INSTITUTO TECNOLÓGICO DE BUENOS AIRES

22.05 ANÁLISIS DE SEÑALES Y SISTEMAS DIGITALES

Trabajo práctico N°4

Entrega N°1

Grupo 3

MECHOULAM, Alan	58438
LAMBERTUCCI, Guido Enrique	58009
RODRIGUEZ TURCO, Martín Sebastian	56629
LONDERO BONAPARTE, Tomás Guillermo	58150

Profesores
Jacoby, Daniel Andres
Belaustegui Goitia, Carlos F.
Iribarren, Rodrigo Iñaki

Presentado: 17/06/20

En el siguiente trabajo se presenta el estudio, investigación y análisis de un proceso de seguimiento del movimiento de un objeto en tiempo real, siendo conocida su posición inicial. Se utilizaron los algoritmos de Shi-Tomasi, Optical Flow de Lucas-Kanade, filtros de Kalman, de probabilidad de distribución de color y de correlación. Se realizaron pruebas tanto con videos como en tiempo real, logrando seguir correctamente al objeto ante breves o largas oclusiones y variaciones bruscas en su trayectoria.

I. Introducción

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 maquinas 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 $f(x, y)$.

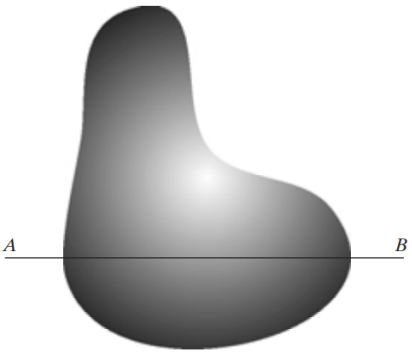


Figura 1: Entrada continua al sistema.

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 cuadrados 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.

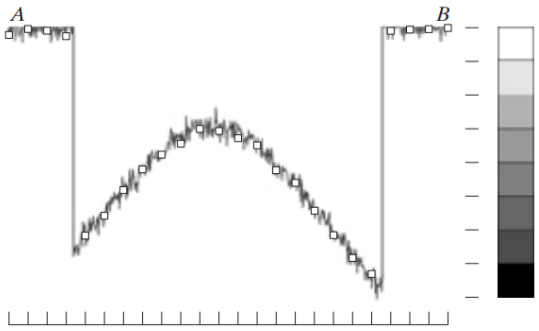


Figura 2: Amplitud de la escala de grises en la recta AB y muestreo de valores.

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].

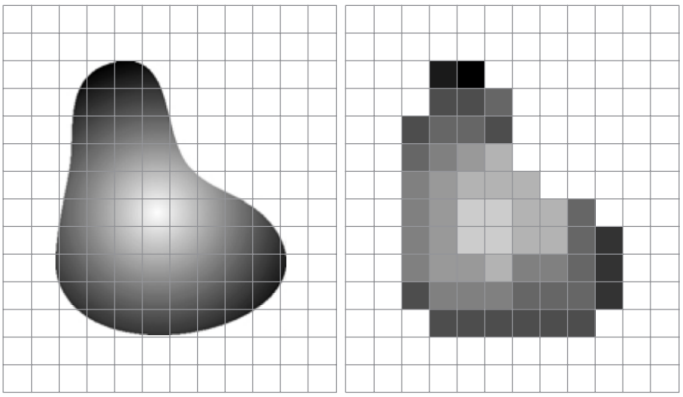


Figura 3: Imagen original comparada con la imagen digitalizada a procesar.

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 centrarse 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 plantearemos:

- 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. Investigación

II-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 Optical Flow o flujo de la imagen como el movimiento aparente de la imagen basado en la percepción visual, y tiene dimensión de velocidad $\vec{V} = (V_x \ , \ V_y)$. Si el Optical Flow se determina de dos imagenes consecutivas, aparece un vector de desplazamiento $\vec{d}$ de las cualidades elegidas, entre el cuadro n y el n+1.

(a) first frame (b) second frame (c) optical flow

Figura 4: Cubo de rubik rotando en una mesa.

Para la implementación del algoritmo existen varios caminos, como el basado en calculo de gradientes de la imagen total. Existe otro que utiliza solo ciertos puntos que se determinan constantes de la imagen. Nosotros utilizaremos el método piramidal de Lucas-Kanade.

II-A1. Lucas-Kanade: Este 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 objeto 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 algoritmo consta de dividir la pantalla en un árbol cuaternario. Esto lo hace debido a que las hipótesis de Lucas-Kanade deben cumplirse, por lo que se toma una pequeña parte de la imagen del menor tamaño posible, donde tienden a cumplirse las hipótesis del algoritmo. De allí calculando se calcula recursivamente cada partición hasta el primer nivel.

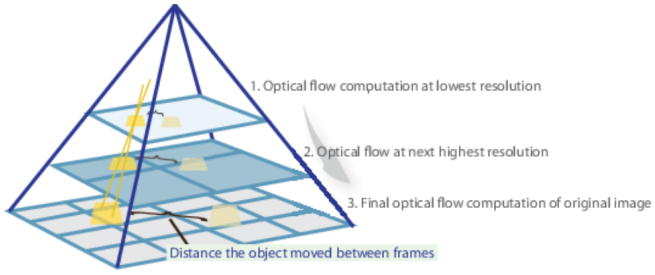


Figura 5: Cálculo del Optical Flow.

II-B. Shi-Tomasi

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 ese pixel corresponde a una esquina. El valor asignado al pixel se obtiene utilizando dos autovalores de una matriz Z. Uno le pasa estos autovalores a una función y esta los manipula y devuelve 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:

$$R = det(Z) - k(traceZ)^2 \quad (1)$$
$$det(Z) = \lambda_1 \cdot \lambda_2 \quad (2)$$
$$traceZ = \lambda_1 + \lambda_2 \quad (3)$$

Figura 6: Valor de pixel para algoritmo de Harris

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

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

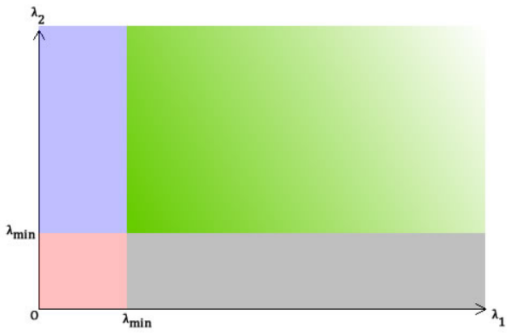


Figura 8: Detección de bordes Shi-Tomasi.

- 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.

II-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].

II-C1. Planteo del problema: Dado un proceso estocástico lineal en tiempo discreto definido por

$$x_k = Ax_{k-1} + w_{k-1}{}^{1} \tag{5}$$

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

$$z_k = Hx_k + v_k \tag{6}$$

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 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.

II-C2. Ecuaciones de predicción:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} \tag{7}$$

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

II-C3. Ecuaciones de observación:

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

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

donde,

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

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

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

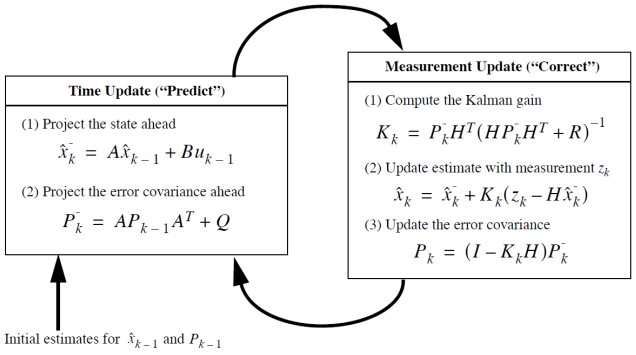


Figura 9: Funcionamiento completo del filtro Kalman simple [8].

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.

[1] Se omitió por simplicidad la función de control.

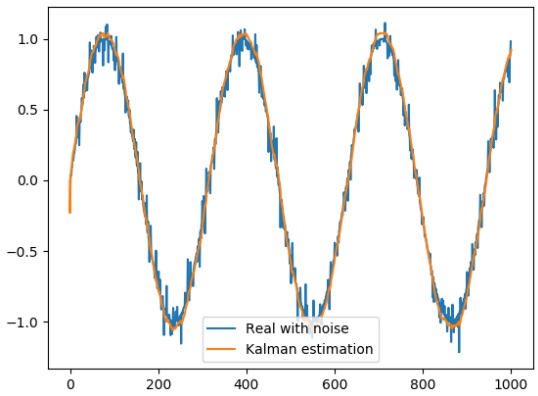


Figura 10: Seno con ruido comparada con estimación de Kalman.

II-D. Filtros de Correlación:

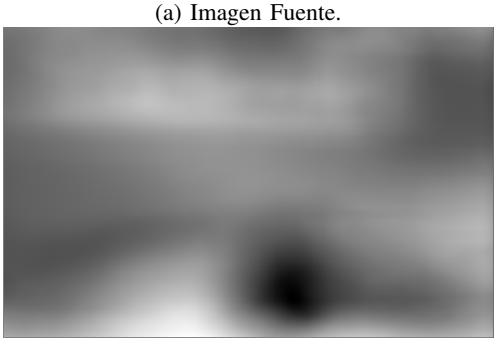
El filtro de correlación o también conocido como "Matched Filter" en inglés, se obtiene al calcular la correlación de una señal conocida, o "kernel", con una señal desconocida con el propósito de detectar este kernel en la última señal. Esto último es equivalente a convolucionar la señal desconocida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

En la implementación del algoritmo se utilizó la siguiente fórmula para el cálculo:

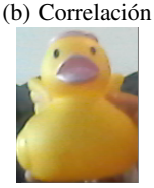
$$R(x,y) = \frac{\Sigma_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2}{\sqrt{\Sigma_{x',y'} T(x',y')^2 \cdot \Sigma_{x',y'} I(x+x',y+y')^2}} \tag{14}$$

Donde R es la función de resultado, I la señal desconocida, y T el template o kernel. A continuación se muestra una imagen y el resultado del cálculo de correlación.

(a) Imagen Fuente.



(b) Correlación



(c) Kernel

Figura 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.

III. Aportes y Desarrollo

III-A. Implementación

III-A1. Primer diseño: A partir de una selección del video proporcionada por el usuario se obtienen las mejores features del objeto seleccionado utilizando el algoritmo de Shi-Tomasi. Luego, mediante el algoritmo de Lucas-Kanade de Optical Flow se obtiene la evolución de dichas features en el tiempo. A este cúmulo de puntos se le calcula el centro de masas μ y el centro de inercia σ. Se rechazan los puntos de este cúmulo que se encuentren a una distancia mayor a $k_\sigma \sigma$ respecto al centro de masas, para luego calcular nuevamente el centro de masas el cual es el que se introduce en el filtro de Kalman como medición de la posición.

La dimensión del vector de estados del filtro de Kalman es de cuatro, poseyendo la posición y la velocidad tanto horizontal como vertical. La dimensión del vector de observación es de dos dado que solamente se mide la posición horizontal y vertical del centro de masas. La matriz A queda definida por las ecuaciones que caracterizan el movimiento del objeto

a seguir que, por simplicidad, se consideraron las ecuaciones cinemáticas de MRU, quedando entonces

$$\begin{pmatrix} x_{k-1} + dt \cdot v_{x_{k-1}} \\ y_{k-1} + dt \cdot v_{y_{k-1}} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \end{pmatrix} \quad (15)$$

Siendo,

$$A = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (16)$$

Luego, se tiene que la matriz de observación o de transición de medición quedará definida por la transformación del espacio de observación al espacio real del vector de estados del filtro. Como los observables a medir forman parte del espacio real del vector de estados y son solo las posiciones, la matriz quedará definida de manera simple

$$\begin{pmatrix} z_{x_k} \\ z_{y_k} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \quad (17)$$

Siendo,

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad (18)$$

Mientras que la matrices de covarianza del proceso Q y de medición R serán cuadradas por definición, de dimensión cuatro y dos respectivamente, y además de la forma $\mathbb{I} \cdot \xi$ dado que se asume independencia de las variables aleatorias, quedando

$$Q = \begin{pmatrix} \xi_Q & 0 & 0 & 0 \\ 0 & \xi_Q & 0 & 0 \\ 0 & 0 & \xi_Q & 0 \\ 0 & 0 & 0 & \xi_Q \end{pmatrix} \quad (19)$$

$$R = \begin{pmatrix} \xi_R & 0 \\ 0 & \xi_R \end{pmatrix} \quad (20)$$

Siendo ξ_Q y ξ_R las varianzas del proceso y de medición respectivamente.

Finalmente, $\hat{x}_{k|k}$ y $P_{k|k}$ tomarán la forma

$$\hat{x}_{k|k} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{v_x} \\ \hat{v_y} \end{pmatrix} \quad (21)$$

$$P_{k|k} = \begin{pmatrix} \gamma_x & 0 & 0 & 0 \\ 0 & \gamma_y & 0 & 0 \\ 0 & 0 & \gamma_{v_x} & 0 \\ 0 & 0 & 0 & \gamma_{v_y} \end{pmatrix} \quad (22)$$

Siendo γ_i la varianza de cada variable de estado. Cabe notar que el modelo utilizado asume independencia entre las variables aleatorias por simplicidad.

Este algoritmo, hasta ahora explicado, cuenta con grandes problemas. Dados dos objetos A y B, donde A es el objeto a seguir:

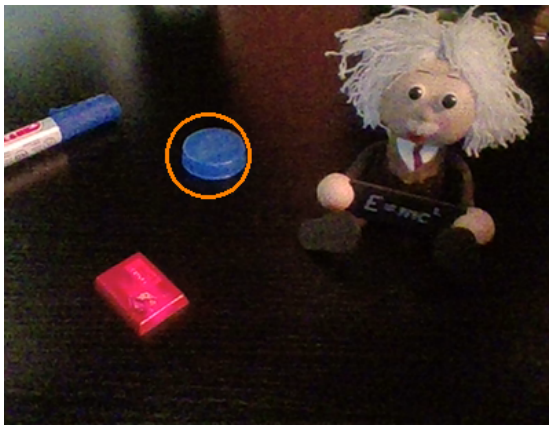
- Como el algoritmo de Shi-Tomasi tiende a colocar features en bordes bien definidos, y estos suelen ser los bordes externos de un objeto, si el objeto a seguir A pasa por delante de un objeto B con bordes bien definidos, las features del algoritmo de Optical Flow tenderán a adosarse a los bordes del objeto B por más que este esté quieto.
- Si el objeto A pasase por detrás del objeto B inmóvil, todas las features sobre el objeto a seguir quedarían adosadas al objeto B. Esto plantea un problema mucho más grande que el anterior, dado que no se podrá utilizar la estimación del filtro de Kalman de manera confiable, debido a que cuando los features se queden quietos adosados en B, se seguirá midiendo de todas formas y se seguirán entregando esas mediciones al filtro de Kalman, por lo que la estimación de este se adosará también al objeto B.
- Profundizando en el item anterior, existe una ambigüedad entre que A pase de un movimiento MRU a que se quede completamente quieto y que A pase por detras de B, dado que en ambas situaciones las features quedarán quietas en el lugar donde A se haya quedado quieto o el lugar de la intersección entre A y B

III-A2. Segundo diseño: La mejora propuesta para sortear estos problemas consiste en agregar un filtro de color, para diferenciar a A y a B por su color. La implementación de este filtro de color comienza en el momento en el que el usuario selecciona el área a seguir. Se calcula la mediana del color de la zona elegida, y ese color es transformado de formato RGB a formato HSV, o Hue-Saturation-Value, dado que es más sencillo definir una máscara de este modo. Dado x e y las coordenadas de los píxeles de la imagen, $\vec{R_k}$, $\vec{G_k}$ y $\vec{B_k}$ los vectores de píxeles de la selección, $mediana(\vec{R_k}, \vec{G_k}, \vec{B_k}) = (m_R, m_G, m_B)$ el vector que define la mediana del color de la selección, $\mathbb{HSV}$ el operador de transformación del espacio de coordenadas RGB a HSV, y $\mathbb{HSV}(m_R, m_G, m_B) = (m_H, m_S, m_V)$ el vector de la mediana del color de la selección en el espacio HSV, queda entonces definida la máscara de color como

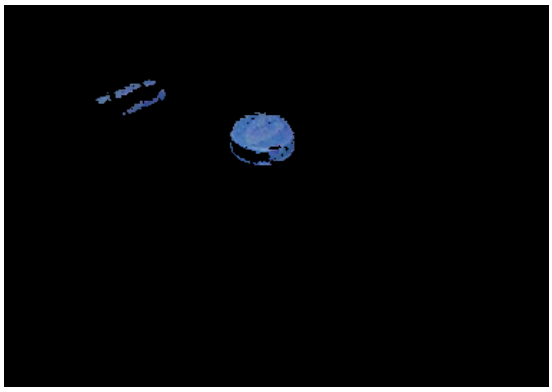
$$f(x, y) = (f_h(x, y), f_s(x, y), f_v(x, y)) \quad (23)$$

$$f_u(x, y) = \begin{cases} 1 & -\Lambda_u < u_{x,y} - m_u < \Lambda_u \\ 0 & \text{sino} \end{cases} \quad (24)$$

donde Λ es el límite para cada propiedad u, siendo estas Hue, Saturation o Value. Una vez obtenida la máscara $f(x, y)$ se realiza la operación AND entre la máscara y la imagen original, logrando obtener una intensidad nula en los píxeles de la imagen que no posean un color cercano a la mediana calculada como se puede ver en las Figuras (12a) y (12b).



(a) Video antes de aplicar la máscara siguiendo al objeto azul.



(b) Video después de aplicar la máscara. Es con esta imagen que opera el algoritmo y no con la normal.

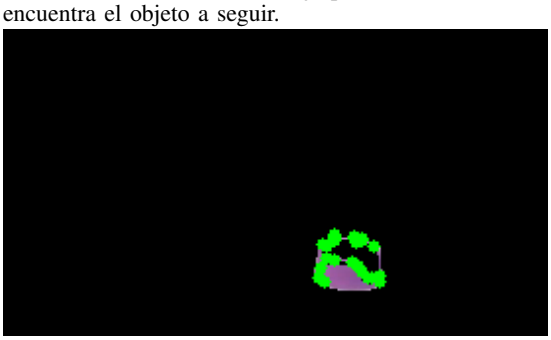
Figura 12: Comparación entre video normal y luego de aplicar la máscara.

Este filtrado de color no solo posee una gran sinergia con el método de obtención de features de Shi-Tomasi, dado que este frame tendrá bordes de mucha mejor calidad, sino que también posee una gran sinergia con el método de Optical Flow, debido a que como se verán menos bordes en la imagen modificada, aumenta notablemente la probabilidad de que los features queden contenidos todos dentro, o sobre los bordes del objeto a seguir.

Volviendo al caso de los objetos A y B:

- Si A y B son de colores distintos, con el filtro de color, el algoritmo nunca detectará que existe un objeto B por lo que seguirlo será imposible, y por consecuencia seguirá únicamente a A y no se tendrá el problema de A pasando por frente de B.
- Si A y B son de colores distintos, y A, el objeto a seguir, pasa por detrás de B, ahora no se adosarán las features al objeto B, sino que el algoritmo verá que el área de A se achica a medida que este queda oculto por B. Como los features no tendrán bordes para seguir, estos desaparecerán, por lo que se dejará de entregar mediciones al filtro de Kalman y este estimará correctamente la posición de A mientras que este no haya sufrido aceleración alguna mientras estaba oculto.

(a) Ventana con el funcionamiento normal del seguidor, el cual utiliza un círculo naranja para detallar donde se encuentra el objeto a seguir.



(b) Ventana la cual solo aparece al seleccionar el modo debug. Aquí se muestra si el filtro de color se encuentra activado, el video luego de pasar por la máscara de color, las features del objeto como puntos verdes y la zona de búsqueda si se diese el caso de tracking failure.

Figura 13: Resultado de la selección de un elemento.

Sin embargo, el filtro de Kalman no puede estimar la posición del objeto indefinidamente. Es por esto que cuando el objeto se pierde, es decir, no se logran detectar más features, se entra en otra fase del programa. En esta fase, se realiza una búsqueda del objeto en cada frame del video a partir de la estimación del filtro de Kalman y el tamaño original del área seleccionada. La altura y el ancho del área de búsqueda se incrementará por un factor constante por cada frame en la que la búsqueda falle. Esta búsqueda se basa en aplicar el algoritmo de Shi-Tomasi sobre el área mencionada.

Finalmente, para reforzar el algoritmo de seguimiento, se implemento además una recalculación periódica de las features con el método de Shi-Tomasi, partiendo de la media del cúmulo de features, removiendo los outliers, volviendo a calcular nuevamente la media y utilizar esta posición para aplicar el algoritmo de Shi-Tomasi, de misma manera que se realiza sobre la primera selección proporcionada por el usuario. Todo esto nos permite subsanar varios de los problemas planteados anteriormente, sin embargo, existen aún varios otros problemas:

- ACA PONER LOS PROBLEMAS QUE TENIAMOS EN LA ENTREGA PASADA

III-A3. Último diseño: EXPLICAR LO SIGUIENTE

- Filtro de color pasó del espacio HSV a CIE-LAB, explicar que es para mas invarianza a la iluminacion
- Al filtro de color se le unió una alternativa, el de camshift, explicar en qué situaciones es mejor uno o el otro

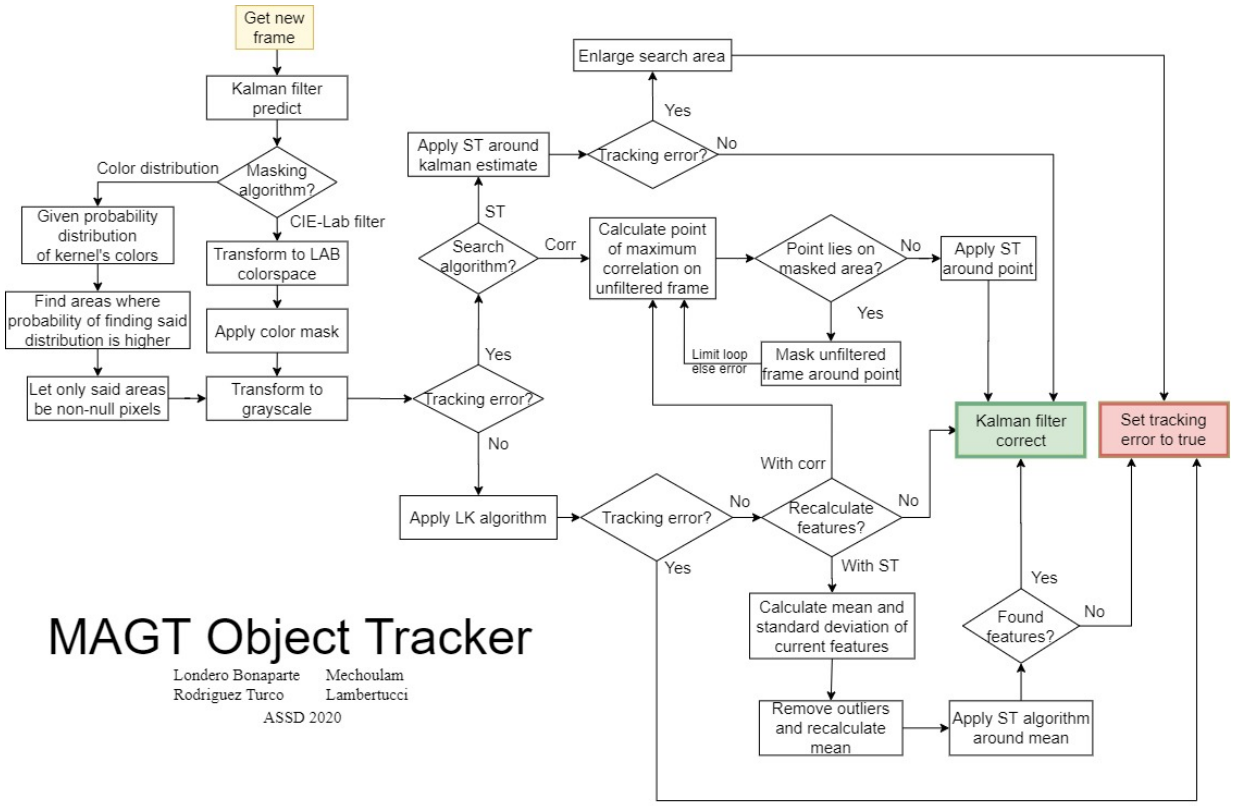


Figura 14: Flow-chart del algoritmo.

- ahora la busqueda de objeto perdido se puede hacer con correlacion, explicar en que casos conviene uno o el otro, explicar el proceso de parcheamiento
- ahora el recalculo periodico se puede realizar con correlacion, explicar en que casos combiene uno o el otro
- mencionar la gui?

III-B. Algoritmo correlación

Para el algoritmo de correlacion no solo se utiliza el filtro explicado en la sección II-D se realiza un filtrado de tanto el frame como el kernel, haciendo un cambio a **HSV** y un filtrado del frame en este espacio bajo las siguientes especificaciones:

$$MSK = H \ \epsilon \ (0, 180) \quad S \ \epsilon \ (50, 255) \quad V \ \epsilon \ (32, 255) \quad (25)$$

luego se utiliza el filtro de correlación y se obtiene el punto donde esta es mayor, y se compara con el frame filtrado, si no comparte las características de color que son calculadas por el filtrado de color, este punto de correlación es desechado. Esto se logra aplicando una mascara en la sección de mayor correlación y sobre este realizar nuevamente el cálculo de máxima correlación. Se realizan unas pocas iteraciones 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 (Oclusión del objeto).

III-C. Optimización del Código

Se realizó una optimización del codigo, quitando redundancias y utilizando únicamente librerías que son corridas en C++, se optó por un enfoque OOP. Esto permitió escalar el proyecto y una mayor modularidad. Contando con una GUI y la posibilidad de un trackeo multi-objeto, adicionalmente se puede interactivamente cambiar especificaciones de los trackers, al igual que diseñar el filtro de color para obtener el mejor resultado posible.

III-D. KTN

Se decidió implementar código en **Python**, apoyándose en la librería OpenCV. El pseudocódigo que corresponde es:

```
kalman = KalmanFilter()

cap = VideoCapture()

lower_thr, upper_thr = [],[]

prev = captureROI(cap)

prev = space_translate(x, y, prev)

frame = cap.read()

while(cap.isOpened()):

    frame_num++
    frame_real = cap.read()


    hsv = cvtColor(frame, COLOR_BGR2HSV)
    mask = inRange(hsv, lower_thr, upper_thr)
    frame = bitwise_and(frame, mask)

    if(recalc and not error):
        frame_num = 0
        prev, x, y = recalculateFeatures()

    if error:
        prev=  searchObject(kalman)
        dyn_h, dyn_w = new_dyn_h, new_dyn_w


    good_new, good_old,prev= measureFeatures()

    drawEstimate(good_new, kalman, frame)
```

Finalmente se presenta un diagrama en bloques del programa, donde se puede apreciar como interaccionan todos los algoritmos trabajando en armonía.

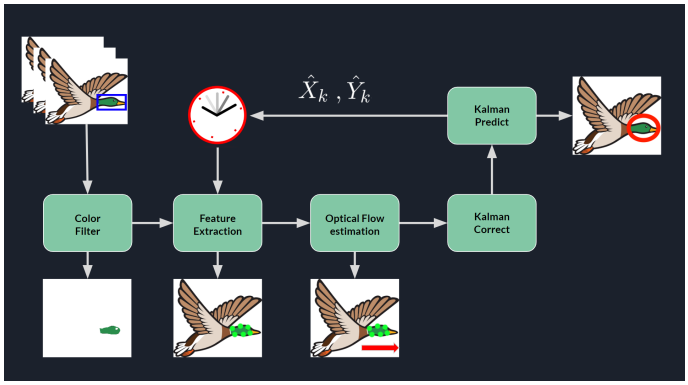


Figura 15: Diagrama en bloques del programa.

III-E. Escenarios de fallas

Algunos casos donde puede fallar el Tracker son:

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

IV. Trabajo en futuro

Algunas mejoras que pueden ser implementadas en futuros desarrollos son:

- Trackeado multiobjeto:
  Da la opción al usuario de trackear mas de un objeto en la pantalla, utilizando filtros de color diferenciados por cada objeto.
- Filtro de Kalman donde la matriz de covarianza de medición no sea constante:
  En el modelo utilizado actualmente la matriz de covarianza es constante, esto asume que la presición de medición es fija. Esto último no suele ser verdad en la realidad. Una mejora es la introducción de modificaciones en la matriz de covarianza de manera dinámica así el filtro proporcionaría mejores predicciones.
- Filtros de correlación:
  El uso de filtros de correlación tomando como cuadro inicial la selección del usuario se calcula la correlación con toda la pantalla entre cuadros, donde la correlación sea mayor será mas probable la coincidencia.
- Invarianza ante cambios de iluminación:
  El cambio de iluminación impacta directamente sobre el color percibido. Esto afecta directamente a nuestra técnica de filtrado por color. Es por eso que se desean explorar formas de hacer más robusto al sistema ante estos cambios.
- Filtro de sensibilidad Multicolor:
  Dado tres objetos A, B y C, donde A cuenta con 2 colores distintivos, siendo estos rojo y azul, B un objeto rojo y C uno azul, lo que propone este filtro es la capacidad de diferenciar al objeto a seguir por mas de un color distintivo, cosa de que la interferencia de B o C en el camino de A no comprometa su seguimiento debido a que ni B ni C posee la combinación de colores de A.
- Un mejor modelado de las ecuaciones dinámicas del filtro de Kalman para incluir la aceleración como variable de estado:
  El modelo utilizado actualmente consta de un movimiento rectilíneo uniforme bidimensional. La inclusión de la aceleración proporcionaría una mejor predicción.

V. Conclusiones

Se logró seguimiento de un objeto especificado por el usuario en base a cualidades distintivas del mismo, siendo estas su color y sus bordes, al igual que por su movimiento. Contando con una predicción certera en el caso de que se encuentre el objeto en pantalla al igual que la situación de que desaparezca de pantalla, aunque la varianza de la predicción aumentará cuanto mas tiempo no sea detectado, con la capacidad encontrar nuevamente a este.
La elección de los algoritmos y filtros utilizados son sinérgicos dado que estos se complementan entre sí, por ejemplo el filtro de color proporciona bordes definidos para el objeto a seguir, que es ideal para la obtención de bordes de Shi-Tomasi

Referencias

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

[2] R. C. Gonzalez, R. E. Woods y S. L. Eddins. *Digital Image Processing Using MATLAB*. Prentice Hall, 2da ed, 2002, pp. 52-54.
[3] D. H. Warren y Edward R. Strelow. *Electronic Spatial Sensing for the Blind: Contributions from Perception*. Springer Netherlands, 1er ed, 1985. pp. 414.
[4] B. D. Lucas y T. Kanade. *An iterative image registration technique with an application to stereo vision*. De 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.