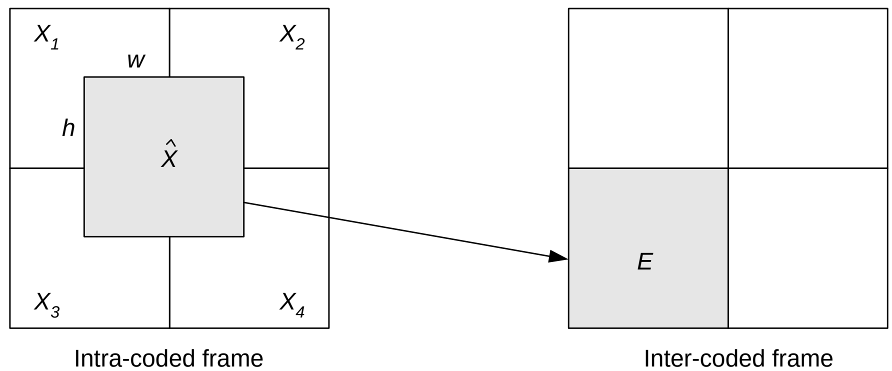
1. Introduction

- Video transcoding, where a pre-coded video bit-stream is converted from one format to another format, is of interest for purposes such as channel bandwidth adaptation and video composition
- Inverse Motion Compensation (IMC) is a necessary step in video transcoding to convert all Inter-frames to Intra-frames
- DCT-domain video transcoding has been shown more efficient than spatial-domain transcoding
- DCT-domain IMC is more complex than its counterpart in spatial-domain since data is organized block by block in the DCT-domain
- Faster DCT-domain IMC algorithms are needed to support real-time video transcoding

2. DCT-domain IMC

- General setup:



$$\hat{x} = \sum_{i=1}^{4} q_{i1} x_i q_{i2} \qquad i = 1, \cdots, 4$$

q_{i1}, q_{i2} are sparse 8×8 matrices that perform windowing and shifting operations. For example

$$q_{11} = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}_{8x8}, \quad q_{12} = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}_{8x8}$$

- Using the linear, distributive and unitary properties of DCT, one can obtain:

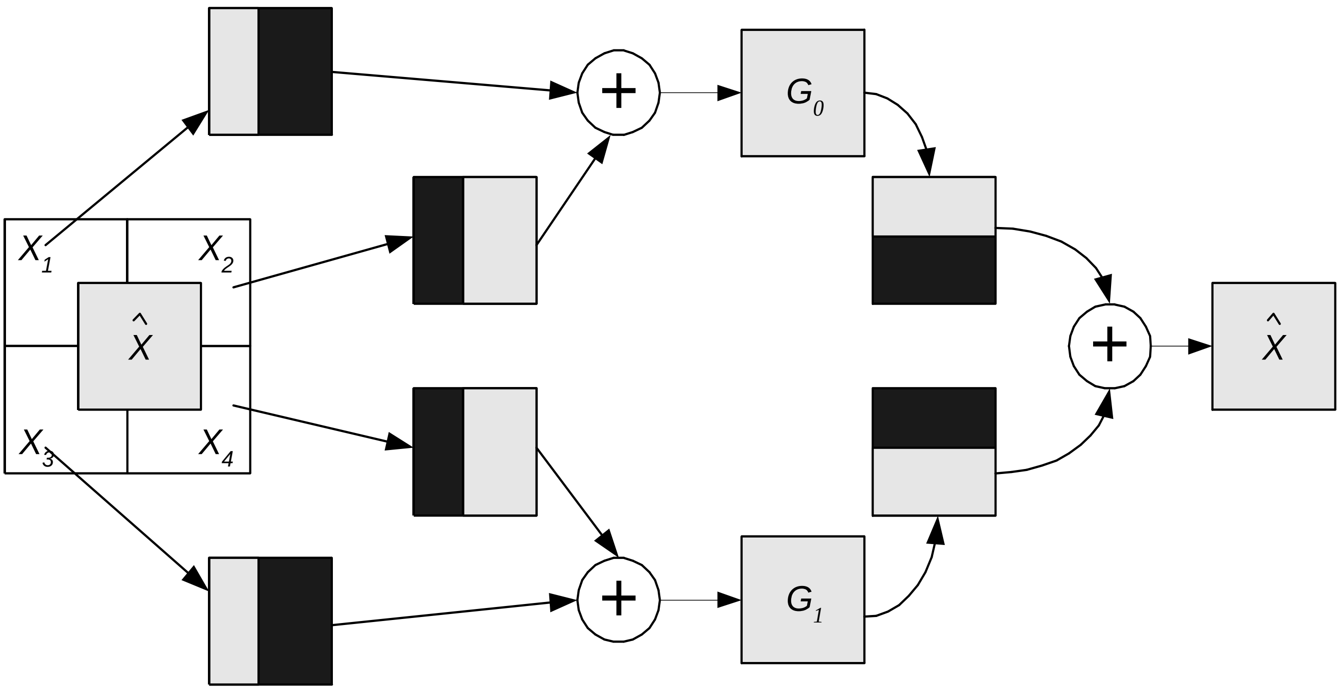
$$\hat{X} = \sum_{i=1}^{4} Q_{i1} X_i Q_{i2} \quad \text{in DCT} - \text{domain}$$

$\hat{X}$, X_i, Q_{ij} are DCT's of $\hat{x}$, x_i and q_{ij} respectively.

3. Existing Algorithms

- Chang *et al.* [Chang'93] proposed to pre-compute the matrices Q_{ij} and stored in memory.
- Merhav *et al.* [Merhav'97] factorized the matrices Q_{ij} into a series of relatively sparse matrices so that some of matrix multiplications can be replaced by simple addition and permutation operations
- Assuncao *et al.* [Assuncao'98] approximated the elements of Q_{ij} by binary numbers with maximum distortion of 1/32 so that all multiplications can be implemented by *shifts* and *adds*.
- Acharya *et al.* [Acharya'98] developed a separable scheme to decompose the 2-D problem to two 1-D problems.

4. Separable Scheme



$$G_0 = X_1 Q_{x0} + X_2 Q_{x1}$$

$$G_1 = X_3 Q_{x0} + X_4 Q_{x1}$$

$$\hat{X} = Q_{y0} G_0 + Q_{y1} G_1$$

where

$$Q_{12} = Q_{32} \equiv Q_{x0},, Q_{22} = Q_{42} \equiv Q_{x1}$$

and

$$Q_{31} = Q_{41} \equiv Q_{y1}, \quad Q_{11} = Q_{21} \equiv Q_{y0}$$

5. LUT Based IMC

- LUT based IMC is proposed by modeling the statistical distribution of DCT coefficients in typical images and video sequences
- The AC components can be modeled as a Laplacian distribution with zero mean as follows:

$$p(x) = \frac{\lambda}{2}\exp(-\lambda\,|\,x\,|)$$

where

$$\lambda = \frac{1}{E[|\,X\,|]}\cdot$$

- The value of λ is estimated as 0.0284.
- Let o^2 be the variance of *X*, then $\sigma = \frac{\sqrt{2}}{\lambda}$. If we set a threshold $TH = 2\sigma \approx 100$, then more than 94% of AC coefficients have absolute value smaller than the threshold *TH*.

6. LUT Design

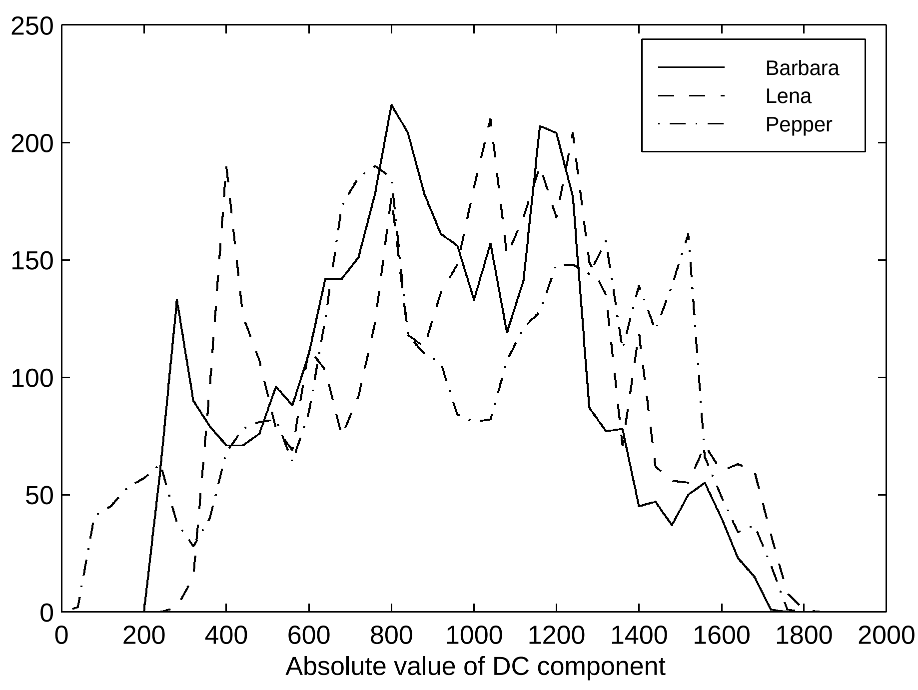
- Pre-computing the multiplication results of those AC coefficients with absolute value less than *TH* can save significant computation.
- For 1-D case, only two tables are needed to store the pre-computed results since

$$Q_{x0} = Q_{y0}^{t}, \quad Q_{x1} = Q_{y1}^{t}.$$

- Suppose four bytes are used to store each entry of the table, the size of the table is 400KB for *TH* = 100. Therefore, the total memory requirement of two tables is 800KB
- The table can be shared by multiple applications running on the same machine
- According to the model, the multiplication results of more than 94% of AC coefficients can be obtained by table look-up, which also includes results of half-pixel motion vectors.

7. DC Coefficient

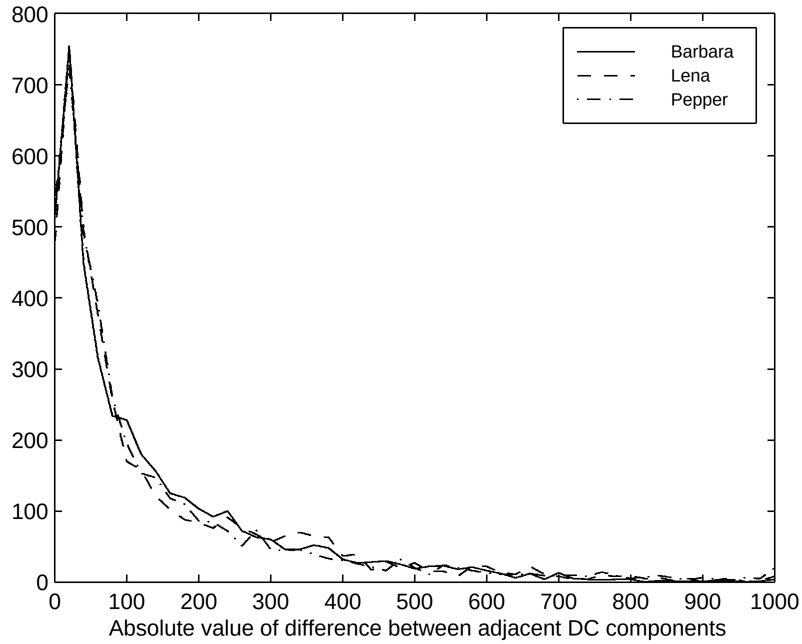
- The tables created by modeling the distribution of AC coefficients do not apply to DC components
- The distribution of DC component has larger mean and variance relative to that of AC coefficients as shown below



Histogram of DC coefficients in images with the Bin size 40

8. DC Coefficient…

- The difference between adjacent DC components has similar distribution as that of AC coefficient



Histogram of difference between adjacent DC components in images with the Bin size 20

- More than 70% of the difference values have absolute value below the threshold *TH*
- Therefore, we can process DC component as

$$G_0 = X_1 Q_{x0} + X_2 Q_{x1}$$
$$= \frac{X_1 + X_2}{2}(Q_{x0} + Q_{x1}) + \frac{X_1 - X_2}{2}(Q_{x0} - Q_{x1})$$

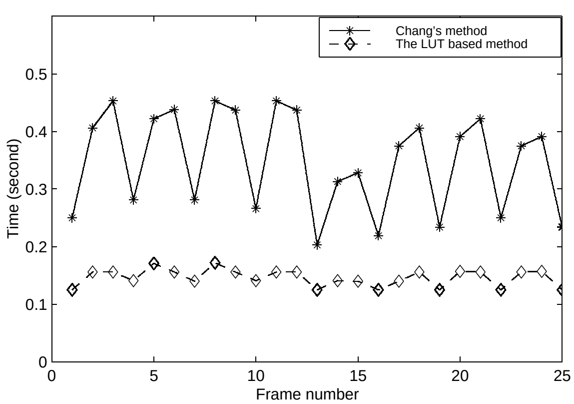
9. Experimental Results

- Both Chang's method and the LUT based method are implemented for comparison
- The computing time is measured on a Windows NT workstation with 512 MB memory and 300 MHz Pentium II. The results are shown below

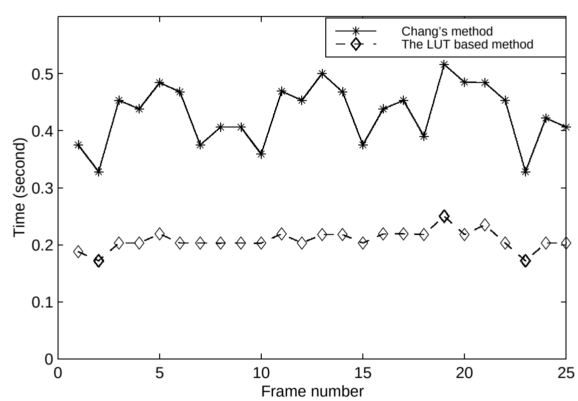
Video sequence	Chang's method		LUT based method	
	P frame	B frame	P frame	B frame
"Foreman"	0.3137	0.4738	0.0931	0.1423
"Coastguard"	0.2374	0.3417	0.0912	0.1190
"Mobile"	0.3487	0.4136	0.1462	0.2000
"Stefan"	0.2057	0.3667	0.0780	0.1416

Table 1. The average time to convert one P or B frame to an I frame (Unit: Second)

- The time for reconstructing each P or B frame to an I frame in "Mobile" sequence is plotted:



Time for reconstructing each P frame to I frame



Time for reconstructing each B frame to I frame

MPEG video is encoded at 1 Mb/s

10. Conclusion

- A LUT based method for DCT-domain inverse motion compensation is proposed
- The proposed method achieves more than 50% computational savings, relative to Chang’s method, with the same quality
- The results obtained by LUT method are the same as that by Chang’s method
- Compared to other existing algorithms, the proposed method is straightforward to implement and introduce no error
- For half-pixel accurate motion vectors, the proposed method has the same computational complexity as that for integer-pixel accurate motion vectors. Therefore, it can reduce the jerkiness in real-time video processing applications
- The tables can be shared by multiple applications running on the same machine