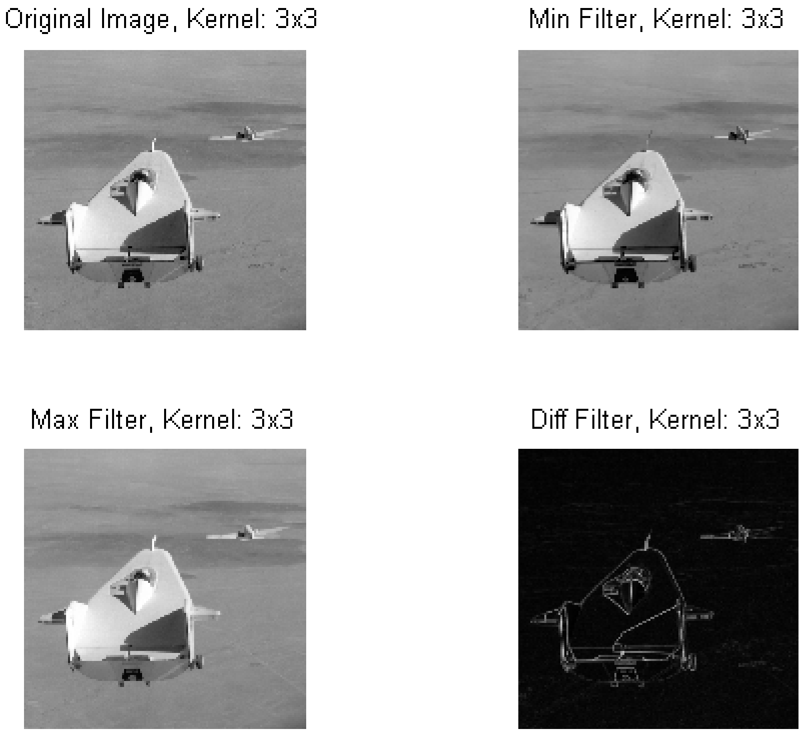
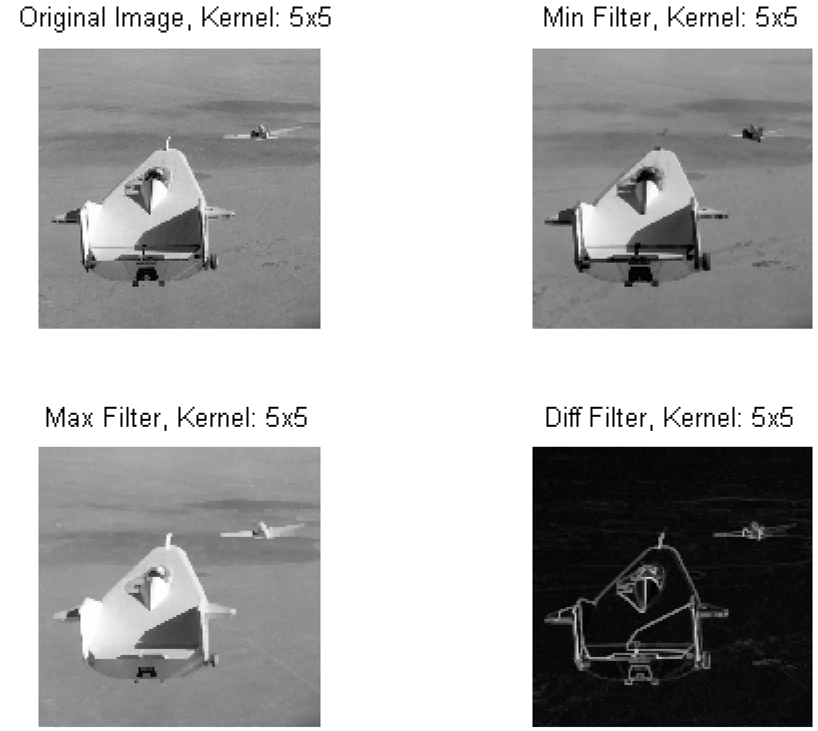
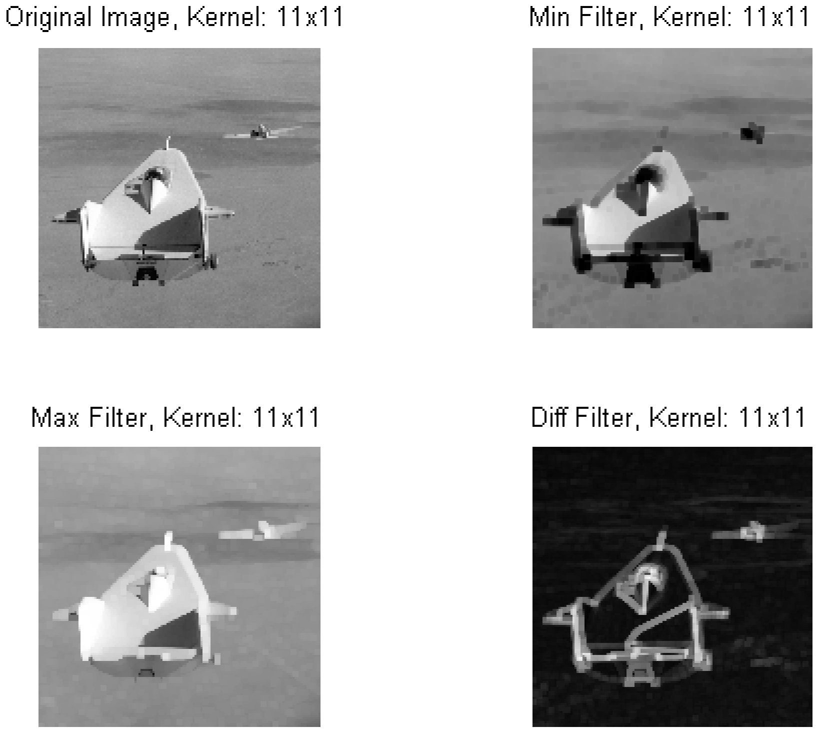
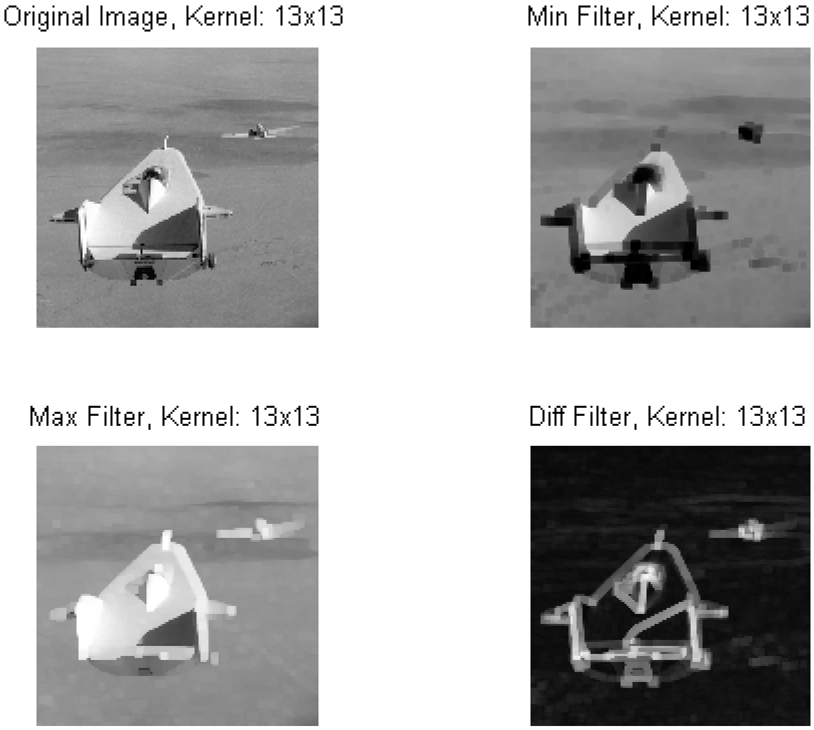
```
%Pratap Luitel
%ENGS 111
%HW 2, Problem 2

%This script implements, calculates and plots min, max and difference
%order filter using the builtin matlab command ordfilt2.

filename='liftingbody.png';
imIn=imread(filename);
imIn=im2double(imIn);
kernel=[3,5,11,13];

for i =1:length(kernel)

    %min filter
    imOutMin=ordfilt2(imIn,1,ones(kernel(i)),'symmetric');
    %clip away values less than 0 to 0 and greater than 1 to 1.
    imOutMin=clip(imOutMin);%clip is a function i wrote,
    %max filter
    imOutMax=ordfilt2(imIn,kernel(i).^2,ones(kernel(i)),'symmetric');
    imOutMax=clip(imOutMax);
    %diff filter
    imOutDiff=imOutMax-imOutMin;
    imOutDiff=clip(imOutDiff);
    %clipping values outside the range[0-1]


    %plotting
    figure(i)
    kStr = num2str(kernel(i)); %kernel string
    subplot(221);imshow(imIn);title(['Original Image, Kernel: ' kStr 'x' kStr])
    subplot(222);imshow(imOutMin);title(['Min Filter, Kernel: ' kStr 'x' kStr])
    subplot(223);imshow(imOutMax);title(['Max Filter, Kernel: ' kStr 'x' kStr])
    subplot(224);imshow(imOutDiff);title(['Diff Filter, Kernel: ' kStr 'x' kStr])



end

%discussion

    fprintf('Each pixel is being replace by the minimum,maximum or\n');
    fprintf('the difference. When the kernel size is higher, \n');
    fprintf('we see bigger patches of brighter or darker pixels in the output.\n')
    fprintf('\n');
    fprintf('The min filter replaces each pixel by a darker pixel values. \n');
    fprintf('the max filter replaces each pixel by a brighter pixel values.\n');
    fprintf('The difference filter replaces each pixel by a difference\n');
    fprintf('between max value and min value. Thus the output\n ');
    fprintf('seems similar to that of a laplacian filter.\n')
    fprintf('\n');
fprintf('---------Boundaries--------------------\n');
fprintf('The boundaries contain a lot of zero pixels because ordfilt2\n');
fprintf('uses the default option of padding boundaries to 0.\n');
fprintf('This can be changed by adding the symmetric padding option as a\n');
fprintf('fourth parameter when calling the ordfilt2 command.\n');
```

```
Each pixel is being replace by the minimum,maximum or

the difference. When the kernel size is higher,
we see bigger patches of brighter or darker pixels in the output.

The min filter replaces each pixel by a darker pixel values.
the max filter replaces each pixel by a brighter pixel values.
The difference filter replaces each pixel by a difference
between max value and min value. Thus the output
 seems similar to that of a laplacian filter.

---------Boundaries--------------------
The boundaries contain a lot of zero pixels because ordfilt2
uses the default option of padding boundaries to 0.
This can be changed by adding the symmetric padding option as a
fourth parameter when calling the ordfilt2 command.
```




Original Image, Kernel: 5x5
Min Filter, Kernel: 5x5
Max Filter, Kernel: 5x5
Diff Filter, Kernel: 5x5


Original Image, Kernel: 11x11
Min Filter, Kernel: 11x11
Max Filter, Kernel: 11x11
Diff Filter, Kernel: 11x11



Original Image, Kernel: 13x13

Min Filter, Kernel: 13x13

Max Filter, Kernel: 13x13

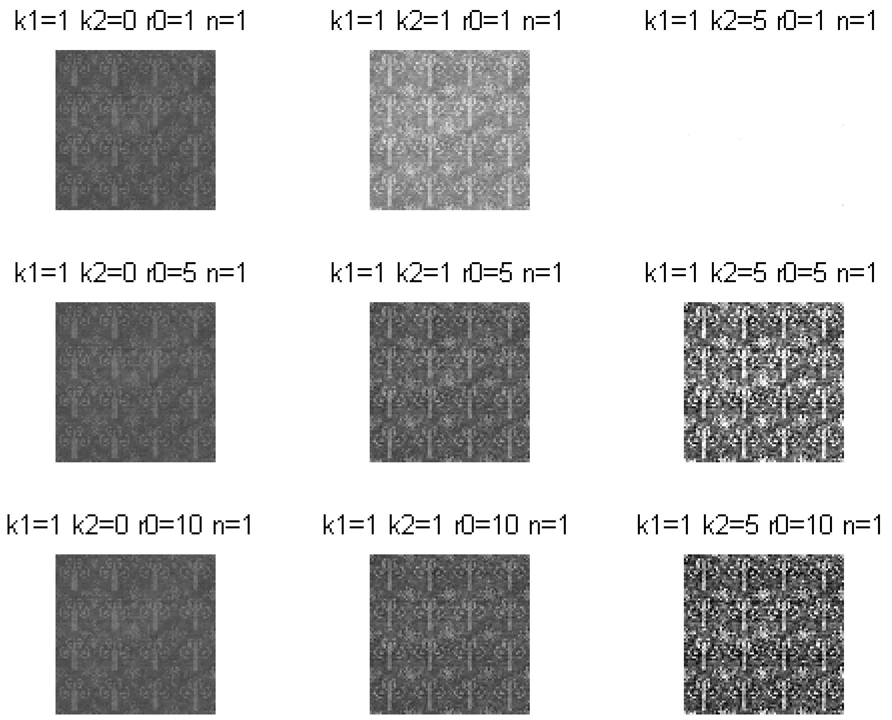
Diff Filter, Kernel: 13x13

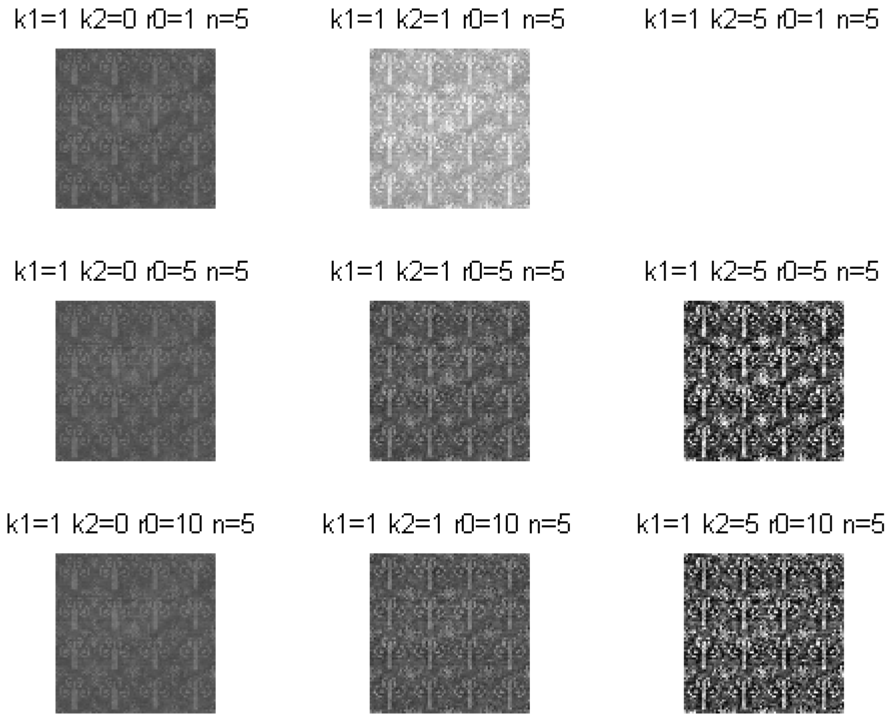
Published with MATLAB® R2014a

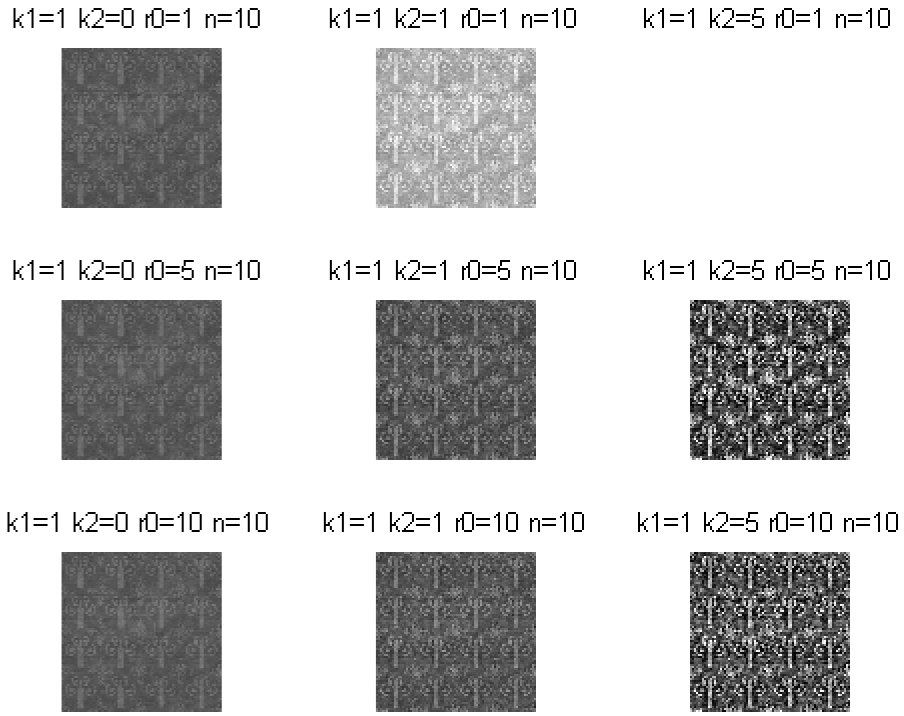
```
%Pratap Luitel
%ENGS 111, HW2 Part C
%This script plots multiple versions of an image, 'WallPaper-1.tif'
%by applying the unsharp mask from the function enhance by varying the
%associated paramerters k1,k2,r0 and n.


filename='WallPaper-1.tiff';
imIn=imread(filename);
imIn=rgb2gray(imIn); %intensity image
imIn=im2double(imIn);%image type double
k1=1;
k2=[0,1,5];
ro=[1,5,10];
n=[1,5,10];


index=1;
for i = 1:3
    for j=1:3
        for k=1:3
            imOut=enhance(imIn,[k1,k2(k),ro(j),n(i)]);

            %converting numeric vals to string for title
            k2Str=num2str(k2(k));
            roStr=num2str(ro(j));
            nStr=num2str(n(i));
            figure(i)

            subplot(3,3,index);
            %positionVector=[left, bottom, width, height]
            %subplot('Position',positionVector)
            imshow(imOut);
            title(['k1=1 k2=' k2Str ' r0=' roStr ' n=' nStr])
            index=index+1;
            if mod(index,9)==1
                index=1;
            end

        end
    end
end
    %plotting half of original and half of output image based on the
    %parameters selected from visual analayis of plotting multiple images
    %in the script above.
    figure
    [nRow,nCol]=size(imIn);
    subplot(131)
    imOut=enhance(imIn,[1,2,5,5]);
    imOut(1:nRow/2,1:nCol)=imIn(1:nRow/2,1:nCol);
    imshow(imOut);
    title('k1=1, k2=1, ro=5, n=5')
```

```
subplot(132)
imOut=enhance(imIn,[1,4,5,5]);
imOut(1:nRow/2,1:nCol)=imIn(1:nRow/2,1:nCol);
imshow(imOut);
title('k1=1, k2=5, ro=5, n=5')


subplot(133)
imOut=enhance(imIn,[1,10,5,5]);
imOut(1:nRow/2,1:nCol)=imIn(1:nRow/2,1:nCol);
imshow(imOut);
title('k1=1, k2=10, ro=5, n=5')

fprintf('To implement the unsharp mask, I implemented the \n');
fprintf('butterworth High Pass Filter. The filter is a function of\n');
fprintf('the radius from the center of the image, ro and n. \n')
fprintf('\n');

fprintf('First I explored the output by passing in multiple variables\n');
fprintf('After visually accessing the output images, I chose some \n');
fprintf('variables that I thought were producing better results \n');
fprintf('Using those values, I have plotted 3 images with original \n');
fprintf('enhanced image stacked together.\n');
```

```
    To implement the unsharp mask, I implemented the
    butterworth High Pass Filter. The filter is a function of
    the radius from the center of the image, ro and n.

    First I explored the output by passing in multiple variables
    After visually accessing the output images, I chose some
    variables that I thought were producing better results
    Using those values, I have plotted 3 images with original
    enhanced image stacked together.
```


k1=1 k2=0 r0=1 n=1
k1=1 k2=1 r0=1 n=1
k1=1 k2=5 r0=1 n=1
k1=1 k2=0 r0=5 n=1
k1=1 k2=1 r0=5 n=1
k1=1 k2=5 r0=5 n=1
k1=1 k2=0 r0=10 n=1
k1=1 k2=1 r0=10 n=1
k1=1 k2=5 r0=10 n=1


k1=1 k2=0 r0=1 n=5
k1=1 k2=1 r0=1 n=5
k1=1 k2=5 r0=1 n=5
k1=1 k2=0 r0=5 n=5
k1=1 k2=1 r0=5 n=5
k1=1 k2=5 r0=5 n=5
k1=1 k2=0 r0=10 n=5
k1=1 k2=1 r0=10 n=5
k1=1 k2=5 r0=10 n=5



k1=1 k2=0 r0=1 n=10

k1=1 k2=1 r0=1 n=10

k1=1 k2=5 r0=1 n=10

k1=1 k2=0 r0=5 n=10

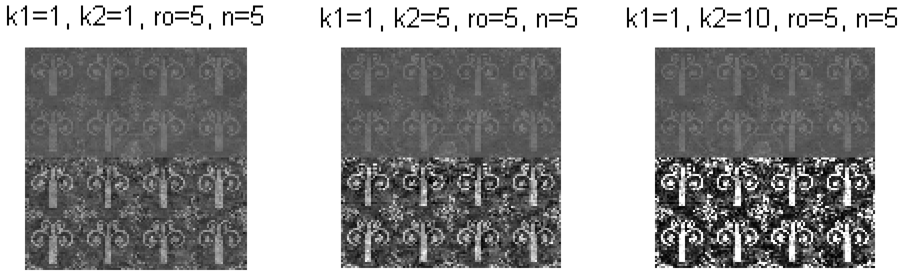
k1=1 k2=1 r0=5 n=10

k1=1 k2=5 r0=5 n=10

k1=1 k2=0 r0=10 n=10

k1=1 k2=1 r0=10 n=10

k1=1 k2=5 r0=10 n=10



k1=1, k2=1, ro=5, n=5

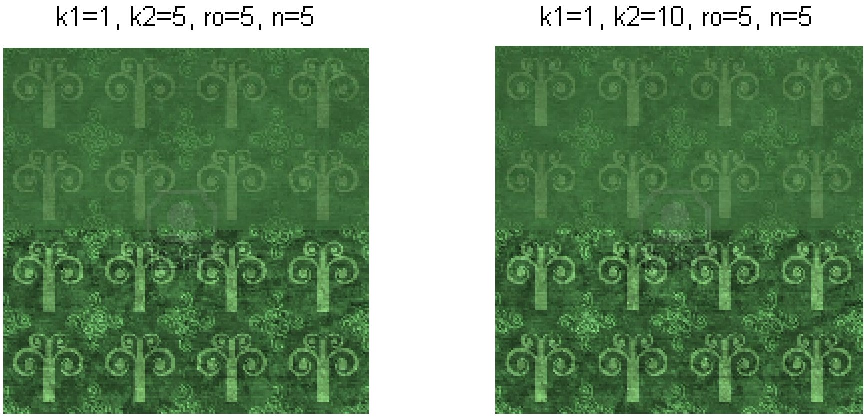
k1=1, k2=5, ro=5, n=5

k1=1, k2=10, ro=5, n=5

Published with MATLAB® R2014a

```
%Pratap Luitel
%ENGS 111, HW2 Part C -Bonus

%This script implements the unsharp mask enhance effect for colored images.
%The input image is in the rgb space which is then converted to hsv space.
%The V from the hsv is then processed to obtain an enhanced V. The new HSV
%is then converted back into rgb space to display the images.

filename='WallPaper-1.tiff';
%filename='Peppers.png';
imIn=imread(filename);
imIn=im2double(imIn);%convert input image to type double
hsvImage=rgb2hsv(imIn); %convert to hsv
h=hsvImage(:,:,1);
s=hsvImage(:,:,2);
v=hsvImage(:,:,3);%V in HSV is the input image

  figure
    [nRow,nCol]=size(imIn);


    subplot(121)
    newV=enhance(v,[1,2,5,5]);
    newHSV=cat(3,h,s,newV);
    imOut=hsv2rgb(newHSV);
    imOut(1:nRow/2,1:nCol)=imIn(1:nRow/2,1:nCol);
    imshow(imOut);
    title('k1=1, k2=5, ro=5, n=5')


    subplot(122)
    newV=enhance(v,[1,2,5,5]);
    newHSV=cat(3,h,s,newV);
    imOut=hsv2rgb(newHSV);
    imOut(1:nRow/2,1:nCol)=imIn(1:nRow/2,1:nCol);
    imshow(imOut);
    title('k1=1, k2=10, ro=5, n=5')

    fprintf('I first converted the rgb space image into HSV.\n');
    fprintf('Then, the V in HSV is used as the input image for\n');
    fprintf('implementing the unsharp mask. \n');
    fprintf('\n');
    fprintf('The new HSV image is then converted back into rgb space.\n');
```

```
I first converted the rgb space image into HSV.
Then, the V in HSV is used as the input image for
implementing the unsharp mask.

The new HSV image is then converted back into rgb space.
```



k1=1, k2=5, ro=5, n=5

k1=1, k2=10, ro=5, n=5

Published with MATLAB® R2014a