

# Visión por computador

## Práctica 4

Arón Collados  
Cristian Román



## Índice

- Introducción
- Calibración
- Emparejamiento de features
  - Ejemplos de emparejamiento
  - Ejemplos de detección de homografía
- Creación de un panorama
- Comparativas
- Conclusiones

## Introducción

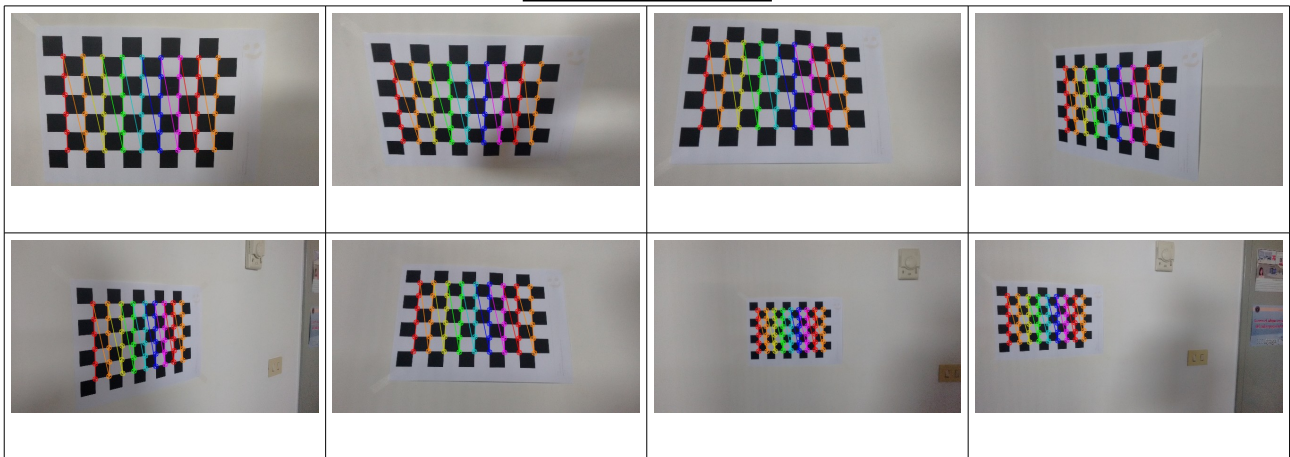
Se ha creado un programa capaz de crear panoramas a partir, tanto de fotografías, como de imágenes en vivo. Para ello primero se puede calibrar la cámara para solucionar la distorsión radial.

Para esta practica se han programado y analizado distintos métodos de extracción de puntos y descriptores (SIFT, SURF, ORB y FAST); y métodos de matching (FLANN y fuerza bruta).

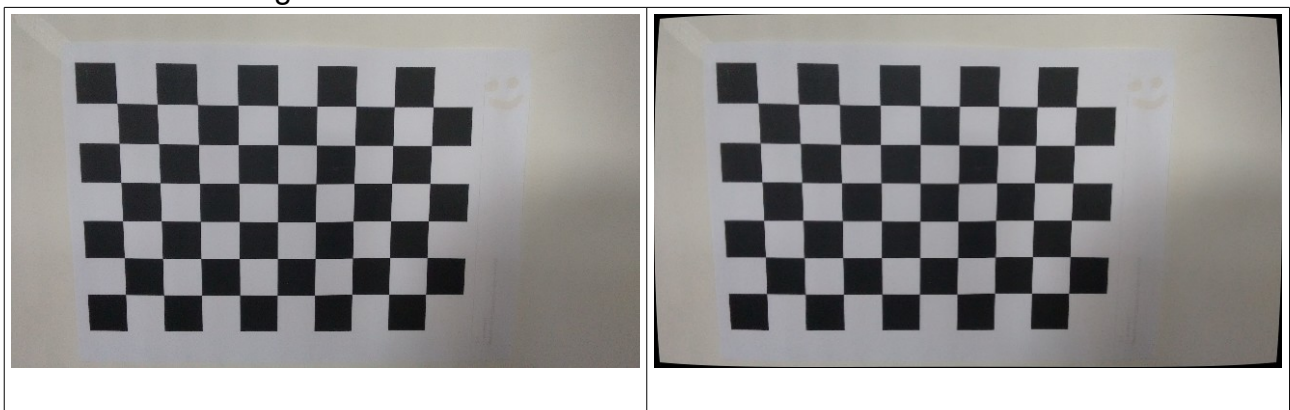
## Calibración

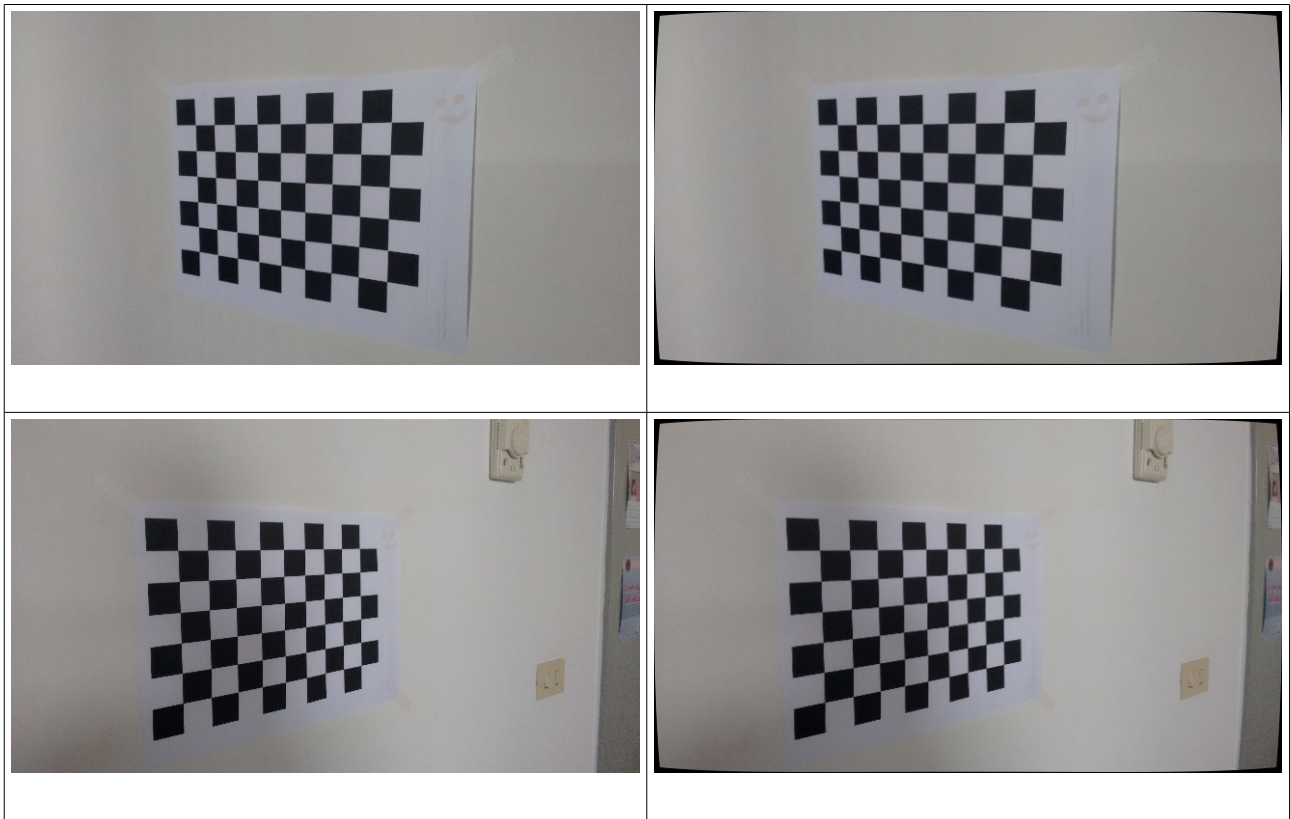
Se ha calibrado la cámara con la librería de calibración proporcionada por opencv, usando el método del tablero de ajedrez.

### Fotos de muestra



Original-----Resultado





## Detección y emparejamiento de features

### Imágenes Originales



Se transforman las imágenes a escala de grises

```
cvtColor(original, originalG, CV_BGR2GRAY);
cvtColor(next, nextG, CV_BGR2GRAY);
```

Detección y extracción de puntos de interés por varios métodos

## SIFT

```
SiftFeatureDetector detector;  
detector.detect(originalG, keypoints1);  
detector.detect(nextG, keypoints2);  
  
SiftDescriptorExtractor extractor;  
extractor.compute(originalG, keypoints1, descriptors1);  
extractor.compute(nextG, keypoints2, descriptors2);
```

## SURF

```
SurfFeatureDetector detector(400); //400 = minHessian  
detector.detect(originalG, keypoints1);  
detector.detect(nextG, keypoints2);  
  
SurfDescriptorExtractor extractor;  
extractor.compute(originalG, keypoints1, descriptors1);  
extractor.compute(nextG, keypoints2, descriptors2);
```

## ORB

```
OrbFeatureDetector detector;  
detector.detect(originalG, keypoints1);  
detector.detect(nextG, keypoints2);  
  
OrbDescriptorExtractor extractor;  
extractor.compute(originalG, keypoints1, descriptors1);  
extractor.compute(nextG, keypoints2, descriptors2);
```

## FAST

```
FastFeatureDetector detector;  
detector.detect(originalG, keypoints1);  
detector.detect(nextG, keypoints2);  
  
SurfDescriptorExtractor extractor;  
extractor.compute(originalG, keypoints1, descriptors1);  
extractor.compute(nextG, keypoints2, descriptors2);
```

\*Fast en opencv se recomienda usar con el extractor de puntos de surf

## Emparejamiento de puntos de interés

### BFM

```
BFMatcher matcher(NORM_L2);  
matcher.match(descriptors1, descriptors2, matches);
```

### FLANN

```
FlannBasedMatcher matcher;  
matcher.match(descriptors1, descriptors2, matches);
```

## Detección de la homografía

Detección de la homografía usando la función el algoritmo de RANSAC de opencv

```
Mat H = findHomography(obj, scene, CV_RANSAC);
```

## Aplicación de offset

Si la imagen con la que se esta comparando, esta a la izquierda o encima de la comparada, se aplicara una translación para que la imagen no se coloque en coordenadas negativas.

Para ello primero se detectara si es necesario, calculando la posición de las 4 esquinas de la imagen tras aplicarle la homografía. Si es el caso de que la posición mínima de estas sea negativa aplicaremos una translación a la homografía.

$H_{0,0}$	$H_{0,0}$	$H_{0,1}$		1	0	$T_x$
$H_{1,0}$	$H_{1,1}$	$H_{1,1}$	X	0	1	$T_y$
$H_{2,0}$	$H_{2,1}$	$H_{2,2}$		0	0	1

Donde  $T_x$  y  $T_y$  son los valores mínimos negativos obtenidos de las esquinas.

## Creación de panorama

```
Mat nueva(original.rows + next.rows +  $T_x$ , original.cols + next.cols +  $T_y$ , original.type(), Scalar(0, 0, 0));  
warpPerspective(original, nueva, H, nueva.size());  
Mat half(nueva, cv::Rect( $T_x$ ,  $T_y$ , next.cols, next.rows));  
next.copyTo(half);
```



## Resultados

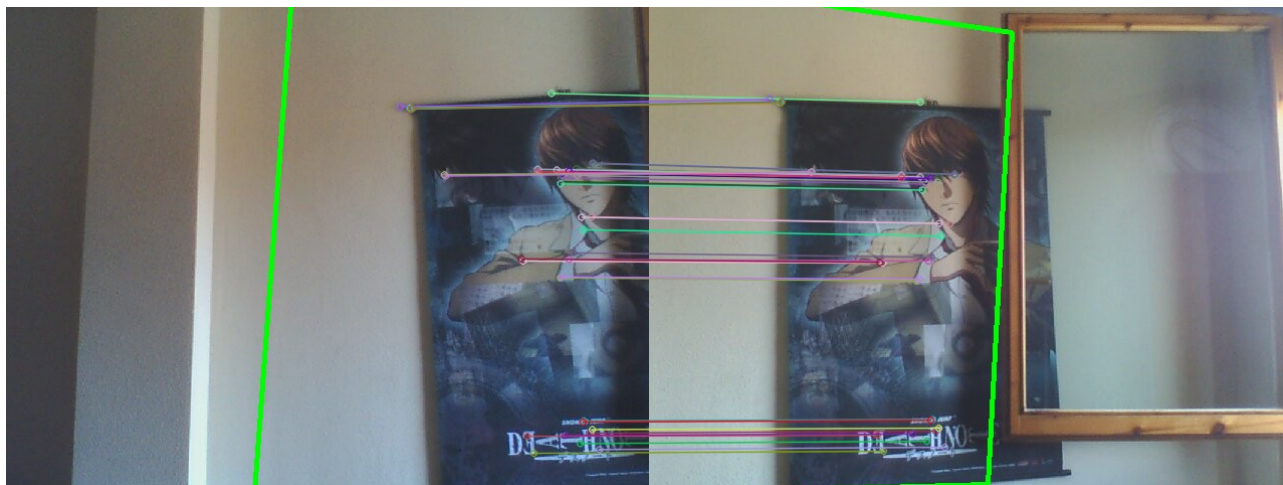
SIFT+ FLANN



SIFT + BFM



ORB + BFM



ORB + FLANN : No es posible aplicar el algoritmo debido a que ORB, no tiene puntos suficientes.



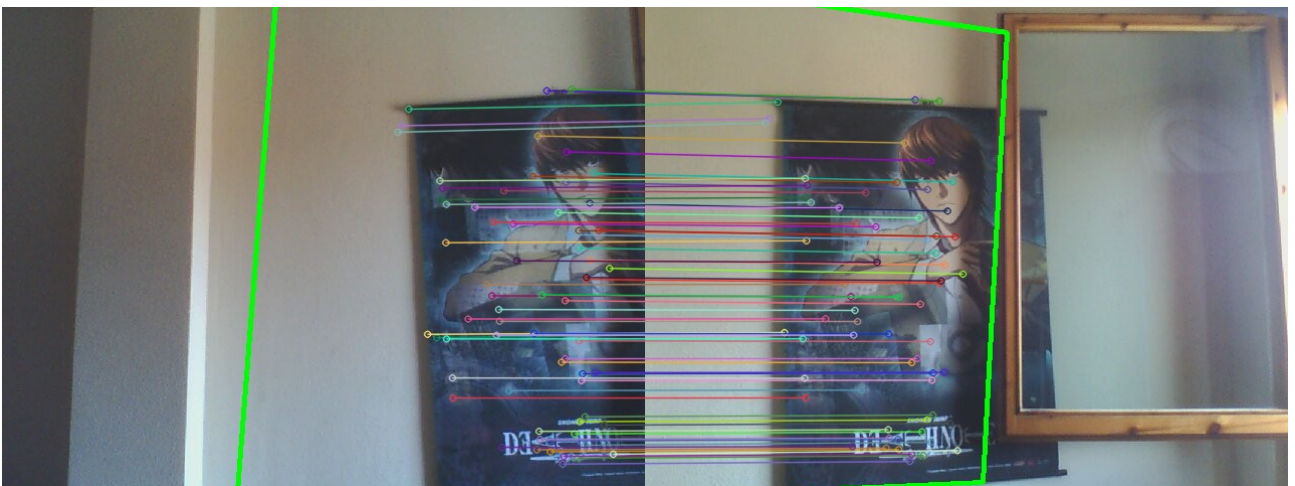
FAST + BFM



FAST + FLAN



SURF+BFM

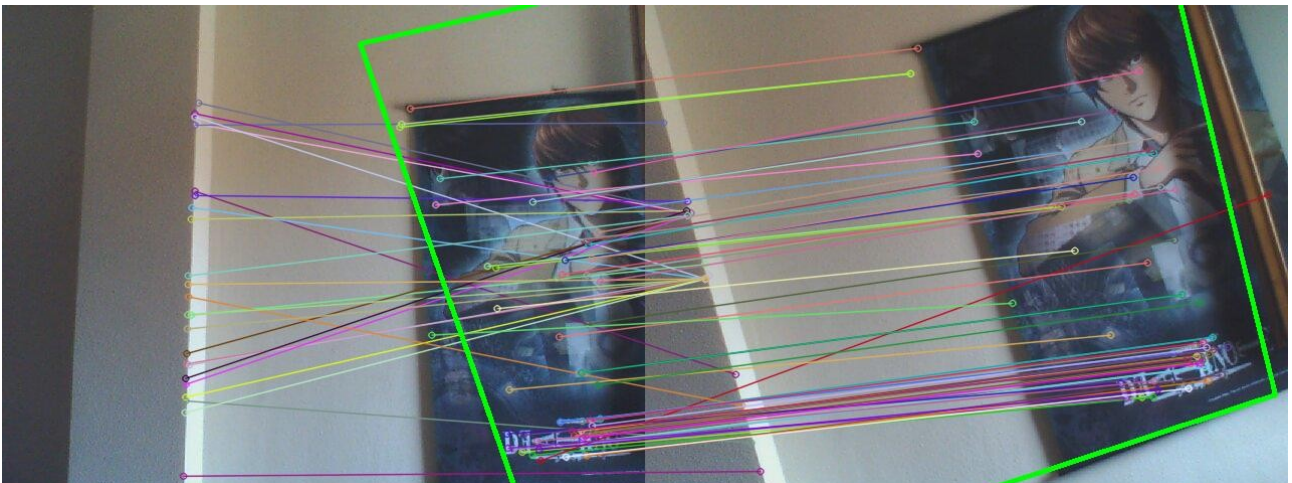
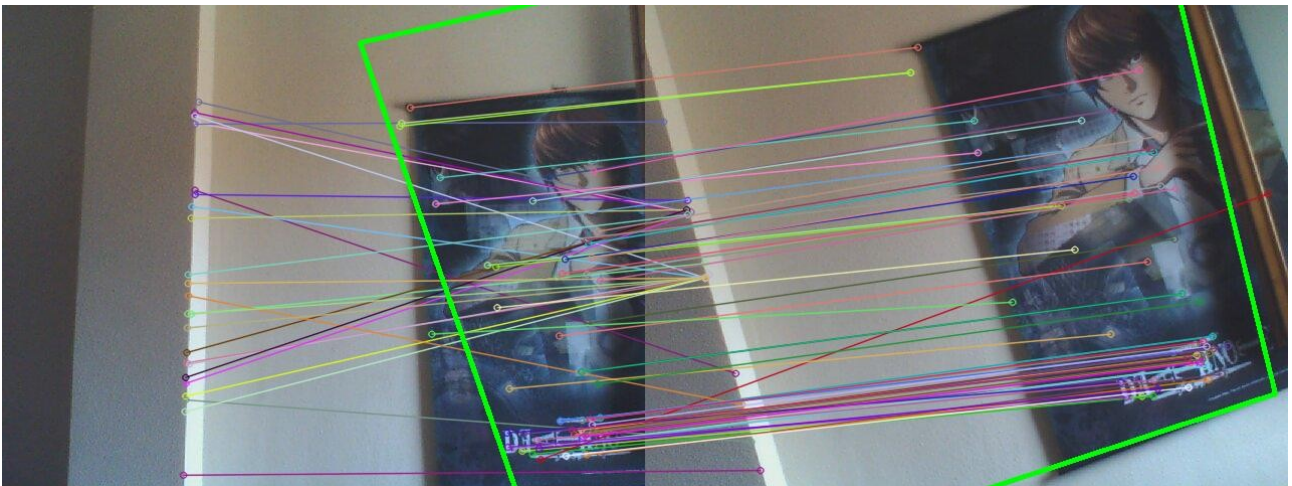




SURF+FLANN



Ejemplos de detección homografía



## Creación de un panorama

### 1ºPaso

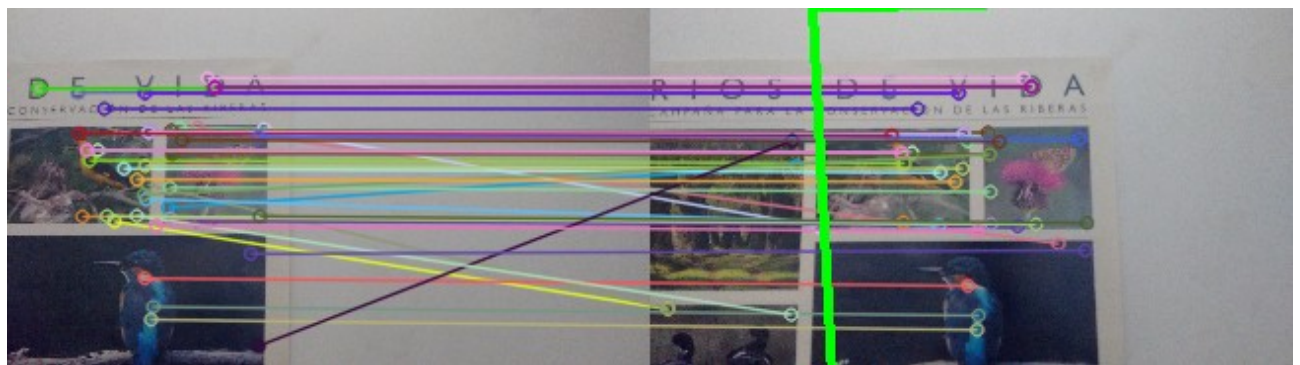


### 2ºPaso

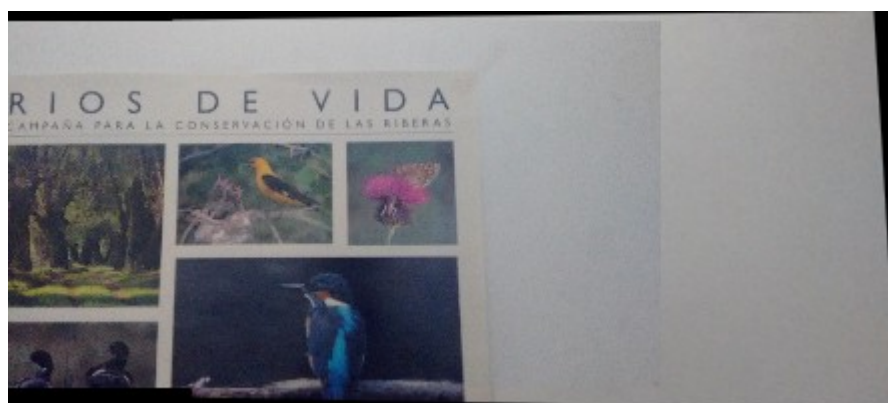
#### Nueva Imagen



#### Emparejamiento



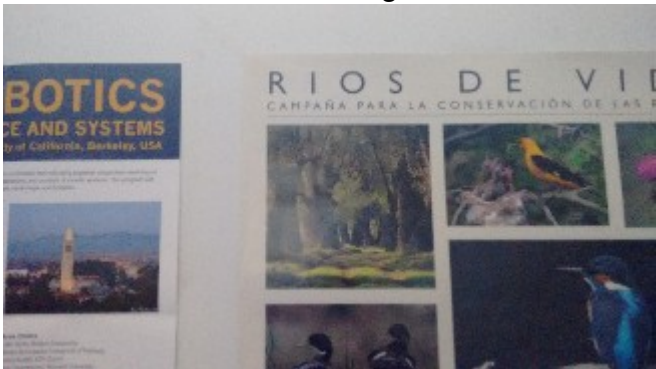
#### Resultado





3º Paso

Nueva Imagen



Emparejamiento

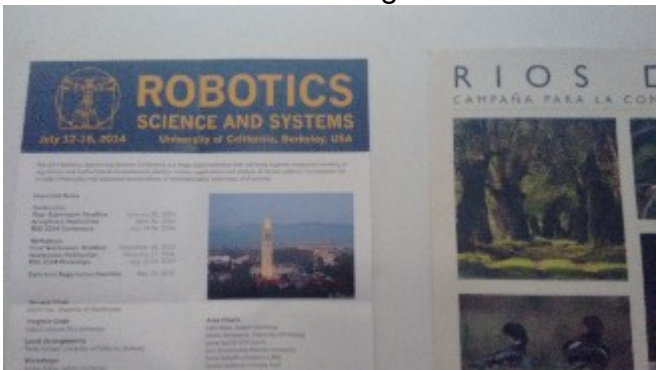


Resultado

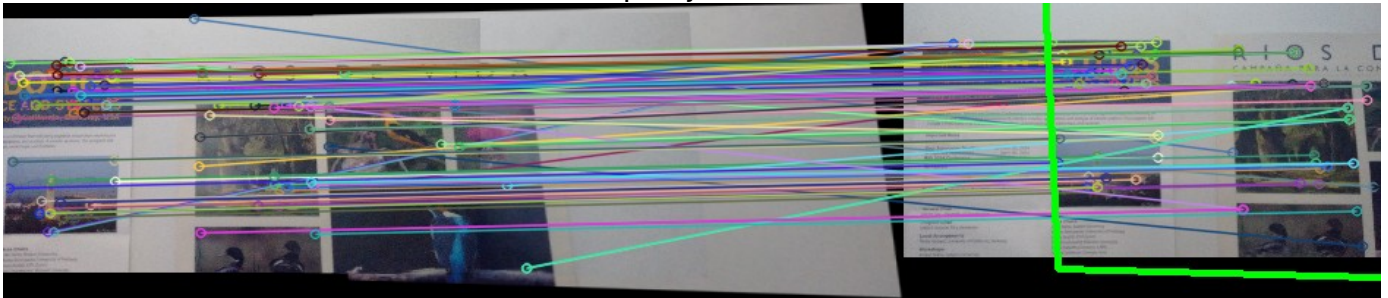


4º Paso

Nueva Imagen



## Emparejamiento



## Resultado



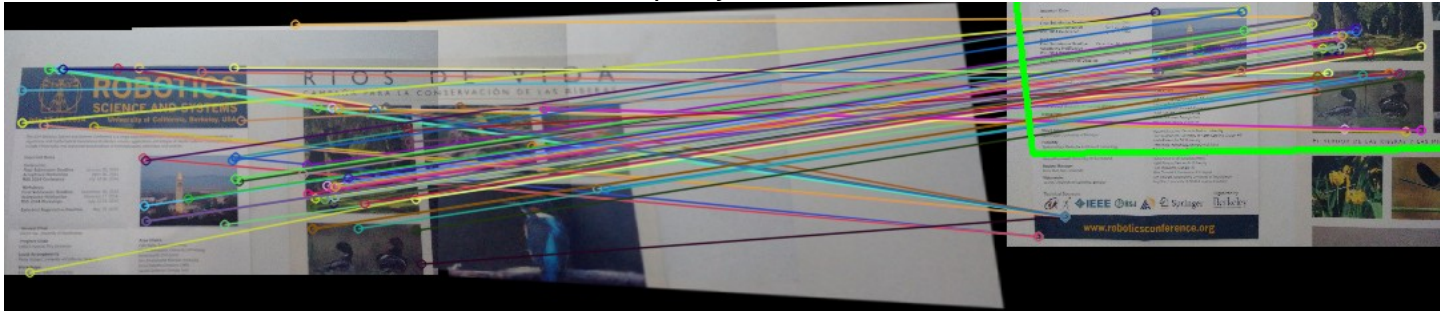
## 5º Paso

## Nueva Imagen

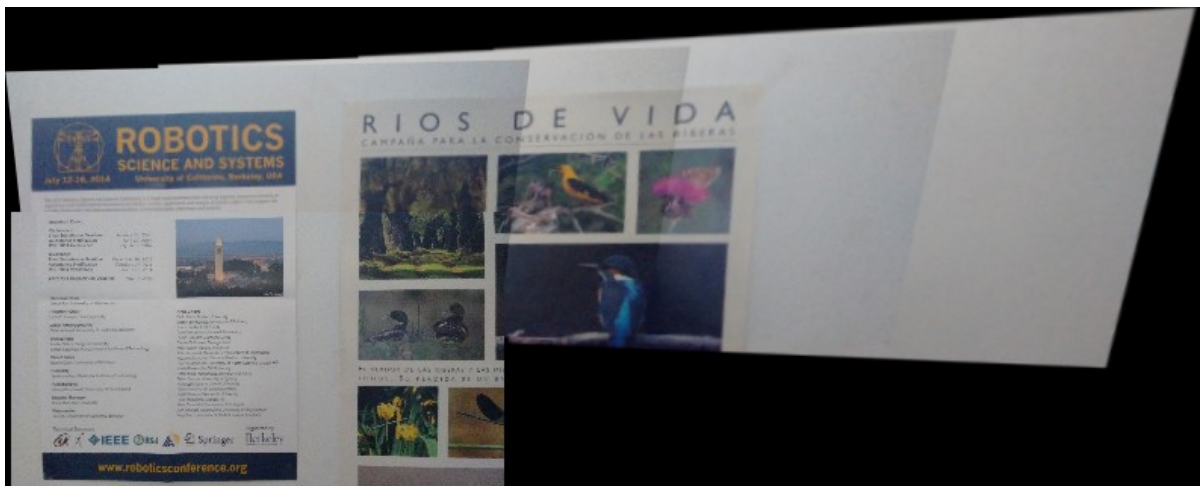




## Emparejamiento

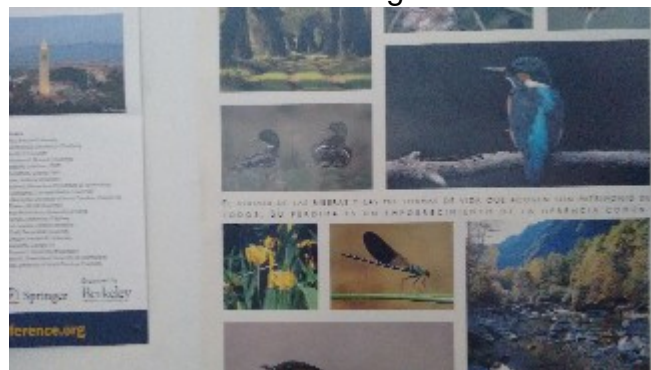


## Resultado

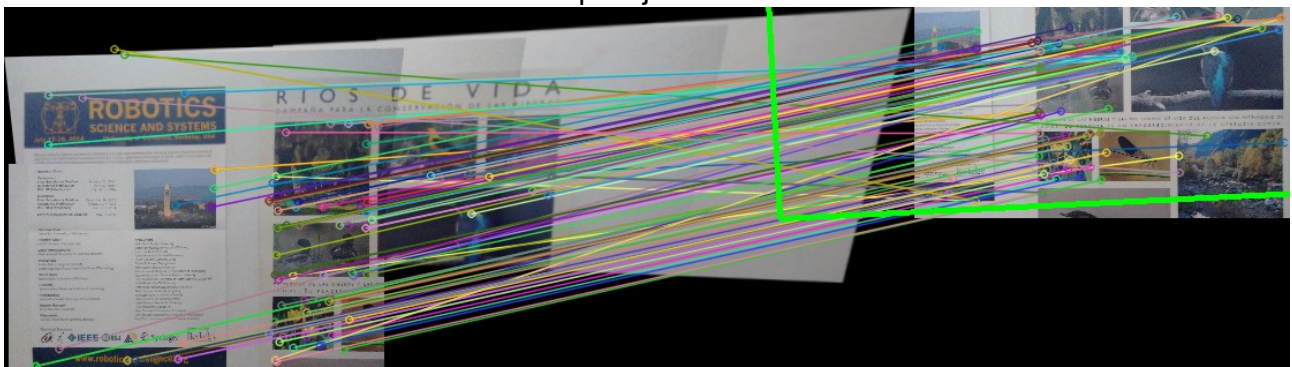


## 6º Paso

## Nueva Imagen



## Emparejamiento




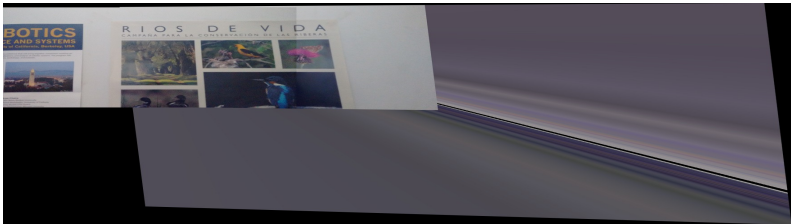




## Resultado



## Comparativas

Metodo	Tiempo/panorama	Resultado
SIFT	309077 ms	
SURF	266026 ms	
ORB	59572 ms	
FAST+ SURF	205559 ms	

\* El resultado FAST se obtiene sin descartar homografías atípicas

## **Conclusiones**

El método SIFT es mediante el cual se obtiene la mayor cantidad de puntos de interés con los descriptores más precisos, aunque a su vez es el método más lento. Por ello solo es recomendable usarlo con fotografías, o procesamiento que no sea necesario un procesamiento en vivo.

El mejor método para la extracción de puntos en vivo es SURF, ya que genera una gran cantidad de puntos de interés con descriptores lo suficientemente precisos para hacer matching correctamente y con un tiempo aceptable.

ORB genera pocos puntos, por lo que no es posible usar FLANN para su emparejamiento, y es necesario hacerlo por fuerza bruta. Descarta muchas imágenes por falta de puntos de interés, pero es el método más rápido de los probados.

FAST+SURF es el segundo método más rápido, obtiene un mayor numero de puntos, pero los descriptores obtenidos tienen una precisión menor, y tienden a formar homógrafas incorrectas