The 2008 International Conference on Embedded Software and Systems Symposia (ICESS2008)

Vector Edge Detection in H.264 Implementation

Wei Zhao[§], Zuying Luo[‡], Jeffrey Fan[§], Sheldon X.-D. Tan[†]
[§]Department of Electrical and Computer Engineering, Florida International University, Miami, Florida, USA
[‡]Department of Electronics, Beijing Normal University, Beijing, China *
[†]Department of Electrical Engineering, University of California, Riverside, USA

ABSTRACT

Motion estimation (ME) is one of the bottlenecks in terms of the computational cost in a video encoder system. In this paper, we present a cost-effective method to calculate the "Vector Edge" of the current frame. We store all the vectors' information within a frame and put them in the Laplacian of Gaussian edge detection operator. It is similar to the image edge detection but we need to consider two vectors instead of one, which is different from the luminous edge detection. The experiments show that the proposed method is different from a luminance based graphic edge detection because the edge doesn't go off with high light or dark shader. The edge recognition is better in quality, based on the proposed two separate matrices along with their convolutions by the Laplacian of Gaussian operator.

Keywords

H.264 Architecture, Laplacian Operator, Edge Detection

1. INTRODUCTION

We compress the video data because of the limited storage space and the restricted transmitting bandwidth we are facing today. As everybody knows that our TV standard years ago was NTSC system, also known as 480p (720×480, 30fps) [1]. But now the upcoming new generation of TV is called 1080p (1920×1080, 30fps) which is unbelievable a big progress. The improvement of TV standard is based on the development of storage space and the transmission bandwidth [2]. As you can imagine, the resolution will go higher and higher since the storage and bandwidth is becoming larger and wider. However, the desire of our human being is always there, and basically that is the reason why we are proposing the new generation of video coding standard H.264/AVC [3] nowadays, which is developed by the ITU-T (International Telecommunication Union) and MPEG (ISO/IEC Moving Picture Experts Group), also known as MPEG-4 Part 10. It provides a much efficient way in compressing the video with ratio about 50% data size as it used to be in older standard.

H.264 encoder was built in both hardware and software. However, the ways are being developed for years and now they are quite different. The hardware is good in parallel computation, but weak in control and negotiation. However, a large scale of computation will cause the software to slow down. So the software builders would always avoid big amount of calculation in developing smart and efficient algorithms. They can expand the border of the search window, so H.264 could find the better match in Macro-Block even they don't compute that much. Furthermore, the expansion of the software search window is linearly increasing with time, but in hardware world, it is an area increasing. It will be even exponential increasing if you count the vector estimation part in. In this paper, we try to compromise the tradeoff between software and hardware and achieve much better compression and video quality.

Some companies today use the stored vector of the reference frame to give a simple estimation of the search window of Motion Estimation [4, 5, 6]. They select the search window intuitively. As all the objects in the video are moving correspondingly, the edge of the vector can easily show the edge of an object, if they are moving in certain direction. It won't be hard for us to detect the edge of the moving object by vectors as long as we detect object by luminance in a static picture. A moving object often comes with identical moving values (vectors) within the boundaries of itself. It is similar to the concept of an object filled with single color in the static image. However, we always need to consider about 2 things simultaneously: the length and the direction of the vector. After we obtain this edge of vector, we can manage a lot of things, such as opening a different search window in reference to the current Macro-Block, or in several directions when we have the extra computational power and even the estimation of the object movement for the further compression of the video image.

2. THE ANGLE AND LENGTH APPROXIMATION OF THE MOTION VECTOR

Figure-1 is a simple demonstration of the area ratio of one Macro-Block (MB) and $(3 \times 3)/(9 \times 16)$ MB Search Window in a single 720p frame and 1080p frame. Things have been changed significantly when the TV standard is evolved from Standard Definition (SD) to High Definition (HD), as the resolution becomes higher with the MB size unchanged. It

*This work is supported in part by National High-Tech Research and Development (863) Program of China (2007AA01Z109, 2006AA01Z223)

978-0-7695-3288-2/08 $25.00 © 2008 IEEE
DOI 10.1109/ICESS.Symposia.2008.31

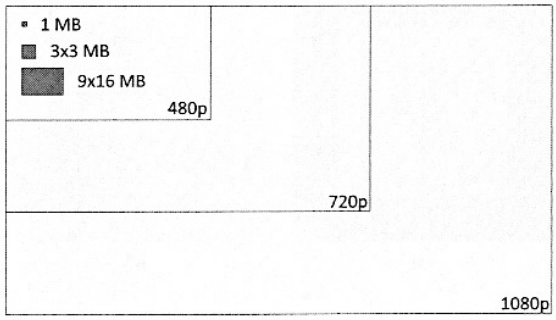

Figure 1: Macro Blocks in 720p and 1080p Frame

means that if we are watching the same frame in the same video, the motion vector goes bigger as the resolution goes higher. In such case, we can only find the right match as we did in the SD video by expanding our search window at the same ratio of HD verse SD. However, expanding the search window will cause a huge hardware cost in all aspects such as the bandwidth, memory and logic gates, etc. Some companies propose to use vector estimation to open the search window which is not fixed alias to the position of current MB, but adjusted by the vectors. The cost is not much and result is quite good. But sometimes it may cause failure because if the vector of reference frame happens to the opposite direction of current frame (e.g. you analyze a border Macro-Block of an object), you might find nothing but big residue value. That is main motivation for us to develop the proposed vector edge detection.

2.1 Motion Vectors

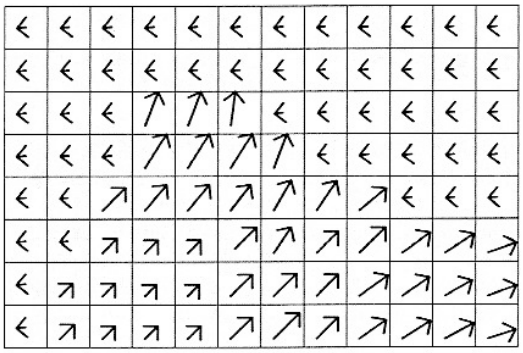

Figure 2: Vectors of Macro-Blocks

Figure-2 gives us a general view of motion vectors in some part of a reference frame. As we can see there are "objects" and "background" in this frame. You can see them clearly because they just have one pure "color". With this objective detection, we can open the search window based on those vectors (or the average of the vectors of the object). But for those blocks on the edge of the objects, we cannot estimate their search windows simply using the average measurement of the object itself, because the Macro-Block of an object can turn out to another object (or background) at any time. By the detection of the edge block, we can either do nothing or estimate the edge driven by previous edge information.

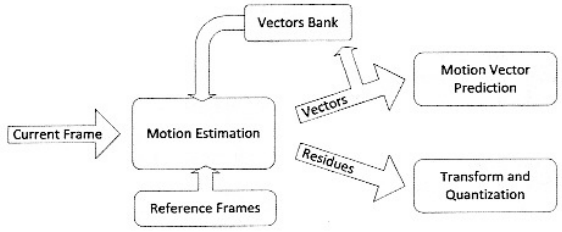

Figure 3: Vector Storage in the ME Sub-Module

Figure-3 shows a preview of vector storage cost: we get the vector data out for the last frame and put them in the "Vector Bank" synchronous to the last reference frame. By the time when the next frame arrives, the vector goes before the reference frames. The Motion Estimation Part will decide how to load the reference data first. The vector storage size doesn't cost much:

$$\text{Bank} = N_{MB} \times (BIT_{Angle} + BIT_{Length}) \quad (1)$$

1. N_{MB} stands for the Number of Vectors, or say, the vector of Macro Blocks in the Current Frame in this paper assumption.
2. BIT_{Angle} is the bits we need for the angle (direction) of the vector approximation (see Figure-4 (a)).
3. BIT_{Length} is the bits we need for the length of the vector approximation (with the unit of Macro Block Size).

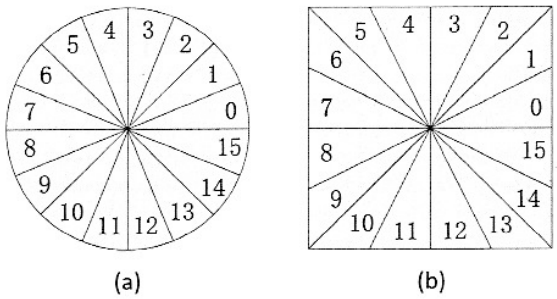

Figure 4: Vector Angle Memory Bits Assumption (a) Without Hardware optimization. (b) With optimization for Hardware builders

As the equation shows, even the 1080p featured H.264 encoder will cost only 11.13KB memory (see eq. 2) of size for

the Vector Bank. It will not impact the critical bandwidth because they are inside the module. However, the turbulent searching window will cause the bandwidth high and low in some way. Therefore we suggest to reserve bigger buffer for this purpose on the chip.

2.2 Directions and Lengths Quantization for Motion Vectors

The original format of a vector value which have been calculated by the Motion Estimation Module is formatted as X by Y pixels. The current MB is from the Matched reference MB. In order to classify a Motion Vector, X and Y is useless, so that we use the quantization to let this vector containing X and Y become the Angel Value and the Length Value of the vector:

1. $Angle = \tan^{-1}(\frac{Y}{X})$ and classified by the Angle Table witch defined by Figure-3.
2. $Length = \sqrt{(\frac{Y}{16})^2 + \frac{X}{16}^2}$

The "tan" or "sqrt" functions are unacceptable complex and consumptive in Hardware design. Thus, we use two assumptions below to make it fit into the framework of hardware implementation:

1. What we need is a 16-divided angle. Dividing it evenly or make an approximation is just the same thing as you can see in Figure-4 (b) and they will be easily calculated in the hardware because of the easier approximation function which is illustrated in Figure-5.
2. $Length = Y/\cos(Angle)$ and if we fixed the Angle, we can just use $\frac{Y}{16}$ to determine the border of two objects who have a same angle of movement.

In the 1^{st} approximation, we divide the circle unevenly. That makes the problem significantly simple. First, we decide which quadrant the vector should belong to by simply using the sign of X and Y (Figure-5 (a)), and then, we compare the X, Y, 2X, 2Y (Figure-5 (b)), followed by getting the angle number from 0000_b (0) to 1111_b (15), which is a 4-digit value. The 2^{nd} approximation is about the length of the vector. We divide them by 16 to see how much Macro-Blocks there are. The 16-dividing mechanism is easy in hardware that we just shift 4 digit out of the number. We need 11 digit to define a number less than 2048 (like 1920) pixels, and for this case, the left will be 7 digit when we just focus on the Macro-block movements. Thus, for a 1080p video Macro-Block, an vector bank memory size should be:

$$\frac{1920}{16} \times \frac{1080}{16} \times (4+7) = 89,100\text{Kbits} = 11.13\text{KBytes} \quad (2)$$

Now, we have generated a matrix for the Macro-Block level Vectors, the original matrix of an $N \times M$ Macro-Block frame

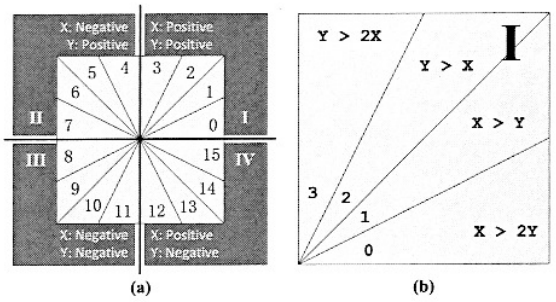

Figure 5: Vector Angle Approximation (a) Full angle quartation by sign of X & Y (b) Angle quartation again in quadrant I by simplified quick algorithm of hardware approximation

becomes:

$$\begin{pmatrix} (X,Y)_{(0,0)} & (X,Y)_{(1,0)} & \cdots & \cdots & (X,Y)_{(M,0)} \\ (X,Y)_{(0,1)} & (X,Y)_{(1,1)} & \cdots & \cdots & (X,Y)_{(M,1)} \\ (X,Y)_{(0,2)} & (X,Y)_{(1,2)} & \cdots & \cdots & (X,Y)_{(M,2)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (X,Y)_{(0,N)} & (X,Y)_{(1,N)} & \cdots & \cdots & (X,Y)_{(M,N)} \end{pmatrix} \quad (3)$$

Using the approximation and quantization above, we can simply drive the (X,Y) matrix into A and R matrix. Note that A is the value of vector Angle and R is the value of vector Length:

$$\begin{pmatrix} A_{(0,0)} & \cdots & A_{(M,0)} \\ A_{(0,1)} & \cdots & A_{(M,1)} \\ A_{(0,2)} & \cdots & A_{(M,2)} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ A_{(0,N)} & \cdots & A_{(M,N)} \end{pmatrix} \& \begin{pmatrix} R_{(0,0)} & \cdots & R_{(M,0)} \\ R_{(0,1)} & \cdots & R_{(M,1)} \\ R_{(0,2)} & \cdots & R_{(M,2)} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ R_{(0,N)} & \cdots & R_{(M,N)} \end{pmatrix} \quad (4)$$

We will discuss these matrices in more details in the next section. Let's denote them A-Matrix and R-Matrix here.

3. EDGE DETECTION FUNCTION

The Edge Detection [7] has been used in Digital Image Processing field. Basically, the edge detection can be classified into the first-derivative and second-derivative edge detect operators. The Roberts, Prewitt and Sobel [7] are 3 popular first-derivative edge detection operators. Laplacian operator is the typical second-derivative operator.

3.1 Laplacian Operator

0	-1	0
-1	4	-1
0	-1	0

-a-

-1	-1	-1
-1	8	-1
-1	-1	-1

-b-

Figure 6: Laplacian Digital Approximations

The Second-Derivative operator can detect the edge horizontally and vertically whereas the first-derivative operator can do either one at a time. The Laplacian of a 2-D function [8] $f(x, y)$ is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{5}$$

From the equation, we can generate 2 digital approximations to the Laplacian which are most useful in practice:

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \tag{6}$$

and

$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9) \tag{7}$$

and finally, we obtain these 2 digital approximations here as shown in Figure-6.

3.2 Laplacian of Gaussian (LoG) Operator

As Laplacian operator may detect edges as well as noise (isolation of out-of-range image), it may be desirable to smooth the image first by doing the convolution with the Gaussian kernel of width σ:

$$G_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{8}$$

So, the Laplacian of Gaussian will be:

$$\nabla^2 G_\sigma = \frac{\partial^2 G_\sigma}{\partial x^2} + \frac{\partial^2 G_\sigma}{\partial y^2} \tag{9}$$

Variable x and y is equal in this equation, we determine the x part first:

$$\frac{\partial^2 G_\sigma}{\partial x^2} = \frac{x^2 - \sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{10}$$

Let $x^2+y^2 = r^2$, and put x, y together back to the equation:

$$\nabla^2 G_\sigma = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{r^2 - 2\sigma^2}{\sigma^4} e^{-\frac{r^2}{2\sigma^2}} \tag{11}$$

Finally, with the selection of σ, we can start our vector edge detection now. Since it is still implemented in digital hardware by finishing the Laplacian of Gaussian Operator construct, we need to find the digital approximation of equation, and it becomes:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

The Laplacian of Gaussian operator is quite an good operator that it is easily built and quickly computable. The vector of the object is very coherent so that the process of vector edge detection won't be as sophisticated as the image edge detection. And the convolution is hard for software but much easy to hardware. All we need to prepare is a group of shift-registers and put the vector table in there.

4. EXPERIMENT RESULT

Figure-7 is two sequential frames of the tailer of movie "Pirates of the Caribbean" [1].

Figure 7: Sample Frame taking by 720p video

If you look into the 480p version, all objects' moving within 1 Macro-Block. But in 720p and 1080p standard, we cannot use 3 × 3 search window since the change of whole video frame is larger than 1 Macro-Block. You need to expand the search window or find the average vector of objects to locate the search window from the prediction of Macro-Block of reference frames.

4.1 Motion Vectors Based on Macro-Blocks

To simplify our calculation, we cropped a block of frame (See Figure-8). There are 2 objects and a blurred background. The man in the front (*Davy Jones*) moves slowly to the right, and with the movie angle changes, the background is moving slowly to the left. The back man (*Captain Jack Sparrow*, on the right shoulder of the front man) is moving very fast.

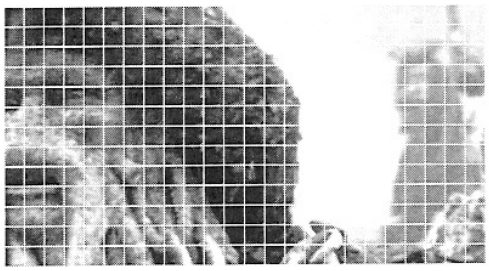

Figure 8: A Part of the Full-720p Frame With Macro-Blocks

[1]http://www.apple.com/trailers/disney/piratesofthecaribbeanatworldsend/

Also, as you can see that the white grid indicates the border of the Macro-Block. Since the Macro-Block becomes very small in 720p resolution standard, the vector is even more smoothly than it was in the low resolution frames. Therefore, it becomes faster for us to recognize those objects by vector based edge detection manner without so much of unwanted noise. With all these observations, we obtain the vector file, thus re-rendering them into our frames in Figure-9.

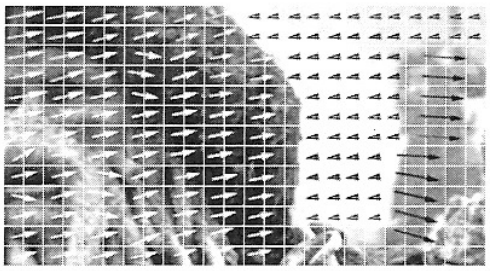

Figure 9: Motion Vectors Re-Rendered

4.2 Motion Vectors Edge Detection Result

As you can see in the figure-10, the result shows that there are three objects and one background in this cropped frame. It is much different from a luminance based graphic edge detection because the edge doesn't go off with high light or dark shader. The detection of edge is meaningful and powerful. The edge recognition is based on 2 separate matrices which are the results from A-Matrix, R-Matrix and their convolutions by the Laplacian of Gaussian operator.

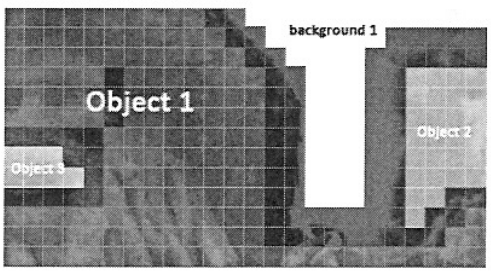

Figure 10: Vector Edge Detection with Laplacian of Gaussian Operator

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we present a cost-effective method to calculate the "Vector Edge" of the current frame. The result is quite good but there are still several parameters that we need adjust. Within different frames, the edge power is not equal, which means that we have to adjust the threshold between the "edges" and "tolerances". There might be static or dynamic adjustment in the future hardware.

As the screen go bigger and bigger, it might be considered that the Macro-Block size should be a little bigger too. But that will cause a huge architecture change whether in software or hardware implementation. However, if the MB size doesn't change, then the equal movement of same video in lower resolution and higher resolution will cause a different compress ratio. In order to enlarge the search window, it may be very costly in the hardware implementation. Even we have a large search window, we can always add some features by the vector recognizing without confirming so many changes in the hardware architecture. On the other hand, as the resolution becomes higher and higher, for same video the higher resolution will have a more smooth vector set. This means that the vector can be detected even faster and easier.

6. REFERENCES

[1] T. C. Chen, C. J. Lian, and L. G. Chen, "Hardware architecture design of an h.264/avc video codec," *IEEE 2006*, August 2006.

[2] I. E. G. Richardson, "H.264 and mpeg-4 video compression," pp. 27–28, August 2003.

[3] J. V. T. of ITU-T, "Draft itu-t recommendation and final draft international standard of joint video specification (itu-t rec. h.264 iso/iec 14496-10 avc)," *Document JVT-GO50*, December 2003.

[4] H. C. Tourapis and A. Tourapis, "Fast motion estimation within the h.264 codec," *Conf. Multimedia and Expo, ICME '03*, vol. 3, pp. III – 517–20, July 2003.

[5] H. Zhihai and S. Mitra, "A unified rate-distortion analysis framework for transform coding," *Circuits and Systems for Video Technology*, vol. 11, December 2001.

[6] C. Chen, S. Chien, Y. Huang, T. Chen, T. Wang, and L. Chen, "Analysis and architecture design of variable block-size motion estimation for h.264/avc," *IEEE Transactions on Circuits and Systems I*, vol. 53, pp. 578–593, March 2006.

[7] R. C. Gonzalez and R. E. Woods, "Digital image processing," vol. 10, no. 2, pp. 585–611, 2001.

[8] H. S. Neoh and A. Hazanchuk, "Adaptive edge detection for real-time video processing using fpgas," vol. 7, no. 3, pp. 2–3, 2004.