A Study on Multi-Screen Sharing System Using H.264/AVC Encoder

Shizhe Tan and Fengyuan Zhang
Department of Electronic Engineering, Ocean University of China, China

Abstract: *H.264/AVC is a standard for video compression developed by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC JTC1 Moving Picture Experts Group (MPEG). However, the computational complexity of H.264/AVC contributed a lot to the delay time of multi-screen sharing system. In this paper, the motion estimation algorithms provided in X264 have been analysed, and an optimized algorithm has been proposed, this optimized algorithm can reduce a lot of unnecessary computation. And more, this paper designed and implemented the multi-screen sharing system with the improved encoder. Experimental results show that the proposed method has increased the encoding speed and decreased the delay time, while incurring little, if any, loss in quality.*

Keywords: *Multi-screen sharing system, H.264/AVC, motion estimation, delay time.*

Received August 25, 2013; accepted October 26, 2014.

1. Introduction

With the development of technology, a variety of intelligent home appliances constantly entered millions of households, and the concept of digital home has been widely accepted by consumers [1]. As one of the most common application of digital home, multi-screen sharing system aims at helping users to access their media resources stored on different devices whenever necessary [5, 13]. Currently, the study of multi-screen sharing system is not yet in-depth, home appliances on the market using it mostly based on the standard video encoder, users obtained poor experience because of the delay time of frames. An efficient video encoder is the key for designing a multi-screen sharing system.

H.264/AVC is a video coding standard that outperforms all of previous standards in terms of coding efficiency. To achieve a higher coding performance and better subjective visual quality, H.264/AVC uses many new techniques such as variable block-size motion estimation, multiple reference frames, in-the-loop deblocking filtering, and so on, which lead to a significant increase in computational complexity [6]. Motion estimation is the most important and time-consuming part of H.264/AVC and it amounts to 74.29 % computation of the encoder [2]. H.264/AVC adopts Block-Matching Algorithm (BMA) in motion estimation [6]. Researchers have proposed many methods for improving performance [8, 10, 11], which include fast BMAs, such as Three-Step Search (TSS) [7], Diamond Search (DS) [19], Hexagon Based Search (HEX) [18, 20], and Unsymmetrical-cross Multi-Hexagon-grid search (UMH) [17] to accelerate the process of block-matching with acceptable distortion performance.

In this paper, basing on the properties of frames, an optimized the HEX algorithm has been proposed. Experimental results show that the optimization has improved the speed of encoding. And more, we designed and implemented the multi-screen sharing system using the optimized H.264/AVC algorithm, and it shown that the improvement has shortened the delay time of frames by contrast.

The rest of this paper is organized as follows. In section 2, the process of H.264/AVC encoder and the existing motion estimation algorithms are analysed, and an optimized algorithm is proposed through research of statistical characteristics of Macro-Blocks (MBs) of video sequence. Section 3 designs the multi-screen sharing system using improved H.264/AVC encoder. We present the setup and results of our experiments in section 4 and we conclude in section 5.

2. The Optimization of H.264/AVC Encoder

2.1. Process of H.264/AVC Encoder

Generally, H.264/AVC encoder can be divided into Motion Estimation (ME), Motion Compensation (MC), intra prediction, Transform (T), Quantifier (Q), and entropy encode. The basic unit in H.264/AVC is MB which is illustrated in Figure 1.

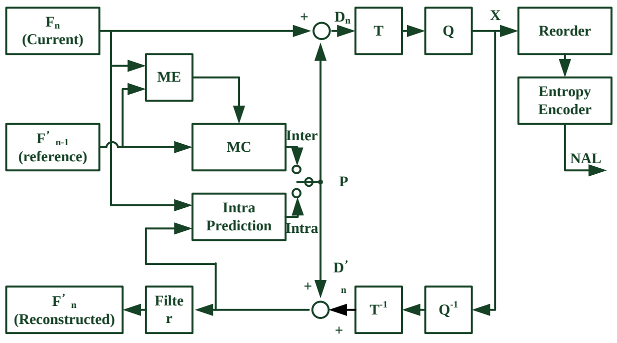


Figure 1. Flowchart of H.264/AVC encode.

In the Figure 1, F_n represents the current encoding frame, and each MB segmented from F_n select predict mode according to the type of F_n. In case of inter prediction, predicted MB (P) is achieved by ME and MC referring to the previous frame (F'_{n-1}), and in intra prediction, P is predicted through its neighbouring MBs from reconstructed frame (F'_n). After predictions, the residuals MB (Dn), which is achieved from the actual MB minus P, will be transformed and quantified. When all MBs have completed the above process, the Dns will be reordered and entropy encoded into bit stream. At the last step, Network Abstract Layer (NAL) adds the control information that decoder requires and NAL Unit (NALU) header into this bit stream, the entire work of one frame have done.

2.2. Motion Estimation Algorithms

For H.264/AVC standard, there are three major open-source projects, JM, X264, and T264. Among these projects, X264 abandoned some functions that have high computational complexity and contribute little to coding performance, so the practicability have been improved greatly. This paper chooses X264 as the research object.

X264 offers four full-pixel motion estimation algorithms, hadamard Transform Exhaustive Search (TESA), DS, HEX, and UMH [4].

TESA uses hadamard transform to motion vectors, then searches all possible candidate transformed vectors in a predetermined neighbourhood search window. Although, TESA produces the best quality, it demands the most computation.

DS in X264 was executed with small diamond pattern which is shown in Figure 2. Using smaller pattern is able to find the suitable vector quickly for the slowly moving MBs, but increases the risk of falling into local optimum when MBs have violent movement.

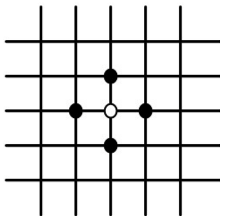


Figure 2. Small diamond pattern.

To avoid falling into local optimum, UMH uses a variety of patterns which have variable radius, and introduces the early termination judged. But compared with HEX, UMH is not suitable for real-time encoding because of the higher complexity of the algorithm.

HEX is the most widely used ME algorithm, which benefit from better search efficiency. The detailed description of each step is as follows:

- *Step* 1: Predict search starting point using spatial correlation. Calculate the average value (*MV*0(*x*0, *y*0)) of the motion vectors which belong to the adjacent MBs from the current block on the left, above, and upper left (*MV*1(*x*1, *y*1), *MV*2(*x*2, *y*2), *MV*3(*x*3, *y*3)). Use the above 4 vectors and vector of original point (0, 0) as candidate, then calculate the error cost of reference MB (*COST*) on those 5 points. The *COST* is given by Equation 1. Mark the minimal *COST* value as BPRED_COST, and the vector of BPRED_COST as PMV.

$$COST\,(i\,,j\,) = \sum_{m=1}^{M}\sum_{n=1}^{N}\left|C\,(m,n) - R\,(m+i\,,n+j\,)\right| \tag{1}$$

COST(*i*, *j*) is the error cost of motion vector (*i*, *j*), and *C*(*m*, *n*) and *R*(*m*+*i*, *n*+*j*) are pixel values in the current MB and the reference MB, respectively.

$$MV\,0 = \frac{MV\,1 + MV\,2 + MV\,3}{3} \tag{2}$$

- *Step* 2: Transform PMV from Step 1 into full-pixel format, set the transformed vector as central point, and calculate *COST* on central point(*MVC*(*xc*, *yc*)) and the 6 vertices of hexagon (solid points in Figure 3). Mark the minimal *COST* as *BCOST_T*, and the vector of *BCOST* as *BMV_T*(*xc*', *yc*'), if *BMV_T* is the current central point skips to Step 4; otherwise next.

$$\begin{aligned}BCOST\_T &= Min(COST\,(xc,yc), COST\,(xc+2,yc),\\ &COST\,(xc-2,yc), COST\,(xc+1,yc+2), COST\,(xc+1,yc-2),\\ &COST\,(xc-1,yc+2), COST\,(xc-1,yc-2))\end{aligned} \tag{3}$$

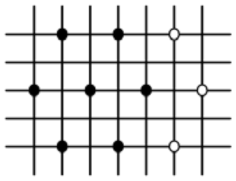


Figure 3. Hexagon search pattern.

- *Step* 3: Set the BMV_T as central point, and calculate *COST* on 3 extra vertices of new hexagon (hollow points in Figure 3). Mark the vector of minimal *COST* as BMV_T, if BMV_T is the current central point skips to step 4; otherwise repeat step 3.

Step 4: Set the BMV_T from above as central point, and calculate *COST* on 8 points around central point (solid points in Figure 4). Mark the minimal *COST* as *BCOST*, and the vector of *BCOST* as BMV, search stops, and BMV is the best vector of estimation.

$$\begin{aligned}&BCOST = Min(COST\,(xc',yc'),COST\,(xc',yc'+1),\\&COST\,(xc',yc'-1),COST\,(xc'+1,yc'),COST\,(xc'-1,yc')\\&COST\,(xc'+1,yc'+1),COST\,(xc'+1,yc'-1),\\&COST\,(xc'-1,yc'+1),COST\,(xc'-1,yc'-1))\end{aligned} \tag{4}$$

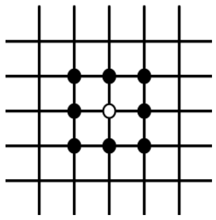


Figure 4. Square search pattern.

2.3. Optimization on HEX

It can be seen through the last section that for the MBs which have violent movement, HEX is able to quickly locate to the surrounding area of best vector with bigger hexagon pattern, then uses square pattern to find the accurate location. But, according to the characteristics of center offset, the probability distribution of motion vectors is decline around the start point [15]. Statistical experiments show that for the video sequences with lower activity there are 90% of MBs whose motion vectors locate in the 3*3 region around the starting point, even for the video sequence which have violent movement, like ‘football’, the probability of motion vectors locate in the 3*3 region is 72%. For those MBs, the calculations of *COST* on the six vertices of hexagon are unnecessary. In order to solve this problem, this paper made the following improvements to HEX, as shown in Figure 5.

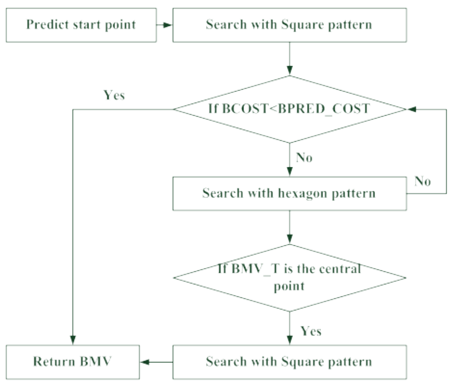

Predict start point
Search with Square pattern
Yes
If BCOST<BPRED_COST
No
Search with hexagon pattern
No
If BMV_T is the central point
Yes
Return BMV
Search with Square pattern

Figure 5. Process of optimized HEX.

Set the transformed PMV as central point, and calculate *COST* on central point and 8 points around it (solid and hollow points in Figure 4). Mark minimal *COST* value as *BCOST*, and the vector of *BCOST* as BMV. If *BCOST* is less than BPRED_COST (the minimal *COST* of prediction), stops search, BMV is the best vector of estimation; otherwise, sets BMV as starting point then executes HEX.

$$\begin{aligned}&BCOST = Min(COST\,(xc,yc),COST\,(xc,yc+1),\\&COST\,(xc,yc-1),COST\,(xc+1,yc),COST\,(xc-1,yc),\\&COST\,(xc-1,yc+1),COST\,(xc-1,yc-1),\\&COST\,(xc+1,yc+1),COST\,(xc+1,yc-1))\end{aligned} \tag{5}$$

For 70% at least of MBs, HEX needs to calculate *COST* on 15 points, and the optimized algorithm only needs 9, the improvement of speed is significant.

3. Design of Multi-Screen Sharing System

The module chart of multi-screen sharing system is shown in Figure 6.

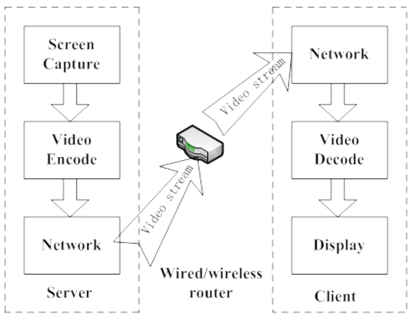


Figure 6. Module chart of the system.

In order to improve the system scalability and portability, server and client has been divided into 3 modules respectively. In server side, screen capture module captures image from the screen of server devices [14], and then video encode module encodes image data into video stream, at last network module packages video stream and sends them to client. network module in client receives packages from server and extracts video stream from packages, then, the video decode module decode video stream into image which finally be displayed on the screen of client by display module.

In the screen capture module, the image data of screen is obtained from the framebuffer device. A framebuffer device is an abstraction for the graphic hardware in Linux system. It represents the frame buffer of some video hardware, and allows application software to access the graphic hardware through a well-defined interface, so that the software doesn't need to know anything about the low-level interface stuff [9]. The framebuffer mechanism is also been supported in Android system. To capture the screen of server, we need to open the device file of framebuffer in read-only mode and read data from the file continuously. The process of screen capture module is shown in Figure 7.

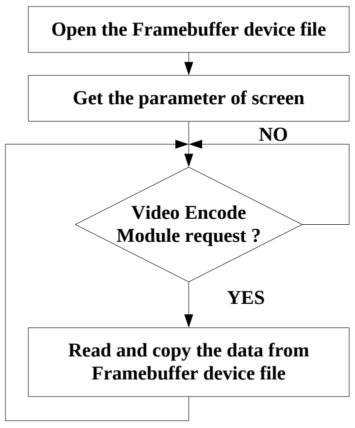

Open the Framebuffer device file
Get the parameter of screen
NO
Video Encode Module request ?
YES
Read and copy the data from Framebuffer device file

Figure 7. Process of screen capture module.

The network module depends on Real-time Transport Protocol (RTP). RTP is a network protocol which published by the IETF working group on multimedia transmission in RFC 1889, it provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services [3, 12, 16]. As shown in Figure 8, data from encoder will be packaged by RTP with information needed by client such as sequence number and timestamp, and then the packages will be transported to client by UDP.

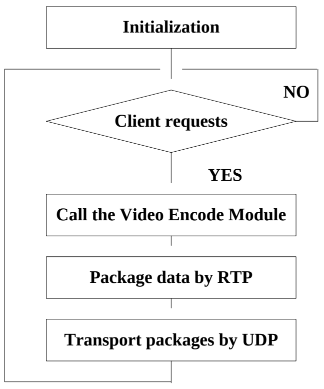


Figure 8. Process of network module.

4. Experiment

This experiment was carried out to test the performance of Optimized HEX algorithm (OHEX) compared with TESA, and HEX. The test sequences of ‘akiyo’ , ‘foreman’ and ‘football’ were been selected, which are shown in Figure 9. The resolution of these test sequences is 176×144 and format is QCIF. The ‘akiyo’ is a video sequences with lower activity, and ‘foreman’ has moderate movement, and ‘football’ is the violent movement video.

a) ‘akiyo’. b) ‘foreman’. c) ‘football’.

Figure 9. test sequences.

The experiment results were shown in Tables 1, 2 and 3. Here, the *PSRN* is the average peak signal to noise ratio. *ΔPSNR* represents the change of *OHEX* algorithm with respect to the *HEX*. *ΔBR* is the percent change of bit rate about *OHEX* algorithm with respect to the *HEX*. The *Δt* is the percent change of motion estimation time for *OHEX* with respect to the *HEX*. These parameters are defined as following:

$$\Delta PSNR = PSNR_{OHEX} - PSNR_{HEX} \tag{6}$$

$$\Delta BR = \frac{BR_{OHEX} - BR_{HEX}}{BR_{HEX}} \times 100\% \tag{7}$$

$$\Delta t = \times \frac{t_{OHEX} \% - t_{HEX}}{t_{HEX}} \quad 100 \tag{8}$$

These sequences have been encoded by X264 encoder with different motion estimation algorithm for 100 times respectively.

The experiment is implemented in a 2.67GHz PC with 2GB memory. The CPU optimizations have been disabled, and the other options have been set to the default value.

Table 1. Contrast of average PSNR (dB) for these algorithms.

Sequence	Frame Number	FS	HEX	OHEX	ΔPSNR
Football	130	36.313	36.308	36.309	0.001
Foreman	150	37.108	37.117	37.080	-0.037
Akiyo	300	38.695	38.727	38.669	-0.058

Table 2. Contrast of bit rate (kbit/s) for these algorithms.

Sequence	Frame Number	FS	HEX	OHEX	ΔBR(%)
Football	130	705.63	716.37	730.395	1.95
Foreman	150	209.65	213.35	215.54	1.02
Akiyo	300	28.74	28.70	28.81	0.38

Table 3. Contrast of motion estimate time (s) for these algorithms.

Sequence	Frame Number	FS	HEX	OHEX	Δt (%)
Football	130	2886.261	271.372	192.595	-29.03
Foreman	150	2152.65	205.894	159.353	-22.60
Akiyo	300	1958.774	228.670	198.810	-13.05

From Tables 1, 2 and 3, the *PSNR* of ‘football’ increases slightly and its *ΔBR* is also increased most which will influence the transmission efficiency. But its percentage of motion estimate time decreases most. For the ‘Foreman’, the motion estimate time decreases about 22.6%, while the change of its *ΔBR* and *PSNR* is relatively small. So the *OHEX* is effective for the moderate movement test sequences. For the ‘akiyo’, the *PSNR* is decreased only 0.058 dB and its bit rate increases 0.38%, but the motion estimate time decreases 13.05%.

Overall, the *PSNR* of *OHEX* algorithm decreased, the average of *ΔPSNR* is about -0.031dB, the change is very small which is almost negligible. The average increase of the bit rate of *OHEX* is about 1.11 %. But the average of motion estimate time decreases about 21.56%. Thus, the speed of *OHEX* algorithm has been improved while the changes of *PSNR* and bit rate are almost negligible. So the *OHEX* algorithm is effective. The proposed method is not only fast but also encodes video sequences at a very high quality.

As described above, the quantitative analysis demonstrated the effectiveness of the *OHEX*. Specially, this algorithm decreased the motion estimate time while the reduction in image quality was almost negligible. In order to further demonstrate the effectiveness, the qualitative analysis was done. Figure 10 showed the contrast of the original image and decoded image. Figures 10-a, d, and g showed the original images which were thirty-third frame of each test sequence. Figures 10-b, e, and h showed the decoded images for the same frame using HEX algorithm and Figures 10-c, f, and i showed the decoded images for the same frame using OHEX algorithm. The contrast results showed that there is almost no difference in image quality, but the speed of the motion estimation increased. So the optimized HEX algorithm is more suitable for the real-time application.

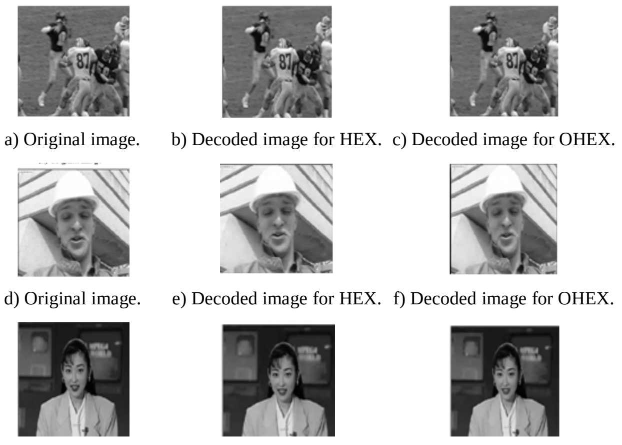

a) Original image. b) Decoded image for HEX. c) Decoded image for OHEX.
d) Original image. e) Decoded image for HEX. f) Decoded image for OHEX.
g) Original image. h) Decoded image for HEX. i) Decoded image for OHEX.

Figure 10. Contrast of original and decoded image.

And more, in order to verify that the improvement of encoder has enhanced the performance of the system, we have tested the delay time of frames comparing with the X264 encoder using original HEX. The delay time is defined as the time interval from the server beginning to capture the screen to the client finally displaying the image to users. In this experiment, the server is based on Android system, with an Nvidia Tegra3, 1.3GHz CPU and 2GB memory; the client using a 2.67GHz CPU and 2GB memory in Linux system; the experiment was taken under a local area network. The screen of server device has the resolution as 1024*600, and the bit rate of encoder has been set as 800kb/s. The experimental scene is shown in Figure 11.

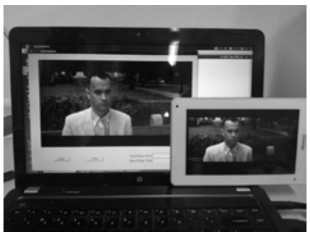


Figure 11. Experimental scene.

In each group, the system has worked for 15 minutes, and during the 15 minutes, the same video has been played in server. The experimental results are shown in Table 4.

Table 4. Test of delay time.

	Maximum Delay Time (ms)	**Average Delay Time (ms)**
Origin	225.40	89.38
Optimized	194.37	82.54
Change	-13.77%	-7.65%

As illustrated in Table 4, the maximum delay time of multi-screen sharing system has been shortened by 13.77%, and the average delay time has been shortened by 6.21%. Even though the delay time is mainly consisted of encoding time and network transmission time, the improvement proposed in this paper has reduced it in a certain extent.

5. Conclusions

In this paper, an optimized motion estimati algorithm of H.264/AVC has been proposed, and the experimental results demonstrate that the new method has saved about 10% of encoding time while PSNR decreasing slightly. Furthermore, a multi-screen sharing system with the improved X264 encoder has been designed and implemented. Experiments show that the delay time of frames has been shortened in a certain extent. This research result can be used for the application development of smart TV, smart phone, tablet PC and other devices, and it will improve the user experience of entertainment.

References

[1] Chazalet A., Bolle S., and Martin S., “Analyzing the Digital Home's Quality of Service,” *in Proceedings of International Conference on Annual Computer Software and Applications Conference*, Germany, pp. 446-451, 2011.

[2] Chien S. and Huang Y., “Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 6, pp. 673-688, 2006.

[3] Diaz C. and Cabrera J., “A Video-Aware FEC-Based Unequal Loss Protection System for Video Streaming Over RTP,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 523-531, 2011.

[4] Duanmu C. "A Fast Hexagon-based Search Algorithm on SIMD Architectures," *in Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems*, Singapore, pp. 1579-1582, 2006.

[5] Jun G., "Home Media Center and Media Clients for Multi-Room Audio and Video Applications," *in Proceedings of 2nd IEEE Consumer Communications and Networking Conference*, Las Vegas, pp. 257-260, 2005.

[6] Khan N., Masud S., and Ahmad A. "A Variable Block Size Motion Estimation algorithm for Real-Time H.264 Video Encoding," *Signal Processing: Image Communication*, vol. 21, no. 4, pp. 306-315, 2006.

[7] Koga T., Iinuma K., Hirano A., Iijima Y., and Ishiguro T., "Motion Compensated Interframe Coding for Video Conferencing," *in Proceedings of NTC Record-National Telecommunications Conference*, New Orleans, pp. G5.3.1-G5.3.5, 1981.

[8] Mittal A., Moorthy A., and Bovik A., "Visually Lossless H.264 Compression of Natural Videos," *Computer Journal*, vol. 56, no. 5, pp. 617-627, 2013.

[9] Schulzrinne H., Casner S., Frederick R., and Jacobson V., *RTP: A Transport Protocol for Real-Time Applications*, Internet Draft, 2003.

[10] Shen L. and Liu Z., Yan T., and Zhang Z., "View-Adaptive Motion Estimation and Disparity Estimation for Low Complexity Multiview Video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 6, pp. 925-930, 2010.

[11] Tai S., Chen Y., Huang Z., and Wang C., "A Multi-pass True Motion Estimation Scheme with Motion Vector Propagation for Frame Rate Up-Conversion Applications," *IEEE/OSA Journal of Display Technology*, vol. 4, no. 2, pp. 188-197, 2008.

[12] Thomas S., Fuente La., Globisch R., Hellge C., and Wiegand T., "Priority-Based Media Delivery Using SVC with RTP and HTTP Streaming," *Multimedia Tools and Applications*, vol. 55, no. 2, pp. 227-246, 2011.

[13] Vun N. and Ooi Y., "Implementation of an Android Phone Based Video Streamer," *in Proceedings of 2010 IEEE/ACM International Conference on Green Computing and Communications*, China, pp. 912-915, 2010.

[14] Wang Q., Chang X., and Feng A., "Implementation and Application of DSP-LCD Driver Based on Framebuffer," *Lecture Notes in Electrical Engineering*, vol. 140, pp. 627-632, 2012.

[15] Wei X. and Jiang J., "A New Adaptive Motion Estimation Algorithm Based on Selecting Predictive Initial Search Point," *Journal of Image and Graphics*, vol. 10, no. 7, pp. 873-877, 2005.

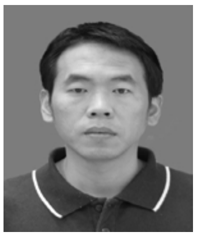
[16] Werda I., Chaouch H., Samet A., Ayed M., and Masmoudi N., "Optimal DSP Based Integer Motion Estimation Implementation for H.264/AVC Baseline Encoder," *The International Arab Journal of Information Technology*, vol. 7, no. 1, pp. 96-104, 2010.

[17] Xu X. and He Y., "Improvements on Fast Motion Estimation Strategy for H.264/AVC," *IEEE Transaction on Circuits and System for Video Technology*, vol. 18, no. 3, pp. 285-293, 2008.

[18] Zhu C., Lin X., and Chau L., "Hexagon-Based Search Pattern for Fast Block Motion Estimation," *IEEE Transactions on Circuits and System for Video Technology*, vol. 12, no. 5, pp. 349-355, 2002.

[19] Zhu S. and Ma K., "A New Diamond Search Algorithm for Fast Block-Matching Motion estimation," *IEEE Transaction on Image Process*ing, vol. 9, no. 2, pp. 287-290, 2000.

[20] Zou B., Shi C., Xu C., and Chen S., "Enhanced Hexagonal-Based Search Using Direction-Oriented Inner Search for Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp. 156-160, 2010.



Shizhe Tan received his PhD degree from Shanghai Jiaotong University, China in 2002. He is currently working a assistant professor in the Department of Electronic Engineering of Ocean University of China. His area of specialization includes video decoding the compression technology, information science technology.

Fengyuan Zhang received his MS degree in Department of Electronic Engineering from Ocean University of China; His research interests include digital home, information technology, and data mining.