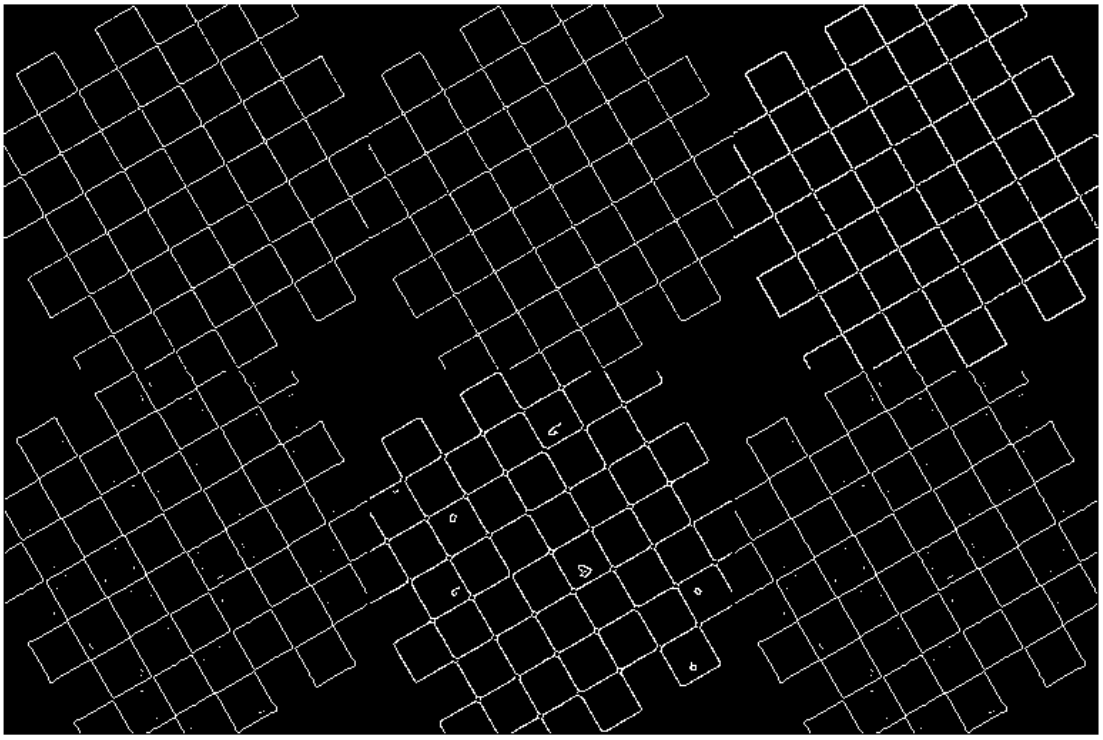
Sesión 1. Extracción de contornos

1. **Cargar la imagen "ajedrez.gif" en el Workspace y rotarla 30 grados en sentido anti-horario manteniendo el mismo tamaño de la imagen (utilizando el parámetro 'crop'). A partir de la imagen rotada, se pide:**

```
>> img = imread('ivc_practica2_imagenes/ajedrez.gif');
>> ajedrez = imrotate(img, 30, 'crop');
```

   a) **Calcular la imagen de bordes, aplicando todos los filtros posibles que admite la función 'edge' (sobel, prewitt, roberts, zerocross, canny y log). Comparar los resultados.**

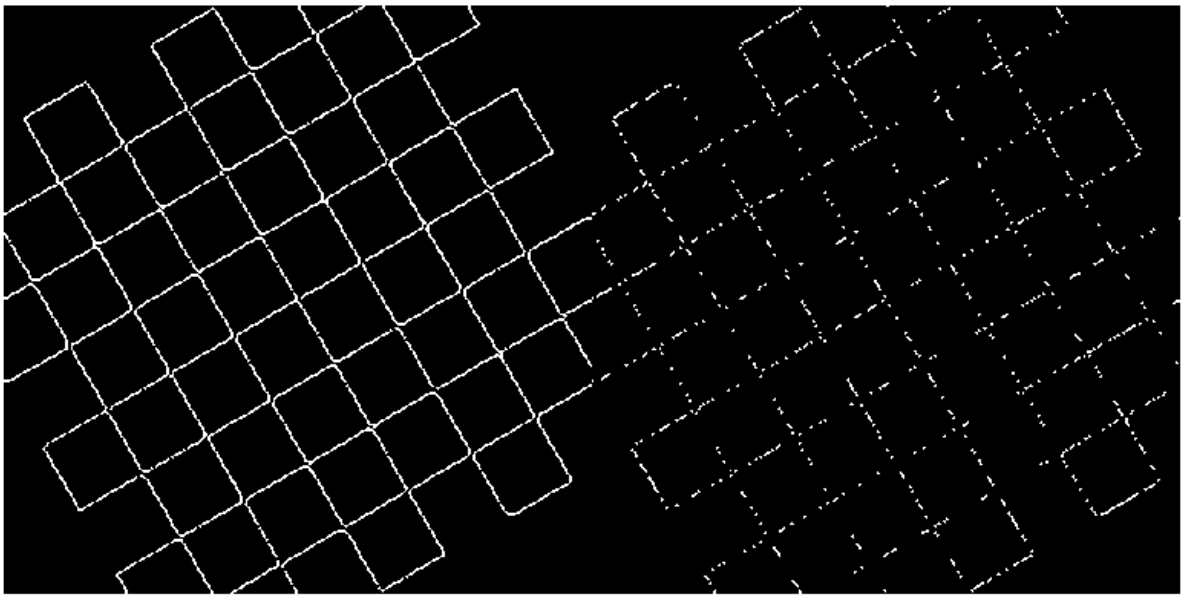
```
>> sobel = edge(ajedrez, 'Sobel');
>> prewitt = edge(ajedrez, 'Prewitt');
>> roberts = edge(ajedrez, 'Roberts');
>> zerocross = edge(ajedrez, 'Zerocross');
>> canny = edge(ajedrez, 'Canny');
>> log = edge(ajedrez, 'Log');

%Creamos un montaje a partir de una imagen multiframe
>> montage({sobel, prewitt, roberts, zerocross, canny, log});
```



```
%El filtro Sobel muestra una línea mejor definida que
%Prewitt e intersecciones más suaves, mientras que el filtro
%Roberts obtiene líneas mejor definidas que las anteriores y
%y las intersecciones se cruzan, mientras que en Prewit y
%Sobel no llegan a cortarse.
%Zerocross y Log visualizan un poco de ruido, mientras que en
%el filtro Canny se muestra un filtro mucho más granulado.
```

b) Explicar con ejemplos cómo afecta el umbral en la detección de bordes utilizando el filtro de Sobel.

```
>> umbral = edge(ajedrez, 'Sobel', 0.05);
>> umbral2 = edge(ajedrez, 'Sobel', 0.1);
>> montage({umbral, umbral2});
```



```
%Por lo tanto, la primera imagen se visualiza con líneas
%mejor definidas debido que ignora menos bordes que la segunda,
%y la continuidad en las líneas es mayor.
%Es probable que si disminuimos demasiado el umbral se muestre
%ruido de la imagen.
```

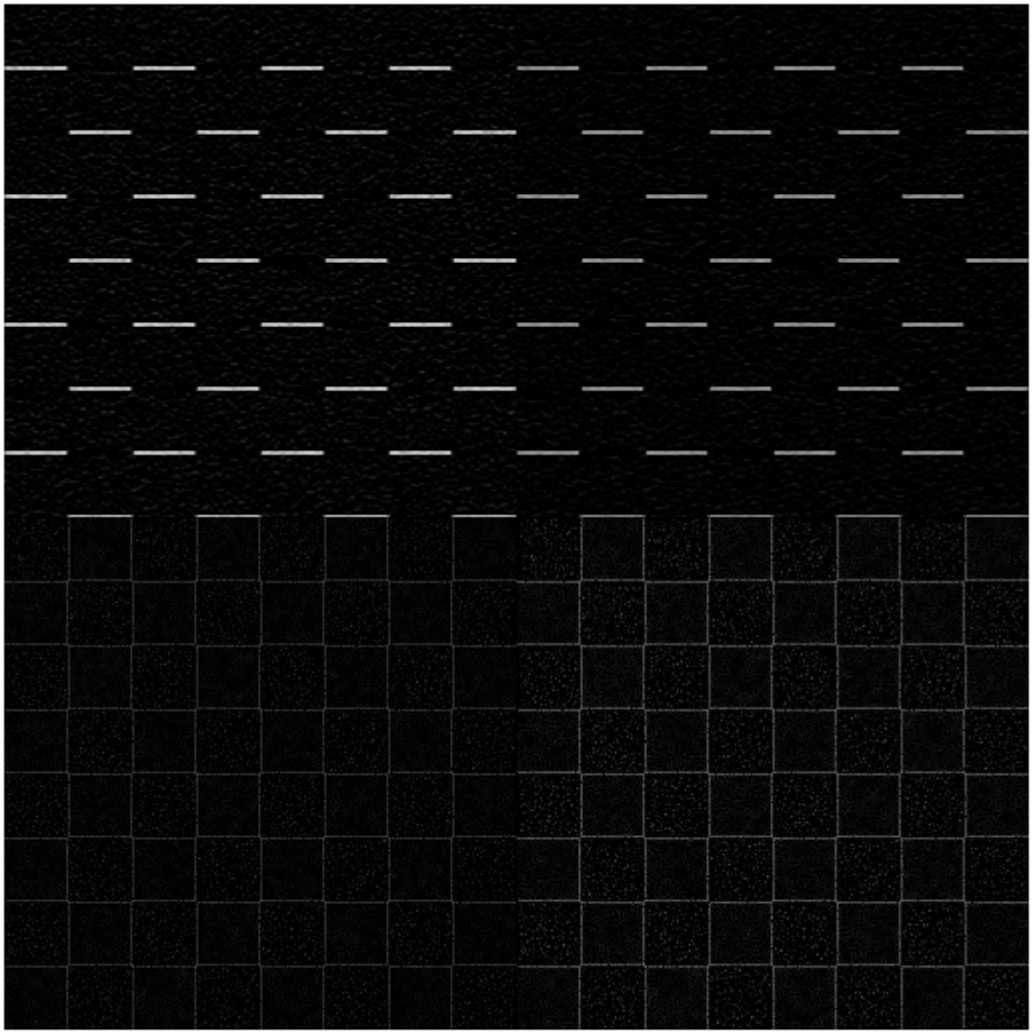
c) Rellenar una tabla con los tiempos de procesamiento de cada uno de los filtros aplicados, ordenándolos por tiempo de proceso. Nota: tic inicia un temporizador para medir tiempos de proceso en Matlab, mientras que toc para el temporizador.

```
>> tic
sobel = edge(ajedrez, 'Sobel');
toc
Elapsed time is 0.012411 seconds.
>> tic
prewitt = edge(ajedrez, 'Prewitt');
toc
Elapsed time is 0.012233 seconds.
>> tic
roberts = edge(ajedrez, 'Roberts');
toc
Elapsed time is 0.016933 seconds.
>> tic
zerocross = edge(ajedrez, 'Zerocross');
toc
Elapsed time is 0.039154 seconds.
>> tic
canny = edge(ajedrez, 'Canny');
```

```
toc
Elapsed time is 0.041569 seconds.
>> tic
log = edge(ajedrez, 'Log');
toc
Elapsed time is 0.030672 seconds.
>>

%1. Canny
%2. Zerocross
%3. Log
%4. Roberts
%5. Sobel
%6. Prewitt

%Los que tardan más tienen que pintar mayor cantidad de blanco.
```

2. **A partir de la imagen "ajedrez.gif", se pide calcular la imagen de bordes, construyendo filtros con la función 'fspecial' y convolucionando la imagen con éstos (utilizando 'imfilter'). En particular escoger como parámetro de fspecial: sobel, prewitt, laplacian, log. Comparar los resultados.**

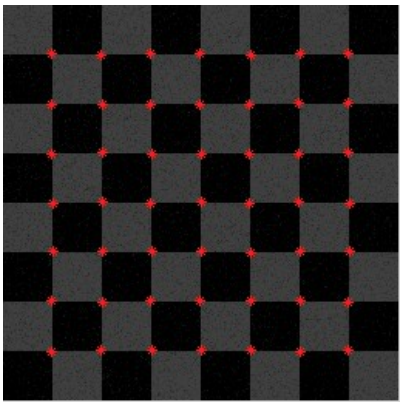
```
>> img = imread('ivc_practica2_imagenes/ajedrez.gif');
>>
>> sobel = fspecial('sobel');
>> prewitt = fspecial('prewitt');
>> laplacian = fspecial('laplacian');
>> log = fspecial('log');
>>
>> imgSobel = imfilter(img, sobel);
>> imgPrewitt = imfilter(img, prewitt);
>> imgLaplacian = imfilter(img, laplacian);
>> imgLog = imfilter(img, log);
>>
>> montage({imgSobel, imgPrewitt, imgLaplacian, imgLog});
```



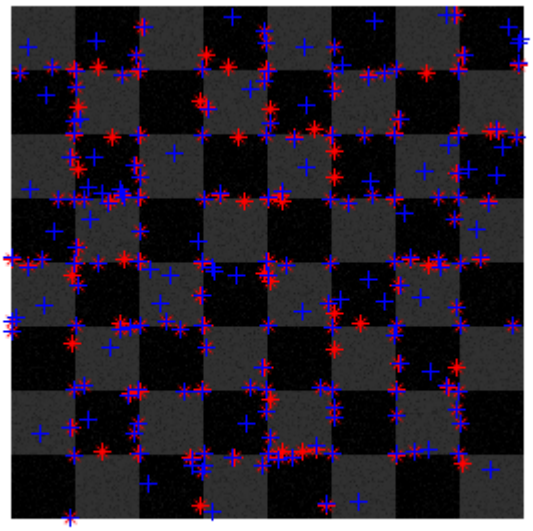
```
%Tanto Prewitt como Sobel enfatizan los contornos horizontales
%aproximando un gradiente vertical. El filtro de Sobel,
%posee, sin embargo, mayor cantidad de ruido.
%El filtro de Laplacian visualiza contornos verticales
%y horizontales junto a una paqueña cantidad de ruido interior.
%El filtro de Log posee caraterísticas similares al Laplacian
%con la diferencia de que tiene mayor intensidad o nivel
%de gris.
```

Sesión 2. Extracción de esquinas

3. **Se pide realizar la extracción de esquinas con el algoritmo de Harris-Stephen y Shi-Tomasi sobre la imagen "ajedrez.gif". Realizar un script dónde se contabilicen cuántas esquinas ha detectado cada método. ¿Cómo se modificaría la búsqueda para obtener únicamente las esquinas mostradas en la siguiente figura? Describe los parámetros de cada método y realiza varias pruebas modificándolos para explicar cómo afectan los parámetros en la búsqueda de esquinas.**

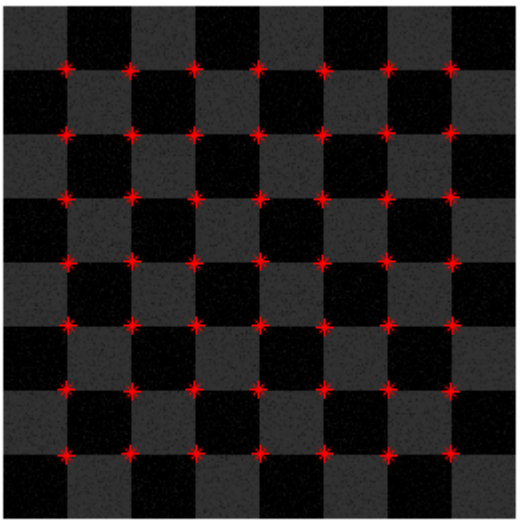


```
>> ejercicio3
Esquinas con Harris-Stephen: 175
Esquinas con Shi-Tomasi: 200
>>
```

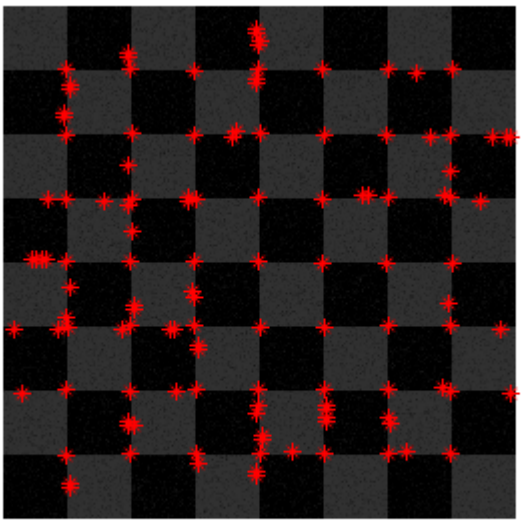


```
%Para conseguir las esquinas de la figura tenemos que
%especificar el nivel de calidad mínimo con QualityLevel

>> harris = corner(ajedrez, 'Harris', 'QualityLevel', 0.2);
>> …
```

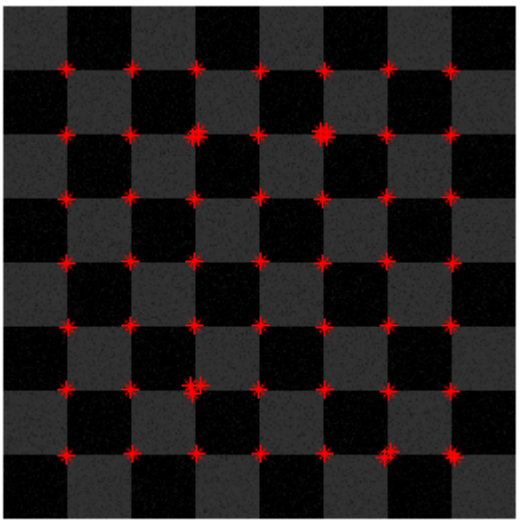
Esquinas con Harris-Stephen: 49



```
%FilterCoefficients

>> harris = corner(ajedrez, 'Harris', 'FilterCoefficients',
fspecial('gaussian',[5 1],0.5));
>> …
Esquinas con Harris-Stephen: 123
```



```
%SensitivityFactor

>> harris = corner(ajedrez, 'Harris', 'SensitivityFactor', 0.24);
>> …
Esquinas con Harris-Stephen: 58
```

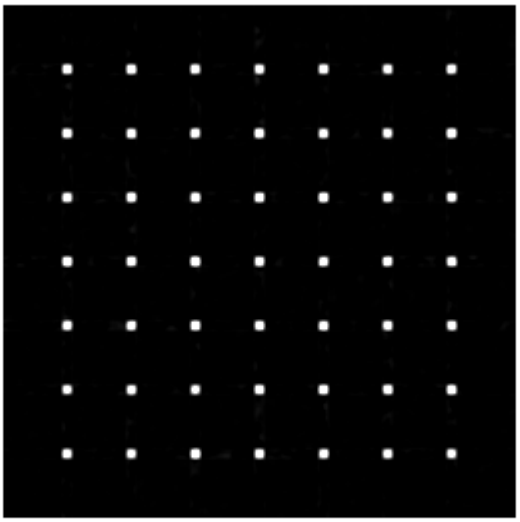


```
%Con el algoritmo de Harris se obtiene un mejor resultado si aplicar
%ningún parámetro. El valor de SensivityFactor' en Harris mejora
%el algoritmo cuánto mayor es el número, en caso de disminuir este
%valor se van a detectar más falsas esquinas.
%Calidad mínima aceptada de esquinas, especificada como el par
%separado por comas consistente en 'QualityLevel' y un escalar
%numérico en el rango (0,1).
%Coeficientes de filtrado para el filtro de suavizado
%separable, especificado como el par separado por comas consistente
%en 'FilterCoefficients' y un vector numérico.
```

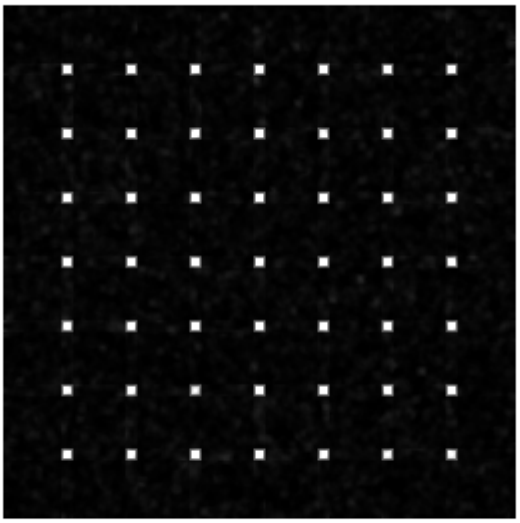
4. Describe el funcionamiento de la función cornermetric. ¿Qué diferencia hay entre corner y cornermetric? Utiliza cornermetric sobre "ajedrez.gif" para obtener las esquinas mostradas en la imagen anterior. Modifica los parámetros de cornermetric para ver cómo afectan a la detección de las esquinas. ¿Crees que el método de Harris-Stephen se ve afectado por el ruido? Modifica el brillo de la imagen, ¿afecta la modificación de brillo al número de puntos detectados con los mismos parámetros en el método de Harris-Stephen?

```
>> ajedrez = imread('ivc_practica2_imagenes/ajedrez.gif');
>> imagen = cornermetric(ajedrez, 'Harris');
%Ajuste la matriz métrica de esquina para visualizarla...
```

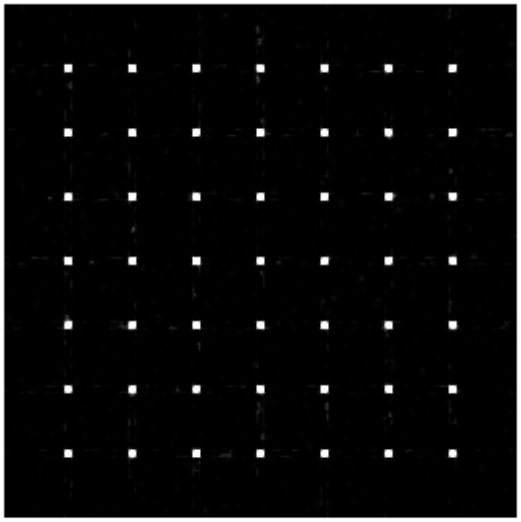
```
>> imagenAjustada = imadjust(imagen);
>> imshow(imagenAjustada);
```



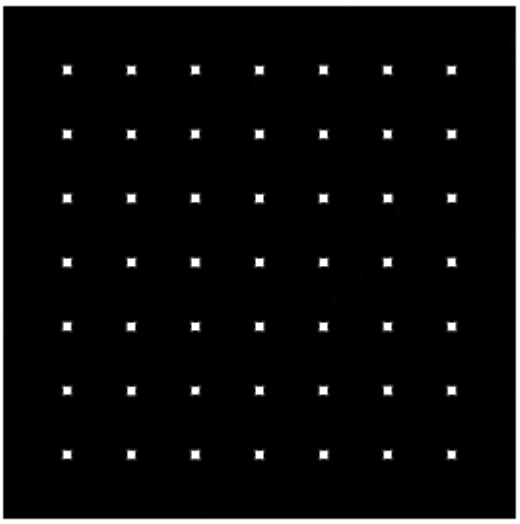
```
>> imagen = cornermetric(ajedrez, 'Harris');
%Ajuste la matriz métrica de esquina para visualizarla...
>> imagenAjustada = imadjust(imagen, 'MinimumEigenvalue');
>> imshow(imagenAjustada);
```



```
>> imagen = cornermetric(ajedrez, 'FilterCoefficients',
fspecial('gaussian',[5 1],0.8));
%Ajuste la matriz métrica de esquina para visualizarla...
>> imagenAjustada = imadjust(imagen);
>> imshow(imagenAjustada);
```



```
>> imagen = cornermetric(ajedrez, 'SensitivityFactor', 0.15);
%Ajuste la matriz métrica de esquina para visualizarla...
>> imagenAjustada = imadjust(imagen);
>> imshow(imagenAjustada);
```



```
%Se puede emplear SensitivityFactor, en caso de ponerle un valor
%superior a 0.22 los puntos no se muestran, si este valor es inferior se
%van a mostrar los puntos. Cuanto más pequeño sea el valor va a ser
%mayor la sensibilidad ya que el rango de valores que coge es más
%pequeño por lo que detecta el cambio de las esquinas ej(0.005).
%Si que se ve afectado por el brillo, en caso de no aplicarle brillo
%al ruido no va a funcionar bien el algoritmo ya que hay muchos
%valores y no detecta las esquinas.
```