UNIVERSIDAD POLITÉCNICA DE MADRID

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, AUTOMÁTICA Y FÍSICA APLICADA

Prácticas de Visión Artificial

Práctica 5

Prácticas de preprocesado de las imágenes

5 Técnicas de pre-procesado de las imágenes

En esta práctica se tratará de experimentar con las técnicas clásicas del procesado de imágenes empleando *"Image Procesing Toolbox"* de Matlab. Se revisarán tanto las que derivan de Procesamiento Digital de Señales como aquellas de carácter más heurístico. En primer lugar se analizarán las técnicas de eliminación del ruido y posteriormente las dedicadas a la detección de bordes.

5.1 Técnicas de eliminación del ruido y suavizado

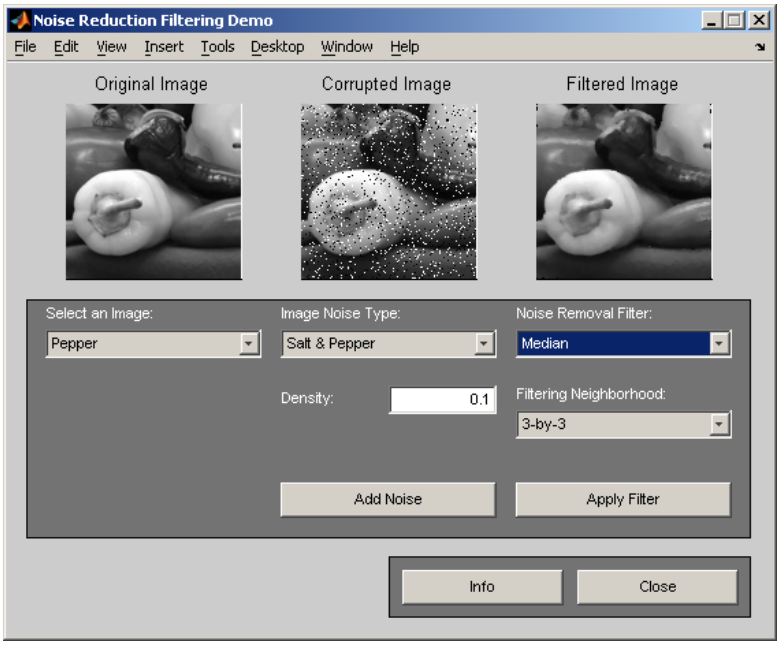
La comparativa entre los distintos tipos de máscaras para el suavizado es presentado mediante una imagen contaminada con ruido blanco.

```
h1=fspecial('average')
h2=conv2([1 2 1],[1 2 1]')/16
h3=fspecial('gaussian',[9 9],1)
imagen= imnoise(imread('cameraman.tif'));
imagen1=imfilter(imagen,h1,'replicate');
imagen2=imfilter(imagen,h2, 'replicate');
imagen3=imfilter(imagen,h3, 'replicate');
imshow([imagen,imagen1,imagen2,imagen3]);
```

Nótese que la máscara de Gauss es la que mejor presenta los resultados. También es de destacar cómo se difuminan los bordes y los pequeños detalles. Para evitar este inconveniente se emplea el filtrado basado en la mediana. Véase la comparativa entre el filtrado lineal, anteriormente propuesto, con esta técnica no lineal:

```
h=fspecial('gaussian',[9 9],1)
imagen= imnoise(imread('circuit.tif'),'salt & pepper');
imagen1=imfilter(imagen,h,'replicate');
imagen2=medfilt2(imagen);
imshow([imagen,imagen1,imagen2])
```

Las diferentes técnicas de eliminación del ruido, pudiendo contaminar artificialmente las imágenes, y, posteriormente aplicar el algoritmo deseado puede ser observado en la demo *nrfiltdemo*:



Hasta ahora se ha simulado el ruido mediante la función *imnoise( )*. Para observarlo sólo hace falta tomar dos imágenes consecutivas con la cámara, de un escenario estático, y restarlas:

```
>>vidobj = videoinput('winvideo')
>> preview(vidobj);
>> start(vidobj);
>> datos=getdata(vidobj,4);
>> imaqmontage (datos);
>>imshow(abs(datos(:,:,:,1)-datos(:,:,:,2))+100);
>>stop(vidobj);
>>delete(vidobj);
```

5.2 Detección de bordes

La detección de bordes es una técnica que pretende localizar la intersección entre los objetos, de forma que se definan los límites que definen las regiones de los objetos presentes en la escena. Su fundamento se basa, bien en el operador derivada local o bien en

el uso de la segunda derivada. En el caso de la primera derivada, para señales bidimensionales, se aplica el operador gradiente. Existen varias formas de la implementación del gradiente. Éste se convierte, al discretizarse, en las mascaras de Roberts, Prewitt o en las de Sobel, con diferentes resultados. Los píxeles correspondientes a los bordes se caracterizan por tener un alto valor de módulo del gradiente:

```
imagen= imread('circuit.tif');
im1 = edge (imagen,'sobel');
im2 = edge (imagen,'prewitt');
im3 = edge (imagen,'roberts');
imshow([im2double(imagen),im1,im2,im3]);
```

En cuanto a la segunda derivada, se usa la discretización del operador laplaciana. En este caso, los píxeles de los bordes corresponden a los pasos por cero de la segunda derivada. Obsérvese la diferencia entre la aplicación del operador gradiente (Sobel) respecto del laplaciana (LoG):

```
imagen1=imfilter(imagen,fspecial('sobel'));
imagen2= imfilter (imagen, fspecial('log'));
imshow([imagen,imagen1,imagen2])
```

La laplaciana ofrece un ancho del borde de un píxel y es de carácter isotrópico. La importancia de los bordes para las siguientes etapas de más alto nivel, ha hecho que existan múltiples estrategias para automatizar la búsqueda de bordes carentes de ruido. El método más empleado es el detector de Canny:

```
im4 = edge (imagen,'canny');
imshow([im1,im2;im3,im4]);
```

La demo *edgedemo* permite variar de estrategia para la detección de los bordes: