ÍNDICE GENERAL

CAPÍTULO 1 INTRODUCCIÓN 3

CAPÍTULO 2 DESARROLLO 4

CAPÍTULO 3 RESULTADOS 6

3.1 DESCRIPCIÓN SOBRE ESCENAS 6

3.1.1 Filtrado de matches 10

3.2 IMAGEN PLANA 11

3.3 VÍDEO 12

CAPÍTULO 4 CONCLUSIONES 13

BIBLIOGRAFÍA 14

ÍNDICE DE FIGURAS

FIGURA 1. COMPARACIÓN VISUAL ENTRE ORB Y BRISK 8

FIGURA 2. COMPARACIÓN BRISK VS ORB ESCENA 4 9

FIGURA 3. ESCENA 4 ORB CON 4000 KEYPOINTS 10

FIGURA 4. ESCENA 7 CON IMAGEN AÑADIDA 11

FIGURA 5. ESCENA 6 CON IMAGEN AÑADIDA 11

FIGURA 6. ESCENA 7 CON VÍDEO AÑADIDO 12

FIGURA 7. ESCENA 3 CON VÍDEO AÑADIDO 12

Capítulo 1

INTRODUCCIÓN

En este proyecto se plantea crear un sistema de Realidad Aumentada que funcione sin apoyarse en marcadores artificiales, utilizando en su lugar superficies planas y texturizadas como referencia.

El problema principal que resolver radica en identificar de manera robusta la superficie de interés en distintas condiciones de iluminación y perspectiva, y proyectar correctamente sobre ella imágenes o vídeos. Para ello, se ha usado OpenCV empleando detectores y descriptores de puntos clave que permitan una detección y emparejamiento efectivos.

En definitiva, este proyecto tiene como objetivo ofrecer una solución al problema de la Realidad Aumentada sin marcadores y poner en práctica los conocimientos adquiridos en la asignatura.

Capítulo 2

DESARROLLO

El trabajo en la práctica se ha dividido en 4 tareas de las cuales las 3 primeras se completaron en el transcurso de las clases.

1. **Cálculo de homografía y deformaciones básicas**

En esta primera tarea se establecen las bases del sistema de Realidad Aumentada. Por un lado, se calcula la homografía [1][3][5] que permite encontrar la transformación proyectiva entre dos planos, utilizando correspondencias de puntos clave entre la imagen modelo y la escena. Además, se implementan funciones auxiliares para dibujar el contorno de la imagen proyectada y para deformar imágenes de acuerdo con la homografía obtenida.

2. **Detección y descripción de puntos clave**

Aquí se introduce la capacidad de reconocer patrones automáticamente en las imágenes. Se detectan keypoints y se generan descriptores usando el detectores como ORB [7] o BRISK [8]. Posteriormente, se realiza la correspondencia entre descriptores de dos imágenes mediante un comparador de fuerza bruta, estableciendo los vínculos necesarios para identificar el objeto en la escena. También se añade una función para visualizar gráficamente los matches, para poder visualizar cómo está funcionando nuestro programa.

3. **Localización del objeto y ajuste de la homografía**

Una vez obtenidos los matches, esta tarea se centra en localizar el objeto modelo dentro de la imagen de la escena. Para ello, se calculan las homografías que permitirán situar correctamente el modelo en la escena real.

4. Filtrado de matches y superposición de elementos virtuales

En esta última fase se filtran los matches para eliminar correspondencias erróneas o poco fiables, basándose en la distancia de los descriptores. Después, se desarrollan funciones que permiten mostrar información superpuesta sobre la imagen como si fuera una placa informativa. También se da la posibilidad de dibujar un patch, que será una imagen, sobre la escena usando la homografía calculada. Reutilizando la idea del patch, pero iterando sobre cada frame, también se puede proyectar un vídeo o cámara.

Capítulo 3

RESULTADOS

3.1 Descripción sobre Escenas

A continuación, se presenta una comparación de los resultados obtenidos aplicando los detectores de keypoints ORB y BRISK. Se han seleccionado las imágenes con resoluciones más adecuadas y se han recogido los puntos clave que recogen cada método, así como los matches y matches filtrados que resultan de cada ejecución. También se aporta capturas de pantalla para poder realizar una comparativa visual del funcionamiento de cada método.

Escena	Detector keypoints	Keypoints modelo	Keypoints escena	Matches	Matches filtrados
2	ORB	500	500	173	173
	BRISK	4425	5078	1456	242
3	ORB	500	500	150	13
	BRISK	4425	8692	1751	152
4	ORB	500	500	166	166
	BRISK	4425	2715	1041	677
6	ORB	500	431	155	125
	BRISK	4425	941	506	26
7	ORB	500	500	172	61
	BRISK	4425	9329	1890	313
8	ORB	500	500	191	140
	BRISK	4425	2416	1152	195
9	ORB	500	397	154	154
	BRISK	4425	1136	654	642
10	ORB	500	500	157	27
	BRISK	4425	29881	2887	133

Tabla 1. Resultados BRISK y ORB

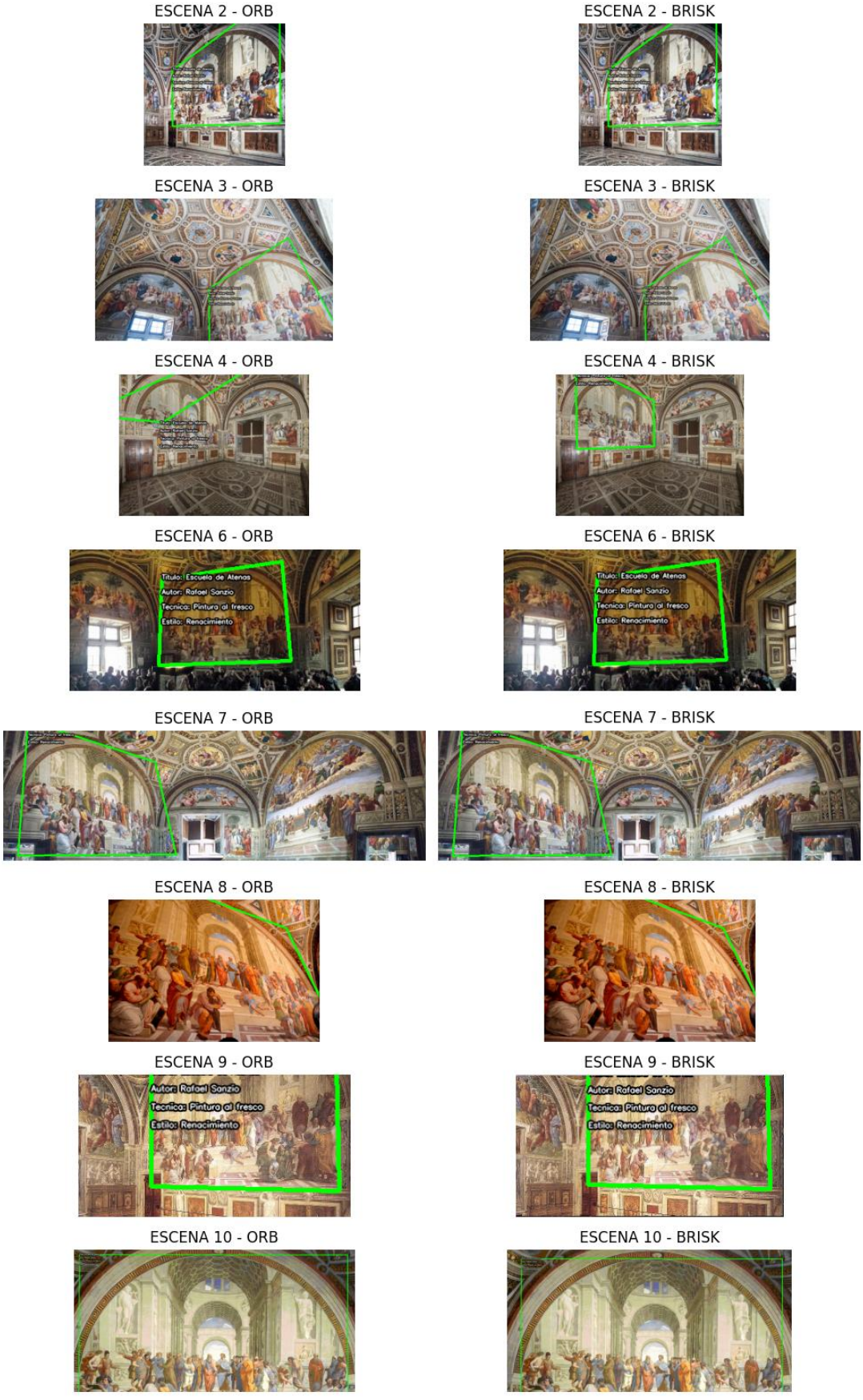

Figura 1. Comparación visual entre ORB y BRISK

Podemos observar como a nivel visual no se observa cambios significativos, ambos métodos presentan un rendimiento prácticamente idéntico. Aunque cabe destacar que ORB puede detectar un máximo de 500 puntos clave de forma predeterminada, pero esto es un parámetro que podemos modificar (nfeatures). De igual forma, con estos 500 puntos ha demostrado ser suficiente para tener unos buenos resultados.

Sin embargo, sí vemos que para imágenes que suponen un mayor reto para emparejar, como la escena 4, BRISK presenta unos resultados significativamente mejores.

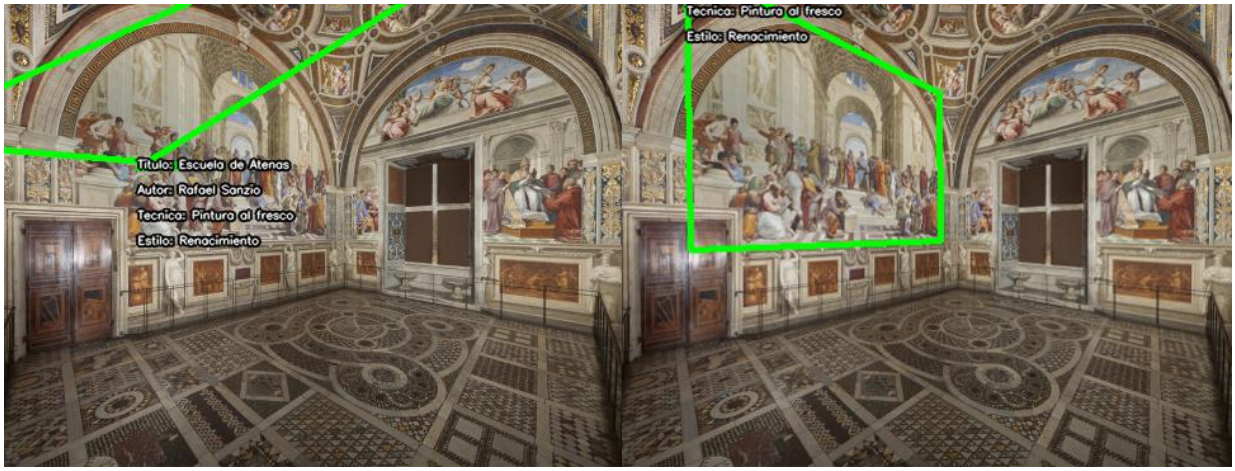

Figura 2. Comparación BRISK vs ORB escena 4

En este caso sería adecuado subir el número de puntos clave máximos que puede obtener ORB. Estableciendo este límite a 4000 obtenemos el siguiente resultado (Figura 3), el cual presenta una gran mejora respecto a los 500 keypoints. Aunque sigue siendo ligeramente peor que el de BRISK.

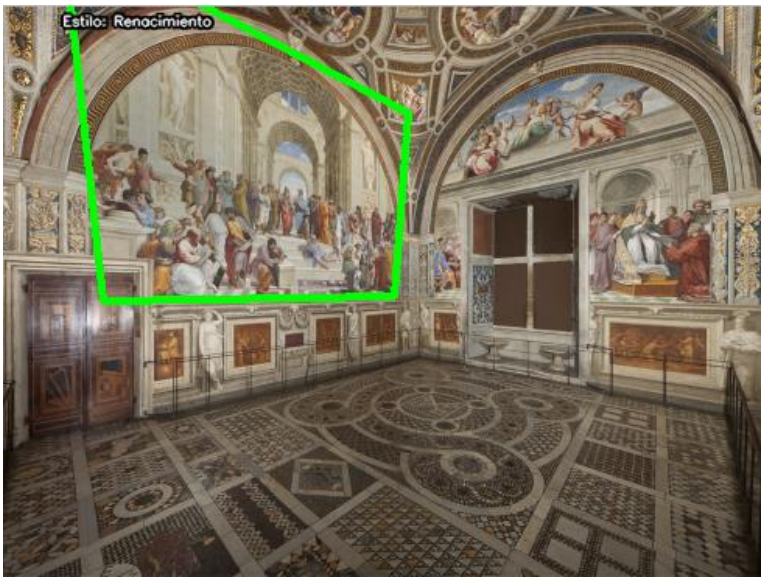

Figura 3. Escena 4 ORB con 4000 keypoints

3.1.1 Filtrado de matches

En cuanto al filtrado de matches, el método rva_filterMatches se utiliza para filtrarlos, basándose en la distancia entre los puntos clave.

Primero, verifica si hay coincidencias, y si no las hay, termina sin hacer nada. Luego, calcula la distancia mínima y máxima entre todas las coincidencias para determinar un umbral de filtrado. A continuación, recorre las coincidencias y solo mantiene aquellas cuya distancia sea menor o igual al umbral.

Al observar los resultados de la tabla 1 y fijándonos en la diferencia entre los matches y los matches filtrados, podemos hacer algunas observaciones interesantes:

- ORB:
    - El filtrado no elimina muchas coincidencias, ya que los matches filtrados suelen ser casi iguales a los matches originales, con pequeñas reducciones (por ejemplo, de 173 a 173 en la escena 2, o de 191 a 140 en la escena 8).
    - Esto sugiere que, con ORB, la mayoría de las coincidencias iniciales son bastante confiables y no necesitan un filtrado muy estricto.
- BRISK:

- Aquí, la diferencia entre matches y matches filtrados es mucho más grande. En algunos casos, el número de matches filtrados se reduce considerablemente (por ejemplo, de 1456 a 242 en la escena 2, o de 1890 a 313 en la escena 7).
- Esto indica que BRISK genera muchas más coincidencias que ORB, pero un porcentaje mayor de ellas es descartado durante el filtrado, probablemente porque no son tan precisas o confiables.

3.2 Imagen Plana

Se presentan ejemplos del funcionamiento del programa usando patch y añadiendo la imagen de ejemplo opencv3_model.jpg sobre las escenas.

Figura 4. Escena 7 con imagen añadida

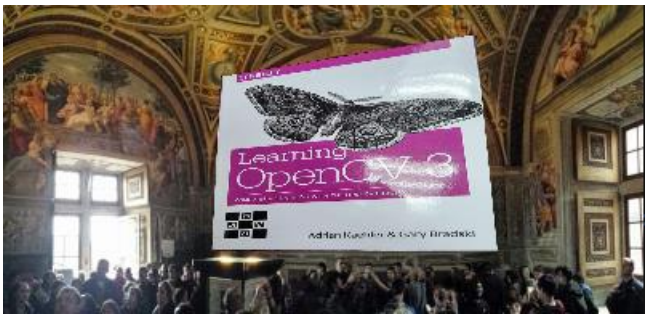

Figura 5. Escena 6 con imagen añadida

3.3 Vídeo

En este punto se muestra el funcionamiento del programa añadiendo un vídeo sobre las escenas.

Figura 6. Escena 7 con vídeo añadido

Figura 7. Escena 3 con vídeo añadido

Capítulo 4

CONCLUSIONES

Después de todo el proceso de desarrollo de las prácticas y estudio de la asignatura, puedo decir que este proyecto me ha permitido comprender en profundidad cómo funciona un sistema de Realidad Aumentada sin marcadores, y sobre todo, los retos técnicos que conlleva hacerlo de manera robusta y eficaz.

Desde el cálculo de homografías, concepto muy importante en este escenario, hasta la implementación del filtrado de matches y la superposición de contenidos, he podido aplicar los conocimientos vistos en clase y comprobar cómo cada bloque del sistema tiene su importancia.

También me ha resultado muy útil comparar los detectores ORB y BRISK y ver el funcionamiento de los diferentes métodos de OpenCV para diferentes escenas usando un mismo modelo.

En resumen, este proyecto me ha ayudado a afianzar conceptos clave de visión por computador y a aplicar conocimientos y visualizar sus resultados.

BIBLIOGRAFÍA

[1] Szeliski, R. (2022). Computer vision: Algorithms and applications (2nd ed.). The University of Washington. Capítulo 7.1.

[2] Marín-Jiménez, M. J. (2025). Introduction to Augmented Reality. Universidad de Córdoba. https://moodle.uco.es/m2425/pluginfile.php/156815/mod_resource/content/5/RA_Part1_mjmarin.pdf

[3] Marín-Jiménez, M. J. (2025). Keypoints: Classic to modern. Universidad de Córdoba. https://moodle.uco.es/m2425/pluginfile.php/156816/mod_resource/content/9/RA_Part2_keypoints_mjmarin.pdf

[4] Marín-Jiménez, M. J. (2025). Human-based AR: body pose estimation. Universidad de Córdoba. https://moodle.uco.es/m2425/pluginfile.php/156818/mod_resource/content/7/RA_Part3_humanpose_mjmarin.pdf

[5] OpenCV. (2022). Conceptos básicos de la homografía explicados con código. OpenCV. https://docs.opencv.org/4.11.0/d9/dab/tutorial_homography.html

[6] OpenCV. (2022). cv::Feature2D Class Reference. OpenCV. https://docs.opencv.org/4.x/d0/d13/classcv_1_1Feature2D.html

[7] OpenCV. (2018). cv::ORB Class Reference. OpenCV. https://docs.opencv.org/3.4/db/d95/classcv_1_1ORB.html

[8] OpenCV. (2025). cv::BRISK Class Reference. OpenCV. https://docs.opencv.org/4.x/de/dbf/classcv_1_1BRISK.html