให้ประมวลผลได้ไม่เร็วเท่าที่ควรสำหรับจำนวนแกนที่เพิ่มมากขึ้น และการจะทำให้ใช้เวลาผสานการทำงานมากเนื่องจากจำนวนเธรดมาก

ผู้วิจัยคาดว่าการออกแบบอัลกอริทึมใหม่ที่ลดการผสานการทำงานและใช้เธรดจำนวนมากทำงานจะสามารถเพิ่มประสิทธิภาพด้านเวลาได้ และเป็นการลดเวลาในการประมวลผลของงานประยุกต์ที่ใช้อัลกอริทึมนี้

ขอบเขตของงานวิจัย

1. พัฒนาอัลกอริทึมเกาซ์-เซเดลเชิงขนานที่มีประสิทธิภาพสูงขึ้นสำหรับใช้งานบนหน่วยประมวลผลกราฟิกส์ที่มีหลายแกน โดยมีการกระจายการทำงานกับเธรดหลายเธรดอย่างเหมาะสมและลดเวลาในการผสานการทำงานของเธรด

2. นำอัลกอริทึมที่พัฒนาได้ไปใช้กับงานประยุกต์ต่างๆ เช่นการหาคำตอบของสมการปัวซง การตัดสัญญาณรบกวนของภาพ การตัดสัญญาณรบกวนบนวีดีโอ การตัดสัญญาณรบกวนของภาพถ่ายดาวเทียมขนาดใหญ่ การตัดสัญญาณรบกวนของภาพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด การตัดสัญญาณรบกวนของภาพอัลตราซาวน์

3. สรุปและประเมินผลด้านประสิทธิภาพเชิงเวลา (รวมทั้งการใช้หน่วยความจำ) ของอัลกอริทึมตัวใหม่ และผลที่นำอัลกอริทึมใหม่ไปใช้กับงานประเภทดังกล่าว

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้อัลกอริทึมที่ทำงานบนหน่วยประมวลผลกราฟิกส์ที่มีประสิทธิภาพในการตัดสัญญาณรบกวนของภาพประเภทต่างๆ ในแบบการแปรผันรวม

2. สามารถประยุกต์อัลกอริทึมที่พัฒนาได้ในการแก้ระบบสมการเชิงเส้นที่มีตัวแปรไม่ทราบค่าจำนวนมากเช่นการหาคำตอบเชิงตัวเลขของสมการปัวซง โดยใช้วิธีการเกาซ์-เซเดลและวิธีการผ่อนปรนเกินสืบเนื่อง

3. ได้งานประยุกต์ที่ทำงานเร็วในการตัดสัญญาณรบกวนของภาพนิ่งและภาพเคลื่อนไหวประเภทต่างๆ

การประเมินผล

ในงานวิจัยนี้จะใช้การประเมินผล 3 ด้านคือ

1. ด้านประสิทธิภาพในการทำงานของอัลกอริทึมที่พัฒนาขึ้น จะวัดประสิทธิภาพโดยการแสดง Work Load ของแต่ละเธรด จำนวนเธรดที่รอคอยเข้าทำงานในแต่ละช่วงเวลาของการประมวลผล หน่วยความจำที่ใช้และจำนวนข้อมูลที่ส่งระหว่างหน่วยความจำหลักไปยังหน่วยประมวลผลกราฟิกส์

2. ในด้านการประยุกต์กับงานต่างๆ จะทำการวัดผลด้านเวลา ซึ่งวัดเป็นอัตราความเร็วที่เพิ่มขึ้น (Speedup) เทียบระหว่างการทำงานบนหน่วยประมวลผลกลางแบบแกนเดียวและการทำงานบนหน่วยประมวลผลกราฟิกส์

3. นอกจากนี้ในการประยุกต์ใช้ จะแสดงประสิทธิภาพด้านเวลาที่ได้จากงานประยุกต์นั้นเมื่อนำอัลกอริทึมใหม่ไปใช้ และประสิทธิผลด้านการทำงานของงานประยุกต์เมื่อใช้อัลกอริทึมใหม่ ในด้านการประมวลผลภาพ จะมีการประเมินผลด้านคุณภาพของภาพที่ถูกตัดสัญญาณรบกวนโดยทำการวัดเป็นค่า Peak signal-to-noise ratio (PSNR) ซึ่งมีสมการของการคำนวณดังสมการที่ (1.1) ดังนี้

$$PSNR = 10 \cdot \log_{10}\left(\frac{\max_I^2}{MSE}\right) \quad (1.1)$$

โดยที่ค่า $\max_I$ นั้นเป็นค่าสูงสุดของความเข้มของภาพต้นฉบับ และค่า MSE เป็นค่าความคาดเคลื่อนกำลังสองเฉลี่ย (Mean Square Error) ของภาพต้นฉบับและภาพที่ต้องการเปรียบเทียบที่มีขนาด $m \times n$ โดยสามารถเขียนได้ดังสมการที่ 1.2

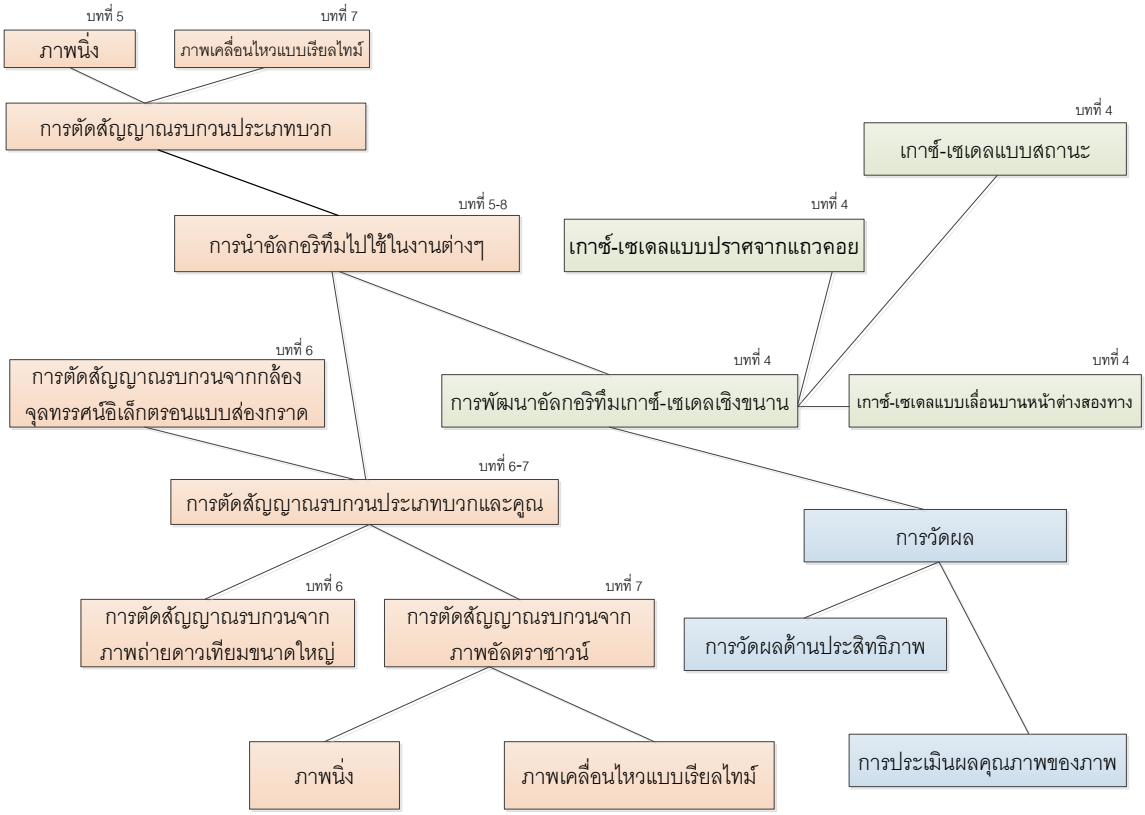
$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (1.2)$$

โดยที่ I เป็นภาพต้นฉบับ และ K เป็นภาพที่ถูกตัดสัญญาณรบกวนแล้วตามลำดับ

ในด้านประสิทธิภาพของการตัดสัญญาณรบกวนบนวีดีโอ ผู้วิจัยได้ใช้ค่าจำนวนเฟรมต่อวินาทีเป็นตัววัด โดยผู้วิจัยกำหนดค่าที่พึงพอใจควรอยู่ที่ 15 เฟรมต่อวินาทีขึ้นไปซึ่งเป็นค่าที่สายตาของมนุษย์ไม่สามารถเห็นการเปลี่ยนแปลงแบบไม่ต่อเนื่องของแต่ละเฟรมได้ [7]

ภาพแผนที่ความคิดในการดำเนินการวิจัย

ภาพแผนที่ความคิดในการดำเนินการวิจัยอัลกอริทึมเกาซ์-เซเดลเชิงขนานสามารถเขียนได้ดังภาพที่ 1.1 ซึ่งส่วนการพัฒนาอัลกอริทึมได้แก่การพัฒนาเกาซ์-เซเดลเชิงขนาน เกาซ์-เซเดลแบบปราศจากแถวคอย เกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทาง และเกาซ์-เซเดลแบบสถานะ



ภาพที่ 1.1 แผนที่ความคิดในการดำเนินงาน

ในบทที่ 2 ผู้วิจัยจะนำเสนอ ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง ซึ่งจะกล่าวถึงทฤษฎีในด้านเกาซ์-เซเดล การประมวลผลบนหน่วยประมวลผลกราฟิกส์ งานวิจัยในด้านเกาซ์-เซเดลและการประยุกต์ใช้ การตัดสัญญาณรบกวนบนภาพแบบการแปรผันรวม

ในบทที่ 3 ผู้วิจัยนำเสนอวิธีและขั้นตอนการวิจัยซึ่งได้แก่การวิเคราะห์ปัญหาที่มีในเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง แนวคิดการพัฒนาเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง รวมทั้งลักษณะของปัญหาอื่นๆ ที่สามารถนำเกาซ์-เซเดลมาประยุกต์ใช้ได้เช่นการตัดสัญญาณรบกวนบนภาพนิ่งและภาพเคลื่อนไหว

ในบทที่ 4 เป็นส่วนที่อธิบายเกี่ยวกับการพัฒนาอัลกอริทึมเกาซ์-เซเดลแบบขนานที่ผู้วิจัยพัฒนาได้รวมทั้งการวัดประสิทธิภาพและในส่วนบทที่ 5-8 จะนำเสนอเกี่ยวกับอัลกอริทึมที่ได้ปรับปรุงไปใช้ในงานประยุกต์ต่างๆ ดังนี้

บทที่ 5 นำเสนองานประยุกต์ด้านการตัดสัญญาณรบกวนแบบบวกในลักษณะการแปรผันรวม ซึ่งใช้วิธีการเกาซ์-เซเดลแบบขนานในการเร่งการทำงานของวิธีการทำซ้ำแบบจุดตรึง ซึ่งในบทนี้เน้นที่การตัดสัญญาณรบกวนของภาพถ่ายทั่วไป

บทที่ 6 นำเสนองานประยุกต์ด้านการตัดสัญญาณรบกวนจากภาพแบบบวกและแบบคูณ ซึ่งใช้โมเดลในการตัดสัญญาณรบกวนที่แตกต่างจากการตัดสัญญาณรบกวนแบบบวก โดยผู้วิจัยได้นำภาพถ่ายดาวเทียมขนาดใหญ่และภาพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดซึ่งมีลักษณะของปัญหาต่างกันมาใช้ทดลอง

บทที่ 7 นำเสนองานประยุกต์ด้านการตัดสัญญาณรบกวนบนภาพเคลื่อนไหว โดยผู้วิจัยได้ทดลองใช้หน่วยประมวลผลกราฟิกส์ในการตัดสัญญาณรบกวนแบบบวกบนภาพวีดีโอตัวอย่าง และการตัดสัญญาณแบบบวกและแบบคูณเพื่อตัดสัญญาณรบกวนบนภาพวีดีโออัลตราซาวน์

และในส่วนสุดท้ายบทที่ 8 ผู้วิจัยได้สรุป และอภิปรายผลการทดลองในส่วนต่างๆ รวมทั้งข้อเสนอแนะและงานวิจัยในอนาคต

บทที่ 2

ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

ในวิทยานิพนธ์เล่มนี้จะมุ่งเน้นไปที่การทบทวนตัวแก้ระบบสมการเชิงเส้นขนาดใหญ่ด้วยวิธีการเกาซ์-เซเดลบนหน่วยประมวลผลกราฟิกส์ และแอปพลิเคชันที่เกี่ยวข้อง สำหรับในบทที่ 2 นี้ทางผู้วิจัยได้แบ่งการศึกษาออกเป็นสองส่วนคือทฤษฎีที่เกี่ยวข้อง และ วรรณกรรมที่เกี่ยวข้อง ซึ่งจะกล่าวถึงในหัวข้อถัดไป

1. ทฤษฎีที่เกี่ยวข้อง

ในหัวข้อนี้ผู้วิจัยจะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับงานวิจัย โดยเริ่มต้นจะกล่าวถึงสถาปัตยกรรม Compute Unified Device Architecture (CUDA) ซึ่งเป็นสถาปัตยกรรมบนหน่วยประมวลผลกราฟิกส์ที่ผู้วิจัยเลือกนำมาวิจัยมาแก้ปัญหา จากนั้นผู้วิจัยจะอธิบายพื้นฐานการแก้ระบบสมการเชิงเส้นด้วยการยกตัวอย่างสมการปัวซงและการทดสอบการประมวลผลตัวแก้สมการเชิงเส้นที่พัฒนาได้บนหน่วยประมวลผลกราฟิกส์

1.1 การประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกส์

หน่วยประมวลผลกราฟิกส์ถูกนำมาใช้ในงานประมวลผลด้านภาพ เช่น การทำงานการออกแบบใช้คอมพิวเตอร์ช่วย (Computer-aided design: CAD) การทำโมเดลสองและสามมิติ รวมทั้งเกมส์คอมพิวเตอร์ ซึ่งในทศวรรษที่ผ่านมานั้นฮาร์ดแวร์ของหน่วยประมวลผลกราฟิกส์ได้มีมีการพัฒนาไปอย่างมาก และมีจำนวนหน่วยประมวลผลเป็นจำนวนมากทำให้มีผู้ริเริ่มนำหน่วยประมวลผลกราฟิกส์มาใช้กับงานทั่วไป เช่น การแก้ระบบสมการเชิงเส้น การสร้างโมเดลจำลอง เป็นต้น ซึ่งการนำหน่วยประมวลผลกราฟิกส์มาทำงานในลักษณะนี้ว่าหน่วยประมวลผลกราฟิกส์หลากประโยชน์ (GPGPU)

การประมวลผลบนหน่วยประมวลผลกราฟิกส์นั้นมีข้อดีหลายประการ เช่น สามารถเพิ่มปริมาณงานต่อหน่วยเวลา (throughput) เนื่องจากมีจำนวนแกนประมวลผลมาก โดยสามารถสร้างเธรดและทำลายเธรดในการทำงานโดยใช้ทรัพยากรที่น้อย รวมทั้งการคำนวณยังประหยัดพลังงาน ซึ่งสามารถวัดเป็นต้นทุนต่อวัตต์ (Cost-per-Watt) ได้ต่ำกว่าการประมวลผลโดยหน่วยประมวลผลประเภทหน่วยประมวลผลกลาง [8]

ในปัจจุบันนี้ได้มีสถาปัตยกรรมและเครื่องมือที่ใช้พัฒนาโปรแกรมบนหน่วยประมวลผลกราฟิกส์จากสองค่ายได้แก่ CUDA จากบริษัท NVIDIA® ซึ่งผลิตเครื่องมือซอฟต์แวร์ที่มีชื่อว่า CUDA Software Development Kit (CUDA SDK) ซึ่งปัจจุบันนี้เป็นเวอร์ชัน 5.0 และเครื่องมืออีกตัวที่มีชื่อว่า Open Computing Language (OpenCL) ซึ่งเป็นลักษณะโอเพนสแตนดาร์ดซึ่งเครื่องมือนี้ไม่ขึ้นกับฮาร์ดแวร์ของหน่วยประมวลผลว่าเป็นของค่ายใดและผู้ผลิตใด รวมทั้งสามารถประมวลผลได้บนหน่วยประมวลผลที่หลายสถาปัตย์กรรมที่ไม่ต้องเจาะจงว่าต้องเป็นหน่วยประมวลผลกราฟิกส์เท่านั้น ซึ่งรายละเอียดของ CUDA และ OpenCL นั้นจะกล่าวถึงในหัวข้อถัดไป

1.1.1 Compute Unified Device Architecture (CUDA) เป็นสถาปัตยกรรมสำหรับการประมวลผลเชิงขนานซึ่งใช้หน่วยประมวลผลกราฟิกส์ที่มีจำนวนหน่วยประมวลผลจำนวนมากจากบริษัท NVIDIA® การประมวลผลด้วยสถาปัตยกรรม CUDA นั้นมีข้อได้เปรียบต่างๆ ดังนี้ [8, 9]

1. สามารถอ่านข้อมูลได้จากหลายๆ ตำแหน่งพร้อมๆกัน (Scatter Read)
2. มีหน่วยความจำที่เรียกว่า Shared Memory ซึ่งเป็นหน่วยความจำความเร็วสูงมีขนาด 48 กิโลไบต์ต่อบล็อก และผู้พัฒนาสามารถเรียกใช้งานได้ทำให้การประมวลผลบางประเภท เช่น การค้นหาบนข้อมูลขนาดเล็กบนหน่วยความจำนี้ทำได้เร็ว
3. สามารถถ่ายโอนข้อมูลระหว่างหน่วยความจำหลักกับหน่วยความจำบนหน่วยประมวลผลกราฟิกส์ได้เร็วกว่าสถาปัตยกรรมของหน่วยประมวลผลกราฟิกส์อื่นๆ

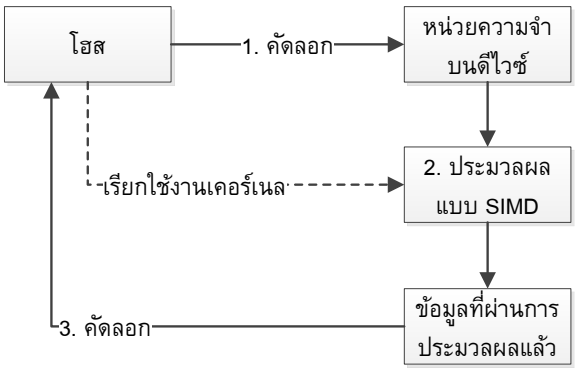
ระบบฮาร์ดแวร์ที่รองรับ CUDA ที่มีขายนั้นในปัจจุบันมีอยู่ทั้งสิ้น 5 รุ่น ในงานวิจัยนี้ได้เลือกใช้รุ่น 2.0 ที่มีชุดคำสั่งที่เรียกว่าเฟอร์มี่ (Fermi) ซึ่งเป็นชุดคำสั่งของบริษัท NVIDIA รองรับ CUDA 2.0 ที่ใช้กับหน่วยประมวลผลกราฟิกส์ตั้งแต่รุ่น GT-400 ขึ้นไป ทำให้สามารถเพิ่มความเร็ว

ในการประมวลผลข้อมูลแบบความเที่ยงสองเท่า (Double Precision) ได้อย่างมาก รวมทั้งการถ่ายโอนข้อมูลระหว่างหน่วยความจำหลักและหน่วยความจำบนหน่วยประมวลผลกราฟิกส์ได้เร็วขึ้น รายละเอียดคุณสมบัติที่สำคัญต่างๆ เปรียบเทียบกับ CUDA รุ่นก่อนหน้านั้นแสดงได้ตารางที่ 2.1

ตารางที่ 2.1 ตารางเปรียบเทียบคุณสมบัติที่สำคัญของ CUDA ในรุ่นต่างๆ

ข้อเปรียบเทียบ	**รุ่นของฮาร์ดแวร์ CUDA**	
	รุ่น 1.1 – 1.3	**รุ่น 2.0 ขึ้นไป (Fermi)**
การประมวลผลจำนวน Double Precision	มีเฉพาะรุ่น 1.3	มี และทำงานได้รวดเร็วกว่ารุ่น 1.3
จำนวนเธรดสูงสุดในแต่ละบล็อก	512	1024 ขึ้นไป
การสร้าง Texture ในงานกราฟิกส์	ไม่สามารถทำได้	สามารถทำได้
การผสานการทำงานของเธรด	ทำได้โดยใช้คำสั่ง __syncthread() และกำหนดเงื่อนไขไม่ได้	สามารถกำหนดเงื่อนไขในการผสานการทำงานได้
ขนาดของ Shared Memory	16 กิโลไบต์	48 กิโลไบต์
ขนาดของ Local Memory ในแต่ละเธรด	16 กิโลไบต์	512 กิโลไบต์

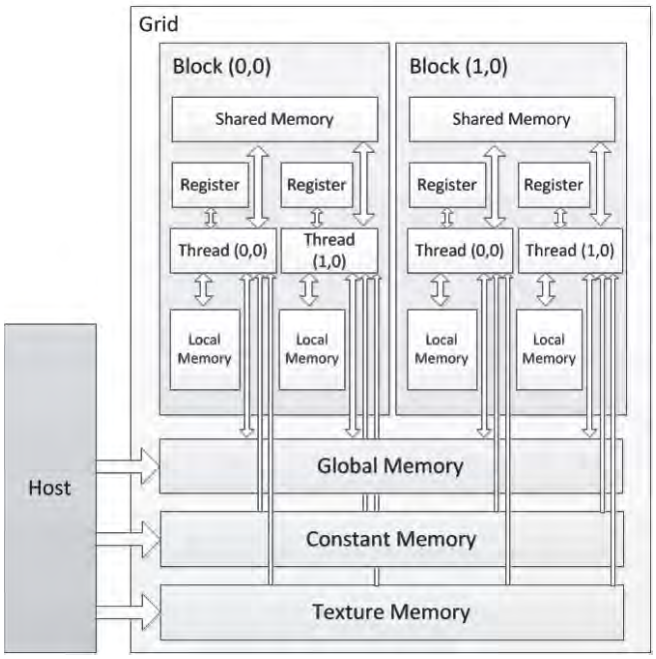
การประมวลผลบนหน่วยประมวลผลชนิดนี้ต้องการการส่งข้อมูลจากหน่วยความจำของหน่วยประมวลผลหลัก (CPU) หรือโฮส (Host) ไปยังหน่วยประมวลผลกราฟิกส์หรือดีไวซ์ (Device) จากนั้นจะทำการประมวลผลบนดีไวซ์ได้ โดยลักษณะการประมวลผลนั้นจะเป็นแบบคำสั่งชุดเดียวแต่กระทำกับข้อมูลหลายชุด (Single Instruction, Multiple Data: SIMD) เมื่อการประมวลผลบนหน่วยประมวลผลกราฟิกส์เสร็จสิ้นก็จะต้องคัดลอกข้อมูลจากดีไวซ์ไปยังโฮสเพื่อทำงานให้โฮสต่อไป ขั้นตอนต่างๆนั้นสามารถแสดงได้ตามภาพที่ 2.1



ภาพที่ 2.1 ขั้นตอนการทำงานของสถาปัตยกรรม CUDA

ภาพที่ 2.1 แสดงขั้นตอนการทำงาน ในตอนเริ่มต้นโฮสส่งข้อมูลไปยังหน่วยความจำของหน่วยประมวลผลกราฟิกส์ เมื่อส่งข้อมูลเรียบร้อยแล้วโฮสจะเรียกใช้งาน Kernel ซึ่งเรียกขั้นตอนนี้ว่า Kernel Invocation เป็นการประมวลผลในประมวลผลกราฟิกส์กับข้อมูลที่ได้จากโฮส โดยใช้คำสั่งตามที่ Kernel ระบุไว้ เมื่อประมวลผลเสร็จก็จะคัดลอกผลลัพธ์จากหน่วยความจำกลับไปยังหน่วยความจำของโฮสต์

สถาปัตยกรรม CUDA นั้นมีหน่วยความจำทั้งหมด 4 ระดับ ระดับแรกเรียกว่า Global Memory ซึ่งมีการเข้าถึงได้ช้าที่สุด ใช้เวลาในการเข้าถึงประมาณ 600 รอบสัญญาณนาฬิกา(Clock Cycle) ต่อการเข้าถึง ส่วนหน่วยความจำระดับถัดไปคือ หน่วยความจำแบบแชร์ (Shared Memory) เป็นการเก็บข้อมูลบนแคชท้องถิ่น (Local Cache) มีความเร็วในการเข้าถึงข้อมูลประมาณ 40 รอบการทำงานและมีขนาดตั้งแต่ 16-64 กิโลไบต์ รองลงมาจากหน่วยความจำแบบรีจิสเตอร์มีความเร็วสูงสุดโดยรีจิสเตอร์ 1 ตัวนั้นจะเก็บข้อมูลได้ 32 บิต และในการ์ดประมวลผลกราฟิกส์ 1 ใบจะมีจำนวนรีจิสเตอร์ขนาดรวมตั้งแต่ 8 เมกะไบต์ (ใน CUDA 1.0 – 1.3) ถึง 32 เมกะไบต์ (ใน CUDA 2.0 ขึ้นไป) ซึ่งสถาปัตยกรรม CUDA นั้นสามารถแสดงได้ตามภาพที่ 2.2



ภาพที่ 2.2 โครงสร้างของสถาปัตยกรรม CUDA [10]

ในการเขียนโปรแกรมบน CUDA นั้น ผู้พัฒนาโปรแกรมสามารถกำหนดจำนวนของเธรดที่จะใช้ทำงานสถาปัตยกรรม CUDA นั้นสามารถสร้างและทำลายเธรดได้อย่างรวดเร็ว และสามารถสร้างเธรดได้จำนวนมาก ทั้งนี้ผู้พัฒนาสามารถสามารถรวมกลุ่มของเธรดให้เป็นบล็อก (Block) และรวมหลายๆ บล็อกเป็นกริด (Grid) ให้เหมาะสมกับงานที่ทำได้ ซึ่งในแต่ละบล็อกนั้นสามารถใช้ข้อมูลในส่วนของ Shared Memory ร่วมกันได้ และหากเครื่องที่พัฒนามีการ์ดหน่วยประมวลผลกราฟิกส์มากกว่า 1 ใบ สามารถกำหนดจำนวน Grid เพิ่มเพื่อใช้หน่วยประมวลผลที่เพิ่มขึ้นมาได้

เนื่องจากสถาปัตยกรรม CUDA นั้นแตกต่างกับสถาปัตยกรรมบนหน่วยประมวลผลหลักการประมวลผลให้มีประสิทธิภาพนั้นมีปัจจัยที่ต้องคำนึงอยู่สามประการได้แก่

4. จำนวนเธรดที่มีมาก เนื่องจาก CUDA นั้นมีหน่วยประมวลผลย่อยจำนวนมาก การทำงานแบบขนานนั้นต้องสร้างเธรดจำนวนมากเพื่อให้ใช้ประโยชน์จากหน่วยประมวลผลย่อยได้มากขึ้น จึงต้องมีการแบ่งงานให้กับเธรดจำนวนมากที่เหมาะสม รวมทั้งการผสานการทำงานระหว่างเธรด (Thread Synchronization) ที่เหมาะสมด้วย

5. หน่วยประมวลผลกราฟิกส์ไม่สามารถอ่านข้อมูลจากหน่วยความจำหลักของโฮสต์ได้โดยตรง ต้องมีการถ่ายโอนข้อมูลระหว่างหน่วยความจำหลักของโฮสต์ และหน่วยความจำบนหน่วยประมวลผลกราฟิกส์ก่อนที่จะมีการประมวลผล ซึ่งการถ่ายโอนข้อมูลนั้นใช้เวลามาก จึงต้องมีการวางแผนในการถ่ายโอนข้อมูล

6. หน่วยประมวลผลกราฟิกส์เดิมนั้นออกแบบการประมวลผลเลขทศนิยม (Floating Point) แบบ ความเที่ยงหนึ่งเท่า (Single Precision) ซึ่งมีความแม่นยำต่ำ แต่เพียงพอในการประมวลผลด้านกราฟิกส์ การใช้หน่วยประมวลผลกราฟิกส์ในการประมวลผลงานเชิงวิทยาศาสตร์ที่ต้องการความแม่นยำสูงในระดับความเที่ยงสองเท่าจะต้องใช้หน่วยประมวลผลตั้งแต่รุ่น 1.3 ขึ้นไปเท่านั้นและการประมวลผลแบบความเที่ยงสองเท่าจะใช้เวลาในการประมวลผลมากกว่าในแบบความเที่ยงหนึ่งเท่ามาก

อย่างไรก็ตามสถาปัตยกรรม CUDA นั้นมีข้อเสียคือยึดติดกับฮาร์ดแวร์ของอุปกรณ์มากเกินไป คือสามารถใช้งานได้กับหน่วยประมวลผลกราฟิกส์ของบริษัท NVIDIA® ที่ใช้สถาปัตยกรรม CUDA เท่านั้น ทางบริษัท Apple® จึงได้มีแนวคิดในการพัฒนาเครื่องมือที่สำหรับพัฒนาบนสถาปัตยกรรมแบบอื่นๆ ซึ่งเป็นลักษณะของภาษากลางขึ้นมา โดยเรียกว่า Open Computing Language (OpenCL) ซึ่งจะกล่าวในหัวข้อถัดไป

1.1.2 Open Computing Language (OpenCL) เป็นสภาพแวดล้อมสำหรับการพัฒนาโปรแกรมที่มีลักษณะการทำงานได้หลายแพลตฟอร์มหรือหลายสถาปัตยกรรมของหน่วยประมวลผลกลางและหน่วยประมวลผลกราฟิกส์ (หรือหน่วยประมวลผลแบบอื่นๆ) ซึ่งในปัจจุบันสามารถทำงานได้กับทั้งหน่วยประมวลผลกราฟิกส์ทั้งของบริษัท Advance Micro Device (AMD) และ Intel® รวมทั้งหน่วยประมวลผลประเภทอื่นๆ เช่น ARM เป็นต้น

การพัฒนาโปรแกรมบน OpenCL นั้นจะพัฒนาบนแพลตฟอร์มของ C99 เป็นหลักโดยสามารถเลือกได้ทั้งแบบ data-parallel และ task parallel ได้ ซึ่งลักษณะขั้นตอนการทำงานนั้นจะใกล้เคียงกับ CUDA มาก คือต้องมีการถ่ายข้อมูลจากหน่วยประมวลผลกลางไปยังหน่วยประมวลผลกราฟิกส์ และมีการเรียกใช้งานเคอร์เนลหลังจากนั้นจึงคัดลอกข้อมูลที่ประมวลผลจากหน่วยความจำของหน่วยประมวลผลกราฟิกส์กลับมายังหน่วยความจำอีกครั้ง ซึ่งมีข้อแตกต่าง

เล็กน้อยด้านชื่อฟังก์ชัน ชื่อคลาสต่างๆ และมีชุดคำสั่งด้านหน่วยความจำสำหรับทำ data-parallel ซึ่งระดับของหน่วยความจำนั้นจะคล้ายกับ CUDA แต่จะมีชื่อเรียกต่างกันดังตารางที่ 2.1

ตารางที่ 2.2 เปรียบเทียบชื่อเรียกของหน่วยความจำระดับต่างๆ ของ CUDA และ OpenCL

CUDA	**OpenCL**
Shared Memory	Local Memory
Global Memory	Global Memory
Constant	Constant
ไม่มี แต่สามารถประกาศตัวแปรประเภทนี้ได้ภายในเคอร์แนล	Private

สำหรับการเปรียบเทียบระหว่าง CUDA และ OpenCL นั้นสามารถเปรียบเทียบได้ตามตารางที่ 2.3

ตารางที่ 2.3 ข้อเปรียบเทียบระหว่าง CUDA และ OpenCL ในด้านต่างๆ [11, 12]

ชื่อการเปรียบเทียบ	**CUDA**	**OpenCL**
การพัฒนาโปรแกรมบน C++	รองรับทั้งแบบการเขียนโปรแกรมแบบโครงสร้างและแบบวัตถุ (OOP)	รองรับทั้งแบบการเขียนโปรแกรมแบบโครงสร้าง(C) และแบบวัตถุ (C++)
ประสิทธิภาพ (มกราคม 2012)	เร็วกว่า OpenCL	ช้ากว่า CUDA

สำหรับงานวิจัยนี้ทางผู้วิจัยสนใจที่จะเลือกใช้สถาปัตยกรรม CUDA เวอร์ชัน 4.1 ในการพัฒนาอัลกอริทึมและงานประยุกต์ เนื่องจากงานวิจัยนี้เริ่มในปี พ.ศ. 2552 ซึ่งในขณะนั้น OpenCL ยังมีประสิทธิภาพและความเสถียรไม่ดีเท่าที่ควร แต่อย่างไรก็ตามทั้ง OpenCL และ CUDA นั้นมีความคล้ายคลึงในด้านการทำงานเป็นอย่างมาก การพัฒนาอัลกอริทึมบน CUDA ในวิทยานิพนธ์เล่มนี้สามารถนำไปประยุกต์ใช้กับ OpenCL ได้ในอนาคตด้วย

1.2 การแก้ระบบสมการเชิงเส้น

ระบบสมการเชิงเส้นคือการรวมกันของสมการเชิงเส้นซึ่งมีตัวแปรชุดเดียวกัน เช่น

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1 \\ a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_2 \\ &\vdots \\ a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_m \end{aligned}$$

เมื่อ $a_{i,j}$ และ b_i เป็นจำนวนจริงหรือจำนวนเชิงซ้อน การแก้ระบบสมการคือการหาค่าตัวแปรไม่ทราบค่า $x_1, x_2, \ldots, x_n$ที่จะทำให้ทุกสมการเชิงเส้นในระบบสมการข้างต้นเป็นจริง

วิธีในการแก้ระบบสมการเชิงเส้น สามารถแบ่งออกเป็น 2 วิธีได้แก่ วิธีการหาคำตอบแบบโดยตรง (Direct Method) และวิธีการทำซ้ำ (Iterative Method)

ในวิธีการแบบโดยตรงนั้น ข้อดีคือเมื่อหาคำตอบได้แล้ว คำตอบที่ได้นั้นจะเป็นผลลัพธ์แม่นตรง (Exact Solution) วิธีการหาคำตอบโดยตรงซึ่งเป็นที่นิยมได้แก่การกำจัดแบบเกาซ์ และการหาเมทริกซ์ผกผัน อย่างไรก็ดี การหาคำตอบโดยใช้วิธีการนี้จะใช้เวลานาน รวมทั้งใช้ทรัพยากรในการประมวลผลมาก เช่น วิธีการกำจัดแบบเกาซ์มีความซับซ้อนในการคำนวณเป็น $O(n^3)$ โดยที่ n นั้นเป็นจำนวนของตัวแปรไม่ทราบค่า

ในการหาคำตอบอีกประเภทหนึ่งเรียกว่าวิธีการทำซ้ำ แนวคิดหลักมูลของวิธีการนี้คือการแบ่งการประมวลผลออกเป็นรอบย่อยๆ การประมวลผลแต่ละครั้งจะทำให้คำตอบโดยประมาณลู่เข้าสู่คำตอบที่แท้จริงมากขึ้นเรื่อยๆ เป็นผลให้สามารถควบคุมเวลาและจำนวนรอบในการประมวลผลได้ อัลกอริทึมส่วนใหญ่ของวิธีการนี้สามารถทำงานแบบขนานได้ดีเนื่องจากสามารถแบ่งการทำงานออกเป็นกลุ่มย่อยๆ ได้ง่าย ซึ่งในปัจจุบันมีหลายวิธี เช่น วิธีการยาโคบิ วิธีการเกาซ์-เซเดล และวิธีการผ่อนปรนเกินสืบเนื่องรวมไปถึงวิธีการในแบบใหม่ เช่น วิธีการเกรเดียนสังยุค

1.3 วิธีการทำซ้ำ

วิธีการแก้ระบบสมการเชิงเส้นขนาดใหญ่มีทั้งแบบวิธีการหยุดนิ่ง (Stationary Method) เช่น วิธีการยาโคบิ วิธีการเกาซ์-เซเดล วิธีการผ่อนปรนเกินสืบเนื่องและแบบไม่หยุดนิ่ง(Non-Stationary

Method) ซึ่งเป็นวิธีแบบใหม่ เช่น วิธีการเกรเดียนสังยุคและวิธีการเกรเดียนสังยุคแบบปรับสภาวะ มีข้อดีข้อเสียตามตาราง 2.4 [3, 13]

ตารางที่ 2.4 สรุปข้อดีข้อเสียของวิธีการต่างๆ ที่ใช้แก้ระบบสมการเชิงเส้นที่นิยมในปัจจุบัน

ชื่อวิธีการ	ข้อดี	ข้อเสีย
ยาโคบิ	ง่าย ไม่ซับซ้อน สามารถคำนวณหาทุกตัวแปรที่ไม่ทราบค่าบนเมทริกซ์ได้พร้อมๆกันทำให้ทำงานแบบขนานได้ง่าย	มีการลู่เข้าช้า แม้ว่าจะมีการใช้พารามิเตอร์ผ่อนปรนเกินสืบเนื่องก็ตามรวมทั้งไม่สามารถแก้เมทริกซ์ขนาดใหญ่ที่มีภาวะแย่ได้
เกาซ์-เซเดล	ไม่ซับซ้อน มีอัตราการลู่เข้าที่ดีกว่าวิธีการยาโคบิมาก	ไม่สามารถแก้เมทริกซ์ขนาดใหญ่ที่มีภาวะแย่ได้ และมีการขึ้นต่อกันระหว่างสมาชิกบนเมทริกซ์ ทำให้ทำเชิงขนานได้ยากกว่า
การผ่อนปรนเกินสืบเนื่อง (เกาซ์-เซเดลแบบถ่วงน้ำหนัก)	มีอัตราการลู่เข้าที่ดีกว่าวิธีการยาโคบิและวิธีการเกาซ์-เซเดล โดยมีการถ่วงน้ำหนักของคำตอบ	เหมือนวิธีการเกาซ์-เซเดล และต้องมีการถ่วงน้ำหนักด้วยค่าที่เหมาะสม
เกรเดียนสังยุค	มีอัตราการลู่เข้าสูงกว่าทั้ง 3 วิธีการที่กล่าวมาข้างต้น	มีการใช้ทรัพยากรเริ่มต้นในการประมวลผลสูง ทำให้ไม่เหมาะกับงานขนาดเล็ก หรือเมทริกซ์ที่ไม่ค่อยมีความซับซ้อนและไม่ต้องการผลลัพธ์ที่แม่นยำมากนัก และใช้งานกับเมทริกซ์ได้บางประเภทเท่านั้น
เกรเดียนสังยุคแบบปรับสภาวะ	เหมือนวิธีการเกรเดียนสังยุค และสามารถแก้เมทริกซ์ที่มีภาวะแย่ ได้เนื่องจากมีการสร้างและใช้ตัวปรับสภาวะ (Preconditioner)	เหมือนวิธีการเกรเดียนสังยุคและยังต้องเสียเวลาในการสร้างตัวปรับสภาวะมากขึ้น

จากตารางที่ 2.4 เห็นได้ว่าการแก้ระบบสมการเชิงเส้นด้วยวิธีการเกาซ์-เซเดลนั้นเป็นวิธีการที่ให้ ผลลัพธ์ในการลู่เข้าสู่คำตอบ (Convergence) ที่ดี ไม่ยุ่งยากซับซ้อนและรวดเร็ว จึงมีการนำวิธีการเกาซ์-เซเดลมาใช้ในการประยุกต์ด้านต่างๆ ซึ่งวิธีการเกาซ์-เซเดล นั้นเป็นวิธีหนึ่งที่สำหรับแก้สมการเชิงเส้นที่มักเขียนอยู่ในรูปแบบ $\mathbf{Au} = \boldsymbol{b}$ โดยเมทริกซ์ $\boldsymbol{A}$ มีขนาด $n \times n$ และมีการกำหนดค่าคาดเดาเริ่มต้นที่ $k=0$ รูปแบบการทำซ้ำของการเกาซ์-เซเดลโดยทั่วไปนั้นสามารถเขียนได้ตามสมการ (2.1) โดยที่ k นั้นเป็นหมายเลขรอบหรือดัชนีของการทำซ้ำ

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \qquad (2.1)$$

ตัวอย่างการแก้ระบบสมการเชิงเส้นด้วยวิธีการเกาซ์-เซเดล พิจารณาระบบสมการ (2.2)

$$\begin{aligned} 12x_1 + 3x_2 - 5x_3 &= 1 \\ x_1 + 5x_2 + 3x_3 &= 28 \\ 3x_1 + 7x_2 + 13x_3 &= 76 \end{aligned} \qquad (2.2)$$

เวกเตอร์เมื่อทำการเขียนระบบสมการ (2.2) ให้อยู่ในรูปเมทริกซ์-เวกเตอร์จะได้สมการที่ (2.3) และ (2.4) ตามลำดับ

$$A = \begin{bmatrix} 12 & 3 & -5 \\ 1 & 5 & 3 \\ 3 & 7 & 13 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ 28 \\ 76 \end{bmatrix} \qquad (2.3),(2.4)$$

โดยที่ $\boldsymbol{x}$ เป็นเวกเตอร์ซึ่งมีส่วนประกอบเป็นตัวแปรไม่ทราบค่าที่ต้องการหา กำหนดให้ค่าคาดเดาเริ่มต้น (Initial guess) ของ $\boldsymbol{x}$ คือ

$$x^{(0)} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \qquad (2.5)$$

ทั้งนี้ปัญหาในสมการ (2.2) มีคำตอบแม่นตรงตามสมการ (2.6)

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} \tag{2.6}$$

หลังจากใช้รูปแบบการทำซ้ำของวิธีการเกาซ์-เซเดลในสมการ (2.1) เพื่อแก้ปัญหาดังกล่าว ในรอบแรกของการทำซ้ำจะได้ผลลัพธ์ตามสมการ (2.7) ถึง (2.9)

$$x_1^{(k+1)} = \frac{1 - 3x_2^{(k)} + 5x_3^{(k)}}{12} = 0.50 \tag{2.7}$$

$$x_2^{(k+1)} = \frac{28 - x_1^{(k+1)} - 3x_3^{(k)}}{5} = 4.90 \tag{2.8}$$

$$x_3^{(k+1)} = \frac{76 - 3x_1^{(k+1)} - 7x_2^{(k+1)}}{13} = 3.09 \tag{2.9}$$

เพราะฉะนั้น เวกเตอร์คำตอบโดยประมาณที่ได้ผลการทำซ้ำในการทำซ้ำรอบแรกจะได้ $\boldsymbol{x}$ คือ

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0.5000 \\ 4.9000 \\ 3.0923 \end{bmatrix} \tag{2.10}$$

ขั้นต่อไปคือการนำเวกเตอร์ที่ได้ใน (2.10) ไปใช้ในการทำซ้ำรอบถัดไป เมื่อผ่านไป 6 รอบ การทำซ้ำ จะได้เวคเตอร์คำตอบโดยประมาณในคอลัมน์เวกเตอร์ $\boldsymbol{x}$ ดังสมการ (2.11)

$$x^{(6)} = \begin{bmatrix} x_1^{(6)} \\ x_2^{(6)} \\ x_3^{(6)} \end{bmatrix} = \begin{bmatrix} 0.99919 \\ 3.0001 \\ 4.0001 \end{bmatrix} \tag{2.11}$$

เห็นได้ว่าคำตอบโดยประมาณที่ได้จากการทำซ้ำของวิธีการเกาซ์-เซเดลนั้นไม่ใช่คำตอบที่แม่นตรง แต่ใกล้เคียงกับคำตอบแม่นตรงในสมการ (2.6) โดยเมื่อจำนวนรอบของการทำซ้ำมากขึ้น คำตอบที่ได้จะใกล้เคียงกับคำตอบแม่นตรงมากขึ้นเรื่อยๆ

ในงานวิจัยฉบับนี้จะใช้วิธีการเกาซ์-เซเดลเชิงขนานเพื่อหาคำตอบระบบสมการเชิงเส้น สมการเชิงเส้นที่ใช้งานมากได้แก่สมการปัวซงเนื่องจากเป็นสมการที่อธิบายปรากฏการณ์ทาง

วิทยาศาสตร์ทั้งแบบกายภาพและชีวภาพได้อย่างสมเหตุสมผล เช่น การกระจายความร้อนแบบไม่เคลื่อนที่หรือในงานพลศาสตร์ของไหล [1] เป็นต้น

1.4 สมการปัวซงแบบสองมิติ

สมการปัวซงแบบสองมิติภายใต้เงื่อนไขขอบแบบดีรีเคล (Dirichlet) [1] สามารถเขียนได้เป็น

$$-\Delta u(x,y)=f(x,y),\ (x,y)\in\Omega \tag{2.12}$$

$$u(x,y)=0,\ (x,y)\in\partial\Omega \tag{2.13}$$

โดยที่

$$\Delta \equiv \frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2} \tag{2.14}$$

โดยที่ Ω แทนโดเมน และ $\partial\Omega$ แทนขอบ

หลังจากใช้วิธีการผลต่างอันตะแบบพื้นฐานที่ใช้การประมาณค่าแบบ 5 จุด แบ่ง Ω ออกเป็นเซลจำนวน $(N+2)^2$ เซล แต่ละเซลมีขนาดเท่ากับ $h\times h$ ($h=1/(N+1)$)

$$\mathbf{A}\boldsymbol{u}=\boldsymbol{f} \tag{2.15}$$

โดยที่ $\mathbf{A}$ เป็นเมทริกซ์สัมประสิทธิ์ซึ่งเป็นเมทริกซ์สมมาตรบวกแน่นอนที่มี 5 แถบ $\boldsymbol{f}$ แทนโหลดเวกเตอร์ที่เกิดจากเทอมทางขวามือของ (2.12) และ $\boldsymbol{u}$ แทนเวกเตอร์ของตัวแปรไม่ทราบค่าที่ต้องการหา

ถ้า (2.15) เป็นระบบสมการเชิงเส้นขนาดใหญ่ การหาคำตอบของระบบสมการนี้ไม่เหมาะสมที่จะใช้วิธีการแบบโดยตรง เช่น การกำจัดแบบเกาซ์ หรือ การหาเมทริกซ์ผกผัน เนื่องจากจะใช้เวลานานมากและใช้หน่วยความจำในการทำงานมาก จึงมีการนำวิธีการทำซ้ำเข้ามาประมวลผล โดยในวิทยานิพนธ์นี้จะมุ่งเน้นไปที่การทำงานเชิงขนานของการทำซ้ำวิธีการต่างๆ โดยเน้นที่กรรมวิธีการทำซ้ำแบบเกาซ์-เซเดล

1.5 ประเภทของสัญญาณรบกวนบนภาพและการตัดสัญญาณรบกวนของภาพประเภทบวก

การตัดสัญญาณรบกวนของภาพมีจุดมุ่งหมายคือการนำสัญญาณรบกวนของภาพออกจากภาพให้มากที่สุดเท่าที่ทำได้ โดยไม่กระทบรายละเอียดหลักๆ ที่ปรากฏบนภาพ เช่น ขอบและพื้นผิวของภาพ

สัญญาณรบกวนบนภาพสามารถแบ่งได้กว้างๆ เป็นสองประเภท คือสัญญาณรบกวนประเภทบวก และสัญญาณรบกวนประเภทคูณ ซึ่งสัญญาณรบกวนประเภทบวกนั้นสามารถเขียนได้ตามสมการ (2.16)

$$z = u + \eta \tag{2.16}$$

โดยที่ตัวแปร z แทนภาพที่มีสัญญาณรบกวน η แทนสัญญาณรบกวนบนภาพ และ u แทนภาพที่ปราศจากสัญญาณรบกวน (ภาพต้นฉบับ) ซึ่งสัญญาณรบกวนประเภทนี้มักพบในขั้นตอนการรับภาพและมักมีการแจกแจงแบบปกติ ส่วนสัญญาณรบกวนประเภทคูณนั้นสามารถเขียนได้เป็น

$$z = u\xi \tag{2.17}$$

โดยที่ $u, \xi > 0$ และเมื่อ ξ แทนสัญญาณรบกวนประเภทคูณ

สัญญาณรบกวนประเภทคูณอาจมีชื่อเรียกอีกแบบหนึ่งว่าสัญญาณรบกวนแบบสเป็คเคิล (Speckle Noise) และในภาพบางชนิดมีการแจกแจงแบบแกมม่า สัญญาณรบกวนชนิดนี้ปรากฏบนภาพจากเรดาร์ช่องรับสัญญาณสังเคราะห์ (Synthetic Aperture Radar: SAR) ภาพการสะท้อนจากโซน่าร์และภาพอัลตราซาวน์ที่ รวมไปถึงภาพที่เกิดจากอุปกรณ์ที่มีการทำงานแบบสหนัย (Coherence Devices) เช่น กล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด (Scanning Electron Microscope: SEM) [14]

ในปัจจุบันการตัดสัญญาณรบกวนของภาพหลายวิธี วิธีหนึ่งได้แก่วิธีการ Graph Cut ซึ่งพิสูจน์แล้วว่ามีประสิทธิภาพและทำงานได้รวดเร็ว มีตัวอย่างในงานวิจัยในการใช้ Graph Cut กับการแปรผันรวม (TV) และ Standard Markov Random Field (MRF) [15] ในการตัดสัญญาณรบกวนของภาพอย่างมากมาย อย่างไรก็ตามปัญหาสำหรับการใช้วิธีนี้ ได้แก่การใช้หน่วยความจำที่

มากกว่าวิธีการอื่น [16] และการทำงานกับกราฟที่มีการขึ้นต่อกันระหว่างโหนด (Node Dependency) ทำให้การพัฒนาวิธีการนี้ให้ทำงานเชิงขนานทำได้ยาก

ในช่วงสองทศวรรษที่ผ่านมา วิธีการแปรผันแบบการแปรผันรวม [4] หรือที่เรียกว่าโมเดล ROF เป็นโมเดลหนึ่งในการตัดสัญญาณรบกวนของภาพประเภทบวกที่ได้รับความนิยม และภาพที่ถูกตัดสัญญาณรบกวนแล้วมีความคล้ายคลึงกับภาพต้นฉบับที่ปราศจากสัญญาณรบกวน [5, 17] โดยในวิธีนี้กำหนดให้ $u:\Omega\subset\Re^2\to V\subset\Re_0{}^+=\Re^+\cup\{0\}$ เป็นภาพที่ถูกกู้คืนจากภาพที่มีสัญญาณรบกวน $z:\Omega\to V$ ประเภทบวกบนที่มีการแจกแจงแบบปกติ โดยมีค่าเบี่ยงเบนมาตรฐาน σ สามารถแก้ปัญหานี้ได้โดยสมการ (2.18)

$$\min_u\left\{J(u)=D^{ROF}(u)+\alpha TV(u)\right\} \tag{2.18}$$

ในที่นี้

$$D^{ROF}(u)=\frac{1}{2}\int_\Omega (u-z)^2\,d\Omega \tag{2.19}$$

แทนเทอมวัดความเหมาะสมของข้อมูล (Data Fitting Term)

$$TV(u)=\int_\Omega |\nabla u|\,d\Omega \tag{2.20}$$

แทนเทอมเร็กกิวลาร์ไรซ์เซชันแบบการแปรผันรวมและ $\alpha>0$ แทนพารามิเตอร์เร็กกิวลาร์ไรซ์เซชัน (Regularization Parameter)

หลังจากใช้แคลคูลัสของการแปรผันจะได้สมการออยเลอร์-ลากรานจ์แบบไม่เป็นเชิงเส้นต่อไปนี้

$$\underbrace{-\alpha\nabla\cdot\frac{\nabla u}{|\nabla u|_\beta}+(u-z)}_{\mathbb{N}[u]}=0 \tag{2.21}$$

ภายใต้เงื่อนไขค่าขอบตามสมการที่ (2.22)

$$\partial_{\mathbf{n}}u=0 \text{ on } \partial\Omega \tag{2.22}$$

โดยที่ $|\nabla u|_\beta=\sqrt{|\nabla u|^2+\beta}, \beta>0$ ซึ่งค่า β เป็นค่าที่น้อยมากๆ เพื่อป้องกันการหารด้วย 0 ซึ่งวิธีการแก้สมการ (2.21) นั้นสามารถทำได้หลายวิธี เช่น วิธีการเดินเวลา (Time Marching) [18] วิธีการไพรมอล-ดูอัล (Primal-Dual Method) [19] และวิธีการทำซ้ำแบบจุดตรึง (Fixed Point Iterative Method) โดยในงานวิจัยนี้จะเลือกใช้วิธีการทำซ้ำแบบจุดตรึงที่นำเสนอโดย C. R. Vogel

และ M. E. Oman [6] ในปี ค.ศ. 1996 ที่มีอัตราการลู่เข้าสู่คำตอบสูงกว่าวิธีการอื่นๆ และสามารถนำวิธีการเกาซ์-เซเดลเข้ามาใช้ในส่วนการแก้ระบบสมการเชิงเส้นที่เกี่ยวข้องได้

1.6 การตัดสัญญาณรบกวนของภาพประเภทบวกและคูณ

ถึงแม้ว่าโมเดลการตัดสัญญาณรบกวนบนภาพประเภทบวกของ L. Rudin และคณะนั้นเป็นวิธีการที่ได้ผลดี แต่อย่างไรก็ตามผลการวิจัยของ T. W. Parks และ K. Hirakawa [20] นั้นได้แสดงว่าภาพโดยทั่วไปนั้นอาจไม่ได้ถูกรบกวนจากสัญญาณรบกวนเพียงประเภทเดียว [20] ต่อมา N. Chumchob และคณะ [21] ได้นำเสนอโมเดลของสัญญาณรบกวนบนภาพที่มีสัญญาณรบกวนทั้งสองประเภทดังนี้

$$z = u + k_0\eta + k_1 u\xi \tag{2.23}$$

โดยที่ η และ ξ แทนสัญญาณรบกวนประเภทบวกและประเภทคูณที่มีการแจกแจงแบบปกติและแบบแกมมาตามลำดับ และ k_0 และ k_1 เป็นพารามิเตอร์เพื่อควบคุมปริมาณสัญญาณรบกวนประเภทบวกและประเภทคูณตามลำดับ

นอกจากนี้ N. Chumchob และ C. Chantrapornchai [38] ได้นำเสนอโมเดลสัญญาณรบกวนชนิดใหม่ในภาพอัลตราซาวน์ที่มีทั้งประเภทบวกและคูณตามสมการ (2.24)

$$z = u + k_0\eta + k_1\sqrt{u}\,\eta \tag{2.24}$$

โดยที่ η เป็นสัญญาณรบกวนที่มีการแจกแจงแบบปกติ

ในการตัดสัญญาณรบกวนออกจากภาพที่มีสัญญาณรบกวนประเภทบวกและคูณตามสมการ (2.23) N. Chumchob และคณะได้เสนอโมเดลการตัดสัญญาณรบกวนบนภาพที่ใช้เร็กกิวลาร์ไรซ์เซชันแบบการแปรผันรวม ดังนี้

$$\min_u \{J_{\alpha_1,\alpha_2}(u) = D^{CCB}(u) + TV(u)\} \tag{2.25}$$

โดยที่ α_1 และ α_2 นั้นเป็นพารามิเตอร์เร็กกิวลาร์ไรซ์เซชันสำหรับสัญญาณรบกวนประเภทบวกและสัญญาณรบกวนประเภทคูณ ตามลำดับ และ

$$TV[u] = \int_\Omega |\nabla u|_\beta \, d\Omega = \int_\Omega \sqrt{u_x^2 + u_y^2 + \beta} d\Omega, \beta > 0 \tag{2.26}$$

เมื่อ β เป็นค่าที่น้อยกว่า 1 มากๆ เพื่อป้องกันการหารด้วย 0 และ

$$D^{CCB}(u)=\frac{\alpha_1}{2}\int_{\Omega}\left(u-z\right)^2 d\Omega+\alpha_2\int_{\Omega}\left(u+ze^{-u}\right)d\Omega \tag{2.27}$$

หลังจากใช้แคลคูลัสของการแปรผันและวิธีการผลต่างอันตะเพื่อคำนวณสมการออยเลอร์-ลากรานต์ที่สอดคล้องกับ (2.25) ด้วยการแบ่งโดเมนออกเป็น N ช่องเท่าๆ กันในแต่ละมิติ โดยแต่ละช่องมีระยะห่างเป็น h จะได้สมการออยเลอร์-ลากรานจ์ที่ถูกดิสครีตไทซ์ ดังนี้

$$\underbrace{-\mathrm{K}^h(u^h)_{i,j}+\alpha_1((u^h)_{i,j}-(z^h)_{i,j})+\alpha_2\left(1-(z^h)_{i,j}e^{-(u^h)_{i,j}}\right)}_{\boldsymbol{N^h(u^h)_{i,j}}}=0, \tag{2.28}$$

โดยที่ u^h แทนภาพที่ถูกดิสครีไทซ์ด้วยระยะห่าง h และ

$$\mathrm{K}^h(u^h)_{i,j}=\left[\frac{\delta_x^-}{h_x}\left(\frac{D(u^h)_{i,j}\delta_x^+(u^h)_{i,j}}{h_x}\right)+\frac{\delta_y^-}{h_y}\left(\frac{D(u^h)_{i,j}\delta_y^+(u^h)_{i,j}}{h_y}\right)\right] . \tag{2.29}$$

โดยที่

$$D(u^h)_{i,j}=\frac{1}{\sqrt{\left(\frac{\delta_x^+(u^h)_{i,j}}{h}\right)^2+\left(\frac{\delta_y^+(u^h)_{i,j}}{h}\right)^2+\beta}}, \tag{2.30}$$

และ

$$\delta_x^+(u^h)_{i,j}=\pm\left((u^h)_{i\pm1,j}-(u^h)_{i,j}\right),$$

$$\delta_y^+(u^h)_{i,j}=\pm\left((u^h)_{i,j\pm1}-(u^h)_{i,j}\right), \tag{2.31)- (2.32}$$

ตามลำดับ ระบบสมการไม่เป็นเชิงเส้นที่เกิดจาก (2.28) สามารถทำได้ในลักษณะเดียวกับการตัดสัญญาณรบกวนของภาพประเภทบวกคือสามารถนำวิธีการทำซ้ำแบบจุดตรึงเข้ามาใช้แก้ปัญหาได้ ซึ่งในการทำซ้ำแบบจุดตรึงนั้นสามารถประยุกต์ใช้วิธีการเกาซ์-เซเดลเชิงขนานที่งานวิจัยนี้พัฒนาในบทที่ 4

2. วรรณกรรมที่เกี่ยวข้อง

ในส่วนนี้จะกล่าวถึงงานวิจัยที่เกี่ยวข้อง ได้แก่ งานวิจัยที่เกี่ยวกับการแก้ระบบสมการเชิงเส้นแบบการทำซ้ำ ซึ่งเน้นไปที่แอปพลิเคชันที่มีการใช้งานตัวแก้สมการประเภทเกาซ์-เซเดลและ

การผ่อนปรนเกินสืบเนื่องในหลากหลายสาขา ซึ่งในการเร่งความเร็วอัลกอริทึม วิธีหนึ่งที่สามารถทำได้คือการทำการทำงานแบบขนานมาช่วย และงานวิจัยที่เกี่ยวข้องในการพัฒนาอัลกอริทึมเกาซ์-เซเดลแบบขนาน รวมทั้งงานวิจัยที่เกี่ยวข้องกับการการกำจัดสัญญาณรบกวนของภาพแบบการแปรผันรวม

2.1 การแก้สมการเชิงเส้นแบบการทำซ้ำ

ในหลายปีมาแล้วได้มีการนำการแก้ระบบสมการเชิงเส้นแบบการทำซ้ำไปใช้ในงานประยุกต์ที่หลากหลาย ในหัวข้อนี้ผู้วิจัยจะยกตัวอย่างงานวิจัยในสาขาต่างๆ ที่ใช้วิธีการทำซ้ำแบบเกาซ์-เซเดลเพื่อแก้ปัญหา

ในปี ค.ศ. 2004 D. Lelic และ Z. Gajic ได้นำวิธีการทำซ้ำแบบเกาซ์-เซเดลไปใช้กับงานด้านการคุมการจ่ายพลังงานการส่งสัญญาณวิทยุ [21] โดยเรียกชื่อปัญหานี้ว่า Spread Spectrum Cellular Radio ซึ่งเป็นการปรับค่าพลังงานในการส่งในเครือข่าย 3G และเครือข่ายโทรศัพท์เคลื่อนที่แบบ CDMA โดยจะทำการปรับค่าพลังงานที่สามารถตัดสัญญาณรบกวนให้เหมาะกับผู้ที่กำลังใช้เครือข่ายไร้สายจำนวน n คน ตามสมการที่ (2.33)

$$\gamma_i = \frac{g_{ii} p_i}{\sum_{j=1, j\neq i}^{n} g_{ij} p_j + \sigma_i^2}, i = 1, 2..., n \tag{2.33}$$

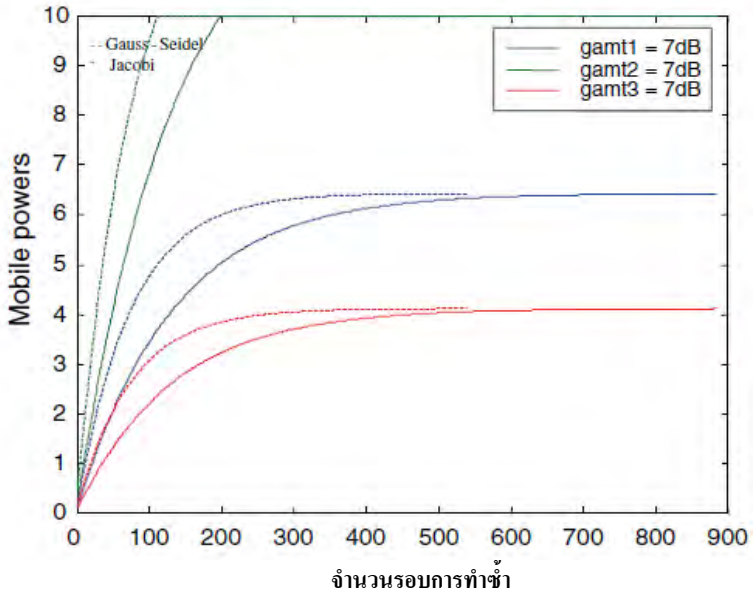
โดยที่ p_i แทนพลังงานที่จะส่งไปยังผู้ใช้คนที่ i, σ_i แทนสัญญาณรบกวน, g_{ij} แทนสัญญาณที่ส่งจากสถานีเคลื่อนที่ i และสถานีหลัก j และ γ_i เป็นค่าพลังงานที่เหมาะสมที่ต้องการหา ซึ่งปัญหาข้างต้นสามารถถูกนำเสนอในรูประบบสมการเชิงเส้นต่อไปนี้

$$\mathbf{A}\boldsymbol{p} = \boldsymbol{b} \tag{2.34}$$

ในปี ค.ศ. 2004 โดยที่ **A** เป็นเมทริกซ์สัมประสิทธิ์ $\boldsymbol{p}$ เป็นเวกเตอร์หนึ่งหน่วยของผู้ใช้ และ $\boldsymbol{b}$ แทนค่าพลังงาน โดย D. Lelic และคณะ [22] ได้พัฒนาการเร่งการลู่เข้าของวิธีการเกาซ์-เซเดลสำหรับปัญหานี้ โดยใช้วิธีการของ F. Melagic [23]โดยกำหนดค่า β ซึ่งเขียนได้ตามสมการที่ (2.35)

$$p_i^{(k+1)} = \left[1 - \beta + \beta \frac{\gamma_i^{tar}}{\gamma_i(k)} \right] p_i^{(k)}, i = 1, 2, ..., n \qquad (2.35)$$

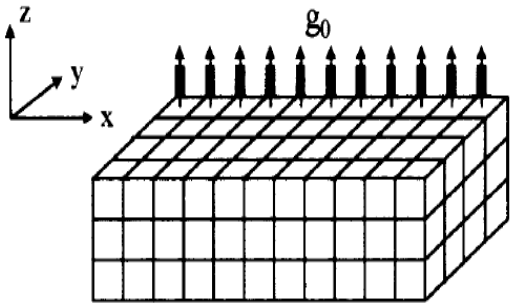
ผลของการทดลองของงานวิจัยนี้นั้นวัดจากพลังงานที่ได้รับ โดยทดลองกับ 3 ปัญหาคือ gamt1, gamt2 และ gamt3 โดยเทียบกับวิธีการยาโคบิ จะเห็นว่าเกาซ์-เซเดลนั้นให้ประสิทธิภาพในด้านพลังงานที่ดีกว่าในยาโคบิแบบเดิมซึ่งแสดงได้ตามภาพที่ 2.3



ภาพที่ 2.3 การเทียบพลังงานระหว่างวิธีการเกาซ์-เซเดลและวิธีการยาโคบิตามจำนวนรอบของการทำซ้ำ(แกน x) บนปัญหาต่างๆ [22]

นอกจากนี้ยังมีการนำวิธีการเกาซ์-เซเดลไปใช้ในงานอื่นๆ เชิงวิทยาศาสตร์อีก เช่น การวิเคราะห์อุณหภูมิแบบ 3 มิติ โดยในปี ค.ศ. 2001 Z. Y. Luo และคณะ[24] ได้นำวิธีการผ่อนปรนสืบเนื่องหรือเกาซ์-เซเดลแบบถ่วงน้ำหนักเข้ามาวิเคราะห์อุณหภูมิเชิงสถิติแบบ 3 มิติ ซึ่งวิธีการผ่อนปรนสืบเนื่องที่นำมาใช้นี้ให้ผลดีกว่าวิธีเดิมที่ใช้วิธีการวิธีการมอนติคาร์โล (Monte-Carlo Method)

ในการวัดความร้อนและการวิเคราะห์นั้นใช้ชิป VLSI ขนาด 45 นาโนเมตร ในการตรวจจับอุณหภูมิซึ่งมีการประสิทธิภาพในการตรวจวัดที่สูงมาก ซึ่งการตรวจจับนั้นจะมีการแบ่งปัญหาออกเป็นเซลแบบ 3 มิติเท่าๆกัน และมีการใช้ตัวตรวจจับอุณหภูมิตามภาพที่ 2.4



ภาพที่ 2.4 การวางตัวตรวจจับอุณหภูมิกับปัญหาแบบสามมิติ [24]

จากปัญหานี้ Z. Y. Luo และคณะได้พัฒนาอัลกอริทึมใหม่ที่เรียกว่า SN-SOR ซึ่งสามารถแยกปัญหาและทำทีละโหนดของการวัดค่าได้ ซึ่งสามารถเขียนปัญหาเป็นลักษณะรอบของการทำซ้ำได้ดังสมการ (2.36)

$$\delta T_b^{(k+1)} = | T_b^{(k+1)} \cdot T_b^{(k)} | \qquad (2.36)$$

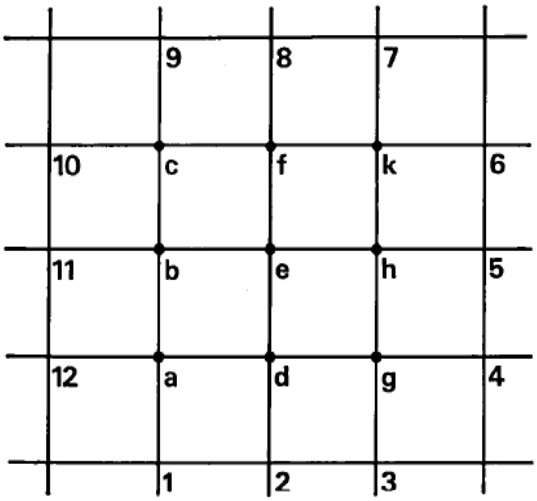
โดยที่ δ เป็นค่าถ่วงน้ำหนัก และ T_b เป็นค่าเวกเตอร์ของอุณหภูมิ ซึ่งเมื่อใช้วิธีการผ่อนปรนสืบเนื่องในการแก้ปัญหาโดยได้แบ่งปัญหาออกเป็น 900 บล็อกและมีตัวอย่าง 5000 ชิ้น พบว่าวิธีการผ่อนปรนเกินสืบเนื่องนั้นจะใช้เวลาเพียง 66.66 วินาทีเท่านั้น เมื่อหารด้วยจำนวนตัวตรวจวัดที่มี 450 ตัวแล้ว แต่ละจุดจะใช้เพียง 0.1481 วินาที และสามารถลดค่าความผิดพลาดได้จาก 5.748% เหลือเพียง 0.139%

อย่างไรก็ดีตัวอย่างงานวิจัยที่เสนอมานั้นใช้กับปัญหาในขนาดเล็ก เมื่อข้อมูลมีขนาดใหญ่ขึ้นจึงมีความจำเป็นที่จะพัฒนาวิธีการเกาซ์-เซเดลเชิงขนานเพื่อลดเวลาในการทำงานได้ ดังจะอธิบายในหัวข้อ 2.2 ต่อไป

2.2 วิธีการเกาซ์-เซเดลเชิงขนาน

ในปี ค.ศ. 1984 D. J. Evans ได้มีความพยายามพัฒนาวิธีการผ่อนปรนสืบเนื่องบนเครื่องคอมพิวเตอร์แบบ MIMD [25] ซึ่งเป็นซุปเปอร์คอมพิวเตอร์ในสมัยนั้น มีจำนวนหน่วยประมวลผลทั้งหมด 4 หน่วยประมวลผล ไม่มีการทำงานแบบเธรด งานวิจัยนี้มุ่งเน้นไปที่การแก้ระบบสมการเชิงเส้นป๋วซงสองมิติ ที่มีการใช้เงื่อนไขค่าขอบแบบดีรีเคลโดยเน้นไปที่การพัฒนาวิธีการผ่อนปรนสืบเนื่องแบบขนานเทียบกับวิธีการเกาซ์-เซเดลแบบดำ-แดง (Red-Black Gauss-Seidel)

ในงานวิจัยนี้วิธีการที่ผู้วิจัยนำเสนอได้เริ่มจากการแบ่งเมทริกซ์ออกเป็นบล็อกย่อยๆ ที่มีขนาดต่างๆคือ 2×1, 2×2 และ 3×3 โดยมีค่าขอบในแต่ละบล็อกร่วมกัน ซึ่งสามารถแสดงได้ตามภาพที่ 2.5



ภาพที่ 2.5 แสดงการทำงานของอัลกอริทึมบนบล็อกขนาดขนาด 3×3 [25]

จากภาพที่ 2.5 จะเห็นว่าในแต่ละบล็อกนั้นมีการทำงาน 3×3 ช่อง เรียงลำดับตามอักษร (a ถึง k) และค่าของขอบรอบข้าง (ตัวเลข 1-12) ในการทำงานของอัลกอริทึมเกาซ์-เซเดลเชิงขนานแต่ละบล็อกจะใช้ขอบร่วมกันแล้วจึงมีการส่งข้อมูลไปยังหน่วยประมวลผลข้างเคียงเพื่อแก้ปัญหาส่วนขอบและเริ่มการทำซ้ำในรอบการทำซ้ำถัดไป

จากอัลกอริทึมเกาซ์-เซเดลเชิงขนานที่ D. J. Evans ได้นำเสนอ เมื่อแสดงการพิสูจน์ทางคณิตศาสตร์บนหน่วยประมวลผล 2 หน่วย โดยมีขนาดบล็อก 2×1 จะได้จำนวนการดำเนินการ (Operation) ตามสมการ (2.37)

$$OP = 2m^2 mul + \frac{7}{2}(m^2 add) \qquad (2.37)$$

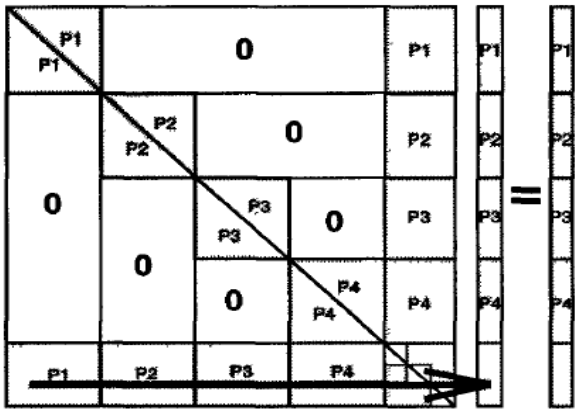
โดยที่ OP คือจำนวนการดำเนินการทั้งหมด m แทนขนาดของกริดใน 1 มิติ mul และ add คือจำนวนการดำเนินการคูณและบวกตามลำดับ ซึ่งจากสมการ (2.37) เมื่อเพิ่มจำนวนหน่วยประมวลผลเป็น 4 หน่วยจะได้จำนวนการดำเนินการตามสมการ (2.38)

$$OP = \frac{5}{4}m^2 mul + \frac{9}{4}(m^2 add) \qquad (2.38)$$

จากสมการ(2.38) จะเห็นได้ว่าจำนวนการดำเนินการเมื่อทำงานกับหน่วยประมวลผลทั้งสี่นั้นมีจำนวนการดำเนินการน้อยกว่าในสมการ (2.37) มาก และ D. J. Evans ยังได้ทดลองกับช่วงการแบ่งโดเมนปัญหาขนาดตั้งแต่ 10-60 พบว่าเมื่อแบ่งบล็อกขนาดใหญ่ขึ้น (เกิน 3×3) ผลที่ได้นั้นจะแย่ลง เมื่อทดลองกับ 4 หน่วยประมวลผล โดยสามารถวัดค่า Speedup ได้ถึง 3.84 เท่าบนกริดขนาด 16×16 เมื่อเทียบกับการทำงานบนหน่วยประมวลผลตัวเดียว

รายงานผลการวิจัยของ D. J. Evans ระบุว่าวิธีการนี้สามารถทำงานได้ดีกว่าการทำงานของวิธีการเกาซ์-เซเดลแบบแดงดำ (Red-Black Gauss-Seidel) เนื่องจากมี Overhead ที่น้อยกว่า และสามารถทำงานกับหน่วยประมวลผลหลายหน่วยประมวลผลได้ซึ่งแตกต่างกับวิธีการเกาซ์-เซเดลแบบแดงดำที่เหมาะสมสำหรับหน่วยประมวลผลสองหน่วย

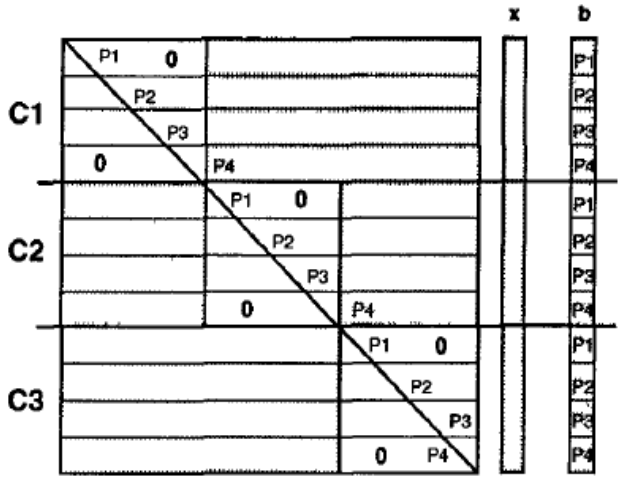
ต่อมาในปี ค.ศ. 1994 D. P. Koester และคณะ [26] ได้พัฒนาวิธีการเกาซ์-เซเดลเชิงขนานบนคลัสเตอร์ที่แต่ละโหนดบนเครื่องคอมพิวเตอร์ CM-5 Thinking Machine ที่มีหน่วยประมวลผลหลายหน่วยและบนสถาปัตยกรรมแบบกระจายเพื่อใช้ในงานประยุกต์ด้านระบบพลังงานไฟฟ้าโดยคณะผู้วิจัยได้เสนอวิธีการแบบ Block-Bordered-diagonal Form และเมทริกซ์แบบหลายสี (Multicolor Matrix) การประมวลผลแบบ Block-Border-diagonal Form นั้น สามารถแสดงได้ดังภาพที่ 2.6



ภาพที่ 2.6 การทำงานในแบบ Block-Border-Diagonal ของวิธีการเกาซ์-เซเดลเชิงขนานที่นำเสนอโดย [26]

จากภาพที่ 2.6 P1, P2, P3 และ P4 แสดงการทำงานของหน่วยประมวลผลตัวที่ 1, 2, 3 และ 4 ตามลำดับ โดยปัญหาที่กำหนดเป็นเมทริกซ์มากเลขศูนย์ ทำให้พื้นที่ส่วนใหญ่มีค่าเป็น 0 และมีการแบ่งข้อมูลออกเป็นบล็อกย่อยๆ ในการประมวลผลด้วยวิธีนี้จะเริ่มทำการประมวลผลเพื่อหาสมาชิกแต่ละตัวของคอลัมน์เวกเตอร์ $\boldsymbol{x}$ ตามแนวแทยงมุม (Diagonal) ของเมทริกซ์ก่อนแล้วจึงส่งข้อความ (message) ไปยังบล็อกข้างเคียงผ่าน Remote Procedure Calls (RPCs)ไปยังหน่วยประมวลผลตัวอื่น โดยทำทุกบล็อกยกเว้นบล็อกขวาล่างสุด เมื่อประมวลผลครบทุกบล็อกในรอบของการทำซ้ำแล้วจึงทำงานกับบล็อกสุดท้ายต่อไป

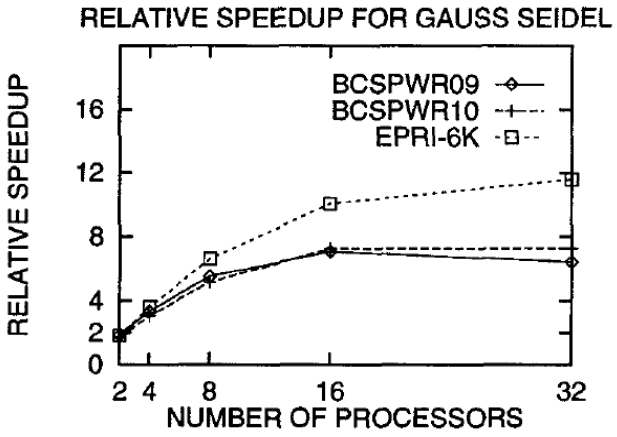
นอกจากนี้คณะวิจัยยังได้นำเสนอวิธีการการทำงานเชิงขนานแบบเมทริกซ์หลายสี ซึ่งเหมาะกับเมทริกซ์ที่สมมาตรตามแนวทแยงมุม โดยได้แบ่งเมทริกซ์ออกเป็นกลุ่ม (หรือสี) ซึ่งในภาพที่ 2.6 แสดงการแบ่ง 3 สีคือ C1 - C3 ในการทำงานนั้นในแต่ละหน่วยประมวลผลจะทำงานในบล็อกของแต่ละสีจนครบทุกสี ทำเฉพาะครึ่งบนของเมทริกซ์ แล้วจึงกระจายข้อมูลไปยังหน่วยประมวลผลอื่นๆ ทุกหน่วย เพื่อหาค่า $\boldsymbol{x}$ ต่อไป โดยการแบ่งสีนั้นสามารถแสดงได้ตามภาพที่ 2.7



ภาพที่ 2.7 แสดงตัวอย่างการทำงานแบบเมทริกซ์หลายสี [26]

ในงานวิจัยที่กล่าวมานี้ D. P. Koester และคณะได้นำทั้งสองวิธีนี้มารวมกัน จัดเมทริกซ์ให้อยู่ในรูปของ Block-Border-Diagonal Form และในการแก้ปัญหาในแต่ละบล็อกจะใช้วิธีการเมทริกซ์หลายสี โดยได้ดำเนินการทดลองโดยเขียนบนภาษา C และใช้การสื่อสารระหว่างหน่วยประมวลผลแบบ Blocking ซึ่งได้ทดลองกับปัญหาทางการจ่ายไฟฟ้า 3 ปัญหาคือ BCSPWR09,

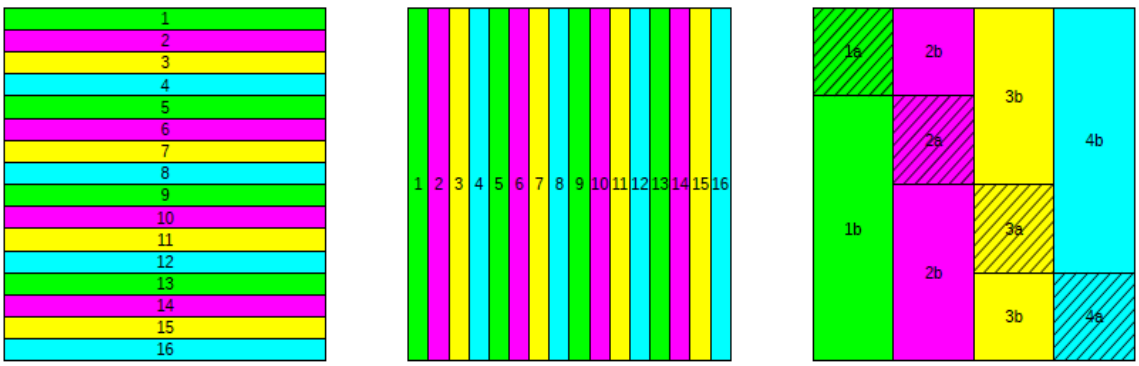
BCSPWR10 และ EPRI-6K ซึ่งมีจำนวนโหนดเท่ากับ 1723, 5300 และ 4180 โหนดตามลำดับ ผลการทดลองนั้นวัดออกมาเป็นค่า Speedup บนจำนวนหน่วยประมวลผลขนาดต่างๆได้ตามภาพที่ 2.8



ภาพที่ 2.8 ค่า Speedup เมื่อทดลองกับปัญหาต่างๆ บนจำนวนหน่วยประมวลผลต่างกัน [27]

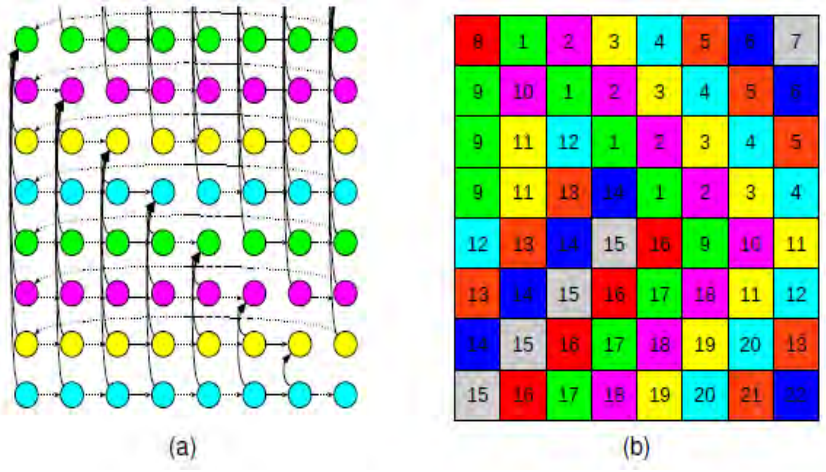
จากภาพที่ 2.8 จะเห็นว่าอัลกอริทึมนี้ได้ผลดีเมื่อจำนวนหน่วยประมวลผลตั้งแต่ 2-16 หน่วย เมื่อเกิน 16 หน่วยประมวลผลจะพบว่าประสิทธิภาพของอัลกอริทึมนี้ลดลงอย่างมาก ซึ่ง D. P. Koester และคณะได้ตั้งสมมติฐานว่าน่าจะเกิดจาก Overhead ในการประมวลผลและการกระจายงานที่ยังไม่ดีนัก

ในปี ค.ศ. 2009 H. Courtecuisse และคณะ [27] ได้เสนอการทำวิธีการเกาซ์-เซเดลบนหน่วยประมวลผลแบบมากแกน (Many-core Processor) โดย H. Courtecuisse และคณะได้ออกแบบการทำงานของวิธีการเกาซ์-เซเดลเชิงขนาน 3 วิธีคือวิธีการแบบแถว วิธีการแบบคอลัมน์ และวิธีการแบบบล็อกซึ่งแสดงได้ตามภาพที่ 2.9



ภาพที่ 2.9 : รูปแบบการประมวลผล Gauss-Seidel แบบต่างๆที่เสนอ [27]

นอกจากนี้ H. Courtecuisse และคณะยังเลือกวิธีกราฟซึ่งสามารถแสดงลำดับขั้นตอนการทำงานดังภาพที่ 2.10



ภาพที่ 2.10 (a) รูปแบบและลำดับขั้นตอนการทำงานแบบกราฟ และ (b) แสดงตำแหน่งแต่ละเธรดเมื่อนำไปใช้บนเมทริกซ์จริง [27]

ในงานวิจัยนี้ H. Courtecuisse และคณะได้นำเสนอการอัพเดทที่เรียกว่า “Atomic Update” ซึ่งการอัพเดทข้อมูลของแต่ละเธรดจะต้องรอเพื่อให้เธรดรอบข้างนั้นประมวลผลให้เสร็จก่อน โดยใช้ตัวนับ (Counter) ซึ่งเก็บไว้บน Shared Memory เพื่อให้เข้าถึงได้รวดเร็ว โดยการตัดสินใจว่าเธรดใดจะทำงานต่อได้นั้น (T_N) สามารถแสดงได้ตามเงื่อนไขใน (2.39)

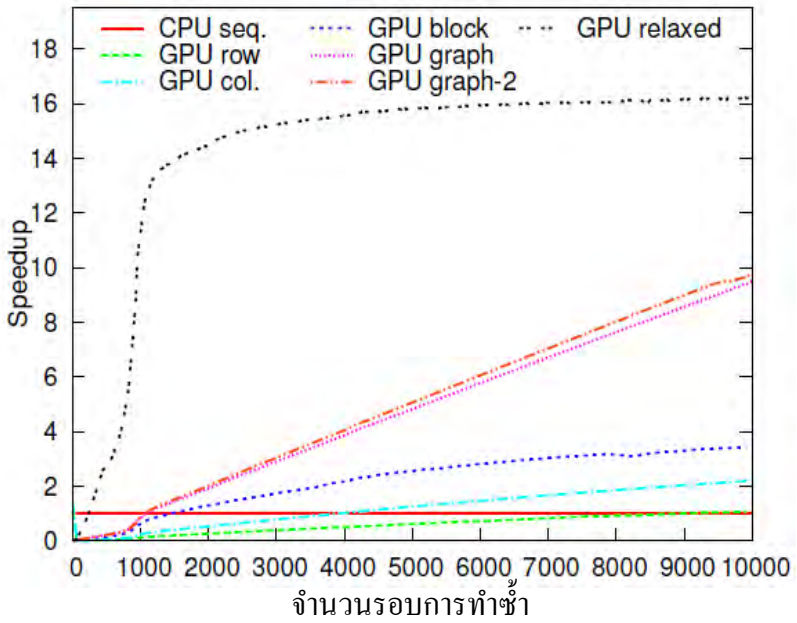
$$T_N = \begin{cases} counter > \lceil n/m \rceil (iter-1) + \lfloor j/m \rfloor ; j > i \\ counter > \lceil n/m \rceil (iter) + \lfloor j/m \rfloor ; j < i \end{cases} \qquad (2.39)$$

เมื่อ n เป็นจำนวนเธรดในแต่ละกลุ่ม และ m เป็นจำนวนกลุ่ม $iter$ แทนหมายเลขรอบของการทำซ้ำ และ i และ j แทนหมายเลขแถวและหลักของเมทริกซ์

โดยทดลองกับปัญหาที่เรียกว่า Linear Complementary Problem (LCP) ที่มีรูปแบบของปัญหาตามสมการ (2.40)

$$\boldsymbol{w} - \mathrm{A}\boldsymbol{x} = \boldsymbol{b} \qquad (2.40)$$

H. Courtecuisse และคณะได้ทดลองงานวิจัยนี้ทั้งบน CPU และ GPUs โดยแบ่งการทดลองออกเป็น Single Precision และ Double Precision โดยใช้ขนาดปัญหาตั้งแต่ 1,000-10,000 ค่าที่ต้องการหา ผลการทดลองถูกแสดงในภาพที่ 2.11



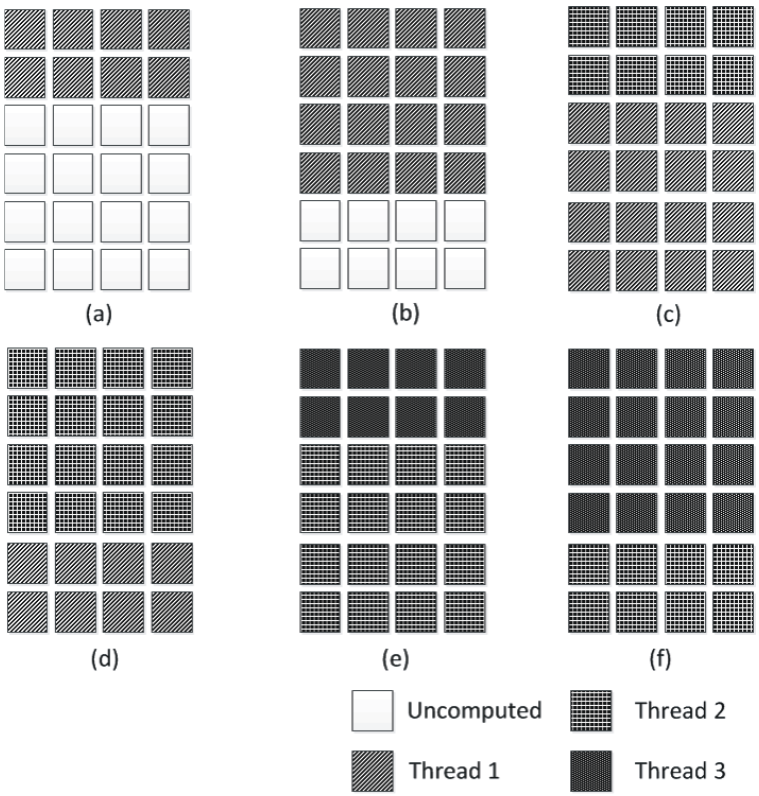
ภาพที่ 2.11 ผลการเปรียบเทียบระหว่างการรันบน CPU และ GPUs ด้วยวิธีการต่างๆ [27]

จากภาพ 2.11 หากตัดเรื่องการถ่วงน้ำหนัก (GPU relaxed) ออกแล้ว พบว่าการทำงานแบบกราฟนั้นทำได้รวดเร็วที่สุด โดยเมื่อปัญหามีขนาดใหญ่ขึ้นจะได้ค่า Speedup มากขึ้นและมีการใช้เธรดจำนวนมากของ GPUs ได้ดี แต่อย่างไรก็ตาม วิธีนี้ใช้การผสานการทำงานของเธรดเป็นจำนวนมาก และต้องมีการสื่อสารระหว่างเธรดจำนวนมากในแต่ละรอบการทำงาน

วิธีการใหม่อีกวิธีการของการประมวลผลแบบขนานของวิธีการเกาซ์-เซเดลเชิงขนาน เรียกว่าวิธีการเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง (Sliding Window Gauss-Seidel: SWGS) [2] เป็นวิธีการของเกาซ์-เซเดลเชิงขนานที่มีประสิทธิภาพ ซึ่งเหมาะสำหรับการแก้ระบบสมการเชิงเส้นขนาดใหญ่ วิธีการนี้มุ่งเน้นในการแก้ไขปัญหาบนเครื่องคอมพิวเตอร์ที่มีหลายหน่วยประมวลผล โดยใช้ OpenMP ซึ่งจะกล่าวอย่างละเอียดต่อไป

วิธีการเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง [2] เป็นอัลกอริทึมแบบขนานสำหรับแก้ระบบสมการเชิงเส้นที่มีประสิทธิภาพดีที่สุดตามเท่าที่ศึกษามา ในการทำเกาซ์-เซเดลแบบเลื่อนบาน

หน้าต่างจะแบ่งการประมวลผลในแต่ละการทำซ้ำออกเป็นหน่วยย่อยๆ โดยทำการแบ่งแถวของเมทริกซ์ออกเป็นบล็อก (Block) และ วินโดว์ (Window) โดยเริ่มจากการประมวลผลในบล็อกแรกก่อน เมื่อประมวลผลเสร็จแล้วทุกเธรดจะเลื่อนการประมวลผลไป 1 บล็อก และทำการเลื่อนไปเรื่อยๆ จนทุกเธรดทำงานได้ทุกครบทุกแถว โดยการทำงานเชิงขนานของอัลกอริทึมนี้สามารถแสดงได้ตามภาพที่ 2.12



ภาพที่ 2.12 ตัวอย่างการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างบนโดเมนขนาด 6×4 กับการใช้เธรด 3 เธรด

ในภาพที่ 2.12 นั้นในขั้นตอน (a) เธรดแรกจะเริ่มทำบล็อกแรกของโดเมนปัญหาซึ่งในที่นี้กำหนดขนาดบล็อกเป็น 2 ซึ่งหมายถึงทำ 2 แถวแรกก่อน และเมื่อทำการประมวลผลเรียบร้อยแล้วจะเลื่อนการทำงานลงมา 1 บล็อกตามขั้นตอน (b) ซึ่งจะทำจนครบตามจำนวนแถวที่ระบุไว้ซึ่งเรียกว่าขนาดวินโดว์ และในขั้นตอน (c) จะทำการเลื่อนลงมาอีก 1 บล็อกเมื่อขั้นตอน (b) เสร็จสิ้นและเธรดที่สองจะเริ่มเข้าไปทำที่ช่องว่าง 2 แถวแรก ซึ่งทำตามขั้นตอนเหล่านี้ไปเรื่อยๆ จนทุกเธรดประมวลผลเสร็จจนครบทุกแถวจึงหยุดการทำงาน

จากที่ H. Courtecuisse และคณะได้ทดลองระบุว่าขนาดของวินโดว์นั้นควรจะมีขนาดใหญ่กว่าขนาดของบล็อกมากๆ เนื่องจากบนหน่วยประมวลผล CPU เมื่อทำงานเดียวต่อเนื่องในแต่ละหน่วยประมวลผลจะลดเวลาในการสื่อสารและการผสานการทำงานของเธรดได้มาก ในวิทยานิพนธ์นี้ ผู้วิจัยจะทำการทดลองกับขนาดบล็อกและวินโดว์ที่มีขนาดต่างๆ เพื่อหาขนาดที่เหมาะสมสำหรับการทำงานบนหน่วยประมวลผลกราฟิกส์อย่างมีประสิทธิภาพ

จากภาพที่ 2.12 จะเห็นว่าเราสามารถทำงานแบบขนานบนวิธีการเกาซ์-เซเดลได้ แต่หากมีเธรดจำนวนมากจะเกิดผลเสียคือเธรดหลังๆ จะใช้เวลารอคอยที่จะเริ่มทำงานนานมาก เช่นหากมีเธรดจำนวน 128 เธรด จะทำให้เธรดสุดท้ายต้องรอถึง 254 รอบจึงจะสามารถเริ่มทำงานได้ ทำให้ประสิทธิภาพของอัลกอริทึมนี้ลดลงไปอย่างมากเมื่อทำงานกับฮาร์ดแวร์ที่รองรับหลายเธรด หรือมีหน่วยประมวลผลจำนวนมาก วิธีแก้ไขปัญหานี้ผู้วิจัยได้แก้ปัญหาโดยเพิ่มการทำงานแบบขนานไปยังแนวคอลัมน์ด้วย ซึ่งจะกล่าวถึงในการพัฒนาของอัลกอริทึมนี้ในบทที่ 4

สำหรับการเปรียบเทียบ รวมทั้งข้อดีข้อเสียของแต่ละอัลกอริทึมของเกาซ์-เซเดลเชิงขนานที่มีอยู่สามารถทำการสรุปและเปรียบเทียบได้ตามตารางที่ 2.5

ตารางที่ 2.5 เปรียบเทียบข้อดีข้อเสียของอัลกอริทึมเกาซ์-เซเดลเชิงขนาน

ชื่ออัลกอริทึม	ข้อดี	ข้อเสีย
วิธีการผ่อนปรนเกินสืบเนื่องของ D. J. Evans [25]	สามารถทำงานเชิงขนานได้ และประสิทธิภาพสูงกว่าอัลกอริทึมเกาซ์-เซเดลแบบแดง-ดำ	เน้นแก้ปัญหาที่เจาะจงเกินไปและเหมาะสำหรับหาคำตอบจากเมทริกซ์มากเลขศูนย์ได้เท่านั้น
Block-Border Diagonal Form [26]	สามารถทำงานได้บนหน่วยประมวลแบบไม่จำกัดจำนวน	มุ่งเน้นทำงานบนเมทริกซ์แบบมากเลขศูนย์เท่านั้น
H. Courtecuisse [27]	ทำงานได้ทั้งเมทริกซ์มากเลขศูนย์และเมทริกซ์หนาแน่น (Dense Matrix) รวมทั้งใช้หน่วยประมวลผลได้เป็นจำนวนมาก	ใช้เวลาในการผสานการทำงาน และการสื่อสารระหว่างเธรดมาก

ตารางที่ 2.5 เปรียบเทียบข้อดีข้อเสียของอัลกอริทึมเกาซ์-เซเดลเชิงขนาน (ต่อ)

ชื่ออัลกอริทึม	ข้อดี	ข้อเสีย
Sliding Window Gauss-Seidel [2]	ลดเวลาผสานการทำงานระหว่างเธรดได้มาก ทำงานได้กับเมทริกซ์ทั้งหนาแน่นและมากเลขศูนย์ ไม่ซับซ้อน	ถูกออกแบบมาให้ใช้กับเครื่องที่มีหน่วยประมวลผลไม่มากบนพื้นฐาน OpenMP

2.3 การกำจัดสัญญาณรบกวนบนภาพโดยการแปรผันรวม

จากโมเดลในการกำจัดสัญญาณรบกวนภาพแบบการแปรผันรวมในสมการ (2.16) นั้น ได้มีงานวิจัยต่างๆ ได้นำโมเดลนี้ไปประยุกต์ใช้ ผู้วิจัยจะขอยกตัวอย่างงานวิจัยที่ได้ใช้โมเดลการแปรผันรวมซึ่งมีการนำไปประยุกต์ใช้ในงานที่หลากหลาย

ในปี ค.ศ. 2011 F. Lenzen และคณะได้นำการกำจัดสัญญาณรบกวนแบบการแปรผันรวมไปใช้กับงานประยุกต์ด้านการบิน โดยได้เชื่อมต่องานนี้กับกล้องที่เรียกว่า Time-of-Flight (ToF) [28] ซึ่งเป็นกล้องที่ทำงานด้วยระบบอินฟราเรด สามารถคำนวณความสูงและความลึกเมื่อถ่ายจากเครื่องบินได้โดยสร้างเป็นแอมปลิจูดสำหรับความสูงต่างๆ แต่กล้องประเภทนี้มีสัญญาณรบกวนประเภทบวกจำนวนมาก

ในการทดลองนั้นผู้ทดลองได้ใช้โมเดลในการกำจัดสัญญาณรบกวนหลายประเภทเพื่อเปรียบเทียบโดยมีการถ่วงน้ำหนักตามระยะใกล้ไกลที่วัดได้โดยกล้อง ToF โดยจากสมการ (2.16) สามารถปรับเปลี่ยนให้มีการถ่วงน้ำหนักในเทอมของเร็กกูล่าไรซ์เซชันดังนี้

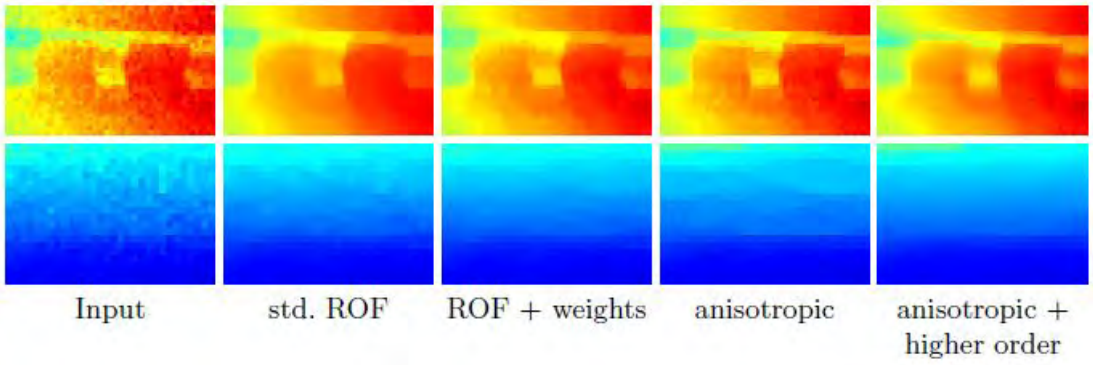
$$\min_{u}\left\{J(u)=\tfrac{1}{2}\int_{\Omega} w_{i,j}(u-z)^2 d\Omega+\Phi(u)\right\} \qquad (2.41)$$

ตัวแปรที่เพิ่มขึ้นมาในสมการ (2.41) ได้แก่ $w_{i,j}$ ซึ่งผู้วิจัยกำหนดให้มีค่าเป็น $A_{i,j}^2/\sigma_0^2$ โดยที่ $A_{i,j}$ เป็นค่าของเมทริกซ์แอมพลิจูดซึ่งแสดงระยะห่างระหว่างกล้องและวัตถุ และค่า σ_k แทนค่าเบี่ยงเบนมาตรฐานของค่า $A_{i,j}$ ในรอบการทำซ้ำ k

และ $\Phi(u)$ สามารถเขียนได้ตามสมการ (2.42)

$$\Phi(u) = \sum_{i,j} \sup\{\upsilon_{i,j}^{T} L_{i,j}(u) \mid \upsilon_{i,j} \in C_{i,j}\} \tag{2.42}$$

F. Lenzen และคณะได้เปรียบเทียบกับการใช้การแปรผันรวมปกติกับแบบถ่วงน้ำหนัก และการกำจัดสัญญาณรบกวนแบบ Anisotropic [29] ผลการทดลองภาพที่ได้จะได้ตามภาพที่ 2.13



ภาพที่ 2.13 การตัดสัญญาณรบกวนภาพของ F. Lensen และคณะเทียบกับวิธีอื่นๆ [28]

ด้านการวัดค่าความแม่นยำนั้น F. Lensen และคณะได้ใช้ค่า l_2-difference โดยใช้ความแตกต่างระหว่างภาพที่ตัดสัญญาณรบกวนแล้วกับภาพต้นฉบับ ผลที่ได้เทียบกับอัลกอริทึมอื่นๆ ได้ตามตารางที่ 2.6

ตารางที่ 2.6 การเปรียบเทียบ l_2-difference ระหว่างอัลกอริทึมต่างๆ [28]

วิธีการ	Cross-bilateral	NL Means	Infimal convolution	Lensen et al.
l_2- difference	0.036866	0.036627	0.039305	**0.033600**

F. Lenzen และคณะได้สรุปว่าจะเห็นได้ว่าการใช้อัลกอริทึมการแปรผันรวมกับงานนี้ จะมีความแม่นยำในการกำจัดสัญญาณรบกวนบนภาพสูงสุด

นอกจากนี้ยังมีการนำวิธีการแปรผันรวมไปใช้ร่วมกับการตัดสัญญาณรบกวนภาพแบบอื่น [29] เช่นในปี ค.ศ. 2011 มีการนำวิธีการการแปรผันรวมไปใช้กับวงการแพทย์ในการตัดสัญญาณรบกวนบนภาพอัลตราซาวด์ [30] ซึ่งภาพอัลตราซาวน์นั้นประกอบด้วยสัญญาณรบกวนแบบสเปกเคิล (Speckle Noise) ซึ่งเป็นสัญญาณรบกวนแบบการคูณเป็นส่วนใหญ่ อย่างไรก็ดี ภาพอัลตรา

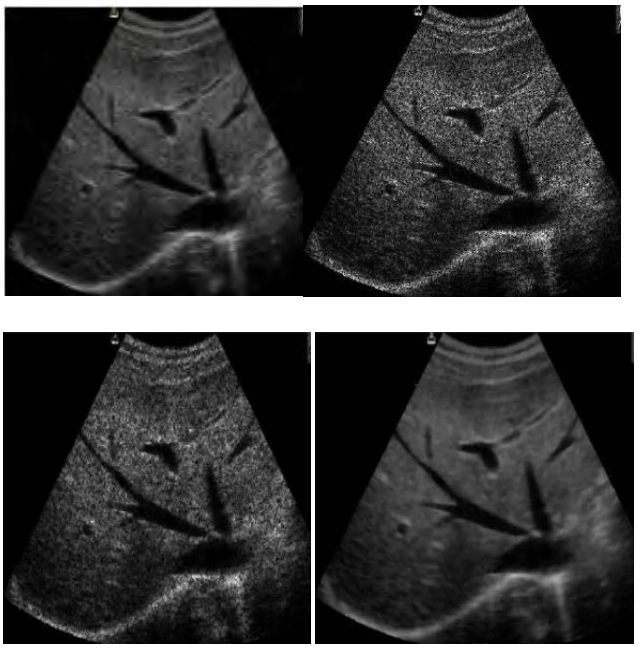
ซาวน์ก็ยังมีสัญญาณรบกวนประเภทบวกซึ่งสามารถตัดสัญญาณรบกวนได้ด้วยวิธีการการแปรผันรวม

ในงานวิจัยนี้ได้มีการรวมระหว่างสองตัวแบบในการกำจัดสัญญาณรบกวนโดยรวมระหว่างวิธีการกรองแบบเวฟเลต (Wavelet Filtering) และการแปรผันรวมสามารถเขียนโมเดลการกำจัดสัญญาณรบกวนได้ตามสมการ (2.43)

$$f = \sum_{k \in I^2} C_{LL}\psi_{0,k} + \sum_{k \in I^2} C_{LH}\psi^1_{0,k} + \sum_{k \in I^2} C_{HL}\psi^2_{0,k} + \sum_{k \in I^2} C_{HH}\psi^3_{0,k} \qquad (2.43)$$

โดยที่ k เป็นตำแหน่งพิกเซลของภาพ เมื่อที่ f เป็นภาพที่จะถูกกู้คืน C คือค่า Wavelet ที่มีการแปลงแล้ว และ ψ เป็นเวฟเลตที่ถูกสร้างขึ้นมาจากเทนเซอร์ (Tensor) แบบ 2 มิติ และมีการแบ่งภาพออกเป็น 4 ช่วงคลื่นความถี่ (band) คือ LL, LH, HL และ HH ซึ่งในความถี่ของภาพขนาด LL จะมีการใช้วิธีการแปรผันรวมเพื่อกำจัดสัญญาณรบกวน

โดย B. A. Abrahim ได้ทดลองกับภาพอัลตราซาวน์ตามภาพที่ 2.14



ภาพที่ 2.14 แสดงภาพต้นฉบับ (บนซ้าย), ภาพต้นฉบับที่ใส่สัญญาณรบกวน (บนขวา), ภาพที่ตัดสัญญาณรบกวนโดย Wavelet Filter (ล่างซ้าย), และการใช้งานร่วมกับการแปรผันรวม (ล่างขวา) [30]

ในการวัดผลเชิงปริมาณนั้นจะใช้ทั้งหมด 5 ค่า คือ ผลรวมของกำลังสองของค่าความคลาดเคลื่อน (Mean Square Error: MSE), รากที่สองของผลรวมของกำลังสองของค่าความคลาดเคลื่อน (Root Mean Square Error:RMSE), อัตราส่วนระหว่างสัญญาณภาพต่อสัญญาณรบกวน (Signal-to-Noise Ratio: SNR), อัตราส่วนระหว่างสัญญาณภาพต่อสัญญาณรบกวนสูงสุด (PSNR) และ ค่าความคลาดเคลื่อนจากการจัดระดับ (Quantization Error: Q) [31] โดยที่ค่า MSE และ RMSE ส่วน SNR นั้นเป็นการวัดความคลาดเคลื่อนจากภาพต้นฉบับ หากภาพที่ตัดสัญญาณรบกวนออกมีค่านี้น้อยแสดงว่าใกล้เคียงกับต้นฉบับ ส่วน PSNR และ Q เป็นค่าวัดความคล้ายคลึงของภาพผลลัพธ์กับภาพต้นฉบับและคุณภาพของภาพ ถ้าตัวเลขมากแสดงว่าภาพที่ได้มีคุณภาพดีและมีความคล้ายคลึงกับต้นฉบับที่ไม่มีสัญญาณรบกวน ซึ่งเมื่อทำการทดลองเปรียบเทียบกับอัลกอริทึมอื่นๆ ได้ผลตามตารางที่ 2.7

ตารางที่ 2.7 การเปรียบเทียบค่าเชิงปริมาณของภาพที่ตัดสัญญาณรบกวนของอัลกอริทึมต่างๆ [30]

Filter Type	Feature set				
	MSE	RMSE	SNR	PSNR	Q
median	112.1565	10.5904	23.6660	30.6429	0.7270
srad	29.4551	9.2014	**23.6755**	31.8640	0.7249
gf4d	30.5884	5.5307	23.6667	36.2855	**0.8681**
lsmv	67.4939	8.2155	23.6668	32.8485	0.7138
waveltc	94.5670	9.7246	22.6721	31.3837	0.7590
TV	124.2650	14.9755	23.6531	27.6335	0.7021
B. A. Abrahim	**25.2279**	**5.0227**	**23.6675**	**37.1223**	**0.8435**

จากการทดลองนำการแปรผันรวมมาใช้ร่วมกับอัลกอริทึมเดิม ทำให้ภาพที่ได้นั้นมีคุณภาพสูงขึ้นอย่างมาก (B. A. Abrahim) เมื่อเทียบกับการใช้วิธีการกรองแบบเวฟเลตอย่างเดียวและอัลกอริทึมอื่นๆที่ B. A. Abrahim ได้นำมาเปรียบเทียบ

2.4 การกำจัดสัญญาณรบกวนบนวีดีโอแบบเรียลไทม์

ในปัจจุบันยังไม่มีงานวิจัยที่ทำด้านการตัดสัญญาณรบกวนบนภาพแบบเรียลไทม์มากนัก เนื่องจากข้อจำกัดด้านประสิทธิภาพของหน่วยประมวลผลกลางและหน่วยประมวลผลกราฟิกส์ โดยทั่วไปเราสามารถจำแนกการตัดสัญญาณรบกวนบนวีดีโอได้ออกเป็น 3 ประเภทคือ

1. Spatial Video Denoising เป็นการตัดสัญญาณรบกวนบนวีดีโอเชิงพื้นที่ ซึ่งคำนวณเฟรมวีดีโอแต่ละเฟรมอย่างเป็นอิสระจากกัน

2. Temporal Video Denoising เป็นการตัดสัญญาณรบกวนบนวีดีโอโดยหาสัญญาณรบกวนแต่ละเฟรมที่ขึ้นต่อกัน และมีการตัดสัญญาณรบกวนที่อยู่ระหว่างเฟรมออกไป

3. Spatial-Temporal Video Denoising หรือเรียกว่า Spatio-temporal เป็นวิธีการตัดสัญญาณรบกวนบนวีดีโอที่ใช้ทั้ง 2 แบบที่กล่าวมาข้างต้นร่วมกัน

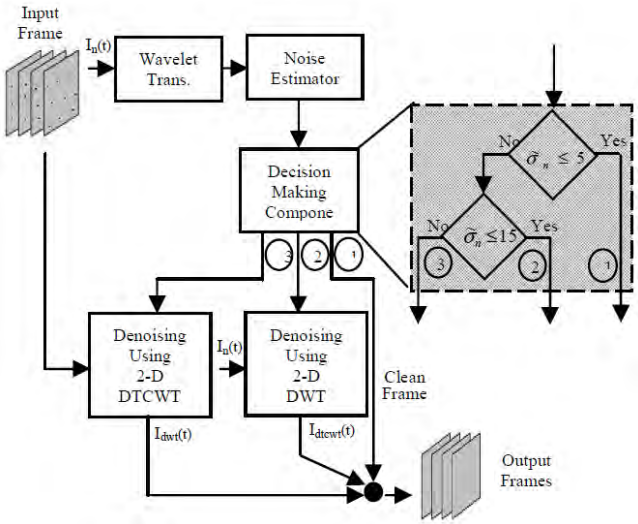
อัลกอริทึมในการตัดสัญญาณรบกวนบนวีดีโอแบบเรียลไทม์นั้นนำเสนอโดย R. O. Mahmoud เป็นการนำเสนอการตัดสัญญาณรบกวนของวีดีโอโดยใช้อัลกอริทึม 2 แบบคือการแปลงเวฟเลตแบบเต็มหน่วยสองมิติ (2D Discrete Wavelet Transform: 2D DWT) และ การแปลงเวฟเลตเชิงซ้อนแบบดูอัลทรีสองมิติ (2D Dual Tree Complex Wavelet Transform: 2D DTCWT) [32] ในการตัดสัญญาณรบกวนบนวีดีโอเชิงพื้นที่ [33]

ในงานวิจัยนี้ R. O. Mahmoud และคณะได้ให้ข้อสังเกตว่า 2D DTCWT นั้นเป็นอัลกอริทึมที่สามารถตัดสัญญาณรบกวนได้ดีกว่าแบบ 2D DWT แต่จะใช้เวลาในการทำงานประมาณ 6 เท่าของอัลกอริทึม 2D DWT และบางเฟรมนั้นใช้อัลกอริทึม 2D DWT ก็เพียงพอแล้ว ผู้วิจัยจึงได้นำเสนอตัวประมาณสัญญาณรบกวน (Noise Estimator) ซึ่งเป็นอัลกอริทึมสำหรับประมาณค่าสัญญาณรบกวนบนภาพ ซึ่งเขียนได้ตามสมการ (2.44)

$$\sigma_n = \frac{Median(|\lambda_{HH}|)}{0.6745} \qquad (2.44)$$

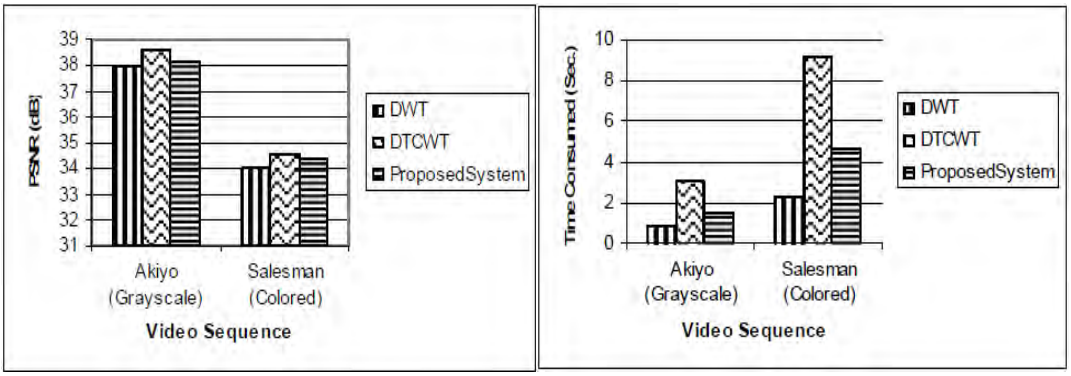
โดยที่ σ_n แทนปริมาณสัญญาณรบกวนบนภาพเฟรมที่ n มีหน่วยเป็นเดซิเบล λ_{HH} แทนสัมประสิทธิ์สัญญาณรบกวนแบบเวฟเล็ท บนช่วงความถี่สูง-สูง ซึ่งจากการใช้ตัวประมาณค่าสัญญาณรบกวนบนภาพ โดย R. O. Mahmoud และคณะแบ่งภาพออกเป็น 3 กลุ่มคือ แบบ Clean ซึ่งมีสัญญาณรบกวนน้อยกว่า 5dB และในการทำงานจะไม่ทำการตัดสัญญาณรบกวนของเฟรมออก แบบที่สองคือแบบมีสัญญาณรบกวนน้อยคือมีสัญญาณรบกวนตั้งแต่ 5-15dB จะใช้อัลกอริทึม 2D DWT และแบบที่สามคือมีสัญญาณรบกวนมาก จะใช้จะใช้อัลกอริทึม 2D DTCWT ในการตัดสัญญาณรบกวนตามลำดับ

ตัวแบบที่ R. O. Mahmoud และคณะนำเสนอนั้นสามารถเขียนได้ตามภาพที่ 2.15



ภาพที่ 2.15 โมเดลในการตัดสัญญาณรบกวนแบบเรียลไทม์ [33]

ในการทดลองนั้น R. O. Mahmoud และคณะได้ใช้ค่า PSNR ในการวัดคุณภาพของภาพแต่ละเฟรมโดยปริมาณสัญญาณรบกวนในแต่ละเฟรม ซึ่งผลที่ได้นั้นเป็นตามภาพที่ 2.16



ภาพที่ 2.16 คุณภาพเฉลี่ยของเฟรมที่ตัดสัญญาณรบกวนออกแล้ว (ซ้าย) และเวลารวมที่ใช้ในการตัดสัญญาณรบกวนบนวีดีโอ (ขวา) [33]

จะเห็นว่าการใช้โมเดลของ DWT และ DTCWT ร่วมกันโดยใช้ตัวประมาณค่าสัญญาณรบกวนนั้นจะทำให้ได้ผล PSNR ที่ใกล้เคียงกับอัลกอริทึม DTCWT และใช้เวลาในการประมวลผลที่น้อยกว่าการใช้วิธี DTCWT มาก ซึ่งเหมาะสำหรับการประมวลผลในแบบเรียลไทม์

บทที่ 3

วิธีดำเนินการวิจัย

ในวิทยานิพนธ์เล่มนี้มุ่งเน้นในการพัฒนาอัลกอริทึมเกาซ์-เซเดลให้ทำงานได้อย่างมีประสิทธิภาพสูงบนหน่วยประมวลผลกราฟิกส์ที่มีมากแกน โดยมีเป้าหมายสำคัญในการลดเวลาผสานการทำงานระหว่างเธรด (Thread Synchronization) และลดเวลาในการรอคอยเข้าทำงานของเธรด (Idle Time) อัลกอริทึมที่พัฒนาขึ้นใหม่จะถูกนำไปใช้กับงานด้านต่างๆ เช่นการแก้สมการปัวซงสองมิติด้วยวิธีการผลต่างอันตะ การตัดสัญญาณรบกวนบนภาพ และการตัดสัญญาณรบกวนบนวีดีโอ ดังจะกล่าวถึงในบทต่อๆ ไป

การดำเนินการพัฒนาอัลกอริทึมเกาซ์-เซเดล

ผู้วิจัยจะพัฒนาอัลกอริทึมเกาซ์-เซเดลเชิงขนานที่สามารถทำงานได้ดีบนหน่วยประมวลผลกราฟิกส์ โดยอัลกอริทึมตั้งอยู่บนแนวคิดของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง (SWGS) และผู้วิจัยได้ทำการออกแบบอัลกอริทึมใหม่ให้มีลักษณะดังนี้

1. สามารถลดลักษณะการทำงานแบบโพรดิวเซอร์-คอนซูเมอร์ที่ทำให้สูญเสียเวลาการผสานการทำงานระหว่างเธรดค่อนข้างมาก
2. สามารถลดเวลารอคอยเข้าทำงานของแต่ละเธรดลงในช่วงเริ่มต้นของการทำงานได้ เนื่องจากอัลกอริทึม SWGS นั้นเดิมถูกออกแบบให้ทำงานในลักษณะหลายแกน ในขณะที่หน่วยประมวลผลกราฟิกส์ จะมีการเรียกใช้งานเธรดเป็นจำนวนมากจึงส่งผลให้เธรดลำดับหลังๆ ต้องรอคอยในการเข้าทำงาน และในขณะที่รอคอย เธรดจะไม่สามารถประมวลผลได้

หลังจากได้พัฒนาอัลกอริทึมตามแนวคิดข้างต้นแล้ว ขั้นตอนต่อไปคือการนำอัลกอริทึมที่พัฒนาได้ไปทดสอบด้านเวลาและประสิทธิภาพในการทำงานกับสมการปัวซงแบบสองมิติต่อไป

การวัดผลและเปรียบเทียบกับอัลกอริทึม SWGS

จากที่ได้กล่าวถึงไว้แล้วในตอนต้น การทดสอบอัลกอริทึมที่พัฒนาขึ้นใหม่ในด้านเวลาและประสิทธิภาพจะเนินการบนระบบสมการเชิงเส้นที่เกิดจากการใช้วิธีการผลต่างอันตะกับสมการปัวซงแบบสองมิติ (2.12) ภายใต้เงื่อนไขขอบ (2.13) เมื่อ g(x, y)=0 และ

$$h = \frac{1}{(N+1)} \tag{3.1}$$

หลังจากใช้วิธีการผลต่างอันตะแบบพื้นฐานที่ใช้การประมาณค่าแบบ 5 จุดพร้อมกับแบ่ง Ω ออกเป็นเซลจำนวน N+1 เซลด้วยจุดกริด N+2 จุดกริดในแต่ละมิติ โดยที่แต่ละเซลมีขนาดเท่ากับ $h \times h$ ที่ซึ่ง

$$\frac{4(u)_{i,j} - \left((u)_{i+1,j} + (u)_{i-1,j} + (u)_{i,j+1} + (u)_{i,j-1}\right)}{h^2} = (f)_{i,j} \tag{3.2}$$

สำหรับแต่ละจุดบนกริด $(x_i, y_j) = (ih, jh)$; $i, j = 1,2,\ldots,N$ บนโดเมนที่ถูกแบ่ง $(u)_{ij}$ แทนฟังก์ชันกริด โดยที่เวกเตอร์ของตัวแปรไม่ทราบค่า $u = \left((u)_{1,1}, (u)_{1,2}, \ldots, (u)_{1,N}, (u)_{2,1}, (u)_{2,2}, \ldots, (u)_{2,N}, \ldots, (u)_{N,N}\right)^T$ และโหลดเวกเตอร์ $f = \left((f)_{1,1}, (f)_{1,2}, \ldots, (f)_{1,N}, (f)_{2,1}, (uf)_{2,2}, \ldots, (f)_{2,N}, \ldots, (f)_{N,N}\right)^T$ ดังนั้นจะได้

$$\mathbf{A}\boldsymbol{u} = \boldsymbol{f} \tag{3.3}$$

ในที่นี้ผู้วิจัยจะทดลองกับโดเมนในขนาดต่างๆ ตั้งแต่ 255×255, 511×511, 1,023×1,023, และ 2,047×2,047 โดยใช้จำนวนเธรดบนหน่วยประมวลผลกราฟิกส์ตั้งแต่ 32, 64, 128, 256 และ 512 เธรด

การประยุกต์ใช้กับแอปพลิเคชันอื่นๆ

นอกจากผู้วิจัยได้นำอัลกอริทึมที่ได้พัฒนาไปทดสอบข้างต้น ผู้วิจัยยังได้นำอัลกอริทึมที่พัฒนาได้จากข้อ 1 ไปใช้งานกับงานเชิงวิทยาศาสตร์อื่นๆ อีก 5 งาน ได้แก่ การตัดสัญญาณรบกวนประเภทบวกบนภาพ การตัดสัญญาณรบกวนบนภาพเคลื่อนไหวแบบเรียลไทม์ การตัดสัญญาณ

รบกวนบนภาพอัลตราซาวน์ ภาพถ่ายดาวเทียม และภาพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด ซึ่งแต่ละงานมีลักษณะของปัญหาที่ต่างกันออกไป

การตัดสัญญาณบนภาพถ่ายทั่วไป

โมเดลการตัดสัญญาณรบกวนแบบ ROF [4] นั้นสามารถเขียนได้ตามสมการ (2.18)

ในงานวิจัยนี้ α เป็นค่าที่ถูกเลือกจนทำให้ PSNR ระหว่างภาพที่ถูกตัดสัญญาณรบกวนด้วยโมเดล (2.18) กับภาพต้นฉบับสูงที่สุด

ตามที่ได้กล่าวไว้ในบทที่สอง การใช้แคลคูลัสของการแปรผันกับโมเดล ROF (2.18) ได้นำไปสู่สมการออยเลอร์-ลากรานจ์ดังที่ถูกกำหนดใน (2.21) หลังจากใช้วิธีการผลต่างอันตะกับโดเมนซึ่งถูกดิสครีไทซ์ดังสมการต่อไปนี้

$$\Omega_h = \left\{ x \in \Omega \mid x = (x_i, x_j)^T = (ih, jh)^T; i, j = 1, 2, \ldots N+1 \right\} \tag{3.4}$$

ภายใต้เงื่อนไขขอบที่ถูกดิสครีไทซ์

$$(u^h)_{i,1} = (u^h)_{i,2}, (u^h)_{i,n} = (u^h)_{i,n-1}, (u^h)_{1,j} = (u^h)_{2,j}, (u^h)_{n,j} = (u^h)_{n-1,j} \tag{3.5}$$

จะได้สมการออยเลอร์-ลากรานต์แบบดิสครีตต่อไปนี้

$$\frac{\alpha}{h^2}((\Sigma)_{i,j}(u^h)_{i,j} - (\overline{\Sigma})_{i,j}(u^h)_{i,j}) + (u^h)_{i,j} = (z^h)_{i,j} \tag{3.6}$$

ซึ่ง $\sum$ และ $\overline{\Sigma}$ นั้นสามารถเขียนได้ดังสมการ (3.8)

$$\begin{aligned}
&(\Sigma)_{i,j}(u^h)_{i,j} = (D_1(u^h)_{i,j} + D_2(u^h)_{i,j} + 2D_3(u^h)_{i,j}) \cdot (u)_{i,j}, \\
&(\overline{\Sigma})_{i,j}(u^h)_{i,j} = D_3(u^h)_{i,j}((u)_{i+1,j} + (u)_{i,j+1}) \\
&\qquad + D_1(u^h)_{i,j}(u)_{i-1,j} + D_2(u^h)_{i,j}(u)_{i,j-1}, \\
&D_1(u^h)_{i,j} = D(u^h)_{i-1,j} \\
&D_2(u^h)_{i,j} = D(u^h)_{i,j-1} \\
&D_3(u^h)_{i,j} = D(u^h)_{i,j}.
\end{aligned} \tag{3.7}$$

โดยที่

$$D(u^h)_{i,j} = \frac{1}{|\nabla u|_\beta} = \frac{1}{\sqrt{\left(\frac{(u)_{i+1,j} - (u)_{i,j}}{h}\right)^2 + \left(\frac{(u)_{i,j+1} - (u)_{i,j}}{h}\right)^2 + \beta}} \qquad (3.8)$$

และ $(u)_{i,j}$ แทนฟังก์ชันกริดซึ่งกำหนดค่าความเข้มของภาพที่ถูกตัดสัญญาณรบกวนในตำแหน่ง $(x_i, y_j)^T$ ในงานวิจัยนี้จะเลือกใช้ค่า β ได้ถูกกำหนดค่าเป็น 10^{-4}

หลังจากใช้วิธีการทำซ้ำแบบจุดตรึงกับ (3.7) จะได้

$$\frac{\alpha}{h^2}((\Sigma^{(\upsilon)})_{i,j}(u^{(\upsilon+1)})_{i,j} - (\overline{\Sigma}^{(\upsilon)})_{i,j}(u^{(\upsilon+1)})_{i,j}) + (u)_{i,j} = (z)_{i,j} \qquad (3.9)$$

โดยที่ค่า $\Sigma^{(\upsilon)}$ และ $\overline{\Sigma}^{(\upsilon)}$ นั้นถูกคำนวณได้จากสมการ (3.8)

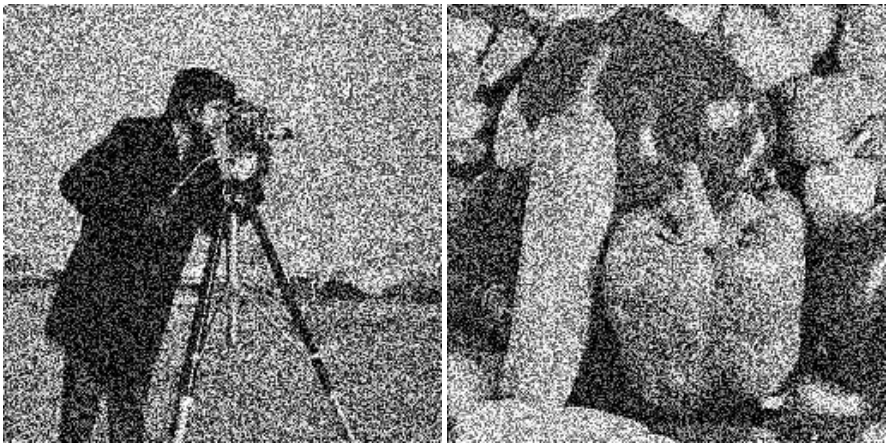
$$\mathbb{N}[\mathbf{u}^{(\upsilon)}]\mathbf{u}^{(\upsilon+1)} = \mathbf{z} \qquad (3.10)$$

โดย $\mathbb{N}[u]$ ถูกกำหนดจากเทอมทางซ้ายมือของสมการ (3.10) และ υ เป็นดัชนีของการทำซ้ำแบบจุดตรึง

สมการ (3.11) เป็นระบบสมการเชิงเส้นที่เราสามารถนำวิธีการเกาซ์-เซเดลมาเพื่อหาคำตอบเชิงตัวเลขได้ ผู้วิจัยได้ทดสอบโมเดล ด้วยภาพ Cameraman และ Pepper [34] ตามภาพที่ 3.1

ภาพที่ 3.1 : ตัวอย่างภาพที่ใช้ในการทดลอง cameraman (ซ้าย) และ pepper (ขวา)

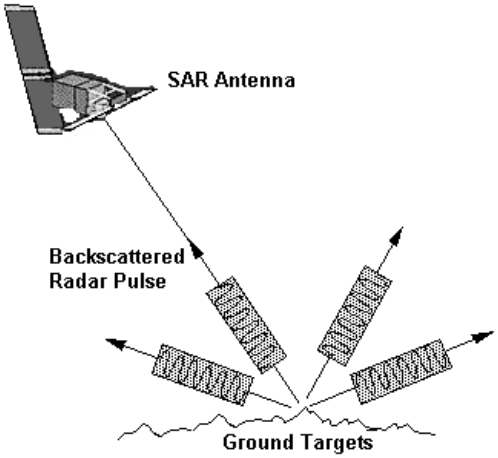
ภาพที่ได้มานั้นเป็นภาพที่มีความเข้มสีเทาและที่มีขนาดต่างๆตั้งแต่ 512×512 ถึง 1,024×1,024 2,048×2,048 และ 4,096×4,096 พิกเซล ผู้วิจัยนั้นได้สังเคราะห์สัญญาณรบกวนแบบเกาซ์และใส่ลงไปในภาพในระดับต่างๆ กันตั้งแต่ 10 – 50% ดังตัวอย่างในภาพที่ 3.2



ภาพที่ 3.2 : ตัวอย่างภาพที่ใช้ในการทดลองที่มีการใส่สัญญาณรบกวนแล้ว 50%

การตัดสัญญาณรบกวนบนภาพถ่ายดาวเทียมขนาดใหญ่

ภาพถ่ายดาวเทียมนั้นเป็นภาพที่เกิดจากดาวเทียมถ่ายภาพลงมายังโลก ประกอบด้วยสัญญาณรบกวนหลายประเภท ซึ่งการจับสัญญาณภาพของดาวเทียมนั้นเกิดจากเรดาร์ช่องเปิดสังเคราะห์ (Synthetic Aperture Radars: SARs) ยิงคลื่นไปกระทบวัตถุและมีเซ็นเซอร์วัดการสะท้อนกลับซึ่งดาวเทียมที่ยิงคลื่นนั้นมีตัวเซ็นเซอร์รับคลื่นที่สะท้อนกลับในแนวยาว และคลื่นที่สะท้อนกลับมาจะมีความถี่ที่เปลี่ยนไปตามการเคลื่อนที่ของตัวปล่อยสัญญาณและตัวรับสัญญาณที่จะนำความถี่ที่สะท้อนกลับมาไปเปรียบเทียบและสร้างเป็นภาพสองมิติต่อไป การทำงานแสดงเป็นภาพได้ดังภาพที่ 3.3



ภาพที่ 3.3 แผนผังการทำงานของดาวเทียมที่มีตัวรับภาพแบบ SAR [35]

การที่ดาวเทียมถ่ายภาพโดยวิธีนี้ทำให้เกิดสัญญาณรบกวนทั้งสองประเภทคือสัญญาณรบกวนประเภทบวกจากเซ็นเซอร์รับภาพ และสัญญาณรบกวนประเภทคูณจากการเปลี่ยนความถี่ของสัญญาณ

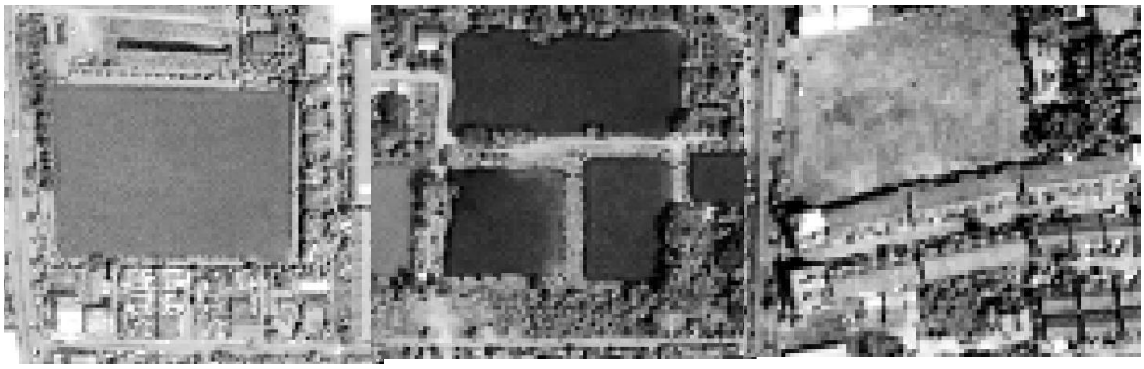
อย่างไรก็ตามการตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียมนั้น ต้องใช้แนวทางที่แตกต่างจากการตัดสัญญาณรบกวนจากภาพประเภทอื่นเนื่องจากสาเหตุหลายประการดังนี้

1. ภาพที่ได้จากดาวเทียมมีขนาดใหญ่มากจนไม่สามารถบรรจุลงในหน่วยความจำหลัก และหน่วยความจำของหน่วยประมวลผลกราฟิกส์ของเครื่องคอมพิวเตอร์ทั่วไปได้ทั้งภาพ จึงต้องมีการจัดการหน่วยความจำที่ดี
2. เนื่องจากภาพถ่ายดาวเทียมมีขนาดใหญ่ และใช้พื้นที่ดิสก์ในการเก็บมาก การดึงข้อมูลจากดิสก์นั้นจึงใช้เวลานานกว่าการดึงภาพทั่วไปมาก ซึ่งหากใช้วิธีการดึงภาพแบบปกติ จะทำให้หน่วยประมวลผลหลักต้องรอคอยการขนถ่ายข้อมูลทำให้ทำงานได้อย่างไม่มีประสิทธิภาพ
3. การใช้หน่วยประมวลผลกราฟิกส์ต้องมีการขนถ่ายข้อมูลระหว่างหน่วยความจำหลักไปยังหน่วยความจำของหน่วยประมวลผลกราฟิกส์ ซึ่งสำหรับภาพขนาดใหญ่จะต้องใช้เวลามาก

ด้วยสาเหตุที่กล่าวมานี้ การตัดสัญญาณรบกวนบนภาพถ่ายดาวเทียมขนาดใหญ่จึงต้องคำนึงถึงการจัดการหน่วยความจำที่เหมาะสมทั้งในหน่วยความจำหลักและในหน่วยความจำของ

หน่วยประมวลผลกราฟิกส์รวมทั้งยังต้องคำนึงถึงการทำงานผสานระหว่างหน่วยประมวลผลหลักและหน่วยประมวลผลกราฟิกส์เป็นอย่างดี

ในวิทยานิพนธ์เล่มนี้ ภาพที่นำมาใช้ในการทดลองนี้ได้จากดาวเทียม IKONOS ซึ่งบางส่วนของภาพใหญ่สามารถแสดงได้ในภาพที่ 3.4



(ก) (ข) (ค)

ภาพที่ 3.4 ตัวอย่างภาพถ่ายดาวเทียมที่นำมาใช้ในการทดลอง

การตัดสัญญาณรบกวนภาพเคลื่อนไหวแบบเรียลไทม์

เมื่อได้อัลกอริทึมในการตัดสัญญาณรบกวนบนภาพนิ่งที่เหมาะสมและมีประสิทธิภาพจากหัวข้อก่อนหน้านี้แล้ว ผู้วิจัยได้ประยุกต์ใช้อัลกอริทึมดังกล่าวในการตัดสัญญาณรบกวนบนภาพเคลื่อนไหว โดยผู้วิจัยจะพัฒนาขั้นตอนในการจัดการเธรดที่เหมาะสมกับการตัดสัญญาณรบกวนบนวีดีโอแบบเรียลไทม์

วีดีโอที่ผู้วิจัยนำมาใช้ทดลองนั้นเป็นวิดีโอสีที่ใช้ทดสอบในงานวิจัยด้านวีดีโออย่างแพร่หลายในรูปแบบ RGB ซึ่งผู้วิจัยเลือกวีดีโอที่มีชื่อว่า Miss America [34] อัตราส่วน 4:3 มีจำนวนเฟรมทั้งสิ้น 150 เฟรมโดยเล่นที่ 15 เฟรมต่อวินาที และ Harbour [34] มีทั้งสิ้น 600 เฟรมโดยเล่นที่ 30 เฟรมต่อวินาที ซึ่งเฟรมแรกของวีดีโอแสดงในภาพที่ 3.5


ภาพที่ 3.5 : ตัวอย่างวิดีโอที่ใช้ในการทดลอง harbor (ซ้าย) และ miss America (ขวา)

ผู้วิจัยได้ใส่สัญญาณรบกวนแบบเกาซ์จำนวน 25-50% ลงไปทุกเฟรมและทุกช่องสี ผู้วิจัยทำการตัดสัญญาณรบกวนโดยใช้ขั้นตอนที่พัฒนาขึ้นเพื่อจัดการเธรดบนหน่วยประมวลผลกราฟิกส์และหน่วยประมวลผลกลางอย่างมีประสิทธิภาพ หลังจากนั้นจึงวัดประสิทธิภาพในการกำจัดสัญญาณรบกวนบนวีดีโอ

การตัดสัญญาณรบกวนบนภาพอัลตราซาวน์

ภาพอัลตราซาวน์นั้นเกิดจากการใช้คลื่นเสียงความถี่สูงเพื่อให้ทะลุพื้นผิววัตถุชั้นนอกไปยังวัตถุที่สนใจ และสร้างภาพที่เกิดขึ้นจากการสะท้อนของคลื่นเสียง ภาพอัลตราซาวน์มีการใช้งานอย่างกว้างขวางโดยเฉพาะวงการแพทย์และชีววิทยาเพื่อศึกษาวัตถุที่อยู่ภายในสิ่งที่ถูกปิดกั้นอยู่ สัญญาณรบกวนบนภาพอัลตราซาวน์นั้นส่วนใหญ่เป็นสัญญาณรบกวนประเภทคูณ แต่มีสัญญาณรบกวนประเภทบวกที่เกิดจากเซ็นเซอร์รับภาพด้วย ซึ่งในงานวิจัยนี้ ผู้วิจัยได้ใช้โมเดล CC ของ [38] ในการกำหนดสัญญาณรบกวนบนภาพอัลตราซาวน์

$$z = u + k_0\eta + k_1\sqrt{u}\eta \qquad (3.11)$$

ในที่นี้ z เป็นภาพที่มีสัญญาณรบกวน u เป็นภาพที่ปราศจากสัญญาณรบกวน k_0 และ k_1 เป็นพารามิเตอร์ควบคุมสัญญาณรบกวนประเภทบวกและสัญญาณรบกวนประเภทคูณ ตามลำดับ และ η เป็นสัญญาณรบกวนที่มีการแจกแจงแบบเกาซ์ซึ่งมีค่าเฉลี่ยเป็น 0

ในการตัดสัญญาณรบกวนของภาพถ่ายประเภทนี้สามารถทำได้โดยใช้โมเดลการตัดสัญญาณรบกวนของ N. Chumchob และ C. Chantrapornchai ใน [38] ซึ่งเรียกว่าโมเดล CC สามารถเขียนได้ในรูปสมการ

$$\min_{u}\{J_{\alpha_1,\alpha_2}[u] = D^{CC}(u) + TV(u)\} \tag{3.12}$$

โดยที่ $D^{CC}(u)$ เป็นเทอมวัดความผิดปกติของข้อมูล ถูกกำหนดโดย

$$D^{CC}(u) = \frac{\alpha_1}{2}\int_{\Omega}(u-z)^2 d\Omega + \alpha_2\int_{\Omega}\frac{(z-u)^2}{u}d\Omega \tag{3.13}$$

และ $TV(u)$ ถูกกำหนดโดย (2.20)

$$TV(u) = \int_{\Omega}|\nabla u|_{\beta}\, d\Omega = \int_{\Omega}\sqrt{u_x^2 + u_y^2 + \beta}\, d\Omega, \beta > 0 \tag{3.14}$$

หลังจากใช้แคลคูลัสของการแปรผัน จะได้สมการออยเลอร์-ลากรานจ์ต่อไปนี้

$$\underbrace{\overbrace{-\nabla\bullet\frac{\nabla u}{|\nabla u|_{\beta}}}^{-\mathrm{K}(u)} + \alpha_1(u-z) + \alpha_2\left(1-\frac{z^2}{u^2}\right)}_{\mathbb{N}(u)} = 0 \tag{3.15}$$

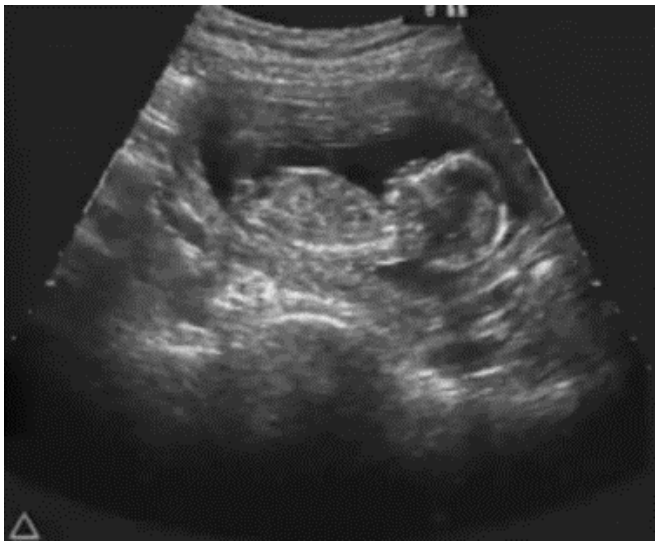
ภายใต้เงื่อนไขค่าขอบ

$$\frac{\partial u}{\partial \mathrm{n}} = 0 \text{ บน } \partial\Omega \tag{3.16}$$

เมื่อใช้วิธีการของผลต่างอันตะบน Ω_h ที่ถูกกำหนดโดยสมการ (3.6) จะได้

$$\mathrm{K}^h(u^h)_{i,j} = \left[\frac{\delta_x^-}{h_x}\left(\frac{D(u^h)_{i,j}\delta_x^+(u^h)_{i,j}}{h_x}\right) + \frac{\delta_y^-}{h_y}\left(\frac{D(u^h)_{i,j}\delta_y^+(u^h)_{i,j}}{h_y}\right)\right] \quad . \tag{3.17}$$

ในงานวิจัยนี้จะใช้ภาพเคลื่อนไหวอัลตราซาวน์ขนาด 400×312 พิกเซล ภาพเคลื่อนไหวต้นฉบับมีอัตราการเปลี่ยนเฟรมอยู่ที่ 30 เฟรมต่อวินาที ภาพที่ 3.6 แสดงเฟรมที่ 300 ของภาพต้นฉบับ



ภาพที่ 3.6 ตัวอย่างภาพเฟรมที่ 300 ของภาพเคลื่อนไหวต้นฉบับ

โดยทั่วไปแล้วการตัดสัญญาณรบกวนในภาพเคลื่อนไหวจากอุปกรณ์ทางการแพทย์มีสิ่งที่ต้องคำนึงถึง 3 ประการคือ

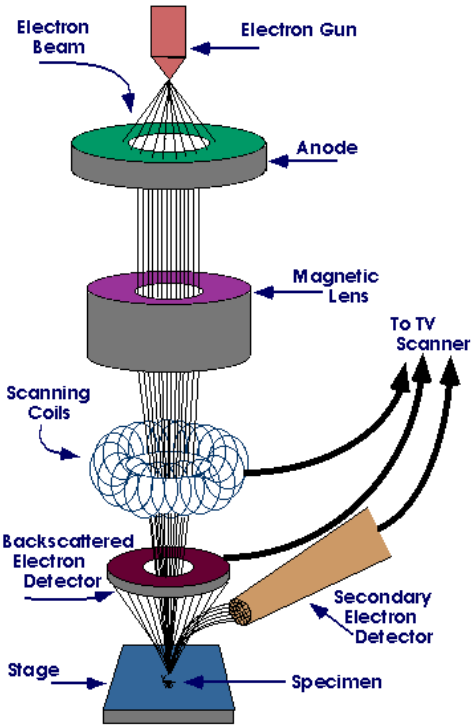
1. อัตราเฟรม คือปริมาณภาพที่ถูกตัดสัญญาณรบกวนได้ใน 1 วินาที โดยในงานวิจัย R. Paul และ M. Mayer ได้สรุปว่าภาพเคลื่อนไหวที่ดีและไม่ทำให้รู้สึกถึงการหน่วงของภาพจะอยู่ที่ 12 เฟรมต่อวินาทีขึ้นไป [7]

2. ลำดับเฟรม เมื่อทำการประมวลผลเชิงขนาน ลำดับเฟรมนั้นต้องคงเดิมเมื่อมีการนำแต่ละเฟรมมาประกอบกันเป็นภาพเคลื่อนไหวอีกครั้ง

3. ความราบรื่น ที่ได้จากการตัดสัญญาณรบกวนนั้นต้องคงที่ตลอดลำดับการแสดงของภาพเคลื่อนไหว

การตัดสัญญาณรบกวนภาพที่ได้จากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด

กลไกของกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้นมีลักษณะดังภาพที่ 3.7



ภาพที่ 3.7 กลไกของกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด [14]

จากภาพที่ 3.7 นั้น จะเริ่มจากปืนยิงอิเล็กตรอน (Electron Gun) ยิงลำอิเล็กตรอนแบบขนานเข้าสู่ตัวกล้อง ลำอิเล็กตรอนนั้นจะผ่านเลนส์ที่ทำจากแม่เหล็กเพื่อรวมลำอิเล็กตรอนให้มีขนาดเล็กมากและยิงไปยังวัตถุที่ต้องการซึ่งอยู่ในห้องเล็กๆ เมื่อลำอิเล็กตรอนกระทบวัตถุที่ต้องการศึกษาจะปลดปล่อยอิเล็กตรอนและรังสีเอ็กซ์ โดยมีตัวตรวจจับคอยรับรังสีเอ็กซ์และอิเล็กตรอนที่ปลดปล่อยออกมาในระหว่างที่มีการยิงอิเล็กตรอน การทำงานเช่นนี้จึงเป็นในลักษณะสหนัย (Coherence) ผู้วิจัยตั้งสมมุติฐานไว้ว่าสัญญาณรบกวนบนภาพนั้นจะมีทั้งประเภทบวกคือขั้นตอนการบันทึกภาพและประเภทคูณเกิดจากขั้นตอนการรับอิเล็กตรอนและรังสีเอ็กซ์ที่ปลดปล่อยจากวัตถุที่ต้องการศึกษา

ผู้วิจัยใช้โมเดลของการตัดสัญญาณรบกวนเช่นเดียวกับภาพถ่ายดาวเทียมได้แก่สมการ (3.19)

อย่างไรก็ตามภาพจุลชีพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้นมีลักษณะพิเศษที่แตกต่างออกไปจากภาพถ่ายดาวเทียมและภาพถ่ายประเภทอื่นๆ ดังนี้

1. ภาพถ่ายจากกล้องจุลทรรศน์อิเล็กตรอนนั้นมีขนาดใหญ่ แต่เล็กกว่าภาพถ่ายดาวเทียมทั่วไปในหัวข้อ “การตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียม” ดังนั้นภาพถ่ายประเภทนี้สามารถบรรจุลงในหน่วยความจำหลักของเครื่องคอมพิวเตอร์ในปัจจุบันได้ อย่างไรก็ตามเวลาที่ใช้ในการตัดสัญญาณรบกวนนั้นยังใช้เวลามาก

2. นักวิจัยในสาขาชีววิทยา และจุลชีววิทยานั้นมักจะสนใจตัวจุลชีพเป็นหลักและไม่สนใจพื้นหลัง ซึ่งนักวิจัยในด้านนี้มักใช้โลหะหนักเช่นทองคำ แพลตินัม รวมถึงแกรไฟต์ในการย้อมจุลชีพเพื่อเพิ่มประสิทธิภาพในการสะท้อนอิเล็กตรอนทำให้ภาพในส่วนตัวจุลชีพนั้นดูสว่างกว่าพื้นหลังมาก

3. ในภาพหลายๆ ภาพที่ได้จากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้น ตัวจุลชีพมักอยู่กระจัดกระจายและอยู่บนภาพแบบหลวมๆ

จากลักษณะของปัญหานี้ ผู้วิจัยจึงได้ออกแบบวิธีการประยุกต์ใช้อัลกอริทึมเกาซ์-เซเดลแบบขนานเพื่อตัดสัญญาณรบกวนของภาพประเภทนี้โดยเฉพาะ ซึ่งจะกล่าวต่อไปในบทที่ 6

การตรวจสอบความถูกต้องและการประเมินผล

ผู้วิจัยจะตรวจสอบความถูกต้องอัลกอริทึมที่พัฒนาและประเมินผลการนำอัลกอริทึมไปใช้ด้วยการทดสอบสามอย่างด้วยกันดังต่อไปนี้

การทดสอบความถูกต้องด้วยสมการปัวซองแบบสองมิติ

ในการตรวจสอบความถูกต้อง จะใช้เศษตกค้าง (Residual) และค่าความคลาดเคลื่อนแม่นตรง (Exact Error) ซึ่งในการแก้ระบบสมการ (3.3) นี้จะทดลองกับโดเมนที่มีการแบ่งขนาด 255×255, 511×511, 1023×1023 และ 2047×2047 ซึ่งสมาชิกของเวกเตอร์ในเทอมทางขวามือของสมการ (3.3) จะคำนวณได้ตามสมการ (3.18)

$$f(x,y) = 2((1-6x)^2y^2(1-x)^2) + ((1-6y)^2x^2(1-x)^2) \tag{3.18}$$

และมีผลลัพธ์แม่นตรง (Exact Solution) ตามสมการ (3.21)

$$u(x,y) = (x^2 - x^4)(y^4 - y^2) \tag{3.19}$$

ในการตรวจหาความคลาดเคลื่อนนั้นจะใช้ค่าความคลาดเคลื่อนแม่นตรงซึ่งได้จาก l_2-norm ของส่วนต่างของค่าที่คำนวณได้จากสมการที่ (3.20) และค่าผลลัพธ์ที่คำนวณได้เมื่อสิ้นสุดรอบของการทำซ้ำ

ส่วนค่าคลาดเคลื่อนจากเศษตกค้างหาได้โดยใช้ค่า l_2-norm ของเศษตกค้าง ε_{l_2} ซึ่งสามารถเขียนได้ตามสมการ (3.22)

$$\varepsilon_{l_2} = \| f - Au \|_{l_2} \tag{3.20}$$

ในการทดลองจะคำนวณค่า l_2-norm ของเศษตกค้าง (ε_{l_2}) เมื่อสิ้นสุดทุกรอบของการทำซ้ำ ซึ่งในการทดลองนั้นจะใช้เศษตกค้างเป็นหลัก โดยโปรแกรมจะหยุดการทำงานเมื่อค่า ε_{l_2} มีค่าน้อยกว่า 10^{-4}

การประเมินผลด้านประสิทธิภาพของอัลกอริทึม

ในการประเมินผลประสิทธิภาพของอัลกอริทึมนั้นทุกงานประยุกต์จะวัดจากเวลาที่ใช้ในการประมวลผลโดยเทียบกับอัลกอริทึมที่ทำงานบนหน่วยประมวลผลกลางและอัลกอริทึมเกาซ์-เซเดลที่พัฒนาได้ นอกจากนี้ยังใช้ค่า Speedup ซึ่งเขียนได้ตามสมการ (3.21)

$$speedup = \frac{t_{old}}{t_{new}} \tag{3.21}$$

โดยที่ค่า t_{old} แทนเวลารวมที่ใช้ในอัลกอริทึมแบบเดิม และ t_{new} แทนเวลารวมที่อัลกอริทึมแบบใหม่ใช้ประมวลผล

นอกจากนี้ในการประเมินผลด้านการกำจัดสัญญาณรบกวนบนวีดีโอ ผู้วิจัยได้เลือกใช้ตัววัดได้แก่จำนวนเฟรมต่อวินาที (Frame-Per-Second: FPS) ในการวัดประสิทธิภาพของอัลกอริทึม

การประเมินผลคุณภาพของภาพ

ในการตรวจสอบคุณภาพในการตัดสัญญาณรบกวนบนภาพนั้นจะใช้ 2 วิธี ได้แก่ (1) การวัดค่า PSNR ระหว่างภาพที่ถูกตัดสัญญาณรบกวนและภาพต้นฉบับและ (2) การวัดการลู่เข้าของคำตอบ ผู้วิจัยวัดเศษตกค้างในลักษณะเดียวกับที่ใช้ตรวจสอบเศษตกค้างของการแก้สมการปัวซง สามารถหาเศษตกค้าง (ε) ได้จากสมการ (3.22)

$$\varepsilon = \left|\left|\boldsymbol{z} - \mathbb{N}(\boldsymbol{u})\boldsymbol{u}\right|\right|_{l_2} \tag{3.22}$$

เครื่องมือในการวิจัย

เครื่องมือที่ใช้ในการวิจัยแบ่งออกเป็นสองด้านคือด้านฮาร์ดแวร์ และด้านซอฟต์แวร์

ด้านฮาร์ดแวร์ มีลักษณะดังต่อไปนี้

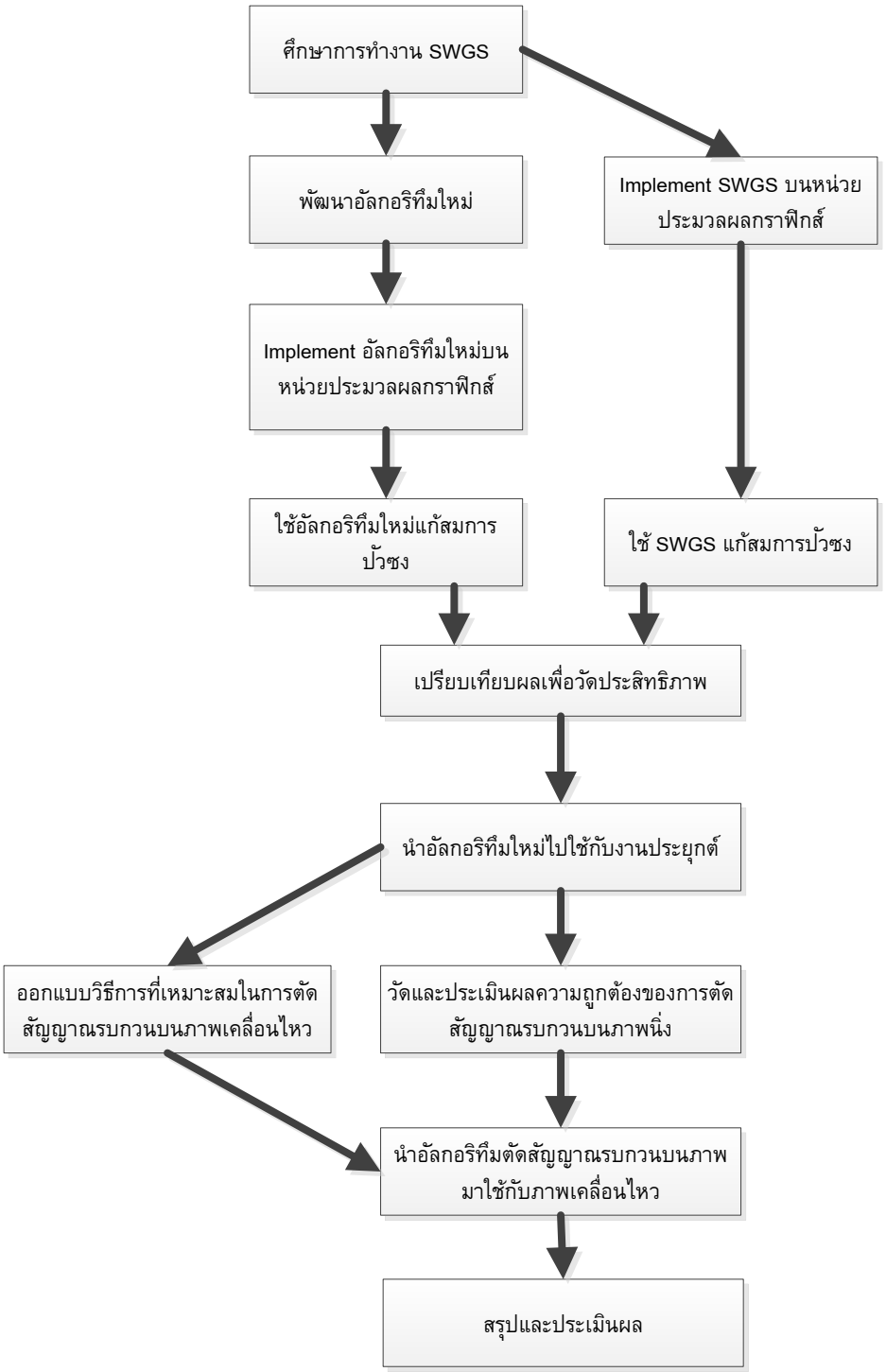
1. เครื่องคอมพิวเตอร์หน่วยประมวลผล Intel® Core 2 Duo E7400 ความเร็ว 2.4 GHz
2. หน่วยประมวลผลกราฟิกส์ NVIDIA® GTX-480 มีหน่วยความจำบนการ์ด 1.5 กิกะไบต์
3. หน่วยความจำหลัก 4 กิกะไบต์

ด้านซอฟต์แวร์

1. ระบบปฏิบัติการ Fedora Linux 14 แบบ 64 บิต
2. CUDA SDK เวอร์ชัน 4.0
3. OpenCV เวอร์ชัน 2.2
4. คอมไพล์เลอร์ GNU GCC เวอร์ชัน 4.4.0

สำหรับขั้นตอนในการวิจัย สามารถสรุปออกมาเป็นแผนผังได้ดังภาพที่ 3.8

แผนผังขั้นตอนในการวิจัย



ภาพที่ 3.8 แผนผังขั้นตอนในการวิจัย

บทที่ 4

การพัฒนาอัลกอริทึมเกาซ์-เซเดลเชิงขนาน

ในบทนี้ผู้วิจัยได้พัฒนาอัลกอริทึมเกาซ์-เซเดลเชิงขนานโดยพัฒนาจากอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง ทางผู้วิจัยมุ่งแก้ปัญหาในเรื่องการผสานการทำงานระหว่างเธรดในแต่ละรอบของการทำงานรวมทั้งลดระยะเวลารอคอยของเธรดที่จะเข้าทำงานบนหน่วยประมวลผลแบบหลายแกนประมวลผลและหน่วยประมวลผลกราฟิกส์ โดยอัลกอริทึมที่พัฒนาได้ตามลำดับได้แก่อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยและอัลกอริทึมเกาซ์-เซเดลแบบสถานะ

อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอย

แนวคิดการพัฒนา

จากปัญหาที่ผู้วิจัยพบในอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทางคือมีการผสานการทำงานระหว่างรอบที่มาก และการทำงานกับเธรดจำนวนมากจะทำให้เธรดต่างๆ รอคอยเข้าทำงานเป็นเวลานาน ผู้วิจัยจึงได้คิดเริ่มแก้ปัญหาโดยการลดปริมาณการผสานการทำงาน ซึ่งในแบบเก่าที่ผู้พัฒนาอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างที่ได้นำเสนอนั้นจะใช้การผสานการทำงานโดยอาศัยหลักแถวรอแบบวงแหวน ซึ่งทำงานในแบบโพรดิวเซอร์-คอนซูเมอร์โดยผู้บริโภคจะรอรับผลลัพท์จากผู้ผลิต วิธีนี้มีการส่งข้อมูลหลายทอด

ผู้วิจัยจึงมีแนวคิดแบ่งการทำงานออกเป็นรอบย่อยๆ ให้แต่ละหน่วยประมวลผลสามารถคำนวณได้เองว่าในแต่รอบย่อยๆ ควรจะทำงานในตำแหน่งใดโดยอาศัยฟังก์ชันการส่ง (Mapping Function) โดยมีสองฟังก์ชันคือ ฟังก์ชันสำหรับคำนวณจุดเริ่ม และฟังก์ชันสำหรับคำนวณจุดปลายในการคำนวณ เขียนได้ดังสมการ (4.1) - (4.3)

$$P_{t,n} = S(n-1) - (S + tW) \tag{4.1}$$

$$L_{t,n} = P_{t,n} + W \tag{4.2}$$

$$F_{t,n} = \begin{cases} P_{t,n}, L_{t,n} \le 0 \\ 0, L_{t,n} > 0 \end{cases} \quad (4.3)$$

โดยที่ F เป็นฟังก์ชันหาแถวที่เริ่มทำการประมวลผลในรอบย่อยนั้น ส่วน L เป็นฟังก์ชันหาแถวสุดท้ายที่จะต้องประมวลผล, n คือหมายเลขรอบย่อย, t คือหมายเลขกำกับเธรด และ S กับ W คือขนาดของบล็อกและขนาดของบานหน้าต่างตามลำดับ ซึ่งวิธีการที่พัฒนาได้สามารถเขียนเป็นโค้ดเทียม (Pseudocode) ได้ดังนี้

อัลกอริทึม 1 : เกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอย (QL-SWGS)

Require: *u, MAXGS, maxrow, maxstep, windowsize, blocksize, total_thread*

1: *tid* = *getThreadID*()
2: **while** *lastthreadstartrow* < *maxrow* and *k* < *MAXGS* **do**
3: **for** *substep* = 1 to *maxstep* **do**
4: *startrow* = (*blocksize* × (*substep* − 1)) − (*blocksize* + *tid* × *windowsize*)
5: *endrow* = *startrow* + *windowsize*
6: **for** *i* = *startrow* to *endrow* **do**
7: **for** *j* = 0 to *maxcol* **do**
8: **if** $i \geq 0$ **then**
9: compute Equation (3.3) for $(u)_{i,j}$
10: **end if**
11: **if** (*tid* == *total_thread*-1) **then** *lastthreadstartrow* = *startrow*
12: syncthread()
13: **end for**
14: syncthread()
15: **end for**
16: **end for**
17: *k* = *k* + 1
18: **end while**
19: **return** *u*

โดยที่ u เป็นโดเมนของปัญหา *getThreadID*() เป็นฟังก์ชันที่คืนค่าหมายเลขเธรดปัจจุบัน *MAXGS* เป็นพารามิเตอร์กำหนดจำนวนรอบสูงสุดของการทำซ้ำ *blocksize* และ *windowsize* คือขนาดบล็อกและบานหน้าต่าง SWGS ตามลำดับ, *lastthreadstartrow* คือตำแหน่งที่เริ่มทำงานของเธรดสุดท้าย

substep คือหมายเลขรอบย่อยของการทำงาน, *maxrow* คือหมายเลขแถวสุดท้ายของโดเมนปัญหา, total_thread คือจำนวนเธรดทั้งหมดที่ใช้ และ syncthread() เป็นฟังก์ชันผสานการทำงานของทุกเธรดซึ่งเมื่อเธรดพบคำสั่งนี้จะหยุดรอจนกระทั่งทุกเธรดหยุดรอที่คำสั่งนี้จึงเริ่มทำงานต่อ

ผลการพัฒนา

ผู้วิจัยได้ทดลองนำอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยนำมาใช้แก้ระบบสมการเชิงเส้นที่เกิดจากใช้วิธีการผลต่างอันตะกับสมการปัวซงแบบสองมิติดังที่ได้กล่าวในบทที่สามบนหน่วยประมวลผลกราฟิกส์จำนวน 2,000 รอบการทำซ้ำ ได้ผลตามตารางที่ 4.1

ตารางที่ 4.1 การเปรียบเทียบระหว่างเวลาที่ใช้ระหว่างอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง (SWGS) เดิม และอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอย (QL-SWGS) ที่ได้พัฒนาใหม่ซึ่งถูกทดสอบบนโดเมนปัญหาที่ต่างกัน

ขนาดปัญหา	เวลาที่ใช้ (มิลลิวินาที)		เวลาที่ลดลง (%)
	SWGS	QL-SWGS	
128 × 128	3,758	946	74.83
256 × 256	14,424	2,893	79.94
512 × 512	57,368	14,348	74.99
1024 × 1024	228,857	56,140	75.47
2048 × 2048	1,050,925	206,768	80.33

ตารางที่ 4.1 แสดงให้เห็นว่าการใช้ QL-SWGS บนโดเมนขนาด 2048×2048 สามารถลดเวลาประมวลผลโดยรวมได้ถึง 80.33% ซึ่งเป็นความสำเร็จอย่างมีนัยสำคัญของการใช้แนวคิดการลดเวลาในการผสานการทำงานในแต่ละขั้นตอนของการทำงานให้ได้มากที่สุด

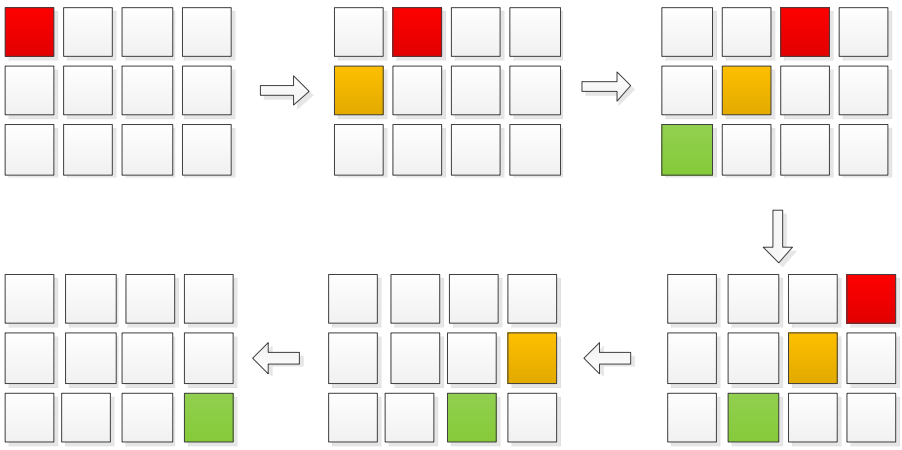
ปัญหาที่พบ

ถึงแม้ว่าอัลกอริทึมที่พัฒนาขึ้นใหม่สามารถลดเวลาในการผสานการทำงานภายในรอบการทำงาน และระหว่างรอบการทำงานได้อย่างมีนัยสำคัญ อย่างไรก็ตามการทำงานของอัลกอริทึมนี้บนหน่วยประมวลผลกราฟิกส์ที่มีเธรดเป็นจำนวนมากยังพบอุปสรรคจากเรื่องการรอคอยการเข้าทำงานของเธรดที่ว่างงาน ด้วยเหตุนี้ผู้วิจัยจึงได้พัฒนาอัลกอริทึมใหม่ที่เรียกว่าอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทาง (TSWGS) ซึ่งจะกล่าวในหัวข้อถัดไป

อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทาง

แนวคิดการพัฒนา

การทำงานของ QL-SWGS เกิดปัญหาเธรดว่างงาน (Idle Threads) เป็นจำนวนมาก เนื่องจากมีการแบ่งการประมวลผลตามแถว (จากบนลงล่าง) อย่างเดียว ผู้วิจัยได้มีแนวคิดในการแก้ปัญหาด้วยการประมวลผลแบบขนานทั้งแถวและคอลัมน์พร้อมกันภายในรอบย่อย ซึ่งขั้นตอนการประมวลผลในรอบย่อย สามารถแสดงได้ตามภาพที่ 4.1



ภาพที่ 4.1 การทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทางภายในรอบย่อย

จากภาพที่ 4.1 จะเห็นได้ว่าในแต่ละขั้นตอนย่อยๆ ของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างนั้นจะแบ่งออกเป็นขั้นตอนย่อยๆ ลงไปอีก โดยในแต่ละแถวจะใช้เธรดในการทำงาน 1 เธรด และมีการทำงานจากซ้ายไปขวาดังภาพ ทำให้ในแต่ละแถวของปัญหาสามารถนำเธรดเข้าไปทำงานได้รวดเร็วขึ้นและจะลดจำนวนเธรดที่มีการรอคอยการทำงานได้อย่างมาก

เมื่อสิ้นสุดกระบวนการในแต่ละขั้นตอนนั้น ผู้วิจัยเลือกที่จะใช้วิธีการผสานการทำงานเธรดเพื่อควบคุมการทำงานในแต่ละขั้นตอนย่อย ซึ่งในการพัฒนาอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสามารถเขียนเป็นโค้ดเทียมได้ดังนี้

อัลกอริทึม 2 : **เกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทาง (TSWGS)**

Require: *u, MAXGS, maxrow, maxstep, windowsize, blocksize, maxcol*

1: *tid* = *getThreadID*()

2: **while** *lastthreadstartrow* < *maxrow* **and** *k* < *MAXGS* **do**

3: **for** *substep* = 1 to *maxstep* **do**

4: //get position for thread

5: *window_id* = *floor*(*tid / windowsize*)

6: *window_thread_id* = (*tid* mod *windowsize*)

7: *window_currentrow* = (*blocksize* × (*substep*−1))−(*blocksize* + *tid* × *windowsize*)

8: *i* = *window_currentrow* + *window_thread_id*

9: **for** *j* = 0 **and** *window_thread_id* < *maxcol* **do**

10: **if** (*tid* == *total_thread*-1) **then** *lastthreadstartrow* = *window_currentrow*+*window_thread_id*

11: **if** $i \geq 0$ **then**

12: compute Equation (3.3) for $(u)_{i,j}$

13: **end if**

14: syncthread()

15: **end for**

16: syncthread()

17: **end for**

18: syncthread()

19: **end while**

20: **return** *u*

จากโค้ดเทียมข้างต้น ตัวแปรที่เพิ่มขึ้นมาคือ *maxcol* ซึ่งใช้เก็บหมายเลขคอลัมน์สุดท้ายของโดเมน, *window_currentrow* คือแถวของเธรดที่กำลังทำงานอยู่ และ *window_thread_id* คือลำดับของเธรดภายในบานหน้าต่าง

ผลการพัฒนา

ผู้วิจัยได้ดำเนินการทดลองโดยใช้สมการปัวซงขนาด 2 มิติที่มีขนาดของปัญหาเป็น 63×63, 127×127, 255×255 และ 511×511 ตามลำดับ โดยได้ทำการทดลองบนเครื่อง CPU Intel Core 2 Duo

ที่สัญญาณนาฬิกา 2.4 GHz หน่วยความจำ 2GB และมีการติดตั้งการ์ดประมวลผลกราฟิกส์ NVIDIA GeForce GTX-480 ซึ่งมีหน่วยประมวลผลจำนวน 480 หน่วย และมีหน่วยความจำขนาด 1.5GB โดยการทดลองนั้นได้ทำบนระบบปฏิบัติการลินุกซ์ Fedora 14 และใช้ CUDA SDK 3.1 ในการคอมไพล์และรันโปรแกรม

ผู้วิจัยเลือกค่า Relaxation (ω) ที่เหมาะสมกับอัลกอริทึมผ่อนปรนเกินสืบเนื่องคือ 1.80 และใช้ค่านี้ตลอดการทดลอง ซึ่งผลการทดลองแสดงดังตารางที่ 4.2

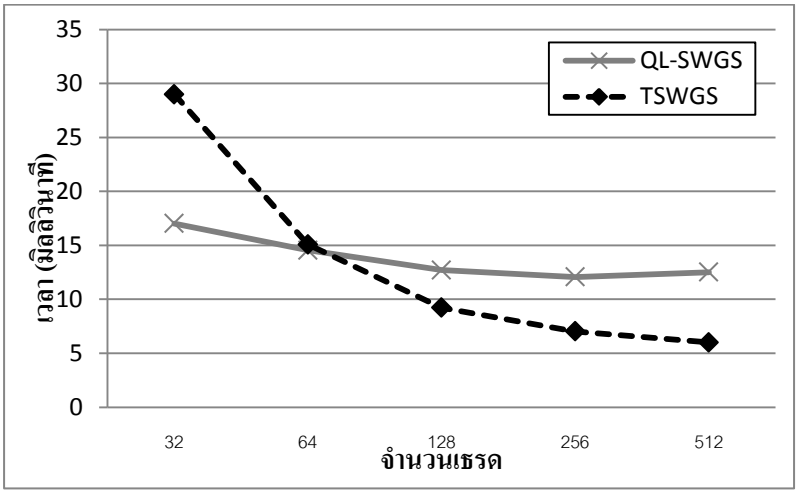
ตารางที่ 4.2 การเปรียบเทียบเวลาที่ใช้เป็นมิลลิวินาทีและอัตราความเร็วที่เพิ่มขึ้นระหว่างอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอย (QL-SWGS) และเกาซ์-เซเดลแบบเลื่อนบานแบบสองทาง (TSWGS) บนเธรด (T) จำนวนต่างๆ

T	**127×127**		**S_1**	**255×255**		**S_2**
	QL-SWGS	**TSWGS**		**QL-SWGS**	**TSWGS**	
32	16299	23830	0.68	266131	307641	0.86
64	14109	14214	0.99	198475	179492	1.11
128	12338	9726	1.26	159188	123555	1.28
256	12301	7604	1.61	141893	102595	1.38
512	12794	6506	1.97	138998	93980	1.48

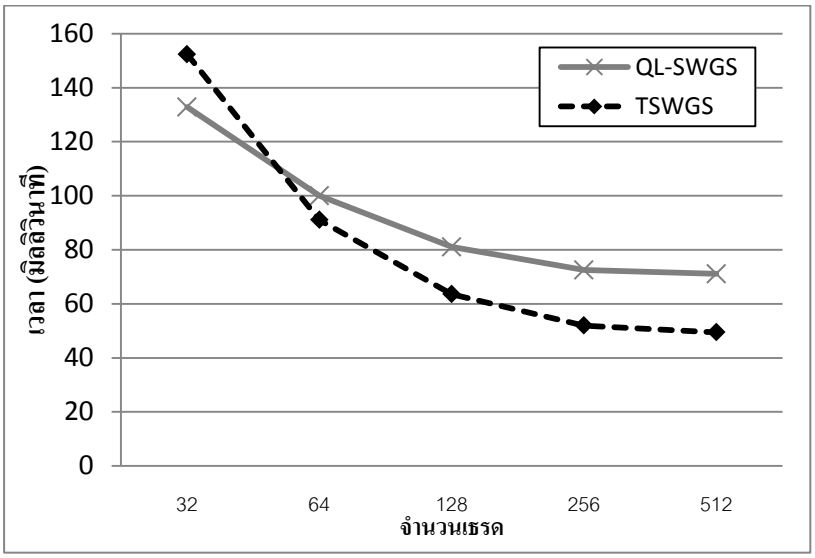
จากตารางที่ 4.2 ในคอลัมน์ T จะแสดงจำนวนเธรดที่ใช้บนโดเมน 2 ขนาดคือขนาด 127×127 และ 255×255 และมีการทำซ้ำจำนวน 2000 รอบ ในการเปรียบเทียบในอัลกอริทึมแบบเดิม (QL-SWGS) และแบบใหม่ที่ผู้วิจัยพัฒนาขึ้น (TSWGS) ทั้งนี้คอลัมน์ S_1 และ S_2 แสดง Speedup ของแบบ TSWGS เทียบกับอัลกอริทึม QL-SWGS บนโดเมนขนาด 127×127 และ 255×255 ตามลำดับ เห็นได้ว่าเมื่อเธรดมีจำนวนมากขึ้น อัลกอริทึม TSWGS ได้ให้ Speedup ที่เพิ่มขึ้นอย่างชัดเจน โดยเฉพาะอย่างยิ่งบนโดเมนขนาด 127×127 การใช้เธรดขนาด 512 กับอัลกอริทึม TSWGS สามารถเพิ่ม Speedup ได้ถึง 1.97 เท่า

อย่างไรก็ดี speedup ที่แสดงในคอลัมน์ S1 และ S2 ชี้ให้เห็นว่าเมื่อโดเมนมีขนาดใหญ่ขึ้น การเพิ่มขึ้นของ Speedup จะลดลงเนื่องจากเวลาที่ใช้ในการรอคอยของเธรดนั้นจะเป็นสัดส่วนน้อยลงเมื่อเทียบกับเวลาประมวลผลโดยรวม

ภาพที่ 4.2 และ 4.3 แสดงเวลา (หน่วยเป็นมิลลิวินาที) เฉลี่ยต่อรอบในการประมวลผลอัลกอริทึม QL-SWGS และอัลกอริทึม TSWGS บนโดเมนขนาด 127×127 และ 255×255 ภายใต้จำนวนเธรดที่มีขนาด 32, 64, 128, 256 และ 512 ตามลำดับ

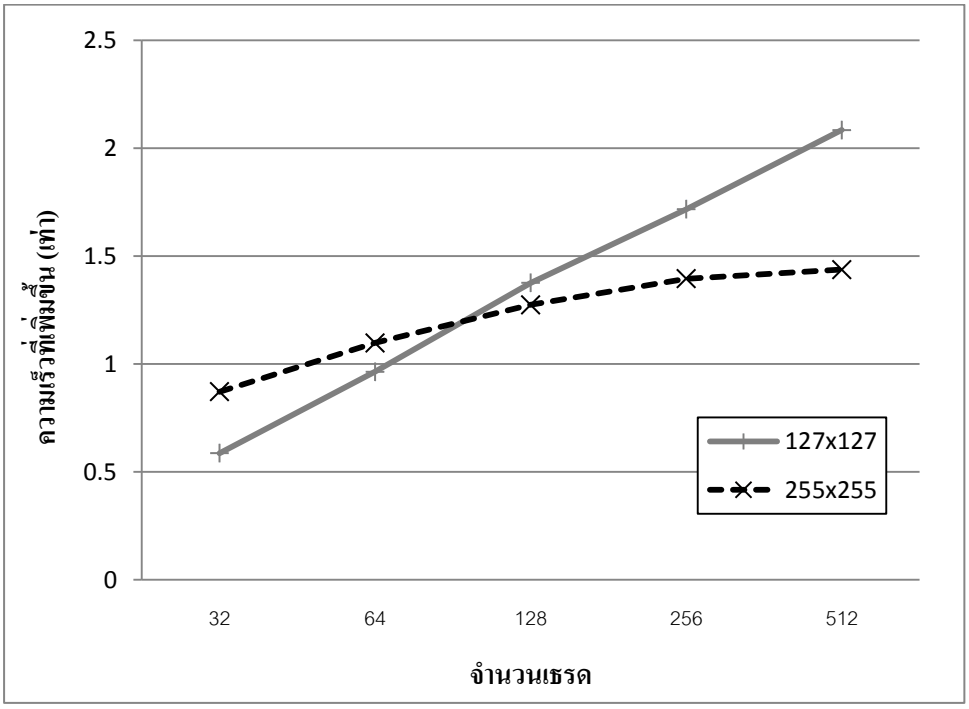


ภาพที่ 4.2 การเปรียบเทียบเวลาเฉลี่ยที่ใช้ต่อรอบการทำซ้ำในของการประมวลผลบนโดเมนขนาด 127×127 บนเธรดที่มีขนาด 32, 64, 128, 256 และ 512



ภาพที่ 4.3 การเปรียบเทียบเวลาเฉลี่ยที่ใช้ต่อรอบการทำซ้ำของการประมวลผลบนโดเมนขนาด 255×255 บนเธรดที่มีขนาด 32, 64, 128, 256 และ 512

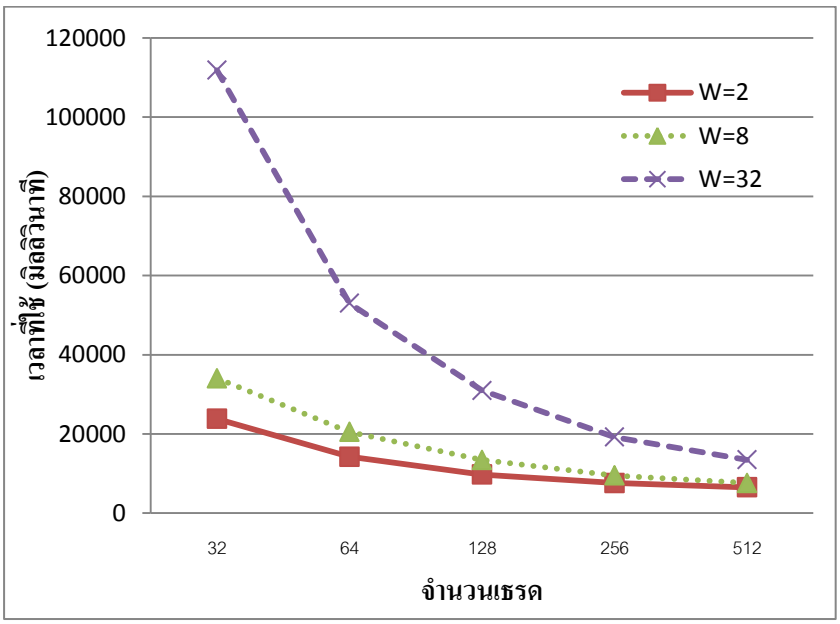
ภาพที่ 4.2 และ 4.3 แสดงให้เห็นว่าเมื่อมีเธรดจำนวนน้อยๆ อัลกอริทึม TSWGS มีประสิทธิภาพด้อยกว่าอัลกอริทึม QL-SWGS เนื่องจากมีจำนวนจุดการคำนวณตามแนวนอนที่สามารถทำได้ในช่วงเวลาหนึ่งน้อยลง แต่เมื่อจำนวนเธรดมีมากกว่า 128 เธรด จะเห็นว่าอัลกอริทึม TSWGS ทำงานได้เร็วกว่า และรองรับการเพิ่มจำนวนเธรดได้มากกว่าในแบบทางเดียว โดยในขณะที่ในแบบทางเดียวความเร็วเริ่มค่อนข้างคงที่เมื่อจำนวนเธรดมีมากกว่า 256 เธรด ทั้งนี้เนื่องจากการใช้จำนวนเธรดขนาดกลางและใหญ่กับอัลกอริทึม TSWGS สามารถเพิ่มประสิทธิภาพให้อัลกอริทึมนี้ทำงานได้อย่างรวดเร็ว และมีจำนวนรอบที่ต้องรอ (Idle Time) น้อยกว่าอัลกอริทึม QL-SWGS ผู้วิจัยจึงได้แสดงค่า Speedup ดังในภาพที่ 4.4



ภาพที่ 4.4 อัตราความเร็วที่เพิ่มขึ้นของอัลกอริทึม TSWGS บนโดเมนขนาด 127×127 และ 255×255 ภายใต้จำนวนเธรดที่เพิ่มขึ้น

ภาพ 4.4 แสดงค่า Speedup โดยเปรียบเทียบระหว่างอัลกอริทึมเดิม (QL-SWGS) เทียบกับอัลกอริทึมใหม่ (TSWGS) โดยเปรียบเทียบบนโดเมน 2 ขนาดคือ 255×255 และ 127×127 โดยทดลองบนจำนวนเธรดที่ต่างกันจะเห็นได้ว่าเมื่อมีจำนวนเธรดเพิ่มขึ้นเกิน 128 เธรด การทำงานในแบบ TSWGS นั้นจะให้ผลที่ดีกว่าในแบบ QL-SWGS สูงสุดถึง 97% บนโดเมนขนาด 127×127 เนื่องจากเธรดที่เพิ่มขึ้นนั้นจะเข้าไปเริ่มทำงานได้รวดเร็วกว่า

อย่างไรก็ดียังมีอีกปัจจัยที่มีผลต่อการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างคือขนาดบานหน้าต่าง (*windowsize*) ซึ่งเป็นจำนวนแถวตามแนวนอนที่ทำในแต่ละรอบย่อย ผู้วิจัยได้ทำการทดลองบนบานหน้าต่างขนาดต่างๆ เพื่อศึกษาผลของขนาดของบานหน้าต่างต่อความเร็วในการประมวลผล ซึ่งเวลาที่ใช้สำหรับรอบการทำซ้ำ 2,000 รอบสามารถแสดงได้ดังภาพที่ 4.5



ภาพที่ 4.5 เปรียบเทียบเวลาที่ใช้โดยอัลกอริทึม TSWGS ในการประมวลผลบนบานหน้าต่าง (W) ขนาดต่างๆ

ผู้ทดลองได้ทดลองบนบานหน้าต่างจำนวน 3 ขนาดคือ 2, 8 และ 32 จะเห็นได้ว่าขนาดบานหน้าต่างนั้นจะมีผลต่อระยะเวลาการประมวลผลเมื่อเธรดมีจำนวนน้อย โดยเมื่อเธรดมีจำนวนน้อยขนาดของบานหน้าต่างที่เล็กลงจะทำให้การทำงานได้รวดเร็วขึ้น และขนาดของบานหน้าต่างจะมีผลน้อยลงในการทดลองที่มีจำนวนเธรดที่มากขึ้นขึ้น เนื่องจากบานหน้าต่างขนาดเล็กนั้นทำให้เธรดจำนวนมากกระจายเข้าทำงานในตอนแรกได้ดีขึ้น ในขณะที่บานหน้าต่างขนาดใหญ่นั้นเธรดเข้าทำงานได้ช้า แต่เมื่อเพิ่มขนาดปัญหาขึ้น สัดส่วนของเวลาที่ต้องรอคอยจะเป็นสัดส่วนน้อยมากกับเวลาที่ประมวลผลโดยรวม ทำให้เวลาที่ได้ใกล้เคียงกันเมื่อปัญหามีขนาดใหญ่ขึ้น

จากผลการทดลองในภาพ 4.5 จะเห็นได้ว่าในโดเมนขนาดเล็ก (127×127) และโดเมนขนาดใหญ่ (255×255) จะได้ผลที่สอดคล้องกันคืออัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง

สองทางสามารถลดระยะเวลาที่ใช้ในการประมวลผลได้ถึงประมาณ 46% ในโดเมนขนาด 127×127 และ 32% ในโดเมนขนาด 255×255 เมื่อเพิ่มจำนวนเธรดถึง 512 เธรด

นอกจากนี้ยังพบว่าการนำอัลกอริทึม TSWGS ไปใช้กับฮาร์ดแวร์ที่มีหน่วยประมวลผลจำนวนมากควรกำหนดจำนวนบานหน้าต่างให้มีขนาดเล็กที่สุดเท่าที่เป็นไปได้เพราะจะเกิดประสิทธิภาพดีที่สุด ทั้งนี้ผู้วิจัยสรุปว่าการเลือกบานหน้าต่างขนาดเล็กนั้นจะทำให้แต่ละเธรดสามารถเข้าทำงานได้รวดเร็ว และลดเวลารอลง อย่างไรก็ตามเมื่อเธรดมีจำนวนมากขึ้น ขนาดของบานหน้าต่างนั้นไม่ได้สร้างความแตกต่างมากนัก ซึ่งจะเห็นได้จากภาพที่ 4.5

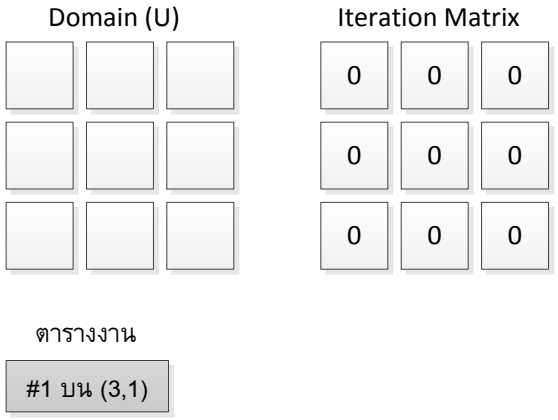
ปัญหาที่พบ

ถึงแม้ว่าอัลกอริทึม TSWGS นั้นจะมีทั้งข้อดีของอัลกอริทึม QL-SWGS ที่สามารถลดเวลาในการผสานการทำงานได้มาก และยังทำให้เธรดเข้าทำงานได้รวดเร็วขึ้นโดยการให้หลายเธรดเข้าทำงานในแต่ละบานหน้าต่าง อย่างไรก็ตามผู้วิจัยยังเห็นว่าจำนวนเธรดที่ทำงานได้แต่ละแถวยังถูกจำกัดเป็น 1 เธรดต่อ 1 แถว ดังนั้นอัลกอริทึมเกาซ์-เซเดลเชิงขนานมีความเป็นไปได้ที่จะสามารถพัฒนาเพื่อเพิ่มประสิทธิภาพได้อีกโดยให้แต่ละแถวสามารถทำงานได้พร้อมกับหลายเธรด ผู้วิจัยจึงได้พัฒนาอัลกอริทึมเกาซ์-เซเดลแบบสถานะ ซึ่งจะกล่าวในหัวข้อต่อไป

อัลกอริทึมเกาซ์-เซเดลแบบสถานะ

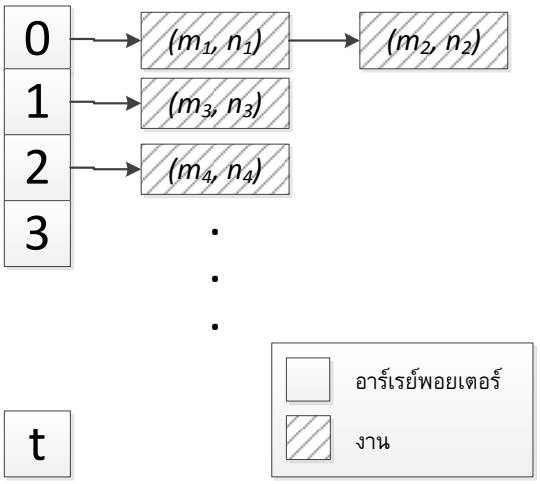
แนวคิดการพัฒนา

ผู้วิจัยได้ออกแบบอัลกอริทึมใหม่ทั้งหมด โดยมีจุดประสงค์เพื่อให้แต่ละเธรดนั้นสามารถทำงานได้อย่างอิสระต่อกันมากขึ้นและในหนึ่งแถวของปัญหาสามารถมีจำนวนเธรดทำงานได้มากกว่าหนึ่ง ผู้วิจัยจึงได้เสนอวิธีการให้แต่ละเธรดมีสถานะต่างๆ โดยเมื่อแต่ละเธรดอยู่ในแต่ละสถานะจะมีวิธีการทำงานที่ต่างกันออกไป และมีตารางงานเพื่อแจกงานให้แต่ละเธรดทำงานอย่างยืดหยุ่น และสามารถนำเธรดที่ประมวลผลเสร็จแล้วกลับมาใช้งานได้ โดยจะใช้เมทริกซ์เพิ่มเติมในการเก็บจำนวนรอบการทำงาน และตารางงานซึ่งเก็บงานของแต่ละเธรดไว้ ซึ่งแสดงในภาพที่ 4.6



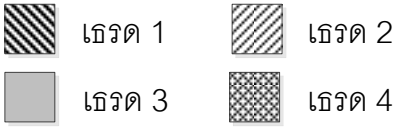
ภาพที่ 4.6 แสดงการสร้างเมทริกซ์ และตารางงาน

จากภาพที่ 4.6 การประมวลผลจะเริ่มที่ตารางงานจะกำหนดงานให้เธรดแรก โดยให้งานที่มุมล่างซ้าย ซึ่งตารางงานนั้นสามารถจะมีการเก็บเป็นอาร์เรย์ของพอยเตอร์และแต่ละพอยเตอร์จะชี้ไปยังโครงสร้างที่เก็บคู่ลำดับ (m_i, n_j) บนโดเมนหลัก ซึ่งแสดงได้ตามภาพที่ 4.7



ภาพที่ 4.7 การเก็บรายการงานของอัลกอริทึมเกาซ์-เซเดลแบบสถานะ

เพื่อให้สามารถเห็นภาพการทำงานของแต่ละเธรดบนโดเมนปัญหาได้อย่างชัดเจน ผู้วิจัยจึงกำหนดพื้นหลังของแต่ละจุดการคำนวณซึ่งแสดงการทำงานของแต่ละเธรดตามภาพที่ 4.8



ภาพที่ 4.8 การกำหนดพื้นหลังแทนการทำงานของเธรดแต่ละเธรดบนโดเมนของปัญหา

เมื่อเริ่มการทำงานเธรดต่างๆ จะมีสถานะได้ 4 แบบคือ

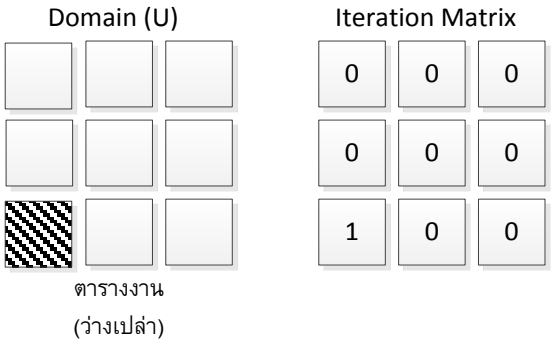
1. สถานะรอ (S1) เป็นสถานะเริ่มต้นของทุกเธรด เธรดที่อยู่ในสถานะนี้จะเข้าไปหางานในตารางงาน และเมื่อพบงานที่ต้องการแล้วจะลบงานออกจากตารางงานและเปลี่ยนสถานะเธรดเป็นสถานะทำงาน (S2)

2. สถานะทำงาน (S2) เธรดที่อยู่ในสถานะนี้จะทำสิ่งต่างๆดังนี้

2.1 เธรดที่อยู่บนจุดการคำนวณ (เซล) ปัจจุบันอ่านข้อมูลจากเซลรอบข้างคือ บน ล่าง ซ้าย และขวา และส่งสัญญาณว่าได้อ่านข้อมูลแล้ว ผู้วิจัยตั้งชื่อสัญญาณนี้ว่า Mark_As_Read สัญญาณนี้ถูกส่งไปยังเซลที่ทำงานอยู่รอบข้างเพื่อให้แน่ใจว่าข้อมูลที่เซลรอบข้างให้นั้นเป็นข้อมูลจากรอบการคำนวณที่ถูกต้อง

2.2 ทำการหาคำตอบโดยประมาณด้วยวิธีการเกาซ์-เซเดลของเซลที่ทำงานอยู่โดยใช้สมการ (3.3) แต่หลังจากทำงานแล้วให้เก็บค่าไว้ในตัวแปรชั่วคราว และยังไม่ปรับปรุงค่าในเซลที่ทำงานอยู่

2.3 เพิ่มค่าในเซลที่ตรงกันใน Iteration Matrix อีก 1 โดยการทำงานของเธรดแรกที่มีสถานะเป็น *สถานะทำงาน* นั้น สามารถแสดงได้ตามภาพที่ 4.9



ภาพที่ 4.9 การทำงานเมื่อเธรดแรกเปลี่ยนจากสถานะรอ เป็นสถานะทำงาน

เมื่อเธรดได้ทำงานตามลำดับดังที่กล่าวมาแล้ว เธรดจะเปลี่ยนสถานะให้เป็น *สถานะตรวจสอบ*

3. สถานะตรวจสอบ (S3) เป็นสถานะที่มีไว้เพื่อตรวจสอบก่อนการปรับปรุงค่าว่าเป็นค่าที่ถูกต้อง โดยเธรดที่อยู่ในสถานะนี้จะทำการตรวจสอบดังนี้

3.1 ถ้าเซลปัจจุบันเป็นเซลแรก (ตำแหน่งพิกัด (3,1) ในภาพตัวอย่าง) เธรดสามารถปรับปรุงข้อมูลได้โดยไม่ต้องตรวจสอบใดๆ เพิ่มเติม

3.2 ถ้าเธรดนั้นคำนวณในแถวแรกหรือคอลัมน์แรก จะต้องได้รับสัญญาณ Mark_As_Read จากเธรดที่อยู่ทางซ้ายมือหรือด้านบนก่อนที่จะปรับปรุงข้อมูล เพื่อให้แน่ใจว่าเซลทางซ้ายนั้นใช้ข้อมูลจากรอบการคำนวณที่ถูกต้องประมวลผล

3.3 เธรดที่ตำแหน่งอื่นๆ นั้นจะต้องได้รับสัญญาณ Mark_As_Read สองครั้งจากเซลทางซ้ายและเซลด้านล่างก่อนการปรับปรุงข้อมูลเพื่อให้แน่ใจว่าเซลรอบข้างนั้นได้นำข้อมูลไปใช้แล้ว แล้วจึงค่อยทำการปรับปรุงค่า

หลังจากปรับปรุงค่า ก็จะเคลียร์ค่าสัญญาณ Mark_As_Read ของเธรดนั้นๆ แล้วจึงเปลี่ยนสถานะเป็นสถานะสุดท้ายคือ*สถานะเปลี่ยนเซล* (S4)

4. สถานะเปลี่ยนเซล (S4) ในสถานะนี้เธรดจะตัดสินใจว่าควรจะทำงานอะไรต่อ โดยมีเงื่อนไขการตัดสินใจดังนี้

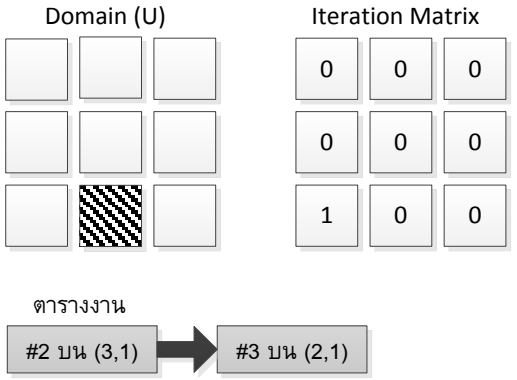
4.1 ถ้าเธรดทำงานอยู่บนคอลัมน์สุดท้าย เธรดจะเปลี่ยนเป็น*สถานะรอ* อีกครั้ง

4.2 ถ้าจำนวนรอบการทำซ้ำใน Iteration Matrix มากกว่าจำนวนรอบสูงสุดที่กำหนดไว้ ให้เธรดหยุดการทำงานและกลับไปยัง*สถานะรอ*

4.3 ถ้าเธรดนั้นทำงานในคอลัมน์แรกและอยู่ในแถวที่มีหมายเลขตรงกับค่า *Max_Row* เธรดนั้นจะเลือกเธรดที่ยังไม่มีงานในตารางงานหรือมีงานน้อยที่สุดในตารางงาน และมอบหมายงานให้เธรดนั้นทำงานในเซลด้านบนและเซลปัจจุบันและเพิ่มค่า Max_Row ขึ้นอีก 1

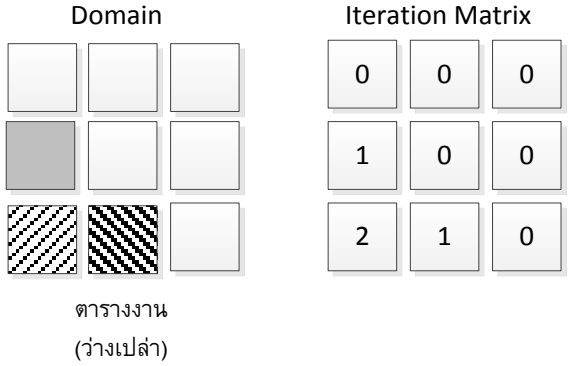
4.4 ถ้าเธรดไม่ได้ทำงานอยู่ในแถวเดียวกับ Max_Row จะเลือกเฉพาะเธรดที่ว่างหรืองานน้อยที่สุดแล้วมอบหมายเธรดนั้นทำงานบนตำแหน่งปัจจุบัน

หลังจากนั้นเธรดจะวนรอบตรวจสอบใน Iteration Matrix ว่าเซลด้านขวานั้นมีค่าน้อยกว่าเซลที่ทำงานอยู่ 1 รอบหรือไม่ ถ้าใช่เธรดจะเลื่อนไปทำงานในเซลถัดไปและเปลี่ยนสถานะของตัวมันเองเป็น *สถานะรอ* ซึ่งแสดงได้ตามภาพที่ 4.10



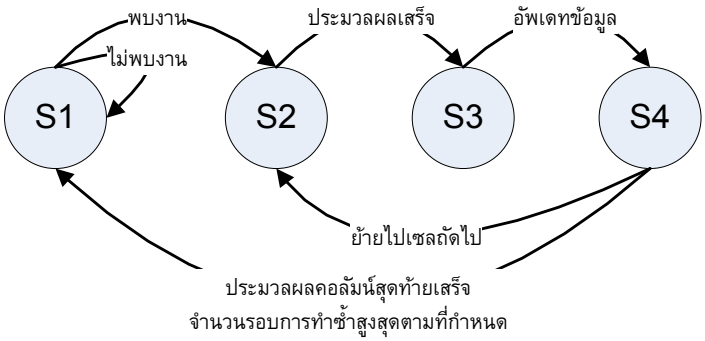
ภาพที่ 4.10 การทำงานเธรดแรกหลังจากสถานะเปลี่ยนเซล

หลังจากนั้นเมื่อเธรดที่ 2 และ 3 พบงานในตารางงาน ก็จะเริ่มเข้าทำงานในตำแหน่งที่ระบุไว้ตามภาพที่ 4.11



ภาพที่ 4.11 การทำงานหลังเธรดที่ 2 และ 3 พบงานในตารางงาน

การเปลี่ยนสถานะต่างๆ ของเธรดนั้น สามารถเขียนเป็นแผนผังแสดงการเปลี่ยนสถานะได้ตามภาพที่ 4.12



ภาพที่ 4.12 แผนผังการเปลี่ยนสถานะ

ในการพัฒนานั้นสามารถเขียนได้เป็นโค้ดเทียม ซึ่งตัวแปรต่างๆ ที่ใช้ในโค้ดเทียมนั้นถูกแสดงในตารางที่ 4.3 และนำไปใช้ในอัลกอริทึม 3

ตารางที่ 4.3 อธิบายตัวแปรต่างๆ และฟังก์ชันต่างๆ ของโค้ดเทียม

ชื่อตัวแปร / ฟังก์ชัน	**ประเภทตัวแปร**	**คำอธิบาย**
N	Integer	ขนาดของปัญหาในแต่ละมิติ
I	Integer	หมายเลขแถวที่ทำงานปัจจุบัน
J	Integer	หมายเลขคอลัมน์ที่ทำงานปัจจุบัน
ThreadID	Integer	หมายเลขกำกับเธรด
STATE	Integer	ตัวแปรสำหรับเก็บค่าสถานะปัจจุบันของเธรด โดยที่ (*1*=สถานะรอคอย, *2*=สถานะทำงาน, *3*=สถานะตรวจสอบ และ *4*=สถานะเลื่อนเซล)
Max_Iteration	Integer	จำนวนรอบการทำซ้ำสูงสุดในการประมวลผล
Max_Row	Integer	แถวสุดท้ายของรอบย่อย
U(N, N)	Double (อาเรย์สองมิติ)	คำตอบโดยประมาณที่เกิดจากการแบ่งโดเมนออกเป็นเซลขนาด $(N+2) \times (N+2)$
Iteration_Matrix*(N, N)*	Integer (อาเรย์สองมิติ)	เมทริกซ์ที่เก็บค่าจำนวนรอบที่ประมวลผลไปแล้วในแต่ละเซล
GET_THREAD_ID()	Function	ฟังก์ชันสำหรับคืนค่าเลขกำกับของเธรดที่ทำงานอยู่ปัจจุบัน
*FIND_ASSIGNED_JOB(*T*)*	Function	ฟังก์ชันสำหรับหางานในตารางงานและคืนค่าเป็นตำแหน่งพิกัด *(I, J)* ที่จะเริ่มประมวลผล
COMPUTE_GS()	Function	ฟังก์ชันสำหรับการคำนวณคำตอบโดยประมาณด้วยวิธีการเกาซ์-เซเดล และมีการคืนค่าที่คำนวณได้
*ADD_JOB(*T, I, J*)*	Function	ฟังก์ชันสำหรับเพิ่มงานตำแหน่ง *(I, J)* ลงในตารางงานให้กับเธรดที่มีเลขกำกับเป็น *T*
FIND_FREE_THREAD()	Function	ฟังก์ชันสำหรับค้นหาตารางงานและคืนค่ากลับเป็นเธรดที่ว่างอยู่ หรือมีงานทำน้อยที่สุด

อัลกอริทึม 3 เกาซ์-เซเดลแบบสถานะ

```
1: declare Iteration_Matrix(N, N), Max_Iteration, U(N,N),  Mark_As_Read(T)
2: declare STATE, Residue_Error, I, J, Max_Row
3: set Max_Error = 10^-4
4: set STATE=Waiting State
5: set ThreadID = GET_THREAD_ID()
6: set Max_Row = N
/* start computation */
7: while NOT DONE do

8:     STATE 1 : WAITING STATE
9:     if STATE=Waiting State then
10:       (I, J) = FIND_ASSIGNED_JOB(ThreadID)
11:     if FOUND (I,J) then
12:       set STATE=Working State
13:    endif

14:    STATE 2 : WORKING STATE
15:    if STATE= Working State then
16:     READ ADJACENT CELLS  OF U (I-1, J), (I+1, J), (I, J+1), (I, J-1)
17:     SET Mark_As_Read(THREADID) where THREADIDs are   threads working on
           (I-1, J), (I+1, J), (I, J+1), (I, J-1)
18:     RESULT = COMPUTE_GS() using Eq. (3.3)
19:     Iteration_Matrix(I, J) =  Iteration_Matrix(I, J) + 1
20:     set STATE=Validation State
21:    endif

22:    STATE 3 : VALIDATION STATE
23:    if STATE=Validation State then
24:      if I=N and J=1 then   /* first cell of computation */
25:         update U(I, J) = [Compute Equation (3.3)]
26:         set STATE=Shifting State
27:         continue  /* back to top of the loop */
28:       endif
29:    if (I=N OR J=1) AND Mark_As_Read(THREADID) >=1 then
```

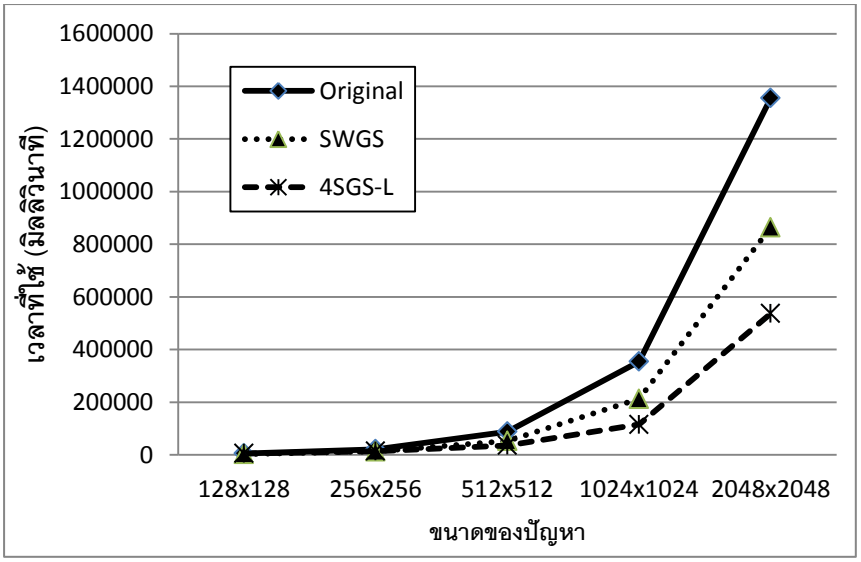
```
30:          update U(I, J) = RESULT
31:          set STATE= Shifting State
32:          continue
33:    endif
34:    if Mark_As_Read(ThreadID)>=2 then
35:          update U(I, J) = RESULT
36:          set STATE= Shifting State
37:          continue
38:    endif
39:  endif

40:    STATE 4 : SHIFTING STATE
41:    if STATE=Shifting State then
42:      if J=N then  /* last column */
43:        set STATE=Waiting State
44:        continue
45:      endif
46:      if Iteration_Matrix(I, J+1) >= Max_Iteration
47:        set STATE=Waiting State
48:        continue
49:      endif
50:      while (Iteration_Matrix(I,J+1) < Iteration_Matrix(I, J)) do
51:        DO NOTHING
52:      if J=1 then
53:        if I=Max_Row then
54:         ADD_JOB(FIND_FREE_THREAD(), I, J)
55:         ADD_JOB(FIND_FREE_THREAD(), I-1, J)
56:         Max_Row = Max_Row - 1
57:        else
58:          ADD_JOB(FIND_FREE_THREAD(), I-1, J)
59:        endif
60:          J = J+1
61:        set STATE=Working State
62:        continue
61:    endif
62:  /* compute residue error using max_error here if need */
63:  endwhile
```

ผลการพัฒนา

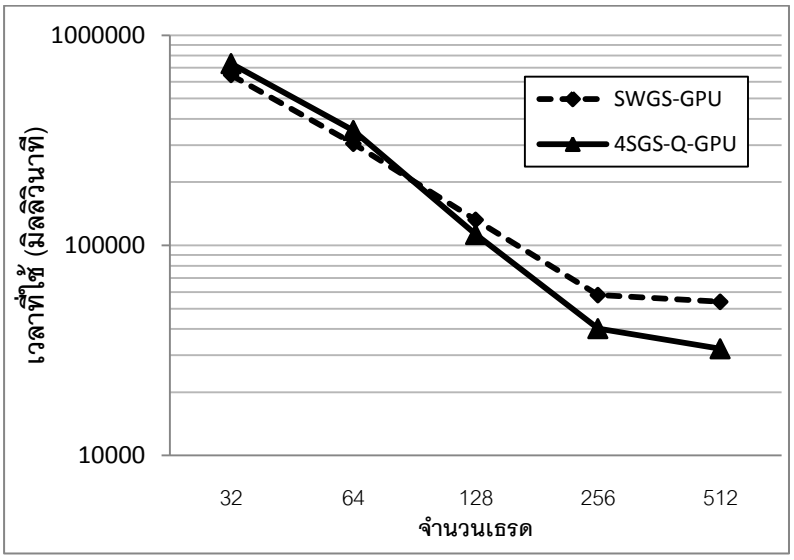
ผู้วิจัยได้แบ่งอัลกอริทึมเกาซ์-เซเดลแบบสถานะออกเป็น 2 แบบคือ แบบที่ทำงานกับหน่วยประมวลผลหลักแบบหลายแกนซึ่งจะมีการเก็บข้อมูลแบบลิสต์เป็นหลัก (4SGS-L) และแบบที่ทำงานกับหน่วยประมวลผลกราฟิกส์ ซึ่งมีการใช้แถวคอยเป็นหลักเพื่อหลีกเลี่ยงการใช้พอยเตอร์ (4SGS-Q)

การเปรียบเทียบเวลาที่ใช้ในการประมวลผลระหว่างอัลกอริทึมเดิม และอัลกอริทึมเกาซ์-เซเดลแบบสถานะที่พัฒนาได้เมื่อทำจนลู่เข้าคำตอบโดยมีนอร์มของเศษตกค้างไม่เกิน 10^{-4} แสดงได้ตามภาพ 4.13



ภาพที่ 4.13 เวลาที่ใช้ในการประมวลผลของเกาซ์-เซเดลแบบปกติ (Original) แบบเลื่อนบานหน้าต่าง (SWGS) และแบบสถานะ (4SGS-L) บน 128 เธรด

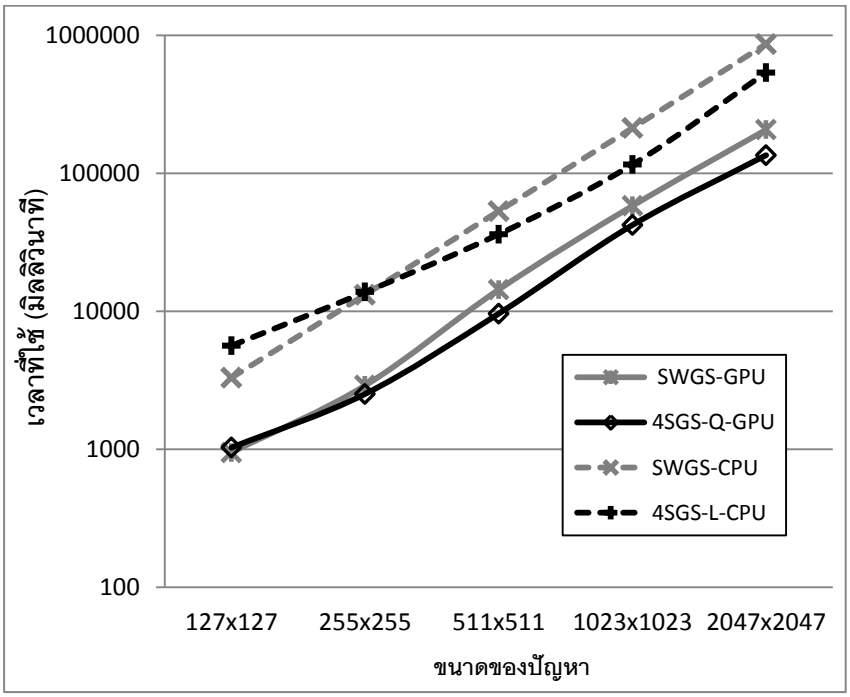
จะเห็นว่าเวลาที่ใช้ของ *4SGS-L* นั้นลดลงไปได้ถึงประมาณ 20% เมื่อเทียบกับ SWGS แบบเดิมบนขนาดปัญหา 2,048×2,048 นอกจากนี้ผู้วิจัยยังได้ทดลองกับจำนวนเธรดขนาดต่างๆ บนปัญหาขนาด 512×512 ซึ่งได้ผลตามภาพที่ 4.14



ภาพที่ 4.14 แสดงระยะเวลาในการทำงานของทั้งสองอัลกอริทึมกับจำนวนเธรดขนาดต่างๆ บนหน่วยประมวลผลกราฟิกส์

จากการทดลองบนหน่วยประมวลผลกราฟิกส์พบว่าเมื่อจำนวนเธรดไม่มาก (น้อยกว่า 128 เธรด) นั้นอัลกอริทึม SWGS ทำงานได้รวดเร็วกว่า เนื่องจากไม่ต้องมีการค้นหางานในตารางงานในขณะที่เมื่อเธรดจำนวนมากขึ้น (128 เธรดขึ้นไป) อัลกอริทึม *4SGS-Q* ที่พัฒนาขึ้นนั้นจะทำงานได้รวดเร็วกว่าเนื่องจากลดระยะเวลาผสานการทำงานระหว่างเธรด และระยะเวลาในการรอคอยเข้าไปทำงานได้มาก

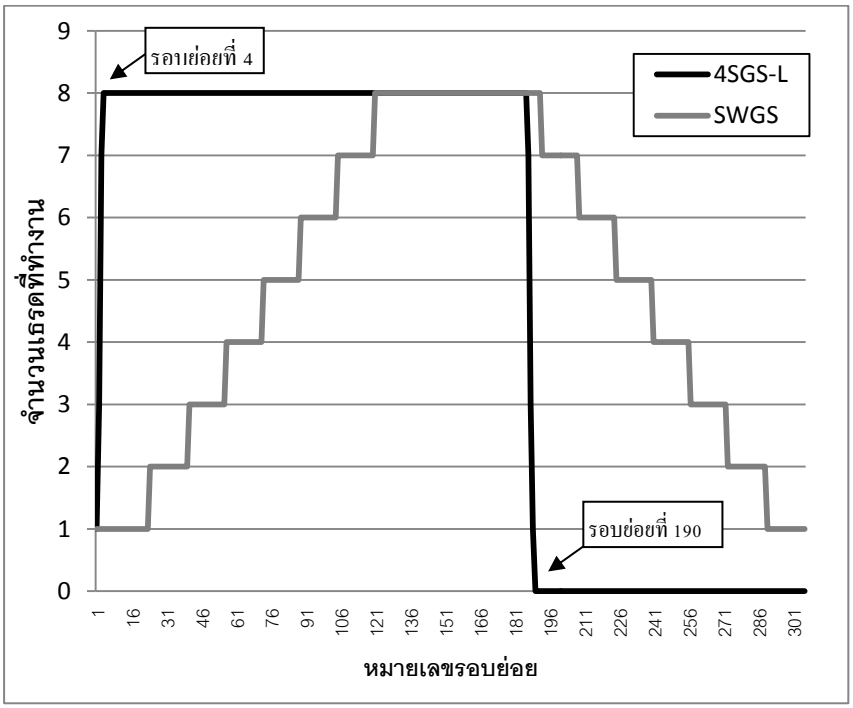
ผู้วิจัยเปรียบเทียบระยะเวลาการทำงานทั้งบนหน่วยประมวลผลหลักและหน่วยประมวลผลกราฟิกส์ได้ตามภาพที่ 4.15



ภาพที่ 4.15 ผลการเปรียบเทียบเวลาการทำงานของอัลกอริทึมที่พัฒนาขึ้นใหม่ (4SGS-Q-GPU และ 4SGS-L-CPU) และอัลกอริทึมเดิมบนหน่วยประมวลผลหลักแบบ 4 แกน และหน่วยประมวลผลกราฟิกส์ (SWGS-CPU และ SWGS-GPU)

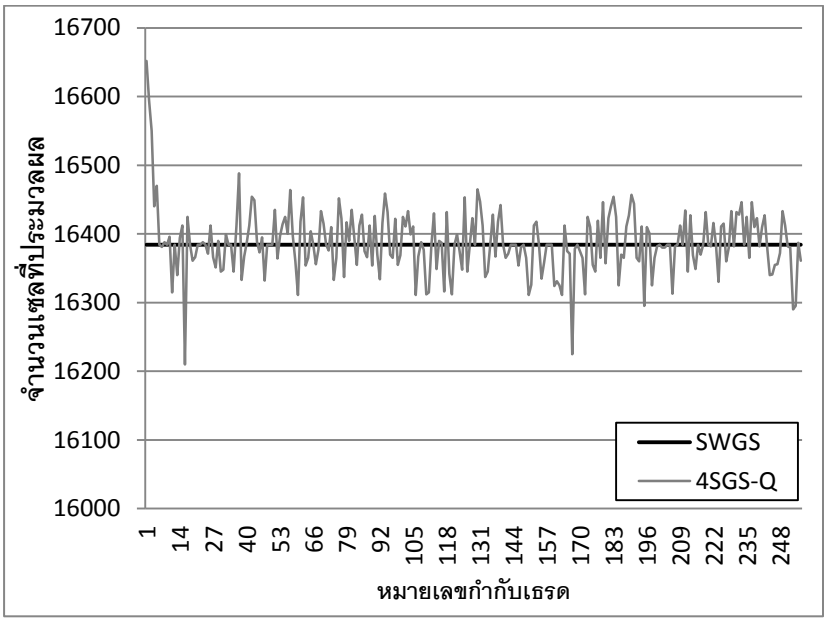
จะเห็นได้ว่าเมื่อขนาดของปัญหาเพิ่มขึ้น อัลกอริทึม *4SGS-Q* และ *4SGS-L* ที่พัฒนาได้นั้นจะมีประสิทธิภาพเพิ่มขึ้นไปด้วยเนื่องจากสามารถลดการผสานการทำงานและการรอคอยเข้าทำงานได้อย่างมาก โดยสำหรับขนาดปัญหา 1023×1023 นั้น อัลกอริทึมเกาซ์-เซเดลแบบสถานะสามารถทำงานได้เร็วกว่าการประมวลผลแบบตามลำดับบนหน่วยประมวลผลหลักถึง 26.88 เท่า

นอกจากนี้ผู้วิจัยยังได้วัดการกระจายงาน และการรอคอยของเธรดของอัลกอริทึมที่พัฒนาได้กับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างเดิม ซึ่งจากการทดลองบนหน่วยประมวลผลหลักแบบ 4 เธรดได้ผลตามภาพที่ 4.16 และ 4.17



ภาพที่ 4.16 แสดงจำนวนเธรดที่เข้าทำงานในแต่ละรอบย่อย

จากภาพที่ 4.16 จะเห็นได้ว่าอัลกอริทึมที่พัฒนาขึ้นใหม่นั้นสามารถใช้งานเธรดทั้งหมดได้ภายใน 4 รอบย่อยของการทำงานในขณะที่อัลกอริทึมเดิมใช้เวลาประมาณ 100 รอบทุกเธรดจึงสามารถเข้าไปทำงานได้ และเมื่อเธรดต่างๆ ทำงานเสร็จสิ้น อัลกอริทึมที่พัฒนาใหม่นี้สามารถนำเธรดที่ทำงานเสร็จสิ้นแล้วกลับมาใช้งานใหม่ได้ โดยในอัลกอริทึม 4SGS-L นั้นมีการอัตราส่วนการใช้งานหน่วยประมวลผลเทียบกับเวลา (% Utilization) เป็น 99.67% ในขณะที่ SWGS นั้นมีค่าเพียง 57.71% เท่านั้น



ภาพที่ 4.17 เปรียบเทียบปริมาณเซลที่ทำงานของแต่ละเธรด

จากภาพ 4.17 อัลกอริทึมที่พัฒนาได้นั้นจะมีการทำงานแบบยืดหยุ่นและมีการเข้าทำงานในลักษณะแข่งขันในขณะที่อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างเดิมจะมีการทำงานกับจำนวนเซลที่คงที่เท่ากันทุกเธรด

การวิเคราะห์ความซับซ้อนด้านการประมวลผลและหน่วยความจำ

ในส่วนนี้ผู้วิจัยได้วิเคราะห์ความซับซ้อนของการประมวลผลและหน่วยความจำของอัลกอริทึม *4SGS-L* และ *4SGS-Q* ที่พัฒนาได้เทียบกับอัลกอริทึม SWGS ของเดิม โดยผู้วิจัยกำหนดตัวแปรที่ใช้ในการวิเคราะห์ต่างๆ ดังนี้

N	ขนาดของปัญหาในแต่ละมิติ
F	ขนาดหน่วยความจำที่ใช้เก็บจำนวนจริง แบบ Double Precision ซึ่งโดยทั่วไปแล้ว GNU C Compiler ใช้ 8 ไบต์
B	ขนาดหน่วยความจำที่ต้องการสำหรับเมทริกซ์ Mark_As_Read
I	ขนาดหน่วยความจำที่ต้องการใช้เก็บเลขจำนวนเต็ม ปกติแล้วใช้ 4 ไบต์
L	ขนาดหน่วยความจำที่ใช้เก็บสมาชิกแต่ละตัวของลิสต์
t	จำนวนเธรดที่ใช้

หน่วยความจำที่ต้องการใช้สำหรับการทำงานอัลกอริทึมเกาซ์-เซเดลแบบสถานะเพื่อแก้สมการปัวซงแบบสองมิติ เขียนได้ตามสมการที่ (4.4)

$$O(FN^2 + IN^2 + tB + tL) \tag{4.4}$$

สำหรับการทำงานบนหน่วยประมวลผลกราฟิกส์นั้นจะต้องใช้หน่วยความจำมากกว่าในแบบการทำงานกับหน่วยประมวลผลหลัก เนื่องจากผู้วิจัยหลีกเลี่ยงการใช้งานพอยเตอร์และใช้งานแถวคอยแทนลิสต์ [37] โดยพื้นที่หน่วยความจำที่ต้องการของแต่ละเอเลเมนต์ของลิสต์นั้นจะแทนด้วยสัญลักษณ์ Q ซึ่งหน่วยความจำที่ต้องการนั้นสามารถเขียนได้ตามสมการ (4.5)

$$O(FN^2 + IN^2 + tB + tQN) \tag{4.5}$$

จากสมการ (4.4) และ (4.5) จะเห็นได้ว่า F, I, B และ Q นั้นเป็นค่าคงที่ โดย F คือพื้นที่ใช้เก็บจำนวนจริงขนาด 8 ไบต์ และ I ใช้เนื้อที่ 4 ไบต์ และ L ประกอบด้วยสามส่วนคือส่วนที่เป็นพอยเตอร์ใช้ 8 ไบต์สำหรับระบบแบบ 64 บิต จำนวนเต็มแบบสั้น (Short Integer) สำหรับเก็บหมายเลขของเธรด และค่าพิกัด I, J ซึ่งแต่ละตัวแปรใช้ 4 ไบต์รวมทั้งสิ้นเป็น 18 ไบต์ สมการ (4.3) และ (4.4) สามารถเขียนใหม่ได้เป็นสมการ (4.5) และ (4.6) ตามลำดับ

$$O(\frac{3}{2}FN^2 + \frac{19}{18}tL) \tag{4.6}$$

$$O(\frac{3}{2}FN^2 + \frac{5}{4}tQN) \tag{4.7}$$

จากสมการที่ (4.6) และ (4.7) แสดงให้เห็นว่าอัลกอริทึม *4SGS-Q* นั้นใช้พื้นที่ในการเก็บข้อมูลมากกว่าอัลกอริทึม *4SGS-L* เมื่อตัวแปร N มีขนาดใหญ่พอ เมื่อเปรียบเทียบกับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างของเดิมคือ

$$O(FN^2) \tag{4.8}$$

จะเห็นได้ว่าอัลกอริทึมทั้งสองตัวที่พัฒนาขึ้นใหม่จะใช้เนื้อที่ในการเก็บข้อมูลมากกว่าประมาณ 50% ในส่วนของสมการความซับซ้อนของเวลาที่ใช้ในการประมวลผลแต่ละเธรดนั้นสามารถเขียนได้ตามสมการ (4.9)

$$O(\frac{k}{t}N^2) \tag{4.9}$$

ในที่นี้ k คือจำนวนของรอบการทำซ้ำ และ t คือจำนวนเธรด ผู้วิจัยได้ทำตารางเปรียบเทียบเรียงลำดับปริมาณหน่วยความจำที่ต้องการใช้ของอัลกอริทึมที่พัฒนาได้ในตารางที่ 4.4

ตาราง 4.4 การเปรียบเทียบความซับซ้อนในการประมวลผลและหน่วยความจำของอัลกอริทึมต่างๆ

ความซับซ้อน	***4SGS-L***	***4SGS-Q***	***SWGS***
หน่วยความจำ	$O(\frac{3}{2}FN^2+\frac{19}{18}tL)$	$O(\frac{3}{2}FN^2+\frac{5}{4}tQN)$	$O(FN^2)$
เวลาประมวลผล	$O(\frac{k}{t}N^2)$	$O(\frac{k}{t}N^2)$	$O(\frac{k}{t}N^2)$
เวลาค้นหางาน	$O(tN)$	$O(c)$	-

จากตารางที่ 4.4 จะเห็นได้ว่าจำนวนช่องที่ใช้ในการประมวลผลต่อเธรดนั้นเท่ากันในทุกอัลกอริทึม อย่างไรก็ดีอัลกอริทึม *4SGS-L* จะใช้พื้นที่ในการเก็บข้อมูลน้อยกว่าอัลกอริทึม *4SGS-Q* โดยเวลาที่ใช้ในการประมวลผลและหน่วยความจำที่ใช้แสดงได้ตามตารางที่ 4.5

ตาราง 4.5 *:* ตารางแสดงการใช้หน่วยความจำและการค้นหาของแต่ละอัลกอริทึม

ความซับซ้อน	**สูงสุด**	**ปานกลาง**	**ต่ำสุด**
หน่วยความจำ	*4SGS-Q*	*4SGS-L*	*SWGS*

จะเห็นได้ว่าอัลกอริทึม SWGS นั้นใช้หน่วยความจำต่ำสุดเนื่องจากไม่มีตารางงาน ในขณะที่อัลกอริทึม *4SGS-Q* ใช้การเก็บตารางงานแบบแถวคอยซึ่งทำให้ใช้หน่วยความจำมากที่สุดแต่เนื่องจากใช้แถวคอยในแต่ละเธรด

บทที่ 5

การประยุกต์ใช้อัลกอริทึมเกาซ์-เซเดลเชิงขนานในการตัดสัญญาณรบกวนภาพประเภทบวก

สัญญาณรบกวนในภาพประเภทบวกนั้นมีวิธีในการตัดสัญญาณรบกวนได้หลากหลายวิธี เช่น วิธีการแปลงเวฟเลตแบบเต็มหน่วย (Discrete Wavelet Transform method), วิธีการปรับเรียบแบบเกาซ์เซียน (Gaussian Smoother method) แต่วิธีการที่มีประสิทธิภาพมากในการตัดสัญญาณรบกวนประเภทบวกคือ โมเดลของ ROF ที่นำเสนอโดย Rudin, Osher และ Fatami [34] โดยโมเดลสัญญาณรบกวนนี้สามารถแก้ได้หลายวิธี แต่วิธีการซึ่งเป็นที่นิยมคือการแก้ระบบสมการไม่เป็นเชิงเส้นที่เกิดจากการใช้วิธีการผลต่างอันตะกับสมการออยเลอร์-ลากรานจ์ที่สอดคล้องกับโมเดล ROF ด้วยวิธีการทำซ้ำแบบจุดตรึง ซึ่งขั้นตอนภายในการทำซ้ำแบบจุดตรึงนี้จะใช้อัลกอริทึมเกาซ์-เซเดลเข้ามาแก้ระบบสมการที่ไม่ถูกทำให้เป็นเชิงเส้น ซึ่งในบทนี้จะนำเกาซ์-เซเดลที่พัฒนาได้ในบทที่ 4 มาประยุกต์ใช้กับโมเดล ROFในการตัดสัญญาณรบกวนประเภทนี้

วิธีการดำเนินงาน

ในการตัดสัญญาณรบกวนบนภาพนี้ ผู้วิจัยได้เลือกใช้หน่วยประมวลผลกราฟิกส์ NVIDIA GeForce GTX-480 ในการตัดสัญญาณรบกวนบนภาพและใช้อัลกอริทึมเกาซ์-เซเดลแบบต่างๆ ที่พัฒนาได้ในบทที่ 4 เพื่อแก้ระบบสมการที่ถูกทำให้เป็นเชิงเส้นจากวิธีการทำซ้ำจุดตรึง ทั้งนี้ผู้วิจัยจะกล่าวทบทวนการพัฒนาอัลกอริทึมเกซ์-เซเดลแบบต่างๆ ที่พัฒนาได้ในบทที่ 4 ดังนี้

จากปัญหาของเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างที่ได้กล่าวถึงในบทที่ 4 ไว้ว่าอัลกอริทึมไม่เหมาะสำหรับการทำงานกับเธรดจำนวนมาก ผู้วิจัยจึงได้เริ่มการปรับปรุงการทำงานของอัลกอริทึมนี้โดยมีเป้าหมายเพื่อลดปริมาณการผสานการทำงานภายในรอบการทำงานและระหว่างรอบการทำงานและเรียกอัลกอริทึมที่ถูกปรับปรุงนี้ว่าอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง

ไร้แถวคอย (อัลกอริทึม QL-SWGS) อย่างไรก็ตามอัลกอริทึมที่ได้นี้ประสบปัญหาการมีเธรดว่างงานจำนวนมาก เพื่อแก้ไขปัญหาที่เกิดขึ้นนี้ผู้วิจัยได้พัฒนาการประมวลผลอัลกอริทึมที่ได้นี้แบบขนานซึ่งให้ความเหมาะสมกับการทำงานบนหน่วยประมวลผลกราฟิกส์ ทั้งนี้ผู้วิจัยได้เรียกอัลกอริทึมที่ถูกพัฒนาขึ้นใหม่นี้ว่าอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทาง (อัลกอริทึม TSWGS) ดังที่ได้กล่าวไว้ในบทที่ 4

ผลการทดลอง

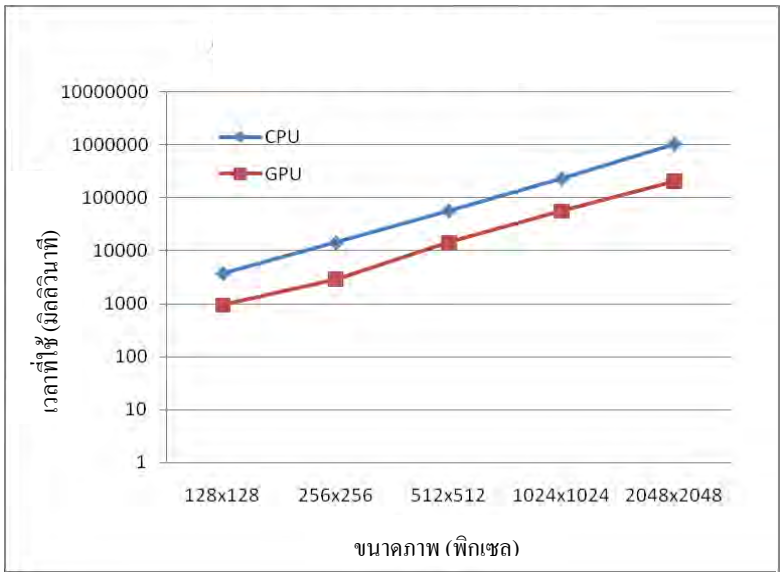
ผู้วิจัยได้แบ่งการทดลองออกเป็นสองส่วนคือการทดลองเพื่อตรวจสอบด้านประสิทธิภาพของการทำงาน โดยการวัดเวลาในส่วนของความเร็วในการทำงานที่เพิ่มขึ้นจากการใช้อัลกอริทึมเกาซ์-เซเดลแบบต่างๆ ที่พัฒนาได้ในการตัดสัญญาณรบกวน และการทดลองเพื่อตรวจสอบด้านคุณภาพของภาพที่ได้จากการตัดสัญญาณรบกวน

ด้านประสิทธิภาพการทำงาน

ในการทดลองนี้ผู้วิจัยใช้เวลาเพื่อตรวจสอบประสิทธิภาพการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยบนหน่วยประมวลผลกลาง (CPU) และหน่วยประมวลผลกราฟิกส์ (GPU) ในการตัดสัญญาณรบกวน

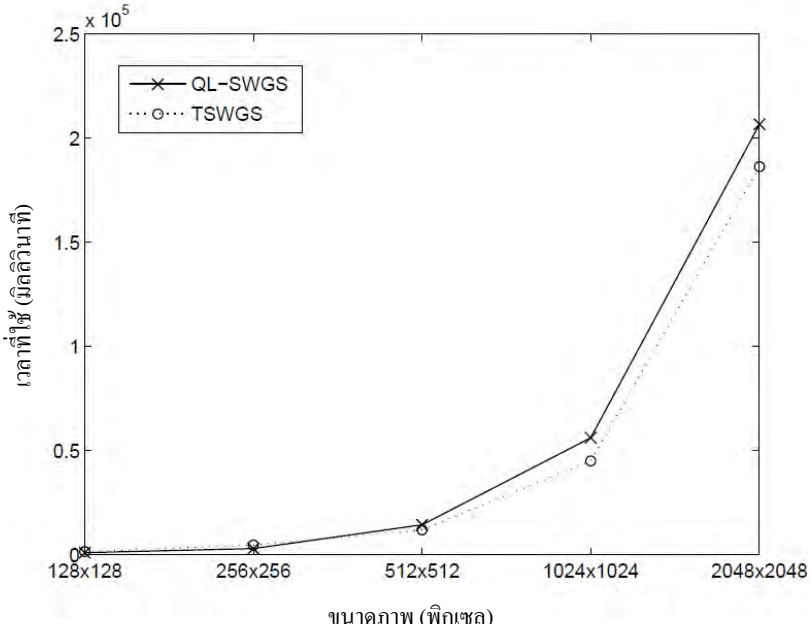
ทั้งนี้ เวลาที่ใช้ในการตัดสัญญาณรบกวนเป็นเวลาที่ได้จากการใช้วิธีการทำซ้ำแบบจุดตรึงร่วมกับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยจนกระทั่งไม่มีการเปลี่ยนแปลงอย่างมีนัยสำคัญของค่า PSNR ระหว่างภาพที่ถูกตัดสัญญาณรบกวนและภาพต้นฉบับ

เวลาที่ใช้ในการตัดสัญญาณรบกวนออกจากภาพถ่ายในการทำซ้ำแบบจุดตรึงจนกระทั่งค่า PSNR ไม่เกิดการเปลี่ยนแปลงระหว่างรอบ สามารถแสดงได้ตามภาพที่ 5.1 ในมาตราส่วนแบบลอการิทึม



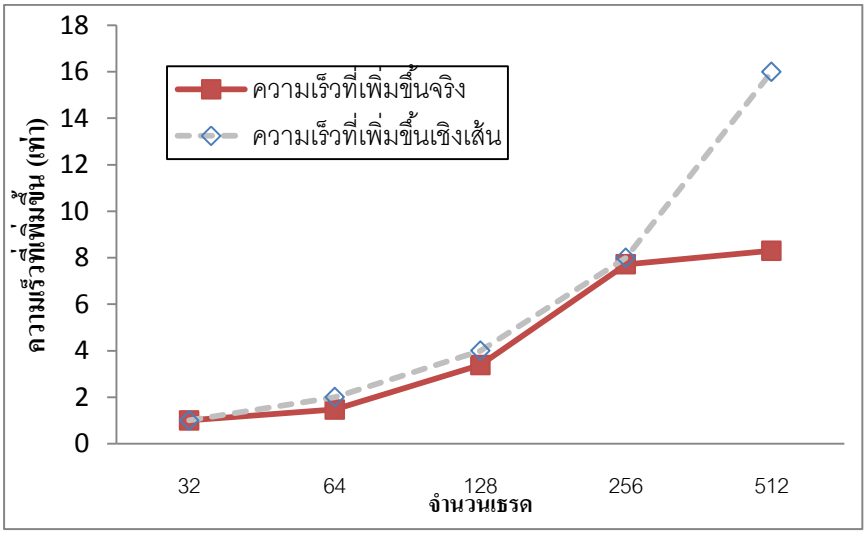
ภาพที่ 5.1 ระยะเวลาที่ใช้ในการตัดสัญญาณรบกวนของภาพขนาดต่างๆ (สเกลแบบลอการิทึม)

จากภาพที่ 5.1 แสดงเวลาที่ใช้ในการตัดสัญญาณรบกวนออกจากภาพที่มีขนาดต่างกันบนหน่วยประมวลผลกลางและหน่วยประมวลผลกราฟิกส์ภายใต้มาตราส่วนลอการิทึม เห็นได้ว่าการใช้อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยบนหน่วยประมวลผลกราฟิกส์มีประสิทธิภาพสูงและสามารถลดเวลาในการประมวลผลได้ถึง 80.32%



ภาพที่ 5.2 การเปรียบเทียบเวลาในการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนหน้าต่างไร้แถวคอย และแบบเลื่อนหน้าต่างสองทางบนภาพขนาดต่างๆ

ภาพที่ 5.2 แสดงผลการเปรียบเทียบการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนหน้าต่างไร้แถวคอยและอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนหน้าต่างสองทางบนภาพที่ขนาดต่างกัน เห็นได้ว่าอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนหน้าต่างสองทางจะลดเวลาในการประมวลผลได้ถึง 9.86% เมื่อเทียบกับการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนหน้าต่างไร้แถวคอยเนื่องจากสามารถลดเวลารอคอยของเธรดได้มาก



ภาพที่ 5.3 ค่าความเร็วที่เพิ่มขึ้นของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทางทำงานบนหน่วยประมวลผลกราฟิกส์ภายใต้จำนวนเธรดขนาดต่างๆ

ภาพที่ 5.3 แสดงความเร็วที่เพิ่มขึ้นของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทางเมื่อเทียบกับการประมวลผลแบบ 32 เธรด มีสัดส่วนความเร็วที่เพิ่มขึ้นใกล้เคียงกับความเร็วเชิงเส้น ซึ่งเกิดจากการทำงานของหน่วยประมวลผลที่ดีขึ้นเมื่อใช้เธรดจำนวนมากทำงานในเวลาเดียวกัน และการทำงานเมื่อใช้เธรดจำนวนมากทำงานกับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยที่พัฒนาได้กับเธรดจำนวนมากนั้นจะลดจำนวนครั้งของการผสานการทำงานระหว่างรอบการทำงานได้ เนื่องจากจำนวนรอบที่ทำงานลดลง

ในส่วนของกรณี 512 เธรดนั้น อัตราการเพิ่มของความเร็วที่เพิ่มขึ้นลดลงเนื่องจากฮาร์ดแวร์หน่วยประมวลผลกราฟิกส์ที่ใช้ในการทดลองมีจำนวนหน่วยประมวลผลเพียง 480 หน่วยประมวลผลเท่านั้น ทำให้การเพิ่มของความเร็วนั้นไม่เติบโตในอัตราเดียวกับจำนวนเธรดอื่นๆ

ด้านคุณภาพของภาพที่ตัดสัญญาณรบกวน

ในการทดลองนี้ได้ใช้ภาพสองภาพซึ่งเป็นภาพที่ใช้ในการทดลองด้านการประมวลผลภาพอย่างกว้างขวางคือภาพที่ชื่อว่า Cameraman และ Pepper ภาพทั้งสองถูกใส่สัญญาณรบกวนประเภทบวกสังเคราะห์สามระดับคือ 10%, 25% และ 50% ซึ่งภาพตัวอย่างนั้นแสดงได้ดังภาพ 5.4 และ 5.5

(ก) 10%

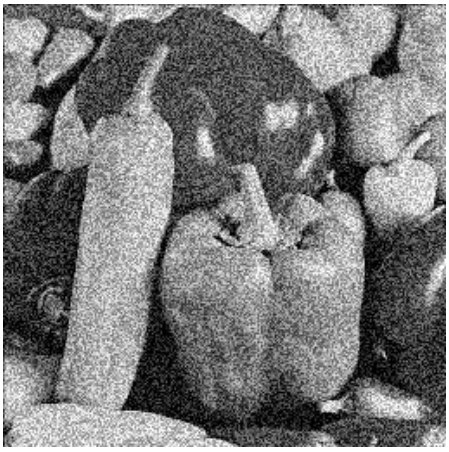
(ข) 25%

(ค) 50%

ภาพที่ 5.4 ภาพ “Cameraman” ต้นฉบับและภาพที่ใส่สัญญาณรบกวนประเภทบวกสังเคราะห์ขนาดต่างๆกัน

ตัวอย่างภาพ Cameraman ต้นฉบับ (ก) ภาพที่สังเคราะห์สัญญาณรบกวน 25% (ข) ภาพที่สังเคราะห์สัญญาณรบกวน 50% (ค) ภาพ Pepper เมื่อสังเคราะห์สัญญาณรบกวนแล้ว จะแสดงได้ตามภาพที่ 5.5

(ก) 10%



(ข) 25%

(ค) 50%

ภาพที่ 5.5 ภาพ “Pepper” ต้นฉบับและภาพที่ใส่สัญญาณรบกวนประเภทบวกสังเคราะห์ขนาดต่างๆกัน

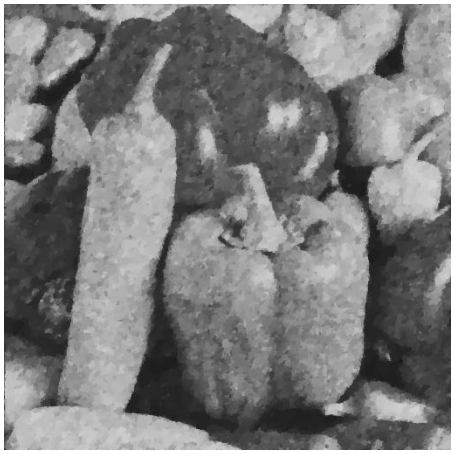
ในภาพที่ 5.5 แสดง ตัวอย่างภาพ Pepper ต้นฉบับ (ก) ภาพที่สังเคราะห์สัญญาณรบกวน 25% (ข) ภาพที่สังเคราะห์สัญญาณรบกวน 50% (ค)

ภาพที่ได้จากการตัดสัญญาณรบกวนนั้น สามารถแสดงได้ตามภาพที่ 5.6

(ก)

(ข)

(ค)



(ง)

ภาพที่ 5.6 ภาพผลลัพธ์จากการตัดสัญญาณรบกวนประเภทบวกด้วยโมเดล ROF จากสัญญาณรบกวนสังเคราะห์ 25% (ก และ ข) และสัญญาณรบกวนสังเคราะห์ 50% (ค และ ง)

เห็นได้ว่าภาพที่ถูกตัดสัญญาณรบกวน แสดงรายละเอียดเพิ่มเติมที่มองเห็นได้ยากในภาพที่มีสัญญาณรบกวน ตาราง 5.1 แสดงค่า PSNR ที่เป็นผลมาจากการทดลองบนภาพ cameraman ระหว่างภาพที่ถูกตัดสัญญาณรบกวนและภาพต้นฉบับภายใต้ปริมาณสัญญาณที่ต่างกัน

ตารางที่ 5.1 ค่า PSNR ของภาพที่เป็นผลมาจากการทดลองบนภาพ cameraman ระหว่างภาพที่ถูกตัดสัญญาณรบกวนและภาพต้นฉบับภายใต้ปริมาณสัญญาณรบกวนที่เพิ่มขึ้น

ขนาดภาพ	**% ของสัญญาณรบกวนแบบเกาซ์**		
	สัญญาณรบกวน 10%	**สัญญาณรบกวน 25%**	**สัญญาณรบกวน 50%**
256×256	66.31	42.16	29.92
512×512	66.36	43.58	30.23
1024×1024	66.45	43.91	30.77
2048×2048	66.73	44.68	31.58

จากตาราง 5.1 เห็นได้ว่าค่า PSNR ที่เกิดจากการใส่สัญญาณรบกวน 10% 25% และ 50% จะอยู่ที่ 31.14 21.02 และ 18.33 ตามลำดับเมื่อเทียบกับภาพต้นฉบับ นอกจากนี้เห็นได้ชัดว่าภาพที่ถูกตัดสัญญาณรบกวนด้วยโมเดล ROF มีคุณภาพดีเมื่อภาพที่นำมาตัดสัญญาณรบกวนมีปริมาณของสัญญาณรบกวนอยู่ระหว่าง 10%-25%

บทที่ 6

การประยุกต์ใช้อัลกอริทึมเกาซ์-เซเดลเชิงขนานในการตัดสัญญาณรบกวนภาพประเภทบวกและคูณ

ในบทนี้จะใช้อัลกอริทึมเกาซ์-เซเดลเชิงขนานที่พัฒนาได้ในการตัดสัญญาณรบกวนบนภาพที่มีสัญญาณรบกวนทั้งประเภทบวกและคูณ ซึ่งผู้วิจัยได้เลือกนำปัญหาสัญญาณรบกวนบนภาพสามประเภทคือ ภาพถ่ายดาวเทียมขนาดใหญ่ ภาพอัลตราซาวน์ และภาพที่ได้จากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดมาทดลอง โดยปัญหาแต่ละประเภทนั้นมีลักษณะของปัญหาแตกต่างกัน การที่จะนำอัลกอริทึมเกาซ์-เซเดลมาใช้กับงานเหล่านี้ได้นั้นจะต้องคำนึงถึงหลายปัจจัยเช่น ขนาดของปัญหา ขนาดของหน่วยความจำ และจุดเด่นของภาพที่ต้องการนำมาตัดสัญญาณรบกวน

1. การตัดสัญญาณรบกวนบนภาพถ่ายดาวเทียมขนาดใหญ่

1.1 ลักษณะของปัญหา

สิ่งที่ต้องคำนึงในการตัดสัญญาณรบกวนบนภาพถ่ายดาวเทียมขนาดใหญ่มีดังนี้

1. ภาพถ่ายดาวเทียมนั้นเป็นภาพสีที่มีขนาดใหญ่มาก (ขนาดมากกว่า $3\times8,192\times8,192$) ซึ่งเมื่อแยกช่องสีแล้วอาจไม่สามารถบรรจุลงหน่วยความจำหลักและหน่วยความจำ

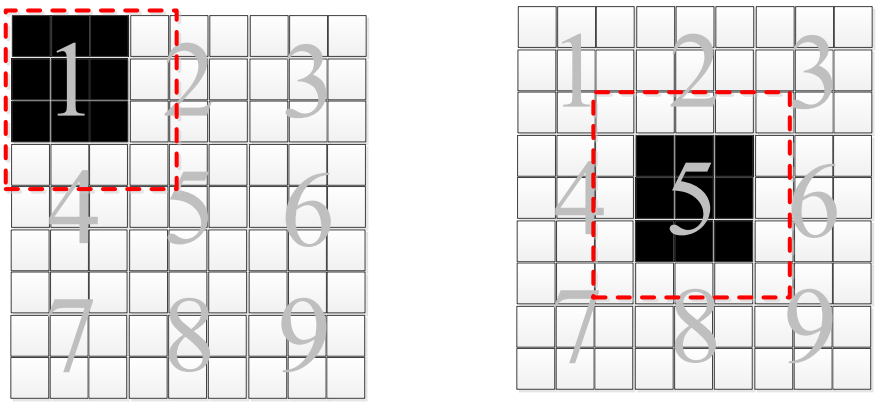
2. ภาพถ่ายดาวเทียมนั้นประกอบด้วยสัญญาณรบกวนทั้งสองชนิดคือสัญญาณรบกวนประเภทบวกและคูณ ซึ่งสามารถเขียนได้ตามสมการ

$$z = u + k_0\eta + k_1 u\xi \quad , \tag{6.1}$$

โดยที่ z คือภาพที่มีสัญญาณรบกวน u คือภาพที่ปราศจากสัญญาณรบกวน η และ ξ คือสัญญาณรบกวนประเภทบวกและประเภทคูณบนภาพที่มีการแจกแจงแบบปกติและแบบแกมมาตามลำดับ และ k_0 กับ k_1 คือพารามิเตอร์เพื่อควบคุมปริมาณสัญญาณรบกวนประเภทบวกและคูณตามลำดับ

1.2 การดำเนินงาน

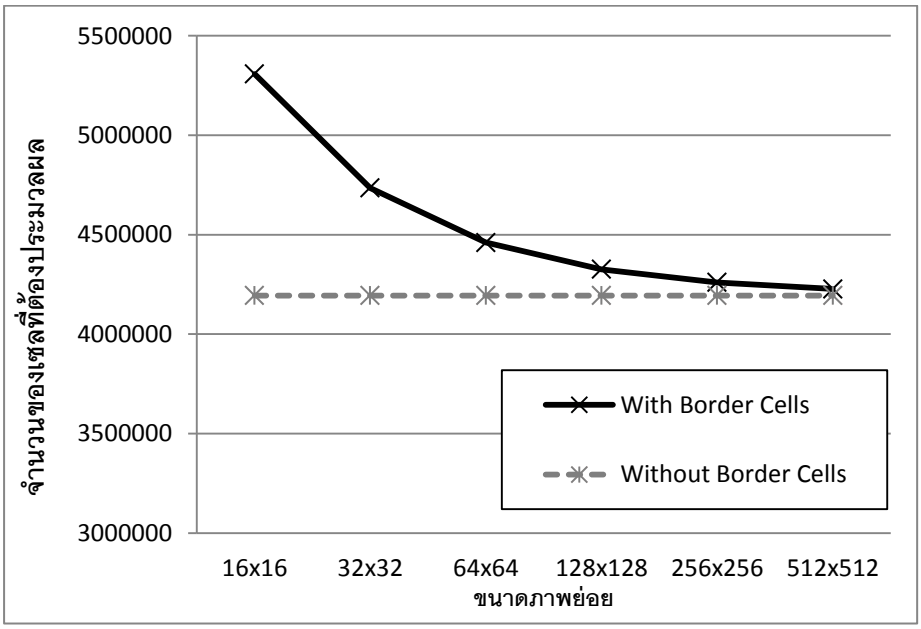
อัลกอริทึม QL-SWGS จะถูกนำมาใช้เพื่อแก้ระบบสมการในการตัดสัญญาณรบกวนออกจากภาพถ่ายดาวเทียม เนื่องจากภาพถ่ายดาวเทียมขนาดใหญ่นำมาประมวลผลโดยใส่ลงหน่วยความจำของหน่วยประมวลผลกราฟิกส์ทั้งหมดไม่ได้ ผู้วิจัยจึงแก้ปัญหาโดยการตัดภาพออกเป็นส่วนเล็กๆ (Chunk) และประมวลผลทีละส่วน เมื่อประมวลผลเสร็จจึงบันทึกกลับลงภาพใหญ่ โดยการแบ่งนั้นต้องอาศัยขอบนอกใช้ในการคำนวณตัวดำเนินการเชิงอนุพันธ์ ภาพที่ 6.1 แสดงวิธีการแบ่งส่วนมุม และส่วนกลางภาพ



(ก) (ข)

ภาพที่ 6.1 ตัวอย่างการแบ่งภาพออกเป็นส่วนย่อยๆ ขนาด 3×3 (สีดำ) ในส่วนมุมของภาพ (ก.) และส่วนกลางภาพ (ข.)

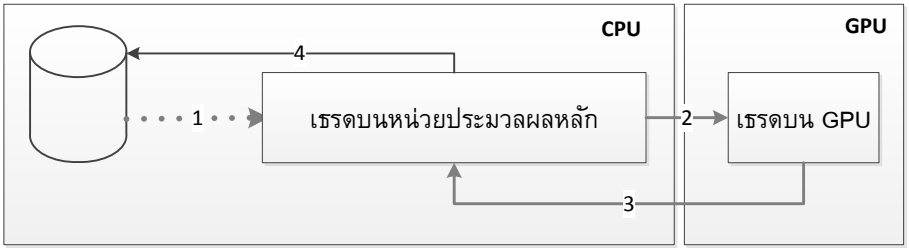
จากภาพที่ 6.1 จะเห็นได้ว่าการแบ่งภาพนั้นจะมีส่วนของขอบที่ต้องเพิ่มเข้ามา ในกรณีของภาพย่อยขนาดเล็ก อาจทำให้เกิดปริมาณเซลที่ต้องคำนวณเพิ่มถึง 77.7% ในกรณีของภาพย่อยขนาด 3×3 ขนาดของภาพย่อยต้องเหมาะสม ภาพที่ 6.2 แสดงกราฟจำนวนเซลที่ต้องคำนวณต่อขนาดภาพย่อย



ภาพที่ 6.2 ปริมาณเซลที่ใช้คำนวณของภาพย่อยที่แบ่งขนาดต่างๆ

จะเห็นได้ว่า การแบ่งภาพย่อย (With Border Cells) ที่มีขนาดใหญ่ขึ้น จะลดอัตราส่วนของเซลที่ต้องคำนวณเมื่อเทียบกับเซลที่ใช้จริง (Without Border Cells) ได้ แต่การเพิ่มขนาดภาพย่อยนั้นทำให้ต้องใช้หน่วยความจำมากขึ้น ผู้วิจัยเลือกขนาดภาพย่อยเป็น 512×512 เนื่องจากมีส่วนที่ต้องคำนวณเพิ่มเพียง 0.78%

ในด้านการประมวลผล โดยทั่วไปแล้วการใช้หน่วยประมวลผลกราฟิกส์ในการประมวลผลนั้นสามารถแสดงเป็นแผนผังได้ดังภาพที่ 6.3



ภาพที่ 6.3 การประมวลผลภาพโดยใช้หน่วยประมวลผลกราฟิกส์โดยทั่วไป

จากภาพที่ 6.3 นั้น การประมวลผลภาพถ่ายดาวเทียมขนาดใหญ่ที่ได้กล่าวมาในหัวข้อ 1.1 ของบทนี้นั้นไม่มีประสิทธิภาพหรืออาจทำไม่ได้เมื่อใช้วิธีตามภาพที่ 6.3 เนื่องจากสาเหตุดังนี้

1. ภาพมีขนาดใหญ่ การนำภาพขนาดใหญ่บรรจุลงในหน่วยความจำของหน่วยประมวลผลกราฟิกส์นั้นอาจทำไม่ได้เนื่องจากหน่วยความจำของหน่วยประมวลผลกราฟิกส์นั้นมีจำกัดมาก ดังนั้นจึงต้องออกแบบการจัดการหน่วยความจำที่เหมาะสม

2. การที่หน่วยประมวลผลกลางทำหน้าที่เพียงส่งงานให้หน่วยประมวลผลกราฟิกส์นั้นเป็นการทำงานที่ไม่มีประสิทธิภาพเนื่องจากในปัจจุบันนั้นหน่วยประมวลผลหลักมีหลายแกน ซึ่งหากใช้งานเพียงแกนเดียวจะทำให้แกนที่เหลือว่างงานและไม่ได้ใช้งาน

3. การอ่านภาพจากดิสก์ เนื่องจากภาพขนาดใหญ่ต้องใช้งานอุปกรณ์ I/O (เช่น ฮาร์ดดิสก์) ปริมาณมากเพื่อดึงภาพเข้ามาเก็บในหน่วยความจำหลัก ซึ่งระหว่างอุปกรณ์ I/O ทำงานนั้นจะต้องมีการรอเพื่อให้อุปกรณ์ดังกล่าวทำงานเสร็จและส่งข้อมูลกลับมา ดังนั้นการจัดการการรอคอยอุปกรณ์เหล่านี้ที่ดีจะทำให้การใช้งานหน่วยประมวลผลประเภทต่างๆ มีประสิทธิภาพมากขึ้น

เพื่อให้การออกแบบระบบเป็นไปได้ด้วยความแม่นยำ และมีประสิทธิภาพ ผู้วิจัยได้วัดเวลาของส่วนต่างๆในการตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียมจากหน่วยประมวลผลหลักเพียงตัวเดียว ได้ผลตามตารางที่ 6.1

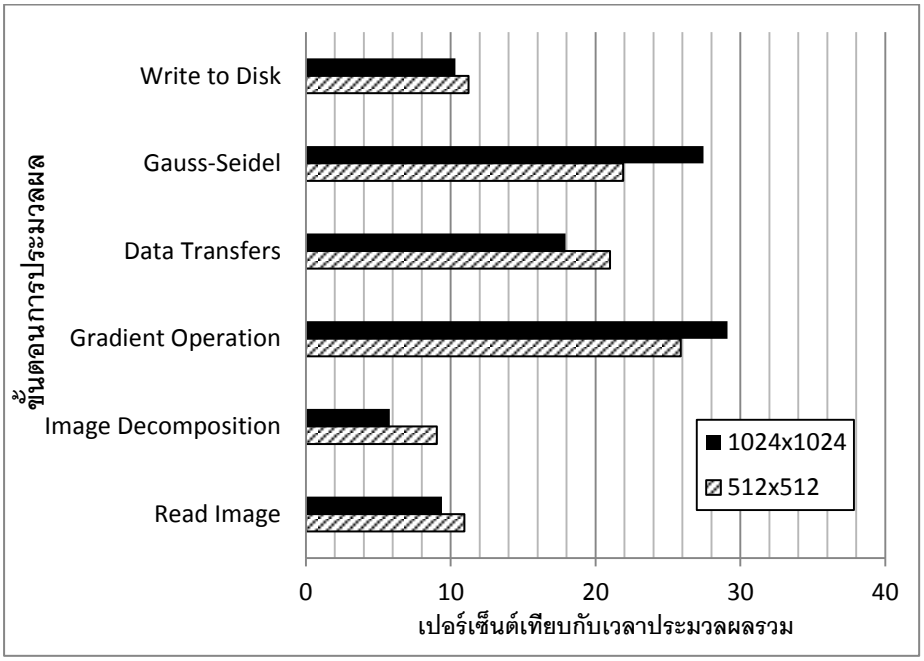
ตารางที่ 6.1 เวลาเฉลี่ยที่ใช้ในการประมวลผลในแต่ละส่วนของการตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียมขนาด 8192×8192 (1 ช่องสี) เป็นมิลลิวินาที

ขนาดภาพย่อย	**512x512**		**1024x1024**		**2048x2048**	
	เวลา (มิลลิวินาที)	**%**	**เวลา (มิลลิวินาที)**	**%**	**เวลา (มิลลิวินาที)**	**%**
การอ่านภาพ	60.5	10.95%	164.0	9.39%	468.5	8.02%
การแยกภาพย่อย	50.0	9.04%	101.0	5.78%	180.5	3.09%
ตัวดำเนินการเชิงอนุพันธ์	143.0	25.88%	508.0	29.11%	2044.0	35.01%
การขนส่งข้อมูล	116.0	20.99%	313.0	17.94%	987.5	16.91%

ตารางที่ 6.1 เวลาเฉลี่ยที่ใช้ในการประมวลผลในแต่ละส่วนของการตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียมขนาด 8192×8192 (1 ช่องสี) เป็นมิลลิวินาที (ต่อ)

ขนาดภาพย่อย	512x512		1024x1024		2048x2048	
	เวลา (มิลลิวินาที)	%	เวลา (มิลลิวินาที)	%	เวลา (มิลลิวินาที)	%
การประมวลผลบน GPU	121.0	21.90%	479.0	27.45%	1852.0	31.72%
การเขียนลงดิสก์	62.0	11.22%	180.0	10.31%	305.0	5.22%

จากตารางที่ 6.1 จะเห็นได้ว่าการทำงานหลักจะอยู่ที่การคำนวณตัวดำเนินการเชิงอนุพันธ์ ซึ่งดำเนินการประมวลผลบน GPU ในกรณีที่ใช้ภาพย่อยขนาดใหญ่ สัดส่วนของการทำงานทั้งสองส่วนนี้เมื่อเทียบกับเวลาทำงานโดยรวมจะเพิ่มขึ้น ตารางที่ 6.4 แสดงกราฟแท่งของภาระการประมวลผลการตัดสัญญาณรบกวนออกจากภาพถ่ายดาวเทียมขนาด 8192×8192 ในขั้นตอนต่างๆ



ภาพที่ 6.4 กราฟแท่งแสดงภาระการประมวลผลการตัดสัญญาณรบกวนออกจากภาพถ่ายดาวเทียมในขั้นตอนต่างๆ

ในการปรับปรุงเพิ่มประสิทธิภาพในการทำงานผู้วิจัยได้เลือกวิธีการทำงานแบบสายท่อ (Pipeline) เข้ามาเนื่องจากมีจุดเด่น และเหมาะสมกับปัญหาในด้านต่างๆ ดังนี้

1. เนื่องจากมีการทำงานร่วมกันทั้งสองสถาปัตยกรรมคือสถาปัตยกรรมของหน่วยประมวลผลกลาง และสถาปัตยกรรมของหน่วยประมวลผลกราฟิกส์ การทำงานจึงต้องมีการส่งข้อมูลไปมาระหว่างกันทำให้สูญเสียเวลาในการส่งข้อมูล รวมทั้งใช้หน่วยประมวลผลทุกแกนได้อย่างไม่มีประสิทธิภาพ ซึ่งการทำงานแบบสายท่อนั้น สามารถทำให้เกิดการทำงานแบบเกยทับ (Overlapping) ของการส่งข้อมูลซึ่งจะซ่อนเวลา (Latency Hiding) ในการส่งข้อมูล อีกทั้งยังสามารถแบ่งงานใหญ่ออกเป็นงานย่อยๆ เพื่อให้หน่วยประมวลผลอื่นทำงานได้อีกด้วย

2. ในการอ่านและบันทึกภาพขนาดใหญ่นั้นจะใช้อุปกรณ์ I/O มาก การแบ่งการทำงานออกเป็นชั้นๆ ด้วยวิธีการสายท่อนั้นสามารถซ่อนเวลาที่ใช้ในการรออุปกรณ์ I/O ได้มาก

ผู้วิจัยได้แบ่งการทำงานออกเป็นส่วนๆ ทำงานบนหน่วยประมวลผลต่างๆ ผู้วิจัยได้แบ่งเธรดการทำงานออกเป็น 4 ประเภทหลักๆ ดังนี้

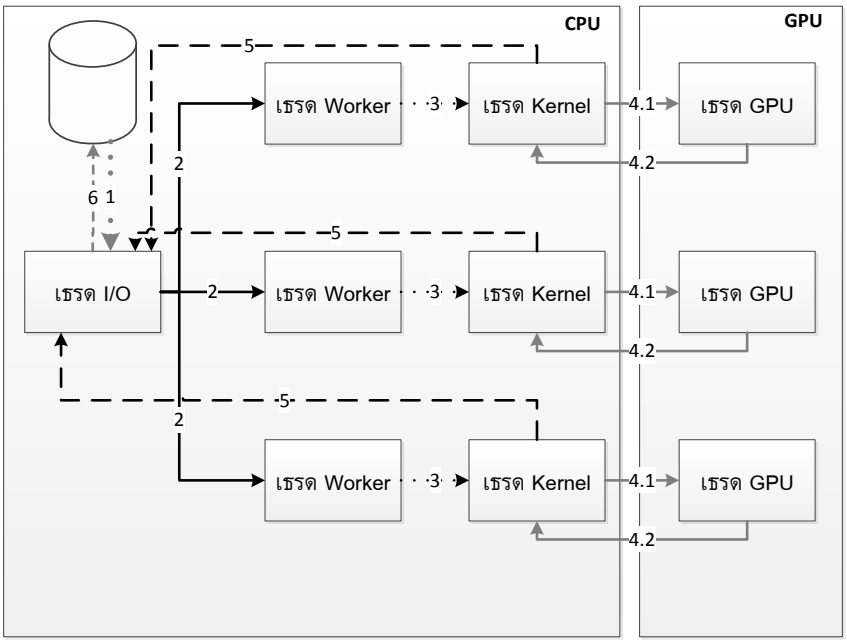
1. เธรด I/O ทำหน้าที่ในการดึงข้อมูลจากดิสก์และเก็บไว้บนหน่วยความจำหลัก แบ่งภาพหลักออกเป็นภาพย่อยๆ เพื่อให้เธรดประเภทอื่นนำไปใช้ต่อไป และเมื่อมีการประมวลผลการตัดสัญญาณรบกวนบนภาพเสร็จสิ้น จะมีหน้าที่บันทึกภาพลงดิสก์

2. เธรด Worker ทำงานบนหน่วยประมวลผลหลัก มีหน้าที่คำนวณตัวดำเนินการเชิงอนุพันธ์ก่อนที่จะส่งให้เธรด Kernel ต่อไป

3. เธรด Kernel ทำงานบนหน่วยประมวลผลหลัก ทำหน้าที่ส่งข้อมูลที่ได้จากเธรด Worker ไปยังหน่วยความจำของหน่วยประมวลผลกราฟิกส์ หลังจากนั้นจึงสั่งหน่วยประมวลผลกราฟิกส์ทำงานและรอรับข้อมูลที่หน่วยประมวลผลเสร็จจากหน่วยประมวลผลกราฟิกส์และส่งไปยังเธรด I/O เพื่อบันทึกส่วนของภาพย่อยลงภาพหลัก

4) เธรด GPU ทำงานบนหน่วยประมวลผลกราฟิกส์สำหรับทำการตัดสัญญาณรบกวนบนภาพในลักษณะจุดตรึงทำซ้ำ ซึ่งใช้วิธีการเกาซ์-เซเดล แบบสถานะที่ได้พัฒนาในบทที่ 4 ในการทำงาน เมื่อทำงานเสร็จจะส่งข้อมูลกลับไปยังเธรด Kernel

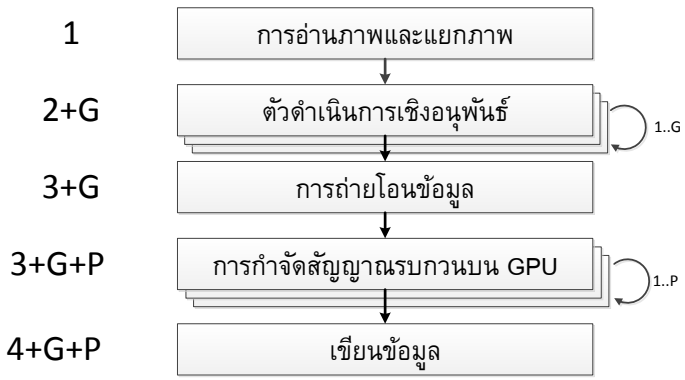
แผนผังการทำงานของทั้งหมดตามที่กล่าวมาสามารถเขียนได้ดังภาพที่ 6.5



ภาพที่ 6.5 แผนผังการทำงานแบบสายท่อที่ได้ออกแบบ

การทำงานจะเริ่มจาก I/O เธรดดึงภาพจากดิสก์ (1) หลังจากนั้นจึงส่งให้เธรด Worker (2) และเมื่อเธรด Worker ประมวลผลตัวดำเนินการเชิงอนุพันธ์เสร็จจึงส่งให้เธรด Kernel (3) ซึ่งมีหน้าที่ส่งและรับข้อมูลระหว่าง CPU และ GPU (4.1 และ 4.2) เมื่อ GPU ประมวลผลเสร็จแล้วจึงส่งกลับไปยังเธรด I/O (5) เพื่อบันทึกลงดิสก์ (6) ต่อไป

จากภาพที่ 6.5 สามารถเขียนเป็นลำดับการทำงานแบบสายท่อได้ดังภาพที่ 6.6 โดย G หมายถึงจำนวนขั้นตอนการทำงานของของตัวดำเนินการเชิงอนุพันธ์ และ P หมายถึงจำนวนขั้นตอนของงานที่ทำบนหน่วยประมวลผลกราฟิกส์



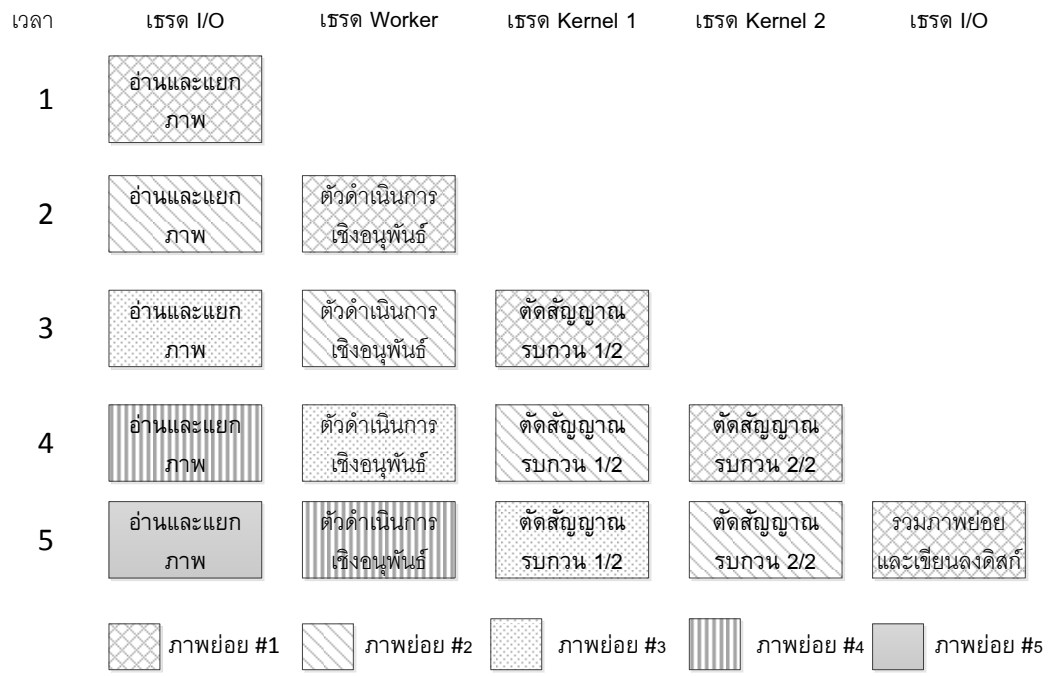
ภาพที่ 6.6 แผนผังการทำงานแบบสายท่อที่ออกแบบ

หลังจากนั้นผู้วิจัยจึงแบ่งส่วนต่างๆ ของการทำงานแบบสายท่อใหม่ โดยให้เวลาแต่ละส่วนมีขนาดไล่เลี่ยกันเพื่อให้สามารถทำงานเกยทับกันได้สนิทมากขึ้น ส่วนที่เป็นตัวดำเนินการเชิงอนุพันธ์ และส่วนที่ตัดสัญญาณรบกวนภาพบนหน่วยประมวลผลกราฟิกส์นั้นสามารถตัดแบ่งออกเป็นส่วนย่อยๆ ให้เกยทับกับส่วนอื่นได้สนิทมากขึ้นได้ ตัวอย่างจากการแบ่งนั้นแสดงในตารางที่ 6.2

ตารางที่ 6.2 การออกแบบการทำงานแบบสายท่อในชั้นต่างๆเพื่อให้การทำงานคร่อมกันสนิท

512×512		**1024×1024**	
ลำดับสายท่อ	**เวลา (%)**	**ลำดับสายท่อ**	**เวลา (%)**
อ่านจากดิสก์และแยกเป็นภาพย่อย	19.99	อ่านจากดิสก์และแยกเป็นภาพย่อย	15.07
ตัวดำเนินการเชิงอนุพันธ์	25.88	ตัวดำเนินการเชิงอนุพันธ์ (ชิ้นแรก)	14.55
การคัดลอกและส่งข้อมูล	20.99	ตัวดำเนินการเชิงอนุพันธ์(ชิ้นที่ 2)	14.56
การประมวลผลบนหน่วยประมวลผลกราฟิกส์	21.90	การคัดลอกและส่งข้อมูล	17.93
การเขียนลงดิสก์	11.22	การประมวลผลบนหน่วยประมวลผลกราฟิกส์ (ชิ้นแรก)	13.52
		การประมวลผลบนหน่วยประมวลผลกราฟิกส์ (ชิ้นที่สอง)	13.20
		การเขียนลงดิสก์	10.31

จากตาราง 6.2 ค่าเปอร์เซ็นต์แทนระยะเวลาที่ใช้ประมวลผลในแต่ละส่วนเทียบกับระยะเวลาประมวลผลรวม เมื่อนำไปเขียนแผนผังการทำงานตามช่วงระยะเวลาจะได้ดังภาพที่ 6.7



ภาพที่ 6.7 แผนผังการทำงานแบบสายท่อที่พัฒนาได้ (G=2, P=2) ตามเวลาที่ผ่านไป

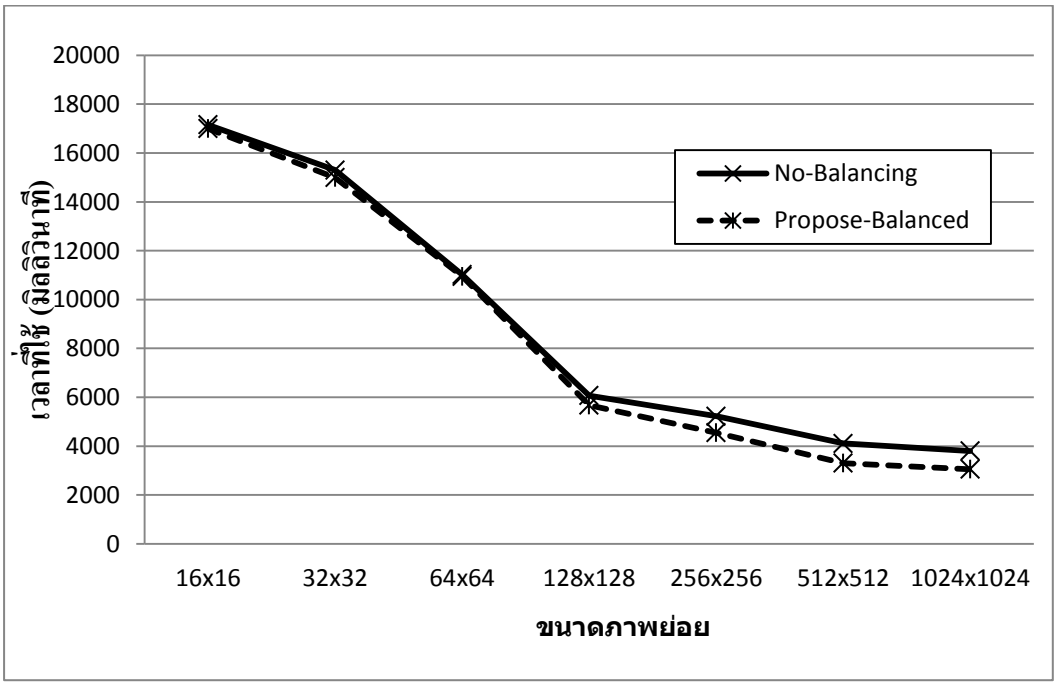
จากภาพที่ 6.7 ในขั้นตอนแรกเธรด I/O Thread จะอ่านส่วนของภาพจากดิสก์และแบ่งเป็นภาพย่อย (Time=1) ในเวลาถัดมา (Time=2) เธรด Worker จะเริ่มประมวลผลตัวดำเนินการเชิงอนุพันธ์ของภาพที่แบ่งได้จากขั้นตอนแรก ในขณะเดียวกันเธรด I/O จะดึงภาพจากดิสก์เพิ่มเติม และในเวลาถัดมา (Time=3) ภาพที่ได้จากการประมวลผลตัวดำเนินการเชิงอนุพันธ์จะส่งให้ GPU ทำงานในส่วนแรก ในขณะที่เธรด Working ทำตัวดำเนินการเชิงอนุพันธ์ให้ภาพถัดไป และ I/O เธรดดึงภาพใหม่จากดิสก์ในเวลาเดียวกัน

เวลาที่ Time=5 จะเห็นได้ว่าสายท่อถูกเติมเต็มด้วยงานซึ่งมีการแยกกันทำงานแบบขนานจนกระทั่งงานต่างๆ เสร็จสิ้น

1.3 ผลการดำเนินงาน

ผู้วิจัยใช้ภายถ่ายดาวเทียมจากดาวเทียม IKONOS ในโซนของดินแดง กรุงเทพมหานคร โดยบริษัท Space Imaging โดยการประเมินผลจะแบ่งออกเป็นสองส่วนคือการประเมินผลด้านเวลาและการประเมินผลคุณภาพของภาพที่ตัดสัญญาณรบกวน

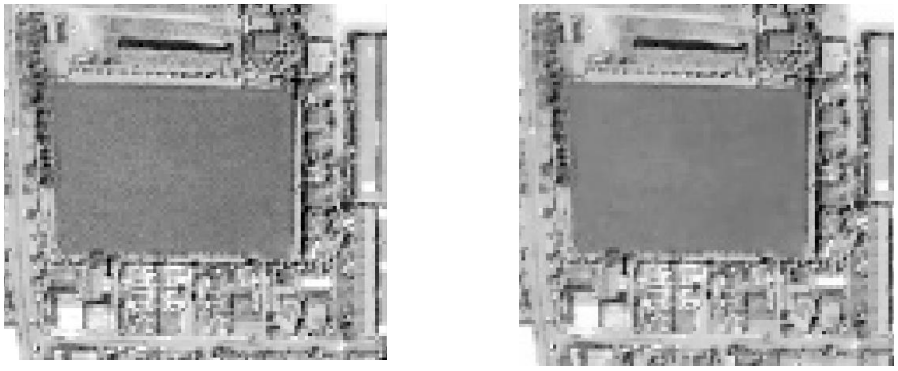
1.3.1 ประสิทธิภาพด้านเวลา ผู้วิจัยวัดประสิทธิภาพการทำงานการทำงานระหว่างการแบ่งงานแบบใหม่โดยให้แต่ละชั้นของการทำงานแบบสายท่อมีขนาดใกล้เคียงกัน กับการทำงานแบบเดิม ซึ่งได้ผลตามภาพที่ 6.8



ภาพที่ 6.8 กราฟเปรียบเทียบเวลาเฉลี่ยที่ใช้ในการประมวลผลระหว่างการประมวลผลที่มีการแบ่งงานใหม่ (Propose-Balanced) และการแบ่งงานแบบเดิมแยกตามขนาดภาพย่อย (No-Balancing)

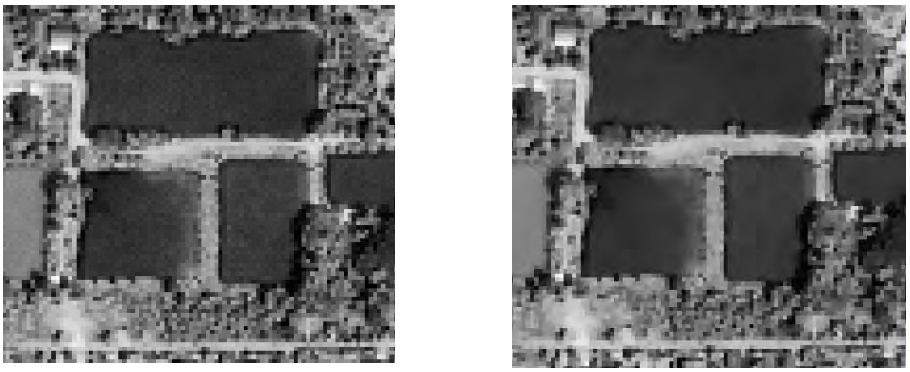
จากภาพที่ 6.8 จะเห็นได้ว่าการประมวลผลที่มีการแบ่งงานใหม่ (Propose-Balanced) นั้นสามารถประมวลผลได้เร็วกว่าแบบเก่า โดยจะเห็นชัดขึ้นเมื่อขนาดของภาพย่อยมีขนาดมากขึ้น โดยที่ภาพย่อยขนาด 1,024×1,024 นั้น จะประมวลผลได้เร็วกว่าประมาณ 23.01% เมื่อเทียบกับแบบเดิมที่มีขนาดภาพย่อยเท่ากัน

1.3.2 ด้านคุณภาพของภาพ ภาพถ่ายดาวเทียมผลลัพธ์ที่ได้จากการตัดสัญญาณรบกวนแสดงในภาพที่ 6.9, 6.10 และ 6.11



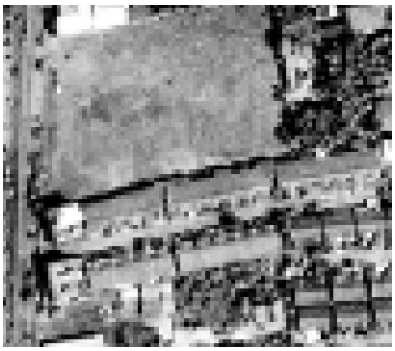
(ก) (ข)

ภาพที่ 6.9 ผลการตัดสัญญาณรบกวนของภาพตัวอย่างภาพแรก โดยก่อนตัดสัญญาณรบกวน (ก) และหลังตัดสัญญาณรบกวน (ข)

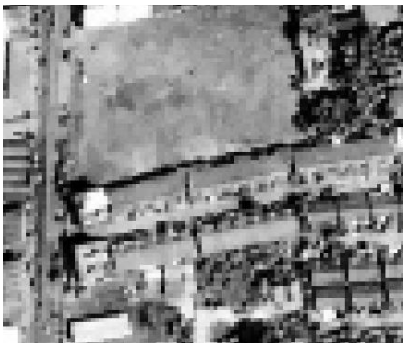


(ก) (ข)

ภาพที่ 6.10 ผลการตัดสัญญาณรบกวนของภาพตัวอย่างภาพที่สอง โดยก่อนตัดสัญญาณรบกวน (ก) และหลังตัดสัญญาณรบกวน (ข)



(ก)



(ข)

ภาพที่ 6.11 ผลการตัดสัญญาณรบกวนของภาพตัวอย่างภาพที่สาม โดยก่อนตัดสัญญาณรบกวน (ก) และหลังตัดสัญญาณรบกวน (ข)

จะเห็นได้ว่าสัญญาณรบกวนของภาพถ่ายดาวเทียมส่วนใหญ่จะถูกตัดออก และรายละเอียดต่างๆ บนภาพนั้นสามารถสังเกตได้ง่ายขึ้น

ผู้วิจัยได้ทำตารางแนะนำจำนวนเธรดที่เหมาะสมของหน่วยประมวลผลหลักและหน่วยประมวลผลกราฟิกส์ที่นำมาใช้ในการตัดสัญญาณรบกวนภาพถ่ายจากดาวเทียมรวมทั้งขนาดของหน่วยความจำที่ใช้ในแต่ละขนาดภาพย่อยในตารางที่ 6.3

ตารางที่ 6.3 ตารางแนะนำจำนวนเธรดของหน่วยประมวลผลประเภทต่างๆบนหน่วยประมวลผลกราฟิกส์และขนาดภาพย่อยที่แบ่งที่ต่างกัน รวมทั้งหน่วยความจำที่ใช้

	NVIDIA® Geforce 310M				**NVIDIA® GeForce GTX-480**			
	256×256	**512×512**	**1024×1024**	**2048×2048**	**256×256**	**512×512**	**1024×1024**	**2048×2048**
จำนวนเธรดหน่วยประมวลผลกราฟิกส์ต่อเคอร์เนล	16	16	16	16	512	512	256	128
จำนวนหน่วยประมวลผลหลักที่ใช้	1	1	2	2	2	2	4	4
จำนวนชั้นของสายท่อสำหรับ Gradient Operation	1	1	2	2	1	1	2	3
จำนวนชั้นของสายท่อบนหน่วยประมวลผลกราฟิกส์	1	1	1	1	1	1	2	3
ประมาณหน่วยความจำที่ต้องการรวมทุกเธรด	199kB	528kB	6.2 MB	25.2MB	199kB	528kB	9.4MB	46.2MB

1.3.3 การวิเคราะห์ความซับซ้อนในการประมวลผล ในการวิเคราะห์ความซับซ้อนในการประมวลผลและหน่วยความจำนั้นผู้วิจัยกำหนดสัญลักษณ์ต่างๆ ดังนี้

N ขนาดของภาพในแต่ละมิติ

n ขนาดของภาพย่อยที่ถูกแบ่งในแต่ละมิติ

T จำนวนของเธรดที่ใช้

F จำนวนไบต์ที่ใช้ในการเก็บเลขจำนวนจริง

I จำนวนไบต์ที่ใช้ในการเก็บเลขจำนวนเต็ม

G จำนวนการ์ดหน่วยประมวลผลกราฟิกส์ที่ใช้

L_{CPU} จำนวนชั้นของสายท่อบนหน่วยประมวลผลหลัก

L_{GPU} จำนวนชั้นของสายท่อบนหน่วยประมวลผลกราฟิกส์

ในการประมวลผลนั้น ภาพย่อยที่ถูกแบ่งออกมาจะต้องเก็บขอบไว้ทุกด้าน โดยการประมวลผลทั้งหมดสามารถเขียนเป็นสมการความซับซ้อนในการประมวลผลคือ

$$O(\frac{N^2}{n^2}(n+2)^2) \qquad (6.1)$$

จากสมการ (6.1) นั้นจะเห็นได้ว่าเมื่อลดขนาดภาพย่อยจำนวนครั้งในการประมวลผลจะเพิ่มขึ้น ซึ่งเกิดจากการทำงานคำนวณเชิงอนุพันธ์กับขอบที่เพิ่มขึ้นมา ดังนั้นการเลือกขนาดภาพย่อยที่เหมาะสมจึงมีความสำคัญในการปรับแต่งการทำงานให้มีประสิทธิภาพสูงสุด

ในส่วนหน่วยความจำที่ใช้ที่ต้องการใช้ในการประมวลผลของแต่ละภาพย่อยคือ

$$O(F(n+2)^2) \qquad (6.2)$$

บนหน่วยประมวลผลหลักนั้น ภาพที่อยู่ในหน่วยความจำจะใช้เนื้อที่ตามจำนวนชั้นของสายท่อ สามารถเขียนสมการความซับซ้อนได้ตามสมการ (6.3)

$$O(L_{\text{CPU}}F(n+2)^2) \tag{6.3}$$

บนหน่วยประมวลผลกราฟิกส์นั้น ในกรณีที่มีจำนวนการ์ดประมวลผลกราฟิกส์ G ใบบนเครื่อง หน่วยความจำที่ต้องการสามารถเขียนเป็นสมการความซับซ้อนได้ตามสมการ (6.4)

$$O(\frac{1}{G}L_{GPU}F(n+2)^2) \tag{6.4}$$

จะสังเกตได้ว่าสมการที่ (6.3) และ (6.4) นั้น ทุกค่าคือ G, L_{CPU}, L_{GPU}, F และ n จะเป็นค่าคงที่ ดังนั้นวิธีการที่ผู้วิจัยเสนอนั้นจะแบ่งภาพใหญ่ออกเป็นภาพย่อย ทำให้ใช้หน่วยความจำคงที่ไม่ว่าขนาดภาพของต้นฉบับนั้นใหญ่เพียงใดก็ตาม ซึ่งสามารถแก้ปัญหาภาพใหญ่เกินกว่าที่จะใส่บนหน่วยความจำได้ โดยเมื่อขนาดภาพย่อย (n) เล็กลง หน่วยความจำที่ต้องการใช้ก็จะน้อยลงด้วย แต่อย่างไรก็ตาม การแบ่งภาพขนาดเล็กเกินไปจะเพิ่มจำนวนจุดที่ต้องการประมวลผลซึ่งแสดงในสมการ (6.1)

2. การตัดสัญญาณรบกวนบนภาพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด

2.1 ลักษณะของปัญหา

กลไกของกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้นมีลักษณะดังภาพที่ 3.17 ซึ่งแสดงอยู่ในบทที่ 3

จากภาพที่ 3.17 นั้น จะเริ่มจากปืนยิงอิเล็กตรอน (Electron Gun) ยิงลำอิเล็กตรอนแบบขนานเข้าสู่ตัวกล้อง ลำอิเล็กตรอนนั้นจะผ่านเลนส์ที่ทำจากแม่เหล็กเพื่อรวมลำอิเล็กตรอนให้มีขนาดเล็กมากส่องไปยังวัตถุที่ต้องการส่องในห้องเล็กๆ เมื่อลำอิเล็กตรอนกระทบวัตถุที่ต้องการศึกษาจะมีการสะท้อนโดยมีตัวตรวจจับอิเล็กตรอนคอยรับอิเล็กตรอนที่สะท้อนออกมาในระหว่างที่มีการยิงอิเล็กตรอน การทำงานเช่นนี้จึงเป็นในลักษณะสหนัย (Coherence) สัญญาณรบกวนนั้นจะมีทั้งแบบบวกคือขั้นตอนการบันทึกภาพ และประเภทคูณเกิดจากขั้นตอนการรับอิเล็กตรอนที่สะท้อนจากวัตถุที่ต้องการศึกษา

ผู้วิจัยจึงใช้โมเดลของการตัดสัญญาณรบกวนเช่นเดียวกับภาพถ่ายดาวเทียมคือ

$$z = u + k_0\eta + k_1 u\xi, \tag{6.5}$$

โดยที่ z คือภาพที่มีสัญญาณรบกวน, u คือภาพที่ปราศจากสัญญาณรบกวน, η และ ξ คือสัญญาณรบกวนประเภทบวกและประเภทคูณบนภาพตามลำดับ และ k_0 กับ k_1 คืออัตราส่วนสัญญาณรบกวนแบบบวกและอัตราส่วนสัญญาณรบกวนประเภทคูณตามลำดับ ซึ่งผู้วิจัยเลือกใช้โมเดลเดียวกับการตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียมขนาดใหญ่ในหัวข้อ 1.1

อย่างไรก็ตามภาพจุลชีพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้นมีลักษณะพิเศษที่แตกต่างออกไปจากภาพถ่ายดาวเทียมและภาพถ่ายประเภทอื่นๆดังนี้

4. ภาพถ่ายจากกล้องจุลทรรศน์อิเล็กตรอนนั้นมีขนาดค่อนข้างใหญ่ แต่เล็กกว่าภาพถ่ายดาวเทียมทั่วไปในหัวข้อ 1.1 ดังนั้นภาพถ่ายประเภทนี้สามารถบรรจุลงในหน่วยความจำหลักของเครื่องคอมพิวเตอร์ในปัจจุบันได้ แต่อย่างไรก็ตามเวลาที่ใช้ในการตัดสัญญาณรบกวนนั้นยังใช้เวลามาก

5. นักวิจัยในสาขาชีววิทยา และจุลชีววิทยานั้นมักจะสนใจบนตัวจุลชีพเป็นหลักและไม่สนใจพื้นหลัง นักวิจัยในด้านนี้มักมีการใช้โลหะหนักเช่นทองคำ แพลตินัม รวมถึงแกรไฟต์ในการย้อมจุลชีพเพื่อเพิ่มประสิทธิภาพในการสะท้อนอิเล็กตรอนทำให้ภาพในส่วนตัวจุลชีพนั้นดูสว่างกว่าพื้นหลังมาก

6. ในภาพหลายๆ ภาพที่ได้จากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้น ตัวจุลชีพมักอยู่กระจัดกระจายและอยู่บนภาพแบบหลวมๆ

จากลักษณะของปัญหานี้ ผู้วิจัยจึงได้ออกแบบวิธีการประยุกต์ใช้อัลกอริทึมเกาซ์-เซเดลแบบขนานเพื่อตัดสัญญาณรบกวนของภาพประเภทนี้โดยเฉพาะ ซึ่งจะกล่าวต่อไปในหัวข้อ 2.2

2.2 การดำเนินงาน

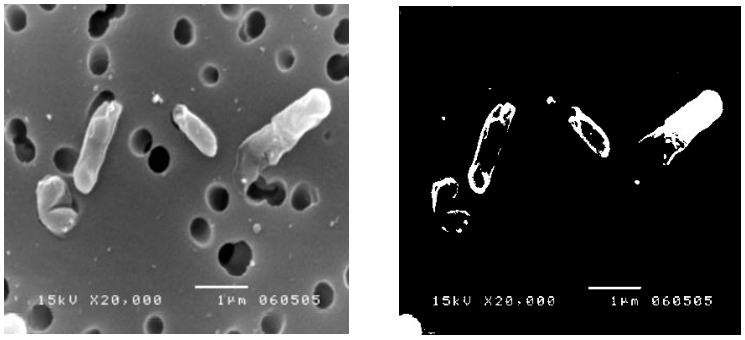
จากลักษณะของปัญหาที่กล่าวมาในหัวข้อ 2.1 นั้น ผู้วิจัยได้มีแนวทางในการปรับปรุงความเร็วในการประมวลผลการตัดสัญญาณรบกวนบนภาพประเภทนี้ดังนี้

1. เนื่องจากภาพจุลชีพนั้นจะถูกย้อมด้วยทองคำ ทำให้มีความสว่างมากเมื่อเทียบกับพื้นหลัง และนักชีววิทยาและจุลชีววิทยานั้นจะสนใจเฉพาะจุลชีพ การตัดพื้นหลังส่วนที่ไม่จำเป็นออกน่าจะทำให้เวลาที่ใช้ในการตัดสัญญาณรบกวนบนภาพชนิดนี้ลดลง

2. จุลชีพในบางภาพนั้นอยู่ห่างกันกระจัดกระจาย การที่แยกจุลชีพออกมาเป็นส่วนๆ นั้น สามารถนำแต่ละส่วนที่แยกออกมาได้ประมวลผลแบบขนานได้ ซึ่งน่าจะลดเวลาในการประมวลผลไปได้มาก

ผู้วิจัยได้ออกแบบระบบสำหรับการตัดสัญญาณรบกวนภาพชนิดนี้เป็นขั้นตอนดังนี้

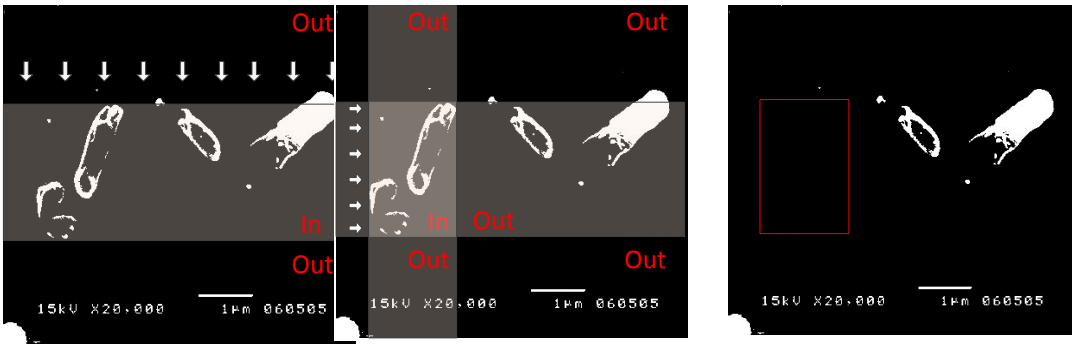
2.2.1 การแยกจุลชีพออกจากพื้นหลัง จุลชีพที่อยู่ในภาพนั้นมีการทำให้เกิดความสว่างโดยการเคลือบด้วยทองคำ เพื่อให้การตรวจจับจุลชีพในภาพทำได้โดยง่ายจึงมีการปรับภาพให้เป็นภาพขาวดำแบบสองบิตโดยใช้ค่าขีดแบ่ง (Threshold Value) จากความสว่าง (Intensity) บนภาพต้นฉบับโดยกำหนดค่าคงที่ไว้ หากพิกเซลใดบนภาพที่มีค่าเท่ากับหรือมากกว่าค่าที่กำหนดจะกำหนดให้เป็นสีขาว และหากต่ำกว่าค่าที่กำหนดก็จะกำหนดให้เป็นสีดำ ตามตัวอย่างภาพที่ 6.12



(ก) (ข)

ภาพที่ 6.12 ตัวอย่างภาพถ่ายที่มีการแปลงเป็นภาพขาวดำแบบสองบิตด้วยค่าขีดแบ่ง

เมื่อได้ภาพขาวดำแบบสองบิตแล้ว จึงทำการสแกนหาวัตถุ โดยสแกนจากบนลงล่าง และซ้ายไปขวาเพื่อหาสีขาวในภาพ หากพบสีขาวแล้วจะสแกนหาส่วนที่อยู่ติดกันและขยายกรอบออกให้เต็มส่วนที่พบ ดังภาพที่ 6.13



(ก) (ข) (ค)

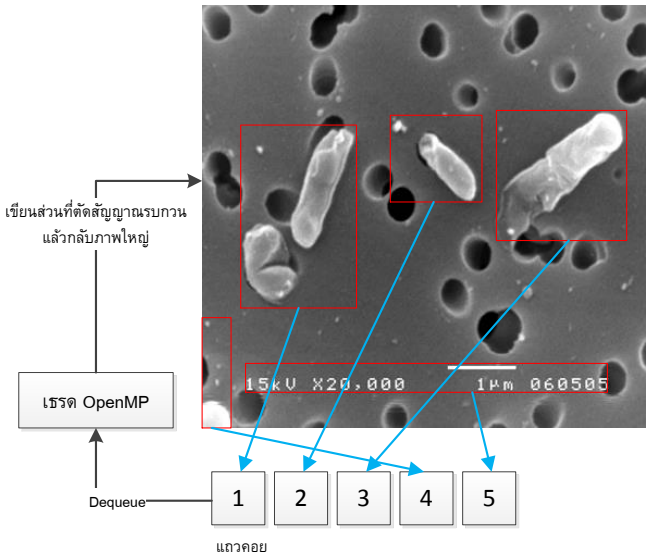
ภาพที่ 6.13 ขั้นตอนการตรวจหาพื้นที่ที่มีจุลชีพภายในภาพ

ในภาพที่ 6.13 จะเริ่มจากการค้นหาสีขาวในภาพขาวดำที่สร้างได้จากบนลงล่าง และจากซ้ายไปขวา (6.13ก) และเมื่อพบแล้วจะหาขอบเขตซ้าย และขวาในแนวดิ่ง และหลังจากนั้นก็จะทำซ้ายไปขวา ในลักษณะเดียวกัน (6.13ข) เมื่อได้แล้วจึงบันทึกพิกัดมุมบนขวาและล่างซ้าย พร้อมลบส่วนสีขาวในบริเวณที่พบให้เป็นสีดำ และทำไปจนกระทั่งไม่พบส่วนใหม่

แต่อย่างไรก็ดี จากภาพขาวดำยังมีจุดสีขาวเล็กๆ ซึ่งเป็นส่วนที่ไม่สนใจ ซึ่งจุดเหล่านี้ไม่ต้องพิจารณาโดยกำหนดขนาดของกรอบต่ำสุดไว้ ซึ่งในการทดลองนี้กำหนดไว้ขนาด 32×32 พิกเซล หากกรอบที่ตีได้มีขนาดต่ำกว่าที่กำหนดจะไม่นำมาพิจารณา เมื่อตีกรอบส่วนต่างๆในภาพแล้วจะได้ตามภาพที่ 6.14

ภาพที่ 6.14 ภาพผลลัพธ์หลังจากการแบ่งส่วนต่างๆของภาคเพื่อหาจุลชีพในภาพ (กรอบสีแดง)

2.2.2 การประมวลผลแบบขนานโดยแยกตามวัตถุที่พบ เมื่อสามารถตัดพื้นหลังส่วนใหญ่จากภาพ แต่ละส่วนที่พบจะถูกบันทึกมุมบนซ้ายและมุมล่างขวาตามพิกัดจริงของภาพหลัก

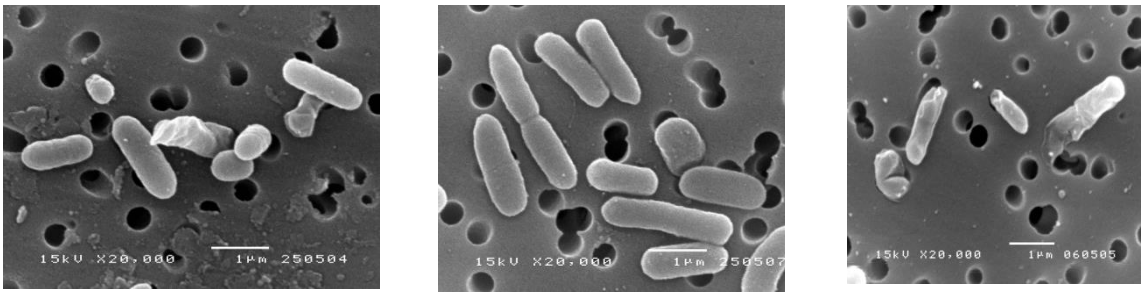


ภาพที่ 6.15 การทำงานแบบขนานโดยแยกตามกลุ่มของจุลชีพที่พบในภาพ

2.2.3 การใช้เกาซ์-เซเดลเชิงขนานกับภาพที่แบ่งได้ขนาดใหญ่ จากภาพตัวอย่าง ในภาพที่สองนั้นลักษณะของภาพตัวจุลชีพจะอยู่กันอย่างหนาแน่นซึ่งทำให้การแบ่งภาพด้วยวิธีจากหัวข้อก่อนหน้าไม่สามารถทำได้เนื่องจากเมื่อใช้วิธีดังกล่าว ภาพที่ได้จะออกมาเป็นภาพใหญ่เพียงภาพเดียว ดังนั้นผู้วิจัยจึงได้นำวิธีการเกาซ์-เซเดลแบบไร้แถวคอยที่พัฒนาได้ในบทที่ 4 เข้ามาใช้ ซึ่งได้กำหนดกฎว่า หากภาพที่ตรวจพบนั้นมีขนาดใหญ่เกิน 33% ของภาพจะใช้เกาซ์-เซเดลแบบไร้แถวคอย

2.3 ผลการดำเนินงาน

หลังจากใช้อัลกอริทึม QL-SWGS เพื่อแก้ระบบสมการในการตัดสัญญาณรบกวนออกจากภาพของจุลชีพร่วมกับวิธีการประมวลผลภาพเบื้องต้นที่กล่าวถึงในหัวข้อย่อยที่ผ่านมา ผู้วิจัยวัดผลสองด้านคือด้านประสิทธิภาพและคุณภาพของภาพหลังตัดสัญญาณรบกวน ในการทดลองนี้ผู้วิจัยได้ใช้ภาพของจุลชีพที่เป็นแบคทีเรีย *Pectobacterium carotovorum* สามภาพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดที่มีกำลังขยาย 20,000 เท่า ภาพต้นฉบับเป็นภาพขาวดำแบบ 8 บิต ซึ่งแสดงได้ตามภาพที่ 6.16

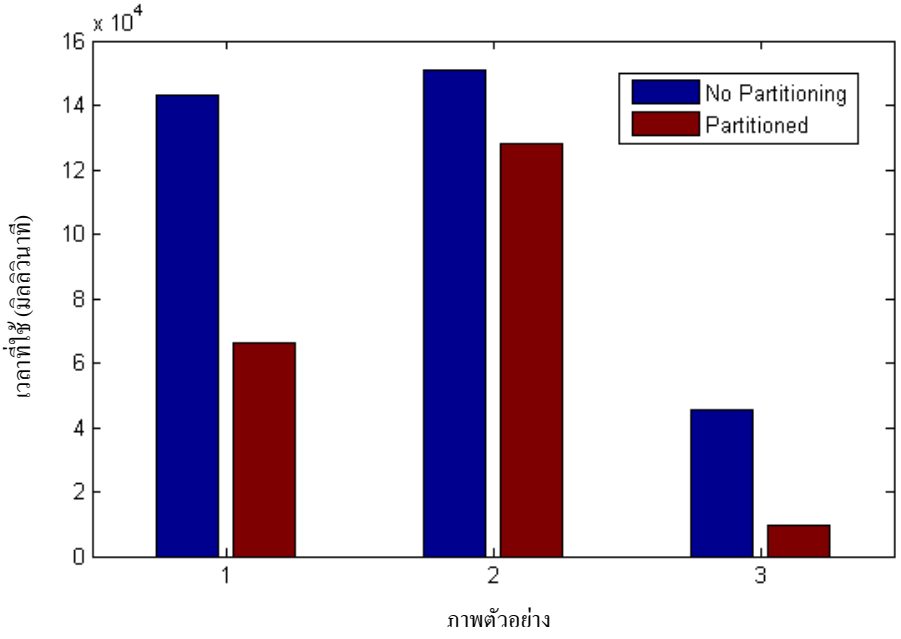


(ก) (ข) (ค)

ภาพ 6.16 ภาพของจุลชีพที่นำมาใช้ในการทดลอง

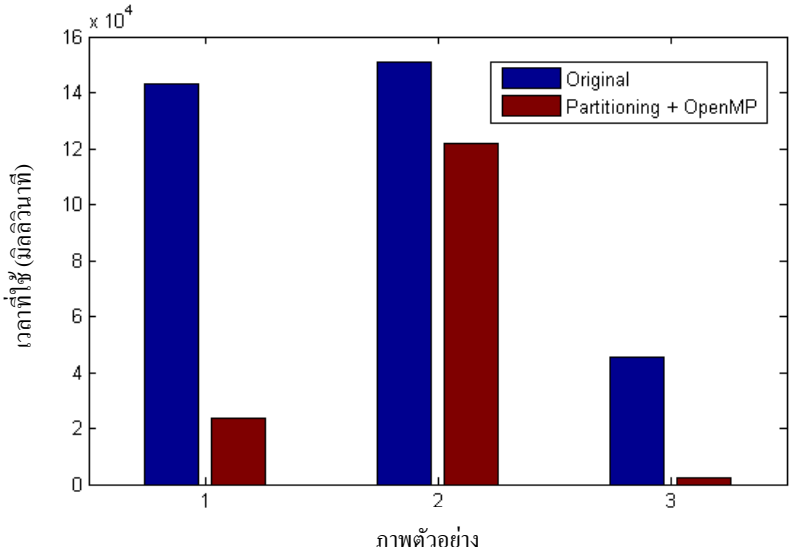
การทดลองนี้จะทดลองบนเครื่องคอมพิวเตอร์แบบสี่แกนประมวลผล (Intel® Core 2 Quad Q9300) ความเร็วสัญญาณนาฬิกาที่ 2.50 กิกะเฮิร์ต การวัดผลนั้นแบ่งออกเป็นสองด้านคือการวัดผลประสิทธิภาพของอัลกอริทึม และคุณภาพของภาพหลังจากตัดสัญญาณรบกวน

2.3.1 ประสิทธิภาพของอัลกอริทึม ผู้วิจัยได้ทดลองและจับเวลาการทำงานของการตัดสัญญาณรบกวนบนภาพขนาด 1,332×1,000 พิกเซล โดยได้ผลเวลาตามภาพที่ 6.17



ภาพที่ 6.17 เวลาที่ใช้ในการตัดสัญญาณรบกวนเป็นมิลลิวินาทีก่อน (No Partitioning) และหลังการแยกพื้นหลังออกจากตัวจุลชีพ (Partitioned)

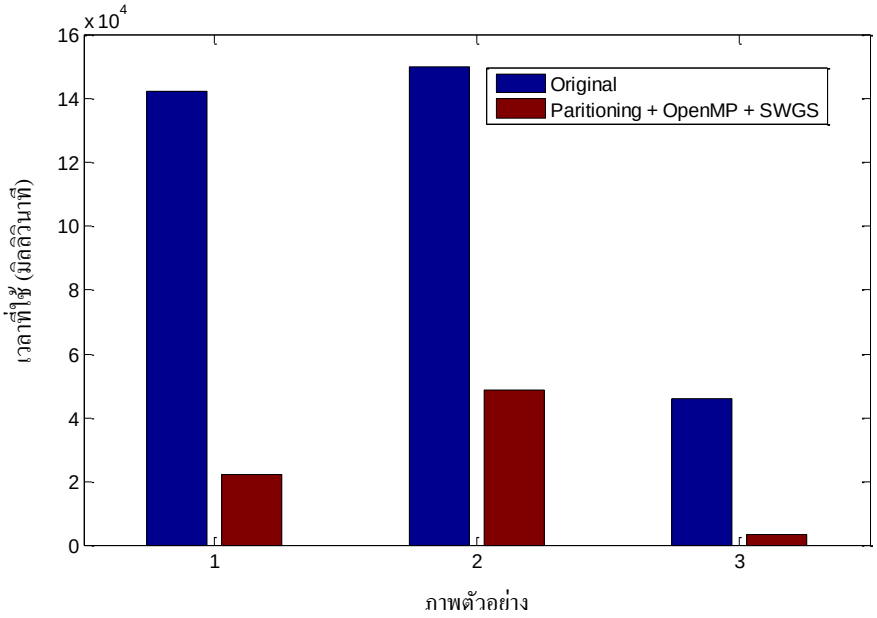
จากภาพที่ 6.18 แสดงให้เห็นว่าเมื่อตัดพื้นหลังซึ่งเป็นส่วนใหญ่ของภาพออกทำให้เวลาของการทำงานลดลงอย่างเห็นได้ชัดเจน หลังจากนั้นผู้วิจัยได้นำจุลชีพที่แยกออกจากพื้นหลังมาประมวลผลเชิงขนานโดยใช้โอเพ็นเอ็มพี



ภาพที่ 6.18 เวลาที่ใช้ในการตัดสัญญาณรบกวนเป็นมิลลิวินาทีก่อน (Original) และหลังการแยกพื้นหลังออกจากตัวจุลชีพ (Partitioning + OpenMP)

แต่อย่างไรก็ตามพบว่าภาพตัวอย่างภาพที่สองนั้นเวลาไม่ได้ลดลงเนื่องจากจุลชีพในภาพนั้นมีความหนาแน่นและการแบ่งภาพนั้นแบ่งได้ออกมาเพียงภาพเดียว ทำให้การประมวลผลแบบขนานโดยยึดตามภาพย่อยๆ นั้นไม่สามารถทำได้ ทางผู้วิจัยจึงได้ใช้อัลกอริทึมเกาซ์-เซเดลแบบไร้แถวคอยที่พัฒนาได้จากบทที่ 4 เข้ามาทำงานเมื่อการแบ่งภาพนั้นออกมาเพียงภาพเดียว หรือมีขนาดมากกว่า 33% ของพื้นที่โดยรวมภายในภาพหลัก

เมื่อใช้อัลกอริทึมเกาซ์-เซเดลแบบไร้แถวคอยแล้ว ผลด้านเวลาที่ได้แสดงในภาพที่ 6.19

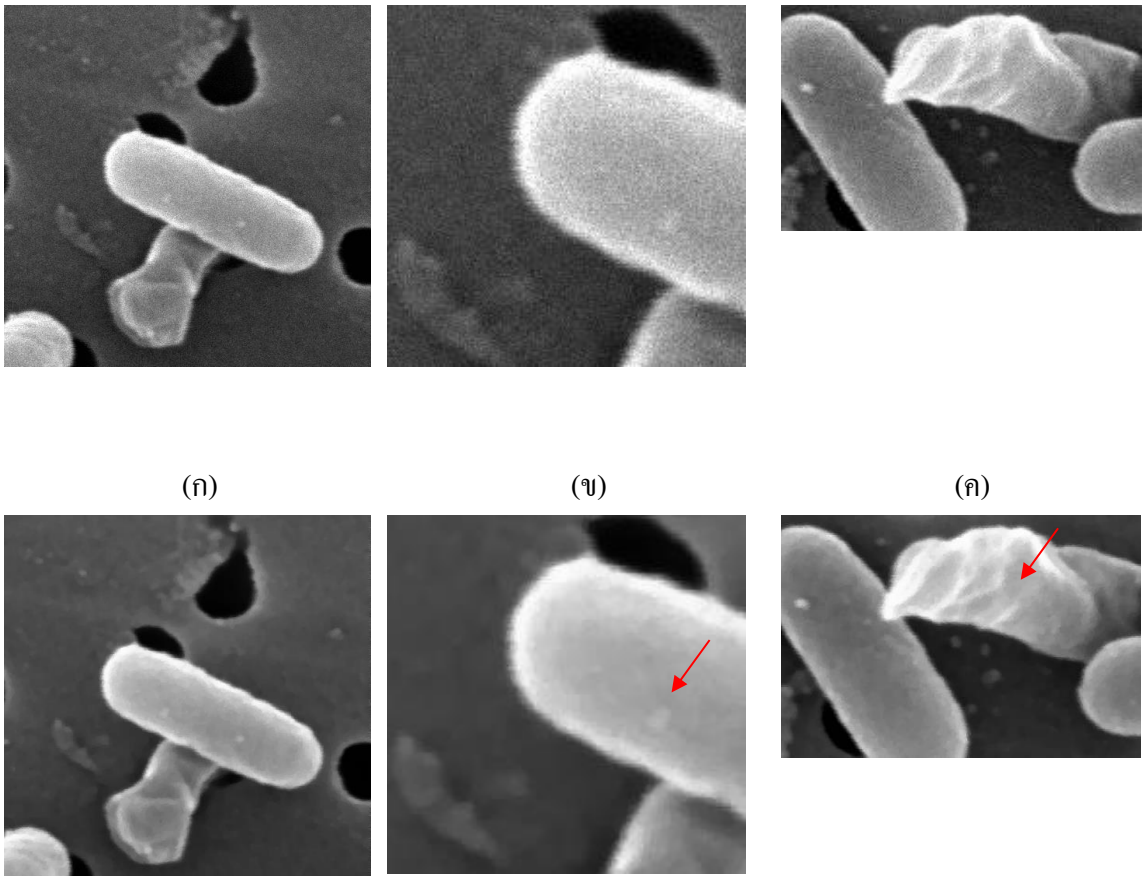


ภาพที่ 6.19 เวลาที่ใช้ในการตัดสัญญาณรบกวนก่อน (Original) และหลังใช้วิธีการต่างๆ ทั้งสามรวมกัน (Partitioning + OpenMP + SWGS)

จะเห็นได้ว่าภาพตัวอย่างที่สองนั้นใช้เวลาลดลงอย่างมาก โดยลดเวลาโดยรวมจากก่อนที่ใช้อัลกอริทึมเกาซ์-เซเดลแบบไร้แถวคอยได้ถึง 61% ในขณะที่ภาพแรก และภาพที่สาม เวลาที่ใช้จะไม่ลดลง

ภาพจุลชีพที่ได้จากการแบ่งส่วนและทำการประมวลผลแบบขนานนี้ สามารถลดเวลาการประมวลผลโดยรวมได้สูงสุดถึง 94.48% ในภาพที่สาม และลดเวลาการประมวลผลเฉลี่ยทั้งสามภาพได้ถึง 77.33% ซึ่งความเร็วในการประมวลผลที่เพิ่มขึ้นน่าจะช่วยลดเวลาที่นักชีววิทยาและนักจุลชีววิทยาใช้ในการวิเคราะห์ภาพถ่ายจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดได้มาก

2.3.2 คุณภาพของภาพหลังการตัดสัญญาณรบกวน ตัวอย่างบางส่วนของภาพที่ผ่านขั้นตอนการตัดสัญญาณรบกวนนั้นแสดงได้ตามภาพที่ 6.20



(ก) (ข) (ค)

(ง) (จ) (ฉ)

ภาพที่ 6.20 ภาพก่อน (6.20ก, 6.20ข, 6.20ค) และหลังการตัดสัญญาณรบกวน (6.20ง, 6.20จ, 6.20ฉ)

จากภาพที่ 6.20 ลูกศรใน 6.20จ ชี้แสดงให้เห็นโครงสร้างของวัตถุแปลกปลอมที่เกาะบนตัวแบคทีเรีย *Pectobacterium carotovorum* ซึ่งมองเห็นยากในภาพที่มีสัญญาณรบกวน และ 6.20ฉ ตรงที่ลูกศรสีแดงชี้แสดงให้เห็นถึงมิติของไซโทพลาซึมที่มีการรั่วไหลออกมาจากเซล ภาพที่ตัดสัญญาณรบกวนแล้วนั้นมีค่า PSNR เป็น 37.37dB เมื่อเทียบกับภาพต้นฉบับที่มีสัญญาณรบกวน

ภาพที่ได้จากการทดลองนี้ นักจุลชีววิทยาจะนำไปใช้ในการวิเคราะห์การทดลองผลของไฟฟ้าบนแบคทีเรีย *Pectobacterium carotovorum* ที่อิเล็กโตรไลต์ความเข้มข้นต่างๆ ต่อไป

2.3.3 การวิเคราะห์ความซับซ้อนในการประมวลผล ในการวิเคราะห์ความซับซ้อนในการประมวลผลและหน่วยความจำของการตัดสัญญาณรบกวนจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้นผู้วิจัยกำหนดสัญลักษณ์ต่างๆดังนี้

N ขนาดของภาพต้นฉบับในแต่ละมิติ

n_{avg} ขนาดเฉลี่ยของภาพย่อยที่ถูกแยกออกจากพื้นหลังในแต่ละมิติ

p จำนวนภาพย่อยที่ถูกแยกออกจากพื้นหลัง

d จำนวนการประมวลผลในการแยกจุลชีพออกจากพื้นหลัง

F จำนวนไบต์ที่ใช้ในการเก็บเลขจำนวนจริง

s เป็นระยะเวลารวมที่เธรดรอคอยเข้าทำงาน และเวลาในการผสานการทำงาน

T จำนวนเธรดที่ใช้ในการประมวลผล

ในการประมวลผลนั้น ในการประมวลผลแต่ละเธรดจะเขียนเป็นสมการความซับซ้อนในการประมวลผลคือ

$$O(\frac{1}{T}p(n_{avg})^2 + d) \tag{6.6}$$

จากสมการที่ (6.6) จะเห็นได้ว่าความเร็วในการประมวลผลนั้นขึ้นกับอัตราส่วนระหว่างภาพต้นฉบับและขนาดรวมของภาพย่อยของจุลชีพที่แบ่งออกมาได้ ซึ่งหากสามารถตัดภาพพื้นหลังออกได้มาก (ทำให้ n_{avg} ลดลง) ก็จะลดเวลาในการประมวลผลโดยรวมได้มากขึ้น ความเร็วที่เพิ่มขึ้นจากการแยกภาพจุลชีพออกจากพื้นหลังคือ

$$\text{Speedup} = \frac{1}{T} \cdot \frac{N^2}{p(n_{avg})^2} \tag{6.7}$$

แต่ในบางกรณีนั้นไม่สามารถแยกจุลชีพออกจากพื้นหลังได้เนื่องจากตัวจุลชีพปรากฏในภาพอย่างหนาแน่น จึงมีการนำอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยมาใช้ภาพย่อยที่แบ่งได้จะออกมาเพียงภาพเดียว จะได้สมการความซับซ้อนในการประมวลผล (6.8)

$$O(\frac{1}{T}(n_{avg})^2 + d + s) \tag{6.8}$$

โดยที่ n มีขนาดใกล้เคียงหรือเท่ากับ N

ส่วนของการวิเคราะห์หน่วยความจำที่ใช้ วิธีการที่ผู้วิจัยเสนอนั้นจะใช้หน่วยความจำเฉลี่ยในแต่ละหน่วยประมวลผลคือ

$$O(FN^2 + F(n_{avg})^2) \quad (6.9)$$

สมการ (6.9) นี้จะต่างกับการประมวลผลภาพถ่ายดาวเทียมขนาดใหญ่ โดยหน่วยความจำที่ใช้ขึ้นอยู่กับขนาดของภาพต้นฉบับและขนาดเฉลี่ยของภาพจุลชีพที่พบและแบ่งออกจากพื้นหลัง โดยส่วนของภาพต้นฉบับจะใช้เก็บเพื่อแยกภาพจุลชีพออก และส่วนภาพจุลชีพที่ถูกแบ่งออกมาจะใช้ในการประมวลผลการตัดสัญญาณรบกวน

บทที่ 7

การตัดสัญญาณรบกวนภาพเคลื่อนไหวแบบเรียลไทม์

ในบทนี้ ผู้วิจัยจะศึกษาการตัดสัญญาณรบกวนภาพเคลื่อนไหวแบบเรียลไทม์ โดยแบ่งการศึกษาออกเป็นสองกรณีคือ การตัดสัญญาณรบกวนประเภทบวกซึ่งเป็นการตัดสัญญาณรบกวนจากภาพเคลื่อนไหวที่ถ่ายโดยกล้องวีดีโอทั่วไป และการตัดสัญญาณรบกวนประเภทบวกและคูณซึ่งเป็นการตัดสัญญาณรบกวนจากภาพอัลตราซาวน์ที่ใช้ในการแพทย์

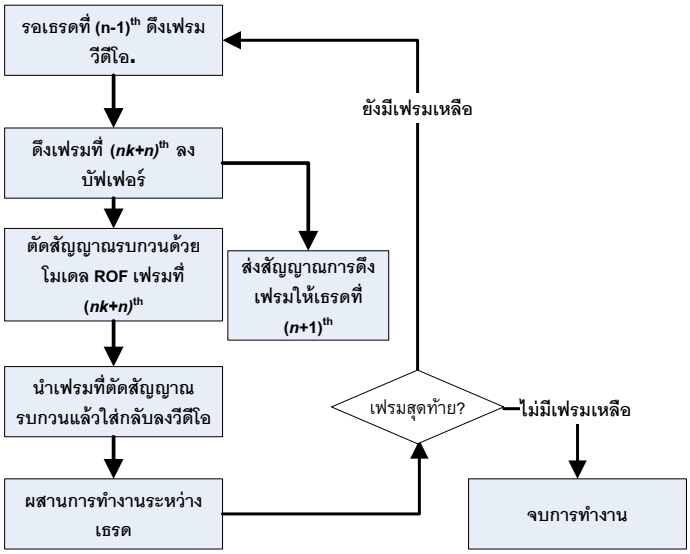
1. การตัดสัญญาณรบกวนประเภทบวกจากภาพเคลื่อนไหวเรียลไทม์

ในการตัดสัญญาณรบกวนประเภทบวกจากภาพเคลื่อนไหวแบบเรียลไทม์นั้นมีสิ่งที่ต้องคำนึงถึงคือลำดับเฟรมหลังการตัดสัญญาณรบกวนนั้นจะต้องคงเดิมและได้อัตราเฟรมที่ตัดสัญญาณรบกวนได้ต่อหนึ่งวินาที (เฟรมเรท) เป็นที่น่าพอใจ คือ 12 เฟรมต่อวินาทีขึ้นไป ผู้วิจัยได้เสนอวิธีการเพื่อให้ใช้หน่วยประมวลผลได้อย่างมีประสิทธิภาพมากที่สุดและพยายามซ่อนเวลาในการติดต่อกับอุปกรณ์ภายนอก (I/O Latency Hiding)

1.1 วิธีการดำเนินงาน

ผู้วิจัยได้ดำเนินการพัฒนาวิธีการตัดสัญญาณรบกวนจากภาพวีดีโอแบบเรียลไทม์เป็นลำดับขั้นตอนดังนี้

ในขั้นแรกสุดผู้วิจัยได้พัฒนาวิธีการแบบบล็อกขึ้นเพื่อทำการตัดสัญญาณรบกวนจากภาพเคลื่อนไหวในแบบขนาน ซึ่งวิธีการนั้นจะแสดงได้ดังภาพที่ 7.1



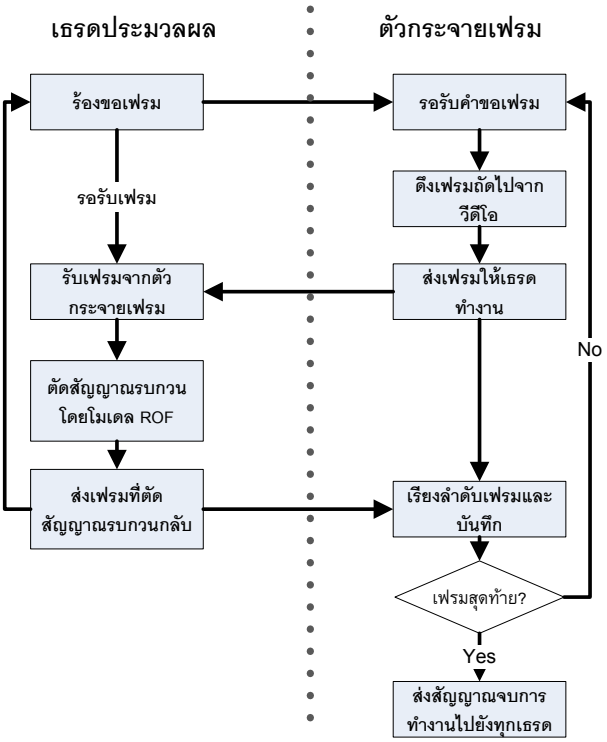
ภาพที่ 7.1 วิธีการทำงานแบบบล็อก

วิธีการในภาพที่ 7.1 จะเริ่มจากเธรดแรกจะรอรับเฟรมที่เข้ามาแบบสายน้ำ (Streaming) และเมื่อได้เฟรมครบจำนวนตามที่ต้องการแล้วจะเก็บลงในพื้นที่เก็บชั่วคราว (Buffer) และเมื่อส่งสัญญาณให้เธรดถัดไปเข้ามารอรับเฟรมชุดถัดไป ในส่วนของ “การตัดสัญญาณรบกวน” ตลอดการทดลองนี้จะใช้โมเดลสัญญาณรบกวนแบบ ROF [4] และทำโดยวิธีการทำซ้ำแบบจุดตรึงโดยใช้ QL-SWGS โดยทำซ้ำจำนวน 5 รอบต่อเฟรม

เมื่อแต่ละเธรดได้เฟรมตามจำนวนที่กำหนดแล้วจะเริ่มทำการตัดสัญญาณรบกวน และเฟรมที่ตัดสัญญาณรบกวนแล้วจะเขียนเรียงกลับลงพื้นที่เก็บชั่วคราวและกลับไปรอสัญญาณให้รับเฟรมจากเธรดอื่นๆ ต่อไป

แต่อย่างไรก็ตามวิธีการแบบบล็อกนั้นก็มีข้อเสียคือเมื่อใช้จำนวนเธรดที่มากขึ้น เธรดหลังๆ จะต้องมีการรอคอยเข้าทำงาน ซึ่งในระหว่างที่เธรดต่างๆ รอคอยที่จะเข้าทำงานนั้นจะว่างงาน ซึ่งเป็นการใช้ทรัพยากรที่ไม่มีประสิทธิภาพ ผู้วิจัยจึงได้พัฒนาวิธีการตัวกระจายเฟรมขึ้น (Frame Distributor) ซึ่งเธรดที่ทำหน้าที่เป็นตัวกระจายเฟรมนั้นมีหน้าที่รอรับเฟรมที่เข้ามา และเลือกเธรดที่ว่างงานเพื่อมอบหมายให้เธรดนั้นทำการตัดสัญญาณรบกวน เมื่อเธรดที่ได้รับมอบหมายงานได้ตัดสัญญาณรบกวนเสร็จสิ้นแล้วจะส่งเฟรมที่ตัดสัญญาณรบกวนกลับไปยังเธรดกระจายงาน และเธรดกระจายงานจะทำการเรียงเฟรมเพื่อแสดงผลต่อไป

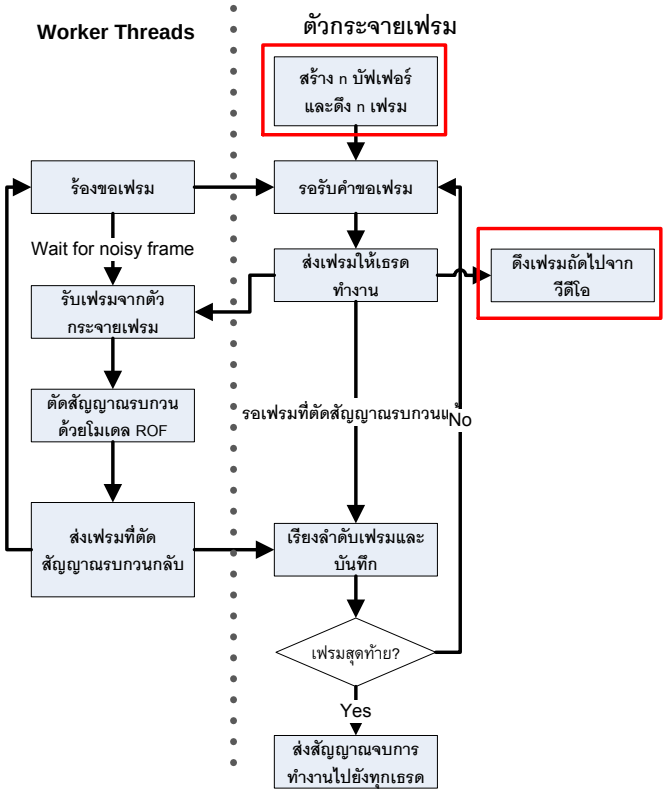
วิธีการใช้ตัวกระจายเฟรมนั้น สามารถแสดงได้ในภาพที่ 7.2



ภาพ 7.2 วิธีการทำงานแบบใช้ตัวกระจายเฟรม

การใช้ตัวกระจายเฟรมในภาพที่ 7.2 นั้นยังสามารถซ่อนเวลาในการรอรับภาพ และการจัดเรียงภาพและการแสดงผลได้อีกด้วย แต่อย่างไรก็ดี วิธีการใช้ตัวกระจายเฟรมนั้นยังสามารถเพิ่มประสิทธิภาพได้อีก ผู้วิจัยได้สังเกตว่าการรอรับเฟรมนั้นจะกินเวลาค่อนข้างมาก และเมื่อตัวกระจายเฟรมได้กระจายเฟรมที่ได้รับไปยังเธรดอื่นๆแล้วจะว่างงาน ผู้วิจัยจึงได้เพิ่มวิธีการอ่านล่วงหน้า (Read-Ahead) ให้กับเธรดที่ทำหน้าที่เป็นตัวกระจายเฟรม โดยเมื่อเธรดที่ทำหน้าที่เป็นตัวกระจายเฟรมแจกเฟรมให้เธรดอื่นทำงานหมดแล้วก็จะดึงเฟรมใหม่มาทันทีลงในพื้นที่เก็บชั่วคราว และเมื่อเธรดที่ทำงานเสร็จมีการร้องขอเฟรมนั้นก็จะให้เฟรมใหม่ที่เก็บบนพื้นที่ชั่วคราวนั้น ทำให้การทำงานของตัวกระจายเฟรมมีประสิทธิภาพมากขึ้น

การพัฒนาตัวกระจายเฟรมโดยใช้วิธีการอ่านล่วงหน้าแสดงได้ตามภาพที่ 7.3

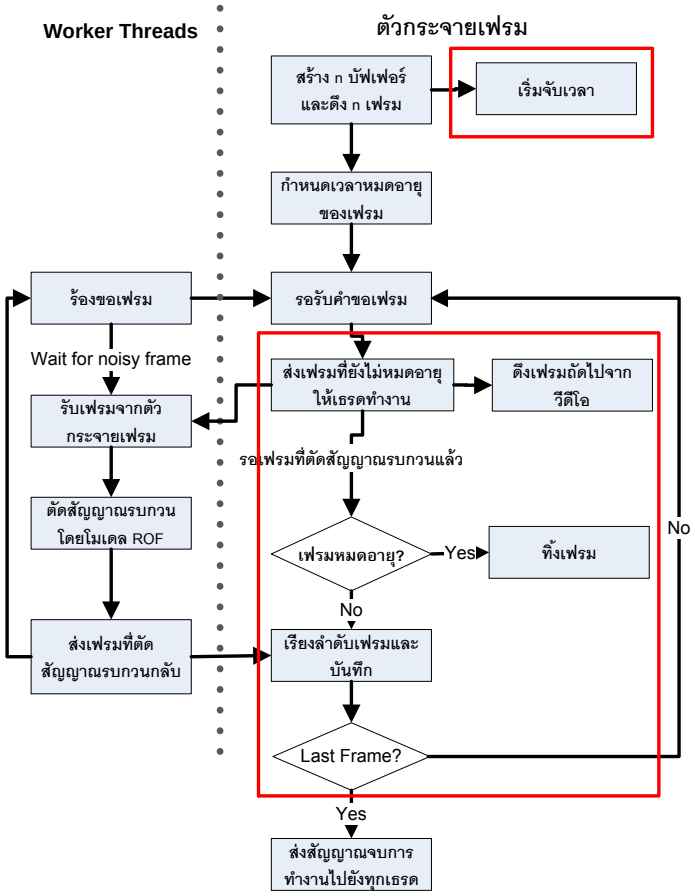


ภาพที่ 7.3 การพัฒนาตัวกระจายเฟรมโดยวิธีการอ่านล่วงหน้า (Prefetch)

ในการพัฒนาตัวกระจายเฟรมและเพิ่มวิธีการการอ่านล่วงหน้าเข้าไปนั้นจะทำให้การทำงานของตัวกระจายเฟรมนั้นมีประสิทธิภาพยิ่งขึ้น แต่อย่างไรก็ดี ผู้วิจัยได้พบปัญหาบนวีดีโอที่แสดงผลคืออัตราเฟรมที่แสดงผลไม่นิ่ง ในการแสดงผลภาพเคลื่อนไหวบางช่วงช้าผิดปกติ หรือเร็วผิดปกติ ซึ่งปัญหานั้นเกิดจากแต่ละเธรดใช้เวลาในการตัดสัญญาณรบกวนไม่เท่ากัน

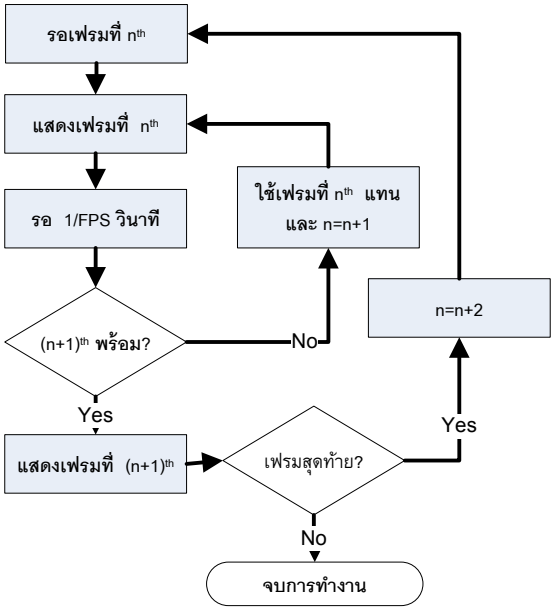
ด้วยปัญหาที่กล่าวมาข้างต้น ผู้วิจัยจึงได้นำวิธีการประทับเวลา (Time Stamp) เข้ามาใช้ในการดำเนินงาน ผู้วิจัยได้ปรับปรุงให้แต่ละเฟรมนั้นมีการเก็บเวลาอ้างอิงเทียบกับเฟรมแรกสุดไว้ด้วย ในการตัดสัญญาณรบกวนจะมีการตรวจสอบอายุของเฟรม หากเฟรมหมดอายุแล้วจะละทิ้งเฟรมนั้นและไม่นำเฟรมนั้นมาประมวลผล และในส่วนของการแสดงผลยังสามารถตรวจสอบได้ว่าควรข้ามเฟรมหรือไม่

วิธีการประทับเวลาที่เพิ่มเข้ามาจะแสดงได้ตามภาพที่ 7.4



ภาพที่ 7.4 การประยุกต์ใช้วิธีการประทับเวลาเพื่อควบคุมอัตราเฟรมให้คงที่

สำหรับการแสดงผลกับระบบประทับเวลานั้น เธรดที่ทำการแสดงผลจะต้องทำการเปลี่ยนแปลงขั้นตอนการแสดงผลเพื่อให้รองรับกับวิธีนี้ด้วย แสดงได้ตามภาพที่ 7.5



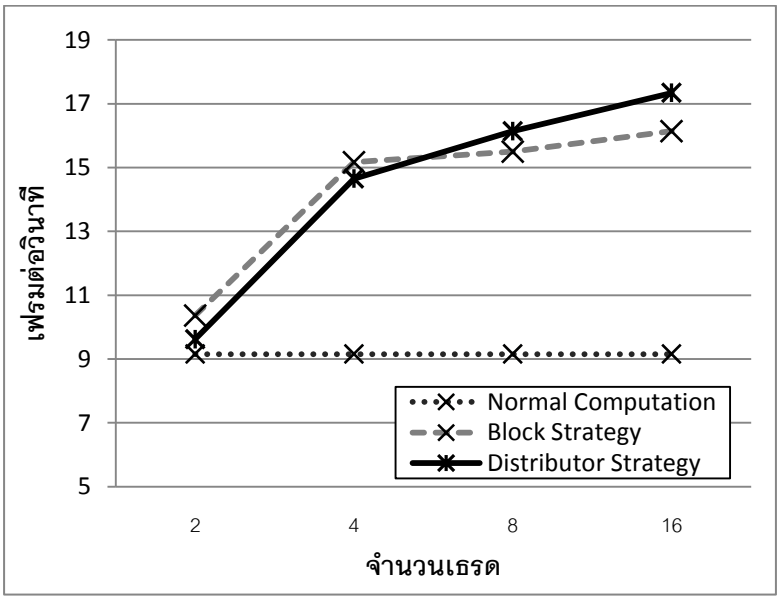
ภาพที่ 7.5 การทำงานของเธรดแสดงผลที่ได้รับการปรับปรุง

จากภาพที่ 7.5 ตัวแสดงผลจะมีการดึงข้อมูลการประทับเวลา และเทียบกับเวลาที่เริ่มต้นแสดงผลของเฟรมแรกสุดว่าเฟรมที่ได้มานั้นควรแสดงผลหรือไม่ หากเฟรมนั้นหมดอายุแล้วก็จะข้ามเฟรมนั้นไป

1.2 ผลการดำเนินงาน

ผู้วิจัยได้แบ่งผลการดำเนินงานออกเป็นสองด้านคือด้านประสิทธิภาพของวิธีการและด้านคุณภาพของภาพที่ผ่านการตัดสัญญาณรบกวนซึ่งแสดงในหัวข้อดังต่อไปนี้

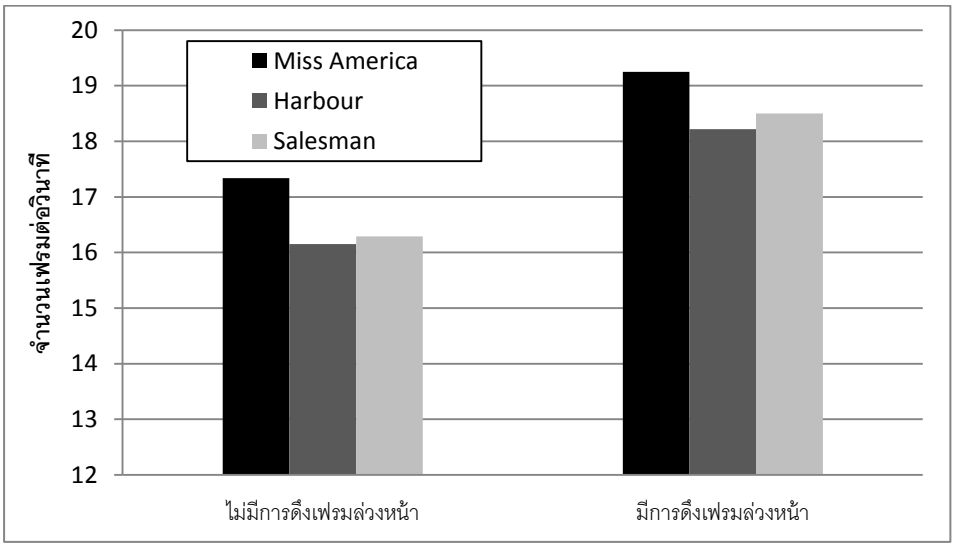
1.2.1 ด้านประสิทธิภาพของวิธีการ อัตราเฟรมต่อวินาทีของวิธีการแบบบล็อก (Block Strategy) และแบบมีตัวกระจายเฟรม (Distributor Strategy) สามารถแสดงได้ตามภาพที่ 7.6



ภาพที่ 7.6 อัตราเฟรมต่อวินาทีของแต่ละวิธีการ

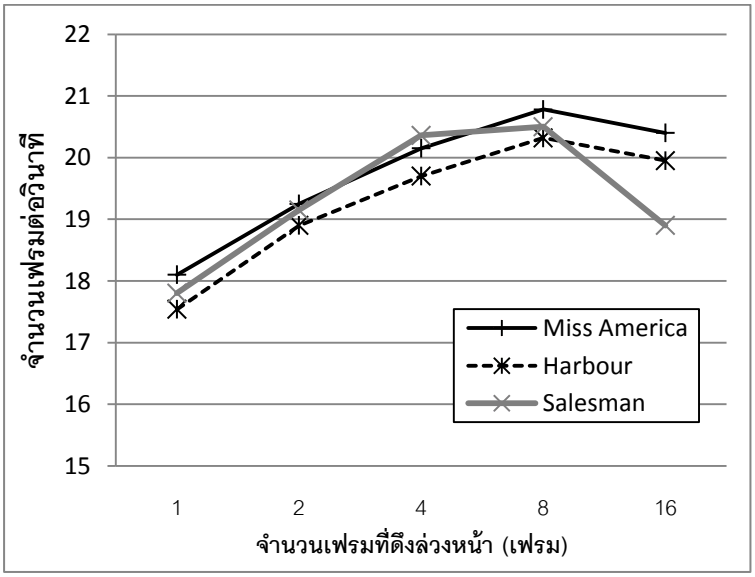
จากภาพที่ 7.6 จะเห็นได้ว่าวิธีการแบบมีตัวกระจายงาน (Distributor Strategy) นั้นจะสามารถเพิ่มอัตราเฟรมต่อวินาทีได้มากกว่าแบบบล็อก (Block Strategy) ซึ่งวิธีแบบบล็อกนั้นดีสำหรับ 2 และ 4 เธรดเนื่องจากการรอคอยของเธรดนั้นใช้เวลาไม่มาก และไม่ต้องใช้เธรดไปทำงานเป็นตัวกระจายเฟรม

เมื่อผู้วิจัยได้ปรับปรุงตัวกระจายเฟรมโดยเพิ่มวิธีการอ่านเฟรมล่วงหน้า (Prefetch) ลงไป อัตราเฟรมต่อวินาทีของภาพเคลื่อนไหวทั้งสามตัวอย่างจะแสดงได้ดังภาพที่ 7.7



ภาพที่ 7.7 เปรียบเทียบจำนวนเฟรมต่อวินาทีก่อนและหลังการประยุกต์ใช้การอ่านเฟรมล่วงหน้า

จากภาพที่ 7.7 จะเห็นว่าการอ่านเฟรมล่วงหน้านั้นสามารถเพิ่มอัตราเฟรมต่อวินาทีได้ถึง 22.02% เมื่อเทียบกับแบบที่ไม่ใช้การอ่านเฟรมล่วงหน้า นอกจากนี้ผู้วิจัยยังได้ศึกษาจำนวนเฟรมที่เหมาะสมในการอ่านเฟรมล่วงหน้า โดยกำหนดขนาดของพื้นที่เก็บข้อมูลชั่วคราวขนาดต่างๆ บนภาพเคลื่อนไหวตัวอย่างต่างๆ ซึ่งผลที่ได้จะแสดงได้ตามภาพที่ 7.8

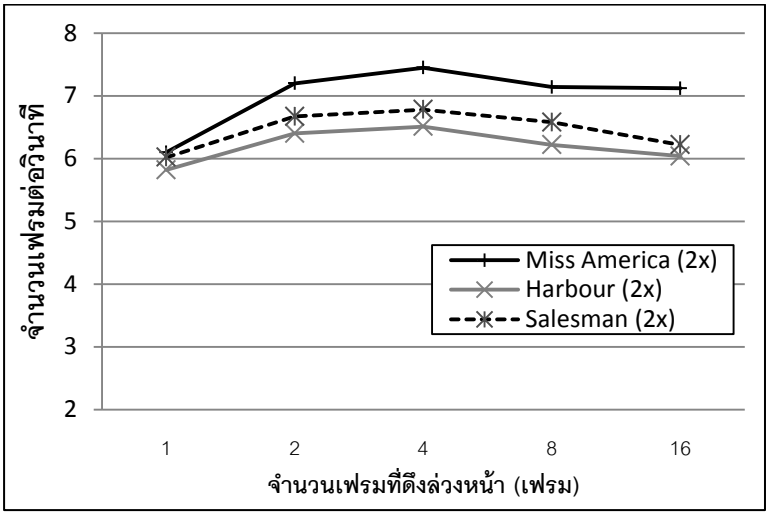


ภาพที่ 7.8 อัตราเฟรมต่อวินาทีที่ได้จากการอ่านเฟรมล่วงหน้าจำนวนต่างๆ

จากภาพที่ 7.8 จะเห็นได้ว่าจำนวนเฟรมที่อ่านล่วงหน้าที่ดีที่สุดในการทดลองนี้คือ 8 เฟรม ซึ่งหากอ่านเฟรมมากกว่า 8 เฟรมนั้น ตัวกระจายเฟรมจะเสียเวลาไปกับการอ่านเฟรมมากเกินไป

และการกระจายงานทำได้ช้าลง ในขณะที่หากน้อยเกินไปจะช้าลงเนื่องจากตัวกระจายงานต้องกลับไปอ่านเฟรมล่วงหน้าบ่อยครั้งมากขึ้น

และในท้ายสุด ผู้วิจัยได้ทดลองการอ่านเฟรมล่วงหน้ากับภาพเคลื่อนไหวที่ได้รับการแปลงให้ใหญ่ขึ้นสองเท่า ซึ่งผลที่ได้คล้ายกัน โดยจำนวนเฟรมที่อ่านล่วงหน้าที่ดีที่สุดอยู่ที่ 4 เฟรมต่อครั้ง ซึ่งแสดงผลได้ตามภาพที่ 7.9



ภาพที่ 7.9 อัตราเฟรมต่อวินาทีแยกตามจำนวนเฟรมที่อ่านล่วงหน้าขนาดต่างๆ (ขนาดสองเท่า)

นอกจากนี้ในส่วนของการควบคุมอัตราเฟรมต่อวินาทีนั้น ผู้วิจัยได้วัดจำนวนเฟรมที่มีการข้ามและความยาววีดีโอที่แสดงผลรวมก่อนและหลังการใช้วิธีการประทับเวลาได้ตามตารางที่ 7.1 (ก่อน) และตารางที่ 7.2 (หลัง) ดังนี้

ตารางที่ 7.1 ความยาวรวมของภาพเคลื่อนไหว และเวลาเฉลี่ยที่เฟรมแสดงผลคลาดเคลื่อนก่อนการใช้วิธีการประทับเวลา

ชื่อวีดีโอ	ความยาวต้นฉบับ (วินาที)	ความยาวเมื่อตัดสัญญาณรบกวน (วินาที)	ค่าความผิดพลาด (%)
Miss America	5.00	5.61	12.20
Harbour	20.00	32.10	60.50
Salesman	29.93	33.47	11.83

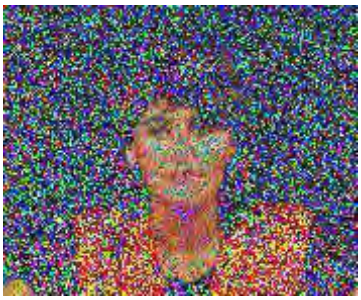
ตารางที่ 7.2 ความยาวรวมของภาพเคลื่อนไหว และเวลาเฉลี่ยที่เฟรมแสดงผลคลาดเคลื่อนหลังการใช้วิธีการประทับเวลา

ชื่อวีดีโอ	จำนวนเฟรมที่ถูกทิ้ง	ความยาวเมื่อตัดสัญญาณรบกวน	ค่าความผิดพลาด (%)	% Improved
Miss America	4	5.04	0.80	11.4
Harbour	173	21.01	5.05	55.45
Salesman	9	30.36	1.20	10.63

จะเห็นได้ว่าวิธีการประทับเวลานั้นจะสามารถลดความคลาดเคลื่อนในการแสดงผลของเฟรมได้สูงสุดถึง 55.45% บนภาพเคลื่อนไหว Harbour ภาพเคลื่อนไหวต้นฉบับนี้แสดงผลที่ 30 เฟรมต่อวินาที ในขณะที่ภาพเคลื่อนไหวต้นฉบับตัวอย่างอื่นแสดงผลที่ 15 เฟรมต่อวินาที แสดงให้เห็นว่าวิธีการประทับเวลานั้นได้ผลดีกับภาพเคลื่อนไหวต้นฉบับที่มีอัตราเฟรมต่อวินาทีสูงๆ

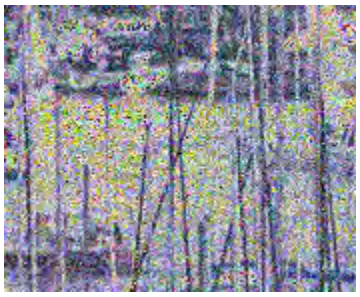
1.2.2 ด้านคุณภาพของภาพ ทางผู้วิจัยได้ใช้ภาพเคลื่อนไหวตัวอย่างที่ใช้อย่างแพร่หลายในการทำวิจัยด้านภาพเคลื่อนไหวคือ “Miss America”, “Harbor” และ “Saleman” นำมาใส่สัญญาณรบกวนประเภทบวกในทุกช่องสีอย่างอิสระต่อกัน ตัวอย่างในเฟรมที่ 80 แสดงได้ตามภาพที่ 7.10

(ก)



(ข)

(ค)



(ง)

(จ)

(ฉ)

ภาพที่ 7.10 ตัวอย่างภาพเคลื่อนไหวก่อน (ซ้าย) และหลัง (ขวา) ใส่สัญญาณรบกวนประเภทบวก

ภาพหลังจากตัดสัญญาณรบกวนประเภทบวกนั้นแสดงได้ดังภาพที่ 7.11

ภาพที่ 7.11 ภาพเฟรมที่ 80 จากวีดีโอตัวอย่างที่ตัดสัญญาณรบกวนแล้ว

จากภาพที่ 7.11 จะเห็นได้ว่าภาพที่ตัดสัญญาณรบกวนแล้วจะมีคุณภาพดีขึ้นอย่างมาก ถึงแม้ว่าเมื่อเทียบรายละเอียดกับภาพต้นฉบับก่อนใส่สัญญาณรบกวนสังเคราะห์จะเสียรายละเอียด

ไปบ้างแต่คุณภาพของภาพที่ได้นั้นเป็นที่น่าพอใจ โดยมีค่า PSNR ต่ำสุดและเฉลี่ยสูงกว่าวิธีการอื่นๆ ซึ่งค่า PSNR ที่วัดได้เมื่อเทียบกับวิธีอื่นนั้นแสดงในตารางที่ 7.3

ตารางที่ 7.3 เปรียบเทียบค่า PSNR แต่ละเฟรมระหว่างวิธีการการแปรผันรวมและวิธีการอื่นๆ

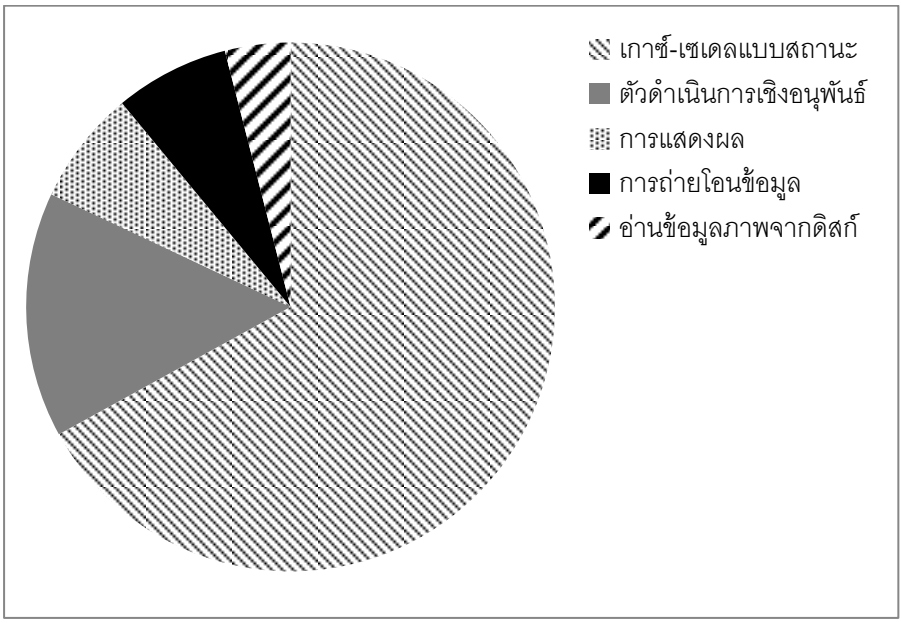
อัลกอริทึม	**PSNR**		
	ต่ำสุด	**สูงสุด**	**เฉลี่ย**
TV+FP	**33.90**	43.07	**34.80**
2D-DWT	29.84	46.79	33.33
2D DTCWT	30.45	47.53	33.89

2. การตัดสัญญาณรบกวนประเภทบวกและคูณจากภาพเคลื่อนไหว

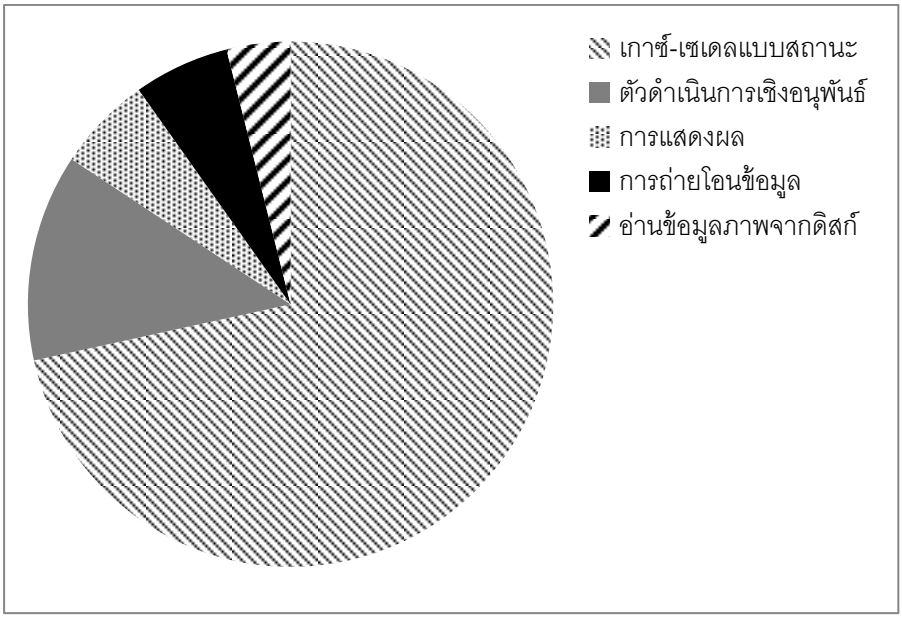
ในการตัดสัญญาณรบกวนประเภทบวกและคูณพร้อมกันของภาพเคลื่อนไหวนั้น ผู้วิจัยได้เลือกชุดภาพเคลื่อนไหวจากอัลตราซาวน์ โดยภาพอัลตราซาวน์ต้นฉบับนั้นเป็นภาพถ่ายสองมิติขาวดำ สามารถใช้การตัดสัญญาณรบกวนโดยวิธีการแปรผันแบบรวมแบบใหม่ซึ่งเสนอโดย N. Chumchob และคณะ [38] ได้อย่างมีประสิทธิภาพ โมเดลของสัญญาณรบกวนของภาพอัลตราซาวน์นั้นสามารถเขียนได้ดังสมการ (3.12) และการตัดสัญญาณรบกวนสามารถเขียนได้ดังสมการ (3.13) ดังปรากฏในบทที่ 3

2.1 วิธีการดำเนินงาน

ผู้วิจัยได้เลือกใช้วิธีการทำซ้ำแบบจุดตรึงร่วมกับอัลกอริทึมเกาซ์-เซเดลแบบสถานะในการแก้ปัญหานี้ เพื่อให้การวิเคราะห์และออกแบบการตัดสัญญาณรบกวนของภาพเคลื่อนไหวอัลตราซาวน์เป็นไปได้อย่างแม่นยำ ผู้วิจัยได้วัดเวลาในการทำงานแต่ละส่วนของการทำงานแบบตามลำดับ (Sequential) บนหน่วยประมวลผลหลักกับภาพสองขนาดคือ 256×256 และ 1024×1024 ได้ตามภาพที่ 7.12 และ 7.13



ภาพที่ 7.12 สัดส่วนเวลาการทำงานเป็นเปอร์เซนต์ของการตัดสัญญาณรบกวนบนภาพอัลตราซาวน์ขนาด 256×256



ภาพที่ 7.13 แสดงสัดส่วนเวลาการทำงานเป็นเปอร์เซนต์ของตัดสัญญาณรบกวนบนภาพอัลตราซาวน์ขนาด 1,024×1,024

จากภาพที่ 7.12 และ 7.13 จะเห็นได้ว่าการทำงานของอัลกอริทึมเกาซ์-เซเดลนั้นเป็นส่วนที่ใช้เวลาในการทำงานมากที่สุด และใช้เวลาเพิ่มขึ้นเมื่อภาพมีขนาดใหญ่ขึ้น ถ้าสามารถลดระยะเวลาในขั้นตอนเกาซ์-เซเดลได้มาก เวลาประมวลผลโดยรวมก็จะลดลงอย่างมากเช่นกัน ในงานวิจัยนี้

ผู้วิจัยได้เลือกใช้อัลกอริทึมเกาซ์-เซเดลแบบสถานะที่พัฒนาได้ในบทที่ 4 ที่มีประสิทธิภาพมาทดลองใช้งานบนหน่วยประมวลผลกราฟิกส์

ในส่วนที่ใช้เวลาในการประมวลผลรองลงมาคือส่วนของการคำนวณดำเนินการเชิงอนุพันธ์ตามสมการที่ (3.18) ในบทที่ 3 ซึ่งในส่วนนี้ใช้เวลาไม่มาก และสามารถประมวลผลเชิงขนานข้อมูลแต่ละจุดได้อย่างอิสระและใช้เวลาไม่มาก ผู้วิจัยจึงออกแบบวิธีการให้ใช้หน่วยประมวลผลทั้งหน่วยประมวลผลหลักและหน่วยประมวลผลกราฟิกส์ได้อย่างมีประสิทธิภาพโดยให้หน่วยประมวลผลหลักทำงานในส่วนตัวดำเนินการเชิงอนุพันธ์และหน่วยประมวลผลกราฟิกส์ทำงานในส่วนอัลกอริทึมเกาซ์-เซเดลแบบสถานะ

ผู้วิจัยได้ออกแบบการทำงานให้ซ้อนทับกันในรูปแบบสายท่อเพื่อให้การทำงานของหน่วยประมวลผลหลักและหน่วยประมวลผลกราฟิกส์สามารถซ้อนทับกันได้ กล่าวคือหน่วยประมวลผลหลักดึงเฟรมจากภาพเคลื่อนไหวและทำตัวดำเนินการเชิงอนุพันธ์หลังจากนั้นจึงส่งเฟรมที่ประมวลผลแล้วไปยังหน่วยประมวลผลกราฟิกส์เพื่อประมวลผลอัลกอริทึมเกาซ์-เซเดลแบบสถานะ และระหว่างที่หน่วยประมวลผลกราฟิกส์กำลังประมวลผลอยู่นั้นหน่วยประมวลผลหลักสามารถทำตัวดำเนินการเชิงอนุพันธ์สำหรับเฟรมต่อไปได้

เพื่อให้การจัดลำดับขั้นตอนการทำงานง่ายขึ้น ผู้วิจัยได้แบ่งเธรดต่างๆ ตามหน้าที่ออกเป็น 5 ประเภทดังนี้

1. **เธรดดึงภาพ** เป็นเธรดของหน่วยประมวลผลหลักมีหน้าที่ดึงเฟรมจากภาพเคลื่อนไหวอัลตราซาวน์และบรรจุลงหน่วยความจำหลัก ซึ่งผู้วิจัยใช้วิธีการดึงเฟรมล่วงหน้าเช่นเดียวกับการตัดสัญญาณรบกวนประเภทบวกเพื่อลดเวลาในการรอคอยอุปกรณ์ I/O
2. **เธรดประมวลผล** เป็นเธรดของหน่วยประมวลผลหลัก อาจมีได้มากกว่า 1 เธรด ใช้สำหรับทำงานตัวดำเนินการเชิงอนุพันธ์แบบขนานกับเฟรมที่เธรดดึงภาพดึงมาได้และเมื่อประมวลผลเสร็จจะส่งข้อมูลไปยังหน่วยประมวลผลกราฟิกส์เพื่อประมวลผลต่อไป
3. **เธรด GPU** เป็นเธรดที่ทำงานบนหน่วยประมวลผลกราฟิกส์ ซึ่งใช้สำหรับการทำงานของอัลกอริทึมเกาซ์-เซเดลแบบสถานะ

4. **เธรดรวมภาพ** เป็นเธรดสำหรับรอรับข้อมูลที่ประมวลผลเสร็จแล้วจากหน่วยประมวลผลกราฟิกส์และจัดเรียงเฟรมต่างๆ กลับที่เดิมและบรรจุไว้ในหน่วยความจำหลัก

5. **เธรดแสดงผล** ใช้สำหรับแสดงผลเฟรมที่ถูกตัดสัญญาณรบกวนแล้ว รวมทั้งควบคุมอัตราเฟรมเรทในการแสดงผล

วิธีการที่พัฒนาได้สามารถแสดงได้ตามภาพที่ 7.14

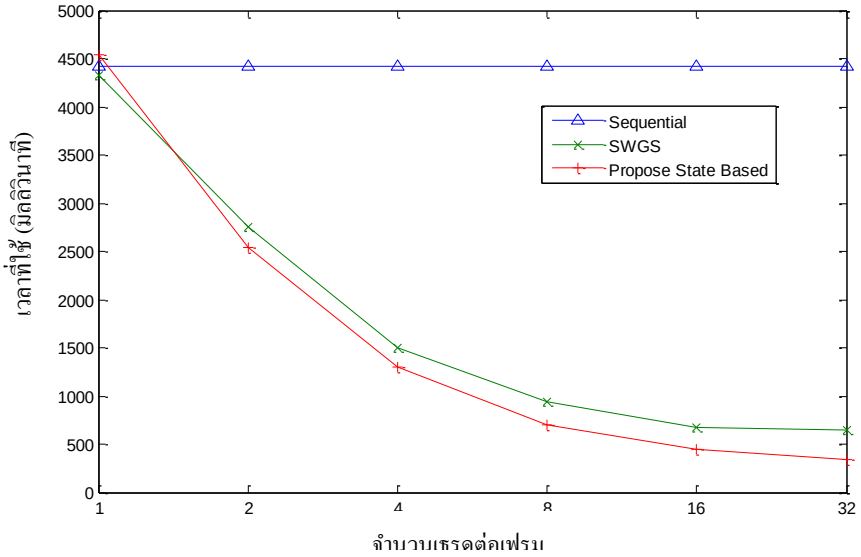
ภาพที่ 7.14 แสดงการทำงานของวิธีการการตัดสัญญาณรบกวนบนภาพเคลื่อนไหวอัลตราซาวน์ที่ได้พัฒนา

2.2 ผลการวิจัย

ในส่วนของผลการวิจัยนั้นผู้วิจัยได้แบ่งออกเป็นสองส่วนคือด้านประสิทธิภาพโดยวัดจากเวลาที่ใช้ในการทำงานและจำนวนเฟรมที่ได้ต่อวินาที และส่วนที่สองคือด้านคุณภาพของภาพที่ตัดสัญญาณรบกวนแล้ว โดยผลการทดลองมีดังต่อไปนี้

2.2.1 ด้านประสิทธิภาพ ผู้วิจัยได้วัดเวลาที่ใช้ต่อเฟรมบนภาพขนาด 480×352 พิกเซล โดยแบ่งเธรดออกเป็นกลุ่มๆ เพื่อทำงานกับแต่ละเฟรม ทั้งนี้เพื่อลดเวลาในการเติมสายท่อของ

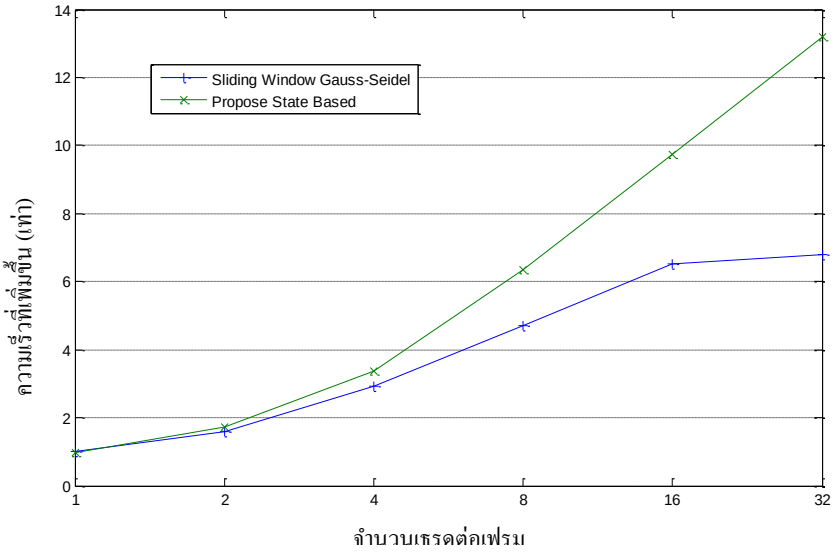
อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง ลดเวลาในการผสานการทำงานระหว่างรอบการทำซ้ำจุดตรึงและทำให้ตัดสัญญาณรบกวนได้หลายๆ เฟรมพร้อมกัน ซึ่งผลด้านเวลานั้นสามารถแสดงได้ตามภาพที่ 7.15



ภาพที่ 7.15 เวลาที่ใช้ของอัลกอริทึมที่พัฒนาได้กับวิธีการเดิมบนจำนวนเธรดต่อเฟรมขนาดต่างๆ

จากภาพที่ 7.15 จะเห็นได้ว่าอัลกอริทึมเกาซ์-เซเดลแบบสถานะ (Propose State Based) นั้นสามารถทำงานได้อย่างมีประสิทธิภาพมากกว่าอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่าง (SWGS) เนื่องจากสามารถลดเวลาการผสานการทำงานและทำให้เธรดต่างๆ เข้าทำงานได้รวดเร็วยิ่งขึ้น

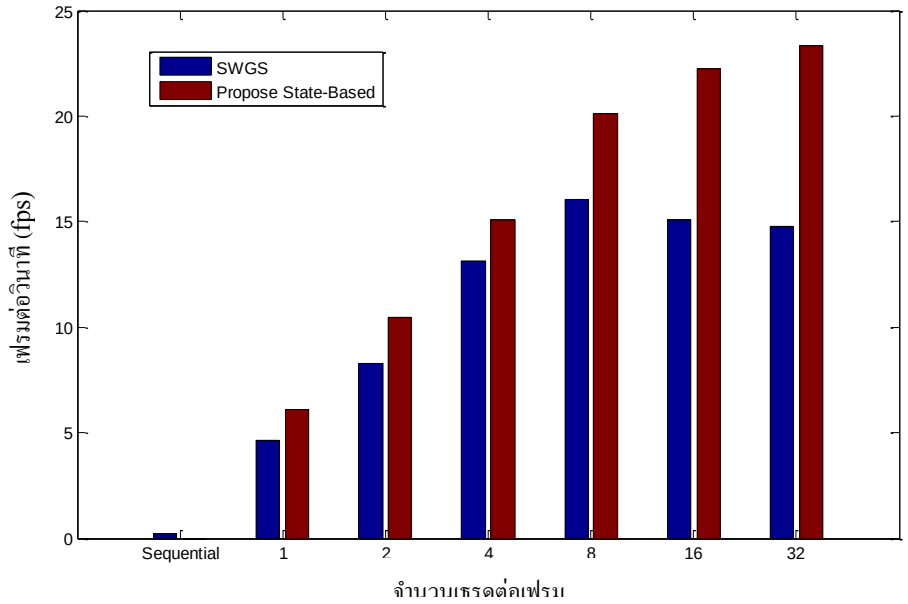
ความเร็วที่เพิ่มขึ้นเป็นจำนวนเท่าเมื่อเทียบกับการประมวลผลตามลำดับ (Sequential) นั้นสามารถแสดงได้ตามภาพที่ 7.16



ภาพที่ 7.16 จำนวนเท่าของความเร็วที่เพิ่มขึ้นของอัลกอริทึมเดิม (Sliding Window Gauss-Seidel) และเกาซ์-เซเดลแบบสถานะ (Propose State-Based)

จากภาพที่ 7.16 จะเห็นได้ว่าเกาซ์-เซเดลแบบสถานะที่พัฒนาได้นั้นสามารถเพิ่มความเร็วในการทำงานได้ถึง 13.18 เท่าที่ 32 เธรดต่อเฟรมเมื่อเทียบกับการประมวลผลอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยตามลำดับ ในขณะที่อัลกอริทึมเกาซ์-เซเดลเลื่อนบานหน้าต่างไร้แถวคอยนั้นสามารถเพิ่มความเร็วได้เพียงประมาณ 7 เท่าเมื่อเทียบกับหน่วยประมวลผลหลักเท่านั้น

นอกจากนี้ผู้วิจัยยังได้วัดจำนวนเฟรมต่อวินาทีของทั้งสองอัลกอริทึม ซึ่งแสดงได้ตามภาพที่ 7.17



ภาพที่ 7.17 จำนวนเฟรมต่อวินาทีของอัลกอริทึมที่พัฒนาได้กับวิธีการเดิมบนเธรดต่อเฟรมขนาดต่างๆ

จากภาพที่ 7.17 ในส่วนของ Sequential นั้นเป็นการประมวลผลบนหน่วยความประมวลผลหลักเพียงอย่างเดียวและไม่มีการซ้อนทับของการประมวลผลในสายท่อ การใช้อัลกอริทึมเกาซ์-เซเดลแบบสถานะนั้นสามารถให้ผลลัพธ์ได้ถึง 23 เฟรมต่อวินาทีเมื่อใช้ 32 เธรดต่อเฟรมเมื่อเทียบกับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างแบบเดิม (SWGS)

นอกจากนี้ยังมีปัจจัยที่เกี่ยวข้องกับความราบรื่นของภาพที่ได้ โดยผู้วิจัยได้วัดความล่าช้าระหว่างเฟรม (Between-Frame Delay : BFD) ทั้งส่วนค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐาน รวมทั้งเฟรมที่ถูกทิ้งเนื่องจากประมวลผลไม่ทันโดยแยกตามจำนวนเธรดต่อเฟรมได้ตามตารางที่ 7.4

ตารางที่ 7.4 จำนวนเฟรมที่ถูกทิ้ง ค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานของค่าความล่าช้าระหว่างเฟรมของจำนวนเธรดต่อเฟรมขนาดต่างๆ

เธรดต่อเฟรม	เฟรมที่ถูกทิ้ง (เฟรมต่อวินาที)		ค่าเฉลี่ย BFD (มิลลิวินาที)		S.D. ของ BFD (มิลลิวินาที)	
	SWGS	Proposed	SWGS	Proposed	SWGS	Proposed
1	19.33	17.89	514	393	61.1	79.3
2	15.73	13.55	289	230	83.5	80.1
4	10.89	8.94	183	159	74.2	82.2
8	7.94	3.90	149	119	90.2	86.8
16	8.96	1.72	159	108	97.9	99.6
32	9.22	0.67	162	103	101.8	99.7

จากตารางที่ 7.4 จะเห็นได้ว่าเมื่อเพิ่มจำนวนเธรดต่อเฟรมนั้น ค่าเฉลี่ยของความล่าช้าระหว่างเฟรมและจำนวนเฟรมที่ถูกทิ้งจะลดลง โดยอัลกอริทึมเกาซ์-เซเดลแบบสถานะนั้นสามารถลดจำนวนเฟรมที่ถูกทิ้งได้อย่างมีนัยสำคัญ โดยที่ 32 เธรดต่อเฟรมนั้นสามารถลดจำนวนเฟรมที่ถูกทิ้งได้เหลือเพียง 0.67 เฟรมต่อวินาที ในขณะที่เมื่อใช้อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างมีเฟรมที่ถูกทิ้งถึง 9.22 เฟรมต่อวินาที

2.2.1 ด้านคุณภาพของภาพ ผู้วิจัยได้แสดงภาพตัวอย่างเฟรมที่ 80 และเฟรมที่ 300 ตามภาพที่ 7.18(ก) และ 7.18 (ข) ตามลำดับ ผู้วิจัยได้แสดงคุณภาพของภาพของโมเดล JY เดิมในภาพที่ 7.18 (ค) และ 7.18 (ง) และวิธีแปรผันรวมด้วยโมเดล CC [38] ในภาพที่ 7.18 (จ) และ 7.18 (ฉ)

(ก)

(ข)

(ค)

(ง)

(จ)

(ฉ)

ภาพที่ 7.18 ตัวอย่างภาพต้นฉบับเฟรมที่ 80 และ 300 (ก และ ข) ภาพที่ตัดสัญญาณรบกวนด้วยโมเดล JY (ค และ ง) และภาพที่ตัดสัญญาณรบกวนด้วยวิธีการแปรผันรวมแบบใหม่ (จ และ ฉ)

ภาพที่ 7.18 จะเห็นได้ว่าวิธีการแปรผันรวมแบบใหม่นั้นสามารถตัดสัญญาณรบกวนได้ดูมีคุณภาพดีกว่าการใช้โมเดล JY ภาพที่ตัดสัญญาณรบกวนด้วยโมเดลการแปรผันรวมด้วยโมเดล CC นั้นมีค่า PSNR เทียบกับภาพต้นฉบับ 78.67dB ถึง 86.43dB

บทที่ 8

สรุปและอภิปรายผล

ในบทนี้จะกล่าวถึงผลการดำเนินงานที่ผ่านมาทั้งหมด โดยแบ่งออกเป็นส่วนต่างๆ สองส่วนคือส่วนการพัฒนาอัลกอริทึม และส่วนการนำอัลกอริทึมที่พัฒนาได้ไปใช้ในด้านต่างๆ

ผลการพัฒนาอัลกอริทึม

อภิปรายผลการทดลอง

ผู้วิจัยได้เล็งเห็นถึงปัญหาของอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างในการทำงานกับสถาปัตยกรรมแบบแกนประมวลผลจำนวนมากเช่นหน่วยประมวลผลกราฟิกส์ ซึ่งในแต่ละเธรดจะต้องรอเข้าทำงาน และมีการผสานการทำงานในระหว่างรอบการทำงานจำนวนมาก ผู้วิจัยจึงได้พัฒนาอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยขึ้น เพื่อลดเวลาในการผสานการทำงานระหว่างรอบโดยใช้ฟังก์ชันการส่ง (Mapping Function) ผลที่ได้สามารถลดเวลาประมวลผลได้ถึง 56.19% เมื่อเทียบกับการประมวลผลอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างเดิมสำหรับการประมวลผล 512 เธรด

นอกจากนี้ผู้วิจัยยังได้พัฒนาอัลกอริทึมเกาซ์-เซเดลเชิงขนานเพิ่มเติมจากแบบเลื่อนบานหน้าต่างไร้แถวคอย โดยเรียกว่าอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างสองทางซึ่งทำให้เธรดต่างๆ สามารถเข้าทำงานได้ทั้งแนวแถวและแนวคอลัมน์ และลดเวลาคอยเข้าทำงานของเธรดต่างๆ ผลจากการพัฒนาเพิ่มเติมนี้สามารถลดเวลาประมวลผลได้ประมาณ 46% เมื่อเทียบกับการประมวลผลเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยเดิมเมื่อทำงานกับ 512 เธรด

อย่างไรก็ตาม ผู้วิจัยยังเห็นว่าปัญหาเกี่ยวกับประสิทธิภาพการทำงานยังมีอยู่เนื่องจากมีจำนวนเธรดที่รอคอยเข้าทำงานเป็นจำนวนมาก จึงมีแนวคิดที่ให้เธรดแต่ละตัวทำงานได้อย่างอิสระ

ต่อกัน โดยแบ่งเธรดออกเป็นสี่สถานะ เธรดแต่ละตัวมีการตรวจสอบข้อมูลและเปลี่ยนสถานะด้วยตัวเอง ผู้วิจัยเรียกอัลกอริทึมนี้ว่าอัลกอริทึมเกาซ์-เซเดลแบบสถานะ ซึ่งอัลกอริทึมใหม่นี้มีประสิทธิภาพเป็น 1.25 เท่าเมื่อเทียบกับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยบนหน่วยประมวลผลกราฟิกส์ 512 เธรดที่ขนาดปัญหา 1,023×1,023 และเร็วขึ้น 26.88 เท่าเมื่อเทียบกับการประมวลผลแบบตามลำดับบนหน่วยประมวลผลกลาง

แนวทางการพัฒนาต่อ

อัลกอริทึมเกาซ์-เซเดลแบบสถานะนั้นอาจทำงานกับหน่วยประมวลผลกราฟิกส์ทำงานได้อย่างไม่เต็มที่ เนื่องจากแต่ละช่วงเวลาในแต่ละเธรดจะอยู่ในสถานะที่ต่างกัน ทำให้มีเส้นทางการทำงาน (Path Divergence) ที่ต่างกัน ในขณะที่หน่วยประมวลผลกราฟิกส์นั้นเหมาะสมสำหรับการทำงานแบบ SIMD มากกว่า

อัลกอริทึมใหม่ที่เสนอในวิทยานิพนธ์เล่มนี้ได้มุ่งเน้นการทำงานบน CUDA เพียงอย่างเดียว โดยยังไม่ได้ทำการทดลองกับ OpenCL ซึ่งเป็นแบบจำลองการเขียนโปรแกรมอีกแบบหนึ่งสำหรับการประมวลผลบนหน่วยประมวลผลแบบมากแกน อย่างไรก็ตาม อัลกอริทึมที่พัฒนาน่าจะนำไปประยุกต์ใช้กับ OpenCL ได้เช่นกัน โดยอาจต้องมีการวัดประสิทธิภาพในทุกๆ ด้านเทียบกับ CUDA อย่างละเอียดอีกครั้ง

ในอนาคต อัลกอริทึมที่นำเสนอนี้อาจนำมาปรับปรุงและประยุกต์เพื่อให้ใช้งานในด้านอื่นๆ ที่มีลักษณะของปัญหาใกล้เคียงกันเช่น มัลติกริดซึ่งนำอัลกอริทึมเกาซ์-เซเดลเข้ามาเป็นตัวทำให้เรียบ (smoother), พลศาสตร์ของไหลตลอดจนการถ่ายเทความร้อนและพลังงาน เป็นต้น

นอกจากนี้ในปัจจุบันมีสถาปัตยกรรมแบบหลายแกนประมวลผลใหม่ๆ เช่นสถาปัตยกรรมแบบแกนประมวลผลจำนวนมาก (Many Integrated Cores: MIC) ซึ่งพัฒนาโดยบริษัท Intel® โดยในปัจจุบันได้มีการนำมาใช้มากขึ้น การใช้อัลกอริทึมที่เป็นลักษณะแบบสถานะที่รันใน Native Mode นั้นจะตัดเวลาการขนถ่ายข้อมูล และความแตกต่างของสถาปัตยกรรม ซึ่งน่าจะทำให้อัลกอริทึมเกาซ์-เซเดลแบบสถานะประมวลผลได้รวดเร็วขึ้นมาก และในอนาคตสามารถปรับแต่งอัลกอริทึมเกาซ์-เซเดล แบบสถานะให้เหมาะสมกับหน่วยประมวลผลประเภทนี้มากขึ้น

การนำอัลกอริทึมไปใช้ด้านการตัดสัญญาณรบกวนภาพแบบบวก

อภิปรายผลการทดลอง

การตัดสัญญาณรบกวนภาพแบบบวกนั้น ทางผู้วิจัยได้นำอัลกอริทึมเกาซ์-เซเดลแบบไร้แถวคอยที่พัฒนาได้มาใช้แก้ปัญหาการทำซ้ำแบบจุดตรึงของการตัดสัญญาณรบกวนภาพแบบบวก ผลที่ได้นั้นสามารถลดเวลาในการประมวลผลภาพไปได้ถึง 56.19% เมื่อเทียบกับการประมวลผลสำหรับการประมวลผลอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างของเดิม และภาพที่ได้นั้นเป็นที่น่าพอใจ โดยมีค่า PSNR สูงสุด 66.73dB สำหรับภาพขนาด 2048×2048

แนวทางการพัฒนาต่อ

ในการทดลองส่วนนี้ผู้วิจัยได้ทดลองกับอัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยเท่านั้น ในอนาคตจะมีการนำอัลกอริทึมอีกทั้งสองตัวที่พัฒนาได้มาทดลองร่วมด้วย ผู้วิจัยคาดว่าอัลกอริทึมที่พัฒนาได้นั้นจะสามารถลดเวลาในการประมวลผลได้มากขึ้นอีก

การนำอัลกอริทึมไปใช้ด้านการตัดสัญญาณรบกวนภาพแบบบวกและคูณ

อภิปรายผลการทดลอง

ในส่วนการตัดสัญญาณรบกวนภาพแบบบวกและแบบคูณนั้น ผู้วิจัยได้ทำการทดลองกับภาพทั้งสองประเภทคือ ภาพถ่ายดาวเทียมขนาดใหญ่ โดยมีลักษณะเด่นคือ ภาพมีขนาดใหญ่มากไม่สามารถบรรจุลงหน่วยความจำของหน่วยประมวลผลกราฟิกส์ได้ และภาพจุลชีพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราด ซึ่งนักชีววิทยาและนักจุลชีววิทยาไม่สนใจพื้นหลัง แต่สนใจตัวจุลชีพ ซึ่งมีการเคลือบด้วยวัตถุที่นำไฟฟ้าเช่นทอง ทำให้สว่างบนพื้นหลังบนภาพที่ถ่ายได้

ในการแก้ปัญหาภาพถ่ายดาวเทียมขนาดใหญ่นั้นผู้วิจัยได้ตัดภาพออกเป็นส่วนเล็กๆ และจัดการทำงานให้เป็นแบบสายท่อโดยการแบ่งงานออกเป็นส่วนย่อยๆ ในการทดลองนี้อัลกอริทึมเกาซ์-เซเดลและวิธีการทำงานแบบสายท่อที่พัฒนาได้สามารถทำงานได้เร็วขึ้น 70.68 เท่าเมื่อเทียบ

กับการทำงานด้วยหน่วยประมวลผลหลักเพียงแกนเดียว นอกจากนี้หน่วยความจำที่ต้องการสำหรับแต่ละส่วนของภาพนั้นลดลงถึง 91.5% และภาพผลลัพท์ที่ได้นั้นมีคุณภาพดี

ส่วนการตัดสัญญาณรบกวนในภาพจุลชีพจากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้น ผู้วิจัยได้ใช้วิธีการแยกภาพจุลชีพออกจากพื้นหลังเพื่อลดขนาดปัญหาและทำการตัดสัญญาณรบกวนแบบขนานตามส่วนของจุลชีพที่ค้นพบ ในกรณีที่ไม่สามารถแยกจุลชีพออกจากพื้นหลังได้ ผู้วิจัยใช้อัลกอริทึมเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างไร้แถวคอยเพื่อตัดสัญญาณรบกวนบนภาพ วิธีนี้สามารถลดเวลาการทำงานได้ถึง 94.48% และและได้ภาพที่เหมาะแก่การศึกษาโครงสร้างวัตถุที่เกาะติดจุลชีพซึ่งโดยปกติสังเกตได้ยากบนภาพที่มีสัญญาณรบกวน

แนวทางการพัฒนาต่อ

ในส่วนของการตัดสัญญาณรบกวนจากภาพถ่ายดาวเทียมขนาดใหญ่ ในอนาคตอันใกล้นั้น ความก้าวหน้าของหน่วยประมวลผลกราฟิกส์จะเพิ่มขึ้นมาก รวมทั้งหน่วยความจำอาจสามารถบรรจุภาพถ่ายดาวเทียมได้ทั้งภาพซึ่งทำให้กระบวนการตัดสัญญาณรบกวนอาจต้องได้รับการปรับปรุงเพื่อให้ได้ผลดีที่สุด นอกจากนี้วิธีการที่พัฒนาได้ยังค่อนข้างเจาะจงกับรุ่นของหน่วยประมวลผลกราฟิกส์ ในอนาคตควรจะมีการพัฒนาอัลกอริทึมให้เข้ากับฮาร์ดแวร์หน่วยประมวลผลกราฟิกส์ได้ทุกรุ่น

ในส่วนของภาพจุลชีพนั้น เนื่องจากภาพที่ได้จากกล้องจุลทรรศน์อิเล็กตรอนแบบส่องกราดนั้นไม่ได้มีเพียงภาพแบคทีเรียที่นำมาทดลองเท่านั้น ยังมีภาพประเภทอื่นด้วยเช่น เชื้อรา พืช และวัสดุต่างๆ ในอนาคตควรมีการพัฒนาอัลกอริทึมให้รองรับภาพประเภทนี้ด้วย

การนำอัลกอริทึมไปใช้ด้านการตัดสัญญาณรบกวนบนภาพเคลื่อนไหว

อภิปรายผลการทดลอง

ในการตัดสัญญาณรบกวนบนภาพเคลื่อนไหว ผู้วิจัยได้ทำการทดลองกับภาพสองประเภท คือภาพวีดีโอสีทั่วไปซึ่งมีสัญญาณรบกวนแบบบวก โดยที่สัญญาณรบกวนไม่ขึ้นต่อกันในแต่ละช่องสี และภาพเคลื่อนไหวอัลตราซาวน์ซึ่งมีลักษณะสัญญาณรบกวนแบบบวกและคูณ ซึ่ง

อัลกอริทึมที่พัฒนาได้สามารถตัดสัญญาณรบกวนกับภาพเคลื่อนไหวทั้งสองประเภทได้ในลักษณะเรียลไทม์

ในงานวิจัยนี้ผู้วิจัยได้ทำการพัฒนาอัลกอริทึมเพื่อเปรียบเทียบหาวิธีการที่ให้อัตราเฟรมต่อวินาทีสูงที่สุด และอัตราการแสดงผลจำนวนเฟรมต่อวินาทีคงที่ เช่น การอ่านดิสก์ล่วงหน้า การทิ้งเฟรม และการแบ่งงานระหว่างหน่วยประมวลผลหลักและหน่วยประมวลผลกราฟิกส์ เพื่อซ่อนเวลาที่ใช้ในการขนถ่ายข้อมูลระหว่างหน่วยความจำหลักและหน่วยความจำของหน่วยประมวลผลกราฟิกส์

ผลการทดลองชี้ให้เห็นว่าอัลกอริทึมเกาซ์-เซเดลที่พัฒนาได้สามารถแก้ปัญหาการทำซ้ำแบบจุดตรึงในการตัดสัญญาณรบกวนบนภาพเคลื่อนไหวทั่วไป โดยให้อัตราจำนวนเฟรมต่อวินาทีเฉลี่ยเพิ่มเป็น 1.27 เท่าเมื่อเทียบกับการประมวลผลโดยเกาซ์-เซเดลแบบเลื่อนบานหน้าต่างเดิม และในการตัดสัญญาณรบกวนภาพอัลตราซาวน์นั้นสามารถเพิ่มประสิทธิภาพการทำงานได้ถึง 3.79 เท่า และได้อัตราเฟรมเป็น 14.8 เฟรมต่อวินาที โดยคุณภาพของภาพหลังจากการตัดสัญญาณรบกวนนั้นดีขึ้นจนเป็นที่น่าพอใจ

แนวทางการพัฒนาต่อ

ในการตัดสัญญาณรบกวนนี้ ผู้วิจัยทำงานในโดเมนเชิงพื้นที่เท่านั้น ยังไม่ได้คำนึงถึงโดเมนเชิงเวลา ทำให้วิธีการที่พัฒนานี้ยังไม่ได้คำนึงถึงสัญญาณรบกวนที่เกิดขึ้นในระหว่างการเปลี่ยนเฟรมแสดงผล นอกจากนี้การคำนวณยังไม่พิจารณาความสม่ำเสมอของความหน่วงระหว่างเฟรม ซึ่งในอนาคตอาจต้องมีการพัฒนาวิธีการอื่นๆ ที่คำนึงถึงสัญญาณรบกวนเชิงเวลาและความสม่ำเสมอของความหน่วงระหว่างเฟรม

รายการอ้างอิง

[1] ปราโมท เดชะอำไพ และ นิพนธ์ วรรณโสภาคย์ (2553). **ระเบียบวิธีเชิงตัวเลขในงานวิศวกรรม.** กรุงเทพฯ: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

[2] Mognnieg, H., and D. A. Lowther (2010) "The solution of electromagnetic field problem using Sliding Window Gauss-Seidel algorithm on a multicore processor." **IEEE Transaction on Magnetic** 46, (February): 3081-3084.

[3] Barrette, R., et al. (2008) **Template for the solution of linear systems: building blocks for iterative methods.** Auckland: Society for Industrial and Applied Mathematics.

[4] Rudin, L. I., S. Osher, and E. Fatemi (1992) "Nonlinear total variation based noise removal algorithms" **Physica D** 60 (January) : 259-268.

[5] Lidenbaum, M., M. Fischer, and A. M. Bruckstein (1994) "On Gabor Contribution to Image Enhancement." **Pattern Recognition** 27 (January): 1-8.

[6] Vogel, C. R., M. E. Oman (1996) "Iterative Methods for Total Variation Denoising." **SIAM Journal of Scientific Computation** 17, (January): 227-238.

[7] Paul, R., and M. Meyer (2000) **Restoration of motion picture film.** New York: Butterworth-Heinemann.

[8] Silberstein, M. (2012) **Efficient computation of sum-products on GPUs** Accessed February 19. Available from http://sites.google.com/site/silbersteinmark/Home/sum-product-gpu-kernel.

[9] Rob, F. (2008) **CUDA Application Design and Development.**, 2nd ed., New York: Elsevier.

[10] NVIDIA® Corperation. (2012) **NVIDIA CUDA programming guide version 1.0** Accessed February 19. Available from http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_ Programming _Guide_1.0.pdf

[11] Herikstad Solutions. (2012) **CUDA and double precision floating point numbers.** Accessed Febuary 19. Available from http://www.herikstad.net/2009/05/cuda-and-double-precision-floating.html

[12] Karimi, K., N. G. Dickson, and F. A. Hamze (2012) **Performance Comparison of CUDA and OpenCL**. Accessed March 10. Availble from http://arxiv.org/ftp/arxiv/papers/1005/1005.2581.pdf

[13] Barlow, R. H., and D. J. Evans (1982) "Parallel algorithms for the iterative solution to linear system." **Computation Journal** 25, (January):56-60.

[14] Cortadellas, N., E. Fernandez, and A. Garcia (2013) **Biomedical and biological applications of scanning electron microscopy**, Paris: CCiTUB.

[15] Bae, E., Juan Shi, and Xue-Cheng Tai (2011) "Graph Cuts for Curvature Based Image Denoising" **IEEE Transactions on Image Processing** 20, (May):1199-1210.

[16] Buades, A., and B. Coll, and Morel J. M. (2000) **On image denoising methods.** Paris:Centre de Mathematiques et de Leurs Applications.

[17] Scjerzer, O., et al. (2009) **Variational Methods in Imaging.** London:Springer.

[18] Maequina, A., and S. Osher (2000) "Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal" **SIAM Journal of Scientific Computation** 20 (January): 1964-1977.

[19] Chan, T. F., G. H. Golub, and P. Mulet (1999) "A nonlinear primal-dual method for total variation based image restoration." **SIAM Journal of Scientific Computation** 20, (January):1964-1977.

[20] Hirakawa, K., and T. W. Parks (2006) "Image denoising using total least squares", **IEEE Trans. Image Process** 15, 9 (September):2730-2742.

[21] Chumchob, N., Ke Chen, and Carlos Brito-Loeza (2012) "A new variational model for removal of combined additive and multiplicative noise and a fast algorithm for its numerical approximation" **International Journal of Computer Mathematics** 90 (January):1-12.

[22] Lelic, D., and Z. Gajic (2004) "Gauss-Seidel Iterations for SIR-Based Power Updates for Wireless CDMA Communication Networks" **International Journal of Wireless Information Networks** 11 (July): 115-121.

[23] Foschini, G., and Z. Miljanic (1994) "A simple distributed autonomous power algorithm and its converfence." **IEEE Transaction on Vehicular Technology** 42 (January): 641-646.

[24] Zuying Luo, J. Fan, and S. X. Tan (2011) "Localized Statistical 3D Thermal Analysis Considering Electro-Thermal Coupling." **IEEE International Symposium on Circuits and System**, Teipei, Taiwan, 24-27 May, 2009. Teipei:IEEE.

[25] Evans, D. J. (1984) "Parallel algoruthms S.O.R. Iterative Method." **Journal of Parallel Computing** 1, (March): 3-18.

[26] Koester, D. P., S. Ranka, and G. C. Fox (1994) "A Parallel Gauss-Seidel for Sparse Power System Matrices." **Proceeding of the Supercomputing**. 184-193. Washington D.C., USA 1994, USA:ACM/IEEE.

[27] Courtecuisse H. and J. Allard (2009) “Parallel dense Gauss-Seidel algorithm on many-core processors.” **High Performance Computation Conference**. 139-147. Seoul, South Korea, June 25-27, 2009. USA: IEEE CS Press.

[28] Lensen, F., H. Schafer, and C. Garbe (2009) “Denoising time-of-flight data with adaptive total variation” **Optical Engineering** 48, (January): 337-346.

[29] Perona, P. and J. Malik (1990) “Scale space and edge detection using anisotropic diffusion.” **IEEE Transaction for Pattern Analysis and Machine Intelligence** 12, 7 (August): 629-639.

[30] Abrahim, B. A. (2011) “Speckle noise reduction method combining total variation and wavelet shrinkage for clinical ultrasound imaging.” **Proceeding of Middle East Conference on Biomedical Engineering (MECBME)**. 80-83. Sharjah, UAE, February 21-24, 2011. USA:IEEE Press.

[31] Bennett, W. R. (1948) "Spectra of Quantized Signals" **Bell System Technical Journal** 27 (July): 446–472.

[32] Katona, M., et al. (2006) “A real-Time wavelet-domain video denoising implementation in FPGA.” **EURASIP Journal on Embedded Systems** (July): 1-12.

[33] Mahmoud, R. O. (2008) “Intelligent denoising technique for spatial video denoising for real-time applications” **Proceeding of International Conference on Computer Engineering & Systems**, 407-142. Cairo, Egypt. November 25-27, 2008. USA:IEEE Press.

[34] National Science Foundation (2012) **YUV Video Sequences** Accessed February 19. Available from http://trace.eas.asu.edu/yuv/

[35] Center for Remote Imaging, Sensing and Processing. (2013) **Microwave Remote Sensing** Accessed April 10. Available from http://www.crisp.nus.edu.sg/~research/tutorial/mw.htm

[36] Zhengmeng Jin and Xiaoping Yang (2010) “Analysis of a new variational model for multiplicative noise removal” **Journal of Mathamatics Analysis and Applications** 362 (February):415-426.

[37] Brietbart, J. (2010) “Data structure design for GPU based heterogeneous system.” **Proceeding of International Conference on High Performance Computing and Simulation**. 44-51. Leipzig, Germany, June 21-24, 2009. USA:IEEE Press.

[38] Chumchob, N. and C. Chantrapornchai (2013) “A fast denoising approach of medical ultrasound images corrupted by combined additive and multiplicative noise.” **Submitted,** (November).

ภาคผนวก ก

งานวิจัยที่ได้รับการตีพิมพ์

วารสารระดับนานาชาติ

Banpot Dolwithayakul, Atchara Chatkaew, Chantana Chantrapornchai, Noppadol Chumchob, “An adaptive parallel method for denoising microorganism image from scanning electron microscope”, Journal of Advance Microscopy Research, Vol.8, No.3, 2013.

Banpot Dolwithayakul, Chantana Chantrapornchai, Noppadol Chumchob, “State-Based Gauss-Seidel Framework for Real-time 2D Ultrasound Image Sequence Denoising on GPUs”, International Journal of Multimedia and Ubiquitous Engineering, Vol.9, No.1, 2014.

Banpot Dolwithayakul, Chantana Chantrapornchai, Noppadol Chumchob, “Two Parallel Strategies for Real-Time Spatial Video Denoising for Multi-Core Processor”, International Journal of Computer Applications, Vol.48, No.16, 2012.

Chantana Chantrapornchai, Banpot Dolwithayakul, , Kanok Huankamnerd, Sergai Gorlatch, “Parallel Mass Transfer Simulation of Nanoparticles using Nonblocking Communication”, International Journal of Multimedia and Ubiquitous Enginneering, Vol.5, No.1, 2010.

Banpot Dolwithayakul, Chantana Chantrapornchai, Noppadol Chumchob, “Additive and Multiplicative Noise Removal Framework for Large Scale Color Satellite Images on OpenMP and GPUs”, Studies in Surveying and Mapping Science, Vol.1, No.1, pp. 1-8. 2013.

B. Dolwithayakul, C. Chantrapornchai, S. Yooyeunyong, “Proper Parallel Method for Magnetic Sounding Equations on Multilayered Earth”, Communications in Computer and Information Science, Vol. 264, pp.122-129 , 2011.

วารสารระดับชาติ

Kanok Hournkamnuard, Chantana Chantrapornchai, Banpot Dolwithayakul, Sergei Gorlatch, Torsten Hoefler, “Two Parallel Algorithms for a Mass Transfer Simulation of Magnetic Nanoparticles”, ECTI Transactions on Electrical Eng. Electronics and Communication, Vol.9, No.2, 2011.

บรรพต ดลวิทยากุล, จันทนา จันทราพรชัย, สืบสกุล อยู่ยืนยง, “การสร้างตัวแก้สมการสะท้อนแม่เหล็ก ไฟฟ้าบนตัวกลางแบบวิวิธพันธุ์สำหรับพื้นโลกแบบหลายชั้นในบนเอ็มพีไอและโอเพนเอ็มพี”, Vol 22, No.3, 2012.

งานประชุมวิชาการระดับนานาชาติ

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, “GPU-Based Total Variation Image Restoration using Sliding Window Gauss-Seidel Algorithm”, International Symposium on Intelligent Signal Processing and Communication System (ISPACS), December 7-9, 2011.

B. Dolwithayakul, C. Chantrapornchai, S. Yooyeunyong, “Solving Magnetic Sounding Integral Equations from Multilayer Earth Using Message Passing Interface”, Annual Symposium on Computer Science and Engineering (ANSCSE 14), March 23-26, 2010.

B. Dolwithayakul, N. Chumchob, C. Chantrapornchai, , “GPU-Based Weighted-Jacobi Solver for 2D Poisson’s Equation with Domain Decomposition”, Annual Symposium on Computer Science and Engineering (ANSCSE 15), March 30 – April 2, 2011.

B. Dolwithayakul, C. Chantrapornchai, S. Yooyeunyong, “Magnetic Sounding from Multilayered Earth Equation Solver using Job Distributor Based on Message Passing Interface”, Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010.

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, “Real-time parallel spatial video denoising schemes on multi-core processors”, International Joint Conference on Computer Science and Software Engineering (JCSSE 2012), 2012.

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, “Real-time Video Denoising for 2D Ultrasound Streaming Video on GPUs”, International Computer Science and Engineering Conference (ICSEC), 2013.

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, “Real-time parallel spatial video denoising schemes on multi-core processors”, International Joint Conference on Computer Science and Software Engineering (JCSSE 2012), 2012.

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, “An efficient asynchronous approach for Gauss-Seidel iterative solver for FDM/FEM equations on multi-core processors”, International Joint Conference on Computer Science and Software Engineering (JCSSE 2012), 2012.

C. Chantrapornchai, B. Dolwithayakul, “On the Parallel Simulation of Magnatic Targeting of Nanoparticles in Capillaries”, International Joint Conference on Computer Science and Software Engineering (JCSSE 2013), 2013.

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, “Real-time Parallel Spatial Video Denoising Schemes on Multi-core Processor”, International Joint Conference on Computer Science and Software Engineering (JCSSE 2012), 2012.

B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, C. Leangsuksan, “Satellite Image Denoising framework on GPU and OpenMP”, Louisiana Alliance for Simulation-Guided Materials Applications, April 14-15, 2013.

งานประชุมวิชาการระดับชาติ

บรรพต ดลวิทยากุล, จันทนา จันทราพรชัย, นพดล ชุมชอบ, “ตัวแก้ระบบสมการเชิงเส้นขนาดใหญ่ด้วยวิธีเกาซ์-เซเดล โดย Sliding Window สองทาง บนหน่วยประมวลผลกราฟิกส์”, Thailand National Grid and Cloud Conference, 2011.

ประวัติผู้วิจัย

ชื่อ – สกุล	ว่าที่ ร.ต. บรรพต ดลวิทยากุล
ที่อยู่	82 ถ.เกษตรสิน ต.ลำพยา อ.เมือง จ.นครปฐม
ที่ทำงาน	นักคอมพิวเตอร์ มหาวิทยาลัยศิลปากร
ประวัติการศึกษา	
พ.ศ. 2546	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร
พ.ศ. 2550	วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์) ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร
ประวัติการทำงาน	
พ.ศ. 2546 – ปัจจุบัน	นักคอมพิวเตอร์ ฝ่ายระบบและพัฒนาโปรแกรม ศูนย์คอมพิวเตอร์ มหาวิทยาลัยศิลปากร