Práctica 2 - Visión por computador

Detección de puntos relevantes y construcción de panoramas

Fecha de entrega: 21 de noviembre

Información de la entrega de prácticas

- 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.
- La entrega se realiza a través del tablón docente de la plataforma DECSAI.
- **Valoración total: 10.5 puntos (+ 5.5 puntos de bonus)**

Informe a presentar

Tanto 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.

Normas de la entrega de prácticas

El inclumplimiento de estas normas significa la pérdida directa de un 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. Incluir el directorio `imagenes` 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 serán mostrados por pantalla. No escribir nada en el disco.

7. La práctica debe poder ejecutarse de principio a fin sin necesidad de ninguna selección de opciones. Para ello, se deben fijar los parámetros por defecto que se consideren óptimos.

8. Solo poner puntos de parada para mostrar imágenes o datos por consola.

Trabajo de implementación

1. (3 puntos) Detección de puntos SIFT y SURF. Aplicar la detección de puntos SIFT y SURF sobre las imágenes, representar dichos puntos sobre las imágenes haciendo uso de la función `drawKeyPoints`. Presentar los resultados con las imágenes `Yosemite.rar`.

   (a) Variar los valores de umbral de la función de detección de puntos hasta obtener un conjunto numeroso ($\geq$ 1000) de puntos SIFT y SURF que sea representativo de la imagen. Justificar la elección de los parámetros en relación a la representatividad de los puntos obtenidos.

   (b) Identificar cuántos puntos se han detectado dentro de cada octava. En el caso de SIFT, identificar también los puntos detectados en cada capa. Mostrar el resultado dibujando sobre la imagen original un círculo centrado en cada punto y de radio proporcional al valor de sigma usado para su detección (ver `circle()`) y pintar cada octava en un color.

   (c) Mostrar cómo con el vector de keyPoint extraídos se pueden calcular los descriptores SIFT y SURF asociados a cada punto usando OpenCV.

2. (2.5 puntos) 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 y los criterios de correspondencias "BruteForce+crossCheck y "Lowe-Average-2NN". (NOTA: Si se usan los resultados propios del puntos anterior en lugar del cálculo de SIFT de OpenCV se añaden 0.5 puntos)

   (a) 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 100 elegidas aleatoriamente.

(b) Valorar la calidad de los resultados obtenidos en términos de las correspondencias válidas observadas por inspección ocular y las tendencias de las líneas dibujadas.

(c) Comparar ambas técnicas de correspondencias en términos de la calidad de sus correspondencias (suponer 100 aleatorias e inspección visual).

3. (2.5 puntos) 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: Usar el flag `BORDER_TRANSPARENT` de `warpPerspective`).

4. (2.5 puntos) Lo mismo que en el punto anterior pero para $N > 5$ (usar las imágenes para mosaico).

Bonus

1. (1 punto) Implementar de forma eficiente el detector propuesto en el paper de Brown & Szeliski & Winder.

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

3. (2 puntos) Implementar de forma eficiente la estimación de una homografía usando RANSAC.