การรู้จำตัวพิมพ์อักขระไทยในประโยค

A Recognition Method of Multi font Thai Characters Printed in Sentences

พิพัฒน์ หิรัญว์ณิชชากร *
Pipat Hiranvanichakorn, Ph.D.
สมยศ จินดาพล **
Somyod Jindapol

บทคัดย่อ

บทความนี้ เสนอผลงานวิจัยเพื่อรู้จำอักขระไทยที่ถูกพิมพ์ในเอกสาร การกระจายของจุดดำตามแนวนอน ถูกใช้ในการตัดแบ่งบรรทัดออกจากภาพเอกสาร และแบ่งสระ วรรณยุกต์ออกจากพยัญชนะ และการกระจายของจุดดำในแนวตั้งและการหาเส้นแสดงขอบของอักขระ ถูกใช้เพื่อแยกตัวอักขระจากประโยค สำหรับการแยกตัวอักขระที่ซ้อนทับกันนั้น ทำได้โดยการกวาดตรวจส่วนที่ยื่นออกไปจากบรรทัดตามแนวตั้ง และการกวาดตรวจตัวอักขระที่อยู่เหนือบรรทัดตามแนวนอน แนวของการตัดกันของการกวาดตรวจนี้ถูกใช้เพื่อแยกตัวอักขระทั้งสองออกจากกัน หลังจากนั้นเส้นแสดงขอบของตัวอักขระถูกตัดแบ่งเป็นส่วนโค้งนูนและส่วนโค้งเว้า และถูกนำไปใช้ในการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งเพื่อรู้จำตัวอักขระ ผลของการรู้จำตัวอักขระพิมพ์แยก 3 รูปแบบจำนวน 1,020 ตัว ได้ผลการรู้จำเป็น 94.5% และการรู้จำตัวอักขระพิมพ์ในประโยค 2 รูปแบบ รูปแบบละ 736 ตัว ได้ผลของการรู้จำเป็น 95.2 % และ 88.9 %

* รองศาสตราจารย์ คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์
** นักศึกษาปริญญาโท คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์

Abstract

This article presents the Thai characters recognition research. In this research, the horizontal distribution of dots is used to separate lines, as well as vowels and alphabets. The vertical distribution of dots is used to separate sentences. The separations of multi-level characters are accomplished by using the intersection of the horizontal and vertical scan of the over-edges. Each chafacter edges are divided into convex and concave curves, which are then deployed in the Dynamic Programming comparison to recognize each character. This approach reported a 94.5% result for the 3 pattersn, 1,020 separated printed characters and 95.2% and 88.9% results for 2 patterns printed sentences, each with 736 characters.

1. บทนำ

ปัจจุบันคอมพิวเตอร์ได้เข้ามามีบทบาทในประเทศไทยมากขึ้น จนกระทั่งเครื่องคอมพิวเตอร์เกือบจะกลายมาเป็นเครื่องมือเครื่องใช้ประจำวันในหน่วยงานต่าง ๆ มากมาย สาเหตุเนื่องมาจากปริมาณงานในหน่วยงาน รวมทั้งข้อมูลและข่าวสารที่ทวีมากขึ้น ทำให้ต้องใช้เวลาในการปฏิบัติงาน และค่าใช้จ่ายเกี่ยวกับบุคลากรสูงขึ้น แต่ในขณะเดียวกันอุปกรณ์คอมพิวเตอร์กลับมีราคาลดต่ำลง อีกทั้งยังมีโปรแกรมสำเร็จรูปที่ถูกพัฒนาขึ้นมาอย่างพร้อมสมบูรณ์ที่จะให้ประมวลผลได้ทันทีตามความต้องการของผู้ใช้ นอกจากนี้แล้ว การนำคอมพิวเตอร์เข้ามาใช้งานยังช่วยให้การประมวลผลข้อมูลทำได้อย่างรวดเร็ว การค้นหาหรือแก้ไขเปลี่ยนแปลงข้อมูลเป็นไปได้โดยสะดวก ทำให้สามารถนำข้อมูลมาใช้ได้ทันต่อเวลา

จากการที่เครื่องคอมพิวเตอร์ ได้เข้ามามีบทบาทในชีวิตประจำวันมากขึ้นนี้ ทำให้พัฒนาการทางคอมพิวเตอร์มีแนวโน้มที่จะอำนวยความสะดวกให้กับผู้ใช้ ในด้านการติดต่อสื่อสารกับเครื่องคอมพิวเตอร์มากขึ้น ไม่ว่าจะเป็นการป้อนข้อมูลเข้า หรือการนำผลลัพธ์ออกจากเครื่องคอมพิวเตอร์ให้มีความใกล้เคียงกับประสาทสัมผัสของมนุษย์ พัฒนาการนี้รวมไปถึงการพัฒนาให้เครื่องคอมพิวเตอร์สามารถรู้จำและสร้างเสียงพูดได้ สามารถอ่านและรู้จำตัวอักขระได้

เนื่องจากการรู้จำอักขระไทยด้วยคอมพิวเตอร์จะก่อให้เกิดผลประโยชน์เชิงพาณิชย์มากมาย จึงทำให้มีการวิจัยเพื่อจะรู้จำอักขระไทยด้วยคอมพิวเตอร์อย่างแพร่หลาย[1]-[9] ส่วนใหญ่มักจะเป็นการวิจัยขั้นต้น เพื่อ

รู้จำตัวอักขระในลักษณะตัวพิมพ์อักขระตัวเดียว ซึ่งในการนำไปใช้งานจริงนั้นจะต้องมีการพัฒนาระบบรู้จำให้สามารถตัดแบ่งอักขระแต่ละตัวออกจากประโยคหรือคำ แล้วนำอักขระที่ได้ไปผ่านขั้นตอนของการรู้จำ

การตัดแบ่งอักขระไทยจากประโยค จะยุ่งยากกว่าการตัดแบ่งอักขระภาษาอังกฤษ จีน ญี่ปุ่น เนื่องจากประโยคของตัวอักขระไทยประกอบด้วยพยัญชนะ สระ และวรรณยุกต์ ที่มีแนวระดับการเขียนที่แตกต่างกันถึง 4 ระดับ และอาจจะมีตัวอักขระที่มีการเหลื่อมล้ำหรือซ้อนทับกัน ทั้งในระดับเดียวกันและในระดับต่างกัน ดังนั้น จึงมีการวิจัยเพื่อจะทำการตัดแบ่งอักขระไทยที่ถูกพิมพ์ซ้อนทับกัน[9] แต่อย่างไรก็ตามงานวิจัยที่ผ่านมามีขั้นตอนในการทำงานสลับซับซ้อน ในงานวิจัยนี้จึงได้เสนอแนวคิดการรู้จำอักขระไทยจากประโยคโดยการใช้การกระจายของจุดดำตามแนวนอน (Horizontal Pixel Projection) เพื่อทำการตัดแบ่งบรรทัดออกจากภาพเอกสารและแบ่งสระ วรรณยุกต์ ซึ่งอยู่ในระดับบนและล่าง ออกจากตัวพยัญชนะที่อยู่ในบรรทัด และใช้การกระจายของจุดดำในตั้ง (Vertical Pixel Projection) และการหาเส้นแสดงขอบของอักขระ เพื่อแยกตัวอักขระออกจากประโยค สำหรับตัวอักขระที่ซ้อนทับกันนั้น จะทำการ scan ส่วนที่ยื่นออกไปจากบรรทัดของอักขระ ไปตามแนวตั้ง และ scan ส่วนที่อยู่เหนือระดับบรรทัดตามแนวนอน การตัดกันของการ scan ทั้งสองแนวนี้ ใช้เป็นพื้นฐานในการตัดแบ่งตัวอักขระออกจากกัน เมื่อตัดแบ่งอักขระแต่ละตัวแล้ว เส้นแสดงขอบของตัวอักขระจะถูกนำมาใช้ในการหาส่วนนูน และส่วนเว้าของอักขระ จากนั้นลักษณะสำคัญของส่วนนูนและส่วนเว้าจะถูกใช้ในการรู้จำตัวอักขระโดยเทคนิคของการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่ง จากการทดลองการรู้จำอักขระไทยตัวเดียว 3 รูปแบบรวม 1,020 ตัว ได้ผลการรู้จำ 94.51% และการรู้จำอักขระที่อยู่ในประโยคจำนวน 736 ตัว ได้ผล 95.24% และ 88.99% ตามลำดับ

1. อักขระไทย

2.1 อักขระที่ใช้ในการรู้จำ

ปัจจุบันอักขระไทยที่นิยมใช้เป็นพื้นฐานสำหรับการรู้จำจะประกอบด้วย พยัญชนะ 42 ตัว สระ 17 ตัว เครื่องหมายวรรณยุกต์ 4 ตัว เครื่องหมายกำกับเสียง 2 ตัว และอักขระ พิเศษ 2 ตัว

ก. พยัญชนะ 42 ตัว ได้แก่

ก ข ค ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ ท ธ น บ ป ผ ฝ พ ฟ
ภ ม ย ร ล ว ศ ษ ส ห ฬ อ ฮ

ข. สระ 17 ตัว ได้แก่

ะ า ำ ฤ ฦ เ แ โ ใ ไ

ค. เครื่องวรรณยุกต์ 4 ตัว ได้แก่

่ ้ ๊ ๋

ง. เครื่องหมายกำกับเสียง 2 ตัว ได้แก่

่ ้

จ. อักขระพิเศษ 2 ตัว ได้แก่

ฯ ๆ

2.2 ลักษณะสำคัญอย่างหนึ่งของอักขระไทยในประโยค

เนื่องจากกลุ่มอักขระไทยประกอบด้วย พยัญชนะ สระ เครื่องหมายวรรณยุกต์ เครื่องหมายกำกับเสียง และอักขระพิเศษ ซึ่งมีแนวระดับการเขียนที่แตกต่างกัน ทำให้สามารถจัดแบ่งกลุ่มอักขระตามแนวระดับการเขียนออกได้เป็น 4 กลุ่ม ดังนี้

ก. อักขระเขียนแนวระดับบนสุด ได้แก่

ั ์ ็ ่

ข. อักขระเขียนแนวระดับบน ได้แก่

ิ ี ึ ื ่ ้ ั ์ ็ ่ ้

ค. อักขระเขียนแนวระดับฐาน ได้แก่

ก ข ค ฆ ง จ ฉ ช ซ ฌ ฑ ฒ ณ ด ต ถ ท ธ น บ ผ พ ภ ม ย ร ล

ว ศ ษ ส ห อ ฮ ป ฝ ฟ พ ญ ฎ ฏ ฐ ฤ

ะ า เ แ โ ไ ใ ๆ

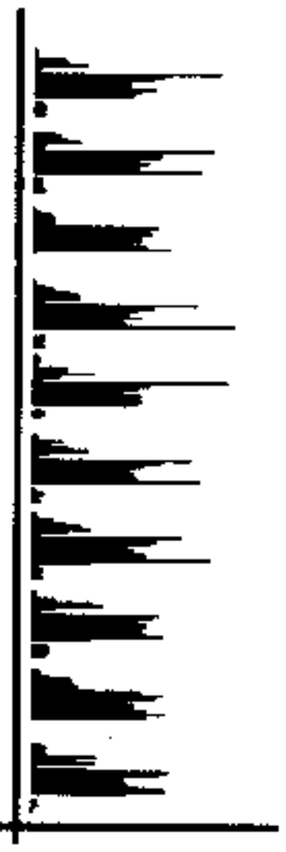
ง. อักขระเขียนแนวระดับล่าง ได้แก่

ุ ู

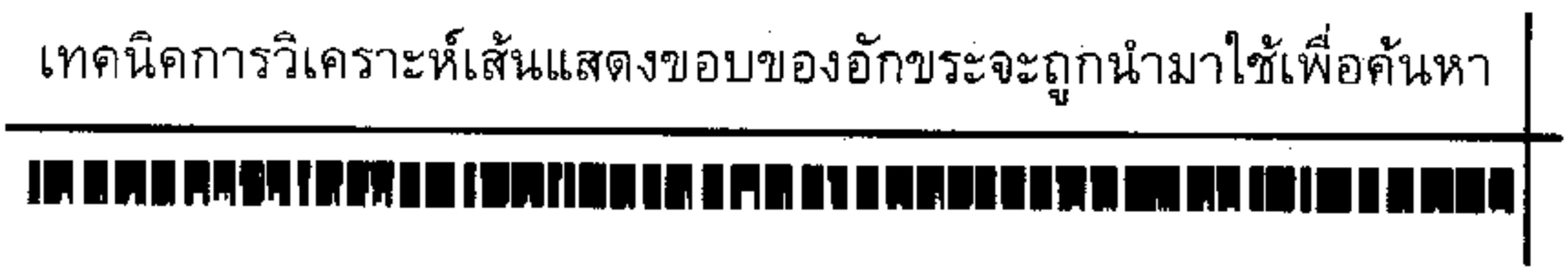
3. การตัดแบ่งอักขระออกจากประโยค

ปกติแล้วภาพเอกสารที่เข้าสู่คอมพิวเตอร์นั้นสามารถแบ่งความเข้มของแสงของภาพออกได้เป็น 2 ระดับ คือ ส่วนของตัวอักษรที่มีสีดำ และส่วนของพื้นกระดาษ (background) ที่มี สีขาว จากรูปที่ 1 จะเห็นได้ว่า เมื่อ scan ภาพเอกสารตามแนวนอน จำนวนจุดดำที่ได้ ณ ตำแหน่งค่า y แต่ละค่าเรียกว่าเป็นการกระจายของจุดดำตามแนวนอน และเมื่อ scan ภาพเอกสารตามแนวตั้ง จำนวนจุดดำที่ได้ ณ ตำแหน่งค่า x แต่ละค่าเรียกว่าเป็นการกระจายของจุดดำตามแนวตั้ง สำหรับรูปประโยคของภาษาอังกฤษ จีน ญี่ปุ่น นั้น โดยการวิเคราะห์การกระจายจุดดำตามแนวนอน จะสามารถตัดประโยคออกจากภาพเอกสารได้ และโดยการวิเคราะห์การกระจายของจุดดำตามแนวตั้งของประโยคที่ตัดมาได้นั้นทำให้สามารถตัดแบ่งตัวอักขระจากประโยคได้

เทคนิคการวิเคราะห์เส้นแสดงขอบของอักขระจะถูกนำมาใช้เพื่อค้นหาลักษณะสำคัญของอักขระแต่ละตัว โดยการใช้รหัสทิศทางแบบลูกโซ่ของฟรีแมน และความแตกต่างของทิศทางของเส้นแสดงขอบของอักขระในการตัดแบ่งเส้นแสดงขอบของอักขระออกเป็นส่วนโค้งย่อยซึ่งประกอบด้วย ส่วนโค้งเว้า และส่วนโค้งนูน จากนั้นก็จะทำการดึงลักษณะสำคัญของส่วนโค้งย่อยแต่ละส่วนโค้ง สำหรับใช้ในการเปรียบเทียบส่วนโค้งหาคู่ส่วนโค้งที่คล้ายกันมากที่สุดระหว่างอักขระที่ต้องการรู้จำกับอักขระต้นแบบ เพื่อใช้เป็นคู่ส่วนโค้งเริ่มต้นในการเปรียบเทียบหาอักขระต้นแบบที่เหมือนหรือคล้ายกับอักขระที่ต้องการรู้จำมากที่สุด ในขั้นตอนการเปรียบเทียบนั้นจะนำเอาเทคนิคการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งมาใช้ ซึ่งเทคนิคการเปรียบเทียบดังกล่าวนี้เป็นเทคนิคที่งานวิจัยในอดีตนำมาประยุกต์ใช้ แล้วได้ผลความ



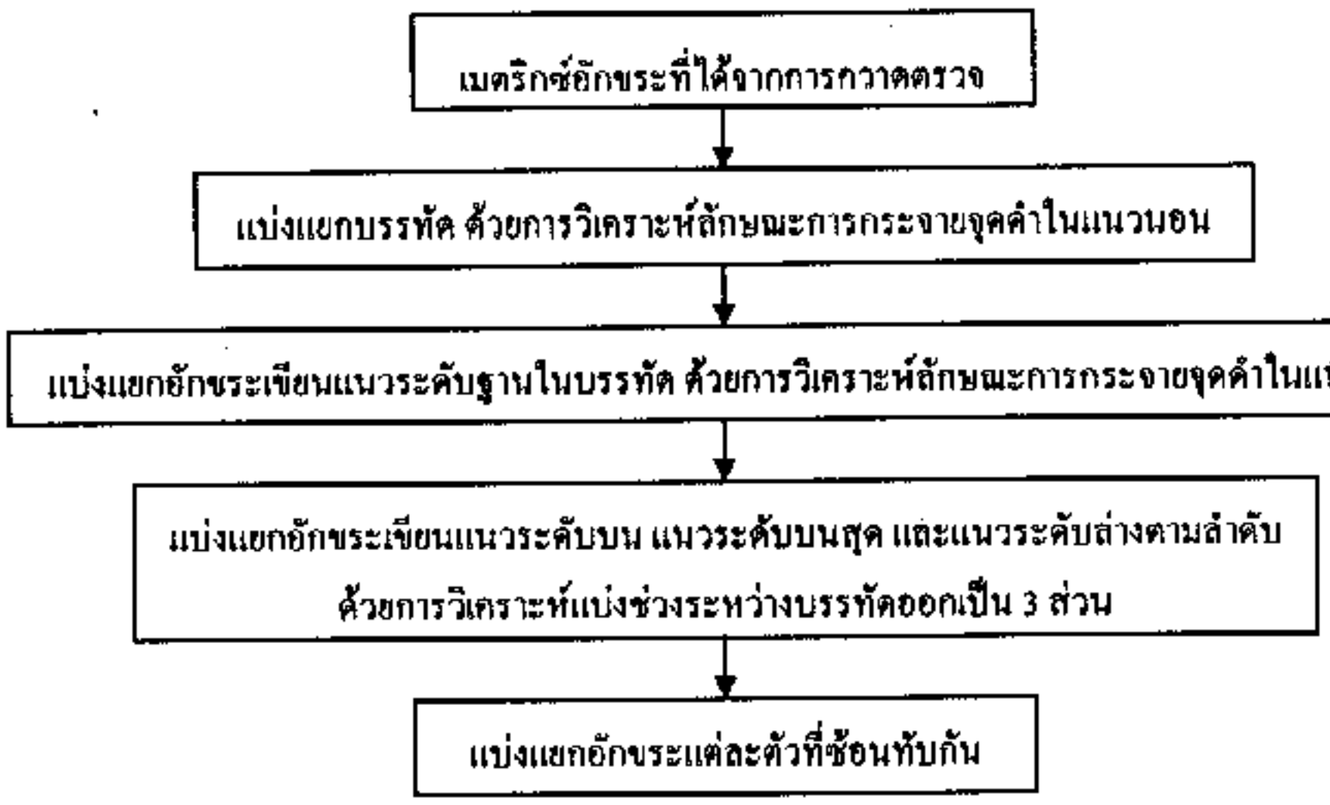
(ก)



(ข)

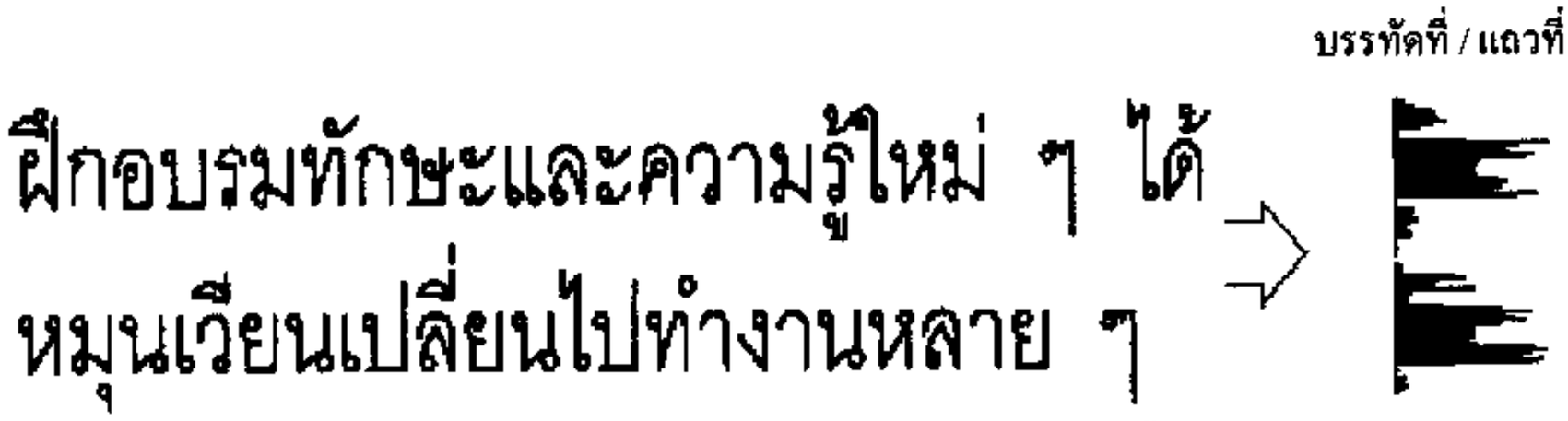
รูปที่ 1 (ก) ภาพแสดงลักษณะการกระจายของจุดดำตามแนวนอน (Horizontal Pixel Projection)
(ข) ภาพแสดงลักษณะการกระจายของจุดดำตามแนวตั้ง (Vertical Pixel Projection)

เนื่องจากประโยคของตัวอักขระไทย ประกอบด้วย พยัญชนะ สระ และวรรณยุกต์ ที่มีแนวระดับการเขียนที่แตกต่างถึง 4 ระดับ และอาจจะมีตัวอักขระที่มีการเหลื่อมล้ำ หรือซ้อนทับกัน ทั้งในระดับเดียวกันและในระดับต่างกัน ดังนั้น การตัดแบ่งตัวอักขระออกจากประโยคภาษาไทย จึงต้องมีการทำงานหลายขั้นตอน แต่อย่างไรก็ตาม เราสามารถใช้การกระจายของจุดดำตามแนวนอนเป็นลักษณะสำคัญ (features) ในการตัดบรรทัดออกจากภาพเอกสาร และแบ่งตัวสระ ตัววรรณยุกต์ ซึ่งอยู่ในระดับบน ระดับบนสุด และระดับล่างออกจากตัวพยัญชนะที่อยู่ในระดับฐาน และยังสามารถใช้การกระจายของจุดดำตามแนวตั้งเพื่อแยกแต่ละพยัญชนะออกจากประโยคได้ ซึ่งขั้นตอนของการตัดแบ่งตัวอักขระออกจากประโยคในงานวิจัยนี้ ได้แสดงในรูปที่ 2 ซึ่งมีรายละเอียดของแต่ละขั้นตอนดังต่อไปนี้



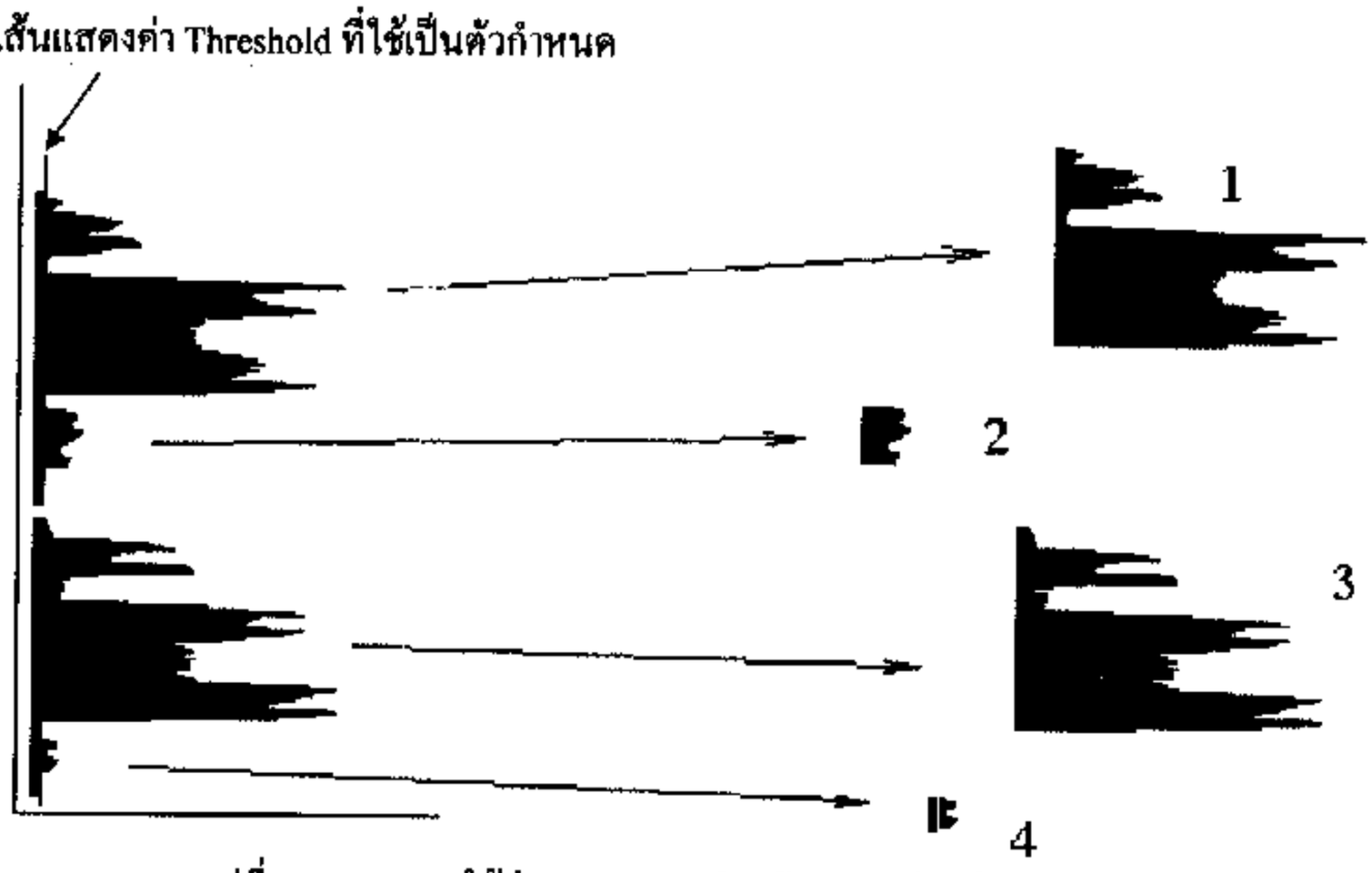
รูปที่ 2 แสดงขั้นตอนของการตัดแบ่งตัวอักขระออกจากประโยค

3.1 การตัดแบ่งตามแนวนอนเพื่อแยกบรรทัด



รูปที่ 3 แสดงกราฟ Horizontal Pixel Projection

จากรูปที่ 3 ซึ่งแสดงการกระจายของจุดดำตามแนวนอน จะเห็นได้ว่าอักขระที่เขียนแนวระดับฐานเป็นกลุ่มอักขระที่ปรากฏอยู่ในคำหรือประโยคมากที่สุด ซึ่งทำให้การกระจายจุดดำของอักขระในแนวนี้มีความหนาแน่นกว่าอักขระที่เขียนในแนวอื่น ดังนั้น ด้วยการใช้ความหนาแน่นของกระจายจุดดำของอักขระในแนวระดับฐาน ทำให้สามารถรู้แนวของแต่ละบรรทัดในเอกสารได้ แต่อย่างไรก็ตาม บรรทัดบางบรรทัดอาจจะสั้น และตัวอักขระบางตัวของระดับฐานยังอาจจะเหลื่อม เข้าไปในแนวของอักขระระดับบนและล่าง จึงทำให้การแยกการกระจายของจุดดำของอักขระในแนวบรรทัดฐานกับอักขระในแนวระดับอื่นไม่ดีพอ ดังนั้น ในการทดลองจึงมีการใช้ threshold ค่าหนึ่งเพื่อช่วยกำจัดส่วนที่เหลื่อมล้ำระหว่างกลุ่มของอักขระในแนวระดับต่าง ๆ โดยไม่กระทบต่อบรรทัดที่สั้น



รูปที่ 4 แสดงการใช้ค่า Threshold สำหรับระบุแบ่งกลุ่มแถว

แต่อย่างไรก็ตาม จากรูปที่ 4 จะเห็นว่ายังมีกลุ่มที่การกระจายของจุดดำของอักขระแนวระดับฐานที่ต่อเนื่องกับอักขระในแนวระดับอื่น ๆ ดังนั้นในที่นี้จึงมีการหาค่าเฉลี่ยความสูงของการกระจายของกลุ่ม (HX) เพื่อช่วยในการแยกกลุ่มดังต่อไปนี้

$$HX = \frac{\sum_{n=1}^{N} n \cdot dy_n}{\sum_{n=1}^{N} dy_n}$$

โดยที่ $n = 1 \sim N$ เป็นค่าความสูงของการกระจาย และ dy_n เป็นจำนวนจุดที่ความสูง n สำหรับตัวอย่างการหาค่าเฉลี่ยของความสูงของการกระจายได้แสดงไว้ในตารางที่ 1 เพื่อความเร็วในการคำนวณ จะไม่มีการหาค่าเฉลี่ยของความสูงสำหรับกลุ่มที่มีความหนา (dy_{max}) แตกต่างกับกลุ่มอื่นอย่างเห็นได้ชัด เพราะกลุ่มนี้จะเป็นกลุ่มของอักขระในแนวระดับบนหรือล่าง เช่น กลุ่ม 2 หรือ 4 ของรูปที่ 4 เป็นต้น

เมื่อคำนวณหาค่าเฉลี่ยของความสูงของการกระจายของกลุ่มได้แล้ว จะนำเอาค่าเฉลี่ยนี้เป็นค่า threshold เพื่อหากลุ่มที่มีความสูงมากกว่าค่า threshold ดังแสดงในรูปที่ 5 ซึ่งจากรูปจะเห็นว่าสามารถหาอักขระในแนวบรรทัดฐานได้เนื่องจากมีความหนา (dy) มากกว่ากลุ่มอื่น

จากข้อมูลกราฟรูปที่ 4 (กลุ่มแถวส่วนที่ 1) สามารถแจกแจงการกระจายของจุดดำได้ดังตารางที่ 1

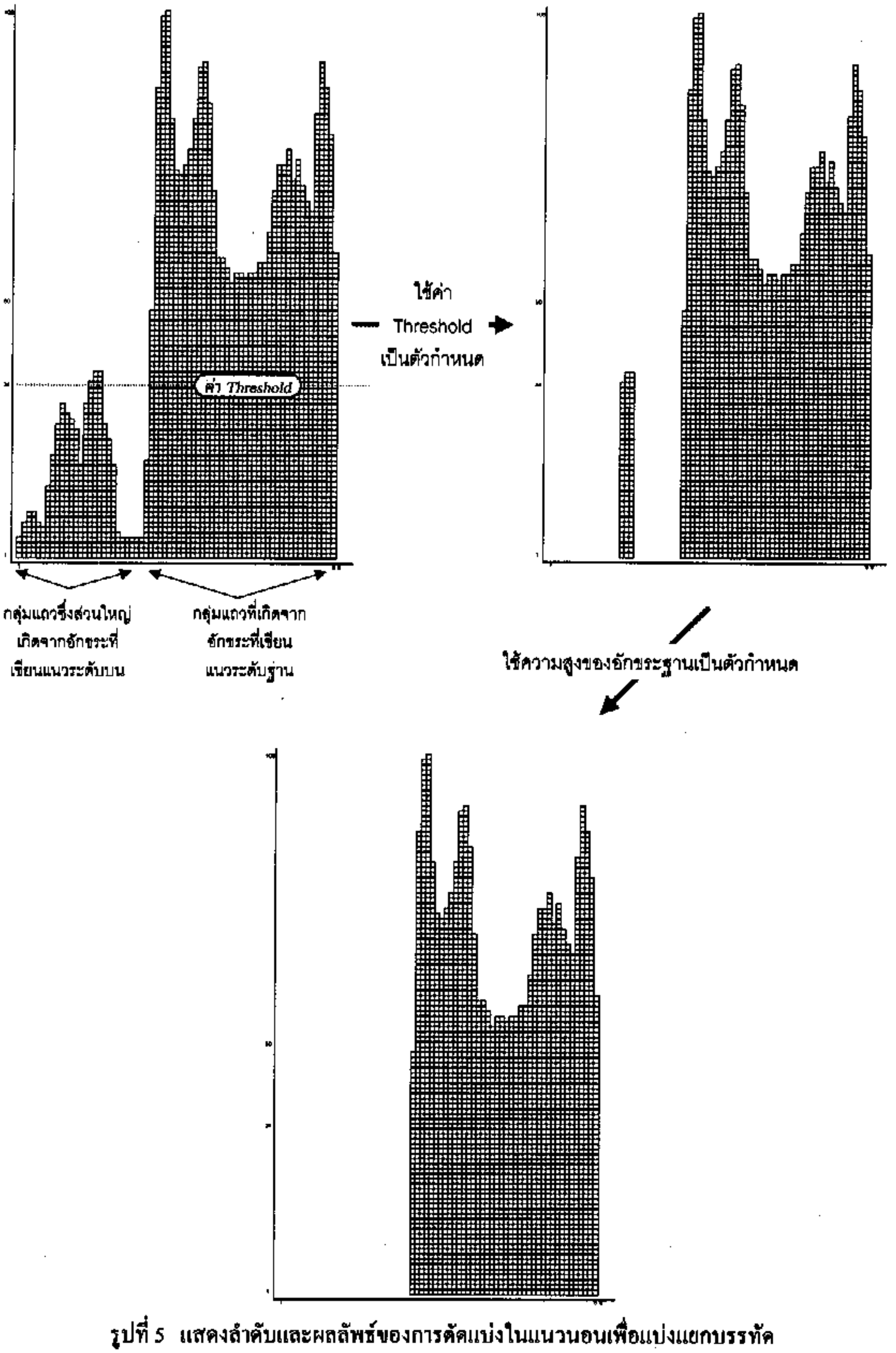
ตารางที่ 1

ความสูงของการกระจาย (n)	จำนวนจุดที่ความสูง n (dy_n)	(dy_n).(n)
0	68	0
1	68	68
2	68	136
3	68	204
4	68	272
5	62	310
6	61	366
7	60	420
8	58	464
9	58	522
10	56	560
11	56	616
12	56	672
13	56	728
14	56	784
15	55	825
16	55	880
17	55	935
18	55	990
19	53	1007
20	52	1040
21	51	1071
22	51	1122
23	51	1173
24	50	1200
25	50	1250
26	49	1274
27	47	1269
28	46	1288
29	45	1305
30	45	1350
31	43	1333
32	43	1376
33	43	1419
34	43	1462
35	42	1470
36	42	1512
37	40	1480
38	40	1520
39	40	1560
40	40	1600
41	40	1640
42	40	1680
43	40	1720
44	40	1760
45	40	1800
46	40	1840
47	40	1880
48	40	1920
49	39	1911
50	39	1950
51	39	1989
52	39	2028

ตารางที่ 1 (ต่อ)

ความสูงของการกระจาย (n)	จำนวนจุดที่ความสูง n (dy_n)	(dy_n).(n)
53	39	2067
54	38	2052
55	37	2035
56	33	1848
57	32	1824
58	30	1740
59	28	1652
60	27	1620
61	27	1647
62	27	1674
63	27	1701
64	26	1664
65	26	1690
66	26	1716
67	26	1742
68	25	1700
69	25	1725
70	24	1680
71	24	1704
72	22	1584
73	21	1533
74	20	1480
75	19	1425
76	18	1368
77	15	1155
78	14	1092
79	14	1106
80	12	960
81	12	972
82	12	984
83	11	913
84	11	924
85	11	935
86	9	774
87	8	696
88	8	704
89	7	623
90	7	630
91	7	637
92	5	460
93	5	465
94	5	470
95	5	475
96	4	384
97	2	194
98	2	196
99	2	198
100	2	200
101	2	202
102	2	204
103	2	206
104	2	208
105	2	210
106	1	106
รวม	3469	119105

ค่าเฉลี่ยจำนวนจุดดำ = 119105 / 3469 = 34.33 ≈ 34

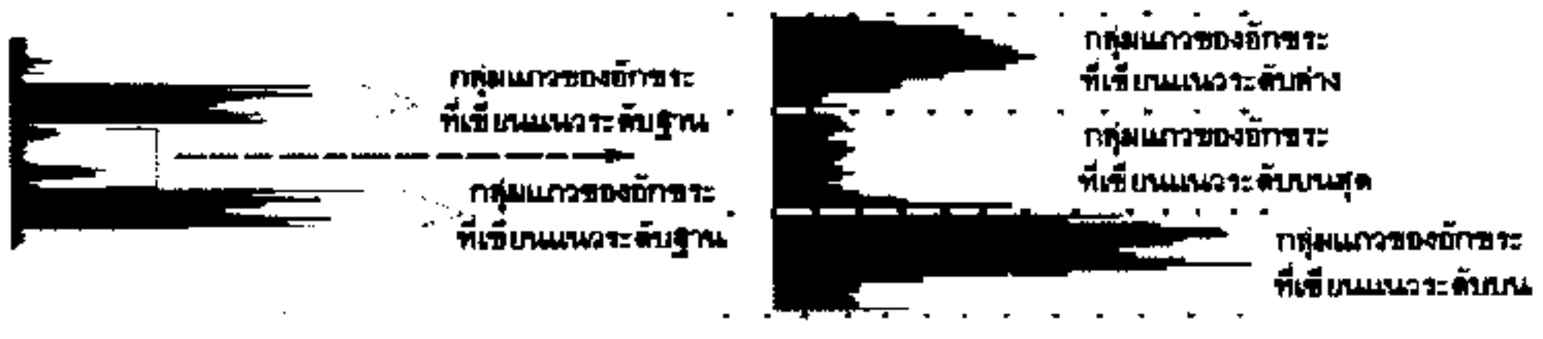


รูปที่ 5 แสดงลำดับและผลลัพธ์ของการตัดแบ่งในแนวนอนเพื่อแบ่งแยกบรรทัด

3.2 การแยกอักขระแนวระดับบนสุด ระดับบนและระดับล่าง

โดยใช้หลักการการตัดแบ่งข้างต้นทำให้สามารถรู้ตำแหน่งของแต่ละบรรทัดได้ แต่จากรูปที่ 6 จะเห็นได้ว่าในบางครั้งอาจจะมีการเหลื่อมล้ำของอักขระที่เขียนในแนวระดับล่างของบรรทัดบน กับอักขระในแนวระดับบนสุดของบรรทัดล่าง ดังนั้น จึงมีความจำเป็นต้องแยกอักขระของแต่ละบรรทัดออกจากกัน ซึ่งจากรูป 6 จะเห็นได้ว่า หากแบ่งช่วงระหว่างบรรทัดที่ติดกันออกเป็น 3 ส่วนเท่า ๆ กัน อักขระที่เขียนในแนวระดับล่างของบรรทัดบนจะอยู่ในช่วงของส่วนที่ 3 ส่วนอักขระที่เขียนในแนวระดับบนสุด และแนวระดับบนจะอยู่ในช่วงของส่วนที่ 2 และ 1 ตามลำดับ ดังนั้น ในการวิจัยจึงได้แบ่งช่วงระหว่างบรรทัดออกเป็น 3 ส่วน เพื่อเป็นกรอบในการตรวจหา (scan) อักขระที่อยู่ในแนวระดับดังกล่าว

(ก)



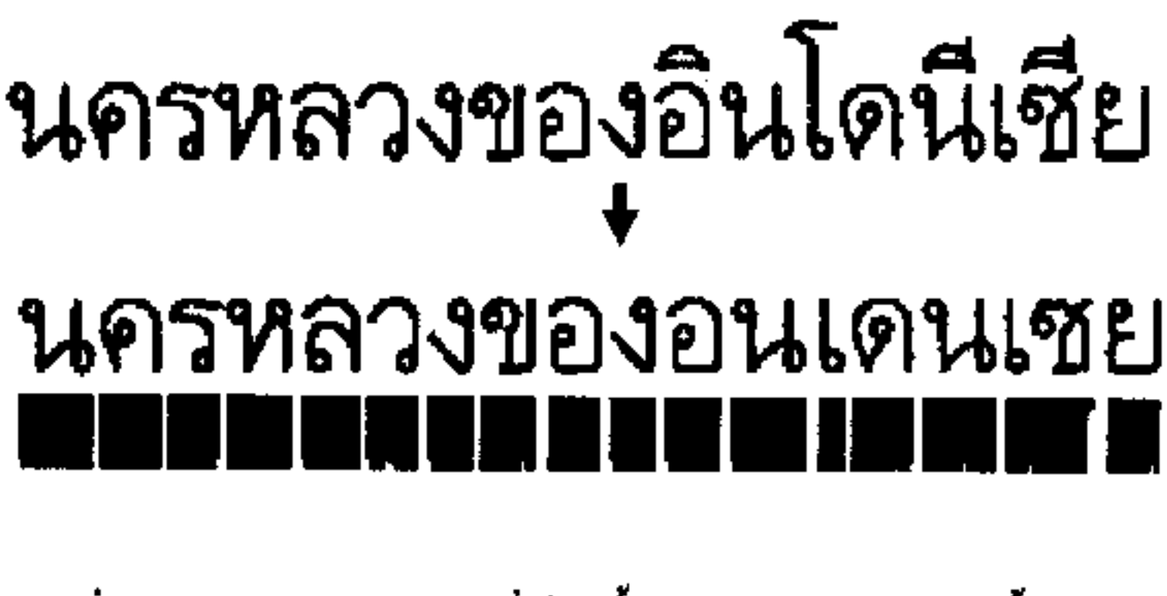
(ข)

รูปที่ 6 การแบ่งเฉลี่ยจำนวนแถวที่อยู่ระหว่างกลุ่มแถวของอักขระฐานทั้งสองออกเป็น 3 ส่วนเท่าๆ กัน

3.3 การตัดแบ่งอักขระในแนวตั้ง

เมื่อได้แนวของแต่ละบรรทัด ก็จะทำการกระจายของจุดดำตามแนวตั้งของแต่ละบรรทัด ดังแสดงในรูปที่ 7 ซึ่งจะทำให้ทราบขอบเขตหรือบล็อกของแต่ละตัวอักษรได้ และเมื่อทำการ scan แต่ละบล็อก เพื่อหาเส้นแสดงขอบ (contour) ของตัวอักขระ ก็จะได้ตัวอักขระแต่ละตัวออกมาได้ ถึงแม้บางบล็อกอาจจะมีอักขระที่เหลื่อมล้ำกันก็ตาม เช่น ตัว เ และ ซ แต่เมื่อตรวจหาเส้นตามขอบของอักขระแล้วก็สามารถแยกตัวอักขระออกจากกันได้

เมื่อได้อักขระในแนวระดับฐานแล้ว ก็จะตรวจที่ด้านบนของบล็อกไปในช่วงที่ 1 ตามที่ได้แบ่งในหัวข้อที่ 3.2 เพื่อหาตัวอักขระที่อยู่ในระดับบน หากมีตัวอักขระก็จะตรวจหาในช่วงที่ 2 เพื่อหาตัวอักขระที่อยู่ในระดับบนสุดต่อไป แต่หากไม่เจอตัวอักขระในช่วงที่ 1 ก็แสดงว่าไม่มีตัวอักขระทั้งแนวระดับบนและบนสุดเลย เมื่อตรวจหาด้านบนของบล็อกแล้วจึงมาทำการตรวจหาตัวอักขระที่เขียนในแนวระดับล่าง ซึ่งอยู่ในช่วงที่ 3 ด้านล่างของบล็อก



รูปที่ 7 กราฟแสดงจำนวนจุดที่เป็นเนื้อของตัวอักขระในแนวตั้ง

3.4 การแยกอักขระที่เขียนทับกัน

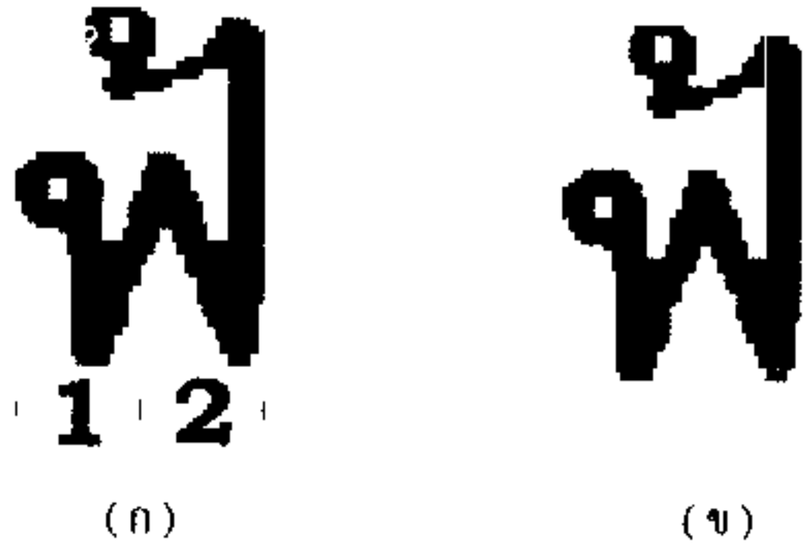
ด้วยการหาเส้นแสดงขอบของตัวอักขระทำให้สามารถแยกตัวอักขระที่การ scan ตามแนวตั้งอยู่ในบล็อกเดียวกันได้ แต่อย่างไรก็ตาม ในการพิมพ์ตัวอักขระไทยอาจจะเกิดการซ้อนทับกัน โดยเฉพาะอักขระแนวระดับฐานที่ส่วนยื่นออกไปในระดับบน เช่น ช ซ ป ฟ ฝ ส ฮ ห กับตัวอักขระที่อยู่ในแนวระดับบน เช่น [illegible] เป็นต้น ในการวิจัยนี้จึงเสนอแนวทางในการแยกอักขระที่ซ้อนทับกันนี้ โดยการหาตำแหน่งที่ซ้อนทับกันดังแสดงในรูปที่ 8 กล่าวคือ ส่วนของอักขระระดับฐานที่ยื่นเข้าไปในแนวระดับบน มักเป็นลักษณะของแนวตั้ง และส่วนของอักขระในแนวระดับบนที่ทับกับอักขระระดับฐานมักเป็นลักษณะของแนวนอน ดังนั้น โดยการ scan ส่วนของอักขระฐานที่ยื่นออกไปจากแนวระดับฐานตามแนวตั้ง และ scan ส่วนของอักขระระดับบน ตามแนวนอน ส่วนที่เกิดจากตัดกันของแนวทั้งสองนี้ พิจารณาได้ว่าเป็นส่วนทับซ้อนกันอักขระสองตัวนี้

(ก)

(ข)

รูปที่ 8 ลักษณะจุดตัดที่ได้จากการลากเส้นตรงต่างแนว ณ ส่วนหางของอักขระทั้งสอง

นอกจากนั้น หากพิจารณาอักขระในระดับฐานที่มีส่วนที่ยื่นออกไปในระดับบน อาจจะแบ่งอักขระเหล่านี้ออกได้ 2 จำพวกใหญ่ ๆ จำพวกแรกคืออักขระที่มีส่วนที่ยื่นออกไปมีลักษณะเป็นลำเหยียดตรง และส่วนที่เกิดการซ้อนทับกันก็เกิดจากส่วนนี้ เช่น ตัวอักขระ ป ฟ ฝ จำพวกที่สองเป็นอักขระที่ส่วนที่ยื่นออกไปนั้นมีลักษณะการยืดออกไปในแนวเฉียง เช่น ตัวอักขระ ช ซ ส ฮ สำหรับอักขระจำนวนแรกนั้น เมื่อมีการทับกันกับตัวอักขระในระดับบน หาก scan ส่วนที่ยื่นออกไปจากแนวของระดับฐานตามแนวตั้งดังแสดงในรูป 9 ก็จะได้ส่วนหางของตัวอักษรอักขระเหล่านี้ ส่วนตัวอักขระในแนวระดับบนหรือบนสุดที่ซ้อนทับกับตัวอักขระเหล่านี้ จะมีส่วนที่ถูกตัดทิ้งไป เนื่องจากส่วนที่ซ้อนทับกันนั้นถูกพิจารณาว่าเป็นเนื้อของหางของอักขระในแนวระดับฐาน นอกจากนั้นในการพิจารณาการแยกอักขระที่ทับกันของอักขระจำพวกนี้ ต้องพิจารณาด้วยว่าเป็นอักขระ ฬ หรือเป็นอักขระที่ซ้อนทับกันด้วย สำหรับตัวอักขระจำพวกที่สองนั้น ส่วนที่ซ้อนทับกันกับตัวอักขระในแนวระดับบนมีไม่มากนัก ดังแสดงในรูปที่ 10 และหางของตัวอักขระจำพวกนี้ไม่เหยียดตรงในแนวตั้ง ดังนั้น หาก scan ตัวอักขระในแนวระดับบนตามแนวนอน และพิจารณาส่วนที่ซ้อนทับกันว่าเป็นเนื้อของตัวอักขระระดับบนก็จะได้ตัวอักขระในแนวระดับบนที่ค่อนข้างจะสมบูรณ์ โดยอักขระในแนวระดับฐานอาจมีส่วนที่ ถูกตัดทิ้งไปบ้าง แต่ก็ยังพอพิจารณาได้ว่าเป็นอักขระใด



(ก) (ข)

รูปที่ 9 (ก) อักขระซ้อนทับที่กำหนดใช้การแบ่งแยกตามแนวตั้ง
(ข) รูปแบบอักขระที่ได้หลังการแบ่งแยกตามแนวตั้ง

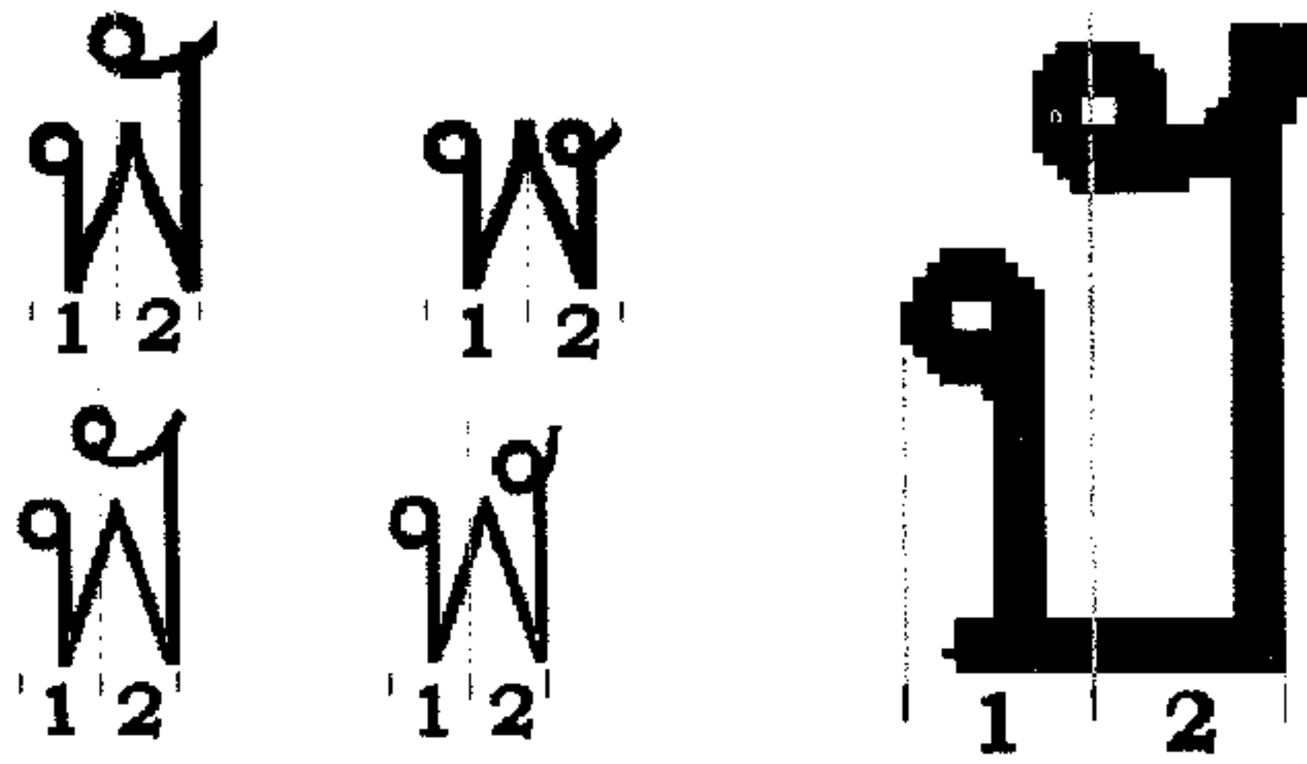
(ก) (ข)

รูปที่ 10 (ก) อักขระซ้อนทับที่กำหนดใช้การแบ่งแยกตามแนวนอน
(ข) รูปแบบอักขระที่ได้หลังการแบ่งแยกตามแนวนอน

ดังนั้น จึงพอจะสรุปได้ว่าสำหรับอักขระระดับฐานที่มีหางเหยียดตรง เช่น ป ฟ ฝ นั้น อัลกอริทึมที่ใช้ในงานวิจัยนี้จะพยายามรักษารูปแบบของตัวอักขระเหล่านี้ให้สมบูรณ์ ส่วนตัวอักขระในแนวระดับบนที่ซ้อนทับกับอักขระเหล่านี้อาจจะถูกตัดทิ้งไปบางส่วน และสำหรับการซ้อนทับของตัวอักขระ ช ซ ฐ ฮ กับตัวอักขระในแนวระดับบนนั้น จะพยายามคงรูปแบบของตัวอักขระแนวระดับบนไว้ โดยบางส่วนของตัวอักษร ช ซ ฐ ฮ อาจจะถูกตัดไปบ้าง ซึ่งรายละเอียดของการตัดแบ่งอธิบายดังต่อไปนี้

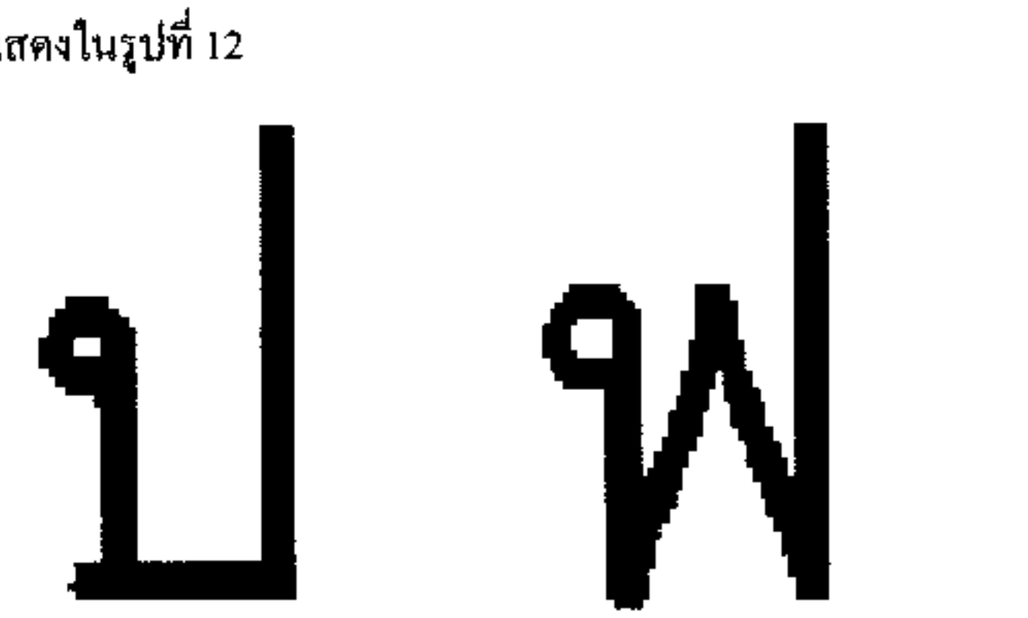
เมื่อทำการ scan หาแนวของอักขระในระดับฐานซึ่งเป็นแนวของบรรทัดของภาษาไทยแล้วก็จะ scan จุดสีดำซึ่งอยู่ในแนวของอักขระระดับฐานนี้ตามแนวตั้ง ซึ่งทำให้ได้ขอบเขตของอักขระในระดับฐานแต่ละตัวได้ หลังจากนั้นจะแบ่งขอบเขตตามแนวตั้ง (ความกว้าง) ของอักขระระดับฐานออกเป็น 2 ส่วน ดังแสดงในรูป 11 การที่แบ่งความกว้างของตัวอักขระออกเป็น 2 ส่วน เพื่อทำการแยกระหว่าง ฬ กับตัว ป ฝ ฟ

ที่ทับซ้อนกับตัวอักขระในแนวระดับบน กล่าวคือ ตัวอักขระ พ จะไม่มีส่วนหางที่ยื่นไปอยู่ในส่วนที่ 1 แต่สำหรับตัวอักขระทับซ้อนกันนั้น จะมีส่วนที่อยู่เหนือระดับฐานที่อยู่ในส่วนที่ 1



รูปที่ 11 แสดงถึงความแตกต่างระหว่างตัวอักขระ " พ " กับ " ฟ "

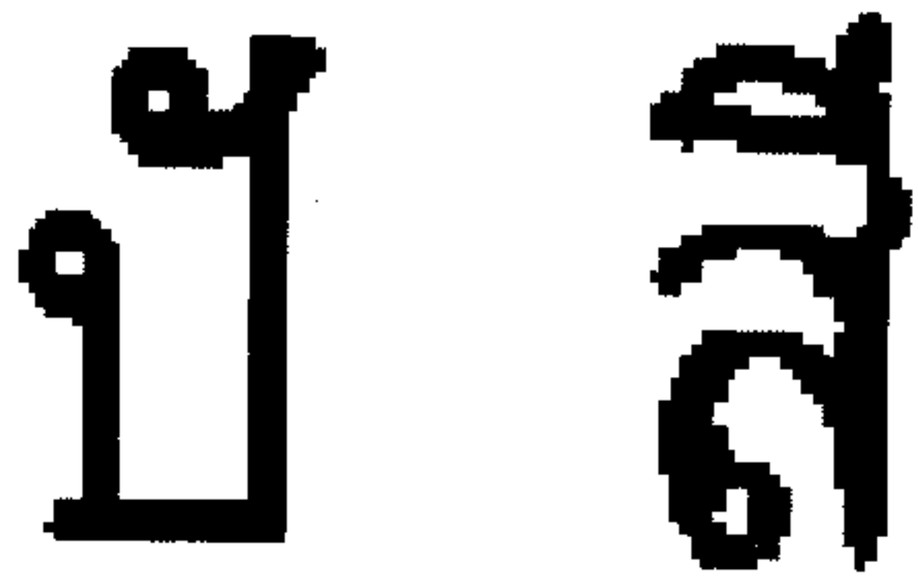
นอกจากนั้น การแยกตัวอักขระ ป ฟ ฝ พ ออกจากตัวอักขระ ช ซ ส ศ ษ นั้น ทำได้โดยการ scan ลงมาข้างล่างจากขอบบนของบรรทัด (แนวระดับฐาน) นอกเหนือจากการ scan ไปข้างบนเหนือระดับบรรทัด ซึ่งจะเห็นว่าสำหรับตัวอักขระ ป ฟ ฝ พ นั้น ส่วนที่ scan ลงมาข้างล่างจะเป็นลักษณะของแท่งตรงเหยียดลงมา ดังแสดงในรูปที่ 12



รูปที่ 12

สำหรับการหาส่วนที่ทับกันระหว่างตัวอักขระระดับฐานกับตัวอักขระระดับบน ทำได้โดยสำหรับตัวอักขระที่มีส่วนที่ยื่นออกไปจากขอบบนของบรรทัดนั้นจะทำการ scan ส่วนที่ยื่นออกไปนี้ตามแนวตั้ง โดยเริ่มจากทางซ้ายสุดของส่วนที่ยื่นออกไปนี้ไปทางขวามือ จนหมดส่วนที่ยื่นออกไป ดังแสดงในรูปที่ 13 หลังจากนั้นจึงทำการ scan ส่วนเนื้อที่เหลือที่อยู่ในระดับบน/บนสุดตามแนวนอน ซึ่งก็คือส่วนของตัวอักขระ

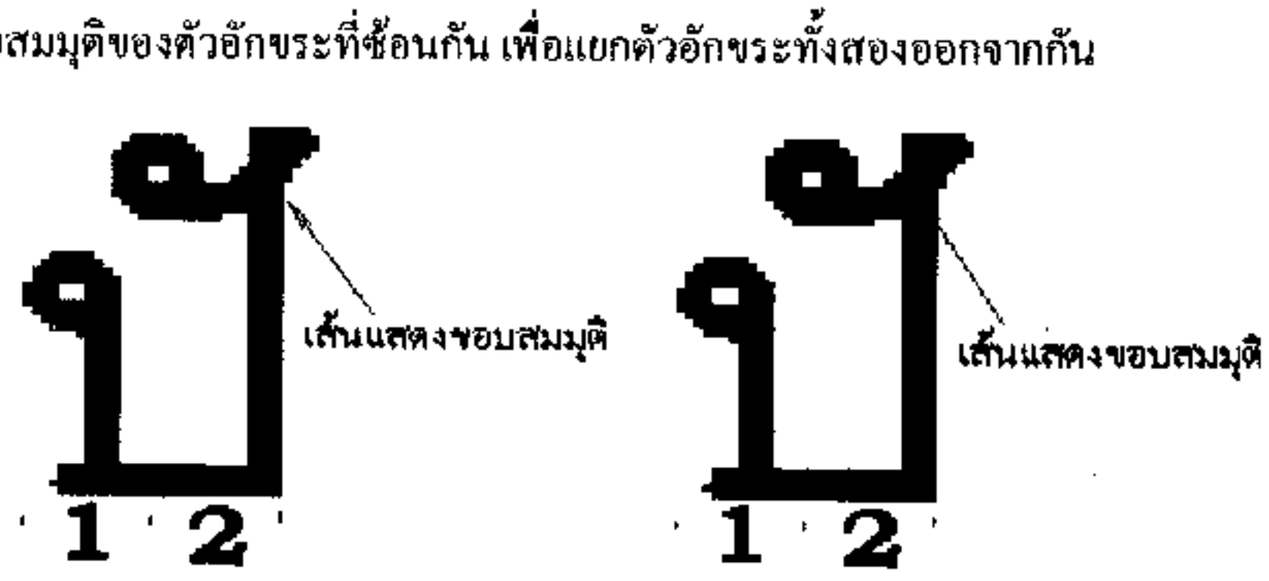
แนวระดับบนและบนสุดนั่นเอง และดังที่กล่าวมาแล้วข้างต้น ส่วนที่ทับกันระหว่างการ scan ตามแนวตั้งและแนวนอนนั้น จะเป็นเนื้อของอักขระระดับฐานสำหรับตัวอักขระ ป ฟ ฝ แต่สำหรับตัวอักขระ ช ซ ส ฮ ศ นั้น ส่วนที่ทับกันจะถือเป็นเนื้อของตัวอักขระระดับบนและบนสุด



รูปที่ 13

3.5 การแยกอักขระด้วยเส้นแสดงขอบสมมุติ

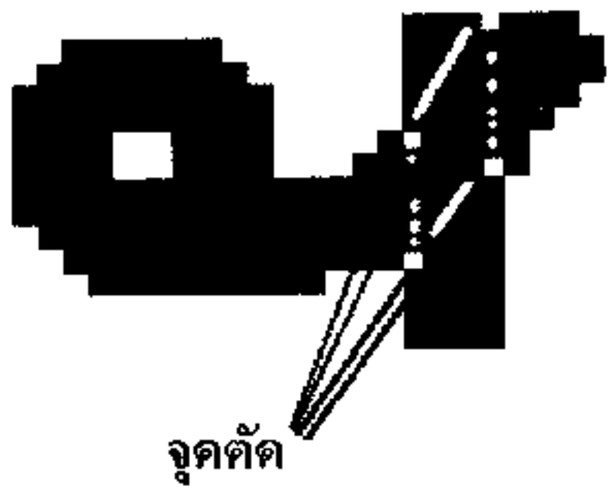
นอกจากการแยกตัวอักขระที่ซ้อนทับกันด้วยวิธีการข้างต้นแล้ว การซ้อนทับกันของอักขระ ป ฟ ฝ กับอักขระระดับบน อาจจะเกิดในลักษณะที่แสดงในรูปที่ 14 กล่าวคือ มีส่วนของตัวอักขระระดับบน/บนสุดที่ยื่นออกไปเกินความกว้างของตัวอักขระระดับฐาน ซึ่งหากใช้วิธีการดังกล่าวข้างต้น จะทำให้ตัวอักขระที่แยกได้ผิดเพี้ยนไปจากรูปแบบที่ควรจะเป็นมาก ดังนั้น ในกรณีที่เกิดการซ้อนทับกันเช่นนี้ จะมีการสร้างเส้นแสดงขอบสมมุติของตัวอักขระที่ซ้อนกัน เพื่อแยกตัวอักขระทั้งสองออกจากกัน



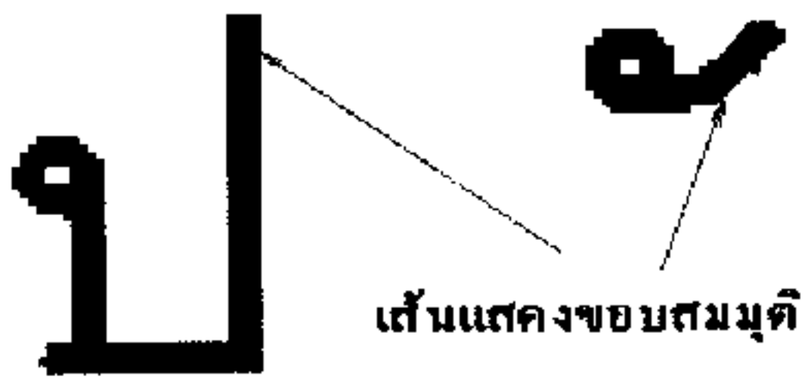
รูปที่ 14

ในการพิจารณาว่าเป็นการซ้อนทับกันในลักษณะนี้นั้น ทำได้โดยพิจารณาว่าตัวอักขระ ป ฟ ฝ มีส่วนที่อยู่ในระดับบนที่ยื่นเกินความกว้างของตัวอักขระจะเกิดรูปแบบของการทับซ้อนแบบนี้ ดังนั้น ด้วยการหาจุดตัดของการ scan ตามแนวตั้งและแนวนอน ดังแสดงในรูปที่ 15 จะทำให้สามารถสร้างเส้นขอบสมมุติของตัวอักขระทั้งสองได้ เส้นขอบสมมุตินี้จะเป็นแนวเพื่อแยกตัวอักขระทั้งสองออกจากกัน ซึ่งผลของ

การแยกตัวอักขระทั้งสองได้แสดงในรูปที่ 16 โดยการ scan ตามแนวตั้ง จะเริ่มตั้งแต่ส่วนที่เกินขอบบนของบรรทัดจากซ้ายไปขวา จนหมดขอบเขตของตัวอักขระ(ความกว้างของตัวอักขระ ป ฟ ฝ นั้น) ส่วนการ scan ตามแนวนอนจะทำในส่วนที่เหลือ



รูปที่ 15 แสดงจุดตัดทั้ง 4 ของเส้นขอบอักขระ ณ จุดซ้อนทับ



รูปที่ 16 ผลลัพธ์ที่ได้จากการแบ่งแยกด้วยเส้นแสดงขอบสมมุติ

3.6 การหาเส้นแสดงขอบของตัวอักขระ

เมื่อผ่านขั้นตอนของการพิจารณาแบ่งแยกตัวอักขระที่ซ้อนทับกันแล้ว ก็จะทำการหาเส้นแสดงขอบของแต่ละอักขระ โดยการ scan หาตัวอักขระจาก block ที่ได้จากการตัดแบ่งอักขระในแนวตั้ง หลังจากนั้นจึง scan ขึ้นไป ในช่วงของอักขระระดับบนของบล็อกนั้น ซึ่งหากมีก็จะ scan ขึ้นไปในช่วงของตัวอักขระระดับบนสุด แต่หากไม่เจอตัวอักขระที่อยู่ในระดับบนก็แสดงว่าไม่มีตัวอักขระที่อยู่ในระดับบนสุด เมื่อทำการ scan ไปในแนวระดับบนแล้ว ก็จะกลับมา scan หาตัวอักขระที่อยู่ระดับล่างของบล็อกต่อไป

ด้วยการหาเส้นขอบของอักขระก็จะทำให้สามารถตัดแบ่งตัวอักขระแต่ละตัวออกจากประโยค และนำไปผ่านขั้นตอนของการรู้จำต่อไป

4. การรู้จำอักขระ

4.1 การวิเคราะห์หาลักษณะสำคัญของอักขระ

เนื่องจากอักขระไทยส่วนใหญ่ประกอบด้วย ส่วนโค้งซึ่งมีลักษณะซับซ้อน จึงมีงานวิจัยซึ่งวิเคราะห์เส้นแสดงขอบของอักขระ เพื่อตัดแบ่งเส้นแสดงขอบอักขระออกเป็นส่วนเว้าและ ส่วนนูน และใช้เป็นลักษณะสำคัญพื้นฐานในการรู้จำอักขระ[1]-[5], [8] ในการวิจัยนี้ใช้หลักการที่เสนอในงานวิจัยข้างต้น เพื่อตัดแบ่งเส้นแสดงขอบของอักขระซึ่งมีลักษณะทั้งที่เป็นส่วนเว้า (concavity) และส่วนนูน (convexity) ออกเป็นส่วนย่อย โดยขั้นตอนแรกรหัสแบบลูกโซ่ของฟรีแมน ซึ่งเป็นรหัสทิศทางที่มีทางทั้งหมด 8 ทิศทาง ดังในรูปที่ 17(ก) จะถูกใช้ในการกำหนดรหัสให้กับทุกจุดบนเส้นแสดงขอบของอักขระ จากนั้นก็จะตรวจหาจุดที่ทำให้เกิดการเปลี่ยนแปลงทิศทางบนเส้นแสดงขอบของอักขระ โดยพิจารณาจากค่ารหัสทิศทาง F_i ที่เปลี่ยนไป เมื่อได้จุดที่ทำให้เกิดการเปลี่ยนแปลงทิศทางบนเส้นแสดงขอบของอักขระแล้ว ก็จะมีการกำหนดค่าพร้อมทั้งเครื่องหมาย + และเครื่องหมาย – ให้กับจุดเปลี่ยนทิศทางเหล่านั้น เพื่อบอกถึงลักษณะการเปลี่ยนทิศทางว่ามีลักษณะตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา โดยเครื่องหมายลบ (–) จะถูกกำหนดให้กับจุดที่ทำให้เกิดการเปลี่ยนทิศตามเข็มนาฬิกา และเครื่องหมายบวก (+) จะถูกกำหนดให้กับจุดที่ทำให้เกิดการเปลี่ยนทิศทวนเข็มนาฬิกา รูปที่ 17(ข) แสดงถึงการกำหนดค่าให้แก่จุดที่เกิดการเปลี่ยนแปลงทิศทาง จากนั้นก็จะเป็นขั้นตอนของการนำค่ารหัสทิศทาง F_i เครื่องหมาย + และเครื่องหมาย – มาใช้ในการกำหนดจุดบ่งความนูน (เครื่องหมาย +) และจุดบ่งความเว้า (เครื่องหมาย –) ซึ่งจุดบ่งความนูนและจุดบ่งความเว้าถูกกำหนดโดยนิยามต่อไปนี้

กำหนดให้ $C_i = (C_{xi}, C_{yi}) \quad : i = 1, \ldots, I$

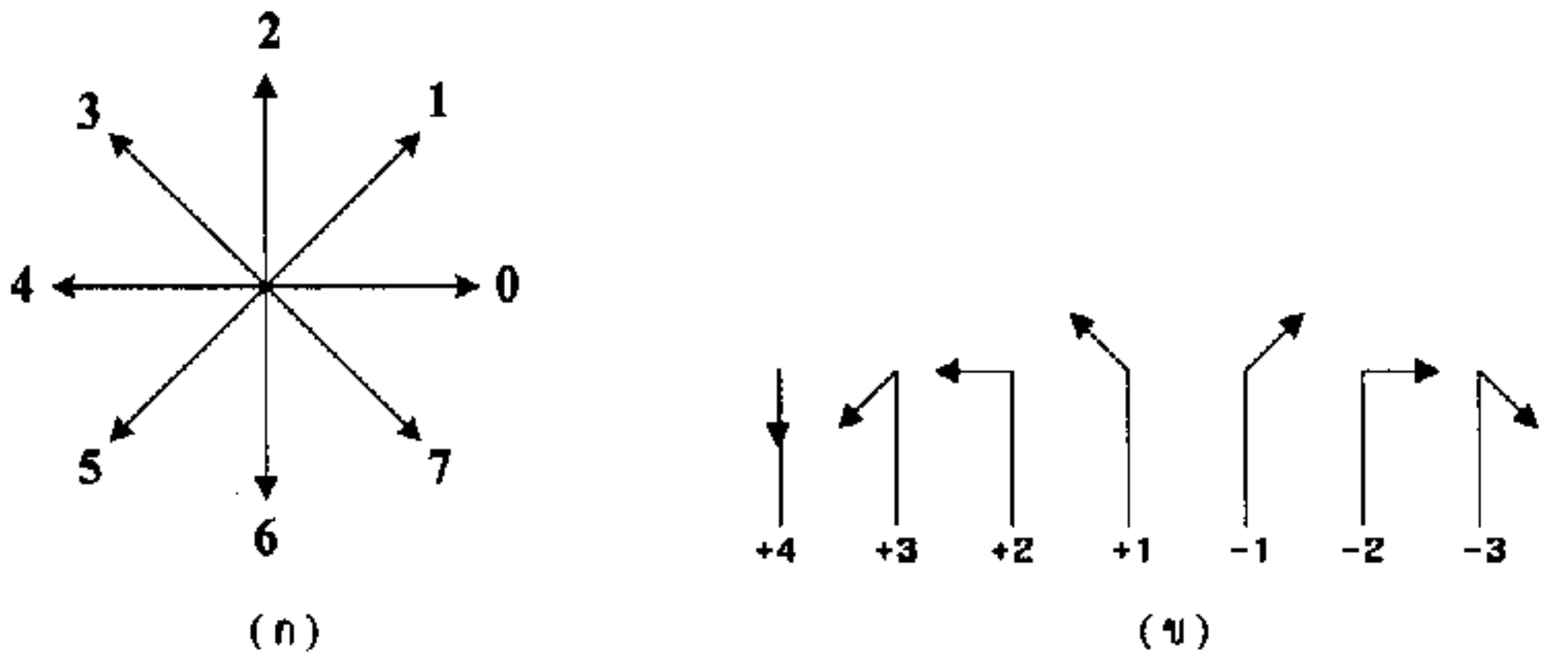
โดย C_i เป็นลำดับของจุดที่มีการเปลี่ยนทิศทางตามรหัสแบบลูกโซ่ของฟรีแมน

และ I เป็นจำนวนจุดเปลี่ยนทิศทางทั้งหมดบนเส้นแสดงขอบของอักขระ

กำหนด $F_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$

กำหนด F_i เป็นค่ารหัสแบบลูกโซ่ของฟรีแมนของจุด C_i แต่ละจุด

และ $S_i \in \{1, 2, 3, 4, -1, -2, -3\}$



(ก) (ข)

รูปที่ 17 (ก) รหัสทิศทางแบบลูกโซ่ของฟรีแมน(F_i)
(ข) รูปแบบต่างๆ ของเครื่องหมาย S_i

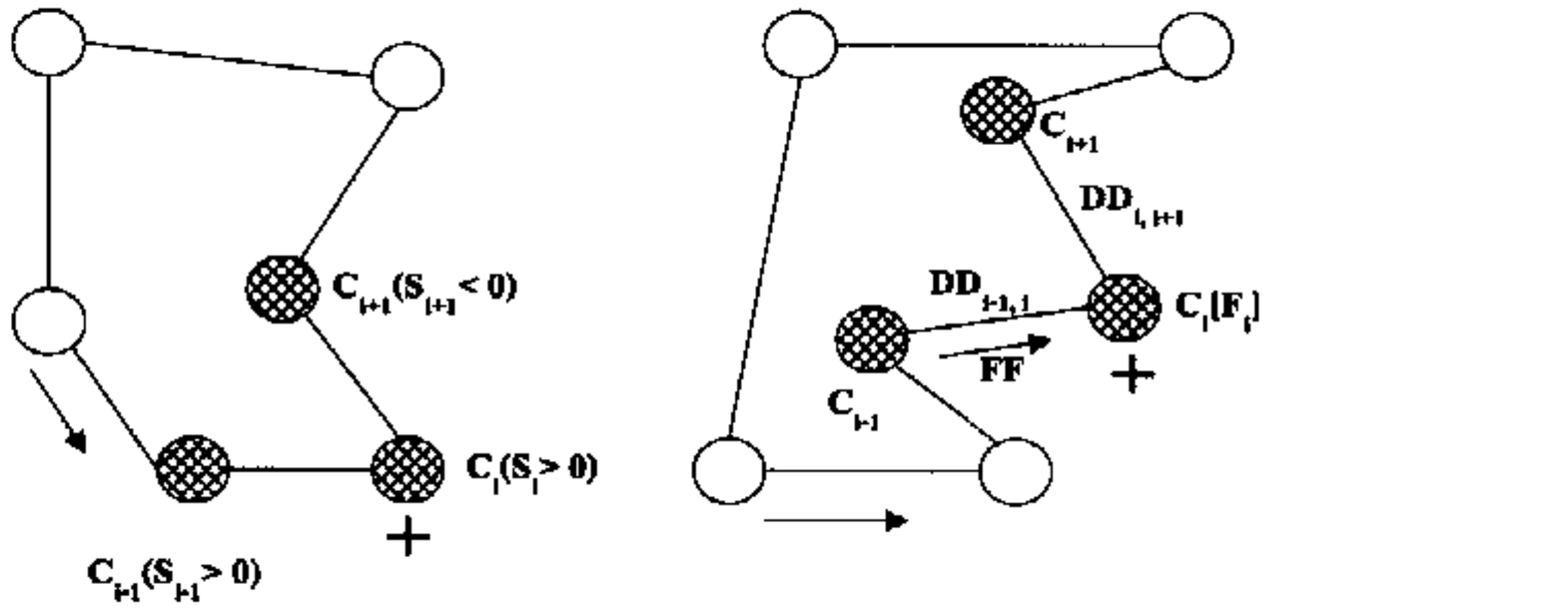
นิยามที่ 1 : ถ้าจุด C_i สอดคล้องกับเงื่อนไขใดเงื่อนไขหนึ่ง ดังต่อไปนี้แล้ว จุด C_i จุดนั้นก็จะถูกกำหนดให้เป็นจุดบ่งความนูน (vertice +)

1.1 $(S_i > 0)$ และ $(S_{i-1} > 0$ หรือ $S_{i+1} > 0)$

1.2 $(S_i > 0)$ และ $(S_{i-1} < 0)$ และ $(S_{i+1} < 0)$
และ $(F_i \neq FF)$ และ $(DD_{i-1,i} \geq K1)$ และ $(DD_{i,i+1} \geq K2)$

โดย $DD_{i-1,i}$ เป็นระยะห่างระหว่างจุด C_{i-1} และ C_i
เมื่อ $DD_{i-1,i}$ = (จำนวนจุดซึ่งอยู่ระหว่างจุดที่ i-1 และจุดที่ i) + 1
$DD_{i,i+1}$ เป็นระยะห่างระหว่างจุด C_i และ C_{i+1}
เมื่อ $DD_{i,i+1}$ = (จำนวนจุดซึ่งอยู่ระหว่างจุดที่ i และจุดที่ i+1) + 1
และ FF เป็นค่ารหัสแบบลูกโซ่ของฟรีแมนของจุดบ่งความนูน/ความเว้า
ซึ่งตรวจพบก่อนที่จะถึงจุด C_i
K1, K2 เป็นค่าคงที่ ซึ่งได้จากการทดลองกับข้อมูลทดสอบ

ถ้าจุด C_i ถูกกำหนดให้เป็นจุดบ่งความนูน และ $DD_{i,i+1}$ มีค่ามากกว่า K3 แล้วค่า FF จะถูกกำหนดให้มีค่าใหม่ตามค่าของ F_i โดย FF จะถูกกำหนดให้มีค่าเริ่มต้นเป็น 4 และ K3 เป็นค่าคงที่ ซึ่งได้จากการทดลองกับข้อมูลทดสอบ รูปที่ 18 แสดงตัวอย่างของการกำหนดจุดบ่งความนูนตามนิยามที่ 1



รูปที่ 18 ตัวอย่างการกำหนดจุดบ่งความนูนให้แก่จุด C_i

นิยามที่ 2 : ถ้าจุด C_i สอดคล้องกับเงื่อนไขใดเงื่อนไขหนึ่ง ดังต่อไปนี้แล้ว จุด C_i จุดนั้นก็จะถูกกำหนดให้เป็นจุดบ่งความเว้า (vertice)

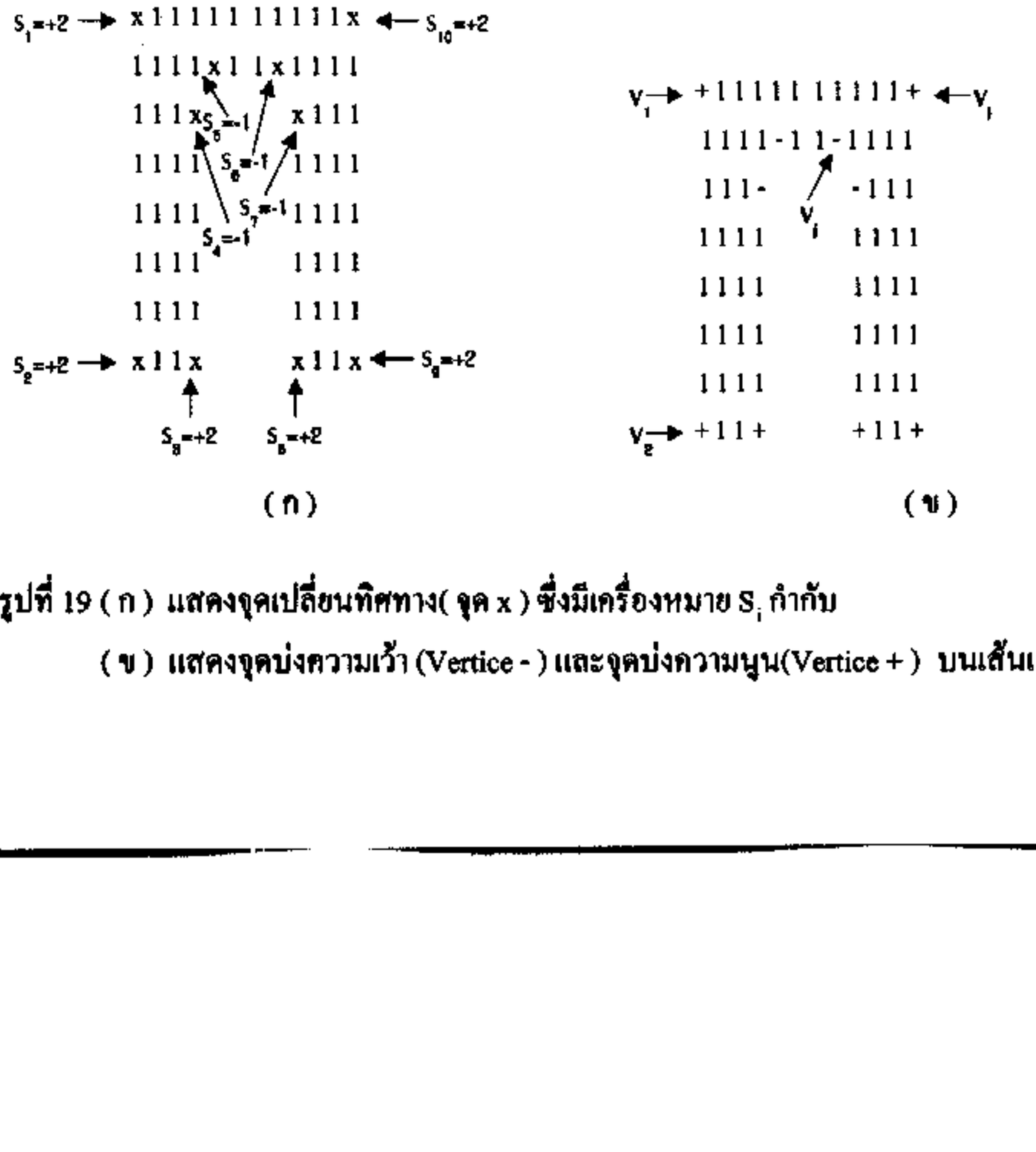
2.1 $(S_i < 0)$ และ $(S_{i-1} < 0$ หรือ $S_{i+1} < 0)$

2.2 $(S_i < 0)$ และ $(S_{i-1} > 0)$ และ $(S_{i+1} > 0)$

และ $(F_i \neq FF)$ และ $(DD_{i-1,i} \geq K1)$ และ $(DD_{i,i+1} \geq K2)$

สำหรับตัวอย่างของการกำหนดค่ารหัสแบบลูกโซ่ของฟรีแมน ค่าเครื่องหมายให้แก่จุดเปลี่ยนทิศทางและการกำหนดจุดบ่งความนูน และจุดบ่งความเว้าให้แก่เมตริกซ์ของตัวอักขระได้แสดงในรูป 19

เมื่อมีการกำหนดจุดบ่งความนูนและจุดบ่งความเว้าให้แก่จุด C_i แล้ว ก็จะทำให้สามารถตัดแบ่งเส้นแสดงขอบของอักขระออกเป็นส่วนโค้งย่อย ตามลักษณะของความเว้าและความนูนได้ต่อไป



(ก) (ข)

รูปที่ 19 (ก) แสดงจุดเปลี่ยนทิศทาง(จุด x) ซึ่งมีเครื่องหมาย S_i กำกับ

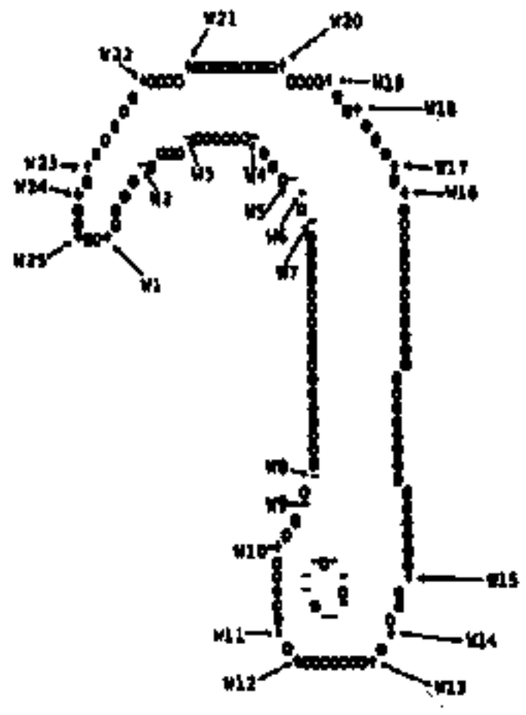
(ข) แสดงจุดบ่งความเว้า (Vertice -) และจุดบ่งความนูน(Vertice +) บนเส้นแสดงขอบของอักขระ

4.2 คำจำกัดความของส่วนโค้งเว้าและส่วนโค้งนูน

ส่วนโค้งเว้า คือส่วนของเส้นโค้งที่ผ่านจุดบ่งความเว้า ซึ่งอยู่ระหว่างจุดบ่งความนูน 2 จุด โดยจุดบ่งความนูนทั้งสองจุดจะเป็นจุดปลายทั้งสองของส่วนโค้งเว้านั้น

ส่วนโค้งนูน คือส่วนของเส้นโค้งที่ผ่านจุดบ่งความนูน ซึ่งอยู่ระหว่างจุดบ่งความเว้า 2 จุด โดยจุดบ่งความเว้าทั้งสองจุดจะเป็นจุดปลายทั้งสองของส่วนโค้งนูนนั้น

รูปที่ 20 แสดงตัวอย่างการตัดแบ่งตัวอักขระไทยเป็นส่วนโค้งนูนและส่วนโค้งเว้า โดยส่วนที่ผ่านจุด W_9, W_{10}, W_{11}, W_{12}, W_{13}, W_{14}, W_{15}, W_{16}, W_{17}, W_{18}, W_{19}, W_{20}, W_{21}, W_{22}, W_{23}, W_{24}, W_{25}, W_1 และ W_2 เป็นส่วนโค้งนูน และส่วนที่ผ่านจุด W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8, W_9 และ W_{10} เป็นส่วนโค้งเว้า



รูปที่ 20 ตัวอย่างอักขระไทยที่มีการกำหนดจุดบ่งความนูน และจุดบ่งความเว้า

4.3 ลักษณะสำคัญของตัวอักขระ (features of character)

เนื่องจากอักขระไทยมีจำนวนมาก ดังนั้น ลักษณะสำคัญทางโกลบอลของเส้นแสดงขอบของอักขระ ได้แก่ จำนวนหัวของอักขระ (H) จำนวนส่วนโค้งทั้งหมดของอักขระ (Z) คือ ทั้งส่วนโค้งเว้าและส่วนโค้งนูนรวมกัน จะถูกนำมาใช้ในการจัดแบ่งกลุ่มของอักขระ

นอกจากนั้น อัตราส่วนความกว้างต่อความสูงของอักขระดังแสดงในรูปที่ 21 จะถูกนำไปใช้เป็นลักษณะสำคัญอีกอันหนึ่ง ในการแยกประเภทของอักขระในขั้นแรก (rough classification) เนื่องจากอักขระบางตัวเมื่อใช้จำนวนหัวและจำนวนส่วนโค้งของอักขระในการแยกประเภทแล้ว จะยังถูกจัดให้อยู่ในกลุ่มเดียวกัน เช่น บ กับ ป หรือ ผ กับ ฝ เป็นต้น แต่เมื่อนำอัตราส่วนความกว้างต่อความสูงของอักขระเข้ามาเป็นลักษณะสำคัญเพิ่มอีกลักษณะหนึ่งในการแยกประเภทแล้ว ก็จะทำให้สามารถแยก บ ออกจาก ป หรือ ผ ออกจาก ฝ ได้ ซึ่งการทำเช่นนี้นอกจากจะช่วยเพิ่มความถูกต้องในการรู้จำอักขระให้สูงขึ้นแล้ว ยังจะมีส่วนช่วยทำให้ลดจำนวนของอักขระที่ต้องถูกนำมาเปรียบเทียบให้น้อยลง ทำให้สามารถลดเวลาในการประมวลผลเพื่อการรู้จำอักขระลงได้

Num_col = 40 จุด : Num_row = 40 จุด
WpH = 40/40 = 1.00
(ก) อักขระตัว พ

Num_col = 40 จุด : Num_row = 60 จุด
WpH = 40/60 = 0.67
(ข) อักขระตัว ฟ

รูปที่ 21 ตัวอย่างการคำนวณค่าอัตราส่วนความกว้างต่อความสูงของอักขระ(WpH)

สำหรับลักษณะสำคัญของส่วนโค้งที่ได้จากการตัดแบ่งเส้นแสดงขอบเขตของอักขระนั้น เนื่องจากมีการใช้การเปรียบเทียบแบบไดนามิคโปรแกรมมิ่ง ซึ่งทำให้เกิดการยืดหยุ่นของการเปรียบเทียบ ในการวิจัยนี้จึงสามารถใช้ลักษณะสำคัญทางโลคอลแบบง่าย ๆ ดังต่อไปนี้ เพื่อทำการหาความคล้ายระหว่างตัวอักขระ

จากส่วนโค้งที่ได้จากการตัดแบ่งเส้นแสดงขอบของตัวอักขระ ให้ $P_j = (P_{xj}, P_{yj}) : j = 1 \sim J$ เป็นลำดับของจุดบ่งความนูน ความเว้าของส่วนโค้งหนึ่ง ๆ แล้ว ระยะห่าง (ความยาว) ระหว่างจุดบ่งความนูนความเว้า (P_j, P_{j+1}) ที่ติดกันบนส่วนโค้งหาได้โดย

$$D_{j,j+1} = \sqrt{(P_{x,j+1} - P_{x,j})^2 + (P_{y,j+1} - P_{y,j})^2}$$

ระยะห่างนี้จะถูกคำนวณระหว่างทุก ๆ จุดบ่งความนูน ความเว้าของส่วนโค้งนั้น ๆ ทำให้ได้ $D_k : k = 1 \sim K$ (เมื่อ $K = J - 1$) เป็นลำดับของระยะห่างที่คำนวณได้

ในงานวิจัย[8] ระยะห่างระหว่างจุดบนส่วนโค้งนี้ถูกใช้ในการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งเพื่อรู้จักตัวพิมพ์อักขระไทยหลายรูปแบบ ซึ่งได้ผลการทดลอง 94.7% เพื่อเพิ่มประสิทธิภาพในการรู้จำตัวอักขระให้มากขึ้น ในงานวิจัยนี้ได้ทดลองใช้มุมของเส้นตรงที่เชื่อมระหว่างจุดที่ติดกันบนส่วนโค้งนูน/เว้าเป็นอีกลักษณะสำคัญของส่วนโค้งนั้นโดยให้

$A_k : k = 1 \sim K$ เป็นลำดับของมุมของเส้นตรงที่เชื่อมโยงจุด P_j, P_{j+1} ของส่วนโค้งจะได้

$$A_k = \tan^{-1}\left[\frac{P_{y,j+1} - P_{y,j}}{P_{x,j+1} - P_{x,j}}\right]$$

ดังนั้น อักขระ II ใด ๆ จึงอาจจะถูกแทนด้วยลักษณะสำคัญของอักขระนั้นดังต่อไปนี้

$$II = \left\{ H_{II}, Z_{II}, WpH_{II}, \left[[D_{k_L}]_{k_L=1}^{K_L} \right]_{L=1}^{Z_{II}}, \left[[A_{k_L}]_{k_L=1}^{K_L} \right]_{L=1}^{Z_{II}} \right\}$$

4.4 ไดนามิคโปรแกรมมิ่ง

4.4.1 การเปรียบเทียบแบบไดนามิคโปรแกรมมิ่ง

การเปรียบเทียบแบบไดนามิคโปรแกรมมิ่ง เป็นอัลกอริทึมในการเปรียบเทียบรูปแบบที่มีการปรับตัวยืดหยุ่นทางแกนเวลา ซึ่งจะทำให้เกิดประสิทธิภาพในการนำไปใช้งาน ดังนั้น ไดนามิคโปรแกรมมิ่งจึงได้มีการนำไปใช้กันอย่างกว้างขวางสำหรับการรู้จำแบบของเสียงพูด และการรู้จำอักขระแบบออนไลน์ (Sakoe, H. and Chiba, S.)[10]-[11] ในการวิจัยนี้จึงได้นำอัลกอริทึม ของการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งมาประยุกต์ใช้กับการรู้จำอักขระไทยในระบบออฟไลน์ เนื่องจากความสามารถในการเปรียบเทียบแบบยืดหยุ่นของไดนามิคโปรแกรมมิ่ง ทำให้สามารถใช้ลักษณะสำคัญแบบง่าย ๆ ของส่วนโค้ง ซึ่งคือระยะห่างระหว่างจุดบ่งความนูนความเว้าที่ติดกัน เพื่อหาความคล้ายของส่วนโค้ง ตลอดจนความคล้ายกันของตัวอักขระได้อย่างมีประสิทธิภาพ

ก่อนที่จะอธิบายถึงการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งที่ใช้ในการวิจัยนี้ จะขอกล่าวถึงหลักการของการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งที่ใช้ในการรู้จำเสียงซึ่งเสนอโดย Sokoe, H. และ Chiba, S.[10]-[11] ดังต่อไปนี้

รูปแบบของเสียงพูด สามารถแทนได้โดยการดึงลักษณะสำคัญของรูปแบบนี้ ๆ ออกมา และเขียนแทนในลักษณะที่เป็นลำดับของเวกเตอร์ของลักษณะสำคัญ (feature vectors) ในแกนของเวลาได้ดังนี้

$$A = a_1, a_2, a_3, \ldots, a_i, \ldots, a_I$$

$$B = b_1, b_2, b_3, \ldots, b_j, \ldots, b_J$$

ความแตกต่างของฟีเจอร์เวกเตอร์ของรูปแบบ 2 เวกเตอร์ คือ a_i และ b_j คำนวณได้ดังนี้

$$d(i, j) = | a_i - b_j |$$

เนื่องจากขนาดของฟีเจอร์เวกเตอร์ I ของ A อาจจะแตกต่างกับขนาดของฟีเจอร์เวกเตอร์ J ของ B ดังนั้น จึงเป็นการยากที่ทราบว่าฟีเจอร์ลำดับที่ a_i ของ A ควรจะนำมาเปรียบเทียบเพื่อหาความแตกต่างกับฟีเจอร์ลำดับที่ b_j ใดของรูปแบบ B จึงจะทำให้สามารถคำนวณหาความแตกต่างของรูปแบบ A และรูปแบบ B ได้ถูกต้องที่สุด แต่อย่างไรก็ตาม โดยการประยุกต์ของไดนามิคโปรแกรมมิ่ง ทำให้สามารถคำนวณหาความแตกต่างสะสมที่น้อยที่สุด $g(i, j)$ เมื่อมีการเปรียบเทียบฟีเจอร์ตั้งแต่ $a_1, a_2, \ldots$ จนถึง a_i ของ A กับฟีเจอร์ตั้งแต่ $b_1, b_2, \ldots$ จนถึง b_j ของ B ได้ดังเช่นสมการไดนามิคโปรแกรมมิ่งต่อไปนี้

แตกต่างสะสมที่น้อยที่สุด g(i, j) เมื่อมีการเปรียบเทียบฟีเจอร์ตั้งแต่ $a_1, a_2, \ldots$ จนถึง a_i ของ A กับฟีเจอร์ตั้งแต่ $b_1, b_2, \ldots$ จนถึง b_j ของ B ได้ดังเช่นสมการไดนามิคโปรแกรมมิ่งต่อไปนี้

$$g(i,j) = \min \begin{bmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{bmatrix}$$

ซึ่งเมื่อ i = I และ j = J แล้ว g(I, J) คือความแตกต่างสะสมที่น้อยที่สุดของการเปรียบเทียบฟีเจอร์ $a_1, a_2, \ldots, a_i$ ของรูปแบบ A และฟีเจอร์ $b_1, b_2, \ldots, b_j$ ของรูปแบบ B ซึ่งก็คือความแตกต่างสะสมของรูปแบบ A และ B นั่นเอง

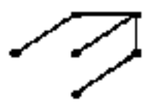
นอกจากสมการไดนามิคโปรแกรมมิ่งที่กล่าวมาข้างต้นแล้ว Sakoe และ Chiba ได้เสนอสมการไดนามิคโปรแกรมมิ่ง g(i, j) รูปแบบต่าง ๆ เพื่อคำนวณหาความแตกต่างสะสมของ รูปแบบของเสียงพูด A และ B ดังแสดงในตารางที่ 2 และโดยการคำนวณหา g(i, j) : i = 1~I ; j = j ~ J จะสามารถหาความแตกต่างของรูปแบบ A และ B ได้โดย

$$D(A, B) = g(1, J)/NP$$

โดยที่ NP = I + J ในกรณีของแกนเวลาสมมาตร (symmetric forms)

NP = I ในกรณีของแกนเวลาไม่สมมาตร (asymmetric forms)

ตารางที่ 2 อัลกอริทึมของสมการไดนามิคโปรแกรมมิ่งแบบสมมาตร และแบบไม่สมมาตร

P	Schematic explanation	Symmetric / Asymmetric	DP-equation g(i, j) =
0		Symmetric	min { g(i, j-1) + d(i, j) ; g(i-1, j-1) + 2d(i, j) ; g(i-1, j) + d(i, j) }
		Asymmetric	min { g(i, j-1) ; g(i-1, j-1) + d(i, j) ; g(i-1, j) + d(i, j) }
1/2		Symmetric	min { g(i-1, j-3) + 2d(i, j-2) + d(i, j-1) + d(i, j) ; g(i-1, j-2) + 2d(i, j-1) + d(i, j) ; g(i-1, j-1) + 2d(i, j) ; g(i-2, j-1) + 2d(i-1, j) + d(i, j) ; g(i-3, j-1) + 2d(i-2, j) + d(i-1, j) + d(i, j) }
		Asymmetric	min { g(i-1, j-3) + [d(i, j-2) + d(i, j-1) + d(i, j)]/3 ; g(i-1, j-2) + [d(i, j-1) + d(i, j)]/2 ; g(i-1, j-1) + 2d(i, j) ; g(i-2, j-1) + 2d(i-1, j) + d(i, j) ; g(i-3, j-1) + 2d(i-2, j) + d(i-1, j) + d(i, j) }
1		Symmetric	min { g(i-1, j-2) + 2d(i, j-1) + d(i, j) ; g(i-1, j-1) + 2d(i, j) ; g(i-2, j-1) + 2d(i-1, j) + d(i, j) }
		Asymmetric	min { g(i-1, j-2) + [d(i, j-1) + d(i, j)]/2 ; g(i-1, j-1) + d(i, j) ; g(i-2, j-1) + d(i-1, j) + d(i, j) }
2		Symmetric	min { g(i-2, j-3) + 2d(i-1, j-2) + 2d(i, j-1) + d(i, j) ; g(i-1, j-1) + 2d(i, j) ; g(i-2, j-2) + 2d(i-2, j-1) + 2d(i-1, j) + d(i, j) }
		Asymmetric	min { g(i-2, j-3) + 2[d(i-1, j-2) + d(i, j-1) + d(i, j)]/3 ; g(i-1, j-1) + d(i, j) ; g(i-2, j-2) + d(i-2, j-1) + d(i-1, j) + d(i, j) }

4.4.2 การเปรียบเทียบอักขระโดยวิธีการไดนามิคโปรแกรมมิ่ง

ในบทความ (1) – (5), Hiranvanichakorn. P, Agui, T และ Nakajima, M. ได้เสนอวิธีการการเปรียบเทียบตัวอักขระระหว่างอักขระข้อมูล และอักขระต้นแบบโดยการคำนวณความคล้ายของแต่ละส่วนโค้งตลอดจนความคล้ายของตัวอักขระทั้งสอง เพื่อหาตัวอักขระต้นแบบที่มีความคล้ายตัวอักขระข้อมูลมากที่สุด ซึ่งวิธีการนี้ใช้ได้ผลดีในการรู้จำตัวอักขระไทย ในการวิจัยนี้จึงนำเอาวิธีการนี้มาประยุกต์ร่วมกับไดนามิคโปรแกรมมิ่ง ดังวิธีการต่อไปนี้

ให้ MM เป็นอักขระต้นแบบ (model) ซึ่งถูกแทนด้วยลักษณะสำคัญของตัวอักขระดังต่อไปนี้

$$MM = \left\{ H_{MM} , Z_{MM} , WpH_{MM} , \left[[D_{k_M}]_{k_M=1}^{K_M} \right]_{M=1}^{Z_{MM}}, \left[[A_{k_M}]_{k_M=1}^{K_M} \right]_{M=1}^{Z_{MM}} \right\}$$

เมื่อทราบลักษณะสำคัญของอักขระที่ต้องการรู้จำแล้ว ก็สามารถนำลักษณะสำคัญนี้ไปใช้ในการเปรียบเทียบกับอักขระต้นแบบที่เก็บอยู่ในพจนานุกรมของอักขระ เพื่อที่จะรู้จำให้ได้ว่า อักขระที่อ่านเข้ามานี้คืออักขระใด ซึ่งก่อนที่จะเข้าสู่วิธีการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งนี้ ลักษณะสำคัญโดยรวม (global features) ของอักขระได้แก่ อัตราส่วนความกว้างต่อความสูงของอักขระ จำนวนหัวของอักขระ และจำนวนส่วนโค้งของอักขระจะถูกนำมาใช้ เพื่อการแยกประเภทของอักขระในขั้นแรกก่อน การแยกประเภทของอักขระในขั้นแรกนี้ นอกจากจะช่วยทำให้สามารถลดจำนวนอักขระที่จะต้องถูกนำไปเปรียบเทียบในขั้นตอนของไดนามิคโปรแกรมมิ่งลงได้แล้ว ยังช่วยทำให้ลดเวลาในขั้นตอนของการรู้จำอักขระลงไปอีกด้วย

หลังจากการเปรียบเทียบลักษณะสำคัญโดยรวมของตัวอักขระทั้งสองแล้ว ไดนามิคโปรแกรมมิ่งได้ถูกนำมาประยุกต์ใช้เพื่อหาความคล้ายของส่วนโค้ง ตลอดจนความคล้ายของตัวอักขระดังต่อไปนี้

ให้ส่วนโค้ง L แทนส่วนโค้งใด ๆ ของอักขระที่ต้องการรู้จำ

และส่วนโค้ง M แทนส่วนโค้งใด ๆ ของอักขระต้นแบบ

ดังนั้น สามารถเขียนส่วนโค้ง L และ M ในรูปของเวกตเตอร์ได้ดังนี้

ส่วนโค้ง $L = l_1, l_2, \ldots, l_{ia}, \ldots, l_{IA}$

ส่วนโค้ง $M = m_1, m_2, \ldots, m_{ja}, \ldots, m_{JA}$

โดย l_{ia} และ m_{ja} จะเป็นลำดับของลักษณะสำคัญของส่วนโค้ง ซึ่งในที่นี้ก็คือ ความยาวระหว่างจุดบ่งความผูน ความเว้า 2 จุด ที่อยู่ติดกันของส่วนโค้ง (D_k โดยที่ $k = 1 \sim IA$ สำหรับส่วนโค้ง L และ $k = 1 \sim JA$ สำหรับส่วนโค้ง M) และมุมของเส้นเส้นตรงที่เชื่อมโยงจุดที่ติดกันของส่วนโค้ง (A_K) โดยที่ $K = 1 \sim IA$ สำหรับส่วนโค้ง L และ $K = 1 \sim JA$ สำหรับส่วนโค้ง M)

ด้วยวิธีการของไดนามิคโปรแกรมมิ่ง สามารถหาความแตกต่างระหว่างเวกเตอร์ l_{ia} และ m_{ja} ได้ดังนี้

ให้ d(ia, ja) แทนความแตกต่างระหว่างเวกเตอร์ l_{ia} และ m_{ja}

ดังนั้น $d(ia, ja) = wd \mid D_{ia} - D_{ja} \mid + wa \mid A_{ia} - A_{ja} \mid$

โดยที่ wd เป็นน้ำหนักถ่วงให้แก่ลักษณะสำคัญในเชิงความยาว และ wa เป็นน้ำหนักถ่วงให้แก่ลักษณะสำคัญโนเชิงมุม

เมื่อหาความแตกต่างระหว่างเวกเตอร์ I_{ia} และ m_{ja} ได้แล้ว ก็จะนำค่าแตกต่างนี้มาใช้ในการคำนวณหาความแตกต่างระหว่างส่วนโค้ง L และ M ได้ดังนี้

ให้ D(L, M) แทนความแตกต่าง (ความคล้าย) ระหว่างส่วนโค้ง L และ M

ดังนี้ $D(L, M) = g(IA, JA) / NA$

เมื่อ $NA = IA + JA$ ในกรณีของแกนเวลาสมมาตร

$NA = IA$ ในกรณีของแกนเวลาไม่สมมาตร

จากหลักการข้างต้นนี้ทำให้สามารถคำนวณหาความแตกต่าง (ความคล้าย) ระหว่างส่วนโค้งใดๆ ของอักขระข้อมูลและอักขระต้นแบบได้ แต่โดยทั่วไปแล้วการป้อนข้อมูลอักขระตัวหนึ่งๆ เข้าสู่คอมพิวเตอร์ในแต่ละครั้งนั้น ตัวอักขระอาจจะถูกหมุนไปทำให้ลำดับของส่วนโค้งที่ได้แต่ละครั้งไม่ตรงกัน ซึ่งเป็นการยากที่จะให้คอมพิวเตอร์ทราบว่าควรจะเปรียบเทียบส่วนโค้งใดของตัวอักขระข้อมูล กับส่วนโค้งของอักขระต้นแบบ ในการวิจัยนี้จึงใช้วิธีการหาส่วนโค้งที่คล้ายกันมากที่สุดระหว่างตัวอักขระ เพื่อใช้เป็นจุดเริ่มต้นของการเปรียบเทียบของแต่ละส่วนโค้งของอักขระทั้งสอง ซึ่งได้เสนอในงานวิจัย (1) – (5) ซึ่งได้ปรับปรุงให้ง่ายขึ้น โดยที่ส่วนโค้ง L และ M ใดๆ จะถูกกำหนดให้เป็นส่วนโค้งที่คล้ายกันมากที่สุดระหว่างตัวอักขระ เมื่อส่วนโค้งทั้งสองสอดคล้องตามเงื่อนไขดังต่อไปนี้

(ก) $D(L, M) < D_c$ เมื่อ D_c เป็นค่าที่กำหนดไว้ค่าหนึ่ง

(ข) $$\left\{ L, M \mid \min_{L=1}^{Z_{II}} \left(\min_{M=1}^{Z_{MM}} (D(L, M)) \right) \right\}$$

เมื่อได้ส่วนโค้งคู่ที่มีความคล้ายกันมากที่สุดระหว่างอักขระที่ต้องการรู้จำกับอักขระต้นแบบแล้ว ก็จะใช้คู่ส่วนโค้งนี้เป็นคู่ส่วนโค้งเริ่มต้นสำหรับการหาค่าความแตกต่างระหว่างส่วนโค้งคู่ถัดไปจนครบหมดทุกส่วนโค้ง จากนั้นก็จะนำค่าความแตกต่างของทุกคู่ของส่วนโค้งที่ได้มาหาผลรวม แล้วหารด้วยจำนวนส่วนโค้งอีกครั้งหนึ่ง ซึ่งผลลัพธ์ที่ได้ในครั้งสุดท้ายนี้ก็คือ ค่าความแตกต่างระหว่างอักขระที่ต้องการรู้จำกับอักขระต้นแบบ ดังต่อไปนี้

ให้ $DC_{II,MM}$ แทนความแตกต่างระหว่างอักขระที่ต้องการรู้จำกับอักขระต้นแบบ

Z แทนจำนวนส่วนโค้ง

$D(L, M)$ แทนความแตกต่างระหว่างส่วนโค้ง L และ M

ดังนั้น $$DC_{II,MM} = \sum_{L,M} D(L, M) / Z$$

การหาค่าความแตกต่างระหว่างอักขระที่ต้องการรู้จำ กับอักขระต้นแบบนั้นจะต้องกระทำกับอักขระต้นแบบทุกตัวที่อยู่ในกลุ่มอักขระที่มีลักษณะคล้ายคลึงกัน เพื่อหาว่าอักขระต้นแบบตัวใดให้ค่าความแตกต่างที่น้อยที่สุด ก็จะถือว่าอักขระต้นแบบตัวนั้นเป็นอักขระที่รู้จำได้ โดยค่าความแตกต่างที่น้อยที่สุดนี้จะต้องมีค่าไม่มากกว่าค่าที่กำหนดไว้ค่าหนึ่งด้วย มิฉะนั้นก็จะถือว่าอักขระที่ต้องการรู้จำนั้นเป็นอักขระที่ไม่สามารถบอกได้ว่าเป็นอักขระใด (rejected) นั่นก็คือ อักขระที่มีค่า $DC_{II,MM}$ สอดคล้องกับเงื่อนไขต่อไปนี้จะถือว่าเป็นอักขระที่รู้จำได้

(ก) $DC_{II,MM} < D_C$ เมื่อ D_C เป็นค่าที่กำหนดไว้ค่าหนึ่ง

(ข) $\min(DC_{II,MM})$ เมื่อ MM = 1, ... , KM

และ KM เป็นจำนวนอักขระต้นแบบทั้งหมดที่อยู่ในกลุ่มเดียวกันกับอักขระที่ต้องการรู้จำ

ในบางกรณีอักขระที่ต้องการรู้จำและอักขระต้นแบบจะมีจำนวนส่วนโค้งต่างกัน ซึ่งอาจเป็นไปได้เนื่องจากอักขระตัวเดิม เมื่อนำไปกวาดตรวจด้วยแสงอีกครั้งหนึ่งแล้ว จะได้ข้อมูลที่แตกต่างจากเดิมดังนั้นในกรณีที่ค่าความแตกต่างที่น้อยที่สุดระหว่างอักขระที่ต้องการรู้จำกับอักขระต้นแบบมีค่ามากกว่าค่าที่กำหนดไว้ค่าหนึ่งแล้ว ก็จะนำอักขระที่ต้องการรู้จำนั้นมาเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งอีกครั้งหนึ่ง

ขั้นตอนของการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งในขั้นนี้จะทำในลักษณะต่างจากเดิมเล็กน้อย โดยในขั้นนี้จะไม่มีการหาค่าความแตกต่างของส่วนโค้งทีละส่วนโค้ง และจะทำโดยการหาส่วนโค้งคู่ที่คล้ายกันมากที่สุดมาให้ได้ก่อน จากนั้นก็จะใช้คู่ส่วนโค้งนี้เป็นส่วนโค้งเริ่มต้นสำหรับหาค่าความแตกต่างระหว่างอักขระที่ต้องการรู้จำกับอักขระต้นแบบ

ดังนั้น สามารถจะแทนอักขระที่ต้องการรู้จำ กับอักขระต้นแบบได้โดยใช้ลำดับของเวกเตอร์ที่แสดงลักษณะสำคัญ ดังต่อไปนี้

สมมุติให้ II แทนรูปแบบของอักขระที่ต้องการรู้จำ

และ MM แทนรูปแบบของอักขระต้นแบบ

ดังนั้น สามารถเขียนรูปแบบของอักขระในลักษณะของเวกเตอร์ได้ดังนี้

$II = ii_1, ii_2, \ldots, ii_s, \ldots, ii_S$

$MM = mm_1, mm_2, \ldots, mm_t, \ldots, mm_T$

โดยที่ ii_1 และ mm_1 เป็นฟีเจอร์แรกของส่วนโค้งคู่ที่คล้ายกันมากที่สุดระหว่างตัวอักขระทั้งสอง

S และ T เป็นผลรวมของจำนวนจุดบ่งความนูนและจุดบ่งความเว้าของอักขระที่ต้องการรู้จำ และอักขระต้นแบบ ตามลำดับ

สำหรับ ii_s และ mm_t ประกอบด้วย 2 ส่วนคือ ความยาวระหว่างจุดบ่งความนูน (หรือจุดบ่งความเว้า) 2 จุด ที่อยู่ติดกันบนส่วนโค้ง และมุมของเส้นตรงที่เชื่อมระหว่างจุดที่ติดกัน ดังนั้นความแตกต่างระหว่างเวกเตอร์ ii_s และ mm_t คำนวณได้โดย

$$d(ii_s, mm_t) = wd\,|D_{ii_s} - D_{mm_t}| + ws + wa\,|A_{ii_s} - A_{mm_t}|$$

โดย ws มีค่าเป็น 0 ในกรณีที่เป็นการเปรียบเทียบกันระหว่างจุดที่มีเครื่องหมายบวก เหมือนกันหรือลบเหมือนกัน และมีค่าเป็น 1 ในกรณีที่เป็นการเปรียบเทียบระหว่างจุดที่มีเครื่องหมายต่างกัน

ดังนั้น ความแตกต่างระหว่างอักขระที่ต้องการรู้จำ กับอักขระต้นแบบเป็นดังนี้

D(II, MM) = g(S, T) / NC

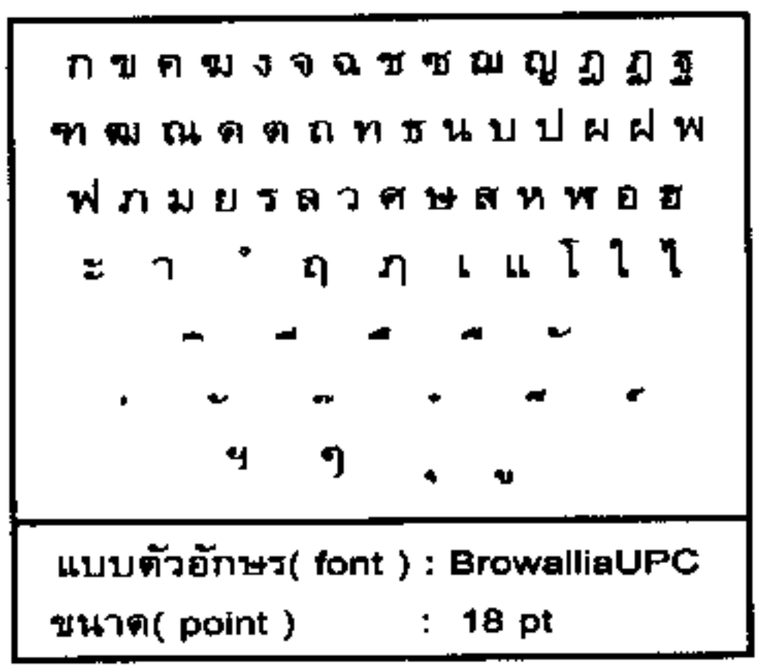
เมื่อ NC = S + T ในกรณีของแกนเวลาสมมาตร

หรือ NC = S ในกรณีของแกนเวลาไม่สมมาตร

อักขระต้นแบบ MM ซึ่งมีค่าความแตกต่าง D(II, MM) ที่น้อยที่สุดจะถือว่าเป็นอักขระต้นแบบที่รู้จำได้ก็ต่อเมื่อค่าความแตกต่างที่น้อยที่สุดนี้มีค่าไม่เกินค่าที่กำหนดไว้ค่าหนึ่ง แต่ถ้าหากว่าค่าความแตกต่างที่น้อยที่สุดนี้มีค่ามากเกินกว่าค่าที่กำหนดไว้ค่าหนึ่ง ก็จะถือว่าอักขระที่ต้องการรู้จำนั้นเป็นอักขระที่ไม่สามารถบอกได้ว่าเป็นอักขระใด

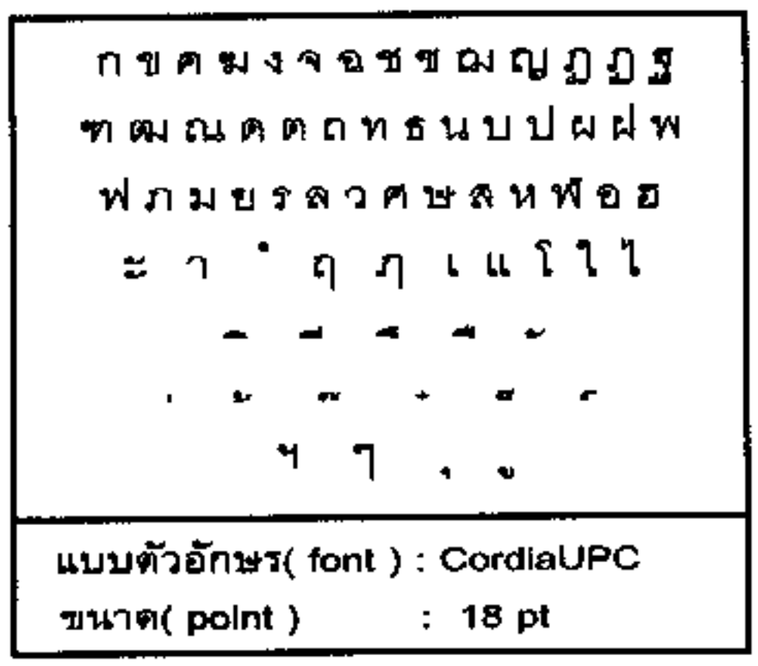
5. ผลการทดลอง

ในการวิจัยนี้ ใช้ตัวพิมพ์อักขระไทยที่มีลักษณะตัวตรงขนาดคงที่ 3 รูปแบบ ดังแสดงตามรูปที่ 22 การทดลองประกอบด้วย 2 ส่วน ส่วนแรกเป็นการทดลองการรู้จำตัวอักขระไทยโดด ๆ อีกส่วนเป็นการรู้จำตัวอักขระที่อยู่ในประโยค สำหรับการรู้จำตัวอักขระโดด ๆ นั้นทำด้วยการพิมพ์ตัวอักขระแต่ละแบบ จำนวน 10 ครั้ง แล้ว scan เข้าสู่คอมพิวเตอร์ด้วยความละเอียดในการ scan 400 dpi และเมตริกซ์ของอินพุทที่ใช้ในการทดลองมีขนาด 60x50 จุด ทำให้มีอักขระที่ใช้ในการทดลองแต่ละรูปแบบ 680 ตัวอักขระ รวม 3 รูปแบบ เป็นตัวอักขระทั้งหมด 2040 ตัว ตัวอักขระที่ถูก scan 5 ตัวแรกถูกใช้ในการเรียนรู้ และ 5 ตัวที่เหลือถูกใช้เป็นอักขระทดสอบ ในการทดลองได้ใช้สมการไดนามิคโปรแกรมมิ่งแบบไม่สมมาตร โดยใช้ค่าความชันเป็น 1 (ซึ่งได้ทดสอบแล้วว่าได้ผลของการรู้จำดีที่สุด ซึ่งสอดคล้องตามงานวิจัย[8])



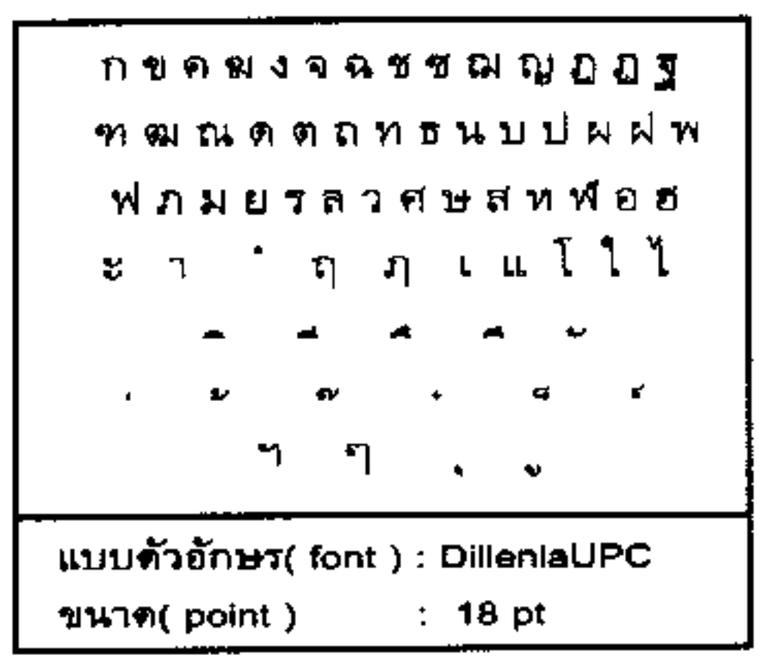
กลุ่มตัวอักขระรูปแบบที่ 1

กวาดตรวจที่ความละเอียด
: 400 dpi
เมตริกซ์ของอักขระ มีขนาด (H x W)
: 60 x 50 จุด



กลุ่มตัวอักขระรูปแบบที่ 2

กวาดตรวจที่ความละเอียด
: 400 dpi
เมตริกซ์ของอักขระ มีขนาด (H x W)
: 60 x 50 จุด



กลุ่มตัวอักขระรูปแบบที่ 3

กวาดตรวจที่ความละเอียด
: 400 dpi
เมตริกซ์ของอักขระ มีขนาด (H x W)
: 60 x 50 จุด

รูปที่ 22

เมื่อนำตัวอักขระ 1020 ตัวมาเรียนรู้ ทำให้ได้ผลของการรู้จำ 100% จึงนำพจนานุกรมที่ได้ไปใช้ในการรู้จำตัวอักขระทดสอบ 1020 ตัวต่อไป ในการทดลองได้ทำทั้งในกรณีที่ใช้เฉพาะความยาวของเวคเตอร์ที่เชื่อมโยงจุดบ่งความนูนความเว้า 2 จุดที่ติดกันของส่วนโค้ง และในกรณีที่ใช้ทั้งความยาว และมุมของเวคเตอร์นี้ในการหาความแตกต่างระหว่างส่วนโค้งและระหว่างอักขระ ซึ่งผลของการทดลองได้แสดงในตารางที่ 3 สำหรับการทดลองการรู้จำอักขระจากประโยคนั้นได้นำอักขระจำนวน 736 ตัว จำนวน 2 รูปแบบที่อยู่ในประโยคทั้งหมดไปทำการ scan จำนวน 1 ครั้ง เพื่อนำไปทดลองการรู้จำ ซึ่งผลของการรู้จำได้แสดงในรูปที่ 23 ก และ ข

	การทดสอบที่ 1	การทดสอบที่ 2
จำนวนอักขระทดสอบรวม 3 รูปแบบ(ตัว)	1020	1020
ความยาวของเวคเตอร์(ใช้/ไม่ใช้)	ใช้	ใช้
มุมของเวคเตอร์(ใช้/ไม่ใช้)	ใช้	ไม่ใช้
จำนวนอักขระต้นแบบ(ตัว)	250	264
จำนวนอักขระที่ไม่รู้จัก(ตัว)	14	18
จำนวนอักขระที่รู้จำผิดพลาด(ตัว)	42	50
จำนวนอักขระที่รู้จำได้ถูกต้อง(ตัว)	964	952
เปอร์เซ็นต์ความถูกต้อง(%)	94.51	93.33

ตารางที่ 3 ผลการทดลองรู้จำตัวอักขระไทยโดดๆ

ผลการทดลองรู้จำอักขระรูปแบบที่ 1 ที่อยู่ในรูปประโยค

เทคนิคการวิเคราะห์เส้นแสดงขอบของอักขระจะถูกนำมาใช้เพื่อค้นหา
ล!กษณะสำค!ญของอักบระแต่ละตัวโดยการใช้รหัสทิศทางแบบลูกโซ่ของฟร!แมน
และความแตกต่!งของทิศทางของเส้นแสดงขอบของอักขระในการตัดแบ่งเส้นแสดง
ขอบของอักขระออกเป!นส่วนโค้งย!อยซ!งประกอบด้วยส่วนโค้งเว้าและส่วนโค้งนูน
จากนั้น ก็จะทำการดึงลักษณะสำคัญของส่วนโค้งย่อยแต่ละส่วนโค้งสำหรับใชฟนการ
เปรียบเทียบส่วนโค้งหาคู่ส่วนโค้งที่คล้ายกันมากที่สุดระหว่างอักขระที่ต้อง!ารรู้จำกับ
อ!กขระต้นแบบเพื่อใชเป็นคู่ส่วนโค้งเริ่มต้นในการเปรียบเทียบหาอักขระต้นแบบที่
เหมือนหรือคล้ายกับอักขระที่ต้องการรู้จำมากที่สุดในขั้นตอนการเปรียบเทียบนิ้น
จะนำเอาเทคนิคการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งมาใช้ช!งเทคนิคการ
เปรียบเทียบดังกล่าวนี้เป!นเทคนิคที่งานวิจิยในอดีตนำมาประยุกตภุช้แล้วได้ผลความ

เทคนิคการวิเคราะห์เส้นแสดงขอบของอักขระจะถูกนำมาใช้เพื่อค้นหา
ลักษณะสำคัญของอักขระแต่ละตัว โดยการใช้รหัสทิศทางแบบลูกโซ่ของฟรีแมน
และความแตกต่างของทิศทางของเส้นแสดงขอบของอักขระในการตัดแบ่งเส้นแสดง
ขอบของอักขระออกเป็นส่วนโค้งย่อยซึ่งประกอบด้วย ส่วนโค้งเว้า และส่วนโค้งนูน
จากนั้นก็จะทำการดึงลักษณะสำคัญของส่วนโค้งย่อยแต่ละส่วนโค้ง สำหรับใช้ในการ
เปรียบเทียบส่วนโค้งหาคู่ส่วนโค้งที่คล้ายกันมากที่สุดระหว่างอักขระที่ต้องการรู้จำกับ
อักขระต้นแบบ เพื่อใช้เป็นคู่ส่วนโค้งเริ่มต้นในการเปรียบเทียบหาอักขระต้นแบบที่
เหมือนหรือคล้ายกับอักขระที่ต้องการรู้จำมากที่สุด ในขั้นตอนการเปรียบเทียบนั้น
จะนำเอาเทคนิคการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งมาใช้ ซึ่งเทคนิคการ
เปรียบเทียบดังกล่าวนี้เป็นเทคนิคที่งานวิจัยในอดีตนำมาประยุกต์ใช้ แล้วได้ผลความ

รูปที่ 23 (ก)

สรุปผลการทดลองรู้จำอักขระรูปแบบที่ 1 ที่อยู่ในรูปประโยค

จำนวนอักขระทั้งหมด	736 ตัว	
สามารถรู้จำอักขระได้	701 ตัว	
คิดเป็น	95.24 %	
รู้จำอักขระผิดพลาด	23 ตัว	คิดเป็น 3.13 %
ไม่รู้จักตัวอักขระ	12 ตัว	คิดเป็น 1.63 %

ผลการทดลองรู้จำอักขระรูปแบบที่ 3 ที่อยู่ในรูปประโยค

เทคนิคการวิเคซะห์เส้นแสดงขอบของอักขระจะถูกนำมาใช้เพื่อค้นหา
ลักษณะสำค!ญของอักขระแต่ระตัวโดยการใช้ !˜ ตทิศทางแบบลูกโซ่ของ!ร!แน
และความแตกต่างขอๆทิศทางของเส้นแสดงขอบของอีกขระในการตัดแบ่งเส้นแสดง
ขอบของอักขระออกเป็นส่วนโค๊งย่อยซึ่งประกอบด้วย!วนโค้งเว้าแซะ!วนโค้งนูน
จากนั้นก็จะทำการดึงลักษ!สำคัญของ!วนโค้งย่อยแต่ละ!วนโค้งสำห!ี ใ ช ฮ นการ
เ!รยบเทียบ!วนโค้งหาคู่!วนโค้งที่คร้ายกันม!กที่สุดระหว่างอักขร !ี ' ต้องการรู้จำกีบ
อ!กขระต้นแบบเพื่อใช้เป็นคู่!วนโค้งเริ่ม ต้นในการเ!รยบเทียบหาอักขระต้นแบบท้ '
เหมือนหรือคล้ายกับอักขระท้ ' ต๊องการรู้จำมากที่สุดในขั้นตอนการเ!รยบเทียบนี้ห
จะนำเอาเทคนิคการเ!รยบเทียบแบบไดนามิคโปรแกรมมิ่งม ! ใช้ช!งเทคนิคการ
เ!รยบเทียบดังกจ่าวนี้เป็นเทตนิคตี่งานวิจัยไหอดีตนำศาประยุกตฎช้แล้วได้ผลความ

เทคนิคการวิเคราะห์เส้นแสดงขอบของอักขระจะถูกนำมาใช้เพื่อค้นหา
ลักษณะสำคัญของอักขระแต่ละตัว โดยการใช้รหัสทิศทางแบบลูกโซ่ของฟรีแมน
และความแตกต่างของทิศทางของเส้นแสดงขอบของอักขระในการตัดแบ่งเส้นแสดง
ขอบของอักขระออกเป็นส่วนโค้งย่อยซึ่งประกอบด้วย ส่วนโค้งเว้า และส่วนโค้งนูน
จากนั้นก็จะทำการดึงลักษณะสำคัญของส่วนโค้งย่อยแต่ละส่วนโค้ง สำหรับใช้ในการ
เปรียบเทียบส่วนโค้งหาคู่ส่วนโค้งที่คล้ายกันมากที่สุดระหว่างอักขระที่ต้องการรู้จำกับ
อักขระต้นแบบ เพื่อใช้เป็นคู่ส่วนโค้งเริ่มต้นในการเปรียบเทียบหาอักขระต้นแบบที่
เหมือนหรือคล้ายกับอักขระที่ต้องการรู้จำมากที่สุด ในขั้นตอนการเปรียบเทียบนั้น
จะนำเอาเทคนิคการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่งมาใช้ ซึ่งเทคนิคการ
เปรียบเทียบดังกล่าวนี้เป็นเทคนิคที่งานวิจัยในอดีตนำมาประยุกต์ใช้ แล้วได้ผลความ

รูปที่ 23 (ข)

สรุปผลการทดลองรู้จำอักขระรูปแบบที่ 3 ที่อยู่ในรูปประโยค

จำนวนอักขระทั้งหมด	736 ตัว		
สามารถรู้จำอักขระได้	654 ตัว	คิดเป็น	88.99 %
รู้จำอักขระผิดพลาด	39 ตัว	คิดเป็น	5.30 %
ไม่รู้จักอักขระ	42 ตัว	คิดเป็น	5.71 %

6. บทสรุป

งานวิจัยนี้เสนอผลของการรู้จำตัวอักขระพิมพ์ไทยหลายรูปแบบที่พิมพ์อยู่ในประโยค การกวาดตรวจหาจุดคำตามแนวนอนและแนวตั้ง ถูกใช้เป็นลักษณะสำคัญในการแยกตัวอักขระออกจากประโยค หลังจากนั้นส่วนนูนและส่วนเว้าของตัวอักขระ ถูกใช้ในการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่ง เพื่อรู้จำตัวอักขระแต่ละตัว

จากผลการทดลองจะเห็นได้ว่าสามารถตัดแยกตัวอักขระที่เกิดการซ้อนทับกันบ่อยครั้ง เช่น พยัญชนะ ฟ ฝ ป ช ซ ฮ ส ศ กับตัวสระวรรณยุกต์ เช่น [illegible] เหล่านี้ ได้ผลดี แต่อย่างไรก็ตาม แนวคิดที่ใช้ในงานวิจัยนี้ ยังไม่สามารถแยกตัวอักขระที่ติดกันในแนวบรรทัด ซึ่งการติดกันเกิดจากการ scan รูปภาพ ในการแยกตัวอักขระที่ติดกันในแนวบรรทัดนี้ อาจจะต้องใช้ข้อมูลความกว้างโดยเฉลี่ยของตัวอักขระในบรรทัดมาช่วย สำหรับการรู้จำตัวอักขระแต่ละตัวนั้น นอกจากใช้ความยาวของจุดบ่งความนูนเว้าที่ติดกัน ในการเปรียบเทียบแบบไดนามิคโปรแกรมมิ่ง ซึ่งใช้ดีในงานวิจัย [8] ก่อนหน้าแล้ว ในงานวิจัยนี้มีการใช้ลักษณะสำคัญเชิงมุมในการเปรียบเทียบด้วย เนื่องจากความคมชัดของตัวอักขระที่อยู่ในประโยคจะไม่ดีนัก ซึ่งผลการรู้จำตัวอักขระได้ผลดีขึ้น สำหรับตัวอักขระที่รู้จำผิดพลาดมักเกิดจาก สัญญานรบกวน (noises) ทำให้จำนวนส่วนโค้งนูนเว้าที่ได้จากตัวอักขระข้อมูลแตกต่างกับตัวอักขระต้นแบบ ดังนั้นจึงควรมีขบวนการที่มีประสิทธิภาพในการ กำจัดสัญญานรบกวนในขั้นตอนของ Preprocessing แต่อย่างไรก็ตาม เนื่องจากอัลกอริทึมที่ใช้ในการรู้จำนี้สามารถเรียนรู้ตัวอักขระเพิ่มเติมได้อย่างง่ายดาย ดังนั้นในการทดลองรู้จำตัวอักขระที่อยู่ในประโยค เมื่อนำตัวอักขระผิดพลาดไปเรียนรู้ เพื่อเพิ่มในฐานข้อมูลตัวอักขระต้นแบบแล้ว จะทำให้ผลการรู้จำดีขึ้นมาก

บรรณานุกรม

Hiranvanichakorn, P., Agui, T. and Nakajima, M. : "A recognition method of Thai characters", Trans. IECE Japan, E 65, 12, pp. 737-744 (Dec. 1982).

Hiranvanichakorn, P., Agui, T. and Nakajima, M. : "Recognition method of Thai characters by using local features", Trans. IECE Japan, E 67, 8, pp. 425-432 (Aug. 1984)

Hiranvanichakorn, P., Agui, T. and Nakajima, M. : "A recognition method of handprinted Thai characters by local features", Trans. IECE Japan, E 68, 2, pp. 83-90 (Feb. 1985).

Hiranvanichakorn, P., Agui, T. and Nakajima, M. : "An on-line recognition method of Thai characters", Trans. IECE Japan, E 68, 9, pp. 594-601 (Nov. 1985).

Hiranvanichakorn, P. : "Recognition of size different Thai characters", Applied Statistics Meeting, 6.

Kimpan, C., Itoh, A. and Kawanishi, K., : "Fine classification of Printed Thai character recognition using the Karhunen-Loeve expansion", IEE Proc. E. Comput & Digital Tech., 1987, 134 (5), pp. 254-264.

บุญธีร์ เครือตราชู และอภิรักษ์ จิราชุสกุล : "การรู้จำตัวพิมพ์อักษรไทย โดยใช้ Counterpropagation Neural Network" การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 18

Hiranvanichakorn, P. and Boonsuwan M. "Recognition of Thai characters", Proceedings of the Symposim on Natural Language Processing in Thailand, pp. 123-166 (Mar. 1993).

ศุภกร รัตนปราการ และ บุญธีร์ เครือตราชู "การวิเคราะห์การตัดกัน และการตัดแยกของอักษรพิมพ์ไทย โดยใช้คุณลักษณะทางแนวตั้งและแนวนอนของฮีสโตแกรม" วารสารสารสนเทศลาดกระบัง Vol.4 No.1, July 1999

Sokoe, H. and Chiba, S. : "Dynamic programming algorithm optimization for spoken word recognition", IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-26, pp 43-49, Feb. 1978.

Sakoe, H. and Chiba, S. : "A dynamic programming approach to continuous speech recognition", in 1971 Proc. 7th ICA, Paper 20 C13, Aug. 1971.