conversion to FFT and Back

inverse fourier gives back the original image

In[1]:=
```
ClearAll["Global`*"]
```

In[2]:=
```
img = ColorConvert[Import["ExampleData/lena.tif"], "Grayscale"]
```

Out[2]=

In[3]:=
```
fft = Fourier[ImageData[img]];
```

In[4]:=
```
Image@Chop@InverseFourier[fft]
```

Out[4]=

another way

In[5]:=
```
ClearAll["Global`*"];
```

In[6]:=
```
img = ColorConvert[Import["ExampleData/lena.tif"], "Grayscale"]
```

Out[6]=

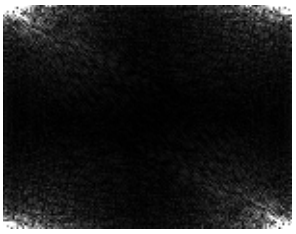
In[7]:=
```
imgdata = ImageData@img;
dim = {nrow, ncol} = Dimensions[imgdata]
```

Out[8]=
```
{116, 150}
```

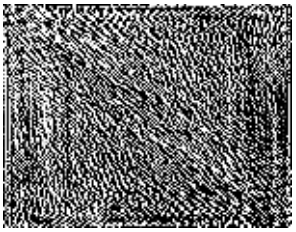
In[9]:=
```
absArray = ConstantArray[0, dim];
```

In[10]:=
```
(abs = Abs[Fourier[ImageData[img]]]) // Image
(arg = Arg[Fourier[ImageData[img]]]) // Image
```

Out[10]=



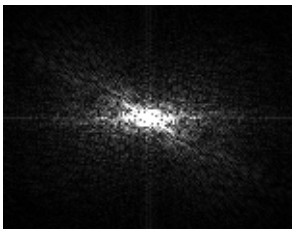
Out[11]=



In[12]:=
```
absArray[[ ;; nrow / 2, ;; ncol / 2]] = abs[[(nrow / 2 + 1) ;;, (ncol / 2 + 1) ;;]];
absArray[[ ;; nrow / 2, (ncol / 2 + 1) ;;]] = abs[[(nrow / 2 + 1) ;;, ;; ncol / 2]];
absArray[[(nrow / 2 + 1) ;;, (ncol / 2 + 1) ;;]] = abs[[ ;; nrow / 2 , ;; ncol / 2]];
absArray[[(nrow / 2 + 1) ;;, ;; ncol / 2]] = abs[[ ;; nrow / 2 , (ncol / 2 + 1) ;;]];
```

In[16]:=
```
Image@absArray
```

Out[16]=

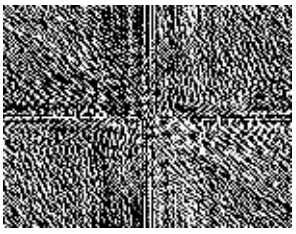


In[17]:=
```
argArray = ConstantArray[0, dim];
```

In[18]:=
```
argArray[[ ;; nrow / 2, ;; ncol / 2]] = arg[[(nrow / 2 + 1) ;;, (ncol / 2 + 1) ;;]];
argArray[[ ;; nrow / 2, (ncol / 2 + 1) ;;]] = arg[[(nrow / 2 + 1) ;;, ;; ncol / 2]];
argArray[[(nrow / 2 + 1) ;;, (ncol / 2 + 1) ;;]] = arg[[ ;; nrow / 2 , ;; ncol / 2]];
argArray[[(nrow / 2 + 1) ;;, ;; ncol / 2]] = arg[[ ;; nrow / 2 , (ncol / 2 + 1) ;;]];
```

In[22]:=
```
Image@argArray
```

Out[22]=



In[23]:=
```
Chop[InverseFourier[(abs E^(I arg))]] // Image
```

Out[23]=

yet another way

In[24]:=
```
ClearAll["Global`*"]
```

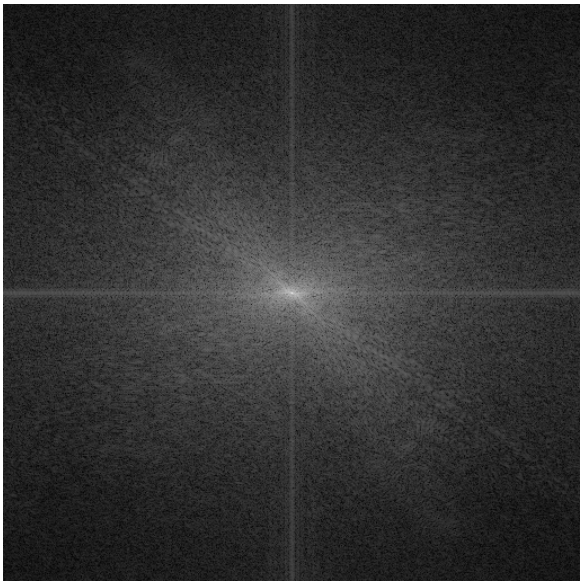
In[25]:=
```
image = Image[ColorConvert[ , "Grayscale"], ImageSize → Small]
```

Out[25]=

In[26]:=
```
data = ImageData[image];
{nRow, nCol} = Dimensions[data];
d = data;
d = d * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
fw = Fourier[d, FourierParameters → {1, 1}];
```

In[31]:=
```
absvis = Log[1 + Abs@fw];
Image[absvis / Max[absvis], ImageSize → 300]
```

Out[32]=



In[33]:=
```
Dimensions[fw]
```

Out[33]=
```
{512, 512}
```

In[34]:=
```
ifw = Chop@InverseFourier[fw, FourierParameters → {1, 1}];
ifw = ifw * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
GraphicsRow[{Image[ifw], Image[data]}, ImageSize → 300]
```

Out[36]=

Amplitude surgery using masks

example 1

In[37]:=
```
ClearAll["Global`*"];
```

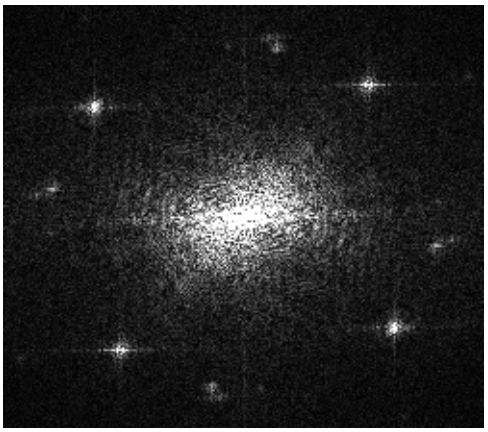
In[38]:=
```
img =  ~ColorConvert~"Grayscale"
```

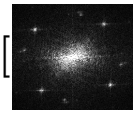
Out[38]=

In[39]:=
```
data = ImageData[img];
{nRow, nCol} = Dimensions[data];
data = data * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
fw = Fourier[data, FourierParameters → {0, 1}];
```

In[43]:=
```
abs = Abs@fw;
Image[abs]
```

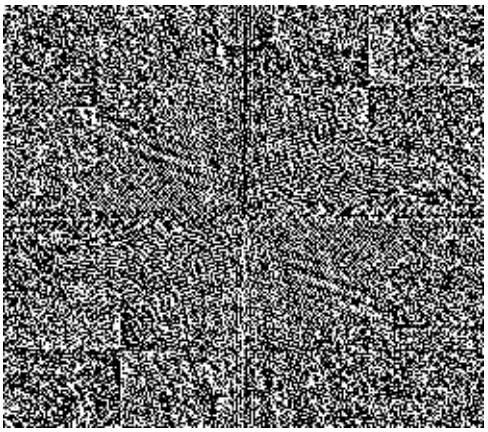
Out[44]=



In[45]:=
```
abs = ImageData[  ];
```

In[46]:=
```
arg = Arg@fw;
Labeled[Image[arg], Style["Phase spectrum", 14]]
```

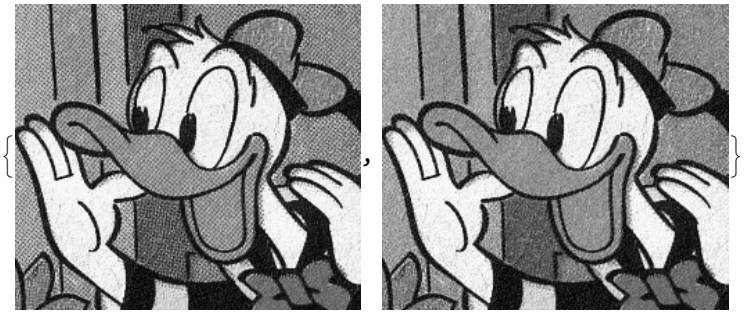
Out[47]=



Phase spectrum

In[48]:=
```
{Image[img, ImageSize → Small],
 Chop[Abs@InverseFourier[(abs E^(I arg))]] // Image[#, ImageSize → Small] &}
```

Out[48]=



example 2

In[49]:=
```
ClearAll["Global`*"]
```

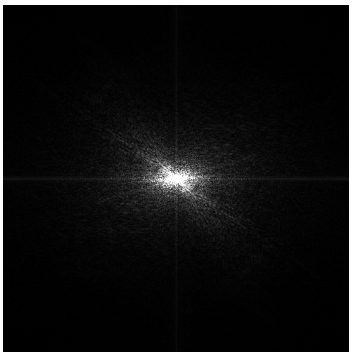
In[50]:=
```
img = ColorConvert[ , "Grayscale"]
```

Out[50]=

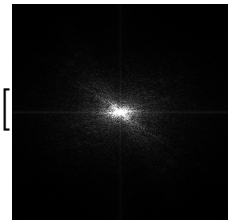
In[51]:=
```
data = ImageData[img];
{nRow, nCol} = Dimensions[data];
data = data * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
fw = Fourier[data, FourierParameters → {0, 1}];
```

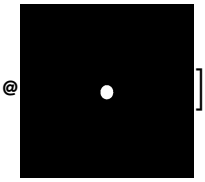
In[55]:=
```
abs = Abs@fw;
Image[abs, ImageSize → Small]
```

Out[56]=



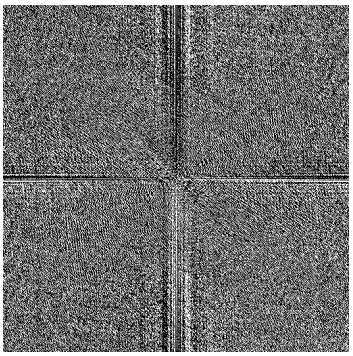
In[57]:=
```
abs = ImageData[ ColorNegate@ ];
```





In[58]:=
```
arg = Arg@fw;
Labeled[Image[arg, ImageSize → Small], Style["Phase spectrum", 14]]
```

Out[59]=



Phase spectrum

In[60]:=
```
{Image[img, ImageSize → Small],
 Chop[Abs@InverseFourier[(abs E^(I arg))]] // Image[#, ImageSize → Small] &}
```

Out[60]=

another way

In[63]:=
```
ClearAll["Global`*"]
```

In[64]:=
```
image = Image[ColorConvert[ , "Grayscale"], ImageSize → Small]
```

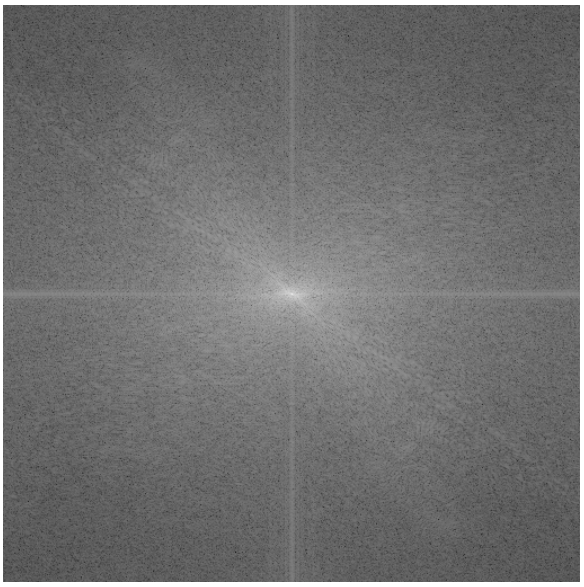
Out[64]=

In[65]:=
```
data = ImageData[image];
{nRow, nCol} = Dimensions[data];
d = data;
d = d * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
fw = Fourier[d, FourierParameters → {0, 1}];

(* below is for making masks *)
```

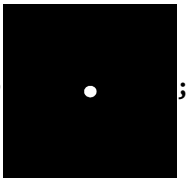
In[70]:=
```
absvis = Rescale@Log[Abs@fw];
Image[absvis / Max[absvis], ImageSize → 300]
```

Out[71]=



```
(* masks can be drawn or given as array of booleans *)
```

In[72]:=
```
mask = 1 - ImageData@ ;
```



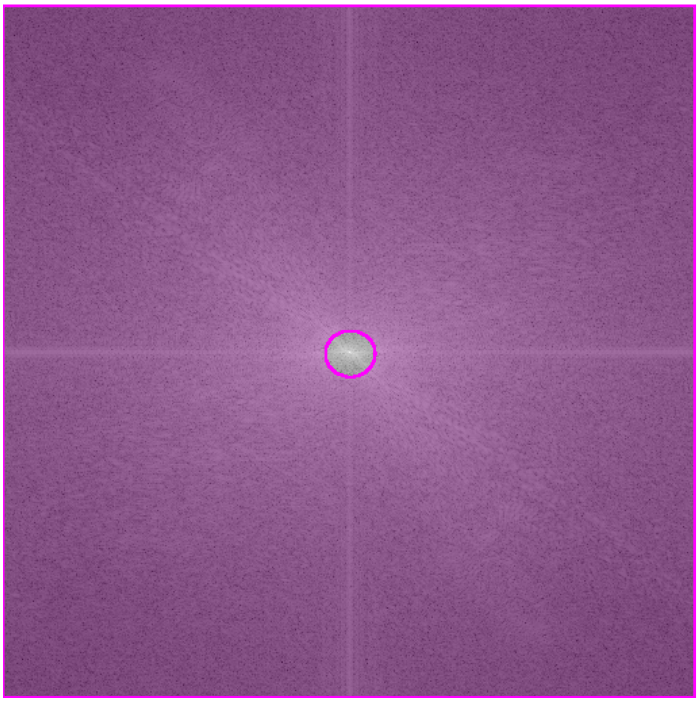
In[73]:=
```
fft = Fourier[data];
fftRotated = RotateLeft[fft, Floor[Dimensions[fft] / 2]];
```

In[75]:=
```
Dimensions /@ {fft, fftRotated}
```

Out[75]=
```
{{512, 512}, {512, 512}}
```

In[76]:=
```
HighlightImage[Image[absvis], Image@mask, ImageSize → Medium]
```

Out[76]=



In[77]:=
```
fftMasked = RotateRight[mask * fftRotated, Floor[Dimensions[fft] / 2]];
```

In[78]:=
```
Image[Chop@Abs@InverseFourier[fftMasked]]
```

Out[78]=

```
(* or we can now do operations on fw with masks *)
```

In[79]:=
```
ifw = Chop@InverseFourier[RotateRight[fw * mask, Floor[Dimensions[fft] / 2]]];
GraphicsRow[{Image[Abs@ifw], Image[data]}, ImageSize → 800]
```

Out[80]=

3. surgery on fourier coefficients

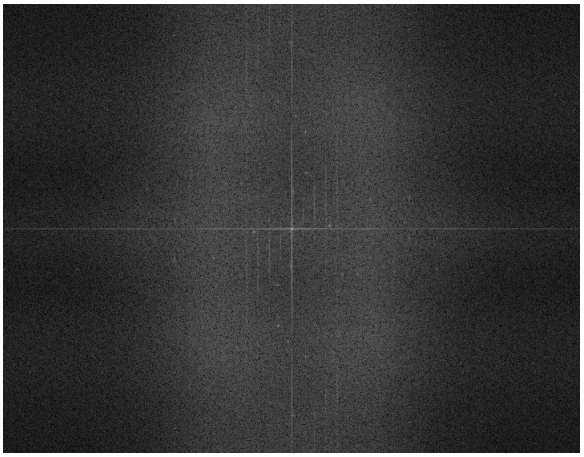
In[81]:=
```
ClearAll["Global`*"]
```

In[82]:=
```
image =
```

```
;
```

In[91]:=
```
data = ImageData[image];(*get data*)
{nRow, nCol, nChannel} = Dimensions[data];
d = data[[All, All, 2]];
d = d * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
fw = Fourier[d, FourierParameters → {1, 1}];
fudgeFactor = 50;
abs = fudgeFactor * Log[1 + Abs@fw];
Labeled[a = Image[abs / Max[abs], ImageSize → 300], Style["Magnitude spectrum", 18]]
```

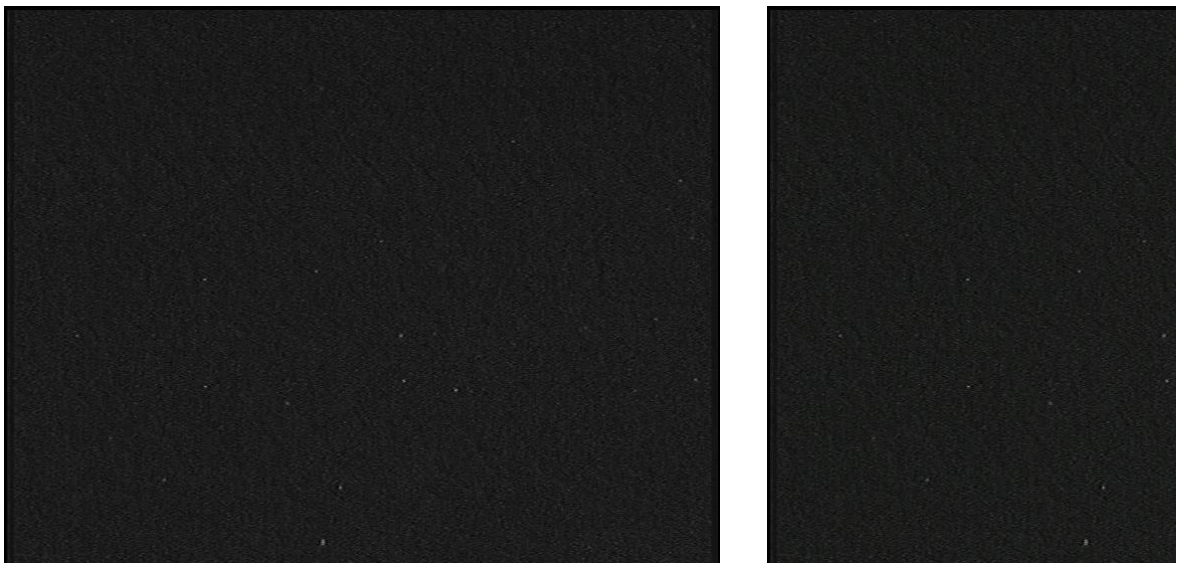
Out[98]=



Magnitude spectrum

In[99]:=
```
ifw = Chop@InverseFourier[fw, FourierParameters → {1, 1}];
ifw = ifw * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
idata = Map[# &, ifw, {2}];
GraphicsRow[{Image[idata], Image[data]}, ImageSize → 800]
```

Out[102]=



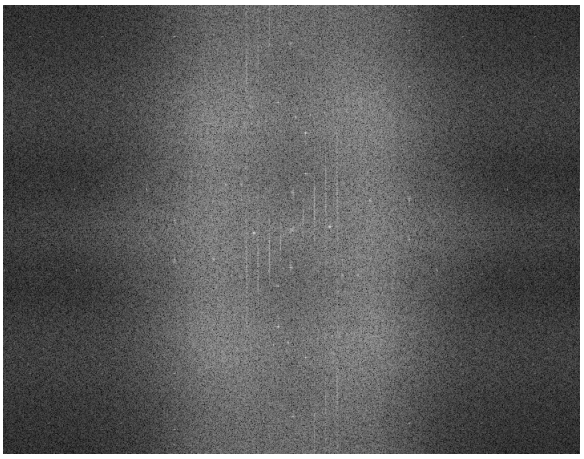
In[ ]:=
```
Dimensions[fw]
```

Out[ ]=
```
{560, 720}
```

In[103]:=
```
fw2 = fw;
fw2[[All, 361]] = Mean /@ fw2[[All, {360, 362}]];
fw2[[281]] = Mean@fw2[[{280, 282}]];
```

In[106]:=
```
fudgeFactor = 50;
abs = fudgeFactor * Log[1 + Abs@fw2];
Labeled[a = Image[abs / Max[abs], ImageSize → 300], Style["Magnitude spectrum", 18]]
```

Out[108]=



Magnitude spectrum

In[109]:=
```
ifw = Chop@InverseFourier[fw2, FourierParameters → {1, 1}];
ifw = ifw * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
idata = Map[{#, #, #} &, ifw, {2}];
GraphicsRow[{Image[idata], Image[data]}, ImageSize → 1200]
```

Out[112]=

```
(* done using all three channels *)
```

In[113]:=
```
{nRow, nCol, nChannel} = Dimensions[data];
d = data[[All, All, #]] & /@ Range[nChannel];
idata = Module[{d, fw, fw2, ifw}, d = # * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
      fw = Fourier[#, FourierParameters → {1, 1}];
      fw2 = fw;
      fw2[[All, 361]] = Mean /@ fw2[[All, {360, 362}]];
      fw2[[281]] = Mean@fw2[[{280, 282}]];
      ifw = Chop@InverseFourier[fw2, FourierParameters → {1, 1}];
      ifw = ifw * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
      ifw] & /@ d;
idata = Transpose[idata, {3, 1, 2}];
Image[idata]
```

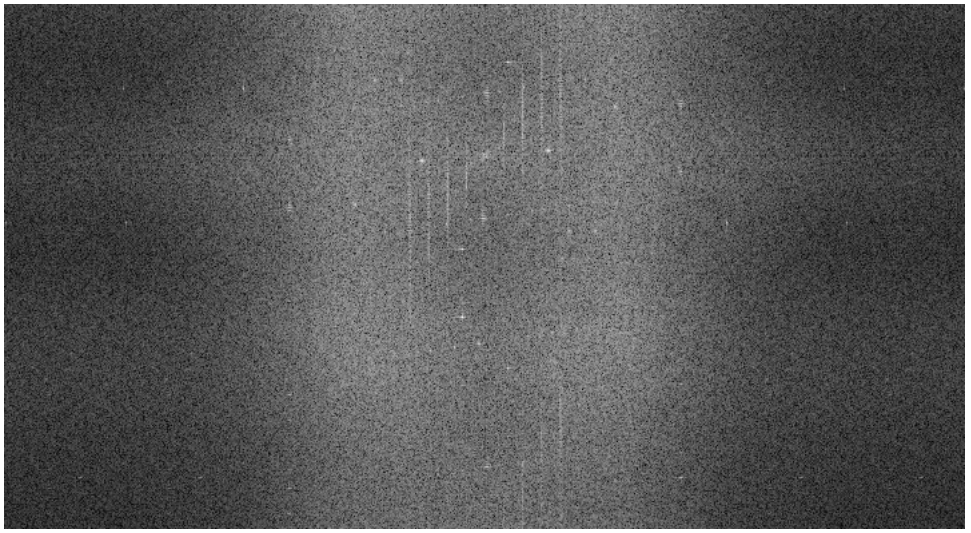
Out[117]=

In[ ]:=
```
(*using manipulate to predict which row and column to remove - using channel 2*)
```

In[118]:= `img =`  `;`

```
data = ImageData[img];(*get data*)
{nRow, nCol, nChannel} = Dimensions[data];
d = data[[All, All, 2]];
d = d * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
fw = Fourier[d, FourierParameters → {1, 1}];
Manipulate[Module[{fw2, fudgeFactor, abs, ifw, idata, message}, fw2 = fw;
  If[remcol, fw2[[All, column]] = Mean /@ fw2[[All, {column - 1, column + 1}]]];
  If[remrow, fw2[[row]] = Mean@fw2[[{row - 1, row + 1}]]];
  fudgeFactor = 50;
  abs = fudgeFactor * Log[1 + Abs@fw2];
  ifw = InverseFourier[fw2, FourierParameters → {1, 1}];
  message = "";
  If[Max@Abs@Im@ifw / Max@Abs@Re@ifw > 10^-10,
   message = "Warning, Inverse FFT returned complex
       results,\n proceeding with real part only"];
  ifw = Re@ifw;
  ifw = ifw * (-1)^Table[i + j, {i, nRow}, {j, nCol}];
  idata = Map[{#, #, #} &, ifw, {2}];
  Row[
   {Labeled[Image[abs / Max[abs], ImageSize → 500], Style["Magnitude spectrum", 18]],
    Labeled[Image[idata, ImageSize → 500], Style["Reconstructed image", 18]],
    message}]], {{remcol, False, "Remove Column?"}, {True, False}},
 {{remrow, False, "Remove Row?"}, {True, False}},
 {{row, Floor[nRow / 2]}, 2, nRow - 1, 1, Appearance → "Open"},
 {{column, Floor[nCol / 2]}, 2, nCol - 1, 1, Appearance → "Open"},
 ContinuousAction → False]
```

Out[124]=



Magnitude spectrum

Reconstructed image