Mohammad Elham Amin

21701543

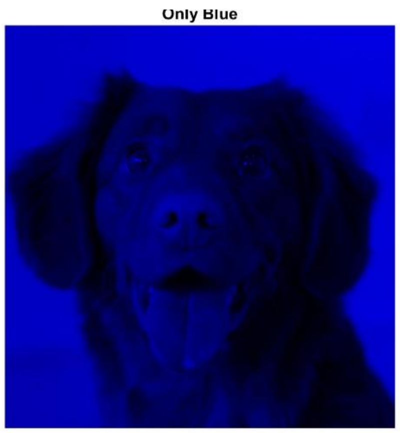
MATLAB ASSIGNMENT 1

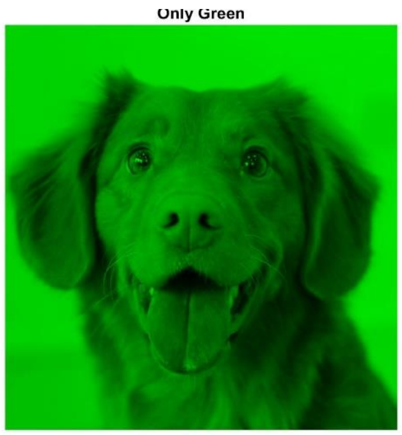
EEE 391

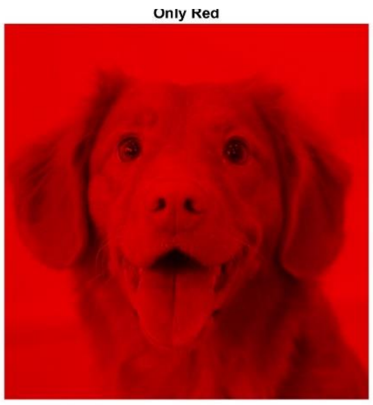
11/21/2021

Part 1

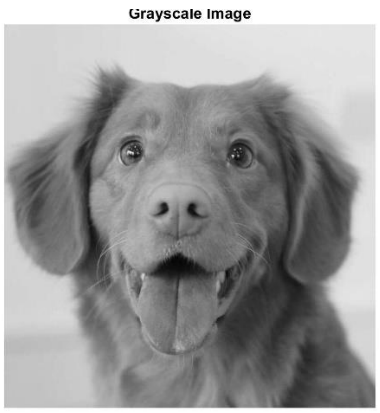
i) Each color channel in an RGB image show how much of contribution that channel has in a pixel. We can see that by setting a channel to zero in all pixels we lose all that color from the image. Therefore, we have three images of each having only one color, because other two channels have been removed.



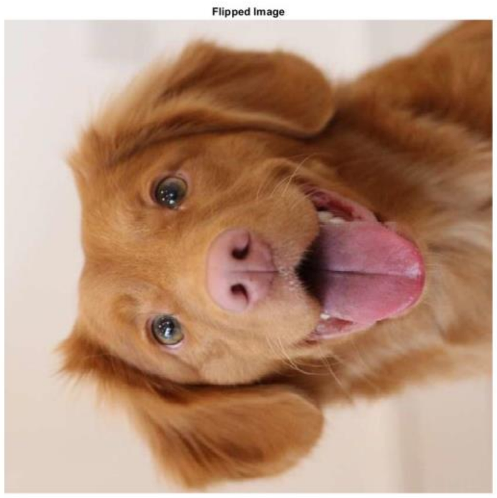


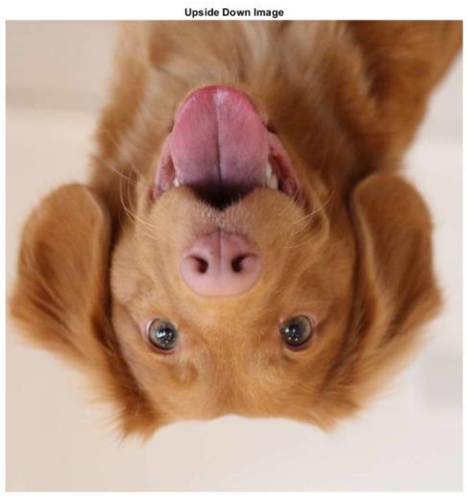


ii) Grayscale image:

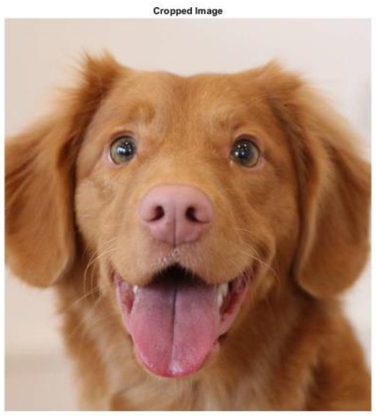


iii) Rotate images

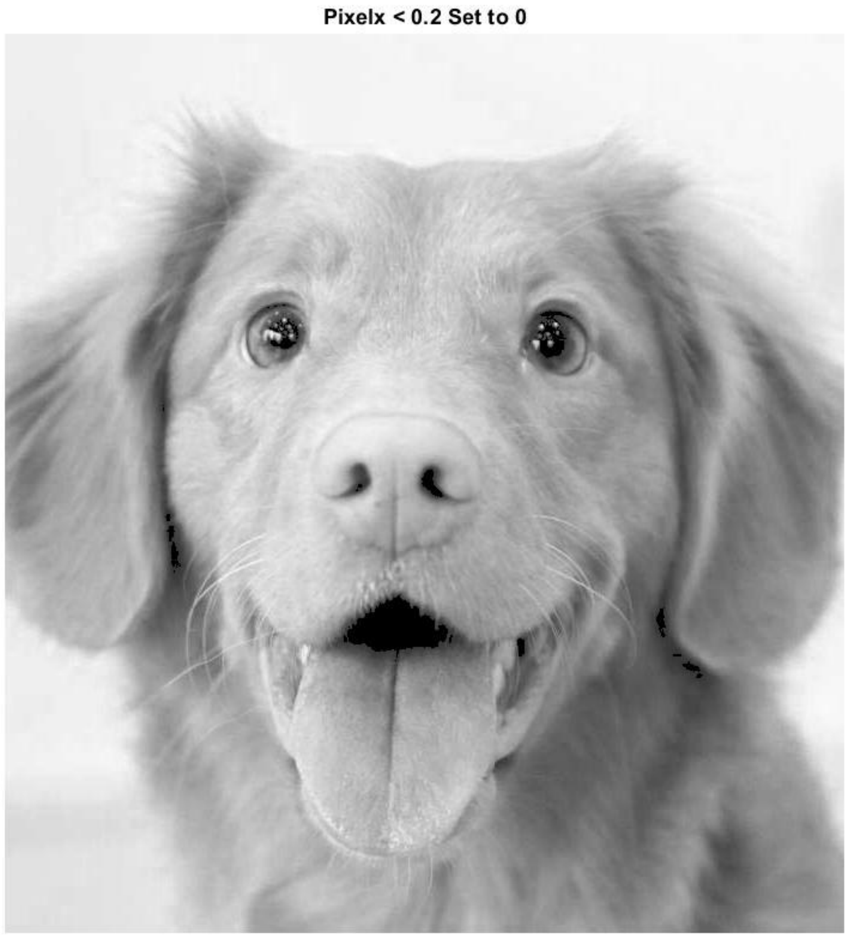




iv) Cropped image between 25-775 and 50-750 in horizontal and vertical directions respectively.



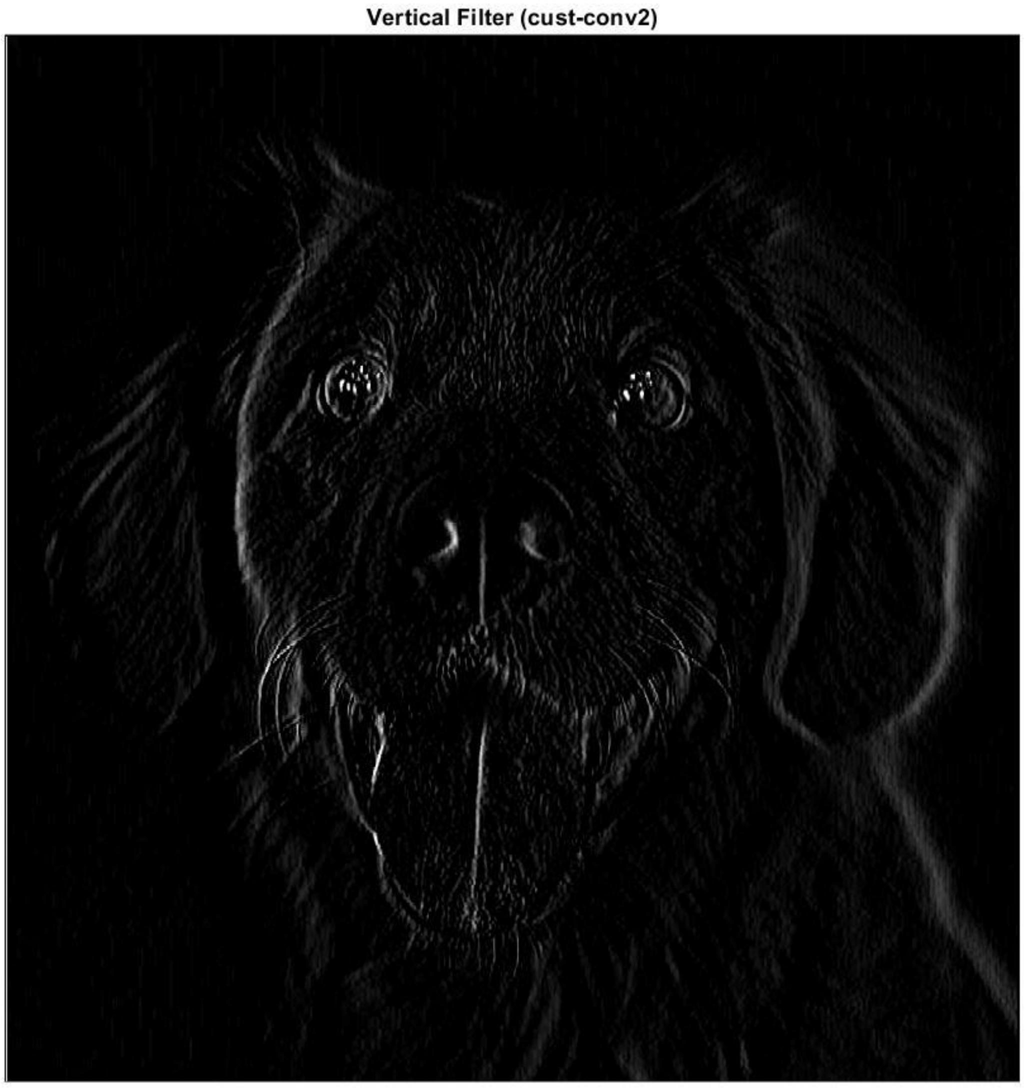
v) As a result of changing pixels smaller than 0.2, some of the parts have become completely dark. For example, from the new picture we can see that the parts under the dog’s ear have become black.



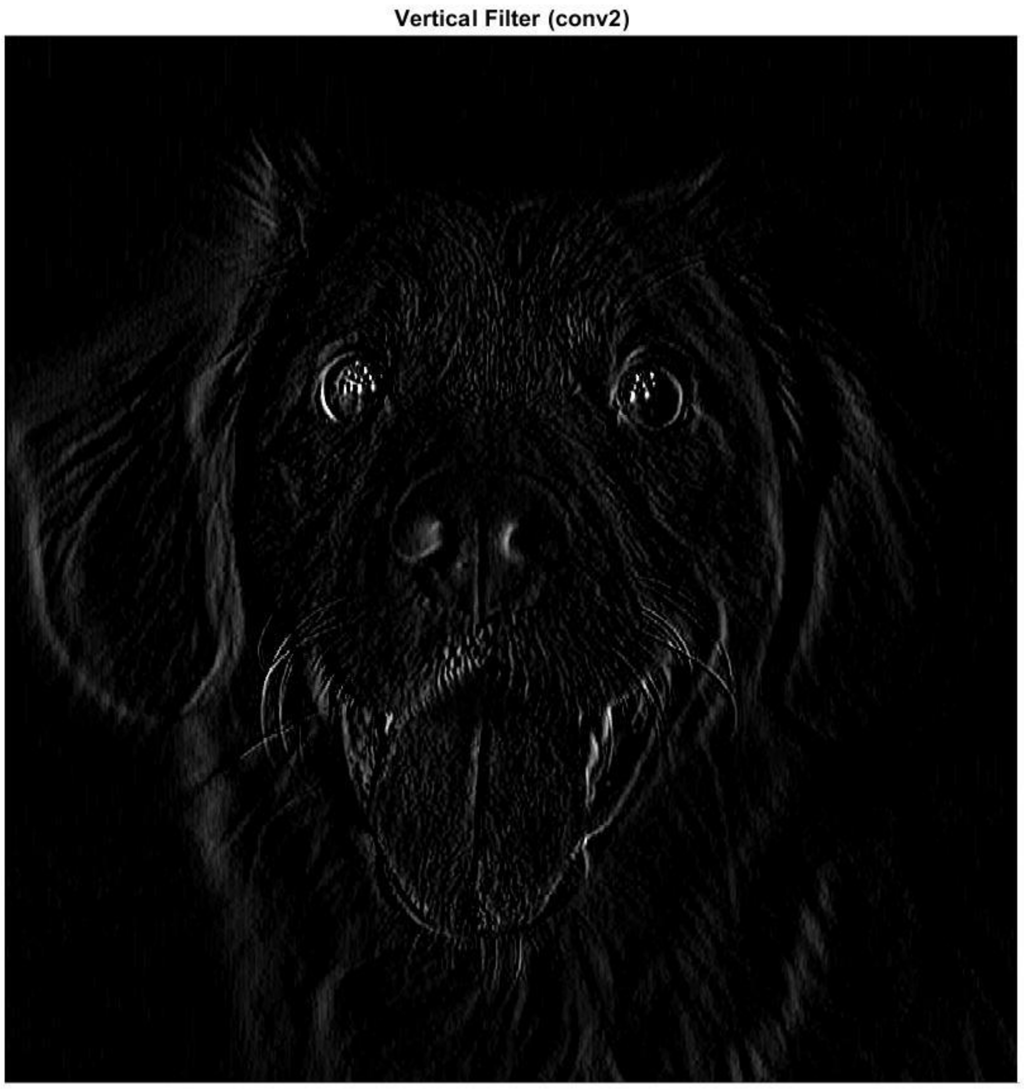
Part 2

i) As we can see from the following image, using vertical edge detector filter causes the filtered image to have intensified pixels in the vertical direction. For this we find the direction of the largest increase from light to dark and the change in the vertical direction. Therefore, change in brightness in the vertical direction is more prominent in the picture. In terms of the custom written conv2 function and the built in MATLAB conv2, the result of the convolution is very similar. Therefore, we can assure that the cust_conv2.m[1] is working as desired.

Vertical filter applied to the picture using custom written conv2 function:

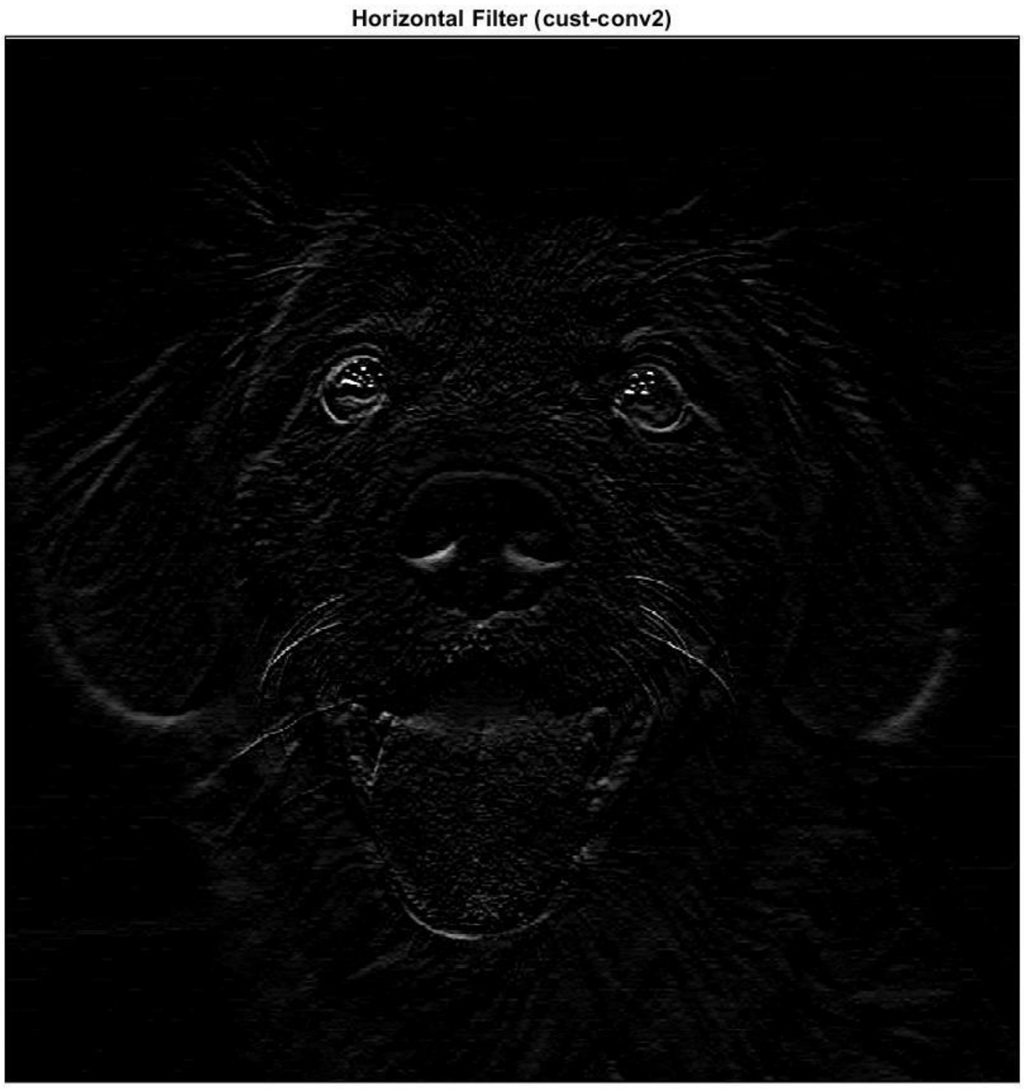


Vertical filter applied to the picture using built in conv2 function:

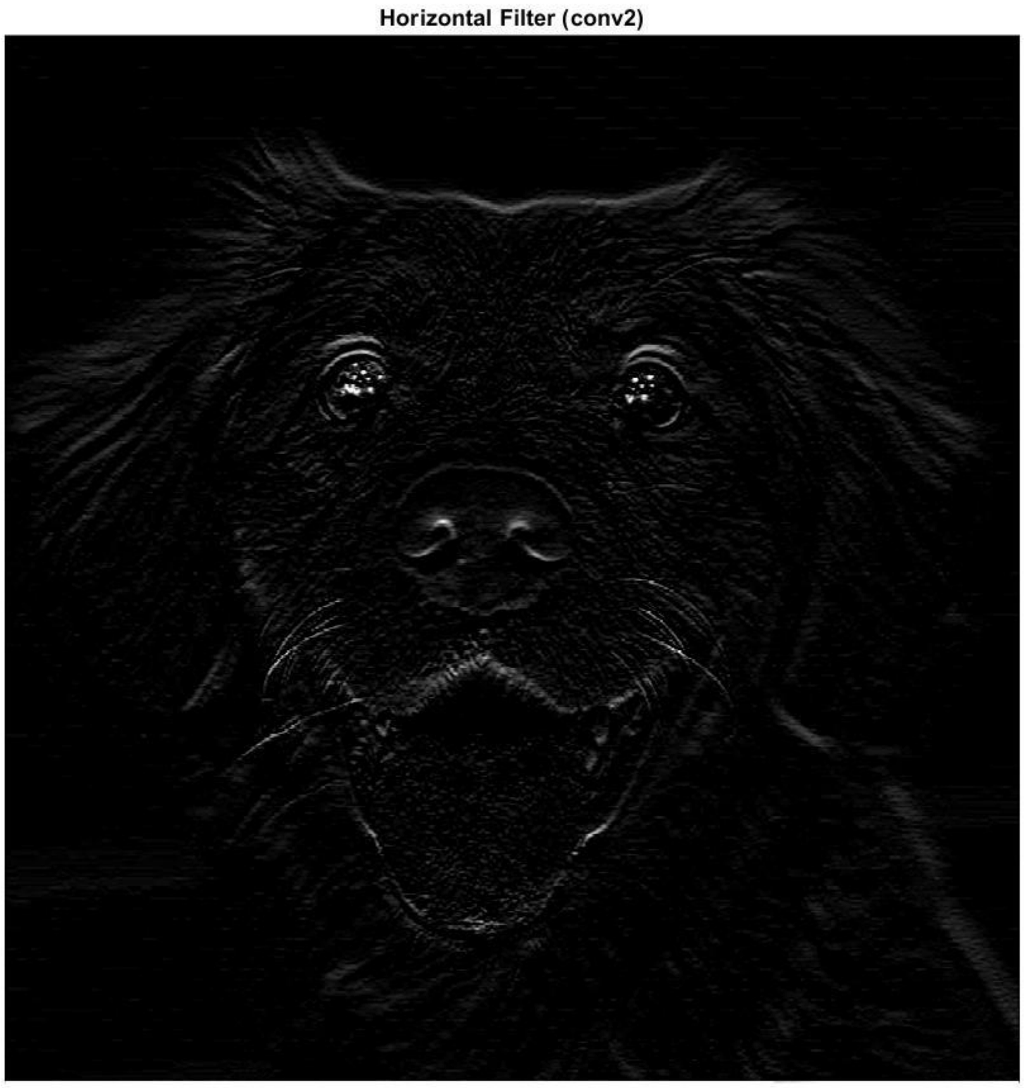


ii) The output of applying the horizontal filter shows that the filtered image has intensified pixels in the horizontal direction. This happens because we find the direction of the largest increase from light to dark and the change in the horizontal direction. Hence, the change in light in the horizontal direction is more visible. The output of the conv2 and cust_conv2.m seem to be the same for performing the convolution. Therefore, we can be sure that our custom written function for convolution works correctly.

Horizontal filter applied to the picture using custom written conv2 function:

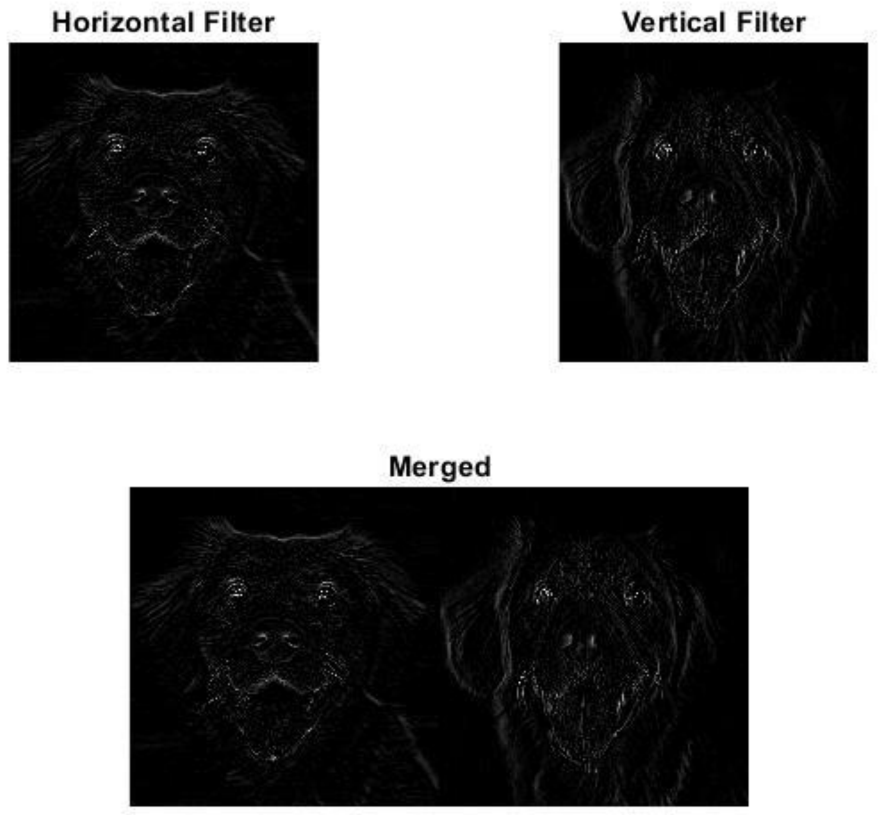


Horizontal filter applied to the picture using built in conv2 function:



iii) The merged form of vertical and horizontal filters output.

Note: Comparisons for the outputs of different filters are provided in the two previous sections.

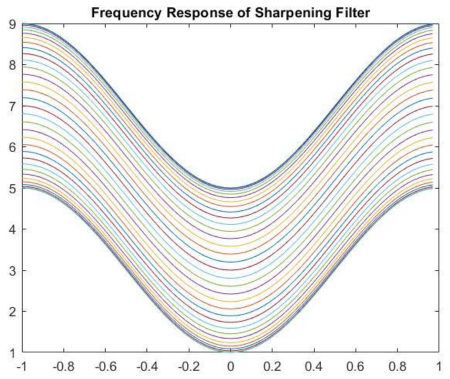


Part 3

i) The output image has gotten a sharper appearance after applying the sharpening filter. The sharpness can be noticed in the edges of the image, i.e., the whiskers.

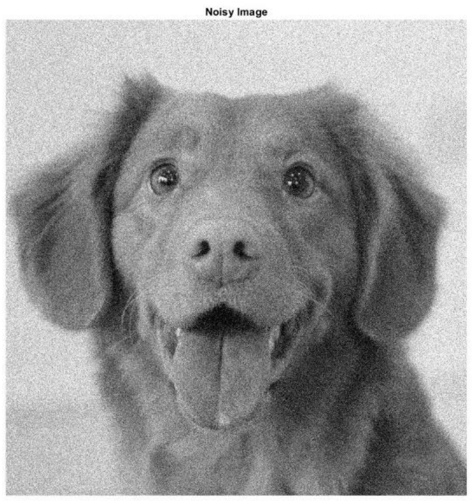
ii) It is a High Pass filter. Because the high frequency components which have a big change in pixel intensity will pass the filter. Therefore, those parts of the image are more visible and significant in the filtered image.

iii) Frequency response of the sharpening filter:



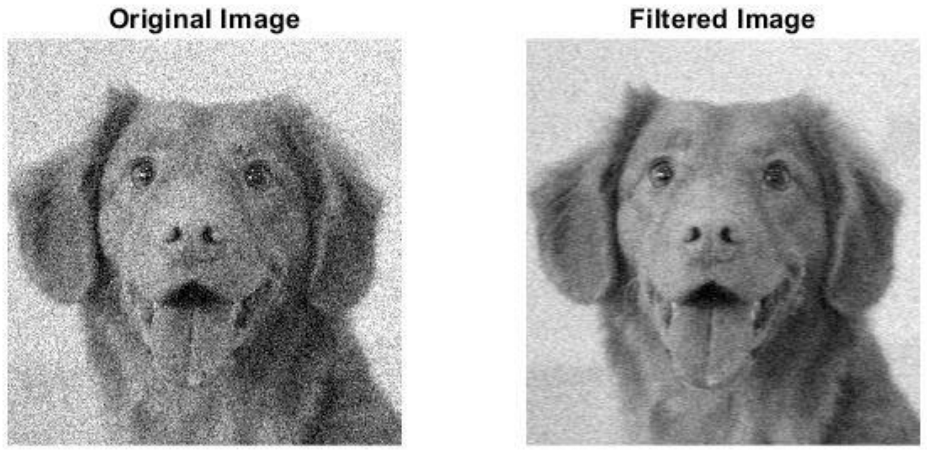
Part 4

i) **Noisy Image:**

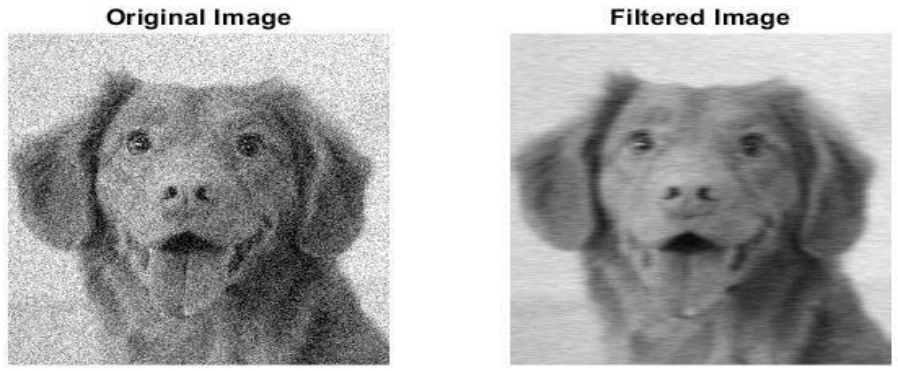


ii) Applying moving average filter:

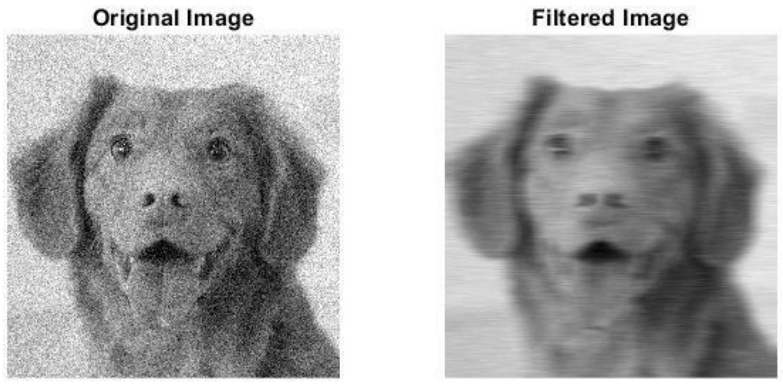
i) M = 7



ii) M = 21:



iii) M = 41



The moving average filter has helped with smoothening the noisy image. As it can be seen from the images, the larger the value of M the smoother the image becomes. As expected, the smoothening seems to be in the horizontal direction. It is expected as such because we are averaging pixels with respect to their left and right pixels values.

Although the image has become smoother, we notice that moving average has caused some side effects as well. It has caused the image to seem blurry. As the value of M increases, the image become more and more blurry. This happens because we lose the details of each individual pixel as we apply the filter. In other words, we can say that the values of pixels blend into each other. Therefore, the image has become blurry.

References

[1] A. automaticaddison, “How the sobel operator works,” *Automatic Addison*, 17-Dec-2019. [Online]. Available: https://automaticaddison.com/how-the-sobel-operator-works/. [Accessed: 20-Nov-2021].

Appendix

Part 1 (MATLAB code)

```
% Read image
img = imread('dog.jpg');

% % Individual Channels
r_channel = img(:,:,1);
g_channel = img(:,:,2);
b_channel = img(:,:,3);
%
% % ------------------------- PART 1 I -------------------------
%
% % Black color to replace channels with
% no_color = zeros(size(img, 1), size(img, 2), 'uint8');
%
% % Individual color
only_red = cat(3, r_channel, no_color, no_color);
only_green = cat(3, no_color, g_channel, no_color);
only_blue = cat(3, no_color, no_color, b_channel);
%
% % Plot
%
% Only red channel is not zero
imshow(only_red);
title('Only Red', 'FontSize', fontSize)

% Only green channel is not zero
imshow(only_green)
title('Only Green', 'FontSize', fontSize)

% Only blue channel is not zero
imshow(only_blue);
title('Only Blue', 'FontSize', fontSize)
% %
%
% % ------------------------- PART 1 II -------------------------
gray = 0.299 * r_channel + 0.587 * g_channel + 0.114 * b_channel;
imwrite(gray, "grayscale_ii.jpg", "Quality", 100);

% Plot
fontSize = 20;
imshow(gray);
title('Grayscale Image', 'FontSize', fontSize)

% ------------------------- PART 1 III -------------------------
upside_down = flipud(img);
trp = pagectranspose(img);
rotated_90 = flipud(trp);

% % Plot
imshow(rotated_90);
```

```
title('Flipped Image')

% ------------------------- PART 1 IV -------------------------
cropped = img(25:775, 50:750, :);
% imwrite(cropped, "cropped.jpg", "Quality", 100);

% % Plot
%
imshow(cropped);
title('Cropped Image')

% ------------------------- PART 1 V -------------------------
new_img = im2double(imread("cropped.jpg"));
new_img = new_img(:,:,1);
%
img_zeroed = new_img;
img_zeroed(new_img < 0.2) = 0;


% % Plot
imshow(img_zeroed, []);
title('Pixelx < 0.2 Set to 0')
```

Part 2 (MATLAB code)

```
img = im2double(imread("grayscale_ii.jpg"));
% Filters to be applied
vertical_filter = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
horizontal_filter = [-1, -2, -1; 0, 0, 0; 1, 2, 1];

% ------ Apply vertical filter with both cust_conv2 and conv2 ---------
% Apply vertical filter using cust_conv2
filtered_y = cust_conv2(img, vertical_filter);
imshow(filtered_y);
title('Vertical Filter (cust-conv2)')

% Apply vertical filter using conv2
filtered_y = conv2(img, vertical_filter);
imshow(filtered_y);
title('Vertical Filter (conv2)')


% ------ Apply horizontal filter with both cust_conv2 and conv2 ---------
filtered_x = cust_conv2(img, horizontal_filter);
imshow(filtered_x);
title('Horizontal Filter (cust-conv2)')

% Apply horizontal filter using conv2
filtered_x = conv2(img, horizontal_filter);
imshow(filtered_x);
title('Horizontal Filter (conv2)')

% Plot merged and filtered images
```

```
subplot(2, 2, 1);
imshow(filtered_x);
title('Horizontal Filter');

subplot(2, 2, 2);
imshow(filtered_y);
title('Vertical Filter');

merged = cat(2, filtered_x, filtered_y);
subplot(2, 1, 2);
imshow(merged);
title('Merged')
```

Custom conv2 function:

```
function [ output ] = cust_conv2(img, filter)
  % Create padded matrix and copy the image to it.
  new_img = zeros(size(img, 1) + 2, size(img, 2) + 2);
  new_img = cast(new_img, class(img));
  new_img(2:end-1, 2:end-1) = img;

  % Create output image matrix
  output = zeros(size(new_img));
  output = cast(output, class(img));

  for i = 2:1:size(img, 1)+1
    for j = 2:1:size(img, 2)+1
      avg = 0;
      for k = -1:1:1
        for l = -1:1:1
          avg = avg + new_img(i + k, j + l) * filter(k + 2, l + 2);
        end
      end
     output(i, j) = avg;
    end
  end
end
```

Part 3 (MATLAB code)

```
% Read the image
img = imread("grayscale_ii.jpg");
sharpening_filter = [0, -1, 0; -1, 5, -1; 0, -1, 0];

convolved_img = conv2(img, sharpening_filter);
% Normalize intensified pixels
convolved_img = uint8(convolved_img);

% Plot
imshow(convolved_img);
title('Sharepened Image')

[h, w] = freqz2(sharpening_filter);
```

```
plot(w, abs(h));
title('Frequency Response of Sharpening Filter')
```

Part 4 (MATLAB code)

```
img = im2double(imread("grayscale_ii.jpg"));
noise = normrnd(0, 0.5, size(img));
noise = noise * 0.2;
noisy = img + noise;

smoothened_7 = m_point_avg(noisy, 7);
show_images(noisy, smoothened_7);

smoothened_21 = m_point_avg(noisy, 21);
show_images(noisy, smoothened_21);

smoothened_41 = m_point_avg(noisy, 41);
show_images(noisy, smoothened_41);
```

M-Point average filter:

```
function [output] = m_point_avg(img, M)
 output = movmean(img, M, 2);
end
```

Helper plot function:

```
function show_images(img1, img2)
 subplot(1, 2, 1);
 imshow(img1);
 title('Original Image');
 subplot(1, 2, 2);
 imshow(img2);
 title('Filtered Image');
end
```