Algoritmo de seguimiento de objetos basado en visión asistida por computador en tiempo real utilizando CAMShift e histogramas ponderados

A. Hernández*‡, R. Verrastro*, L. Di Matteo*, M. Prieto*, M. Marufo*, J. Gomez*§, C. Verrastro*†

*Grupo de Inteligencia Artificial y Robótica (GIAR), Univ. Tecnológica Nacional Fac. Regional Bs. As., Buenos Aires, Argentina.
†Centro Atómico Ezeiza, Comisión Nacional de Energía Atómica (CNEA), Buenos Aires, Argentina.
‡Centro de Micro y Nanoelectrónica, Instituto Nacional de Tecnología Industrial (INTI), Buenos Aires, Argentina.
§Centro de Electrónica e Informática, Instituto Nacional de Tecnología Industrial (INTI), Buenos Aires, Argentina.

ariel.h.estevenz@ieee.org, ramiroverrastro@gmail.com, leandro.dimatteo@ieee.org

***Resumen*—En el campo de visión por computadora el seguimiento de objetos en video es la base de muchas aplicaciones que van desde la producción de video hasta la vigilancia remota, y desde la robótica hasta los juegos interactivos. Básicamente esta tarea se compone por tres puntos clave: la detección de objetos en movimiento, el seguimiento de los mismos en cada instante, y el respectivo análisis. Con el objetivo de mejorar el funcionamiento de técnicas tradicionales de seguimiento, se toma como base el algoritmo CAMShift y se propone un algoritmo con mejoras basadas en el histograma de color del objeto. El histograma es actualizado dinámicamente. Se aplican técnicas de filtrado dando como resultado una segmentación del objeto de interés con un menor nivel de ruido facilitando mejores resultados en su detección y seguimiento. Para evaluar el rendimiento obtenido en el algoritmo se realiza una medición manual de la posición exacta del objeto y se la contrasta con la posición estimada por el algoritmo. En los ensayos el objeto se detecta y se sigue continuamente sin interrupciones, incluso en oclusiones cortas. Las mejoras propuestas implican mayor necesidad de procesamiento que CAMShift, y como ventaja se demuestra que el error en seguimiento es menor y más robusto porque se logra un seguimiento continuo que el algoritmo estándar no logra alcanzar y presenta discontinuidades en el seguimiento para los ensayos realizados.**

***Palabras clave*—Procesamiento de imágenes, visión asistida por computador, seguimiento de objetos, CAMShift, comparación de histogramas, detector Canny**

I. Introducción

La disponibilidad de cámaras de video de alta calidad y de bajo costo, la proliferación de computadoras de alta performance y la creciente necesidad de análisis de video automatizada ha generado un gran interés en los algoritmos de seguimiento de objetos [1], dando como resultado que este tipo de aplicaciones sea una tarea relevante dentro del campo de la visión por computadora [2][3].

En su forma más simple, el seguimiento puede ser definido como el problema de la estimación de la posición y trayectoria de un objeto en el plano de la imagen mientras se mueve alrededor de una escena [4][5][6][7][8][9].

Nuestro trabajo de investigación se ha enfocado en desarrollar algoritmos optimizados de seguimiento de objetos en video [10][11]. Esto ha dado origen a dos trabajos, uno enfocado en la optimización con respecto al tiempo de respuesta y otro enfocado en la optimización con respecto la robustez de seguimiento. En este trabajo presentamos un algoritmo con una performance de robustez en el seguimiento superior al ya conocido algoritmo **CAMShift** [10][11][12][13][14][15][16].

Se propone un algoritmo de seguimiento de un objeto basado en su histograma de color. Dicho histograma es actualizado dinámicamente en tiempo real. El algoritmo propuesto consta de las siguientes etapas: Inicialización, Cálculo de histograma y Seguimiento. Se utilizan algoritmos estándar y se implementan mejoras para lograr un seguimiento continuo y en tiempo real ante diferentes cambios en las condiciones de iluminación y perspectiva visual.

II. Desarrollo

Hay tres pasos clave para el análisis de video: la detección de objetos en movimiento, el seguimiento de este tipo de objetos frame a frame, y el respectivo análisis [1][2]. Para el diseño del algoritmo de seguimiento de objetos propuesto se realiza una implementación basada en la funcionalidad **CAMShift** de *OpenCV*[14], en el cual se agregan características para mejorar la detección del objeto y el seguimiento. Estas mejoras se basan en el filtrado de la imagen inicial dentro de la **ROI** inicial y en la detección del color del objeto a seguir. En las siguientes secciones se detallan las características de cada una de las partes del algoritmo.

En la Fig. 1 se observa el diagrama en bloques del algoritmo, el cual se divide principalmente en tres grandes etapas:

1. **Inicialización**: Se adquiere la imagen y se realiza la transformación de espacio de colores de **RGB** a **HSV**. Esto permite calcular el histograma inicial en base a el *matiz del color* una vez que es seleccionada la zona donde se encuentra el objeto a seguir (sección A).
2. **Cálculo de histograma**: Durante esta etapa se realizan los cálculos de *comparación* y *promedio ponderado* del histograma de color (sección B).
3. **Seguimiento**: Recibe como parámetro el histograma inicial perteneciente a la etapa de *inicialización*, y en esta

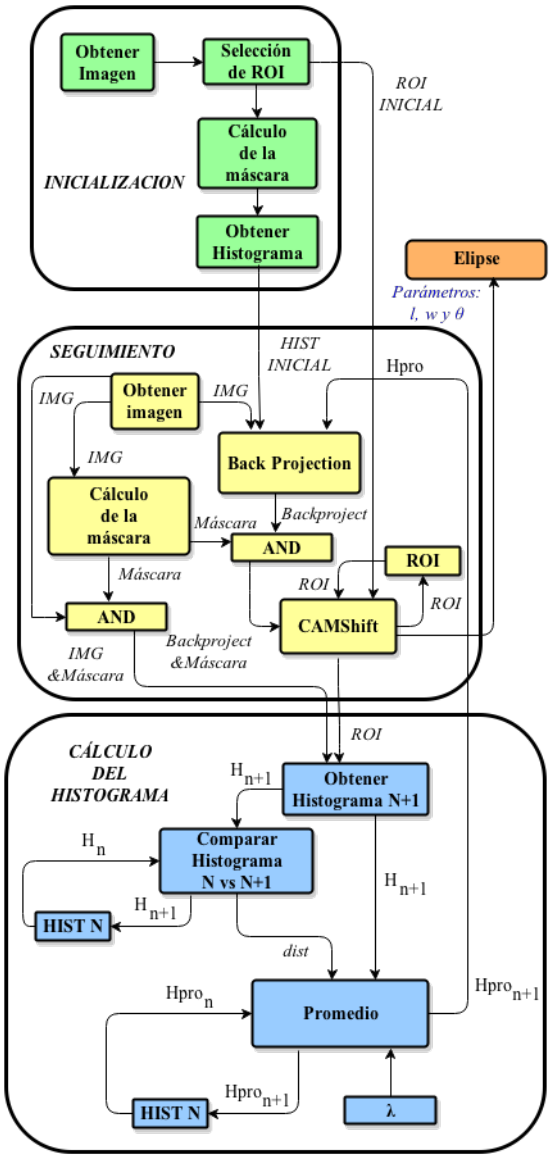


Figura 1: Esquema del algoritmo de seguimiento.

fase realiza los cálculos correspondiente a la posición del objeto dando como resultado una nueva zona o región de interés (*region of interest*, **ROI**). Esta información se realimenta al bloque *cálculo de histograma* para obtener el valor del histograma promedio y así, nuevamente, determinar la posición actualizada del objeto en seguimiento (sección C).

A. *Inicialización*

En esta etapa el algoritmo realiza un pre-procesamiento en la imagen tomada de un archivo de video o directamente desde una cámara (Fig. 1). Luego, el usuario selecciona de forma manual la región de interés que determina la zona de búsqueda inicial del objeto a seguir.

Uno de los ajustes realizados durante el pre-procesamiento es la transformación de **RGB** a **HSV**, un espacio de colores en el que se utilizan los siguientes componentes de color: matiz (*hue*), saturación (*saturation*) y luminancia (*value*). Este espacio permite independizar, por ejemplo, el tono del brillo en un color determinado, siendo muy útil al momento de realizar la detección de un objeto a partir su histograma de color[21]. Una diferencia que existe entre ambos espacios, es que **RGB** es la manera en que una computadora trata al color, en cambio **HSV** captura cada componente del mismo en forma analoga a la que los seres humanos lo perciben.

A partir de la selección de la **ROI**, se obtiene el *histograma de color* correspondiente al objeto de interés para realizar la detección por color del mismo y como condición inicial para el cálculo del promedio de histograma (sección B). El cálculo de este histograma se realiza solamente mediante el uso del canal *hue* del espacio **HSV** para disminuir los tiempos de cálculo durante la ejecución del algoritmo.

Además, como se verá más adelante en la etapa de *seguimiento* o *tracking* (sección C), es necesario brindarle al algoritmo el histograma para realizar el cálculo del *Back Projection* o simplemente *backproject*, que es una imagen en donde el valor de cada uno de los píxeles corresponde a la probabilidad de que dicho pixel pertenezca al objeto de interés.

B. *Cálculo, comparación y actualización del histograma*

Esta fase comienza con el cálculo del valor del histograma en la **ROI** actual, esto es, en la zona de seguimiento como se muestra en el último bloque de la Fig. 1. La finalidad es comparar los histogramas que se calculan inicialmente y el actual. En función de este resultado se evalúa el promedio histórico del histograma, que permite corregir divergencias de los valores del histograma del objeto de interés.

Para observar discrepancias entre histogramas es necesario obtener un valor numérico que indique el grado de relación entre ellos. Entonces, dadas las distribuciones normales p y q, la *distancia de Bhattacharyya* está definida como[22],

$$\mathtt{d}(p,q) = \frac{1}{4} ln(\frac{1}{4}(\frac{\sigma_p^2}{\sigma_q^2} + \frac{\sigma_q^2}{\sigma_p^2} + 2)) + \frac{1}{4}(\frac{\mu_p^2 - \mu_q^2}{\sigma_p^2 + \sigma_q^2}) \qquad (1)$$

Donde μ_p, σ_p, μ_q y σ_q son la media y el desvío estándar de las distribuciones p y q respectivamente. En este trabajo se utiliza la implementación orientada a la comparación de histogramas de la *distancia de Bhattacharyya*[14] perteneciente a las librería *OpenCV*, cuya ecuación está dada por,

$$\mathtt{d}(H_{n+1}, H_n) = \sqrt{1 - \sum_i^N \frac{\sqrt{H_n(i).H_{n+1}(i)}}{\sum_i^N H_{n+1}(i) \sum_i^N H_n(i)}} \qquad (2)$$

Donde H_n y H_{n+1} son el histograma *anterior* y *actual* respectivamente. Cada histograma es un vector compuesto por N contenedores o *bins*. En nuestro algoritmo hemos considerado 30. El resultado en (2) es un valor entre $[0, 1]$, y es interpretado como la distancia entre los histogramas, esto es, a medida que nos aproximamos a 0 las discrepancias entre ellos es menor.

Esta distancia es utilizada para obtener un *valor de ponderación* en el cálculo del promedio del histograma, que permite darle mayor prioridad a los histogramas que se encuentren "más cerca" del *histograma promedio*. A continuación se define la ecuación del promedio (ponderado exponencialmente y actualizado en forma dinámica),

$$Hpro_{n+1} = Hpro_n + (H_{n+1} - Hpro_n).\mathsf{e}^{(-\lambda.\mathsf{d}(H_{n+1}, Hpro_n))} \quad (3)$$

Donde $Hpro_n$ y $Hpro_{n+1}$ son el histograma promedio *anterior* (inicialmente $Hpro_n$ corresponde al histograma inicial como se observa en la Fig. 1) y *actual* respectivamente, y λ es un parámetro de ajuste a la ecuación[20]. En *valor de ponderación* $\mathsf{e}^{(-\lambda.\mathsf{d}(H_{n+1}, Hpro_n))}$ evita que cualquier histograma que no represente la distribución de color del objeto de interés sea incorporado en el promedio y haga que el algoritmo de seguimiento diverja, perjudicando la detección en el algoritmo de seguimiento.

Las ecuaciones (2) y (3) brindan mayor estabilidad en la obtención del histograma de color en la imagen, dado que evita cambios imprevistos en el *backproject* o muy diferentes del histograma del objeto en interés, pero a su vez la información obtenida es ponderada e incorporada al histograma histórico.

C. Seguimiento

En esta etapa de *seguimiento* (segundo bloque de la Fig. 1) se realiza el cálculo de la ubicación del objeto de interés en el cuadro de la imagen (*frame*) en forma continua.

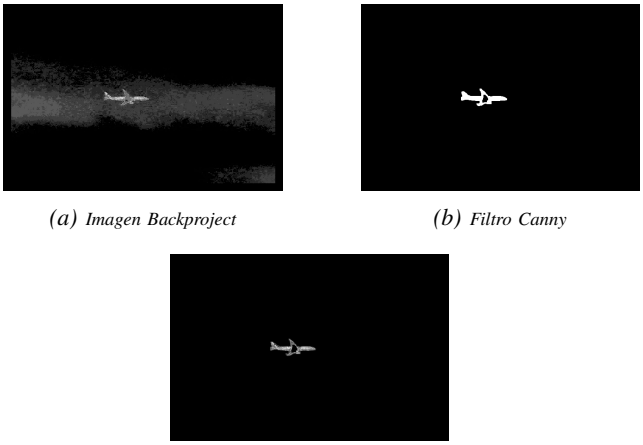
A partir del histograma de color el cálculo de la imagen de *backproject* se realiza de la siguiente manera:

1. En primer lugar, se realiza el calculo del *Back Projection* a partir del *histograma de color* del objeto a seguir, se crea una imagen que indica la zona de probabilidad de la ubicación de éste (Fig. 2a).
2. Por otro lado se realiza la aplicación del *filtro Canny* a la imagen. El algoritmo de *Canny*[18] es un operador que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes. Además, una vez efectuado este filtrado se realiza una operación de *cierre*. Ambas operaciones permiten filtrar gran parte del ruido y focalizarse en el objeto de interés (Fig. 2b).
3. Finalmente se realiza una AND entre estos dos resultados formando una única imagen de *backproject* como se muestra en la Fig. 2c. En esta etapa se aplica la operación lógica AND píxel a píxel entre las dos imágenes obtenidas anteriormente.

Con este procedimiento se obtiene en el *backproject* una imagen del objeto a seguir mejor segmentada, esto le permite a la técnica de *seguimiento* poder visualizarlo con un menor nivel de ruido.

Se ha utilizado el algoritmo **CAMShift** (*Continously Adaptive Mean Shift*) para realizar el seguimiento de un objeto en la secuencia de imágenes, que está basado en la técnica *MeanShift*[10][11][13][14][15][16][17]. Esta técnica es un proceso iterativo, donde para cada iteración se tiene una nube de puntos en el espacio, y evaluando la densidad de las intensidades de los puntos dentro de una **ROI**, que encierra a dicha nube, se obtiene un máximo local, es decir un punto (*centroide* de la región) en donde la densidad de intensidades es máxima.

Para distribuciones de probabilidad discreta de una imagen 2D (*backproject*), los puntos corresponden a pixeles y las intensidades al valor de probabilidad, la ubicación del *centroide* dentro de la ventana de búsqueda se calcula de la siguiente manera [17]:



(a) Imagen Backproject *(b) Filtro Canny*

(c) Backproject AND *Canny*

Figura 2: Tracking

1. Primero se calcula el Momento cero:

$$M_{0,0} = \sum_x \sum_y I(x, y) \quad (4)$$

2. En donde $I(x, y)$ es el valor del píxel (probabilidad) en la posición (x, y) en la imagen, luego los primeros momentos de x e y:

$$M_{1,0} = \sum_x \sum_y xI(x, y) \quad (5)$$
$$M_{0,1} = \sum_x \sum_y yI(x, y) \quad (6)$$

3. Entonces el centroide de la ventana de búsqueda de *MeanShift* es:

$$x_c = \frac{M_{1,0}}{M_{0,0}} \quad (7)$$
$$y_c = \frac{M_{0,1}}{M_{0,0}} \quad (8)$$

El algoritmo de *MeanShift* opera sobre distribuciones de probabilidad estáticas e identifica objetos a través de su color en secuencias de video mediante *histogramas de color*, ya que los datos en una imagen color deben ser representado mediante una distribución de probabilidad. Para adaptar de forma dinámica la distribución de probabilidad y realizar el seguimiento de patrones la técnica *Meanshift* tuvo que ser modificada, y **CAMShift** es el resultado de esta modificación.

CAMShift ajusta el tamaño de la ventana en cada *frame* de la siguiente manera:

1. Los segundos momentos:

$$M_{2,0} = \sum_x \sum_y x^2 I(x, y) \quad (9)$$
$$M_{0,2} = \sum_x \sum_y y^2 I(x, y) \quad (10)$$
$$M_{1,1} = \sum_x \sum_y xyI(x, y) \quad (11)$$

2. Luego se calculan los valores a, b y c para obtener las

distancias de la nueva ventana:

$$a = \frac{M_{2,0}}{M_{0,0}} - x_c^2 \tag{12}$$

$$b = 2(\frac{M_{1,1}}{M_{0,0}} - x_c y_c) \tag{13}$$

$$c = \frac{M_{0,2}}{M_{0,0}} - y_c^2 \tag{14}$$

3. Entonces la inclinación , el largo y el ancho de la nueva ventana están dados por:

$$\theta = \frac{1}{2} arctan \frac{b}{a-c} \tag{15}$$

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 - (a-c)^2}}{2}} \tag{16}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 - (a-c)^2}}{2}} \tag{17}$$

La función **CAMShift** devuelve en cada instante la ubicación de la ventana a través de estos tres parámetros (θ, w y l), los cuales son requeridos para dibujar sobre el *frame* una elipse que muestra la ubicación del objeto de interés (Fig. 3).

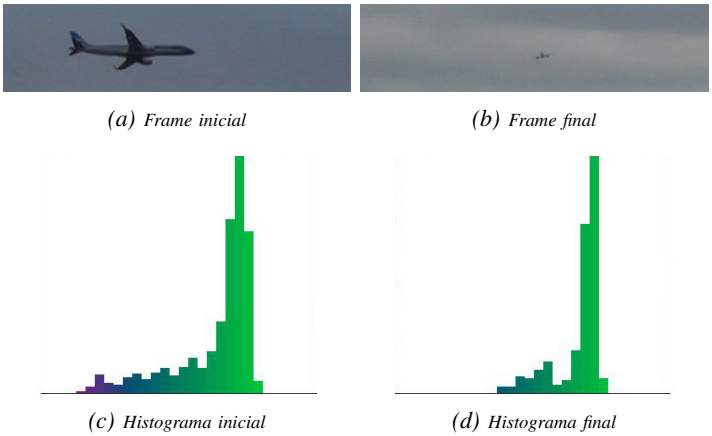
Figura 3: Elipse sobre el objeto en seguimiento

Una vez obtenida esta nueva posición, se calcula el histograma nuevamente como se indica en la sección B para continuar con el seguimiento en el siguiente *frame* con los valores actualizados.

Las mejoras introducidas posibilitan el seguimiento de objetos que cambian gradualmente sus dimensiones y colores, en la Fig. 4 se muestran las imágenes e histogramas de color correspondientes al instante inicial y final del video bajo ensayo (sección III).

D. *Parámetros de calidad*

Para evaluar la calidad del algoritmo se tienen en cuenta dos parámetros, el error en la medición de la posición del objeto y el tiempo de procesamiento del algoritmo. En cuanto a la evaluación correspondiente a la calidad de medición de la posición, se tomaron diferentes valores de la ubicación del objeto en el cuadro de la imagen en forma manual. A partir de estas mediciones se calcula la posición patrón del objeto mediante el promedio de los puntos en x e y obtenidos en ellas y son denominados x_v e y_v. El Cuadro I muestra la media y el desvío estándar correspondiente a la distribución estadística inherente a las mediciones.



(a) Frame inicial *(b) Frame final*

(c) Histograma inicial *(d) Histograma final*

Figura 4: Cuadro inicial y final del video utilizado para el ensayo

Distancia	Media (μ)	Desvío estándar (σ)
δ_x	0.3 %	0.4 %
δ_y	0.4 %	0.5 %

Cuadro I: Valores estadísticos correspondiente al error de la posición patrón del objeto

El error δ_x, δ_y y δ_{xy}, relativo al las dimensiones del *frame* ($ancho_{frame}$ y $alto_{frame}$) se evalúan según la siguiente fórmula:

$$\delta_{x\,\%} = \frac{100.|x_m - x_v|}{ancho_{frame}} \tag{18}$$

$$\delta_{y\,\%} = \frac{100.|y_m - y_v|}{alto_{frame}} \tag{19}$$

$$\delta_{xy\,\%} = \frac{100.\sqrt{(x_m - x_v)^2 + (y_m - y_v)^2}}{\sqrt{alto_{frame}.alto_{frame} + ancho_{frame}.ancho_{frame}}} \tag{20}$$

Donde x_m e y_m son los valores obtenidos mediante el algoritmo de seguimiento. Con estas mediciones se analiza su distribución normal para los posibles valores de error y se obtiene un nivel de confianza aceptado por el sistema.

Con respecto al tiempo de procesamiento se realizaron pruebas de rendimiento en una PC con procesador *Intel Core i3* y $3GB$ de memoria RAM bajo el sistema operativo *Debian*. De todas maneras, como se ha mencionado anteriormente, este algoritmo no tiene como objetivo mejorar la performance en tiempo, sino la robustez en cuanto a la detección de la posición del objeto bajo diferentes circunstancias.

III. Análisis de resultados

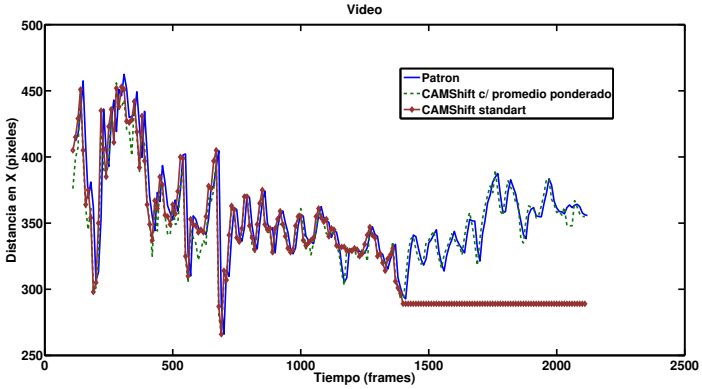
Los ensayos se realizaron en base a tres videos de diferentes características: *normal*, *reducido a la mitad de su tamaño* y *con la adición de ruido gaussiano*, cada uno con una resolución y frecuencia de $720x480$ y $30fps$ respectivamente. Para la comparación se utilizó el algoritmo de seguimiento estándar **CAMShift**[17].

En las Fig. 5a y Fig. 5b se muestra la posición obtenida con las técnicas de **CAMShift** estándar y **CAMShift** con promedio de histograma con respecto a la posición real del objeto en función de cada *frame*, correspondiente al primer video

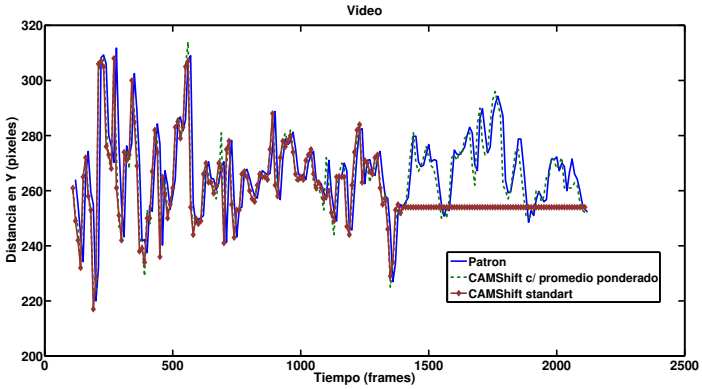
(*normal*). Se puede observar que el algoritmo con promedio de histograma tiene un comportamiento mucho más eficiente en cuanto al cálculo de la posición del objeto. La técnica estándar no logra estimar la posición en todos los *frames* e incluso una vez que sale fuera de su área de búsqueda pierde al objeto por completo (aproximadamente en el frame 1450).

Aplicando una compresión del 50 % al video, el algoritmo con promedio de histograma mantiene el mismo rendimiento que en el caso anterior, a diferencia de la técnica clásica que pierde el objeto dos veces más. El objetivo de realizar este ensayo es disminuir la calidad de la imagen a la mitad, de esta manera los algoritmos deben lograr segmentar el objeto con un mayor nivel de dificultad (Fig. 6)

Para determinar las mejoras en la robustez del algoritmo, se adiciona al video *ruido blanco gaussiano* de $\mu = 0$ y $\sigma = 0{,}01$, verificando incrementos en el rechazo al ruido respecto del algoritmo de **CAMShift** estándar (Fig. 7).



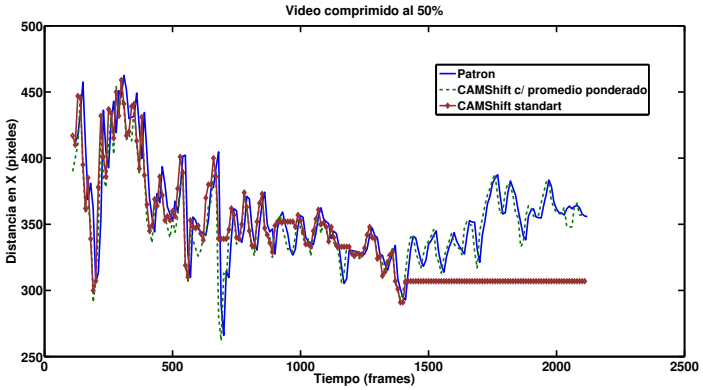
(a) Posición en x



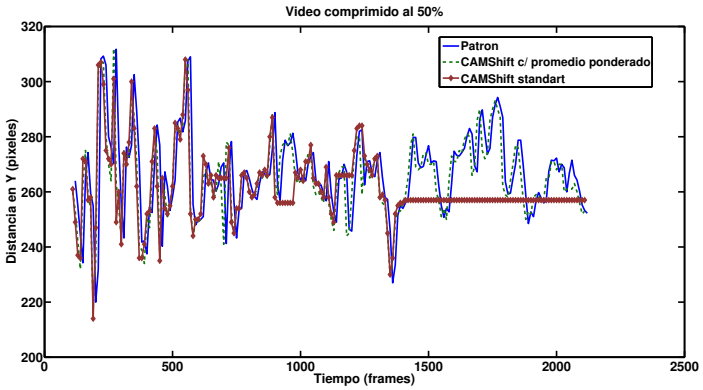
(b) Posición en y

Figura 5: Comparación en la detección del objeto entre su posición real y estimada en x e y

Se evaluó la distribución de error en pixeles para las dos coordenadas por separado y en conjunto (δ_x, δ_y y δ_{xy}). En el Cuadro II se muestran los valores estadísticos para cada distribución correspondiente a cada ensayo. Buscando un nivel de confianza del 95 % ($\mu \mp 2\sigma$) se obtienen errores alrededor del 4.5 %, llegando al 7 % con la adición de ruido blanco. Adicionalmente se puede observar que los valores estadísticos obtenidos en las mediciones del patrón de referencia (Cuadro I) se encuentran



(a) Posición en x



(b) Posición en y

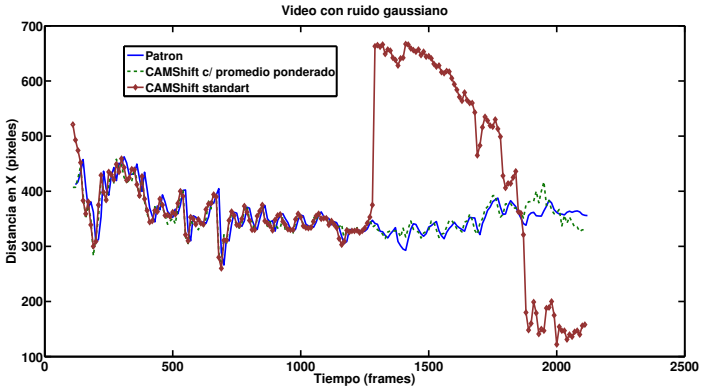
Figura 6: Comparación en la detección del objeto mediante la aplicación de una compresión al video original

en el orden de la tercera parte de los valores correspondientes al error de posición durante los ensayos (Cuadro II).

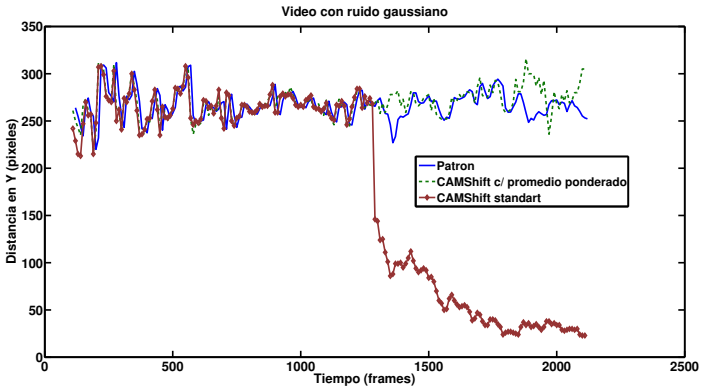
Video	δ	Media (μ)	Desvío estándar (σ)
Normal	δ_x	1 %	1.7 %
	δ_y	0.8 %	2.3 %
	δ_{xy}	1.1 %	1.8 %
Comprimido al 50 %	δ_x	1 %	1.6 %
	δ_y	0.7 %	2.3 %
	δ_{xy}	1 %	1.8 %
Con ruido	δ_x	1.2 %	1.3 %
	δ_y	1.5 %	2.7 %
	δ_{xy}	1.5 %	1.7 %

Cuadro II: Valores estadísticos para el error de posición horizontal, vertical y horizontal-vertical

En el Cuadro III se muestran los resultados del tiempo de procesamiento consumido por cada algoritmo en cada uno de los videos ensayados. Se observa que el algoritmo tiene un tiempo de respuesta mayor con respecto al algoritmo seguidor de objetos basado en **CAMShift** estándar, lógicamente esto es debido a las mejoras introducidas en este algoritmo que permiten incrementar la robustez del seguimiento.



(a) Posición en x



(b) Posición en y

Figura 7: Comparación en la detección del objeto con ruido adicional

Video	CAMShift con promedio de histograma	CAMShift estándar
Normal	29ms	14ms
Con ruido	29ms	16ms

Cuadro III: Cuadro de tiempos

IV. Conclusión

En este trabajo se proponen mejoras al algoritmo de **CAMShift**, basadas en el histograma de color del objeto, actualizado dinámicamente. El algoritmo se adapta a las necesidades enunciadas en la sección I para las técnicas de seguimiento de objetos basadas en visión por computador.

En cuanto al rendimiento del algoritmo, se puede observar que el objeto se detecta y se sigue continuamente, incluso en oclusiones cortas, siendo que no se aleje de la región de búsqueda, logrando una mejor segmentación del objeto y una disminución del ruido en comparación con el algoritmo estándar utilizado como referencia. Es importante destacar que la implementación presentada es un aporte a las técnicas clásicas de seguimiento de objetos, el cual admite futuras mejoras del algoritmo, como la combinación con modelos probabilísticos para la estimación de la posición del objeto a seguir durante oclusiones largas.

Referencias

[1] A. Yilmaz O. Javed M. Shah, *Object Tracking: A Survey*, ACM Computing Surveys, Vol. 38, No. 4, Article 13, 2006.

[2] D. Ballard C. Brown, *Computer Vision.* Prentice-Hall. 1982.

[3] C. Papageorgiou M. Oren T. Poggio, *A general framework for object detection.* In IEEE International Conference on Computer Vision (ICCV). 555–562. 1998.

[4] T. Broida R. Chellappa, *Estimation of object motion parameters from noisy images.* IEEE Trans. Patt. Analy. Mach. Intell. 8, 1, 90–99. 1986.

[5] J. Kang I. Cohen G. Medioni, *Continuous tracking within and across camera streams.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 267–272. 2003.

[6] M. Black P. Anandan, *The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields.* Computer Vision Image Understand. 63, 1, 75-104. 1996.

[7] C. Veenman M. Reinders E. Backer, *Resolving motion correspondence for densely moving points.* IEEE Trans. Patt. Analy. Mach. Intell. 23, 1, 54–72. 2001.

[8] T. Cootes G. Edwards C. Taylor, *Robust real-time periodic motion detection, analysis, and applications.* IEEE Trans. Patt. Analy. Mach. Intell. 23, 6, 681–685. 2001.

[9] D. Huttenlocher J. Noh W. Rucklidge,*Tracking nonrigid objects in complex scenes.* In IEEE International Conference on Computer Vision (ICCV). 93–101. 1993.

[10] Y. Bar-shalom T. Foreman, *Tracking and Data Association.* Academic Press Inc. 1988.

[11] D. B. Reid *An algorithm for tracking multiple targets.* IEEE Trans. Autom. Control 24, 6, 843–854. 1979.

[12] MeanShiftSegmentSrc. *Mean-Shift Segmentation Source Code.* http://www.caip.rutgers.edu/riul/research/code.html.

[13] MeanShiftTrackSrc. *Mean-Shift Tracking Source Code.* http://www.intel.com/technology/computing/opencv/index.htm.

[14] G. Bradski, *Learning OpenCV Computer Vision with OpenCV Library*, O'Reilly Media, 2008.

[15] Y. Cheng, Mean Shift, *Mode Seeking, and Clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol.17, pp.790-799, 1995.

[16] D. Comaniciu P. Meer, *Mean shift: A robust approach toward feature space analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol.24, pp.603-619, 2002.

[17] G. Bradski, *Computer Vision Face Tracking For Use in a Perceptual User Interface*, Microcomputer Research Lab, Intel Corporation, 1998.

[18] J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, 1986.

[19] D. Comaniciu V. Ramesh P. Meer, *Kernel-based object tracking*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI),Vol.25, pp.564-577, 2003.

[20] P. Pérez C. Hue, *Color-Based Probabilistic Tracking*, Computer Vision — ECCV 2002, Lecture Notes in Computer Science, Vol 2350, pp 661-675, 2002.

[21] L. Shuhua G. Gaizhi, *The application of improved HSV color space model in image processing*, Future Computer and Communication (ICFCC), 2010 2nd International Conference, Vol 2, pp 10-13, 2010.

[22] G. Coleman H. Andrews, *Image Segmentation by Clustering*, Proc IEEE, Vol. 67, No. 5, pp. 773-785,1979.