

Práctica 2: Detección de puntos relevantes y construcción de panoramas

Ignacio Aguilera Martos

Visión por Computador

Ejercicio 1

Detección de puntos SIFT y SURF. Aplicar la detección de puntos SIFT y SURF sobre las imágenes, representar dichos puntos sobre las imágenes haciendo uso de la función drawKeypoints. Presentar los resultados con las imágenes Yosemite.rar.

Apartado A

Enunciado: Variar los valores de umbral de la función de detección de puntos hasta obtener un conjunto numeroro (≥ 1000) de puntos SIFT y SURF que sea representativo de la imagen. Justificar la elección de los parámetros en relación a la representatividad de los puntos obtenidos.

Solución: Para la elección de los puntos SIFT he usado los valores nfeatures=0, nOctaveLayers=3, contrastThreshold=0.06, edgeThreshold=6 y sigma=1.6.

Los valores que he empleado para SURF y Yosemite1 los valores nfeatures, nOctaveLayers y sigma no los he variado de los que venían por defecto. En cambio contrastThreshold y edgeThreshold los he modificado para obtener menos puntos dentro de la imagen. Al iniciar el algoritmo con los valores que venían por defecto he podido observar que el número de puntos que obtenían los algoritmos era muy elevado pero muy concentrados en ciertas zonas de la imagen y con puntos en zonas negras o el cielo. He modificado contrastThreshold aumentándolo puesto que cuanto mayor sea este número menos elementos filtramos y he modificado edgeThreshold para que obtuviera menos puntos en los bordes de las figuras, puesto que estaban muy saturados.

Los puntos que he obtenido han sido:

En la detección de los puntos SURF he modificado únicamente el parámetro hessianThreshold. Los puntos que SURF obtiene tienen que tener una hessiana mayor que este valor, por lo que si lo vamos aumentando se obtienen menos puntos clave en la imagen. El valor que he tomado para este parámetro ha sido

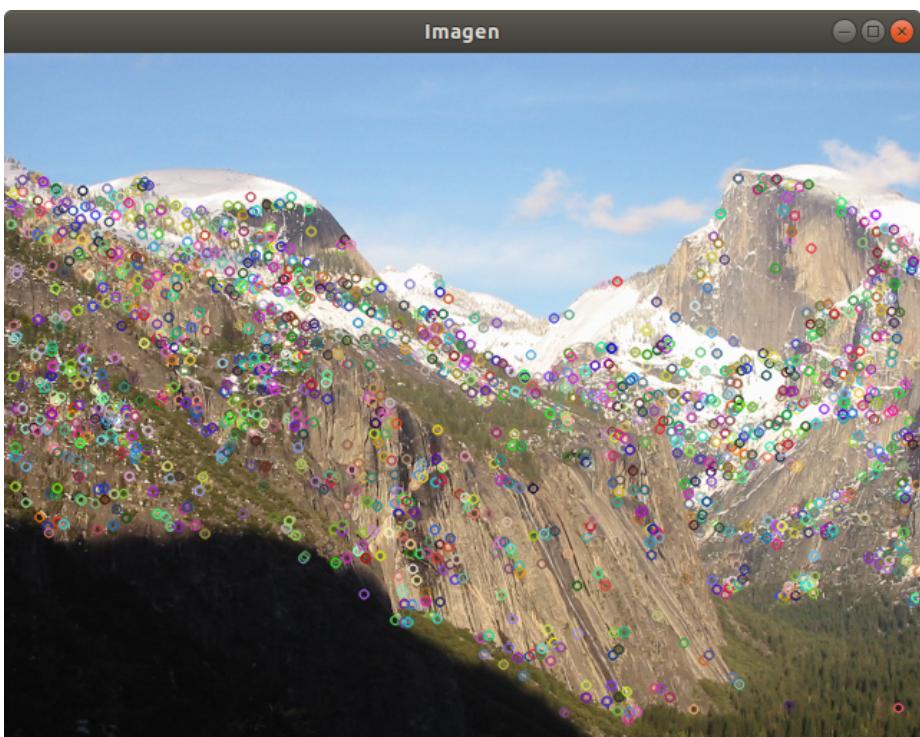


Figure 1: SIFT yosemite1

400. Al igual que con SIFT, obtuve muchos puntos en la imagen que estaban colocados en zonas negras y el cielo, fui aumentando el valor de umbral hasta que conseguí que estos puntos no estuvieran y hubiese un conjunto de al menos 1000 puntos.

Los puntos que he obtenido han sido:

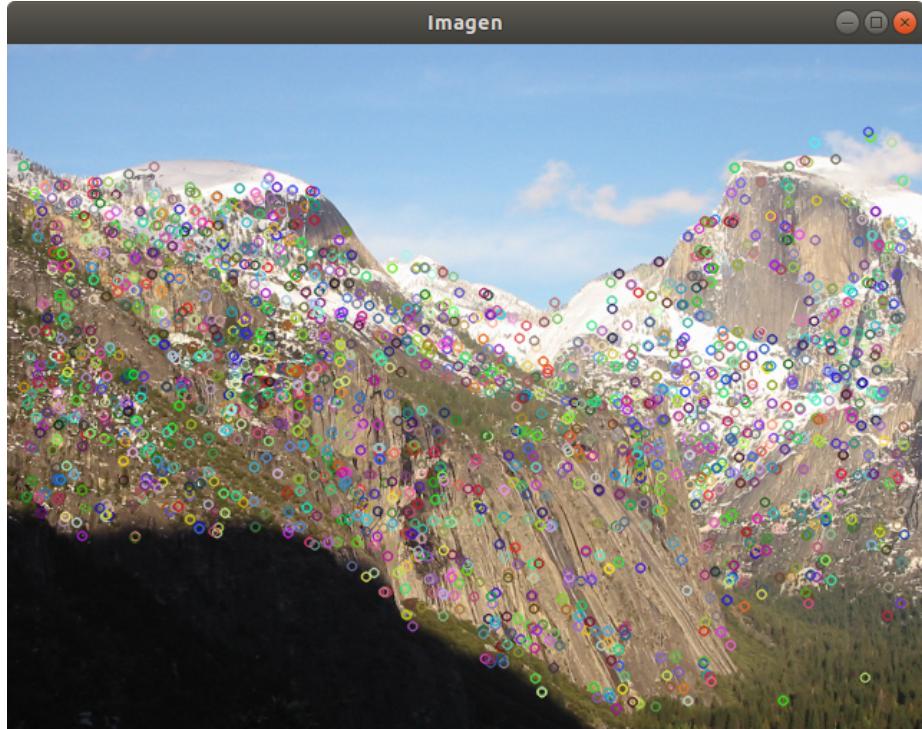


Figure 2: SURF yosemite1

Para la imagen Yosemite2 he utilizado en SIFT los valores contrastThreshold a 0.06 y edgeThreshold a 4, puesto que detectaba muchos puntos en los bordes de las nubes.

Los resultados obtenidos para SIFT han sido:

Para la imagen Yosemite2 he utilizado en SURF un valor de hessianThreshold de 500, ya que al igual que con SIFT se obtenían muchos puntos en las nubes que posteriormente no iban a ser relevantes por la gran concentración de los mismos.

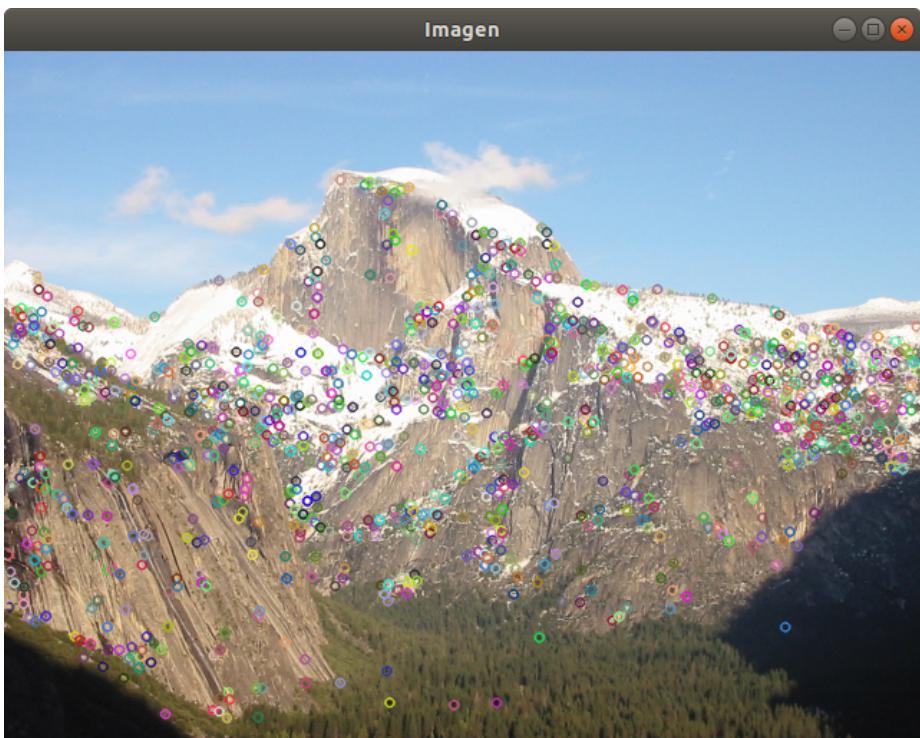


Figure 3: SIFT yosemite2

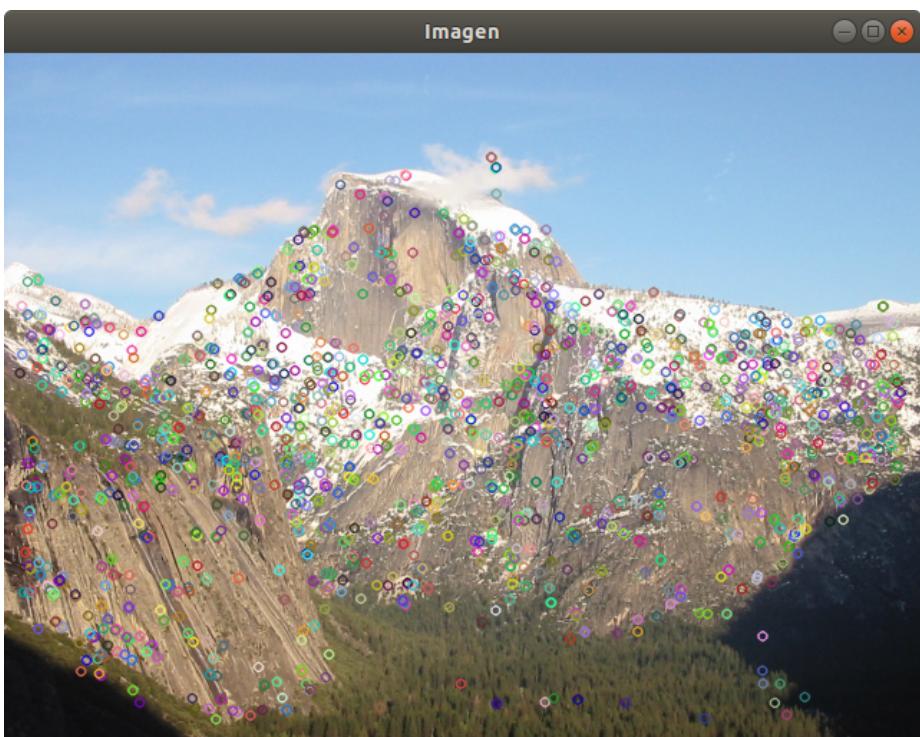


Figure 4: SURF yosemite2

Apartado B

Enunciado: Identificar cuántos puntos se han detectado dentro de cada octava. En el caso de SIFT, identificar también los puntos detectados en cada capa. Mostrar el resultado dibujando sobre la imagen original un círculo centrado en cada punto y de radio proporcional al valor de sigma usado para su detección (ver circle()) y pintar cada octava en un color.

Solución:

Lo primero que hay que hacer para poder obtener las octavas y capas es extraer la información de los KeyPoints. Los KeyPoints, por eficiencia, tienen gran parte de la información (incluidas la escala y octava) en un sólo número entero. Para poder obtener esta información es necesario, para la octava, aplicar una máscara al número, en concreto la que viene dada por 0xFF o 255. Para poder obtener la información de la capa tenemos que hacer un shift binario de desplazamiento 8.

Una vez que tenemos la información anterior extraída es sencillo poder contar los puntos en cada octava y capa (sólo para SIFT). Como hemos visto en teoría, se aplica un subsampling a la imagen y un blur reduciendo la resolución a la mitad cada vez que se hace esta operación, cuyo resultado es conocido como octava. En cada octava se realiza un algoritmo de detección, por lo que podemos saber de qué octava proviene cada punto. Además en el algoritmo SIFT se le aplica un suavizado Gaussiano por capas (en cada capa el suavizado es mayor que en la anterior) para cada octava, de forma que no sólo podemos obtener de qué octava proviene un punto de interés si no que también podemos saber de qué capa proviene.

```
Imagen Yosemite1
El número de puntos por octava en SIFT ha sido: {'0': 1568, '1': 75, '2': 29, '3': 3, '4': 3, '5': 1}
El número de puntos por octava en SURF ha sido: {'0': 1331, '2': 76, '1': 293, '3': 11}
El número de puntos por capa en SIFT ha sido: {'2': 513, '3': 339, '1': 827}
Imagen Yosemite2
El número de puntos por octava en SIFT ha sido: {'0': 1193, '1': 51, '2': 14, '3': 4, '4': 3}
El número de puntos por octava en SURF ha sido: {'0': 1001, '1': 227, '2': 54, '3': 11}
El número de puntos por capa en SIFT ha sido: {'3': 258, '1': 660, '2': 347}
```

Figure 5: Número de puntos

Como se puede observar la información obtenida se representa en un diccionario de Python, en el que el identificador es la octava o capa correspondiente y el valor es el número de puntos obtenidos en dicha capa u octava.

Como podemos observar la detección de puntos de interés es más efectiva en la primera octava de SIFT y SURF y en las primeras capas de SIFT. Esto es razonable, puesto que cuanto más suavizado y subsampling apliquemos más

restringimos los valores de los píxeles destacados. Aún así la detección en octavas y capas mayores a la primera sigue siendo útil.

Para poder visualizar mejor los puntos de interés obtenidos tal y como se dice en el enunciado, puede ser de interés pintar los puntos por colores en función de sus octavas y pintar los puntos como círculos cuyo radio sea proporcional al sigma usado para hallarlo, de forma que podemos ver el nivel de suavizado que se ha requerido para llegar al mismo. Veamos las imágenes para SIFT:

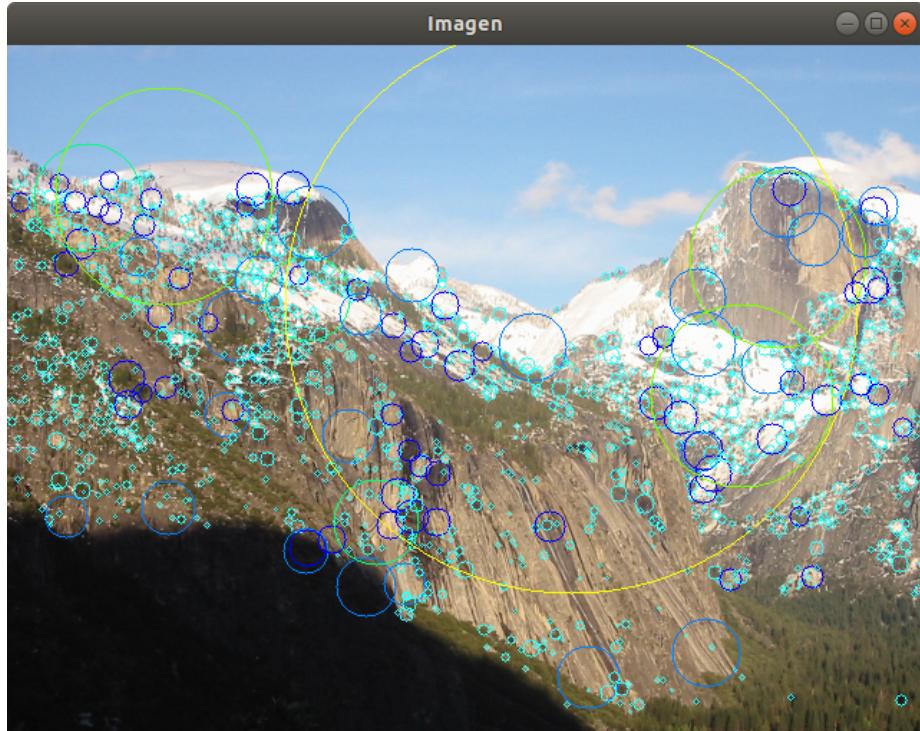


Figure 6: Círculos SIFT

Como podemos observar hay un punto en concreto que destaca por el gran radio de su círculo, esto es debido a que pertenece a una de las capas más profundas de SIFT, de forma que se ha aplicado un suavizado muy grande hasta llegar a él. Cabe destacar que los colores empleados para pintar las octavas han sido (según el orden): amarillo, rojo, naranja, verde, verde azulado y azul claro. De esta forma los puntos de la primera octava se pintan en amarillo y los de la última en azul claro.

Se puede observar en el caso de SURF que se comparten puntos con SIFT en los que se ha aplicado un suavizado muy grande hasta detectarlos como puntos de interés.

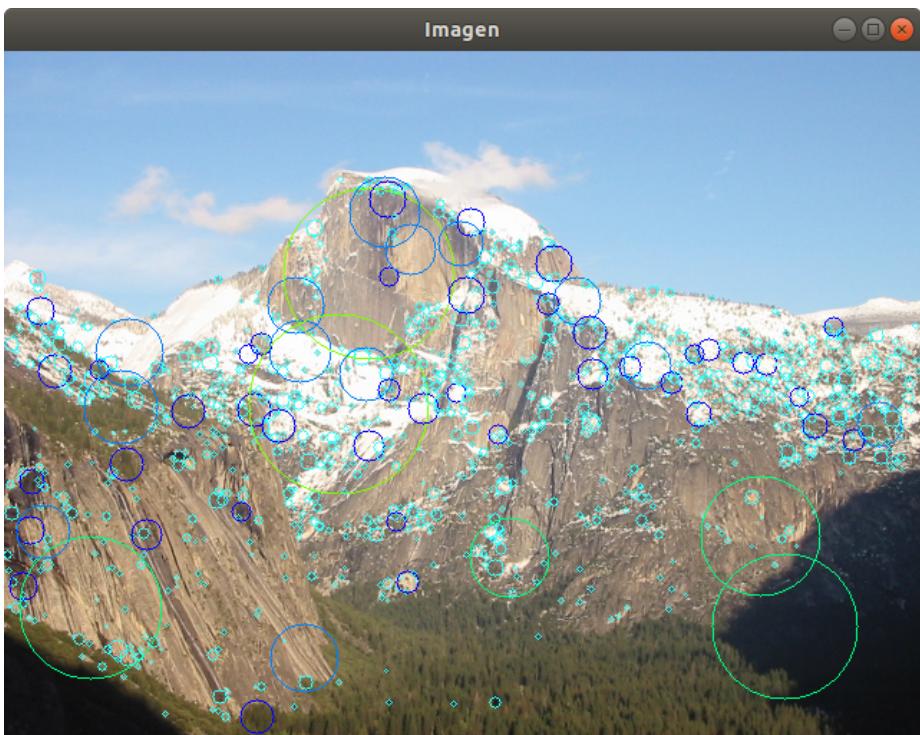


Figure 7: Círculos SIFT

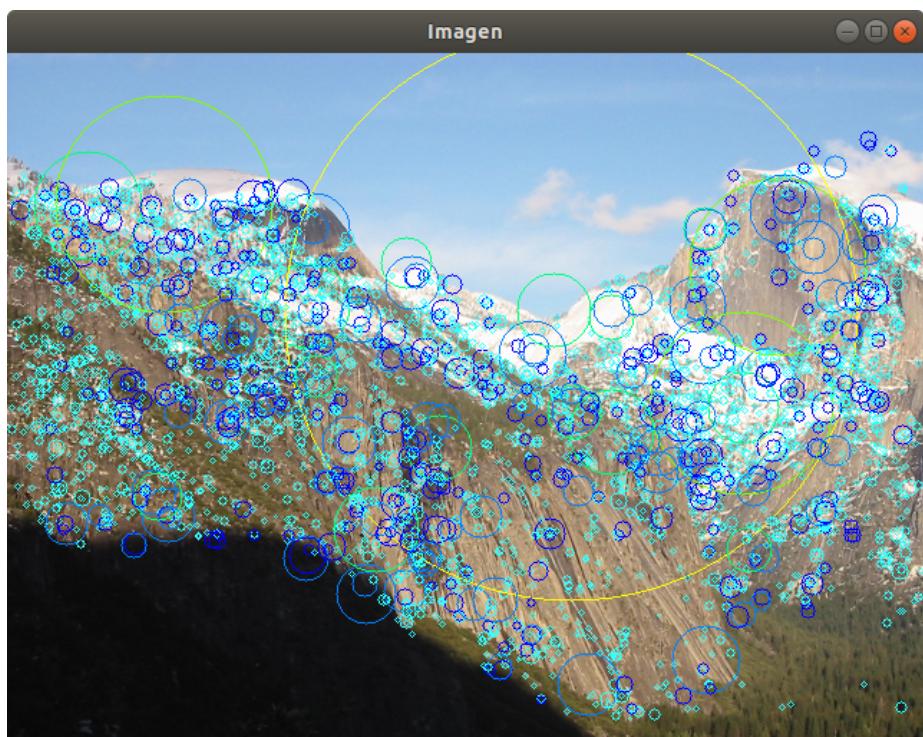


Figure 8: Círculos SURF

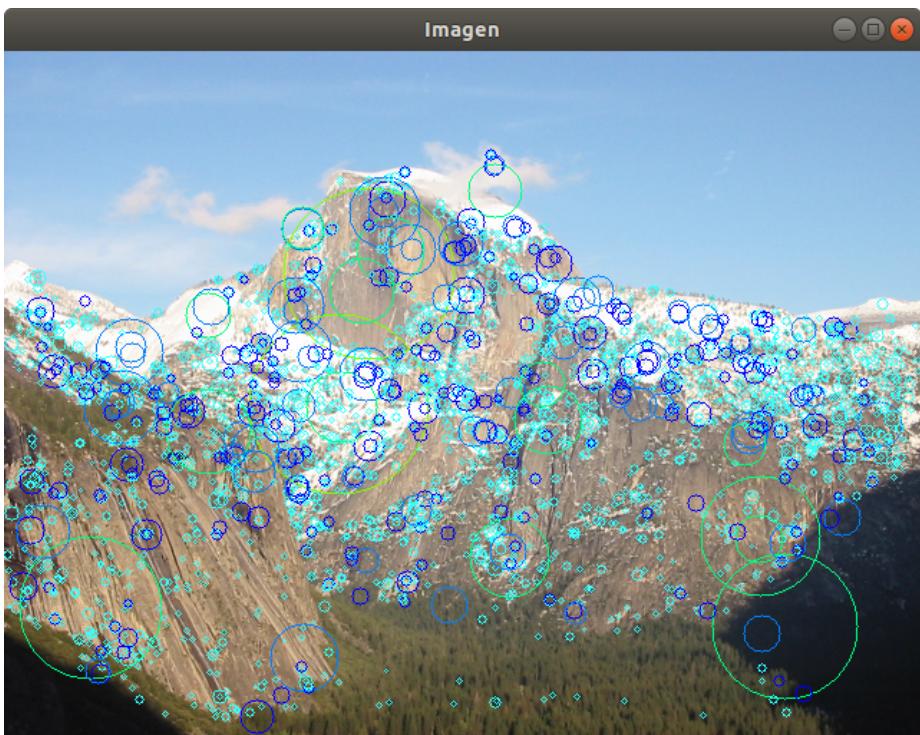


Figure 9: Círculos SURF

Apartado C

Enunciado: Mostrar cómo con el vector de keyPoint extraídos se pueden calcular los descriptores SIFT y SURF asociados a cada punto usando OpenCV.

Solución: Para poder calcular los descriptores utilizando ya los keyPoints calculados en los apartados anteriores OpenCV tiene la función ‘compute’.

La función compute es válida tanto para objetos SIFT como SURF, de forma que dados los keyPoints y la imagen se pueden obtener los descriptores asignados a dichos puntos de interés.

Aquí podemos ver la salida de los keyPoints para SIFT y SURF sobre Yosemite1 y Yosemite2:

```
Descriptores SIFT: ([<KeyPoint 0x7f058e056a20>, <KeyPoint 0x7f058e056a50>, <KeyPo
int 0x7f058e056a80>, <KeyPoint 0x7f058e056ab0>, <KeyPoint 0x7f058e056ae0>, <KeyPo
int 0x7f058e056b10>, <KeyPoint 0x7f058e056b40>, <KeyPoint 0x7f058e056b70>, <KeyPo
int 0x7f058e056ba0>, <KeyPoint 0x7f058e056bd0>, <KeyPoint 0x7f058e056c00>, <KeyPo
int 0x7f058e056c30>, <KeyPoint 0x7f058e056c60>, <KeyPoint 0x7f058e056c90>, <KeyPo
int 0x7f058e056cc0>, <KeyPoint 0x7f058e056cf0>, <KeyPoint 0x7f058e056d20>, <KeyPo
int 0x7f058e056d50>, <KeyPoint 0x7f058e056d80>, <KeyPoint 0x7f058e056db0>, <KeyPo
int 0x7f058e056de0>, <KeyPoint 0x7f058e056e10>, <KeyPoint 0x7f058e056e40>, <KeyPo
int 0x7f058e056e70>, <KeyPoint 0x7f058e056ea0>, <KeyPoint 0x7f058e056ed0>, <KeyPo
int 0x7f058e056f00>, <KeyPoint 0x7f058e056f30>, <KeyPoint 0x7f058e056f60>, <KeyPo
int 0x7f058e056f90>, <KeyPoint 0x7f058e056fc0>, <KeyPoint 0x7f058e07f600>, <KeyPo
int 0x7f058e07f630>, <KeyPoint 0x7f058e07f660>, <KeyPoint 0x7f058e07f690>, <KeyPo
int 0x7f058e07f6c0>, <KeyPoint 0x7f058e07f6f0>, <KeyPoint 0x7f058e07f720>, <KeyPo
int 0x7f058e07f750>, <KeyPoint 0x7f058e07f780>, <KeyPoint 0x7f058e07f7b0>, <KeyPo
int 0x7f058e07f7e0>, <KeyPoint 0x7f058e07f810>, <KeyPoint 0x7f058e07f840>, <KeyPo
int 0x7f058e07f870>, <KeyPoint 0x7f058e07f8a0>, <KeyPoint 0x7f058e07f8d0>, <KeyPo
int 0x7f058e07f900>, <KeyPoint 0x7f058e07f930>, <KeyPoint 0x7f058e07f960>, <KeyPo
int 0x7f058e07f990>, <KeyPoint 0x7f058e07f9c0>, <KeyPoint 0x7f058e07f9f0>, <KeyPo
int 0x7f058e07fa20>, <KeyPoint 0x7f058e07fa50>, <KeyPoint 0x7f058e07fa80>, <KeyPo
int 0x7f058e07fab0>, <KeyPoint 0x7f058e07fae0>, <KeyPoint 0x7f058e07fb10>, <KeyPo
int 0x7f058e07fb40>, <KeyPoint 0x7f058e07fb70>, <KeyPoint 0x7f058e07fba0>, <KeyPo
int 0x7f058e07fdb0>, <KeyPoint 0x7f058e07fc00>, <KeyPoint 0x7f058e07fc30>, <KeyPo
```

Figure 10: Descriptores SIFT Yosemite1

OpenCV también ofrece un método que calcula los descriptores y los puntos de interés en una sola orden: detectAndCompute. Esta orden funciona igual que la detección de puntos de interés, con la única diferencia de que devuelve dos objetos: los puntos de interés y los descriptores.

Ejercicio 2

Usar el detector-descriptor SIFT de OpenCV sobre las imágenes de Yosemite.rar (cv2.xfeatures2d.SIFT_create()). Extraer sus listas de keyPoints y descriptores asociados. Establecer las correspondencias existentes entre ellos usando el objeto BFMatcher de OpenCV y los criterios de correspondencias “Brute-Force+crossCheck” y “Lowe-Average-2NN”. (NOTA: Si se usan los resultados

```
nacheteam@nacheteam-Equipo: ~/MEGA/Doble Grado DGIIM/5to Curso/Primer Cuatrimestre... ● ○ ×
Archivo Editar Ver Buscar Terminal Ayuda
int 0x7f058e069a20>, <KeyPoint 0x7f058e069a50>, <KeyPoint 0x7f058e069a80>, <KeyPo
int 0x7f058e069ab0>, <KeyPoint 0x7f058e069ae0>, <KeyPoint 0x7f058e069b10>, <KeyPo
int 0x7f058e069b40>, <KeyPoint 0x7f058e069b70>, <KeyPoint 0x7f058e069ba0>, <KeyPo
int 0x7f058e069bd0>, <KeyPoint 0x7f058e069c00>, <KeyPoint 0x7f058e069c30>, <KeyPo
int 0x7f058e069c60>, <KeyPoint 0x7f058e069c90>, <KeyPoint 0x7f058e069cc0>, <KeyPo
int 0x7f058e069cf0>, <KeyPoint 0x7f058e069d20>, <KeyPoint 0x7f058e069d50>, <KeyPo
int 0x7f058e069d80>, <KeyPoint 0x7f058e069db0>, <KeyPoint 0x7f058e069de0>, <KeyPo
int 0x7f058e069e10>, <KeyPoint 0x7f058e069e40>, <KeyPoint 0x7f058e069e70>, <KeyPo
int 0x7f058e069ea0>, <KeyPoint 0x7f058e069ed0>, <KeyPoint 0x7f058e069f00>, <KeyPo
int 0x7f058e069f30>, <KeyPoint 0x7f058e069f60>, <KeyPoint 0x7f058e069f90>], array
([[ 41., 112., 74., ..., 1., 0., 0.],
 [ 7., 47., 42., ..., 0., 0., 0.],
 [ 0., 0., 0., ..., 7., 6., 60.],
 ...,
 [ 48., 69., 17., ..., 0., 0., 0.],
 [ 76., 36., 7., ..., 6., 2., 0.],
 [ 17., 6., 25., ..., 0., 0., 0.]], dtype=float32))
Descriptoros SURF: ([<KeyPoint 0x7f058e069fc0>, <KeyPoint 0x7f058e06a030>, <KeyPo
int 0x7f058e06a060>, <KeyPoint 0x7f058e06a090>, <KeyPoint 0x7f058e06a0c0>, <KeyPo
int 0x7f058e06a0f0>, <KeyPoint 0x7f058e06a120>, <KeyPoint 0x7f058e06a150>, <KeyPo
int 0x7f058e06a180>, <KeyPoint 0x7f058e06a1b0>, <KeyPoint 0x7f058e06a1e0>, <KeyPo
int 0x7f058e06a210>, <KeyPoint 0x7f058e06a240>, <KeyPoint 0x7f058e06a270>, <KeyPo
int 0x7f058e06a2a0>, <KeyPoint 0x7f058e06a2d0>, <KeyPoint 0x7f058e06a300>, <KeyPo
int 0x7f058e06a330>, <KeyPoint 0x7f058e06a360>, <KeyPoint 0x7f058e06a390>, <KeyPo
```

Figure 11: Descriptoros SIFT y SURF Yosemite1

```
nacheteam@nacheteam-Equipo: ~/MEGA/Doble Grado DGIIM/5to Curso/Primer Cuatrimestre... ● ○ ×
Archivo Editar Ver Buscar Terminal Ayuda
int 0x7f058dfe20f0>, <KeyPoint 0x7f058dfe2120>, <KeyPoint 0x7f058dfe2150>, <KeyPo
int 0x7f058dfe2180>, <KeyPoint 0x7f058dfe21b0>, <KeyPoint 0x7f058dfe21e0>, <KeyPo
int 0x7f058dfe2210>, <KeyPoint 0x7f058dfe2240>, <KeyPoint 0x7f058dfe2270>, <KeyPo
int 0x7f058dfe22a0>, <KeyPoint 0x7f058dfe22d0>, <KeyPoint 0x7f058dfe2300>, <KeyPo
int 0x7f058dfe2330>, <KeyPoint 0x7f058dfe2360>, <KeyPoint 0x7f058dfe2390>, <KeyPo
int 0x7f058dfe23c0>, <KeyPoint 0x7f058dfe23f0>, <KeyPoint 0x7f058dfe2420>, <KeyPo
int 0x7f058dfe2450>, <KeyPoint 0x7f058dfe2480>, <KeyPoint 0x7f058dfe24b0>, <KeyPo
int 0x7f058dfe24e0>, <KeyPoint 0x7f058dfe2510>, <KeyPoint 0x7f058dfe2540>, <KeyPo
int 0x7f058dfe2570>, <KeyPoint 0x7f058dfe25a0>], array([[ 0.00021616, -0.00027961
, 0.0009371 , ..., 0.00122399,
 0.00176429, 0.004422998],
 [ 0.00296671, -0.00143716, 0.00521279, ..., -0.00346219,
 0.00272918, 0.00530223],
 [-0.00692849, 0.00245703, 0.00791069, ..., -0.00054284,
 0.00444722, 0.00065153],
 ...,
 [-0.00837958, -0.00385381, 0.00868957, ..., -0.00031707,
 0.00503991, 0.00435023],
 [ 0.0019105 , 0.00255177, 0.00978464, ..., -0.00526859,
 0.00873021, 0.00668345],
 [-0.00822145, 0.00339035, 0.01074585, ..., 0.00051001,
 0.00574227, 0.00660396]], dtype=float32))
(cv) nacheteam@nacheteam-Equipo:~/MEGA/Doble Grado DGIIM/5to Curso/Primer Cuatrimestre/Vision por Computador/Practicas/ComputerVision/Practica 2$ █
```

Figure 12: Descriptoros SURF Yosemite1

```
nacheteam@nacheteam-Equipo: ~/MEGA/Doble Grado DGIIM/Sto Curso/Primer Cuatrimest... ● ○ ×
Archivo Editar Ver Buscar Terminal Ayuda
Imagen Yosemite2
Descriptores SIFT: ([<KeyPoint 0x7f05ba882150>, <KeyPoint 0x7f05ba882180>, <KeyP
oint 0x7f05ba8821b0>, <KeyPoint 0x7f05ba8821e0>, <KeyPoint 0x7f05ba882210>, <Key
Point 0x7f05ba882240>, <KeyPoint 0x7f05ba882270>, <KeyPoint 0x7f05ba8822a0>, <Ke
yPoint 0x7f05ba8822d0>, <KeyPoint 0x7f05ba882300>, <KeyPoint 0x7f05ba882330>, <K
eyPoint 0x7f05ba882360>, <KeyPoint 0x7f05ba882390>, <KeyPoint 0x7f05ba8823c0>, <
KeyPoint 0x7f05ba8823f0>, <KeyPoint 0x7f05ba882420>, <KeyPoint 0x7f05ba882450>,
<KeyPoint 0x7f05ba882480>, <KeyPoint 0x7f05ba8824b0>, <KeyPoint 0x7f05ba8824e0>,
<KeyPoint 0x7f05ba882510>, <KeyPoint 0x7f05ba882540>, <KeyPoint 0x7f05ba882570>,
<KeyPoint 0x7f05ba8825a0>, <KeyPoint 0x7f05ba8825d0>, <KeyPoint 0x7f05ba882600>,
<KeyPoint 0x7f05ba882630>, <KeyPoint 0x7f05ba882660>, <KeyPoint 0x7f05ba88269
0>, <KeyPoint 0x7f05ba8826c0>, <KeyPoint 0x7f05ba8826f0>, <KeyPoint 0x7f05ba8827
20>, <KeyPoint 0x7f05ba882750>, <KeyPoint 0x7f05ba882780>, <KeyPoint 0x7f05ba88
7b0>, <KeyPoint 0x7f05ba8827e0>, <KeyPoint 0x7f05ba882810>, <KeyPoint 0x7f05ba88
2840>, <KeyPoint 0x7f05ba882870>, <KeyPoint 0x7f05ba8828a0>, <KeyPoint 0x7f05ba8
828d0>, <KeyPoint 0x7f05ba882900>, <KeyPoint 0x7f05ba882930>, <KeyPoint 0x7f05ba
882960>, <KeyPoint 0x7f05ba882990>, <KeyPoint 0x7f05ba8829c0>, <KeyPoint 0x7f05b
a8829f0>, <KeyPoint 0x7f05ba882a20>, <KeyPoint 0x7f05ba882a50>, <KeyPoint 0x7f05
ba882a80>, <KeyPoint 0x7f05ba882ab0>, <KeyPoint 0x7f05ba882ae0>, <KeyPoint 0x7f0
5ba882b10>, <KeyPoint 0x7f05ba882b40>, <KeyPoint 0x7f05ba882b70>, <KeyPoint 0x7f
05ba882ba0>, <KeyPoint 0x7f05ba882bd0>, <KeyPoint 0x7f05ba882c00>, <KeyPoint 0x7
f05ba882c30>, <KeyPoint 0x7f05ba882c60>, <KeyPoint 0x7f05ba882c90>, <KeyPoint 0x
7f05ba882cc0>, <KeyPoint 0x7f05ba882cf0>, <KeyPoint 0x7f05ba882d20>, <KeyPoint 0
x7f05ba882d50>, <KeyPoint 0x7f05ba882d80>, <KeyPoint 0x7f05ba882db0>, <KeyPoint
```

Figure 13: Descriptores SIFT Yosemite2

```
nacheteam@nacheteam-Equipo: ~/MEGA/Doble Grado DGIIM/Sto Curso/Primer Cuatrimest... ● ○ ×
Archivo Editar Ver Buscar Terminal Ayuda
[<KeyPoint 0x7f05ba7a0810>, <KeyPoint 0x7f05ba7a0840>, <KeyPoint 0x7f05ba7a087
0>, <KeyPoint 0x7f05ba7a08a0>, <KeyPoint 0x7f05ba7a08d0>, <KeyPoint 0x7f05ba7a09
00>, <KeyPoint 0x7f05ba7a0930>, <KeyPoint 0x7f05ba7a0960>, <KeyPoint 0x7f05ba7a0
990>, <KeyPoint 0x7f05ba7a09c0>, <KeyPoint 0x7f05ba7a09f0>, <KeyPoint 0x7f05ba7a
0a20>, <KeyPoint 0x7f05ba7a0a50>, <KeyPoint 0x7f05ba7a0a80>, <KeyPoint 0x7f05ba7
a0ab0>, <KeyPoint 0x7f05ba7a0ae0>, <KeyPoint 0x7f05ba7a0b10>, <KeyPoint 0x7f05ba
7a0b40>, <KeyPoint 0x7f05ba7a0b70>, <KeyPoint 0x7f05ba7a0ba0>, <KeyPoint 0x7f05b
a7a0bd0>, <KeyPoint 0x7f05ba7a0c00>, <KeyPoint 0x7f05ba7a0c30>, <KeyPoint 0x7f05
ba7a0c60>, <KeyPoint 0x7f05ba7a0c90>, <KeyPoint 0x7f05ba7a0cc0>, <KeyPoint 0x7f0
5ba7a0acf0>, <KeyPoint 0x7f05ba7a0d20>, <KeyPoint 0x7f05ba7a0d50>, <KeyPoint 0x7
f05ba7a0d80>], array([[ 6.,   5.,  17., ...,  0.,   0.,   0.],
   [ 0.,   0.,   0., ...,  5.,  18.,  75.],
   [ 26.,   3.,   0., ...,  1.,   2.,   2.],
   ...,
   [ 0.,   2.,   5., ..., 23.,  37., 114.],
   [ 0.,   0.,   0., ...,  1.,   1.,   15.],
   [ 0.,   0.,   0., ...,  4.,   2.,   8.]], dtype=float32))
Descriptores SURF: ([<KeyPoint 0x7f05ba7a0db0>, <KeyPoint 0x7f05ba7a0de0>, <KeyP
oint 0x7f05ba7a0e10>, <KeyPoint 0x7f05ba7a0e40>, <KeyPoint 0x7f05ba7a0e70>, <Key
Point 0x7f05ba7a0ea0>, <KeyPoint 0x7f05ba7a0ed0>, <KeyPoint 0x7f05ba7a0f00>, <Ke
yPoint 0x7f05ba7a0f30>, <KeyPoint 0x7f05ba7a0f60>, <KeyPoint 0x7f05ba7a0f90>, <K
eyPoint 0x7f05ba7a0fc0>, <KeyPoint 0x7f05ba7a1030>, <KeyPoint 0x7f05ba7a1060>, <
KeyPoint 0x7f05ba7a1090>, <KeyPoint 0x7f05ba7a10c0>, <KeyPoint 0x7f05ba7a10f0>,
<KeyPoint 0x7f05ba7a1120>, <KeyPoint 0x7f05ba7a1150>, <KeyPoint 0x7f05ba7a1180>
```

Figure 14: Descriptores SIFT y SURF Yosemite2

```
nacheteam@nacheteam-Equipo: ~/MEGA/Doble Grado DGIIM/Sto Curso/Primer Cuatrimestre... ● ◻ ✕
Archivo Editar Ver Buscar Terminal Ayuda
0>, <KeyPoint 0x7f05ba7b25a0>, <KeyPoint 0x7f05ba7b25d0>, <KeyPoint 0x7f05ba7b26
00>, <KeyPoint 0x7f05ba7b2630>, <KeyPoint 0x7f05ba7b2660>, <KeyPoint 0x7f05ba7b2
690>, <KeyPoint 0x7f05ba7b26c0>, <KeyPoint 0x7f05ba7b26f0>, <KeyPoint 0x7f05ba7b
2720>, <KeyPoint 0x7f05ba7b2750>, <KeyPoint 0x7f05ba7b2780>, <KeyPoint 0x7f05ba7
b27b0>, <KeyPoint 0x7f05ba7b27e0>, <KeyPoint 0x7f05ba7b2810>, <KeyPoint 0x7f05ba
7b2840>, <KeyPoint 0x7f05ba7b2870>, <KeyPoint 0x7f05ba7b28a0>, <KeyPoint 0x7f05b
a7b28d0>, <KeyPoint 0x7f05ba7b2900>, <KeyPoint 0x7f05ba7b2930>, <KeyPoint 0x7f05
ba7b2960>, <KeyPoint 0x7f05ba7b2990>, <KeyPoint 0x7f05ba7b29c0>, <KeyPoint 0x7f05
ba7b29f0>, <KeyPoint 0x7f05ba7b2a20>, <KeyPoint 0x7f05ba7b2a50>, <KeyPoint 0x7f
05ba7b2a80>, <KeyPoint 0x7f05ba7b2ab0>, <KeyPoint 0x7f05ba7b2ae0>, <KeyPoint 0x7
f05ba7b2b10>, <KeyPoint 0x7f05ba7b2b40>, <KeyPoint 0x7f05ba7b2b70>], array([[ 1.
1199357e-02, 1.11434632e-03, 1.26695223e-02, ...,
8.78521940e-04, 3.49142030e-03, 2.15772795e-03],
[-2.08570994e-03, 2.31291819e-03, 5.08099468e-03, ...,
6.43282663e-04, 3.10497894e-03, 6.75790617e-03],
[-9.69605811e-04, 1.07174375e-04, 1.34203711e-03, ...,
4.91164392e-04, 1.44300249e-03, 2.06682342e-03],
...,
[ 1.83863426e-03, 9.92628839e-03, 1.08790323e-02, ...,
5.23267966e-03, 5.72338188e-03, 7.01669883e-03],
[ 1.83270348e-03, 4.76835034e-04, 1.05972579e-02, ...,
-1.25534916e-05, 1.28395446e-02, 2.00430979e-03],
[-1.43512932e-03, 6.44768355e-03, 3.85684893e-03, ...,
-3.02879082e-04, 1.38120744e-02, 6.09390344e-03]], dtype=float32))
```

Figure 15: Descriptores SURF Yosemite2

propios del punto anterior en lugar del cálculo de SIFT de OpenCV se añaden 0.5 puntos).

Apartado A

Enunciado: Mostrar ambas imágenes en un mismo canvas y pintar líneas de diferentes colores entre las coordenadas de los puntos en correspondencias. Mostrar en cada caso 100 elegidas aleatoriamente.

Solución: Para la detección de correspondencias con fuerza bruta y cross check tenemos que crear, tal y como dice el enunciado un objeto BFMatcher. En nuestro caso para el CrossCheck tenemos que ponerle como parámetros la norma que debe emplear, usando la norma L2 (también conocida como euclídea) que es la que se emplea por defecto además de un booleano llamado crossCheck que debemos colocar a True. Tras esto se comprueban las correspondencias mediante la función match de OpenCV y tomamos una muestra de 100 elementos sin reemplazamiento, es decir, sin repetir elementos. Tras esto sólo tenemos que llamar a la función drawMatches con la muestra que hemos obtenido y obtenemos la imagen de las correspondencias.

A continuación vemos el resultado:

Como podemos observar las correspondencias de los puntos que no comparten las imágenes es bastante pobre, haciendo que haya muchas líneas cruzadas entre

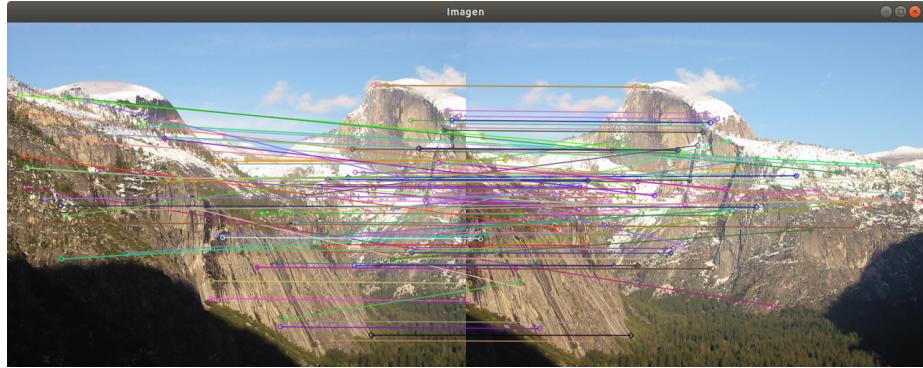


Figure 16: Matches BFCK

sí yendo a puntos que no se corresponden visualmente entre imágenes. Aún así los puntos que sí tienen una correspondencia visual clara entre ambas imágenes si se pegan de forma conveniente.

Para la detección de correspondencias Lowe-Average-2NN tenemos que aplicar los criterios que Lowe define en su paper. Debemos buscar las correspondencias esta vez con el crossCheck a False y con la función knnMatch que nos busca las correspondencias usando los dos puntos más cercanos, es decir con 2NN. Tras esto tenemos que aplicar el test definido por Lowe en el que sólo nos quedamos con puntos que distan poco entre sí, es decir que las correspondencias entre las imágenes sean cercanas con la intención de poder hacer un pegado de las mismas de buena calidad. Tras esta selección de Lowe tomamos la muestra de 100 puntos de la imagen al igual que en el caso anterior y obtenemos la imagen con las correspondencias.

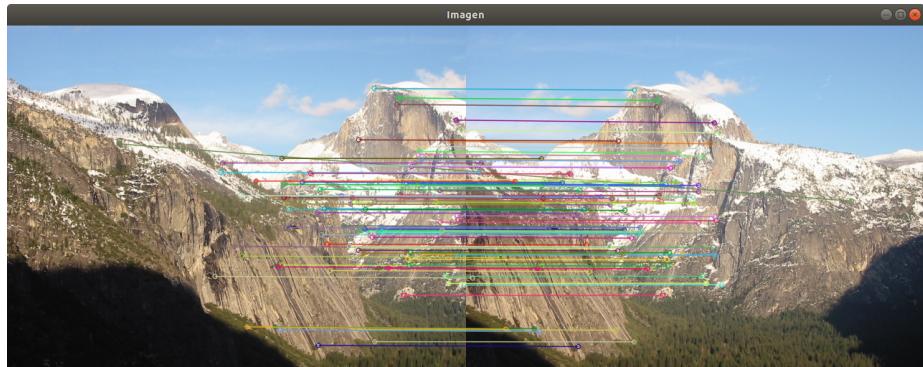


Figure 17: Matches LA2NN

Como podemos ver esta vez los puntos muy distantes entre las imágenes no han pasado el test de Lowe, de forma que sólo tenemos correspondencias cercanas

al colocar las dos imágenes juntas. Si nos paramos a observar el test lo que estamos es comparando la distancia obtenida de los dos vecinos al punto, es decir, distancias de dos correspondencias muy distantes serán rechazadas ya que tienen menos cohesión, al hacer esto es mucho más fácil que las correspondencias que obtengamos sean muy ajustadas a lo que percibimos visualmente, cosa evidente al comparar zonas muy similares de las imágenes.

Si comparamos los dos métodos podemos ver que en el primero obtenemos puntos repartidos por toda la imagen, de forma que hay muchas líneas cruzadas, ya que al ser puntos muy distantes entre sí no vamos a encontrar una correspondencia. Por ejemplo, si tomamos la imagen generada, podemos ver que en la montaña de la izquierda hay varios puntos de interés marcados. Estos puntos no tienen una correspondencia real en la imagen de al lado, puesto que dicha parte de la primera imagen no se encuentra en la segunda. Aquí es donde el test de Lowe nos da una gran mejora con respecto a la fuerza bruta.