FICH, UNL - Departamento de Informática - Ingeniería Informática

Procesamiento Digital de Imágenes

Guía de Trabajos Prácticos 7

Nociones de segmentación

1 Objetivos

- Comparar en forma práctica las ventajas y desventajas de los diferentes operadores de detección de bordes y su desempeño en presencia de ruido.
- Mejorar la comprensión sobre el funcionamiento de la transformada de Hough y su utilidad práctica.
- Introducir conceptos básicos de las técnicas más usuales de segmentación basada en regiones.

2 Trabajos Prácticos

Antes de comenzar, le recomendamos estudiar el funcionamiento de las siguientes funciones de openCV (`https://docs.opencv.org/`), numpy (`https://numpy.org/`):

- `dst = cv.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])`
- `dst = cv.Sobel(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[, borderType]]]]])`
- `dst = cv.Laplacian(src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]]])`
- `edges = cv.Canny(src, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])`
- `lines = cv.HoughLines( src, rho, theta, threshold[, lines[, srn[, stn[, min_theta[, max_theta]]]]])`
- `lines = cv.HoughLinesP( src, rho, theta, threshold[, lines[, minLineLength[, maxLineGap]]])`

- `circles = cv.HoughCircles( src, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]])`
- `retval, labels = cv.connectedComponents( src[, labels[, connectivity[, ltype]]])`

Ejercicio 1: Detección de bordes

1. Implemente el detector de bordes de **Prewitt** en una función que retorne una imagen binaria con los bordes detectados.

2. Estudie detalladamente los parámetros de la implementación de **Sobel** de OpenCV. Realice un programa que aplique Sobel y le permita variar parámetros:
   - el tipo de dato del resultado (CV_8U; CV_64F)
   - derivadas en x e y (dx, dy)
   - el tamaño de la máscara (ksize)

   Cargue la imagen `patron_bordes.jpg` y obtenga los siguientes resultados:
   - perfiles de intensidad que le permitan intuir los resultados posteriores
   - los bordes en dirección x
   - los bordes en dirección y
   - los bordes en todos los sentidos
   - los bordes en todos los sentidos variando `ddepth`
   - los bordes en todos los sentidos variando el parámetro `ksize` (3, 5 y 7)
   - los bordes utilizando la apertura de Scharr (`ksize = FILTER_SCHARR (-1)`))

   Repita con una imagen real, analice los resultados y comparelos.

3. Estudie la implementación del detector de bordes **Laplacian** (2da. derivada) y analice su uso variando la talla del filtro.

4. Implemente un programa que le permita evaluar el comportamiento del detector de bordes de **Canny**, mientras varia sus parámetros de forma interactiva. Evalúe los resultados al cambiar el parámetro `L2gradient`.

5. Cargue la imagen `'mosquito.jpg'` y genere versiones con ruido impulsivo unimodal (variando el gris en que aparece) y con ruido gaussiano ($\mu = 0$) para distintos valores de desvío estándar.
   Aplique los distintos detectores de bordes a cada caso y compare los desempeños.

   Algunas preguntas de guía: ¿En qué zonas (debido a qué) funciona mejor y en cuales no?, ¿Qué sucede con el ruido?, ¿Con qué tipo de imágenes sacaría mejor provecho de los métodos?, ¿Qué tipo de preprocesamientos, de los que ya conoce, se le ocurren que serían útiles?, etc.

6. [*Opcional*]: implemente una función que le permita realizar la conexión local de bordes, a partir de las condiciones:

$$| \nabla f(x,y) - \nabla f(x_0,y_0) | \leq T_1$$
$$| \alpha(x,y) - \alpha(x_0,y_0) | < T_2$$

Ejercicio 2: Transformada de Hough (TH)

Antes de comenzar, recuerde ¿qué tipo de imágenes son apropiadas para utilizar con TH?, ¿qué preprocesos serían útiles?, ¿qué se obtiene (en el espacio transformado) al aplicar TH a un punto?, ¿qué particularidad presentan (en el espacio transformado) los puntos colineales?, ¿qué espera y qué no, como resultado del proceso de la TH?

1. Estudie los parámetros de la función `cv.HoughLines`, y los formatos en los que devuelve el resultado ($[\rho, \theta]$ o $[\rho, \theta$, votes]).
   Implemente un algoritmo que haga uso de esta función, debe permitir ajustar el rango de los ángulos en la búsqueda de puntos colineales y el umbral para el acumulador. *Consejo: use trackbars.*

2. Estudie la implementación de la TH probabilística `cv.HoughLinesP`, sus parámetros y el formato vectorial en el que devuelve los resultados.
   Implemente un algoritmo que haga uso de esta función, debe permitir ajustar los parámetros `(minLineLength,maxLineGap)` y el umbral para el acumulador.

   Utilice ambas implementaciones con las imágenes `'letras1.tif'`, `'letras2.tif'`, `'snowman.png'` y `'building.jpg'`.
   Explique sus diferencias, ventajas y desventajas. ¿Cuando utilizaría uno y cuando el otro?

3. [*Opcional*]: implemente un algoritmo de post-proceso, que permita unir pequeñas discontinuidades en puntos o segmentos colineales, a partir del resultado de la transformada de hough. Se debe permitir el ajuste del parámetro de tolerancia.

   ¿Se le ocurre alguna alternativa en la que se aproveche el tono de gris o el color de la imagen original?

Ejercicio 3: Segmentación mediante crecimiento de regiones

1. Algunas alternativas para hacer ésto:
   - Implemente el algoritmo siguiendo las indicaciones de la teoría, haciendo recursiva la función de crecimiento.
     Como propiedad a cumplir utilice un rango de grises alrededor del valor de gris de la semilla (`cv.inrange()`).
     Implemente un componente que le permita variar el rango de inclusión y vicualice el resultado utilizando pseudocolor.

- Estudie, instale y ponga en funcionamiento alguna función de crecimiento de regiones.
  Una alternativa puede ser: `https://simpleitk-prototype.readthedocs.io/en/latest/user_guide/segmentation/plot_region_growing.html`

2. Cargue imágenes médicas, del repositorio o de internet, y segmente algún área de interés utilizando el método de crecimiento de regiones.
   La semilla seleccionada por el usuario (puede hacerlo mediante un click o por teclado).
   Compare este resultado con la aplicación de la función de pseudocolor que implementó en la guía 4, para el ejercicio 2.

Ejercicio 4: Segmentación en color y etiquetado

El objetivo del ejercicio es poder identificar las rosas presentes en la imagen `'rosas.jpg'` para obtener información al respecto.

1. Utilice alguno de los métodos de segmentación color vistos previamente, y obtenga una máscara binaria con las rosas segmentadas.

2. Mejore la máscara, descartando información errónea (ruido) con el método que considere apropiado.
   Observación: es posible que su proceso pueda ser usado luego del etiquetado.

3. Identifique las diferentes regiones por el método de etiquetado de componentes conectadas.
   ¿Podría obtener el mismo resultado utilizando el algoritmo de crecimiento de regiones? ¿Cómo lo aplicaría?
   Modifique su implementación previa, pruébela y saque conclusiones.

4. Cuente automáticamente la cantidad de rosas presente en la imagen original.
   Sobre la imagen original, dibuje un círculo en el centro de cada rosa.

5. *Opcional*: investigue con que funciones de OpenCV puede obtener más información de las regiones, por ejemplo: tamaño de las regiones, tamaño promedio, cantidad de regiones bajo y sobre la media, tasa de circularidad, etc.
   Puede ver: `cv.connectedComponentsWithStats, cv.minEnclosingCircle(), cv.minAreaRect(), etc.`

Ejercicio 5: Aplicaciones

1. Realice un programa que le permita encontrar un objeto rectangular en una imagen, usted elija la imagen real y el objeto: una mesa, un teclado, un celular. Realice todos los procesamientos que considere necesarios.

   Ayuda: se restringe el uso a imágenes donde los bordes del objeto estén paralelos a los bordes de la imagen. Utilice versiones rotadas de la imagen (90,180 y 270 grados) para comprobar la robustez del método.

2. Estudie la implementación de la TH para círculos `cv.HoughCircles`.
   Utilizando la imagen `'latas.png'`, realice un programa que:
   - cuente e informe el número de latas,
   - que informe el número de latas discriminando en grandes y pequeñas,

   Realice los preprocesamientos que crea necesarios, y puede probar la robustez de su implementación rotando la imagen (180 grados).

3. Implemente un código para segmentar en forma automática la pista de aterrizaje principal en las imágenes de aeropuertos (`corrientes_ruidogris.jpg` e `iguazu_ruidogris.jpg`), las cuales poseen una combinación de ruido gaussiano e impulsivo. La salida del proceso debe ser la imagen restaurada con la pista principal coloreada (por ejemplo, con rectas rojas).

   Tenga en cuenta que el método debe ser útil para imágenes de otros aeropuertos, con características similares pero con variaciones en la localización, el largo de la pista, la inclinación de la pista, etc.

   Para probar la robustez de su código, se le sugiere que genere imágenes rotadas y/o desplazadas de las propuestas.