Controller Free Gaming and Gesture Recognition Via H.264 SoC

Wei Zhao [§], Xiaolong Yuan [¶], Raul Batista [§], Jeffrey Fan [§]
[§] Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, U.S.A.
[¶] School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China

***Abstract*—Current technologies in video encoding require real time solutions in motion gesture recognition for high definition video. Such needs will be a key determinant in the performance of hands-free video gaming consoles utilizing motion gesture recognition for interactive game play. Software solutions present higher costs due to increased bandwidth and increased processing speeds needed for high integrity video compression. We propose a hardware SoC solution to motion gesture recognition via modifications to motion estimation (ME) and motion vector (MV) modules of H.264 codec to reduce the associated overhead costs of real-time video processing.**

Keywords—H.264; System-on-a-Chip Design; Laplacian Operator; Edge Detection;

I. INTRODUCTION

One relevant phenomenon in today's society is the prevalence of the home video game console. Since the first home video gaming systems of the 1970's like Magnavox's Odyssey and Atari's Pong, gaming systems have evolved drastically from their Neolithic analog technology to today's high performance image recognition and peripherally interactive gaming like the XaviXport [12]. One common variable throughout the evolution of the home video game console has been the two player controller.

Most gaming consoles use a 2 hand controller and may require the use of a joystick to perform certain actions in the gaming environment. Drawbacks to the use of these controllers may be attributed to the learning curve associated with learning the new joystick parameters and limiting the interactivity of the gaming experience. A solution to the removal of the joystick and evolving the interactive gaming experience of home video game consoles was introduced by the Nintendo Corporation in late 2006 with its debut of the Nintendo Wii console.

The Nintendo Wii revolutionized the gaming experience by using a wireless controller with solid state 3 axis accelerometers, and gyroscopes and a direct pointing device resulting in high accuracy reactive and interactive gaming [10]. Tilt, rotation and acceleration in x, y, and z vectors of the controller can be sensed by these onboard control sensors reacting with the gaming console with fast response times. Financially, the Wii gaming console outsold its competitor Sony PlayStation 3 in 2006 [10]. As of March 31, 2008 cumulative sales demonstrate 24.5 million Wii units sold worldwide and 10.6 million units sold in the Americas, resulting in a 73.0% increase in the net sales division of Nintendo over the previous fiscal year [11]. Market trends show the current demand of the Wii console can be attributed to the interactive gaming controller which no other gaming system possesses. The future of interactive gaming would therefore evolve to a new level where the controller is completely removed, letting the gamer have a truly disconnected interactive gaming experience via a console with motion gesture recognizing capability.

Instead of using a joystick that needs to be held by the user, one would rather wave his/her hands or shake his/her body to perform a user input. To achieve this, a video camera needs to be used. Today, high-definition video cameras are more and more common in normal families, as well as available in gaming platforms. Motion detection is also a well studied model. However, nearly all related works on motion detections are software based – even for those based on the hardware, are very expensive and hard to accommodate into gaming platforms. During several previous years, the author has presented several papers based on the techniques of optimizing H.264 and motion detection. In this paper, based on these techniques [3-5, 7], the author suggested a minor modification on H.264 based SoC hardware design to detect motions. With some extra software based analysis, the motion can be well recognized and analyzed.

Motion recognition, in particular gesture recognition in real time video encoding technologies has been a popular topic in the recent years. There are a lot of different algorithms today in used to track moving objects [13-16]. Real time processes will have to require video data compression with cost efficient solutions guaranteeing minimal bandwidth and data storage space. The current H.264/AVC [1] video coding standard developed by the ITU-T (International Telecommunication Union) and MPEG (Motion Picture Experts Group) provides a video compression rate 50% more efficient than the previous standard [2]. Unfortunately as the data compression ratio rises so does the complexity causing a higher complexity in hardware design of an H.264 SoC (System on Chip). According to the instruction profiling with HDTV1024P (2048 × 1024, 30fps) specification, H.264/AVC decoding process requires 83 Giga-Instructions Per Second (GIPS) computation and 70 Giga-Bytes Per Second (GBPS) memory access. In H.264/AVC encoder, up to3600 GIPS and 5570 GBPS are required for HDTV720P (1280 × 720, 30fps) specification [1,2]. Such speeds would require bandwidths that are inefficient and costly, demanding the need for bandwidth reduction with low data integrity loss.

The identification of moving objects may be time-consuming and relates to one of the greater computational costs in video

CHINACOM 2010, August 25-27, Beijing, China

DOI 10.4108/chinacom.2010.57

encoding [3, 4]. Therefore the implementation of an SoC H.264 framework for motion estimation (*ME*) can be used. In this paper H.264 codec will be implemented to show the use of the Vector Bank in Section 2, edge detection of an object with corresponding vector edge in Section 3, and gesture recognition of a moving object via modifications to the *ME* modules of H.264 codec in Section 4.

II. Vector Bank based H.264 SoC design

A. H.264 Soc Design

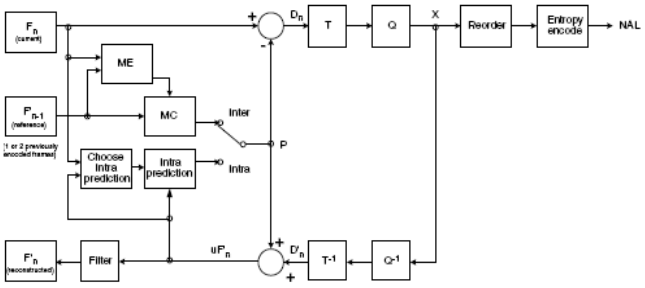

Figure 1 H.264 General Encoder Core [9]

The depiction of the general architecture of H.264 encoder core is expressed in figure 1 [9]. Basically, a moving object is detected by color, and then isolated from the background as does a human eye function. In H.264 *ME* block use well known algorithms to compress video residues by detecting movements of every object in a current frame [10]. Variations in the block size of *ME* algorithm of H.264 allows for the partitioning of small sub-blocks of a 16x16 Macro-Block (*MB*) where are as many as 41 Motion Vectors (*MV)* to be determined in the motion detection process [5,6]. The implementation of H.264 Soc hardware solution is used for the purpose of bandwidth reduction in real time processing. In reference [2] the associated bandwidth costs according to H.264 Encoder Architecture processes were determined. By minimizing the search window in the *ME* algorithm of H.264 and implementing the Bandwidth-Min algorithm, significant bandwidth reduction was experienced [2].

B. Vector Bank

The addition of a Vector Bank on a typical video codec core for use in object motion detection plus boundary detect operator will dramatically save bandwidth, processing load and memory resources [7].

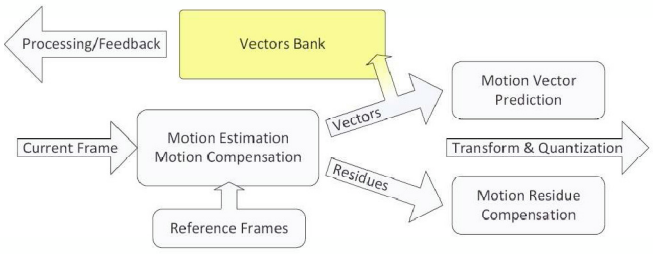

Figure 2 Vector Bank Implementation on H.264

The Vector Bank is performed by attaching a memory based analyzer to the *ME* block of H.264 as shown in figure 2[7]. Bypassing the processing of the output *ME* vectors and residues to be transformed and quantized, the Vector Bank allows for direct grabbing of the block-based motion vectors in a frame based form, allowing for Digital Signal Processing (*DSP*) interfacing. Post processing of H.264 with the inclusion of the Vector Bank data extracted, can then interface with DSP where data can then be analyzed to estimate moving directions, analyze the object, and ultimately determine and define gestures.

III. Edge Detection Operators

Object edge detection has been an important component of Digital Image Processing. Isolating a background image from the object in question can be performed by object edge detection. First derivative operators such as Roberts, Prewitt and Sobel can be used to detect edges of single dimensional images [8]. For two dimensional images second derivative Laplacian operators may be utilized [7]. In reference [7] a Sobel and Laplacian of Gaussian (*LoG*) operator were demonstrated to show their performance in object edge detection.

IV. Gesture recognition via H.264

Gesture movement is not a simple movement. It changes its direction, and simple prediction is not accurate anymore. Sometimes there are rotations. *ME* module of H.264 always tries to find the closest Macro Block (*MB*) in those reference frames. But it is based on luminance difference, doesn't work well with rotations. If some moving object is rotating in the frame, then the correlations of the same object in the reference frame and current frame is dropping. If the background's luminance vector is somehow similar to the object's vector, the *ME* module will use other "closer" *MB* to substitute current *MB* – in terms of luminance difference. It doesn't matter for the video compression but the vector information is not useful anymore in this case.

1. Before taking motion videos, a background image is taken which will be stored for the entire process and continuously compared with each of the frames that was taken instantly.

Figure 3 Background Still Image

2. Follows by taking sequential images which are analyzed and demonstrated in figure 4.

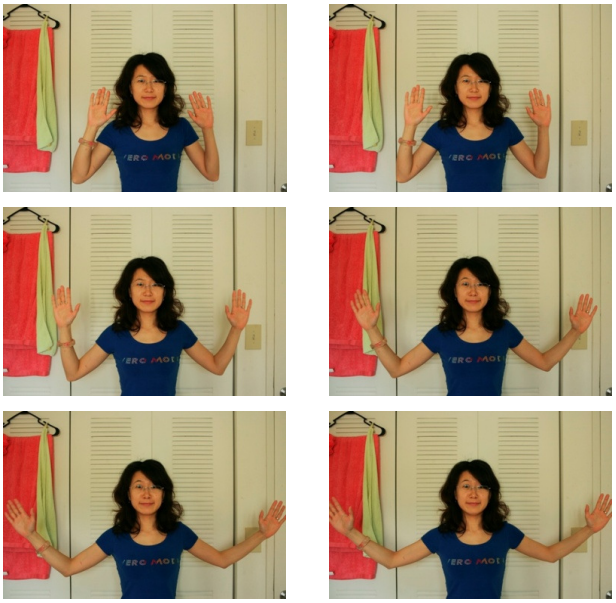

Figure 4 Sequential Images

3. A direct subtraction of the background image is shown in figure 5.

Figure 5 Direct Background Subtraction

As can be seen, the removal of the background cannot be removed directly due to slight changes in lighting ratio when there is an object moving in or out of the screen. Therefore a balance to the light ratio is needed.

4. The actual light ratio of the entire background does not change evenly, yet are similar to the light ratio of the object and is assumed before adjustments to the lighting ratio. Procedures are as follows:

a) Take a small portion (10x10 pixels) but same position image of the current and background image.

b) Do the subtraction and get the matrix result of it.

c) There is going to be two kinds of results as displayed in figure 6.

32	5	102	5	63	178	164	204	101	40
97	183	99	218	133	220	141	20	49	120
36	106	94	16	140	36	97	31	19	152
218	122	85	104	78	228	230	42	207	210
111	39	29	164	28	192	91	196	91	79
196	127	112	157	103	200	220	160	147	143
85	254	99	61	214	137	228	193	61	108
40	31	61	207	214	157	4	123	16	189
238	80	12	96	34	83	190	221	237	144
85	106	133	37	127	142	254	89	210	203

46	45	45	48	46	48	49	47	47	47
47	49	46	48	45	49	48	49	47	47
45	48	49	46	46	48	46	48	47	46
48	50	45	48	45	47	46	49	46	50
46	50	46	49	49	46	46	48	49	47
49	48	49	46	46	48	47	47	46	49
45	50	47	47	45	46	50	50	45	47
50	46	46	47	47	46	47	49	48	45
49	46	47	49	50	49	48	48	47	48
47	46	49	49	49	47	47	46	47	48

Figure 6 Matrix Subtraction Result

d) The upper one means that the block of the image that was taken from the current frame and the background frame is totally different, which means that there is some object coming in. The second one means that the image pattern is the same but only with a lighting ratio change. A standard deviation is performed for the result matrix to determine if the result matrix is upper or lower. If it is the upper one we got from the 10x10 pixels sample image, we need to find another 10x10 pixels sample until we found one block of image that we want. If we got the lower one, we just simply get the average of the difference which is the lighting ratio change.

e) This is performed for every each channel red (*r*), blue (*b*), and green (*g*) of the image. Because if the luminance vector of the image is adjusted, even different color (say blue and red) can have the same luminance values sometime. Why not take the advantage of the information that the image already have? It costs more time because the Matlab being used is software, but are going to fit this algorithm in hardware, hardware can do non-related calculations simultaneously so separated channel calculation won't be a problem. The only cost would be a little bit more adders and multipliers.

f) After obtaining the lighting ratio change, we compensate the lighting ratio change to the current frame and do the subtraction for every channel. But lighting ratio doesn't change smoothly. They will remain in a zone but never get the exact same number change. The *r*, *g* and *b* image difference after compensation is displayed in figure 7.

As shown the foreground is cleaner than the previous result. Observed as well the difference between *r, g*, and *b* channels are different, proving need for three channels.

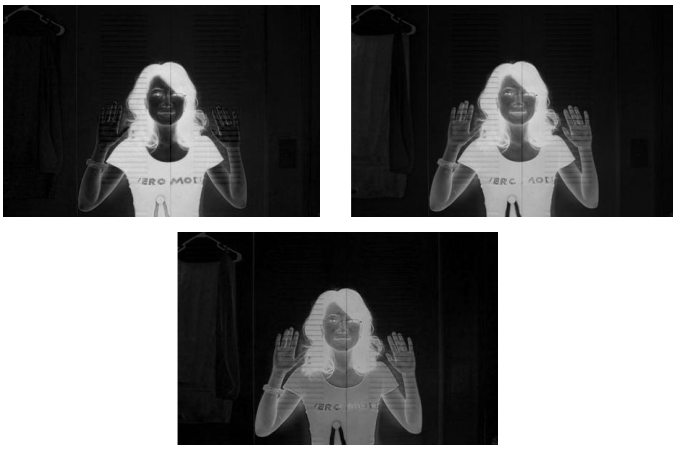

Figure 7 R, G, B Image Result

g) And then we setup a threshold. Threshold is a mask, after the compensated current frame subtracts the background frame, most of the background places are near zero. The big numbers remain (no matter positive or negative) is the object we wanted. We give them a "1". So the threshold mask will be a plane constructed by "0" and "1", means pass or block:

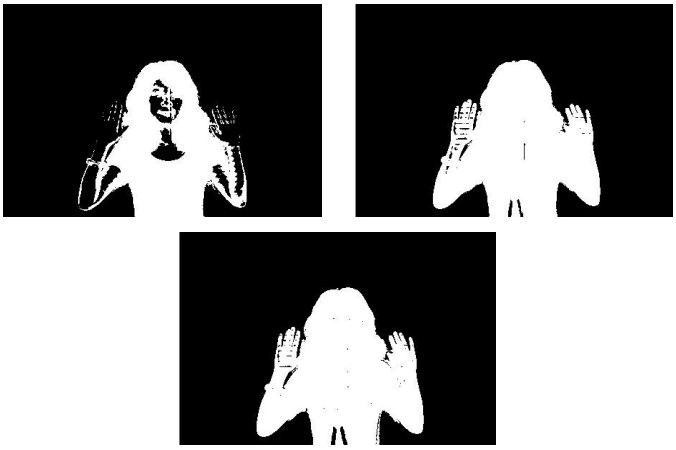

Figure 8 Threshold Mask of R, G, B

h) Then the r, g, and b masks are merged and resulting as shown in figure 9.

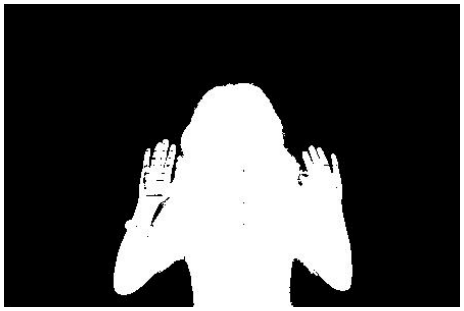

Figure 9 Merging of R, G, B Masks

i) Finally multiplying the current frame to the merged mask resultant, results with the object through the mask yet a blocked background image as shown in figure 9.

Figure 10 Object with Background Removed

As you can see there is still some background within the frame. That is because the distance between the person and the back wall is too short and thus the shadow on the wall changes the light ratio dramatically. But it still can be removed if we adjust our threshold as long as we could keep the most important information inside our result.

Applying the *LoG*, the resultant images are shown with their edges detected as shown in figure 11. Final application of the Motion Estimation algorithm shows movement direction of the motion vectors of the object in question with removal of the background image, ultimately determining gesture recognition as depicted in figure 12.

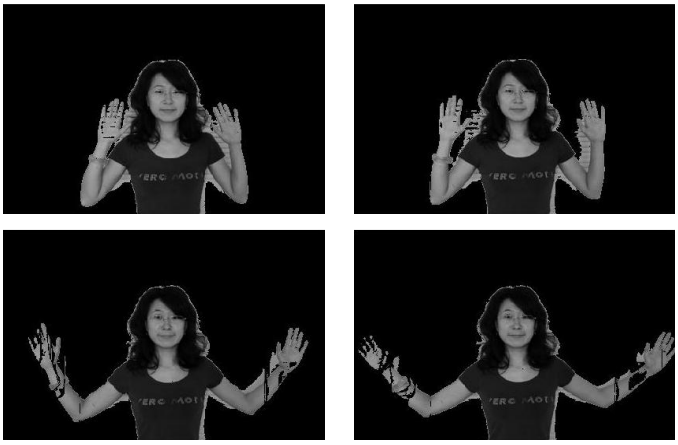

Figure 11 Log Result of Image

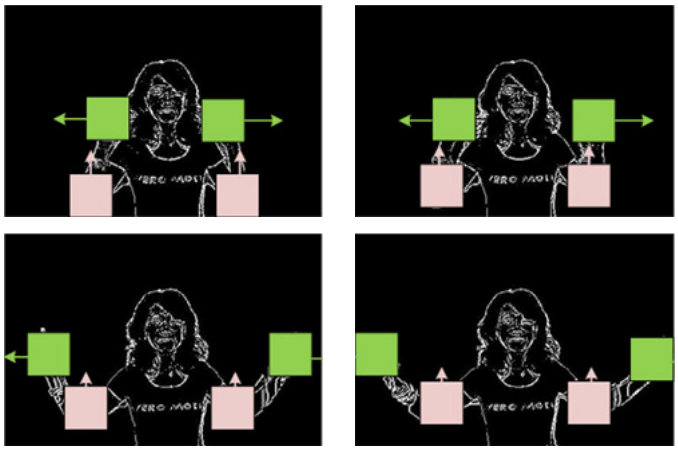

Figure 12 Motion Estimation Results

V. CONCLUSION AND FUTURE WORKS

In this paper the integration of *DSP* with SoC H.264 codec was utilized as an approach to the identification of gesture movements in a moving object. First, bandwidth was reduced by implementing a Bandwidth-min algorithm, and a vector bank was utilized to reduce needed processing time while storing needed motion vectors of previous frames for comparative post processing. *DSP* was then applied to the output vector bank data by means of separating background from the object in reference images. The idea of applying a mask to r, g, and b channels was implemented to separate the image in question from the background image. The LoG algorithm was then applied to the isolated image frames to reveal the edge detection vectors that would result in motion estimation of the final moving vectors of the frame in reference. Future work will require defining the meaning of associated gesture movements. Once the gesture movements are defined they can then be used to command and control a hands free gaming console allowing for completely interactive gaming.

It is simple for hardware to analyze linear vectors, but for more complicated movements like spinning, punching movements, time is needed for developers to link them with all kinds of vector behaviors. Since all kinds of moving vectors are stored in our Vector Bank and hundreds of ways to analyze motion based on 20 years global motion analysis studies, gesture recognition is feasible in the near future.

REFERENCES

[1] J.V.T of ITU-T and I.J.1, “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.

[2] R. Chen, W. Zhao, Q. Liu, Jeffrey Fan, "Efficient H.264 architecture using modular bandwidth estimation", IEEE 5th International Conference on Embedded Software and Systems (ICESS'08), pp. 277-282, Chengdu, China, July 29-31, 2008..

[3] W. Zhao, Z. Luo, Jeffrey Fan, S. Tan, "Vector edge detection in H.264 Implementation", IEEE 5th International Conference on Embedded Software and Systems Symposia (ISHSO'08), pp. 208-212, Chengdu, China, July 29-31, 2008.

[4] W. Zhao, Jeffrey Fan, A. Davari, "Vector bank based target tracking via vision sensors in aviation systems", IEEE 41st Southeastern Symposium on System Theory (SSST'09), pp. 73-76, Tullahoma, TN, March 15-17, 2009.

[5] W. Zhao, C. Castello, Jeffrey Fan, "Design considerations of SOPC-based H.264/AVC systems", 1st International Workshop on Video Coding and Video Processing (VCVP'08), Session S7-3, Shenzhen, China, November 26-28, 2008

[6] S. Yalcin, H.F. Ates, and I. Hamzaoglu, “A high performance hardware architecture for an SAD reuse based hierarchical motion estimation algorithm for H.264 video coding,” IEEE International Conference on Field Programmable Logic and Applications, pp. 509-514, Aug. 2005.

[7] R. Chen, W. Zhao, Jeffrey Fan, A. Davari, "Vector bank based multimedia codec system-on-a-chip (SoC) design", IEEE 10th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN09), pp. 515-520, Kaohsiung, Taiwan, December 14-16, 2009.

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

[9] I. E. G. Richardson, ”H.264 and mpeg-4 video compression”, August2003.

[10] Brain, Marshall. "How the Wii Works." 05 September 2007. HowStuffWorks.com. <http://electronics.howstuffworks.com/wii.htm> 14 January 2010.

[11] Nintendo Annual Financial Report 2008. Nintendo Corporation. Web. Dec. & jan. 2010. <http://www.nintendo.co.jp/ir/pdf/2008/annual0803e.pdf>.

[12] A Brief History of the Home Video Game Console. Web. Dec. & jan. 2010. <http://www.thegameconsole.com/>.

[13] D. Li, ”Moving objects detection by block comparison”, Electronics, Circuits and Systems, vol. 1, pp. 341-344, Dec, 2000.

[14] R. Cucchiara, C. Grana, M. Piccardi and A. Prati, ”Statistic and knowledge-based moving object detection in traffic scenes”, IEEE Proceedings. Intelligent Transportation Systems, pp. 27-32, Oct, 2000.

[15] Y.K. Jung, K.W. Lee and Y.S. Ho, ”Content-based event retrieval using semantic scene interpretation for automated traffic surveillance”, IEEE Transactions on Intelligent Transportation Systems, vol. 2, pp. 151- 163, Sep, 2001.

[16] R. Montoliu and F. Pla, ”Multiple parametric motion model estimation and segmentation”, ICIP 2001, vol. 2, pp. 933-936, Oct, 2001.