Visión por Computador

Práctica 3

Curso 2020-2021

Javier Gálvez Obispo
javiergalvez@correo.ugr.es

Ejercicio 1

En este ejercicio nos piden implementar una función que obtenga los puntos Harris para dos imágenes distintas de Yosemite.

Figure 1: Yosemite1.jpg

Figure 2: Yosemite2.jpg

Para ello, primero construimos la pirámide Gaussiana de nivel 3 para cada imagen. Una vez la tenemos, detectamos el valor del criterio Harris en cada pixel que se calcula con la siguiente fórmula:

$$f(x,y)=\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}$$

Obtenemos los eigenvalues (λ_1, λ_2) para cada punto (x, y) con la función *cornerEigenValsAndVecs* de **OpenCV** utilizando como parámatros *blockSize* = 3 y *kSize* = 3.

Una vez conocemos el valor de f para cada pixel, realizamos supresión de no máximos. Vamos recorriendo la matriz con los valores de f y para cada posición en el que se supere un umbral fijado a 10, comprobamos si es el valor máximo en una ventana de 7x7 centrada en ese punto.

Con estos parámetros obtenemos 4496 puntos de interés para Yosemite1 y 4597 para Yosemite2. Para no saturar la imagen a la hora de pintar todos los puntos nos quedamos con los 2000 con mayor valor f de entre los obtenidos, manteniendo la proporción 70%-25%-5% de puntos por escala indicada en el apartado.

Los valores asignados a los distintos parámetros son los indicados en [1] excepto para el tamaño de la ventana utilizada en la supresión de no máximos, que se ha aumentando para que no hubiese una acumulación de puntos en ciertas regiones de las imágenes. También se ha probado con otros valores de los parámetros para obtener directamente un conjunto cercano a los 2000 puntos pero en esos casos no se conseguía una buena representación de puntos para todas las escalas.

Mostramos los máximos locales obtenidos en cada escala para las dos imágenes de Yosemite.

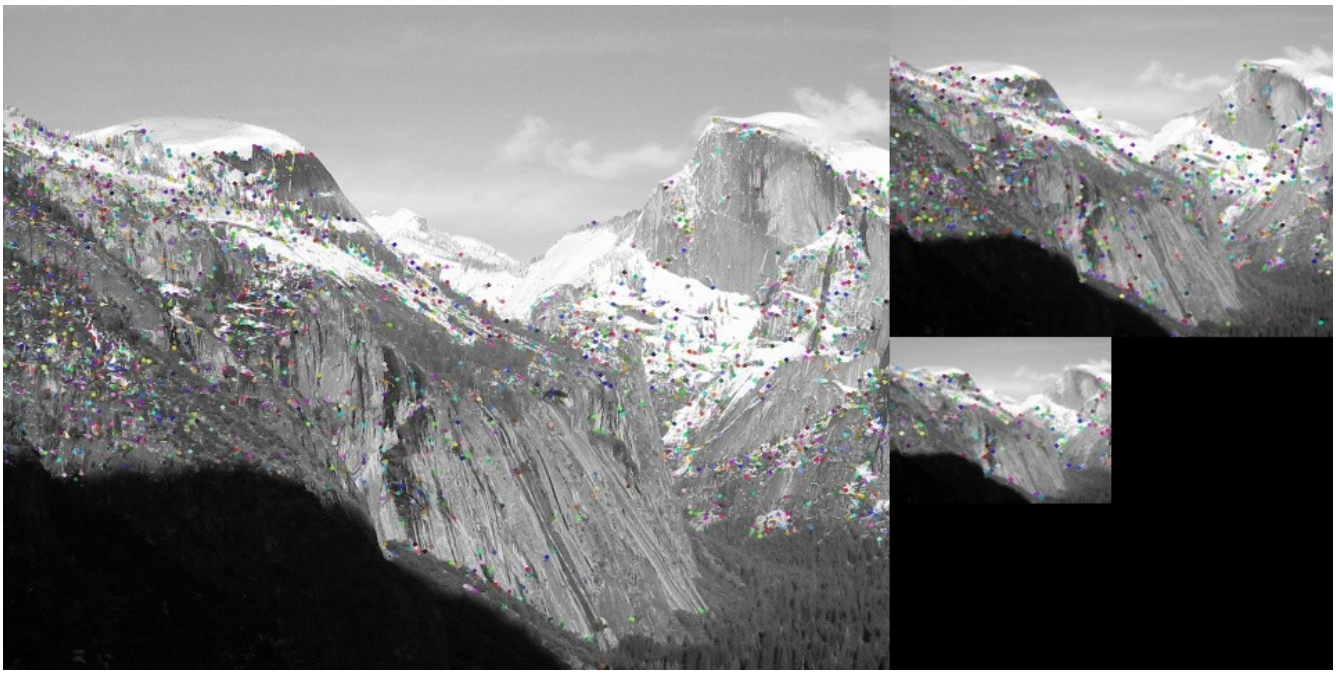

Figure 3: Máximos locales obtenidos en las distintas escalas de Yosemite1.jpg

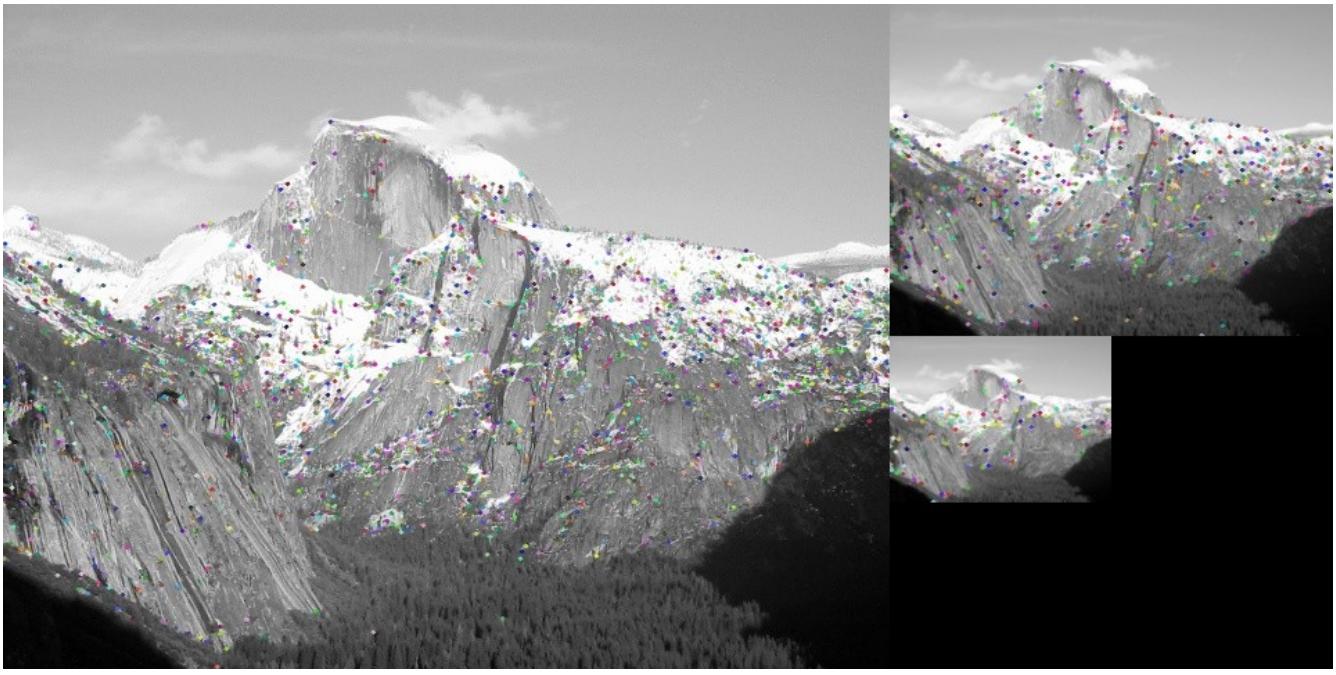

Figure 4: Máximos locales obtenidos en las distintas escalas de Yosemite2.jpg

Vemos que en los dos casos, los puntos obtenidos son representativos de toda la imagen ya que, aunque hay algunas aglomeraciones en ciertas zonas, en general los puntos están repartidos por toda ella mostrando así las esquinas más importantes de la montaña.

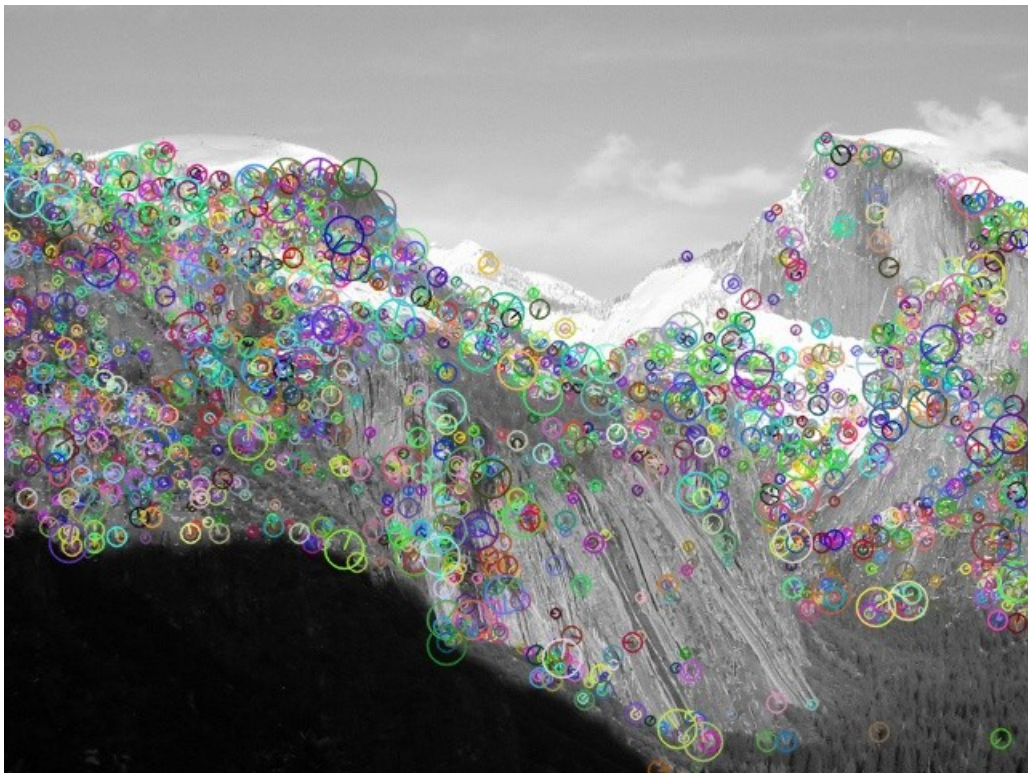

Figure 5: Todos los puntos obtenidos en Yosemite1.jpg

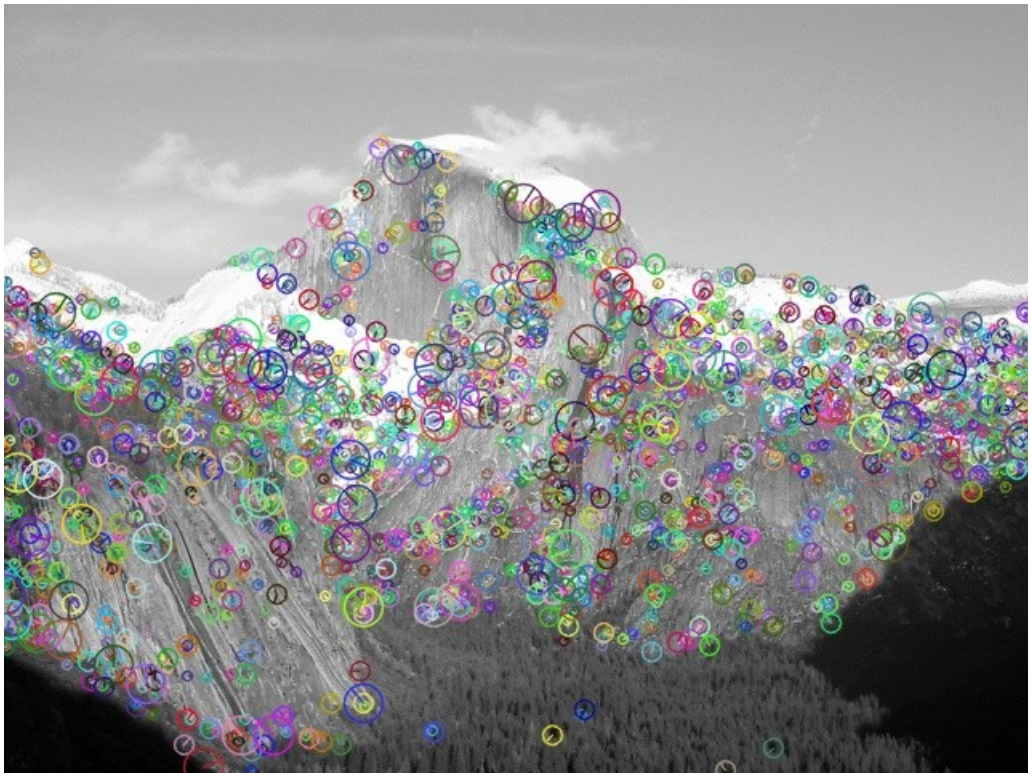

Figure 6: Todos los puntos obtenidos en Yosemite1.jpg

En estas dos últimas imágenes se muestran los puntos obtenidos en todas las escalas sobre la imagen original. El tamaño de cada círculo refleja el nivel de la pirámide en el que se ha detectado. Se utiliza la siguiente fórmula para calcularlo:

$$radio = 2^{nivel+1} * blockSize$$

Vemos que aquellos puntos que han sido detectados en el nivel más alto de la pirámide, los de mayor radio, aparecen siempre en lugares donde hay una aglomeración de puntos detectados en el resto de escalas. Esto es algo esperable, ya que, por cada nivel de la pirámide se van perdiendo detalles y por tanto solo quedan las esquinas más pronunciadas de toda la imagen.

Por último, realizamos un refinamiento sobre las coordenadas de todos los puntos detectados para mejorar la precisión de sus coordenadas. Solamente permitimos que los puntos se muevan en una región de tamaño 7x7.

Para poder ver el cambio producido por el refinamiento, elegimos tres puntos de forma aleatoria y mostramos un entorno 9x9 centrado en ellos con un zoom de x10. Pintamos de rojo el punto original y de azul el punto corregido.

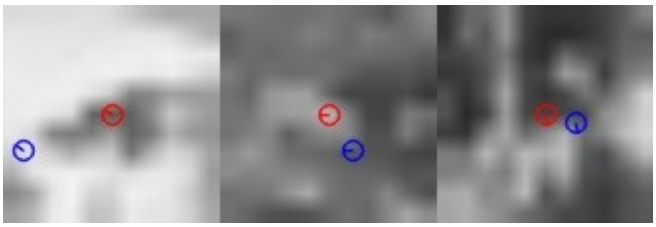

Figure 7: Ejemplo refinamiento Yosemite1.jpg

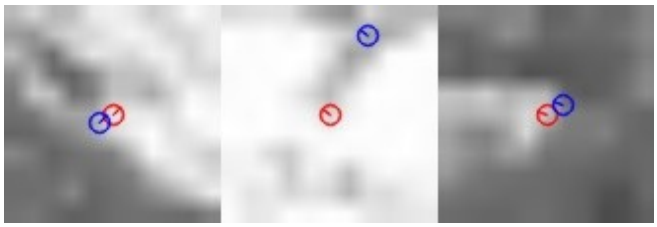

Figure 8: Ejemplo refinamiento Yosemite2.jpg

Como podemos ver, el posicionamiento de los puntos ha mejorado respecto a su posición inicial, no obstante, hay casos en los que es complicado ver la mejoraría. Para ello, mostramos el resultado del mismo proceso en una de las imágenes de pruebas en las que se muestra una rejilla.

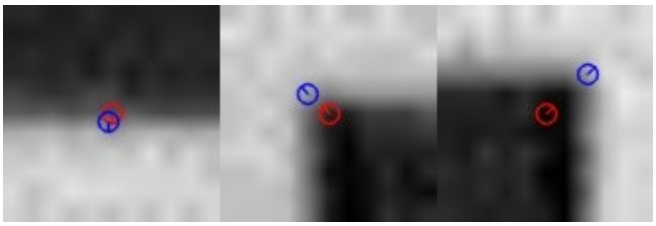

Figure 9: Ejemplo refinamiento rejilla

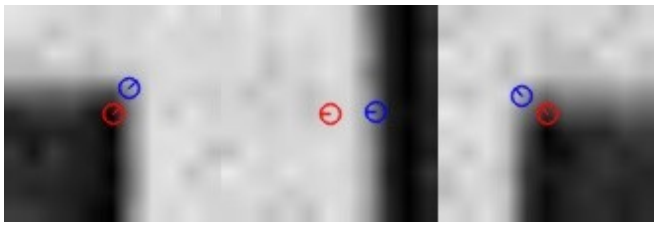

Figure 10: Ejemplo refinamiento rejilla

En este caso si que se ve fácilmente como aumenta la precisión de los puntos tras el refinado.

Ejercicio 2

En este ejercicio nos piden establecer las correspondencias entre las dos imágenes anteriores de Yosemite utilizando AKAZE para extraer los descriptores y usando dos criterios distintos para hallar las correspondencias. Para ambos casos mostramos un subconjunto aleatorio de todas las correspondencias establecidas.

El primer método será Brute Force más Cross Check en el que cada punto de la primera imagen se corresponde con el punto con el descriptor más cercano en la segunda. Al incluir Cross Check la correspondencia debe darse en ambos sentidos.

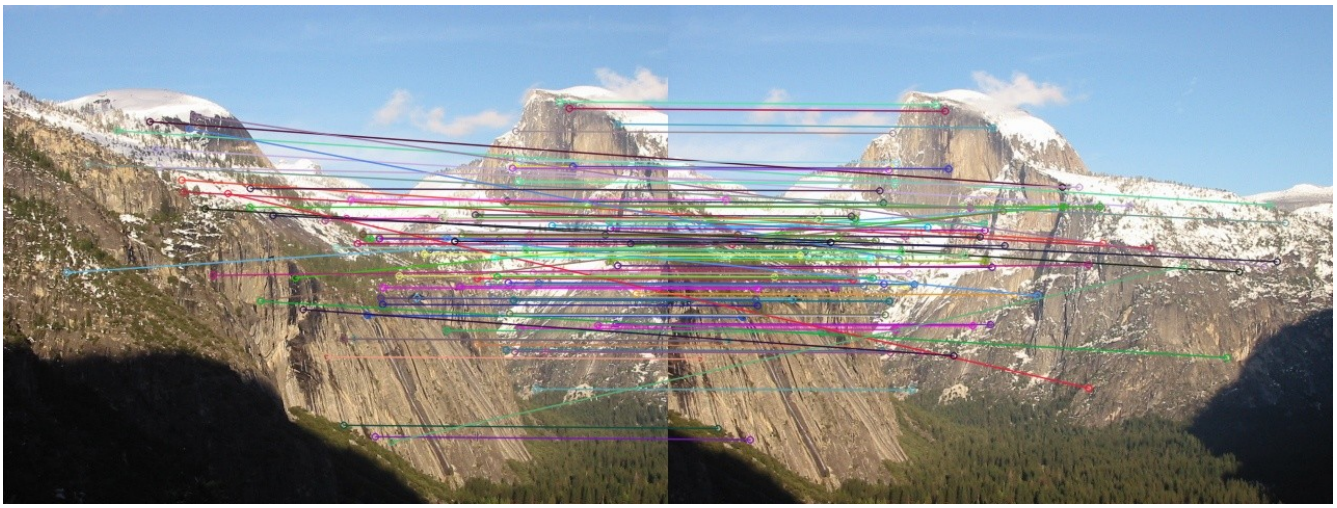

Figure 11: Resultados correspondencias Yosemite utilizando Brute Force + Cross Check

El segundo criterio a utilizar es Lowe Average 2NN. En este caso se obtienen los dos descriptores más cercanos y si el cociente entre la distancia del más cercano y el segundo es menor que un umbral fijado entonces establecemos la correspondencia. Conseguimos así eliminar la ambigüedad. El umbral utilizado ha sido 0.75 siguiendo como ejemplo el tutorial proporcionado por **OpenCV** [2].

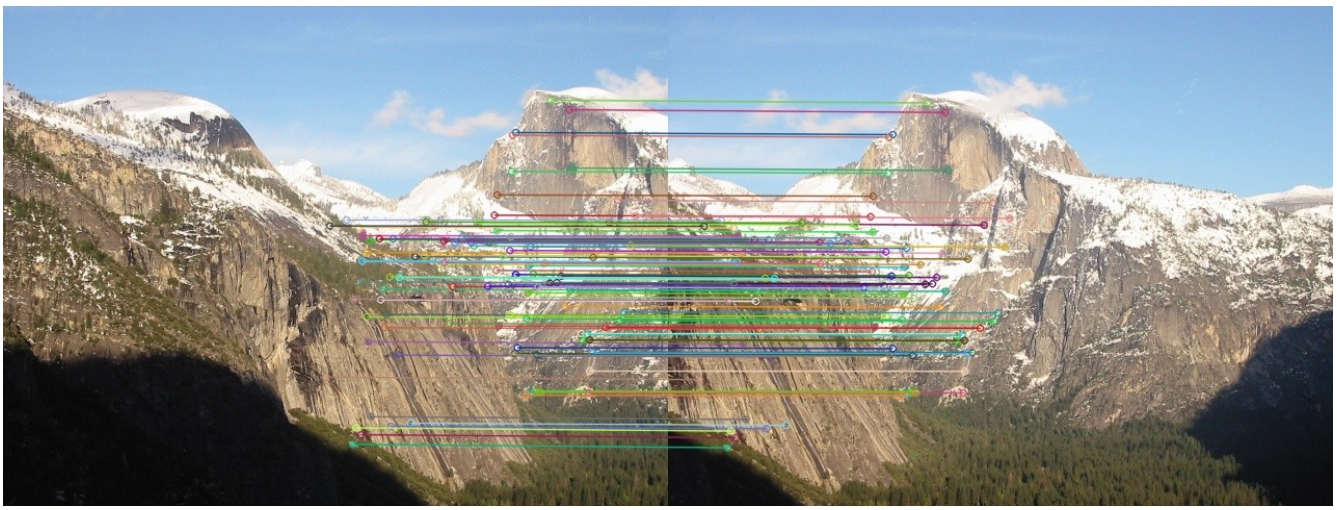

Figure 12: Resultados correspondencias Yosemite utilizando Lowe Average 2NN

Como podemos ver, el segundo criterio utilizado obtiene mejores resultados. En el primer caso, podemos observar que se han establecido correspondencias que no son correctas, para estas imágenes, aquellas líneas que no son horizontales representan correspondencias erróneas. Esto no ocurre para el segundo caso, en el que se han eliminado todas las líneas no horizontales ya que han sido descartadas al no cumplir la condición del cociente de las distancias.

Ejercicio 3

En este apartado nos piden generar un mosaico a partir de tres imágenes. Comenzamos generando una imagen lo suficientemente grande en la que vamos a pintar el mosaico. Luego, calculamos las distintas homografías que nos llevan las imágenes al mosaico. Las homografías se estiman utilizando las correspondencias obtenidas utilizando el segundo criterio del ejercicio anterior.

Comenzamos por la imagen central de las tres y la trasladamos al centro del canvas, esta traslación será la homografía H_0. Después, obtenemos las homografías que llevan la imagen de la izquierda a la central, H_1, y la imagen de la derecha a la central, H_2. Por último, multiplicamos las matrices que definen las homografías para llevar las imágenes de los extremos al canvas, H_0H_1 para la imagen de la izquierda y H_0H_2 para la imagen de la derecha. Y utilizamos las matrices resultantes para llevar cada imagen al mosaico.

Una vez tenemos el mosaico generado, eliminamos toda la parte del canvas que no se ha utilizado.

Figure 13: Imágenes utilizadas para generar el mosaico

El mosaico que obtenemos al utilizar estas imágenes es de gran calidad. Si eliminásemos los bordes negros al completo sería difícil decir que ha sido generada a partir de otras 3 imágenes. Todas las imágenes encajan perfectamente y no se produce ningún cambio de color brusco.

Figure 14: Mosaico obtenido a partir de las imágenes de la figura anterior

Ejercicio 4

Nos piden ahora resolver el mismo problema que en el ejercicio anterior pero en el caso de utilizar N imágenes para generar el mosaico. Usamos la misma metodología que antes, empezamos desde la imagen central y vamos llevando las imágenes de los lados al canvas hasta llegar a las imágenes de los extremos.

Ya sabemos como estimar las homografías para las imágenes adyacentes a la central. Para el resto de imágenes simplemente repetimos el proceso encadenando multiplicaciones de matrices de forma que para la imagen *i*-ésima a la izquierda del centro primero tenemos que calcular las homografías de las imágenes entre la *i*-ésima y el centro. Entonces, para estimar la homografía que lleva la imagen en la posición *i*-ésima a la izquierda del centro hacemos:

$$H_i = H_0 H_1 H_2 ... H_{i-1}$$

Siendo H_0 la homografía que traslada la imagen al centro. H_1, la homografía que lleva la primera imagen de la izquierda a la central. H_2, la homografía que lleva la segunda imagen de la izquierda a la primera, etc. Lo mismo ocurre para las imágenes a la derecha del centro.

A la hora de implementar la función que genera el mosaico, las imágenes se encuentran ordenadas en una lista de izquierda a derecha, por tanto, la imagen central se encuentra en la posición N/2 de la lista siendo N el tamaño de esta. Para saber que imagen hay que tomar para estimar las homografías hay que tener en cuenta su posición, para una imagen a la izquierda, *i*, se utiliza la imagen *i*+1 mientras que para una imagen a la derecha, *j*, se usa la imagen *j*-1.

Solamente se ha implementado una función que resuelve tanto el problema del ejercicio 3 como este problema más genérico.

Figure 15: Imágenes utilizadas para generar el mosaico

Figure 16: Mosaico obtenido a partir de las imágenes de la figura anterior

Aunque obtenemos un buen mosaico, se puede apreciar a la derecha un cambio de color en los árboles lo cual no depende de las homografías utilizadas.

Referencias

[1] Multi-Image Matching using Multi-Scale Oriented Patches

[2] Tutorial feature matching Opencv