

Homework #10: Object detection and tracking using image keypoints and descriptors.

Visión Computacional 14:30 hrs



Alumno:
Juan Carlos Chávez Villarreal 543653

Instructor del curso:
Phd. Andres Hernandez Gutierrez

"Doy mi palabra de que he realizado esta actividad con integridad académica"

Homework #10: Object detection and tracking using image keypoints and descriptors.

Visión Computacional 14:30 hrs
Universidad de Monterrey

1 Introducción

El uso de técnicas de visión para la detección y seguimiento de objetos ha generado un interés considerable en varias industrias, gracias a sus numerosas aplicaciones y su capacidad para aumentar la eficiencia, automatización y seguridad.

Existen varios algoritmos que pueden ser empleados para realizar detección de objetos, como el algoritmo SIFT. SIFT es un algoritmo de visión computacional utilizado para detectar y describir características locales en imágenes. El método identifica puntos de interés invariables a escala, rotación y cambios de iluminación, lo que permite el reconocimiento robusto de objetos en diferentes entornos y condiciones.

2 Planteamiento del Problema

Para esta práctica se tiene como objetivo principal la identificación y seguimiento de un objeto en un plano 2D utilizando el algoritmo SIFT (Scale-Invariant Feature Transform) y cualquier otra herramienta previamente obtenida en el curso. Algunas de estas herramientas serán la segmentación de características de una imagen usando el estado de color HSV (Hue saturation Value) y dibujar líneas y figuras geométricas con funciones de Open CV. Adicionalmente, se debe analizar y desplegar la cantidad de veces que el objeto analizado cruza por una línea trazada por el usuario en el plano.

Debido a que la selección del objeto a analizar quedó a criterio de alumno, se decidió realizar el seguimiento de un perro en un patio. Pensando en replicar una aplicación utilizada en la vida diaria de detección y seguimientos de objetos se decidió realizar el seguimiento de un jugador en un campo. El jugador seleccionado será un perro de nombre Arlo y portará un identificador en su espalda con su nombre y un número aleatorio (simulando la espalda de una jersey deportiva de un jugador de fútbol).

3 Materiales

Para esta práctica se necesitarán los siguientes materiales:

- Computadora personal
- Ratón de computadora.
- Cámara monocular.
- Github (Da Click [aquí](#) para ver el repositorio)
- Identificador que portará el perro
- Imagen del perro con su identificador (arlo-identifier-real5.jpg).
- Video del patio donde se observa al perro en movimiento (dogs_in_patio4.mp4).
- Python 3.10 o superior junto con las siguientes librerías: OpenCV, NumPy y ArgParse.
- Visual Studio Code

4 Metodología

Como se mencionó anteriormente, en esta práctica se busca identificar y seguir un perro en un patio utilizando el algoritmo SIFT. En la Figura 1 podemos observar un diagrama de flujo que describe de manera simple y visual la metodología empleada para cumplir con todos los criterios mencionados en el planteamiento del problema. A lo largo de esta sección del reporte se explicará cada paso de este diagrama.

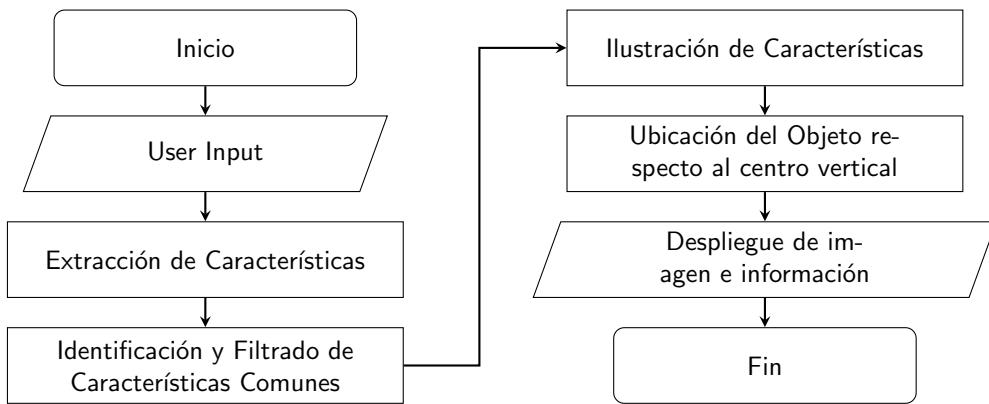


Figure 1: Diagrama del flujo para identificación y seguimiento de un objeto

4.1 User Input

En la sección "user input" recolectamos la imagen del objeto a identificar y seguir (en nuestro caso la imagen del perro) y el vídeo en el cual identificaremos nuestro objeto a seguir. Para esto haremos uso de la librería argparse de python la cual nos permite especificar los argumentos que será requeridos y luego proporciona una forma de parsear esos argumentos de una manera estructurada. En la figura ?? podemos apreciar la función encargada de realizar esta tarea y en la figura 2 podemos observar el diseño digital del identificador (figura 2a) y la imagen de referencia que será utilizada (figura 2b).



(a) Identificador digital



(b) Identificador montado en el perro

Figure 2: Diseño e implementación del Identificador

4.2 Extracción de Características

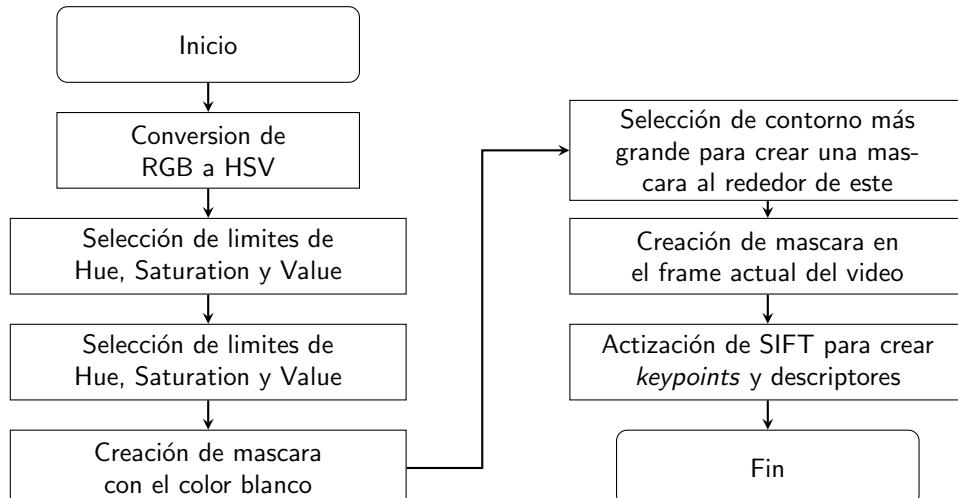


Figure 3: Diagrama del flujo para Extracción de Características

Para la extracción de imágenes se decidió trabajar en el espacio de color HSV para segmentar el identificador del resto de la imagen. Esto se hizo porque al procesar la información en cualquier otro estado de color producía una muy alta cantidad de *keypoints* en todo el suelo y resultaba en muchos resultados de tipo "falso-positivo". Al crear una máscara que extraiga únicamente el color blanco quedamos solo con la información del identificador y ruido como los tenis de quien paseaba al perro y las partes blancas del pelaje del perro)(como podemos observar en la figura (4a). Una vez creada esta máscara blanca, eliminó el ruido buscando el contraste más grande el cual fue utilizado para crear una segunda máscara que sería aplicada al *frame* del video para segmentar el identificador y poder tener una mejor creación de puntos y descriptores de la imagen de referencia y el video (figura (4b)).

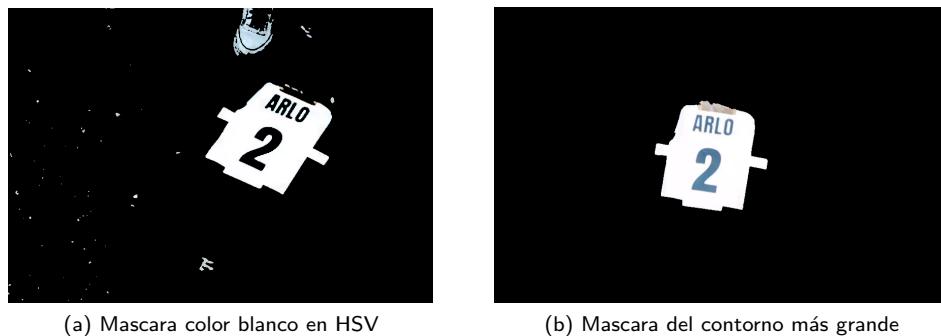


Figure 4: Mascaras para segmentación de colores

4.3 Identificación y Filtrado de Características Comunes

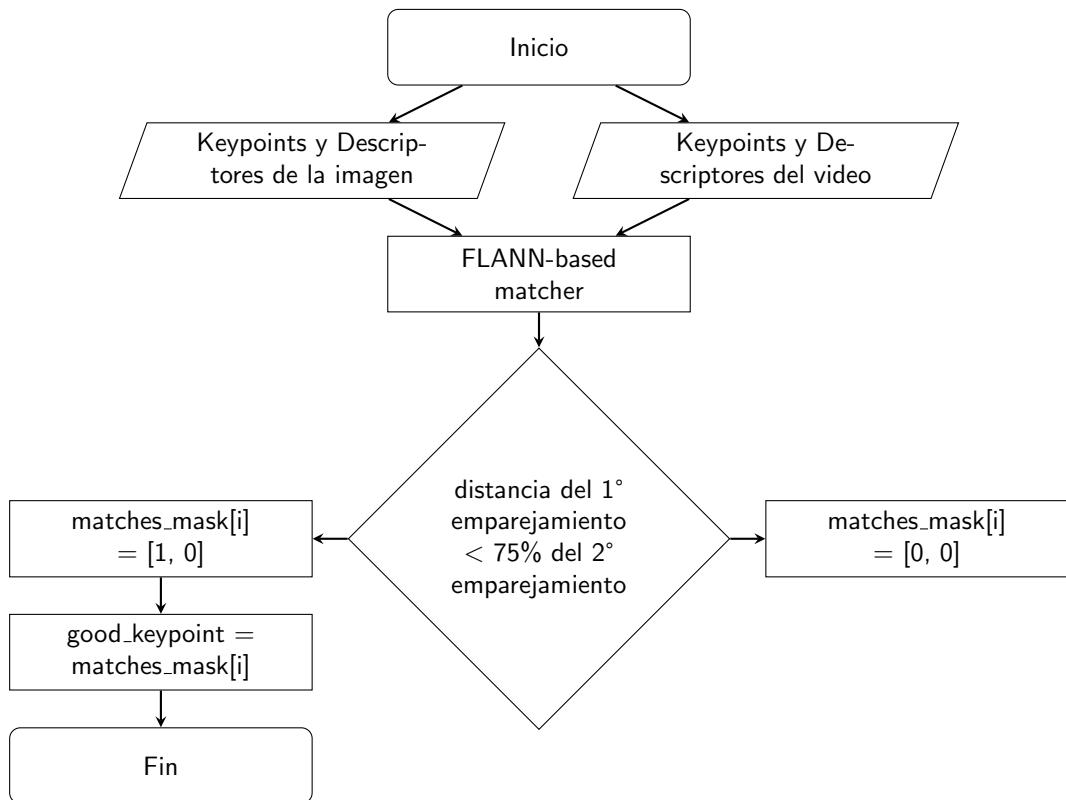


Figure 5: Diagrama del flujo para Identificación y Filtrado de Características Comunes

La función *match_features()* es utilizada para encontrar coincidencias entre los *keypoints* de la imagen de referencia y el *frame* actual. Para esto se realizó un filtro de Lowe, el cual compara las distancias de los dos emparejamientos más próximos de un *keypoint*. Se considera que un emparejamiento es significativo si la distancia de dicho emparejamiento es menor que el 75% de la distancia de un segundo emparejamiento. Si un emparejamiento es bueno, se registra un *[1,0]* en la variable *matches_mask*. Este método fue efectivo para distinguir entre emparejamientos ambiguos y aquellos que están claramente diferenciados.

Posteriormente en la función `extract_good_points()` identifica todos los emparejamientos que identificamos y los agrega a un nuevo arreglo llamado: `good keypoints`. (Figura 5)

4.4 Ilustración de Características

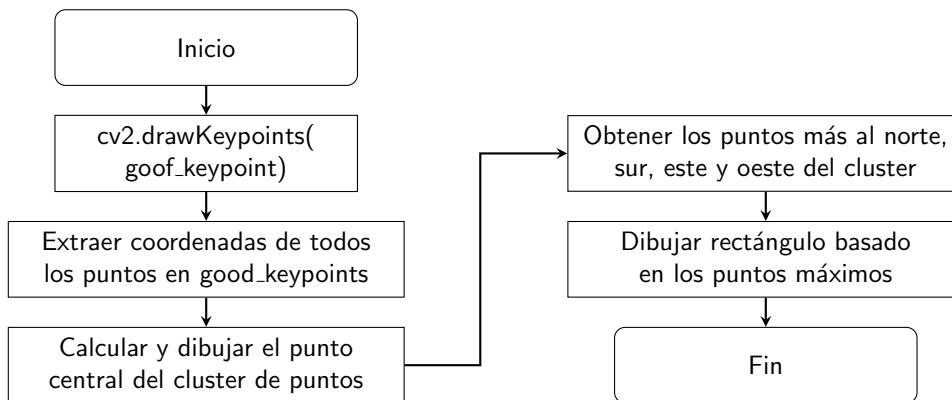


Figure 6: Diagrama del flujo para Extracción de Características

En la función `draw_characteristics()` se realizan la identificación de todas las características que son de relevancia para nosotros, las cuales serían:

- Keypoints que coinciden entre la imagen de referencia y el vídeo
- El punto central del cluster de keypoints.
- Rectángulo rojo al rededor del objeto identificado

Usando la función `cv2.drawKeypoints()` se dibujaron los `keypoints` que coinciden entre la imagen de referencia y el vídeo. Posteriormente se extrajeron y almacenaron las coordenadas de todos los `keypoints` para poder encontrar el punto promedio entre todos los puntos del cluster y así obtener el punto central del objeto. Para poder dibujar un rectángulo al rededor del objeto usando la función `cv2.rectangle()` primero se calcularon los puntos extremos de este cluster, es decir, el punto mas al norte, sur, este y oeste para poder crear dos pares de coordenadas indicando el inicio y el fin del rectángulo.

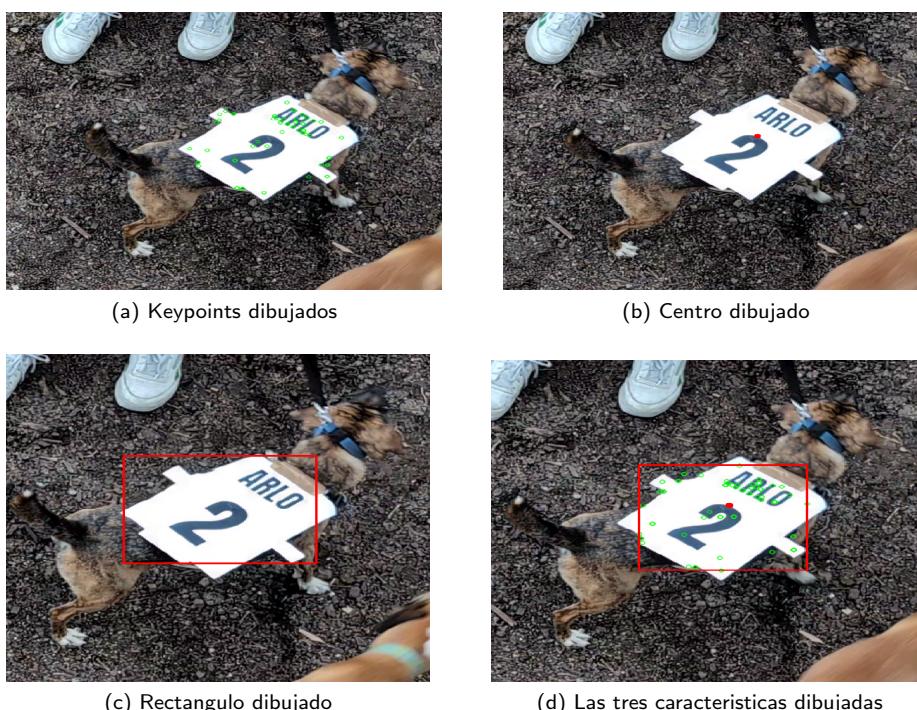


Figure 7: Diseño e implementación del Identificador

4.5 Ubicación del Objeto respecto al centro vertical

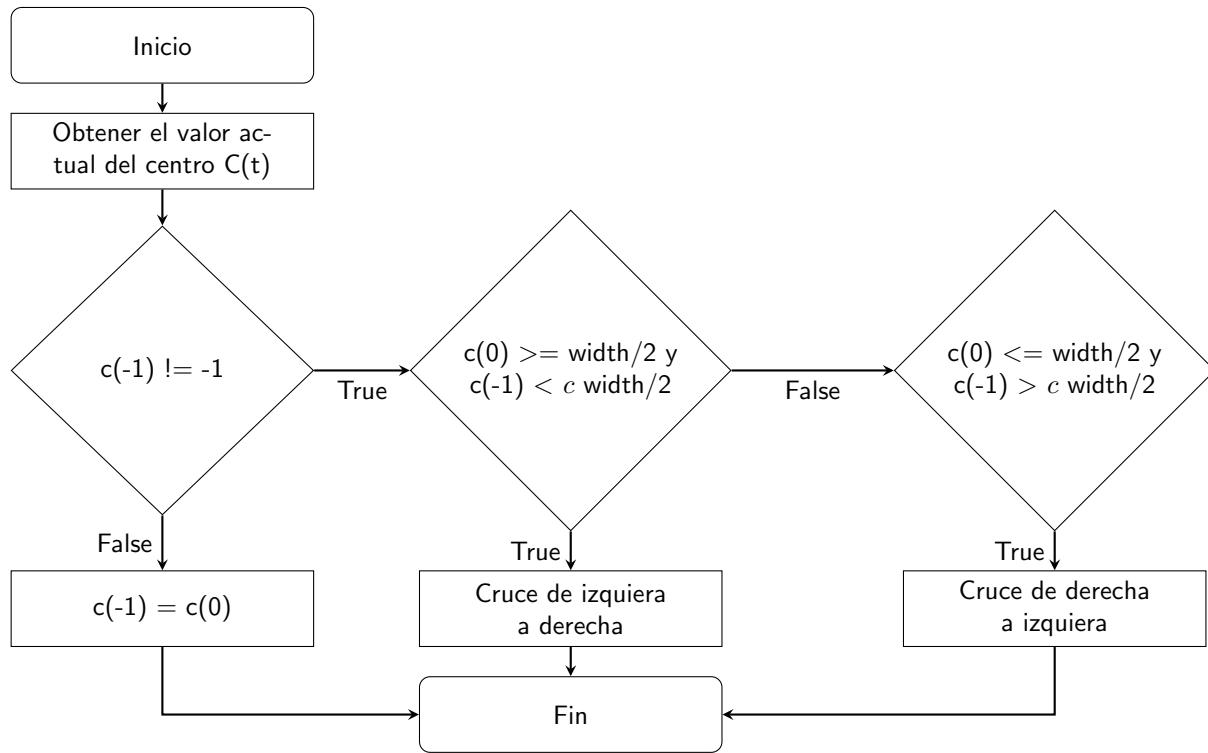


Figure 8: Diagrama del flujo para Extracción de Características

Para determinar cuando el objeto cruza el centro vertical del video se creó una variable de memoria para almacenar la posición del centro en tiempo 0 y -1. Considerando que al iniciar el programa habrá una breve instante en la que no exista una posición en tiempo -1, se estructuró una condicional que detecte si existe o no una posición anterior. En caso de que esta si exista, se verificará si el punto actual tiene una posición en el eje x mayor o igual al centro y si el punto anterior tiene una posición menor a la mitad. Si esta condición se cumple, se sumará el contador de cruces de izquierda a derecha. Por otro lado si el punto actual tiene una posición en el eje x menor o igual al centro y si el punto anterior tiene una posición mayor a la mitad se sumará el contador de cruces de derecha a izquierda.

4.6 Despliegue de imagen e información

Finalmente, utilizando la función `cv2.putText()` desplegamos el número de veces que el objeto identificado cruce por el centro vertical.

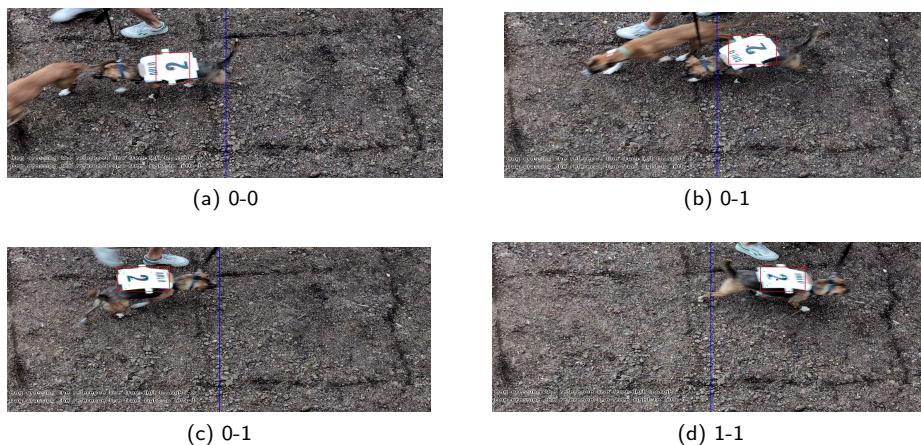


Figure 9: Cruce del objeto por la pantalla

5 Conclucion

Esta fue una tarea muy retadora que consumió mucho mas tiempo del esperado. Sin embargo me siento muy orgulloso de todos los resultados, tanto del código y del reporte. Considero que a lo largo del semestre he aprendido técnicas y herramientas de visión computacional y esta tarea me ha permitido verdaderamente ponerlas a prueba ya que use prácticamente todo lo que he visto a lo largo de este curso. Sin duda alguna la mejor parte de esta tarea fue enseñársela a mi familia y novia y ver sus caras de sorpresa al ver el resultado, porque a pesar de verse relativamente simple, llevó mucho esfuerzo y trabajo.