Motion estimation algorithm for images of vital signs monitors using Block Matching

Rafael Alvarez-Bossa
System Communitacions & Networking Master Degree s
Universidad Santo Tomás
Bucaramanga, Colombia
ORCID: 0000-0002-1599-9887

Yudy Natalia Flórez-Ordóñez
Faculty of Telecommunications Engineering
Universidad Santo Tomás
Bucaramanga, Colombia
ORCID: 0000-0002-2142-9047

Sergio Andrés Zabala-Vargas
Faculty of Telecommunications Engineering
Universidad Santo Tomás
Bucaramanga, Colombia
ORCID: 0000-0001-5803-1123

Holguer Andrés Becerra-Daza
Faculty of Biomedical Engineering
University of British Columbia
Vancouver, Canada
ORCID: 0000-0002-6079-1006

Tito Raúl Vargas Hernandez
System Communitacions & Networking Master Degree
Universidad Santo Tomás
Bucaramanga, Colombia
ORCID: 0000-0002-8418-2379

***Abstract*—The present article describes the design of an algorithm that allows the motion estimation in images using the Block Matching strategy. The search criteria used to design the algorithm is called the sum of absolute differences (SAD). For research development, a VGA reference image of 640 x 480 pixels, divided into 8x8 macroblocks, is taken. The algorithm has been validated through different simulation scenarios. Among the main results are obtained processing times per frame of 18 ms (without parallelization) and 8.7 ms (with a first level of parallelization); what is useful for video compression systems for transmission. These times were obtained with images of telemedicine tests, particularly in the transmission of video on vital signs monitors.**

Keywords—Block Matching, motion estimation, video compression, telemedicine.

I. INTRODUCCION

Uno de los aspectos fundamentales en la creación de nuevos servicios multimedia es la apropiación de los estándares de codificación, estos permiten reducir considerablemente la cantidad de información además de mantener una calidad de video sobresaliente sin olvidar interoperabilidad entre usuarios de los servicios y desarrolladores de tecnología. Generar tasas de compresión sin sacrificar calidad de video es el objetivo de los estándares de codificación como H.264. Estos estándares son la aplicación de diversas técnicas que se enfocan en reducir la redundancia en la información que existe en los videos.

Explorar nuevas y eficientes estrategias para la transmisión de video es la necesidad de diferentes áreas de trabajo, entre ellos la telemedicina, donde es requerido la transmisión de señales de video (VGA y/o HDMI) a puntos centrales para el análisis de variables e información específica que faciliten el diagnóstico, monitoreo y tratamiento de enfermedades.

En el proceso de codificación de video, la estimación de movimiento es una etapa de gran importancia, esta se encarga de reducir de forma significativa la redundancia temporal que se encuentra en los diferentes frames, explotando la relación que existe entre los pixeles de los cuadros consecutivos de video. Aunque existen múltiples proveedores de codificadores de video, particularmente para la codificación H.264 no se encuentra estandarizada la estimación de movimiento. Esto último ha promovido diversas investigaciones al desarrollo de algoritmos para esta etapa [1].

En la literatura se han agrupado los estimadores de movimiento principalmente en dos categorías, aquellos que trabajan en el dominio de la frecuencia o en el dominio en el tiempo. Organizaciones de estandarización internacionales como ISO o ITU-T han adoptados los algoritmos de estimación de movimiento que pertenecen al dominio en el tiempo, esto se debe principalmente a su eficiencia, permitiendo operaciones simples y directas, ideales para ser implementado en sistemas embebidos con naturaleza combinacional, como es el caso de los FPGA [1].

Si bien se evidencia en la revisión bibliográfica experiencias donde se destacan estimadores que utilizan la técnica de Block Matching para la comprensión de video [2]–[4]; así como investigaciones donde se resalta el uso de sistemas embebidos, para la compresión de video [5]–[7]; el objetivo del presente trabajo es diseñar e implementar un algoritmo de estimación de movimiento en un sistema embebido, donde se pueda adaptar el estándar H.264 empleando técnicas de Block Matching.

II. METODOLOGÍA

El algoritmo de estimación de movimiento Block Matching es un algoritmo que compara directamente dos imágenes temporalmente relacionadas (imagen de referencia e imagen actual), esta comparación inicia cuando se dividen la imagen en grupos de NxM píxeles llamados macrobloques -MB. Para el desarrollo del presente trabajo se toma una imagen de referencia VGA 640 x 480 píxeles y se decide dividir la imagen

978-1-7281-1491-0/19/$31.00 ©2019 IEEE

en macrobloques de 8 x 8 píxeles, logrando obtener en total 4800 macrobloques para la resolución propuesta.

La imagen debe ser barrida por todos los macrobloques buscando la semejanza que existe en la imagen de referencia con la imagen actual, para determinar dicha similitud se emplea el criterio de búsqueda y a partir de éste se determinan los vectores de movimiento. El criterio de búsqueda que se emplea para el diseño del algoritmo propuesto es la suma de diferencias absolutas (SAD) [8], [9]:

$$\boldsymbol{SAD} = \sum_{i=1,j=1}^{N,N} \left| \boldsymbol{C_{i,j}} - \boldsymbol{r_{i,j}} \right|$$

El criterio de búsqueda SAD realiza una suma de los pixeles que se encuentran en cada macrobloque tanto de la imagen de referencia como de la actual, obteniendo un único valor para cada macrobloque en cada imagen. El valor obtenido en el macrobloque de la imagen de referencia es comparado de forma secuencial por cada macrobloque en la imagen actual hasta encontrar que el resultado de la operación sea igual a cero, al encontrar esta condición indicaría que el macrobloque en la imagen actual tuvo un desplazamiento respecto al mismo macrobloque en la imagen de referencia y se procederá a calcular la posición y a almacenar en un vector.

A. Diseño del algoritmo

El algoritmo fue diseñado para realizar la estimación de movimiento empleando el método Block Matching búsqueda exhaustiva. El algoritmo se basa en una máquina de estados que barre la imagen progresivamente, buscando cuando se han presentado cambios entre la posición de los macrobloques, usando como criterio de búsqueda el SAD. A continuación, se describirán el funcionamiento de los estados.

IDLE, es un estado que espera la señal de inicio, al presentarse la señal avanzara de estado.

READ_MEM, este estado es el punto de partida del algoritmo, verifica si la imagen es almacenada en la memoria RAM y barre la imagen hasta el límite de la imagen de referencia que es 4800 macrobloques, este contador volverá a iniciar si sobrepasa el límite de la imagen de referencia.

BUSCAR_SIMILAR, es uno de los estados más relevantes del algoritmo ya que se encarga de definir si un macrobloque en el frame de referencia cambio su posición en el frame actual. Este inicia consultado si el macrobloque de referencia es diferente al macrobloque actual barriendo todo el frame. Para este estado pueden surgir dos posibles decisiones, el primero es si el MB de referencia es diferente al MB actual el algoritmo valida primero que el MB actual se encuentre dentro de la ventana límite, si es así buscara el siguiente estado INCREASE_ACT, esto permitirá aumentar el contador de desplazamiento y seguir barriendo el frame. La segunda decisión es cuando el MB de referencia y el MB actual son iguales, esto quiere decir que encontró el MB en la imagen actual e inicia el proceso de obtener el vector de movimiento. Para esto se activa el siguiente estado SET_REF_BIT_LOAD, donde se aumenta el contador de referencia y el contador actual toma su ultimo valor, volviendo a empezar el proceso de comparación.

SET_REF_BIT_AND_ACT_BIT_1_LOAD, este estado surge cuando el algoritmo consulta la posición del MB de referencia y el MB actual. Si son iguales esto quiere decir que no existe ninguna variación entre el frame de referencia y el frame actual. Los contadores actual y referencia se les aumentara una posición y empieza nuevamente a realizar el mismo procedimiento con los siguientes macrobloques.

GUARDAR_VECTOR_LOAD, es un estado para cuando los macrobloques de referencia y actual son iguales en una posición diferente, automáticamente este estado se activa y almacenara el vector encontrado. Si el contador de referencia es igual o sobrepasa el límite de la imagen, este automáticamente concluirá el proceso de estimación de movimiento.

B. Simulación

La simulación se realizó utilizando la herramienta ModelSim, permitiendo verificar el funcionamiento y comportamiento del algoritmo implementado en verilog. Para la simulación del algoritmo se escogieron dos frames diferentes y consecutivos temporalmente, estos frames provienen de un monitor de signos vitales. La Fig. 1. es un ejemplo de la imagen tomada como frame de referencia.

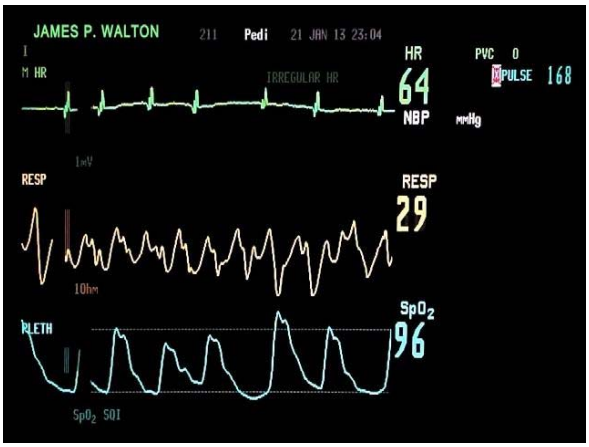

Fig. 1. Frame de referencia

Como frame actual se dispone de la Fig. 2. Los dos frames se analizaron macrobloque por macrobloque para determinar sus vectores de movimiento.

Previo al inicio de la simulación se adicionaron los frames a la memoria RAM creada en el simulador, por lo tanto, las imágenes son previamente convertidas en macrobloques y se les aplica el criterio SAD para posteriormente calcular los vectores de movimiento.

La simulación se realizó en dos escenarios diferentes variando en cada uno de ellos la ventana de búsqueda. El primero es un barrido total de los 4800 macrobloques y en el segundo se realizó la paralelización del hardware para tener 3 ventanas de búsqueda diferentes, esto significa que el barrido

se hace al mismo tiempo 3 veces sobre 1600 macrobloques diferentes como se muestra en la Fig. 3.

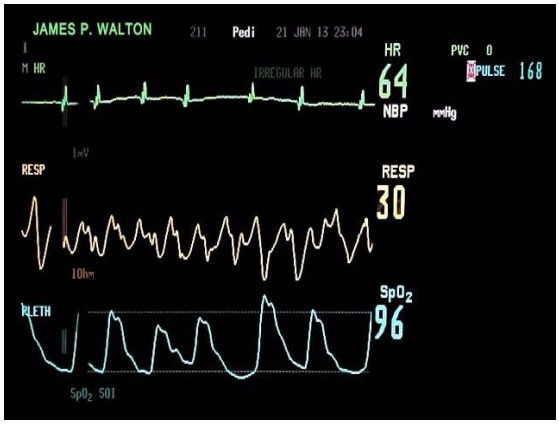

Fig. 2. Frame de actual

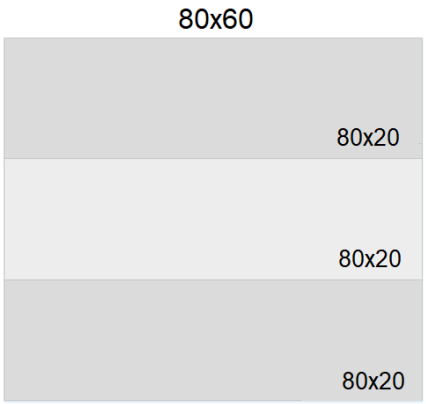

Fig. 3. Ventanas de búsquedas

Los parámetros del primer escenario están representados en la Tabla I. Se configuró, en los parámetros de simulación, una señal de reloj de 50Mhz con el objetivo de conocer el tiempo de procesamiento por macrobloque, y también cuánto tardaría en realizar toda la búsqueda por los 4800 macrobloques y la cantidad de vectores de movimiento encontrados.

TABLA I. PARÁMETROS DEL PRIMER ESCENARIO: BARRIDO TOTAL

Característica	*Valor*
Frecuencia de reloj	50 MHz
Resolución Frame	640 x 480 pixeles
Macrobloque	8 x 8 pixeles
Ventanas de búsqueda	1
Macrobloques por ventana	4800

Para el segundo escenario se plantearon los parámetros presentados en la Tabla II, la diferencia principal respecto al primer escenario es la división de la ventana de búsqueda en tres partes.

TABLA II. PARÁMETROS DEL SEGUNDO ESCENARIO: TRES VENTANAS

Característica	*Valor*
Frecuencia de reloj	50 MHz
Resolución Frame	640 x 480 pixeles
Macrobloque	8 x 8 pixeles
Ventanas de búsqueda	3
Macrobloques por ventana	1600

III. ANÁLISIS DE RESULTADOS Y DISCUSIÓN

Los resultados de la simulación obtenidos empleando el primer escenario se presentan en la Tabla III, el algoritmo encontró entre los dos frames 576 vectores de movimiento en 18 milisegundos, es decir a 3.75 microsegundos por macrobloque en promedio.

TABLA III. RESULTADOS PRIMER ESCENARIO

Característica	*Valor*
Tiempo por macrobloque	3.75 µs
Tiempo total del cálculo	18 ms
Vectores encontrados	576

Para el segundo escenario se presentan resultados en la Tabla IV, se observa que paralelizando las ventanas de búsqueda se reduce considerablemente el tiempo de procesamiento total a 8.4 milisegundos que es el total de tiempo de procesamiento de la ventana de búsqueda que más se demoró. Es importante resaltar que la precisión en este escenario se redujo por las ventanas de búsqueda.

TABLA IV. RESULTADOS SEGUNDO ESCENARIO

Table Head	**Resultados segundo escenario**	
	Característica	***Valor***
1	Tiempo por macrobloque	3.7 µs
2	Tiempo total del cálculo	8.4 ms
3	Vectores encontrados ventana 1	185
4	Tiempo de procesamiento ventana 1	8.4 milisegundos
5	Vectores encontrados ventana 2	124 vectores
6	Tiempo de procesamiento ventana 2	4.7 milisegundos
7	Vectores encontrados ventana 3	170 vectores
8	Tiempo de procesamiento ventana 3	4.9 milisegundos
9	Vectores totales encontrados	479 vectores
10	Tiempo promedio por macrobloque	3.7 microsegundos

Comparando los resultados obtenidos con los propuestos en el trabajo [10], donde los autores emplean el SAD como criterio de búsqueda, tamaños de macro bloque 16 x 16 y una ventana de búsqueda de 32 x 32 se obtienen los siguientes resultados presentados en la Tabla V.

TABLA V. COMPARACIÓN DE RESULTADO DEL PRIMER ESCENARIO DE SIMULACIÓN

Característica	***Propuesta autor*** [10]	***Escenario 1, 1 ventana de búsqueda***	***Escenario 2, paralelización, 3 ventanas de búsqueda***
Frecuencia de reloj	125 MHz	50 MHz	50 MHz
Resolución Frame	640 x 480 pixeles	640 x 480 pixeles	640 x 480 pixeles
Macrobloque	16x16 pixeles	8 x 8 pixeles	8 x 8 pixeles
Ventana de búsqueda	32 x 32	80 x 60	80 x 20
Macro frame	4800	4800	4800
Tiempo promedio por Macrobloque	39.34 microsegundos	3.75 microsegundos	3.7 microsegundos
Macrobloques por segundo	25.419	276.000	270.270
Tiempo de procesamiento por frame	N/A	18 milisegundos	8.7 milisegundos

En [10], [11] se propone paralelizar el cálculo del SAD para ser más eficientes y reducir el coste computacional, en el presente trabajo se propone que se ejecute el algoritmo sin paralelizar y también utilizar esta técnica solamente en la búsqueda exhaustiva con el objetivo de reducir el tiempo usado en el procesamiento. Aunque en la TABLA V se observa que el escenario 1 y el escenario 2 poseen tiempos de procesamiento por macrobloque muy similares, el tiempo total que se emplea en el escenario 2 en procesar un frame es reducido 2/3 del tiempo empleado en el escenario 1. Aunque los autores no plantean los tiempos de procesamiento por frames, si realizaron un cálculo de macrobloques procesados por un segundo.

IV. CONCLUSIONES

El presente trabajo permitió contar con un estimador de movimiento basado en la técnica Block Matching, en el marco de dos escenarios de simulación cuentan con tiempos de procesamiento por frame de 18 ms y 8.4 ms, respectivamente. Estos tiempos son útiles para sistemas de comprensión de video para transmisión de telemedicina porqué está aportando en uno de los procesos de mayor complejidad del codificador H.264.

Si bien en este artículo no se compararon los resultados con otras técnicas de procesamientos, los mismos pueden ser base para futuras investigaciones.

Como trabajo futuro se propone hacer un análisis del algoritmo enfocado en la utilización de los recursos limitados de un sistema embebido, como es el caso de un FPGA's. Esto permitirá determinar las posibilidades de mejora del algoritmo y la integración con diferentes procesos de la codificación o la generación de nuevas aplicaciones que tengan como base el algoritmo desarrollado.

V. REFERENCIAS

[1] A. Mora-Campos, "Estudio de Arquitecturas VLSI de la etapa de predicción de la compensación de movimiento, para compresión de imágenes y video con Algoritmos full-search. Aplicación al estándar H.264/AVC." Universidad Politécnica de Valencia, Valencia, p. 295, 2008.

[2] S. Abdelaal and Y. Yusof, "Adaptation of Motion Estimation Algorithms for Real Time Video Sequences," in *2018 International Conference on Computer and Applications, ICCA 2018*, 2018, pp. 90–94.

[3] L. Trudeau, S. Coulombe, and C. Desrosiers, "Cost-Based Search Ordering for Rate-Constrained Motion Estimation Applied to HEVC," *IEEE Trans. Broadcast.*, vol. 64, no. 4, pp. 922–932, 2018.

[4] Z. Xu, X. Jiang, and X. Ma, "A Full-Reference Video Quality Assessment Algorithm Based on MOVIE Index," in *2018 10th International Conference on Communication Software and Networks, ICCSN 2018*, 2018, pp. 531–536.

[5] M. Praveena, N. Balaji, and C. D. Naidu, "Hardware efficient block matching algorithm based on modified differential evolution optimization for fast motion estimation," *Analog Integr. Circuits Signal Process.*, 2018.

[6] G. Hegde and B. Vijay, "An efficient hardware realization of diamond search algorithm for motion estimation task in video compression applications," in *2017 International Conference on Microelectronic Devices, Circuits and Systems, ICMDCS 2017*, 2017, vol. 2017–Janua, pp. 1–6.

[7] N. N. Shah and U. D. Dalal, "Register array-based sum of absolute difference processor with parallel memory system for fast motion estimation," *IET Comput. Digit. Tech.*, vol. 12, no. 3, pp. 95–104, 2018.

[8] G. Eason, B. Noble, I. Sneddon, and J. Lennard-Jones, "On certain integrals of Lipschitz-Hankel type involving products of bessel functions," *Philos. Trans.*, vol. 247, no. 935, pp. 529–551, 1955.

[9] J. J. Sanabria-Sarmiento and S. Zabala-Vargas, "Metodología para la determinación de usos del suelo mediante procesamiento de imágenes satelitales," *ITECKNE; Vol. 7, Núm. 1*, vol. 7, no. 1, pp. 98–107, 2010.

[10] J. Olivares, I. Benavides, J. Hormigo, J. Villalba, and E. Zapata, "Fast Full-Search Block Matching Algorithm Motion Estimation Alternatives in FPGA," in *2006 International Conference on Field Programmable Logic and Applications*, 2006, pp. 1–4.

[11] P. Muralidhar, C. Rama-Rao, and D. Cyn, "Efficient Architecture for Variable Block Size Motion Estimation in H.264/AVC," *ACEEE Int. J. Signal Image Process.*, vol. 5, no. 1, 2014.