ระบบสั่งงานด้วยเสียง

Voice-to-Command Agent

โดย

นายรวินทร์ วิรัชพินทุ

นางสาวสุวารี เหลือเพิ่มพร

อาจารย์ที่ปรึกษา

อาจารย์สมเกียรติ วังศิริพิทักษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบสั่งงานด้วยเสียง

Voice-to-Command Agent

คณะผู้จัดทำ นายรวินทร์ วิรัชพินทุ รหัส 43010348

นางสาวสุวารี เหลือเพิ่มพร รหัส 43010504

.....อาจารย์ที่ปรึกษา

(อ.สมเกียรติ วังศิริพิทักษ์)

ระบบสั่งงานด้วยเสียง

นายรวินทร์ วิรัชพินทุ 43010348
นางสาวสุวารี เหลือเพิ่มพร 43010504
อ.สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ในปัจจุบันเทคโนโลยีทางด้านข้อมูลข่าวสารได้ถูกพัฒนาจากเดิมที่ข้อมูลเป็นตัวอักษรธรรมดาให้มีรูปแบบที่หลากหลายมากขึ้น ในขณะที่การสั่งงานยังอยู่ในรูปแบบของการคีย์ข้อมูลใส่ซึ่งเป็นรูปแบบที่ไม่สะดวกในงานบางอย่าง ทางกลุ่มจึงสนใจที่จะทำการพัฒนาการรับข้อมูลเข้าสู่ระบบคอมพิวเตอร์จากการคีย์คำสั่งเป็นการสั่งงานด้วยเสียงด้วยรูปแบบคำสั่งที่สามารถให้ผู้ใช้ (เจ้าของเครื่อง) ทำการสอนให้เครื่องคอมพิวเตอร์เรียนรู้ได้ (ในระดับหนึ่ง) เพื่อนำไปพัฒนาระบบให้สามารถสั่งงานด้วยเสียงทั้งหมดในอนาคต

Voice-to-Command Agent

Mr. Rawin Viruchpintu
Miss Suwaree Luapermporn
Mr. Somkiat Wangsiripitak Advisor
Academic Year 2003

ABSTRACT

Nowadays technology of input device is variety such as keyboard , joystick , touch-screen etc. Commonly input device is keyboard but some case inconvenience. Therefore we are developing a system , that uses voice command instead of keyboard device. Step by user record speech and train computer then test computer to work. It will be trend to develop many applications.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำ คำปรึกษาและคอยดูแลจากหลาย ๆ ฝ่ายด้วยกัน โดยเฉพาะอาจารย์ที่ปรึกษาที่คอยให้ความเอาใจใส่ แนะนำและช่วยเหลือเสมอมา ซึ่งก็คือ อาจารย์สมเกียรติ วังศิริพิทักษ์ ขอขอบพระคุณเป็นอย่างสูง

นอกเหนือจากนี้ต้องขอขอบพระคุณอาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังเป็นอย่างยิ่งที่ได้ช่วยประสิทธิ์ประสาทวิชาความรู้ให้แก่คณะผู้จัดทำ

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เช่น อินเทอร์เน็ตความเร็วสูง ซึ่งช่วยให้การวิจัย การค้นคว้าหาความรู้ต่าง ๆ และพัฒนาโปรแกรมเป็นไปได้ด้วยความสะดวกและรวดเร็ว

สุดท้ายต้องขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดาและบุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดู คอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ ความรักเสมอมา ข้าพเจ้าขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

รวินทร์ วิรัชพินทุ

สุวารี เหลือเพิ่มพร

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญตาราง

บทที่ 1
หลักการและเหตุผล

1.1 หลักการและเหตุผล

โดยปกติแล้วการติดต่อสื่อสารระหว่างมนุษย์กับเครื่องคอมพิวเตอร์ทำได้โดยการป้อนคำสั่งผ่านทางคีย์บอร์ดและเมาส์ ในขณะที่มีความต้องการหาวิธีการอื่นที่เป็นไปอย่างสะดวกและเป็นธรรมชาติมากกว่า จึงมีการพัฒนาให้คอมพิวเตอร์สามารถรับรู้คำสั่งจากเสียงพูดของมนุษย์ได้ ซึ่งจะทำให้การติดต่อสื่อสารระหว่างมนุษย์และคอมพิวเตอร์เป็นไปอย่างสะดวกและง่ายขึ้น

สมัยก่อนการใช้คอมพิวเตอร์เพื่อรับรู้เสียงพูดของมนุษย์และวิเคราะห์เสียงพูดนั้นทำได้ยาก และใช้เวลานาน เนื่องจากการพูดของมนุษย์มีความซับซ้อนและมีความแตกต่างในแต่ละบุคคลทำให้การพัฒนาเป็นไปอย่างล่าช้า แต่ในปัจจุบันได้มีการพัฒนาวิธีการต่างๆ ให้มีความสามารถในการที่จะหาตัวแทนของเสียงในรูปแบบต่าง ๆ รวมทั้งมีวิธีการเพิ่มมากขึ้น และมีความแม่นยำในการรู้จำเสียงที่ป้อนให้แก่คอมพิวเตอร์ด้วยพื้นฐานความรู้ทางด้านคณิตศาสตร์และพื้นฐานความรู้ทางด้านสถิติจึงทำให้เราลดความยุ่งยากลงได้มาก

ปริญญานิพนธ์นี้ได้มีการศึกษาและทดลองเอาส่วนของอัลกอริทึม (Algorithm) การรู้จำเสียงที่เป็นทฤษฎีนำมาใช้งานจริงบนเครื่องคอมพิวเตอร์ส่วนบุคคล โดยการบันทึกเสียงที่จะทำการทดลองโดยใช้อุปกรณ์ ไมโครโฟน การ์ดเสียงและลำโพง รูปแบบของเสียงที่บันทึกจะอยู่ในรูปแบบคำสั่งที่ผู้ใช้งานคอมพิวเตอร์พูด โดยทดลองกับตัวอย่างคำสั่งทั้งหมด 10 คำสั่งและมีช่วงความถี่อยู่ระหว่าง 300-3400 เฮิร์ต ด้วยความละเอียดขนาด 16 บิท และความถี่ในการสุ่ม 11025 เฮิร์ต (บันทึกเสียงอยู่ในรูปแบบ ชื่อไฟล์.wav) เพื่อให้คำสั่งนั้นมีค่าพารามิเตอร์ในการวิเคราะห์น้อยลง เราจะทำการแทนค่าของสัญญาณเสียงในแต่ละช่วงด้วยพารามิเตอร์เอ็มเอฟซีซี (MFCC parameter) เพื่อให้ค่าที่ได้มีเสถียรภาพที่ดีขึ้นเรานำไปสร้างรูปแบบอ้างอิงโดยใช้หลักการของเวคเตอร์ควอนไตเซชั่น (Vector Quantization) ซึ่งสามารถนำไปใช้ในการรู้จำเสียงเพื่อรับรู้ว่าเสียงที่เราป้อนให้แก่เครื่องคอมพิวเตอร์นั้นเป็นเสียงอะไร จะเห็นได้ว่าถ้าเราเลือกวิธีการทางทฤษฎีที่เหมาะสมแล้วจะทำให้เวลาในการคำนวณหาค่าต่าง ๆ ในการรู้จำเสียงใช้เวลาน้อยลง

1.2 วัตถุประสงค์

1. เพื่อศึกษาการทำงานของระบบรู้จำเสียง
2. เพื่อศึกษาการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการ Microsoft Windows
3. เขียนโปรแกรมประยุกต์เพื่อพัฒนาระบบรู้จำเสียงมาใช้งานจริง

1.3 ผลที่คาดว่าจะได้รับ

1. ได้รับความรู้ ความสามารถในการรู้จำเสียง
2. ได้รับความรู้ ความสามารถในการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการ Microsoft Windows

3. โปรแกรมที่ทำงานตามคำสั่งของผู้ใช้งานคอมพิวเตอร์

1.4 ขอบเขตของการพัฒนา

- พัฒนาระบบรู้จำเสียงที่ไม่จำกัดลักษณะคำสั่งของเจ้าของเครื่อง ในที่นี้เราทำการทดลองเพียง 10 คำสั่ง เพราะต้องการให้การประมวลผลใช้เวลาไม่นานมากนัก
- พัฒนาระบบบันทึกการทำงานของผู้ใช้โดยใช้ Windows Hook
- พัฒนาโปรแกรมการเรียนรู้และทำงานระบบด้วยภาษา C++ แบบ MFC , Win32API และเก็บข้อมูลด้วย Microsoft Access XP

1.5 คุณลักษณะของระบบสั่งงานด้วยเสียง

- มีระบบรู้จำเสียงที่สามารถจำแนกคำสั่งของบุคคลคนเดียวกันว่าเป็นคำสั่งใด
- มีระบบการสอนให้คอมพิวเตอร์เรียนรู้ว่า "เสียงคำสั่งใด หมายถึง การสั่งงานอย่างไร" ด้วยการพูดคำสั่ง และแสดงขั้นตอนการทำงานโดยใช้เมาส์และคีย์บอร์ด
- มีระบบทดสอบการทำงาน โดยผู้ใช้สั่งงานด้วยเสียง
- มีระบบทำงานอัตโนมัติในการยืนยันการทำงานการสั่งงานด้วยเสียง
- มีระบบฐานข้อมูลสำหรับเก็บผลการเรียนรู้ไว้สำหรับใช้ในการทำงานครั้งต่อ ๆ ไปได้
- มีระบบปรับเพิ่มเติม/ลบ/แก้ไขคำสั่งที่ทำการเรียนรู้ไปแล้ว
- มีระบบล็อคอินของสมาชิกเข้าสู่ระบบสั่งงานด้วยเสียง

1.6 คุณลักษณะของอุปกรณ์ที่ใช้กับระบบสั่งงานด้วยเสียง

- ไมโครโฟน ไม่ควรมีเสียงรบกวน (Noise) จากสายไฟ
- การ์ดเสียง (Sound card) ต้องมีความคมชัด
- ห้องที่ใช้ทำการทดสอบไม่ควรมีเสียงดังรบกวน

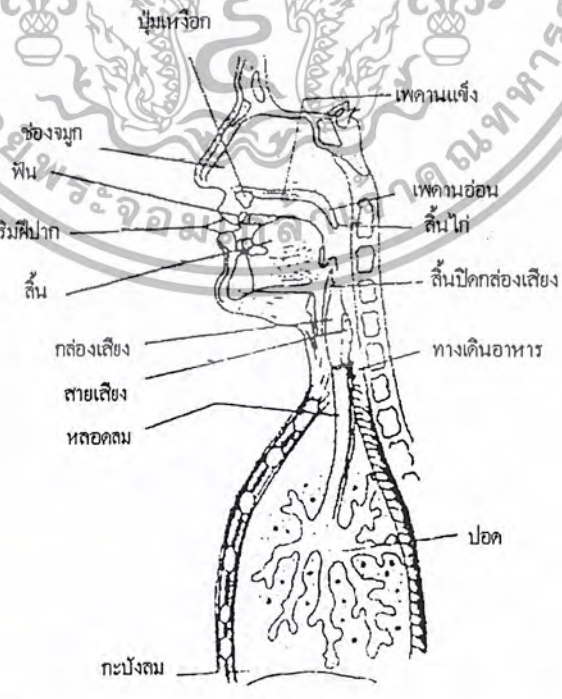
บทที่ 2
ทฤษฎี

ในปัจจุบันความก้าวหน้าทางด้านการประมวลผลสัญญาณต่าง ๆ ได้รุดหน้าไปอย่างรวดเร็ว ซึ่งในบางโอกาสสัญญาณอาจถูกประมวลผลในระบบอนาลอก (Analog) ในบางโอกาสก็ถูกประมวลผลในระบบดิจิตอลด้วยเครื่องคอมพิวเตอร์ ซึ่งในระบบของการประมวลผลด้วยเครื่องคอมพิวเตอร์ สัญญาณที่ถูกประมวลผลจะถูกแปลงจากสัญญาณอนาลอกที่ต่อเนื่องไปเป็นสัญญาณดิสครีท (Discrete) มีการสุ่มเอาตัวอย่างสัญญาณไปประมวลผล ซึ่งอยู่ในส่วนการทำงานของการ์ดเสียง จากนั้นจะเป็นขั้นตอนการประมวลผลซึ่งมีมากมายหลายวิธี ซึ่งล้วนแต่มีวัตถุประสงค์ในการประมวลผลสัญญาณเพื่อแยกแยะสัญญาณต่าง ๆ ไปทำประโยชน์ตามต้องการ

2.1 กายวิภาคของระบบการเปล่งเสียงของมนุษย์

จากการศึกษาด้านกายวิภาคศาสตร์ของมนุษย์ (human anatomy) วิชาที่ว่าด้วยเสียงของภาษา (phonetics) และศาสตร์ทางด้านเสียง (acoustics) ช่วยให้เข้าใจขั้นตอนการทำงานร่วมกันของอวัยวะต่าง ๆ ในการเปล่งเสียงพูด ตลอดจนลักษณะทางกายภาพของเสียงพูดเพื่อนำมาวิเคราะห์ และสร้างแบบจำลองเลียนแบบเสียงพูดของมนุษย์

การทำให้เกิดเสียงเป็นหน้าที่หนึ่งของระบบหายใจ การออกเสียงหรือการพูดของมนุษย์แต่ละครั้งจะต้องมีการทำงานร่วมกันของอวัยวะต่าง ๆ ของร่างกาย ดังรูปที่ 2.1 อันประกอบด้วย

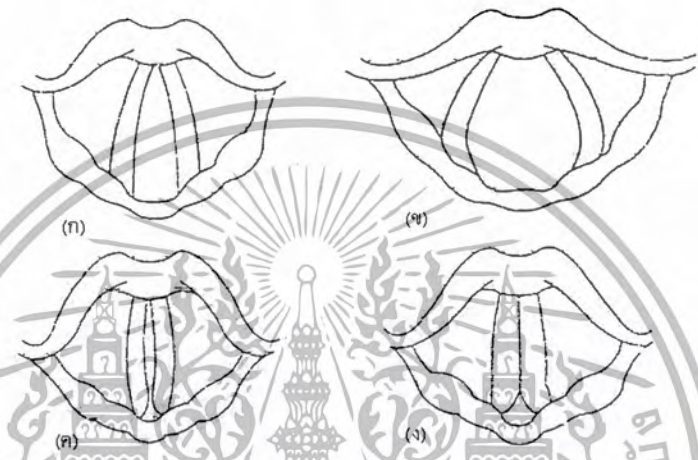


รูปที่ 2.1 ภาพตัดขวางแสดงอวัยวะในระบบการพูดของมนุษย์

1. ปอดและกระบังลม ทำหน้าที่สำคัญในการหายใจ และเป็นต้นกำเนิดการไหลของอากาศในกระบวนการผลิตเสียง

2. หลอดลม ทำหน้าที่นำอากาศจากปอดผ่านกล่องเสียง และเป็นอวัยวะที่อยู่ด้านหน้าของหลอดอาหาร

3. กล่องเสียง เป็นอวัยวะพิเศษที่ทำหน้าที่เป็นทางเดินอากาศเวลาหายใจ และเป็นตัวผลิตพัลส์ (pulse) ของอากาศขณะเปล่งเสียง ซึ่งประกอบด้วยเส้นเสียง (vocal cords) และช่องสายเสียง (glottis) รูปร่างของกล่องเสียง และเส้นเสียงในลักษณะต่าง ๆ แสดงในรูปที่ 2.2



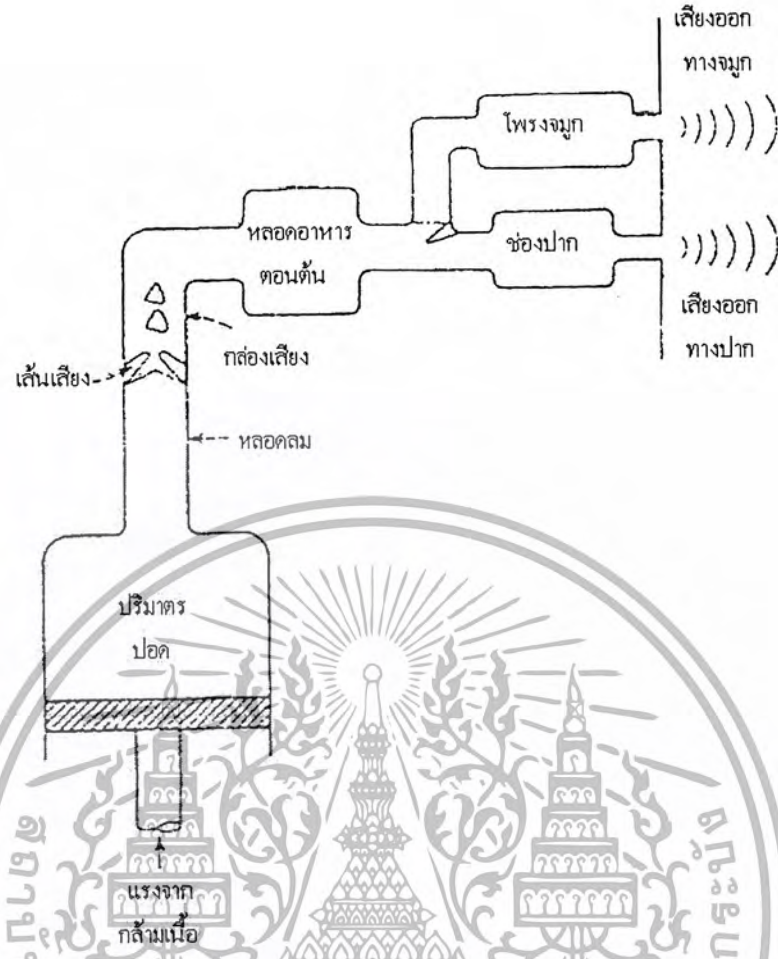
รูปที่ 2.2 รูปกล่องเสียงขณะ
(ก) หายใจปกติ (ข) หายใจเข้าลึก ๆ
(ค) กำลังส่งเสียง (ง) ส่งเสียงกระซิบหรือเสียงแผ่ว

4. ช่องปากและส่วนของหลอดอาหารตอนต้น อวัยวะกลุ่มนี้อยู่ต่อจากกล่องเสียง อาจเรียกว่าอวัยวะกำทอนเสียง (vocal tract) ทำหน้าที่กำทอนเสียง โดยให้กำทอนทั้งเสียงที่เกิดจากกล่องเสียง และเสียงที่เกิดภายในช่องปาก ขนาดของอวัยวะกำทอนเสียงขึ้นอยู่กับตำแหน่งของลิ้น ริมฝีปาก ขากรรไกร และเพดานอ่อน และเปลี่ยนแปลงไปตามการออกเสียง

5. โพรงจมูก เริ่มจากเพดานอ่อนจนถึงรูจมูกทั้งสอง ทำหน้าที่กำทอนเสียงร่วมกับช่องปาก เมื่อมีการเปล่งเสียงที่ออกจากจมูก (nasal sounds) เช่นเสียง /ม/ , /น/ และ /ง/ เป็นต้น

2.2 กระบวนการผลิตเสียงพูด

จากระบบเสียงพูด สามารถแสดงเป็นแผนภาพของระบบกำเนิดเสียง ดังรูปที่ 2.3



รูปที่ 2.3 แผนภาพระบบเสียงพูดของมนุษย์

ซึ่งสามารถจำแนกกลไกการสร้างเสียงพูดของมนุษย์ได้ 3 แบบ ดังนี้

1. อากาศที่ไหลจากปอดจะถูกมอดูเลท (modulate) โดยการสั่นของเส้นเสียงทำให้เกิดคลื่นเสียงลักษณะคล้ายพัลส์ที่มีคาบเวลาแบบควอไซ (quasi-periodic pulse-like excitation)

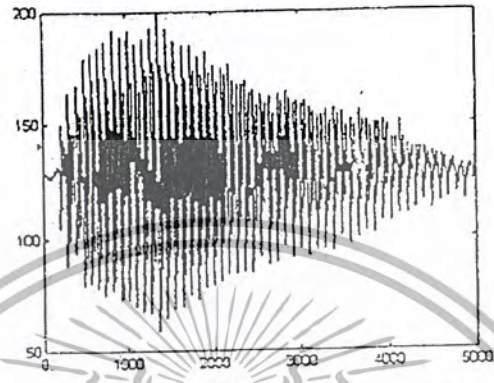
2. อากาศที่ไหลจากปอดถูกทำให้ปั่นป่วนด้วยการบังคับให้ไหลผ่านช่องแคบอันเกิดจากการบีบตัวของอวัยวะในช่องปากทำให้เกิดเสียงลักษณะคล้ายเสียงรบกวน (noise-like excitation)

3. อากาศที่ไหลถูกกัก และเกิดแรงดันอยู่ภายในส่วนของช่องปากที่ปิด จากนั้นจึงปล่อยให้อากาศที่มีแรงดันพุ่งออกไปอย่างเร็ว ทำให้เกิดการกระตุ้นเป็นเสียงในช่วงเริ่มต้น (transient excitation)

2.3 เสียงพูดของมนุษย์

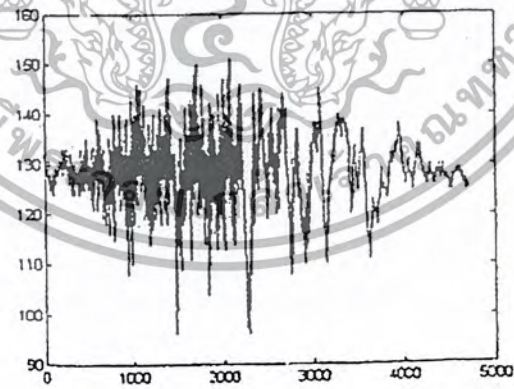
เสียงพูดเป็นคลื่นตามยาว (longitudinal wave) เกิดจากการสั่นของอนุภาคตัวกลางนั่นคือ อากาศ และทิศทางการสั่นของอนุภาค จะอยู่ในทิศเดียวกันกับทิศทางของการเคลื่อนที่ คลื่นเสียงเป็นคลื่นที่เปลี่ยนแปลงไปตามเวลา เสียงพูดแบ่งออกได้เป็น 2 ชนิด ตามการกำเนิดเสียง หรือโหมด (mode) การกระตุ้น คือ

1. เสียงก้องหรือเสียงโฆษะ (voiced) เกิดจากการบังคับอากาศให้ผ่านช่องสายเสียงทำให้มีการเปลี่ยนแปลงความตึงหย่อนของเส้นเสียง โดยเส้นเสียงจะสั่นและเกิดเป็นพัลส์ (pulse) ของอากาศไปกระตุ้นอวัยวะกำทอนเกิดเป็นเสียงก้อง ตัวอย่างเสียงก้องได้แก่ เสียงสระ เสียงพยัญชนะ ที่ต้องออกเสียงจากลำคอ (voiced consonants) เช่น เสียง /a/ ดังรูป 2.4 (ก)



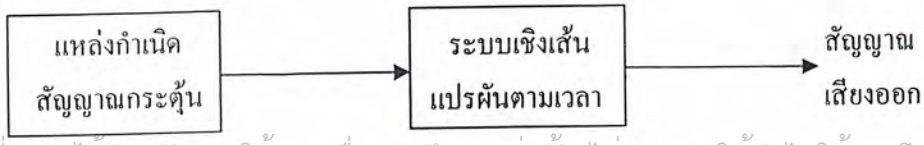
รูปที่ 2.4 (ก) ตัวอย่างรูปคลื่นของสัญญาณเสียงก้อง /a/

2. เสียงไม่ก้องหรืออโฆษะ (unvoiced หรือ voiceless) เป็นเสียงที่เกิดในช่องปาก หรือโพรงจมูก โดยอวัยวะภายในช่องปาก ริมฝีปาก ขวางการไหลของอากาศให้ผ่านได้เป็นช่องเล็ก ๆ อากาศจึงไหลผ่านอย่างรวดเร็ว และปั่นป่วนจนกระทั่งสร้างเป็นเสียงรบกวนช่องความถี่กว้าง (broad-spectrum noise) ตัวอย่างเสียงไม่ก้องได้แก่ เสียงพยัญชนะที่ไม่ได้เกิดจากลำคอ (voiceless consonants) เช่น เสียง /sh/. ดังรูปที่ 2.4 (ข)



รูปที่ 2.4 (ข) ตัวอย่างรูปคลื่นของสัญญาณเสียงไม่ก้อง /sh/

2.4 แบบจำลองระบบกำเนิดเสียงพูด

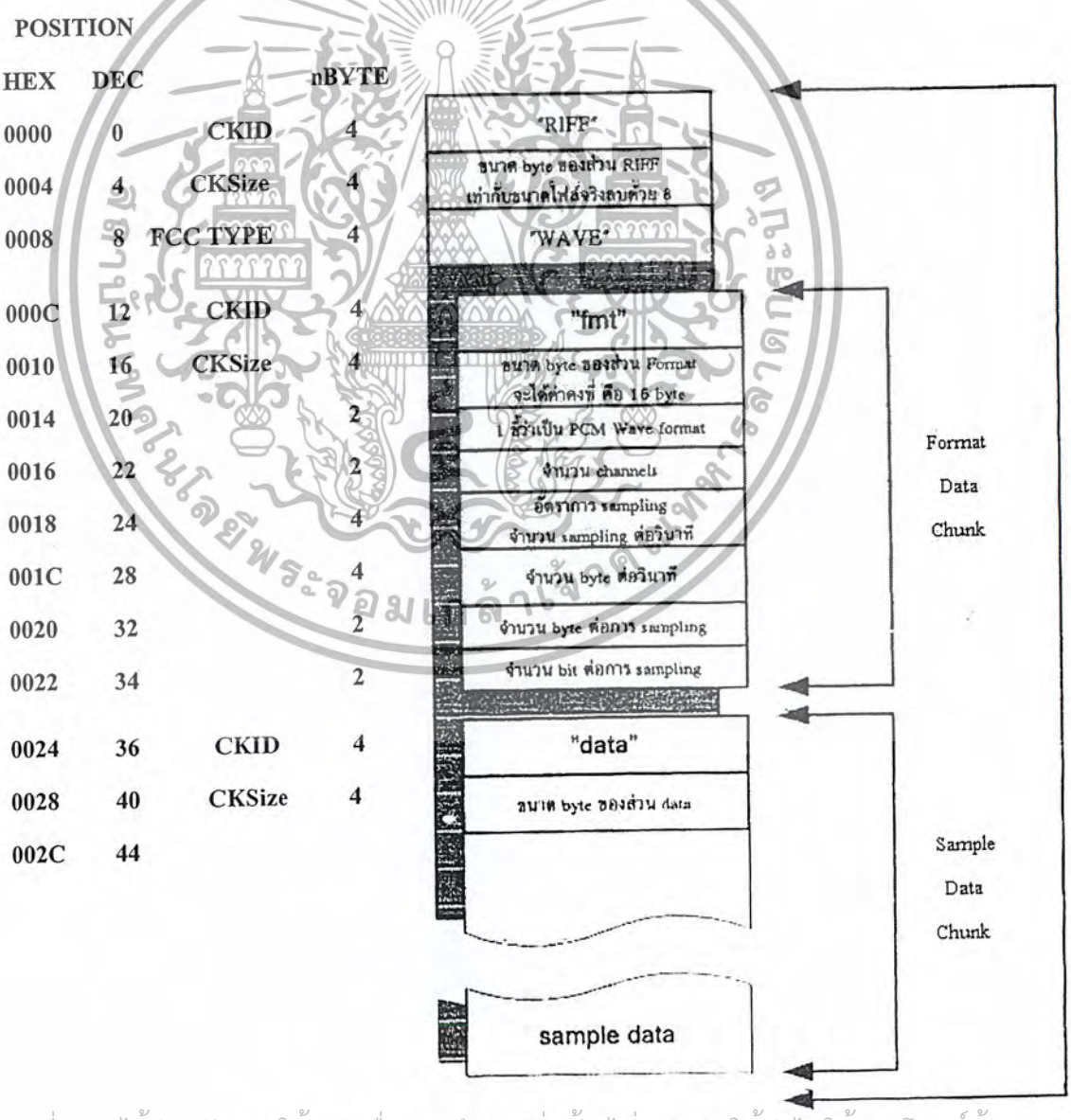


รูปที่ 2.5 แผนภาพกรอบจำลองระบบกำเนิดเสียงเริ่มต้น

จากรูปที่ 2.5 แหล่งกำเนิดสัญญาณกระตุ้นทำหน้าที่แทนการทำงานของปอด และกล่องเสียง ส่วนนี้จะผลิตขบวนพัลส์ที่มีคาบเวลาพิตช์ขณะเปล่งเสียง และให้กำเนิดเสียงซึ่งคล้ายเสียงรบกวนขณะเปล่งเสียงอโฆษะ (unvoiced) ส่วนที่สองเป็นท่อกำทอนเสียง จะแทนการทำงานของช่องปาก และโพรงจมูก ทำหน้าที่เสมือนตัวกรองสัญญาณ (filter) ที่ยอมให้ความถี่ฟอร์แมนท์ผ่านได้ ซึ่งสามารถแทนด้วยระบบเชิงเส้นแปรผันตามเวลา (time-varying linear system)

2.5 รูปแบบของไฟล์เสียง

ในการบันทึกเสียงพูดเป็นข้อมูลดิจิตอลโดยใช้การ์ดเสียง จะทำให้ได้เวฟไฟล์ที่มีรูปแบบของไฟล์ RIFF (Resource International File Format) ซึ่งเป็นรูปแบบมาตรฐานที่ใช้กันอย่างกว้างขวาง ไฟล์ RIFF นั้นมีโครงสร้างเชิงซ้อนคือ มีลักษณะเป็นกลุ่ม (Chunk) ซึ่งภายในกลุ่มนี้ยังประกอบด้วยกลุ่มย่อย ซึ่งมีโครงสร้างคล้ายกัน ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 แสดงโครงสร้างเวฟไฟล์ที่มีรูปแบบของไฟล์ RIFF

รูปแบบการบันทึก	ค่าสูงสุด	ค่าต่ำสุด	ค่าที่ตำแหน่งตรงกลาง
8 บิท พีซีเอ็ม	255 (FFH)	0	128 (80H)
16 บิท พีซีเอ็ม	32767 (7FFFH)	-32768 (-800H)	0

รูปที่ 2.7 แสดงค่าสูงสุด ค่าต่ำสุด ค่ากลางของรูปแบบการบันทึกแต่ละอัน

จากรูปที่ 2.6 จะเห็นว่า กลุ่ม RIFF (RIFF Chunk) จะเป็นกลุ่มใหญ่ที่สุด ซึ่งประกอบด้วยกลุ่มรูปแบบของข้อมูล (Format Data Chunk) และกลุ่มข้อมูลจากการแซมปลิ้ง (Sample Data Chunk) ซึ่งในแต่ละกลุ่มจะมีโครงสร้างที่ประกอบด้วย

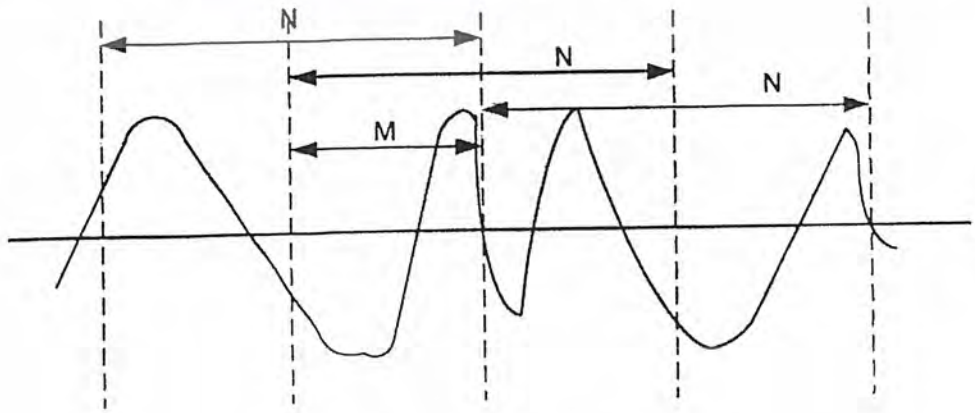
1. ชื่อกลุ่ม (CKID : Chunk Identification) เช่น RIFF , fmt_ , data ซึ่งมีกฎกำหนดว่าต้องมี 4 ตัวอักษร (FCC : Four Character Code) ถ้ามี 3 ตัวอักษรก็จะต้องเพิ่มช่องว่าง (Blank) เข้าไปอีก 1 ตัวอักษรให้รวมเป็น 4 ตัวอักษรในกรณี fmt_

2. ตัวเลขบอกขนาดข้อมูลของกลุ่ม (CKSize : Chunk Size) มีหน่วยเป็นไบต์ (Byte) โดยจะไม่รวมไบต์ที่ใช้ไปในการเก็บชื่อกลุ่ม (CKID) แต่จะเก็บตัวเลขบอกขนาดของกลุ่มเองด้วย ตัวเลขบอกขนาดของกลุ่มจะแสดงด้วยข้อมูล 4 ไบต์โดยไบต์ที่มีความสำคัญต่ำสุด (Least Significant Byte) จะถูกเก็บเข้าไปก่อน

3. ข้อมูลของกลุ่ม (CKData : Chunk Data) มีจำนวนไบต์เท่ากับตัวเลขที่แสดงในข้อที่ 2 เช่น ในกลุ่มรูปแบบของข้อมูล (Format Data Chunk) จะมีข้อมูลของกลุ่มเป็นรูปแบบในการบันทึกเสียง ยกตัวอย่างเช่น อัตราแซมปลิ้ง , จำนวนบิทต่อ 1 แซมเปิล เป็นต้น หรือในกลุ่มข้อมูลจากการแซมปลิ้ง (Sample Data Chunk) จะมีข้อมูลของกลุ่มเป็นแอมปลิจูด (Amplitude) ที่ได้จากการแซมปลิ้ง

2.6 การแบ่งช่วงสัญญาณ (Frame Blocking)

ข้อมูลเสียงซึ่งอยู่ในโดเมนเวลา จะถูกแบ่งออกเป็นช่วงสัญญาณ (Block) ซึ่งแต่ละช่วงสัญญาณจะประกอบด้วยสัญญาณเสียง N สัญญาณ และในแต่ละช่วงสัญญาณจะถูกวิเคราะห์โดยการเลื่อนสัญญาณไปครั้งละ M สัญญาณ ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 แสดงการแบ่งช่วงสัญญาณ

จะเห็นได้ว่าถ้าค่า M มีค่าน้อยกว่าค่า N มากเท่าใด จะทำให้การวิเคราะห์สัญญาณมีความแม่นยำมากเท่านั้น แต่ข้อเสียคือจะต้องใช้เวลาในการวิเคราะห์นานขึ้น และถ้าค่า M มีค่ามากกว่าค่า N แล้วจะทำให้สัญญาณบางส่วนไม่ถูกใช้ในการวิเคราะห์จะเกิดการผิดพลาดในการวิเคราะห์สัญญาณในส่วนต่อไป

การกำหนดขนาดของช่วงสัญญาณมีเงื่อนไขในการเลือกดังนี้

1) ค่า M จะต้องสั้นพอที่ทำให้คุณสมบัติของเสียงไม่เปลี่ยนแปลง
2) ค่า N จะต้องยาวพอที่จำนวนของตัวอย่างมีเพียงพอต่อการหาสัมประสิทธิ์
3) การเลื่อนการวิเคราะห์ (ค่า M) ต้องไม่ข้ามข้อมูล

2.7 การวินโดว์ (Windowing)

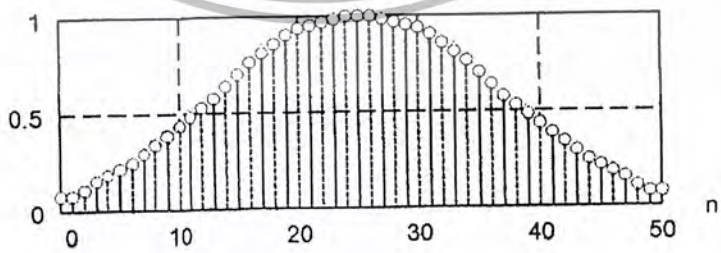
เนื่องจากเราใช้วิธีตัดสัญญาณเพื่อมาหาสเปกตรัมเป็นบล็อก ๆ ซึ่งเท่ากับเป็นการหาสเปกตรัมของ "บล็อกของสัญญาณ" ไม่ใช่ตัวสัญญาณจริง ๆ ที่เข้ามาติดต่อกัน ไม่มีจุดเริ่มต้นหรือสิ้นสุด การตัดสัญญาณเป็นบล็อกนี้จะทำให้เกิดความคลาดเคลื่อนในสเปกตรัมที่ได้ วิธีลดผลของความคลาดเคลื่อนนี้ทำได้โดยคูณสัญญาณแต่ละบล็อก หรือ $X_i(n)$ ด้วยฟังก์ชั่นหน้าต่าง (window function) ก่อนที่จะทำการหาสเปกตรัม

ฟังก์ชั่นหน้าต่างมีหลายแบบ เป็นฟังก์ชั่นที่มีลักษณะสมมาตร และมีค่าสูงสุดที่จุดกึ่งกลาง มีความยาวเท่าไหร่ก็ได้ตามต้องการ (ตามค่าที่แทนลงในสูตร) ตัวอย่างของฟังก์ชั่นหน้าต่างที่เป็นที่นิยมก็คือ หน้าต่างแฮมมิ่ง (Hamming window) ซึ่งมีสมการคือ

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad ;n = 0,1,2,\ldots,N-1 \qquad (2.1)$$

$$x_{inew}(n) = x_i(n)w(n) \qquad (2.2)$$

เราต้องการ w(n) ที่มีความยาวเท่ากับ 1 บล็อกสัญญาณ ดังนั้นต้องใช้ N เท่ากับความยาว 1 บล็อกสัญญาณ รูปร่างของหน้าต่างแบบแฮมมิ่งแสดงดังรูปที่ 2.9 ซึ่งจะสังเกตเห็นได้ว่า ฟังก์ชั่นหน้าต่างมีค่าเล็กที่ส่วนต้นและส่วนปลาย ซึ่งเป็นส่วนที่ลดผลของการเปลี่ยนแปลงที่จุดเริ่มต้น และจุดสิ้นสุดของบล็อก



รูปที่ 2.9 ตัวอย่างของฟังก์ชั่นหน้าต่างแบบแฮมมิ่ง ที่ N = 50

2.8 การวิเคราะห์ฟูริเยร์ (Fourier Analysis)

การวิเคราะห์ฟูริเยร์ช่วงสั้นเป็นเทคนิคการหาความถี่ที่ใช้กันมานานแล้ว การวิเคราะห์ฟูริเยร์ให้ตัวแทนของสัญญาณเสียงพูดเป็นฟังก์ชั่นความถี่ในเทอมของขนาดและเฟส เนื่องจากเสียงพูดไม่ได้นิ่งตลอดเวลา ดังนั้นจึงจำเป็นต้องทำการวิเคราะห์ในช่วงสั้น ๆ โดยใช้วินโดว์

การแปลงฟูริเยร์ช่วงสั้นมีสมการดังนี้

$$X_n(e^{-jw}) = \sum_{m=-\infty}^{\infty} x(m)[\exp(-jwm)w(n-m)] \qquad (2.3)$$

ในการคำนวณต้องใช้ DFT แทนการแปลงฟูริเยร์แบบต่อเนื่อง โดยใช้ฟังก์ชั่นวินโดว์ลดข้อมูลแบบไม่ต่อเนื่องทั้งหมดให้เหลือจำนวน N ตัว (N คือช่วงเวลาหรือขนาดของวินโดว์ที่ใช้ในการแปลง DFT) ข่าวสารต่าง ๆ ใน $X_n(e^{-jw})$ จะไม่สูญหายไปจากข้อมูลเดิมถ้าการแปลงนั้นสุ่มมาด้วยความถี่สูงเพียงพอ (คือช่วงระยะห่างระหว่าง N) และวินโดว์ w(n) ไม่มีจุดสุ่มที่เป็นศูนย์ตลอดช่วง N ตัวแปร N เป็นตัวแปรที่จะต้องระวังมากเป็นพิเศษในการวิเคราะห์ความถี่ช่วงสั้น ถ้าค่าของ N ต่ำจะทำให้ความละเอียดในโดเมนความถี่หยาบมาก เพราะจะให้ผลที่ดีในโดเมนเวลา เพราะการเฉลี่ยถูกทำในช่วงสั้น ๆ เท่านั้น ในทางตรงกันข้ามถ้า N มีขนาดใหญ่จะให้ผลของความละเอียดเวลาที่แย่ในโดเมนเวลา แต่จะทำให้โดเมนความถี่มีความละเอียดสูงกว่า

2.8.1 การแปลงฟาสท์ฟูริเยร์ (Fast Fourier Transform)

โดยทั่วไปการแปลงฟูริเยร์ (Discrete Fourier Transform) เป็นการคำนวณที่ใช้เวลาค่อนข้างมาก ลำดับขั้นตอนในการคำนวณ DFT ให้เร็วขึ้นเรียกว่า ฟาสท์ฟูริเยร์ (Fast Fourier Transform หรือ FFT) โดยการแปลง FFT แบ่งได้เป็น 2 ชนิดใหญ่ ๆ คือ ชนิดลดทอนทางเวลา (Decimation In Time หรือ DIT) และชนิดลดทอนทางความถี่ (Decimation In Frequency หรือ DIF) สำหรับในส่วนนี้จะแสดงเฉพาะชนิดลดทอนทางเวลาซึ่งเกี่ยวกับในโครงงานนี้เท่านั้น

2.8.2 ขั้นตอนวิธีลดทอนทางเวลา

วิธีนี้เป็นการจัดแบ่งกลุ่มลำดับสัญญาณในโดเมนเวลา X(n) ขนาด N จุดออกเป็น 2 ลำดับสัญญาณขนาด N/2 จุดเท่ากันคือ ลำดับคู่และลำดับคี่ โดยที่ลำดับคู่เกิดจากการเอาลำดับในตำแหน่งคู่มาเรียงกัน ที่เหลือเป็นลำดับคี่ ดังนั้นจะได้

$$x_c(m) = x(2n) \qquad ; m = 0,1,2,...,(N/2)-1$$
$$x_o(m) = x(2n+1) \qquad ; m = 0,1,2,...,(N/2)-1 \qquad (2.4)$$

ถ้าให้ W_N เท่ากับ exp(-2j/N) จะทำให้การคำนวณ DFT ของลำดับ x(n) ที่ยาว N จุด สามารถเขียนใหม่ได้เป็น

$$X(k) = \sum_{m=0}^{N/2-1} x_e(2m)W_k^{2km} + \sum_{m=0}^{N/2-1} x(2m+1)W_N^{(2m+1)k} \qquad (2.5)$$

โดยที่

$$W_N^2 = \{\exp[j2\pi / N]^2\} = \exp[j2\pi / N/2] = W_{N/2} \qquad (2.6)$$

ซึ่ง $W_{N/2}$ เป็นค่า W ลำดับความยาว N/2 จุด สมการ 2.5 สามารถเขียนใหม่ได้เป็น

$$X(k) = \sum_{m=0}^{N/2-1} x_e(m) W_{N/2}^{km} + \sum_{m=0}^{N/2-1} x(m) W_{N/2}^{km} \qquad (2.7)$$

การนำผลการแปลง DFT ขนาด 2 จุด จำนวน N/2 ภาคมารวมกัน เพื่อให้เป็นการคำนวณ DFT ขนาด N จุด จะต้องมีหลักเกณฑ์ที่ถูกต้องด้วย จากสมการ 2.7 ถ้าเขียนให้อยู่ช่วง $0 \leq k \leq N/2 - 1$ สามารถเขียนใหม่ได้เป็น

$$X(k) = X_e(k) + (W_N^k) X_0(k) \qquad ;0 \leq k \leq N/2-1 \qquad (2.8)$$

$$= X_e(k - N/2) + (W_N^k) X_0(k - N/2) \quad ;0 \leq k \leq N/2-1$$

เทอม W_N^k เรียกว่า ตัวประกอบหมุน (Twiddle Factor) ซึ่งใช้ร่วมกันกับ DFT ขนาด 2 จุด หรือขนาด N/2 จุด ในการนำมาประกอบเป็น DFT ขนาด N จุดได้เหมือนเดิม และจากความสัมพันธ์

$$(W)_N^{k-N/2} = -(W)_N^k$$

จะได้

$$X(k) = X_1(k) + (W)_N^k X_2(k) \qquad ;0 \leq k \leq N/2-1$$

$$= X_1(k - N/2) + (W)_N^{k-N/2} X_2(k - N/2) \; ; N/2 \leq k \leq N/2-1 \qquad (2.9)$$

ผลจากสมการนี้สามารถนำไปใช้สร้างหน่วยคำนวณที่เรียกว่า หน่วยผีเสื้อ (Butterfly Unit) โดยมีข้อมูลเข้าคือ A และ B และข้อมูลออก คือ X และ Y เป็น

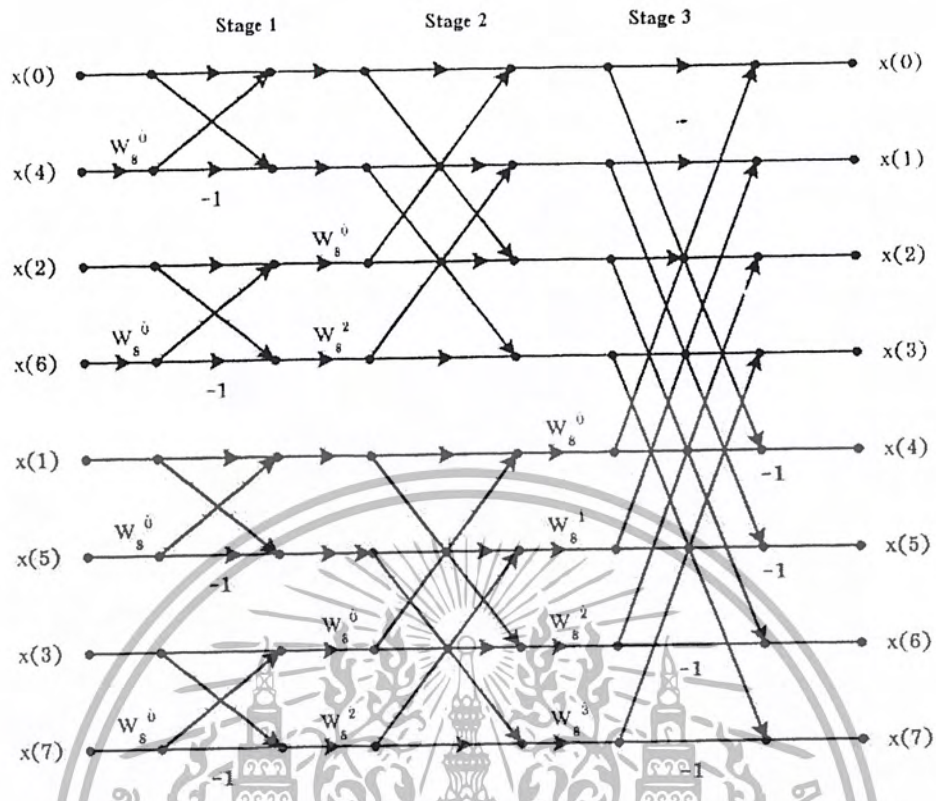
$$X = A + (W)_N^k B$$

$$Y = A + (W)_N^k B \qquad (2.10)$$

ซึ่งสามารถเขียนอธิบายแทนด้วยโฟล์วชาร์ทดังแสดงในรูป 2.10

รูปที่ 2.10 แสดงหน่วยผีเสื้อของการคำนวณตามขั้นตอนวิธีลดทอนทางเวลา

สำหรับตัวอย่างการคำนวณ DFT โดยใช้ FFT แสดงดังรูป 2.11

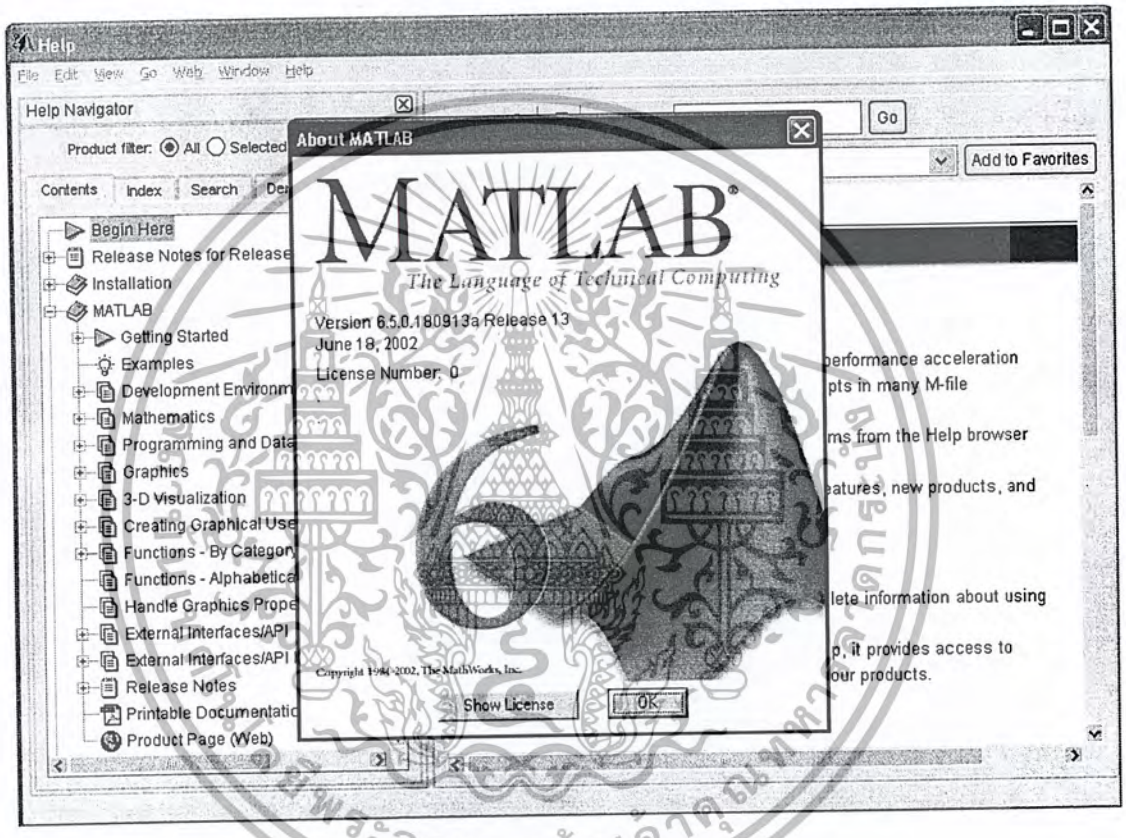


รูปที่ 2.11 แสดงวิธีการของ FFT แบบลดทอนทางเวลา (DIT) สำหรับข้อมูลขนาดจุด

บทที่ 3
การออกแบบและโครงสร้างโปรแกรม

3.1 การรู้จำเสียงคำสั่ง

ทำการทดสอบทฤษฎีและการทำงานโดยใช้ Matlab 6.5 เนื่องจากมี Toolbox สำหรับการทดสอบสัญญาณเสียง และใช้งานได้ง่าย

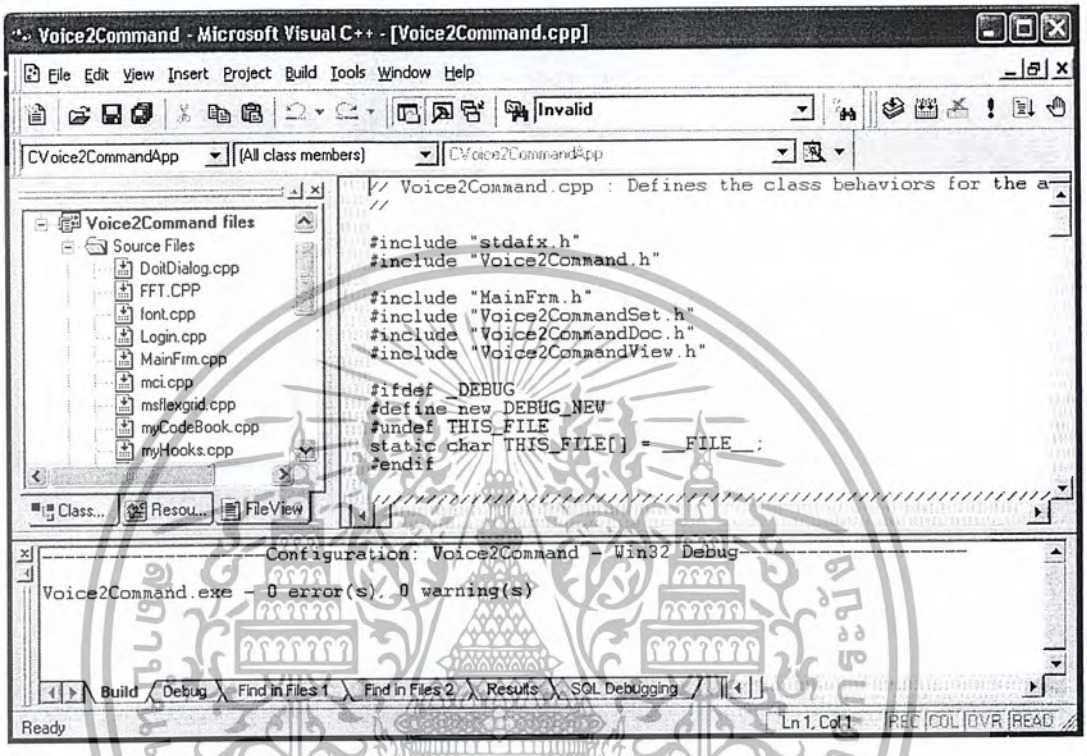


รูปที่ 3.1 แสดงการใช้งาน MATLAB 6.5

ส่วนขั้นตอนการพัฒนา พัฒนาด้วย Visual C++ 6.0 พัฒนาด้วยรูปแบบ MFC เพื่อให้สะดวกต่อการนำไปรวมกับส่วนโปรแกรมดักจับแมสเซจ โดยออกแบบให้ส่วนนี้เป็นคลาส แล้วเรียกใช้ผ่านเมธอดของออบเจ็คแทนการเรียกใช้แบบฟังก์ชัน ทำให้การแก้ไขและพัฒนาโมดูลทำได้ง่ายและสะดวก

3.2 การบันทึก/ใช้งานเสียง

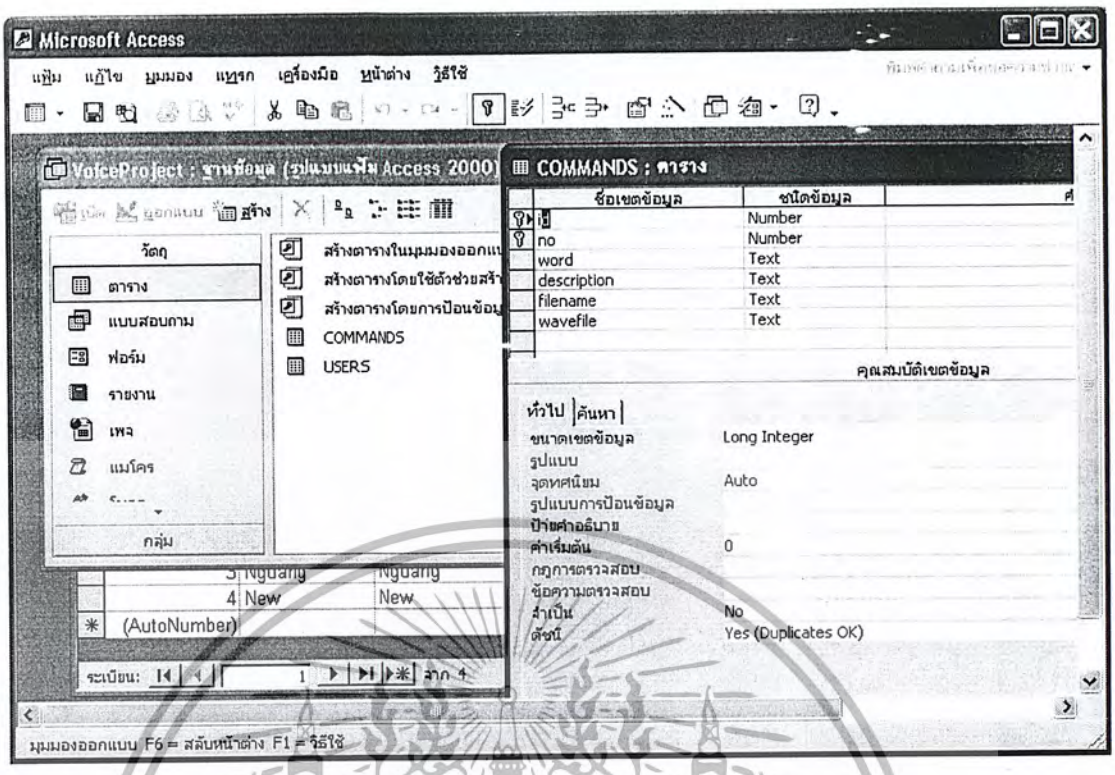
พัฒนาด้วย Visual C++ 6.0 โดยใช้ ActiveX Control ที่ชื่อ Microsoft Multimedia Control 6.0 และ WAVE class เพราะ MMC ใช้งานง่าย และมีความสามารถเพียงพอต่อการใช้งานในระบบ โดย WAVE class ใช้ในส่วนที่ทำการดึงข้อมูลเสียงมาวิเคราะห์



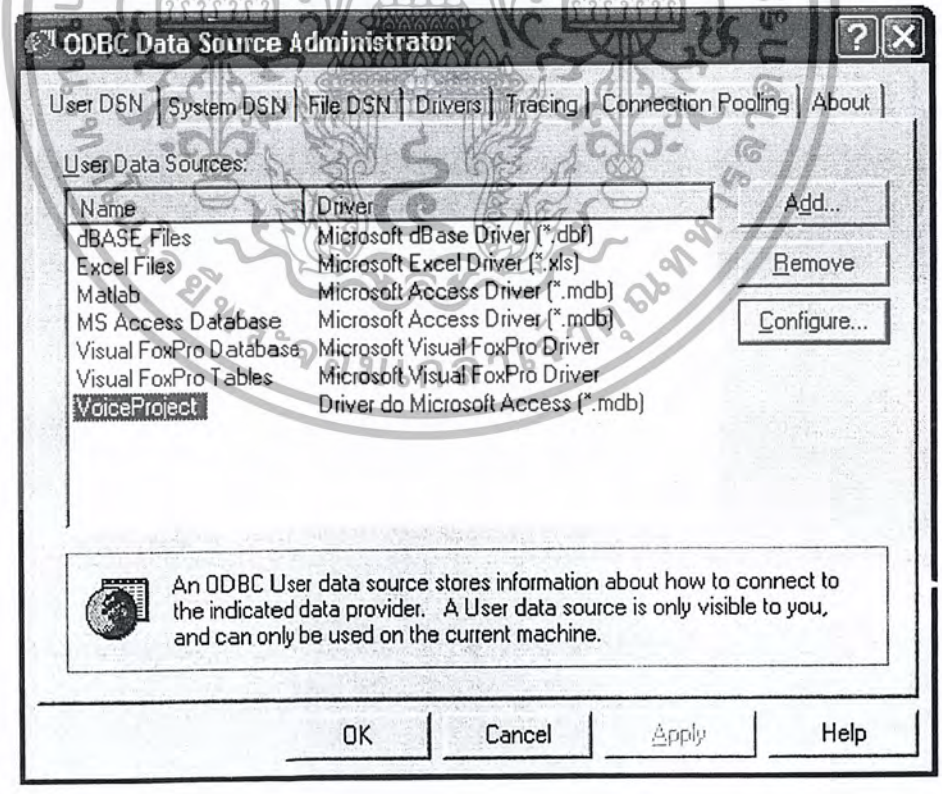
รูปที่ 3.2 แสดงการใช้งาน Visual C++ 6.0

3.3 การติดต่อระบบฐานข้อมูล

พัฒนาด้วย Microsoft Access XP โดยติดต่อผ่าน ODBC เนื่องจากต้องการให้ผู้ใช้งานสะดวกในการใช้งานเนื่องจากเป็นของบริษัทไมโครซอฟท์ ทำให้สามารถเรียกใช้งานได้เลย



รูปที่ 3.3 แสดงการใช้งาน Microsoft Access XP



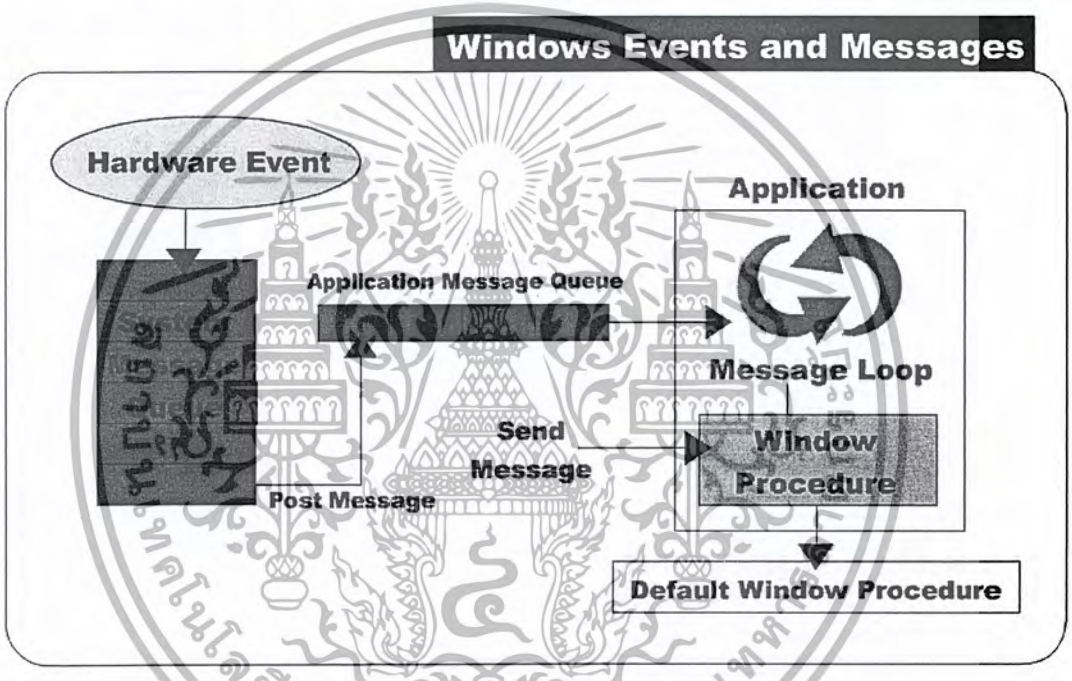
รูปที่ 3.4 แสดงการเซ็ต ODBC

3.4 การบันทึก/เล่นซ้ำการทำงาน

พัฒนาด้วย Visual C++ 6.0 พัฒนาด้วยรูปแบบ Win32API เพราะ เป็นการเข้าไปจัดการระบบวินโดว์แมสเซจซึ่งถูกซ่อนไว้ในการพัฒนาแบบ MFC

3.4.1 การเขียนโปรแกรมดักจับแมสเซจของการทำงานของระบบด้วยวินโดว์ฮุค

การทำงานของระบบปฏิบัติการ Microsoft Windows ทุกรุ่น จะเป็นแบบ Graphics User Interface (GUI) โดยระบบจะรับคำสั่ง (Event) ต่างๆจากผู้ใช้ทางอุปกรณ์ (Input Devices) แล้วส่งมาให้ระบบปฏิบัติการทำการตรวจสอบว่าเป็นคำสั่งของงาน (Task) ใด แล้วจึงทำการส่งค่า Windows Messages ไปให้โปรแกรมนั้น ๆ ทำการตัดสินว่าจะให้ระบบทำงานอย่างไรต่อไป



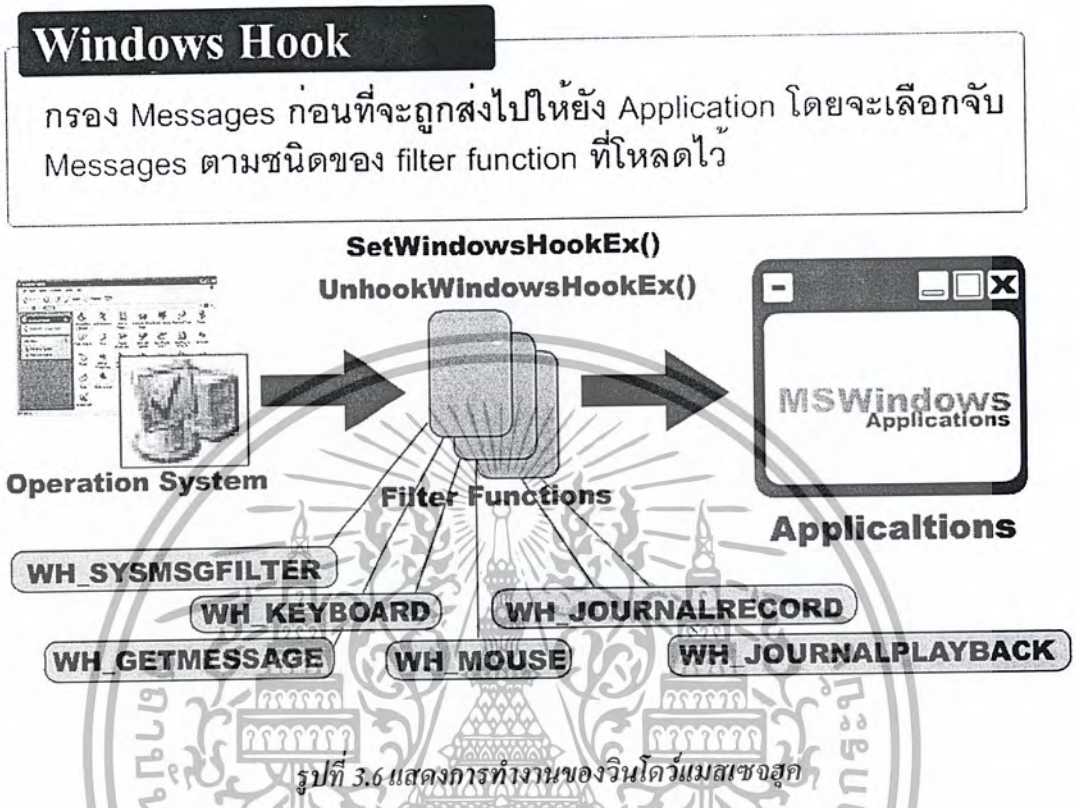
รูปที่ 3.5 แสดงการทำงานของระบบแมสเซจของระบบปฏิบัติการวินโดว์

โดยในส่วนของ Windows Messages นั้นจะมีหลากหลายชนิด ขึ้นอยู่กับประเภทของสิ่งที่สร้างตัว Windows Messages ขึ้นมา ถ้าเป็นอุปกรณ์ (Input Devices) ก็มีหลายอย่าง เช่น WM_MOUSEMOVE (Mouse Event) ,WM_KEYDOWN, WM_KEYUP (Keyboard Event) ถ้า Windows Control บางตัวเป็นตัวสร้าง เช่น WM_COMMAND , WM_DESTROY , WM_PAINT เป็นต้น

จากหลักการที่กล่าวมาข้างต้น จึงเกิดสมมติฐานขึ้นว่า ถ้าหากเราสามารถทำการเก็บ Windows Messages ที่เกิดจากการทำงานของผู้ใช้ไว้ได้ แล้วสามารถนำมาทำการ Replay โดยทำการส่ง Windows Message ให้ระบบเสียเองการทำงานซ้ำก็น่าจะสามารถทำได้

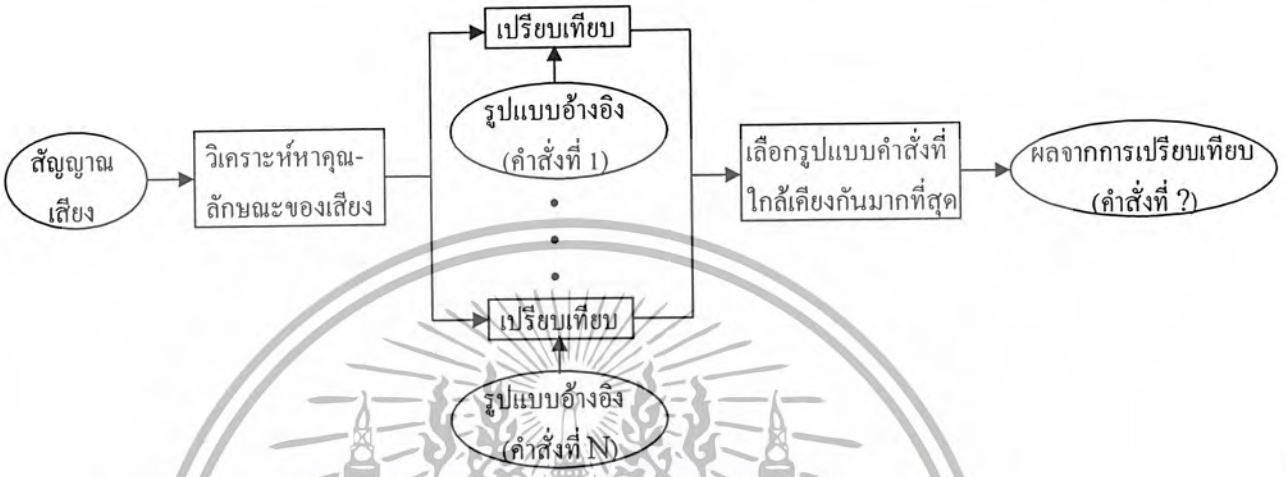
จากการศึกษาจาก MSDN ก็พบว่ามีคำสั่งที่สนับสนุนแนวคิดนี้อยู่คือ Windows Hook Filter โดยตัว Windows Hook จะทำหน้าที่เป็นเสมือนตัวกรองคำสั่ง Windows Messages ที่ส่งผ่าน Windows Queue

ของระบบ และเลือกเอาเฉพาะ Windows Message ตามประเภทที่ของ Hook ที่เราติดตั้งไว้ แล้วทำการ copy ออกมาเพื่อนำมาใช้สำหรับการทำซ้ำ/บันทึกต่อไป



รูปที่ 3.6 แสดงการทำงานของวินโดว์แมสเซจฮุค

บทที่ 4
การรู้จำเสียงคำสั่ง



รูปที่ 4.1 แสดงโครงสร้างของการรู้จำเสียงคำสั่ง

4.1 ภาคการเรียนรู้

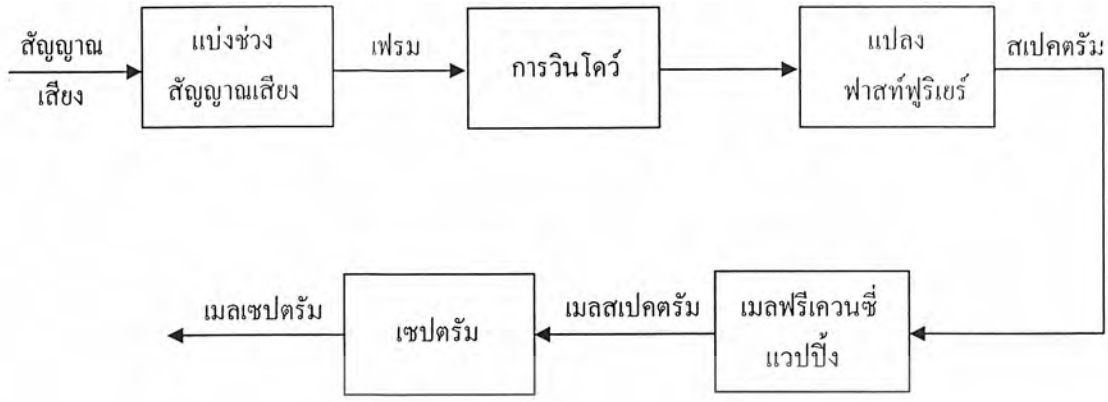
ในภาคนี้เสียงที่เข้ามาในแต่ละคำสั่งจะถูกสุ่มขึ้นมา และแทนด้วยพารามิเตอร์ในรูปแบบที่สะดวกในการใช้งาน โดยนำไปสร้างรูปแบบอ้างอิง (reference model) สำหรับทุกคำสั่งที่เข้ามา และรูปแบบอ้างอิงจะถูกใช้ในการรู้จำเสียงคำสั่งในภาคการทดสอบ ภาคการเรียนรู้ประกอบด้วยขั้นตอนดังนี้

4.1.1 การวิเคราะห์หาคุณลักษณะของเสียง (Feature Extraction)

ขั้นตอนนี้ใช้ในการแปลงรูปแบบคำสั่งให้อยู่ในรูปแบบพารามิเตอร์ เพื่อใช้ในการวิเคราะห์และใช้ในกระบวนการต่าง ๆ โดยใช้การวิเคราะห์สเปคตรัมช่วงสั้น เทคนิคที่ใช้ในการหาพารามิเตอร์ที่เป็นตัวแทนเสียงมีหลายวิธี เช่น ลิเนียร์พรีดิกชั่นโค้ดดิ้ง (Linear Preduction Coding หรือ LPC), เมลฟรีเควนซี่เซปตรัมโคเอฟฟิเซี่ยน (Mel-Frequency Cepstrum Coefficients หรือ MFCC) เราเลือกใช้วิธีเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเซี่ยน เพราะเป็นที่นิยมใช้กันอย่างกว้างขวางและมีความถูกต้องสูง

4.1.2 การประมวลผลของเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเซี่ยน (Mel-Frequency Cepstrum Coefficients Processor)

แผนภาพโครงสร้างของการประมวลผลเอ็มเอฟซีซีแสดงดังรูปที่ 4.2



รูปที่ 4.2 แสดงโครงสร้างของการประมวลผลเอ็มเอฟซีซี

โดยเราเลือกความถี่ในการสุ่ม 11025 เฮิร์ต ซึ่งความถี่นี้ครอบคลุมความถี่ของเสียงพูดมนุษย์ โดยจุดประสงค์ของการประมวลผลเอ็มเอฟซีซีคือการเลียนแบบพฤติกรรมของหูมนุษย์

4.1.2.1 การแบ่งช่วงสัญญาณ (Frame Blocking)

เสียงต้นฉบับจะถูกแบ่งเป็นบล็อก (block) แต่ละบล็อกประกอบด้วย 1024 แซมเปิลและบล็อกที่ติดกันมีช่วงที่ซ้อนทับกัน 512 แซมเปิล โดยบางแซมเปิลที่จุดสิ้นสุดของไฟล์ไม่ได้นำมาใช้ในการคำนวณ

4.1.2.2 การวินโดว์ (Windowing)

แต่ละบล็อกจะถูกวินโดว์เพื่อลดผลของการเปลี่ยนแปลงที่จุดเริ่มต้นและจุดสิ้นสุดของบล็อก เราเลือกใช้หน้าต่างแฮมมิ่ง (Hamming Window) ดังสมการ (4.1)

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad ;n = 0,1,2,\ldots,N-1 \qquad (4.1)$$

โดยฟังก์ชั่นหน้าต่าง (window function) จะถูกคูณกับเมตริกซ์ (matrix) ที่ได้จากการแบ่งช่วงสัญญาณ

4.1.2.3 การแปลงฟาสท์ฟูริเยร์ (Fast Fourier Transform หรือ FFT)

นำบล็อกที่ผ่านการวินโดว์แล้วมาผ่านการแปลงฟาสท์ฟูริเยร์ เพื่อแปลงข้อมูลจากโดเมนเวลาสู่โดเมนความถี่ และสร้างสัมประสิทธิ์สเปกตรัม

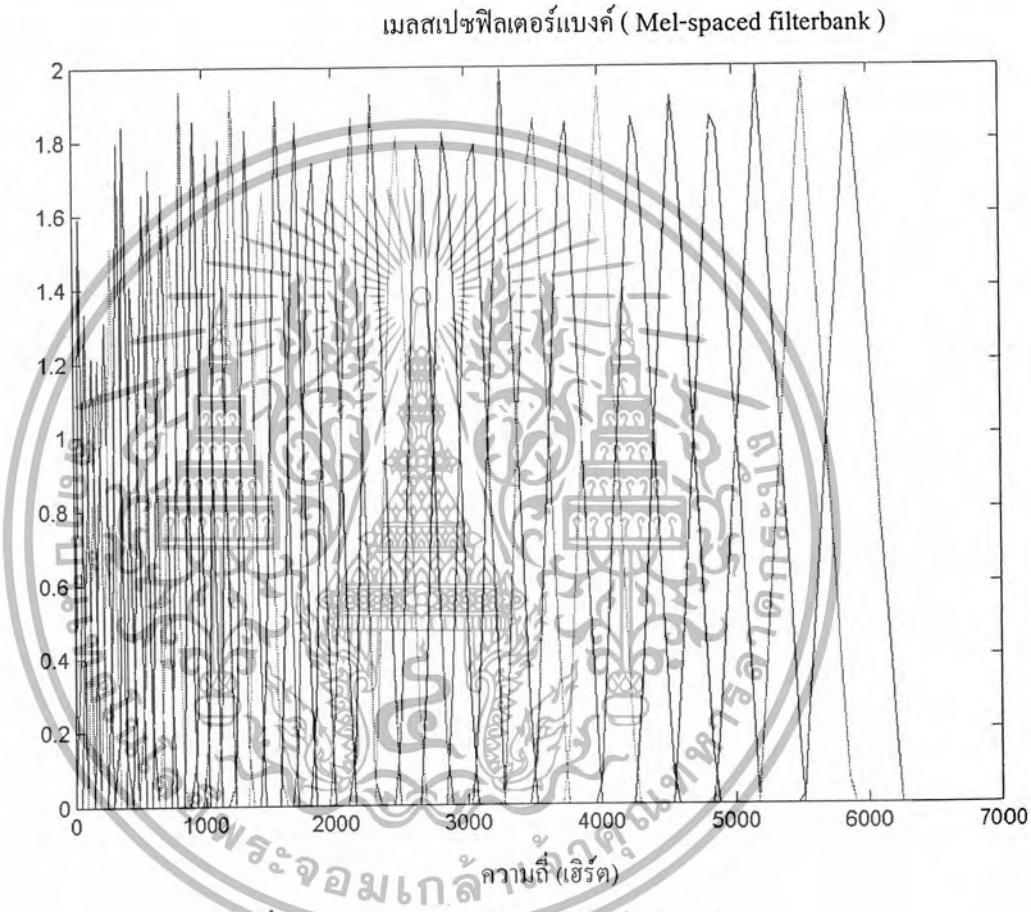
4.1.2.4 เมลฟรีเควนซี่แวปปิ้ง (Mel-Frequency Wrapping)

นำการแปลงเมลฟรีเควนซี่มาใช้ในการแปลงแต่ละสเปกตรัมให้อยู่ในรูปแบบเมลสเกล (melscale) โดยค่าที่อ้างอิงคือ ระดับเสียง 1 กิโลเฮิร์ตและ 40 เดซิเบลเหนือค่าขอบเขตการได้ยินของเพอเซปตรัม

(perceptral) นิยามเป็น 1000 เมล (mel) สูตรตามสมการ (4.2)ใช้ในการคำนวณเมล จากความถี่ f ที่ได้

$$mel(f) = 2595 * \log_{10}(1 + f/700) \qquad (4.2)$$

สเปกตรัมสามารถสร้าง (simulate) โดยใช้ฟิลเตอร์แบงค์ที่มีการกระจายตัวด้วยระยะห่างที่สม่ำเสมอบนเมลสเกล ฟิลเตอร์แบงค์ตอบสนองต่อการแบนด์พาสความถี่เป็นรูปสามเหลี่ยม โดยระยะห่างและแบนด์วิดธ์จะถูกกำหนดโดยค่าความแตกต่างระหว่างค่าคงที่ของความถี่เมล เราเลือกฟิลเตอร์แบงค์ขนาด 40 ฟิลเตอร์แบงค์ที่เราสร้างขึ้นเป็นดังรูปที่ 4.3



รูปที่ 4.3 แสดงเมลสเปซฟิลเตอร์แบงค์ขนาด 40

4.1.2.5 เซปตรัม (Cepstrum)

ใช้ฟังก์ชันล็อก (Log) เพื่อแปลงเมลสเปกตรัมกลับสู่โดเมนเวลา เพื่อหาเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเชี่ยนเนื่องจากเมลสเปกตรัมโคเอฟฟิเชี่ยนเป็นจำนวนจริง เราจึงใช้การแปลงดิสครีสโคซายน์ (Discrete Cosine Transform หรือ DCT) โดยเอ็มเอฟซีซีคำนวณได้จากสมการ (4.3)

$$\widetilde{c}_n = \sum_{k=1}^{K} (\log \widetilde{S}_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K} \right] \qquad ;n = 1,2,...,K \qquad (4.3)$$

ซึ่ง $\widetilde{S}_k, k = 1,2,...,K$ แทนสัมประสิทธิ์เมลสเปกตรัม

เราจะได้เมลฟรีเควนซี่เซปตรัมโคเอฟฟิเชี่ยนสำหรับเสียงในแต่ละช่วงสัญญาณซึ่งเราเรียกว่า อคูสติกเวกเตอร์ (acoustic vector)

ในแถวแรกของเมตริกซ์ที่ได้เราไม่นำมาใช้ เพราะเป็นค่าเฉลี่ยของสัญญาณอินพุทซึ่งเป็นข้อมูลที่มีความสำคัญน้อยมาก ทำให้เราได้เมตริกซ์ขนาด 39 แถว

4.1.3 การสร้างรูปแบบอ้างอิง

เพื่อหาพารามิเตอร์ที่เป็นเป็นตัวแทนของคำสั่ง เราจึงสร้างรูปแบบอ้างอิงสำหรับแต่ละคำสั่ง โดยเทคนิคในการเปรียบเทียบคุณลักษณะของเสียงที่ใช้ในการรู้จำเสียงมีหลายวิธี เช่น ไดนามิกไทม์วาปปิ้ง (Dynamic Time Warping หรือ DTW) , ฮิดเดนมาคอฟโมเดล (Hidden Markov Modeling หรือ HMM) และ เวคเตอร์ควอนไตเซชั่น (Vector Quantization หรือ VQ) เราเลือกใช้วิธีเวคเตอร์ควอนไตเซชั่น เพราะเป็นวิธีที่มีความถูกต้องสูง โดยโค้ดบุค (codebook) จะสร้างจากคำสั่งที่ได้จากการอัดในครั้งแรก แล้วทำการคลัสเตอร์ริ่งเวคเตอร์การเรียนรู้ (clustering training vectors)

4.1.3.1 อัลกอริทึมในการสร้างเวคเตอร์ควอนไตเซชั่นโค้ดบุค

ส่วนการเรียนรู้นี้จะทำการหาเซ็ต (set) ของอคูสติกเวคเตอร์สำหรับทุกคำสั่ง โดยลักษณะเฉพาะของเวคเตอร์ควอนไตเซชั่นโค้ดบุค ได้จากเวคเตอร์การเรียนรู้โดยใช้อัลกอริทึมของ Linde Buzo Gray (LBG) โดยอัลกอริทึมนี้สามารถอิมพลีเมนต์ (implement) โดยการทำซ้ำขั้นตอนข้างล่างนี้

1. ออกแบบ 1 เวคเตอร์โค้ดบุค ซึ่งก็คือเซนทรอยด์ (centroid) ของเซ็ตเวคเตอร์การเรียนรู้ทั้งหมด
2. เพิ่มขนาดของโค้ดบุคเป็น 2 เท่าโดยแบ่งโค้ดบุคปัจจุบัน โดย y_n ได้จาก

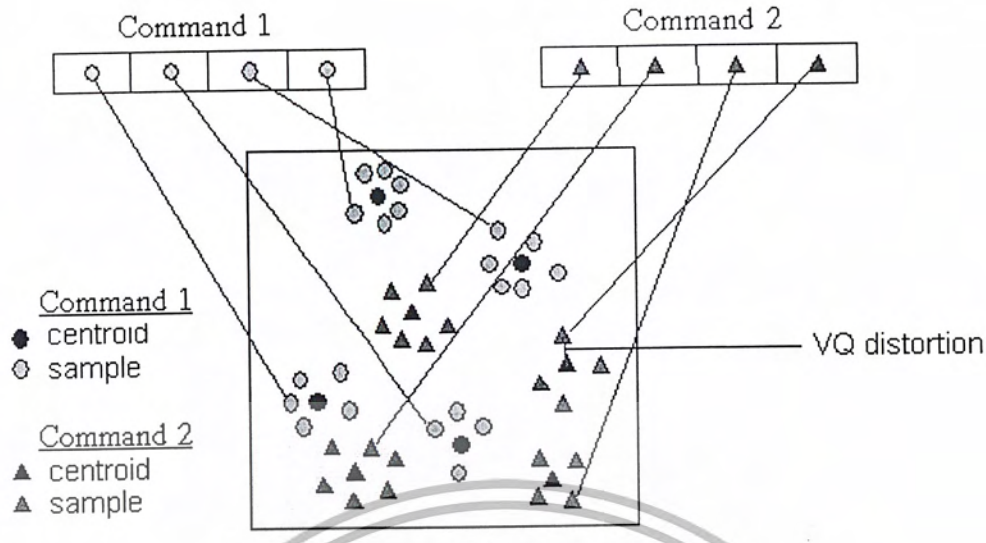
$$y_n^+ = y_n(1+\varepsilon) \qquad (4.4)$$

$$y_n^- = y_n(1-\varepsilon) \qquad (4.5)$$

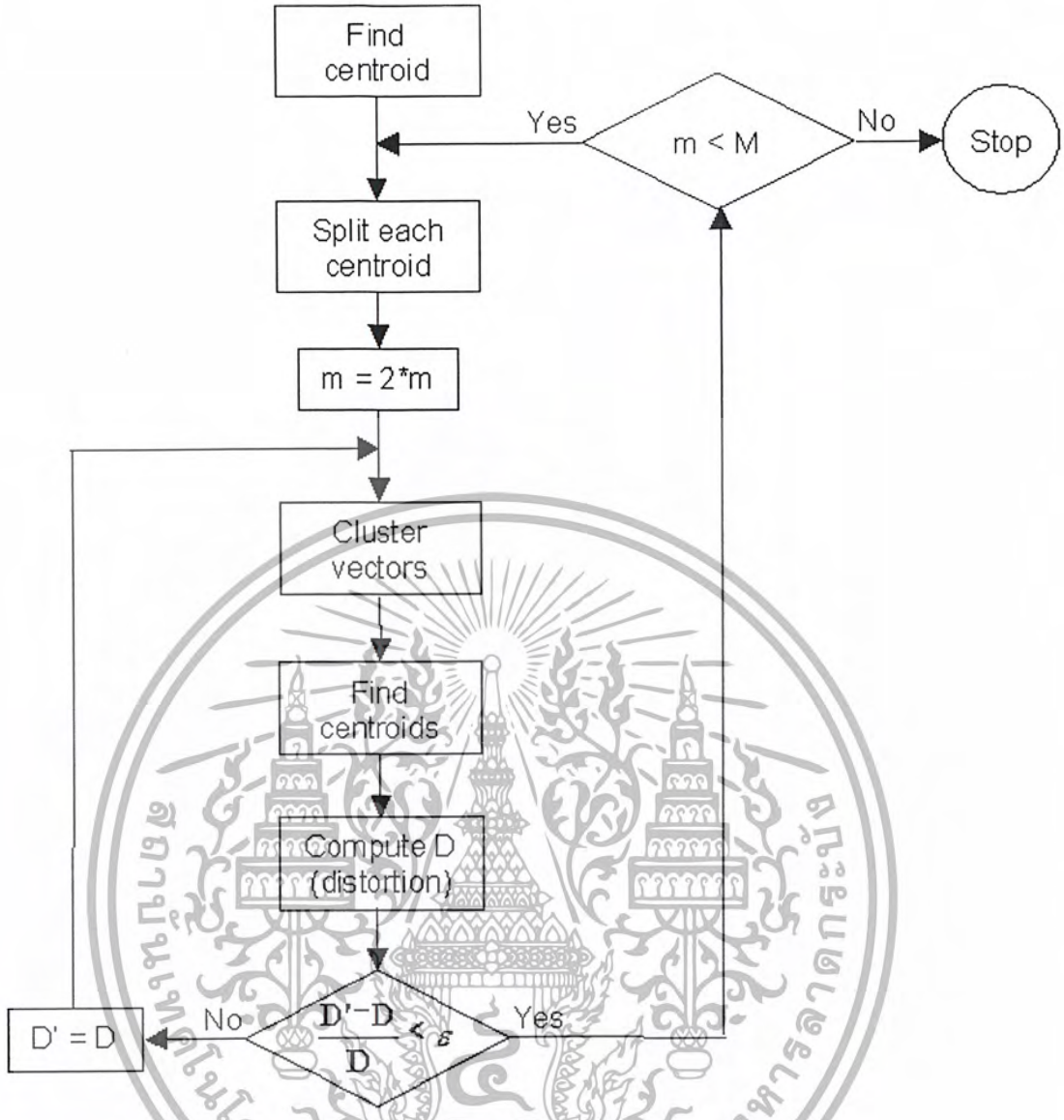
โดย n คือ ค่าตั้งแต่ 1 ถึง ขนาดปัจจุบันของโค้ดบุค

ε คือ พารามิเตอร์ที่ใช้ในการแบ่ง

3. หา Nearest- Neighbor สำหรับแต่ละเวคเตอร์การเรียนรู้ โดยการหาโค้ดเวิร์ด (codeword) ในโค้ดบุคปัจจุบันที่ใกล้เคียงกันมากที่สุด และกำหนดให้เวคเตอร์นั้นไปอยู่ที่เซล (cell) ที่มีลักษณะเช่นเดียวกัน
4. ปรับปรุงเซนทรอยด์ ปรับปรุงโค้ดเวิร์ดในแต่ละเซลโดยใช้เซนทรอยด์ของเวคเตอร์การเรียนรู้กำหนดไปในเซล
5. รอบที่ 1 ทำซ้ำ ข้อ 3 และ 4 จนกระทั่งระยะทางเฉลี่ยต่ำกว่าค่าขอบเขตที่ตั้งไว้ก่อน
6. รอบที่ 2 ทำซ้ำ ข้อ 2 , 3 และ 4 จนกระทั่งได้โค้ดบุคขนาด M ที่ต้องการออกแบบ ในที่นี้เราใช้ M = 16 ทำให้ได้เมตริกซ์ขนาด 16 คอลัมน์



รูปที่ 4.4 แผนภาพแสดงการหาโค้ดบุคของวิธีเวคเตอร์ควอนไตเซชั่น



รูปที่ 4.5 แสดงโฟล์วชาร์ทของแอลบีจีอัลกอริทึม

ซึ่ง ε คือ พารามิเตอร์ที่ใช้ในการแบ่ง

D คือ ดิสทอชั่น

D′ คือ ดิสทอชั่นที่ใช้ในการแบ่ง

M คือ จำนวนเซนทรอยด์ที่ต้องการ

m คือ ขนาดปัจจุบันของโค้ดบุค

โค้ดบุคของแต่ละคำสั่งมีขนาดเมตริกซ์ 39*16 แต่ละโค้ดบุคประกอบด้วย 16 เซนทรอยด์ ดังนั้นแต่ละคอลัมน์ของเมตริกซ์แทนเซนทรอยด์

4.2 ภาคการทดสอบ

นำคำสั่งที่ต้องการทดสอบมาผ่านขั้นตอนเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเชี่ยน แล้วนำพารามิเตอร์ที่ได้มาเปรียบเทียบกับโค้ดบุค โดยใช้วิธียูคลิเดียนดิสแทนซ์ (Euclidean distance) จะได้ค่าดิสทอชั่น (distortion) ของแต่ละรูปแบบอ้างอิงออกมา ค่าดิสทอชั่นเมื่อเปรียบเทียบกับรูปแบบอ้างอิงใดมีค่าน้อยที่สุดก็จะได้ผลว่าคำสั่งที่นำมาทดสอบตรงกับรูปแบบอ้างอิงนั้น

บทที่ 5
การทดลองและผลการทดลอง

5.1 การทดลอง

ระบบจะทำการบันทึกเสียงคำสั่งทุกเสียง ด้วยความละเอียดขนาด 16 บิท และความถี่ในการสุ่ม 11025 เฮิร์ต เนื่องจากเป็นช่วงความถี่ที่เหมาะสมในการรู้จำเสียงคำสั่งของโครงงานนี้ โดยทางคณะผู้จัดทำแบ่งการทดลองเป็น 2 การทดลอง คือ

1. การทดลองเพื่อหาขนาดบล็อกที่เหมาะสมในการรู้จำเสียง ใช้ผู้ทดลอง 4 คน เป็นผู้หญิง 2 คน และผู้ชาย 2 คน โดยทดลองในห้องเงียบ โค้ดบุคได้จากการสั่งงานครั้งแรก
2. การทดลองเพื่อวิเคราะห์สภาวะแวดล้อมที่เหมาะสมกับการรู้จำเสียง ใช้ผู้ทดลองคนเดียว เป็นผู้ชาย โดยทดลองในสภาวะแวดล้อมที่ต่างกัน 4 กรณี (โค้ดบุคได้จากการสั่งงานในห้องเงียบ) คือ

กรณีที่ 1 ทดลองในห้องเงียบ

กรณีที่ 2 ทดลองเสียงก้อง (Echo)

เพิ่มเสียงก้องจากการทดลองในห้องเงียบ โดยใช้โปรแกรม Sound Forge

กรณีที่ 3 ทดลองเสียงที่มีเสียงรบกวนจากสัญญาณไฟฟ้า

เพิ่มเสียงรบกวนจากการทดลองในห้องเงียบ โดยใช้โปรแกรม Sound Forge

กรณีที่ 4 ทดลองในห้องที่มีเสียงรบกวนจากห้องทำงาน

5.2 การสร้างรูปแบบอ้างอิง

ผู้ทดลองแต่ละคนทำตามขั้นตอนดังนี้

1. บันทึกเสียงคำสั่งทุกเสียง (บันทึกครั้งแรก) ที่ต้องการให้ระบบเรียนรู้
2. นำข้อมูลเสียงของแต่ละคำสั่งมาทำการหาพารามิเตอร์ของเสียงโดยใช้วิธีเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเชี่ยน
3. นำพารามิเตอร์เสียงแต่ละคำสั่งมาหาพารามิเตอร์ที่เป็นตัวแทนของคำสั่งเพื่อสร้างรูปแบบอ้างอิงของคำสั่งนั้น โดยใช้วิธีเวคเตอร์ควอนไตเซชั่น ซึ่งจะได้เวคเตอร์ตัวแทนของคำสั่งที่ระบบสามารถรู้จำได้ของผู้ทดลองแต่ละคน เรียกว่า “โค้ดบุค”

5.3 การทดสอบ

ผู้ทดลองแต่ละคนทำตามขั้นตอนดังนี้

1. นำคำสั่งเสียงที่ต้องการทดสอบเพียง 1 คำสั่ง มาผ่านขั้นตอนเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเชี่ยน จะได้พารามิเตอร์ของคำสั่งนั้น
2. นำพารามิเตอร์มาเปรียบเทียบกับโค้ดบุคของแต่ละคำสั่ง (ของผู้ทดลองคนนั้น) โดยใช้วิธียูคลิเดียนดิสแท้นซ์ (Euclidean distance) จะได้ค่าดิสทอชั่นของแต่ละรูปแบบอ้างอิงออกมา

ค่าดิสทอชั่นเมื่อเปรียบเทียบกับรูปแบบอ้างอิงใดมีค่าน้อยที่สุด ก็จะได้ผลว่าคำสั่งที่นำมาทดสอบตรงกับรูปแบบอ้างอิงนั้น

3. แสดงผลของคำสั่งที่นำมาทดสอบว่าเป็นคำสั่งใด

4. นำคำสั่งนั้นไปใช้ในการสั่งงานของเมาส์และคีย์บอร์ด

โดยคำสั่ง 10 คำสั่งที่ใช้ทดสอบมีดังนี้

คำสั่งที่	คำพูด
1	ฟังเพลง
2	เปิดไมโครซอฟท์เวิร์ด
3	msn
4	ดูหนัง
5	เข้าเว็บไซต์
6	เล่นเกมส์
7	explorer
8	ดูรูปภาพ
9	ปิดโปรแกรม
10	พิมพ์งาน

ตารางที่ 5.1 แสดงคำสั่ง 10 คำสั่งที่ใช้ทดสอบการรู้จำเสียงคำสั่ง

10 คำสั่งข้างต้นเป็นเพียงการยกตัวอย่างเท่านั้น ในการใช้งานจริงผู้ใช้สามารถพูดคำสั่งใดก็ได้ตามที่ผู้ใช้ต้องการ

5.4 ผลการทดสอบ

5.4.1 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้หญิงคนที่ 1

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	บล็อกขนาด 256		บล็อกขนาด 512		บล็อกขนาด 1024	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	2	5	2	5	2	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7
8	4	8	4	8	4	8
9	9	9	9	9	9	9
10	10	10	10	10	10	10
เปอร์เซ็นต์ความถูกต้อง	80%	100%	80%	100%	80%	100%

ตารางที่ 5.2 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้หญิงคนที่ 1

5.4.2 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้หญิงคนที่ 2

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	บล็อกขนาด 256		บล็อกขนาด 512		บล็อกขนาด 1024	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	1	1	9	1	1	1
2	2	5	2	7	2	2
3	3	9	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7
8	8	8	8	8	8	4
9	9	9	9	9	9	9
10	10	10	10	10	10	10
เปอร์เซ็นต์ความถูกต้อง	100%	80%	90%	90%	100%	90%

ตารางที่ 5.3 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้หญิงคนที่ 2

5.4.3 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 1

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	บล็อกขนาด 256		บล็อกขนาด 512		บล็อกขนาด 1024	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	1	1	1	1	1	1
2	5	9	5	9	5	5
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7
8	8	8	8	8	8	8
9	9	9	9	9	9	9
10	10	10	10	10	10	10
เปอร์เซ็นต์ความถูกต้อง	90%	90%	70%	90%	90%	90%

ตารางที่ 5.4 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 1

5.4.4 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 2

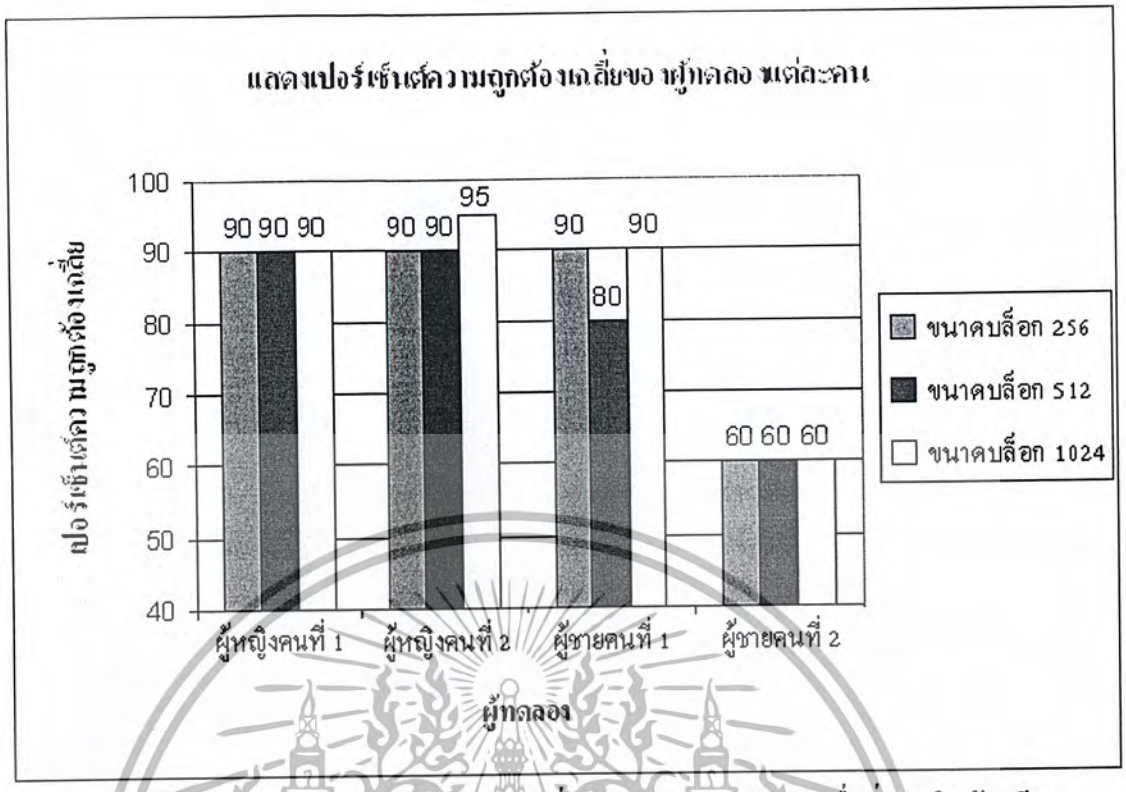
คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	บล็อกขนาด 256		บล็อกขนาด 512		บล็อกขนาด 1024	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	5	1	5	1	5	1
2	8	2	2	5	2	2
3	6	3	6	6	6	6
4	8	2	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	5	5
8	5	5	5	5	5	5
9	2	9	1	9	1	9
10	10	10	1	10	10	10
เปอร์เซ็นต์ความถูกต้อง	40%	80%	50%	70%	50%	70%

ตารางที่ 5.5 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 2

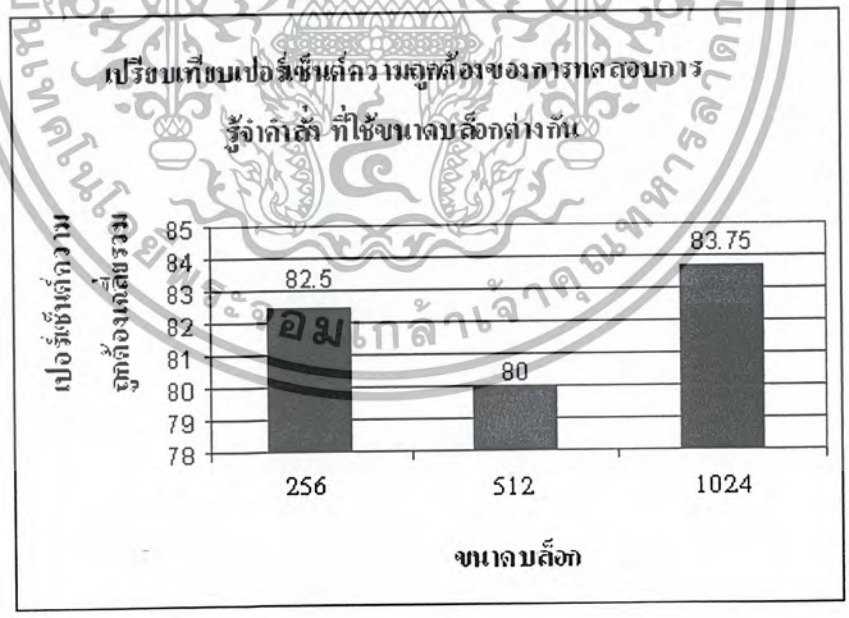
5.4.5 ผลการเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการทดสอบการรู้จำเสียงคำสั่ง เมื่อสั่งงานในห้องเงียบ

ผู้ทดลอง	เปอร์เซ็นต์ความถูกต้องเฉลี่ย		
	บล็อกขนาด 256	บล็อกขนาด 512	บล็อกขนาด 1024
ผู้หญิงคนที่ 1	90%	90%	90%
ผู้หญิงคนที่ 2	90%	90%	95%
ผู้ชายคนที่ 1	90%	80%	90%
ผู้ชายคนที่ 2	60%	60%	60%
เปอร์เซ็นต์ความถูกต้องเฉลี่ยรวม	82.5%	80%	83.75%

ตารางที่ 5.6 แสดงผลการเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ



รูปที่ 5.1 กราฟแสดงเปอร์เซ็นต์ความถูกต้องเฉลี่ยของผู้ทดลองแต่ละคน เมื่อสั่งงานในห้องเงียบ

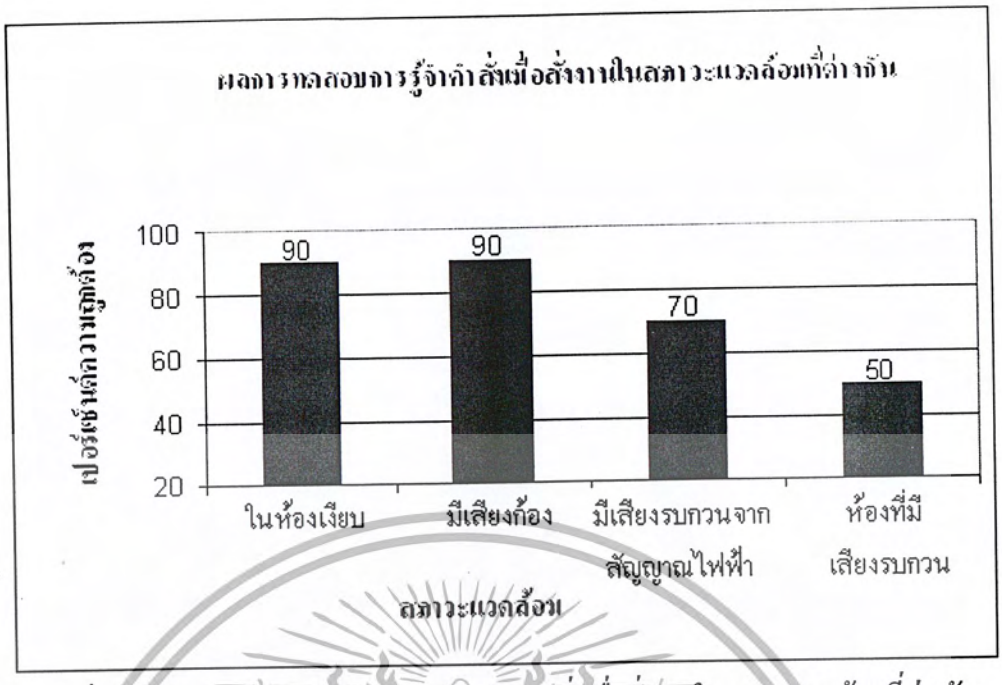


รูปที่ 5.2 กราฟแสดงผลการเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการทดสอบการรู้จำเสียงคำสั่ง ที่ใช้ขนาดบล็อกต่างกัน เมื่อสั่งงานในห้องเงียบ

5.4.6 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในสภาวะแวดล้อมที่ต่างกัน ของผู้ทดลองผู้ชายคนที่ 1 โดยใช้บล็อกขนาด 1024

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง			
	ในห้องเงียบ	มีเสียงก้อง	มีเสียงรบกวนจากสัญญาณไฟฟ้า	ห้องที่มีเสียงรบกวน
1	1	1	3	1
2	5	5	5	2
3	3	3	3	1
4	4	4	4	4
5	5	5	5	2
6	6	6	3	2
7	7	7	7	2
8	8	8	8	8
9	9	9	9	2
10	10	10	10	10
เปอร์เซ็นต์ความถูกต้อง	90%	90%	70%	50%

ตารางที่ 5.7 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในสภาวะแวดล้อมที่ต่างกัน ของผู้ทดลองผู้ชายคนที่ 1 โดยใช้บล็อกขนาด 1024



รูปที่ 5.3 กราฟแสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในสภาวะแวดล้อมที่ต่างกัน โดยใช้บล็อกขนาด 1024

บทที่ 6
สรุปและวิจารณ์ผลการทดลอง

6.1 สรุปผลการทดลอง

จากการทดลองระบบการรู้จำเสียงคำสั่งจำนวน 10 คำสั่ง โดยใช้สัญญาณเสียงด้วยความละเอียดขนาด 16 บิท และความถี่ในการสุ่ม 11025 เฮิร์ต ผลการทดลองสรุปได้ดังนี้

1. ขนาดของบล็อกมีผลต่อการหาค่าความถูกต้อง ดังนั้นในระบบของเราจึงเลือกบล็อกขนาด 1024 ซึ่งมีเปอร์เซ็นต์ความถูกต้องสูงที่สุดคือ 83.75%
2. เสียงพูดของคน ๆ เดียวกันในแต่ละคำสั่ง ในการพูดแต่ละครั้งพบว่าเสียงที่พูดออกมาไม่เหมือนกัน และลักษณะของคำสั่งที่มีการออกเสียงคล้ายกัน มีผลทำให้เปอร์เซ็นต์ความถูกต้องลดลง
3. การบันทึกเสียงเพื่อนำมาสร้างรูปแบบอ้างอิง และเสียงที่ต้องการทดสอบ ควรจะบันทึกในสภาพห้องที่เงียบ มีเสียงรบกวนน้อย และมีอุปกรณ์บันทึกเสียงที่ดี เพื่อให้ได้สัญญาณเสียงที่มีคุณภาพ
4. ในการรู้จำเสียงโดยทำการหาค่าพารามิเตอร์ที่เป็นตัวแทนเสียงด้วยวิธีเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเซี่ยน และนำพารามิเตอร์เสียงทั้งหมดมาสร้างรูปแบบอ้างอิงโดยใช้วิธีเวคเตอร์ควอนไตเซชั่น (VQ) สามารถรู้จำเสียงได้ดีระดับหนึ่ง โดยมีเปอร์เซ็นต์ความถูกต้องเท่ากับ 83.75%

6.2 บทวิจารณ์และแนวทางพัฒนา

การที่ผลการทดลองออกมายังไม่สามารถรู้จำเสียงได้ดีเท่าที่ควร เนื่องจากเกิดสัญญาณรบกวนในขณะทำการอัดเสียง และลักษณะของคำสั่งที่มีการออกเสียงคล้ายกัน ทำให้ผลการทดสอบเกิดความคลาดเคลื่อน

การพัฒนาและปรับปรุงในขั้นต่อไป ควรปรับปรุงวิธีหาค่าพารามิเตอร์ที่เป็นตัวแทนเสียงให้เหมาะสม หรืออาจจะปรับเปลี่ยนวิธีสร้างรูปแบบอ้างอิง ซึ่งคาดว่าจะทำให้ประสิทธิภาพในการรู้จำเสียงดีขึ้น

ภาคผนวก ก
การติดตั้งโปรแกรม

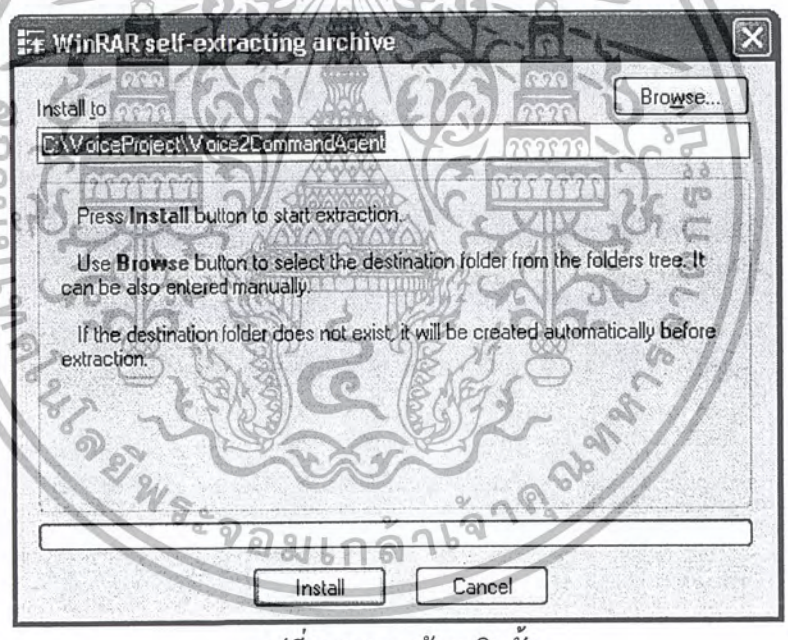
ขั้นตอนการติดตั้งโปรแกรม Voice-to-Command Agent

1. เรียกไฟล์ Voice2CommandAgent.exe จากแผ่นติดตั้ง

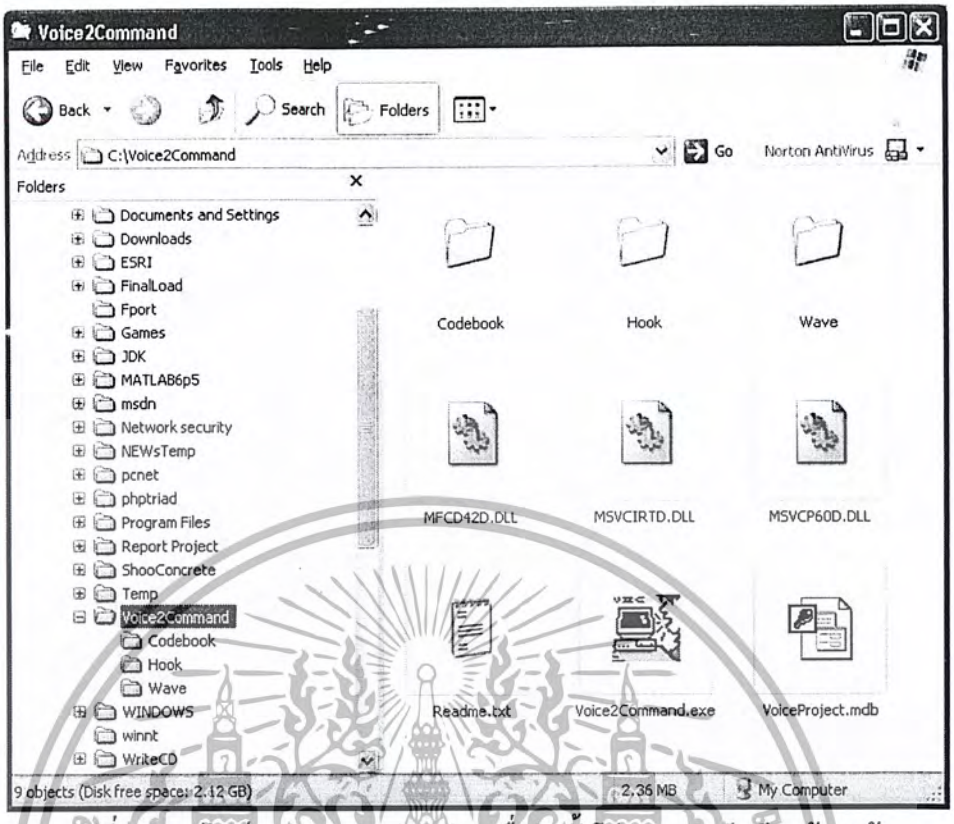
Voice2CommandAgent...

รูปที่ 1 แสดงไฟล์สำหรับติดตั้ง

2. เลือก Folder ที่จะทำการติดตั้ง แล้วกดปุ่ม Install เพื่อทำการติดตั้งโปรแกรม

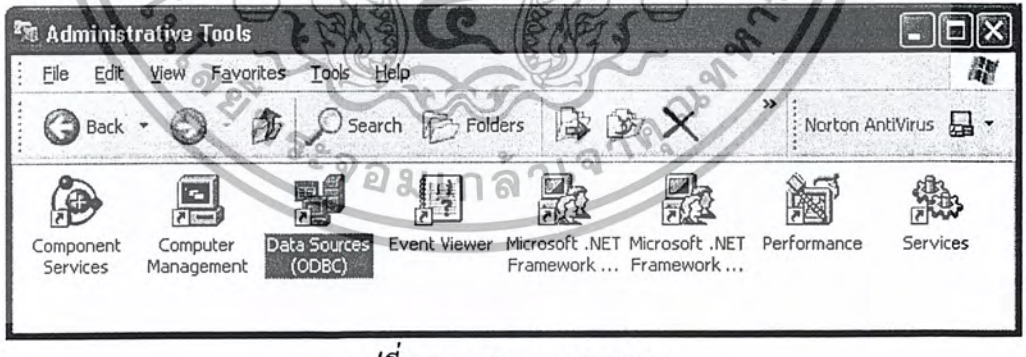


รูปที่ 2 แสดงหน้าจอติดตั้ง

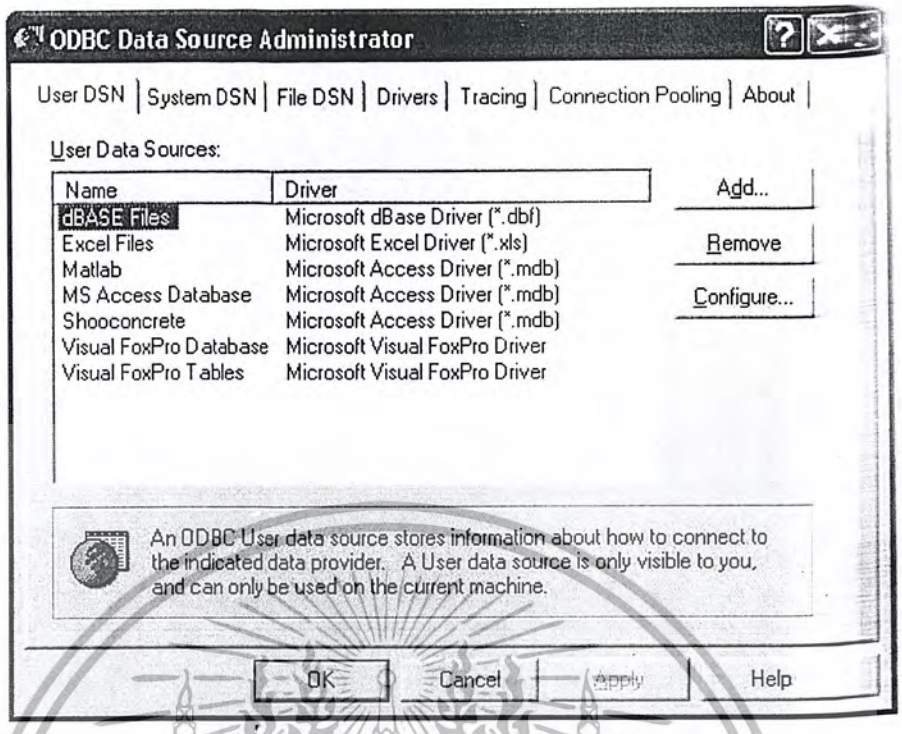


รูปที่ 3 แสดงไฟล์ Voice2Command.exe เมื่อติดตั้งโปรแกรมเสร็จเรียบร้อยแล้ว

3. ทำการติดตั้งระบบฐานข้อมูลของโปรแกรม กด Control Panel เลือก Administrative Tools เลือก Data Sources (ODBC)

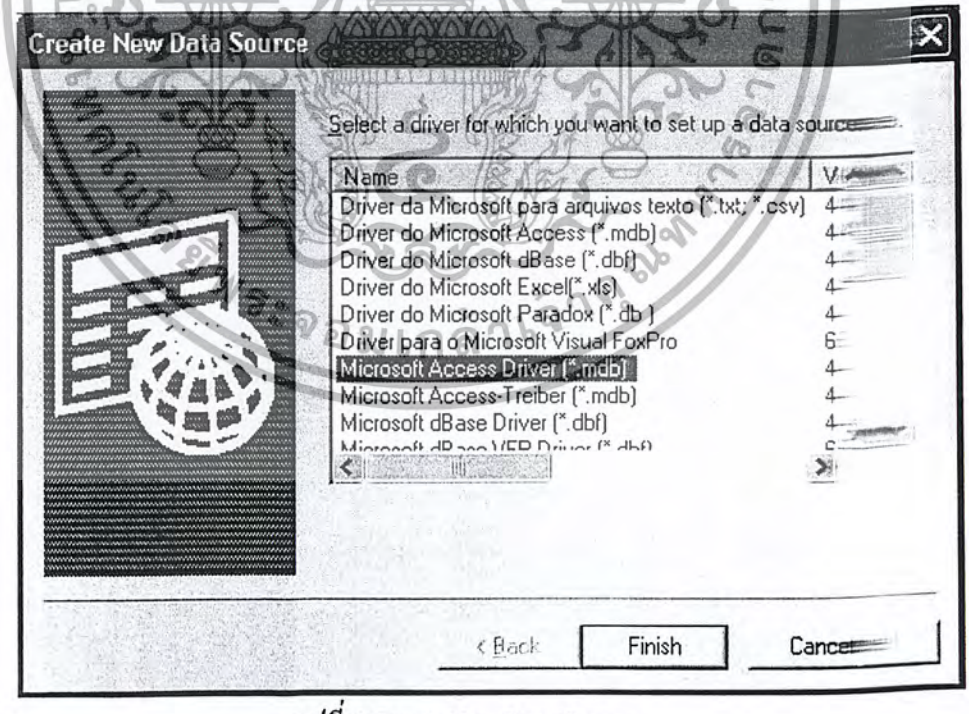


รูปที่ 4 Data Sources (ODBC)



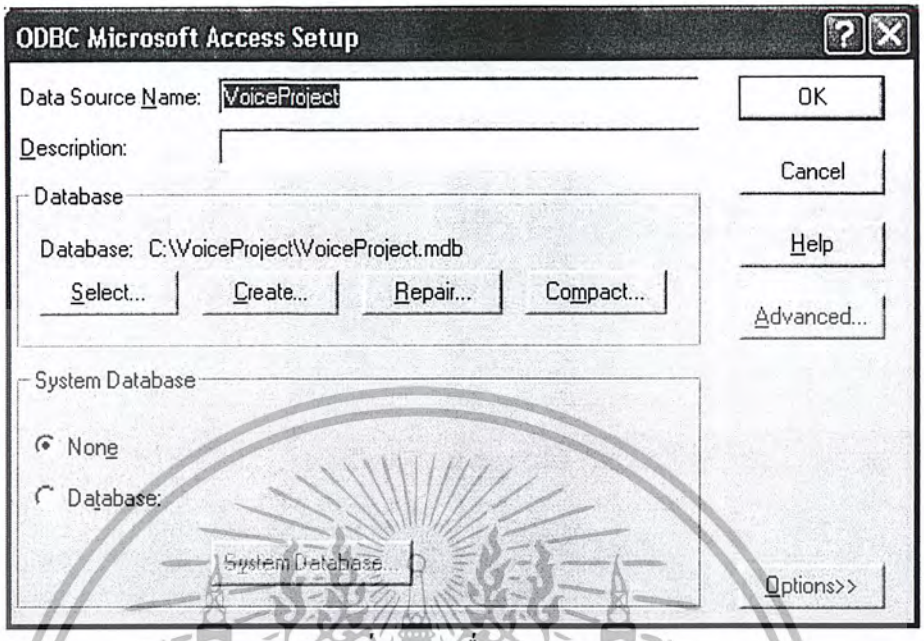
รูปที่ 5 ODBC Data Source Administrator

4. กด Add และเลือก Microsoft Access Driver (*.mdb)



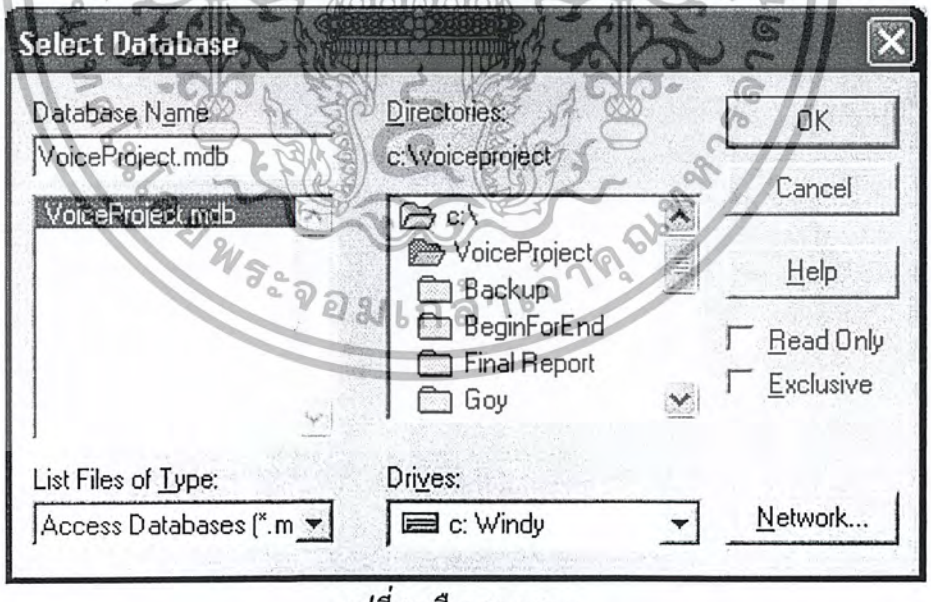
รูปที่ 6 Create New Data Source

5. กรอกชื่อ Data Source Name : VoiceProject

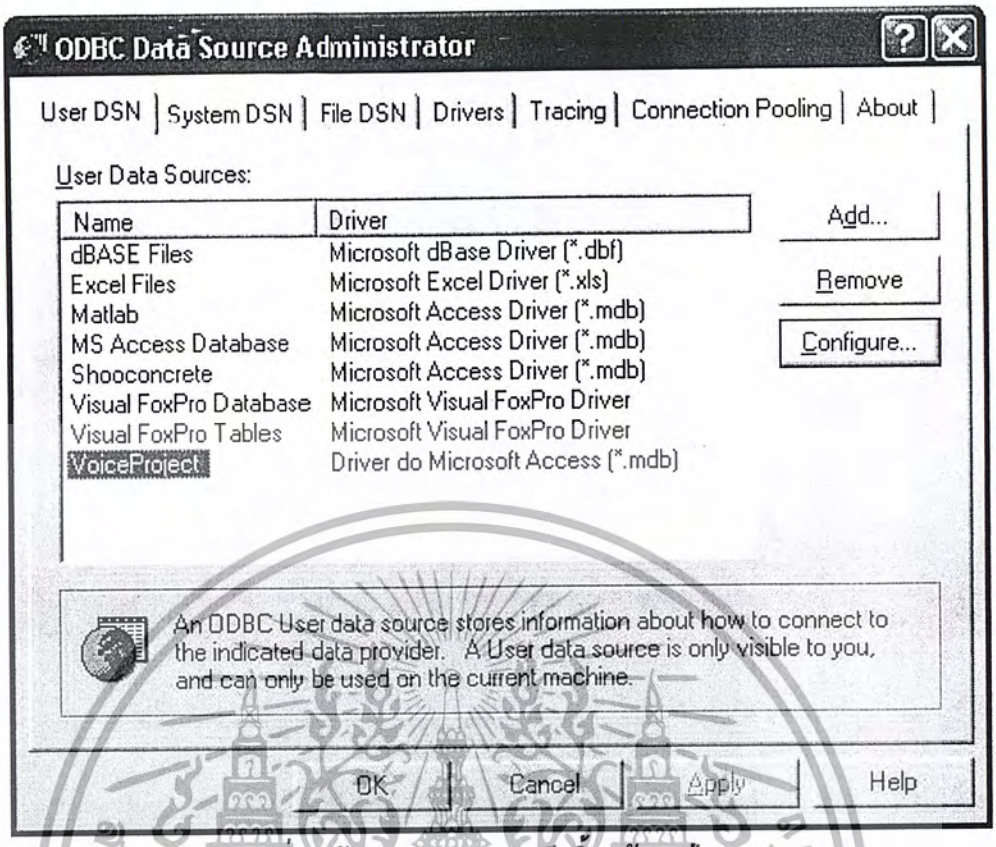


รูปที่ 7 กรอกชื่อ Database

6. เลือกไฟล์ VoiceProject.mdb จาก Folder ที่ทำการติดตั้งโปรแกรมลงไป



รูปที่ 8 เลือก Database



รูปที่ 9 สร้าง Data Source เสร็จสิ้นแล้วกดปุ่ม OK

ภาคผนวก ข

การใช้งานโปรแกรม

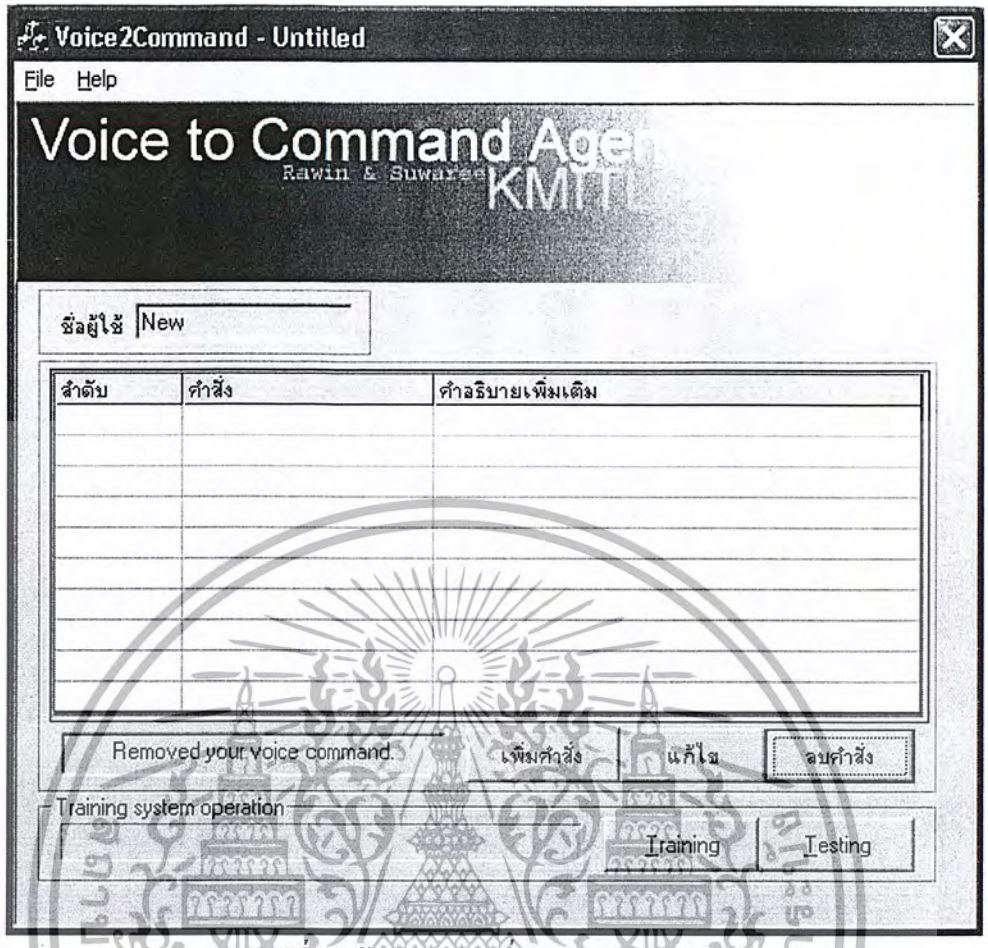
วิธีการใช้งานโปรแกรม Voice-to-Command Agent

1. เรียกโปรแกรมโดยกดที่ไฟล์ Voice2Command.exe (อยู่ใน Folder ที่ทำการติดตั้ง)

รูปที่ 1 รูปไฟล์ Voice2Command.exe

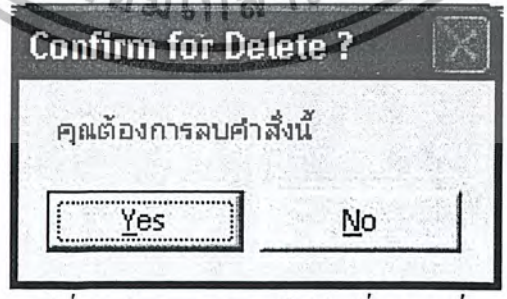
2. จะปรากฏหน้าจอสำหรับทำการ log in ให้ทำการกรอกชื่อ username และ password (ในกรณีที่ยังไม่มีให้เลือก check box : New User) แล้วกดปุ่ม OK (หมายเหตุ : radio button - test mode จะใช้ในกรณีที่ผู้ใช้ผ่านทำการ train ระบบเรียบร้อยแล้ว) ระบบจะเข้าสู่หน้าจอแสดงคำสั่งของระบบ training

รูปที่ 2 หน้าจอ log in



รูปที่ 3 หน้าจอแสดงคำสั่งของระบบ training

3.ในที่นี้ระบบจะยังไม่มีคำสั่งใดๆ ให้ User ทำการเพิ่มคำสั่งให้ระบบ โดยการกดปุ่ม "เพิ่มคำสั่ง" เพื่อเข้าสู่หน้าจอของระบบ training หากต้องการแก้ไขคำสั่ง กดปุ่ม "แก้ไข" และหากต้องการลบ คำสั่ง กดปุ่ม "ลบคำสั่ง"



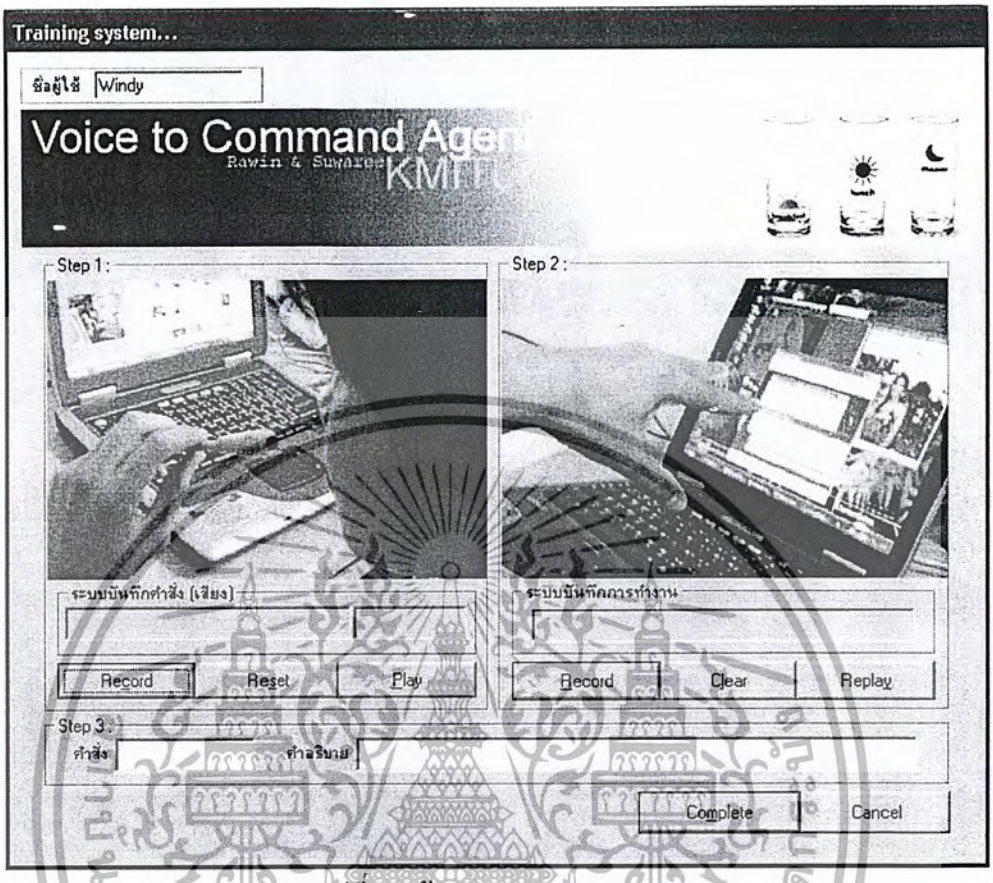
รูปที่ 4 หน้าจอแสดงการยืนยันเพื่อลบคำสั่ง

4. ในหน้าจอ training จะมีอยู่ 3 ขั้นตอน คือ

4.1) ระบบบันทึกคำสั่ง (เสียง)

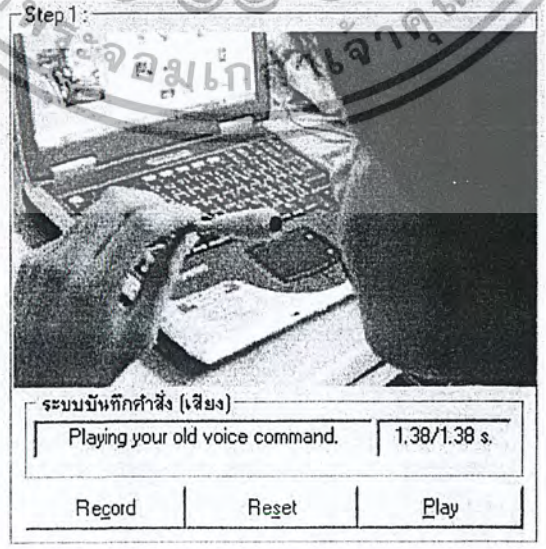
4.2) ระบบบันทึกการทำงาน

4.3) ส่วนอธิบายคำสั่ง (ประกอบด้วย คำสั่ง และ คำอธิบาย)



รูปที่ 5 หน้าจอของระบบ training

4.1) ระบบบันทึกคำสั่ง (เสียง)



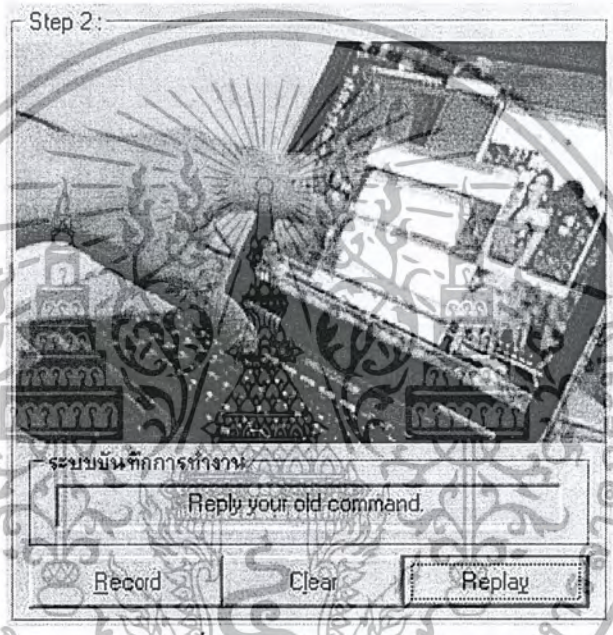
รูปที่ 6 ระบบบันทึกคำสั่ง (เสียง)

ปุ่ม **Record** – กดเริ่มบันทึกเสียง / หยุดการบันทึกเสียง

ปุ่ม **Reset** – ยกเลิกการบันทึกเสียง (ในกรณีที่เคยมีการบันทึกเสียงจนครบทั้ง 3 ขั้นตอน และกดปุ่ม Complete แล้ว จะเป็นการเรียกไฟล์เสียงเดิมกลับมาเป็นเสียงคำสั่งปัจจุบัน)

ปุ่ม **Play** – ทดลองฟังเสียงที่บันทึกล่าสุด (ในกรณีที่ยังไม่มีการบันทึกไฟล์เสียงใหม่ แต่เคยมีการบันทึกเสียงแล้วกดปุ่ม Complete จะเรียกเสียงเดิมมาเล่น)

4.2) ระบบบันทึกการทำงาน



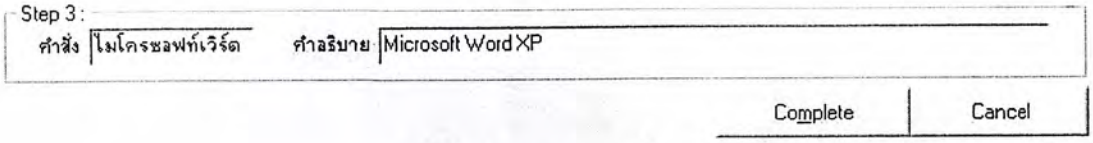
รูปที่ 7 ระบบบันทึกการทำงาน

ปุ่ม **Record** – กดปุ่มนี้เพื่อเริ่มบันทึกการทำงานของคีย์บอร์ดและเมาส์ของผู้ใช้ เมื่อสิ้นสุดการทำงานแล้วกดปุ่ม F2

ปุ่ม **Clear** – ยกเลิกการบันทึกการทำงานครั้งล่าสุด

ปุ่ม **Replay** – แสดงการทำงานที่ทำการบันทึกไว้ล่าสุด (ในกรณีที่ไม่มีการบันทึกครั้งล่าสุด แต่เคยมีการบันทึกการทำงานไว้และกดปุ่ม Complete แล้วจะแสดงการบันทึกครั้งล่าสุดขึ้นมา)

4.2) ส่วนอธิบายคำสั่ง



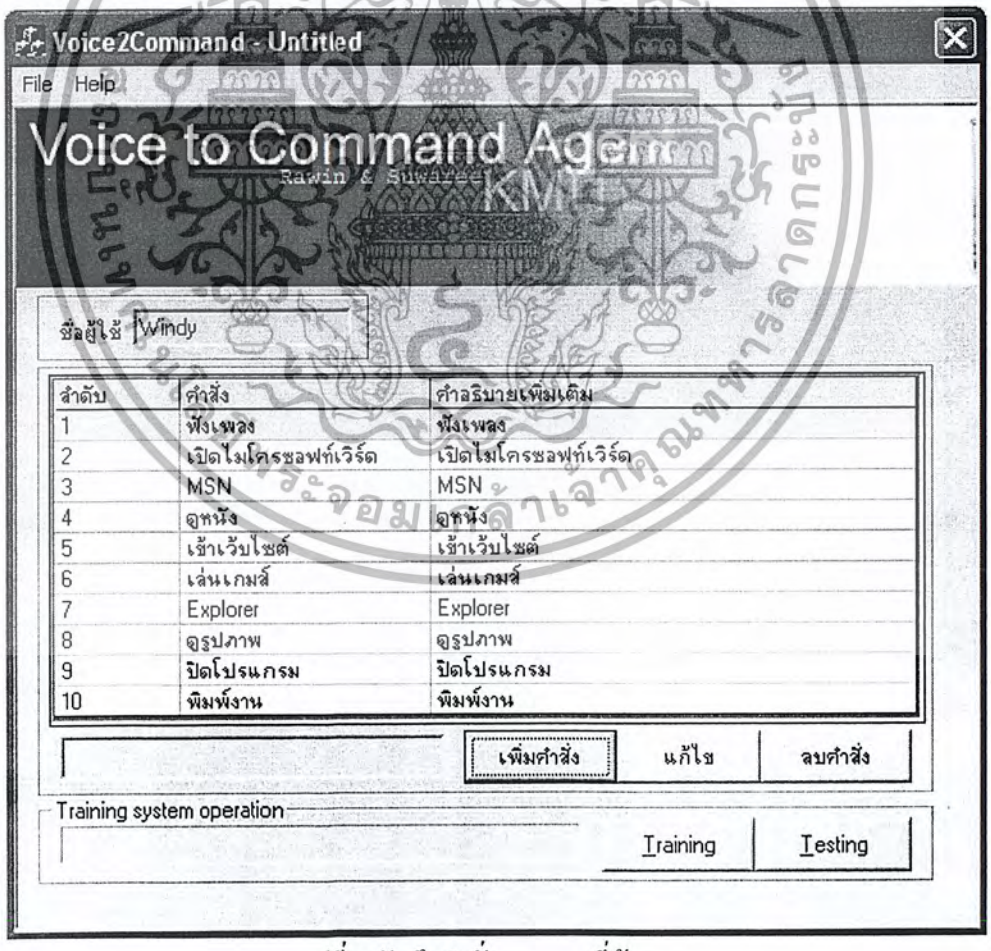
รูปที่ 8 ส่วนอธิบายคำสั่ง

คำสั่ง – ใส่ชื่อคำสั่ง

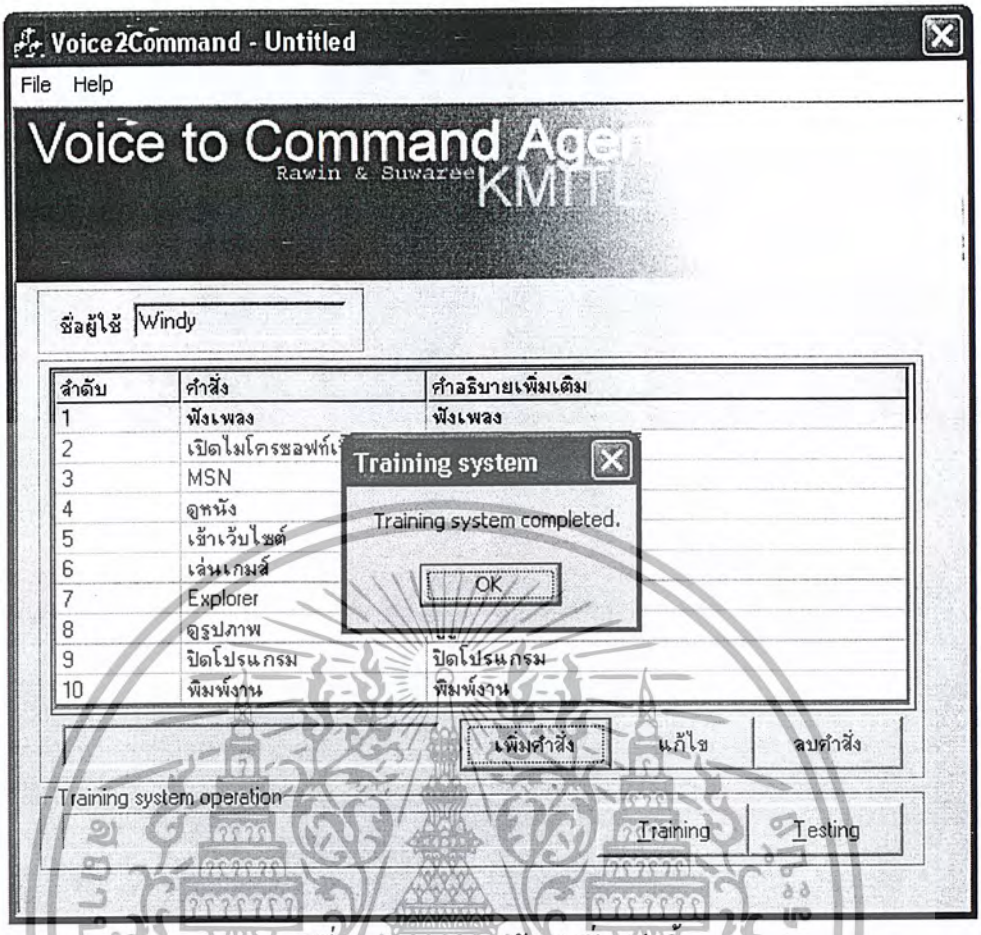
คำอธิบาย – ใส่คำอธิบายเพิ่มเติมของคำสั่ง

เมื่อทำการบันทึกข้อมูลครบทั้ง 3 ขั้นตอนแล้วกดปุ่ม Complete ระบบจะบันทึกข้อมูลลงฐานข้อมูล

เมื่อทำการบันทึกคำสั่งจนเสร็จตามจำนวนที่ต้องการ (ในที่นี้เราออกแบบไว้ไม่เกิน 10 คำสั่ง เพราะต้องการให้การประมวลผลใช้เวลาไม่นานมากนัก) ให้กดปุ่ม Training เพื่อสั่งให้ระบบรู้จำคำสั่ง

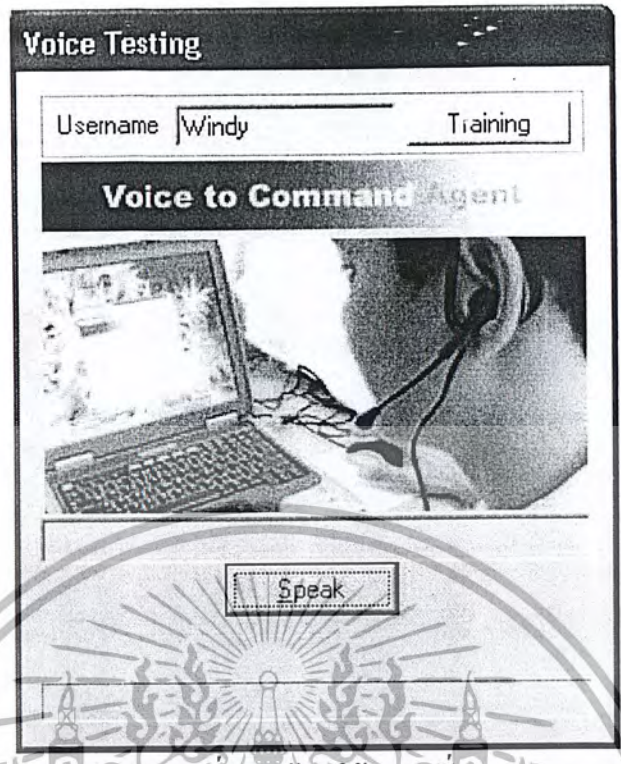


รูปที่ 9 บันทึกคำสั่งครบตามที่ต้องการ

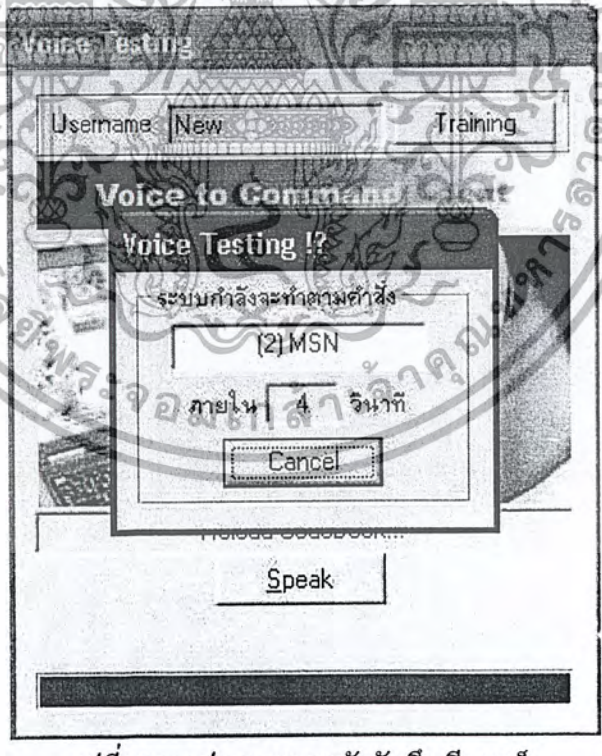


รูปที่ 10 ระบบทำการรู้จำคำสั่งเสร็จสิ้น

เมื่อผู้ใช้ต้องการใช้งานระบบสั่งงานด้วยเสียง ทำได้โดยกดปุ่ม Testing จะปรากฏหน้าจอใช้งานคำสั่งขึ้นมา โดยจะมีปุ่มใช้งาน 2 ปุ่ม คือ



รูปที่ 11 หน้าจอใช้งานคำสั่ง

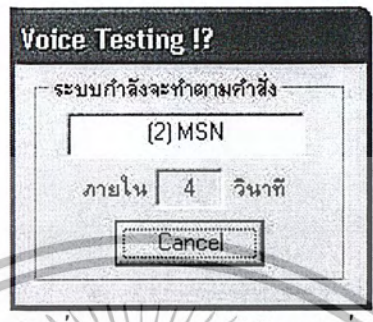


รูปที่ 12 การประมวลผลหลังบันทึกเสียงเสร็จ

ปุ่ม **Speak** – สำหรับทำการเริ่มบันทึกเสียงคำสั่ง / สิ้นสุดการบันทึกคำสั่ง (เพื่อทำการประมวลผล)

ปุ่ม **Training** – กลับสู่หน้าจอแสดงคำสั่ง

เมื่อทำการบันทึกเสียงเสร็จสิ้นแล้ว ระบบจะนำคำสั่งที่บันทึกไปเปรียบเทียบกับผลการรู้จำเสียงจากการ Training แล้วจะเลือกคำสั่งที่ใกล้เคียงที่สุดออกมา โดยถ้าไม่กดปุ่ม Cancel ภายใน 4 วินาที ระบบจะทำตามคำสั่งที่เลือกขึ้นมา ถ้าเลือก Cancel ระบบจะทำการยกเลิกคำสั่งนั้น และกลับไปสู่หน้าจอใช้งานคำสั่งเพื่อรอรับคำสั่งต่อไป



รูปที่ 13 แสดงผลการใช้งานคำสั่ง

ภาคผนวก ค

ประเภทของวินโดว์แมสเซจฮุค

Hook	Scope	Description
WH_CALLWNDPROC	Thread or system	Monitors Windows messages before they are sent to a particular window's window procedure.
WH_CALLWNDPROCRET	Thread or system	Monitors Windows messages after they have been processed by a particular window's window procedure.
WH_CBT	Thread or system	Used for computer-based training applications.
WH_DEBUG	Thread or system	Used when debugging other hook functions.
WH_FOREGROUNDIDLE	Thread or system	The hook procedure will be called when an application's foreground window goes idle.
WH_GETMESSAGE	Thread or system	Monitors messages posted to a message queue.
WH_JOURNALPLAYBACK	System only	Plays back a series of keystrokes and mouse movements previously recorded with a journal record hook (a macro). Normal mouse and keyboard input is disabled while the macro is playing.
WH_KEYBOARD	Thread or system	The hook procedure will be called when a WM_KEYDOWN or WM_KEYUP message is about to be processed.
WH_JOURNALRECORD	System only	The hook procedure will be called for all keyboard and mouse messages. Useful for recording macros.
WH_KEYBOARD_LL	Thread or system	A low-level keyboard hook (Windows NT only).
WH_MOUSE	Thread or system	The hook procedure will be called when a mouse message is about to be processed.
WH_MOUSE_LL	Thread or system	A low-level mouse hook (Windows NT only).
WH_MSGFILTER	Thread or system	Monitors messages as a result of an action in a menu, dialog box, or scroll bar created by a specific application.
WH_SHELL	Thread or system	Monitors messages affecting the Windows shell.
WH_SYSMSGFILTER	System only	Monitors messages as a result of an action in a menu, dialog box, or scroll bar created by any application.

ตารางที่ 1 แสดงประเภทของวินโดว์ฮุค

Hook	Scope	Description
WH_CALLWNDPROC	Thread or system	Monitors Windows messages particular window's window
WH_CALLWNDPROCRET	Thread or system	Monitors Windows messages by a particular window's
WH_CBT	Thread or system	Used for computer-based train
WH_DEBUG	Thread or system	Used when debugging other h
WH_FOREGROUNDIDLE	Thread or system	The hook procedure will be c foreground window goes idle.
WH_GETMESSAGE	Thread or system	Monitors messages posted to
WH_JOURNALPLAYBACK	System only	Plays back a series of keystro previously recorded with a jou Normal mouse and keyboard macro is playing.
WH_KEYBOARD	Thread or system	The hook procedure will be c WM_KEYDOWN or WM_K processed.
WH_JOURNALRECORD	System only	The hook procedure will be c mouse messages. Useful for
WH_KEYBOARD_LL	Thread or system	A low-level keyboard hook
WH_MOUSE	Thread or system	The hook procedure will be c about to be processed.
WH_MOUSE_LL	Thread or system	A low-level mouse hook (W
WH_MSGFILTER	Thread or system	Monitors messages as a resu box, or scroll bar created by
WH_SHELL	Thread or system	Monitors messages affecting
WH_SYSMSGFILTER	System only	Monitors messages as a resu box, or scroll bar created by

ตารางที่ 1 แสดงประเภทของวินโดว์ฮุค

ภาคผนวก ง

ซอร์สโค้ดการทำงานของวินโดว์แมสเซจฮุค

```
#include "windows.h"
#include "iostream.h"
#include "fstream.h"

#include "string"
using std::string;

//Global name
string HooksForPlay;
HHOOK hhookHooks;
DWORD dwStartRecordTime;
int m_Max = 0;
int m_Now = 0;
extern "C"{
LRESULT CALLBACK JournalRecordFunc (int nCode,WPARAM wParam,LPARAM lParam)
{
        LPEVENTMSG  lpEvent;
        if (nCode >=0)    {
        lpEvent = (LPEVENTMSG) lParam;
        if (lpEvent->message == WM_KEYDOWN && LOBYTE(lpEvent->paramL)==VK_F2)
        {       UnhookWindowsHookEx(hhookHooks);                //Remove Hook;
                return 0;
        }
        if (m_Max == 0)  {dwStartRecordTime = (DWORD) GetTickCount();   } else {}
        ofstream file;
        HooksForPlay = "Hook\\TestHooks.bin";  // For use char strings with "HooksForPlay.c_str()"
        file.open (HooksForPlay.c_str(),ios::out|ios::app|ios::binary);
        file<<lpEvent->message<<endl;
        file<<lpEvent->paramL<<endl;
        file<<lpEvent->paramH<<endl;
        file<<lpEvent->time<<endl;
        file.close();
        m_Max++;
        return 0;
```

```
        }
        return (CallNextHookEx(hhookHooks,nCode,wParam,lParam));
}

LRESULT CALLBACK JournalPlaybackFunc (int nCode,WPARAM wParam,LPARAM lParam)
{
        static int          nRepeatRequests;
        static DWORD        dwTimeAdjust;
        static DWORD        dwLastEventTime;
        LPEVENTMSG          lpEvent;
        long                lReturnValue;
        if (nCode >= 0)
        {
            ifstream file;
            file.open (HooksForPlay.c_str(),ios::in);
            char buffer[64];
            m_Max = 0;
            file.getline(buffer,64);
            int Res = atoi(buffer);
            while (!file.eof() && (Res > 0))
            {
                file.getline(buffer,64);
                file.getline(buffer,64);
                file.getline(buffer,64);
                if (m_Max == 0) // Add by myself becoz it must initial
                {
                        file.getline(buffer,8,'\n');
                        long time           = atol(buffer);
                        dwStartRecordTime = time;
                }
                file.getline(buffer,64);
                Res = atoi(buffer);
                m_Max++;
            }
```

```
        file.close();
if (m_Max == 0)  {
        UnhookWindowsHookEx(hhookHooks);                //RemoveHook
        return 0; //((int)
        CallNextHookEx(WH_JOURNALPLAYBACK,nCode,wParam,lParam));
}
if (m_Now == 0)  {
        dwTimeAdjust = GetTickCount() - dwStartRecordTime;
        dwLastEventTime = (DWORD) GetTickCount();
        nRepeatRequests = 1;
        m_Now++;
}
if (nCode == HC_SKIP){
        nRepeatRequests = 1;
        if (m_Now >= m_Max)
        {
                m_Max = 0;
                m_Now = 0;
                UnhookWindowsHookEx(hhookHooks);          //Remove JOURNAL
        }
        else    // Case running playback...
        {
                ifstream file;
                file.open (HooksForPlay.c_str(), ios::in);
                char buffer[64];
                int  R = 0;
                while ((R < m_Now) && (!file.eof()))
                {
                        file.getline(buffer,64);
                        file.getline(buffer,64);
                        file.getline(buffer,64);
                        file.getline(buffer,64);
                }
                file.getline(buffer,64);
```

```
                file.getline(buffer,64);
                file.getline(buffer,64);
                file.getline(buffer,64);
                long time = atol(buffer);
                ++m_Now;
                dwLastEventTime = time;
                file.close();
        }
}
else if (nCode == HC_GETNEXT)
{
        ifstream file;
        file.open (HooksForPlay.c_str(), ios::in);
        char buffer[64];
        int  R = 0;
        while ((R < m_Now) && (!file.eof()))
        {
                file.getline(buffer,64);
                file.getline(buffer,64);
                file.getline(buffer,64);
                file.getline(buffer,64);
                R++;
        }
        lpEvent = (LPEVENTMSG) lParam;
        file.getline(buffer,8,'\n');    int message = atoi(buffer);
        file.getline(buffer,8,'\n');    int paramL   = atoi(buffer);
        file.getline(buffer,8,'\n');    int paramH   = atoi(buffer);
        file.getline(buffer,8,'\n');    long time    = atol(buffer);
        lpEvent->paramH  = paramH;
        lpEvent->paramL  = paramL;
        lpEvent->message = message;
        lpEvent->time    = (DWORD) GetTickCount(); //0; //time + dwTimeAdjust;
        file.close();
```

```
                lReturnValue = lpEvent->time - GetTickCount();
                if (lReturnValue < 0L )
                {
                        lReturnValue = 0L;
                        lpEvent->time = GetTickCount();
                }
                return ((DWORD) lReturnValue);
        }
        } // if nCode >= 0
        return (CallNextHookEx(hhookHooks,nCode,wParam,lParam));
}
} // end of C extern
```

ภาคผนวก จ

ซอร์สโค้ดการแปลงฟาสท์ฟูริเยร์

```
// ================================== FFT.H ==================================//

#define FFT             -1
#define REV_FFT          1

// Radix 2 FFT calculation
// param = FFT     - performs FFT
// param = REV_FFT  - performs reverse FFT
// order = the order of FFT ( pow(2,order) calculated complex points )

void fft(double *real,double *imaginary,int order,int param);
```

```
// ================================== FFT.CPP ==================================//

/*
Radix 2 FFT calculation
param = FFT     - performs FFT
param = REV_FFT  - performs reverse FFT
order = the order of FFT ( pow(2,order) calculated complex points )
x = pointer to real
y = pointer to immaginary
*/

#include <math.h>
#include "FFT.H"

#if !defined M_PI
#define M_PI      3.14159265359
#endif

void fft( double *x,double *y,int order,int param ){
        unsigned int n,l,e,f,i,j,o,o1,j1,i1,k;
        double u,v,z,c,s,p,q,r,t,w,a;
        n=1u<<order;
         for( l = 1; l <= order; l++){
                u = 1.0;
                v = 0.0;
                e = 1u << ( order-l+1 );
                f = e/2;
                z = M_PI/f;
                c = cos(z);
                s = sin(z);
                if ( param == FFT )  s = -s;
                for ( j = 1; j <= f; j++ ){
                        for ( i = j; i <=n; i += e){
                                o = i+f-1;
```

```
                o1 = i-1;
                p = x[o1]+x[o];
                r = x[o1]-x[o];
                q = y[o1]+y[o];
                t = y[o1]-y[o];
                x[o] = r*u-t*v;
                y[o] = t*u+r*v;
                x[o1] = p;
                y[o1] = q;
            }
            w = u*c-v*s;
            v = v*c+u*s;
            u = w;
        }
    }
    j = 1;
    for ( i = 1; i < n; i++ ){
        if ( i < j ){
            j1 = j-1;
            i1 = i-1;
            p = x[j1];
            q = y[j1];
            x[j1] = x[i1];
            y[j1] = y[i1];
            x[i1] = p;
            y[i1] = q;
        }
        k = n/2;
        while ( k < j ){
            j = j-k;
            k = k/2;
        }
        j += k;
    }
```

```
        if ( param == FFT ) return;

        a = 1.0/n;
        for ( k = 0; k < n; k++ ){
                x[k] *= a;
                y[k] *= a;
        }
        return;
}
```

บรรณานุกรม

[1] ผศ.น.ท.ดร.สุธรรม ศรีเกษม และน.ต.เมธินทร์ ทรงชัยกุล , MATLAB เพื่อการแก้ปัญหาทางวิศวกรรม , สำนักพิมพ์มหาวิทยาลัยรังสิต

[2] รศ.ดร.มนัส สังวรศิลป์ และวรรัตน์ ภัทรอมรกุล , คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์ , อินโฟเพรส , 2543

[3] Richard M. Jones , Introduction to MFC Programming with Visual C++ , Prentice-Hall,Inc

[4] PAUL M, EMBREE , BRUCE KIMBLE , C Language Algorithms for Digital Signal Processing , Prentice Hall , Englewood Cliffs, New Jersey 0763

[5] MSDN Library - January 2000

[6] http://www.seas.upenn.edu/~archanaa/DSP_Project.html