Marwadi
University

IMAGE PROCESSING
01CE0507

Unit - 4
Spatial Filters

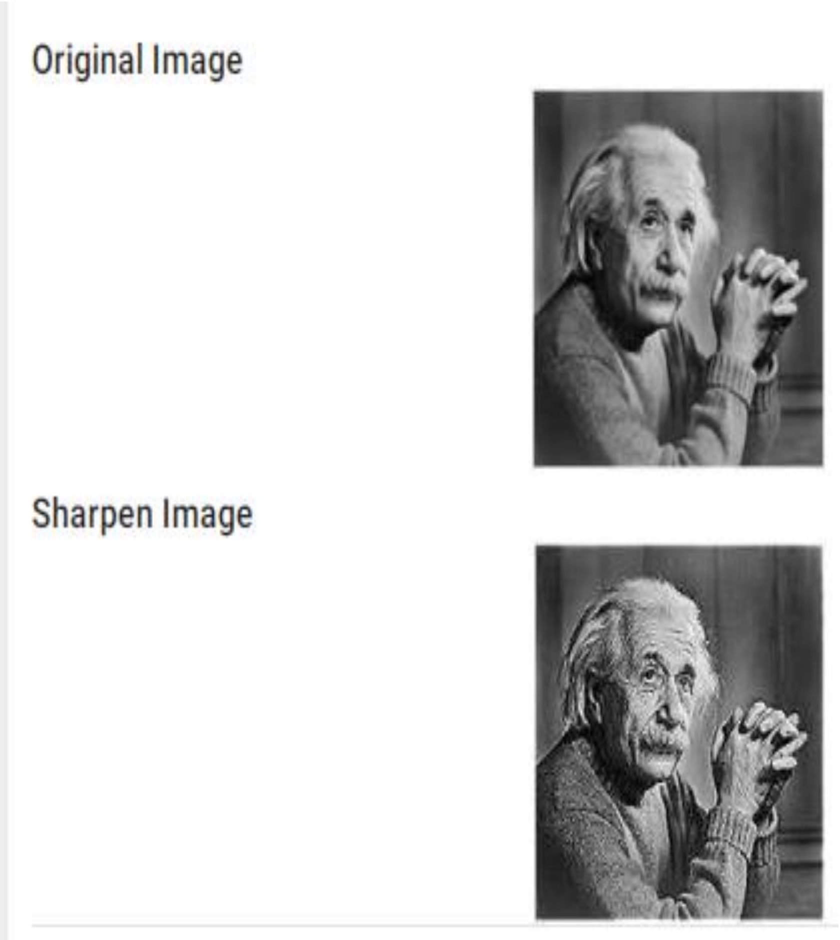
Prof. Urvi Y. Bhatt
Department of Computer Engineering

Outline

- Spatial Filtering
- Spatial Filtering techniques can be divided into two broad categories:
  - The Smoothing Spatial Filter / Low Pass Filters
  - **The Sharpening Filters / High Pass Filters**

Sharpening

- Sharpening is opposite to the blurring.
- In blurring, we reduce the edge content and in Sharpening, we increase the edge content.
- It is Useful for emphasizing transitions in image intensity



Original Image

Sharpen Image

Sharpening (Cont.)

- In order to increase the edge content in an image, we have to find edges first.
- Edges can be find by one of the any method by using any operator.
- After finding edges, we will add those edges on an image and thus the image would have more edges, and it would look sharpen.

Blurring vs. Sharpening

- Blurring/smoothing is done in spatial domain by pixel averaging in a neighbors, it is a process of integration.
- Sharpening is an inverse process, to find the difference by the neighborhood, done by spatial differentiation.

Applications of Sharpening

- In Sharpening , we try to Highlight fine details in an image
- Applications are
  - Electronic Printing
  - Medical Imaging
  - Industrial Inspections
  - Autonomous Guidance in Military Systems

What are Edges?

- We can also say that sudden changes of discontinuities in an image are called as edges.
- Significant transitions in an image are called as edges.

Edge Detection

- Edge detection refers to the process of identifying and locating sharp discontinuities in an image.
- The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene.
- Image Edge detection significantly reduces the amount of data and filters out useless information

Why detect Edges

- Most of the shape information of an image is enclosed in edges.
- So first we detect these edges in an image and by using filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Problems in Edge Detection

- There are problems like
  - False edge detection
  - Missing true edges
  - Producing thin or thick lines
  - Problems due to noise

Types of Edges

- Generally edges are of three types:
  - Horizontal Edges
  - Vertical Edges
  - Diagonal Edges

Derivative Operator

- Derivative operator is used to perform Edge Detection
- The strength of the response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied.
- Image differentiation
  - enhances edges and other discontinuities (noise)
  - deemphasizes area with slowly varying gray level values.

Derivative Operator (Cont.)

- All the derivative masks should have the following properties:
  - Opposite sign should be present in the mask.
  - Sum of mask should be equal to zero.
  - More weight means more edge detection.

Edge Detection

- Gradient Based Edge Detection / 1st derivative
  - The Gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image
  - 1st derivative sharpening produces **thicker edges** in an image
  - 1st derivative sharpening has stronger response to gray level change
- Laplacian Based Edge Detection / 2nd derivative
  - The Laplacian method searches for zero crossings in the second derivative of the image to find edges
  - 2nd derivative sharpening has stronger response to fine details, such as **thin lines** and **isolated points**.
  - 2nd derivative sharpening has double response to gray level change

Edge Detection Techniques

- **Sharpening filters / High Pass Filters / Operators used for Edge Detection / Derivative Operators or Derivative Masks**
- Sharpening filters are based on spatial differentiation

Edge Detection Techniques (Cont.)

$$\frac{\partial f}{\partial x} = f'(x)$$
$$= f(x+1) - f(x)$$
$$\frac{\partial^2 f}{\partial x^2} = f''(x)$$
$$= f(x+1)$$
$$+ f(x-1) - 2f(x)$$

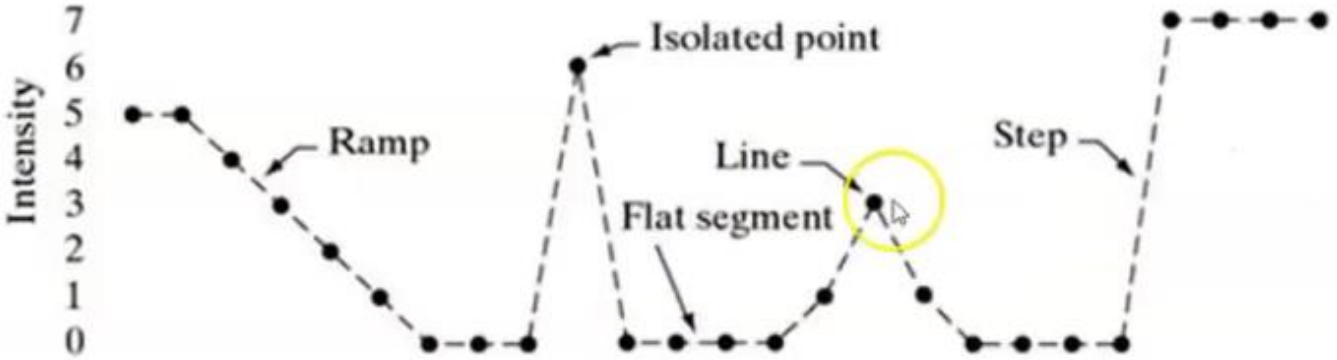


Image strip 5 5 4 3 2 1 0 0 0 6 0 0 0 0 1 3 1 0 0 0 0 7 7 7 7 • •

First derivative −1 −1 −1 −1 −1 0 0 6 −6 0 0 0 1 2 −2 −1 0 0 0 7 0 0 0

Second derivative −1 0 0 0 0 1 0 6 −12 6 0 0 1 1 −4 1 1 0 0 7 −7 0 0

Second Order derivative identify more no of edges as compare to First order derivative. Justify the Statement.

Edge Detection Techniques (Cont.)

- **Gradient Based Operators / 1st derivative**
  - Robert Operator / Roberts Cross Edge Detector
  - Prewitt Operator
  - Sobel Operator
  - Robinson Compass Masks / Direction Mask / Robinson Compass Operator
  - Krisch Compass Mask / Krisch Compass Operator
- **Laplacian Based Operators / 2nd derivative**
  - Laplacian Operator
  - Laplacian of Gaussion Operator

Roberts Cross Edge Detector / Robert Operator

- This gradient-based operator computes the sum of squares of the differences between diagonally adjacent pixels in an image through discrete differentiation.
- Then the gradient approximation is made. It uses the following 2 x 2 kernels or masks –

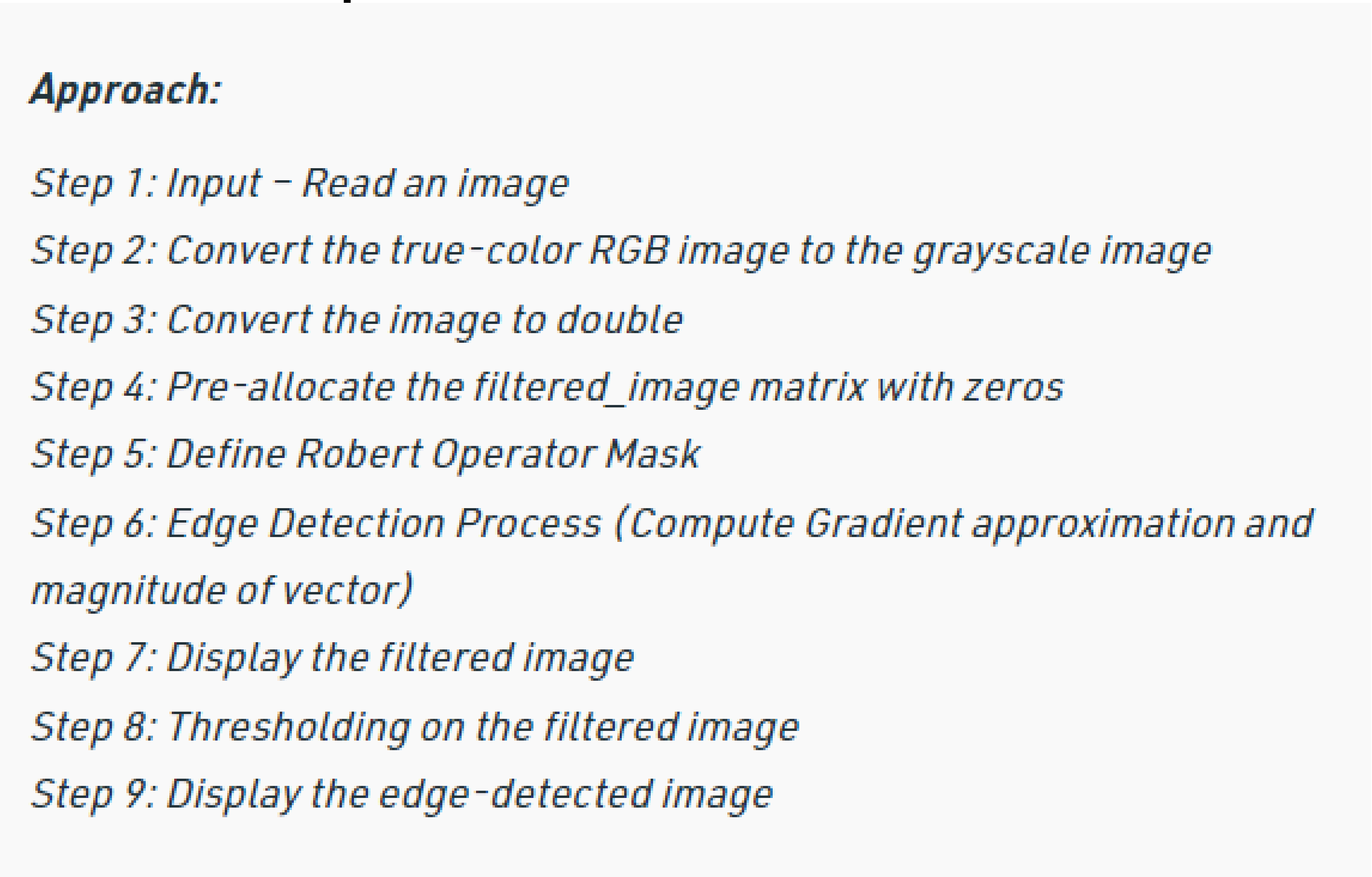
+1	0
0	-1

0	+1
-1	0

Roberts Cross Edge Detector / Robert Operator

- Advantages:
  - Detection of edges and orientation are very easy
  - Diagonal direction points are preserved
- Limitations:
  - Very sensitive to noise
  - Not very accurate in edge detection

Roberts Cross Edge Detector / Robert Operator in MATLAB



Approach:

Step 1: Input – Read an image

Step 2: Convert the true-color RGB image to the grayscale image

Step 3: Convert the image to double

Step 4: Pre-allocate the filtered_image matrix with zeros

Step 5: Define Robert Operator Mask

Step 6: Edge Detection Process (Compute Gradient approximation and magnitude of vector)

Step 7: Display the filtered image

Step 8: Thresholding on the filtered image

Step 9: Display the edge-detected image

Roberts Cross Edge Detector / Robert Operator in MATLAB

```
% Displaying Input Image
input_image = imread('peppers.png');
figure,
 imshow(input_image);
 title('Input Image');

% Convert the truecolor RGB image to the grayscale image
input_image = rgb2gray(input_image);
% Convert the image to double
input_image = double(input_image);

% Pre-allocate the filtered_image matrix with zeros
filtered_image = zeros(size(input_image));

% Robert Operator Mask
Mx = [1 0; 0 -1];
My = [0 1; -1 0];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(1, 1)
% The mask is of 2x2, so we need to traverse
% to filtered_image(size(input_image, 1) - 1
%, size(input_image, 2) - 1)
for i = 1:size(input_image, 1) - 1
        for j = 1:size(input_image, 2) - 1

                % Gradient approximations
                Gx = sum(sum(Mx.*input_image(i:i+1, j:j+1)));
                Gy = sum(sum(My.*input_image(i:i+1, j:j+1)));

                % Calculate magnitude of vector
                filtered_image(i, j) = sqrt(Gx.^2 + Gy.^2);
        end
End

% Displaying Filtered Image
filtered_image = uint8(filtered_image);
figure, imshow(filtered_image); title('Filtered Image');

% Define a threshold value
thresholdValue = 10; % varies between [0 255]
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;

% Displaying Output Image
output_image = imbinarize(output_image);
figure,
imshow(output_image); title('Edge Detected Image');
```

Prewitt Operator

- Prewitt operator is used for detecting edges horizontally and vertically.
- Prewitt operator provides us two masks
  - for detecting edges in horizontal direction
  - for detecting edges in an vertical direction.

Vertical Direction

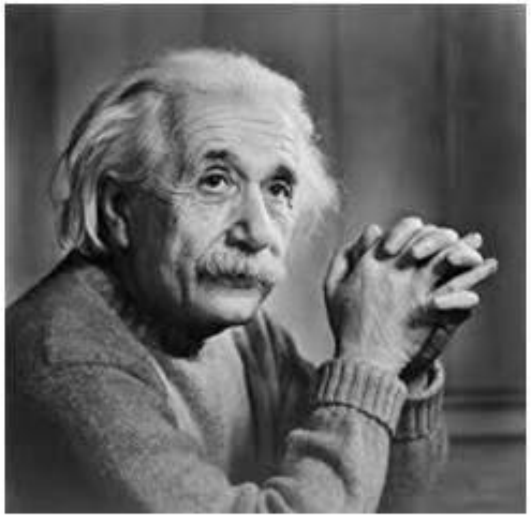
-1	0	1
-1	0	1
-1	0	1

Horizontal Direction

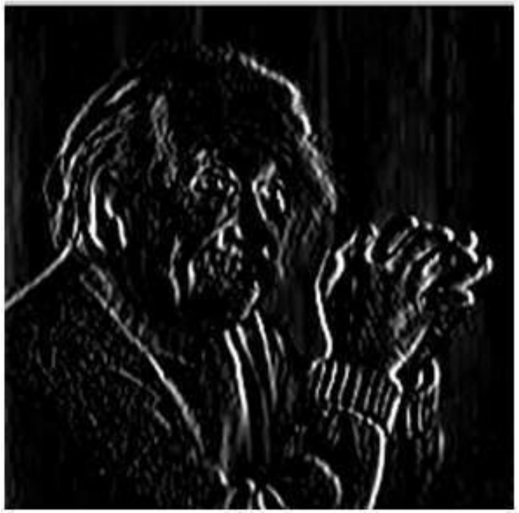
-1	-1	-1
0	0	0
1	1	1

Prewitt Operator (Cont.)

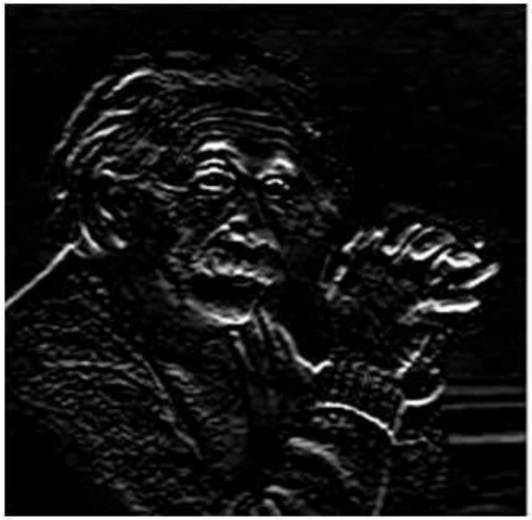
Original Image



Apply Vertical Direction Mask



Apply Horizontal Direction Mask



Prewitt Operator (Cont.)

- How to apply Vertical Direction Mask?
  - When we apply this mask on the image it prominent vertical edges.
  - It simply works like as first order derivate and calculates the difference of pixel intensities in a edge region.
  - As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge.
  - This increase the edge intensity and it become enhanced comparatively to the original image.

Prewitt Operator (Cont.)

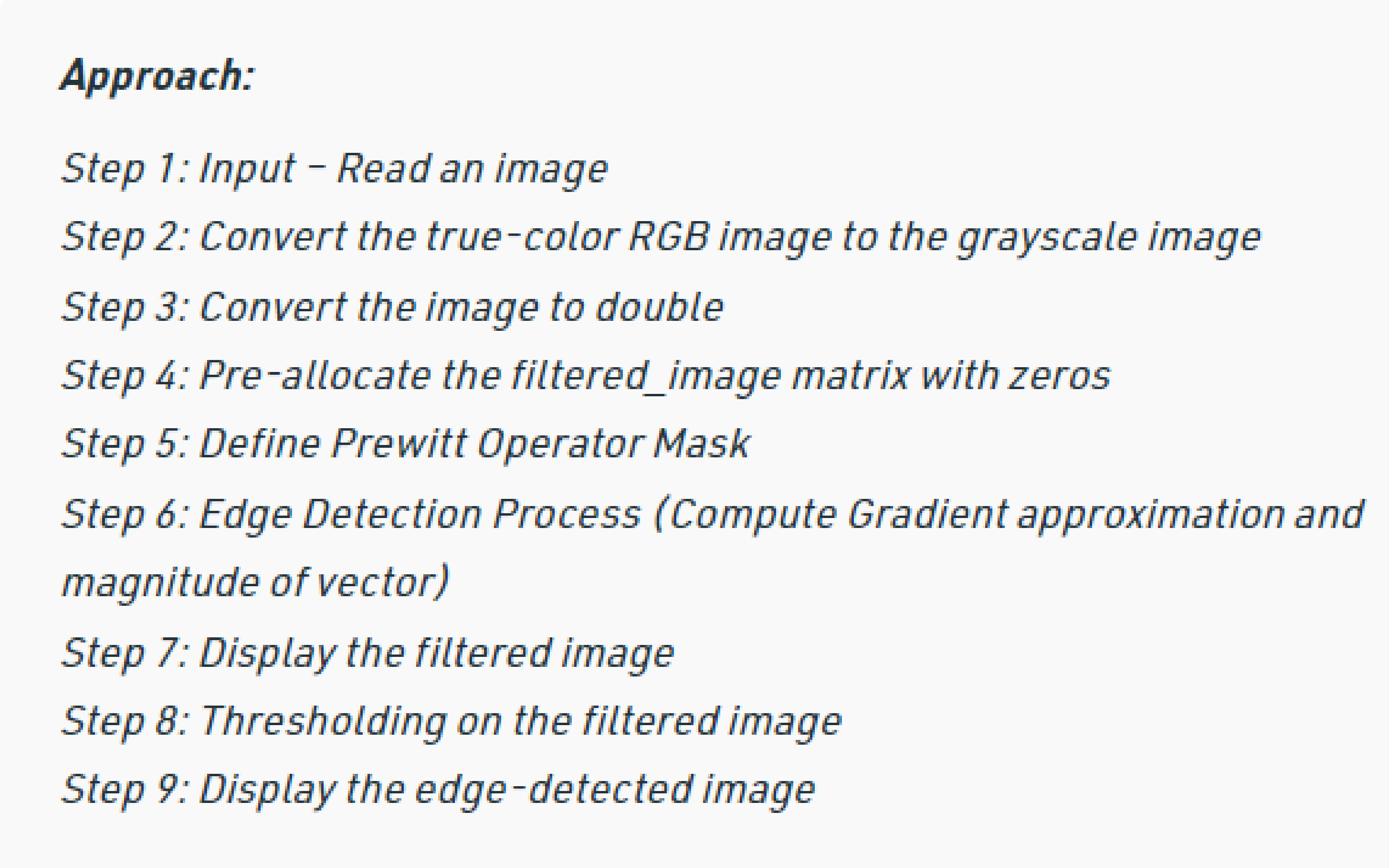
- How to apply Horizontal Direction Mask?
  - This mask will prominent the horizontal edges in an image.
  - As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge.
  - Thus increasing the sudden change of intensities and making the edge more visible.

Prewitt Operator (Cont.)

- Advantages:
  - – Good performance on detecting vertical and horizontal edges
  - – Best operator to detect the orientation of an image

- Limitations:
  - – The magnitude of coefficient is fixed and cannot be changed
  - – Diagonal direction points are not preserved always

Prewitt Operator in MATLAB



Approach:

Step 1: Input – Read an image

Step 2: Convert the true-color RGB image to the grayscale image

Step 3: Convert the image to double

Step 4: Pre-allocate the filtered_image matrix with zeros

Step 5: Define Prewitt Operator Mask

Step 6: Edge Detection Process (Compute Gradient approximation and magnitude of vector)

Step 7: Display the filtered image

Step 8: Thresholding on the filtered image

Step 9: Display the edge-detected image

Prewitt Operator in MATLAB

```
% Displaying Input Image
input_image = imread('peppers.png');
figure,
imshow(input_image);
title('Input Image');

% Convert the truecolor RGB image to the grayscale image
input_image = rgb2gray(input_image);

% Convert the image to double
input_image = double(input_image);

% Pre-allocate the filtered_image matrix with zeros
filtered_image = zeros(size(input_image));

% Prewitt Operator Mask
Mx = [-1 0 1; -1 0 1; -1 0 1];
My = [-1 -1 -1; 0 0 0; 1 1 1];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(2, 2)
% The mask is of 3x3, so we need to traverse
% to filtered_image(size(input_image, 1) - 2
%, size(input_image, 2) - 2)
% Thus we are not considering the borders.
for i = 1:size(input_image, 1) - 2
        for j = 1:size(input_image, 2) - 2
                % Gradient approximations
                Gx = sum(sum(Mx.*input_image(i:i+2, j:j+2)));
                Gy = sum(sum(My.*input_image(i:i+2, j:j+2)));
                % Calculate magnitude of vector
                filtered_image(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);
        end
End


% Displaying Filtered Image
filtered_image = uint8(filtered_image);
figure, imshow(filtered_image); title('Filtered Image');

% Define a threshold value
thresholdValue = 100; % varies between [0 255]
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;

% Displaying Output Image
output_image = imbinarize(output_image);
figure,
imshow(output_image);
title('Edge Detected Image');
```

Sobel Operator

- The Sobel Operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection.
- Like Prewitt operator Sobel Operator is also used to detect two kinds of edges in an image:
  - Vertical direction
  - Horizontal direction

Sobel Operator (Cont.)

Vertical Direction

-1	0	1
-2	0	2
-1	0	1

Horizontal Direction

-1	-2	-1
0	0	0
1	2	1

Sobel Operator (Cont.)

- Applying more weight to mask
  - Now we can also see that if we apply more weight to the mask, the more edges it will get for us.
  - There is no fixed coefficients in sobel operator, so here is another weighted operator

Vertical Direction

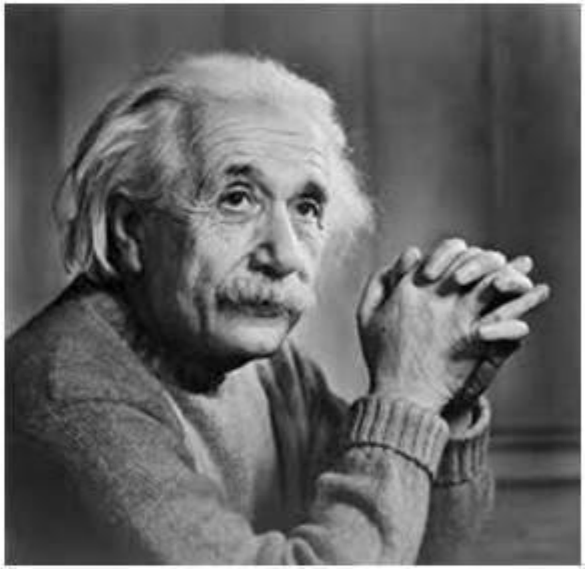
-1	0	1
-5	0	5
-1	0	1

Horizontal Direction

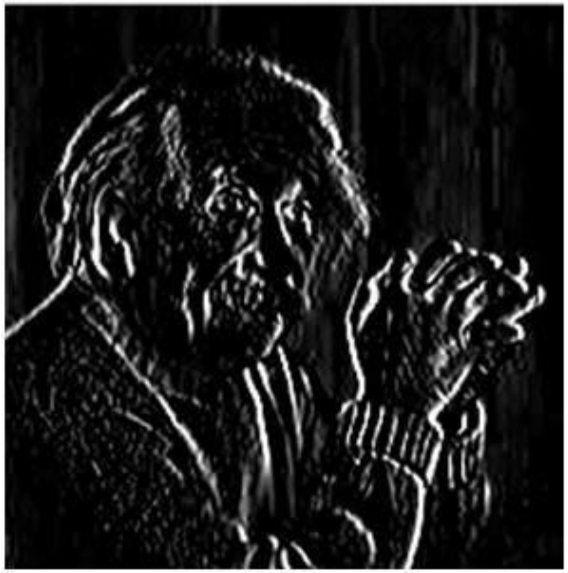
-1	-5	-1
0	0	0
1	5	1

Sobel Operator (Cont.)

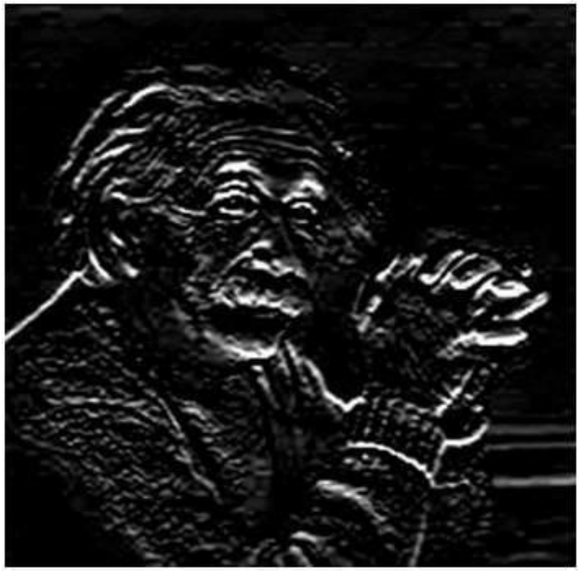
Original Image



Apply Vertical Direction Mask



Apply Horizontal Direction Mask



Sobel Operator (Cont.)

- How to apply Vertical Direction Mask?
  - When we apply this mask on the image it prominent vertical edges. It simply works like as first order derivate and calculates the difference of pixel intensities in a edge region.
  - As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. Also the center values of both the first and third column is 2 and -2 respectively.
  - This give more weight age to the pixel values around the edge region. This increase the edge intensity and it become enhanced comparatively to the original image.

Sobel Operator (Cont.)

- How to apply Horizontal Direction Mask?
  - This mask will prominent the horizontal edges in an image.
  - It also works on the principle of above mask and calculates difference among the pixel intensities of a particular edge.
  - As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge.
  - Thus increasing the sudden change of intensities and making the edge more visible.

Sobel Operator (Cont.)

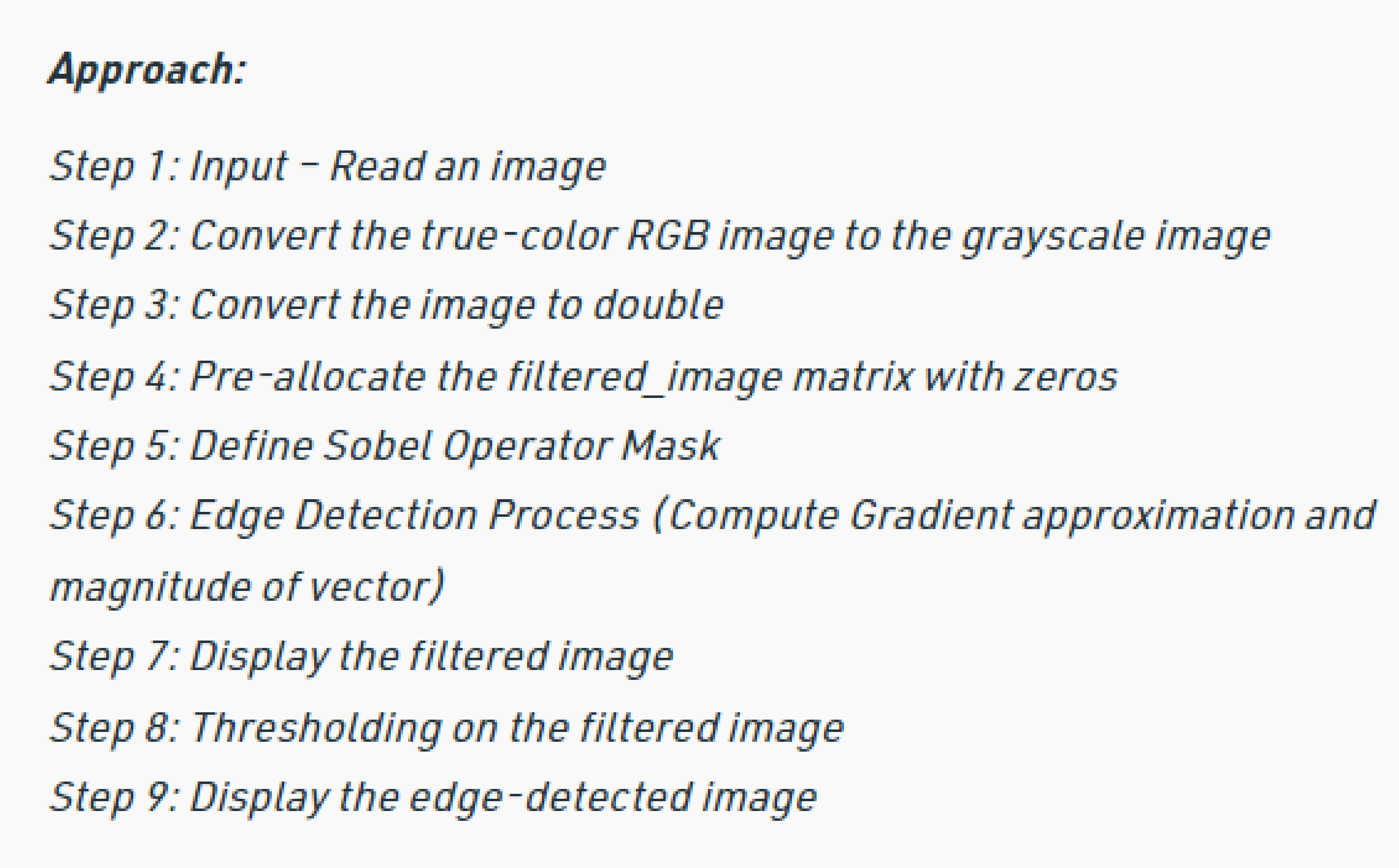
- Difference with Prewitt Operator
  - The major difference is that in sobel operator the coefficients of masks are not fixed and they can be adjusted according to our requirement unless they do not violate any property of derivative masks.

Sobel Operator (Cont.)

- Advantages:
  - Simple and time efficient computation
  - Very easy at searching for smooth edges

- Limitations:
  - Diagonal direction points are not preserved always
  - Sensitive to noise
  - Not very accurate in edge detection
  - Detect with thick and rough edges does not give appropriate results

Sobel Operator in MATLAB



Approach:

Step 1: Input – Read an image

Step 2: Convert the true-color RGB image to the grayscale image

Step 3: Convert the image to double

Step 4: Pre-allocate the filtered_image matrix with zeros

Step 5: Define Sobel Operator Mask

Step 6: Edge Detection Process (Compute Gradient approximation and magnitude of vector)

Step 7: Display the filtered image

Step 8: Thresholding on the filtered image

Step 9: Display the edge-detected image

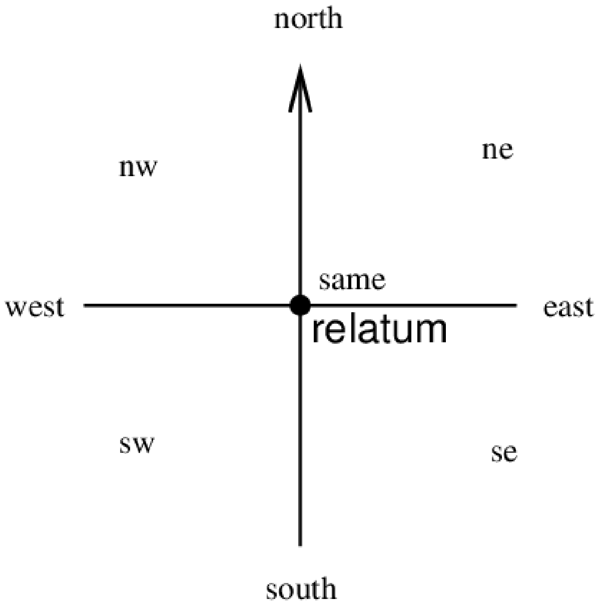
Sobel Operator in MATLAB

```
% Displaying Input Image
input_image = imread('peppers.png');
figure,
imshow(input_image);
title('Input Image');

% Convert the truecolor RGB image to the grayscale image
input_image = rgb2gray(input_image);

% Convert the image to double
input_image = double(input_image);

% Pre-allocate the filtered_image matrix with zeros
filtered_image = zeros(size(input_image));

% Sobel Operator Mask
Mx = [-1 0 1; -2 0 2; -1 0 1];
My = [-1 -2 -1; 0 0 0; 1 2 1];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(2, 2)
% The mask is of 3x3, so we need to traverse
% to filtered_image(size(input_image, 1) - 2
%, size(input_image, 2) - 2)
% Thus we are not considering the borders.
```

```
for i = 1:size(input_image, 1) - 2
        for j = 1:size(input_image, 2) - 2
                % Gradient approximations
                Gx = sum(sum(Mx.*input_image(i:i+2, j:j+2)));
                Gy = sum(sum(My.*input_image(i:i+2, j:j+2)));
                % Calculate magnitude of vector
                filtered_image(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);
        end
End

% Displaying Filtered Image
filtered_image = uint8(filtered_image);
figure, imshow(filtered_image); title('Filtered Image');

% Define a threshold value
thresholdValue = 100; % varies between [0 255]
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;

% Displaying Output Image
output_image = imbinarize(output_image);
figure,
imshow(output_image);
title('Edge Detected Image');
```

Robinson Compass Masks / Direction Mask / Robinson Compass Operator

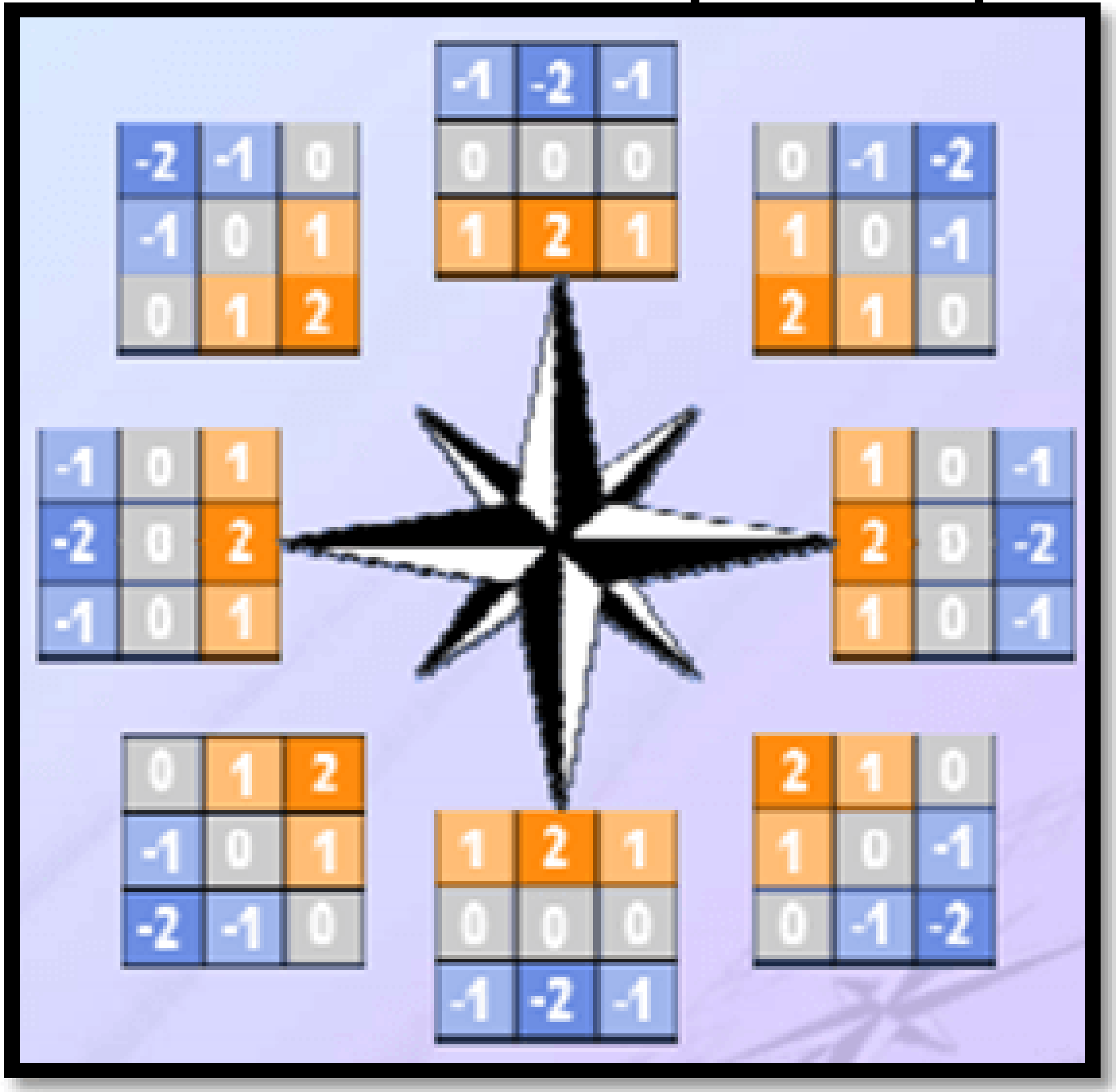
- Robinson compass masks are another type of derivate mask which is used for edge detection.
- In this operator we take one mask and rotate it in all the 8 compass major directions that are following:
  - North
  - North West
  - West
  - South West
  - South
  - South East
  - East
  - North East



Robinson Compass Masks / Direction Mask / Robinson Compass Operator (Cont.)

- It does not have any fixed mask.
- We have to rotate the mask to find an edge in any one direction above mention.
- Masks are rotated from zero columns on the basis of direction.

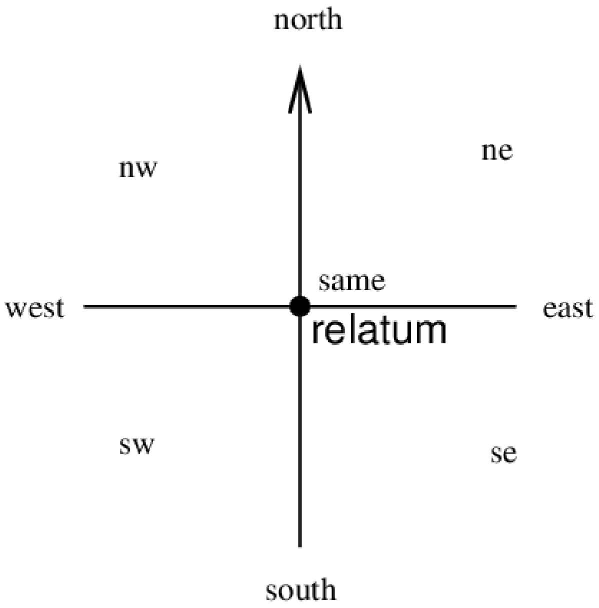
Robinson Compass Masks / Direction Mask / Robinson Compass Operator (Cont.)



$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

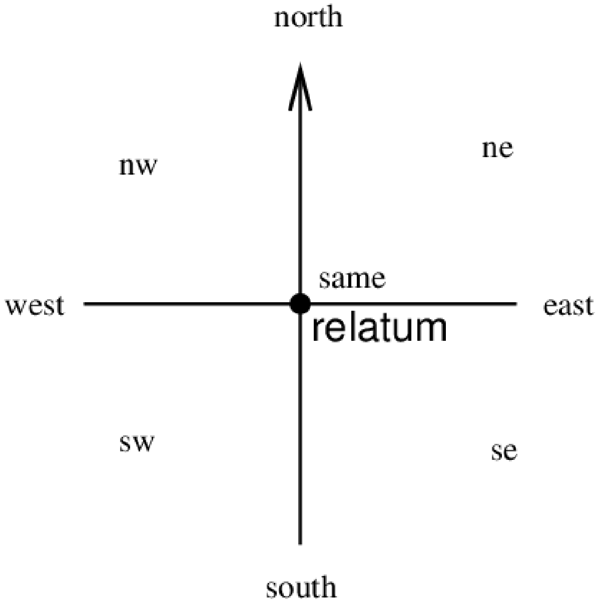


Krisch Compass Mask / Krisch Compass Operator

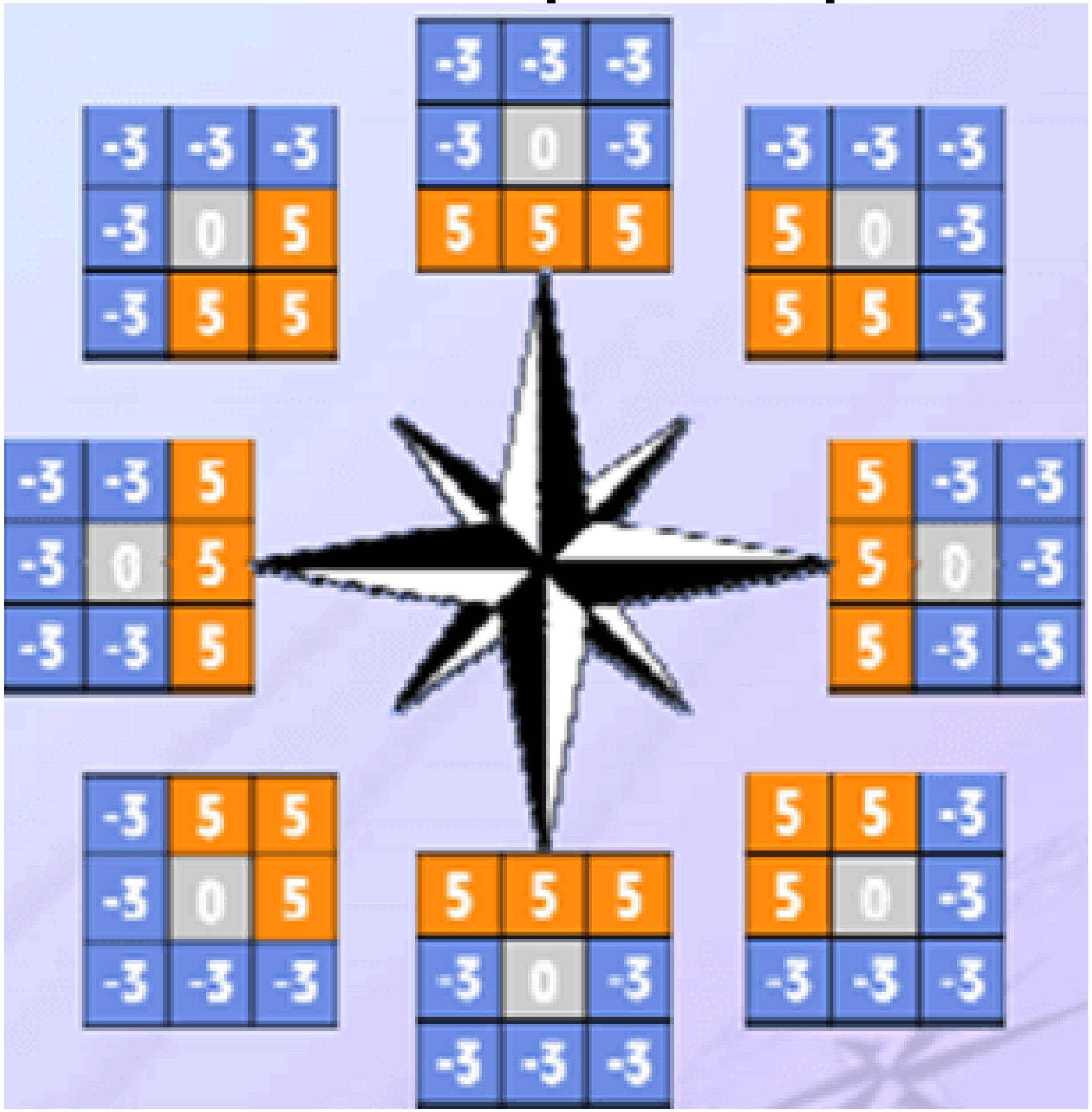
- Kirsch Compass Mask is also a derivative mask which is used for finding edges.
- This is also like Robinson compass find edges in all the eight directions of a compass.
- The only difference between Robinson and kirsch compass masks is that in Robinson we have a standard mask but in Kirsch we change the mask according to our own requirements.

Krisch Compass Mask / Krisch Compass Operator (Cont.)

- With the help of Kirsch Compass Masks we can find edges in the following eight directions.
  - North
  - North West
  - West
  - South West
  - South
  - South East
  - East
  - North East



Krisch Compass Mask / Krisch Compass Operator (Cont.)



-3	-3	-3
-3	0	5
-3	5	5

-3	-3	-3
-3	0	-3
5	5	5

-3	-3	-3
5	0	-3
5	5	-3

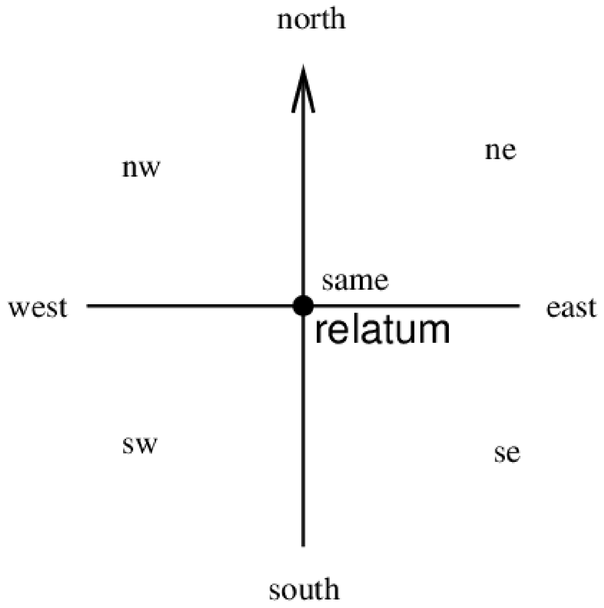
-3	-3	5
-3	0	5
-3	-3	5

5	-3	-3
5	0	-3
5	-3	-3

-3	5	5
-3	0	5
-3	-3	-3

5	5	5
-3	0	-3
-3	-3	-3

5	5	-3
5	0	-3
-3	-3	-3



Laplacian Operator

- Laplacian Operator is also a derivative operator which is used to find edges in an image.
- The major difference between Laplacian and other operators like Prewitt, Sobel, Robinson and Kirsch is that these all are first order derivative masks but Laplacian is a second order derivative mask.

Laplacian Operator (Cont.)

- Another difference between Laplacian and other operators is that unlike other operators Laplacian didn’t take out edges in any particular direction but it take out edges in following classification.
  - Inward Edges
  - Outward Edges
- In 1st order derivative filters, we detect the edge along with horizontal and vertical directions separately and then combine both. But using the Laplacian filter we detect the edges in the whole image at once.

Laplacian Operator (Cont.)

- Disadvantages:
  - We should note that first derivative operators exaggerate the effects of noise. Second derivatives will exaggerate noise twice as much.
  - No directional information about the edge is given.

Laplacian Operator (Cont.)

- In this mask we have two further classifications
  - Positive Laplacian Operator
  - Negative Laplacian Operator

- **Note:** The sum of all values of the filter is always 0.

Positive Laplacian Operator

- In Positive Laplacian we have standard mask in which center element of the mask should be negative and corner elements of mask should be zero.
- Positive Laplacian Operator is use to take out outward edges in an image.

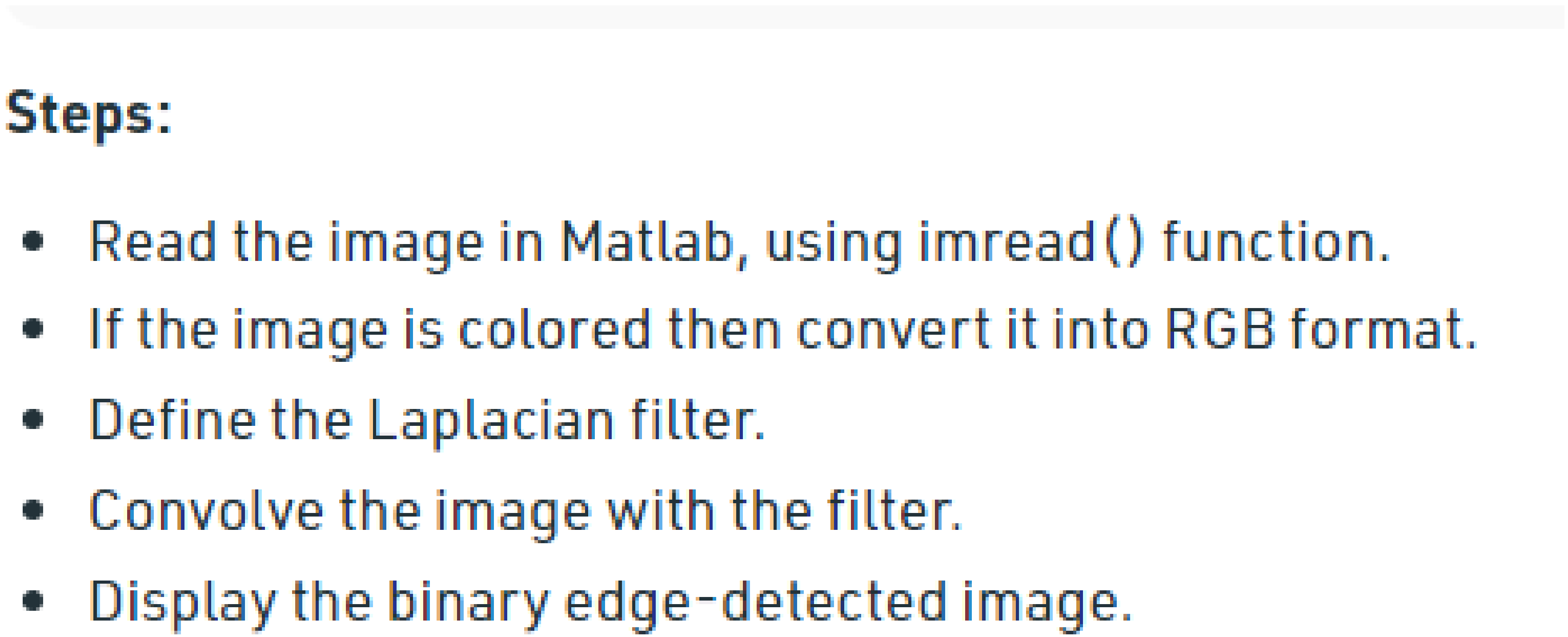
0	1	0
1	-4	1
0	1	0

Negative Laplacian Operator

- In negative Laplacian operator we also have a standard mask, in which center element should be positive. All the elements in the corner should be zero and rest of all the elements in the mask should be -1.
- Negative Laplacian operator is use to take out inward edges in an image

0	-1	0
-1	4	-1
0	-1	0

Laplacian Operator in MATLAB



Steps:

- Read the image in Matlab, using imread() function.
- If the image is colored then convert it into RGB format.
- Define the Laplacian filter.
- Convolve the image with the filter.
- Display the binary edge-detected image.

Laplacian Operator in MATLAB (Cont.)

- Code:

Laplacian of Gaussian Filter

- The Laplacian filter is used to detect the edges in the images. But it has a disadvantage over the noisy images. It amplifies the noise in the image.
- Hence, first, we use a Gaussian filter on the noisy image to smoothen it and then subsequently use the Laplacian filter for edge detection.

Laplacian of Gaussian Filter in MATLAB

Thank you