Universidad Pontificia de Comillas

ICAI

LABORATORIO 4

Visión por Ordenador I

Lydia Ruiz Martínez, Pablo Tuñón Laguna

DETECCIÓN DE MOVIMIENTO Y SEGUIMIENTO DE OBJETOS

Curso 2024-2025

Lydia Ruiz Martínez, Pablo Tuñón Laguna

Índice

1. Introducción

A lo largo de anteriores estudios, se ha estudiado la visiónpor ordenador aplicada a imágenes detectando características, como esquinas, bordes, líneas rectas y puntos de interés. Sin embargo, la visión por ordenador no se limita al procesado de imágenes estáticas, también se aplica al procesado de secuencias de imágenes tales como vídeos. Para ello, se aplican distintas técnicas de visión por ordenador a los distintos fotogramas que componen el vídeo.

En el siguiente escrito se recoge el estudio realizado sobre el análisis de objetos en movimiento, para ello se va a aislar un objeto en movimiento de su fondo estático, se va a realizar un análisis de flujo óptico y se va a realizar un seguimiento de los objetos en movimiento.

El desarrollo del estudio se ha realizado en el lenguaje de programación Python, utilizando la librería OpenCV, que es una librería de visión por ordenador de código abierto. Cabe destacar que para descomponer los vídeos en un conjunto ordenado de dotogramas se ha usado la clase *VideoCapture* de OpenCV, que permite obtener información sobre las propiedades del vídeo, como el ancho, el alto y los FPS así como extraer cada uno de los frames a partir de la dirección del vídeo. Con dicha función se desarrolló la función propia *read_video* que devuelve los frames y las propiedades del vídeo en cuestión.

Cabe destacar que para el desarrollo del estudio se han empleado dos vídeos: slow_traffic_small.mp4 y visiontraffic.avi. El primero de ellos es un vídeo de tráfico en el que se observan coches y peatones en movimiento a través de un puente mientras que el segundo muestra únicamente coches en movimiento desde un punto de vista más cenital, lo que permite una visualización de un mayor número de coches en movimiento.

2. Sustracción de fondo

La sustracción de fondo es una técnica que permite identificar los objetos en movimiento de una secuencia de imágenes, esta técnica devuelve una imagen en la cual se retiran los objetos estáticos, dejando únicamente los objetos en movimiento. La imagen restante puede ser binaria o estar en escala de grises, en la cual lo que se muestra es el objeto en mmovimiento con las sombras del objeto en gris.

2.1. Diferecia de fotogramas

El método más sencillo y rápido de extracción de fondo es la diferencia de fotogramas, en el cual se selecciona un fotograma de referencia y se resta cada uno de los fotogramas de la secuencia a este fotograma de referencia. Para ello se emplea la función *absdiff* de OpenCV que toma una imagen de referencia y una imagen a comparar (ambas en escala de grises) y devuelve una imagen en la que se muestra la diferencia entre ambas imágenes. A continuación se muestra un ejemplo de la sustracción de fondo, a la izquierda la imagen de referencia y a la derecha la imagen resultante de una diferencia de fotogramas:

Figura 1: Frame de referencia de visiontraffic.avi para la diferecia de fotogramas, autoría propia

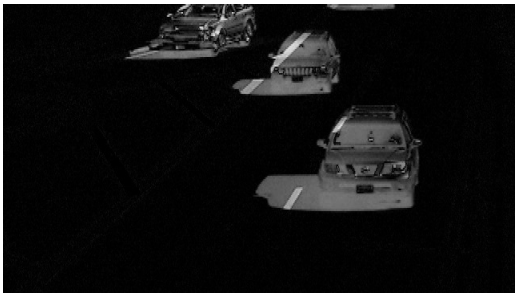

Figura 2: Resultado de la diferencia de fotogramas, autoría propia

Tal y como se puede observar los objetos en movimiento de la imagen de referencia se marcan en la imagen diferenciada en todo momento (a pesar de que no haya movimiento en el otro fotograma diferenciado), también se muestra el objeto en movimiento de la imagen que se ha restado a la imagen de referencia. Es por ello que no es un método muy robusto, ya que marca objetos en movimiento que no lo están, aparte es sensible a sutiles movimientos por lo que puede aumentar

el ruido al identificar árboles meciéndose al viento, no detetcta las sombras y es considerablemente a los cambios de iluminación.

2.2. Mezcla de gaussianas adaptativas

Con la intención de solventar los problemas de la diferencia de fotogramas, surge el método de mezcla de gausianas adaptativas (MOG). Este método se basa en la suposición de que los píxeles de una imagen pueden ser modelados como una mezcla de varias distribuciones gaussianas que representan los distintos valores de intensidad de los píxeles a lo largo del tiempo. Es precisamente una variación en la intensidad lo que permite detectar los objetos en movimiento. El hecho de que se trate de una mezcla de gaussianas adaptativas permite que el modelo se adapte a cambios en la iluminación y en el fondo de la imagen, lo que lo hace más robusto que la diferencia de fotogramas. El proceso de sustracción es el siguiente:

- Se inicializa el modelo con múltiples distribuciones gaussianas para cada píxel que representan sus posibles valores.
- Para cada fotograma se calcula la posibilidad de que un pixel pertenezca al fondo o a un objeto en movimiento.
- Si la probabilidad es menor que un umbral, se actualiza la distribución gaussiana y se asume que el píxel es estático.
- En caso de que la probabilidad sea mayor que el umbral se asume que se representa una parte de un objeto en movimiento, añadiendo una nueva distribución gaussiana al pixel.

Opencv implementa el modelo de las gausianas adaptativas de dos formas distintas las cuales se recogen en el siguiente estudio:

2.2.1. Modelo MOG

El modelo MOG es la implementación original del proceso anteriormente explicado, en el se puede escoger el número de gaussianas que modelan cada píxel, el número de fotogramas que se utilizan para modelar el fondo, el umbral de posibilidad que permite determinar si un píxel pertenece al fondo o a un objeto en movimiento y un parámetro que elimina el ruido. Para estudiar este modelo se hace uso de la clase *bgsegm.createBackgroundSubtractorMOG* de Opencv así como de su método *apply* que permite aplicar el modelo a un fotograma en concreto. El estudio se ha

realizado modificando los parámetros de la clase para obtener distintos resultados almacenando estos en un video de formato mp4 [1]:

- **history** : representa el número de fotogramas que se usan para modelar el fondo, a mayor número de fotogramas mayor robustez del modelo. En el estudio se han usado [100, 150, 200, 250, 300, 350].
- **nmixtures** : es el número de gaussianas a usar por pixel, cuantas más gaussianas más precisión en la detección pero también mayor es el tiempo de computación. En el estudio se han usado [2, 3, 4, 5].
- **backgroundratios** : selecciona el umbral de posibilidad para determinar si un píxel pertenece al fondo o a un objeto en movimiento, a mayor umbral menor sensibilidad tiene a los puntos en movimiento detectando únicamente los de mayor calidad. En el estudio se han usado [0.2, 0.4, 0.6, 0.8].
- **noiseSigma** : fija el parámetro que elimina el ruido, a mayor valor menos ruido se elimina, sin embargo como ya se ha estudiado en múltiples ocasiones el ruido de una imagen, es un problema que soluciona el modelo MOG fijando el parámetro a 0.

2.3. Modelo MOG2

El modelo MOG2 es una variación del modelo MOG que busca detectar sombras en el objeto en movimiento, es por ello que los parámetros modificables varían respecto al anterior modelo:

- **history** : representa el número de fotogramas que se usan para modelar el fondo, a mayor número de fotogramas mayor robustez del modelo. En el estudio se han usado [100, 150, 200, 250, 300, 350].
- **varThreshold** : es un umbral de posibilidad para determinar si un píxel pertenece al fondo o a un objeto en movimiento (distinto al de MOG), a mayor umbral mayor es la calidad de los puntos en movimiento. En el estudio se han usado [10, 20, 30].
- **detectShadows** : selecciona si se detectan sombras o no en el objeto en movimiento, en el estudio se han usado [True, False].

[1] El método *write* de la clase *VideoWriter_fourcc* de Opencv permite almacenar los fotogramas en un video haciendo uso de distintos codecs, se ha decidido usar mp4 debido a que genera vídeos algo menos pesados que avi ya que debido a la alta cantidad de videos generados no se quiso sobrecargar el almacenamiendo del ordenador

En este caso se generaron 36 vídeos en formato mp4 con los distintos parámetros de la clase *createBackgroundSubtractorMOG2* de Opencv, los resultados se compararán en el siguiente apartado junto con los resultados obtenidos con el modelo MOG con el fin de determinar cuál de los dos modelos es más robusto y eficiente. Cabe destacar que se empleó el mismo video que en el estudio del modelo MOG para poder comparar los resultados de ambos modelos.

2.4. Comparación de los modelos MOG y MOG2

Se va a iniciar la comparación con el único parámetro común que tienen ambos modelos el parámetro *history*, que representa el número de fotogramas anteriores usados para evaluar si un pixel pertenece a un objeto en movimiento o no. Ambas implementaciones lo usan de la misma forma, identifica un total de n fotogramas y usa su información exponencialmente pesada para identificar el fondo y el movimiento, de tal manera que el anterior píxel evaluado es el que más afecta a la decisión mientra que el pixel n fotogramas atrás el que menos afecta. Por todo ello se fija el parámetro 350 para ambos modelos como óptimo. En caso de que no haya 350 fotogramas en el video (tal y como sucede hasta la mitad de la evaluación ya que el video usado tiene 531 fotogramas) se usan todos los anteriores con un pesado exponencial.

Por otro lado, al modelo MOG2 hay que aumentarle el parámetro *varThreshold*(al menos a 30) para que no detecte el ruido (el aire en movimiento) presente lo cual dificulta la detección de los objetos en movimiento, así como la identificación de sombras en el móvil (el principal objetivo del modelo):

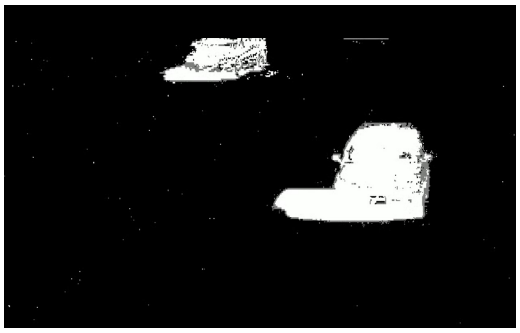

Figura 3: Resultado de la sustracción de fondo con el modelo MOG2 con los parámetros history=350, varThreshold=10 y detectShadows=True, autoría propia

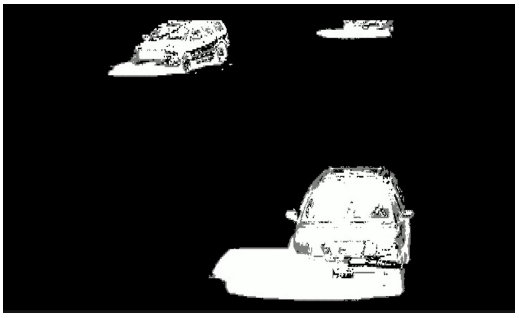

Figura 4: Resultado de la sustracción de fondo con el modelo MOG2 con los parámetros history=350, varThreshold=30 y detectShadows=True, autoría propia

Evaluando el modelo MOG, el primer parámetro a ajustar es el de *nmixtures*, que representa el número de gaussianas que modelan cada píxel, es cierto que a mayor número, mayor precisión, pero también mayor tiempo de computación, por lo que se fija a 5 ya que da resultados buenos sin emplear demasiado tiempo. Por otro lado el parámetro *backgroundratios* se fija a 0.6 ya que al aumentarlo aunque no selecciona el ruido genérico, sí que permite ruido en el propio objeto en movimiento mientras que reducirlo demasiado le resta precisión al modelo:

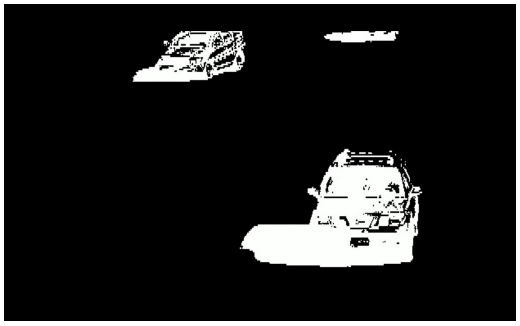

Figura 5: Resultado de la sustracción de fondo con el modelo MOG con los parámetros history=350, nmixtures=5 y backgroundratios=0.2, autoría propia

Figura 6: Resultado de la sustracción de fondo con el modelo MOG con los parámetros history=350, nmixtures=5 y backgroundratios=0.8, autoría propia

La implementación MOG2 es algo más veloz que la implementación MOG siendo la primera capaz de procesar 36 vídeos en unos 2 minutos mientras que la segunda tarda 7 minutos en procesar 96 vídeos, que es algo más que los 6 minutos que tardaría MOG2 en procesar 102 vídeos (calculo realizado a través de una mera regla de tres). Además, MOG2 permite la detección de sombras en el objeto en movimiento, lo cual es un punto a favor. Sin embargo, MOG2 es más sensible al ruido que MOG, teniendo MOG dos formas de reducir el ruido de la imagen, una por filtrado gaussiano y otra reduciendo o aumentando la sensibilidad del modelo mientras que MOG2 sólo permite la segunda opción. Además, observando el conjunto de vídeos generados, parece que MOG es más preciso que MOG2 a la hora de identificar los detalles de los objetos en movimiento tal y como se puede apreciar a continuación:

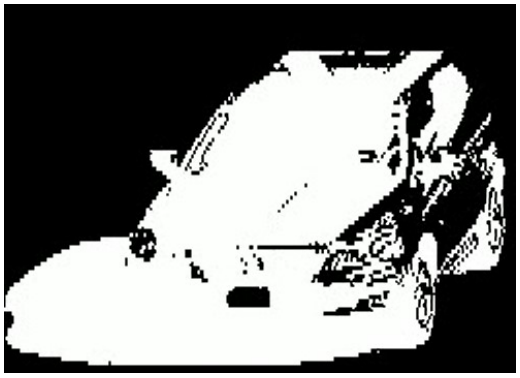

Figura 7: Resultado de la sustracción de fondo con el modelo MOG con los parámetros history=350, nmixtures=5 y backgroundratios=0.6, autoría propia

Figura 8: Resultado de la sustracción de fondo con el modelo MOG con los parámetros history=350, varThreshold = 30 y detectShadows = True, autoría propia

3. Flujo óptico

El flujo óptico es la detección de movimiento de los objetos en una secuencia de imágenes, es decir, la variación de la posición de los objetos en el tiempo. Estq definición es también la de derivada espacial de la imagen, es decir, la variación de la intensidad de los píxeles en el tiempo. Por ello se asume la siguiente ecuación:

$$I(x, y, t) = I(x + ux, y + vy, t + 1) \tag{3.1}$$

Tomando la aproximación de series de Taylor de primer orden (equivalente a una primera derivada) se obtiene lo siguiente:

$$\begin{gathered} I(x, y, t + 1) + I_x u + I_y v - I(x, y, t) \approx 0) \\ (I(x, y, t + 1) - I(x, y, t)) + I_x u + I_y v \approx 0 \\ I_t + I_x u + I_y v \approx 0 \end{gathered}$$

En un limite de tiempo inifinitesimal en el que el tiempo, u y v tienden a 0 se obtiene la siguiente ecuación exacta:

$$I_x u + I_y v + I_t = 0 \tag{3.2}$$

Eso implica que cada punto del flujo óptico del pixel (x,y) estará sobre la recta del espacio (u,v) enunciada por la ecuación anterior. Sin embargo este método no es robusto ya que se necesita al menos una condición más para poder resolver el sistema de ecuaciones. Por ello se usa una modificación de la ecuación anterior, el método de Lucas-Kanade que asume que el flujo óptico es constante entorno a una pequeña bola que rodea el pixel (x,y). Esto permite crear un sistema de ecuaciones que se puede resolver con el método de mínimos cuadrados. Ejemplificando, el sistema propuesto para el método con una vecinidad de 3x3 alrededor del pixel sería la siguiente:

$$\begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ I_x(x_2, y_2) & I_y(x_2, y_2) \\ \vdots & \vdots \\ I_x(x_9, y_9) & I_y(x_9, y_9) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(x_1, y_1) \\ -I_t(x_2, y_2) \\ \vdots \\ -I_t(x_9, y_9) \end{bmatrix}$$

Es por ello que este método no funciona bien para zonas planas o sin cambios ya que las derivadas de la imagen en el conjunto son 0.

3.1. Implementación del flujo óptico

La detección de flujo óptico inicia extrayendo los puntos a seguir de una primera imagen en escala de grises, en un primer momento se toma el primer fotograma. Posteriormente se usa la función *goodFeaturesToTrack* de OpenCV que permite detectar los puntos de interés a partir de una imagen en escala de grises. La función consta de una serie de parámetros para realizar un correcto ajuste de los puntos de interés:

- **mask** : máscara de la imagen en la que se detectan los puntos de interés, en este caso no se ha usado ninguna máscara para que tenga acceso a todos los puntos.
- **maxCorners** : número máximo de esquinas a detectar, en este caso se ha fijado a 100 para que no sea un limitante.
- **qualityLevel** : calidad mínima aceptada para las esquinas de la imagen, se ha decidido ajustar el número de puntos a través de la calidad en lugar de la cantidad. Con un valor de 0.38 se detectan menos de 100 puntos, pero de gran calidad todos. Se hicieron pruebas hasta 0.9 momento en el cual se dejaron de detectar puntos, por lo que se fue descendiendo hasta que se obtuvo un número suficiente de puntos de calidad.
- **minDistance** : distancia mínima euclídea entre los píxeles detectados, se ha fijado a 5 para que no se detecten puntos superpuestos o que sigan un mismo objeto.
- **blockSize** : tamaño del bloque medio para realizar la derivada de la imagen, se ha fijado a 7 para que sea un tamaño medio.

Además tiene dos parámetros para aplicar el detector de Harris aunque se ha dejado en 0 para que use el de mínimos autovalores. Una vez que se tienen los puntos de interés se aplican los siguientes pasos que conforman el método de Lucas-Kanade para cada fotograma:

- Se dispone el fotograma en escala de grises.
- Cálculo del flujo óptico mediante la función *calcOpticalFlowPyrLK* de OpenCV que toma como parámetros la imagen anterior, la imagen actual, los puntos de interés de la imagen anterior así como los puntos de interés de la imagen actual (None). También emplea el tamaño de la ventana de bíusqueda que se ha fijado a (5,5) de tal manera que se buscan los puntos de interés en un radio de 5 píxeles alrededor del punto de interés anterior ya que si se hacía

más grande algunos puntos de interés dejaban de seguir al objeto por otro que pasara cerca. Además también se ha fijado un número máximo de pirámides así como una serie de flags que permiten una mejor y más rápida detección de los puntos de interés.

- Se seleccionan los puntos bien identificados.
- Se dibujan las trayectorias (en caso de que se pueda) del movimiento de los puntos de interés.
- Actualización de la imagen interior por la actual en escala de grises y de sus puntos de interés.
- Cada 50 fotogramas se verifica que no hay puntos de interés nuevos aplicando de nuevo la función *goodFeaturesToTrack* para que se detecten nuevos puntos de interés. Tan sólo se modifican 3 parámetros, la distancia entre puntos nuevos a 20, la calidad a 0.8 para que sólo se identifiquen los puntos con mayor calidad y se pasan los puntos de interés actuales como máscara para que no se detecten puntos en los mismos lugares. El objetivo es que se detecten nuevos puntos de interés en caso de que los anteriores se hayan perdido o hayan aparecido nuevos puntos muy relevantes en la imagen.
- Finalmente se muestra el fotograma con las trayectorias y los puntos marcados.

A continuación se muestra el resultado de la extracción de flujo óptico en el vídeo $slow\_traffic\_small.mp4$:

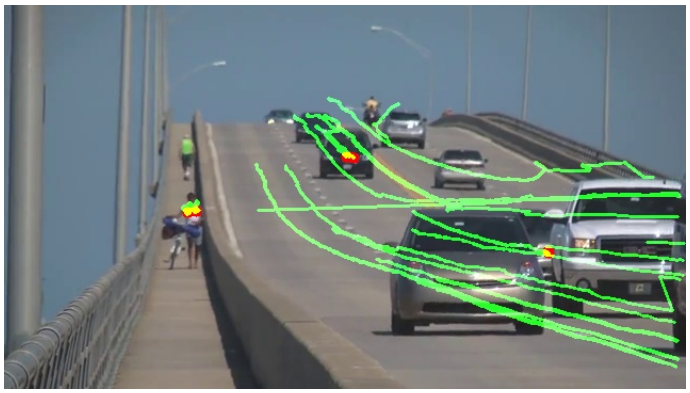

Figura 9: Flujo óptico en slow_traffic_small.mp4, autoría propia

Tal y como se puede observar aunque el grueso de puntos que seguían a los coches iniciales hayan abandonado la imagen por la derecha, se han localizado nuevos puntos de interés. Además debido a las pocas convoluciones que se observan en el camino de los puntos de seguimiento se puede afirmar que el flujo óptico ha sido exitoso y estable (en parte gracias a la ventana de búsqueda de 5X5 píxeles).

4. Seguimiento de objetos a través del filtro de Kalman

En la anterior sección se ha estudiado cómo realizar el seguimiento de puntos de interés a partir de una secuencia de imagenes pasadas, el flujo óptico. El filtrado de Kalman es una técnica que permite no seguir sino predecir la posición de un objeto en movimiento a partir de una serie de observaciones pasadas. El filtro de Kalman se vale del teorema de Bayes para predecir la posición de un conjunto de píxeles en un fotograma futuro a partir de la posición fotogramas pasados, actualizando la predicción y la covarianza de la predicción a partir de la observación del fotograma actual. El filtro de Kalman consta de dos fases, la predicción, actualización, medida y corrección, cabe destacar que tanto el prior como el posterior de dichas fases están modelados por gaussianas:

- **Predicción** : se predice la posición futura del objeto en movimiento a partir de la posición actual y de la velocidad del objeto. Para ello se emplea la siguiente ecuación:

$$x_t = F_t x_{t-1} + \epsilon \tag{4.3}$$

  Donde x_t es el estado del objeto en el instante t, F es la matriz de transición de estados en el instante t, x_{t-1} es el estado del objeto en el instante k-1, y ϵ es el ruido del proceso.

- **Actualización** : se actualiza la predicción a partir de la observación del fotograma actual. Para ello se emplea la siguiente ecuación para la media:

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} \tag{4.4}$$

  Mientras que para la covarianza se emplea la siguiente ecuación:

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_{t-1} \tag{4.5}$$

  Donde Q_t es la matriz de covarianza del proceso en el instante t.

- **Medida** : se mide la posición del objeto en el fotograma actual a partir de la observación del fotograma actual. Para ello se emplea la siguiente ecuación:

$$z_t = H_t x_t + \epsilon_2 \tag{4.6}$$

  Donde z_t es la observación del objeto en el instante t, H es la matriz de medición en el instante t y ϵ_2 es el ruido de la observación.

- **Corrección** : se corrige la predicción a partir de la observación del fotograma actual. Para ello se emplean dos ecuaciones, una primera que explica cuánto

se ha equivocado la predicción y otra que actualiza la predicción en base a la observación y a la equivocación:

$$K_t = P_{t|t-1} H_t^T H_t^T S_t^{-1} \tag{4.7}$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{4.8}$$

Donde K_t es la ganancia de Kalman en el instante t (nivel de confianza en la predicción), S_t es la covarianza de la observación en el instante t, I es la matriz identidad y P_t es la covarianza de la predicción en el instante t.

4.1. Implementación del filtro de Kalman

La implementación del filtro de Kalman se ha realizado en dos pasos y se ha aplicado sobre el vídeo $slow_traffic_small.mp4$, en primer lugar se selecciona un área a seguir de un fotograma y posteriormente se hace el seguimiento. El filtro de Kalman creado a través de la clase *KalmanFilter* de OpenCV recibe dos parámetros, el número de variables de estado y el número de dimensiones de la observación. En este caso se ha fijado a 4 y 2 respectivamente ya que se está siguiendo un objeto en movimiento en un plano 2D. Las variables de estado representan la posición y la velocidad del objeto (en ambos ejes). Definido el filtro, se deben definir tres matrices (mencionadas en el fundamento matemático del apartado anterior):

- **F** : matriz de transición de estados, define cómo se calcula el prior en el instante a partir de los anteriores, si se fija a la matriz identidad se asume que el objeto se mueve de manera constante, es por ello que rápidamente se pierde la referencia del objeto en movimiento (en este caso el capó del coche blanco). Para solucionar este problema, se ha fijado la matriz a la siguiente:

$$\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  De esta manera se tiene en cuenta no sólo la posición del objeto en movimiento sino también su velocidad. Si en la anterior matriz se define dt como el tiempo entre fotogramas, produce un mejor seguimiento, ya que se tiene en cuenta la velocidad de forma proporcional en el cálculo de la posición actual.

- **H** : matriz de medición, mide cuál es la diferencia entre la predicción realizada y la realidad para poder ajustar el modelo. Tiene únicamente dos filas debido a que no se posee una forma de medir la velocidad del objeto a lo largo de los ejes, es por ello que tiene una fila asignada a cada variable de estado medible:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- **Q** : la matriz de covarianza es la que le aporta flexibilidad al modelo, ya que permite decidir cuánta importancia se le da a la predicción en el cálculo de la siguiente posición. Se usa la matriz identidad ya que se asume que el modelo tiene un error uniforme en todas las variables de estado. Sin embargo esta matriz identidad se multiplica por 0.1 de tal manera que se le da más importancia a la observación que a la predicción ya que se considera que el coche se va a desplazar de manera estable conforme al tiempo.

A continuación se muestran dos imágenes, una que muestra una matriz de transición identitaria y otra que muestra una matriz de transición que tiene en cuenta la velocidad del objeto. Cabe destacar que la variación de la matriz de transición es la diferencia entre un filtro de Kalman y un filtro extendido de Kalman que pretende parametrizar objetos en movimiento no lineales:

Figura 10: Filtro de Kalman con matriz de transición unitaria con seguimiento sobre el capó, autoría propia

Figura 11: Filtro de Kalman con matriz de transición que tiene en cuenta la velocidad del objeto con seguimiento sobre el capó, autoría propia

Tal y como se puede observar se realiza un seguimiento más estable y preciso incluso cuando el objeto se acerca al tener en cuenta la velocidad.

Inicializado el filtro con los filtros adecuados, se emplea la función *selectROI* que recibe un fotograma y permite seleccionar un área de interés en la imagen, esta devuelve el área seleccionada como el origen del área y su tamaño en cada eje (se recuerda que en visión por ordenador, por convenio y porgramación el origen de coordenadas se encuentra en la esquina superior izquierda de una imagen). Una vez seleccionada el área de interés, se calcula el centro de ese área que junto con $v_x = v_y = 0$ conforma el estado inicial de las variables de estado, a su vez se inicializa la matriz de covarianza y finalmente se predice la futura posición con el método *predict*. Para finalizar la parte de identificación del primer punto sólo resta corregir la medida con el método *correct* que toma el centro del área seleccionada, recortar el área de interés, tranformarla a escala HSV, calcular el histograma de la

imagen y normalizar dicho histograma. El histograma permite la aplicación de una máscara, pero no se usa para limitar las fallas del filtro por una variación en el píxel. Con este mismo objetivo el filtro se realiza sobre el canal "Hue" de la imagen que es el que más información aporta sobre el color del píxel seguido.

Seleccionado y parametrizado el primer punto de interés, se procede a aplicar un bucle que recorre todos los fotogramas del video y que sigue los siguientes pasos:

- Conversión del filtro de Kalman a un fotograma en escala HSV.
- Cómputo de la proyección hacia atrás del filtro de Kalman con la función *calcBlanckProject* de OpenCV que toma como parámetros la imagen en HSV, el canal de la imagen sobre el que se ha aplciado el histograma, el histograma normalizado, así como el rango del histograma para obtener una respuesta de cómo de bien se ajusta la imagen al histograma.
- Recalculación de la ventana de seguimiento así como de su centro.
- Actualización de la medida del filtro a través del método *correct* que toma como parámetro el centro de la ventana de seguimiento.
- Muestreo del punto predicho, el punto observado, el fotograma y la ventana de seguimiento.