การควบคุมอุปกรณ์ด้วยการวิเคราะห์เสียง

SPEECH ANALYZING CONTROL THE DEVICE

โดย

นายพิเชษฐ์ บรรยงพิศุทธิ์

นายสุริยนต์ สาระมูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การควบคุมอุปกรณ์ด้วยการวิเคราะห์เสียง

SPEECH ANALYZING CONTROL THE DEVICE

โดย

นายพิเชษฐ์ บรรยงพิศุทธิ์ 41013060

นายสุริยนต์ สาระมูล 41013079

อาจารย์ที่ปรึกษา

อาจารย์กฤษณ์ วงรุจิระ

ดร.สุทธิชัย นพนาคีพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การควบคุมอุปกรณ์ด้วยการวิเคราะห์เสียง**

Speech analyzing control device

ผู้จัดทำ

1. นายพิเชษฐ์ บรรยงพิศุทธิ์ 41013060
2. นายสุริยนต์ สาระมูล 41013079

อาจารย์ที่ปรึกษา

(อาจารย์กฤษณ์ วงรุจิระ)

อาจารย์ที่ปรึกษา

(ดร.สุทธิชัย นพนาคีพงษ์)

การควบคุมอุปกรณ์ด้วยการวิเคราะห์เสียง

Speech analyzing control the device

โดย นายพิเชษฐ์ บรรยงพิศุทธิ์ 41013060
นายสุริยนต์ สาระมูล 41013079

อาจารย์ที่ปรึกษา อาจารย์กฤษณ์ วงรุจิระ
ดร.สุทธิชัย นพนาคีพงษ์

บทคัดย่อ

ปริญญานิพนธ์นี้นำกระบวนการจดจำเสียงพูดไปใช้ในการควบคุมอุปกรณ์ โดยในการจดจำเสียงจะมีกระบวนการในการวิเคราะห์สัญญาณเสียง จากนั้นจึงนำไปเปรียบเทียบกับสัญญาณอ้างอิง แล้วนำไปวิเคราะห์ความเป็นไปได้ของสัญญาณเสียงว่าเป็นสัญญาณเสียงตัวใด เมื่อได้สัญญาณเสียงที่ถูกต้องแล้วต่อไปก็เป็นกระบวนการนำสัญญาณเสียงที่ได้ไปประยุกต์ใช้งาน

ABSTRACT

This project presents the process of speech recognition that uses for control the device. The process of speech recognize is analyze speech signal, then compare with the reference signal and lead the signal to analyze with probability's process. The result is used to control the device.

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

ในปัจจุบันการติดต่อสื่อสารด้วยเสียงได้ถูกพัฒนาก้าวหน้าไปมาก เนื่องจากมีอุปกรณ์และวิธีการในการวิเคราะห์ที่มีประสิทธิภาพ ทำให้สามารถลดความยุ่งยากซับซ้อนของการวิเคราะห์ได้ ดังเช่น เครื่องพิมพ์ข้อความด้วยเสียง ซึ่งเป็นการช่วยคนพิการในการทำงานได้เป็นอย่างดี

โดยปกติแล้วเสียงของมนุษย์เรานั้น จะมีความแตกต่างกันไปในเรื่องของโทนเสียง การออกเสียง เสียงสูงเสียงต่ำ ความเร็วในการพูดของแต่ละคน แต่ก็ได้มีการวิจัยเรื่อยมาจนปัจจุบันการรู้จำเสียงพูดสามารถจะใช้งานได้ดี ด้วยการใช้เครื่องคอมพิวเตอร์ส่วนบุคคล การ์ดเสียง และขบวนการในการวิเคราะห์เสียงต่าง ๆ

สำหรับปริญญานิพนธ์นี้จะทำการศึกษาและทดลองนำเอาส่วนของการรู้จำเสียงพูดมาเขียนเป็นโปรแกรมภาษาซี โดยทำการบันทึกเสียงที่จะนำมาเปรียบเทียบผ่านไมโครโฟนซึ่งจะบันทึกรูปแบบของเสียงพูดให้อยู่ในรูปแบบที่เหมาะสม โดยจะจำแนกเสียงพูดเป็น หน้า หลัง ซ้าย ขวา และหยุด มีช่วงความถี่อยู่ระหว่าง 300-3400 เฮิรตซ์ และสุ่มโดยใช้ความถี่ 8000 เฮิรตซ์ ตามกฎของไนควิสต์ ซึ่งเราจะใช้การ์ดเสียงของเครื่องคอมพิวเตอร์ทำหน้าที่นี้แทน และจะนำเสียงพูดที่ได้นี้ไปผ่านกระบวนการแอลพีซี (LPC) เพื่อลดจำนวนข้อมูลในการวิเคราะห์ให้น้อยลง จากนั้นจะนำไปผ่านกระบวนการเวคเตอร์ควอนไตซ์ (Vector Quantization) แล้วจะได้ค่าปรากฏของเสียงพูดนั้น แล้วนำค่าปรากฏนี้เข้าไปผ่านกระบวนการฮิดเดนมาร์คอฟ (Hidden Markov) เพื่อนำไปสร้างโมเดลเสียงที่ต้องการจะจำ จากนั้นเมื่อนำเสียงผ่านกระบวนการต่าง ๆ ดังที่ได้กล่าวมาจนถึงกระบวนการหาค่าปรากฏ เมื่อได้ค่าปรากฏแล้วก็นำไปเปรียบเทียบหาค่าความน่าจะเป็นกับโมเดลเสียงที่ได้สร้างไว้แล้ว โมเดลเสียงไหนให้ค่าความน่าจะเป็นสูงสุด เสียงที่ทดสอบที่เราไม่รู้ว่าเป็นเสียงอะไรนั้นก็คือเสียงของโมเดลเสียงนั่นเอง เมื่อเรารู้ว่าเป็นเสียงพูดใดแล้วก็นำไปควบคุมอุปกรณ์ได้ โดยใช้พอร์ตสื่อสารอนุกรม หรือขนานของเครื่องคอมพิวเตอร์นำไปใช้งานต่อไป

บทที่ 2
ทฤษฎีและหลักการ

2.1 ลักษณะของเสียงพูด

คนเราเปล่งเสียงพูดด้วยอวัยวะที่ใช้ในการออกเสียง (organs of speech) ทำเสียงตามที่มีในระบบภาษาของตน แม้ว่าคนที่อยู่ในสังคมเดียวกันจะใช้ภาษาเดียวกันแต่ถ้าพิจารณาเสียงที่เปล่งออกมาจริง ๆ แล้ว แต่ละครั้งก็อาจจะสังเกตุลักษณะที่แตกต่างกันได้ เราจึงสามารถจำเสียง จำวิธีของคนที่เราคุ้นเคยได้ เสียงพูดที่จะอธิบายด้วยหลักเกณฑ์ทางวิยาศาสตร์แม้ว่าในภาษาหนึ่ง ๆ จะมีเสียงต่างกันมากบ้างน้อยบ้าง แต่ละเสียงก็สามารถนำมาพิจารณาและอธิบายให้รู้ลักษณะการออกเสียงและตำแหน่งที่เกิดเสียงได้คำอธิบายนี้จะทำให้เข้าใจลักษณะเสียงทุกเสียง วิชาที่ว่าด้วยเสียงพูดเรียกว่า "วิชาสัทศาสตร์ " (Phonetics)

ในการศึกษาเสียงพูดแบ่งได้เป็น 2 ลักษณะ คือ

ก. สรีรสัทศาสตร์ (Articulatory) เป็นการศึกษาเสียงพูดจากอวัยวะและการเคลื่อนไหวอวัยวะที่ทำให้เกิดเสียงพูด การอธิบายนี้ก็จะอธิบายโดยอาศัยลักษณะและอาการเคลื่อนไหวของอวัยวะที่เกี่ยวข้องในการเปล่งเสียงพูดนั้น

ข. กลสัทศาสตร์ (Acoustic Phonetics) เป็นการศึกษาเสียงพูดจากลักษณะคลื่นเสียงที่ผู้พูดเปล่งออกมาแล้ว และผู้ฟังได้ยินว่ามีลักษณะทางกลศาสตร์อย่างไร การศึกษาตามแนวนี้ต้องอาศัยความรู้ทางฟิสิกส์และคณิตศาสตร์ช่วยอธิบายลักษณะของคลื่นเสียง

2.1.1 อวัยวะที่ใช้ในการออกเสียง

อวัยวะที่ใช้ในการออกเสียงมีอยู่หลายส่วน แต่ละส่วนสามารถทำให้เสียงพูดแตกต่างกันไป

อวัยวะเหล่านี้มีปากและส่วนต่าง ๆ ในปาก ช่องคอ กล่องเสียง ช่องว่างในปากและช่องว่างในจมูก อวัยวะที่ใช้ในการเปล่งเสียงพูดแบ่งออกเป็น 2 ประเภท คือ

ก. อวัยวะที่ใช้ในการทำอาการ (Articulator) คืออวัยวะที่เคลื่อนไหวเพื่อผลักลมไปยังส่วนต่าง ๆ อวัยวะที่สำคัญ คือ ลิ้น ซึ่งเป็นส่วนที่เคลื่อนไหวได้มากที่สุด

ข. อวัยวะซึ่งเป็นตำแหน่งที่เกิดเสียงต่าง ๆ (Point Of Articulator) คือตำแหน่งที่เกิดเสียงต่าง ๆ เช่น ริมฝีปาก ฟัน เพดาน ส่วนต่าง ๆ เป็นต้น

อวัยวะที่มีหน้าที่ในการออกเสียงโดยตรงมีดังนี้

2.1.1.1 ริมฝีปาก เป็นอวัยวะส่วนที่เคลื่อนไหวได้มากและทำให้เสียแตกต่างกันได้มากเราอาจบังคับให้ริมฝีปากอยู่ชิดกัน ห่างกัน ยื่นออกมา เป็นต้น ก็ได้ ลักษณะริมฝีปากแบบต่าง ๆ นี้ล้วนมีอิทธิพลต่อการออกเสียงและทำให้เสียงแตกต่างกันไปทั้งสิ้น

2.1.1.2 ฟัน เป็นอวัยวะที่ทำให้เกิดเสียงหลายชนิด เช่นเมื่อฟันกดลงบนริมฝีปากล่างลมที่ผ่านออกมาโดยแรงจะลอดช่องออกมาซึ่งทำให้เกิดเสียงได้

2.1.1.3 ปุ่มเหงือก เป็นส่วนนูนออกมาอยู่หลังฟันด้านบน ถ้าเอาลิ้นแตะดูจะรู้สึกว่าเป็นคลื่น ปุ่มเหงือกเป็นบริเวณที่ทำให้เกิดเสียงปุ่มเหงือก

2.1.1.4 เพดานแข็งหรือเพดานอ่อน คือส่วนที่เป็นกระดูกแข็ง

2.1.1.5 เพดานอ่อน คือส่วนเพดานที่ต่อเพดานแข็งไปข้างใน มีลักษณะเป็นกระดูกอ่อนที่ขยับขึ้นลงได้เวลาที่เราหายใจ เพดานอ่อนและลิ้นไก่ซึ่งอยู่ปลายเพดานอ่อนลดระดับลงมา เปิดช่องว่างให้ลมออกทางจมูกเวลาพูดส่วนใหญ่ปลายเพดานอ่อนและลิ้นไก่จะถูกยกขึ้นไปจรดหลังคอนอกจากเวลาออกเสียงนาสิกเท่านั้น

2.1.1.6 ลิ้นไก่ เป็นก้อนเนื้อเล็ก ๆ อยู่ปลายเพดานอ่อนตรงกลางปาก ลิ้นรั่วได้

2.1.1.7 ลิ้น ลิ้นเป็นส่วนที่เคลื่อนที่มากที่สุดในการออกเสียง ซึ่งสามารถแบ่งได้เป็น 3 ส่วน ดังนี้

ปลายลิ้น คือส่วนของปลายลิ้นซึ่งสามารถยกขึ้นไปแตะกับอวัยวะส่วนต่าง ๆ ได้

หน้าลิ้น คือส่วนที่อยู่ตรงข้ามกับเพดานแข็งถ้าวางลิ้นราบกับปากเช่นเดียวกับเวลาที่ได้พูด

หลังลิ้น ถ้าวางลิ้นราบกับปาก ลิ้นส่วนนี้จะอยู่ตรงข้ามกับเพดานอ่อน

2.1.1.8 แผ่นเนื้อปากหลอดลม เป็นก้อนเนื้อเล็ก ๆ คล้ายลิ้นไก่ อยู่ตรงโคนลิ้นลงไปในคอ มีหน้าที่ปิดช่องลมในขณะรับประทานอาหารและเปิดช่องลมเมื่อพูด

2.1.1.9 ช่องคอ อยู่ถัดจากช่องปากไปจนถึงเส้นเสียง

2.1.1.10 เส้นเสียง เป็นอวัยวะสำคัญที่ทำให้เกิดเสียง เส้นเสียงมีลักษณะเป็นกล้ามเนื้อแผ่นภายในกล่องเสียง ปิดขวางอยู่ปากช่องหลอดลมจากด้านหลังมาด้านหน้า ระหว่างเส้นเสียงจะมีช่องว่างซึ่งเป็นช่องผ่านให้ลมไปถึงปอดและออกมาจากปอดได้ ช่องนี้เรียกว่าช่องว่างระหว่างเส้นเสียง

2.1.1.11 ช่องจมูก หมายถึงโพรงในช่องจมูกซึ่งอยู่เหนือลิ้นไก่ขึ้นไป เป็นช่องที่ลมผ่านเส้นเสียงขึ้นมาจะผ่านออกไปทางจมูกได้เมื่อเวลาหายใจและเวลาออกเสียงนาสิก

เสียงที่เกิดขึ้นนั้นไม่ว่าจะเป็นเสียงประเภทใด จะผ่านไปตามทางเดินของเสียง โดยเริ่มตั้งแต่ช่องว่างระหว่างเส้นเสียงถึงริมฝีปาก ในทางวิทยาศาสตร์สรุปได้ว่าทางเดินของเสียงคือ ท่อนำเสียงที่มีรูปร่างไม่แน่นอน

2.1.2 การเกิดของเสียง (Speech Production)

การเกิดของเสียงแบ่งออกเป็น 3 ขั้นตอนคือ

ขั้นตอนที่ 1 จุดเริ่มต้น เป็นขั้นตอนที่ลมเริ่มถูกขับออกจากปอด ผ่านเข้าไปสู่ขั้นตอนที่ 2

ขั้นตอนที่ 2 การดัดแปลงลมที่เส้นเสียง อวัยวะที่ใช้ในขั้นตอนนี้คือส่วนที่ต่อจากปอดขึ้นมาถึงกล่องเสียงและที่กล่องเสียงเส้นเสียงจะทำหน้าที่เป็นลิ้นปิดเปิดทำให้เกิดเสียง 2 ชนิดคือ

1. เสียงก้อง (Voiced) เกิดจากเส้นเสียงปิดกั้นลมไว้ ลมที่ผ่านออกมาจะเพิ่มแรงดันมากขึ้นจนเส้นเสียงปิดเปิดสลับกันไป ทำให้เกิดเสียงก้องขึ้นมา ซึ่งสามารถเรียกความถี่ในการปิดเปิดของเส้นเสียงว่า ความถี่มูลฐาน

2. เสียงไม่ก้อง (Unvoiced) เสียงชนิดนี้เส้นเสียงจะเปิดตลอดเวลาที่ลมผ่าน ลมจึงผ่านออกมาได้สะดวกทำให้เกิดเสียงไม่ก้องขึ้น

ขั้นตอนที่ 3 การเปลี่ยนแปลงลักษณะเส้นเสียง อวัยวะที่ใช้ก็คือส่วนที่ต่อจากกล่องเสียงจนถึงริมฝีปาก โดยลมที่ผ่านออกจากกล่องเสียงจะทำให้เกิดเสียงในลักษณะต่าง ๆ ซึ่งจะเกิดจากการเปลี่ยนแปลงของอวัยวะที่กล่าวไว้ในหัวข้อที่แล้ว

2.2 การวิเคราะห์สัญญาณเสียงในช่วงเวลาสั้น ๆ

เนื่องจากว่าสัญญาณเสียงเป็นสัญญาณที่แปรตามเวลา มีการแปรเปลี่ยนที่ไม่แน่นอน เช่น ในขณะพูดช้า ๆ รูปร่างของโพรงเสียงรวมทั้งลักษณะรูปแบบของการกระตุ้นอาจจะไม่มีการเปลี่ยนแปลงในช่วงเวลาที่นานที่สุดประมาณ 200 มิลลิวินาที แต่ในขณะที่พูดอย่างรวดเร็ว อาจจะมีช่วงที่ไม่มีการเปลี่ยนแปลงสั้นมากคือประมาณ 80 มิลลิวินาที ก็ได้

ดังนั้นเทคนิคในการวิเคราะห์เสียงพูดส่วนใหญ่แล้ว จะสมมุติให้สัญญาณเสียงมีคุณสมบัติที่เปลี่ยนแปลงสัมพันธ์กับเวลาอย่างเชื่องช้า นั่นก็คือเราจะต้องแบ่งทำการวิเคราะห์หาพารามิเตอร์ของสัญญาณเสียงพูดในช่วงเวลาสั้น ๆ เหมือนมองผ่านช่องแคบ ๆ ที่เรียกว่า ช็อตไทม์วินโดว์ (short time window) เมื่อเทียบตามเวลาที่เสียงจะมีการเปลี่ยนแปลงได้ เพื่อจะได้มองเห็นเหมือนกับว่า เราหาพารามิเตอร์นั้น ๆ ได้มากจากสัญญาณที่อยู่ภายในช่องแคบ ๆ และมีความเสถียรภายในช่วงเวลาสั้น ๆ

เทคนิคส่วนใหญ่จะกำหนดให้พารามิเตอร์ได้มาจากค่าเฉลี่ยจากการวิเคราะห์ภายในช่องเวลาแคบ ๆ นั้น สำหรับกรณีที่ต้องพิจารณาพารามิเตอร์ต่าง ๆ ที่มีการเปลี่ยนแปลง ก็จะทำการแบ่งสัญญาณเสียงออกเป็นช่องหลาย ๆ ช่องหรืออาจจะเรียกว่า “ กรอบการวิเคราะห์ “ (analysis frame) ดังนั้นพารามิเตอร์ต่าง ๆ จะสามารถหาได้ทันเพียงพอที่จะติดตามการเปลี่ยนแปลงของสัญญาณ สำหรับในช่วงที่สัญญาณมีการเปลี่ยนแปลงช้า อาจจะกำหนดให้ช่องแคบมีขนาดใหญ่ประมาณ 100 มิลลิวินาที แต่ในทางตรงข้ามถ้าสัญญาณมีการเปลี่ยนแปลงเร็ว ก็ต้องใช้ช่องแคบที่มีขนาดเล็กมาก ๆ ประมาณ 5-10 มิลลิวินาที เพื่อป้องกันการสูญหายของรายละเอียดของสัญญาณถัดไป

2.2.1 รูปแบบของช่องแคบ (Windows)

การกำหนดขนาดของช่องแคบที่ใช้ขึ้นอยู่กับ

2.2.1.1 ช่องจะต้องสั้นพอ ที่จะทำให้คุณสมบัติของเสียงที่กำลังพิจารณาไม่มีการเปลี่ยนแปลงอย่างนัยสำคัญในช่องแคบนั้น

2.2.1.2 ช่องแคบจะต้องยาวพอที่จะทำให้การจัดเตรียมตัวอย่างของเสียงเพื่อจะนำไปคำนวณหาค่าพารามิเตอร์ให้ได้ตามต้องการอย่างเช่น ในกรณีที่มีสัญญาณรบกวนเข้ามาแทรกอยู่บางช่วงในสัญญาณเสียงด้วย ถ้าเราเลือกใช้ช่องแคบที่มีขนาดใหญ่กว่า เมื่อทำการหาค่าพารามิเตอร์โดยเฉลี่ย ก็จะทำให้ส่วนประกอบของสัญญาณรบกวนถูกตัดทิ้งหรือมองข้ามไป

2.2.1.3 ช่องแคบที่เหมาะสม ไม่ควรสั้นเกินกว่าช่วงหนึ่งคาบของสัญญาณเสียงในช่วงที่กำลังวิเคราะห์ เงื่อนไขนี้จะมีผลต่อค่าเฟรมเรท (frame rate ซึ่งก็คือจำนวนครั้งต่อวินาทีที่ทำการวิเคราะห์สัญญาณเสียง โดยการขยับช่องแคบไปเป็นคาบ ๆ ตามแกนเวลา) ตามปกติเฟรมเรทจะมีค่า

ประมาณ 2 เท่าของส่วนกลับของขนาดช่องแคบ นั่นก็คือช่องแคบถัด ๆ กันไปจะมีการซ้อนทับกัน 50 เปอร์เซ็นต์

การนำฟังก์ชั่นของช่องแคบที่มีช่วงขนาดจำกัด w(n) มาคูณเข้ากับสัญญาณ s(n) จะทำให้ได้กลุ่มตัวอย่างของเสียงพูดที่ถูกกำหนดน้ำหนักให้แปรไปตามรูปร่างช่องแคบ รูปแบบของช่องแคบที่ง่ายที่สุดก็คือ ช่องแคบสี่เหลี่ยม (rectangular window) ซึ่งมีนิยามดังนี้

$$w(n)=\begin{cases}1, n=0,1,\ldots,N-1\\ 0, n \quad other\end{cases} \tag{2.1}$$

ในสมการนี้ คือการกำหนดช่องของการวิเคราะห์ให้มีจำนวนตัวอย่าง N ตัวอย่าง

รูปแบบของช่องแคบในลักษณะนี้ ก็มีฟังก์ชั่นของช่องแคบหลายลักษณะด้วยกัน ตัวอย่างเช่น Blackman , Barlett , Hamming , Hanning เป็นต้น โดยฟังก์ชั่นของช่องแคบที่นิยมใช้กันมากในการวิเคราะห์สัญญาณเสียงก็คือ ฟังก์ชั่น Hamming ซึ่งมีรูปร่างตามลักษณะของ (cosine pulse) ที่นิยมดังนี้

$$w(n)=\begin{cases}0.54-0.46\cos(2\pi n)(N-1), n=0,1,\ldots,N-1\\ 0, \quad n \quad other\end{cases} \tag{2.2}$$

2.3 การหาค่าพลังงานของสัญญาณเสียง

พลังงานของสัญญาณเป็นตัวแทนอันหนึ่งที่เรามักจะนำมาใช้ในการวิเคราะห์ลักษณะต่าง ๆ ของสัญญาณทั่ว ๆ ไป โดยพลังงานของสัญญาณ s(n) ใด ๆ ที่แปรตามเวลาสามารถนิยามได้ว่า

$$E=\sum_{n=-\infty}^{\infty}S^2(n) \tag{2.3}$$

แต่สำหรับสัญญาณเสียงซึ่งเป็นสัญญาณที่แปรเปลี่ยนอยู่ตลอด ไม่มีเสถียรภาพตามเวลา เราจะต้องแบ่งสัญญาณออกมาพิจารณาเป็นช่วงเล็ก ๆ ตามแกนเวลาหรือเรียกว่าแบ่งออกเป็นเฟรม เช่น เฟรมละประมาณ 10-30 วินาที หรือเฟรมละ 100 ตัวอย่าง เป็นต้นดังนั้นก็จะสามารถหาพลังงานของเสียงในแต่ละเฟรมได้เป็น

$$E_l(m)=\sum_{n=0}^{N-1}s^2(n) \tag{2.4}$$

โดยที่ l แทนลำดับของเฟรมข้อมูลเสียง, l= 0,1,2,3,...,L

N แทนจำนวนข้อมูลเสียงในแต่ละเฟรม

การวัดค่าพลังงานดังในสมการที่ 2.3.2 นั้นมีข้อจำกัดตรงที่ว่ามันจะมีความไวต่อสัญญาณที่มีขนาดใหญ่ ๆ เนื่องจากเราใช้วิธียกกำลังสองค่าของสัญญาณอินพุท ดังนั้นการแก้ปัญหาอย่างหนึ่งก็คือวัดพลังงานของเสียงโดยใช้สมการดังนี้

$$E_1(m) = \sum_{n=0}^{N-1} |s(n)| \tag{2.5}$$

2.4 การหาค่าอัตราการตัดศูนย์ (Zero Crossing)

การเกิดอัตราการตัดศูนย์จะเกิดขึ้นจากการที่รูปคลื่นของสัญญาณมีการตัดกับแกนเวลานั่นคือค่าของสัญญาณจะมีการเปลี่ยนสัญลักษณ์ทางคณิตศาสตร์ นั่นเอง อัตราการเกิดอัตราการตัดศูนย์เป็นเครื่องมืออย่างง่ายที่ใช้อธิบายการเปลี่ยนแปลงข้อมูลของสัญญาณ ค่าอัตราการตัดศูนย์นั้นสามารถนำมาใช้ในการตัดสินว่าสัญญาณเสียงนั้นเป็นเสียงก้อง (Voiced) หรือเสียงไม่ก้อง (Unvoiced) เนื่องจากเสียงก้องส่วนใหญ่จะมีค่าพลังงานอยู่ในช่วงความถี่ต่ำ ส่วนเสียงไม่ก้องนั้นจะมีพลังงานอยู่ในช่วงความถี่สูง และค่าอัตราการตัดศูนย์ก็มีความสัมพันธ์โดยตรงกับค่าความถี่ของสัญญาณ ดังนั้นจึงอาจสรุปได้ว่า สัญญาณเสียงที่มีค่าอัตราการตัดศูนย์สูงจะเป็นเสียงไม่ก้องและสัญญาณที่มีค่าอัตราการตัดศูนย์ต่ำจะเป็นเสียงก้อง แต่อย่างไรก็ตามการกำหนดขนาดของค่าอัตราการตัดศูนย์ที่แน่นอนเพื่อจำแนกชนิดของเสียงนั้น จะต้องอาศัยผลจากการทดลองเป็นหลัก

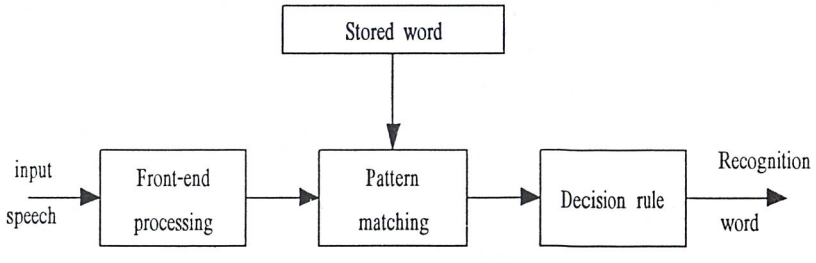
ในการหาค่าอัตราการตัดศูนย์สามารถทำได้โดยสมการดังนี้

$$Z = \frac{1}{2N} * \sum_{n=1}^{N} |Sign(s(n)) - Sign(s(n-1))| \tag{2.6}$$

$$Sign(s(n)) = \begin{cases} 1 & ; \ s(n) > 0 \\ 0 & ; \ \text{at other} \end{cases}$$

2.5 หลักการวิเคราะห์เสียง

จากการศึกษารูปแบบการรู้จำเสียงพูดแบบคำเดี่ยว (Isolated word recognition) มีรูปแบบและหลักการพื้นฐานดังรูปที่ 2.1



รูปที่ 2.1 แสดง Basic isolated-word recognition system

สัญญาณที่เข้าโมเดลเป็นคลื่นเสียงที่ผ่านกระบวนการก่อนหน้านี้ คือ การหาค่าพลังงาน และการหาค่าอัตราการตัดศูนย์ ส่วนผลลัพธ์ที่ได้เป็นเสียงวิเคราะห์ที่ได้จากสัญญาณเข้า โมเดลนี้มีการใช้กันอย่างแพร่หลาย แบ่งเป็นส่วนต่าง ๆ ได้ดังนี้

- Front-end processing เป็นขั้นตอนการวิเคราะห์เสียงที่ได้รับมาให้อยู่ในรูปแบบที่สามารถนำไปวิเคราะห์ได้
- Pattern matching เป็นขั้นตอนการสร้างโมเดลของเสียง
- Decision rule เป็นขั้นตอนการตัดสินใจในการเลือกโมเดลเสียงที่ใกล้เคียงกับเสียงทดสอบมากที่สุด

2.5.1 Front-end processing เป็นขั้นตอนที่ทำการแปลงข้อมูลที่มีอยู่มากมายเป็นส่วนเล็ก ๆ ที่สามารถแสดงคุณสมบัติของคลื่นเสียงนั้น ๆ โดยผ่านขั้นตอนดังนี้

- การประมาณเชิงเส้น (Linear Predictive coding: LPC)
- การจัดระดับเวกเตอร์ (Vector Quantization: VQ)

2.5.1.1 การประมาณเชิงเส้น (Linear Predictive coding)

เป็นการวิเคราะห์เพื่อหาค่าพารามิเตอร์ที่เหมาะสมเป็นข้อมูลในการวิเคราะห์เสียง โดยแบ่งสัญญาณเสียงที่จะวิเคราะห์ออกเป็นส่วน ๆ แต่ละส่วนใช้ระยะเวลาช่วงสั้น ๆ ประมาณ 15-20 มิลลิวินาที ซึ่งช่วงนี้สัญญาณเสียงพูดจะมีการเปลี่ยนแปลงคุณลักษณะอย่างช้า ๆ จนอาจถือว่าระบบกำเนิดเสียงมีคุณสมบัติที่คงที่

ก่อนที่จะนำข้อมูลเสียงพูดมาทำการประมาณเชิงเส้นนั้นจะต้องทำการปรับแต่งข้อมูลให้เหมาะสม โดยการตัดส่วนหัวและส่วนท้ายคำพูดที่เป็นส่วนเกินออกก่อน เนื่องจากเป็นส่วนของสัญญาณรบกวน จากนั้นก็นำข้อมูลมาผ่านขั้นตอนส่วนต่าง ๆ ดังนี้

2.5.1.1.1 การพรีเอมฟาซิส (pre-emphasis)

2.5.1.1.2 การแบ่งช่วงสัญญาณ (frame blocking)

2.5.1.1.3 การวินโดว์ (windowing)

2.5.1.1.4 การวิเคราะห์ออโตคอร์รีเลชั่น (autocorrelation analysis)

2.5.1.1.5 การหาอัตราการขยาย G

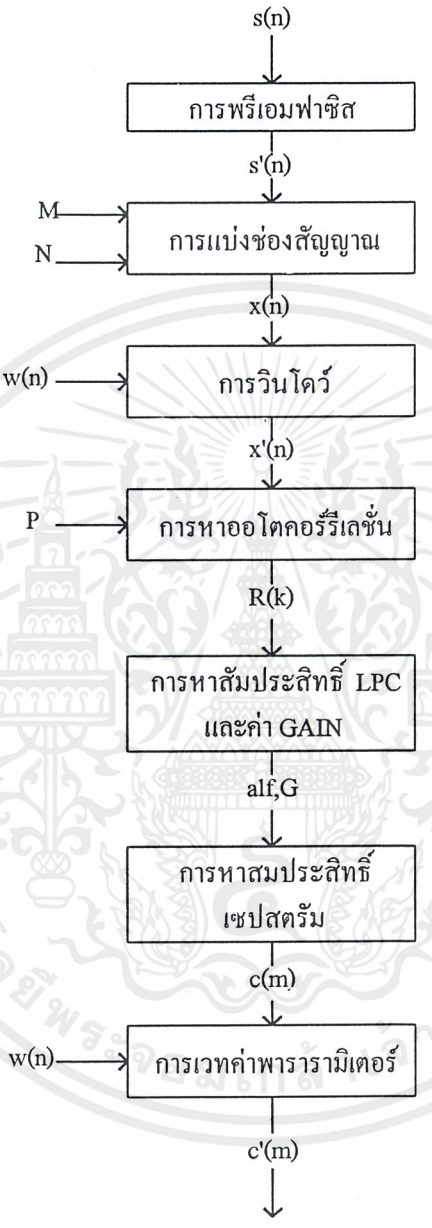
2.5.1.1.6 การหาสัมประสิทธิ์เซปสตรัม (ceptrum)

2.5.1.1.7 การเวทค่าพารามิเตอร์ (parameter weighting)

การประมาณเชิงเส้น จะทำการประมาณค่าสัญญาณจากผลรวมเชิงเส้นของสัญญาณก่อนหน้านี้ โดยใช้หลักการผลรวมกำลังสองของความคลาดเคลื่อนที่มีค่าต่ำสุด ซึ่งการประมาณเชิงเส้นมีอยู่หลายวิธี ได้แก่

- วิธีโควาเรียนซ์ (Co-variance Method)
- วิธีออโตคอร์รีเลชั่น (Auto-correlation Method)
- วิธีแลตทิส (Lattice Method)

และวิธีอื่น ๆ อีกหลายวิธี แต่ที่นิยมใช้คือวิธีออโตคอร์รีเลชั่น หรือ วิธีอัตตสัมพันธ์ (Auto-correlation) หลังจากผ่านขั้นตอนดังกล่าวจะได้ค่าสัมประสิทธิ์แอลพีซี (LPC Coeffient :α) และอัตราการขยาย (G) ซึ่งในแต่ละขั้นตอนอธิบายได้ดังนี้



รูปที่ 2.2 แสดงขั้นตอนการเตรียมสัญญาณในการวิเคราะห์

2.5.1.1.1 การพรีเอมฟาซิส (pre-emphasis)

เนื่องจากสัญญาณเสียงพูดของมนุษย์มีองค์ประกอบส่วนใหญ่อยู่ในช่วงความถี่ต่ำ เมื่อเทียบกับแถบความถี่ที่ปฏิบัติงาน (bandwidth) ไม่เกิน 5 กิโลเฮิรต์ ดังนั้น เพื่อให้อัตราส่วนสัญญาณเสียงต่อ

สัญญาณรบกวน (signal to noise ratio : SNR) มีค่าค่อนข้างคงที่ตลอดช่วงความถี่ที่ปฏิบัติงานนี้ เราจึงต้องมีการพรีเอมฟาซิส เพื่อเน้นความถี่สูงให้มีขนาดสูงขึ้น โดยการกรองสัญญาณด้วยวงจรกรองความถี่สูง (high pass filter) ซึ่งจะใช้วงจรกรองดิจิตอลแบบ first order มีรูปแบบสมการดังนี้

$$y(n)=a_0x(n)-b_1y(n-1) \tag{2.7}$$

มีฟังก์ชั่นถ่ายโอนเป็น

$$H(z)=1-a.z^{-1} \quad ; \quad 0.9<a<1.0 \tag{2.8}$$

สมมุติว่าสัญญาณเดิมเป็น S(n) เมื่อประมาณค่าสัญญาณแล้วจะเป็น S'(n) จะได้ว่า

$$S'(n)=s(n)-aS(n-1) \tag{2.9}$$

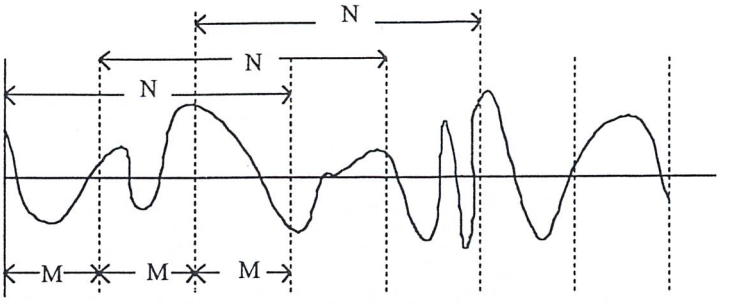
ยิ่งค่า α ใกล้ 1 เท่าใดความถี่สูงก็จะถูกขยายมากขึ้นเท่านั้น ค่า α ที่ควรใช้ในการพรีเอมฟาซิสคือ 0.9375

2.5.1.1.2 การแบ่งช่วงสัญญาณ (block into frames)

สัญญาณที่ผ่านการพรีเอมฟาซิสแล้วจะตัดแบ่งออกเป็นช่วง ๆ หรือเฟรม ช่วงละ N ตัวอย่างสัญญาณ การวิเคราะห์จะวิเคราะห์ทีละช่วงของแต่ละ N ตัวอย่างสัญญาณ โดยช่วงในการวิเคราะห์แต่ละช่วงจะถูกเลื่อนระยะไปเป็นระยะ M ช่วงสัญญาณ จะเห็นว่าถ้าค่า M โตกว่าค่า N ในการเลื่อนของช่วงในการวิเคราะห์ก็จะเป็นการสูญเสียสัญญาณส่วนหนึ่งทำให้ผลที่ได้ไม่ถูกต้องเท่าที่ควร ถ้าค่า M เล็กกว่า N ก็จะทำให้ตัวอย่างสัญญาณทุกตัวถูกนำมาวิเคราะห์ ยิ่งค่า M เล็กเท่าใด ความแม่นยำในการวิเคราะห์ก็จะยิ่งสูงขึ้นเท่านั้น แต่ก็จะทำให้การคำนวณช้าลง

2.5.1.1.3 การวินโดว์ (windowing)

พิจารณาช่วงสัญญาณ N ตัวอย่างสัญญาณของช่วงใด ๆ ที่ตัดมาวิเคราะห์จะเห็นว่าที่ขอบของเฟรมที่ตัดมาพิจารณามานี้มีความไม่ต่อเนื่องของสัญญาณ ถ้ามองในโดเมนความถี่ที่สูงเหล่านี้ เราจะคูณด้วยฟังก์ชั่นวินโดว์เพื่อลดความไม่ต่อเนื่องของสัญญาณที่ขอบของช่องของสัญญาณ และไม่ทำให้สเปคตรัมของสัญญาณในช่วงความถี่ต่ำเปลี่ยนไปไม่มากนัก



รูปที่ 2.3 แสดงการแบ่งช่วงสัญญาณ

ฟังก์ชั่นวินโดว์ที่ใช้ในการรวมกับสัญญาณมีหลายชนิด เช่น วินโดว์แบบสี่เหลี่ยม (Rectangular window) สำหรับฟังก์ชั่นวินโดว์ที่เหมาะสมที่ใช้กันก็คือ วินโดว์แบบแฮมมิ่ง (Hamming window)

ฟังก์ชั่นวินโดว์แบบแฮมมิ่งมีสมการดังนี้

$$w(n) = 0.54 - 0.46\cos(2\pi n/(N-1)) \quad \text{เมื่อ} \quad n = 0,1,2,\ldots,N-1 \qquad (2.10)$$

ในการวิเคราะห์เสียงโดยใช้ฟังก์ชั่นวินโดว์ จะพบว่าสัญญาณเสียงที่ผ่านการกรองโดยวินโดว์นั้นจะมีการแกว่งขึ้นลงมากน้อยขึ้นกับช่วงเวลาของการวินโดว์ (ความกว้างของวินโดว์) คือถ้าช่วงเวลาของการวินโดว์สั้นจะมีการแกว่งขึ้นลงอย่างรวดเร็ว และถ้าช่วงเวลาของการวินโดว์มากจะมีการแกว่งขึ้นลงช้า ๆ ดังนั้นใน
การเลือกช่วงเวลาในการวินโดว์จึงต้องให้อยู่ในช่วงเวลาที่เหมาะสม คือไม่ให้เอาท์พุทของสัญญาณแกว่งช้าหรือเร็วเกินไป ให้อยู่ในช่วงระหว่างเวลา 10 - 30 มิลลิวินาที เมื่อคูณกับฟังก์ชั่นวินโดว์แล้วก็จะได้

$$x'(n) = w(n)*x(n) \qquad (2.11)$$

2.5.1.1.4 การวิเคราะห์ออโตคอร์รีเลชั่น (auto – correlation)

สมมุติว่าสัญญาณเดิมเป็น s(n) การประมาณค่าสัญญาณเป็น s'(n) ดังนั้นสามารถอธิบายการประมาณเชิงเส้นด้วยสมการต่อไปนี้

$$s'(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \qquad (2.12)$$

เมื่อ α_k เป็นค่าคงที่ เรียกวิธีการนี้ว่าการประมาณเชิงเส้นอันดับ p โดยมีเงื่อนไขว่าค่า α_k ที่ใช้ในการประมาณเชิงเส้นจะต้องทำให้ผลรวมของกำลังสองของความคลาดเคลื่อน $\{ s(n) - s'(n) \}^2$

มีค่าน้อยที่สุด นั่นคือ $\sum e^2(n) = \sum \{ s(n) - s'(n) \}^2$ มีค่าต่ำที่สุด ซึ่งจะใช้การประมาณเชิงเส้นวิธีออโตคอร์รีเลชั่น (Auto - correlation Method) หรือ วิธีการอัตสัมพันธ์

การคำนวณออโตคอร์รีเลชั่นเป็นวิธีการหาค่าสัมประสิทธิ์ LPC โดยฟังก์ชั่นออโตคอร์รีเลชั่นซึ่งเป็นการเปรียบเทียบสัญญาณกับสัญญาณของตัวมันเองที่ถูกเลื่อนไปตามแกนเวลา ที่ใช้วิธีนี้ก็เนื่องมาจากเป็นการคำนวณที่มีการแก้สมการที่น้อยกว่าวิธีการอื่น ๆ และมีความแน่นอนในด้านเสถียรภาพ อีกทั้งมีการเก็บข้อมูลที่น้อยกว่าด้วย

การประมาณเชิงเส้นอันดับ P ของอันดับสัญญาณ s(n) และผลรวมเชิงเส้นของสัญญาณนี้ที่ถูกกำหนดด้วยการถ่วงน้ำหนักด้วยค่า α_1 ถึง α_p จะใช้ในการประมาณค่าของสัญญาณถัดไป s'(n) เขียนได้ในรูปของสมการดังนี้

$$s'(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \qquad ;10 \le p \le 26 \qquad (2.13)$$

ผลต่างของค่าสัญญาณประมาณ s'(n) กับ ค่าของสัญญาณปัจจุบันเรียกว่าค่าความคลาดเคลื่อน e(n)

$$e(n) = s(n) - s'(n) \qquad (2.14)$$

การวิเคราะห์การประมาณเชิงเส้นคือการหาค่า α_k ที่ทำให้ค่ากำลังสองของความคลาดเคลื่อนมีค่าน้อยที่สุด ซึ่งสามารถหาได้จากสมการ

$$R(k) = \sum_{n=0}^{N-1-k} s(n)s(n+k) \qquad ;0 \le k \le p \qquad (2.15)$$

และแทนค่าในเมตริกซ์เพื่อหา α_k จะได้

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

หรือ $$R_n . \alpha = r_n \qquad (2.16)$$

เมื่อ $$R_n = \begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & \cdots & R_n(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix},$$

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} \quad , \quad r_n = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

2.5.1.1.5 การหาอัตราการขยาย G

อัตราการขยายสามารถหาได้โดยตรงจากสมการ

$$G^2 = \frac{R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k)}{\sum_{m=0}^{N-1} u^2(m)} \qquad (2.17)$$

2.5.1.1.4 การหาสัมประสิทธิ์เซปสตรัม (Cepstrum)

หลังจากที่หาสัมประสิทธิ์ LPC และ Gain ใน 1 เฟรมแล้ว จะเปลี่ยนให้เป็นสัมประสิทธิ์เซปสตรัม เนื่องจากการรู้จำเสียงพูดนั้น สัมประสิทธิ์เซปสตรัมนี้เป็นพารามิเตอร์ที่มีลักษณะที่น่าเชื่อถือได้ดีกว่าสัมประสิทธิ์ LPC ทั้งยังมีความสัมพันธ์ใกล้ชิดกับการรับรู้เสียง ตามความรู้สึกของมนุษย์โดยแท้จริง สัมประสิทธิ์เซปสตรัมสามารถหาได้โดยตรงจากสัมประสิทธิ์ LPC ดังนี้

$C_0 = \ln G$ เป็นสัมประสิทธิ์ตัวแรกซึ่งเป็นเกน

$Q \approx \frac{3}{2} p$ โดย P เป็นมิติของสัมประสิทธิ์แอลพีซี และ Q เป็นสัมประสิทธิ์เซปสตรัม

$$C_m = \alpha_m + \sum_{k=1}^{m-1} \left(\frac{k}{m} \right) C_k \alpha_{m-k} \qquad ;1 \le m \le p \qquad (2.18)$$

$$C_m = \sum_{k=1}^{m-1} \left(\frac{k}{m} \right) C_k \alpha_{m-k} \qquad ;m > p \qquad (2.19)$$

2.5.1.1.7 การเวทค่าพารามิเตอร์ (parameter weighting)

เนื่องจากสัมประสิทธิ์เซปสตรัมที่ได้นั้น ช่วงลำดับต้น ๆ และลำดับท้าย ๆ ของเฟรมที่นำมาวิเคราห์จะเกิดความคลาดเคลื่อนมากกว่าบริเวณส่วนอื่น เพราะฉะนั้นจึงทำการถ่วงน้ำหนัก เพื่อลดค่าความคลาดเคลื่อนดังกล่าวนี้ ด้วยฟังก์ชั่นเวทติ้ง ดังนี้

$$W_m = \left[1 + \frac{Q}{2} sin(\frac{\pi n}{Q}) \right] \qquad ;1 \le m \le Q \qquad (2.20)$$

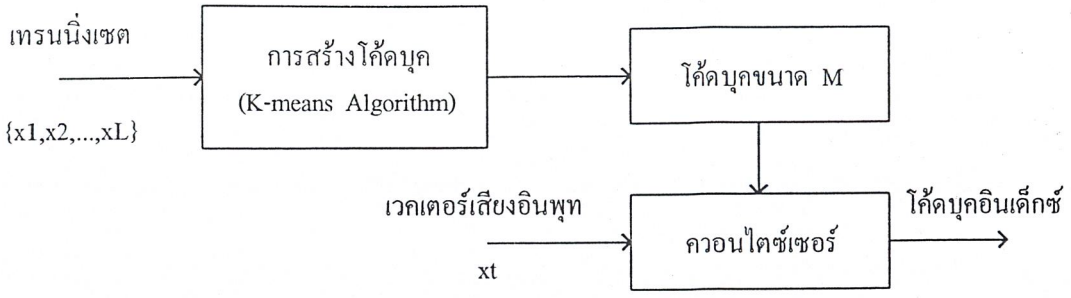
จะได้พารามิเตอร์สุดท้าย คือ

$$C'_m = C_m * W_m \qquad (2.21)$$

จากนั้นก็พิจารณาให้ครบทุกเฟรมของข้อมูล เมื่อพิจารณาเรียบร้อยแล้วก็จะนำไปจัดกลุ่มเสียงและสร้างแบบจำลองของเสียงเพื่อใช้ในการเปรียบเทียบต่อไป

2.5.1.2 การจัดระดับเวคเตอร์ (Vector Quantization)

เวคเตอร์ ควอนไตซ์เซชั่น เป็นการลดไดเมนชั่น (dimension) หรือขนาดของข้อมูลเวคเตอร์อินพุท หรือ เซตเทรนนิ่ง หรือ พารามิเตอร์ที่ได้จากขั้นตอน LPC จะถูกเลือกมากลุ่มหนึ่งซึ่งใช้เป็นตัวแทนของข้อมูลจำนวนหนึ่งหรือเรียกว่าการหา Codebook อินพุทที่เข้ามาจะถูกทำการเปรียบเทียบกับ Codebook ที่มีอยู่ โดยจะพิจารณาว่าอินพุทที่เข้ามานั้นห่างจาก Codebook ใดน้อยที่สุด อินพุทดังกล่าวจะถูกแทนด้วย เวคเตอร์โค้ด (index) นั้น อินพุททุกตัวที่เป็นสมาชิกของเวคเตอร์โค้ดใด ๆ จะถูกนำมาหาจุดศูนย์กลางร่วมใหม่ และนำจุดศูนย์กลางที่ได้นี้ไปทำการหาความคลาดเคลื่อนกับสมาชิกทุกตัว ถ้าค่าความคลาดเคลื่อนที่ได้มีค่ามากกว่าค่าที่กำหนดไว้ค่าหนึ่งหรือค่าที่ยอมรับได้ ก็จะนำศูนย์ใหม่ที่ได้นั้นไปเป็น Codebook แทน และจะทำการจัดกลุ่มอินพุทเข้ากับ Codebook ใหม่ที่ได้และหาความคลาดเคลื่อนอีกครั้งทำอย่างนี้ซ้ำ ๆ จนกระทั่งค่าความคลาดเคลื่อนมีค่าน้อยถึงค่าความคลาดเคลื่อนมีค่าน้อยถึงค่าที่ยอมรับได้ ก็ถือได้ว่าได้ Codebook ที่ดีสุดที่จะเป็นตัวแทนของอินพุททั้งหมด จะสังเกตุได้ว่าทุกครั้งที่มีการหา Codebook ใหม่นั้น ค่าความคลาดเคลื่อนที่ได้จะมีค่าลดลงทุกครั้งด้วย



รูปที่ 2.4 แสดงบล็อกไดอะแกรมของเวคเตอร์ควอนไตซ์เซอร์

การทำงานของการควอนไตซ์แบบเวคเตอร์ แบ่งได้เป็น 2 ขั้นตอนดังนี้

2.5.1.2.1 การสร้างโค้ดบุค (Codebook) โดยวิธีการของ K – means

จากขั้นตอนการประมาณเชิงเส้นของเสียงตัวอย่างจำนวนมากจะได้เทรนนิ่งเซตซึ่งประกอบด้วยเวคเตอร์สเปคตรัมจำนวน L เฟรม ; $x = \{ x_i ; 1 \leq i \leq L \}$ เฟรมละ P มิติ ; $x_i = \{ x_1, x_2, \ldots, x_p \}$ แล้วนำข้อมูลที่ได้มาทำการสร้างเป็นกลุ่มของแบบอ้างอิง

ในระบบการรับรู้เสียงพูดแบบต่างบุคคล จะใช้แบบอ้างอิงของคำหนึ่ง ๆ จากผู้พูดเป็นจำนวนมากเพื่อที่จะได้ครอบคลุมถึงความแปรปรวนต่าง ๆ ที่เกิดขึ้นระหว่างผู้พูดแต่ละคน เนื่องจากถ้าใช้แบบอ้างอิงจำนวนมาก เวลาที่ใช้ในการตอบสนองจะมาก เนื้อที่ในหน่วยความจำสำรองที่ใช้เก็บแบบอ้างอิงจะเพิ่มและเมื่อเพิ่มแบบอ้างอิงไปจนถึงระดับหนึ่งความถูกต้องในการรับรู้จะเริ่มคงที่ ดังนั้นจึงมีการจัดกลุ่มของแบบอ้างอิงใหม่เพื่อให้ได้แบบอ้างอิงที่เหมาะสม และสามารถใช้เป็นตัวแทนของแบบอ้างอิงที่มีอยู่ทั้งหมดได้ อัลกอริทึมที่ใช้ได้แก่ K – means Algorithm ซึ่งขั้นตอนที่ใช้ในการสร้างโค้ดบุคมีดังนี้

2.5.1.2.1.1 นำเทรนนิ่งเซตมาใช้ในการสร้างโค้ดบุค

ขนาดโค้ดบุคของการควอนไตซ์แบบเวคเตอร์ คือ $M = 2^B$ เวคเตอร์ (B – bit codebook) และเพื่อที่จะหาเซตของ M โค้ดบุคที่ดีที่สุด จำนวนเวคเตอร์อินพุทจะต้องมากกว่าขนาดโค้ดบุคมากเป็นจำนวน ($L >> M$)

2.5.1.2.1.2 การสุ่มค่าเริ่มต้น

การสุ่มค่าเริ่มต้น เป็นวิธีการหนึ่งในการออกแบบโค้ดบุค ซึ่งคือ การเลือกค่าเริ่มต้นของโค้ดบุคเรียกโค้ดบุคที่ได้จากการสุ่มค่าเริ่มต้นนี้ว่า แรนดอมโค้ดบุค (Random Codebook) ถึงแม้วิธีนี้จะไม่ใช่วิธีที่ดีนัก แต่โค้ดบุคที่ได้จากการสุ่มก็ให้ผลเป็นที่ยอมรับกัน

2.5.1.2.1.3 การหาความคลาดเคลื่อน

การวัดความคลาดเคลื่อนเป็นส่วนที่จำเป็นและมีประโยชน์ต่อการออกแบบโค้ดบุคมาก สมการทางพีชคณิตที่ใช้ในการหาระยะทางมีหลายวิธี แต่วิธีที่นิยมนำมาใช้ก็คือ การหาความคลาดเคลื่อนกำลังสองรวม (Total square error) ซึ่งเป็นวิธีการคำนวณที่ง่ายและรวดเร็ว

ถ้าสัญญาณมี P มิติ สามารถหาระยะห่างระหว่างสัญญาณอินพุท (x) กับเวคเตอร์โค้ด (y) ได้โดยสมการ

$$d(v_1, v_2) = \left\| v_1 - v_2 \right\|^2 = \sum_{i=0}^{k-1} (x_i - y_i)^2 \qquad (2.22)$$

2.5.1.2.1.4 การจัดกลุ่ม (Classification) และการหาจุดศูนย์กลางของกลุ่ม (Center cluster)

การจัดกลุ่มเป็นการแบ่งเวคเตอร์อินพุทเข้าไปตามกลุ่มต่าง ๆ ของแรนดอมโค้ดบุค โดยการพิจารณาระยะทาง หรือความคลาดเคลื่อนน้อยที่สุด ของแต่ละเวคเตอร์อินพุท x กับเวคเตอร์โค้ดบุค y ซึ่งเป็นโค้ดบุค จากนั้นจะทำการหาค่าเฉลี่ยของแต่ละกลุ่ม เพื่อเป็นค่ากลางของกลุ่มนั้น ๆ จะได้

$$\bar{Y} = \frac{1}{L} \sum_{i=1}^{L} x_i \qquad (2.23)$$

$\bar{Y}$ เป็นจุดศูนย์กลางซึ่งเป็นเวคเตอร์ที่อยู่ตรงกลางของกลุ่ม $\{x_i\}_{i=1}^{L}$ ซึ่งแต่ละมิติจะไม่ขึ้นแก่กันหมายความว่า แต่ละ y_k เป็นค่ากลางของ $\{x_i\}_{i=1}^{L}$

ทำ 2 ขั้นตอนนี้ซ้ำจนกว่าจะเกิดการลู่เข้า (Convergent) โดยความคลาดเคลื่อนรวมจะต่ำกว่าค่าหนึ่ง ๆ ซึ่งค่าความคลาดเคลื่อนรวมจะลดลงทุกครั้งที่มีการคำนวณซ้ำใหม่ จึงขึ้นกับค่าที่กำหนดว่าต้องการให้ความคลาดเคลื่อนรวมน้อยเท่าใด ค่ากลางดังกล่าวของแต่ละกลุ่มจะถูกเก็บเป็นเวคเตอร์โค้ด จะได้ว่า y
เป็นควอนไตซ์ของค่า x

$$y = q(x)$$

โดย $q(.)$ เป็นโอเปอร์เรเตอร์ของควอนไตซ์ และ y ถูกเรียกว่าเอาท์พุทเวคเตอร์ของค่า x โดย y เป็นค่าหนึ่งค่าใดใน $Y = \{y_i, 1 \le i \le M\}$ โดยที่ $y_i = [y_{i1}, y_{i2}, ..., y_{ip}]$ Y เป็นเซตของโค้ดบุค และ M เป็นขนาดของโค้ดบุค และ $\{y_i\}$ เป็นเซตของเวคเตอร์โค้ด y_i อาจเรียกว่าเป็นโค้ดบุคอ้างอิงก็ได้ และ M อาจเรียกว่าจำนวนระดับขั้น จะทำการแบ่งเวคเตอร์ x ไปใน M เซล $\{C_i, 1 \le i \le M\}$ เมื่อ x อยู่ในเซล C_i

$$q(x) = Y_i \quad \text{ถ้า} \quad x \in C_i$$

2.5.1.2.2 การเปรียบเทียบ

เวคเตอร์ควอนไตซ์เซชั่นที่ใช้เพื่อการออกแบบการรับรู้เสียงพูดนั้น มีจำนวนควอนไตซ์เซอร์จำนวน M ตัว ซึ่งหมายถึง มี M ระดับเสียงเพื่อการรับรู้ แต่ละระดับเสียงพิจารณามาจาก

เซตของข้อมูลเทรนนิ่ง ซึ่ง M เป็นดัชนีระดับ แต่ละเซตของเทรนนิ่งในแต่ละระดับจะเก็บเสียงที่อยู่ในระดับเดียวกัน

เมื่อมีเสียงที่เราไม่ทราบ (Unknown) ; xt เข้ามา จะเป็นอินพุทเข้าไปยังทุก ๆ ควอนไตซ์เซอร์ ค่าดัชนีระดับ (Index) ที่ถูกเลือกจะเป็นระดับที่มีความคลาดเคลื่อนเฉลี่ยน้อยที่สุด ซึ่งความคลาดเคลื่อนเฉลี่ยนั้นหาได้จากการวัดระยะทาง โดยใช้วิธีการหาความคลาดเคลื่อนกำลังสองรวม

วิธีการหาจุดศูนย์กลางของกลุ่มข้อมูล (K – means Algorithm)

มีขั้นตอนในการจัดกลุ่มดังนี้

ขั้นตอนที่ 1 เลือกศูนย์กลางของกลุ่มเริ่มต้นจำนวน K ตัว คือ Z_1 (1), $Z_2(1),\ldots,Z_k(1)$ ค่าเหล่านี้เลือกได้ตามชอบใจและปกติแล้วจะถือเอารูปแบบ k รูปแรกของชุดรูปแบบที่กำหนดให้

ขั้นตอนที่ 2 ในระหว่างตอนที่ k ให้กระจายรูปแบบ {x} ให้อยู่ในหมู่ K โดยใช้ความสัมพันธ์ดังต่อไปนี้

$$X \in S_j(k) \quad \text{ถ้า} \quad \|X - Z_j \quad (k)\| < \|X - Z_i(k)\| \qquad (2.24)$$

สำหรับทุก ๆ $i = 1,2,\ldots,K$ ยกเว้น $i = j$ โดยที่ $S_j(k)$ เป็นชุดของรูปแบบที่เป็นเจ้าของจุดศูนย์กลาง $Z_j(k)$ เงื่อนไขที่ตั้งขึ้นเองตามชอบใจในสมการที่ (2.24)

ขั้นตอนที่ 3 จากผลที่ได้ในตอนที่ 2 จะนำมาคำนวณจุดศูนย์กลางกลุ่มใหม่ $Z_j(k+1)$, $j = 1,2,\ldots,K$ นั่นคือเพื่อหาผลบวกกำลังสองของระยะทางจากทุกจุดใน $S_j(k)$ ไปยังจุดศูนย์กลางกลุ่มใหม่ที่มีค่าน้อยที่สุด ถ้าหาไม่ได้จุดศูนย์กลางกลุ่มใหญ่ $Z_j(k+1)$ จะถูกคำนวณต่อไปเพื่อทำให้

$$J_i = \sum_{X \in S_j(k)} \left\| X - Z_j(k+1) \right\|^2, \qquad j = 1,2,\ldots,K \qquad (2.25)$$

มีค่าน้อยที่สุด $Z_j(k+1)$ ที่มีค่าน้อยที่สุดนี้จะเป็นรูปแบบตัวกลางง่าย ๆ ของ $S_j(k)$ ดังนั้นจุดศูนย์กลางกลุ่มใหม่กำหนดได้โดย

$$Z_j(k+1) = \frac{1}{N_j} \sum_{X \in S_j(k)} X, \qquad j = 1,2,\ldots,K \qquad (2.26)$$

โดยที่ N_j คือจำนวนของรูปแบบใน $S_j(k)$ ชื่อตัวกลาง K (K – means) ตั้งขึ้นมาจากการที่จุดศูนย์กลางกลุ่มถูกปรับปรุงเรียงกันไปตามลำดับ

ขั้นตอนที่ 4 ถ้า $Z_j(k+1) = Z_j(k)$ สำหรับ $j = 1,2,\ldots,K$ แสดงว่ากฎเกณฑ์ความจริงได้เกิดย้อนกลับและดำเนินการจะสิ้นสุดลง แต่ถ้าไม่เป็นไปดังกล่าวให้ย้อนกลับไปยังตอนที่ 2

ปรากฏการณ์ของกฎเกณฑ์ความจริงตัวกลาง K ได้รับอิทธิพลจากจำนวนของจุดศูนย์กลางกลุ่มที่เจาะจงลงไป จากการเลือกจุดศูนย์กลางกลุ่มเริ่มแรก จากลำดับที่ตัวอย่างรูปแบบเกิดขึ้น และจากคุณ

สมบัติทางเรขาคณิตของข้อมูล ถึงแม้ว่าไม่ได้มีการพิสูจน์ให้เห็นว่ากฎเกณฑ์ความจริงนี้ได้เกิดการย้อนกลับขึ้นก็ตามแต่เชื่อได้ว่าผลที่พอจะมีได้จะไม่เกิดขึ้นเมื่อข้อมูลแสดงให้เห็นถึงคุณสมบัติเฉพาะที่มีความสัมพันธ์ไกลออกไปซึ่งกันและกัน ในทางปฏิบัติที่ใช้กันมากที่เกี่ยวกับกฎเกณฑ์ความจริงนี้จะต้องมีการทดลองกันมากมายโดยใช้ค่าของ K ที่แตกต่างกันไปเช่นเดียวกับการเลือกค่าเริ่มต้นที่แตกต่างกันไป

ตัวอย่าง เพื่อที่จะแสดงเกี่ยวกับกฎเกณฑ์ความจริงตัวกลาง K ขอให้เรามาพิจารณารูปแบบดังต่อไปนี้ ซึ่งเป็นเวคเตอร์แบบ 2 มิติ ได้แก่ $X_1 = (0,0)$, $X_2 = (1,0)$, $X_3 = (0,1)$, $X_4 = (1,1)$, $X_5 = (2,1)$, $X_6 = (1,2)$, $X_7 = (2,2)$,$X_8 = (3,2)$, $X_9 = (6,6)$, $X_{10} = (7,6)$, $X_{11} = (8,6)$, $X_{12} = (6,7)$, $X_{13} = (7,7)$, $X_{14} = (8,7)$, $X_{15} = (9,7)$, $X_{16} = (7,8)X_{17} = (8,8)$, $X_{18} = (9,8)$, $X_{19} = (8,9)$, $X_{20} = (9,9)$ โดยการทำตามวิธีการที่ได้กล่าวมาแล้วเราจะได้

ขั้นตอนที่ 1 ให้ $K = 2$ และเลือกให้ $Z_1(1) = X_1 = (0,0)'$, $Z_2(1) = X_2 = (1,0)'$

ขั้นตอนที่ 2 เพราะว่า $\| X_1 - Z_1(1) \| < \| X_1 - Z_i(1) \|$ และ $\| X_3 - Z_1(1) \| < \| X_3 - Z_i(1) \|$, $i = 2$ เราจะได้

$S_1(1) = \{ X_1 , X_3 \}$ ในทำนองเดียวกัน รูปแบบที่ยังเหลืออยู่ที่อยู่ใกล้ $Z_2(1)$ จะถูกกระทำดังนั้น $S_2(1) = \{ X_2 , X_4 , X_5 ,..., X_{20} \}$

ขั้นตอนที่ 3 ปรับปรุงจุดศูนย์กลางกลุ่ม

$$Z_1(2) = \frac{1}{N_1} \sum_{X \in S_1(1)} X$$

$$= \frac{1}{2}(X_1 + X_3)$$

$$= \begin{pmatrix} 0.0 \\ 0.5 \end{pmatrix}$$

$$Z_2(2) = \frac{1}{N_2} \sum_{X \in S_2(1)} X$$

$$= \frac{1}{18}(X_2 + X_4 + \cdots + X_{20})$$

$$= \begin{pmatrix} 5.67 \\ 5.33 \end{pmatrix}$$

ขั้นตอนที่ 4 เพราะว่า $Z_j(2) \neq Z_j(1)$, $j = 1,2$ เราจึงกลับไปทำขั้นตอนที่ 2 อีก

ขั้นตอนที่ 2 ด้วยจุดศูนย์กลางกลุ่มใหม่ เราจะได้ $\| X_k - Z_1(2) \| < \| X_k - Z_2(2) \|$, สำหรับ $k = 1,2,...,8$ และ $\| X_k - Z_2(2) \| < \| X_k - Z_1(2) \|$ สำหรับ $k = 9,10,....,20$ ดังนั้น $S_1 = \{ X_1 , X_2 ,..., X_8 \}$และ $S_2(2) = \{ X_9 , X_{10} ,..., X_{20} \}$

ขั้นตอนที่ 3 ปรับปรุงจุดศูนย์กลางกลุ่ม

$$Z_1(3) = \frac{1}{N_1} \sum_{X \in S_1(2)} X$$

$$= \frac{1}{8}(X_1 + X_2 + \cdots + X_8)$$

$$= \begin{pmatrix} 1.25 \\ 1.13 \end{pmatrix}$$

$$Z_2(3) = \frac{1}{N_2} \sum_{X \in S_2(2)} X$$

$$= \frac{1}{12}(X_9 + X_{10} + \cdots + X_{20})$$

$$= \begin{pmatrix} 7.67 \\ 7.33 \end{pmatrix}$$

ขั้นตอนที่ 4 เพราะว่า $Z_j(3) \neq Z_j(2)$ สำหรับ $j = 1,2$ เราจะต้องกลับไปยังขั้นตอนที่ 2

ขั้นตอนที่ 2 กระทำในลักษณะเดียวกับที่ได้ผลเกิดขึ้นมาแล้ว

$S_1(4) = S_1(3)$

$S_2(4) = S_2(3)$

ขั้นตอนที่ 3 กระทำเช่นเดียวกันกับทีได้ผลมาแล้ว

ขั้นตอนที 4 เพราะว่า $Z_j(4) = Z_j(3)$ สำหรับ $j = 1,2$ กฎเกณฑ์ความจริง (K – means) จะเกิดการย้อนกลับผลของจุดศูนย์กลางกลุ่มจะได้

$$Z_1 = \begin{pmatrix} 1.25 \\ 1.13 \end{pmatrix} \qquad Z_2 = \begin{pmatrix} 7.67 \\ 7.33 \end{pmatrix}$$

ถ้าเราจะสังเกตเอาจากข้อมูลที่กำหนดมาให้ ก็จะได้ผลเช่นเดียวกันกับทีได้คำนวณมานี้

2.6 ฮิดเดนมาร์คอฟโมเดล (Hidden Markov Model)

แบบจำลองมาร์คอฟเป็นแบบจำลอง (model) ทางสถิติซึ่งพัฒนาเพื่อแบ่งกลุ่มของอนุกรมทางเวลาหรือสัญญาณที่ไม่คงที่ นั่นคือใช้สำหรับการจัดกลุ่มของสัญญาณที่ไม่รู้จัก (Unknow signal) ให้ไปอยู่ในกลุ่มใดกลุ่มหนึ่งของสัญญาณ ซึ่งแบบจำลองมาร์คอฟ ได้ถูกนำมาประยุกต์ใช้ในการรู้จำเสียง

แบบจำลองมาร์คอฟแบ่งเป็น 2 ประเภทคือแบบต่อเนื่อง (Continuous) และแบบไม่ต่อเนื่อง (Discrete-time) ในที่นี้จะเลือกใช้แบบไม่ต่อเนื่องเพราะเป็นวิธีที่ซับซ้อนน้อยกว่าและใช้ได้ดีกับคำพูดสั้นๆ

2.6.1 ส่วนประกอบของแบบจำลอง HMM

2.6.1.1 N คือจำนวนสเตทในแบบจำลอง โดยสามารถย้ายจากสเตทหนึ่งไปอีกสเตทหนึ่งได้ เราให้เซตของสเตทเป็น { 1,2,3,..., N } และสเตทที่เวลา t ใดๆเป็น q_t

2.6.1.2 M คือจำนวนของค่าปรากฏต่อ 1 สเตท แทนด้วยสัญลักษณ์

$$V = \{ v_1, v_2, v_3, \ldots, v_M \}$$

2.6.1.3 $A = \{ a_{ij} \}$ คือความน่าจะเป็นในการเปลี่ยนสเตทที่

$$a_{ij} = P\{ q_{t+1} = j \mid q_t = i \} \text{ เมื่อ } 1 \leq i,j \leq N$$

2.6.1.4 $B = \{ b_j (k) \}$ คือความน่าจะเป็นของการเกิดค่าปรากฏที่

$$b_j (k) = P\{ o_t = v_k \mid q_t = j \} \text{ เมื่อ } j = 1,2,3,\ldots,N$$

2.6.1.5 π_i คือความน่าจะเป็นที่แต่ละสเตทจะเป็นสเตทเริ่มต้นเมื่อ

$$\pi_i = P\{ q_1 = i \} \text{ เมื่อ } i = 1,2,3,\ldots,N$$

2.6.2 โครงสร้างแบบจำลอง HMM

แบ่งตามลักษณะการเปลี่ยนสเตท (transition) ของเมตริกซ์ A

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

2.6.2.1 แบบ Egordic Model หรือ Fully Connected Model แบบจำลองนี้ทุกสเตทสามารถเปลี่ยน สเตทไปสเตทอื่นๆได้ทุกสเตท

2.6.2.2 แบบ Left – Right Model หรือ Bakis Model แบบจำลองนี้มีการเปลี่ยนสเตทจากซ้ายไปขวา มีคุณสมบัติการเปลี่ยนสเตทดังนี้

2.6.2.2.1 $a_{ij} = 0 , j < 1$ หมายความว่าเมื่อผ่านสเตทใดไปแล้วจะไม่มีการย้อนกลับมายังสเตทนั้นอีก

2.6.2.2.2 $\pi_i = \{ 0 ;$ เมื่อ $i \neq 1 , 1 ;$ เมื่อ $i = 1 \}$ หมายความว่า ลำดับของสเตทต้องเริ่มต้นที่สเตทที่ 1 สเตทที่เหลือจึงมีความน่าจะเป็นที่จะเป็นสเตทเริ่มต้นเท่ากับศูนย์ Left – Right Model นี้มีกฎข้อ บังคับการเปลี่ยนสเตทดังนี้ $a_{ij} = 0$ เมื่อ $i > i + \Delta i$ โดยค่าของ $\Delta i = 2$ หมายความว่าการเปลี่ยนสเตทจะสามารถเปลี่ยนได้เกิน 2 สเตท จะได้เมตริกซ์การเปลี่ยนสเตทเป็น

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

จะเห็นว่าสเตทสุดท้ายมีการเปลี่ยนสเตทเป็น

$a_{NN} = 1$, $a_{Ni} = 0$ เมื่อ $i < N$ แบบจำลองนี้จึงเหมาะกับสัญญาณที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง เช่น คำพูด

2.6.2.3 แบบ Parallel Left – Right Model มีคุณสมบัติการเปลี่ยนสเตทคล้ายกับแบบที่ 2 แต่มีความยืดหยุ่นมากกว่า

ปัญหาของ HMM มี 3 ข้อ ซึ่งต้องใช้อัลกอริทึมวิธีต่างๆในการคำนวณเพื่อแก้ปัญหานี้

ปัญหาที่ 1 เมื่อมีลำดับของค่าปรากฏ $O = \{ o_1o_2o_3...o_T \}$ และมีแบบจำลอง $\lambda = \{ A,B,\pi \}$ เราจะคำนวณหาค่า $P\{ O \mid \lambda \}$ ของลำดับของค่าปรากฏได้อย่างไร

ปัญหาที่ 2 เมื่อมีลำดับของค่าปรากฏ $O = \{ o_1o_2o_3...o_T \}$ และมีแบบจำลอง $\lambda = \{ A,B,\pi \}$ เราจะหาลำดับสเตท $q = \{ q_1q_2q_3...q_T \}$ ที่เหมาะสมในการให้ค่าปรากฏนั้นได้อย่างไร

ปัญหาที่ 3 จะหาแบบจำลอง $\lambda = \{ A,B,\pi \}$ ที่ให้ค่า $P\{ O \mid \lambda \}$ มากที่สุดได้อย่างไร

ลำดับของค่าปรากฏที่ใช้ปรับพารามิเตอร์ A,B,π เพื่อให้ได้แบบจำลองที่ดีที่สุดนั้นเรียกว่าลำดับเทรนนิ่ง (training sequence)

2.6.3 การคำนวณเพื่อแก้ปัญหาของ HMM

2.6.3.1 การแก้ปัญหาที่ 1 เป็นการคำนวณว่าแบบจำลองจะให้ความน่าจะเป็นที่จะได้ลำดับค่าปรากฏมากน้อยเพียงใด มีวิธีการเพื่อช่วยแก้ปัญหาโดยใช้กระบวนการต่อไปนี้

2.6.3.1.1 กระบวนการไปข้างหน้า (Forward Procedure)

เมื่อกำหนดให้ตัวแปรไปข้างหน้า (forward variable) $\alpha_t(i) = P\{ o_1o_2o_3...o_T, q_t = i \mid \lambda \}$ หมายถึงความน่าจะเป็นของการเกิดลำดับค่าปรากฏ $o_1o_2o_3...o_T$ ที่จะอยู่ในสเตทที่ i ณ เวลา t โดยมีแบบจำลองเป็น λ โดยเราจะคำนวณหา $\alpha_t(i)$ ได้ดังนี้

2.6.3.1.1.1 การเริ่มต้น (initialization)

เมื่อกำหนด $\alpha_1(i) = \pi_i b_i(o_1)$ ที่เวลาเริ่มต้น $t = 1$ และเหตุการณ์เริ่มต้น o_1 เมื่อ $1 \le i \le N$

2.6.3.1.1.2 การเหนี่ยวนำ (induction)

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \qquad (2.27)$$

เมื่อ $1 \le t \le T-1$ และ $1 \le j \le N$ หมายถึงความน่าจะเป็นของ สเตท j ที่เวลา t+1 ได้มาจากสเตท i ที่เป็นไปได้ถึง N สเตท ที่เวลา t

2.6.3.1.1.3 การสิ้นสุด (termination)

$$P\{ O \mid \lambda \} = \sum_{i=1}^{N} \alpha_T(i) \qquad (2.28)$$

ความน่าจะเป็นของลำดับค่าปรากฏ O ได้จากผลรวมของ $\alpha_T(i)$ จากทุกสเตท

2.6.3.1.2 กระบวนการย้อนกลับ (Backward Procedure)

เมื่อกำหนดให้ตัวแปรย้อนกลับ (backward variable) $\beta_T(i) = P\{ o_{t+1}o_{t+2}o_{t+3}\ldots o_T q_t = i \mid \lambda \}$ หมายถึงความน่าจะเป็นของลำดับค่าปรากฏส่วนหลังจากเวลา t+1 ไปจนจบโดยกำหนดว่าต้องอยู่ที่สเตท i ที่เวลา t และมีแบบจำลองเป็น λ เราจะคำนวณหา $\beta_T(i)$ ได้ดังนี้

2.6.3.1.2.1 การเริ่มต้น (initialization)

$\beta_T(i) = 1 ; 1 \le i \le N$ หมายความว่าที่เวลา T ณ ทุกๆสเตท จะได้ค่า $\beta = 1$

2.6.3.1.2.2 การเหนี่ยวนำ (induction)

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \qquad (2.29)$$

เมื่อ t = T-1,T-2,T-3,...,1 และ $1 \le i \le N$

2.6.3.2 การแก้ปัญหาที่ 2 เพื่อหาลำดับสเตทที่เหมาะสม

เราจะใช้วิธี วิทเทอร์บี อัลกอริทึม (Viterbi Algorithm) เพื่อหาลำดับสเตทที่ดีที่สุด ณ เวลา t หนึ่งๆ เมื่อกำหนดลำดับเหตุการณ์ $O = \{ o_1o_2o_3\ldots o_T \}$ โดยนิยามให้

$\delta_t(i) = \max_{q1,q2,q3,\ldots qt-1} P\{ q_1q_2q_3\ldots q_{t-1}, q_t = i, o_1o_2o_3\ldots o_t \mid \lambda \}$ หมายถึงความน่าจะเป็นสูงสุดของเส้นทาง (path) ณ เวลา t ซึ่งเริ่มนับจากเหตุการณ์ที่เวลาเริ่มต้นจนถึงเวลา t ที่สเตท i และโดยการอาศัยคุณสมบัติการเหนี่ยวนำ (induction) เราจะได้

$\delta_{t+1}(i) = [\max_i \delta_t(i) a_{ij}] b_j(o_{t+1})$ เราสามารถหาลำดับสเตทที่ดีที่สุดได้โดยใช้กระบวนการต่อไปนี้ เมื่อกำหนดให้ $\psi_t(i)$ เป็นอาเรย์ (array)

2.6.3.2.1 การเริ่มต้น (initialization)

$\delta_1(i) = \pi_i b_i(o_1)$ เมื่อ $1 \le i \le N$

$\psi_1(i) = 0$

2.6.3.2.2 การย้อนกลับ (recursion)

$\delta_t(i) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(o_t)$ เมื่อ $2 \le t \le T, 1 \le j \le N$

$\psi_t(i) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}]$

2.6.3.2.3 การสิ้นสุด (termination)

$P^* = \max_{1 \le i \le N} [\delta_T(i)]$

$q_T^* = \arg \max_{1 \le i \le N} [\delta_T(i)]$

2.6.3.2.4 เส้นทางเดินย้อนกลับ (Path backtracking)

$q_t^* = \Psi_{t+1}(q^*_{t+1})$ เมื่อ t = T-1,T-2,T-3,...,1

2.6.3.3 การแก้ปัญหาที่ 3 เพื่อหาโมเดลที่จะให้ผลตามลำดับค่าปรากฏหนึ่งๆ โดยเลือกค่าพารามิเตอร์ A,B,π ที่ดีที่สุดโดยใช้ กระบวนการทำซ้ำ (Iterative) วิธีที่เราเลือกใช้ คือ วิธีบาม-เวลช์ (Baum-Welch) หรือ EM (Expectation-Maximization) เมื่อนิยามให้

1. $\gamma_t(i) = P\{ q_t = i \mid O,\lambda \}$ หมายถึงความน่าจะเป็นที่จะอยู่ที่สเตท i ณ เวลา t โดยกำหนดลำดับเหตุการณ์ O และแบบจำลอง λ เราสามารถแสดงค่า $\gamma_t(i)$ ได้ดังนี้

$$\gamma_t(i) = P(q_t = i \mid O,\lambda)$$

$$= \frac{P(O, q_t = i \mid \lambda)}{P(O \mid \lambda)}$$

$$= \frac{P(O, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(O, q_t = i \mid \lambda)}$$

เนื่องจาก $P\{ O,q_t = i \mid \lambda \}$ มีค่าเท่ากับ $\alpha_t(i)\beta_t(i)$ จึงได้

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \qquad (2.30)$$

2. $\xi_t(i,j) = P\{ q_t = i, q_{t+1} = j \mid O,\lambda \}$ หมายถึงความน่าจะเป็นที่จะอยู่ที่สเตท i ที่เวลา t และ สเตท j ที่เวลา t+1 เมื่อกำหนดแบบจำลองและลำดับค่าปรากฏให้ ซึ่งจากนิยามของตัวแปรไปข้างหน้าและตัวแปรย้อนกลับ สามารถนำมาสัมพันธ์กับ $\xi_t(i,j)$ ได้ดังนี้

$$\xi_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O \mid \lambda)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)} \qquad (2.31)$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

และจะได้ความสัมพันธ์ของ $\gamma_t(i)$ กับ $\xi_t(i,j)$ ดังนี้

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \qquad (2.32)$$

และโดยที่ $\sum_{t=1}^{T-1} \gamma_t(i)$ คือจำนวนของการเปลี่ยนสเตทออกจากสเตท i ในลำดับค่าปรากฏ O

$\sum_{t=1}^{T-1} \xi_t(i,j)$ คือจำนวนของการเปลี่ยนสเตทออกจากสเตท i ไป j ในลำดับค่าปรากฏ O

ดังนั้นสามารถหาค่าพารามิเตอร์ได้ดังนี้

$$\bar{\pi}_j = \gamma_1(i);\ 1 \le i \le N \tag{2.33}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.34}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1,\ s.t.\ o_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{2.35}$$

จากกระบวนการข้างต้น ถ้าเราจะคำนวณซ้ำๆ โดยให้ $\lambda' = \{ A', B', \pi' \}$ แทน $\lambda = \{ A, B, \pi \}$ ซึ่งเป็นแบบจำลองเริ่มต้นแล้ว จะทำให้ความน่าจะเป็นของการเกิดลำดับค่าปรากฏ O ดีขึ้นจนกระทั่งถึงจุดวิกฤตจึงหยุด ซึ่งเราจะได้จุดวิกฤตของฟังก์ชันความน่าจะเป็นในกรณีที่ $\lambda' = \lambda$ หรือถ้า λ'มีความน่าจะเป็นมากกว่าแบบจำลอง λ ทำให้ได้ $P\{ O \mid \lambda' \} > P\{ O \mid \lambda \}$ นั่นก็คือเราเราจะได้แบบจำลอง λ' ใหม่ที่ทำให้เกิดลำดับค่าปรากฏได้ดีกว่า

2.7 การปรับค่าพารามิเตอร์ของ HMM

2.7.1 การสเกลลิ่ง (Scaling)

เนื่องจาก $\alpha_t(i)$ จะประกอบไปด้วยผลรวมของเทอมจำนวนมาก ซึ่งก็คือ

$$\left[\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(o_s) \right]$$

และเนื่องจากแต่ละเทอมของ a และ b มีค่า 1 อยู่แล้ว เมื่อพิจารณาผลรวมของการคูณค่ายิ่งน้อยลงเรื่อยๆ แสดงว่าเมื่อ t มากขึ้น แต่ละเทอมของ $\alpha_t(i)$ จะเข้าสู่ศูนย์ ทำให้ Dynamic Range ของการคำนวณ $\alpha_t(i)$ เกิน Rangeของเครื่องคอมพิวเตอร์ ทำให้ค่าที่ได้ไม่ถูกต้อง จึงได้มีการสเกลลิ่งขึ้นเพื่อทำให้ค่า $\alpha_t(i)$ อยู่ภายใน Dynamic Range ของเครื่องคอมพิวเตอร์ การสเกลลิ่งทำได้โดยการคูณ $\alpha_t(i)$ ด้วยสัมประสิทธิ์การ สเกลลิ่ง ซึ่งสัมประสิทธิ์นี้ไม่ขึ้นกับ i การสเกลลิ่ง $\beta_t(i)$ ก็เหมือนกัน หลังการคำนวณค่าการสเกลลิ่งก็จะตัดกันหมดไปเอง พิจารณาจากสมการ

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \quad (2.36)$$

เมื่อเราให้ $\alpha_t(i)$ แทน α ที่ยังไม่ได้สเกลลิ่ง

$\hat{\alpha}_t(i)$ เป็น α ที่สเกลลิ่งแล้ว

$\hat{\hat{\alpha}}_t(i)$ แทนเวอร์ชันของ α ก่อนการสเกลลิ่ง

เมื่อ $t=1$ เราจะให้ $\hat{\hat{\alpha}}_1(i) = \alpha_1(i)$

เมื่อ $c_1 = \frac{1}{\sum_{i=1}^{N} \alpha_1(i)}$

และ $\hat{\alpha}_1(i) = c_1 \alpha_1(i)$

เมื่อ $2 \le t \le T$ คำนวณ $\hat{\hat{\alpha}}_t(i)$ จากสมการ (2.27) ในเทอมของ $\hat{\alpha}_t(i)$ ค่าก่อน

$$\hat{\hat{\alpha}}_t(i) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} b_i(o_t) \quad (2.37)$$

เมื่อสัมประสิทธิ์การสเกลลิ่งเป็น

$$c_t = \frac{1}{\sum_{i=1}^{N} \hat{\hat{\alpha}}_t(i)}$$

เมื่อให้ $\hat{\alpha}_t(i) = c_t \hat{\hat{\alpha}}_t(i)$

จากสมการ (2.37) จะเขียนได้ว่า

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ij} b_j(o_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ij} b_j(o_t)} \quad (2.38)$$

และโดยการเหนี่ยวนำจะได้

$$\hat{\alpha}_{t-1}(j) = \left[\prod_{\tau=1}^{t-1} c_\tau\right] \alpha_{t-1}(j)$$

จะได้ว่า

$$\hat{\alpha}_t(i)=\frac{\sum_{j=1}^{N}\alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1}c_\tau\right)a_{ji}b_i(o_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1}c_\tau\right)a_{ji}b_i(o_t)}=\frac{\alpha_t(i)}{\sum_{i=1}^{N}\alpha_t(i)} \quad (2.39)$$

นั่นคือสเกล $\alpha_t(i)$ ได้โดยหารด้วยผลรวมของ $\alpha_t(i)$ ทั้งหมด และสเกล $\beta_t(i)$ ด้วยค่าเดียวกันนี้ในเทอมของการสเกลนี้สมการ (2.36) จะเป็น

$$\bar{a}_{ij}=\frac{\sum_{t=1}^{T-1}\hat{\alpha}_t(i)a_{ij}b_j(o_{t+1})\hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N}\hat{\alpha}_t(i)a_{ij}b_j(o_{t+1})\hat{\beta}_{t+1}(j)} \quad (2.40)$$

โดยแต่ละ $\hat{\alpha}_t(i),\hat{\beta}_{t+1}(j)$ จะได้เป็น

$$\hat{\alpha}_t(i)=\left[\prod_{s=1}^{t}c_s\right]\alpha_t(i)=C_t\alpha_t(i) \quad (2.41)$$

$$\hat{\beta}_{t+1}(j)=\left[\prod_{s=t+1}^{t}c_s\right]\beta_{t+1}(j)=D_{t+1}\beta_{t+1}(j) \quad (2.42)$$

ดังนั้นสมการ (2.40) จะเขียนได้เป็น

$$\bar{a}_{ij}=\frac{\sum_{t=1}^{T-1}C_t\alpha_t(i)a_{ij}b_j(o_{t+1})D_{t+1}\beta_{t+1}(j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N}C_t\alpha_t(i)a_{ij}b_j(o_{t+1})D_{t+1}\beta_{t+1}(j)} \quad (2.43)$$

ซึ่งเทอม C_tD_{t+1} จะเขียนได้ในเทอม

$$C_tD_{t+1}=\prod_{s=1}^{t}c_s\prod_{s=t+1}^{T}c_s=\prod_{s=1}^{T}c_s=C_T \quad (2.44)$$

ซึ่งไม่ขึ้นกับเวลา t ดังนั้น C_tD_{t+1} จะถูกตัดทิ้งทั้งเศษและส่วนของสมการ (2.43) ซึ่งทำให้ได้สูตรการคำนวณซ้ำๆ (reestimate) เดิมกลับคืนมา กระบวนการสเกลลิ่งดังกล่าวนี้สามารถใช้ได้กับสัมประสิทธิ์ β และ π ในการสเกลลิ่งนี้จะทำให้การคำนวณค่า $P\{O\mid\lambda\}$ เปลี่ยนไป เราจะไม่สามารถหาได้จากการรวมกับเทอมของ $\hat{\alpha}_T(i)$ แต่จะหาจากคุณสมบัติดังนี้

$$\prod_{t=1}^{T}c_t\sum_{i=1}^{N}\alpha_T(i)=C_T\sum_{i=1}^{N}\alpha_T(i)=1$$

ดังนั้นจะได้

$$\prod_{t=1}^{T} c_t P\{O \mid \lambda\} = 1$$

$$P(O \mid \lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{2.45}$$

ทำให้อยู่ในรูป log ของ P เพื่อไม่ให้เกิน Dynamic Range ของเครื่องคอมพิวเตอร์

$$\log\left[P\{O \mid \lambda\}\right] = \sum_{t=1}^{T} \log c_t \tag{2.46}$$

2.8 ลำดับของค่าปรากฏหลายเหตุการณ์ (Multiple Observation Sequence)

ในการใช้แบบจำลองแบบ Left-Rihgt นั้น การเทรนแบบจำลองต้องใช้หลายๆเหตุการณ์ของลำดับค่าปรากฏเข้ามาเทรน เพื่อให้ได้ค่าพารามิเตอร์ที่ถูกต้องมากขึ้น

ถ้าให้เซตของ k ลำดับค่าปรากฏเป็น

$$O = [O^{(1)}, O^{(2)}, O^{(3)}, \ldots, O^{(K)}]$$

เมื่อ $O^{(k)} = (o_1^{(k)} o_2^{(k)} o_3^{(k)} \ldots o_{Tk}^{(k)})$ เป็นลำดับค่าปรากฏของเหตุการณ์ที่ k โดยให้แต่ละเหตุการณ์เป็นอิสระต่อกันจะได้

$$P\{O \mid \lambda\} = \prod_{k=1}^{K} P\{O^{(k)} \mid \lambda\}$$

$$= \prod_{k=1}^{K} P_k$$

นำเอาจำนวนเหตุการณ์ของการเกิดค่าปรากฏแต่ละเหตุการณ์มารวมกัน จะได้สูตรหา a_{ij}, b_j (k)เป็น

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i) a_{ij} b_j(o_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i) \beta_t^k(i)} \tag{2.47}$$

และ

$$\bar{b}_j(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i) \beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i) \beta_t^k(i)} \quad s.t. \; o_t = v_\ell \tag{2.48}$$

ส่วนของ π_i ไม่ต้องคำนวณเนื่องจาก $\pi_1 = 1$ และ $\pi_i = 0$ เมื่อ $i \neq 1$ จะได้การสเกลลิ่งที่เหมาะสมของสมการ (2.47),(2.48) คือ

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1} \hat{\alpha}_t^{(k)}(i)a_{ij}b_j(o_{t+1}^{(k)})\hat{\beta}_{t+1}^{(k)}(j)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\sum_{i=1}^{N} \hat{\alpha}_t^{(k)}(i)a_{ij}b_j(o_{t+1}^{(k)})\hat{\beta}_{t+1}^{(k)}(j)} \qquad (2.49)$$

$$b_j(\ell) = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1} \hat{\alpha}_t^{(k)}(i)a_{ij}b_j(o_{t+1}^{(k)})\hat{\beta}_{t+1}^{(k)}(j)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\sum_{i=1}^{N} \hat{\alpha}_t^{(k)}(i)a_{ij}b_j(o_{t+1}^{(k)})\hat{\beta}_{t+1}^{(k)}(j)} \qquad (2.50)$$

$s.t. o_t = v_\ell$

2.9ระบบแบบจำลองมาร์คอฟ

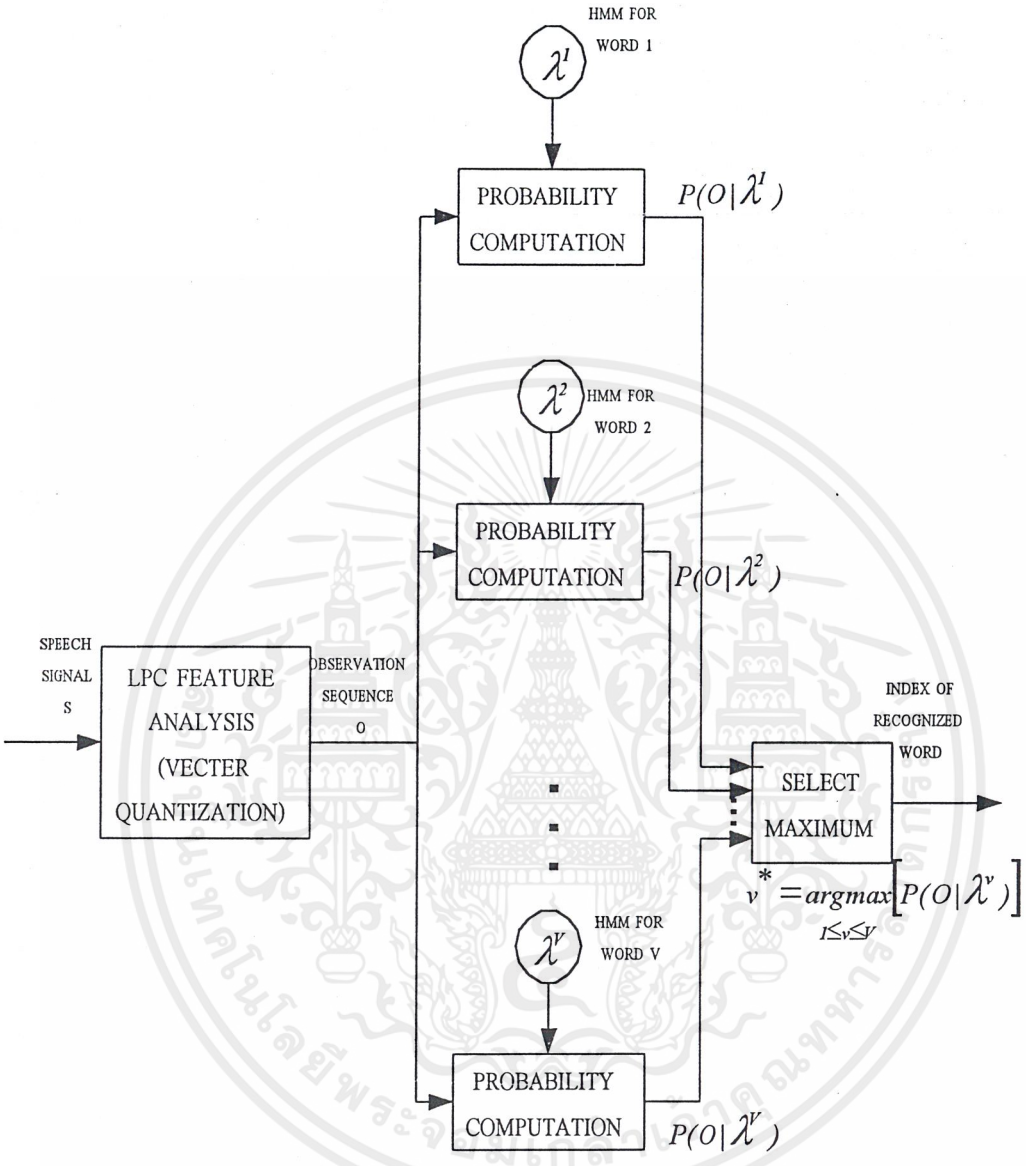
เมื่อเรามีคำศัพท์อยู่ V คำ ในขบวนการทำการรู้จำได้ เราจะต้องทำการสร้างแบบจำลองของคำแต่ละคำที่แตกต่างกัน คำแต่ละคำจะมีลำดับเทรนนิ่งที่ได้มาจากคุณลักษณะเฉพาะของคำศัพท์นั้นๆ การที่เราจะรู้จำคำพูดได้ต้องทำได้ตามบล็อคไดอะแกรมดังต่อไปนี้ บล็อคไดอะแกรมการรู้จำคำโดดด้วยแบบจำลองมาร์คอฟ

1.เมื่อมีคำศัพท์อยู่ V คำ เราต้องสร้างแบบจำลองมาร์คอฟ : λ_v ของแต่ละคำนั้นๆ นั่นคือการหาค่า {A,B,π} ที่เหมาะสมกับลำดับเทรนนิ่งของคำนั้นๆ

2.ในการจะรู้จำคำศัพท์แต่ละคำ เราจะทำการหาค่า $P\{O|\lambda\}$ ของทุกๆแบบจำลอง แล้วเลือกแบบจำลองที่มีค่าความน่าจะเป็นในการเกิดค่าปรากฏสูงสุดคือ

$$v^* = arg \max_{1 \le v \le V} [P\{O|\lambda_v\}]$$

คำศัพท์ที่สอดคล้องกับแบบจำลองดังกล่าวนี้ จะเป็นคำเดียวกับคำศัพท์ที่เราต้องการจะรู้จำนั่นเอง โดยขั้นตอนการคำนวณหาค่าความน่าจะเป็น จะใช้วิธี วิทเทอร์บี อัลกอริทึม ดังที่ได้กล่าวไปแล้วในตอนต้น



รูปที่ 2.5 บล็อคไดอะแกรมการรู้จำคำโดดด้วยแบบจำลองมาร์คอฟ

บทที่ 3
การคำนวณและการสร้าง

โปรแกรมการสั่งงานอุปกรณ์ด้วยเสียงพูด ประกอบด้วย 2 ส่วนใหญ่ ๆ คือ

1. ส่วนการวิเคราะห์เสียงพูด
2. ส่วนของการควบคุมอุปกรณ์

โปรแกรมทั้งสองส่วนนี้เขียนด้วยภาษาซี

3.1 ส่วนการวิเคราะห์เสียงพูด

โดยส่วนของการวิเคราะห์เสียงพูดสามารถแบ่งได้เป็น 2 ขั้นตอน คือ ขั้นตอนการเรียนรู้เสียงพูด และขั้นตอนการรับรู้เสียงพูด

1. ขั้นตอนการเรียนรู้เสียง

เป็นขั้นตอนที่ต้องการสร้างแบบจำลองของเสียงพูดมาเพื่อนำไปใช้ในขั้นตอนของการรู้จำเสียงพูดประกอบด้วยส่วนต่าง ๆ ดังนี้

1.1 การวิเคราะห์สัญญาณเสียงเบื้องต้น

มีการเลือกใช้ค่าต่างในการคำนวณและออกแบบดังต่อไปนี้

1) การพรีเอมฟาซีส

ใช้วงจรอันดับหนึ่ง ซึ่งมีฟังก์ชั่นถ่ายโอน คือ

$$H(z) = 1 - \alpha z^{-1}$$

ค่า α ที่ใช้คือ 15/16 = 0.9375

2) การแบ่งช่วงสัญญาณ

ขนาดของช่วงสัญญาณที่ใช้ในการวิเคราะห์มีเงื่อนไขในการเลือก คือ

- ค่า N ต้องสั้นพอที่คุณสมบัติของเสียงพูดไม่เปลี่ยนแปลง
- ค่า N ต้องยาวพอที่จำนวนของตัวอย่างมี เพียงพอสำหรับการหาสัมประสิทธิ์
- การเลื่อนในการวิเคราะห์ (ค่า M) ต้องไม่ข้ามข้อมูล

ดังนั้นค่า M จะน้อยกว่า N แต่ถ้าค่า M มีขนาดเล็กเกินไปจะทำให้การคำนวณช้าลง ดังนั้นจึงเลือกค่า M = 100 แซมเปิ้ล และค่า N = 300 แซมเปิ้ล

3) ความถี่ที่ใช้ในการสุ่มสัญญาณ

เนื่องจากความถี่ที่ใช้ในการแซมปลิ้งมากกว่า หรือเท่ากับสองเท่าของความถี่เสียง ($f_s \geq f_N$) ดังนั้น

$f_s \geq$ 8 kHz แต่เนื่องจากในโปรแอมกับ sound card มีค่าให้ใช้คือ 11.025 kHz

ดังนั้น ช่วงเวลาที่ใช้ในการวิเคราะห์แต่ละเฟรม คือ 300 / 11.025 = 27.21 msec และระยะเวลาที่ใช้ในการเลื่อนเฟรมก็คือ 100 / 11.025 = 9.07 msec

4) การเลือกวินโดว์ที่เหมาะสมสำหรับการวิเคราะห์เสียง

โดยพิจารณาลักษณะสเปคตรัม คือ

- ความถี่เรโซลูชั่นสูง (high frequency resolution) คือ มีโลบหลักแคบและแหลม

- การลดทอน (attenuation) นอกช่วงความถี่ที่ผ่านได้ต่ำ คือ ไซด์โลบมีค่าน้อย ฟังก์ชั่นวินโดว์มีหลายชนิด แต่ชนิดที่เหมาะสมที่สุดที่นำมาใช้ได้แก่ แฮมมิ่งวินโดว์ ซึ่งมีค่าไซด์โลบต่ำ (-40 dB) และเมนโลบแคบพอใช้

5) การหาคุณลักษณะของเสียงพูด

ใช้การประมาณพันธะเชิงเส้นในการวิเคราะห์หาค่าสัมประสิทธิ์ LPC ซึ่งการประมาณเชิงเส้นที่เลือกใช้กัน คือ วิธีอัตสัมพันธ์ (autocorrelation method) ซึ่งวิธีนี้มีการคำนวณที่ง่ายกว่าวิธีอื่น ๆ และมีความแน่นอนด้านของเสถียรภาพ อีกทั้งมีวิธีการเก็บข้อมูลที่น้อยกว่า

เนื่องจากเป็นวิธีการวิเคราะห์โดยวิธีอัตสัมพันธ์ อันดับการประมาณเชิงเชิงเส้น (P) ที่มากจะทำให้การประมาณเสียงมีความใกล้เคียงมากยิ่งขึ้น แต่ถ้าอันดับ P มีค่ามากเกินไปจะทำให้การคำนวณมีความยุ่งยาก และใช้เวลานาน ดังนั้น เพื่อความเหมาะสมค่าอันดับ P ที่ใช้ในการคำนวณคือ 12

1.2 การสร้างโค้ดบุค

จากการทดสอบ (L.R. Rabiner, S.E. Levinson และ Sondhi , 1982) จะได้ว่าที่ขนาดโค้บุคเท่ากับ 64 จะมีค่าความคลาดเคลื่อนเฉลี่ยประมาณ 0.2 ซึ่งเป็นค่าที่น้อยมากสำหรับเวคเตอร์ควอนไตซ์เซชั่น

การวัดค่าความคลาดเคลื่อน ใช้วิธีการคำนวณแบบ square error distortion ในการหาระยะทาง เนื่องจากเป็นวิธีการที่ง่าย และรวดเร็ว

การสร้างโค้ดบุค โดยนำเทรนนิ่งเซตที่ได้จากการประมาณเชิงเส้นมาผ่านกระบวนการดังนี้

1) สุ่มค่าโค้ดบุคเริ่มต้นมา 64 ตัว ๆ ละ 16 มิติ
2) หาระยะทางระหว่างโค้ดบุค กับเทรนนิ่งเซตแต่ละตัว โดยใช้ความคลาดเคลื่อนกำลังสอง
3) จัดกลุ่มของเวคเตอร์อินพุท โดยพิจารณาจากระยะทางที่น้อยที่สุด
4) หาจุดศูนย์กลางของกลุ่ม
5) ทำขั้นตอนที่ 3 และ 4 ซ้ำจนกว่าความคลาดเคลื่อนรวมจะน้อยกว่า 0.001 ซึ่งจุดศูนย์กลางที่ได้ก็คือโค้ดบุคนั่นเอง

1.2 การสร้างแบบจำลอง HMM ของเสียงพูด มีขั้นตอนดังนี้

1) สุ่มค่าเริ่มต้น a , b และกำหนดให้ π = [100000] ตามเงื่อนไขในการใช้แบบจำลองแบบ Left-Right
2) หาค่า α , β จากค่า a , b เริ่มต้น และลำดับค่าปรากฏ $O = \{ O_1 O_2 O_3 \dots O_T \}$ ซึ่งเรียกว่าลำดับ เทรนนิ่ง ตามวิธีของ forward-backward procedure โดยใช้ลำดับของค่าปรากฏหลาย ๆ ลำดับเข้ามาเทรนเพื่อความถูกต้องมากขึ้น
3) ทำการสเกลลิ่ง α เพื่อให้ค่าอยู่ในย่านที่คอมพิวเตอร์สามารถคำนวณได้อย่างถูกต้อง
4) หาค่าพารามิเตอร์ a , b , π ที่ให้ค่าความน่าจะเป็นสูงสุดที่จะเป็นแบบจำลอง λ ที่เหมาะสมของคำพูดนั้น
5) ตรวจสอบค่าพารามิเตอร์ของแบบจำลองที่ได้ว่าลู่เข้าหรือยัง โดยใช้วิธีการคำนวณค่า a , b ซ้ำประมาณ 50 – 70 รอบ เมื่อมีการเปลี่ยนแปลงน้อยมากจนเป็นที่พอใจตามระดับค่าที่ตั้ง

ไว้ ในที่นี้เราจะใช้ค่าเท่ากับ 10-5 ก็จะหยุด และได้ค่าพารามิเตอร์ a , b , π ของแบบจำลองที่ต้องการ

6) เก็บค่าพารามิเตอร์ a , b ,π ที่ได้จากข้อ 5 เป็นพารามิเตอร์ของแบบจำลองไว้

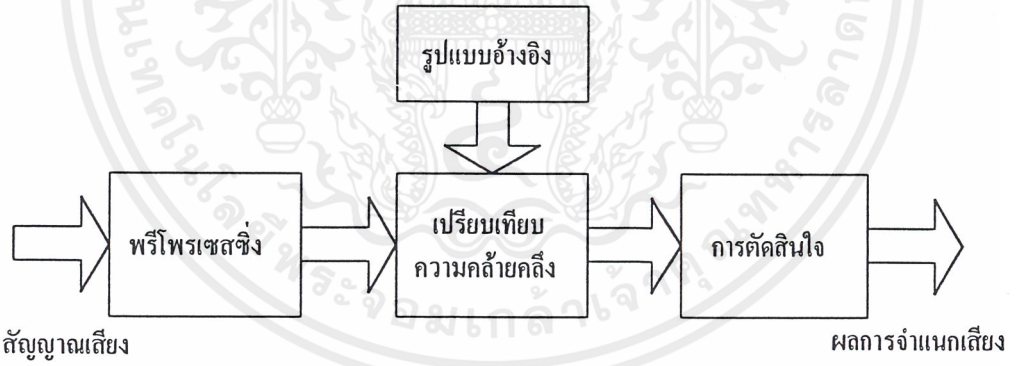
2. ขั้นตอนการรับรู้เสียงพูด

2.1 การหาดัชนีโค้ดบุค

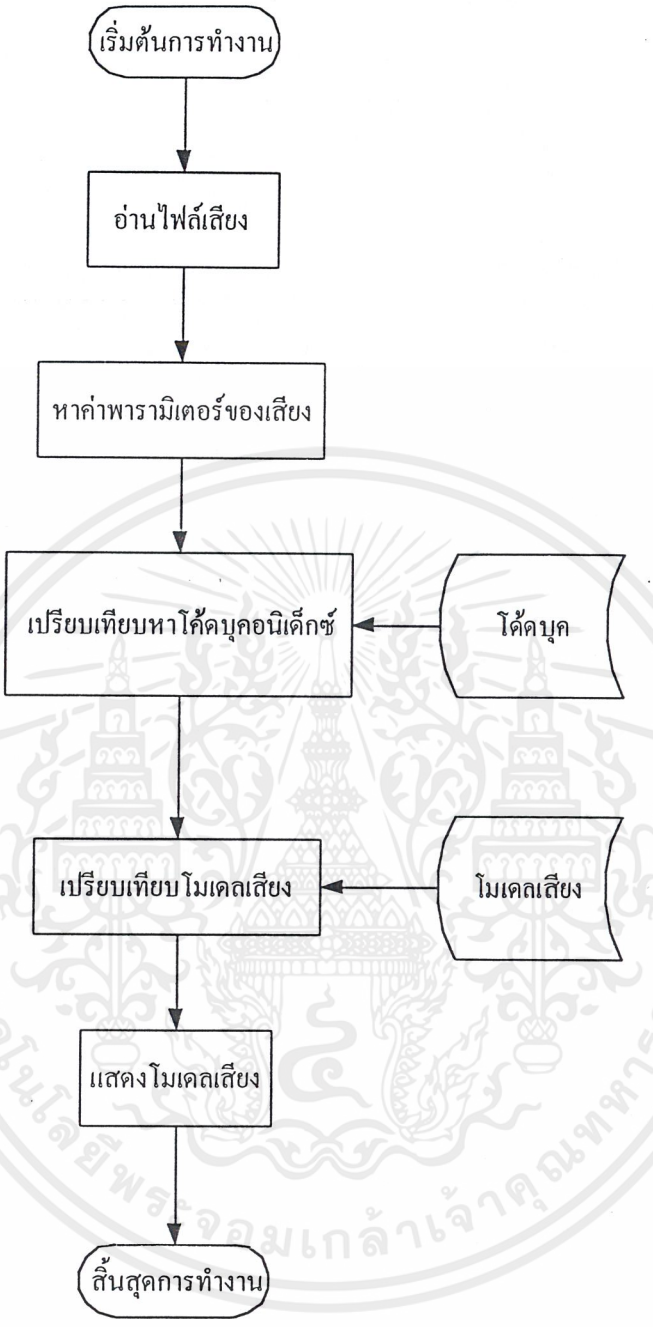
โดยการนำเวคเตอร์เสียงจากการประมาณเชิงเส้นมาทีละเสียง และเปรียบเทียบกับโค้ดบุคที่ได้จากการสร้างในขั้นตอนของการเรียนรู้เสียงพูดทีละเฟรม โดยวิธีความคลาดเคลื่อนกำลังสอง เวคเตอร์เสียงห่างจากโค้ดบุคใดน้อยที่สุดจะได้ว่าเป็นดัชนีโค้ดบุคเฟรมเสียงพูดนั้น และเก็บดัชนีโค้ดบุคของแต่ละเฟรมในแต่ละเสียงไว้เป็นลำดับค่าปรากฏ (observation sequence) สำหรับการสร้างแบบจำลองต่อไป

2.2 การรู้จำเสียง

หลังจากที่ได้แบบจำลอง HMM ของแต่ละเสียงพูดแล้ว เมื่อมีลำดับของค่าปรากฏ O = { O1O2O3...OT } ของเสียง unknown ซึ่งเป็นเสียงที่ต้องการทดสอบเข้ามา เราจะทำการคำนวณหาค่าความน่าจะเป็น P (O | λ) ทุกแบบจำลองของแต่ละเสียงพูดโดยใช้วิธี Viterbi Algorithm แล้วเลือกเอาเสียงพูดที่มีค่าความน่าจะเป็นสูงสุด ซึ่งก็คือเสียงพูดที่แบบจำลองจำได้นั่นเอง



รูปที่ 3.1 แสดงบล็อกไดอะแกรมการจำแนกเสียงพูด



รูปที่ 3.2 แผนผังขั้นตอนการทำงานของโปรแกรมวิเคราะห์เสียงพูด

3.2 การควบคุมอุปกรณ์

เมื่อส่วนของการรู้จำเสียงพูดสามารถรู้จำได้ว่าเป็นเสียงพูดใด เราก็นำเสียงพูดนั้นไปใช้ควบคุมอุปกรณ์ได้ โดยผ่านทางพอร์ตสื่อสารของคอมพิวเตอร์

บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลอง

ขั้นตอนในการทดลองแบ่งเป็น 2 ส่วน คือ ขั้นตอนของการเรียนรู้ และขั้นตอนของการวิเคราะห์

ขั้นตอนของการเรียนรู้

เป็นขั้นตอนที่นำข้อมูลของเสียงพูดที่ได้มีการบันทึกไว้แล้ว โดยใช้โปรแกรมบันทึกเสียงของการ์ดเสียง โดยใช้เสียงพูดเพียงคนเดียว ให้พูดคำว่า หน้า หลัง ซ้าย ขวา หยุด พูดคำละ 10,20 และ 30 ครั้ง และมีขั้นตอนของการวิเคราะห์เสียงดังนี้

1) นำเสียงทั้งหมดที่ได้เก็บบันทึกไว้แล้วมาทำการหาค่าพารามิเตอร์ ที่เป็นตัวแทนของเสียงพูดโดยใช้วิธี Linear Predictive Coding (LPC)

2) นำพารามิเตอร์เสียงที่ผ่านกระบวนการ LPC นั้นแล้วทั้งหมดมาทำการลดข้อมูลโดยวิธีการ Vector Quantization (VQ) ซึ่งจะได้เวคเตอร์ตัวแทนของเสียงที่ระบบสามารถรับรู้ได้ทั้งหมดออกมาเรียกว่า โค้ดบุค
(Code Book)

3) ทำการหาโค้ดบุคอินเด็กซ์ ซึ่งก็คือค่าปรากฏของแต่ละเสียงพูด

4) ทำการสร้างโมเดลของเสียงพูด โดยการนำค่าปรากฏของแต่ละเสียงพูดที่ได้จากข้อ 3 ไปสร้างโมเดลเสียงพูดแต่ละเสียง โดยใช้วิธีการของ ฮิตเดนมาร์คอฟ

ขั้นตอนการวิเคราะห์

เป็นขั้นตอนของการที่จะวิเคราะห์เสียงพูดที่ไม่ทราบว่าเป็นคำใด ว่ามีความเหมือนกับโมเดลของเสียงพูดใดที่เก็บไว้มากที่สุด ขั้นตอนในช่วงแรกจะคล้ายกับขั้นตอนการเรียนรู้ ดังนี้

1) นำเสียงพูดที่ต้องการทดสอบไปผ่านกระบวนการ LPC จะได้พารามิเตอร์ของเสียงพูดนั้นแล้วนำค่าพารามิเตอร์ที่ได้ไปเปรียบเทียบกับโค้ดบุค เพื่อหาโค้ดบุคอินเด็กซ์

2) นำโค้ดบุคอินเด็กซ์ หรือค่าปรากฏมาหาค่าความน่าจะเป็นกับทุกโมเดลเสียงพูดที่เก็บไว้โดยวิธีการ Viterbi Algorithm ก็จะได้ค่าความน่าจะเป็นของแต่ละโมเดลออกมา ค่าความน่าจะเป็นเมื่อเปรียบเทียบกับโมเดลเสียงพูดใดมีค่าสูงสุด ผลจะได้ว่าเสียงพูดที่นำมาทดสอบนั้นตรงกับโมเดลนั้นนั่นเอง

3) นำเสียงพูดที่จำได้ มาเขียนโปรแกรมควบคุมอุปกรณ์ต่อไป

ที่กล่าวมาข้างต้นเป็นแนวทางในการทำงานทั้งหมด และในการเขียนโปรแกรมก็แบ่งออกเป็น 2 ส่วนด้วยเหมือนกัน คือ ในการเรียนรู้เสียงพูดของคอมพิวเตอร์ และส่วนของการจดจำเสียงพูดได้ของคอมพิวเตอร์

ส่วนของการเรียนรู้เสียงพูดของคอมพิวเตอร์

เราจะทำการเขียนโปรแกรมด้วยภาษาซี มาทำการวิเคราะห์เสียงพูดมีการเขียนโปรแกรมดังนี้

ขั้นของการเรียนรู้เสียงพูด มีการเขียนโปรแกรมดังนี้

1) โปรแกรมชื่อ ENERGY.C ทำหน้าที่นำสัญญาณเสียงมาหาค่าพลังงานเพื่อทำการตัดเสียงที่ไม่ต้องการทิ้งไป

2) โปรแกรมชื่อ NORMALIZE.C ทำหน้าที่นำสัญญาณเสียงที่ผ่านการหาค่าพลังงานแล้ว มาทำให้ขนาดของไฟล์เสียงมีขนาดเท่ากันทุก ๆ เสียง

3) โปรแกรมชื่อ LPC.C ทำหน้าที่หาค่าพารามิเตอร์ของสัญญาณเสียงที่อยู่ในรูปของ wave form แล้วเก็บค่าพารามิเตอร์ของเสียงพูดแต่ละเสียงที่ได้ลงในไฟล์ (*.lpc)

4) โปรแกรมชื่อ MIX.LPC ทำหน้าที่รวมไฟล์ของสัญญาณเสียงที่อยู่ในไฟล์ (*.lpc) ทุกไฟล์ให้เป็นเพียง 1 ไฟล์ ให้ชื่อไฟล์นี้ว่า big.lpc

4) โปรแกรมชื่อ RANDCODE.C ทำหน้าที่สุ่มค่าโค้ดบุคเริ่มต้น (Random Codebook)

5) โปรแกรมชื่อ FINECODE.C ทำหน้าที่ในการหาค่าโค้ดบุคที่ดีที่สุด

6) โปรแกรมชื่อ INDEX.C ทำหน้าที่หาค่าดัชนีโค้ดบุค (Codebook Index)

7) โปรแกรมชื่อ HMM.C ทำหน้าหน้าที่หาโมเดลเสียง

<u>ส่วนของการจดจำเสียงพูดของคอมพิวเตอร์</u>

ขั้นตอนของการจดจำเสียงพูด มีการเขียนโปรแกรมดังนี้

1) โปรแกรม VITERBI.C ทำหน้าที่ในการรู้จำเสียงของเสียงที่นำมาทดสอบ โดยนำเสียงที่จะทำการทดสอบซึ่งผ่านการหาอินเด็กซ์แล้ว มาทำการเปรียบเทียบหาค่าความน่าจะเป็นของแต่ละโมเดลเสียง โมเดลเสียงใดให้ค่าความน่าจะเป็นสูงสุดก็บอกได้ว่าเป็นเสียงนั้น

4.2 ผลการทดลอง

4.2.1 สัมประสิทธิ์แอลพีซี สัมประสิทธิ์เซปสตรัล และค่าสัมประสิทธิ์เซปสตรัลที่ได้ทำการเวท (weighted)แสดงดังตารางที่ 4.1

ลำดับ	สัมประสิทธิ์แอลพีซี	สัมประสิทธิ์เซปสตรัล	ค่าเวทเซปสตรัล
1	1.641610	4.713723	4.713723
2	-0.083104	1.64161	4.207174
3	0.130053	-0.083104	-0.338913
4	-0.052434	0.084578	0.465178
5	0.019395	0.004393	0.029806
6	-0.006065	-0.00636	-0.050212
7	0.001700	0.005951	0.052333
8	0.014372	-0.00138	-0.013048
9	-0.077656	0.014459	0.142613
10	0.143253	-0.074409	-0.744087
11	-0.057588	0.128962	1.271991
12	-0.003145	-0.028148	-0.266205
13		-0.031361	-0.275794

14		0.020759	0.163877
15		-0.007065	-0.047937
15		-0.002343	-0.012885
17		0.003587	0.014627
18		-0.002898	-0.00742
19		0.005022	0.005022

ตารางที่ 4.1 แสดงค่าสัมประสิทธิ์แอลพีซี เซปสตรัล และเซปสตรัลที่ถูกเวทน้ำหนัก

จากตารางที่ 4.1 เป็นการแสดงค่าสัมประสทธิ์แอลพีซี ค่าสัมประสิทธ์เซปสตรัล และค่าสัมประสิทธิ์เซปสตัลที่ถ่วงน้ำหนัก ค่า P ที่ใช้คือ 12 โดยข้อมูลที่ได้เป็นสัญญาณเสียงขวา

4.2.2 การหาโค้ดบุค โดยการสุ่มค่าโค้ดบุคเริ่มต้นขึ้นมาก่อน 64 โค้ดบุค ในแต่ละโค้ดบุคจะมี 19 มิติ เพื่อที่จะนำมาหาโค้ดบุดสุดท้ายที่ดีที่สุด ตารางที่ 4.2 จะเป็นการเปรียบเทียบให้เห็นถึงความแตกต่างระหว่างค่าโค้ดบุคที่ได้จากการสุ่มเริ่มต้น กับโค้ดบุคที่ดีที่สุด โดยเปรียบเทียบเพียง 1 โค้ดบุค จากทั้งหมด 64 โค้ดบุ้ค

	โค้ดบุคที่ได้จากการสุ่ม	โค้ดบุคที่ดีที่สุด
มิติ	โค้ดบุคที่ 1	โค้ดบุคที่ 1
1	4.725902	4.747700
2	4.218761	4.209919
3	0.004552	0.084590
4	0.016781	0.032942
5	0.0027208	0.001929
6	0.042620	0.014530
7	0.053572	-0.043165
8	0.060835	0.071528
9	0.066493	0.028971
10	0.069402	-0.017952
11	0.083080	0.157214
12	0.087862	0.980936
13	-0.082742	-0.451068
14	-0.009704	-0.052024
15	0.00817	-0.005683

16	0.000614	0.003138
17	0.000434	0.005578
18	0.000213	-0.004236
19	0.000064	0.002043

ตารางที่ 4.2 แสดงการเปรียบเทียบโค้ดบุคที่ได้จากการสุ่ม กับโค้ดบุคที่ดีที่สุด

4.2.3 ผลการทดลองการรู้จำเสียง ได้ทำการทดลองเป็นดังนี้

เสียงที่ใช้เป็นต้นแบบเป็นเสียงของผู้ชายเพียงคนเดียวพูดคำว่า หน้า หลัง ซ้าย ขวา และหยุด โดยพูดคำละ 10,20 และ30 ครั้ง

ผลการทดสอบการจดจำเสียงที่ได้ ดังตารางต่อไปนี้

1.ใช้จำนวนเสียงพูดคำละ 10 ครั้ง เป็นเสียงต้นแบบ ทำการทดสอบกับเสียงต้นแบบ เสียงต้นแบบ (พูดใหม่) และเสียงนอกต้นแบบ โดยทดสอบเสียงละ 10 ครั้ง ได้ผลการทดลองดังนี้

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	9	-	-	1	90
ซ้าย	-	-	10	-	-	100
ขวา	-	-	-	10	-	100
หยุด	-	-	-	-	10	100

ตารางที่ 4.3 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 10 ครั้ง โดยทดสอบกับเสียงต้นแบบ

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	9	-	-	1	90
ซ้าย	-	-	10	-	-	100
ขวา	-	3	1	5	1	50
หยุด	-	-	-	-	10	100

ตารางที่ 4.4 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 10 ครั้ง โดยทดสอบกับเสียงต้นแบบที่ได้ทำการพูดใหม่

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	8	2	-	-	-	80
หลัง	-	9	-	-	1	90
ซ้าย	3	1	6	-	-	60
ขวา	-	5	-	-	5	0
หยุด	-	1	-	-	9	90

ตารางที่ 4.5 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 10 ครั้ง โดยทดสอบกับเสียงนอกต้นแบบ

2. ใช้จำนวนเสียงพูดคำละ 20 ครั้ง เป็นเสียงต้นแบบ ทำการทดสอบกับเสียงต้นแบบ เสียงต้นแบบ (พูดใหม่) และเสียงนอกต้นแบบ โดยทดสอบเสียงละ 10 ครั้ง ได้ผลการทดลองดังนี้

เสียงที่ใช้ในการทดสอบ	เสียงทีจำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	9	-	1	-	-	90
หลัง	-	10	-	-	-	100
ซ้าย	-	-	10	-	-	100
ขวา	-	-	-	10	-	100
หยุด	-	-	-	-	10	100

ตารางที่ 4.6 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 20 ครั้ง โดยทดสอบกับเสียงต้นแบบ

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	10	-	-	-	100
ซ้าย	-	-	10	-	-	100
ขวา	-	1	3	6	-	60
หยุด	-	-	-	-	10	100

ตารางที่ 4.7 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 20 ครั้ง โดยทดสอบกับเสียงต้นแบบที่ได้ทำการพูดใหม่

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	10	-	-	-	100
ซ้าย	3	1	6	-	-	60
ขวา	-	4	-	-	6	0
หยุด	-	-	-	-	10	100

ตารางที่ 4.8 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 20 ครั้ง โดยทดสอบกับเสียงนอกต้นแบบ

3. ใช้จำนวนเสียงพูดคำละ 30 ครั้ง เป็นเสียงต้นแบบ ทำการทดสอบกับเสียงต้นแบบ เสียงต้นแบบ(พูดใหม่) และเสียงนอกต้นแบบ โดยทดสอบเสียงละ 10 ครั้ง ได้ผลการทดลองดังนี้

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	8	1	-	1	80
ซ้าย	-	-	10	-	-	100
ขวา	1	-	-	9	-	90
หยุด	-	-	-	-	10	100

ตารางที่ 4.9 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 30 ครั้ง โดยทดสอบกับเสียงต้นแบบ

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	10	-	-	-	100
ซ้าย	-	-	10	-	-	100
ขวา	-	1	-	9	-	90
หยุด	-	-	-	-	10	100

ตารางที่ 4.10 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 30 ครั้ง โดยทดสอบกับเสียงต้นแบบที่ได้ทำการพูดใหม่

เสียงที่ใช้ในการทดสอบ	เสียงที่จำได้					เปอร์เซ็นต์ความถูกต้อง
	หน้า	หลัง	ซ้าย	ขวา	หยุด	
หน้า	10	-	-	-	-	100
หลัง	-	9	-	-	1	90
ซ้าย	3	1	6	-	-	60
ขวา	-	1	-	4	5	40
หยุด	-	-	-	-	10	100

ตารางที่ 4.11 แสดงผลการจำเสียงได้ของโมเดลเสียงที่ใช้ต้นแบบเสียงเพียงคนเดียวพูดคำละ 30 ครั้ง โดยทดสอบกับเสียงนอกต้นแบบ

การทดสอบ	สูงสุด	ต่ำสุด	เฉลี่ย
<u>กรณีที่ใช้เสียงต้นแบบพูดคำละ 10 ครั้ง</u>			
1. ทดสอบกับเสียงต้นแบบ	100%	90%	98%
2. ทดสอบกับเสียงต้นแบบ(พูดใหม่)	100%	50%	88%
3. ทดสอบกับเสียงนอกต้นแบบ	90%	0%	64%
ผลโดยรวม			83.33%
<u>กรณีที่ใช้เสียงต้นแบบพูดคำละ 20 ครั้ง</u>			
1. ทดสอบกับเสียงต้นแบบ	100%	90%	98%
2. ทดสอบกับเสียงต้นแบบ(พูดใหม่)	100%	60%	92%
3. ทดสอบกับเสียงนอกต้นแบบ	100%	0%	72%
ผลโดยรวม			87.33%
<u>กรณีที่ใช้เสียงต้นแบบพูดคำละ 30 ครั้ง</u>			
1. ทดสอบกับเสียงต้นแบบ	100%	80%	94%
2. ทดสอบกับเสียงต้นแบบ(พูดใหม่)	100%	90%	98%
3. ทดสอบกับเสียงนอกต้นแบบ	100%	40%	78%
ผลโดยรวม			90%

ตารางที่ 4.12 แสดงผลของค่าความถูกต้องทุกกรณี

บทที่ 5
บทวิจารณ์และบทสรุป

บทสรุป

แบ่งการสรุปได้เป็น 2 ขั้นตอน คือ ขั้นตอนการเรียนรู้เสียงพูด และขั้นตอนการจดจำเสียงพูดได้

ขั้นตอนการเรียนรู้เสียงพูด

1. เสียงพูดของคน ๆ เดิมจะมีลักษณะที่ต่าง ๆ กันทุกครั้งไป
2. การหาขอบเขตของคำ และการนอร์มอลไลซ์จะช่วยให้ความถูกต้องมีค่าแน่นอนขึ้น
3. จำนวนเสียงที่น้อยที่นำมาใช้สร้างโมเดล จะมีความถูกต้องมากถ้าใช้เสียงต้นแบบมาทดสอบ สำหรับจำนวนเสียงที่มากที่นำมาสร้างโมเดล จะมีความถูกต้องมากถ้าใช้เสียงนอกต้นแบบมาทดสอบ
4. จากวิธีการฮิดเดนมาร์คอฟนั้น เมื่อมีการคำนวณค่าต่าง ๆ จะเกิดค่าความผิดพลาดขึ้นได้ เนื่องจากค่าในการคำนวณมีขนาดเกินย่านของตัวแปรที่จะรับได้
5. การใช้เสียงจากผู้พูดจำนวนมากเพื่อนำมาสร้างโมเดล จะทำให้มีฐานข้อมูลขนาดใหญ่ ซึ่งจะทำให้เวลาในการคำนวณหาค่าโมเดลใช้ระยะเวลานานมากขึ้น

ขั้นตอนการจดจำเสียงพูด

1. เสียงพูดที่จะนำมาทดสอบจะต้องผ่านกระบวนการหาค่าพลังงาน นอร์มอลไลซ์ สัมประสิทธิ์แอลพีซี ค่าปรากฏ และค่าความน่าจะเป็นของแต่ละโมเดลด้วยวิธีการของ Viterbi ซึ่งเราจะรวมการทำงานทั้งหมดเป็น batch file ซึ่งอาจจะไม่สะดวกเท่าที่ควร
2. ผลที่ได้นั้นจะนำไปควบคุมอุปกรณ์ ซึ่งในที่นี้เราใช้เสียงนี้ไปควบคุมการเคลื่อนที่ของรถบังคับวิทยุ โดยผ่านทางพอร์ตขนานของเครื่องคอมพิวเตอร์
3. ความเร็วในการคำนวณนั้นจะมีความไวที่ค่อนข้างดีประมาณ 2 –3 วินาที

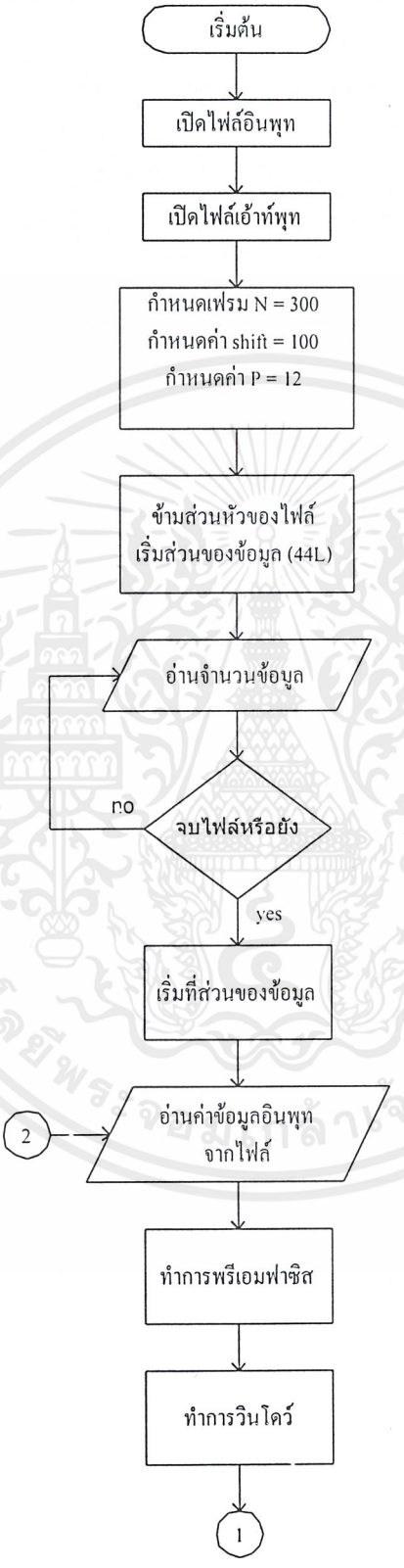
บทวิจารณ์และแนวทางพัฒนา

1. การทดสอบนี้ใช้เฉพาะเสียงต้นแบบที่เป็นผู้ชายเท่านั้น จึงควรพัฒนาให้สามารถจดจำได้ทั้งเสียงผู้ชายและผู้หญิง
2. ในการหาพารามิเตอร์ของเสียงนั้นจะมีความซับซ้อนน้อยกว่า เนื่องจากทำการหาในโดเมนของเวลา
3. ควรจะมีแหล่งต้นแบบของเสียงที่จะนำมาสร้างโมเดลมากเพื่อครอบคลุมความแปรปรวนของสัญญาณเสียง
4. ควรจะเขียนโปรแกรมที่มีการทำงานบนวินโดว์ เพื่อที่จะมีความคล่องตัวมากยิ่งขึ้น

ภาคผนวก

รูปที่ 1 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม Energy.c

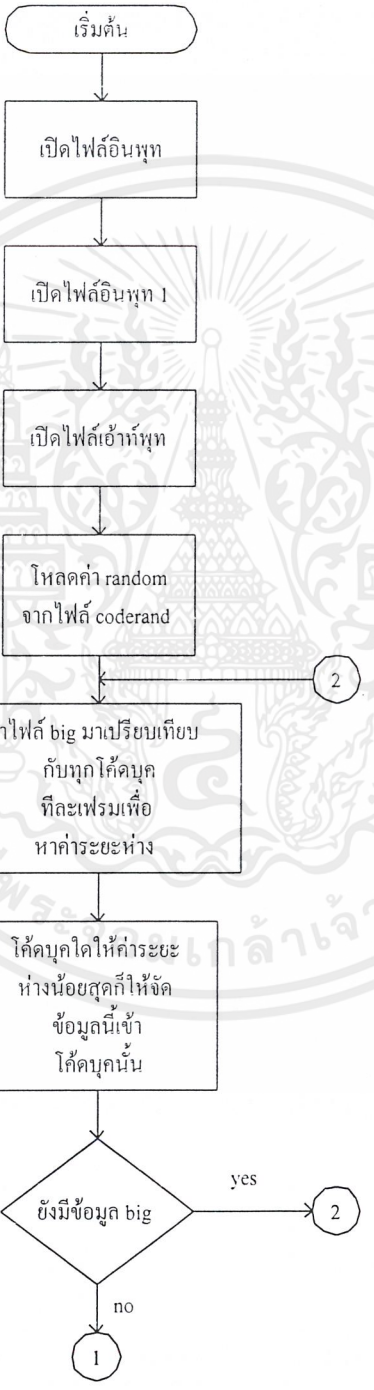
รูปที่ 2 โฟล์วชาร์ตแสดงการทำงานของโปรแกรม Normalize.c

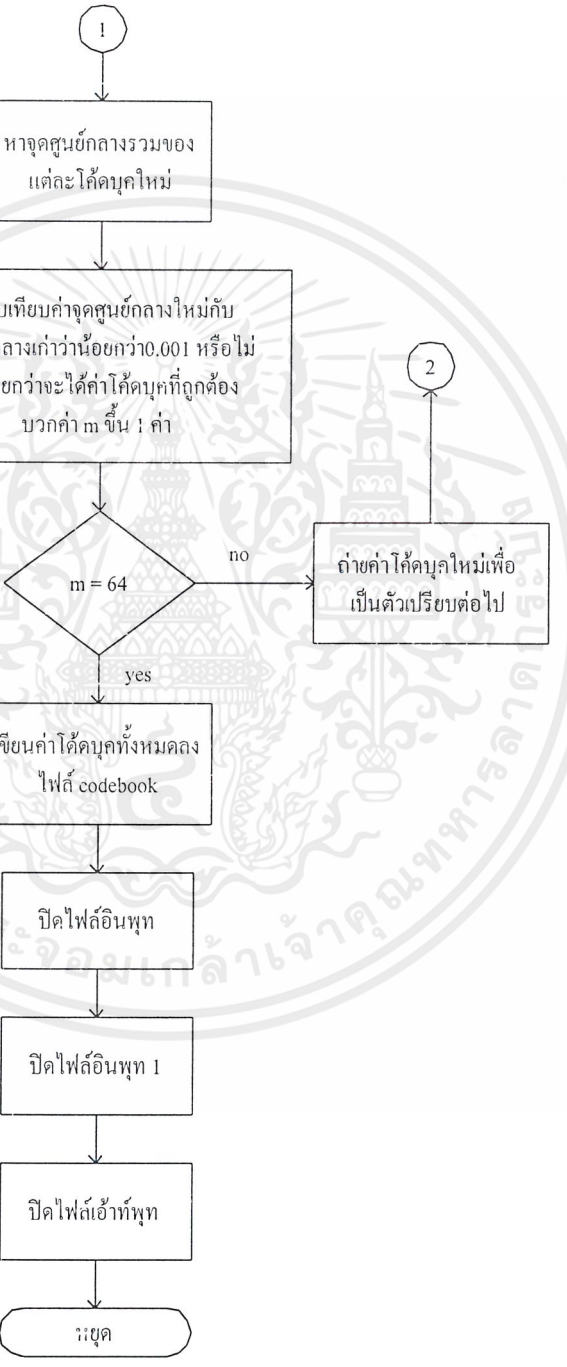


รูปที่ 3 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม LPC.C

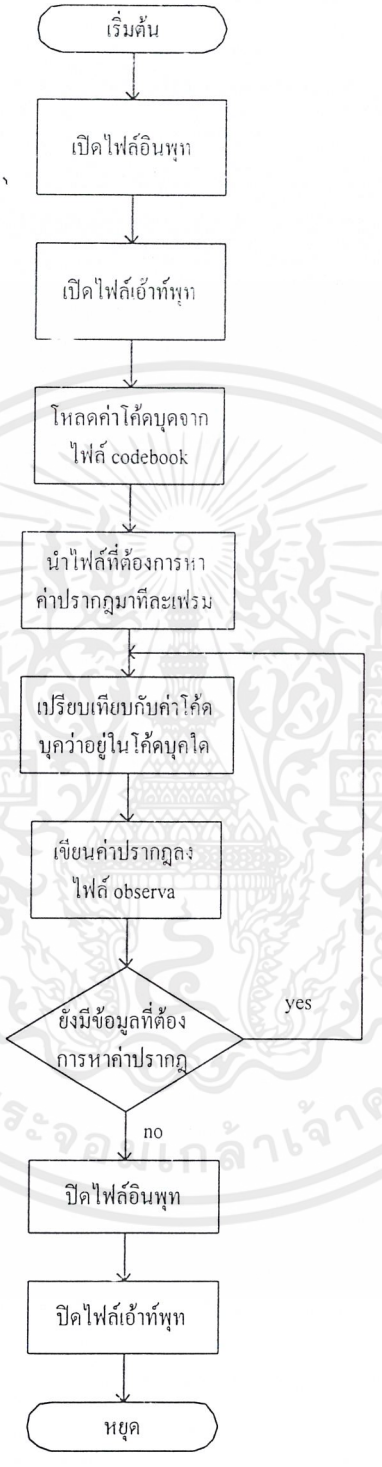
รูปที่ 4 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม MIX.C

รูปที่ 5 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม RANDCODE.C

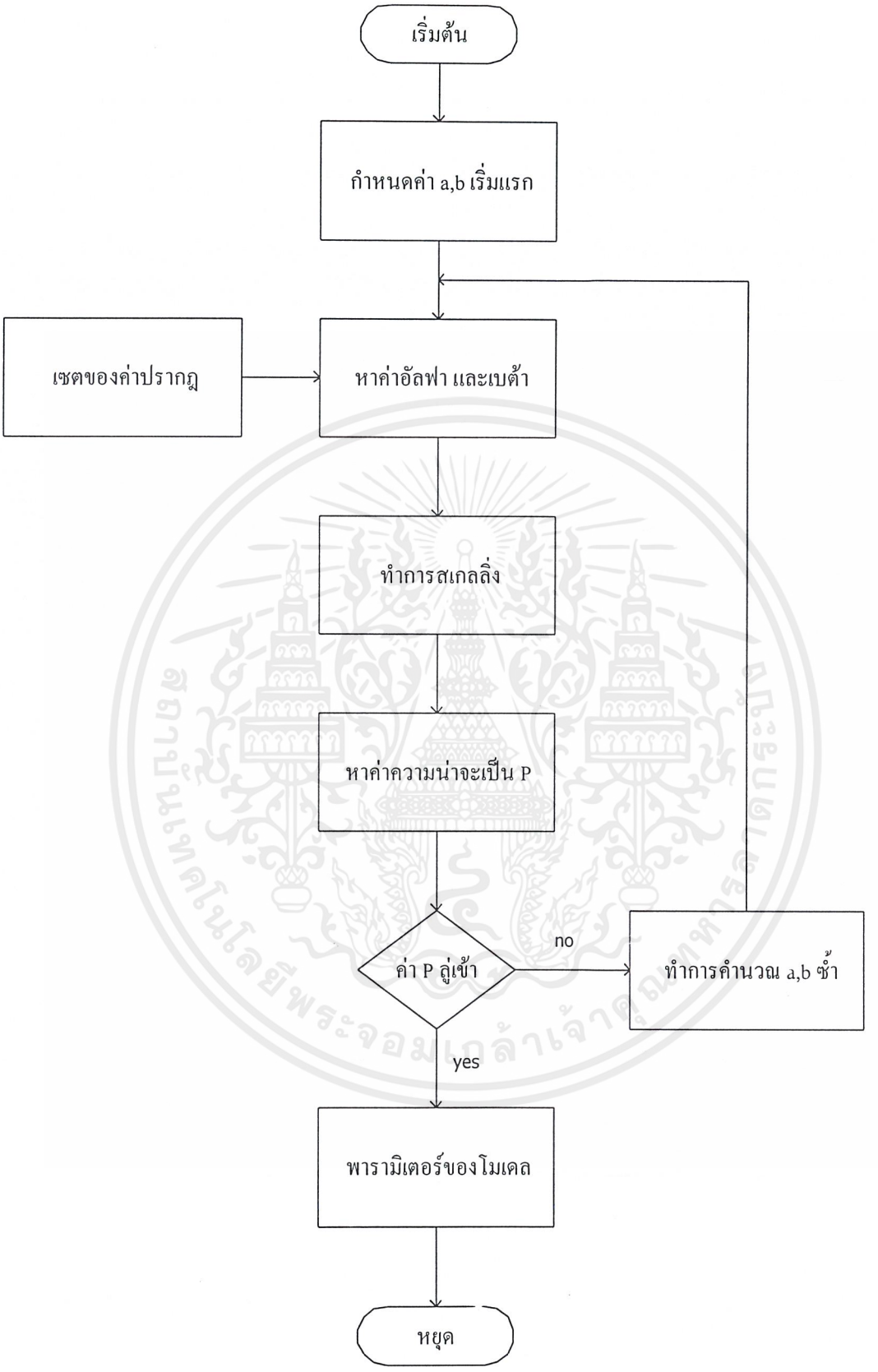




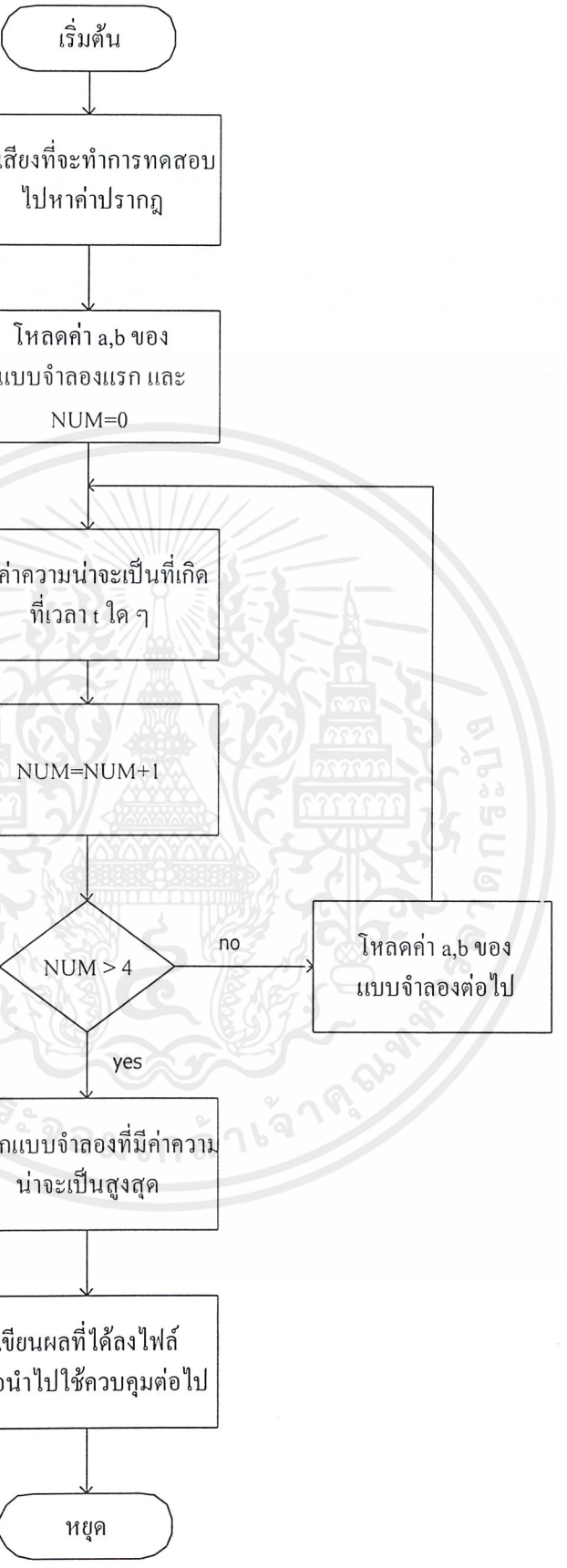
รูปที่ 6 โฟลว์ชาร์ตแสดงขั้นตอนของโปรแกรม FINDCODE.C



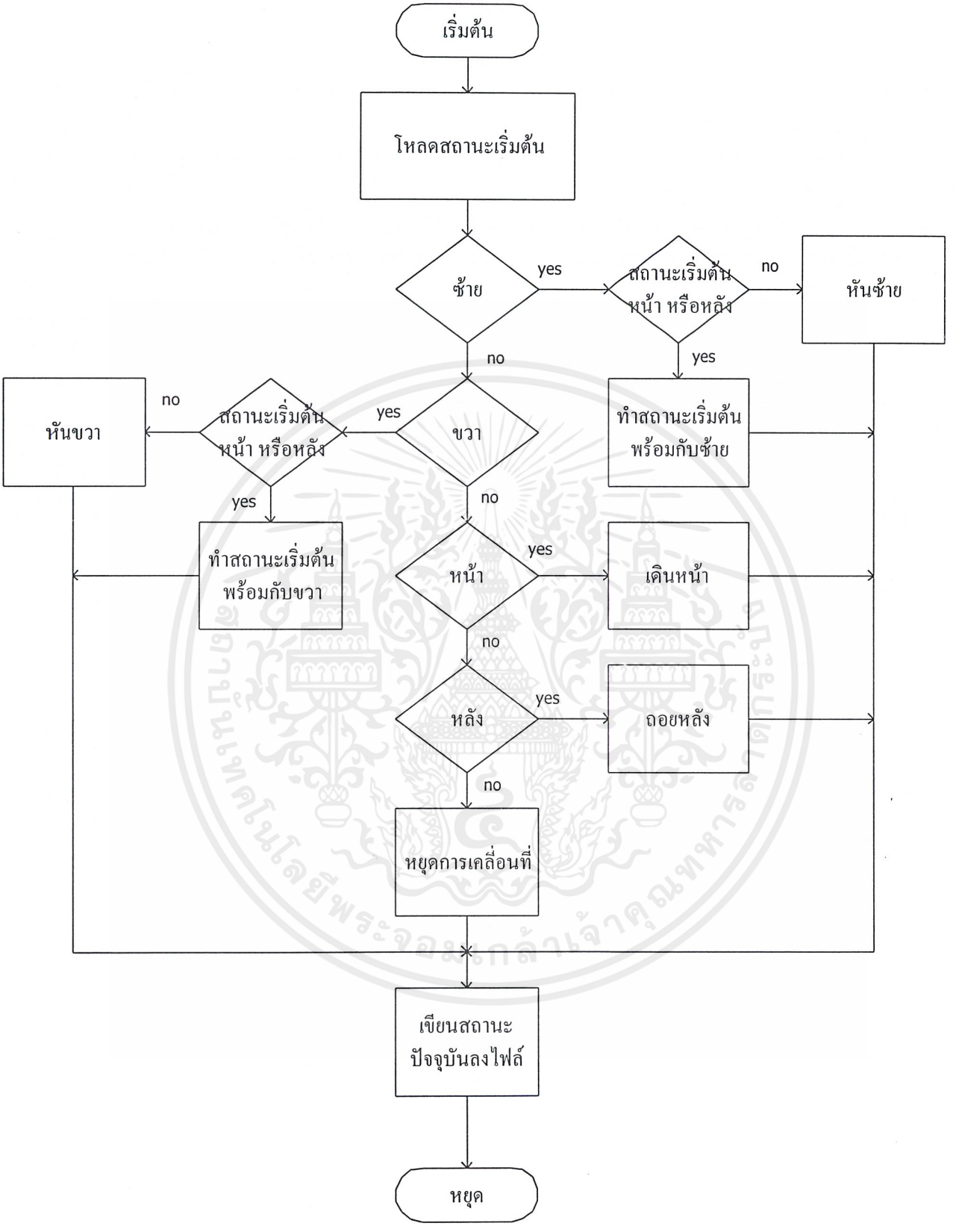
รูปที่ 7 โฟลว์ชาร์ตแสดงขั้นตอนการทำงาน INDEX.C



รูปที่ 8 โฟล์วชาร์ตแสดงการทำงานของโปรแกรม HMM.C



รูปที่ 9 โฟล์วชาร์ตแสดงการทำงานโปรแกรม VITERBI.C



รูปที่ 10 โฟล์วชาร์ตแสดงการทำงานของโปรแกรม CONTROL.C

```
/* PROGRAM ENERGY.C */

 #include<stdio.h>
 #include<stdlib.h>
 #include<math.h>
 #include<io.h>
 #include<time.h>
 FILE *fp,*fp1;
 int a,b,c,i,f,start,final;
 unsigned long int *d,g;
 char ch;
 main()
 {
  if((fp=fopen("voice\\ote\\left11.wav","rb"))==NULL){
    printf("Error");
    exit(1);}
  if((fp1=fopen("voice\\ote\\lgy5.lpc","wb"))==NULL){
    printf("Error");
    exit(1);}
    fseek(fp,0,2);
    a=ftell(fp);
    a=a/100;
    d=calloc(a,sizeof(unsigned long int));
    if(d==NULL){
     printf("memory\n");
     exit(1);}
    rewind(fp);
    for(b=1;b<=a;++b)
    {
     d[b]=0;
     for(i=1;i<=100;++i){
      c=fgetc(fp);
      c=c-127;
      c=pow(c,2);
      d[b]=d[b]+c;}
     if(b==2)
      g=d[b];
     if(b>1)
      if(d[b]>g)
      {
     g=d[b];
     f=b;
      }
    }
    g=g*0.2;
    for(i=f;i>0;i--)
     if(d[i]<g)
     {
      start=i;
      break;
     }
    for(i=f;i>0;i++)
     if(d[i]<g)
     {
      final=i;
      break;
     }
    f=final-start+1;
    rewind(fp);
    fseek(fp,/*44+*/((start-1)*100),0);
    for(b=f;b>0;b--)
```

```
  for(i=1;i<=100;i++)
  {
   c=(int)fgetc(fp);
   fwrite(&c,sizeof(char),1,fp1);
  }
 fclose(fp);
 fclose(fp1);
}
```

```
/* PROGRAM NORMALIZE.C */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<io.h>
#include<time.h>
FILE *fp,*fp1;
int a,b,c,size,low,high,xl,xh;
float step,number,y;
main()
{
 if((fp=fopen("voice\\rad\\lgy5.lpc","rb"))==NULL){
   printf("Erro");
   exit(1);
   }
 if((fp1=fopen("voice\\rad\\nlgy5.lpc","wb"))==NULL){
   printf("Erro");
   exit(1);
   }
 fseek(fp,0,2);
 a=ftell(fp);
 rewind(fp);
 c=(int)fgetc(fp);
 fwrite(&c,sizeof(char),1,fp1);
 size=4000;
 step=(float)(a-1)/(size-1);
 for(b=2;b<=size;++b)
 {
  number=(1+(b-1)*step);
  low=floor(number);
  high=ceil(number);
  if(low==high)
  {
   c=(int)fgetc(fp);
   fwrite(&c,sizeof(char),1,fp1);
  }
  else
  {
   fseek(fp,high-1,0);
   c=(int)fgetc(fp);
   xh=c-127;
   fseek(fp,low-1,0);
   c=(int)fgetc(fp);
   xl=c-127;
   y=xl+(xh-xl)*((number-low)/(high-low));
   y=y+127;
   c=floor(y);
   fwrite(&c,sizeof(char),1,fp1);
  }
 }
 fclose(fp);
 fclose(fp1);
}
```

```
/* PROGRAM LPC.C */


#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<alloc.h>
#include<io.h>
#define FRAME 300
#define SHIFT 100
#define PI 3.141592654
#define p 12
#define q 18
int fr,nf,l,L,i,k,j,m,t;
FILE *infile,*outfile;
char wave_file[25],lpc_file[25];
main()
{
 float pre[FRAME+1],win[FRAME+1],R[p+1],r[p][p+1],a[p],factor,temp;
 float alp[q+1],weig[q+1],cep[q+1];
 double gain,g;
 int s[FRAME+1],pivot;
 printf("Enter wave file name : ");
 gets(wave_file);
 if((infile=fopen(wave_file,"rb"))==NULL)
 {
  printf("\nError in open file\n");
  exit(1);
 }
 printf("Enter lpc file name : ");
 gets(lpc_file);
 if((outfile=fopen(lpc_file,"wb"))==NULL)
 {
  printf("\nError in open lpc file\n");
  exit(1);
 }

 /* Find the length of wave file */

 fseek(infile,0L,SEEK_END);
 L=ftell(infile);
 printf("The number of character : %d\n",L);
 L=(L-44L)/100;
 rewind(infile);
 fseek(infile,44L,SEEK_SET);
 for(t=1;t<=L-2;++t)
 {

 /* End */

  /* find preemphasis and windowing */

  if(t==1)
  {
   for(fr=1;fr<=FRAME;++fr)
   {
    s[fr]=(int)fgetc(infile);
    s[0]=s[1];
    pre[fr]=s[fr]-0.9375*s[fr-1];
    win[fr]=pre[fr]*(0.54-0.46*cos(2*PI*(fr-1)/(FRAME-1)));
   }
```

```
}
else
{
 for(fr=1;fr<=(FRAME-SHIFT);++fr)
 {
  s[fr]=s[fr+SHIFT];
  pre[fr]=pre[fr+SHIFT];
  win[fr]=pre[fr]*(0.54-0.46*cos(2*PI*(fr-1)/(FRAME-1)));
 }
 for(fr=(FRAME-SHIFT);fr<=FRAME;++fr)
 {
  s[fr]=(int)fgetc(infile);
  pre[fr]=s[fr]-0.9375*s[fr-1];
  win[fr]=pre[fr]*(0.54-0.46*cos(2*PI*(fr-1)/(FRAME-1)));
 }
}

/* End */

/*find autocorrelation */

for(k=0;k<=p;++k)
{
 R[k]=0;
 for(fr=0;fr<=(FRAME-1-1);++fr)
 {
  R[k]+=win[fr+1]*win[fr+k+1];
 }
}

/* End */

/* Show autocorelation */

for(k=0;k<=p;++k)
printf("autocorelation R[%d] = %f\n",k,R[k]);

/* End */

/* make matrix */

for(i=0;i<=p-1;++i)
{
 m=0;
 for(j=i;j<=p-1;++j)
 {
  r[i][j]=R[m];
  m++;
  r[j][i]=r[i][j];
 }
  r[i][p]=R[i+1];
}

/* End */

/* find lpc coefficient */

for(i=0;i<p;++i)
{
 if(r[i][i]==0.0)
```

```
 {
  pivot=0;

 /* End */

 /* find nect nonzero entry in col i */
 for(j=i+i;j<p;++j)
 if(r[j][i]!=0.0)
 {
  pivot=j;
  break;
 }

 /* End */

 /* if no nonzero entry in col i,system is singular */

 if(pivot==0)
 {
  printf("System is singular\n");
  exit(1);
 }

 /* End */

 /* swap so r[i][i] != 0 */

 for(j=0;j<p+1;++j)
 {
  temp=r[i][j];
  r[i][j]=r[pivot][j];
  r[pivot][j]=temp;
 }
}

 /* End */

 /* make col i,row j>=i+1, zero */

 for(j=i+1;j<p;++j)
 {
  factor=-r[j][i]/r[i][i];
  for(k=i;k<p+1;++k)
   r[j][k]+=factor*r[i][k];
 }
}

 /* End */

/* solve for unknowns */

a[p-1]=r[p-1][p]/r[p-1][p-1];
for(j=p-2;j>=0;--j)
{
 a[j]=r[j][p];
 for(k=j+1;k<p;++k)
  a[j]-=r[j][k]*a[k];
 a[j]/=r[j][j];
}
for(i=1;i<=p;++i)
 alp[i]=a[i-1];
```

```
for(i=1;i<=p;++i)
 printf("\talp[%d] = %f\n",i,alp[i]);
getch();
g=(double)R[0];
for(i=1;i<=p;++i)
 g-=(double)alp[i]*R[i];
g=fabs(g);
gain=sqrt(g);

/* End */

/* Set initial value */

alp[0]=0.0;alp[13]=0.0;alp[14]=0.0;alp[15]=0.0;alp[16]=0.0;
alp[17]=0.0;alp[18]=0.0;

/* End */

/* find cepstrum */

cep[0]=log(gain);
for(i=1;i<=q;++i)
{
 if(i<=p)
  cep[i]=alp[i];
 else
  cep[i]=0;
 for(j=1;j<i-1;++j)
  cep[i]+=((float)j/(float)i)*cep[j]*alp[i-j];
}

/* End */

/* find weight */

for(i=0;i<=q;++i)
 weig[i]=cep[i]*(1+((float)(q/2))*sin(PI*i/(float)q));

/* End */

/* Show cepstral and weighted cepstral */

for(i=0;i<=q;++i)
 printf("cep[%d] = %f\t weig[%d] = %f\n",i,cep[i],i,weig[i]);
getch();

/* End */

/* write parameter to outfile */
for(i=0;i<=q;++i)
fwrite(&weig[i],sizeof(float),1,outfile);
if(ferror(outfile))
{
 printf("error in writing file\n");
 exit(1);
}
/* End */
 }
fclose(infile);
fclose(outfile);
}
```

```
/* PROGRAM MIX.C */


#include<stdio.h>
#include<stdlib.h>
FILE *fpt,*fb;
char infile[25];
int j;
long int l;
char ch,i;
main()
{
 if((fb=fopen("big.lpc","wb"))==NULL)
 {
  printf("error\n");
  exit(1);
 }
 do{
    printf("\nPlease enter name of the file to read : \n");
    gets(infile);
    if((fpt=fopen(infile,"rb"))==NULL)
    {
     printf("Can't open file\n");
     exit(1);
    }
    l=0;
    while(!feof(fpt))
    {
     i=fgetc(fpt);
     l++;
    }
    l=l-1;
    rewind(fpt);
    printf("The number of file = %ld\n",l);
    fseek(fpt,76L,0);
    for(j=1;j<=(l-76);++j)
    {
     i=fgetc(fpt);
     fwrite(&i,sizeof i,1,fb);
    }
    fclose(fpt);
    printf("Do you write again (y/n)");
    ch=getche();
   }while(ch!='n');
   fclose(fb);
}
```

```
/* PROGRAM RANDCODE.C */

#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<conio.h>
#define M 64
#define N 19
FILE *infile,*outfile;
long int l,k;
int h,i,j;
float NEW[N];
main()
{
 if((infile=fopen("big.lpc","rb"))==NULL)
 {
  printf("error in open infile\n");
  exit(1);
 }
 if((outfile=fopen("coderand.lpc","wb"))==NULL)
 {
  printf("Error in open file codebook\n");
  exit(1);
 }

 /* Count number of data */
 while(!feof(infile))
 {
  i=fgetc(infile);
  l++;
 }
 rewind(infile);
 l=l-1;
 j=l/76;

 /* End */

 /* Random codebook from infile and restore value to outfile */

 randomize();
 h=0;
 clrscr();
 do{
    k=random(j)+1;

    /* Show random value */

    printf("%ld ",k);
    getch();

    /* End */

    k=k*76;
    k=k-76;
    fseek(infile,k,0);
    for(i=0;i<=N-1;++i)
    {
     fread(&NEW[i],sizeof(float),1,infile);
     if(ferror(infile))
```

```
        {
         printf("error in read\n");
         exit(1);
        }
        fwrite(&NEW[i],sizeof(float),1,outfile);
        if(ferror(infile))
        {
         printf("error in write\n");
         exit(1);
        }
       }
       h=h+1;
       rewind(infile);
      }while(h<=63);

  /* End */
 fclose(infile);
 fclose(outfile);
}
```

```
/* PROGRAM FINDCODE.C */

#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<conio.h>
#define M 64
#define N 19
FILE *infile,*infile1,*outfile;
unsigned long int loop,loop1,l,number,q;
int i,j,codebooknumber,count[M],compleatcodebook;
float OLD[M][N],INPUTDATA[N],NEW[M][N],DIFFERENCE[N]
      ,Ncenter,Ocenter;
main()
{
 for(j=0;j<=M-1;j++)
  count[j]=0;
 if((infile=fopen("big.lpc","rb"))==NULL)
 {
  printf("error in open infile\n");
  exit(1);
 }
 if((infile1=fopen("coderand.lpc","rb"))==NULL)
 {
  printf("error in open infile\n");
  exit(1);
 }
 if((outfile=fopen("codebook.lpc","wb"))==NULL)
 {
  printf("Error in open file codebook\n");
  exit(1);
 }

 /* Count number of data */

 loop=l=0;
 while(!feof(infile))
 {
  i=fgetc(infile);
  l++;
 }
 rewind(infile);
 l=l-1;
 j=l/76;

 /* End */

 loop1=loop=j;

 /* Load random value to NEW[j][i] */

 for(j=0;j<=M-1;++j)
  for(i=0;i<=N-1;++i)
  {
   fread(&NEW[j][i],sizeof(float),1,infile1);
   q++;
   if(ferror(infile1))
```

```
 {
  printf("error in read\n");
  exit(1);
 }
}

/* End */

do{
   loop=loop1;
   compleatcodebook=0;

   /* Tranfer data from NEW[j][i] to OLD[j][i] */

   for(j=0;j<=M-1;++j)
    for(i=0;i<=N-1;++i)
    {
     OLD[j][i]=NEW[j][i];
     NEW[j][i]=0;
    }

    /* End */

   /* Load data to b[i] */

   rewind(infile);
   do{
      for(i=0;i<=N-1;++i)
      fread(&INPUTDATA[i],sizeof(float),1,infile);

   /* End */

      /* Find center of codebook */

      Ocenter=-1;
      codebooknumber=0;
      for(j=0;j<=M-1;++j)
      {
       Ncenter=0;
       for(i=0;i<=N-1;++i)
       {
        DIFFERENCE[i]=OLD[j][i]-INPUTDATA[i];
        DIFFERENCE[i]=pow(DIFFERENCE[i],2);
        Ncenter+=DIFFERENCE[i];
       }
       Ncenter=sqrt(Ncenter);
       if(j==0)
       {
        codebooknumber=j;
        Ocenter=Ncenter;
       }
       if(Ocenter>Ncenter)
       {
        codebooknumber=j;
        Ocenter=Ncenter;
       }
       if(Ncenter==0)
       {
        codebooknumber=j;
        break;
       }
```

```
            }
            for(i=0;i<=N-1;++i)
            NEW[codebooknumber][i]=NEW[codebooknumber][i]+INPUTDATA[i];
            count[codebooknumber]+=1;
            loop-=1;
           }while(loop!=0);
         for(j=0;j<=M-1;++j)
          for(i=0;i<=N-1;++i)
          {
           if(count[j]==0)
           {
            NEW[j][i]=OLD[j][i];
            continue;
           }
           NEW[j][i]/=count[j];
          }

            /* End */

        /* Find the best center */

        for(j=0;j<=M-1;++j)
         count[j]=0;
        for(j=0;j<=M-1;++j)
        {
         Ncenter=0;
         for(i=0;i<=N-1;++i)
         {
          DIFFERENCE[i]=OLD[j][i]-NEW[j][i];
          DIFFERENCE[i]=pow(DIFFERENCE[i],2);
          Ncenter+=DIFFERENCE[i];
         }
         Ncenter=sqrt(Ncenter);
         if(Ncenter<0.0001)
          compleatcodebook++;
        }
        number++;
       }while(compleatcodebook<=63);

        /* End */

/* Write codebook to outfile */

for(j=0;j<=M-1;++j)
{
 for(i=0;i<=N-1;++i)
 {
  fwrite(&NEW[j][i],sizeof(float),1,outfile);

/* End */

  /* Show value of codebook */

  printf("%d %f \n",j,NEW[j][i]);
 }
 getch();
 clrscr();
  /* End */
} fclose(infile);
fclose(outfile);
}
```

```
/* PROGRAM INDEX.C */


#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#define M 64
#define N 19
FILE *infile,*infile1,*outfile;
int i,j,codebooknumber,observation;
unsigned long int L;
float CODEBOOK[M][N],INPUTDATA[N],DIFFERENCE[N],Ocenter,Ncenter;
main()
{
 if((infile=fopen("back.lpc","rb"))==NULL)
 {
  printf("Error in open back.lpc\n");
  exit(1);
 }
 if((infile1=fopen("codebook.lpc","rb"))==NULL)
 {
  printf("Error in open codebook.lpc\n");
  exit(1);
 }
 if((outfile=fopen("observa.mod","wb"))==NULL)
 {
  printf("Error in open back.model\n");
  exit(1);
 }

 /* Count number of data */

 fseek(infile,0,2);
 L=ftell(infile);
 L/=64;
 rewind(infile);

 /* End */

 /* Tranfer codebook from file to CODEBOOK[j][i] */

 for(j=0;j<=M-1;++j)
  for(i=0;i<=N-1;++i)
   fread(&CODEBOOK[j][i],sizeof(float),1,infile1);

 /* End */

 /* Find observation and write to outfile */

 do{
    for(i=0;i<=N-1;++i)
     fread(&INPUTDATA[i],sizeof(float),1,infile);
    Ocenter=-1;
    codebooknumber=0;
    for(j=0;j<=M-1;++j)
    {
     Ncenter=0;
     for(i=0;i<=N-1;++i)
     {
      DIFFERENCE[i]=CODEBOOK[j][i]-INPUTDATA[i];
```

```
        DIFFERENCE[i]=pow(DIFFERENCE[i],2);
        Ncenter+=DIFFERENCE[i];
       }
       Ncenter=sqrt(Ncenter);
       if(j==0)
       {
        codebooknumber=j;
        Ocenter=Ncenter;
       }
       if(Ocenter>Ncenter)
       {
        codebooknumber=j;
        Ocenter=Ncenter;
       }
       if(Ncenter==0)
       {
        codebooknumber=j;
        break;
       }
      }
      observation=codebooknumber;
      fwrite(&observation,sizeof(int),1,outfile);
      L--;
     }while(L!=0);

 /* End */

 fclose(infile);
 fclose(infile1);
 fclose(outfile);
}
```

```
/* PROGRAM HMM.C */

#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<alloc.h>
#define M 64
#define N 6
#define NUM 5
FILE *fp,*fp1,*fp2;
char name[20];
int i,j,k,l,t,T,m,n,code;
long double suma,sumb,al,be,up,down,*alpha,*beta,*SUM,*a,*b,*B;
int pi[N]={1,0,0,0,0,0};
main()
{
 printf("Enter observation file : " );
 gets(name);
 if((fp=fopen(name,"rb"))==NULL){
  printf("Error in open file\n");
  exit(1);}
 fseek(fp,0L,2);
 T=ftell(fp);
 T=T/(NUM*2);
 fclose(fp);
 printf("The number of observation : %d",T);

 a=calloc(N*N,sizeof(long double));
 b=calloc(N*M,sizeof(long double));
 B=calloc(NUM*N*T,sizeof(long double));
 SUM=calloc(NUM,sizeof(long double));
 if(((a)&&(b)&&(B)&&(SUM))==NULL)
 {
  printf("Memmory\n");
  exit(1);
 }
 for(i=0;i<N;++i)
  for(j=0;j<N;++j)
   a[i*N+j]=0;
 a[0]=a[1]=a[7]=a[8]=a[14]=a[15]=a[21]=a[22]=a[28]=a[29]=(long
double)1/2;
 a[35]=(long double)1/1;
 for(i=0;i<N;++i)
  for(j=0;j<M;++j)
   b[(i*M)+j]=(long double)1/64;

 for(m=0;m<50;++m)
 {
  alpha=calloc(2*N,sizeof(long double));
  if((alpha)==NULL)
  {
   printf("Memmory\n");
   exit(1);
  }
  if((fp=fopen(name,"rb"))==NULL){
   printf("Error in open file\n");
   exit(1);}
  if((fp1=fopen("alpha1.lpc","wb"))==NULL){
```

```
   printf("Error in open file\n");
   exit(1);}
  for(n=0;n<NUM;++n)
  {
   for(t=0;t<T;++t)
   {
    fread(&code,sizeof(int),1,fp);
    for(i=0;i<N;++i)
     for(k=0;k<M;++k)
      if(code==k)
     B[(n*N*T)+(i*T)+t]=b[i*M+k];
   }
   /* For */
   suma=0;
   sumb=0;
   SUM[n]=1;
   for(i=0;i<N;++i)
   {
    alpha[i]=(pi[i]*B[(n*N*T)+(i*T)+0]);
    fwrite(&alpha[i],sizeof(long double),1,fp1);
   }
   for(t=0;t<T-1;++t)
   {
    sumb=0;
    for(j=0;j<N;++j)
    {
     suma=0;
     for(i=0;i<N;++i)
      suma=suma+(alpha[(0*N)+i]*a[i*N+j]);
     suma=(suma*B[(n*N*T)+(j*T)+(t+1)]);
     alpha[(1*N)+j]=suma;
     fwrite(&alpha[(1*N)+j],sizeof(long double),1,fp1);
    }
    for(i=0;i<N;++i)
     alpha[(0*N)+i]=alpha[(1*N)+i];
  }
  SUM[n]=0;
  for(i=0;i<N;++i){
   SUM[n]=SUM[n]+alpha[(1*N)+i];
   k++;}
}
fclose(fp);
fclose(fp1);
free(alpha);

if((fp1=fopen("beta.lpc","wb"))==NULL){
printf("Error in open file\n");
exit(1);
}
/* Back */
beta=calloc(2*N,sizeof(long double));
if((beta)==NULL)
{
 printf("Memmory\n");
 exit(1);
}
for(n=0;n<NUM;++n)
{
 for(i=0;i<N;++i)
 {
  beta[(1*N)+i]=1;
```

```
      fwrite(&beta[(1*N)+i],sizeof(long double),1,fp1);
     }
     for(t=T-2;t>=0;--t)
     {
      for(i=0;i<N;++i)
      {
       suma=0;
       for(j=0;j<N;++j)
        suma=suma+(a[i*N+j]*B[(n*N*T)+(j*T)+(t+1)]*beta[(1*N)+j]);
       beta[(0*N)+i]=suma;
       fwrite(&beta[(0*N)+i],sizeof(long double),1,fp1);
      }
      for(i=0;i<N;++i)
       beta[(1*N)+i]=beta[(0*N)+i];
     }
    }
  fclose(fp1);
  if((fp1=fopen("beta.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);
   }
  if((fp2=fopen("beta1.lpc","wb"))==NULL){
   printf("Error in open file\n");
   exit(1);
   }
  for(n=0;n<NUM;++n)
   for(t=T-1;t>=0;--t)
    for(i=0;i<N;++i)
    {
     rewind(fp1);
     fseek(fp1,((n*N*T*10)+(t*N*10)+(i*10)),0);
     fread(&be,sizeof(long double),1,fp1);
     fwrite(&be,sizeof(long double),1,fp2);
    }
  fclose(fp1);
  fclose(fp2);
  free(beta);

  if((fp=fopen(name,"rb"))==NULL){
    printf("Error in open file\n");
    exit(1);
   }
  if((fp1=fopen("alpha1.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);
   }
  if((fp2=fopen("beta1.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);
   }
  /* Find a */
  for(i=0;i<N;++i)
   for(j=0;j<N;++j)
   {
    up=0;
    for(n=0;n<NUM;++n)
    {
     suma=0;
     sumb=0;
     for(t=0;t<T-1;++t)
     {
```

```
      rewind(fp1);
      rewind(fp2);
      fseek(fp1,(n*N*T*10)+(t*N*10)+(i*10),0);
      fseek(fp2,(n*N*T*10)+((t+1)*N*10)+(j*10),0);
      fread(&al,sizeof(long double),1,fp1);
      fread(&be,sizeof(long double),1,fp2);
      suma=(al*a[i*N+j]*B[(n*N*T)+(j*T)+(t+1)]*be);
      sumb=sumb+suma;
     }
     up=up+sumb/SUM[n];
    }
    down=0;
    for(n=0;n<NUM;++n)
    {
     suma=0;
     sumb=0;
     for(t=0;t<T-1;++t)
      for(k=0;k<N;++k)
      {
       rewind(fp1);
       rewind(fp2);
       fseek(fp1,(n*N*T*10)+(t*N*10)+(i*10),0);
       fseek(fp2,(n*N*T*10)+((t+1)*N*10)+(k*10),0);
       fread(&al,sizeof(long double),1,fp1);
       fread(&be,sizeof(long double),1,fp2);
       suma=(al*a[i*N+k]*B[(n*N*T)+(k*T)+(t+1)]*be);
       sumb=sumb+suma;
      }
     down=down+sumb/SUM[n];
    }
    a[i*N+j]=up/down;
   }
/* Find b */
for(i=0;i<N;++i)
 for(j=0;j<M;++j)
 {
  up=0;
  rewind(fp);
  for(n=0;n<NUM;++n)
  {
   suma=0;
   sumb=0;
   for(t=0;t<T;++t)
   {
    fread(&code,sizeof(int),1,fp);
    if(code==j)
     l=1;
    else
     l=0;
    rewind(fp1);
    rewind(fp2);
    fseek(fp1,(n*N*T*10)+(t*N*10)+(i*10),0);
    fseek(fp2,(n*N*T*10)+(t*N*10)+(i*10),0);
    fread(&al,sizeof(long double),1,fp1);
    fread(&be,sizeof(long double),1,fp2);
    suma=(al*be*l);
    sumb=sumb+suma;
   }
   up=up+sumb/SUM[n];
  }
  down=0;
```

```
    for(n=0;n<NUM;++n)
    {
     suma=0;
     sumb=0;
     for(t=0;t<T;++t)
     {
      rewind(fp1);
      rewind(fp2);
      fseek(fp1,(n*N*T*10)+(t*N*10)+(i*10),0);
      fseek(fp2,(n*N*T*10)+(t*N*10)+(i*10),0);
      fread(&al,sizeof(long double),1,fp1);
      fread(&be,sizeof(long double),1,fp2);
      suma=(al*be);
      sumb=sumb+suma;
     }
     down=down+sumb/SUM[n];
    }
    b[i*M+j]=up/down;
   }
   fclose(fp);
   fclose(fp1);
   fclose(fp2);


   if((fp=fopen("voice\\aijf1.lpc","wb"))==NULL){
   printf("Error in open file\n");
   exit(1);
   }
   if((fp1=fopen("voice\\bjkf1.lpc","wb"))==NULL){
    printf("Error in open file\n");
    exit(1);
    }
   for(i=0;i<N;++i)
    for(j=0;j<N;++j)
     fwrite(&a[i*N+j],sizeof(long double),1,fp);
   for(j=0;j<N;++j)
    for(k=0;k<M;++k)
     fwrite(&b[j*M+k],sizeof(long double),1,fp1);
   fclose(fp);
   fclose(fp1);
   printf("\007");
   printf("\nTURN %d ",m+1);
   /*getch();*/
  }

  free(a);
  free(b);
  free(B);
}
```

```
/* PROGRAM VITERBI.C */

#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<alloc.h>
#define M 64
#define N 6
FILE *fp,*fp1,*fp2;
char name[15];
int i,j,k,l,t,T,code,delstatemax,deltastatemax,numprobmax,*q,*phi;
long double *delta,*a,*b,*B,*totalprob,delmax,deltamax,probmax;
int pi[N]={1,0,0,0,0,0};
main()
{
 printf("Enter observation file : " );
 gets(name);
 if((fp=fopen(name,"rb"))==NULL){
  printf("Error in open file\n");
  exit(1);}
 fseek(fp,0L,2);
 T=ftell(fp);
 T=T/2;
 printf("The number of observation : %d",T);

 q=calloc(T,sizeof(int));
 delta=calloc(N*T,sizeof(long double));
 phi=calloc(N*T,sizeof(int));
 a=calloc(N*N,sizeof(long double));
 b=calloc(N*M,sizeof(long double));
 B=calloc(N*T,sizeof(long double));
 totalprob=calloc(5,sizeof(long double));
 if(((q)&&(delta)&&(phi)&&(a)&&(b)&&(B)&&(totalprob))==NULL)
 {
  printf("Memmory\n");
  exit(1);
 }

 for(l=0;l<5;++l)
 {
  if(l==0)
  {
   if((fp1=fopen("sound10\\aijf10.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);}
   if((fp2=fopen("sound10\\bjkf10.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);}
  }
  if(l==1)
  {
   if((fp1=fopen("sound10\\aijb10.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);}
   if((fp2=fopen("sound10\\bjkb10.lpc","rb"))==NULL){
   printf("Error in open file\n");
   exit(1);}
  }
```

```
if(l==2)
{
 if((fp1=fopen("sound10\\aijl10.lpc","rb"))==NULL){
 printf("Error in open file\n");
 exit(1);}
 if((fp2=fopen("sound10\\bjkl10.lpc","rb"))==NULL){
 printf("Error in open file\n");
 exit(1);}
}
if(l==3)
{
 if((fp1=fopen("sound10\\aijr10.lpc","rb"))==NULL){
 printf("Error in open file\n");
 exit(1);}
 if((fp2=fopen("sound10\\bjkr10.lpc","rb"))==NULL){
 printf("Error in open file\n");
 exit(1);}
}
if(l==4)
{
 if((fp1=fopen("sound10\\aijs10.lpc","rb"))==NULL){
 printf("Error in open file\n");
 exit(1);}
 if((fp2=fopen("sound10\\bjks10.lpc","rb"))==NULL){
 printf("Error in open file\n");
 exit(1);}
}
for(i=0;i<N;++i)
 for(j=0;j<N;++j)
  fread(&a[i*N+j],sizeof(long double),1,fp1);
for(j=0;j<N;++j)
 for(k=0;k<M;++k)
  fread(&b[j*M+k],sizeof(long double),1,fp2);
fclose(fp1);
fclose(fp2);
rewind(fp);
for(t=0;t<T;++t)
{
 fread(&code,sizeof(int),1,fp);
 for(i=0;i<N;++i)
  for(k=0;k<M;++k)
   if(code==k)
    B[(i*T)+t]=b[i*M+k];
}

/* Viterbi */
for(i=0;i<N;++i)
{
 delta[i]=pi[i]*B[(i*T)+0];
 phi[i]=0;
 ++j;
}
for(t=1;t<T;++t)
 for(j=0;j<N;++j)
 {
  delmax=delta[((t-1)*N)+0]*a[0*N+j]*B[(j*T)+t];
  delstatemax=0;
  for(i=1;i<N;++i)
   if((delta[((t-1)*N)+i]*a[i*N+j]*B[(j*T)+t])>=delmax)
   {
    delmax=delta[((t-1)*N)+i]*a[i*N+j]*B[(j*T)+t];
```

```
          delstatemax=i;
         }
        delta[(t*N)+j]=delmax;
        phi[(t*N)+j]=delstatemax;
       }
     deltamax=delta[((T-1)*N)+0];
     deltastatemax=0;
     for(i=1;i<N;++i)
      if((delta[((T-1)*N)+i])>=deltamax)
      {
       deltamax=delta[((T-1)*N)+i];
       deltastatemax=i;
      }
     q[T-1]=deltastatemax;
     for(t=T-2;t>=0;--t)
     {
      q[t]=phi[((t+1)*N)+(q[t+1])];
      i++;
     }
     /* Total prob */
     totalprob[l]=pi[(q[0])]*B[((q[0]*T)+0)];
     for(t=1;t<T;++t)
      totalprob[l]=totalprob[l]*(a[((q[t-1])*N)+(q[t])]*B[((q
[t])*T)+t]);
   }
   probmax=totalprob[0];
   numprobmax=0;
   for(i=1;i<5;++i)
    if((totalprob[i])>=probmax)
    {
     probmax=totalprob[i];
     numprobmax=i;
    }
   if(numprobmax==0)
    printf("The unknow sound is FORWARD\n");
   if(numprobmax==1)
    printf("The unknow sound is BACKWARD\n");
   if(numprobmax==2)
    printf("The unknow sound is LEFT\n");
   if(numprobmax==3)
    printf("The unknow sound is RIGHT\n");
   if(numprobmax==4)
    printf("The unknow sound is STOP\n");
   fclose(fp);
   if((fp=fopen("result.lpc","wb"))==NULL){
    printf("Error in open file\n");
    exit(1);}
   fwrite(&numprobmax,sizeof(int),1,fp);
   fclose(fp);
}
```

```
/* PROGRAM CONTROL.C */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
int before,result;
FILE *fp1,*fp2;
main()
{
 if((fp1=fopen("before.lpc","rb"))==NULL){
  printf("Error in open file\n");
  exit(1);}
 if((fp2=fopen("result.lpc","rb"))==NULL){
  printf("Error in open file\n");
  exit(1);}
 fread(&before,sizeof(int),1,fp1);
 fread(&result,sizeof(int),1,fp2);
 fclose(fp1);
 fclose(fp2);
 if((fp1=fopen("before.lpc","wb"))==NULL){
  printf("Error in open file\n");
  exit(1);}
 if(result==2)
 {
  if(before==0)
  {
   outport(0x378,0x00);
   outport(0x378,0x05);
   fwrite(&result,sizeof(int),1,fp1);
  }
  if(before==1)
  {
   outport(0x378,0x00);
   outport(0x378,0x06);
   fwrite(&result,sizeof(int),1,fp1);
  }
  else if((before==2)||(before==3)||(before==4))
  {
   outport(0x378,0x00);
   outport(0x378,0x04);
   fwrite(&result,sizeof(int),1,fp1);
  }

 }
 if(result==3)
 {
  if(before==0)
  {
   outport(0x378,0x00);
   outport(0x378,0x09);
   fwrite(&result,sizeof(int),1,fp1);
  }
  if(before==1)
  {
   outport(0x378,0x00);
   outport(0x378,0x0a);
   fwrite(&result,sizeof(int),1,fp1);
  }
  else if((before==2)||(before==3)||(before==4))
```

```
    {
     outport(0x378,0x00);
     outport(0x378,0x08);
     fwrite(&result,sizeof(int),1,fp1);
    }
   }
   if(result==0)
   {
    outport(0x378,0x00);
    outport(0x378,0x01);
    fwrite(&result,sizeof(int),1,fp1);
   }
   if(result==1)
   {
    outport(0x378,0x00);
    outport(0x378,0x02);
    fwrite(&result,sizeof(int),1,fp1);
   }
   if(result==4)
   {
    outport(0x378,0x00);
    fwrite(&result,sizeof(int),1,fp1);
   }
   fclose(fp1);
}
```

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ สำเร็จลงได้ด้วยความช่วยเหลือของหลาย ๆ ฝ่าย ที่ได้ให้คำแนะนำและรับฟังปัญหา และที่สำคัญที่สุดคือกำลังใจที่พร้อมจะสู้กับปัญหา

ขอขอบพระคุณอาจารย์กฤษณ์ วงรุจิระ และ ดร.สุทธิชัย นพนาคีพงษ์ สำหรับคำแนะนำ และความเป็นห่วงเป็นใยในตัวลูกศิษย์

ขอขอบคุณพี่สุริยัณห์ สาระมูล สำหรับเครื่องคอมพิวเตอร์ที่ใช้ในการทำงานในครั้งนี้ พี่อาณัติ นิลขาว สำหรับวงจรอินเตอร์เฟสผ่านทางพอร์ตขนาน ที่สำคัญพี่พงศ์สิริ จุฑาจรัส และพี่ขวัญชัย คงสุข ที่ให้คำปรึกษาในเรื่องของทฤษฎีที่ใช้ในการเขียนโปรแกรมจดจำเสียงพูดในครั้งนี้ และเพื่อน ๆ ในห้องที่คอยถามเสมอว่า โปรเจคทำถึงไหนแล้ว ขอบคุณจริง ๆ

สุดท้ายเลยคงเป็นพ่อ แม่ พี่ น้อง ที่คอยรับฟังปัญหาที่ระบายให้ฟัง

หนังสืออ้างอิง

[1]. Lawrene Rabiner and Biing-Hwang Jung , Fundamentals of Speech Recognition , New Jersey : Prentic Hall , 1993

[2]. มนตรี พจนารถลาวัณย์ , การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี , กรุงเทพมหานคร : ซีเอ็ดยูเคชั่น , 2521