บทที่ 2
อัลกอริทึมที่ใช้ในการประมาณการเคลื่อนที่
(Motion Estimation Algorithm)

ในหัวข้อนี้จะกล่าวถึงทฤษฎีและหลักการพื้นฐานต่าง ๆ ที่เกี่ยวข้อกับการประมาณการเคลื่อนที่ของภาพวิดีโอ ซึ่งเนื้อหาในบทนี้จะกล่าวถึงขั้นตอนในการเปรียบเทียบภาพ 2 ภาพ อัลกอริทึมที่ใช้ในการหาจุดพิกัดที่ต้องเปรียบเทียบแบบต่าง ๆ ฟังก์ชันสำหรับแสดงความแตกต่างของภาพ ค่าผลลัพธ์ที่ได้ และการนำค่าผลลัพธ์กลับมาสร้างเป็นภาพ

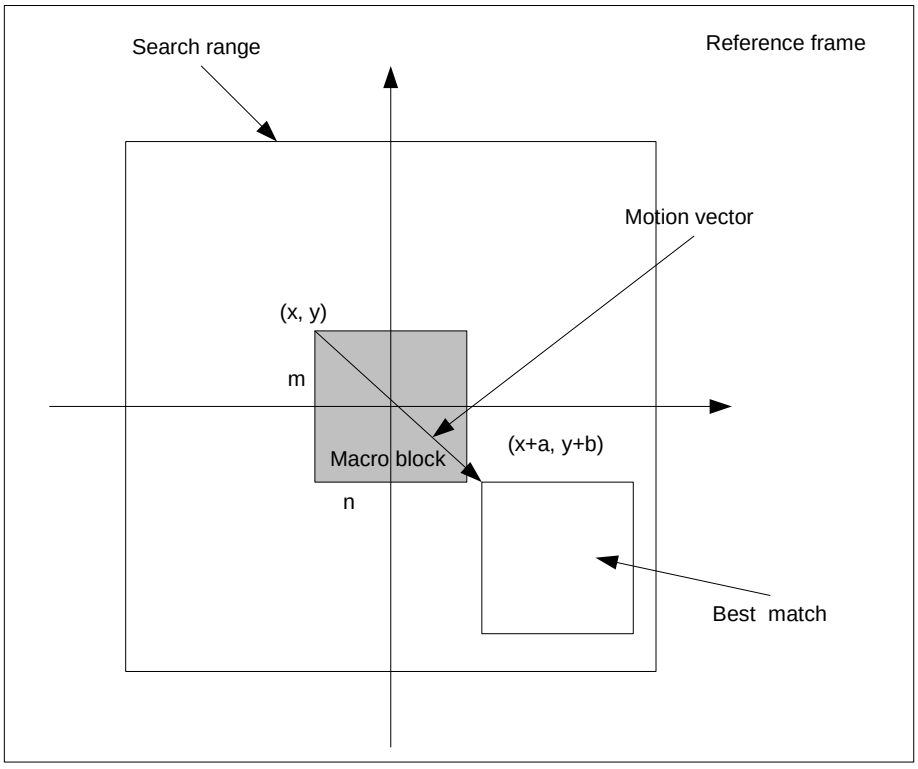
2.1 การประมาณการเคลื่อนที่ (Motion Estimation)

การประมาณการเคลื่อนที่เป็นวิธีการที่ถูกนำมาใช้เพื่อลดขนาดของข้อมูลภาพวิดีโอ โดยใช้พฤติกรรมที่ภาพวิดีโอในลำดับภาพที่ต่อเนื่องกันมีจะการเปลี่ยนน้อยมาก พื้นที่ส่วนใหญ่ของภาพจะยังเหมือนกับภาพก่อนหน้าหรือภาพถัดไปดังรูปที่ 2.1 จะเห็นได้ว่าในทั้ง 3 ภาพนั้นนอกจากบริเวณใบหน้าแล้ว ส่วนอื่น ๆ ในภาพจะไม่มีการเปลี่ยนแปลงใด ๆ จากเหตุผลดังกล่าว จึงได้เกิดแนวคิดในการจัดเก็บภาพวิดีโอทั้งภาพเพียงบางลำดับภาพของวิดีโอ ส่วนภาพวิดีโอนอกเหนือจากนั้นให้จัดเก็บเฉพาะการเปลี่ยนแปลงตำแหน่งภาพในภาพ เมื่อเทียบกับภาพในลำดับใกล้เคียงกันที่มีการจัดเก็บซึ่งเรียกว่าลำดับภาพอ้างอิง ซึ่งเมื่อมีทั้งภาพอ้างอิงและความเปลี่ยนแปลงตำแหน่งในส่วนต่าง ๆ ของภาพ ก็จะสามารถนำมาสร้างเป็นภาพที่ไม่ได้ถูกจัดเก็บไว้ได้ นอกจากการจัดเก็บแบบดังกล่าวแล้วยังมีการจัดเก็บอีกแบบที่ใช้ในมาตรฐานเอ็มเป็ก-1 (MPEG-1) คือจะใช้การจัดเก็บทั้งภาพเป็นบางลำดับภาพ โดยจะมีการเว้นช่วงที่ไม่จัดเก็บเป็นช่วงที่ไม่ยาวมาก เช่น จัดเก็บภาพในลำดับที่ 1 เว้นไป 3 ลำดับ และจัดเก็บใหม่ในลำดับที่ 5 จากนั้นเว้นไปอีก 3 ลำดับ แล้วจึงจัดเก็บลำดับที่ 9 เมื่อต้องการจะถอดรหัสออกมาเป็นลำดับภาพวิดีโอก็ให้คำนวณหาการประมาณการเคลื่อนที่ระหว่างภาพลำดับที่ 1 และ 5 เมื่อได้ทิศทางการเปลี่ยนแปลงระหว่าง 1 กับ 5 แล้ว ก็สามารถจะนำไปคำนวณหาการเปลี่ยนแปลงระหว่างภาพที่ 1 กับภาพที่ 2 หรือ 3 หรือ 4 แล้วนำมาสร้างเป็นภาพเหล่านั้นได้

รูปที่ 2.1 ลำดับภาพที่ 1, 2 และ 3 ของภาพวิดีโอมิสอเมริกา (MISS_AM)

2.2 อัลกอริทึมการเปรียบเทียบทั้งกลุ่ม (Block Matching Algorithm : BMA)

เนื่องจากภาพวิดีโอจะมีลักษณะที่ไม่สามารถแบ่งแยกเป็นวัตถุต่าง ๆ ได้ชัดเจน การจะเปรียบเทียบความเปลี่ยนแปลงของวัตถุต่าง ๆ ระหว่างภาพ 2 ภาพจึงทำได้ยาก จึงต้องใช้วิธีแบ่งภาพออกเป็นภาพย่อย ๆ ขนาดเล็ก (macro block) ที่มีขนาดเท่ากัน และหาการเปลี่ยนแปลงตำแหน่งของภาพย่อย ๆ เหล่านี้แทน วิธีการที่เป็นที่นิยมสำหรับนำมาใช้เพื่อหาการเปลี่ยนแปลงตำแหน่งของภาพย่อยเหล่านี้คือการเปรียบเทียบแบบกลุ่ม ซึ่งเป็นการเปรียบเทียบภาพย่อยแบบภาพต่อภาพที่มีขนาดเดียวกัน วิธีการเปรียบเทียบแบบกลุ่มนี้จะใช้ในมาตรฐานการบีบอัดวิดีโอ เช่น เอ็มเป็ก 1 (MPEG-1), เอ็มเป็ก 2 (MPEG-2), เอ็มเป็ก 4 (MPEG-4), เอช 261 (H.261) และ เอช 263 (H.263)



รูปที่ 2.2 การประมาณการเคลื่อนที่ด้วยวิธีเปรียบเทียบกลุ่ม

รูปที่ 2.2 แสดงให้เห็นถึงกระบวนการพื้นฐานของการประมาณการเคลื่อนที่โดยใช้วิธีการเปรียบเทียบกลุ่มโดยจะประกอบไปด้วยลำดับภาพอ้างอิง ซึ่งเป็นลำดับภาพที่อยู่ก่อนหน้า และภาพย่อยขนาด m x n ซึ่งเป็นส่วนหนึ่งของลำดับภาพในปัจจุบัน ภาพย่อยของลำดับภาพที่แบ่งออกมานี้เรียกว่ามาโครบล็อค (macro block) จุดประสงค์ของการประมาณการเคลื่อนที่ด้วยวิธีนี้ คือ การหาภาพย่อยขนาด m x nในลำดับภาพอ้างอิงที่มีความใกล้เคียงกับภาพย่อยขนาด m x n จากลำดับภาพปัจจุบัน รวมทั้งหาพฤติกรรมการเปลี่ยนแปลงของภาพย่อยนั้นในลำดับภาพปัจจุบัน ซึ่งโดยทั่วไปนั้นจะกำหนดให้ m = n ถ้ากำหนดให้ลำดับภาพปัจจุบันเป็นลำดับภาพที่เวลา t ลำดับภาพอ้างอิงที่นำมาใช้สามารถใช้ได้ทั้งลำดับภาพในช่วงเวลาที่ผ่านมาแล้ว t - p ซึ่งเรียกว่าการประมาณการเคลื่อนที่แบบไปข้างหน้า (forward motion estimation) หรือลำดับภาพในเวลาต่อไปข้างหน้า t - q ซึ่งเรียกว่าการประมาณการเคลื่อนที่แบบย้อนหลัง (backward motion estimation) ก็ได้ เมื่อ p และ q เป็นหน่วยวัดเวลาใด ๆ ทั้งนี้ขึ้นอยู่กับความเหมาะสมของการใช้งาน ซึ่งส่วนมากจะใช้ในลักษณะการประมาณการเคลื่อนที่แบบไปข้างหน้าเสียเป็นส่วนใหญ่ เนื่องจากการบันทึกภาพจากกล้องวิดีโอจะต้องเป็นไปตามลำดับก่อนหลัง จึงไม่สามารถนำลำดับภาพถัดไปมาใช้ได้ นอกจากการเข้ารหัสจากไฟล์ข้อมูลจึงจะสามารถใช้แบบย้อนหลังได้

การอ้างอิงตำแหน่งของภาพย่อยจะใช้พิกัด (x, y) ของมุมบนซ้าย ในการอ้างอิง ในการค้นหาจะต้องมีการกำหนดขอบเขตในการค้นหา (search range) เนื่องจากการค้นหาจากลำดับภาพอ้างอิงทั้งภาพจะใช้เวลามากเกินความจำเป็น เพราะองค์ประกอบต่าง ๆ ในลำดับภาพของวิดีโอนั้นจะมีการเปลี่ยนแปลงจากตำแหน่งเดิมไม่มากนัก ดังนั้นในการค้นหาจะทำการกำหนดขอบเขตโดยรอบของตำแหน่งพิกัดของภาพย่อยในลำดับภาพปัจจุบัน โดยการกำหนดขอบเขตนั้น จะต้องกำหนดขอบเขตค้นหาในแนวตั้ง V และขอบเขตค้นหาในแนวนอน H ขึ้นมา ซึ่งมาตรฐานการเข้ารหัสวิดีโอส่วนใหญ่นั้น ค่า V และ H จะกำหนดให้เท่ากัน ขอบเขตการค้นหานั้นเรียกว่าพื้นที่ค้นหา (search area) ในกรณีของทั่วไปของมาตรฐาน MPEG และ H.261/H.263 นั้นจะกำหนดให้ V = H = 6 และ m = n = 16 ถ้ากำหนดให้ (x+a, y+b) เป็นพิกัดของภาพย่อยที่มีความใกล้เคียงที่สุดในลำดับภาพอ้างอิง เวคเตอร์จาก (x, y) ไปยัง (x+a, y+b) คือเวคเตอร์การเคลื่อนที่ (motion vector) ของภาพย่อย (x, y) ค่าเวคเตอร์นี้คือสิ่งที่ใช้แสดงการเปลี่ยนแปลงระหว่างลำดับภาพในช่วงเวลาที่แตกต่างกัน ซึ่งเมื่อใช่ค่าเวคเตอร์นี้ร่วมกับลำดับภาพอ้างอิง ก็จะสามารถสร้างภาพที่ใกล้เคียงกับภาพปัจจุบัน

สำหรับวิธีการที่นำมาใช้เพื่อเปรียบเทียบความเหมือนหรือความแตกต่างของภาพย่อยนั้นเรียกว่าคอสท์ฟังก์ชัน (cost function) นิยมใช้วิธีเอ็มเอดี (MAD : Mean Absolute Difference) เป็นส่วนมาก เนื่องจากมี การคำนวณไม่ซับซ้อนมาก ซึ่งมีรูปแบบสมการดังสมการที่ 2.1

$$MAD(dx, dy) = \frac{1}{mn} \sum_{i=\frac{-n}{2}}^{\frac{n}{2}} \sum_{i=\frac{-m}{2}}^{\frac{m}{2}} \left| F(i, j) - G(i + dx, j + dy) \right| \quad (2.1)$$

เมื่อ $F(i, j)$ คือ มาโครบล็อคขนาด m x n ของเฟรมที่กำลังบีบอัด

$G(i, j)$ คือ มาโครบล็อคขนาด m x n ของเฟรมอ้างอิง

(dx, dy) คือ โมชันเวคเตอร์ของพื้นที่ที่กำลังเปรียบเทียบ

ค่า MAD ที่ได้ออกมาแสดงถึงค่าความแตกต่างระหว่างภาพย่อยที่นำมาเปรียบเทียบกัน ดังนั้นในการเปรียบเทียบกลุ่มโดยใช้ MAD เป็นคอสท์ฟังก์ชัน ภาพย่อยในลำดับภาพอ้างอิงที่นำมาเปรียบเทียบแล้วที่มีค่า MAD น้อยที่สุด หรือมีค่าน้อยกว่าขอบเขตที่กำหนด จะถือว่าเป็นภาพย่อยที่มีความใกล้เคียงมากที่สุด

การเปรียบเทียบภาพย่อยนั้นใช้วิธีปรับเปลี่ยนค่า dx, dy ไปเป็นค่าพิกัดต่าง ๆ ภายในพื้นที่ค้นหาซึ่งวิธีที่ใช้กันในช่วงแรก ๆ นั้นจะใช้วิธีที่เรียกว่าการค้นหาแบบเต็มพื้นที่ (Full Search : FS) ซึ่งเป็นการค้นหาในทุก ๆ พิกัดตำแหน่งภายในพื้นที่ค้นหาเป็นวิธีที่ให้ผลลัพธ์ที่ถูกต้องที่สุด แต่เนื่องจากการที่ต้องเปรียบเทียบในทุก ๆ พิกัดตำแหน่งภายในพื้นที่ค้นหาจึงทำให้ต้องใช้เวลาในการคำนวณมาก นอกจากนี้การคำนวณค่า MAD ยังเป็นการเปรียบเทียบในลักษณะจุดต่อจุดในทั้งภาพย่อย ซึ่งถือเป็นการคำนวณที่ใช้เวลามากอยู่แล้ว จึงทำให้การคำนวณหาเวคเตอร์การเคลื่อนที่ด้วยวิธีการนี้ไม่เป็นที่นิยม เนื่องจากใช้เวลานาน และใช้ทรัพยากรที่สูง

2.3 อัลกอริทึมการเปรียบเทียบทั้งกลุ่มแบบรวดเร็ว (Fast Block Matching Algorithm : FBMA)

เนื่องจากการเปรียบเทียบพิกัดเพื่อหาเวคเตอร์การเคลื่อนที่ด้วยวิธีค้นหาแบบเต็มพื้นที่นั้นใช้เวลานาน จึงไม่เหมาะสมสำหรับการใช้งานจริง ๆ จึงได้มีการพัฒนาอัลกอริทึมสำหรับค้นหาตำแหน่งของภาพย่อยภายในพื้นที่ค้นหาที่มีความใกล้เคียงกับภาพย่อยที่นำมาหาเวคเตอร์การเคลื่อนที่ โดยจะใช้วิธีเลือกเปรียบเทียบเป็นบางตำแหน่ง อย่างเป็นขั้นเป็นตอน และใช้ความสัมพันธ์ของผลลัพธ์จากขั้นตอนก่อนหน้าในการเลือกจุดพิกัดที่ต้องทำการเปรียบเทียบในขั้นตอนต่อไป ซึ่งผลลัพธ์ในด้านความถูกต้องนั้นจะมีความคลาดเคลื่อนจากวิธีค้นหาแบบเต็มพื้นที่อยู่ในระดับหนึ่ง ขึ้นอยู่กับวิธีการค้นหาที่นำมาใช้ ตัวอย่างอัลกอริทึมในกลุ่มนี้ที่นิยมใช้กันทั่วไป คือ การค้นหาแบบ 3 ขั้นตอน (Three-Step Search : 3SS) ซึ่งมีจำนวนจุดที่ต้องเปรียบเทียบน้อยกว่าการค้นหาแบบเต็มพื้นที่มาก ดังตารางที่ 2.1

ตารางที่ 2.1 แสดงจำนวนพิกัดที่ต้องเปรียบเทียบระหว่าง 2 อัลกอริทึมที่ความละเอียดที่แตกต่างกัน โดยกำหนดขนาดของภาพย่อย 16 x 16 พิกเซล และระยะค้นหา 8 พิกเซล

อัลกอริทึม \ ขนาดของภาพ	QCIF (176 X 144)	CIF (352 X 288)
Full Search	28611	114444
Three-Step Search	2673	10692

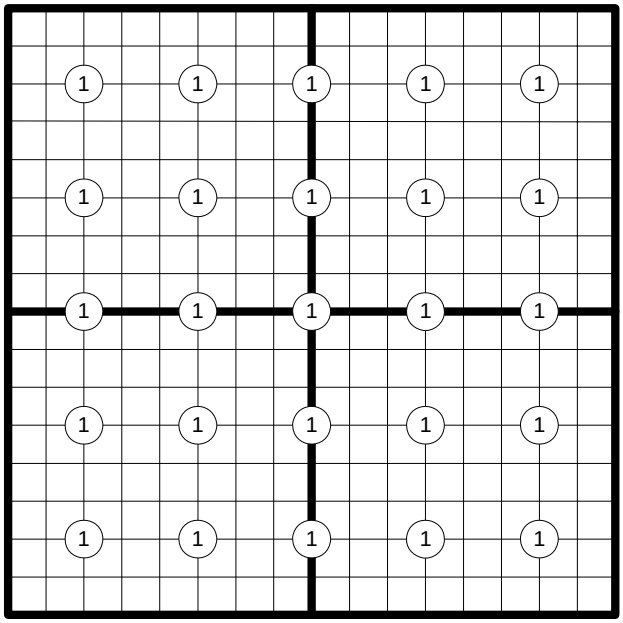
จากตารางที่ 2.1 จะเห็นได้ว่าการค้นหาแบบ 3 ขั้นตอนมีจำนวนพิกัดที่ต้องเปรียบเทียบเพียง 9.34% ของการค้นหาแบบเต็มพื้นที่ ซึ่งทำให้ความเร็วในการประมาณการเคลื่อนที่สูงขึ้นเป็นอย่างมาก ด้วยเหตุผลดังกล่าวจึงทำให้เกิดการพัฒนาอัลกอริทึมในกลุ่มนี้ออกมาเป็นจำนวนมาก แต่ละอัลกอริทึมก็มีข้อดี และข้อเสียที่แตกต่างกันไป นอกจากนี้ยังมีการพัฒนาอัลกอริทึมที่เหมาะสมสำหรับสภาพแวดล้อมการทำงานบางอย่างออกมาอีกด้วย ตัวอย่างเช่น การค้นหาแบบ 2 ขั้นตอนเป็นอัลกอริทึมที่เหมาะสมสำหรับการนำมาสร้างเป็นวงจรดิจิตอล เพื่อนำไปสร้างเป็นหน่วยประมวลผลเฉพาะงานขึ้นมา

2.4 อัลกอริทึมการค้นหาแบบ 2 ขั้นตอน (Two-Step Search : 2SS)

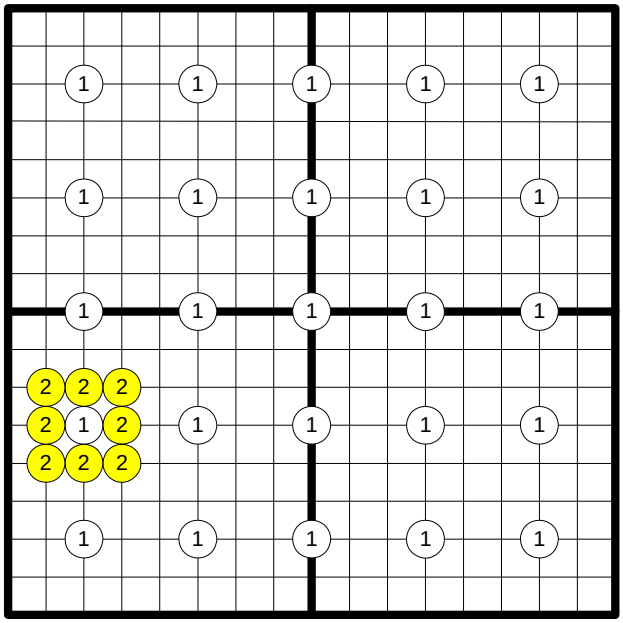
การค้นหาแบบ 2 ขั้นตอนเป็นอัลกอริทึมในกลุ่มของการค้นหาแบบรวดเร็ว จุดมุ่งหมายของการพัฒนาอัลกอริทึมนี้เน้นในด้านของความถูกต้องของผลลัพธ์ ซึ่งมีค่าความถูกต้องสูงใกล้เคียงกับการค้นหาแบบเต็มพื้นที่ และการคำนวณหาตำแหน่งพิกัดที่ต้องเปรียบเทียบในแต่ละขั้นตอนที่ใช้เพียงคณิตศาสตร์พื้นฐานในการคำนวณ หลักการพื้นฐานของอัลกอริทึมนี้คือ การแบ่งการค้นหาออกเป็น 2 ขั้นตอน ขั้นตอนแรกเป็นการค้นหาแบบกระจายทั่วพื้นที่ค้นหา เมื่อได้ค่าที่ดีที่สุดแล้ว จึงทำการค้นหาอย่างละเอียดในขั้นตอนต่อไป ผลลัพธ์ที่ได้จากขั้นตอนที่ 2 คือ เวคเตอร์การเคลื่อนที่ ซึ่งขั้นตอนการทำงานอย่างละเอียดของอัลกอริทึมการค้นหาแบบ 2 ขั้นตอน เมื่อกำหนดให้ระยะค้นหาเท่ากับ 8 พิกเซล มีดังต่อไปนี้

1. กำหนดจุดเริ่มต้น ซึ่งเป็นตำแหน่งเดิมของของภาพย่อยที่นำมาเปรียบเทียบให้เป็นพิกัด (0, 0) จากนั้นทำการหาค่า MAD ของทุก ๆ พิกัด (3a, 3b) เมื่อ a, b มีค่าเป็น -2, -1, 0, 1, 2 หาตำแหน่งพิกัด (x, y) ที่มีค่า MAD น้อยที่สุด ดังรูปที่ 2.3

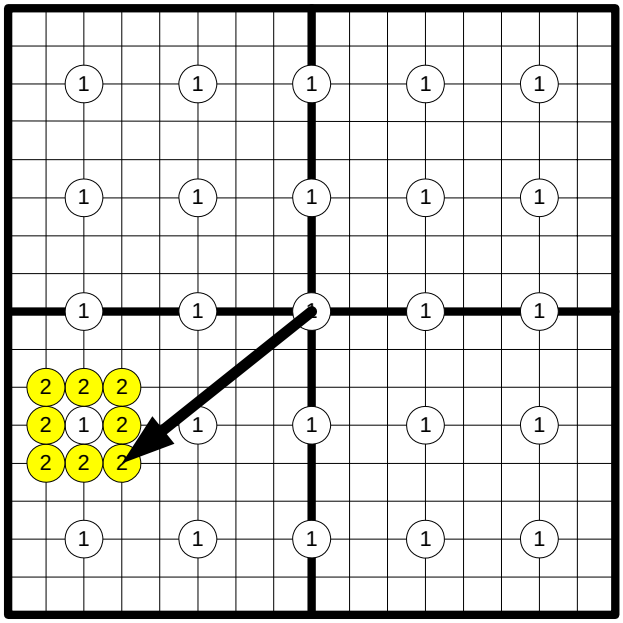
2. จากค่าพิกัด (x, y) ที่ได้จากขั้นตอนที่ 1 ทำการหาค่า MAD ของทุก ๆ ภาพย่อยที่อยู่บนพิกัด (x+a, y+b) เมื่อ a, b มีค่าเป็น -1, 0, 1 ดังรูป 2.4 ซึ่งค่าพิกัด (x+a, y+b) ที่มีค่า MAD น้อยที่สุด คือค่าเวคเตอร์การเคลื่อนที่ของอัลกอริทึมนี้ ดังรูป 2.5



รูปที่ 2.3 การเปรียบเทียบพิกัดในขั้นตอนที่ 1 ของ 2SS



รูปที่ 2.4 การเปรียบเทียบพิกัดในขั้นตอนที่ 2 ของ 2SS



รูปที่ 2.5 เวคเตอร์การเคลื่อนที่ของ 2SS

การปรับเปลี่ยนตำแหน่งพิกัดในแต่ละขั้นตอน สามารถปรับเปลี่ยนได้ตามความเหมาะสมของระยะการค้นหา และการใช้งาน เนื่องจากอัลกอริทึมนี้ไม่ได้มีการกำหนดสมการสำหรับหาตำแหน่งที่แน่นอน ในแต่ละค่าระยะการค้นหา จำนวนพิกัดที่ต้องเปรียบเทียบของอัลกอริทึมนี้ ตามการกำหนดตำแหน่งตามตัวอย่างข้างต้น จะพบว่าจะมีจำนวนพิกัดที่ต้องเปรียบเทียบเป็นจำนวน 11.76% ของการค้นหาแบบเต็มพื้นที่ ดังตารางที่ 2.2

ตารางที่ 2.2 แสดงจำนวนพิกัดที่ต้องเปรียบเทียบระหว่างอัลกอริทึมการค้นหาแบบ 2 ขั้นตอน และการค้นหาแบบเต็มพื้นที่ โดยกำหนดขนาดของภาพย่อย 16 x 16 พิกเซล และระยะค้นหา 8 พิกเซล

อัลกอริทึม \ ขนาดของภาพ	QCIF (176 X 144)	CIF (352 X 288)
Full Search	28611	114444
Two-Step Search	3366	13464

จากลักษณะของอัลกอริทึมที่ใช้การคำนวณหาตำแหน่งแบบพื้นฐาน คือ บวก ลบ และคูณ จึงทำให้มีผู้นำเสนอการนำเอาอัลกอริทึมนี้ไปใช้สร้างเป็นหน่วยประมวลผลเฉพาะทางสำหรับประมาณการเคลื่อนที่ โดยมีจุดมุ่งหมายเพื่อนำไปใช้ในอุปกรณ์มือถือ เพราะการคำนวณที่ไม่ซับซ้อนมาก จึงทำให้วงจรที่ออกมามีขนาดไม่ใหญ่มาก จึงสามารถนำไปประยุกต์ใช้ในอุปกรณ์ขนาดเล็กที่มีข้อจำกัดด้านขนาดและพลังงานได้

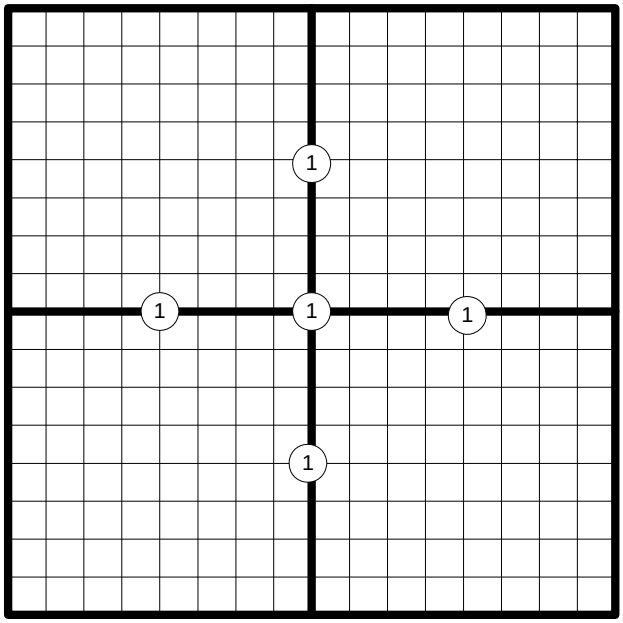
2.5 อัลกอริทึมการค้นหา 1 มิติเป็นลำดับขั้นแบบขนาน (Parallel Hierarchical One-Dimensional Search : PHODS)

อัลกอริทึมการค้นหา 1 มิติเป็นลำดับขั้นแบบขนาน เป็นอัลกอริทึมที่มีแนวคิดแตกต่างจากการหาเวคเตอร์โดยปกติ โดยใช้การค้นหาในแบบมิติเดียว มาหาผลลัพธ์ซึ่งเป็นเวคเตอร์แบบ 2 มิติ เวคเตอร์ที่เป็นผลลัพธ์ของอัลกอริทึมนี้ส่วนใหญ่จะไม่มีการเปรียบเทียบค่าคอสต์ฟังก์ชันที่ตำแหน่งดังกล่าวเลย จากเวคเตอร์ 1 มิติ 2 เวคเตอร์เป็นเวคเตอร์ในทิศทางแนวนอน 1 เวคเตอร์ และเป็นเวคเตอร์ในทิศทางแนวตั้ง 1 เวคเตอร์ ถูกนำมารวมกันเพื่อสร้างเป็นเวคเตอร์ 2 มิติในขั้นตอนสุดท้าย ในการขั้นตอนการเปรียบเทียบจะเป็นการคำนวณหาตำแหน่งบนแต่ละแกน x และ y แยกจากกันในลักษณะที่เป็นมิติเดียว ก่อนจะรวมกันเป็นเวคเตอร์ 2 มิติ ตามสมการที่ 2.2

$$(x,0)+(0,y)=(x,y) \tag{2.2}$$

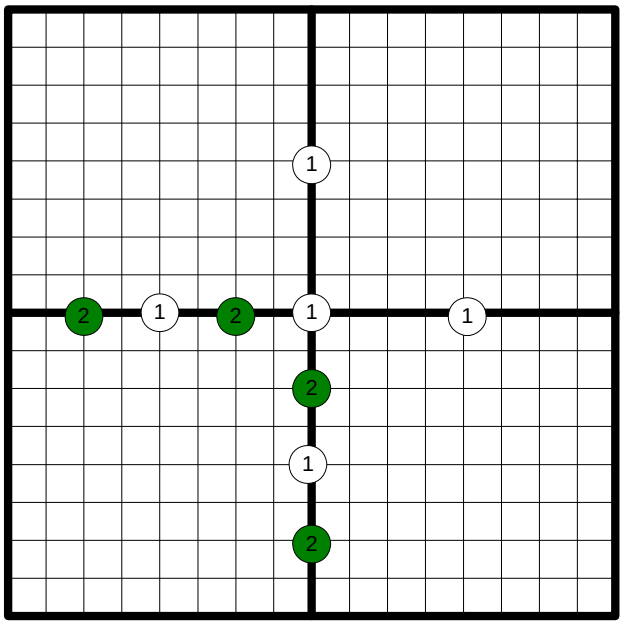
ขั้นตอนในการหาเวคเตอร์การเคลื่อนที่ด้วยวิธีการค้นหา 1 มิติเป็นลำดับขั้นแบบขนานมีดังต่อไปนี้

1. เริ่มต้นจากการคำนวณหาความแตกต่างที่จุดกึ่งกลางของพื้นที่ (0, 0) จากนั้นกำหนดให้เป็นค่าที่ดีที่สุด จากนั้นทำการกำหนดจุดต่อไปที่ต้องเปรียบเทียบ โดยจุดบนแกน y คือ $\left(0,\frac{P}{2}\right)$ และ $\left(0,-\frac{P}{2}\right)$ และจุดบนแกน x คือ $\left(\frac{P}{2},0\right)$ และ $\left(-\frac{P}{2},0\right)$ จากนั้นคำนวณหาจุดที่ดีที่สุดด้วยคอสต์ฟังก์ชัน ซึ่งในที่นี้เลือกใช้ MAD แล้วทำการเลือกจุดที่ดีที่สุดในแกน x และแกน y โดยเทียบรวมกับจุดกึ่งกลางด้วย ดังรูปที่ 2.6



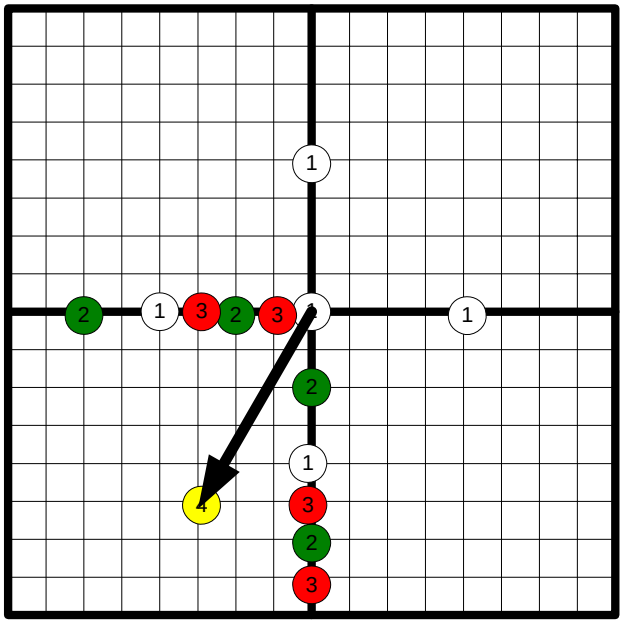
รูปที่ 2.6 แสดงขั้นตอนที่ 1 ของวิธีการ PHODS

2. เมื่อได้จุดที่ดีที่สุดบนทั้ง 2 แกนแล้ว จากนั้นกำหนดจุดต่อไปที่ต้องคำนวณ MAD บนแกน x และ y แกนละ 2 จุด โดยวัดระยะจากจุดที่ดีที่สุดทั้ง 2 จุดออกไปครึ่งหนึ่งของระยะเดิม จะได้เป็นข้างล่ะ $\frac{P}{4}$ จากนั้นคำนวณหาค่า MAD ของแต่ละจุดบนแกน x และ y จากนั้นเลือกค่าที่ดีที่สุด บนแต่ละแกนจากทั้ง 3 จุด รวมจุดเดิมด้วย จะได้จุดที่ดีที่สุด 2 จุดบนแต่ละแกน ดังรูปที่ 2.7



รูปที่ 2.7 แสดงขั้นตอนที่ 2 ของวิธีการ PHODS

3. กำหนดจุดที่ดีที่สุดบนแต่ละแกนเป็นจุดเริ่มต้น จากนั้นกำหนดจุดเปรียบเทียบบนทั้ง 2 แกนอีกแกนละ 2 จุดโดยลดระยะจากจุดเริ่มต้นไปครึ่งหนึ่ง ซึ่งจะได้เป็น $\frac{P}{8}$ จากนั้นทำซ้ำตามขั้นตอนที่ผ่านมาแล้ว จนกระทั่งระยะเป็น 1 จะถือว่าเป็นรอบสุดท้าย ซึ่งจะได้จุดที่ดีที่สุดบนแกน x และ y นำค่า x จากจุดที่ดีที่สุดเป็นแกน x และค่า y จากจุดที่ดีที่สุดบนแกน y มาจับเป็นคู่ลำดับ ก็จะได้ตำแหน่งที่เหมาะสมที่สุด ซึ่งเป็นเวคเตอร์การเคลื่อนที่ของวิธีการนี้ ดังรูปที่ 2.8



รูปที่ 2.8 เวคเตอร์ความเคลื่อนที่ของวิธีการ PHODS

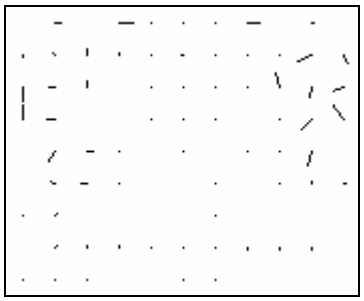
การหาเวคเตอร์ความเคลื่อนที่ด้วยวิธีการนี้มีจุดเด่นในด้านของความเร็ว เนื่องจากมีจำนวนพิกัดที่ต้องทำการเปรียบเทียบน้อยมาก คือประมาณ 4.5% ของการค้นหาแบบเต็มพื้นที่ ดังตารางที่ 2.3 ด้วยลักษณะเด่นในด้านความเร็วอันเนื่องมากจากมีจำนวนพิกัดที่ต้องเปรียบเทียบน้อยของอัลกอริทึมนี้ และมีการคำนวณที่ไม่ซับซ้อน รวมทั้งลักษณะของอัลกอริทึมที่มีลักษณะจึงมีความเหมาะสมที่จะนำไปสร้างเป็นวงจรดิจิตอล เพื่อใช้เป็นหน่วยประมวลผลเฉพาะทางสำหรับการประมาณการเคลื่อนที่

ตารางที่ 2.3 แสดงจำนวนพิกัดที่ต้องเปรียบเทียบระหว่างอัลกอริทึมการค้นหา 1 มิติเป็นลำดับขั้นแบบขนานและการค้นหาแบบเต็มพื้นที่

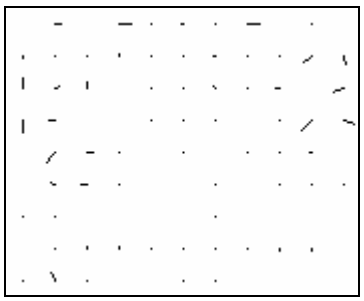
อัลกอริทึม \ ขนาดของภาพ	QCIF (176 X 144)	CIF (352 X 288)
Full Search	28611	114444
PHODS	1287	5148

2.6 การเปรียบเทียบคุณภาพของอัลกอริทึม

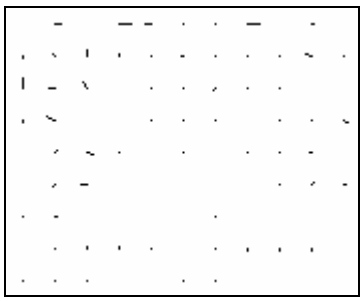
ในการเปรียบเทียบข้อดีและข้อเสียของอัลกอริทึมที่ใช้ในการเลือกตำแหน่งสำหรับเปรียบเทียบภาพย่อยนั้น หลัก ๆ จะมีด้วยกัน 2 ด้าน คือในด้านของความเร็วที่ใช้ในการคำนวณ และในด้านของความคลาดเคลื่อนของผลลัพธ์ ซึ่งเป็นเหตุมาจากการที่วิธีการสุ่มเลือกเปรียบเทียบที่แตกต่างกัน ทำให้ค่าที่ได้แตกต่างกัน ดังรูปที่ 2.9



(FS)



(2SS)



(PHODS)

รูปที่ 2.9 เวคเตอร์ของลำดับภาพ miss_am ระหว่างภาพที่ 1 และ 4 ที่แตกต่างกันตามอัลกอริทึม

การเปรียบเทียบในด้านของความเร็วนั้นมีวิธีวัดที่นิยมใช้ด้วยกัน 2 แบบ คือ การวัดแบบจับเวลาที่ใช้คำนวณจริง และการวัดจากจำนวนพิกัดที่ต้องเปรียบเทียบในพื้นที่ค้นหาขนาดเท่ากัน วิธีการวัดแบบจับเวลาจริงนั้น ค่าเวลาจะเปลี่ยนแปลงไปตามระบบที่ใช้ประมวลผล จึงต้องมีอัลกอริทึมหลักสำหรับใช้อ้างอิง 1 อัลกอริทึม เพื่อใช้ในการอ้างอิงความเร็ว ซึ่งส่วนมากจะใช้

วิธีการคนหาแบบเต็มพื้นที่ในการอ้างอิง การแสดงค่าของเวลาที่ใช้นั้น อาจจะแสดงเป็นเวลาตรง ๆ หรือแสดงเป็นอัตราส่วนเมื่อเทียบกับเวลาที่อัลกอริทึมอ้างอิงใช้ก็ได้

ในส่วนของค่าความถูกต้องนั้นจะใช้การอ้างอิงกับวิธีการเปรียบเทียบแบบเต็มพื้นที่ ซึ่งยอมรับกันว่าเป็นวิธีที่ให้ผลลัพธ์เฉลี่ยดีที่สุด สำหรับวิธีการที่นำมาใช้วัดความถูกต้องของผลลัพธ์นั้นจะใช้การคำนวณหาค่า PSNR (Peak Signal to Noise Ratio) ซึ่งมีรูปแบบสมการดังนี้

$$PSNR = 10\log_{10}\left[\frac{(Peak-to-peak-value-of-the-original-data)^2}{MSE}\right] \qquad (2.3)$$

ค่า MSE (Mean Square Error) สามารถคำนวณได้จากสมการ

$$MSE = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[O(i,j)-R(i,j)]^2 \qquad (2.4)$$

เมื่อ O(i, j) คือภาพดั้งเดิม

R(i, j) คือภาพขนาด N X M ที่สร้างขึ้นมาใหม่จากค่าเวคเตอร์การเคลื่อนที่ และภาพอ้างอิงที่เป็นภาพที่สมบูรณ์ ไม่สามารถใช้ภาพอ้างอิงที่ถูกสร้างขึ้นมาจากเวคเตอร์มาคำนวณค่าความผิดพลาดได้

ในส่วนของค่า Peak to peak value of the original data นั้นขึ้นอยู่กับค่าสีของพิกเซล เช่นในกรณีที่เป็นค่า 8 หรือ 24 บิต ค่าที่เป็นไปได้ในแต่ละไบต์ คือ ต่ำสุด 0 และสูงสุด 255 ดังนั้นจึงใช้ค่า 255 ซึ่งเป็นผลต่างของค่าทั้ง 2 ในสมการที่ 2.3 หน่วยวัดของค่า PSNR คือเดซิเบล (dB) อัลกอริทึมที่มีค่าสูงกว่าจะมีความถูกต้องมากกว่าอัลกอริทึมที่มีค่าน้อย

ในการเปรียบเทียบในด้านความถูกต้องนั้นจะต้องใช้การเปรียบเทียบจากข้อมูลวิดีโอที่มีลักษณะการเคลื่อนที่ และการเปลี่ยนแปลงในภาพที่แตกต่างกัน ซึ่งในวิทยานิพนธ์นี้จะใช้ลำดับภาพมาตรฐานจำนวน 5 ชุด เพื่อเปรียบเทียบในด้านของของความถูกต้อง ซึ่งประกอบไปด้วย akiyo, coastguard, container, mobile และ tempete ลำดับภาพทั้งหมดจะเป็นมาตรฐาน CIF เนื่องจากมีขนาดของภาพที่ใหญ่พอสมควร ทำให้เห็นความแตกต่างของตัวเลขได้ชัดเจน

2.6.1 Akiyo

ลำดับภาพ akiyo เป็นลำดับภาพที่มีพื้นหลังนิ่ง มีการเปลี่ยนแปลงระหว่างภาพน้อยมากจะเกิดขึ้นส่วนใหญ่ที่กลางภาพ

รูปที่ 2.10 ลำดับภาพที่ 1 ของ akiyo

ตารางที่ 2.4 แสดงค่า PSNR ระหว่างลำดับภาพต่าง ๆ ในลำดับมาตรฐาน akiyo

Ref. frame->Recon. frame	FS	3SS	2SS	PHODS
1->2	45.827	45.827	45.827	45.827
3->4	46.296	46.296	46.296	46.296
5->6	40.955	40.955	40.050	39.397
7->8	42.310	42.310	42.310	42.101
1->4	44.614	44.614	44.614	44.614
1->8	38.690	38.302	37.715	37.529

2.6.2 Coastguard

ลำดับภาพ coastguard เป็นลำดับภาพที่มีพื้นหลังเคลื่อนไปทางขวา มีการเปลี่ยนแปลงระหว่างภาพที่ค่อนข้างเร็วในทิศทางที่ไปด้านขวาไปซ้าย

รูปที่ 2.11 ลำดับภาพที่ 1 ของ coastguard

ตารางที่ 2.5 แสดงค่า PSNR ระหว่างลำดับภาพต่าง ๆ ในลำดับมาตรฐาน coastguard

Ref. frame->Recon. frame	FS	3SS	2SS	PHODS
1->2	30.476	30.476	30.441	29.989
3->4	30.690	30.690	30.674	30.599
5->6	30.818	30.818	30.838	30.430
7->8	30.304	30.304	30.273	30.175
1->4	25.503	25.457	25.457	23.993
1->8	22.3066	21.891	21.972	21.311

2.6.3 Container

ลำดับภาพ container เป็นลำดับภาพที่พื้นหลังมีการเปลี่ยนแปลงในลักษณะขึ้นลงเป็นบางส่วน มีการเปลี่ยนแปลงระหว่างภาพที่ค่อนข้างช้าในทิศทางที่ไปด้านซ้ายไปขวา

รูปที่ 2.12 ลำดับภาพที่ 1 ของ container

ตารางที่ 2.6 แสดงค่า PSNR ระหว่างลำดับภาพต่าง ๆ ในลำดับมาตรฐาน container

Ref. frame->Recon. frame	FS	3SS	2SS	PHODS
1->2	36.433	36.433	36.433	36.432
3->4	37.396	37.395	37.396	37.395
5->6	36.684	36.683	36.683	36.683
7->8	37.197	37.196	37.196	37.196
1->4	32.877	32.876	32.876	32.876
1->8	30.690	30.602	30.145	30.054

2.6.4 Mobile

ลำดับภาพ mobile เป็นลำดับภาพที่พื้นหลังมีการเปลี่ยนแปลงในลักษณะเลื่อนลงอย่างช้า ๆ มีการเปลี่ยนแปลงระหว่างภาพที่ค่อนข้างเร็วในทิศทางด้านขวาไปซ้ายและเกิดในบริเวณด้านล่างของภาพเป็นส่วนใหญ่

รูปที่ 2.13 ลำดับภาพที่ 1 ของ mobile

ตารางที่ 2.7 แสดงค่า PSNR ระหว่างลำดับภาพต่าง ๆ ในลำดับมาตรฐาน mobile

Ref. frame->Recon. frame	FS	3SS	2SS	PHODS
1->2	22.627	22.189	22.350	20.937
3->4	22.416	22.272	22.282	20.837
5->6	22.406	22.161	22.300	20.684
7->8	22.146	21.996	22.037	20.625
1->4	21.343	18.830	19.332	16.016
1->8	19.217	17.761	17.929	14.075

2.6.5 Tempete

ลำดับภาพ tempete เป็นลำดับภาพที่พื้นหลังมีการเปลี่ยนแปลงในลักษณะที่เป็นการซูมภาพออกอย่างช้า ๆ มีการเปลี่ยนแปลงระหว่างภาพที่ค่อนข้างเร็วในทิศทางจากบนลงล่างและเกิดกระจายไปทั่วภาพ

รูปที่ 2.14 ลำดับภาพที่ 1 ของ tempete

ตารางที่ 2.8 แสดงค่า PSNR ระหว่างลำดับภาพต่าง ๆ ในลำดับมาตรฐาน tempete

Ref. frame->Recon. frame	FS	3SS	2SS	PHODS
1->2	26.234	26.142	26.160	25.945
3->4	26.188	26.035	26.116	25.853
5->6	26.124	26.063	26.037	25.860
7->8	26.641	26.603	26.570	26.359
1->4	25.067	24.187	24.319	22.767
1->8	24.162	23.012	22.816	19.514

จากตารางแสดงค่า PSNR ตั้งแต่ตารางที่ 2.4 ถึง 2.8 จะเห็นได้ว่าอัลกอริทึมการค้นหา 1 มิติเป็นลำดับขั้นแบบขนานมีค่าความถูกต้องน้อยกว่าอัลกอริทึมอื่น ๆ ที่นำมาเปรียบเทียบอย่างเห็นได้ชัด โดยเฉพาะเมื่อระยะห่างจากลำดับภาพอ้างอิงมีค่ามาก เช่น ลำดับที่ 1 กับ 8 และในกรณีที่ลำดับภาพมีการเปลี่ยนแปลงอย่ารวดเร็วอย่างเช่น ลำดับภาพ mobile ถึงแม้ว่าอัลกอริทึมนี้จะมีความเร็วที่สูงกว่าอัลกอริทึมอื่น ๆ เป็นอย่างมาก แต่ค่าความผิดพลาดที่เกิดขึ้นดังกล่าว ทำให้เกิดปัญหาเมื่อนำไปใช้งานจริง ตัวอย่างดังรูปที่ 2.15 แสดงภาพลำดับที่ 4 ที่สร้างขึ้นจากเวคเตอร์ความเคลื่อนที่

ด้วยวิธีค้นหาแบบเต็มพื้นที่ โดยมีภาพลำดับที่ 1 เป็นภาพอ้างอิง และรูปที่ 2.16 แสดงภาพลำดับที่ 4 ที่สร้างขึ้นจากเวคเตอร์ของวิธีการค้นหา 1 มิติเป็นลำดับขั้นแบบขนาน ซึ่งจะเห็นได้ว่าคุณภาพของภาพต่างกันอย่างเห็นได้ชัด ซึ่งถ้าหากปรับปรุงในด้านของความถูกต้องได้ โดยกระทบกับความเร็วของอัลกอริทึมไม่มากจนเกินไป และยังคงลักษณะที่เป็นการประมวลผลแบบขนานเอาไว้ได้ อัลกอริทึมการค้นหา 1 มิติเป็นลำดับขั้นแบบขนาน ก็จะเป็นอัลกอริทึมที่เหมาะสำหรับนำไปใช้สร้างเป็นหน่วยประมวลผลเฉพาะทาง สำหรับไปใช้ในระบบประมวลผลที่มีความเร็วไม่สูงได้

รูปที่ 2.15 ภาพลำดับที่ 4 ของลำดับภาพ mobile ที่สร้างขึ้นจากเวคเตอร์ของ FS

รูปที่ 2.16 ภาพลำดับที่ 4 ของลำดับภาพ mobile ที่สร้างขึ้นจากเวคเตอร์ของ PHODS