การหารูปทรง 3 มิติจากภาพ 2 มิติ

RECOVERY OF THE 3-D SHAPE FROM SINGLE 2-D IMAGE

กาญจนา กาญจนพิบูลย์

KANCHANA KANCHANAPIBOON

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ มนัส สังวรศิลป์

ADVISOR

ASSOCIATE PROFESSOR MANUS SANGWORASILP

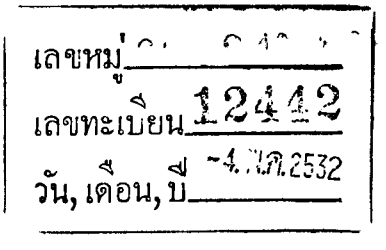
วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์มหาบัณฑิต

สาขาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2531



บทคัดย่อ

การคืนรูปทรง 3 มิติจากภาพ 2 มิติ คือการพิจารณาวิธีการคำนวณหาส่วนโค้ง และพื้นผิวที่ปรากฏบนภาพ 2 มิติ จากคอมพิวเตอร์กราฟฟิกเราได้พบว่าข้อมูล 3 มิตินั้นได้สูญเสียไปเนื่องจากการโปรเจคภาพลงบนระนาบ 2 มิติ และข้อมูลบางส่วนที่สูญเสียไปสามารถชดเชยได้ด้วยระดับความเข้มของแสงที่ปรากฏบนภาพวัตถุ ซึ่งสัมพันธ์กับทิศทางแหล่งกำเนิดแสงและตำแหน่งผู้สังเกต แต่การที่จะคืนรูปทรงของวัตถุจากภาพ 2 มิติเพียงภาพเดียว จำเป็นต้องกำหนดเงื่อนไขของภาพวัตถุนั้น เพื่อทำให้สามารถคำนวณหาลักษณะพื้นผิวได้ถูกต้อง

ดังนั้นวิทยานิพนธ์เล่มนี้กล่าวถึงการคืนรูปทรง 3 มิติจากภาพ 2 มิติเพียงภาพเดียวภายใต้เงื่อนไขว่าแหล่งกำเนิดแสงและตำแหน่งผู้สังเกตอยู่ในทิศทางเดียวกัน และพื้นที่ของวัตถุจะต้องมีความกลมกลืนต่อเนื่องกันไป ภาพที่ใช้ทดลองนั้นเป็นภาพที่เกิดจากการจำลองภาพวัตถุด้วยคอมพิวเตอร์กราฟฟิกและภาพวัตถุจริงตามธรรมชาติซึ่งถ่ายจากกล้องวิดิทัศน์ เราใช้ข้อมูลของภาพจากบริเวณที่เรียกว่า occluding boundary และอัตราความแตกต่างของระดับความเข้มแสงนำมาเป็นค่าเริ่มต้นในการคำนวณหาลักษณะพื้นผิวของวัตถุโดยวิธี iterative ผลของการคำนวณจะได้ความชัน f,g ซึ่งเป็นค่าบนระนาบ stereographic บ่งบอกลักษณะพื้นผิวเล็ก ๆ แต่ละตำแหน่งบนภาพวัตถุ เราใช้ทฤษฎีเรขาคณิตคำนวณระดับความสูงของพื้นผิวและเขียนรูปทรงของวัตถุที่คำนวณได้นั้นด้วยคอมพิวเตอร์กราฟฟิก

จากการกำหนดภาพ 2 มิติเพียงภาพเดียว ทำให้ข้อมูลพื้นผิวบริเวณที่ถูกบังจากสายตาไม่สามารถนำคืนกลับมาได้ รูปทรงของวัตถุที่คำนวณได้นั้นจึงเป็นค่าพื้นผิว $2\frac{1}{2}$ มิติเท่านั้น

ABSTRACT

Recovery of a 3-D shape of objects from a 2-D image is the determination of the 3-D spatial curves and surfaces appearing in an image. From computer graphics, depth information is lost by a projection to 2-D picture plane. We can define some of depth information from irradiance which depends on the position of a light source and viewpoint. However, in order to fully recover 3-D shape from only a single 2-D image, additional constraint is required for a unique solution.

This thesis presents a procedure for recovery of 3-D shapes from single 2-D images under restrictions that the light source is near the viewer and that surface is relatively smooth. The 2-D pictures used are those generated by a computer and taken by a camera. An iteration method for computing shape from shading using occluding boundary and gradient is proposed. The calculation give us both f and g, which are the reflectance function on stereographic plane. Using interpolation and computer graphics, 3-D images are produced.

Given a single picture, we can not recover hidden surfaces and the recovery shape of objects are all most 2½ model.

สารบัญ

สารบัญภาพประกอบ

หน้า

หน้า

หน้า

บทที่ 1

บทนำ

ในระบบการมองเห็นได้โดยคอมพิวเตอร์นั้น มีจุดประสงค์เพื่อให้เครื่องคอมพิวเตอร์สามารถที่จะวิเคราะห์เข้าใจในส่วนประกอบต่าง ๆ ของข้อมูลที่ได้มา พยายามที่จะยึดถือแนวทางที่จะเลียนแบบความฉลาดของมนุษย์ วัตถุที่นำมาเป็นข้อมูลสำหรับเครื่องคอมพิวเตอร์ อาจเป็นภาพวัตถุภายในห้องทดลอง ภาพตามธรรมชาติ ตัวอักษร หรือกลไกต่าง ๆ ในทางปฏิบัติการส่วนใหญ่จะเป็นภาพที่เกี่ยวข้องกับภาพ 2 มิติ ส่วนในระบบการมองเห็นของคอมพิวเตอร์ 3 มิตินั้น ส่วนประกอบต่าง ๆ จะเป็นส่วนประกอบบนแกน 3 มิติ อาจเป็นภาพจากความต่างระดับความเข้มของแสงสว่าง ภาพสี หรือภาพที่แสดงขอบเขตของข้อมูล

วิธีการในระบบภาพ 3 มิติถูกพัฒนาในแต่ละแนวทาง เช่น การขจัดสัญญาณรบกวน การแยกคุณลักษณะของภาพ การจดจำรูปแบบของวัตถุ ซึ่งในวิทยานิพนธ์ฉบับนี้เน้นถึงการวิจัยในระบบ 3 มิติ จากระดับความเข้มแสงและเส้นขอบภาพ เพื่อค้นหารูปทรงในโครงสร้าง 3 มิติของวัตถุที่ปรากฏในรูปภาพนั้น ในบทที่ 2 และบทที่ 3 เป็นการบรรยายถึงทฤษฎีต่าง ๆ ซึ่งเป็นพื้นฐานในการนำเข้าสู่ความเข้าใจ ส่วนในทฤษฎีบทที่ 4 เป็นข้อมูลที่ได้มาจากบทความที่ตีพิมพ์ในวารสารต่างประเทศ ซึ่งอธิบายถึงแนวความคิดซึ่งได้มีการทดลองและผลการทดลองด้วยวิธีการต่าง ๆ ในหัวข้อสุดท้าย 4.5 ของบทที่ 4 เป็นการสรุปผลจากการค้นคว้าและกำหนดวิธีการที่ใช้ในการทดลองสำหรับหัวข้อวิทยานิพนธ์นี้

บทที่ 2 ทัศนาการโดยคอมพิวเตอร์

เป็นการอธิบายการหาโครงสร้าง ในความหมายทางฟิสิกซ์ของวัตถุที่ปรากฏบนภาพการมองเห็นด้วยสายตามนุษย์ทางด้านชีววิทยา

ภาพซึ่งเกิดโดยธรรมชาติ เมื่อถูกนำมาใช้เป็นข้อมูลให้เครื่องคอมพิวเตอร์ ส่วนมากจะอยู่ในรูปของภาพ 2 มิติที่มีความต่างระดับของสัญญาณแสง ซึ่งเป็นค่าบ่งบอกระดับความสูงของวัตถุดังกล่าว ดังนั้นภาพเหล่านี้จะต้องถูกนำมาแปลงให้เป็นข้อมูลภาพด้วยฮาร์ดแวร์ สำหรับให้สามารถประมวลผลด้วยเครื่องคอมพิวเตอร์ได้

กระบวนการเกี่ยวกับภาพจะเริ่มต้นจากการขจัดสัญญาณรบกวน ซึ่งเป็นการรบกวนในสัญญาณภาพที่มีค่าความต่างระดับของสัญญาณความเข้มแสง ด้วยวิธีการที่จะทำให้ไม่เกิดการสูญเสียขอบ

ภาพ เพราะในระบบภาพ 3 มิตินั้นเส้นขอบภาพทุกเส้นจะให้ความหมายของข้อมูล 3 มิติหลายอย่าง และแตกต่างกันไปตามลักษณะการเกิดของขอบภาพนั้น สัญญาณขอบภาพที่ได้มักใช้เป็นข้อมูลเบื้องต้นในการเริ่มค้นหาค่ารูปทรง 3 มิติของรูปวัตถุ จากภาพวัตถุที่มีค่าความต่างระดับความเข้มแสง

บทที่ 3 คอมพิวเตอร์กราฟฟิก

คอมพิวเตอร์กราฟฟิกคือการสร้างรูปภาพในความหมายของคอมพิวเตอร์ เป็นแนวทางซึ่งย้อนกลับกับการมองเห็นด้วยระบบของเครื่องคอมพิวเตอร์ ข้อมูลที่นำมาใช้ป้อนเข้าในคอมพิวเตอร์กราฟฟิก เป็นภาพโครงสร้าง 3 มิติจากภาพจริงของวัตถุ ซึ่งภาพเหล่านั้นจะถูกนำมาสร้างใหม่เพื่อสื่อความหมายแทนรูปภาพจริง หรือผลการปฏิบัติการ ให้แก่บุคคลทั่ว ๆ ไป

จากทฤษฎีของคอมพิวเตอร์กราฟฟิกนี้ทำให้สามารถศึกษาโครงสร้างของวัตถุ ในระบบ 3 มิติ เพื่อหาความสัมพันธ์กับระบบภาพใน 2 มิติ เช่นภาพถ่ายจากกล้องถ่ายรูป หรือภาพวิดิทัศน์ จากการกำหนดโครงสร้างของมิติในระบบ 3 มิติ และจำลองภาพโดยกำหนดตำแหน่งแสง ตำแหน่งการมอง ภาพที่ถูกจำลองจึงมีความเหมือนภาพที่ปรากฏจริงของวัตถุตามธรรมชาติ

บทที่ 4 การหารูปทรงจากระดับความเข้มแสง

สามารถหาได้จากความสัมพันธ์ระหว่างความเข้มแสงที่ปรากฏในภาพ และรูปร่างทางเรขาคณิตของวัตถุ จากการศึกษาสมการของพื้นผิวระนาบ และ พื้นผิวที่มีความกลมกลืนต่อเนื่อง ลักษณะทางเรขาคณิตของความลาดชันของพื้นผิวมาสัมพันธ์กับคณิตศาสตร์ของความเข้มแสง จะได้เป็นค่าของ reflectance map ซึ่งบ่งบอกความเกี่ยวเนื่องของความลึกของภาพกับความเข้มแสงที่ตำแหน่งนั้น

แนวทางการวิเคราะห์ ซึ่งได้จากการโปรเจคของภาพหลายแบบ และศึกษาการแทนค่าพื้นผิวด้วยรูปทรงกลมเกาส์เซียน ซึ่งอธิบายถึงเวกเตอร์ของความสัมพันธ์ของสภาพพื้นผิวในความลาดชัน ความเอียงในระดับต่าง ๆ รวมทั้งการเกิดเส้นขอบของวัตถุที่ปรากฏบนภาพในเงื่อนไขที่สัมพันธ์กับลักษณะของแสง ซึ่งขอบภาพเหล่านี้จะให้ข้อมูลของพื้นผิวบริเวณขอบภาพ โดยเฉพาะขอบภาพที่เกิดจากการสัมผัสของพื้นผิวนั้นกับทิศทางของแสง

บทที่ 5 การทดลอง

การทดลองได้แบ่งออกเป็นสองขั้นตอน ในขั้นตอนแรกเป็นการจำลองภาพของวัตถุในลักษณะภาพกราฟฟิก 3 มิติบนจอภาพ 2 มิติ ด้วยหลักการของคอมพิวเตอร์กราฟฟิก กำหนดตำแหน่งของ

ภาพ เมื่อมีการเคลื่อนที่ของวัตถุจากแกนของมิติ การคำนวณเพื่อแสดงความสัมพันธ์ระหว่างทิศทางจุดกำหนดแสง แสงสะท้อน ตำแหน่งวัตถุ และเงาที่ปรากฏ โดยการเปลี่ยนตำแหน่งของจุดมอง แสง และวัตถุได้ทุกตำแหน่ง เน้นถึงการสร้างแสงและเงาบนภาพวัตถุให้เห็นความตื้นลึกของภาพ ประยุกต์วิธีการที่เหมาะสม เพื่อให้เกิดความต่อเนื่องของความโค้งและเว้าบนผิวของวัตถุได้ตามความเป็นจริง

จากนั้นจึงเริ่มพิจารณาขั้นตอนของการแปลงภาพ 3 มิติมาเป็นภาพ 2 มิติ ได้พบว่าเกิดการสูญเสียข้อมูลของโครงสร้าง 3 มิติไป เช่น บริเวณของภาพที่เกิดการถูกบังตามแนวทิศทางการมอง ซึ่งเป็นข้อมูลที่ไม่อาจนำคืนมาได้ไม่ว่าในกรณีใด ๆ และการสูญเสียข้อมูลเกี่ยวกับความลึกของวัตถุในขณะโปรเจคภาพลงบนระนาบ 2 มิติ แต่ข้อมูลเหล่านี้จะสามารถนำคืนมาได้โดยวิธีใด เพื่อที่จะคืนรูปทรง 3 มิติเหล่านี้ เราได้เลือกวิธีการที่จะนำเอาระดับความแตกต่างของสัญญาณแสง และคุณสมบัติของขอบวัตถุที่ปรากฏบนภาพมาเป็นข้อมูลในกระบวนการ จากการหาค่าความสัมพันธ์ระหว่างสมการแสงบนภาพวัตถุ กับ reflectance map ทำให้สามารถคำนวณหาค่าความลึกของวัตถุจากพื้นฉากคืนมาได้ ภาพที่ได้นำมาทดลองนั้นเป็นภาพซึ่งเกิดจากการจำลองภาพวัตถุ และภาพที่เกิดจากการถ่ายด้วยกล้องวิดิทัศน์จากวัตถุตามธรรมชาติ

บทที่ 6 บทสรุป

สรุปผลการทดลองตามทฤษฎีในบทที่ 5

บทที่ 2

ทัศนาการโดยคอมพิวเตอร์

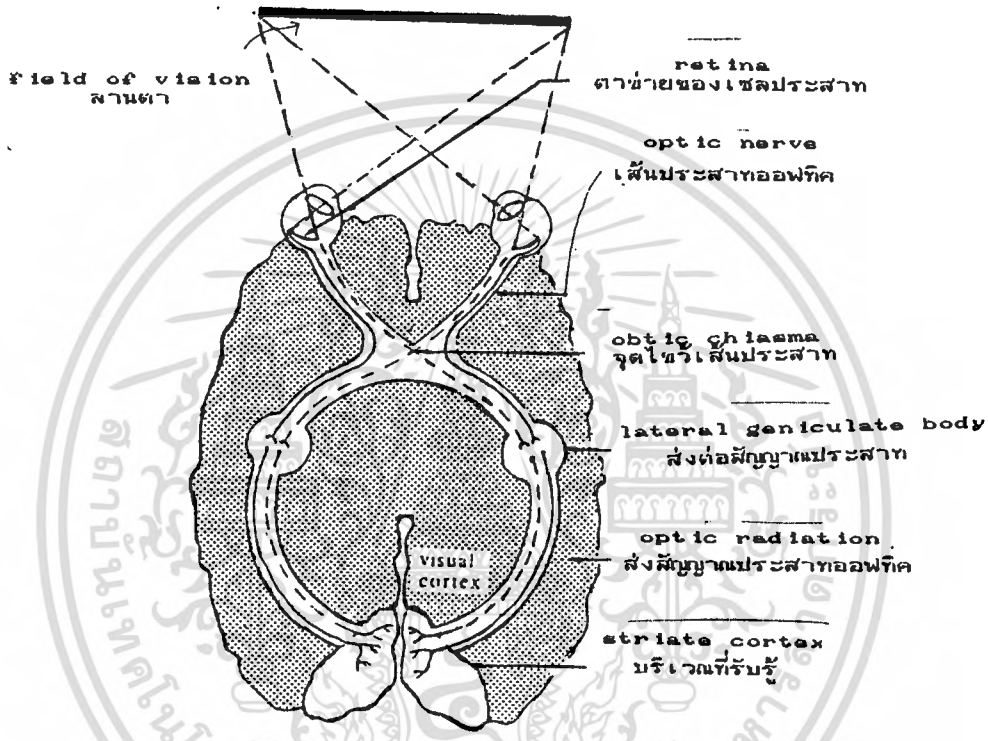
เทคนิคการใช้คอมพิวเตอร์ประมวลผลของรูปภาพ ได้ถูกนำมาพัฒนาและใช้ประโยชน์อย่างกว้างขวาง ถึงแม้ว่าเครื่องมือในการนำข้อมูลภาพเข้ามาวิเคราะห์จะมีความยุ่งยากซับซ้อน โดยทั่วไประบบของ image-proceccing ในคอมพิวเตอร์มีขั้นตอนการทำงานซึ่งเน้นถึง การเกิดภาพ วิธีการนำภาพเข้าประมวลผล การคำนวณเกี่ยวกับภาพ และผลที่ได้ การบรรยายในบทนี้เกี่ยวกับการเกิดภาพในลักษณะของฟิสิกซ์ ซึ่งสัมพันธ์กับการมองเห็นได้ของมนุษย์ ส่วนกระบวนการเกี่ยวกับภาพโดยคอมพิวเตอร์จะเน้นเฉพาะส่วนที่เกี่ยวข้องกับขอบภาพ เนื่องจากการหารูปทรง 3 มิติจากภาพ 2 มิตินั้นส่วนข้อมูลที่จำเป็นในการที่จะเริ่มต้นประมวลผลโครงสร้าง 3 มิติของภาพ คือข้อมูลจากลักษณะลายเส้นบนขอบภาพของวัตถุ ซึ่งรูปภาพต่าง ๆ ที่ถูกนำมาใช้เป็นภาพสำหรับการประมวลผลจากเครื่องคอมพิวเตอร์มีหลายประเภทด้วยคุณลักษณะต่าง ๆกัน เราอาจแบ่งการพิจารณาออกได้เป็น 2 กรณีคือจากแหล่งกำเนิดแสงและสภาวะทางเรขาคณิต

2.1 ธรรมชาติแห่งการมองเห็น[18]

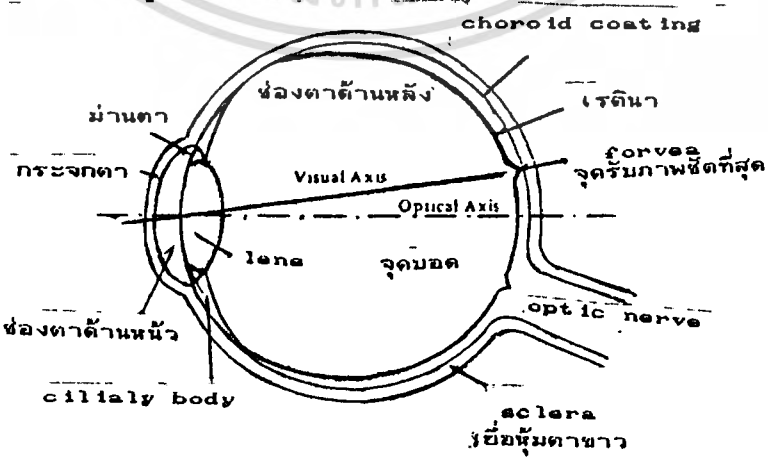
จากการศึกษากระบวนการความสามารถแห่งการมองเห็นได้ของมนุษย์ ทำให้เกิดความเคลื่อนไหวในแนวทางที่จะพัฒนาการมองเห็นได้ของคอมพิวเตอร์ งานใดที่ทำได้ด้วยความสามารถของมนุษย์ย่อมเป็นการยากที่จะกล่าวว่างานนั้นไม่สามารถทำได้ด้วยเครื่องกล หรือกล่าวอีกอย่างได้ว่าจากการศึกษาทางด้านกระบวนการของรูปภาพและการจดจำ มีบทบาทสำคัญที่ให้ความหวังในการสร้างประสิทธิภาพของเครื่องคอมพิวเตอร์ให้รับรู้สภาวะทางฟิสิกซ์ได้มากขึ้น

ในเนื้อเรื่องส่วนนี้เป็นการกล่าวสั้น ๆ ถึงคุณลักษณะของระบบการมองเห็นและพารามิเตอร์ซึ่งสำคัญในการออกแบบเทคนิคของกระบวนการรูปภาพ เนื่องจากผลของกระบวนการรูปภาพนี้ประเมินจากการสังเกตและการตัดสินใจ ดังนั้นจึงเป็นสิ่งสำคัญที่สร้างระบบการมองให้เหมือนจริงบนรูปภาพที่ได้ ขั้นแรกเราต้องพิจารณาถึงการมองเห็นโดยธรรมชาติอย่างแท้จริง หลักฐานจากการบรรยายของ Cornsweet (1970) ว่า แฟกเตอร์ที่สำคัญเกี่ยวกับคุณลักษณะของการมองคือฟิสิกซ์และสรีรศาสตร์ ระบบการรับภาพสามารถอธิบายได้ดีที่สุดด้วยรายละเอียดทางกายวิภาคของระบบการ

มองเห็น (VHS) แบ่งเป็นองค์ประกอบสี่ส่วน กล่าวคือ ดวงตา ประสาทเชื่อมโยงจากดวงตา ส่วน lateral geniculate body และส่วนของ visual cortex ดังแสดงในรูป 2.1 ฉากปรากฏบนเรตินาทั้งด้านซ้ายและขวา ผ่านกระบวนการทางเคมีภายในเปลี่ยนพลังงานแสงไปเป็นพัลซ์ทางไฟฟ้า พัลซ์เหล่านี้จะส่งผ่านไปสู่ optic chiasma ผ่าน lateral geniculate bodies และในที่สุดก็เข้าสู่บริเวณ visual cortex ใน opticpital lobe ของสมอง



รูป 2.1 โครงสร้างของระบบการมองเห็น

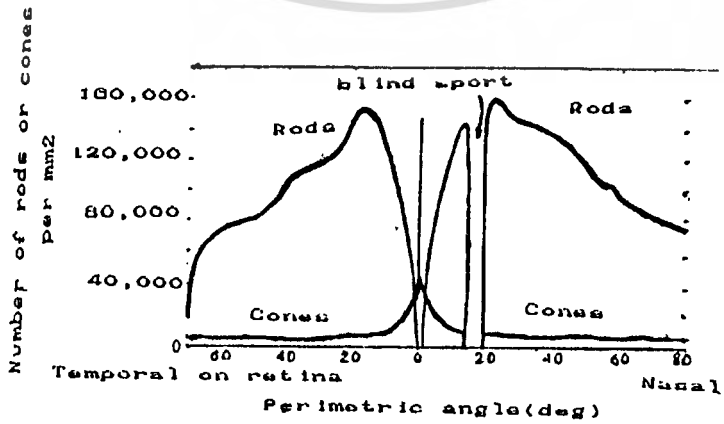


รูป 2.2 ภาพตัดขวางอย่างง่ายของดวงตามนุษย์

จากภาพตัดขวางดวงตามนุษย์ในรูป 2.2 เนื้อเยื่อทึบที่อยู่ชั้นนอกสุด sclera มีรูปร่างเกือบเป็นทรงกลมซึ่งมีรัศมีประมาณ 11 ม.ม และความหนา 1 ม.ม ที่ส่วนด้านหน้าของดวงตาเนื้อเยื่อ sclera จะกลมกลืนกันไปกับ cornea ซึ่งพองนูนมาด้านหน้าด้วยรัศมีประมาณ 8 ม.ม หนา 1 ม.ม ทางด้านหลัง optic nerve สอดเข้าไปในส่วน sclera จากทางด้านจมูก ส่วน ciliary body อยู่บริเวณด้านหลังจุดรอยต่อระหว่าง cornea และ sclera ด้านหน้าของ ciliary body เป็น Iris ซึ่งเป็นช่องเกือบเป็นรูปวงกลมและเป็นองค์ประกอบของแก้วตา ขนาดการเปิดของแก้วตาเปลี่ยนแปลงได้ประมาณ 2 ถึง 8 ม.ม. โดยการหดและคลายตัวของกล้ามเนื้อที่ควบคุม Iris ปรากฏเป็นการตอบรับความสัมพันธ์ระหว่างแสงและสรีรศาตร์

ภายในแก้วตานั้นเลนซ์นูนทั้งสองด้านลอยอยู่ในลักษณะที่เปลี่ยนรูปร่างได้ การเปลี่ยนรูปร่างได้นี้มีผลต่อการเปลี่ยนความยาวโฟกัสของเลนซ์ ทำให้สามารถปรับระยะทางของการมองเห็นได้ ช่องที่แยกจากกันโดยเลนซ์จึงเกิดขึ้น anterior chamber เป็นส่วนที่มีของเหลวคล้ายน้ำขังอยู่ ขณะที่ส่วนของ posterior chamber เต็มไปด้วยของเหลวที่มีความเหนียว ชั้นในสุดของดวงตาปกคลุมด้วยส่วนที่มีความไวต่อแสงคือเรตินา

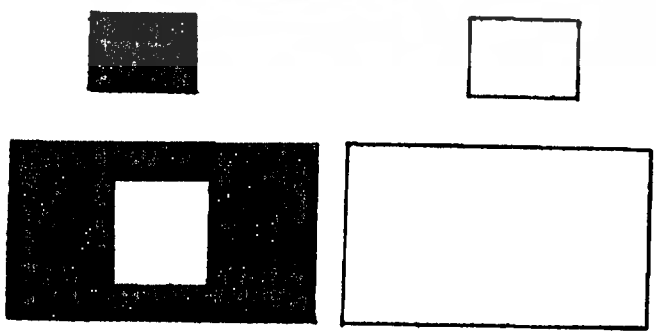
บนผิวเรตินาประกอบด้วยเซลรับแสงที่มีความหนาแน่นไม่เท่ากัน เรียกว่า rod และ cone ดังรูป 2.4 การกระจายตัวของมันแสดงดังรูป 2.3 ช่วงตรงกลางซึ่งมีความแคบประมาณ 0.4 ม.ม. เรียกว่า forvia ประกอบด้วยส่วนที่เป็น cone เท่านั้น เนื่องจากตำแหน่งและมีความหนาแน่นของ cone ทำให้เป็นส่วนที่รับสีดีที่สุด rod มีความหนาประมาณ 1-2 μm และยาว 60 μm สำหรับ



รูป 2.3 การวางตัวของ rod และ cone ที่ระยะทางห่างจากศูนย์กลางของ fovea

รูป 2.4 แสดง rod (ซ้าย)และ cone (ขวา)

cone มีขนาดแตกต่างกันไป แต่ส่วนที่อยู่บริเวณศูนย์กลางของ fovea มีความหนาประมาณ 1-2 μm เช่นเดียวกัน จำนวนของ cone มีประมาณ 5-7 ล้านและจำนวน rod ประมาณ 75-150 ล้าน ใน

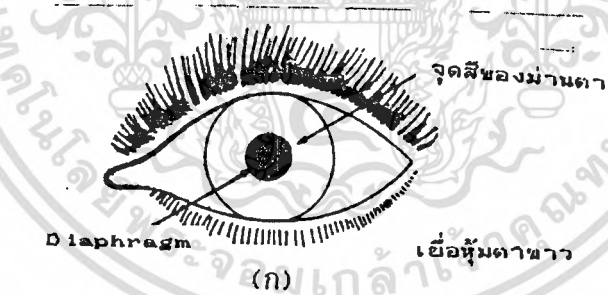


รูป 2.5 แสดงอิทธิพลจากพื้นฉากรอบด้านต่อความเป็นสีเทา

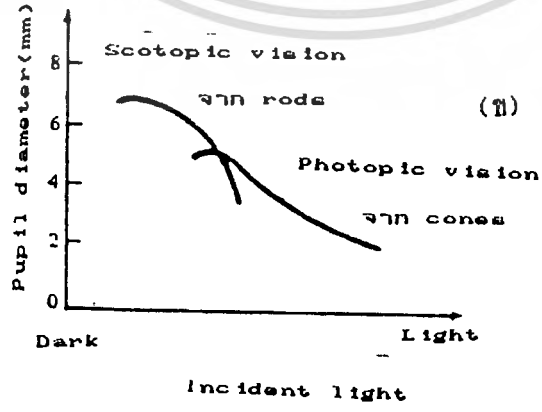
บริเวณ fovea การเชื่อมต่อระหว่าง cone กับ optical fibers ประมาณค่าเป็นหนึ่งต่อหนึ่ง เมื่อระยะทางจาก fovea เพิ่มขึ้น จำนวนของการเชื่อมต่อยิ่งมีค่ามากขึ้น rod จะมีความไวต่อความส่องสว่างต่ำและมีการตอบรับต่อ scotopic ส่วน cone ซึ่งมีความหนาแน่นมากในบริเวณ forvea อยู่ในตำแหน่งตามเส้นทางการมอง และตอบรับกับระบบการมอง photopic มากที่สุด

ช่องเปิดของลูกตาจะเปิดกว้างขึ้นเมื่อมีแสงสว่างปริมาณมากขึ้น เพื่อลดจำนวนแสงที่ตกกระทบสู่เรตินา อย่างไรก็ดีเมื่อเวลาผ่านไประยะหนึ่งเรตินาจะปรับตัวเข้าสู่ระดับใหม่ และช่องเปิดกลับเข้าสู่ระดับเดิม ช่องเปิดนี้สามารถควบคุมปริมาณแสงด้วยแฟกเตอร์ 30 เรตินานั้นสามารถปรับตามขนาดของแสงได้เช่นกันแต่ด้วยแฟกเตอร์ 10^{10}

หลังจากแสงผ่าน equeous humer มันจะผ่านช่องเปิดของตาบริเวณศูนย์กลางของม่านตา จากคุณสมบัติของเม็ดสีที่ม่านตาเป็นเหตุให้เกิดสีต่างๆของดวงตา ดวงตาของผู้หญิงมีช่องเปิดดวงตาใหญ่กว่าผู้ชาย ม่านตาทำหน้าที่คล้ายเครื่องกลในการควบคุมการปิดเปิดช่องตา ดังรูป 2.6 (ก) เป็นภาพด้านหน้าของดวงตามนุษย์ การเปลี่ยนแปลงขนาดของแผ่นปิดเปิดนี้ขึ้นอยู่กับกล้ามเนื้อสองชนิดคือ

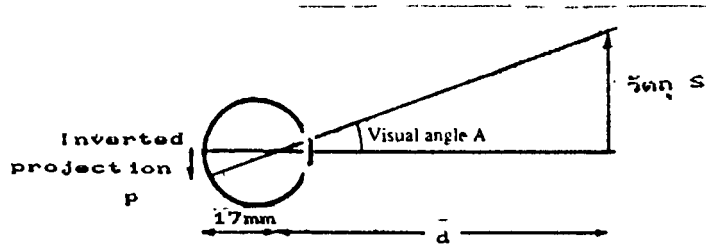


(ก)



รูป 2.6 แสดงการทำงานสองลักษณะของช่องตา

(ก) ภาพดวงตาของมนุษย์ (ข) การเปลี่ยนของช่องตาตามฟังก์ชั่นของแสง



รูป 2.7 การโปรเจคของวัตถุบนเรตินา

sphincler และ dilator pupillae กล้ามเนื้อส่วน sphincler ใช้สำหรับการบีบรัดโดยเคลื่อนที่ขนานไปกับวงกลมของม่านตา และ dilator ใช้สำหรับการขยาย ขนาดของช่องว่างตามฟังก์ชั่นแสงตกกระทบนั้นแสดงดังรูป 2.6 (ข)

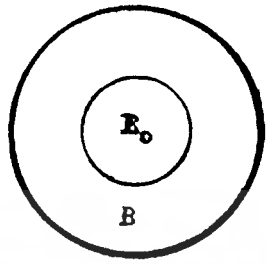
การเปลี่ยนขนาดของช่องตานั้นสนองต่อ 3 วัตถุประสงค์ คือผลต่อฟังก์ชั่นการสะท้อนแสงเพื่อปรับขนาดปริมาณของแสงที่เข้าสู่ดวงตา การปรับสายตาเพื่อรับวัตถุ และปรับภาพภายใต้เงื่อนไขจากแสงที่สว่างมาก

ภาพถูกปรับระยะการมองได้ โดยเปลี่ยนระยะโฟกัสให้ภาพตกลงที่เรตินา และผลความสูงของภาพกลับหัวบนพื้นเรตินาจะสัมพันธ์กับวัตถุดังรูป 2.7

พารามิเตอร์ที่สัมพันธ์กับการมองเห็น และใช้ประโยชน์ในกระบวนการรูปภาพ เพื่ออธิบายให้เข้าใจในรายละเอียดคุณสมบัติเกี่ยวกับการมอง จากแนวคิดพื้นฐานทั่ว ๆไปพารามิเตอร์เหล่านี้คือ ความสว่าง ความชัด ความละเอียด และความคม

Brightness เป็นแนวคิดทางจิตศาสตร์หรือเป็นความรู้สึกสัมพันธ์กับจำนวนของแสงที่ถูกสร้างขึ้น เนื่องจากดวงตามนุษย์มีความสามารถปรับตัวตามขนาดของแสงอย่างมาก ดังนั้นมนุษย์จึงไม่สามารถจะตัดสินใจเกี่ยวกับค่าปริมาณแสงสว่างได้อย่างถูกต้อง

Lightness นั้นสัมพันธ์กับการรับรู้ของผู้สังเกตของความแตกต่างใน ความเป็นสีดำขาว หรือความเป็นสีเทาระหว่างวัตถุ ดังนั้นจึงแตกต่างจาก brightness คำว่า contrast ใช้สำหรับบ่งบอกลักษณะความแตกต่างในความสว่างนั้น ค่าความเป็นสีเทาขึ้นอยู่กับฉากเปรียบเทียบเบื้องหลัง รูป 2.5 แสดงคุณสมบัติทางธรรมชาติซึ่งเรียกว่า simultaneous contrast โดยปกติถ้าอัตราส่วน



รูป 2.8 contrast ระหว่างวัตถุกับฉากโดยรอบ

ของ contrast ระหว่างวัตถุและฉากเบื้องหลังมีค่าคงที่ ค่า lightness จะคงที่เช่นกัน

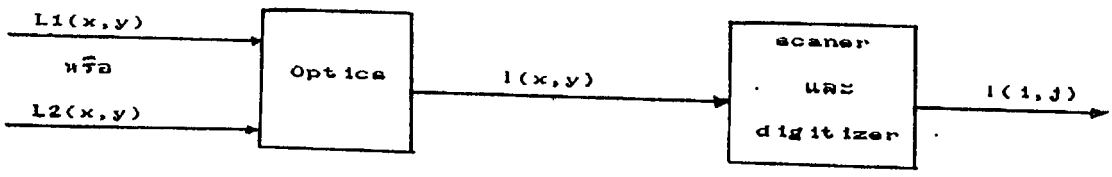
วิธีการวัด contrast ที่ใช้กันอย่างกว้างขวางนั้น ค่า C เป็นอัตราของ contrast ระหว่างวัตถุ B_o และพื้นที่โดยรอบในขณะนั้น B ดังแสดงในรูป 2.8

$$C = (B_o - B)/B \qquad (2.1)$$

จากนิยามนี้ค่าของ contrast สามารถเป็นได้ทั้งค่าบวกและลบ ถ้าค่าเป็นลบนั้นหมายถึงค่าความสว่างของวัตถุมีค่าน้อยกว่าพื้นฉากโดยรอบ

2.2 การแปลงภาพจริงเป็นข้อมูลภาพ

เราพิจารณาแหล่งพื้นฐานของข้อมูลภาพ 2 ลักษณะ กรณีแรกคือการสร้างสัญญาณของสิ่งมีชีวิตจากภาพจริง และในกรณีของการคืนภาพเดิมจากข้อมูลที่เก็บไว้ในตัวกลางทางฟิสิกซ์ ถ้าเราสังเกตจากรูป 2.9 ที่ขั้นตอน optic ซึ่งจะสร้างสัญญาณภาพ I(x,y) ป้อนให้กับส่วนของการสแกน และ digitizer เป็นส่วนของการนำข้อมูลของภาพเข้าสู่คอมพิวเตอร์ ภาพอนาลอก $L_1(x,y)$ หรือ $L_2(x,y)$ ถูกเปลี่ยนเป็นภาพสัญญาณดิจิตอล I(x,y) และบางครั้งเรียกว่าเป็นส่วนเรตินาของคอมพิวเตอร์ เราจะอธิบายถึง hardware ในส่วนที่จำเป็นสำหรับการให้รูปภาพเป็นอินพุทของคอมพิวเตอร์ เรากำหนดให้ภาพดิจิตอล I(i,j) เป็นฟังก์ชั่นจริง 2 แกนของตัวแปร discrete ค่า i,j ซึ่งเป็นค่าอ้างถึงแสงและเงา สี ระดับที่ปรากฏบนภาพ ค่าเหล่านี้สัมพันธ์กับอินพุท $L_1(x,y)$ หรือ $L_2(x,y)$ อย่างใดอย่างหนึ่ง ในบริเวณ (x,y) ในบริเวณนี้เป็นค่า discrete ของจุดต่างๆบนภาพและเรียกว่า



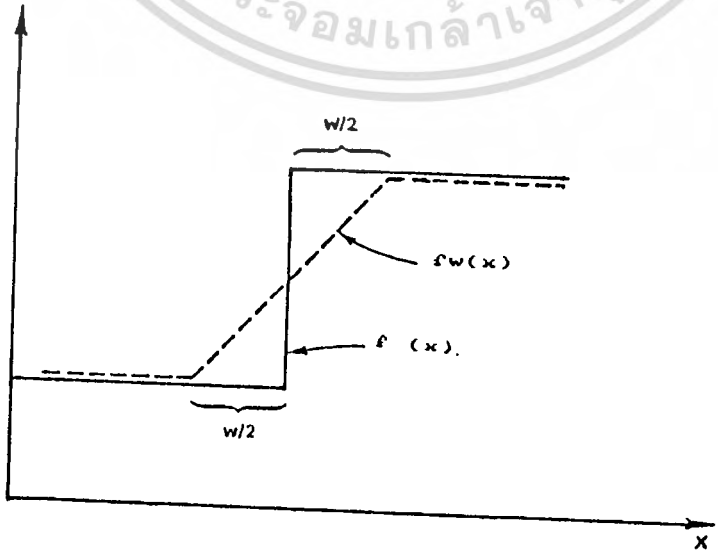
รูป 2.9 เป็นการแสดงตัวสแกนและ digitizer ของอินพุท

pixel ในเรื่องของ hardware มีการทำงานสองขั้นตอน คือส่วนของการสุ่มข้อมูลจากภาพเป็น pixels และการทำ quantize ของระดับสีเทา ขนาดแนวกว้างยาวของภาพจะเป็น 128 x 128 256 x 256 และ 512 x 512 แต่เราอาจขยายขึ้นถึงอันดับ 2048 x 2048 ซึ่งมักจะไม่ใช้ในงานปกติของ remote sening ระดับของ gray level นั้นมีระดับความแตกต่างกันเพียง 64 ระดับก็สามารถให้ความสมบูรณ์พอเพียง ถึงแม้ว่าในบางครั้งมีการใช้ถึง 256 ระดับ

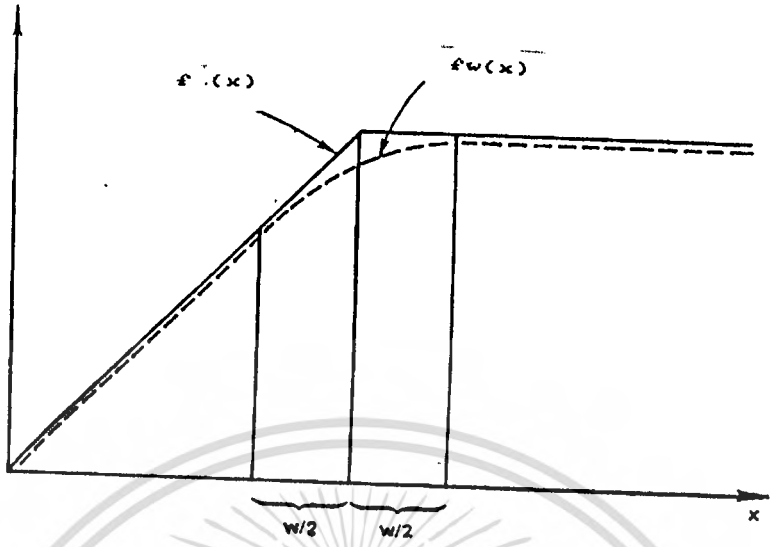
2.3 กระบวนการเกี่ยวกับภาพโดยคอมพิวเตอร์[22]

2.3.1 การกำจัดสัญญาณรบกวนของภาพโดยวิธีการเฉลี่ย

เทคนิคการจำแนกสัญญาณรบกวนซึ่งปนมากับภาพ เราใช้วิธีการเฉลี่ย ถึงแม้ว่าจุดประสงค์ของเราคือการประมาณค่าอัตราการเปลี่ยนแปลงของฟังก์ชั่นก็ตาม แต่เทคนิคการเฉลี่ยนี้สา



รูป 2.10 (ก) ตัวอย่างของ regularization ในหนึ่งมิติ



รูป 2.10(ข) ตัวอย่างอีกแบบหนึ่งของ regularization ในหนึ่งมิติ

มารถทำให้ภาพที่มีสัญญาณรบกวนเรียบขึ้น ด้วยความคิดพื้นฐานว่าสัญญาณเทาที่จุดใดๆแทนได้โดยค่าเฉลี่ยของฟังก์ชั่นภาพในบริเวณใกล้เคียง ในกรณีของฟังก์ชั่นภาพทางอนาลอก f(x) หนึ่งมิติ เรานิยามค่าความกลมกลืน f_w นี้ด้วย

$$f_w = (1/W) \int_{x-w/2}^{x+w/2} f(u)\, du \qquad (2.2)$$

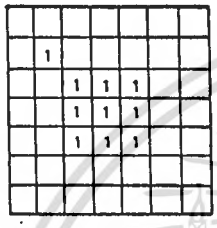
การประมวลผลนี้เรียกว่า regularization ของฟังก์ชั่น ซึ่งไม่เป็นการยากในการคำนวณ เพื่อพิสูจน์ว่า regularization ของฟังก์ชั่นความกลมกลืนนั้นถูกต้อง ถ้า f(x) เป็นค่าไม่ต่อเนื่องค่า $f_w(x)$ จะต่อเนื่อง และถ้า f(x) ต่อเนื่องแล้ว $f_w(x)$ มีค่าอนุพันธ์ที่ต่อเนื่อง คุณสมบัตินี้แสดงในรูป 2.10 (ก)และ (ข) สังเกตได้ว่าความกว้างของบริเวณที่เปลี่ยนแปลงใน $f_w(x)$ ขึ้นอยู่กับขนาดของหน้าต่างภาพ การเพิ่ม W เป็นผลทำให้ภาพมัวไม่ชัด ในทางกลับกันถ้าขนาดของ W ลดลงทำให้ $f_w(x)$ มีค่าประมาณเท่ากับ f(x) เราเรียกฟังก์ชั่น regularization นี้ว่า slide หรือการเฉลี่ยเคลื่อนที่ของ f(x)

การขยายวิธีประมวลผลออกเป็นสองมิติโดยตรง ถ้าให้ฟังก์ชั่นภาพทางอนาลอกเป็น f(x,y) เรานิยามฟังก์ชั่น regularization $f_w(x,y)$ ได้ดังนี้

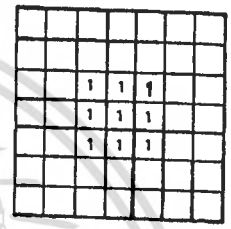
$$f_w(x,y) = (1/A_w) \iint_{w(x,y)} f(u,v)\, du\, dv \qquad (2.3)$$

โดย $W(x,y)$ เป็นค่าของหน้าต่างภาพของพื้นที่ A_w ซึ่งมีจุดกึ่งกลางบนเซล (i,j) โดยเฉพาะอย่างยิ่งถ้าใช้หน้าต่างภาพที่เป็นรูปทรงเรขาคณิต หากเราใช้หน้าต่างภาพเป็นรูปสี่เหลี่ยมผืนผ้าที่มีความยาวฐานเป็น $(2b+1)$ และสูง $(2h+1)$ แล้ว regularized ของ $f(i,j)$ จะเป็น

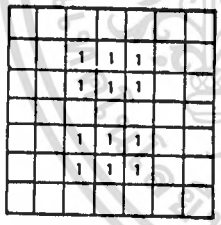
$$f_w(i,j) = \sum_{-b\le m\le b} \sum_{-h\le n\le h} f(i+m,j+n)/[(2b+1)(2h+1)] \qquad (2.4)$$



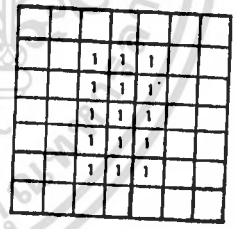
รูป 2.11



รูป 2.12



รูป 2.13



รูป 2.14

ตัวอย่างของ regularization ใน 2 มิติ

การ regularizing จะมีผลเช่นเดียวกับการทำให้กล้องไม่อยู่ในโฟกัส และด้วยขนาดของหน้าต่างภาพของการเฉลี่ยก็ยิ่งทำให้กล้องออกนอกโฟกัสมากขึ้นนี้ เป็นผลที่ไม่ต้องการ เราต้องการภาพที่มีความคมชัดเท่าที่เป็นไปได้ ดังนั้นเราต้องจัดการ regularization ให้เป็นไปตามจุดประสงค์ตามความต้องการ และเหมาะสมที่นำมาใช้กับภาพไบนารี่ดิจิตอล ภาพไบนารี่คือภาพที่มีสองระดับ ขาว(1) ดำ(0) เท่านั้น กลุ่มของเซล $f(i,j) = 1$ เรียกว่า ภาพ (figure) กลุ่มของ

f(i,j) = 0 เรียกว่า พื้น (ground) รูปภาพไบนารีนี้มีความน่าสนใจอย่างมาก เพราะผลจากการประมวลอื่นๆ การทำภาพไบนารี f(i,j) ให้มีระดับกลมกลืน โดยการเปรียบเทียบ $f_w(i,j)$ กับสัญญาณเทรสโฮลด์ ถ้าเทรสโฮลด์มีค่ามากกว่าแล้ว ค่าฟังก์ชั่นความกลมกลืนเป็น 1 หรือน้อยกว่าจะมีค่าเป็น 0 การกระทำแบบนี้จะทำให้สร้างลายเส้นคอนทัวร์ของภาพไบนารี่ได้คงที่ ตัวอย่างเช่นถ้าเรากำหนดค่าเทรสโฮลด์ ไว้ที่ 0.4 การกระทำนี้จะกำหนดให้เซลใน (i,j) ในภาพที่กลมกลืนแล้วมีค่าเป็น 1 ถ้าเซล 4 เซลรอบ ๆ จุด (i,j) ใด ๆ ในหน้าต่างเป็น 1 ภาพไบนารี่ในรูป 2.11 ถูกทำให้กลมกลืน ได้ผลลัพธ์ดังรูป 2.12 โดยส่วนที่ยื่นบนรูปด้านซ้ายบนถูกนำออกไป (เซลที่ว่างเป็น 0) ในอีกกรณีหนึ่งเมื่อนำการกระทำนี้มาใช้กับภาพที่ไม่ต่อเนื่องแสดงในรูป 2.13 ผลลัพธ์แสดงในรูป 2.14 โดยสังเกตุได้ว่าช่องว่างซึ่งแยกรูปเป็นสองส่วนถูกเติมให้เต็ม ถ้าเราพิจารณาช่องว่างหนึ่งในรูปภาพเป็นสำคัญเราจะเห็นว่าจุดนี้เป็นจุดที่บกพร่องของวิธีการนี้ แม้ว่าจะเป็นตัวอย่างง่าย ๆ แต่แสดงให้เห็นว่าวิธีการนี้ก็มีจุดบกพร่องเช่นกัน

ถ้าเราทำเกี่ยวกับภาพไบนารี่มีเทคนิคของการทำภาพให้กลมกลืน และสามารถควบคุมให้ถูกต้องมากกว่า regulariztion เทคนิคอันนี้เรียกว่า logical smoothing หรือ logical averaging ซึ่งเป็นพื้นฐานของการสังเกตว่าองค์ประกอบของภาพที่ปรากฏภายในหน้าต่างของการเฉลี่ยสามารถใช้ฟังก์ชั่นบูลีนหรือตัวแปรทาง logic ค่าของภาพที่กลมกลืนที่จุด ๆ หนึ่งสามารถนิยามโดยฟังก์ชั่นคณิตศาสตร์บูลีนของตัวแปรเหล่านี้ ดังตัวอย่างถ้าเราต้องการสร้าง ตัวกระทำภาพให้กลมกลืนทาง logic กำหนดดังต่อไปนี้

ก) เซล "0" จะเปลี่ยนเป็นเซล "1" ถ้าเซลอยู่ข้างเคียงเป็น "1"

ข) เซล "1" จะเปลี่ยนเป็นเซล "0" ถ้าเซลอยู่ข้างเคียงเป็น "0"

ข้อกำหนดนี้ของตัวกระทำจะทำให้เกิดการกลับค่าของ "1" หรือ "0" ที่โดดเดี่ยวเท่านั้นแต่จะไม่เปลี่ยน

a	b	c
d	e	f
g	h	i

แปลงภาพส่วนอื่น ๆ ดังตัวอย่างการใช้หน้าต่างภาพ 3 x 3 ดังรูป

เรานิยาม e ค่าของภาพใหม่ที่จุดกึ่งกลางเซลของหน้าต่างภาพเป็น

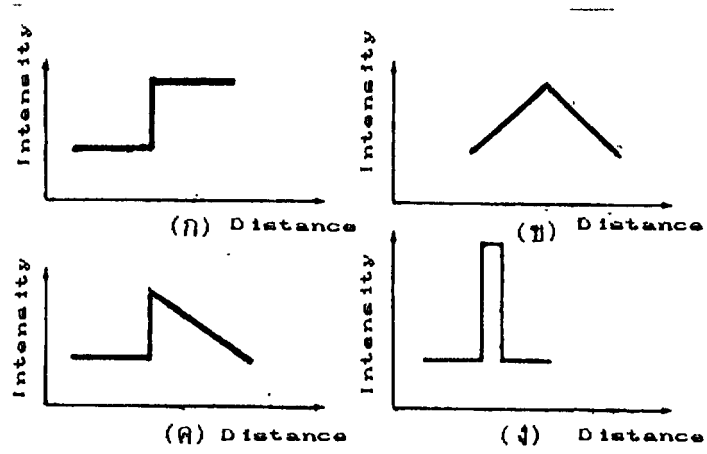
$$e = \overline{e}.(a.b.c.d.e.f.g.h.i) \vee e\ (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h \vee i)$$

โดย a V b คือ a or b a คือ "not a" และ a.b คือ "a and b" เราจะเห็นว่าฟังก์ชั่นทางตรรกวิทยาสามารถออกแบบได้ง่าย ค่าใหม่ e เป็น "1" ถ้า e เป็น "0" และค่าข้างเคียงทั้งหมดเป็น "1" หรือค่า e เป็น "1" ถ้าค่าข้างเคียงอย่างน้อยที่สุดหนึ่งจุดเป็น "1" เราจะเห็นข้อได้เปรียบของ logical smoothing กว่า regularization แล้วคือสามารถให้เรากำหนดเงื่อนไขที่สร้างขึ้นที่จะทำให้ค่าของเซลใด ๆ เปลี่ยนแปลงได้ เราอาจพิจารณาจากความเป็นจริงว่าการทำภาพให้กลมกลืนโดยวิธี regularization ของภาพไบนารี่เป็นกรณีพิเศษของ logical smoothing เพราะการกระทำโดยวิธี regularization ทางตัวเลขสามารถนิยามได้ด้วยการแสดงผลทางวิธีการของ boolean

2.3.2 การกำจัดสัญญาณรบกวนบนข้อมูลภาพโดยคุณสมบัติทางสถิติ

ข้อมูลใด ๆ ที่มีค่าแตกต่างไปจากข้อมูลข้างเคียงซึ่งถือเป็นสัญญาณรบกวนนั้น จะถูกปรับค่าให้ใกล้เคียงกับกลุ่มข้อมูลที่ล้อมรอบข้างเคียง โดยไม่ทำให้ภาพเสียความคมชัด ซึ่งการให้ความหมายของข้อมูลภาพผิดพลาดไป จากการหาความกลมกลืนของภาพโดยวิธีการเฉลี่ยดังที่ได้กล่าวมาแล้วนั้น เมื่อนำไปใช้กับภาพที่มีความซับซ้อนจะทำให้สูญเสียรายละเอียดของข้อมูลภาพไป โดยเกิดการเลือนหายของบริเวณขอบภาพไม่คมชัด การแก้ไขโดยวิธีการอาศัยคุณสมบัติทางสถิติ จากการทำภาพให้กลมกลืน ด้วยการแทนค่าเฉลี่ยของสัญญาณที่ตำแหน่งนั้นด้วยค่าเฉลี่ยของสัญญาณที่มีความเป็นเนื้อเดียวกัน (homogeneous) สูงสุดจากพื้นที่ที่มีรูปร่างลักษณะต่าง ๆ โดยแต่ละพื้นที่จะมีจำนวนจุดภาพเท่ากันและมีจุดภาพ ณ ตำแหน่งที่ต้องการนั้นเป็นจุดร่วมของทุกพื้นที่

กระบวนการการกำจัดสัญญาณรบกวนโดยที่ยังรักษาความคมชัดของขอบเขตของภาพมีวิธีการกล่าวคือ ขั้นตอนแรกสร้างพื้นที่ที่มีลักษณะต่าง ๆ โดยแต่ละพื้นที่มีจำนวนจุดภาพเท่ากันและให้พื้นที่ที่มีความเป็นเนื้อเดียวกันมากที่สุด ในขั้นตอนที่สองคือการคำนวณหาค่าเฉลี่ย (mean) และค่าความผันผวน (variance) ของพื้นที่ดังกล่าวทั้งหมด ตรวจสอบค่าความผันผวนต่ำสุดซึ่งแสดงว่าพื้นที่นั้นเป็นพื้นที่ที่มีความเป็นเนื้อเดียวของภาพมากที่สุด นำเอาค่าเฉลี่ยของพื้นที่นี้ไปเป็นค่าความเข้มของระดับสัญญาณ



รูป 2.15 ฟังก์ชั่นของขอบภาพในอุดมคติ

(ก) ฟังก์ชั่นหนึ่งหน่วย (ข) เปลี่ยนแบบรูปหลังคา

(ค) ผสมระหว่างรูปแบบหลังคาและฟังก์ชั่นหนึ่งหน่วย (ง) ขอบยอดแหลม

เทาของภาพที่ตำแหน่งนั้น

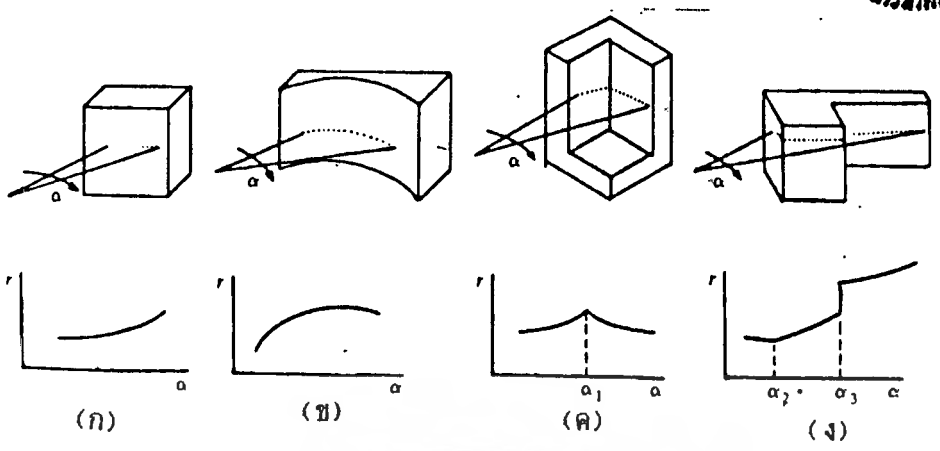
ผลลัพธ์ในการจำกัดการรบกวนด้วยวิธีนี้ สามารถรักษาความคมชัดของขอบภาพไว้ได้

2.3.3 คุณลักษณะของขอบภาพ

ขอบภาพอาจให้ความหมายว่าเป็นบริเวณที่ขาดการต่อเนื่องของระดับสัญญาณเทา หรือสัญญาณสี โดยทั่วไปขอบภาพจะมีขนาดต่าง ๆ กัน ทั้งขนาดสั้นและยาว รูปแบบของขอบในอุดมคติมักใช้แทนด้วยฟังก์ชั่นทางคณิตศาสตร์ในรูปแบบใด ๆ รูปแบบหนึ่ง

รูปแบบตามอุดมคติของขอบภาพ แทนได้ด้วยฟังก์ชั่นทางคณิตศาสตร์ดังตัวอย่างในรูป 2.15 (ก) ซึ่งเป็นการแสดงลักษณะของขอบภาพที่คล้ายคลึงกับความรู้สึกของเรา รูปภาพที่ประกอบด้วยวัตถุที่มีพื้นผิวเป็นรูปเหลี่ยม ย่อมแสดงการหักมุมของขอบในคุณลักษณะของความเข้มแสงต่อระยะทางดังรูป 2.15 (ข) เป็นสัญญาณรูปหลังคาประกอบด้วยสัญญาณลาดเอียงสองรูป และเราอาจรวมระหว่างฟังก์ชั่นหนึ่งหน่วยกับสัญญาณลาดเอียงดังรูป 2.15 (ค) ภาพที่มีลายเส้นที่มีความแตกต่างอย่างกระทันหัน อาจแสดงลายเส้นนั้นดังรูป 2.15 (ง) ซึ่งเป็นการสร้างจากฟังก์ชั่นหนึ่งหน่วยสองรูป บางครั้งเราเรียกขอบยอดแหลม (sprike edge)

ที่ศูนย์รวมของลายเส้นในภาพหนึ่งมิติ ย่อมสัมพันธ์กับส่วนของเรขาคณิตในภาพสามมิติสำหรับรูปทึบหลายเหลี่ยม เส้นขอบบนภาพจะปรากฏบริเวณตัดกันของผิวหน้ารูปเหลี่ยมนั้น ชนิดต่าง ๆ ของขอบที่สามารถสังเกตเห็นได้ตามลักษณะความลึก แสดงดังรูป 2.16 ขอบเหล่านี้เขียนเป็นรูปกราฟ



รูป 2.16 แสดงโครงสร้างของขอบและผิวหน้าของรูปเหลี่ยมโดยระยะความลึกแสดงอยู่ด้านล่าง ซึ่งเขียนแสดงระยะทางจากผู้สังเกตครอบคลุมไปตามแนวผิวหน้าด้วยมุม α ในรูป 2.16 (ก) และ (ข) เส้นสแกนมีความต่อเนื่องและกราฟมีลักษณะกลมกลืน มุมเว้า $\alpha 1$ ในรูปที่ 2.16 (ค) และมุมยื่น $\alpha 2$ ในรูป 2.16 (ง) จะเกิดการไม่ต่อเนื่องที่ระยะทาง r ทั้งสองชนิดแตกต่างที่เครื่องหมาย บนผิวหน้าต่างกันของวัตถุอาจเกิดการขาดตอนในแนวความลึก ดังรูป 2.16 (ง) ที่มุม $\alpha 3$ ลักษณะขอบแบบนี้เรียกว่า occluding หรือ disoccluding ขึ้นอยู่กับว่ามีพื้นที่ฉากหลังคลุมตลอดพื้นที่หรือไม่ อีกชนิดหนึ่งคือเส้นทางเดินของขอบ (contour edge) แสดงให้เห็นแนวเส้นของขอบกับพื้นของฉาก ซึ่งอาจปรากฏบนภาพ 2.16 ถ้าขยายขอบเขตการสแกนออกไปทั้งทางด้านซ้ายและขวา

ภาพลายเส้นของวัตถุนั้นบางครั้งอาจเกี่ยวข้องกับลักษณะพื้นผิวหลายแบบ ตัวอย่างเช่น ยอดแหลม หรือภาพยอดแหลมซึ่งปรากฏบนฟังก์ชั่นหนึ่งหน่วย อาจเป็นผลของขอบที่ยื่นออกมา ในทำนองเดียวกันรูปหลังคาหรือรูปหลังคาซ้อนอยู่บนฟังก์ชั่นหนึ่งหน่วย เป็นลักษณะของขอบเว้าเข้า รวมทั้งรูปยอดแหลมหรือรูปยอดแหลมซ้อนบนฟังก์ชั่นหนึ่งหน่วย เป็นทั้งลักษณะของขอบยื่นและขอบเว้า กรณีดังที่กล่าวมานั้นเป็นกรณีที่พบได้บ่อยครั้ง ถ้าเราสามารถจัดวิธีการย้อนกลับในการเกิดภาพลายเส้นนี้ได้ย่อมทำให้การหารูปทรงสามมิติจากภาพสองมิติมีความง่ายขึ้น

สัญญาณในกรณีอุดมคตินั้น มักจะไม่ปรากฏในข้อมูลจริง มีการแสดงออกในรูปแบบหลายชนิดขึ้นอยู่กับอิทธิพลของสภาพแวดล้อม สิ่งหนึ่งที่สำคัญคือการส่องสว่างของแหล่งกำเนิดแสงที่เรียกว่าขอบความส่องสว่าง เป็นต้นเหตุให้เห็นความส่องสว่างบนผิวหน้า กำหนดคุณลักษณะโดยตำแหน่งและความลาดเอียงในสามมิติจากแหล่งกำเนิดแสงเมื่อเปรียบเทียบกับผิววัตถุนั้น ยกตัวอย่างเช่นขอบภาพ

จากเงา ขอบภาพในกรณีนี้ขึ้นอยู่กับการสะท้อนของผิวหน้าวัตถุในสามมิติ และเรียกว่าขอบของการสะท้อน (reflecitivity edge) เป็นที่น่าสนใจว่ามนุษย์แยกขอบที่เกิดจากทั้งสองกรณีได้โดยพื้นฐานของความสัมพันธ์ระหว่างพื้นผิวหน้าใกล้เคียง ซึ่งต่างจากความรู้สึกจากค่าความส่องสว่างที่ปรากฏบนเรตินา อุปกรณ์ทางด้านตัวแปลงเป็นเชิงเลขทำให้เกิดสัญญาณรบกวนและความสามารถในการแยกขอบภาพมีค่าต่ำลง กระบวนการสุ่มตัวอย่างเป็นต้นเหตุของความผิดพลาดเช่นกัน รวมทั้งข้อมูลตามธรรมชาตินั้นมีส่วนประกอบครอบคลุมหลายลักษณะ เช่นทำให้เกิดขนาดต่าง ๆ ของสัญญาณเทาคละปนกันไปทุกช่วงของภาพ เราเรียกโครงสร้างผิวหน้าแบบนี้ว่า texture ซึ่งเป็นผลจากอิทธิพลของแหล่งกำเนิดการส่องสว่างและผิวหน้านั้น

ขอบภาพมีความสำคัญสำหรับระบบการมองของสิ่งมีชีวิตและระบบคอมพิวเตอร์ เพราะเป็นส่วนที่แสดงให้เห็นเด่นชัดถึงรูปทรงของวัตถุในรูปภาพ แต่กระนั้นก็ยังไม่สมบูรณ์ สิ่งที่เราต้องแก้ปัญหาต่อไปก็คือ เราจะใช้วิธีการใดเพื่อหาลักษณะข้อมูลที่ปรากฏอยู่ในขอบเหล่านี้

2.3.4 การหาขอบภาพในหนึ่งมิติ[17]

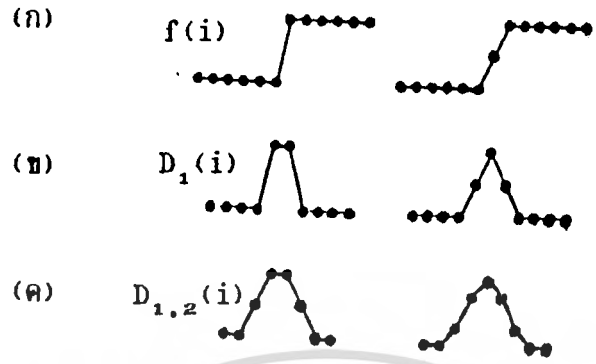
การหาขอบของภาพในกรณีที่จะกล่าวนี้เป็นการหาค่าจากระดับความต่างของความเข้มแสง ถ้ากำหนดให้ i เป็นตำแหน่งเป็นตำแหน่งของจุดต่าง ๆ ในทิศทางตั้งฉากกับแนวเส้นของขอบภาพ วิธีการที่ง่ายในการคำนวณคือ

$$D_d(i) = f(i+d)-f(i-d) \qquad (2.5)$$

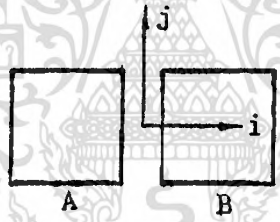
ถ้า $D_d(i)$ เป็นค่าบวก แสดงว่าขอบภาพเป็นแบบฟังก์ชั่นหนึ่งหน่วยซึ่งมีค่าเพิ่มขึ้นทางด้านบวก และถ้าเป็นค่าลบขอบภาพก็มีลักษณะตรงข้าม ในรูป 2.17 แสดงลายเส้นที่ได้ผลจากการใช้สมการนี้โดยกำหนดค่า d เท่ากับ 1 รูป 2.17(ข) เป็นกรณีที่ขอบอยู่ตรงกลางของจุดภาพ ค่าของ $D_d(i)$ อาจมีค่าสูงมากถ้าเป็นบริเวณที่เกิดสัญญาณรบกวนมาก เพื่อลดผลจากเหตุนี้เราจะเฉลี่ยค่าความแตกต่างเป็น

$$D_{d_1,d_2}(i) = \sum_{k=d_1}^{d_2} \{f(i+k)-f(i-k)\}/(d_2-d_1+1) \qquad (2.6)$$

โดยปกติแล้วค่า $d_1 = 1$ และค่า d_2 มีค่าระหว่าง 2 ถึง 5 ซึ่งในรูปที่ 2.17 แสดงผลของการเฉลี่ยกำหนดค่า $d_1 = 1$ และ $d_2 = 2$ เห็นได้ว่าเสมือนกับการเฉลี่ยค่าของ $D_d(i)$ ดังตัวอย่าง



รูป 2.17 ลายเส้นของความเข้มแสงและผลจากการหาค่าความแตกต่าง



รูป 2.18 เป็นการหาค่าเฉลี่ยแบบ 2 มิติจากวิธีการหาค่าความแตกต่าง

$$D_{1,3}(i) = \sum_{k=-1}^{1} D_2(i+k)/3 \qquad (2.7)$$

ถ้าเราต้องใช้วิธีการดังกล่าวกับทุก ๆ จุดภาพ(ยกเว้นบริเวณจุดภาพภายนอก) เราสามารถคำนวณได้มีประสิทธิภาพดีขึ้นถ้าใช้วิธีการทำ smooth ภาพเสียก่อน แทนที่จะใช้วิธีการ $D_{1,2}$ เราใช้สมการดังต่อไปนี้เป็นการ smooth

$$f_1(i) = \sum_{k=-1}^{1} f(i+k) \qquad (2.8)$$

จากนั้นจึงใช้ $D_2(i)$ กระทำต่อ $f_1(i)$ อีกครั้งหนึ่ง

การใช้วิธีนี้สามารถใช้เป็นค่าเฉลี่ยแบบ 2 มิติได้ดังรูป 2.18 ซึ่ง j เป็นทิศทางของ

ขอบภาพตั้งฉากกับ i ผลลัพธ์ของการกระทำจะเป็นค่าแตกต่างระหว่างบริเวณแค่ละด้าน ถ้าค่า A และ B ใช้กำหนดเป็นบริเวณสองแห่ง ดังรูป เราสามารถเขียนสมการได้ดังนี้

$$DD(i,j) = \sum_{(k,l)\in B} f(k,l)/|B| - \sum_{(k,l)\in A} f(k+l)/|A| \qquad (2.9)$$

ผลที่ได้รับก็ไม่แตกต่างกันจากวิธีคำนวณอย่างง่าย ๆ ดังในสมการที่ 2.5

วิธีการต่อมานั้นได้ปรับปรุงมาจากการกระทำแบบขอบฟังก์ชั่นหนึ่งหน่วย ขอบที่มีลักษณะรูปแบบหลังคา สามารถหาได้จากการเปรียบเทียบความชันของลายเส้นสองด้าน ความชันที่ตำแหน่ง i แทนด้วยสมการ $D_d(i)/2d$ ดังนั้นขอบรูปหลังคาจะกระทำด้วยสมการดังนี้

$$L_{d,d'}(i) = \{D_d(i+d')-D_d(i-d')\}/2d \qquad (2.10)$$

เครื่องหมายของผลลัพธ์จะบอกได้ว่าขอบรูปหลังคามีทิศทางใด เพราะพารามิเตอร์ d มีค่าคงที่ สมการที่ 2.10 สามารถเขียนให้ง่ายเป็น

$$L_{d,d'}(i) = \{D_d(i+d')-D_d(i-d')\} \qquad (2.11)$$

สมการ 2.11 เป็นการหาค่าความแตกต่างในอันดับสอง และค่าพารามิเตอร์ทั้งสอง d และ d' นั้นแทนด้วยค่าเท่ากัน และใช้ $L_d(i)$ เขียนแทน $L_{d,d}(i)$ เราจะได้

$$\begin{aligned} L_d(i) &= \{D_d(i+d)-D_d(i-d)\} \\ &= f(i+2d)+f(i-2d)-2f(i) \end{aligned} \qquad (2.12)$$

ซึ่งวิธีการนี้ก็สามารถปรับปรุงวิธีการลดสัญญาณรบกวน ในทำนองเดียวกันกับวิธีการแบบฟังก์ชั่นหนึ่งหน่วย

ผลของการหาค่าขอบจากค่าความแตกต่างของระดับความเข้มแสงในแต่ละด้าน วิธีการที่ง่ายที่สุดโดยการใช้ $L_d(i)$ ในสมการที่ 2.12 ซึ่งค่าของ d จะต้องมีค่าน้อย (d=1 หรือ d=1/2) ถ้าค่าผลลัพธ์มีค่าเป็นบวก แสดงว่าเส้นขอบมีผลเป็นชนิดขอบลบ และในทำนองกลับกันถ้าค่าผลลัพธ์ของการกระทำเป็นลบแสดงว่าเส้นขอบมีผลเป็นชนิดขอบบวก การหาค่าขอบโดยวิธีนี้เราลดค่าสัญญาณรบกวนได้จากการเฉลี่ยดังตัวอย่าง

$$L_{d,m}(i) = \sum_{k=-m}^{m} L_d(i+k)/(2m+1) \qquad (2.13)$$

จากตัวอย่างที่ได้แสดงขอบภาพในลักษณะต่าง ๆ แล้วนั้น การอธิบายต่อไปอ้างถึงรูปขอบชนิดแบบฟังก์ชั่นหนึ่งหน่วย

จากฟังก์ชั่นของตัวกระทำต่าง ๆ มักเป็นแบบเชิงเส้นของ f(i) ดังนั้นจึงสามารถ

แทนได้โดยการคอนโวลูชั่นของภาพ กับช่องหน้าต่างของฟังก์ชั่นน้ำหนัก (weighting function) ดังตัวอย่าง $D_{d1,d2}(i)$ ในสมการ 2.6 สามารถเขียนได้ใหม่ดังนี้

$$D_{d1,d2}(i) = \sum_{k=-d2}^{d2} \{f(i+k)w(k)\}/(d_2-d_1+1) \qquad (2.14)$$

ซึ่ง

$$w(k) = \begin{cases} -1 & -d_2 \leq k \leq -d_1 \\ 0 & -d_1 < k < d_1 \\ 1 & d_1 \leq k \leq d_2 \end{cases} \qquad (2.15)$$

2.3.5 การหาขอบในสองมิติ[17]

ถ้าทิศทางของขอบภาพไม่อาจคาดหมายทิศทางได้ วิธีการหาขอบแบบสองมิติจะถูกนำมาเพื่อหาเส้นของขอบภาพในทุกทิศทาง จุดของภาพที่อยู่ข้างเคียงถูกนำมาว่าจุดใดเป็นบริเวณขอบหรือไม่ ค่าของจุดข้างเคียงเหล่านั้นขึ้นอยู่กับช่องหน้าต่างในการกระทำ โดยทั่วไปช่องหน้าต่างที่มีขนาดเล็กจะพิจารณาความละเอียดได้มากกว่าและการคำนวณย่อมง่ายกว่าด้วย

เมื่อ Roberts(1975) ได้พัฒนาโปรแกรมสำหรับจดจำภาพรูปทรงเหลี่ยมหลายหน้าใช้ช่องหน้าต่างขนาด 2x2 เพื่อหาขอบภาพ(สำหรับฟังก์ชั่นหนึ่งหน่วย) เป็นการหาความแตกต่างชนิด

A	B	C
D	E	F
G	H	I

รูป 2.19 ค่าความเข้มแสงขนาดช่องหน้าต่าง 3x3

สองมิติ

$$D_2(i,j) = [\{f(i,j)-f(i+1,j+1)\}^2 + \{f(i+1,j)-f(i,j+1)\}^2]^{1/2} \qquad (2.16)$$

สมการที่ 2.16 มีค่าเสมือนการหาค่าความชัน $f(i,j)$ สมการนี้สามารถปรับปรุงจากการใช้ค่าผลบวกของสมการกำลังสองมาเป็นการใช้ผลบวกของค่าสมบูรณ์แทน เพื่อประหยัดเวลาในการคำนวณ

ช่องหน้าต่างที่ใช้เป็นตัวกระทำส่วนมากเป็นขนาด 3x3 เพราะสามารถบรรจุค่าของ

จุดภาพโดยรอบได้หมด ดังรูป 2.19 เราสามารถคำนวณค่าความลาดชัน G ที่จุดภาพใด ๆ จากการคำนวณส่วนของ G_x และ G_y จากการคำนวณค่าความเข้มของแสงสว่างจากจุดภาพข้างเคียงดังนี้

$$G_x = (C+2F+I)-(A+2D+G)$$

$$G_y = (A+2B+C)-(G+2H+I) \qquad (2.17)$$

$$G = (G_x, G_y)$$

ถ้าค่าขนาดความชัน $|G|$ มีค่ามาก ทิศทางความชันสูงสุด (α) และทิศทางของขอบภาพ (θ) หาได้จาก

$$\alpha = \tan(G_y/G_x)$$

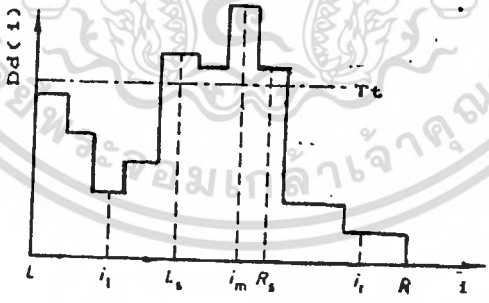
$$\theta = \alpha + \pi/2 \qquad (2.18)$$

ซึ่งอาจจะประมาณโดยสมการง่ายขึ้นดังนี้

$$G_x = (C+F+I)-(A+D+G)$$

$$G_y = (A+B+C)-(G+H+I) \qquad (2.19)$$

$$|G| = |G_x| + |G_y|$$



รูป 2.20 การพิจารณาขีดเริ่มเปลี่ยนและตำแหน่งของขอบภาพ

สำหรับขอบรูปหลังคาหรือรูปยอดแหลม จุดซึ่งเป็นขอบภาพจะใช้การกระทำแบบ Laplacian วิธีการนี้ใช้กับฟังก์ชั่นที่มีความต่อเนื่อง $f(x,y)$

$$\nabla^2 f(x,y) = \partial^2 f(x,y)/\partial x^2 + \partial^2 f(x,y)/\partial y^2 \qquad (2.20)$$

สำหรับภาพดิจิตอล วิธีการ Laplacian โดยใช้ช่องหน้าต่างขนาด 3x3 และค่าข้างเคียง 4 จุดภาพดังนี้

$$DD_2 = (B+D+F+H-4E)/4 \qquad (2.21)$$

และใช้ค่าข้างเคียง 8 จุดภาพดังนี้

$$DD_2 = (A+B+C+D+F+G+H+I-8E)/8 \qquad (2.22)$$

การหาขอบแบบ 2 มิติเหล่านี้เป็นวิธีการพื้นฐาน ซึ่งวิธีการเหล่านี้จะถูกนำไปประยุกต์ใช้แล้วแต่กรณี และช่องหน้าต่างอาจใช้ขนาดต่าง ๆ ไป เพื่อลดผลอันเกิดจากสัญญาณรบกวน

2.3.6 การพิจารณาจุดขอบภาพ[17]

จากวิธีการที่ได้อธิบายการหาจุดขอบภาพ ผลลัพธ์ที่ได้จากการกระทำดังกล่าวนั้นต้องนำมาพิจารณาต่อไปอีก จุดขอบภาพจะมีหรือไม่และถ้ามีจุดขอบภาพเกิดขึ้น จุดเหล่านั้นควรอยู่ที่ใด

วิธีการง่ายที่สุดคือการพิจารณาค่าเทรสโฮลด์ของผลลัพธ์จากการกระทำนั้น ตัวอย่างเช่น ค่า $D_{1,2}(i)$ ของรูป 2.17 ซึ่งค่าสูงสุดขึ้นอยู่กับภาพ วิธีการนี้มักจะทำการ smooth เสียก่อนเพื่อเป็นการลดผลจากสัญญาณรบกวน อันจะทำให้เกิดค่าสูงมาก ค่าผลลัพธ์ที่ได้จากการหาขอบภาพโดยไม่มีการ smooth จะมีผลกระทบจากลักษณะขึ้นลง ดังตัวอย่างจากการหาค่า $D_d(i)$ ในรูป 2.20 ขอบของภาพชนิดฟังก์ชั่นหนึ่งหน่วยถูกพิจารณาที่ช่วงระหว่าง L,R

ถ้ามีขอบของภาพเกิดขึ้นจะพิจารณาภายใต้เงื่อนไขต่อไปนี้

1. มีค่าสูงสุดของ $D_d(i)$ที่จุด i_m ซึ่ง i_m เป็นจุดใด ๆ ที่ไม่ใช่ L,R
2. จะต้องมีค่าของ $D_d(i_m) > T_a$ ซึ่ง T_a เป็นเทรสโฮลด์ความสูงของภาพที่กำหนดไว้
3. จะต้องมีค่าต่ำสุดของ $D_d(i)$ ที่ $i=i_l$ ระหว่าง L กับ i_m และค่าต่ำสุดของ $D_d(i)$ ที่ $i=i_r$ ระหว่าง i_m และ R ซึ่ง

$$D_d(i_m)-D_d(i_l) > T_r$$

$$D_d(i_m)-D_d(i_r) > T_r \qquad (2.23)$$

ค่า T_r เป็นค่าเทรสโฮลด์สัมพัทธ์ที่กำหนดไว้

ค่า T_a มักจะพิจารณาจากคุณลักษณะทั้งหมดของรูปภาพ และค่า T_r พิจารณาจากคุณลักษณะเฉพาะส่วนของภาพ ตัวอย่างเช่น ค่า $D_d(i_m)$

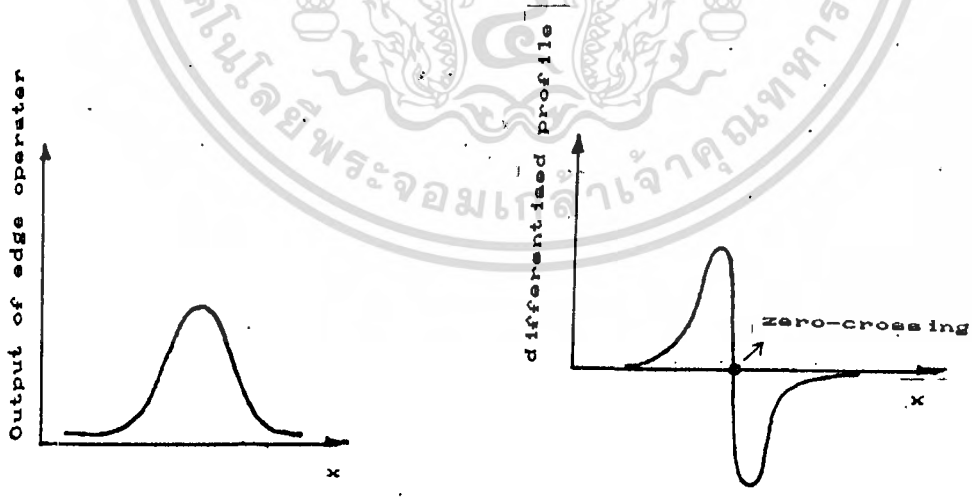
ถ้าเป็นไปได้ตามเงื่อนไขทั้งสาม หมายความว่ามีขอบภาพเกิดขึ้น ส่วนยอดที่ผ่านการทดสอบมักจะมีขอบเขตกว้าง แบนราบ สัญญาณรบกวนเพียงเล็กน้อยอาจทำให้ความกว้างของส่วนบนยอดขอบภาพขยายมากขึ้น ดังนั้นการเก็บค่าของขอบจึงไม่เก็บในรูปค่าการตัดยอด แต่จะเก็บเป็นค่าจุดกลางระหว่างช่วงกว้างของความคมขอบภาพ ที่ขอบซ้าย L_e และขอบขวา R_e เป็นค่าที่ใกล้ที่สุดของจุดตัดระหว่าง $D_d(i)$ และเส้นตามแนวนอนของ T_f ดังรูป 2.20 ค่า T_f พิจารณาจากสมการต่อไปนี้

$$T_f = C_1 D_d(i_m) + C_o \qquad (2.24)$$

ซึ่ง $C_o(>0)$ และ $C_1(0<C_1<1)$ เป็นค่าคงที่

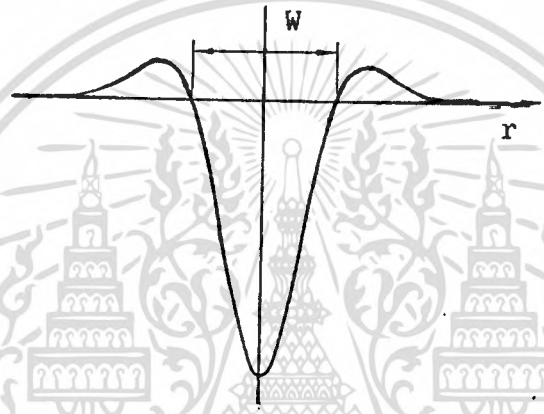
2.3.7 การเปลี่ยนแปลงที่ตำแหน่งข้ามศูนย์(Zero-Crossing)

หลักการของการเปลี่ยนแปลงที่ตำแหน่งข้ามศูนย์เป็นการหาค่าลายเส้นความแตกต่างของระดับความเข้มแสงที่ผ่านจุดศูนย์ แทนการหาค่าสูงสุดตามที่ได้อธิบายในหัวข้อที่ผ่านมา การหาขอบภาพในหนึ่งมิติเป็นการหาค่าความแตกต่างของระดับแสง ซึ่งเพิ่มค่าอย่างรวดเร็วทำให้เกิดรูปทรงที่มียอดสูงสุดตรงกลางในอนุพันธ์อันดับแรก และวิธีการเปลี่ยนแปลงที่ตำแหน่งข้ามศูนย์ เป็นการกระทำในอนุพันธ์อันดับสอง ซึ่งเมื่อขยายเป็นระดับแสงในรูปสองมิติ คุณสมบัติดังกล่าวยังคงเป็นจริง



รูป 2.21 การหาขอบภาพจากตำแหน่งข้ามศูนย์

ถึงแม้ว่าสามารถกำหนดหลักการที่เหมาะสมเกี่ยวกับทิศทางของอนุพันธ์ เพื่อประยุกต์ใช้ในรูปภาพได้ แต่ในทางปฏิบัตินั้นเป็นการสิ้นเปลือง เนื่องจากต้องมีการพิจารณากลุ่มของการกระทำในแต่ละทิศทาง รวมทั้งการคอนโวลูชั่นซึ่งสิ้นเปลืองเวลาในการคำนวณมาก การใช้วิธีการกระทำแบบไม่คิดทิศทางเป็นวิธีการหนึ่งที่ลดการคำนวณคอนโวลูชั่นได้ และวิธีการอนุพันธ์อันดับสองที่ไม่คิดทิศทางคือวิธี Laplacian สามารถนำมาใช้แทนการอนุพันธ์แบบคิดทิศทาง อย่างไรก็ดีในทางปฏิบัติรูปภาพนั้น



รูป 2.22 ภาพตัดขวางของ $\nabla^2 G(r)$

ควรถูกเตรียมการโดยการ smooth รูปภาพ โดยการคอนโวลูชั่นกับฟังก์ชั่นของเกาส์เซียนเสียก่อน สมการ Gaussian distribution ของสองมิติแทนได้ด้วยสมการ

$$G(x,y) = (1/2\pi\sigma^2)\exp\{-(x^2+y^2)/2\sigma^2\} \qquad (2.25)$$

ซึ่ง σ เป็นค่าของความเบี่ยงเบนมาตราฐาน ดังนั้นการกระทำทั้งหมดกับรูปภาพจริงมีรูปแบบดังนี้

$$\nabla^2[G(x,y)*f(x,y)] = \nabla^2 G(x,y)*f(x,y) \qquad (2.26)$$

ค่า $\nabla^2 G(x,y)$ แทนค่าด้วย

$$\nabla^2 G(r) = \{(r^2-2\sigma^2)/2\pi\sigma^6\}\exp(-r^2/2\sigma^2) \qquad (2.27)$$

ซึ่ง $r = (x^2+y^2)^{1/2}$

ฟังก์ชั่นที่ได้เป็นฟังก์ชั่นที่เหมือนกันโดยรอบด้านดังแสดงในรูป 2.22 เส้นผ่าศูนย์กลางของวงกลมส่วนที่

เป็นค่าลบในรูปมีค่า

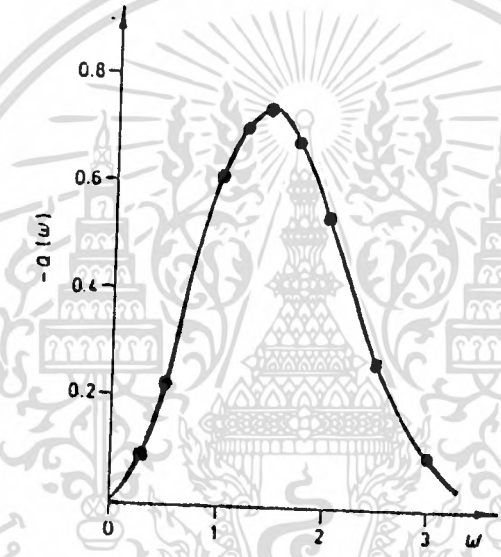
$$w = 2(2\sigma)^{1/2} \tag{2.28}$$

ขนาดของช่องหน้าต่างที่ใช้หาขอบโดยวิธีนี้ กำหนดได้จาก w หรือ σ

คุณลักษณะทางความถี่ของการดำเนินการวิธีนี้ สามารถวิเคราะห์ได้โดยการใช้การแปลงค่าแบบฟูเรียร์ 2 มิติ ในสมการที่ 2.26 จะได้ว่า

$$F\{[\nabla^2 G(x,y)] * f(x,y)\} = F\{\nabla^2 G(x,y)\}\, F\{f(x,y)\} \tag{2.29}$$

พจน์แรกทางขวามือจะได้ดังนี้



รูป 2.23 ฟูเรียร์ทรานฟอร์มของ $\nabla^2 G$

$$F\{\nabla^2 G(x,y)\} = Q(u,v) = -(u^2+v^2)\exp(-\sigma^2(u^2+v^2)/2)$$

$$= -w^2\exp(-\sigma^2 w^2/2) \tag{2.30}$$

ซึ่ง $w^2=u^2+v^2$ รูป 2.23 แสดงฟังก์ชั่น $-Q(w)$ จะเห็นว่าการกระทำของสมการเป็นไปในลักษณะของวงจรกรองผ่านแถบมีศูนย์กลางความถี่ที่ $w=(2/\sigma)^{1/2}$

ค่าการเปลี่ยนแปลงที่ตำแหน่งข้ามศูนย์ของภาพที่ปราศจากสัญญาณรบกวนจะให้ค่าตอบสนองตามขอบของภาพนั้น แต่อย่างไรก็ดีการเปลี่ยนแปลงเพียงเล็กน้อยในพื้นผิวเนื้อเดียวกันย่อมทำให้เกิดค่าการเปลี่ยนแปลงที่ตำแหน่งข้ามศูนย์ได้ เพื่อหลีกเลี่ยงผลของการรบกวนดังกล่าวจึงไม่ควรใช้วิธีการนี้กับระดับความเข้มแสงที่มีการเปลี่ยนแปลงน้อย เราสามารถจะตรวจสอบเพื่อเลือกวิธีการหาค่า

ความแตกต่าง f(x,y) หรือ G(x,y)*f(x,y) ได้ ถ้าค่าความแตกต่าง f(x,y) มีค่าน้อยมาก ในกรณีนี้ไม่ควรใช้วิธีของการเปลี่ยนแปลงตำแหน่งข้ามศูนย์

2.3.8 การติดตามขอบวัตถุโดยใช้หลักการหมุนของเวกเตอร์

การตรวจหาจุดขอบของวัตถุทำได้โดยอาศัยกฎเกณฑ์โดยสรุปดังนี้ เมื่อพบจุดภาพที่ไม่ใช่จุดภาพของวัตถุให้หมุนตัวไปทางซ้ายหนึ่งตำแหน่ง เมื่อพบจุดภาพที่เป็นจุดภาพของวัตถุให้หมุนตัวไปทางขวา

รูป 2.24 การหาจุดภาพโดยวิธีการหมุนซ้ายเมื่อไม่พบวัตถุและหมุนขวาเมื่อพบวัตถุ

หนึ่งตำแหน่งดังรูป 2.24 ด้วยหลักการดังกล่าวทำให้ได้ขอบภาพที่หนาเกินความจำเป็น เมื่อใช้เทคนิคของการหมุนเวกเตอร์ในแบบ 8 ทิศทาง โดยใช้ช่องหน้าต่างขนาด 3x3 วางทาบไปบนจุดภาพที่เป็นจุดขอบของวัตถุ และเริ่มหมุนซ้ายหรือขวาในทิศทางใดทิศทางหนึ่งเท่านั้น ซึ่งต้องใช้ทิศทางเดียวตลอดกระบวนการทั้งหมด

การใช้ทฤษฎีของคอมพิวเตอร์กราฟฟิก มาประยุกต์เพื่อกำหนดตำแหน่งของจุดภาพเมื่อได้หมุนไปนั้น เราใช้จากสมการสำหรับการหมุนทวนเข็มนาฬิกาเป็นมุม θ

$$(x',y') = (x\cos\theta - y\sin\theta,\quad x\sin\theta + y\cos\theta) \qquad (2.31)$$

และสมการสำหรับหมุนตามเข็มนาฬิกาเป็นมุมθ

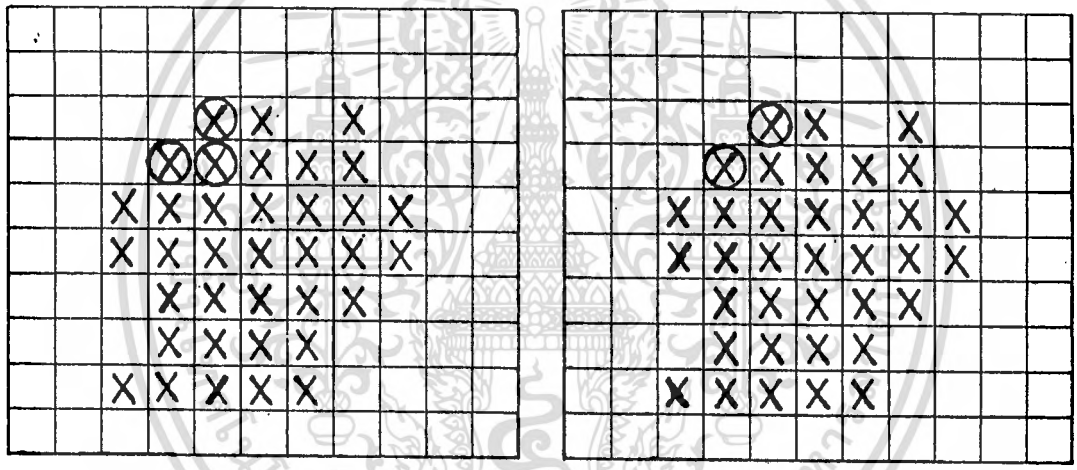
$$(x',y') = (x\cos\theta + y\sin\theta,\quad -x\sin\theta + y\cos\theta) \qquad (2.32)$$

โดยที่ x,y เป็นจุดพิกัดเดิมก่อนการหมุนโดยรอบจุดพิกัด 0,0 และ x',y' เป็นจุดพิกัดหลังจากการหมุน เนื่องจากค่าตำแหน่งของช่องหน้าต่างเป็นจำนวนเต็ม และกำหนดพิกัดเฉพาะในตาราง 3x3 เท่านั้น

ค่าที่อยู่ตรงกลางของช่องตาราง ให้เป็นค่า i,j (i=-y,j=x) ดังนั้นเราจะกำหนดตำแหน่งของตาราง i',j' หลังการหมุนทวนเข็มนาฬิกาได้จากความสัมพันธ์ของเครื่องหมายในสมการที่ 2.31 และการหมุนตามเข็มนาฬิกาของสมการที่ 2.32 และค่า i,j ที่เปลี่ยนไปมีค่าเพิ่มขึ้นครั้งละ ±1 เท่านั้น

การใช้วิธีการหมุนเวกเตอร์ 8 ทิศทางนี้ก็คือการหมุนไปครั้งละ 45 องศาเสมอ ซึ่งอาจสรุปวิธีการได้ดังนี้

1. สแกนหาจุดขอบแรกของวัตถุ
2. กำหนดทิศทางหมุนซ้าย(หรือขวา)
3. ทาบตาราง 3x3 โดยให้จุดกลางทับอยู่บนขอบภาพแรก



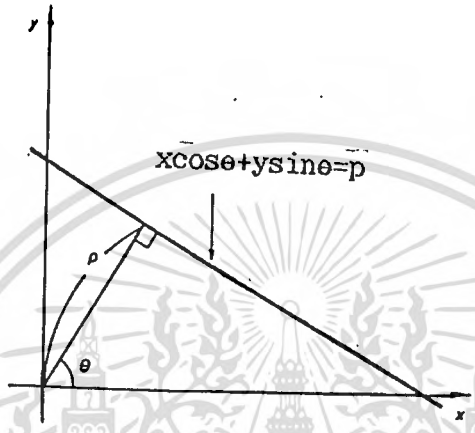
รูป 2.25 เปรียบเทียบระหว่างวิธีเดิมและวิธีการหมุนเวกเตอร์

4. ลากเวกเตอร์จากจุดกลางช่องหน้าต่างไปทางซ้าย(หมุนทวนเข็มนาฬิกา 180 องศา)
5. ถ้าตำแหน่งจุดภาพที่หมุนไปไม่ใช่จุดภาพ ให้หมุนทวนเข็มนาฬิกาต่อไปครั้งละ 45 องศา
6. ถ้าหมุนครบ 8 ทิศไม่พบจุดภาพแสดงว่า จุดแรกที่พบนั้นไม่เชื่อมต่อกับจุดภาพใดเลย
7. เมื่อพบจุดภาพให้เลื่อนช่องหน้าต่างมาที่จุดภาพใหม่ ให้จุดกลางช่องทับจุดภาพใหม่พอดี
8. ลากเวกเตอร์จากจุดภาพใหม่กลับไปสู่จุดขอบภาพเดิม(เพื่อหาตำแหน่งเริ่มต้นการหมุน)
9. ใช้จุดขอบภาพใหม่เป็นศูนย์กลางการหมุน ตามช่องหน้าต่างที่วางทาบไว้
10. จากแนวเวกเตอร์ในข้อ 8 หมุนเวกเตอร์ตามข้อ 9 ทวนเข็มนาฬิกาเพิ่มขึ้นต่อไปครั้ง

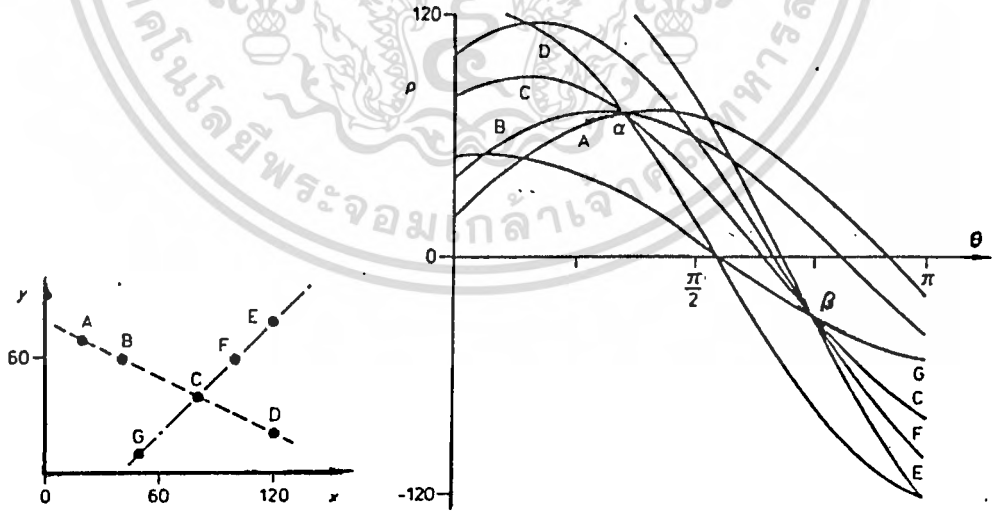
ละ 45 องศาจนกว่าจะพบจุดขอบภาพใหม่(ถ้าจุดภาพเป็นจุดขอบภาพแรกถือว่าสิ้นสุดการตามขอบภาพ)

11. ทำตามขั้นตอนที่ 7 ถึง 10 ซ้ำต่อไปจนกว่าจะพบจุดขอบภาพจุดแรก

ผลของวิธีการหมุนเวกเตอร์ตามแนวช่องหน้าต่าง 3x3 ทำให้สามารถลดจำนวนจุดของขอบภาพลงได้



รูป 2.26 ความหมายทางเรขาคณิตของสมการเส้นตรง



รูป 2.27 การหาเส้นตรงโดยการแปลงพารามิเตอร์ของ Hough

จุดภาพบนระนาบ x-y และแนวทางเดินในมิติของพารามิเตอร์

2.3.9 การจัดกลุ่มขอบภาพโดยการแปลงพารามิเตอร์[17]

ถ้ามีจำนวนจุดขอบภาพอยู่บนระนาบของรูปภาพ การที่จะแยกจุดของเส้นขอบต่าง ๆ ว่าเป็นเส้นขอบของกลุ่มใด ถ้ากำหนดให้ขอบภาพเป็นเส้นตรงซึ่งมีสมการอยู่ในรูปของพารามิเตอร์ ถ้าจุดขอบภาพเหล่านี้ถูกประมาณด้วยการแปลงค่าไปอยู่ในรูปของพารามิเตอร์อื่น เพื่อจุดประสงค์ในการจัดกลุ่มเส้นขอบ วิธีการแรกคิดโดย Hough สำหรับการหาเส้นตรง ซึ่งต่อมาสามารถขยายวิธีการที่จะหาคุณลักษณะอื่น ๆ เช่น รูปทรงโค้งหรือรูปทรงที่ถูกกำหนดไว้ .

การหาเส้นตรงจากกลุ่มของจุดขอบโดยไม่อาจกำหนดทิศทาง จึงแทนสมการเส้นตรงด้วย $y = ax + b$ และต่อมา Duda และ Hart (1972)[17] ได้แทนด้วยสมการใหม่ดังนี้

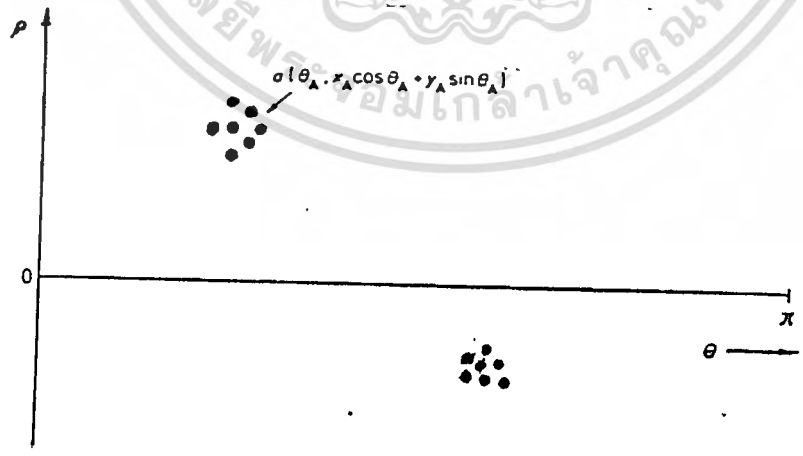
$$x \cos\theta + y \sin\theta = p \qquad (2.33)$$

พารามิเตอร์ของสมการใหม่คือ θ และ p ซึ่งเป็นสิ่งที่กำหนดการเปลี่ยนแปลงทิศทางของเส้นตรง ถ้า p ในสมการ 2.33 เป็นค่าลบได้ ดังนั้นเส้นตรงทุกเส้นถูกแทนค่าด้วย p และ θ $(0 < \theta < \pi)$

สมมุติว่าจุดภาพ A อยู่บนระนาบของภาพ เส้นตรงทุกเส้นที่ผ่านจุด A สามารถแทนได้ด้วยสมการ

$$x_A \cos\theta + y_A \sin\theta = p \qquad (2.34)$$

ซึ่ง (x_A, y_A) คือตำแหน่งพิกัดของ A



รูป 2.28 การแปลงขอบภาพโดยวิธีของ Hough ซึ่งรู้ทิศทาง

ถ้าเราจะพิจารณาในระบบ 2 มิติบนแกนพารามิเตอร์ p และ θ ดังรูป 2.27 เส้นตรงหนึ่งเส้นบนระนาบของภาพ จะถูกแทนด้วยจุดหนึ่งจุดบนแกนมิติของพารามิเตอร์ p และ θ และเส้นตรงทุกเส้นที่ผ่านจุด A จะวางอยู่บนเส้นโค้งรูปซายน์ที่ได้จากสมการ 2.34

สมมุติว่า A,B,C และ D วางอยู่บนเส้นตรงเดียวกันในภาพ เส้นตรงทุกเส้นที่ผ่านจุดเหล่านี้ จะถูกแปลงในมิติของพารามิเตอร์เป็นเส้นโค้งที่มีความคล้ายคลึงกัน และเส้นโค้งทุกเส้นเหล่านี้จะผ่านที่จุดเดียวกัน(ที่จุด α ในรูป 2.27)เพื่อเป็นที่สังเกตว่าเป็นเส้นตรงในระนาบของภาพ ส่วนจุดอื่นที่ปรากฏคือ β แทนกลุ่มของจุดซึ่งวางอยู่บนเส้นตรงอีกเส้นหนึ่งบนระนาบภาพ

กลุ่มของจุดบนเส้นตรงเดียวกันอาจไม่ตัดกันที่จุดเดียวกันเนื่องจากการรบกวนหรือผลของการแปลงเป็นสัญญาณดิจิตอลในการคำนวณด้วยคอมพิวเตอร์

วิธีการของ Hough สามารถขยายขอบเขตการพิจารณาทิศทางของขอบภาพ สมมุติว่าทิศทางของขอบภาพ A สามารถรู้ได้ (θ_A) สมการที่ผ่านจุดนี้จะพิจารณาได้เหมือนกันคือ

$$x \cos \theta_A + y \sin \theta_A = x_A \cos \theta_A + y_A \sin \theta_A \qquad (2.35)$$

เส้นตรงทั้งเส้นจะถูกแปลงไปเป็น $a(\theta, x \cos \theta_A + y \sin \theta_A)$ ดังรูป 2.27 ทำให้สามารถจัดกลุ่มได้ง่าย

การประยุกต์ด้านอื่นเช่นการหาขอบเส้นโค้งซึ่งรู้ค่าพารามิเตอร์ สมมุติว่าสมการเส้นโค้งนั้นเขียนได้เป็น

$$y = ax^2 + c \qquad (2.36)$$

ค่าพารามิเตอร์ใหม่บนแกนมิติ a และ c จะต้องกำหนด สำหรับแต่ละจุดของขอบภาพ จะเกิดลายเส้นโค้งบนแกนมิติของพารามิเตอร์

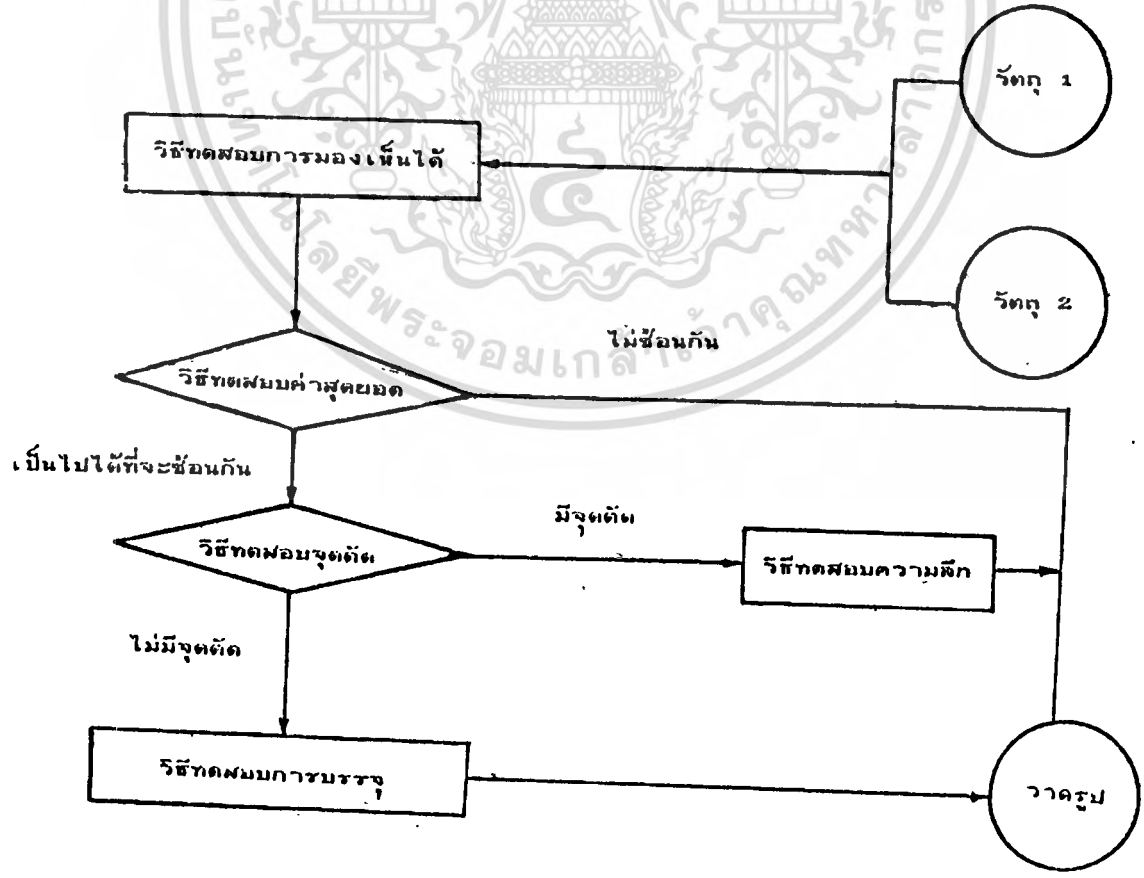
บทที่ 3

คอมพิวเตอร์กราฟฟิก

ในการวิเคราะห์ภาพตามธรรมชาตินั้น ต้องคำนึงถึงสภาวะทางฟิสิกซ์ เช่น ตำแหน่ง โครงสร้าง สี และรายละเอียดอื่น ๆ ซึ่งปรากฏบนภาพแบบสองมิติ เน้นถึงลักษณะของคอมพิวเตอร์กราฟฟิกเกี่ยวกับการเปลี่ยนรูปแบบตามหลักการของเรขาคณิต รวมทั้งส่วนของเส้นต่าง ๆ ประกอบด้วยส่วนที่ถูกบังไว้จากสายตาและส่วนที่มองเห็นได้ ระดับความเข้มของแสง ย่อมสัมพันธ์กับสภาพทางเรขาคณิตของรูปทรงวัตถุ ในเบื้องต้นนี้เราพิจารณาถึงรูปแบบของการมองเห็นภาพ ซึ่งง่ายต่อการเข้าใจ

3.1 เส้นขอบของวัตถุในระบบ 3 มิติ[14]

การที่จะพิจารณาพื้นผิวของวัตถุหรือส่วนของเส้นต่าง ๆ ที่ปรากฏบนภาพวัตถุ 3 มิติ ว่าส่วนใดเป็นส่วนที่มองเห็นได้ และส่วนใดที่ถูกบังไว้จากการมองจากมุมต่าง ๆ ย่อมเป็นการยากที่จะคาดคะเนและอาจเกิดการสับสนได้ เมื่อมีรูปวัตถุหรือภาพลายเส้นอยู่แล้ว การที่จะเขียนภาพส่วนอื่นเพิ่มเติม จำเป็นต้องพิจารณาถึงส่วนของวัตถุหรือภาพเหล่านั้นว่ามีส่วนที่ซ้อนกันหรือไม่ กระบวนการดังนี้



รูป 3.1 แผนภูมิการทดสอบการมองเห็นภาพวัตถุ

กล่าวอาจแบ่งได้เป็น 3 ขั้นตอนดังนี้

1. ส่วนของบริเวณภาพนั้นสามารถมองเห็นหรือไม่

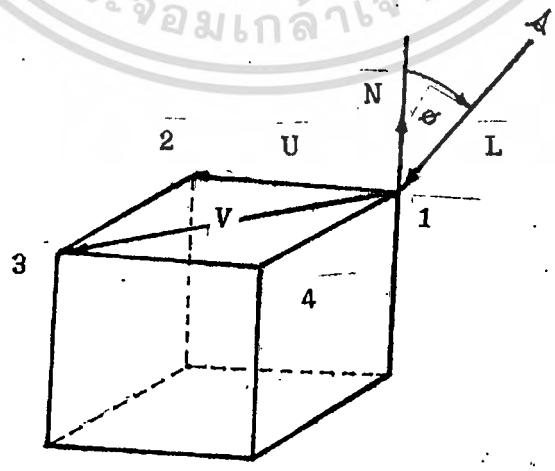
2. ถ้าเป็นส่วนที่มองเห็นได้ ส่วนดังกล่าวมีการซ้อนหรือบังกันจากจุดมุมมองหรือไม่

3. ถ้าเป็นส่วนที่มีการซ้อนกัน จะต้องพิจารณาหาจุดตัด ทดสอบความลึกของภาพและสำดับก่อนหลังของตำแหน่งภาพวัตถุ หากไม่มีจุดตัดเกิดขึ้นจะต้องพิจารณาต่อไปว่ามีภาพวัตถุใดซ้อนอยู่ภายในภาพวัตถุอื่นหรือไม่ หากไม่ปรากฏกรณีเหล่านี้ย่อมหมายความว่าภาพวัตถุทั้งสองไม่มีความสัมพันธ์ใด ๆ ต่อกัน เราสามารถเขียนแผนภูมิแสดงขั้นตอนการทดสอบได้ดังรูปที่ 3.1

3.1.1 วิธีทดสอบการมองเห็น

ทำการทดสอบโดยกำหนดให้จุดตำแหน่งการมองเป็นจุด P มีพิกัด (D,θ,ϕ) และให้รูปสี่เหลี่ยมลูกบาศก์เป็นวัตถุที่จะทำการทดสอบว่ามีผิวหน้าที่มองเห็นจำนวนเท่าใด ให้ศูนย์กลางการมองเป็นจุดยอดที่ 1 และให้นับจุดยอดต่อไปในทิศทางทวนเข็มนาฬิกาเป็นจุดที่ 2,3,4 ตามลำดับ เส้นตรงที่ตั้งฉากกับผิวระนาบที่จุด 1 แทนด้วยเวกเตอร์ N เส้นตรงที่ลากจากจุด P ไปยังจุด 1 แทนด้วยเวกเตอร์ L เส้นตรงที่ลากจากจุด 1 ไปจุด 2 และจุด 1 ไปจุด 3 แทนด้วยเวกเตอร์ U และ V ตามลำดับมุมเวกเตอร์ N กับ L แทนด้วยมุม ϕ ซึ่งค่าต่าง ๆ แสดงดังรูปที่ 3.2

เวกเตอร์ U เป็นค่าเวกเตอร์ซึ่งได้จากการใช้ตำแหน่งของจุดปลายลบด้วยจุดเริ่มต้น แทนค่าของเวกเตอร์ด้วย (a,b,c)



รูปที่ 3.2 เวกเตอร์ที่ใช้ในการทดสอบบริเวณพื้นที่ซึ่งมองเห็น

เวกเตอร์ V แทนค่าด้วย (d,e,f)

ค่าของ N จะหาได้จาก

$$U \times V = (bf-ce, cd-af, ae-bd) \tag{3.1}$$

ค่าของ L จะหาได้จาก

$$L = (D \sin\phi \cos\theta, D \sin\phi \sin\theta, D \cos\phi) - (x_1, y_1, z_1) \tag{3.2}$$

ค่า x_1, y_1, z_1 เป็นค่าตำแหน่งของจุด 1

ค่ามุม ϕ หาได้จาก

$$\phi = \arccos[N.L/(|N|.|L|)] \tag{3.3}$$

เมื่อ

$$N.L = (\bar{n}_1, \bar{n}_2, \bar{n}_3).(\bar{l}_1, \bar{l}_2, \bar{l}_3)$$

$$= n_1 l_1 + n_2 l_2 + n_3 l_3$$

เมื่อ $|N|$ เป็นความยาวของเวกเตอร์ N และ $|L|$ เป็นความยาวของเวกเตอร์ L ถ้าผลของการคำนวณได้ค่ามุม ϕ มีค่ามากกว่า 0 และไม่เกิน 90 องศา หมายความว่าพื้นที่บริเวณนั้นเป็นบริเวณพื้นผิวที่สามารถมองเห็นได้ แต่ถ้าผลการคำนวณได้ผลของมุม ϕ น้อยกว่า 0 หรือมากกว่า 90 องศา พื้นผิวนั้นเป็นพื้นผิวที่ถูกบังอยู่ไม่สามารถมองเห็นได้ โดยปกติรูปลูกบาศก์นั้นสามารถมองเห็นได้ถึง 3 ผิวหน้าจากจุดกำหนดการมองที่ตำแหน่งหนึ่ง จำนวนผิวหน้าซึ่งมองเห็นได้สามารถทดสอบได้จากวิธีการที่ได้อธิบายมาแล้ว

3.1.2 ค่าสุดยอดของภาพ

กำหนดภาพวัตถุ A และ B เป็นวัตถุที่จะทำการทดสอบ การพิจารณานั้นจะต้องทำการโปรเจคภาพทั้งสองให้อยู่ในระนาบเดียวกัน ค่ามากสุด (max) และค่าต่ำสุด (min) ของภาพวัตถุทั้งสองจะถูกนำมาเปรียบเทียบกันตามเงื่อนไขต่อไปนี้

1. $\max x_A < \min x_B$
2. $\max x_B < \min x_A$
3. $\max y_A < \min y_B$
4. $\max y_B < \min y_A$

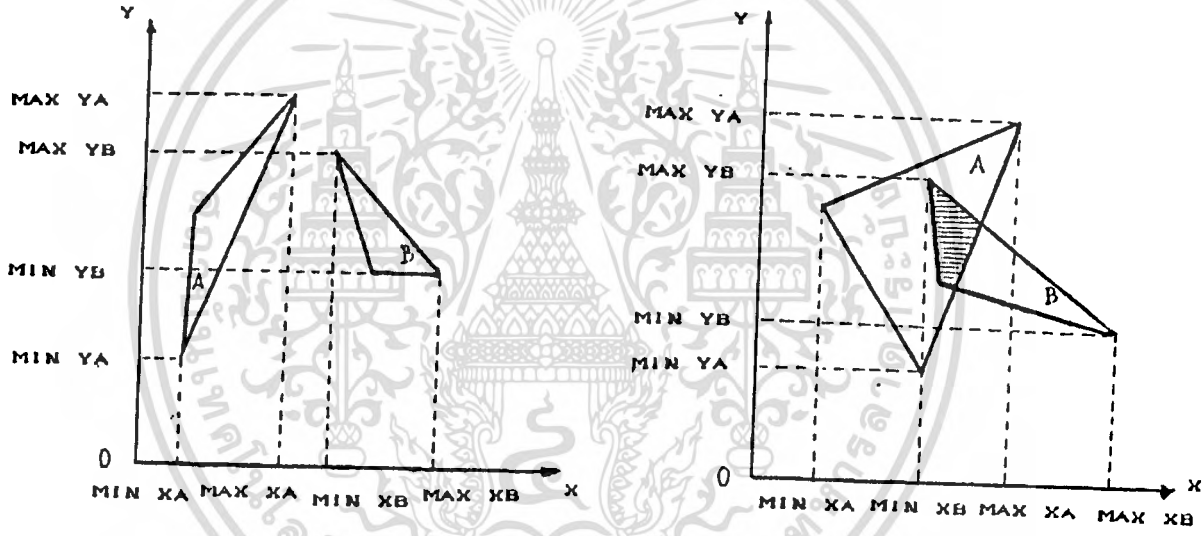
โดยที่ค่า x_A, x_B เป็นค่าตามแนวแกน X ของวัตถุ A และ B ตามลำดับ

y_A, y_B เป็นค่าตามแนวแกน Y ของวัตถุ A และ B ตามลำดับ

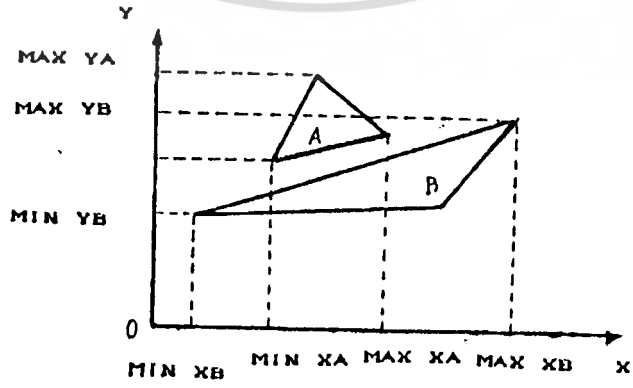
ถ้าค่าการเปรียบเทียบพบว่า เงื่อนไขใดเป็นจริงแสดงว่าภาพวัตถุทั้งสองนั้นไม่ซ้อนกัน แต่ถ้าพบว่าไม่มีสภาวะใดเป็นจริงเลย ย่อมแสดงว่าวัตถุทั้งสองอาจจะซ้อนกัน ซึ่งจะต้องทดสอบในขั้นตอนต่อไป เช่นในบางกรณีแม้ทุกสภาวะของเงื่อนไขไม่เป็นจริง วัตถุทั้งสองไม่ซ้อนกันเลยดังรูป 3.4

3.1.3 จุดตัดของเส้นขอบ

เราสามารถสร้างสมการทางพิชคณิตเพื่อหาจุดตัดของเส้นตรงสองเส้นได้โดย ให้เส้นตรง L มีสมการเป็น $A_1x+B_1y+C_1y$ และสมการของเส้นตรง L' มีสมการเป็น $A_2x+B_2y+C_2y$



รูป 3.3 การพิจารณาการซ้อนของภาพจากค่าสุดยอด



รูป 3.4 กรณีที่เงื่อนไขการทดสอบไม่เป็นจริงแต่ภาพวัตถุไม่ซ้อนกัน

โดยที่ A_1, B_1, A_2 และ B_2 มีค่าไม่เท่ากับ 0

พิจารณาความเป็นไปได้ของสมการทั้งสองใน 3 กรณี ดังนี้

1. ถ้า $A_1B_2 - B_1A_2 = 0$ แล้วเส้นตรงทั้งสองจะขนานกันและไม่มีจุดตัดกัน

2. ถ้า $A_1/A_2 = B_1/B_2 = C_1/C_2$ แล้ว เส้นตรงทั้งสองเส้นจะเป็นเส้นตรงเดียวกันและจะตัดกันที่อนันต์

3. หากไม่เป็นไปตามสองกรณีแรกแล้ว เส้นตรงทั้งสองเส้นจะตัดกันที่จุด (x_1, y_1) โดยที่ค่าของ

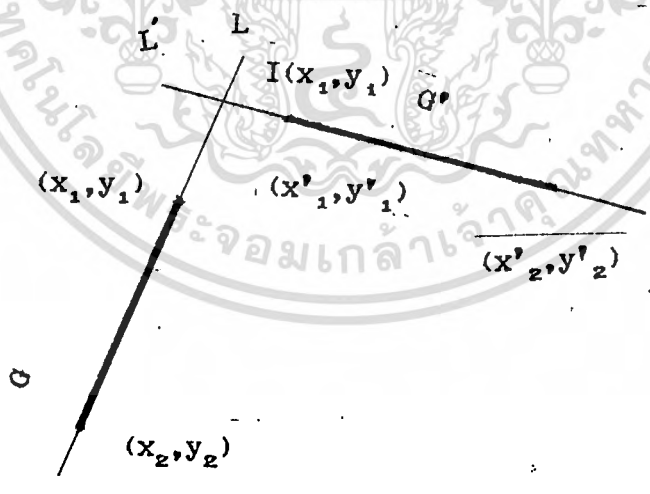
$$x_1 = (B_1C_2 - B_2C_1)/(A_1B_2 - A_2B_1)$$

$$y_1 = (C_1A_2 - C_2A_1)/(A_1B_2 - A_2B_1) \qquad (3.4)$$

การหาจุดตัดกันของเส้นตรง จะทำได้โดยสมมุติให้ G และ G' เป็นส่วนของเส้นตรง L และ L' เส้นตรง G และ G' เป็นเส้นตรงอยู่ในระหว่างพิกัด (x_1, y_1) กับ (x_2, y_2) และ พิกัด (x'_1, y'_1) กับ (x'_2, y'_2) ตามลำดับ และให้จุดตัดของเส้นตรงตัดกันที่จุด I มีพิกัดอยู่ที่ (x_1, y_1) ซึ่งเส้นตรง G และ G' จะไม่ตัดกันถ้ามีสภาวะดังต่อไปนี้

$$x_1 < A_1, \quad x_1 > B_1, \quad x_1 < C_1, \quad x_1 > D_1$$

$$y_1 < A_2, \quad y_1 > B_2, \quad y_1 < C_2, \quad y_2 > D_2$$



รูป 3.5 จุดตัดของเส้นขอบภาพ

ซึ่ง

$$A_1 = \min(x_1, x_2); \; B_1 = \max(x_1, x_2); \; C_1 = \min(x'_1, x'_2); \; D_1 = \max(x'_1, x'_2)$$

$$A_2 = \min(y_1, y_2); \; B_2 = \max(y_1, y_2); \; C_2 = \min(y'_1, y'_2); \; D_2 = \max(y'_1, y'_2)$$

นอกจากสภาวะดังกล่าวนี้ ส่วนของเส้นตรง G และ G' จะตัดกันที่จุด $I(x_1,y_1)$ ดังรูป 3.5

เราไม่ได้พิจารณาแต่เพียงว่าเส้นขอบของรูปเหลี่ยมทั้งสองตัดกันหรือไม่ แต่ย่อมต้องพิจารณาหาตำแหน่งของจุดซึ่งเส้นขอบทั้งสองตัดกันด้วย เส้นขอบของภาพรูปเหลี่ยมทั้งสองนั้นเราหาได้จากตำแหน่งของจุดปลายทั้งสองของขอบภาพ หรือเป็นยอดเหลี่ยมของภาพวัตถุนั้น ถ้าจุด (x_1,y_1) และ (x_2,y_2) เป็นจุดปลายของเส้นขอบบนรูปเหลี่ยม A สมการเส้นตรงของเส้นขอบนี้เขียนได้เป็น

$$y = y_1+m(x-x_1)$$

ซึ่ง

$$m = (y_2-y_1)/(x_2-x_1) \qquad (3.5)$$

ในทำนองเดียวกัน

$$m'= (y'_2-y'_1)/(x'_2-x'_1) \qquad (3.6)$$

เพื่อที่จะหาค่าตัดกันของจุดตัดกันของเส้นขอบของรูปเหลี่ยมทั้งสองเป็นขั้นตอนดังนี้

ขั้นตอน 1 คำนวณหาค่า $m'-m$ ถ้าค่า $m'-m = 0$ ดังนั้นเส้นขอบทั้งสองเส้นจะขนานกัน และให้ข้ามไปยังขั้นตอนที่ 5 ส่วนกรณีอื่นให้ทำขั้นตอนที่ 2 ต่อไป

ขั้นตอน 2 ถ้า $m/m'= 1 = (y_1-mx_1)/(y'_1-m'x'_1)$ แสดงว่าเส้นตรงทั้งสองทับกัน ให้ไปทำในขั้นตอนที่ 5 ถ้ากรณีอื่นไปทำขั้นตอน 3

ขั้นตอน 3 คำนวณหาจุดตัดจากสมการ

$$x_1 = [(y_1-y'_1)-(mx_1-m'x'_1)]/(m'-m)$$

$$y_1 = [(y_1-mx_1)m'-(y'_1-m'x'_1)m]/(m'-m) \qquad (3.7)$$

ขั้นตอน 4 ใช้เงื่อนไขดังที่อธิบายมาแล้วพิจารณาว่า เส้นขอบทั้งสองได้ตัดกันที่จุด $I(x_1,y_1)$

ขั้นตอน 5 เลือกเส้นขอบเส้นอื่นของรูปเหลี่ยม B และทำกระบวนการซ้ำเช่นเดิมจนครบทุกเส้นขอบของการเปรียบเทียบรูปเหลี่ยมทั้งสอง จึงเสร็จสิ้นกระบวนการหาจุดตัด

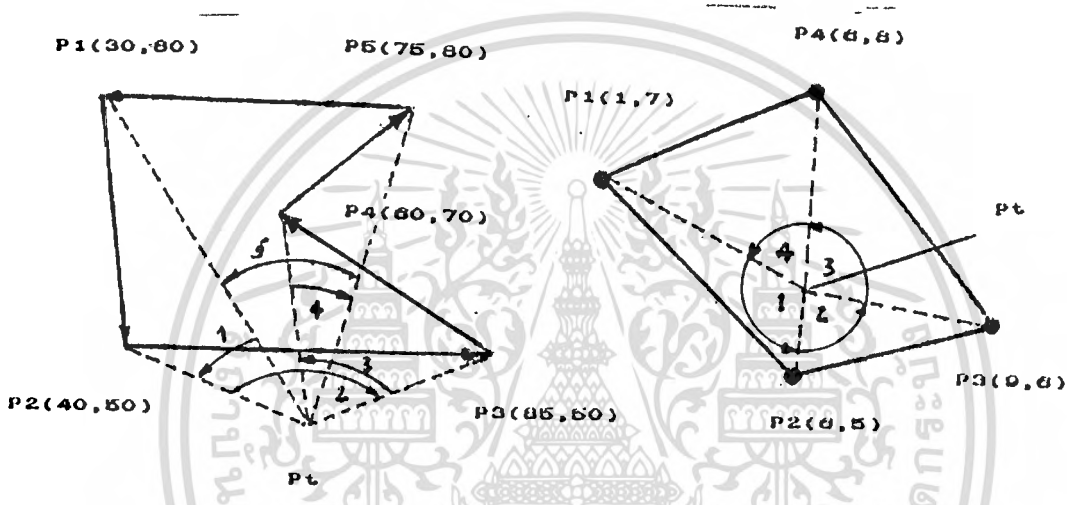
3.1.4 บริเวณภาพซึ่งบรรจุในภาพอื่น

เมื่อไม่มีจุดตัดกันระหว่างรูปเหลี่ยมสองรูป ย่อมหมายถึงรูปใดรูปหนึ่งบรรจุอยู่ภายในอีกรูปหนึ่ง หรือมิฉะนั้นรูปเหลี่ยมทั้งสองไม่มีส่วนซ้อนกันเลย เพื่อจะแยกสองกรณีให้เห็นชัดเจน จะต้องทดสอบยอดมุมต่าง ๆ ของรูปเหลี่ยมรูปซึ่งซ้อนอยู่ในรูปเหลี่ยมรูปอื่น

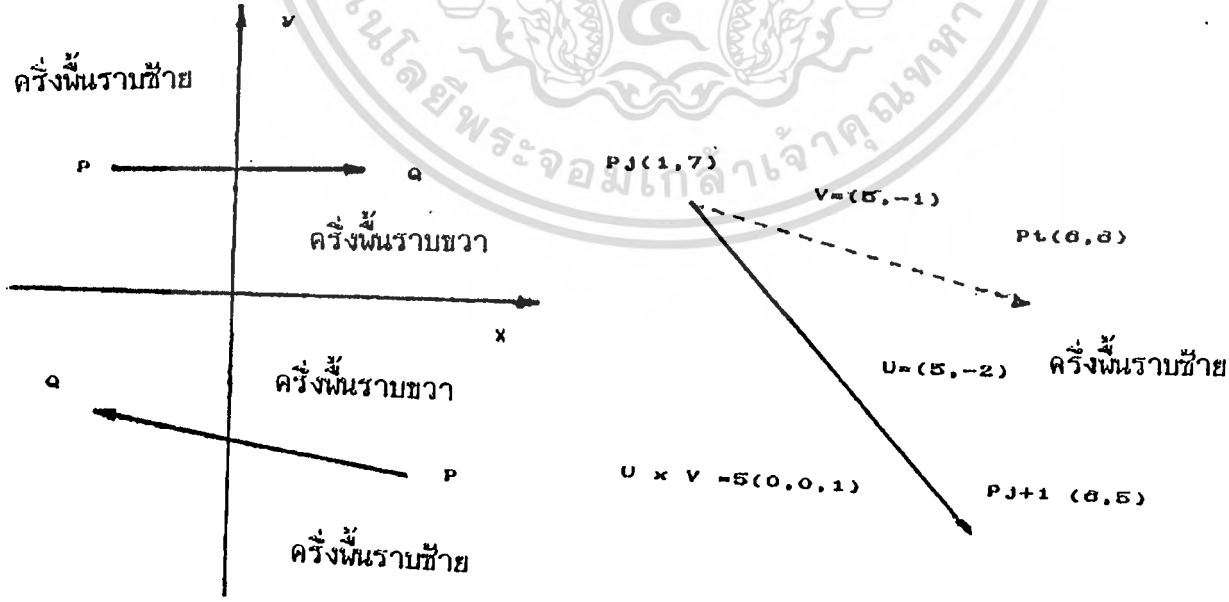
1. ทดสอบโดยคำนวณผลรวมของมุม

เราใช้การรวมค่าจากมุมยอดที่เกิดจากจุดทดสอบซึ่งเป็นยอดเหลี่ยมบนรูปเหลี่ยม A

ลากไปสู่มุมยอดต่าง ๆ ของรูปเหลี่ยม B กำหนดตำแหน่งของจุดยอดบนรูปเหลี่ยม B ให้เป็นลำดับเรียงทวนเข็มนาฬิกา วัดค่าของมุมจากเส้นต่อมุมยอดแรกบนรูป B กับจุดยอดทดสอบของ A ไปสู่เส้นที่ลากจากจุดที่สองและวัดค่ามุมจากเส้นลากจุดยอดมุมที่สอง (ไปสู่จุดยอดทดสอบ) กับจุดที่ลากจากมุมยอดของ B ต่อ ๆ ไป จนกลับมาสู่เส้นแรก ค่ามุมซึ่งทวนเข้มนาฬิกาถือเป็นค่าบวก มุมตามเข็มนาฬิกาเป็นค่าลบ ผลรวมของมุมทั้งหมดถ้าเท่ากับ 0 หมายความว่าจุดยอดของ A ซึ่งเป็นจุดทดสอบนั้นอยู่ภายนอกรูปเหลี่ยม B แต่ถ้าผลรวมมีค่าเท่ากับ 360 องศา หมายความว่าจุดทดสอบซึ่งเป็นจุดยอดจุดหนึ่งบนรูป A อยู่ภายในรูปเหลี่ยม B



รูป 3.6 ทดสอบบริเวณภาพซึ่งถูกบรรจุภายในภาพอื่นโดยผลบวกของมุม



รูป 3.7 ทดสอบบริเวณของภาพซึ่งบรรจุในภาพอื่นโดยวิธีครึ่งพื้นระนาบ

2. ทดสอบโดยวิธีครึ่งพื้นระนาบ

จากการกำหนดหมายเลขตำแหน่งของจุดมุมยอดบนรูปเหลี่ยม เป็นลำดับตามทิศทางทวนเข็มนาฬิกา ทิศทางของเวกเตอร์ลากลูกศรจากหมายเลขน้อยมาสู่หมายเลขมากตามลำดับ และเมื่อถึงค่าสุดท้ายเวกเตอร์จะถูกลากหัวลูกศรกลับเข้าสู่หมายเลขตำแหน่งแรก ดังรูป ที่ 3.7 ถ้าเรามองตามทิศทางเวกเตอร์ของเส้นขอบภาพไปตามทิศทางหัวลูกศร จะพบว่าถ้าจุดใด ๆ อยู่ทางด้านซ้ายมือของทิศทางลูกศร จุดนั้นจะเป็นจุดซึ่งบรรจุอยู่ภายในรูปเหลี่ยม และถ้าจุดใดอยู่ทางด้านขวามือของทิศทางลูกศร จุดนั้นจะอยู่ภายนอกรูปเหลี่ยมนั้นเสมอ

วิธีการที่จะทดสอบว่าจุดนั้น ๆ อยู่ทางด้านซ้ายหรือด้านขวา ของเวกเตอร์บนเส้นขอบภาพ เราใช้ทฤษฎีของสมการพื้นระนาบมาพิจารณา โดยการหาค่าสมการจากการคูณทางเวกเตอร์ระหว่างเวกเตอร์บนเส้นขอบภาพตามทิศทางจากต้นลูกศรถึงปลายลูกศร (u) กับเวกเตอร์ที่ลากจากจุดต้นลูกศรนั้นมายังจุดที่ต้องการทดสอบ (v) ผลของการคูณทางเวกเตอร์ (cross product) u x v ถ้าได้ผลเป็นบวกแสดงว่าอยู่ทางด้านซ้าย และถ้าผลเป็นค่าลบแสดงว่าจุดทดสอบนั้นอยู่ทางด้านขวาของขอบภาพ

สรุปได้ว่าถ้าจุดทดสอบนั้นอยู่ทางด้านซ้ายของเวกเตอร์บนขอบภาพของรูปเหลี่ยม ในแต่ละด้าน ย่อมหมายความว่าจุดทดสอบนั้นอยู่ภายในบริเวณรูปเหลี่ยมนั้น และจุดทดสอบนั้นจะอยู่ภายนอกบริเวณรูปเหลี่ยม ถ้าอยู่ทางด้านขวามือของเวกเตอร์บนขอบภาพของรูปเหลี่ยมแม้เพียงด้านหนึ่งด้านใดก็ตาม

3.1.5 ความลึกของภาพ

การพิจารณาความลึกของภาพนั้น เราจะแปลงจากระบบแกนมิติที่มองด้วยสายตามาเป็นระบบแกนมิติซึ่งขึ้นอยู่กับพื้นระนาบบนจอภาพ

จากสมการของพื้นระนาบ ค่าสัมประสิทธิ์ในสมการสามารถหาได้จากพื้นผิวของรูปเหลี่ยมนั้น และถ้าจุดยอดของรูปเหลี่ยมมีลักษณะไม่เป็นเชิงเส้นร่วมกัน เราสามารถหาค่าเวกเตอร์ซึ่งความสัมพันธ์กับคุณลักษณะของพื้นระนาบนั้นได้ จากจุดยอดสามจุดของรูปเหลี่ยมนั้น การคูณกันทางเวกเตอร์ที่สร้างจากจุดยอดดังกล่าวทำให้สามารถกำหนดค่าสัมประสิทธิ์ของสมการพื้นระนาบ ดังนี้

$$Ax + By + Cz + D = 0 \qquad (3.8)$$

ถ้า $u = P1-P2 = (x_2,y_2,z_2)-(x_1,y_1,z_1)$

$v = P1-P3 = (x_3,y_3,z_3)-(x_1,y_1,z_1)$

ดังนั้นค่า A,B,C คำนวณได้จากสมการ $(A,B,C) = u \times v$ (3.9)

$$A = y_1(z_2-z_3)+y_2(z_3-z_1)+y_3(z_1-z_2),$$

$$B = z_1(x_2-x_3)+z_2(x_3-x_1)+z_3(x_1-x_2),$$

$$C = x_1(y_2-y_3)+x_2(y_3-y_1)+x_3(y_1-y_2),$$

และ $D = -Ax_1-By_1-Cz_1$ (3.10)

การพิจารณาลำดับก่อนหลังของภาพวัตถุจากตำแหน่งมองนั้น ใช้การคำนวณสองขั้นตอนย่อมพอเพียง ในการคำนวณแรกก็คือการคำนวณหาจุดตัดของรูปเหลี่ยม ซึ่งอาจมีได้หลายค่าแต่เราใช้เพียงค่าเดียวสำหรับทดสอบความลึก การคำนวณที่สองคือคำนวณหาค่า z_s ของแต่ละรูปเหลี่ยมที่จุดตัดกันนั้น ตัวอย่างเช่น

กำหนดให้ (x_1,y_1) เป็นจุดตัดบนระบบแกนมิติของจอภาพ

หาค่าสมการพื้นระนาบของรูปเหลี่ยมในระบบแกนมิติของจอภาพ

$a_1x_s + b_1y_s + c_1z_s + d_1 = 0$ สำหรับรูปเหลี่ยมที่ 1

$a_2x_s + b_2y_s + c_2z_s + d_2 = 0$ สำหรับรูปเหลี่ยมที่ 2

แทนค่า $x_s = x_1$ และ $y_s = y_1$ และหาค่า z_s จากสมการ

เปรียบเทียบค่า z_s ของทั้งสองรูป ซึ่งค่า z_s ที่มีค่าน้อยกว่าย่อมอยู่ใกล้กว่า

3.2 การจำลองภาพ[20]

การจำลองภาพด้วยการใช้ทฤษฎีการมองเห็นแสงบนวัตถุของ Lambert ในขั้นตอนแรกเราต้องพิจารณาถึงโครงสร้างใน 3 มิติของวัตถุนั้น แบ่งชิ้นส่วนของวัตถุเป็นส่วนเล็ก ๆ เรียงตามลำดับ และนำส่วนต่าง ๆ เหล่านั้นมาคำนวณหาความสว่างที่จะปรากฏในตำแหน่งทิศทางการมอง รวมทั้งพิจารณาสภาวะการมองเห็นได้เนื่องจากการซ้อนกัน หรือบังทิศทางของการมองเห็นบนชิ้นวัตถุนั้น ชิ้นส่วนต่าง ๆ ที่สามารถมองเห็นได้จะถูกนำมาคำนวณ

3.2.1 ความเข้มแสงบนภาพวัตถุ

เมื่อแสงตกกระทบพื้นผิว แสงนั้นอาจถูกดูดกลืน สะท้อน หรือผ่านไปได้ แสงส่วนที่ถูกดูดกลืนจะกลายเป็นความร้อน ส่วนที่เหลือเป็นส่วนที่ทำให้เรามองเห็นวัตถุได้ สิ่งที่ชี้บ่งว่าพลังงาน

จะถูกดูดกลืนหรือสะท้อนก็คือความยาวคลื่นของแสง เช่นบนวัตถุซึ่งสามารถดูดกลืนแสงได้ทุกความถี่เท่ากัน ดังนั้นเมื่อมีแสงสีขาวมาตกกระทบ เราจะเห็นวัตถุนั้นเป็นสีเทา ถ้าวัตถุนั้นถูกดูดกลืนแสงได้มาก วัตถุนั้นจะมองเห็นเป็นสีดำ แต่ถ้าวัตถุนั้นสามารถดูดกลืนได้เป็นบางช่วงของความยาวคลื่น วัตถุนั้นจะมองเห็นเป็นสีของความยาวคลื่นที่ไม่ถูกดูดกลืน

คุณสมบัติของการสะท้อนจากพื้นผิวของวัตถุขึ้นอยู่กับส่วนประกอบของ ทิศทาง คุณสมบัติทางเรขาคณิตของแหล่งกำเนิดแสง รูปทรงของวัตถุและคุณสมบัติบองพื้นผิววัตถุ แสงที่สะท้อนออกมาจากวัตถุสามารถแบ่งตามคุณสมบัติได้เป็น 2 ชนิดคือ

1. Diffusely Reflected คือแสงส่วนที่ผ่านพื้นผิวของวัตถุมีบางส่วนที่ถูกดูดกลืนและสะท้อนออกมา คุณสมบัติของแสงสะท้อนแบบนี้คือจะสะท้อนออกมาเท่ากันทุกทิศทาง ขึ้นอยู่กับตำแหน่งของผู้สังเกต

2. Specularly Reflected คือแสงที่สะท้อนจากพื้นผิวภายนอกของวัตถุตาม Lambert's cosine law กล่าวว่าแสงจากแหล่งกำเนิดที่เป็นจุด เมื่อส่องไปบนวัตถุที่มีการสะท้อนอย่างสมบูรณ์จะทำให้เกิดความเข้มตามสัดส่วนเป็นค่า cosine ของมุมระหว่างทิศทางของแสงกับเส้นตั้งฉากของพื้นผิว

$$I = I_1 * K_d * \cos\theta \qquad (3.11)$$

$$0 < K_d < 1 \ , \ 0 < \theta < \pi/2$$

เมื่อ I คือ ความเข้มของแสงสะท้อน

I_1 คือ ความเข้มของแสงที่ตกกระทบ

K_d คือ ค่าคงที่ Diffuse Reflected

ขึ้นอยู่กับชนิดของวัตถุและมีความสัมพันธ์กับความยาวคลื่นของแสง

θ คือ มุมของแสงตกกระทบกับเส้นตั้งฉากของพื้นผิว

ถ้า $\theta > \pi/2$ แสดงว่าแหล่งกำเนิดอยู่หลังวัตถุ

ภาพวัตถุที่ทำการ Shading ด้วยวิธีการกำหนดแหล่งกำเนิดแสงเป็นจุดนั้น ปรากฏเป็นภาพที่ไม่กลมกลืน เนื่องจากความเป็นจริงวัตถุได้รับแสงกระจายจากสิ่งแวดล้อม ดังนั้นต้องพิจารณา

แสงจากแสงสะท้อนของสิ่งแวดล้อม ดังความสัมพันธ์

$$I = I_a * K_a + I_1 * K_d * \cos \theta \qquad (3.12)$$

$$0 < K_a < 1$$

I_a คือ ความเข้มของแสงตกกระทบเนื่องจากการสะท้อนของสิ่งแวดล้อม

K_a คือ ค่าคงที่การสะท้อนของสิ่งแวดล้อม

จากวิธีการข้างบนในกรณีที่วัตถุมีรูปทรงเหมือนกันทุกประการ แต่อยู่ที่ระยะทางต่างกันจะทำให้ได้ความเข้มเท่ากัน ดังนั้นเมื่อวัตถุสองชิ้นอยู่เหลื่อมล้ำกัน เราจะไม่สามารถแยกความแตกต่างได้ ซึ่งในความเป็นจริงความเข้มของแสงจะเป็นปฏิภาคผกผันกับระยะทางกำลังสอง ถ้าในภาพใช้ระบบ Perspective Tranformation ระยะทางระหว่างจุดมองถึงวัตถุสามารถกำหนดให้เป็นค่าคงที่ d แต่อย่างไรก็ตาม เมื่อจุดมองอยู่ใกล้วัตถุ เทอม 1/d จะมีค่าเปลี่ยนแปลงอย่างรวดเร็ว ทำให้ได้ภาพที่มีความเข้มผิดจากความเป็นจริง เราสามารถแก้ไขปัญหานี้ได้โดยใช้กฎ Linear Attenuation ดังนี้

$$I = I_a * K_a + (I_1 * K_d * \cos \theta)/(d+k) \qquad (3.13)$$

เมื่อ k คือ ค่าคงที่ใด ๆ

ค่าความเข้มของแสงที่เกิดจาก Speccular Reflected ขึ้นอยู่กับแสงตกกระทบ ความยาวคลื่นของแสงตกกระทบและคุณสมบัติของเนื้อวัตถุ ในกรณีของวัตถุที่มีการสะท้อนโดยสมบูรณ์ เช่น กระจกเงา ค่ามุมของแสงตกกระทบจะเท่าของแสงสะท้อน ดังนั้นผู้สังเกตจะสามารถเห็นวัตถุได้เมื่ออยู่ในระนาบเดียวกับทิศทางสะท้อนของแสงเท่านั้น สำหรับในกรณีที่วัตถุที่มีการสะท้อนอย่างไม่สมบูรณ์ แสงที่สะท้อนมายังผู้สังเกตขึ้นอยู่กับค่าการกระจายในระนาบต่าง ๆ ของ Specular Reflected ค่าการกระจายนี้จะขึ้นอยู่กับลักษณะผิวของวัตถุด้วย คือวัตถุผิวเรียบการกระจายจะเป็นลำรังสีที่แคบ ๆ และในทางกลับกัน เมื่อผิวของวัตถุอยาบ การกระจายจะมีลักษณะที่แผ่ออกไปไม่จำกัดเป็นลำรังสี

ในทางธรรมชาติความเข้มที่จุดใด ๆ ของวัตถุยังขึ้นอยู่กับแสงสะท้อนจากสิ่งแวดล้อม และ Diffuse Reflection อีก ดังนั้นเมื่อนำผลทั้ง 3 อย่างมารวมกับค่า Specularly Reflected

$$I = I_a * K_a + I_1 * (K_d * \cos \theta + w(i,\lambda) * \cos^n \alpha) \; / \; (d+k) \qquad (3.14)$$

ค่า $w(i,\lambda)$ เป็นความสัมพันธ์ที่ยุ่งยากมาก ดังนั้นจากการทดลองสามารถแทนได้ค่าด้วยค่าคงที่ K_a

$$I = I_a*K_a + I_l*(K_d*\cos\theta + K_s*\cos^n\alpha) / (d+k) \qquad (3.15)$$

$$\cos\theta = (n.L) / |n||L| = n.L$$

เมื่อ n และ L เป็น ยูนิตเวกเตอร์ ของพื้นผิวและทิศทางของแสงตามลำดับ ในทำนองเดียวกัน

$$\cos\alpha = (R.S) / |R||S| = R.S$$

เมื่อ R และ S เป็น ยูนิตเวกเตอร์ ของรังสีสะท้อนและทิศทางของผู้สังเกตตามลำดับ ดังนั้นสมการของความเข้มสามารถเขียนสามารถเขียนได้อีกแบบหนึ่ง คือ

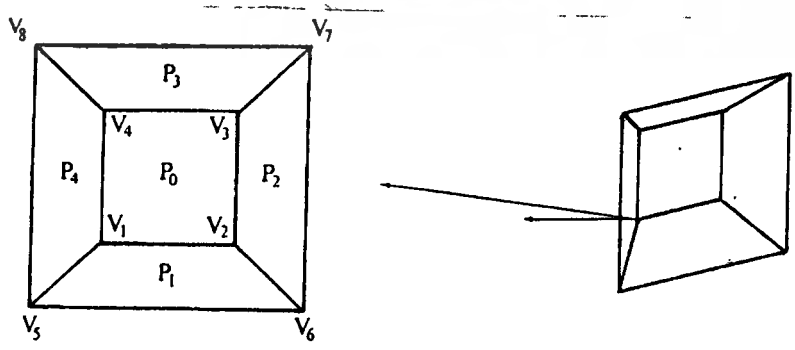
$$I = I_a*K_a + I_l*[\, K_d*(n.L) + K_s*(R.S)\,] / (d+k) \qquad (3.16)$$

3.2.2 การหาทิศทางตั้งฉากของยอดเหลี่ยม

ทิศทางของพื้นผิวหาได้จากลักษณะความโค้งของพื้นผิวในบริเวณนั้น และย่อมหาค่าได้หากมีข้อมูลในการวิเคราะห์เพียงพอ สำหรับรูปเหลี่ยมที่สามารถหาค่าสมการของพื้นระนาบที่ประกอบเป็นมุมเหลี่ยมนั้น ก็สามารถนำมาพิจารณาเป็นทิศทางของพื้นผิวตรงยอดเหลี่ยมนั้นได้ ซึ่งทิศทางประมาณของเวกเตอร์ตั้งฉากกับยอดมุมเหลี่ยมดังรูป 3.8 นั้น เขียนได้เป็น

$$nv_1 = (a0+a1+a4)i+(b0+b1+b4)j+(c0+c1+c4)k \qquad (3.17)$$

ซึ่ง a0,a1,a4,b0,b1,b4,c0,c1,c4 เป็นสัมประสิทธิ์ของสมการพื้นระนาบของพื้นเหลี่ยม P0,P1,P4 ซึ่งเป็นพื้นระนาบที่ล้อมรอบยอดเหลี่ยม V1 แต่ถ้าไม่อาจหาสมการของพื้นระนาบ เราหาค่าทิศทางตั้งฉากของยอดเหลี่ยมจากการเฉลี่ยผลคูณทางเวกเตอร์ของเส้นขอบ ซึ่งเส้นขอบนั้นเชื่อมต่อกับยอดเหลี่ยมนั้น ถ้าเราใช้ V1 เป็นตัวอย่าง เราจะเขียนสมการได้เป็น



รูป 3.8 การประมาณพื้นผิวตั้งฉากของยอดมุมเหลี่ยมของวัตถุรูปเหลี่ยม

$$nv_i = V1V2 \times V1V4 + V1V5 \times V1V2 + V1V4 \times V1V5 \qquad (3.18)$$

ซึ่งค่าประมาณที่ได้นี้ขึ้นอยู่กับจำนวนและพื้นที่ของแต่ละรูปเหลี่ยม หรือจำนวนและความยาวของแต่ละเส้นขอบ ผลของความเข้มแสงที่ได้จากการประมาณจึงแตกต่างกันขึ้นอยู่กับวิธีการที่นำไปใช้

3.2.3 การคำนวณทิศทางแสงสะท้อน[20]

การคำนวณนั้นอยู่ภายใต้เงื่อนไขว่า เวกเตอร์ทั้งสามของ ทิศทางตั้งฉากพื้นผิว ทิศทางของแสงตกกระทบ และทิศทางของแสงสะท้อน ต้องอยู่บนพื้นระนาบเดียวกัน ซึ่งค่าผลของการคูณเวกเตอร์ dot product ของยูนิตทิศทางตั้งฉากกับยูนิตของแสงและทิศทางแสงสะท้อนจะต้องเป็นค่าที่แสดงว่า มุมของแสงตกกระทบต้องเท่ากับมุมของแสงสะท้อน เราจึงได้เงื่อนไข

$$n \times L = R \times n \qquad (3.19)$$

หรือ

$$(n_yL_z-n_zL_y)i+(n_zL_x-L_zn_x)j+(n_xL_y-L_xn_y)k =$$

$$(n_zR_y-n_yR_z)i+(n_xR_z-n_zR_x)j+(n_yR_x-n_xR_y)k \qquad (3.20)$$

ซึ่งจะได้ว่า

$$\begin{aligned} -n_zR_y+n_yR_z &= n_zL_y-n_yL_z \\ n_zR_x \quad -n_xR_z &= n_xL_z-n_zL_x \\ -n_yR_x+n_xR_y &= n_yL_x-n_xL_y \end{aligned} \qquad (3.21)$$

จากสมการที่ได้ยังไม่พอเพียง เราจึงใช้เงื่อนไขจากค่ามุมสะท้อนและมุมตกกระทบของแสงย่อมเท่ากัน

$$n \cdot L = n \cdot R \qquad (3.22)$$

หรือ

$$n_xR_x+n_yR_y+n_zR_z = n_xL_x+n_yL_y+n_zL_z \qquad (3.23)$$

ถ้าเรานำมาเขียนแมททริกซ์ในแบบสี่เงื่อนไข และค่าที่ไม่รู้สามค่า คือ R_x, R_y, R_z คือ

$$\begin{vmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \\ n_x & n_y & n_z \end{vmatrix} \begin{vmatrix} R_x \\ R_y \\ R_z \end{vmatrix} = \begin{vmatrix} n_zL_y-n_yL_z \\ n_xL_z-n_zL_x \\ n_yL_x-n_xL_y \\ n_xL_x-n_yL_y+n_zL_z \end{vmatrix}$$

หรือ

$$[N][R] = [B]$$

เนื่องจาก [N] ไม่เป็นแมททริกซ์แบบจตุรัส เราจึงใช้วิธีการในการหาคำตอบดังนี้

$$[R] = [\,[N]^T[N]\,]^{-1}\,[N]^T[B]$$

จะได้ผลลัพท์ดังนี้

$$\begin{aligned} R_x &= -L_x + [2n_x(n_xL_x+n_yL_y+n_zL_z)]/(n_x^2+n_y^2+n_z^2) \\ R_y &= -L_y + [2n_y(n_xL_x+n_yL_y+n_zL_z)]/(n_x^2+n_y^2+n_z^2) \\ R_z &= -L_z + [2n_z(n_xL_x+n_yL_y+n_zL_z)]/(n_x^2+n_y^2+n_z^2) \end{aligned} \qquad (3.24)$$

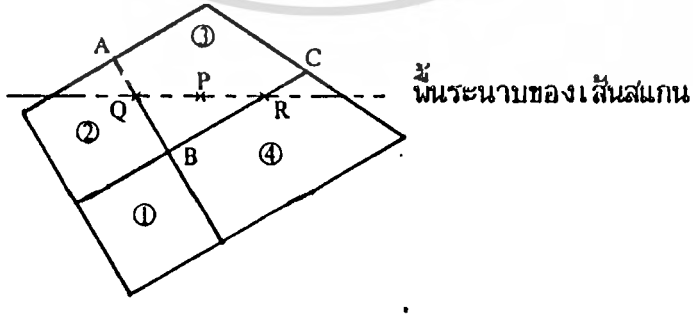
3.2.4 การเฉลี่ยความเข้มแสงบนภาพวัตถุ[12]

ภาพจำลองสามารถสร้างได้จากค่าความเข้มของแสงที่ตำแหน่งต่าง ๆ ของวัตถุ โดยแบ่งพื้นที่ผิวของวัตถุเป็นรูปเหลี่ยมขนาดเล็กมาประกอบกัน ความเข้มจากการคำนวณจากรูปเหลี่ยมดังกล่าวจะไม่ต่อเนื่องกัน ทำให้ภาพที่ปรากฏไม่เหมือนวัตถุจริง เราต้องเฉลี่ยตามความเข้มของแสงบนรูปเหลี่ยมขนาดเล็กอีกขั้นตอนหนึ่ง ดังรูป 3.9 เป็นการเฉลี่ยแสง โดยวิธี Gouraud Shading จากรูป ในการที่จะหาความเข้มที่ P จำเป็นจะต้องหาความเข้มที่จุด Q และ R ก่อนดังต่อไปนี้

$$I_q = uI_a + (1-u)I_b \qquad (3.25)$$

เมื่อ $0 < u < 1$

$$u = QB \,/\, AB$$



รูป 3.9 การเฉลี่ยความเข้มโดยวิธีของ Gouraud

$$I_r = wI_b + (1+w)I_c \qquad (3.26)$$

$$0 < w < 1$$

เมื่อ $w = RC / BC$

ในที่สุดจะได้

$$I_p = tI_q + (1-t)I_r \qquad (3.27)$$

$$0 < t < 1$$

เมื่อ $t = PR / QR$

จากวิธีการดังกล่าว เราสามารถคำนวณหาความเข้มทุกจุดบนภาพวัตถุได้ จากการใช้วิธีการเฉลี่ยค่าความเข้มแสงแบบ linearly interpolation ตามแนวเส้นสแกน ความเข้มที่ได้ยังไม่สอดคล้องกับลักษณะความโค้งของพื้นผิววัตถุดีพอ เนื่องจากการเฉลี่ยของความเข้มแสงที่แท้จริงนั้น ไม่ได้เป็นเชิงเส้นตามแนวเส้นสะแกน แต่สัมพันธ์โดยตรงกับเวกเตอร์ตั้งฉากกับพื้นผิวความโค้งของวัตถุนั้น จากวิธีการของ Phong shading ได้แก้ไขการประมาณค่าเฉลี่ยความเข้มแสงได้ดีขึ้น โดยในขั้นตอนแรกเริ่มต้นด้วยการประมาณค่าทิศทางตั้งฉากที่จุดยอดของแต่ละมุมเหลี่ยมเสียก่อน จากนั้นจึงทำการเฉลี่ยค่าเวกเตอร์ของทิศทางตั้งฉากในลักษณะ linearly interpolation ทุกจุดบนภาพวัตถุ เช่นหาค่า Q จากระหว่าง A และ B หาค่า R ระหว่าง B และ C ในที่สุดก็จะได้ค่าทิศทางตั้งฉากที่จุด P จากค่าระหว่าง Q และ R ดังนี้

$$n_q = un_a + (1-u)n_b \qquad 0 \le u \le 1 \qquad (3.28)$$

$$n_r = wn_a + (1-w)n_b \qquad 0 \le w \le 1 \qquad (3.29)$$

$$n_p = tn_a + (1-t)n_b \qquad 0 \le t \le 1 \qquad (3.30)$$

ซึ่งค่า $u = BQ/AB, w=CR/BC, t=RP/QR$, จากนั้นเราก็สามารถคำนวณหาค่าเวกเตอร์ตั้งฉากพื้นผิวค่าอื่นได้ต่อไปโดยวิธีที่บรรยายมานี้ทำให้สามารถหาค่าเฉลี่ยได้ทุกตำแหน่งบนพื้นผิวของวัตถุ โดยสัมพันธ์กับความโค้งของพื้นผิววัตถุ จากนั้นจึงนำค่าเวกเตอร์ที่ได้มาคำนวณหาค่าระดับความเข้มแสงบนแต่ละจุดนั้น ค่าความเข้มแสงบนวัตถุซึ่งประมาณด้วยวิธีการนี้จะมีความสิ้นเปลืองเวลาคำนวณมาก แต่ภาพของวัตถุที่จำลองมานั้น จะมีความกลมกลืนสอดคล้องกับลักษณะความโค้งของพื้นผิววัตถุดีขึ้น และมีความเหมือนจริงมากยิ่งขึ้น

บทที่ 4

การหารูปทรงจากระดับความเข้มแสง

ในบทนี้บรรยายถึงความสัมพันธ์ระหว่างระดับความเข้มของแสงที่ปรากฏภาพ และรูปร่างของวัตถุนั้น โดยปกติเรายังใช้ประโยชน์จากค่าของความเข้มแสงไม่มากนัก ส่วนใหญ่เพียงแต่เป็นการเปรียบเทียบค่าของความเข้มแสงกับระดับเฉลี่ยของความเข้มแสงในระดับเดียวกัน ซึ่งค่าความเข้มของแสงนั้นสามารถให้ข้อมูลอื่น ๆ โดยเฉพาะในส่วนที่เกี่ยวกับระดับของพื้นผิวภาพ

การคำนวณหารูปทรงของวัตถุ จากระดับของแสงในภาพ คือการหาระดับความสูงของวัตถุจากระดับของพื้นระนาบที่อยู่เบื้องหลังของวัตถุนั้น หรือกล่าวได้ว่าเป็นการหาระดับของพื้นผิวที่ต้องการจากพื้นผิวอื่น ๆ ข้อมูลที่ปรากฏในแต่ละจุดบนภาพนำมาคำนวณหาระดับพื้นผิวของวัตถุที่ตำแหน่งเดียวกับจุดบนภาพได้ โดยการตั้งสมมุติฐานว่าพื้นผิวมีระนาบต่อเนื่องกันไป ซึ่งแตกต่างจากวิธีการของ machine-vision ซึ่งวิเคราะห์ความลึกของพื้นผิวจากลักษณะพื้นผิวที่ไม่ต่อเนื่อง และข้อมูลจากลักษณะที่เป็นขอบเท่านั้น

4.1 ความรู้พื้นฐาน[3]

ระดับของแสงที่นำมาพิจารณาพื้นผิวของวัตถุนั้น ถูกจัดอยู่ในรูปแบบของสมการอนุพันธ์อันดับหนึ่งไม่เป็นเชิงเส้น มีจำนวนไม่ทราบค่า 2 จำนวนซึ่งสามารถหาค่าได้โดยใช้วิธีการของ strip expansion

reflectance map แสดงถึงความเกี่ยวเนื่องระหว่างระดับความลึกของพื้นผิวและระดับความเข้มของแสง ใช้กำหนดเป็นค่า gradient space ถ้าเราสมมุติให้ภาพที่เกิดขึ้นเป็น orthographic projection และทิศทางการมองภาพขนานกับแกน z รูปทรงของวัตถุแสดงระดับความสูง z ซึ่งวางอยู่ในระนาบ x-y ถ้า p และ q เป็น partial derivative ของ z ดังนี้

$$p = \partial z/\partial x \quad \text{และ} \quad q = \partial z/\partial y \tag{4.1}$$

และระนาบ p-q ในที่นี้ถูกอ้างอิงเป็น gradient space เนื่องจากทุก ๆ ค่าบนระนาบ p-q สัมพันธ์กับความชันของผิว ระยะห่างจากจุดกำเนิดของ gradient space มีค่าเท่ากับความชันของผิว และขณะเดียวกันทิศทางของมันเป็นทิศทางของ steepest ascent

จากการกำหนดแสงที่คงที่ คุณสมบัติการสะท้อนของผิวและคุณลักษณะทางเรขาคณิต

เป็นแนวทางให้เห็นชัดเจนถึงความสัมพันธ์โดยตรงของระดับความเข้มในภาพไปสู่ระดับความสูงของพื้นผิววัตถุ ถ้าเราหาลักษณะการสะท้อนของผิววัตถุ ให้เป็นฟังก์ชั่นของทิศทางของแสง ทิศทางการมองและมุมระหว่างทิศทางทั้งสองได้ ดังนั้นเราสามารถจะวิเคราะห์ความสัมพันธ์ทางทฤษฎีได้ และคำนวณหา reflectance map ได้

reflectance map ใช้อธิบายคุณลักษณะของระดับความเข้มของแสงในภาพ ถ้าค่า E(x,y) เป็นค่าความเข้มของแสงที่จุดใด ๆ ย่อมเกี่ยวเนื่องไปถึงระดับความลาดชันของพื้นผิวที่จุดนั้นได้ ระดับความเข้มของแสงซึ่งวัดได้จากภาพนำไปสู่การคำนวณหาระดับของพื้นผิวบนวัตถุภายใต้ข้อบังคับซึ่งเรียกว่าสมการ image-irradiance

$$R(p,q) = E(x,y) \tag{4.2}$$

p,q เป็นค่าความลาดชันที่ควรจะเป็นไปได้ และ E(x,y) คือค่าความเข้มของแสงที่จุด (x,y) โดยปกติค่าซึ่งวัดได้เพียงค่าเดียวไม่อาจกำหนดค่า p และ q ทั้งสองค่าได้ เพียงแต่กำหนดค่าความสัมพันธ์ระหว่างตัวแปร p และ q เท่านั้น เรายังต้องการข้อจำกัดอื่นเพิ่มเติมเพื่อให้ได้ค่า p และ q ที่ถูกต้อง ดังนั้นจำเป็นต้องตั้งสมมุติฐานบางอย่างจากลักษณะของพื้นผิว

4.1.1 ค่าความลาดชันในเชิงคณิตศาสตร์[3]

จากสมการ patial differential ที่ได้กล่าวไว้ นำมาแทนที่ด้วยกลุ่มสมการอนุพันธ์ทั่วไปของ x,y,z,p และ q ถ้า s เป็นระยะทางเคลื่อนที่ตามแนว strip

$$dx/ds = R_p \text{ และ } dy/ds = R_q ,$$

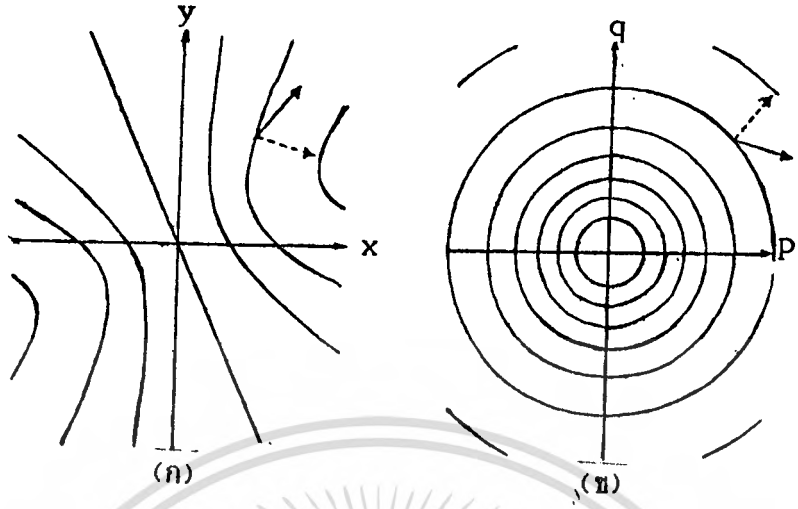
$$dz/ds = pR_p + qR_q ,$$

$$dp/ds = E_x \text{ และ } dq/ds = E_y \tag{4.3}$$

ซึ่ง R_p และ R_q เป็น patial derivative ของ p และ q ตามลำดับ ในขณะที่ E_x และ E_y เป็น patial derivative ของ x,y ตามลำดับ

จะเห็นได้ว่าความลาดชัน (p,q) บนพื้นผิวของวัตถุจะสัมพันธ์กับตำแหน่ง (x,y) บนภาพ และผลลัพธ์ต่อเนื่องจากจุดนี้หาได้โดยเปลี่ยนค่า (x,y) ไปเล็กน้อยในทิศทางที่พิจารณาได้จากค่าความลาดชันของ reflectance map ในทำนองเดียวกันทิศทางที่เปลี่ยนค่าไปบน reflectance map พิจารณาได้จากค่าความชันของระดับความเข้มแสงบนภาพ ดังรูป 4.1

สิ่งที่สำคัญกล่าวคือค่าต่าง ๆ ที่นำมาคำนวณนั้นถ้ามีความผิดพลาดใดเกิดขึ้น จะทำให้



รูป 4.1 ลักษณะของวิธีการ strip-expansion

รูป (ก) เป็นกลุ่มเส้นทางของแสง รูป (ข) เป็น reflectance map ผลของ characteristic strip ทิศทางต่างไปจากทิศทางที่ถูกต้องและเพิ่มมากขึ้นตามลำดับการคำนวณ วิธีการที่ใช้งานของ characteristic strip โดยลดผลของความผิดพลาดนี้คือการตั้งสมมุติฐานให้พื้นที่ผิวมีลักษณะต่อเนื่องระหว่าง strip

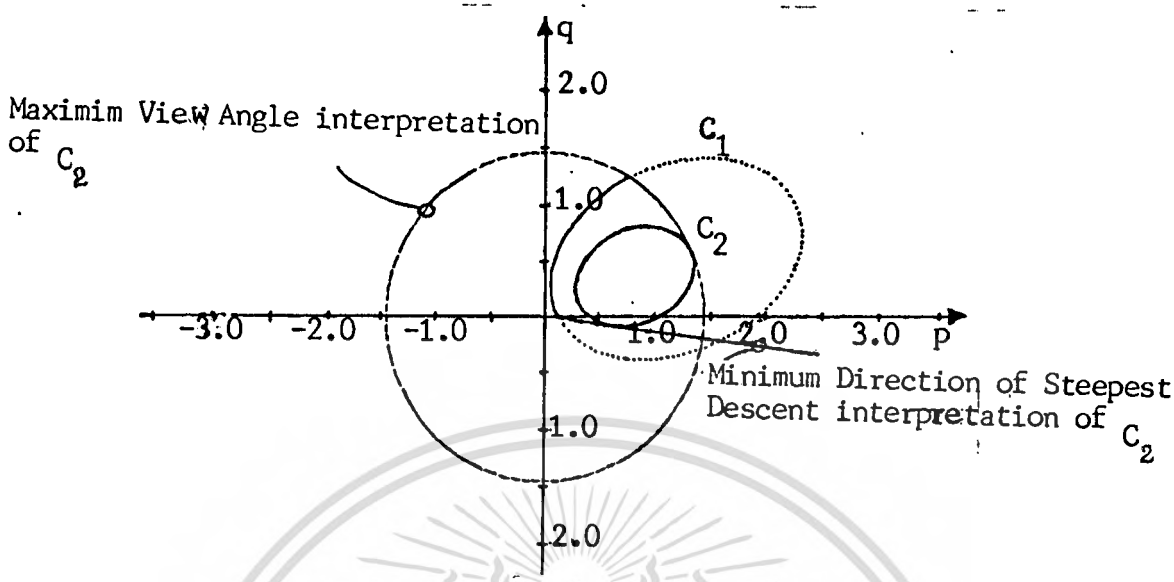
จุดเริ่มต้นของ strip อยู่ใกล้กับตำแหน่งซึ่งเรียกว่า singular point และเริ่มเคลื่อนการคำนวณห่างจากจุดนี้ไป แต่ขั้นตอนสำคัญของ characteristic strip นั้นไม่ได้อยู่ที่ขั้นตอนสุดท้าย ซึ่งปัญหานี้สามารถแก้ไขได้จากข้อมูลบริเวณที่เรียกว่า occluding boundary

การหาลักษณะ 3 มิติจากภาพ 2 มิติ แบ่งออกเป็น 2 วิธีคือ ให้ภาพหลายภาพจากตำแหน่งเดียวกันแต่ทิศทางแสงต่างกัน และใช้ภาพเพียงภาพเดียวร่วมกับการตั้งสมมุติฐานเกี่ยวกับธรรมชาติของพื้นผิว

4.1.2 วิธีการคำนวณหารูปทรง.[3]

วิธีการแยกคำตอบที่ถูกต้อง จากค่าที่คำนวณได้ ซึ่งจากวิธีการของ Woodham [Woodham, 1977][3] ได้หาข้อบังคับขึ้น 2 กรณี เช่น จุด P1 และ P2 ที่ตำแหน่ง (x1,y1) และ (x2,y2) ซึ่งอยู่ใกล้กันบนผิวของวัตถุที่เป็นพื้นผิวต่อเนื่องกัน ถ้ากำหนดว่ามุมของการมองระหว่างทิศทางตั้งฉากกับพื้นผิวกับทิศทางการมองมีค่าเพิ่มขึ้น และทิศทางของ steepest ascent มีค่าลดลง ขณะที่เคลื่อนที่จาก P1 ไปยัง P2 กำหนดให้ C1 และ C2 เป็นเส้นแสดงค่าที่เป็นไปได้ของระดับพื้นผิวที่ตำ

แหน่ง (x1,y1) และ (x2,y2) ใน gradient space โดยสัมพันธ์กับ R(p,q) = E(x1,y1) และ



รูป 4.2 ค่าความลาดชันที่เป็นไปได้ของจุด P1 และ P2

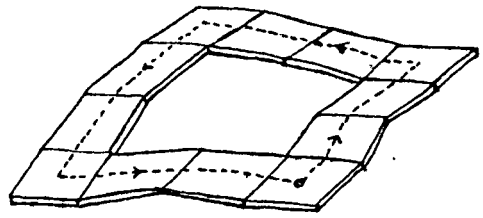
R(p,q) = E(x2,y2) ค่า R(p,q) คือค่า Reflectance map และ E(x1,y1),E(x2,y2) เป็นระดับความเข้มที่ตำแหน่ง P1 และ P2 ดังรูป 4.2 เห็นได้ว่าทุก ๆ ค่า (p,q) ของ P1 อยู่บนเส้น C1 เฉพาะส่วนที่อยู่ในวงกลมที่แสดงค่ามุมการมองสูงสุดของ C2 นอกจากนั้นค่า (p,q) ของ P1 เป็นค่าอยู่บนเส้น C1 ซึ่งเหนือเส้นแสดงค่าต่ำสุดของ steepest decent ที่ P2

เพื่อให้เป็นไปตามสมมุติฐานที่กำหนด ดังนั้นค่า patial derivative อันดับสองของระดับความสูงบนพื้นผิวมีค่าทุกค่าและต่อเนื่องกันค่า partial derivative ของ z โดยค่า x และ y เขียนได้เป็น

$$\partial p/\partial y = \partial^2 z/\partial y \partial x = \partial^2 z/\partial x \partial y = \partial q/\partial x \qquad (4.4)$$

ถ้าพื้นที่ติดต่อกันในบริเวณ B ได้ว่า

$$\iint (\partial p/\partial y - \partial q/\partial x) = 0 \qquad (4.5)$$



รูป 4.3 การ integral วงปิดของ (p,q)

ค่า integrand เป็นศูนย์ทุกค่า จาก Green's theorem [Hildebrand, 1965][3] ค่า integral ของ p,q ตามเส้นโค้งปิด (ในขอบเขต ∂B) จะมีค่าเป็นศูนย์

$$\int\{pdx + qdy\} = 0 \qquad (4.6)$$

โดยทั่วไป line integral มีค่าแตกต่างกันระหว่างปลายทั้งสองแต่เราได้กำหนดให้อยู่ในพื้นที่ผิวบริเวณเดียวกัน ดังรูป 4.3 ถ้าเราใช้วิธีการทางตัวเลข ซึ่งข้อมูลทุกจุดบนพื้นผิวไม่สมบูรณ์พอ ค่าผลของการ integral ก็ไม่เป็นศูนย์ ดังนั้นจึงต้องพยายามที่จะหาผลของ loop integral โดยที่มีค่าผิดพลาดน้อยที่สุดและเป็นไปตามสมการของ image-irradiance ให้ใกล้เคียงที่สุด จากแนวคิดนี้ Start [Start,1979][3] และ Brooks [Brooks,1979][3] ทั้งคู่ได้ติดตามวิธีการนี้ต่อไป โดย Brooks แสดงให้เห็นว่าเขาสามารถหาค่า loop integral เล็ก ๆ บนระนาบของภาพ เขากำหนดค่าสภาพผิวตามควรจะเป็นในทุกจุด แล้วนำมาแยกหาความเป็นจริงโดยเปรียบเทียบกับบรรทัดฐานของ loop integral โดยใช้วิธี relaxation

Start ลดความผิดพลาดในการ integral บนพื้นที่วงปิดเล็ก ๆ และค่าผิดพลาดในสมการ image-irradance เขาได้ปรับปรุงวิธีการหาที่เหมาะสม แต่มีข้อจำกัดว่าเป็นพื้นที่ของส่วนโค้งที่ปิดเท่านั้น

4.1.3 โฟโตเมตริก สเตอริโอ[3]

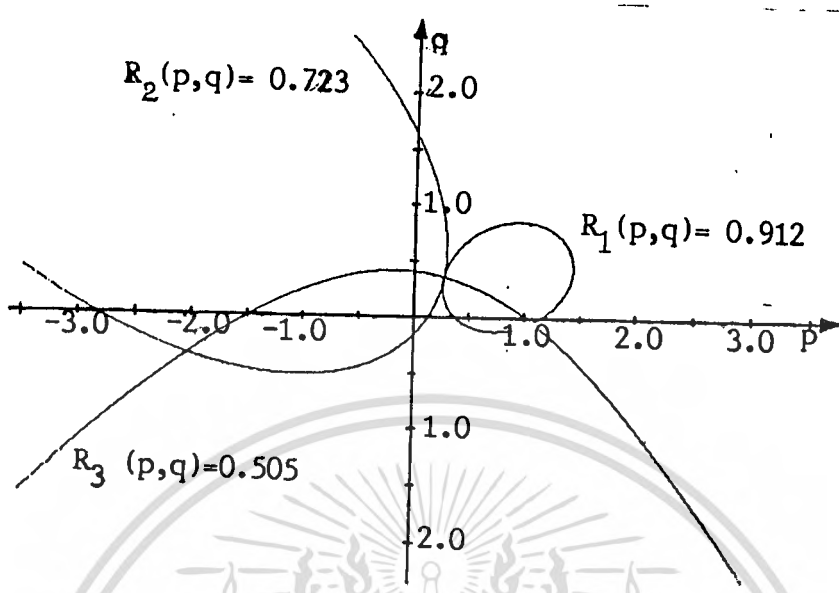
เราได้เพิ่มข้อกำหนดขึ้น โดยการสร้างภาพขึ้นหลายภาพและแต่ละภาพสร้างจากตำแหน่งเดียวกันแต่เปลี่ยนทิศทางของแสงเรียกว่า photometric stereo ทำให้สามารถพิจารณาหาค่าพื้นที่ของผิวได้ และไม่ต้องกำหนดสมมุติฐานให้เป็นผิวชนิดต่อเนื่องกัน

เนื่องจากภาพถูกมองจากตำแหน่งเดียวกัน ดังนั้นค่า particular point บนวัตถุปรากฏที่ตำแหน่งเดียวกันทุกภาพ จึงไม่เกิดปัญหาจากลักษณะพื้นผิวเหมือนภาพที่มองจากหลายตำแหน่ง อย่างไรก็ดี reflectance map ต่างกันไปในแต่ละภาพ เนื่องจากลักษณะของแสงแตกต่างกันไป เมื่อกำหนดจุดใดจุดหนึ่งบนภาพ เราจะได้ค่าของแสงที่ตำแหน่งนั้น (x0,y0) ค่าความสว่างนี้สามารถวัดได้ เราจะได้สมการ 2 สมการ

$$R_1(p,q) = E_1(x0,y0) \ , \ R_2(p,q) = E_2(x0,y0) \qquad (4.7)$$

R_1 และ R_2 เป็น reflectance map ที่เกิดจากแสงต่างตำแหน่งกัน ขณะที่ E_1

และ E_2 เป็นค่าความสว่างของแสงจากตำแหน่งผู้สังเกต จุดตัดระหว่างเส้นทางเดินของแต่ละ



รูป 4.4 พิจารณาลักษณะพื้นผิวของจุดซึ่งกำหนดบนภาพ โดยวิธี photometric stereo ค่าของ gradient space นำไปสู่การค้นหาลักษณะที่ถูกต้องของพื้นผิว ซึ่งเป็นหลักการสำคัญของ photometric stereo จากสมการ nonlinear 2 สมการ เพื่อหาคำตอบที่ชี้เฉพาะลงไปได้ ดังรูป 4.4

ค่า p,q หาได้ทุก ๆ จุดบนภาพนำไปสู่การหาค่าของสภาพพื้นผิวในบริเวณนั้น ขณะเดียวกันสามารถแสดงลักษณะรูปทรงของวัตถุและตำแหน่งของวัตถุใน 3 มิติได้

4.2 แนวทางการวิเคราะห์[3]

เส้นขอบของเงาซึ่งปรากฏบนพื้นของภาพเรียกว่า silhouette ส่วนต่าง ๆ ของ silhouette สัมพันธ์กับส่วนของผิวที่เป็นขอบหรืออาจเป็นส่วนโค้งของพื้นผิว ส่วนโค้งของพื้นผิวที่ smooth นี้เราเรียกว่า occluding boundary ซึ่งเป็นส่วนที่บ่งบอกลักษณะข้อมูลของพื้นผิวได้อย่างดี

อย่างไรก็ดีเราสามารถหาลักษณะของพื้นผิวได้ เพียงพิจารณาจากระดับความเข้มของแสงบนพื้นผิวนั้น โดยใช้ occluding boundary เป็นตัวกำหนดเงื่อนไขจากค่า partial derivative ของระดับความสูงที่บริเวณขอบมีค่าเป็นอนันต์

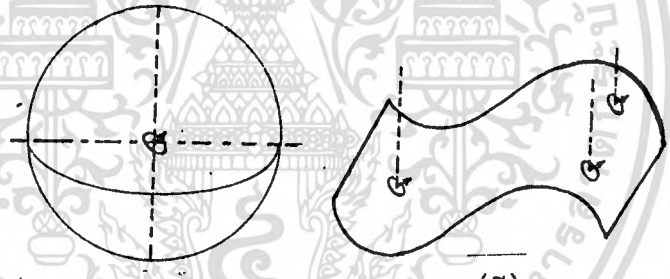
4.2.1 รูปทรงกลมเกาส์เซียนและฟังก์ชั่น reflectance

เรากำหนดให้พื้นที่ในสภาวะต่าง ๆ คือตำแหน่งของแต่ละจุดบนรูปทรงกลมมีรัศมีหนึ่ง

หน่วย ถ้ากำหนดตำแหน่งการมองอยู่ทางขั้วเหนือของรูปทรงกลมตามแกน z ซึ่งแกน z นี้คือเส้นตรงที่ลากผ่านจุดศูนย์กลางสู่ขั้วด้านเหนือสมมุติให้บริเวณพื้นผิวของวัตถุถูกนำไปวางไว้ที่ศูนย์กลางของทรงกลมโดยไม่มีการเปลี่ยนทิศทางใด ๆ พื้นผิวบริเวณนั้นจะหันด้านหน้าสู่จุดใดจุดหนึ่งบนผิวทรงกลม นั่นคือทิศทางตั้งฉากกับพื้นผิวลากผ่านจุด ๆ นั้น พื้นผิวที่อยู่ในแนวนอนแทนได้ด้วยจุดตรงขั้วด้านเหนือของทรงกลมหรือทิศทางตรงไปสู่ตำแหน่งการมอง ส่วนพื้นผิวที่ขนานกับทิศทางการมองแทนด้วยจุดบนเส้นศูนย์สูตรและพื้นผิวที่หันด้านกลับออกจากทิศทางการมองแทนด้วยส่วนล่างของรูปทรงกลม

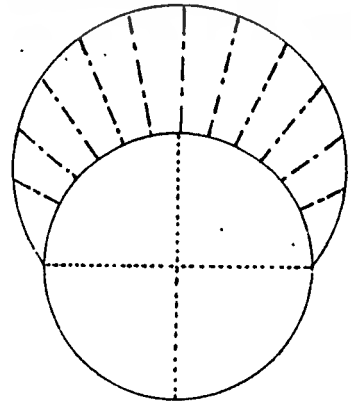
4.2.2 ความสัมพันธ์ระหว่างความสว่างบนพื้นผิวและทรงกลมเกาส์เซียน

ถ้าเราสมมุติให้การโปรเจคของ 3 มิติเป็นแบบขนาน ดังนั้นมุมระหว่างเส้นของการมองและทิศทางของผิวจะเป็นอิสระจากตำแหน่งที่ปรากฏบนวัตถุ จากลักษณะการสะท้อนแสงมันจะเกี่ยวข้องเฉพาะความเอียงของพื้นผิวบริเวณนั้นและแทนได้โดยจุดบนทรงกลมเกาส์เซียน ลักษณะพื้นผิวที่มี



รูป 4.5 บริเวณพื้นผิวเล็ก ๆ ของวัตถุ

(ก) ความลาดชันเดียวกันสัมพันธ์กันจุดเดียวกันบนทรงกลมเกาส์เซียน (ข) พื้นผิวบนวัตถุ



รูป 4.6 แสดงค่าความสว่างที่ปรากฏบนแต่ละจุดของทรงกลม

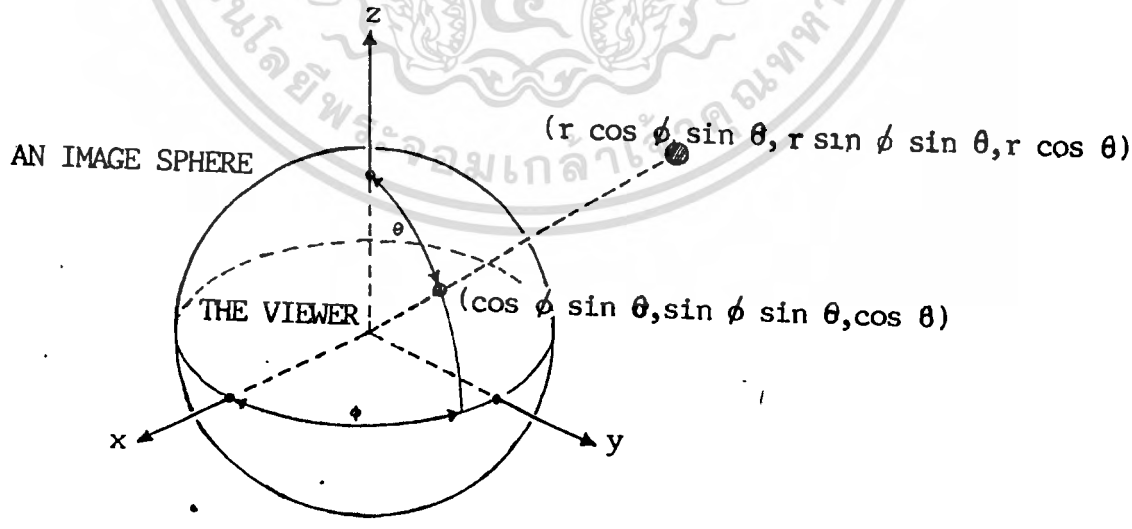
ความลาดเอียงคล้ายกัน จะสัมพันธ์กับจุดเพียงจุดเดียวบนทรงกลม และไม่เกี่ยวข้องกับตำแหน่งใน space ดังรูป 4.5

ความเข้มของแสงบนพื้นผิวขึ้นอยู่กับชนิดของผิววัตถุ และการสะท้อนตามหลักเรขาคณิต มุมระหว่างทิศทางการมองและทิศทางของพื้นผิวรวมทั้งแสงตกกระทบ เนื่องจากแต่ละจุดบนทรงกลมเกาส์เซียน สามารถอธิบายเป็นค่าเฉพาะทางเรขาคณิตเพียงค่าใดค่าหนึ่งซึ่งสัมพันธ์กัน เราจึงได้ค่าระดับความเข้มเพียงค่าเดียวบนพื้นผิวของมัน ตัวอย่างเช่น บริเวณพื้นผิวที่ตั้งฉากกับทิศทางการมองจะได้ค่าระดับความเข้มค่าหนึ่งเสมอในสภาวะนี้เรากำหนดค่าที่ขั้วด้านเหนือ และด้วยวิธีการเดียวกันเราจึงกำหนดค่าทุกค่าบนทรงกลมเกาส์เซียนได้ แสดงในรูป 4.6

4.3 การโปรเจคแบบทรงกลม

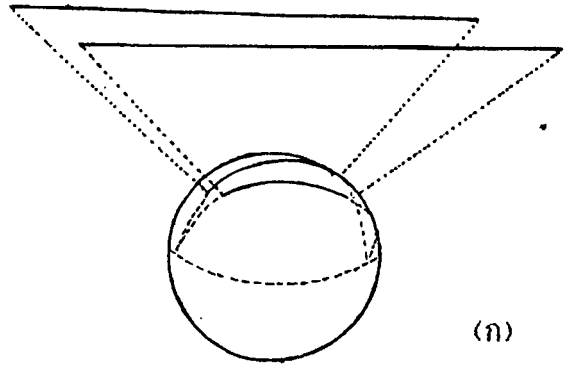
เราใช้การโปรเจคแบบทรงกลมเป็นหลักในการพิจารณาระบบในการมองภาพ โดยกำหนดให้ผู้สังเกตอยู่ที่ศูนย์กลางของทรงกลม ลายเส้นต่าง ๆ ของภาพถูกโปรเจคลงบนผิวของทรงกลมผ่านศูนย์กลางเดียวกัน ทิศทางการมองคือรัศมีของทรงกลม ดังรูป 4.7

ถ้าให้จุดอยู่ที่ตำแหน่ง $(r\cos(\phi)\sin(\theta), r\sin(\phi)\sin(\theta), r\cos(\theta))$ ซึ่ง r คือระยะทางระหว่างจุดและผู้สังเกต มุม θ และ ϕ เป็นมุมเปรียบเทียบกับยอดของทรงกลมและแนวแกนศูนย์สูตรตามลำดับ เพื่อให้ง่ายต่อการเข้าใจเราถือว่าภาพอยู่ด้านหน้าของผู้สังเกตและจุดนั้นถูกโปร

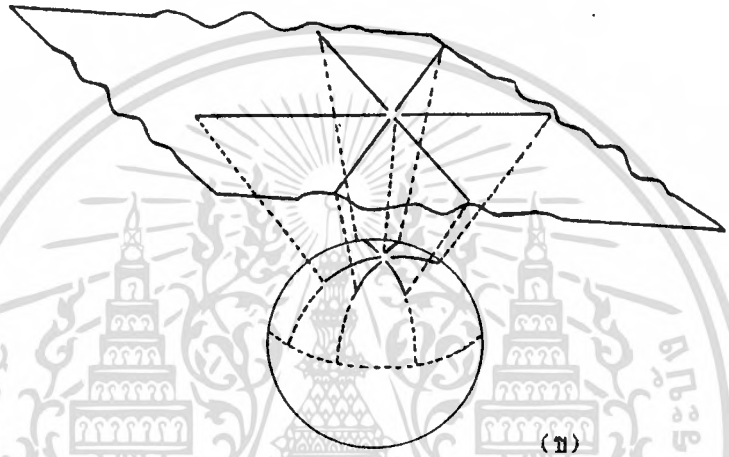


รูป 4.7 แสดง image sphere

ซึ่งเส้นทางเดินต่าง ๆ เกิดจากโปรเจคภาพผ่านศูนย์กลางลงบนผิวทรงกลม



(ก)



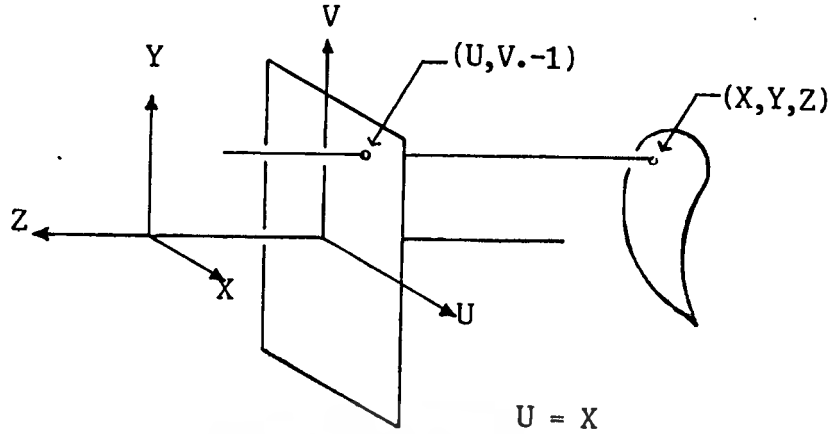
(ข)

รูป 4.8 คุณลักษณะของการโปรเจคแบบ spherical perspective

เจคที่ (cos(ø) sin(θ),sin(ø) sin(θ),cos(θ)) ดังนั้นเส้นตรงที่มีความยาวอนันต์ย่อมถูกโปรเจคลงเป็นเส้นวงกลมขนาดใหญ่ ดังรูป 4.8(ก) ซึ่งเส้นวงกลมนั้นผ่านจุดรวมกันของเส้นสองจุด เส้นตรงสองเส้นที่ยาวอนันต์เกิดเส้นวงกลมสองวงตัดกันที่จุดรวมของเส้น และระนาบที่มีขนาดอนันต์สามารถโปรเจคคลุมซีกด้านบนของทรงกลม ทิศทางตั้งฉากของระนาบจะผ่านศูนย์กลางของซีกโลกดังรูป 4.8(ข)

4.3.1 เปรียบเทียบกับการโปรเจคชนิดอื่น

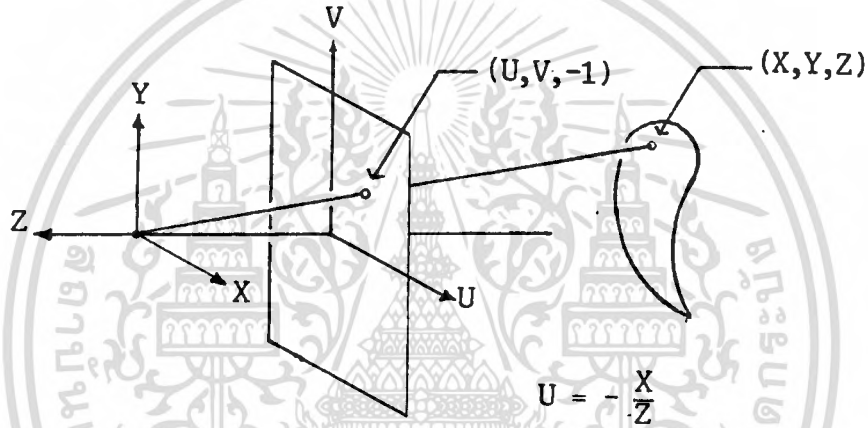
การโปรเจคที่นิยมใช้นั้นคือ orthographic และ perspective ซึ่งเราจะถือว่าเป็น perspective ของระนาบ ในกรณีของ orthrographic นั้น ถ้าขนาดของวัตถุเล็กเมื่อเปรียบเทียบกับระยะการมอง ทำให้ระบบการมองเสมือนเป็น orthrographic ซึ่งตามหลักมาตราฐานทางเรขาคณิตถือว่าเส้นการมองอยู่บนแกน z และจุดของวัตถุ (x,y,z) ปรากฏบนภาพ (u,v) ซึ่ง u = x และ v = y ดูรูป 4.9(ก) กรณีของ perspective บนระนาบ ดังรูป 4.9(ข) เห็นได้ว่าตำแหน่งบนวัตถุ (x,y,z) ปรากฏบนภาพ (u,v) ซึ่ง u = (x/-z) และ v = (y/-z)



$$U = X$$

$$V = Y$$

รูป 4.9(ก) การโปรเจคแบบ orthographic



$$U = -\frac{X}{Z}$$

$$V = -\frac{Y}{Z}$$

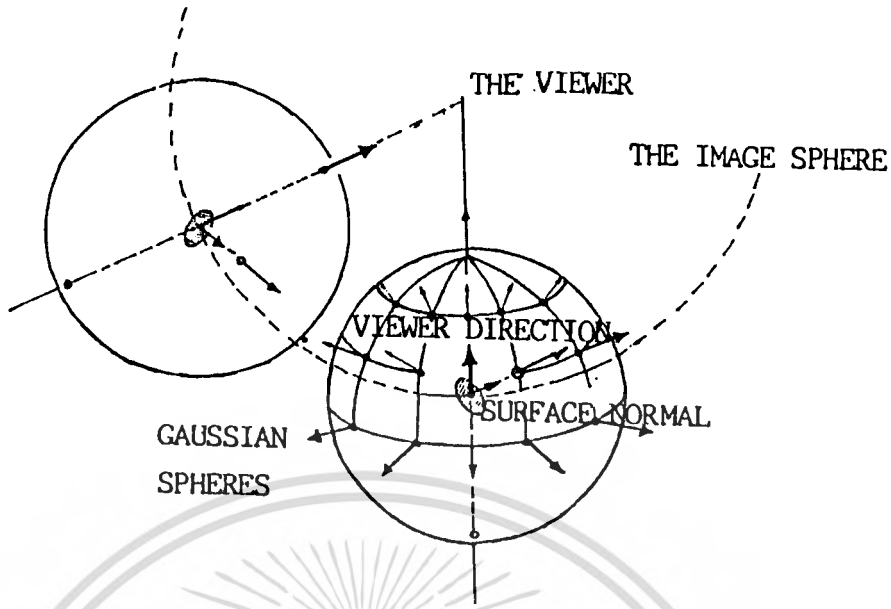
รูป 4.9(ข) การโปรเจคแบบ plane perspective

การโปรเจคทั้งแบบ orthographic และ spherical นั้น เส้นการมองตั้งฉากกับระนาบของภาพ(ทรงกลม)เสมอ และไม่มีผลการผิดเพี้ยนที่เกิดจากตำแหน่งบนระนาบของภาพ ซึ่งการโปรเจคแบบ perspective มีผลมากเนื่องจากมุมที่เกิดจากเส้นการมองและระนาบของภาพขึ้นอยู่กับระยะต่าง ๆ ของวัตถุ ทำให้รูปทรงของวัตถุที่ปรากฏบนภาพขึ้นอยู่กับตำแหน่งการวางของวัตถุ

ในกรณีของ spherical perspective นั้น เราสามารถสร้างรายระเอียดเป็นรูปแบบที่แน่นอนของ gradient map ได้ เนื่องจากทิศทางการมองตั้งฉากกับระนาบของภาพทุกตำแหน่ง และลักษณะการเกิดของภาพสอดคล้องกับความเป็นไปได้ทางฟิสิกซ์

4.3.2 Gaussian sphere และ Image sphere

gaussian sphere เป็นรูปทรงกลมซึ่งเส้นต่อระหว่างขั้วเหนือและขั้วใต้วางอยู่บนแนวแกน z และมีรัศมีหนึ่งหน่วย ถ้าบริเวณพื้นผิววางอยู่ตรงศูนย์กลางของทรงกลมพื้นที่บนผิวนั้นจะมีทิศ



รูป 4.10 ทรงกลมเกาส์เซียนและทรงกลมภาพ

ทางพุ่งออกสู่จุดใดจุดหนึ่งบนทรงกลม เมื่อกำหนดตำแหน่งการมองอยู่บนแนวแกน z จากทางขั้วเหนือของทรงกลม

ยังมีลักษณะทรงกลมอีกส่วนหนึ่ง ดังรูป 4.10 ซึ่งเรียกว่า image sphere ตำแหน่งผู้สังเกตอยู่ที่ศูนย์กลาง เส้นต่อขั้วเหนือและใต้ของ gaussian sphere จะผ่านศูนย์กลางของ image sphere

4.3.3 การโปรเจคของทรงกลมเกาส์เซียน[3]

เนื่องจากทรงกลมเกาส์เซียนเป็นรูป 3 มิติ จึงไม่สะดวกในการแทนลักษณะความเข้มของแสงบนระดับของพื้นผิว จึงให้โปรเจคลงบนพื้นระนาบซึ่งการ mapping นี้มีหลายวิธีการ เมื่อกำหนดให้ n เป็น unit vector ของทิศทางที่พุ่งออกมาจากพื้นผิว ทำให้ coordinate ของจุดบนผิวของทรงกลมสัมพันธ์กับค่าความลาดเอียง (p,q) ในกรณีนี้ พื้นระนาบของการโปรเจคเรียกว่า gradient space

$$n = (-p,-q,1)/(1 + p^2 + q^2)^{1/2} \qquad (4.8)$$

เราพิจารณาการโปรเจคโดยใช้หลักการทางเรขาคณิต เสมือนเป็นลำแสงจากศูนย์กลางไปยังพื้นระนาบที่สัมผัสกับขั้วเหนือ(โดย p และ q มีแกนตรงข้ามกับ x และ y) การโปรเจคแบบนี้เรียกว่า gnomonic projection เห็นได้ว่าขั้วเหนือจะ map ลงบนจุด origin และส่วนบน

ของทรงกลมจะโปรเจคลงบนระนาบ p-q จุดต่าง ๆ บนเส้นศูนย์สูตรจะอยู่ที่อนันต์ และจุดที่อยู่ส่วนล่างของทรงกลมจะไม่โปรเจคลงบนระนาบ p-q

แต่ละจุดบน gradient space จะสัมพันธ์กับความลาดเอียงของผิว และสัมพันธ์กับระดับความเข้มของมันด้วย ผลอันนี้เรียกว่า reflectance map ซึ่งการใช้วิธีดังกล่าวมีเหตุผลจาก

(1) ค่าต่าง ๆ บน coordinate สัมพันธ์กับ partial derivative อันดับแรกของความสูง z

(2) สามารถคำนวณความสูงจากการ integrate ค่า p และ q

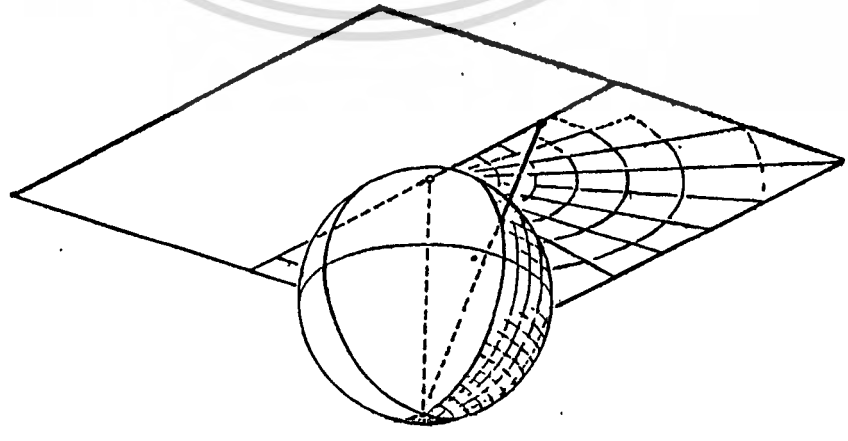
(3) ค่า line integration ของ (p,q) บนเส้นทางวงปิดจะเป็นศูนย์เสมอ

อย่างไรก็ดี เนื่องจากบนเส้นศูนย์สูตรของทรงกลมเกาส์เซียนนั้น สัมพันธ์โดยตรงกับ occluding boundary ซึ่ง gradient space ที่เกิดจากการโปรเจคของจุดเหล่านี้อยู่ที่อนันต์ทำให้ข้อมูลต่าง ๆ ของ occluding boundary ไม่สามารถโปรเจคลงใน gradient space

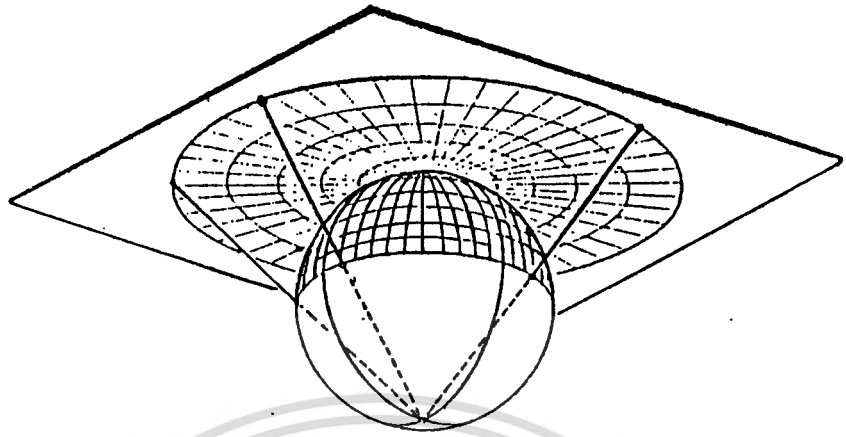
4.3.4 การโปรเจคแบบ stereographic[3]

วิธีการหนึ่งที่ใช้แก้ปัญหานี้ได้คือ การใช้โปรเจคแบบ stereographic เราสามารถคิดในลักษณะเรขาคณิตโดยถือว่าเกิดการโปรเจคไปสู่ระนาบซึ่งสัมผัสกับขั้วด้านเหนือ และศูนย์กลางของการโปรเจคอยู่ที่ขั้วด้านใต้ ดังรูป 4.11 แกนของระนาบ stereographic คือ f และ g เพื่อให้ต่างจาก gradient space ซึ่งสามารถแสดงความสัมพันธ์ระหว่างกันได้ดังนี้

$$f = 2p[(1 + p^2 + q^2)^{1/2} - 1)/(p^2 + q^2)]$$



รูป 4.11 การโปรเจคแต่ละจุดบนผิวทรงกลมตามแนวเส้นจากขั้วด้านหนึ่งไปสู่ขั้วตรงข้าม



รูป 4.12 การโปรเจคเส้นศูนย์สูตรเป็นวงกลมรัศมี 2 หน่วย

$$g = 2q[(1 + p^2 + q^2)^{1/2} - 1)/(p^2 + q^2)] \qquad (4.9)$$

หลังจากโปรเจคและ map วงกลมบนผิวของทรงกลมเกาส์เซียนไปสู่เส้นวงกลมบนระนาบแล้ว ทรงกลมทั้งหมดจึงถูก map ลงบนระนาบ มีจุดที่ขั้วใต้เพียงจุดเดียวเท่านั้นซึ่งปลายของเส้นโปรเจคจะเป็นอนันต์ ขณะที่บริเวณเส้นศูนย์สูตรทั้งหมดได้ถูก map ไปสู่เส้นวงกลมที่มีค่ารัศมีเท่ากับสอง ดังรูป 4.12 ซึ่งสามารถกำหนดค่าความสว่างของแต่ละจุดบน stereo plane เช่นเดียวกับที่กำหนดลงบน gradient space

การโปรเจคอีกชนิดหนึ่งคือ azimuthal equidistant เส้นรุ้งบนทรงกลมกลายเป็นเส้นวงกลมบนระนาบ และความยาวจากศูนย์กลางในระนาบ เท่ากับความยาวของพื้นผิวบนทรงกลมซึ่งวัดจากขั้วด้านเหนือ ไม่สามารถอธิบายในลักษณะการโปรเจคจากจุดใดที่คงที่ได้ เราสามารถมองในลักษณะของการกลิ้งลูกทรงกลมไปบนระนาบตามแนวของเส้นเวง โดยให้ศูนย์กลางบนระนาบอยู่ที่จุดบนขั้วด้านเหนือ ซึ่งเป็นจุดเริ่มต้น ถ้าเราให้แกนบนระนาบคือ a และ b ในกรณีนี้เราจะได้

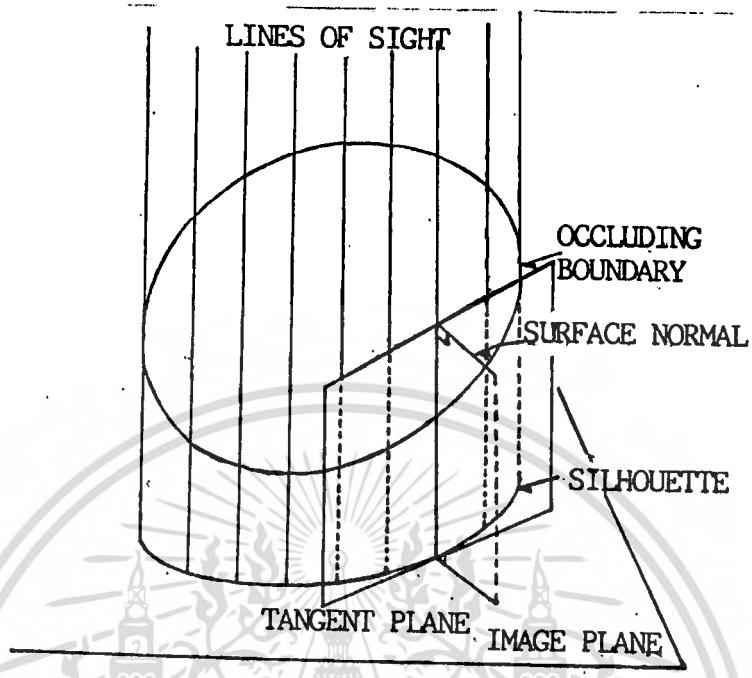
$$a = [p/(p^2+q^2)]\tan^{-1}(p^2+q^2)^{1/2}$$

$$b = [q/(p^2+q^2)]\tan^{-1}(p^2+q^2)^{1/2} \qquad (4.10)$$

ในทำนองเดียวกันกับ gnomonic projection นั่นคือมีการโปรเจคเฉพาะครึ่งบนของทรงกลม จุดบนเส้นศูนย์สูตรไม่ได้จบลงที่อนันต์ แต่ map ลงบนวงกลมรัศมี $\pi/2$ และบางครั้งเรียกว่า orthographic projection

4.3.5 Occluding boundary

ที่จุดต่าง ๆ บนขอบเขต occluding boundary เราสามารถหาค่าที่ถูกต้องของทิศ



รูป 4.13 เส้นสัมผัสของ occluding boundary

ทางตั้งฉากกับพื้นผิวได้ โดยยึดหลักความจริงดังนี้ กรณีแรกคือพิจารณา occluding boundary ซึ่งเส้นตรงลากจากพื้นผิวมายังตำแหน่งการมองจะต้องสัมผัสที่จุดนั้น ดังรูป 4.13 ซึ่งทิศทางการมองวางอยู่บนพื้นระนาบสัมผัสกับพื้นผิว บนจุดตำแหน่งซึ่งทิศทางการมองลากครูดไปตลอดแนวของพื้นผิว ทำมุมฉากกันทิศทางตั้งฉากของผิวที่ตำแหน่งนั้นด้วย กรณีที่สองคือเส้นแนวของการมองตั้งฉากกับระนาบของภาพ (เมื่อเรากำหนดให้เป็นการโปรเจคภาพแบบขนาน เส้นทางของการมองทุกเส้นจึงขนานกับแกน z และตั้งฉากกับระนาบของภาพ) และระนาบสัมผัสย่อมตั้งฉากกับทิศทางตั้งฉากของพื้นผิวด้วย ดังนั้นจึงโปรเจคลงมาเป็นเส้นตรง และเส้นตรงนี้เป็นเส้นสัมผัสของ silhouette ในระนาบของภาพ

เห็นได้ว่าทิศทางตั้งฉากในระนาบของภาพขนานกับทิศทางตั้งฉากกับพื้นผิวที่สัมพันธ์กับจุดบน occluding boundary เราจึงสามารถหาลักษณะของพื้นผิวจากจุดทุกจุดบน occluding boudary

4.3.6 Specular point และ singular point

สมมุติให้ความส่องสว่างของแสงเกิดจากจุดกำเนิดแสงเพียงจุดเดียว ในขั้นตอนแรกพิจารณาได้ว่าลักษณะพื้นผิวที่ตำแหน่งของทิศทางตั้งฉากกับพื้นผิว เป็นทิศทางเดียวกับแหล่งกำเนิดแสง เราเรียกตำแหน่งของพื้นผิวอันนี้บนทรงกลมเกาส์เซียนว่า source spot เพื่อให้เห็นชัดว่าเป็นทิศทาง

เดียวกับแหล่งกำเนิดแสง สังเกตได้ว่าถ้าแสงมาทางด้านหลังของวัตถุ source spot ปรากฏที่ขั้วด้านใต้ ดังนั้น source spot สามารถกำหนดเป็นจุดบนระนาบของ stereographic projection ขั้นตอนต่อไปเราพิจารณาลักษณะการสะท้อนแสงแบบกระจกเงา พื้นผิวบริเวณที่แสงสะท้อนจากทิศทางแหล่งกำเนิดพุ่งตรงเข้าสู่ตำแหน่งการมองเรียกว่า specular point ซึ่งจะเกิดขึ้นเฉพาะเมื่อมุมตกกระทบ i เท่ากับมุม emittance e เมื่อลำแสงตกกระทบทิศทางตั้งฉากกับผิวและลำแสงสะท้อนวางอยู่บนระนาบเดียวกัน พื้นผิวบริเวณนี้จะอยู่กึ่งกลางระหว่าง source spot บนทรงกลมเกาส์เซียนและขั้วด้านเหนือ

ถ้าเราให้ (p_s,q_s) เป็นจุดใน gradient space ซึ่งสัมพันธ์กับ source spot บนทรงกลมเกาส์เซียน ดังนั้นพื้นผิวบริเวณของ specular reflection มีความลาดเอียง (p_m,q_m) จะได้

$$p_m = p_s[(1+ p_s^2 + q_s^2)^{1/2}]/(p_s^2 + q_s^2)$$
$$q_m = q_s[(1+ p_s^2 + q_s^2)^{1/2}]/(p_s^2 + q_s^2) \qquad (4.11)$$

สามารถแสดงให้เห็นจากข้อสังเกตว่า (p_m,q_m) วางอยู่บนทิศทางเดียวกันใน gradient space กับ (p_s,q_s) และมุมระหว่าง $(0,0,1)$ และ $(-p_m,-q_m,1)$ ต้องเท่ากับมุมระหว่าง $(-p_m,-q_m,1)$ และ $(-p_s,-q_s,1)$ ในทำนองเดียวกันเราสามารถหาค่า specular spot บน stereographic plane ได้ดังนี้

$$f_m = f_s[(1 + f_s^2 + g_s^2)^{1/2} - 1]/(f_s^2 + g_s^2)$$
$$g_m = g_s[(1 + f_s^2 + g_s^2)^{1/2} - 1]/(f_s^2 + g_s^2) \qquad (4.12)$$

และสำหรับ azimuth equidistant คือ

$$a_m = a_s/2 \quad \text{และ} \quad b_m = b_s/2 \qquad (4.13)$$

สำหรับ point source และพื้นผิว specular ซึ่งเป็นอุดมคติค่า specular point จะมีความสว่างเป็นอนันต์ ส่วนแหล่งกำเนิดที่มีลำแสงกระจายและพื้นผิวที่ราบเรียบอย่างไม่สมบูรณ์ ทำให้การสะท้อนของ specular เกิดการกระจายขึ้นได้ ค่าความสว่างจึงมีค่าสูงอยู่ในขอบเขตจำกัด

สำหรับพื้นที่ผิวส่วนที่ไม่เกิด specular reflection จะยังคงเป็นค่าที่มีลักษณะของผิวเป็นตัวกำหนดค่าความสว่างของแสงเช่น พื้นผิวของ Lambertian ค่าที่สว่างมากที่สุดคือบริเวณพื้น

ผิวตั้งฉากกับลำแสง เมื่อค่าความสว่างของแสงบนวัตถุสัมพันธ์กับลักษณะพื้นผิว จุดที่เรียกว่า singular point ก็คือจุดบนภาพซึ่งค่าระดับความเข้มของแสงปรากฏต่อผู้สังเกต ดังนั้น specular point ก็คือ singular point ตำแหน่งที่มีเงื่อนไขพิเศษ ด้วยเหตุผลดังกล่าวเราจึงพิจารณาลักษณะพื้นผิวโดยอ้างถึง singular point แต่เพียงอย่างเดียว

4.4 การเกิดภาพวัตถุ

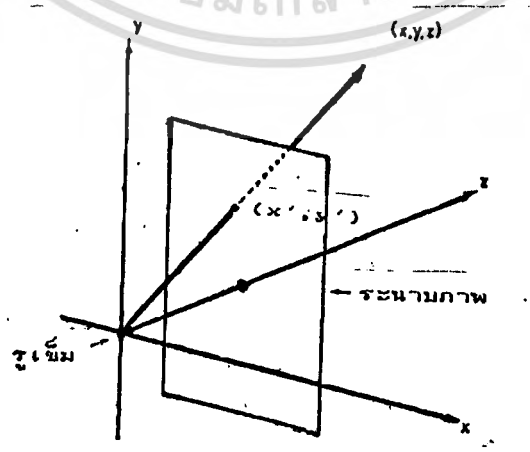
การที่เรามองเห็นวัตถุที่มีขนาดทึบในตัวกลางโปร่งแสง ภาพที่ได้เป็นภาพ 2 มิติที่มีระดับความเข้มของแสง เราจึงแบ่งปัญหาออกได้เป็นสองส่วน ส่วนแรกเกี่ยวข้องกับลักษณะทางเรขาคณิตและส่วนที่สองเกี่ยวข้องกับความเข้มของแสงที่ปรากฏบนภาพ ความสัมพันธ์ระหว่าง coordinate ของวัตถุและภาพนั้นพิจารณาจากสมการของการโปรเจคแบบ perspective ดังรูป 4.14 ซึ่ง f เป็นความยาวของโฟกัส

$$x' = (x/z)f \text{ และ } y' = (y/z)f \qquad (4.14)$$

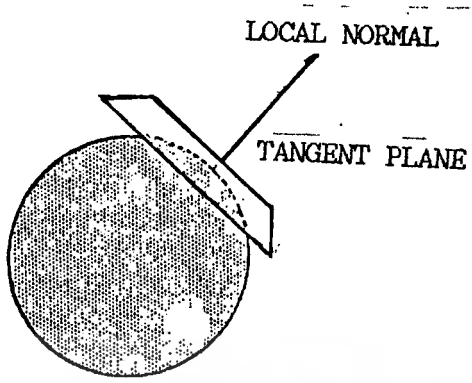
แต่เพื่อความสะดวกในกรณีที่ตำแหน่งผู้สังเกตอยู่ไกลจากวัตถุ ภาพที่เกิดขึ้นจึงเสมือนเป็นการโปรเจคแบบ orthographic และ z ในสมการถูกพิจารณาเป็นค่าคงที่

4.4.1 ลักษณะของพื้นผิว[17]

เราต้องศึกษาลักษณะของเส้นต่อระหว่างลำแสงวัตถุและจุดสังเกต เพื่อพิจารณาค่าความเข้มของแสงสะท้อนไปยังจุดสังเกตซึ่งเกิดจากพื้นผิวของวัตถุ ในการหาพื้นผิวแต่ละบริเวณนั้นมีหลายวิธีการ ตัวอย่างเช่นการสร้างสมการกำหนดผิวของระนาบหรือทิศทางตั้งฉากกับพื้นผิว ถ้าสมการ



รูป 4.14 การโปรเจคภาพทางเรขาคณิต



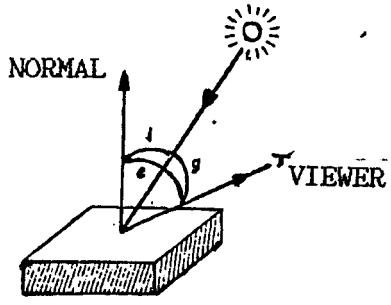
รูป 4.15 ลักษณะของผิววัตถุตามคำจำกัดความ

ของพื้นระนาบคือ $ax + by + cz = d$ ดังนั้น ค่าของทิศทางที่ตั้งฉากกับพื้นผิวคือ (a,b,c)

เรานำสมการของแต่ละสมการมาประยุกต์ใช้กับระนาบสัมผัสที่พื้นผิว ในช่วงบริเวณพื้นผิวซึ่ง smooth คือ $(z_x, z_y, -1)$ ซึ่ง z_x และ z_y คือค่า partial derivative ของ z เพื่อความเหมาะสมจึงใช้ค่า p,q แทนค่า ดังนั้นทิศทางตั้งฉากของพื้นผิวมีค่าเป็น $(p,q,-1)$ เห็นได้ว่าลักษณะของผิวกำหนดได้จากปริมาณ (p,q) ซึ่งเรียกว่า gradient ดูรูป 4.15

4.4.2 ความเข้มแสงของภาพ

ปริมาณแสงสะท้อนโดยพื้นผิวเล็ก ๆ ขึ้นกับลักษณะโครงสร้างเล็ก ๆ ของพื้นผิวปริมาณแสงตกกระทบ ถ้าเราพิจารณาว่าพื้นระนาบสัมผัสวางอยู่เหนือจุดตำแหน่งของผิว หมายความว่าแสงได้มาจากทิศทางส่วนบนของทรงกลม ซึ่งเราเริ่มพิจารณาแต่ละส่วนก่อนจึงนำผลที่ได้มาประกอบกัน พื้นผิวส่วนใหญ่มีสภาพของการสะท้อนที่แน่นอน และด้วยเหตุนี้ความเข้มแสงบนภาพจึงให้ค่าลักษณะพื้นผิวได้โดยไม่คำนึงถึงลักษณะซับซ้อนของแหล่งกำเนิดแสง เรานำเอาการประยุกต์ใช้ reflectance map มาเป็นแนวทาง



รูป 4.16 ลักษณะของมุมตกกระทบ มุมสะท้อน และมุม phase

กำหนดให้แหล่งกำเนิดแสงเป็น point source ลักษณะทางเรขาคณิตประกอบด้วย มุม 3 มุม คือมุมตกกระทบ มุมสะท้อน และมุม phase ดังรูป 4.16 มุมตกกระทบ i คือมุมระหว่างลำแสงตกกระทบกับทิศทางตั้งฉากของพื้นผิว มุมสะท้อน e คือมุมระหว่างทิศทางการมองกับทิศทางตั้งฉากของพื้นผิว และมุม phase g เป็นมุมระหว่างมุมตกกระทบและมุมสะท้อน ฟังก์ชั่นของการสะท้อนหาได้จากปริมาณสะท้อนของแสงตกกระทบในแต่ละทิศทาง นั่นคืออัตราส่วนระหว่างปริมาณสะท้อนของแสงตกกระทบต่อหน่วยของพื้นที่ ต่อหน่วยของมุมในทิศทางการมอง

ถ้าความส่องสว่างของแสงคือ E และค่าความสว่างของแสงในทิศทางการมองคือ B ค่าการสะท้อนหาได้จาก E/B ซึ่งอาจเขียนเป็น $\rho(i,e,g)$ เพื่อเน้นให้เห็นว่าขึ้นอยู่กับลักษณะมุมทั้งสาม

4.4.3 Reflectance Map

ปริมาณของแสงสะท้อนบนพื้นผิวขึ้นอยู่กับลักษณะของพื้นผิว และชนิดของแหล่งกำเนิดแสง เมื่อเรากำหนดชนิดของวัตถุและลักษณะของแหล่งกำเนิดแสง ทำให้สามารถกำหนดลักษณะของพื้นผิวจากทิศทางตั้งฉากของมันได้ทุก ๆ จุดใน gradient space ดังนั้นความเข้มบนภาพเป็นฟังก์ชั่นของ p,q ซึ่งไม่ใช่การเปลี่ยนจากภาพโดยการมอง แต่มันขึ้นอยู่กับคุณสมบัติของผิวและตำแหน่งจุดกำเนิดแสง

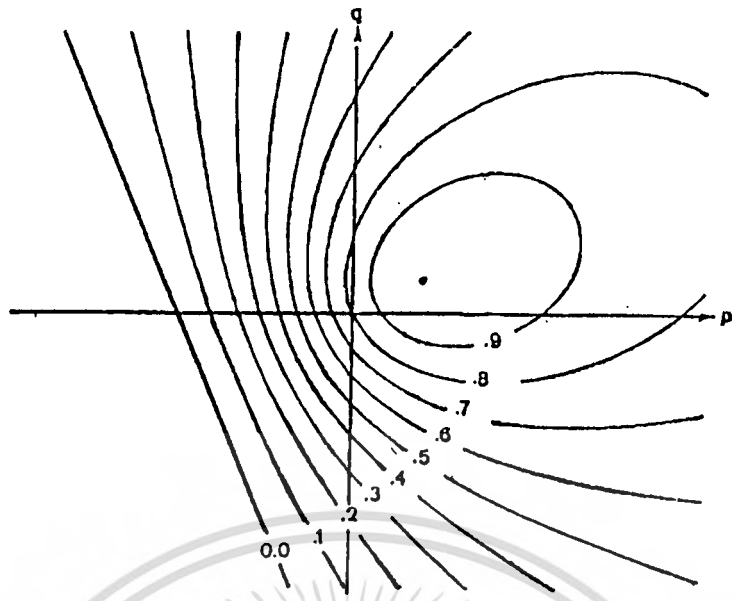
ถ้าเราวัดแสงจากจุดใดจุดหนึ่งบนวัตถุ เราทราบแต่เพียงว่าค่าพื้นผิวนั้นเป็นหนึ่งในหลาย ๆ ค่าที่น่าจะเป็นไปได้ แต่ไม่อาจกำหนดให้แน่นอนลงไป เราอาจจำกัดลงได้ว่ามันเป็นจุด ๆ หนึ่งบน gradient space ซึ่งมีระดับความเข้มของแสงเท่า ๆ กัน

บนพื้นผิวที่ราบเรียบและทิศทางแหล่งกำเนิดแสงอยู่ในทิศทางเดียวกัน บนพื้นผิวที่มีลักษณะการสะท้อนเท่ากันทุกทิศทาง ขนาดของแสงสะท้อนขึ้นอยู่กับค่าโคซายน์ของมุมตกกระทบ เพื่อแสดงการคำนวณให้เห็นได้ชัด เราจะกำหนดให้แหล่งกำเนิดแสงวางอยู่ใกล้ผู้สังเกต มุมตกกระทบเท่ากับมุมซึ่งเกิดจากทิศทางตั้งฉากของพื้นผิวและผู้สังเกต ค่าโคซายน์ของมุมตกกระทบคือการ dot product ของ unit vector ต่อไปนี้

$$\cos(i) = [(p,q,-1).(0,0,-1)]/[|(p,q,-1)||(0,0,-1)|]$$

$$= 1/[(1+p^2+q^2)^{1/2}] \qquad (4.15)$$

เราอาจหาคำตอบเดียวกันได้จากการหาระยะทางจากศูนย์กลางใน gradient



รูป 4.17 เส้นทางเดินของ E = cos(e)

space ซึ่งเป็นค่า tangent ของมุมระหว่างทิศทางตั้งฉากพื้นผิวและทิศทางการมอง

$$(p^2+q^2)^{1/2} = \tan(e) \qquad (4.16)$$

$$\cos(e) = 1/(1+\tan^2(e))^{1/2} \qquad (4.17)$$

และ i = e

ถ้าเขียน reflectance ในฟังก์ชั่นของ p และ q เราจะได้ค่าสูงสุดอยู่ที่ศูนย์กลางเท่ากับ 1 และรูปวงกลมที่สมดุลย์เรียงลำดับลงไปถึงศูนย์ ซึ่งเขียนได้บน gradient space ที่ตำแหน่งอนันต์ ได้เป็น reflectance map ที่มีลักษณะดังรูป 4.17

ค่าความเข้มของแสงตามที่กำหนดนั้นเป็นแนวเส้นทางเดินบน gradient space ในกรณีนี้เป็นเส้นวงกลม ค่าความเข้มบนภาพบอกเราได้เพียงว่ามันอยู่บนวงกลมหนึ่งใน gradient space เมื่อแหล่งกำเนิดแสงอยู่ห่างจากจุดสังเกตเราจะพบความยุ่งยากทางเรขาคณิตของมุมตกกระทบและมุมสะท้อนที่เกิดจากทิศทางของแหล่งกำเนิดแสงนั้น

มุมตกกระทบ มุมสะท้อน และมุม phase[17] สำหรับพื้นผิวที่มีการสะท้อนของมุมตกกระทบ มุมสะท้อนและมุม phase อย่างราบรื่น เราเขียนสมการได้เป็น I = cos(i),E =cos(e) และ G = cos(g) ซึ่งหาได้จากการคูณทางเวกเตอร์ มีแหล่งกำเนิดแสงหนึ่งตำแหน่งในทิศทาง $(p_s,q_s,-1)$ และตำแหน่งการมองในทิศทาง (0,0,-1) ดังนั้น

$$G = 1/(1+p_s^2+q_s^2)^{1/2} \qquad (4.18)$$

$$E = 1/(1+p^2+q^2)^{1/2} \tag{4.19}$$

และ

$$I = (1+p_s p+q_s q)/[(1+p^2+q^2)^{1/2}(1+p_s^2+q_s^2)^{1/2}]$$

$$= (1+p_s p+q_s q)EG \tag{4.20}$$

เราสามารถที่จะคำนวณหาค่า I,E,G ของจุดใด ๆ บน gradient space เนื่องจาก G เป็นค่าคงที่และสมมุติฐานของการโปรเจคแบบ orthogonal เราจึงเห็นว่าเส้นเดินทางของค่าคงที่ E เป็นรูปวงกลมโดยรอบจุดศูนย์กลางบน gradient space เมื่อกำหนดให้ I มีค่าคงที่ เราจะได้สมการ polynomial อันดับสองใน p และ q และเส้นทางเดินของค่า I คงที่เป็นภาคตัดขวางของรูปทรงกรวย โดยมีเส้นตรงเป็นส่วนแยกระหว่างส่วนเงาและส่วนสว่าง ตัวอย่างเช่น เส้นตรงเกิดเมื่อ $i = \pi/2$ นั่นคือ $I = 0$ หรือ $1+p_s p+q_s q = 0$ และถ้า $I = 1$ ผลคือจุดเพียงจุดเดียวที่ $p = p_s$ และ $q = q_s$

จากวิธีการทางเรขาคณิตซึ่งทำให้เส้นทางเดินคงที่เป็นภาคส่วนตัดของทรงกรวยซึ่งเกิดจากทิศทางของมุมตกกระทบ แกนของรูปกรวยอยู่ที่ทิศทางของแสง $(p_s,q_s,-1)$ เราหาจุดอื่น ๆ บน gradient space ด้วยการหาค่าเส้นตัดของรูปทรงกรวยบนแผ่นระนาบที่มีระยะห่างจากศูนย์กลางหนึ่งหน่วย เมื่อเปลี่ยนค่ามุมของ I ไปย่อมเกิดเป็นเส้นทางรูปลายเส้นโอบรอบต่อเนื่องกันไปในลักษณะรูปตัดขวางของทรงกรวย

4.4.4 การหาค่า reflectance map[11]

จากสมมุติฐานของความ smooth เราหาค่าส่วนของความกลมกลืนได้คือ

$$s_{ij} = \tfrac{1}{4}[(f_{i\pm1,j}-f_{i,j})^2+(f_{i,j\pm1}-f_{i,j})^2$$

$$+(g_{i\pm1,j}-g_{i,j})^2+(g_{i,j\pm1}-g_{i,j})^2 \tag{4.21}$$

ค่าความผิดพลาดหาได้จาก

$$r_{i,j} = [I_{ij}-R(f_{ij},g_{ij})^2] \tag{4.22}$$

ค่า f และ g เป็นลักษณะพื้นผิว I_{ij} เป็นค่าความเข้มวัดที่ (i,j) ส่วน R เป็น reflectance map ผลรวมของค่าความเพี้ยนคือ

$$e = \sum_i \sum_j (s_{ij}+\lambda r_{ij}) \tag{4.23}$$

factor λ เป็นค่าที่ปรับความเกี่ยวเนื่องระหว่างความผิดพลาดกับส่วนที่เกิดจากความ smooth ค่าอนุ

พันธ์ของ e สามารถนำมาสู่สมการ

$$f_{ij}^{n+1} = f_{ij}^{*n} + \lambda[I_{ij}-R(f_{ij}^{*n},g_{ij}^{*n})](\partial R/\partial f) \qquad (4.24)$$

$$g_{ij}^{n+1} = g_{ij}^{*n} + \lambda[I_{ij}-R(f_{ij}^{*n},g_{ij}^{*n})](\partial R/\partial g) \qquad (4.25)$$

ซึ่ง

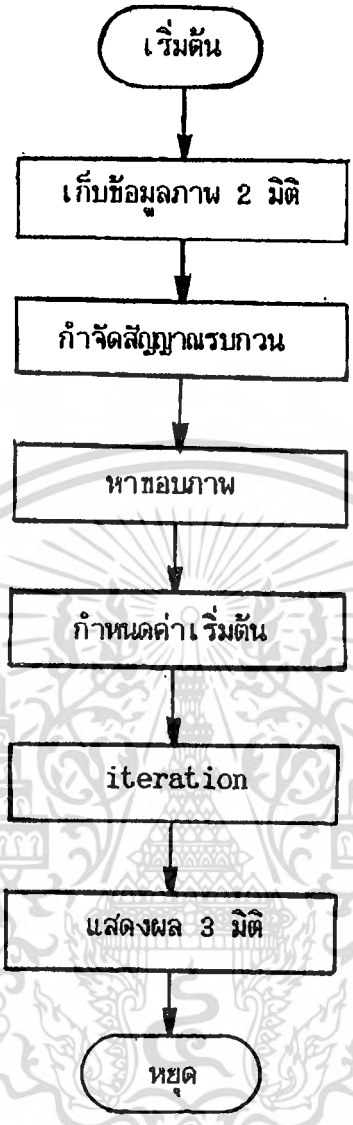
$$f_{ij}^{*n} = \tfrac{1}{4}[f_{i+1,j}^{n} + f_{i,j+1}^{n} + f_{i-1,j}^{n} + f_{i,j-1}^{n}]$$

$$g_{ij}^{*n} = \tfrac{1}{4}[g_{i+1,j}^{n} + g_{i,j+1}^{n} + g_{i-1,j}^{n} + g_{i,j-1}^{n}]$$

ค่า f และ g ที่หาได้นี้จะเป็นค่าที่บอกลักษณะพื้นผิวของรูปภาพ 2 มิติของความเข้มแสงที่ตำแหน่งต่าง ๆ

4.5 การหารูปทรง 3 มิติจากภาพ 2 มิติ

จากทฤษฎีพื้นฐานที่ได้ค้นคว้ามานั้น ความเป็นไปได้ในการที่จะเริ่มต้นทดลองหาโครงสร้าง 3 มิติจากภาพ 2 มิตินั้น จำเป็นต้องกำหนดขอบเขตและเงื่อนไขของข้อมูลภาพที่นำมาประมวลผล ข้อมูลภาพวัตถุนั้นจะต้องเป็นภาพที่มีพื้นผิวต่อเนื่อง มีความกลมกลืนของระดับพื้นผิว และใช้ข้อมูลแสงเป็นสิ่งที่จะบ่งบอกระดับความลึกของพื้นผิววัตถุ ในขั้นตอนแรกนั้นการกำหนดทิศทางของแหล่งกำเนิดแสงให้เป็นทิศทางเดียวกับทิศทางการมอง เพื่อให้สามารถวิเคราะห์พื้นผิวที่มีระนาบในทิศทางซึ่งมองเห็นได้มีระดับความเข้มแสงมากกว่าศูนย์ เพราะค่าโคซายน์ของมุมระหว่างทิศทางการมองกับทิศทางตั้งฉากกับพื้นระนาบมากกว่าศูนย์ ตามทฤษฎีในบทที่ 3 ข้อมูลแสงนั้นจะถูกนำมาเป็นข้อมูลชดเชยในการที่ค่าระดับความลึกสูญเสียไปเมื่อมีการโปรเจคภาพ 3 มิติลงบนระนาบ 2 มิติ แต่ข้อมูลที่จะถูกนำมาเป็นค่าเริ่มต้นนั้นต้องเป็นข้อมูลที่กำหนดระดับพื้นผิวบริเวณนั้นได้อย่างถูกต้อง ดังนั้นภาพวัตถุจึงต้องเป็นภาพที่มีบริเวณของ occluding boundary โดยรอบวัตถุนั้นและมีทิศทางจากบริเวณดังกล่าวต่อเนื่องกันตลอดพื้นผิวของวัตถุ เมื่อได้ข้อมูลภาพภายใต้เงื่อนไขที่กำหนดไว้แล้ว นำมากำจัดสัญญาณรบกวน จากทฤษฎีในบทที่ 2 หัวข้อ 2.3 เมื่อได้ภาพที่มีสัญญาณรบกวนน้อยโดยที่ไม่สูญเสียข้อมูลบริเวณขอบภาพ นำมาหาเส้นขอบของวัตถุ เนื่องจากเป็นบริเวณ occluding boundary โดยใช้ zero crossing จากทฤษฎีในหัวข้อ 4.3.4 ค่าบริเวณนี้จะมีทิศทางพื้นผิวสัมผัสกับแสงเป็นมุมฉาก หมายความว่า $(p^2+q^2)^{1/2}$ เป็นค่าอนันต์ หรือ $(f^2+g^2)^{1/2}$ มีค่าเท่ากับ 2 ทุกค่าบนเส้นขอบนั้น และนำค่ามาเป็นค่าเริ่มต้นประมวลผลตามทฤษฎีในหัวข้อ 4.4.4 ใช้วิธี iteration จนได้ค่าพื้นผิว จากนั้นนำเอาคอมพิวเตอร์กราฟฟิกและหลักการเรขาคณิตมาเขียนภาพวัตถุให้สามารถสื่อความหมายความลึกของภาพบนจอภาพได้



รูป 4.18 แสดงผังงานของการหารูปทรง 3 มิติจากภาพ 2 มิติ

บทที่ 5

การทดลอง

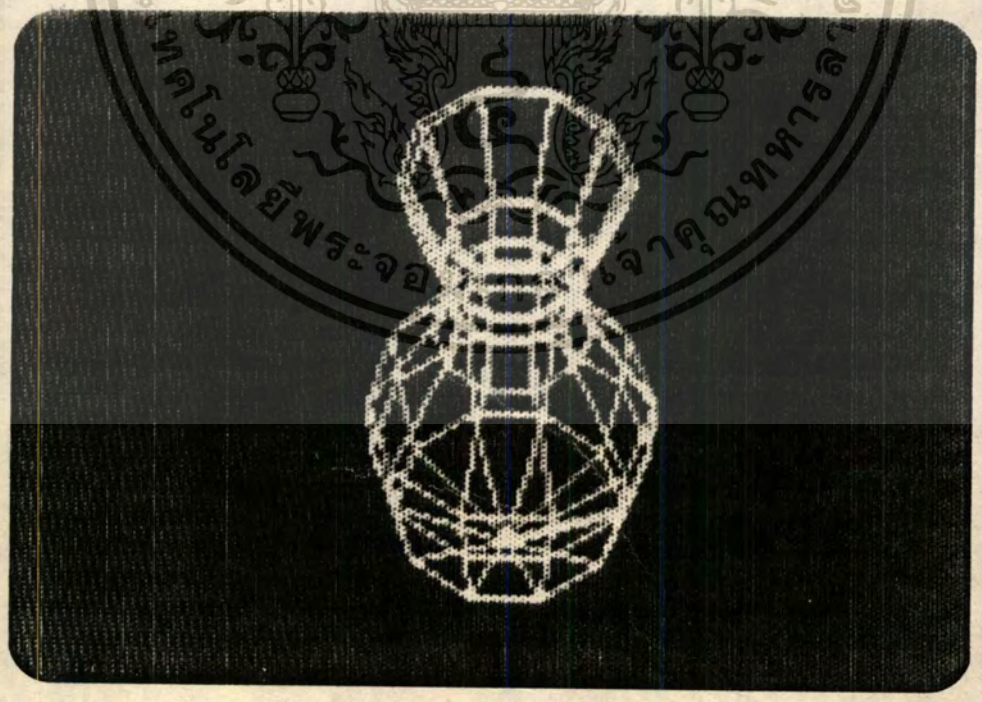
การคืนรูปทรง 3 มิติจากรูปภาพ 2 มิตินั้น ประกอบด้วยการทดลอง 2 ขั้นตอนคือ การจำลองภาพวัตถุบนระนาบ 2 มิติ และการทำย้อนกลับเพื่อหาค่าโครงสร้าง 3 มิติที่ปรากฏอยู่ในภาพบนระนาบ 2 มิติ ตามลำดับดังนี้

5.1 การจำลองภาพ

การพิจารณาความสัมพันธ์ระหว่างโครงสร้าง 3 มิติที่ปรากฏอยู่ในรูปวัตถุ 2 มิตินั้น เราต้องเริ่มจากการใช้หลักการของคอมพิวเตอร์กราฟฟิกสร้างภาพจำลองของวัตถุขึ้น โปรเจคลงบนพื้นระนาบ 2 มิติ และกำหนดระดับความเข้มของแสงให้แก่ภาพวัตถุบนระนาบ 2 มิติ ตามลำดับดังนี้

1. โครงสร้างวัตถุ

สร้างโครงสร้างภาพวัตถุ 3 มิติ โดยแบ่งส่วนของวัตถุเป็นรูปเหลี่ยมเล็ก ๆ การ

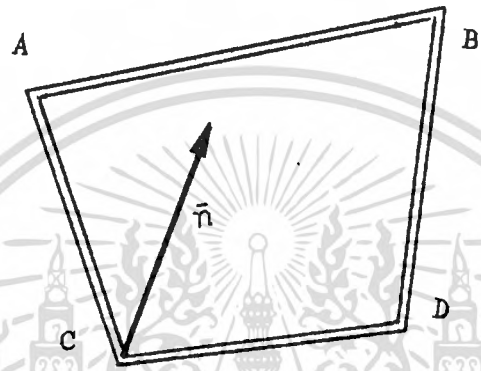


รูป 5.1 ส่วนประกอบเล็ก ๆ บนพื้นวัตถุ

สร้างภาพวัตถุนั้น เราอาจสร้างสมการของรูปวัตถุตามที่กำหนดขึ้น หรือสร้างจากส่วนประกอบจากโครงสร้างของวัตถุ แบ่งพื้นผิวของวัตถุนั้นออกเป็นส่วนย่อย ๆ ดังรูป 5.1

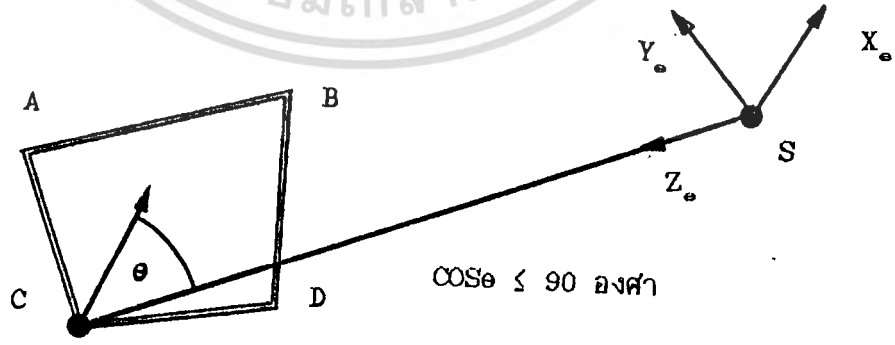
2 หาสมการพื้นระนาบของพื้นผิววัตถุ

วิธีการง่าย ๆ ในการหาสมการพื้นผิวบนรูปเหลี่ยมเล็ก ๆ ของวัตถุ หาจากการผลคูณทางเวกเตอร์ของด้านประกอบมุมยอดของรูปพื้นผิวเหลี่ยมเล็ก ๆ ดังกล่าว



รูป 5.2 พื้นระนาบของรูปเหลี่ยม

สมการพื้นระนาบจะได้จาก CD x CA ถ้ากำหนดลักษณะพื้นระนาบและจุดยอดเรียงไปตามลำดับ การหาค่าของสมการพื้นระนาบเหล่านั้นจะมีการคำนวณต่อเนื่องกันไป



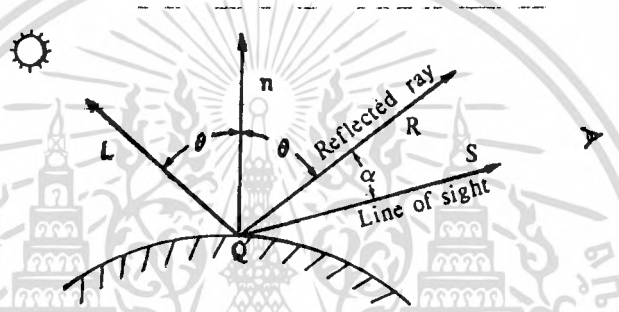
รูป 5.3 การพิจารณาชิ้นส่วนที่มองเห็นได้ของวัตถุ

3 คำนวณหาชิ้นส่วนของวัตถุที่สามารถมองเห็นได้

คำนวณหาชิ้นส่วนบนพื้นผิววัตถุ โดยแยกเฉพาะแต่ละชิ้นส่วนที่มองเห็นได้ จากการคำนวณหามุมที่เกิดขึ้นระหว่างทิศทางตั้งฉากของพื้นระนาบและทิศทางของตำแหน่งจุดมอง ซึ่งถ้ามุมที่เกิดขึ้นมีค่ามากกว่า 90 องศา ชิ้นส่วนนั้นจะเป็นชิ้นส่วนที่ไม่อาจมองเห็นได้

4 คำนวณระดับความเข้มแสง

ค่าความเข้มแสงบนพื้นวัตถุที่มองเห็นได้สัมพันธ์กับค่าโคซายน์ของมุมระหว่างทิศทางแสง ทิศทางการมอง และทิศทางตั้งฉากกับพื้นผิวระนาบนั้น



รูป 5.4 เวกเตอร์ที่ใช้คำนวณความเข้มของผิววัตถุ

I ความเข้มแสงที่ปรากฏบนพื้นผิววัตถุ

N ทิศทางตั้งฉากกับพื้นผิววัตถุ

L ทิศทางของแสง

R ทิศทางของแสงสะท้อน

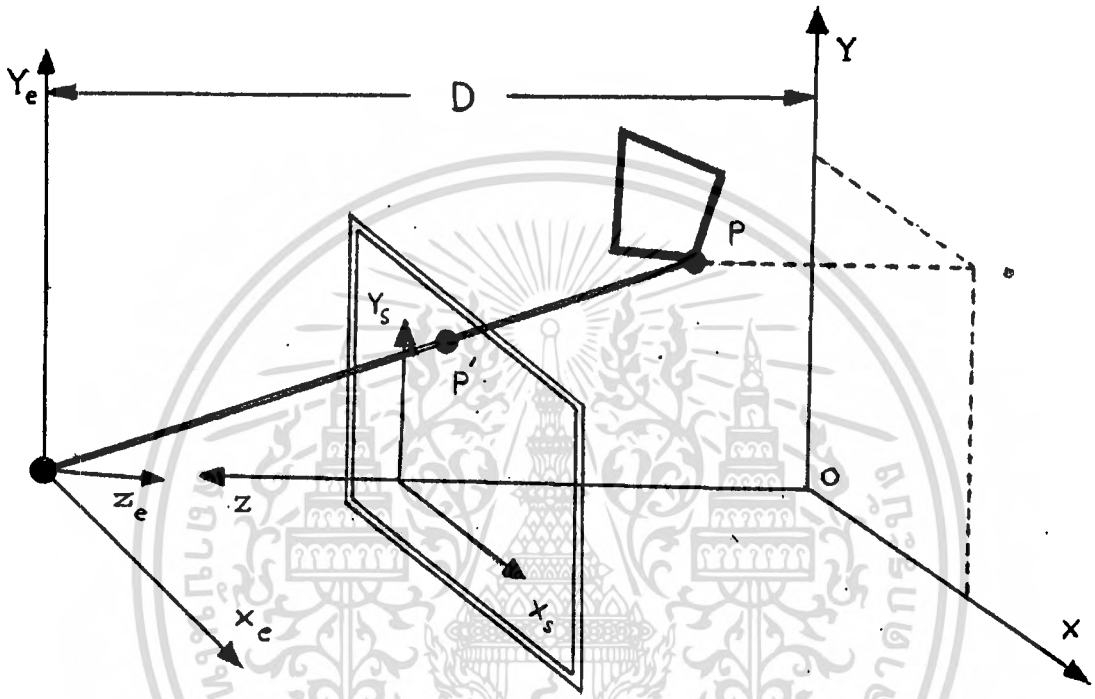
S ทิศทางจากตำแหน่งการมอง

ค่า L และ S เป็นค่าที่กำหนดให้ ส่วนค่า R คำนวณจากทฤษฎีสมการที่ 3.24 ค่า N คำนวณจากเวกเตอร์บนขอบระนาบของชิ้นภาพแต่ละชิ้น ค่า I คำนวณจากสมการที่ 3.16 โดยกำหนดค่าคงที่ตามสมควร ($d = 0, k = 1, I_a = 1, I_l = 10, K_s = 0.8, K_d = K_a = 0.15, n = 1$)

5 ทำเป็นภาพ 2 มิติ

โปรเจคชิ้นส่วนของพื้นผิววัตถุทุกชิ้นที่สามารถมองเห็นได้ ลงบนระนาบ 2 มิติ ซึ่งเป็นจอมอนิเตอร์ที่มีค่าต่างระดับสัญญาณความเข้มแสง 127 ระดับ

ใช้การโปรเจคแบบขนานของชิ้นภาพซึ่งมองเห็นได้ลงบนระนาบ 2 มิติบนจอคอมพิวเตอร์ในขณะเดียวกัน ค่าขนาดของชิ้นภาพนี้ก็โปรเจคลงบนหน่วยความจำที่ เซกเมนต์ (Segment) 8000 ขนาด 256 x 256 หน่วย โดยมีขนาดสัมพันธ์กัน การคำนวณแต่ละชิ้นของภาพซึ่งได้ผลว่าสามารถมองเห็นได้จะถูกคำนวณค่าความเข้มแสงในขั้นตอนที่ 4 และเก็บค่าลงในหน่วยความจำเพื่อนำค่าที่ได้ไป



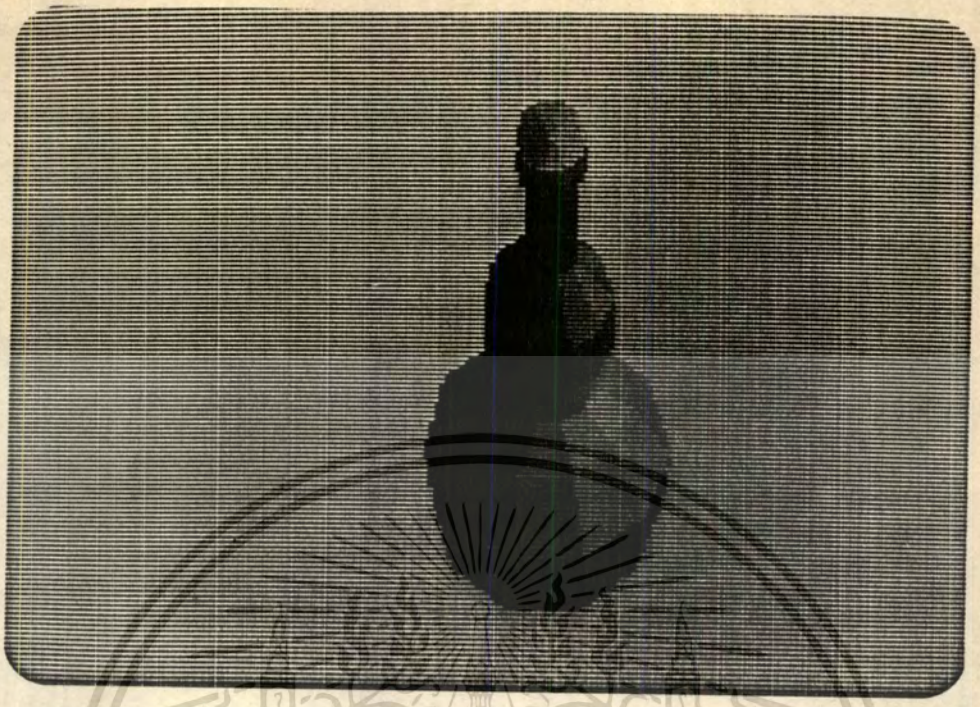
รูป 5.5 การโปรเจคภาพชิ้นส่วนวัตถุบนพื้นระนาบ 2 มิติ 127 ระดับ

สร้างบนจอภาพมอนิเตอร์ให้เป็นภาพวัตถุที่มีค่าความต่างระดับสัญญาณเทา

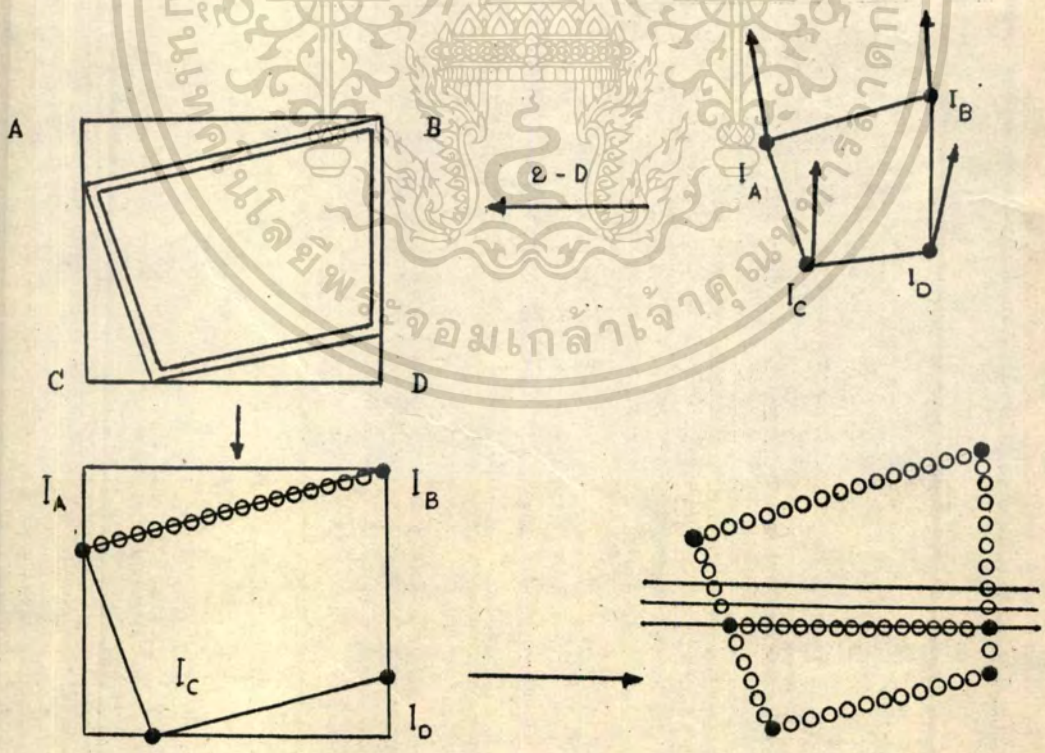
จากผลการทดลองเราจะได้ภาพที่มีความไม่ต่อเนื่องบนแต่ละชิ้นของภาพ ดังรูปที่ 5.6 ซึ่งเป็นภาพที่ยังไม่เหมือนภาพวัตถุจริง

6 เฉลี่ยความเข้มแสงบนพื้นวัตถุ

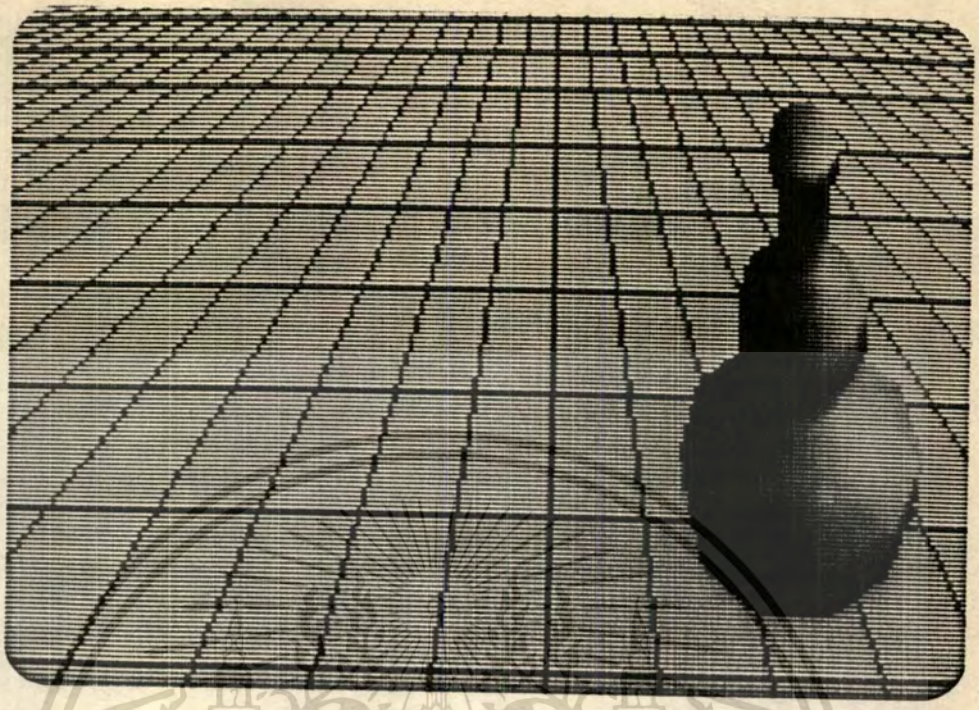
จากผลที่ได้ในขั้นตอนที่ 5 เราจำเป็นต้องแก้ไข โดยต้องเฉลี่ยค่าของแสงบนทุกชิ้นภาพให้มีค่าความเข้มต่อเนื่องกันไป การคำนวณค่าของความเข้มจะใช้การคำนวณ 1 ค่าต่อ 1 ชิ้นเช่นเดิมไม่ได้ ต้องคำนวณหาค่าความเข้มบนแต่ละยอดเหลี่ยมของชิ้นภาพแต่ละชิ้นนั้น จากการแทนค่า N ด้วยค่าเฉลี่ยของทิศทางตั้งฉากบนระนาบของชิ้นภาพที่อยู่โดยรอบของจุดยอดเหลี่ยมนั้น ส่วนค่าอื่นแทนค่าเช่นเดียวกับในขั้นตอนที่ 4



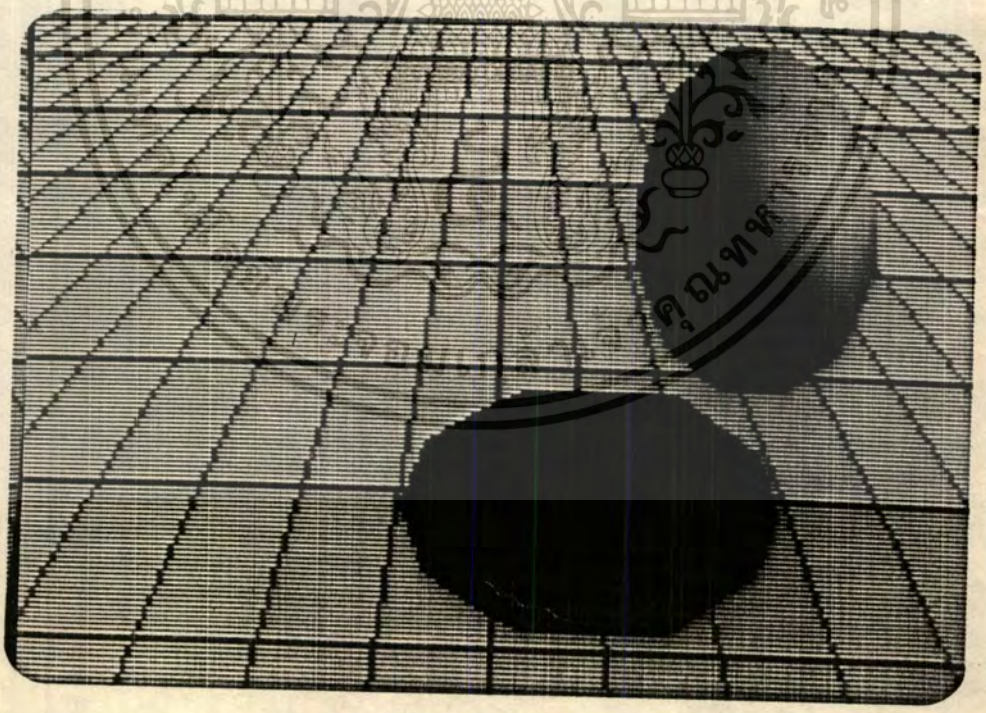
รูป 5.6 ภาพจำลองที่มีระดับความเข้มบนแต่ละชิ้นภาพไม่ต่อเนื่องกัน



รูป 5.7 แสดงการเฉลี่ยความเข้มระดับแสงบนชิ้นภาพของวัตถุ



รูป 5.8 ตัวอย่างภาพจำลองวัตถุ



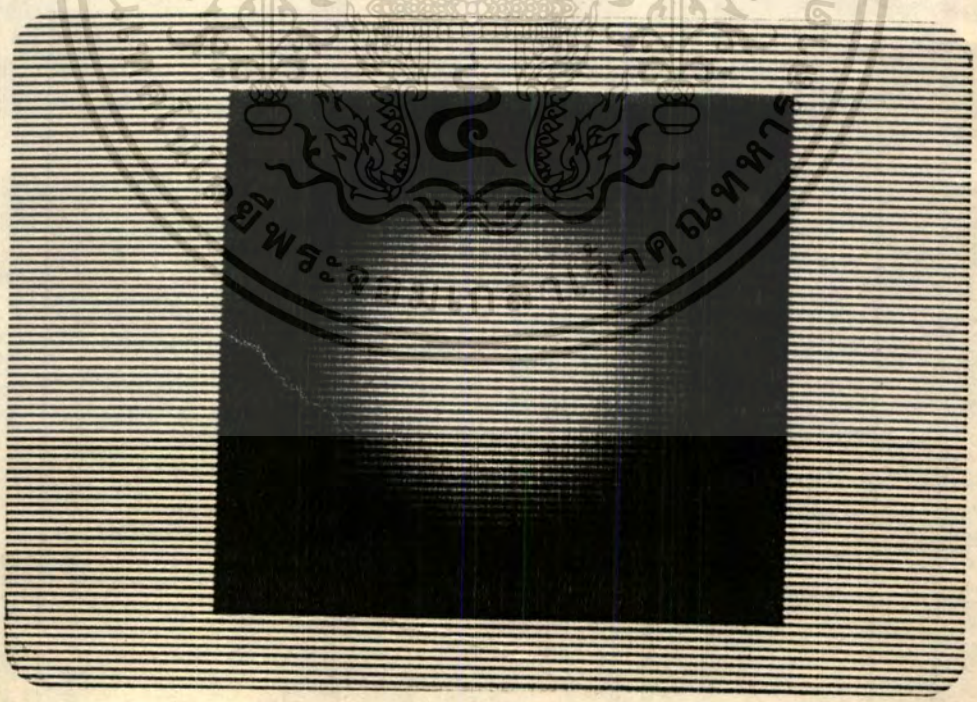
รูป 5.9 รูปจำลองของวัตถุอีกลักษณะ

คำนวณค่าความเข้มแสงบนจุดยอดทุกจุดบนพื้นผิวรูปเหลี่ยมของชิ้นวัตถุนั้น เฉลี่ยความเข้มจากจุดยอดดังกล่าวตามแนวเส้นขอบทุกด้านของรูปเหลี่ยมเล็ก ๆ นั้น ลากเส้นสแกนผ่านรูปเหลี่ยมโดยเฉลี่ยแสงแบบเชิงเส้นตามแนวเส้นสแกนนั้นจากค่าความเข้มแสงบนเส้นขอบของรูปเหลี่ยม ของแต่ละเส้นสแกนที่ผ่านไป เฉลี่ยตามแนวสแกนทุกเส้นที่ผ่านรูปเหลี่ยมของพื้นผิววัตถุแต่ละชิ้นจนครบบริเวณพื้นผิววัตถุในส่วนที่มองเห็นทั้งหมด

เมื่อทำครบทุกชิ้นจะได้ภาพจำลองวัตถุที่มีความเหมือนจริง ดังตัวอย่างรูป 5.8 และ 5.9

5.2 การหารูปทรง 3 มิติ จากภาพจำลอง 2 มิติ

จากทฤษฎีของคอมพิวเตอร์กราฟฟิกในการจำลองภาพ เรานำข้อมูลที่ได้ศึกษานั้นมาเป็นสมมุติฐานใช้ในการพิจารณาหาค่าความตื้นลึกของวัตถุที่ปรากฏในภาพธรรมชาติ E(x,y) และได้ทำการจัดข้อมูลภาพให้เหมาะสม (ภาษา BASIC ขนาด 64 x 64 และ PASCAL ขนาด 128 x 128) กำหนดตามตำแหน่งแบบดิจิตอลด้วยค่า i,j เป็นค่า I(i,j) ค่าความเข้มแสงซึ่งจัดขนาดได้นี้นำมาเป็นฟังก์ชั่นของความเข้มแสงบนวัตถุ R(f,g) เป็นฟังก์ชั่นของ reflectance map และอยู่ภายใต้ข้อกำ



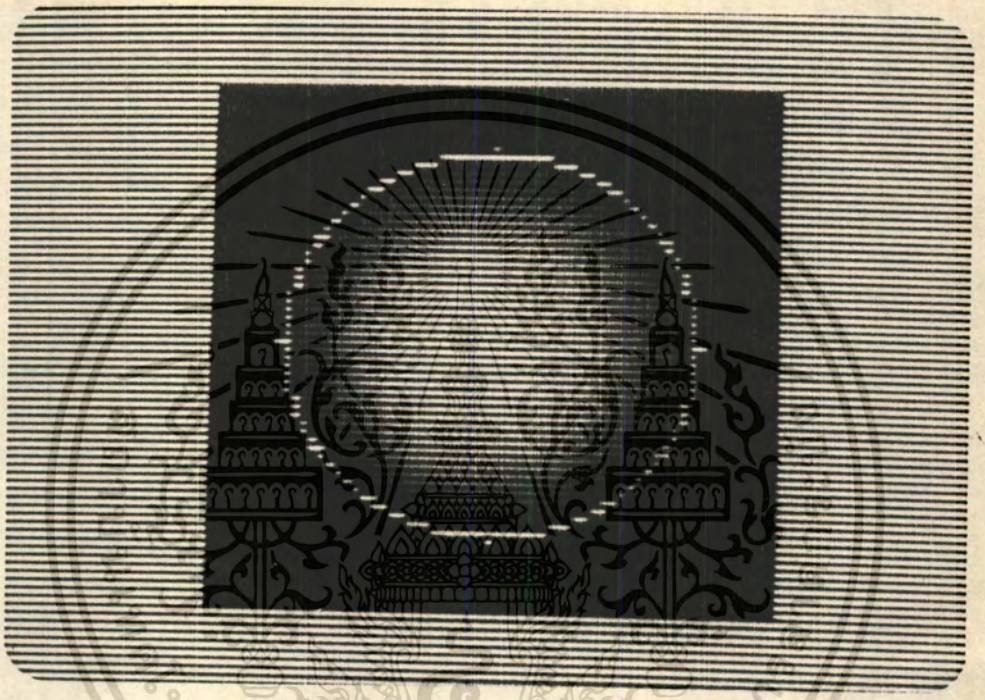
รูป 5.10 แสดงรูปทรงกลมจำลองบนจอมอนิเตอร์ 256 ระดับ

หนดของความสัมพันธ์ดังนี้

$$E(x,y) = I(i,j) = R(f,g)$$

ขั้นตอน 1

จำลองรูปทรงกลม คำนวณหาระดับแสง และโปรเจคลงบนจอมอนิเตอร์ดังรูป 5.10

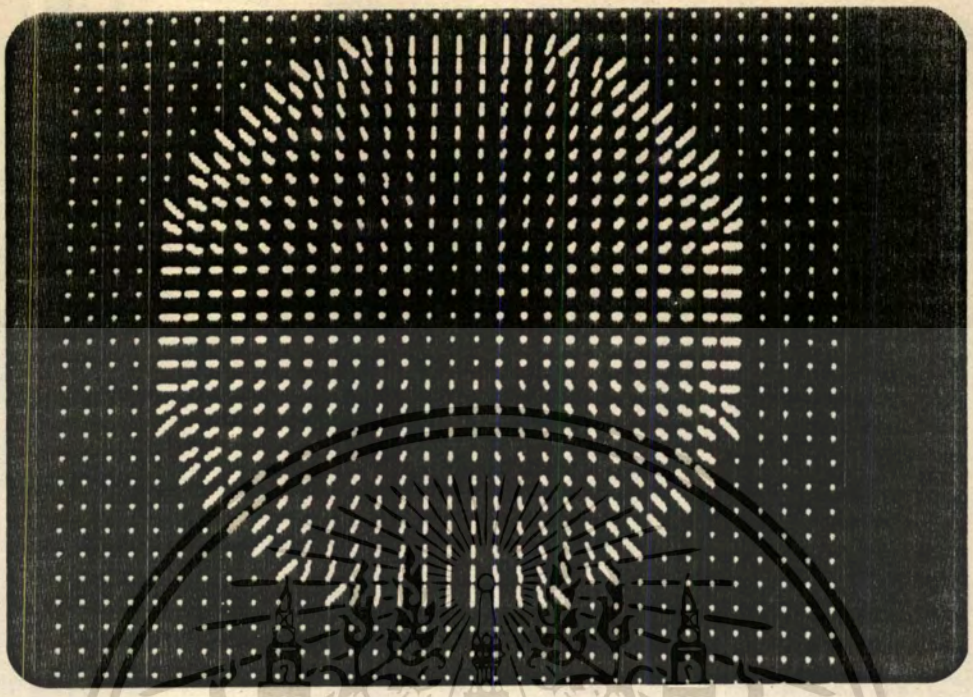


รูป 5.11 แสดงลักษณะพื้นผิวจากบริเวณ occluding boundary

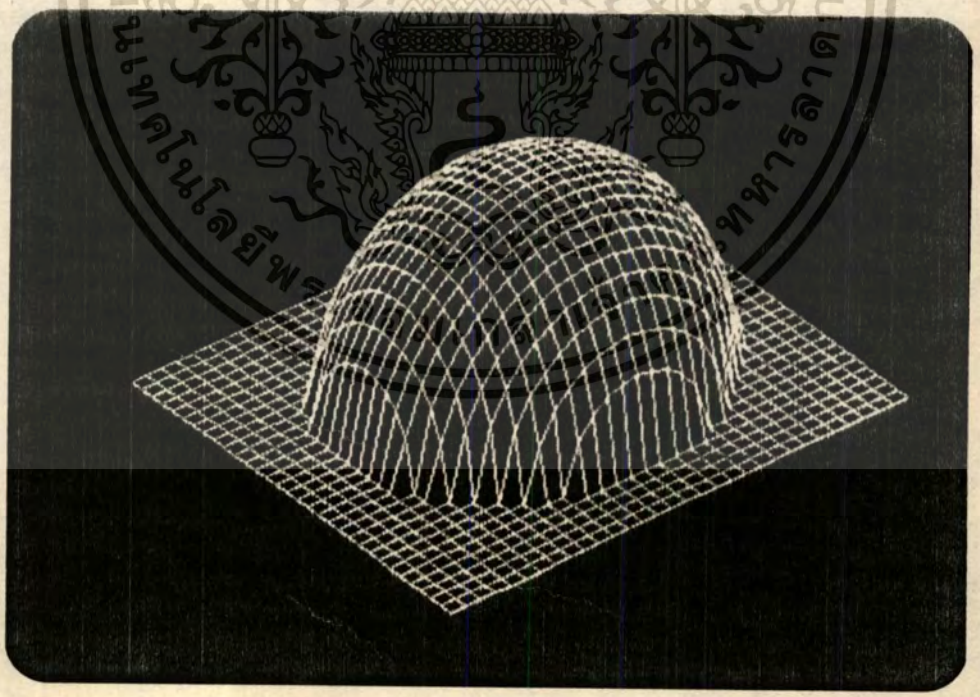
ขั้นตอน 2

นำภาพที่ได้มาตัดขอบโดยวิธี zero crossing กำหนดค่าที่ขอบเป็น occluding boudary ในกรณีนี้มีค่าเป็นมุมตั้งฉากกับพื้นผิว 90 องศา นำไปเป็นค่าเริ่มต้นของ การคำนวณแบบ propagation ตามทฤษฎีบทที่ 4 สมการ 2.24 และ 2.25

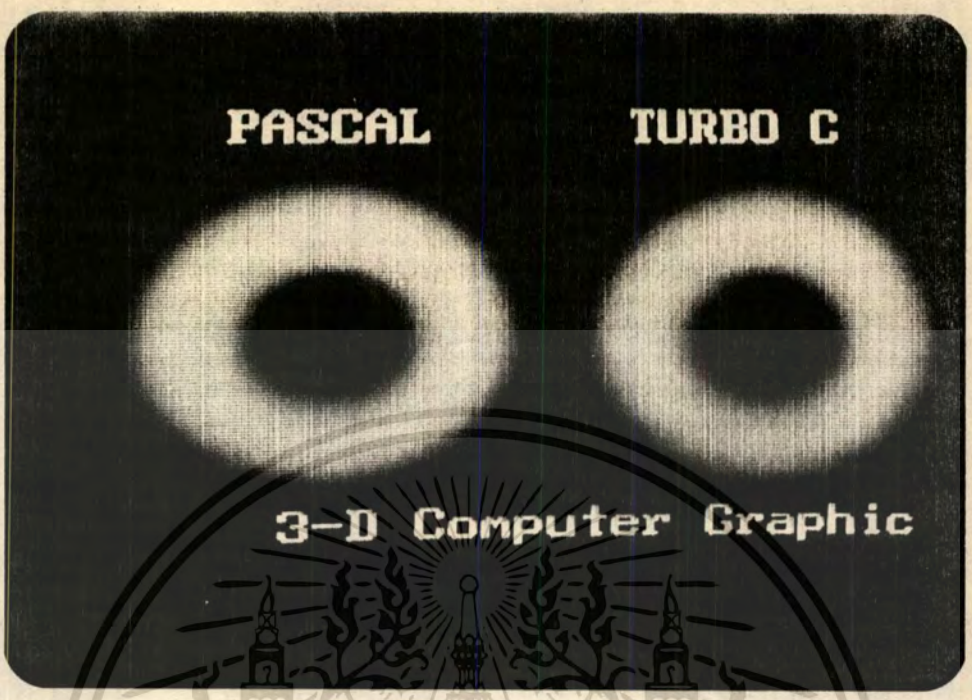
จากการเริ่มต้นด้วยค่าเฉพาะขอบให้มีค่า $(f^2 + g^2)^{1/2} = 2$ และค่าอื่นทุกค่าเป็นศูนย์หมด ผลปรากฏว่าในบริเวณพื้นผิวที่มีมุมสัมผัสมีค่าน้อย ค่าเริ่มต้นซึ่งเป็นค่าเฉลี่ยของ f,g จากจุดภาพโดยรอบมีค่าน้อยมากทำให้ค่าเริ่มต้นในสมการที่ 2.24 และ 2.25 มีค่าเป็นศูนย์ ดังนั้นการคำนวณจึงไม่สามารถให้ผลออกมาได้ หลังจากได้ทดลองโดยการให้ค่าคงที่ใด ๆ เป็นค่าเริ่มต้นเมื่อเกิด



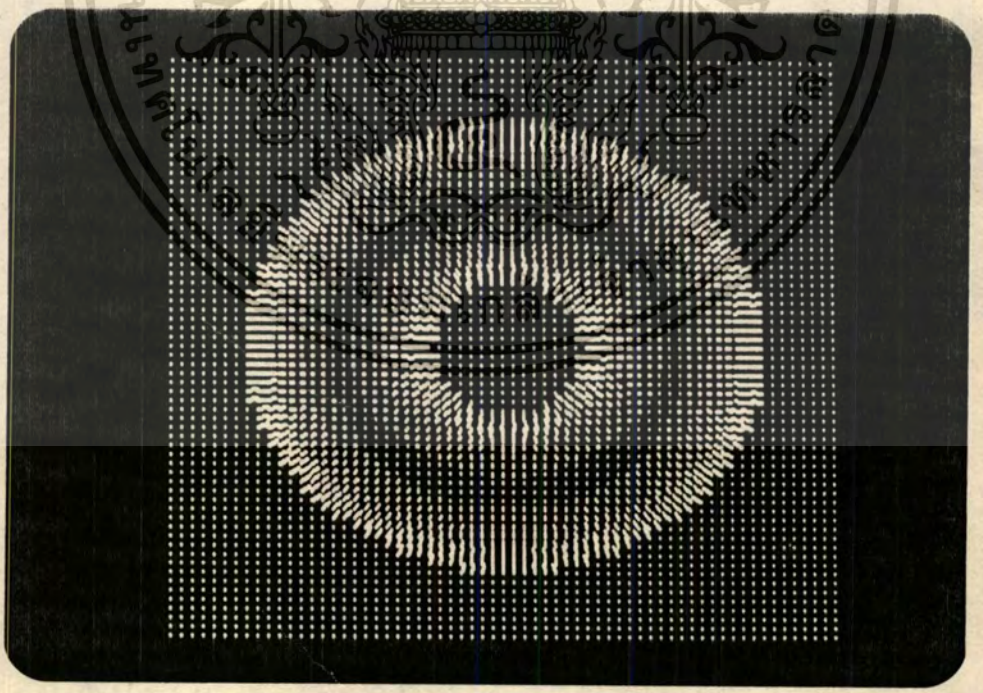
รูป 5.12 แสดงผลของการคำนวณพื้นผิวด้วย needle diagram



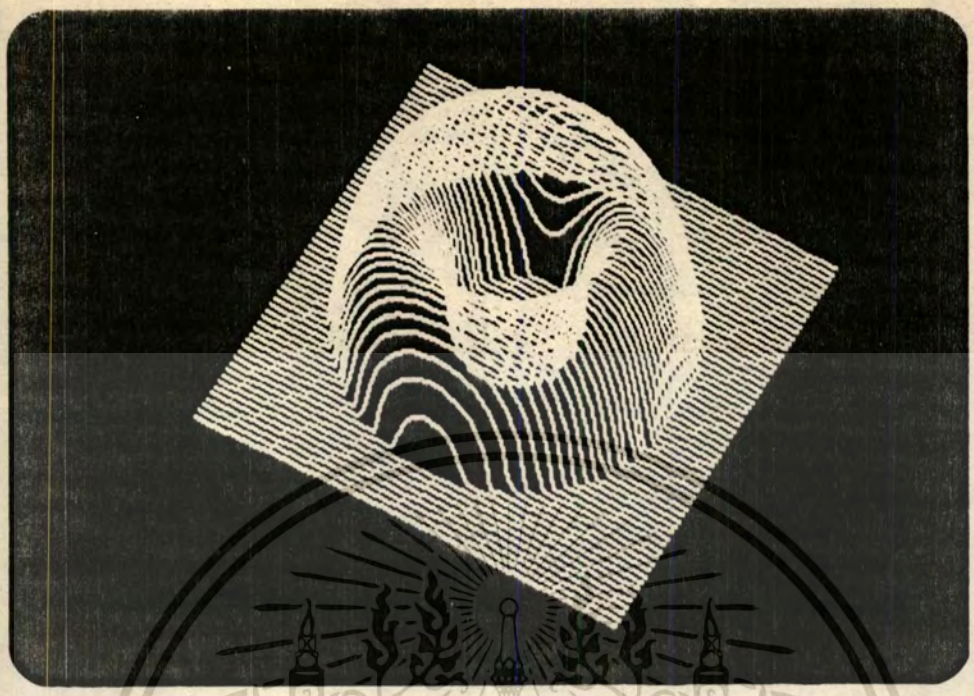
รูป 5.13 แสดงพื้นผิวของวัตถุในระบบแกนมิติ



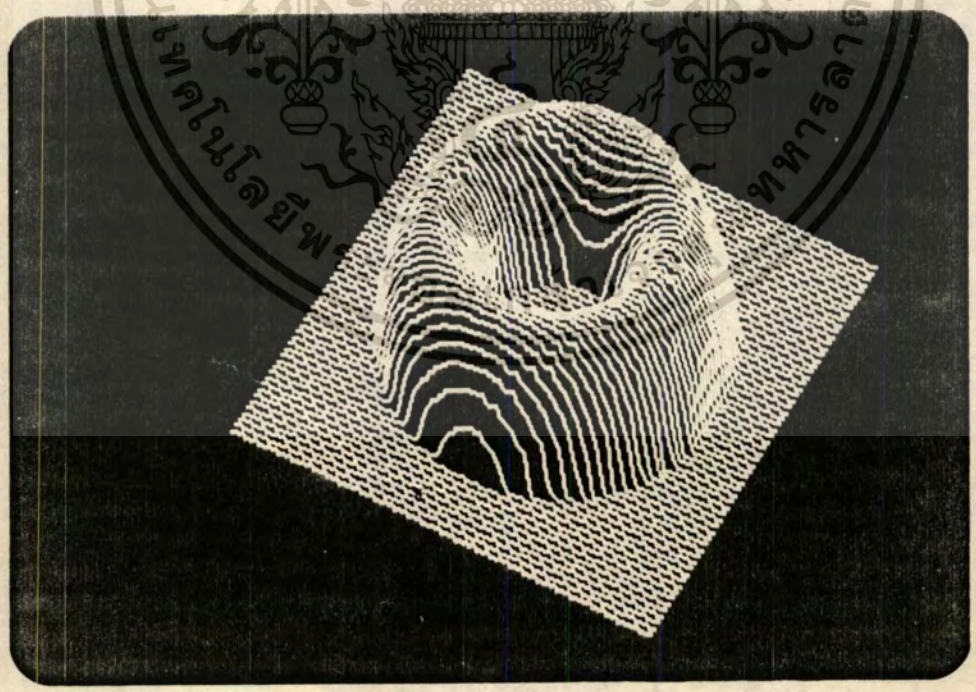
รูป 5.14 ภาพจำลองรูปโดนัท



รูป 5.15 แสดง needle diagram ของรูปโดนัท



รูป 5.16 แสดงพื้นผิวของรูปโดนัทในระบบแกนมิติ



รูป 5.17 แสดงพื้นผิวของรูปโดนัทในระบบแกนมิติ

เงื่อนไขดังกล่าว ผลของการคำนวณจะได้ขนาดของมุมสัมผัสที่ถูกต้องแต่ทิศทางผิดพลาด จึงได้แทนค่าดังกล่าวด้วยค่าความชันของความเข้มแสง (gradient) โดยจัดขนาดให้เหมาะสม

ผลของการคำนวณที่ได้จะเป็นค่า normal vector ของพื้นผิวในแต่ละตำแหน่ง สามารถนำมาเขียนแสดงได้ดังรูป 5.12

ขั้นตอน 3

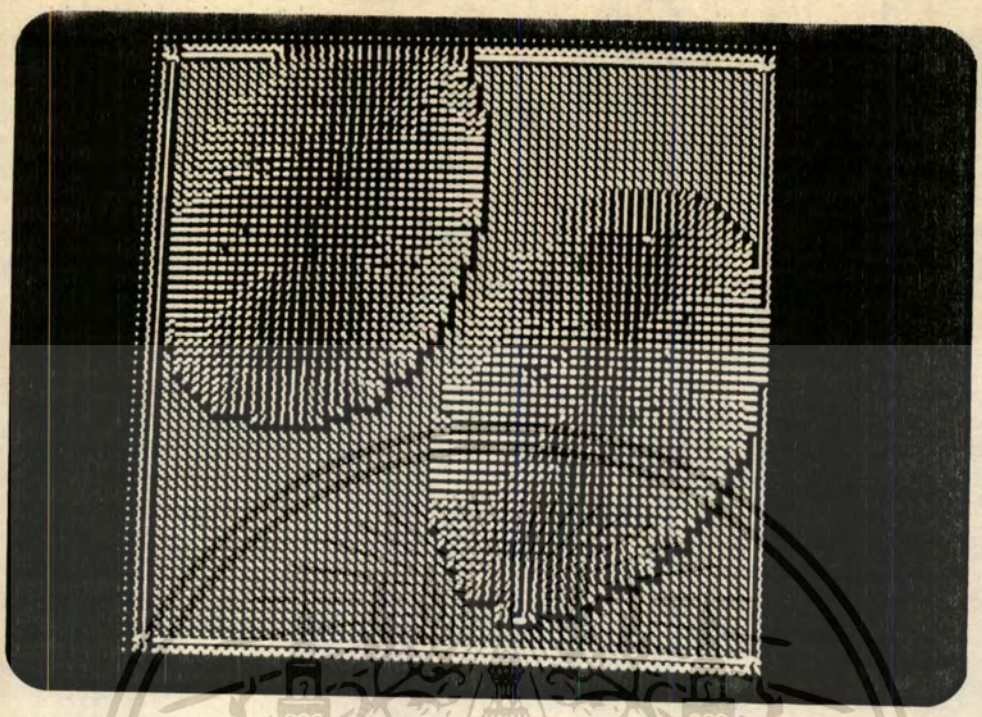
นำผลที่คำนวณได้นั้นมาแสดงผลในแกน 3 มิติ โดยวิธีการของคอมพิวเตอร์กราฟฟิก ดังรูป 5.13

เมื่อทดลองได้ผลดังกล่าว จึงได้ทดลองจำลองภาพวัตถุลักษณะอื่น ๆ ซึ่งอยู่ภายใต้ข้อจำกัดเดิม ดังตัวอย่างรูปที่ 5.14 และรูป 5.15

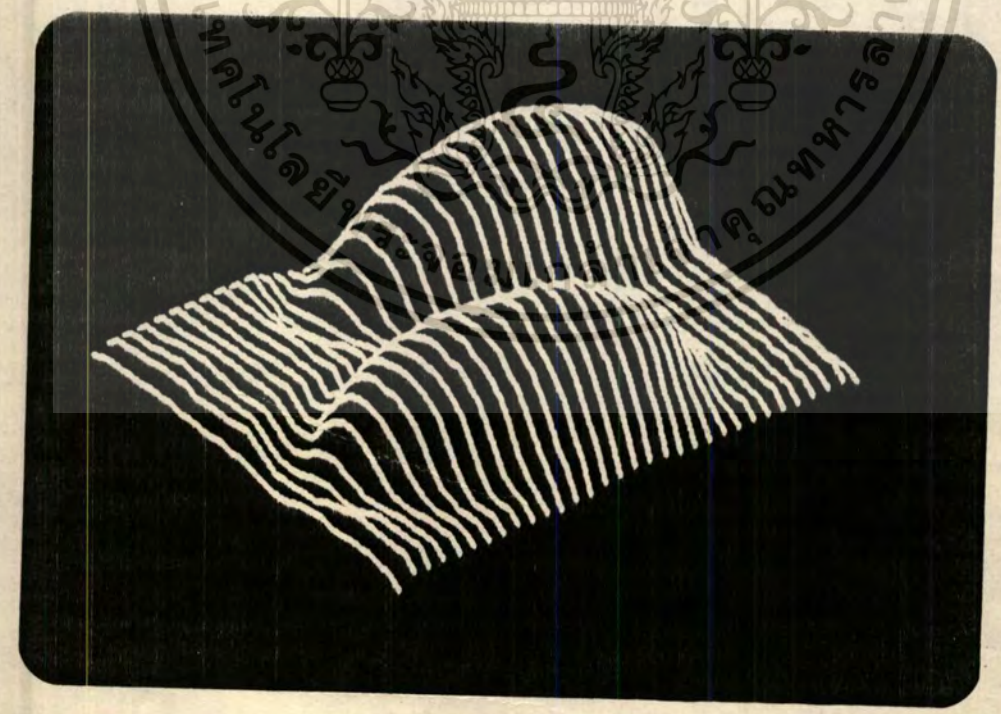
5.3 การหารูปทรง 3 มิติจากภาพจริงตามธรรมชาติ

ไข่ไก่เป็นวัตถุที่นำมาใช้ในทดลอง ถ่ายภาพด้วยกล้อง vidicon เก็บภาพเข้าไฟล์ข้อมูล นำภาพที่ได้มาประมวลผลตามวิธีการเดิม

รูป 5.18 แสดงรูปวัตถุจากกล้องบนจอมอนิเตอร์ 127 ระดับ



รูป 5.19 แสดงผลของการคำนวณพื้นผิวด้วย needle diagram



รูป 5.20 แสดงพื้นผิวของวัตถุในระบบแกนมิติ

ขั้นตอน 1

ภาพวัตถุจากกล้องถ่ายภาพ vidicon บนจอมอนิเตอร์ดังรูป 5.18 จากนั้นจึงได้สุ่มข้อมูลให้ได้ภาพขนาด 64 x 64 ระดับ

ขั้นตอน 2

นำภาพที่ได้มาตัดขอบโดยวิธี zero crossing กำหนดค่าที่ขอบเป็น occluding boudary ในกรณีนี้มีค่าเป็นมุมตั้งฉากกับพื้นผิว 90 องศา นำไปเป็นค่าเริ่มต้นของ การคำนวณแบบ propagation แต่เนื่องจากภาพที่ได้เป็นภาพที่ได้จากภาพตามธรรมชาติ ดังนั้นจึงมีสัญญาณรบกวนซึ่งเกิดการไม่เป็นไปตามข้อจำกัดทางทฤษฎี ดังนั้นจึงต้องทำการกำจัดการรบกวนตามทฤษฎีในบทที่สอง แต่เนื่องจากพื้นของรูปภาพบริเวณที่เป็นฉากมีความเข้มของแสงน้อยมาก จึงสามารถตัดรูปวัตถุออกจากฉากได้โดยง่ายและกำหนดค่าบริเวณฉากให้เป็นศูนย์ ดังนั้นจึงสามารถใช้วิธี zero crossing ตัดขอบของวัตถุได้โดยปรับขนาดของช่องหน้าต่างให้เหมาะสมกับลักษณะความถี่ของรูปภาพ และให้ได้เส้นขอบชัดเจนบริเวณที่ทิศทางแสงสัมผัสกับแสง (occluding boundary) และกำหนดค่าเริ่มต้นด้วยค่าความชันของความเข้มแสงเพื่อให้ค่าผลของการคำนวณมีทิศทางถูกต้อง เช่นเดียวกับการวิธีการคำนวณหาโครงสร้าง 3 มิติจากภาพจำลองวัตถุบนระนาบ 2 มิติ

ผลของการคำนวณที่ได้จะเป็นค่า normal vector ของพื้นผิวในแต่ละตำแหน่ง สามารถนำมาเขียนแสดงได้ดังรูป 5.19

ขั้นตอน 3

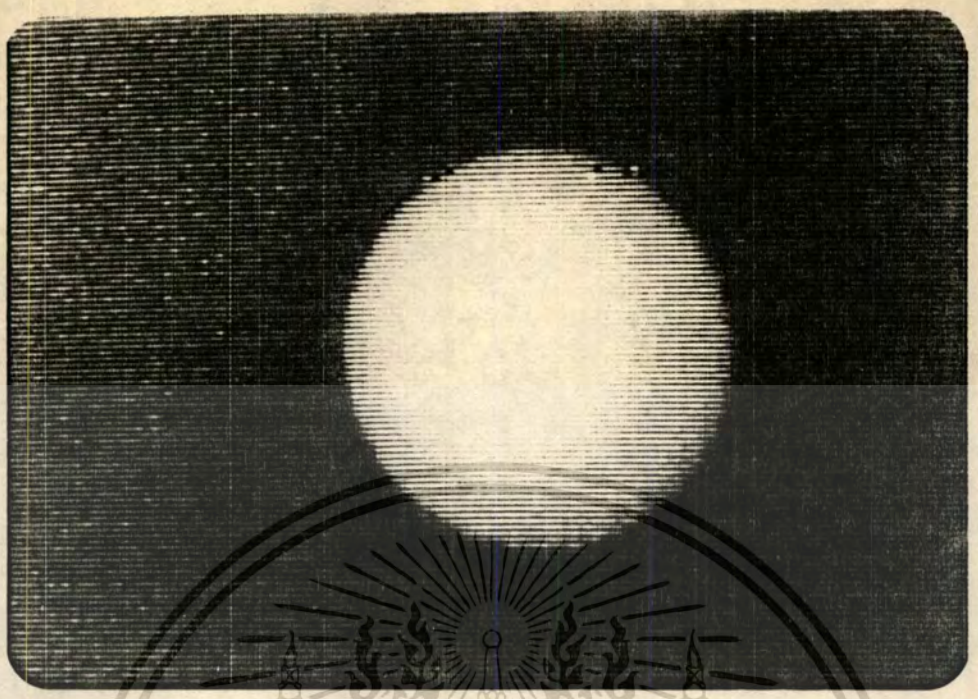
นำผลที่คำนวณได้นั้นมาแสดงผลในแกน 3 มิติ โดยวิธีการของคอมพิวเตอร์กราฟฟิกดังรูป 5.18

จากการทดลองทุกขั้นตอนจะพบว่าการคำนวณใช้เวลามาก ซึ่งขึ้นอยู่กับลักษณะของรูปวัตถุ โดยเฉพาะวัตถุซึ่งมีค่ายอดเนินของพื้นผิวมาก เช่น รูปโดนัท ใช้เวลา 1 ชั่วโมง 25 นาที รูปทรงกลมใช้เวลา 35 นาที (เครื่องคอมพิวเตอร์ IBM PC/AT) และรูปไข่ไก่ซึ่งเป็นรูปภาพจากธรรมชาติใช้เวลา 45 นาที สำหรับค่าผิดพลาดของลักษณะพื้นผิวจากรูปทรงกลม โดยเปรียบเทียบโครงสร้างสามมิติของภาพจำลองกับผลจากการทดลอง ดังรูปที่ 5.21 เป็นค่าความผิดพลาดคิดเป็นเปอร์เซ็นต์ต่อมุมเส้น

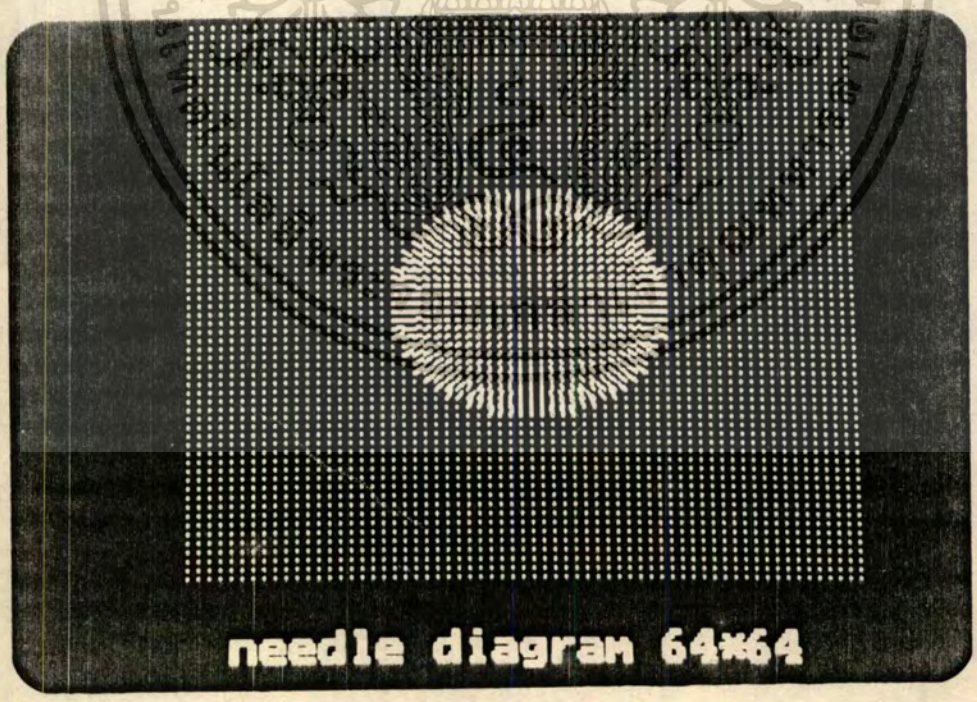
สัมผัสของพื้นระนาบบนพื้นผิววัตถุ ค่าผิดพลาดที่เกิดขึ้นนั้นเกิดจากวิธีการคำนวณด้วยคอมพิวเตอร์เนื่องจากมีการคำนวณที่มีการยกกำลังของค่าโคซายน์และซายน์ซึ่งเป็นค่าจุดทศนิยม เมื่อมีขั้นตอนการการคำนวณหลายขั้นตอนและการคำนวณซ้ำ จากวิธี propagation

รูป 5.21 ค่าผิดพลาดต่อมุมเส้นสัมผัสกับจุดบนพื้นผิววัตถุ

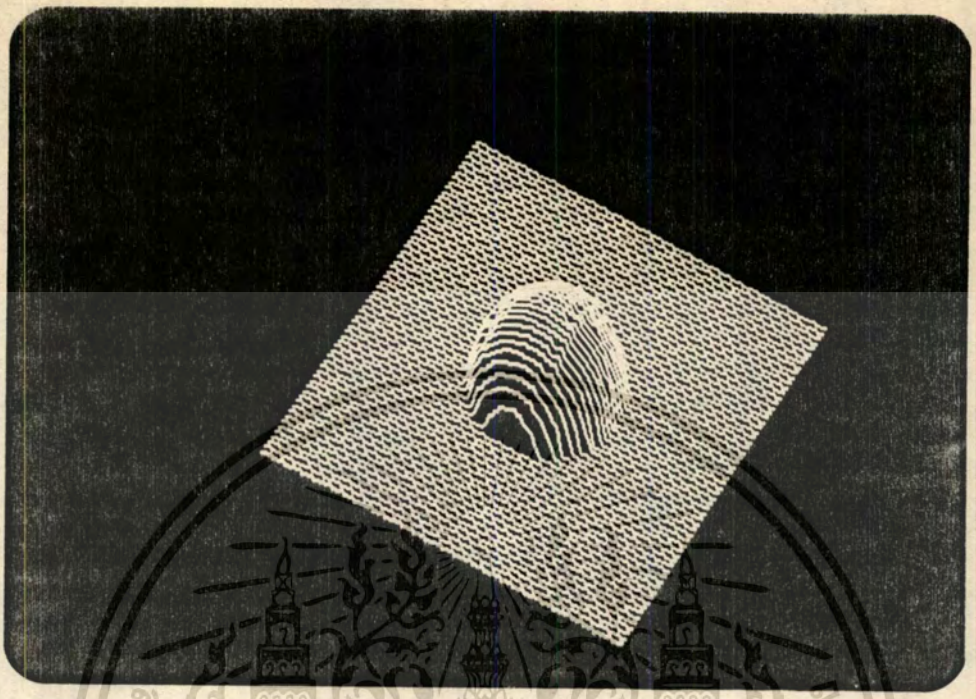
เมื่อทดลองได้ผลดังกล่าว จึงได้ทดลองจำลองภาพวัตถุลักษณะอื่น ๆ ซึ่งอยู่ภายใต้ข้อจำกัดเดิม ดังตัวอย่างรูปลูกปิงปองซึ่งถ่ายจากสภาพแวดล้อมตามธรรมชาติ แสดงผลการคำนวณหาลักษณะพื้นผิวด้วย needle diagram และรูปทรงในระบบแกนมิติ



รูป 5.22 แสดงรูปลูกปิงปองจากกล้องบนจอมอนิเตอร์ 127 ระดับ



รูป 5.23 แสดงผลของการคำนวณพื้นผิวด้วย needle'diagram



รูป 5.24 แสดงพื้นผิวของวัตถุในระบบแกนมิติ

บทที่ 6

บทสรุป

การคำนวณหาภาพ 3 มิติจากรูปภาพ 2 มิติด้วยวิธีการนี้ ยังต้องอยู่ภายใต้ข้อจำกัดทางทฤษฎีหลายอย่าง และผลของการทดลองย่อมขึ้นอยู่กับความถูกต้องของสมมุติฐานในทฤษฎีซึ่งเกี่ยวกับการเกิดความสว่างของแสงบนวัตถุที่ปรากฏบนภาพ

ส่วนวิธีการทดลอง รูปภาพที่นำมาถ่ายด้วยกล้องวิดิทัศน์นั้น เป็นรูปภาพซึ่งเกิดในสิ่งแวดล้อมตามธรรมชาติ ย่อมไม่สามารถทำให้เป็นไปตามข้อจำกัดทางทฤษฎี อย่างไรก็ดีผลที่ได้จากการทดลองนับว่าเป็นที่น่าพอใจ ค่าความผิดพลาดที่เกิดจากวิธีการทางตัวเลขขึ้นอยู่กับความสามารถของเครื่องคอมพิวเตอร์ที่ใช้ รวมทั้งสาเหตุจากขนาดจำนวนจุดของภาพและค่าหน่วยความจำที่มี ความเร็วในการประมวลผลนั้นขึ้นอยู่กับขนาดของภาพและลักษณะของโครงสร้างของวัตถุ ทั้งค่าความผิดพลาดและความเร็วในการประมวลผลนั้น อาจลดลงได้ถ้าการกำหนดค่าเริ่มต้นมีค่าของขนาดและทิศทางใกล้เคียงกับความเป็นจริงของพื้นผิววัตถุของแต่ละตำแหน่ง ในการทดลองนี้ ใช้ค่าความชันของสัญญาณแสงที่มีขนาดพอเหมาะมาใช้เป็นค่าเริ่มต้นซึ่งช่วยให้ทิศทางของพื้นผิวสามารถประมวลผลได้อย่างถูกต้อง

สิ่งที่น่าทำการศึกษาต่อไปก็คือ ถ้าลักษณะพื้นผิวมีลักษณะต่างไปจากการทดลองนี้ เช่น ถ้ารูปวัตถุมีพื้นผิวชันขึ้นสู่ยอดและขณะที่กลับลาดต่ำลงมานั้น ก่อนที่จะลาดลงมาสู่ค่าต่ำสุด ได้ย้อนกลับขึ้นไปใหม่ดังลักษณะของหลุมภูเขาไฟ ลักษณะพื้นผิวเช่นนี้ย่อมอยู่ในข้อจำกัดของพื้นผิวต่อเนื่องแต่การใช้ค่าเริ่มต้นระหว่างรอยต่อของการย้อนกลับนั้นควรจะเป็นเช่นใด สิ่งที่น่าสนใจอีกประการก็คือการหาวิธีการที่จะจดจำวัตถุ 3 มิติ ซึ่งจะต้องมีวิธีการที่ยุ่งยากซับซ้อนแตกต่างไปจากวิธีการในวัตถุ 2 มิติ การใช้ข้อมูลจากแสง เงา และลายเส้นขอบในลักษณะ perspective และอื่น ๆ ย่อมเป็นแนวทางในการวิจัยต่อไป

บรรณานุกรม

1. Ikuechi,K.,"Shape from Regular Patterns", Artificial,Intelligence , Vol. 22,pp.49-75,1984.
2. Witkin, A.P.,"Recovering surface shape and oraintation from texture", Artificial Intelligence , Vol.17,pp.17-45,1981.
3. Ikuechi,K. Horn.B.K.P.,"Numerical shape from shading and occluding boundaries",Artificial Intelligence , Vol.17,pp.141-181,1981.
4. Lowe, D.G.,"Three-Dimention Object Recognition from Single Two-Dimentional Images",Artificial Intelligence ,Vol.31,pp.355-395, 1987.
5. Steven,K.A.,"The visual interpretation of surface contours",Artificial Intelligence,Vol.17,pp.47-73,1981.
6. Barnard, S.T.,"Interpreting perspective images",Artificial Intelligence,Vol.21,pp.435-462,1983.
7. Brooks,R.A.,"Symbolic reasoning among 3-D model and 3-D images", Artificial Intelligence,Vol.17,pp.285-348,1981.
8. Kanade,T.,"Recovery of the Three-Dimensional Shap of an Object from a Single View",Artificial Intelligence,Vol.17,pp.409-460,1981.
9. H.G. Barraw,J.M. Tenenbaum,"Interpreting Line Drawings as Three-Dimensional Surface",Atificial Intelligence,Vol.17,pp.75-166,1981.
10. Terzopoulos,D.,"The Computation of Visible-Surface Representations", IEEE Trans Pattern Anal. Machine Intell.,Vol. 10,pp.417-437,1988.
11. Frankot,R.T.,"A Method for Enforcing Integrability in Shape from

Shading Algorithms",IEEE Trans Pattern Anal. Machine Intell.,Vol. 10, pp.439-451,1988.

12. Gouraud, H.,"Computer Display of Curve Surfaces",IEEE Trans.C-20, pp.623-628,1971.

13. ชาญชัย นิสิทธิวิทยานนท์,"การติดตามขอบวัตถุโดยใช้ตารางขอบหน้าต่าง", บทความทางวิชาการ Seminar MI, คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าลาดกระบัง,พฤศจิกายน 2529.

14. Chan S.Park,"Interactive Microcomputer Graphics",Addision-Wesley Publishing Company,Inc.,1985.

15. Steven Harrington,"COMPUTER GRAPHICS A Programing Approach", McGraw-Hill,Inc.,1983.

16. Roy A. Plastock,Gordon Kalley,"COMPUTER GRAPHICS",McGraw-Hill,Inc., 1986.

17. Yoshiaki Shirai,"Three Dimentional Computer Vision",Springer-Verlag Berlin Heidellberge,1987.

18. Levine,Martin D.,"VISION IN MAN AND MACHINE" , McGraw-Hill,Inc.,1985.

19. G.Springer,"International Series in Pure and Applied Mathematics", McGraw-Hill,Inc.,1980.

20. David F. Roger,"PROCEDURAL ELEMENTS FOR COMPUTER GRAPHICS", McGraw-Hill,Inc.,1985.

21. Grimson,William Eric Leifur,"FROM IMAGES TO SURFACES",The Massachusetts Institute of Technology,1981.

22. Richard O. Duda,Peter E. Hart,"PATTERN CLASSIFICATION AND SCENE ANALYSIS",John Wiley & Sons,Inc.,1973.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สามารถสำเร็จได้ด้วยดี ก็เนื่องด้วยได้รับความกรุณาให้คำปรึกษาและแนะนำจาก รองศาสตราจารย์ มนัส สังวรศิลป์ และ รองศาสตราจารย์ ดร.สมเกียรติ ศุภเดช ตลอดจนอาจารย์ เจ้าหน้าที่ และนักศึกษาปริญญาโท แผนกชีวะอีเล็กทรอนิกส์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง ทุก ๆ ท่านที่ให้การสนับสนุนเป็นอย่างดี โดยเฉพาะอย่างยิ่งอาจารย์ประภากร สุวรรณะ ได้ช่วยถ่ายภาพผลการทดลองงานวิทยานิพนธ์ ตั้งแต่เริ่มต้นการทดลองจวบจนกระทั่งผลงานทั้งหมดสำเร็จลุล่วงไป รวมทั้งคณะอาจารย์แผนกไฟฟ้ากำลัง สถาบันเทคโนโลยีราชมงคล วิทยาเขตเทคนิคภาคตะวันออกเฉียงเหนือ นครราชสีมา ที่กรุณาให้ความอนุเคราะห์อุปกรณ์เครื่องพิมพ์และร่วมมือช่วยเหลือด้วยความเต็มใจ จึงใคร่ขอขอบพระคุณท่านผู้ได้กล่าวนามมา ณ ที่นี้

ผู้จัดทำ

ภาคผนวก

โปรแกรมที่ใช้ทดลอง

```
  1: program trgr;
  2: uses tsort,image2;
  3: procedure loopy;
  4: var h,sm,lmax,q,a :integer;
  5: begin
  6:    norli;       {*** simulate picture ***}
  7:    wl;          {*** save picture ***}
  8:    rl;          {*** read image for a procedure ***}
  9:    dimage;      {*** load image to digitizer buffer ***}
 10:    cle9;        {*** clear memory seg 9000}
 11:    smooth;
 12:    i9tol;       {*** load memory to array ***}
 13:    zeroc;       {*** cal window zerocrossing ***}
 14:    cle9;
 15:    ezero;       {*** find occluding boundary ***}
 16:    i9tol;
 17:    dl84ge;      {*** linear data for ploting ***}
 18:    writeln('..... min zero crossing value .....');
 19:    readln(q);  {*** select threshold ***}
 20:    pt(q,0);     {*** plote freq of data ***}
 21: { zeroc circle = q  which  plane = 0 on ..light[i,j]..}
 22: { integer image in linear ..8000:0.. }
 23: { zero crossing from ezeroc(true)  save in linear ..8400:0..}
 24:    writeln('***** gradient of image *****');
 25:    rl;          {*** read image ***}
 26:    gxy;
 27:    w2gfile;   {*** save gradient ***}
 28:    iteration;
 29:    w2ffile;   {*** save normal plane vecter ***}
 30:    readln(a);
 31:    if a <>10 then loopy;
 32: end;
 33: begin
 34:    loopy;
 35: end.
```

```
 1: program demon;
 2: uses tsort,image2;
 3: procedure loopy;
 4: var h,sm,lmax,q,a :integer;
 5: begin
 6:    norli;
 7:    zeroc;
 8:    lmax:=200;                          {cal window zerocrossing}
 9:    rlze;
10:    writeln('..... min zero crossing value .....');
11:    readln(q);
12:    pt(q,0);
13: { zeroc circle = 10  which  plane = 0 on ..light[i,j]..}
14: { integer image in linear ..8000:0.. }
15: { zero crossing from ezeroc(true)  save in linear ..8400:0..}
16:      writeln('***** gradient of image *****');
17:     igslope;
18:     ifslope;
19:     writeln ('***** read plane normal vecter *****');
20:     r2ffile; {read gy,gx}
21:     tpny;
22:     tpny1;
23:     writeln('***** read 3-D contour of image *****');
24:     tangent;
25:     wrefile;
26:     readln;
27:     r1;
28:     writeln(***** read 3 dimension  surface *****);
29:     r64file;
30:     v3d1;
31:     readln(a);
32:     if a <>10 then loopy;
33: end;
34: begin
35:    loopy;
36: end.
```

Listing of b:tsort.pas
Page No. 1

```
 1:
 2: {            Copyright (c) 1985, 87 by Borland International, Inc.
 3:
 4: unit tsort;
 5: interface
 6: {$R-}
 7: {$S-}
 8: uses Crt,graph;
 9:
10: const max =16383;
11: var
12:   v : word;
13:   i : integer;
14: procedure isort(var b,smax:integer);{*** qsort seg 9400 4k ***}
15: procedure m9pt(var j:integer);  {*** plot data seg 9000 4k ***}
16: procedure w2ffile;  {*** write f,g to fx.dta & fy.dta ***}
17: procedure r2ffile;  {*** read f,g from fx.dta & fy.dta ***}
18: procedure w2gfile;  {*** write gradient to gx.dta & gy.dta ***}
19: procedure r2gfile;  {*** read gradient from gx.dta & gy.dta ***}
20: procedure igslope;  {***** needle of gradient *****}
21: procedure ifslope;  {***** needle of normal plane *****}
22: procedure tpnx;     {***** needle 128 from memory *****}
23: procedure tpny;  {***** needle 64 from mem *****}
24: procedure tpny1; {***** needle 32 from memory *****}
25:
26: implementation
27:
28: procedure quicksort(  Lo,Hi: integer);
29: procedure sort(l,r: integer);
30: var
31:   i,j,x,y,s: integer;
32: begin
33:   v:=0;
34:   i:=l; j:=r; x:=mem[$9400 :v+((l+r) DIV 2)];
35:   repeat
36:      while mem[$9400 :v+i]<x do i:=i+1;
37:      while x<mem[$9400:v+j] do j:=j-1;
38:      if i<=j then
39:      begin
40:        y:=mem[$9400:v+i];
```

```
 41:          mem[$9400:v+i]:=mem[$9400:v+j];
 42:          mem[$9400:v+j]:=y;
 43:          i:=i+1; j:=j-1;
 44:        end;
 45:    until i>j;
 46:    if l<j then sort(l,j);
 47:    if i<r then sort(i,r);
 48: end;
 49:
 50: begin {quicksort};
 51:   sort(Lo,Hi);
 52: end;
 53: procedure m9pt(var j:integer);  {***** plote freq of light *****}
 54: var
 55:   Gd, Gm : integer;
 56:   s,k,b,l,s1,s2 : integer;
 57:   v : word;
 58: const Clipon =true;
 59: begin
 60:   writeln('***** plote freq of data *****');
 61:   Gd := Detect;
 62:   InitGraph(Gd, Gm, '');
 63:   if GraphResult <> grOk then
 64:     Halt(1);
 65:     v :=0;
 66:     setlinestyle(dottedln,0,normwidth);
 67:     line(45,270,45,10);
 68:     line(45,270,715,270);
 69:     setlinestyle(solidln,0,normwidth);
 70:     for i:= 1 to 20 do
 71:         begin
 72:           line(46,270-i*10,44,270-i*10);
 73:         end;
 74:     for i:= 1 to 10 do
 75:         begin
 76:           line(47,270-i*20,43,270-i*20);
 77:         end;
 78:     for i:=1 to 30 do
 79:       begin
 80:          line (45+i*20,271,45+i*20,269);
```

```
  81:         end;
  82:       for i:=1 to 15 do
  83:         begin
  84:           line (45+i*40,272,45+i*40,268);
  85:         end;
  86:       SetTextStyle(DefaultFont, horizDir,2);
  87:       OutTextXY(200, 50, 'frequency of data');
  88:       SetTextStyle(DefaultFont, horizDir,1);
  89:       OutTextXY(260, 80, '  scale  1 : 20  ');
  90:       SetTextStyle(DefaultFont, vertDir,1);
  91:       OutTextXY(25, 100, 'frequency of data ');
  92:       SetTextStyle(DefaultFont, horizDir,1);
  93:       OutTextXY(305, 285, 'qsort data ');
  94:       k:=1;b:=0;
  95:       while j > b  do
  96:         begin
  97:           s1:=45+2*(mem[$9000:v+k]);
  98:           s2:=270-(mem[$9000:v+k+1]);
  99:           line(s1,s2,s1,270);
 100:           k:=k+2; inc(b);
 101:         end;
 102:       delay(1000);readln;
 103:       CloseGraph;
 104: end;
 105: procedure isort(var b,smax:integer);
 106: var  v       : word;
 107:      i,s,j,k : integer;
 108:      ch      : char;
 109: begin {qsort}
 110:   v:=0;
 111:   Writeln('***** Now sorting  0-16384  from $9400:00 *****');
 112:   quicksort(0,max);
 113:   writeln('***** list data [Y/N] ***** ');
 114:   smax:=mem[$9400:v+max];
 115:   writeln(smax);
 116:   ch :=Readkey;
 117:   if ch = 'y' then
 118:      begin
 119:        writeln('location : data');
 120:        delay(1000);
```

```
121:         i:=0;
122:         while i <= max do
123:            begin
124:               write(i:8,' :');
125:                 for j:=1 to 16 do
126:                    begin
127:                       Write(mem[$9400:v+i]:4);
128:                         i:=i+1;
129:                    end;
130:                    writeln;
131:            end;
132:      end;
133:   Write('***** Now count level 0-16384 from $9400:00 *****');
134:   v:=0;
135:   s:=mem[$9400 : v];
136:   j:=1; k:=0; b:=0;
137:   for i:=1 to max+1 do
138:      begin
139:      k:=k+1;
140:      if i= max+1 then
141:         begin
142:            mem[$9000 : v+j]:=s;
143:            mem[$9000 :v+j+1]:=k;
144:            b:=b+1;
145:         end;
146:      if mem[$9400 :v+i] >s then
147:         begin
148:            mem[$9000 : v+j] := s;
149:            s:=mem[$9400 : v+i];
150:            j:=j+1; mem[$9000 : v+j]:=k;
151:            j:=j+1;b:=b+1;
152:            k:=0;
153:         end;
154:      end;
155:   writeln(b);
156:   i:=0;
157:   writeln('***** list level data [Y/N] ******');
158:   ch:=readkey;
159:      if ch = 'y' then
160:      begin
```

```
161:          writeln ('[ order ]     ¦ data ¦ frequency ¦');
162:          delay(1000);
163:          j:=0;
164:          while j < b do
165:           begin
166:             write('[',j:3,']');
167:             k:=wherey;
168:             gotoxy(15,k);
169:             for s:=0 to 4 do
170:              begin
171:               i:=i+1;
172:               write(' ¦',mem[$9000:v+i]:4,mem[$9000:v+i+1]:4);
173:               i:=i+1;j:=j+1;
174:              end;
175:            writeln(' ¦');
176:            delay(200);
177:           end;
178:       end;
179:       delay(1000);
180:       clrscr;
181: end;
182: procedure w2ffile;{*** write f,g to fx.dta & fy.dta ***}
183: type  rfile = file of integer;
184: var
185:       w : word;
186:       i,j,t1,t2 : integer;
187:       infd1,infd2 : word;
188:       gxfile : rfile;
189:       gyfile : rfile;
190: begin
191:       assign(gxfile,'b:fx.dta');
192:       assign(gyfile,'b:fy.dta');
193:       rewrite(gxfile);
194:       rewrite(gyfile);
195:       w:=0;
196:       for i := 0 to 127 do
197:          begin
198:          for j := 0 to 127 do
199:            begin
200:               infd1 := memw[$8800:w];
```

```
201:                infd2 := memw[$9000:w];
202:                t1:=infd1;t2:=infd2;
203:                write(gxfile ,t1);
204:                write(gyfile ,t2);
205:                w:=w+2;
206:              end;
207:           end;
208:       close(gxfile); close(gyfile);
209: end;
210: procedure r2ffile;{*** read f,g from fx.dta & fy.dta ***}
211: var
212:       infd1,infd2 : word;
213:       i,j,t1,t2 : integer;
214:       code : integer;
215:       gxfile : file of integer;
216:       gyfile : file of integer;
217:       w : word;
218: begin
219:       assign(gxfile,'b:fx.dta');
220:       assign(gyfile,'b:fy.dta');
221:       reset(gxfile);
222:       reset(gyfile);
223:       w:=0;
224:       for i := 0 to 127 do
225:          begin
226:          for j := 0 to 127  do
227:                begin
228:                read(gxfile ,t1);
229:                read(gyfile ,t2);
230:                infd1:=t1;infd2:=t2;
231:                memw[$8800:w]:=infd1;
232:                memw[$9000:w]:=infd2;
233:                w:=w+2;
234:                end;
235:          end;
236:       close(gxfile); close(gyfile);
237: end;
238: procedure w2gfile;
239: type  rfile = file of integer;
240: var
```

```
241:       w : word;
242:       i,j,t1,t2 : integer;
243:       infd1,infd2 : word;
244:       gxfile : rfile;
245:       gyfile : rfile;
246: begin
247:       assign(gxfile,'b:gx.dta');
248:       assign(gyfile,'b:gy.dta');
249:       rewrite(gxfile);
250:       rewrite(gyfile);
251:       w:=0;
252:       for i := 0 to 127 do
253:          begin
254:          for j := 0 to 127 do
255:                begin
256:                infd1 := memw[$8800:w];
257:                infd2 := memw[$9000:w];
258:                t1:=infd1;t2:=infd2;
259:                write(gxfile ,t1);
260:                write(gyfile ,t2);
261:                w:=w+2;
262:                end;
263:          end;
264:      close(gxfile); close(gyfile);
265: end;
266: procedure r2gfile;
267: var
268:       infd1,infd2 : word;
269:       i,j,t1,t2 : integer;
270:       code : integer;
271:       gxfile : file of integer;
272:       gyfile : file of integer;
273:       w : word;
274: begin
275:       assign(gxfile,'b:gx.dta');
276:       assign(gyfile,'b:gy.dta');
277:       reset(gxfile);
278:       reset(gyfile);
279:       w:=0;
280:       for i := 0 to 127 do
```

```
281:          begin
282:          for j := 0 to 127  do
283:               begin
284:               read(gxfile ,t1);
285:               read(gyfile ,t2);
286:               infd1:=t1;infd2:=t2;
287:               memw[$8800:w]:=infd1;
288:               memw[$9000:w]:=infd2;
289:               w:=w+2;
290:               end;
291:          end;
292:       close(gxfile); close(gyfile);
293: end;
294: procedure ifslope;  {***** needle *****}
295: var
296:   Gd, Gm : integer;
297:   s,x,y : integer;
298:   v : word;
299:   infd1,infd2 : integer;
300:   i,j,t1,t2 : integer;
301:   tx,ty:real;
302:   code : integer;
303:   gxfile : file of integer;
304:   gyfile : file of integer;
305:   w : word;
306: const Clipon =true;
307: begin
308:   writeln('***** 0 plote zero crossing *****');
309:   Gd := Detect;
310:   InitGraph(Gd, Gm, '');
311:   if GraphResult <> grOk then
312:     Halt(1);
313:     settextstyle(0;0,2);
314:     moveto(180,290);
315:     outtext('normal plane vector');
316:     assign(gxfile,'b:fx.dta');
317:     assign(gyfile,'b:fy.dta');
318:     reset(gxfile);
319:     reset(gyfile);
320:     w:=0;     v := 0;
```

```
321:      for i:= 0 to 127 do
322:      begin
323:          for j := 0 to 127 do
324:          begin
325:               read(gxfile ,t1);
326:               read(gyfile ,t2);
327:               if t1 >=3000 then t1:=t1-5000;
328:               if t2 >=3000 then t2:=t2-5000;
329:               tx := t1/250;ty := t2/500;
330:               infd1:=round(tx);infd2:=round(ty);
331:               x:=3*(j)+150; y:=2*(i+20);
332:               line(x,y,x+infd1,y+infd2);
333:               w:=w+2;
334:
335:          end;
336:      end;
337:      delay(4000);
338:       close(gxfile); close(gyfile);
339:      readln;
340:      CloseGraph;
341: end;
342: procedure igslope;  {***** needle *****}
343: var
344:   Gd, Gm : integer;
345:   s,x,y : integer;
346:   v : word;
347:   infd1,infd2 : integer;
348:   i,j,t1,t2 : integer;
349:   tx,ty:real;
350:   code : integer;
351:   gxfile : file of integer;
352:   gyfile : file of integer;
353:   w : word;
354: const Clipon =true;
355: begin
356:   writeln('***** 0 plote zero crossing *****');
357:   Gd := Detect;
358:   InitGraph(Gd, Gm, '');
359:   if GraphResult <> grOk then
360:     Halt(1);
```

```
361:      settextstyle(0,0,2);
362:      moveto(180,290);
363:      outtext('gradient of surface');
364:      assign(gxfile,'b:gx.dta');
365:      assign(gyfile,'b:gy.dta');
366:      reset(gxfile);
367:      reset(gyfile);
368:      w:=0;     v := 0;
369:      for i:= 0 to 127 do
370:      begin
371:          for j := 0 to 127 do
372:          begin
373:                read(gxfile ,t1);
374:                read(gyfile ,t2);
375:                if t1 >=3000 then t1:=t1-5000;
376:                if t2 >=3000 then t2:=t2-5000;
377:                tx := t1/250;ty := t2/500;
378:                infd1:=round(tx);infd2:=round(ty);
379:                x:=3*(j)+150; y:=2*(i+20);
380:                line(x,y,x+infd1,y+infd2);
381:                w:=w+2;
382:
383:          end;
384:      end;
385:      delay(4000);
386:       close(gxfile); close(gyfile);
387:      readln;
388:      CloseGraph;
389: end;
390: procedure tpnx;  {***** needle 128 from memory *****}
391: var
392:    Gd, Gm : integer;
393:    s,x,y : integer;
394:    v : word;
395:    infd1,infd2 : integer;
396:    i,j,t1,t2 : integer;
397:    tx,ty:real;
398:    code : integer;
399:    gxfile : file of integer;
400:    gyfile : file of integer;
```

```
401:    w : word;
402: const Clipon =true;
403: begin
404:    writeln('***** 0 plote zero crossing *****');
405:    Gd := Detect;
406:    InitGraph(Gd, Gm, '');
407:    if GraphResult <> grOk then
408:      Halt(1);
409:      settextstyle(0,0,2);
410:      moveto(180,290);
411:      outtext('needle diagram 128*128');
412:      w:=0;
413:      for i:= 0 to 127 do
414:      begin
415:          for j := 0 to 127 do
416:          begin
417:              t1 := memw[$8800:w];
418:              t2 := memw[$9000:w];
419:              if t1 >=3000 then t1:=t1-5000;
420:              if t2 >=3000 then t2:=t2-5000;
421:              tx :=t1/1000;ty := t2/1000;
422:              infd1:=round(3*tx);infd2:=round(2*ty);
423:              x:=3*(j)+150; y:=2*(i);
424:              line(x,y,x+infd1,y+infd2);
425:             w:=w+2;
426:          end;
427:      end;
428:      delay(4000);readln;
429:      CloseGraph;
430: end;
431: procedure tpny;  {***** needle 64 from mem *****}
432: var
433:    Gd, Gm : integer;
434:    s,x,y : integer;
435:    v : word;
436:    infd1,infd2 : integer;
437:    i,j,t1,t2 : integer;
438:    tx,ty:real;
439:    code : integer;
440:    gxfile : file of integer;
```

```
441:    gyfile : file of integer;
442:    w : word;
443:  const Clipon =true;
444:  begin
445:    writeln('***** 0 plote zero crossing *****');
446:    Gd := Detect;
447:    InitGraph(Gd, Gm, '');
448:    if GraphResult <> grOk then
449:      Halt(1);
450:      settextstyle(0,0,2);
451:      moveto(180,280);
452:      outtext('needle diagram 64*64');
453:      w:=0;
454:      for i:= 1 to 64 do
455:      begin
456:          for j := 1 to 64 do
457:          begin
458:              t1 := memw[$8800:w];
459:              t2 := memw[$9000:w];
460:              if t1 >=3000 then t1:=t1-5000;
461:              if t2 >=3000 then t2:=t2-5000;
462:              tx :=t1/1000;ty := t2/1000;
463:              infd1:=round(3*tx);infd2:=round(2*ty);
464:              x:=6*(j)+150; y:=4*(i);
465:              line(x,y,x+infd1,y+infd2);
466:              w:=w+4;
467:          end;
468:        w:=w+2*128;
469:      end;
470:      delay(4000);readln;
471:      CloseGraph;
472:  end;
473:  procedure tpny1;  {***** needle 32 from memory *****}
474:  var
475:    Gd, Gm : integer;
476:    s,x,y : integer;
477:    v : word;
478:    infd1,infd2 : integer;
479:    i,j,t1,t2 : integer;
480:    tx,ty:real;
```

```
481:    code : integer;
482:    gxfile : file of integer;
483:    gyfile : file of integer;
484:    w : word;
485: const Clipon =true;
486: begin
487:    writeln('***** 0 plote zero crossing *****');
488:    Gd := Detect;
489:    InitGraph(Gd, Gm, '');
490:    if GraphResult <> grOk then
491:      Halt(1);
492:      settextstyle(0,0,2);
493:      moveto(180,280);
494:      outtext('needle diagram 32*32');
495:      w:=0;
496:      for i:= 1 to 32 do
497:      begin
498:          for j := 1 to 32 do
499:          begin
500:              t1 := memw[$8800:w];
501:              t2 := memw[$9000:w];
502:              if t1 >=3000 then t1:=t1-5000;
503:              if t2 >=3000 then t2:=t2-5000;
504:              tx :=t1/1000;ty := t2/1000;
505:              infd1:=round(3*tx);infd2:=round(2*ty);
506:              x:=12*(j)+150; y:=8*(i);
507:              line(x,y,x+infd1,y+infd2);
508:              w:=w+8;
509:          end;
510:        w:=w+2*3*128;
511:      end;
512:      delay(4000);readln;
513:      CloseGraph;
514: end;
515:
516: begin
517: end.
```

```
41: procedure rrefile;   {*****   read gradient data   *****}
42: procedure wlze;  {***** write zercrossing.dta to light[i,j] ***** }
43: procedure rlze;  {***** read zercrossing.dta to light[i,j] ***** }
44: procedure contour;   {*****   plote contour *****}
45: procedure tangent;   {***** needle *****}
46: procedure error;     {*****  plote error *****}
47: procedure v3d; {*****  hidden line process   *****}
48:
49: implementation
50: procedure  norli;  { *****simulate picture vary light***** }
51: var
52:     rdot,md,md1,zz1,md2,sz,k: real;
53:     xx,yy,zz : integer;
54:     lx,ly,lz,r,s1 : integer;
55: begin
56:     writeln( '*****simulate picture vary light*****' );
57:     r:=50;
58:     s1:=64;
59:     writeln ('enter vecter of light [lx ly lz]');
60:     readln (lx,ly,lz);
61:     for i:= 0 to 127 do
62:     begin
63:         for j := 0 to 127 do
64:         begin
65:             xx:=j-s1;
66:             yy:=s1-i;
67:             sz:=(r*r)-(xx*xx+yy*yy);
68:             if sz <= 0 then
69:               light[i,j]:=0 else
70:               begin
71:                 zz1:=(sqrt(sz));
72:                 if sz = 0 then zz1 :=0;
73:                 zz:=round(zz1);
74:                 rdot:=(lx*xx+ly*yy+lz*zz);
75:                 if rdot < 0 then
76:                   light[i,j] :=0 else
77:                   begin
78:                     md1:=sqrt(lx*lx+ly*ly+lz*lz);
79:                     md2:=sqrt(xx*xx+yy*yy+zz*zz);
80:                     md:=170*rdot/(md1*md2);
```

```
 81:                          light [i,j] := round(md);
 82:
 83:                      end;
 84:                  end;
 85:
 86:          end;
 87:
 88:      end;
 89: end;
 90: procedure  norlil;  { *****simulate picture vary light***** }
 91: var
 92:      rdot,md,md1,zz1,md2,sz: real;
 93:      xx,yy,zz,k : integer;
 94:      lx,ly,lz,r,s1 : integer;
 95: begin
 96:      writeln( '*****simulate picture vary light*****' );
 97:      r:=30;
 98:      s1:=64;
 99:      writeln ('enter vecter of light [lx ly lz]');
100:      readln (lx,ly,lz);
101:      for i:= 0 to 127 do
102:      begin
103:          for j := 0 to 127 do
104:          begin
105:              k:=light[i,j];
106:              xx:=j-s1;
107:              yy:=s1-i;
108:              sz:=(r*r)-(xx*xx+yy*yy);
109:              if sz <= 0 then
110:                light[i,j]:=k else
111:                begin
112:                   zz1:=(sqrt(sz));
113:                   if sz = 0 then zz1 :=0;
114:                   zz:=round(zz1);
115:                   rdot:=(lx*xx+ly*yy+lz*zz);
116:                   if rdot < 0 then
117:                     light[i,j] :=0 else
118:                     begin
119:                        md1:=sqrt(lx*lx+ly*ly+lz*lz);
120:                        md2:=sqrt(xx*xx+yy*yy+zz*zz);
```

```
121:                          md:=80*rdot/(md1*md2);
122:                          light [i,j] :=k+ round(md);
123:                      end;
124:                  end;
125:              end;
126:          end;
127: end;
128:
129: procedure  dimage; { *****put l[i,j] to memory seg 8000***** }
130: var
131:       s : integer;
132:       v : word;
133: begin
134:     writeln( '*****put l[i,j] to memory seg 8000***** ' );
135:     writeln ('use memory nonlinear seg 8000:00');
136:     v := 64*256+64;
137:     for i:= 0 to 127 do
138:     begin
139:         for j := 0 to 127 do
140:         begin
141:             Mem[$8000 : v] := light[i,j];
142:             v:= v + 1;
143:         end;
144:     v := v + 128;
145:     end;
146: end;
147: procedure  uimage; { ***** pull memory seg 8000 to light[i,j] ***** }
148: var
149:       s : integer;
150:       v : word;
151: begin
152:     writeln( '***** pull memory seg 8000 to light[i,j] ***** ' );
153:     v := 64*256+64;
154:     for i:= 0 to 127 do
155:     begin
156:         for j := 0 to 127 do
157:         begin
158:             light[i,j] := Mem[$8000 : v] ;
159:             v:= v + 1;
160:         end;
```

```
161:      v := v + 128;
162:      end;
163: end;
164: procedure  dl8age; { *****put l[i,j] to memory seg 8000***** }
165: var
166:       s : integer;
167:       v : word;
168: begin
169:      writeln( '*****put l[i,j] to memory seg 8000***** ' );
170:      writeln ('use memory linear seg 8000:00');
171:      v := 0;
172:      for i:= 0 to 127 do
173:      begin
174:          for j := 0 to 127 do
175:          begin
176:              Mem[$8000 : v] := light[i,j];
177:              v:= v + 1;
178:          end;
179:      end;
180: end;
181: procedure  ul8age; { ***** pull memory seg 8000 to light[i,j] ***** }
182: var
183:       s : integer;
184:       v : word;
185: begin
186:      writeln( '***** pull linear memory seg 8000 to light[i,j] ***** '
187:      v := 0;
188:      for i:= 0 to 127 do
189:      begin
190:          for j := 0 to 127 do
191:          begin
192:              light[i,j] := Mem[$8000 : v] ;
193:              v:= v + 1;
194:          end;
195:      end;
196: end;
197: procedure  194age; { ***** linear l[i,j] to memory seg 9400:00 ***** }
198: var
199:       s : integer;
200:       v : word;
```

```
201: begin
202:     writeln( '***** linear l[i,j] to memory seg 9400:00 *****');
203:     v := 0;
204:     for i:= 0 to 127 do
205:     begin
206:         for j := 0 to 127 do
207:         begin
208:             Mem[$9400 : v] := light[i,j];
209:             v:= v + 1;
210:         end;
211:     end;
212: end;
213: procedure  dl84ge; { *****put l[i,j] to memory seg 8400***** }
214: var
215:       s : integer;
216:       v : word;
217: begin
218:     writeln( '*****put l[i,j] to memory seg 8400***** ' );
219:     writeln ('use memory linear seg 8000:00');
220:     v := 0;
221:     for i:= 0 to 127 do
222:     begin
223:         for j := 0 to 127 do
224:         begin
225:             Mem[$8400 : v] := light[i,j];
226:             v:= v + 1;
227:         end;
228:     end;
229: end;
230: procedure  ul84ge; { ***** pull memory seg 8400 to light[i,j] ***** }
231: var
232:       s : integer;
233:       v : word;
234: begin
235:     writeln( '***** pull linear memory seg 8400 to light[i,j] ***** '
236:     v := 0;
237:     for i:= 0 to 127 do
238:     begin
239:         for j := 0 to 127 do
240:         begin
```

```
241:                light[i,j] := Mem[$8400 : v] ;
242:                v:= v + 1;
243:            end;
244:        end;
245: end;
246: procedure  ltoi9;     { *****put l[i,j] to mem 9000***** }
247: var
248:        s : integer;
249:        v : word;
250: begin
251:      writeln( '*****put l[i,j] to mem 9000*****');
252:      v := 64*256+64;
253:      for i:= 0 to 127 do
254:      begin
255:          for j := 0 to 127 do
256:          begin
257:              Mem[$9000 : v] := light[i,j];
258:              v:= v + 1;
259:          end;
260:      v := v + 128;
261:      end;
262: end;
263: procedure  cle9;     { *****clear 1 page memory at 9000***** }
264: var
265:        s : integer;
266:        v : word;
267:
268: begin
269:      writeln( '***** clear page memory at 9000*****');
270:      v := 0;
271:      for i:= 0 to 255 do
272:      begin
273:          for j := 0 to 255 do
274:          begin
275:              Mem[$9000 : v] :=0;
276:              v:= v + 1;
277:          end;
278:      end;
279: end;
280: procedure i9tol;  { *****load mem 9000 to l[i,j]***** }
```

```
281: var
282:       s : integer;
283:       v : word;
284: begin
285:     writeln( '*****load mem 9000 to l[i,j]*****');
286:     v := 64*256+64 ;
287:     for i:= 0 to 127 do
288:     begin
289:         for j := 0 to 127 do
290:         begin
291:             light[i,j]:= Mem[$9000 : v];
292:             v:= v + 1;
293:         end;
294:     v := v + 128;
295:     end;
296: end;
297: procedure smooth;  { ***** everage put on seg 9000 ***** }
298: var
299:       s1,s2,s3 : real;
300:       s : integer;
301:       v : word;
302: begin
303:     writeln( '***** everage put on nonlinear seg 9000 *****');
304:     v := 64*256+64;
305:     for i:= 1 to 126 do
306:     begin
307:         for j := 1 to 126 do
308:         begin
309:           if i=0 then  light[i-1,j]:=light[i,j]
310:           else
311:              begin
312:                 if i = 127 then light[i+1,j]:=light[i,j]
313:                  else
314:                    begin
315:                       if j = 0 then  light[i,j-1]:=light[i,j]
316:                        else
317:                           begin
318:                             if j=127 then light[i,j+1] := light[i,j]
319:                             else
320:                               begin
```

```
                                        s1:=light[i+1,j]+light[i-1,j];
                                        s2:=light[i,j+1]+light[i,j-1];
                                        s3:=0.25*(s1+s2);
                                        Mem[$9000 : v]:=round(s3);
                                    end;
                                end;
                            end;
                        end;
                        v:= v + 1;
                    end;
        v := v + 126;
        end;
        v := 64*256+64;
        for i:= 1 to 126 do
            begin
                for j := 1 to 126 do
                begin
                    light[i,j]:= Mem[$9000 : v];
                    v:= v + 1;
                end;
                v := v + 126;
            end;
end;
procedure iedge;    { ***** edge detect put on seg 9000 *****}
var
        s1,s2,s3,s4,s5,s6,s7 : real;
        s : integer;
        v : word;
begin
        writeln( '***** edge detect put on seg 9000 *****');
        v := 64*256+64;
        for i:= 1 to 126 do
        begin
            for j := 1 to 126 do
            begin
                if i=0 then  light[i-1,j]:=light[i,j]
                else
                    begin
                        if i = 127 then light[i+1,j]:=light[i,j]
                        else
```

```
361:                    begin
362:                       if j = 0 then  light[i,j-1]:=light[i,j]
363:                        else
364:                           begin
365:                             if j=127 then light[i,j+1] := light[i,j]
366:                             else
367:                               begin
368:                                 s1:=light[i+1,j]+light[i-1,j];
369:                                 s2:=light[i,j+1]+light[i,j-1];
370:                                 s3:=light[i+1,j+1]+light[i+1,j-1];
371:                                 s4:=light[i-1,j+1]+light[i-1,j-1];
372:                                 s5:=8*light[i,j];
373:                                 s6:=s5-(s1+s2+s3+s4);
374:                                 s7:=abs(s6);
375:                                 if s7 >=38 then s7:=255 ;
376:                                 if s7 < 38 then s7:=0;
377:                                 Mem[$9000 : v]:=round(s7);
378:                               end;
379:                           end;
380:                    end;
381:                 end;
382:                 v:= v + 1;
383:           end;
384:      v := v + 126;
385:      end;
386:      v := 64*256+64;
387:      for i:= 1 to 126 do
388:         begin
389:            for j := 1 to 126 do
390:            begin
391:              light[i,j]:= Mem[$9000 : v];
392:              v:= v + 1;
393:            end;
394:            v := v + 126;
395:         end;
396: end;
397: procedure zeroc;
398: var i,j,k : integer;
399:     r,s,t1,t2,l : real;
400:     t,t3,r1,r2 : real;
```

```
401: begin
402:      l:=0.4 ;
403:      for i:= 1 to 7 do
404:        begin
405:          for j:= 1 to 7 do
406:            begin
407:            s:=(i*0.5-2)*(i*0.5-2)+(j*0.5-2)*(j*0.5-2);
408:            t1:=(s-2*l)/(l*l);
409:            t2:=-(s/(2*l));
410:            t3:=exp(t2);
411:            t :=t1*t3;
412:            k :=round(t);
413:            zc[i,j] := k;
414:            end;
415:        end;
416:      for i:= 1 to 7 do
417:        begin
418:          for j:= 1 to 7 do
419:          begin
420:            k:=zc[i,j];
421:            write(k:3);
422:          end;
423:          writeln;
424:        end;
425: end;
426: procedure ezero;   { ***** edge zero crossing put on seg 9000 *****}
427: var
428:       s1,s2,s : real;
429:       i1,j1,i2,j2 : integer;
430:       v : word;
431: begin
432:     writeln( '***** light[i,j] edge zero crossing put on seg 9000 *****'
433:     writeln ('use nonlinear $9000:00and put data in light[i,j]');
434:     v := 64*256+64;
435:     for i:=3 to 124 do
436:     begin
437:         for j := 3 to 124 do
438:         begin
439:                 s:=0.0;
440:                 s2:=0.0;
```

```
                    for il:=-3 to 3  do
                      begin
                        for jl:=-3 to 3 do
                          begin
                            s2:=light[i+il,j+jl]*zc[il+4,jl+4];
                            s:=s2+s;
                            s1:=Abs(s);
                          end;
                      end;
                    if s1>=255 then
                            begin
                              s1:=255.0;
                            end;
                    Mem[$9000 : v] :=round(s1);
                    v:=v+1;
            end;
      v := v + 122
      end;
      v := 64*256+64;
      for i:= 3 to 124 do
          begin
            for j := 3 to 124 do
            begin
              light[i,j]:= Mem[$9000 : v];
              v:= v + 1;
            end;
            v := v + 122;
          end;
end;
procedure pt(p,q:integer);  {***** plote zero crossing *****}
var
   Gd, Gm : integer;
   s,k : integer;
   v : word;
const Clipon =true;
begin
   writeln('***** plote zero crossing *****');
   writeln('..... 0 to cover.....');
   k:=0;
   Gd := Detect;
```

```
481:    InitGraph(Gd, Gm, '');
482:    if GraphResult <> grOk then
483:      Halt(1);
484:      settextstyle(0,0,2);
485:      moveto(350,40);
486:      outtext('zero crossing');
487:      v := 64*256+64;
488:      for i:= 0 to 127 do
489:      begin
490:          for j := 0 to 127 do
491:          begin
492:              s := light[i,j];
493:              if s>=p then
494:              begin
495:                putpixel(3*j+100,2*(i+20),2);
496:                putpixel(j+500,i+150,2);
497:              end;
498:          end;
499:      v := v + 128;
500:      end;
501:      delay(500);
502:      if k=0 then
503:      begin
504:          v := 64*256+64;
505:          for i:= 0 to 127 do
506:          begin
507:              for j := 0 to 127 do
508:              begin
509:                  s := light[i,j];
510:                  if s=q then
511:                  begin
512:                    putpixel(3*j+100,2*(i+20),2);
513:                    putpixel(j+500,i+150,2);
514:                  end;
515:              end;
516:          v := v + 128;
517:          end;
518:      end;
519:      delay(2000);
520:      CloseGraph;
```

```
521: end;
522: procedure tpt(q:integer);  {***** 0  plote zero crossing *****}
523: var
524:   Gd, Gm : integer;
525:   s : integer;
526:   v : word;
527: const Clipon =true;
528: begin
529:   writeln('***** 0 plote zero crossing *****');
530:   Gd := Detect;
531:   InitGraph(Gd, Gm, '');
532:   if GraphResult <> grOk then
533:     Halt(1);
534:     settextstyle(0,0,2);
535:     moveto(350,80);
536:     outtext('zero crossing');
537:     v := 64*256+64;
538:     for i:= 0 to 127 do
539:     begin
540:         for j := 0 to 127 do
541:         begin
542:             s := light[i,j];
543:             if s=q then
544:             begin
545:               putpixel(3*j+100,2*i,2);
546:               putpixel(j+500,i+150,2);
547:             end;
548:         end;
549:     v := v + 128;
550:     end;
551:     delay(2000);
552:     CloseGraph;
553: end;
554: procedure wl;{*****  write  light[i,j] to  file newfile.dta  *****}
555: var
556:      i,j : integer;
557:      infd : integer;
558:      outfile : file of integer;
559: begin
560:      writeln('***** write newfile.dta to light[i,j] *****');
```

```
561:       assign(outfile ,'b:newfi.dta');
562:       rewrite(outfile);
563:       for i := 0 to 127 do
564:          begin
565:          for j := 0 to 127 do
566:               begin
567:               infd := light[i,j];
568:               write(outfile ,infd);
569:               end;
570:          end;
571:       close(outfile);
572: end;
573: procedure w1ze;{*****  write  light[i,j] to  file zercr.dta  *****}
574: var
575:       i,j : integer;
576:       infd : integer;
577:       outfile : file of integer;
578: begin
579:       writeln('***** write zercr.dta to light[i,j] *****');
580:       assign(outfile ,'b:zercr.dta');
581:       rewrite(outfile);
582:       for i := 0 to 127 do
583:          begin
584:          for j := 0 to 127 do
585:               begin
586:               infd := light[i,j];
587:               write(outfile ,infd);
588:               end;
589:          end;
590:       close(outfile);
591: end;
592: procedure r1; {***** read newfile.dta to light[i,j] ***** }
593: var
594:       infd,i,j,k : integer;
595:       infile : file of integer;
596: begin
597:       writeln('***** read newfile.dta to light[i,j] *****');
598:       assign(infile ,'b:newfi.dta');
599:       reset(infile);
600:       for i := 0 to 127 do
```

```
601:          begin
602:          for j := 0 to 127  do
603:               begin
604:               read(infile ,infd);
605:               if infd < 10 then  infd:=0;
606:               light[i,j] := infd;
607:               if i=64 then
608:                    begin
609:                         k:=light[i,j];
610:                         rel[j]:=round(k);
611:                         write(rel[j]:10);
612:                    end;
613:               end;
614:          end;
615:       close(infile);
616: end;
617: procedure rize; {***** read zercr.dta to light[i,j] ***** }
618: var
619:       infd,i,j : integer;
620:       infile : file of integer;
621: begin
622:       writeln('***** zercr.dta to light[i,j] *****');
623:       assign(infile ,'b:zercr.dta');
624:       reset(infile);
625:       for i := 0 to 127 do
626:          begin
627:          for j := 0 to 127  do
628:               begin
629:               read(infile ,infd);
630:               light[i,j] := infd;
631:               end;
632:          end;
633:       close(infile);
634: end;
635: procedure setzec; {*****  set threshold  of scene   *****}
636: var i,j,s :integer;
637: begin
638:   for i:= 0 to 127 do
639:     begin
640:       for j:= 0 to 127 do
```

```
641:          begin
642:             s:=light[i,j];
643:             if s > 0 then  light[i,j]:=10;
644:          end;
645:      end;
646:     tpt(0);
647: end;
648: procedure gxy; {*****   initial gradint for iteration   *****}
649: var
650:       i,j : integer;
651:       a,b,c,d,ia,ib,ic,id ,s :integer;
652:       ax,ay,bx,by,iax,iay,ibx,iby : integer;
653:       gx1,igx1,gx,igx,gy1,igy1,gy,igy : integer;
654:       gx2,igx2,gy2,igy2 : real;
655:       infd,iinfd : real ;
656:       v,w : word;
657: begin
658:       writeln('use 8800 fo gx,9000 for gy');
659:       v:=0;w:=0;
660:       for i := 0 to 127 do
661:          begin
662:          for j := 0 to 127 do
663:             begin
664:                s:=mem[$8400:w];
665:                if s > 0 then
666:                   begin
667:                      ax:=2*light[i,j+5];bx:=2*light[i,j-5];
668:                      ay:=2*light[i+5,j];by:=2*light[i-5,j];
669:                      a:=light[i+5,j-5]; b :=light[i+5,j+5];
670:                      c:=light[i-5,j-5]; d :=light[i-5,j+5];
671:                      gx1 := ((d+ax+b)-(c+bx+a));
672:                      gy1 := ((a+ay+b)-(c+by+d));
673:                      infd := 0.1 + abs(gx1)+abs(gy1);
674:                      gx2 := 2000*(gx1/infd);
675:                      gy2 := 2000*(gy1/infd);
676:                      gx := round(gx2);
677:                      gy := round(gy2);
678:                      iax:=2*light[i,j+1];ibx:=2*light[i,j-1];
679:                      iay:=2*light[i+1,j];iby:=2*light[i-1,j];
680:                      ia:=light[i+1,j-1]; ib :=light[i+1,j+1];
```

```
                        ic:=light[i-1,j-1]; id :=light[i-1,j+1];
                        igx1 := ((id+iax+ib)-(ic+ibx+ia));
                        igy1 := ((ia+iay+ib)-(ic+iby+id));
                        iinfd := 0.1 + abs(igx1)+abs(igy1);
                        igx2 := 2000*(igx1/iinfd);
                        igy2 := 2000*(igy1/iinfd);
                        igx:=round(igx2); igy:=round(igy2);
                        if igx<0 then igx:=igx+5000;
                        if igy<0 then igy:=igy+5000;

                        if gx = 0 then gx :=igx;
                        if gy = 0 then gy :=igy;

                        if gx<0 then gx:=gx+5000;
                        if gy<0 then gy:=gy+5000;
                        memw[$8800:v] :=gx;
                        memw[$9000:v] :=gy;
                      end
                    else
                      begin
                        memw[$8800:v] := 0;
                        memw[$9000:v] := 0;
                      end;
                    v:=v+2; w:=w+1;
                end;
                write('w');
            end;
end;
procedure iteration(var lmax : integer);
var i,j,a,b,qc : integer;
    tp1,tp2,a2,b2 : integer;
    v,w : word;
    ze : byte;
    c,c1,c2,c3,d,d1,t1 : real;
    f,g,a1,b1,a3,ex,ey  : real;
begin
    v:=0;w:=0;
    for i:=0 to 127 do
      begin
        for j:= 0 to 127 do
```

```
721:           begin
722:              ze := mem[$8400:v];
723:              if ze > 0 then
724:                begin
725:                  a:=memw[$8800:w];b:=memw[$9000:w];
726:                  if a>=3000 then a:=a-5000;
727:                  if b>=3000 then b:=b-5000;
728:                  a1:=a/1000; b1:=b/1000;
729:                  d1 := 10;
730:                  while d1>0.1 do
731:                    begin
732:                                t1:=a1*a1+b1*b1;
733:                                c:=(t1)+4;c1:=8/c;
734:                                c2:=c*c; c3:=c1-1;
735:                                d:=light[i,j]-c3*lmax;
736:                                d1:=abs(d);
737:                                a3:=16/c2;
738:                                ex:=(-a1)*a3;
739:                                ey:=(-b1)*a3;
740:                                f:=a1+1*(d*ex)/lmax;
741:                                g:=b1+1*(d*ey)/lmax;
742:                                a1:=f;b1:=g;
743:                     end;  {while}
744:
745:                  tp1:=round(1000*a1);tp2:=round(1000*b1);
746:                  if tp1<0 then tp1:=tp1+5000;
747:                  if tp2<0 then tp2:=tp2+5000;
748:                  a2:=tp1;b2:=tp2;
749:                  memw[$8800:w]:=a2;
750:                  memw[$9000:w]:=b2;
751:                end;  {zero}
752:               v:=v+1;w:=w+2;write('j');
753:           end;
754:          writeln('w',i);
755:        end;
756: end;
757: procedure v3d1; {*** computer graphic non hidden line process ***}
758: type initia = array[0..640] of integer;
759: var x,y,i,j : integer;
760:     th1,th2 : real;
```

```
761:      ymin,ymax : initia;
762:      flag,flag1,flag2 : integer;
763:      oldx,oldy,xs,ys,xsi,ysi,deltax,deltay   : integer;
764:      tempx,tempy,newx:integer;
765:      slope : integer;
766:      tempy2,tempx2 : boolean;
767:    Gd, Gm : integer;
768:  const Clipon =true;
769:  begin
770:    Gd := Detect;
771:    InitGraph(Gd, Gm, '');
772:    if GraphResult <> grOk then
773:      Halt(1);
774:      th1:=3.14159/4;th2:=3.14159/3;
775:      for i:=63 downto 2 do
776:        begin
777:          for j := 63 downto 1 do
778:            begin
779:              xs:=xe[i,j];xsi:=xe[i,j+1];
780:              ys:=ye[i,j];ysi:=ye[i,j+1];
781:              xs:=350+xs;xsi:=350+xsi;
782:              ys:=100-ys;ysi:=100-ysi;
783:              line(xs,ys,xsi,ysi);
784:            end;
785:        end;
786:      readln;
787:      CloseGraph;
788:  end;
789:  procedure c3d; {***  cal rotation 3-D image  ***}
790:  var x,y,i,j,xs,ys : integer;
791:      xc,yc,z  : real;
792:      th1,th2 : real;
793:      w : word;
794:  begin
795:      th1:=3.14159/3;th2:=3.14159/4;
796:      for i:=1 to 64 do
797:        begin
798:           for j := 1 to 64 do
799:             begin
800:               x:=(i*4)-64;y:=(j*4)-64;
```

```
801:                z:=light[2*i,2*j]/3;
802:                xc:=-x*sin(th1)+y*cos(th2);
803:                yc:=-x*cos(th1)*cos(th2)-y*sin(th1)*cos(th2)+z*sin(th2);
804:                xs:=round(xc);
805:                ys:=round(yc);
806:                xe[i,j]:=xs;
807:                ye[i,j]:=ys;
808:             end;
809:       end;
810: end;
811: procedure w64file;   {*****    write gradient of 64 x 64    *****}
812: type  rfile = file of integer;
813: var
814:       w : word;
815:       i,j,t1,t2 : integer;
816:       gxfile : rfile;
817:       gyfile : rfile;
818: begin
819:       assign(gxfile,'b:xs.dta');
820:       assign(gyfile,'b:ys.dta');
821:       rewrite(gxfile);
822:       rewrite(gyfile);
823:       w:=0;
824:       for i := 1 to 64 do
825:          begin
826:          for j := 1 to 64 do
827:                begin
828:                t1 := xe[i,j];
829:                t2 := ye[i,j];
830:                write(gxfile ,t1);
831:                write(gyfile ,t2);
832:                end;
833:          end;
834:      close(gxfile); close(gyfile);
835: end;
836: procedure wrefile;   {*****    write gradient data  *****}
837: type  rfile = file of integer;
838: var
839:       w : word;
840:       i,j,t1,t2 : integer;
```

```
841:       gxfile : rfile;
842:       gyfile : rfile;
843: begin
844:       assign(gxfile,'b:re1.dta');
845:       assign(gyfile,'b:re2.dta');
846:       rewrite(gxfile);
847:       rewrite(gyfile);
848:       w:=0;
849:       for j := 0 to 127 do
850:          begin
851:               t1 := re1[j];
852:               t2 := re2[j];
853:               write(gxfile ,t1);
854:               write(gyfile ,t2);
855:          end;
856:     close(gxfile); close(gyfile);
857: end;
858: procedure r64file; {*****   read gradient of 64 x 64   *****}
859: var
860:       i,j,t1,t2 : integer;
861:       code : integer;
862:       gxfile : file of integer;
863:       gyfile : file of integer;
864:       w : word;
865: begin
866:       assign(gxfile,'b:xs.dta');
867:       assign(gyfile,'b:ys.dta');
868:       reset(gxfile);
869:       reset(gyfile);
870:       w:=0;
871:       for i := 1 to 64 do
872:          begin
873:          for j := 1 to 64  do
874:               begin
875:               read(gxfile ,t1);
876:               read(gyfile ,t2);
877:               xe[i,j]:=t1;
878:               ye[i,j]:=t2;
879:               end;
880:          end;
```

```
881:       close(gxfile); close(gyfile);
882: end;
883: procedure rrefile;  {*****   read gradient data   *****}
884: var
885:       i,j,t1,t2 : integer;
886:       code : integer;
887:       gxfile : file of integer;
888:       gyfile : file of integer;
889:       w : word;
890: begin
891:       assign(gxfile,'b:re1.dta');
892:       assign(gyfile,'b:re2.dta');
893:       reset(gxfile);
894:       reset(gyfile);
895:       w:=0;
896:       for i := 0 to 127 do
897:         begin
898:               read(gxfile ,t1);
899:               read(gyfile ,t2);
900:               re1[j]:=t1;
901:               re2[j]:=t2;
902:         end;
903:      close(gxfile); close(gyfile);
904: end;
905: procedure contour;  {*****   plote contour *****}
906: var
907:   Gd, Gm : integer;
908:   s : integer;
909:   v : word;
910: const Clipon =true;
911: begin
912:   Gd := Detect;
913:   InitGraph(Gd, Gm, '');
914:   if GraphResult <> grOk then
915:     Halt(1);
916:    .settextstyle(0,0,2);
917:     moveto(150,290);
918:     outtext('contour line');
919:     for i:= 0 to 127 do
920:     begin
```

```
921:          for j := 0 to 127 do
922:          begin
923:              s := light[i,j];
924:              case s of
925:              1..40 :  putpixel(3*j+150,2*i,1);
926:              96   :  putpixel(3*j+150,2*i;1);
927:              124  :  putpixel(3*j+150,2*i,1);
928:              147  :  putpixel(3*j+150,2*i,1);
929:              163   :  putpixel(3*j+150,2*i,1);
930:              176  :  putpixel(3*j+150,2*i,1);
931:              185   :  putpixel(3*j+150,2*i,1);
932:              192  :  putpixel(3*j+150,2*i,1);
933:              197   :  putpixel(3*j+150,2*i,1);
934:              197  :  putpixel(3*j+150,2*i,1);
935:              200   :  putpixel(3*j+150,2*i,1);
936:              end;
937:          end;
938:      end;
939:      delay(4000);
940:      CloseGraph;
941: end;
942: procedure tangent;  {***** needle *****}
943: var
944:   s,x,y : integer;
945:   infd1,infd2 : integer;
946:   i,j,t1,t2,t5 : integer;
947:   tx,ty,t3,t4,k:real;
948:   code : integer;
949:   gxfile : file of integer;
950:   gyfile : file of integer;
951:   w,v : word;
952:   Gd, Gm : integer;
953: const Clipon =true;
954: begin
955:   writeln('***** 0 plote zero crossing *****');
956:   Gd := Detect;
957:   InitGraph(Gd, Gm, '');
958:   if GraphResult <> grOk then
959:     Halt(1);
960:     settextstyle(0,0,2);
```

```
 961:      moveto(350,80);
 962:      outtext('tangent contour');
 963:      w:=0;v:=0;
 964:      for i:= 0 to 127 do
 965:      begin
 966:          for j := 0 to 127 do
 967:          begin
 968:            if light[i,j]>0 then
 969:              begin
 970:                t1 := memw[$8800:w];
 971:                t2 := memw[$9000:w];
 972:                if t1 >=3000 then t1:=t1-5000;
 973:                if t2 >=3000 then t2:=t2-5000;
 974:                tx :=t1/1000;ty := t2/1000;
 975:                t3:=0.5*sqrt(tx*tx+ty*ty);t4:=2*180*ArcTan(t3);
 976:                t5:=round(t4/3.14159);
 977:                if i=64 then
 978:                  begin
 979:                    k:=2*arctan(t3);
 980:                    re2[j]:=round(170*cos(k));
 981:                  end;
 982:                mem[$8000:v]:=t5;
 983:              end
 984:            else
 985:              begin
 986:                mem[$8000:v]:=90;
 987:                if i=64 then  re2[j]:=0;
 988:              end;
 989:            case mem[$8000:v] of
 990:              0:putpixel(3*i+150,2*j+50,2);
 991:              3:putpixel(3*i+150,2*j+50,3);
 992:              5:putpixel(3*i+150,2*j+50,2);
 993:              9..10:putpixel(3*i+150,2*j+50,3);
 994:              14..15:putpixel(3*i+150,2*j+50,2);
 995:              19..20:putpixel(3*i+150,2*j+50,3);
 996:              24..25:putpixel(3*i+150,2*j+50,2);
 997:              29..30:putpixel(3*i+150,2*j+50,3);
 998:              34..35:putpixel(3*i+150,2*j+50,2);
 999:              39..40:putpixel(3*i+150,2*j+50,3);
1000:              44..45:putpixel(3*i+150,2*j+50,2);
```

```
1001:                49..50:putpixel(3*i+150,2*j+50,3);
1002:                54..55:putpixel(3*i+150,2*j+50,2);
1003:                59..60:putpixel(3*i+150,2*j+50,3);
1004:                64..65:putpixel(3*i+150,2*j+50,2);
1005:                69..70:putpixel(3*i+150,2*j+50,3);
1006:                79..80:putpixel(3*i+150,2*j+50,2);
1007:
1008:                90:putpixel(3*i+150,2*j+50,2);
1009:                end;
1010:
1011:             w:=w+2;v:=v+1;
1012:          end;
1013:
1014:       end;
1015:       readln;delay(4000);
1016:       closegraph;
1017: end;
1018: procedure error;  {*****   plote error *****}
1019: var
1020:   Gd, Gm : integer;
1021:   s,t1,t2 : integer;
1022:   tt1,tt2 : real;
1023:   v : word;
1024: const Clipon =true;
1025: begin
1026:   Gd := Detect;
1027:   InitGraph(Gd, Gm, '');
1028:   if GraphResult <> grOk then
1029:     Halt(1);
1030:     settextstyle(0,0,2);
1031:     moveto(150,290);
1032:     outtext('error line');
1033:     for j := 0 to 127 do
1034:         begin
1035:            t1:=180-round(re1[j]/2 );
1036:            putpixel(j*4+20,t1,2);
1037:            t2 :=180-round(re2[j]/2);
1038:            putpixel(j*4+20,t2,5);
1039:         end;
1040:     delay(4000);
```

```
1041:      CloseGraph;
1042:  end;
1043:  procedure v3d; {*****  hidden line process    *****}
1044:  type initia = array[0..640] of integer;
1045:  var x,y,z,i,j : integer;
1046:      xe,ye  : real;
1047:      th1,th2 : real;
1048:      ymin,ymax : initia;
1049:      flag,flag1,flag2 : integer;
1050:      oldx,oldy,xs,ys,deltax,deltay   : integer;
1051:      tempx,tempy,newx:integer;
1052:      slope : integer;
1053:      tempy2,tempx2 : boolean;
1054:  begin
1055:      for j:= 1 to 640 do;
1056:        begin
1057:           ymax[j]:=0;ymin[j]:=399;
1058:        end;
1059:      th1:=3.14159/3;th2:=3.14159/4;
1060:      for j:=31 downto 0 do
1061:        begin
1062:           for i := 31 downto 0 do
1063:             begin
1064:               x:=4*i;y:=4*j;
1065:               z:=light[x,y];
1066:               xe:=-x*sin(th1)+y*cos(th2);
1067:               ye:=-x*cos(th1)*cos(th2)-y*sin(th1)*cos(th2)+z*sin(th2);
1068:               xs:=round(300+xe);
1069:               ys:=round(100-ye);
1070:    tempx2:=(tempx < 0) or (tempx > 639 );
1071:    tempy2:=(tempy<0) or (tempy>399);
1072:    if flag=0 then
1073:       begin  {540}
1074:          flag :=1;flag2 :=0;
1075:          oldx:=xs; oldy:=ys;
1076:       end
1077:       else
1078:       begin  {570}
1079:          deltax:=oldx-xs;
1080:          slope:=round(deltay/deltax);
```

```
1081:          tempy:=oldy;
1082:          newx:=oldx+1;
1083:          {b620}
1084:          for tempx:=newx to xs do
1085:             begin
1086:                flag1:=1;tempy:=tempy+slope;
1087:                if tempx2 then   {650}
1088:                   begin
1089:                    flag1:=0;flag2:=0;
1090:                   end
1091:              else
1092:                   begin
1093:                   if tempy2 then    {660}
1094:                    begin
1095:                    flag1:=0;flag2:=0;
1096:                    end
1097:                   else
1098:                    begin
1099:                      if tempy <= ymin[tempx] then   {670}
1100:                       begin   {b730}
1101:                         ymin[tempx]:=tempy;
1102:                         if flag1=0 then
1103:                           begin  {770}
1104:                             if tempy < ymax[tempx] then   {680}
1105:                               begin
1106:                                writeln;
1107:                               end
1108:                               else
1109:                               begin   {780}
1110:                                ymax[tempx]:=tempy;      {b780}
1111:                                if flag1=0 then
1112:                                begin
1113:                                  writeln;
1114:                                end
1115:                                else
1116:                                begin     {e800}
1117:                                  if flag2=0 then
1118:                                     begin
1119:                                      putpixel(tempx,tempy,1);
1120:                                      flag2:=1;
```

```
1121:                                              end;
1122:                                             line(getx,gety,tempx,tempy)
1123:                                            end;    {e800}
1124:                                 end;  {780}
1125:                              end  {770}
1126:                              else
1127:                              begin  { 750 }
1128:                                if flag2=0 then
1129:                                 begin
1130:                                   putpixel(tempx,tempy,1);
1131:                                   flag:=2;
1132:                                 end;
1133:                                line(getx,gety,tempx,tempy);
1134:                                if tempy < ymax[tempx] then
1135:                                 begin
1136:                                   writeln;
1137:                                 end
1138:                                 else
1139:                                 begin    {780}
1140:                                   ymax[tempx]:=tempy;      {b780}
1141:                                   if flag1=0 then
1142:                                   begin
1143:                                     writeln;
1144:                                   end
1145:                                   else
1146:                                   begin    {e800}
1147:                                     if flag2=0 then
1148:                                       begin
1149:                                         putpixel(tempx,tempy,1);
1150:                                         flag2:=1;
1151:                                         end;
1152:                                     line(getx,gety,tempx,tempy);
1153:                                   end;    {e800}
1154:                                end;  {780}
1155:                           end;   {750}
1156:                      end  {e730}
1157:                      else
1158:                      begin
1159:                        if tempy>=ymax[tempx] then
1160:                         begin        {b780}
```

```
1161:                                   ymax[tempx]:=tempy;
1162:                                       if flag1=0 then
1163:                                         begin
1164:                                            writeln;
1165:                                         end
1166:                                         else
1167:                                         begin      {e800}
1168:                                            if flag2=0 then
1169:                                               begin
1170:                                                  putpixel(tempx,tempy,1);
1171:                                                  flag2:=1;
1172:                                               end;
1173:                                            line(getx,gety,tempx,tempy);
1174:                                         end;     {e800}
1175:                                   end            {e780}
1176:                                   else
1177:                                      begin
1178:                                      flag2:=0;
1179:                                   end;      {680}
1180:                              end; {670}
1181:                         end; {660}
1182:                    end;  {650}
1183:                 end;    {620}
1184:       end;   {570}
1185:       flag:=1;flag2:=0;
1186:       oldx:=xs;oldy:=ys;
1187:             end;  {i loop}
1188:        end;    {j loop}
1189: end;
1190: begin
1191: end.
```