Obtención en tiempo real del vector de posición en secuencias de imágenes utilizando técnicas de visión computacional

Lujan Ramírez, C.[1], Atoche Enseñat, J.[2], Mora Más, F.

Fecha de recepción: 02 de febrero de 2011 – Fecha de aprobación: 01 de julio de 2011

RESUMEN
En este trabajo se presenta un estudio para la obtención en tiempo real del vector de posición de crustáceos con tenazas en secuencias de imágenes aplicando técnicas de visión computacional. Estos vectores se utilizan en estudios de conducta de los crustáceos para su conservación y su explotación. En el trabajo se describe la metodología utilizada para obtener el vector de posición a partir de un algoritmo de segmentación orientado a realizar el procesamiento en tiempo real. Finalmente, se presentan los resultados comparativos mostrando una reducción en más de 4000 veces del tiempo de procesamiento al utilizar el algoritmo propuesto en comparación con métodos tradicionales.

Palabras Clave: visión computacional, crustáceos, vector de posición, procesador digital de señales, tiempo real.

Real-time position vector from image sequences using computer vision techniques

ABSTRACT
This paper presents a study, for real-time obtaining of position vector, from crustacean with chelae, in image sequences, using computer vision techniques. These vectors are used in behavioral studies conducted to understand the behavior of crustacean, both for preservation and exploitation. The paper describes the methodology used to obtain the position vector from a segmentation algorithm designed to perform processing in real time. Finally, we present comparative results showing the reduction of more than 4000 times in processing time when using the proposed algorithm compared with traditional methods.

Keywords: image processing, crustaceans, position vector, digital signal processor, real-time.

---

[1] Profesor del Instituto Tecnológico de Mérida, Departamento de Ingeniería Eléctrica y Electrónica. E-Mail: clujan@itmerida.mx.
[2] Profesor del Instituto Tecnológico de Mérida, Departamento de Ingeniería Eléctrica y Electrónica. jatoche@itmerida.mx
[3] Profesor de la Universidad Politécnica de Valencia, Departamento de Ingeniería Electrónica.
Nota: El período de discusión está abierto hasta el 1º de marzo de 2012. Este artículo de investigación es parte de **Ingeniería–Revista Académica de la Facultad de Ingeniería, Universidad Autónoma de Yucatán**, Vol. 15, No. 2, 2011, ISSN 1665-529X.

1. INTRODUCCIÓN

La necesidad de reducir el tiempo en el procesamiento de imágenes requiere de nuevos métodos para mejorar el tiempo de procesamiento en algoritmos complejos. En esta propuesta, el seguimiento en tiempo real de diferentes especies para el estudio de las conductas de los animales acuáticos, y el poder reducir el tiempo de procesamiento entre una imagen y otra, proporciona la información necesaria para poder realizar la predicción de la trayectoria de escape ante un depredador. Para poder resolver este problema es necesario abarcar el problema desde la captura de la imagen.

La investigación global requiere de una serie de procesamientos, que a lo largo de su desarrollo se han ido evaluando y determinando las mejores opciones para su implementación. En la Figura 1 se muestra el flujo de procesamiento para obtener los vectores de posición de los crustáceos en estudio.

Hoy en día las cámaras fotográficas pueden entregar fotografías en varios formatos, por lo que es necesario realizar un pre-procesamiento a la imagen digital. Existen cámaras fotográficas que pueden entregar imágenes sin ningún pre-procesamiento, esto es, en formato Bayer. Gracias a este formato las imágenes se obtienen directamente del convertidor analógico a digital de los sensores CCD sin ningún pre-procesado, esto evita retrasos en la transmisión de las mismas. En específico, la cámara que se utilizó en este estudio emplea para la transmisión de la imagen el protocolo Camera-Link, el cual únicamente requiere de una etapa de deserielisadores que se encarga de recuperar la imagen, que es transmitida de manera serial, y en otra etapa, registros en paralelo que colocan la imagen en formato Bayer en una memoria, de donde es tomada para los siguientes procesos (Lujan *et al*., 2007). En la búsqueda de la optimización de los recursos se ve la necesidad de evaluar varias plataformas para ver cuál es la que mejor que se adapta a las necesidades de procesamiento. En el caso de las etapas de adquisición (deserialización y almacenamiento en memoria), resta de imágenes (para eliminar el fondo de la imagen) y umbralización (binarización de la imagen), es adecuado elegir un FPGA (Lujan *et al*., 2008) ya que estos algoritmos tienen características altamente paralelizables que permiten explotar al máximo la capacidad del FPGA de realizar arquitecturas específicas con múltiples procesadores simples trabajando en paralelo, con lo cual se puede mejorar mucho el tiempo de procesamiento y dejar las etapas que requieren algoritmos esencialmente secuenciales para ser procesadas en un DSP.

El resultado de restar dos imágenes es muy utilizado cuando se desean estimar trayectorias de objetos (González and Woods, 2002), sin embargo, en este estudio se utilizó para dejar únicamente a los crustaceos en estudio en las secuencias de imágenes, eliminado todo lo demás (el fondo de la misma). Posteriormente se realiza un proceso de umbralización (González and Woods, 2002), que es la operación encargada de binarizar la imagen, el umbral seleccionado fue de 60, es decir los pixeles que fueron menores a 60 eran considerados como pixeles negros y los pixeles iguales o mayores a 60 eran considerados como blancos. La imagen umbralizada es segmentada por área, generando una lista de segmentos con sus datos importantes, área y centro de masa. Esta lista se filtra para discriminar las áreas muy pequeñas que representan ruido en la imagen.

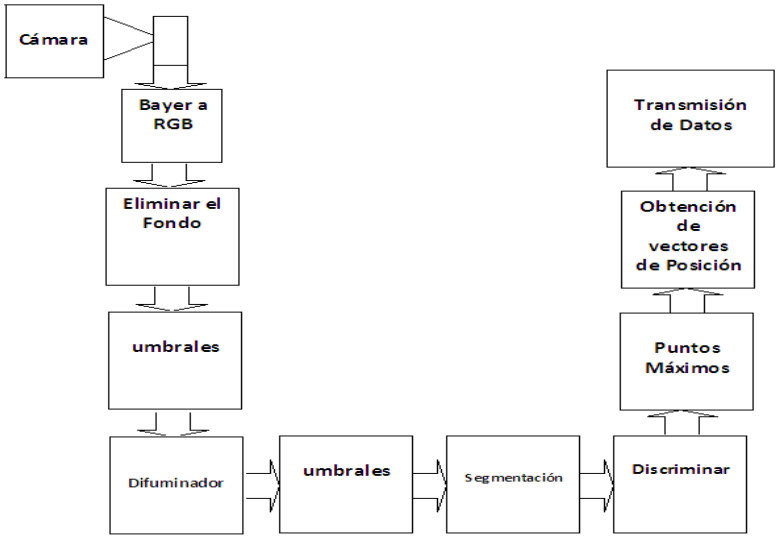

Figura 1. Diagrama a bloques del software requerido para la obtención del vector de posición de los animales.

A partir de la lista discriminada y de la imagen segmentada, se inicia el proceso de extracción de los vectores de posición de los crustáceos en la imagen. En estudios previos sobre el comportamiento de los animales en presencia de depredadores, normalmente presentan conductas bien definidas ya sea cuando el animal es un depredador o cuando es víctima (Arnott *et al*., 1998), (Arnott *et al*., 1999). En base a esto y considerando en específico el caso de peces y crustáceos, se pueden encontrar dos estructuras principales: los animales alargados con el centro de masa ligeramente corrido hacia uno de sus extremos, y los crustáceos con tenazas, aunque existen muchas otras formas que no pueden adaptarse a estas dos, se consideró de gran interés el estudio de estas estructuras debido a la gran cantidad de formas de vida marinas que sí se adaptan a una de estas estructuras.

En ambos casos, los algoritmos se basan en la extracción del o de los puntos más alejados al centro de masa, en el caso de las estructuras alargadas (como un camarón, ver Figura 2), el vector de posición se traza entre el centro de masa y el punto más alejado de este, siendo los crustáceos con tenazas la estructura más compleja de analizar. Por lo tanto, en este artículo se presenta el análisis de la extracción de los vectores de posición para esta estructura específica.

Como ejemplo de crustáceo con tenazas y debido al interés particular por estudiar el comportamiento de camarones en presencia de un depredador natural se decidió utilizar a la Jaiba para este estudio, y así mejorar el desempeño de las granjas de camarones en la península, se utilizará a la Jaiba para este estudio.

Debido a la morfología y comportamiento de estos crustáceos después de evaluar varias opciones se llegó a la conclusión de que la mejor forma para poder obtener su vector de orientación es considerar el hecho de que al atacar, despliegan las quelas, quedando estas como dos apéndices alargados que marcan los puntos más lejanos del crustáceo con respecto a su centro de masa. En la Figura 2, se muestra la posición tomada por la Jaiba en el momento de ataque. Además, se muestran tres puntos que pertenecen a la Jaiba, el que está dentro de ella es el centro de masa y los puntos que están en las quelas son los puntos más lejanos del centro de masa. Con estos puntos y con la aplicación de la geometría analítica se calculará el cuarto punto que está en medio de la recta formada por los dos puntos más lejanos al centro de masa. Este último punto con el centro de masa nos da la recta que representa el vector de posición de la Jaiba.

Como se mencionó, en este artículo se analizará la obtención de puntos máximos y la obtención de los vectores de posición. Por lo que se parte de una imagen segmentada y discriminada.

Se analizarán dos métodos para la obtención de los puntos más alejados del centro de masa, el primero de ellos requiere la extracción de los contornos del segmento a analizar, para lo cual se ha elegido el detector de bordes SUSAN, el cual se presenta en la siguiente sección. El segundo método propuesto, aunque requiere del cumplimiento de ciertos supuestos presenta ventajas significativas en cuanto a velocidad de procesamiento.

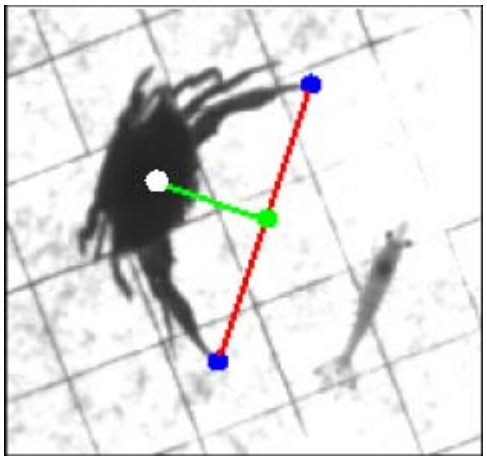

Figura 2. Imagen de la Jaiba cuando se prepara para atacar a un camarón

2. OPERADOR SUSAN

El algoritmo SUSAN (Smallest Inivalued Segment Assimilating Nucleus) para la detección de contornos y esquinas fue desarrollado por S. Smith y J. Brady, (Smith and Brady, 1995). En dicho trabajo se presenta un enfoque para el procesamiento de imágenes a bajo nivel en el que se busca introducir un aspecto clave en el desarrollo de un sistema de visión artificial: la

rapidez.

2.1 Principio SUSAN

El principio de este método consiste en aplicar una máscara circular en cada píxel de la imagen y calcular el número de píxeles dentro de dicha máscara, con un nivel de gris similar al del píxel central. La técnica SUSAN está basada en el hecho de que cada píxel en una imagen tiene un área asociada con un nivel de gris similar al central.

El número de píxeles o área calculada con la máscara contiene información acerca de la estructura de la imagen que rodea al píxel evaluado por la máscara y tiene un valor de 50% en regiones cercanas a un contorno pero es reducida alrededor de un 25% en regiones cercanas a una esquina de la imagen.

Esta propiedad determina la detección de contornos y esquinas en una imagen. Para ejemplificar lo anterior, en la Figura 3 se muestra una imagen en la que se ha resaltado el USAN dentro de cada círculo.

Para segmentar con el algoritmo SUSAN se necesita obtener una imagen de entrada, filtrar con una máscara predefinida y aplicar una serie de reglas con los datos locales para al final, obtener una imagen con los contornos resaltados. Utilizar máscaras circulares asegura una respuesta isotrópica (Figura 4). La más usada contiene 37 píxeles, aunque es posible obtener buenos resultados usando una máscara pequeña de 3 x 3 píxeles.

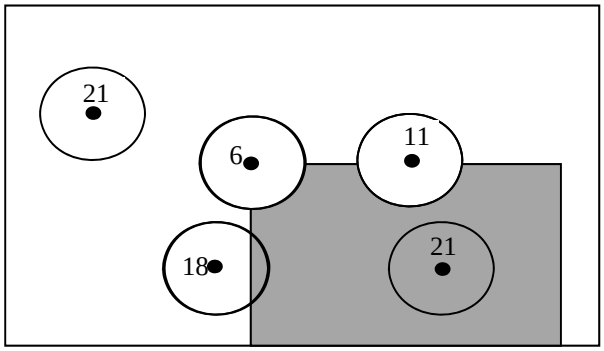

Figura 3. Máscaras circulares localizadas en las diferentes regiones de una imagen. El número representa el área USAN para una máscara circular de 21 píxeles.

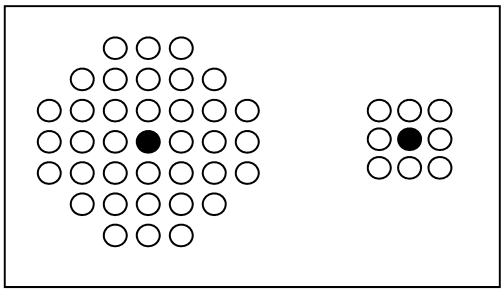

Figura 4. Mascaras circulares de 37 y de 9 píxeles.

Esta segmentación genera resultados muy rápidos en comparación con otros métodos de extracción de contornos, en principio porque solo necesita pasar una vez la máscara sobre la imagen.

3. METODOLOGIA

Analizando la Figura 2, con la ubicación de los tres puntos de interés: el centro de masa y los dos puntos más lejanos; se ve la necesidad de utilizar geometría analítica para obtener la línea que pasa del centro de masa y el punto medio que pasa por la recta que une los puntos más lejanos. La primera ecuación que se necesita es la distancia "*d*" entre los puntos (x_1,y_1) y (x_2,y_2) del plano viene dada por:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (1)$$

Las coordenadas del punto medio del segmento que une dos puntos puede hallarse "promediando" las coordenadas *x* y *y* de los dos puntos, es decir, el punto medio del segmento que une los puntos (x_1, y_1) y (x_2, y_2) del plano es:

$$punto\ medio = (\frac{x_1 + x_2}{2}), (\frac{y_1 + y_2}{2}) \qquad (2)$$

Se sabe que, la pendiente de una recta que pasa por los puntos (x_1,y_1) y (x_2,y_2) está dada por:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{\Delta y}{\Delta x}\ si\ x_1 \neq x_2 \qquad (3)$$

También se conoce que, cualquier recta se puede describir escribiendo su ecuación en la forma pendiente-ordenada:

$$y = mx + b \qquad (4)$$

Con base a lo anterior, se probaron 2 métodos, uno utilizando un filtro de contornos estándar basado en el operador de SUSAN y otro de diseño propio con el fin de investigar cómo se comportan en rapidez, ya que este es un factor muy importante debido a que lograr obtener la información requerida de las imágenes en el menor tiempo posible permite procesarlas en tiempo real sin tener la necesidad de guardarlas, evitando con esto la necesidad de contar con una gran cantidad de espacio de almacenamiento.

Ambos métodos se implementaron bajo el ambiente del CODE COMPOSER STUDIO V3.1 fabricado por Texas Instrument. En este ambiente se programa en lenguaje C o C++ y la implementación fue especificada para el procesador de señal digital (DSP) TMS320C6416. Una de sus características es que maneja imágenes y realiza el procesamiento de datos a 1GHz.

3.1 METODO 1:

Para encontrar los puntos más alejados del centro de masa el primer paso es obtener el borde de la imagen, para esto se eligió el operador de contornos SUSAN porque es un operador muy rápido y preciso comparado con otros métodos, como el operador CANNY (Canny, 1986) que es preciso y nos entrega contornos enlazados, pero su gran defecto es que es muy lento; y los operadores basados en la primera derivada son bastante rápidos pero sacrifican la precisión (Pajares and Cruz, 2008).

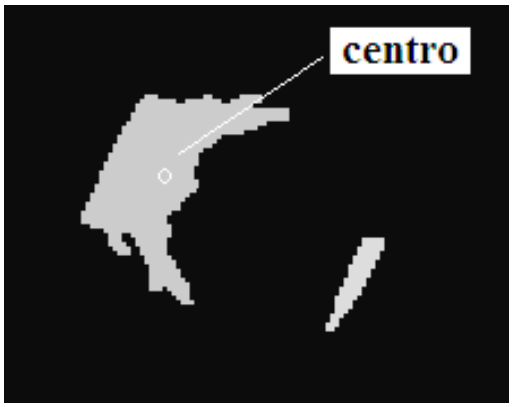

Figura 5. Imagen segmentada en regiones.

Obtención de contornos

Como resultado del análisis del funcionamiento del operador de contornos SUSAN, se diseñó un algoritmo para implementarlo en C, partiendo de una imagen segmentada en regiones y binarizada, como se muestra en la Figura 5.

El sistema, junto con la imagen segmentada, entrega datos adicionales que pueden requerirse en el proceso de

obtención de los puntos máximos. Estos datos son el área y las coordenadas del centro de masa para cada uno de los animales en estudio.

Una vez conociendo de que se dispone y los requerimientos que se necesita para el operador SUSAN se implementó un algoritmo que detectara el borde de la Jaiba.

El resultado obtenido después de ejecutar el algoritmo de SUSAN es una tabla de datos donde están almacenadas las coordenadas del contorno, tomando en cuenta que no son todas ya que con el umbral geométrico se puede discriminar algunas de ellas para no tener todas y solo algunas muestras significativas. Si se representan estos datos en una imagen se obtiene el resultado presentado en la Figura 6.

Teniendo en cuenta los puntos que delimitan el borde de la Jaiba se procede a encontrar los puntos más lejanos al centro de masa que se ha calculado con anterioridad utilizando la fórmula de la distancia entre dos puntos (1).

Esta fórmula se aplica a todos los puntos del contorno de la Jaiba, como se representa en la Figura 7.

Figura 6. Imagen resultado del detector de bordes SUSAN

Figura 7. Distancia de un punto al centro de masa.

Adicionalmente se le agregó una protección para evitar que los puntos estén muy cerca y así evitar falsos positivos en las quelas de la jaiba.

Teniendo ya los dos puntos más lejanos se encuentra el punto medio de la recta imaginaria que se encuentra entre ellos, línea roja que se muestra en la Figura 8.

Las coordenadas del punto medio del segmento que une dos puntos pueden hallarse con (2).

En la Figura 8, se muestra el resultado obtenido del punto medio una *x* de color azul y así como el centro de masa un punto blanco. Al tener el punto medio y las coordenadas del centro de masa, estos conforman parte de una recta imaginaria.

La pendiente que une al centro de masa y el punto medio se puede encontrar utilizando (3).

Al ya conocer la pendiente, y sustituyendo uno de los puntos como valores de *x*, y obtenemos el valor "*b*" correspondiente a (4), procedemos a representarla en el intervalo que limitan nuestros puntos. En la Figura 9 se muestra una recta de color blanco resultante con las coordenadas del centro de masa y el punto medio que es

lo que representa el vector de posición de la jaiba.

En pruebas ejecutadas mediante una simulación con el software Code Composer Studio y utilizando una herramienta del programa mismo se obtuvieron los resultados mostrados en las Tablas 1 y 2.

Teniendo en cuenta que la velocidad del sistema es de 1 GHz lo que es equivalente a mil millones de instrucciones por segundo. Tenemos los siguientes valores pronosticados para las rutinas.

El total de instrucciones implementadas se refiere a las implementadas en el lenguaje ensamblador.

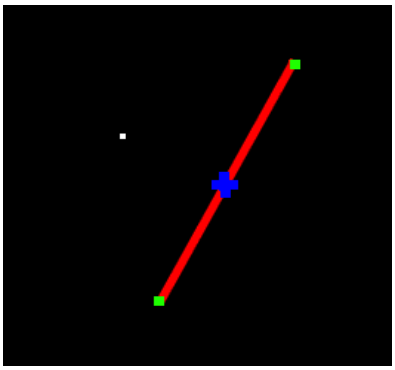

Figura 8. Centro de masa (Blanco), Puntos Máximos (Verde), Segmento (Rojo), Punto medio (azul).

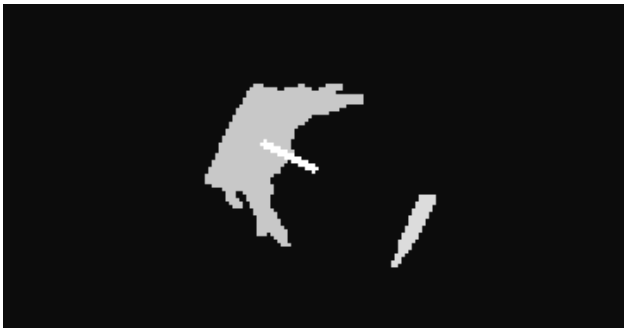

Figura 9. Imagen obtenida en blanco se puede observar el vector de la jaiba.

Tabla 1. Recursos implementados

Rutina	Instrucciones Totales
SUSAN	82,957,800
Grafica del Vector	5,196,609

Tabla 2. Tiempos requeridos para la ejecución del algoritmo

Rutina	Tiempo Estimado
SUSAN	116.48 ms
Grafica del Vector	5.1966 ms

3.2 METODO 2:

Método propuesto para encontrar el vector de posición para la jaiba.

Partiendo como en el método 1 de la imagen segmentada por regiones bien determinadas, con valores diferentes en sus pixeles, así como de los datos obtenidos de la segmentación (el área y las coordenadas del centro de masa), como se muestra en la Figura 5.

El algoritmo propuesto utiliza las coordenadas del centro de masa como punto de inicio, recupera el valor del píxel en este punto y utiliza este valor como semilla ya que es el valor que tienen todos los puntos de la región de interés.

El siguiente paso será encontrar el borde de esta región, debido las características de los elementos que se están segmentado y a las características del segmentador, no existe la posibilidad de encontrar huecos dentro de las regiones, por lo que el algoritmo buscará el borde, realizando una caminata hacia arriba hasta encontrar el cambio de tono, posteriormente realiza un recorrido sobre el borde en sentido anti-horario, comparando la

distancias entre el centro de masa y cada uno de los puntos del borde para obtener la mayor distancia, como se muestra en la Figura 10.

El recorrido para encontrar el punto máximo terminará hasta que esté nuevamente en el punto del borde donde inició. De esa manera se tiene el primer punto más lejano.

Es necesario encontrar el otro punto más lejano. Para evitar que algún punto cercano al más lejano interviniera, se discriminó parte de la imagen, para esto se obtiene el perímetro de la región durante el primer recorrido.

Primero se ubica en el punto más lejano, de este punto comienza el recorrido del borde sin realizar cálculo de distancias, hasta que llega a un cuarto del recorrido, que es donde comienza nuevamente el cálculo de las distancias, para encontrar el segundo punto más lejano y terminará un cuarto antes de que llegue al punto donde inició. Esto es siempre para discriminar cualquier punto cercano al primer punto máximo. Esto se muestra en la Figura 11.

Figura 10. Imagen representativa del recorrido del algoritmo para encontrar el primer punto más lejano.

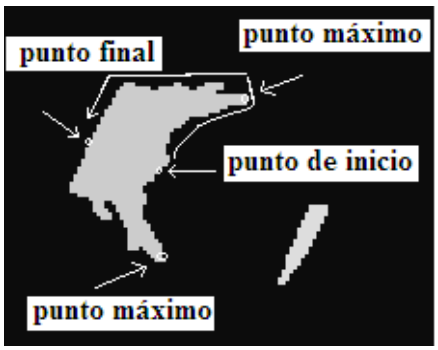

Figura 11. Imagen que representa el recorrido para encontrar el segundo punto más lejano.

Una vez obtenido los dos puntos más lejanos procedemos al cálculo del vector de posición como se expuso anteriormente.

En pruebas ejecutadas mediante una simulación con el software Code Composer Studio y utilizando una herramienta implementada dentro del mismo programa, teniendo en cuenta que la velocidad del sistema es de 1 GHz, se obtuvieron los resultados que se presentan en las Tablas 4 y 5.

Una observación es que de los 200 casos analizados el error de convergencia de los puntos máximos de la jaiba fue máximo dos pixeles de diferencia, llevando a cabo los cálculos del punto medio prácticamente no se cometía error alguno entre los dos métodos, de esta manera se validó el método propuesto, un ejemplo de los datos obtenidos de los puntos más lejanos están representados en las Tablas 5 y 6, en la primera puede verse una columna con el nombre loop_susan que son los puntos obtenidos con el método de susan y en la Tabla 6, otra columna con el nombre de loop_intr7 con los puntos obtenidos con el método propuesto. Una observación importante es que en ambos métodos el centro de masa es el mismo.

Tabla 3. Recursos implementados

Rutina	Instrucciones Totales
Método propuesto	17,407
Grafica del Vector	5,196,609

Tabla 4. Tiempos requeridos para la ejecución del algoritmo

Rutina	Tiempo Medido
Método propuesto	0.0 27 ms
Gráfica del Vector	5.1966 ms

Tabla 5. Puntos encontrados con el método de SUSAN

	loop_susan			
	Puntos Máximos			
Imagen	1	2	Cx	C/I
CCS_foto5	*335, 237*	*312, 280*	*306, 250*	C
CCS_foto13	*334, 236*	*313, 281*	*307, 251*	C
CCS_foto22	*336, 238*	*313, 281*	*307, 252*	C
CCS_foto37	*335, 238*	*316, 283*	*308, 254*	C
CCS_foto42	*335, 241*	*318, 285*	*309, 255*	C
CCS_foto65	*371, 229*	*377, 274*	*356, 253*	C
CCS_foto77	*380, 220*	*390, 269*	*366, 246*	C
CCS_foto95	*382, 219*	*354, 231*	*372, 244*	I
CCS_foto136	*382, 219*	*393, 266*	*372, 244*	C
CCS_foto193	*392, 211*	*400, 261*	*379, 238*	C
CCS_foto199	*549,170*	*578,202*	*566,184*	C

Tabla 6. Puntos encontrados con el método propuesto

	loop_intr7			
	Puntos Máximos			
Imagen	1	2	Cx	C/I
CCS_foto5	*334, 236*	*313, 279*	*306, 250*	C
CCS_foto13	*335, 237*	*314, 280*	*307, 251*	C
CCS_foto22	*335, 237*	*314, 281*	*307, 252*	C
CCS_foto37	*333, 239*	*317, 282*	*308, 254*	C
CCS_foto42	*334, 240*	*319, 284*	*309, 255*	C
CCS_foto65	*370, 228*	*379, 274*	*356, 253*	C
CCS_foto77	*381, 221*	*390, 268*	*366, 246*	C
CCS_foto95	*384, 219*	*393, 266*	*372, 244*	C
CCS_foto136	*384, 219*	*395, 266*	*372, 244*	C
CCS_foto193	*393, 211*	*402, 261*	*379, 238*	C
CCS_foto199	*549,171*	*577,201*	*566,184*	C

En la foto 95 en donde la Jaiba y el camarón estaban montados el algoritmo de SUSAN perdió el punto 2 ya que no correspondía el punto encontrado a la Jaiba sino que le pertenecía al camarón.

4. CONCLUSIÓN

Analizando los datos obtenidos se observa que el algoritmo de SUSAN es muy lento, esto se justifica debido a que al ser un procesamiento lineal de imágenes, hace dos ciclos de procesamiento sobre la imagen y se calculan los promedios que distingue entre sombras, por lo tanto se incrementa el tiempo del procesado significativamente.

Sin embargo, esto no descarta al algoritmo SUSAN para otras posibles aplicaciones en donde el tiempo no sea de vital importancia y la calidad de resultados que se busque sea sumamente eficiente ya que al ser modificable el umbral y el hecho de poder buscar bordes o esquinas deja abierto el algoritmo para un amplio abanico de posibilidades.

El método propuesto tiene la ventaja de ser muy rápido, aprovecha el tener los datos necesarios para poder realizar el recorrido del borde del objeto, sin embargo está limitado a objetos que no tengan huecos dentro del objeto que se va a analizar ya que al momento de realizar la caminata del centro de masa para buscar el borde, en caso de toparse con un hueco recorrería el borde del hueco en lugar del borde exterior del segmento.

De acuerdo con los resultados obtenidos (Tabla 5) el segundo método propuesto se ejecuta más de 4000 veces más rápido en comparación con el tiempo requerido al utilizar métodos convencionales.

Gracias a esta nueva rutina (método 2) se acorta el tiempo de procesado de una imagen lográndose analizar más imágenes dentro de un lapso de tiempo determinado por la transmisión de imágenes, con ello se pueden procesar las imágenes en tiempo real evitando la necesidad de contar con un costoso espacio de almacenamiento y reduciendo la latencia entre la realización del experimento y la obtención de resultados, al evitar la necesidad de un post-procesamiento de los videos de los experimentos.

Tabla 5. Comparación entre los tiempos de ejecución por los dos métodos

Rutina	Tiempo Estimado
SUSAN	116.48 ms
Método propuesto	0.027 ms

REFERENCIAS BIBLIOGRAFICAS

Arnott S.A. Neil D.M. and Ansell A.D (1998). *Tail-Flip mechanism and size-dependent kinematics of escape swimming in the brown shrimp crangon crangon*. The Journal of Experimental Biology 201, pp 1771-1784.

Arnott S.A. Neil D.M. and Ansell A.D (1999). *Escape trajectories of the brown shrimp crangon crangon, and a theoretical consideration of initial escape angles from predators*. The Journal of Experimental Biology 202, pp 193-209.

Canny J. F. (1986). A computational approach to edge detection. *IEEE Transaction on Pattern Análysis and Machine Inteligence*, pp 679-698.

Gonzáles R and Woods R. (2002). *Digital Image Processing*. Second Edition, Prentice Hall. Upper Saddle River, New Jersey.

Gonzalo Pajares Martinsanz, and Jesús M. de la Cruz García. (2008). *Visión por Computador*. Addison-Wesley.

Lujan C, Mora F, Martinez J, (2007). Comparative analysis between the Stratix II (Altera) and Virtex 4 (Xilinx) for implementing a LVDS bus receiver. *ICEEE 2007. 4th International Conference on 5-7 Sept. 2007*, Page(s):373 - 376.

Lujan C. A, Mora F. J, Atoche J. R, (2008). Comparative Analysis in the Implementation of Subtraction and Thresholding for Digital Image Processing. *ICEEE 2008. 5th International Conference on 12-14 Nov. 2008*, Page(s):465 - 469.

Smith S.M and J.M. Brady (1995). SUSAN A New Approach to Low Level Image Processing. *Technical Report, FMRIB.*

Este documento debe citarse como:
Lujan Ramírez, C., Atoche Enseñat, J., Mora Más, F. (2011). **Obtención en tiempo real del vector de posición en secuencias de imágenes utilizando técnicas de visión computacional**. Ingeniería, Revista Académica de la FI-UADY, 15-2, pp 129-139, ISSN: 1665-529-X.