Trabajos de Visión por Computador

TRABAJO-2

Detección de puntos relevantes y Construcción de panoramas

FECHA DE ENTREGA: 21 de noviembre

Valoración total: 10

TRABAJO de IMPLEMENTACIÓN :

1.- Detección de puntos Harris multiescala. Por cada región detectada necesitaremos guardar la siguiente información de cada punto: (coordenadas x , coordenada y, escala, orientación)). Usar para ello usar un vector de estructuras KeyPoint de OpenCV. Presentar los resultados con las imágenes Yosemite.rar.

a) Escribir una función que extraiga la lista potencial de puntos Harris a distintas escalas de una imagen de nivel de gris. Para ello construiremos una Piramide Gaussiana usando escalas definidas por sigma= 1,2,3,4,5. Sobre cada nivel de la pirámide usar la función OpenCV cornerEigenValsAndVecs para extraer la información de autovalores y autovectores de la matriz Harris en cada píxel (fijar valores de blockSize y ksize equivalentes al uso de máscaras gaussianas de $sigma_I$=1.5 y $sigma_D$=1 respectivamente). Usar uno de los criterios de selección estudiados a partir de los autovalores y crear una matriz con el valor del criterio selección asociado a cada píxel (para el criterio Harris usar k=0.04). Implementar la fase de supresión de valores no-máximos sobre dicha matriz (ver descripción al final). Ordenar de mayor-a-menor(ver sortIdx()) los puntos resultantes de acuerdo a su valor. Seleccionar al menos los 500 puntos de mayor valor. Mostrar el resultado dibujando sobre la imagen original un círculo centrado en cada punto y de radio proporcional al valor del sigma usado para su detección (ver circle()). (2 puntos)
b) Extraer los valores (cx,cy, escala) de cada uno de los puntos resultantes en el apartado anterior y refinar su posición espacial a nivel sub-píxel usando la función OpenCV **cornerSubPix()**) con la imagen del nivel de pirámide correspondiente. Actualizar los datos (cx,cy ,escala) de cada uno de los puntos encontrados. (0.5 puntos)
c) Calcular la orientación relevante de cada punto Harris usando el arco tangente del gradiente en cada punto. Previo a su cálculo se deben aplicar un alisamiento fuerte a las imágenes derivada-x e y derivada-y, en la escala correspondiente, como propone el paper MOPS de Brown&Szeliski&Winder. (Apartado 2.5).Añadir la información del ángulo al vector de información del punto. Pintar sobre la imagen original círculos con un segmento indicando la orientación estimada en cada punto (1 punto)

d) Usar el vector de keyPoint extraidos para calcular los descriptores SIFT asociados a cada punto (ver `cv2.DescriptorExtractor_create.compute ()`)

2.- Usar el detector- descriptor SIFT de OpenCV sobre las imágenes de Yosemite.rar (cv2.xfeatures2d.SIFT_create()). Extraer sus listas de keyPoints y descriptores asociados. Establecer las correspondencias existentes entre ellos usando el objeto BFMatcher de OpenCV**.** Valorar la calidad de los resultados obtenidos en términos de correspondencias válidas usando los criterios de correspondencias "BruteForce+crossCheck y "Lowe-Average-2NN". (2.0 puntos) (NOTA: Si se usan los resultados del puntos anterior en lugar del cálculo de SIFT de OpenCV la valoración es de 2.5 puntos)

3.- Escribir una función que genere un Mosaico de calidad a partir de N = 3 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(p1,p2,CV_RANSAC,1). 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). (2.5 puntos)

4.- Lo mismo que en el punto anterior pero para N > 5 (1.5 puntos)

Ayuda para la Supresión de No-máximos: Esta fase del algoritmo elimina como candidatos aquellos píxeles que teniendo un valor alto de criterio Harris no son máximos locales de su entorno para un tamaño de entorno prefijado (parámetro de entrada). En consecuencia solo estamos interesados en píxeles cuyo valor Harris represente un máximo local. La selección máximos locales puede implementarse de distintas maneras, pero una que es razonablemente eficiente es la siguiente: a) escribir una función que tomando como entrada los valores de una ventana nos diga si el valor del centro es máximo local; b) escribir una función que sobre una imagen binaria inicializada a 255, sea capaz de modificar a 0 todos los píxeles de un rectángulo dado; c) Fijar un tamaño de entorno/ventana y recorrer la matriz binaria ya creada preguntando, en cada posición de valor 255, si el valor del pixel correspondiente de la matriz Harris es máximo local o no; d) en caso negativo no hacer nada ; e) en caso afirmativo, poner a cero en la imagen binaria todos los píxeles de la ventana y copiar las coordenadas del punto central a la lista de salida junto con su escala (nivel de la pirámide o tamaño equivalente de ventana).

BONUS: (Usar las imágenes de Yosemite)

1.- Implementar de forma eficiente el detector propuesto en el paper de Brown&Szeliski&Winder (http://matthewalunbrown.com/papers/cvpr05.pdf) (1 punto)

2.- Implementar de forma eficiente el descriptor propuesto en el paper de Brown&Szeliski&Winder (2 puntos)

3.- Implementar de forma eficiente la estimación de una homografía usando RANSAC. (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
9. NO ESTA PERMITIDO EL USO DE CODIGO C/C++ que no sea absolutamente estándar. Por ejemplo no se permite dimensionar matrices a partir de variables o usar tipos de datos específicos de un compilador.

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