Project 3: Correlation and Convolution in the Spatial Domain

Contents

Course No: ECE 5256

Due Date: 2/14/2021

Q1.) Consider two images. One should be relatively large, like 512 x 512 pixels

that contains several sparsely populated points with magnitude 255, and the other is small such as 15 x 15 that contains a non-symmetric object like, with the object white and the background black.

```
%Perform both the correlation, and convolution of the two images separately
%and correctly label each result. Make the resulting image the same size as
%the larger input image.
```

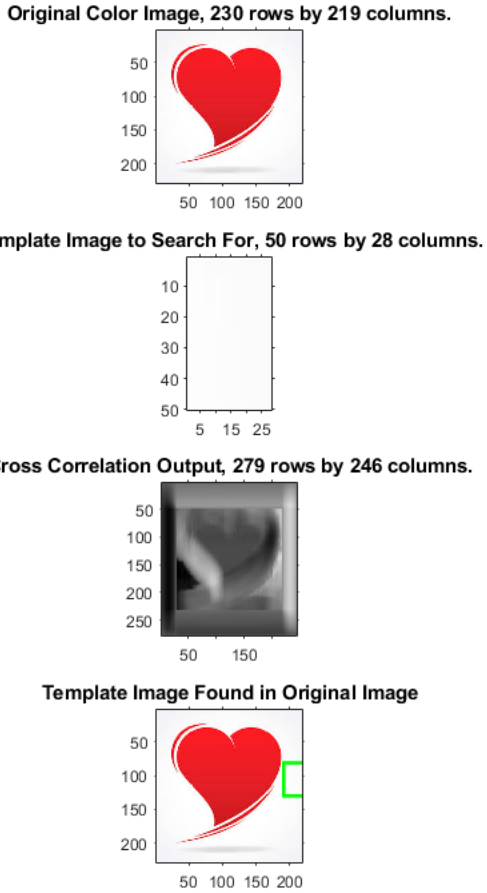
Ans 1) To solve this question, I am using the normxcorr function. As the name suggest, normxcorr2 function computes cross correlation of matrices.

Use normxcorr2 to find a template on Image 1(Asymetric Heart)of pixel 230x219x3.

```
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
format long g;
format compact;
fontSize = 11;

% Read in a standard MATLAB color demo image.
folder = fullfile(matlabroot, '\toolbox\images\imdemos');
Image_FileName_1 = 'Asym heart small.jpg';

% Get the full filename, with path prepended.
ImageFileName = fullfile(folder, Image_FileName_1);
if ~exist(ImageFileName, 'file')
        % Didn't find it there.  Check the search path for it.
        ImageFileName = Image_FileName_1; % No path this time.
        if ~exist(ImageFileName, 'file')
                % Still didn't find it.  Alert user.
                errorMessage = sprintf('Error: %s does not exist.', ImageFileName);
                uiwait(warndlg(errorMessage));
                return;
        end
end


rgbImage = imread(ImageFileName);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(rgbImage);


% Display the original color image.
subplot(4, 1, 1);
imshow(rgbImage, []);
axis on;
caption = sprintf('Original Color Image, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);
% Enlarge figure to full screen.
set(gcf, 'units','normalized','outerposition',[0, 0, 1, 1]);

% Let's get our template by extracting a small portion of the original image.
```

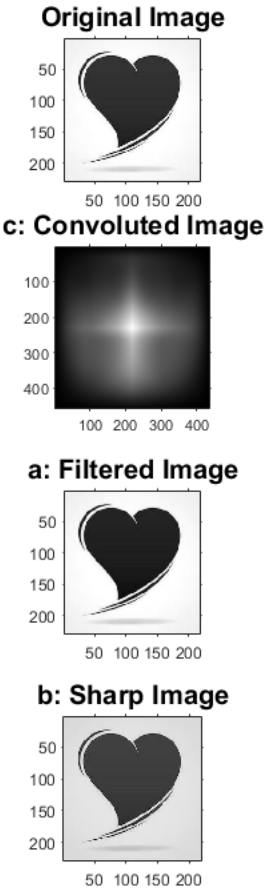
```
templateWidth = 71
templateHeight = 49
smallSubImage = imcrop(rgbImage, [192, 82, templateWidth, templateHeight]);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(smallSubImage);
subplot(4, 1, 2);
imshow(smallSubImage, []);
axis on;
caption = sprintf('Template Image to Search For, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);

% Ask user which channel (red, green, or blue) to search for a match.
% channelToCorrelate = menu('Correlate which color channel?', 'Red', 'Green', 'Blue');
% It actually finds the same location no matter what channel you pick,
% for this image anyway, so let's just go with red (channel #1).
% Note: If you want, you can get the template from every color channel and search for it in every color channel,
% then take the average of the found locations to get the overall best location.

% Using normxCorr2 Function:
channelToCorrelate = 1;  % Use the red channel.
correlationOutput = normxcorr2(smallSubImage(:,:,1), rgbImage(:,:, channelToCorrelate));
subplot(4, 1, 3);
imshow(correlationOutput, []);
axis on;
% Get the dimensions of the image.  numberOfColorBands should be = 1.
[rows, columns, numberOfColorBands] = size(correlationOutput);
caption = sprintf('Cross Correlation Output, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);

% Find out where the normalized cross correlation image is brightest.
[maxCorrValue, maxIndex] = max(abs(correlationOutput(:)));
[yPeak, xPeak] = ind2sub(size(correlationOutput),maxIndex(1))
% Because cross correlation increases the size of the image,
% we need to shift back to find out where it would be in the original image.
corr_offset = [(xPeak-size(smallSubImage,2)) (yPeak-size(smallSubImage,1))]

% Plot it over the original image.
subplot(4, 1, 4); % Re-display image in lower right.
imshow(rgbImage);
axis on; % Show tick marks giving pixels
hold on; % Don't allow rectangle to blow away image.
% Calculate the rectangle for the template box.  Rect = [xLeft, yTop, widthInColumns, heightInRows]
boxRect = [corr_offset(1) corr_offset(2) templateWidth, templateHeight]
% Plot the box over the image.
rectangle('position', boxRect, 'edgecolor', 'g', 'linewidth',2);
% Give a caption above the image.
title('Template Image Found in Original Image', 'FontSize', fontSize);
```

```
templateWidth =
    71
templateHeight =
    49
yPeak =
   131
xPeak =
   219
corr_offset =
   191    81
boxRect =
   191    81    71    49
```



Convolution applied to Asymetric small heart image

```
grayImage = double(rgb2gray(imread('Asym heart small.jpg')));
grayImage = double(grayImage);
subplot(4,1,1);
imshow(grayImage, []);
axis on;

title('Original Image', 'FontSize', 15);
myfilter = fspecial('gaussian',[3 3], 0.5);
subplot(4,1,2);
imshow(myfilter, []);
axis on;

title('Gaussian Filter applied on Image', 'FontSize', 15);
a = imfilter(grayImage, myfilter);
subplot(4,1,3);
imshow(a, []);
axis on;

title('a: Filtered Image', 'FontSize', 15);
b = imsharpen(grayImage,'Radius',0.5);
subplot(4,1,4);
imshow(b, []);
axis on;

title('b: Sharp Image', 'FontSize', 15);
c=conv2(a,b, 'full');
subplot(4,3,5);
imshow(c, []);
title('c: Convoluted Image', 'FontSize', 15);
axis on;
```

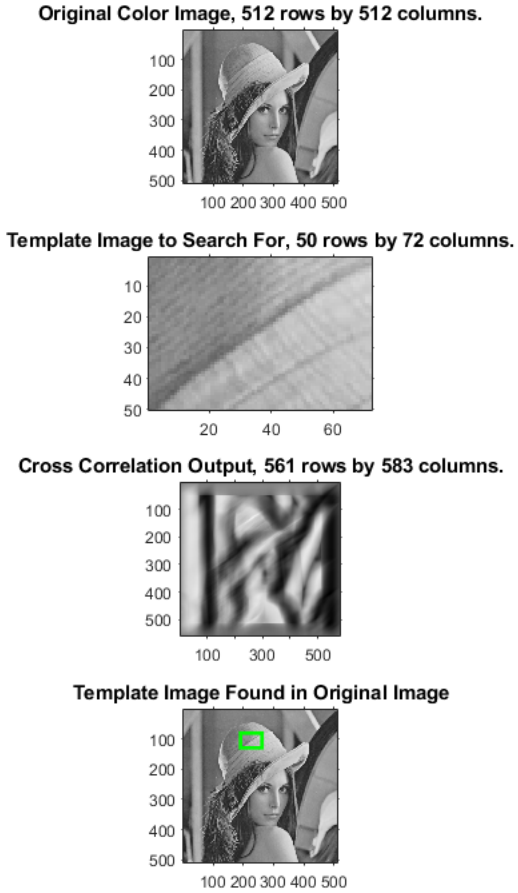


Use normxcorr2 to find a template on Image 2(Lena.png) of pixel 512x512

```
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
format long g;
format compact;
fontSize = 11;

% Read in a standard MATLAB color demo image.
folder = fullfile(matlabroot, '\toolbox\images\imdemos');
Image_FileName_2 = 'Lena.png';

% Get the full filename, with path prepended.
ImFileName = fullfile(folder, Image_FileName_2);
if ~exist(ImFileName, 'file')
        % Didn't find it there.  Check the search path for it.
        ImFileName = Image_FileName_2; % No path this time.
        if ~exist(ImFileName, 'file')
                % Still didn't find it.  Alert user.
                errorMessage = sprintf('Error: %s does not exist.', ImFileName);
                uiwait(warndlg(errorMessage));
                return;
        end
end


rgbImage = imread(ImFileName);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(rgbImage);


% Display the original color image.
subplot(4, 1, 1);
imshow(rgbImage, []);
axis on;
caption = sprintf('Original Color Image, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);
% Enlarge figure to full screen.
```

```
set(gcf, 'units','normalized','outerposition',[0, 0, 1, 1]);

% Let's get our template by extracting a small portion of the original image.
templateWidth = 71
templateHeight = 49
smallSubImage = imcrop(rgbImage, [192, 82, templateWidth, templateHeight]);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(smallSubImage);
subplot(4, 1, 2);
imshow(smallSubImage, []);
axis on;
caption = sprintf('Template Image to Search For, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);

% Ask user which channel (red, green, or blue) to search for a match.
% channelToCorrelate = menu('Correlate which color channel?', 'Red', 'Green', 'Blue');
% It actually finds the same location no matter what channel you pick,
% for this image anyway, so let's just go with red (channel #1).
% Note: If you want, you can get the template from every color channel and search for it in every color channel,
% then take the average of the found locations to get the overall best location.

% Using normxCorr2 Function:
channelToCorrelate = 1;  % Use the red channel.
correlationOutput = normxcorr2(smallSubImage(:,:,1), rgbImage(:,:, channelToCorrelate));
subplot(4, 1, 3);
imshow(correlationOutput, []);
axis on;
% Get the dimensions of the image.  numberOfColorBands should be = 1.
[rows, columns, numberOfColorBands] = size(correlationOutput);
caption = sprintf('Cross Correlation Output, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);

% Find out where the normalized cross correlation image is brightest.
[maxCorrValue, maxIndex] = max(abs(correlationOutput(:)));
[yPeak, xPeak] = ind2sub(size(correlationOutput),maxIndex(1))
% Because cross correlation increases the size of the image,
% we need to shift back to find out where it would be in the original image.
corr_offset = [(xPeak-size(smallSubImage,2)) (yPeak-size(smallSubImage,1))]

% Plot it over the original image.
subplot(4, 1, 4); % Re-display image in lower right.
imshow(rgbImage);
axis on; % Show tick marks giving pixels
hold on; % Don't allow rectangle to blow away image.
% Calculate the rectangle for the template box.  Rect = [xLeft, yTop, widthInColumns, heightInRows]
boxRect = [corr_offset(1) corr_offset(2) templateWidth, templateHeight]
% Plot the box over the image.
rectangle('position', boxRect, 'edgecolor', 'g', 'linewidth',2);
% Give a caption above the image.
title('Template Image Found in Original Image', 'FontSize', fontSize);
```

```
templateWidth =
    71
templateHeight =
    49
yPeak =
   131
xPeak =
   263
corr_offset =
   191    81
boxRect =
   191    81    71    49
```



Convolution applied Lena Image Of size 512x512

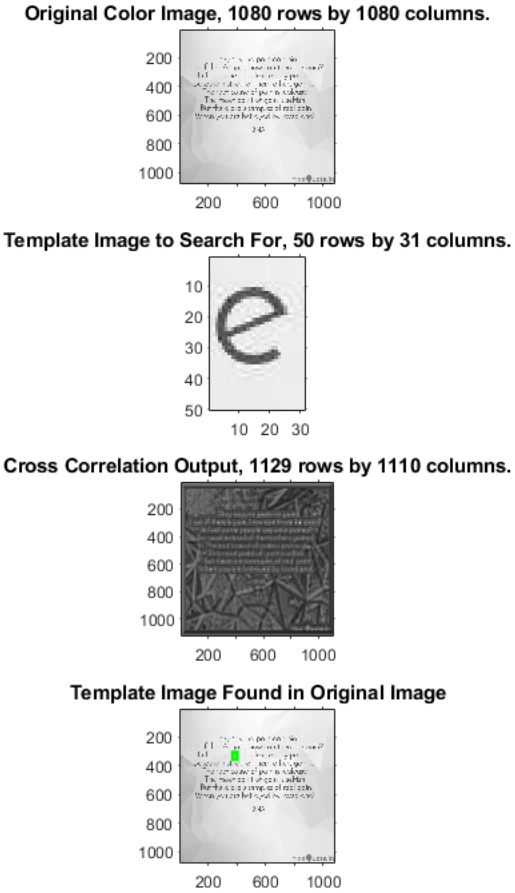
```
grayImage_2 = double(rgb2gray(imread('Lena.png')));
grayImage_2 = double(grayImage_2);
subplot(2,3,1);
imshow(grayImage_2, []);
axis on;
title('Original Image', 'FontSize', 15);

% Applying a gaussian Filter
myfilter = fspecial('gaussian',[3 3], 0.5);
subplot(4,1,2);
imshow(myfilter, []);
axis on;
title('Gaussian Filter applied on Image', 'FontSize', 15);

a = imfilter(grayImage_2, myfilter);
subplot(4,1,3);
imshow(a, []);
axis on;
title('a: Filtered Image', 'FontSize', 15);

b = imsharpen(grayImage_2,'Radius',0.5);
subplot(4,1,4);
imshow(b, []);
axis on;
title('b: Sharp Image', 'FontSize', 15);

%'full' returns the full 2D convolution of the image.
c=conv2(a,b, 'full');
subplot(4,3,5);
imshow(c, []);
title('c: Convoluted Image', 'FontSize', 15);
axis on;
```

Q2) Create or download an image of several lines of text. Don't make the text too small, and make the text lighter than the background. Extract one character from the text image and create a small image from it. Perform the correlation between the two images with the result the same size as the larger image. Note the position(s) of the maxima.

```
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
format long g;
format compact;
fontSize = 11;

% Read in a standard MATLAB color demo image.
folder = fullfile(matlabroot, '\toolbox\images\imdemos');
Text_FileName = 'Text.jpg';

% Get the full filename, with path prepended.
TextFileName = fullfile(folder, Text_FileName);
if ~exist(TextFileName, 'file')
        % Didn't find it there.  Check the search path for it.
        TextFileName = Text_FileName; % No path this time.
        if ~exist(TextFileName, 'file')
                % Still didn't find it.  Alert user.
                errorMessage = sprintf('Error: %s does not exist.', TextFileName);
                uiwait(warndlg(errorMessage));
                return;
        end
end


rgbImage = imread(TextFileName);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(rgbImage);


% Display the original color image.
subplot(4, 1, 1);
imshow(rgbImage, []);
axis on;
caption = sprintf('Original Color Image, %d rows by %d columns.', rows, columns);
```

```
title(caption, 'FontSize', fontSize);
% Enlarge figure to full screen.
set(gcf, 'units','normalized','outerposition',[0, 0, 1, 1]);

% Let's get our template by extracting a small portion of the original
% image. Here I am trying to detect character 'e'
templateWidth = 30
templateHeight = 49
smallSubImage = imcrop(rgbImage, [370, 305, templateWidth, templateHeight]);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(smallSubImage);
subplot(4, 1, 2);
imshow(smallSubImage, []);
axis on;
caption = sprintf('Template Image to Search For, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);

% Ask user which channel (red, green, or blue) to search for a match.
% channelToCorrelate = menu('Correlate which color channel?', 'Red', 'Green', 'Blue');
% It actually finds the same location no matter what channel you pick,
% for this image anyway, so let's just go with red (channel #1).
% Note: If you want, you can get the template from every color channel and search for it in every color channel,
% then take the average of the found locations to get the overall best location.

% Using normxCorr2 Function:
channelToCorrelate = 1;  % Use the red channel.
correlationOutput = normxcorr2(smallSubImage(:,:,1), rgbImage(:,:, channelToCorrelate));
subplot(4, 1, 3);
imshow(correlationOutput, []);
axis on;
% Get the dimensions of the image.  numberOfColorBands should be = 1.
[rows, columns, numberOfColorBands] = size(correlationOutput);
caption = sprintf('Cross Correlation Output, %d rows by %d columns.', rows, columns);
title(caption, 'FontSize', fontSize);

% Find out where the normalized cross correlation image is brightest.
[maxCorrValue, maxIndex] = max(abs(correlationOutput(:)));
[yPeak, xPeak] = ind2sub(size(correlationOutput),maxIndex(1))
% Because cross correlation increases the size of the image,
% we need to shift back to find out where it would be in the original image.
corr_offset = [(xPeak-size(smallSubImage,2)) (yPeak-size(smallSubImage,1))]

% Plot it over the original image.
subplot(4, 1, 4); % Re-display image in lower right.
imshow(rgbImage);
axis on; % Show tick marks giving pixels
hold on; % Don't allow rectangle to blow away image.
% Calculate the rectangle for the template box.  Rect = [xLeft, yTop, widthInColumns, heightInRows]
boxRect = [corr_offset(1) corr_offset(2) templateWidth, templateHeight]
% Plot the box over the image.
rectangle('position', boxRect, 'edgecolor', 'g', 'linewidth',2);
% Give a caption above the image.
title('Template Image Found in Original Image', 'FontSize', fontSize);
```

```
templateWidth =
    30
templateHeight =
    49
yPeak =
   354
xPeak =
   400
corr_offset =
   369   304
boxRect =
   369   304    30    49
```

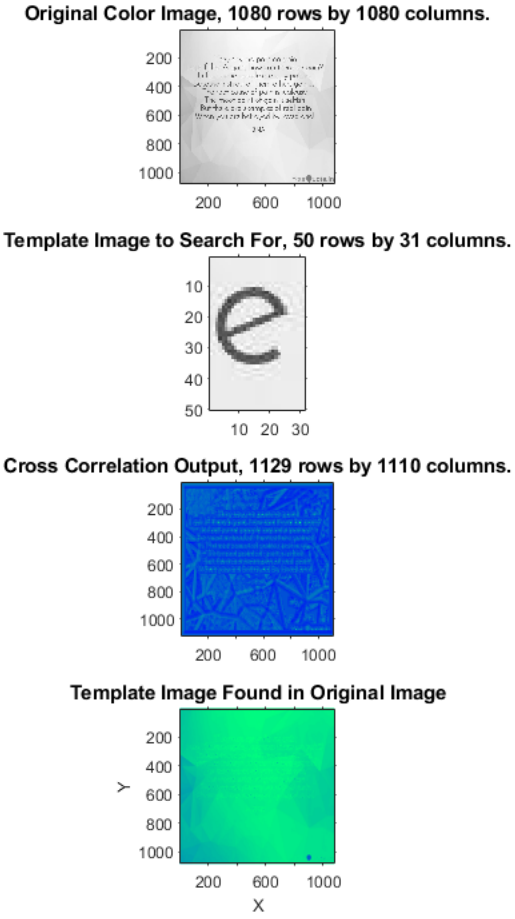


(a) Do they identify the characters in the text?

```
% No, the characters are not identified in the text.
```

(b) Determine the ratio of the correlation peak corresponding to a letter and the next highest (or highest peak).

```
Pprs3 = imread('Text.jpg');                 % Colour Image
Pprs1 = rgb2gray(Pprs3);                       % Grayscale Image
x = 0:size(Pprs1,2)-1;
y = 0:size(Pprs1,1)-1;
[X,Y] = meshgrid(x,y);                         % Coordinate Matrices (Not Necessary)
figure(1)
meshc(X, Y, Pprs1)                             % Mesh Plot
grid off
xlabel('X')
ylabel('Y')
zlabel('Intensity')
colormap(winter)
```

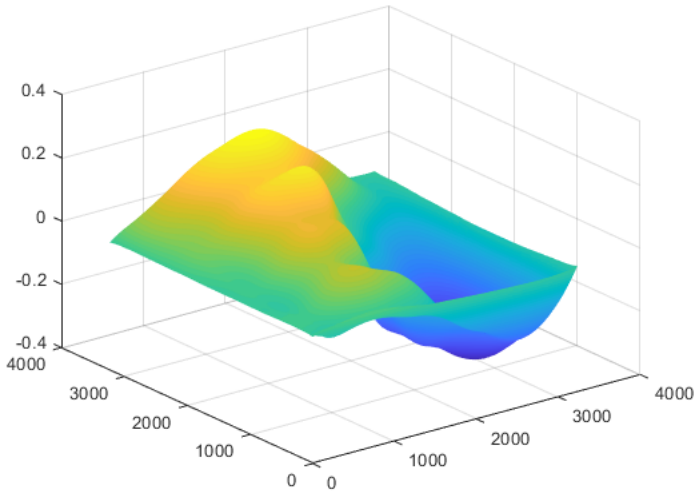


(c) Normalize the correlation result by dividing it by an image created from the convolution of image of the text and a kernel the same size as your one character image, but the kernel contains all 1's.

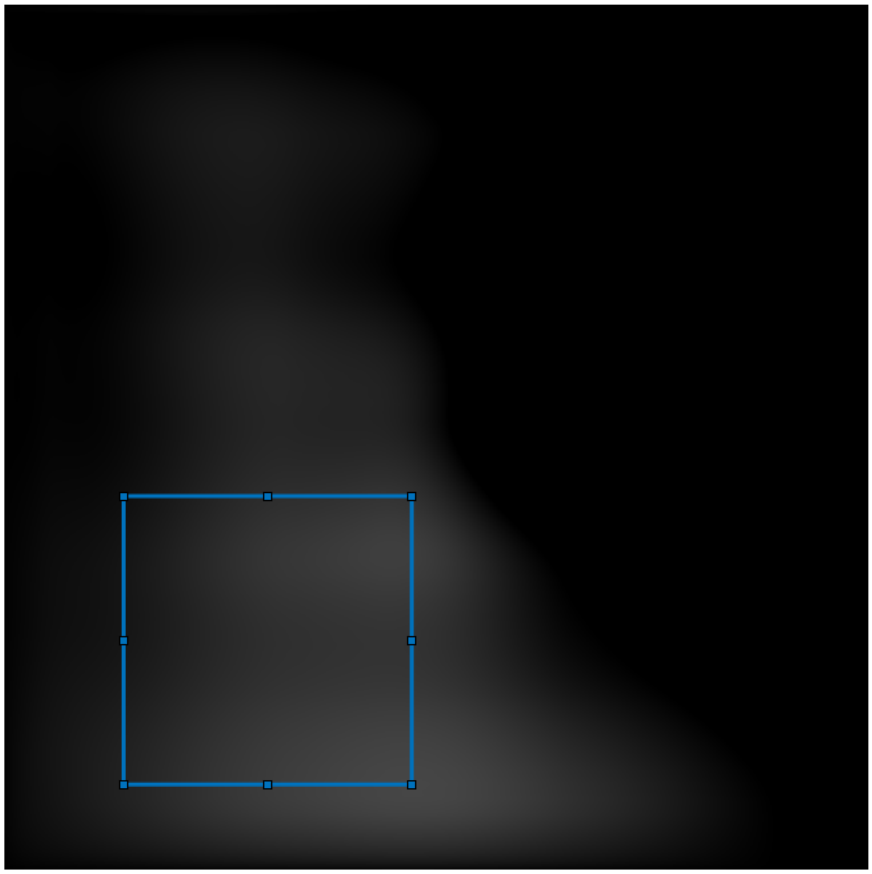
```
% Convolution of Text Image:
grayTextImage = double(rgb2gray(imread('Text.jpg')));
grayTextImage = double(grayTextImage);
subplot(4,1,1);
imshow(grayTextImage, []);
axis on;
title('Original Image', 'FontSize', 15);

% Applying a gaussian Filter
myfilter = fspecial('gaussian',[3 3], 0.5);
subplot(4,1,2);
imshow(myfilter, []);
axis on;
title('Gaussian Filter applied on Image', 'FontSize', 15);

a = imfilter(grayTextImage, myfilter);
subplot(4,1,3);
imshow(a, []);
axis on;
title('a: Filtered Image', 'FontSize', 15);

b = imsharpen(grayTextImage,'Radius',0.5);
subplot(4,1,4);
imshow(b, []);
axis on;
title('b: Sharp Image', 'FontSize', 15);

%'full' returns the full 2D convolution of the image.
c_text=conv2(a,b, 'full');
subplot(4,3,5);
imshow(c_text, []);
title('c: Convoluted Image', 'FontSize', 15);
axis on;
```

Computing the normalized corelation of the image.

```
I  = rgb2gray(imread('Text.jpg'));

% Perform cross correlation and display the result as a surface
c_corr = normxcorr2(I,c_text);
figure;imshow(c_corr);
figure, surf(c_corr), shading flat
```




```
[ypeak, xpeak] = find(c_corr==max(c_corr(:)));
% Compute translation from max location in correlation matrix
yoffSet = ypeak-size(I,1);
xoffSet = xpeak-size(I,2);

% Display the matched area by using the drawrectangle function.
imshow(c_corr)
drawrectangle(gca,'Position',[xoffSet,yoffSet,size(I,2),size(I,1)], ...
    'FaceAlpha',0);
% Display matched area
% figure
% hAx  = axes;
% imrect(hAx, [xoffSet+1, yoffSet+1, size(I,2), size(I,1)]);
```



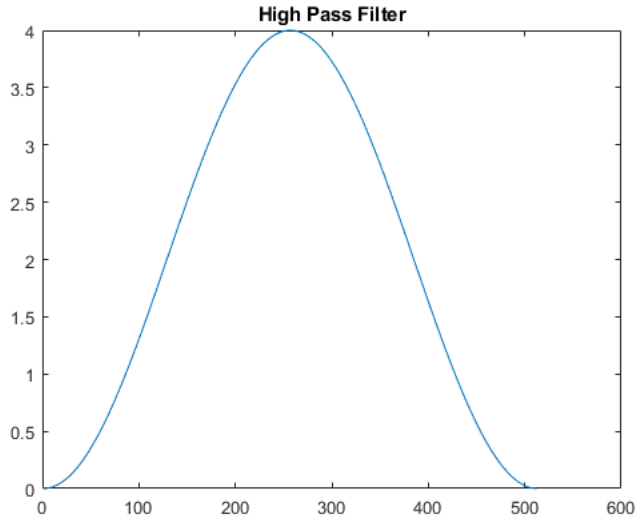
(d) What is the new ratio of the correlation peak corresponding to a letter and the next highest peak.

```
%Pprs3 = imread('Text.jpg');                % Colour Image
Pprs_t = c_corr;
x = 0:size(Pprs_t,2)-1;
y = 0:size(Pprs_t,1)-1;
[X,Y] = meshgrid(x,y);                        % Coordinate Matrices (Not Necessary)
figure(1)
meshc(X, Y, Pprs_t)                           % Mesh Plot
grid off
xlabel('X')
ylabel('Y')
zlabel('Intensity')
colormap(winter)
```

3) Frequency response of spatial filtering

(a) Compare the frequency response in 1-D of the cross-section of two different spatial filters of size 1 x 3: all 1's [1 1 1], an approximation to the Laplacian [-1 2 -1]. Identify the differences.

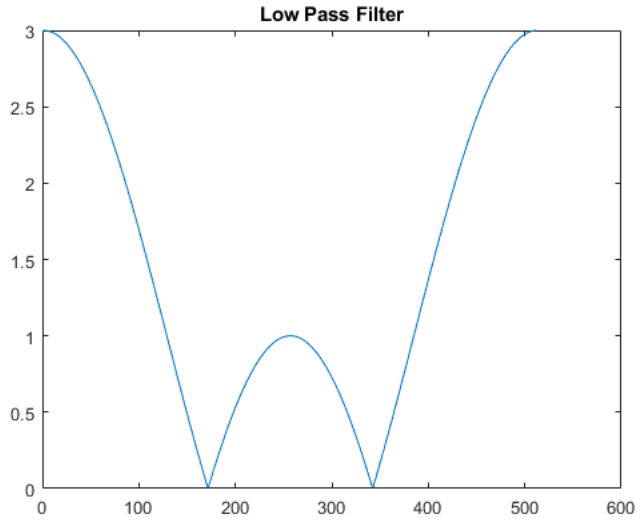
```
v2 = zeros(1,512);
v2(1)= -1;
v2(2) = 2;
v2(3) = -1;
plot(abs(fft(v2)));title("High Pass Filter");
```




```
v3 = zeros(1,512)
v3(1) = 1;
v3(2) = 1;
v3(3) = 1;
plot(abs(fft(v3)));title("Low Pass Filter");
```

```
v3 =
  Columns 1 through 13
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 14 through 26
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 27 through 39
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 40 through 52
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 53 through 65
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 66 through 78
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 79 through 91
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 92 through 104
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 105 through 117
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 118 through 130
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 131 through 143
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 144 through 156
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 157 through 169
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 170 through 182
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 183 through 195
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 196 through 208
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 209 through 221
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 222 through 234
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 235 through 247
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 248 through 260
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 261 through 273
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 274 through 286
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 287 through 299
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 300 through 312
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 313 through 325
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 326 through 338
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 339 through 351
     0     0     0     0     0     0     0     0     0     0     0     0     0
  Columns 352 through 364
```

```
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 365 through 377
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 378 through 390
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 391 through 403
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 404 through 416
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 417 through 429
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 430 through 442
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 443 through 455
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 456 through 468
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 469 through 481
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 482 through 494
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 495 through 507
    0     0     0     0     0     0     0     0     0     0     0     0     0
Columns 508 through 512
    0     0     0     0     0
```



Published with MATLAB® R2019a