ASSIGNMENT-05

Course Title: Image Processing
Course ID: CSC 420
Submitted By: Abu Bakar Siddik Nayem.
ID: 1510190.
Section: 01.

Date of Submission: 24th November, 2018.
Submitted To: Md. Ashraful Amin, PhD.

1. Motion detection using image subtraction from a video.

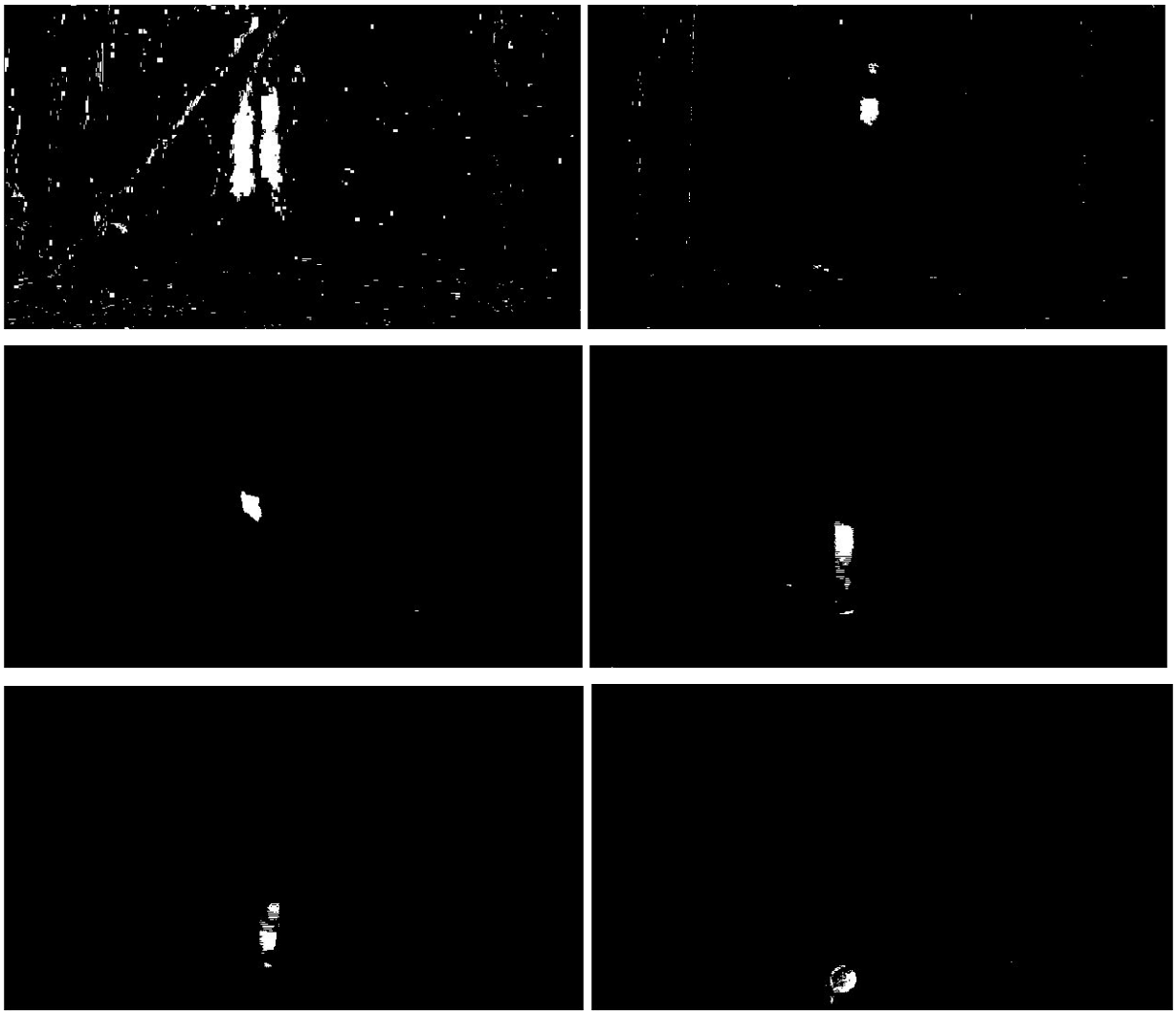
Ans:

Input: Input is a .mp4 video file of bouncing ball.
Dimensions: 1920 x 1080
FPS: 30
Duration: 09.9850 seconds.

Some frames from the input mp4 files are:

Output:
Output is the binarized frames with motion detected by subtracting the background from the moving object. Only the moving object is white(0) and the background is black(1)
Output format:
i. PNG image files for each frame. (9.9850x30 = 299 frames in total)
Ii. Avi video file. (1 .avi file made from adding the frames in a sequence)

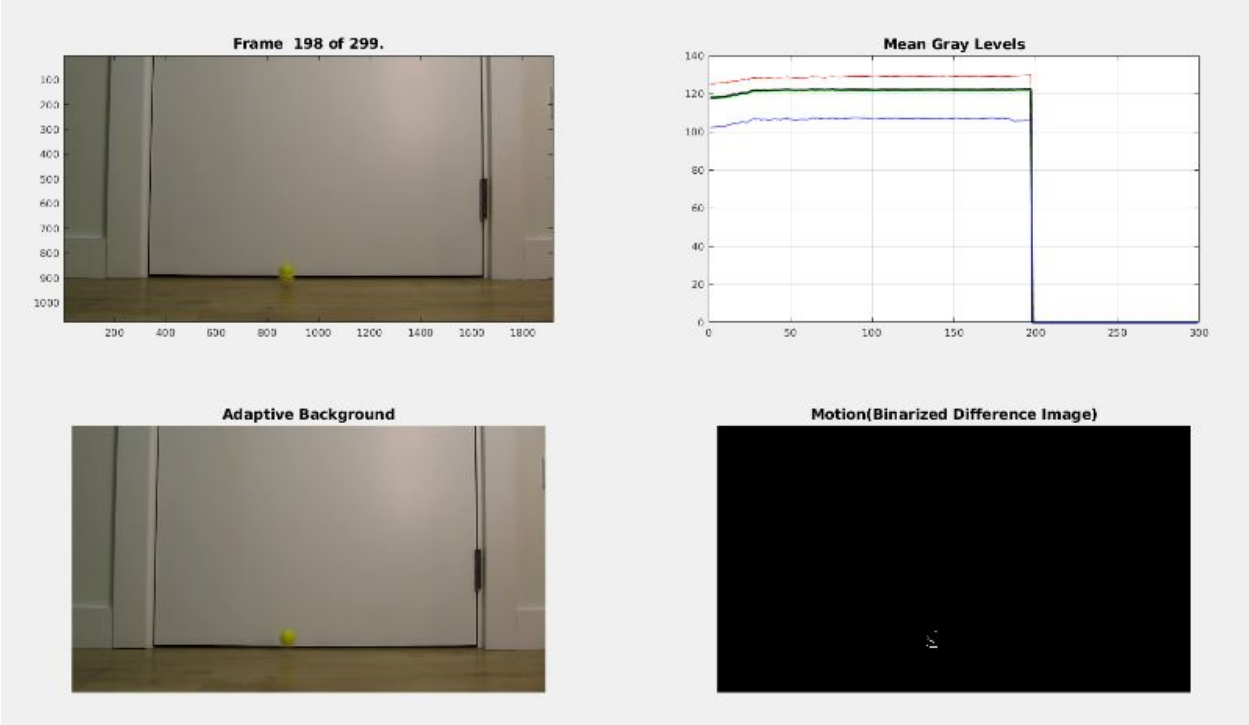
Some output frames are :



The output video file and frames(images) with detected motion is attached in the assignment folder.

The program(.m file) also shows rgb values in a graph at each individual frame.

GUI of the program:



2. Apply median, mode and laplacian filter. Discuss your observation.

Ans:

Median filter:

Input image: **Output Image**

1.


2.


Input image: **Output Image**

3.


4.

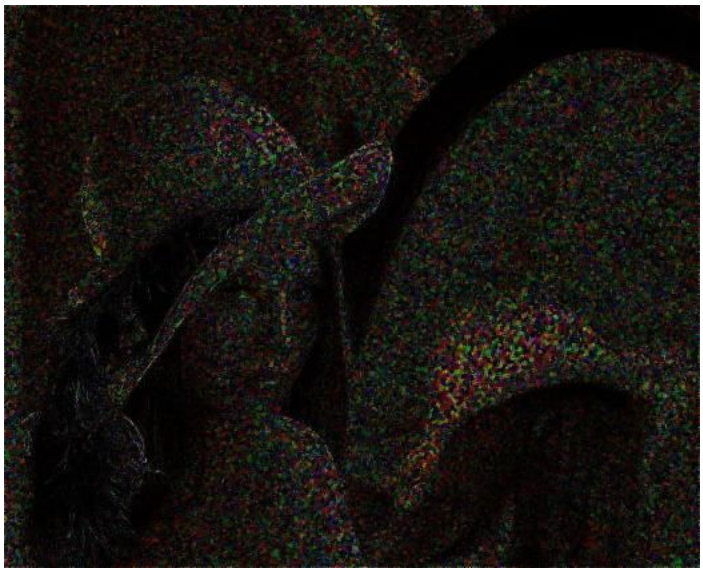

Observation:
Median filter works really well in images with saltpepper noise. Most of the noise is gone and the quality of the image is well preserved.

But for images with speckle noise median filter can not remove most of the noise and also the image becomes blurry.

For 'Lena.png' image with saltpepper noise the difference with the median filtered image and the clear lena image is:

The above shows that very small amount from the original(clear) image is lost.

For ’Lena.png’ image with speckle noise the difference with the median filtered image and the clear lena image is:



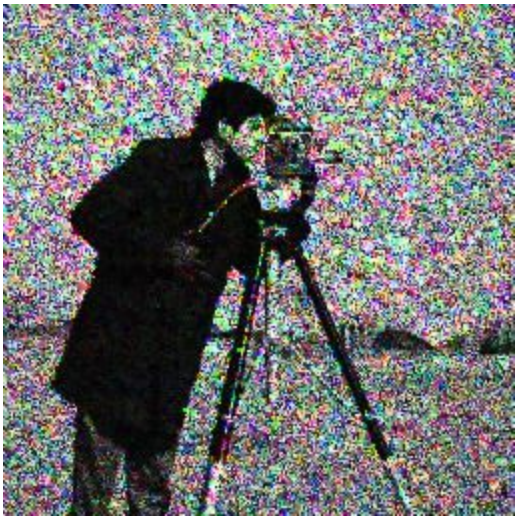
The above shows that much more information is lost while denoising the speckle noised image using the median filter.

Mode filter:

Input image: **Output Image**

1.

2.



Input image: **Output Image**

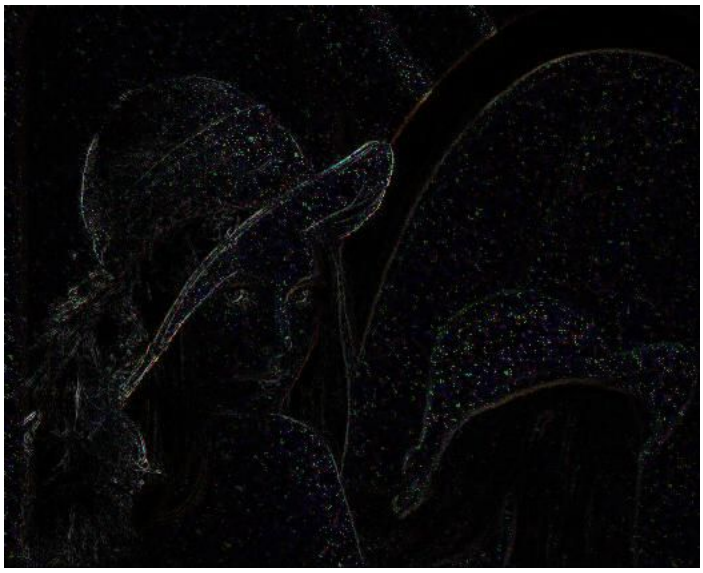
3.


4.


Observation:
Mode filter does not work as well as median filter in images with saltpepper noise. Some of the noise is gone and the quality of the image is well preserved.

But for images with speckle noise, mode filter can not remove most of the noise and also the image becomes blurry.

For ’Lena.png’ image with saltpepper noise the difference with the mode filtered image and the clear lena image is:



The above shows that very small amount from the original(clear) image is lost.

For ’Lena.png’ image with speckle noise the difference with the mode filtered image and the clear lena image is:

The above shows that much more information is lost while denoising the speckle noised image using the mode filter.

For ’c.png’ image with saltpepper noise the difference with the mode filtered image and the clear lena image is:

The above shows that very small amount from the original(clear) image is lost.

For ’c.png’ image with speckle noise the difference with the mode filtered image and the clear lena image is:

The above shows that much more information is lost while denoising the speckle noised image using the mode filter.

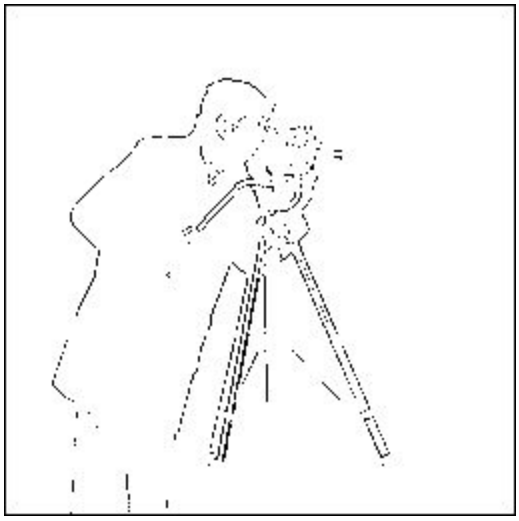
[Note: The ‘c.png’ image was single channel(grayscale) and the noised images were RGB, so for calculating the difference the filtered rgb images were turned to grayscale]

Laplacian Filter:

Laplacian filter in image processing is used to find area of rapid change(edges) in images.

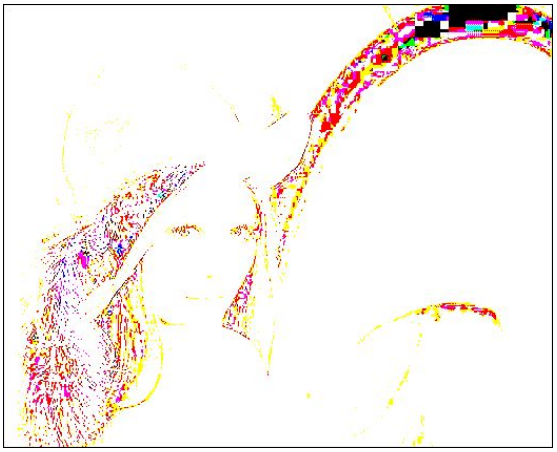
Input image: **Output Image**

1.

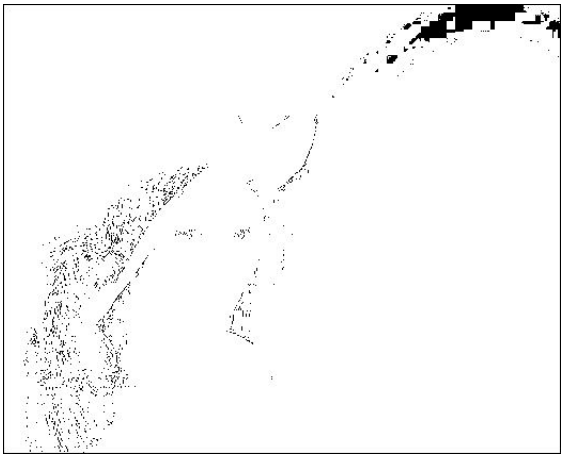


2.

2.



3.



Observation:

Laplacian filters work great detecting the edges in an image.
The laplacian filter works equally well for rgb(sample-2) and greyscale(sample-3) images.
The edges in the image pops out and the rest of the image is gone.

Detected Edges in the image by overlaying the laplacian filtered image:

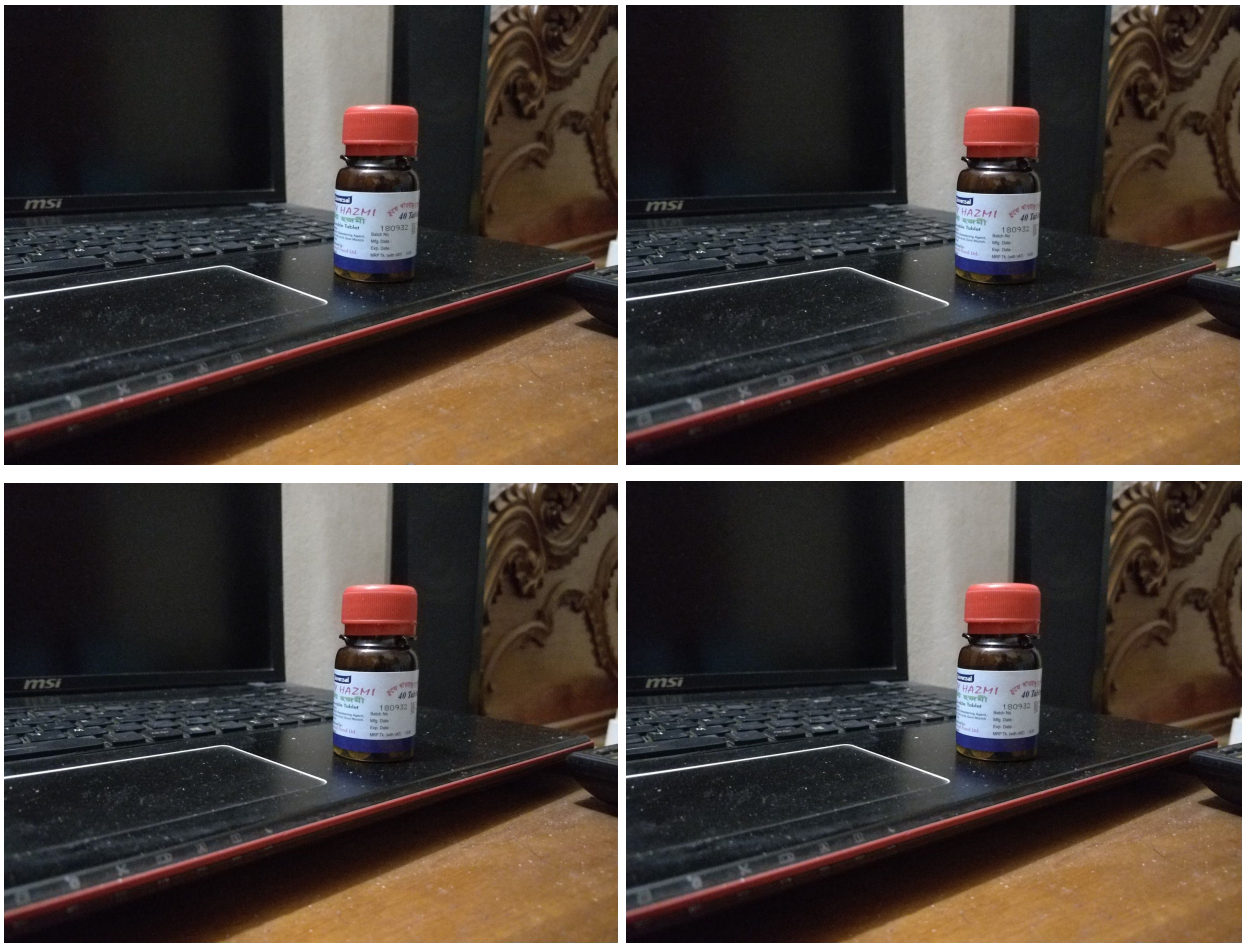
3. Capture image in burst and normal mode. Compare between them. Apply image averaging technique. Discuss the findings.

Ans:

This experiment is done in 2 stages.

Stage 1:

18 images were captured in burst mode using a stand/tripod to stabilize the image sensor. Some samples:



16 images were captured in burst mode without any stand/tripod. SO all the images do not necessarily represent the same frame.

Some samples:

Since human hands tend to shake a little even while trying to be perfectly still, the images are slightly different in terms of capturing the area. This will affect the averaging technique.

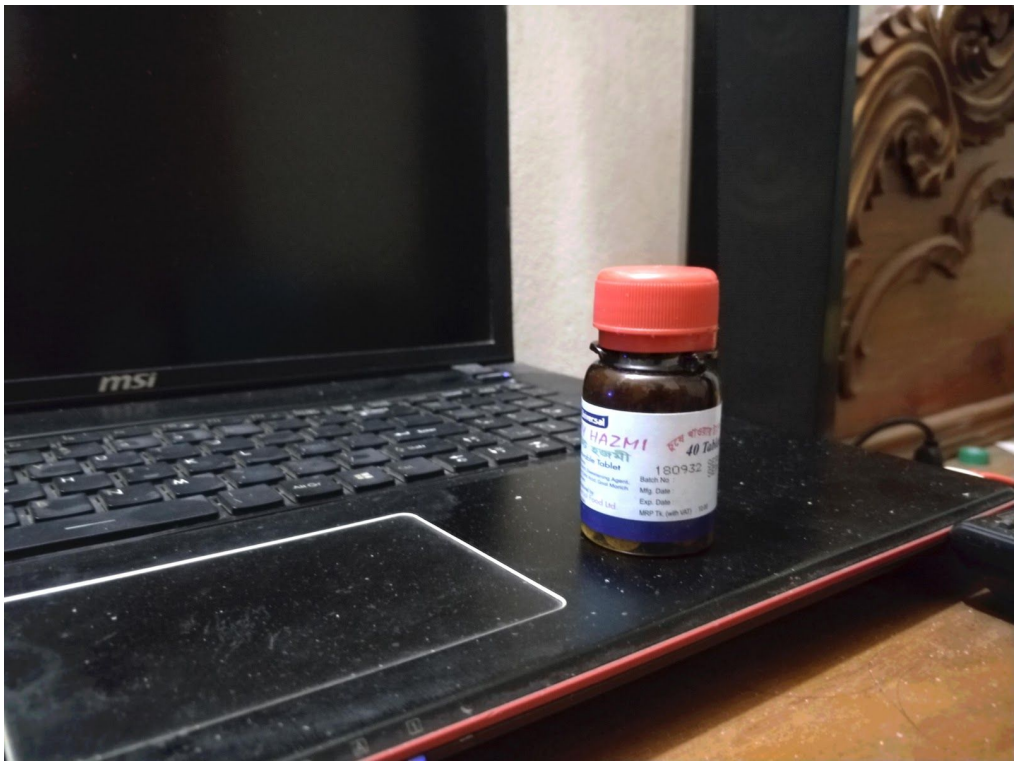
Image taken in normal mode (without burst):

After applying averaging technique in the images taken (with tripod) :
Average of first 5 images:

Average of all 18 images:

After applying averaging technique in the images taken (without tripod) :

Average of all 16 images:



Findings:

1. Averaging an image takes the noise out as it should(theoretically).
2. The brightness slightly increases.
3. Some sharpness is lost due to the ever so slight movement even with the tripod.
4. Without a tripod/stabilizer(OIS/EIS) the images taken gives more un-sharp(slightly blurry) image.
5. The colors pop more in the averaged images than the normal image or the burst images.
6. Averaging is good to denoise an image but using too many images to apply the averaging does not hold a better result since the chance of movement with the subject or the camera itself is bigger. So in that case the averaged image is even worse than the normal images or average of only some(before any movement).

Appendix:

1) motion_detection.m:

```
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
fontSize = 14;

% Change the current folder to the folder of this m-file.
% (The line of code below is from Brett Shoelson of The Mathworks.)
if(~isdeployed)
	cd(fileparts(which(mfilename)));
end

% Open the rhino.avi demo movie that ships with MATLAB.
folder = fullfile('\');
movieFullFileName = fullfile('Bouncing.mp4');
% Check to see that it exists.
if ~exist(movieFullFileName, 'file')
	strErrorMessage = sprintf('File not found:\n%s\nYou can choose a new one, or cancel',
movieFullFileName);
	response = questdlg(strErrorMessage, 'File not found', 'OK - choose a new movie.',
'Cancel', 'OK - choose a new movie.');
	if strcmpi(response, 'OK - choose a new movie.')
		[baseFileName, folderName, FilterIndex] = uigetfile('*.avi');
		if ~isequal(baseFileName, 0)
			movieFullFileName = fullfile(folderName, baseFileName);
		else
			return;
		end
	else
		return;
	end
end

try
	videoObject = VideoReader(movieFullFileName)
	% Determine how many frames there are.
	numberOfFrames = videoObject.NumberOfFrames;
	vidHeight = videoObject.Height;
```

```
	vidWidth = videoObject.Width;

	numberOfFramesWritten = 0;
	% Prepare a figure to show the images in the upper half of the screen.
	figure;
	%	screenSize = get(0, 'ScreenSize');
	% Enlarge figure to full screen.
	set(gcf, 'units','normalized','outerposition',[0 0 1 1]);

	% Ask user if they want to write the individual frames out to disk.
	promptMessage = sprintf('Do you want to save the individual frames out to individual
disk files?');
	button = questdlg(promptMessage, 'Save individual frames?', 'Yes', 'No', 'Yes');
	if strcmp(button, 'Yes')
		writeToDisk = true;

		% Extract out the various parts of the filename.
		[folder, baseFileName, extentions] = fileparts(movieFullFileName);
		% Make up a special new output subfolder for all the separate
		% movie frames that we're going to extract and save to disk.
		% (Don't worry - windows can handle forward slashes in the folder name.)
		folder = pwd;   % Make it a subfolder of the folder where this m-file lives.
		outputFolder = sprintf('%s/Movie Frames from %s', folder, baseFileName);
		% Create the folder if it doesn't exist already.
		if ~exist(outputFolder, 'dir')
			mkdir(outputFolder);
		end
	else
		writeToDisk = false;
	end

	% Loop through the movie, writing all frames out.
	% Each frame will be in a separate file with unique name.
	meanGrayLevels = zeros(numberOfFrames, 1);
	meanRedLevels = zeros(numberOfFrames, 1);
	meanGreenLevels = zeros(numberOfFrames, 1);
	meanBlueLevels = zeros(numberOfFrames, 1);
	for frame = 1 : numberOfFrames
		% Extract the frame from the movie structure.
		thisFrame = read(videoObject, frame);

		% Display it
		hImage = subplot(2, 2, 1);
```

```
		image(thisFrame);
		caption = sprintf('Frame %4d of %d.', frame, numberOfFrames);
		title(caption, 'FontSize', fontSize);
		drawnow; % Force it to refresh the window.



		% Calculate the mean gray level.
		grayImage = rgb2gray(thisFrame);
		meanGrayLevels(frame) = mean(grayImage(:));

		% Calculate the mean R, G, and B levels.
		meanRedLevels(frame) = mean(mean(thisFrame(:, :, 1)));
		meanGreenLevels(frame) = mean(mean(thisFrame(:, :, 2)));
		meanBlueLevels(frame) = mean(mean(thisFrame(:, :, 3)));

		% Plot the mean gray levels.
		hPlot = subplot(2, 2, 2);
		hold off;
		plot(meanGrayLevels, 'k-', 'LineWidth', 2);
		hold on;
		plot(meanRedLevels, 'r-');
		plot(meanGreenLevels, 'g-');
		plot(meanBlueLevels, 'b-');
		grid on;

		% Put title back because plot() erases the existing title.
		title('Mean Gray Levels', 'FontSize', fontSize);
		if frame == 1
			xlabel('Frame Number');
			ylabel('Gray Level');
			% Get size data later for preallocation if we read
			% the movie back in from disk.
			[rows, columns, numberOfColorChannels] = size(thisFrame);
		end

		% Update user with the progress.  Display in the command window.
		if writeToDisk
			progressIndication = sprintf('Wrote frame %4d of %d.', frame,
numberOfFrames);
		else
			progressIndication = sprintf('Processed frame %4d of %d.', frame,
numberOfFrames);
```

```
        end
        disp(progressIndication);
        % Increment frame count (should eventually = numberOfFrames
        % unless an error happens).
        numberOfFramesWritten = numberOfFramesWritten + 1;

        % Now let's do the differencing
        alpha = 0.5;
        if frame == 1
            Background = thisFrame;
        else
            % Change background slightly at each frame
            %                       Background(t+1)=(1-alpha)*I+alpha*Background
            Background = (1-alpha)* thisFrame + alpha * Background;
        end
        % Display the changing/adapting background.
        subplot(2, 2, 3);
        imshow(Background);
        title('Adaptive Background', 'FontSize', fontSize);
        % Calculate a difference between this frame and the background.
        differenceImage = thisFrame - uint8(Background);
        % Threshold with Otsu method.
        grayImage = rgb2gray(differenceImage); % Convert to gray level
        thresholdLevel = graythresh(grayImage); % Get threshold.
        binaryImage = im2bw( grayImage, thresholdLevel); % Do the binarization
        % Plot the binary image.
        subplot(2, 2, 4);
        imshow(binaryImage);
        title('Motion(Binarized Difference Image)', 'FontSize', fontSize);

if writeToDisk
   outputBaseFileName = sprintf('Frame %4.4d.png', frame);
   outputFullFileName = fullfile(outputFolder, outputBaseFileName);


   % Extract the image with the text "burned into" it.
   frameWithText = getframe(gca);
   % frameWithText.cdata is the image
   % Write it out to disk.
   imwrite(frameWithText.cdata, outputFullFileName, 'png');
end

   end
```

```
	% Alert user that we're done.
	if writeToDisk
		finishedMessage = sprintf('Done!  It wrote %d frames to folder\n"%s"',
numberOfFramesWritten, outputFolder);
	else
		finishedMessage = sprintf('Done!  It processed %d frames of\n"%s"',
numberOfFramesWritten, movieFullFileName);
	end
	disp(finishedMessage); % Write to command window.
	uiwait(msgbox(finishedMessage)); % Also pop up a message box.

	% Exit if they didn't write any individual frames out to disk.
	if ~writeToDisk
		return;
	end

	% Ask user if they want to read the individual frames from the disk,
	% that they just wrote out, back into a movie and display it.
	promptMessage = sprintf('Do you want to recall the individual frames\nback from disk
into a movie?\n(This will take several seconds.)');
	button = questdlg(promptMessage, 'Recall Movie?', 'Yes', 'No', 'Yes');
	if strcmp(button, 'No')
		return;
	end

	% Create a VideoWriter object to write the video out to a new, different file.
	writerObj = VideoWriter('New.avi');
	open(writerObj);

	% Read the frames back in from disk, and convert them to a movie.
	% Preallocate recalledMovie, which will be an array of structures.
	% First get a cell array with all the frames.
	allTheFrames = cell(numberOfFrames,1);
	allTheFrames(:) = {zeros(vidHeight, vidWidth, 3, 'uint8')};
	% Next get a cell array with all the colormaps.
	allTheColorMaps = cell(numberOfFrames,1);
	allTheColorMaps(:) = {zeros(256, 3)};
	% Now combine these to make the array of structures.
	recalledMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps)
	for frame = 1 : numberOfFrames
		% Construct an output image file name.
		outputBaseFileName = sprintf('Frame %4.4d.png', frame);
```

```
            outputFullFileName = fullfile(outputFolder, outputBaseFileName);
            % Read the image in from disk.
            thisFrame = imread(outputFullFileName);
            % Convert the image into a "movie frame" structure.
            recalledMovie(frame) = im2frame(thisFrame);
            % Write this frame out to a new video file.
            writeVideo(writerObj, thisFrame);
        end
        close(writerObj);
        % Get rid of old image and plot.
        delete(hImage);
        delete(hPlot);
        % Create new axes for our movie.
        subplot(1, 3, 2);
        axis off;  % Turn off axes numbers.
        title('Movie recalled from disk', 'FontSize', fontSize);
        % Play the movie in the axes.
        movie(recalledMovie);
        % Note: if you want to display graphics or text in the overlay
        % as the movie plays back then you need to do it like I did at first
        % (at the top of this file where you extract and imshow a frame at a time.)
        msgbox('Done with this demo!');

catch ME
        % Some error happened if you get here.
        strErrorMessage = sprintf('Error extracting movie frames from:\n\n%s\n\nError: %s\n\n)',
movieFullFileName, ME.message);
        uiwait(msgbox(strErrorMessage));
end
```

2)

Median_filter.m:

```
function gr = median_filter(image)

    gr = image;

    Lint = 1;
    Pint = 1;

    % Lines
    for l = Lint+1 : size(image,1)-Lint
  % Pixels
  for p = Pint+1 : size(image,2)-Pint
    % Extract of sub-image (window)
    window = image(l-Lint : l+Lint, p-Pint : p+Pint);
    [n1,n2] = size(window);
    vector = zeros(n1*n2);
    i = 1;
    for j = 1 : n1
      for k = 1 : n2
        vector(i) = window(j,k);
        i = i + 1;
      end
    end
    sorted = sort(vector);
    % convolution between sub-image and mask
    gr(l,p) = sorted(fix(length(sorted)/2) + 1);
  end
 end
end
```

Median_filter_run.m:

```
lenaNoise = imread('lena_speckle.jpg');
% lenaSmooth = median_filter(lenaNoise);
% imwrite(lenaSmooth, 'results/LenaSmooth_Median_Filter.jpg');
channel_r = lenaNoise(:,:,1);
channel_g = lenaNoise(:,:,2);
channel_b = lenaNoise(:,:,3);
lenaSmooth(:,:,1) = median_filter(channel_r);
lenaSmooth(:,:,2) = median_filter(channel_g);
```

```
lenaSmooth(:,:,3) = median_filter(channel_b);
imwrite(lenaSmooth, 'lena_speckle_Median_Filter.jpg');
```

Mode_filter.m:

```
function gr = mode_filter(image)

        gr = image;

        Lint = 1;
        Pint = 1;

        % Lines
   for l = Lint+1 : size(image,1)-Lint
      % Pixels
      for p = Pint+1 : size(image,2)-Pint
         % Extract of sub-image (window)
         window = image(l-Lint : l+Lint, p-Pint : p+Pint);
         [n1,n2] = size(window);
         vector = zeros(n1*n2);
         i = 1;
         for j = 1 : n1
            for k = 1 : n2
               vector(i) = window(j,k);
               i = i + 1;
            end
         end
         vectorLength = size(vector, 1);
         modeValue = -1;
         modeCount = 0;
         for i = 1 : vectorLength
            count = 0;
            for j = 1: vectorLength
               if (vector(j) == vector(i))
                  count = count+1;
               end
            end
            if (count > modeCount)
               modeCount = count;
               modeValue = vector(i);
            end
         end
         if(modeValue > -1)
```

```
            gr(l,p) = modeValue;
          end
        end
    end
end
```

Mode_filter_run.m:

```
lenaNoise = imread('c_speckle.jpg');
channel_r = lenaNoise(:,:,1);
channel_g = lenaNoise(:,:,2);
channel_b = lenaNoise(:,:,3);
lenaSmooth(:,:,1) = mode_filter(channel_r);
lenaSmooth(:,:,2) = mode_filter(channel_g);
lenaSmooth(:,:,3) = mode_filter(channel_b);
imwrite(lenaSmooth, 'c_speckle_Mode_Filter.jpg');
```

Laplacian_filter:

```
clc
clear all
close all
image=imread('lena_gray.png');
filter=[0 -1 0 ; -1 9 -1 ; 0 -1 0];
rows = size(image,1);
cols = size(image,2);
outputimage = zeros(rows,cols);

for row = 2:rows-1
    for col = 2:cols-1
        outputimage (row,col)= sum (sum(double(image(row-1: row+1, col-1: col+1 )) .* filter ));
    end
end
image =uint8(image);
figure,imshow (outputimage);
imwrite(outputimage, 'lena_gray_Laplacian.png')
```

3)

Average.m:

```
im1 = imread('IMG_20181125_234224_001.jpg');
im2 = imread('IMG_20181125_234224_002.jpg');
im3 = imread('IMG_20181125_234224_003.jpg');
im4 = imread('IMG_20181125_234224_004.jpg');
im5 = imread('IMG_20181125_234224_005.jpg');
im6 = imread('IMG_20181125_234224_006.jpg');
im7 = imread('IMG_20181125_234224_007.jpg');
im8 = imread('IMG_20181125_234224_008.jpg');
im9 = imread('IMG_20181125_234224_009.jpg');
im10 = imread('IMG_20181125_234224_010.jpg');
im11 = imread('IMG_20181125_234224_011.jpg');
im12 = imread('IMG_20181125_234224_012.jpg');
im13 = imread('IMG_20181125_234224_013.jpg');
im14 = imread('IMG_20181125_234224_014.jpg');
im15 = imread('IMG_20181125_234224_015.jpg');
im16 = imread('IMG_20181125_234224_016.jpg');
im17 = imread('IMG_20181125_234224_017.jpg');
im18 = imread('IMG_20181125_234224_018.jpg');

avg = uint8(zeros(size(im1)));
[rows,columns,channels] = size(avg);

for k = 1:channels
   for i = 1:rows
      for j = 1: columns
         avg(i,j,k) =
(im1(i,j,k)/18+im2(i,j,k)/18+im3(i,j,k)/18+im4(i,j,k)/18+im5(i,j,k)/18+im6(i,j,k)/18+im7(i,j,k)/18+im8(
i,j,k)/18+im9(i,j,k)/18+im10(i,j,k)/18+im11(i,j,k)/18+im12(i,j,k)/18+im13(i,j,k)/18+im14(i,j,k)/18+im
11(i,j,k)/18+im12(i,j,k)/18+im13(i,j,k)/18+im14(i,j,k)/18+im15(i,j,k)/18+im16(i,j,k)/18+im17(i,j,k)/18
+im18(i,j,k)/18);
      end
   end

end
imwrite(avg,'avg18.jpg');
```