An Adaptive Motion Estimation Algorithm for Embedded Mobile Vehicle Surveillance Systems

Journal:	*IEEE Transactions on Vehicular Technology*
Manuscript ID:	VT-2008-00356
Suggested Category:	Regular Paper
Date Submitted by the Author:	23-Apr-2008
Complete List of Authors:	Wu, Bing-Fei; National Chiao Tung University, Department of Electrical and Control Engineering Peng, Hsin-Yuan; National Chiao Tung University, Department of Electrical and Control Engineering
Keywords:	Image coding, Image motion analysis, Communication systems

An Adaptive Motion Estimation Algorithm for Embedded Mobile Vehicle Surveillance Systems

Bing-Fei Wu, and Hsin-Yuan Peng*

Department of Electrical and Control Engineering, National Chiao Tung University,

1001, Ta Hsueh Road, Hsinchu, 300, Taiwan, R. O. C.

***Abstract*—This manuscript addresses the development of an adaptive motion estimation algorithm (AMEA) especially for the vehicle surveillance videos. In the real environments, the videos contain several problems, such as the different weathers, and the changing of the light sources, and these will lead to the heavy computation and the bad video quality. On one hand, in order to overcome the drawbacks, AMEA applies an easy histogram extension algorithm (HEA) to pre-process the images to keep the images stable and increase the estimation accuracy greatly. On the other hand, the purpose for the monitoring information is to provide clear faces of the intruders, so the images are separated into two parts, in and out of the region of interest (ROI). In ROI, AMEA can maintain the coding quality using the least numbers of manipulations, and meantime it can also enormously reduce the computational complexity for the blocks out of ROI. Moreover, AMEA is integrated into a low-power embedded mobile vehicle surveillance system (EMVSS), which can transfer the real-time videos to users' mobile phones through the 3.5G/3G network, so that the users can know the situations of the vehicles anytime and anywhere. EMVSS has been successfully tested in the real road environment of Taiwan.**

***Index Terms*—H.264, motion estimation, embedded system, surveillance video.**

* Corresponding author: Hsin-Yuan Peng, sypeng.ece91g@nctu.edu.tw, +886-3-5712121#54428

I. Introduction

The most complex part of popular video compression standards, including MPEG-1/2/4 [1]-[3] and H.261/3/4 [4]-[6], is motion estimation (ME). The goal of ME is to remove the temporal redundancies existing in adjacent frames, and the block-matching algorithm is used as a method for most of the video coding systems. It is used to find a block which is most similar to a current block within a pre-defined search area in a reference frame, and it dominates the encoded image quality, the compression ratio, and the computation time. The most straight forward method, known as the full search block matching algorithm (FSBMA) of obtaining motion vectors (MVs), is to search all possible locations within a given area. Since FSBMA uses an exhaustive search to locate the minimum block-distortion measure (BDM) for each candidate block, it provide optimal performance but at the expense of very high computation. Usually, FSBMA spends about 70% of the total encoding time and this is the reason why FSBMA is not used in real-time systems. Indeed, ME is the major bottleneck in video coding applications, hence the need for faster algorithms.

To reduce the number of search steps of FSBMA in order to increase the overall speed is essential. The fast FSBMA, including the 2-D logarithmic search (2DLOG) [7], the three-step search (TSS) [8], the new three-step search (NTSS) [9], the advanced center biased search [10], the four-step search (4SS) [11], the cross-search [12], the prediction search [13], the successive elimination algorithm (SEA) [14]-[16], partial distortion elimination (PDE) [17], the winner-update algorithm [18], the advanced diamond search algorithm (DSA) [19], and the hexagon-based search (HEXBS) [20], [21], are proposed to reduce the computational heavy load of FSBMA while maintaining its quality. DSA has achieved a significant speed gain by considering diamond-shaped search patterns instead of the conventional square ones with a view to approximate the optimal (but unrealizable) circular shape as closely as possible. Recently, HEXBS algorithm has surpassed the speed of DSA by using hexagon-shaped search patterns. Moreover, in order to refine the accuracy of DSA, several new algorithms, such as motion vector field adaptive search technique (MVFAST) [22], predictive MVFAST (PMVFAST) [23], and enhanced predictive zonal search (EPZS) are proposed

[24]. MVFAST improve DSA in both terms of visual quality and speed up by initially considering a small set of predictors. Unlike DSA where only a large moving diamond pattern was considered, MVFAST also introduced a smaller moving diamond. PMVFAST uses basically the same architecture and patterns as MVFAST does, but a significant difference of PMVFAST compared to MVFAST is the way the small versus the large diamond is selected. Dissimilar to MVFAST where motion was characterized as low, medium, or high by considering the largest motion vector candidate, in PMVFAST a different selection strategy, which can improve the overall speed of the algorithm by using the large diamond less often, is used. Furthermore, EPZS that improves upon PMVFAST by considering several other additional predictors in the generalized predictor selection phase of PMVFAST. EPZS also selects a more robust and efficient adaptive threshold calculation where as, due to the high efficiency of prediction stage, the pattern of the search can be considerably simplified. In addition, an efficient hierarchical motion estimation algorithm adopting the averaging filter to downsample the image in order to reduce the computational complexity greatly while maintaining the good coding quality are also proposed, but it is a hardware architecture, which can not gain the same performance using the software implementation [25].

All of these fast algorithms [7]-[25] assume that either the error surface is unimodal over the entire search area (i.e., there is only one global minimum) or MV is center-biased. These hypotheses essentially require that either BDM increases monotonically as the search point moves away from the global minimum position or MV exists in a small range. However, they are generally invalid for many real video sequences because the highly nonstationary characteristics of the signals. Moreover, despite their respective differences, these fast search algorithms all have one common feature that none of them has been designed to provide flexibility in controlling the performance in terms of predicted picture quality and processing time. Golam et al. propose a fully adaptive distance-dependent thresholding search (FADTS) [26], which can adjust the required threshold control parameter via an adaptive process which uses the information from previous frames to achieve specified image quality and manipulation speed. However, it needs to be modified when dealing with the vehicle surveillance videos.

As illustrated in Fig. 1, the images of the vehicle surveillance videos have some key features. First, the important information, like the faces of the drivers and the judgments of the existence of the intruders, usually locates in a specific area, which is the region of interest (ROI), and other parts of the image, out of ROI, often do not contain significant data. The coding results in ROI cam be as clear as possible in order to provide evidences for the police department, and the complexity also should be kept as low as it can. Second, since the vehicles are frequently in the moving state, and the surrounding environments keep changing, the parameters of the images, such as brightness and the light source, are varied case by case. The gray level of the pixels will be influenced and the accuracy of the MVs will be forced to be lower due to the undesired environment effects. Third, the postures of the motormen do not move wildly when they are driving. Therefore, the ME framework for the specific application should be modified to increase the performance in terms of the coding speed and the image quality. Moreover, when the recorders in cars encode the monitoring videos with high quality, the users should be able to browse them as convenient as possible. With the popularity of mobile communication in recent years, people always have a mobile phone or a PDA. When the systems can transmit the vehicle security information to the owners' mobile devices, more properties can be protected by real-time alarm messages. Moreover, if the guard system of the vehicle can capture the images which are in or out of the vehicle anywhere, the vehicle owners can handle the situation of the vehicle more precisely. A preliminary version of this kind of system has been presented in [27].

The main contributions of this paper are to develop an adaptive ME algorithm (AMEA) especially for the surveillance videos in vehicles, and to realize it with an embedded mobile vehicle surveillance system (EMVSS), which has been successfully implemented and tested with the real cars and roads. In ROI, AMEA can do its best to maintain the encoding quality, using the least number of operations, and it can adaptively change the search procedures to overcome the big motions and the changes of the environments. Therefore, a histogram extension algorithm (HEA) is adopted to pre-process the blocks in order to overcome the lighting effects. Furthermore, since the MVs of these out-of-ROI blocks are usually not large, and the importances of them are not high, AMEA can manipulate them as fast as possible by optimizing the search procedure according to the MV characteristic. The experimental results show that AMEA can even provide better quality than FSBMA does when the video is seriously influenced by the environmental change. Moreover, in the system level design, EMVSS has greatly enhanced the functionalities of those in [27]. The 3.5G/3G network can provide much higher bandwidth than it of GPRS, and H.264 video encoding has better compression ratio than JPEG. Hence, EMVSS can offer a real-time videos inside the vehicles to the users at anytime and anywhere.

The rest of this paper is organized as follows. Section II analyzes the real processing problems for the vehicle surveillance videos. Section III presents the AMEA architecture, and Section IV illustrates the integration of the AMEA with the EMVSS. Section V depicts the results of implementation and Section VI draws conclusions.

II. The Analyses of the vehicle surveillance videos

The special features of the vehicle surveillance videos will be addressed in this Section. One is the lighting effects, and the other is the characteristics of MVs in and out of ROI. As shown in Fig. 1, the camera is installed in the front-right corner of the vehicle since not only the face of the driver but also the scene outside the window, which can help the police to recognize the exact location, is important. The detail analyses are described as below.

2.1. Lighting effects

Lighting conditions are one of the most important problems to be solved in real tests. As shown in Fig. 2, the vehicle is passing through a long tunnel, which has rectangular windows on one side, so the sunlight will influence the brightness of the images periodically. When passing the middle of the window, outer lights directly illuminate the driver, increasing the pixel levels quickly. Once the car has left, the pixel values decline immediately. Fig. 3(a) and Fig. 3(b) illustrate the brightest and the darkest images, respectively, and the time difference between them is only 360 milliseconds. In these two pictures, it can be observed that the driver's posture only changes a little, but the pixel values of them are quite different. Another method to exam the effects is to compare the mean, $\overline{M}_k$, defined in (1), of the k^{th} frame in the whole video,

$$\overline{M}_k = \frac{1}{W \times H} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} I_k(i, j), \tag{1}$$

where W, H are the width and the height of the image $I_k(\cdot)$, and $I_k(i, j)$ represents the value at position (i, j) of the k^{th} frame, respectively. The mean of the each frame in this video (V1) is depicted in Fig. 4, and the lighting effects can be easily observed. Fig. 5(a) and Fig. 5(b) also shows the most luminous and the obscurest frames of another video (V2), respectively, and the position of the vehicle is closer to the windows than it of V1. Fig. 6 depicts the average value of all pixels of each frame in V2, and it is higher than it of V1 due to the location of the automobile. The lighting effect appears to be smaller than it of V1, but the trend of the diagram is almost the same. The serious problem exists and needs to be solved.

2.2. The coding performance degradation due to the lighting effects

As described in Section I, when using the block-matching algorithm, MVs are obtained by locating the minimum BDM, which is usually measured by the sum of absolute difference (SAD), for candidate blocks. However, the lighting effect will cause the pixel values to change greatly even though the motion of driver is small. Thus, the MVs manipulated by the minimum SAD will be wrong and the quality will be decreased. Fig. 7 and Fig. 8 show the correlations between the adjacent frames of V1 and V2, and the results are highly related to Fig.4 and Fig. 6, respectively. The quality will be increased when higher correlation exists between the successive images. The correlation coefficient, ρ, is defined as

$$\rho = \frac{\sum_i \sum_j \left[\left(I_k(i, j) - \overline{M}_k\right) \times \left(I_{k+1}(i, j) - \overline{M}_{k+1}\right) \right]}{\sqrt{\sum_i \sum_j \left(I_k(i, j) - \overline{M}_k\right)^2} \times \sqrt{\sum_i \sum_j \left(I_{k+1}(i, j) - \overline{M}_{k+1}\right)^2}}, \tag{2}$$

where $\overline{M}_k$ and $\overline{M}_{k+1}$ are the means of the image pairs, $I_k(\cdot)$ and $I_{k+1}(\cdot)$, and $I_k(i, j)$ represents the value at position (i, j) of the k^{th} frame, respectively.

If the low correlation exists in a neighboring frame pair, it means that the distortion between them is large, and will lead to poor quality and compression ration. The peak signal-to-noise ratio (PSNR) is used for the measurement of performance, and it is defined as

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{W\times H}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\left[I_k(i,j)-\hat{I}_k(i,j)\right]^2}, \tag{3}$$

where W, H are the width and the height of the image $I_k(\cdot)$, and $\hat{I}_k(\cdot)$ is the k^{th} motion compensated image. Table I illustrates the PSNR of these videos, where V0 is a surveillance video when driving on a normal road without the lighting effects. The ME method is FSBMA, the block size is 16×16 and the search area is [-16, +15]. These videos, V0, V1, and V2, are all in CIF format, and a total of 450 frames. Fig. 9(a) and Fig. 9(b) are the mean and the correlation of V0, respectively. The diagram shows that a normal motorist has quite small motion during driving and PSNR is much higher than those of V1 and V2. However, V0, V1, and V2 are filmed at the same angle, with the same driver, in the same vehicle, and on the same day with only five minutes time interval each, and the behavior of the same person will not change enormously in a short time. Therefore, from Fig. 4 to Fig. 9 and Table I, the lighting effect indeed results the bad estimation performance, and this problem needs to be solved.

2.3. *The MV characteristics*

In Section 2.2, the hypothesis of the drivers' behavior is made, and some statistical results will support the assumption. First of all, because the camera of the vehicle surveillance system is firmly fixed in the car, and the capture area will be also bounded, ROI can be pre-defined. ROI should cover almost the moving region of the driver's head to recognize the intruder's face, so the ROI like Fig. 1 is set artificially. Fig. 10 shows the manipulated MVs, which is drawn in the shape of arrows, on the reference frame and the area besieged by the green dotted line is ROI. Fig. 11(a) and Fig. 11(b) depict the probability of the magnitude of MVs in and out of ROI for all frames of V0, respectively. It can be observed that there exists high chance of short MVs when its corresponding block is out of ROI, so the ME algorithm for these blocks can be modified based on the results. On the other hand, the drivers' heads will move to all directions in order to check different angles, and their faces will turn left and right to see the side view. Therefore, BDM in ROI will be larger than it out of ROI, and hence the MVs in ROI will have high probability to be longer. To maintain the coding quality of the blocks in ROI, which will have much more situations than those out of ROI, is essential since the main goal of the surveillance videos is to provide clear evidences.

III. Adaptive Motion Estimation Algorithm (AMEA)

AMEA can be divided into three parts. One is the histogram extension algorithm (HEA) for pre-processing the image, one is the adaptive ME framework for the blocks in ROI to uphold the image quality, and the other is the simplified MV search procedure for those out of ROI. The complete algorithm is described as below.

3.1. Histogram extension algorithm (HEA)

The lighting effect is a serious problem for the image processing for vehicular technology. Bergasa et al. propose a real-time system for monitoring driver vigilance [28], and they are facing the same issue. In order to minimize the interference from light sources beyond the IR light emitted by the LEDs in [28], a narrow bandpass filter centered at the LED wavelength has been attached between the CCD and the lens. However, the solution is not suitable for the video encoding system since the types of input devices are quite different. For minimizing the BDM due to the numerous reasons except from the motion, the brightness preserving algorithm should be adopted. Several methods, including histogram equalization [29], Brightness preserving bi-histogram equalization [30], equal area dualistic sub-image histogram equalization [31], minimum mean brightness error bi-histogram equalization [32], and brightness preserving histogram equalization with maximum entropy [33] are proposed to reduce the artifacts. These frameworks are very powerful to overcome the different light sources at any angle, and are quite complex. Since EMVSS is put in the real vehicle and works whenever the car is started, it must be a low-power embedded system. Therefore, a simple HEA is derived to fit the request and release more time for CPU to processing the H.264 video compression.

HEA pre-processing is addressed to adjust the images which are under different luminance to the ones with a similar luminance condition. A color image can be separated into three dimensions, Y, U, and V, and ME only calculates the MVs for Y plane. Therefore, only the histogram of the Y component needs to extended, and the linear normalization with mean shift (LNMS) is applied to modify the images of different conditions to a similar one.

As illustrated in Fig. 12(a) and Fig. 12(b), x_i and x_i' are the intensity of a pixel value of the gray-level image, $I_k(\cdot)$, before and after the HEA manipulation, where *i* is from 0 to 255, s_i is the total amounts of the x_i th level and *a*, *b*, *c*, *d*, *e*, and *f* are random numbers with the relationship of $a<b<c<d<e<f$, respectively. Fig. 12(a) and Fig. 12(b) are the histograms of the input frame before and after the pre-processing. Suppose the mean of Fig. 12(a), $\overline{M}_k$, which is calculated by (1), is located on $x_{\overline{M}_k}$ with $s_{\overline{M}_k}$ pixels, and HEA will shift the value of all these $s_{\overline{M}_k}$ points to 128. In another word, $x_{\overline{M}_k}$ is moved to x_{128}, and the remaining pixels will be linearly shifted to the two sides of x_{128}. Two transform parameters, α and β, are used to move the x_i to x_i', and they are defined as

$$\alpha=\frac{128}{x_{\overline{M}_k}}, \tag{4}$$

and

$$\beta=\frac{128}{255-x_{\overline{M}_k}}. \tag{5}$$

The transfer function from x_i to x_i' is defined as

$$\begin{cases} x_i' = \alpha \times x_i , & 0 \le i < \overline{M}_k \\ x_i' = \beta \times x_i , & \overline{M}_k < i \le 255 . \\ x_i' = 128 , & i = \overline{M}_k \end{cases} \tag{6}$$

By (6), $\overline{M}_k$ will be shifted to around 128, x_i, $0 \le i < \overline{M}_k$, will be modified as x_i', $0 \le i < 128$, and x_i, $\overline{M}_k < i \le 255$, will be substituted as x_i', $128 < i \le 255$. Therefore, the average value of each input image, which can be influence by the light source, will be around 128, and the lighting effect problem can be solved. Moreover, the MV search procedure can avoid the wrong minimum BDM, and can reduce the search points to increase the speed.

3.2. AMEA in ROI

In performance-management ME, given a target prediction image quality in terms of average SAD per pixel, the motion search algorithm tries to achieve it using as few search checking points as possible. Performance-management ME also assumes real time constraint, which allows very limited number of passes per macroblock (MB). Without such a constraint, trivial trial and error technique with a very high number of passes would suffice the adaptation. Without any loss of generality, this paper assumes the strictest constraint where only one ME pass is performed per MB.

A. AMEA framework

The concept of AMEA is not linked to any specific search pattern shape. In the wake of improved speed gain by nonsquare search patterns, AMEA has been implemented using search diamonds, SD_τ, as shown in Fig. 13 where the number of checking points in SD_τ is

$$\begin{cases} 1 , & \tau = 0 \\ 4\tau , & \tau = 1, 2, ..., 2d \end{cases}, \tag{7}$$

where the search range of AMEA is set as [-d, d-1] and SD_τ represents the MV of length in the range of $\left\lfloor \tau/\sqrt{2}, \tau \right\rfloor$. Some of the checking points in the search diamond fall outside the search windows that are obviously ignored.

Like all block-base ME search techniques, AMEA starts at the center of the search space. The search then progresses outwards by using SD_τ in order while monitoring the current minimum SAD. A parametric thresholding function, *r*, defined as

$$r(\tau, Th_m) = Th_m \times \tau\,, \tag{8}$$

is used to determine the various thresholds to be used in the search involving each SD_τ where Th_m is an adaptive index for *r* of the m^{th} frame (the adaptation of Th_m and the definition of *m* will be addressed in the next paragraph), and it is set at the start of each search and acts as a control parameter. After searching each SD_τ, the current minimum SAD is compared against the threshold value, $r(\tau, Th_m)$, of that specific search procedure in SD_τ and the search is terminated if this SAD value is not higher than that threshold value.

B. AMEA closed-loop adaptation model

AMEA works sequentially on the frames of an input video sequence. Although the consecutive frames are considered to be highly correlated, the input video signal can be considered time variable or nonstationary from the adaptation point of view. Therefore, a closed-loop adaptation model, as shown in Fig. 14, is presented for AMEA, and it has the following four modules.

- Adaptation of Th_m: Since the adjacent frames have high correlations, the number of updating the Th_m can be reduced. In other words, the Th_m will maintain the same value during *L* frames, and the iteration index *m* are changing from 1, 1+*L* to $(\lfloor (N-1)/L \rfloor)L$ for a video sequence with *N* frames. The value of Th_{m+1} for the next iteration are defined as

$$Th_{m+1} = Th_m + f\left(e^{[m]}, y^{[m]}\right), \tag{9}$$

where $e^{[m]}$ and $y^{[m]}$ are the average value of each e_k and y_k, which are the error signal and the average SAD output of the k^{th} frame, in a total of L frames, respectively.

- HEA: This module will manipulate the histogram of the input image, and then shift the histogram using HEA, which is described in Section 3.1. The results of HEA will directly replace the original frame to reduce the memory usage. Two blocks of the memory, which are acted as a ping-pong buffer, Mem$_1$, and Mem$_2$, are allocated for AMEA. First, in the intra frame mode, the input images are always stored in Mem$_1$, and the reconstructed frame will be saved in Mem$_2$. Second, in the inter frame mode, since the current frame will be the reference frame in the next encoding loop, these two parts will switching their status mutually until the next intra frame mode, and the scheme is illustrated in Fig. 15.
- MV search procedure: This module calculates MVs using the AMEA framework, which is depicted in Section 3.2.A. The inputs of the module are the video frame pair $\left(I_k(\cdot), I_{k+1}(\cdot)\right)$ and the control parameter Th_m. The output of the model y_k is the predicted image quality in terms of average SAD and it is a monotonically increasing function.
- Performance calculation: The performance of the adaptive system are estimated by calculating the error signal, e_k, as

$$e_k = T_{out} - y_k, \tag{10}$$

where T_{out} is the target output, defined by the users. The value of e_k must be minimized as the adaptation process progresses.

The performance of an adaptive system largely depends on how the function $f\left(e^{[m]}, y^{[m]}\right)$ is defined. A few gradient search algorithms [34]–[38] exist that can adapt a system in searching for the optimal parameter to minimize error signal in (10). Among these, the least mean square (LMS) is the most well-known and popular method for its computational simplicity, robustness, and relatively easy implementation for online estimation of time-varying system parameters. A number of variants on the LMS theme have been conceived in order to ratify potential problems of the original LMS algorithm such as the need to guess the best value of step size, slow convergence, and numerical instability. The normalized block LMS (NBLMS) [38] is considered as the best option for automatically adjusting the control parameter *Th* in order to achieve a target average SAD while coding a video sequence, where this sequence can be considered as a time varying nonstationary input to the adaptation system. Based on NBLMS, the threshold control parameter is updated as

$$Th_{m+1} = Th_m + ue^{[m]} \frac{\frac{1}{L}\sum_{i=0}^{L-1} y^{[m+i]}}{E_y}, \tag{11}$$

where *u* is the step size and $E_y = \sum_{i=0}^{L-1}\left(y^{[m+i]}\right)^2$.

C. The initializations of the system parameters, Th_0, L, and u

In FADTS, the initialization of Th_0 is chosen by estimating the first frame pair using a normal method, like FSBMA, DSA, or etc., and performs the MV search procedure again after Th_0 is decided. The flow is achievable when the powerful PC is used to develop the algorithm when the processing framerate is higher than 60 frame per second. Thus, Th_0 is appropriate for the video sequence since the frame pair for estimation is still the same. However, when the low computing power embedded system is chosen in order to save the consuming energy, Th_0 will be calculated after a period of time, and it may not suitable for the current frame pair. Therefore, Th_0 is directly set to 0, and it will be updated to increase the speed after the first *L* frames. The parameter *L* for AMEA is defined as 4 since the most serious lighting effects usually take 15 to 20 frames to change from the darkest image to the brightest one. Although lower *L* also provided similar performance in satisfying the targets, according to (9), it increases the overhead computational cost for the adaptation process. Conversely, higher *L* can be considered in order to reduce the overhead cost, though the block length of *L* in the NBLMS algorithm cannot be too high if it is assumed that the content of a video sequence may be unstable.

With regard to the parameter *u*, Meghriche et al. [39] highlighted that there is no universal solution for finding the optimal value of *u*. In [38], the NLMS algorithm considers a step size range of ($0<u<2$) for signal processing applications. The lower the value of *u*, the slower the convergence rate while a high step size can lead to system instability. The value of *u*=2 was defined for all the various standard video sequences tested, with no instability encountered for AMEA.

3.3. AMEA out of ROI

As mentioned in Section 2.3, the probability of the MVs whose lengths are less than 2 is much higher than the MVs with other magnitudes, so the ME algorithm can be greatly modified based on these results. Moreover, since the instruction memory of EMVSS is quite small and the computing power of the embedded processor is low, the software optimization for reducing the code size and increasing the performance is necessary. Therefore, the programs for manipulating the MVs out of ROI should re-use the functions in Section 3.2 well to achieve the needs. In the whole searching procedure, the most frequently used part is the diamond-shaped SAD comparator, hence it is written by assembly codes efficiently. No matter the incoming block is in or out of ROI, the high speed calculations are performed according to the system parameters. The main differences between AMEA in and out of ROI are the stop conditions and the search ranges. Because the importance of the blocks out of ROI is not as high as those in ROI, the maintenance of the estimation quality is no longer essential. Thus, the adaptations of the encoding parameters can be removed and the stop condition can be the same as DSA to decrease the complexity. Furthermore, based on the analysis results in Section 2.3, the search range of these out-of-ROI blocks can be shrunk to [-2, +2]. With these prior techniques, the computational overhead can be much lower.

3.4. The complexity analysis of AMEA

Consider an ME system with the following parameters: frame size = $W \cdot H$ pixels, MB size = $M \cdot M$, and the search range is [-*d*, *d*-1]. The overall search procedure includes the HEA, and the different methods for the blocks in and out of ROI. Therefore, the complexity of AMEA can be defined as

$$C_{AMEA} = C_{HEA} + MB_{in} \times C_{in} + MB_{out} \times C_{out}, \tag{12}$$

where C_{HEA}, C_{in}, and C_{out} represent the complexity of HEA, the estimation algorithm for the in-ROI and the out-of-ROI blocks, and MB_{in} and MB_{out} are the number of the blocks in and out of ROI, respectively. C_{HEA} contains the operations for calculating $Mean_k$, which requires $W \cdot H$ additions and one division, and shift each input pixel to a new value due to the corresponding formulas with one multiplication and one division. Therefore, C_{HEA} can be defined as

$$C_{HEA} = (W \cdot H + 1) + (W \cdot H) + (W \cdot H) = 3(W \cdot H) + 1. \tag{13}$$

If there are Γ numbers of operations required for the SAD calculation of one search checking point, then FSBMA needs $\Gamma(2d)^2$ manipulations per MB using integer-pel accuracy. AMEA requires extra *d* operations to compare the current minimum SAD with the predefined threshold. Therefore, the maximum computational bound of C_{in} is

$$C_{in}^{\max} = \Gamma(2d)^2 + d \tag{14}$$

operations per MB. Conversely, by using a very high threshold value, when only the corresponding center of the search space is checked and only one calculation is required to compare SAD. Hence, the minimum bound of C_{in} is

$$C_{in}^{\min} = \Gamma + 1 \tag{15}$$

operations per MB. As for the out-of-ROI MBs, the search range is reduced to 2, and the algorithm is the same as DSA. Thus, the upper and the lower bound of C_{out} are defined as

$$\begin{cases} C_{out}^{\max} = 16\Gamma \\ C_{out}^{\min} = \Gamma \end{cases}. \tag{16}$$

For example, consider AMEA for a typical CIF video sequence with $W = 352$, $H = 288$, $M = 16$, $d = 16$ and $\Gamma = 3M^2 = 768$. The pre-defined ROIs of V0, V1, and V2 are all set to 108 blocks, and the number of the blocks, which are out of ROI, is 288. Therefore, the largest and the smallest numbers of calculations for

AMEA per frame are $C_{AMEA}^{\max} = 88.78$ million and $C_{AMEA}^{\min} = 0.61$ million, respectively while $C_{FSBMA} = 311.43$ million. After testing several vehicle surveillance videos, C_{AMEA} is usually around $0.1 \times C_{FSBMA}$ since the probability of the short MVs is much higher than those of the long ones.

IV. System Level Design for An Embedded Mobile Vehicle Surveillance System (EMVSS)

Since EMVSS should work whether the vehicle is started or not, it has to be as low power consumption as it can. Therefore, a dual-core embedded platform, TI OMAP5912, is chosen, and the algorithms for EMVSS are all modified to be much simpler than their original versions in order to save the energy. EMVSS can be integrated with the signal triggers, such as open door, start engine, and etc., of the existing burglarproof systems, and it will automatically enter to the suspend mode when nothing happened for a period of time. The hardware and software architectures are shown in Fig. 16, and Fig. 17, respectively, and all programs, including the embedded Linux operating system, and the peripheral controllers, except for the H.264 encoder are executed by ARM.

4.1. Corporation of ARM and DSP

Figure 18 describes the efficient cross work between the ARM and the DSP cores, operating in parallel when the H.264 encoding function is enabled. At first, when the function is called, ARM initializes the system, reads frames in the storage devices and then writes into the external memory of DSP. When the source is ready, ARM activates the H.264 encoding routine of DSP. Then, the next frame will input into the buffer whenever the side information of fame is processed. After encoding of the current proceeding frame, the routine restores the necessary information and transits it to the next frame. In this configuration, ARM performs a part of the H.264 encoding process in addition to system control. It should be noted that ARM generally has the better architecture for the frame buffering than DSP.

Figure 19 shows the synchronization of audio and video encoding of both CPU cores in OMAP5912. Three threads, video, packaging, and streaming thread (VT, PT, and ST), are created at the ARM side. VT fetches the video data from the USB camera and sends to DSP for video encoding. Once the data are acquired, a timestamp is recorded for synchronization. The communication between ARM and DSP is solved by DSPGateway[40]. After DSP completes encoding a video frame, it sends a notifying signal through the mailbox to ARM, and VT starts receiving the bitstream and the related information from DSP. After the receiving process is done, VT also sends the new information and a starting signal through mailbox to DSP immediately to ensure that there is a minimum latency between ARM and DSP. PT takes charge of packaging the video bitstream into H.264 file format, and adds the necessary header to make it compatible with RTP. ST takes charge of gathering necessary information for the streaming server. Furthermore, it transfers the data in UDP for remote clients to receive them via 3.5G/3G network. Figure 20 shows that when switching the threads, PT and ST acquire timestamps and bitstream created from VT. If the data are dropped due to the heavy network traffic, the frames which fall behind are dropped in order to keep in phase and they will not re-send again according to UDP.

4.2. The implementation of the mobile clients

Since more and more mobile phones support Java programming, the J2ME Java version is addressed for developing applications on mobile devices. But the API of J2ME is not as complete as that of J2SE. There are still many restrictions in developing the Java program. The socket programming is used to establish the connection between EMVSS and the mobile phone. The server-push method is applied in the data transmission between them. That is, the mobile phone will not download the files until the server is ready for the downloading. Figure 21 shows the state chart of this method and Fig. 22 is the processing flowchart of this program on a mobile phone.

Users can download the continuous real-time images to their mobile phones from EMVSS anywhere if this Java applet has been embedded in their mobile phones. However, the surveillance videos are very private information, so the authentications are required. The computing power and memory size of the mobile phone are both much lower than other peripherals so that software decoding of the received H.264 videos relies on H.264 hardware decoder on mobile phones. Therefore, a cellular phone with H.264 de-compressor, such as Nokia N95, is necessary.

V. Implementation Results

AMEA has been successfully developed and integrated into EMVSS, which has installed and tested in a vehicle and the real road environments. The implementation results are described as follows.

5.1. The estimation quality comparisons of AMEA and other ME algorithms

The test video sequences: “V0,” “V1,” and “V2” are used to evaluate the performance of AMEA. The degrees of the influences of the lighting sources to these three videos are “almost zero”, “very serious”, and “middle”, respectively. All the sequences consist of 450 frames; the frame rate is 30 fps, and the image size is CIF. The search range is defined as $[-d, d-1]$ where d=16. The performance of AMEA is compared to that of four well-known algorithms: FSBMA, NTSS, DSA, and HEXBS. In addition to these non adaptive algorithms, FADTS is also implemented by the standard C language on a normal PC with Pentium 4 processor at 3.0GHz and a total of 1G byte DDR400 SDRAM. The input sequences are all the same, which can avoid the distinct quality due to the different fragment in the same test patterns, and the target output PSNR of FADTS and AMEA is set to 30 dB.

Tables II and III present the results. Table II describes the estimation quality of these six algorithms in terms of PSNR, and Table III shows the comparisons of the normalized processing speed. In order to truly reflect the MV accuracy, the estimation results are made by the MB and the corresponding MV without the inflation of the error residuals. In Table II, AMEA can maintain the pre-defined PSNR by the adaptive threshold, and it even outperforms the FSBMA when the input video sequences have lighting effects. The HEA adopted by AMEA can reduce the influences from the light sources, and it can provide higher correlations between the adjacent frames, as shown in Fig. 23, and Fig. 24. Therefore, the performance of AMEA in V1 and V2 is the highest among these frameworks. Since AMEA adopts the simple DSA with small search range for the blocks out of ROI, the quality of V0 is worse than other algorithms. However, the difference is negligible. Furthermore, the processing speed of AMEA is as fast as HEXBS and DSA while providing better quality. It can be observed that FADTS is almost as slow as FSBMA if the target output PSNR is unachievable. From these experimental results, AMEA can solve the major problems of the vehicle surveillance videos, and it is suitable for the embedded platform due to its lower complexity.

5.2. *The performance and functionalities of EMVSS*

With the tight cooperation between ARM and DSP, the system parameters are shown in Table IV. If the images are always ready for processing, the H.264 encoder with the efficient AMEA can calculate 7 FPS. However, after adding the system overhead, such as the dual-core intercommunication, the frame grabbing latency, the memory collision hazard, the 3.5G/3G internet transmission delay, and etc., the actual frame rate of the surveillance video, which is decoded and shown on the screen of the mobile phone, is around 4 FPS, which is enough for the checking the status of the vehicles and recognizing the intruders. Moreover, EMVSS will be activated only when the triggers from the sensors and the users, or it will stay in the suspend mode to greatly save the battery energy when the car is not moving and the fuel-to-electricity procedure is disable. The power consumption of EMVSS is quite small no matter if it is enable or not, and EMVSS can wake up from the suspend mode in a very short time to provide the users real-time notifications. Therefore, EMVSS is very suitable for developing the vehicle applications, and the GUI developed on mobile phones is shown in Fig. 25.

VI. Conclusions

This paper has overcame the video encoding problems, such as lighting effects, outside environmental influences, and the quality issues of the in-ROI part, in the vehicle surveillance videos. Moreover, the design and implementation of AMEA, which adopts an efficient HEA to pre-process the images to reduce the luminance changing due to the different environments, are proposed. An adaptive and low complexity MV search procedure in and out of ROI is developed to increase the calculation speed and the encoding quality, and be integrated into EMVSS, which is suitable for telematics because of its low power consumption and stability. EMVSS can capture the real-time images inside the car, compress it with H.264 standard, and transfer it to the mobile phones. The programs with friendly GUI are completed to make the users to browse the instant videos through 3.5G/3G network to determine whether their properties are stolen or not anytime and anywhere.

Acknowledgment

This work was supported by National Science Council under Grand no. NSC 96-2752-E-009-012-PAE.

References

[1] *Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to About 1.5Mbit/s*, ISO/IEC 11182. MPEG-1, 1993.

[2] *Information Technology: Generic Coding of Moving Pictures and Associated Audio Information: Video*, ISO/IEC 13818. MPEG-2, 1995.

[3] *Information Technology – Coding of Audio-Visual Objects – Part 2: Visual*, ISO/IEC 14496-2. MPEG-4, 2001.

[4] *Video Codec for Audiovisual Services at p*64 kbit/s*, ITU-T Rec. H.261, 1993.

[5] *Video Coding for Low Bitrate Communication*, ITU-T Rec. H.263, 2000.

[6] *Joint Video Team (JVT) of ISO MPEG and ITU-T VCEG, JVT-G050*, ITU-T Rec. H.264/ISO/ICE 14496-10 AVC, 2003.

[7] J. R. Jain and A. K. Jain, "Displacement measurement and its application in inter frame image coding," *IEEE Trams. Commun.*, vol. COM-29, no. 12, pp. 1799-1808, Dec. 1981.

[8] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated inter frame coding for videoconferencing," in *Proc. Nat. Telecomm. Conf.*, New Orleans, LA, pp. G5. 3.1-G5. 3.5, Dec. 1981.

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

[10]H. Nisar and T.-S Choi, " An advanced center biased search algorithm for motion estimation," in *Proc. IEEE Int. Conf. Image Process.*, 2000, vol. 1, pp. 832-835.

[11]L. M. Po and W. C. Ma, "Novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313-317, Jun. 1996.

[12]M. Ghanbari, "The cross-search algorithm for motion estimation (image coding)," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 950-953, Jul. 1990.

[13]L. Luo, C. Zou, X. Gao, and H. Zhenya, "A new prediction search algorithm for block motion estimation in video coding," *IEEE Trans. Consumer Electron.*, vol. 42, no. 1, pp. 56-61, Feb. 1997.

[14]W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing*, vol. 4, pp. 231-236, June 1995.

[15]X. Q. Gao, C. J. Duanmu, and C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Processing*, vol. 9, pp. 501-504, Mar. 2000.

[16]M. Brunig and W. Niehsen, "Fast full-search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 241-247, Feb. 2001.

[17]Digital Video Coding Group, *ITU-T Recommendation H.263 Software Implementation*, Telenor R&D,

1995.

[18]Y. S. Chen, Y. P. Huang, and C. S. Fuh, “Fast block matching algorithm based on the winner-update strategy,” *IEEE Trans. Image Processing*, vol. 10, pp. 1212-1222, Aug. 2001.

[19]S. Zhu and K. K. Ma, “A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation,” *IEEE Trans. Image Processing*, vol. 9, no.2, pp. 287-290, Feb. 2000.

[20]C. Zhu, K.-P. Chau, and X. Lin, “Hexagon-based search pattern for fast block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 349-355, May 2002.

[21]C. Zhu, X. Lin, L. Chau, and L.-M. Po, “Enhanced hexagonal search for fast block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1210-1214, Oct. 2004

[22]P. I. Hosur and K. K. Ma, “Motion Vector Field Adaptive Fast Motion Estimation,” *Proc. of Second International Conference on Information, Communications and Signal Processing*, Singapore, Dec. 1999.

[23]A. M. Tourapis, O. C. Au, and M. L. Liou, “ Predictive Motion Vector Field Adaptive earch Technique (PMVFAST) – Enhancing Block Based Motion Estimation, ” *Proc. of Visual Communications and Image Processing*, San Jose, CA, Jan. 2001.

[24]A. M. Tourapis, “Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation,” *Proc. of Visual Communications and Image Processing*, San Jose, CA, Jan. 2002.

[25]B. F. Wu, H. Y. Peng, and T. L. Yu, “Efficient Hierarchical Motion Estimation Algorithm and Its VLSI Architecture”, accepted by *IEEE Transactions on Very Large Scale Integration Systems*, Sep. 2007.

[26]G. Sorwar, M. Murshed, and L. S. Doolwey, “A Fully Adaptive Distance-Dependent Thresholding Search (FADTS) Algorithm for Performance-Management Motion Estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, Apr. 2007.

[27]B. F. Wu, H. Y. Peng, C. J. Chen, and Y. H. Chan, “An Encrypted Mobile Embedded Surveillance System,” in *Proc. 2005 IEEE Intelligent Vehicles Symposium*, IV05, pp. 502-507, June 6-8, 2005, Las

Vegas, Nevada, USA.

[28]L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, ”Real-Time for Monitoring Driver Vigilance,” *IEEE Trans. Intelligent Transportation Syst.*, vol. 7, no. 1, pp. 63-77, Mar. 2006.

[29]R. C. Gonzalez and R. E. Woods, “Digital Image Processing,” 2nd Edition, *Prentice Hall*, 2002.

[30]Y.-T Kim, “Contrast Enhancement Using Brightness Preserving Bi-Histogram Equalization,” *IEEE Trans. Consum. Electr.*, vol. 7, no. 4, pp. 304-312, 1988.

[31]Y. Wang, Q. Chen, and B. M. Zhang, “Image Enhancement based on Equal Area Dualistic Sub-image Histogram Equalization Method,” *IEEE Trans. Consum. Electr.*, vol. 45, no. 1, pp. 68-75, 1999.

[32]S.-D. Chen and A. R. Ramli, “Minimum Mean Brightness Error Bi-Histogram Equalization in Contrast Enhancement,” *IEEE Trans. Consum. Electr.*, vol. 49, no. 4, pp. 1310-1319, 2003.

[33]C. Wang and Z. Ye, “Brightness Preserving Histogram Equalization with Maximum Entropy: A Variational Perspective,” *IEEE Trans. Consum. Electr.*, vol. 51, no. 4, pp. 1326-1334, 2005.

[34]G. B. Thomas, *Calculus and Analytic Geometry*, 4th ed., New York: Addison-Wesley, 1968.

[35]B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Engle-wood Cliffs, NJ: Prentice-Hall, 1985.

[36]J. Y. Stein, *Digital Signal Processing: A Computer Science Perspective*, New York: Wiley, 2000.

[37]E. Eweda and O. Macchi, “Convergence of the RLS and LMS adaptive filters,” *IEEE Trans. Circuits Syst.*, vol. 34, no. 7, pp. 799-803, Jul. 1987.

[38]S. C. Douglas, “A family of normalized LMS algorithms,” *IEEE Signal Process. Lett.*, vol. 1, no. 3, pp. 49-51, Mar. 1994.

[39]K. Meghriche, S. Femmam, B. Derras, and M. Haddadi, ”A non linear adaptive filter for digital data communication,“ in *Proc. 6th Int. Sump. Signal Process. Appl.*, Piscataway, NJ, 2001, vol. 1, pp. 304-306.

[40]DSPGateway. URL:http://dspgateway.sourceforge.net/pub/index.php

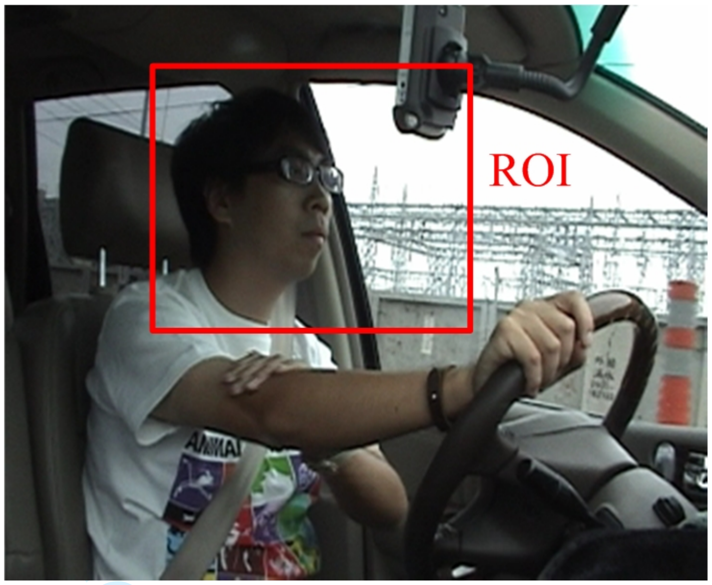


Fig. 1. The vehicle surveillance image with ROI.

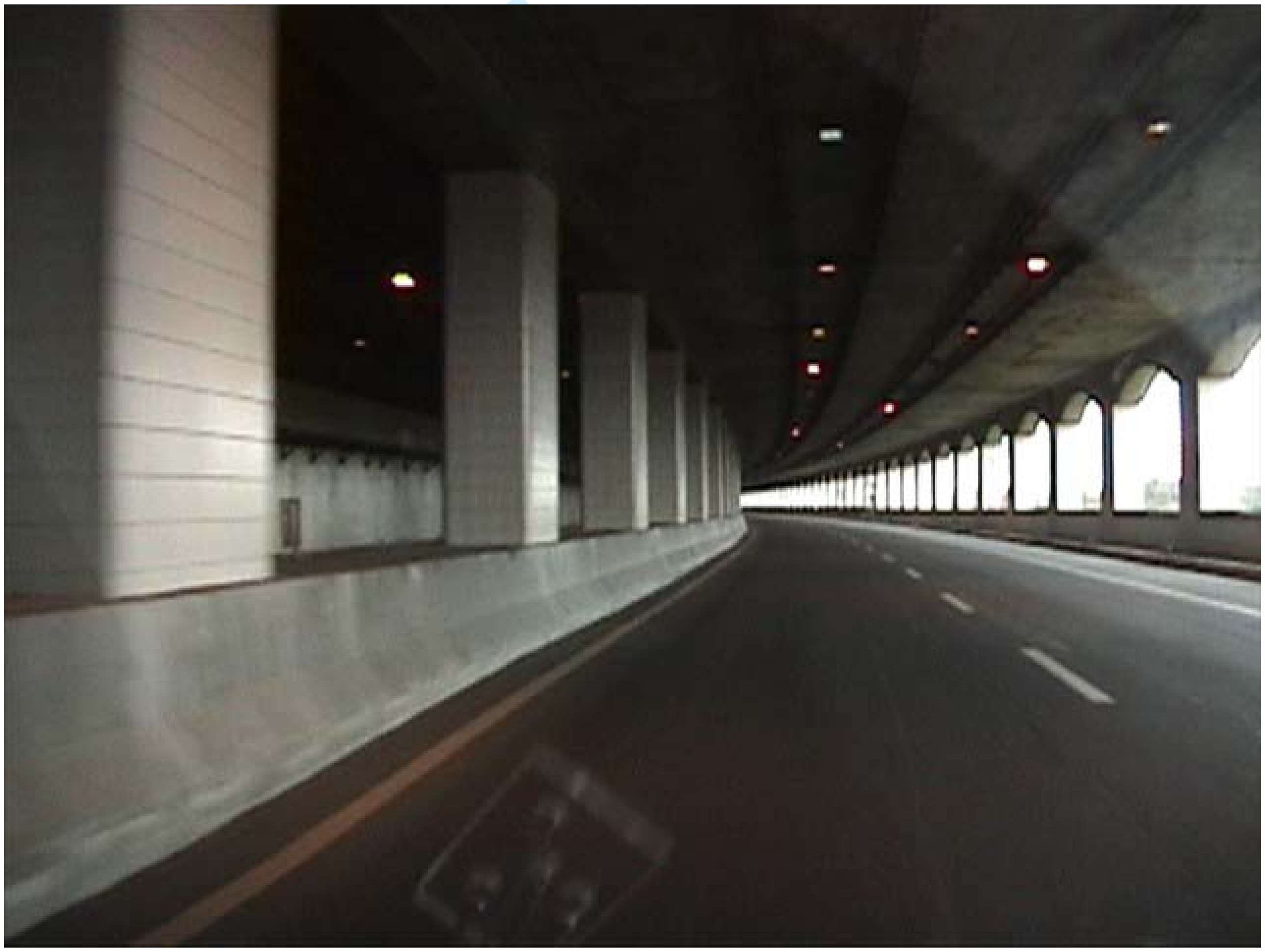
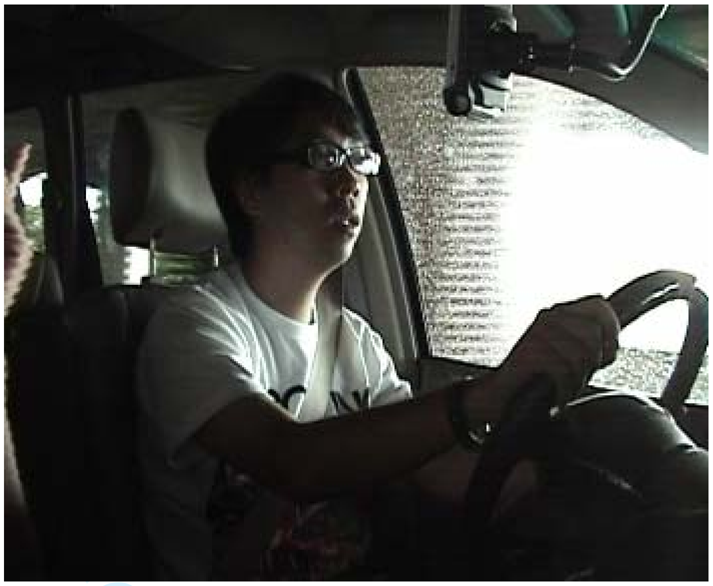
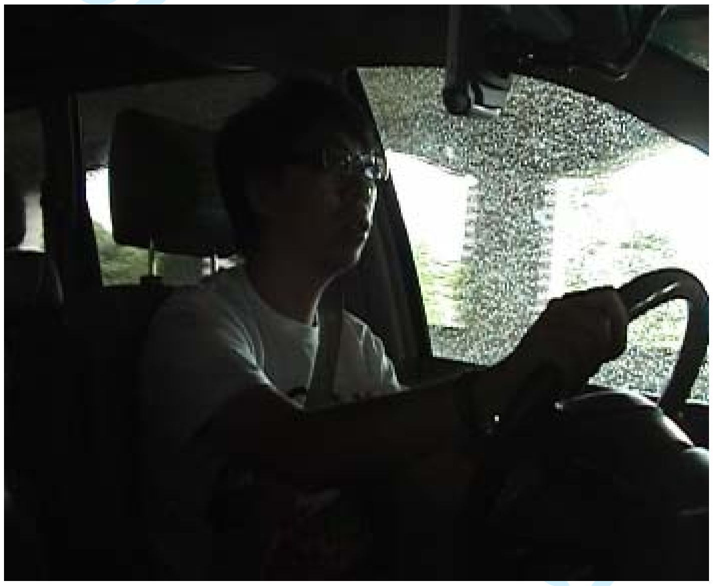


Fig. 2. The tunnel which will cause lighting effects.



(a)



(b)

Fig. 3. (a)The brightest image, (b)The darkest image of V1.

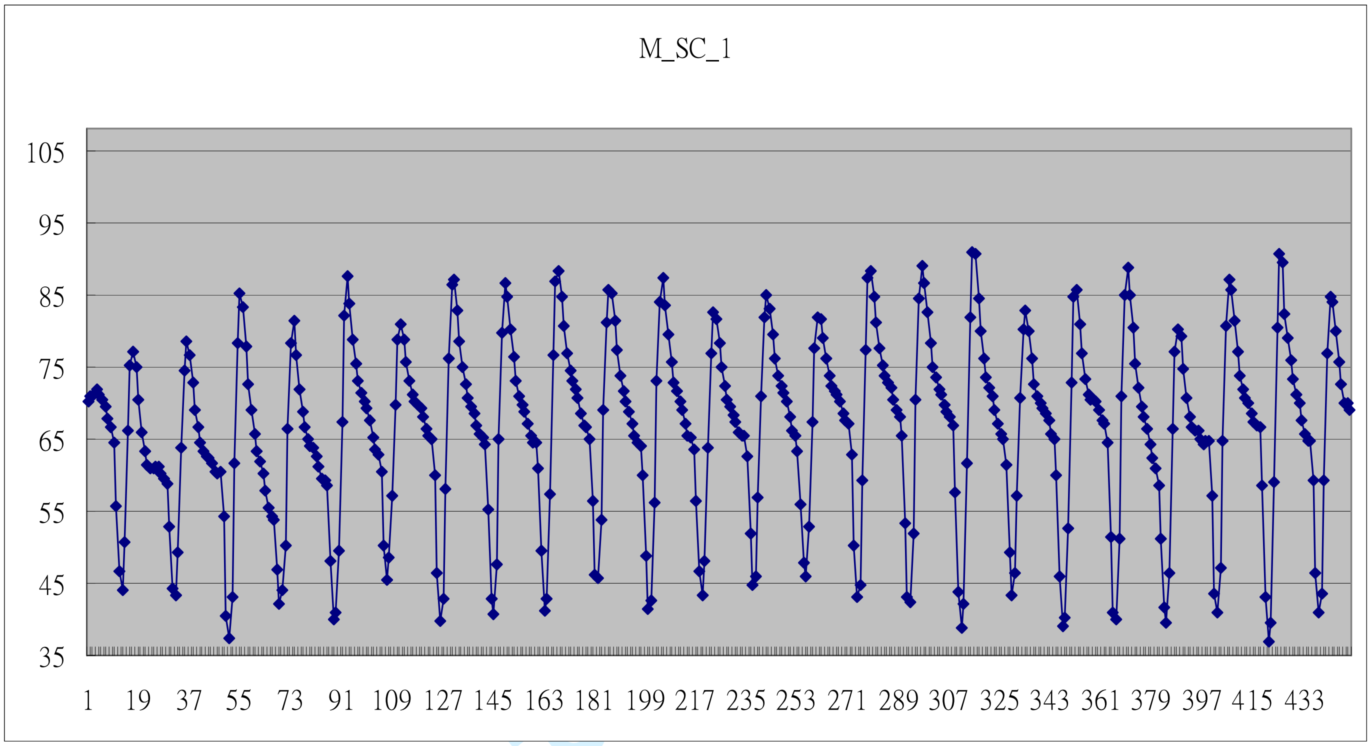
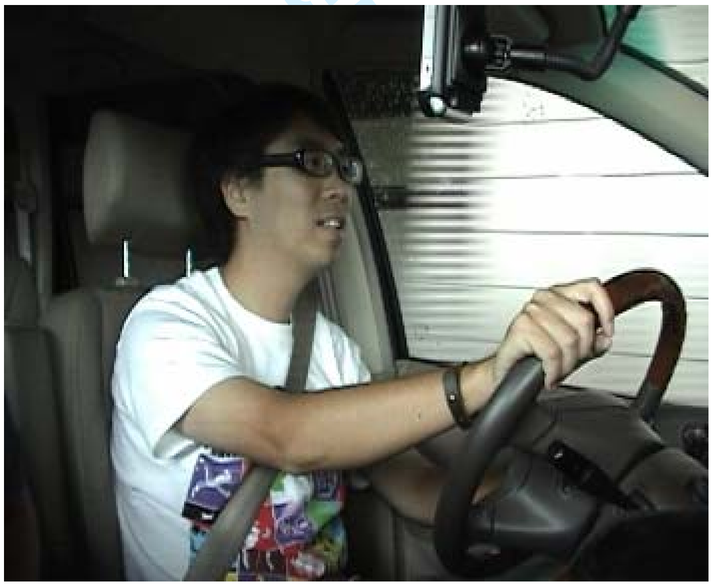
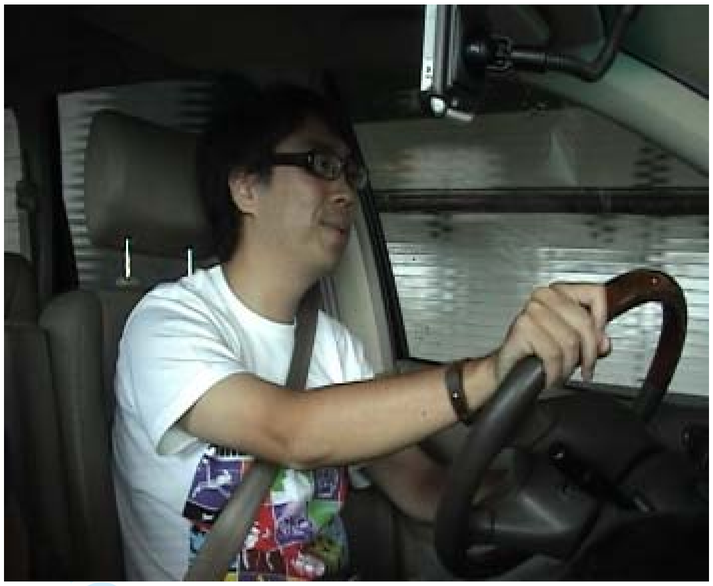


Fig. 4. The $\overline{M}_k$ of each frame of V1



(a)



(b)

Fig. 5. (a)The brightest image, (b)The darkest image of V2.

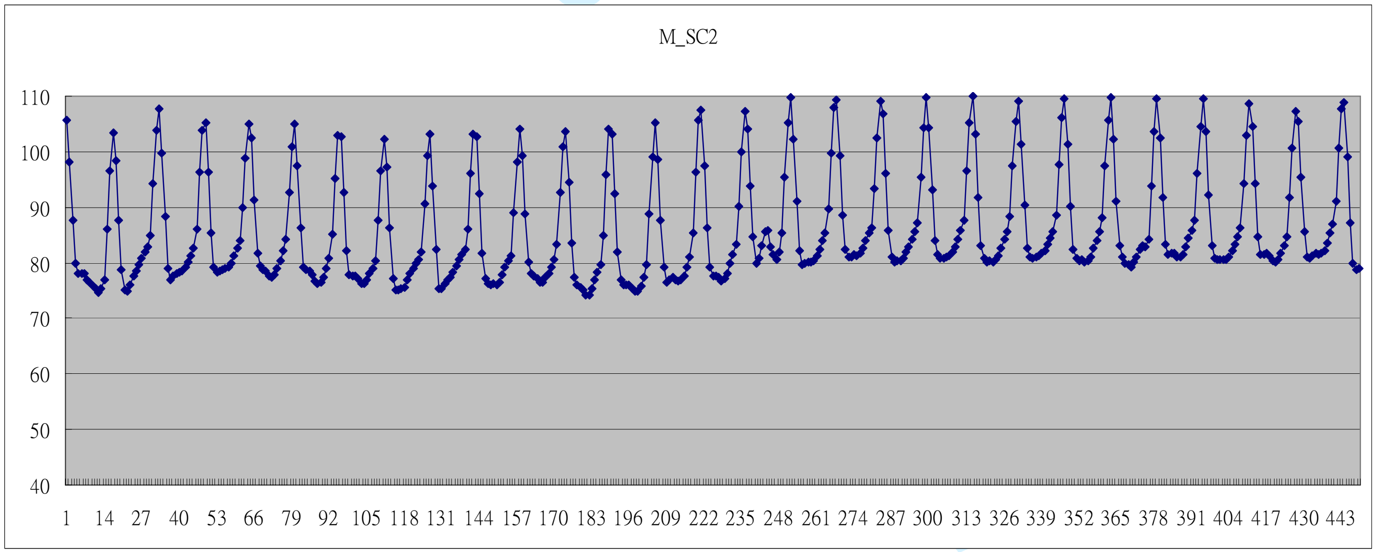


Fig. 6. The $\overline{M}_k$ of each frame of V2

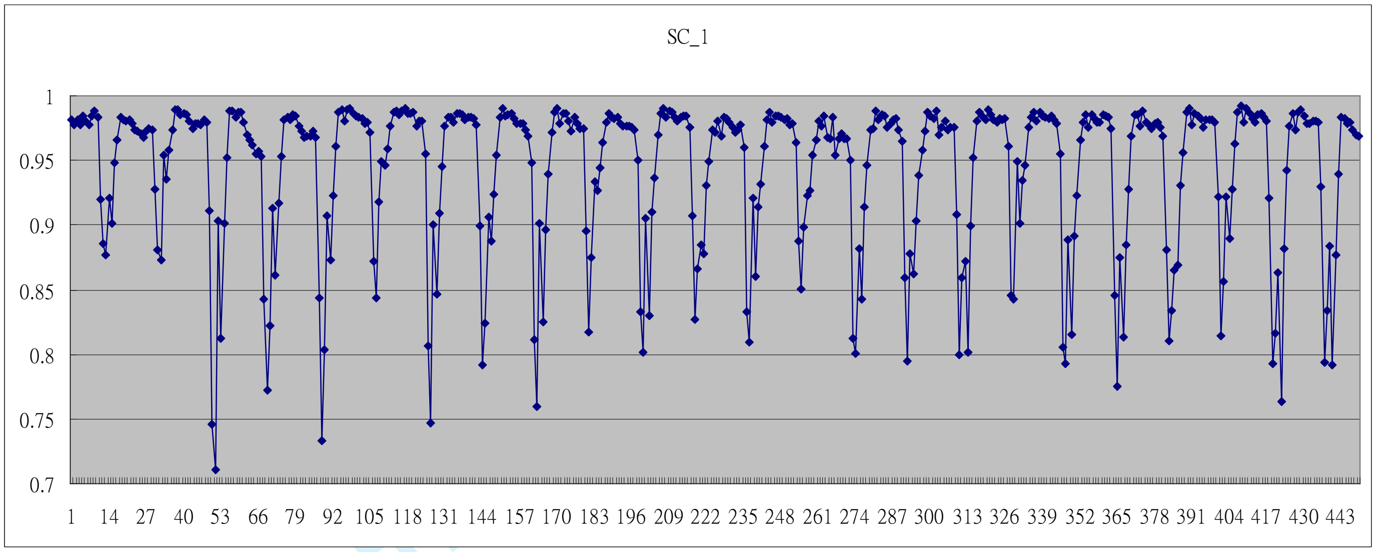


Fig. 7. The correlation between each frame pairs of V1

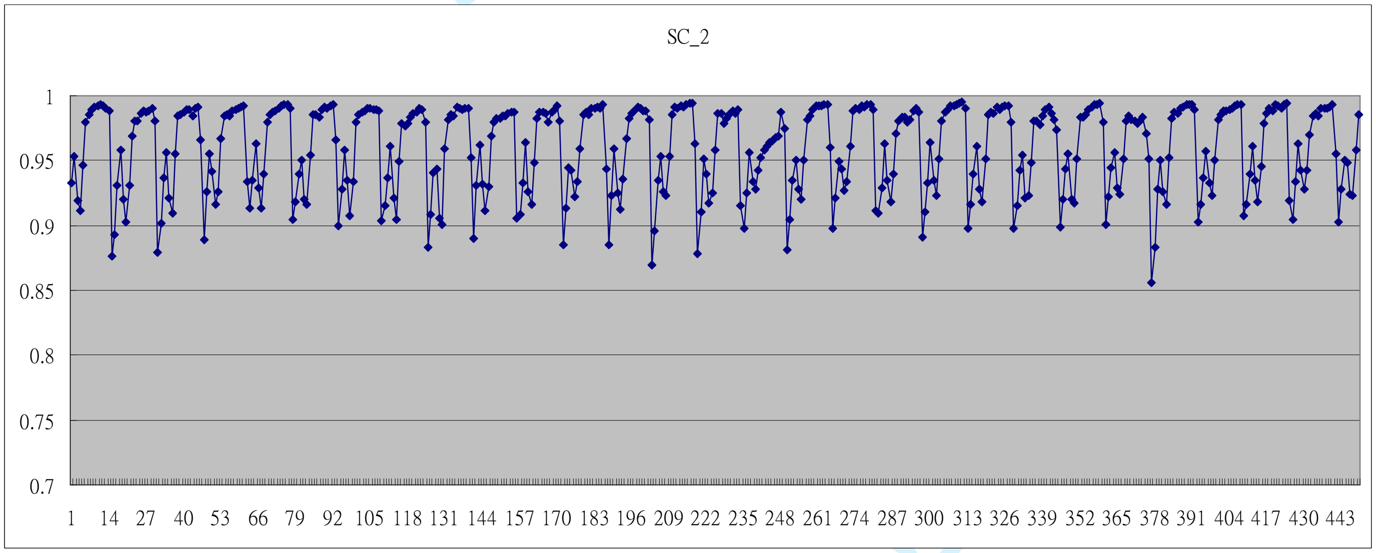
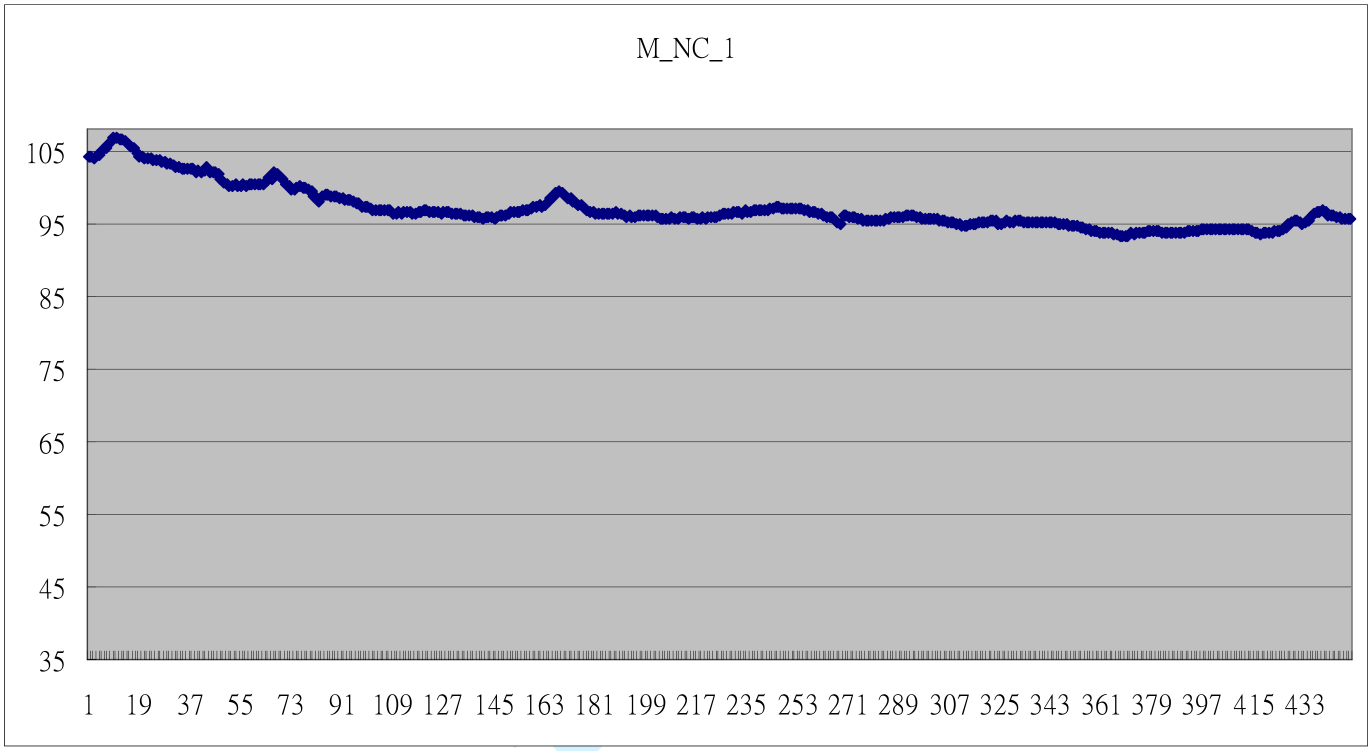
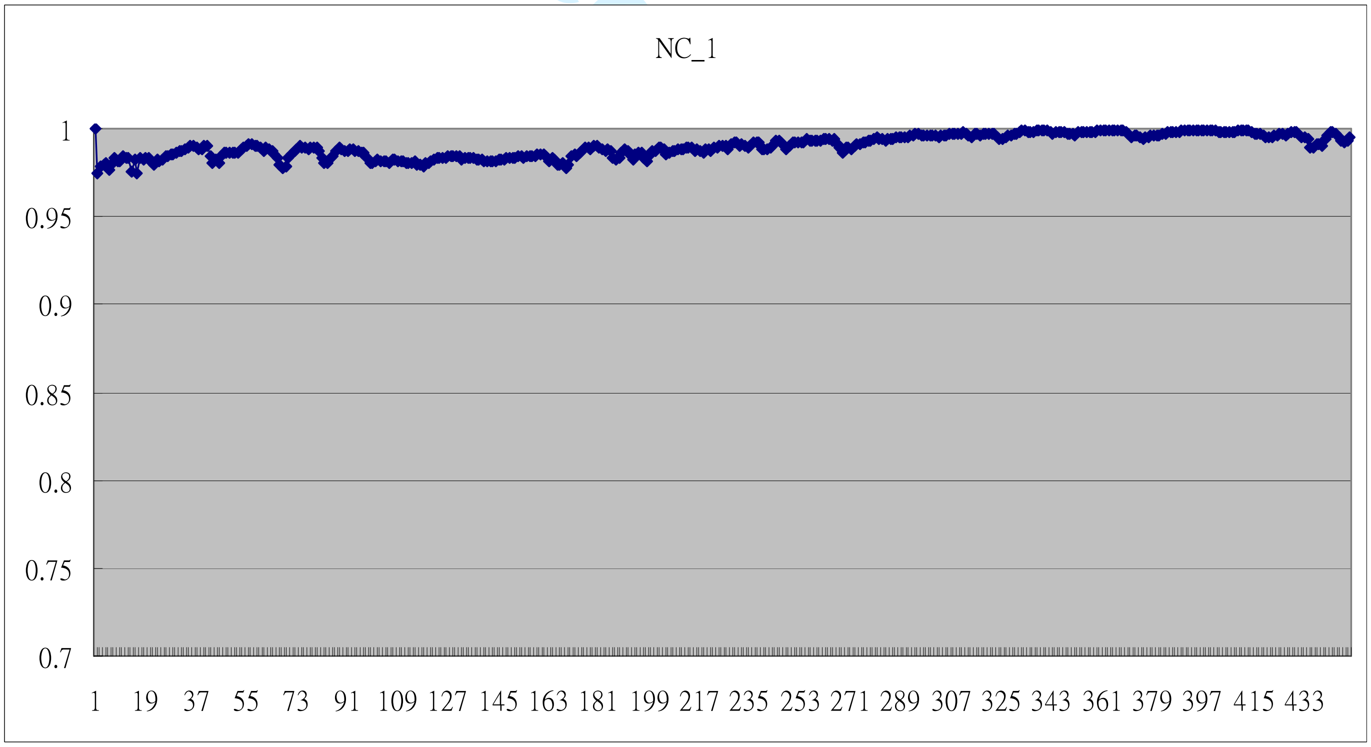


Fig. 8. The correlation between each frame pairs of V2



(a)



(b)

Fig. 9. (a) The $\overline{M}_k$ of each frame ,(b)The correlation between each frame pairs of V0

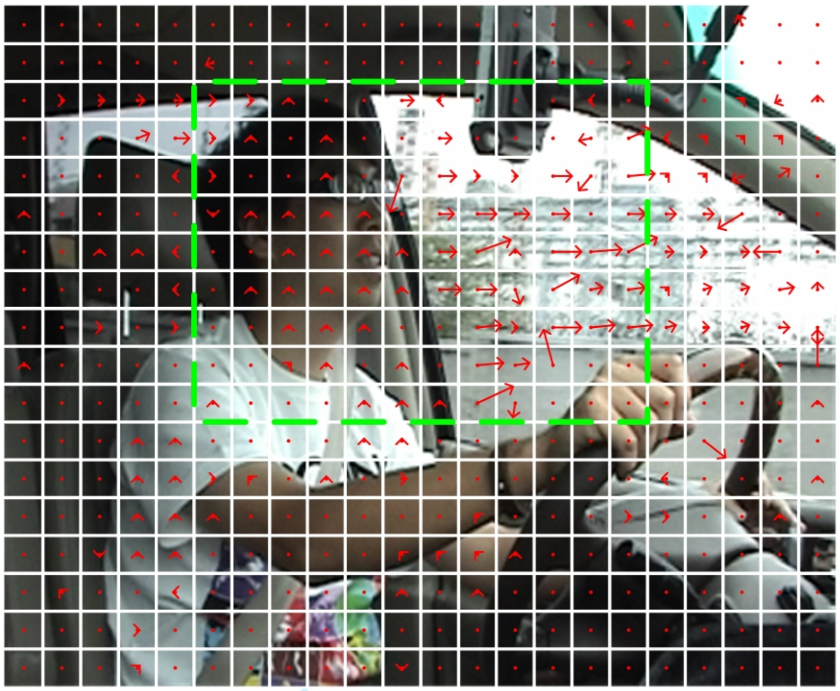
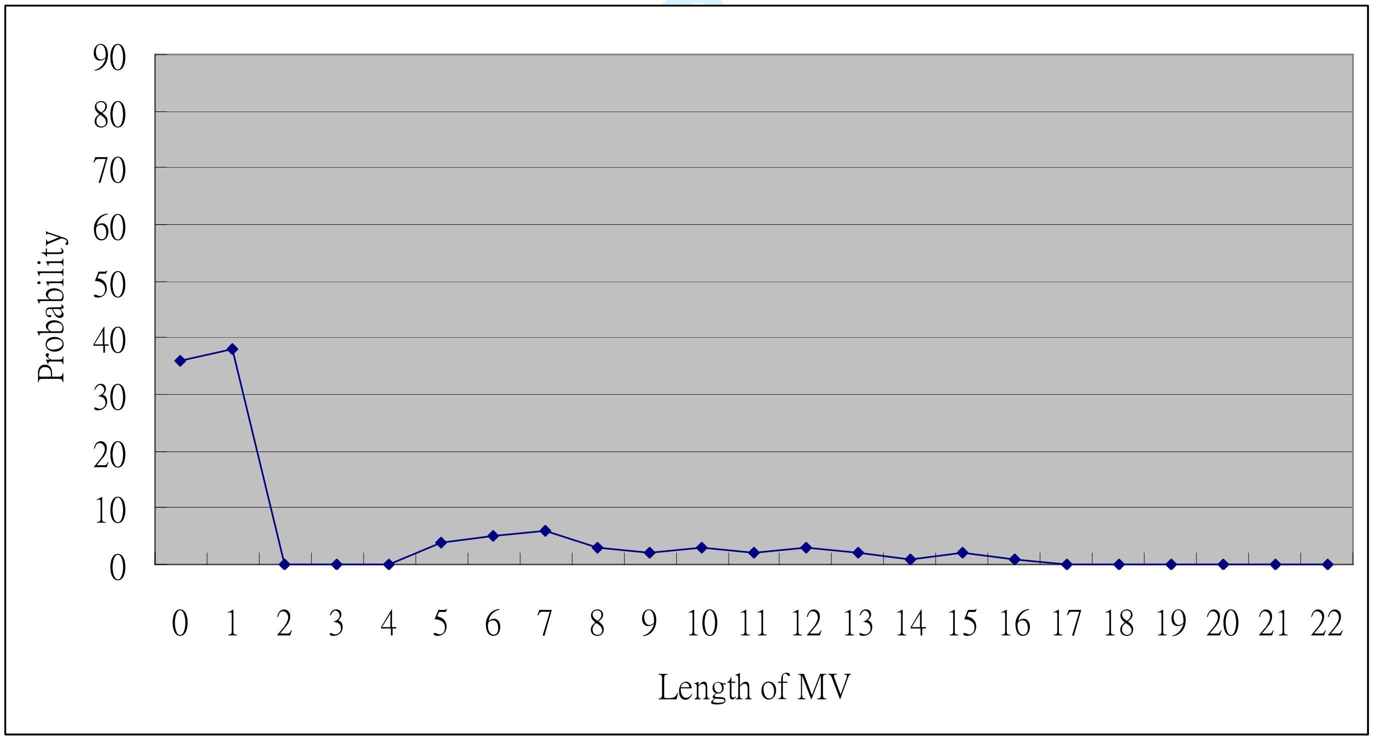
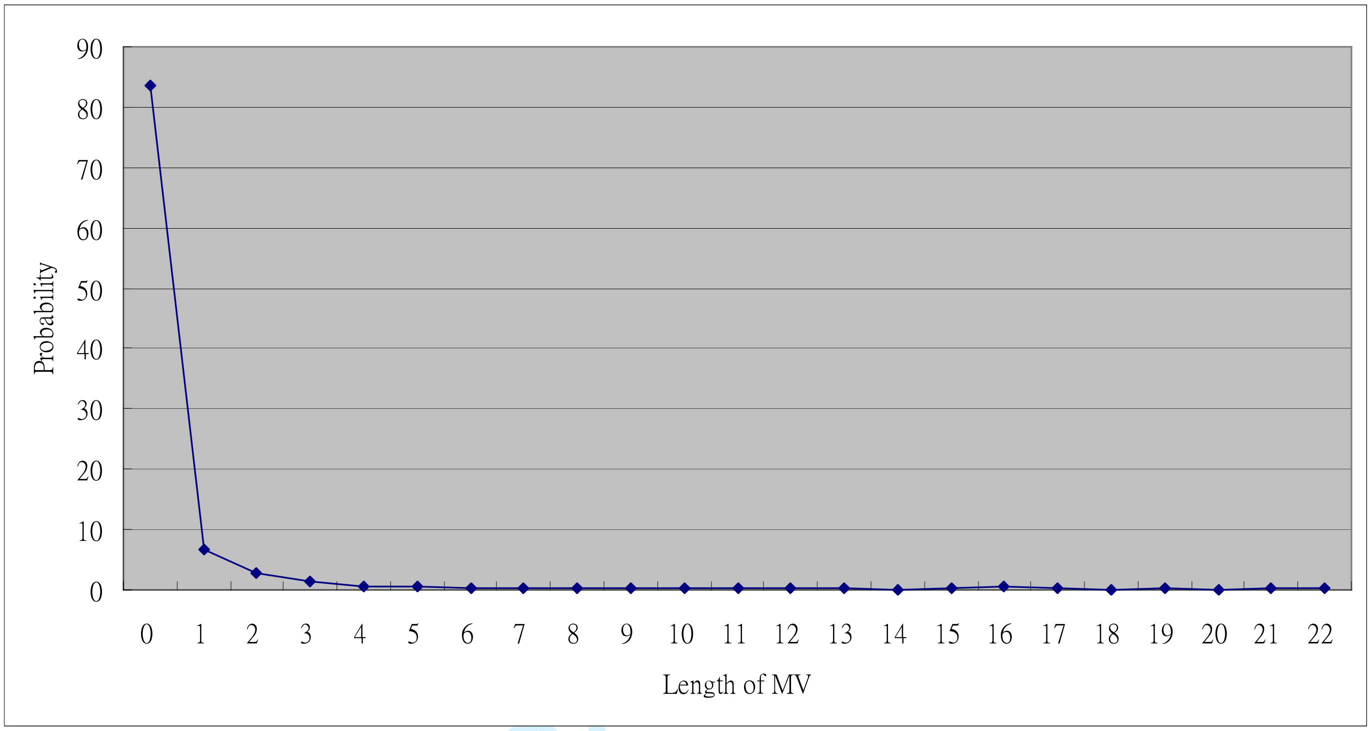


Fig. 10. The manipulated MVs

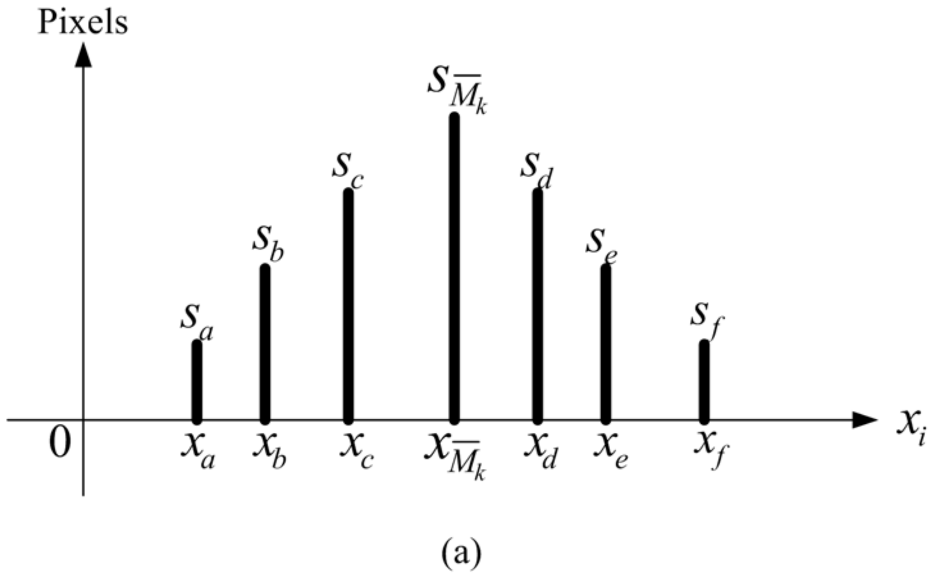


(a)



(b)

Fig. 11. (a) depicts the probability of the magnitude of MVs in and (b) out of ROI for all frames of V0.



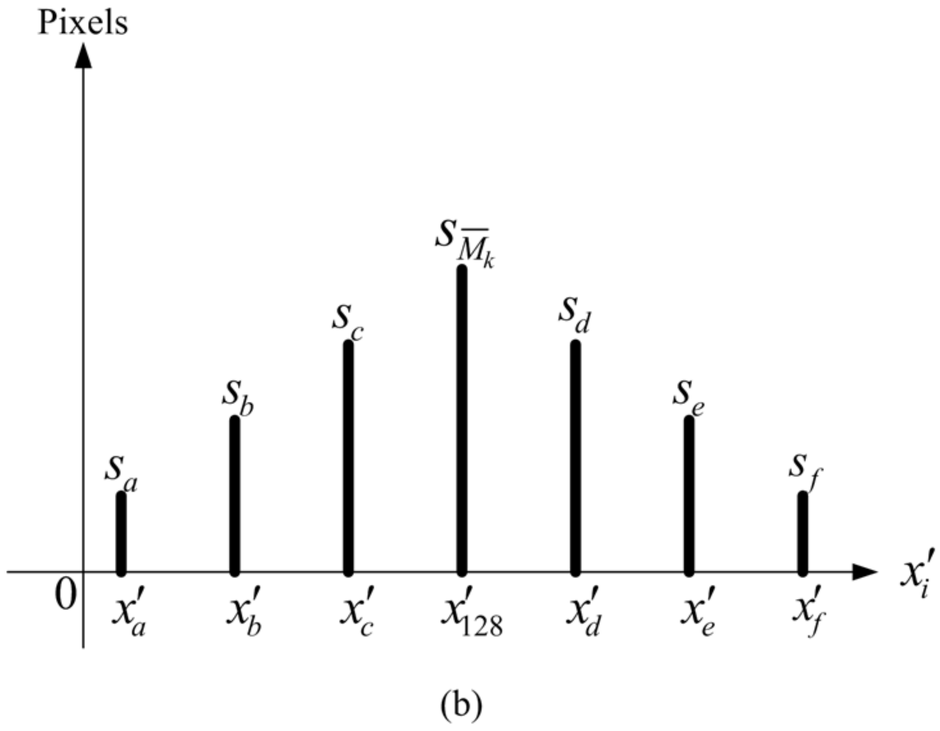


Fig. 12. (a) the histograms of the input frame before and (b) after the pre-processing.

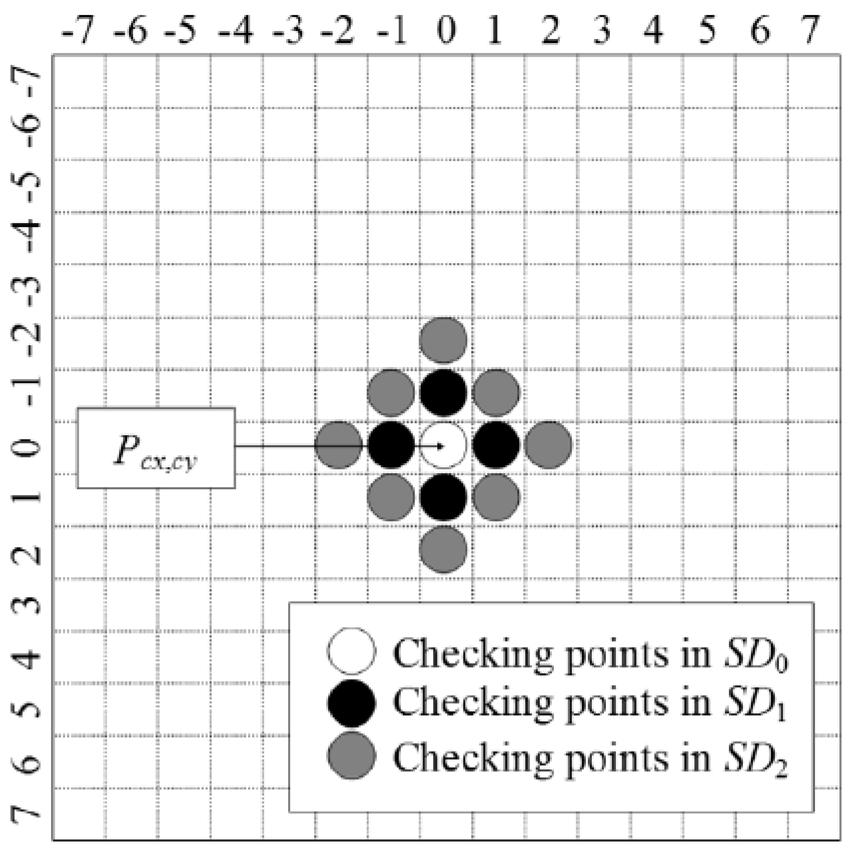

-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
$P_{cx,cy}$
Checking points in SD_0
Checking points in SD_1
Checking points in SD_2

Fig. 13. AMEA search diamonds SD_0, SD_1 and SD_2.

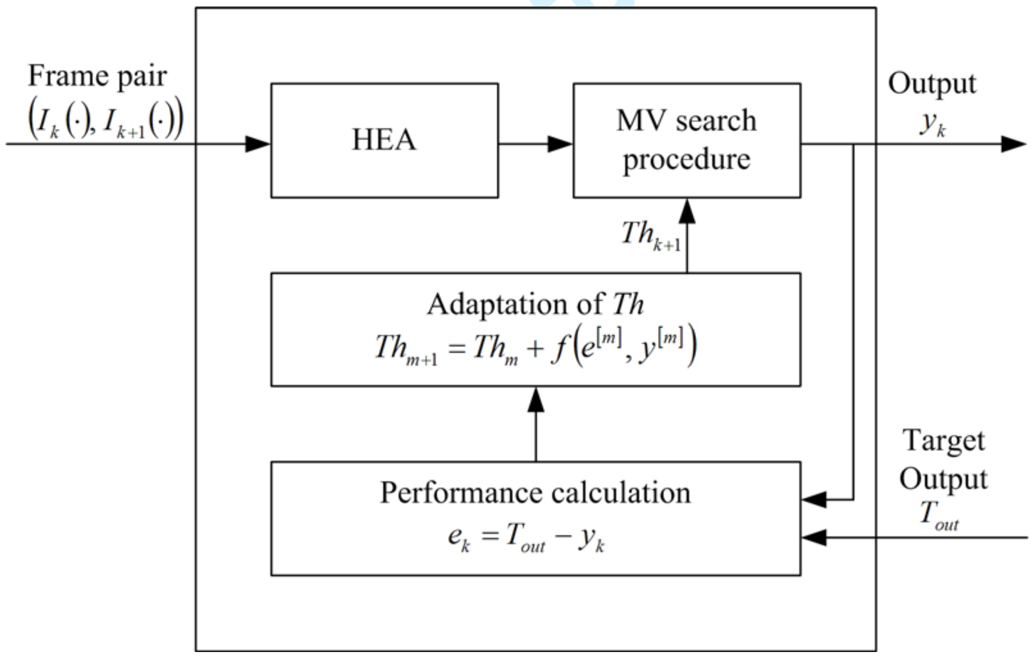
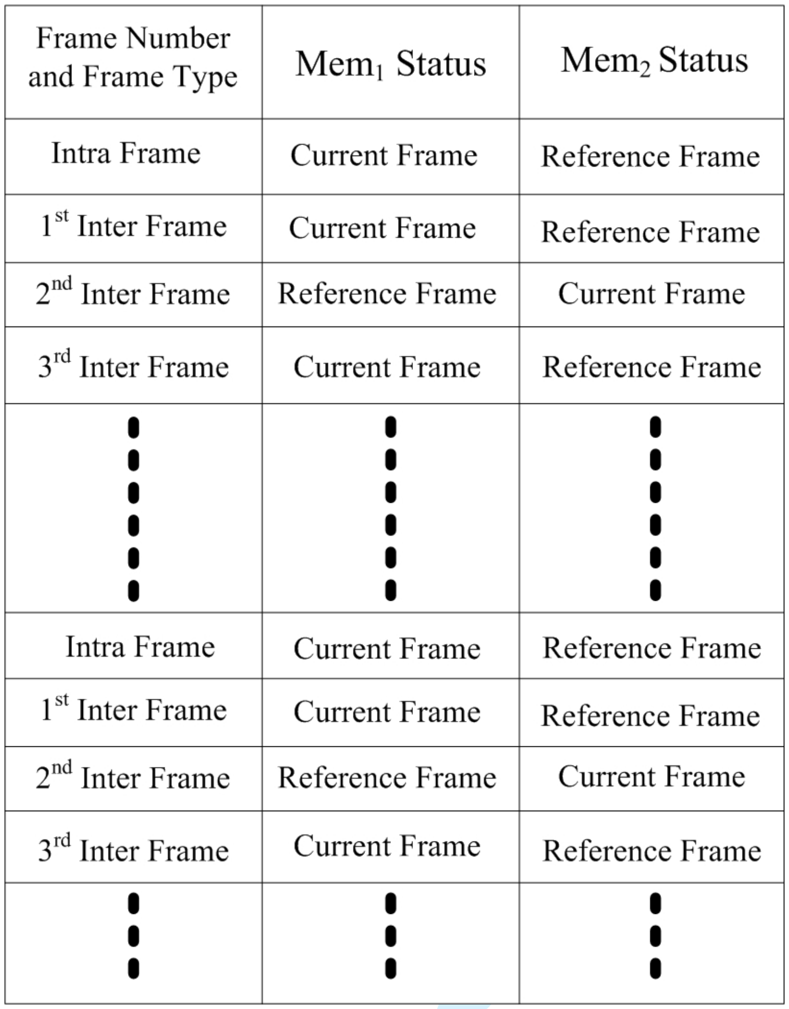


Fig. 14. Closed-loop adaptation process for AMEA.



Frame Number and Frame Type	Mem_1 Status	Mem_2 Status
Intra Frame	Current Frame	Reference Frame
1st Inter Frame	Current Frame	Reference Frame
2nd Inter Frame	Reference Frame	Current Frame
3rd Inter Frame	Current Frame	Reference Frame
⋮	⋮	⋮
Intra Frame	Current Frame	Reference Frame
1st Inter Frame	Current Frame	Reference Frame
2nd Inter Frame	Reference Frame	Current Frame
3rd Inter Frame	Current Frame	Reference Frame
⋮	⋮	⋮

Fig. 15 The state of the ping-pong buffer

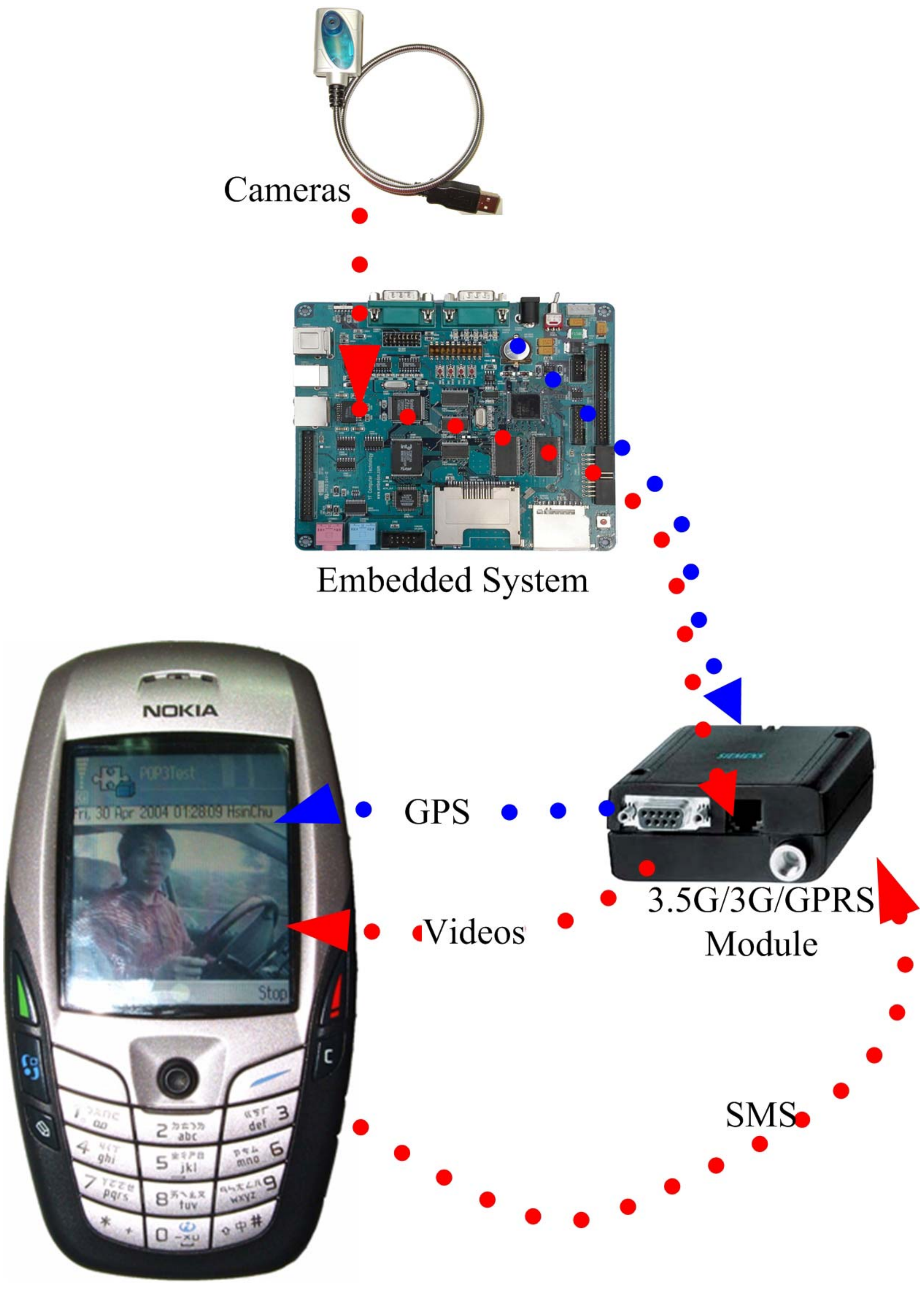


Fig. 16 The hardware architecture of EMVSS

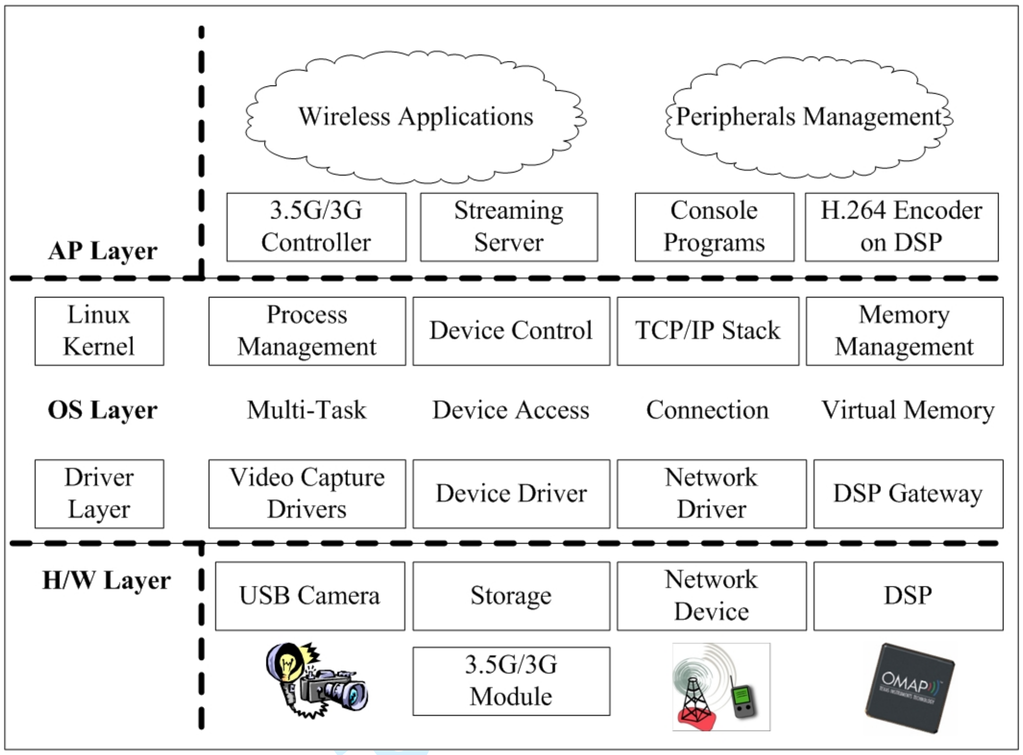


Fig. 17 The software architecture of EMVSS

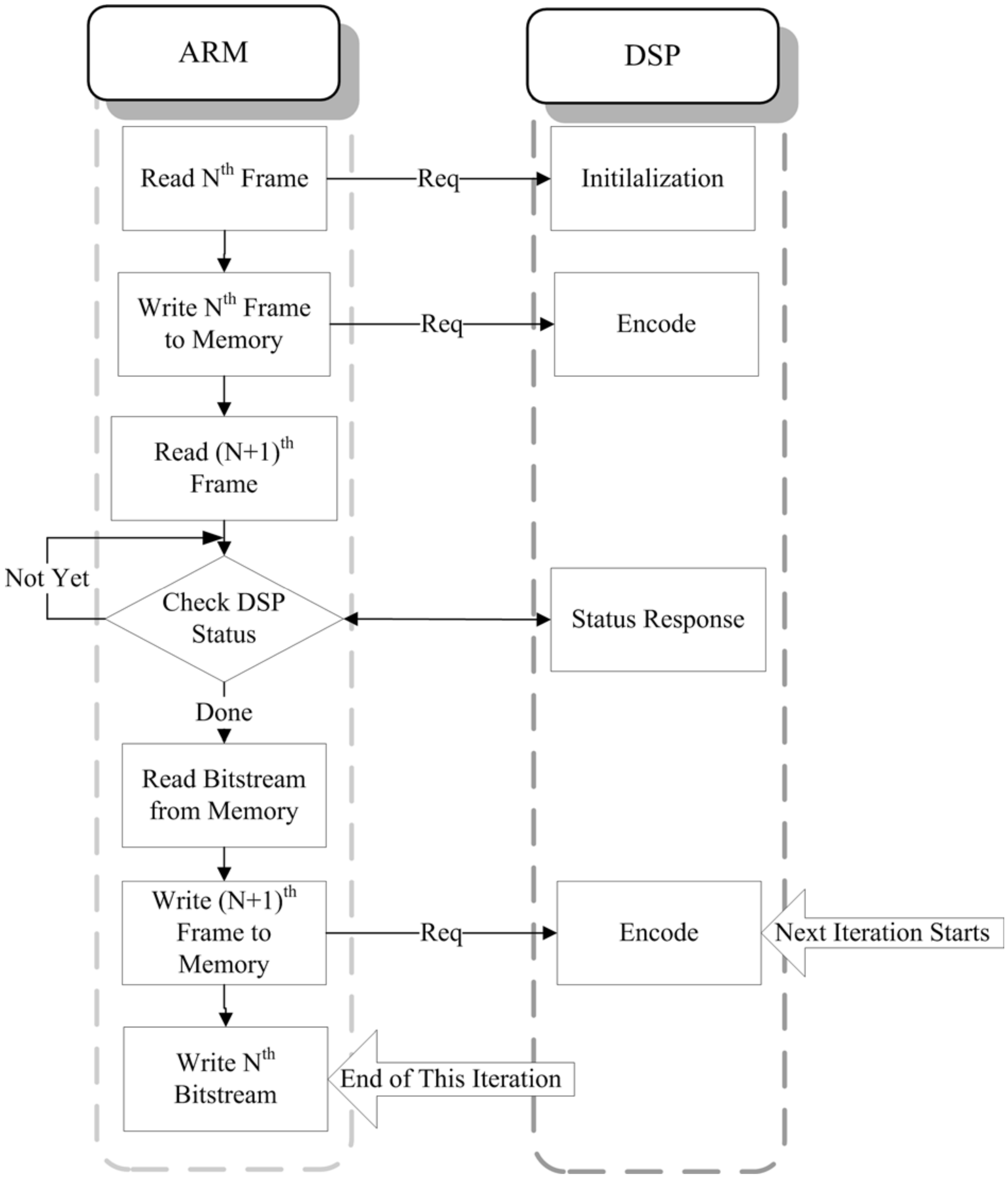


Fig. 18 The proposed H.264 encoding data path

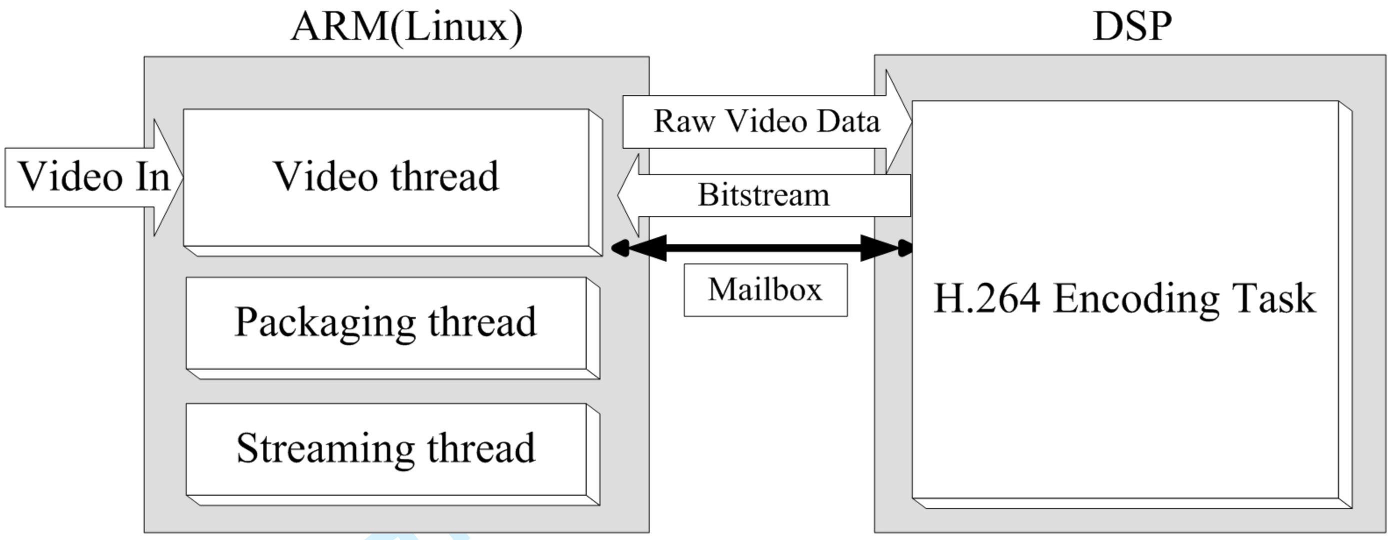


Fig. 19 The synchronization of audio and video encoding

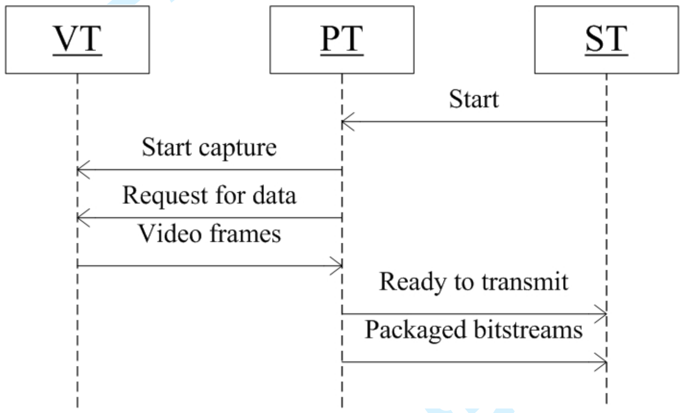


Fig. 20 Communication between related threads

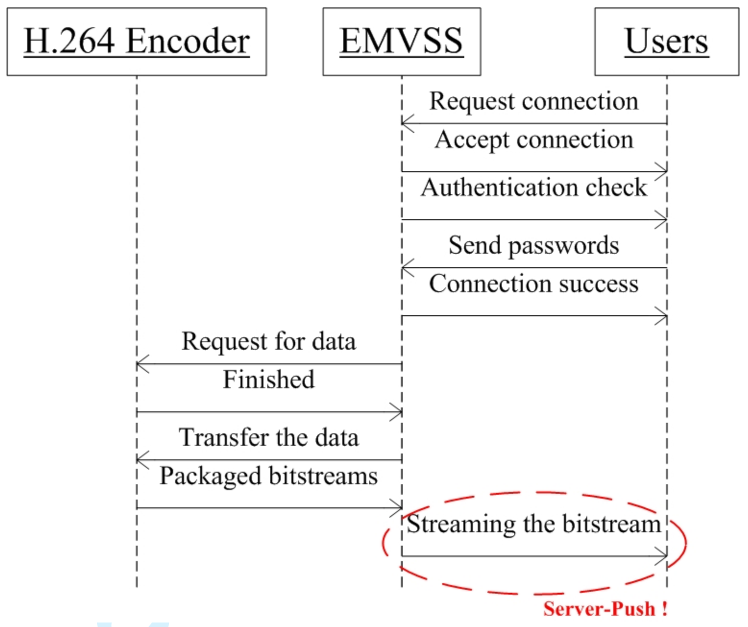


Fig. 21 The state chart of server-push we addressed.

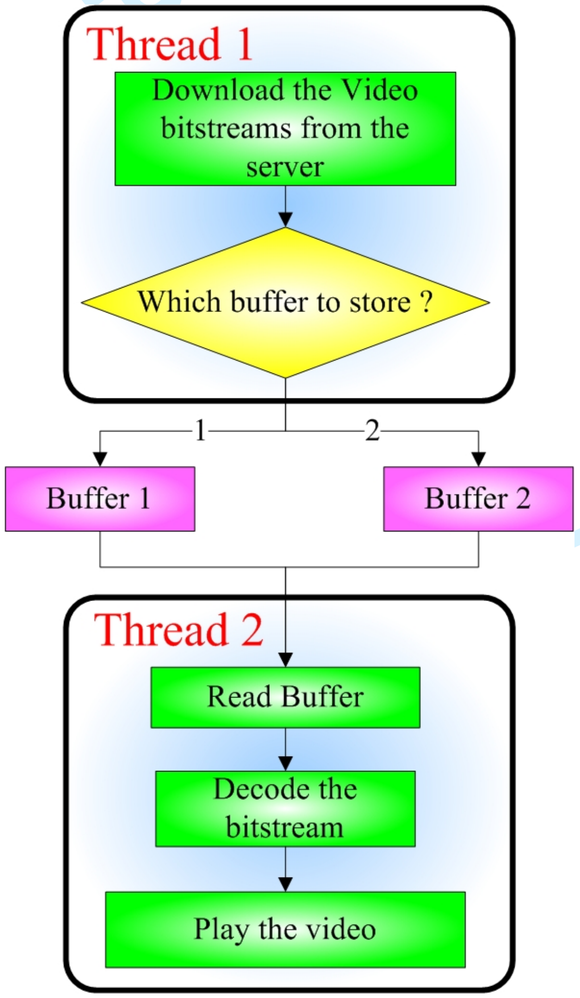
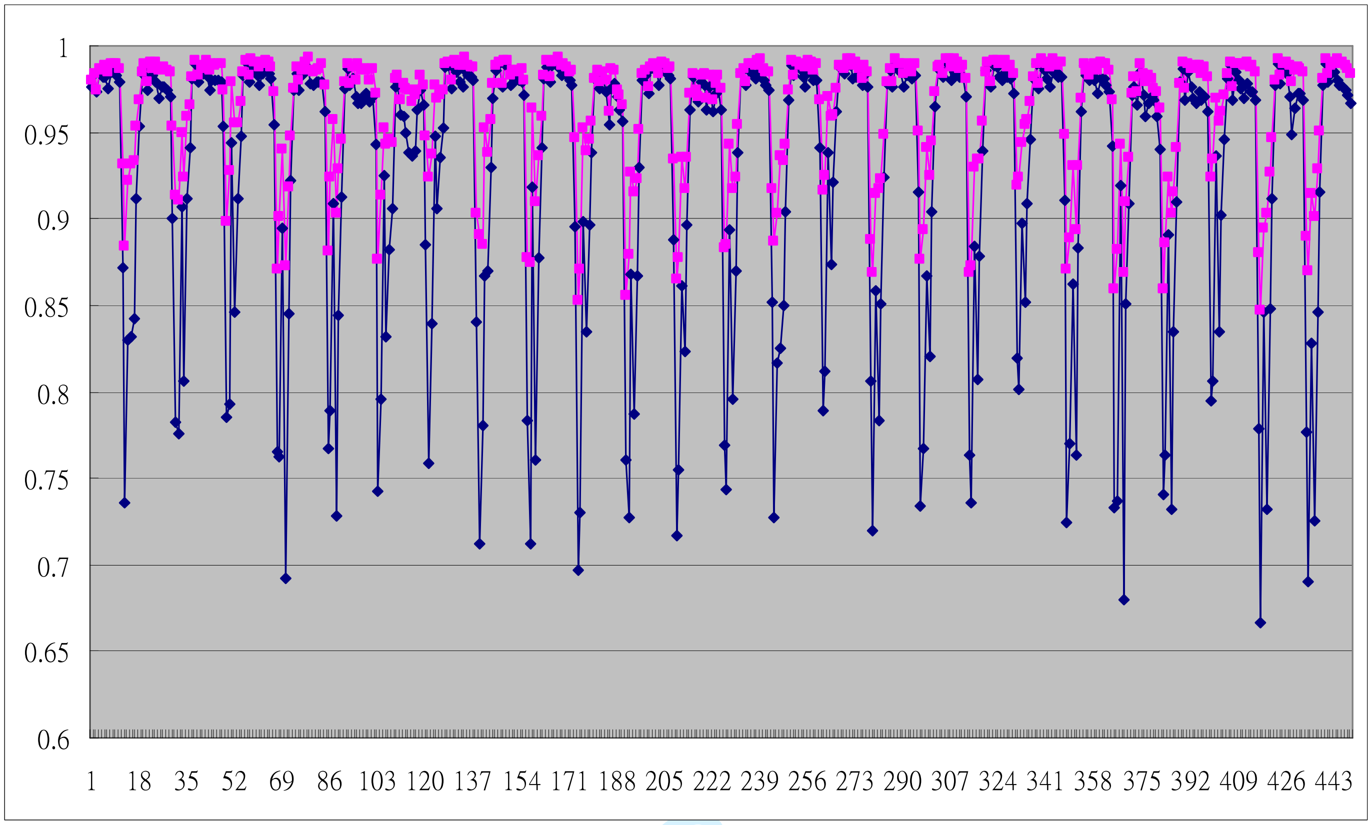
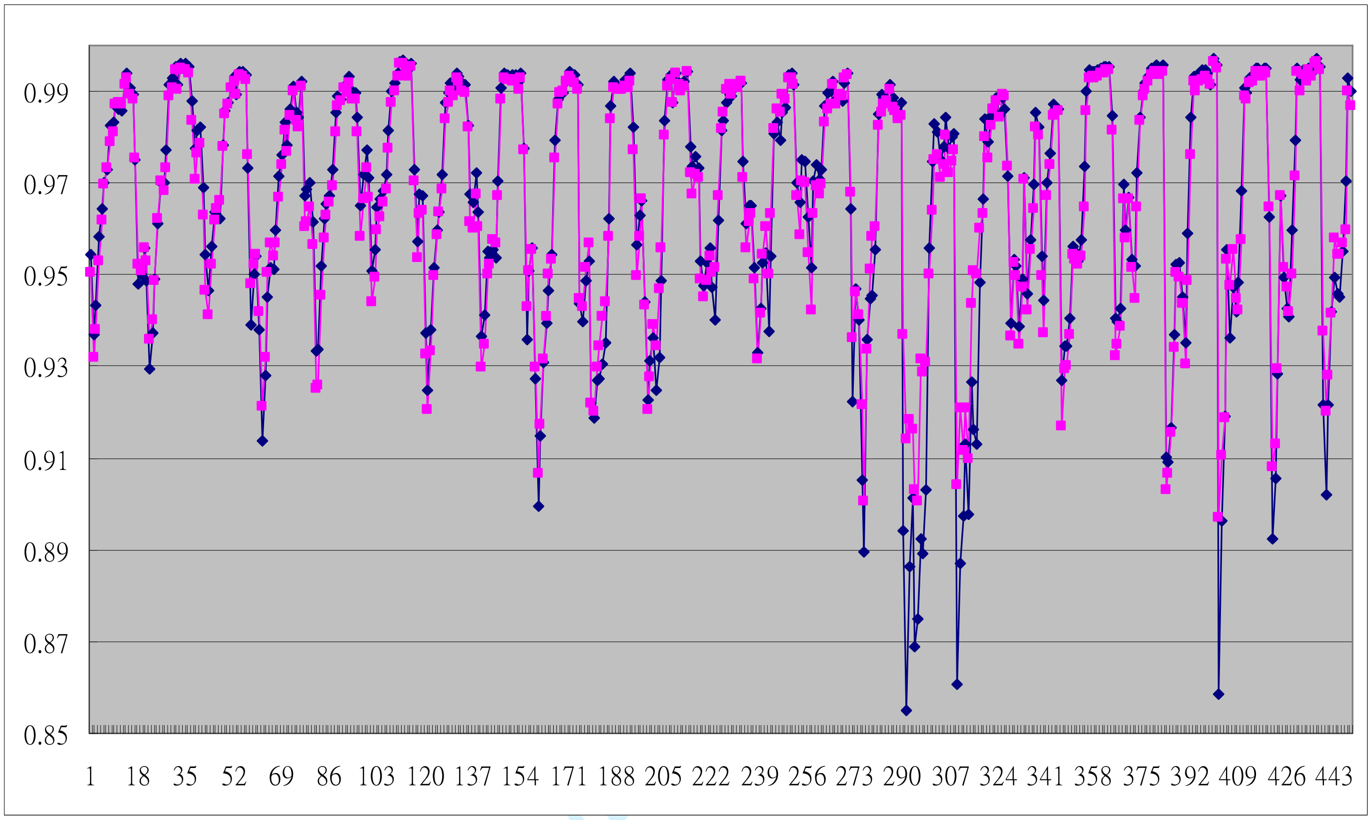


Fig. 22 The processing flowchart of the program on a mobile phone

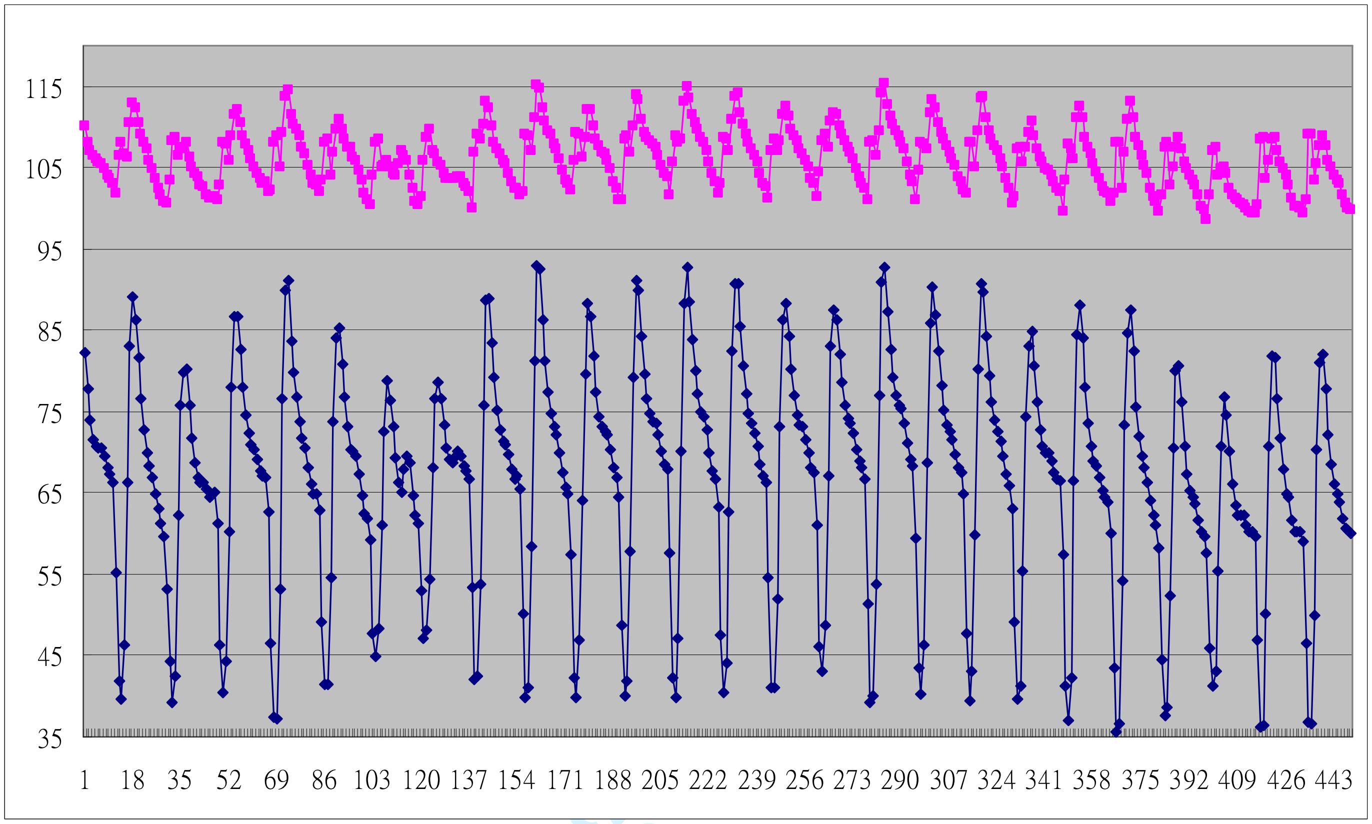

1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
1
18
35
52
69
86
103
120
137
154
171
188
205
222
239
256
273
290
307
324
341
358
375
392
409
426
443

(a)V1

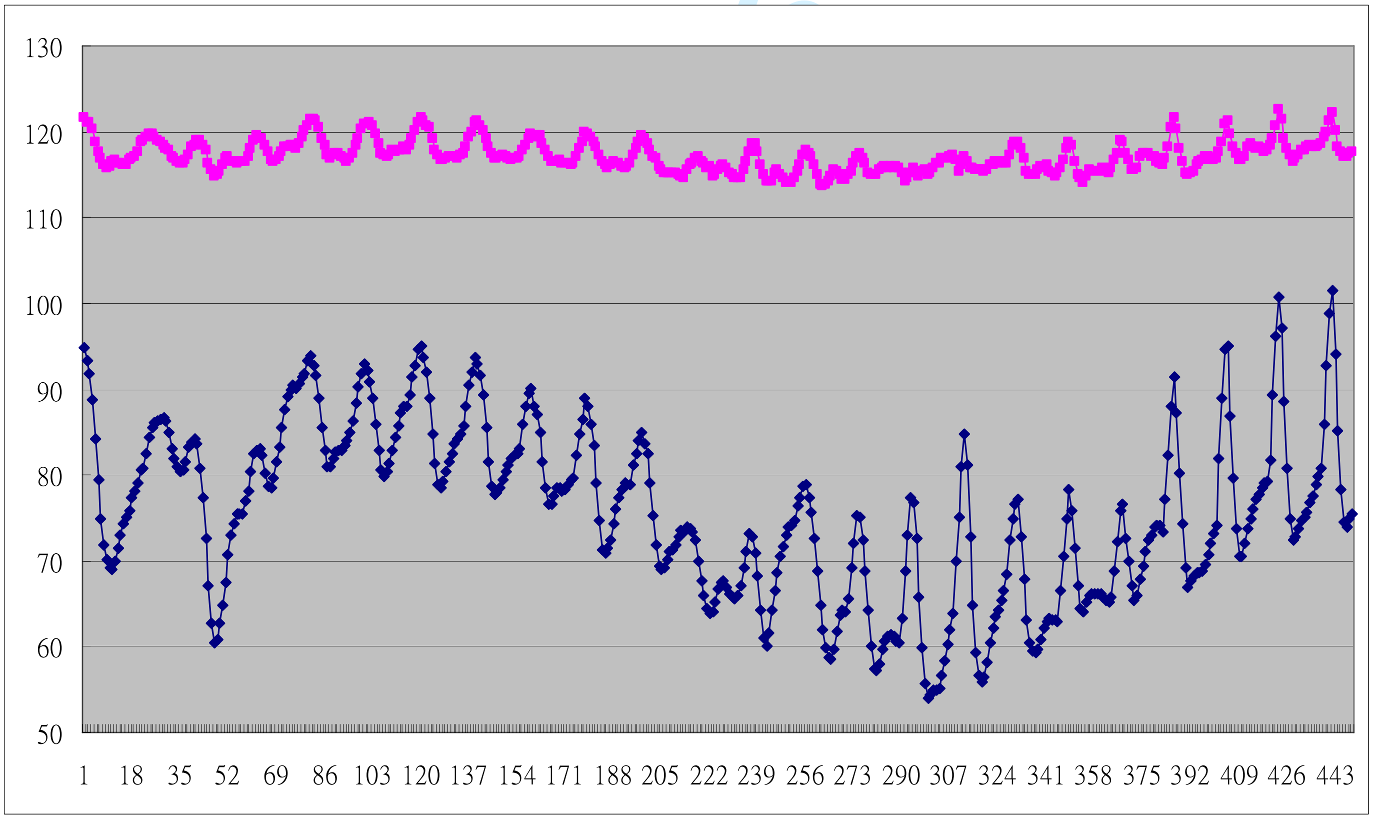


(b)V2

Fig. 23 The correlation between each frame pairs of (a)V1 and (b)V2 before and after HEA


115
105
95
85
75
65
55
45
35
1 18 35 52 69 86 103 120 137 154 171 188 205 222 239 256 273 290 307 324 341 358 375 392 409 426 443

(a)


130
120
110
100
90
80
70
60
50
1 18 35 52 69 86 103 120 137 154 171 188 205 222 239 256 273 290 307 324 341 358 375 392 409 426 443

(b)

Fig. 24 The $\overline{M}_k$ of each frame of (a)V1 and (b)V2 before and after HEA

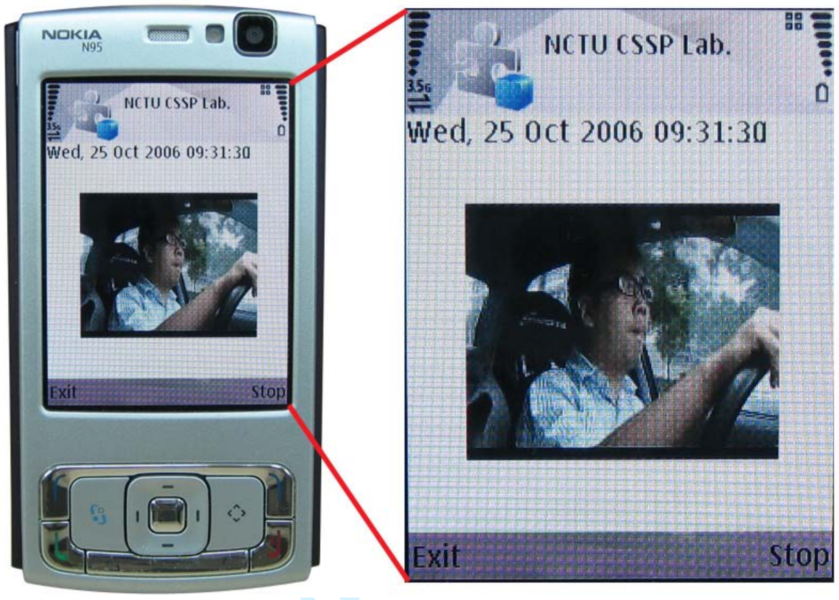


Fig. 25 The GUI developed on mobile phones.

Table I The encoding quality comparisons of the different videos in dB.

Video Sequence	PSNR
V0	34.09
V1	25.23
V2	27.55

Table II The quality comparisons of the blocks in and out of ROI of different video sequences in dB.

Video Sequence	AMEA		FSBMA		FADTS	
	In	Out	In	Out	In	Out
V0	30.37	34.62	32.27	35.38	30.75	30.86
V1	29.86	31.35	24.73	25.91	24.67	25.79
V2	30.12	32.97	26.24	28.47	26.16	28.42
Video Sequence	HEXBS		DSA		NTSS	
	In	Out	In	Out	In	Out
V0	31.87	35.13	31.95	35.17	30.55	34.87
V1	23.82	25.67	23.89	25.74	22.78	23.19
V2	25.91	28.03	26.11	28.12	25.07	26.55

Table III The normalized processing speed comparisons of the different videos

Video Sequence	AMEA	FSBMA	FADTS
V0	0.07	1	0.25
V1	0.15	1	0.89
V2	0.14	1	0.77
Video Sequence	HEXBS	DSA	NTSS
V0	0.11	0.12	0.07
V1	0.21	0.22	0.07
V2	0.16	0.17	0.07

Table IV The specification of EMVSS.

Processor		OMAP5912 @ 192 MHz	
Profile		H.264 baseline	
Frame rate	Only encode	7 frames/second	
	Overall	4 frames/second	
Average compression ratio		91.78	
Image size		QCIF	
Power consumption		Active	5W
		Suspend	20 μW
Transmission method		3.5G / 3G	
Storage		Compact Flash Card	