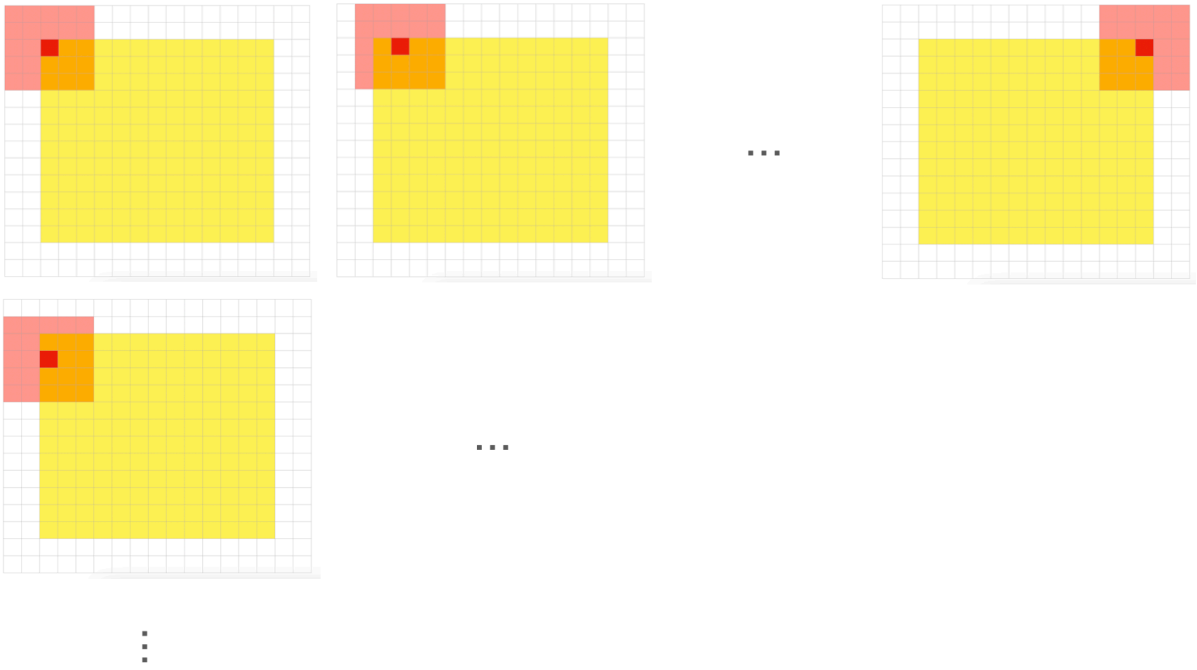
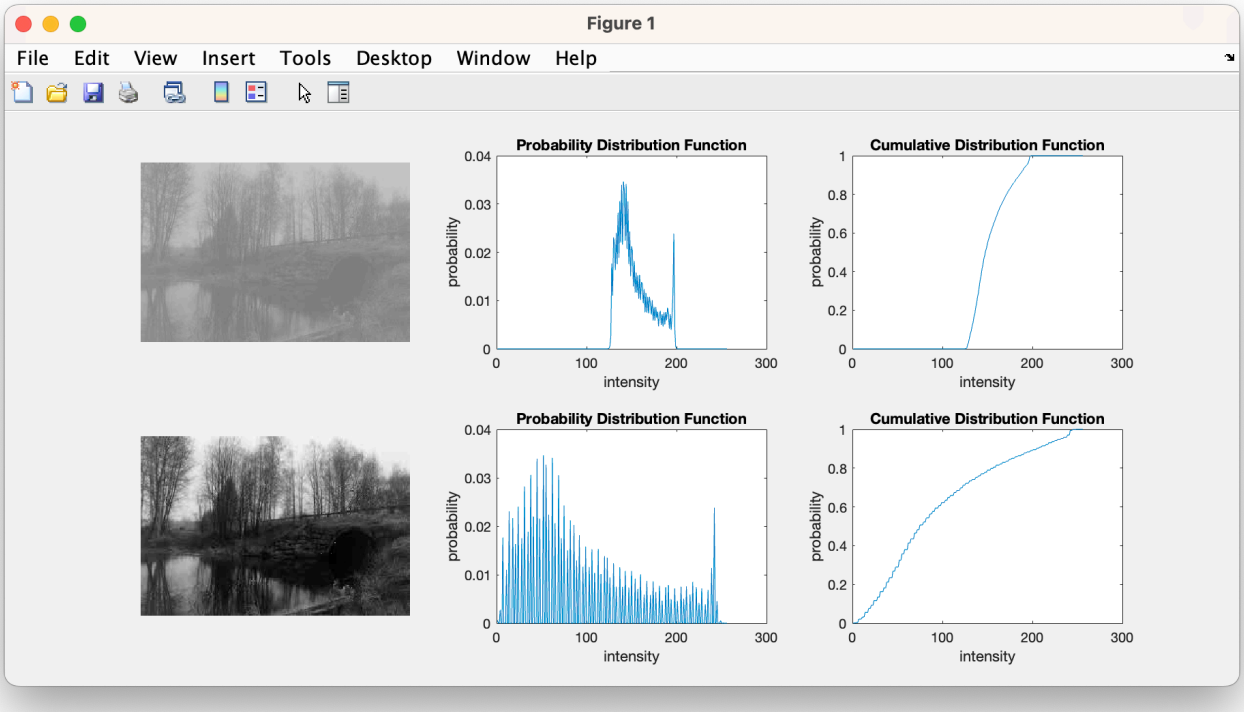
Q1

1. Read image
2. Add padding to image
    - padding size = $(\frac{(\text{filter}_x-1)}{2}, \frac{(\text{filter}_y-1)}{2})$
    - padded image size = $(\text{img}_x + \text{filter}_x - 1, \text{img}_y + \text{filter}_y - 1,)$
3. 2D sliding window using mask on padded image (illustrated in following diagram)



    - Calculate the result using specific method (average, median, gaussian filter) coverd by the mask and padded image; save that result into the center of mask (red)
4. Store all the centric pixel (red) results with its corresponding position into the new image
5. New image is filtered image

Q2




```m
% main.m

img = imread('tree.bmp');
tiledlayout(2,3);

nexttile;
imshow(img);

nexttile;
plot_pdf(img);

nexttile;
plot_cdf(img);

stretched = histo_stretch(img, 125, 200, 0, 255);

nexttile;
imshow(stretched);

nexttile;
plot_pdf(stretched);

nexttile;
plot_cdf(stretched);
```

```m
% plot_histogram.m
```

```
function histogram = plot_histogram(img)
  [~, ~, channel] = size(img);
  hist = zeros(1, 256, channel);
  for g = 0:255
    hist(1, g+1, :) = sum(sum(img == g, 1), 2);
    end

  if channel == 3
    plot(0:255, hist(1, :, 1), 'r', 0:255, hist(1, :, 2), 'g', 0:255, hist(1, :, 3),
'b');
        title('Histogram (colored)');
      xlabel('g');
      ylabel('pixels');
  else
    plot(0:255, hist(1, :, 1));
        title('Histogram (monochrome)');
      xlabel('g');
      ylabel('pixels');
    end
    histogram = hist;
end
```

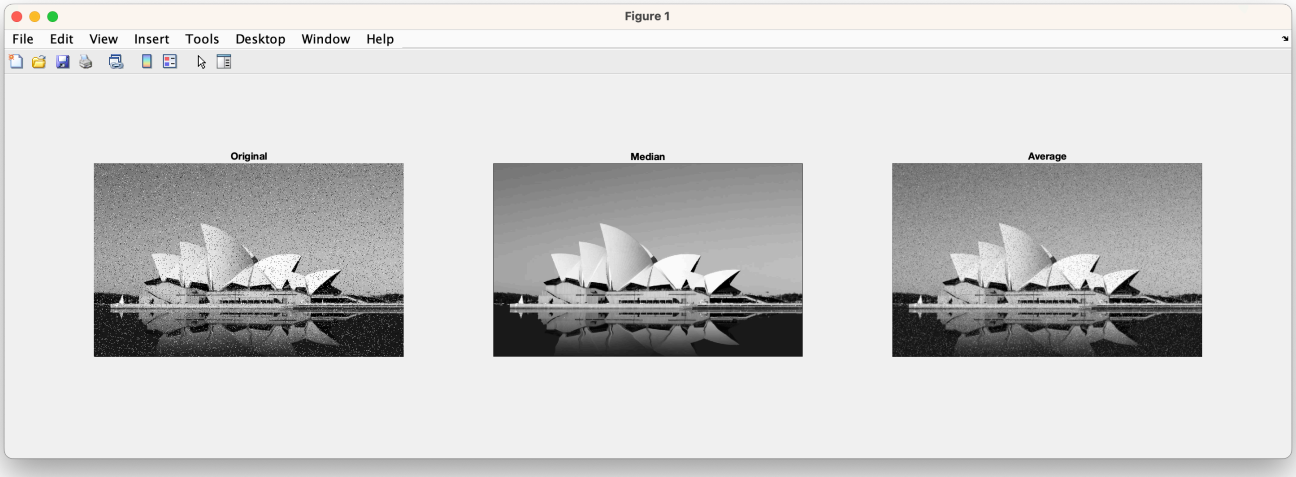
```
% plot_pdf.m

function distribution = plot_pdf(img)
  [row, column, ~] = size(img);
  hist = plot_histogram(img);
  pdf = hist / (row * column);
  plot(pdf);
  title('Probability Distribution Function');
  xlabel('intensity');
  ylabel('probability');
  distribution = pdf;
end
```

```
% plot_cdf.m

function distribution = plot_cdf(img)
  pdf = plot_pdf(img);
  cdf = zeros(256, 1);
  for i = 1:256
    cdf(i) = sum(pdf(1:i));
  end
  plot(cdf);
  title('Cumulative Distribution Function');
  xlabel('intensity');
```

```
  ylabel('probability');
  distribution = cdf;
end
```

```
% histo_stretch.m

function img_strch = histo_stretch(img, a, b, A, B)

  if length(size(img)) == 3
    img = rgb2gray(img);
  end

  img_strch = double(img);
  mask = double((img >= a) & (img <= b));
  img_temp = floor((B - A) * (img_strch - a) / (b - a) + A);
  img_strch = img_temp .* mask + img_strch .* (1 - mask);
  img_strch = uint8(img_strch);

end
```

Q3




```
% main.m

img = imread('operahouse.tif');
med = median_filter(img);
avg = average_filter(img, 3);
imwrite(med, 'operahouse_med.tif');
imwrite(avg, 'operahouse_avg.tif');

tiledlayout(1, 3);

nexttile;
```

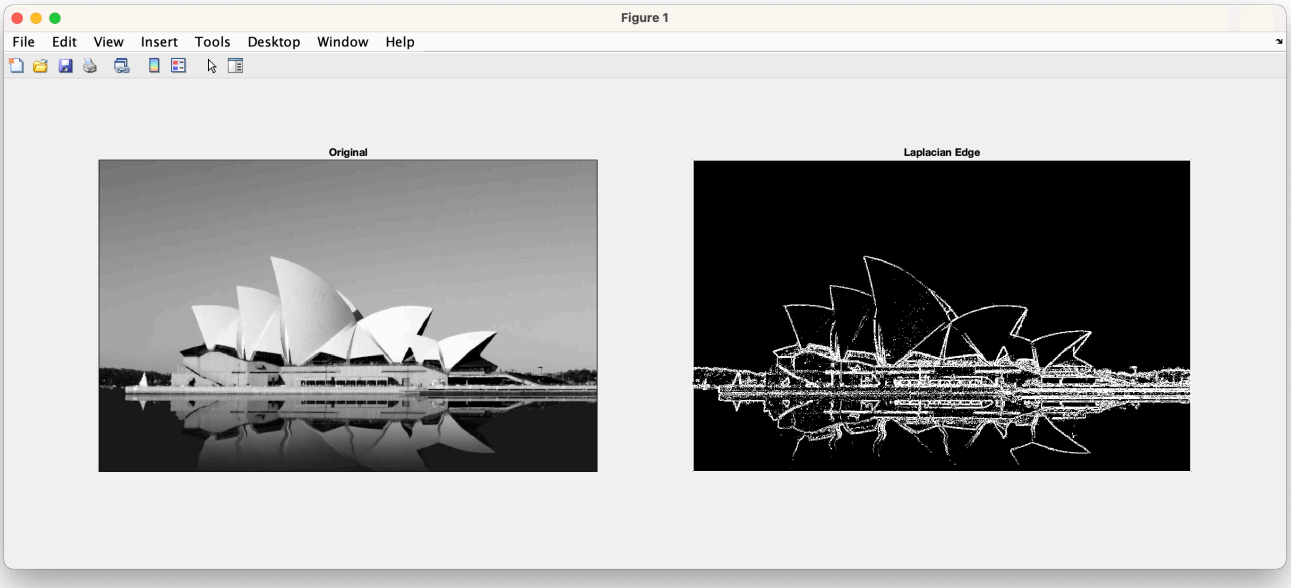
```
imshow(img);
title('Original');

nexttile;
imshow(med);
title('Median');

nexttile;
imshow(avg);
title('Average');
```

```
% median_filter.m

function new_img = median_filter(img)

    [row, column] = size(img);

    new_img = zeros(row, column);
    img = double(img);
    for i = 2:row-1
      for j = 2:column-1
        new_img(i, j) = median([img(i-1, j-1), img(i-1, j), img(i-1, j+1), img(i, j-1),
img(i, j), img(i, j+1), img(i+1, j-1), img(i+1, j), img(i+1, j+1)]);
      end
    end
    new_img = uint8(new_img);

end
```

```
% average_filter

function new_img = average_filter(img, scale)

    [row, column] = size(img);
    w = ones(scale, scale) / (scale * scale);
    [filter_row, filter_column] = size(w);
    img = double(img);

    if length(size(img)) ~= 2
        disp("Error: not a grayscale image")
        return
    end

    row_padding = (filter_row - 1) / 2;
    column_padding = (filter_column - 1) / 2;

    temp = zeros(row + 2 * row_padding, column + 2 * column_padding);
    result = zeros(row + 2 * row_padding, column + 2 * column_padding);
```

```
    temp(row_padding + 1 : row + row_padding, column_padding + 1 : column +
column_padding) = img;

    for i = row_padding + 1 : row + row_padding
      for j = column_padding + 1 : column + column_padding
        result(i, j) = abs(sum(sum(temp(i - row_padding : i + row_padding, j -
column_padding : j + column_padding) .* w)));
      end
    end

    new_img = uint8(result(row_padding + 1 : row + row_padding, column_padding + 1 :
column + column_padding));

end
```

Q4




```
% main.m

tiledlayout(1, 2);

img = imread('operahouse_med.tif');
nexttile;
imshow(img);
title('Original');

res = laplacian_detector(img);

th = 10;
res(res>th) = 255;
res(res<th) = 0;
```

```
nexttile;
imshow(res);
title('Laplacian Edge');
imwrite(res, 'operahouse_edge.tif')
```

```
% my_filter.m

function new_img = my_filter(img, w)

    [row, column] = size(img);
    [filter_row, filter_column] = size(w);
    img = double(img);

    if length(size(img)) ~= 2
        disp("Error: not a grayscale image")
        return
    end

    row_padding = (filter_row - 1) / 2;
    column_padding = (filter_column - 1) / 2;

    temp = zeros(row + 2 * row_padding, column + 2 * column_padding);
    result = zeros(row + 2 * row_padding, column + 2 * column_padding);
    temp(row_padding + 1 : row + row_padding, column_padding + 1 : column +
column_padding) = img;

    for i = row_padding + 1 : row + row_padding
      for j = column_padding + 1 : column + column_padding
        result(i, j) = abs(sum(sum(temp(i - row_padding : i + row_padding, j -
column_padding : j + column_padding) .* w)));
      end
    end

    new_img = uint8(result(row_padding + 1 : row + row_padding, column_padding + 1 :
column + column_padding));

end
```

```
% laplacian_detector.m

function newimg = laplacian_detector(img)
    w = [0 1 0; 1 -4 1; 0 1 0];
    newimg = my_filter(img, w);
end
```