

Prácticas de Visión por Computador

Grupo 2

Trabajo 2:

Detección de puntos relevantes y Construcción de panoramas

Pablo Mesejo

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA



Índice

- Normas de entrega
- Breve repaso teórico
- Presentación de la práctica

Índice

- **Normas de entrega**
- Breve repaso teórico
- Presentación de la práctica

Normas de la Entrega de Prácticas

- Dos opciones:
 - a) Se entrega memoria (PDF) y código (.py o .ipynb)
→ ZIP/RAR
 - b) Se entrega solamente un fichero .ipynb
integrando directamente el análisis y discusión

Normas de la Entrega de Prácticas

- Solo se entrega memoria y código fuente → no imágenes!
- Lectura de imágenes o cualquier fichero de entrada:
“imagenes/nombre_fichero”
- Todos los resultados numéricos serán mostrados por pantalla. No escribir nada en el disco.
- La práctica deberá poder ser ejecutada de principio a fin sin necesidad de ninguna selección de opciones.
- Puntos de parada para mostrar imágenes, o datos por terminal.

Entrega

- Fecha límite: 18 de Noviembre
- Valoración:
 - hasta 12 ptos (9 + 3 bonus), si se realiza toda la práctica con código propio, sin usar funciones de OpenCV en tareas de Visión por Computador (no se incluyen aquí cuestiones de visualización, p.ej).
 - Hasta 11 ptos (8 + 3 bonus), si se emplean funciones de OpenCV
- Lugar de entrega: PRADO
- Como siempre, **se valorará mucho la memoria:** descripción de qué se ha hecho y cómo, justificación de las decisiones tomadas, discusión y análisis de los resultados obtenidos

Dudas

Enviad todas las dudas que tengáis a
pmesejo@go.ugr.es

Si vemos que no es posible resolver las dudas por email, o en clase, concertaríamos una reunión para hacer una tutoría

Índice

- Normas de entrega
- **Breve repaso teórico**
- Presentación de la práctica

Prácticas anteriores

- P0: introducción a OpenCV

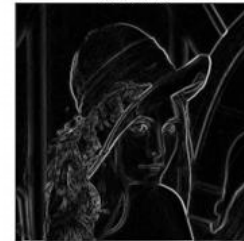


- P1: procesamiento de imágenes

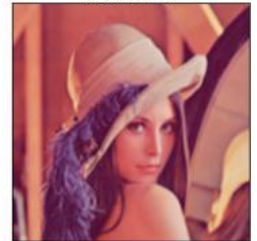
Original image



Sobel filter



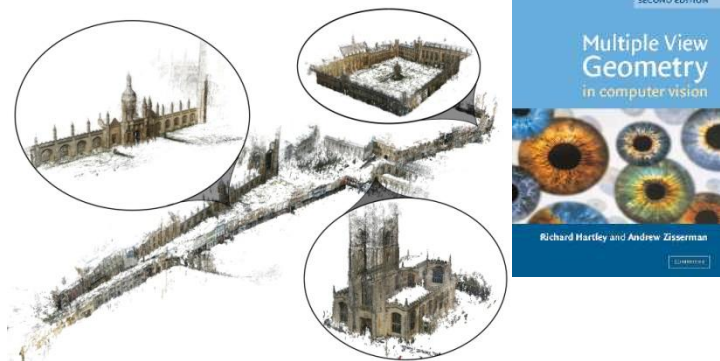
Gaussian blur



P2: detección de puntos de interés y creación de panoramas

Práctica 3

Geometría (*geometry*)



https://alexgkendall.com/computer_vision/have_we_forgotten_about_geometry_in_computer_vision/

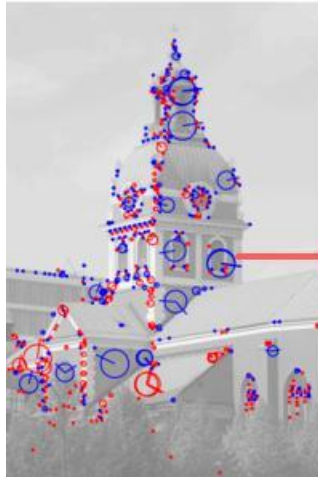
→
Aprendizaje (*learning*)



<http://cs231n.stanford.edu/>

Detección de puntos de interés

Feature = keypoint + descriptor



keypoints

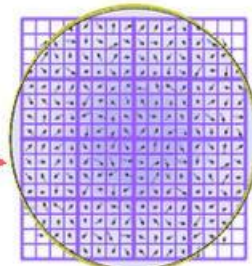
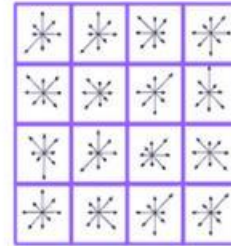
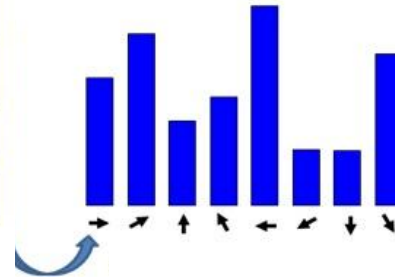


Image gradients



Keypoint descriptor



Feature descriptor

<https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/>

<https://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O>

Detección de puntos de interés

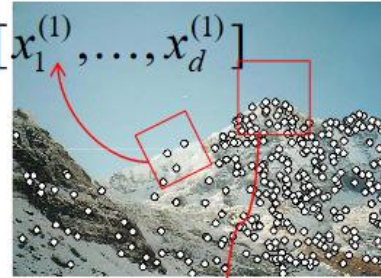


1) Detection: Identify the interest points



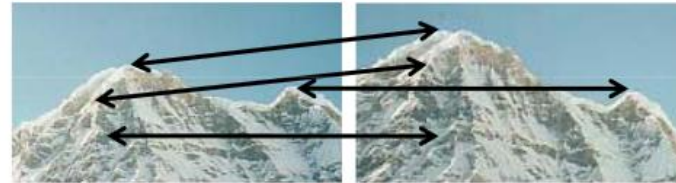
2) Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$

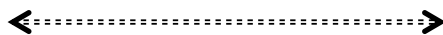


$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views

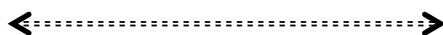


Detección de puntos de interés



Vectores describiendo
cada punto/región

FEATURE DETECTION



FEATURE DESCRIPTION

SIFT es a la vez un detector y un descriptor y, para la parte de detección de keypoints, utiliza la Diferencia de Gaussianas en imágenes reescaladas

Edge detection (Canny, Sobel,...)

Corner detection (Harris, FAST,...)

Blob detection (LoG, DoG,...)

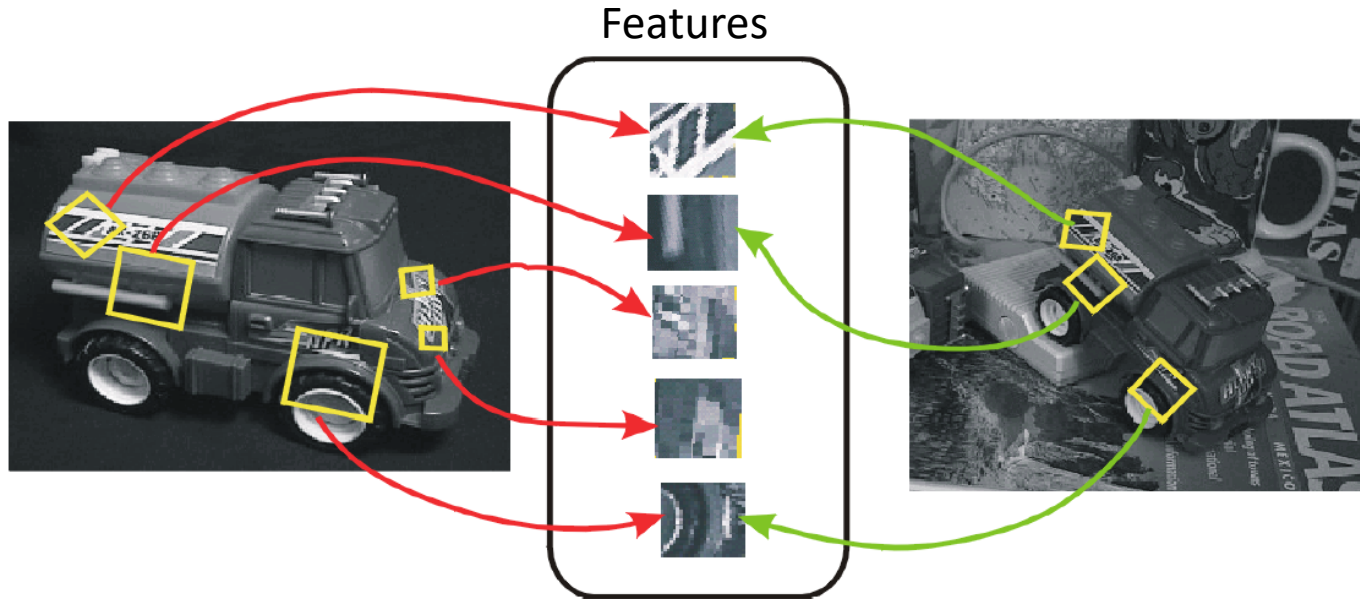
SIFT, SURF, HOG, KAZE,...

GLCM, LBP,...

Más información sobre *Feature Detection* con *OpenCV*:

https://docs.opencv.org/4.4.0/db/d27/tutorial_py_table_of_contents_feature2d.html

Detección y descripción de puntos de interés



Detection is *covariant*:

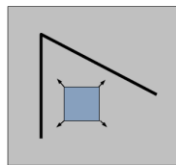
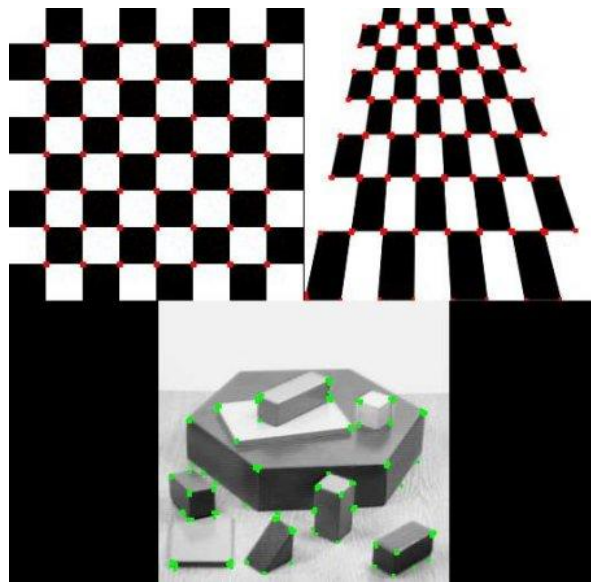
$$\text{features}(\text{transform}(\text{image})) = \text{transform}(\text{features}(\text{image}))$$

Description is *invariant*:

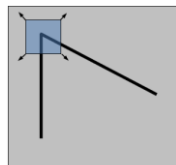
$$\text{features}(\text{transform}(\text{image})) = \text{features}(\text{image})$$

Importancia de la *scale invariance*

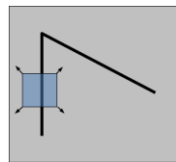
- Detector de Harris



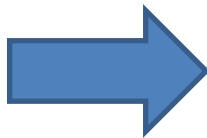
"flat" region:
no change in all
directions



"corner":
significant change in
all directions



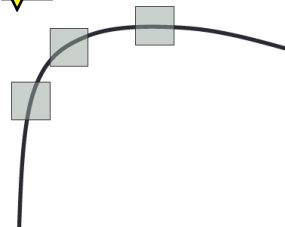
"edge":
no change along the
edge direction



- Rotation invariance



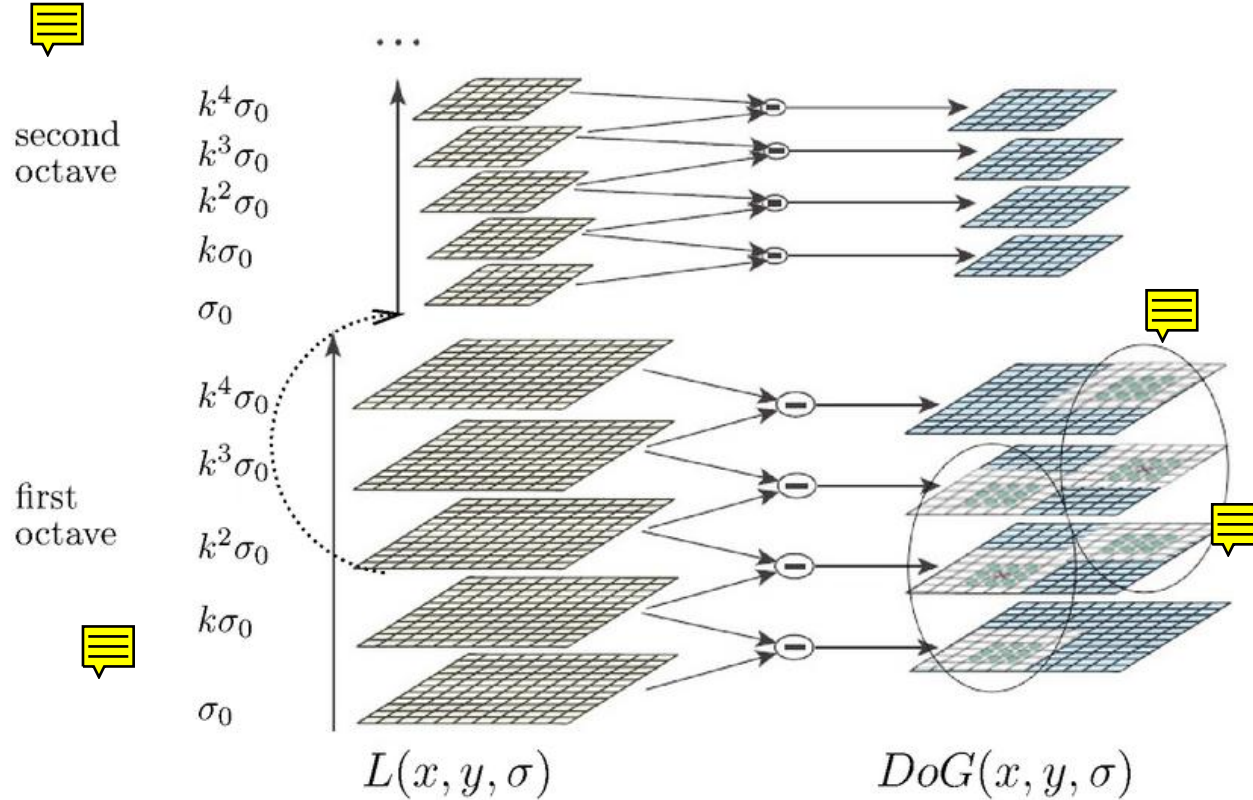
- Not invariant to *image scale*!



All points will be
classified as *edges*

Corner !

SIFT (Scale Invariant Feature Transform)



Invariante a:

- Traslación
- Rotación
- Escalado

Robusto a:

- Cambios de punto de vista
- Cambios de iluminación
- Ruido
- Cambios de contraste

Image extracted from Younes, L., Romaniuk, B., & Bittar, E. (2012). A comprehensive and comparative survey of the SIFT algorithm. In *International Conference on Computer Vision Theory and Applications* (Vol. 2, pp. 467-474)

SIFT (Scale Invariant Feature Transform)

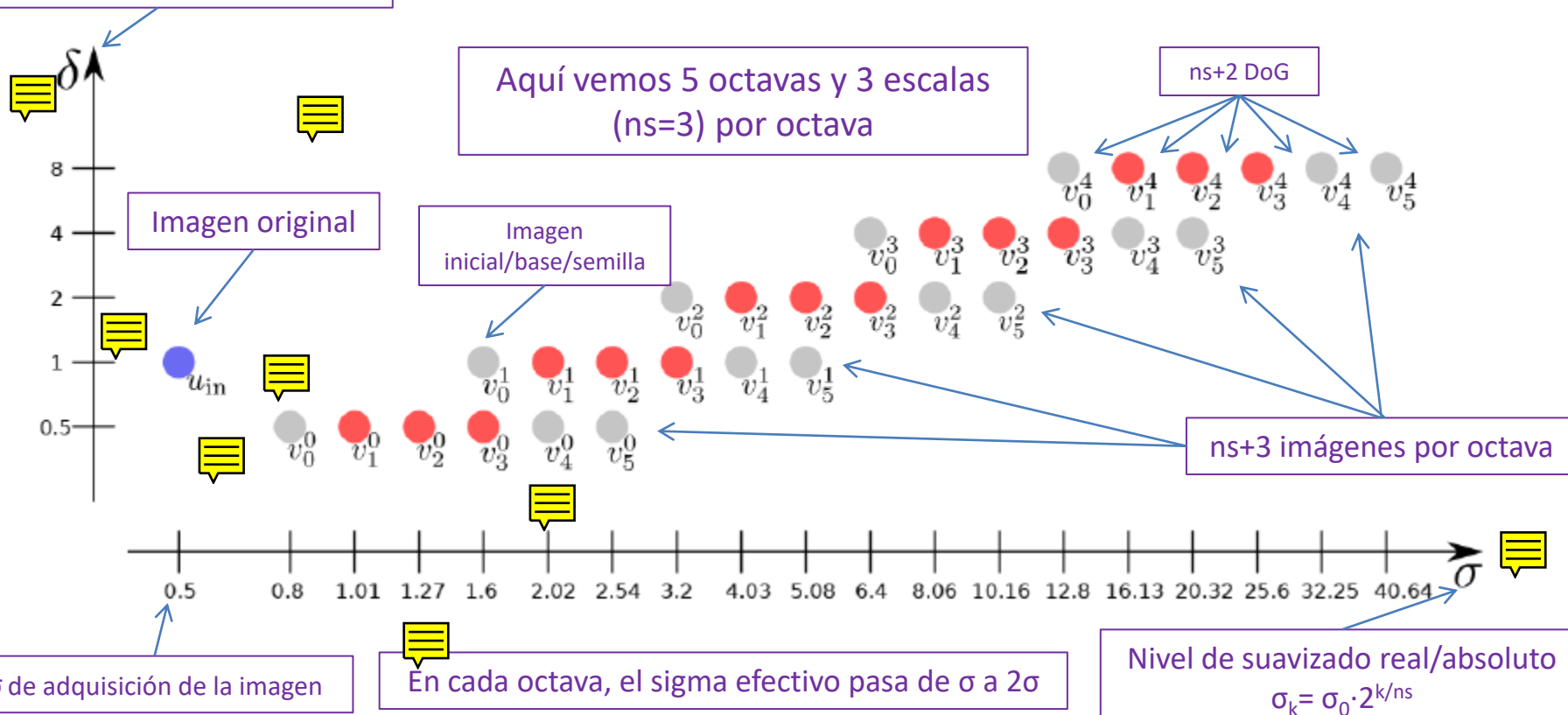
1. We have an input image with assumed $\sigma = 0.5$ (acquisition)
2. To correctly compute Differences-of-Gaussian on the input image, an octave with index -1 is included:
 1. Input image is smoothed to an initial scale ($\sigma_{ini} = 0.8$) (Lowe)
 2. Sub-sampling by bilinear interpolation with step $\delta_0 = 0.5$: $\sigma_0 = \frac{\sigma_{ini}}{\delta_0} = 1.6$
3. Assuming 3 scales by each octave, $ns = 3$, apply iterative smoothing with $\sigma_s = \sigma_0 \times \sqrt{2^{\frac{2s}{ns}} - 2^{\frac{2(s-1)}{ns}}}$, $s = 1, \dots, ns + 2$
4. Subsampling the scale with $ns = 3$ step $\delta = 2$ we compute the initial scale of the next octave. Use function `ceil()` to round the sizes.
5. Repeat 3 and 4 for each new octave.

SIFT (Scale Invariant Feature Transform)

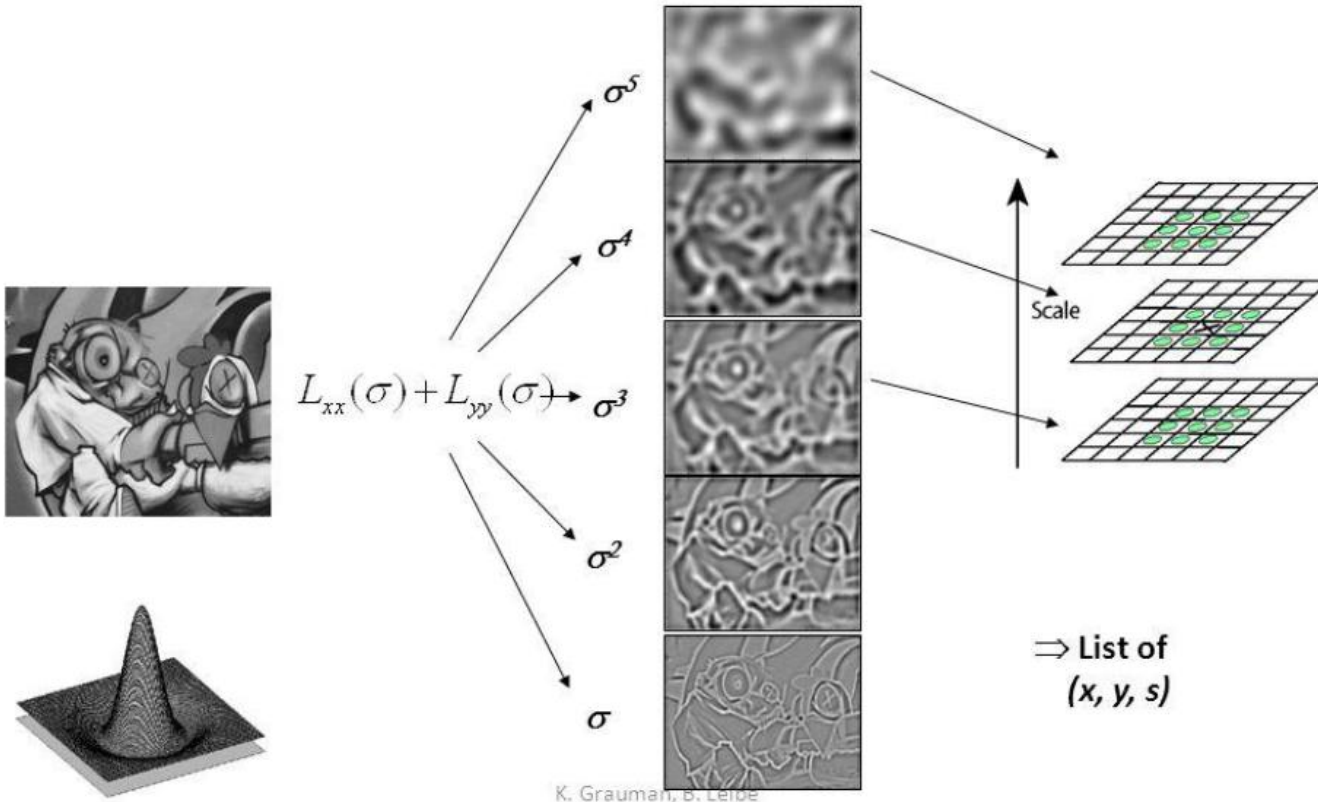
Distancia entre píxeles. $\delta=1$ para la imagen en tamaño original

"Anatomy of the SIFT Method" (Ives Rey-Otero, Mauricio Delbracio)

Image Processing On Line, 2014 <http://dx.doi.org/10.5201/ipol.2014.82>



SIFT (Scale Invariant Feature Transform)



Seleccionar el
píxel **X** si es el
máximo/mínimo
que sus 26
vecinos

Creación de Panoramas (*Image Stitching*)

- Proceso de combinar múltiples imágenes parcialmente solapadas para producir un panorama o imagen de alta resolución.



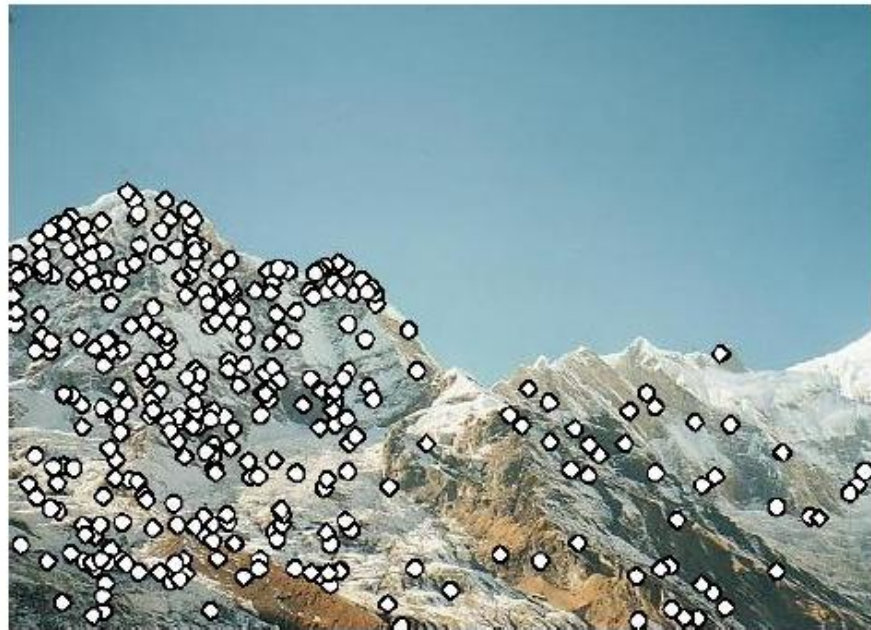
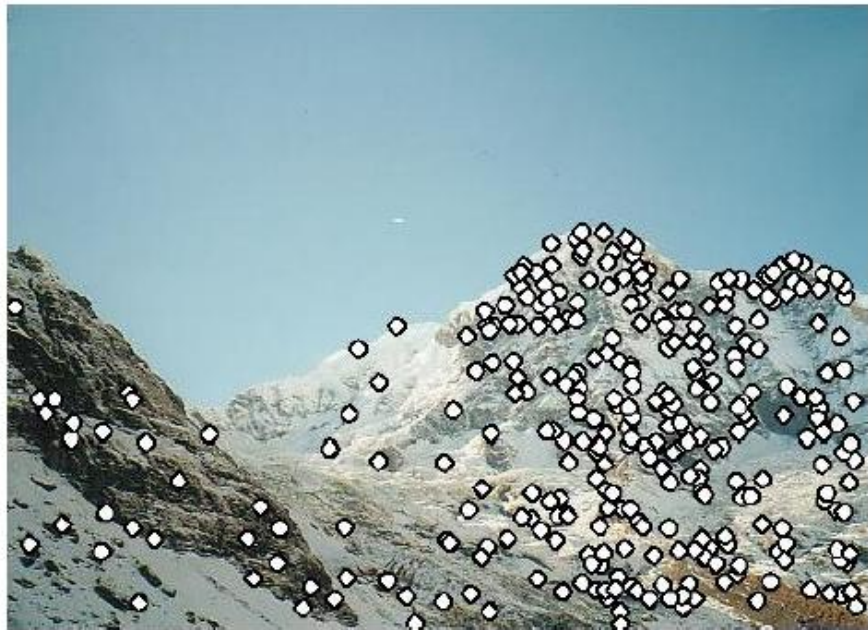
¿Cómo construimos un panorama?

- Necesitamos alinear imágenes 



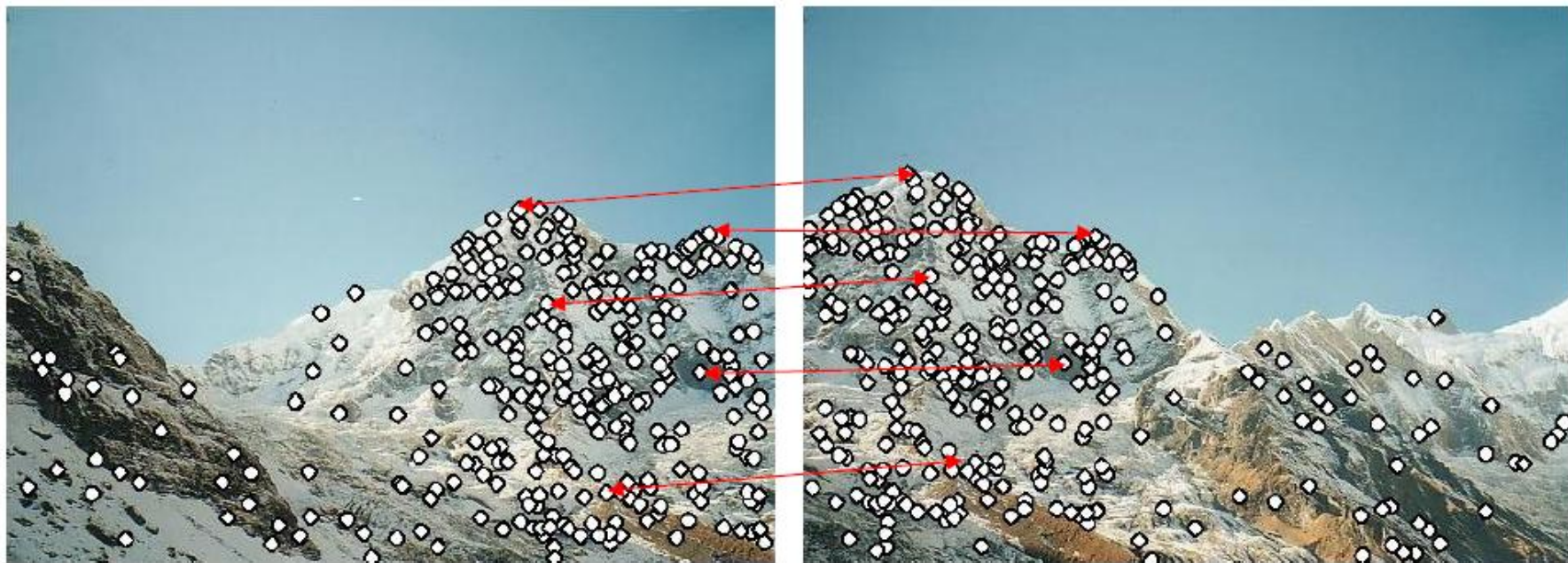
¿Cómo construimos un panorama?

- Detectamos puntos en ambas imágenes



¿Cómo construimos un panorama?

- Encontramos pares correspondientes



¿Cómo construimos un panorama?

- Usamos dichos pares para alinear las imágenes



Problema 1

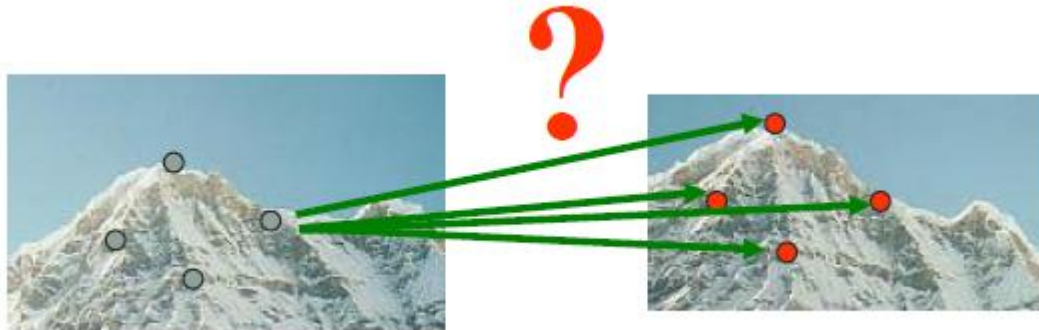
- Detectar el mismo punto independientemente en ambas imágenes
 - Necesitamos un **detector** repetible/estable



no chance to match!

Problema 2

- Para cada punto, debemos ser capaces de encontrar su punto correspondiente en la otra imagen
 - Necesitamos un **descriptor** fiable y distintivo

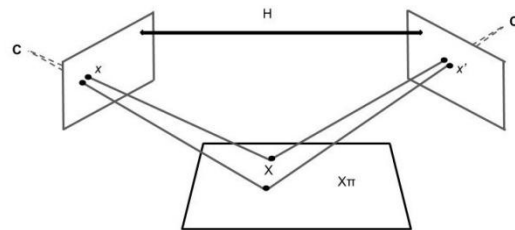
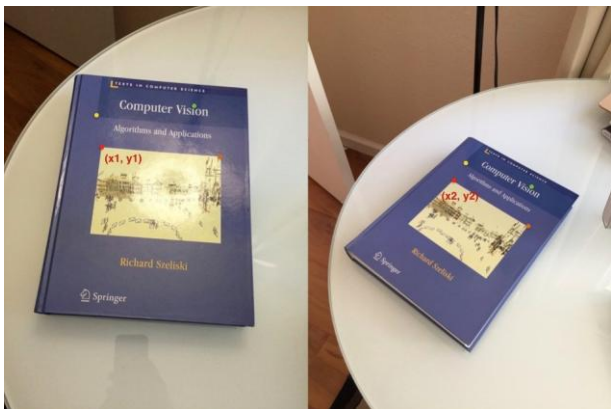


Construcción de Panoramas

- Aquí es donde entra en juego SIFT, como detector y descriptor
- Y la construcción de panoramas por medio de homografías

Homografías

- Cualesquiera dos imágenes de la misma superficie están relacionadas por una homografía.
- Una homografía es una transformación (matriz 3x3) que permite mapear (*map/warp*) puntos de una imagen en la otra.



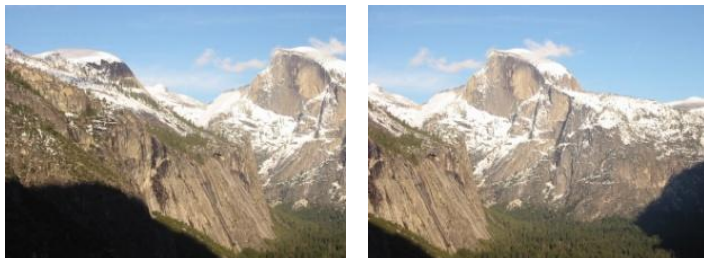
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Índice

- Normas de entrega
- Breve repaso teórico
- **Presentación de la práctica**

Ejercicio 1: detección de puntos en una pirámide de Lowe (4 pts)

Construir pirámide de Lowe de 4 octavas y 3 escalas para cada imagen de Yosemite.zip.



- a) ¿Qué operación sobre la imagen original nos permite fijar un $\sigma_0=1.6$?
- ¡Revisad la teoría! Punto 2 de “How to build it?”
 - NO aplicamos directamente ningún filtro sobre la imagen original (u_{in})!
 - Construir la imagen inicial/base del espacio de escalas
- b) Implementar una función que calcule las imágenes de la primera octava de la forma más **eficiente** y reusable posible
- ¡Kernels pequeños!
 - Función que proporciona todas las escalas de una octava, a la que se le pasa como entrada una imagen inicial, el número de escalas por octava, y σ_0
- c) Construir todas las octavas, y mostrar las primeras 3 escalas de cada octava.

Ejercicio 1: detección de puntos en una pirámide de Lowe (4 ptos)

- d) Calcular el espacio de escalas Laplaciano de la pirámide y extraer los 100 extremos locales con mayor respuesta.
- e) Mostrar la imagen con los extremos locales.
 - Círculo de radio 6σ sobre la escala de detección σ
 - Véase `cv2.KeyPoint()` y `cv2.drawKeypoints()`
- f) BONUS: Si todo el ejercicio se realiza con código propio se obtendrá 1 pto extra.

Ejercicio 2: descriptores SIFT (1.5 ptos)

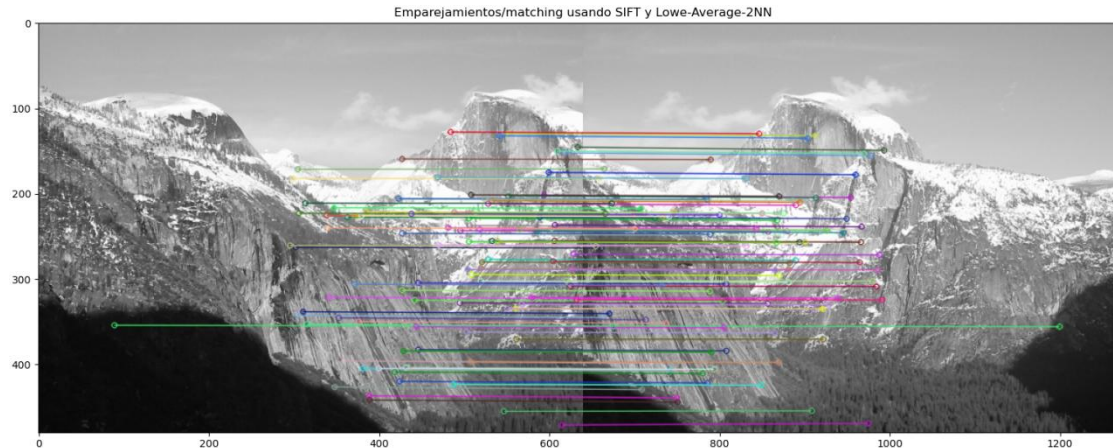
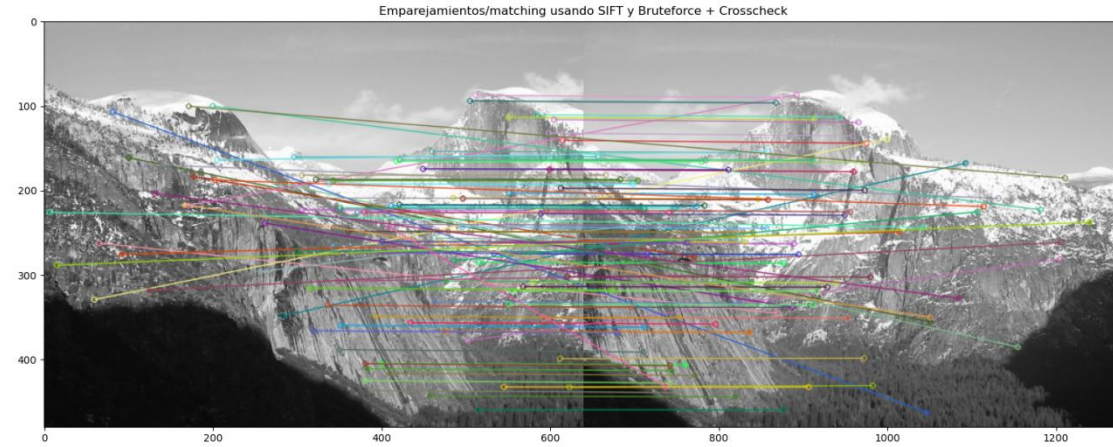
- Usar descriptores SIFT y calcular el emparejamiento/correspondencia entre las dos imágenes de Yosemite.

```
sift = cv2.SIFT_create()  
kpts1, desc1 = sift.detectAndCompute(img1, None)  
kpts2, desc2 = sift.detectAndCompute(img2, None)
```

https://docs.opencv.org/4.5.4/d7/d60/classcv_1_1SIFT.html

https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html

Ejercicio 2: descriptores SIFT (1.5 ptos)



Ejercicio 3: construir panorama con 3 imágenes (2.5 ptos)

- Se trata de hacer lo mismo que en el anterior ejercicio, pero ahora, junto con el cálculo de las correspondencias, aplicamos la transformación correspondiente para alinearlas.
- Se trabaja con 3 imágenes de mosaico.rar

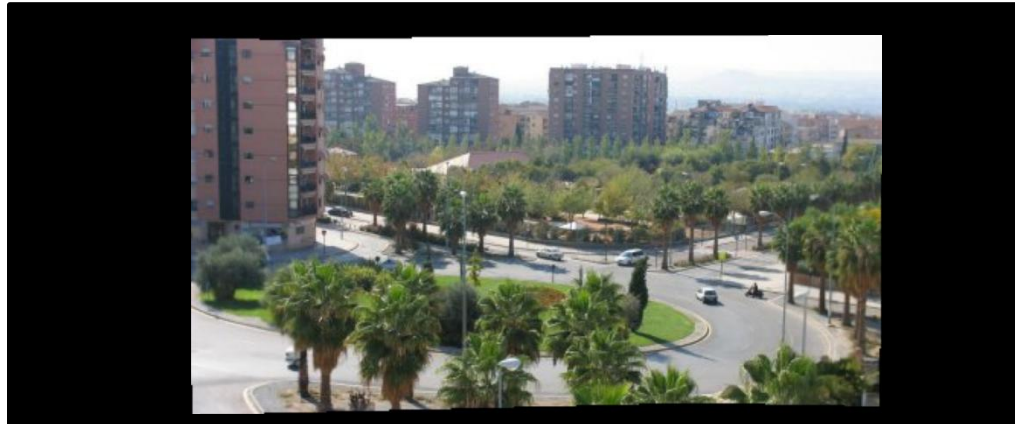


Ejercicio 3: construir panorama con 3 imágenes (2.5 pts)

- a) Extraer los KeyPoints SIFT y establecer un conjunto de puntos en correspondencias entre cada dos imágenes solapadas,
- b) Estimar la homografía entre las imágenes a partir de dichas correspondencias → `cv2.findHomography()`
- c) Componer y visualizar el mosaico obtenido → `cv2.warpPerspective()` con `cv.BORDER_TRANSPARENT`



Ejercicio 3: construir panorama con 3 imágenes (2.5 ptos)



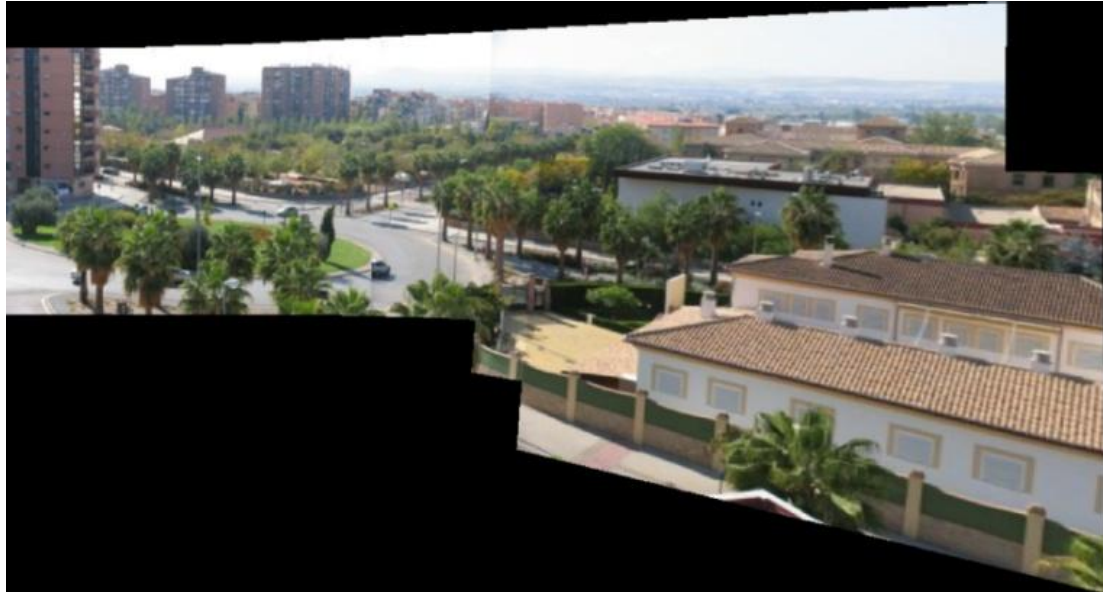
BONUS B1:

refinar extremos en espacio de escalas Laplaciano (1.5 ptos)

- Implementar el refinamiento de puntos extremos en el espacio de escalas Laplaciano. Aplicarlo a los puntos encontrados en el ejercicio 1, y mostrar que la implementación realmente mejora la estimación inicial.
 - Interpolare keypoint dentro del “cubo” en que ha sido detectado para moverlo a la mejor localización posible.
 - Revísese "3D-interpolation" de la teoría.

BONUS B2: construir mosaico completo (1.5 ptos)

- Igual que el ejercicio 3, pero empleando las 10 imágenes que están en mosaico.rar y, por tanto, reconstruyendo un panorama más amplio.



BONUS B2: construir mosaico completo (1.5 ptos)

- Importancia de seleccionar correctamente la imagen de referencia
 - Si la primera imagen en el canvas es la imagen de más a la derecha, al llegar al otro extremo la imagen estará muy deformada (fruto de la composición de múltiples homografías)



Referencias Útiles

- Interest Point Detector and Feature Descriptor Survey:
<https://core.ac.uk/download/pdf/81870989.pdf>
- Image Matching: <https://ai.stanford.edu/~syjeung/cvweb/tutorial2.html>
- Local features: detection and description.
https://www.cs.utexas.edu/~grauman/courses/trento2011/slides/Monday_LocalFeatures.pdf
- Documentación OpenCV relevante:
 - https://docs.opencv.org/4.5.4/d7/d60/classcv_1_1SIFT.html
 - https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html
 - https://docs.opencv.org/4.4.0/db/d27/tutorial_py_table_of_contents_feature2d.html
 - https://docs.opencv.org/4.4.0/d9/dab/tutorial_homography.html
 - <https://theailearner.com/tag/cv2-integral/>
 - https://docs.opencv.org/master/d7/d1b/group_imgproc_misc.html

Prácticas de Visión por Computador

Grupo 2

Trabajo 2:

Detección de puntos relevantes y Construcción de panoramas

Pablo Mesejo

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA

