

# **LAB 4: PANORAMA**

**VISIÓN POR COMPUTADOR**

**10/04/2022**

---

**Aarón Ibáñez Espés - 779088  
Sergio Gabete César - 774631**

## Instalación de librerías

Para instalar las librerías nuevas se ha descargado la librería opencv\_contrib-4.2.0 desde el repositorio de GitHub. A continuación se ha extraído la carpeta xfeatures2d de la carpeta modules y se ha introducido en el módulo de opencv-4.2.0. Finalmente se ha vuelto a instalar la librería de openCV entera pero al comando cmake se le han proporcionado los flags -D BUILD\_opencv\_xfeatures2d=ON y -D OPENCV\_ENABLE\_NONFREE=ON para que se instale correctamente xfeatures2d.

## Parte 1. Extracción de características y emparejamiento

Para la extracción de puntos de interés de las imágenes se ha decidido realizarlo sobre las imágenes en tono de grises pero luego se mostrarán sobre las imágenes normales.

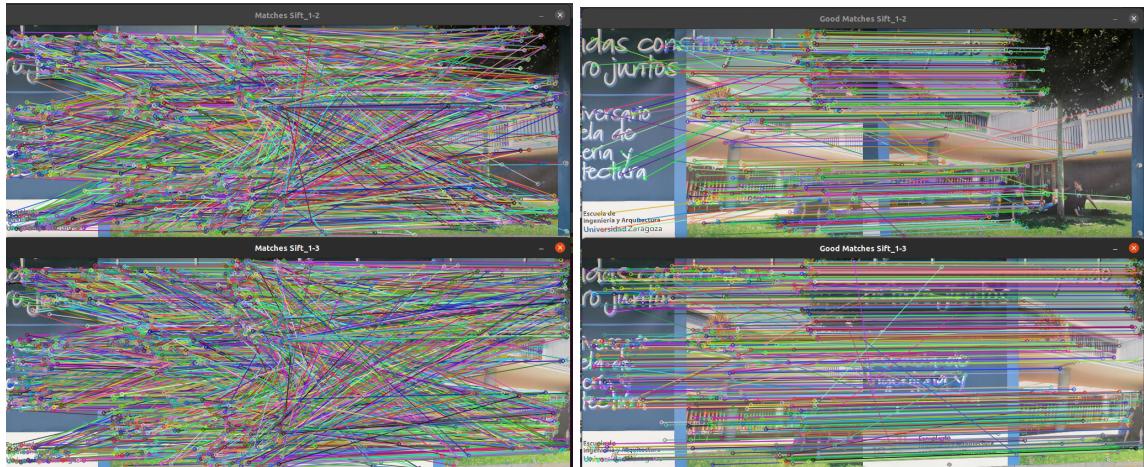
Tras pasar las imágenes a tono de grises se extraen los keypoints con los distintos detectores (Harris, Sift, Surf, Orb y Akaze). Tras obtener los keypoints de las imágenes es necesario obtener los distintos descriptores de esos keypoints para poder relacionarlos con los de otra imagen. Una vez obtenidos los keypoints se procede a encontrar los keypoints de la primera imagen que corresponden con los de la segunda. Se va a utilizar el método de fuerza bruta para los detectores Harris, Orb y Akaze y para Sift y Surf se va a usar el método Flann (ya que su uso no es compatible con Harris, Orb o Akaze). Una vez se han obtenido los matches entre keypoints se deben filtrar los que no sirven o no cumplen un umbral. Esto se puede realizar comprobando el vecino más próximo y con el ratio al segundo vecino (se ha usado un ratio de 0.8, si se disminuye el ratio aparecerán menos keypoints pero serán mejores matches). Una vez obtenidos los buenos matches se mostrarán en ambas imágenes. Para probar su funcionamiento se han usado tres imágenes de un póster que se ha encontrado en el edificio Betancourt y para el escenario 3D se han usado fotos de una maqueta 3D que se ha encontrado en el edificio Torres Quevedo. El principal parámetro que se ha ido modificando es el número de features detectadas (por defecto son 500 pero se ha ampliado a 1000).

En el Anexo1 se pueden ver las imágenes que se han usado como prueba. A continuación se muestran los resultados obtenidos de los distintos métodos.

- Surf



- Sift



- Harris



- Orb



- Akaze

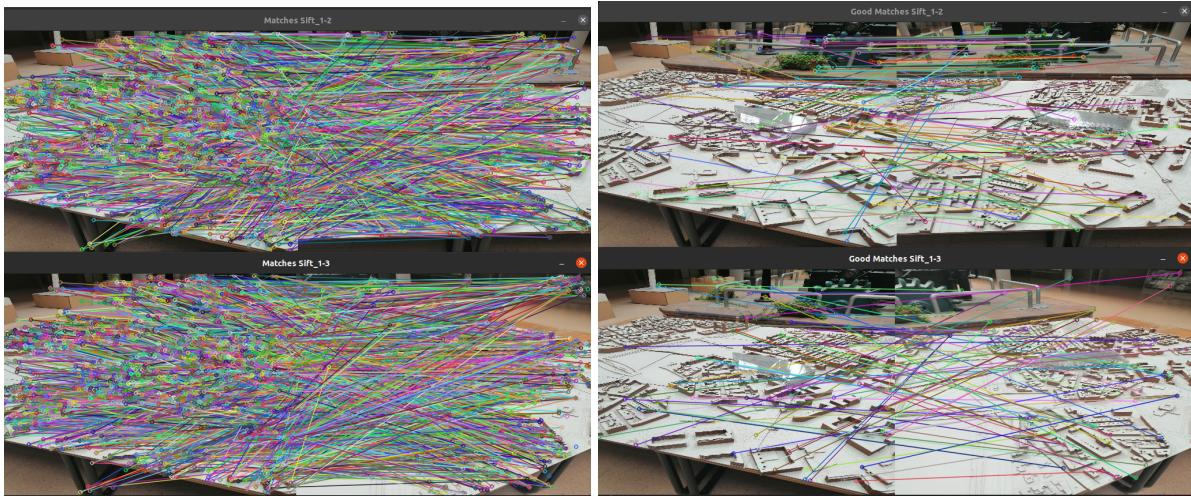


Tras analizar los resultados de los distintos detectores se puede observar que los matches son demasiado variados y bastante malos hasta que se filtran con el método del ratio al segundo vecino. Se van a comparar los resultados solo para los matches filtrados. Se puede observar que Sift, Surf y Akaze tienen buenos matches pero Orb y Harris tienen pocos.

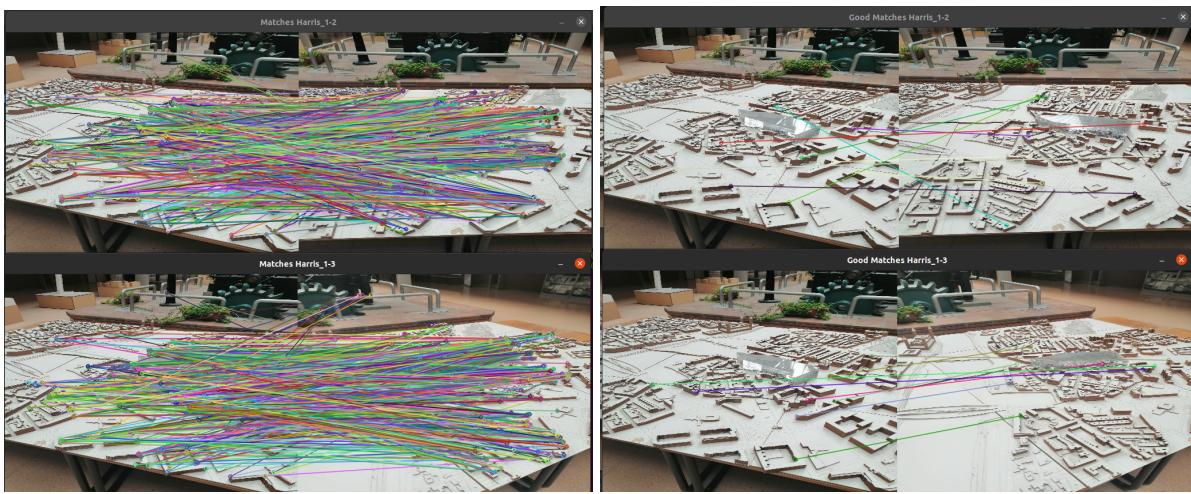
- Surf



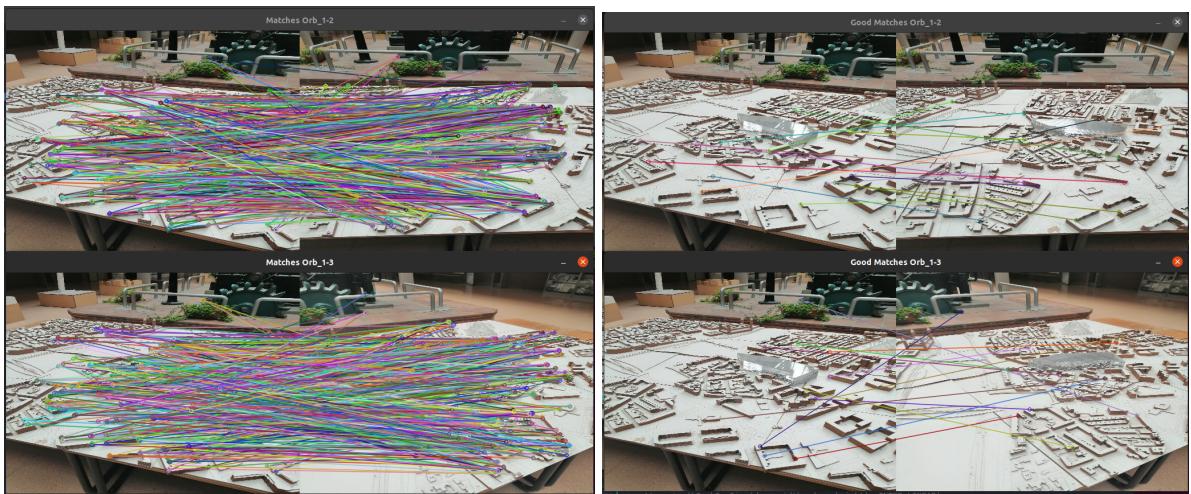
- Sift



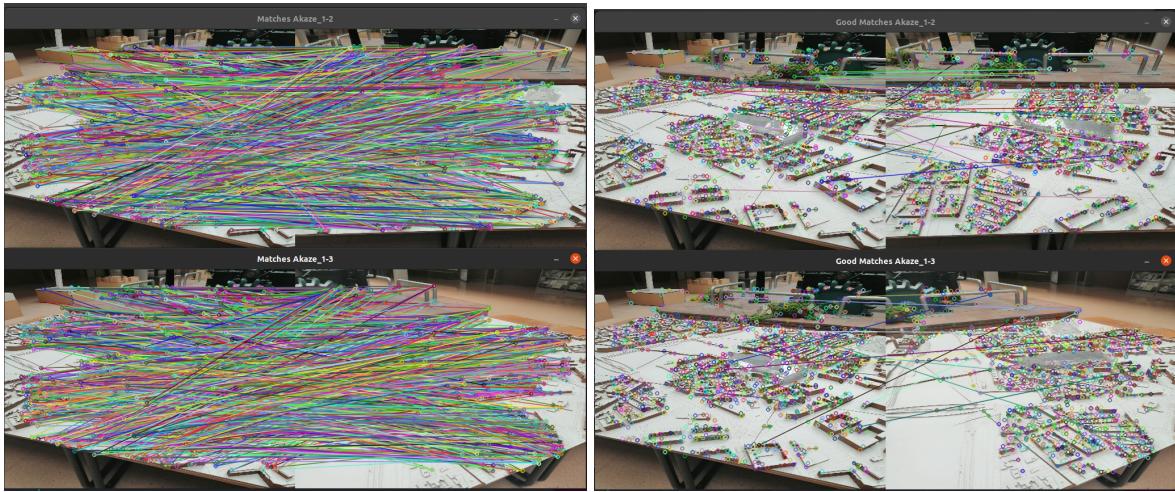
- Harris



- Orb



- Akaze



En el escenario 3D pasan cosas similares al escenario 2D pero en este caso Akaze funciona peor. Finalmente se ha decidido usar Sift ya que es el que mejor funciona en ambos casos.

## Panorama

El siguiente paso de la práctica es crear un panorama con distintas imágenes. El primer paso es encontrar la homografía de las imágenes. La homografía determina la correspondencia entre los keypoints de cada imagen. Para encontrar la homografía se deben obtener los matches como se ha realizado en el apartado anterior. Una vez obtenidos los matches se obtiene la homografía con la función `findHomography()` de openCV.

Tras obtener la homografía se procede a añadir la imagen nueva al panorama. Para añadir la imagen hay que corregir los tamaños de imagen y como se va a pegar con la matriz de homografía. Para encontrar el punto de la nueva imagen el panorama se multiplica el punto por la matriz de homografía (es una transformación). También hay que buscar la dirección en la que se va a colocar la imagen ya que se puede añadir en varias direcciones (simplemente se observa donde están los puntos de la nueva imagen tras la transformación). Luego se actualiza la homografía para que sea correcta con los nuevos puntos. Por último se utiliza la función `warpPerspective()` de openCV para que se ejecute una transformación perspectiva a la nueva imagen y luego se copia encima la imagen previa del panorama.

Se van a tomar fotos de la cámara en distintas direcciones para ver que funciona correctamente.

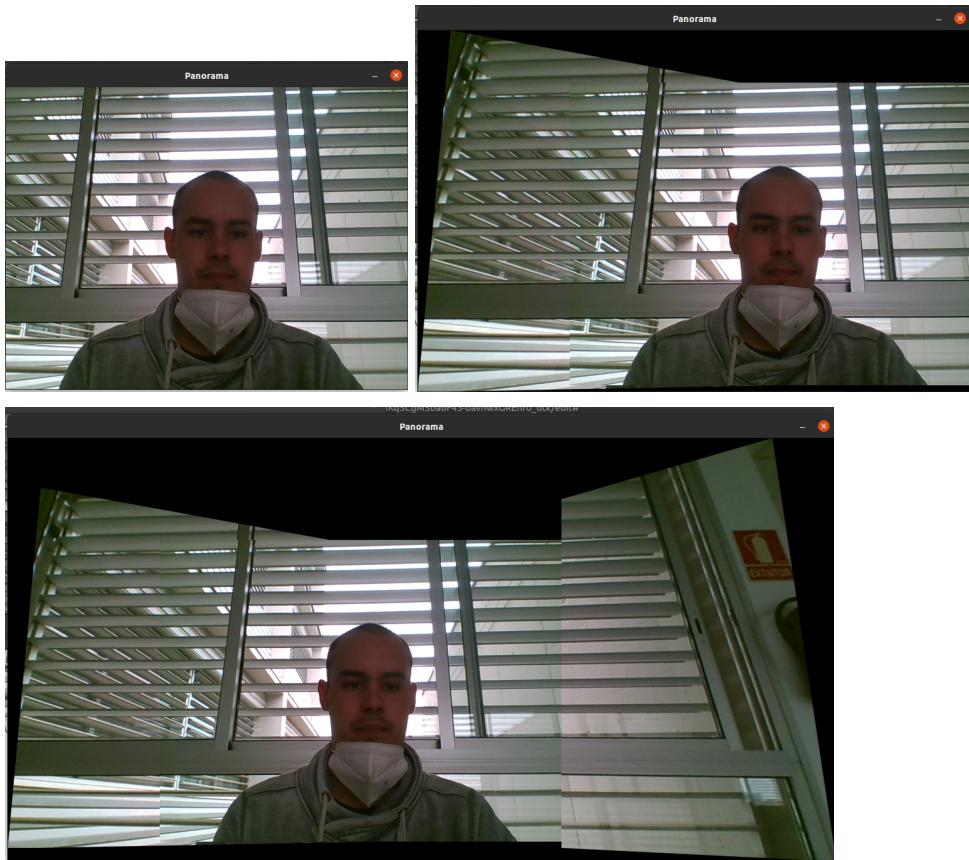
- Añadir imágenes por arriba



- Añadir imágenes por abajo



- Añadir imágenes a los lados



Se puede observar como al añadir varias imágenes el panorama se va deformando cada vez más y aparecen franjas negras.

## Anexo 1

Imágenes de prueba para el escenario 2D





Imágenes de prueba para el escenario 3D





