**Chapter 6**

Example 1. For given X and Y,

1. Calculate Z=X+Y, using normalization and
2. Truncation

Ans.

X =

200 100 100

0 10 50

50 250 120

Y =

100 220 230

45 95 120

205 100 0

(a)

W=uint16(X)+uint16(Y)

W =

300 320 330

45 105 170

255 350 120

Z=255/305\*(W-45)

Z =

213 230 238

0 50 105

176 255 63

(b) Z=imadd(X,Y)

Z =

255 255 255

45 105 170

255 255 120

Example 2. For given X and Y(in Ex.1), calculate

1. Z=X-Y(using truncation)
2. Z=Y-X(using truncation)
3. Z=|Y-X|

Ans.

(a)

Z= imsubtract(X,Y)

Z =

100 0 0

0 0 0

1. 150 120

(b)

Z= imsubtract(Y,X)

Z =

0 120 130

45 85 70

155 0 0

(c )

Z=imabsdiff(Y,X)

Z =

100 120 130

45 85 70

155 150 120

Example 3. For given X, Y(in Ex1) and Z, calculate (X+Y+Z)/3 using

1. imadd and imdivide without handling truncations and round-offs
2. imadd and imdivide handling truncations and round-offs
3. imlincomb

Ans

(a)

S=imdivide(imadd(imadd(X,Y),Z),3)

S =

85 85 85

63 85 85

85 85 85

(b)

S=uint8(imdivide(imadd(imadd(uint16(X),uint16(Y)),uint16(Z)),3))

S =

167 160 153

63 100 97

120 197 90

(c)

S=imlincomb(1/3,X,1/3,Y,1/3,Z)

S =

167 160 153

63 100 97

120 197 90

Tutorial – arithmetic operations

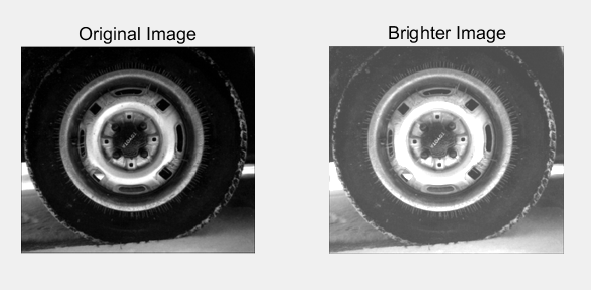
1. Use the imadd function to brighten an image by adding a constant value to all its pixel values.

I=imread('tire.tif');

>> I2=imadd(I,75);

>> figure, subplot(1,2,1),imshow(I), title('Original Image');

>> subplot(1,2,2), imshow(I2), title('Brighter Image');



Q1. What are the maximum and minimum values of the original and the adjusted image?

A. max(max(I))

ans =

255

max(max(I2))

ans =

255

Both are 255.

Q2. How many pixels had a value of 255 in the original image and how many have a value of 255 in the resulting image?

A.

>> [i,j] = find(I==255);[i,j];

>> size(i)

ans =

301 1

>> [i,j] = find(I2==255);[i,j];

>> size(i)

ans =

2614 1

301, 2614.

1. Use the imadd function to blend two images.

Ia=imread('rice.png');

Ib=imread('cameraman.tif');

Ic=imadd(rgb2gray(Ia),Ib);

figure, imshow(Ic);



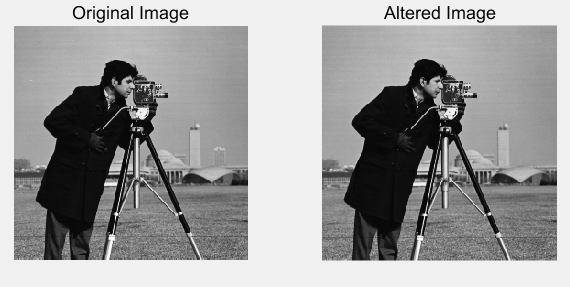
1. Close all open figures and clear all workspace variables.
2. Load two images and display them.

>> I=imread('cameraman.tif');

>> J=imread('cameraman2.tif');

>> figure, subplot(1,2,1),imshow(I), title('Original Image');

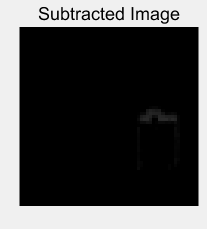
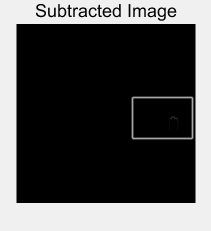
>> subplot(1,2,2),imshow(J), title('Altered Image');



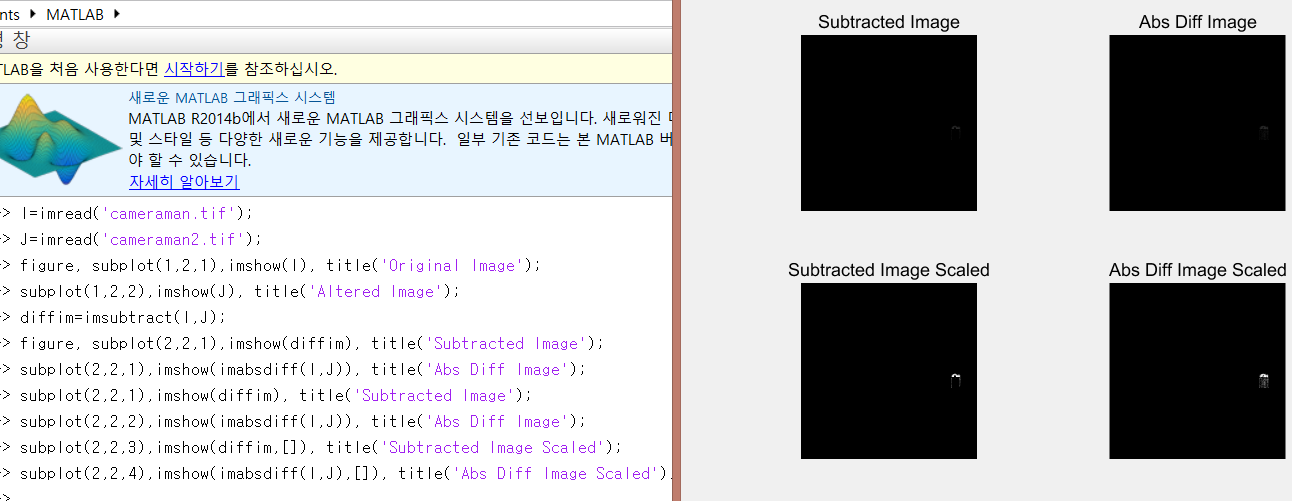
1. Subtract both images and display the result.
2. Use the zoom tool to zoom into the right area of the difference image about halfway down the image. You will notice that a small region of pixels is faintly white.
3. To zoom back out, double-clock anywhere on the image.

>> diffim=imsubtract(I,J);

>> figure, subplot(2,2,1),imshow(diffim), title('Subtracted Image');



1. Calculate the absolute difference. Make sure Figure 2 is selected before executing this code.
2. Use the zoom-in tool to inspect the new difference image.
3. Show scaled versions of both difference images.
4. Use the zoom tool to see the differences between all four difference images.



Q3. How did we scale the image output?

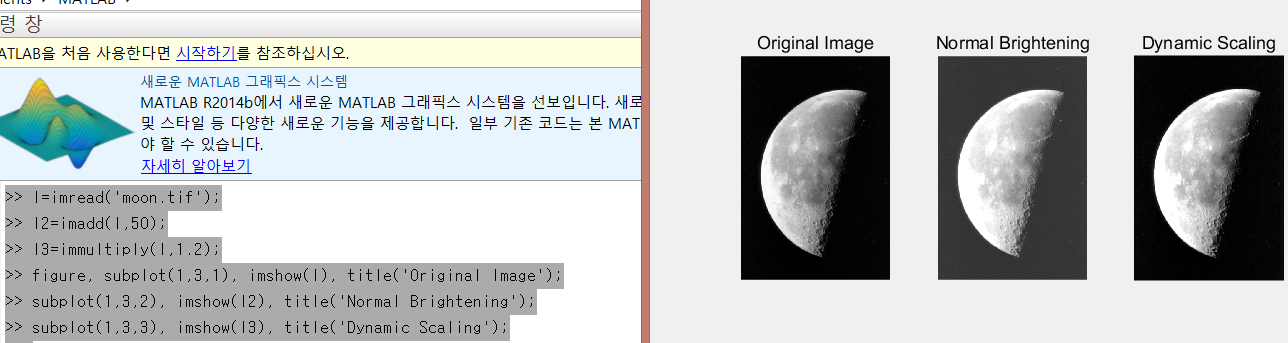
1. The range of pixel’s intensity has been scaled.

Q4. What happened when we scaled the difference images?

1. The thing we couldn’t see when we just subtract those image is seen.

Q5. Why does the last image show more detail than the others?

1. The last image is scaled, and the truncated things appeared.
2. Close all open figures and clear all workspace variables.
3. Use immultiply to dynamically scale the moon image.

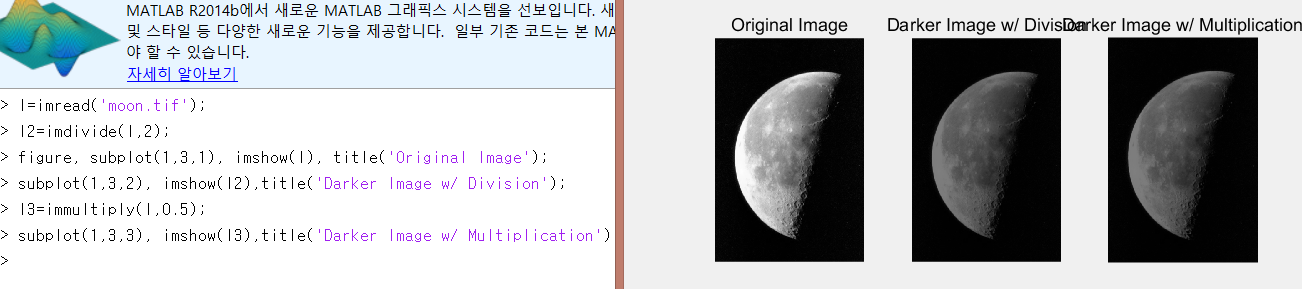


Q6. When dynamically scaling the moon image, why did the dark regions around the moon not become brighter as in the normally adjusted image?

1. Because when a pixel is dark, the intensity of the pixel is small, so multiplicating 1.2 to it barely changes the intensity.
2. Close all open figures and clear all workspace variables.
3. Create an artificial 3D planet by using the immultiply function to multiply the earth1 and earth2 images.



1. Close all open figures and clear all workspace variables.
2. Use image division to dynamically darken the moon image.
3. Display the equivalent darker image using image multiplication.



Q7. Why did the multiplication procedure produce the same result as division?

1. Because dividing by nonzero a is equivalent to multiplying the reciprocal of a.

Q8.Write a small script that will verify that the images produced from division and multiplication are equivalent.

I4=imabsdiff(I2,I3);

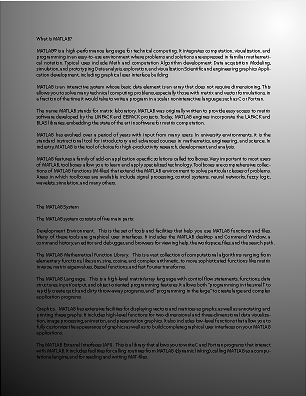
Imshow(I4, []);

1. Close all open figures and clear all workspace variables.
2. Load the images that will be used for background subtraction.

>> notext=imread('gradient.tif');

>> text=imread('gradient\_with\_text.tif');

>> figure,imshow(text), title('Original Image');

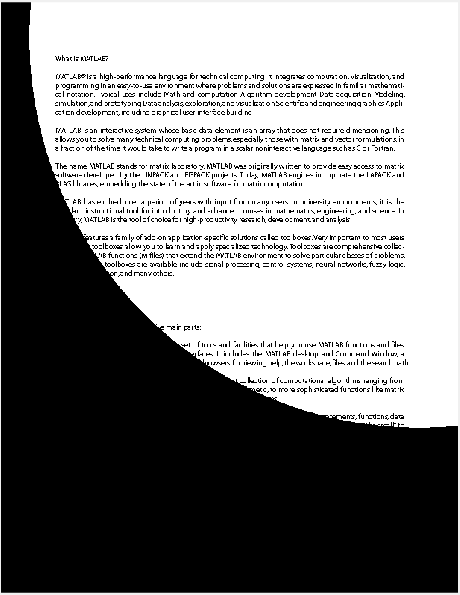


1. Show how thresholding fails in this case.

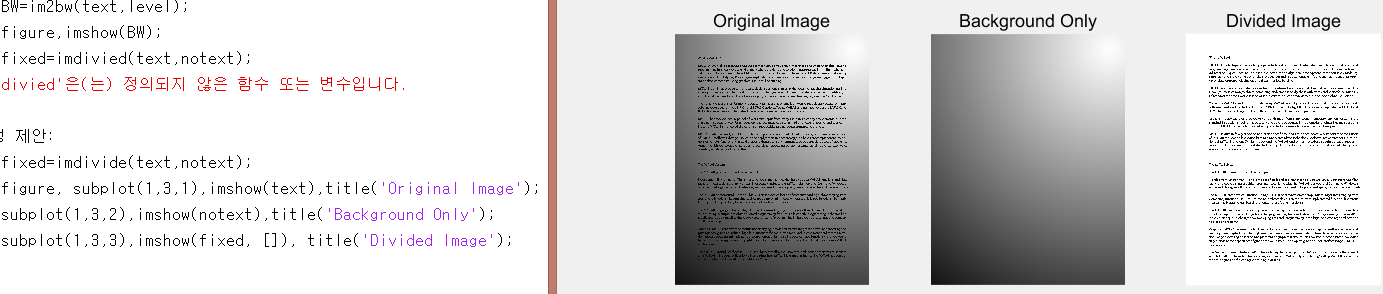
level=graythresh(text);

>> BW=im2bw(text,level);

>> figure,imshow(BW);

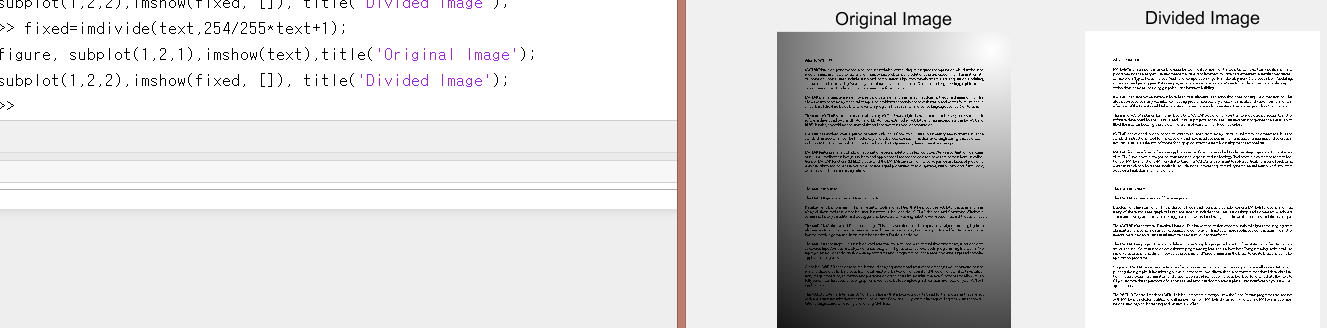


1. Divide the background from the image to get rid of the background.



Q9. Would this technique still work if we were unable to obtain the background image?

1. Yes, it would.



Tutorial – logic operations and region of interest processing

1. Use the MATLAB help system to learn how to use the roipoly function when only an image is supplied as a parameter.

Q1. How do we add points to the polygon?

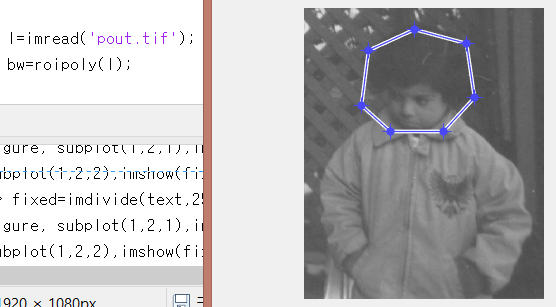
A. To add new vertices, position the pointer along an edge of the polygon and press the "A" key. The pointer changes shape. Left-click to add a vertex at the specified position.

Q2. How do we delete points from the polygon?

A. To delete the polygon, press Backspace, Escape or Delete, or choose the cancel option from the context menu. If the polygon is deleted, all return values are set to empty.

Q3. How do we end the process of creating a polygon?

1. Double-click to add a final vertex to the polygon and close the polygon. Right-click to close the polygon without adding a vertex.
2. Use the roipoly function to generate a mask for the pout image.

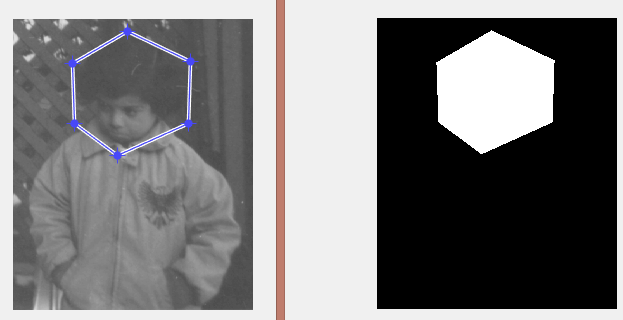


Q4. What class is the variable bw?

1. Logical.

Q5. What does the variable bw represent?

1. The region inside the polygon I made.



1. Convert the mask image to class uint8.

bw2=uint8(bw);

Q6. In the above conversion step, what would happen if we used the im2uint8 function to convert the bw image as opposed to just using uint8(bw)?

A.

max(max(bw2))

ans =

1

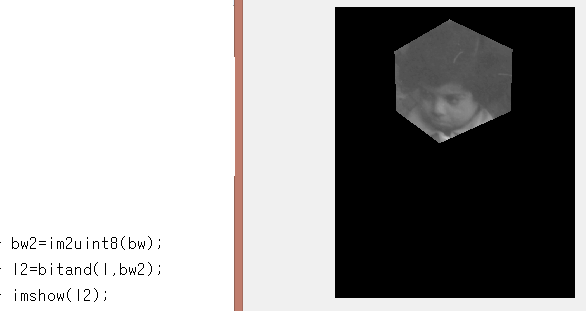
>> max(max(im2uint8(bw)))

ans =

255

When we use uint8(bw), bw2 has the only values 0 and 1, but when we use im2uint8, bw2 has values 0 and 255.

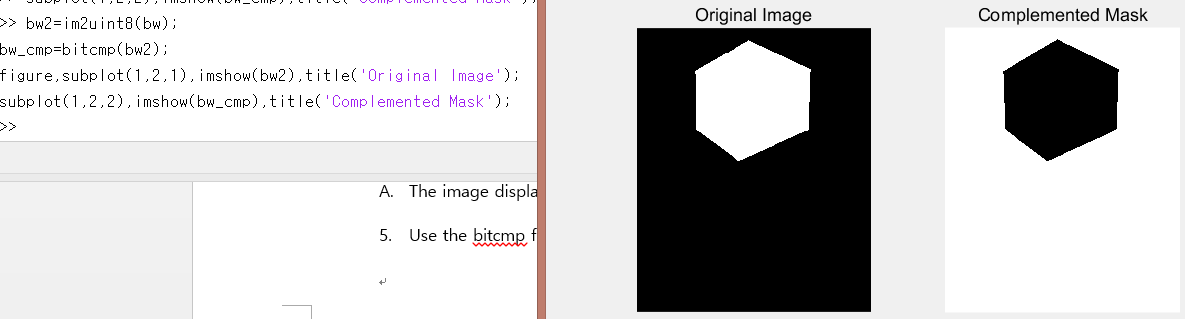
1. Use the bitand function to compute the logic AND between the original image and the new mask image.



(used im2uint8 : uint8(bw) doesn’t make proper result)

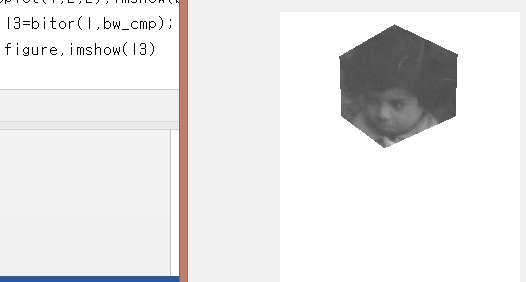
Q7. What happens when we logically AND the two images?

1. The image displays only the region we selected, otherwise black.
2. Use the bitcmp function to generate a complemented version of the bw2 mask.



Q8. What happened when we complemented the bw2 image?

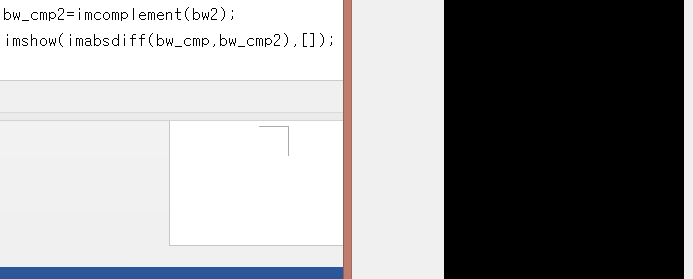
1. The white region became black, black region became white.
2. Use bitor to compute the logic OR between the original image and the complemented mask.



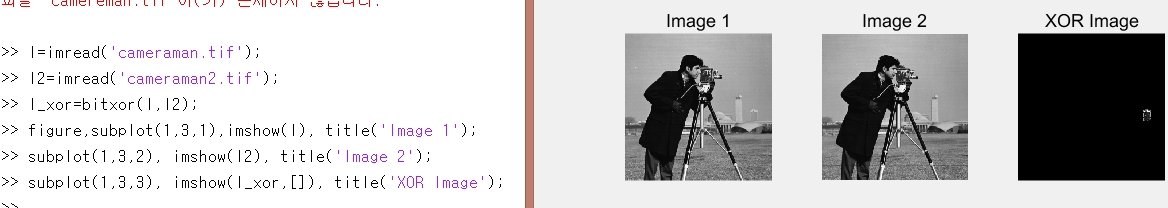
Q9. Why did we need to complement the mask? What would have happened if we used the original mask to perform the OR operation?

1. When we perform OR with the complemented mask, the region we selected displays the original image, whether displays only white.
2. Complement an image using the imcomplement function.

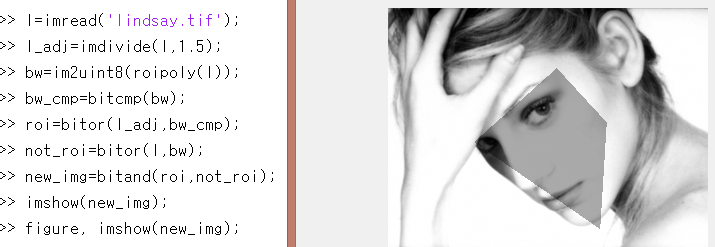
Q10. How can we check to see that the bw\_cmp2 image is the same as the bw\_cmp image?



1. The results is only black, it means those are same.
2. Close all open figures and clear all workspace variables.
3. Use the bitxor function to find the difference between two images.



1. Close all open figures and clear all workspace variables.
2. Read in image and calculate an adjusted image that is darker using the imdivide function.
3. Generate a mask by creating a region of interest polygon.
4. Use logic operators to show the darker image only within the region of interest, while displaying the original image elsewhere.



Q11. How could we modify the above code to display the original image within the region of interest and the darker image elsewhere?

A.

