Prácticas de Visión por Computador
Grupo 2

Presentación de la Práctica 3:
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

- Debéis entregar un único fichero Python (puede ser .py o .ipynb).
- El código debe estar comentado para facilitar su comprensión.
- Se puede entregar:
  - memoria (PDF) y código (Python), o
  - memoria y código integrados en un Cuaderno de Google Colab

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. Hay que fijar los parámetros que se consideren óptimos.
- Puntos de parada para mostrar imágenes, o datos por terminal.

Entrega

- Fecha límite: 30 de Diciembre
- Valoración:
  - hasta 13 ptos (8 + 5 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

  https://pradogrado2021.ugr.es/course/view.php?id=14596#section-4

- Como siempre, **se valorará mucho la memoria:** descripción de qué se ha hecho y cómo, justificación de las decisiones tomadas, discusión de los resultados obtenidos

Dudas

- Enviad todas las dudas que tengáis o solicitad tutorías online a pmesejo@go.ugr.es o pablomesejo@gmail.com.
  - Preferentemente Martes y Miércoles de 10:00 a 12:00 y Viernes de 11:30 a 13:30.

Índice

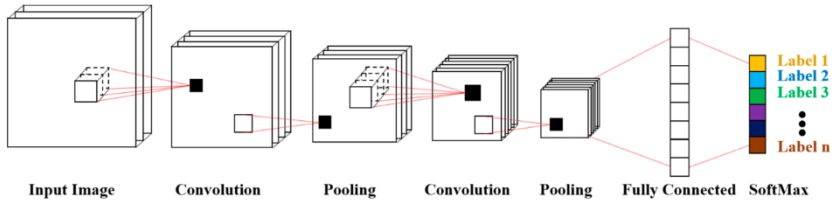
- Normas de entrega
- **Breve repaso teórico**
- Presentación de la práctica

Prácticas anteriores

- P0: introducción a OpenCV

- P1: filtrado de imágenes

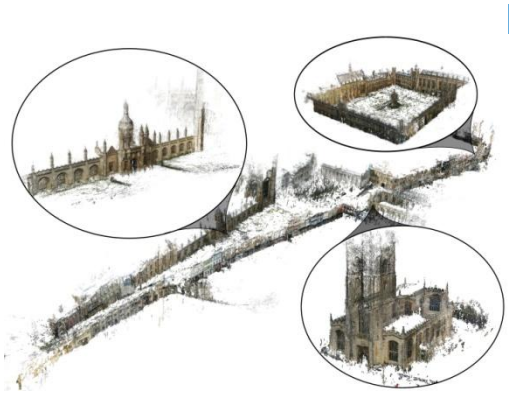
- P2: clasificación de imágenes con redes convolucionales profundas



P3: detección de puntos de interés y creación de panoramas

Aprendizaje (*learning*) → Geometría (*geometry*)

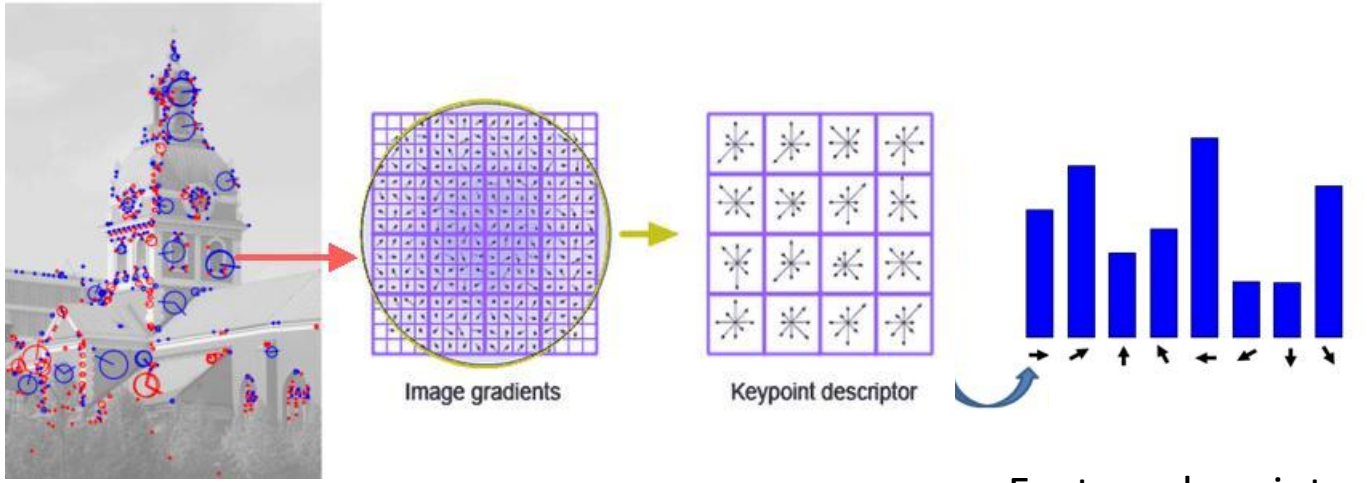
http://cs231n.stanford.edu/



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

Detección de puntos de interés

- Puntos de interés (*keypoints*) y características (*features*)



keypoints

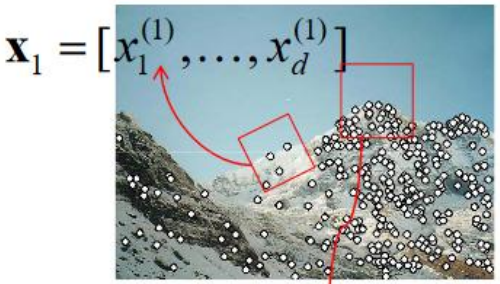
Feature descriptor

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

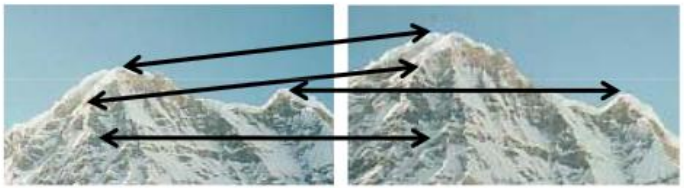
Detección de puntos de interés

1) Detection: Identify the interest points

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



3) Matching: Determine correspondence between descriptors in two views



https://www.cs.utexas.edu/~grauman/courses/trento2011/slides/Monday_LocalFeatures.pdf

Detección de puntos de interés

Vectores describiendo cada punto/región

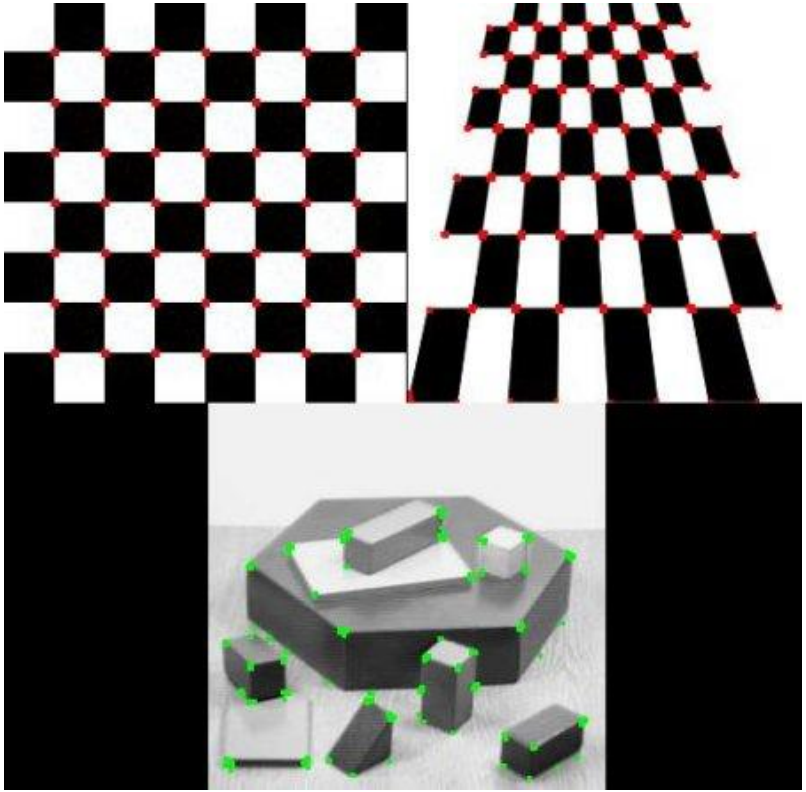
FEATURE DETECTION **FEATURE DESCRIPTION**

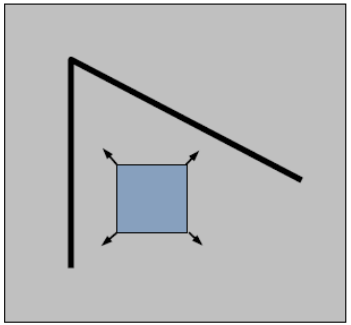
Edge detection (Canny, Sobel,…)
Corner detection (**Harris**, FAST,…)
Blob detection (LoG, DoG,…)

SIFT, SURF, HOG, **KAZE**,…
GLCM, LBP,…

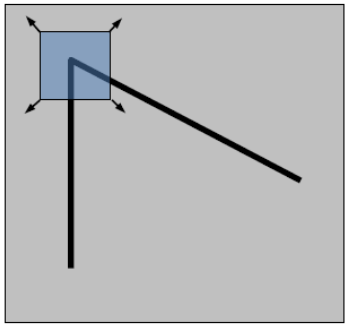
Más información sobre *Feature Detection* con *OpenCV*:
https://docs.opencv.org/4.4.0/db/d27/tutorial_py_table_of_contents_feature2d.html

Harris Detector

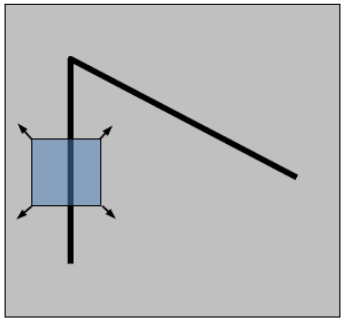




"flat" region:
no change in all directions



"corner":
significant change in all directions



"edge":
no change along the edge direction

Harris Detector

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

AKAZE Features

KAZE FEATURES

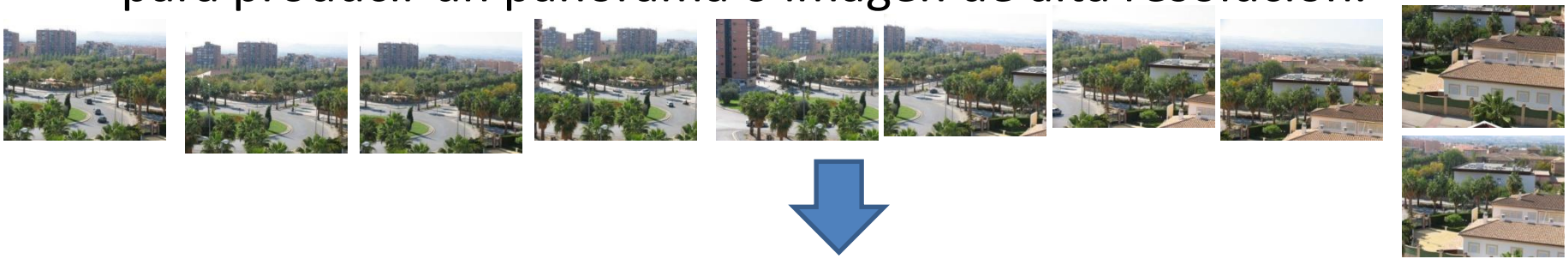
KAZE Features. Pablo F. Alcantarilla, Adrien Bartoli and Andrew J. Davison. In European Conference on Computer Vision (ECCV), *Fiorenze, Italy, October 2012.* bibtex

ACCELERATED-KAZE FEATURES

Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. Pablo F. Alcantarilla, Jesús Nuevo and Adrien Bartoli. In British Machine Vision Conference (BMVC), *Bristol, UK, September 2013.* bibtex

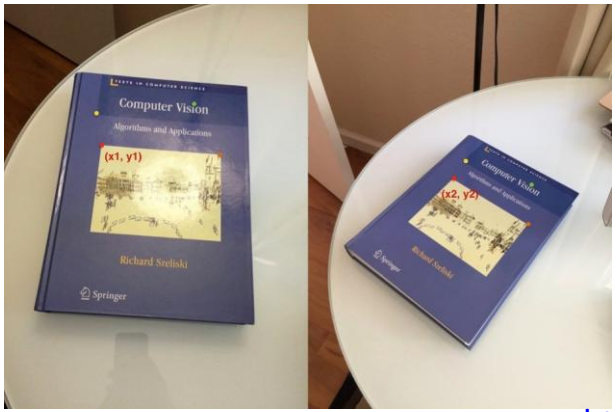
Creación de Panoramas (*Image Stitching*)

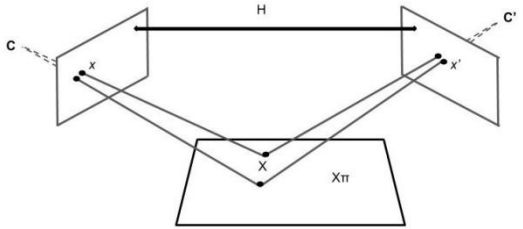
- Proceso de combinar múltiples imágenes parcialmente solapadas para producir un panorama o imagen de alta resolución.



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.





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

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

Índice

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

Ejercicio 1: detección de puntos de Harris (3 ptos)

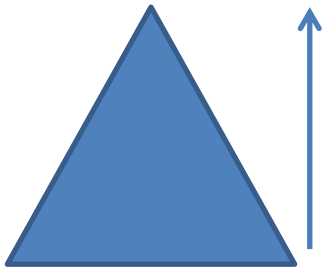
1. Construir pirámide Gaussiana de 3 niveles (σ=[1,1.5]) para cada imagen en la carpeta Yosemite.zip.


2. Calcular el criterio de Harris en cada píxel de cada uno de los niveles de la pirámide.
   - blockSize: tamaño de la ventana utilizada para detectar puntos Harris. En el nivel más bajo de la pirámide: 3x3, 5x5 o 7x7
   - Ksize: tamaño de filtro para calcular la derivada (ksize de 3 o 5)
3. Supresión de no máximos en cada nivel de la pirámide (2000 puntos en total).
   - Umbral mínimo del criterio *f* de Harris que debe ser establecido de modo empírico.
   - El tamaño de la ventana de búsqueda para realizar dicha supresión lo podéis escoger vosotros para conseguir el número de puntos que se os solicita. Posibles ventanas: 5x5, 7x7, 9x9, 11x11
     - Si el centro es mayor que todo lo que tiene alrededor → máximo, y suprimimos el resto

Ejercicio 1: detección de puntos de Harris (3 ptos)

4. Extracción de puntos en cada nivel (70%-25%-5%): aproximadamente 2000 puntos en total entre todos los niveles (~1400 ptos – ~500 ptos – ~100 ptos).
   - A cada punto se le asocia una estructura cv2.KeyPoint() con coordenadas (x,y), escala y orientación.
   - Estimar la escala teniendo en cuenta los niveles de la pirámide

     escala_nivel_{i} = 2*escala_nivel_{i-1} + 1;
     con escala_nivel_{1} = blockSize = 3x3



Nivel_3 = 2*7x7 + 1 = 15x15

Nivel_2 = 2*3x3 + 1 = 7x7

Nivel_1 = 3x3

   - Calcular la orientación tal y como se indica en http://matthewalunbrown.com/papers/cvpr05.pdf: alisar la imagen (σ=4.5) y calcular dirección del gradiente en ese punto.

5. Refinar la posición de dichos puntos a nivel subpíxel, y mostrar las localizaciones originales y las refinadas haciendo zoom en 3 regiones diferentes de la imagen.

Ejercicio 1: detección de puntos de Harris (3 ptos)

- Algunas ideas:
  - Principalmente, los parámetros con los que podéis jugar son el tamaño de la ventana para supresión de no máximos, el sigma de la pirámide Gaussiana, el umbral mínimo para el criterio de Harris, el filtro/ksize para calcular derivadas, y el blockSize (escala para encontrar puntos Harris)
  - Tenéis que ordenar los keypoints (por el criterio de Harris dentro de cada escala) y quedaros con 2000.
    - Primero, implementad una sola escala y, cuando todo os funcione bien, pasad al escenario multiescala. Es decir, la construcción de la pirámide va después de que os aseguréis de que un único nivel está funcionando.
  - Leed el paper de referencia para este apartado (http://matthewalunbrown.com/papers/cvpr05.pdf): Brown, Matthew, Richard Szeliski, and Simon Winder. "Multi-image matching using multi-scale oriented patches." *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005.

    https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html

Ejercicio 2: descriptores AKAZE (1.5 ptos)

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

```
akaze = cv.AKAZE_create()
kpts1, desc1 = akaze.detectAndCompute(img1, None)
kpts2, desc2 = akaze.detectAndCompute(img2, None)
```

https://github.com/pablofdezalc/kaze

http://www.robesafe.com/personal/pablo.alcantarilla/kaze.html

Ejercicio 3: construir mosaico con 3 imágenes (1.5 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 4: construir mosaico con 10 imágenes (2 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 (3 puntos)

- B1- Implementar de forma eficiente (tiempo real) el algoritmo RANSAC para homografías a partir de correspondencia de puntos Harris (1.5 puntos)

- B2- Implementar todo el ejercicio 1 de detección de puntos Harris con código propio. Comparar con los resultados de OpenCV para los mismos valores de los parámetros de entrada (1.5 puntos) (Obligatorio que el ejercicio 1 sea correcto: al menos 75% de la puntuación)

- B3- Escribir una función que ejecute de forma eficiente (paralela) la interpolación de los valores de todos los píxeles de una imagen. Probarla comparando el resultado con el ejercicio 4 (1.5 puntos) (Obligatorio que el ejercicio 4 sea correcto: al menos 75% de la puntuación)

Referencias Útiles

- Interest Point Detector and Feature Descriptor Survey: https://core.ac.uk/download/pdf/81870989.pdf
- Image Matching: https://ai.stanford.edu/~syyeung/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/master/dc/d0d/tutorial_py_features_harris.html
  - https://docs.opencv.org/4.4.0/db/d70/tutorial_akaze_matching.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

Presentación de la Práctica 3:
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