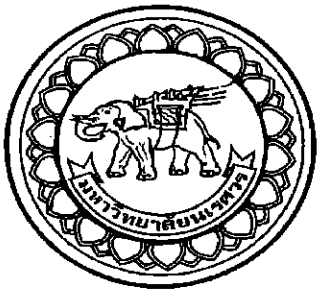



# การสร้างภาพ 3 มิติ จากภาพ 2 มิติ

**Create 3D image structure from 2D image structure**

นางสาวกฤติกา อภิวงค์งาม รหัส 47361787

นางสาวปรางนภา ทารัตน์ รหัส 47362017

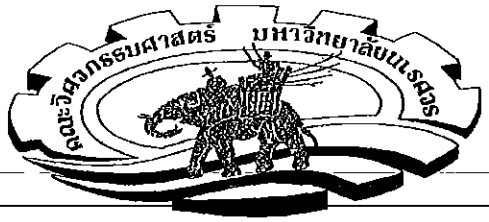


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550




# ใบรับรองโครงงานวิศวกรรม

**หัวข้อโครงงาน** การสร้างภาพ 3 มิติ จากภาพ 2 มิติ

**ผู้ดำเนินโครงงาน** นางสาวกฤติกา อภิวงค์งาม รหัส 47361787

นางสาวปรางนภา ทารัตน์ รหัส 47362017

**อาจารย์ที่ปรึกษา** ดร. สุรเดช จิตประไพกุลศาล

**สาขาวิชา** วิศวกรรมคอมพิวเตอร์

**ภาควิชา** วิศวกรรมไฟฟ้าและคอมพิวเตอร์

**ปีการศึกษา** 2550

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.สุรเดช จิตประไพกุลศาล)

.....กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ

(ดร.ไพศาล มุณีสว่าง)

| | | | | | |
|---|---|---|---|---|---|
| **หัวข้อโครงงาน** | การสร้างภาพ 3 มิติ จากภาพ 2 มิติ | | | | |
| **ผู้ดำเนินโครงงาน** | นางสาวกฤติกา | อภิวงค์งาม | รหัส | 47361787 | |
| | นางสาวปรางนภา | ทารัตน์ | รหัส | 47362017 | |
| **อาจารย์ที่ปรึกษา** | ดร. สุรเดช | จิตประไพกุลศาล | | | |
| **สาขาวิชา** | วิศวกรรมคอมพิวเตอร์ | | | | |
| **ภาควิชา** | วิศวกรรมไฟฟ้าและคอมพิวเตอร์ | | | | |
| **ปีการศึกษา** | 2550 | | | | |


## บทคัดย่อ

โครงงานนี้ได้ทำการสร้างภาพ 3 มิติ จากภาพ 2 มิติ โดยศึกษาจากงานวิจัยที่มีชื่อว่า Image - based 3D Reconstruction of Generalized Box with User Sketches ซึ่งเขียนโดย Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets ซึ่งจุดประสงค์ของโครงงานนี้จะเป็นการสร้างภาพ 3 มิติของวัตถุโดยการจำกัดรูปทรงจากรูปภาพ รวมทั้งให้ผู้ใช้ได้มีส่วนร่วมด้วย สำหรับรูปทรงที่ใช้นั้นคือวัตถุทรงกล่อง มี 6 ด้าน แต่ละด้านประกอบไปด้วยเส้นเล็ก ๆ เส้นตรงรวมไปถึงส่วนโค้งที่สร้างขึ้นจากโปรแกรมสร้างภาพสามมิติ ขั้นตอนแรกจะเริ่มสร้างเส้น 3 มิติจากเส้นที่เห็นก่อน จากนั้นจึงจะสร้างเส้น 3 มิติที่มองไม่เห็นจากเส้นที่มองเห็น ขั้นตอนสุดท้ายเป็นการสร้างพื้นผิวของแต่ละด้าน หลังจากนั้นจึงได้รูปร่าง 3 มิติ จาก input ที่เป็นรูปภาพ

| | |
|---|---|
| **Project Title** | Create 3D image structure from 2D image structure |
| **Name** | Ms. Krittika Apiwongngam ID. 47361787 |
| | Ms. Prangnapa Tarat ID. 47362017 |
| **Project Advisor** | Dr. Suradet Jitprapaikulsarn |
| **Major** | Computer Engineering |
| **Department** | Electrical and Computer Engineering |
| **Academic Year** | 2007 |


## ABSTRACT

This project reconstructs a 3D image structure from a 2D image structure using a method described in "Image – based 3D Reconstruction of Generalized Box with User Sketch" by Anton Yakubenko, Anton Konouchine and Vladimir Vezhnevets. In this project, we will reconstruct an image of a box created from a 3D program. First, 3D edges are reconstructed from the given lines. Then missing 3D edges are obtained from those 3D edges. Finally surface is constructed for each face.

# กิตติกรรมประกาศ

ขอขอบพระคุณ ดร.สุรเดช จิตประไพกุลศาล อาจารย์ที่ปรึกษาโครงงานนี้ ดร.พนมขวัญ ริยะมงคล และ ดร.ไพศาล มุณีสว่าง ที่คอยให้คำปรึกษา ความช่วยเหลือตลอดจนคำแนะนำและแนวทางต่างๆ ในการทำโครงงานนี้ และสุดท้ายขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ ทุกคนที่ยังไม่ได้เอ่ยนามที่คอยให้การสนับสนุนผู้ดำเนินโครงงาน ให้สามารถทำโครงงานนี้จนสำเร็จลุล่วงไปได้ด้วยดี

# สารบัญ

หน้า

# สารบัญ(ต่อ)

# สารบัญตาราง

หน้า

# สารบัญรูปภาพ

หน้า

# สารบัญรูปภาพ(ต่อ)

หน้า

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญของโครงงาน

ในอดีตที่ผ่านมาการเก็บหลักฐาน เริ่มจากการใช้ภาพถ่ายขาวดำ จากนั้นจึงพัฒนามาเป็นภาพสี รวมถึงการเก็บเป็นวีดีทัศน์ แต่อย่างไรก็ตามข้อมูลทั้งหมดที่เก็บได้ก็อยู่ในรูปของภาพ 2 มิติ ซึ่งเราไม่ทราบความกว้างและความลึก รวมถึงสัดส่วนของวัตถุนั้นอย่างชัดเจน

โครงงานการสร้างภาพ 3มิติ จากภาพ 2 มิติ จัดทำขึ้นเพื่อศึกษาและพัฒนาวิธีการสร้างภาพ 3 มิติ จากรูปภาพ 2 มิติ อันเป็นพื้นฐานของการพิสูจน์หลักฐานทั้งทางโบราณคดี และทางนิติวิทยาศาสตร์ โดยการสร้างภาพ 3 มิตินี้จะประกอบไปด้วยการนำรูปภาพ 2 มิติ เข้ามาในโปรแกรมคอมพิวเตอร์เพื่อทำการคำนวณหาความกว้างและระยะลึกของภาพ ก่อนจะประมวลผลเพื่อสร้างภาพ 3 มิติต่อไป

ผลที่คาดว่าจะได้รับจากโครงงานนี้คือ สามารถนำเอาเทคนิควิธีการสร้างภาพ 3 มิติ จากภาพ 2 มิติ ไปใช้ในงานพิสูจน์หลักฐานต่อไป

## 1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษากระบวนการสร้างภาพ 3 มิติจากภาพ 2 มิติ
   - เพื่อศึกษาทฤษฎี epipolar geometry
   - เพื่อศึกษา Fumdamental matrix ที่เป็นพื้นฐานในการหาข้อผิดพลาดน้อยที่สุดในการหา reprojection error
   - เพื่อศึกษาโปรแกรม MATLAB
2. เพื่อเป็นพื้นฐานในการพัฒนางานด้านการพิสูจน์หลักฐาน

## 1.3 ขอบข่ายของโครงงาน

1. โครงงานนี้จะสร้างภาพ 3 มิติ จากภาพกล่องที่สร้างขึ้นเท่านั้น
2. ภาพ 3 มิติที่ได้จะมีรายละเอียดเฉพาะ ด้านที่มีแสดงในภาพ 2 มิติ

## 1.4 ขั้นตอนการดำเนินงาน

โครงงานนี้เริ่มต้นด้วยการศึกษาทฤษฎีสี และ ศึกษาลักษณะของภาพและโครงสร้างของภาพ 2 มิติ และ 3 มิติ จากนั้นจึงเลือกภาพ 3 มิติที่เหมาะสมกับโครงงาน พร้อมทั้งศึกษา

กระบวนการหาระยะลึกจากภาพแล้วจึงศึกษาการใช้โปรแกรมที่เกี่ยวข้อง สุดท้ายจะเป็นการเขียนโปรแกรมเพื่อสร้างภาพ 3 มิติ

| กิจกรรม | ปี 2549 | ปี 2550 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ธ.ค. | ม.ค. | ก.พ. | มี.ค. | เม.ย. | พ.ค. | มิ.ย. | ก.ค. | ส.ค. | ก.ย. |
| 1. วางแผนการทำงาน | ← | | | | | | | | | |
| 2. ศึกษาอัลกอริทึมการสร้างภาพ 3 มิติจากภาพ 2 มิติของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets | | ← | | | | → | | | | |
| 3. ศึกษาโปรแกรม Java และ MATLAB | | | ← | | | → | | | | |
| 4. ศึกษาวิธีการ interpolation | | | | ← | | → | | | | |
| 5. ศึกษาการหาค่า error | | | | | ← | | | → | | |
| 6. ศึกษาการหาเส้นที่มองไม่เห็นจากเส้นที่มองเห็น | | | | | | | ← | → | | |
| 7. ศึกษาการหาสีพื้นผิวของวัตถุ | | | | | | | | | ↔ | |
| 6. เขียนโปรแกรมสร้างภาพ 3 มิติ | | | | | | ← | | | | → |
| 7. ทดสอบการทำงานและแก้ไขข้อผิดพลาดโปรแกรม | | | | ← | | | | | | → |
| 8. จัดทำคู่มือโครงงาน | | | | ← | | | | | | → |
| 9. ตรวจสอบและปรับปรุงแก้ไขคู่มือ | | | | | ← | | | | | → |
| 10. ส่งโครงงานฉบับสมบูรณ์ | | | | | | | | | | ↔ |

## 1.5 ผลที่คาดว่าจะได้รับ

1. เพื่อศึกษากระบวนการสร้างภาพ 3 มิติจากภาพ 2 มิติ
   - ได้วัตถุ 3 มิติจากภาพ 2 มิติ
   - ความรู้และความเข้าใจในทฤษฎี epipolar geometry
   - สามารถนำความรู้เรื่อง Fundamental matrix ใช้ในการหาข้อผิดพลาดที่น้อยที่สุดในการหา reprojection error
   - ความรู้และทักษะในการใช้โปรแกรม MATLAB
2. พัฒนางานด้านการพิสูจน์หลักฐาน

## 1.6 งบประมาณที่ต้องใช้

| | | | |
|---|---|---|---|
| 1. | ค่าเอกสาร | เป็นเงิน | 500 บาท |
| 2. | ค่าวัสดุอุปกรณ์ | เป็นเงิน | 1,200 บาท |
| 3. | ค่าจัดทำรูปเล่มโครงงาน | เป็นเงิน | 300 บาท |
| | รวมค่าใช้จ่าย | รวมเป็นเงิน | 2,000 บาท (สองพันบาทถ้วน) |

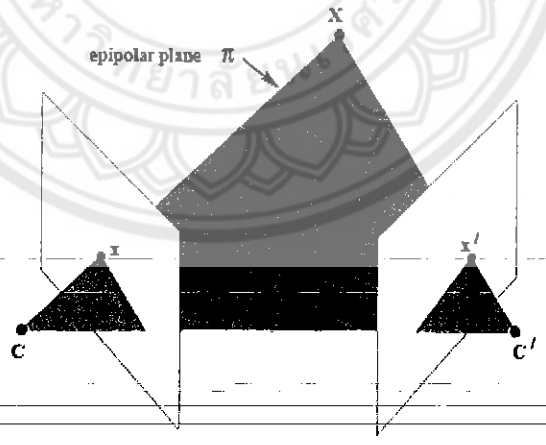
# บทที่ 2
# หลักการและทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงหลักการและทฤษฎีต่างๆ ที่นำมาประยุกต์ใช้เพื่อทำโครงงานนี้ ซึ่งทฤษฎีที่ได้ใช้ ได้แก่ ทฤษฎีและการคำนวน Epipolar geometry เพื่อให้เป็นพื้นฐานของการคำนวนหา Fundamental matrix หลักการคำนวน Fundamental matrix เพื่อการคำนวนหา error ที่น้อยที่สุดในการ projection และแสดงการคำนวนหา error ที่น้อยที่สุดในการ projection และจากที่กล่าวมาทั้งหมดจะเป็นพื้นฐานในอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets

## 2.1 เรขาคณิตอีพิโพลาร์ (Epipolar Geometry)

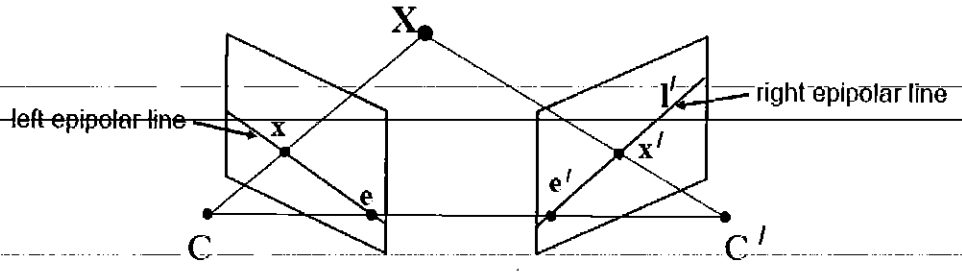
เรขาคณิตอีพิโพลาร์ (Epipolar Geometry) เป็นการเปรียบเทียบวัตถุชิ้นเดียวกันจากต่างมุมมองโดยมี baseline เป็นจุดอ้างอิง

จากรูปที่ 2.1 กำหนดให้ $x$ อยู่ในภาพแรก และ $x'$ ในภาพที่สอง จากรูป 2.1 พื้นที่สีเทาจะเรียกว่า epipolar plane $\pi$ โดยมีจุดยอดคือ $X$ , $C$ และ $C'$ เป็น camera centres ซึ่งจุด $C$ จะลากผ่านจุด $x$ ไปตัดกับ $C'$ ที่ลากผ่านจุด $x'$ ที่จุด $X$




a

**รูปที่ 2.1** Point correspondence geometry [2]




**รูปที่ 2.2** แสดงเส้น epipolar line [2]

จากรูปที่ 2.2

- $l'$ เป็น epipolar line ที่เกิดจาก ray จาก $X$
- e (epipole) เป็นจุดที่ใช้อ้างอิงระหว่าง 2 ภาพ

## 2.2 การเขียนเมทริกซ์ของเวคเตอร์จากผลคูณเชิงเวคเตอร์ (cross product)

จาก เวคเตอร์ $V \times X$ สามารถเขียนเป็นเมทริกซ์ที่คูณกันได้ดังนี้

$$V \times X = \begin{pmatrix} v_2 x_3 - v_3 x_2 \\ v_3 x_1 - v_1 x_3 \\ v_1 x_2 - v_2 x_1 \end{pmatrix} = [V]_X X \qquad (1)$$

โดยที่

$$[V]_X = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \qquad (2)$$

- จะได้ว่า $[V]_X$ เป็นเมทริกซ์ขนาด $3 \times 3$ และมี rank = 2

## 2.3 เมทริกซ์ความสัมพันธ์ระหว่างจุด 2 จุดบนเส้นตรง

กำหนดให้จุด $(x, y)$ ใน 2 มิติถูกอธิบายด้วย homogeneous 3 – vector คือ $x = (x_1, x_2, x_3)^T$ ที่ซึ่ง $x = x_1 / x_3, y = x_2 / x_3$ จะได้

$l$ คือเส้น 2 มิติที่ถูกอธิบายด้วย homogeneous 3 – vector

$$l = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix}$$

โดยที่ $l_1 x + l_2 y + l_3 = 0$

**ตัวอย่าง** กำหนดให้เส้น $y = 1$ เป็น homogeneous vector

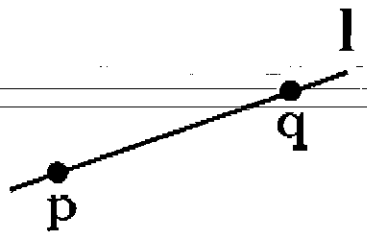
สามารถเขียนเส้นได้ดังนี้คือ $-y + 1 = 0$ ดังนั้น $l_1 = 0$, $l_2 = -1$, $l_3 = 1$ จะได้ $l = (0, -1, 1)^T$

สามารถเขียนจุดเป็นเส้นจากสมการ $l_1 x_1 + l_2 x_2 + l_3 x_3 = 0$


**รูปที่ 2.3** จุด x บนเส้น $l$ [2]

จากจุดบนเส้นจะได้ $l \cdot x = 0$ หรือ $l^T x = 0$ หรือ $x^T l = 0$

ถ้ามีจุด $p$ และ $q$ และมีเส้น $l$ ลากผ่าน จะได้ว่า $l = p \times q$




จากรูปจะได้

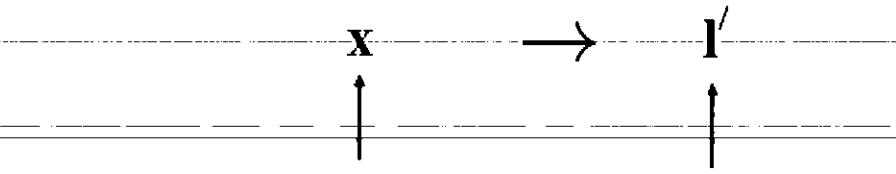
$$l \cdot p = (p \times q) \cdot p = 0 \quad (3)$$

$$l \cdot q = (p \times q) \cdot q = 0 \quad (4)$$

**รูปที่ 2.4** จุด $p$ และ $q$ โดยมีเส้น $l$ ลากผ่าน [2]

## 2.4 เมทริกซ์พื้นฐาน (The Fundamental matrix; F)

เมทริกซ์พื้นฐาน (Fundamental matrix) เป็นการคำนวนชนิดหนึ่งของเรขาคณิตอีพิโพลาร์ (epipolar geometry) ซึ่งจะคำนวนจากการ map ระหว่างจุดของเส้นอีพิโพลาร์ (epipolar line) ของมัน




**รูปที่ 2.5** การ map ระหว่างจุดและ epipolar line [2]

การ map จะขึ้นอยู่กับ camera centres $C$ และ $C'$ จาก $x \to l'$ แสดงให้เห็นว่าเป็นการ map แบบ linear จึงสามารถเขียนเป็น $l' = Fx$ ได้ ที่ซึ่ง F เป็นเมทริกซ์ขนาด $3 \times 3$ ดังนั้นจึงเรียก F ว่าเป็น fundamental matrix

**ขั้นตอนการคำนวนหาสมการ Fundamental matrix**

สมการการหา matrix camera ใช้สมการดังนี้

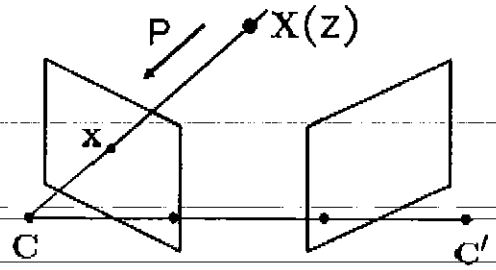
$$P = K[R \mid t] \tag{5}$$

internal calibration — rotation — translation (from world to camera coordinate frame)

Camera ที่หนึ่งจะได้สมการ matrix camera คือ $P = K[I \mid 0]$ (6)

Camera ที่สองจะได้สมการ matrix camera คือ $P' = K'[R \mid t]$ (7)

**ขั้นตอนที่ 1**




รูปที่ 2.6 การ project ย้อนกลับไปยังจุด x [2]

จากจุด $x$ จะ project กลับไปยัง camera centre ($C$) จะได้สมการ matrix camera คือ $P = K[I \mid 0]$

จากรูปที่ 2.6 จุด $x$ project กลับมาจาก ray $X(z)$ ซึ่งทำให้

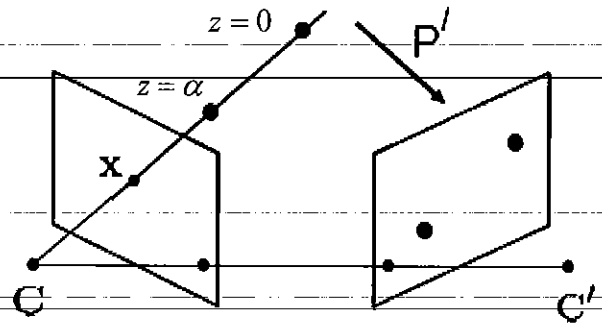
$$PX(z) = K[I \mid 0]X(z) = x \tag{8}$$

โดยที่ z เป็นความลึกของจุด

$$x = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K[I \mid 0] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = K \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{9}$$

$$X(z) = \begin{pmatrix} zK^{-1}x \\ 1 \end{pmatrix} \tag{10}$$

**ขั้นตอนที่ 2**




**รูปที่ 2.7** การเลือกจุดบน ray [2]

เลือกจุดบน ray และ project กลับไปยัง camera centre ($C'$) จะได้สมการ matrix camera คือ $P' = K'[R \mid t]$

จากภาพ กำหนดจุดสองจุดบน ray $X(z) = \begin{pmatrix} zK^{-1}x \\ l \end{pmatrix}$

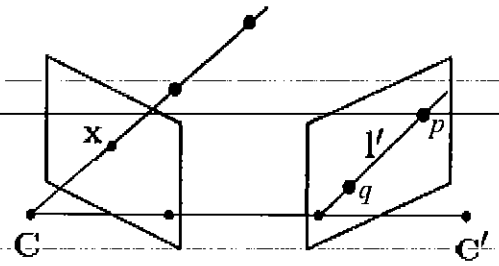
- $z = 0$ เป็น camera centre $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- $z = \alpha$ เป็นจุดที่อนันต์ $\begin{pmatrix} K^{-1}x \\ 0 \end{pmatrix}$

project 2 จุดนี้ไปยังมุมมองที่สองจะได้ว่า

$$P'\begin{pmatrix} 0 \\ 1 \end{pmatrix} = K''[R \mid t]\begin{pmatrix} 0 \\ 1 \end{pmatrix} = K't \tag{11}$$

และ $$P\begin{pmatrix} K^{-1}x \\ 0 \end{pmatrix} = K'[R \mid t]\begin{pmatrix} K^{-1}x \\ 0 \end{pmatrix} = K'RK^{-1}x \tag{12}$$

**ขั้นตอนที่ 3**




**รูปที่ 2.8** การคำนวนหาเส้นจากจุด 2 จุด [2]

**คำนวน**เส้นจากจุด 2 จุด โดยใช้ความสัมพันธ์ $l' = p \times q$

**คำนวน**เส้นจากจุดโดยใช้สมการ $l' = (K't) \times (K'RK^{-1}x)$

**จาก**คุณ**สมบัติ** identity

$$(Ma) \times (Mb) = M^{-\mathrm{T}}(a \times b) \text{ ที่ซึ่ง } M^{-\mathrm{T}} = (M^{-1})^{\mathrm{T}} = (M^{\mathrm{T}})^{-1} \tag{13}$$

**จะได้**

$$l = K'^{-\mathrm{T}}(t \times (RK^{-1}x)) = \underbrace{K'^{-\mathrm{T}}[t]_x RK^{-1}}_{F} x \tag{14}$$

**โดยที่** F เป็น fundamental matrix

**ดังนั้น**

$$l' = Fx \tag{15}$$

$$F = K'^{-\mathrm{T}}[t]_x RK^{-1} \tag{16}$$

จุด $x$ และ $x'$ สัมพันธ์กัน $(x \leftrightarrow x')$ ดังนั้น $x'^{\mathrm{T}}l' = 0$ จะได้

$$x'^{\mathrm{T}}Fx = 0 \tag{17}$$

**2.4.1 คุณสมบัติของ Fundamental matrix**

- F เป็น rank 2 และ homogeneous matrix โดยมี 7 degrees ที่เป็นอิสระต่อกัน
- Point correspondence:
  ถ้า $x$ และ $x'$ เป็นจุดเดียวกัน ดังนั้น $x'^{\mathrm{T}}Fx = 0$
- Epipolar lines:
  - $l' = Fx$ เป็น epipolar line ของจุด $x$
  - $l = F^{\mathrm{T}}x'$ เป็น epipolar line ของจุด $x'$
- Epipole:
  - $Fe = 0$
  - $F^{\mathrm{T}}e' = 0$
- การคำนวนจาก camera matrix $P, P'$ :
  $P = K[I \mid 0]$
  $P' = K'[R \mid t]$
  $F = K'^{-\mathrm{T}}[t]_X RK^{-1}$

## 2.5 ตำแหน่งมาตรฐานภายใน (Internal calibration)

Internal calibration matrix (K) คือ การเปลี่ยนจากพิกัดของภาพในหน่วย pixels ไปยัง world coordinated ในหน่วยมิลลิเมตร ซึ่งประกอบไปด้วย parameter ที่อยู่ภายใน camera นี้ได้เช่น ความยาวโฟกัส, ตำแหน่งของจุดในภาพ เป็นต้น

$$K = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{18}$$

โดยที่

$\alpha_u, \alpha_v$ เป็นความยาวโฟกัสในเทอมของ pixel

$u_0, v_0$ เป็นตำแหน่งของจุดในภาพ

## 2.6 การแปลงค่าเมทริกซ์ (Matrix transformation)

Matrix transformation เป็นการเปลี่ยนตำแหน่งของจุดใด ๆ ด้วยการดำเนินการแบบเมท-ริกซ์เพื่อทำการเลื่อนตำแหน่ง (translation) หมุน (rotation) และการเปลี่ยนแปลงขนาดของวัตถุ โดยการ transformation โดยจะกล่าวถึงการคำนวนสำหรับจุดใน coordinate เพียงจุดเดียว แต่

เนื่องจากวัตถุสามารถประกอบขึ้นจากจุดเหล่านี้ได้ ดังนั้นการกระทำกับจุดทุกจุดบนวัตถุดังกล่าวก็คือการทำกับวัตถุทั้งหมด

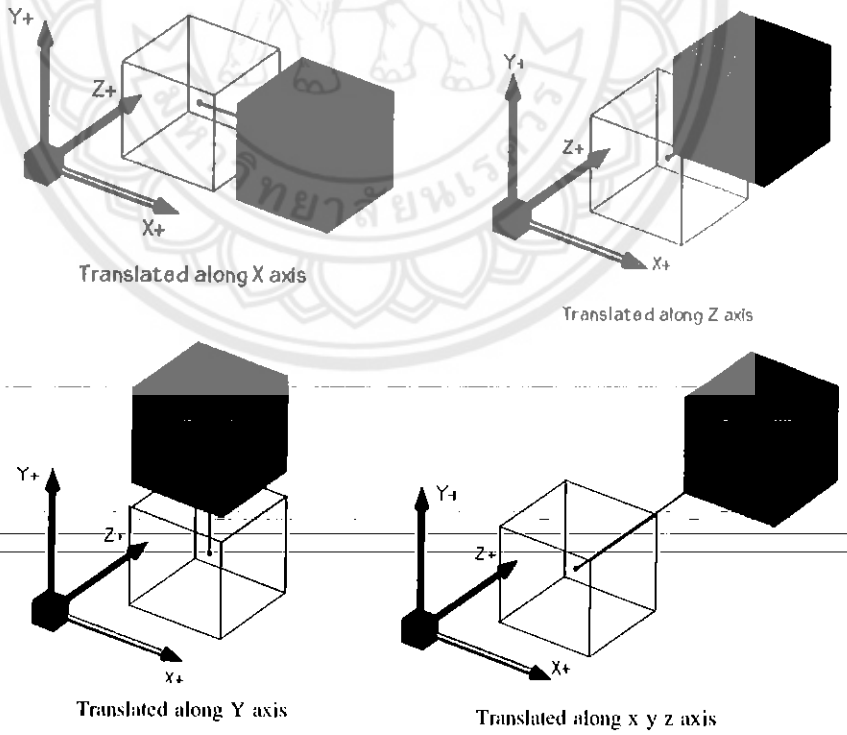
### 2.6.1 การเลื่อนตำแหน่ง (Translation)

Translation หรือการเลื่อนจุดเป็นการเลื่อนตำแหน่งของจุดในแนวเส้นตรงในสามมิติ ถ้าเป็นการเลื่อนวัตถุ วัตถุทั้งอันก็จะเลื่อนตำแหน่งในทิศทางเดียวกันด้วยระยะทางที่เท่ากันทุกจุด ซึ่งอาจจะเป็นการเลื่อนในแนวแกน x, y หรือ z หรือเป็นการเลื่อนตำแหน่งในทั้งสามแกน

ถ้าหากเราต้องการเลื่อนจุดจุดหนึ่งจาก $x_1,\ y_1$ ไปยังจุด $x_2,\ y_2,\ z_2$ จะต้องมีความสัมพันธ์ดังนี้

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{19}$$

โดยที่ $t_x$, $t_y$ และ $t_z$ คือระยะที่ต้องการเลื่อนตำแหน่งในแนวแกน x, y และ z




**รูปที่ 2.9** ภาพแสดงการ transform วัตถุแบบ translate [3]

**2.6.2 การหมุน (Rotation)**

Rotation คือการหมุนจุดในสามมิติรอบแกน แกนหนึ่งที่เรากำหนดให้ ถ้าเป็นการหมุนวัตถุ จุดทุกจุดบนวัตถุก็จะถูกหมุนกวาดเป็นมุมที่เท่ากันรอบแกนหมุนดังกล่าว การ rotation นี้สามารถกระทำการได้โดยหมุนรอบแกน x, y หรือ z หรือเป็นการหมุนรอบแกนใด ๆ ในสามมิติ

ถ้าหากต้องการที่จะหมุนวัตถุรอบแกน x, y หรือ z สามารถทำได้ดังนี้

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(pitch) & -\sin(pitch) \\ 0 & \sin(pitch) & \cos(pitch) \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad (20)$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos(yaw) & 0 & \sin(yaw) \\ 0 & 1 & 0 \\ -\sin(yaw) & 0 & \cos(yaw) \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad (21)$$

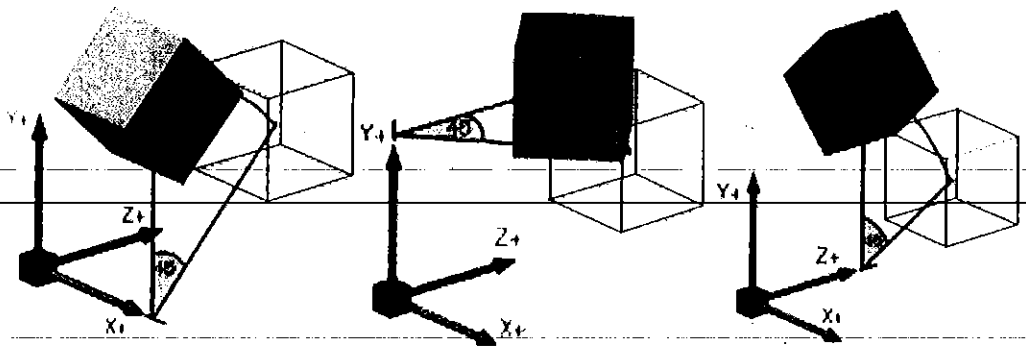
$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos(roll) & -\sin(roll) & 0 \\ \sin(roll) & \cos(roll) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad (22)$$

โดยที่ pitch, yaw และ row คือมุมที่ใช้ในการหมุนในแนวแกน x, y หรือ z ตามลำดับ

**ข้อสังเกต**

เมทริกซ์ไม่มีคุณสมบัติการสลับที่ของการคูณ

ดังนั้น $[rotate]_X[rotate]_Y \neq [rotate]_Y[rotate]_X$




45 degree rotated along X axis  45 degree rotated along Y axis  45 degree rotated along Z axis

**รูปที่ 2.10** ภาพแสดงการ transform วัตถุแบบ rotate [3]

## 2.7 การลดความผิดพลาดของการฉายภาพซ้ำให้น้อยที่สุด (Minimization of reprojection error)

จากที่ผ่านมาถ้าจุดที่อยู่ในทั้งสองภาพอยู่ตรงกับ epipoles จุดนั้นจะอยู่บนเส้นที่ติดกับ camera centres ในกรณีนี้เป็นไปไม่ได้ที่จะชี้ชัดไปว่าจุดนั้นอยู่ในตำแหน่งใดของระนาบ แต่ถ้ามีจุดใดจุดหนึ่งตรงกับจุดที่อยู่บน epipole เราจะสรุปว่าจุดในระนาบนั้นจะต้องเป็นจุดที่อยู่ตรง camera centres อื่น ๆ ดังนั้นเราจะสมมติว่าจุด $u$ และ $u'$ ไม่ได้สอดคล้องกับ epipole

ในกรณีนี้ เราจะวิเคราะห์แบบง่าย ๆ โดยการใช้ประโยชน์จาก rigid transformation ในแต่ละรูปเพื่อแทน $u$ และ $u'$ ที่จุด origin คือ $(0,\ 0,\ 1)^{\mathrm{T}}$ ใน homogeneous coordinates ยิ่งกว่านั้น epipoles อาจจะถูกแทนที่บนแกน x ที่จุด $(1,\ 0,\ f)^{\mathrm{T}}$ และ $(1,\ 0,\ \hat{f})^{\mathrm{T}}$ ตามลำดับ ค่าของ $f$ เท่ากับ 0 หมายความว่า epipole เข้าสู่ infinity ประโยชน์จาก rigid transformation 2 ตัวคือไม่มีผลกระทบจากผลรวมของฟังก์ชันระยะห่างของราก

ดังนั้นเราจะสมมติ homogeneous coordinates คือ $u = u' = (0,\ 0,\ 1)^{\mathrm{T}}$ และ epipoles 2 ตัวจะอยู่ที่จุด $(1,\ 0,\ f)^{\mathrm{T}}$ และ $(1,\ 0,\ \hat{f})^{\mathrm{T}}$

เมื่อ $F(1,\ 0,\ f)^{\mathrm{T}} = (1,\ 0,\ \hat{f})^{\mathrm{T}} = 0$ ดังนั้น fundamental matrix จะมีรูปแบบพิเศษดังนี้

$$F = \begin{pmatrix} \hat{f}fd & -fc & -\hat{f}d \\ -fd & a & b \\ -fd & c & d \end{pmatrix} \tag{23}$$

พิจารณาเส้น epipolar ที่ผ่านไปยังจุด $(0,\ t,\ 1)^{\mathrm{T}}$ (ยังคงอยู่ใน homogeneous coordinates) และ epipole $(1,\ 0,\ f)^{\mathrm{T}}$ เราแทน epipolar line ด้วย $\lambda(t)$ ส่วนเวคเตอร์ของเส้นนี้ถูกอธิบายด้วย cross product $(0, t, 1) \times (1,\ 0,\ f) = (tf,\ 1,\ -t)$

ดังนั้น กำลังสองของระยะห่างจากเส้นถึงจุด origin คือ

$$d(u,\lambda(t))^2 = \frac{t^2}{1+(tf)^2} \qquad (24)$$

ใช้ fundamental matrix เพื่อหาเส้น epipolar ที่สอดคล้องกันในรูปอื่น ๆ ดังนี้

$$\lambda'(t) = F(0,t,1)^T = (-f'(ct+d), at+b, ct+d)^T \qquad (25)$$

สมการต่อไปนี้จะแสดงว่าเส้น $\lambda'(t)$ เป็น homogeneous vector แล้วกำลังสองของระยะห่าง ของเส้นนี้จากจุด origin คือ

$$d(u',\lambda'(t))^2 = \frac{(ct+d)^2}{(at+b)^2 + f'^2(ct+d)^2} \qquad (26)$$

ดังนั้นกำลังสองของระยะห่างทั้งหมดคือ

$$s(t) = \frac{t^2}{1+f^2t^2} + \frac{(ct+d)^2}{(at+b)^2 + f'^2(ct+d)^2} \qquad (27)$$

งานของเราคือการหาค่าที่น้อยที่สุดของฟังก์ชันนี้

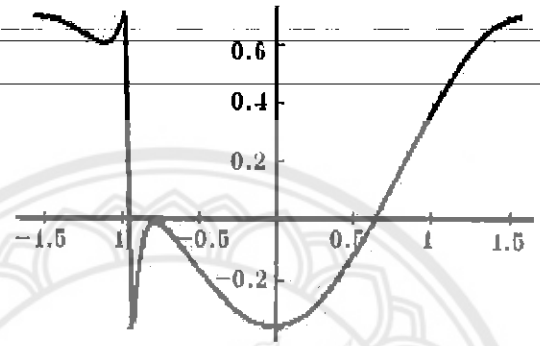
เราสามารถหาค่าที่น้อยที่สุดโดยเทคนิคแคลคูลัสเบื้องต้น โดยเราจะทำการคำนวนโดยการหาค่า differential

$$s'(t) = \frac{2t}{(1+f^2t^2)^2} - \frac{2(ad-bc)(at+b)(ct+d)}{((at+b)^2 + f'^2(ct+d)^2)^2} \qquad (28)$$

Maxima และ minima ของ $s(t)$ จะเกิดขึ้นเมื่อ $s'(t) = 0$ ค่าของทั้งสองเทอมใน $s'(t)$ มากว่าตัวหารร่วม และเมื่อค่าของเศษเท่ากับ 0 ได้เงื่อนไข

$$f(t) = t((at+b)^2 + f'^2(ct+d)^2)^2 - (ad-bc)(1+f^2t^2)^2(at+b)(ct+d)$$

$$= 0 \qquad (29)$$




รูปที่ 2.11 ตัวอย่างของฟังก์ชัน cost และ three minima [4]

ค่า minima และ maxima ของ $s(t)$ จะเห็นได้จากรากของ polynomial นี้ มันเป็น polynomial กำลัง 6 ซึ่งแต่ละดีกรีจะมีรากจริงได้มากกว่า 6 ราก ซึ่งจะเป็นค่า minima 3 ค่าและ ค่า maxima 3 ค่าจากฟังก์ชัน $s(t)$ ค่าน้อยสุดสัมบูรณ์ของฟังก์ชัน $s(t)$ จะหาได้จากรากของฟังก์ชัน $f(t)$ และหาค่าฟังก์ชัน $s(t)$ จากค่ารากจริงแต่ละค่าจากสมการ (4) ที่ง่ายมากกว่านั้นคือเราจะตรวจสอบค่าของ $s(t)$ ที่ส่วนจริงของแต่ละราก (complex หรือ real) ของ $f(t)$ ทำให้ลดความยุ่งยากในการตัดสินว่ารากใดเป็น real หรือ complex อีกอย่างหนึ่งคือควรจะตรวจสอบค่าลิมิตของ $s(t)$ ที่ $t \to \infty$ ถ้าระยะทางน้อยที่สุดเกิดขึ้นเมื่อ $t = \infty$ จะสอดคล้องกับเส้น epipolar $-fu = 1$ ในรูปแรก

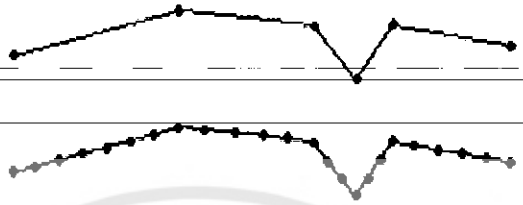
## 2.8 หลักการของ Anton Yakubenko, Anton Konouchine, Vladimir Vezhnevets

### 2.8.1 การสร้างเส้นขอบ 3 มิติ (3d Edge Reconstruction)

ขั้นตอนแรกของการสร้างแบบจำลองคือการสร้างเส้นโค้งของเส้นสามมิติที่ตรงกับอินพุตจากเส้นสองมิติ โดยปกติแล้วเมื่อต้องการสร้างเส้นสามมิติจากเส้นที่ถูก project (ฉาย) จะใช้รูปภาพตั้งแต่สองรูปขึ้นไป พร้อมทั้งจำเป็นต้องทราบความสัมพันธ์ของทุกจุดบนเส้นสองมิติ หรืออย่างน้อยสุดก็ต้องทราบความสัมพันธ์ระหว่างจุดบางจุดที่ต้องทราบ และจุดที่จะถูกนำไปแบ่ง จุดเหล่านี้จะถูก triangulated แล้วจะได้เส้นเล็กย่อย ๆ ที่เป็นเส้นสามมิติออกมา

จุดประสงค์ของกระบวนการสำหรับการทำนายจุดที่ตรงกันจากจุดของ polylines สองมิติที่มีอยู่มากมายหรือส่วนที่เป็นส่วนโค้งเพื่อทำการสร้างเส้นขอบสามมิติขึ้นมา ยกตัวอย่างเช่น กำหนดให้มีเส้น input สามเส้น

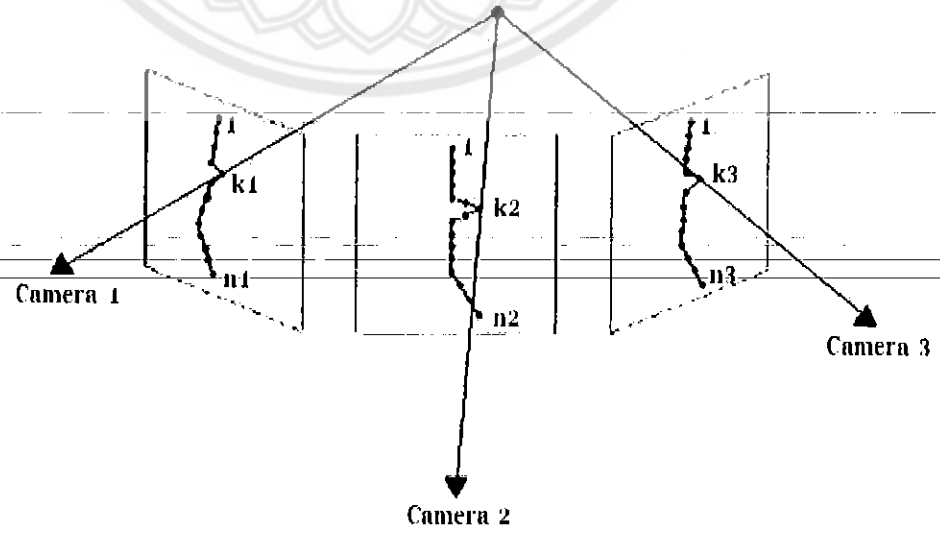
1. แบ่งเส้นinput ซึ่งคือ polyline สองมิติทั้งหมดไปเป็นส่วนเล็ก ๆ ท้ายสุดจะได้ polyline สองมิติที่มีจุดจำนวนมาก



**รูปที่ 2.12** Input polyline (ด้านบน) และ splitted polyline (ด้านล่าง) [1]

หลังจากที่แบ่ง polyline เส้นแรกจะได้ $n_1$ จุดที่มีดัชนี $1..n_1$ polyline เส้นที่สองจะแบ่งได้ $n_2$ จุดและดัชนี $1..n_2$ จุด ส่วนเส้นที่สามจะได้ $n_3$ จุดและดัชนี $1..n_3$ จุด

2. ความสอดคล้องกันของระหว่างจุดของ polylines สามารถอธิบายได้จากเวคเตอร์สามมิติของดัชนี ตัวอย่างเช่น เวคเตอร์ $\{k_1, k_2, k_3\}$ อธิบายว่าจุดสามมิติของเส้นขอบสามมิติสามารถได้มาจากการ triangulation ของจุดสามจุดบนรูปภาพเช่น จุดจากดัชนี $k_1$ จาก polyline เส้นแรก จุดจากดัชนี $k_2$ จาก polyline เส้นที่สองและจุดจากดัชนี $k_3$ จาก polyline เส้นที่สาม




**รูปที่ 2. 13** จุดที่ตรงกันของทั้งสามภาพ [1]

เราใช้การลดความผิดพลาดของการฉายภาพซ้ำ ด้วยอัลกอริทึม Minimization of reprojection error สำหรับ triangulation

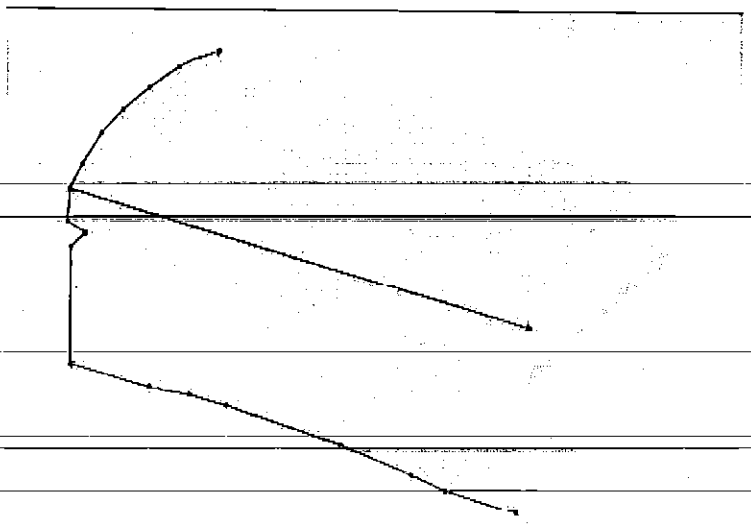
3. เริ่มต้นจากจุดแรก จุดที่ตรงกันสำหรับด้านหนึ่งของเส้นกำหนดให้เป็น {1, 1, 1} จาก $k_1 = 1, k_2 = 1, k_3 = 1$ จุดเหล่านี้เป็นจุดแรกของผลลัพท์ของเส้นขอบสามมิติ
4. จุดที่ตรงกันสำหรับเวคเตอร์
   $\{k_1, k_2, k_3 + 1\}, \{k_1, k_2 + 1, k_3\}, \{k_1, k_2 + 1, k_3 + 1\}, \{k_1 + 1, k_2, k_3\}, \{k_1 + 1, k_2, k_3 + 1\},$
   $\{k_1 + 1, k_2 + 1, k_3\}, \{k_1 + 1, k_2 + 1, k_3 + 1\}$
   สำหรับแต่ละจุดที่ลดความผิดพลาดของการฉายภาพซ้ำแล้วจะเป็นผลลัพท์ของเส้นขอบสามมิติ ทำซ้ำในเวคเตอร์ถัดไป
5. ทำซ้ำตั้งแต่ขั้นตอนที่ 4 จนกระทั่งดึงดัชนีสุดท้าย $-\{n_1, n_2, n_3\}$
6. ผลลัพท์ของ polyline สามมิติจะได้จุดเป็นจำนวนมาก ทำการลดจุดเหล่านี้จากกระบวนการดังต่อไปนี้

เลือกเฉพาะจุดที่ตรงกับ polyline ใน original point (จุดต้นฉบับ) ก่อนที่จะทำการแบ่งเส้นระหว่างในขั้นตอนที่ 1 ได้ทำการเก็บดัชนีของ original point (จุดต้นฉบับ) ไว้ก่อนที่จะทำการแบ่งเส้น ถ้าดัชนีของเวคเตอร์สามมิติตรงกับจุดในผลลัพท์ของ polyline สามมิติที่อยู่ในเซตของดัชนีของต้นฉบับ จะเก็บจุดนั้นไว้นอกนั้นจะถูกนำออกไปจาก polyline

ถ้าอะเรย์ของเวคเตอร์สามมิติตรงกับ polyline ในขั้นตอนก่อนหน้านี้มีมากกว่าหนึ่งเวคเตอร์ที่มีดัชนีตรงกับเซตของดัชนีต้นฉบับ ตัวอย่างเช่น $\{k_1, k_2, k_3\}$ และ $\{k_1, k_4, k_5\}$ โดยที่ $k_1$ อยู่ในเซตของดัชนีต้นฉบับ ดังนั้นจึงมีเพียงจุดเดียวที่ตรง บรรทัดฐานแรกจึงพิจารณาจำนวนที่อยู่ในเซตที่ตรงกับเซตของต้นฉบับ ตัวอย่างที่สอง ถ้า $k_2, k_3$ และ $k_4$ ไม่อยู่ในเซตของดัชนีต้นฉบับ และ $k_5$ อยู่ในเซตของดัชนีของต้นฉบับที่ตรงกับ polyline เส้นที่สาม จากนั้นจุดที่ตรงกับ $\{k_1, k_2, k_3\}$ จะไม่เก็บไว้ส่วนจุดที่ตรงกับ $\{k_1, k_4, k_5\}$ จะเก็บไว้ ถ้าเวคเตอร์ทั้งหมดตรงกับจำนวนของดัชนีของต้นฉบับแล้ว จุดที่คำนวนได้ reprojection error น้อยที่สุดจะถูกเก็บไว้และที่เหลือไม่นำออกจากเซตผลลัพท์ของ polyline สามมิติ

### 2.8.2 การสร้างเส้นขอบที่มองไม่เห็น (Missing Edges Construction)

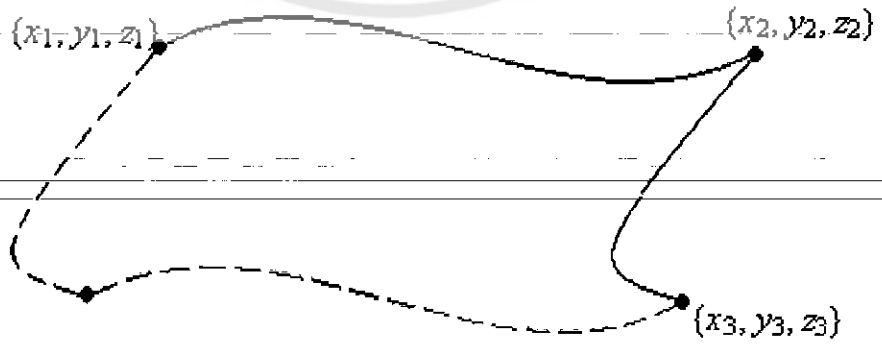
จากขั้นตอนที่ผ่านมา จะได้เส้นขอบสามมิติที่ตรงกับเส้นที่ projection สองมิติมาจากรูปภาพ



รูปที่ 2.14 เส้นขอบสามมิติที่ได้จากขั้นตอนที่ผ่านมา [1]

ถ้าผู้ใช้วาดเพียงเส้นที่จำเป็น เราต้องสร้างเส้นที่มองไม่หายไปออกมา จะวิเคราะห์จากพื้นผิวแต่ละด้านที่มีสองเส้นที่ไม่ขนานกันและมีสามเส้นที่สร้างขึ้นแล้วเท่านั้น โดยจะวิเคราะห์พื้นผิวทุก ๆ ด้านจนกว่าจะหาเส้นที่มองไม่เห็นจนครบทุกด้าน หรือจนกว่าจะหาเส้นต่อไปไม่ได้แล้ว

ถ้าพื้นผิวนั้นมีสองเส้นที่ติดกันอยู่ เราจะได้เส้นที่มองไม่เห็นสองเส้นจากการเลื่อนจุดต่อจุด สมมติให้จุดปลายของเส้นแรกเป็น $\{x_1, y_1, z_1\}$ และ $\{x_2, y_2, z_2\}$ และจุดของเส้นที่สองคือ $\{x_2, y_2, z_2\}$ และ $\{x_3, y_3, z_3\}$ เส้นที่สามจะได้มาจากจากการบวก $\{x_3 - x_2, y_3 - y_2, z_3 - z_2\}$ เข้ากับพิกัดของจุดบนเส้นแรก ส่วนเส้นที่สี่จะหาได้จากการบวก $\{x_1 - x_2, y_1 - y_2, z_1 - z_2\}$ เข้ากับพิกัดของจุดบนเส้นที่สอง




**รูปที่ 2.15** สร้างเส้นที่มองไม่เห็นจากสองเส้นที่ติดกัน [1]

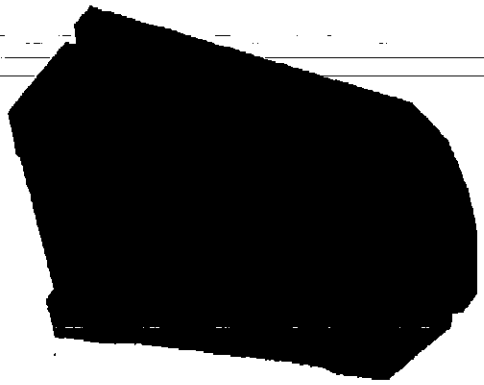
ถ้าพื้นผิวนั้นมีเส้นที่สร้างขึ้นแล้วสามเส้น เส้นที่มองไม่เห็นจะหาได้จากการเลื่อนจุดและการสอดแทรกของเส้นขอบกลาง สำหรับตัวอย่าง จุดปลายของเส้นแรกคือ $\{x_1, y_1, z_1\}$ และ $\{x_2, y_2, z_2\}$ จุดปลายของเส้นที่สองคือ $\{x_2, y_2, z_2\}$ และ $\{x_3, y_3, z_3\}$ และจุดปลายของเส้นที่สามคือ $\{x_3, y_3, z_3\}$ และ $\{x_4, y_4, z_4\}$ เรากำหนดให้เส้นที่สองมี $N$ จุด และมีดัชนีคือ 1..$N$ ดังนั้นเส้นที่มองไม่เห็นจะมี $N$ จุดและมีดัชนีคือ $i$ จะได้มาจากการบวก $(\{x_1 - x_2, y_1 - y_2, z_1 - z_2\} * i + \{x_4 - x_3, y_4 - y_3, z_4 - z_3\} * (N - i))$ เข้ากับจุดของเส้นที่สองที่ดัชนี $i$


**รูปที่ 2.16** สร้างเส้นที่หายไปจากเส้นสามเส้น [1]

**2.8.3 การสร้างพื้นผิว (Face triangulation)**

หลังจากที่สร้างเส้นทุกเส้นออกมาเป็นแบบจำลองได้แล้ว สำหรับพื้นผิวทั้งหกด้านนั้นสามารถสร้างได้ด้วยอัลกอริทึม discrete coons patches [5]

เนื่องจากจุดในเส้นขอบสามมิติมีจำนวนน้อย (จากขั้นตอนที่ 6 ในหัวข้อ 3d edge reconstruction) และการร่างแบบที่ไม่แม่นยำของผู้ใช้พื้นผิวของแบบจำลองที่สร้างขึ้นจะขรุขระ ดังนั้นจึงเพิ่ม mesh smoothing เข้าไป



**รูปที่ 2.17** render 3d model [1]

## 2.9 รูปแบบของสีระบบ RGB

RGB เป็นระบบสีที่แบ่งข้อมูลของสีออกเป็น R (แดง) G (เขียว) B (น้ำเงิน) ซึ่งเป็นสามสีหลักของแสงสี ส่วน YUV กับ YCrCb จะแบ่งข้อมูลของสีออกในลักษณะเดียวกับกระบวนการรับรู้ของประสาทตาคน นั่นคือแบ่งเป็น ส่วนของความเข้มแสงที่เป็นขาวดำและส่วนของแสงสีต่างๆ โดย Y จะเก็บข้อมูลของความเข้มแสง เรียกว่า Luminance และ U, V กับ Cr,Cb จะเก็บข้อมูลของแสงสี เรียกว่า Chrominance YUV และ YCrCb จะเป็นระบบของสีที่คล้ายๆ กัน ซึ่งสามารถแปลงไปเป็น RGB ได้ด้วยสมการเฉพาะแบบ

ระบบสี RGB เป็นระบบสี 24 บิต ที่สามารถแสดงสีได้จำนวน 256 สี ที่เราเห็นบนหน้าจอซึ่งระบบสีในหน้าจออิเล็กทรอนิกส์ทั่วไป มีระบบการแสดงผลผ่านหลอดลำแสงที่เรียกว่า CRT (Cathode ray tube) โดยมีลักษณะระบบสีแบบบวกที่ใช้บนหน้าจออิเล็กทรอนิกส์ แสดงออกมาโดยอาศัยการผสมของแสงสีแดง สีเขียว และสีน้ำเงิน จากการรวมสีของแม่สีหลักนี้จะทำให้เกิดแสงสีขาวซึ่งในลักษณะนี้จะแตกต่างจากการผสมสีของแม่สี

ในจอคอมพิวเตอร์ถูกกำเนิดสีขึ้นประกอบด้วยแสงสีแดง สีเขียว และสีน้ำเงิน ที่เป็นแม่สีหลักทั้งหมด โดยมีลักษณะเป็นจุดเล็ก ๆ ในหน้าจอจนไม่สามารถมองเห็นด้วยตาเปล่าได้ ตาเราจะมองเห็นเป็นสีที่ถูกผสมเป็นเนื้อสีเดียวกันแล้ว ถ้าจุดสีแดงและสีเขียวกำลังส่องแสง 100% และสีน้ำเงินไม่แสดงเลยจะเห็นเป็นสีเหลืองบริสุทธิ์ แต่ละจุดทั้ง 3 จุด สามารถปรับค่าใด ๆ ได้จาก 0 ถึง 255 ได้ ซึ่งหมายถึงจำนวนรวมของสีที่เป็นไปได้ คือ 256*256*256 = 16,777,216 สี ในจำนวนแรกนั้นเป็นสีแดง(R) จำนวนที่สองคือสีเขียว (G) และจำนวนสุดท้ายนั้นคือสีน้ำเงิน (B) หรือเรียกว่าโมเดลแบบ RGB ดังนั้นถ้าจะกำหนดสีใด ๆ ก็ตามจะต้องระบุจำนวนตัวเลขลงไป เช่น สีน้ำเงินบริสุทธิ์คือ 0,0.255 สีน้ำเงินเข้ม 0,0,100 ถ้าต้องการสีขาวก็ต้องกำหนดเป็น 255,255,255 ส่วนสีดำก็กำหนดเป็น 0,0,0 เป็นต้น

ในการใช้งานระบบสี RGB ยังมีการสร้างมาตรฐานที่แตกต่างกันออกไปที่นิยมใช้งานได้แต่ $RGB_{CIE}$ และ $RGB_{NTSC}$

**ระบบสีแบบ RGB ของ CIE**

เป็นระบบสีที่พัฒนาขึ้นโดย CIE (Commission International l Éclair age) ซึ่งอ้างอิงสีด้วยสีแดงที่ 700 nm สีเขียวเท่ากับ 546.1 nm และสีน้ำเงิน 435.8 nm

**ระบบสีแบบ RGB ของ NTSC**

เป็นระบบที่พัฒนาโดย NTSC (National Television System Committee) เพื่อใช้สำหรับการแสดงภาพของจอภาพแบบ CRT เป็นมาตรฐานสำหรับผู้ผลิตแบบ CRT ให้มีลักษณะเดียวกัน

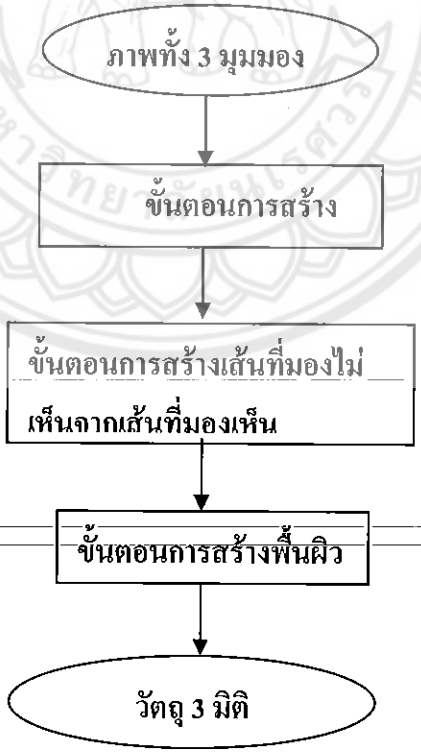
# บทที่ 3
# วิธีการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนการทำงานเพื่อให้ได้มาซึ่งโปรแกรมที่สามารถสร้างภาพ 3 มิติจากภาพ 2 มิติ ซึ่งมีขั้นตอนหลัก ๆ ได้แก่ การเริ่มต้นจากการศึกษาและรวบรวมข้อมูล ขั้นตอนศึกษาอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets ขั้นตอนสุดท้ายคือการเขียนโปรแกรมสร้างภาพ 3 มิติ จากภาพ 2 มิติ

จากการศึกษาอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets ทำให้ทราบถึงขั้นตอนต่าง ๆ โดยเริ่มจากการสร้างเส้น 3 มิติจากเส้นที่มองเห็น จากนั้นจึงสร้างเส้นที่มองไม่เห็นจากเส้นที่มองเห็น สุดท้ายคือการสร้างพื้นผิวในแต่ละด้าน

## 3.1 องค์ประกอบของโปรแกรมทั้งหมด

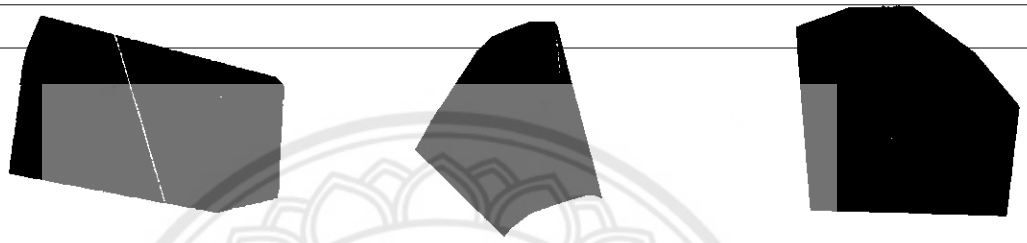
จากการศึกษาและรวบรวมข้อมูลตามอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets จึงได้ภาพรวมของโปรแกรมทั้งหมดดังรูปที่ 3.1




**รูปที่ 3.1** แสดงองค์ประกอบของโปรแกรมทั้งหมด

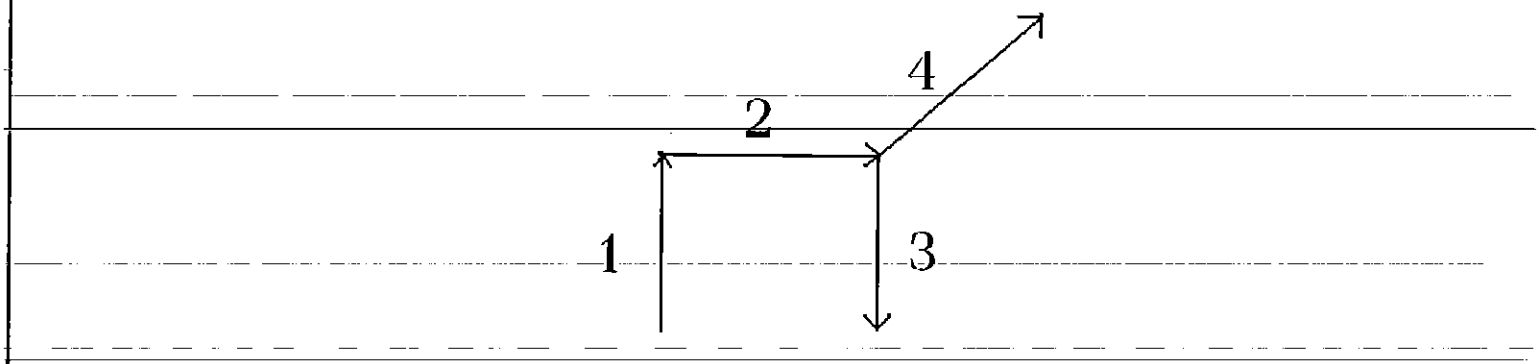
### 3.2 ขั้นตอนการสร้างเส้น 3 มิติ

ขั้นตอนแรกของการสร้างภาพ 3 มิตินั้น คือการสร้างเส้น 3 มิติที่ตรงกับอินพุดจากเส้น 2 มิติโดยการใช้รูปภาพ 3 รูป จาก 3 มุมมอง และหาความสัมพันธ์ระหว่างจุดที่ตรงกันของทั้ง 3 รูป ซึ่งวิธีการที่ผู้ดำเนินโครงงานได้จัดทำขึ้นคือการใช้โปรแกรม Java กำหนดจุดลงไปในภาพแต่ละมุมมองโดยตัวผู้ใช้ ซึ่งเมื่อเสร็จสิ้นการกำหนดจุดลงไปในภาพแล้ว ค่าของแต่ละจุดจะเก็บไว้ในไฟล์ .txt โดยลักษณะของวัตถุที่นำมาใช้จะต้องเป็นวัตถุที่มีลักษณะคล้ายกล่องดังรูปที่ 3.2



**รูปที่ 3.2** แสดงตัวอย่างภาพวัตถุที่นำมาใช้ในโครงงาน

**รูปที่ 3.3** แสดงตัวอย่างมุมมองที่นำมาใช้ทั้ง 3 มุมมอง




**รูปที่ 3.4** แสดงการเรียงการกำหนดจุดบนภาพแต่ละมุมมอง

**3.2.1 หลักการกำหนดจุดลงไปในภาพแต่ละมุมมองของผู้ดำเนินโครงงาน**

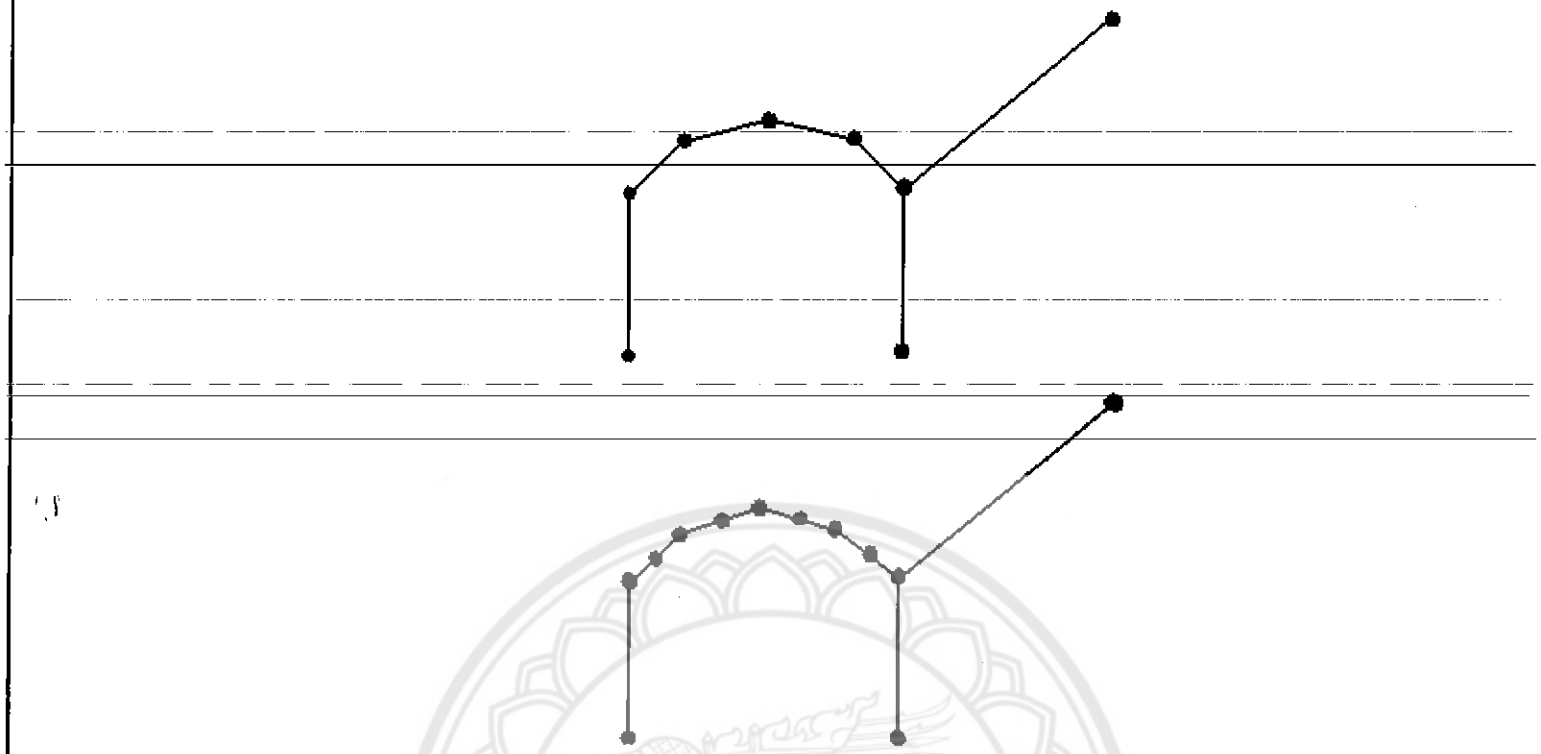
1. กำหนดจุดตามลำดับดังรูปที่ 3.4 จากปลายด้านหนึ่งไปยังปลายอีกด้านหนึ่ง
2. ถ้าเป็นเส้นโค้งหรือเส้นเว้า ให้กำหนดจุดไปตามเส้นขอบโดยมองว่าเส้นโค้งเกิดจากเส้นตรงเส้นเล็ก ๆ หลายเส้นที่ต่อกัน ตัวอย่างเช่น


3. จำนวนจุดในแต่ละด้านของทั้ง 3 มุมมองจะต้องมีจำนวนจุดเท่ากัน
4. ให้ผู้กำหนดจุดพิจารณาว่าเมื่อเลือกเส้นใดแล้วภาพจะผิดเพี้ยนหรือไม่

**3.2.2 ขั้นตอนการสร้างเส้น 3 มิติ จากเส้น 2 มิติ**

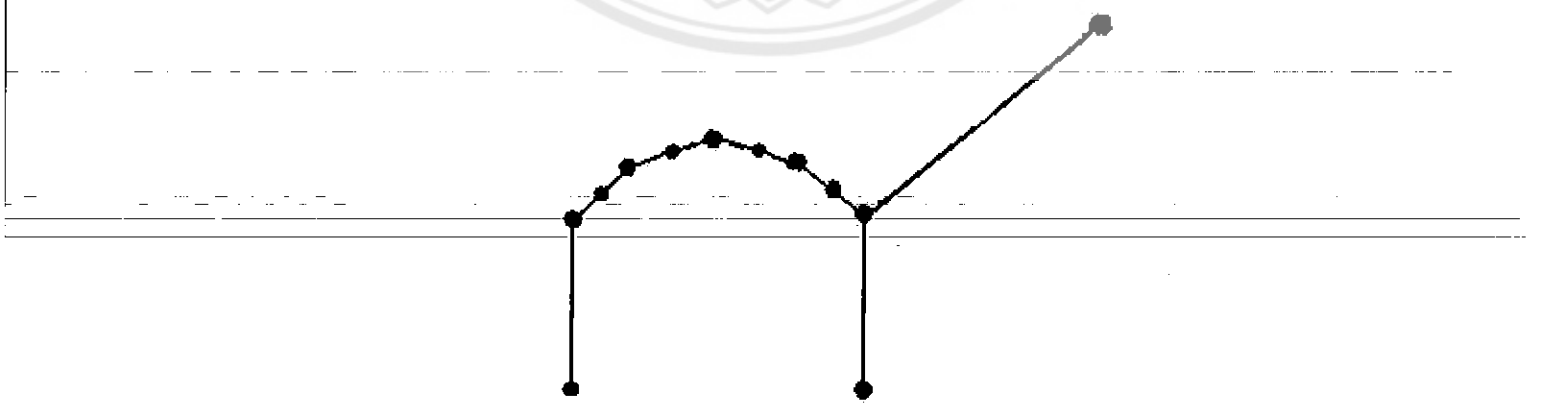
เมื่อทราบค่าตำแหน่งของจุดในแต่ละมุมมองแล้วผู้ดำเนินโครงงานได้ทำการแยกว่าจุดใดคือตำแหน่งใด ของเส้นในด้านไหน เพราะว่าจากไฟล์ .txt ทั้ง 3 ไฟล์จากทั้ง 3 มุมมองที่อ่านค่าเข้ามาจะเรียงกันแต่ไม่แบ่งแยกว่าค่าใดคือตำแหน่งใด จึงต้องการแยกจุดว่าค่าตำแหน่งของจุดนี้ เป็นค่าตำแหน่งที่อยู่ในเส้นด้านใด

จากนั้นเมื่อทำการแยกกลุ่มของจุดว่าเป็นเส้นใดของมุมมองใดแล้ว จึงทำการ split (แตก) เส้นที่ได้แต่ละด้านออกเป็นเส้นเล็ก ๆ จึงได้จุดที่อยู่บนเส้นเพิ่มขึ้นโดยในเส้นที่มีลักษณะเป็นเส้นตรงอยู่แล้ว หรือเส้นที่เกิดจากจุดเพียงสองจุดจะไม่ทำการ split (แตก) ดังแสดงในรูปต่อไปนี้



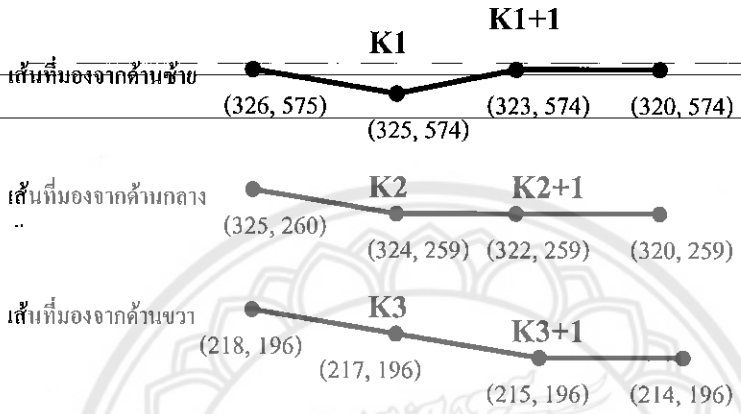
**รูปที่ 3.5** แสดงเส้นอินพุด (ด้านบน) และเมื่อ split เส้นแล้ว (ด้านล่าง)

จากรูปที่ 3.5 เมื่อ split เส้นในแต่ละด้านนั้นจะมีจุดอยู่ 2 กลุ่มซึ่งคือ original point (จุดที่ได้จากการกำหนดด้วยผู้ใช้) และ split point (จุดที่ได้จากการ interpolated (การแยก) จากโปรแกรม) วิธีการในโครงงานนี้จะนำค่าตำแหน่งของ original point (จุดที่ได้จากการกำหนดด้วยผู้ใช้) มาคิดหาเวคเตอร์ 2 มิติซึ่งเวคเตอร์ตัวนี้จะเป็นค่าตำแหน่งในระนาบ 3 มิติ



**รูปที่ 3.6** แสดงจุด original point (สีแดง) และ split point (สีดำ)

ส่วน split point (จุดที่ได้จากการ interpolated (การแยก) จากโปรแกรม) นั้นจะนำมาคิดหา error ที่น้อยที่สุดโดยนำ split point ทั้ง 4 เส้นจาก 3 มุมมองมาแยกออกเป็นเวคเตอร์ 7 เวคเตอร์ ดังนี้คือ *{k1, k2, k3+1},{k1, k2+1, k3}, {k1, k2+1, k3+1}, {k1+1, k2, k3}, {k1+1, k2, k3+1},{k1+1, k2+1, k3}, {k1+1, k2+1, k3+1}*




รูปที่ 3.7 แสดงตัวอย่างค่าตำแหน่งของเส้น 1 เส้นจาก 3 มุมมอง

| เวคเตอร์ | จุดที่มองจากมุมมองด้านซ้าย | จุดที่มองจากมุมมองตรงกลาง | จุดที่มองจากมุมมองด้านขวา |
|---|---|---|---|
| **vector1** = {k1, k2, k3+1} | (325, 574) | (324, 259) | (215, 196) |
| **Vector2** = {k1, k2+1, k3} | (325, 574) | (322, 259) | (217, 196) |
| **Vector3** = {k1, k2+1, k3+1} | (325, 574) | (322, 259) | (215, 196) |
| **Vector4** = {k1+1, k2, k3} | (323, 574) | (324, 259) | (217, 196) |
| **Vector5 = {k1+1, k2, k3+1}** | (323, 574) | (324, 259) | (215, 196) |
| **Vector6 = {k1+1, k2+1, k3}** | (323, 574) | (322, 259) | (217, 196) |
| **Vector7 = {k1+1, k2+1, k3+1}** | (323, 574) | (322, 259) | (215, 196) |

**ตารางที่ 3.1** แสดงตัวอย่างค่าตำแหน่งของเวคเตอร์ทั้ง 7 เวคเตอร์ที่ได้จาก split point (จุดที่ได้จากการ interpolated (การแยก) จากโปรแกรม) 1 จุด

จากนั้นจึงนำแต่ละเวคเตอร์มาคำนวนหา error จากวิธีการในหัวข้อ 2.4 จากบทที่ 2 โดยคิด error ทีละคู่ คู่แรกคือมุมมองด้านซ้ายกับมุมมองตรงกลาง และคู่ที่สองคือมุมมองด้านขวากับมุมมองตรงกลาง

ตัวอย่างการหา error ที่น้อยที่สุดระหว่างจุด (325, 574) และจุด (324, 259) ที่เกิดจากมุมมองด้านซ้ายและมุมมองตรงกลางตามลำดับ

1. หาค่า $K$ ของจุด (325, 574) และ $K'$ ของจุด (324, 259) จากสมการ

$$K = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

จะได้

$$K = \begin{pmatrix} 1 & 0 & 325 \\ 0 & 1 & 574 \\ 0 & 0 & 1 \end{pmatrix}$$

$$K' = \begin{pmatrix} 1 & 0 & 324 \\ 0 & 1 & 259 \\ 0 & 0 & 1 \end{pmatrix}$$

2. หาค่า Translation โดยคิดเป็นแบบ cross product , $[t]_X$

คิดจาก

$$[t]_X = \begin{bmatrix} 0 & -t_Z & t_Y \\ t_Z & 0 & -t_X \\ -t_Y & t_X & 0 \end{bmatrix}$$

โดย

$t_X$ – คือ ค่า translation ในแนวแกน X

$t_Y$ คือ ค่า translation ในแนวแกน Y

$tz$ คือ ค่า translation ในแนวแกน Z

จะได้

$$[t]_X = \begin{bmatrix} 0 & 0.113 & 0 \\ -0.113 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

3. หาค่า rotation โดยในโครงงานนี้หาค่า rotation เฉพาะในแนวแกน Z

คิดจาก

$$R = \begin{bmatrix} \cos(roll) & -\sin(roll) & 0 \\ \sin(roll) & \cos(roll) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

จะได้

$$R = \begin{bmatrix} \cos(75.30) & -\sin(75.30) & 0 \\ \sin(75.30) & \cos(75.30) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.2538 & -0.9673 & 0 \\ 0.9673 & 2.2538 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4. หาค่า Fundamental matrix ,F

คิดจาก

$$F = K'^{-\mathrm{T}}[t]_x RK^{-1}$$

จะได้

$$F = \begin{pmatrix} 1 & 0 & 324 \\ 0 & 1 & 259 \\ 0 & 0 & 1 \end{pmatrix}^{-\mathrm{T}} \begin{bmatrix} 0 & 0.113 & 0 \\ -0.113 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.2538 & -0.9673 & 0 \\ 0.9673 & 2.2538 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 & 0 & 325 \\ 0 & 1 & 574 \\ 0 & 0 & 1 \end{pmatrix}^{-1}$$

$$F = \begin{bmatrix} -0.0111 & 0.1125 & -33.5654 \\ -0.1125 & -0.0111 & 32.7163 \\ 42.9087 & -60.9479 & 8.6338\mathrm{e}+003 \end{bmatrix}$$

5. หาค่า error ที่น้อยที่สุด

คิดจาก

$$f(t) = t((at+b)^2 + \hat{f}^2(ct+d)^2)^2 - (ad-bc)(1+f^2t^2)^2(at+b)(ct+d)$$

$$= 0$$

ในที่นี้ t หมายถึงค่า error

จะได้

$$f(t) = t((-0.0111t + 32.7163)^2 + \hat{f}^2(-60.9479t + 8.6338e + 003)^2)^2$$
$$-((-0.0111)(8.6338e + 003)$$
$$-(32.7163)(-60.9479))(1 + f^2\hat{t}^2)^{-2}((-0.0111)t + 32.7163)((-60.9479)t + (8.6338e + 003$$
$$= 0$$

จากสมการข้างต้นจะได้ค่า error ที่น้อยที่สุดคือ 7.2265e-005

จากตัวอย่างการคำนวนหาค่า error ที่น้อยที่สุดข้างต้นจึงสามารถคิดค่า error ที่น้อยที่สุดของเวคเตอร์ทั้ง 7 เวคเตอร์ได้ดังตารางต่อไปนี้

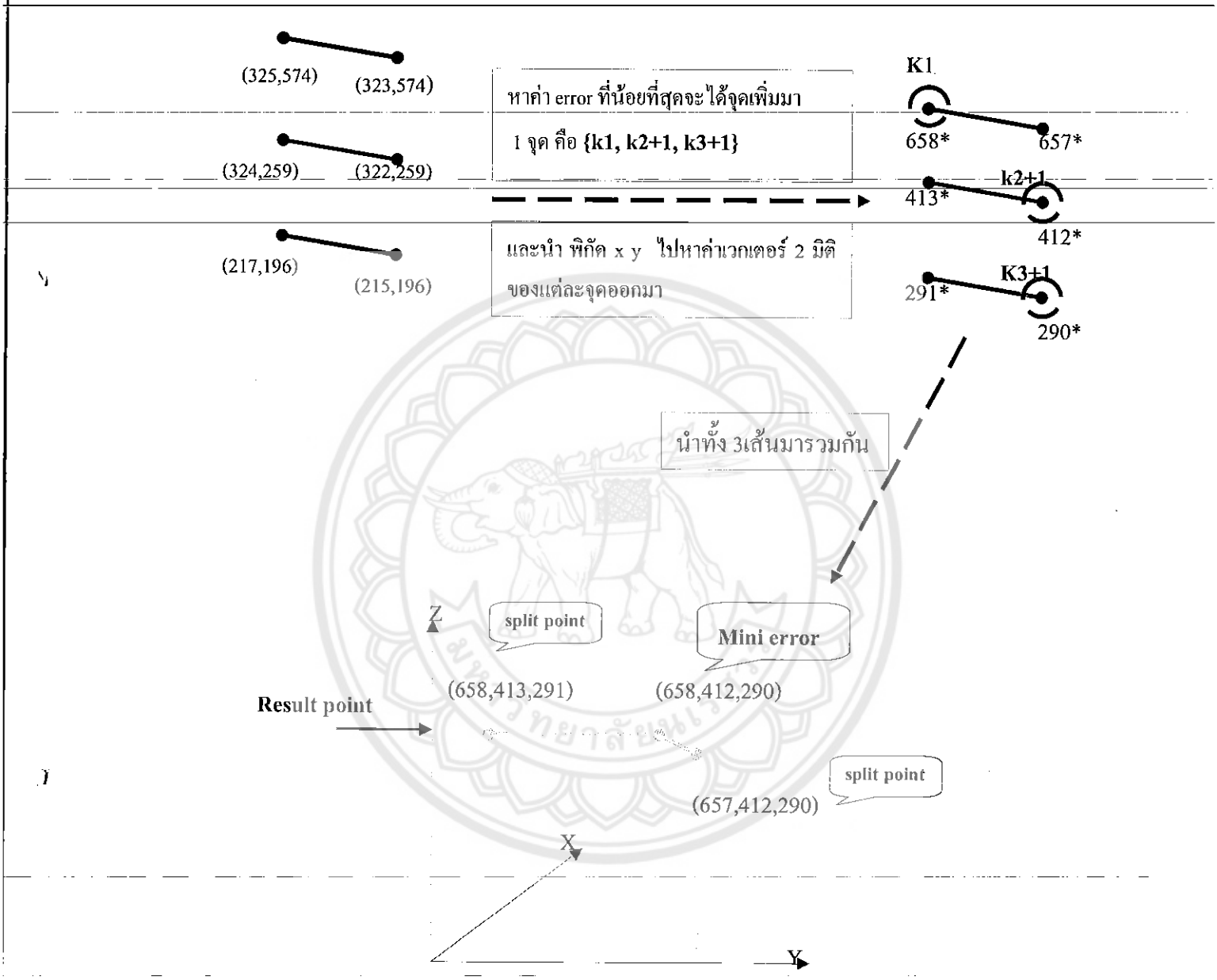
| เวคเตอร์ | error ระหว่างมุมมองด้านซ้ายและมุมมองตรงกลาง | error ระหว่างมุมมองด้านขวาและมุมมองตรงกลาง | error ที่น้อยที่สุดจาก 3 มุมมอง |
|---|---|---|---|
| **vector1** = {k1, k2, k3+1} | 7.2265e-005 | -8.7763e-004 | -8.7763e-004 |
| **Vector2** = {k1, k2+1, k3} | 7.2286e-005 | -8.7287e-004 | -8.7287e-004 |
| **Vector3** = {k1, k2+1, k3+1} | 7.2286e-005 | -8.7799e-004 | -8.7799e-004 |
| **Vector4** = {k1+1, k2, k3} | 7.2199e-005 | -8.7253e-004 | -8.7253e-004 |
| **Vector5** = {k1+1, k2, k3+1} | 7.2199e-005 | -8.7763e-004 | -8.7763e-004 |
| **Vector6** = {k1+1, k2+1, k3} | 7.2219e-005 | -8.7287e-004 | -8.7287e-004 |
| **Vector7** = {k1+1, k2+1, k3+1} | 7.2219e-005 | -8.7799e-004 | -8.7799e-004 |
| ค่า error ที่น้อยที่สุดจาก | | | -8.7799e-004 |

ตารางที่ 3.2 แสดงค่า error ของ 7 เวคเตอร์พร้อมทั้งค่า error ที่น้อยที่สุด

ดังนั้นแต่ละ split point จึงมี error 7 ค่า ต่อจากนั้นจึงนำค่า error 7 ค่านั้นมาเปรียบเทียบกัน เลือกตำแหน่งที่มีค่า error น้อยที่สุด ซึ่งในที่นี้มี 2 ค่าคือ Vector3 = {k1, k2+1, k3+1} และ Vector7 = {k1+1, k2+1, k3+1} ผู้จัดทำโครงงานได้เลือกเวคเตอร์ 3 เพราะว่าเวคเตอร์ 7 คือจุดที่ตรงกับ split point (จุดที่ได้จากการ interpolated (การแยก) จากโปรแกรม) ที่ตำแหน่งถัดไปจากที่เราพิจารณา

เมื่อเลือกเวคเตอร์ได้แล้วจะนำมาสร้าง result point (จุดที่รวมเอา split point และจุดที่มี error น้อยที่สุด) โดยลักษณะการเรียงค่าของ result point คือ ตำแหน่ง split point ที่พิจารณา, ตำแหน่งของเวคเตอร์ที่มีค่า error น้อยที่สุดของ split point ที่พิจารณา ตามด้วย split point จุดถัดไปที่จะพิจารณา ทำตามขั้นตอนที่กล่าวมาทั้งหมดจนถึง split point จุดสุดท้าย

นำ result point ไปหาค่าเวคเตอร์ 2 มิติซึ่งเวคเตอร์ตัวนี้จะเป็นค่าตำแหน่งในระนาบ 3 มิติ ตามขั้นตอนดังรูป

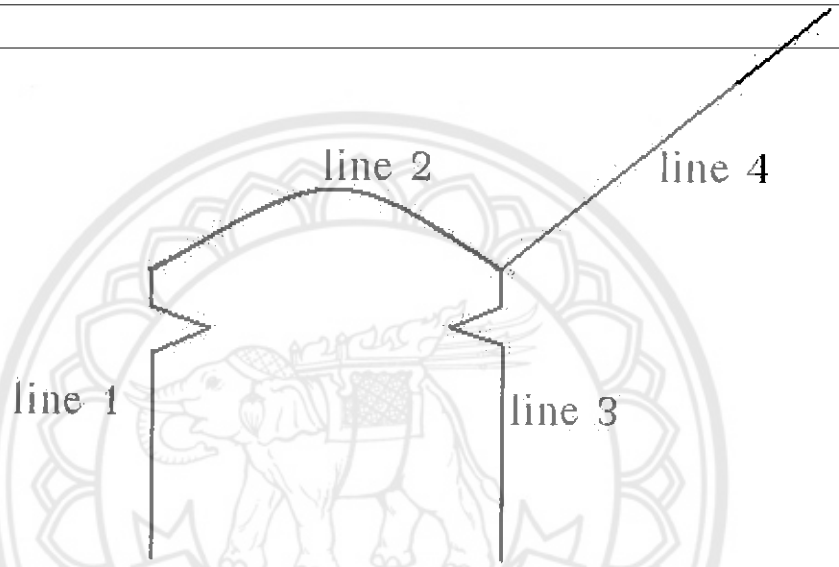



**รูปที่ 3.8** ภาพแสดงขั้นตอนการหา result point

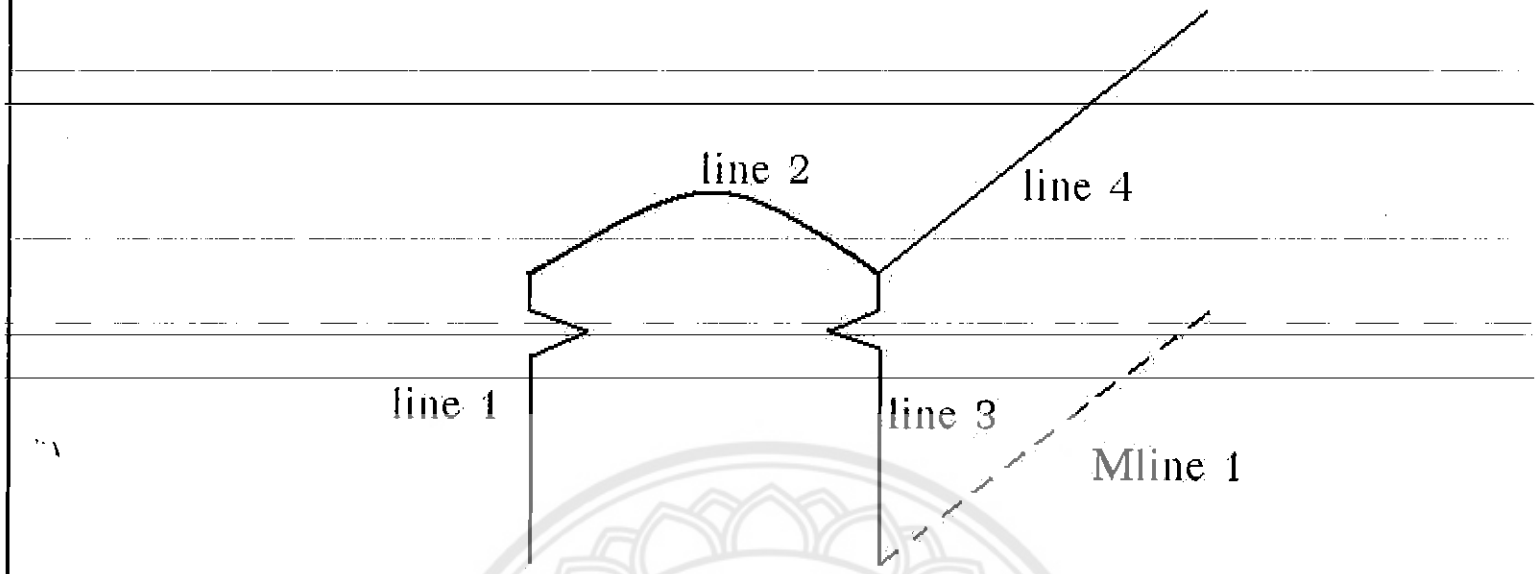
เนื่องจากจุด 3 มิติทั้งหมดที่ได้มานั้นมีจำนวนมากมาย จึงต้องลดความซ้ำซ้อนของจุดเหล่านี้ โดยทำตามวิธีในข้อ 6 ในหัวข้อย่อย 2.4.1 ในบทที่ 2

### 3.3 ขั้นตอนการสร้างเส้นที่มองไม่เห็นจากเส้นที่มองเห็น

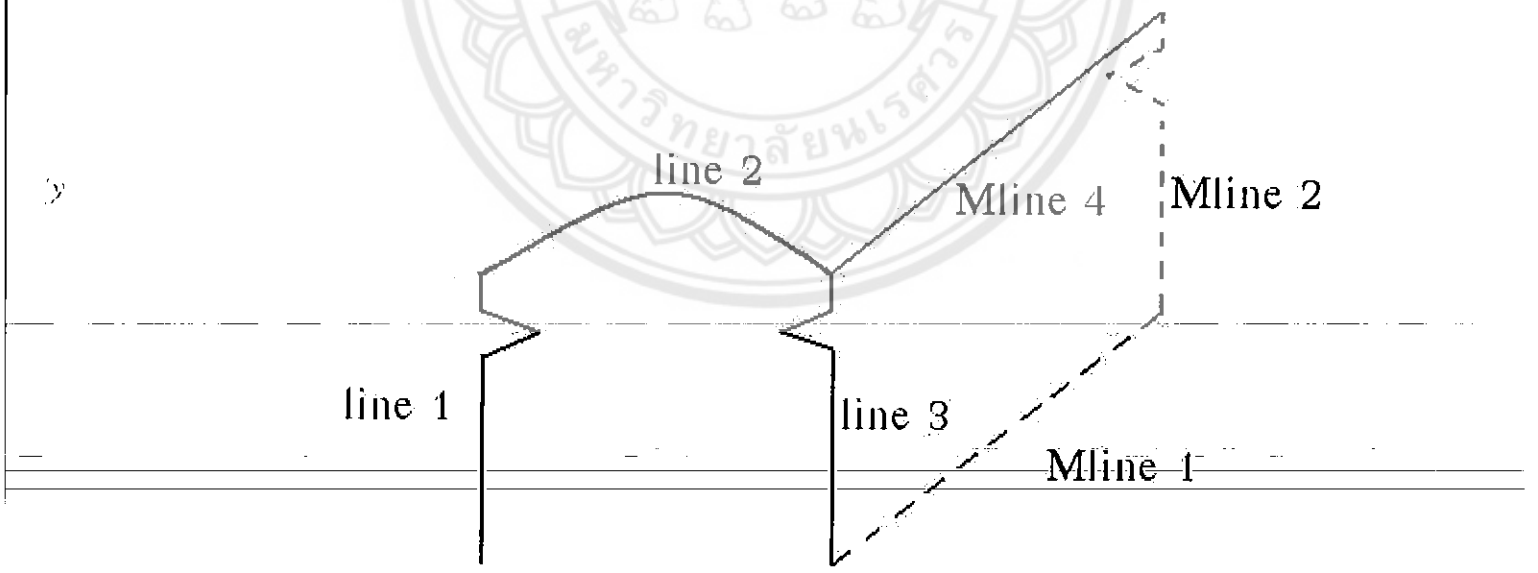
จากขั้นตอนที่ผ่านมานั้น ทำให้ได้เส้น 3 มิติที่ตรงกับภาพ 2 มิติ ซึ่งผ่านการหา error ที่น้อยที่สุดและทำการลดความซ้ำซ้อนของจุด 3 มิติมาแล้ว ผู้ดำเนินโครงงานจึงได้ทำขั้นตอนต่อไปคือการสร้างเส้นที่มองไม่เห็น (missing edges) จากเส้นที่มองเห็นตามหลักการของอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir ในหัวข้อ 2.4.2 ในบทที่ 2 โดยลำดับการสร้างเส้นตามรูปต่อไปนี้



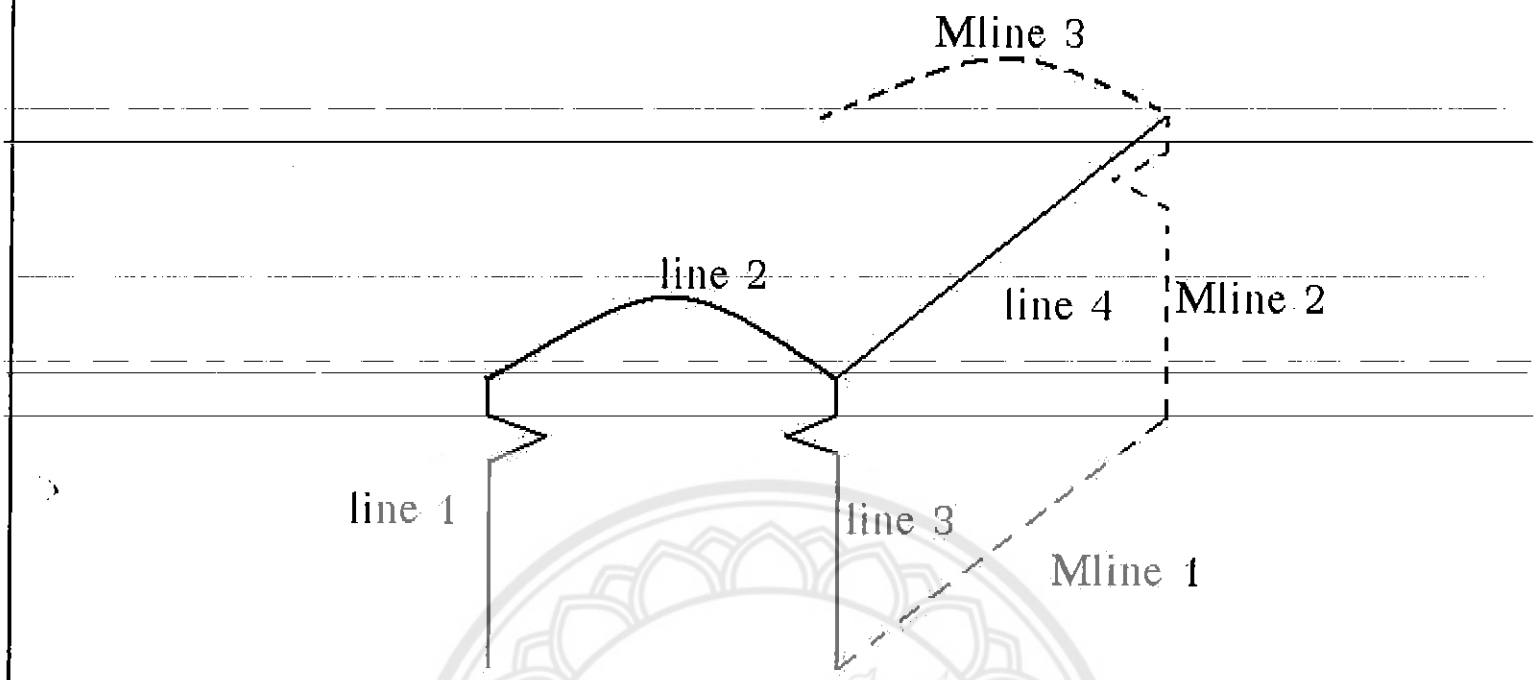

**รูปที่ 3.9** แสดงเส้น original ก่อนที่จะทำการสร้างเส้นที่มองไม่เห็น



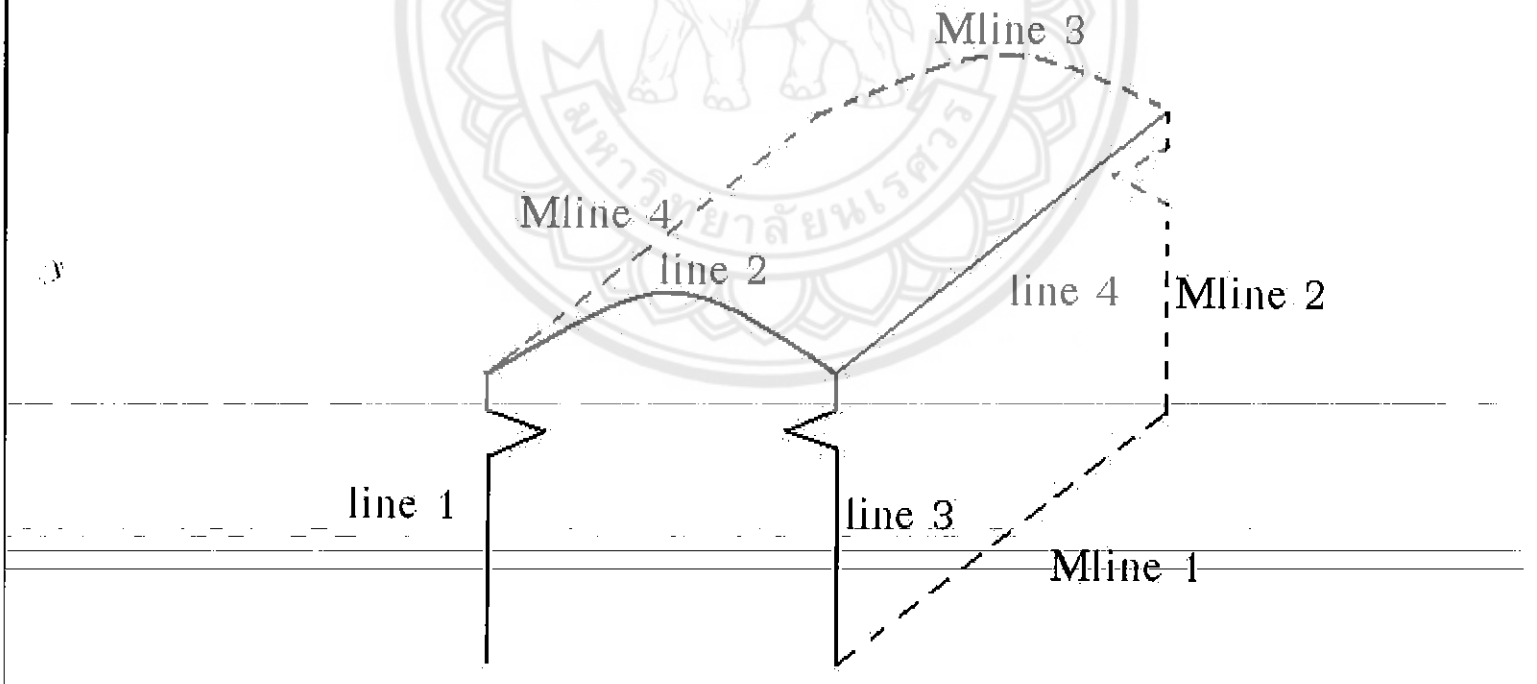

**รูปที่ 3.10** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้นลำดับที่ 1 (Mline 1)



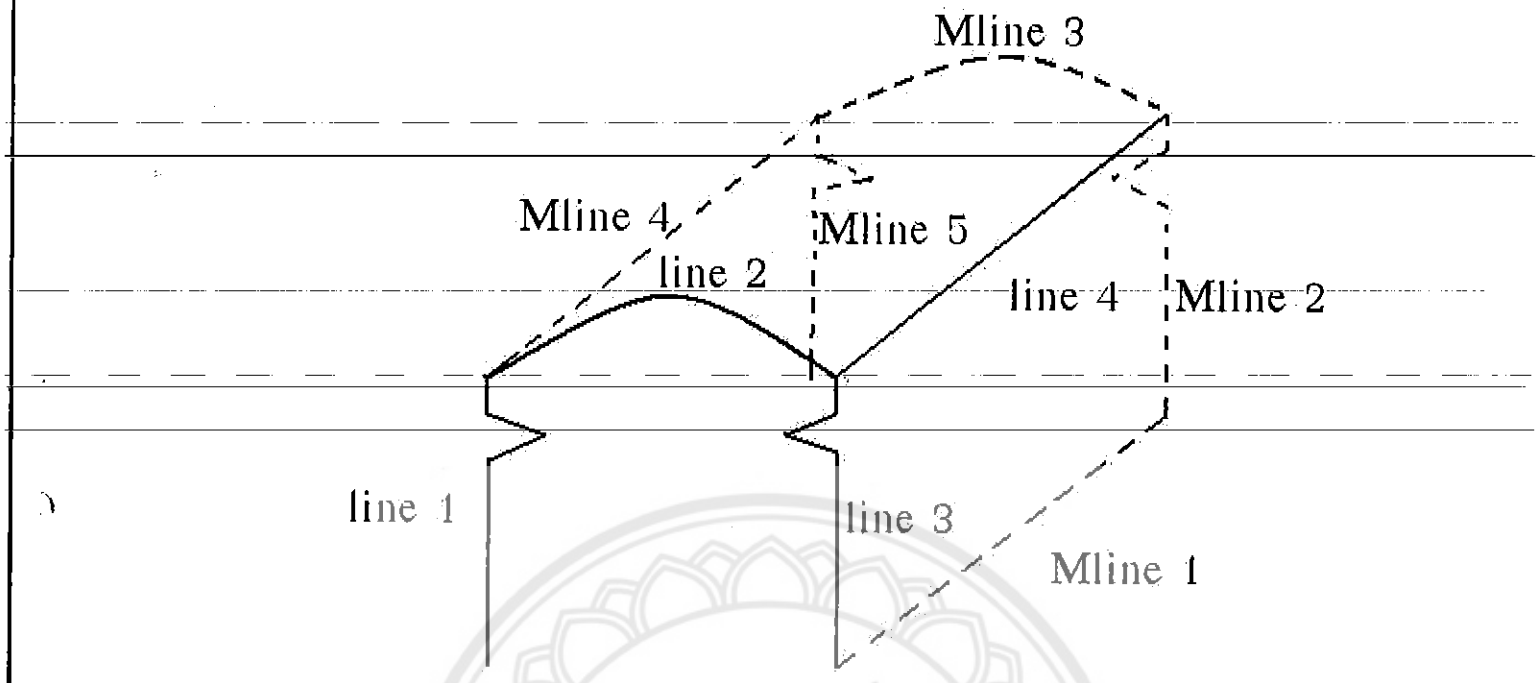

**รูปที่ 3.11** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้นลำดับที่ 2 (Mline 2)



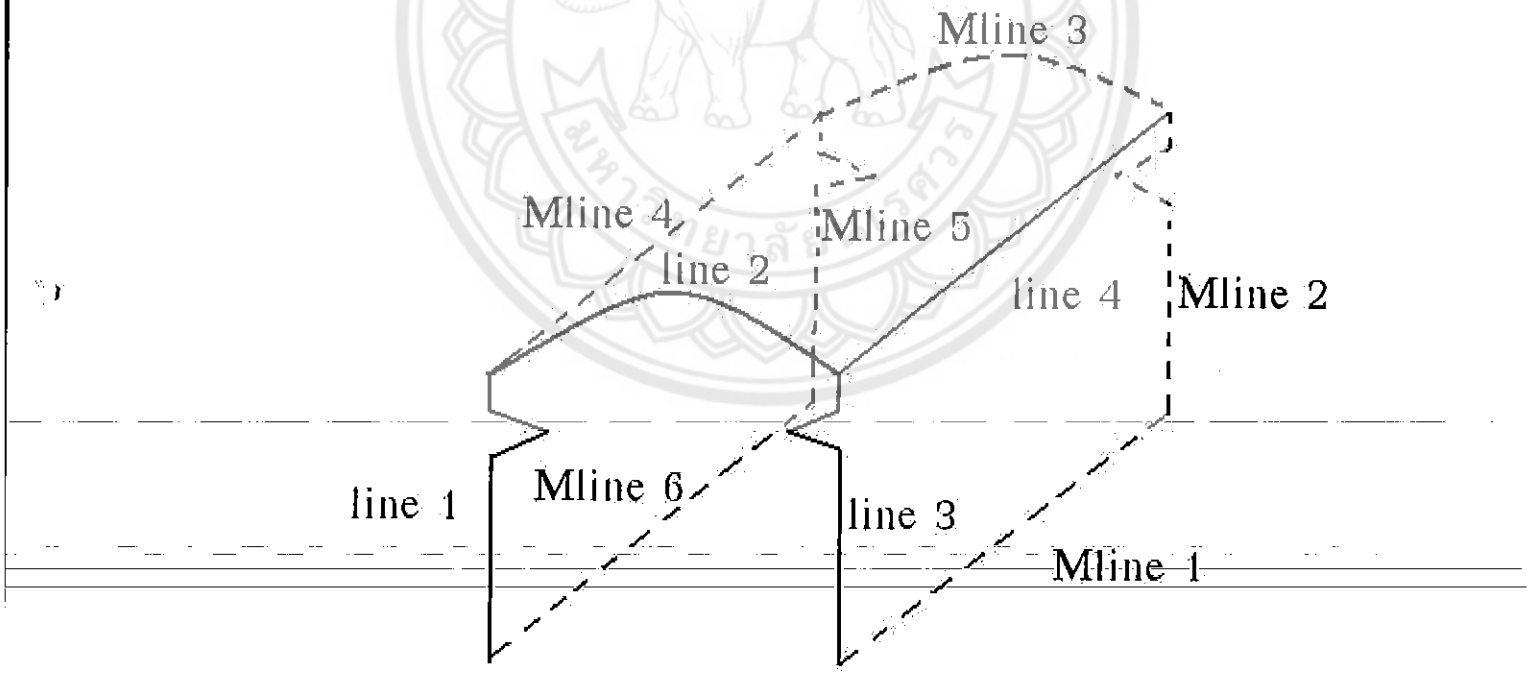

**รูปที่ 3.12** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้นลำดับที่ 3 (Mline 3)



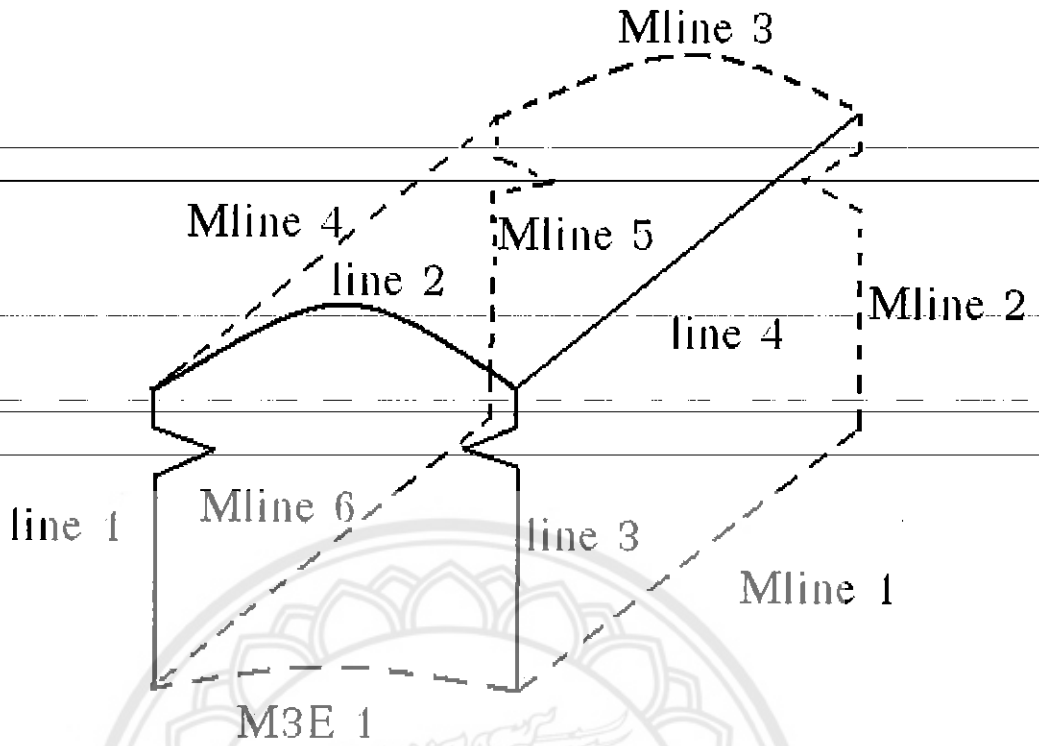

**รูปที่ 3.13** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้นลำดับที่ 4 (Mline 4)



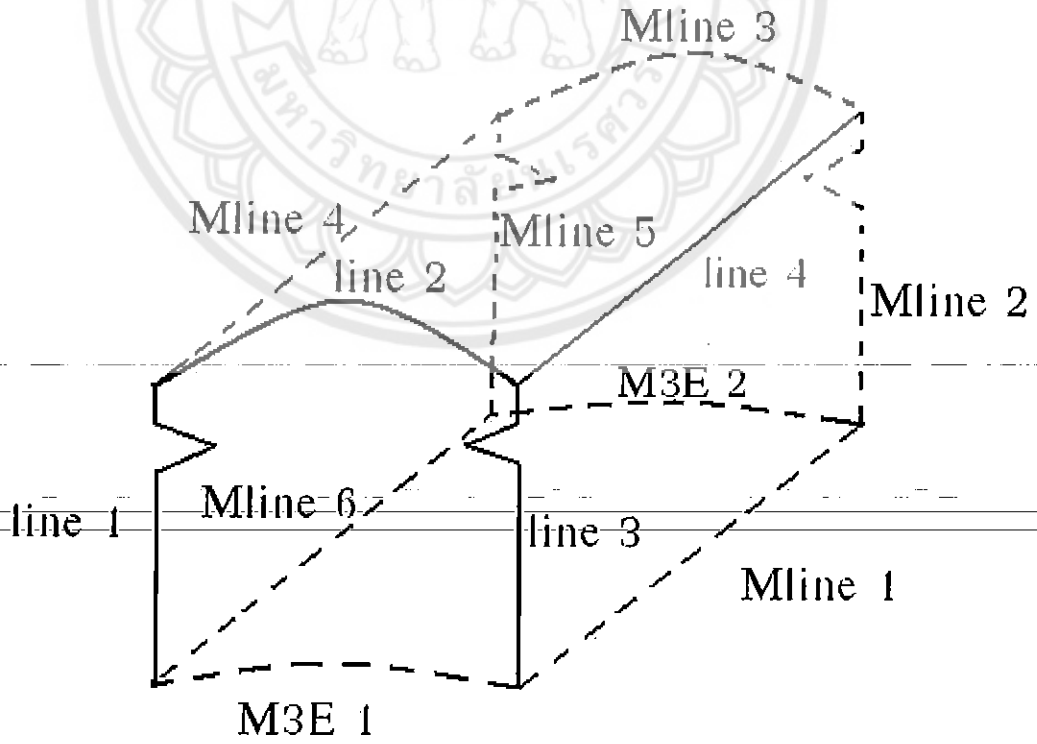

**รูปที่ 3.14** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้นลำดับที่ 5 (Mline 5)




**รูปที่ 3.15** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้นลำดับที่ 6 (Mline 6)




**รูปที่ 3.16** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 3 เส้นลำดับที่ 1 (M3E 1)




**รูปที่ 3.17** แสดงเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 3 เส้นลำดับที่ 2 (M3E 2)

ลำดับการสร้างเส้นที่มองไม่เห็นจากการใช้เส้นที่มองเห็น 2 เส้น (Mline) และ 3 เส้น (M3E) จากรูปที่ผ่านมา สามารถนำมาเขียนเป็นตารางได้ โดยตารางที่ 3.1 แสดงเส้นที่มองไม่เห็นที่เกิดจาก 2 เส้นที่ติดกัน และตารางที่ 3.2 แสดงเส้นที่มองไม่เห็นที่เกิดจาก 3 เส้นที่ติดกันตามวิธีการในหัวข้อย่อยที่ 2.4.2 ในบทที่ 2

| ลำดับที่ในการสร้าง | เส้น | เกิดจาก |
|---|---|---|
| 1 | Mline 1 | line 4 |
| 2 | Mline 2 | line 3 |
| 3 | Mline 3 | line 2 |
| 4 | Mline 4 | line 4 |
| 5 | Mline 5 | line1 |
| 6 | Mline 6 | Mline 4 |

**ตารางที่ 3. 3** ลำดับและที่มาของการสร้างเส้นที่มองไม่เห็นที่เกิดจากเส้นที่มองเห็น 2 เส้นที่ติดกัน

| ลำดับที่ในการสร้าง | เส้น | เกิดจาก |
|---|---|---|
| 1 | M3E 1 | line 1, line 2 และ line 3 |
| 2 | M3E 2 | Mline2, Mline 3 และ Mline 5 |

**ตารางที่ 3. 4** ลำดับและที่มาของการสร้างเส้นที่มองไม่เห็นที่เกิดจากเส้นที่มองเห็น 3 เส้นที่ติดกัน

## 3.4 ขั้นตอนการสร้างพื้นผิว

จากขั้นตอนที่ผ่านมาทั้งหมดนั้น ทั้งการสร้างจุดและเส้น 3 มิติ การหาค่า error ที่น้อยที่สุด การลดความซ้ำซ้อนของจุด จากนั้นจึงทำการหาเส้น 3 มิติที่หายไปทั้งหมดได้ครบแล้ว จะได้เส้นโครงสร้าง 3 มิติ ซึ่งแต่ละด้านนั้นยังไม่มีพื้นผิว ผู้ดำเนินโครงงานจึงทำการสร้างพื้นผิวโดยการกำหนดสีลงไปในแต่ละด้านของโครงสร้าง 3 มิติ

การหาสีพื้นผิวของวัตถุจะใช้ทฤษฎี automatic thresholding แยกวัตถุจากพื้นหลังออกมา โดยพื้นหลังของวัตถุจะต้องเป็นสีขาว จะได้ได้ตำแหน่งของวัตถุออกมา แล้วนำตำแหน่งของวัตถุที่ได้นั้นไปเปรียบเทียบกับภาพอินพุต แล้วจึงหาค่าเฉลี่ยของสีของภาพอินพุตนำไปใส่ในโครงสร้าง 3 มิติ จึงได้ภาพ 3 มิติที่สมบูรณ์

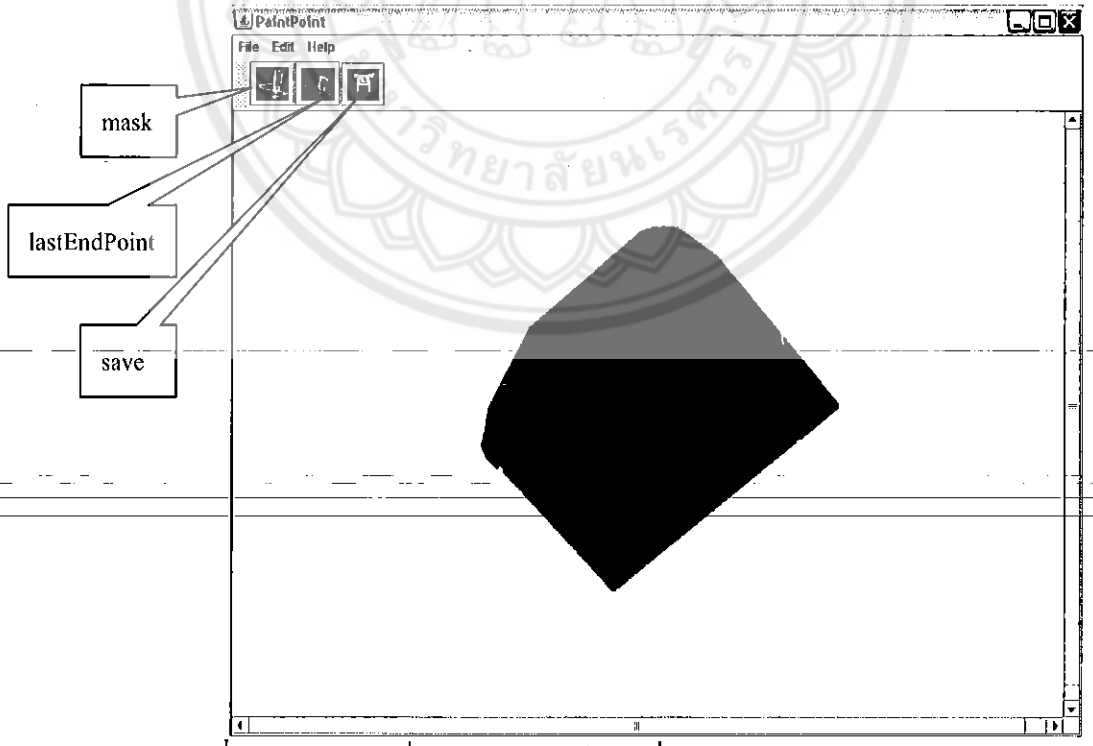
# บทที่ 4

# ผลการทดลอง

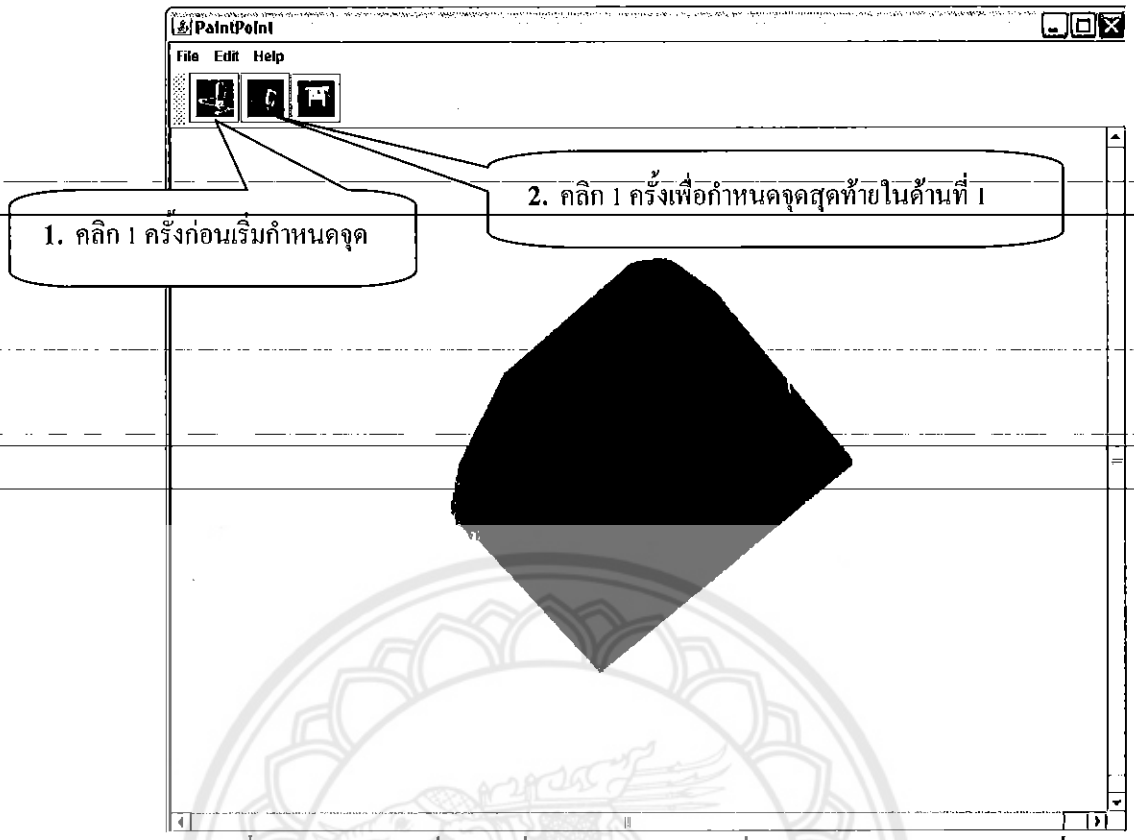
ในบทนี้จะกล่าวถึงการทดสอบโปรแกรมและผลการทดสอบโปรแกรมการสร้างภาพ 3 มิติจากภาพ 2 มิติ โดยขั้นตอนการทดสอบโปรแกรมจะเริ่มต้นจากการทดสอบการกำหนดจุดลงในภาพทั้งสามมุมมอง และขั้นตอนที่สองจะเป็นขั้นตอนการทดสอบโปรแกรมส่วนที่สร้างจุด และเส้น 3 มิติ, การสร้างเส้น 3 มิติที่มองไม่เห็น และการสร้างพื้นผิว

## 4.1 การทดสอบโปรแกรมกำหนดจุดลงในแต่ละภาพ

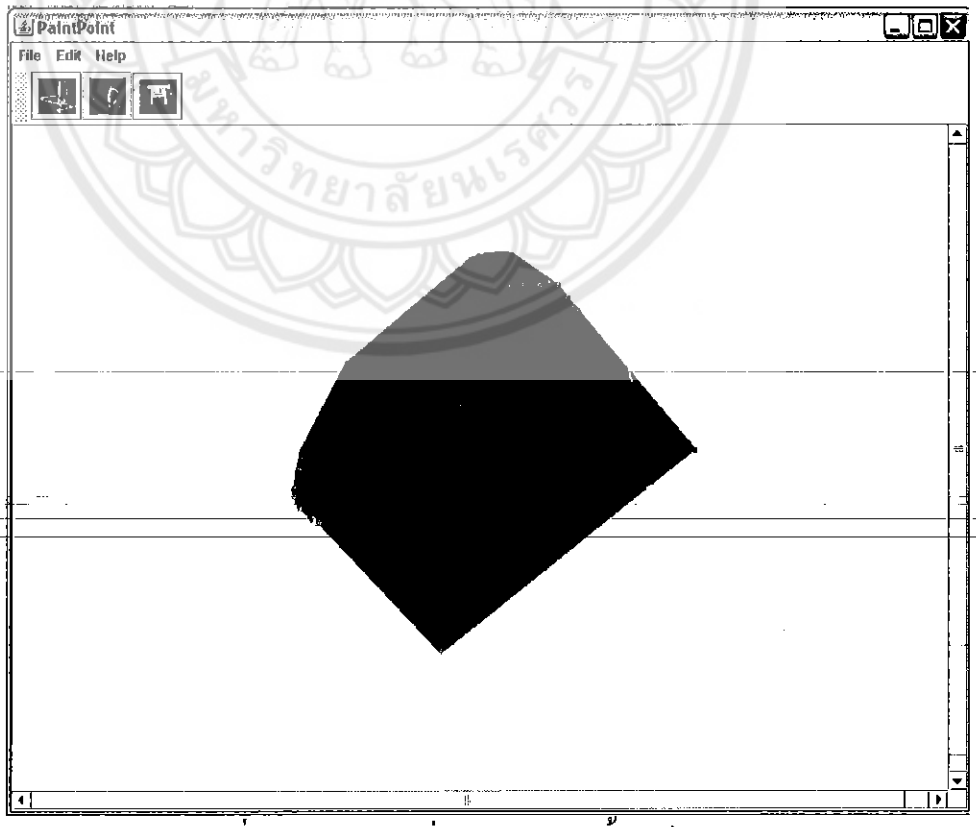
การทดสอบโปรแกรมกำหนดจุดลงไปในภาพแต่ละมุมมองนั้นผู้ดำเนินโครงงานได้ใช้ภาษา Java ซึ่งขั้นตอนนั้นเริ่มจากการอ่านค่าไฟล์ภาพในแต่ละมุมมองออกมาก่อน จากนั้นจึงทำการกำหนดจุดไปตามมุมมองตามหลักการในหัวข้อ 3.1 ในบทที่ 3 โดยคลิกที่ปุ่ม mask 1 ครั้งก่อนที่จะกำหนดจุดซึ่งในแต่ละด้านนั้นเมื่อถึงจุดปลายของแต่ละด้านให้คลิกที่ปุ่ม lastEndPoint ตามรูปที่ 4.1 เมื่อกำหนดจุดเสร็จทั้ง 4 เส้นแล้วจากนั้นจะบันทึกข้อมูลลงในไฟล์ .txt เพื่อนำไปใช้ในขั้นตอนการสร้างเส้น 3 มิติ จากเส้น 2 มิติ



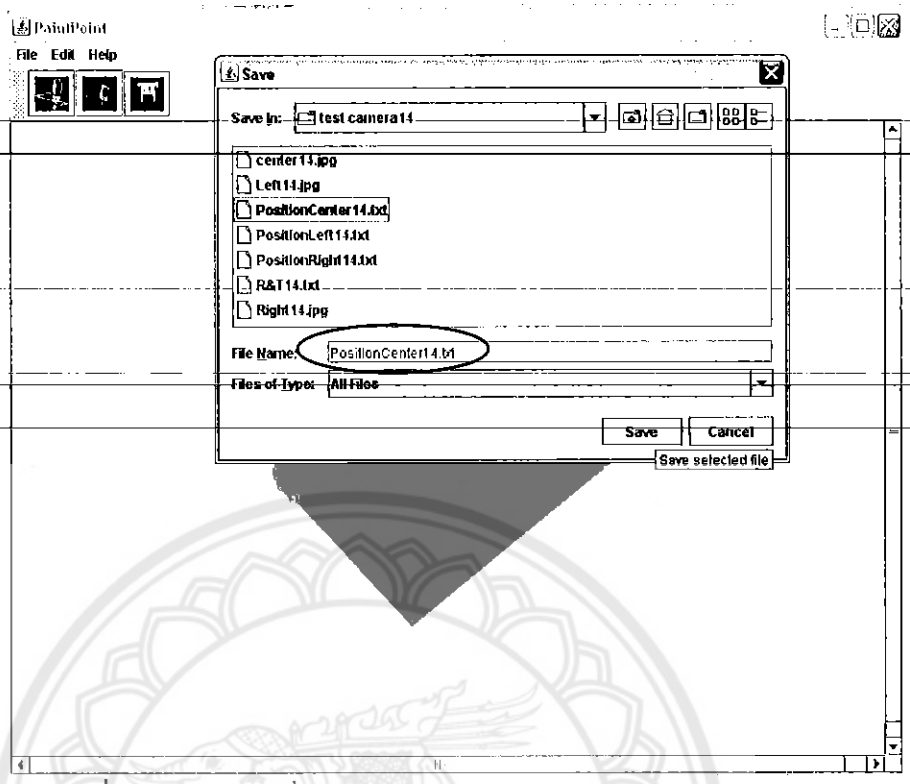

**รูปที่ 4.1** แสดงภาพที่อ่านมาจากไฟล์ภาพเพื่อทำการกำหนดจุดของมุมมองตรงกลาง



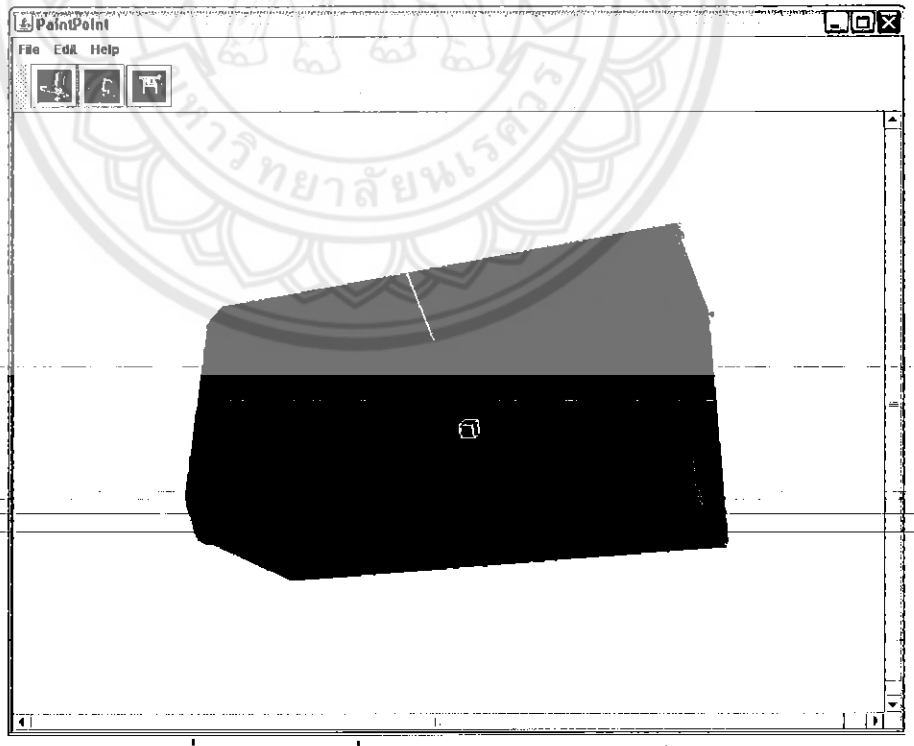

รูปที่ 4.2 แสดงภาพเมื่อคลิกที่ปุ่ม lastEndPoint เพื่อกำหนดจุดสุดท้ายในด้านที่ 1



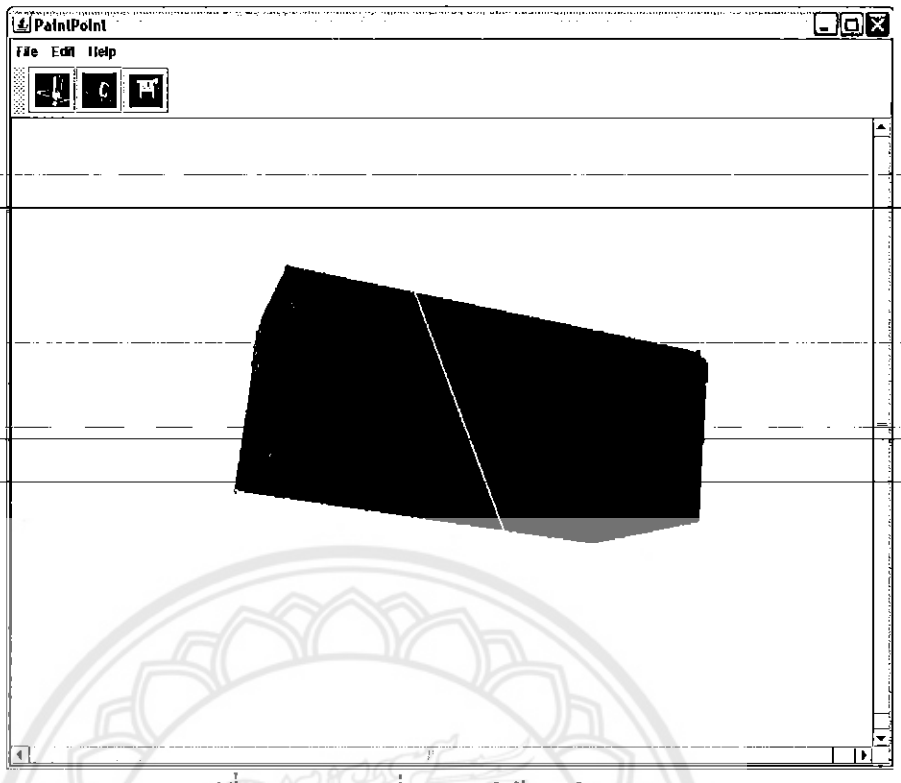

**รูปที่ 4.3** แสดงภาพเมื่อเลือกจุดครบทั้ง 4 ด้าน




รูปที่ 4.4 แสดงภาพเมื่อทำการบันทึกค่าของแต่ละจุดเป็นไฟล์ .txt



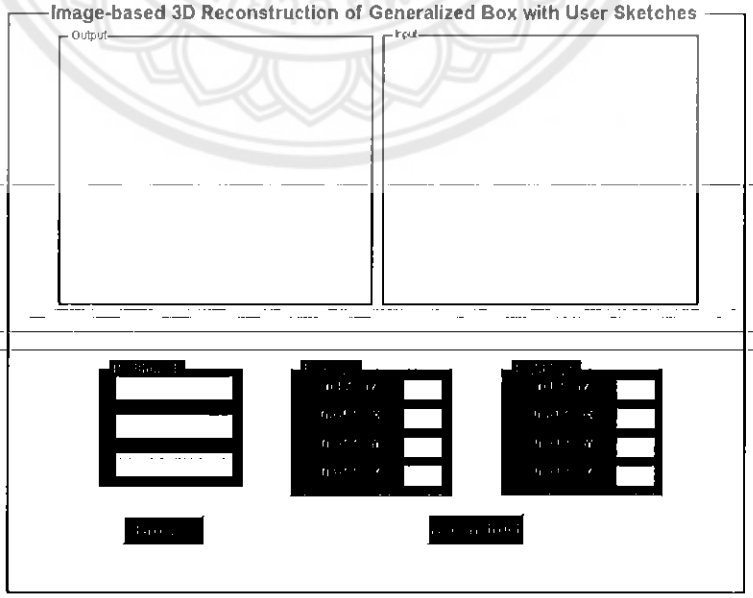

**รูปที่ 4.5** แสดงจุดที่กำหนดให้ภาพในมุมมองซ้าย



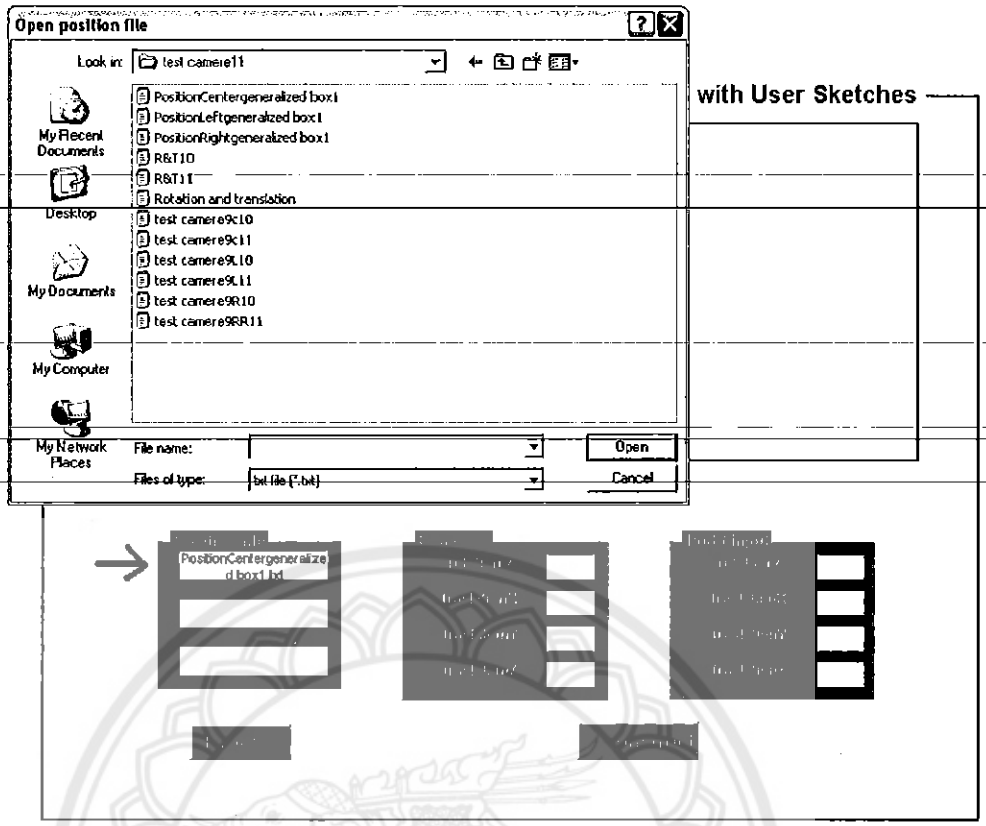

รูปที่ 4.6 แสดงจุดที่กำหนดให้ภาพในมุมมองขวา

## 4.2 การทดสอบโปรแกรมส่วนที่ใช้สร้างเส้น 3 มิติ

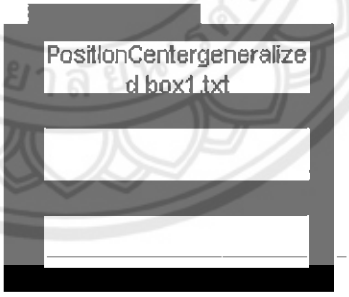
การทดสอบโปรแกรมส่วนที่ใช้สร้างเส้น 3 มิตินั้นผู้ดำเนินโครงงานใช้โปรแกรม MATLAB ในการเขียนและทดสอบโปรแกรม ซึ่งผลการทดสอบโปรแกรมแสดงตามภาพดังต่อไปนี้



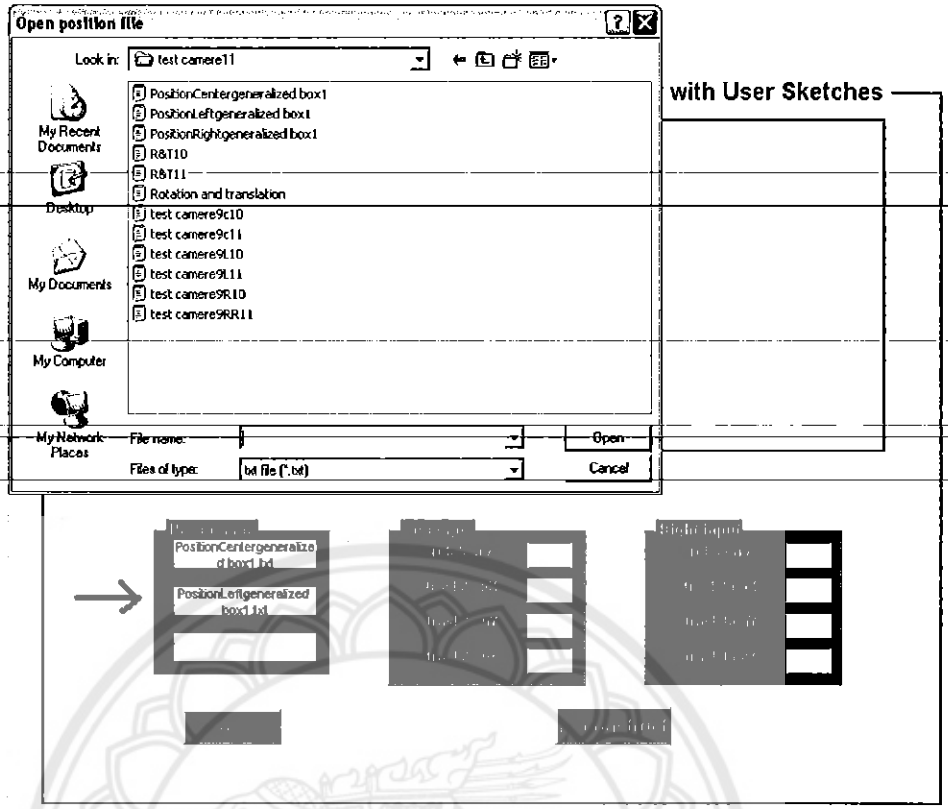

**รูปที่ 4.7** แสดง Graphic User Interface ของโปรแกรมการสร้างภาพ 3 มิติจากภาพ 2 มิติ



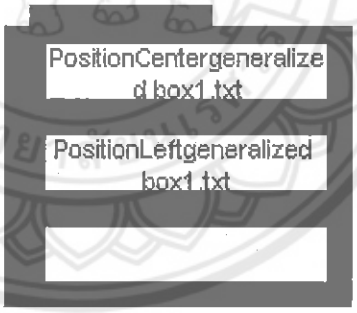

รูปที่ 4.8 แสดงการเปิดไฟล์ .txt ที่มีค่าตำแหน่งของภาพมุมมอง**ตรงกลาง**



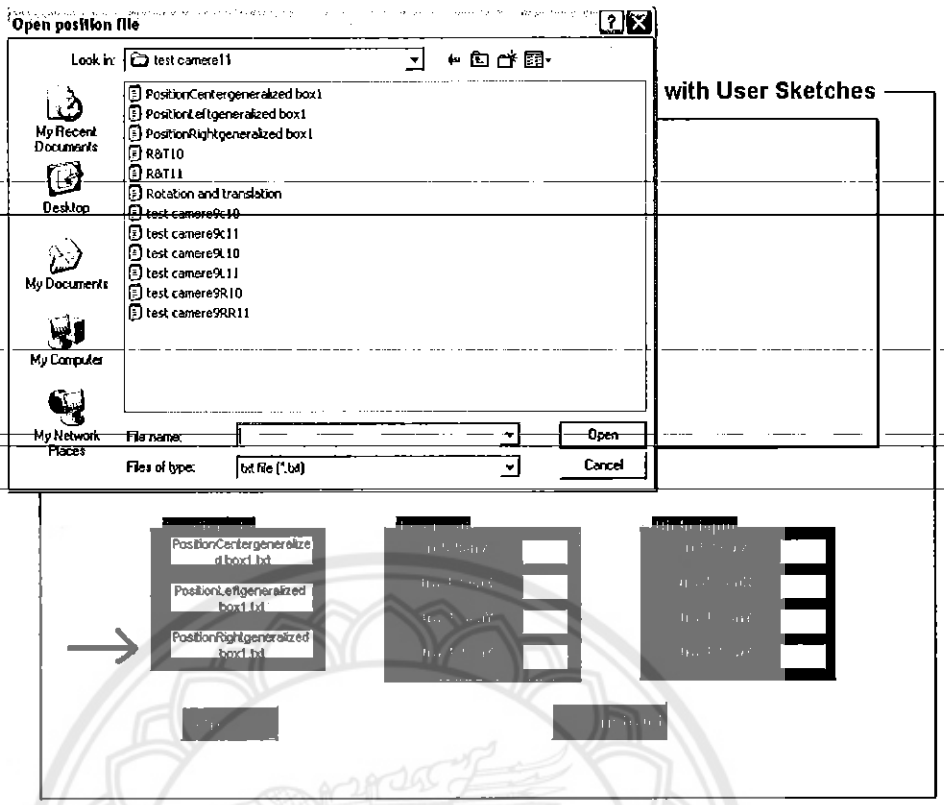

**รูปที่ 4.9** เมื่อเปิดไฟล์ข้อมูลตำแหน่งมุมมองตรงกลางแล้วชื่อไฟล์จะปรากฏดังภาพ



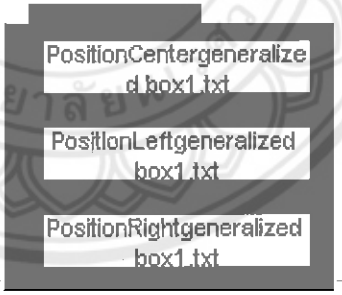

รูปที่ 4.10 แสดงการเปิดไฟล์ .txt ที่มีค่าตำแหน่งของภาพมุมมองด้านซ้าย



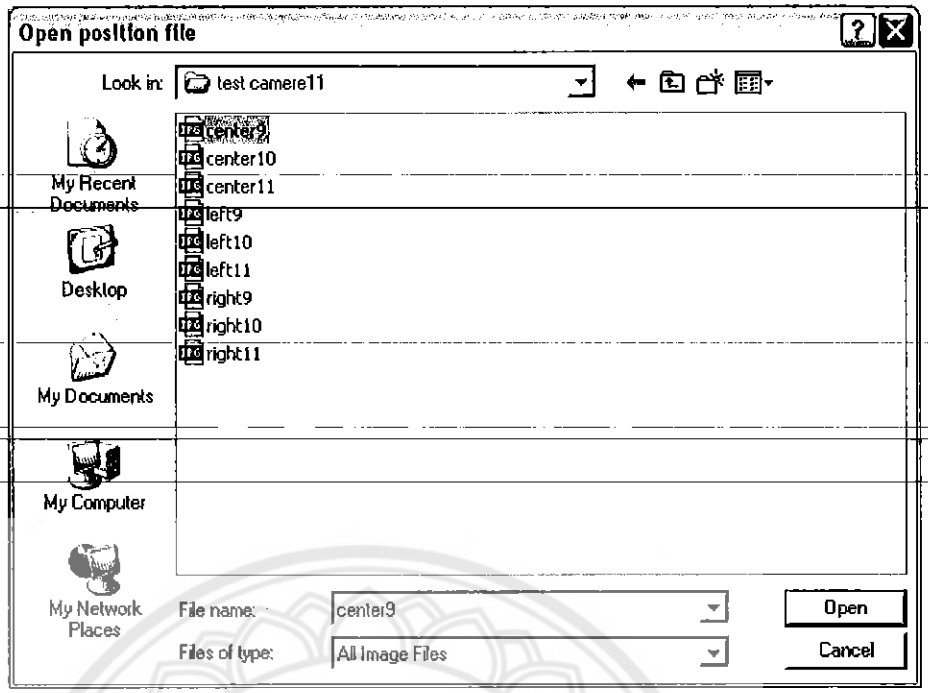

**รูปที่ 4.11** เมื่อเปิดไฟล์ข้อมูลตำแหน่งมุมมองด้านซ้ายแล้วชื่อไฟล์จะปรากฏดังภาพ



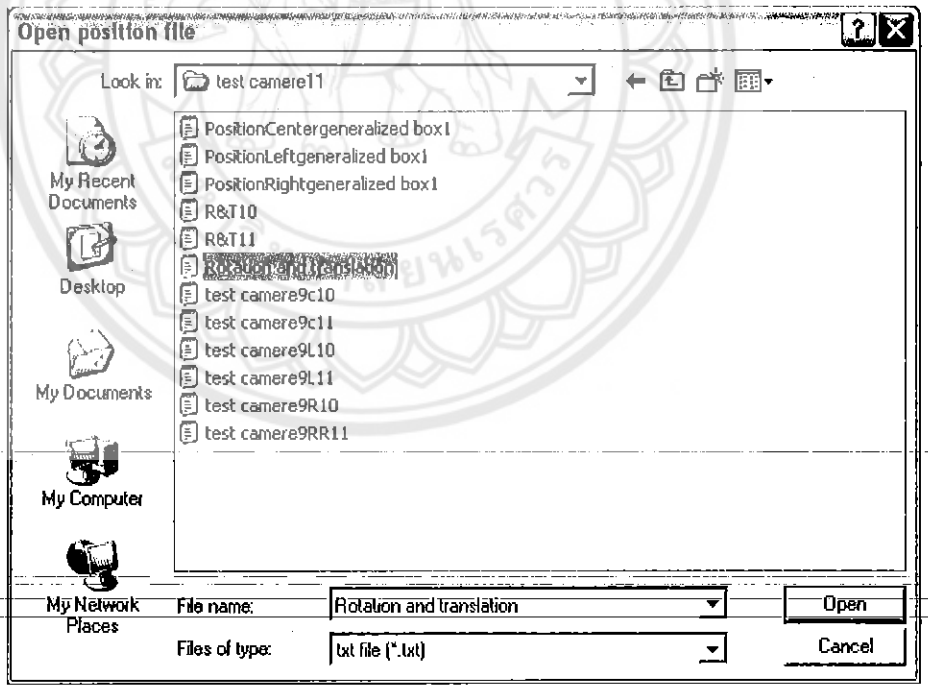

รูปที่ 4.12 แสดงการเปิดไฟล์ .txt ที่มีค่าตำแหน่งของภาพมุมมองด้านขวา



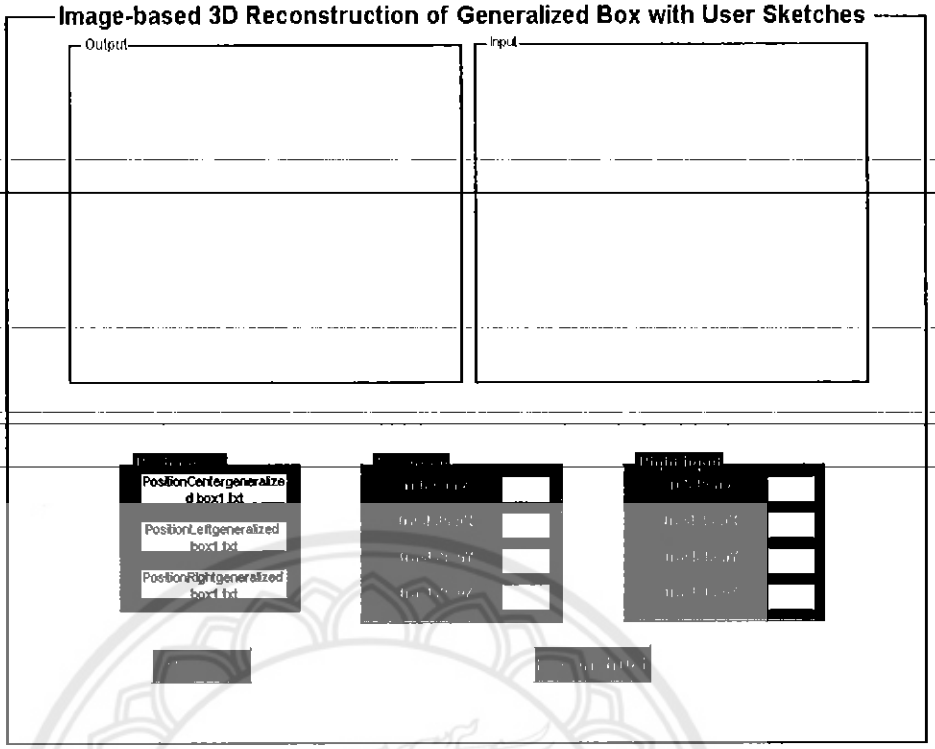

**รูปที่ 4.13** เมื่อเปิดไฟล์ข้อมูลตำแหน่งมุมมองด้านขวาแล้วชื่อไฟล์จะปรากฏดังภาพ



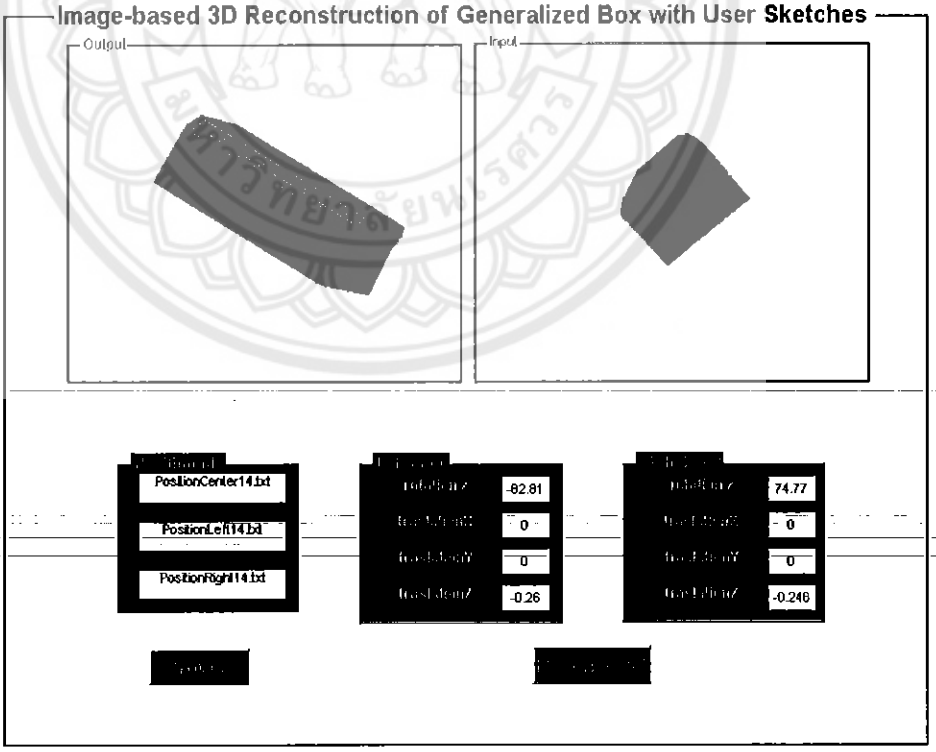

รูปที่ 4.14 แสดงการเลือกภาพที่จะนำมาใช้เป็นภาพอินพุต



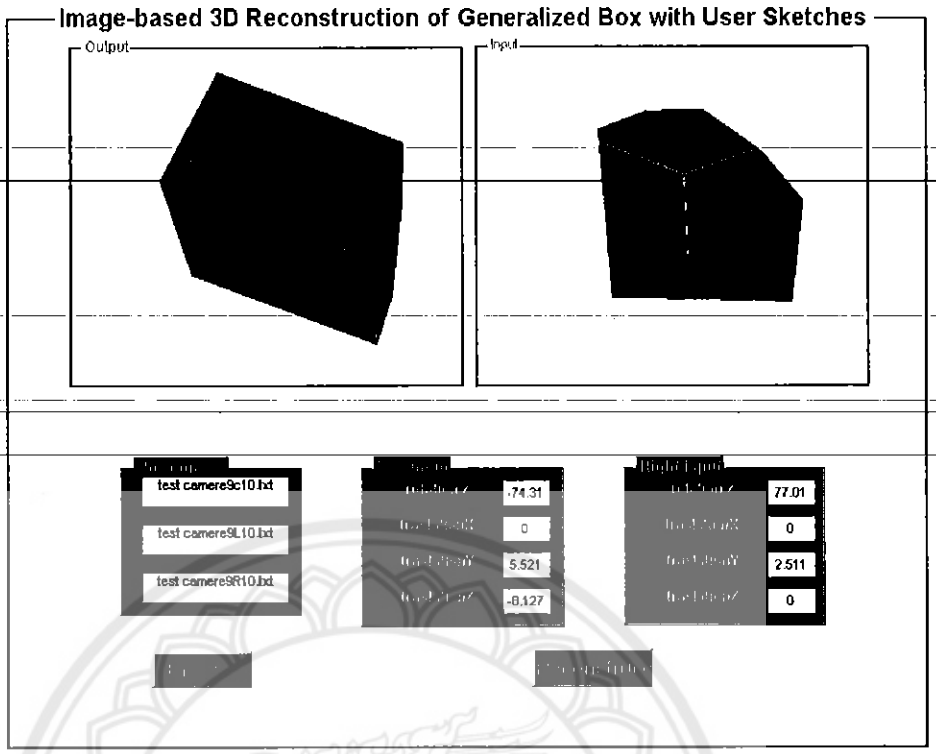

**รูปที่ 4.15** แสดงการเลือกค่า Rotation และค่า Translation



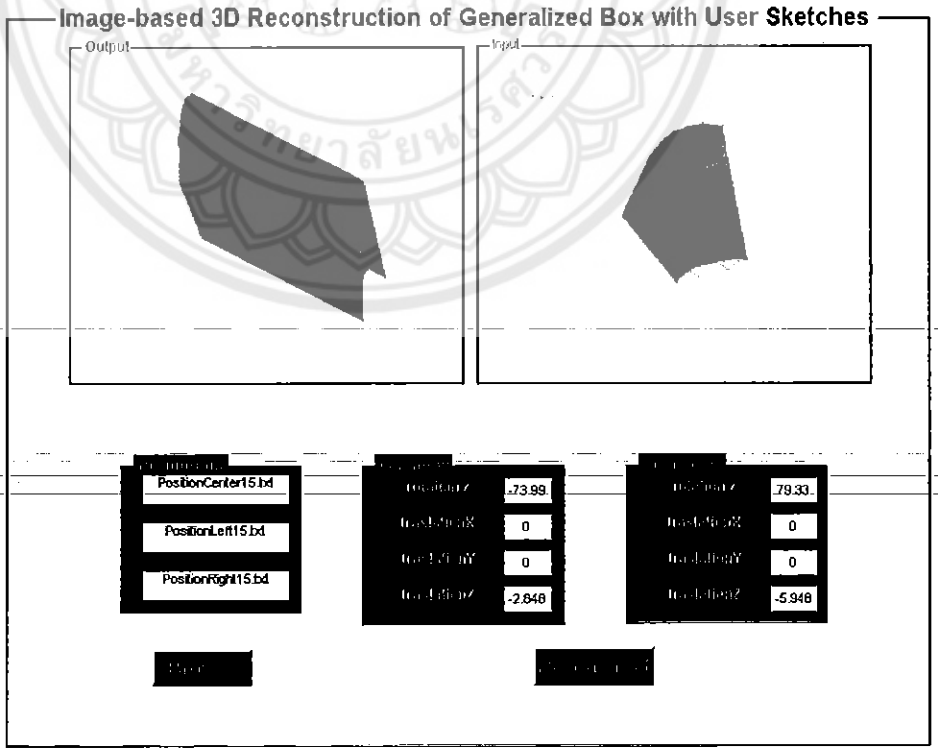

**รูปที่ 4.16** แสดงภาพ Graphic User Interface ก่อนทำการ Reconstruct




**รูปที่ 4.17** แสดงผลการสร้างภาพ 3 มิติจากภาพ 2 มิติ




รูปที่ 4. 18 แสดงตัวอย่างของภาพอื่น ๆ (1)




**รูปที่ 4. 19** แสดงตัวอย่างของภาพอื่น ๆ (2)

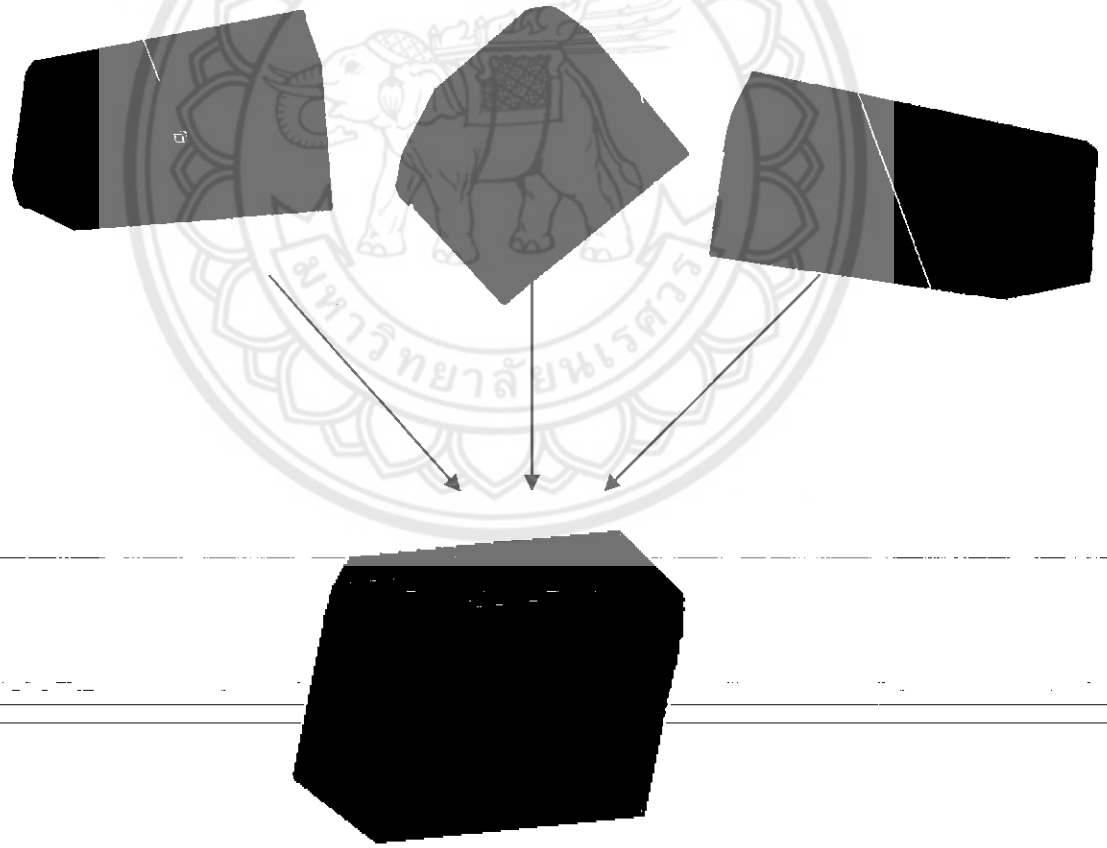
# บทที่ 5

# บทสรุป

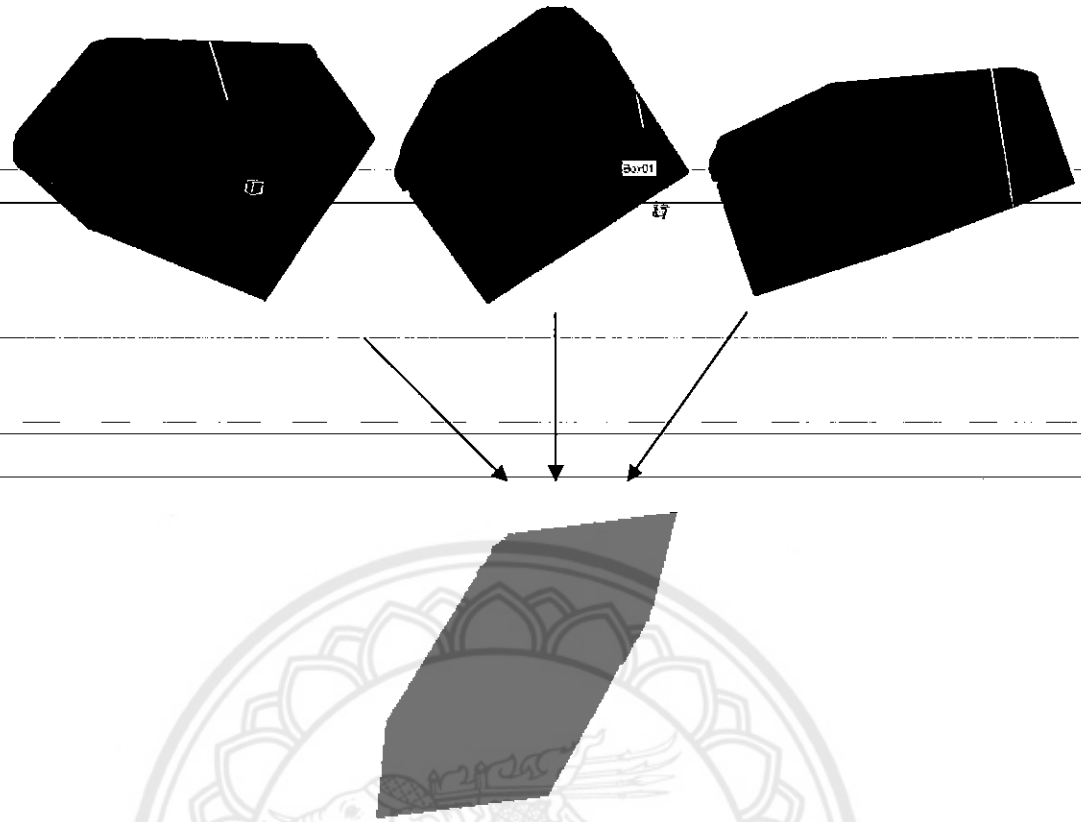
โครงงานการสร้างภาพ 3 มิติจากภาพ 2 มิติโดยศึกษาจากอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets สามารถสร้างโปรแกรมสร้างภาพ 3 มิติจากภาพ 2 มิติซึ่งเป็นภาพจาก 3 มุมมองได้ และมี Graphic User Interface (GUI) ให้ผู้ใช้ ใช้งานได้สะดวกยิ่งขึ้น

## 5.1 วิเคราะห์ผลการทดลอง

จากการทดลอง อินพุต ที่ใช้คือรูป 3 รูปจาก 3 มุมมอง ซึ่งได้แก่มุมมองทางด้านซ้าย, มุมมองตรงกลาง และมุมมองด้านขวา จะทำให้ได้วัตถุ 3 มิติดังรูป



**รูปที่ 5. 1** แสดงภาพด้านซ้ายและด้านขวาทำมุมจากมุมมองตรงกลาง 80 องศา




รูปที่ 5. 2 แสดงภาพด้านซ้ายและด้านขวาทำมุมจากมุมมองตรงกลาง 40 องศา

จากผลการทดลองจะสังเกตได้ว่าภาพแต่ละมุมมองที่นำมาใช้มีผลต่อรูปทรงของวัตถุอย่างมาก คือ ถ้ามุมที่มองจากด้านซ้ายและด้านขวาทำมุมจากมุมมองตรงกลางมาก ๆ จะทำให้วัตถุที่ได้ใกล้เคียงกับวัตถุจริง ส่วนสีของวัตถุที่ได้มีความแตกต่างจากวัตถุต้นแบบเล็กน้อยเนื่องจากวัตถุต้นแบบมีแสงมากระทบทำให้เกิดเงาจึงส่งผลต่อการวิเคราะห์สี

## 5.2 ปัญหาและแนวทางแก้ไข

ปัญหาที่เกิดขึ้นในระหว่างขั้นตอนต่าง ๆ ในการทำโครงงานนี้มีทั้งส่วนในการศึกษาอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets และส่วนในการเขียนโปรแกรมซึ่งได้แก่

1. ในขั้นตอนศึกษาอัลกอริทึมนั้นปัญหาคือเมื่ออ่านอย่างละเอียดแล้วพบว่ามีจุดที่ไม่เข้าใจเริ่มตั้งแต่หลักของการ split point ว่าหลังจากที่ได้จุด original point มาแล้วนั้นจะทำให้จุดนั้นเพิ่มขึ้นได้อย่างไร วิธีการแก้ปัญหานี้คือ เริ่มต้นจากการศึกษาเรื่อง interpolation ต่อมาจึงศึกษาเรื่องสมการเส้นตรง จึงพบว่าหลักการของสมการเส้นตรงสามารถแก้ปัญหาเรื่องการแยกจุดออกมาได้

2. ปัญหาเรื่องการหา error ที่น้อยที่สุด เพราะว่าจากอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets ได้บอกเพียงว่าใช้หลักการอัลกอริทึมเรื่อง gradient descent ในเรื่อง triangulation ผู้จัดทำโครงงานจึงแก้ปัญหาโดยการหาข้อมูลเพิ่มเติมเกี่ยวกับการหา error โดยใช้หลักการ triangulation จึงพบว่าสามารถใช้วิธีการทำ Fundamental matrix สามารถที่จะหาค่า error ที่น้อยที่สุดได้
3. เมื่อหาว่าการใช้ Fundamental matrix หาค่า error ที่น้อยที่สุดได้แต่หลักการของ triangulation ซึ่งเขียนโดย R. I. Hartley, P. Sturm นั้นเข้าใจได้ยากและซับซ้อนมากเกินไป ผู้ดำเนินโครงงานจึงเปลี่ยนมาใช้หลักการเรื่อง Epipolargeometry and Fundamental matrix ของ A. Zisserman ซึ่งเข้าใจง่ายกว่าและนำมาใช้ในโครงงานนี้
4. ในขั้นตอนการเขียนโปรแกรมนั้น เริ่มแรกผู้จัดทำโครงงานคิดจะใช้โปรแกรมภาษา C++ ในการเขียนโปรแกรมแต่พบว่าถ้าใช้โปรแกรมภาษา C++ เขียนจะทำให้โปรแกรมมีขนาดใหญ่และมีความซับซ้อนมากเกินไป ผู้ดำเนินโครงงานจึงเปลี่ยนมาใช้โปรแกรม MATLAB ในการเขียนโปรแกรมแทน
5. เนื่องจากโปรแกรมต้องอ่านค่าจุดที่กำหนดลงไปในภาพ ผู้ดำเนินโครงงานไม่สามารถหาฟังก์ชันการทำงานของโปรแกรม MATLAB ในการทำงานส่วนนี้ได้และการทดลองทุกครั้งจะต้องพิมพ์ค่าของจุดซึ่งมีจำนวนมากลงไปในโปรแกรม ผู้ดำเนินโครงงานจึงแก้ปัญหาส่วนนี้ด้วยการเปลี่ยนมาใช้โปรแกรมภาษา Java ในการกำหนดจุดลงไปในภาพและการอ่านค่าตำแหน่งของจุด และเมื่ออ่านค่าตำแหน่งแล้วจะบันทึกลงไปไฟล์ .txt เพื่อนำไปอ่านค่าในโปรแกรมที่เขียนด้วยโปรแกรม MATLAB แทน
6. เมื่อได้โครงสร้าง 3 มิติแล้วจะต้องหาพื้นผิวของวัตถุซึ่งในอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets นั้นบอกเพียงว่าใช้อัลกอริทึม discrete coons patches ผู้ดำเนินโครงงานไม่สามารถที่จะทำตามอัลกอริทึมนั้นได้ จึงทำการแก้ปัญหาโดยใช้ทฤษฎีเรื่อง automatic thresholding ในการกำหนดสีบนพื้นผิวของวัตถุ

## 5.3 สรุปผลการทดลอง

การสร้างวัตถุ 3 มิติจากภาพ 2 มิตินั้นมีรูปแบบการสร้างมากมายซึ่งในโครงงานชิ้นนี้ได้เลือกศึกษาตามอัลกอริทึมของ Anton Yakubenko, Anton Konouchine และ Vladimir Vezhnevets เพราะว่าในโครงงานชิ้นนี้เลือกที่จะศึกษาการสร้างวัตถุ 3 มิติโดยกำหนดรูปร่างของภาพอินพุตว่าจะต้องเป็นรูปทรงกล่อง รวมถึงให้ผู้ใช้ได้มีส่วนร่วมในการสร้างด้วย ขั้นตอนที่สำคัญของโปรแกรมการสร้างภาพ 3 มิติ จากภาพ 2 มิติคือ ขั้นตอนการศึกษาอัลกอริทึม และขั้นตอนการเขียน

โปรแกรมตามอัลกอริทึมที่ศึกษา ซึ่งจากผลการทดลองแสดงให้เห็นว่าโปรแกรมสามารถสร้างภาพ 3 มิติจากภาพ 2 มิติ โดยใช้รูปภาพ 3 ภาพได้

## 5.4 ข้อเสนอแนะ

1. การสร้างพื้นผิวของโครงสร้าง 3 มิติที่ได้ สามารถใช้วิธีการอื่น ๆ ในการพัฒนาได้ เช่น อัลกอริทึม discrete coons patches เป็นต้น
2. สามารถนำหลักการเขียนโปรแกรมนี้ไปใช้กับภาพอินพุตในรูปแบบอื่น ๆ เช่น ปิระมิด เป็นต้น
3. สามารถนำโปรแกรมนี้ไปพัฒนาต่อเพื่อเพิ่มความเร็วในการสร้างได้
4. ถ้ามีรูปถ่ายอินพุตมากกว่าเดิม จะทำให้ได้รายละเอียดได้มากขึ้น

# เอกสารอ้างอิง

[1] Anton Yakubenko, Anton Konouchine and Vladimir Vezhnevets. **"Image – based 3D Reconstruction of Generalized Box with User Sketch"**. [Online]. Available: www.graphicon.ru/proceedings2006/papers/we10_88_Yakubenko.pdf. 2006

[2] A. Zisserman. **"Epipolargeometry and Fundamental matrix"**. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook1/HZepipolar.pdf

[3] พรพล สาครินทร์และกฤษฎา แก้วมณี. **พื้นฐานการก้าวสู่โลก 3 มิติ 3D Graphics.** กรุงเทพมหานคร : ซัคเซสมี - เดีย. 2543

[4] R. I. Hartley, P. Sturm. **"Triangulation"** ARPA Image Understanding Workshop. Monterey California. pp. 957-966. November 1994.

[5] G. Farin, D. Hansford. **"Discrete Coons patche".** Computer Aided Geometric Design 16. pp. 691–700. 1999.

# ประวัติผู้เขียนโครงงาน

**ชื่อ** นางสาวกฤติกา อภิวงค์งาม

**ภูมิลำเนา** 136 หมู่ที่ 5 ตำบลบ้านธิ อำเภอบ้านธิ จังหวัดลำพูน 51180

**ประวัติการศึกษา**

- จบการศึกษาระดับมัธยมศึกษาจาก โรงเรียนสาธิตมหาวิทยาลัยเชียงใหม่
- ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

**E-mail** cuppucino_coffee@hotmail.com

**ชื่อ** นางสาวปรางนภา ทารัตน์

**ภูมิลำเนา** 94/1 หมู่ 2 ตำบลวังแดง อำเภอตรอน จังหวัดอุตรดิตถ์ 53140

**ประวัติการศึกษา**

- จบการศึกษาระดับมัธยมศึกษาจาก โรงเรียนอุตรดิตถ์ดรุณี
- ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

**E-mail** ayame_pr@hotmail.com