Spatial Mask Processing in the Frequency Domain
(See other side for filtering with a frequency domain filter.)

Given image \( f \) of size \( M \times N \) and spatial domain mask \( m \) of size \( R \times S \).

1. Image and mask must be padded to size \( P \times Q \) where \( P = (M+R-1) \) and \( Q = (N+S-1) \).

2. Compute 2D FFT of padded \( f \) and \( m \) (\texttt{fft2} can do the padding). The results will be complex matrices of size \( P \times Q \).

\[
\begin{align*}
\hat{f} &= \text{fft2}(f, P, Q); \\
\hat{m} &= \text{fft2}(m, P, Q);
\end{align*}
\]

3. Compute the inverse FFT of the product. Convert to \texttt{uint8} if \( f \) was \texttt{uint8}. This result will be the same as that obtained by convolving \( f \) and \( m \) using \texttt{conv2} with the 'full' (default) option. \( g_{\text{full}} \) will be of size \( P \times Q \).

\[
\begin{align*}
g_{\text{full}} &= \text{uint8}(\text{real}(\text{ifft2}(\hat{f} \cdot \hat{m}))); \\
\end{align*}
\]

4. To recover the same result as \texttt{conv2} would give with the 'same' option. Extract the central portion of \( g_{\text{full}} \). Let \( \text{mind} = (M-1)/2 + 1 \) and \( \text{nind} = (N-1)/2 + 1 \). The result will be \( M \times N \).

\[
\begin{align*}
g_{\text{same}} &= g_{\text{full}}(\text{mind}:\text{mind}+M, \text{nind}:\text{nind}+N); \\
\end{align*}
\]
Filtering in the Frequency Domain

(See other side for filtering in the frequency domain with a spatial mask.)

Given image f of size M x N.

1. Pad image to size P x Q where P=2M and Q = 2N.

2. Compute the shifted 2D FFT of padded f (fft2 can do the padding). The result will be complex matrix of size P x Q.

   \[ f_{\text{fft}} = \text{fftshift}(\text{fft2}(f, P, Q)); \]

3. Generate a real filter function H of size P x Q centered at the fftshifted coordinates. A distance squared matrix with values containing pixel squared distances from the center may be useful when creating the filter. The code below creates a distance squared matrix (DSQ) and then uses that matrix to create a Gaussian low-pass filter:

   ```matlab
   function d = d_squared_ctr(M, N)
   xx = ((1:M) - idivide(M+2,2))' * ones(1, N);
   yy = ones(M, 1) * ((1:N) - idivide(N+2,2));
   d = xx.^2 + yy.^2;
end

   DSQ = d_squared_ctr(P, Q);
   H = exp(-2*DSQ/(2*D0^2));
```

4. Compute the product of the spectrum and the filter:

   \[ g_{\text{fft}} = f_{\text{fft}} \cdot H; \]

5. Compute the inverse FFT of the product. **Be sure to ifftshift back first!** Convert to uint8 if the original image was uint8.

   \[ g_{\text{full}} = \text{uint8}(\text{real}(\text{ifft2}(\text{ifftshift}(g_{\text{fft}))))); \]

6. Crop the image to the same size as the original:

   \[ g_{\text{same}} = g_{\text{full}}(1:M, 1:N); \]