PRÁCTICA 7: PUNTOS DE HARRIS STEPHENS

ÍNDICE

Realizado por: Juan Pablo Cano López, jupacanolopez@correo.ugr.es

1-INTRODUCCIÓN DE LA PRÁCTICA

Esta práctica consiste en implementar el algoritmo de Harris, utilizado para la detección de puntos singulares en la construcción de imágenes panorámicas. Las tareas realizadas en la práctica estriban en completar los diferentes procesos vistos en las clases de teoría seguidos por este algoritmo, que se encuentran incompletos en el guión de prácticas.

2-RESPUESTA A PREGUNTAS Y TAREAS

p1 Los gradientes horizontales corresponden a las fronteras verticales y los gradientes verticales a las fronteras horizontales. ¿Por qué?

Porque en la operación de convolución el filtro se rota 180 grados (cambiando su orientación) y al realizarse la multiplicación y la agregación con la imagen tiene este resultado.

p2 Completa la función Harris.m (disponible en Google Drive UGR) para encontrar los puntos de Harris. Compara su resultado con la salida proporcionada por la función detectHarrisFeatures de Matlab. Comprueba su funcionamiento en la creación de panoramas de la práctica anterior

La diferencia no es visible pero el profesor explicó en clase que en el algoritmo de Harris implementado por matlab se consideran valores interpolados entre píxeles como posibles puntos singulares, en nuestra implementación los valores de los puntos solo pueden ser píxeles enteros.

p3 Comenta las partes de código que se han completado

```
Fghorizontal=[-1 0 1];
Fgvertical=[-1 0 1]';
Ix = imfilter(I2,Fghorizontal); %gradientes horizontales
Iy = imfilter(I2,Fgvertical);%gradientes verticales
%gradientes horizontales, fronteras verticales
%gradientes verticales, fronteras horizontales
```

Fig 1- Convolución de los filtros gradiente con la imagen

Opté por utilizar imfilter porque con conv2 los tamaños resultantes no eran exáctamente iguales y se obtienen errores en el script final

```
A =(Ix.^2); %Esto parece ser lo correcto
B =(Iy.^2);
C =(Ix.*Iy);

% HASTA AQUI

% Crear un filtro gausiano 5x5 con desv. 5/3.
filtroGausiano = fspecial('gaussian', 5, 5/3);

% Filtrar A, B, C con dicho filtro
A = imfilter(A, filtroGausiano, 'replicate');
B = imfilter(B, filtroGausiano, 'replicate');
C = imfilter(C, filtroGausiano, 'replicate');
```

Fig 2- Cálculo de A, B y C en la matriz de momentos de segundo orden H

```
detH = (A.*B)-(C.*C);  % determinante de H
trH = A+C; % traza de H
k = 0.04; % k
R = detH-(k.*trH.^2); % determinante de H - k * cuadrado de la traza de H;
```

Fig 3-Cálculo de R, imagen sin umbralizar

```
%todo lo que esté por debajo de un 1% del máximo no es una esquina
umbral = 0.01 * max(R(:));
% COMPLETAR AQUI
R(R < umbral) = 0;
%Lo debemos poner a 0
% HASTA AQUI

subplot(1,2,2); imagesc(R);  axis image; colormap(jet); colorbar;  title('Métrica de Harris umbralizada');
truesize;
```

Fig 4-Cálculo y representación de R umbralizada

```
maximos(y+1:y+radio,x+1:x+radio)=0;
maximos(y-1:y-radio,x-1:x-radio)=0;
maximos(y,x)=true; %Esto es correcto pero debo poner a 0 el entorno
```

Fig 5-Eliminación de los no máximos locales

De este último paso estoy muy inseguro en cuanto a si es correcto lo que he hecho pero el resultado final no ha sufrido cambios visibles y el script fue más rápido así que quizás sea correcto.

```
%% Detectar puntos singulares y extraer descriptores
%%%% COMPLETAR AQUI
puntos = Harris_clase2(imgGris);
[caract, puntos] = extractFeatures(imgGris, puntos); %Todo esto es de la primera
% imagen
%%%% HASTA AQUI
```

Fig 6-Se ejecuta la función creada en lugar de “detectHarrisfeatures”

p4 Muestra resultados obtenidos

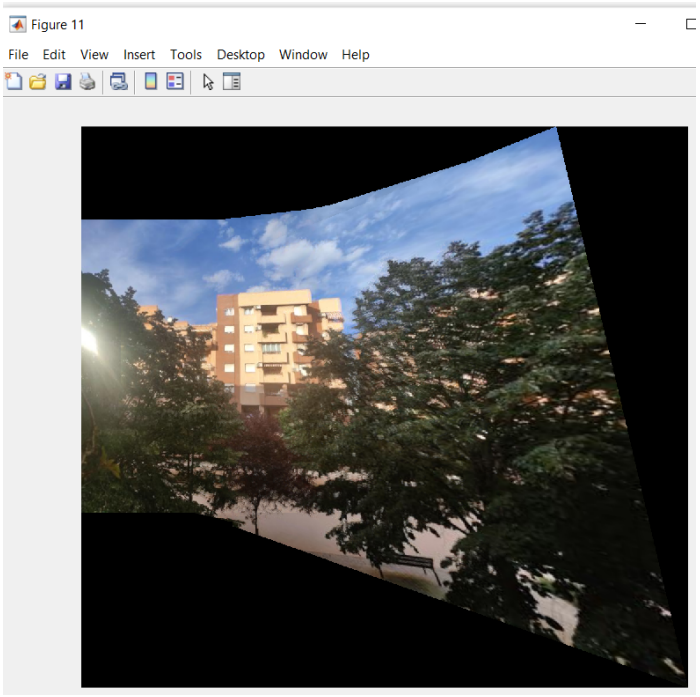

Fig7-Imagen panorámica resultante

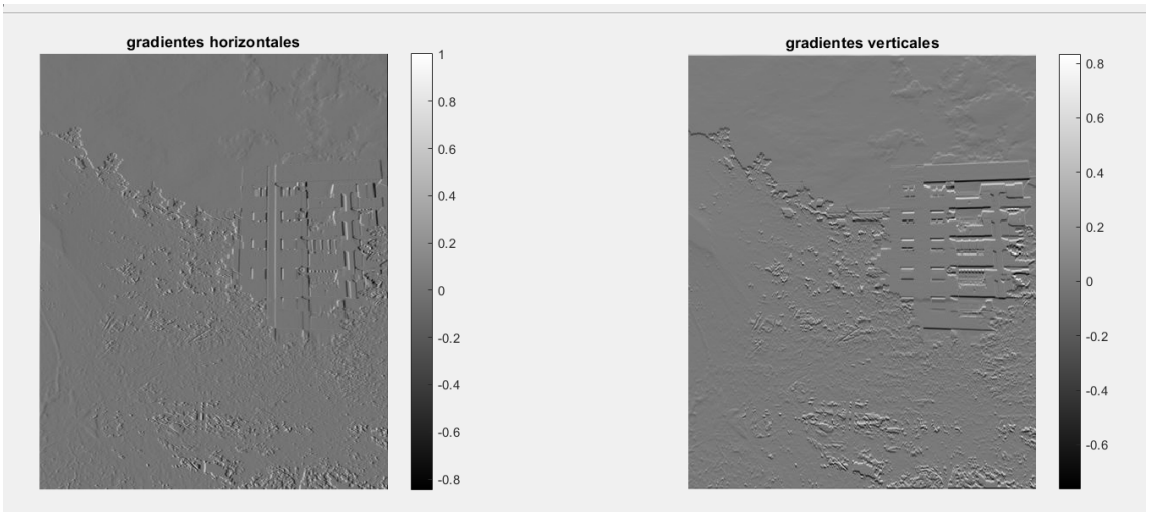

Fig8-Imagen que muestra el resultado de aplicar los filtros de gradiente

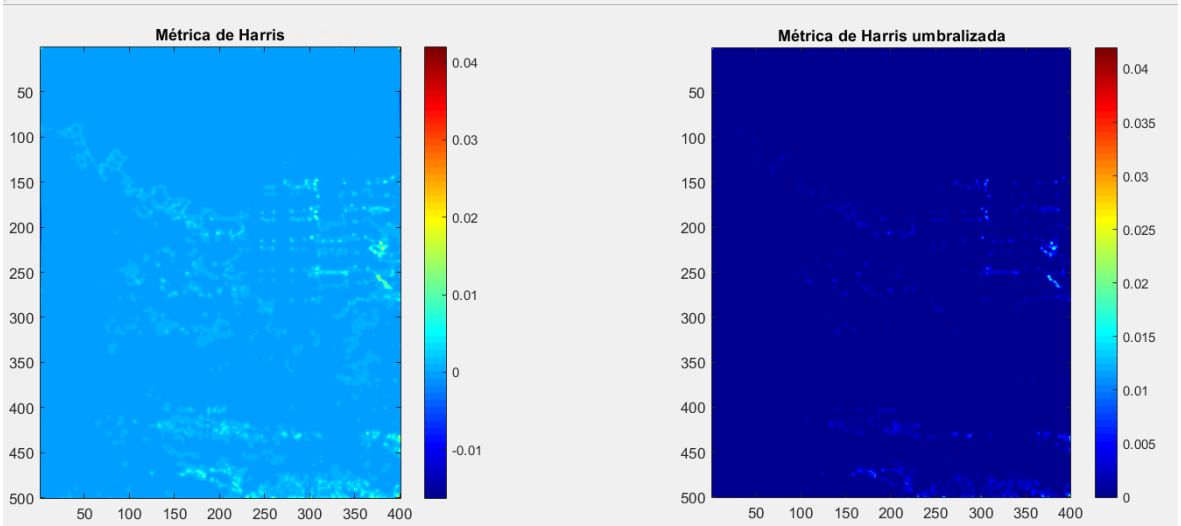

Fig9-Ejemplo de R sin umbralizar vs R umbralizada