# Image Processing with Matlab

Nigel Allinson

Maria Pavlou

# Fast Fourier transform

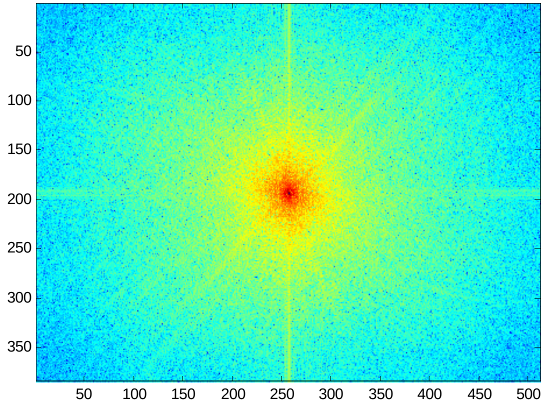
- The Fourier transform is a representation of a signal as a sum of complex exponentials of varying magnitudes, frequencies, and phases.

$$F(p,q) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j(2\pi/M)pm} e^{-j(2\pi/N)qn} \qquad \begin{array}{l} p = 0, 1, \ldots, M-1 \\ q = 0, 1, \ldots, N-1 \end{array}$$

$$f(m,n) = \frac{1}{MN} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} F(p,q) e^{j(2\pi/M)pm} e^{j(2\pi/N)qn} \qquad \begin{array}{l} m = 0, 1, \ldots, M-1 \\ n = 0, 1, \ldots, N-1 \end{array}$$

# FFT In Matlab

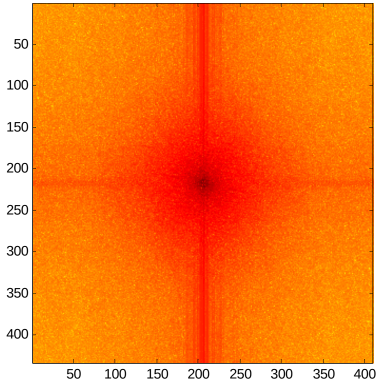
- Discreet Fourier Transform (DFT)
  - fft() % one-dimensional discreet FT
  - fft2() % two-dimensional discreet FT
  - ifft(),ifft2() % inverse transforms
  - fftshift(), ifftshift() % shift quadrants of DFT

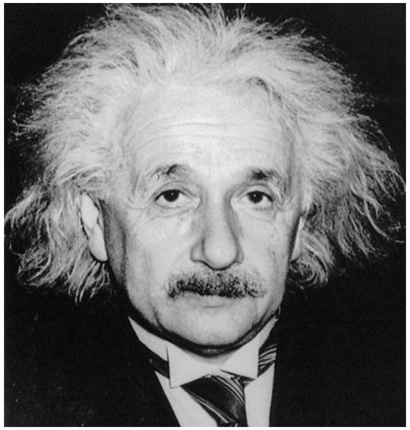


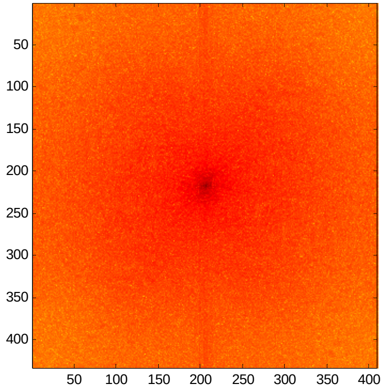

```
>> I = double(rgb2gray(imread('peppers.png')));
>> I = I - mean2(I);
>> I_fft = fftshift(fft2(I));
>> imagesc(log(abs(I_fft)));
>> colormap(jet); axis image;

>> IR = ifft2(I_fft);
>> imagesc(IR);
>> colormap(gray); axis image;
```

# Some fun with the FFT

- Albert and Marilyn and their DFTs



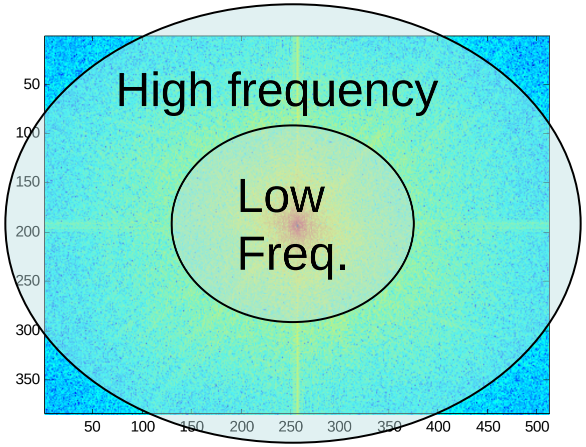



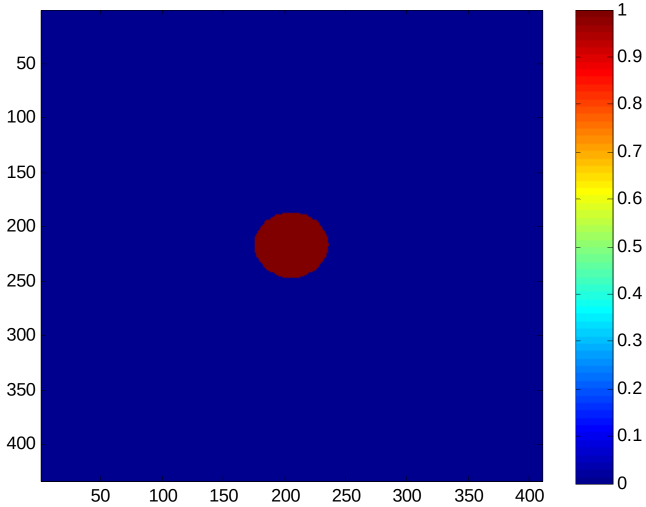



```
>>mm = double(imread('mm.jpg'));  % Read image and convert to double prec.
>>mm = mm - mean2(mm);            % Subtract the mean of the image

>>mmfft = fftshift(fft2(mm));      % Perform FFT and shift quadrants



>>figure; imagesc(log(abs(mmfft))); % Display FFT
>>colormap jet; axis image
```

# Some fun with the FFT

- Create a mask to filter out low and high frequencies

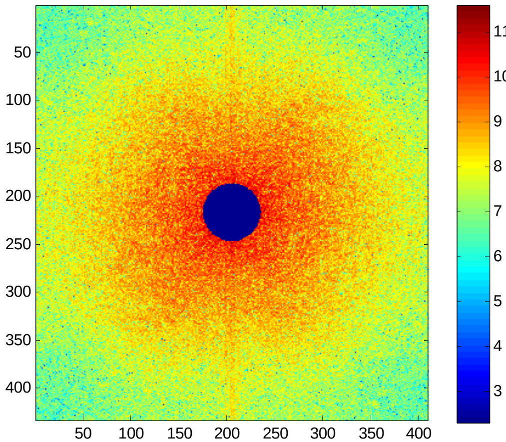


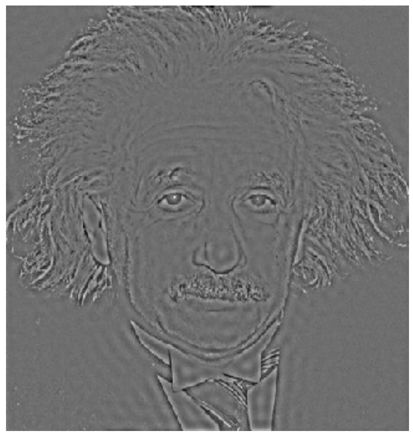



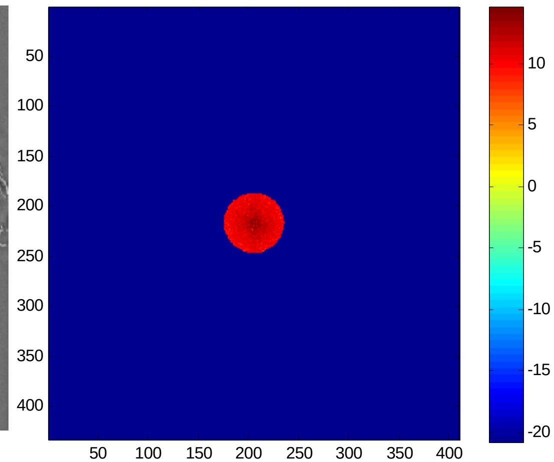

```
>> [X,Y] = pol2cart(-pi:pi/16:pi,25);   % Create points of a circle, radius 25
>> CentPoint = floor(size(aa)/2);       % Find center point of image
>> X = X + CentPoint(1); Y+CentPoint(2) % Shift circle to cent. of image
>> MASK = poly2mask(X,Y, size(aa,1),size(aa,2)); %Create mask
>> figure; imagesc(MASK)                % View the mask
```

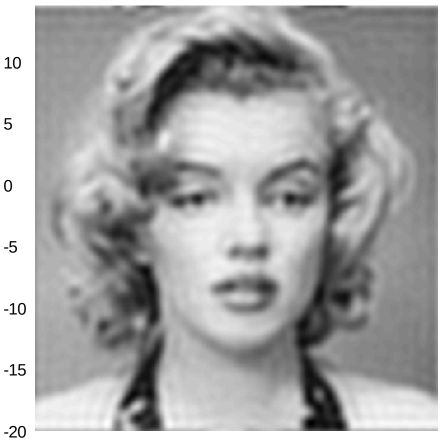
# Some fun with the FFT

- Mask Marilyn’s high frequency attributes and Albert’s low frequency attributes, and reconstruct.





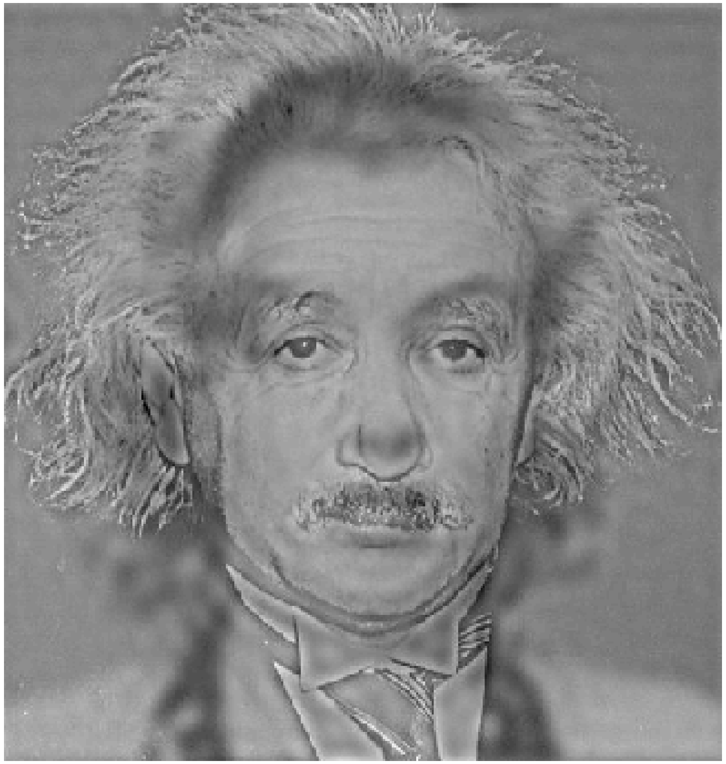



```
>> aafftmask = aafft .* (~MASK);             % Multiply fft with complement of mask(~)
>> mmfftmask = mmfft .* MASK;                % Multiply fft with mask
>> figure; imagesc(log(abs(mmfftmask))); colormap jet; axis image;
>> figure; imagesc(log(abs(aafftmask))); colormap jet; axis image;
>> aahigh = ifft2(ifftshift(aafftmask));      % Reconstrcut with ifft2
>> mmlow = ifft2(ifftshift(mmfftmask));      % Reconstrcut with ifft2
>>figure; imagesc(real(mmlow)); colormap gray; axis image
>>figure; imagesc(real(aahigh)); colormap gray; axis image
```

# Some fun with the FFT

- Create a hybrid image with Marilyn’s high, and Albert’s low frequency attributes, and reconstruct.

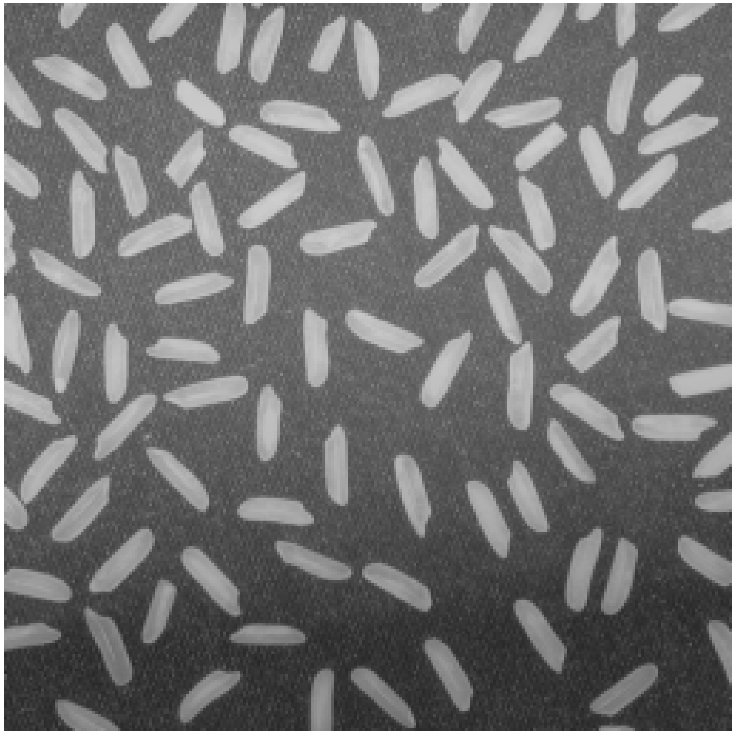
```
>> hybridfft = aafftmask + mmfftmask;
>> aammhybrid = ifft2(ifftshift(hybridfft));
>> figure; imagesc(real(aammhybrid));
>> colormap gray; axis image;
```



# Something more useful!?

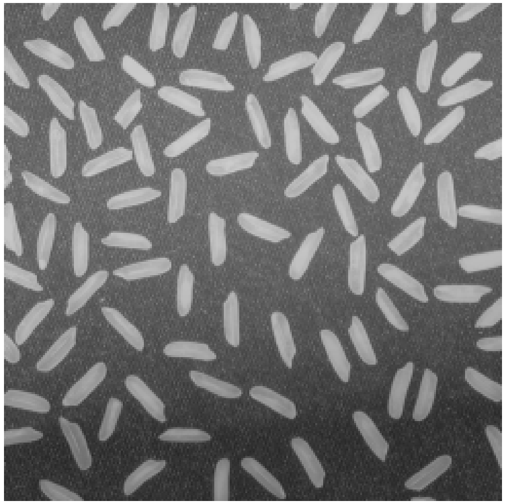
- Locating, counting and measuring attributes of rice grains in an image.
  - How many rice grains
  - Where are they
    - What is their average size
    - How regular is their shape

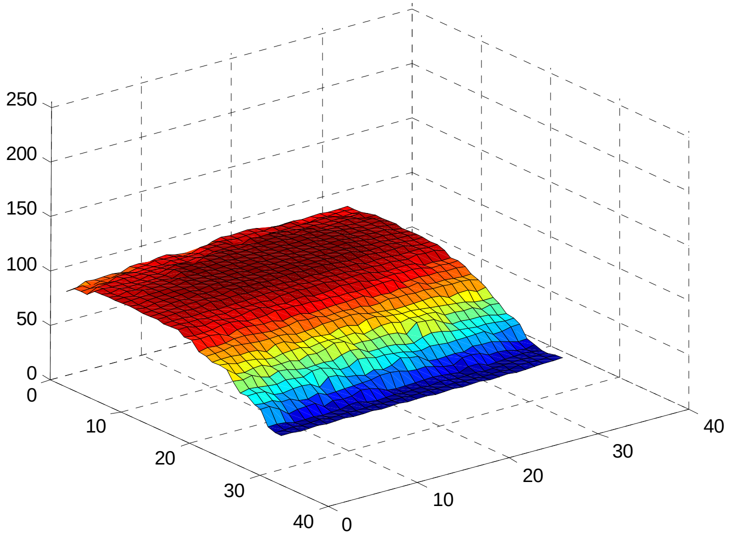
```
%% Step 1: Read Image
I = imread('rice.png');
imshow(I)
```



# Use Morphological Opening to Estimate the Background

- Notice that the background illumination is brighter in the center of the image than at the bottom.
- Use the IMOPEN function to estimate the background illumination.

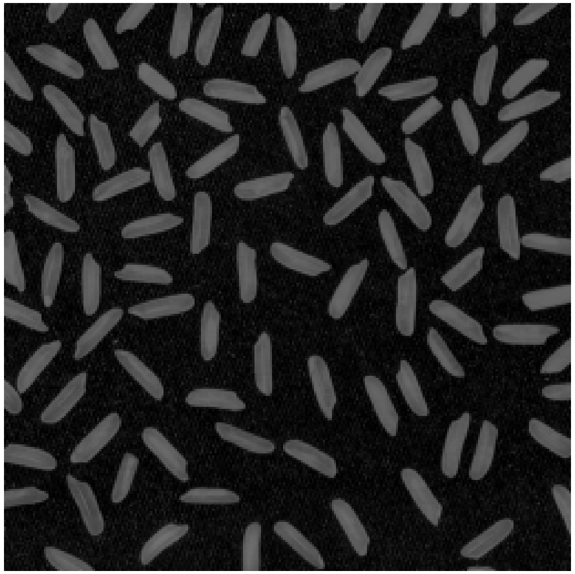





```
background = imopen(I,strel('disk',15));

% Display the Background Approximation as a Surface
figure, surf(double(background(1:8:end,1:8:end))),zlim([0 255]);
set(gca,'ydir','reverse');
```

# Subtract the Backround and Increase Contrast

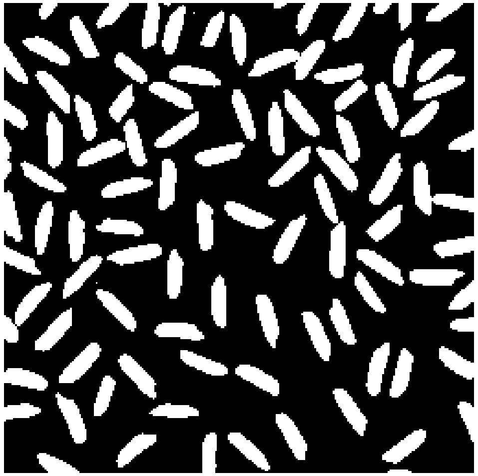
- Since the image and background are of class uint8, use the function IMSUBTRACT to subtract the background.

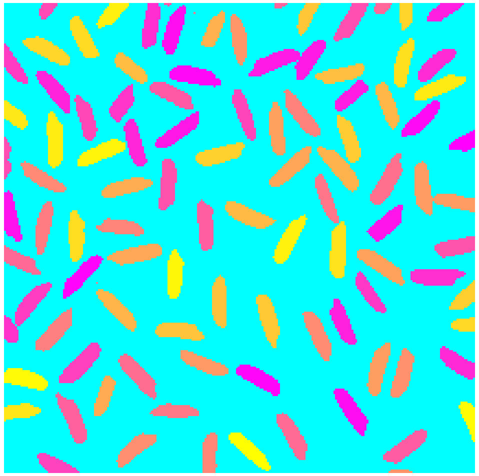


```
I2 = imsubtract(I,background);       % Subtract the background
imshow(I2)

I3 = imadjust(I2);                   % Increase image contrast
imshow(I3);
```

# Threshold the Image and Label Object in The Image

- Create a new binary image by thresholding the adjusted image.
- The function BWLABEL labels all connected component in the binary image.

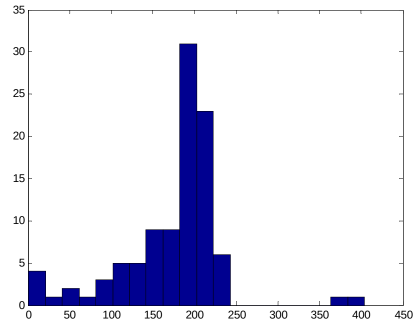




```
level = graythresh(I3);
bw = im2bw(I3,level);
imshow(bw)

[labeled,numObjects] = bwlabel(bw,4);           % 4-connected

RGB_label = label2rgb(labeled, @spring, 'c', 'shuffle');
imshow(RGB_label)
```

# Measure Object Properties in the Image

- The REGIONPROPS command measures object properties in an image.
- Compute Statistical Properties of Objects in the Image
- Create a Histogram of the Area of the Grains




```
graindata = regionprops(labeled,'basic') % Generate basic region properties

allgrains = [graindata.Area];           % Gather region area data into one vector
max_area = max(allgrains)               % Find the maximum area of all the grains.
biggrain = find(allgrains==max_area) % Find the grain number that has this area.
mean(allgrains)                          % Find the mean of the area of all the grains.

nbins = 20;
figure,hist(allgrains,nbins)            % Plot a histogram of the grain areas/size
```

# Matlab References

- Matlab Program Help
- MathWorks Online Documentation http://www.mathworks.com/access/helpdesk/help/helpdesk.html
- Numerical Methods with Matlab - Gerald Recktenwald
- Digital Image Processing Using MATLAB -Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins