```
% ### EXfourier2D.m ###     10.27.14
% Shows a dynamic reconstruction of an image stemming from
adding
% successively higher order terms from a 2D Fourier
transform
% [source: Richard Murray, York CVR]
% NOTE: This code reqs. the Image Processing Toolbox, plus
several
% subfunctions (defarg.m, matrt.m, and matxy.m;
Einstein.tif also reqd.)

clear;
% ------------
fileI= 'Einstein.tif';    % image to analyze
N = 4;  % set number of radial sections per step
rstep= 1/2;     % set radial step size going outward
(larger means smaller steps)
delay= 0.1; % animation delay [s]
Mmovie= 0;  % make a movie version (avi) (0-no,1-yes)
% ------------

im = double(imread(fileI)); % load image
imft = fftshift(fft2(im));  % compute 2D FFT
% +++
dim = size(im,1);
ftmax = max(abs(imft(:)));  % determine largest amplitude
(for scaling)
[r,t] = matrt([ dim dim ]); % make distance map (via
external function)
f = find(t<0);
t(f) = t(f)+pi;
t = t/pi;
rmap = floor(power(r,rstep)/.6);
dmap = rmap+t;
F = ceil(N*max(dmap(:)));

% +++
imC = uint8(127*[ im/255 500*abs(imft)/ftmax ]); % write
initial frame (whole image and FFT)
figure(1); clf; colormap(bone)
image(imC); drawnow;
```

```
% +++
% make movie as well
if Mmovie==1
    obj = VideoWriter('fourier_synth_einstein.avi');  %
open movie file
    open(obj); writeVideo(obj,imC);
end

% +++
% step through frames
figure(2); clf; colormap(bone);
for f=1:F
    mask = (dmap<(f/N));    % create low-pass filter mask
    imftmask = imft.*mask;  % filter image by applying mask
in freq. domain
    immask = real(ifft2(ifftshift(imftmask)))/255;  %
inverse Fourier transform
    imC = uint8(127*[ max(min(immask,1),0)
500*abs(imftmask)/ftmax ]);  % update image
    image(imC); drawnow;    pause(delay);
    if (Mmovie==1), writeVideo(obj,imC); end
end

if (Mmovie==1), close(obj); end     % close movie file
```