PERCEPCIÓN COMPUTACIONAL
Gonzalo Pajares

Práctica 07

1) Detección de cambios (carpeta Otros)

a) Leer las imágenes **Tema07-1.jpg** y **Tema07-2.jpg**. Transformarlas al formato HSI mediante la función de Matlab **rgb2hsv**. Elegir las correspondientes imágenes de intensidad en este formato (recordamos que esta imagen es exactamente la componente 3 de la imagen resultante). Calcular el valor absoluto de la diferencia entre las dos imágenes de intensidad. Utilizar la función **mat2gray** para visualizarla por pantalla.

2) Diferencia acumulada (carpeta Acumulación)

a) Tomar como imagen de referencia dentro de la secuencia **Pica30.jpg**. Crear una imagen de la mismas dimensiones que la de referencia e inicializarla a ceros (función **zeros** de Matlab), sobre esta imagen se acumularán las diferencias. Para leer el resto de imágenes (desde la 31 a la 57) con las que acumular las diferencias, crear un bucle *for* como sigue:

```
for num_img=31:1:57
  I = double(imread(strcat('Pica',num2str(num_img),'.jpg')));

  % código del acumulador
  ………
end
```

De esta manera tenemos la nueva imagen I con la que realizaremos la diferencia con respecto a la de referencia. Utilizar la siguiente expresión con N = 17.

$$a_k = \frac{num\_img - 1}{N - 1}$$

Mostrar el resultado por pantalla mediante la función **mat2gray**

b) Binarizar la imagen obtenida anteriormente y eliminar píxeles superfluos. Para ello calcular un umbral de binarización automático mediante la función **graythresh**. Si el umbral es T y la imagen de diferencias acumulada es D, binarizar D como sigue: Binaria = D > T. Con la imagen binarizada, erosionar la imagen con la función **imerode** y un elemento estructural de 3 x 3. Mostrar el resultado por pantalla.

3) Práctica opcional: flujo óptico mediante Lukas-Kanade

a) Leer las imágenes **Tema07-3.jpg** y **Tema07-4.jpg**. Transformarlas al formato HSI mediante la función de Matlab **rgb2hsv**. Elegir las correspondientes imágenes de intensidad en este formato (recordamos que esta imagen es exactamente la componente 3 de la imagen resultante). Calcular las siguientes derivadas f_x, f_y y f_t como sigue. Donde *ho* y *hv* son los operadores de Sobel horizontal y vertical respectivamente (ver el tema extracción de bordes); *hu* es una matriz de dimensión 3x3 con toso sus elementos con valor 1.

```
fx = conv2(imagen1, ho, 'same') + conv2(imagen1, ho, 'same');
fy = conv2(imagen1, hv, 'same') + conv2(imagen2, hv, 'same');
ft = conv2(imagen1, hu, 'same') - conv2(imagen2, hu, 'same');
```

Tenemos que implementar lo siguiente (transparencia de teoría), con **Ω** una región de vecindad de 3x3:

$$\underbrace{\begin{bmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} u \\ v \end{bmatrix} = -\underbrace{\begin{bmatrix} \sum_\Omega f_t f_x \\ \sum_\Omega f_t f_y \end{bmatrix}}_{\mathbf{b}} \equiv A\boldsymbol{v} = \boldsymbol{b}$$

Solución: $$\boldsymbol{v} = \left(A^T A\right)^{-1} A^T \boldsymbol{b}$$

Creamos dos matrices *u* y *v* de las mismas dimensiones que la imagen original con valores iniciales nulos (función **zeros**).

Para cada píxel, calculamos los valores de las matrices **A** y **b** con el fin de resolver la ecuación anterior para $\boldsymbol{v}$ como sigue: `U = pinv(A'*A)*A'*B;` Tras lo cual se asigna el correspondiente valor al píxel en la posición dada, esto es

```
u(i,j)=U(1);
v(i,j)=U(2);
```

Para visualizar finalmente el resultado obtenido, implementamos el siguiente código:

```
% 1) Cambiamos las filas de orden
u = flipud(u);  v = flipud(v);

%2) Aplicamos el filtro de la mediana en vecindades [5,5]
mu = medfilt2(u,[5 5]); mv = medfilt2(v,[5 5]);

%3) Aplicamos dos descomposiciones piramidales gaussianas para reducri la
%dimensión de las matrices u y v
ru = reducir(reducir(mu)); rv = reducir(reducir(mv));

escala = 0; %valor por defecto (escalado de las flechas de vectores)
figure; quiver(ru, -rv, escala,'r','LineWidth',2), %axis equal
```