การจำลองภูมิประเทศแบบดิจิตอลโดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ

DIGITAL TERRAIN MODELLING BY USING TRIANGULATED IRREGULAR NETWORK

สุลักษณ์ สุ่มมาตย์

SULAK SOOMMART

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2543

ISBN 974-622-704-1

การจำลองภูมิประเทศแบบดิจิตอลโดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ

DIGITAL TERRAIN MODELLING BY USING TRIANGULATED IRREGULAR NETWORK

สุลักษณ์ สุ่มมาตย์

SULAK SOOMMART

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2543

ISBN 974-622-704-1

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การจำลองภูมิประเทศแบบดิจิตอลโดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ

DIGITAL TERRAIN MODELLING BY USING TRIANGULATED IRREGULAR NETWORK

ชื่อนักศึกษา นายสุลักษณ์ สุ่มมาตย์

รหัสประจำตัว 38061231

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.กิตติ ไพฑูรย์วัฒนกิจ

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ดร.วิศิษฏ์	หิรัญกิตติ	
ผศ.ดร.สุรพันธุ์	เอื้อไพบูลย์	
รศ.ดร.มนัส	สังวรศิลป์	
รศ.ดร.จเร	สุรวัฒน์ปัญญา	
รศ.ดร.กิตติ	ไพฑูรย์วัฒนกิจ	

วัน/เดือน/ปี ที่สอบ 13 ธันวาคม 2542 **เวลา** 12.00-13.00 น.

สถานที่สอบ ณ. ห้องสอบวิทยานิพนธ์ คณะวิศวกรรมศาสตร์ ตึก 12 ชั้น ชั้น 4 ห้อง (E12-403)

บัณฑิตวิทยาลัยรับรองแล้ว

รศ.ดร.มนัส สังวรศิลป์

รองคณบดีบัณฑิตวิทยาลัย

วันที่ 14 **เดือน** ธันวาคม **พ.ศ.** ๒๕๔๓

หัวข้อวิทยานิพนธ์	การจำลองภูมิประเทศแบบดิจิตอลโดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ
นักศึกษา	นาย สุลักษณ์ สุ่มมาตย์
รหัสประจำตัว	38061231
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2543
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ. ดร. กิตติ ไพฑูรย์วัฒนกิจ

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนออัลกอริทึม Incremental เพื่อสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ (Triangulated Irregular Network : TIN) ชนิดดีลอนเนย์ (Delaunay Triangulation) สำหรับจำลองลักษณะภูมิประเทศ ประโยชน์ของการใช้งานโครงข่ายสามเหลี่ยมในการจำลองลักษณะภูมิประเทศแบบดิจิตอล (Digital Terrain Modelling : DTM) คือ ระนาบสามเหลี่ยมสามารถบรรยายลักษณะความซับซ้อนของพื้นผิวได้ทุกรูปแบบ โดยการใช้โครงสร้างข้อมูลแบบพิเศษทำให้การคำนวณโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบปฏิบัติงานแบบพลวัต (Dynamic) เมื่อสามารถเพิ่มจุดข้อมูลเข้าหรือลบออกจากแบบจำลองได้อย่างต่อเนื่อง ซึ่งขึ้นอยู่กับว่าจุดข้อมูลนั้นมีส่วนร่วมกับแบบจำลองหรือไม่ โครงสร้างของโครงข่ายจะถูกจัดเรียงใหม่เสมอหลังจากมีการเพิ่มหรือลดจำนวนจุดจนกระทั่งตรงตามกฎเกณฑ์ของสามเหลี่ยมดีลอนเนย์ ส่งผลให้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบสามารถประมาณค่าโครงสร้างของภูมิประเทศได้เป็นอย่างดี ภายในข้อกำหนดที่ได้ตั้งไว้และใช้จำนวนของสามเหลี่ยมน้อยที่สุด อัลกอริทึม Incremental ถูกนำไปทดลองกับข้อมูลหลากรูปแบบ อาทิเช่น เซ็ตของจุดจากการสุ่ม จุดที่คัดเลือกจากแผนที่ลายเส้นแสดงระดับความสูง (Contour Map) และ แบบจำลองระดับความสูง (Digital Elevation Model : DEM) ผลลัพธ์จากการทดลองเปรียบเทียบกับวิธีการอื่น ๆ แสดงให้เห็นถึงประสิทธิภาพของอัลกอริทึม Incremental

Thesis Title	Digital Terrain Modelling By Using Triangulated Irregular Network
Student	Mr. Sulak Soommart
Student ID.	38061231
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2000
Thesis Advisor	Assoc.Prof.Dr. Kitti Paithoonwattanakij

ABSTRACT

This thesis presents the incremental Delaunay triangulation algorithm for constructing a Triangulated Irregular Network (TIN) to represent the terrain surface. The advantage of using triangles in Digital Terrain Modelling (DTM) is the possibility of adapting the triangular elements to fit variation of the terrain surface. By using the novel data structures, the computation of TIN is dynamic where the algorithm enables a point to be added, or removed from a model, depending on the contribution of the point relates with the model. The structure of TIN is recursively reorganized until the Delaunay criterion is satisfied. Consequently, TIN can approximate any surface at any desired tolerance with a minimal number of triangles. The incremental algorithm is implemented with various data sets, random points, digitized spots from contour map, and Digital Elevation Models (DEM). The results of experiment and performance compared with other algorithms show the efficiency of the Incremental algorithm.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษา รวมทั้งได้รับตรวจสอบจาก รศ. ดร. กิตติ ไพฑูรย์วัฒนกิจ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้ทำวิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอกราบขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคุณพ่อและคุณแม่ ผู้ให้กำเนิดข้าพเจ้า ผู้ให้ความเอื้อเฟื้อทางการศึกษา ให้การอบรมดูแลด้วยดีตลอดมา

ขอขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ทุกท่าน ที่ได้ช่วยแก้ไขวิทยานิพนธ์ฉบับนี้ให้สมบูรณ์มากยิ่งขึ้น

ขอขอบคุณรุ่นพี่และเพื่อนๆทุกคนที่ช่วยให้กำลังใจ พร้อมทั้งมอบคำแนะนำและคำติชมจนสำเร็จสมบูรณ์ได้ด้วยดี

สุดท้ายขอขอบคุณบัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ให้ทุนสนับสนุนการทำวิทยานิพนธ์ครั้งนี้

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

สุลักษณ์ สุ่มมาตย์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

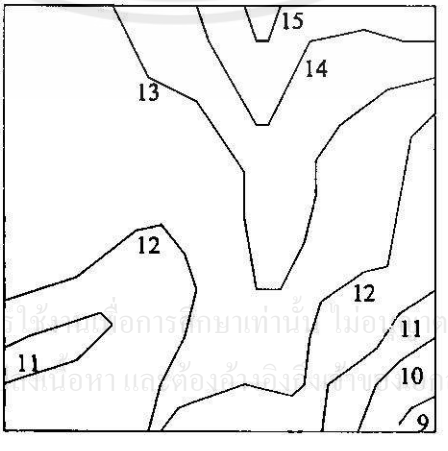
สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

การจำลองภูมิประเทศแบบดิจิตอล (Digital Terrain Modelling หรือ DTM) ถือเป็นส่วนประกอบหลักที่สำคัญอันหนึ่งของระบบประมวลผลข้อมูลทางภูมิศาสตร์ เช่น ระบบสารสนเทศภูมิศาสตร์ (Geographical Information System หรือ GIS) การจำลองภูมิประเทศแบบดิจิตอล คือ การสร้างแบบจำลองจากข้อมูลทางภูมิศาสตร์โดยใช้คอมพิวเตอร์ เพื่อประโยชน์ในการวิเคราะห์รูปพรรณสัณฐาน แสดงรายละเอียดและปรากฏการณ์ต่างๆที่เกี่ยวข้องกับภูมิประเทศหรือพื้นผิวของโลก ณ บริเวณนั้น ข้อมูลทางภูมิศาสตร์จากแหล่งข้อมูลต่างๆ เช่น แผนที่ทางทหาร ภาพถ่ายทางอากาศ ภาพถ่ายจากดาวเทียม ฯลฯ จะถูกคำนวณและถ่ายทอดให้อยู่ในรูปแบบของตัวเลข (แปลงข้อมูลจากอนาลอกเป็นดิจิตอล) ก่อนที่จะนำมาประมวลผลโดย DTM ด้วยเหตุนี้จึงเรียกการจำลองภูมิประเทศโดยใช้คอมพิวเตอร์ว่าเป็นแบบดิจิตอล

ข้อมูลพื้นฐานที่จำเป็นสำหรับการจำลองลักษณะภูมิประเทศคือ ข้อมูลความสูงของพื้นผิว (เหนือระดับน้ำทะเล) ซึ่งจะเป็นตัวอธิบายรูปพรรณสัณฐานของภูมิประเทศที่ถูกสำรวจ ข้อมูลความสูงจะถูกพิจารณาในรูปของแบบจำลองความสูง (Elevation Model) ในอดีตแบบจำลองความสูงถูกนำเสนอโดยแผนที่ลายเส้นแสดงชั้นความสูงหรือ Contour map (ดังในรูปที่ 1.1) โดยเส้นคอนทัวร์ (Contour lines) แต่ละเส้นแทนความสูงแต่ละระดับ ข้อมูลความสูงได้มาจากการสำรวจภาคพื้นดิน (Ground survey) เมื่อเทคโนโลยีก้าวหน้ายิ่งขึ้นข้อมูลความสูงจากแผนที่เส้นความสูงได้ถูกถ่ายทอดให้อยู่ในรูปของข้อมูลดิจิตอลเพื่อจัดเก็บบนฐานข้อมูลของคอมพิวเตอร์ โดยการใช้เครื่องมือดิจิไตเซอร์ (Digitizer) อ่านค่าพิกัดตามแนวเส้นคอนทัวร์ แบบจำลองความสูงนี้จึงถูกเรียกว่าแบบจำลองความสูงดิจิตอล (Digital Elevation Model)



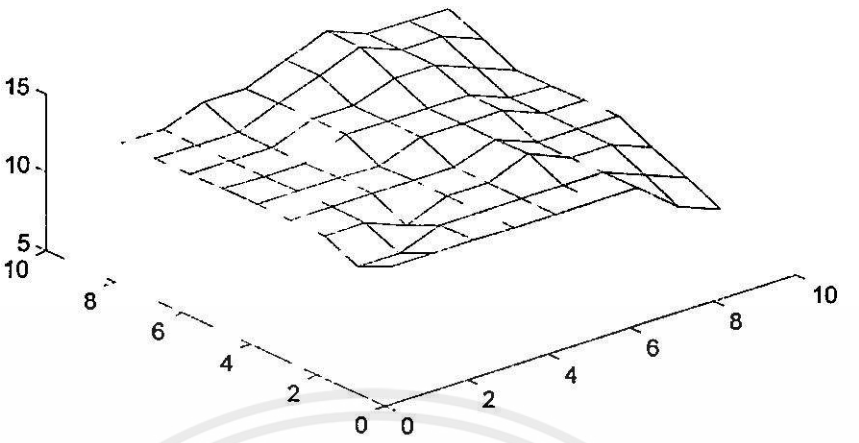
รูปที่ 1.1 แผนที่เส้นความสูง (Contour map)

แต่ด้วยสาเหตุที่ว่า แบบจำลองความสูงชนิดเส้นคอนทัวร์เป็นการนำเสนอลักษณะภูมิประเทศในแบบ 2 มิติ และข้อมูลความสูงไม่มีความต่อเนื่องกันระหว่างเส้นคอนทัวร์ จึงไม่เหมาะที่จะนำมาสร้างเป็นแบบจำลองในแบบ 3 มิติ วิธีการที่จะทำให้ข้อมูลความสูงมีความต่อเนื่องกันมากขึ้นคือ การแทรกข้อมูลโดยการประมาณค่าความสูง (Linear interpolation) ระหว่างเส้นคอนทัวร์ เพื่อสร้างแบบจำลองความสูงชนิดกริดคงที่ (Regular grid) ดังแสดงในรูปที่ 1.2 และ 1.3 เมื่อระยะห่างระหว่างตำแหน่งทดสอบความสูงมีค่าเท่ากันในรูปของเมตริกความสูง (Elevation matrix) การจัดการข้อมูลสำหรับแบบจำลองความสูงชนิดกริดจึงทำได้ง่าย เมื่อค่าความสูงถูกเก็บอยู่ในรูปของอาร์เรย์ 2 มิติ (Two-dimensional array) การเข้าถึงข้อมูลความสูง ณ ตำแหน่งนั้นๆทำได้โดยกำหนดหมายเลขแถวและคอลัมน์ เช่น elevation = matrix[row][column] ดังนั้นในบางครั้งจึงมักเรียกการจำลองภูมิประเทศโดยใช้แบบจำลองความสูงแบบกริดว่าเป็นแบบ 2.5 มิติ เพราะข้อมูลที่ให้ความสำคัญมีเพียงข้อมูลความสูงเท่านั้น

12	12	13	13	14	15	14	14	14
12	12	12	13	14	15	14	14	13
12	12	12	13	13	14	14	13	12
12	12	12	13	13	13	13	12	12
12	12	12	12	13	13	13	12	12
12	12	12	12	12	13	12	12	12
12	12	11	12	12	13	12	12	11
11	11	12	12	12	12	12	11	10
12	12	12	12	12	12	12	10	9

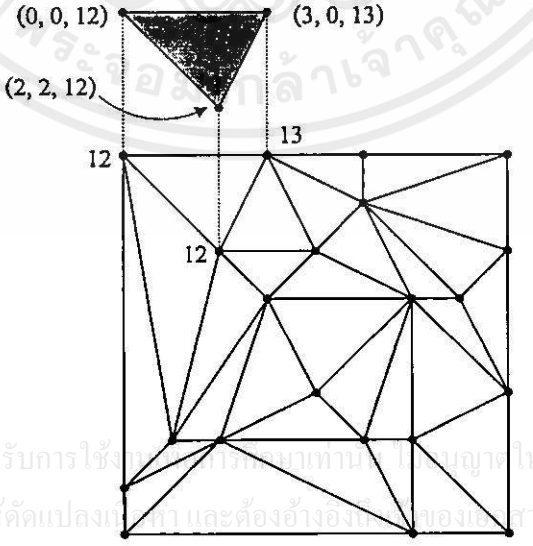
รูปที่ 1.2 แบบจำลองความสูงชนิดกริดคงที่ (Regular grid)

แบบจำลองความสูงชนิดกริดนอกจากจะคำนวณได้จากแผนที่คอนทัวร์แล้ว ยังสามารถคำนวณโดยตรงจากภาพถ่ายทางอากาศ และภาพถ่ายทางไกลจากดาวเทียม ด้วยเทคโนโลยีอันทันสมัยทำให้การเก็บรวบรวมและผลิตข้อมูลทางภูมิศาสตร์จำนวนมหาศาลทำได้โดยง่าย การจัดการกับข้อมูลที่มีประสิทธิภาพจึงเป็นสิ่งที่จำเป็น แบบจำลองความสูงชนิดกริดได้รับความนิยมในการใช้งานมาเป็นเวลานาน ด้วยสาเหตุของการจัดเก็บข้อมูลที่เป็นระเบียบและการเข้าถึงข้อมูลก็ทำได้อย่างสะดวกสบาย แต่ในทางตรงกันข้ามความหนาแน่นที่สม่ำเสมอของข้อมูลความสูงกลับไม่สามารถอธิบายความซับซ้อนของภูมิประเทศได้อย่างเต็มที่ ความหนาแน่นของข้อมูลปริมาณมากจึงจำเป็นเมื่อต้องถ่ายทอดลักษณะภูมิประเทศที่ต้องการความถูกต้องในระดับสูง ซึ่งส่งผลเสียในเรื่องของความซ้ำซ้อนของข้อมูล

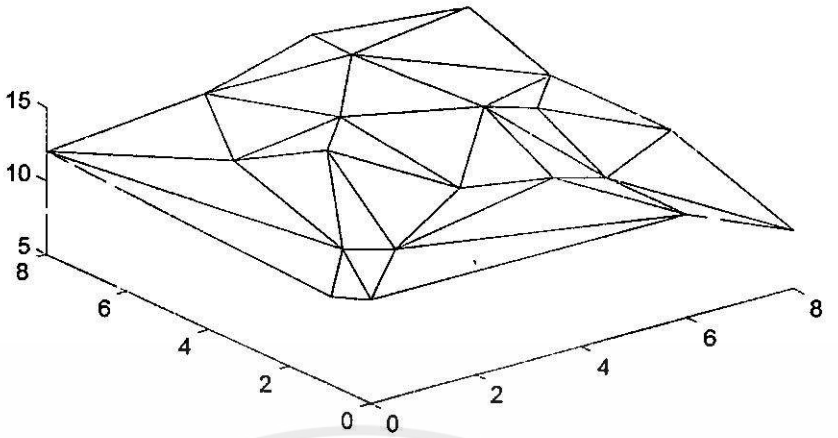


รูปที่ 1.3 แบบจำลองความสูงชนิดกริด (เมื่อแสดงในแบบ 3 มิติ)

ทางเลือกใหม่สำหรับแก้ไขปัญหานี้คือการประยุกต์ใช้ โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ (Triangulated Irregular Network หรือ TIN) เพื่อเป็นแบบจำลองความสูงและโครงสร้างหลักของการจำลองภูมิประเทศ แบบจำลองความสูงชนิด TIN ประกอบขึ้นจากจุดซึ่งกระจัดกระจายอย่างไม่เป็นระบียบตามตำแหน่งสำคัญของพื้นผิว จุดสำคัญเหล่านี้ถูกเชื่อมเข้าหากันในรูปของโครงข่ายสามเหลี่ยม ความซับซ้อนของภูมิประเทศถูกอธิบายด้วยระนาบสามเหลี่ยมซึ่งเรียงติดต่อกันครอบคลุมทั่วทุกซอกทุกมุมของพื้นที่ ระนาบสามเหลี่ยมประกอบขึ้นจากจุด 3 จุด (ดังรูปที่ 1.4) ประโยชน์ที่ได้รับจากการใช้รูปสามเหลี่ยมในการจำลองพื้นผิวคือ สามารถอธิบายโครงร่างของพื้นผิวได้ใกล้เคียงที่สุดและใช้จุดข้อมูลน้อยที่สุดภายใต้ค่าความแม่นยำที่ตั้งไว้ (ดังรูปที่ 1.5)



รูปที่ 1.4 โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ (TIN)



รูปที่ 1.5 โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ (เมื่อแสดงในแบบ 3 มิติ)

รูปแบบของโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบที่ได้รับการยอมรับ และถูกประยุกต์ใช้ในงานจำลองภูมิประเทศมากที่สุดคือ สามเหลี่ยมดีลอนเนย์ (Delaunay Triangulation) หลักการคำนวณของสามเหลี่ยมดีลอนเนย์ทำให้โครงข่ายผลลัพธ์มีรูปแบบของโครงสร้างที่แน่นอน การสร้างโครงข่ายสามเหลี่ยมให้ตรงตามทฤษฎีดีลอนเนย์จากเซ็ตของจุดใดๆ จะให้ผลลัพธ์ที่เหมือนเดิมเสมอ ไม่ว่าจะใช้วิธีการใดหรือทำการคำนวณซ้ำซ้อนซักกี่ครั้ง อีกทั้งรูปสามเหลี่ยมแต่ละรูปซึ่งบรรจุอยู่ภายในโครงข่ายจะมีรูปร่างที่สมบูรณ์มากที่สุด คือมีลักษณะใกล้เคียงกับรูปสามเหลี่ยมด้านเท่ามากที่สุด ทำให้โครงสร้างของแบบจำลองมีความมั่นคง การใช้งานรูปสามเหลี่ยมที่สมบูรณ์ในการสร้างภาพกราฟิกจะช่วยลดการเกิดความผิดเพี้ยนที่ขอบภาพ (Anti-alias)

มีวิธีการมากมายที่ได้ถูกคิดค้นขึ้นสำหรับสร้างโครงข่ายสามหลี่ยมดีลอนเนย์ ซึ่งสามารถจัดแบ่งออกเป็น 2 กลุ่มใหญ่ๆ คือ กลุ่มวิธีการแบบสถิต (Static methods) และ กลุ่มวิธีการแบบพลวัติ (Dynamic methods) การคำนวณโครงข่ายสามเหลี่ยมดีลอนเนย์สำหรับการจำลองภูมิประเทศในแบบสถิต เซ็ตของจุดข้อมูลสำคัญบนผิวพื้นต้องได้รับการคัดเลือกและจัดเตรียมไว้ก่อน จากนั้นจึงจะใช้วิธีการคำนวณโครงข่ายสามเหลี่ยมบนเซ็ตของจุดข้อมูลเหล่านี้ คุณสมบัติดีลอนเนย์ที่สมบูรณ์ของโครงข่ายจะปรากฏขึ้นเมื่อเสร็จสิ้นขบวนการคำนวณแล้วเท่านั้น สำหรับวิธีการแบบพลวัติ เซ็ตของจุดข้อมูลสำคัญบนพื้นผิวอาจถูกจัดเตรียมไว้ก่อน หรืออาจทำการคัดเลือกในขณะทำการคำนวณโครงข่ายอยู่ก็ได้ คุณสมบัติความเป็นดีลอนเนย์ของโครงข่ายจะเกิดขึ้นเสมอตั้งแต่เริ่มการคำนวณจนสิ้นสุดการคำนวณ อันเนื่องมาจากโครงข่ายจะมีการปรับโครงสร้างเสมอเมื่อมีจุดข้อมูลถูกเพิ่มหรือลดเข้าออกจากโครงข่าย

ในวิทยานิพนธ์นี้ได้เลือกใช้วิธีการแบบพลวัติ ซึ่งเหมาะที่จะนำมาประยุกต์ใช้กับงานจำลองโครงสร้างของพื้นผิวด้วยข้อมูลความสูง เพราะสามารถเพิ่มหรือลบจุดเข้าออกจากโครงข่ายได้ตลอดเวลาซึ่งขึ้นอยู่กับว่าจุดข้อมูลนั้นมีส่วนร่วมกับแบบจำลองหรือไม่ ซึ่งวิธีการแบบสถิตไม่

สามารถปฏิบัติเช่นนี้ได้ เนื่องจากวิธีการแบบสถิตเหมาะที่จะใช้คำนวณกับเซ็ตของจุดจำนวนคงที่เท่านั้น เมื่อมีการเปลี่ยนแปลงใดๆเกิดขึ้นเซ็ตข้อมูล (เช่น การเพิ่ม การลด หรือการเปลี่ยนตำแหน่งของจุด) โครงข่ายสามเหลี่ยมต้องได้รับการคำนวณใหม่ทั้งหมดตั้งแต่เริ่มต้น

วิธีการแบบพลวัตจะไม่สามารถปฏิบัติงานได้อย่างเต็มประสิทธิภาพ ถ้าปราศจากโครงสร้างข้อมูล (Data Structure) ที่เหมาะสมคอยให้การสนับสนุน เนื่องจากโครงสร้างข้อมูลสำหรับการคำนวณโครงข่ายสามเหลี่ยมดีลอนเนย์มีอยู่หลายรูปแบบ แต่ละรูปแบบต่างก็เหมาะกับวิธีการเฉพาะของอัลกอริทึมนั้น ในวิทยานิพนธ์นี้ได้นำเสนอโครงสร้างข้อมูลชนิดขอบ (Edge-Based data structure) สำหรับจัดการกับโครงสร้างของโครงข่ายสามเหลี่ยม โครงสร้างข้อมูลชนิดนี้ได้รับการออกแบบเพิ่มเติม เพื่อให้สามารถจัดการกับข้อมูลขนาดใหญ่ภายในโครงข่ายอย่างมีประสิทธิภาพ ระหว่างการปรับปรุงโครงสร้างของโครงข่าย และสามารถคำนวณหาจุดสำคัญบนพื้นผิวของแบบจำลองภายใต้ระยะทางที่กำหนดไว้ได้โดยอัตโนมัติ

การจำลองภาพภูมิประเทศ (Visualization of the terrain) เป็นอีกส่วนหนึ่งที่วิทยานิพนธ์นี้ให้ความสำคัญในการนำเสนอ โดยการประยุกต์ใช้เทคนิคต่างๆในการสร้างภาพกราฟิก อาทิเช่น การแสดงโครงสร้างของโครงข่ายสามเหลี่ยมโดยใช้เทคนิค Wirefame การสร้างแผนที่คอนทัวร์จากโครงข่ายสามเหลี่ยม (TIN-to-Contour lines) การแสดงความสูงของพื้นผิวโดยใช้ระดับสีเทา (Gray scale) การให้แสงเงากับระนาบสามเหลี่ยมเพื่อเพิ่มความรู้สึกในการรับรู้ของผู้ชมโดยการใช้เทคนิค Hill Shading การสร้างภาพกราฟิกให้สมจริงยิ่งขึ้นโดยใช้เทคนิค Gouraud Shading และแสดงภาพภูมิประเทศในรูปแบบมุมมองจริง (Perspective view)

วัตถุประสงค์ของการทำวิจัย

1. เสนอแนวทางในการจำลองแบบภูมิประเทศโดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบแทนการใช้โครงสร้างแบบกริด
2. ศึกษาทฤษฎีและวิธีการสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบชนิดดีลอนเนย์ โดยมุ่งเน้นไปที่วิธีการสร้างแบบพลวัติ
3. พัฒนาและปรับปรุงโครงสร้างข้อมูลและอัลกอริทึม สำหรับการสร้างและจัดการกับโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ
4. แสดงผลการจำลองภาพลักษณะภูมิประเทศในรูปแบบต่างๆ โดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบเป็นโครงสร้างพื้นฐาน

โครงร่างวิทยานิพนธ์

ในวิทยานิพนธ์นี้ถูกจัดแบ่งออกเป็น 6 บทหลักดังต่อไปนี้

บทที่ 1 อธิบายความเป็นมาของการจำลองภูมิประเทศแบบดิจิตอล กล่าวถึงรายละเอียดของแบบจำลองความสูงในรูปแบบเดิม (แผนที่คอนทัวร์ และ กริดคงที่) กล่าวถึงปัญหาที่เกิดขึ้น และนำเสนอแนวทางแก้ไขปัญหาโดยการใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบชนิดดีลอนเนย์ เพื่อเป็นแบบจำลองความสูงสำหรับการจำลองภูมิประเทศแบบดิจิตอล

บทที่ 2 กล่าวถึงการคำนวณเชิงเรขาคณิต (Computational Geometry) การหาขอบเขตคอนเว็กซ์ฮัล (Convex Hull) การจัดแบ่งพื้นที่โดยใช้ วอร์โรนอยไดอะแกรม (Voronoi Diagram) และ สามเหลี่ยมดีลอนเนย์ อธิบายขั้นตอนการทำงานของ 7 อัลกอริทึมสำหรับการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์และเปรียบเทียบการทำงานของแต่ละอัลกอริทึม

บทที่ 3 กล่าวถึงทฤษฎีพื้นฐานทางเรขาคณิตของอัลกอริทึม Incremental สำหรับการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ เมื่อมีการแทรกจุดใหม่เข้าสู่โครงข่าย การปรับโครงสร้างของโครงข่ายหลังการแทรกจุด ผลกระทบของการแทรกจุดที่มีต่อขอบรอบข้าง และขอบเขตจำกัดของผลกระทบที่เกิดขึ้น ทฤษฎีพื้นฐานจะได้รับการอธิบายโดยการแสดงรูปเรขาคณิตประกอบ

บทที่ 4 อธิบายโครงสร้างข้อมูลชนิดต่างๆสำหรับจัดการกับโครงข่ายสามเหลี่ยม นำเสนอโครงสร้างข้อมูลชนิด TwinEdge ซึ่งถูกใช้เป็นหลักในการทดลองทั้งหมด วิธีการและอัลกอริทึมเสริมต่างๆสำหรับการคำนวณโครงข่ายสามเหลี่ยม รวมไปถึงการสำรวจโครงข่ายเพื่อนำสามเหลี่ยมออกมาใช้งาน ในตอนท้ายจะแสดงผลการทำงานของอัลกอริทึม Incremental

บทที่ 5 นำเสนอวิธีการคำนวณโครงข่ายสามเหลี่ยมจากแบบจำลองความสูงแบบกริดเพื่อการประมาณเค้าโครงพื้นผิวของภูมิประเทศด้วยการใช้อัลกอริทึม Incremental ร่วมกับวิธีการคัดเลือกจุดสำคัญ อธิบายวิธีการลบจุดออกจากโครงข่ายสามเหลี่ยม อธิบายหลักการเชื่อมต่อโครงข่ายสามเหลี่ยมเพื่อใช้ในการเพิ่มความเร็วของการคำนวณโครงข่ายสามเหลี่ยมสำหรับแบบจำลองความสูงขนาดใหญ่ และสุดท้ายแสดงการนำเอาโครงข่ายสามเหลี่ยมมาใช้ประโยชน์ในการจำลองภาพของภูมิประเทศในรูปแบบต่างๆ

บทที่ 6 สรุปเนื้อหาทั้งหมดของวิทยานิพนธ์ สรุปผลการทดลอง กล่าวถึงปัญหา และการแก้ปัญหาที่เกิดขึ้นในงานวิจัย

บทที่ 2
โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ

วอร์โรนอยไดอะแกรม (Voronoi Diagram) และ สามเหลี่ยมดีลอนเนย์ (Delaunay Triangulation) คือโครงสร้างพื้นฐานในการคำนวณเชิงเรขาคณิต (Computational Geometry) สำหรับการจัดแบ่งพื้นที่บนระนาบ (Planar subdivision) ทั้งสองโครงสร้างต่างมีความสัมพันธ์ซึ่งกันและกัน โดยสามารถคำนวณสามเหลี่ยมดีลอนเนย์ได้จากวอร์โรนอยไดอะแกรม และในทางกลับกันก็สามารถคำนวณวอร์โรนอยไดอะแกรมได้จากสามเหลี่ยมดีลอนเนย์ ในบทนี้จะอธิบายถึงความเป็นมาและหลักการของโครงสร้างทั้งสอง จากนั้นจะนำเสนอ 7 อัลกอริทึมสำหรับสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ ในตอนท้ายจะเป็นการวิจารณ์และเปรียบเทียบประสิทธิภาพการทำงานของแต่ละอัลกอริทึม

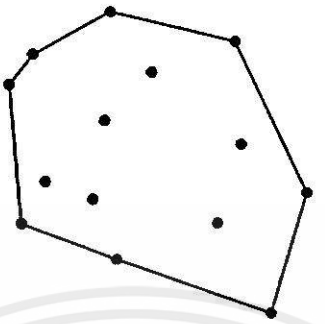
2.1 การคำนวณเชิงเรขาคณิต

การคำนวณเชิงเรขาคณิต (Computational Geometry) [22] คือวิชาที่ว่าด้วยการศึกษาทฤษฎีและหลักการของ อัลกอริทึม และ โครงสร้างข้อมูล เพื่อนำมาแก้ปัญหาทางเรขาคณิต เช่น การคำนวณหาคอนเว็กซ์ฮัล (ดังรูปที่ 2.1) [13] วอร์โรนอยไดอะแกรม สามเหลี่ยมดีลอนเนย์ เป็นต้น เมื่ออัลกอริทึมคือแนวคิดหรือวิธีการสำหรับแก้ไขปัญหาอย่างเป็นขั้นตอนในรูปของโปรแกรมคอมพิวเตอร์ และโครงสร้างข้อมูลคือวิธีการสำหรับจัดการกับข้อมูลอย่างเป็นรูปแบบซึ่งอำนวยประโยชน์ให้กับการทำงานของอัลกอริทึม อาทิเช่น การจัดเก็บข้อมูล การจัดหาตัวดำเนินการสำหรับการสร้าง การปรับปรุง และการเข้าถึงข้อมูล

องค์ประกอบพื้นฐานทางเรขาคณิตประกอบด้วย จุด (Point) ขอบ (Edge) รูปหลายเหลี่ยม (Polygon) และ รูปทรงหลายเหลี่ยม (Polyhedron) ซึ่งบรรจุอยู่ภายใน E^d หรืออาณาเขตว่างเปล่า d มิติของยูคลิเดียน (d-dimensional Euclidean space) (เมื่อ $d = 2$ หรือ 3) ขอบถูกเรียกแทนเส้นตรง รูปหลายเหลี่ยมคือระนาบซึ่งประกอบขึ้นจากการเชื่อมโยงจุดยอด (Vertex) ด้วยขอบ รูปทรงหลายเหลี่ยมประกอบขึ้นจากจุดยอด ขอบ และด้าน (ด้านคือระนาบของรูปหลายเหลี่ยม) ภายในอาณาเขต 3 มิติรูปทรงหลายเหลี่ยมสามารถถูกคลี่ออก และวางราบบนระนาบ (จาก 3 มิติเป็น 2 มิติ) ผลลัพธ์ของรูปทรงหลายเหลี่ยมที่ถูกคลี่ออกนี้คือ การจัดแบ่งพื้นที่ระนาบ

นิยามที่ 1 กำหนดให้ S คือเซ็ตของจุดภายในอาณาเขต E^d คอนเว็กซ์ฮัล (Convex Hull) หรือ CH(S) ของเซ็ต S คือ ขอบเขตคอนเว็กซ์ (Convex boundary) ที่เล็กที่สุดซึ่งได้บรรจุสมาชิก

ของเซ็ต S ทั้งหมดไว้ภายใน ใน 2 มิติขอบเขตคอนเว็กซ์คือระนาบโพลิกอน ใน 3 มิติขอบเขตคอนเว็กซ์คือรูปทรงโพลิฮีดรอน



รูปที่ 2.1 ขอบเขตคอนเว็กซ์ฮัลของเซ็ตจุดใน 2 มิติ

2.2 วอร์โรนอยไดอะแกรม

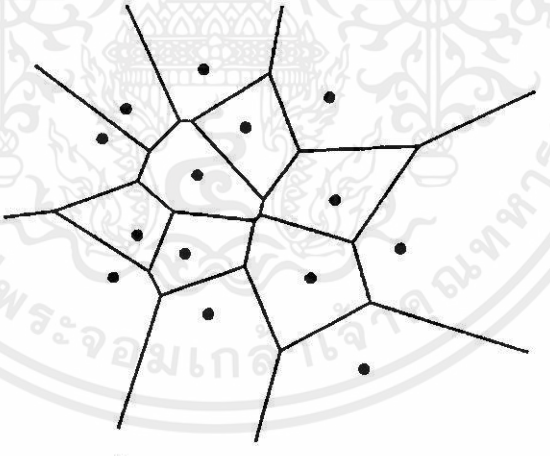
วอร์โรนอยไดอะแกรม (Voronoi Diagram) คือ โครงสร้างทางเรขาคณิตสำหรับการจัดแบ่งพื้นที่ซึ่งได้บรรจุเซ็ตของจุดข้อมูลออกเป็นพื้นที่ย่อยๆ วอร์โรนอยไดอะแกรมถูกตั้งชื่อขึ้นตามชื่อของ M. G. Voronoi นักคณิตศาสตร์ผู้ค้นพบโครงสร้างนี้ในปี ค.ศ. 1908 [29] อย่างไรก็ตามเมื่อย้อนหลังกลับไปอีกเมื่อปี ค.ศ. 1850 ก็ได้มีนักคณิตศาสตร์อีกคนหนึ่งที่มีชื่อว่า G. L. Dirichlet [5] ทำการศึกษาปัญหาดังกล่าวนี้เช่นกัน ดังนั้นในบางครั้งวอร์โรนอยไดอะแกรมจึงถูกเรียกว่า Dirichlet tessellation

นิยามที่ 2 เมื่อจุด s ภายในเซ็ต S ถูกเรียกว่า "ที่ตั้ง" (site) วอร์โรนอยไดอะแกรมของ S หรือ VR(S) คือการแบ่งอาณาเขต E^d ออกเป็นส่วนย่อยๆ (convex regions) หนึ่งส่วนย่อยต่อหนึ่งที่ตั้ง ดังนั้นจุดใดๆซึ่งบรรจุอยู่ภายในแต่ละส่วนย่อยจะมีตำแหน่งอยู่ใกล้กับที่ตั้ง s ของส่วนย่อยนั้นมากกว่าที่ตั้งอื่นๆ

วอร์โรนอยไดอะแกรมได้ถูกนำไปประยุกต์ใช้งานอย่างกว้างขวาง กับหลากหลายสาขาวิชา บางสาขาวิชาได้มีการตั้งชื่อของวอร์โรนอยไดอะแกรมขึ้นใหม่ เช่น การประยุกต์ใช้งานทางด้านอุตุนิยมวิทยา โดย A. H. Thiessen ซึ่งใช้รูปหลายเหลี่ยมที่ชื่อ Thiessen [27] สำหรับแบ่งขอบเขตพื้นที่รับผิดชอบของสถานีตรวจอากาศ ขอบข่ายการเก็บข้อมูลของแต่ละสถานีถูกแทนด้วยพื้นที่ของรูปหลายเหลี่ยม แต่ละด้านของรูปหลายเหลี่ยมกำหนดขึ้นจากการแบ่งครึ่งระยะทางระหว่างสถานีซึ่งมีที่ตั้งอยู่ใกล้เคียงกัน นอกจากนั้นวอร์โรนอยไดอะแกรมยังรู้จักกันในชื่อของ Wigner-Seitz cells [34] ในงานด้านการผสมโลหะ (Metallurgy) และอีกชื่อหนึ่งคือ Blum's transform [3] ในงานด้านชีววิทยาและโสตน์วิทยาการ (Visual Science)

ในรูปที่ 2.2 แสดงรูปหลายเหลี่ยมวอร์โรนอยซึ่งประกอบขึ้นจากเส้นตรง ซึ่งแบ่งครึ่งเส้นประที่เชื่อมระหว่างจุดกลางของรูปหลายเหลี่ยมกับจุดรอบข้าง เส้นตรงแบ่งครึ่งและเส้นประเชื่อมจุดต่างตั้งฉากซึ่งกันและกัน เมื่อเราใช้นิยามของการสร้างรูปหลายเหลี่ยมวอร์โรนอยนี้กับเซ็ตของจุดใดๆในพื้นที่หนึ่ง พื้นที่นี้จะถูกครอบคลุมไปด้วยรูปหลายเหลี่ยมวอร์โรนอยเรียงติดต่อกัน วอร์โรนอยไดอะแกรมสำหรับเซ็ตของจุดใดๆแสดงดังรูปที่ 2.3 จะสังเกตเห็นได้ว่ารูปหลายเหลี่ยมซึ่งอยู่บริเวณขอบของไดอะแกรมจะมีด้านหนึ่งเปิดออกเสมอ เพราะไม่มีจุดอื่นใดตั้งอยู่ในทิศทางเปิดนั้น

รูปที่ 2.2 โครงสร้างพื้นฐานของรูปหลายเหลี่ยมวอร์โรนอย

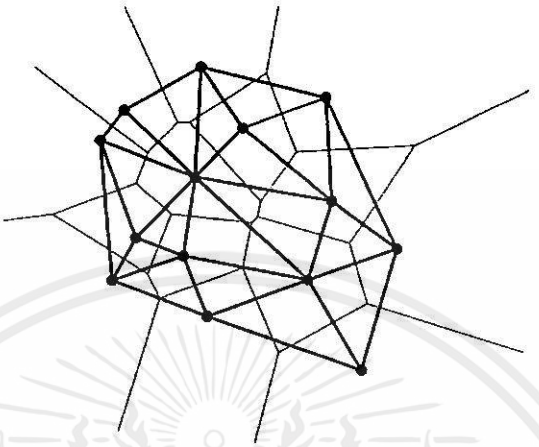


รูปที่ 2.3 วอร์โรนอยไดอะแกรมกับเซ็ตของจุด

2.3 โครงข่ายสามเหลี่ยมดีลอนเนย์

โครงข่ายสามเหลี่ยมดีลอนเนย์ (Deluanay Triangulation) มีโครงสร้างที่สัมพันธ์โดยตรงกับวอร์โรนอยไดอะแกรม โครงข่ายสามเหลี่ยมดีลอนเนย์ตั้งชื่อขึ้นตามชื่อของ B. Delaunay [4] ผู้ซึ่งค้นพบความสัมพันธ์นี้เป็นคนแรก ถ้าเราใช้วอร์โรนอยไดอะแกรมเป็นโครงสร้างพื้นฐาน เราสามารถสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ได้โดยการลากเส้นตรงระหว่างจุดกลางของรูปหลายเหลี่ยมหลักซึ่งเรากำลังพิจารณากับจุดกลางของรูปหลายเหลี่ยมรอบข้างซึ่งมีด้านที่สัมผัสกัน เมื่อ

เชื่อมจุดดังนิยามดังกล่าวจนครบทุกจุด เราก็จะได้โครงข่ายของรูปสามเหลี่ยมชนิดดีลอนเนย์ครอบคลุมทั่วพื้นที่ ความสัมพันธ์ระหว่างวอร์โรนอยไดอะแกรมและโครงข่ายลามเหลี่ยมดีลอนเนย์แสดงดังรูปที่ 2.4

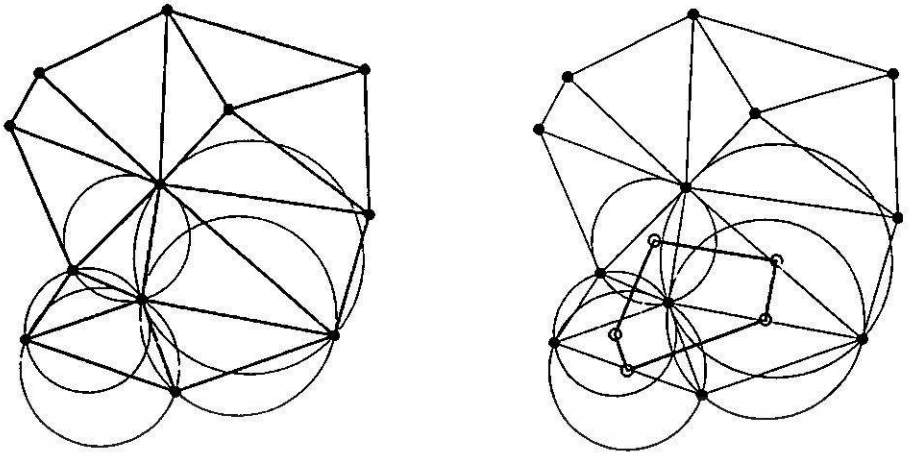


รูปที่ 2.4 ความสัมพันธ์ระหว่างวอร์โรนอยไดอะแกรมและโครงข่ายสามเหลี่ยมดีลอนเนย์

เพื่อความรวดเร็ว เราสามารถสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ได้โดยที่ไม่จำเป็นต้องสร้างวอร์โรนอยไดอะแกรมขึ้นมาก่อน เพราะในทางปฏิบัติแล้วการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์กระทำได้ง่ายกว่าการสร้างวอร์โรนอยไดอะแกรม การสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์ต้องตรงตามกฎเกณฑ์ที่เรียกว่า "Circle criterion" หรือการปลอดจุดใดๆในวงกลมทดสอบของสามเหลี่ยมทุกรูปในโครงข่าย

นิยามที่ 3 โครงข่ายสามเหลี่ยมดีลอนเนย์ใน E^2 ประกอบด้วยรูปสามเหลี่ยมซึ่งไม่ซ้อนทับกัน ไม่มีจุดข้อมูลใดบรรจุอยู่ภายในวงกลมทดสอบของรูปสามเหลี่ยมทุกรูปในโครงข่าย วงกลมทดสอบถูกสมมุติขึ้นจากจุดยอด (Vertex) ทั้ง 3 ของรูปสามเหลี่ยม

โครงข่ายสามเหลี่ยมดีลอนเนย์จะเป็นแบบ Unique [26] หรือไม่มีรูปสามเหลี่ยมใดในโครงข่ายที่มีรูปร่างซ้ำกัน เมื่อมีเพียงแค่จุด 3 จุดเท่านั้นที่บรรจุอยู่บนเส้นรอบวงของวงกลมทดสอบของทุกรูปสามเหลี่ยมในโครงข่าย อย่างไรก็ตามก็อาจเกิดเหตุการณ์ที่มีจุดมากกว่า 3 จุดอยู่บนเส้นรอบวงของวงกลมได้ แต่ยังคงเป็นโครงข่ายสามเหลี่ยมแบบดีลอนเนย์เพียงแต่ไม่เป็นแบบ Unique เท่านั้น โครงข่ายสามเหลี่ยมดีลอนเนย์และวงกลมทดสอบแสดงดังรูปที่ 2.5(ก) และในรูปที่ 2.5(ข) จะแสดงให้เห็นถึงความสัมพันธ์ระหว่างวงกลมทดสอบกับรูปหลายเหลี่ยมของวอร์โรนอยไดอะแกรม โดยที่จุดศูนย์กลางของวงกลมทดสอบแต่ละรูปคือจุดยอดของรูปหลายเหลี่ยม



(ก) (ข)

รูปที่ 2.5 (ก) โครงข่ายสามเหลี่ยมดีลอนเนย์และวงกลมทดสอบ (ข) ความสัมพันธ์ระหว่างวงกลมทดสอบและรูปหลายเหลี่ยมวอร์โรนอย

2.4 วิธีการสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ

เมื่อเซ็ตของจุดใดๆซึ่งกระจัดกระจายอยู่ภายในพื้นที่หนึ่งๆ ถูกเชื่อมต่อให้เป็นโครงข่ายสามเหลี่ยม (Triangulated network) โครงข่ายนี้จะมีคุณสมบัติเป็นโครงข่ายแบบไม่เป็นระเบียบ (Irregular network) ถ้าต้องการสร้างโครงข่ายสามเหลี่ยมแบบเป็นระเบียบ (Regular network) จากโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบสามารถทำได้โดยการแทรกข้อมูล (Interpolation) ระหว่างจุดที่กระจัดกระจายนั้นในระยะห่างที่เท่าๆกันในแนวแกน x และ y

มีงานวิจัยมากมายได้นำเสนอเทคนิคต่างๆในการสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ ในหัวข้อนี้จะอธิบายวิธีการต่างๆในการสร้างโครงข่ายสามเหลี่ยมสำหรับเป็นโครงสร้างพื้นฐานของพื้นผิวใดๆ โดยเฉพาะพื้นผิวของภูมิประเทศ (Terrain surface) ซึ่งมีความซับซ้อนและเปลี่ยนแปลงอย่างไม่เป็นระเบียบ การสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ถือเป็นเทคนิคพื้นฐานในการสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ เพราะข้อมูลภายในโครงข่ายสามารถรองรับต่อโครงสร้างที่สลับซับซ้อนของแบบจำลองภูมิศาสตร์ได้เป็นอย่างดี วิธีการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์สามารถจำแนกได้ดังนี้

1. วิธีการแบบสถิต
   - 1.1 อัลกอริทึม Radial Sweep
   - 1.2 อัลกอริทึม Recursive Split
   - 1.3 อัลกอริทึม Divide-and-Conquer
   - 1.4 อัลกอริทึม Step-by-Step
   - 1.5 อัลกอริทึม Modified Hierarchical

2. วิธีการแบบพลวัต

2.1 อัลกอริทึม Incremental

2.2 อัลกอริทึม Incremental Delete-and-Build

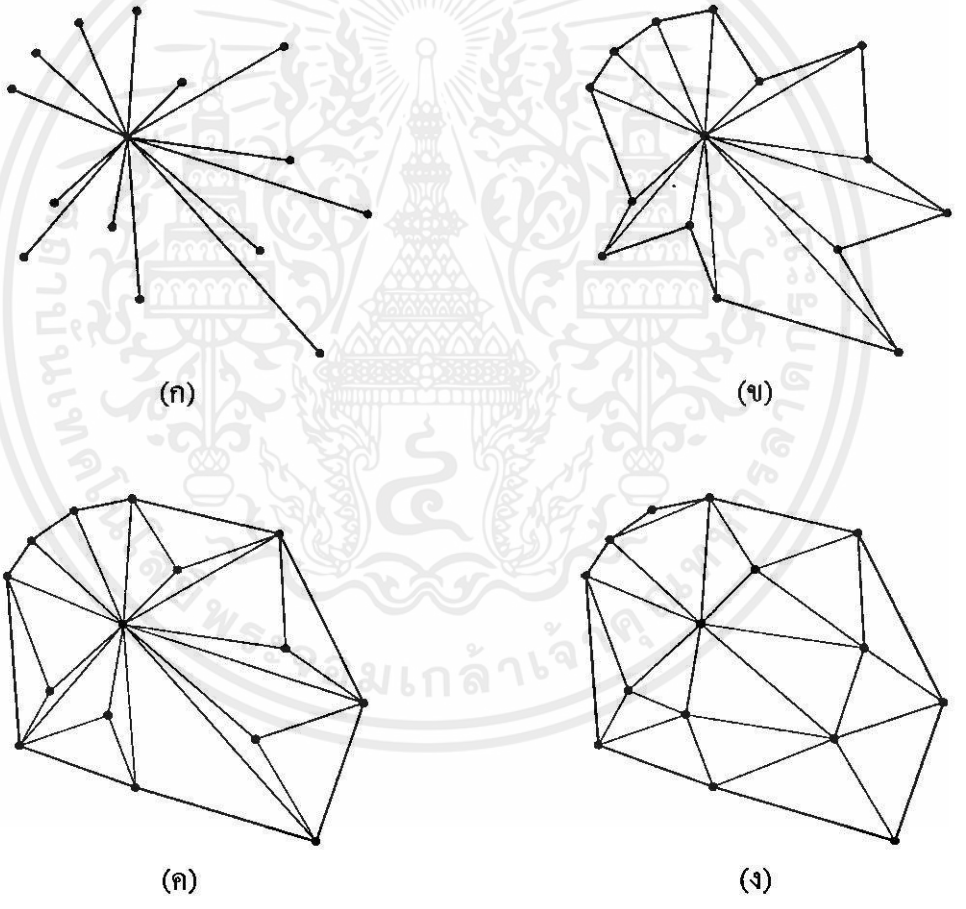
เห็นได้ว่าวิธีการสร้างโครงข่ายสามเหลี่ยมสามารถแบ่งออกเป็น 2 กลุ่มหลักคือ วิธีการแบบสถิต (Static triangulation) และวิธีการแบบพลวัต (Dynamic triangulation) วิธีการสร้างแบบสถิตหมายถึงโครงข่ายสามเหลี่ยมจะยังไม่เกิดขึ้นอย่างสมบูรณ์ จนกระทั่งจุดทุกจุดจากเซ็ตข้อมูลจะถูกเพิ่มเข้าไปในโครงข่ายจนหมดสิ้น ไม่มีการคัดเลือกจุดเกิดขึ้นขณะทำการเพิ่มจุด และโครงข่ายจะไม่เป็นไปตามนิยามของดีลอนเนย์จนกระทั่งการสร้างโครงข่ายทั้งหมดสำเร็จลง สำหรับวิธีการแบบพลวัต รูปแบบของโครงข่ายสามเหลี่ยมจะเป็นไปตามนิยามของดีลอนเนย์อยู่ตลอดเวลาตั้งแต่เริ่มต้นจนจบกระบวนการ ทุกครั้งที่มีจุดใหม่ถูกเพิ่มเข้าสู่โครงข่ายโครงสร้างของโครงข่ายจะได้รับการปรับปรุง และสามารถคัดเลือกจุดซึ่งมีส่วนร่วมกับแบบจำลองได้

2.4.1 อัลกอริทึม Radial Sweep

อัลกอริทึม Radial Sweep ถูกตีพิมพ์เป็นครั้งแรกโดย [20] และหลังจากนั้นโดย [21] ข้อมูลสำหรับการสร้างโครงข่ายสามเหลี่ยมคือเซ็ตของจุดซึ่งกระจายอยู่บนพื้นที่ใดๆ วิธีการสร้างสามารถอธิบายอย่างเป็นขั้นตอนได้ดังนี้

1. เลือกจุดที่อยู่ใกล้จุดศูนย์กลางมากที่สุด แล้วคำนวณหาระยะทางและมุมที่กระทำระหว่างจุดศูนย์กลางนี้กับจุดอื่นๆที่เหลือ จากนั้นจัดเรียงลำดับจุดตามค่าองศาของมุมและระยะทางที่คำนวณได้ รูปที่ 2.6(ก) แสดงเส้นรัศมี (the radiating lines) ซึ่งถูกกำหนดขึ้นจากจุดศูนย์กลางไปยังจุดโดยรอบทุกจุด
2. Neighbouring lines หรือเส้นตรงปิดล้อมจะถูกสร้างขึ้นตรงข้ามกับจุดศูนย์กลาง โดยการเชื่อมจุดปลายของเส้นรัศมี (อีกด้านจากจุดศูนย์กลาง) เข้าหากัน ดังรูปที่ 2.6(ข) ถ้ามีจุดใดที่ทับซ้อนอยู่บนเส้นรัศมีการเชื่อมต่อเส้นตรงปิดล้อมนี้จะยังคงดำเนินต่อไป โดยที่เส้นปิดล้อมจะซ้อนทับกับเส้นรัศมี
3. จุดทุกจุดซึ่งอยู่บนเส้นตรงแสดงอาณาเขตจะถูกจัดเรียงเก็บไว้ในลำดับข้อมูลอาณาเขต (boundary list) ส่วนหนึ่งของจุดเหล่านี้จะถูกรวมกับจุดซึ่งอยู่ในลำดับถัดไปอีก 2 จุด ถ้าการรวมกันของจุดทั้งสามจุดนี้เป็นการสร้างรูปสามเหลี่ยม ข้อมูลรูปสามเหลี่ยมใหม่จะถูกเพิ่มเข้าสู่ลำดับสามเหลี่ยม (triangle list) กระบวนการเช่นนี้จะดำเนินต่อไปเรื่อยๆ จนกระทั่งไม่สามารถสร้างรูปสามเหลี่ยมซึ่งอยู่โดยรอบอาณาบริเวณได้แล้ว เมื่อเสร็จขั้นตอนนี้ขอบเขตของเซ็ตจุดจะเปลี่ยนเป็นขอบเขตที่เรียกว่าคอนเว็กซ์ฮัล ดังรูปที่ 2.6(ค)

4. โครงข่ายสามเหลี่ยมที่ได้จากขั้นตอนที่ 3 เป็นโครงข่ายที่ไม่มีการซ้อนทับกัน อย่างไรก็ตามรูปสามเหลี่ยมผลลัพธ์ที่ได้ยังถือว่าไม่สมบูรณ์ เพราะมีลักษณะที่ผอมและยาวเหมือนลิ่ม ดังนั้นจึงต้องมีการปรับโครงสร้างของโครงข่ายใหม่ รูปสามเหลี่ยม 2 รูปซึ่งอยู่ติดกันจะเสมือนเป็นรูปสี่เหลี่ยมด้านไม่เท่า (quadrilateral) 1 รูป โดยใช้หลักการเลือกเส้นทะแยงมุมของรูปสี่เหลี่ยมด้านไม่เท่าที่มีระยะทางที่สั้นที่สุด จะทำให้ได้รูปสามเหลี่ยมใหม่ที่สมบูรณ์ยิ่งขึ้น ถ้าเส้นทะแยงมุมที่ปรากฏในรูปสี่เหลี่ยมด้านไม่เท่านั้นยาวกว่าอีกเส้น การปรับปรุงจะเกิดขึ้นโดยเลือกเอาเส้นทะแยงมุมที่สั้นกว่าแทน กระบวนการสลับเส้นทะแยงจะดำเนินต่อไปจนกระทั่งไม่มีเส้นทะแยงมุมให้สลับต่อไปอีก โครงข่ายสามเหลี่ยมผลลัพธ์แสดงดังรูปที่ 2.6(ง)



รูปที่ 2.6 อัลกอริทึม Radial Sweep (ก) เส้นรัศมีพุ่งตรงออกจากจุดศูนย์กลางแพร่ขยายไปสู่จุดอื่นๆ (ข) เส้นตรงรัศมีถูกเชื่อมจุดปลายเข้าหากัน (ค) สร้างคอนเว็กซ์ฮัลปิดล้อมอาณาบริเวณ (ง) โครงข่ายสามเหลี่ยมผลลัพธ์

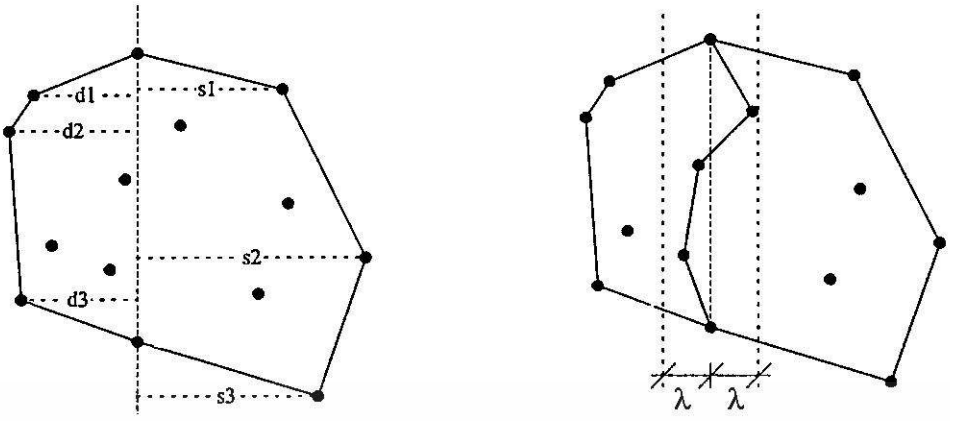
ในทางทฤษฎีแล้วการสร้างโครงข่ายสามเหลี่ยมโดยใช้อัลกอริทึม Radial Sweep นี้จะให้โครงข่ายสามเหลี่ยมที่เกือบเป็นดีลอนเนย์ เนื่องจากโครงสร้างของโครงข่ายไม่สอดคล้องกับหลัก

การของวงกลมทดสอบทั้งหมด (ตามนิยามที่ 2) แต่หากใช้หลักการทดสอบผลรวมของมุมภายใน Max-Min angle sum [15] (อธิบายในหัวข้อที่ 3.2) หรือหลักการ Incircle ของ [10] จะทำให้โครงข่ายผลลัพธ์เป็นแบบดีลอนเนย์อย่างสมบูรณ์

2.4.2 อัลกอริทึม Recursive Split

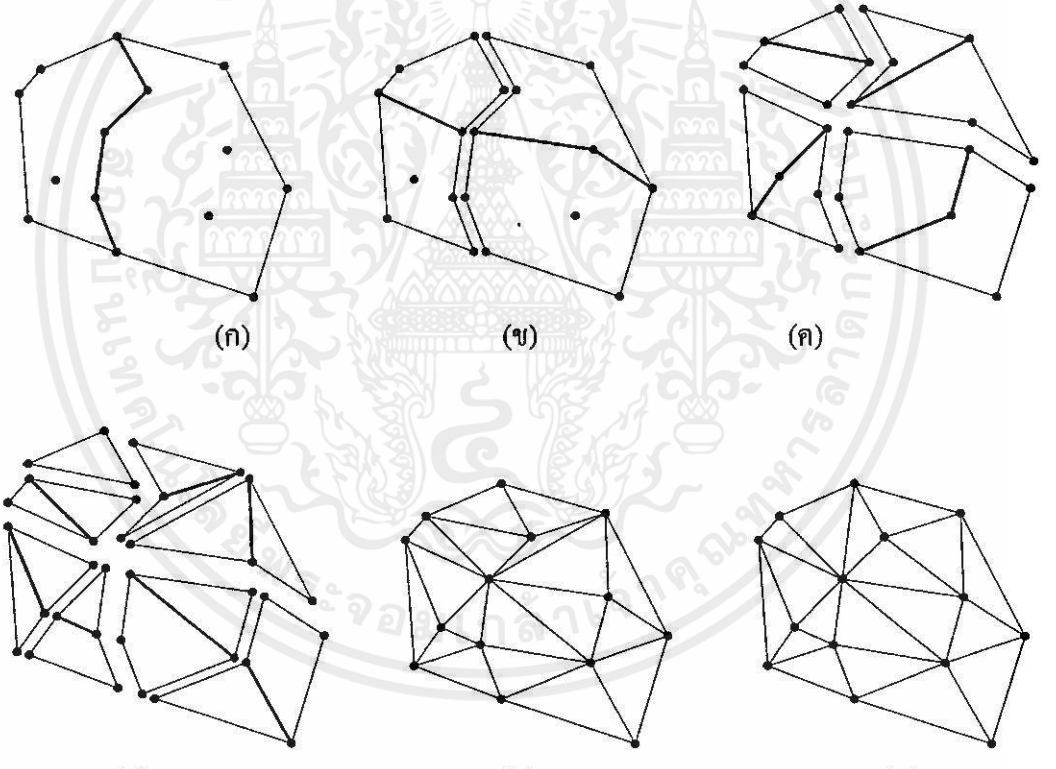
อัลกอริทึมถูกนำเสนอเป็นครั้งแรกโดย [18] จุดประสงค์คือการแบ่งพื้นที่ทั้งหมดออกเป็นพื้นที่ย่อยๆเล็กลงเรื่อยๆแบบรีเคอร์สีฟ (Recursive) จนกระทั่งไม่สามารถแบ่งต่อไปได้อีก พื้นที่ย่อยที่เล็กที่สุดนี้ประกอบขึ้นจากจุด 3 จุดซึ่งคือรูปสามเหลี่ยม ขั้นตอนการทำงานของอัลกอริทึม Recursive Split อธิบายได้ดังนี้

1. พื้นที่ทั้งหมดซึ่งได้บรรจุเซ็ตของจุดไว้ภายในถูกแบ่งครึ่งออกเป็น 2 ส่วน และต้องรักษารูปร่างของพื้นที่ของเซ็ตจุดที่ถูกแบ่งนี้ให้ใกล้เคียงกับรูปวงกลมให้มากที่สุดโดยความกลม (circularity) สามารถคำนวณได้จากการวัดค่าผลคูณรวม Π ของระยะทาง (d_n, s_n) ระหว่างจุดซึ่งอยู่บนขอบเขตและเส้นตรงสมมุติ ดังรูปที่ 2.7(ก) หลังจากนั้นขอบเขตที่แท้จริงของพื้นที่ย่อยทั้งสองจะถูกเลือก โดยการกำหนดค่าระยะห่าง λ ที่เหมาะสม เพื่อเลือกจุดที่จะใช้เป็นจุดกำหนดขอบเขตรูปยัก (zig-zag boundary) ของพื้นที่ย่อยทั้งสอง ดังรูปที่ 2.7(ข)
2. ส่วนย่อยแต่ละส่วนจะถูกแบ่งให้เป็น 2 ส่วนย่อยลงไปอีกโดยอาศัยหลักการดังขั้นตอนที่ 1 จนกระทั่งพื้นที่ย่อยนั้นเหลือจุดเพียง 3 จุด ในที่สุดโครงข่ายสามเหลี่ยมเริ่มต้นซึ่งประกอบด้วยรูปสามเหลี่ยมที่ไม่ซ้อนทับกันจึงเกิดขึ้น หลักการของการแบ่งพื้นที่ย่อยไปเรื่อยๆนี้ผู้นำเสนออัลกอริทึม [18] ได้แนวคิดมาจากหลักการเรียงข้อมูลแบบ Quick-Sort [25] ขั้นตอนการทำงานของอัลกอริทึม Recursive Split ทั้งหมดแสดงดังรูปที่ 2.8
3. เนื่องจากข้อกำหนดของการแบ่งพื้นที่ออกเป็นสองส่วยย่อยดังขั้นตอนที่ 1 ทำให้รูปร่างของสามเหลี่ยมผลลัพธ์มีลักษณะที่ยาวและแหลม ดังนั้นโครงข่ายสามเหลี่ยมนี้จึงถูกเรียกว่าโครงข่ายสามเหลี่ยมเริ่มต้น ซึ่งต้องใช้หลักเกณฑ์ของดีลอนเนย์ช่วยในการปรับปรุงโครงสร้างเพื่อให้ได้ลักษณะของรูปสามเหลี่ยมที่สมบูรณ์ที่สุด เมื่อรูปสี่เหลี่ยมด้านไม่เท่าในโครงข่ายจะได้รับการทดสอบมุมภายใน มุม α และ β ถูกกำหนดขึ้นเพื่อการคำนวณ ดังรูปที่ 2.9 เมื่อ α คือมุมที่เล็กที่สุดของสามเหลี่ยมเริ่มต้น และ β คือมุมที่เล็กที่สุดหลังจากการเลือกสามเหลี่ยมอันใหม่ ถ้า $\alpha < \beta$ สามเหลี่ยมสองรูปอันใหม่จะเกิดขึ้นโดยการสลับเส้นทะแยงมุม การทดสอบและปรับปรุงจะดำเนินต่อไปจนกระทั่งไม่มีการเปลี่ยนแปลงใดๆเกิดขึ้นอีกในโครงข่าย



(ก) (ข)

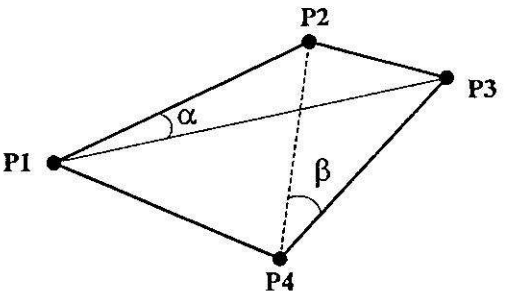
รูปที่ 2.7 วิธีการแบ่งพื้นที่ออกเป็นสองส่วน (ก) พื้นที่ถูกแบ่งออกเป็นสองส่วนย่อยโดยเส้นตรงสมมุติ (ข) เส้นแบ่งอาณาเขตที่แท้จริง



(ก) (ข) (ค)

(ง) (จ) (ช)

รูปที่ 2.8 อัลกอริทึม Recursive Split (ก-ง) พื้นที่ถูกแบ่งครึ่งไปเรื่อยๆจนเหลือเฉพาะพื้นที่เล็กที่สุดที่เป็นสามเหลี่ยม (จ) โครงข่ายสามเหลี่ยมเริ่มต้น (ช) โครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์เมื่อใช้กฎเกณฑ์ Max-Min angle



รูปที่ 2.9 กฎเกณฑ์ Max-Min angle sum

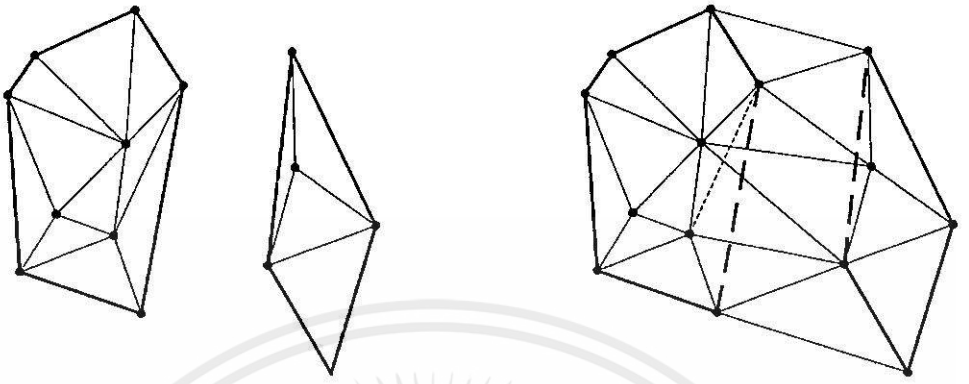
[18] ไม่ได้กล่าวถึงทฤษฎีของดีลอนเนย์แต่อย่างใด แต่อย่างไรก็ตามหลักการของการทดสอบมุมภายใน (ดังรูปที่ 2.9) นี้สอดคล้องกับกฎเกณฑ์ Max-Min angle sum [15] (กฎเกณฑ์นี้สามารถใช้แทนกฎเกณฑ์การทดสอบด้วยวงกลมได้) จุดด้อยของอัลกอริทึม Recursive Split คือ มุมทั้ง 8 ภายในของสี่เหลี่ยมด้านไม่เท่าแต่ละรูปต้องได้รับการทดสอบทั้งหมดทำให้สิ้นเปลืองเวลาในการคำนวณเป็นอย่างมาก

2.4.3 อัลกอริทึม Divide-and-Conquer

หลักการทำงานของอัลกอริทึม Divide-and-Conquer คล้ายกับอัลกอริทึม Recursive Split คือ เซ็ตของข้อมูลจะถูกแบ่งออกเป็น 2 ส่วนย่อย แต่ละส่วนจะถูกแบ่งออกเป็น 2 ส่วนย่อยๆต่อไปเรื่อยๆ จนกระทั่งไม่สามารถแบ่งได้อีก หลังจากนั้นคู่ของสองส่วนย่อยจะถูกเชื่อม (merged) เข้าหากันกลับคืนย้อนกลับไปจนกระทั่งรวมกันเป็นโครงข่ายสามเหลี่ยมดีลอนเนย์ในที่สุด อัลกอริทึม Divide-and-Conquer ได้รับการนำเสนอเป็นครั้งแรกโดย [17] และอีกครั้งโดย [10] ซึ่งใช้โครงสร้างข้อมูลในการจัดการที่แตกต่างกัน ทั้งสองงานวิจัยต่างอ้างว่าประสิทธิภาพการปฏิบัติงานของอัลกอริทึมนี้เท่ากับ $O(n \log n)$ ซึ่งสามารถอธิบายขั้นตอนการทำงานได้ดังนี้

1. อัลกอริทึม Divide-and-Conquer จะแบ่งข้อมูลออกเป็น 2 ส่วน ด้วยจำนวนจุดที่เท่าๆกัน (ไม่ขึ้นกับรูปร่างของพื้นที่ซึ่งต้องคล้ายคลึงกับรูปวงกลม เช่นเดียวกับอัลกอริทึม Recursive Split) ข้อมูลที่ถูกแบ่งแล้วนี้จะถูกแบ่งย่อยออกอีกเป็น 2 ส่วน ดังรูปที่ 2.10(ก) ไปเรื่อยๆจนกระทั่งเหลือจำนวนจุดที่น้อยที่สุดคือ 4 จุด
2. รูปสามเหลี่ยมจะถูกสร้างขึ้นในระดับที่ต่ำที่สุด เมื่อจุดสี่จุดนี้คือรูปสี่เหลี่ยมด้านไม่เท่าซึ่งประกอบกันขึ้นจากรูปสามเหลี่ยม 2 รูป โดยในขณะที่กระบวนการแบ่งข้อมูลกำลังดำเนินอยู่นั้น ขอบเขตของพื้นที่ (คอนเว็กซ์ฮัล) สำหรับข้อมูลย่อยนั้นก็ได้ถูกกำหนดขึ้นตามไปด้วย คอนเว็กซ์ฮัลจะถูกใช้ประโยชน์เมื่อทำการเชื่อมต่อ 2 โครงข่ายย่อยเข้าหากัน ดังรูปที่ 2.10(ข) ในระหว่างนี้การทดสอบนิยามของดีลอนเนย์จะ

ถูกปฏิบัติพร้อมไปด้วย ดังนั้นหลังจากที่สองโครงข่ายย่อยสุดท้ายได้รับการเชื่อมเข้าหากันเสร็จแล้ว โครงข่ายสามเหลี่ยมดีลอนเนย์ผลลัพธ์จึงสำเร็จอย่างสมบูรณ์



(ก) (ข)

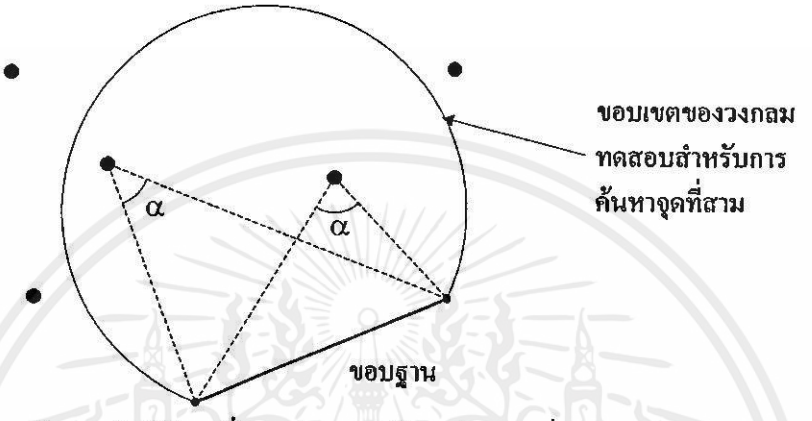
รูปที่ 2.10 อัลกอริทึม Divide-and-Conquer (ก) โครงข่ายสามเหลี่ยม 2 ส่วนซึ่งถูกกำหนดอาณาเขตโดยคอนเว็กซ์ฮัล (ข) โครงข่ายย่อย 2 ส่วนสุดท้ายถูกเชื่อมเข้าหากัน (เส้นประแสดงขอบเขตของคอนเว็กซ์ฮัลที่ต้องกำจัดออกจากโครงข่าย)

2.4.4 อัลกอริทึม Step-by-Step

อัลกอริทึม Step-by-Step [19] ถือเป็นวิธีการที่พื้นฐานที่สุดในการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ การสร้างโครงข่ายเริ่มต้นจากด้านใดด้านหนึ่งของขอบเขตข้อมูล โดยการกำหนดขอบเริ่มต้นจากจุด 2 จุด จากนั้นทำการค้นหาจุดที่สามเพื่อนำมาประกอบกันขึ้นเป็นรูปสามเหลี่ยม ขอบใหม่ 2 ขอบจะถูกสร้างขึ้นเพื่อเชื่อมจุดที่สามและจะกลายเป็นขอบเริ่มต้นเพื่อค้นหาจุดที่สามมาประกอบต่อไป กระบวนการเช่นนี้จะดำเนินต่อไปจนกระทั่งครอบคลุมทั่วพื้นที่และเป็นโครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์ในที่สุด ดังรายละเอียดต่อไปนี้

1. ขอบฐานเริ่มต้นจะถูกเลือกจากด้านใดด้านหนึ่งของคอนเว็กซ์ฮัล เมื่อสมมุติให้ขอบฐานนี้เป็นเส้นผ่านวงกลมใดๆในโครงข่ายสามเหลี่ยม โดยมีจุดปลายทั้งสองของขอบอยู่บนเส้นรอบวงของวงกลมสมมุติ ตามนิยามของดีลอนเนย์วงกลมสมมุตินี้ต้องไม่มีจุดข้อมูลใดบรรจุอยู่ภายใน
2. จุดที่สามจะถูกค้นหาเพื่อมาประกอบกับขอบฐานเป็นรูปสามเหลี่ยมดีลอนเนย์ โดยการคำนวณมุม α ที่จุดที่สามนี้กระทำกับเส้นฐานดังรูปที่ 2.11 เมื่อจุดใดทำให้เกิดมุมที่มีขนาดกว้างที่สุด จุดนั้นก็คือจุดที่จะนำมาประกอบเป็นรูปสามเหลี่ยม
3. สร้างขอบขึ้นใหม่ 2 ขอบเพื่อเชื่อมต่อกับจุดที่ได้รับเลือก ขอบทั้งสองนี้จะกลายเป็นขอบฐานอันใหม่เพื่อเริ่มต้นในการหาจุดที่สามเพื่อมาประกอบกันต่อไป ขอบฐานที่ปรากฏก่อนหน้านี้ในฐานข้อมูลจะถูกลบทิ้งไป และใส่ขอบฐานใหม่เข้าไปแทน

4. กระบวนการจะดำเนินต่อไปดังเช่นขั้นตอนที่ 2 และ 3 ขยายอาณาเขตของขอบฐานออกไปเรื่อยๆจนครบพื้นที่ทั้งหมดของเซ็ตข้อมูล และระหว่างการสร้างโครงข่ายนั้น การค้นหาจุดจะกระทำเพียงด้านใดด้านหนึ่งของเส้นฐานเท่านั้น เพราะด้านหนึ่งของขอบฐานคือเซ็ตของจุดที่ต้องการค้นหา และอีกด้านหนึ่งคือโครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์แล้ว



รูปที่ 2.11 การเลือกจุดใกล้เคียงเพื่อมาประกอบเป็นรูปสามเหลี่ยม

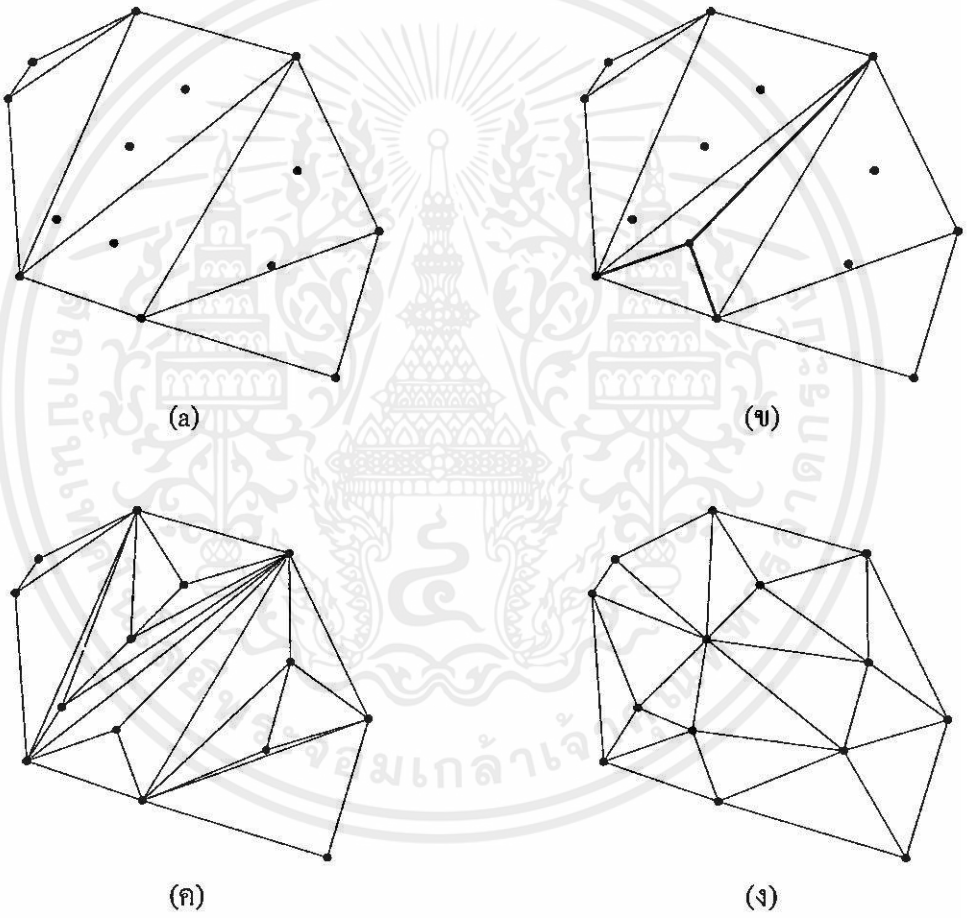
2.4.5 อัลกอริทึม Modified Hierarchical

อัลกอริทึม Modified Hierarchical มีหลักการทำงานคล้ายกับอัลกอริทึม Hierarchical structure ของ [7] เมื่อโครงสร้างข้อมูลแบบระดับชั้น (Hierarchical structure) ถูกสร้างขึ้นเพื่อเป็นโครงข่ายเบื้องต้นในการสร้างโครงข่ายสามเหลี่ยม หลังจากนั้นการจัดโครงสร้างท้องถิ่น (local transformation) หรือเรียกอีกชื่อหนึ่งว่า "การสลับขอบ" (Edge swaps) [14] จะดำเนินการปรับปรุงโครงสร้างให้ตรงตามนิยามของดีลอนเนย์ อัลกอริทึม Modified Hierarchical เกิดจากการประยุกต์วิธีการร่วมระหว่างอัลกอริทึม Radial Sweep และอัลกอริทึม Incremental (ซึ่งจะอธิบายในหัวข้อถัดไป) เมื่อโครงข่ายเบื้องต้นถูกสร้างขึ้นด้วยวิธีการแทรกจุด (Point insertion) และโครงสร้างของโครงข่ายจะถูกปรับปรุงให้ถูกต้องด้วยวิธีการสลับขอบ ขั้นตอนการสร้างทั้งหมดมีดังต่อไปนี้

1. สร้างโครงข่ายสามเหลี่ยมเบื้องต้นให้ครอบคลุมจุดข้อมูลทุกจุด (ใน [10] ใช้รูปสามเหลี่ยมขนาดใหญ่หนึ่งรูปครอบคลุมพื้นที่ข้อมูลทั้งหมด) จากรูปที่ 2.12(ก) โครงข่ายสามเหลี่ยมของคอนเว็กซ์ฮัลถูกใช้เป็นโครงข่ายสามเหลี่ยมเบื้องต้น
2. จุดแรก (ซึ่งอยู่ภายในพื้นที่) จะถูกแทรกเข้าสู่โครงข่าย และสร้างขอบใหม่ 3 ขอบเพื่อเชื่อมจุดใหม่นี้กับมุมทั้ง 3 ของรูปสามเหลี่ยมซึ่งปิดล้อมจุดนี้อยู่ ดังรูปที่ 2.12(ข)
3. จุดที่เหลืออยู่จะถูกเพิ่มเข้าสู่โครงข่ายดังเช่นวิธีการในขั้นตอนที่ 2 จนกลายเป็นโครงข่ายสามเหลี่ยมซึ่งไม่มีการซ้อนทับกันของรูปสามเหลี่ยม ดังรูปที่ 2.12(ค)

4. รูปสี่เหลี่ยมด้านไม่เท่าทุกรูปภายในโครงข่ายจะถูกทดสอบมุมภายใน (รูปที่ 2.9) และสลับขอบ (ขอบในที่นี้คือเส้นทะแยงมุมของรูปสี่เหลี่ยมด้านไม่เท่า) จนกระทั่งเป็นโครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์ ดังรูปที่ 2.12(ง)

ในขั้นตอนที่ 4 อาจถูกแบ่งย่อยเป็น 2 ขั้นตอนเพื่อเพิ่มประสิทธิภาพในการทำงาน โดยขั้นตอนแรกการทดสอบความยาวของเส้นทะแยงมุมที่สั้นที่สุดจะถูกนำมาใช้ เนื่องจากการคำนวณความยาวใช้เวลาน้อยกว่าการคำนวณมุมภายใน ผลลัพธ์ของโครงข่ายสามเหลี่ยมที่ได้ในขั้นตอนแรกนี้จะเกือบเป็นดีลอนเนย์ ในขั้นตอนสุดท้ายการทดสอบมุมภายในจะถูกนำมาใช้เพื่อตรวจสอบความถูกต้องอีกครั้งหนึ่ง



รูปที่ 2.12 อัลกอริทึม Modified Hierarchical (ก) โครงข่ายสามเหลี่ยมเบื้องต้น (ข) จุดแรกถูกแทรกเข้าสู่โครงข่าย (ค) จุดทุกจุดถูกแทรกเข้าสู่โครงข่าย (ง) โครงข่ายถูกจัดโครงสร้างใหม่ให้เป็นโครงข่ายสามเหลี่ยมดีลอนเนย์

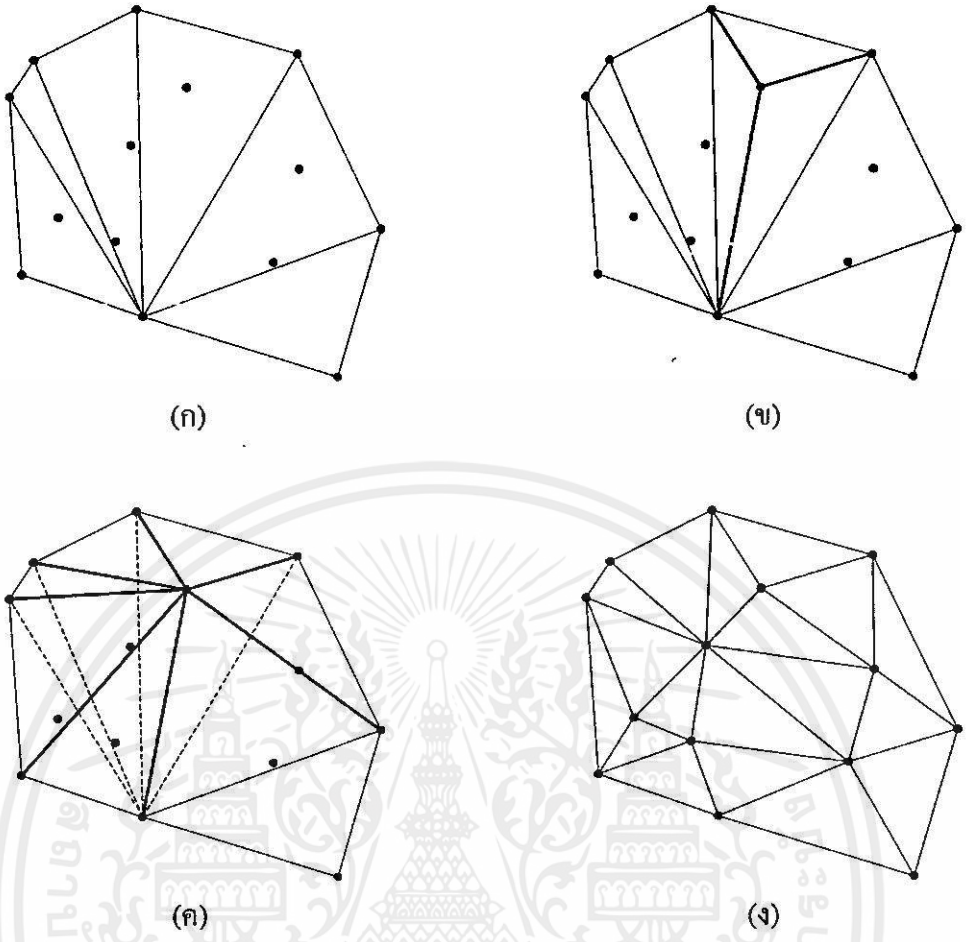
2.4.6 อัลกอริทึม Incremental

อัลกอริทึม Incremental ถูกนำเสนอโดย [17] และ [10] หลังจากนั้น [8] ใช้หลักการนี้ในการประมาณค้าโครงพื้นผิว (The surface approximation) โดยใช้หลักการสลับขอบของ [14] และ

[15] อัลกอริทึมที่คล้ายๆกันนี้ก็ได้ถูกนำเสนอโดย [9] เช่นกัน จุดเด่นที่ทำให้วิธีการแบบพลวัตเช่น อัลกอริทึม Incremental นี้แตกต่างจากวิธีการแบบสถิตดังที่ได้อธิบายไปแล้วในหัวข้อก่อนหน้านี้ คือ ความสามารถในการรักษารูปแบบของโครงข่ายสามเหลี่ยมให้เป็นแบบดีลอนเนย์อยู่ตลอดกระบวนการ โครงข่ายเริ่มต้นสำหรับวิธีการนี้คือโครงข่ายสามเหลี่ยมดีลอนเนย์ (รูปสามเหลี่ยมอย่างน้อย 1 รูป) สำหรับครอบคลุมพื้นที่ของจุดข้อมูลทั้งหมด เมื่อมีจุดใหม่เพิ่มเข้าสู่โครงข่าย โครงสร้างของโครงข่ายจะได้รับการปรับปรุงจนกระทั่งรูปสามเหลี่ยมทุกรูปสอดคล้องกับนิยามวงกลมทดสอบของดีลอนเนย์

1. ดังเช่นอัลกอริทึม Modified Hierarchical ที่ต้องสร้างโครงข่ายสามเหลี่ยมเริ่มต้นก่อน ดังรูปที่ 2.13(ก) โดยใช้โครงข่ายสามเหลี่ยมของคอนเว็กซ์ฮัลเป็นโครงข่ายเริ่มต้น และโครงข่ายนี้จำเป็นต้องสอดคล้องกับนิยามของการทดสอบมุมภายในด้วย แต่สำหรับโครงข่ายเริ่มต้นที่ใช้ในวิทยานิพนธ์เล่นนี้จะใช้โครงข่ายที่เกิดจากสามเหลี่ยม 2 รูปประกบด้านยาวเข้าหากันเป็นรูปสี่เหลี่ยมด้านเท่า 1 รูปครอบคลุมพื้นที่ข้อมูลทั้งหมด
2. จุดแรกจากภายในพื้นที่ของโครงข่ายเริ่มต้นจะถูกเพิ่มเข้าสู่โครงข่าย โดยการเชื่อมจุดเข้ากับรูปสามเหลี่ยมซึ่งปิดล้อมจุดนี้อยู่ด้วยขอบใหม่ 3 ขอบระหว่างจุดและมุมยอดของรูปสามเหลี่ยม ดังรูปที่ 2.13(ข)
3. รูปสี่เหลี่ยมด้านไม่เท่าซึ่งมีขอบเก่าของรูปสามเหลี่ยมปิดล้อมจุดใหม่เป็นเสมือนเส้นทะแยงมุม จำเป็นต้องมีการทดสอบมุมภายใน ถ้ามุมภายในไม่ตรงตามกฎเกณฑ์เส้นทะแยงมุมนี้จะต้องถูกสลับขอบ โดยใช้จุดใหม่เป็นปลายของเส้นทะแยงมุมเชื่อมไปยังปลายอีกด้านหนึ่งที่จุดของมุมตรงข้าม ผลของการสลับเส้นทะแยงมุมหรือสลับขอบนี้แสดงดังรูปที่ 2.13(ค) การทดสอบมุมภายในและการสลับขอบจะถูกกระทำแบบรีเคอร์สีฟขยายผลต่อเนื่องไปเรื่อยๆ
4. จุดที่เหลือในพื้นที่จะถูกเพิ่มเข้าสู่โครงข่ายอย่างต่อเนื่องตามขั้นตอนที่ 2 และ 3 จนเป็นโครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์ ดังรูปที่ 2.13(ง)

ข้อได้เปรียบของอัลกอริทึม Incremental เมื่อเปรียบเทียบกับวิธีการที่ผ่านมาคือ ความเป็นไปได้ที่จะเลือกเฟ้นและคำนวณหาจุดที่มีส่วนร่วมกับแบบจำลองพื้นผิว ซึ่งแทนด้วยโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ



รูปที่ 2.13 อัลกอริทึม Incremental (ก) โครงข่ายสามเหลี่ยมเริ่มต้น (ข) การแทรกจุดแรกเข้าสู่โครงข่าย (ค) เส้นทะแยงมุมที่ไม่ตรงตามนกฎเกณฑ์จะถูกสลับขอบ (ง) โครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์หลังจากการแทรกจุดสุดท้าย

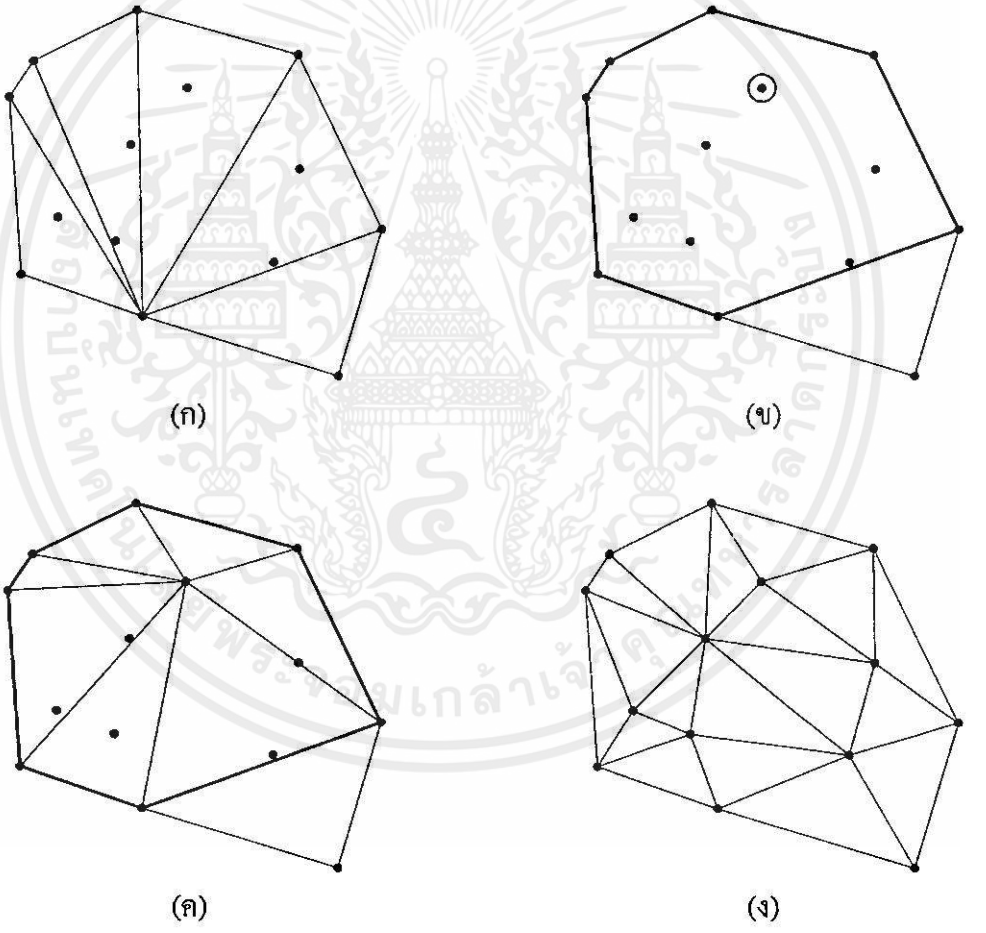
2.4.7 อัลกอริทึม Incremental Delete-and-Build

อัลกอริทึมนี้รู้จักในอีกชื่อหนึ่งว่า Watson's algorithm สำหรับการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์ใน n มิติ [30] มีข้อแตกต่างจากอัลกอริทึม Incremental คือไม่มีการสลับขอบเกิดขึ้นระหว่างการดำเนินการเพิ่มจุด เมื่อมีจุดใหม่เพิ่มเข้าสู่โครงข่ายสามเหลี่ยม ขอบบางขอบภายในขอบเขตท้องถิ่นของโครงข่ายจะถูกลบทิ้ง และสร้างขึ้นใหม่ภายในขอบเขตท้องถิ่นที่กำหนดขึ้นใหม่ ขั้นตอนการสร้างมีดังนี้

1. โครงข่ายสามเหลี่ยมดีลอนเนย์ของคอนเว็กซ์ฮัลถูกใช้เป็นโครงข่ายเริ่มต้น ดังรูปที่ 2.14(ก)
2. ก่อนจุดใหม่จะถูกเพิ่มเข้าสู่โครงข่าย รูปหลายเหลี่ยมซึ่งแสดงถึงขอบเขตท้องถิ่นจะถูกกำหนดขึ้นเมื่อรัศมีและจุดศูนย์กลางของวงกลมทดสอบ (circumscribing circle) ของแต่ละรูปสามเหลี่ยมถูกนำมาทดสอบกับจุดใหม่ (หลักการนี้จะถูกอธิบายอย่าง

ละเอียดในหัวข้อที่ 3.3.3.3) สามเหลี่ยมใดที่มีจุดใหม่บรรจุอยู่ภายในวงกลมทดสอบของตัวเองจะถูกลบทิ้งออกจากโครงข่าย ดังนั้นโครงข่ายจึงเกิดช่องว่างล้อมรอบจุดใหม่ในลักษณะของรูปหลายเหลี่ยมหรือที่เรียกว่า ขอบเขตท้องถิ่น ซึ่งคือพื้นที่ที่ได้รับผลกระทบจากการทดสอบก่อนการเพิ่มจุดใหม่ ดังรูปที่ 2.14(ข)

3. โครงข่ายสามเหลี่ยมภายในขอบเขตท้องถิ่นจะถูกสร้างขึ้น โดยเชื่อมจุดยอดทุกจุดของขอบเขตท้องถิ่นกับจุดใหม่ ดังรูปที่ 2.14(ค)
4. ขบวนการของการทดสอบจุด การลบขอบที่ได้รับผลกระทบ และการสร้างขอบใหม่จะดำเนินต่อไปตามขั้นที่ 2 และ 3 จนกระทั่งไม่เหลือจุดใหม่ที่จะถูกเพิ่มอีก สุดท้ายก็จะได้โครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์ ดังรูปที่ 2.14(ง)



รูปที่ 2.14 อัลกอริทึม Incremental Delete-and-Build (ก) โครงข่ายเริ่มต้น (ข) ขอบเขตท้องถิ่นสำหรับการแทรกจุด และจุดภายในขอบเขตถูกลบทิ้ง (ค) ขอบใหม่ถูกสร้างขึ้นเพื่อเชื่อมจุดใหม่กับขอบเขตท้องถิ่น (ง) โครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์

2.5 วิจารณ์ประสิทธิภาพ

2.5.1 เวลาในการปฏิบัติงาน

ชนิดของโครงสร้างข้อมูลและวิธีการเข้าถึงจุดข้อมูลเป็นตัวแปรที่สำคัญ ซึ่งจะมีผลโดยตรงกับการใช้เวลาในการทำงานของแต่ละอัลกอริทึม โครงสร้างข้อมูลที่เหมาะสมกับอัลกอริทึมหนึ่งอาจจะไม่เหมาะสมที่จะใช้กับอีกอัลกอริทึมหนึ่งก็เป็นได้ นอกจากนั้นวิธีการสร้างโครงข่ายสามเหลี่ยมแต่ละวิธีต่างก็มีความซับซ้อนและความยุ่งยากในการจัดการที่แตกต่างกันและยากที่จะประเมินประสิทธิภาพ แต่อย่างไรก็ตามในหัวข้อนี้จะขอนำเสนอบทวิจารณ์ประสิทธิภาพ และการปฏิบัติงานของบางอัลกอริทึม โดยให้ n แทนจำนวนโหนด (nodes) หรือจุดข้อมูลในโครงข่าย

ใน [20] ไม่ได้กล่าวถึงเวลาในการปฏิบัติงานของอัลกอริทึม Radial Sweep แต่ก็สามารถสังเกตการปฏิบัติงานอย่างหยาบๆได้ว่า การเรียงลำดับข้อมูลตามค่าองศาของมุมในขั้นตอนเริ่มต้นนั้นจะใช้เวลาปฏิบัติงานประมาณ $n \log n$ [25] สำหรับในขั้นตอนของการสลับขอบนั้นเวลาที่ใช้ในการทำงานที่เกิดกรณีเลวร้ายที่สุด (worst case) ควรจะเป็น $O(n^2)$ อย่างไรก็ตามกรณีเลวร้ายที่สุดนี้ก็เกิดขึ้นได้ยาก

ใน [18] ได้อ้างไว้ว่าเวลาในการปฏิบัติงานของอัลกอริทึม Recursive Split เป็น $O(n \log n)$ แต่จากงานทดลองของ [17] กลับอ้างว่าเป็น $O(n^2)$ ซึ่งก็น่าจะเป็นจริงเมื่อตำแหน่งของจุดมีการกระจายตัวสูง อย่างไรก็ตามจากการทดลองพบว่าประสิทธิภาพในเรื่องของเวลาของวิธีการนี้เข้าใกล้ $O(n \log n)$

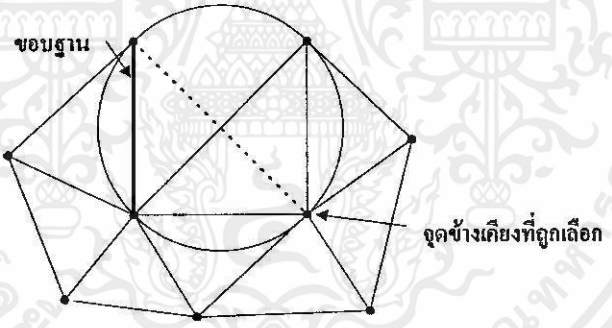
สำหรับอัลกอริทึม Divide-and-Conquer มีประสิทธิภาพเป็น $O(n \log n)$ อย่างแท้จริง เนื่องจากในทุกๆระดับของการประมวลผลของอัลกอริทึมนี้ สามเหลี่ยมที่ได้จะเป็นดีลอนเนย์เสมอ และไม่มีการทำงานที่ซ้ำซ้อนและสิ้นเปลือง

การค้นหาจุดที่สามหรือจุดข้างเคียง (Neighboring point) เพื่อมาประกอบเป็นรูปสามเหลี่ยมคืออุปสรรคที่สำคัญที่ใช้เวลาค่อนข้างมากสำหรับอัลกอริทึม Step-by-Step อย่างไรก็ตามการค้นหาจุดสามารถเพิ่มความเร็วได้ถึง $O(n \log n)$ การสร้างรูปสามเหลี่ยมดีลอนเนย์แต่ละรูปนั้นใช้เวลาที่คงที่คือประมาณ n จำนวนครั้ง

อัลกอริทึมต่างๆที่กล่าวมาล้วนแต่เป็นวิธีการแบบสถิต สำหรับวิธีการแบบพลวัติ เช่น อัลกอริทึม Modified Hierarchical, อัลกอริทึม Incremental algorithm, และอัลกอริทึม Incremental Delete-and-Build ล้วนแต่ปฏิบัติงานในกรณีเลวร้ายที่สุดเป็น $O(n^2)$ อันเนื่องมาจากความซับซ้อนในการปรับโครงสร้างของโครงข่าย ซึ่งสัมพันธ์โดยตรงกับการกระจายตัวของเซ็ตข้อมูลที่ไม่แน่นอน แต่ในทางปฏิบัติ [11] ทดลองพบว่าประสิทธิภาพเวลาในการปฏิบัติงานกลับเป็น $O(n \log n)$ สำหรับอัลกอริทึม Incremental

2.5.2 เสถียรภาพ

ในการทดลองของวิทยานิพนธ์นี้ได้ทำการเขียนโปรแกรมสำหรับ 2 อัลกอริทึมหลักคือ Step-by-Step และ Incremental เพื่อเปรียบเทียบความมีเสถียรภาพ นอกจากนั้นการทดลองยังได้กระทำกับรูทีน (routine) สำหรับขั้นตอนของการเชื่อมต่อสองโครงข่ายเข้าด้วยกัน จากการทดลองที่ผ่านมาทำให้ทราบถึงความไม่มีเสถียรภาพบางอย่างซึ่งยากที่จะจัดการ อัลกอริทึม Incremental ซึ่งถูกเน้นหนักในวิทยานิพนธ์นี้เป็นพิเศษ พบว่ามีเสถียรภาพมากเมื่อกฎเกณฑ์วงกลมได้รับการทดสอบครบทุกสามเหลี่ยมในโครงข่าย แม้แต่สามเหลี่ยมที่มีพื้นที่เป็นศูนย์ (Zero triangle) (อธิบายในบทที่ 4) ซึ่งมีรัศมีของวงกลมทดสอบเป็นอนันต์ (radius = ∞) ก็สามารถจัดการได้ และถูกต้องตรงตามกฎเกณฑ์วงกลมทุกประการ โดยสามเหลี่ยมพื้นที่ศูนย์จะมีตำแหน่งอยู่ที่อาณาเขตของพื้นที่ (boundary) ความไม่มีเสถียรภาพของอัลกอริทึม Step-by-Step ที่ปรากฏคือโครงข่ายผลลัพธ์มีความไม่เป็นหนึ่งเดียว (non-unique triangular meshes) ในรูปที่ 2.15 แสดงถึงจุด (ซึ่งไม่สอดคล้องตามกฎเกณฑ์ของวงกลม) อาจจะเป็นจุดที่ได้รับเลือก เพราะตามหลักการของอัลกอริทึมนี้จะเลือกจุดที่อยู่ใกล้ที่สุดเป็นจุดข้างเคียง เพื่อมาประกอบเป็นรูปสามเหลี่ยมใหม่ ด้วยเหตุผลนี้จึงทำให้ได้โครงข่ายสามเหลี่ยมดีลอนเนย์ที่ไม่สมบูรณ์เท่าที่ควร



รูปที่ 2.15 ความไม่สมบูรณ์ของอัลกอริทึม Step-by-Step เมื่อจุดข้างเคียงที่อยู่ใกล้กับเส้นฐานมากที่สุดที่ได้รับเลือกอาจจะไม่ตรงตามกฎเกณฑ์ของวงกลมทดสอบ

2.5.3 ข้อได้เปรียบเสียเปรียบ

ในทางปฏิบัติแล้วแบบจำลองพื้นผิวแบบดิจิตอล (The digital terrain models) [33] ซึ่งประกอบขึ้นจากข้อมูลจำนวนมาก ต้องการวิธีการที่มีประสิทธิภาพในการจัดการกับข้อมูลขนาดใหญ่ อัลกอริทึม Divide-and-Conquer ให้ผลลัพธ์ในการทำงานเป็นที่น่าพอใจ โดยใช้กระบวนการแยกโครงข่ายออกจากกัน ซึ่งเหมาะสำหรับการปฏิบัติงานบนสถาปัตยกรรมแบบขนาน (The parallel architecture) และการประมวลผลแบบขนานยังสามารถนำมาประยุกต์ใช้ได้เป็นอย่างดีกับขั้นตอนเริ่มต้นสำหรับอัลกอริทึม Recursive Split, Modified Hierarchical และ Radial Sweep เพื่อ

เพิ่มความเร็วในการทำงาน แต่อย่างไรก็ตามขั้นตอนสุดท้ายของอัลกอริทึมเหล่านี้ค่อนข้างที่จะใช้เวลาในการประมวลผลมาก

มีความเป็นไปได้ที่อัลกอริทึม Modified Hierarchical สามารถใช้โครงสร้างข้อมูลแบบถ่ายทอดลำดับชั้น (Hierarchical) ในขั้นตอนเริ่มต้น เมื่อสามเหลี่ยมแต่ละรูปถูกแบ่งออกเป็นสามเหลี่ยมย่อย 3 รูป โครงสร้างข้อมูลแบบรากไม้สำหรับจัดเก็บรูปสามเหลี่ยม (The triangle tree) ซึ่งมีจำนวนสาขาของแต่ละรูปสามเหลี่ยมเท่ากับ 3 จะจัดเก็บรูปสามเหลี่ยมย่อยเหล่านี้เป็นลำดับชั้น อย่างไรก็ตามโครงสร้างข้อมูลแบบรากไม้ 3 สาขานี้ค่อนข้างที่จะจัดการลำบากระหว่างการดำเนินการสลับขอบ

สำหรับการประยุกต์ใช้โครงข่ายสามเหลี่ยมในการสร้างแบบจำลองพื้นผิว โดยการคัดเลือกจุดสำคัญจากแบบจำลองระหว่างการคำนวณโครงข่ายสามเหลี่ยม อัลกอริทึม Incremental ถือว่าทำได้ดีที่สุด อัลกอริทึม Modified Hierarchy แม้ว่าจะสามารถนำมาประยุกต์ใช้ได้ในทางทฤษฎี แต่การคัดเลือกจุดกระทำอยู่บนพื้นฐานของระนาบสามเหลี่ยมซึ่งมีรูปร่างที่ไม่เหมาะสม (ยาวและแหลม) สำหรับวิธีการอื่นๆการกลั่นกรองจุดก่อนและหลังการคำนวณโครงข่ายสามเหลี่ยมจำเป็นต้องได้รับการจัดการที่ดี แม้ว่าจะให้ผลลัพธ์ที่ถูกต้องมากขึ้นแต่ในขั้นตอนนี้ก็ค่อนข้างสิ้นเปลืองเวลาในการคำนวณมากพอสมควร วิธีการต่างๆสำหรับการกลั่นกรองจุดข้อมูลได้อธิบายไว้ใน [16]

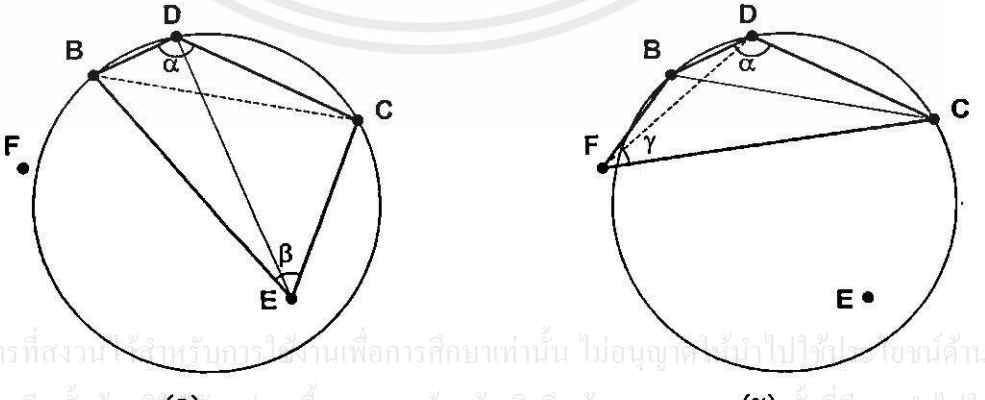
การใช้อัลกอริทึม Incremental ในการสร้างโครงข่ายสามเหลี่ยมร่วมกับวิธีการคัดเลือกจุดเป็นการลดความซ้ำซ้อนของข้อมูล (redundant points) จากแหล่งข้อมูลเริ่มต้นขนาดใหญ่ได้เป็นอย่างดี ซึ่งการคัดเลือกจุดสำคัญบนพื้นผิวสำหรับสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบถือเป็นหัวใจหลักในการสร้างแบบจำลองภูมิประเทศในวิทยานิพนธ์ฉบับนี้ ในบทที่ 3 จะกล่าวถึงทฤษฎีพื้นฐานทางเรขาคณิตของอัลกอริทึม Incremental ว่าเมื่อมีการแทรกจุดใหม่เข้าสู่โครงข่ายสามเหลี่ยมจะเกิดการเปลี่ยนแปลงต่อโครงสร้างของโครงข่ายอย่างไรบ้าง

บทที่ 3
ทฤษฎีพื้นฐานของอัลกอริทึม Incremental

การเพิ่มจุดเข้าสู่โครงข่ายสามเหลี่ยม ถือเป็นหัวใจหลักของการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์โดยใช้อัลกอริทึม Incremental เนื่องจากการแทรกจุดใหม่แต่ละครั้งจะมีผลกระทบโดยตรงกับโครงสร้างบางส่วนของโครงข่าย ในบทนี้จะนำเสนอทฤษฎีพื้นฐานทางเรขาคณิต 7 ข้อ สำหรับอัลกอริทึม Incremental เมื่อมีจุดใหม่ถูกแทรกเข้าสู่โครงข่าย โครงสร้างของโครงข่ายจะได้รับการจัดเรียงใหม่บางส่วน โดยการจัดเรียงนี้ต้องสอดคล้องกับทฤษฎีทั้ง 7 ข้อ ซึ่งจะเป็นเงื่อนไขที่จะกำหนดให้โครงข่ายผลลัพธ์มีคุณสมบัติของความเป็นสามเหลี่ยมดีลอนเนย์อย่างสมบูรณ์

3.1 สัญลักษณ์และข้อกำหนด

สัญลักษณ์ทางเรขาคณิตได้ถูกนำมาใช้ร่วมในการอธิบายเพื่อความกระชับของเนื้อหา โดยมีข้อกำหนดดังต่อไปนี้ สามเหลี่ยมของจุด 3 จุด (ABC) แทนด้วย ΔABC วงกลมถูกกำหนดขึ้นจากจุดยอดทั้ง 3 จุดของรูปสามเหลี่ยม แทนด้วย O_{ABC} (Circumscribing circle of a triangle) รูปสี่เหลี่ยมด้านไม่เท่า (Quadrilateral) ถูกกำหนดขึ้นจากจุดยอดทั้ง 4 ของรูปสี่เหลี่ยม แทนด้วย $\square ABCD$ ส่วนโค้งของวงกลม (Arc) ระหว่างจุด A และ B แทนด้วย $\frown AB$ จุดต่างๆบนส่วนโค้งของรูปสามเหลี่ยม วงกลม และรูปสี่เหลี่ยมถูกจัดเรียงตามเข็มนาฬิกา (Clockwise) จุดใหม่ที่ถูกแทรกเข้าสู่ ΔABC แทนด้วย D สามเหลี่ยมดีลอนเนย์สำหรับเซ็ต S_n ของจุดจำนวน n จุดใน 2 มิติแทนด้วย τ_n เมื่อ $n \in I^2$ สามเหลี่ยมหนึ่งรูปหรือกลุ่มของรูปสามเหลี่ยมใน τ_n ถูกเรียกว่าเป็นส่วนหนึ่ง (Segment) ของ τ_n



รูปที่ 3.1 การทดสอบผลรวมของมุมภายใน

3.2 การทดสอบผลรวมของมุมภายใน (Maximum angle sum test)

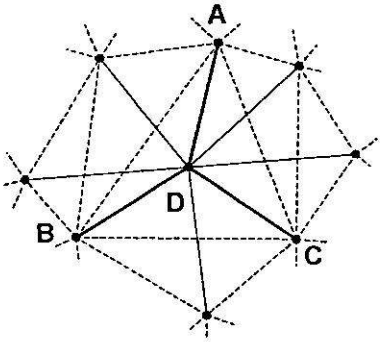
[15] ได้แสดงให้เห็นว่าการนำเอาหลักการของการทดสอบผลรวมของมุมภายในมาใช้ทำให้โครงข่ายสามเหลี่ยมสามารถบรรลุถึงคุณสมบัติดีลอนเนย์ที่สมบูรณ์ได้ (ดังรูปที่ 3.1) โดยการทดสอบเพื่อเลือกเส้นทะแยงมุมที่เหมาะสมภายในรูปสี่เหลี่ยมด้านไม่เท่าในโครงข่ายสามเหลี่ยมดีลอนเนย์ ผลลัพธ์ที่ได้จากการทดสอบมุมภายในนี้สอดคล้องกับหลักการของการทดสอบจุดภายในวงกลมหรือ InCircle ของ [10]

<u>ทฤษฎีที่ 1</u> รูปสี่เหลี่ยมด้านไม่เท่าใดๆในโครงข่ายสามเหลี่ยมดีลอนเนย์ซึ่งประกอบขึ้นจากรูปสามเหลี่ยม 2 รูปเรียงติดกัน ต้องมีผลรวมของมุมภายในที่ปลายของเส้นทะแยงมุมทั้ง 2 มีค่ามากกว่าหรือเท่ากับ π

<u>พิสูจน์</u> จากรูปที่ 3.1(ก) $\square BDCE$ คือส่วนหนึ่งของ τ_n และมีจุด E บรรจุอยู่ภายใน O BDC ดังนั้น τ_n จึงไม่สามารถกำหนดให้มีได้ถ้า $\overline{BC}$ (เส้นประ) ถูกเลือกให้เป็นเส้นทะแยงมุมสำหรับ $\square BDCE$ (แทนที่จะเป็น $\overline{DE}$) $\overline{BC}$ จะเป็นเส้นทะแยงมุมใน τ_n ได้ก็ต่อเมื่อจุด E มีตำแหน่งอยู่ภายนอก O_{BDC} โดยกฎพื้นฐานของเรขาคณิตกล่าวไว้ว่า $\alpha = (\frown CB)/2$ และ $\frown CB + \frown BC = 2\pi$ ดังนั้น E จะอยู่บนเส้นรอบวงของ O_{BDC} เมื่อ $\alpha + \beta = \pi$, เมื่อ E เลื่อนตำแหน่งเข้าสู่ภายในวงกลม $\alpha + \beta > \pi$, และเมื่อ E เลื่อนตำแหน่งออกจากวงกลม $\alpha + \beta < \pi$ ผลรวมของมุมภายในทั้ง 4 จะมีค่าเท่ากับ 2π เสมอ เพื่อให้ถูกต้องตามกฎเกณฑ์วงกลมผลรวมของมุมภายในระหว่างมุมที่ปลายของเส้นทะแยงมุมทั้งสองข้างต้องมากกว่าหรือเท่ากับ π ในรูปที่ 3.1 (ก) $\overline{DE}$ คือเส้นทะแยงมุมที่ถูกต้องของ $\square BDCE$ และจากรูปที่ 3.1 (ข) $\overline{BC}$ เป็นเส้นทะแยงมุมที่ถูกต้องของ $\square BDCF$ เพราะ $\alpha + \gamma < \pi$ ที่สำคัญ $\overline{FD}$ มีขนาดสั้นกว่า $\overline{BC}$ ดังนั้นจะเห็นได้ว่าการเลือกเส้นทะแยงมุมที่สั้นที่สุด (The minimum diagonal test) ในหัวข้อที่ 2.4.1 จึงไม่สอดคล้องกับกฎเกณฑ์วงกลมและทำให้โครงข่ายสามเหลี่ยมผลลัพธ์ที่ได้มีความเป็นดีลอนเนย์ที่ไม่สมบูรณ์

3.3 ผลกระทบของการแทรกจุดที่มีต่อขอบรอบข้าง

ในหัวข้อนี้จะอธิบายให้ทราบถึงการแทรกจุด (n+1) เข้าสู่ τ_n จะมีผลกระทบอย่างไรต่อขอบต่างๆในโครงข่ายสามเหลี่ยมเมื่อผลลัพธ์สุดท้ายเป็น τ_{n+1} เมื่อ D ถูกแทรกเข้าสู่ ΔABC ดังรูปที่ 3.2 ขอบใหม่ 3 ขอบจะถูกสร้างขึ้นและบรรจุใน τ_{n+1} ขอบรอบข้าง D บางขอบจำเป็นต้องมีการสลับ (Edge swapped) โดยมีปลายด้านหนึ่งของขอบมีตำแหน่งชี้ไปที่ D เสมอ ขอบที่ได้รับการสลับเป็นสมาชิกของ τ_{n+1} และการสลับจะถูกกระทำเพียงครั้งเดียวเท่านั้น



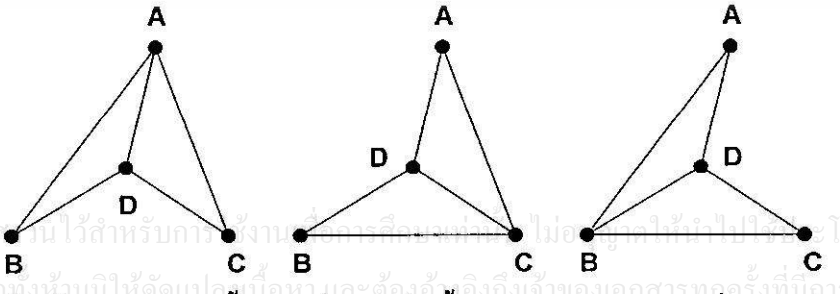
รูปที่ 3.2 ผลลัพธ์ที่เกิดขึ้นหลังจากที่ D ถูกแทรกเข้าสู่ ΔABC เมื่อขอบใหม่ได้ถูกสร้างขึ้นและมีบางขอบถูกสลับ (เส้นประแสดงขอบเดิมก่อนที่จะมีการแทรกจุดใหม่)

3.3.1 ผลกระทบที่เกิดขึ้นกับขอบใหม่ 3 ขอบ

เมื่อ D ถูกแทรกเข้าสู่ ΔABC ขอบใหม่ 3 ขอบได้ถูกสร้างขึ้นเพื่อเชื่อมต่อ D กับ τ_n ดังในรูปที่ 3.2 รูปสามเหลี่ยมเดิม (ΔABC) จะถูกแบ่งออกเป็น 3 ส่วน คือ ΔADB, ΔACD, และ ΔBDC

<u>ทฤษฎีที่ 2</u> ขอบใหม่ทั้ง 3 ขอบซึ่งถูกสร้างขึ้นจากการเชื่อมจุดใหม่กับมุมทั้ง 3 ของรูปสามเหลี่ยมเดิมจะไม่ได้รับผลกระทบใดๆ จากการปรับโครงสร้างของขอบต่างๆ (The edge reorganization) ซึ่งจะกระทำหลังจากการแทรกจุดใหม่

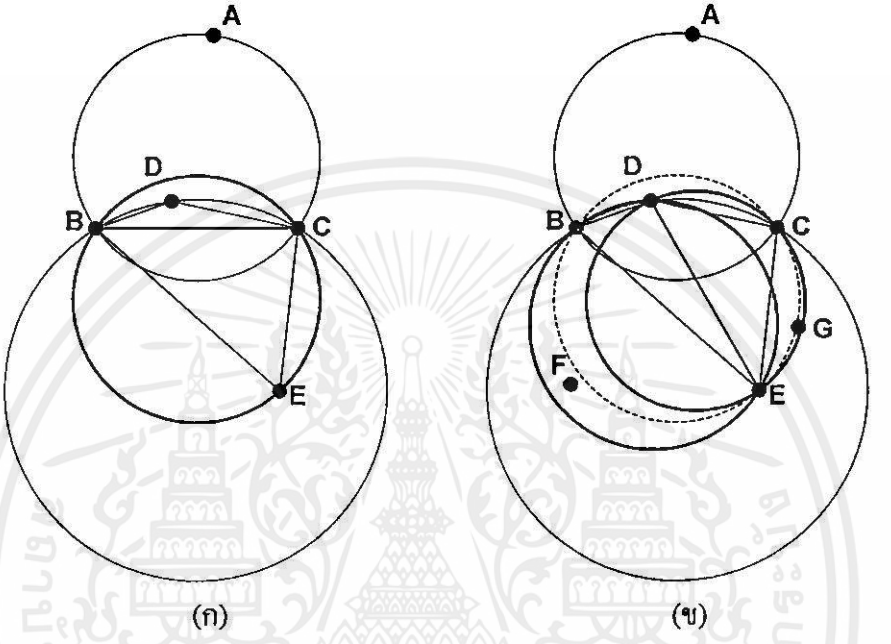
<u>พิสูจน์</u> จากรูปที่ 3.2 สังเกตเห็นได้ว่ารูปสี่เหลี่ยมด้านไม่เท่าทั้ง 3 รูปภายใน ΔBAC ต่างก็ไม่เป็นรูปสี่เหลี่ยมด้านไม่เท่าแบบคอนเว็กซ์ (Non-convex) ดังรูปที่ 3.3 ทำให้ผลรวมของมุมภายในระหว่างเส้นทะแยงมุมมีค่ามากกว่า π ซึ่งสอดคล้องกับทฤษฎีที่ 1 ดังนั้นขอบใหม่ทั้ง 3 ขอบจึงไม่มีการถูกสลับอย่างแน่นอน ก่อนที่จะเข้าสู่กระบวนการของการปรับโครงสร้างเพื่อให้บรรลุถึงโครงข่ายสามเหลี่ยม τ_{n+1}



รูปที่ 3.3 รูปสี่เหลี่ยมด้านไม่เท่าทั้ง 3 รูปซึ่งถูกสร้างขึ้นใหม่ภายในรูปสามเหลี่ยมเดิมมีลักษณะไม่เป็นแบบคอนเว็กซ์

3.3.2 ผลกระทบที่เกิดขึ้นกับขอบเดิม

ในหัวข้อนี้จะศึกษาถึงผลกระทบที่จะเกิดขึ้นกับขอบเดิมที่มีอยู่แล้วในโครงข่าย เมื่อ $\overline{DE}$ คือขอบซึ่งถูกสมมุติให้ได้รับการสลับ และผลจากการเปลี่ยนแปลงนี้อาจจะมีจุดอื่นๆซึ่งมีตำแหน่งอยู่ใกล้เคียงมีผลกระทบต่อการสลับตำแหน่งของขอบนี้อีกก็เป็นได้



รูปที่ 3.4 (ก) รูปสามเหลี่ยมก่อนที่จะถูกสลับขอบ (O_{BCE} ถูกเน้นด้วยเส้นหนาให้เห็นเด่นชัด) (ข) รูปสามเหลี่ยมหลังการสลับขอบ (O_{BCD} ถูกแสดงเป็นเส้นประในขณะที่ O_{BDE} และ O_{DCE} ถูกวาดด้วยเส้นหนา)

<u>ทฤษฎีที่ 3</u> ขอบซึ่งได้รับผลกระทบจากการแทรกจุดใหม่ และถูกสลับเส้นทะแยงมุมเรียบร้อยแล้วจะมีปลายข้างหนึ่งชี้ไปที่จุดใหม่เสมอ

<u>พิสูจน์</u> หลักการพื้นฐานที่จะใช้ในการพิสูจน์ได้แสดงในรูปที่ 3.4(ก) เมื่อเกิดสถานะการณ์ที่ D ถูกแทรกเข้าสู่ τ_n ΔBCE คือสามเหลี่ยมเดิมซึ่งปรากฏอยู่แล้วใน τ_n ก่อนที่ D จะถูกเพิ่มเข้ามา วงกลมทดสอบถูกวาดขึ้นจาก ΔBCE ตามกฎเกณฑ์ของดีลอนเนย์แล้ว ใน τ_n ต้องไม่มีจุดใดๆบรรจุอยู่ภายในวงกลมนี้ สอดคล้องกับการทดสอบผลรวมของมุมภายในที่ปลายเส้นทะแยงมุมของ $\square BDCE$ ต้องได้รับการสลับจาก $\overline{BC}$ ไปเป็น $\overline{DE}$ ดังแสดงในรูปที่ 3.4(ข) ซึ่ง O_{BCE} ถูกวาดด้วยเส้นประ ส่วน O_{BDE} และ O_{DCE} ต่างถูกวาดด้วยเส้นหนา ดังนั้น O_{BDE} และ O_{DCE} จึงใช้ $\overline{DE}$ เป็นขอบร่วม O_{BDE} มีพื้นที่บางส่วนอยู่ภายนอก O_{DCE} ทางด้านซ้ายของ $\overline{DE}$ และ O_{DCE} ก็มีพื้นที่บางส่วนอยู่ภายนอก O_{BDE} ทางด้านขวาของ $\overline{DE}$

จุดใดๆซึ่งจะส่งผลกระทบต่อโครงข่ายจะมีตำแหน่งอยู่ภายนอก O_{BCE} และบรรจุอยู่ภายใน O_{BDE} หรือ O_{DCE} จากรูปที่ 3.4(ข) จุด F และ G คือตัวอย่างของจุดที่จะส่งกระทบในการปรับปรุงโครงสร้างในอนาคต จุด F มีตำแหน่งอยู่ในพื้นที่ระหว่าง O_{BCE} และ O_{BDE} และเนื่องจาก O_{DCE} ไม่มีโอกาสที่จะครอบคลุม O_{BCE} ในพื้นที่ปัจจุบัน ดังนั้น $\overline{DE}$ จะไม่มีทางถูกสลับขอบเป็น $\overline{CF}$ อย่างเด็ดขาด เช่นเดียวกับที่ $\overline{DE}$ ก็ไม่มีทางถูกสลับให้เป็น $\overline{BG}$ ได้

ทฤษฎีที่ 4 ขอบใดๆซึ่งได้รับผลกระทบจากการแทรกจุดจะถูกสลับไม่เกิน 1 ครั้งเสมอหลังจากการแทรกจุดใหม่แต่ละครั้ง

พิสูจน์ จากทฤษฎีที่ 3 ทำให้ทราบว่าขอบทุกขอบซึ่งถูกสลับจะมีปลายด้านหนึ่งชี้อยู่ที่จุดแทรกใหม่เสมอ ถ้าหนึ่งในจำนวนขอบที่ถูกสลับเหล่านี้ถูกสลับมากกว่าหนึ่งครั้งจะทำให้จุดปลายนี้เคลื่อนที่ออกจากจุดแทรกใหม่ ดังนั้นจึงสอดคล้องกับทฤษฎีที่ 3 ที่ขอบจะไม่ถูกสลับมากกว่า 1 ครั้งอย่างแน่นอน

จากการสังเกตยังพบอีกว่าในสี่เหลี่ยมด้านไม่เท่าแต่ละรูปนั้น จะมีเพียงขอบที่อยู่ตรงกันข้ามกับจุดที่แทรกใหม่เท่านั้นที่จะได้รับการพิจารณา และขอบเหล่านี้ต้องได้รับการทดสอบผลรวมของมุมภายในเพื่อพิจารณาการสลับเส้นทะแยงมุมของรูปสี่เหลี่ยมด้านไม่เท่านี้ต่อไป

3.3.3 ส่วนขยายของพื้นที่ใน T_n ซึ่งได้รับผลกระทบจากการแทรกจุด

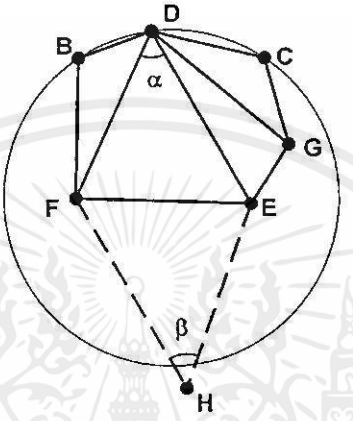
งานวิจัยหลายงานได้อ้างไว้ว่าการแทรกจุดเข้าสู่ T_n นี้จะส่งผลกระทบเป็นพื้นที่ในลักษณะท้องถิ่น (Localization) โดยขนาดของพื้นที่ซึ่งได้รับผลกระทบนี้จะขึ้นอยู่กับการกระจายตัวของจุดใน T_n ในหัวข้อนี้จะนำเสนอการวิเคราะห์ส่วนขยายของพื้นที่ท้องถิ่นที่ได้รับผลกระทบรวมไปถึงการศึกษาการกระจายตัวของจุดว่ามีผลอย่างไรต่อส่วนขยายนี้

3.3.3.1 ขอบเขตจำกัด (Upper limitation)

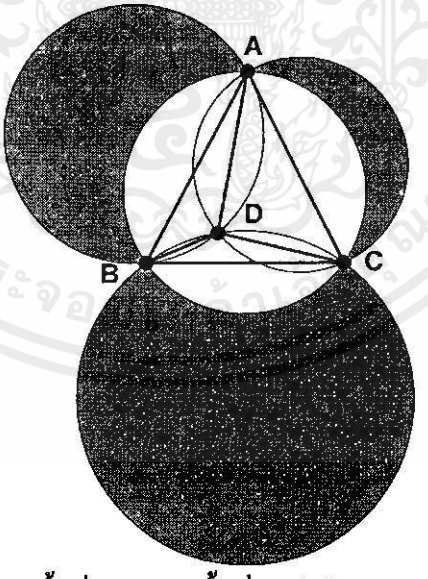
ทฤษฎีที่ 5 พื้นที่ซึ่งได้รับผลกระทบหลังการแทรกจุดใหม่จะไม่มากกว่าพื้นที่รวมของวงกลมทดสอบ 3 รูปของรูปสามเหลี่ยมใหม่ซึ่งถูกแบ่งจากสามเหลี่ยมเดิม

พิสูจน์ ในรูปที่ 3.1 ได้แสดงให้เห็นถึงขอบต่างๆซึ่งเรียงติดกับจุดแทรกใหม่ (D) จะไม่ได้รับผลกระทบใดๆจากจุดซึ่งมีตำแหน่งอยู่ภายนอก O_{BDC} ในรูปที่ 3.5 ก็เช่นกัน $\overline{DE}$, $\overline{DF}$ และ $\overline{DG}$ ต่างก็ถูกสลับขอบตามผลกระทบของการแทรกจุด D และ $\overline{FE}$ คือขอบต่อไปที่จะได้รับการ

พิจารณา แม้ว่า $\overline{FE}$ จะอยู่ภายในพื้นที่ของวงกลมทดสอบ แต่ H มีตำแหน่งอยู่ภายนอก O_{BDC} และ $\alpha + \beta$ ก็มีค่าน้อยกว่า π ซึ่งจุด H จะเป็นเพื่อนบ้านดีลอนเนย์ (Delaunay neighbor) ของ D ได้ก็ต่อเมื่อ $\alpha + \beta > \pi$ ดังนั้น $\overline{FE}$ จึงไม่ได้รับผลกระทบและไม่ต้องถูกสลับขอบให้เป็น $\overline{DH}$ ในรูปที่ 3.6 แสดงให้เห็นถึงพื้นที่เป็นไปได้สำหรับจุดที่จะมาเป็นเพื่อนบ้านดีลอนเนย์ของ D ไม่มีขอบใดภายนอกพื้นที่แลเงาที่จะได้รับผลกระทบจากการแทรกจุดใหม่ของ D ใน Δ_{BAC}



รูปที่ 3.5 จุด H ซึ่งอยู่ภายนอก O_{BDC} ไม่เป็นเพื่อนบ้านดีลอนเนย์ของ D

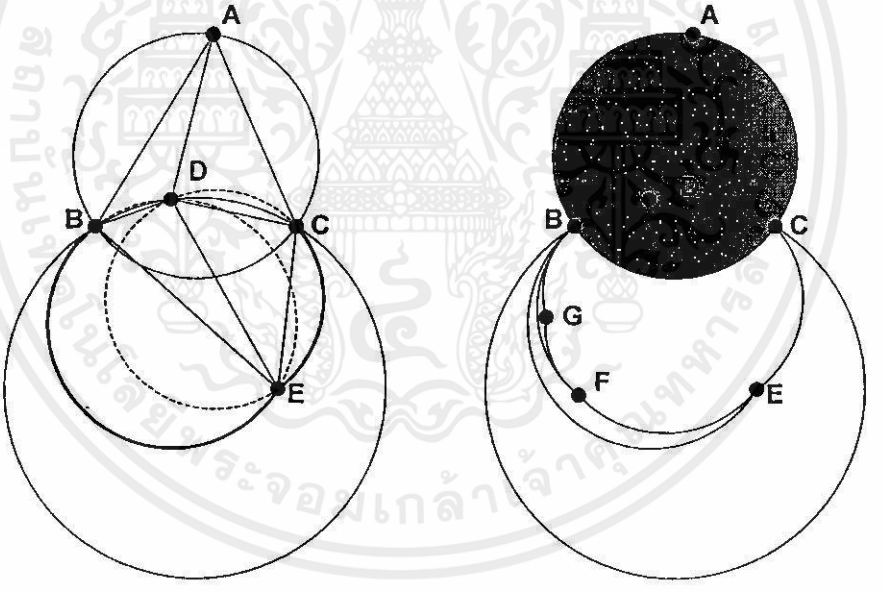


รูปที่ 3.6 มีเพียงขอบซึ่งอยู่ภายในพื้นที่แลเงาเท่านั้นที่อาจได้รับผลกระทบจากการแทรก D เข้าสู่ Δ_{BAC}

3.3.3.2 การลดลงของพื้นที่ซึ่งได้รับผลกระทบ

<u>ทฤษฎีที่ 6</u> เนื้อที่เพื่อนบ้านดีลอนเนย์ที่เป็นไปได้ของจุดแทรกใหม่จะมีตำแหน่งอยู่ภายในวงกลมทดสอบของรูปสามเหลี่ยมปัจจุบันซึ่งมีจุดแทรกใหม่เป็นหนึ่งในมุมยอด

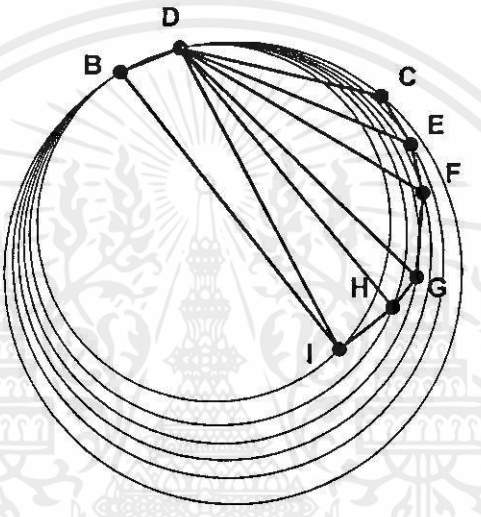
<u>พิสูจน์</u> ในรูปที่ 3.7(ก) E ได้ถูกเชื่อมต่อกับจุดใหม่ D หลังจากการสลับขอบ ดังในหัวข้อที่ 3.2 ซึ่งแสดงให้เห็นแล้วว่า F ต้องอยู่ภายใน O_{BCE} จึงจะมีคุณสมบัติเป็นจุดเพื่อนบ้านดีลอนเนย์ของ D ได้ และในที่นี้ O_{BDE} ถูกบรรจุอยู่ภายใน O_{BDC} ของพื้นที่ด้านล่างของขอบ $\overline{BD}$ เช่นเดียวกับที่ O_{DCE} ก็ได้มีตำแหน่งอยู่ภายใน O_{BDC} ภายในพื้นที่ปัจจุบันของการสลับขอบ ดังนั้นจึงหมายความได้ว่า $\cap CE$ และ $\cap EB$ จะกลายเป็นขอบเขตจำกัดปัจจุบันสำหรับพื้นที่ซึ่งจะได้รับผลกระทบของการแทรก D ในรูปที่ 3.7(ข) จะแสดงให้เห็นถึงการลดลงของพื้นที่ซึ่งได้รับผลกระทบว่าเป็นอย่างไร เมื่อ D ถูกเชื่อมต่อเข้ากับจุดเพื่อนบ้านต่างๆ



รูปที่ 3.7 (ก) O_{BDE} และ O_{DCE} กลายเป็นขอบเขตจำกัดใหม่ซึ่งอาจจะได้รับผลกระทบจากการทดสอบ (ข) พื้นที่การสลับขอบจะลดลงเมื่อแต่ละจุดได้รับการเชื่อมต่อเข้ากับ D

จากรูปที่ 3.7(ข) แสดงลำดับของขอบเขตจำกัดที่ลดลง เริ่มจากลำดับแรกคือ $\cap CB$ ลำดับที่สองคือ $\cap CE$ และ $\cap EB$ ลำดับสามคือ $\cap CE$, $\cap EF$, และ $\cap FB$ และลำดับที่สี่คือ $\cap CE$, $\cap EF$, $\cap FG$, และ $\cap GB$ จำนวนครั้งของการลดลงจะขึ้นอยู่กับการกระจายตัวของจุดภายในโครงข่าย ในรูปที่ 3.7(ข) ขอบเขตจำกัดมีการลดลงอย่างรวดเร็ว ในขณะที่การลดลงของขอบเขต

จำกัดในรูปที่ 3.8 เป็นไปอย่างค่อยเป็นค่อยไป เนื่องจากกลุ่มของจุดที่ได้รับผลกระทบอยู่ห่างจากจุดแทรกใหม่และมีการกระจายตัวแบบจัดเรียงใกล้ชิด ซึ่งรูปแบบของการกระจายตัวของกลุ่มจุดแบบนี้พบได้ง่ายในข้อมูลจุดที่ได้จากแผนที่เส้นแสดงระดับความสูง (Contour map) ผลลัพธ์ของโครงข่ายดีลอนเนย์ที่ได้จากข้อมูลนี้จะมีลักษณะเป็นรูปสามเหลี่ยมที่แหลมและบาง อันเนื่องมาจากความถี่ของการเรียงตัวของจุดที่ได้จากการคัดเลือก (sampling) จากเส้นคอนทัวร์แต่ละเส้น ในขณะที่จุดในพื้นที่ว่างระหว่างเส้นความสูงไม่ได้รับการพิจารณา ในรูปที่ 3.9 แสดงให้เห็นถึงโครงข่ายสามเหลี่ยมดีลอนเนย์ซึ่งคำนวณได้จากจุดสำคัญตามแนวเส้นคอนทัวร์



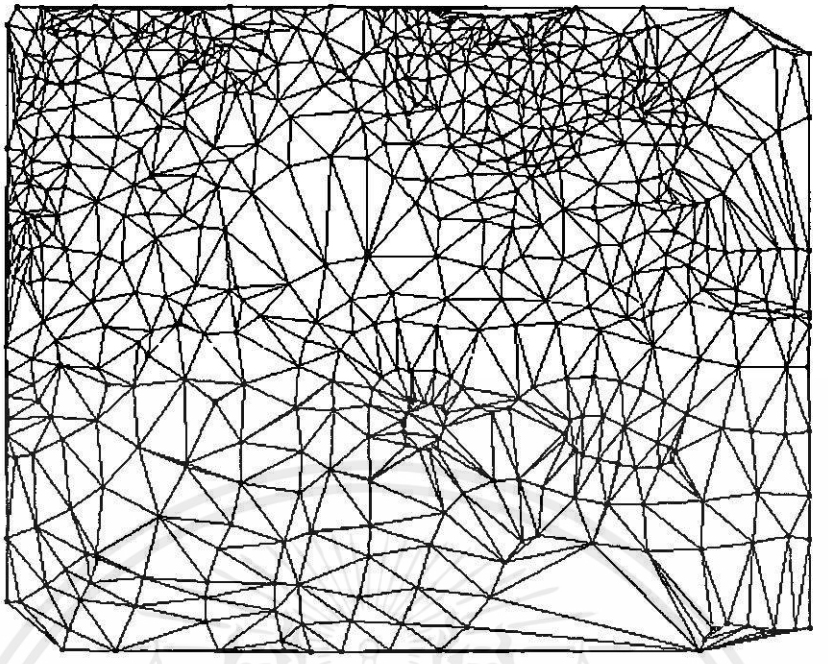
รูปที่ 3.8 การเปลี่ยนแปลงของขอบเขตจำกัดอย่างค่อยๆลดลง

3.3.3.3 ขอบเขตจำกัดที่แท้จริงของพื้นที่ได้รับผลกระทบ

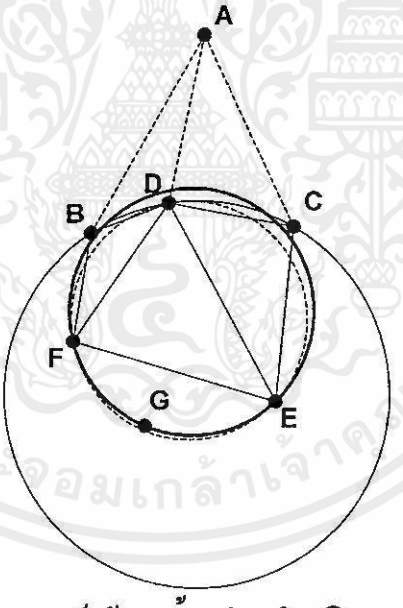
ในหัวข้อนี้จะแสดงให้เห็นถึงการคำนวณเพื่อหาพื้นที่ท้องถิ่นที่แท้จริงของขอบเขตจำกัดในรูปของรูปหลายเหลี่ยม เมื่อมีการแทรก D เข้าสู่โครงข่ายการคำนวณสามารถกระทำได้ก่อนที่ D จะถูกแทรก

<u>ทฤษฎีที่ 7</u> รูปสามเหลี่ยมทุกรูปใน T_n ซึ่งมีจุดแทรกใหม่อยู่ภายในวงกลมทดสอบของตนเองจะได้รับผลกระทบจากการแทรกจุดใหม่นี้

<u>พิสูจน์</u> จากรูปที่ 3.10 แสดงให้เห็นว่า G อยู่ภายใน O_{DEF} และ G ก็เป็นจุดเพื่อนบ้านดีลอนเนย์ของ D ดังนั้น D จึงอยู่ภายใน O_{FEG} ตราบเท่าที่ G เป็นจุดเพื่อนบ้านดีลอนเนย์ปัจจุบันของ D สรุปได้ว่าสามเหลี่ยมทุกรูปที่บรรจุ D ไว้ภายในวงกลมทดสอบของตัวเองจะได้รับผลกระทบจากการแทรก D



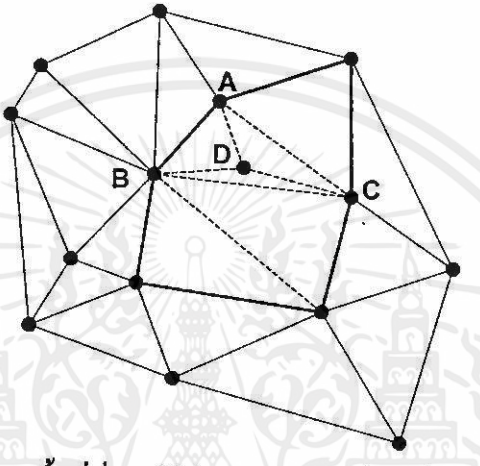
รูปที่ 3.9 โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบของจุดต่างๆบนเส้นคอนทัวร์



รูปที่ 3.10 G คือจุดเพื่อนบ้านดีลอนเนย์ ถ้า D นั้นอยู่ภายใน O_{FEG}

เมื่อสามเหลี่ยมที่ได้รับผลกระทบจาก D ได้ถูกค้นพบ สามเหลี่ยมเหล่านี้สามารถถูกลบออกไปได้ รูปหลายเหลี่ยมซึ่งปิดล้อม D ใน T_{n+1} แสดงในรูปที่ 3.11 คือผลลัพธ์ที่ได้จากการคำนวณหารูปหลายเหลี่ยมซึ่งเป็นขอบเขตจำกัดของพื้นที่ซึ่งได้รับผลกระทบ ขอบที่วาดด้วยจุดประแสดงขอบที่ต้องถูกสลับระหว่างกระบวนการของการปรับปรุงโครงสร้างของโครงข่าย และการคำนวณหาขอบเขตจำกัดในรูปของรูปหลายเหลี่ยมนี้ เป็นประโยชน์อย่างยิ่งในขั้นตอนเริ่มต้นของการลบจุดออกจากโครงข่าย

ทฤษฎีทั้ง 7 ข้อที่ได้กล่าวมาในบทนี้ถือเป็นความรู้พื้นฐานที่สำคัญยิ่งในการศึกษาหลักการของอัลกอริทึม Incremental สำหรับการสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบชนิดดีลอนเนย์ เงื่อนไขง่ายๆในการปรับปรุงโครงสร้าง คือ โครงข่ายต้องบรรลุตามกฎเกณฑ์ของวงกลมทดสอบเสมอก่อนที่จะมีการแทรกจุดใหม่ในครั้งต่อไป ในขั้นตอนของการนำเอาทฤษฎีพื้นฐานไปประยุกต์ใช้โครงสร้างข้อมูลต้องสามารถปรับตัวให้สอดคล้องตามเงื่อนไขที่ได้ตั้งไว้ทั้ง 7 ข้อ โดยมีอัลกอริทึมหรือโปรแกรมคอมพิวเตอร์เป็นตัวดำเนินการให้เกิดการเปลี่ยนแปลง



รูปที่ 3.11 รูปหลายเหลี่ยมแสดงพื้นที่ซึ่งจะได้รับผลกระทบอย่างแน่นอน

บทที่ 4
โครงสร้างข้อมูลและอัลกอริทึม

ในอดีตโครงสร้างแบบกริด (Grid) มักถูกใช้สำหรับการอธิบายความซับซ้อนของพื้นผิวทางภูมิศาสตร์ เป็นเพราะกริดมีพื้นฐานมาจากโครงสร้างข้อมูลที่ถูกจัดเรียงอย่างเป็นระเบียบ ซึ่งสามารถพูดได้ว่าโครงสร้างพื้นผิวแบบกริดเป็นโครงสร้างซึ่งถูกกำหนดขึ้นให้เหมาะสำหรับโครงสร้างข้อมูล (เช่นอาร์เรย์ 2 มิติ) ในทางกลับกันโครงสร้างข้อมูลชนิดโครงข่ายสามเหลี่ยม (TIN) ถูกสร้างขึ้นเพื่อให้เหมาะสมกับความซับซ้อนของพื้นผิว แบบจำลองกริดประกอบด้วยเซ็ตของจุดซึ่งมีระยะห่างและทิศทางการเชื่อมโยงที่เท่ากัน ในขณะที่โครงข่ายสามเหลี่ยมประกอบจากเซ็ตของจุดซึ่งกระจัดกระจาย มีระยะห่างและทิศทางการเชื่อมโยงที่ต่างกัน ดังนั้นโครงสร้างข้อมูลที่จะมาจัดการกับโครงข่ายสามเหลี่ยมจึงต้องมีคุณสมบัติและความสามารถที่พิเศษกว่าโครงสร้างข้อมูลโดยทั่วไปที่ใช้จัดการกับโครงสร้างแบบกริด

ในส่วนแรกของบทนี้จะนำเสนอโครงสร้างข้อมูลชนิดต่างๆ สำหรับจัดการกับโครงข่ายสามเหลี่ยม ซึ่งนำไปสู่การออกแบบโครงสร้างข้อมูลรูปแบบใหม่ (โครงสร้างข้อมูล TwinEdge) ที่รองรับกับการสร้างโครงข่ายสามเหลี่ยมแบบพลวัติ ในส่วนที่สองจะนำเสนอวิธีการและอัลกอริทึมสำหรับสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์อย่างเป็นขั้นตอน อาทิ การสร้างโครงข่ายเริ่มต้น การแทรกจุดใหม่เข้าสู่โครงข่าย การปรับโครงสร้างหลังการแทรกจุด การจัดลำดับการแทรกจุด การจัดการกับจุดระหว่างการสร้างโครงข่ายสามเหลี่ยม ในตอนท้ายจะอธิบายวิธีการสำรวจโครงข่ายสามเหลี่ยมเพื่อดึงเอาพิกัดของสามเหลี่ยมทุกรูปภายในโครงข่ายออกมาใช้งาน รวมไปถึงการแสดงผลการทำงานของอัลกอริทึม Incremental เมื่อเปรียบเทียบกับอัลกอริทึม Step-by-Step และการกำจัดสามเหลี่ยมศูนย์หรือสามเหลี่ยมรูปลิ่มออกจากบริเวณอาณาเขตของโครงข่ายสามเหลี่ยม

4.1 ข้อมูลในโครงข่ายสามเหลี่ยม

ส่วนประกอบพื้นฐานของโครงข่ายสามเหลี่ยมแบ่งออกเป็น 3 ระดับคือ จุด (Point หรือ Vertex) ขอบ (Edge) และสามเหลี่ยม (Triangle) จุดคือตำแหน่งพิจารณาใดๆซึ่งถูกกำหนดโดยค่าพิกัด x, y และ z ในระบบคาร์เทเชียน (Cartesian system) ขอบคือเส้นตรงซึ่งทำหน้าที่เชื่อมโยงจุด 2 จุด ระนาบสามเหลี่ยมเกิดขึ้นจากขอบ 3 ขอบซึ่งเชื่อมโยงจุด 3 จุดในรูปแบบวงรอบปิด โครงสร้างข้อมูลทางเรขาคณิตสำหรับจัดการกับส่วนประกอบทั้งสามประกอบด้วยข้อมูลสำคัญ 2 ส่วน คือ ข้อมูลเชิงพิกัด (Geometrical information) และ ข้อมูลเชิงโครงร่าง (Topological information) ข้อมูลเชิงพิกัดคือข้อมูลที่แสดงถึงตำแหน่งของจุดนั้น อาจอยู่ในรูปของค่าพิกัดโดยตรง หรือใช้การอ้างอิงถึงจุดเป้าหมายโดยการใช้ตัวชี้ข้อมูลหรือพอยเตอร์ (Pointer) ข้อมูลเชิงโครงร่างคือข้อมูลซึ่ง

อ้างอิงถึงส่วนประกอบข้างเคียง (Adjacent components) เพื่อประโยชน์ในการสร้างความสัมพันธ์ระหว่างส่วนประกอบ และลดความซ้ำซ้อนของข้อมูล รายละเอียดข้อมูลภายในโครงสร้างข้อมูลแต่ละระดับแสดงได้ดังนี้

(1) โครงสร้างข้อมูลชนิดจุด

ข้อมูลเชิงพิกัด

- ค่าพิกัด x, y และ z

ข้อมูลเชิงโครงร่าง

- พอยเตอร์ชี้ไปที่จุดรอบข้าง
- พอยเตอร์ชี้ไปที่ขอบข้างเคียง
- พอยเตอร์ชี้ไปที่สามเหลี่ยมข้างเคียง

(2) โครงสร้างข้อมูลชนิดขอบ

ข้อมูลเชิงพิกัด

- พิกัดของจุดปลายทั้ง 2 ข้าง หรือพอยเตอร์ชี้ไปที่จุดปลายทั้ง 2 ข้าง

ข้อมูลเชิงโครงร่าง

- พอยเตอร์ชี้ไปที่ขอบข้างเคียง
- พอยเตอร์ชี้ไปที่สามเหลี่ยมข้างเคียง

(3) โครงสร้างข้อมูลชนิดสามเหลี่ยม

ข้อมูลเชิงพิกัด

- พิกัดของมุมยอดทั้ง 3 หรือพอยเตอร์ชี้ไปที่มุมยอดทั้ง 3

ข้อมูลเชิงโครงร่าง

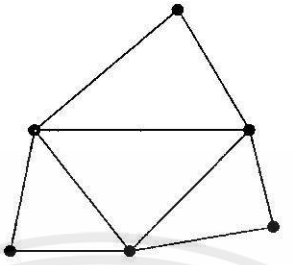
- พอยเตอร์ชี้ไปที่ขอบของรูปสามเหลี่ยม
- พอยเตอร์ชี้ไปที่สามเหลี่ยมข้างเคียง

ข้อมูลเพียงบางส่วนเท่านั้นที่จะถูกนำไปใช้เป็นส่วนประกอบของโครงสร้างข้อมูลสำหรับจัดการกับโครงข่ายสามเหลี่ยม ซึ่งขึ้นอยู่กับว่าจะนำเอาโครงสร้างข้อมูลรูปแบบนั้นไปใช้ร่วมกับอัลกอริทึมใด และอัลกอริทึมนั้นต้องการใช้ข้อมูลใดเป็นพิเศษ

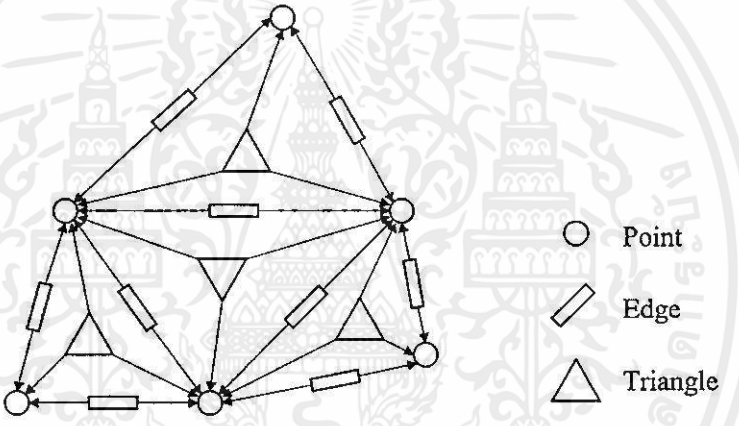
4.2 โครงสร้างข้อมูลเชิงพิกัด

โครงสร้างข้อมูลเชิงพิกัด คือโครงสร้างข้อมูลที่ให้ความสำคัญต่อข้อมูลเชิงพิกัด โครงข่ายสามเหลี่ยมดีลอนเนย์ในรูปที่ 4.1 ถูกแสดงเพื่อใช้เทียบเคียงกับไออะแกรม (Diagram) ของโครงสร้างข้อมูลที่ได้รับการออกแบบในรูปที่ 4.2 จากไดอะแกรมโครงสร้างข้อมูลชนิดจุดไม่มีการเชื่อมโยงออกไปสู่โครงสร้างข้อมูลชนิดอื่น โดยทำหน้าที่เพียงแค่จัดเก็บค่าพิกัดเท่านั้น โครงสร้างข้อมูลชนิดขอบและโครงสร้างข้อมูลชนิดสามเหลี่ยมไม่มีการเก็บค่าพิกัดโดยตรง แต่ใช้ข้อมูลเชิง

โครงร่างอ้างอิงไปที่จุดพิกัดข้างเป้าหมายแทน การอ้างอิงระหว่างส่วนประกอบเป็นแบบทิศทางเดียว โดยโครงสร้างข้อมูลชนิดขอบจะอ้างอิงไปที่จุดปลาย 2 ข้าง (จุดเริ่มต้นและจุดสิ้นสุด) และโครงสร้างข้อมูลชนิดสามเหลี่ยมจะอ้างอิงไปที่จุดที่มุมทั้งสาม



รูปที่ 4.1 โครงข่ายสามเหลี่ยมตัวอย่าง



รูปที่ 4.2 ไดอะแกรมของโครงสร้างข้อมูลเชิงพิกัด

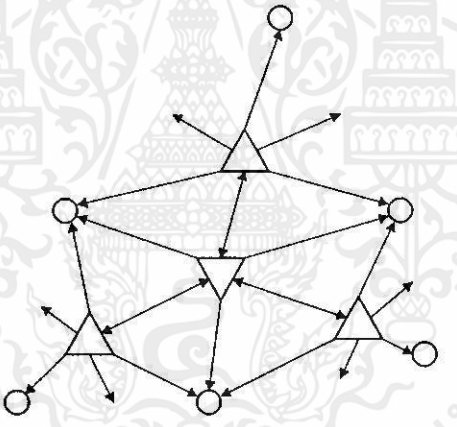
จากไดอะแกรมในรูปที่ 4.2 นำมาเขียนเป็นคลาส (Class) หรือโครงสร้างข้อมูลต้นแบบชนิดจุด, ขอบ และสามเหลี่ยมได้ดังนี้ (เมื่อ * แทนพอยเตอร์ที่อ้างถึงโครงสร้างข้อมูลชนิดอื่น org และ dest ย่อมาจาก original และ destination ตามลำดับ)

```
class Point {
        float x, y, z ;
}
class Edge {
        Point *org, *dest;
}
```

```
class Triangle {
        Point *p1, *p2, *p3;
}
```

4.3 โครงสร้างข้อมูลเชิงโครงร่าง

โครงสร้างข้อมูลเชิงโครงร่าง คือโครงสร้างข้อมูลที่ให้ความสำคัญกับข้อมูลเชิงโครงร่าง ซึ่งแบ่งออกได้เป็น 2 รูปแบบ รูปแบบแรกเป็นชนิด Triangle-Based และรูปแบบที่สองเป็นชนิด Edge-Based [32] ทั้งสองรูปแบบสามารถถูกตรวจเยี่ยม (Traverse) ได้ทั่วทุกส่วนของโครงข่ายสามเหลี่ยม เช่น เดินทางจากสามเหลี่ยมหนึ่งไปสู่อีกสามเหลี่ยมหนึ่ง หรือค้นหาสามเหลี่ยมทุกรูปซึ่งอยู่ล้อมรอบจุดพิจารณานั้น โครงสร้างข้อมูลทั้งสองได้รับการปรับปรุงให้สามารถจัดการง่ายขึ้นจากโครงสร้างข้อมูลแบบเดิม เช่น Doubly-Connected-Edge-List [22], Winged Edge [1], และ Quad Edge [10]

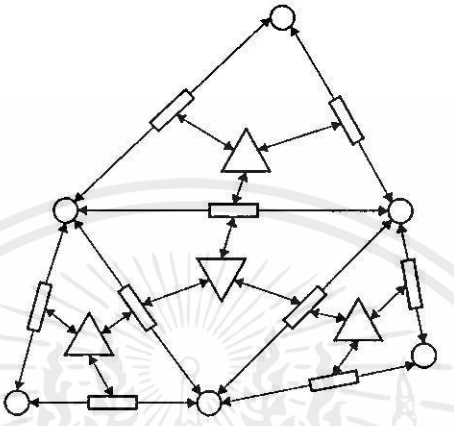


รูปที่ 4.3 ไดอะแกรมของโครงสร้างข้อมูลชนิด Triangle-Based

4.3.1 โครงสร้างข้อมูลชนิด Triangle-Based

ข้อมูลของรูปสามเหลี่ยมและจุดจะถูกจับเก็บในรูปของอ็อปเจ็คท์ (Object) อ็อปเจ็คท์สามเหลี่ยมจะบรรจุข้อมูลอ้างอิงอยู่ 6 ตัว สามตัวแรกสำหรับอ้างอิงถึงสามเหลี่ยมรอบข้าง และอีก 3 ตัวสำหรับอ้างอิงถึงจุดที่มุมทั้ง 3 ดังไดอะแกรมในรูปที่ 4.3 อ็อปเจ็คท์จุดจัดเก็บเพียงค่าพิกัด x, y และ z ไม่มีการอ้างอิงออกไปสู่อ็อปเจ็คท์ข้างเคียง สำหรับข้อมูลสำหรับขอบไม่มีการจัดเก็บอย่างเปิดเผย แต่สามารถอ้างอิงถึงได้ในภายหลัง เมื่อขอบถูกกำหนดขึ้นจากมุม 2 มุมของอ็อปเจ็คท์สามเหลี่ยม จากไดอะแกรมในรูปที่ 4.3 สามารถนำมาเขียนเป็นโครงสร้างข้อมูลต้นแบบได้ดังนี้

```
class TriangleBased {
        TriangleBased *t1, *t2, *t3;
        Point *p1, *p2, *p3;
}
```



รูปที่ 4.4 ไดอะแกรมของโครงสร้างข้อมูลชนิด Edge-Based

4.3.2 โครงสร้างข้อมูลชนิด Edge-Based

โครงสร้างข้อมูลชนิด Edge-Based ประกอบด้วยรายละเอียดดังนี้ เมื่อขอบคืออ็อปเจ็คท์ซึ่งมีจุดประสงค์อยู่ 2 ประการคือ (1) เชื่อมโยงจุดสองจุด และ (2) แบ่งแยกสามเหลี่ยม 2 รูปซึ่งอยู่ติดกัน อ็อปเจ็คท์ขอบจะบรรจุข้อมูลอ้างอิง 2 อันไปที่จุดปลายทั้งสอง และอีก 2 ตัวไปที่สามเหลี่ยมสองรูปซึ่งถูกแบ่งแยกด้วยขอบนี้ อ็อปเจ็คท์สามเหลี่ยมบรรจุข้อมูลอ้างอิงถึงขอบ 3 ขอบซึ่งอยู่รอบข้าง อ็อปเจ็คท์จุดก็บรรจุเพียงค่าพิกัดเช่นเดิม เห็นได้ว่าไม่มีการอ้างอิงระหว่างจุดและสามเหลี่ยมเลย เนื่องจากสามเหลี่ยมสามารถค้นหาจุดที่มุมได้โดยการอ้างผ่านขอบซึ่งชี้ไปที่จุดนั้น จากไดอะแกรมในรูปที่ 4.4 สามารถเขียนเป็นโครงสร้างข้อมูลต้นแบบได้ดังนี้

```
class EdgeBased {
        Point *p1, *p2;
        TriangleEdgeBased *t1, *t2;
}
class TriangleEdgeBased {
        EdgeBased *e1, *e2, *e3;
}
```

4.3.3 ความเหมาะสมของโครงสร้างข้อมูล

โครงสร้างข้อมูลเชิงพิกัดเป็นโครงสร้างข้อมูลที่ไม่ซับซ้อน ข้อมูลพอยเตอร์ที่ใช้มีจำนวนน้อย ทำให้ง่ายต่อการจัดการ แต่มีข้อเสียที่ไม่เหมาะสำหรับการประมวลผลขั้นสูง แม้ว่าการประมวลผลบางอย่างสามารถกระทำได้แต่ก็ต้องใช้เวลาในการประมวลผลมากขึ้น การตรวจเยี่ยมโครงข่ายจากส่วนประกอบหนึ่งไปยังส่วนประกอบอื่นๆไม่สามารถเกิดขึ้นได้เนื่องจากการอ้างอิงข้อมูลต่างไปกระจุกตัวอยู่ที่ข้อมูลเชิงพิกัดเพียงอย่างเดียว

โครงสร้างข้อมูลชนิด Edge-Based และ Triangle-Based จึงได้เปรียบโครงสร้างข้อมูลเชิงพิกัดในเรื่องของความต่อเนื่องของข้อมูลระหว่างส่วนประกอบ การตรวจเยี่ยมและการเข้าถึงข้อมูลในส่วนต่างๆเพื่ออธิบายโครงร่างของโครงข่ายสามเหลี่ยมจึงกระทำได้อย่างเต็มที่ ทำให้สามารถนำไปประมวลผลกับงานประยุกต์ที่ซับซ้อนมากขึ้นได้ แต่การจัดการค่อนข้างยุ่งยากเพราะข้อมูลพอยเตอร์ที่ใช้มีจำนวนมากพอสมควร

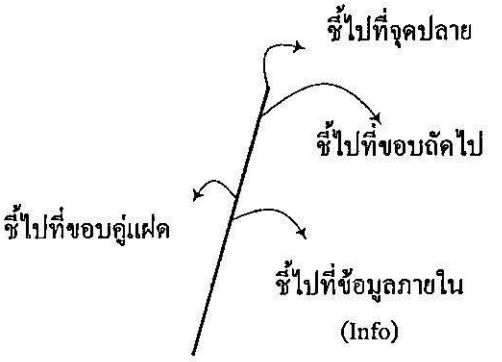
แบบจำลองขนาดใหญ่ต้องการเนื้อที่ในการเก็บข้อมูลจำนวนมาก นอกจากนั้นโครงสร้างข้อมูลสำหรับจัดการกับแบบจำลองโดยใช้โครงข่ายสามเหลี่ยมแบบพลวัติจำเป็นต้องมีความพร้อมสำหรับการเปลี่ยนแปลงอยู่ตลอดเวลา เช่น พร้อมสำหรับการแทรกจุด การลบจุด และการสลับขอบ (อธิบายหัวข้อที่ 2.4.2 และ 3.2) ดังนั้นโครงสร้างข้อมูลต้องง่ายต่อการปรับปรุง

การลดความฟุ่มเฟือยของพอยเตอร์ให้น้อยลงที่สุดก็เป็นสิ่งที่จำเป็น เพราะยิ่งมีพอยเตอร์มากขึ้นเท่าใด การปรับปรุงก็ต้องมีมากขึ้นเท่านั้น เนื้อที่ในการจัดเก็บข้อมูลก็ต้องสิ้นเปลืองเพิ่มขึ้นด้วย อย่างไรก็ตามการประหยัดต้องไม่ทำให้การเชื่อมโยงระหว่างส่วนประกอบภายในโครงข่ายสูญเสียไป โครงสร้างข้อมูลที่ดีจึงต้องมีขนาดกระทัดรัดและรักษาการเชื่อมโยงที่จำเป็นไว้

โครงสร้างข้อมูลเชิงพิกัดใช้พอยเตอร์ทั้งหมด 5 ตัว โครงสร้างข้อมูลชนิด Triangle-Based ใช้พอยเตอร์ทั้งหมด 6 ตัว โครงสร้างข้อมูลชนิด Edge-Based ใช้พอยเตอร์มากที่สุดเป็นจำนวนทั้งสิ้น 7 ตัว ในหัวข้อถัดไปจะนำเสนอโครงสร้างข้อมูลซึ่งได้รับการปรังปรุงให้มีขนาดเล็กลงโดยใช้พอยเตอร์เพียงแค่ 4 ตัว แต่สามารถปฏิบัติงานได้เช่นเดียวกับโครงสร้างข้อมูลชนิด Triangle-Based และ Edge-Based

4.4 โครงสร้างข้อมูล TwinEdge

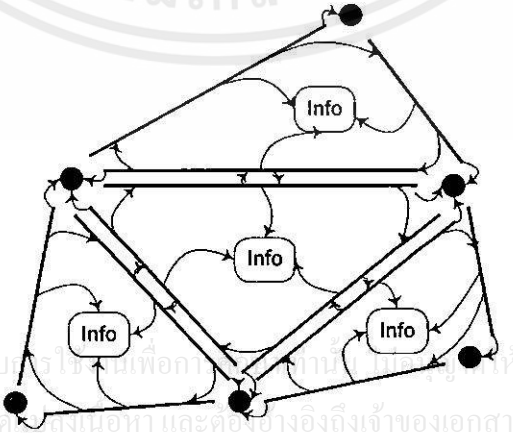
TwinEdge เป็นโครงสร้างข้อมูลแบบกระทัดรัดที่มีพื้นฐานเช่นเดียวกับโครงสร้างข้อมูลชนิด Edge-Based แต่ต่างกันตรงที่ไม่มีการกำหนดโครงสร้างข้อมูลชนิดสามเหลี่ยม ข้อมูลของสามเหลี่ยมถูกเก็บแฝงไว้ในโครงสร้างข้อมูลชนิดขอบ เมื่อขอบซึ่งเชื่อมต่อจุด 3 จุดในแบบวงรอบปิดแสดงถึงสามเหลี่ยมหนึ่งรูป โครงข่ายประกอบด้วยโครงสร้างข้อมูลเพียง 2 ชนิดคือ ขอบและจุด โครงสร้างข้อมูลชนิดขอบจะเป็นตัวดำเนินการหลักในการเชื่องโยงความสัมพันธ์ระหว่างส่วนประกอบภายในโครงข่าย ในรูปที่ 4.5 แสดงโครงสร้างของ TwinEdge



รูปที่ 4.5 โครงสร้างของ TwinEdge

TwinEdge ประกอบด้วยพอยเตอร์ 4 ตัวคือ (1) พอยเตอร์ชี้ไปที่จุดปลาย (2) พอยเตอร์ชี้ไปที่ขอบถัดไปในรูปสามเหลี่ยม (3) พอยเตอร์ชี้ไปที่ขอบคู่แฝด (Twin edge หรือ Dual edge) ขอบคู่แฝดคือขอบซึ่งอยู่เคียงคู่กัน มีลักษณะทางกายภาพเหมือนกัน ต่างกันที่พอยเตอร์สำหรับชี้ไปที่จุดเป้าหมายจะตรงข้ามกัน และ (4) พอยเตอร์ชี้ไปที่ข้อมูลเพิ่มเติม โดยขอบแต่ละขอบซึ่งประกอบเป็นรูปสามเหลี่ยมจะชี้ไปที่ข้อมูลร่วม (Info) ซึ่งบรรจุเซ็ตของจุดไว้ภายในรูปสามเหลี่ยม โครงสร้างข้อมูลต้นแบบของ TwinEdge แสดงได้ดังนี้

```
class TwinEdge {
        Point *point;
        TwinEdge *next;
        TwinEdge *twin;
        Info *info;
}
```



รูปที่ 4.6 การเชื่อมโยงระหว่าง TwinEdge

หนึ่งอ็อปเจ็คท์ของ TwinEdge คือด้านใดด้านหนึ่งของสามเหลี่ยมเพียงหนึ่งรูป ไม่มีการใช้ TwinEdge ร่วมกันระหว่างรูปสามเหลี่ยมซึ่งมีด้านติดกัน สามเหลี่ยมหนึ่งรูปใช้ TwinEdge จำนวน 3 อ็อปเจ็คท์ ดังนั้นถ้าในโครงข่ายมีสามเหลี่ยมจำนวน m สามเหลี่ยม TwinEdge ที่ใช้ในโครงข่ายสามเหลี่ยมนี้จึงมีจำนวนทั้งสิ้น $3 \cdot m$ อ็อปเจ็คท์ รูปที่ 4.6 แสดงการเชื่อมโยงระหว่าง TwinEdge สำหรับโครงข่ายสามเหลี่ยมเทียบเคียงในรูปที่ 4.1

4.5 การจัดการกับจุดระหว่างการสร้างโครงข่ายสามเหลี่ยม

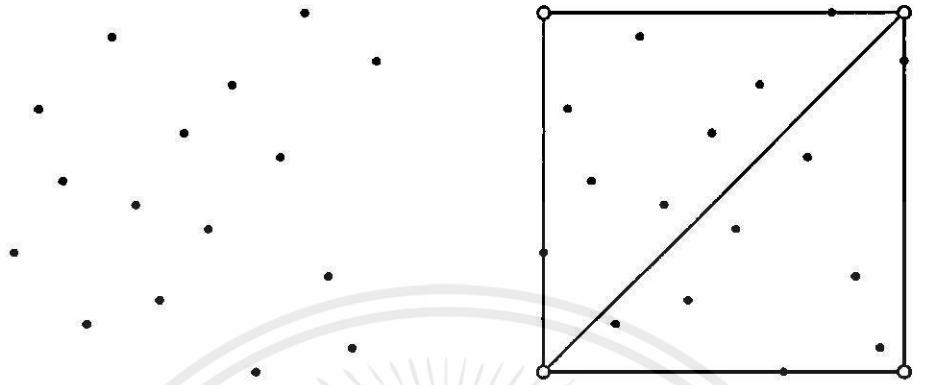
ข้อมูลซึ่งแทนตัวแหน่งสำคัญบนพื้นผิวของภูมิประเทศประกอบขึ้นจากเซ็ตของจุดจำนวนมาก โครงสร้างการจัดการกับจุดที่มีประสิทธิภาพระหว่างการสร้างโครงข่ายสามหลี่ยมจึงเป็นสิ่งจำเป็น เนื่องจากเวลาส่วนใหญ่จะสูญเสียไปกับการจัดการกับจุด เช่น การจัดเก็บ การเรียงลำดับ และการค้นหาจุดที่ต้องการเป็นต้น

จากงานวิจัยที่ผ่านมา [2] ได้ใช้ Quadtree [23][24] ในการจัดการกับเซ็ตของจุดสำหรับอัลกอริทึม Step-by-Step เมื่อ Quadtree คือโครงสร้างข้อมูลที่ถูกออกแบบมาเพื่อประโยชน์ในการค้นจุดเป้าหมายได้อย่างรวดเร็ว แต่ใน [19] ผู้ซึ่งนำเสนออัลกอริทึม Step-by-Step เป็นครั้งแรกได้ใช้โครงสร้างข้อมูล Grid ในการกำหนดตำแหน่งของจุดเป้าหมาย ใน [10] ได้ใช้ลิงค์ลิสต์สำหรับจัดการกับเซ็ตของจุด เมื่อลำดับของจุดจะถูกแบ่งย่อยลงไปเรื่อยๆในขั้นตอนเริ่มต้นของอัลกอริทึม Divide-and-Conquer (อธิบายในหัวข้อ 2.4.3) จากตัวอย่างของงานวิจัยที่กล่าวมาพบว่าเซ็ตของจุดจะถูกจัดการแยกออกจากโครงสร้างของโครงข่ายสามเหลี่ยม

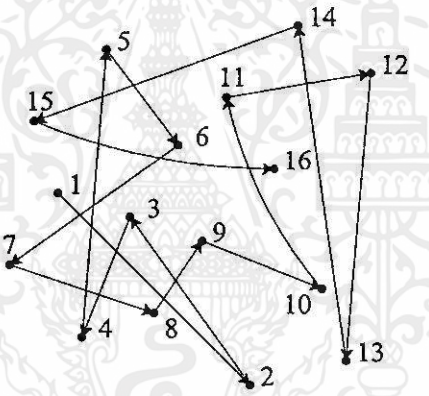
สำหรับการสร้างโครงข่ายสามเหลี่ยมโดยใช้อัลกอริทึม Incremental และมี TwinEdge เป็นโครงสร้างข้อมูลหลัก เซ็ตข้อมูลอินพุทและโครงข่ายสามเหลี่ยมจะไม่ถูกแยกออกจากกัน โครงข่ายสามเหลี่ยมจะทำหน้าที่เป็นสถานที่สำหรับจัดเก็บเซ็ตข้อมูลไปในตัว รูปที่ 4.7 แสดงเซ็ตของจุดตัวอย่างและโครงข่ายเริ่มต้น (Initial network) สำหรับเซ็ตของจุดตัวอย่าง โครงข่ายสามเหลี่ยมเริ่มต้นมีรูปร่างเป็นรูปสี่เหลี่ยมด้านขนาน (Circumscribing rectangle) ถูกแบ่งครึ่งโดยเส้นทะแยงมุมของรูปสี่เหลี่ยมออกเป็นสามเหลี่ยม 2 รูป โครงข่ายเริ่มต้นในรูปแบบนี้สร้างได้ง่ายและรวดเร็วกว่าโครงข่ายสามเหลี่ยมของคอนเว็กซ์ฮัล (อธิบายในหัวข้อที่ 2.4.5 และ 2.4.6)

จุดทั้งหมดในเซ็ตจะถูกจัดเรียงในลิงค์ลิสต์ดังรูปที่ 4.8 และคัดแยกลงสู่ลิงค์ลิสต์ของแต่ละสามเหลี่ยมซึ่งปิดล้อมจุดนั้นๆอยู่ ดังรูปที่ 4.9 การคัดเลือกสามเหลี่ยมเป้าหมายสำหรับจัดเก็บจุดคำนวณได้จากสมการที่ (4.1) ในรูปที่ 4.10 แสดงการเปลี่ยนแปลงของลิงค์ลิสต์เมื่อมีจุดใหม่ถูกแทรกเข้าสู่โครงข่าย จุดแทรกใหม่จะถูกแยกออกจากลิงค์ลิสต์เดิม แล้วเชื่อมต่อกับมุมทั้งสามของรูปสามเหลี่ยมที่ปิดล้อมจุดนี้อยู่ทำให้เกิดสามเหลี่ยมใหม่ภายในสามเหลี่ยมเดิม 3 รูป จุดที่เหลือจะถูกถ่ายเทลงสู่สามเหลี่ยมใหม่เหล่านี้ ขอบเดิมซึ่งล้อมรอบสามเหลี่ยมใหม่ทั้งสามที่ได้รับผล

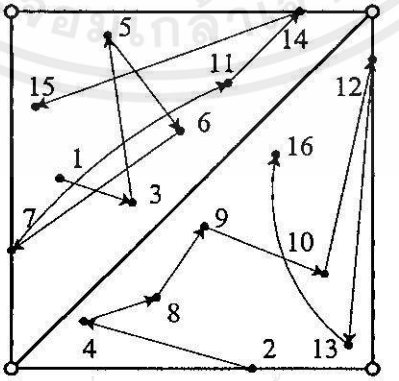
กระทบจากการแทรกจุดจะได้รับการปรับโครงสร้างโดยการสลับขอบ จุดซึ่งอยู่ระหว่างสามเหลี่ยมซึ่งได้รับผลกระทบก็จะได้รับการถ่ายเทลงสู่สามเหลี่ยมที่เหมาะสม ดังรูปที่ 4.11



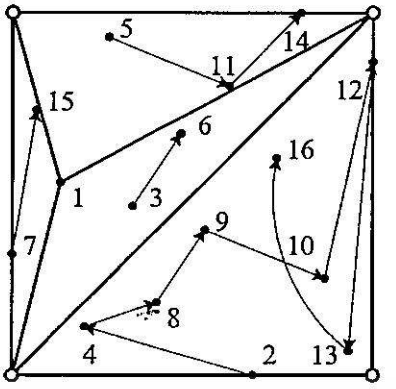
รูปที่ 4.7 เซ็ตของจุดและโครงข่ายสามเหลี่ยมเริ่มต้น



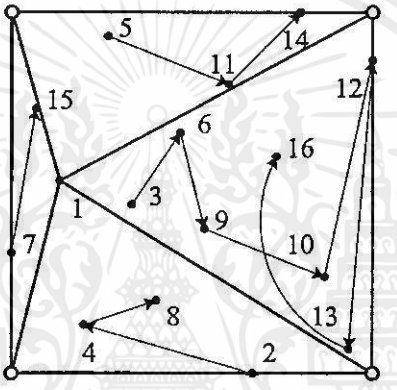
รูปที่ 4.8 ลำดับของจุดตัวอย่างก่อนที่จะนำเข้าสู่โครงข่ายเริ่มต้น



รูปที่ 4.9 ลำดับจุดถูกแบ่งออกเป็นสองส่วน



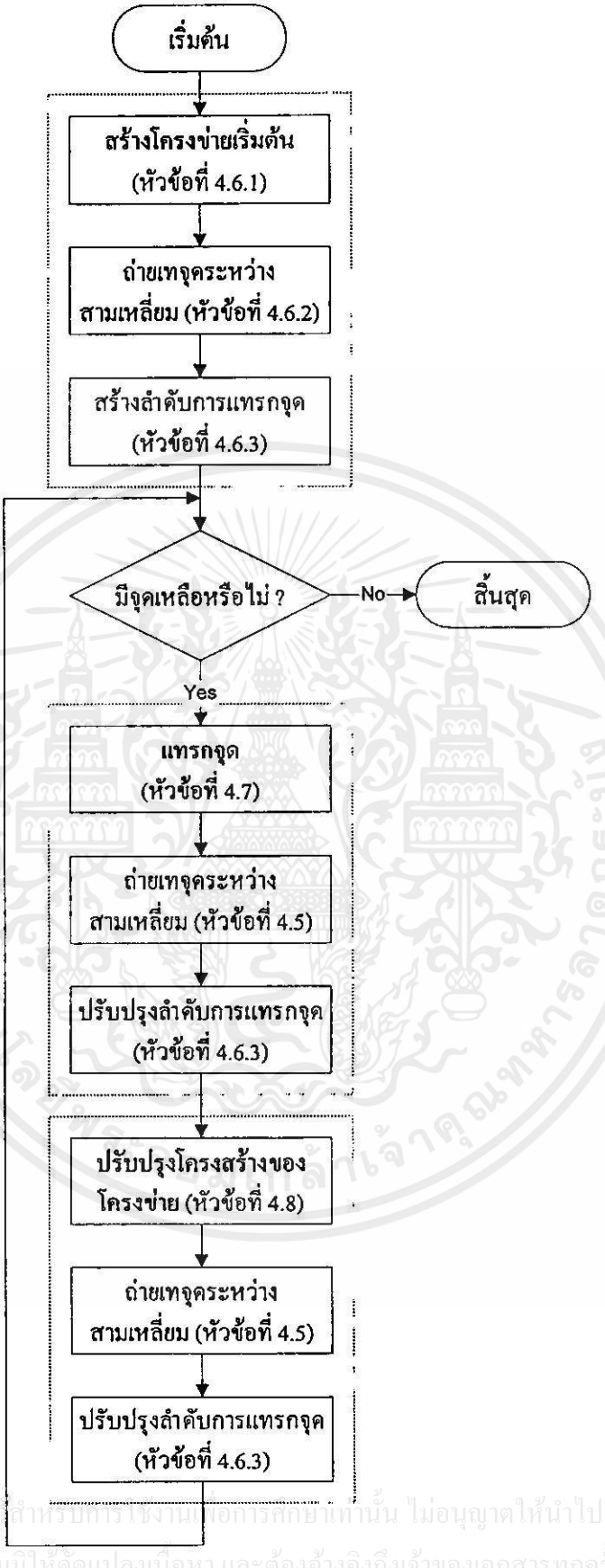
รูปที่ 4.10 ลำดับจุดถูกแบ่งออกเป็น 3 ส่วนเมื่อจุดแรกถูกแทรกเข้าสู่สามเหลี่ยม



รูปที่ 4.11 ลำดับของจุดได้รับการถ่ายเทระหว่างสามเหลี่ยมหลังจากการสลับขอบ

4.6 ขั้นตอนหลักของการสร้างโครงข่ายสามเหลี่ยม

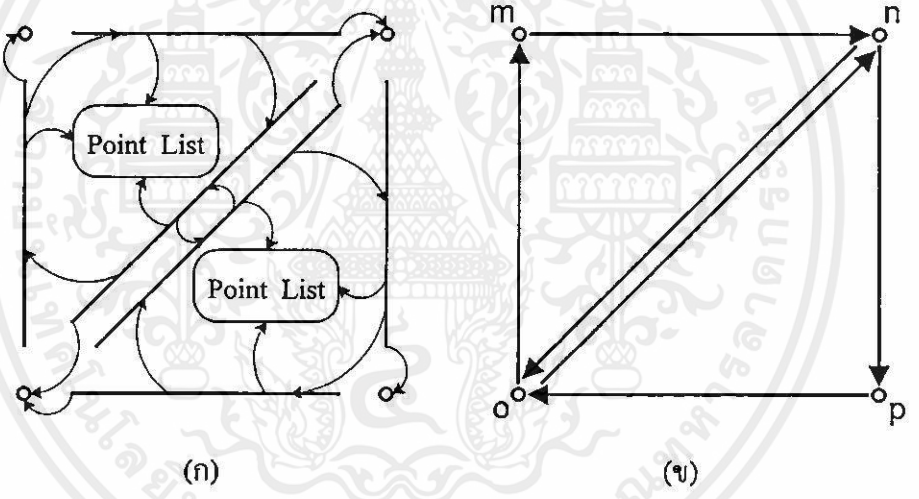
การสร้างโครงข่ายสามเหลี่ยมโดยใช้อัลกอริทึม Incremental แบ่งออกเป็น 3 ขั้นตอนหลัก คือ (1) การสร้างโครงข่ายเริ่มต้น (2) การแทรกจุด และ (3) การปรับโครงสร้างของโครงข่าย ขั้นตอนการทำงานทั้งหมดแสดงได้ดังโฟลว์ชาร์ตในรูปที่ 4.12 เมื่อโครงข่ายเริ่มต้นถูกสร้างขึ้นจุดทั้งหมดจะถูกถ่ายเทลงสู่สามเหลี่ยมเริ่มต้น และลำดับของการแทรกจุดจะถูกสร้างขึ้นตามมา จากนั้นการทำงานจะเข้าสู่วงรอบของการแทรกจุด และการปรับโครงสร้างของโครงข่าย จนกระทั่งไม่เหลือจุดที่จะถูกแทรกเข้าสู่โครงข่ายอีกต่อไป หลังจากการแทรกจุดและการปรับโครงสร้างสิ้นสุดลงแต่ละครั้ง จุดภายในพื้นที่เปลี่ยนแปลงนั้นจะได้รับการถ่ายเทระหว่างสามเหลี่ยม และลำดับการแทรกจุดจะได้รับการปรับปรุงตามไปด้วย



รูปที่ 4.12 ขั้นตอนการสร้างโครงข่ายสามเหลี่ยมโดยใช้อัลกอริทึม Incremental

4.6.1 การสร้างโครงข่ายเริ่มต้น

โครงข่ายเริ่มต้นสร้างขึ้นจาก TwinEdge ทั้งหมด 6 อ็อปเจ็คท์ซึ่งเชื่อมต่อกันดังรูปที่ 4.13 (ก) TwinEdge ซึ่งอยู่ขอบนอกสุดจะถูกกำหนดค่าของพอยเตอร์ twin = null เพราะไม่มีขอบคู่แฝดใดให้อ้างถึง (Null pointers นี้จะถูกใช้ประโยชน์ในขั้นตอนการสำรวจโครงข่ายในหัวข้อที่ 4.9) รูปที่ 4.13(ข) แสดงโครงข่ายเริ่มต้นอย่างง่ายเทียบเคียงกับโครงข่ายของ TwinEdge ที่เชื่อมโยงอย่างสมบูรณ์ในรูปที่ 4.13(ก) TwinEdge หนึ่งอ็อปเจ็คท์จะถูกสมมุติให้แทนด้วยลูกศรหนึ่งรูปซึ่งปลายของลูกศรจะชี้ไปที่จุดเป้าหมาย จุดที่มุมทั้ง 4 ของโครงข่ายเริ่มต้นถูกกำหนดขึ้นจากการเปรียบเทียบหาค่าพิกัด x และ y ที่น้อยที่สุดและมากที่สุดจากพิกัดของเซ็ตจุดทั้งหมด (minx, miny, maxx และ maxy) เมื่อ o คือจุดที่มุมล่างซ้าย, m คือจุดที่มุมบนซ้าย, n คือจุดที่มุมบนขวา และ p คือจุดที่มุมล่างขวา ดังนั้น o = (minx, miny), m = (minx, maxy), n = (maxx, maxy) และ p = (maxx, maxy) ดังรูปที่ 4.13(ข)



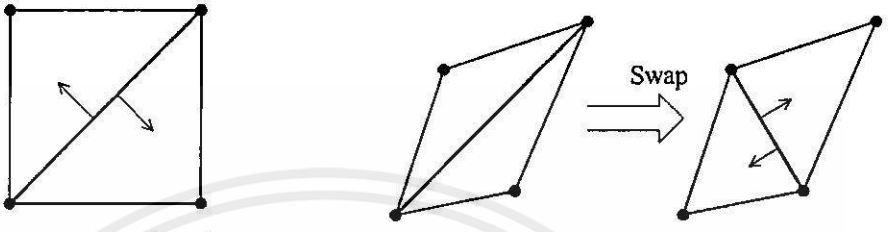
รูปที่ 4.13 โครงข่ายสามเหลี่ยมเริ่มต้นในรูป TwinEdge และโครงสร้างเทียบเคียงอย่างง่าย

4.6.2 การถ่ายเทจุดระหว่างสามเหลี่ยม

หลังจากที่มีการเปลี่ยนแปลงกับสามเหลี่ยมเกิดขึ้นในขั้นตอนต่างๆ (การสร้างโครงข่ายเริ่มต้น การแทรกจุด และการสลับขอบ) เซ็ตของจุดซึ่งบรรจุดอยู่ภายในพื้นที่เปลี่ยนแปลงนั้นต้องได้รับการถ่ายเทลงสู่สามเหลี่ยมที่เหมาะสม ในกรณีของการถ่ายเทจุดในขั้นตอนของการสร้างโครงข่ายเริ่มต้นและการสลับขอบจะเหมือนกันคือ การถ่ายเทจุดจะเกิดขึ้นภายในกรอบของรูปสี่เหลี่ยมด้านไม่เท่า โดยภายในรูปสี่เหลี่ยมจะมีเส้นทะแยงมุมเป็นตัวตัดสินว่าจุดนั้นๆควรจะอยู่ในสามเหลี่ยมข้างใดดังรูปที่ 4.14

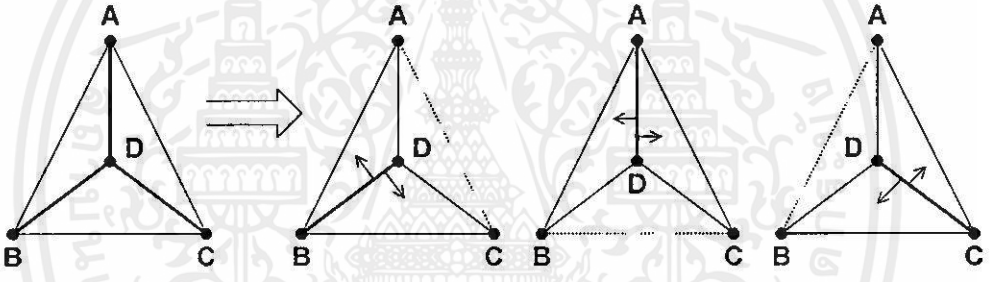
สำหรับกรณีของการถ่ายเทจุดในขั้นตอนการแทรกจุดจะซับซ้อนยิ่งขึ้น เนื่องจากเมื่อมีจุดใหม่ถูกแทรกเข้าสู่สามเหลี่ยมใดสามเหลี่ยมนั้นก็จะถูแบ่งออกเป็นสามเหลี่ยมย่อย 3 รูป อย่างไรก็

ตามหลักการของการตัดสินว่าจุดนั้นควรจะอยู่ในสามเหลี่ยมใดยังใช้หลักการเดิม คือมีเส้นทะแยงมุมเส้นหนึ่งเป็นตัวตัดสิน โดยที่ขอบใหม่จำนวน 3 ขอบซึ่งถูกสร้างขึ้นเพื่อเชื่อมต่อระหว่างมุมของสามเหลี่ยมและจุดใหม่จะเป็นเสมือนเส้นทะแยงมุมของสี่เหลี่ยมด้านไม่เท่า (ชนิดไม่เป็นคอนเว็กซ์) ดังรูปที่ 4.15



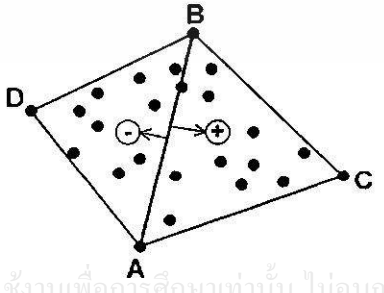
(ก) สามเหลี่ยมเริ่มต้น (ข) การสลับขอบ

รูปที่ 4.14 เส้นทะแยงมุมภายในรูปสี่เหลี่ยมด้านไม่เท่า



รูปที่ 4.15 เส้นทะแยงมุมของรูปสี่เหลี่ยมด้านไม่เท่าชนิดไม่เป็นคอนเว็กซ์ภายในรูปสามเหลี่ยม

การคำนวณหาด้านของจุดที่กระทำกับเส้นทะแยงมุม ดังรูปที่ 4.16 สามารถคำนวณได้จากสมการของการหาระยะทางระหว่างจุดกับเส้นตรง ดังสมการที่ (4.1) เมื่อ d คือระยะทางระหว่างจุดใดๆกับ $\overline{AB}$ เครื่องหมายของ d จะเป็นตัวบอกด้านของจุด



รูปที่ 4.16 การใช้เครื่องหมายของค่าระยะทางเพื่อตัดสินด้านของจุด

$$d = \frac{ax + by + c}{\sqrt{a^2 + b^2}} \tag{4.1}$$

เมื่อ $a = y_A - y_B$, $b = x_B - x_A$, $c = x_A y_B - y_A x_B$

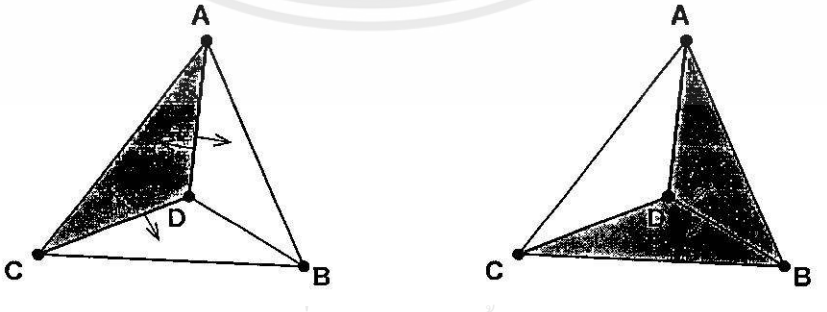
เนื่องจากผลลัพธ์ที่ต้องการจากสมการที่ (4.1) คือเครื่องหมายไม่ใช่ค่าระยะทาง ดังนั้นจึงสามารถลดทอนความซับซ้อนของสมการที่ (4.1) เป็นสมการอย่างง่ายที่ (4.2) โดยการคำนวณเฉพาะเศษบน (Numerator) ของสมการ ซึ่งจะช่วยลดเวลาในการคำนวณลงไปได้มาก

$$s_i = ax_i + by_i + c \tag{4.2}$$

เมื่อ $(i = 1,2,..,n)$

n คือ จำนวนของจุดทั้งหมดซึ่งอยู่ภายในสี่เหลี่ยมด้านไม่เท่า เมื่อผลลัพธ์ S_i มีค่าเป็นบวก (+) แสดงว่าจุดนั้นอยู่ทางด้านขวาของเส้นทะแยงมุม เมื่อ S_i เป็นลบ (-) แสดงว่าจุดนั้นอยู่ทางด้านซ้ายของเส้นทะแยงมุม และเมื่อ S_i เท่ากับศูนย์แสดงว่าจุดนั้นซ้อนทับอยู่บนเส้นทะแยงมุมพอดี เมื่อจุดถูกคำนวณว่าอยู่ทางด้านขวาของเส้นทะแยงมุมดังนั้นจุดนี้ก็จะถูกบรรจุลงสู่สามเหลี่ยมอันขวา และจะถูกบรรจุลสู่สามเหลี่ยมอันซ้ายเมื่ออยู่ทางด้านซ้ายของเส้นทะแยงมุม สำหรับจุดซึ่งมีตำแหน่งซ้อนทับอยู่บนเส้นทะแยงมุมพอดี ในทางปฏิบัติจุดนี้จะถูกกำหนดให้อยู่กับสามเหลี่ยมใดสามเหลี่ยมหนึ่งก็ได้ (จากการทดลองกำหนดให้อยู่กับสามเหลี่ยมอันขวามือเสมอ)

สำหรับการถ่ายเทจุดเมื่อสามเหลี่ยมถูกแบ่งออกเป็น 3 ส่วน มีเทคนิคในการจัดการดังนี้ จากรูปที่ 4.17(ก) จุดทั้งหมดใน ΔABC จะถูกคัดเลือกโดยเส้นทะแยงมุม $\overline{DA}$ และ $\overline{DC}$ เพื่อบรรจุลงสู่ ΔADC เป็นอันดับแรก จากนั้นจุดที่เหลือจะถูกแบ่งแยกลงสู่ ΔABD และ ΔDBC โดยใช้ $\overline{DB}$ เป็นเส้นทะแยงมุมเปรียบเทียบดังรูปที่ 4.17(ข)



(ก) (ข)

รูปที่ 4.17 การถ่ายเทจุดจาก ΔABC ไปสู่ ΔADC, ΔABD และ ΔDBC

4.6.3 วิธีการสำหรับการแทรกจุด

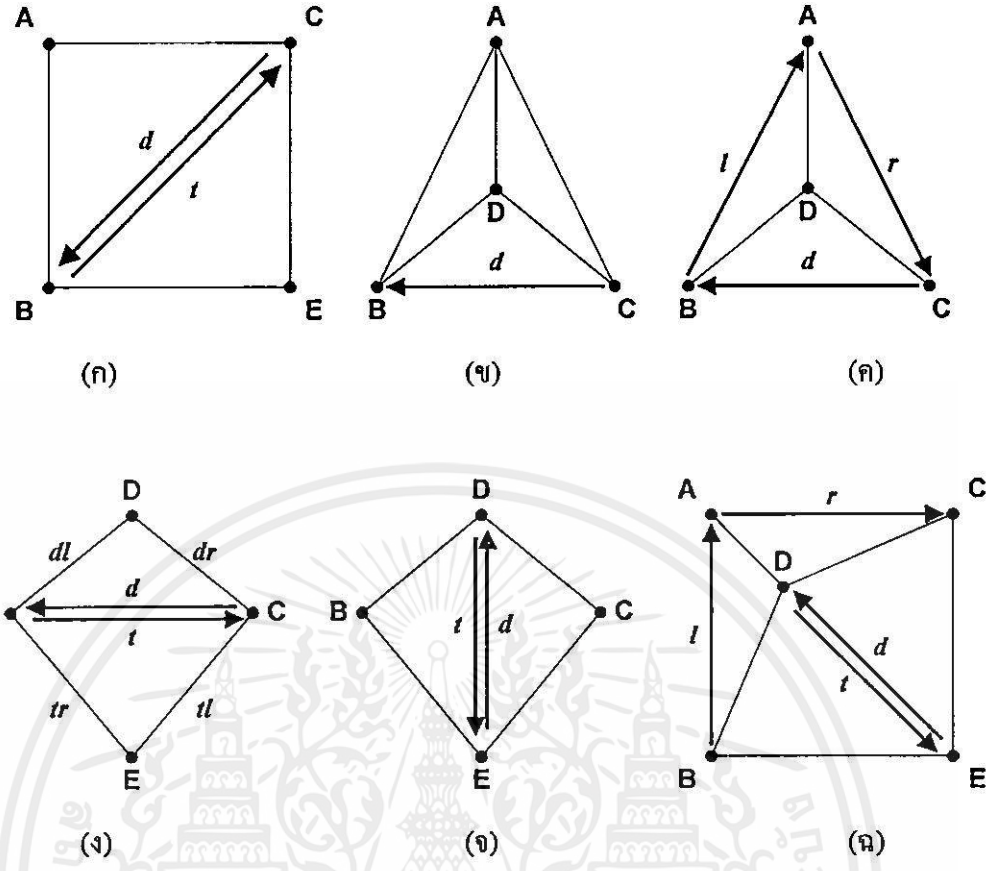
การแทรกจุดคือขั้นตอนการทำงานที่สำคัญของอัลกอริทึม Incremental ในขณะที่หัวข้อที่ 3.3 กล่าวถึงเงื่อนไขทางเรขาคณิตต่างๆสำหรับการสลับขอบและผลกระทบที่เกิดขึ้นตามมา ในหัวข้อนี้จะนำเสนออัลกอริทึม, วิธีการ และรายละเอียดเสริมที่เกี่ยวข้องกับขั้นตอนการแทรกจุด

4.6.3.1 ลำดับของการแทรกจุด

ลำดับของการแทรกจุดคือลำดับของสามเหลี่ยมซึ่งยังมีจุดบรรจุอยู่ภายในเพื่อรอการถูกดึงออกมาใช้งานในการแทรกจุด เมื่อโครงข่ายสามเหลี่ยมเริ่มต้นได้ถูกสร้างขึ้นและจุดทั้งหมดได้ถูกถ่ายเทลงสู่สามเหลี่ยมที่เหมาะสมเรียบร้อย ทำให้มีสามเหลี่ยมอย่างน้อย 2 รูปซึ่งบรรจุจุดอยู่ภายใน ลำดับของสามเหลี่ยมสำหรับการแทรกจุดจึงถูกสร้างขึ้นแยกออกจากโครงข่ายสามเหลี่ยม แต่เนื่องจากโครงสร้างข้อมูลที่ถูกเลือกใช้คือ TwinEdge ซึ่งไม่มีการจัดเก็บข้อมูลของสามเหลี่ยมโดยตรง ดังนั้นลำดับของการแทรกจุดจึงอยู่ในรูปของลำดับของขอบ (TwinEdges) ซึ่งเป็นตัวแทนของรูปสามเหลี่ยมนั้นๆ TwinEdge หนึ่งอ็อปเจ็คท์ต่อสามเหลี่ยมหนึ่งรูป

พิจารณารูปที่ 4.18(ก) เมื่อตรวจสอบแล้วพบว่าภายใน ΔACB และ ΔBCE ต่างก็มีจุดบรรจุอยู่ภายใน ดังนั้นลำดับการแทรกจุดเริ่มต้นจึงประกอบด้วยขอบ *d* และ *t* (แทนด้วย *d->t*) *d* คือขอบแรกที่จะถูกดึงออกมาจากลำดับ (ทำให้เหลือเฉพาะ *t*) เนื่องจากมีจุด *D* ที่ต้องถูกแทรกดังรูปที่ 4.18(ข) ทำให้ภายในพื้นที่ของ ΔACB ถูกแบ่งเป็นสามเหลี่ยมใหม่ 3 รูปคือ ΔADB, ΔACD และ ΔDCB สามเหลี่ยมแต่ละรูปจะถูกตรวจสอบว่ามีจุดบรรจุอยู่ภายในหรือไม่ (หลังจากการถ่ายเทจุดสิ้นสุดลง) สมมุติว่าทั้ง 3 รูปมีจุดบรรจุอยู่ภายใน ขอบ *l* จะทำหน้าที่แทน ΔADB, *r* แทน ΔACD และ *d* รับหน้าที่ใหม่แทน ΔDCB ดังรูปที่ 4.18(ค) ขอบทั้ง 3 จะถูกเพิ่มเข้าสู่ลำดับของขอบเป็น *t->l->r->d*

จากรูปที่ 4.18(ง) แสดงกรณีที่ *d* และ *t* ต้องถูกสลับขอบภายใน $\square DCBE$ ซึ่งประกอบด้วยสามเหลี่ยมเดิมคือ ΔDCB และ ΔBCE ให้เป็นสามเหลี่ยมใหม่คือ ΔDCE และ ΔDEB ดังรูปที่ 4.18(จ) ดังนั้น *d* และ *t* จึงต้องถูกลบออกจากลำดับของการแทรกจุดเป็น *l->r* ขอบทั้ง 4 ซึ่งอยู่ล้อมรอบ $\square DCEB$ (*dl*, *dr*, *tl* และ *tr*) จะถูกตรวจสอบว่ามีอยู่ในลำดับของการแทรกจุดหรือไม่ ถ้าพบว่ามีก็จะถูกลบออกจากลำดับทันทีเพื่อกำจัดความซ้ำซ้อน เนื่องจากจะมีเพียง *d* และ *t* เท่านั้นที่จะรับผิดชอบสามเหลี่ยมใหม่ทั้ง 2 นี้ เมื่อตรวจสอบแล้วพบว่าภายใน ΔDCE และ ΔDEB มีจุดบรรจุอยู่ภายใน *d* และ *t* ก็จะถูกเพิ่มเข้าสู่ลำดับของขอบอีกครั้งเป็น *l->r->d->t* เพื่อรอการถูกดึงออกมาใช้งานต่อไป เมื่อขอบใดไม่มีจุดบรรจุอยู่ภายในแล้วก็จะไม่ถูกรวมเข้ามาในลำดับอีก เห็นได้ว่าลำดับของขอบซึ่งแทนสามเหลี่ยมสำหรับการแทรกจุดจะได้รับการปรับปรุงตลอดเวลาการสร้างโครงข่ายสามเหลี่ยม รูปที่ 4.18(ฉ) แสดงขอบในลำดับทั้ง 4 ซึ่งแทนสามเหลี่ยม 4 รูปในโครงข่ายสามเหลี่ยม



รูปที่ 4.18 การสร้างและการปรับปรุงลำดับของขอบสำหรับการแทรกจุด

(เมื่อ d = diagonal, t = twin edge, l = left และ r = right)

4.6.3.2 ลำดับก่อนหลังในการเลือกขอบ

ถ้าจุดทุกจุดจากเซ็ตของข้อมูลต้องถูกรวมไว้ในโครงข่าย ลำดับก่อนหลัง (Succession) ของการแทรกจุดจะมีผลเพียงเล็กน้อย ผลลัพธ์จากการคำนวณจะได้รูปแบบของโครงข่ายที่เหมือนเดิมเสมอแม้ว่าลำดับก่อนหลังของการแทรกจัดจะไม่เหมือนกัน อย่างไรก็ตามลำดับก่อนหลังก็อาจมีผลกระทบในเรื่องของเวลาที่ใช้ในการคำนวณเนื่องมาจากโครงสร้างของการจัดการเซ็ตจุด ดังนั้นการแบ่งสามเหลี่ยมซึ่งมีพื้นที่มากๆก่อนจึงน่าจะเป็นข้อได้เปรียบ อย่างไรก็แล้วแต่วิธีการคำนวณใดๆสำหรับการค้นหาจุดซึ่งเหมาะสมที่สุดในการแบ่งพื้นที่สามเหลี่ยมหรือการแทรกจุดนี้ต่างก็ใช้เวลาค่อนข้างมาก

การเลือกขอบตามลำดับเข้าก่อนออกก่อน (FIFO: First-In-First-Out หรือ Ordinal succession) เป็นวิธีการที่พื้นฐานที่สุดและทำได้สะดวกที่สุด เมื่อจุดใดๆถูกแทรกเข้าสู่โครงข่ายเสร็จอัลกอริทึมก็จะเคลื่อนที่ไปสู่ขอบถัดไป การเลือกขอบจากลำดับยังสามารถพัฒนาต่อไป โดยการเลือกแบบสุ่ม (Randomized succession) ซึ่งให้ผลลัพธ์ในการทำงานเป็นที่น่าพอใจเพราะจุดที่ถูกแทรกนั้นกระจายไปตามส่วนต่างๆของโครงข่ายสามเหลี่ยม ส่งผลให้การปรับโครงสร้างของโครงข่ายกินพื้นที่น้อย (ดังอธิบายในหัวข้อที่ 3.3.3.2) การสุ่มทำได้ถึง 3 รูปแบบคือ (1) สุ่มเมื่อ

แทรกขอบเข้าสู่ลำดับ (2) สุ่มเมื่อดึงขอบออกจากลำดับ และ (3) สุ่มทั้งตอนแทรกและตอนดึงขอบเข้าออกจากลำดับ

วิธีการเลือกขอบแบบตามลำดับและแบบสุ่มสรุปเป็นขั้นตอนได้ดังอัลกอริทึมที่ 4.1 และ 4.2 ตามลำดับ อัลกอริทึมที่ 4.3 เป็นอัลกอริทึมหลักสำหรับการสร้างโครงข่ายสามเหลี่ยมซึ่งจะตัดสินว่าจะเลือกใช้วิธีการจัดลำดับขอบแบบใด และอัลกอริทึมที่ 4.4 จะสรุปขั้นตอนการสร้างโครงข่ายเริ่มต้น

อัลกอริทึมที่ 4.1 the Ordinal Succession

```
algorithm ordinalSuccession()
  while edgeList.isNotEmpty() do
    e = edgeList.getFirstEdge();
    splitTriangle(e);
  endwhile
end
```

อัลกอริทึมที่ 4.2 the Randomized Succession

```
algorithm randomizedSuccession()
  while edgeList.isNotEmpty() do
    r = randomNumber(edgeList.size());
    e = edgeList.getEdgeAt(r);
    splitTriangle(e);
  endwhile
end
```

อัลกอริทึมที่ 4.3 the Incremental Triangulation

```
algorithm incrementalTriangulate(pointSet, succession)
  constructInitialNetwork(poinSet);
  if succession.isOrdinal() then
    ordinalSuccesssion();
  else if succession.isRandomized() then
    randomizedSuccession();
  endif
end
```

อัลกอริทึมที่ 4.4 Constructing an Initial Network

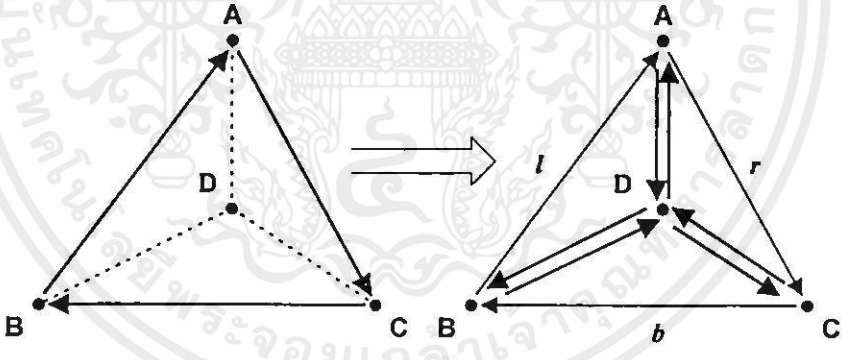
```
algorithm constructInitialNetwork(pointSet)
  findMinMaxXYcoordinates(pointSet);
  constructSixNewEdges(m, n, o, p);
  transferPointBetweenTwoTriangles(on, pointSet);
  createEdgeSuccession(on, no);
end
```

อัลกอริทึมที่ 4.5 Splitting a Triangle

```
algorithm splitTriangle(basedEdge)
  D = basedEdge.point();
  constructSixNewEdges(D, b, l, r);
  transferPointsBetweenThreeTriangles(BD, AD, CD);
  updateEdgeSuccession(l, r, b);
  reorganize(b);
  reorganize(l);
  reorganize(r);
end
```

4.7 การแบ่งสามเหลี่ยม

เมื่อจุดถูกแทรกเข้าสู่โครงข่าย ขอบจำนวน 6 ขอบ (TwinEdge จำนวน 3 คู่) จะถูกสร้างขึ้นใหม่เพื่อเชื่อมต่อกับจุดที่มุมทั้ง 3 ของรูปสามเหลี่ยมซึ่งปิดล้อมจุดนี้อยู่ (ดังรูปที่ 4.19) สามเหลี่ยมเดิมจึงถูกแบ่งออกเป็น 3 ส่วน พอยเตอร์ next ของขอบเดิม (*b*, *l* และ *r*) จะถูกปรับเปลี่ยนให้ชี้ไปที่ขอบใหม่ หลังจากนั้นขอบเดิมทั้ง 3 จะต้องถูกส่งไปตรวจสอบเพื่อการปรับโครงสร้างต่อไป (อธิบายในบทที่ 3) ขั้นตอนของการแทรกจุดหรือการแบ่งสามเหลี่ยมสรุปได้ดังอัลกอริทึมที่ 4.5



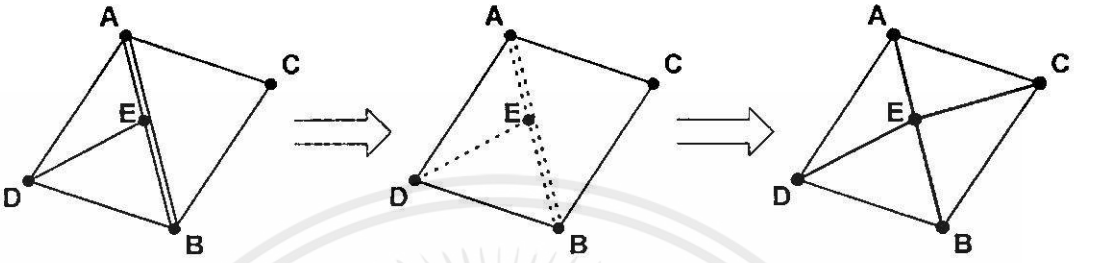
รูปที่ 4.19 TwinEdge จำนวน 6 อ็อปเจ็คท์ถูกสร้างขึ้นเมื่อสามเหลี่ยมถูกแบ่งแยกออกเป็น 3 ส่วน

4.7.1 สามเหลี่ยมศูนย์

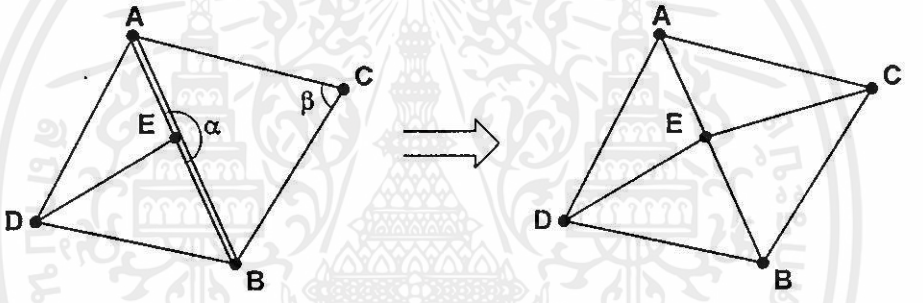
ในบางครั้งตำแหน่งของจุดที่ถูกแทรกลงในเครือข่ายอาจจะซ้อนทับกับขอบใดบางหนึ่งของรูปสามเหลี่ยมดังรูปที่ 4.20 [10] ได้เสนอวิธีจัดการกับปัญหานี้เป็นกรณียกเว้นโดยการลบขอบที่ซ้อนทับกันนั้นทิ้งเสีย และสร้างขอบขึ้นมาใหม่จำนวน 4 ขอบเพื่อเชื่อมต่อจุดใหม่นี้กับมุมทั้ง 4 ของรูปสี่เหลี่ยมด้านไม่เท่า

สำหรับอัลกอริทึม Incremental เมื่อสามเหลี่ยมเดิมถูกแบ่งออกเป็นสามเหลี่ยมใหม่ 3 รูป ปรากฏว่ามุมยอดของสามเหลี่ยมทั้ง 3 นี้มีตำแหน่งซ้อนทับอยู่บนขอบของสามเหลี่ยมเดิม หนึ่งในสามเหลี่ยมใหม่จะมีพื้นที่เท่ากับศูนย์ ดังนั้นจึงเรียกสามเหลี่ยมชนิดนี้ว่าเป็นสามเหลี่ยมศูนย์ (Zero

triangle) จากรูปที่ 4.21 แสดงส่วนประกอบบางส่วนภายในโครงข่ายสามเหลี่ยม เมื่อ E มีตำแหน่งตรงกับ $\overline{AB}$ ทำให้ $\alpha = \pi$ และ $\alpha + \beta = \pi$ ดังนั้น $\overline{AB}$ จึงต้องถูกสลับขอบให้เป็น $\overline{ED}$ ซึ่งเป็นการแก้ปัญหาโดยปริยาย โดยที่ไม่ต้องมีการลบขอบเดิมทิ้งหรือสร้างขอบใหม่ขึ้นมาให้เสียเวลาแต่อย่างใด

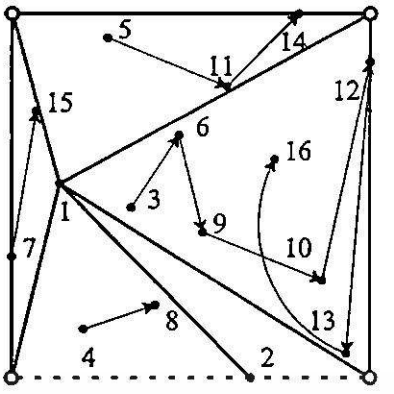


รูปที่ 4.20 การจัดการกับสามเหลี่ยมศูนย์โดยการลบและสร้างขอบใหม่



รูปที่ 4.21 การสลับขอบเมื่อเกิดสามเหลี่ยมศูนย์

ด้วยเหตุนี้จึงไม่มีสามเหลี่ยมศูนย์ปรากฏอยู่ภายในโครงข่ายสามเหลี่ยมเลยระหว่างการดำเนินการปรับโครงสร้างด้วยอัลกอริทึม Incremental อย่างไรก็ตามสามเหลี่ยมศูนย์อาจมีปรากฏอยู่ในบริเวณอาณาเขตของโครงข่ายดังรูปที่ 4.22 (แสดงจุดหมายเลข 2 ถูกแทรกเข้าสู่โครงข่ายหลังจากการสลับขอบในรูปที่ 4.11) โดยที่รัศมีของวงกลมทดสอบของสามเหลี่ยมศูนย์มีค่าเป็นอนันต์ แต่โครงข่ายสามเหลี่ยมยังมีคุณสมบัติเป็นดีลอนเนย์เพราะภายนอกของโครงข่ายไม่มีจุดอื่นใดบรรจุอยู่ภายในวงกลมทดสอบ สามเหลี่ยมศูนย์ซึ่งมีตำแหน่งอยู่ที่อาณาเขตของโครงข่ายจะมีประโยชน์ในขั้นตอนของการผนวก (Merging) สองโครงข่ายสามเหลี่ยมดีลอนเนย์เข้าหากัน (อธิบายในหัวข้อ 5.5) โดยที่ไม่ต้องมีการเพิ่มหรือลบขอบเข้าออกจากโครงข่ายแต่อย่างใด



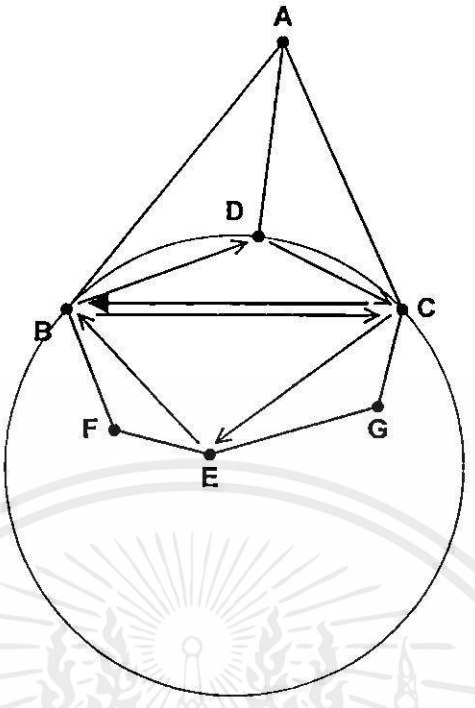
รูปที่ 4.22 สามเหลี่ยมศูนย์ที่อาณาเขตของโครงข่ายสามเหลี่ยม

4.8 การปรับโครงสร้างแบบรีเคอร์ซีฟ

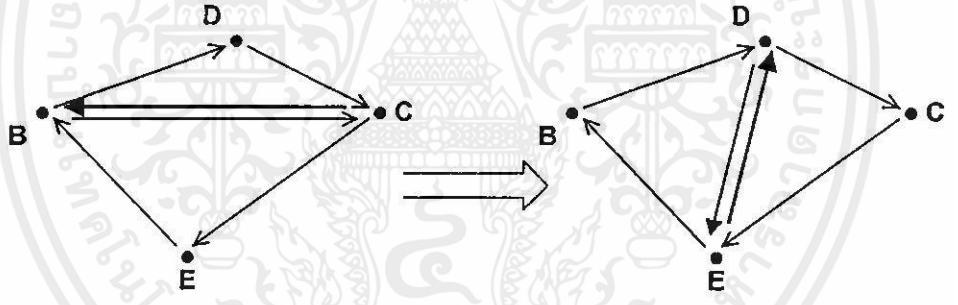
ในหัวข้อที่ 3.3 ได้แสดงผลกระทบที่มีต่อขอบต่างๆเมื่อมีการแทรกจุดใหม่เข้าสู่โครงข่ายโดยที่สี่เหลี่ยมด้านไม่เท่าทุกรูปซึ่งมีจุดแทรกใหม่เป็นมุมยอดต้องได้รับการทดสอบผลรวมของมุมภายในที่เส้นทะแยงมุมชี้อยู่ ดังรูปที่ 4.23 แสดง $\square DCEB$ ซึ่งมี D หรือจุดแทรกใหม่เป็นมุมยอด และมี $\overline{CB}$ เป็นเส้นทะแยงมุม ถ้าเส้นทะแยงมุมของสี่เหลี่ยมด้านไม่เท่าได้รับการสลับขอบ (ดังรูปที่ 4.24) สี่เหลี่ยมด้านไม่เท่าอีก 2 รูปซึ่งเกิดจากการสลับขอบนี้ (ดังรูปที่ 4.25) ต้องได้รับการทดสอบผลรวมของมุมภายในตามไปด้วยเสมอ ปัญหานี้สามารถได้รับการจัดการอย่างกระชับและถูกต้องโดยใช้หลักการประมวลผลแบบเรียกตัวเองกลับหรือแบบรีเคอร์ซีฟ (Recursive procedure) ดังอัลกอริทึมที่ 4.6 ในรูปที่ 4.26 แสดงผลลัพธ์หลังจากการปรับโครงสร้างเสร็จสิ้น โดยในวงกลมทดสอบไม่มีจุดอื่นใดบรรจุอยู่ภายในเลย เปรียบเทียบกับรูปที่ 4.23 ซึ่งมีจุด F, E และ G บรรจุอยู่ภายในวงกลมทดสอบก่อนการปรับโครงสร้าง

อัลกอริทึมที่ 4.6 Edge Reorganization

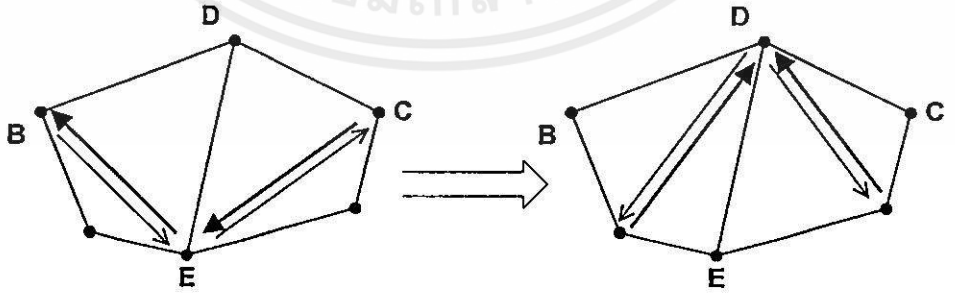
```
algorithm reorganize(diagonal)
  if angleSum(diagonal) < π then
    edge1 = diagonal.twin.next;
    edge2 = diagonal.twin.next.next;
    swapEdge(diagonal);
    reorganize(edge1);
    reorganize(edge2);
  endif
end
```



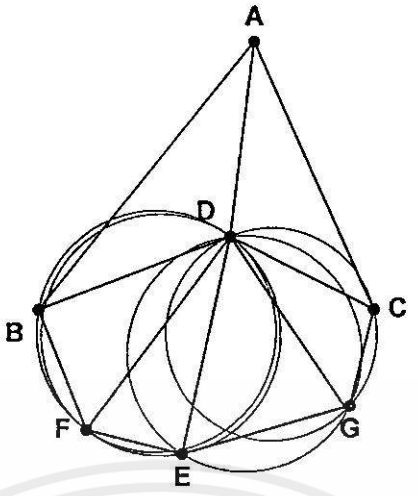
รูปที่ 4.23 $\square BDCE$ ต้องได้รับการสลับขอบ โดยมี $\overline{CB}$ เป็นเส้นทะแยงมุม



รูปที่ 4.24 การสลับขอบจาก $\overline{CB}$ เป็น $\overline{ED}$



รูปที่ 4.25 $\overline{EB}$ และ $\overline{CE}$ ได้รับการสลับขอบไปเป็น $\overline{FD}$ และ $\overline{GD}$ ตามลำดับ



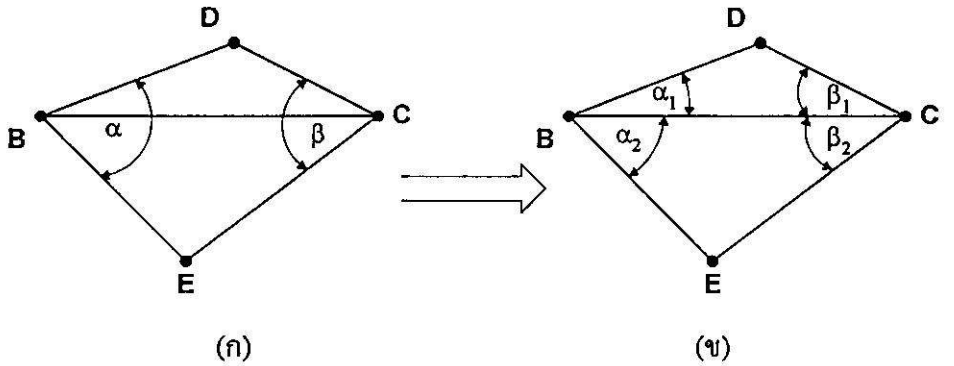
รูปที่ 4.26 ผลลัพธ์ของสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์หลังจากการสลับขอบเป็นที่เรียบร้อย

ฟังก์ชันแบบรีเดอร์สีฟโดยปกติมักมีขนาดสั้นและง่ายต่อการจัดการ เมื่อเปรียบเทียบกับฟังก์ชันแบบไม่เป็นรีเดอร์สีฟกับปัญหาชนิดเดียวกัน แต่อย่างไรก็ตามการใช้พื้นที่หน่วยความจำของฟังก์ชันแบบรีเดอร์สีฟมักสร้างปัญหาตามมา ดังนั้นการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์โดยใช้วิธีการ Incremental จึงได้รับการทดสอบทั้งแบบรีเดอร์สีฟและไม่เป็นรีเดอร์สีฟ จากผลการทดลองพบว่าความเร็วในการปฏิบัติงานแทบจะไม่แตกต่างกัน ซึ่งเป็นเพราะการแทรกจุดหนึ่งครั้งจะเกิดการสลับขอบเป็นจำนวน 3 ขอบโดยเฉลี่ย ดังนั้นปัญหาการเกิดสแต็คเกินล้น (Stack overflow) จึงไม่เกิดขึ้น

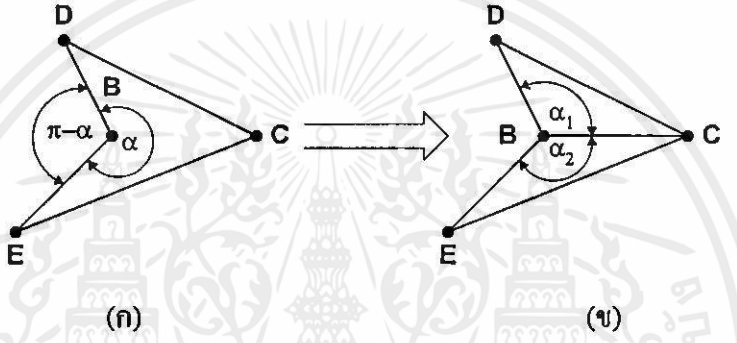
4.8.1 วิธีคำนวณผลรวมของมุมภายใน

จากอัลกอริทึมที่ 4.6 ฟังก์ชัน angleSum(diagonal) ทำหน้าที่คำนวณผลรวมของมุมภายใน ($\alpha + \beta$) ดังรูปที่ 4.27(ก) โดยที่ α คือมุมที่เกิดจาก $\overline{BD}$ และ $\overline{BE}$ และ β คือมุมที่เกิดจาก $\overline{CD}$ และ $\overline{CE}$ แต่ในทางปฏิบัติการคำนวณ α และ β จะถูกแยกออกเป็น 2 มุมย่อยดังรูปที่ 4.27(ข) เมื่อ $\alpha = \alpha_1 + \alpha_2$ และ $\beta = \beta_1 + \beta_2$ เนื่องจากในกรณีที่ $\square DCEB$ มีลักษณะไม่เป็นคอนเว็กซ์ทำให้ $\alpha > \pi$ ดังรูปที่ 4.28(ก) ผลลัพธ์ที่ได้จากการคำนวณมุมที่ $\overline{BD}$ กระทำกับ $\overline{BE}$ จะมีค่าเป็นเท่ากับ $2\pi - \alpha$ แทนที่จะเป็นเพียงแค่ α อย่างที่ต้องการ ดังนั้นเพื่อป้องกันการเกิดกรณีผิดพลาดเช่นนี้ขึ้น α จึงต้องถูกแบ่งออกเป็น α_1 และ α_2 เมื่อ α_1 คือมุมที่เกิดจาก $\overline{BD}$ และ $\overline{BC}$ และ α_2 คือมุมที่เกิดจาก $\overline{BC}$ และ $\overline{BE}$ ดังรูปที่ 4.28(ข) และ β ก็ต้องถูกปฏิบัติอย่าง α เช่นกัน การหาค่ามุม (θ) ระหว่างเส้นตรง 2 เส้น (แทนด้วยเส้นตรง a และ b) คำนวณได้จากสมการที่ (4.3) [12]

$$\cos\theta = \frac{a \cdot b}{|a||b|} \tag{4.3}$$



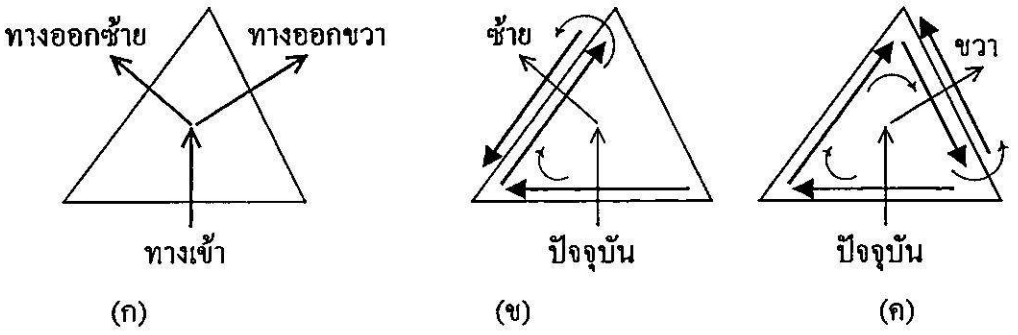
รูปที่ 4.27 ในทางปฏิบัติต้องแบ่งมุม α และ β ออกเป็น 2 มุมย่อย



รูปที่ 4.28 กรณีผิดพลาดเมื่อ $\alpha > \pi$ ผลลัพธ์ของมุมที่ได้เท่ากับ $2\pi - \alpha$

4.9 วิธีการสำรวจสามเหลี่ยมภายในโครงข่าย

การสำรวจสามเหลี่ยมภายในโครงข่าย (Traversal of the TIN) เป็นขั้นตอนที่สำคัญหลังจากที่สร้างโครงข่ายสามเหลี่ยมเสร็จ หรือระหว่างการสร้างโครงสร้างข่ายเพื่อตรวจสอบความก้าวหน้าของรูปสามเหลี่ยมในโครงข่าย การตรวจเยี่ยมจากสามเหลี่ยมรูปหนึ่งไปยังสามเหลี่ยมซึ่งอยู่ข้างเคียงกระทำได้ดังรูปที่ 4.29(ก) เมื่อด้านใดด้านหนึ่งของสามเหลี่ยมถูกกำหนดให้เป็นทางเข้าและด้านที่เหลือเป็นทางออกซ้ายและขวา การตรวจเยี่ยมถูกกำหนดให้เริ่มต้นจากขอบที่อยู่ด้านล่างสุด และให้ความสำคัญต่อการตรวจเยี่ยมสามเหลี่ยมทางด้านซ้ายก่อนสามเหลี่ยมทางด้านขวา เมื่อตรวจเยี่ยมออกนอกบริเวณโครงข่ายให้วกกลับสู่โครงข่าย ณ สามเหลี่ยมเดิม แล้วใช้ทางออกอีกทางหนึ่งในการตรวจเยี่ยมต่อไป สามเหลี่ยมใดที่ถูกตรวจเยี่ยมแล้วจะไม่ถูกตรวจเยี่ยมซ้ำ จนกระทั่งสามเหลี่ยมทุกรูปในโครงข่ายถูกตรวจเยี่ยมจนครบดังรูปที่ 4.30 (เส้นประแสดงการตรวจเยี่ยมออกนอกโครงข่าย) สังเกตได้ว่าสามเหลี่ยมหนึ่งรูปจะถูกผ่านเข้าและออกรวมกันเป็นจำนวน 6 ครั้งโดยการข้ามผ่านขอบหรือด้านของสามเหลี่ยม ขอบหนึ่งขอบจึงถูกข้ามเข้า 1 ครั้งและข้ามออกอีก 1 ครั้งรวมเป็น 2 ครั้งต่อหนึ่งขอบ เมื่อข้ามเข้าจากขอบทางด้านใดก็จะข้ามออกจากขอบทางด้านนั้นเสมอ ดังนั้นขอบแรกที่ถูกข้ามผ่านคือขอบสุดท้ายที่ถูกข้ามออกเมื่อสิ้นสุดการทำงาน



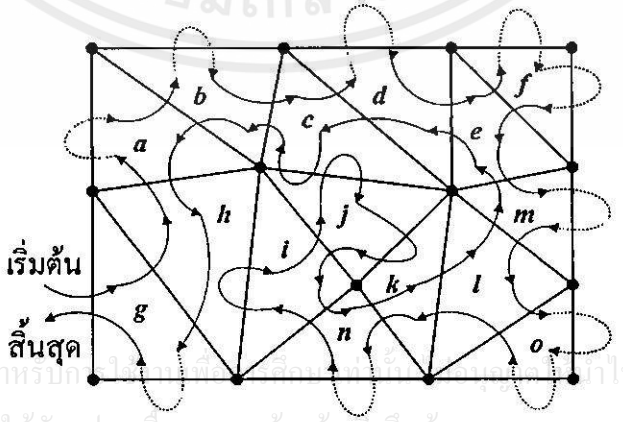
รูปที่ 4.29 ทิศทางของการเข้าและออกจากรูปสามเหลี่ยม

จากวิธีการข้างต้นเมื่อมาประยุกต์ใช้กับ TwinEdge การเคลื่อนย้ายจากขอบปัจจุบันไปสู่ขอบของสามเหลี่ยมซ้ายมือและขวามือ ดังรูปที่ 4.29(ข) และ (ค) ทำได้โดยการถ่ายทอดค่าตำแหน่งของพอยเตอร์ดังนี้

```
left_edge = current.next.twin;
right_edge = current.next.next.twin;
```

4.9.1 แบบรีเดอร์ซีฟ

อัลกอริทึมที่ 4.7 แสดงขั้นตอนการตรวจเยี่ยมโครงข่ายสามเหลี่ยมโดยใช้วิธีการแบบรีเดอร์ซีฟ เมื่อ currentEdge คือขอบปัจจุบัน, triangleList คือลำดับของสามเหลี่ยมทั้งหมดที่ได้จากการตรวจเยี่ยม เมื่อ currentEdge มีค่าเป็น null แสดงว่าบริเวณนั้นอยู่นอกโครงข่าย สามเหลี่ยมที่ถูกตรวจเยี่ยมแล้วจะถูกกำหนดค่าของ info.visit ให้เป็นจริง เมื่อขอบทั้ง 3 ขอบที่ประกอบกันขึ้นเป็นรูปสามเหลี่ยมจะชี้ไปที่ข้อมูล info อันเดียวกัน (อธิบายในหัวข้อที่ 4.4)



รูปที่ 4.30 เส้นทางการสำรวจสามเหลี่ยมทุกรูปในโครงข่ายโดยใช้วิธีการแบบรีเดอร์ซีฟ

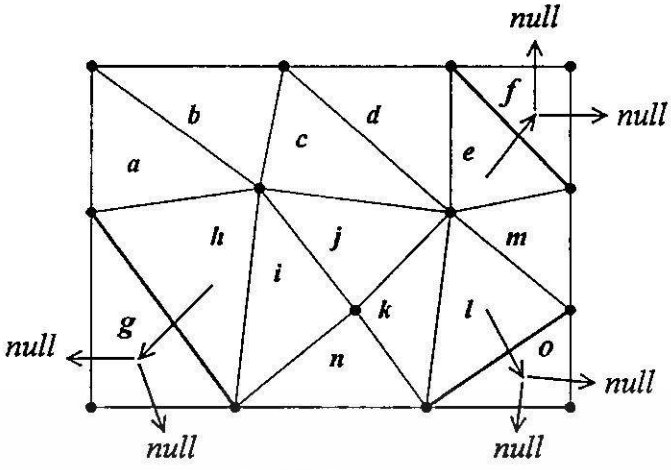
อัลกอริทึมที่ 4.7 Recursive traversal of a TIN

```
algorithm traverseTIN(currentEdge, TriangleList)
  if currentEdge ≠ null then
    if currentEdge.info.visit == false then
      currentEdge.info.visit = true;
      traverseTIN(currentEdge.next.twin, triangleList);
      triangleList.addTriangle(currentEdge.point,
                               currentEdge.next.point,
                               currentEdge.next.next.point);
      traverseTIN(currentEdge.next.next.twin, triangleList);
    endif
  endif
end
```

จากรูปที่ 4.30 สามเหลี่ยมจำนวน 15 รูปในโครงข่ายได้ถูกตั้งชื่อตามลำดับตัวอักษร *a* ถึง *o* ขอบเริ่มต้นคือขอบล่างซ้ายของสามเหลี่ยม g หลังจากการสำรวจโครงข่ายสิ้นสุดลง ลำดับของรูปสามเหลี่ยมที่ได้คือ *g h a b c d e f m l o k n i* และ *j* ตามหลักการแล้วเราสามารถเริ่มต้นสำรวจโครงข่ายจากขอบใดๆก็ได้ แต่ในทางปฏิบัติมีข้อยกเว้นสำหรับ TwinEdge บางตำแหน่งในโครงข่ายซึ่งไม่เหมาะที่จะเป็นขอบเริ่มต้น จากรูปที่ 4.31 แสดงรูปสามเหลี่ยม 3 รูป (*g, f* และ *o*) ซึ่งมีทางเข้าหันหลังให้กับโครงข่าย และมีทางออกทั้ง 2 ข้างหันออกสู่นอกอาณาเขต การเลือก TwinEdge เป็นขอบเริ่มต้นในลักษณะนี้การสำรวจโครงข่ายจะทำได้อย่างไม่สมบูรณ์ด้วยการใช้อัลกอริทึม traverseTIN() เพียงลำพัง ยกตัวอย่างเช่นเริ่มต้นที่ขอบหนึ่งของสามเหลี่ยม *f* (แสดงในเป็นเส้นหนาดังรูปที่ 4.31) ในทิศทางดังลูกศร ผลลัพธ์ของการสำรวจจะมีเพียงสามเหลี่ยม *f* รูปเดียวเท่านั้น ในทางกลับกันเมื่อถ่ายทอดขอบเริ่มต้นนี้ให้เป็นขอบคู่แฝด (e = e.twin;) ลำดับของสามเหลี่ยมที่ได้จะเป็น *e m l o k n i h g a b c d* และ *j* จำนวน 14 รูปซึ่งขาดสามเหลี่ยม *f* ไป 1 รูป เห็นได้ว่าขอบซึ่งไม่เหมาะที่จะเป็นขอบเริ่มต้น ทางแก้ไขของปัญหานี้คือ ขอบเริ่มต้นนี้ต้องได้รับการตรวจสอบและถ่ายทอดให้เป็นขอบใดขอบหนึ่งของ 2 ขอบที่เหลือในสามเหลี่ยมเริ่มต้นนั้น ดังการกำหนดต่อไปนี้

```
if ((e.next.twin == null) and (e.next.next.twin == null)) or
  ((e.twin.next.twin == null) and (e.twin.next.next.twin == null)) then
  e = e.next; หรือ e = e.next.next;
endif
```

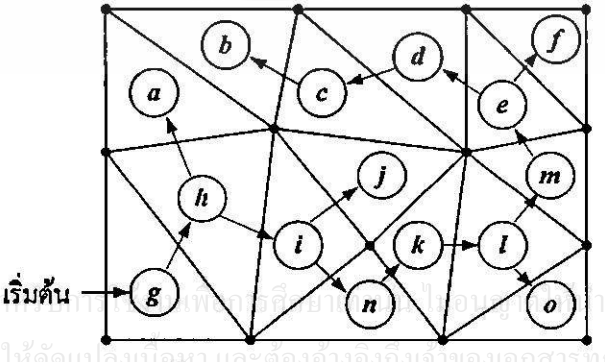

ก่อนที่จะเรียกใช้ฟังก์ชันหรืออัลกอริทึม traverseTIN(e, t) เมื่อ e คือขอบเริ่มต้น และ t คือลำดับของสามเหลี่ยมผลลัพธ์



รูปที่ 4.31 ตำแหน่งของ TwinEdge ซึ่งไม่เหมาะที่จะเป็นขอบเริ่มต้น

4.9.2 แบบไม่เป็นรีเคอร์สีฟ

การสำรวจโครงข่ายสามเหลี่ยมแบบรีเคอร์สีฟในบางครั้งจะเกิดปัญหาสแต็คเกินล้น เมื่อจำนวนของสามเหลี่ยมที่บรรจุอยู่ในโครงข่ายมีมากเกินไป ในอัลกอริทึมที่ 4.8 แสดงการสำรวจโครงข่ายโดยใช้วิธีการแบบไม่เป็นรีเคอร์สีฟ สแต็ค (Stack) สำหรับเก็บขอบซึ่งเป็นตัวแทนของสามเหลี่ยมจะถูกสร้างขึ้นและปรับปรุงตลอดเส้นทางการสำรวจ เมื่อสำรวจไปถึงสามเหลี่ยมใดๆ แล้วพบว่าสามเหลี่ยมด้านซ้ายหรือด้านขวายังไม่ถูกสำรวจขอบซึ่งแทนสามเหลี่ยมนั้นๆก็จะถูกเติม (push) ลงสู่สแต็ค และขอบซึ่งอยู่บนสุดของสแต็คก็จะถูกดึง (pull) ออกมาใช้งานเพื่อตรวจเยี่ยมอย่างต่อเนื่อง จนกระทั่งไม่มีขอบใดๆเหลืออยู่ในสแต็คแล้วจึงจบการสำรวจโครงข่าย ในรูปที่ 4.32 แสดงเส้นทางการสำรวจโครงข่ายเมื่อใช้หลักการจากอัลกอริทึมที่ 4.8 เห็นได้ว่าการสำรวจจะให้ความสำคัญกับสามเหลี่ยมด้านขวาก่อน เนื่องจากหลักการของสแต็ค (FILO : First-In-Last-Out หรือเข้าก่อนออกหลัง) เมื่อขอบของสามเหลี่ยมซ้ายถูกเติมลงสู่สแต็คก่อนแล้วจึงตามด้วยขอบของสามเหลี่ยมขวา ดังนั้นขอบของสามเหลี่ยมขวาจึงถูกเรียกใช้ก่อนจากสแต็ค



รูปที่ 4.32 เส้นทางการสำรวจสามเหลี่ยมทุกรูปในโครงข่ายโดยใช้วิธีการแบบไม่เป็นรีเคอร์ซีฟ

อัลกอริทึมที่ 4.8 Nonrecursive traversal of a TIN

```
algorithm nonRecursiveTraverseTIN(firstEdge, triangleList)
  stack.push(firstEdge);
  while(stack.isNotEmpty())
    currentEdge = stack.pop();
    currentEdge.info.visit = true;
    triangleList.addTriangle(currentEdge.point,
                             currentEdge.next.point,
                             currentEdge.next.next.point);
    if currentEdge.next.twin ≠ null then
      if currentEdge.next.twin.info.visit == false then
        stack.push(currentEdge.next.twin);
      endif
    endif
    if currentEdge.next.next.twin ≠ null then
      if currentEdge.next.next.twin.info.visit == false then
        stack.push(currentEdge.next.next.twin);
      endif
    endif
  endwhile
end
```

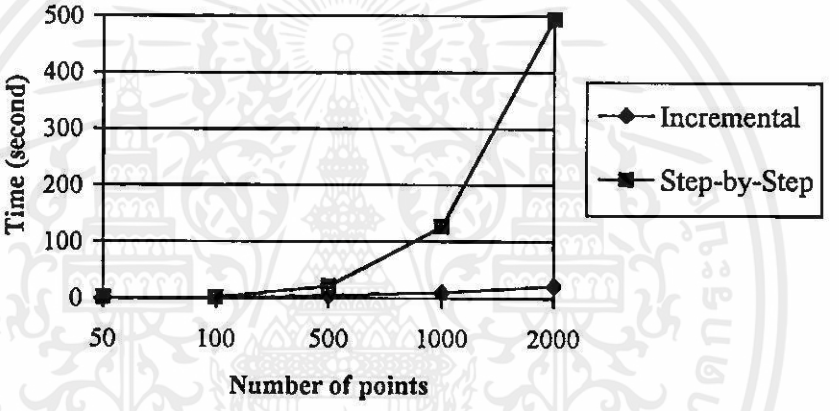
4.10 ผลการทำงานของอัลกอริทึม Incremental

การทดสอบประสิทธิภาพการทำงานของอัลกอริทึม Incremental ได้ถูกทดลองเปรียบเทียบกับอัลกอริทึม Step-by-Step อัลกอริทึม Incremental เป็นตัวแทนของวิธีการคำนวณโครงข่ายสามเหลี่ยมดีลอนเนย์แบบพลวัต และอัลกอริทึม Step-by-Step เป็นตัวแทนของวิธีการแบบสถิต การทดลองได้กระทำกับเซ็ตของจุดซึ่งได้จากการสุ่มจำนวนตั้งแต่ 50 - 2000 จุด เมื่อจุดทุกจุดต้องถูกรวมเข้าสู่โครงข่าย ประสิทธิภาพการทำงานเรื่องเวลาของทั้งสองอัลกอริทึมแสดงได้ดังตารางที่ 4.1 และกราฟในรูปที่ 4.33 จากผลลัพธ์ในตารางที่ 4.1 เห็นได้ว่าอัลกอริทึม Incremental ได้ถูกกำหนดให้มีการจัดลำดับการแทรกจุด 2 รูปแบบ คือแบบก่อนหลังและแบบสุ่ม จากผลการทดลองเห็นได้ว่าการแทรกจุดแบบสุ่มทำงานได้เร็วกว่าแบบก่อนหลัง (แต่แตกต่างกันไม่มากนัก)

เมื่อเปรียบเทียบระหว่างอัลกอริทึม Incremental และ Step-by-Step แล้ว อัลกอริทึม Incremental มีความเร็วในการคำนวณสูงกว่ามาก ยิ่งจุดมีจำนวนมากขึ้นเท่าใดยิ่งเห็นความแตกต่างอย่างมากขึ้นเท่านั้น ประสิทธิภาพของอัลกอริทึม Incremental นั้นใกล้เคียงกับ $O(n \log n)$ เนื่องจากหัวใจหลักของอัลกอริทึมในการแทรกจุด และการปรับปรุงโครงสร้างล้วนมีหลักการทำงานเป็นแบบรีเดอร์สีฟทั้งหมด สำหรับอัลกอริทึม Step-by-Step มีประสิทธิภาพการคำนวณเป็น $O(n^2)$ เนื่องจากสูญเสียเวลาส่วนมากไปในการค้นหาจุดที่สามเพื่อมาประกอบเป็นรูปสามเหลี่ยมรูปต่อไป และในการค้นหาแต่ละครั้งจุดทุกจุดต้องถูกนำมาเปรียบเทียบหาทั้งหมด

ตารางที่ 4.1 ผลการทำงานของอัลกอริทึม Incremental เมื่อเปรียบเทียบกับอัลกอริทึม Step-by-Step (เมื่อหน่วยของเวลาเป็นวินาที)

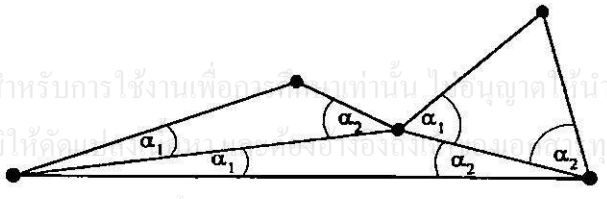
จำนวนจุด	อัลกอริทึม Incremental		อัลกอริทึม Step-by-Step
	ลำดับแบบก่อนหลัง	ลำดับแบบสุ่ม	
50	0.69	0.62	0.92
100	1.13	1.07	1.83
500	4.96	4.88	21.53
1000	10.10	10.03	126.24
2000	21.42	21.34	494.07



รูปที่ 4.33 ผลการทำงานของอัลกอริทึม Incremental เมื่อเปรียบเทียบกับอัลกอริทึม Step-by-Step

4.11 วิธีการกำจัดสามเหลี่ยมรูปลิ่มบริเวณอาณาเขตโครงข่าย

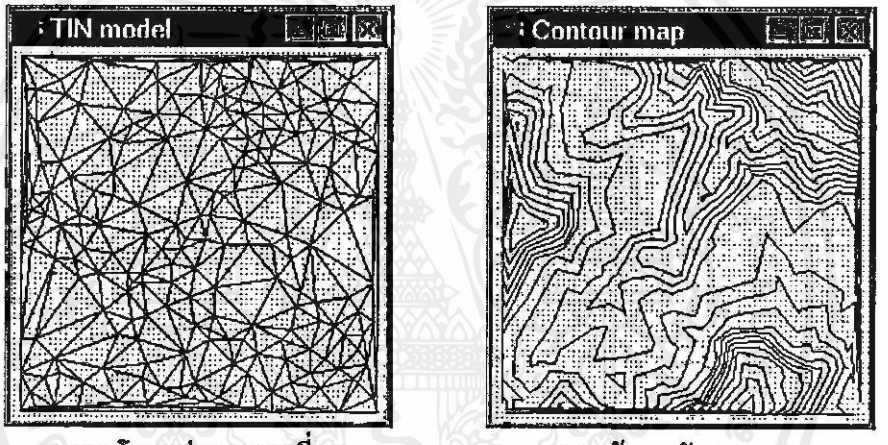
ด้วยการสร้างโครงข่ายเริ่มต้นในรูปแบบของรูปสามเหลี่ยม 2 รูปประกบกันเป็นคอนเว็กซ์ฮัลรูปสี่เหลี่ยมด้านขนาน หลังจากที่คำนวณโครงข่ายสามเหลี่ยมเสร็จก็มักเกิดสามเหลี่ยมฐานย์ (หัวข้อที่ 4.7.1) และสามเหลี่ยมรูปลิ่มที่บริเวณอาณาเขตของโครงข่ายดังรูปที่ 4.35(ก) สามเหลี่ยมเหล่านี้มักสร้างปัญหาในขั้นตอนของการสร้างภาพแบบจำลองชนิดต่างๆ เช่น แผนที่คอนทัวร์ (ดังรูปที่ 4.35(ข)) ซึ่งจะเกิดปรากฏการณ์ที่เส้นคอนทัวร์ไปกระจุกตัวอยู่ที่ขอบของแผนที่



รูปที่ 4.34 มุมที่ฐานของสามเหลี่ยมทั้งสอง (α_1 และ α_2) ที่ต้องถูกทดสอบ

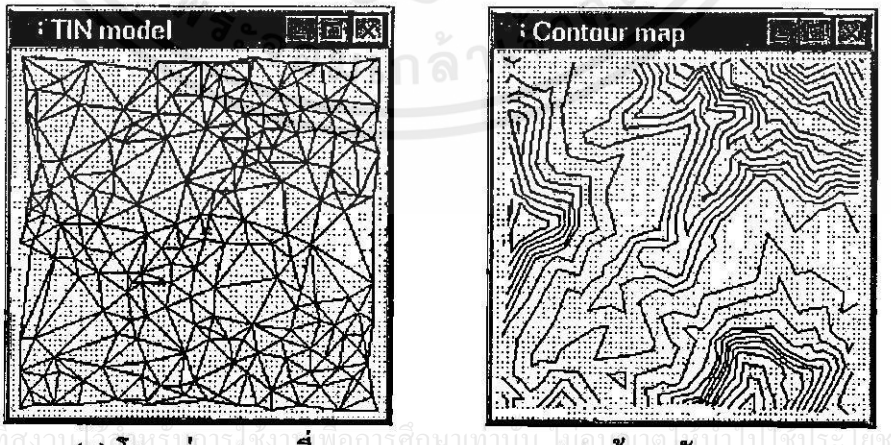
อัลกอริทึมที่ 4.9 Remove Thin Triangles

```
algorithm removeThinTriangle(edge, creaseAngle)
    org  = edge.next.next.point;
    dest = edge.point;
    p    = e.next.point;
    alpha1 = angle(p, dest, org);
    alpha2 = angle(p, org, dest);
    if (alpha1 < creaseAngle) or (alpha2 < creaseAngle) then
      leftEdge  = e.next.twin;
      rightEdge = e.next.next.twin;
      leftEdge.twin  = null;
      rightEdge.twin = null;
      removeThinTriangle(leftEdge,  creaseAngle);
      removeThinTriangle(rightEdge, creaseAngle);
    endif
end
```



(ก) โครงข่ายสามเหลี่ยม (ข) เส้นระดับความสูง

รูปที่ 4.35 ก่อนการกำจัดสามเหลี่ยมรูปลิ่มบริเวณอาณาเขต



(ก) โครงข่ายสามเหลี่ยม (ข) เส้นระดับความสูง

รูปที่ 4.36 หลังการกำจัดสามเหลี่ยมรูปลิ่มบริเวณอาณาเขต

การกำจัดสามเหลี่ยมรูปลิ่มเหล่านี้กระทำได้โดยเริ่มต้นจากขอบซึ่งอยู่นอกสุดของอาณาเขตก่อนดังรูปที่ 4.34 (แสดงด้วยเส้นหนา) แล้วทำการตรวจสอบขนาดของมุมที่ปลายทั้ง 2 สอง (α_1 และ α_2) ของสามเหลี่ยมปัจจุบัน ถ้ามุมใดมุมหนึ่งมีขนาดต่ำกว่าขนาดของมุมที่กำหนดไว้ (Crease angle) ก็ให้ลบสามเหลี่ยมนี้ทิ้ง จากนั้นก็ค้นหาสามเหลี่ยมที่อยู่ถัดไป (ซ้ายและขวา) ว่ามีขนาดของมุมที่ฐานนี้ต่ำกว่าค่าที่กำหนดไว้อีกหรือไม่ ดังสรุปในอัลกอริทึมที่ 4.9 ซึ่งเป็นการทำงานแบบรีเคอร์ซีฟ ผลจากการกำจัดสามเหลี่ยมที่ไม่ต้องการเหล่านี้ออกจากโครงข่ายสามเหลี่ยมแสดงได้ดังรูปที่ 4.36(ก) (จากการทดลองกำหนดค่าของมุมทดสอบไว้ที่ $\pi/20$ หรือ 9°) และรูปที่ 4.36(ข) แสดงแผนที่คอนทัวร์ซึ่งสอดคล้องกับโครงข่ายสามเหลี่ยมในรูปที่ 4.36(ก)

4.12 วิเคราะห์อัลกอริทึม Incremental

อัลกอริทึม Incremental มีจำนวนรอบการทำงาน (Main loop count) ทั้งหมดเท่ากับ n รอบ เมื่อ n = จำนวนจุดทั้งหมดที่ต้องถูกแทรกเข้าสู่โครงข่ายเริ่มต้น (หัวข้อที่ 4.6.1) จุดจำนวน 1202 จุดได้ถูกสุ่มขึ้นเพื่อใช้ประกอบในการวิเคราะห์การทำงานของอัลกอริทึม Incremental หลังจากที่เซ็ตจุดนี้ผ่านการคำนวณให้เป็นโครงข่ายสามเหลี่ยมจะมีโครงสร้างดังรูปที่ 4.37 ในแต่ละรอบการทำงานซึ่งหมายถึงการแทรกจุดใหม่แต่ละครั้งนั้นลำดับของขอบ (ซึ่งเป็นตัวแทนของสามเหลี่ยมซึ่งมีจุดบรรจุอยู่ภายในเพื่อรอการถูกดึงออกมาใช้ในการแทรกจุด) จะได้รับการปรับปรุงตามไปด้วยทุกครั้ง กราฟในรูปที่ 4.38 แสดงผลลัพธ์การเปลี่ยนแปลงจำนวนของขอบ เห็นได้ว่าในช่วงแรกนั้นจำนวนของขอบจะเพิ่มขึ้น เพราะยังมีจุดเหลือในสามเหลี่ยมต่างๆอยู่จำนวนมาก และจำนวนของขอบจะลดลงในช่วงหลัง เมื่อมีจุดที่เหลืออยู่ในสามเหลี่ยมน้อยลง

4.12.1 ผลกระทบของลำดับการแทรกจุด

จากอัลกอริทึมที่ 4.1 และ 4.2 แสดงวิธีการจัดลำดับการแทรกจุด แบบก่อนหลัง และแบบสุ่ม ซึ่งมีรอบการทำงานที่คล้ายคลึงกัน คือการดึงขอบจากลำดับขึ้นมาประมวลผล แต่ต่างกันที่อัลกอริทึมที่ 4.1 (Ordinal Succession) จะดึงขอบแรก (โดยใช้ฟังก์ชัน edgeList.getFirstEdge()) ขึ้นมาประมวลผลก่อนเสมอ ส่วนอัลกอริทึมที่ 4.2 (Randomized Succession) จะใช้การสุ่มตัวเลขเพื่อใช้เป็นดัชนีอ้างอิงขอบใดๆจากลำดับของขอบ (โดยใช้ฟังก์ชัน edgeList.getEdgeAt(r) เมื่อ r คือดัชนีที่เกิดจากการสุ่ม) ผลลัพธ์เวลาในการคำนวณของทั้งสองอัลกอริทึมเมื่อเปรียบเทียบกันแล้วใกล้เคียงกันมาก ดังตารางที่ 4.1 (เมื่อเวลาที่ใช้ต่างกันเพียงเสี้ยววินาที) ดังนั้นจึงสรุปได้ว่าลำดับของการแทรกจุดนั้นมีผลน้อยมากต่อความเร็วในการคำนวณโครงข่ายสามเหลี่ยม

4.12.2 การถ่ายเทจุดระหว่างสามเหลี่ยม

เวลาที่ใช้ในการคำนวณในแต่ละรอบจะไม่คงที่ ตัวแปรที่สำคัญก็คือจำนวนของการถ่ายเทจุดระหว่างสามเหลี่ยม จำนวนการถ่ายเทจุดจะเกิดขึ้นควบคู่ไปกับจำนวนการเปรียบเทียบ (จากสมการที่ (4.2)) เพื่อตัดสินว่าจุดนั้นจะถูกถ่ายเทไปสู่สามเหลี่ยมด้านใดของเส้นทะแยงมุม เราจะแทนจำนวนการถ่ายเทจุดด้วยฟังก์ชัน $T(n)$ และจำนวนการเปรียบเทียบด้วยฟังก์ชัน $C(n)$ กรณีการถ่ายเทจุดแบ่งได้เป็น 3 กรณีดังนี้คือ

4.12.2.1 หลังจากสร้างโครงข่ายเริ่มต้น

จำนวนการถ่ายเทจุดจะเท่ากับจำนวนจุดทั้งหมด และ จำนวนการเปรียบเทียบจะเท่ากับจำนวนจุดทั้งหมด ดังสมการที่ (4.4) และ (4.5)

$$T_1(n) = n \tag{4.4}$$

$$C_1(n) = n \tag{4.5}$$

4.12.2.2 หลังจากการแทรกจุด

หลังจากการแทรกจุดทุกครั้ง จำนวนการถ่ายเทจุดจะเท่ากับจำนวนจุดภายในสามเหลี่ยมดังสมการที่ (4.6) จำนวนการเปรียบเทียบจะเท่ากับจำนวนจุดภายในสามเหลี่ยมคูณด้วย 2 ดังรูปที่ 4.17 (ก) ในกรณีที่จุดนั้นอยู่ภายใน ΔADC และ จำนวนการเปรียบเทียบจะเท่ากับจำนวนจุดภายในสามเหลี่ยมคูณด้วย 3 ดังรูปที่ 4.17 (ข) ในกรณีที่จุดนั้นอยู่ภายใน ΔABD หรือ ΔDBC ดังสมการที่ (4.7) และ (4.8)

$$T_2(t) = t \tag{4.6}$$

$$C_2(t) = 2t \tag{4.7}$$

$$C_2(t) = 3t \tag{4.8}$$

เมื่อ t = จำนวนจุดภายในสามเหลี่ยม

4.12.2.3 หลังจากการสลับขอบ

จำนวนการถ่ายเทจุดจะเท่ากับจำนวนจุดทั้งหมดในสี่เหลี่ยมด้านไม่เท่า และ จำนวนการเปรียบเทียบจะเท่ากับจำนวนจุดทั้งหมดในสี่เหลี่ยมด้านไม่เท่า ดังสมการที่ (4.7) และ (4.8)

$$T_3(q) = q \tag{4.9}$$

$$C_3(q) = q \tag{4.10}$$

เมื่อ q = จำนวนจุดภายในสี่เหลี่ยมด้านไม่เท่า

จำนวนการถ่ายเทจุด และจำนวนการเปรียบเทียบในกรณีหลังจากการสร้างโครงข่ายเริ่มต้นจะคงที่และเกิดขึ้นพียงครั้งเดียว จำนวนการถ่ายเทจุดและจำนวนการเปรียบเทียบในกรณีหลังจากการแทรกจุด และหลังจากการสลับขอบนั้นจะเปลี่ยนแปลงตลอดรอบการทำงานของการแทรกจุดใหม่ ในช่วงแรกนั้นการถ่ายเทจุดระหว่างสามเหลี่ยมนั้นจะมีสูงมาก และจะค่อยๆลดลงเมื่อจำนวนจุดที่เหลือในสามเหลี่ยมลดลง ในลักษณะเอ็กซ์โปรเนนเชียล (Exponential) ดังกราฟในรูปที่ 4.40 ซึ่งแสดงจำนวนการถ่ายเทจุดภายในสามเหลี่ยมหลังจากการแทรกจุดใหม่แต่ละครั้ง และกราฟในรูปที่ 4.41 ซึ่งแสดงจำนวนการถ่ายเทจุดภายในสี่เหลี่ยมด้านไม่เท่าเมื่อมีการสลับขอบหลังจากการแทรกจุดใหม่ จำนวนการถ่ายเทจุดตลอดรอบการทำงานจะเป็นดังสมการที่ (4.11) และ (4.12) และจำนวนการเปรียบเทียบตลอดรอบการทำงานจะเป็นดังกับสมการที่ (4.13) และ (4.14)

$$T(n) = T_1 + n\,T_2 + nT_3 \tag{4.11}$$

$$T(n) = n + nT(t) + nT(q) \tag{4.12}$$

$$C(n) = C_1 + n\,C_2 + nC_3 \tag{4.13}$$

$$C(n) = n + nC(3t) + nC(q) \tag{4.14}$$

4.12.3 จำนวนการสลับขอบ

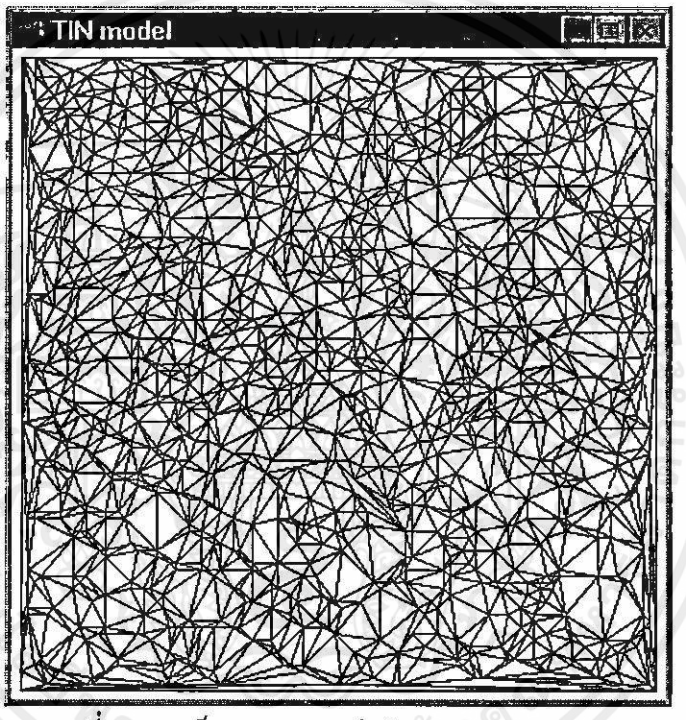
จำนวนการเปรียบเทียบก่อนการสลับขอบ โดยการทดสอบผลรอบของมุมภายในเพื่อตัดสินว่าจะให้มีการสลับขอบนั้นๆหรือไม่จะมากกว่าหรือเท่ากับ 3 ครั้ง เมื่อเมื่อฟังก์ชัน reorganize (edge) ในอัลกอริทึมที่ 4.5 จะถูกเรียก 3 ครั้งเสมอ โดยมีขอบทั้ง 3 ด้านของสามเหลี่ยมเป็นข้อมูลอินพุทเพื่อทดสอบมุมภายใน ดังนั้นจำนวนการสลับขอบทั้งหมดจึงเป็นดังสมการที่ (4.15)

$$C(n) \geq 3n \tag{4.15}$$

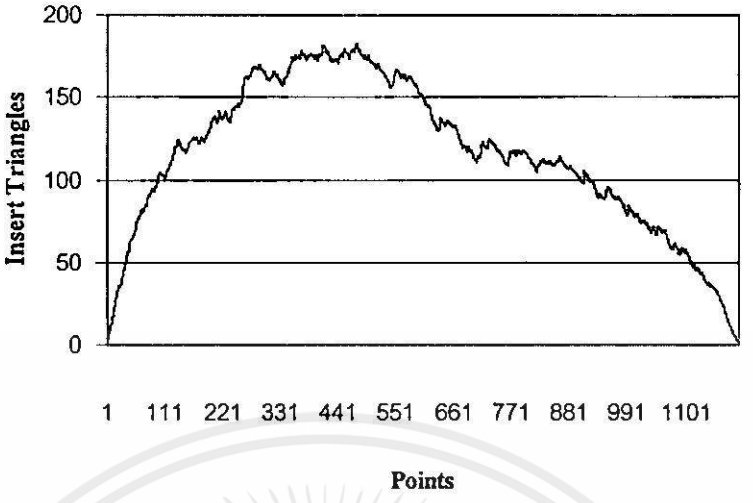
สำหรับจำนวนการสลับขอบโดยเฉลี่ยจะประมาณ 3 ครั้งดังสมการที่ (4.16) ซึ่งอ้างอิงจากกราฟในรูปที่ 4.39

$$S(n) \approx 3n \tag{4.16}$$

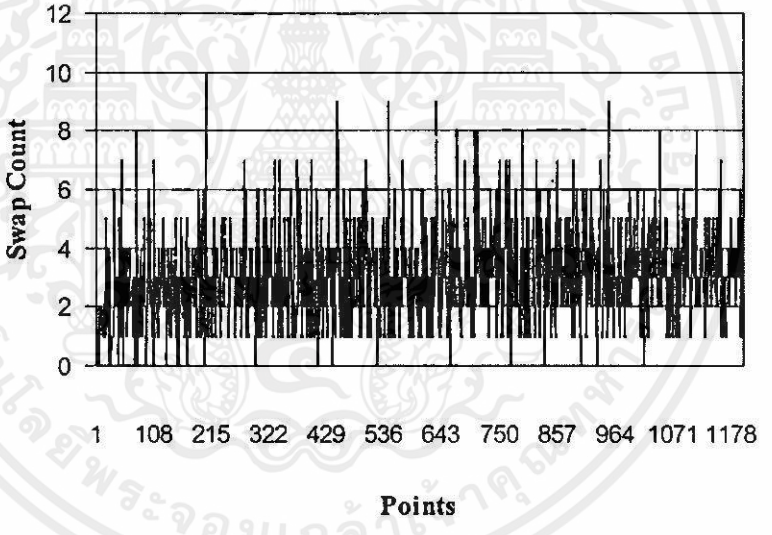
ในทางทฤษฎีเป็นการยากที่จะคำนวณจำนวนการสลับขอบเบื้องต้นออกมาได้ รวมไปถึงจำนวนของการถ่ายเทจุดในแต่ละรอบการทำงาน เนื่องจากการกระจายตัวของจุดมีความไม่แน่นอน แต่จากการทดลองได้แสดงให้เห็นแล้วว่าประสิทธิภาพของอัลกอริทึม Incremental ซึ่งนำเสนอในวิทยานิพนธ์นี้เป็น $O(n \log n)$ เมื่อมีรอบการทำงานเท่ากับ n รอบ ในแต่ละรอบมีความซับซ้อนในการทำงานโดยรวมเท่ากับ $O(\log n)$ เปรียบเทียบกับอัลกอริทึม Incremental ใน [10] ซึ่งมีประสิทธิภาพเป็น $O(n^2)$ เนื่องจากมีรอบการทำงานที่ซ้อนกันอยู่ 2 รอบ รอบการทำงานนอก (Outer loop) คือรอบของการแทรกจุดเท่ากับ n ส่วนรอบการทำงานภายใน (Inner loop) คือรอบของการค้นหาสามเหลี่ยมซึ่งจุดแทรกใหม่นั้นบรรจุอยู่ภายในโดยมี worse case เป็น $O(n)$



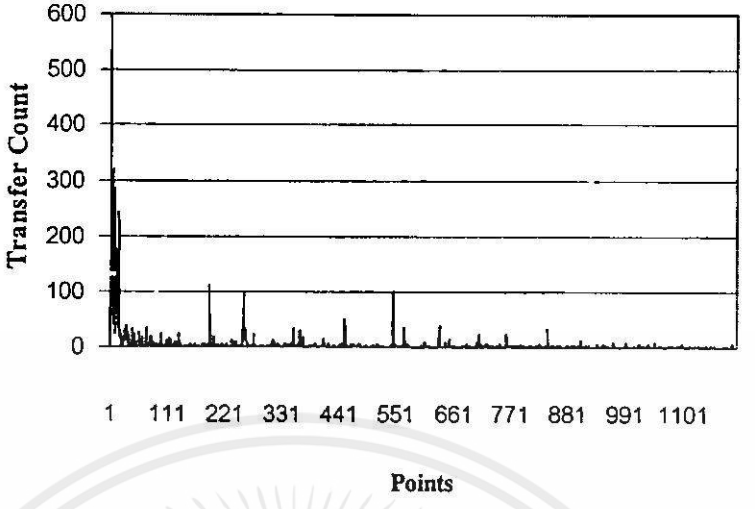
รูปที่ 4.37 โครงข่ายสามเหลี่ยมของเซ็ตจุดจากการสุ่มจำนวน 1202 จุด



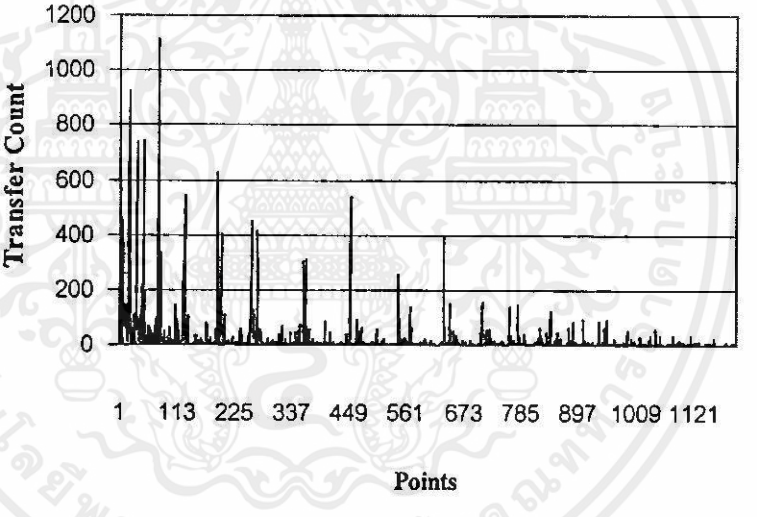
รูปที่ 4.38 การเปลี่ยนแปลงของจำนวนสามเหลี่ยมซึ่งมีจุดบรรจุอยู่ภายใน



รูปที่ 4.39 จำนวนการสลับขอบที่เกิดขึ้นหลังจากการแทรกจุดใหม่แต่ละครั้ง ค่าเฉลี่ย = 3.018303 ครั้ง



รูปที่ 4.40 จำนวนการถ่ายเทจุดภายในสามเหลี่ยมหลังจากการแทรกจุดใหม่แต่ละครั้ง



รูปที่ 4.41 จำนวนการถ่ายเทจุดภายในสี่เหลี่ยมด้านไม่เท่าเมื่อมีการสลับขอบหลังจากการแทรกจุดใหม่

บทที่ 5
การจำลองภูมิประเทศ

ข้อดีของวิธีการสร้างโครงข่ายสามเหลี่ยมแบบไดนามิคคือสามารถคัดเลือกจุดก่อนที่จะถูกรวมเข้ากับโครงข่ายสามเหลี่ยม จุดที่ถูกเลือกจะเป็นตัวแทนของจุดทั้งหมดจากแบบจำลองความสูง ดังนั้นการแปลงโครงสร้างข้อมูลจากกริดไปเป็นโครงข่ายสามเหลี่ยมจึงเป็นการประมาณเค้าโครงของภูมิประเทศ ในบทนี้จะอธิบายวิธีการคัดเลือกจุดโดยใช้ระยะทางกำหนด จากนั้นจะอธิบายหลักการลบจุดออกจากโครงข่ายสามเหลี่ยม วิธีการเชื่อมต่อสองโครงข่ายสามเหลี่ยมเข้าหากันเพื่อปรับปรุงความเร็วในการคำนวณโครงข่ายสามเหลี่ยมสำหรับข้อมูลความสูงขนาดใหญ่ โดยการแบ่งข้อมูลความสูงออกเป็นบล็อคๆแล้วทำการคำนวณสามเหลี่ยมแยกจากกัน จากนั้นนำโครงข่ายสามเหลี่ยมทั้งหมดมาเชื่อมต่อในภายหลัง ในตอนท้ายจะแสดงตัวอย่างของการนำเสนอแบบจำลองภูมิประเทศโดยใช้โครงข่ายสามเหลี่ยมเป็นโครงสร้างข้อมูลพื้นฐานในรูปแบบต่างๆ

5.1 การคัดเลือกจุด

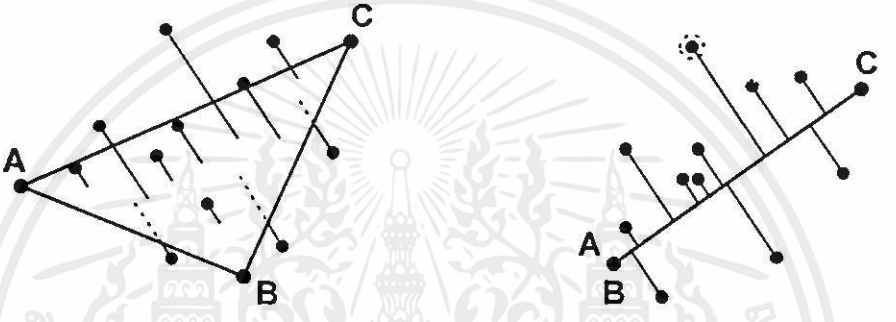
ประโยชน์หลักอันหนึ่งของการใช้ TIN เป็นโครงสร้างข้อมูลสำหรับจัดการกับเซ็ตของจุดคือความสามารถในการคัดเลือกจุด เมื่อจุดต่างๆซึ่งมีตำแหน่งอยู่ภายในสามเหลี่ยมได้รับการจัดเก็บเป็นเสมือนสมบัติภายในของสามเหลี่ยมแต่ละรูปทำให้ไม่ต้องเสียเวลาเพิ่มเติมในการคำนวณหาว่าจุดนั้นๆอยู่ภายในสามเหลี่ยมใด รูปสามเหลี่ยมประกอบขึ้นจากจุด 3 จุดแทนการอธิบายระนาบใดๆในอาณาเขตพิจารณา

เมื่อจุดใดๆมีตำแหน่งวางราบอยู่บนระนาบซึ่งอธิบายด้วยรูปสามเหลี่ยม จุดเหล่านี้จะไม่มีส่วนร่วมกับแบบจำลอง ยกตัวอย่างเช่นเมื่อพื้นผิวของทะเลถูกแทนด้วยแบบจำลองชนิดกริด จุดต่างๆซึ่งมีพิกัดความสูงที่ใกล้เคียงและซ้ำซ้อนกันจำนวนมากจะถูกจัดเก็บอย่างสิ้นเปลือง แต่ในแบบจำลองความสูงแบบ TIN พื้นผิวราบจะถูกแทนด้วยระนาบสามเหลี่ยมเพียงไม่กี่รูปเท่านั้น

ด้วยหลักการนี้จึงสามารถลดความซ้ำซ้อนของข้อมูลจุดซึ่งแทนแบบจำลองนั้นๆได้ ดังนั้นจะมีจุดบางจุดที่ไม่ได้ถูกรวมเข้ากับโครงข่ายสามเหลี่ยม ถ้าจุดนั้นมีตำแหน่งอยู่ใกล้กับหนึ่งในสามเหลี่ยมของโครงข่ายเกินค่ากำหนด ผลลัพธ์คือความเป็นไปได้ที่จะสามารถกำหนดความละเอียดของพื้นผิวแบบจำลองตามระดับความแม่นยำต่างๆ ความแม่นยำถูกกำหนดขึ้นจากค่าเทร์สโฮลด์ (Threshold) เริ่มต้นสำหรับการคัดเลือกจุดในการแทรกเข้าสู่โครงข่าย

เมื่อแบบจำลองเค้าโครงของพื้นผิวเป็นสิ่งที่ต้องการคำนวณหา จุดสำคัญซึ่งได้รับการคัดเลือกก็จะเป็นตัวแทนอธิบายรูปร่างของพื้นผิว ระหว่างขั้นตอนของการแทรกจุดในอัลกอริทึม Incremental จุดแยก (Split point) ของสามเหลี่ยมจะถูกเลือกออกมาจากสับเซ็ต (subset) ของจุดซึ่ง

ถูกปิดล้อมด้วยสามเหลี่ยมปัจจุบัน จุดซึ่งมีระยะห่างมากที่สุดในแต่ละรูปสามเหลี่ยมคือจุดซึ่งมีส่วนร่วมมากที่สุดสำหรับแบบจำลองและจุดนี้ก็จะถูกเลือกให้เป็นจุดแยกใหม่ (ดังรูปที่ 5.1) หลังจากที่สามเหลี่ยมถูกแบ่งพื้นที่และขอบโดยรอบถูกปรับปรุง สามเหลี่ยมทุกรูปซึ่งได้รับผลกระทบจากการปรับปรุงนี้ก็จะมีจุดแยกใหม่เป็นของตัวเอง เมื่อจุดที่เหลือมีระยะห่างไปสู่สามเหลี่ยมปัจจุบันต่ำกว่าค่าเทร์สโฮลด์ (λ) ที่กำหนดไว้ก็จะไม่มีการแทรกจุดเข้าสู่สามเหลี่ยมนั้นอีก การสร้างโครงข่ายสามเหลี่ยมจะสิ้นสุดลงเมื่อจุดทุกจุดที่เหลือซึ่งไม่ได้ถูกรวมเข้าสู่โครงข่ายมีระยะห่างไปสู่สามเหลี่ยมซึ่งปิดล้อมตัวมันเองต่ำกว่า λ

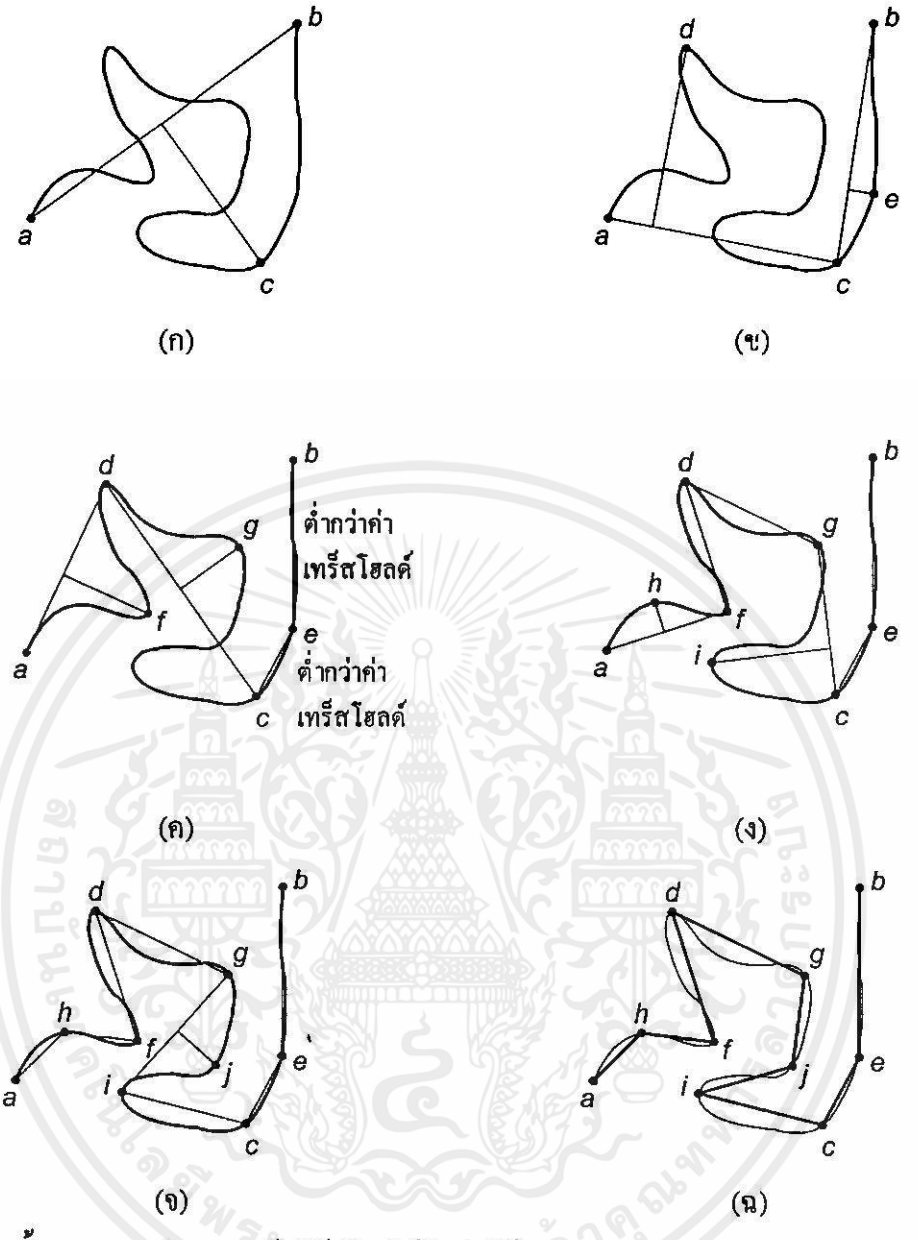


(ก) ภาพด้านบน (ข) ภาพตัดด้านข้าง

รูปที่ 5.1 จุดซึ่งมีระยะห่างมากที่สุดจากระนาบสามเหลี่ยมจะกลายเป็นจุดแยกสำหรับการแทรกจุดใหม่เข้าสู่โครงข่ายต่อไป

ลำดับของสามเหลี่ยมซึ่งกำลังจะถูกแทรกสามารถถูกเลือกเป็นแบบสุ่ม ซึ่งเป็นหลักการที่ธรรมดาแต่มีประสิทธิภาพ และจากการทดลองก็ได้ให้ผลลัพธ์ที่ดี อย่างไรก็ตามแบบจำลองพื้นผิวจะมีโครงสร้างของโครงข่ายไม่เหมือนกันทั้งหมด เมื่อเซ็ตข้อมูลถูกประมวลผลเปรียบเทียบกัน 2 ครั้งซึ่งใช้ลำดับของการแทรกจุดที่ไม่เหมือนกัน (เนื่องจากการสุ่ม) แต่ความแตกต่างที่เกิดขึ้นนี้จะมีผลน้อยมากและความถูกต้องของแบบจำลองพื้นผิวก็ยังคงเดิม มีความเป็นไปได้ที่จะหลีกเลี่ยงปัญหาที่เกิดขึ้นนี้โดยการเลือกสามเหลี่ยมที่จะถูกแทรกด้วยวิธีการที่เป็นแบบแผนยิ่งขึ้น เมื่อขั้นตอนการสร้างโครงข่ายสามเหลี่ยมกำลังดำเนินไปก็จะเลือกแทรกสามเหลี่ยมซึ่งมีระยะห่างของจุดที่มากที่สุดในโครงข่ายเสมอ และความแตกต่างซึ่งเป็นผลจากลำดับของการแทรกจุดที่ต่างกันอาจเกิดขึ้นเมื่อเราต้องเลือกระหว่างสามเหลี่ยม 2 รูป (หรือมากกว่านั้น) ซึ่งมีระยะห่างมากที่สุดของจุดเท่ากันเท่านั้น

วิธีการหาเค้าโครงของภูมิประเทศในรูปของแบบจำลองพื้นผิวโดยการคัดเลือกจุดซึ่งมีระยะห่างมากที่สุดของแต่ละสามเหลี่ยมในโครงข่ายสามเหลี่ยมถูกประยุกต์มากจากวิธีการประมาณเส้นโค้ง เมื่อเส้นโค้ง (ใน 2 มิติ) ถูกแทนด้วยลำดับของเส้นตรง เช่นเดียวกับที่แบบจำลองพื้นผิว (ใน 3 มิติ) จะถูกแทนด้วยเซ็ตของระนาบสามเหลี่ยม



รูปที่ 5.2 ขั้นตอนการประมาณเส้นโค้งโดยใช้อัลกอริทึมของ Douglas-Peucker

5.1.1 วิธีการประมาณเส้นโค้ง

การประมาณเค้าโครงของเส้นโครง (Discretizing arcs) ด้วยลำดับของเส้นตรง (Polyline) โดยใช้อัลกอริทึมของ Douglas-Peucker [6][35] เมื่อพิจารณาเส้นโค้งในรูปที่ 5.2(ก) จุดใหม่จะถูกกำหนดขึ้นที่จุดปลายของเส้นโค้งต้นแบบ เส้นตรงใหม่ $\overline{ab}$ ได้ถูกสร้างขึ้นจากจุดปลายทั้งสอง (a และ b) จากนั้นจุด c จะถูกกำหนดขึ้นบนส่วนของเส้นโค้ง เมื่อ c คือจุดซึ่งอยู่ห่างที่สุดจากเส้นโค้งไปสู่ $\overline{ab}$ (ระยะทางจาก c ตั้งฉากกับ $\overline{ab}$) c จะถูกแทรกเข้าสู่ลำดับของจุดซึ่งทำหน้าที่แทนจุดสำคัญบนเส้นโค้ง เส้นโค้งได้ถูกแบ่งออกเป็น 2 ส่วนคือ $\overline{ac}$ และ $\overline{cb}$ การแบ่งส่วนของส่วนโค้งดำเนินไปแบบรีเดอร์สีฟ ในรูปที่ 5.2(ข) ส่วนของเส้นโค้งได้ถูกแบ่งออกเป็น $\overline{ad}$, $\overline{dc}$, $\overline{ce}$ และ

$\overline{eb}$ การแบ่งในแต่ละส่วนจะหยุดลงเมื่อระยะห่างจากจุดที่ไกลที่สุดบนส่วนของเส้นโค้งไปถึงเส้นตรงต่ำกว่าค่าเทร็สโฮลด์ ดังในตัวอย่าง $\overline{ce}$ และ $\overline{eb}$ ตรงตามคุณสมบัตินี้จึงไม่ถูกแบ่งต่อไป ยิ่งค่าของเทร็สโฮลด์มีค่าน้อยเท่าใดจำนวนของจุดสำหรับประมาณเส้นโค้งยิ่งมีจำนวนมากขึ้น วิธีการประมาณเส้นโค้งยังคงดำเนินต่อไปดังรูปที่ 5.2(ค) ถึง 5.2(ฉ) จนกระทั่งไม่มีส่วนของเส้นโค้งใดถูกแบ่งต่อไปอีก จากขั้นตอนการทำงานทั้งหมดในรูปที่ 5.2 จึงได้ลำดับของเส้นตรงซึ่งประมาณส่วนของเส้นโค้งดังนี้ $\overline{ah}$, $\overline{hf}$, $\overline{fd}$, $\overline{dg}$, $\overline{gj}$, $\overline{ji}$, $\overline{ic}$, $\overline{ce}$ และ $\overline{eb}$

5.1.2 เทร็สโฮลด์สำหรับการคัดเลือกจุดของสามเหลี่ยม

รูปที่ 5.3 แสดงระยะทางเทร็สโฮลด์ (λ_p) จุดใดซึ่งอยู่ใกล้สามเหลี่ยมน้อยกว่า λ_p จะไม่ถูกนำมารวมเข้ากับโครงข่ายสามเหลี่ยม ค่าเทร็สโฮลด์ซึ่งแสดงในรูปที่ 5.3 ถูกออกแบบสำหรับระยะทางซึ่งตั้งฉากระหว่างจุด $Q(x_Q, y_Q, z_Q)$ กับระนาบของ ΔABC ระยะทางนี้คำนวณได้ดังสมการที่ (5.1)

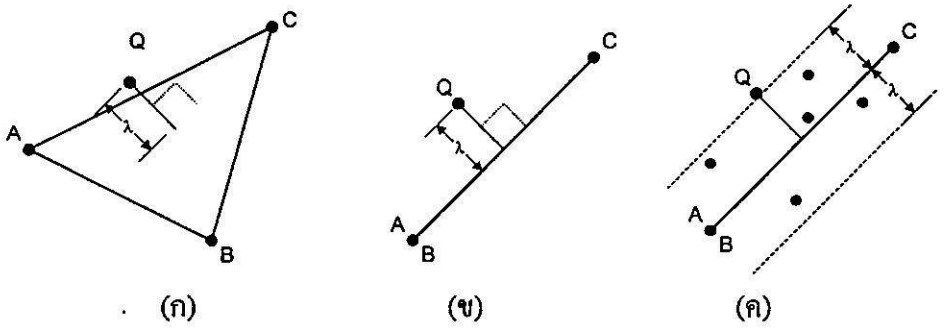
$$dist_p = \frac{ax_Q + by_Q + cz_Q + d}{\sqrt{a^2 + b^2 + c^2}} \tag{5.1}$$

เมื่อ
$$a = y_A(z_B - z_C) + y_B(z_C - z_A) + y_C(z_A - z_B)$$
$$b = x_A(z_C - z_B) + x_B(z_A - z_C) + x_C(z_B - z_A)$$
$$c = x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B)$$
$$d = x_A(y_C z_B - y_B z_C) + x_B(y_A z_C - y_C z_A) + x_C(y_B z_A - y_A z_B)$$

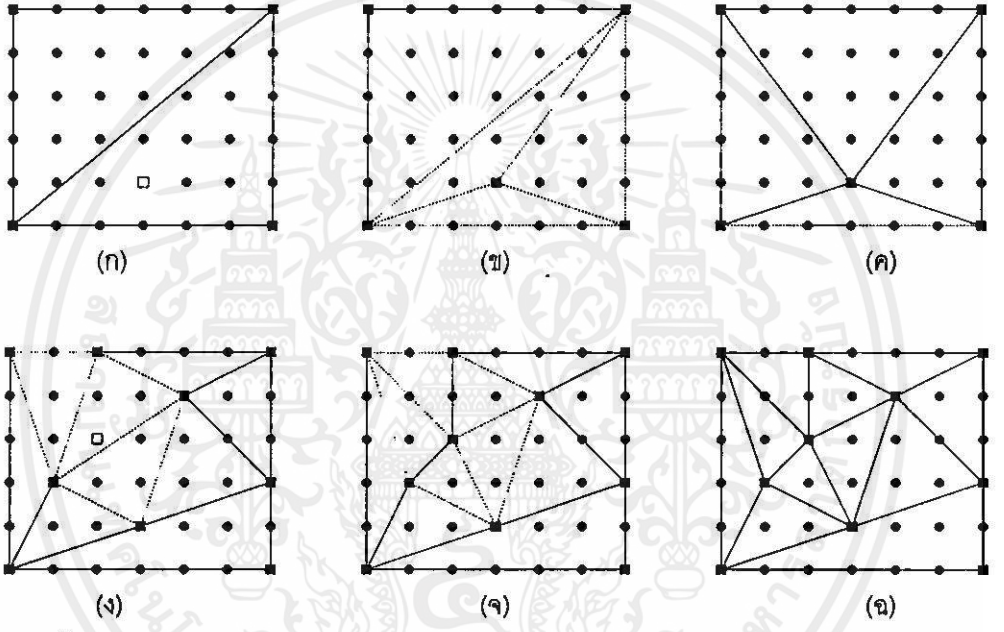
ในการคำนวณหาจุดซึ่งมีระยะทางห่างจากระนาบของสามเหลี่ยมมากที่สุด เราสามารถลดเวลาของการคำนวณลงได้โดยการคำนวณเฉพาะเศษบนของสมการที่ (5.1) เนื่องจากผลลัพธ์ของตัวหารส่วนล่าง (Denominator) จะมีค่าเท่ากันสำหรับทุกจุดซึ่งถูกปิดล้อมด้วยรูปสามเหลี่ยมเดียวกัน ดังนั้นจุดซึ่งถูกเลือกเพื่อเป็นจุดแทรกใหม่สามารถคำนวณได้จากสมการที่ (5.2)

$$s_p = \max\left|ax_i + by_i + cz_i + d\right| \tag{5.2}$$

เมื่อ $i = 1, 2, \ldots, n$ และ n คือจำนวนของจุดทั้งหมดซึ่งถูกปิดล้อมด้วย ΔABC



รูปที่ 5.3 (ก) ระยะทางเทร์สโฮลด์ตั้งฉากกับระนาบสามเหลี่ยม (ข) ภาพตัดขวางของสามเหลี่ยม (ค) พื้นที่ภายใต้ค่าเทร์สโฮลด์



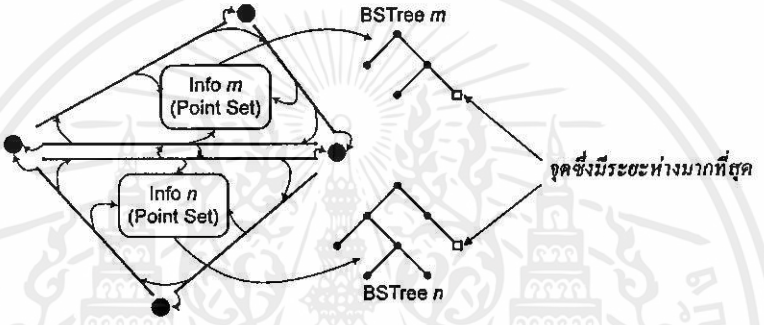
รูปที่ 5.4 ขั้นตอนการสร้างโครงข่ายสามเหลี่ยมจากกริดความสูง (ก) โครงข่ายเริ่มต้น (ข) จุดแรกถูกแทรกเข้าสู่โครงข่าย (ค) ขอบรอบข้างได้รับการสลับ (ง) จุดสุดท้ายถูกแทรกเข้าสู่โครงข่าย (จ) ขอบบางขอบถูกสลับ (ฉ) โครงข่ายผลลัพธ์หลังจากจุดสุดท้ายถูกรวมเข้าแล้วเสร็จ

5.2 สร้างโครงข่ายสามเหลี่ยมเพื่อประมาณกริดความสูง

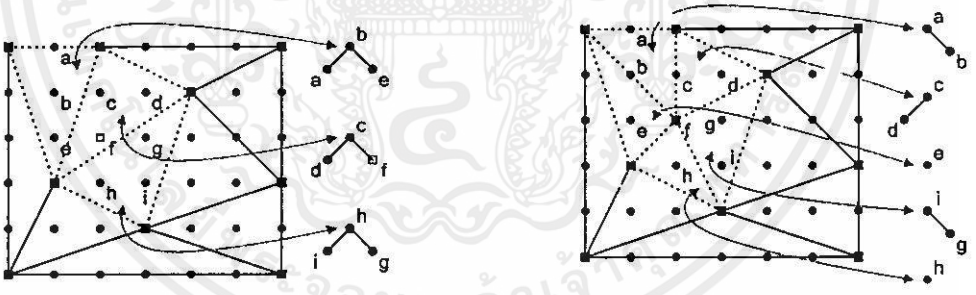
หนึ่งในปัญหาหลักเมื่อต้องจัดเก็บและคำนวณข้อมูลความสูงคือจำนวนของข้อมูลที่มากเกินไป ยกตัวอย่างเช่น ข้อมูลความสูงของพื้นที่ 9 × 18 ตารางกิโลเมตร มีระยะห่างระหว่างจุดสำรวจความสูง (Resolutions) เท่ากับ 30 เมตร จะได้กริดความสูงกว่า 2 แสนจุด ถ้าพื้นที่สำรวจยิ่งมีขนาดใหญ่หรือต้องการความละเอียดมากขึ้นจำนวนของจุดข้อมูลก็จะเพิ่มมากขึ้นเป็นทวีคูณ การสร้างโครงข่ายสามเหลี่ยมจากแบบจำลองความสูงชนิดกริดเป็นการลดความซ้ำซ้อนของข้อมูลความสูง เมื่อสับเซ็ตของจุดข้อมูลทั้งหมดจากกริดความสูงจะถูกคัดเลือกโดยใช้หลักการจากหัวข้อที่ 5.1

ดังนั้นโครงข่ายสามเหลี่ยมดีลอนเนย์ของสับเซ็ตข้อมูลนี้คือ TIN ซึ่งประมาณความสูงของจุดทุกจุดในกริดภายใต้ค่าเทร็สโฮลด์ที่กำหนด

จากรูปที่ 5.4 แสดงขั้นตอนของการสร้างโครงข่ายสามเหลี่ยมจากกริดความสูง เริ่มต้นจากโครงข่ายเริ่มต้นจะถูกสร้างขึ้นครอบคลุมกริดข้อมูลทั้งหมด จากนั้นจุดซึ่งได้รับการคัดเลือกจากกริดก็จะถูกเพิ่มเข้าสู่โครงข่ายสามเหลี่ยมอย่างต่อเนื่อง จนกระทั่งเป็นโครงข่ายสามเหลี่ยมดีลอนเนย์ที่สมบูรณ์ซึ่งประมาณค่าโครงของกริดความสูง และจากรูปที่ 5.4 ยังสังเกตเห็นได้อีกว่ามีจุดจำนวนหนึ่งซ้อนทับกับขอบบางขอบภายในโครงข่าย การแก้ไขปัญหานี้ได้อธิบายไว้ในหัวข้อที่ 4.11 เรื่องการกำจัดสามเหลี่ยมรูปลิ่มรวมไปถึงสามเหลี่ยมศูนย์บริเวณขอบอาณาเขตด้วย



รูปที่ 5.5 จุดภายในสามเหลี่ยมถูกจัดเก็บในโหนดของ BSTree



รูปที่ 5.6 การถ่ายเทจุดระหว่างสามเหลี่ยมโดยใช้ BSTree (ก) ก่อนการแทรกจุด (ข) จุดต่างๆถูกถ่ายเทไปสู่ BSTree อันใหม่

5.2.1 ปรับปรุงการจัดเก็บจุดภายในสามเหลี่ยม

เนื่องจากจุดทุกจุดจากกริดไม่ได้ถูกรวมเข้าสู่โครงข่ายทั้งหมด แต่ใช้การคัดเลือกเพียงจุดจำนวนหนึ่งจากเซ็ตข้อมูล การจัดเก็บจุดภายในสามเหลี่ยมแต่ละรูปจึงถูกปรับปรุงจากลำดับ (List) ของจุดไปเป็นไบนารีเสิร์ชทรี (Binary Search Tree หรือ BSTree) [25] เมื่อจุดแต่ละจุดภายในสามเหลี่ยมจะถูกเก็บในแต่ละโหนดของ BSTree ในขั้นตอนของการถ่ายเทจุดระหว่างสามเหลี่ยมเมื่อจุดถูกเปรียบเทียบกับเส้นทะแยงมุมของรูปสี่เหลี่ยมด้านไม่เท่าเพื่อหาสามเหลี่ยมที่เหมาะสมที่

จะบรรจุจุดนี้เสร็จลง จุดนี้จะถูกนำมาคำนวณหาระยะห่างจากจุดไปสู่สามเหลี่ยมซึ่งปิดล้อมจุดนี้อยู่ (ดังสมการที่ 5.2) ค่าระยะห่างจะถูกนำมาเปรียบเทียบกับค่าเทร์สโฮลด์ที่ตั้งไว้ ถ้าระยะห่างมากกว่าค่าเทร์สโฮลด์จุดนี้ก็จะถูกบรรจุลงใน BSTree จุดซึ่งมีระยะห่างมากที่สุดในเซ็ตของจุดในสามเหลี่ยมสามารถถูกค้นพบได้อย่างง่ายดายในโหนดขวาสุดของ BSTree (ดังรูปที่ 5.5) ในรูปที่ 5.6 แสดงการเปลี่ยนแปลงของ BSTree สำหรับสามเหลี่ยมแต่ละรูปที่ได้รับผลกระทบระหว่างการปรับปรุงโครงสร้างของโครงข่ายสามเหลี่ยม

5.3 การทดลองประมาณเค้าโค้งพื้นผิว

ในหัวข้อที่ 4.10 เป็นการแสดงผลการทำงานของอัลกอริทึม Incremental ในการคำนวณโครงข่ายสามเหลี่ยมจากเซ็ตของจุดแบบสุ่มซึ่งเป็นเพียงการทดสอบหลักการของวิธีการและอัลกอริทึมว่าได้ผลหรือไม่ ซึ่งจากผลการทดลองได้แสดงให้เห็นแล้วว่าหลักการต่างๆที่ได้ออกแบบมานั้นใช้การได้ผลและให้ประสิทธิภาพที่ดีขึ้นกว่าวิธีการสถิตแบบเดิมอย่างมาก จากหลักการดังกล่าวเมื่อมาประกอบกับวิธีการคัดเลือกจุด (ในหัวข้อที่ 5.1) เราสามารถนำมาประยุกต์ใช้งานกับข้อมูลทางภูมิศาสตร์จริงๆ

5.3.1 Digital Elevation Models

ข้อมูลความสูงของ 2 พื้นที่ในประเทศสหรัฐอเมริกาซึ่งเรียกว่า DEM หรือ Digital Elevation Models [28] ถูกนำมาใช้เป็นตัวอย่างในการทดลอง พื้นที่ทั้งสองนี้คือ ภูเขาไฟเซนต์เฮนเลนส์ (ภูเขาไฟดับแล้ว) ในรัฐวอชิงตัน และ แกรนแคนยอนในรัฐอาริโซน่า

DEM ถูกผลิตขึ้นโดยหน่วยงาน United States Geological Survey (USGS) ประกอบขึ้นจากข้อมูลความสูงของภูมิประเทศในระยะห่างของจุดสำรวจที่เท่ากันในรูปของกริด หรืออาร์เรย์ 2 มิติ DEM รูปแบบ 7.5 minute เป็น DEM ที่มีความละเอียดระหว่างจุดสำรวจมากที่สุดที่มีใช้ในปัจจุบันซึ่งคือ 30 เมตร ลำดับของข้อมูลความสูงจัดเรียงจากทิศใต้ไปสู่ทิศเหนือรวมเป็น 1 แถว (หรือ 1 profile) และแต่ละแถวเรียงจากทิศตะวันตกไปสู่ทิศตะวันออก ข้อมูลภายในอาร์เรย์ของ DEM ใช้ระบบ Universal Transverse Mercator (UTM) ในการอ้างอิงพิกัดตำแหน่ง ทำให้จำนวนของข้อมูลภายในแต่ละแถวอาจไม่เท่ากัน เนื่องจากความโค้งของเส้นกริดบนผิวโลกซึ่งอธิบายด้วย UTM ในรูปที่ 5.7 แสดง DEM ซึ่งมีขอบอาณาเขตที่ไม่สม่ำเสมอ ในการทดลองเลือกใช้เฉพาะข้อมูลความสูงในกรอบสี่เหลี่ยมผืนผ้าที่ใหญ่ที่สุดภายในกรอบของ DEM ทั้งสอง

รูปที่ 5.7 ใช้เทคนิค Grey-Scale ในการแสดงผลข้อมูลความสูงของ DEM โดยการใช้ระดับความเข้มของสีเท่าเทียบเคียงกับค่าความสูงของ DEM เมื่อสีดำแทนบริเวณที่ต่ำที่สุดแล้วไล่ระดับสีเทาไปตามค่าความสูงที่เปลี่ยนแปลงเพิ่มขึ้นไปสู่สีขาวซึ่งมีความสูงมากที่สุด ในรูปที่ 5.8

แสดงลักษณะภูมิประเทศของ DEM ทั้งสองพื้นที่ซึ่งใช้โปรแกรม DEM 3D (พัฒนาโดย USGS) ในการจำลองภาพ (Rendering)

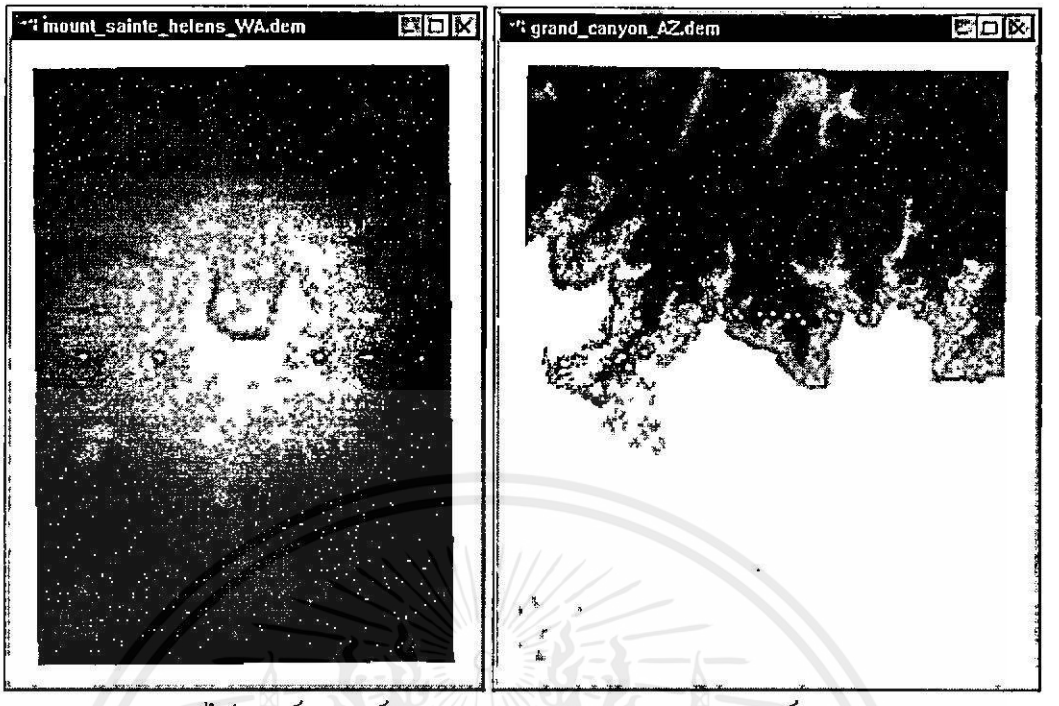
5.3.2 ผลการทดลอง

ในตารางที่ 5.1 และ 5.2 แสดงผลการทดลองของการคำนวณโครงข่ายสามเหลี่ยมจากข้อมูลความสูงแบบกริดของ DEM ของพื้นที่ภูเขาไฟเซนต์เฮเลนส์และแกรนแคนยอนเมื่อค่าเทร็สโฮลด์มีการเปลี่ยนแปลง ตารางที่ 5.1(ก) และ 5.2(ก) แสดงผลการคำนวณโครงข่ายสามเหลี่ยมโดยใช้ลำดับการแทรกจุดแบบก่อนหลัง เพื่อเปรียบเทียบกับการคำนวณโครงข่ายสามเหลี่ยมโดยใช้ลำดับการแทรกจุดแบบสุ่ม จากผลการทดลองทำให้ทราบว่าลำดับการแทรกจุดแบบสุ่มให้ผลลัพธ์ที่รวดเร็วกว่าลำดับการแทรกจุดแบบสุ่ม

จำนวนการแทรกจุด จำนวนการทดสอบภายมุมภายใน และจำนวนขอบที่ถูกสลับแสดงให้เห็นถึงความซับซ้อนของภูมิประเทศซึ่งถูกอธิบายด้วยโครงข่ายสามเหลี่ยมเมื่อค่าเทร็สโฮลด์เป็นตัวกำหนดระดับความซับซ้อน เมื่อเทร็สโฮลด์มีค่าสูงขึ้นแบบจำลอง TIN จะมีความซับซ้อนน้อยลง จำนวนจุดผลลัพธ์ และจำนวนสามเหลี่ยมผลลัพธ์คือจำนวนจุดและจำนวนสามเหลี่ยมที่หลงเหลือหลังจากการคำนวณโครงข่ายสามเหลี่ยมเสร็จสิ้น เมื่อนำเอาจำนวนผลลัพธ์ของจุดมาหาอัตราส่วนเปรียบเทียบกับจำนวนจุดต้นฉบับจาก DEM ทำให้เราทราบว่าการคำนวณโครงข่ายสามเหลี่ยมช่วยลดจำนวนของจุด (ซึ่งทำหน้าที่อธิบายแบบจำลองความสูง) ลงไปได้มาก ข้อมูลความเปลี่ยนแปลงของอัตราส่วนของจุดจากตารางที่ 5.1 และ 5.2 ถูกนำมาแสดงผลในรูปของกราฟในรูปที่ 5.9 เพื่อให้เห็นภาพการเปลี่ยนแปลงอย่างชัดเจน

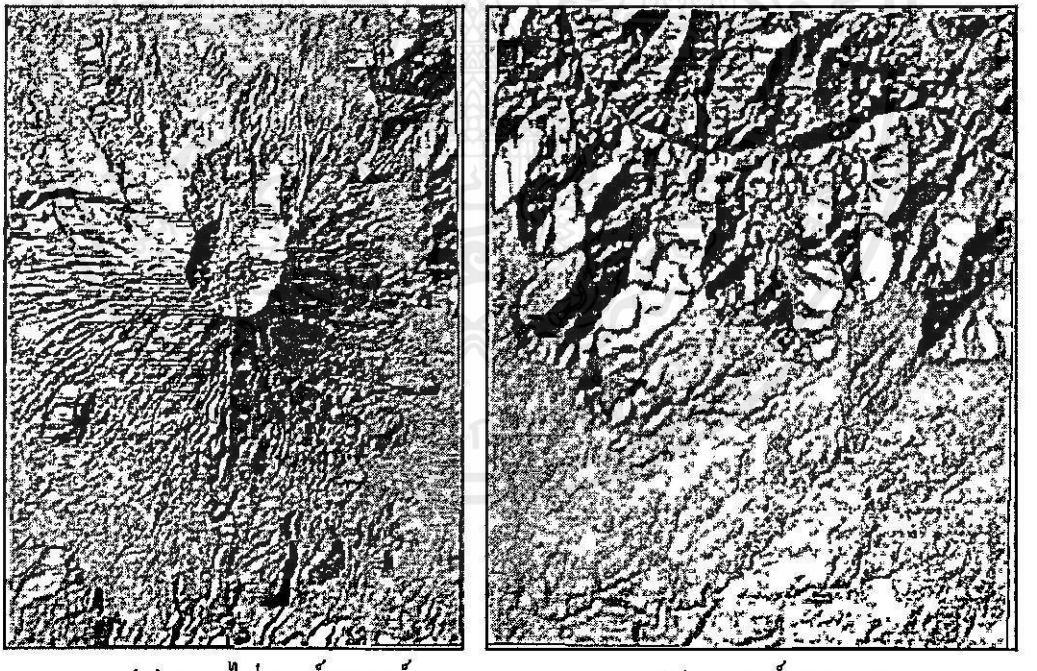
ความซับซ้อนของโครงข่ายสามเหลี่ยมแปรผันโดยตรงกับเวลาที่ใช้ในการคำนวณ เมื่อค่าเทร็สโฮลด์มากขึ้นความซับซ้อนของโครงข่ายจะน้อยลงสวนทาง เช่นเดียวกับที่เวลาที่ใช้ในการคำนวณจะน้อยลงตามไปด้วย ข้อมูลความเปลี่ยนแปลงของเวลาที่ใช้คำนวณโครงข่ายสามเหลี่ยมจากตารางที่ 5.1 และ 5.2 ถูกนำมาแสดงผลในรูปแบบกราฟดังรูปที่ 5.10

เพื่อให้มองเห็นภาพการเปลี่ยนแปลงของโครงข่ายสามเหลี่ยมอย่างชัดเจนยิ่งขึ้นเมื่อค่าของเทร็สโฮลด์มีการเปลี่ยนแปลงเพิ่มขึ้น ในรูปที่ 5.11 และ 5.12 แสดงโครงข่ายสามเหลี่ยมซึ่งครอบคลุมพื้นที่บางส่วนของภูเขาไฟเซนต์เฮเลนส์และแกรนด์แคนยอน พื้นที่ของแกรนด์แคนยอนที่นำเสนอนี้จะมีความซับซ้อนมากกว่าพื้นที่ของภูเขาไฟเซนต์เฮเลนส์ แผนที่คอนทัวร์ได้ถูกนำเสนอพร้อมไปด้วยเพื่อให้รับรู้ถึงเค้าโครงของภูมิประเทศ จากผลการทดลองได้แสดงให้เห็นว่าเค้าโครงหลักของแบบจำลองยังคงอยู่ตลอดกระบวนการ และเค้าโครงอย่างหยาบของโครงข่ายสามเหลี่ยมสามารถนำมาถ่ายทอดให้เป็นแผนที่คอนทัวร์ได้เพื่อพิจารณาเค้าโครงโดยรวมของภูมิประเทศนั้นๆ



(ก) ภูเขาไฟเซนต์เฮเลนส์ (ข) แกรนด์แคนยอน

รูปที่ 5.7 พื้นที่ปกคลุมด้วย DEM รูปแบบ 7.5-minute (ใช้เทคนิคระดับสีเทาในการแสดงผล)



(ก) ภูเขาไฟเซนต์เฮเลนส์ (ข) แกรนด์แคนยอน

รูปที่ 5.8 พื้นที่ปกคลุมด้วย DEM รูปแบบ 7.5-minute (ใช้เทคนิค Gouraud Shading ในการให้สี)

ตารางที่ 5.1 ผลลัพธ์การทดลองเมื่อ DEM ของพื้นที่ภูเขาไฟเซนต์เฮเลนส์ถูกคำนวณสามเหลี่ยม (เมื่อจำนวนจุดต้นฉบับ = 145820 (317 × 460), จำนวนสามเหลี่ยมต้นฉบับ = 290088)

(ก) ลำดับแบบก่อนหลัง

เทร์สโฮลด์ระยะทาง (เมตร)	20	30	40	60	80	100
จำนวนการแทรกจุด	16676	3907	1936	716	411	270
จำนวนการทดสอบมุมภายใน	166302	35657	17222	6111	3465	2238
จำนวนขอบที่ถูกสลับ	58151	11982	5721	1996	1131	729
จำนวนจุดผลลัพธ์	16680	3911	1940	720	415	274
จำนวนสามเหลี่ยมผลลัพธ์	33354	7816	3874	1434	824	542
อัตราส่วนของจุด (%)	11.44	2.68	1.33	0.49	0.29	0.19
อัตราส่วนของสามเหลี่ยม (%)	11.50	2.69	1.34	0.49	0.28	0.19
เวลาที่ใช้คำนวณ (นาที)	30.59	5.47	4.29	3.41	3.31	3.22

(ข) ลำดับแบบสุ่ม

เทร์สโฮลด์ระยะทาง (เมตร)	20	30	40	60	80	100
จำนวนการแทรกจุด	16463	3786	1894	747	424	274
จำนวนการทดสอบมุมภายใน	161189	33671	16478	6369	3663	2199
จำนวนขอบที่ถูกสลับ	55919	11174	5415	2080	1214	707
จำนวนจุดผลลัพธ์	16467	3790	1898	751	428	278
จำนวนสามเหลี่ยมผลลัพธ์	32928	7574	3790	1496	850	550
อัตราส่วนของจุด (%)	11.30	2.60	1.30	0.52	0.29	0.19
อัตราส่วนของสามเหลี่ยม (%)	11.35	2.61	1.31	0.52	0.29	0.19
เวลาที่ใช้คำนวณ (นาที)	25.36	4.35	4.15	3.13	3.03	2.53

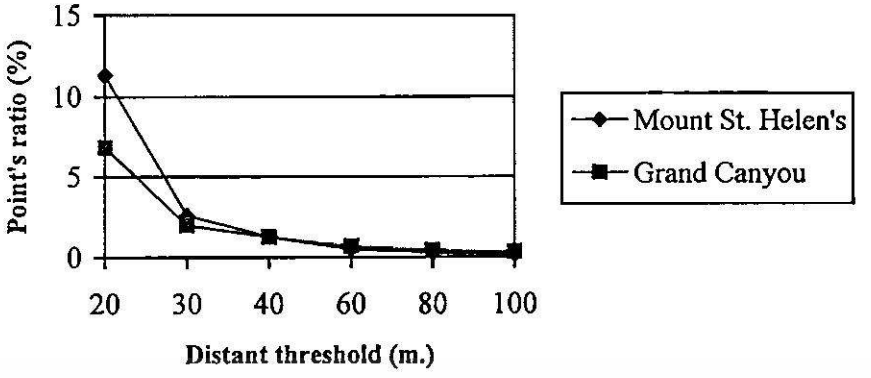
ตารางที่ 5.2 ผลลัพธ์การทดลองเมื่อ DEM ของพื้นที่แกรนด์แคนยอนถูกคำนวณสามเหลี่ยม (เมื่อจำนวนจุดต้นฉบับ = 169002 (369 × 458), จำนวนสามเหลี่ยมต้นฉบับ = 336352).

(ก) ลำดับแบบก่อนหลัง

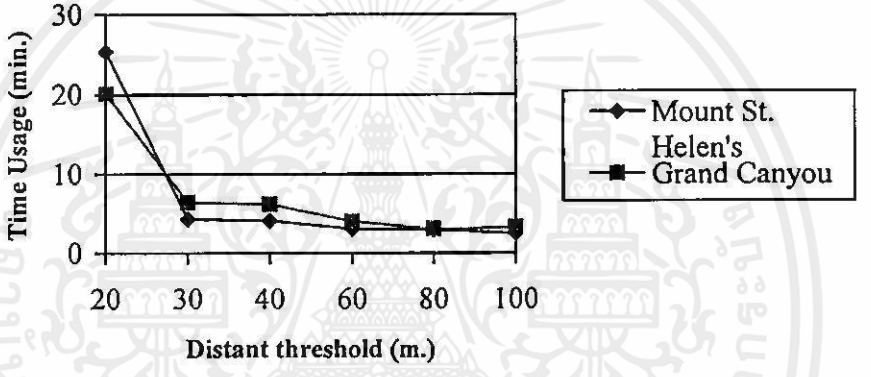
เทร็สโฮลด์ระยะทาง (เมตร)	20	30	40	60	80	100
จำนวนการแทรกจุด	11911	3512	2216	1210	797	551
จำนวนการทดสอบมุมภายใน	114051	31587	19610	10540	6869	4688
จำนวนขอบที่ถูกสลับ	39183	10550	6500	3470	2252	1530
จำนวนจุดผลลัพธ์	11915	3516	2220	1214	801	555
จำนวนสามเหลี่ยมผลลัพธ์	23824	7026	4434	2422	1596	1104
อัตราส่วนของจุด (%)	7.05	2.08	1.31	0.72	0.47	0.32
อัตราส่วนของสามเหลี่ยม (%)	7.08	2.0	1.32	0.72	0.48	0.32
เวลาที่ใช้คำนวณ (นาที)	20.37	7.29	7.13	4.41	3.53	3.41

(ข) ลำดับแบบสุ่ม

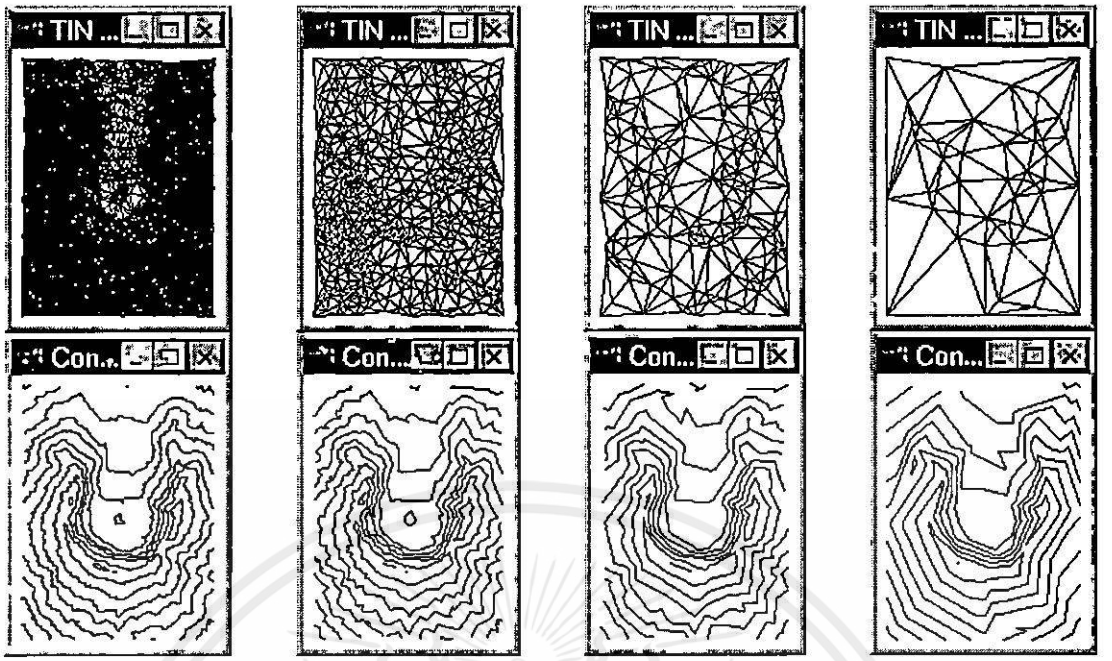
เทร็สโฮลด์ระยะทาง (เมตร)	20	30	40	60	80	100
จำนวนการแทรกจุด	11530	3371	2146	1194	774	565
จำนวนการทดสอบมุมภายใน	107606	29424	18730	10327	6630	4767
จำนวนขอบที่ถูกสลับ	36535	9683	6174	3394	2169	1548
จำนวนจุดผลลัพธ์	11534	3375	2150	1198	778	569
จำนวนสามเหลี่ยมผลลัพธ์	23062	6744	4294	2390	1550	1132
อัตราส่วนของจุด (%)	6.82	2.00	1.27	0.71	0.46	0.34
อัตราส่วนของสามเหลี่ยม (%)	6.86	2.00	1.28	0.71	0.46	0.34
เวลาที่ใช้คำนวณ (นาที)	20.10	6.46	6.24	4.06	3.10	3.32



รูปที่ 5.9 อัตราส่วนของจำนวนจุดที่ลดลงต่อค่าเทร็สโฮลด์ต่างๆ

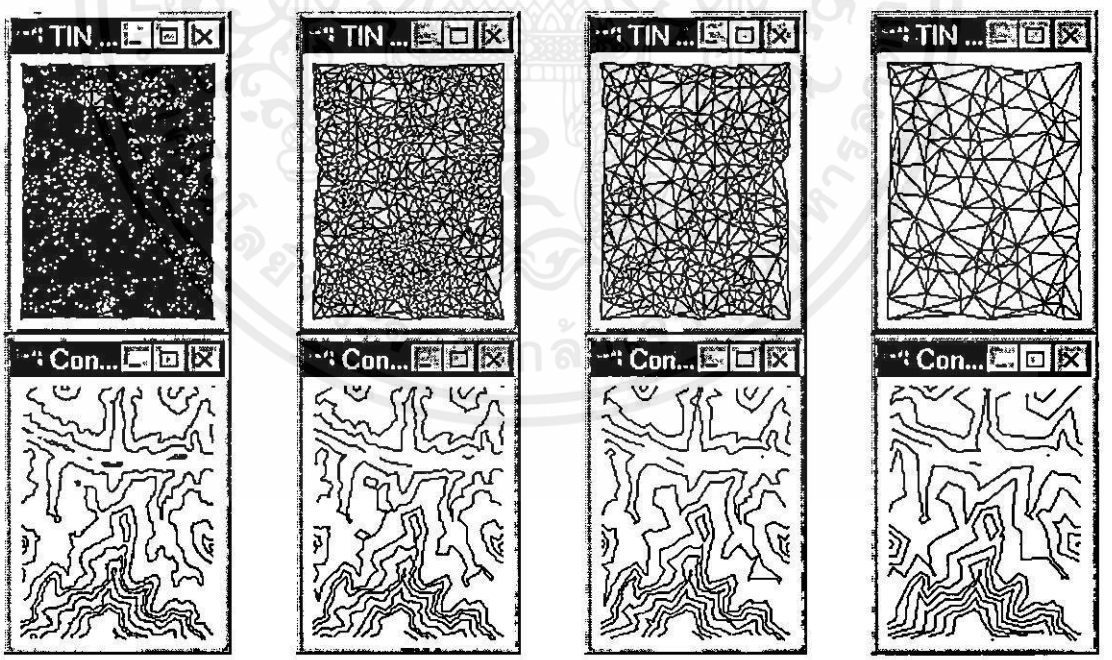


รูปที่ 5.10 ความเร็วในการคำนวณโครงข่ายสามเหลี่ยมต่อค่าเทร็สโฮลด์ต่างๆ



$\lambda = 0$	$\lambda = 30$	$\lambda = 50$	$\lambda = 100$
11257 จุด (58.63%)	677 จุด (3.52%)	186 จุด (0.97%)	50 จุด (0.26%)

รูปที่ 5.11 ภาพความแตกต่างของโครงข่ายสามเหลี่ยมและแผนที่คอนทัวร์ของภูเขาไฟเซนต์เฮเลนส์เมื่อค่าของเทร์สโฮลด์ (λ) เปลี่ยนแปลง (จำนวนจุดต้นฉบับ=19200)



$\lambda = 0$	$\lambda = 30$	$\lambda = 50$	$\lambda = 100$
9799 จุด (51.04%)	703 จุด (3.66%)	342 จุด (1.78%)	125 จุด (0.65%)

รูปที่ 5.12 ภาพความแตกต่างของโครงข่ายสามเหลี่ยมและแผนที่คอนทัวร์ของแกรนแคนยอนเมื่อค่าของเทร์สโฮลด์ (λ) เปลี่ยนแปลง (จำนวนจุดต้นฉบับ=19200)

5.4 การลบจุดออกจากโครงข่ายสามเหลี่ยม

การลบจุดเป็นกระบวนการที่สำคัญอันหนึ่งของการจัดการโครงข่ายสามเหลี่ยมดีลอนเนย์แบบไดนามิก อย่างไรก็ตามการศึกษาในเรื่องนี้ยังมีอยู่น้อย แตกต่างจากการแทรกจุดเข้าสู่โครงข่ายซึ่งที่ผ่านมาได้รับการศึกษาอย่างกว้างขวาง สำหรับการจัดการโครงข่ายสามเหลี่ยมดีลอนเนย์แบบสถิตโดยทั่วไปแล้วต้องคำนวณโครงข่ายสามเหลี่ยมใหม่ทั้งหมดเมื่อเกิดการเปลี่ยนแปลงกับเซ็ตของจุด (เพิ่มหรือลบจุด)

จากทฤษฎีที่ 3 เมื่อจุดถูกแทรกเข้าสู่โครงข่ายดีลอนเนย์ขอบซึ่งได้รับผลกระทบจากการแทรกจุด ปลายด้านหนึ่งของขอบเหล่านี้จะถูกสลับให้ชี้ไปที่จุดแทรกใหม่เสมอ ส่วนปลายอีกด้านหนึ่งจะพุ่งออกจากจุดแทรกใหม่ไปสู่มุมภายในของรูปหลายเหลี่ยมซึ่งปิดล้อมจุดนี้อยู่ดังรูปที่ 5.13 อย่างไรก็ตามรูปหลายเหลี่ยมนี้ไม่จำเป็นต้องเป็นชนิดคอนเว็กซ์เสมอไปดังรูปที่ 5.15 จากทฤษฎีที่ 7 สามเหลี่ยมซึ่งอยู่ภายนอกพื้นที่ของรูปหลายเหลี่ยมจะไม่ได้รับผลกระทบจากการแทรกจุด ดังนั้นการลบจุดเดียวกันนี้จึงไม่มีผลกระทบต่อโครงข่ายภายนอกพื้นที่นี้

5.4.1 อัลกอริทึม Basis

อัลกอริทึม Basis ถูกนำเสนอโดย [11] เมื่อกำหนดให้ τ_{n+1} คือโครงข่ายสามเหลี่ยมก่อนการลบจุด และ τ_n คือโครงข่ายสามเหลี่ยมผลลัพธ์หลังการลบจุด

(ก) (ข) (ค)

รูปที่ 5.13 (ก) ขอบซึ่งอยู่ภายในรูปหลายเหลี่ยมจะถูกลบ (ข) วงกลมทดสอบที่เล็กที่สุดตามแนวขอบเขตของรูปหลายเหลี่ยม (ค) ขอบใหม่ถูกสร้างขึ้นปิดด้านที่หายไปของสามเหลี่ยมซึ่งมีวงกลมทดสอบเล็กที่สุด

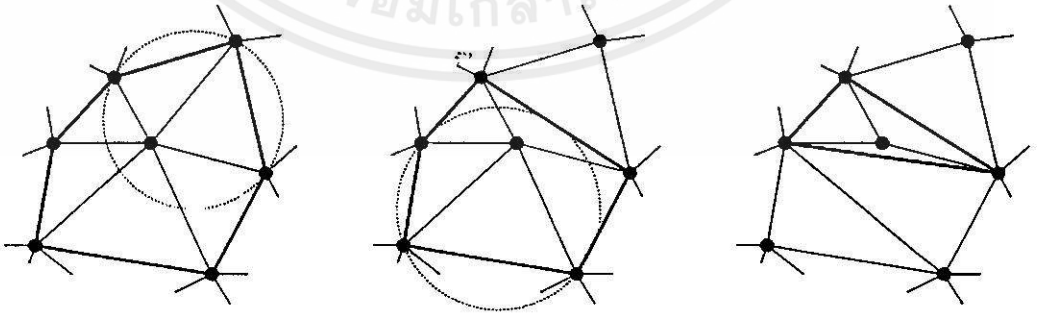
เมื่อรูปหลายเหลี่ยมซึ่งปิดล้อมจุดที่ต้องการลบถูกค้นพบ สามเหลี่ยมภายในพื้นที่นี้ก็จะถูกลบออกโดยการลบขอบซึ่งชี้ไปที่จุดเป้าหมาย ขอบ 2 ขอบซึ่งอยู่ติดกันตามแนวขอบเขตของรูปหลายเหลี่ยมอาจจะประกอบกันขึ้นเป็นสามเหลี่ยม ดังนั้นวงกลมทดสอบของขอบ 2 ขอบซึ่งอยู่ติดกันทุกคู่ตามแนวขอบเขตจะถูกคำนวณเพื่อเปรียบเทียบหาวงกลมทดสอบซึ่งมีเส้นผ่านศูนย์ที่สั้นที่

สุด เมื่อพบก็จะสร้างขอบใหม่ขึ้นปิดขอบคู่ทั้งสองนี้ให้เกิดเป็นสามเหลี่ยมที่สมบูรณ์ซึ่งเป็นส่วนหนึ่งของ τ_n พื้นที่ของรูปหลายเหลี่ยมจะเล็กลงเพราะด้านและเหลี่ยมถูกลดจำนวนลง กระบวนการค้นหาและสร้างสามเหลี่ยมซึ่งมีวงกลมทดสอบเล็กที่สุดจะดำเนินต่อไปภายใต้พื้นที่ของรูปหลายเหลี่ยมใหม่ที่เล็กลงเรื่อยๆ จนกระทั่งรูปหลายเหลี่ยมเหลือด้านเพียง 3 ด้าน (สามเหลี่ยมรูปสุดท้ายของ τ_n) จึงสิ้นสุดการทำงาน

กรณียกเว้นอาจเกิดขึ้นเมื่อรูปหลายเหลี่ยมมีลักษณะไม่เป็นคอนเว็กซ์ทำให้มีมุมภายในบางมุมของรูปหลายเหลี่ยมมีค่ามากกว่า π ปัญหานี้จะถูกแก้ไขโดยใช้หลักการในหัวข้อที่ 5.4.2 เพราะภายในวงกลมทดสอบนี้จะไม่มีจุดอื่นใดบรรจุอยู่อย่างแน่นอน หากเลือกสามเหลี่ยมอื่นซึ่งมีจุดบรรจุอยู่ภายใน แสดงว่ายังมีสามเหลี่ยมอื่นซึ่งมีวงกลมทดสอบที่เล็กกว่า ผลลัพธ์ของโครงข่ายสามเหลี่ยมสุดท้ายจึงตรงตามนิยามของดีลอนเนย์ทุกประการ

5.4.2 อัลกอริทึม Reversion

อัลกอริทึม Reversion ถูกปรับปรุงจากอัลกอริทึม Basis (หัวข้อที่ 5.4.1) โดยมีวัตถุประสงค์เพื่อสร้างกระบวนการย้อนกลับสำหรับอัลกอริทึม Incremental ระหว่างขั้นตอนการแทรกจุดขอบใหม่ 3 ขอบจะถูกเพิ่มเข้าสู่โครงข่ายและตามด้วยการปรับปรุงโครงสร้างภายใต้พื้นที่ผลกระทบ จากการทดลองพบว่าการดำเนินการย้อนกลับโดยใช้การทดสอบผลรวมของมุมภายในรูปสี่เหลี่ยมด้านไม่เท่าและสลับขอบกลับคืนไม่มีทางเป็นไปได้เนื่องจากไม่สามารถค้นหาลำดับของการสลับขอบหลังจากการแทรกจุดก่อนหน้านี้ได้ ดังนั้นวงกลมทดสอบตามแนวขอบเขตของรูปหลายเหลี่ยมจึงต้องถูกคำนวณ เมื่อวงกลมทดสอบที่เล็กที่สุดถูกค้นพบขอบซึ่งเชื่อมระหว่างมุมของรูปหลายเหลี่ยม (มุมตรงกลางระหว่างด้านทั้งสอง) และจุดที่ต้องการลบจะถูกสลับ เส้นผ่านศูนย์กลางของวงกลมทดสอบคำนวณได้จากสมการที่ (5.3)



(ก) (ข) (ค)

รูปที่ 5.14 (ก) วงกลมทดสอบที่เล็กที่สุดตามแนวขอบเขตของรูปหลายเหลี่ยม (ข) ขอบถูกสลับและวงกลมทดสอบสามเหลี่ยมอันใหม่ถูกค้นหาต่อไป (ค) เมื่อรูปหลายเหลี่ยมมีพื้นที่ลดลงเหลือเพียงสามขอบ ขอบภายในทั้งสามขอบซึ่งชี้ไปที่จุดเป้าหมายจะถูกลบทิ้ง

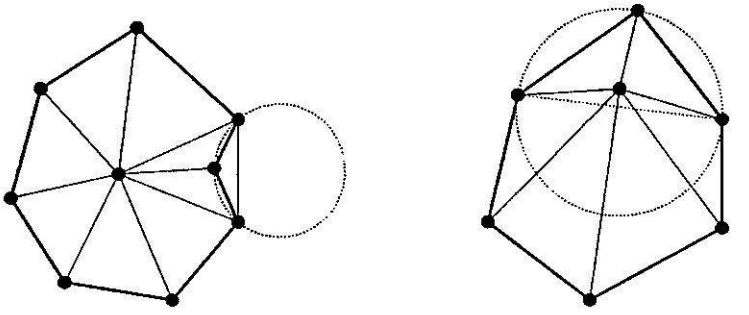
$$d = \frac{a}{\sin\alpha} \qquad ; \alpha \notin [0, \pi] \tag{5.3}$$

เมื่อ a คือด้านหนึ่งของสามเหลี่ยม และ α คือมุมซึ่งตรงข้ามกับ a ขั้นตอนการลบจุดโดยใช้อัลกอริทึม Reversion แสดงได้ดังนี้

1. รูปหลายเหลี่ยมซึ่งล้อมรอบจุดที่ต้องการลบจะถูกกำหนดขึ้น
2. เส้นผ่านศูนย์กลางของวงกลมทดสอบทุกรูปตามแนวขอบเขตของรูปหลายเหลี่ยมจะถูกคำนวณ (ดังรูปที่ 5.14(ก)) และนำมาจัดเก็บในลำดับข้อมูลของเส้นผ่านศูนย์กลางวงกลมทดสอบซึ่งมีขนาดเล็กที่สุดก็จะถูกเลือกออกมาจากลำดับข้อมูลนี้
3. ขอบซึ่งเชื่อมระหว่างมุมยอดปัจจุบันและจุดที่ต้องการลบจะถูกสลับ (ดังรูปที่ 5.14 (ข))
4. เมื่อขอบถูกสลับ ขอบเขตของรูปหลายเหลี่ยมก็จะถูกปรับลดโดยอัตโนมัติ และวงกลมทดสอบที่เล็กที่สุดก็จะถูกค้นหาต่อไป
5. ขั้นตอนที่ 1 ถึง 4 จะถูกกระทำซ้ำจนกระทั่งรูปหลายเหลี่ยมถูกลดขนาดลงเหลือด้านเพียง 3 ด้าน (ดังรูปที่ 5.14(ค)) ขอบภายในทั้ง 3 ขอบซึ่งชี้ไปที่จุดเป้าหมายก็จะถูกลบทิ้งเพื่อให้โครงข่ายบรรลุถึง τ_n

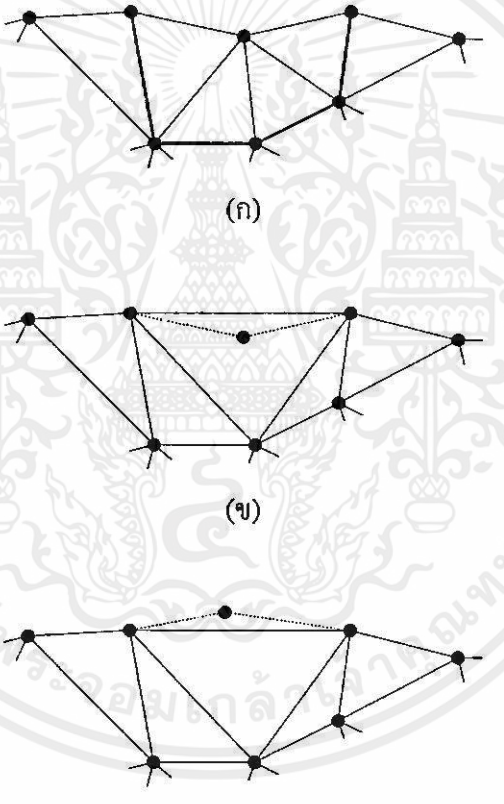
เห็นได้ว่าอัลกอริทึมนี้ไม่ซับซ้อนมากนัก แต่อย่างไรก็ตามมีข้อยกเว้นอยู่ 2 กรณีที่ต้องจัดการคือ

1. รูปหลายเหลี่ยมอาจไม่เป็นแบบคอนเว็กซ์เสมอไป ถ้ามุมภายในของรูปหลายเหลี่ยมมีค่ามากกว่า π สามเหลี่ยมทดสอบซึ่งเกิดจากด้านที่ติดกันสองด้านของรูปหลายเหลี่ยมจะปรากฏอยู่ด้านนอกรูปหลายเหลี่ยม (ดังรูปที่ 5.15(ก)) ดังนั้นขอบซึ่งเชื่อมระหว่างมุมยอดนี้กับจุดเป้าหมายจึงไม่สามารถอนุญาตให้ถูกสลับได้
2. เมื่อจุดใหม่ถูกแทรกเข้าสู่ τ_n ในครั้งแรก จุดนี้จะถูกเชื่อมเข้ากับสามเหลี่ยมปิดล้อมโดยขอบใหม่ 3 ขอบ ด้วยการใช้อัลกอริทึม Reversion เพื่อย้อนรอยอัลกอริทึม Incremental ขอบใหม่ทั้ง 3 ขอบนี้จะเป็นขอบสุดท้ายที่ต้องถูกลบออก ผลที่ตามมาคือสามเหลี่ยมปิดล้อมเริ่มต้นเมื่อครั้งที่เกิดการแทรกจุดใหม่สามเหลี่ยมนี้ต้องเป็นสามเหลี่ยมเดียวกันกับรูปหลายเหลี่ยมสุดท้ายที่บรรจุขอบภายใน 3 ขอบที่ต้องถูกลบทิ้ง กรณีผิดพลาดอาจเกิดขึ้นเมื่อสามเหลี่ยมปิดล้อมเริ่มแรกนี้อาจถูกลบทิ้งก่อนเวลาอันควรระหว่างขั้นตอนการสลับขอบแบบย้อนกลับ สามเหลี่ยมซึ่งปิดล้อมจุดที่ต้องการลบและเป็นสามเหลี่ยมที่มีวงกลมทดสอบที่เล็กที่สุดจะไม่ถูกอนุญาตให้เกิดการสลับขอบ (ดังรูปที่ 5.15(ข)) สามเหลี่ยมนี้จะถูกข้ามผ่านไปและพิจารณาเลือกสามเหลี่ยมซึ่งมีวงกลมทดสอบที่เล็กที่สุดในลำดับถัดไปแทน



(ก) (ข)

รูปที่ 5.15 ขอบซึ่งไม่สามารถถูกสลับได้ (ก) เมื่อมุมภายในของรูปหลายเหลี่ยมมีค่ามากกว่า π (ข) เมื่อจุดที่ต้องการลบมีตำแหน่งอยู่ภายในสามเหลี่ยมทดสอบ

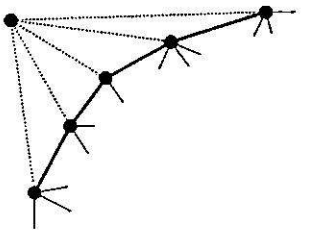


(ก)

(ข)

(ค)

รูปที่ 5.16 (ก) รูปหลายเหลี่ยมดรึ่งรูป (ข) สองขอบสุดท้ายถูกลบออกเมื่อจุดเว้าเข้าสู่โครงข่าย (ค) สองขอบสุดท้ายถูกลบออกเมื่อจุดนูนออกจากโครงข่าย



รูปที่ 5.17 จุดที่มุมและขอบข้างเดียงถูกลบออกจากโครงข่ายได้ทันที

5.4.3 การลบจุดที่ขอบเขตของโครงข่ายสามเหลี่ยม

อัลกอริทึมซึ่งได้อธิบายไปก่อนหน้านี้เป็นการลบจุดซึ่งมีตำแหน่งอยู่ภายในโครงข่ายสามเหลี่ยม กับการปรับปรุงเพียงเล็กน้อยอัลกอริทึมนี้ก็สามารถลบจุดซึ่งอยู่บริเวณขอบเขตของโครงข่ายได้ ด้วยการใช้กฎเกณฑ์เดียวกันสำหรับรูปหลายเหลี่ยมครึ่งรูปซึ่งปิดล้อมจุดเป้าหมายบางส่วนที่ต้องการลบ ในรูปที่ 5.16(ก) แสดงการทำงานของอัลกอริทึม Reversion สำหรับการลบจุดซึ่งอยู่บริเวณขอบเขตของโครงข่าย ความแตกต่างที่สำคัญกับอัลกอริทึมข้างต้นคือ มีขอบเพียง 2 ขอบเท่านั้นที่จะถูกลบเมื่อถึงขั้นตอนสุดท้าย ถ้าจุดเป้าหมายคอนเว็กซ์หรืออยู่ภายในโครงข่ายขอบทั้งสองจะอยู่นอกโครงข่าย (ดังรูปที่ 5.16(ค)) ในทางกลับกันขอบทั้งสองจะอยู่ภายในสามเหลี่ยมเมื่อจุดนั้นคอนเคฟ (concave) หรือเว้าเข้าสู่โครงข่าย (ดังรูปที่ 5.16(ข)) เมื่อรูปหลายเหลี่ยมครึ่งรูปเว้าเข้าหาจุดเป้าหมายทั้งหมด (ดังรูปที่ 5.17) ในกรณีเช่นนี้จุดเป้าหมายและขอบซึ่งพุ่งออกจากจุดเป้าหมายสามารถถูกลบออกจากโครงข่ายได้ทันทีโดยที่ไม่ต้องมีการสลับขอบแต่อย่างใด

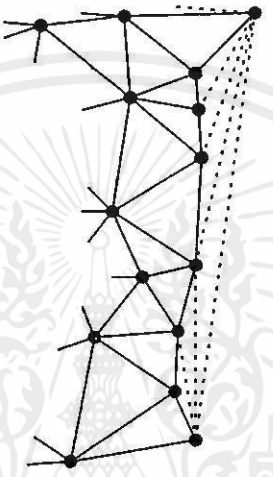
5.5 การเชื่อมต่อสองโครงข่ายดีลอนเนย์

การเชื่อมโครงข่ายสามเหลี่ยมดีลอนเนย์ 2 โครงข่ายเข้าหากันมีประโยชน์อย่างยิ่งสำหรับจัดการกับแบบจำลองขนาดใหญ่ เมื่อเราสามารถจัดการกับบางส่วนของแบบจำลองแล้วจึงเชื่อมต่อส่วนนี้เข้ากับส่วนอื่นของโครงข่าย การทำงานกับพื้นที่เล็กๆทำให้การสำรวจทำได้ละเอียดและดำเนินการง่ายกว่า ในหัวข้อนี้จะอธิบายทฤษฎีและอัลกอริทึมสำหรับการเชื่อมต่อสองโครงข่ายสามเหลี่ยม ซึ่งเป็นเรื่องที่น่าสนใจที่จะทำการศึกษาว่าโครงข่ายข้างเดียงมีผลกระทบซึ่งกันและกันอย่างไรระหว่างขั้นตอนของการเชื่อมโยงกำลังดำเนินไป

โครงข่ายสามเหลี่ยมดีลอนเนย์ 2 โครงข่ายถูกแทนด้วย τ_L และ τ_R เมื่อ τ_L คือโครงข่ายทางด้านซ้าย และ τ_R คือโครงข่ายทางด้านขวา ทั้งสองโครงข่ายถูกเชื่อมเข้ารวมกันเป็น τ ถ้าภายใน τ_L ประกอบด้วยจุดจำนวน l จุด และภายใน τ_L ประกอบด้วยจุดจำนวน m จุด ดังนั้น τ จึงประกอบด้วยจุดจำนวน $l+m=n$ จุด สามเหลี่ยมจะเป็นสมาชิกของ $\tau_R(\tau_L)$ ถ้าจุดที่มุมยอดทั้งสามเป็นสมาชิกของเซ็ตข้อมูลของ $\tau_R(\tau_L)$ และไม่มีจุดอื่นใดในโครงข่ายถูกปิดล้อมด้วยวงกลมทดสอบของสามเหลี่ยมนี้

5.5.1 โครงข่ายข้างเคียงต่างก็มีผลกระทบต่อกัน

เมื่อโครงข่ายสามซึ่งแยกจากกัน 2 โครงข่ายถูกเชื่อมเข้าหากันจะเกิดการเปลี่ยนแปลงเกิดขึ้นในพื้นที่อาณาเขตระหว่างสองโครงข่าย ขอบเขตของการเปลี่ยนแปลงจะขึ้นอยู่กับการกระจายตัวของจุดบริเวณอาณาเขตและลักษณะโครงสร้างของโครงข่ายสามเหลี่ยมบริเวณนี้ สามเหลี่ยมซึ่งมีลักษณะยาวและแหลมตลอดแนวอาณาเขตจะถูกลบออกจากโครงข่าย ยกตัวอย่างเช่นเมื่อโครงข่ายถูกล้อมรอบด้วยคอนเว็กซ์ฮัลของตัวเอง ดังรูปที่ 5.18



รูปที่ 5.18 สามเหลี่ยมซึ่งแหลมและยาวมักปรากฏอยู่บริเวณอาณาเขตเมื่อโครงข่ายถูกล้อมรอบด้วยคอนเว็กซ์ฮัล

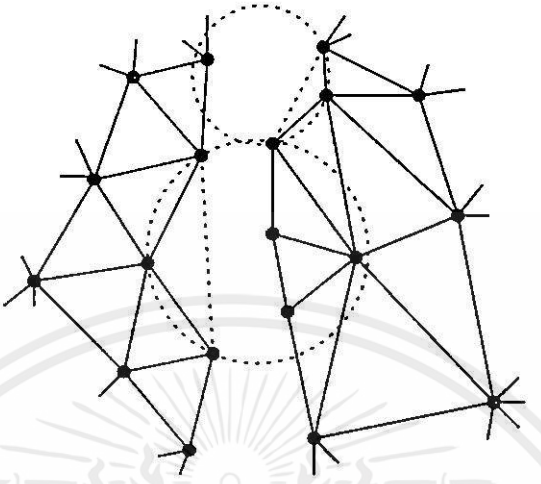
5.5.2 ขอบเขตจำกัดของพื้นที่ซึ่งได้รับผลกระทบจากการเชื่อมโยง

พื้นฐานสำหรับการเชื่อมโยงก่อตัวขึ้นจากโครงข่ายสามเหลี่ยม 2 โครงข่ายซึ่งมีคุณสมบัติสอดคล้องกับกฎเกณฑ์วงกลมทดสอบของดีลอนเนย์ (นิยามที่ 3) เมื่อกระบวนการเชื่อมโยงสำเร็จลงสามเหลี่ยมผลลัพธ์ต้องสอดคล้องกับกฎเกณฑ์วงกลมทดสอบเช่นกัน ในหัวข้อนี้จะอธิบายถึง τ_R ส่งผลกระทบอย่างไรต่อ τ_L และในทางกลับกัน τ_L จะส่งผลกระทบต่อ τ_R อย่างไร

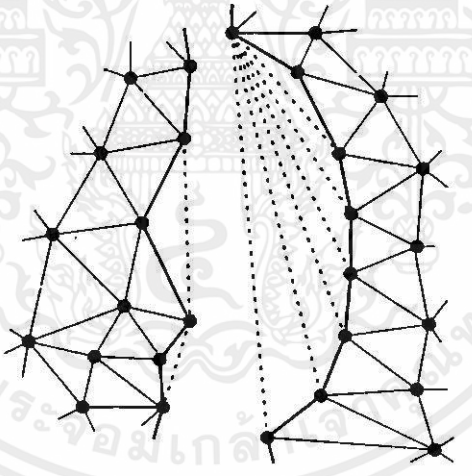
<u>ทฤษฎีที่ 8</u> กำหนดให้มีโครงข่ายสามเหลี่ยมซึ่งไม่ซ้อนทับกันอยู่ 2 โครงข่าย (τ_L และ τ_R) และโครงข่าย τ ผลลัพธ์ของการเชื่อมโยง ($\tau_L + \tau_R$) สามเหลี่ยมใดๆของ τ_L ซึ่งมีวงกลมทดสอบซึ่งบรรจุจุดอย่างน้อย 1 จุดของ τ_R จะไม่เป็นสมาชิกของ τ

พิสูจน์ กรณีที่เกิดขึ้นนี้เปรียบเทียบได้กับการค้นหาพื้นที่จำกัดของการสลับขอบ ดังเช่นในหัวข้อที่ 3.3.3.3 การตรวจสอบด้วยวงกลมทดสอบจะล้มเหลวถ้าสามเหลี่ยมตามแนวอาณาเขต

บรรจุจุดภายในวงกลมทดสอบดังรูปที่ 5.19 ดังนั้นจึงสรุปได้ว่าสามเหลี่ยมของ τ_L และ τ_R ซึ่งสอดคล้องกับกฎเกณฑ์วงกลมทดสอบใน τ เท่านั้นจึงจะเป็นสามเหลี่ยมที่แท้จริงของ τ



รูปที่ 5.19 สามเหลี่ยมซึ่งบรรจุจุดอื่นของโครงข่ายข้างเคียงไว้ภายในวงกลมทดสอบของตัวเองจะไม่ปรากฏใน τ

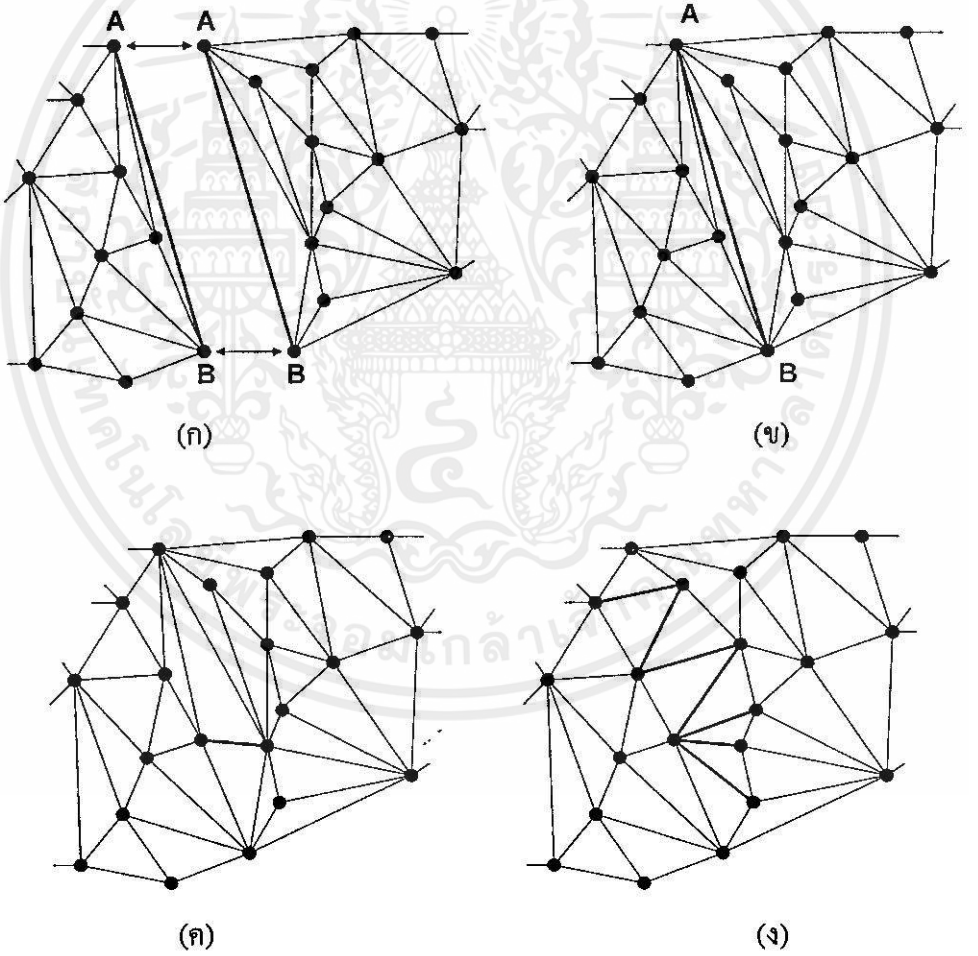


รูปที่ 5.20 ขอบเขตของสามเหลี่ยมซึ่งยังคงเหลืออยู่ใน τ (ถูกเน้นด้วยเส้นหนา)

ในรูปที่ 5.20 แสดงอาณาเขตของโครงข่าย τ_L และ τ_R ซึ่งมีอยู่ใน τ และยังแสดงให้เห็นอีกว่าการกระจายตัวของจุดมีผลกระทบอย่างไรกับสามเหลี่ยมซึ่งอยู่บริเวณอาณาเขตระหว่างโครงข่ายทั้งสอง เส้นอาณาเขตในรูปที่ 5.20 เปรียบเสมือนขอบเขตจำกัดของรูปหลายเหลี่ยมในรูปที่ 3.11 สามเหลี่ยมซึ่งผิดกฎดีลอนเนย์ใน τ_L และ τ_R สามารถถูกลบออกจากโครงข่ายก่อนที่จะมีการเชื่อมโยงเกิดขึ้นหรืออาจจะถูกปรับปรุงโครงสร้างให้ตรงกับ τ ได้

5.5.3 อัลกอริทึม Merge-and-Swap

ในหัวข้อนี้จะนำเสนออัลกอริทึมสำหรับเชื่อมโยงโครงข่ายสามเหลี่ยมดีลอนเนย์ 2 โครงข่ายเข้าหากันโดยที่ไม่ต้องลบหรือสร้างขอบขึ้นใหม่ระหว่างขบวนการเชื่อมโยงกำลังดำเนินไป ในอดีตนั้นอัลกอริทึมสำหรับการเชื่อมต่อโครงข่ายดีลอนเนย์ได้ถูกนำเสนอโดย [17] ซึ่งเป็นส่วนหนึ่งในอัลกอริทึม Divide-and-Conquer (หัวข้อที่ 2.4.3) และถูกนำมากล่าวถึงอีกครั้งใน [10] อัลกอริทึมนี้ถูกตั้งชื่อขึ้นใหม่เป็นอัลกอริทึม Delete-and-Build เพื่อให้สื่อความหมายของการทำงาน เมื่อสามเหลี่ยมบางส่วนซึ่งอยู่บริเวณอาณาเขต จะถูกกำจัดทิ้งโดยการลบขอบซึ่งผิดกฎของวงกลมทดสอบและขอบใหม่จะถูกสร้างขึ้นเพื่อเชื่อมจุดต่างๆของสองโครงข่ายเข้าหากัน ในขั้นตอนนี้เปรียบได้กับขั้นตอนของอัลกอริทึม Step-by-Step (หัวข้อที่ 2.4.4) ในการหาจุดที่สามเพื่อมาประกอบเป็นสามเหลี่ยมใหม่ แต่ในที่นี้เป็นสามเหลี่ยมซึ่งอยู่ระหว่างสองโครงข่าย



รูปที่ 5.21 (ก) ขอบร่วมของสองโครงข่ายซึ่งถูกคำนวณสามเหลี่ยมแยกกัน (ข) ขอบร่วมถูกเชื่อมเข้าหากัน (ค) ขอบแรกถูกสลับและขอบข้างเคียงทั้งสี่ต้องถูกทดสอบเพื่อการปรับโครงสร้าง (ง) โครงข่ายผลลัพธ์เมื่อการสลับขอบสิ้นสุดลง

สำหรับอัลกอริทึม Merge-and-Swap จะใช้การปรับโครงสร้างของขอบที่มีอยู่เดิมแล้วโดยการสลับขอบแทนการลบแล้วสร้างใหม่ หลักการนี้เป็นไปได้เนื่องจากขอบที่อาณาเขตของ τ_L และ τ_R นั้นบรรจบกันพอดี ดังนั้นขอบดังกล่าวจึงต้องครอบคลุมอาณาเขตระหว่าง τ_L และ τ_R เมื่อขอบอาณาเขตทั้งสองซึ่งยาวเท่ากันและถูกเชื่อมทับกันแล้วเสร็จกระบวนการปรับโครงสร้างของขอบบริเวณอาณาเขตก็จะเริ่มต้น ดังนั้นจุดทุกจุดตลอดแนวอาณาเขตคือมุมยอดของสามเหลี่ยมซึ่งไม่เว้นแม้แต่จุดซึ่งมีตำแหน่งบนเส้นอาณาเขตพอดี กรณีนี้เกิดขึ้นเมื่อมุมยอดทั้ง 3 มุมของสามเหลี่ยมเรียงอยู่บนเส้นตรงเดียวกันทำให้มีพื้นที่เท่ากับศูนย์ ซึ่งเรียกว่าสามเหลี่ยมศูนย์ดังที่ได้อธิบายไว้ในหัวข้อที่ 4.7.1 และขั้นตอนการทำงานของอัลกอริทึมอธิบายได้ดังนี้

1. อาณาเขตระหว่างสองโครงข่ายข้างเคียงถูกกำหนดขึ้นโดยเส้นตรงซึ่งเชื่อมจุด 2 จุด จุดทั้งสองต้องเป็นส่วนหนึ่งของทั้งสองโครงข่าย เส้นตรงซึ่งเชื่อมจุดสองจุดนี้ทำให้เกิดขอบอาณาเขตซึ่งซ้อนทับกันพอดีทั้งสองโครงข่าย (ดังรูปที่ 5.21(ก))
2. หลังจากที่ทั้งสองโครงข่ายต่างถูกสร้างเป็นโครงข่ายสามเหลี่ยมแล้วเสร็จ ขอบอาณาเขตจากทั้งสองโครงข่ายก็จะถูกเชื่อมติดกันเป็นหนึ่งเดียว (ดังรูปที่ 5.21(ข)) ขั้นตอนนี้กระทำได้ง่ายเมื่อใช้ TwinEdge ซึ่งอธิบายไว้ในหัวข้อที่ 4.4 เมื่อ TwinEdge ทั้งสองต่างก็กำหนดพอยเตอร์ twin ให้ชี้ซึ่งกันและกัน
3. ขอบอาณาเขตและขอบรอบข้างของรูปสี่เหลี่ยมด้านไม่เท่าจะถูกทดสอบผลรวมของมุมภายใน (หัวข้อที่ 3.2) เส้นทะแยงมุมของรูปสี่เหลี่ยมด้านไม่เท่ารูปแรกอาจต้องถูกสลับขอบ เมื่อเป็นเช่นนี้ด้านทั้งสี่ของรูปสี่เหลี่ยมด้านไม่เท่านี้และรูปสี่เหลี่ยมด้านไม่เท่าข้างเคียงก็จะต้องถูกทดสอบมุมภายในเพื่อการสลับขอบต่อไป (ดังรูปที่ 5.21 (ค))
4. ขั้นตอนที่ 3 จะถูกประมวลผลซ้ำแบบรีเคอร์สีฟจนกระทั่งไม่เหลือขอบให้สลับอีก (ดังรูปที่ 5.21(ง))

5.5.3.1 ข้อได้เปรียบและเสียเปรียบ

อัลกอริทึมสำหรับเชื่อมต่อโครงข่ายสามเหลี่ยมช่วยเพิ่มประสิทธิภาพทางด้านเวลาและความมีเสถียรภาพในการทำงาน ยิ่งไปกว่านั้นยังสามารถนำไปประยุกต์ใช้กับการเชื่อมต่อโครงข่ายสามเหลี่ยมสำหรับแบบจำลองความสูง (Generalized triangular network) เมื่อโครงข่ายถูกเชื่อมต่อจะยังคงมีจุดสำคัญจำนวนหนึ่งปรากฏอยู่ตามแนวอาณาเขต ดังนั้นกระบวนการเชื่อมต่อจึงต้องสามารถจัดการกับจุดที่ต้องเพิ่มเติมภายหลังนี้

ในอัลกอริทึม Delete-and-Build โครงสร้างข้อมูลพิเศษต้องถูกสร้างเพิ่มเติมสำหรับจัดการกับจุดภายในสามเหลี่ยมซึ่งถูกลบทิ้งก่อนที่จะมีการสร้างใหม่ตลอดแนวอาณาเขต เมื่อขอบใหม่ถูกสร้างขึ้นจุดภายในของสามเหลี่ยมใหม่จะถูกดึงออกจากโครงสร้างข้อมูลพิเศษนี้ สำหรับอัลกอริ

ทีม Merge-and-Swap จุดต่างๆถูกเก็บรวมไว้ในลำดับ (หรือ BSTree) ภายในสามเหลี่ยมแต่ละรูป (ดังหัวข้อที่ 5.2.1) โครงสร้างสำหรับการจัดเก็บจุดจะถูกปรับปรุงสำหรับการสร้างขอบแต่ละครั้ง ทำให้ไม่ต้องสร้างโครงสร้างข้อมูลใหม่ขึ้นมาจัดการเป็นพิเศษ แต่จะเกิดความเชื่องช้าบ้างเมื่อปริมาณของจุดบริเวณอาณาเขตมีมากเกินไป

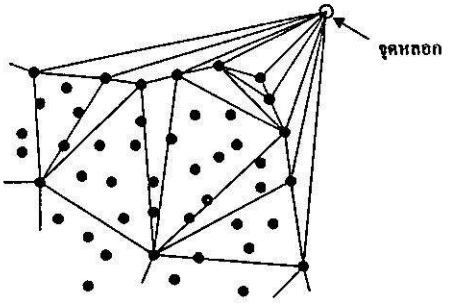
5.6 การเชื่อมต่อโครงข่ายสามเหลี่ยมสำหรับแบบจำลองความสูง

ในหัวข้อที่ 5.2 ได้อธิบายถึงการสร้างเค้าโครงของแบบจำลองพื้นผิวทำได้โดยการคัดเลือกจุดบางจุดซึ่งเป็นตัวแทนความโดดเด่น ณ บริเวณนั้นๆ เป็นไปได้ที่จะเชื่อมต่อสองโครงข่ายเค้าโครง τ_L และ τ_R เข้าหากัน ถ้าโครงข่ายต้นแบบทั้งสองเป็นชนิดสามเหลี่ยมดีลอนเนย์จะสามารถถูกเชื่อมต่อโดยใช้อัลกอริทึมในหัวข้อที่ 5.4 ได้ แต่ทั้งนี้ทั้งนั้น τ_L และ τ_R ต้องมีระดับการประมาณค่าเค้าโครง (λ) ที่เท่าเทียมกัน เนื่องจากการประมาณเค้าโครงจะถูกประมวลผลอีกครั้งสำหรับพื้นที่ซึ่งได้รับผลกระทบหลังการเชื่อมต่อ

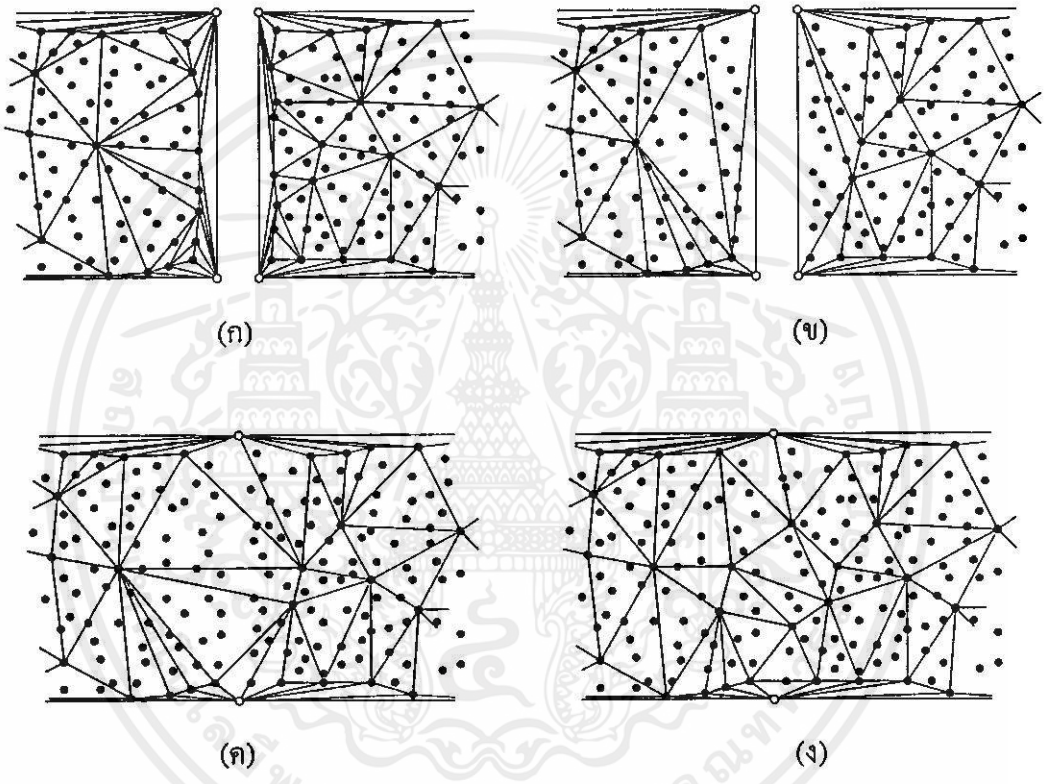
5.6.1 การประมาณเค้าโครงตามแนวเชื่อมต่อ

หลังจาก τ_L และ τ_R ถูกเชื่อมต่อกันแล้วเสร็จจะมีขอบจำนวนหนึ่งถูกปรับเปลี่ยนตำแหน่ง จุดซึ่งบรรจุอยู่ภายในสามเหลี่ยมซึ่งมีรูปร่างเปลี่ยนแปลงไปก็จะได้รับผลกระทบตามไปด้วย เมื่อระยะทางจากจุดเหล่านี้ไปสู่สามเหลี่ยมใหม่ซึ่งปิดล้อมจุดนี้อยู่จะเปลี่ยนไป ถ้าค่าระยะห่างของจุดใดมีค่ามากกว่าค่าเทร์สโฮลด์ (หัวข้อที่ 5.1.2) ที่ได้กำหนดไว้จุดนั้นจะถูกเพิ่มเข้าสู่โครงข่ายอีกครั้ง เนื่องจากสามเหลี่ยมใดซึ่งได้รับผลกระทบจากการเชื่อมต่ออาจบรรจุจุดที่ต้องถูกแทรกเพิ่มอีก ในทางกลับกันเมื่อโครงข่ายถูกเชื่อมต่อจะมีจุดบางจุดตามแนวเชื่อมต่อ ซึ่งมีส่วนร่วมกับแบบจำลองน้อยลง เป็นไปได้เช่นกันที่จะมีการตรวจสอบและกำจัดจุดเหล่านี้ออกจากโครงข่าย แต่การดำเนินการเช่นนี้สิ้นเปลืองเวลาเป็นอันมาก โครงข่ายจะไม่ผิดพลาดเพิ่มขึ้นถ้าจุดเหล่านี้ยังคงอยู่ แต่จะถูกต้องมากขึ้นด้วยซ้ำบริเวณอาณาเขตการเชื่อมต่อ ดังนั้นแบบจำลองข้อมูลจึงมีขนาดใหญ่เพิ่มขึ้นเล็กน้อย ในอัลกอริทึมที่จะนำเสนอต่อไปนี้จุดซึ่งอาจซ้ำซ้อนใน τ จะถูกกำจัดออกจาก τ_L และ τ_R ก่อนการเชื่อมต่อ

เมื่อโครงข่ายสองโครงข่ายถูกเชื่อมต่อโดยใช้อัลกอริทึม Merge-and-Swap ต้องมีจุดร่วม 2 จุดซึ่งปรากฏอยู่ในทั้งสองโครงข่าย จุดสองจุดนี้คือจุดปลายของขอบอาณาเขตระหว่างสองโครงข่าย ขอบนี้ขนานกับแกนหลัก (x หรือ y) ดังนั้นจุดหลอก (False points) ชั่วคราวซึ่งเป็นจุดร่วมระหว่าง τ_L และ τ_R จึงถูกกำหนดขึ้น ค่าความสูง (z) ของจุดหลอกถูกประมาณได้จากจุดข้างเคียง และสามเหลี่ยมซึ่งอยู่ติดกับจุดหลอกจะไม่บรรจุจุดใดๆไว้ภายใน (ดังรูปที่ 5.22) ขั้นตอนการเชื่อมต่อสองโครงข่ายมีดังนี้



รูปที่ 5.22 จุดหลอกถูกสร้างขึ้นเพื่อเป็นจุดร่วมของสองโครงข่าย



รูปที่ 5.23 (ก) โครงข่ายสองโครงข่ายถูกคำนวณสามเหลี่ยมแยกกัน (ข) จุดข้างเคียงของจุดหลอกถูกกำจัดออกจากโครงข่าย (ค) โครงข่ายถูกเชื่อมเข้าหากัน (ง) สามเหลี่ยมซึ่งได้รับผลกระทบถูกคำนวณสามเหลี่ยมซ้ำเพื่อเพิ่มความถูกต้องให้กับแบบจำลอง

1. ในรูปที่ 5.23(ก) τ_L และ τ_R ต่างก็คือโครงข่ายสามเหลี่ยมแทนแบบจำลองความสูงซึ่งกำลังจะถูกเชื่อมรวมกัน จุดหลอกทั้ง 4 จุดถูกสร้างขึ้น สองจุดสำหรับ τ_L และอีกสองจุดสำหรับ τ_R
2. ในรูปที่ 5.23(ข) เห็นได้ชัดว่ามีจุดหลายจุดซึ่งอยู่ใกล้กับอาณาเขตของ τ_L และ τ_R อาจจะซ้ำซ้อนใน τ ดังนั้นจุดค้างเคียงของจุดหลอกทุกจุดจะถูกกำจัดออกจากโครงข่ายก่อนที่จะทำการเชื่อมโยง

3. τ_L และ τ_R ถูกเชื่อมต่อโดยใช้อัลกอริทึม Merge-and-Swap ดังรูปที่ 5.23(ค)
4. จุดซึ่งมีตำแหน่งอยู่ภายในสามเหลี่ยมซึ่งได้รับผลกระทบจากการเชื่อมต่อจะได้รับการพิจารณาเพื่อคำนวณสามเหลี่ยมซ้ำ จุดซึ่งมีค่าระยะทางมากกว่าค่าเทร์สโฮลด์จะถูกแทรกเข้าสู่โครงข่ายอีกครั้ง ดังรูปที่ 5.23(ง)

อัลกอริทึมซึ่งได้อธิบายในหัวข้อที่ 5.3 จะถูกใช้สำหรับการลบจุดออกจากโครงข่าย จุดซึ่งถูกลบออกจะถูกจัดเก็บไว้ในสามเหลี่ยมใหม่ซึ่งปิดล้อมจุดนี้อยู่ จุดหลอกจะถูกลบออกเมื่อเสร็จสิ้นภาระกิจในการเชื่อมต่อโครงข่าย จุดบางจุดซึ่งมีส่วนร่วมน้อยลงกับแบบจำลองอาจยังคงอยู่ในโครงข่ายผลลัพธ์ถึงแม้ว่าได้มีการกลั่นกรองแล้วในขั้นตอนการลบจุดและการคำนวณสามเหลี่ยมซ้ำตามแนวเชื่อมต่อ อย่างไรก็ตามจุดที่หลงเหลือนี้มีจำนวนน้อยมาก และไม่มีความจำเป็นที่ต้องตรวจสอบหาความซ้ำซ้อนของจุดภายในโครงข่ายอีก

5.7 การทดลองเชื่อมต่อโครงข่ายแบบจำลองความสูง

เมื่อโครงข่ายสามเหลี่ยมถูกคำนวณได้จาก DEM ขนาดใหญ่มากๆ โดยการใช้อัลกอริทึม Incremental เพียงอย่างเดียวการทำงานจะค่อนข้างเชื่องช้ามาก อันมีสาเหตุเนื่องมาจากประสิทธิภาพโดยรวมในการคำนวณโครงข่ายสามเหลี่ยมจะขึ้นอยู่กับการถ่ายเทจุดระหว่างสามเหลี่ยมขนาดใหญ่ เมื่อจุดเหล่านี้ถูกจัดเก็บไว้ในลิสต์หรือในไบนารีเซิร์ชทรีภายในสามเหลี่ยมนั้นๆทำให้ไม่ต้องเสียเวลาเพิ่มเติมในการคำนวณเพื่อค้นหาว่าจุดนั้นๆอยู่ภายในขอบเขตของสามเหลี่ยมใด (หัวข้อที่ 5.2.1) แต่ด้วยวิธีนี้เมื่อข้อมูลมีขนาดใหญ่มากๆกลับทำให้อัลกอริทึม Incremental ทำงานช้าลง โดยเฉพาะช่วงเริ่มแรกของกระบวนการเมื่อพื้นที่ถูกครอบคลุมด้วยสามเหลี่ยมขนาดใหญ่

ในรูปที่ 5.24 และ 5.24 (ข) แสดงภาพขั้นตอนการคำนวณโครงข่ายสามเหลี่ยมจาก DEM โดยใช้อัลกอริทึม Incremental ในการคำนวณโครงข่ายสามเหลี่ยมแยกเป็นบล็อคๆ (รูปที่ 5.4(ก)) และใช้อัลกอริทึม Merge-and-Swap ในการเชื่อมต่อแต่ละบล็อคเข้าหากัน สามเหลี่ยมซึ่งอยู่บริเวณรอยต่อซึ่งได้รับผลกระทบจะได้รับการคำนวณสามเหลี่ยมใหม่ (รูปที่ 5.24(ข) และ 5.25(ก)) ในรูปที่ 5.25(ข) แสดงโครงข่ายสามเหลี่ยมซึ่งถูกคำนวณสามเหลี่ยมในขั้นตอนเดียวโดยไม่มีการแบ่งเป็นส่วนๆ (No partitioning Triangulation) เพื่อเปรียบเทียบกับโครงข่ายสามเหลี่ยมในรูปที่ 5.25(ก) ซึ่งถูกคำนวณโดยใช้วิธีการแบ่งส่วน (Partitioning Triangulation) เห็นได้ว่าโครงสร้างของทั้งสองโครงข่ายมีลักษณะใกล้เคียงกันในหลายๆส่วนของพื้นที่ และในรูปที่ 5.26 และ 5.27 จะแสดงภาพจำลองแบบมุมมองจริง (หัวข้อที่ 5.8.6) ของโครงข่ายสามเหลี่ยมซึ่งเกิดจากการคำนวณทั้ง 2 แบบ เพื่อเปรียบเทียบให้เห็นถึงเค้าโครงของภูมิประเทศที่เหมือนกัน

5.7.1 ผลการทดลอง

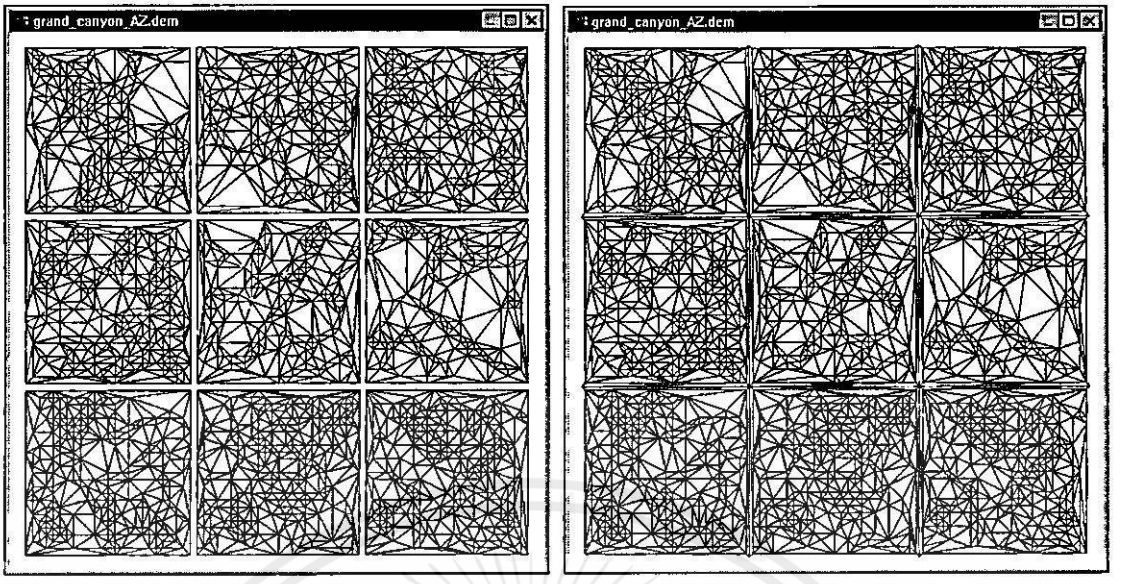
ในตารางที่ 5.3 และกราฟในรูปที่ 5.28 แสดงผลการทดลองของการคำนวณโครงข่ายสามเหลี่ยมจาก DEM ของพื้นที่บางส่วนของแกรนด์แคนยอน เมื่อกำหนดให้เทร์สโฮลด์คงที่ = 10 เมตร ขนาดของบล็อค = 20×20 จุด แล้วทำการเพิ่มขนาดของ DEM ให้ใหญ่ขึ้น จากผลการทดลองเห็นชัดได้ว่าการคำนวณโครงข่ายสามเหลี่ยมแบบแบ่งส่วนทำงานได้เร็วกว่า (หรือใช้เวลาน้อยกว่า) การคำนวณแบบไม่แบ่งส่วน ยิ่งขนาดของ DEM มีขนาดใหญ่ขึ้นเท่าใดก็จะยิ่งเห็นความแตกต่างมากขึ้นเท่านั้น การทดลองยังได้ปฏิบัติรวมไปถึงการค้นหาขนาดของบล็อคที่เหมาะสมที่สุด จากผลการทดลองซึ่งแสดงในรูปที่ 5.4 และกราฟในรูปที่ 5.29 ทำให้ทราบว่าบล็อคซึ่งมีขนาดเล็กจะใช้เวลาในการคำนวณน้อยกว่าบล็อคซึ่งมีขนาดใหญ่ และขนาดของบล็อคที่เหมาะสมคือ 20×20 จุด

ตารางที่ 5.3 เวลาที่ใช้ในการคำนวณเปรียบเทียบกันระหว่างการคำนวณสามเหลี่ยมแบบไม่แบ่งส่วนและแบบแบ่งส่วน เมื่อขนาดของ DEM เปลี่ยนแปลง (เทร์สโฮลด์ = 10 เมตร, ขนาดของบล็อค = 20×20, หน่วยเวลา = วินาที)

ขนาดของ DEM	40x40	60x60	100x100	180x180	300x300
คำนวณสามเหลี่ยมแบบไม่แบ่งส่วน (No partitioning triangulation)	1.322	3.395	10.495	56.011	306.881
คำนวณสามเหลี่ยมแบบแบ่งส่วน (Partitioning triangulation)	1.241	2.193	5.107	17.125	56.08

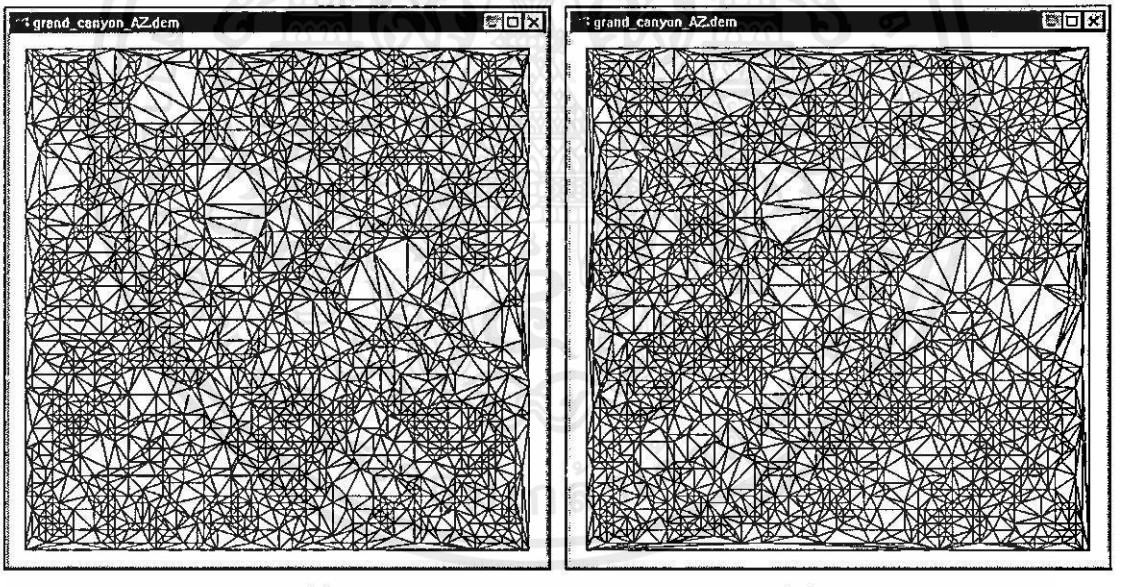
ตารางที่ 5.4 เวลาที่ใช้ในการคำนวณเมื่อขนาดของบล็อคเปลี่ยนแปลง(เทร์สโฮลด์ = 10 เมตร, ขนาดของ DEM 300×300, หน่วยเวลา = วินาที)

ขนาดของบล็อค	20x20	25x25	50x50	100x100	150x150
จำนวนของบล็อคต่อแบบจำลอง	225	144	36	9	4
(1) เวลาที่ใช้คำนวณสามเหลี่ยม	46.076	54.058	94.405	145.469	196.843
(2) เวลาที่ใช้เชื่อมต่อ	6.500	4.507	4.426	4.396	4.276
(3) เวลาที่ใช้คำนวณสามเหลี่ยมซ้ำอีกครั้ง	1.141	1.021	0.501	0.240	0.161
เวลารวม	53.717	59.586	99.332	150.105	201.28



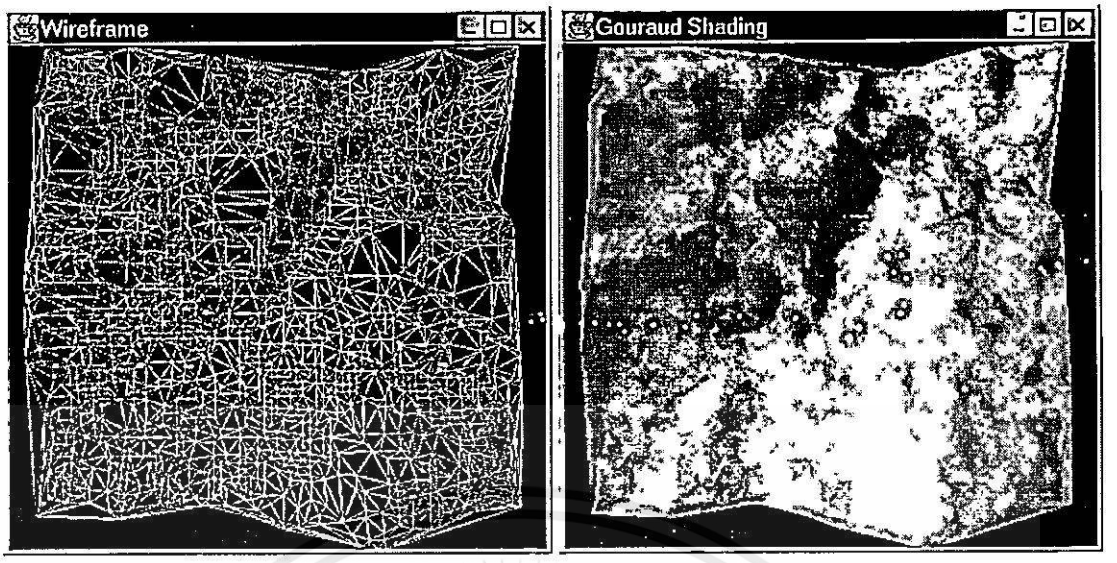
(ก) (ข)

รูปที่ 5.24 (ก) แต่ละบล็อคถูกคำนวณสามเหลี่ยมแยกจากกัน (ข) จุดหลอกที่มุมถูกสร้างขึ้นและขอบอาณาเขตข้างเคียงถูกเชื่อมเข้าหากัน



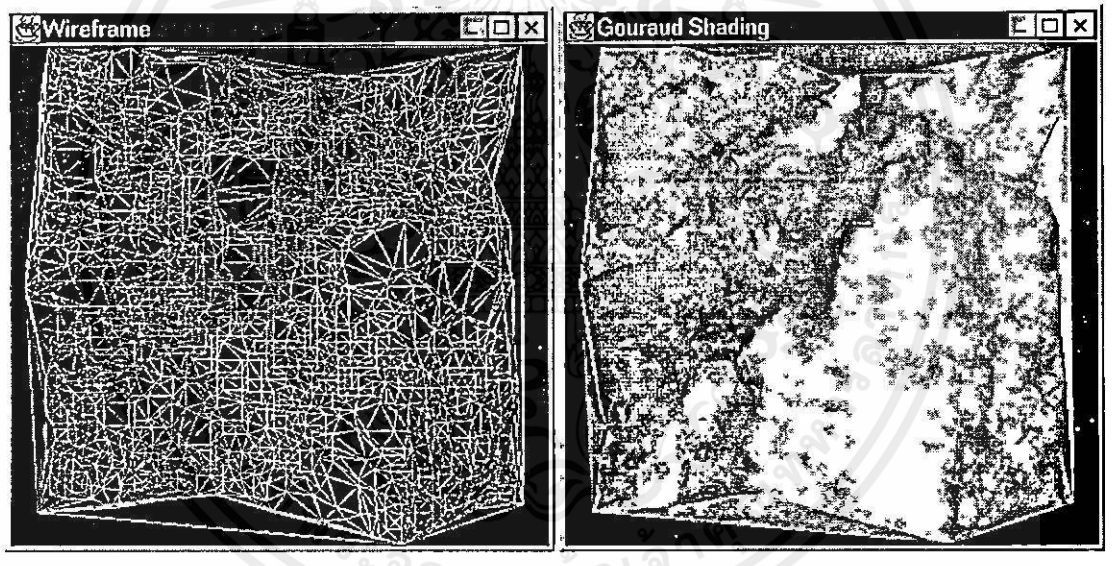
(ก) (ข)

รูปที่ 5.25 (ก) แต่ละบล็อคถูกเชื่อมหากัน สามเหลี่ยมซึ่งได้รับผลกระทบถูกคำนวณสามเหลี่ยมใหม่ (ข) โครงข่ายสามเหลี่ยมซึ่งถูกคำนวณเพียงครั้งเดียวโดยไม่มีการส่วน



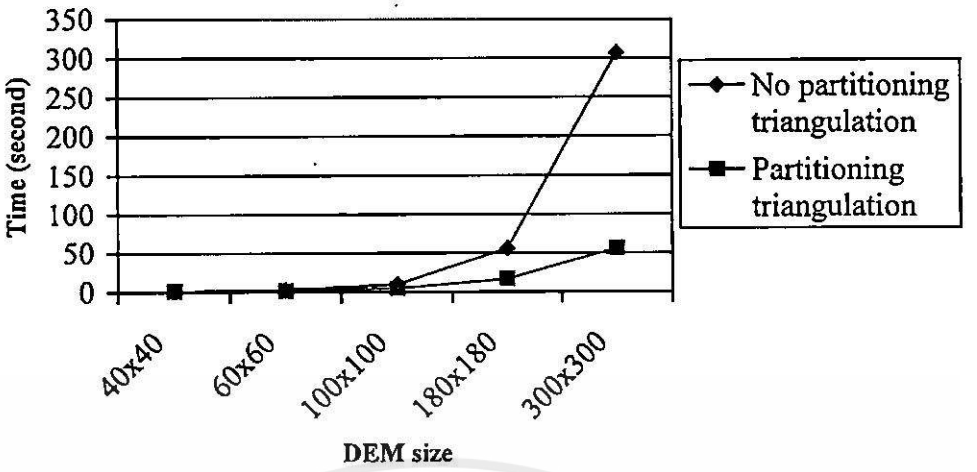
(ก) Wireframe (ข) Gouraud Shading

รูปที่ 5.26 โครงข่ายสามเหลี่ยมซึ่งถูกคำนวณโดยใช้วิธีการแบ่งส่วน

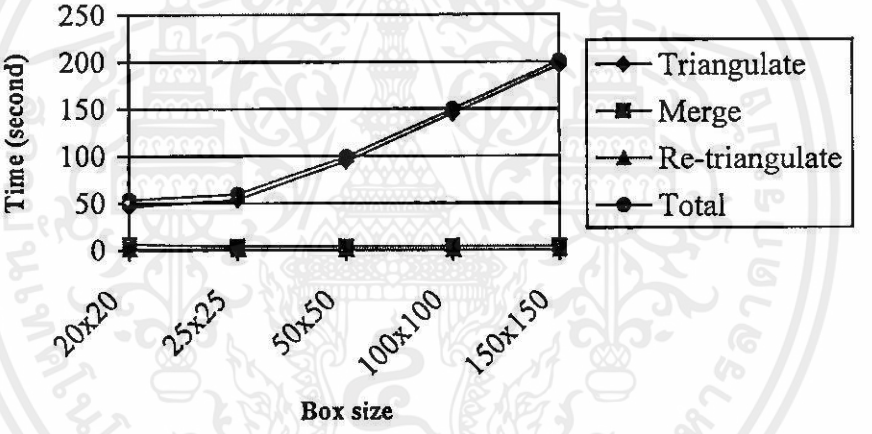


(ก) Wireframe (ข) Gouraud Shading

รูปที่ 5.27 โครงข่ายสามเหลี่ยมซึ่งถูกคำนวณเพียงครั้งเดียวโดยไม่มีการแบ่งส่วน



รูปที่ 5.28 เวลาที่ใช้ในการคำนวณเปรียบเทียบกันระหว่างการคำนวณสามเหลี่ยมแบบไม่แบ่งส่วนและแบบแบ่งส่วน (ข้อมูลจากตารางที่ 5.3)



รูปที่ 5.29 เวลาที่ใช้ในการคำนวณเมื่อขนาดของบล็อคเปลี่ยนแปลง (ข้อมูลจากตารางที่ 5.4)

5.8 การนำเสนอโครงข่ายสามเหลี่ยม

โดยพื้นฐานแล้วโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบเป็นโครงสร้างข้อมูลสำหรับจัดเก็บเค้าโครงของแบบจำลองพื้นผิว อย่างไรก็ตามการที่ TIN สามารถอธิบายรายละเอียดของพื้นผิวได้เป็นอย่างดีทำให้มันสามารถถูกนำมาประยุกต์ใช้ในการนำเสนอลักษณะของภูมิประเทศ ในหัวข้อนี้จะอธิบายวิธีการดั้งเดิมในการแสดงภาพลักษณะภูมิประเทศโดยใช้ TIN เป็นโครงสร้างเริ่มต้นซึ่งจะถูกแต่งเติมให้แสดงผลโดยใช้เทคนิคต่างๆ

5.8.1 แบบจำลอง Triangulated Irregular Network

พื้นที่บางส่วนของแกรนด์แคนยอนและภูเขาไฟเซนต์เฮเลนส์ถูกใช้เป็นตัวอย่างในการนำเสนอลักษณะภูมิประเทศในรูปแบบต่างๆ รูปที่ 5.31 แสดงโครงข่ายสามเหลี่ยมของพื้นที่ทั้งสองแบบจำลองความสูงชนิดกริด (DEM) ที่ใช้มีความละเอียดระหว่างจุดสำรวจความสูง = 30 x 30 เมตร เทร์สโฮลด์ที่ใช้ในการสร้างโครงข่ายสามเหลี่ยมมีค่า = 30 เมตร จำนวนจุดทั้งหมดของข้อมูลต้นฉบับเท่ากับ 152,382 (327 x 466) สำหรับภูเขาไฟเซนต์เฮเลนส์ และ 177,927 (381 x 467) สำหรับแกรนด์แคนยอน จากรูปทำให้เราสามารถแยกความแตกต่างระหว่างพื้นที่ที่เป็นที่ราบและพื้นที่ที่มีความซับซ้อนของสันเขาและร่องน้ำ

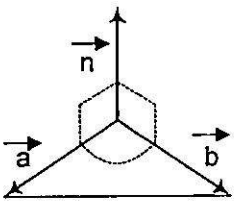
5.8.2 แผนที่เส้นความสูง

แผนที่เส้นความสูง (Contour map) สำหรับพื้นที่ทั้งสองแสดงในรูปที่ 5.32 เส้นความสูงถูกสร้างขึ้นโดยตรงจากแบบจำลอง TIN โดยการแทรกข้อมูลเชิงเส้น (Linear Interpolation) โดยไม่มีการทำให้เส้นตรงโค้งมนขึ้น แผนที่เส้นความสูงถือว่าเป็นวิธีการดั้งเดิมสำหรับการนำเสนอเค้าโครงของลักษณะภูมิประเทศ ถึงแม้ว่าแผนที่เส้นความสูงนี้จะไม่ใช่การนำเสนอแบบ 3 มิติ แต่ผู้ใช้จะสามารถเข้าใจและรับรู้ความซับซ้อนของภูมิประเทศได้ ทั้งนี้ทั้งนั้นก็ขึ้นอยู่กับประสบการณ์ของผู้ใช้แต่ละคน

5.8.3 แบบจำลอง Hill shading

การให้สีกับพื้นผิวของแบบจำลองเป็นเทคนิคการแสดงผลที่ทำให้ภาพจำลองของภูมิประเทศถูกมองเป็น 3 มิติ หลักการให้สีมีพื้นฐานมาจากแหล่งกำเนิดแสงในสามมิติส่องสว่างไปที่พื้นผิวของแบบจำลอง ความเข้มของแสงที่สะท้อนออกจากพื้นผิวจะแปรเปลี่ยนไปตามความลาดชันของพื้นผิว ทิศทางของแสงที่พุ่งออกจากแหล่งกำเนิดทำให้เกิดความแตกต่างในการรับรู้ของผู้ชม แหล่งกำเนิดแสงที่ถือว่าเหมาะสมที่สุดคือจากทิศตะวันตกเฉียงเหนือ (หรือมุมบนซ้าย) พุ่งตรงลงมาสู่แบบจำลองซึ่งจะให้ภาพที่เป็นเสมือน 3 มิติ มุมของแหล่งกำเนิดแสงควรเป็น 45° พุ่งลงมาสู่แบบจำลอง ใน [36] ได้แสดงวิธีการคำนวณมุมเอียงและทิศทางของระนาบสี่เหลี่ยมบนพื้นผิว สำหรับกรณีของระนาบสามเหลี่ยม มุมเอียงและทิศทางถูกกำหนดขึ้นโดยเว็กเตอร์ปกติ (Normal vector) หรือเว็คเตอร์ซึ่งตั้งฉากับระนาบ เว็คเตอร์ปกติของระนาบคำนวณได้จากการคูณไขว้ (Cross product) ระหว่าง 2 เว็คเตอร์ซึ่งแทนขอบทั้งสองของระนาบสามเหลี่ยมดังสมการที่ (5.3) รูปที่ 5.30 แสดงระนาบสามเหลี่ยมและเว็คเตอร์ปกติ

$$\vec{n}=\vec{a}\times\vec{b} \tag{5.3}$$



รูปที่ 5.30 เว็คเตอร์ปกติของระนาบสามเหลี่ยม

เมื่อ $\vec{n}$ คือเว็คเตอร์ปกติ และ $\vec{a}$ และ $\vec{b}$ คือขอบของสามเหลี่ยม จุดซึ่งแทนแหล่งกำเนิดแสงแทนด้วยเว็คเตอร์ $\vec{l}$ กำหนดให้ $\vec{l}$ = (1, -1, 1) เป็นแหล่งกำเนิดแสงที่เหมาะสมซึ่งอยู่ในตำแหน่งมุมซ้ายบนเหนือจากระนาบอ้างอิงของแบบจำลอง ระดับความเข้มของสีเทาของระนาบสามเหลี่ยมจะแปรผันเป็นสัดส่วนกับมุม θ ระหว่าง $\vec{n}$ กับ $\vec{l}$ (ดังสมการที่ 5.4) รูปที่ 5.33 แสดงภาพของแบบจำลองโดยใช้เทคนิค Hill shading

$$\cos\theta=\frac{\vec{n}\cdot\vec{l}}{|\vec{n}||\vec{l}|} \tag{5.4}$$

หลักการนี้เป็นเพียงการให้ระดับสีบนพื้นผิวของแบบจำลองเท่านั้น วัตถุขนาดใหญ่เช่น สันเขาจะไม่ถ่ายทอดเงาลงสู่พื้นผิวของพื้นที่ใกล้เคียง

5.8.4 แบบจำลองความชัน

ในรูปที่ 5.34 แสดงโครงข่ายสามเหลี่ยมโดยใช้เทคนิค Hill shading เช่นกัน แต่ต่างกันที่ทิศทางของจุดกำเนิดแสง เมื่อ $\underline{l}$ = (0, 0, -1) ซึ่งหมายความว่าแสงมีแหล่งกำเนิดจากด้านบนเหนือแบบจำลอง ภาพที่ออกมาจึงเป็นการนำเสนอความลาดเอียงของพื้นผิวแบบจำลอง พื้นราบแนวนอนจะเป็นสีขาวในขณะที่ระดับสีเทาจะเพิ่มขึ้นเมื่อระนาบมีความลาดชันมากขึ้น เทคนิคการแสดงผลนี้ถูกเรียกว่าแบบจำลองความชัน (Slope model)

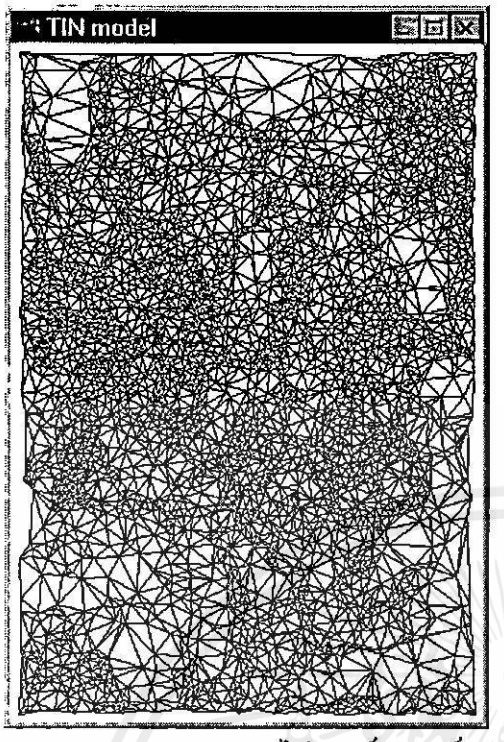
5.8.5 แผนที่ระดับความสูง

ในรูปที่ 5.35 แสดงแผนที่ระดับความสูง (Level map) ค่าความสูงถูกแบ่งออกเป็น 10 ระดับตามความเข้มของสีเทาที่แตกต่างกัน สีเทาจะเข้มขึ้นเมื่อความสูงเพิ่มขึ้น ความสูงเฉลี่ยของรูปสามเหลี่ยมแต่ละรูปถูกคำนวณเพื่อนำมาจัดระดับความสูง เทคนิคนี้สามารถนำไปประยุกต์ใช้งานกับการนำเสนอข้อมูลทางธรรมชาติอื่นๆ เช่น ปริมาณมลพิษ อุณหภูมิ ความหนาแน่นของป่าไม้ ปริมาณน้ำฝน ฯลฯ ยกตัวอย่างกรณีของการแสดงปริมาณมลพิษในอากาศ พื้นที่ซึ่งถูกแทน

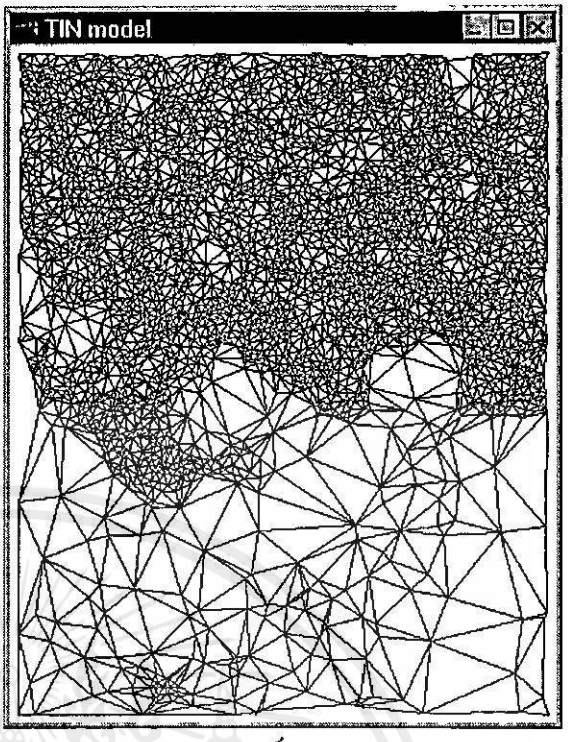
ด้วยสีดำหรือสีเทาเข้มจึงน่าจะเป็นบริเวณที่มีมลพิษอยู่สูง ส่วนพื้นที่เป็นสีขาวหรือเทาอ่อนก็น่าจะเป็นบริเวณที่มีปริมาณมลพิษต่ำ

5.8.6 แบบจำลองมุมมองจริงและการสร้างภาพแบบ Gouraud Shading

ภาพจำลองของภูเขาไฟเซนต์เฮเลนส์และแกรนด์แคนยอนถูกแสดงในรูปที่ 5.36 - 5.39 เป็นการสร้างภาพของแบบแบบจำลองแบบมุมมองจริง (Perspective view) [31] เมื่อมองจากด้านบนลงสู่แบบจำลอง (Arial view) พิกัดของจุดทุกจุดในโครงข่ายสามเหลี่ยมด้นแบบถูกถ่ายทอดจากพิกัดจริงเป็นพิกัดสำหรับแสดงผลในแบบ 3 มิติบนระนาบ 2 มิติบนจอภาพ โดยในรูปที่ 5.36 - 5.39 กลุ่ม (ก) เป็นการแสดงภาพ โครงสร้าง Wireframe ของ TIN และในรูปที่ 5.36 - 5.39 กลุ่ม (ข) ใช้เทคนิคการให้สีแบบกูราด (Gouraud Shading) [31] เพื่อเปลี่ยนแปลงภาพของระนาบสามเหลี่ยมบนพื้นผิวของแบบจำลอง TIN ที่แบบราบให้เป็นพื้นผิวที่โค้งมนเสมือนภาพของพื้นผิวภูมิศาสตร์ที่มีความต่อเนื่องจริงๆและมีแสงเงา ในรูปที่ 5.36 และ 5.37 แสดงภาพความแตกต่างของแบบจำลองภูเขาไฟเซนต์เฮเลนส์เมื่อใช้ค่าเทร์สโฮลด์ที่ต่างกัน (30 และ 50 เมตร) ในการคำนวณโครงข่ายสามเหลี่ยม เช่นกันที่ในรูปที่ 5.38 และ 5.39 แสดงภาพความแตกต่างของแบบจำลองแกรนด์แคยอนระหว่างการใช้เทร์สโฮลด์ 2 ค่า (30 และ 50 เมตร) ในรูปที่ 5.40 และ 5.41 แสดงภาพจำลองภูมิประเทศของภูเขาไฟเซนต์เฮเลนส์และแกรนด์แคนยอนในอีกมุมมองหนึ่งทำให้รับรู้ได้ถึงความซับซ้อนโดยรวมของภูมิประเทศ

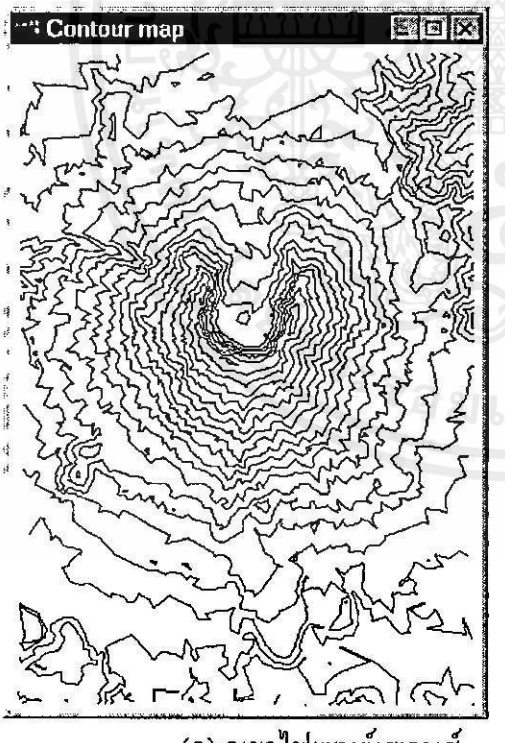


(ก) ภูเขาไฟเซนต์เฮเลนส์

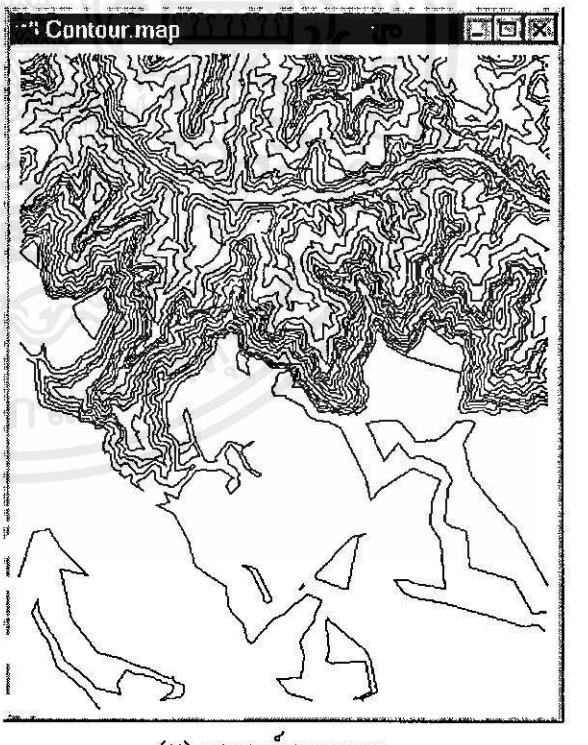


(ข) แกรนด์แคนยอน

รูปที่ 5.31 แบบจำลอง TIN

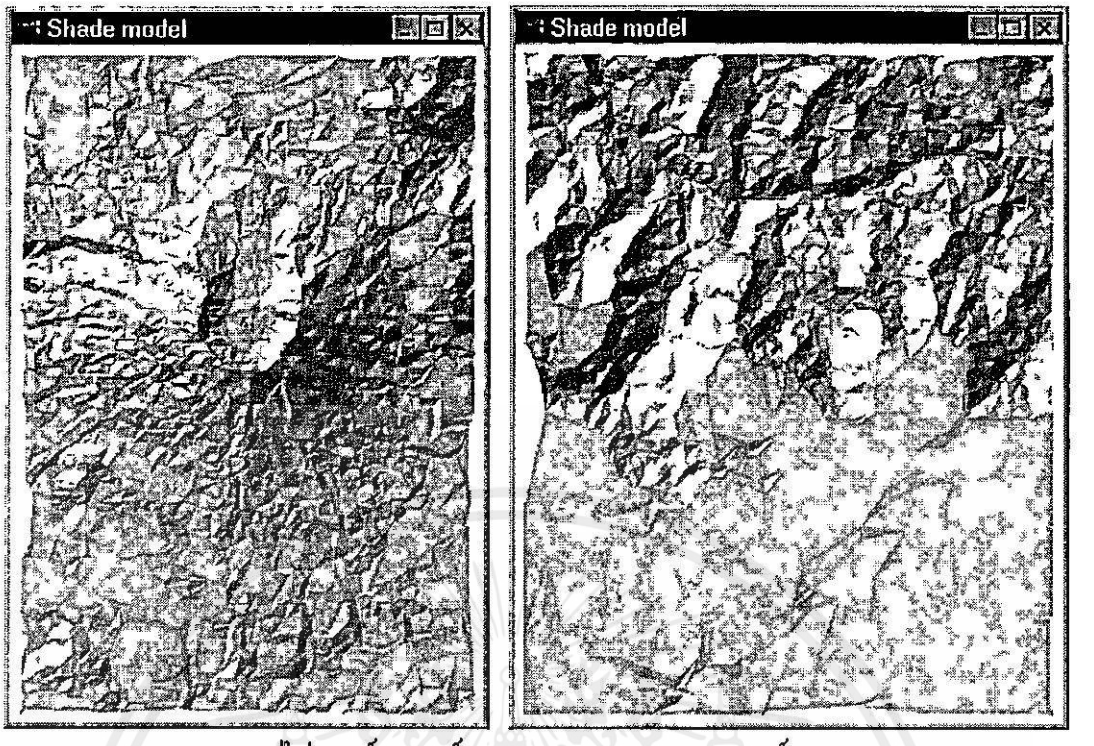


(ก) ภูเขาไฟเซนต์เฮเลนส์



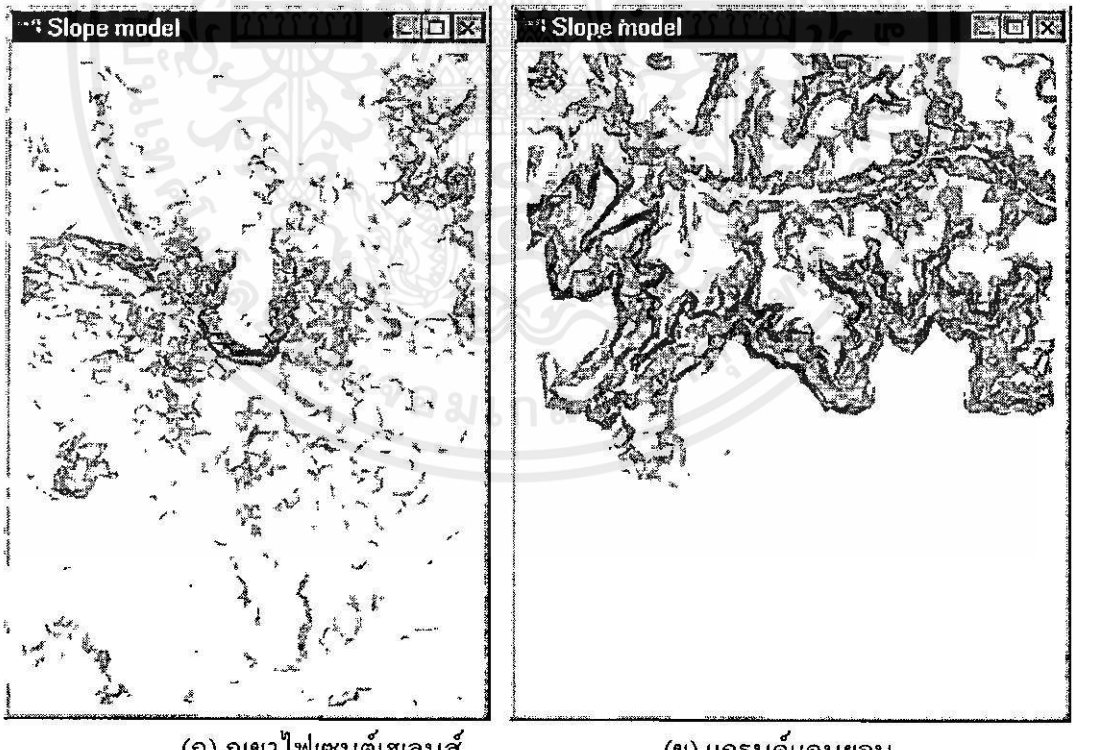
(ข) แกรนด์แคนยอน

รูปที่ 5.32 แผนที่เส้นความสูง



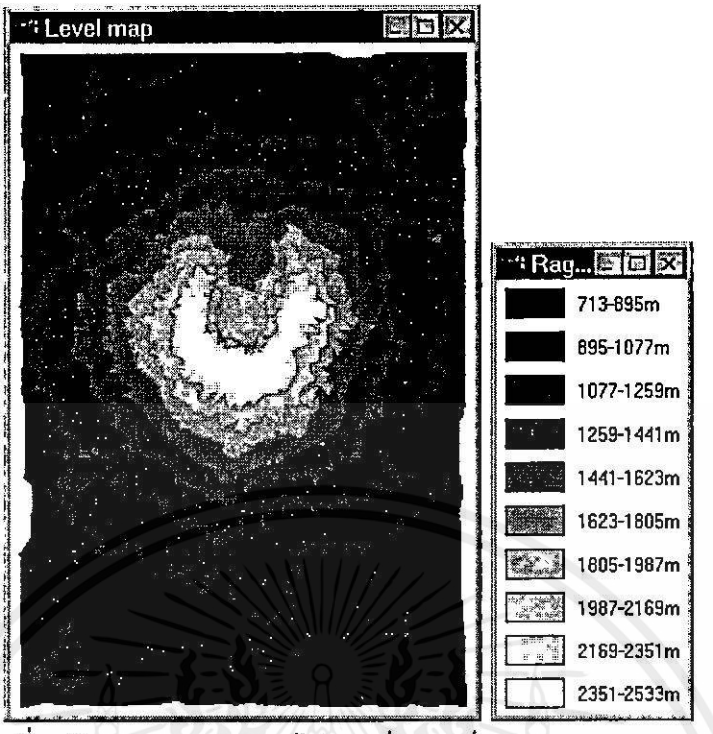
(ก) ภูเขาไฟเซนต์เฮเลนส์ (ข) แกรนด์แคนยอน

รูปที่ 5.33 แบบจำลอง Hill Shading

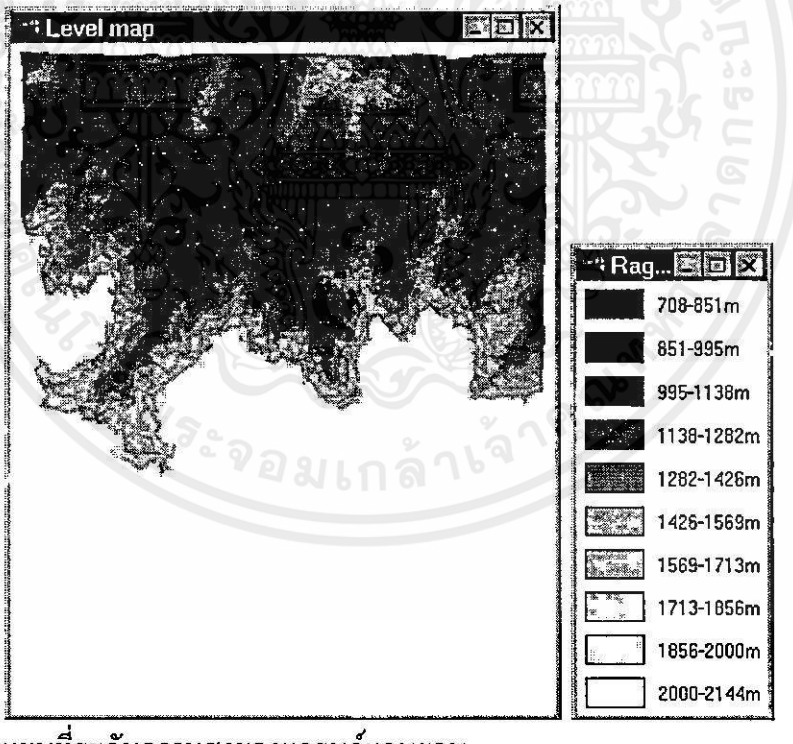


(ก) ภูเขาไฟเซนต์เฮเลนส์ (ข) แกรนด์แคนยอน

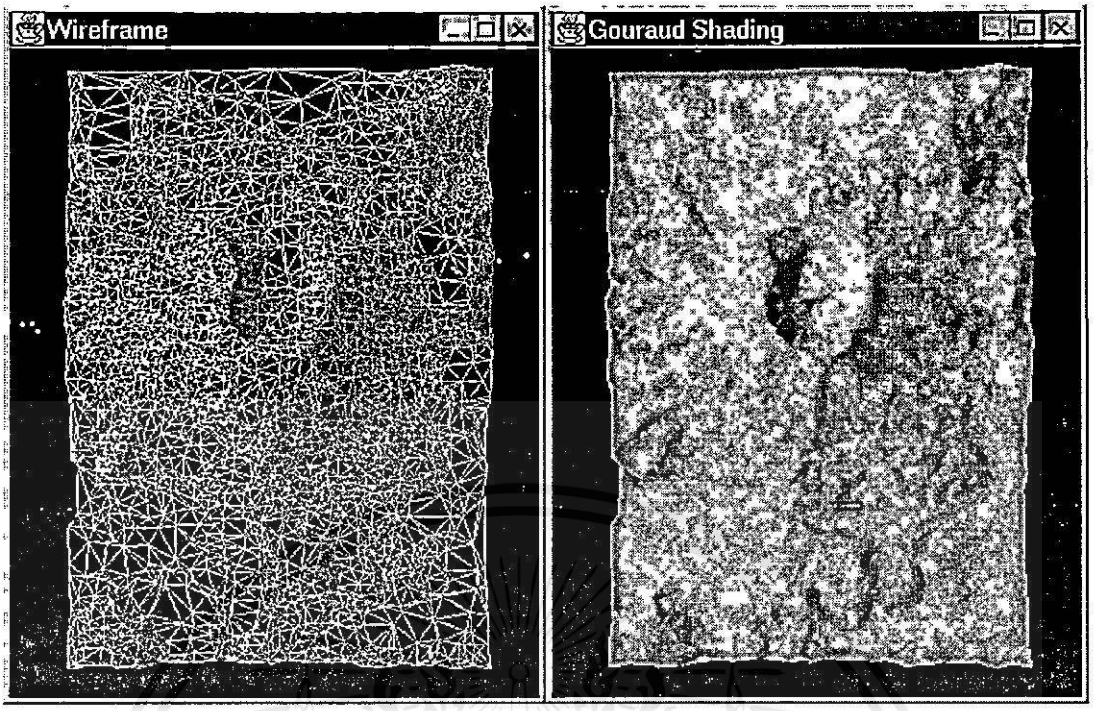
รูปที่ 5.34 แบบจำลองความชัน



รูปที่ 5.35 (ก) แผนที่ระดับความสูงของภูเขาไฟเซนต์เฮเลนส์

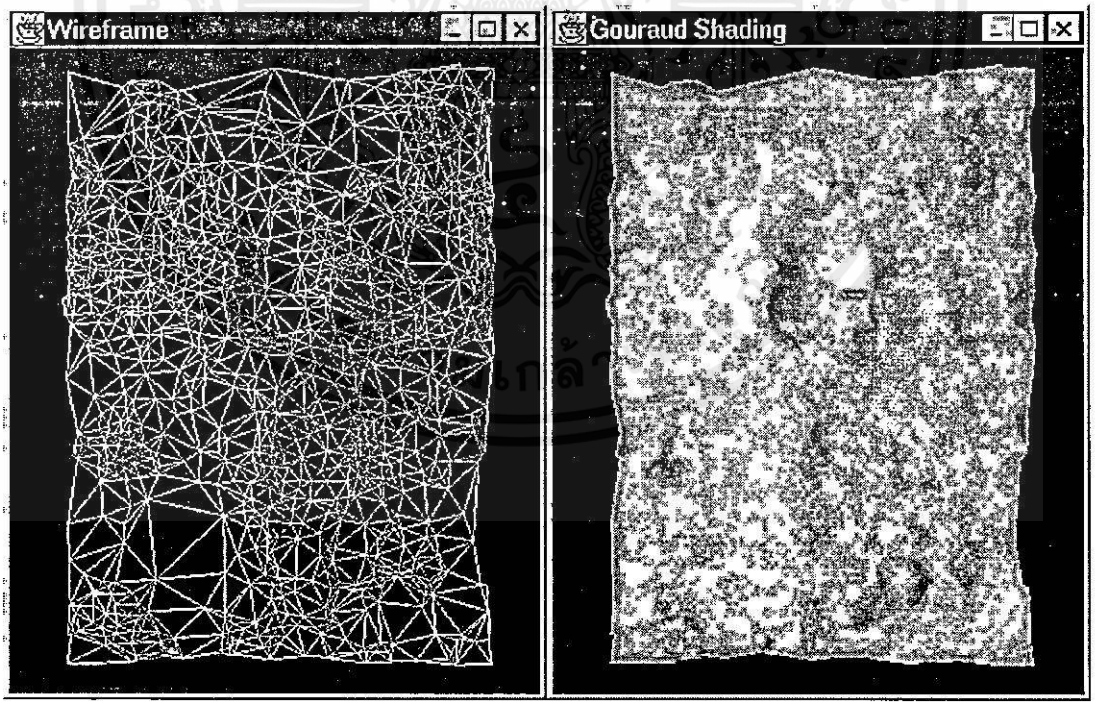


รูปที่ 5.35 (ข) แผนที่ระดับความสูงของแกรนด์แคนยอน



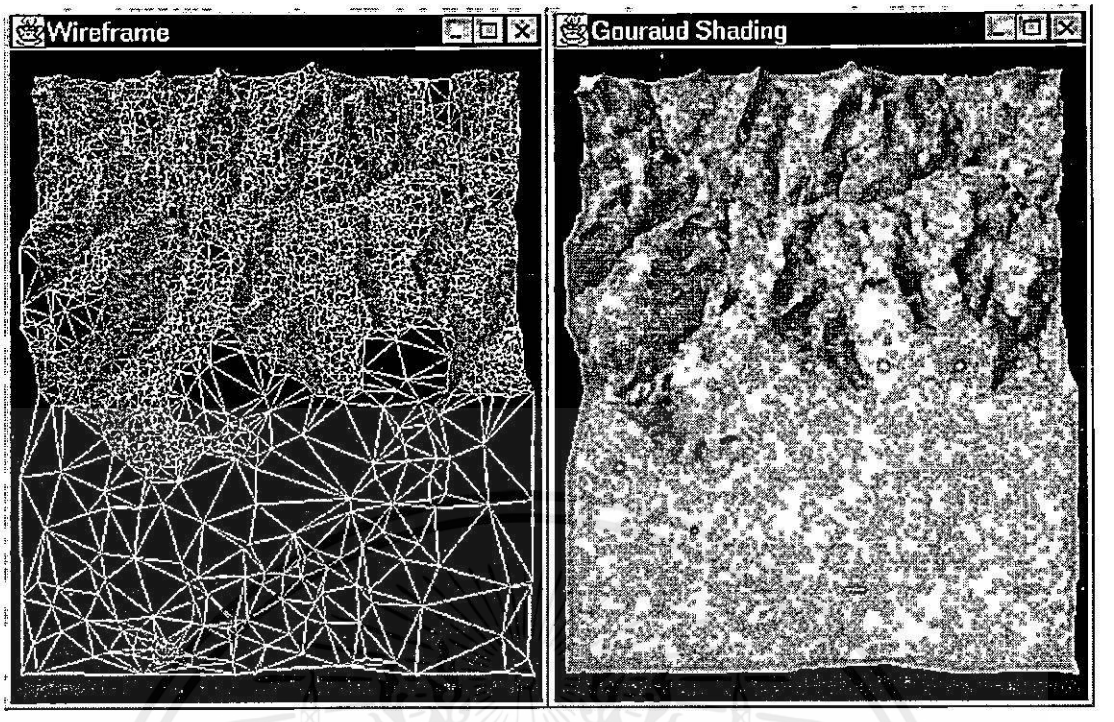
(ก) Wireframe (ข) Gouraud Shading

รูปที่ 5.36 ภาพจำลองจากมุมบนของภูเขาไฟเซนต์เฮเลนส์ เมื่อค่าเทร็สโฮลด์ = 30 เมตร และใช้สามเหลี่ยมจำนวน 11386 รูป



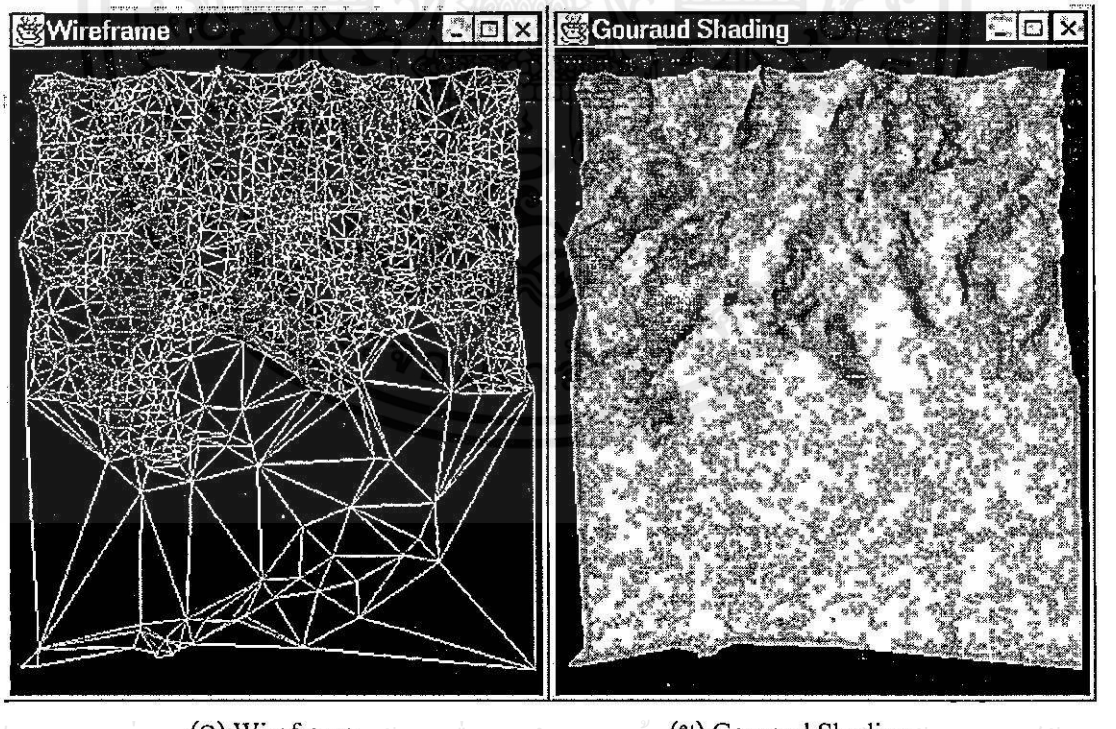
(ก) Wireframe (ข) Gouraud Shading

รูปที่ 5.37 ภาพจำลองจากมุมบนของภูเขาไฟเซนต์เฮเลนส์ เมื่อค่าเทร็สโฮลด์ = 50 เมตร และใช้สามเหลี่ยมจำนวน 3195 รูป



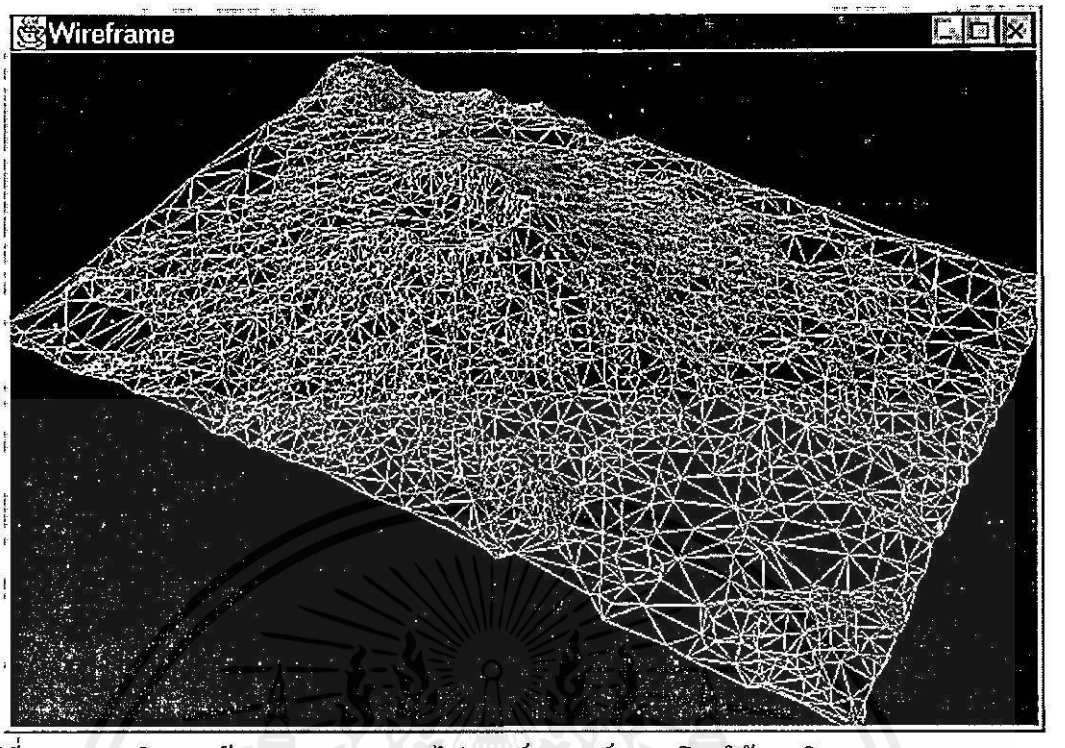
(ก) Wireframe (ข) Gouraud Shading

รูปที่ 5.38 ภาพจำลองจากมุมบนของแกรนด์แคนยอน เมื่อค่าเทร์สโฮลด์ = 30 เมตร และใช้สามเหลี่ยมจำนวน 10191 รูป

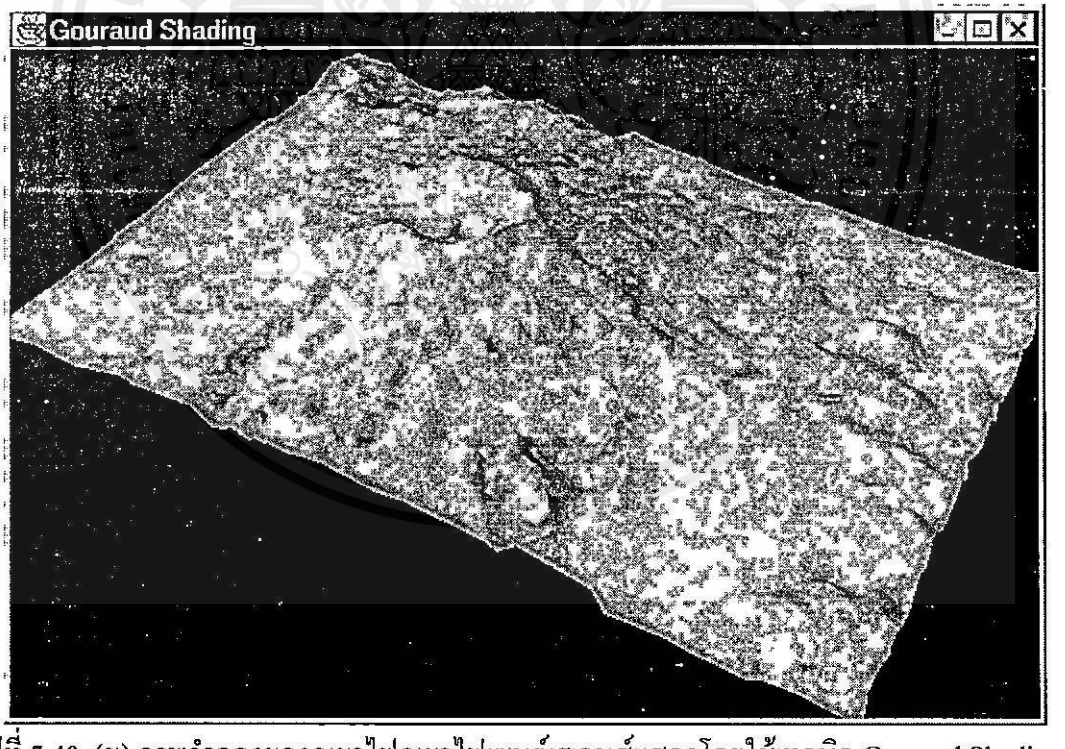


(ก) Wireframe (ข) Gouraud Shading

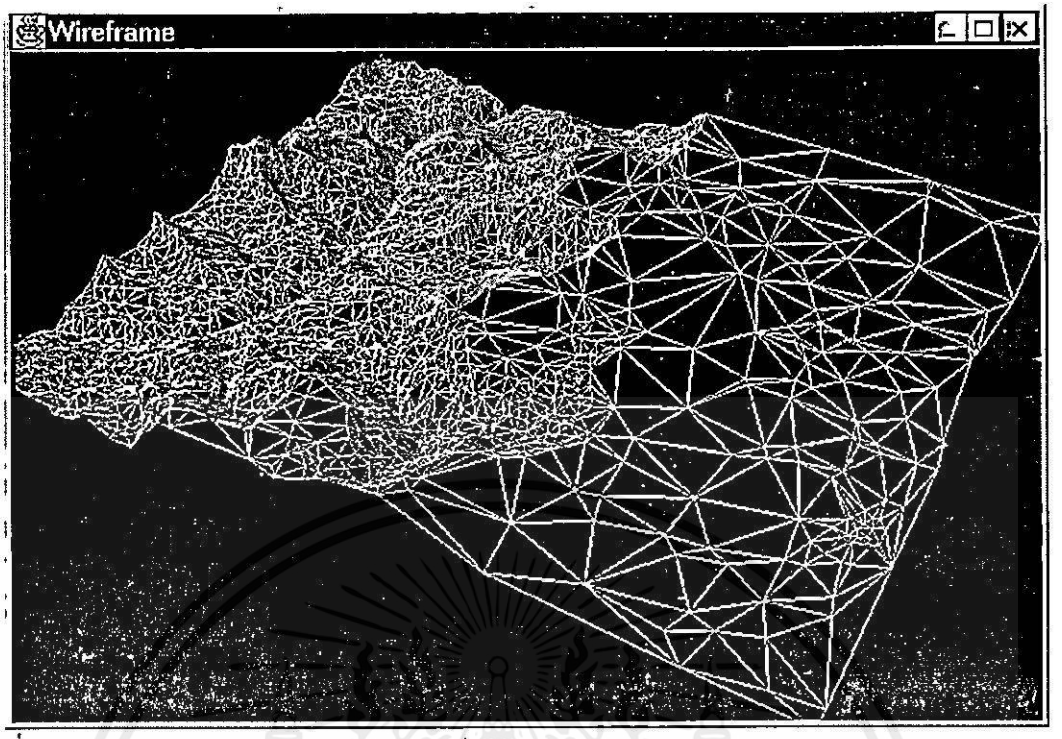
รูปที่ 5.39 ภาพจำลองจากมุมบนของแกรนด์แคนยอน เมื่อค่าเทร์สโฮลด์ = 50 เมตร และใช้สามเหลี่ยมจำนวน 4559 รูป



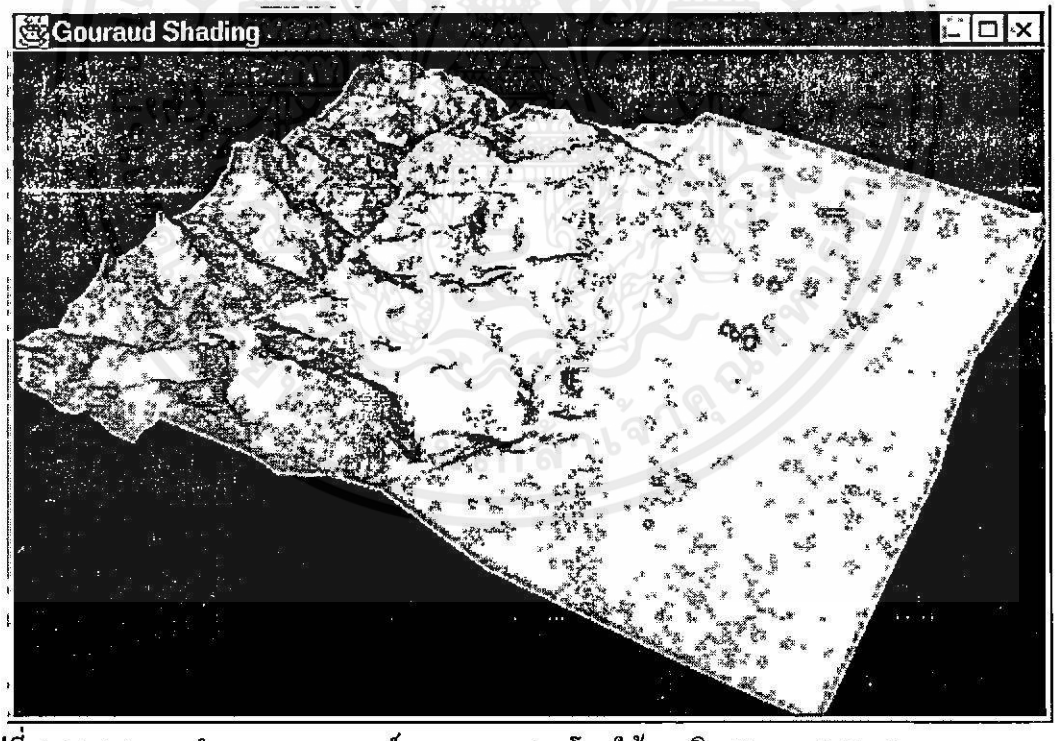
รูปที่ 5.40 (ก) โครงสร้างจำลองของภูเขาไฟเซนต์เฮเลนส์แสดงโดยใช้เทคนิค Wireframe



รูปที่ 5.40 (ข) ภาพจำลองของภูเขาไฟภูเขาไฟเซนต์เฮเลนส์แสดงโดยใช้เทคนิค Gouraud Shading



รูปที่ 5.41 (ก) โครงสร้างจำลองของแกรนด์แคนยอนแสดงโดยใช้เทคนิค Wireframe



รูปที่ 5.41 (ข) ภาพจำลองของแกรนด์แคนยอนแสดงโดยใช้เทคนิค Gouraud Shading

บทที่ 6
สรุปและวิจารณ์

วิทยานิพนธ์นี้ได้นำเสนอทฤษฎีและวิธีการในการสร้างโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบเพื่อใช้เป็นโครงสร้างพื้นฐานสำหรับการจำลองลักษณะภูมิประเทศ โครงข่ายสามเหลี่ยมชนิดดีลอนเนย์คือรูปแบบของโครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบที่ถูกนำมาใช้ซึ่งมีเอกลักษณ์เฉพาะ มีการจัดเรียงสามเหลี่ยมที่เป็นรูปแบบแน่นอนตายตัว รูปร่างของสามเหลี่ยมทุกรูปในโครงข่ายดีลอนเนย์จะมีลักษณะที่ใกล้เคียงกับสามเหลี่ยมด้านเท่ามากที่สุด

วิธีการสร้างโครงข่ายสามเหลี่ยมดีลอนเนย์จากเซ็ตของจุดมีอยู่หลายวิธี ในวิทยานิพนธ์ได้นำเสนอ 7 วิธีซึ่งถูกจัดแบ่งออกเป็น 2 กลุ่มหลักคือ กลุ่มวิธีการแบบสถิต และกลุ่มวิธีการแบบพลวัต วิธีการสร้างโครงข่ายสามเหลี่ยมแบบพลวัติมีข้อได้เปรียบกว่าวิธีการแบบสถิตคือสามารถเพิ่ม หรือลบจุด เข้าออกจากโครงข่ายได้ตลอดเวลา สามารถคัดเลือกจุดก่อนการนำเข้าสู่โครงข่ายและโครงสร้างภายในของโครงข่ายจะมีการจัดเรียงตัวใหม่หลังจากเกิดการเปลี่ยนแปลงเสมอ ทำให้วิธีการแบบพลวัติเหมาะสมเป็นอย่างยิ่งสำหรับการคำนวณเค้าโครงของภูมิประเทศ

อัลกอริทึม Incremental หนึ่งในวิธีการสร้างโครงข่ายสามเหลี่ยมแบบพลวัติถูกนำมาใช้เป็นหลักในการทดลองของวิทยานิพนธ์นี้ การสร้างโครงข่ายสามเหลี่ยมโดยใช้อัลกอริทึม Incremental ให้ถูกต้องตามนิยามของสามเหลี่ยมดีลอนเนย์ต้องสอดคล้องกับทฤษฎีพื้นฐานทางเรขาคณิตทั้ง 7 ข้อ หัวใจหลักของอัลกอริทึม Incremental คือการแทรกจุดและการปรับปรุงโครงสร้างโดยการสลับขอบซึ่งได้รับผลกระทบจากการแทรกจุดใหม่ การปรับปรุงโครงสร้างของโครงข่ายจึงเกิดขึ้นแบบท้องถิ่น

อัลกอริทึม Incremental จะไม่สามารถทำงานได้อย่างเต็มประสิทธิภาพถ้าขาดโครงสร้างข้อมูลที่เหมาะสม โครงสร้างข้อมูลแบบดั้งเดิมสำหรับจัดการกับโครงข่ายสามเหลี่ยมประกอบด้วยพอยเตอร์จำนวนมากทำให้การปฏิบัติงานมีความซับซ้อน และสิ้นเปลืองหน่วยความจำเป็นจำนวนมาก โครงสร้างข้อมูลชนิด TwinEdge จึงได้รับการออกแบบและปรับปรุงจากโครงสร้างข้อมูลแบบเดิมให้มีพอยเตอร์น้อยลงแต่สามารถครอบคลุมการทำงานที่จำเป็นได้ทั้งหมด รองรับการทำงานของอัลกอริทึม Incremental ได้เป็นอย่างดีในการแทรกจุด ง่ายต่อการปรับปรุงโครงสร้างของโครงข่าย รวมไปถึงการสำรวจโครงข่ายเพื่อนำเอาสามเหลี่ยมออกมาใช้ประโยชน์ เซ็ตของจุดได้รับการจัดเก็บไว้ภายในสามเหลี่ยมซึ่งปิดล้อมจุดนั้นๆอยู่ ทำให้ไม่ต้องมาเสียเวลาในการค้นหาว่าจุดที่กำลังจะถูกแทรกใหม่นั้นบรรจุอยู่ภายในสามเหลี่ยมใด

จากการทดสอบประสิทธิภาพการทำงานของอัลกอริทึม Incremental เมื่อเปรียบเทียบกับอัลกอริทึม Step-by-Step (วิธีการแบบสถิต) ทำให้ทราบว่าอัลกอริทึม Incremental ใช้เวลาในการ

คำนวณโครงข่ายสามเหลี่ยมกับเซ็ตของจุดแบบสุ่มน้อยกว่าอัลกอริทึม Step-by-Step มาก ความแตกต่างในเรื่องของความเร็วจะยิ่งเห็นผลชัดเจนเมื่อจำนวนของจุดมีมากขึ้น อัลกอริทึม Step-by-Step เป็นวิธีการพื้นฐานที่สุดในการสร้างโครงข่ายสามเหลี่ยม เหมาะที่จะใช้กับเซ็ตของจุดปริมาณไม่มากนัก

เมื่อเปรียบเทียบการจัดลำดับในการแทรกจุดเข้าสู่โครงข่ายระหว่างแบบก่อนหลัง และแบบสุ่ม จากการทดลองพบว่าลำดับการแทรกจุดแบบสุ่มทำงานได้เร็วกว่าแบบก่อนหลัง เนื่องจากตำแหน่งของการแทรกจุดมีการกระจายตัวไปตามพื้นที่ต่างๆของโครงข่าย อัลกอริทึม Incremental มีเสถียรภาพในการทำงานในการจัดการกับปัญหาที่เกิดขึ้น เช่นการจัดการกับสามเหลี่ยมศูนย์เมื่อจุดแทรกใหม่มีตำแหน่งซ้อนทับกับขอบเดิมโดยที่ไม่ต้องมีการลบขอบเดิมแล้วสร้างขอบใหม่ขึ้นแต่กระทำเพียงแค่การสลับขอบ

การทำงานหลายขั้นตอนอัลกอริทึม Incremental เป็นแบบรีเดอร์สีฟทำให้การคำนวณเป็นไปอย่างรวดเร็วเมื่อปัญหาขนาดใหญ่ถูกจับแบ่งย่อยๆในการจัดการดังเช่นการปรับปรุงโครงสร้างแบบรีเดอร์สีฟและการสำรวจโครงข่าย แต่อย่างไรตามปัญหาก็เกิดขึ้นเมื่อต้องสำรวจโครงข่ายซึ่งบรรจุสามเหลี่ยมอยู่เป็นจำนวนมากเกินที่การทำงานแบบรีเดอร์สีฟจะรองรับได้ จึงเกิดปรากฏการณ์สแต็คเกินล้น ในวิทยานิพนธ์ได้เสนอวิธีการแก้ปัญหาโดยใช้อัลกอริทึมแบบไม่เป็นรีเดอร์สีฟในการสำรวจโครงข่ายสามเหลี่ยม นอกจากนั้นวิธีการแบบรีเดอร์สีฟยังถูกนำมาประยุกต์ใช้กับการกำจัดสามเหลี่ยมศูนย์ และสามเหลี่ยมรูปลิ่มบริเวณอาณาเขตของโครงข่ายสามเหลี่ยมอีกด้วย สามเหลี่ยมทั้งสองสร้างปัญหาในขั้นตอนของการนำเสนอโครงข่ายสามเหลี่ยมในรูปแบบต่างๆทำให้เกิดภาพที่ไม่สวยงามและผิดจากรูปแบบที่ควรจะเป็น

จากหลักการเบื้องต้นของการประมาณค่าโครงเส้นโค้ง (2 มิติ) โดยใช้ดำลับของเส้นตรงทำให้เกิดแนวความคิดในการประมาณค่าโครงของพื้นผิว (3 มิติ) โดยใช้เซ็ตของระนาบสามเหลี่ยมในรูปของโครงข่ายสามเหลี่ยม การคัดเลือกจุดซึ่งมีระยะห่างมากที่สุดที่ตั้งฉากไปสู่ระนาบสามเหลี่ยมซึ่งปิดล้อมจุดนี้อยู่และมากกว่าค่าระยะห่างที่กำหนดไว้ (เทร์สโฮลด์) จุดนี้จะถูกคัดเลือกให้เป็นจุดแทรกใหม่เข้าสู่โครงข่ายสามเหลี่ยม การจัดการกับจุดภายในโครงข่ายได้รับการปรับปรุงให้ตอบรับกับหลักการดังกล่าวโดยใช้ไบนารีเสิร์ชทรีในการเก็บเซ็ตของจุด จุดซึ่งมีระยะห่างมากที่สุดสามารถถูกพบได้ทันทีที่โหนดขวาสุดของไบนารีเสิร์ชทรี จากผลการทดลองของการสร้างโครงข่ายสามเหลี่ยมจากข้อมูลความสูงแบบกริด (DEM) ทำให้ทราบว่าโครงข่ายสามเหลี่ยมช่วยลดปริมาณข้อมูลความสูงลงได้อย่างมาก เมื่อค่าเทร์สโฮลด์มีค่ามากขึ้นความซับซ้อนของโครงข่ายและจำนวนของจุดตัวแทนจะน้อยลง แต่อย่างไรก็ตามเค้าโครงโดยรวมจะยังคงอยู่

วิธีการสร้างและจัดการโครงข่ายสามเหลี่ยมแบบพลวัตทำให้ง่ายต่อการปรับปรุงโครงสร้างของโครงข่ายสามเหลี่ยม ในวิทยานิพนธ์ได้นำเสนอวิธีการสำหรับลบจุดออกจากโครงข่ายสามเหลี่ยม โดยใช้หลักการพื้นฐานของอัลกอริทึม Basis ในการค้นหาสี่เหลี่ยมด้านไม่เท่าซึ่งมี

ความยาวเส้นผ่านศูนย์กลางของวงกลมทดสอบที่น้อยที่สุด แล้วพัฒนามาเป็นอัลกอริทึม Reversion ซึ่งมีหลักการทำงานที่ย้อนรอยกับอัลกอริทึม Incremental เมื่อขอบต่างๆที่เคยถูกสลับมาก่อนหน้านี่จะถูกสลับกลับคืนสู่ที่เดิมดังเช่นก่อนที่จะมีการแทรกจุดใหม่ แล้วขอบทั้ง 3 ซึ่งเคยถูกสร้างขึ้นใหม่ก็จะถูกลบทิ้งในขั้นตอนสุดท้ายของการลบจุดออกจากโครงข่ายสามเหลี่ยม

เกิดความจำเป็นที่ต้องแบ่งพื้นที่ของแบบจำลองออกเป็นส่วนๆ แล้วคำนวณโครงข่ายสามเหลี่ยมแยกจากกันในกรณีที่แบบจำลองมีขนาดใหญ่มากๆ เนื่องจากการคำนวณโครงข่ายสามเหลี่ยมในครั้งเดียว จะเกิดความเชื่องช้าในขั้นตอนเริ่มต้นของการถ่ายเทจุดจำนวนมหาศาลระหว่างสามเหลี่ยมขนาดใหญ่ แบบจำลองความสูงจะถูกแบ่งออกเป็นบล็อคๆ แต่ละบล็อคถูกคำนวณโครงข่ายสามเหลี่ยมแยกจากกันโดยใช้อัลกอริทึม Incremental จากนั้นบล็อคทุกบล็อคจะถูกเชื่อมรวมกันโดยใช้อัลกอริทึม Merge-and-Swap สามเหลี่ยมบริเวณรอยตะเข็บของการเชื่อมต่อซึ่งได้รับผลกระทบจะถูกคำนวณสามเหลี่ยมซ้ำอีกเพื่อเพิ่มความถูกต้องให้กับแบบจำลอง จากผลการทดลองแสดงให้เห็นว่าการคำนวณโครงข่ายสามเหลี่ยมแบบแบ่งเป็นส่วนๆเร็วกว่าการคำนวณแบบครั้งเดียว ยิ่งบล็อคมีขนาดเล็กลงการคำนวณก็จะยิ่งใช้เวลาน้อยลง

ผลลัพธ์ของโครงข่ายสามเหลี่ยมที่สมบูรณ์จะถูกนำมาทำการสำรวจเพื่อดึงเอาสามเหลี่ยมทุกรูปในโครงข่ายออกมาใช้ในการจำลองแบบภูมิประเทศ ด้วยตัวของโครงข่ายสามเหลี่ยมเองเมื่อถูกแสดงผลในแบบ 2 มิติจะไม่สามารถอธิบายลักษณะของภูมิประเทศได้ ดังนั้นเทคนิคในการนำเสนอรูปแบบต่างๆโดยมีโครงข่ายสามเหลี่ยมเป็นพื้นฐานจึงถูกสร้างขึ้น อาทิ แผนที่เส้นความสูง แบบจำลอง Hill Shading แบบจำลองความชัน แผนที่ระดับความสูง การจำลองแบบโครงข่ายสามเหลี่ยมยังได้กระทำในแบบ 3 มิติ เมื่อแบบจำลองถูกแสดงผลในแบบมุมมองจริงและการให้สีกับแบบจำลองโดยใช้เทคนิค Gouraud Shading ทำให้แบบจำลองมีความสมจริงยิ่งขึ้น

เอกสารอ้างอิง

[1] B. G. Baumgart. "Geometric Modelling for Computer Vision." Ph.D. Dissertation, Stanford University. 1974.

[2] J. T. Bjorke. "Quadtrees and Triangulation in Digital Elevation Models." International Archives of Photogrammetry and Remote Sensing, International Society for Photogrammetry and Remote Sensing, Committee of the 16th International Congress of ISPRS, vol. 27, part B4, Commission IV, 1988. pp. 38-44.

[3] H. Blum. "A transformation for extracting new descriptors of shape." In Proceedings of the Symposium on Models for the Perception of Speech and Visual Form, 1967. pp. 365-380.

[4] B. Delaunay. "Sur la sphere vide." Bulletin of Academy of Science of the USSR, 1934. pp. 793-800.

[5] G. L. Dirichlet. "Uber die reduction der position quadratischen formen mit drei unbestimmten ganzen zalen." J. Reine u. Angew. Math., no. 40, 1850. pp. 209-227.

[6] D. H. Douglas, T. K. Peuker. "Algorithms for the Reduction of the Number of Points Reuired to Represent a Line or its Caricature." The Canadian Cartographer, vol. 10, no. 2, 1973. pp. 112-122.

[7] L. De Floriani, B. Falcidieno, G. Nagy and C. Pienovi. "A Hierarchical structure for surface approximation." Computer & Graphics, vol. 8, no. 2, 1984. pp. 183-193.

[8] L. De Floriani, B. Falcidieno and C. Pienovi. "Delaunay-based representation of surfaces defined over arbitrarily shaped domains." Computer Visions, Graphics and Image Processing, no. 32, 1985. pp. 127-140.

[9] P. J. Green, R. Sibson. "Computing dirichlet tessellations in the plane." The Computer Journal, vol. 21, no. 2, 1977. pp. 168-173.

[10] L. Guibas, J. Stolfi. "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams." ACM Transactions on Graphics, vol. 4, no. 2, 1985. pp. 74-123.

[11] M. Heller. "Triangulation algorithms for adaptive terrain modeling." In Proceedings of the 4th International Symposium on Spatial Data Handling, July 1990. pp. 163-174.

[12] E. Kreyszig. **Advanced Engineering Mathematics.** 7th ed. NY : John Wiley & Sons, Inc. 1993.

[13] M. J. Laszlo. **Computational Geometry and Computer Graphics in C++.** NJ : Prentice-Hall, Inc. 1996.

[14] C. L. Lawson. "Transforming triangulations." Discrete Mathematics, no. 3, 1972. pp. 365-372.

[15] C. L. Lawson. "Software for c^1 surface interpolation." In J. Rice editor, Mathematical Software III, Academic Press, New York , 1977. pp. 161-194.

[16] J. Lee. "Comparison of existing methods for building Triangular Irregular Network of terrain from grid Digital Elevation Models." International Journal of Geographical Information Systems, vol. 5, no. 3, 1991. pp. 267-285.

[17] D. T. Lee, B. J. Schachter. "Two algorithms for constructing a Delaunay triangulation." International Journal of Computer and Information Sciences, vol. 9, no. 3, 1980. pp. 219-242.

[18] B. A. Lewis, J. S. Robinson. "Triangulation of planar regions with applications." The Computer Journal, vol. 21, no. 4, 1978. pp. 324-332.

[19] M. J. McCullagh, C. G. Ross. "Delaunay triangulation of a random data set for isarithmic mapping." The Cartographic Journal, vol. 17, no. 2, 1980. pp. 93-99.

[20] A. Mirante, N. Weingarten. "The radial Sweep algorithm for constructing triangulated irregular network." *IEEE* computer Graphics and Applications, vol. 2, no. 3, 1982. pp. 11-21.

[21] G. Petrie, T. J. M. Kennie. "Terrain Modelling in Surveying and Civil Engineering." Whittles Publishing in association eith Thomas Telford Ltd, 1990.

[22] F. P. Preparata, M. I. Shamos. **Computational Geometry.** New York : Springer-Verlag. 1985.

[23] H. Samet. "The quadtree and related hierarchical and data structures." ACM Computing Surveys, vol. 16, no. 2, 1984. pp. 187-260.

[24] H. Samet. **The Design and Analysis of Spatial Data Structures.** MA : Addison Wesley. 1990.

[25] R. Sedgewick. **Algorithms in C++.** Singapore : Addison-Wesley Publishing Company, Inc. 1992.

[26] S. Saxena, P. C. P. Bhatt and V. C. Prased. "Efficient VLSI parallel algorithm for Delaunay triangulation on orthogonal tree network in two and three dimensions." *IEEE* Transactions on Computers, vol. 39, no. 3, 1990. pp. 400-404.

[27] A. J. Thiessen, J. C. Alter. "Precipitation averages for large areas." Monthly Weather Review, no. 39, 1911. pp. 1082-1084.

[28] United States Geological Survey. **Digital Elevation Models – Data Users Guide 5.** Virginia : Reston, U. S. Department of Interior. 1987.

[29] M. G. Voronoi. "Nouvelles applications des parametres continus a la theorie des formes quadratiques." J. Reine u. Angew. Math., vol. 134, no. 2, 1981. pp. 167-172.

[30] D. F. Watson. "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes." The Computer Journal, vol. 24, no. 2, 1981. pp.167-172.

[31] A. Watt. **Three-Dimensional Computer Graphics.** England : Addison Wesley. 1990.

[32] K. Weiler. "Edge-Based Data Structure for Solid Modeling in Curved-Surface Environments." *IEEE* Computer Graphics and Applications, vol. 5, no. 1, 1985. pp. 21-40.

[33] R. Weibel, Heller. "Digital Terrain Modelling, Geographical Information Systems - Principles and Applications." Longman, London, 1991. pp. 269-297.

[34] E. Wigner, F. Seitz. "On the construction of metallic sodium. Phys. Review." vol. 43, 1933. pp. 804-810.

[35] M. F. Worboys. **GIS A Computing Perspective.** London : Taylor & Francis. 1995.

[36] P. Yoeli. "An experimental electronic system for contours into hill shaded relief." In International Yearbook of Cartography, Kirschbaum Verlag, 1971. pp. 111-114.

ประวัติผู้เขียน

นาย สุลักษณ์ สุ่มมาตย์ เกิดเมื่อวันที่ 4 กันยายน พ.ศ. 2515 ที่จังหวัดร้อยเอ็ด สำเร็จการศึกษาอุตสาหกรรมศาสตร์บัณฑิต (อิเล็กทรอนิกส์) จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2537 เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ เมื่อปี พ.ศ. 2538

บทความวิชาการที่ได้รับการตีพิมพ์

1. สุลักษณ์ สุ่มมาตย์, กิตติ ไพฑูรย์วัฒนกิจ. "การจำลองภูมิประเทศแบบดิจิตอลโดยใช้โครงข่ายสามเหลี่ยมแบบไม่เป็นระเบียบ." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 21, สถาบันเทคโนโลยีพระจอมเกล้าธนบุรี, วันที่ 12-13 พฤศจิกายน 2541. หน้า 573-576.

2. Sulak Soommart, Kitti Paithunwattanakij. "Incremental Delaunay Triangulation Algorithm for Digital Terrain Modelling." Thammasat International Journal of Science and Technology (TIJSAT), vol. 4, no. 2, July 1999. pp. 65-76.

3. Sulak Soommart, Kitti Paithunwattanakij. "A Faster Method for Computing a TIN from a DEM Using the Block-Based Delaunay Triangulation Algorithm." การประชุมใหญ่ทางวิชาการ ประจำปี 2542, สมาคมวิศวกรรมสถานแห่งประเทศไทยฯ, วันที่ 1-2 พฤศจิกายน 2542.