

Visión por computador

Trabajo 3



Ejercicios

1. Estimación de la matriz de una cámara a partir del conjunto de puntos en correspondencias.

- a) Generar la matriz de una cámara finita P a partir de valores aleatorios en $[0,1]$. Verificar si representa una cámara finita y en ese caso quedársela.

Para generar la matriz de una cámara finita P , vamos a generar 12 números de aleatorios que serán los que compongan la matriz P $3x4$. Una vez generada P , la vamos a descomponer como: $P = [M|l_m]$ siendo M $3x3$. De esta forma vamos a comprobar que el determinante de M sea distinto de 0. En ese caso, será una cámara finita válida.

- b) Suponer un patrón de puntos del mundo 3D compuesto por el conjunto de puntos $\{(0,x_1,x_2) \text{ y } (x_2,x_1,0)\}$, para $x_1=0.1:0.1:1$ y $x_2=0.1:0.1:1\}$

Vamos a tener una función para generar el patrón 3D. Para ello vamos a generar puntos como $p_1 = (0,x_1,x_2)$ y $p_2 = (x_2,x_1,0)$ siendo x_1 y x_2 valores entre 0,1 y 1, aumentando de 0,1 en 0,1. De esta forma crearemos 200 puntos.

- c) Proyectar el conjunto de puntos del mundo con la cámara simulada y obtener las coordenadas píxel de su proyección.

Para comenzar, vamos a transformar los puntos 3D generados en el punto anterior, en matrices $4x1$, siendo la nueva coordenada 1.

Seguidamente, vamos a generar los puntos proyectados, es decir multiplicaremos la matriz P $3x4$, por la matriz de punto $4x1$, dando como resultado el punto proyectado $3x1$.

Por último, para generar las coordenadas pixel, vamos a usar los puntos proyectados para transformarlos en coordenadas pixel. Para obtenerlas, la coordenada X e Y, la dividiremos entre la coordenada Z, obteniendo así las coordenadas pixel $2x1$.

- d) Implementar el algoritmo DLT para estimar la cámara P a partir de los puntos 3D y sus proyecciones en la imagen.

Para este algoritmo vamos a usar distintas ayudas como son:

- CalcularRotaciones, donde a partir de la matriz M , sacada de la P original, vamos a obtener K y R .
- Seguidamente vamos a obtener T multiplicando la inversa de K por m .
- Por último, vamos a obtener P a partir K , R y T .

- e) Calcular el error de la estimación usando la norma de Frobenius (cuadrática)

Vamos a obtener el error estimado usando la norma cuadrática, dando como resultado lo siguiente, siendo P la matriz P generada de forma aleatoria y PP la generada a partir del algoritmo DLT:

```
EJERCICIO 1
P =
[0.060565587, -0.60148162, -0.19788112, 0.62877017;
-0.12573405, -0.5024206, 0.54621011, 0.52418745;
-0.38441104, 0.40486339, -0.043105587, 0.58438003]

PP =
[0.060565561, -0.60148156, -0.19788112, 0.62877017;
-0.12573408, -0.50242054, 0.54621005, 0.52418739;
-0.38441098, 0.4048633, -0.043105572, 0.58438003]

El error minimo cuadratico es: 2.75474e-014
```

- f) Mostrar en una única imagen los puntos 3D proyectados con la cámara estimada y la cámara simulada.

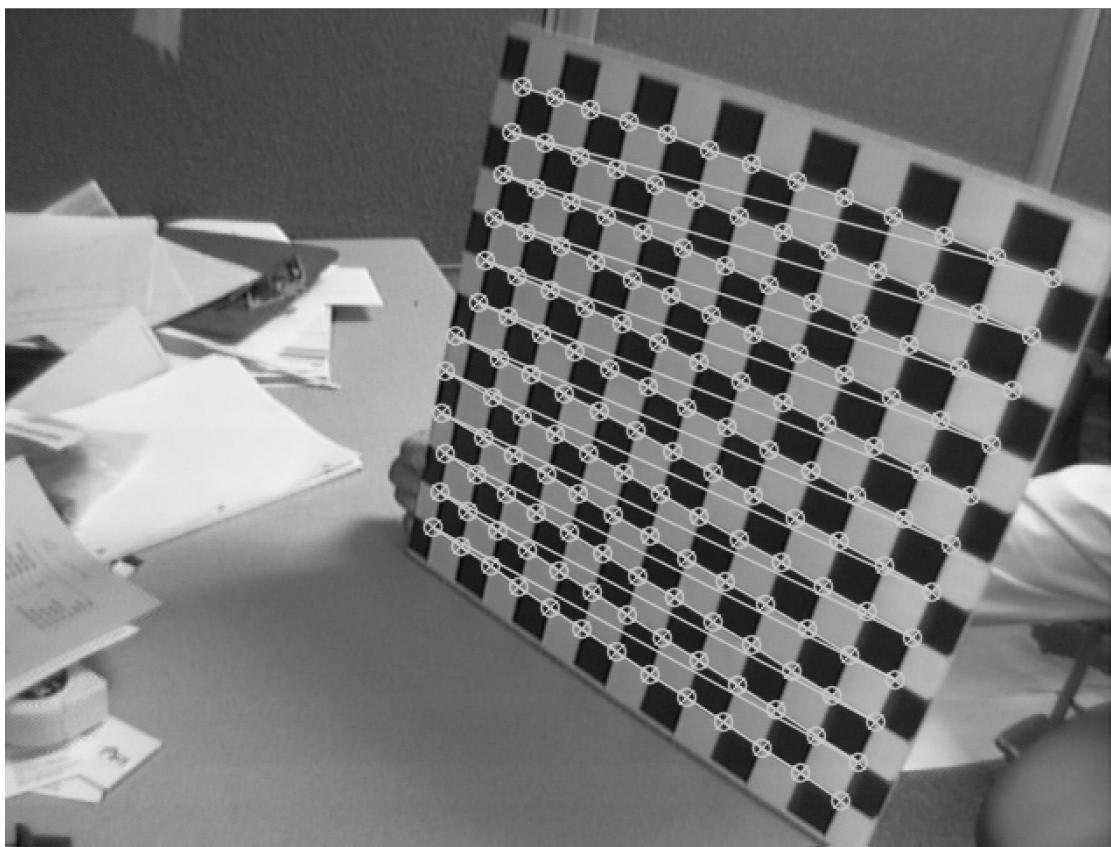
2. Calibración de la cámara usando homografías:

- a) Escribir una función que sea capaz de ir leyendo las sucesivas imágenes en chessboard.rar y determine cuáles son válidas para calibrar una cámara. Usar las 25 imágenes tiff que se incluyen en el fichero datos. Usar la función cv::findChessboardCorners(). Determinar valores precisos de las coordenadas de las esquinas presentes en las imágenes seleccionadas usando cv::cornerSubpix(). Pintar sobre la imagen los puntos estimados usando la función cv::drawChessboardCorners() (ver código de ayuda en la documentación de OpenCV).

Lo primero que vamos a realizar será leer todas las imágenes de chessboard y guardarlas en un vector. Seguidamente recorreremos el vector para determinar si una imagen es válida para calibrar una cámara.

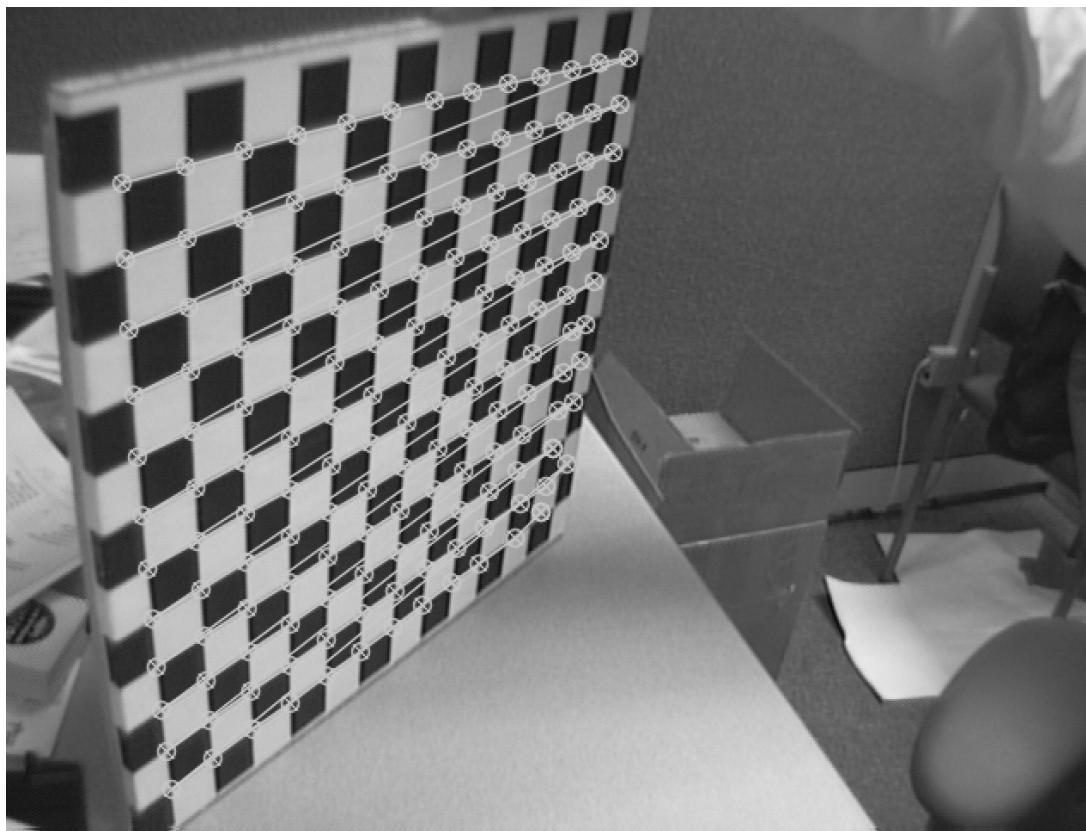
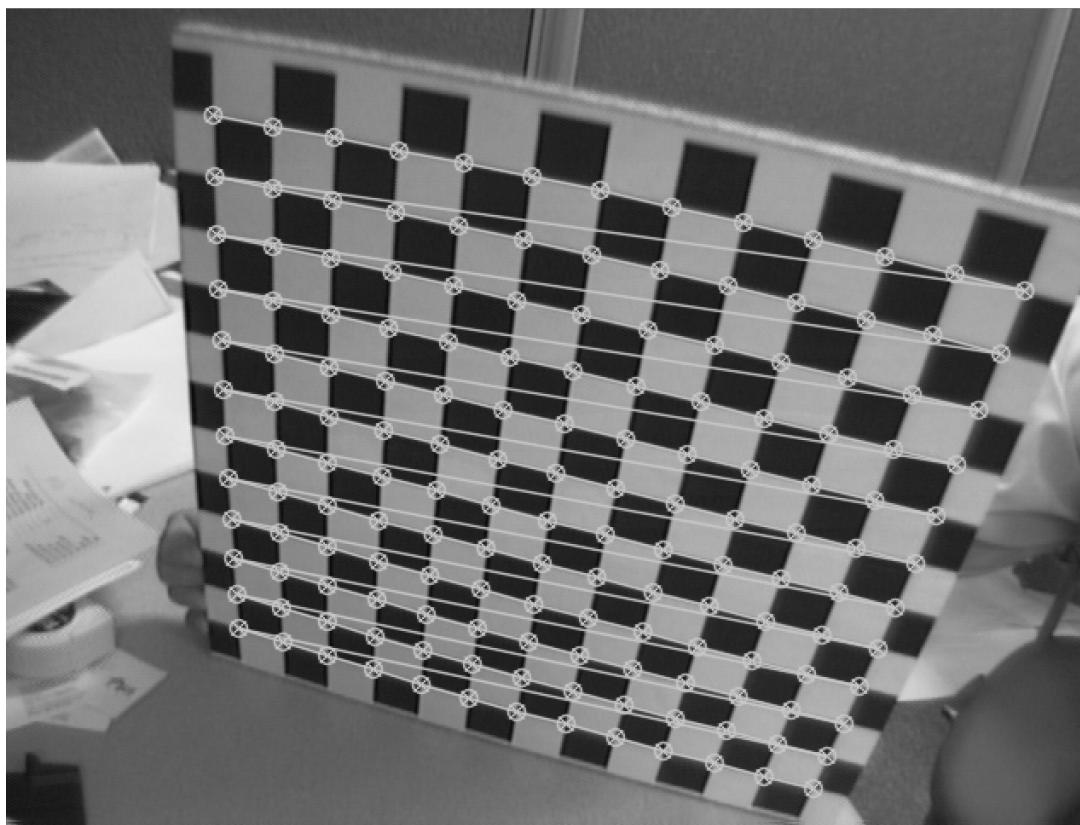
Para ello vamos a transformar primero la imagen. Seguidamente vamos a buscar los "corners" en la imagen con ayuda de la función "findChessboardCorners" de OpenCV. Si esta función nos devuelve true pasaremos a obtener los "corners" para seguidamente dibujarlos sobre la imagen. Como resultado obtendremos, que 4 de las 25 imágenes son válidas para calibrar la cámara. Estas imágenes con sus "corners" dibujados son:

```
EJERCICIO 2
La imagen numero 9 es valida para calibrar una camara.
La imagen numero 11 es valida para calibrar una camara.
La imagen numero 17 es valida para calibrar una camara.
La imagen numero 20 es valida para calibrar una camara.
```



20076629-K

Francisco Pérez Hernández



- b) Usando las coordenadas de los puntos extraídos en las imágenes seleccionadas del punto anterior, calcular los valores de los parámetros intrínsecos y extrínsecos de la cámara para cada una de dichas imágenes. Usar la función cv::calibrateCamera(). Suponer dos situaciones: a) sin distorsión óptica y b) con distorsión óptica. Valorar la influencia de la distorsión óptica en la calibración y la influencia de la distorsión radial frente a la distorsión tangencial.**

En este apartado, hemos usado las imágenes que son válidas para calibrar la cámara, y con ayuda de la función que nos aporta OpenCV, “calibrateCamera” hemos sacado los coeficientes, que serían para cada una de las imágenes, los que se muestran en la siguiente figura:

```
EJERCICIO 2
La imagen numero 9 es valida para calibrar una camara.

distCoeffs = [-0.6399658899104348, 2.926395477413574, -0.01949961003939519, 0.0190937759665582, -8.310353446107298]

La imagen numero 11 es valida para calibrar una camara.

distCoeffs = [-0.3618252595824593, 0.7681761664593191, -0.0007363111075011157, 0.004362491882521519, -2.629923561532904]

La imagen numero 17 es valida para calibrar una camara.

distCoeffs = [-0.2473768009418226, 0.001735223408095234, 0.001882086048260534, 0.003515975506244204, 0.3023541368639636]

La imagen numero 20 es valida para calibrar una camara.

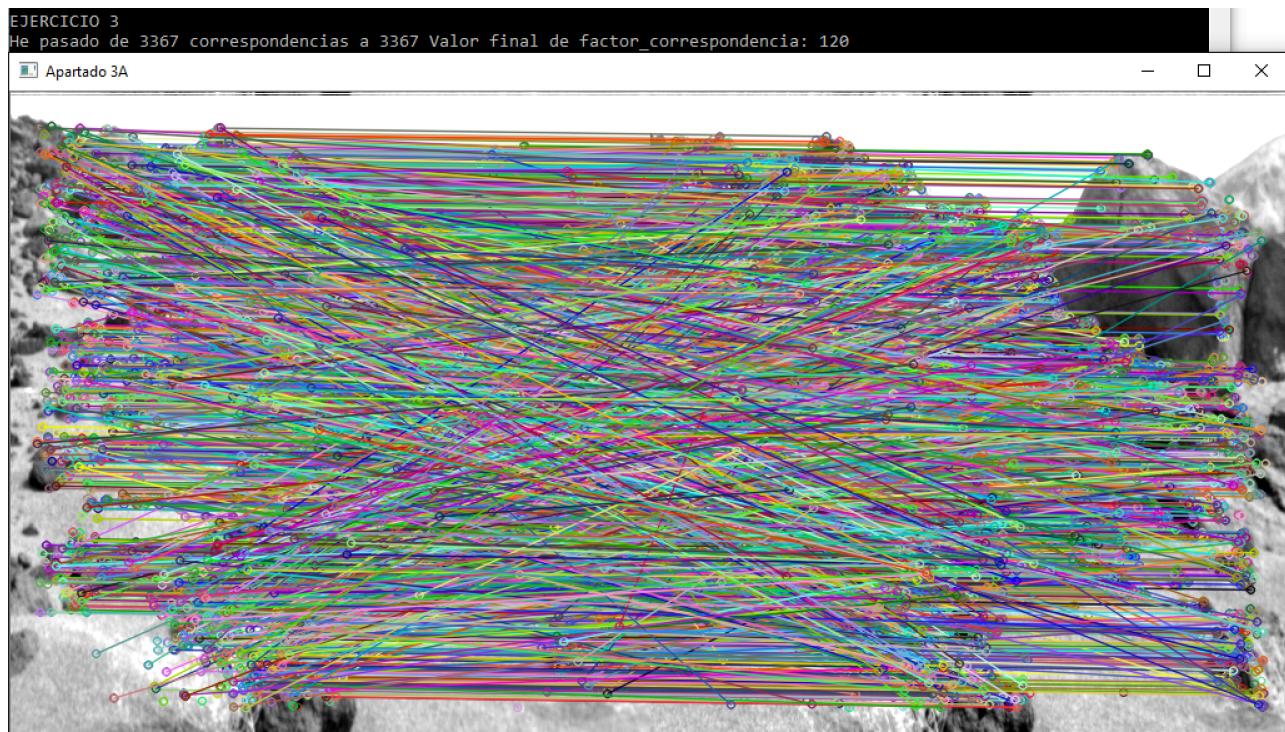
distCoeffs = [-0.1595889833669649, -0.3087889924356769, 0.001867393720066007, -0.01525838233565798, 0.4920198617872188]
```

3. Estimación de la matriz fundamental F:

- a) Obtener puntos en correspondencias sobre las imágenes Vmорт[*].pgm de forma automática usando las funciones de BRISK/ORB**

Para este apartado, vamos a hacer uso de la práctica anterior. De esta forma, voy a hacer uso del descriptor BRISK para sacar los keypoints y descriptores. Una vez obtenidos estos, llamaremos a la función de la práctica anterior, “PuntosEnCorrespondencia” para calcular las correspondencias buenas determinado un factor de correspondencia bueno. Por último, obtendremos una salida con la imagen resultante y sus correspondencias.

Como se ve en la imagen no se han eliminado correspondencias dado ese valor del factor de correspondencias que he visto más oportuno para este caso.

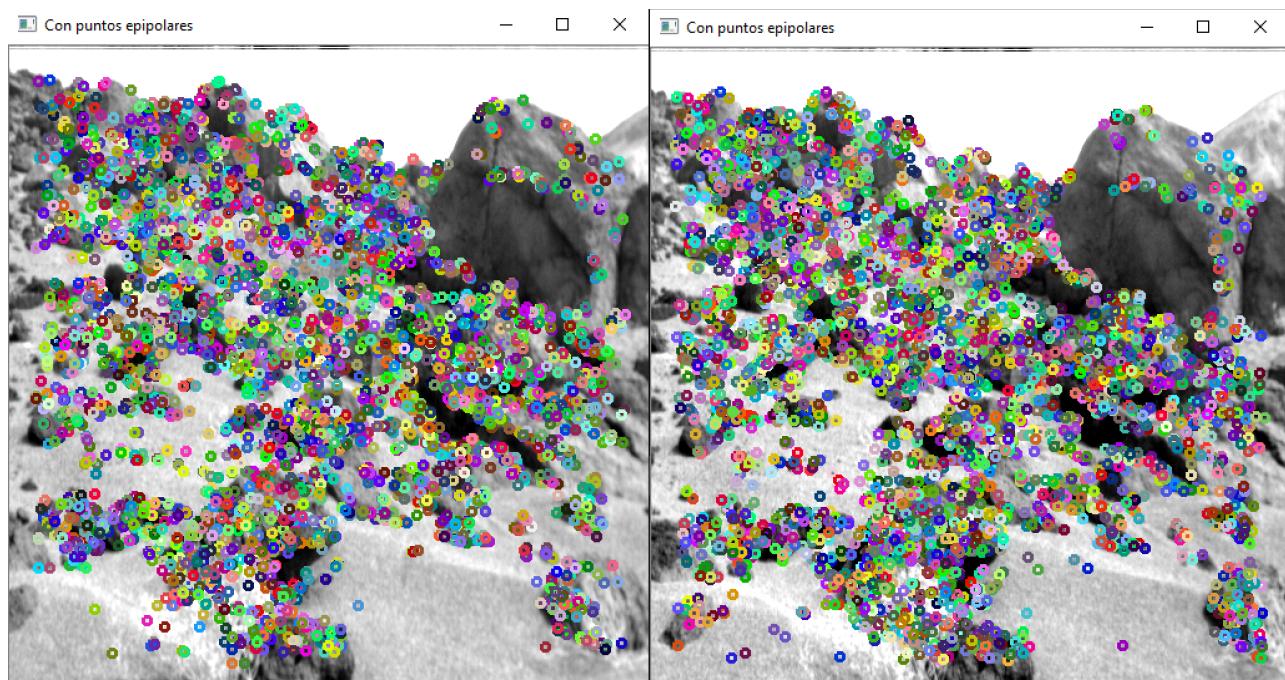


b) Calcular F por el algoritmo de los 8 puntos + RANSAC (usar un valor pequeño para el error de RANSAC)

Lo primero que vamos a hacer antes de obtener F, será transformar los keypoints obtenidos en el apartado anterior en puntos 2f. Una vez que los tenemos, vamos a dibujar esos puntos epipolares en las imágenes. Por último, y con ayuda de OpenCV y RANSAC vamos a buscar la matriz fundamental que en este caso es:

```
F = [-3.915414931569849e-008, -2.143657903775861e-006, 0.0005304517403714182;
6.303010628904167e-006, 4.218250210839995e-005, 0.01279863705498857;
-0.001626275235378727, -0.0241555123880032, 1]
```

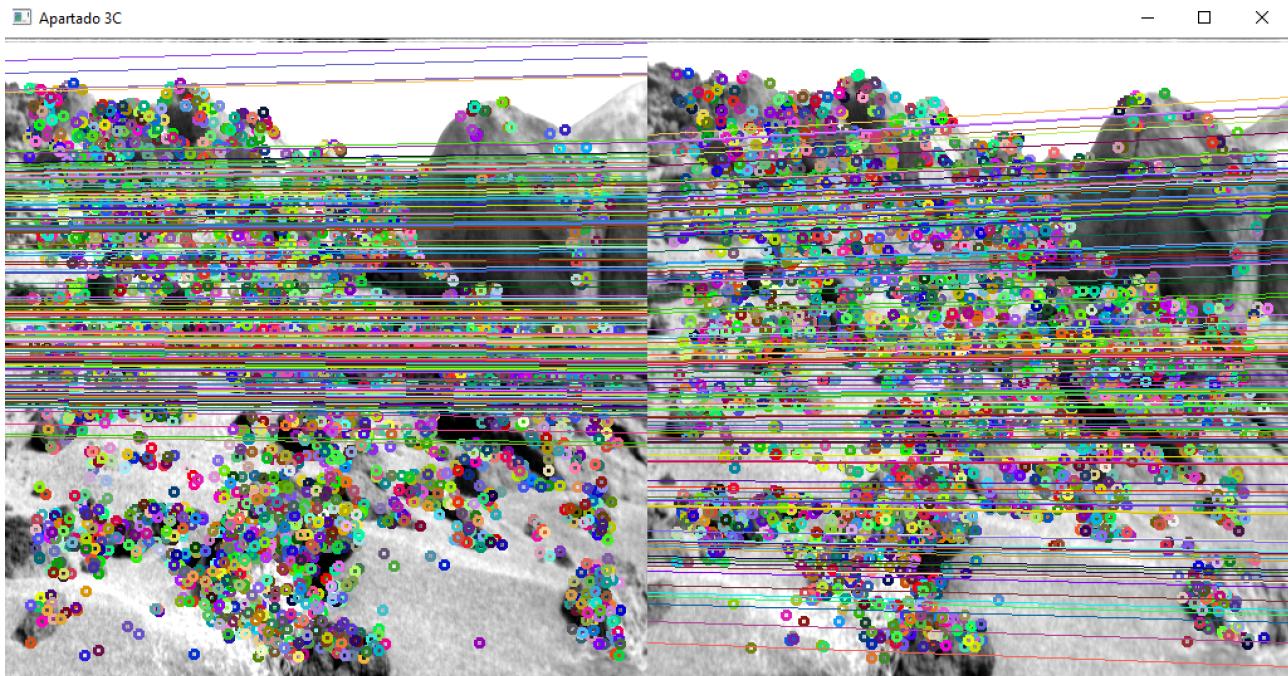
Además las imágenes con los puntos epipolares queda como:



c) Dibujar las líneas epipolares sobre ambas imágenes (< 200).

En este apartado vamos a dibujar las líneas epipolares, primero llamando a la función “computeCorrespondEpilines”, de OpenCV, y seguidamente a nuestra propia función “dibujarLineaEpipolar” que de todas las líneas propuestas por la función anterior, dibujará como máximo 200 de forma aleatoria.

Mi resultado ha sido el mostrado a continuación, el cual se puede apreciar que no es demasiado bueno:



d) Verificar la bondad de la F estimada calculando la media de la distancia ortogonal entre los puntos soporte y sus líneas epipolares en ambas imágenes. Mostrar el valor medio del error.

Para este apartado vamos a calcular el error medio, para lo que vamos a calcular la media de las distancias entre las lineas y los puntos. Como resultado he obtenido el que se ve en la siguiente imagen, siendo un resultado muy malo:

```
EJERCICIO 3
He pasado de 3367 correspondencias a 3367 Valor final de factor_correspondencia: 120
F = [-3.915414931569849e-008, -2.143657903775861e-006, 0.0005304517403714182;
 6.303010628904167e-006, 4.218250210839995e-005, 0.01279863705498857;
 -0.001626275235378727, -0.02415551238880032, 1]

La media distancia ortogonal es: 332.697
```

4. Calcular el movimiento de la cámara (R, t) asociado a cada pareja de imágenes calibradas.

a) Usar las imágenes y datos de calibración dados en el fichero reconstrucción.rar

Lo primero que se hace es leer las imágenes de reconstrucción, junto con los archivos .ppm.camera.

b) Calcular parejas de puntos en correspondencias entre las imágenes

En este apartado lo que voy a hacer es leer los ficheros .ppm.camera y sacaré los puntos en correspondencias de las imágenes seleccionadas. Seguidamente convertiré los keypoints al vector de puntos.

c) Estimar la matriz esencial y calcular el movimiento.

Para calcular la matriz esencial, primero buscaré la matriz fundamental mediante RANSAC. Una vez obtenida F, con las salidas del apartado anterior conseguiré la matriz Esencial como $E = K2(\text{Transpuesta}) * F * K1$.

Una vez tengo todas las matrices, calculo el movimiento.

Por último tendré como resultado las siguientes matrices:

```
EJERCICIO 4
He pasado de 3159 correspondencias a 31 Valor final de factor_correspondencia: 2
Matriz Esencial
E =
[-0.10902789, 0.93382478, -0.18690929;
 0.98092377, 0.48517132, -0.51765871;
 -0.26737416, -0.02786619, 0.11526921]

R_E =
[0, 0, 0;
 0, 0, 0;
 0, 0, 0]

T_E =
[0;
 0;
 0]
```

5. Reconstruir coordenadas 3D:

- a) Usando las imágenes del punto anterior reconstruir las coordenadas 3D, desde el sistema de referencia de la cámara izquierda, de parejas de puntos en correspondencias
- b) Estimar el mayor número posible de parejas de puntos a reconstruir
- c) señalar sobre las imágenes los puntos que ha sido capaz de encontrar
- d) aplicar el algoritmo lineal de reconstrucción.
- e) mostrar el mapa de profundidad de los puntos reconstruidos (usar una escala de gris sobre el rango de profundidad encontrado)

No he tenido tiempo suficiente para hacer el ejercicio.

Cuestionario

No he tenido tiempo para poder realizar el cuestionario.