Crear una imagen im (inicializada a 128) donde guardar el resultado final: la forma más fácil es multiplicar una de las imágenes originales por 0 y sumarle 128.

Llenar la imagen im con valores de im1 o im2 según qué imagen esté más enfocada en cada píxel. Decidir un umbral y asignar las imágenes según el valor de d1/d2:

- Si d1/d2 > umbral usar valor de im1 para ese píxel
- Si d1/d2 < 1/umbral usar valor de im2.
- En el resto de los casos (no hay un claro ganador) conservar el valor 128.

Adjuntar la imagen resultante para un umbral de 2.0 (usar el colormap adecuado para verla correctamente como una imagen en grises). Adjuntar vuestro código.
Se pueden usar bucles o (más valorado) aprovechar el indexado lógico en Matlab.

La imagen resultado es ruidosa (ver abajo a la izquierda) porque la decisión sobre si usar im1 o im2 se ha hecho a nivel de píxel. Sería mejor tomar esta decisión considerando una vecindad: si alrededor de un píxel la mayoría de los píxeles de la imagen 1 son mejores asignaremos ese píxel a la imagen 1 y viceversa.

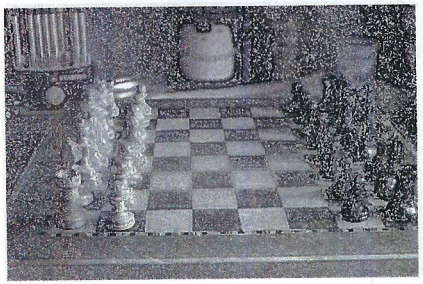
Para implementar este enfoque construiremos una matriz lógica M que valga 1/0 según "gane" la 1ª o 2ª imagen:

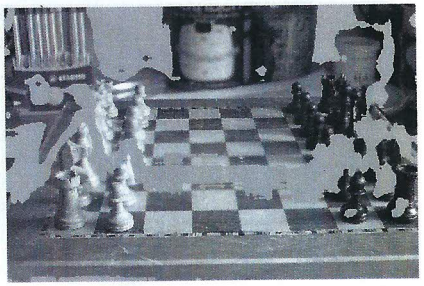
M = (d1>d2);

Si usáramos M para decidir estaríamos haciendo esencialmente lo mismo que antes.

En vez de usar directamente los valores de M, los filtraremos previamente con un filtro promedio de tamaño NxN: P=ones(N)/(N*N); usando fc_imfilter. Al estar promediando 0's y 1's un valor de 0.5 indica que en el área NxN considerada hay el mismo número de píxeles adjudicados a ambas imágenes. Valores por debajo o por encima indican que una de las imágenes es mayoritariamente mejor en esa zona.

Aplicar los pasos anteriores y usar la salida del filtrado para decidir la imagen a usar. Como antes, dejar un intervalo [0.4,0.6] de "indecisos" y asignar a una u otra imagen los valores menores de 0.4 o mayores que 0.6.





Adjuntar código e imagen resultante (debe ser algo similar a imagen de la derecha).

LAB1: Los ficheros 'ajedrez1.jpg' y 'ajedrez2.jpg' contienen dos imágenes en grises. Cargarlas (con imread) en im1 e im2 respectivamente y visualizarlas con image(). Usar el colormap adecuado para que se vean bien. Adjuntar imagen de im1.

Veréis que las dos imágenes son de la misma escena pero han sido enfocadas en puntos distintos. Al haberse usado una apertura grande su profundidad de foco (zona enfocada correctamente) es pequeña. Vamos a tratar de combinar ambas imágenes escogiendo las partes mejor enfocadas de cada una de ellas.

Para ello debemos intentar cuantificar el grado de "desenfoque" de cada punto de la imagen. El primer paso es aplicar un filtrado promedio a cada imagen. Como el efecto de un filtro promedio es básicamente emborronar la imagen, tendrá poco efecto en aquellas zonas que ya estaban desenfocadas. Calculando la diferencia con la imagen original podemos tener una medida del desenfoque: diferencias pequeñas indican que ese punto ya estaba desenfocado y no cambia mucho al aplicar el filtro.

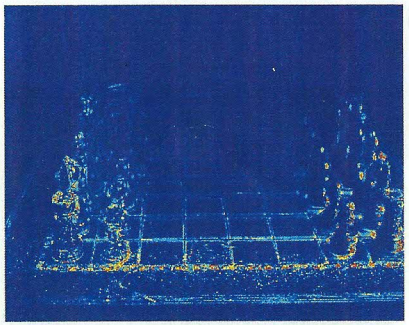
Crear un filtro 2D gaussiano con $\sigma = 5$ y un soporte x=-10:10. (ver los detalles en LAB4). El filtro H resultante debe ser una matriz de 21x21. Aseguraros de que el filtro esté normalizado, esto es, sum(H(:))=1.

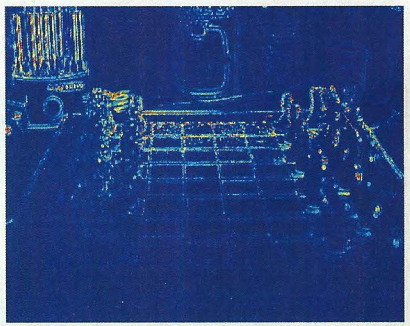
Adjuntar código para crear H.

Nota: si no sabéis generar el filtro gaussiano, crear un filtro promedio constante con H=ones(21)/(21*21) y seguir con el problema.

Filtrar las imágenes originales usando fc_imfilter(double(im) , H , 'symmetric'). Calcular en ambos casos el valor absoluto de la diferencia entre la imagen filtrada y la original. Guardar los resultados en d1 y d2.

Visualizar d1 y d2 (sin especificar colormap, dejando que MATLAB use su paleta de colores por defecto). El resultado debe ser algo similar a la imagen adjunta.





Adjuntar código usado y vuestras propias imágenes obtenidas para d1 y d2

Crear una imagen im (inicializada a 128) donde guardar el resultado final: la forma más fácil es multiplicar una de las imágenes originales por 0 y sumarle 128.

Llenar la imagen im con valores de im1 o im2 según qué imagen esté más enfocada en cada píxel. Decidir un umbral y asignar las imágenes según el valor de d1/d2:

- Si d1/d2 > umbral usar valor de im1 para ese píxel
- Si d1/d2 < 1/umbral usar valor de im2.
- En el resto de los casos (no hay un claro ganador) conservar el valor 128.

Adjuntar la imagen resultante para un umbral de 2.0 (usar el colormap adecuado para verla correctamente como una imagen en grises). Adjuntar vuestro código.
Se pueden usar bucles o (más valorado) aprovechar el indexado lógico en Matlab.

La imagen resultado es ruidosa (ver abajo a la izquierda) porque la decisión sobre si usar im1 o im2 se ha hecho a nivel de píxel. Sería mejor tomar esta decisión considerando una vecindad: si alrededor de un píxel la mayoría de los píxeles de la imagen 1 son mejores asignaremos ese píxel a la imagen 1 y viceversa.

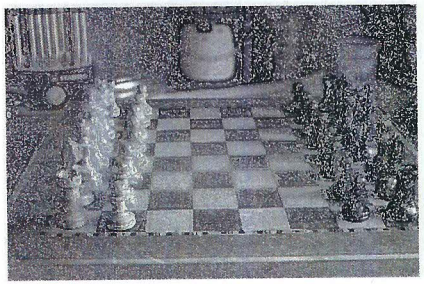
Para implementar este enfoque construiremos una matriz lógica M que valga 1/0 según "gane" la 1ª o 2ª imagen:

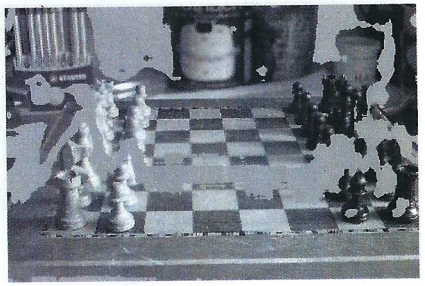
M = (d1>d2);

Si usáramos M para decidir estaríamos haciendo esencialmente lo mismo que antes.

En vez de usar directamente los valores de M, los filtraremos previamente con un filtro promedio de tamaño NxN: P=ones(N)/(N*N); usando fc_imfilter. Al estar promediando 0's y 1's un valor de 0.5 indica que en el área NxN considerada hay el mismo número de píxeles adjudicados a ambas imágenes. Valores por debajo o por encima indican que una de las imágenes es mayoritariamente mejor en esa zona.

Aplicar los pasos anteriores y usar la salida del filtrado para decidir la imagen a usar. Como antes, dejar un intervalo [0.4,0.6] de "indecisos" y asignar a una u otra imagen los valores menores de 0.4 o mayores que 0.6.





Adjuntar código e imagen resultante (debe ser algo similar a imagen de la derecha).

LAB1: Los ficheros 'ajedrez1.jpg' y 'ajedrez2.jpg' contienen dos imágenes en grises. Cargarlas (con imread) en im1 e im2 respectivamente y visualizarlas con image(). Usar el colormap adecuado para que se vean bien. Adjuntar imagen de im1.

Veréis que las dos imágenes son de la misma escena pero han sido enfocadas en puntos distintos. Al haberse usado una apertura grande su profundidad de foco (zona enfocada correctamente) es pequeña. Vamos a tratar de combinar ambas imágenes escogiendo las partes mejor enfocadas de cada una de ellas.

Para ello debemos intentar cuantificar el grado de "desenfoque" de cada punto de la imagen. El primer paso es aplicar un filtrado promedio a cada imagen. Como el efecto de un filtro promedio es básicamente emborronar la imagen, tendrá poco efecto en aquellas zonas que ya estaban desenfocadas. Calculando la diferencia con la imagen original podemos tener una medida del desenfoque: diferencias pequeñas indican que ese punto ya estaba desenfocado y no cambia mucho al aplicar el filtro.

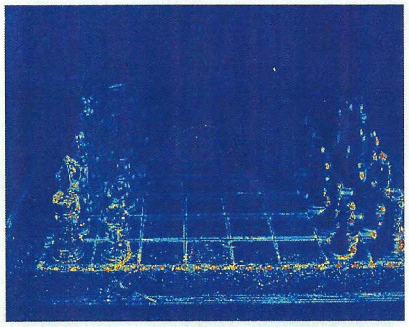
Crear un filtro 2D gaussiano con $\sigma = 5$ y un soporte x=-10:10. (ver los detalles en LAB4). El filtro H resultante debe ser una matriz de 21x21. Aseguraros de que el filtro esté normalizado, esto es, sum(H(:))=1.

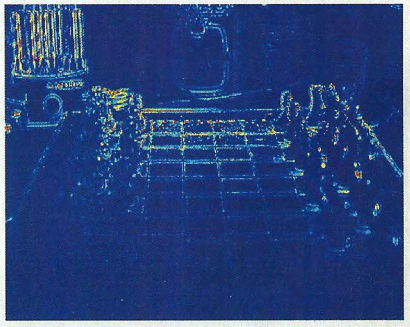
Adjuntar código para crear H.

Nota: si no sabéis generar el filtro gaussiano, crear un filtro promedio constante con H=ones(21)/(21*21) y seguir con el problema.

Filtrar las imágenes originales usando fc_imfilter(double(im) , H , 'symmetric'). Calcular en ambos casos el valor absoluto de la diferencia entre la imagen filtrada y la original. Guardar los resultados en d1 y d2.

Visualizar d1 y d2 (sin especificar colormap, dejando que MATLAB use su paleta de colores por defecto). El resultado debe ser algo similar a la imagen adjunta.





Adjuntar código usado y vuestras propias imágenes obtenidas para d1 y d2