การปรับระบบการเข้ารหัสและถอดรหัสวิดีโอภาพมาตรฐาน H.261 ให้เหมาะสม โดยการเลือกอัลกอริธึมที่มีประสิทธิภาพสูง

An Optimization for the H.261 Digital Video Signal Encoding and Decoding System by Selection of High Efficient Algorithms

เทอดศักดิ์ ธนกิจประภา ไกรสิน ส่งวัฒนา***
นักศึกษาปริญญาโท อาจารย์** ภาควิชาวิศวกรรมโทรคมนาคม*
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ABSTRACT – This paper presents an optimization for H.261 digital video signal encoding and decoding system, which can be used to increase the efficiency of a videoconferencing system. Since the most part of computational process of a videoconferencing system is in encoding and decoding video signal, a faster and more intelligent video encoder and decoder will increase the system performance. The optimization is done by studying the time spent by each individual function of the process. These functions are replaced with more efficient algorithms if large amounts of computational times are detected.

KEY WORDS – H.261 video encoding, H.261 video decoding, Optimization

บทคัดย่อ -- บทความนี้จะนำเสนอการเข้ารหัสและถอดรหัสวิดีโอภาพมาตรฐาน H.261 ให้เหมาะสม ซึ่งใช้ในการเพิ่มประสิทธิภาพของระบบการประชุมทางไกลผ่านวิดีโอภาพ (video conferencing) เนื่องจากการประมวลผลในระบบการประชุมทางไกลผ่านวิดีโอภาพส่วนมากจะเป็นการเข้ารหัสและถอดรหัสวิดีโอภาพ ดังนั้นการสร้างส่วนการเข้ารหัสและถอดรหัสวิดีโอภาพให้เร็วขึ้นและมีความฉลาดมากขึ้นจะเป็นการช่วยเพิ่มประสิทธิภาพของระบบ การปรับระบบจะทำโดยการศึกษาเวลาที่ใช้ในการคำนวณฟังก์ชั่นในแต่ละส่วนของการเข้ารหัสและถอดรหัสภาพ เพื่อพิจารณาหาอัลกอริธึมทดแทนเมื่อเวลาที่ใช้ในการคำนวณฟังก์ชั่นนั้นสูงเกินไป

คำสำคัญ – การเข้ารหัสวิดีโอภาพระบบ H.261, การถอดรหัสวิดีโอภาพระบบ H.261, การปรับแต่งระบบให้เหมาะสม

1. บทนำ

เนื่องจากการประมวลผลของระบบการประชุมทางไกลผ่านวิดีโอภาพตามมาตรฐาน ITU H.320, H.321, H.322, H.323 และ H.324 ส่วนใหญ่จะเป็นการเข้าและถอดรหัสวิดีโอภาพโดยเฉพาะเมื่อมีการประชุมทางไกลแบบหลายจุดพร้อมกัน ดังนั้นอุปกรณ์ที่ทำหน้าที่เข้ารหัสและถอดรหัสวิดีโอภาพระบบ H.261 จำเป็นต้องได้รับการออกแบบให้ทำงานได้รวดเร็วและถูกต้องมากที่สุดในสภาวะการทำงานแบบ real time ดังนั้นบทความนี้จึงมุ่งเน้นการเพิ่มความเร็วของอุปกรณ์เข้ารหัสและถอดรหัสวิดีโอภาพ ระบบ H.261 ด้วยการเลือกอัลกอริธึมประสิทธิภาพสูงในฟังก์ชั่นการทำงานหลักของระบบเพื่อนำอุปกรณ์นี้ไปใช้ในระบบการประชุมทางไกลผ่านวิดีโอภาพต่อไป และผลที่ได้จากการวิจัยนี้สามารถนำไปใช้ได้กับการเข้ารหัสและถอดรหัสวิดีโอภาพในระบบอื่น ๆ เช่น H.263 และ MPEG

การวิจัยเพื่อเพิ่มประสิทธิภาพของระบบเข้ารหัสและถอดรหัสวิดีโอภาพมีอย่างต่อเนื่องในช่วงสิบปีที่ผ่านมา โดยมุ่งเน้นการพัฒนาระบบฮาร์ดแวร์และอัลกอริธึมที่มีประสิทธิภาพสูงเพื่อนำมากับแต่ละส่วนฟังก์ชั่นในระบบเข้ารหัสและถอดรหัสวิดีโอภาพ เช่น การแปลงข้อมูลระหว่างระบบสี, การแปลงข้อมูล FDCT และ IDCT, การหา motion vector, การเข้ารหัส entropy coding และอื่น ๆ โดยในบทความนี้ทำการเลือกอัลกอริธึมประสิทธิภาพสูงจากผลการวิจัยที่ผ่านมาเพื่อนำมาใช้กับระบบเข้ารหัสและถอดรหัสวิดีโอภาพมาตรฐาน H.261

2. วิธีการที่ปรับแต่งระบบเข้ารหัสและถอดรหัสวิดีโอภาพระบบ H.261

การปรับแต่งระบบการเข้ารหัสและถอดรหัสวิดีโอภาพระบบ H.261 จะทำโดยการพิจารณาฟังก์ชั่นหลักที่สำคัญในการเข้ารหัสและถอดรหัสวิดีโอภาพระบบ H.261 คือการแปลงข้อมูลระหว่างระบบสี RGB กับ YCbCr, forward และ inverse Discrete Cosine Transform (FDCT และ IDCT), การหา motion vector และการเข้ารหัสและถอดรหัส Huffman

2.1 การเลือกอัลกอริธึมในการแปลงข้อมูลระหว่างระบบสี RGB กับ YCbCr

ระบบสีที่ใช้ในมาตรฐาน H.261 คือระบบสี YCbCr แต่อินพุทโดยทั่วไปในระบบ video conference ซึ่งรับมาจาก video capture card มักจะเป็นระบบสี RGB ดังนั้นจำเป็นต้องมีการแปลงข้อมูลในระบบ RGB เป็น YCbCr และในทางกลับกันเอาท์พุทจากระบบ H.261 จะต้องแปลงระบบสีเป็น RGB ก่อนที่จะถูกแสดงผลบนจอภาพคอมพิวเตอร์ โดยมีสมการดังนี้

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix}\begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad \ldots(1)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 298.082 & 0 & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0 \end{bmatrix}\begin{bmatrix} Y-16 \\ Cb-128 \\ Cr-128 \end{bmatrix} \quad \ldots(2)$$

การคำนวณค่าสีของพิกเซลจาก RGB เป็น YCbCr สามารถแยกออกเป็น 2 กรณี คือ กรณีที่เป็นตารางสี และกรณี true colors ในกรณีที่เป็นตารางสีจะทำการแปลงตารางสีเพียงครั้งเดียว ส่วนในกรณีของ true color จะต้องทำการแปลงค่าสีของทุกจุดพิกเซล

การคำนวณค่าสีของพิกเซลจาก RGB เป็น YCbCr และ YCbCr เป็น RGB จะใช้อัลกอริธึม Dither ซึ่งเปลี่ยนการคูณแบบทศนิยมเป็นการคูณด้วยจำนวนเต็มและการหารด้วยเลื่อนบิตซึ่งเป็นการประมาณด้วยการปัดเศษและการคูณจำนวนเต็มทั้งหมดจะถูกคำนวณไว้ล่วงหน้าและเก็บไว้ในหน่วยความจำเพื่อสามารถดึงมาใช้งานได้ทันทีโดยไม่ต้องคำนวณซ้ำอีก

2.2 การเลือกอัลกอริธึมในการ FDCT และ IDCT

การแปลงข้อมูล FDCT และ IDCT เป็นอัลกอริธึมหลักในการเข้ารหัสและถอดรหัสวิดีโอภาพโดยมีสมการ FDCT และ IDCT ดังนี้

$$F(u,v) = \frac{C(u)C(v)}{4}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right) \quad \ldots(3)$$

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right) \quad \ldots(4)$$

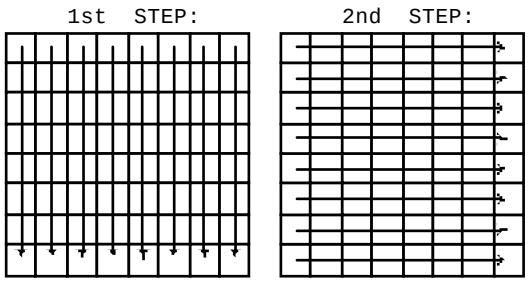
เมื่อ x, y คือ spatial coordinate ในบล็อกขนาด 8x8

u, v คือ coordinate ใน transform domain

$C(i) = \frac{1}{\sqrt{2}}$ เมื่อ $i = 0$ และ $C(i) = 1$ เมื่อ $i \neq 0$

2D-DCT สามารถแบ่งออกเป็นโดยการแปลงข้อมูล 1D-DCT 2 ครั้งโดยจะแปลง DCT แถวข้อมูลในแนวตั้งก่อนซึ่งผลที่ได้ถูกแปลง DCT แถวในแนวนอนอีกครั้งหนึ่งดังรูปที่ 1[1]

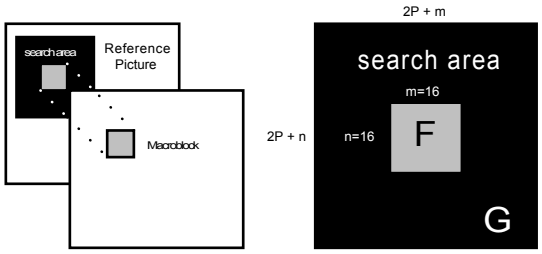
อัลกอริธึมสำหรับ 1D-DCT หลายวิธีถูกเสนอขึ้นมาภายในช่วงเวลา 15 ปีที่ผ่านมา รวมทั้งอัลกอริธึมของ Chen[2] ที่มีพื้นฐานบน sparse matrix factorization โดยได้ลดจำนวน arithmetic operation ลงเหลือเป็นการคูณ 16 ครั้งและการบวก 26 ครั้งต่อการแปลง 1D-DCT ข้อมูล 8 จุด



รูปที่ 1. แสดงการแปลง 2-D DCT โดยการทำ 1-D DCT 2 ขั้นตอน

2.3 การเลือกอัลกอริธึมในการหา motion vector

การหา motion vector มีหลายวิธีที่ใช้ได้กับตัวเข้ารหัส โดยวิธีที่มีการคำนวณมากจะให้ผลที่ดีกว่าแต่ก็ต้องใช้กำลังการคำนวณที่สูงกว่า จึงทำให้ต้องมีการหาจุดสมดุลย์ระหว่างกำลังในการคำนวณกับคุณภาพของวิดีโอภาพที่ต้องการโดยวิธีการหา motion vector ที่นิยมใช้กันมากที่สุดคือเทคนิค block-matching เป้าหมายของเทคนิคนี้คือการประมาณค่า motion vector ของบล็อกขนาด m x n (16x16) ในเฟรมปัจจุบันเทียบกับพิกเซลในเฟรมอ้างอิง (เฟรมในอดีตหรืออนาคต) บล็อกจะถูกเปรียบเทียบกับบล็อกที่เกี่ยวข้องภายในพื้นที่ search area ดังรูปที่ 2



รูปที่ 2. แสดงวิธีการหา motion vector ด้วยเทคนิค block-matching

เทคนิค block-matching จะอาศัยใช้การหา cost function ของค่า luminance จุดต่อจุดระหว่าง macroblock ปัจจุบันกับค่าภายใน search area ที่มีค่าน้อยที่สุด โดย cost function มีอยู่หลายรูปแบบเช่น

- Mean-Absolute Difference (MAD) ดังสมการ

$$MAD(dx,dy) = \frac{1}{mn}\sum_{i=-n/2}^{n/2}\sum_{j=-m/2}^{m/2}\left|F(i,j) - G(i+dx, j+dy)\right| \quad \ldots(5)$$

- Mean-Squared Difference (MSD) ดังสมการ

$$MSD(dx,dy) = \frac{1}{mn}\sum_{i=-n/2}^{n/2}\sum_{j=-m/2}^{m/2}\left[F(i,j) - G(i+dx, j+dy)\right]^2 \quad \ldots(6)$$

- Cross-Correlation Function (CCF) ดังสมการ

$$CCF(dxdy) = \frac{\sum_i\sum_j F(i,j)G(i+dx, j+dy)}{\left(\sum_i\sum_j F^2(i,j)\right)^{\frac{1}{2}}\left(\sum_i\sum_j G^2(i+dx, j+dy)\right)^{\frac{1}{2}}} \quad \ldots(7)$$

เมื่อ $F(i,j)$ คือข้อมูล macroblock ขนาด m x n จากเฟรมปัจจุบัน

$G(i,j)$ คือข้อมูล macroblock ขนาดเดียวกันจากเฟรมอ้างอิง (ในอดีตหรืออนาคต)

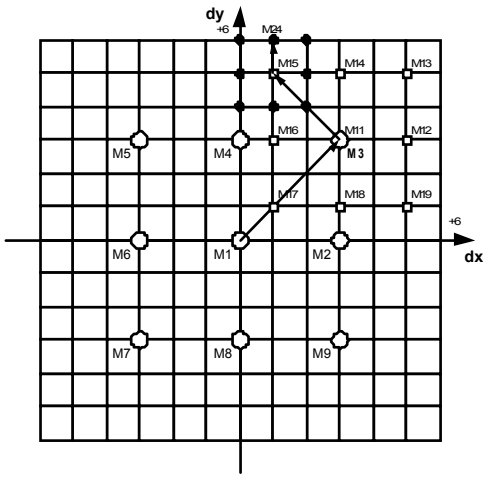
(dx,dy) คือ vector ที่แสดงตำแหน่งของการหา cost function โดยมีช่วงการหา $dx = [-p,+p]$ และ $dy = [-p,+p]$

เนื่องจากการหา cost function ตลอดทั้งพื้นที่ search area ซึ่งเรียกว่า exhaustive ต้องสิ้นเปลืองพลังการคำนวณสูง จึงได้มีการใช้คิดอัลกอริธึมที่ช่วยในการหา minimum cost function ขึ้นด้วยการคำนวณที่น้อยครั้งกว่าเช่นอัลกอริธึม logarithmic ที่อาศัยการแบ่งหาการ cost fuction เป็นลำดับ ๆ โดยระยะในการหาจะสั้นลงในแต่ละลำดับและทิศทางในการหาจะขึ้นอยู่กับค่า cost function ที่หาได้ในลำดับที่ผ่านมา ยกตัวอย่างดังรูปที่ 3

จากรูปแสดงตัวอย่างการหา motion vector ในช่วง ∂6 พิกเซล โดยแบ่งการหาเป็นลำดับได้เป็น 4 ลำดับโดยแต่ละลำดับจะมีระยะในการหาคือ 3 2 1 พิกเซลตามลำดับ

- ลำดับแรกระยะ 3 พิกเซลหาที่ตำแหน่ง M1-M9 โดยมีจุดอ้างอิง (0,0) เป็นจุดศูนย์กลางซึ่งได้ค่า cost function ต่ำที่สุดที่ตำแหน่ง M3
- ลำดับที่สองระยะ 2 พิกเซลหาที่ตำแหน่ง M11-M19 โดยมีตำแหน่ง M3 จากลำดับที่แล้วเป็นจุดศูนย์กลางซึ่งได้ค่า cost function ต่ำที่สุดที่ตำแหน่ง M15
- ลำดับสุดท้ายระยะ 1 พิกเซลหาที่ตำแหน่งพิกเซลรอบจุด M15 ซึ่งจะได้ cost function ต่ำสุดที่ตำแหน่ง M24

ดังนั้น motion vector คือ vector จากจุด (0,0) ไปยังจุด M24



รูปที่ 3. แสดงตัวอย่างการหา motion vector โดยใช้อัลกอริธึม logarithmic

2.4 การเลือกอัลกอริธึมในการเข้ารหัสและถอดรหัส Huffman

เนื่องจากตาราง huffman ที่ใช้ในการเข้ารหัสวิดีโอภาพได้ถูกกำหนดไว้แล้ว ทำให้ไม่จำเป็นต้องใช้อัลกอริธึมที่ซับซ้อนในการเข้ารหัสเพียงแต่กำหนดค่าตาราง huffman ทั้งหมดไว้ในหน่วยความจำของคอมพิวเตอร์เพื่อความรวดเร็วในการประมวลผล ส่วนการถอดรหัสจำเป็นต้องใช้อัลกอริธึมเพื่อให้การถอดรหัสมีความรวดเร็วและใช้หน่วยความจำอย่างมีประสิทธิภาพ อัลกอริธึมที่ใช้ในการถอดรหัส huffman ที่ใช้กันทั่วไปคือวิธีการสร้าง look-up table แต่เนื่องจากการสร้างตาราง look-up table นี้ต้องครอบคลุมค่าที่เป็นไปได้ทั้งหมดภายในตาราง huffman (คือ 2^n เมื่อ n คือความยาวของ codeword ที่ยาวที่สุดภายในตาราง) ทำให้การสร้าง look-up table สำหรับตาราง huffman ซึ่งมี codeword ที่ยาวมากจะทำให้ตารางมีความยาวมากซึ่งจะทำให้สิ้นเปลืองหน่วยความจำจำนวนมากในการเก็บตาราง การแก้ปัญหานี้ทำได้โดยการแบ่ง look-up table ออกเป็นหลาย ๆ ตารางโดยกำหนดให้แต่ละตารางครอบคลุมบิตข้อมูลที่เป็นไปได้ของ codeword จำนวนหนึ่งเช่นตาราง Transform coefficient (TCOEF) มี codeword ยาวที่สุด 12 บิต การสร้าง look-up table จะแบ่งออกเป็น 2 ตารางโดยตารางแรกครอบคลุม 6 บิตแรกและตารางที่สองครอบคุลม 6 บิตที่เหลือ

3. ผลการทดลอง

เพื่อเปรียบเทียบอัลกอริธึมที่ได้เลือกใช้ในแต่ละฟังก์ชั่น ผู้วิจัยได้สร้างโปรแกรมเข้ารหัสและถอดรหัสวิดีโอภาพระบบ H.261 ด้วยภาษา visual c++ โดยทำการทดลองบนเครื่องคอมพิวเตอร์ Pentium II 300 MHz มีหน่วยความจำ 128 MB ในระบบ Windows NT โดยรับอินพุทจาก avi file (*.avi) และ video capture card ซึ่งใช้การ์ดและกล้องยี่ห้อ Winnov รุ่น videum ได้ผลการทดลองดังนี้

3.1 ผลการปรับระบบเข้ารหัสวิดีโอภาพ H.261

การเข้ารหัสวิดีโอภาพระบบ H.261 มีฟังก์ชั่นที่ต้องปรับ คือ การแปลงระบบสีจาก RGB เป็น YCbCr, การแปลง DCT ซึ่งมีทั้ง FDCT และ IDCT เนื่องจากต้องทำการจำลองเอาท์พุทด้านตัวถอดรหัสเพื่อสร้าง error signal ที่ถูกต้องและการหา motion vector ด้วยการหา MAD cost function ซึ่งได้ผลดังนี้ (เวลาที่วัดเป็นเวลาเฉลี่ยที่ใช้ในแต่ละฟังก์ชั่นต่อการเข้ารหัส 1 เฟรมภาพ)

ตารางที่ 1. แสดงผลแปลงระบบสี RGB เป็นระบบสี YCbCr ระหว่างการคำนวณ Direct กับ Lookup-table ในการเข้ารหัสวิดีโอภาพระบบ H.261

วิดีโอภาพ ขนาด QCIF	Direct เวลา (ms)	Dither เวลา (ms)
การบรรยาย (150เฟรม)	14.44	2.64
ตึกถล่ม (307 เฟรม)	14.66	2.12
นักอวกาศ (303 เฟรม)	15.23	2.70

ผลจากตารางแสดงให้เห็นว่าการใช้อัลกอริธึม dither ในการแปลงระบบสี RGB เป็น YCbCr ช่วยลดเวลาในการคำนวณลดประมาณ 5 เท่า

ตารางที่ 2. แสดงผลการ DCT (FDCT และ IDCT) ระหว่างอัลกอริธึม Direct กับ Chen's ในการเข้ารหัสวิดีโอภาพระบบ H.261

วิดีโอภาพ ขนาด QCIF	Direct		Chen's	
	เวลา (ms)	PSNR (dB)	เวลา (ms)	PSNR (dB)
การบรรยาย (150เฟรม)	21.27	38.47	10.51	38.46
ตึกถล่ม (307 เฟรม)	22.64	37.65	11.38	37.64
นักอวกาศ (303 เฟรม)	24.14	37.66	11.91	37.66

ผลจากตารางแสดงให้เห็นว่าการใช้อัลกอริธึม Chen's 1D-DCT ในการ FDCT และ IDCT ให้คุณภาพของวิดีโอภาพคือค่า PSNR (peak signal-to-reconstructed image) ที่เกือบเหมือนการคำนวณแบบ direct และใช้เวลาน้อยกว่าประมาณครึ่งหนึ่ง

ตารางที่ 3. แสดงผลการหา motion vector ระหว่างอัลกอริธึม exhaustive และ logarithmic ด้วยการคำนวณ MAD ในการเข้ารหัสวิดีโอภาพระบบ H.261

วิดีโอภาพ ขนาด QCIF	Exhaustive		Logarithmic	
	เวลา (ms)	PSNR (dB)	เวลา (ms)	PSNR (dB)
การบรรยาย (150เฟรม)	2.20	38.47	0.09	38.46
ตึกถล่ม (307 เฟรม)	5.92	37.64	0.22	37.64
นักอวกาศ (303 เฟรม)	14.82	37.67	0.58	37.65

ผลจากแสดงให้เห็นว่าอัลกอริธึม logarithmic ให้ผลการหา motion vector ที่ให้คุณภาพของวิดีโอภาพคือค่า PSNR ใกล้เคียงกับการหาแบบ full search โดยใช้เวลาน้อยกว่ามาก

ตารางที่ 4. แสดงผลก่อนและหลังการปรับระบบเข้ารหัสวิดีโอภาพ H.261

วิดีโอภาพ ขนาด QCIF	ก่อนการปรับ เวลาเฉลี่ย (ms)	หลังการปรับ เวลาเฉลี่ย (ms)
Color conversion	14.78	2.49
FDCT, IDCT	22.68	11.27
MV search	7.65	0.30
อื่น ๆ	30.45	29.90
Frame/Sec.	13.20	22.75

ผลจากตารางเป็นผลการทดลองเฉลี่ยของการเข้ารหัสวิดีโอภาพทั้งสาม โดยเวลาอื่น ๆ ที่แสดงในตารางคือเวลาที่ใช้ในการ quantization และ dequantization, การเรียงข้อมูลแบบ zigzag, การสร้าง predictive macroblock จาก motion vector, การเข้ารหัส huffman, การแสดงผลบนจอภาพ, การอ่านไฟล์ AVI input จาก harddisk และการเขียนเอาท์พุทไฟล์ H.261 ลงใน harddisk ผลการทดลองแสดงให้เห็นว่าการปรับแต่งระบบช่วยให้การเข้ารหัสรวดเร็วขึ้นประมาณ 70% โดยที่คุณภาพของวิดีโอภาพแทบไม่เปลี่ยนแปลงคือ PSNR ลดลง 0 dB ถึง 0.02 dB

3.2 ผลการปรับระบบถอดรหัสวิดีโอภาพ H.261

การถอดรหัสวิดีโอภาพระบบ H.261 มีฟังก์ชั่นที่ต้องปรับ คือ การแปลงระบบสีจาก YCbCr เป็น RGB, การแปลง IDCT และการถอดรหัส huffman ซึ่งได้ผลดังนี้ (เวลาที่วัดเป็นเวลาเฉลี่ยที่ใช้ในแต่ละฟังก์ชั่นต่อการถอดรหัส 1 เฟรมภาพ)

ตารางที่ 5. แสดงผลแปลงระบบสี YCbCr เป็นระบบสี RGB ระหว่างการคำนวณ Direct กับอัลกอริธึม Dither ในการถอดรหัสวิดีโอภาพระบบ H.261

วิดีโอภาพ ขนาด QCIF	Direct เวลา (ms)	Dither เวลา (ms)
การบรรยาย (150เฟรม)	25.61	3.07
ตึกถล่ม (307 เฟรม)	25.46	3.02
นักอวกาศ (303 เฟรม)	25.46	3.04

ผลจากแสดงให้เห็นว่าการใช้อัลกอริธึม Dither ช่วยลดเวลาลงได้อย่างมาก

ตารางที่ 6. แสดงผลการ IDCT ระหว่างอัลกอริธึม Direct กับ Chen's ในการถอดรหัสวิดีโอภาพระบบ H.261

วิดีโอภาพ ขนาด QCIF	Direct เวลา (ms)	Chen's เวลา (ms)
การบรรยาย (150เฟรม)	12.14	4.51
ตึกถล่ม (307 เฟรม)	14.39	5.37
นักอวกาศ (303 เฟรม)	17.04	6.43

ผลจากตารางแสดงให้เห็นว่าการใช้อัลกอริธึม Chen's 1D-DCT ในการ IDCT ใช้เวลาน้อยกว่าการคำนวณโดยตรงประมาณครึ่งหนึ่ง

ตารางที่ 7. แสดงเวลาที่ใช้ในการถอดรหัส huffman ระหว่างอัลกอริธึม Direct และ Look-up table ในการถอดรหัสวิดีโอภาพระบบ H.261

วิดีโอภาพ ขนาด CIF	Direct (ms)	Look-up table (ms)
การบรรยาย (150เฟรม)	2.64	2.28
ตึกถล่ม (307 เฟรม)	2.87	2.48
นักอวกาศ (303 เฟรม)	4.20	3.61

ผลจากตารางแสดงให้เห็นว่าการถอดรหัส huffman ด้วยอัลกอริธึม look-up table ช่วยเพิ่มความเร็วในการถอดรหัสได้เล็กน้อย

ตารางที่ 8. แสดงผลก่อนและหลังการปรับระบบถอดรหัสวิดีโอภาพ H.261

วิดีโอภาพ ขนาด QCIF	ก่อนการปรับ เวลาเฉลี่ย (ms)	หลังการปรับ เวลาเฉลี่ย (ms)
Color conversion	25.51	3.04
IDCT	14.52	5.44
Huffman decoding	3.24	2.79
อื่น ๆ	9.72	9.92
Frame/Sec.	18.87	47.18

ผลจากตารางเป็นผลการทดลองเฉลี่ยของการถอดรหัสวิดีโอภาพทั้งสาม โดยเวลาอื่น ๆ ที่แสดงในตารางคือ เวลาที่ใช้ในการ dequantization, การเรียงข้อมูลแบบ zigzag, การสร้าง predictive macroblock จาก motion vector, การอ่านไฟล์ H.261 จาก harddisk, การแสดงผลบนจอภาพ ผลการทดลองแสดงให้เห็นว่าการปรับแต่งระบบช่วยทำให้การถอดรหัสรวดเร็วขึ้นถึงประมาณ 250%

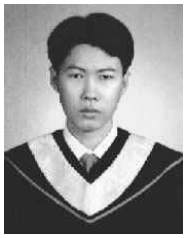
4. สรุป

การทดลองกับตัวอย่างวิดีโอทั้งสามไฟล์ข้างต้นซึ่งมีลักษณะเนื้อหาของวิดีโอภาพแตกต่างกันมากแต่ได้ผลการทดลองในแนวทางเดียวกันคือเพิ่มความเร็วทั้งในส่วนการเข้ารหัสและถอดรหัสในระดับที่ใกล้เคียงกันในแต่ละฟังก์ชั่นที่ทำการทดลอง ทำให้สรุปได้ว่าการเลือกใช้อัลกอริธึมประสิทธิภาพสูงในฟังก์ชั่นการทำงานหลักของการเข้ารหัสและถอดรหัสวิดีโอภาพระบบ H.261 ส่งผลให้การทำงานมีความเร็วมากขึ้น โดยในการเข้ารหัสมีความเร็วเพิ่มขึ้นประมาณ 70% และในการถอดรหัสมีความเร็วเพิ่มขึ้นประมาณ 250%

เอกสารอ้างอิง

[1] K.R. Rao and P. Yip, “Discrete Cosine Transform”, Academic Press, 1990.

[2] W.A. Chen, C. Harrison and S.C. Fralick, “A Fast computational Algorithm for Discrete Cosine Transform”, *IEEE Transactions on Communications*, Vol. COM-25, No. 9, Sept. 1977. pp. 1024



เทอดศักดิ์ ธนกิจประภา เกิดเมื่อวันที่ 4 มีนาคม 2516, สำเร็จการศึกษาวิทยาศาสตร์บัณฑิต สาขาฟิสิกส์ คณะวิทยาศาสตร์ จากมหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2536 ปัจจุบันกำลังศึกษาปริญญาโทคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ไกรสิน ส่งวัฒนา เกิดเมื่อวันที่ 16 มิถุนายน 2507, ได้รับปริญญาตรี, ปริญญาโท และปริญญาเอก จาก University of Wisconsin-Madison ประเทศอเมริกา ในปี พ.ศ. 2530, 2532 และ 2536 ตามลำดับ ขณะนี้เป็นผู้ช่วยศาสตราจารย์ประจำภาควิชาโทรคมนาคมสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง