SaVE: Sensor-assisted Motion Estimation for Efficient H.264/AVC Video Encoding

Xiaoming Chen[1], Zhendong Zhao[1], Ahmad Rahmati[2], Ye Wang[1] and Lin Zhong[2]

[1]School of Computing
National University of Singapore
117590 Singapore

{chenxm, zhaozd, wangye}@comp.nus.edu.sg

[2]Department of Electrical & Computer Engineering
Rice University
Houston, TX, 77005

{rahmati, lzhong}@rice.edu

ABSTRACT

Motion estimation is a key component of modern video encoding and is very compute-intensive. We present a novel Sensor-assisted Video Encoding (SaVE) method to reduce the computational complexity of motion estimation in H.264/AVC encoders, leveraging accelerometers and digital compasses that are increasingly available on mobile devices. Using these sensors, SaVE calculates the rotational movement of a camera and then infers the global motion in the camera image sensor; it subsequently employs the estimated global motion to simplify the state-of-the-art motion estimation algorithms, UMHS and EPZS used in H.264/AVC encoders. We have constructed a prototype of SaVE and report extensive evaluation of it. Our experimental results show that SaVE can reduce the computations of UMHS and EPZS algorithms by up to 27% and 18%, respectively, while achieving the same or better video quality.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Video

General Terms

Algorithms, Performance, Design

Keywords

Sensor, Digital Compass, Accelerometer, Motion Estimation, Video Encoding, H.264/AVC, MPEG

1. INTRODUCTION

Video cameras have already become a standard component of Smartphones and other handheld devices. Amateur video clips captured by such cameras have populated social networking portals such as YouTube and enabled amateur journalism such as iReport. Yet, capturing videos on mobile devices is compute-intensive and therefore power-hungry. A key compute-intensive module in modern video encoding is *motion estimation*. Because the same object may appear in consecutive frames but at different locations within the frames, motion estimation seeks to identify blocks from consecutive frames that match each other and subsequently eliminate redundancy. In modern video-coding standards such as H.264/AVC, motion estimation may examine a frame for block matching from multiple reference frames and using multiple block sizes [1]. Not surprisingly, the power and computational cost of video encoding is posing a significant challenge to video capturing on mobile devices.

Our solution toward addressing this challenge is Sensor-assisted Video Encoding, or SaVE. SaVE leverages low-power sensors to estimate camera movement; and subsequently applies the estimation to significantly simplify motion estimation. SaVE is motivated by the following observations. Firstly, the motion of an object in video frames can be decomposed to *global motion,* introduced by camera movement and *local motion* introduced by the movement of the object itself. In many video sequences, particularly in amateur-captured video clips, global motion due to camera movement, particularly rotation, is very common. Secondly, modern mobile devices (e.g. the HTC G1) have embraced ultra low-power and low-cost sensors, including digital compasses and accelerometers. These sensors can efficiently provide accurate information regarding the camera movement.

Our previous work [2] presented the preliminary results on employing sensors to improve video encoding. It detected camera rotational movement with a pair of tri-axis accelerometers, and showed that it can effectively improve the Full Search algorithm in MPEG2. In this work, we present a comprehensive study on the use of sensors in video encoding with sophisticated motion estimation algorithms, in particular in the H.264/AVC framework that is extremely relevant to resource-constrained mobile devices.

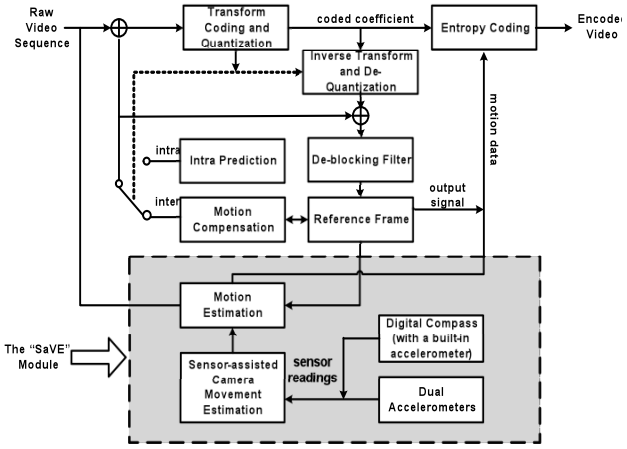


Figure 1. Basic structure of a H.264/AVC encoder together with the proposed SaVE scheme

ACM Multimedia 09, October 19–24, 2009, Beijing, China.

Our solution is the SaVE scheme, which advances our previous work [2] not only in its sophisticated use of movement estimated from sensors, but also in its exploration of new sensors, in particular the digital compass.

Figure 1 shows the basic structure of a H.264/AVC encoder together with the SaVE scheme. SaVE employs either two accelerometers or the combination of one accelerometer and one digital compass to estimate camera rotation. Using these estimations, it infers the global motion in the subsequent frames. SaVE then utilizes the estimated global motion as predictors (initial search positions) in H.264/AVC motion estimation to reduce computation.

We have built a prototype using a commercial camcorder with two different combinations of the digital compass and accelerometers. Our experimental evaluations show that SaVE can significantly reduce the complexity of H.264/AVC motion estimation with UMHS [3-4] and EPZS [1, 5] algorithms by up to 27% and 18%, respectively. To the best of our knowledge, SaVE is the first publicly reported attempt in using sensors to improve H.264/AVC video encoding. It embodies a new research direction in multimedia processing that explicitly employs physical information obtained from sensors.

The rest of the paper is structured as follows. In Section 2, we outline background information and related work in video coding. In Section 3, we describe our method for camera movement estimation. The technical details of utilizing the estimated global motion as a SaVE predictor in H.264/AVC are explained in Section 4. We present a prototype of SaVE and the experiment setup in Section 5 and we present experimental results based on the prototype in Section 6. We address the limitation of SaVE in Section 7 and conclude in Section 8.

2. BACKGROUND AND RELATED WORK

We first provide background for SaVE and discuss related work.

2.1 Motion Estimation

When the camera or an object in its view moves, the captured image moves too. Therefore, a part of an image may appear in multiple consecutive video frames, at different but close locations, creating an opportunity that modern video encoding technologies leverage to compress the video sequence. The key to such compression is *motion estimation*, which seeks to identify blocks in a frame that matches those in a reference frame at different but close locations.

The naïve Full Search algorithm attempts to locate the moved image by searching all possible positions within a certain distance (search window). While the naïve Full Search yields optimal search results, it is extremely compute-intensive. There has been extensive research on more efficient motion estimation. Known techniques can be classified into three categories. The first category seeks to reduce the number of candidate blocks in the search window, e.g. three-step search (TSS) [6], new 3-step search (N3SS) [7], four-step search (FSS) [8], diamond search (DS) [9], cross-diamond search (CDS) [10], and kite cross-diamond search (KCDS) [11]. The second category attempts to reduce the number of pixels involved in the block comparison of each candidate, e.g. partial distortion search (PDS) [12], alternative sub-sampling search algorithm (ASSA) [13], normalized PDS (NPDS) [14], adjustable PDS (APDS) [15], and dynamic search window adjustment [16]. The third category takes a hybrid approach of the first two, e.g. Motion Vector Field Adaptive Search Technique (MVFAST) [17], predictive MVFAST (PMVFAST) [18], Unsymmetrical-cross Multi-Hexagon-grid Search (UMHS) [3-4] and Enhanced Predictive Zonal Search (EPZS) [1, 5]. In particular, UMHS and EPZS are very efficient. In comparison to the Full Search algorithm, they can reduce the computational requirement by 90%, while maintaining a fairly good video quality [1, 3]. UMHS and EPZS are adopted in the H.264/AVC standard. In this work, we use the implementation of UMHS and EPZS in H.264/AVC version JM14.2 as the baselines.

Predictive motion estimation is very efficient in reducing the number of candidate blocks. Instead of attempting all motion vectors within a search range, the efficiency of the motion estimation algorithm can be improved by only checking a few highly promising predictors, which are expected to be close to the best motion vector. With simple yet efficient checking patterns, the motion estimation algorithms can find the optimal motion vector around the predictor quickly with a reliable early-termination criterion [3, 5]. Predictive motion estimation algorithms, such as UMHS and EPZS, have provided predictors based on block correlations, such as median predictor and neighboring reference predictor, among others. The median predictor is believed to be more reliable and is more likely to be accurate [5] in comparison to others. A median predictor refers to the median motion vector of the top, left and top-right (or top-left) neighbor blocks of the current block, which is frequently used as the initial search predictor and also for motion vector prediction encoding [5].

However, the predictors in both UMHS and EPZS are obtained by estimating the motion vector purely based on temporal or spatial correlations, which may not be reliable. There are quite a few methods, e.g. [19-21], explored the way of applying a global motion estimation (GME) process to obtain an initial position (we regard as a predictor) for local motion estimation. However, the GME of these methods requires additional operations (e.g. frame matching), leading to intensive computations. The development of sensor technology has inspired us to estimate camera movement (and apply the GME) by reliable sensors through very simple calculations, and then utilize the estimated movement as predictors in motion estimation. Our previous work [2] presented early results on employing sensors to improve video encoding. In this work, we provide a comprehensive treatment of SaVE and present solutions that significantly improves H.264/AVC video encoding, the state-of-the-art standard for mobile devices.

2.2 Sensors

In addition to our previous work [2], [22] also presented an idea of using sensors to assist compression for vehicle-captured videos. This work focuses on detect and use the palpable vehicle movement in video compression. Our SaVE scheme, however, is aiming at detecting and utilizing the rotational movement in amateur-captured videos for simplifying H.264/AVC encoding.

SaVE employs ultra low-power and low-cost sensors to estimate camera rotations. Assuming negligible linear acceleration of the camera, a single tri-axis accelerometer can provide the vertical angle (respect to the ground), but it is unable to provide the horizontal angle. Dual accelerometers placed apart can measure rotational acceleration, both horizontally, and vertically. Similarly, a gyroscope can measure rotation speed. However, they are unable to provide the absolute angle of the device. Integrating the rotation speed or double integrating the rotational acceleration to calculate

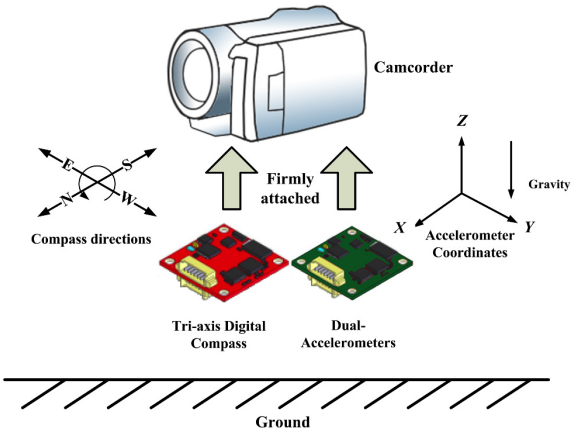


Figure 2. SaVE prototype

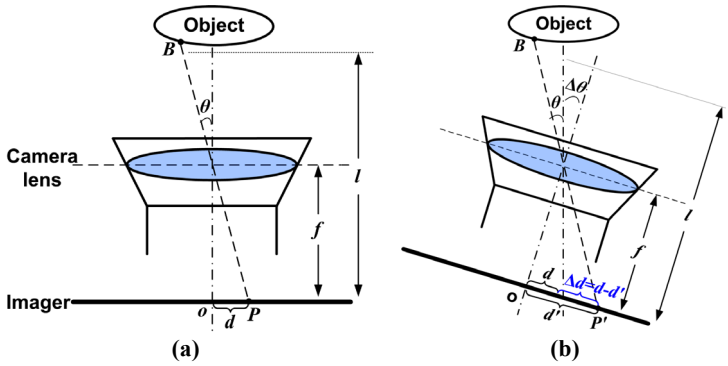


Figure 3. From camera movement to projection movement in the video frame

angle is impractical because sensor noise rapidly becomes egregious. In contrast, a tri-axis digital compass can directly measure both horizontal and vertical angles. However, digital compasses may be subject to external influence, such as nearby magnets and ferromagnetic objects, and radio interference (e.g. from mobile phones), as discussed in Section 7.1. Therefore, SaVE uses readings from a single accelerometer for the vertical angle. For the horizontal angle, we have implemented SaVE to use either the single tri-axis digital compass or the two accelerometers. The rotational acceleration measured from the dual accelerometers is directly used in SaVE, without integration.

The power consumption of digital compasses and accelerometers is very small in comparison to the power required for H.264/AVC video encoding. For example, according to our measurements, the commercial sensor board in our prototype that has a digital compass and a tri-axis accelerometer consumes 66mW, and our custom sensor with two accelerometers consumes 15mW without Bluetooth. Furthermore, much of the power is consumed by components that will be obsolete when the sensors are properly integrated into the camera. For example, the Honeywell HMC6042/1041z tri-axis compass consumes 23mW, and each of the KXM52 tri-axis accelerometers used in our custom sensor board consume less than 5mW. All three sensors would add up to approximately 3% of the video encoding power of a typical H.264/AVC video encoding chip, consuming over a Watt [23]. Since sensors are increasing embedded on mobile devices (to decide screen orientation or for navigation purposes), this power consumption would not be considered as overheads caused by merely our SaVE scheme.

3. GLOBAL MOTION ESTIMATION

Camera movement can be linear or rotational. Linear movement is introduced by camera location change, while rotational movement can be introduced by tilting, i.e. turning the camera vertically, or by panning, i.e. turning the camera horizontally. In amateur video capturing with handheld devices, rotational movement is extremely common. Camera rotation will lead to significant global motion in the video frames. In this work, the global motion is described by a vector, or Global Movement Vector (*GMV*), specifying both vertical and horizontal movements of objects in two successive frames due to camera movement. In this section, we present several sensor-assisted methods of camera movement estimation and then address *GMV* estimation from camera movement. In Section 4, we will further apply the *GMV* estimation to motion estimation.

3.1 Rotational Change Estimation

SaVE employs a single tri-axis accelerometer to estimate absolute vertical angle. SaVE can employ two different methods for calculating the horizontal movement, using a digital compass to estimate absolute horizontal angle, or by using dual accelerometers to estimate rotational acceleration. In this subsection, we briefly present the mathematical foundations of calculating the horizontal and vertical angles, and the rotational acceleration from raw sensor data.

3.1.1 Vertical Angle (Single Accelerometer)

The vertical angle of the camera can be calculated using a single tri-axis accelerometer, similar to [2]. The effect of the earth's gravity on acceleration measurements in three axes, a_x, a_y, and a_z, can be utilized to calculate the static angle of the camera accurately. For instance, when the camera rolls down from the illustrated position in Figure 2, a_x will increase and a_z will decrease. The following equation can calculate the vertical angle P_n of the camera at frame F_n :

$$P_n = \tan^{-1}\left\{\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right\} \quad (1)$$

In this equation, a_x , a_y and a_z are the acceleration readings from a tri-axis accelerometer (Figure 2). Hence the vertical rotational change $\Delta\theta_v$ for 2 successive video frames F_n and F_{n-1} can be calculated as:

$$\Delta\theta_v(n) = P_n - P_{n-1} \quad (2)$$

3.1.2 Horizontal Angle (Compass)

The horizontal angle of the camera can also be calculated with a method identical to the above, using the readings from a tri-axis compass instead of the accelerometer. It would effectively calculate the angle of the camera with respect to the magnetic north instead of ground, producing the horizontal angle. Therefore, the horizontal rotational movement $\Delta\theta_h$ between F_n and F_{n-1} can also be directly obtained as:

$$\Delta\theta_h(n) = H_n - H_{n-1} \quad (3)$$

where H_n and H_{n-1} are the horizontal angles at frame F_n and F_{n-1} from the compass readings.

3.1.3 Horizontal Acceleration (Two Accelerometers)

In [2], we showed that a pair of properly located accelerometers, by sensing rotational acceleration, can provide information regarding *relative horizontal rotational movement*. For convenience of discussion, we summarize the calculation of the horizontal rotational movement $\Delta\theta_h$ below. For frame F_n we have:

$$\Delta\theta_h(n) = \Delta\theta_h(n-1) + k \cdot (S_{0y} - S_{1y}) \quad (4)$$

In this equation, S_{0y} and S_{1y} are the acceleration measurements in the *Y* direction from the dual accelerometers, respectively, and k is a constant that can be directly calculated from the distance between the two acceleration sensors, the frame rate and the pixel-per-degree resolution of the camera.

3.2 Global Movement Vector (*GMV*) Estimation

When a camera rotates, the projection of an object in the view to the camera image sensor also moves, as illustrated in Figure 3. The movement of the projection on the image sensor can be described by the *GMV* that specifies the vertical and horizontal movements.

To calculate the *GMV*, we must understand the camera characteristics and build an optical model. In Figure 3 (a) and (b), *O* denotes the optical center of the camera image sensor; *f* denotes the focal length and *l* denotes the distance between the object to the camera lens; *B* is a point in the object. In (a), the projection *P* of point *B* on the image sensor is located at a distance of *d* from *O*; θ is the angle between the line *BP* and the perpendicular bisector of camera lens. The situation when the camera is turned by $\Delta\theta$ is shown in Figure 3 (b), where the new projection *P'* is located at *d'* from *O*. The movement for projections of point *B* on the imager can be calculated as $\Delta d = d - d'$. From the optical model, we can easily calculate *d* and *d'* with:

$$\begin{cases} d = f \cdot \tan\theta \\ d' = f \cdot \tan(\theta + \Delta\theta) \end{cases} \quad (5)$$

Hence the projection movement Δd can be calculated as:

$$\Delta d = d - d' = f \cdot \{\tan(\theta + \Delta\theta) - \tan\theta\} \quad (6)$$

As $\Delta\theta$ is usually very small between two successive frames of a video clip, we have:

$$\tan(\theta + \Delta\theta) - \tan\theta \approx \Delta\theta \cdot \frac{d(\tan\theta)}{d\theta} = \Delta\theta \cdot \sec^2(\theta) \quad (7)$$

Thus we can obtain Δd as:

$$\Delta d \approx f \cdot \Delta\theta \cdot \sec^2(\theta) \quad (8)$$

Typically, θ ranges between zero and half of the Field of View (FOV) of the lens. Hence, for all camera lenses except for extreme wide-angle and fisheye ones, θ is reasonably small and Δd can be further reduced to:

$$\Delta d \approx f \cdot \Delta\theta \cdot \sec^2(\theta) \approx f \cdot \Delta\theta \quad (9)$$

From the above formulas, we have that f and $\Delta\theta$ are adequate to calculate the movement. We can then convert f in pixels by dividing the calculated distance by the pixel pitch of the image sensor (denoted by f'). The current focal length of the camera, f, and the pixel pitch of the image sensor are intrinsic parameters of the camera, and known to the camera at any time without extra cost. However, as our camera does not report these parameters to us, we have adopted an easy-to-use tool based on MATLAB from [24-26] to obtain these parameters (this is not a part of video encoding and will not be required by real applications).

Having the horizontal and vertical rotation $\Delta\theta_h$ and $\Delta\theta_v$, we can calculate the *GMV* for 2 successive frames F_n and F_{n-1} as:

$$GMV_n(\Delta d_h, \Delta d_v) = (f' \cdot \Delta\theta_h, f \cdot \Delta\theta_v) \quad (10)$$

where Δd_h and Δd_v are the movement of the projection along the horizontal and vertical directions, respectively; f' is the focal length in pixels.

4. SaVE MOTION ESTIMATION

With the *GMV* estimation, we next describe how SaVE applies the *GMV* to H.264/AVC motion estimation.

4.1 *GMV* Per Reference Frame

Multiple-reference-frame motion vector prediction is an important feature of H.264/AVC. For a video frame F_n, a single *GMV* calculated for merely its previous reference frame F_{n-1} is inadequate to obtain accurate predictors in other reference frames. To address this problem, SaVE dynamically calculates the reference-dependant *GMV*s. For example, when using F_{n-k} as the reference frame, GMV_n^k for the frame F_n can be calculated as:

$$GMV_n^k(\Delta d_h, \Delta d_v) = \sum_{i=n-k}^{n} GMV_i \quad (11)$$

Using dynamic *GMV*s allows motion estimation to be started from different positions for different reference frames.

4.2 SaVE Predictor Insertion

To improve motion estimation, we can insert the calculated $GMV(\Delta d_h, \Delta d_v)$ into the UMHS and EPZS algorithms as the SaVE predictor (*SPx*, *SPy*). In our SaVE scheme, the SaVE predictor is given preference to be attempted first before trying those original predictors in UMHS and EPZS. In particular, we have:

$$\begin{cases} SPx = x + \Delta d_h \\ SPy = y + \Delta d_v \end{cases} \quad (12)$$

where x and y are the horizontal and vertical coordinates of the current block to be encoded.

We examine two strategies to use the SaVE predictor as the initial search position. The first, we shall call Arbitrary Strategy, is adopted from our preliminary work reported in [2]. The Arbitrary Strategy employs the SaVE predictor as the initial predictor for all macroblocks in a video frame. The obvious drawback of the Arbitrary Strategy is that it excessively emphasizes on the measured global motion while ignoring the local motion and the correlations between spatially adjacent blocks. Our initial experiments showed that the Arbitrary Strategy will not provide significant gains over UMHS and EPZS.

Therefore, in order to consider both global and local motion, we present a *Selective Strategy* based on our examination of many

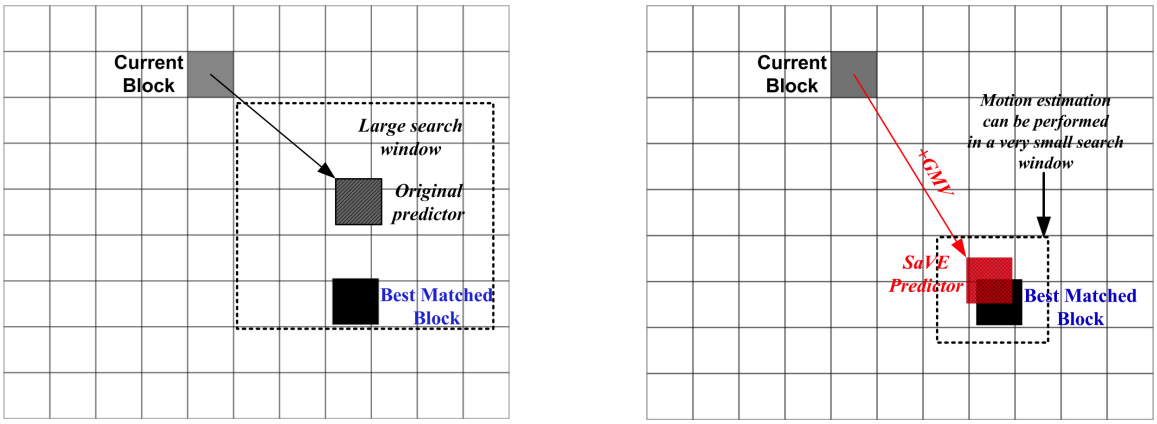
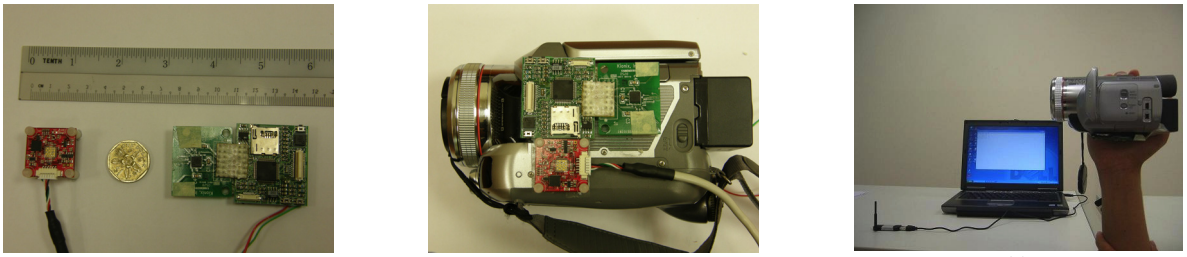


Figure 4. SaVE predictor reduces the search window of motion estimation: (a) Original predictor; (b) SaVE predictor



(a) **(b)** **(c)**

Figure 5. Prototype implementation of SaVE: (a) the commercial board with a digital compass and a tri-axis accelerometer (left) and the in-house built board with dual tri-axis accelerometers (right) used in SaVE; (b) camcorder and the digital compass and dual accelerometers bundled for video capturing; and (c) the prototype in working

insertion strategies, i.e. attempting the insertion with different number of blocks and in different locations of the picture. The Selective Strategy inserts the SaVE predictors into the top and left boundary of a frame. As mentioned in Section 2, the original predictors in UMHS and EPZS can *spread* the current motion vector tendency to the remaining blocks in the *lower* and *right* part of the video picture, because they highly rely on the *top* and *left* neighbors of the current block, As a result, the Selective Strategy spreads the global motion estimated from sensors to the entire frame. Let $MB_{(i,j)}$ denote the macroblock located at i^{th} column and j^{th} row in the video picture ($MB_{(0,0)}$ is regarded as the top-left macroblock). The Selective Strategy will use the SaVE predictor as the initial search position only when ($i < n$ OR $j < n$) and will use the original predictors in UMHS and EPZS otherwise. We have experimentally determined that n=2 yields good results.

The Selective Strategy improves UMHS/EPZS due to the following two reasons. First, it benefits from the SaVE predictor that reflects the global motion estimated from sensors. Second, it respects the spatial correlations of adjacent blocks by using the original UMHS and EPZS predictors.

4.3 Effectiveness of SaVE Predictor

We next explain how the SaVE predictor can reduce the complexity of motion estimation using Figure 4. Figure 4 (a) shows that UMHS original predictor has no knowledge of global motion, and therefore the motion estimation may just start from the neighboring motion vectors. In this case, original UMHS may require a fairly large search window to identify the best matching block (black) for a given block (grey), especially for clips that contain fast camera movement. In another word, original UMHS is not able to quickly reach the real predictors.

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

As we reviewed in Section 2.1, the methods in [19-21] employ an additional GME process and therefore are able to provide more accurate predictors. Consequently, this kind of algorithms can reduce the search window size. Our SaVE scheme also takes the advantage of the GME process but it obtains the global motion and the predictor in a completely new way: the camera global motion is precisely estimated from reliable sensors. With SaVE, the motion estimation can immediately start from a position that is close to the real predictor. Hence, SaVE will only need a much smaller search window in motion estimation, as shown in Figure 4 (b). This will be confirmed in our prototype-based experimentation, reported later.

5. PROTOTYPE IMPLEMENTATION

5.1 Hardware Implementation

In order to evaluate SaVE, we have implemented a prototype with a consumer-grade camcorder and two sensor boards (Figure 5). One sensor board was custom designed and carries dual tri-axis accelerometers. The other board, an OS5000 from OceanServer Technology [27], is a commercial tri-axis digital compass with an embedded tri-axis accelerometer. The commercial sensor

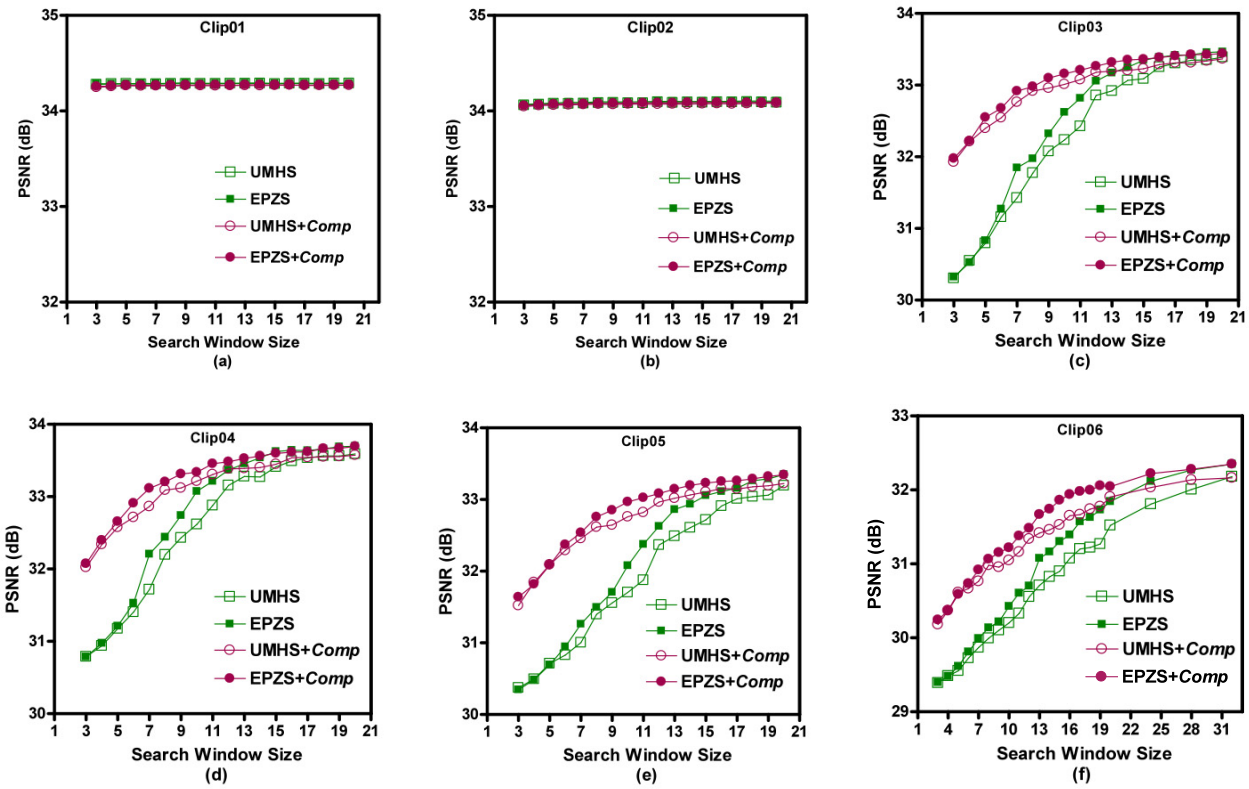


Figure 6. PSNR comparisons for UMHS, EPZS and SaVE-enhanced UMHS and EPZS (clips with vertical movement). Note that because SaVE/*Comp* and SaVE/*DAcc* are identical in vertical motion estimation, we only show the results of SaVE/*Comp*

computes and reports the absolute horizontal and vertical angles using its tri-axis compass and tri-axis accelerometer respectively. Our custom sensor outputs raw accelerometer readings and we perform the necessary calculations offline. Using the methods presented in Section 3, we have implemented SaVE with both boards, denoted as SaVE/*DAcc* (using dual accelerometers) and SaVE/*Comp* (using the digital compass).

The camcorder used in this prototype has a resolution of 576×480, and we set its frame rate to 25fps. Since the camcorder does not support raw video sequence format, we have converted the captured sequences into the YUV format with software. We firmly attach both sensor boards to the camcorder so that the sensor boards and the camcorder lens are aligned in the same direction as shown in Figure 5.

5.2 Data Collection and Synchronization

We have systematically captured 12 video clips with different combinations of global (camera) and local (object) motions, as described in Table 1. Snapshots of the clips are shown in Figure 8. The location motions were introduced by walking pedestrians.

We collect the sensor data while capturing the video clips and then synchronize them manually, because our hardware prototype is limited in that the video and its corresponding sensor data are collected separately: video were captured directly by the camcorder and the sensor data were captured by the digital compass or accelerometers but stored by a laptop. The synchronization between the dual accelerometers and video clips has been introduced in [2]. For the digital compass, we align the maximum recorded angle of the digital compass with the frame taken at largest vertical angle in a video clip.

We should note that this manual synchronization will not be required in real applications. On an integrated hardware implementation, it would be fairly straightforward to synchronize video and sensor readings, e.g., the sensor data recording and video capturing start simultaneously when a user presses the *Record* button of a camcorder or mobile device.

5.3 Software Implementation

To implement SaVE, we start from the standard H.264/AVC encoder (version JM 14.2), which implements up-to-date UMHS and EPZS algorithms. For each predictive frame (P- and B- frame), we employ SaVE predictors in UMHS and EPZS with the selective insertion strategy (n = 2) according to Sections 3 and 4. We then encode each sequence using the Baseline profile with variable block sizes and 5 reference frames. The Rate Distortion Optimization (RDO) is turned on. A Group of Picture (GOP) of 10 frames is used in the encoding. The first frame of each GOP is encoded as an I-frame and all the other 9 frames are encoded as P-frames. Each sequence was cut to 250 frames (10 seconds for 25 fps). All sequences were encoded with a fixed bitrate at 1.5Mbps. For each sequence, we expect that the original encoder will produce bitstreams with the same bitrate and different video quality when the search window size (SWS) varies: a larger search window will produce smaller residual error in motion estimation and thus better overall video quality.

The SaVE implementation is based on C language and consists of only about 200 lines of code in addition to the H.264/AVC encoder. Such simplicity makes it easy to be incorporated into practical encoding systems.

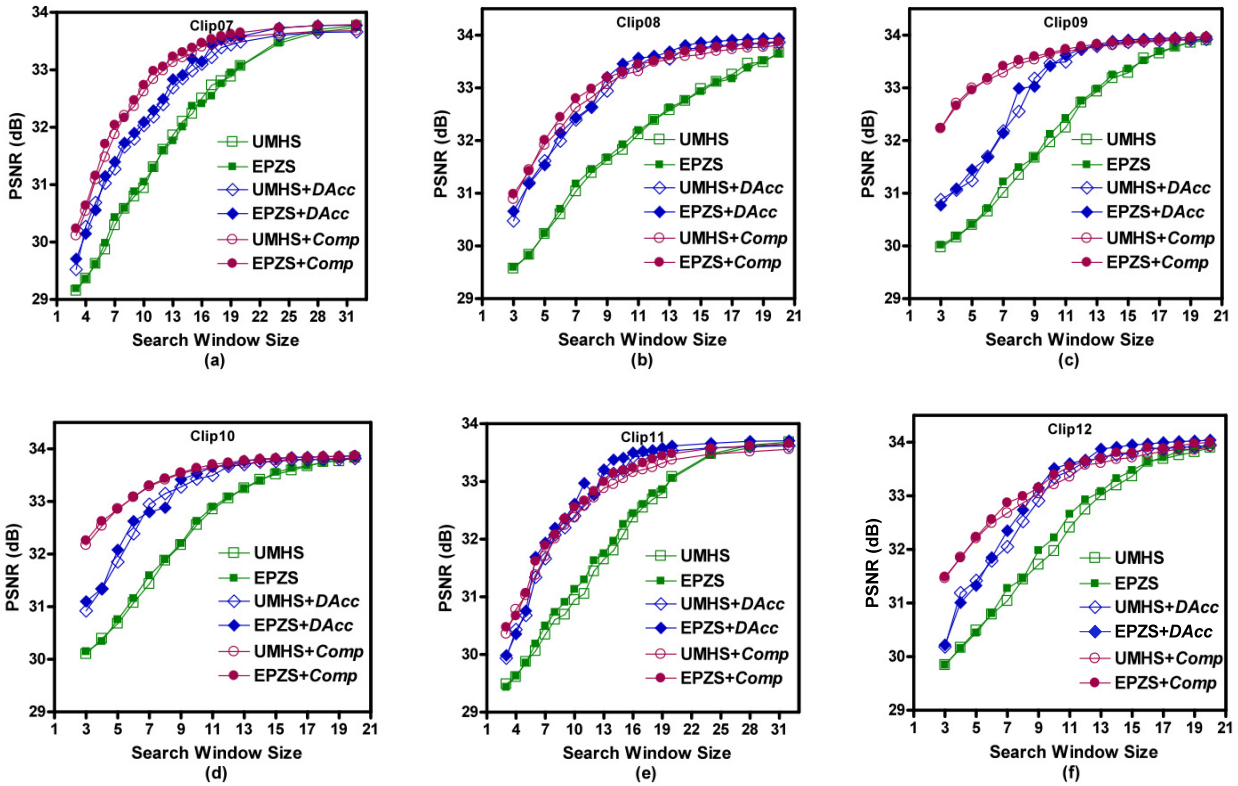


Figure 7. PSNR comparisons for UMHS, EPZS and SaVE-enhanced UMHS and EPZS (clips contains horizontal movement)

6. EXPERIMENTAL EVALUATION

We encode each clip collected with the hardware prototype with original UMHS and EPZS, and the enhanced algorithms with SaVE predictors, i.e. UMHS+*DAcc*, UMHS+*Comp*, EPZS+*DAcc* and EPZS+*Comp*. Here "+*DAcc*" and "+*Comp*" refer to SaVE predictors obtained by SaVE/*Comp* and SaVE/*DAcc* respectively. The search window size (SWS) ranges from ±3 to ±32 (denoted as SWS = 3 to SWS = 32). All encodings were carried on a PC with a 2.66GHz Intel Core 2 Duo processor and 4GB memory.

We next show the superiority of SaVE in terms of video quality improvement and computational reductions.

6.1 Video Quality Improvement

Peak Signal-to-Noise Ratio (PSNR) is an objective measurement of video quality. A higher PSNR usually indicates a higher quality. Figure 6 and Figure 7 present the PSNR gains obtained by SaVE in comparison to the original H.264/AVC encoder with UMHS and EPZS. For clips with only vertical movement, we only present the results obtained by using SaVE/*Comp*, as both the SaVE/*DAcc* and SaVE/*Comp* use a single accelerometer to calculate the vertical rotation. For clips containing horizontal movement, we compare the results obtained by SaVE/*DAcc* and SaVE/*Comp* respectively. For Clip06, Clip07 and Clip12, we show the results for SWS ranging from 3 to 31. For other clips we only show SWS = 3 to 20, as our SaVE will not provide gains over this range.

Still camera: Clip01 and Clip02

Clip01 and Clip02 were captured with the camera held still. While, none of the SaVE-enhanced algorithms can help in achieving higher PSNR, as there is no camera rotation. However, we can see that SaVE does not hurt the performance in such cases.

Vertical Movement: **Clip03 to Clip06**

Clip03 to Clip06 were captured with the camera moving vertically. With the same SWS, the PSNRs obtained by UMHS+*Comp* and EPZS+*Comp* are clearly higher than those of the original UMHS and EPZS, especially for small SWS. For example, when SWS = 5, the PSNR gains obtained by UMHS+*Comp* over UMHS are 1.61dB, 1.40dB, 1.38dB and 1.05dB for Clip03-06 respectively; when SWS = 11, the gains by EPZS+*Comp* over EPZS are 0.40dB, 0.25dB, 0.65dB and 0.78dB. UMHS+*Comp* and EPZS+*Comp* can maintain superior PSNR performance over the original algorithms until SWS ≥ 16 for Clip03 and Clip04, SWS ≥ 19 for Clip05 and SWS ≥ 28 for Clip06.

Horizontal Movement: **Clip07 to Clip10**

Clip07 to Clip10 were captured with the camera moving horizontally. We evaluated SaVE/*Comp and* SaVE/*DAcc*, and discovered that both the methods can achieve significant improvement over the original algorithms. For SaVE/*Comp* on one hand, the gains by UMHS+*Comp* over UMHS can be up to 2.59dB for Clip09. But this gain is achieved only when SWS = 5 is used According to our results, SaVE can particularly obtain gains when smaller SWS is used. For larger SWS, e.g. 11, UMHS+*Comp* still can achieve more than 1dB improvement for most of the clips. For SaVE/*DAcc* on the other hand, the performance of UMHS+*DAcc* and EPZS+*DAcc* can be close to UMHS+*Comp* and EPZS+*Comp* in some cases, e.g. for Clip08. But for clips with faster camera movement, such as Clip09 and Clip10, it appeared that the benefits of using UMHS+*Comp* and EPZS+*Comp* are quite obvious, especially at a small SWS.

Irregular Movement: **Clip11 and Clip12**

Table 2. CSWS, PSNR Gains, and Speedup achieved by SaVE-enhanced UMHS and EPZS when SWS = 3 and SWS = 11 for clips with vertical movement

SaVE-enhanced UMHS						
Clip	SWS = 3			SWS = 11		
	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)
03	8	+0.16	**+23.61**	14	+0.01	**+7.98**
04	7	+0.30	**+14.70**	14	+0.03	**+7.28**
05	8	+0.12	**+23.71**	15	+0.10	**+8.00**
06	9	+0.08	**+26.59**	16	+0.09	**+8.94**
SaVE-enhanced EPZS						
Clip	SWS = 3			SWS = 11		
	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)
03	8	0.00	**+12.32**	13	+0.04	**+3.47**
04	6	+0.54	**+7.58**	13	+0.01	**+3.21**
05	8	+0.14	**+11.76**	14	+0.09	**+3.01**
06	9	+0.02	**+13.51**	15	+0.08	**+5.08**

Clip11 and Clip12 were captured with irregular and random movements (real-world video capturing scenario). The figure shows that the SaVE-enhanced algorithms can achieve considerable PSNR gains over the original algorithms when SWS ≤ 24 (for Clip11) or SWS ≤ 18 (for Clip12). When medium SWSs are used, the PSNR gains are usually from 1.0dB to 1.5dB for Clip11 and 0.4dB to 1.6dB for Clip12.

The above results show that, with the current prototype, SaVE can provide reasonable PSNR gains when SWS ≤ 20 for most clips. When larger SWS (e.g. 24 to 32) is used, SaVE only shows a reduced improvement for Clip06, Clip07 and Clip11. However, these results only show the potential of our SaVE scheme. We expect the performance of SaVE will be further improved with an industrial implementation.

Figure 8 shows an example (frame 76 of Clip11) of decoded pictures by EPZS (27.01dB) and EPZS+*Comp* (31.42dB) with same SWS = 11. Due to the camera movement, the picture decoded by EPZS is highly blurred. It is obvious that the picture offered by EPZS+*Comp* has much better quality**.**

To sum up, since the estimated global motion is well-utilized, the SaVE predictor is often closer to the real predictor than others. Hence, in rate-distortion optimized motion estimation, SaVE is able to produce smaller block SAD, reduce the MCOST (which is the block SAD plus the motion vector encoding cost), and therefore to obtain a higher PSNR at a given SWS.

6.2 Speedup

We now present the results regarding the computation reduction realized by SaVE. We measure the computation load of encoding with the motion estimation time. The motion estimation time of UMHS and EPZS increases as SWS increases. Since the SaVE-enhanced algorithms using a small SWS, usually can achieve the same PSNR of the original algorithms using a much larger SWS, we can practically save the motion estimation time by reducing the SWS while maintaining the same video quality.

Vertical Movement: Clip03 to Clip06

Table 2 shows, for clips with only vertical movement, the speedup achieved by UMHS+*Comp* and EPZS+*Comp* over the original algorithms while obtaining the same or even higher PSNR. We have selected a very small SWS = 3 and a relatively large SWS = 11 for the SaVE-enhanced algorithms. The "CSWS" in Table 2 denotes the Corresponding SWS used in the original UMHS (EPZS) that is able to obtain the similar PSNR to UMHS+*Comp* (EPZS+*Comp*) using SWS = 3 or SWS = 11. It is important to note that UMHS+*Comp* with SWS = 3, usually can obtain *higher* PSNR than original UMHS with SWS = 7 to 9. This results in up to 26.59% motion estimation time being saved.

Horizontal / Irregular Movement: Clip07 to Clip12

In Table 3, we present and compare the results of UMHS+*DAcc*, UMHS+*Comp*, EPZS+*DAcc* and EPZS+*Comp* for clips that contain horizontal movement. We discovered that SaVE-enhanced UMHS and EPZS achieve speedups by up to 24.60% and 17.96% respectively. It is also discovered that using the digital compass is more stable and efficient than dual accelerometers in reducing the motion estimation time overall.

In summary, SaVE achieves significant speedups for the tested video clips, which represent a wide variety of combinations of global and local motions. Our SaVE scheme takes the advantage of traditional GME for predictive motion estimation, but it estimates the global motion with a novel approach. With very simple calculations, SaVE is able to remove a great amount of redundant computations from H.264/AVC motion estimation.

6.3 Extensive Local Motion

While we have showed the benefits of SaVE for various natural video clips, we have also considered one extreme circumstance – when the video clip contains very complicated and extensive local

Table 3. CSWS, PSNR Gains, and Speedup achieved by UMHS+*DAcc* and UMHS+*Comp* when SWS = 3 and SWS = 11 for clips that contain horizontal movement

SaVE-enhanced UMHS												
Clip	UMHS+*DAcc* (SWS = 3)			UMHS+*Comp* (SWS = 3)			UMHS+*DAcc* (SWS = 11)			UMHS+*Comp* (SWS = 11)		
	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)
07	4	+0.16	+13.88	6	+0.24	**+16.41**	16	+0.19	+5.92	18	+0.03	**+12.93**
08	5	+0.24	**+17.31**	6	+0.29	+15.16	19	+0.06	+8.47	17	+0.06	**+11.45**
09	6	+0.22	+17.84	10	+0.26	**+24.25**	18	0.00	+7.99	17	+0.04	**+11.61**
10	5	+0.24	+16.71	9	0.00	**+23.58**	17	+0.03	+7.53	16	+0.05	**+10.82**
11	5	+0.06	+16.21	7	+0.01	**+17.99**	20	+0.05	+11.20	17	+0.02	**+13.20**
12	4	+0.02	+13.79	8	+0.02	**+24.60**	17	+0.03	+6.07	14	+0.15	**+7.98**
SaVE-enhanced EPZS												
Clip	EPZS+*DAcc* (SWS = 3)			EPZS+*Comp* (SWS = 3)			EPZS+*DAcc* (SWS = 11)			EPZS+*Comp* (SWS = 11)		
	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)	CSWS	PSNR Gains (dB)	Speedup (%)
07	5	+0.09	+ 7.01	6	+0.25	**+11.61**	18	+0.07	+3.95	19	+0.03	**+9.34**
08	5	+0.42	+9.94	6	+0.28	**+10.79**	20	+0.04	+7.03	18	+0.07	**+7.65**
09	6	+0.06	+12.51	10	+0.12	**+13.91**	18	+0.04	+6.03	17	+0.05	**+6.75**
10	5	+0.34	+ 8.70	9	+0.06	**+13.68**	18	+0.03	+5.28	17	+0.02	**+6.70**
11	5	+0.13	+6.09	6	+0.29	**+10.94**	20	+0.15	**+17.96**	17	+0.06	+7.93
12	4	+0.07	+5.10	8	+0.02	**+13.64**	19	0.00	+5.29	15	+0.08	**+6.52**

(a)

(b)

Figure 8. Example of decoded pictures: (a) Picture decoded by EPZS with SWS = 11 (27.01dB) (b) Picture decoded by EPZS+*Comp* with SWS = 11 (31.42dB)

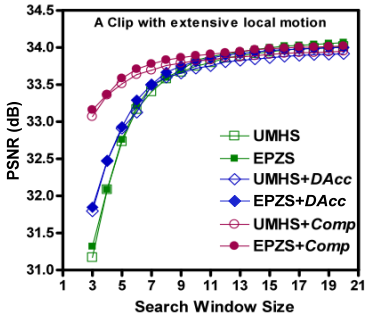


Figure 9. Performance of SaVE under extensive local motion

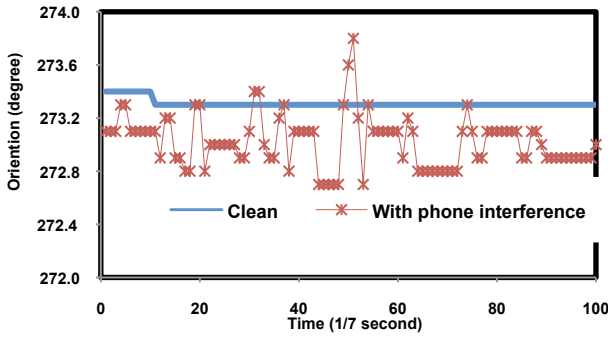


Figure 10. Compass readings affected by mobile phone signals

motion. We have taken a video clip in a busy crossroad with numerous local motion introduced by fast moving vehicles and slow moving pedestrians, at various distances to the camera. As shown in Figure 9, SaVE/*Comp* can still outperform the original algorithms but with reduced improvement (compared to Clip03-12 in Section 6.1). The improvement is further reduced in SaVE/*DAcc* because it partially relies on the motion vectors in the previous frame [2]. The reduction in improvement is expected because SaVE only provides extra information about the global motion.

7. DISCUSSIONS

While we show that SaVE can reduce video encoding complexity, we acknowledge that our work is limited in the following aspects due to limitations in the space, scope, and available hardware. Nevertheless, we believe our work has demonstrated the potential of the sensor-assisted multimedia processing and paved the way for further investigations.

7.1 Compass vs. Accelerometers

As apparent from Section 6, the digital compass-based SaVE implementation (SaVE/*Comp*) provides more improvement than the accelerometer-only implementation (SaVE/*DAcc*), in particular when horizontal camera rotation dominates. This is because the digital compass directly provides the absolute horizontal angle with much higher accuracy. In contrast, as shown in Equation 4, the accelerometer-only implementation calculates camera rotation during the current frame with the help from that during the previous frame, which leads to reduced accuracy.

The accelerometer-only implementation does enjoy a few practical advantages. First, accelerometers are cheaper and lower-power than digital compasses because of advancement in microelectromechanical system (MEMS) technologies. For example, the digital compass used in our experiments costs over $300 and consumes about 66mW while the board with two accelerometers used costs about $40 and consumes about 15mW. More importantly, accelerometers are more immune to external interference than digital compasses, which rely on the Earth's magnetic field. As Figure 10 shows, the reading of the compass used in this work can be affected when it is closely placed with a mobile phone in use. This problem will be addressed in our future work.

7.2 Limitations on Global Motion Estimation

Firstly, we have only considered rotational camera movement. Though the linear movement is less common in video clips taken by handheld devices, SaVE cannot handle it adequately at this moment. The reasons are multiple: 1) it is well-known that accelerometers are unreliable to detect linear movement, even over modest distances [28], and compasses cannot determine location change; 2) knowing the linear movement of the camera without knowing the distance between the object and the camera is inadequate for global motion estimation because the object movement in the frame will be proportional to the inverse distance. Secondly, we have not tackled another source of motion: zooming. However, we believe zooming data can be easily captured from the camera itself and be utilized in improving SaVE. Moreover, we have only used a simple translational model for global motion estimation, and this only shows the potential of SaVE. In our future work, we may attempt to use an affine or perspective model to estimate the global motion.

In spite of these limitations, we believe our SaVE scheme does provide obvious advantages. To our knowledge, the existing global motion estimation algorithms may require complicated

computations. The global motion used in SaVE, however, is obtained from sensors (through very simple calculations), which are already available on mobile devices. We have also shown that the power consumption of sensors of SaVE is ignorable. Indeed, we could consider SaVE as a "free lunch" for video capturing on mobile devices.

8. CONCLUSIONS

We reported the first attempt, SaVE, to utilize sensors to simplify motion estimation in H.264/AVC. We showed that a digital compass with a tri-axis accelerometer or two tri-axis accelerometers can accurately estimate global motion vector predictors. With SaVE, a fair PSNR performance can be maintained even with a much smaller search window for motion estimation, which leads to significantly reduced computation and therefore reduced hardware requirement and longer battery lifetime for handheld video recording devices.

SaVE improves global motion estimation, i.e. motion due to a moving camera. Although a stable camera is essential for capturing high-quality professional video, the increasingly popular handheld camcorders, intended for user-created, amateur video, *inherently* experience significant motion, including panning, tilting and zooming. Due to the explosive growth in video-enabled small devices such as video phones, the creation of video by amateur users is increasingly popular. The phenomenal success of portals such as YouTube has demonstrated such a social trend. It was one of the motivations to our research and also attests the practical significance and broader impact of our work.

An emerging trend in the mobile device industry is the integration of the multiple components and functionalities. Conventional wisdom would expect the computational complexity and thus the 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, angular rotation and vision in our case. We believe that there is plenty of *redundant computation* in a multi-component multimedia device such as a video phone, which could be removed by exploiting a similar methodology presented in this paper.

9. ACKNOWLEDGEMENT

The work by NUS team was supported by Singaporean MOE grant R-252-000-236-112. The work by Rice team was supported in part by NSF awards CNS/CSR-EHS 0720825 and IIS/HCC 0713249 and equipment donation from Texas Instruments. We thank Mr. Guangming Hong for helpful discussions. Moreover, we thank our anonymous reviewers and shepherd, Prof. Pascal Frossard, for their help in improving the final version of the paper.

10. REFERENCES

[1] M. Tourapis, "Fast ME in the JM reference software," *JVT Document JVT-P026, 16th Meeting*, pp. 24-29, 2005.

[2] G. Hong, Y. Wang, A. Rahmati *et al.*, "SenseCoding: Accelerometer-Assisted Motion Estimation for Efficient Video Encoding," *ACM Multimedia Conference*, Oct. 26-31, 2008.

[3] Z. Chen, J. Xu, Y. He *et al.*, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC," *Journal of Visual Communication and Image Representation,* vol. 17, no. 2, pp. 264-290, 2006.

[4] Z. Chen, P. Zhou, and Y. He, "Fast Motion Estimation for JVT," *JVT Document JVT-G016*, pp. 7-14, 2003.

[5] A. M. Tourapis, "Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation," *Proc. SPIE Conf. Visual Communications and Image Processing,* vol. 4671, pp. 1069-1079, Jan., 2002.

[6] T. Koga, K. Iinuma, A. Hirano *et al.*, "Motion compensated interframe coding for video conferencing," *Proc. IEEE National Telecommunication Conference*, pp. G5.3.1-5.3.5, Nov. 29 - Dec. 3, 1981.

[7] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 4, no. 4, pp. 438-442, 1994.

[8] L. M. Po, and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 6, no. 3, pp. 313-317, 1996.

[9] Z. Shan, and M. Kai-Kuang, "A new diamond search algorithm for fast block-matching motion estimation," *Image Processing, IEEE Transactions on,* vol. 9, no. 2, pp. 287-290, 2000.

[10] C. H. Cheung, and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 12, no. 12, pp. 1168-1177, 2002.

[11] C. W. Lam, L. M. Po, and C. H. Cheung, "A novel kite-cross-diamond search algorithm for fast block matching motion estimation," *Proc. Int. Symp. Circuits and Systems (ISCAS),* vol. 3, pp. 729-32, 2004.

[12] S. Eckart, and C. Fogg, "ISO/IEC MPEG-2 software video codec," *Proc. SPIE Conf. Visual Communications and Image Processing,* vol. 2419, pp. 100-118, 1995.

[13] B. Liu, and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 3, no. 2, pp. 148-157, 1993.

[14] C. K. Cheung, and L. M. Po, "Normalized partial distortion search algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 10, no. 3, pp. 417-422, 2000.

[15] C. H. Cheung, and L. M. Po, "Adjustable partial distortion search algorithm for fast block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 13, no. 1, pp. 100-110, 2003.

[16] L. W. Lee, J. F. Wang, J. Y. Lee *et al.*, "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 3, no. 1, pp. 85-87, 1993.

[17] K. K. Ma, and P. I. Hosur, "Performance Report of Motion Vector Field Adaptive Search Technique (MVFAST)," *ISO/IEC JTC1/SC29/WG11 MPEG99/m5851*, March, 2000.

[18] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive motion vector field adaptive search technique (PMVFAST) enhancing block-based motion estimation," *Proc. SPIE Conf. Visual Communication and Image Processing*, pp. 883-892, Jan., 2001.

[19] K.-y. Yoo, and J.-k. Kim, "A new fast local motion estimation algorithm using global motion," *Signal Processing,* vol. 68, no. 2, pp. 219-224, 1998.

[20] H. Jozawa, K. Kamikura, A. Sagata *et al.*, "Two-stage motion compensation using adaptive global MC and local affine MC," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 7, no. 1, pp. 75-85, 1997.

[21] D. Adolph, and R. Buschmann, "1.15 Mbit/s coding of video signals including global motion compensation," *Signal Processing: Image Communication,* vol. 3, no. 2-3, pp. 259-274, 1991.

[22] M. Ulrich, and S. Martin, *"Sensor Assited Video Compression"*, 2008 CA2605320 (A1)

[23] "Texas Instruments DM6446 H.264 media processor, http://focus.ti.com.cn/cn/lit/an/spraad6a/spraad6a.pdf."

[24] J. Heikkila, and O. Silven, "A four-step camera calibration procedure with implicit image correction," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1106-1112, 1997.

[25] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *Proc. IEEE Int. Conf. Computer Vision,* vol. 1, pp. 666-673, 1999.

[26] "Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/."
[27] "OS5000-S and OS5000 -US digital compass, product of OceanServer Technology, Inc., http://www.ocean-server.com/download/OS5000_Compass_Manual.pdf."
[28] "AN013: Position determination using Accelerometers, http://www.kionix.com/sensors/application-notes.html."