CSC 481

Assignment 2 (35/25 points)

Problem 1: Images Scaling by Pixel Replication (15/15)

Write your own function capable of shrinking and zooming an image by pixel replication and decimation. Assume that the desired zoom/shrink factors will be the inputs for your function and will have integer values: a negative input means shrink and a positive input means expand. Do not use MatLab built in functions (or other language libraries) for decimation and replication, although you can use other MatLab functions. Specifically, you must deal with pixels in your code.

(a) Use your program to shrink an image by a factor of 4 in each dimension. Show the shrunk image.

(b) Use your program to zoom the image back to its original size. Show the zoomed image and explain how and why the original image and the shrunk/zoomed images are different.

CODE:

```
inputImage = imread('b2.jpg');

% Shrunk Image function call
shrunk_Image = ZI(inputImage, -4);

figure;
subplot(1,3,1);
imshow(inputImage);
title(['Original Image ' num2str(size(inputImage, 1)) 'x' num2str(size(inputImage, 2))]);

subplot(1, 3, 2);
imshow(shrunk_Image);
title(['Shrunk Image ' num2str(size(shrunk_Image, 1)) 'x' num2str(size(shrunk_Image, 2))]);

% Zoom back > shrunk image function call
zoomedBackImage = ZI(shrunk_Image, 4);

subplot(1, 3, 3);
imshow(zoomedBackImage);
title(['Zoomed Back Image ' num2str(size(zoomedBackImage, 1)) 'x' num2str(size(zoomedBackImage, 2))]);


% Child function for shrinking image
function shrunkImage = shrinkImage(inputImage, factor)
   shrunkImage = inputImage(1:factor:end, 1:factor:end, :);
end

% Child function to zoom image
function zoomedImage = zoomImage(inputImage, factor)
   [height, width, ~] = size(inputImage);
```

```
    newHeight = height * factor;
    newWidth = width * factor;
    zoomedImage = zeros(newHeight, newWidth, size(inputImage, 3));
    zoomedImage(1:factor:end, 1:factor:end, :) = inputImage;
end

function ZI = ZI(inputImage, zoomFactor)
    if zoomFactor < 0
        ZI = shrinkImage(inputImage, abs(zoomFactor));
    elseif zoomFactor > 0
        ZI = zoomImage(inputImage, zoomFactor);
    else
        ZI = inputImage;
    end
end
```

O/P:

Problem 2: Basic Grey Level Transformations (5/5)

(a) Read and display an image.

(b) Calculate the negative of the image and display it.

(c) Perform contrast stretching. You can use the imadjust.m function (or similar function from your favorite language/library) to perform the image transformation. Be creative and try to do terrible things to your image. Display the result.

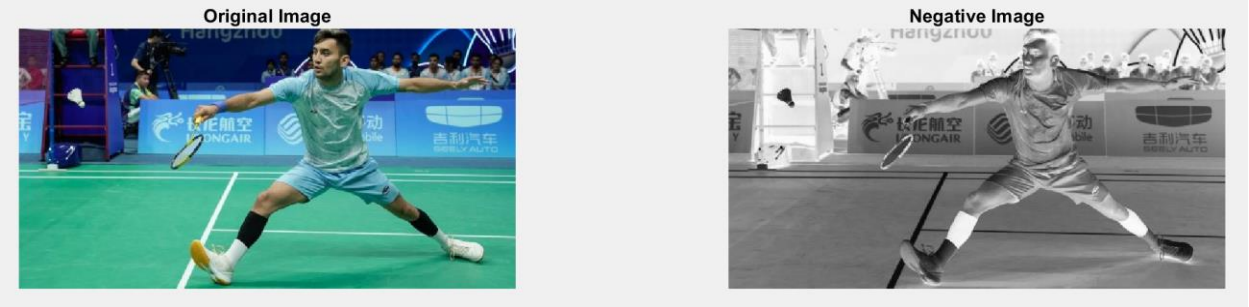
CODE:

```
% (a) Read and display an image
inputImage = imread('b2.jpg');
figure;
subplot(2, 2, 1);
imshow(inputImage);
title('Original Image');
input_GS = rgb2gray(inputImage);

% (b) Calculate the negative of the image and display it
negativeImage = 255 - input_GS;
subplot(2, 2, 2);
```

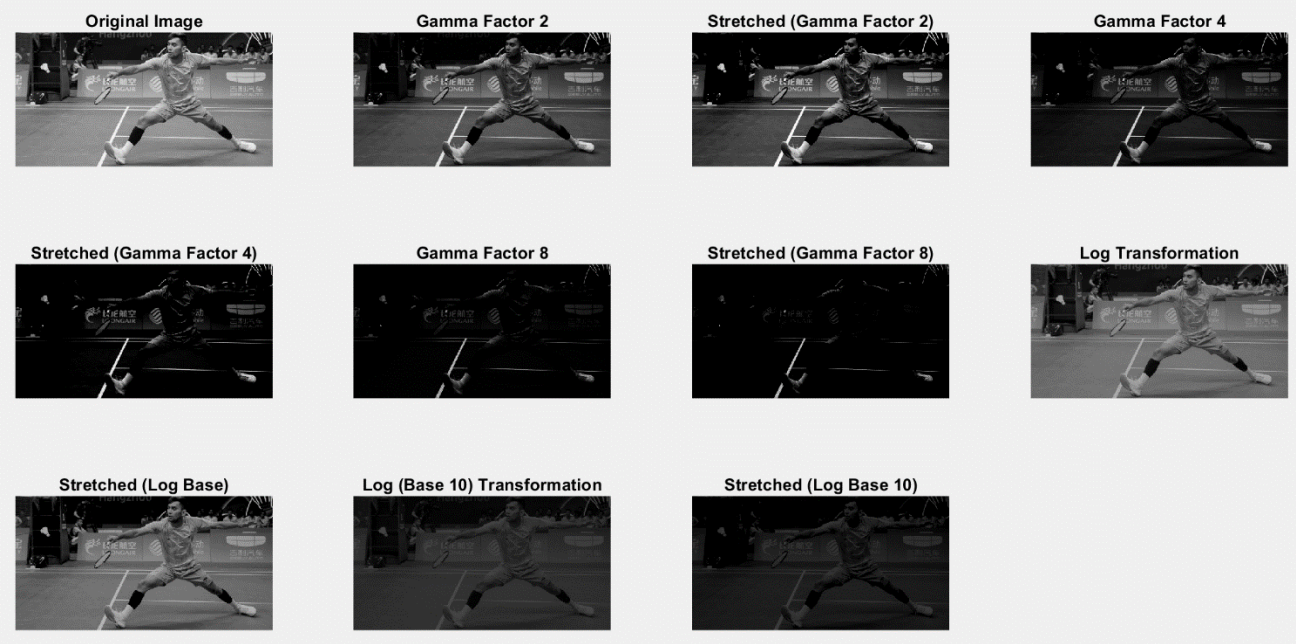
```
imshow(negativeImage);
title('Negative Image');

% (c) Perform contrast stretching
Image_double = im2double(input_GS);

Image_gamma2 = 1 * Image_double.^2;
Image_gamma4 = 1 * Image_double.^4;
Image_gamma8 = 1 * Image_double.^8;

logFactor = 1 * log(1 + Image_double);
log_10 = 1 * log10(1 + Image_double);

Image_gamma2_stretched = imadjust(Image_gamma2, [0.1, 0.9]);
Image_gamma4_stretched = imadjust(Image_gamma4, [0.1, 0.9]);
Image_gamma8_stretched = imadjust(Image_gamma8, [0.1, 0.9]);

logFactor_stretched = imadjust(logFactor, [0.1, 0.9]);
log_10_stretched = imadjust(log_10, [0.1, 0.9]);

figure;
subplot(3, 4, 1), imshow(Id), title('Original Image');

subplot(3, 4, 2), imshow(Image_gamma2), title('Gamma Factor 2');
subplot(3, 4, 3), imshow(Image_gamma2_stretched), title('Stretched (Gamma Factor 2)');

subplot(3, 4, 4), imshow(Image_gamma4), title('Gamma Factor 4');
subplot(3, 4, 5), imshow(Image_gamma4_stretched), title('Stretched (Gamma Factor 4)');

subplot(3, 4, 6), imshow(Image_gamma8), title('Gamma Factor 8');
subplot(3, 4, 7), imshow(Image_gamma8_stretched), title('Stretched (Gamma Factor 8)');

subplot(3, 4, 8), imshow(logFactor), title('Log Transformation');
subplot(3, 4, 9), imshow(logFactor_stretched), title('Stretched (Log Base)');

subplot(3, 4, 10), imshow(log_10), title('Log (Base 10) Transformation');
subplot(3, 4, 11), imshow(log_10_stretched), title('Stretched (Log Base 10)');
```

O/P:

A) and B)



C)



Problem 3: Image Transformations (10/0) (481 only)

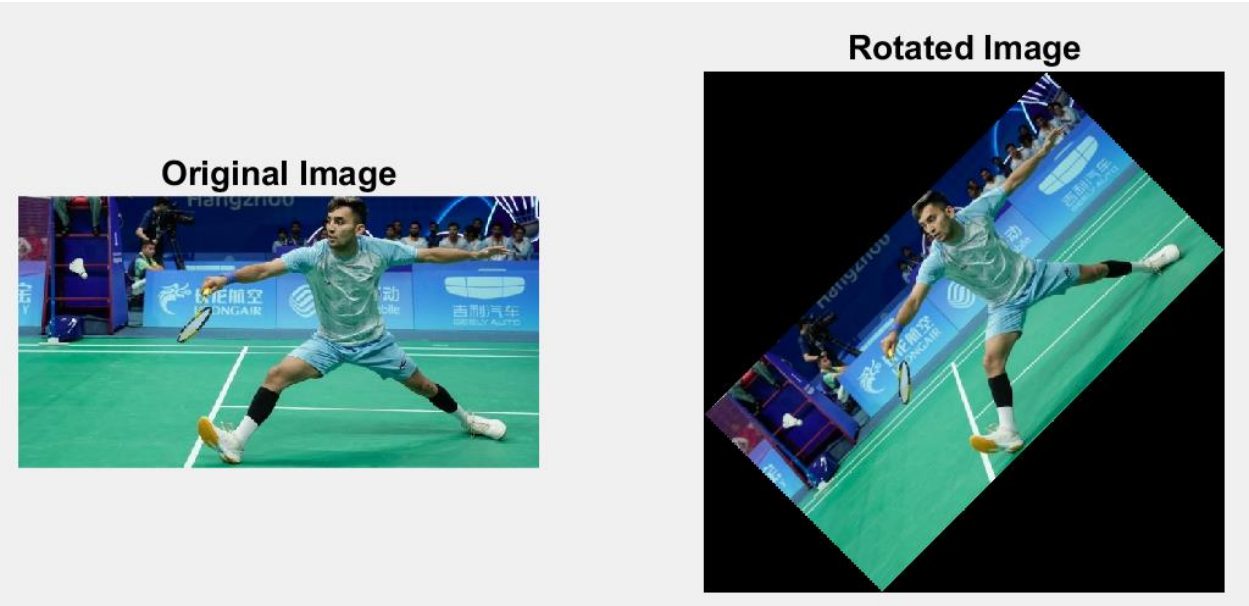
Image scaling is just one kind of transformation that we could use to modify an image. It belongs to a more general class of transformations called affine transformations. Do some research on affine transformations (Wikipedia is typically a good place to start) and write a function to rotate (only) an image by some number of degrees. (For rotation only, you do not need to use homogenous coordinates, although you can.) Run your code on an example image and show the result. Best if you use small angle values to prevent losing your image entirely.

CODE:

```
inputImage = imread(['b2.jpg']);
angleDegrees = 45;
```

```
% using built in function affine2d
t_form = affine2d([cosd(angleDegrees) -sind(angleDegrees) 0; sind(angleDegrees) cosd(angleDegrees) 0;
0 0 1]);
rotatedImage = imwarp(inputImage, t_form);

figure;
subplot(1, 2, 1);
imshow(inputImage);
title('Original Image');

subplot(1, 2, 2);
imshow(rotatedImage);
title('Rotated Image');
```

O/P:



Work by hand:

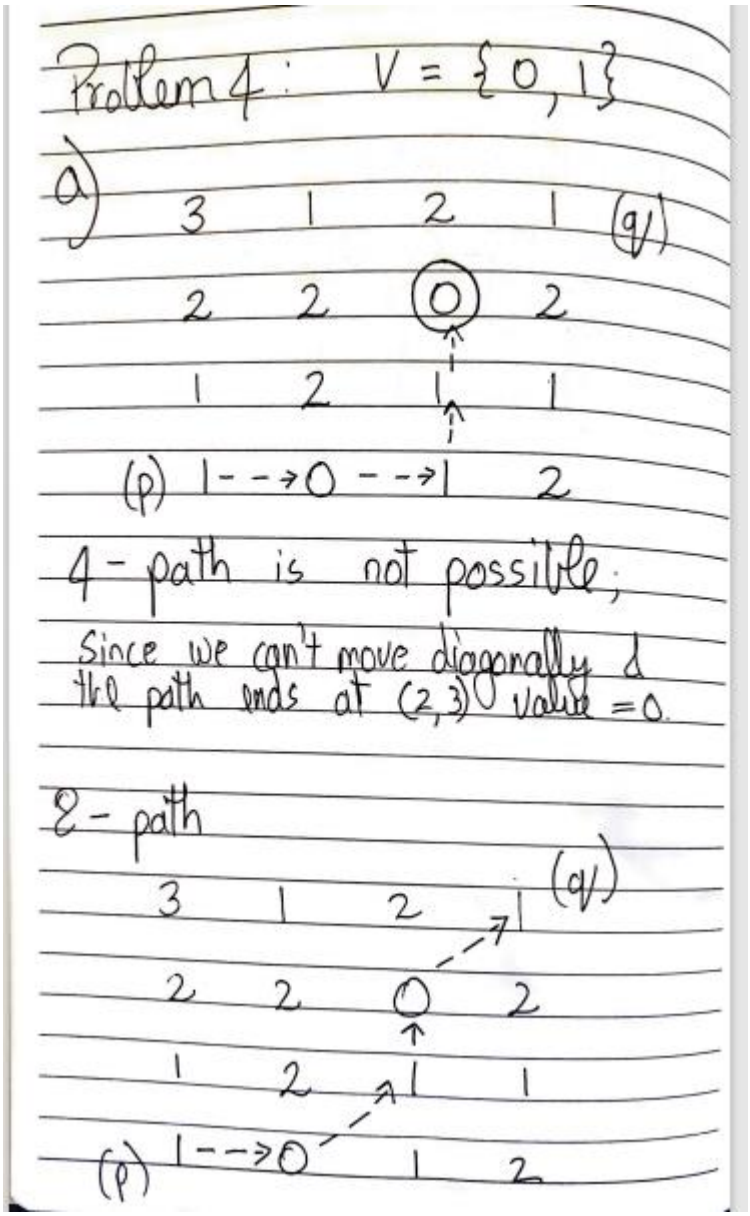
Problem 4: Some basic relationships between pixels (5 points) Note: You do not need to use code to solve this problem.

Consider the image segment shown below (the values in blue represent the p and q pixels):

	3	1	2	1	(q)
	2	2	0	2	
	1	2	1	1	
(p)	1	0	1	2	

(a) Let V= {0,1} and compute the lengths of the shortest 4-, 8- paths between p and q. If a particular path does not exist between these two points, explain why.

(b) Calculate the D_4 distance (city-block distance) and the D_8 distance (chessboard distance) between pixels p and q. Do these two distances depend on which path you choose between p and q? Explain your answer.



path length = 4 units

b) coordinates of p(x, y) = (0, 0)
coordinates of q(s, t) = (3, 3)

i) $D_4(p,q) = |x-s| + |y-t|$
$= |0-3| + |0-3|$
$= 3 + 3$
$= 6$ units

ii) $D_8(p,q) = \max(|x-s|, |y-t|)$
$= \max(|0-3|, |0-3|)$
$= \max(3, 3)$
$= 3$ units

The D_4 & D_8 paths are not influenced by the specific path chosen btw the points as long as the path consists of valid grid movements.

General submission instructions:

(a) Be kind to your aging, over-worked professor and submit only a single document. This can be pdf, MS Word, OpenOffice, etc. Do not submit a zip file.

(b) Your single document should include the input image for your problem, if required, and answers to each of the sub-problems (text, image or both, as appropriate). Your document should also include code that you wrote to generate your answers.

(c) You may use any images you like for the programming; I encourage you to use images that might be useful/interesting for your final project.

(d) Feel free to use whatever functions MatLab supplies, except where otherwise specified. Also feel free to write your own, if you are so inclined; it will take more time, but you will gain a deeper understanding of the material. It is one thing, for example, to use imadjust.m and quite another to write your own contrast stretching function.

(e) Point values for each question are indicated as ***x/y*** in which ***x*** is the point value for 481 students and ***y*** is the point value for 381 students.