

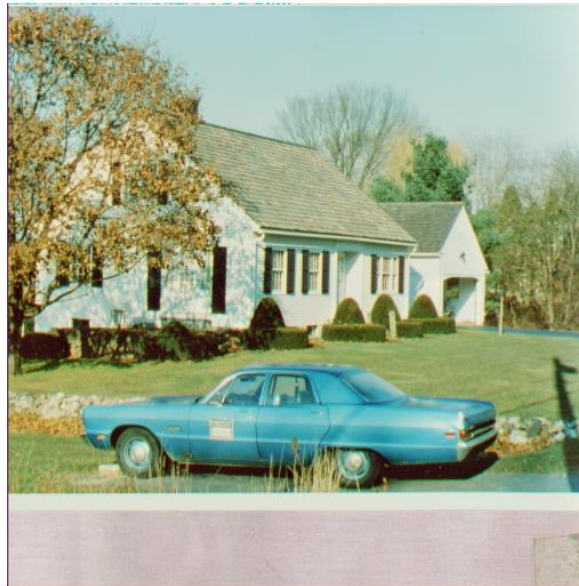
1. Reading and Displaying Images:

Read and display the given images “peppers.tiff”, “house.tiff”. Find the dimensions of each image.

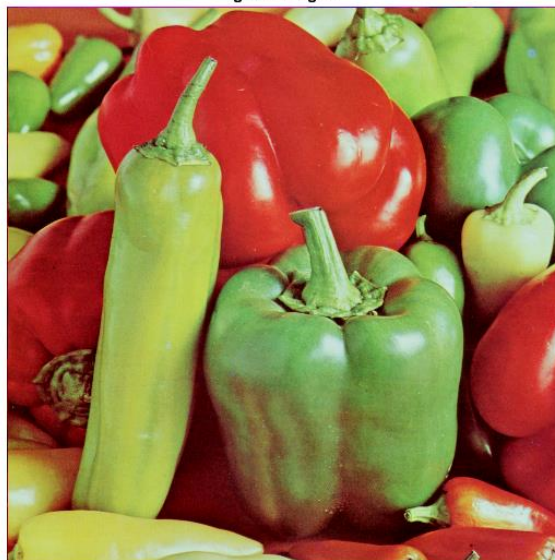
Aim: Display the given images “peppers.tiff”, “house.tiff” and Find the dimensions of each image.

Output:


given image-1



given image-2



Command Window				
Name	Size	Bytes	Class	Attributes
img1	512x512x3	786432	uint8	
img2	512x512x3	786432	uint8	

 >>

Inferences:

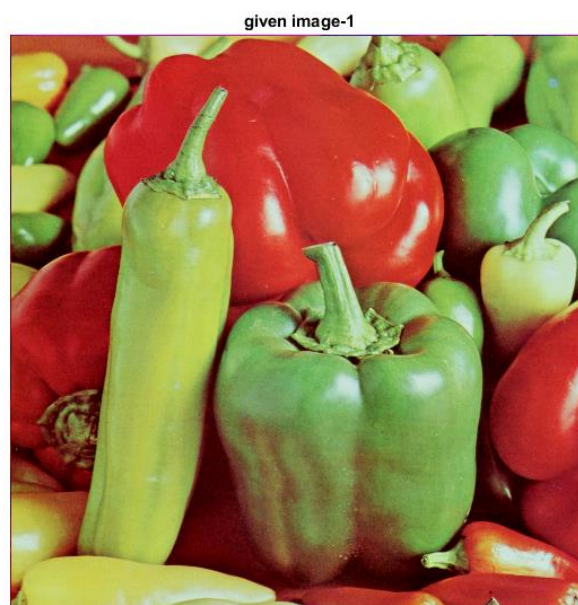
1. The dimensions of "house.tiff" -----> 512*512*3
"peppers.tiff" -----> 512*512*3
2. Both the images have 3 channels red, green, blue.
3. Both images are in uint8 format so the pixel intensity range from 0 to 255.

2. Manipulation of Colour Images:

- a. Swap the red and blue pixels of image 1 in problem-1, and display the new image.
- b. Create a monochrome image (M1g) by selecting the green channel of image 1, and display the new image.
- c. Create a monochrome image (M1r) by selecting the red channel of image 1, and display the new image.
- d. Which one of the above two outputs looks more like what you'd expect a monochrome image to look like? Would you expect a computer vision algorithm to work on one better than the other? Justify your answer.

Aim: To Swap the red and blue pixels of image 1 in problem-1 , Create a monochrome image (M1g) , image (M1r) by selecting the green and red channel of image 1 respectively.

Output:



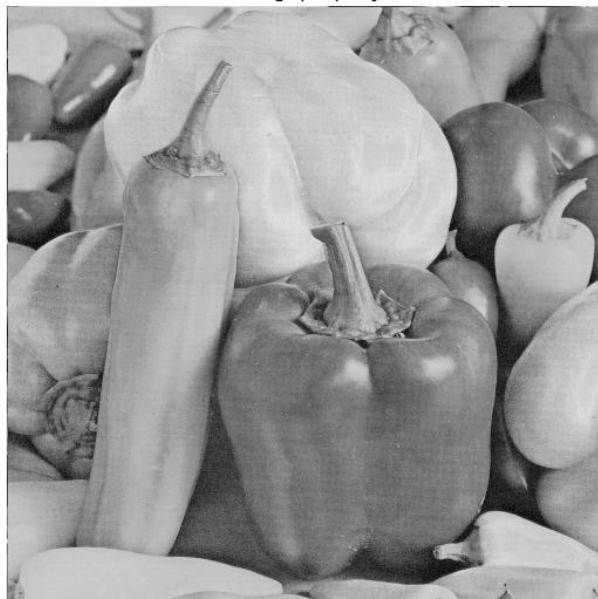
after swaping the colours of image-1



monochrome image (M1g) only green channel

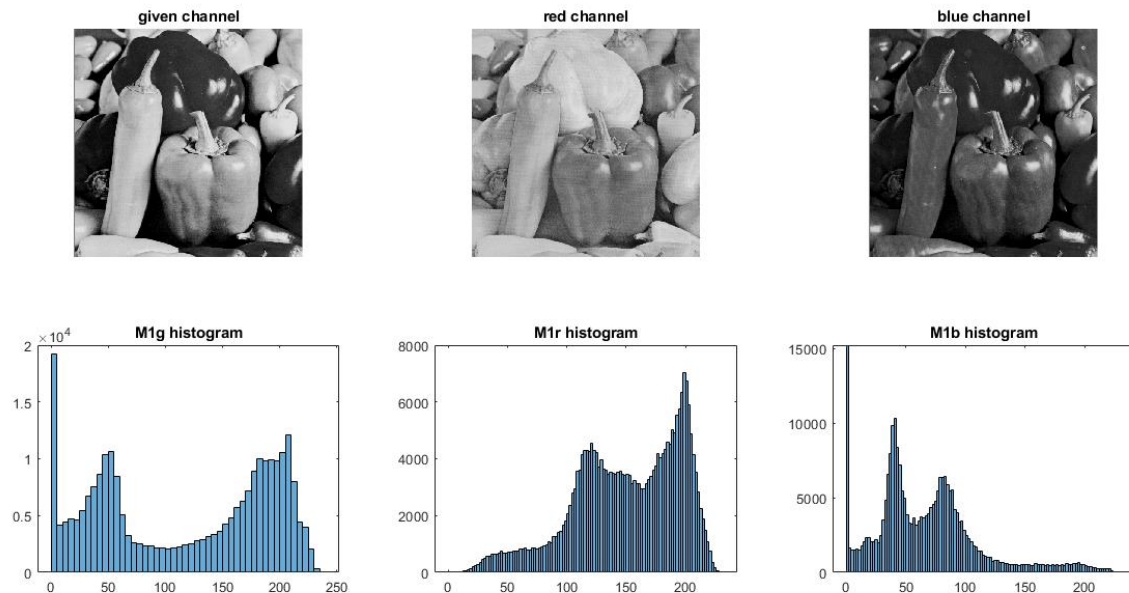


monochrome image (M1r) only red channel



Inferences:

1. Generally in any colour image have 3 channels , green, red and blue. From these 3 channels, the green channels (M1g) is looks more like what you'd expect a monochrome image to look like. Because if we look at the histogram of the 3 channels bellow figure,



The green channel pixel intensities are more spread throughout the intensity levels when compared to the remaining 2 channels.

2. So green channel is mostly preferable to perform a computer vision algorithm(e.g., thresholding, segmentation) to work on one better than the other.

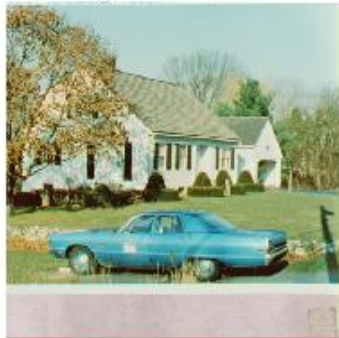
3. Replacement of Pixels:

Take the inner square of 50x100 pixels (that's 50 rows by 100 columns – a horizontal image) of the monochrome version of image 1 in problem-1, and insert them into the monochrome version of image 2 of problem-1. Display the resulting image.

Aim: Replacing 50*100 pixel size from the monochrome version of image 1 in problem-1, and insert them into the monochrome version of image 2 of problem-1.

Output:

given image-1



given image-2



monochrome version of image 1



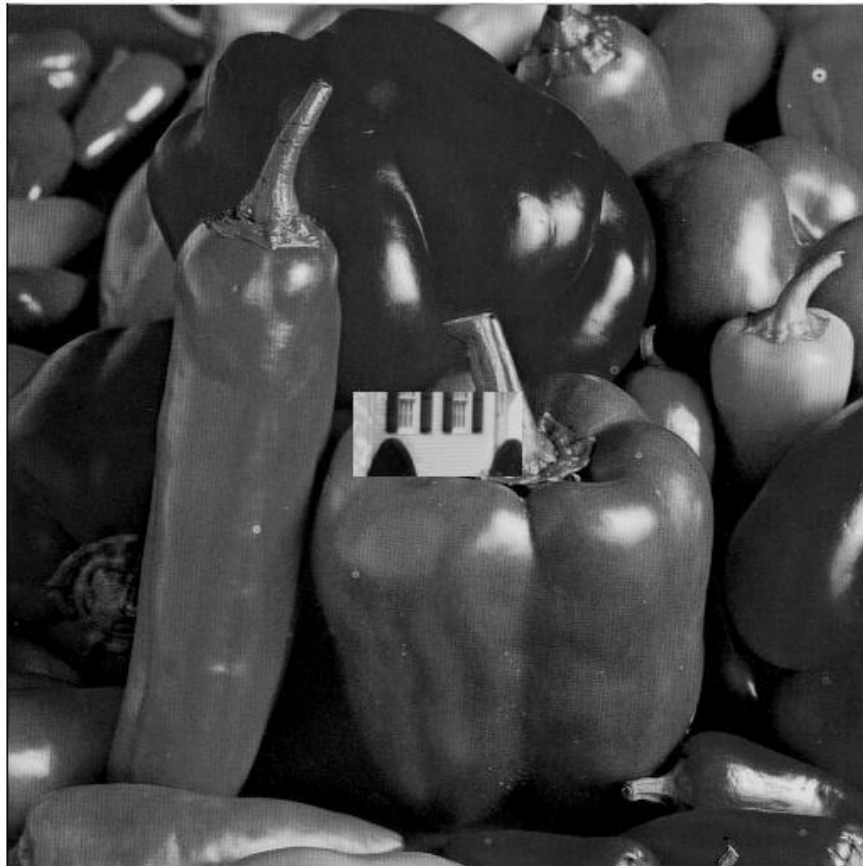
monochrome version of image 2



selected 50*100 pixel of image-1 at center



50*100 pixel of image-1 inserted in image-2 as same position



Inferences:

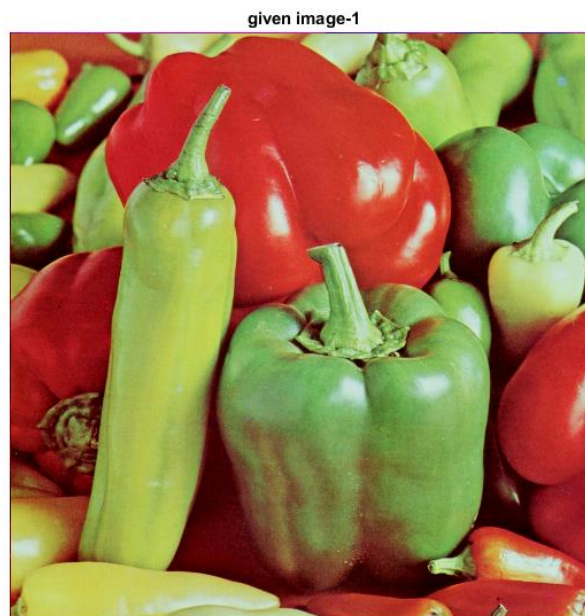
1. I used blue channel of the image for this question because that is giving me more contrast and showing the 50*100 pixel in the centre of the replaced image properly.

4. Arithmetic and Geometric Operations:

- What is the min and max of the pixel values of M1g? What is the mean? What is the standard deviation? And how did you compute these?
- Subtract the mean from all the pixels, then divide by the standard deviation, then multiply by 10 (if your image is zero to 255) or by 0.05 (if your image ranges from 0.0 to 1.0). Now add the mean back in. Display the resulting image.
- Shift M1g to the left by 2 pixels, and display the resulting image.
- Subtract the shifted version of M1g from the original and make sure that the values are legal (what do negative numbers for pixels mean anyway?).

Aim: To find the min, max, mean, standard deviation of the image M1g. and perform the operation of Shifting M1g to the left by 2 pixels, and display the resulting image. And subtract the shifted version of M1g from the original and display the result.

Output:



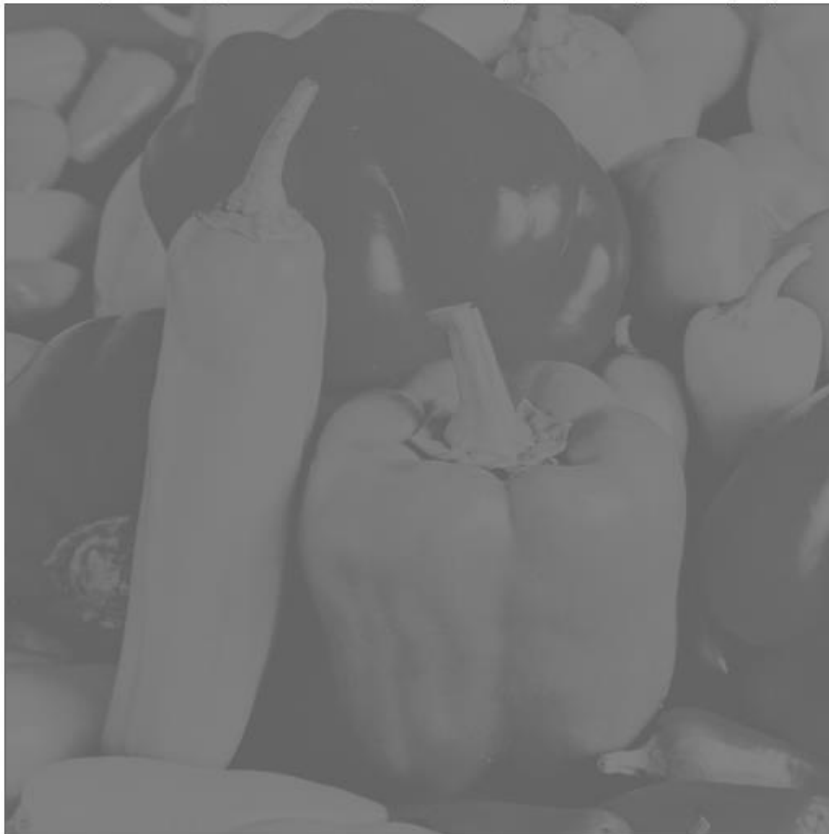
```
Command Window
minimum pixel value= 0
maximum pixel value= 237
mean of M1g is = 115.5683
standard deviation of M1g is = 75.0461
fx >>
```

Zoom: 100% UTF-8 CRLF script Ln 57 Col 55

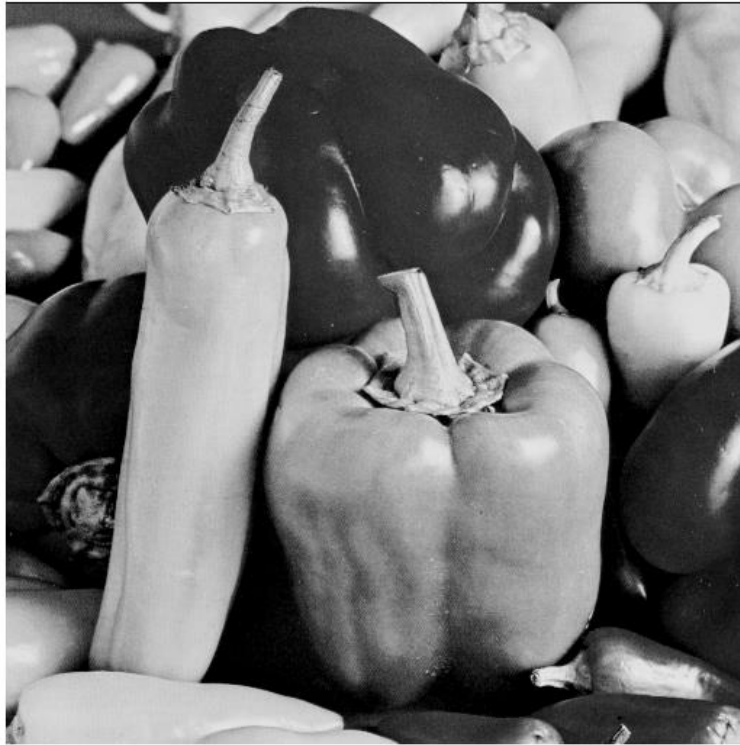
monochrome image (M1g) only green channel



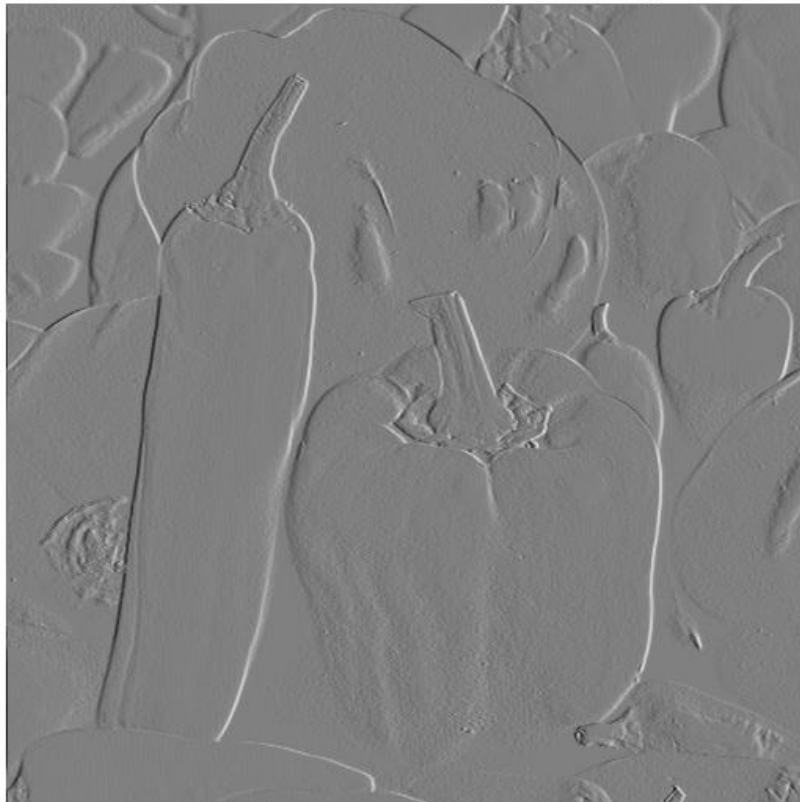
Required image after doing required operation in question(4-b)



Shift M1g to the left by 2 pixels



Subtract the shifted version of M1g from the original



Inferences:

1. In MATLAB minimum value of an array can be found using `min()` command. But minimum values of 2D array or image can be found using 2 different ways,
 1. First convert 2D array convert into 1D array then find min value using `min()` command.
Eg. `Img1` is a image or 2D array then `min(img1 (:))` gives the minimum value. Or `min(min(img1))` also gives the minimum value.
 2. **Minimum value of M1g = 0**
2. In MATLAB maximum value of an array can be found using `max()` command. But maximum values of 2D array or image can be found using 2 different ways,
 1. First convert 2D array convert into 1D array then find max value using `max()` command.
Eg. `Img1` is a image or 2D array then `max(img1 (:))` gives the maximum value. Or `max(max(img1))` also gives the maximum value.
 2. **maximum value of M1g = 237**
3. In MATLAB mean value of an array can be found using `mean()` command. But mean values of 2D array or image can be found using 2 different ways,
 1. First convert 2D array convert into 1D array then find mean value using `mean()` command.
Eg. `Img1` is a image or 2D array then `mean(img1 (:))` gives the mean value. Or `mean(mean(img1))` also gives the mean value.
 2. 2nd method is using the formula **Mean = {Sum of pixel values} ÷ {Total numbers of pixel values}**.
 3. **Mean value of the M1g =115.5683**

4. Standard deviation can be found using the formula $\sigma = \sqrt{\frac{\sum(x-\mu)^2}{N}}$

σ =standard deviation

x =pixel value

μ = mean of the image

N = number of pixels.

And standard deviation can be found different ways in MATLAB other than using above formula. **Var()** is a command is used to find the variance of the 1D array. But image is a 2D so we can find the variance of the image by converting 1D array. **Standard deviation = $\sqrt{\text{variance}}$** .

But the given image, the data type is unsigned int (uint8).

Calculate SD using in uint8 data type

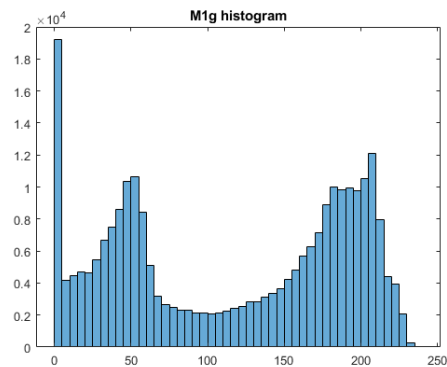
Standard deviation = 11.3484

This is **wrong** because in finding the standard deviation, any value more than 255 is always assigned as 255 so it gives the wrong value.

Calculate SD by converting Uint8 to double data type

Standard deviation = 75.0461 [**correct**]

By looking at the histogram we can clearly observe that standard deviation is around 75.



5. After circular shifting M1g to the left by 2 pixels, the resultant image the first 2 columns added at the end of M1g.
6. when we convert the image into double, the values should be in the range 0.0 to 1.0, then only in MATLAB `imshow()` command display the image properly. So, converting image into Uint8 to double, the range of the double image from 0 to 255 so by dividing resultant image by 255 then it gives the new range from 0.0 to 1.0
7. when we subtract the image of left shifted by 2 pixel from the original image, it gives the edges information of the image.
8. Negative pixel values get due to some subtraction operation or To represent more intensity levels.
If your data type allows it, like signed integer or floating point, it makes perfect sense to use negative values and it is a very common way to specify ignored pixels. In this case there is no special meaning to a negative value except your interpretation of it.