

Illumination-invariant Object Tracking Using Optical Flow, Correlation Filtering and Color-Distribution-Based Image Masking

Jacoby Daniel Andres, Mechoulam Alan, Lambertucci Guido, Rodriguez Turco Martín, Londero Bonaparte Tomás
Buenos Aires Institute of Technology, Argentina

Abstract

Color distribution and correlation theory, the Shi-Tomasi algorithm, the Lucas-Kanade variant of Optical Flow and Kalman filtering were used synergistically in an attempt to produce a real-time, high-reliability object tracking algorithm that could be robust against illumination variance, partial or total occlusion and abrupt changes in trajectory even during total occlusion. A prototype for testing and measuring was programmed using Python and C/C++ libraries. **aca escribir un resumido sobre los resultados que obtengamos.**

I. INTRODUCTION

Una imagen puede ser interpretada como una función bidimensional $f(x, y)$, donde tanto x como y representan en un plano el espacio visualizado, mientras que la misma función $f(x, y)$ es la intensidad de la imagen bajo un punto dado. Cuando x , y y $f(x, y)$ son valores cuantizados y discretizados, la imagen se transforma en una imagen digital.

El procesamiento de dichas se define como el conjunto de técnicas aplicadas a estas imágenes, con el objetivo extraer información de ellas. Estas actividades cubren un campo que abarca un sin fin de aplicaciones, ya que se vale de máquinas capaces de detectar la totalidad del espectro electromagnético. Esto significa que se pueden obtener imágenes generadas por fuentes que captan información la cual para los humanos no se asocian con imágenes propiamente dichas, como lo son las ondas de radio, entre tantas otras.

Es posible considerar tres tipos de procesos computarizados en el procesamiento de imágenes, basándose en el nivel de tratamiento que se aplique, siendo así clasificados en bajo, medio y alto nivel. Los primeros incluyen actividades tales como reducción de ruido y aumento de contraste, tareas caracterizadas por el hecho de que tanto la entrada como la salida son imágenes. Las actividades de medio nivel de procesamiento incluyen trabajos de segmentación, es decir, identificar regiones u objetos dentro de las imágenes, descripción y clasificación de dichos elementos. Es así que esta categoría es destacada por sus salidas, ya que suelen ser información extraída de las imágenes a la entrada. Por último, los procesos de alto nivel se caracterizan por no solo reconocer objetos y analizarlos, sino también por darles un tratado normalmente asociado con la visión, tales así como “darles sentido” [1].

Dada una señal continua a la entrada del sistema, una imagen sufre de dos procesos claves: **cuantización** y **discretización**. Si bien ambos refieren a tomar variables continuas y almacenarlas en memoria como variables discretas, se realiza esta diferenciación entre ambas ya que la primera hace referencia a la amplitud de la señal mientras que la segunda a coordenadas, que para el caso del estudio de imágenes, se refiere a pixeles. Estos son el mínimo elemento que compone una imagen digital y se pueden pensar como un cuadrado de color uniforme.

Ejemplificando lo anterior, se toma una entrada al sistema, como puede ser la presentada en la Figura (1), la cual, como ya se ha mencionado, es continua en x , y y $f(x, y)$.

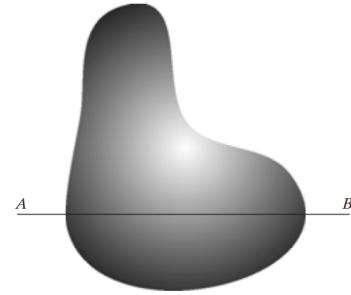


Figure 1: Entrada continua al sistema.

Por lo tanto, se deben tomar coordenadas finitas, por ejemplo, aquellas que se encuentran sobre la recta AB, y asignarle a cada una un valor dado de amplitud. En la Figura (2) se observa como una recta continua paralela al eje x (horizontal), la cual posee ciertas variaciones aleatorias dadas por el ruido existente, es dividida en una cierta cantidad de posiciones equiespaciadas (discretización), marcadas con cuadros blancos sobre la curva, asignándoles un nivel específico en la escala de grises (cuantización), marcado con una linea negra por la izquierda de dicha escala.

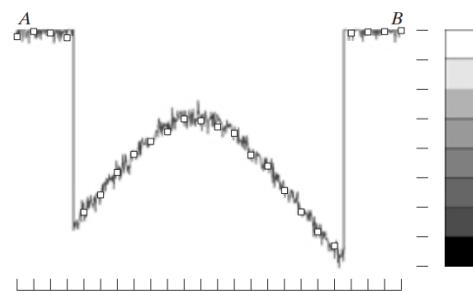


Figure 2: Amplitud de la escala de grises en la recta AB y muestreo de valores.

Realizando el mismo proceso para todos los niveles de discretización en el eje y (vertical), se obtiene finalmente una imagen digitalizada, la cual se la compara a continuación con la original [2].

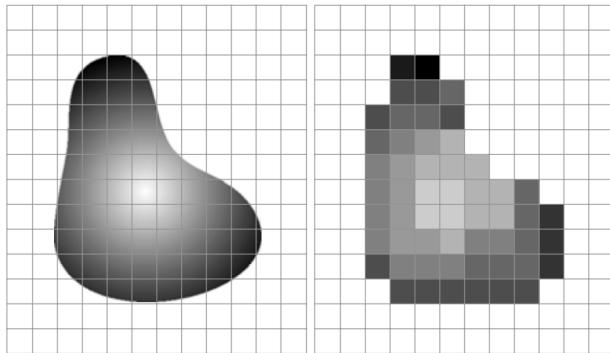


Figure 3: Imagen original comparada con la imagen digitalizada a procesar.

Este trabajo se centra en procesos de medio nivel. Dicha definición es muy amplia, por lo cual es necesario acotar este camino. Es por ello que se decidió focalizarse en el seguimiento de objetos en imágenes en movimiento. Se buscó que, dada ciertas condiciones iniciales conocidas (brindadas por el usuario) en una imagen en movimiento en tiempo real, tomar un conjunto de datos de x , y y $f(x, y)$ para así seleccionar un elemento y seguir su trayectoria a través del tiempo. Como hipótesis se plantea que:

- El objeto no cambiará rápidamente de color ni su iluminación o exposición-

Para esto, se utilizaron los algoritmos de Shi-Tomasi, Optical Flow de Lucas-Kanade asimismo como filtros de correlación, de Kalman y de probabilidad de distribución de color.

II. REVIEW OF LITERATURE

A. Optical Flow

El campo de movimiento de una imagen es el movimiento real del objeto en el espacio proyectado sobre el plano de la imagen. El Optical Flow (flujo óptico) se define como el flujo de la intensidad en escala de grises en el plano de la imagen, a medida que evoluciona en el tiempo. También se puede interpretar el flujo de la imagen como el movimiento aparente de esta basado en la percepción visual, teniendo dimensión de velocidad $\vec{V} = (V_x, V_y)$. Si el Optical Flow se determina de dos imágenes consecutivas, aparece un vector de desplazamiento \vec{d} de las cualidades elegidas, entre el cuadro n y el $n+1$.

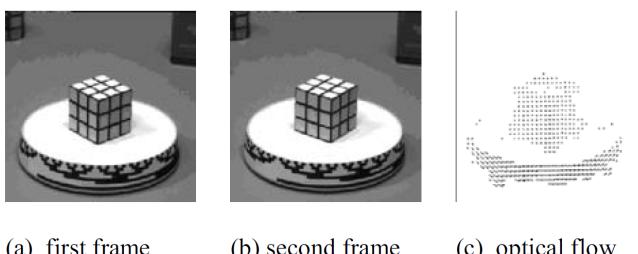


Figure 4: Cubo de rubik rotando en una mesa.

Para la implementación del algoritmo existen varios caminos, como lo es, por ejemplo, el basado en cálculo de

gradientes de la imagen total. Existe otro que utiliza solo ciertos puntos que se determinan constantes de la imagen. El presente se emplea el método piramidal de Lucas-Kanade.

1) *Lucas-Kanade*: Este algoritmo consta del uso de información obtenida a partir de la intensidad del gradiente espacial para buscar la posición que mejor se acomoda a una imagen en movimiento [4] [6]. Algunas de las hipótesis que postula Lucas-Kanade son:

- Los movimientos entre cuadros consecutivos son pequeños. Tan pequeños como un pixel.
- La intensidad de los objetos se mantiene constante cuadro a cuadro.

Este algoritmo se basa en el principio de “divide y conquistarás” al realizar la tarea de detectar movimiento en toda la imagen en problemas más sencillos. El proceso consta de dividir la pantalla en un árbol cuaternario. Esto lo hace debido a que se requiere que las hipótesis de Lucas-Kanade deben cumplirse, se toma una pequeña parte de la imagen, del menor tamaño posible, donde tienden a cumplirse las condiciones necesarias del algoritmo. De allí se calcula recursivamente cada partición hasta el primer nivel.

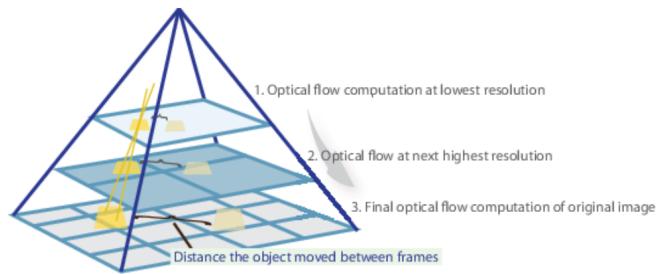


Figure 5: Cálculo del Optical Flow.

B. Shi-Tomasi

Por su parte, el algoritmo de Shi-Tomasi se basa en el algoritmo de Harris para la detección de bordes. El funcionamiento del algoritmo se basa en asignarle un valor a cada pixel del bitmap recibido, y si el valor del pixel es mayor a cierto límite se determina que este corresponde a una esquina. El valor asignado al pixel se obtiene utilizando dos autovalores de una matriz Z . Una función recibe estos autovalores, para poder manipularlos y devolver el valor. La diferenciación entre el algoritmo de Harris y el de Shi-Tomasi, es que mientras Harris utiliza la función mencionada previamente, Shi-Tomasi decide utilizar únicamente los autovalores [9].

Los valores de cada pixel se determinan de la siguiente manera:

$$R = \det(Z) - k(\text{trace}Z)^2 \quad (1)$$

$$\det(Z) = \lambda_1 \cdot \lambda_2 \quad (2)$$

$$\text{trace}Z = \lambda_1 + \lambda_2 \quad (3)$$

Figure 6: Valor de pixel para algoritmo de Harris

$$R = \min(\lambda_1, \lambda_2) \quad (4)$$

Figure 7: Valor de pixel para algoritmo de Shi-Tomasi

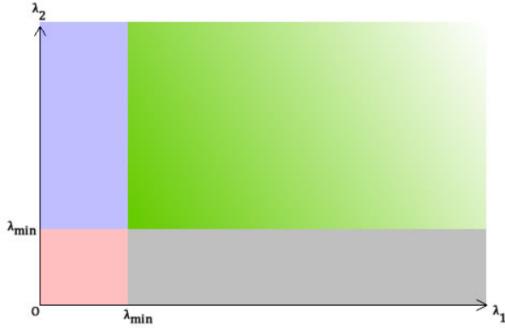


Figure 8: Detección de bordes Shi-Tomasi.

- La zona verde corresponde a ambos λ_1 y λ_2 mayores al valor límite, por lo que estos píxeles son tomados como esquina.
- La zona azul y gris corresponden a que uno de los autovalores es menor al límite.
- La zona roja corresponden a ambos autovalores menores al mínimo.

C. Filtro de Kalman

El filtro de Kalman es un filtro recursivo que busca estimar el estado de un sistema dinámico lineal discretizado de dimensión n mediante una serie mediciones con ruido a partir de la descripción del modelo físico que rige las variables del vector de estado [7], [8].

1) *Planteo del problema:* Dado un proceso estocástico lineal en tiempo discreto definido por

$$x_k = Ax_{k-1} + w_{k-1}^1 \quad (5)$$

donde $x_k \in \mathcal{R}^N$ y siendo una medición en tiempo k definida por

$$z_k = Hx_k + v_k \quad (6)$$

con $z_k \in \mathcal{R}^m$, donde w_k y v_k son variables aleatorias que representan el ruido del proceso y de observación respectivamente, las cuales se asumen que son independientes entre sí y con distribuciones de probabilidad normales multivariadas definidas como

$$p(w) \sim N(0, Q)$$

$$p(v) \sim N(0, R)$$

donde Q y R son las matrices de covarianza del ruido del proceso y ruido de medición respectivamente, las cuales se

¹Se omitió por simplicidad la función de control.

asumen constantes; donde la matriz de transición de estados A de dimensión $n \times n$ —siendo n la dimensión de estados dinámicos del modelo— define la relación entre estados del paso $k-1$ al paso k sin contar el ruido del proceso; donde la matriz de transición de observación H de dimensión $m \times n$ —siendo m la dimensión del vector de medición— fija la relación entre el espacio de los observables medidos y el espacio de las variables del vector de estados, sin contar el ruido de medición; y finalmente donde el estado del filtro puede representarse como $\hat{x}_{k|k}$ y $P_{k|k}$, siendo estos el estimador del vector de estados a posteriori dadas las mediciones hasta un tiempo k , y el estimador de la matriz de covariaza a posteriori dadas las mediciones hasta un tiempo k respectivamente. Se puede definir la operación del filtro de Kalman separando a esta en dos fases: la predicción, y la corrección.

2) Ecuaciones de predicción:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} \quad (7)$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \quad (8)$$

3) Ecuaciones de observación:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k \quad (9)$$

$$P_{k|k} = (\mathbb{I} - K_k H)P_{k|k-1} \quad (10)$$

donde,

$$y_k = z_k - H\hat{x}_{k|k-1} \quad (11)$$

$$S_k = HP_{k|k-1}H^T + R_k \quad (12)$$

$$K_k = P_{k|k-1}H^T S_k^{-1} \quad (13)$$

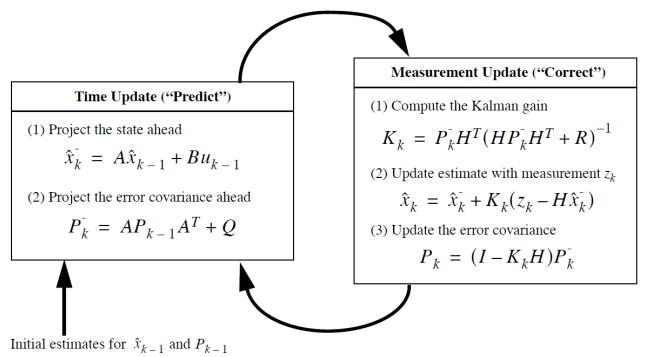


Figure 9: Funcionamiento completo del filtro Kalman simple [8].

En la Figura (10) se puede observar el funcionamiento del algoritmo con un vector de estados dinámicos de dimensión dos compuesto por las coordenadas (x, y) sobre el plano cartesiano. Realizando mediciones periódicamente, se comparan dos curvas. Por un lado, una senoidal con un ruido gaussiano montado sobre ella, simulando una serie de mediciones con ruido, y por el otro lado, la estimación obtenida a partir del filtro desarrollado.

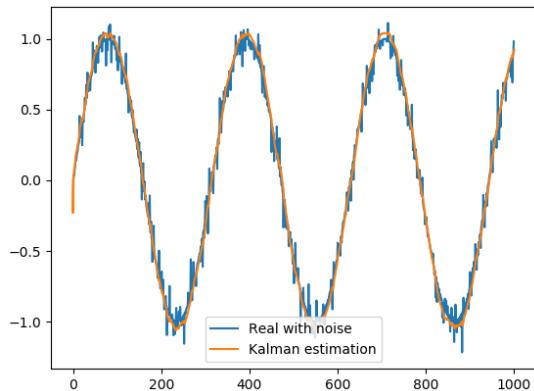


Figure 10: Seno con ruido comparada con estimación de Kalman.

D. Filtros de Correlación:

El filtro de correlación o también conocido como “Matched Filter” [10] en inglés, se obtiene al calcular la correlación de una señal conocida, o “kernel”, con una señal desconocida con el propósito de detectar este kernel en la última señal. Esto último es equivalente a convolucionar la señal desconocida con la inversión temporal del conjugado de la conocida. Este filtro es el óptimo lineal que maximiza la relación señal a ruido (SNR) en presencia de ruido aditivo.

Existen diversas maneras de calcular esta correlación, los dos métodos propuestos fueron:

- Cuadrados mínimos: Se basa en el cálculo de cuadrados mínimos entre el kernel y la imagen mientras se realiza la convolución y corresponde a la siguiente ecuación

$$R(x, y) = \frac{\sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}} \quad (14)$$

- Estimación de la correlación: Se calcula la correlación por definición del frame con respecto al kernel a medida que se desliza por la imagen, le corresponde la siguiente fórmula

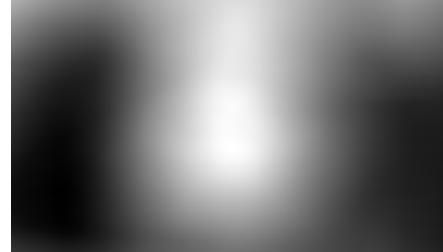
$$R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}} \quad (15)$$

Donde R es la función de resultado, I la señal desconocida, y T el template o kernel.

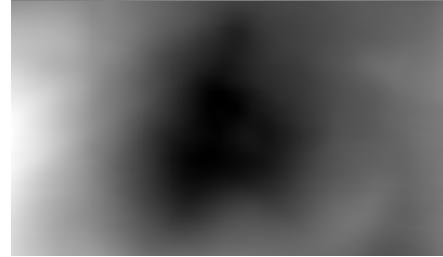
En la Figura (11) se puede observar una imagen y el resultado del cálculo de correlación utilizando la correlación por cuadrados mínimos y por correlación.



(a) Imagen Fuente.



(b) Correlación utilizando definición.



(c) Correlación utilizando cuadrados mínimos.



(d) Kernel.

Figure 11: Aplicación del algoritmo de correlación.

Se puede apreciar como el punto más oscuro de la pantalla indica el punto de mayor correlación entre la imagen fuente y el kernel seleccionado para el algoritmo de cuadrados mínimos (Figura (11c)) mientras que el mas claro indica la mayor correlación para el otro (Figura (11b)).

E. Espacios de Color HSV y CIE-L*a*b*:

El espacio de color HSV, o hue-saturation-value logra representar a un color mediante tres ejes distintos. A diferencia del espacio de colores usual, RGB, que describe un color mediante sus componentes de rojo, verde y azul, en el espacio HSV se describe mediante la oscuridad/luminosidad del color, al cual se le asigna la componente value; la cantidad de pigmentación o pureza del color, al cual se le asigna la componente de saturación; y finalmente la componente de hue, que describe el *chroma* según la teoría de colores, es decir la posición en la rueda de colores.

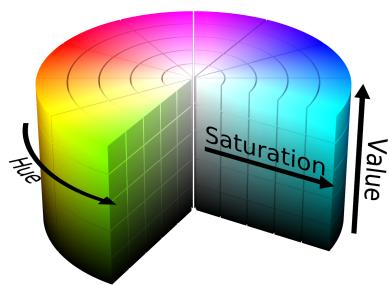


Figure 12: Espacio de colores HSV representado como un cilindro.

Este espacio se puede representar como un cilindro, como se demuestra en la Figura (12). Si bien la componente de value representa la oscuridad/luminosidad de un color, este modelo es incorrecto debido a que presenta uniformidades. Es por esto, que para obtener una verdadera representación de luminosidad se utiliza el espacio Lab. El espacio de color L*a*b* de la comisión internacional en iluminación o CIE por sus siglas en francés es un espacio de color perceptualmente uniforme que representa a un color mediante tres parámetros: lightness, a y b, siendo lightness un modelo correctamente calculado e uniforme de la luminosidad de un color, y a y b las componentes de rojo/verde y amarillo/azul del color respectivamente².

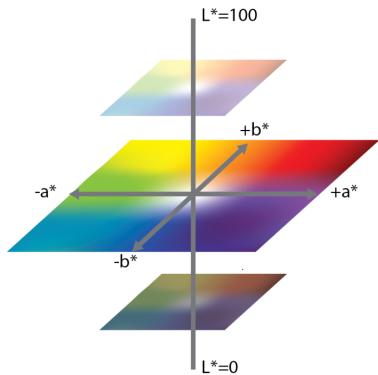
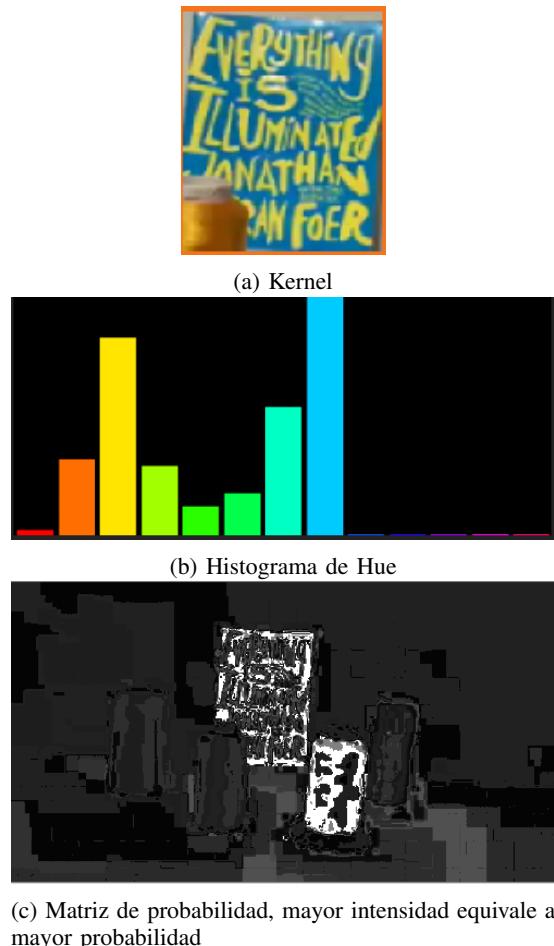


Figure 13: Espacio de colores CIE-Lab en coordenadas cartesianas.

F. Filtros basados en distribución de probabilidad de Hue:

Los filtros de color tradicionales usualmente se basan en la detección de un color específico con la adición de una cierta tolerancia para poder incluir una gama de colores más amplia. Sin embargo dicha elección resulta ser demasiado restrictiva cuando aparecen sombras o cambios de iluminación. Es por eso que una alternativa más general es la de utilizar máscaras que permitan el paso de ciertos pixeles según una función de masa de probabilidad dada. Dicha distribución se computa usando el objeto que deseamos seguir. En la Figura (14a) se observa un ejemplo de una selección realizada por el usuario y en la Figura (14b) su distribución asociada. Para obtenerla

se transforma la selección al espacio de color HSV y se aplica un pre-filtrado del cual obtendremos las componentes menos ruidosas de la imagen. El espacio HSV mismo nos permite independizarnos de la variación de brillo dentro de la escena y concentrarnos únicamente en la componente de hue, cuyo valor nos indicara a qué gama de colores pertenece el objetivo.



(c) Matriz de probabilidad, mayor intensidad equivale a mayor probabilidad

La Figura (14c) ilustra el proceso de asignar a cada pixel de la imagen la correspondiente probabilidad de ocurrencia de hue según indique el histograma. Cabe destacar que cada bin del histograma engloba un cierto rango de valores de la componente de hue. Para poder traducir estas probabilidades a valores de intensidad de pixel se realiza un mapeo lineal $[0, 1] \rightarrow [0, 255]$ para poder vizualizarlas. En las secciones posteriores veremos ciertas mejoras que se incorporaron para conseguir una máscara que permita aislar al objeto con más definición.

III. METHODS AND RESULTS

A. First Sketch

From a selection of the video provided by the user, the best characteristics of the desired object are obtained, using the Shi-Tomasi algorithm.

Afterwards, through Lucas-Kanade's Optical Flow, the evolution of these features over time is obtained. The center of mass μ and the center of inertia σ are calculated from this cluster of points. The points that lie at a distance greater than $k_\sigma\sigma$ with respect to the center of mass are rejected, to then

²Imagen atribuida en <https://opentextbc.ca/graphicdesign/chapter/colour-science/>

calculate said center again, which is the one that is introduced in the Kalman filter as position measurement.

The dimension of the state vector of the Kalman filter is four, having both horizontal and vertical position and velocity. The dimension of the observation vector is two, since only the horizontal and vertical position of the center of mass is measured. The matrix A is defined by the equations that characterize the motion of the object to be tracked, which, for simplicity and both good results, were considered as the kinematic equations of MRU, obtaining thus

$$\begin{pmatrix} x_{k-1} + dt \cdot v_{x_{k-1}} \\ y_{k-1} + dt \cdot v_{y_{k-1}} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \end{pmatrix} \quad (16)$$

where,

$$A = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (17)$$

Then, the observation or measurement transition matrix is defined by the transformation of the observation space to the real space of the filter state vector. As the observables to be measured are part of the real space of the state vector and are only the positions, the matrix is defined in a simple way.

$$\begin{pmatrix} z_{x_k} \\ z_{y_k} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ y_k \\ v_{x_k} \\ v_{y_k} \end{pmatrix} \quad (18)$$

where,

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad (19)$$

The process covariance matrix Q and measurement covariance matrix R are square by definition, of dimension four and two respectively. Furthermore, they are of the form $\mathbb{I} \cdot \xi$ since independence of the random variables is assumed for simplicity, leaving

$$Q = \begin{pmatrix} \xi_Q & 0 & 0 & 0 \\ 0 & \xi_Q & 0 & 0 \\ 0 & 0 & \xi_Q & 0 \\ 0 & 0 & 0 & \xi_Q \end{pmatrix} \quad (20)$$

$$R = \begin{pmatrix} \xi_R & 0 \\ 0 & \xi_R \end{pmatrix} \quad (21)$$

Where ξ_Q and ξ_R are the variances of the process and measurement respectively.

Finally, $\hat{x}_{k|k}$ and $P_{k|k}$ take the form

$$\hat{x}_{k|k} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{v}_x \\ \hat{v}_y \end{pmatrix} \quad (22)$$

$$P_{k|k} = \begin{pmatrix} \gamma_x & 0 & 0 & 0 \\ 0 & \gamma_y & 0 & 0 \\ 0 & 0 & \gamma_{v_x} & 0 \\ 0 & 0 & 0 & \gamma_{v_y} \end{pmatrix} \quad (23)$$

Γ_i being the variance of each state variable.

The algorithm explained until now has big problems. For example, given two objects A and B , where A is the object to follow:

- Since the Shi-Tomasi algorithm tends to place features on well-defined edges, and these are usually the outer edges of an object, if the object to track A passes in front of B with well-defined edges, the Features of the Optical Flow algorithm tend to stick to the edges of B no matter how long it is still.
- If the object A passes now behind the immobile object B , all the features on the object to be followed are attached to the object B . This poses a much bigger problem than the previous one, given that the estimation of the Kalman filter cannot be used reliably, because when the features stay still and attached to the outer edges of B , the position is still being measured as if there were no problem and those measurements are continually being delivered to the Kalman filter. This occasionates that the kalman estimate will remain attached to the B object instead of continue through the estimated trajectory of A .

B. Second Sketch

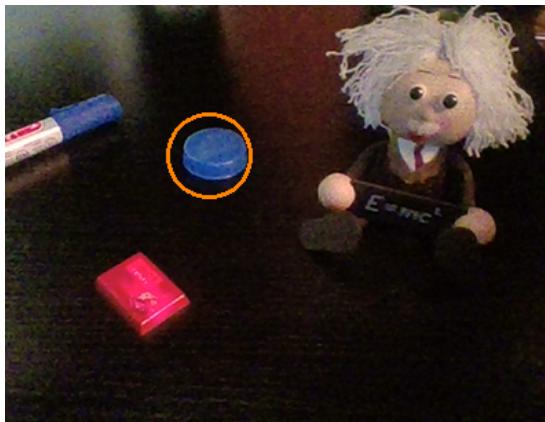
1) *HSV-Based Color Masking*: The proposed improvement to overcome the previous problems consists of adding a color filter to differentiate between A and B by their color. The implementation of this color filter begins the moment the user selects the area to follow. The median of the color of the chosen area is calculated, and that color is transformed from RGB to HSV format, or Hue-Saturation-Value, posing an easier way of defining a mask this way.

Given x and y as the coordinates of the pixels of the image, \vec{R}_k , \vec{G}_k and \vec{B}_k as the vectors containing the pixels of the initial selection, $\text{median}(\vec{R}_k, \vec{G}_k, \vec{B}_k) = (m_R, m_G, m_B)$ as the vector that defines the median of the color of the selection, HSV as the transformation operator between RGB and HSV, and $\text{HSV}(m_R, m_G, m_B) = (m_H, m_S, m_V)$ as the vector that defines the messdian of the color of the selection in the HSV space, the color mask is then defined as

$$f(x, y) = (f_h(x, y), f_s(x, y), f_v(x, y)) \quad (24)$$

$$f_u(x, y) = \begin{cases} 1 & -\Lambda_u < u_{x,y} - m_u < \Lambda_u \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

where Λ is the limit for each u property, these being Hue, Saturation or Value. Once the mask $f(x, y)$ is obtained, the AND operation is carried out between the mask and the original image, obtaining a null intensity in the pixels of the image that do not possess a color close to the calculated median. This can be seen in the Figures (15a) and (15b).



(a) Screencap before applying the color mask to follow the blue cap.



(b) Screencap after applying the color mask. It is with this new image instead of the old one that the algorithm operates.

Figure 15: Comparison between before and after applying the color mask.

This color filtering not only has great synergy with the Shi-Tomasi feature obtaining method, since this frame has much better quality edges, but it also has great synergy with the Optical Flow method, due to the fact that as fewer edges are seen in the modified image, the probability that the features are all contained within or on the edges of the object to be tracked increases notably.

2) Search for Lost Object: However, the Kalman filter cannot estimate the position of the object indefinitely. This is why when the object is lost, that is, no more features are detected, the algorithm enters another phase. In this phase, a search for the object is performed in each frame of the video based on the estimation of the Kalman filter and the original size of the selected area. The height and width of the search area are increased by a constant factor for each frame in which the search fails. This search is based on applying the Shi-Tomasi algorithm on the mentioned area.

3) Periodic Correction of the Characteristic Points: Finally, to reinforce the tracking algorithm, a periodic recalculation of the features was also implemented with the Shi-Tomasi method. We start from the mean of the cluster of features, removing the outliers, recalculating the mean again and using this position to apply the Shi-Tomasi algorithm, in the same way as it is done on the first selection provided by the user. All of this allows to even further raise the reliability



(a) Normal operation of the prototype program where an orange circle is used to indicate the estimated position.



(b) Actual vision of the algorithm after color masking and indicating the optical flow points. It can be seen how color masking synergically enhances the reliability of the Shi-Tomasi and Optical Flow algorithms.

Figure 16: Aftermath of selecting an object both in the back-end and front-end of the algorithm.

of the tracking. However, there are still more problems.

4) Fail Scenarios: Algunos casos donde puede fallar hasta ahora el Tracker son:

- El cambio de trayectoria de un objeto una vez que desaparece, provocando una falla en la predicción.
- Interacción entre el objeto trackeado y un objeto de características similares, causando un error de detección
- Cambios bruscos de iluminación o en color del objeto.

Having improved the reliability since the last sketch considerably, let again be the objects *A* and *B*, where *A* is the object to follow:

- If *A* and *B* are of different colors, with the color filter, the algorithm never detects that there is a *B* object, so tracking it is impossible. Consequently, only *A* is followed and there is no problem of *A* passing in front of *B*.
- If *A* and *B* are of different colors, and *A* now passes behind *B*, the features are now not attached to the *B* object, but the algorithm sees that the area of *A* gets smaller as it is hidden by *B* instead. As the features have no edges to follow, they disappear, so the Kalman filter receives no new measurements and it correctly estimates the position of *A* and long as it has not undergone any acceleration while it was hidden.

C. Último diseño

En la última iteración del proceso de diseño se sacrificó bajo costo computacional por una mayor robustez en el

algoritmo, reduciendo significativamente las probabilidades de:

- Error tipo I: no se encuentra el objeto a seguir por más que sea visible.
- Error tipo II: se detecta un objeto por más que el objeto a seguir no se encuentre en pantalla.
- Error tipo III: se detecta a otro objeto por más que el objeto a seguir este visible.

mediante el uso de *Matched Filters*. Además, se logró obtener una gran robustez frente a cambios en la iluminación utilizando un algoritmo alternativo de enmascaramiento basado en la distribución de hue del objeto a seguir. Por último, se desarrolló una interfaz gráfica donde se puede cargar un video o bien utilizar una cámara en tiempo real, seguir varios objetos a la vez, modificar cualquier parámetro de cualquiera de los algoritmos utilizados en tiempo real e incluso se desarrolló un método de optimización automática de los parámetros del algoritmo de enmascaramiento. En la Figura (18) se puede observar una captura de la interfaz gráfica desarrollada y en la Figura (17) se puede observar el diagrama de flujo entero del algoritmo de seguimiento desarrollado. La Figura (20) aporta una visión de cómo se organiza el programa mencionado.

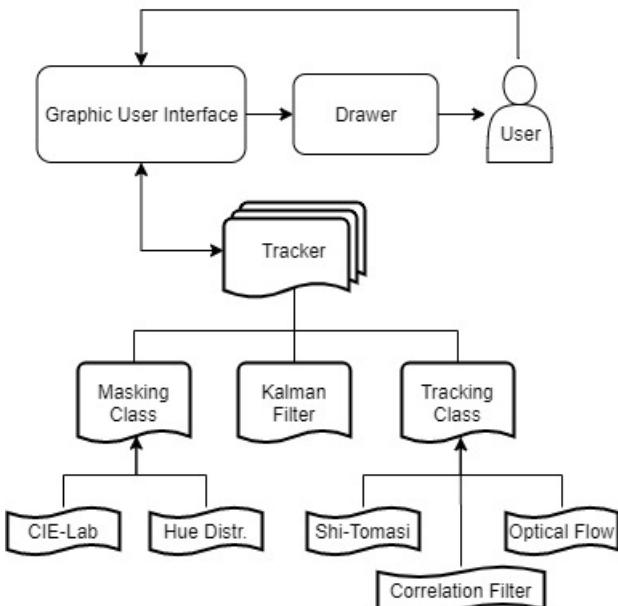


Figure 20: Diagrama de Clases del programa utilizado para presentar la funcionalidad del algoritmo desarrollado.

1) *Enmascaramiento Basado en CIE-Lab*: El diseño anterior poseía un algoritmo de enmascaramiento basado en el espacio de color HSV. Si bien este funcionaba correctamente, este espacio de color fue reemplazado por CIE-Lab debido a que este posee un solo parámetro que caracteriza la iluminación de un objeto, es decir, con un valor nulo de L se obtiene el negro, mientras que con un valor máximo de L se obtiene blanco, pasando entre medio de esos dos valores por el color representado por a y b. En el espacio HSV, si se utiliza el máximo valor de la componente value, no necesariamente se obtiene blanco, sino que depende del valor de saturación elegido, mientras que si value es nulo, siempre se obtiene negro. Poseer un solo parámetro que

correctamente caracterice la iluminación de un objeto brinda una mejor optimización de los valores de semiamplitudes L-a-b utilizados en el enmascaramiento.

La generación de la máscara se realiza de la misma forma que en el diseño anterior, con el único agregado de un suavizamiento de la máscara computada previo a la utilización de esta para el filtrado de la imagen. Esto se utiliza debido a que generalmente los bordes de un objeto poseen mayor o menor iluminación dependiendo de dónde está la fuente de luz más cercana. Estas diferencias en la iluminación pueden ocasionar que un objeto de similar color al de interés sea vez correctamente excluido de la imagen a excepción de sus bordes, lo cual genera problemas al pasar por delante o por detrás del objeto a seguir por causas detalladas anteriormente.



Figure 21: Ejemplo del uso del suavizado de la máscara de color. La segunda foto es la máscara previo al suavizado y la tercera foto es posterior al suavizado.

Sin embargo, si bien la utilización del espacio de color CIE-Lab subsanó varios problemas, el enmascaramiento de la imagen tomando en cuenta solamente uno de los varios colores del objeto a seguir falla cuando se encuentra en pantalla un objeto del mismo color que el utilizado para la máscara. Para lidiar con este problema, se decidió desarrollar un método de enmascaramiento que tenga en cuenta todos los colores del objeto a seguir.

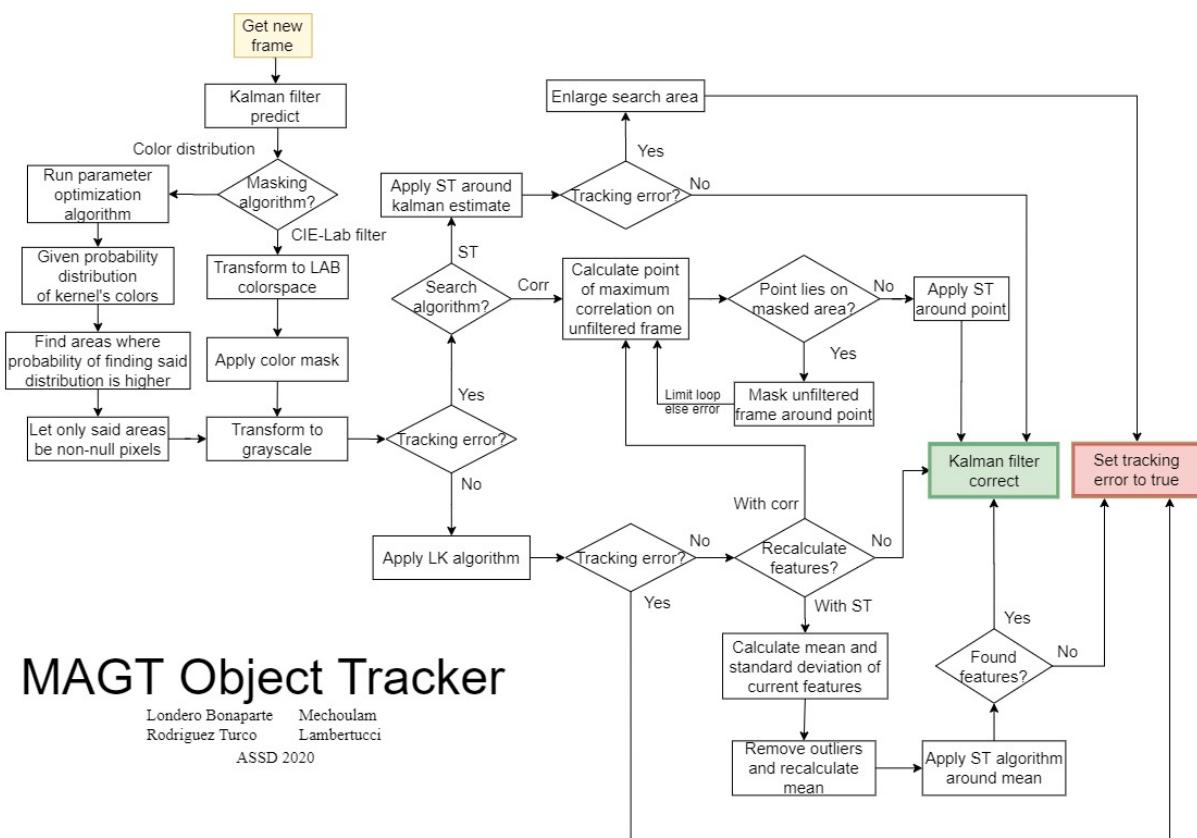


Figure 17: Flow-chart del algoritmo.

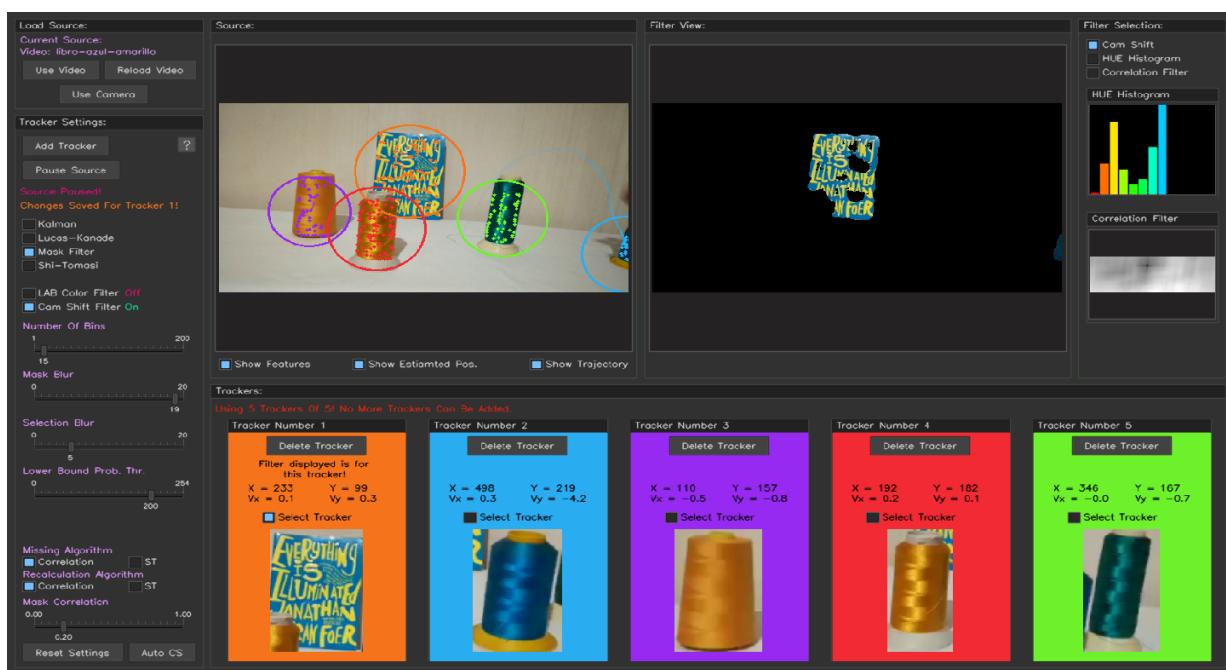


Figure 18: Captura de pantalla de la interfaz gráfica desarrollada.

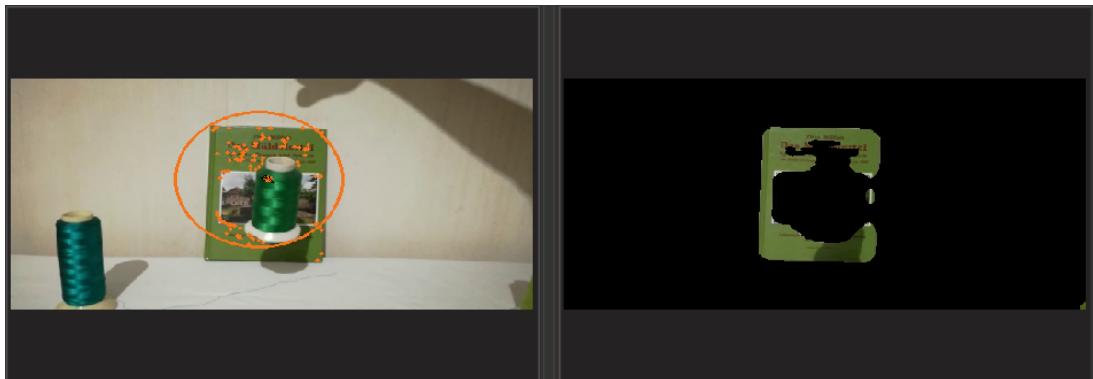
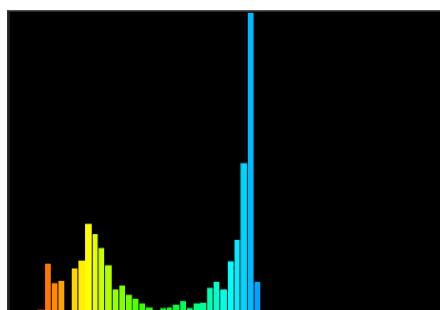
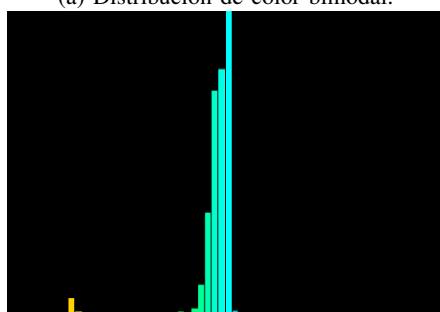


Figure 19: Primer ejemplo del algoritmo de enmascaramiento basado en histogramas de hue. En este ejemplo se puede notar la eficiencia conseguida en la máscara, debido a que por más que se encuentren varios objetos en escena de un color muy similar al objeto de interés, estos son excluidos por la máscara. Además, y de mayor importancia, se puede notar como el algoritmo de enmascaramiento no excluye la zona del libro bajo sombra. Esto muestra la gran robustez frente a cambios en la iluminación del algoritmo.

2) *Enmascaramiento Basado en Histogramas de Hue:* Para este método de enmascaramiento, el espacio de color HSV fue utilizado debido a que un solo parámetro es el encargado de denotar el color, lo cual genera que la distribución de color de un objeto sea una variable aleatoria unidimensional y no bidimensional como lo sería en el espacio de color CIE-Lab. Aclarado esto, los objetos pueden adquirir perfiles de color complejos, sus colores pueden estar distribuidos entorno a un solo valor de hue dominante, o pueden tener más de uno en su composición, obteniendo distribuciones unimodales o multimodales, respectivamente.



(a) Distribución de color bimodal.



(b) Distribución de color unimodal.

Para obtener una máscara que podamos utilizar debemos establecer un límite para así dejar pasar solamente aquellos pixeles cuyo hue tenga una alta probabilidad de pertenecer al objeto seleccionado.

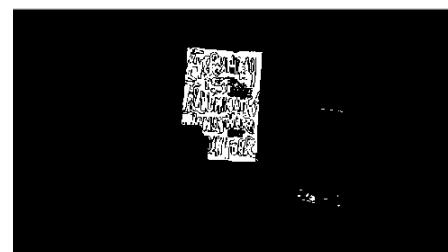


Figure 23: Máscara obtenida luego de quedarnos con los pixeles más probables.

Debemos notar en la Figura (23) que se pueden observar claramente las letras y el fondo de la tapa del libro. Como se puede apreciar, el histograma del ejemplo presenta una distribución bimodal la cual ha sido exitosamente capturada por el método de filtrado.



Figure 24: Aplicamos el equivalente a un filtro pasabajos para disminuir el ruido.

Sin embargo, la Figura (23) presenta ciertos pixeles que pueden interferir con las features calculadas mediante Shi-Tomasi. Es por eso que utilizamos la operación de suavizado para eliminar el ruido. Esto trae acarreado otro beneficio, el interior de la máscara se vuelve más uniforme y permisivo para los pixeles que pertenecen al interior del objeto.

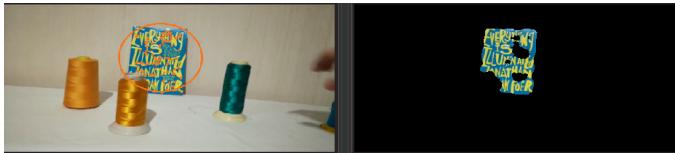


Figure 25: Comparación antes y después del filtrado.

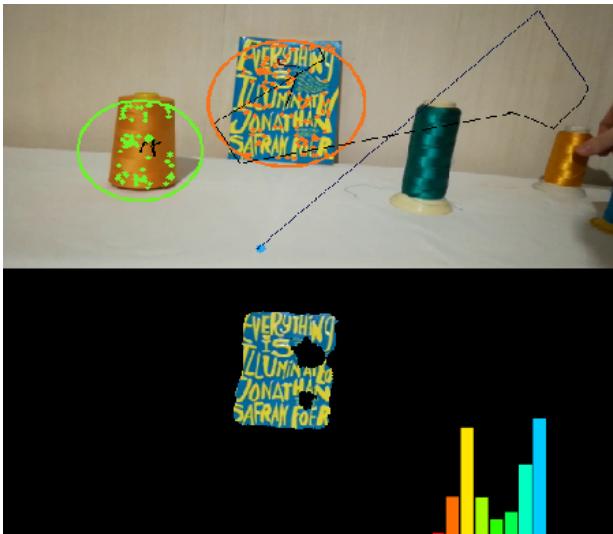


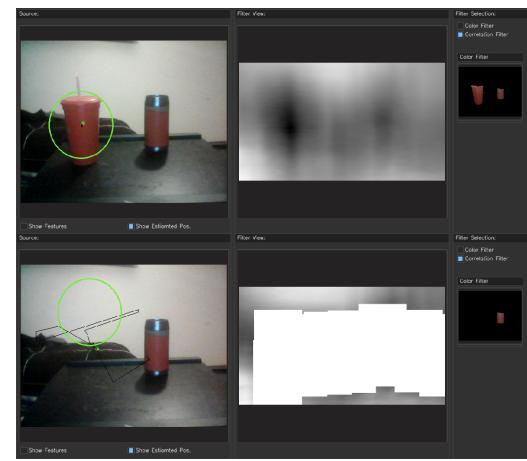
Figure 26: Segundo ejemplo del algoritmo de enmascaramiento basado en camshift. Se observa como el enmascaramiento se realiza a partir de la distribución de colores y no de los colores por separado, dado que por más que los carretes de hilo azules y amarillos posean colores dentro de la selección original (el libro), estos son excluidos por el algoritmo de enmascaramiento. Se editó a posteriori el histograma de colores de la selección sobre la imagen ya filtrada.

3) Algoritmo correlación: Para el algoritmo de correlación no solo se utiliza el filtro explicado en la Sección II-D, sino que además se trata tanto al frame como al kernel, aplicando una transformación al espacio HSV. La transformación de RGB a HSV posee un requerimiento computacional significativamente menor a la transformación de RGB a CIE-Lab y por eso se utilizó nuevamente el espacio HSV, dado que este filtrado sería complementario al método de la correlación y no lo más importante en el algoritmo. El filtrado se realizó bajo las siguientes especificaciones:

$$MSK = H \in (0, 180) \quad S \in (50, 255) \quad V \in (32, 255) \quad (26)$$

Este filtrado del kernel logra eliminar aquellos píxeles de muy baja saturación (grises) y de muy bajo valor (negros), lo cual se comprobó experimentalmente que aumenta considerablemente la precisión en el algoritmo. Luego, se utiliza el filtro de correlación y se obtiene el punto donde esta es mayor. Se toma una sección alrededor de este punto proporcional a la selección inicial y se compara con el frame original. Si no comparten las características de color que son calculadas por el enmascaramiento de color o distribución de color, este punto de correlación es desechar. Esto se logra aplicando una máscara en la sección del frame de mayor

correlación, realizando sobre este nuevamente el cálculo de máxima correlación. Este proceso se realiza unas pocas veces con el propósito de no comprometer la velocidad de procesamiento entre frames. En caso de no encontrar al objeto se considera que hubo un error de trackeo (occlusión del objeto).



(a) Correlación con máscara ante occlusión del objeto



(b) Kernel

Figure 27: Aplicación del algoritmo de correlación con máscara.

Este algoritmo es utilizado en dos escenarios. En el primero se utiliza para la búsqueda de un objeto perdido. En caso de occlusión se utiliza el algoritmo para encontrar nuevamente el objeto, siendo efectivo siempre que no haya objetos que compartan tanto forma y color con el de interés. Por otro lado, en caso de que así sea y el objeto de interés se encuentre oculto, es posible que el algoritmo devuelva un falso positivo, siendo conveniente utilizar el alternativo de Shi-Tomasi y medias móviles, dado que el tener información de la posición previa para el tracking resulta fundamental para poder distinguir entre estos.

El otro escenario es utilizar la correlación para el recálculo periódico de features, esto es beneficioso cuando se da la interacción de dos objetos *A* y *B* siendo *A* el de interés y *B* uno similar. Si *B* pasa por delante de *A* quitandole las features, el cálculo periódico por correlación puede detectar que se está siguiendo a *B* y no a *A*, por lo que corrige este error. Una situación similar a esta no puede ser resuelta por el algoritmo de Shi-Tomasi con medias móviles.

4) Optimización del Código: Se realizó una optimización del código, quitando redundancias y utilizando únicamente librerías que son corridas en C++ con la finalidad de que el procesamiento en tiempo real sea eficiente. Se optó por un enfoque OOP, esto permitió escalar el proyecto y obtener una mayor modularidad.

Además se diseñó una GUI con la capacidad de trackeo

multi-objeto, personalización de algoritmos, contando con la opción de variar algoritmos tanto de búsqueda como de recálculo entre medias móviles y correlación, al igual que filtrado por color o camshift. Existen la opción de cambiar las especificaciones de los filtros en tiempo real para obtener el filtro óptimo o que un algoritmo de optimización las calcule automáticamente. También se permiten diversas pantallas de debug, es decir, observar los distintos filtros a la vez, al igual que una estimación de la posición y velocidad del objeto. Finalmente cuenta con la capacidad de utilizar tanto un video en tiempo real como un video pre-existente.

5) *Escenarios de fallas:* Si bien se han minimizado las probabilidades de cometer errores de tipo I y tipo II, aún se plantean los siguientes problemas:

- El algoritmo se encuentra en su situación más frágil al haber un objeto idéntico o de gran similaridad tanto en color como en forma con el que se quiere seguir, debido a que ni el filtro de distribución de color ni el filtro de correlación lograrán distinguir entre ambos. En este caso la única distinción entre ambos será la retención de esquinas características al objeto correcto proporcionado por el algoritmo de Optical Flow.
- La correcta elección de parámetros de los algoritmos encargados de la fase de enmascaramiento para cada situación proporcionan una gran robustez frente a cambios en la iluminación. Sin embargo, cuanto más robusto frente a cambios en la iluminación es el algoritmo, mayor será la probabilidad de cometer errores del tipo II (seguir a un objeto incorrecto) debido a que el enmascaramiento tendrá una mayor permisividad.

IV. DISCUSSION

Se desarrolló un algoritmo capaz de seguir a un objeto frente a occlusiones parciales y totales, cambios en la iluminación y cambios bruscos en su trayectoria. Se logró minimizar la probabilidad de cometer error de tipo I, tipo II y tipo III mencionados anteriormente utilizando una combinación de varios algoritmos y técnicas distintas, entre ellas: Sparse Optical Flow de Lucas Kanade para la detección de movimiento; el algoritmo de Shi-Tomasi para la búsqueda de esquinas características de un objeto; filtros de Kalman para lograr estimar la trayectoria frente a una occlusión total; la utilización de filtros de distribución de color para detectar al objeto a seguir según su distribución de color y la probabilidad de ubicar esta distribución en la pantalla junto a filtros de color utilizando el espacio de colores CIE-Lab y HSV para el enmascaramiento de las zonas sin interés de la pantalla; y finalmente filtros de correlación adaptativos para desambiguar el seguimiento de un objeto solamente tomando como característica su color, lo cual reduce significativamente la probabilidad de seguir a un objeto de mismo color que el seleccionado.

REFERENCES

- [1] R. C. Gonzalez, R. E. Woods y S. L. Eddins. *Digital Image Processing Using MATLAB*. Prentice Hall, 2nd ed, 2002.
- [2] R. C. Gonzalez, R. E. Woods y S. L. Eddins. *Digital Image Processing Using MATLAB*. Prentice Hall, 2nd ed, 2002, pp. 52-54.

- [3] D. H. Warren y Edward R. Strelow. *Electronic Spatial Sensing for the Blind: Contributions from Perception*. Springer Netherlands, 1st ed, 1985. pp. 414.
- [4] B. D. Lucas y T. Kanade. *An iterative image registration technique with an application to stereo vision*. From Proceedings of Imaging Understanding Workshop, pp. 121-130.
- [5] J. Shi y C. Tomasi. *Good Features to Track*. 9th IEEE Conference on Computer Vision and Pattern Recognition, June 1994, Springer, pp. 593-600.
- [6] W.S.P. Fernando, L. Udawatta , P. Pathirana. *Identification of Moving Obstacles with Pyramidal Lucas Kanade Optical Flow and k means Clustering*. Faculty of Engineering, University of Moratuwa, Sri Lanka.
- [7] G. Terejanu, "Discrete Kalman Filter Tutorial", Cse.sc.edu, 2020. [Online]. Available: <https://cse.sc.edu/~terejanu/files/tutorialKF.pdf>. [Accessed: 15- Jun- 2020].
- [8] G. Welch and G. Bishop, An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill: Department of Computer Science, 2006.
- [9] J. Shi y C. Tomasi. *Good Features to Track*. 9th IEEE Conference on Computer Vision and Pattern Recognition, June 1994, Springer, pp. 593-600.
- [10] G. Turin, "An introduction to matched filters," in IRE Transactions on Information Theory, vol. 6, no. 3, pp. 311-329, June 1960, doi: 10.1109/TIT.1960.1057571.
- [11] Gary R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface" Microcomputer Research Lab, Santa Clara, CA, Intel Corporation