Prácticas de Visión por Computador

P3

Detección de puntos relevantes y Construcción de panoramas

FECHA DE ENTREGA: 30 diciembre

Valoración total: 8 puntos

TRABAJO de IMPLEMENTACIÓN :

1.- (3 puntos) Detección de puntos Harris. Este punto se centra en aplicar la detección de puntos Harris sobre las imágenes de Yosemite.rar (usar versiones en rango de gris) haciendo uso de la función drawKeyPoints() para su visualización. Para ello,

a) Construir una Piramide Gaussiana de tres niveles de cada imagen y detectar el valor del Criterio Harris.en cada pixel de cada uno de los niveles a partir de la información dada por la función **`cornerEigenValsAndVecs().`** Probar con tamaños de block-size de 3x3 a 7x7 y elegir.el valor de ksize=3 o 5.
b) Implementar la supresión de No-Máximos, en cada uno de los niveles. (búsqueda de máximos locales) (ayuda:hacer uso de una imagen auxiliar binaria que lleve cuenta de los puntos que quedan por reexplorar). Usar una distancia entre máximos en el rango 5-10 píxeles. Variar los valores de umbral mínimo de la función de detección de puntos hasta obtener un conjunto numeroso (> 2000) de puntos HARRIS en total que sea representativo de las distintas escalas de la imagen. Usar (70%-25%-5%) como proporción inicial de selección de cada una de las tres escalas Justificar la elección de los valores de los distintos parámetros usados en relación a la representatividad de los puntos obtenidos.
c) Por cada punto construir una estructura KeyPoint :(x,y, escala, orientación). Estimar la escala en píxeles como $2^{(\text{nivel\_piramide}-1)}$*blockSize y la orientación preferente de cada ventana 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)
d) Identificar cuantos puntos se han detectado dentro de cada octava. Para ello mostrar el resultado dibujando los KeyPoints con drawKeyPoints sobre la imagen original mostrando la escala de detección (valor del radio del circulo) y la orientación de cada detección (radio pintado). Valorar el resultado.
e) Calcular las coordenadas subpixel (x, y) de cada KeyPoint usando la función cornerSubPix de OpenCV y mostrar imágenes interpoladas a un zoom 10x de los entornos 9x9 de una selección aleatoria de 3 puntos, en donde se marquen las coordenadas originales y las corregidas. Fijar una región de tamaño 7x7 como máximo para la búsqueda del refinamiento en cada capa. Las coordenadas refinados

en las capas superiores admiten un último refinamiento en la imagen original en un entorno pequeño 5x5.

2.- (1.5 puntos)

Detectar y extraer los descriptores KAZE o AKAZE de OpenCV, usando para ello detectAndCompute() . Establecer las correspondencias existentes entre las dos imágenes del punto anterior 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 dadas tres imágenes relacionadas por homografías genere, a) sus listas de keyPoints calculados de acuerdo al punto anterior, b) las correspondencias encontradas entre dichas listas, c) estime la homografía a partir de dichas correspondencias y d) muestre un Mosaico bien registrado a partir de dichas imágenes. Estimar las homografías usando la función cv2.findHomography(). Para construir el mosaico será necesario a) definir la imagen en la que pintaremos el mosaico, b) definir la homografía que lleva cada una de las imágenes a la imagen del mosaico usando la función cv2.warpPerspective() con el flag BORDER_TRANSPARENT. Usar las imágenes del fichero mosaico.rar

4.- (2 puntos) Lo mismo que en el punto anterior pero usando todas las imágenes de mosaico.rar. Suponer un movimiento de cámara de derecha a izquierda.

REALIZACIÓN ALTERNATIVA: En caso de que se desee realizar toda la práctica con código propio es posible hacerlo, pero no podrá usar ninguna ayuda de OpenCV en ningún punto, salvo las funciones presentación de imágenes y dibujo de puntos. Si se obtiene más del 80% de los puntos obligatorios obtendrá también un bonus de 5*proporción obtenida. Incompatible con el resto de Bonus (Deberá de avisar al profesor si elige esta opción para fijar todos los puntos del desarrollo)

BONUS (máximo 2 bonus):

En general los puntos de Bonus solo se tendrán en cuenta si se ha alcanzado en la parte obligatoria el 75% o más de los puntos

B1.- Implementar de forma eficiente (tiempo real) el algoritmo RANSAC. para Homografías a partir de correspondencia de puntos Harris. Explicar el enfoque y probarlo con ejemplos de cálculo de homografías (1.5 puntos)

B2.- Implementar el punto.1 (parte obligatoria) con código propio. . Comparar con los resultados de OpenCV sobre las imágenes del punto.1 para los mismos valores de los parámetros de entrada. Mostrar imágenes con regiones. (1.5 puntos) (Obligatorio que el punto.1 este correctamente desarrollado, 75% o más)

B3.- Escribir una función que ejecute de forma eficiente (paralela) la interpolación de los valores de todos los píxeles de una imagen. Probarla comparando el resultado con el punto.4 (1.5 puntos) (Obligatorio que el punto.4 este correctamente desarrollado, 75% o más)

En PRADO se encuentran todas las imágenes citadas. Las imágenes del fichero CalibHarris.rar pueden usarse para verificar la implementación propia del algoritmo de Harris.

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 en cuaderno colab)

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..
4. SOLO ENTREGAR EL CODIGO FUENTE. NO incluir las “imágenes” dadas.
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. Poner puntos de parada para mostrar imágenes o datos por consola

Forma de entrega:. Subir el zip a PRADO.