Trabajos de Visión por Computador

TRABAJO-3

Detección de puntos relevantes y Construcción de panoramas

FECHA DE ENTREGA: 22 diciembre

Valoración total: 10 puntos

TRABAJO de IMPLEMENTACIÓN :

1.- (3 puntos) Detección de puntos Harris. Aplicar la detección de puntos Harris sobre una pirámide Gaussiana de la imagen, presentar dichos puntos sobre las imágenes haciendo uso de la función drawKeyPoints. Presentar los resultados con las imágenes Yosemite.rar. Para ello,

a) Detectar los puntos Harris en cada nivel de la pirámide a partir de la información de la función **`cornerEigenValsAndVecs().`** Por cada punto extraemos una estructura KeyPoint :(x,y, escala, orientación). Estimar la escala como blockSize*nivel_piramide y la orientación del parche como la orientación del gradiente en su punto central tras un alisamiento de la imagen con un sigma=4.5. (ver http://matthewalunbrown.com/papers/cvpr05.pdf)
b) Variar los valores de umbral de la función de detección de puntos hasta obtener un conjunto numeroso (> 2000) de puntos HARRIS en total que sea representativo a distintas escalas de la imagen. Justificar la elección de los parámetros en relación a la representatividad de los puntos obtenidos.
c) Identificar cuantos puntos se han detectado dentro de cada octava. Para ello mostrar el resultado dibujando los KeyPoints con drawKeyPoints. Valorar el resultado.
d) Calcular las coordenadas subpixel (x, y) de cada KeyPoint usando la función cornerSubPix de OpenCV y mostrar una imagen interpolada del entorno de las coordenadas (solo un entorno 10x10 a un zoom 5x) de una selección aleatoria de 3 de ellos en donde se marquen las coordenadas originales y las corregidas.

2.- (2 puntos)

Detectar y extraer los descriptores AKAZE de OpenCV, usando para ello detectAndCompute() . Establecer las correspondencias existentes entre cada dos imágenes usando el objeto BFMatcher de OpenCV y los criterios de correspondencias “BruteForce+crossCheck y “Lowe-Average-2NN”. Mostrar ambas imágenes en un mismo canvas y pintar líneas de diferentes colores entre las coordenadas de los puntos en correspondencias. Mostrar en cada caso un máximo de 100 elegidas aleatoriamente.

a) Valorar la calidad de los resultados obtenidos a partir de un par de ejemplos aleatorios de 100 correspondencias. Hacerlo en términos de las correspondencias válidas observadas por inspección ocular y las tendencias de las líneas dibujadas.

3.- (1.5 puntos) Escribir una función que genere un Mosaico de calidad a partir de N = 2 imágenes relacionadas por homografías, sus listas de keyPoints calculados de acuerdo al punto anterior y las correspondencias encontradas entre dichas listas. Estimar las homografías entre ellas usando la función cv2.findHomography(). Para el mosaico será necesario. a) definir una imagen en la que pintaremos el mosaico; b) definir la homografía que lleva cada una de las imágenes a la imagen del mosaico; c) usar la función cv2. warpPerspective() para trasladar cada imagen al mosaico (ayuda: mirar el flag BORDER_TRANSPARENT de warpPerspective para comenzar

4.- (3.5 puntos) Lo mismo que en el punto anterior pero usando todas las imágenes para el mosaico.

BONUS:

1.- Implementar de forma eficiente la estimación de una homografía usando RANSAC. Probarla con algún ejemplo (2 puntos)

En el fichero imagenes.rar se encuentran todas las imágenes citadas y varios grupos de imágenes con los que poder componer mosaicos en distintas proyecciones

No está permitido usar los recursos del módulo stitching de OpenCV.

Informe a presentar

Para este trabajo como para los demás proyectos debe presentar un informe escrito con sus valoraciones y decisiones adoptadas en cada uno de los apartados de la implementación. También deberá incluirse una valoración sobre la calidad de los resultados encontrados. (hacer en pdf o texto plano)

Normas de la entrega de Prácticas: EL INCUMPLMIENTO DE ESTAS NORMAS SIGNIFICA PERDIDA DIRECTA DE 1 PUNTO CADA VEZ QUE SE DETECTE UN INCUMPLIMIENTO.

1. El código se debe estructurar en funciones, una por cada apartado de la práctica.
2. El código debe estar obligatoriamente comentado explicando lo que realizan los distintos apartados y/o bloques.
3. Todos los ficheros juntos se podrán dentro de un fichero zip, cuyo nombre debe ser Apellido1_P[1-3].zip.
4. En C++ SOLO ENTREGAR EL CODIGO FUENTE. En phyton incluir el directorio “imágenes” con las imágenes usadas.
5. Los path que se usen en la lectura de imágenes o cualquier fichero de entrada debe ser siempre “imagenes/nombre_fichero”

6. Todos los resultados numéricos serán mostrados por pantalla. No escribir nada en el disco.
7. La práctica deberá poder ser ejecutada de principio a fín sin necesidad de ninguna selección de opciones. Para ellos fijar los parámetros por defecto que se consideren óptimos.
8. Solo poner puntos de parada para mostrar imágenes o datos por consola

Forma de entrega:. Subir el zip al Tablón docente de CCIA.