Practica1-2019

October 2, 2019

1 Procesamiento de Imágenes Digitales

Visión Computacional 2019-20 Practica 1. 3 de octubre de 2019
Autor1: Autor2:

1.1 Objetivos

Los objetivos de esta práctica son: * Programar algunas de las rutinas de transformaciones puntuales de procesamiento de imágenes y analizar el resultado de su aplicación. * Repasar algunos conceptos de filtrado de imágenes y programar algunas rutinas para suavizado y extracción de bordes. * Implementar un algoritmo de segmentación de imágenes y otro de extracción de líneas mediante la transformada de Hough.

1.2 Requerimientos

Para esta práctica es necesario disponer del siguiente software: * Python 2.7 ó 3.X, preferiblemente el segundo * Jupyter http://jupyter.org/. * Los paquetes `python-pip` y/o `python-pip3` y el paquete "PyMaxFlow" * Las librerías científicas de Python: NumPy (`python-numpy`), SciPy (`python-scipy`) y Matplotlib (`python-matplotlib`). * El paquete `python-pygame` * La librería OpenCV, que puedes instalar desde el paquete `python-opencv`.

Las versiones preferidas del entorno de trabajo puedes consultarlas en el Aula Virtual en el archivo "ConfiguracionPC2018.txt".

El material necesario para la práctica se puede descargar del Aula Virtual.

1.3 Condiciones

- La fecha límite de entrega será el martes 23 de octubre a las 23:55.
- La entrega consiste en dos archivos con el código, resultados y respuestas a los ejercicios:

1. Un "notebook" de Jupyter con el fuente y los resultados (ejecuta "Restart & Run all" antes de guardar)
2. Un documento "pdf" generado a partir del fuente de Jupyter, por ejemplo usando el comando `jupyter nbconvert --execute --to pdf notebook.ipynb`. Asegúrate de que el documento "pdf" contiene todos los resultados correctamente ejecutados (previamente ejecuta en el menú "Kernel" la opción "Restart & Run All").

- Las respuestas a los ejercicios debes introducirlas en tantas celdas de código o texto como creas necesarias, insertadas inmediatamente después de un enuciado y antes del siguiente.

- Las prácticas puede realizarse en parejas. Sólo es necesario que uno de los miembros del equipo entregue la práctica.

1.4 Instala el entorno de trabajo

1. Instala el entorno de trabajo.

   `apt install python apt install python-scipy apt install python-numpy apt install python-matplotlib apt install python-opencv apt install jupyter apt install jupyter-nbconvert`

   Para para trabajar con la versión 3.X de Python, basta sustituir la palabra "python" por "python3" en los comandos anteriores.

2. Instala el paquete PyMaxflow

`pip install PyMaxflow` o `pip3 install PyMaxflow`
Si no tienes el paquete "pip" debes instalarlo: `apt install python-pip` o `apt install python3-pip` 3. Instala el paquete "pygame"

``apt install python-pygame``

`En Python 3.X, la versión 18.04 de Ubuntu no tiene el paquete "python3-pygame" pero puedes inst`

1.5 Transformaciones puntuales

En este apartado te recomiendo que uses al menos la imagen indicada, que puedes encontrar en el directorio de imágenes del aula virtual. También puedes probar con otras que te parezcan interesantes.

Ejercicio 1. Carga la imagen `escilum.tif`. Calcula y muestra su histograma con la función `hist()` de `matplotlib.pyplot`. A la vista del histograma, discute qué problema tiene la imagen para analizar visualmente la región inferior izquierda.

Ejercicio 2. Escribe una función `eq_hist(histograma)` que calcule la función de transformación puntual que ecualiza el histograma. Aplica la función de transformación a la imagen anterior. Calcula y muestra nuevamente el histograma y la imagen resultantes. Discute los resultados obtenidos. £Cuál sería el resultado si volviésemos a ecualizar la imagen resultante?

1.6 Filtrado

Para realizar las convoluciones utiliza la función `convolve` o `convolve1d` de `scipy.ndimage`.

Carga y muestra las imágenes `escgaus.bmp` y `escimp5.bmp` que están contaminadas respectivamente con ruido de tipo gaussiano e impulsional. En los siguientes ejercicios también también puedes utilizar otras imágenes que te parezcan interesantes.

Ejercicio 3. Escribe una función `masc_gaus(sigma, n)` que construya una máscara de una dimensión de un filtro gaussiano de tamaño n y varianza . Filtra las imágenes anteriores con filtros gaussianos bidimensionales de diferentes tamaños de n, y/o .

En este ejercicio tenéis que implementar vosotros la función que construye la máscara. No podéis usar funciones que construyan la máscara o realicen el filtrado automáticamente.

Muestra cómo afecta este filtrado a los dos tipos de ruido que contaminan las imágenes anteriores y discute los resultados. Pinta alguna de las máscaras utilizadas.

Ejercicio 4. Escribe una función `masc_deriv_gaus(sigma, n)` que construya una máscara de una dimensión de un filtro derivada del gaussiano de tamaño n y varianza . Filtra la imagen `telefonica.jpg` con filtros bidimensionales de derivada del gaussiano para extraer los bordes de la imagen. Prueba con diferentes valores de n y/o .

Muestra y discute los resultados. Pinta alguna de las máscaras construidas.

Ejercicio 5. Utiliza la función `median_filter` del paquete `scipy.ndimage` que realice el filtrado de la imagen con un filtro de la mediana de tamaño n Œ n.

Muestra y discute los resultados para diferentes valores del parámetro n en ambas imágenes. Compáralos con los obtenidos en el Ejercicio 3.

Ejercicio 6. Utiliza la funcion `cv2.bilateralFilter()` de OpenCV para realizar el filtrado bilateral de una imagen. Selecciona los parámetros adecuados y aplícalo a las imágenes `tapiz.jpg`, `escgaus.bmp` y `escimp5.bmp` y otras que elijas tú.

Si llamamos σ_r a la varianza de de la gaussiana que controla la ponderación debida a la diferencia entre los valores de los píxeles y σ_s a la varianza de la gaussiana que controla la ponderación debida a la posición de los píxeles. Responde a la siguientes preguntas: * £Cómo se comporta el filtro bilateral cuando la varianza σ_r es muy alta? £En este caso qué ocurre si σ_s es alta o baja? * £Cómo se comporta si σ_r es muy baja? £En este caso cómo se comporta el filtro dependiendo si σ_s es alta o baja?

Muestra y discute los resultados para distintos valores de los parámetros, tanto para las imágenes contaminadas con ruido gaussiano como impulsivo. Compáralos con los obtenidos en los Ejercicios 3 y 5.

1.7 Transformada Hough

Ejercicio 7. Utiliza la función `cv2.HoughLines()` de OpenCV para encontrar líneas en la imagen `telefonica.jpg`. Para extraer los bordes de la imagen utiliza las funciones escritas más arriba.

Discute el funcionamiento para distintos valores de los parámetros de la función, así como de los filtros utilizados para extraer los bordes de la imagen. Pinta los resultados sobre la imagen (te proporcionamos algo de código por si fuese útil).

```
In [ ]: import cv2
        import matplotlib.pyplot as plt

        def draw_lines(img, lines, color=(0, 0, 255), thickness=2):
            """
            Draws a set of lines detected using the OpenCV Hough transform
            :param img: An input image in BGR format of type np.int8
            :param lines: List or Numpy array containing the parameters of the homogeneous lin
            :param color: The color used to draw the lines. Red by default.
            :param thickness: The thickness of the lines to be drawn
            """
            if lines is not None:
                for i in range(len(lines)):
                    eq = lines[i]
                    rho = -eq[2]
                    a = eq[0]
                    b = eq[1]
                    x0 = a * rho
```

```
            y0 = b * rho
            x1 = int(x0 - 1000 * b)
            y1 = int(y0 + 1000 * a)
            x2 = int(x0 + 1000 * b)
            y2 = int(y0 - 1000 * a)

            cv2.line(img, (x1, y1), (x2, y2), color, thickness)

    ##############################################################################

    lines = np.squeeze(cv2.HoughLines( ... ))
    # Convert the lines to homogeneous coordinates
    lines = np.array([np.cos(lines[:, 1]), np.sin(lines[:, 1]), -lines[:, 0]]).T

    # Draw andshow the lines
    draw_lines(img, lines)
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

1.8 Segmentación

Ejercicio 10. Escribe una función que segmente el objeto central de una imagen a partir de una segmentación manual inicial realizada por el usuario. Puedes utilizar el código proporcionado en el archivo `segm.py`. En la optimización 1. toma como afinidad entre una pareja de píxeles la diferencia en sus valores de color y; 2. sólo establece los términos unitarios de los píxeles marcados por el usuario.

Aplícalo al menos a las imágenes `persona.png` y `horse.jpg`. Muestra y discute los resultados.

Ejercicio 11. Mejora el algoritmo anterior. Sugerencia: * Refina la segmentación iterativamente. * Mejora la función de afinidad entre píxeles. * Mejora los términos unitarios

mejora los resultados de algunas de las imágenes anteriores. Muestra y discute los resultados.