





Universidad Autónoma de Yucatán

MÉTODO SACON

PROCESAMIENTO DE VÍDEO





EQUIPO

Juan Pablo Ballinas Aquino

Joel Roberto Bellos

Julián Adrian Viana Palomo

Diana Lorelei Maldonado Cuevas



¿Cómo se calcula el modelo de fondo?

SACON es un modelo de fondo donde construye un conjunto de muestras de fondo para cada pixel a lo largo del tiempo. A partir de muestras, estima un modelo estadístico de fondo usando un criterio de consenso.

Podríamos definir que el modelo se basa en los siguiente procesos:

- Almacenamiento de muestra
- Definición de consenso
- Clasificación de fondo
- Uso de espacio de color normalizado
- Estrategia de actualización selectiva

1. Almacenamiento de muestra

Para cada píxel m se guarda un conjunto de N muestras anteriores:

$$x_i(m) \mid i = 1, \dots, N, N < t$$

- Donde $x_i(m)$ es una observación de un píxel m en el tiempo t .
- Esta observación puede tener varios canales, por ejemplo, en una imagen en color seria (R, G, B), algo así:

$$x_i^{(m)} = \left(x_i^{(m)} C_1, x_i^{(m)} C_2, \dots, x_i^{(m)} C_k \right) \quad [\text{vario canales}]$$

2. Definición de consenso

Para cada muestra en la caché, el consenso de muestras en el modelado de fondo se define como:

$$\Gamma_i^c(m, t) = \begin{cases} 1 & \text{si } |x_i^c(m) - x_t^c(m)| \leq T_r \\ 0 & \text{en otro caso} \end{cases}$$

Donde T_r es un umbral selectivo de tolerancia al error y diferentes píxeles tienen diferentes umbrales.

- Se compara el valor actual del píxel con cada muestra almacenada. Si la diferencia es menor a un umbral de tolerancia (T_r), esa muestra se considera en “acuerdo” con el valor actual.

El algoritmo generalmente establece un valor global de T_r para todos los píxeles y simultáneamente establece

$$T_r = \eta \sigma_i^2$$

T_r es la varianza estandar en cada pixel i -esimo de la imagen y η generalmente se establece en 2.5 o 3 veces la desviación estándar

3. Clasificación de fondo

SACON clasifica un píxel como fondo si el valor actual es similar a la mayoría de sus muestras históricas dentro de un rango de tolerancia.

Si el valor es muy distinto y no tiene suficiente consenso, lo marca como primer plano.

$$B_i(m) = \begin{cases} 1 & \text{if } \sum_{i=1}^N \Gamma_i^c(m, t) \geq T_n \quad \forall c \in C_1, \dots, C_K \\ 0 & \text{in another case} \end{cases}$$

Clasificación de fondo

Muestras de fondo por píxel $x_{i(m)}$ y observación actual $x_t(m)$

Consenso por tolerancia por canal:

$$\Gamma_i(m) = \begin{cases} 1, & \text{si } |x_i^t(m) - x_i^f(m)| \leq T_c \forall c \\ 0, & \text{en otro caso} \end{cases}$$

Clasificación por conteo de consensos:

$$S(m) = \sum_{i=1}^N \Gamma_i(m), \quad S(m) \geq T_n \Rightarrow \text{BG}, \text{ si no } \Rightarrow \text{FG}$$

Umbrales:

$$T_c = \min(T_c^1, \eta \sigma^c), \quad \eta \approx 2.5 - 3$$
$$T_n \approx T_r, N$$

Significado

$x_i(m)$: recuerdos de fondo que guarda un píxel, son las muestras históricas en la posición m .

$x_t(m)$: valor del píxel en el instante actual t .

$\Gamma_i(m)$: voto de coincidencia de la muestra i con la observación actual.

$S(m) = \sum_{i=1}^N \Gamma_i(m)$: Suma de votos (número de coincidencias)

T_n : mínimo de votos requeridos para declarar que el píxel pertenece a BG.

T_r : margen de tolerancia permitido, cuán diferente puede ser $x_t(m)$ respecto a los recuerdos $x_i(m)$

T_r^c : marca cuánto puede desviarse el canal c del valor "recordado" antes de seguir considerándolo fondo.

Muestra por pixel

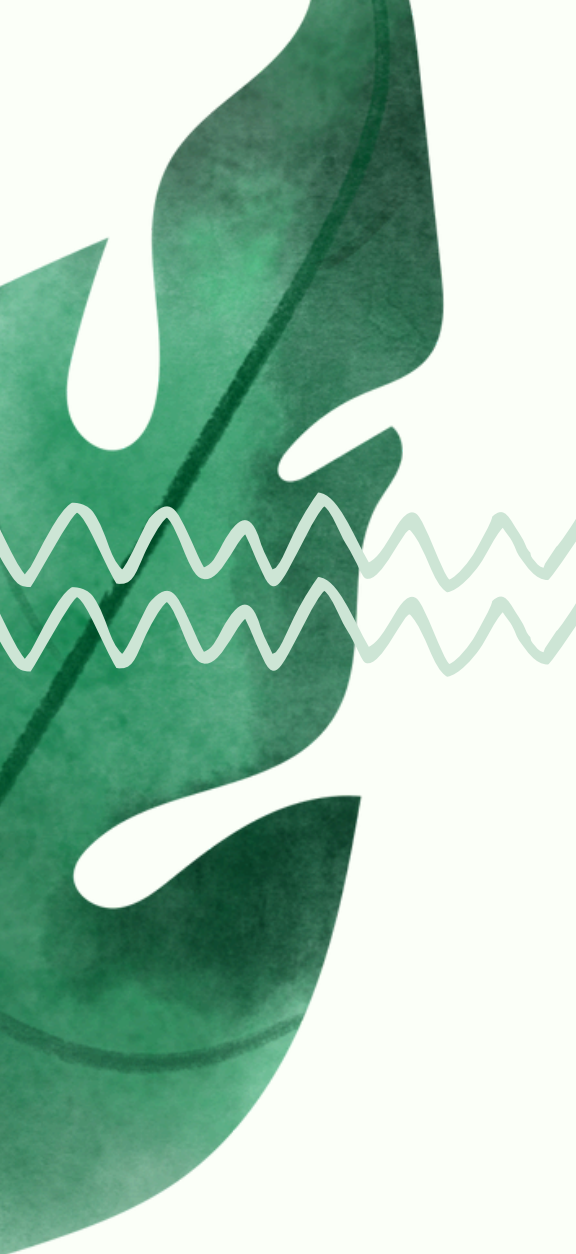
- Piensa que cada píxel de la cámara tiene una "memoria" donde guarda varios valores anteriores (sus colores o intensidades pasadas). Esas muestras se denotan por $x_{t(m)}$.

Consenso por tolerancia T_r

- Cuando llega un valor nuevo $x_{t(m)}$ en el cuadro actual, el sistema lo compara con esas memorias.
- Si el valor nuevo es muy parecido (dentro del margen T_r) a muchas de sus memorias, se dice que "hay consenso".

Clasificación con votos

- Se cuentan cuántas memorias coinciden con el valor actual.
- Si se juntan suficientes votos, es decir $S(m) \geq T_n$, el píxel se queda como fondo, si no alcanza, se declara objeto.

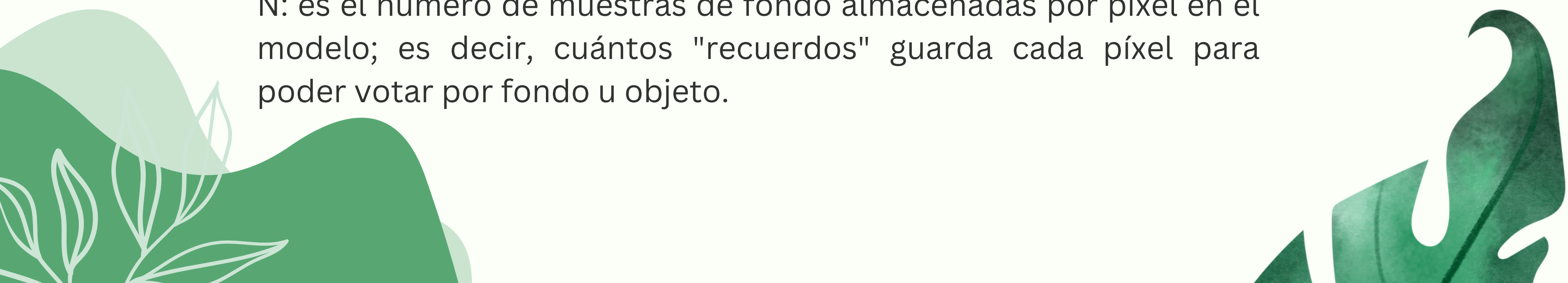


η : Controla cuántas desviaciones estándar σ^c se permiten al comparar el valor actual con una muestra del fondo.

Valores mayores de η hacen la tolerancia más amplia y facilitan que más muestras "voten" como fondo; valores menores la vuelven estricta y aumentan la sensibilidad a cambios.

Rango típico sugerido: $\eta \approx 2,5-3$.

N: es el número de muestras de fondo almacenadas por píxel en el modelo; es decir, cuántos "recuerdos" guarda cada píxel para poder votar por fondo u objeto.

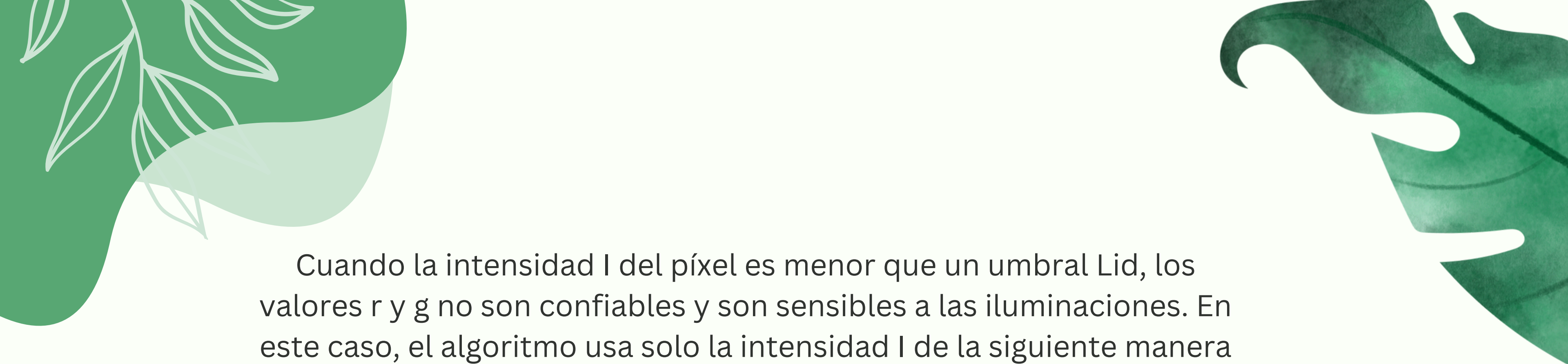


4. Uso de espacio de color normalizado

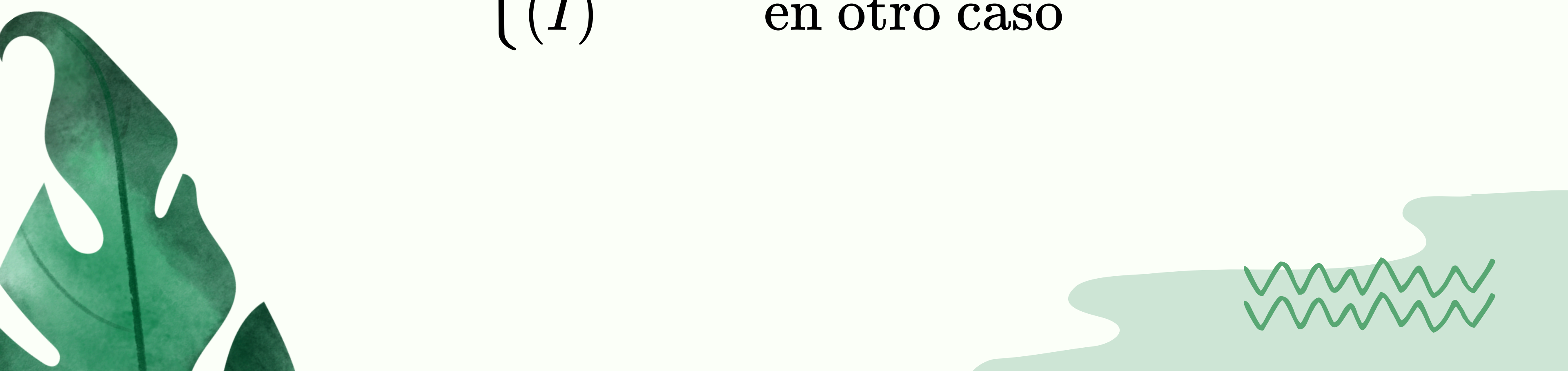
- Para reducir sensibilidad a cambios de iluminación, SACON usa un espacio de color normalizado (r,g,l) como espacio de característico.
- Además, introduce reglas adicionales para suprimir sombras basadas en la intensidad.
- Utiliza la versión modificada de la definición de consenso:

$$\Gamma_t^c(m, t) = \begin{cases} 1 & \text{si } |x_i^c(m) - x_t^c(m)| \leq T_r, \forall c \in \{r, g\} \text{ and } \beta \leq \frac{x_c^t}{x_c^l} \leq \gamma, \forall c \in I \\ 0 & \text{en otro caso} \end{cases}$$

Cuando la intensidad I del píxel es menor que un umbral Lid, los valores rr y gg no son confiables y son sensibles a las iluminaciones. En este caso, el algoritmo usa solo la intensidad I de la siguiente manera



Cuando la intensidad I del píxel es menor que un umbral I_{id} , los valores r y g no son confiables y son sensibles a las iluminaciones. En este caso, el algoritmo usa solo la intensidad I de la siguiente manera

$$x = \begin{cases} (r, g, I) & \text{si } I \geq I_{id} \\ (I) & \text{en otro caso} \end{cases}$$


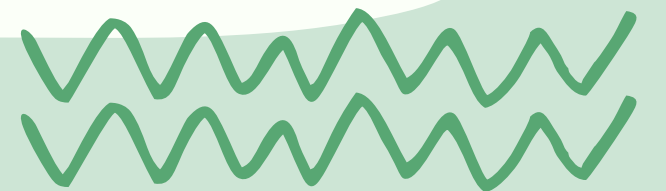
5. ¿Cómo se realiza la actualización del modelo del fondo?

Mecanismo propuesto en SACON: actualización selectiva con Time Out Map (TOM)

- Se define un mapa TOM que lleva la cuenta del número de cuadros consecutivos en que un píxel ha sido clasificado como primer plano.
- Cuando un píxel vuelve a ser clasificado como fondo, su valor en TOM se reinicia a cero
- Si el valor de TOM en un píxel supera un umbral T_{tm} , ese píxel se reasigna como parte del fondo (interpretando que el objeto que cubría esa región se ha vuelto estático o pertenece al fondo)


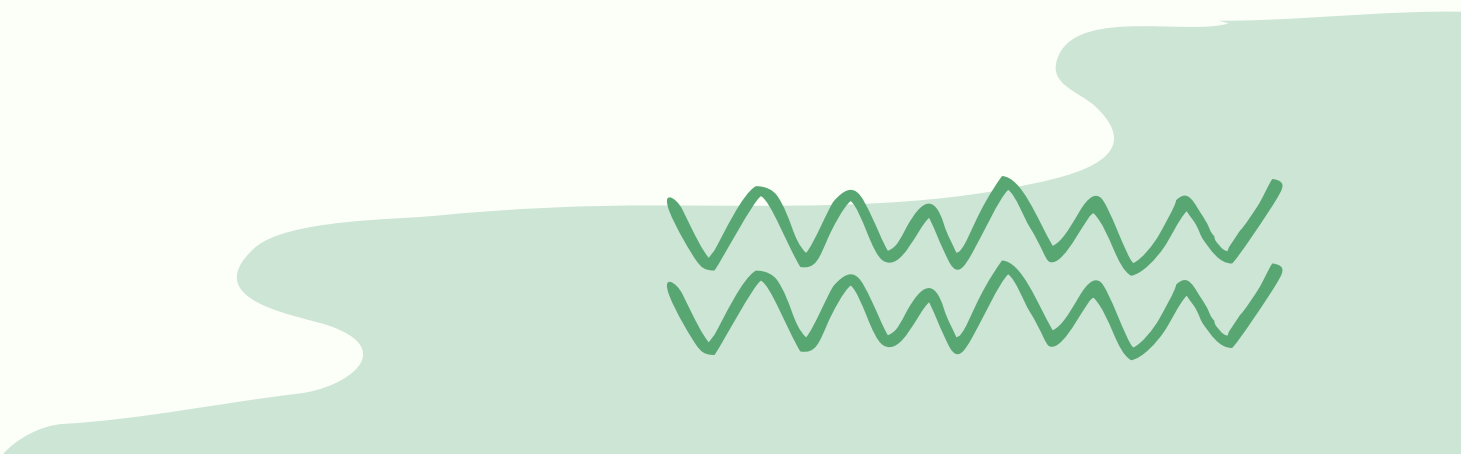
$$\begin{aligned} TOM_t(m) &= TOM_{t-1}(m) + 1 && \text{if } B_t(m) = 0 \\ TOM_t(m) &= 0 && \text{otherwise} \end{aligned}$$

donde $B_t(m)$ indica si el píxel m en el cuadro t es clasificado como fondo (1) o primer plano (0)





Ventajas del método de actualización TOM:

- Si un objeto estático cubre un área grande, se maneja a nivel de región conectada en lugar de solo píxeles individuales
 - Se distingue entre objetos en movimiento y estáticos analizando:
 - + El cambio en la posición del centro del objeto.
 - + El número de píxeles que lo componen.
 - Si el objeto se juzga estático y su TOM supera el umbral, todos sus píxeles se añaden al fondo.
 - Con ello se evita que zonas de objetos en movimiento se absorban erróneamente al fondo
- 
- 

¿Qué problema o problemas de la escena toma en cuenta el método?

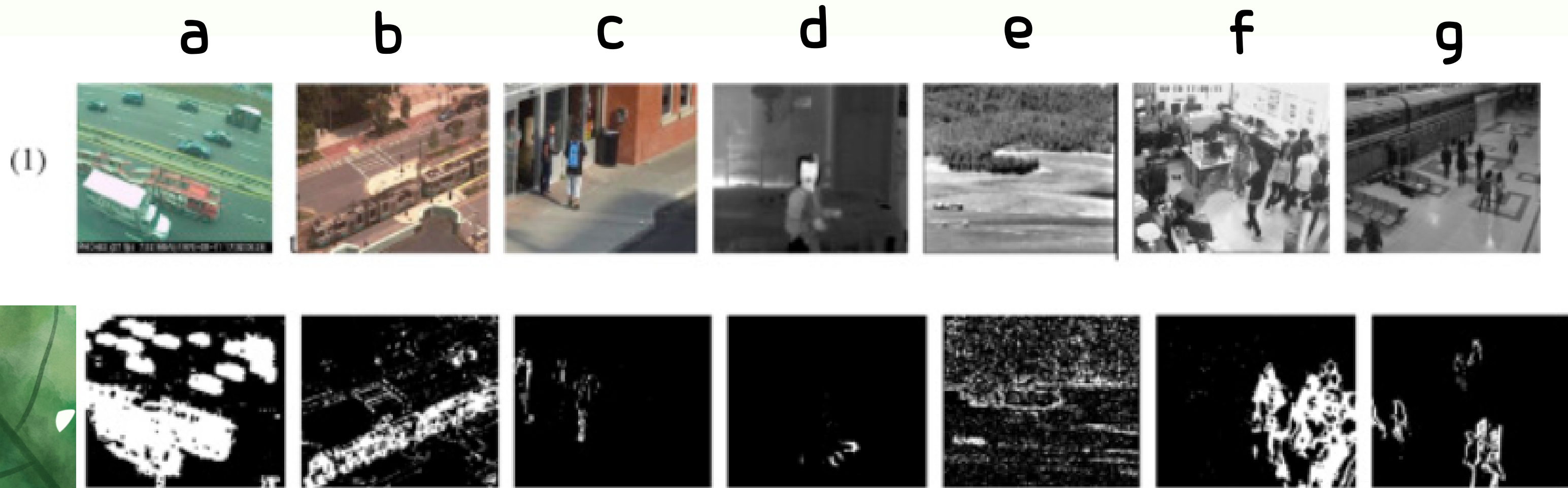
Ruido de color normalizado: La razón principal es que el espacio de color RG es sensible a los cambios de iluminación.

Actualización del modelo de fondo, si el modelo de fondo cambia las muestras del fondo deben ser actualizadas para que el modelo pueda adaptarse a objetos agregados o que fueron cambiados de lugar, adaptarse a cambios en las condiciones de luz.

Sombras: Separa la información de color de la de luminosidad y usa las coordenadas de cromaticidad para suprimir las sombras, pero esto puede causar pérdida de información.

¿Qué problema o problemas de la escena toma en cuenta el método?

Ruido de video: Sólo necesita comparar el píxel actual con una pequeña cantidad de muestras de fondo cercanas en lugar de la mayoría de las muestras del modelo de fondo, de modo que puedan debilitar los efectos del ruido en el modelo.



Codigo

- Lo que hace el código es crear un arreglo tridimensional de NumPy de tamaño (alto, ancho, N).
- Si el frame de entrada es a color, se convierte a escala de grises.
- Luego, el arreglo cache se llena completamente, capa por capa, con la imagen de ese primer frame en escala de grises.
- Esto crea un historial inicial de 30 cuadros idénticos, que servirá como la representación inicial del fondo estático. Este historial se usará más adelante para comparar los cuadros subsiguientes y detectar el movimiento.

```
import cv2
import numpy as np

# Inicializar historial de tamaño (alto, ancho, N)
def init_cache(frame, N=30):
    # if len(frame.shape) == 3:
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    h, w = frame.shape
    cache = np.zeros((h, w, N), dtype=np.uint8)
    for i in range(N):
        cache[:, :, i] = frame
    return cache
```


Codigo

En la función `consensus_check`, se determina qué píxeles en un nuevo cuadro pertenecen al fondo mediante un proceso de votación. El código compara cada píxel del cuadro actual con los píxeles correspondientes en los cuadros históricos almacenados en el caché. Un píxel se considera en "consenso" (es decir, parte del fondo) si su diferencia con una cantidad suficiente de cuadros históricos es menor que un umbral. El resultado es una máscara binaria que distingue los píxeles del fondo de los que probablemente forman parte de un objeto en movimiento.

```
# Verificación de consenso vectorizada
def consensus_check(frame, cache, Tr, Tn):
    # Calcular diferencias absolutas
    diffs = np.abs(cache - frame[:, :, np.newaxis])
    # Contar cuántas muestras están dentro del umbral
    agreements = np.sum(diffs < Tr, axis=2)
    # Devolver máscara de consenso
    return agreements >= Tn
```


Codigo

La función `init_cache` inicializa un historial de cuadros de video, `consensus_check` utiliza ese historial para comparar un cuadro nuevo y determinar qué píxeles son parte del fondo, y `classify` usa el resultado de esa verificación para generar una máscara binaria que aísla los objetos en movimiento. En esencia, el sistema aprende el fondo estático y luego lo usa para identificar y segmentar objetos que se mueven en primer plano.

```
# Clasificación vectorizada
def classify(frame, cache, Tr, Tn):
    if len(frame.shape) == 3:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Verificar consenso para todos los píxeles a la vez
    bg_mask = consensus_check(frame, cache, Tr, Tn)

    # Crear máscara de primer plano (donde NO hay consenso)
    fg_mask = np.where(bg_mask, 0, 255).astype(np.uint8)

    return fg_mask
```

Codigo

```
# Actualización de cache vectorizada
def update_cache(frame, fg_mask, cache, tom, TTM):
    if len(frame.shape) == 3:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Para píxeles de fondo: actualizar siempre
    bg_mask = fg_mask == 0
    cache[bg_mask, :-1] = cache[bg_mask, 1:]
    cache[bg_mask, -1] = frame[bg_mask]
    tom[bg_mask] = 0

    # Para píxeles de primer plano: actualizar solo si TTM se excede
    fg_mask_bool = fg_mask == 255
    tom[fg_mask_bool] += 1

    update_fg = fg_mask_bool & (tom > TTM)
    cache[update_fg, :-1] = cache[update_fg, 1:]
    cache[update_fg, -1] = frame[update_fg]
    tom[update_fg] = 0

    return cache, tom
```

En la función `update_cache`, el historial de cuadros se actualiza de forma selectiva. Si un píxel ha sido clasificado como fondo, su valor se desplaza a la posición más reciente en el caché para mantener la historia. Por otro lado, si un píxel ha sido clasificado como primer plano (objeto en movimiento), su valor no se actualiza inmediatamente; en su lugar, se incrementa un contador `tom`

Codigo

```
def sacon(video_path, N=30, Tr=20, Tn=2, TTM=50):
    video = cv2.VideoCapture(video_path)
    if not video.isOpened():
        print("No se pudo abrir el video")
        return

    ret, frame = video.read()
    if not ret:
        print("No se pudo leer el primer frame del video")
        video.release()
        return

    # Inicializar cache y TOM
    cache = init_cache(frame, N)
    tom = np.zeros((frame.shape[0], frame.shape[1]), dtype=np.int32)

    # Para mostrar estadísticas
    frame_count = 0
    start_time = cv2.getTickCount()

    while True:
        ret, frame = video.read()
        if not ret:
            break


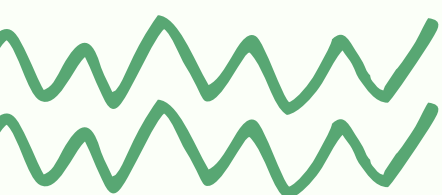
        frame_count += 1

        # Clasificar frame
        fg_mask = classify(frame, cache, Tr, Tn)

        # Actualizar cache
        cache, tom = update_cache(frame, fg_mask, cache, tom, TTM)


        # Aplicar apertura morfológica para reducir ruido
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
        fg_mask_clean = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)
```

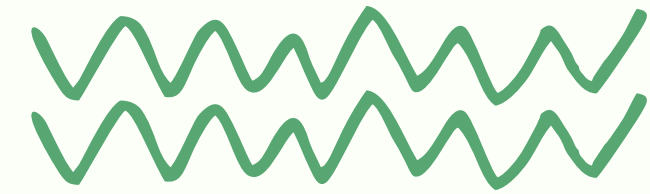
La función principal sacon, que orchestra el método de sustracción de fondo completo. . Primero, inicializa la captura de video y verifica que se haya abierto correctamente. Luego, llama a init_cache para establecer el historial de cuadros y crea el arreglo tom para rastrear el tiempo de permanencia. En un bucle continuo, la función procesa cada cuadro del video. Para cada cuadro, llama a classify para generar la máscara de primer plano y a update_cache para actualizar el historial



```
# Aplicar apertura morfológica para reducir ruido  
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))  
fg_mask_clean = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)
```

Entonces para aclarar lo que realiza lo que se presento es porque el resultado tenia un limpia, de por este metodo genera mucho ruido sin embargo se le aplica un filtro de la libreria de openCV:

- `cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)` aplica una operación de apertura morfológica.
 - Con la operación `cv2.MORPH_OPEN` se elige precisamente porque es una solución estándar y muy efectiva
 - La razón principal es que el algoritmo suelen producir "ruido" en forma de pequeños puntos o manchas que no son parte de los objetos en movimiento reales entonces este metodo se podria decir que es muy sensible al ruido y que solo usamos una imagen a escala a grises para la prueba, es decir usamos un solo canal para alamecenar las muestra. Segun este tipo de operacion funciona perfectamente con ese tipo de datos que solo usan un canal.
 - En otras palabras se le hace un post procesamiento aplicando la erosion y dilatacion que es lo que trata el metodo de MORPH OPEN
- 



Thank You

