

Detección y Emparejamiento de rasgos en imágenes.

J.F. Sánchez Castillo, Área Posgrado ITNM.

I. INTRODUCCIÓN.

En esta tarea se realizó una prueba a los principales métodos de detección y emparejamiento de rasgos, para hacerlo se tomaron como referencia 3 pares de imágenes, el primer par tiene una imagen rotada, el segundo tiene una imagen con escala reducida, y el tercer par tiene una imagen con solo una parte adyacente en común.

Este proyecto fue programado en Python 3 y se utilizó la librería de Opencv-contrib 3.3.1. El programa consta de una serie de módulos que son llamados de acuerdo a un menú de opciones que el usuario puede operar. El menú principal (main.py) contiene las siguientes etapas, de acuerdo al índice que le corresponde a cada opción se elegir ingresando su valor en línea de comandos.




Etapas1 Selección archivo	Etapas2 Selección detector	Etapas3 Selección emparejador	Etapas4 Selección Norma medición. (*BF)
n -Abre el buscador y permite seleccionar las imágenes deseadas.	1 -Good Features to Track	a -Brute Force	*(Hamming Predefinido)
d - Imágenes con rotación.	2 -FAST	b -Flann	*(Hamming Predefinido)
db - Imágenes con escalamiento.	3 -BRIEF		*(Hamming Predefinido)
dc - Imágenes con adyacencia.	4 -ORB		*(Hamming Predefinido)
	5 -AGAST		*(Hamming Predefinido)
	6 -AKAZE		*(Hamming Predefinido)
	7 -BRISK		*(Hamming Predefinido)
	8 -KAZE		1 -Norm_L1 2 -Norm_L2
	9 -SIFT		1 -Norm_L1 2 -Norm_L2
	10 -SURF		1 -Norm_L1 2 -Norm_L2

Figura1. Menú Opciones.

II. DETECCIÓN RASGOS.

La etapa de detección se encarga de detectar los rasgos característicos de cada imagen, para posteriormente utilizarlos en función de emparejamiento. Entonces cada detector necesita un descriptor para conocer la ubicación de cada rasgo detectado, sin embargo, algunos métodos de detección no cuentan con un descriptor, es por eso que se utiliza el descriptor BRIEF para obtener los datos, la siguiente tabla muestra la relación de detectores con descriptores.

Las funciones utilizadas en OpenCV son:

- 1- cv.<Detector>_create() : Inicia detectores por separado.
- 2- cv.xfeatures2d.<Detector>_create() : Inicia descriptores por separado..
- 3- cv.xfeatures2d.BriefDescriptorExtractor_create() : Inicia Descriptor Brief.
- 4- <Detector>.detect(img) : Detecta Rasgos sin calcular ubicaciones.
- 5- brief.compute(img,kpa) : Calcula Rasgos sin detectar rasgos previamente.
- 6- kp, des = <Detector>.detectAndCompute(img2,None) : Detecta y calcula rasgos cuando el detector y descriptor pertenecen al mismo método.
- 7- cv.drawKeypoints(img, kp) : Coloca los rasgos detectados en la imagen.

Detector	Descriptor
Good Features to Track	BRIEF
FAST	BRIEF
STAR	BRIEF
ORB	ORB
AGAST	BRIEF
AKAZE	AKAZE
BRISK	BRISK
KAZE	KAZE
SIFT	SIFT
SURF	SURF

Tabla2. Relación Detectores y Descriptores.

III. EMPAREJAMIENTO DE RASGOS

En la etapa previa de detección ya se generó el detector y el descriptor de rasgos, en esta etapa la función de emparejamiento toma ambos datos para identificar emparejamientos entre los rasgos de ambas imagenes. Los principales métodos de emparejamiento son Brute force y FLANN. Cada método de emparejamiento requiere especificar su parámetro de búsqueda o su norma para medición de distancia, en la siguiente tabla se especifica esta relación.

Tipo Descriptor	Descriptor	Norma Medición distancia	
		Brute Force	FLANN
Binario	BRIEF	Hamming	LSH
Binario	ORB	Hamming	LSH
Binario	AKAZE	Hamming	LSH
Binario	BRISK	Hamming	LSH
Punto Flotante	KAZE	Norm_L1 Norm_L2	KDTREE
Punto Flotante	SIFT	Norm_L1 Norm_L2	KDTREE
Punto Flotante	SURF	Norm_L1 Norm_L2	KDTREE

Tabla3. Relación del Emparejador con su norma de medida.

Las funciones utilizadas para iniciar el emparejador y especificar su norma de medición son las siguientes:

Emparejador Brute Force:

```
bf = cv.BFMatcher(cv.NORM_L1,crossCheck=True)
bf = cv.BFMatcher(cv.NORM_L2,crossCheck=True)
bf = cv.BFMatcher(cv.NORM_HAMMING,crossCheck=True)
```

Emparejador FLANN:

```
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
```

```
FLANN_INDEX_LSH = 6
index_params= dict(algorithm = FLANN_INDEX_LSH,
                  table_number = 6,
                  key_size = 12,
                  multi_probe_level = 1)
```

IV. RESULTADOS.

En cada método probado se realizó una evaluación relativa a la cantidad de emparejamientos y los errores que se pudieron observar, en la siguiente tabla se muestran las tablas comparativas en donde la máxima puntuación es 6 y la mínima 1.

Además, se muestran las imágenes de los dos métodos que obtuvieron mayor puntuación en cada categoría.

Brute Force		FLANN		
Detector	Evaluación	Detector	Radio búsqueda	Evaluación
SURF	6	SURF	0.1	6
SIFT	5	ORB	0.1	5
ORB	4	AKAZE	0.2	5
AKAZE	4	BRISK	0.3	5
BRISK	3	SIFT	0.3	5
KAZE	3	FAST	0.7	2
GFTT	2	KAZE	0.9	2
FAST	2	AGAST	0.7	2
BRIEF	2	GFTT	xxx	1
AGAST	2	BRIEF	xxx	1

Tabla 4. Comportamiento observado en imágenes con rotación.

Brute Force			FLANN		
Detector	Numero hallazgos	Evaluación	Detector	Radio búsqueda	Evaluación
SURF	163 (L1) 149 (L2)	5	BRISK	0.1	5
KAZE	200 (L1) 143 (L2)	5	KAZE	0.3	5
ORB	128	4	ORB	0.7	4
AKAZE	100	4	SIFT	0.2	4
BRISK	100	4	SURF	0.3	4
SIFT	178 (L1) 181 (L2)	4	AGAST	0.7	3
GFTT	10	3	GFTT	0.7	2
FAST	10	3	FAST	0.7	2
BRIEF	10	2	BRIEF	xx	1
AGAST	20	2	AKAZE	xx	1

Tabla 5. Comportamiento observado en imágenes con rotación.

Brute Force			FLANN		
Detector	Numero hallazgos	Evaluación	Detector	Radio búsqueda	Evaluación
FAST	1000	5	GFTT	0.6	5
AGAST	1000	5	BRISK	0.6	5
KAZE	948(L1) 965(L2)	5	FAST	0.5	4
SIFT	810 (L1) 800(L2)	5	AGAST	0.5	4
SURF	1000(L1) 1000(L2)	5	KAZE	0.5	4
GFTT	350	4	SIFT	0.4	4
ORB	128	4	SURF	0.5	4
AKAZE	500	4	BRIEF	0.6	3
BRISK	500	4	ORB	0.6	3
BRIEF	10	2	AKAZE	0.6	3

Tabla 6. Comportamiento observado en imágenes con adyacencia.

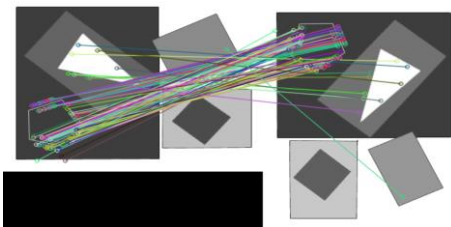
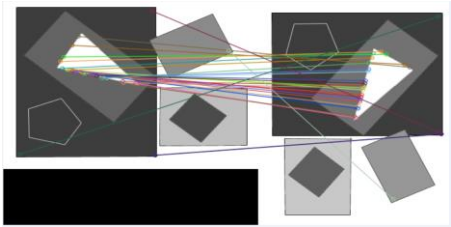
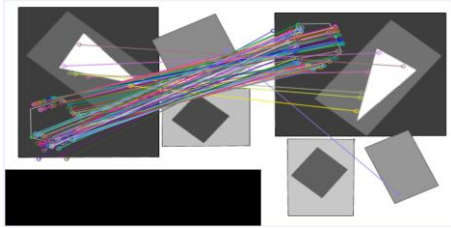
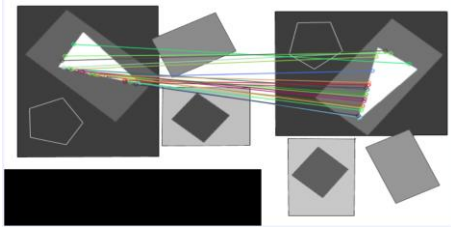
Emparejador	Imagen con rotación	
	SURF	SIFT
Brute Force BF Norma L1 (Sift, Surf, Kase)		
BF Norma L2		

Tabla 7. Métodos con mejores resultados en imágenes con rotación.

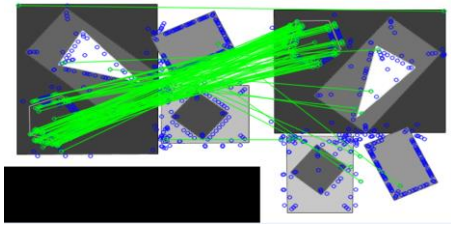
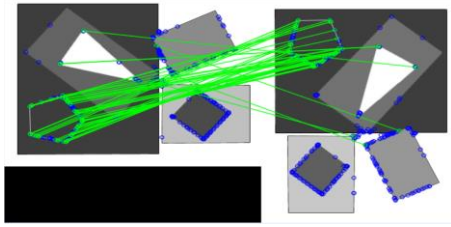
	Imagen con rotación	
Emparejador	SURF	ORB
FLANN		

Tabla 8. Métodos con mejores resultados en imágenes con rotación (FLANN).

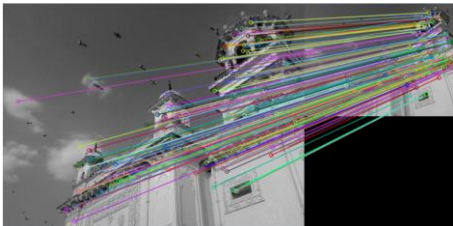



	Imagen a escala	
Emparejador	KAZE	SURF
Brute Force BF Norma L1 (Sift, Surf, Kase)		
BF Norma L2		

Tabla 9. Métodos con mejores resultados en imágenes a escala.

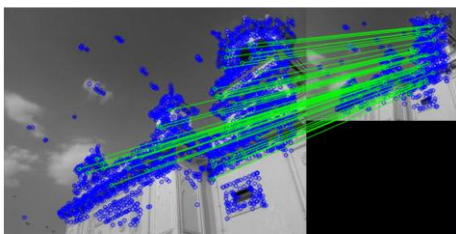
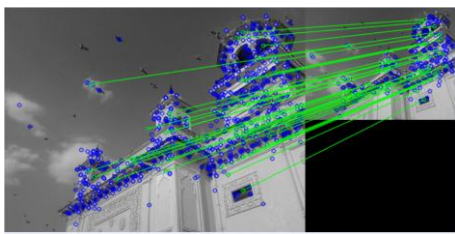
	Imagen a escala	
Emparejador	BRISK	KASE
FLANN		

Tabla 10. Métodos con mejores resultados en imágenes a escala (FLANN).




	Imagen con adyacencia	
Emparejador	AGAST	SURF
Brute Force BF Norma L1 (Sift, Surf, Kase)		
BF Norma L2		

Tabla 11. Métodos con mejores resultados en imágenes con adyacencia.


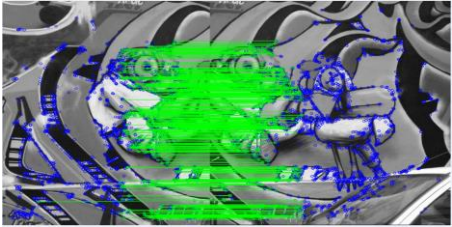
	Imagen con adyacencia	
Emparejador	GFTT	BRISK
FLANN		

Tabla 12. Métodos con mejores resultados en imágenes con adyacencia (FLANN).

V. CONCLUSIONES.

En este proyecto se pudieron evaluar los principales métodos para emparejamiento y detección de rasgos, se verificó la robustez de los métodos SURF, ORB y SIFT bajo condiciones de escalado y rotación, por otra parte se observó una debilidad de los métodos GFTT, FAST, BRIEF bajo estas mismas condiciones. En condiciones en donde las imágenes se encuentran con la misma escala, rotación y tienen una parte adyacente similar entre ellas GFTT, FAST, BRIEF presentan buenos resultados. Sin embargo, de forma general los métodos que mostraron mejores resultados fueron ORB y SURF. Si se desea ver con más detalles las imágenes resultantes puede dirigirse a la ruta del código fuente en la carpeta ./resultados.

VI. BIBLIOGRAFÍA.

- [1] OPENCV, Shi-Tomasi Corner Detector & Good Features to Track, https://docs.opencv.org/master/d4/d8c/tutorial_py_shi_tomasi.html, [online], Ultimo acceso: 20 dic. 2020.
- [2] OPENCV, FAST Algorithm for Corner Detection ,[www.opencv.org, https://docs.opencv.org/master/df/d0c/tutorial_py_fast.html](https://docs.opencv.org/master/df/d0c/tutorial_py_fast.html), [online], Ultimo acceso: 20 dic. 2020.
- [3] OPENCV, BRIEF (Binary Robust Independent Elementary Features), [www.opencv.org, https://docs.opencv.org/master/dc/d7d/tutorial_py_brief.html](https://docs.opencv.org/master/dc/d7d/tutorial_py_brief.html), [online], Ultimo acceso: 20 dic. 2020.
- [4] OPENCV, ORB (Oriented FAST and Rotated BRIEF), [www.opencv.org, https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html](https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html), [online], Ultimo acceso: 20 dic. 2020.
- [5] Mair, Elmar & Hager, Gregory & Burschka, Darius & Suppa, Michael & Hirzinger, Gerhard. (2010). Adaptive and Generic Corner Detection Based on the Accelerated Segment Test.
- [6] OPENCV, AKAZE local features matching ,[www.opencv.org, https://docs.opencv.org/3.4/db/d70/tutorial_akaze_matching.html](https://docs.opencv.org/3.4/db/d70/tutorial_akaze_matching.html), [online], Ultimo acceso: 20 dic. 2020.
- [7] Stefan Leutenegger, Margarita Chli, Roland Yves Siegwart. Brisk: Binary robust invariant scalable keypoints. In Computer Vision (ICCV), 2011 IEEE International Conference, pages 2548-2555.
- [8] Pablo Fernández Alcantarilla, Adrien Batolli, Andrew J Davison, Kaze features. In Computer Vision-ECCV 2012, pages 214-227. SPRINGER, 2012.
- [9] OPENCV, Introduction to SIFT (Scale-Invariant Feature Transform), [www.opencv.org, https://docs.opencv.org/trunk/da/df5/tutorial_py_sift_intro.html](https://docs.opencv.org/trunk/da/df5/tutorial_py_sift_intro.html), [online], Ultimo acceso: 20 dic. 2020.
- [10] OPENCV, Introduction to SURF (Speeded-Up Robust Features), [www.opencv.org, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html), [online], Ultimo acceso: 20 dic. 2020.
- [11] OPENCV, Feature Detection and Description, [www.opencv.org, https://docs.opencv.org/3.4/d5/d51/](https://docs.opencv.org/3.4/d5/d51/)

group__features2d__main.html, [online], Ultimo accesso: 20 dic. 2020.

[12] OPENCV, Feature Matching, www.opencv.org,
[https://docs.opencv.org/master/dc/dc3/
tutorial_py_matcher.html](https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html), [online], Ultimo accesso: 20 dic. 2020.