SenseCoding: Accelerometer-Assisted Motion Estimation for Efficient Video Encoding

Guangming Hong[1], Ahmad Rahmati[2], Ye Wang[1], Lin Zhong[2]

[1]School of Computing
National University of Singapore,
Singapore, 117590
{honggm, wangye}@comp.nus.edu.sg

[2]Department of Electrical & Computer Engineering
Rice University
Houston, TX, 77005
{rahmati, lzhong}@rice.edu

ABSTRACT

Accelerometers have appeared on many camcorders, cameras and mobile phones. We present algorithms that estimate camera movement from accelerometer readings and apply the estimation to significantly improve the compute-intense motion estimation in video encoding. We have implemented a working prototype that simultaneously captures video and three-axis acceleration data. The video is then compressed with a reference MPEG-2 encoder, modified to incorporate the accelerometer readings to assist motion estimation. Our experimental data shows a two to three times speed improvement for the entire encoding process, in comparison with full search.

Categories and Subject Descriptors

H.5.1 [Multimedia Information Systems]: Video

General Terms

Algorithms, Design , Performance

Keywords

Accelerometer, Motion Estimation, Video Encoding, MPEG-2

1. INTRODUCTION

Video capturing has become extremely popular, in particular in the age of social networking and YouTube. Compact digital cameras and an increasing number of mobile phones are capable of capturing short video clips. However, video capturing is well known to be compute-intense and power-hungry, in particular because of video encoding procedure to compress captured video clips. Figure 1 shows the basic structure of a conventional hybrid DCT/DPCM video encoder widely used in modern video capturing devices. The encoder consists of three major computation-intensive components: Motion Estimation (ME), Discrete Cosine Transform (DCT), and Variable Length Coding (VLC). Of the three, motion estimation is dominant in computation cost. Yet it is critical to leveraging inter-frame redundancy for compression. Our experiment shows that motion estimation accounts for 50%-95% of the overall encoding time in MPEG-2 reference encoder when full search algorithm is adopted.

On the other hand, ultra low-power, low-cost accelerometers have

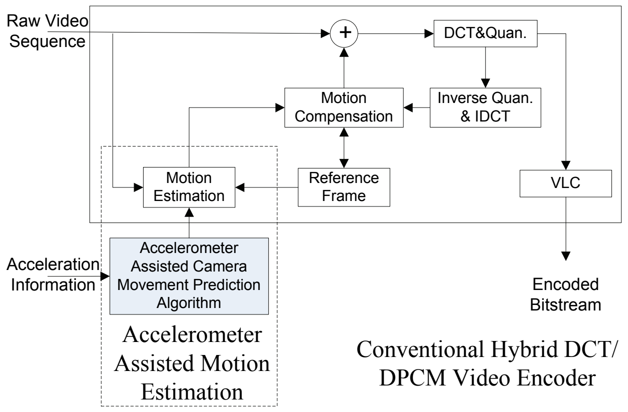

Figure 1. Basic structure of a hybrid DCT/DPCM video encoder and the accelerometer-assisted motion estimation

appeared in mobile devices and consumer electronics. Their power consumption is usually only a fraction of that by video encoding, e.g. over 200mWatt on a T-Mobile MDA Pocket PC phone by our measurement. This inspires us to leverage such sensors to detect camera movement and improve the efficiency of motion estimation in video encoding. There are, however, multiple challenges: accelerometers detect acceleration but motion estimation requires displacement, or distance; the motion and orientation of video capturing devices can be complicated and difficult to detect based solely on accelerometers; objects in view can move (known as *local motion*) at the same time as the capturing device moves (known as *global motion*). Since global motion is increasingly common in amateur video making with handheld devices, we intend to facilitate global motion estimation without undermining local motion estimation.

In this work, we present our accelerometer-assisted video encoder, SenseCoding, to effectively address these challenges and dramatically improve the efficiency of MPEG encoding by 2-3 times. SenseCoding employs two three-axis accelerometers to accurately capture acceleration information of the video capturing device. It then converts the acceleration data into a motion vector for adjacent frames. To evaluate SenseCoding, we have built a prototype using a commercial digital camcorder and an in-house sensor board with dual three-axis accelerometers. We collect raw video clips with various camera movements along with the corresponding acceleration data and carefully synchronize them. We then explore various design issues with SenseCoding and demonstrate its effectiveness. Our experiments shows that SenseCoding can reduce the total computation load by two to three times, which can

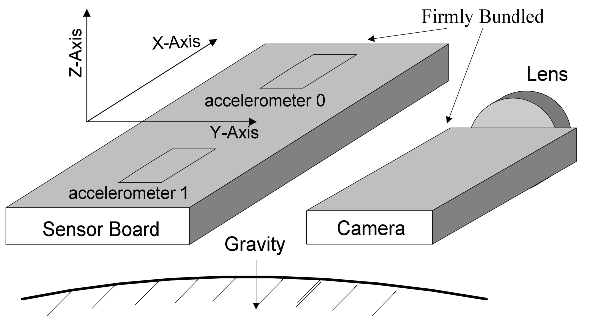

Figure 2. Model of sensor board with two accelerometers

lead to reduced hardware requirements and costs as well as longer battery lifetime for video capturing devices.

To the best of our knowledge, SenseCoding is the first published work on applying sensors to improve motion estimation in video coding. It represents a new direction in exploiting the synergy between different components in a handheld device to improve the efficiency of multimedia processing. A more detailed documentation of SenseCoding can be found at [1].

The rest of the paper is organized as follows. In Section 2, we address the technical details of SenseCoding, accelerometer-based motion estimation and its application in video encoding. We present a prototype implementation of SenseCoding and discuss our experimental results based on the prototype in Section 3. We address the limitation of SenseCoding and conclude in Section 4.

2. SENSECODING ALGORITHMS

We next present SenseCoding, a technique to significantly reduce the high computational complexity of video encoding by leveraging accelerometers found in many modern video capturing devices. SenseCoding consists of two key components: sensor-assisted motion vector estimation and simplified video encoding with the estimated motion vectors, which are presented next.

2.1 Sensor-Assisted Motion Vector Estimation

SenseCoding employs readings from two three-axis accelerometers to predict motion vectors due to camera movement in video recording. The accelerometers are placed apart on the same board and the board is firmly attached to the camera, as shown in Figure 2. Most camera movements in handheld video capturing can be considered as a combination of vertical (rolling up and down) and horizontal (turning left and right) angular movements. Based on the earth's gravity, one accelerometer is sufficient to calculate vertical movement. However, two accelerometers placed apart from each other are necessary to calculate horizontal movement.

A change in the angle of a camera results in the movement of the captured image. To calculate the relationship between the camera angle change and movement of the image in the camera CCD, we must understand the camera characteristics and build an optical model. Figure 3 illustrates the optical model used in this work. In the figure, A denotes an object in the view; *z* denotes its distance from the camera lens; and *O* denotes the optical center of the CCD. The projection of A on the camera CCD is located at a distance of h_1 from *O*. When the camera lens rotates by θ, the new projection of A on the rotated camera CCD is located at h_2 from the center of the CCD. To predict the motion vector, we calculate the image movement $(h_2 - h_1)$ due to the rotation θ. Assume that

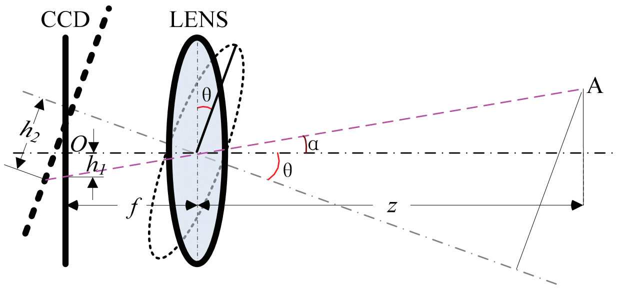

Figure 3. Effects of a moving camera

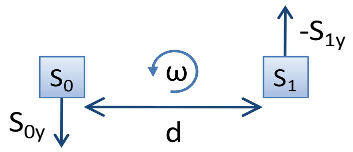

Figure 4. Two sensors (S_0 and S_1) placed apart

the camera focal length is *f*, a straightforward geometric and optical analysis leads to:

$$\Delta h = h_2 - h_1 = f \cdot \{\tan(\alpha + \theta) - \tan\alpha\} \approx f \cdot \theta \cdot \sec^2(\alpha)$$
$$\approx f \cdot \theta \quad \text{(For a small } \alpha \text{ and } \theta) \qquad (0)$$

Therefore, we only need *f* and θ to estimate the image movement. The focal length *f* of the camera is an intrinsic parameter of the camera and can be easily supplied by the camera. However, in our project, no such mechanism is provided with the camera. We have to resort to camera calibration tools to get these parameters [2].

The next key technical component of SenseCoding is the estimation of θ from the accelerometer readings for the vertical and horizontal directions, θ_v and θ_h, respectively.

For vertical direction: A single three-axis accelerometer is sufficient for calculating the vertical movement of the camera. The effect of the earth's gravity on acceleration measurements in three axes, a_x, a_y, and a_z, (Figure 2) can be utilized to calculate the static angle of the camera. For instance, when the camera rolls down from the illustrated position, a_x will increase and a_z will decrease. We can calculate the vertical angle of the camera by: $\alpha = \tan^{-1}[a_x/(a_y{}^2 + a_z{}^2)]$. We then calculate the change in angle, θ_v, by deducting the angle (α) of two subsequent frames (*n*, *n-1*).

$$\theta_v = \alpha_n - \alpha_{n-1} \qquad (1)$$

For horizontal direction: While a single accelerometer is enough to calculate the vertical angle of the camera, it cannot provide the horizontal angle. To overcome this limitation, we use two accelerometers placed apart from each other, to measure the angular acceleration of the device. According to Figure 4, we can calculate the angular acceleration ($\dot{\omega}$) of the device in the horizontal direction using measurements from the two accelerometers: $\dot{\omega} = (S_{0y} - S_{1y})/d$, where S_{0y} and S_{1y} are the acceleration measurements in the y direction of the two accelerometers. Assuming the time between two subsequent frames is *t*, for frame *n* we have $\theta_h(n) - \theta_h(n-1) = \dot{\omega} \cdot t = \frac{S_{0y} - S_{1y}}{d} \cdot t = k \cdot (S_{0y} - S_{1y})$, So we have:

$$\theta_h = \theta_h(n) = \theta_h(n-1) + k \cdot (S_{0y} - S_{1y}) \qquad (2)$$

In conjunction with Eq. (0), for each frame n, we can calculate the horizontal motion vector Δh_h according to the motion vector of the previous frame $\Delta h_h(n-1)$ and the accelerometer readings. As we already know $\Delta h_h(n-1)$ when we encode the current frame, and we can get the S_{0y} and S_{1y} from the sensor board, the only variable unknown is k. It can be calculated by knowing the frame rate and the distance between two accelerometers. However, we experimentally found the optimum k.

2.2 Sensor-Simplified Motion Estimation

Once we have the image movements in vertical and horizontal direction, Δh_v and Δh_h, from Eq. (0), (1) and (2), we can simplify the motion estimation procedure in video encoding by reducing the motion search window size. If Δh_v and Δh_h are absolutely accurate and the objects are static, the search window size can be reduced to a single pixel. However, acceleration data has noise. And objects in the camera view can move a very small number of pixels in the time period between two adjacent frames, i.e., *1/25*s. Therefore, SenseCoding employs a search window size greater than one, but significantly smaller than that required by the original full search, for comparable compression results.

We apply SenseCoding only to inter (P- and B-) frames and encode intra (I-) frames using the conventional method. To apply SenseCoding to blocks in inter frames, we first use the image movement $(\Delta h_v, \Delta h_h)$ predicted by the algorithm discussed in Section 2.1 as an initial search center. We then use the native full search algorithm for local motion estimation and produce an optimal motion vector. Note that for all the blocks in the same frame, the initial search center would be the same and be calculated once from corresponding acceleration data. Since the initial search center is usually near the optimal value, we can use a much smaller search window than conventional approach without sacrificing quality.

3. PROTOTYPE IMPLEMENTATION AND EXPERIMENTAL EVALUATION

To evaluate SenseCoding, we have implemented a prototype using an in-house designed sensor board [3] and a digital camcorder. We implement SenseCoding based on the MPEG2 reference encoder Test Model 5 [4] for encoding the raw video sequences.

3.1 Prototype Implementation

We built a sensor board ourselves because of the lack of access to the camcorder's built-in accelerometer and the need for two three-axis accelerometers for accurate motion vector estimation, as described in Section 2.1. The sensor board was based on interconnecting an in-house designed Bluetooth sensor [3] with a development board from Kionix (KXM52-1050). The sensor board employs a TI MSP430 microcontroller to read the two 3-axis accelerometers with 12-bit per sample and 64 samples per second. It sends the readings through Bluetooth to a PC in real time.

We use a hand-held camcorder to capture video sequences. The resolution is 576x480, and the frame rate is 25fps. We convert the captured sequences to raw video format in the post processing stage on the host PC. Since the sampling rate of the accelerometer is higher than the frame rate of the video, and acceleration data may have some noise, we employ a low-pass filter and linear interpolation to calculate the corresponding sample for each video frame. We firmly bundle the sensor board with the digital camcorder so that the sensors and the camcorder lens are aligned in the same way as illustrated by Figure 2.

Table 1. Video Sequences for systematical recording

Object / Camera	Still	Moving
Keep almost still	**Clip01**	**Clip02**
Slow Vertical Movement	**Clip03**	**Clip04**
Fast Vertical Movement	**Clip05**	**Clip06**
Slow Horizontal Movement	**Clip07**	**Clip08**
Fast Horizontal Movement	**Clip09**	**Clip10**
Irregular Movement	**Clip11**	**Clip12**

Synchronization between video sequence and its corresponding acceleration data is very important. While it is straightforward on an integrated hardware, our hardware prototype is limited in that the video and its corresponding acceleration are collected separately: video captured by the camcorder and the acceleration data captured by the sensor board but stored by a PC. We employ a simple yet effective punch method to do the synchronization. For each recording, we give our prototype a quick, moderate punch before and after the recording. The punch produces a visibly scene glitch in the video sequence and a visibly jolt in the acceleration data. We assume the glitch and the jolt are synchronized and manually synchronize the other parts according to the sample rate of the sensor board and the frame rate of the camcorder.

We modified the MPEG-2 reference encoder [4] in the motion estimation routine to utilize the acceleration data during video encoding, as shown in Figure 1. For each prediction frame (P- and B- frame), we calculate the global horizontal and vertical motion vectors from acceleration readings according to Section 2. For each sequence, we expect that the original encoder will produce bitstreams with the same bitrate and different video quality versus the motion estimation search range. A larger search range will produce smaller residual error in motion estimation and thus better overall video quality. We demonstrate that our approach will produce the same video quality with much smaller search range.

3.2 Experimental Evaluation

We have systematically collected 12 video clips with different camera and object movements, as described in Table 1.

Because SenseCoding only modifies the motion estimation, we shall focus on the impact of SenseCoding on the motion estimation part. Therefore, we employ the Mean Sum of Absolute Difference (MSAD) after motion estimation, instead of the overall PSNR, to evaluate the effectiveness of SenseCoding. That is, we calculate the sum of absolute difference (SAD) between the original macroblock and the predicted macroblock by motion estimation, and then we average the SAD by all the macroblocks in P- and B- frames to obtain MSAD. We measure the computation load of video encoding in terms of the runtime, or *total encoding time*, on a Windows-based PC with a 2.33GHz Intel Core 2 Duo processor and 4GB memory.

SenseCoding has very small overhead in hardware cost and acceleration data processing. Accelerometers are increasingly low power (less than 1mWatt) and low cost (around $10 for small quantities). Their power consumption is negligible in comparison to the much higher power consumption by the processor for encoding (often several hundred mWatts or higher). We observe that the acceleration data processing by SenseCoding is very efficient,

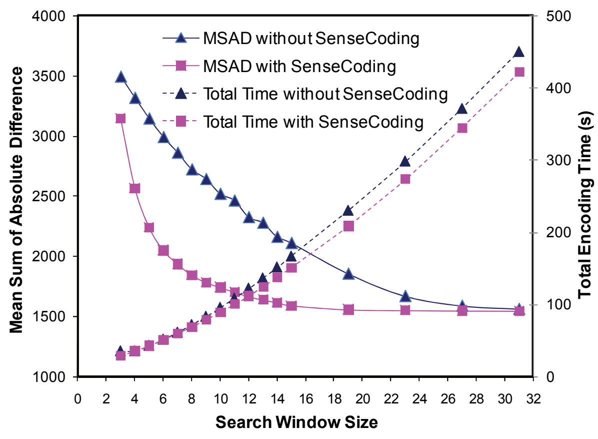

Figure 5. Search Window Size vs. MSAD and Total Encoding Time with and without SenseCoding (Clip12)

accounting for less than 1% of time taken by motion estimation. This is because SenseCoding estimates motion vectors for global motion and only once for each frame. We consider the computation load of acceleration data processing negligible.

We now examine the reduction in computation achieved by SenseCoding for the benchmark clips. For each clip, we encode it with and without SenseCoding for a range of search window size (from 3 to 32 pixels). For each encoding, we record the MSAD and the total encoding time. Figure 5 present the tradeoffs between the search window size and the achieved MSAD and encoding time for Clip12. The results for other clips are similar. From the figure, it is clear that a larger search window leads to increased encoding time and usually leads to reduced MSAD; <u>the application of SenseCoding leads to much lower MSAD for the same search window size and therefore much less encoding time for the same MSAD.</u>

Table 2 summarizes the speedup of the whole encoding process by SenseCoding for all benchmark clips. It presents the PSNR and total time of encoding procedure that SenseCoding achieves with the same MSAD of the conventional encoder with a full search window of 11x11 pixels. It shows that SenseCoding produces the same or even slightly better PSNR and is 2 to 3 times faster, while achieving the same MSAD. It is important to note that SenseCoding speeds up encoding by over two times even for clips with a moving object, highlighting its strength in capturing global motion.

4. DISCUSSIONS AND CONCLUSION

While we show that SenseCoding can speed up video encoding by two to three times, we acknowledge that our work is limited in the following aspects due to limitations in the space, scope, and available hardware. First, we only considered angular camera movements. However, a camera can have six degrees of freedom. Second, we evaluated SenseCoding based on MPEG-2, rather than advanced standards such as H.264, and we only implemented a baseline full search for local motion estimation. Nevertheless, we believe our work has made the case for sensor-assisted multimedia processing and paved the way for further investigations.

Table 2: Computational saving for the benchmark clips

Clip	Conventional Encoding		SenseCoding with Equivalent MSAD		Speedup (X)
	PSNR	Total Time (s)	PSNR	Total Time (s)	
01	27.7	73.1	27.6	30.9	2.37
02	27.6	73.0	27.5	35.4	2.06
03	27.7	100.0	28.4	33.8	2.96
04	29.6	104.5	29.9	48.8	2.14
05	28.6	101.8	29.4	34.1	2.99
06	29.2	106.4	30.2	34.5	3.08
07	27.2	93.3	28.8	33.0	2.82
08	26.5	90.8	27.7	43.3	2.10
09	26.1	89.5	27.2	37.5	2.39
10	25.8	92.2	27.0	32.5	2.84
11	28.0	103.3	28.9	41.4	2.50
12	27.6	107.8	28.8	42.7	2.53

We showed that the use of accelerometer can speed up video encoding by two to three times, leading to reduced computing hardware requirement and improved battery lifetime for handheld video capturing devices. An important trend in the consumer electronics is that more and more components and functionalities are integrated into a single device. Conventional wisdom would expect the computational complexity and thus energy consumption increase monotonously with the number of integrated components and functionalities. Our work has shown, for the first time, that the system-level computational complexity thus energy consumption can be effectively reduced by leveraging the synergy between different modalities, acceleration and vision in our case. We believe that there is plenty of *redundant computation* in a multi-component multimedia device such as video phone, which could be removed by exploiting a similar methodology presented in this paper. Our work has shown a new direction for the industry practitioners to mitigate the pressure on device energy efficiency.

5. ACKNOWLEDGMENTS

The work by the Rice team was supported in part by NSF awards CNS/CSR-EHS 0720825 and IIS/HCC 0713249 and equipment donation from Texas Instruments. The work by the NUS team was supported by Singaporean MOE grant with the WBS number of R-252-000-236-112. We thank Prof. Nam Ling and Mr. Yicheng Huang for helpful discussions.

6. REFERENCES

[1] Guang-ming Hong, Ahmad Rahmati, Ye Wang, and Lin Zhong, "SenseCoding: Accelerometer-assisted motion estimation for efficient video encoding," Joint Technical Report, School of Computing, National University of Singapore and Dept. of Electrical & Computing Engineering, Rice University. http://www.recg.org/publications/hong08sensecoding.pdf

[2] Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/

[3] Rice Orbit Sensor Platform, http://www.recg.org/orbit/

[4] MPEG2 Test Model 5 (TM5), http://www.mpeg.org/MPEG/MSSG/tm5/