Seguimiento de Objetos en Video usando Contornos Activos y *Bounding Boxes*

Diego R. Park **María J. Gambini** **Marta E. Mejail**

Departamento de Computación, Universidad de Buenos Aires
Intendente Güiraldes 2160, Ciudad Universitaria
Ciudad de Buenos Aires, C1428EGA, Argentina

{dpark, jgambini, marta}@dc.uba.ar

Abstract

The automatic tracking of objects along a sequence of images has applications in different areas as robotics, animation, security systems or medical diagnosis. The tracking algorithm used in this paper starts fitting the contour of an object, using a B-Spline curve as the initial search region. The next step is to sample normal vectors at regularly-spaced points along this curve and to detect points on the border of the object by applying some image-processing filter along the curve normals. A good initial estimate is required for the tracking algorithm to be successful.

This paper presents a method to avoid parametrization errors when fitting the outline of the object at the beginning of the tracking. It has the advantage of being simple and efficient. Conflicts when fitting the contour of the object are avoided using an acceleration structure. The modified algorithm is tested against real videos with excellent results.

Keywords: Image Processing, Active Contours, Bounding Boxes, Video Tracking.

Resumen

El seguimiento de objetos en forma automática a lo largo de una secuencia de imágenes tiene aplicaciones en áreas tan diversas como robótica, animación, sistemas de seguridad o diagnóstico médico. El algoritmo de seguimiento utilizado en este trabajo comienza con la definición de una curva B-Spline que es el área inicial de búsqueda del contorno de un objeto. Luego se consideran una serie de segmentos de rectas normales a esta curva y se aplica algún método de detección de bordes para hallar puntos sobre el contorno a lo largo de las rectas. Para que el algoritmo de seguimiento del objeto sea exitoso es necesario que la estimación inicial sea muy precisa.

En este trabajo se presenta un nuevo método estable y eficiente para evitar errores de parametrización al ajustar el contorno del objeto con una curva B-Spline al comienzo del método de seguimiento. Se utiliza una estructura de aceleración para evitar conflictos al estimar el contorno del objeto. El algoritmo modificado se prueba en videos reales y se observan excelentes resultados.

Palabras clave: Procesamiento de Imágenes, Contornos Activos, *Bounding Boxes*, Seguimiento en Video.

1 INTRODUCCIÓN

El seguimiento de objetos en video es de gran utilidad en numerosas aplicaciones como robótica [16, 14], diagnóstico médico [6, 5, 20], monitoreo de sistemas de seguridad [7, 18], animación [28, 1, 22] y análisis de imágenes de Radar de Apertura Sintética (SAR) [11, 10], entre otros ejemplos.

Los algoritmos de contornos activos y *snakes* son muy utilizados para seguimiento de objetos en video, por su robustez y tratabilidad. El primer algoritmo de detección de bordes para imágenes ópticas, basado en evolución de curvas por medio de la minimización de la energía y cálculo variacional, fue desarrollado por Kass *et al.* [15]. Se trata de curvas que se deforman hasta ajustar el contorno de un objeto de interés.

En este trabajo se utiliza la representación de curvas B-Spline como herramienta fundamental en la descripción de contornos. El contorno formulado por medio de curvas B-Spline, tiene varias ventajas con respecto a otras representaciones de curvas porque permite control local, requiere pocos parámetros y es una función suave. Además de aplicarse al seguimiento de objetos en secuencias de video [3, 2, 17], la representación B-Spline se utiliza también para la aproximación de formas [26, 21, 8] y detección de bordes [4, 13].

Utilizamos aquí el algoritmo de seguimiento en secuencias de imágenes desarrollado por Blake *et al.* [2] que utiliza la representación B-Spline para describir curvas y el espacio de formas. El algoritmo comienza con una curva B-Spline inicial que ajusta al contorno del objeto en el primer cuadro de la secuencia. Para encontrar esta curva es necesario hallar los puntos de borde del objeto. Con este objetivo, se define un área inicial de búsqueda determinada también por una curva B-Spline y se consideran rectas equiespaciadas normales a esta curva. Luego se aplica algún algoritmo de detección de bordes sobre los segmentos de recta. Entonces la imagen se recorre por regiones en lugar de hacerlo sobre toda la imagen, lo que significa un gran ahorro en costo computacional. Luego, la curva inicial se deforma según movimientos permitidos, restringidos al espacio de formas. Sin embargo, una mala elección de normales al comienzo del algoritmo de seguimiento puede significar un ajuste poco adecuado, razón por la cual el algoritmo falla.

En este trabajo, presentamos una nueva solución para evitar los problemas que acarrea una mala elección de segmentos de recta normales a la curva. Se utilizan *Bounding Boxes* para obtener una curva B-Spline de ajuste muy preciso al comienzo del seguimiento.

Las *Bounding Boxes* se han utilizado en muchas aplicaciones, tanto para la detección aproximada de colisiones [24, 27, 29, 25] como estructura de aceleración [19, 9, 12]. Esta representación simplifica la geometría del objeto tratándolo como un rectángulo que lo contiene (de ahí *caja contenedora*).

Este trabajo está organizado de la siguiente manera: la sección 2 está dedicada a la representación de curvas B-Spline, al ajuste de puntos por medio de B-Splines y al espacio de formas. En la sección 3 se explica la aplicación de filtros a lo largo de normales para detectar los puntos de borde de un objeto y se discuten los problemas que surgen de una mala elección de normales. También se presenta el método propuesto que utiliza *Bounding Boxes* para solucionar el problema del cruce de normales, lo que constituye el aporte más importante de este trabajo.. En la sección 4 se muestran los resultados obtenidos. Finalmente, en la sección 5 se presentan las conclusiones.

2 FUNDAMENTOS TEÓRICOS

En esta sección se presenta un resumen de las herramientas que utiliza el algoritmo de seguimiento: la representación B-Spline para curvas, el ajuste de un conjunto discreto de puntos por una curva B-Spline y el espacio de formas. Para más detalles ver [2, 23].

2.1 Curvas B-Splines

Dado los puntos de control $\mathbf{Q}_0, \ldots, \mathbf{Q}_{N_B-1}$ donde $\mathbf{Q}_n = (x_n, y_n)^T \in \Re^2$ con $0 \leq n \leq N_B - 1$ y un conjunto de L nodos $\{s_0 < s_1 < \ldots < s_L\} \subset \Re$, una curva B-Spline de orden d se define como la suma ponderada de N_B funciones polinomiales $\mathbf{B}_{n,d}(s)$ de grado $d-1$ dentro del intervalo $[s_i, s_{i+1}]$, $0 \leq i \leq L-1$. La curva B-Spline se construye como $\mathbf{r}(s) = (\mathbf{x}(s), \mathbf{y}(s))^T$ con $0 \leq s \leq L-1$

$$\mathbf{r}(s) = \sum_{n=0}^{N_B-1} \mathbf{B}_{n,d}(s)\mathbf{Q}_n \tag{1}$$

donde $\mathbf{x}(s) = \mathbf{B}(s)^T\mathbf{Q}^x, \mathbf{y}(s) = \mathbf{B}(s)^T\mathbf{Q}^y$ y los vectores de peso $\mathbf{Q}^x$ y $\mathbf{Q}^y$ son las primeras y segundas componentes de $\mathbf{Q}_n$, respectivamente.

El vector de funciones base $\mathbf{B}(s)$ se define como $\mathbf{B} = (\mathbf{B}_{0,d}(s), \ldots, \mathbf{B}_{N_B-1,d}(s))^T$. Si se denota

$$U(s) = \begin{pmatrix} \mathbf{B}(s)^T & 0 \\ 0 & \mathbf{B}(s)^T \end{pmatrix} \tag{2}$$

entonces $\mathbf{r}(s)$ puede escribirse como $\mathbf{r}(s) = U(s)\mathbf{Q}$.

Dado los puntos $D_1, \ldots, D_N$ del plano de la imagen, donde $\mathbf{D}_i = (x_i, y_i)^T, i = 1, \ldots, N$, se desea encontrar la curva B-Spline definida por los puntos de control $\mathbf{Q}_0, \ldots, \mathbf{Q}_{N_B-1}$ que mejor los ajusta. Entonces, se debe satisfacer

$$x_i = \mathbf{B}(t_i)^T\mathbf{Q}^x, y_i = \mathbf{B}(t_i)^T\mathbf{Q}^y \tag{3}$$

para ciertos valores t_i, donde $i = 1, \ldots, N$ y $N_B \leq N$.

Ahora el problema consiste en hallar los puntos de control de manera que la curva B-Spline pase por todos los puntos $\mathrm{D} = \mathrm{K} \begin{pmatrix} \mathbf{Q}^x & \mathbf{Q}^y \end{pmatrix}$, $\mathrm{K} \in \Re^{N \times N_B}$, donde

$$\mathrm{D} = \begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_N & y_N \end{pmatrix}, \mathrm{K} = \begin{pmatrix} \mathbf{B}(t_1)^T \\ \vdots \\ \mathbf{B}(t_N)^T \end{pmatrix} \tag{4}$$

Definiendo

$$\begin{aligned} t_1 &= 0 \\ t_i &= N_B \frac{\sum_{j=1}^{i} \|D_j - D_{j-1}\|}{\sum_{j=1}^{N} \|D_j - D_{j-1}\|}, 1 < i \leq N \end{aligned} \tag{5}$$

la solución es

$$\begin{pmatrix} \mathbf{Q}^x & \mathbf{Q}^y \end{pmatrix} = \begin{cases} \mathrm{K}^{-1}\mathrm{D} & \textbf{si } N_B = N \\ \mathrm{K}^{+}\mathrm{D} & \textbf{si } N_B < N \end{cases} \tag{6}$$

Si K es cuadrada entonces es inversible. Sin embargo, si $N_B < N$ la matriz no es inversible y por lo tanto se debe calcular la pseudo-inversa.

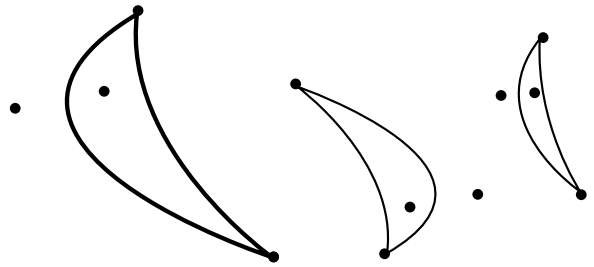


Figura 1: Tranformaciones afines aplicadas a los puntos de control de una curva B-Spline.

2.2 Espacio de Formas

Sea $\mathcal{S} = \mathcal{L}(W, \mathbf{Q}_0)$ el conjunto de las transformaciones y sean $\mathbf{X} \in \Re^{N_\mathbf{X}}$ y $\mathbf{Q} \in \Re^{N_\mathbf{Q}}$ un vector que representa los puntos de control que genera una curva B-Spline. Una transformación $f : \Re^{N_\mathbf{X}} \rightarrow \Re^{N_\mathbf{Q}}$, $f \in \mathcal{S}$ asigna a cada vector $\mathbf{X}$ un vector $\mathbf{Q}$ tal que $f(\mathbf{X}) = \mathbf{Q}$, con $\mathbf{Q} = W\mathbf{X} + \mathbf{Q}_0$ donde $\mathbf{Q}_0$ es la curva patrón o *template* contra la cual se miden las variaciones de las formas y $W \in \Re^{N_\mathbf{Q}} \times \Re^{N_\mathbf{X}}$ es una matriz que corresponde a las transformaciones permitidas de $\mathbf{Q}_0$. Los movimientos permitidos son la traslación, rotación y escalado.

Típicamente ocurre que $N_\mathbf{X} \ll N_\mathbf{Q}$, de manera que se restringe el desplazamiento de los puntos de control a un espacio con menor dimensión. Así, se intenta preservar la forma de la curva, como se ilustra en la Figura 1. En este gráfico se observa el resultado de aplicar las transformaciones permitidas en el espacio de formas a una figura generada por curvas B-Spline (imagen de la izquierda). Al restringir la transformación de la curva a movimientos permitidos en el espacio de formas, se preserva la forma de la curva. Por el contrario, la manipulación arbitraria de los puntos de control no mantiene la forma.

2.2.1 Espacio de las Similitudes Euclideano

Sea $\mathbf{r}_0(s) = U(s)\mathbf{Q}_0$ la curva patrón generada por el vector $\mathbf{Q}_0$. La matriz W que genera el espacio de similitudes euclideanas se define como

$$W = \begin{pmatrix} 1 & 0 & \mathbf{Q}_0^x & -\mathbf{Q}_0^y \\ 0 & 1 & \mathbf{Q}_0^y & \mathbf{Q}_0^x \end{pmatrix} \tag{7}$$

donde la primera y la segunda columna actúan sobre las traslaciones horizontal y vertical, respectivamente. La tercera y cuarta determinan la rotación y la escala a partir del vector $\mathbf{Q}_0$. Es decir, cada columna genera uno de los cuatro movimientos permitidos en este espacio.

Por ejemplo, tomando $\mathbf{X} = (1, 0, 0, 0)^T$ se genera una traslación horizontal sobre la curva $\mathbf{Q}_0$

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & \mathbf{Q}_0^x & -\mathbf{Q}_0^y \\ 0 & 1 & \mathbf{Q}_0^y & \mathbf{Q}_0^x \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{Q}_0^x \\ \mathbf{Q}_0^y \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_0^x + 1 \\ \mathbf{Q}_0^y \end{pmatrix} \tag{8}$$

De la misma manera, $\mathbf{X} = (0, y, 0, 0)^T$, $\mathbf{X} = (0, 0, f_1(\theta), f_2(\theta))^T$ y $\mathbf{X} = (0, 0, z, 0)^T$ determinan una traslación vertical, rotación y escalado isotrópico, respectivamente.

2.2.2 Espacio de Formas Afín Planar

Dadas una curva $\mathbf{r}_0(s)$, una matriz de rotación y escalado $M \in \Re^{2\times 2}$ y un vector de traslación $u = (u_1, u_2)^T$, una transformación afín de $\mathbf{r}_0(s)$ está dada por $\mathbf{r}(s) = u + M\mathbf{r}_0(s)$. El vector u y las coordenadas de la matriz M representan los seis grados de libertad del espacio afín. La matriz de formas W se define como

$$W = \begin{pmatrix} 1 & 0 & \mathbf{Q}_0^x & 0 & 0 & \mathbf{Q}_0^y \\ 0 & 1 & 0 & \mathbf{Q}_0^y & \mathbf{Q}_0^x & 0 \end{pmatrix} \tag{9}$$

Las primeras dos columnas de W representan las traslaciones horizontal y vertical, respectivamente, mientras que las restantes determinan los movimientos de rotación y escalado vertical, horizontal y diagonal. Como $\mathbf{Q} = W\mathbf{X} + \mathbf{Q}_0$, los elementos de $\mathbf{X}$ actúan como pesos en las columnas de W y están dados por $\mathbf{X} = (u_1, u_2, M_{11} - 1, M_{22} - 1, M_{21}, M_{12})^T$. Para más detalles sobre el espacio de formas ver [2].

3 DETECCIÓN DEL CONTORNO DE UN OBJETO

El algoritmo de seguimiento comienza definiendo un área inicial de búsqueda determinada por una curva B-Spline. Luego se plantea una serie de segmentos de recta normales a esta curva y se hallan los puntos de borde del objeto sobre estos segmentos. Finalmente se interpolan estos puntos por una curva B-Spline. A partir de esta estimación inicial, se sigue al objeto utilizando como curva inicial en una imagen, la curva encontrada en el cuadro anterior. Por esa razón es muy importante que el ajuste en el primer cuadro sea muy preciso. El método trabaja restringiendo la transformación de una curva B-Spline a movimientos permitidos dentro del espacio de formas. De esta manera, se preserva la forma de la curva y es un gran ahorro en costo computacional, pero no es posible utilizarlo cuando los movimientos del objeto de interés no son planares. La representación B-Spline tiene la ventaja de que las transformaciones se aplican a los puntos de control en lugar de hacerlo en toda la imagen, lo cual sería excesivamente costoso. El problema ocurre cuando los segmentos de rectas normales a la curva inicial se entrecruzan, provocando que la curva semilla esté muy lejos de la solución óptima. En esos casos el método falla.

3.1 Estimación del Contorno de un Objeto

Para hallar los puntos de borde, se define una curva B-Spline inicial de búsqueda y un conjunto de segmentos normales a ésta. Luego se hallan los puntos de borde que intersecan a estas rectas, como se ve en la Figura 2. Si un punto pertenece al borde del objeto entonces una muestra tomada en una vecindad de ese punto exhibe una discontinuidad en el valor de los niveles de gris y por lo tanto, es un punto de transición. Luego, se encuentra el punto de borde sobre la recta convolucionando los niveles de gris tomados de la imagen con el operador $[-2, -1, 0, 1, 2]$.

Una mala elección de normales puede implicar que los puntos de borde queden invertidos como consecuencia del cruce de normales. Esto introduce errores en la parametrización de la curva, como se ilustra en la Figura 3. Para seleccionar normales no conflictivas, se puede intentar reordenar los puntos de borde. Sea $\overline{\mathbf{R}} = \{\overline{\mathbf{r}}(s_0), \ldots, \overline{\mathbf{r}}(s_N)\}$ un conjunto de puntos de muestra sobre la curva de ajuste. Se aplica el filtro de detección de bordes para encontrar el conjunto de puntos de borde $\mathbf{R}_f = \{\mathbf{r}_f(s_0), \ldots, \mathbf{r}_f(s_N)\}$ sobre la curva característica. Se desea encontrar una función biyectiva $g : \overline{\mathbf{R}} \rightarrow \mathbf{R}_f$, definida por $g(\overline{\mathbf{r}}(s_i)) = \mathbf{r}_f(s_j), 1 \leq i, j \leq N$ tal que no haya cruce de normales, como se muestra en la Figura 4(b). Sin embargo, no está garantizado de que exista g y además, este mecanismo es computacionalmente muy costoso.

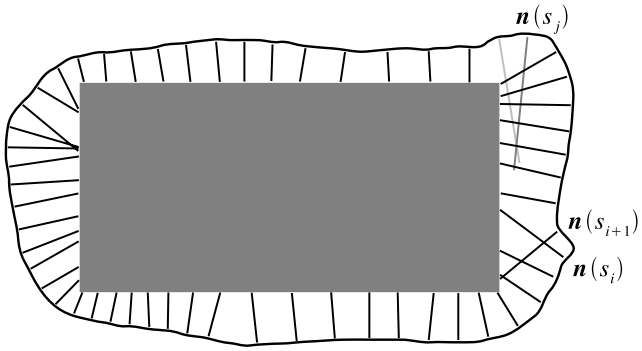
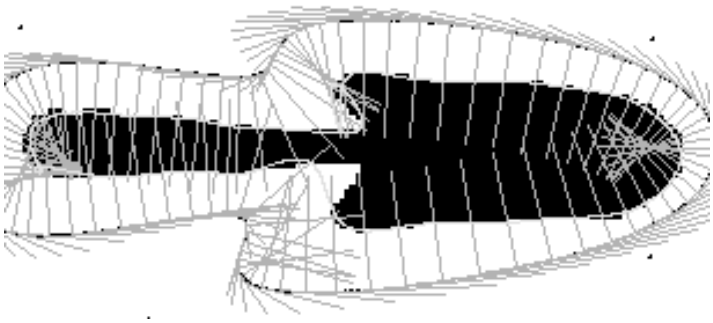
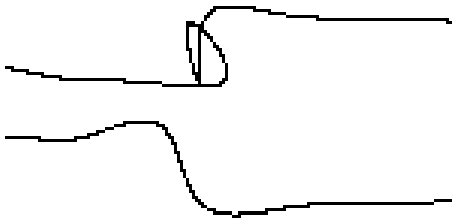


Figura 2: Detección de bordes. A partir de una estimación inicial del contorno del objeto, se trazan rectas normales para encontrar los puntos de borde. $\mathbf{n}(s_i)$ es la normal en el punto s_i.



(a) Generación de rectas normales a la curva inicial.



(b) La curva resultante no ajusta el contorno del objeto como consecuencia del cruce de rectas normales.

Figura 3: El cruce de rectas normales provoca errores en el ajuste del contorno del objeto.

Otra forma de evitar el problema del cruce de normales consiste en analizar secuencialmente el ángulo formado por tres puntos de borde consecutivos, $\mathbf{r}_f(s_i)$, $\mathbf{r}_f(s_{i+1})$ y $\mathbf{r}_f(s_{i+2})$. Se descarta $\mathbf{n}(s_{i+1})$ y se elimina el primer cruce si $ang(\mathbf{r}_f(s_i), \mathbf{r}_f(s_{i+1}), \mathbf{r}_f(s_{i+2})) < 45°$. De esta manera, este mecanismo permite ignorar las normales conflictivas. La Figura 4(c) ilustra esta alternativa.

En este trabajo se propone el método de las *Bounding Boxes* (BB) para solucionar este problema.

Finalmente, una vez hallados los puntos de borde correctos, se construye una curva B-Spline que los ajusta, como se explica en la Sección 2.1.

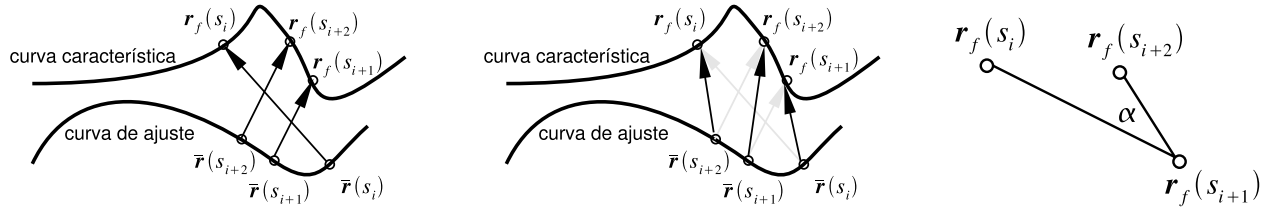
3.2 *Bounding Boxes*

La propuesta consiste en recorrer secuencialmente las normales de a pares. Para cada segmento normal se construye una BB y se realizan cuatro comparaciones. Sea $\mathbf{C}_0^f$ la BB generada por los puntos $\mathbf{C}_0$, $\mathbf{C}_f$ de la siguiente manera

$$\mathbf{C}_0^f = \{P \in I \mid (\text{mín}(\mathbf{C}_0^x, \mathbf{C}_f^x) \leq P^x \leq \text{máx}(\mathbf{C}_0^x, \mathbf{C}_f^x)) \wedge (\text{mín}(\mathbf{C}_0^y, \mathbf{C}_f^y) \leq P^y \leq \text{máx}(\mathbf{C}_0^y, \mathbf{C}_f^y))\} \tag{10}$$

donde I es la imagen. La Figura 5(a) muestra una BB determinada por $\mathbf{C}_0$, $\mathbf{C}_f$.

Sea $\overline{\mathbf{r}}(s)$ la estimación inicial de la forma del objeto en la imagen, $\mathbf{n}(s_i)$ y $\mathbf{n}(s_{i+1})$ dos normales de $\overline{\mathbf{r}}(s)$ muestreadas consecutivamente y P el punto de intersección entre ambas. Para cada i se construye

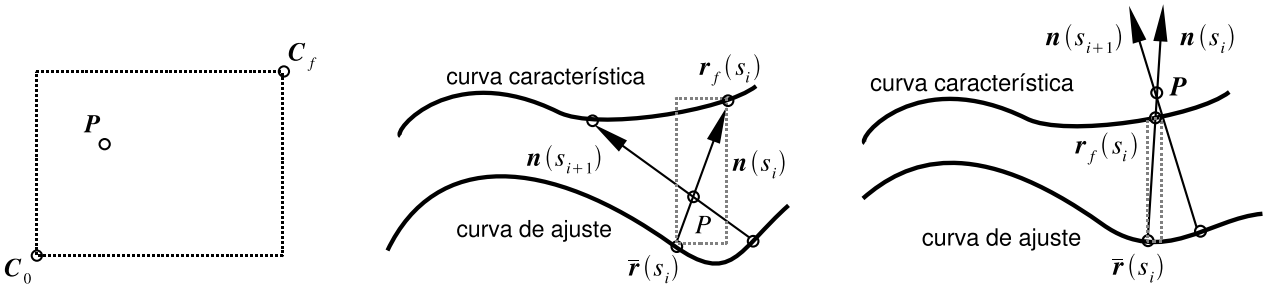


(a) El cruce de normales provoca que puntos de borde queden invertidos.

(b) Función de permutación g tal que $g(\overline{\mathbf{r}}(s_i)) = \mathbf{r}_f(s_{i+1}), g(\overline{\mathbf{r}}(s_{i+1})) = \mathbf{r}_f(s_{i+2}), g(\overline{\mathbf{r}}(s_{i+2})) = \mathbf{r}_f(s_i)$

(c) Ángulo formado por tres puntos de borde consecutivos.

Figura 4: Posibles soluciones al problema del cruce de normales.



(a) Caja Contenedora. (b) $P \in \mathbf{C}^{\mathbf{r}_{f_i}}_{\overline{\mathbf{r}}_i}$. (c) $P \notin \mathbf{C}^{\mathbf{r}_{f_i}}_{\overline{\mathbf{r}}_i}$.

Figura 5: Cajas contenedoras como solución al problema del cruce de normales.

$\mathbf{C}^{\mathbf{r}_{f_i}}_{\overline{\mathbf{r}}_i}$ definida por $\overline{\mathbf{r}}(s_i)$, $\mathbf{r}_f(s_i)$. Si las normales se intersecan antes de alcanzar el borde del objeto, se cumple que $P \in \mathbf{C}^{\mathbf{r}_{f_i}}_{\overline{\mathbf{r}}_i}$, como muestra la Figura 5(b). Entonces se descartan $\mathbf{n}(s_{i+1})$ y $\mathbf{r}_f(s_{i+1})$. En cambio, se considera $\mathbf{r}_f(s_{i+1})$ cuando $P \notin \mathbf{C}^{\mathbf{r}_{f_i}}_{\overline{\mathbf{r}}_i}$, como se ilustra en Figura 5(c).

Una vez seleccionados los puntos de borde, se construye la curva B-Spline que los ajusta.

El Algoritmo 1 muestra el resumen de este proceso de selección de puntos de borde.

Algoritmo 1 Algoritmo de estimación inicial del contorno de un objeto

1: Determinar una región de interés definida por una curva B-Spline.
2: Determinar una serie de segmentos equiespaciados sobre la curva B-Spline.
3: **for all** segmento $s^{(i)}$ **do**
4: Hallar la posición sobre el segmento $s^{(i)}$ donde se encuentra la máxima discontinuidad entre los valores de $s^{(i)}$ convolucionando con el operador $[-2, -1, 0, 1, 2]$.
5: Calcular el punto P de intersección entre el segmento $s^{(i)}$ y el último segmento aceptado S.
6: Construir la caja contenedora C determinada por S.
7: **if** $P \notin C$ **then**
8: $s^{(i)}$ y el punto encontrado se marcan como aceptados.
9: **end if**
10: **end for**
11: Construir la curva B-Spline que interpola los puntos encontrados.

4 RESULTADOS

En esta sección se presentan los resultados obtenidos al aplicar el algoritmo con la modificación del método de las *Bounding Boxes*. La Figura 6 muestra una secuencia de cuadros de un seguimiento sin evitar el cruce de normales. Se observa que las curvas de ajuste en cada cuadro no son aceptables, razón por la cual el algoritmo de seguimiento falla.

En la Figura 7 se observa que las curvas B-Spline de ajuste en cada cuadro son apropiadas y el algoritmo funciona correctamente obteniendo el contorno del objeto en cada cuadro.

5 CONCLUSIONES

En este trabajo se presenta un nuevo método para evitar errores de parametrización debido al cruce de normales. El método de las *Bounding Boxes* permite obtener una curva B-Spline de ajuste muy preciso. Esto es muy importante para inicializar los algoritmos de seguimiento de un objeto en video.

La solución presentada en este trabajo es estable y eficiente ya que requiere cuatro comparaciones para determinar si la intersección entre dos normales consecutivas está dentro del rectángulo generado por una recta. La aplicación del método propuesto presenta resultados visuales muy satisfactorios con un aceptable costo computacional.

REFERENCIAS

[1] A. Agarwala, A. Hertzmann, D. H. Salesin, and M. S. Steven. Keyframe-based tracking for rotoscoping and animation. *International Conference on Computer Graphics and Interactive Techniques. ACM SIGGRAPH 2004*, 2004.

[2] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, 1998.

[3] P. Brigger, J. Hoeg, and M. Unser. B-Spline snakes: A flexible tool for parametric contour detection. *IEEE Trans. on Image Processing*, 9(9):1484–1496, September 2000.

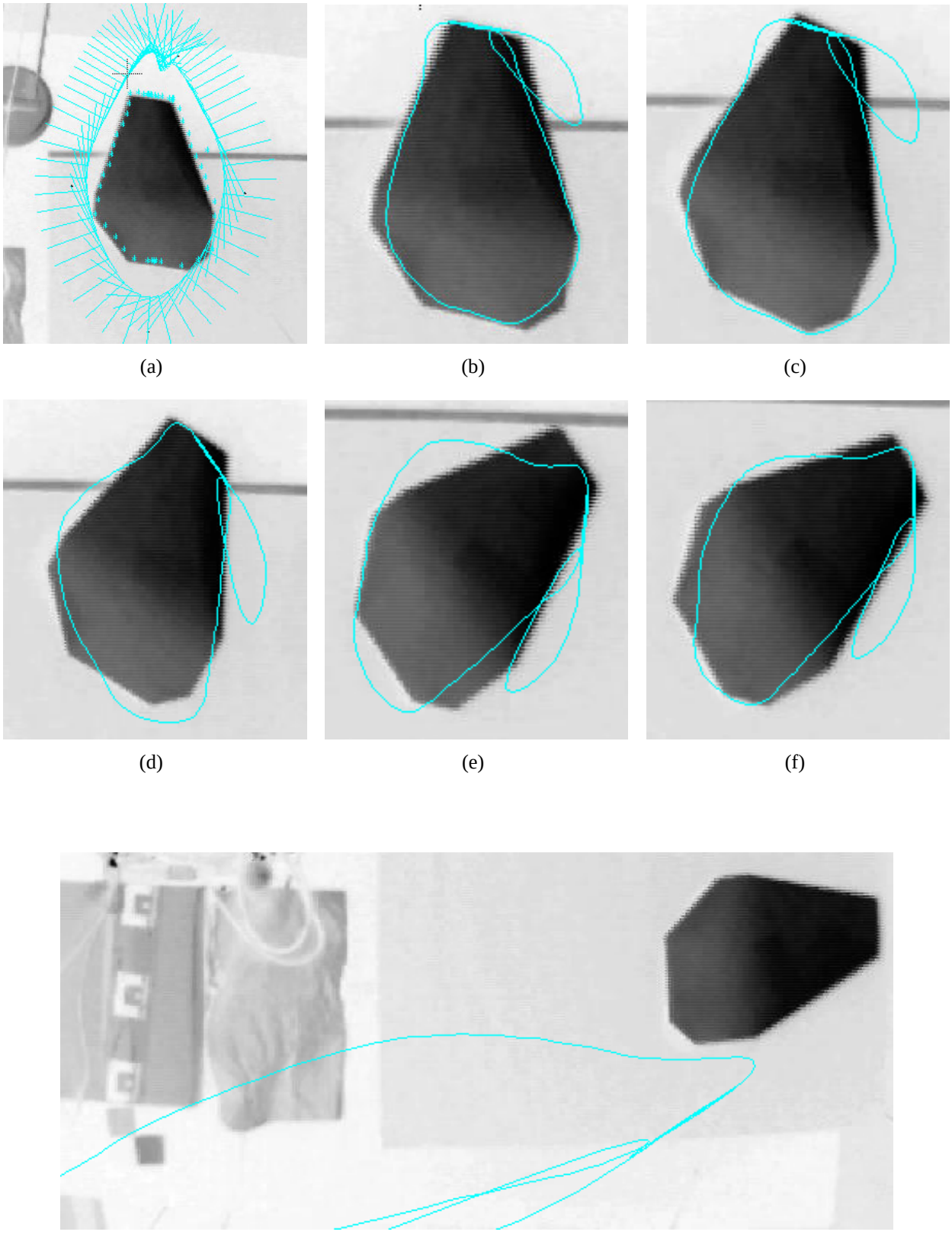
[4] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 616–625, 1990.

[5] E. Debreuve, M. Barlaud, G. Aubert, I. Laurette, and J. Darcourt. Space-time segmentation using level set active contours applied to myocardial gated SPECT. *MedImg*, 20(7):643–659, July 2001.

[6] F. Derraz, M. Beladgham, and M. Khelif. Application of Active Contour Models in Medical Image Segmentation. *International Conference on Information Technology: Coding and Computing (ITCC'04)*, 2:675, 2004.

[7] P. Dickson, J. Li, Z. Zhu, A. Hanson, E. Riseman, H. Sabrin, H. Schultz, and G. Whitten. Mosaic Generation for Under Vehicle Inspection. In *Workshop on Applications of Computer Vision*, Orlando, USA, December 2002.

[8] P. F. Felzenszwalb. Representation and Detection of Deformable Shapes. In *CVPR (1)*, pages 102–108, 2003.

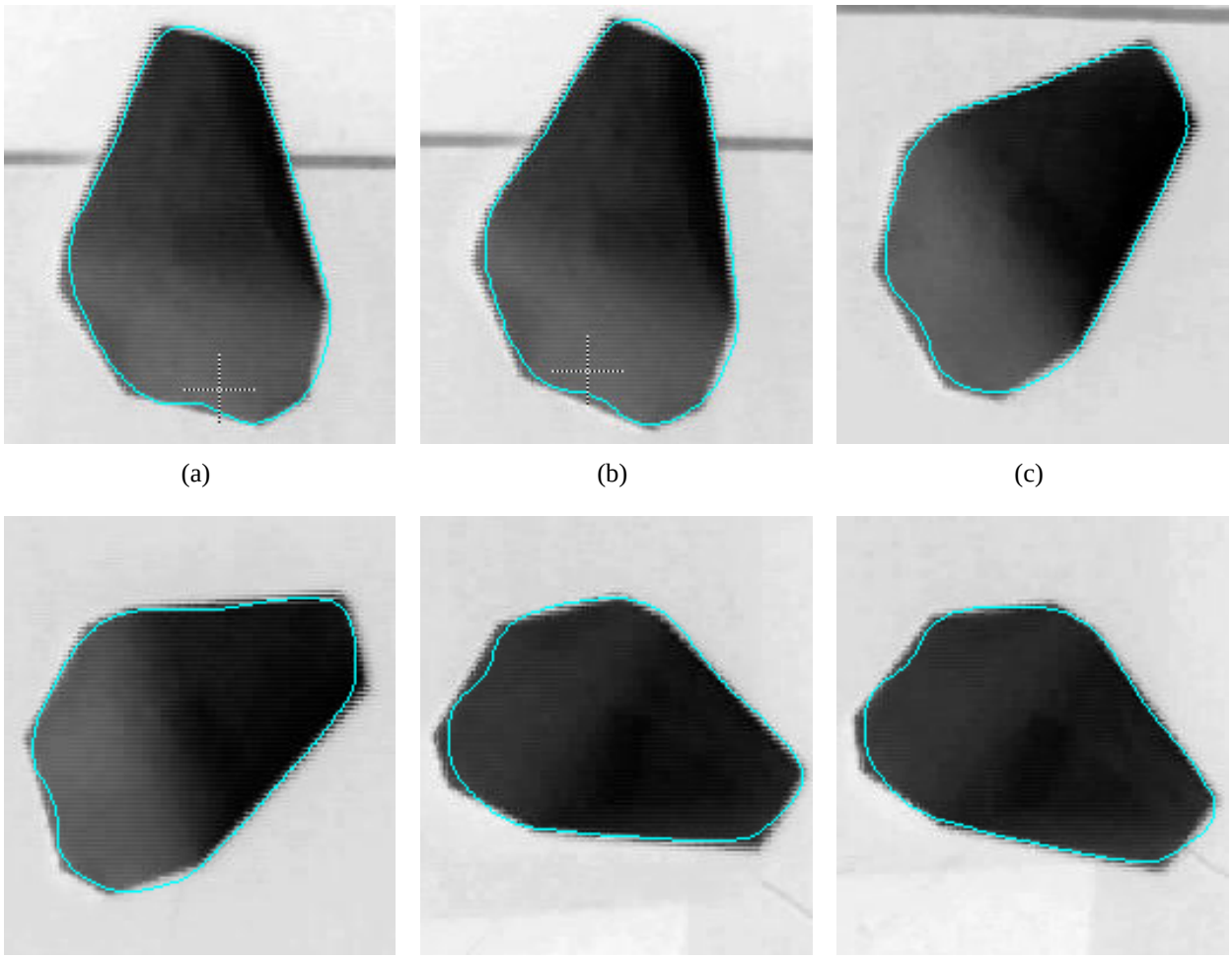


(a) (b) (c)

(d) (e) (f)

(g)

Figura 6: Seguimiento del objeto sin evitar el cruce de normales. El algoritmo es incapaz de seguir al objetos debido a los problemas en el ajuste.



(a) (b) (c)

(d) (e) (f)

Figura 7: Seguimiento del objeto utilizando el método propuesto.

[9] A. Fournier and P. Poulin. A ray tracing accelerator based on a hierarchy of 1D sorted lists. In *Proceedings of Graphics Interface '93*, pages 53–61, Toronto, Ontario, May 1993. Canadian Information Processing Society.

[10] J. Gambini, M. Mejail, J. Jacobo Berllés, and A. Frery. Feature Extraction in Speckled Imagery using Dynamic B-Spline Deformable Contours under the $\mathcal{G}^0$ Models. *International Journal of Remote Sensing*, 27:5037–5059, November 2004.

[11] M. J. Gambini, M. E. Mejail, J. Jacobo Berlles, and A. Frery. Polarimetric SAR Region Boundary Detection using B-Spline Deformable Contours under the $\mathcal{G}$ Model. In *XVII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI) / II Ibero-American Symposium on Computer Graphics (SIAGG)*, pages 764–769, Natal, Brasil, October 2005.

[12] A. S. Glassner. *An Introduction to Ray Tracing*. Academic Press, London, 1989.

[13] F. Huang and J. Su. Deformable Pedal Curves with Application to Face Contour Extraction. In *CVPR (1)*, pages 328–333, 2003.

[14] J. Denzler and H. Niemann. Active Rays: Polar-transformed Active Contours for Real-Time Contour Tracking. *Real-Time Imaging*, 5(3):203–213, 1999.

[15] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Model. *International Journal of Computer Vision*, 1(1):321–333, March 1988.

[16] A. Katz, D. Wassermann, J. Gambini, J. Jacobo, and M. Mejail. Real Time object tracking for Mirosot League. In *FIRA Robot World Congress*, 2003.

[17] D. Kim. B-Spline Representation of Active Contours. In *Fifth International Symposium on Signal Processing and its Applications, ISSPA99*, Brisbane, Australia, 1999.

[18] S. Lefèvre and N. Vincent. Real Time Multiple Object Tracking Based on Active Contours. In *ICIAR (2)*, pages 606–613, 2004.

[19] J. Mahovsky and B. Wyvill. Fast ray-axis aligned bounding box overlap tests with Plücker coordinates. *journal of graphics tools*, 9(1):35–46, 2004.

[20] S. Malassiotis and M. G. Strintzis. Tracking the left ventricle in echocardiographic images by learningheart dynamics. *IEEE Transactions on Medical Imaging*, 18:282–290, March 1999.

[21] G. Medioni and Y. Yasumoto. Corner detection and curve representation using curve B-Splines. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 764–769, 1986.

[22] F. Precioso, M. Barlaud, T. Blu, and M. Unser. Smoothing B-Spline Active Contour for Fast and Robust Image and Video Segmentation. In *Proceedings of the 2003 IEEE International Conference on Image Processing (ICIP'03)*, pages 137–140, Barcelona, Spain, September 2003.

[23] D. F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill, New York, 2nd edition, 1990.

[24] H. Schmidl, N. Walker, and M. Lin. CAB: Fast update of OBB trees for collision detection between articulated bodies. *journal of graphics tools*, 9(2):1–9, 2004.

[25] S. Suri, P. M. Hubbard, and J. F. Hughes. Analyzing bounding boxes for object intersection. *ACM Transactions on Graphics (TOG)*, 18(3):257–277, July 1999.

[26] D. Wassermann, M. Mejail, J. Gambini, and M. E. Buemi. Segmentation with active contours: a comparative study of B-spline and level set techniques. In *SIBGRAPI XVII:17th Brazilian Symposium on Computer Graphics and Image Processing*. IEEE, 2004.

[27] G. Zachmann. Minimal Hierarchical Collision Detection. In *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 121–128, Hong Kong, China, November11–13 2002.

[28] Z. Zhu, G. Xu, E. Riseman, and A. Hanson. Fast Generation of Dynamic and Multi-Resolution 360 Degrees Panorama from Video Sequences. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 400–406, Florence, Italy, 1999.

[29] A. Zomorodian and H. Edelsbrunner. Fast software for box intersections. *International Journal of Computational Geometry and Applications*, 12(1-2):143–172, 2002.