Algoritmo de seguimiento de objetos en imágenes mediante reconstrucción iterativa de histograma en tiempo real

M. Prieto*, M. Marufo*, L. Di Matteo*, R. Verrastro*, A. Hernández*‡, 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.

prietocanalejom@gmail.com, maximiliano.marufo@gmail.com, leandro.dimatteo@ieee.org

***Resumen*—El seguimiento de objetos es una tarea relevante dentro del campo de la visión por computador, es la base en gran cantidad de aplicaciones como puede ser la Navegación de Vehículos o el Monitoreo de tráfico. Los algoritmos utilizados para realizar seguimientos en tiempo real deben ser rápidos y eficientes, ya que no permiten grandes demoras ni errores. Con el objetivo de optimizar tiempos de procesamiento, sin introducir errores de posición de seguimiento mayores a la de los algoritmos estándar, se propone un algoritmo basado en la reconstrucción iterativa en tiempo real del histograma del objeto a seguir con una baja cantidad de puntos exploratorios en un área delimitada. Por medio de filtrados se reduce dicha área a los pixels pertenecientes al objeto, separando al mismo del fondo. A su vez, se adapta el histograma patrón a las variaciones de brillo, color y tamaño que puede sufrir dicho objeto.**

Implementando este algoritmo se obtienen mejoras en los tiempos de procesamiento en comparación con algoritmos estándar de seguimiento como Filtro de Partículas y CAMShift. Los resultados obtenidos indican que en el cálculo de posición del objeto no hay perdida de seguimiento del mismo y el error de posición cometido se encuentra en los mismos órdenes de magnitud que los algoritmos estándar de seguimiento indicados y se alcanza una mejora considerable en cuanto a tiempo de procesamiento.

Palabras clave*—*Image processing, visión asistida por computador, object tracking, reconstrucción de histograma

I. Introducción

El seguimiento de objetos es una tarea relevante dentro del campo de la visión por computador [1][2] con la proliferación de computadoras de alta performance, la disponibilidad de cámaras de vídeo de alta calidad y de bajo costo, y la creciente necesidad de análisis de video automatizada se ha generado un gran interés en los algoritmos de seguimiento de objetos [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 tiempo de procesamiento en el seguimiento de objetos en imágenes superior a los ya conocidos algoritmos **CAMShift**[12][13][14][15][16][17] y filtro de partículas [18][19][20][21].

En este trabajo se propone un algoritmo de seguimiento que se enfoca en optimizar tiempos de procesamiento para seguimiento en tiempo real de objetos que se encuentran sobre un fondo con baja frecuencia espacial de color. Dado que la característica a seguir del objeto es el color, el algoritmo se basa en la reconstrucción iterativa de su histograma por medio de puntos exploratorios que toman muestras de color. Dichos puntos exploratorios se distribuyen aleatoriamente dentro de la región de interés donde se halla el objeto.

Se realiza un ajuste continuo al histograma patrón que caracteriza al objeto incorporando a este en forma iterativa a cada frame las variaciones de brillo y color producidos a causa de los cambios de iluminación, tamaño y perspectiva que sufre el mismo propio del movimiento con respecto al dispositivo de captura de imagen.

Se considera solo una cantidad de puntos exploratorios menor al 20 % del tamaño de la región de interés que encierra al objeto a seguir. De esta manera en comparación con la cantidad de puntos analizados que consideran los algoritmos estándar de filtro de partículas se permite optimizar el tiempo de procesamiento. Con estos puntos exploratorios se reconstruye el histograma patrón que caracteriza al objeto de manera iterativa a fines de maximizar su correspondencia al objeto y garantizar su seguimiento.

II. Desarrollo

Para el diseño del algoritmo de seguimiento de objetos propuesto se caracteriza del objeto a seguir, se implementan la detección de dicha característica *frame* a *frame* y la actualización de ésta en caso sufrir modificaciones en el tiempo de procesamiento.

Para la caracterización del objeto a seguir se utiliza su histograma. Como modelo de color se utiliza el **HSV**, un espacio con los siguientes componentes de color: matiz (*hue*),

saturación (*saturation*) y luminancia (*value*), este espacio es ampliamente utilizado en aplicaciones de procesamiento de imágenes [23] ya que permite eliminar las variaciones de brillo al encontrarse dicha información separada de la del color particular.

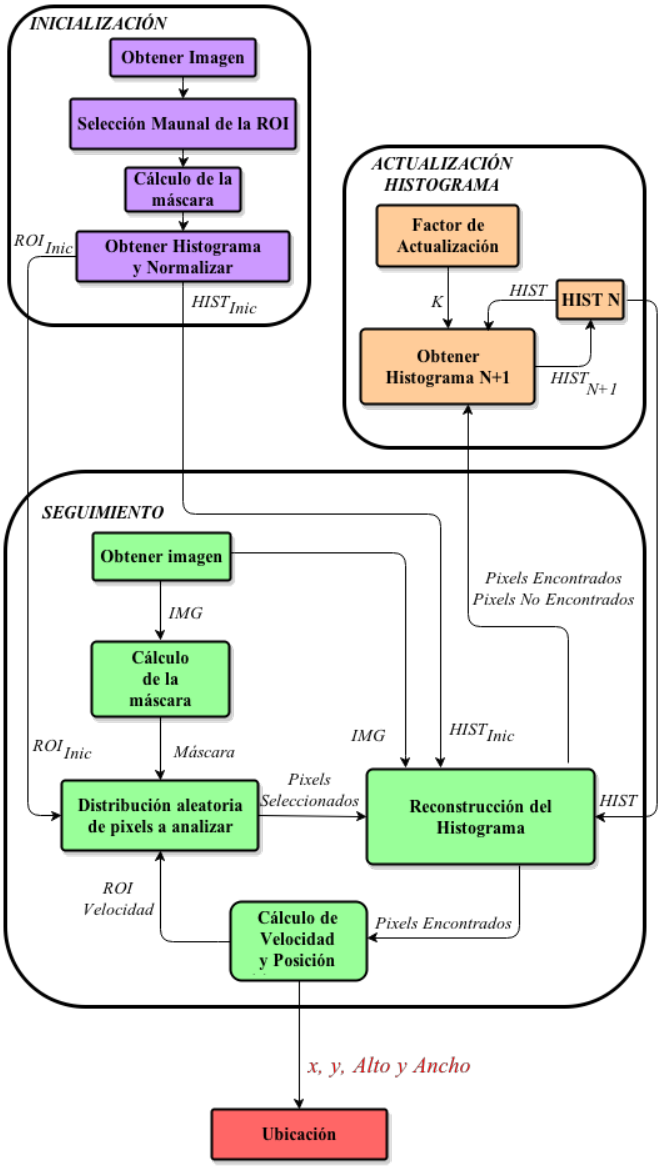


Figura 1: Esquema del algoritmo de seguimiento.

En la Fig. 1 se observa el diagrama en bloques perteneciente al algoritmo de detección, seguimiento y actualización del histograma del objeto a seguir. Este algoritmo se divide principalmente en tres grandes etapas:

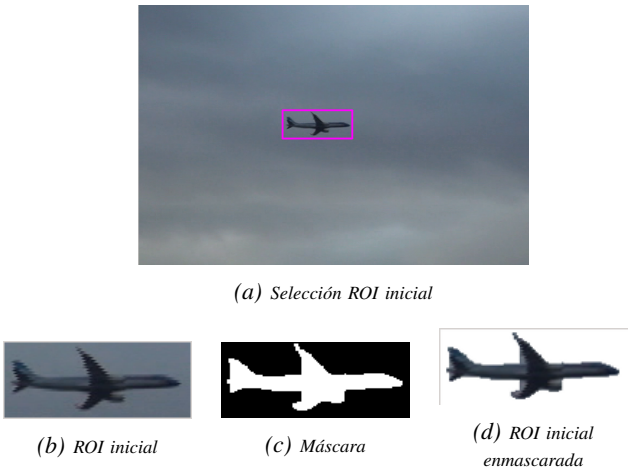
1. **Inicialización**: Se adquiere la imagen inicial, esta se trasforma al modelo de color elegido (**HSV**) y se selecciona la región de interés (**ROI**[1]) del objeto a seguir. Con dicha **ROI** se calcula el histograma patrón ($Hpatron$) incial, eliminando la información de color del fondo (sección A).
2. **Seguimiento**: Por medio de la reconstrucción del $Hpatron$ con pixels exploratorios en un área de búsqueda determinada se ubica al objeto en el siguiente *frame*. (sección B).
3. **Actualización del histograma**: Con el resultado de la fase anterior se actualiza el $Hpatron$ para incorporar las variaciones de brillo y de color del objeto seguido (sección C).

[1]Siglas en inglés de *region of interest*.

A. *Inicialización*

En esta estapa se selecciona el objeto a seguir, para ello se toma una imagen inicial en la cual se define manualmente la **ROI** a seguir, es decir, la región del cuadro actual (lo denominaremos N) donde se encuentra dicho objeto. Como se puede observar en la Fig. 2a dicha **ROI** contiene además del objeto a seguir parte del fondo donde se encuentra el mismo, al ser ese fondo desconocido, previo al calculo del histograma del objeto se construye una máscara que permite extraer el objeto del fondo.

Para la construcción de la máscara se utiliza el algoritmo *Canny* para detección de bordes[22] , y luego se aplica una *dilatación* y *erosión* (operación de cierre), para obtener de esta manera una *imagen máscara*, cuyos pixeles blancos corresponden a parte constitutivas de los objetos, y los negros al fondo (ya que suponemos al mismo homogéneo). Con el resultado de aplicar dicha máscara (Fig. 2c) a la **ROI** definida (Fig. 2b) se logra obtener los pixeles únicamente pertenecientes al objeto (Fig. 2d) que serán tomados para el calculo del histograma patrón inicial ($Hpatron_{inic}$).



(a) Selección ROI inicial

(b) ROI inicial *(c) Máscara* *(d) ROI inicial enmascarada*

Figura 2: Selección objeto a seguir

A partir de esta región enmascarada, se calcula y normaliza el $Hpatron_{inic}$ (Fig. 3) para que el área de la curva encerrada debajo de la envolvente del histograma resulte en un valor igual a la cantidad de pixels que luego serán explorados. En el cuadro siguiente ($N+1$) se procura reconstruir dicho $Hpatron$. La cantidad de pixels a explorar en la **ROI** (Px) es un parámetro que depende del área de dicha **ROI**.

B. *Seguimiento*

Se captura la imagen del cuadro siguiente ($N+1$), en la cual se define una nueva área de búsqueda de exploración.

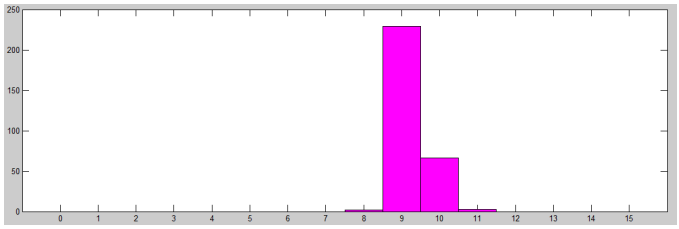


Figura 3: Histograma patron inicial

Esta nueva área de búsqueda se estima linealmente tomando en cuenta la velocidad a la cual se traslada la **ROI** de seguimiento y además se adiciona un margen (Ma) a lo largo y ancho de dicha región (Fig 4). El cálculo de la siguiente manera:

$$AreaB_x = (AreaROI_x + Vel_x.t); \quad (1)$$
$$AreaB_y = (AreaROI_y + Vel_y.t); \quad (2)$$
$$AreaB_{ancho} = AreaROI_{alto}.Ma; \quad (3)$$
$$AreaB_{alto} = AreaROI_{ancho}.Ma; \quad (4)$$

Ese margen se toma en cuenta para considerar variaciones de la velocidad de la **ROI** de seguimiento. Dicho margen es un parámetro de ajuste en el sistema. Este parámetro delimita la aceleración máxima en la cual el sistema es capaz de seguir al objeto y debe elegirse analizando el modelo de movimiento del objeto, procurando lograr una situación de compromiso entre la máxima aceleración a seguir y la minimización de información extra de la imagen que puede dificultar la identificación del $Hpatron$.

Figura 4: Area busqueda (Rojo: Area de Busqueda - Amarillo: ROI)

Dentro de la imagen en la zona delimitada por el área de búsqueda se calcula nuevamente una mascara mediante *Canny* como se realizó en la imagen inicial (Fig. 2c). Luego se distribuyen pixels de exploración sobre el área de búsqueda restringida por dicha máscara.

Para cada pixel de exploración distribuido se toma en cuenta su color y se analiza:

1. Si pertenece al $Hpatron$ se establece que es parte de la **ROI** donde se encuentra el objeto en cuadro actual,entonces se decrementa en una unidad el valor del bin correspondiente al $Hpatron$
2. Si no pertenece al $Hpatron$ se encontrará el bin del histograma correspondiente al color en un valor nulo, entonces este pixel no pertenece a la población de color del objeto a seguir, por lo tanto se registra como color de pixel no perteneciente a los colores del $Hpatron$.
3. Una vez analizados todos los pixels exploratorios distribuidos, se evalúa si la cantidad de pixels que corresponden al $Hpatron$ es inferior al un umbral de corte (Uc) ,en dicho caso se vuelven a distribuir aleatoriamente una cantidad adicional de pixels exploratorios correspondiente a la cantidad de pixels no encontrados en la zona previamente mencionada.

Esto último se reiterará hasta alcanzar una cantidad de pixels representativa conforme a la población muestreal de pixels tomadas originalmente que dieron origen al $Hpatron$. La iteración continuará hasta alcanzar el criterio de corte (Uc) establecido o una determinada cantidad de repeticiones con fines de optimizar el tiempo de procesamiento.

Ahora se determina la nueva **ROI** del objeto en seguimiento, conformada por el área rectangular que contiene a todos los pixeles que formaron parte del $Hpatron$, obteniendo así la **ROI** del objeto en seguimiento la cual representa la salida de nuestro sistema (Fig. 5)

Figura 5: ROI Salida

C. *Actualización del histograma*

Una vez analizados los pixeles exploratorios distribuidos y alcanzada alguna de las condiciones de corte descriptas, se tendrá como resultado:

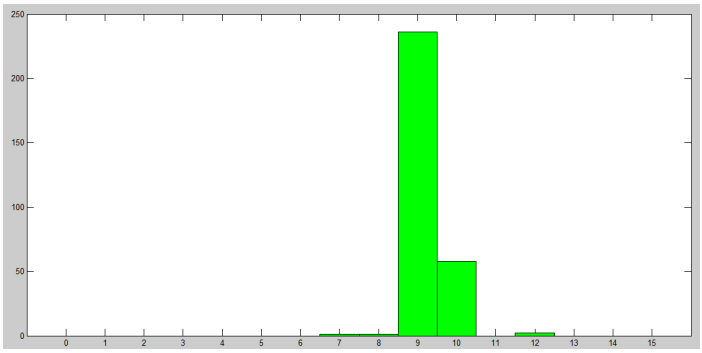
1. La nueva **ROI** del objeto en seguimiento
2. Los pixels encontrados en la **ROI**
3. Los no encontrados (de los no encontrados solamente nos quedaremos con los que están dentro de la nueva **ROI**).

Con esta información se conforma dos nuevos histogramas, el histograma de pixels no encontrados denominado H_{ROIne} y el histograma de pixels encontrados denominado H_{ROIe}, con estos histogramas se calcula el nuevo histograma ROI (H_{ROI}) de acuerdo con (5).

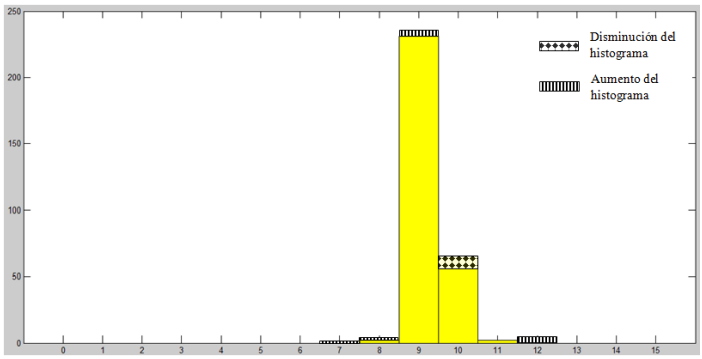
$$H_{ROI} = H_{ROIne} + H_{ROIe} \quad (5)$$

Considerando que el nuevo histograma H_{ROI} resultado del proceso indica las variaciones del patrón calculamos un nuevo histograma patrón, que utilizamos como referencia para analizar el siguiente cuadro.

Por medio de (6) se calculará el histograma patrón del siguiente cuadro ($N + 1$). Realizando un promedio en donde



(a) Histograma ROI



(b) Modificación del histograma patrón

Figura 6: Actualización histograma

se pondera la velocidad de variación del histograma patrón (k). El resultado de este cálculo se puede observar en la Fig. 6

$$Hpatron_{n+1} = Hpatron_n + (H_{ROI} - Hpatron_n).k \quad (6)$$

El nuevo histograma ($Hpatron_{n+1}$) será el $Hpatron$ en el análisis del próximo *frame*

III. Resultados

A. Parámetros de calidad

Para evaluar los objetivos propuestos por el algoritmo, optimizar tiempos de procesamiento sin introducir errores de ubicación en el seguimiento mayores a la de los algoritmos estándar, se evalúan el tiempo de procesamiento del algoritmo en videos con distintas resoluciones y la ubicación del objeto *frame* a *frame*

Para evaluar la ubicación del objeto se usaron videos de referencia de Bonn Benchmark on Tracking [25], en ellos se dispone de los datos de posición y tamaño, *frame* a *frame*, del objeto a seguir (x_p,y_p). Se compararon estos valores con los resultados del algoritmo propuesto y se determinaron los errores de posición en ambas direcciones. De dichos videos se utilizó el titulado Seq. A (Fig. 7), ya que el mismo cumple con tener un fondo sin variaciones bruscas de tonalidad lo cual es una especificación fundamental para el correcto funcionamiento del algoritmo. También se ejecuta el algoritmo implementado con el video mencionado con ruido gaussiano adicionado ($\mu = 0$ y $\sigma = 0{,}01$)

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:

$$\delta_{x\,\%} = \frac{100.|x_m - x_v|}{ancho_{frame}} \quad (7)$$

$$\delta_{y\,\%} = \frac{100.|y_m - y_v|}{alto_{frame}} \quad (8)$$

$$\delta_{xy\,\%} = \frac{100.\sqrt{(x_m - x_v)^2 + (y_m - y_v)^2}}{\sqrt{alto_{frame}.alto_{frame} + ancho_{frame}.ancho_{frame}}} \quad (9)$$

Donde x_m e y_m son los valores obtenidos mediante el algoritmo de seguimiento.

Con estos valores se analiza su distribución normal para los posibles valores de error. A su vez, se busca a partir de cual valor de error se tiene un nivel de confianza aceptado por el sistema.

B. Análisis de resultados

Las pruebas de rendimiento se realizaron con tres videos de diferentes resoluciones en una PC con procesador *Intel Core i3* y 3GB de Memoria RAM. Para realizar la comparación con el algoritmo planteado se evalúan dos algoritmos estándar de seguimiento de objetos, un **CAMShift** con poca robustez [16] (Fig. 8 y Fig. 9) y un Filtro Partículas [26] En el Cuadro I se muestran los resultados.

Resolución Video	Tiempo Algoritmo Propuesto	Tiempo Filtro de Partículas	Tiempo CAMShift
720x480	12ms	62ms	14ms
480x320	7.5ms	9ms	31.5ms
320x240	3.3ms	5.3ms	13ms

Cuadro I: Cuadro de tiempos

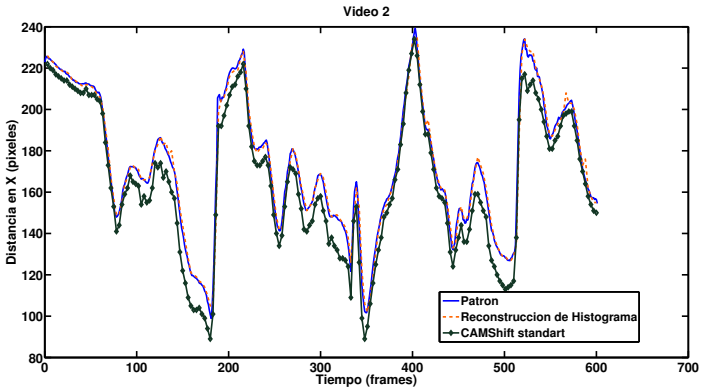
Se observa que el algoritmo presentado, debido a la optimización en procesamiento, tiene un tiempo de respuesta más eficiente (Cuadro I) con respecto a los algoritmos seguidores de objetos estándar. A su vez se observa que el tiempo de procesamiento es proporcional a la resolución de las imágenes procesadas, por lo que se puede analizar el límite de procesamiento conociendo dicha resolución.

En la Fig. 7 se pueden observar distintos puntos del recorrido del objeto en el video Seq A. Durante toda la ejecución del programa se detecta el objeto (pelota) a pesar de su variación en aceleración y el movimiento de objetos cercanos (pies).

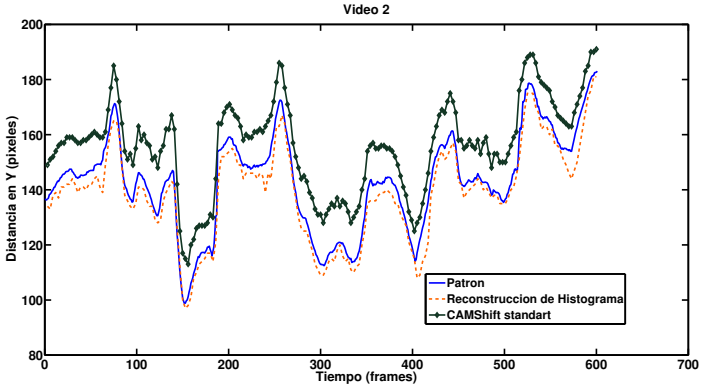
Figura 7: Secuencia de movimiento video Seq A.

Con el objetivo de analizar la eficiencia del algoritmo en las Fig. 8a y Fig. 8b se compara valores medidos (x_m,y_m) y los valores de posición patrón (x_p e y_p) en función de cada *frame* para el video Seq A (Fig. 7). Se comprueba que la posición del objeto obtenida por el algoritmo presenta diferencias respecto

a la posición real del mismo por debajo del 4 % (Cuadro II se presentan los errores). se puede observar claramente que la técnica de seguimiento **CAMShift** estándar tiene un error de detección mayor al calculado por el algoritmo presentado.



(a) Posición en x



(b) Posición en y

Figura 8: Comparación en la detección del objeto entre su posición real y estimada en x e y

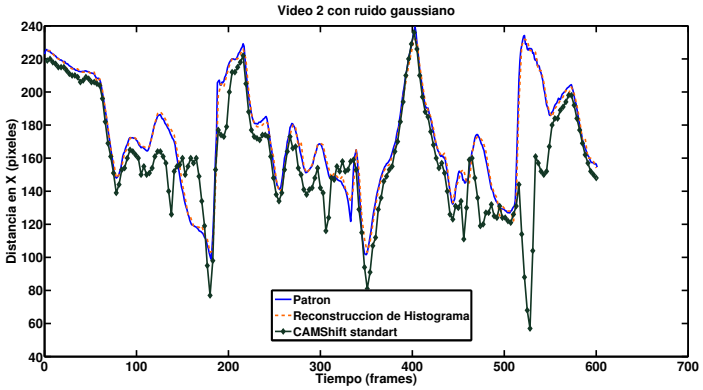
En las Fig. 9a y Fig. 9b se vuelve a realizar la comparación valores medidos contra valores patrones para el video Seq A con rudio adicionado. Se comprueba que la incorporación de ruido no interfiere en la detección del objeto y las diferencias con el partón son apenas mayores que sin ruido (Cuadro II se presentan los errores). En cambio el algoritmo **CAMShift** por momentos pierde al objeto (Fig. 9 *frames*: 138, 474 y 528) y la diferencia con el patrón es visiblemente mayor.

Se evaluó la distribución de error en pixeles para las dos coordenadas por separado y en conjunto (δ_x, δ_y y δ_{xy}), para el video con y sin ruido. En el Cuadro II se muestran los valores estadísticos para cada distribución y video.

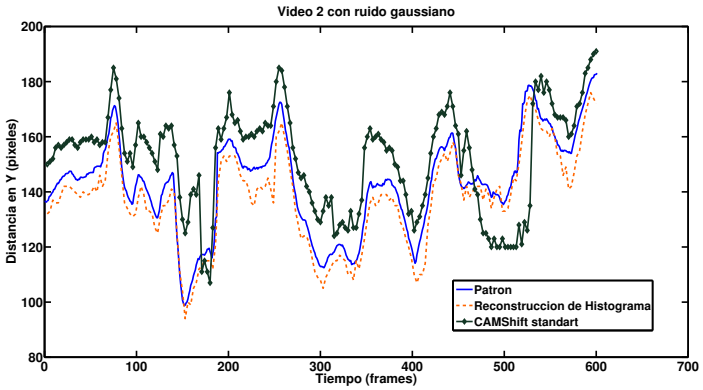
Buscando un nivel de confianza del 95 % obtenemos como resultado que en dicho intervalo tendremos errores menores a $\mu \pm 2\sigma$. Esto resulta δ_x menor a 2,4 % δ_y menor a 4 % y δ_{xy} menor a 2,9 %. Y al adicionar ruido el error no supera al 5 % en ninguna de sus coordenadas ni en su conjunto.

IV. Conclusiones y Futuros Trabajos

En este trabajo hemos presentado un algoritmo de seguimiento de objetos por medio de la reconstrucción de su histograma



(a) Posición en x



(b) Posición en y

Figura 9: Comparación en la detección del objeto entre su posición real y estimada en x e y

Video	δ	Media (μ)	Desvío estándar (σ)
Normal	δ_x	0.7 %	0.9 %
	δ_y	1.7 %	1.1 %
	δ_{xy}	1.3 %	0.8 %
Con Ruido	δ_x	0.8 %	0.9 %
	δ_y	2.3 %	1.3 %
	δ_{xy}	1.7 %	0.9 %

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

con una baja cantidad de puntos exploratorios (menor al 20 % del tamaño de la **ROI** del objeto a seguir). El algoritmo se adapta a las necesidades enunciadas en la introducción para los algoritmos de seguimiento de objetos basados en visión por computador y cumple con el objetivo planteado de optimizar tiempos de procesamiento sin introducir errores de posición de seguimiento.

Los ensayos experimentales arrojaron un tiempo de procesamiento igual 12ms para imágenes de alta definición (720x480) y tiempos entre 7.5ms y 3.3ms para imágenes de baja resolución, lo que supera en eficacia de procesamiento a los algoritmos estándares (Cuadro I) y permite procesar videos de incluso 83fps para alta resolución. A su vez, los errores de ubicación

del objeto (con un nivel de confianza del 95 %) no superan al 4 % un valor completamente aceptable para la mayoría de aplicaciones de seguimiento de objetos asistida por computador.

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, actualmente el algoritmo presentado pierde el objeto cuando el mismo se oculta en la imagen por ello será necesario incorporar una combinación con modelos probabilísticos para la estimación de la posición del objeto a seguir durante oclusiones o la selección automática de la **ROI** inicial.

Como técnica probabilísticas se puede considerar un filtro de partículas , el cual analiza la probabilidad de encontrar al objeto en un lugar determinado. El mayor costo computacional de este algoritmo como se observó en el Cuadro I impide ejecutar este tipo de algoritmo en el análisis de cada *frames*, pero si el mismo se aplica únicamente en el momento de perder el objeto por una larga oclusión hasta encontrarlo se logrará evitar las perdidas por largas oclusiones y el costo computacional no se verá afectada. A su vez con un previo conocimiento de la forma y/o color del objeto que se desea seguir se puede incluir un pre-procesamiento de la imagen inicial que detecte automáticamente el objeto que se desea seguir e indique la **ROI** inicial que actualmente es seleccionada de forma manual.

REFERENCIAS

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

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

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

[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] D. Comaniciu P. Meer, *Mean shift analysis and applications.* IEEE International Conference on Computer Vision (ICCV). Vol. 2. 1197–1203. 1999.

[13] D. Comaniciu P. Meer, *Mean shift: A robust approach toward feature space analysis.* IEEE Trans. Patt. Analy. Mach. Intell. 24, 5, 603–619. 2002.

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

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

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

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

[18] H. Tanizaki, *Non-gaussian state-space modeling of nonstationary time series.* J. Amer. Statist. Assoc. 82, 1032–1063. 1987.

[19] M. Isard A. Blake, *Condensation - conditional density propagation for visual tracking.* J.Comput. Vision 29, 1, 5–28. 1998.

[20] D. Mackay, *Introduction to Monte Carlo methods.* Learning in Graphical Models,M.I. Jordan, Ed. NATO Science Series. Kluwer Academic Press, 175–204. 1998.

[21] PaticleFltSrc, *Particle Filtering Source Code.* http://www-sigproc.eng.cam.ac.uk/smc/software.html.

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

[23] 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.

[24] 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.

[25] D. A. Klein D. Schulz S. Frintrop A. B. Cremers. *Adaptive real-time video tracking for arbitrary object.* Proc. of IROS, 2010. - http://www.iai.uni-bonn.de/ kleind/tracking/

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