2. Load the original video, produce a complement image for each frame (you may use imcomplement) and save the new video file.
   The saved filename must be Vid2.avi. The function name is Q2B_GROUP_XX_YY
3. Load the original video, and superimpose the edge map (displayed in perfect red color) over the original frame for all the frames, and save them to a new video file. This will require several temporary variables –choose them as you wish.
   For computing the edge map use the Sobel operator (you can find it in Wikipedia) and convolve it with the gray frame using conv2. Do not use the function edge!
   The saved filename must be Vid3.avi. The function name is Q2C_GROUP_XX_YY
4. Repeat question 3, but this time compute the edge map using edge(I,'sobel') where I is the current gray frame. **The saved filename must be Vid4.avi**
   The function name is Q2D_GROUP_XX_YY

Part 3 – Harris Corner Detection (50 points)

In this part you will implement the Harris corner detection and run it on 2 images.

You'll create 2 functions: myHarrisCornerDetector.m and createCornerPlots.m

myHarrisCornerDetector accepts an image input (color or B&W), the value K and the Threshold, finds all the corners using a 5x5 neighborhood for each pixels and returns as its output a binary matrix, the same *dimensions* as the dimensions of the original image where pixels that are corners have the value '1' and all other pixels are of value '0'.

createCornerPlots accepts the original image and corner detection results and creates a plot showing the original image with red circles on the corners.

This plot *has* to contain 2 subplots: the left one shows image I1 with red circles on the corners and the right side does the same for image I2.

For I1 (checkerboard) I suggest you use these parameters in the plot command **'ro', 'linewidth', 5.** For I2 (Giraffe) I suggest you use: **'ro', 'linewidth', 2**

CLARIFICATION (Calling both functions):

OUT = **myHarrisCornerDetector**(IN,K,Threshold)

IN = input image

OUT = output binary matrix

K = K value in the formula (see instruction 2 for this question)

Threshold = Threshold value in the formula (see instruction 2 for this question)

createCornerPlots(I1 , I1_CORNERS , I2 , I2_CORNERS

Instructions:

1. You may ignore pixels on the very edge of the images (the 'frame' pixels).
2. Use the formula you saw in class with a 5x5 neighborhood:
   - Compute all derivatives in x and in y in the image.
   - Compute the M matrix (defined in class) for every neighborhood.
   - For each pixel check if:

     $$\det(M) - k * \left(trace(M)\right)^2 \geq threshold$$

     If it is – this can be a corner.

     *** K and threshold are set for you in the HW1_Q3.m function (included to the assignment).

     This is similar to the equation you saw in class. It's the equation that appears in the original article.

3. You may NOT use the function corner.m or ready-made functions off the internet.
4. Use rgb2gray for transforming the color image to grayscale before computing the corners, but at the end you have to show the circles superimposed on the *original* images.
5. For finding the derivatives in x and y you should create a shifted copy of the image file (after converting to grayscale) and simply subtract in x and in y). Such that:

   $$I_x = I - I\_shifted\_right\,, \quad I_y = I - I\_shifted\_down$$