160010031_160010011_160010058_assignmentFiltering

by Krishna Wadhwani

Submission date: 21-Aug-2018 02:11AM (UTC+0800)
Submission ID: 991584136
File name: Assign2-DIP.txt (6.41K)
Word count: 686
Character count: 5212

```
function output = myUnsharpMasking(input_path, sigma, scale)

load(input_path);

input = imageOrig;

blurred = imgaussfilt(input, sigma);

output = input + (input-blurred)*scale;

% Displaying the input and output image

myNumOfColors = 200;

myColorScale = [ [0:1/(myNumOfColors-1):1]' , [0:1/(myNumOfColors-1):1]' ,

[0:1/(myNumOfColors-1):1]' ];

figure('name', 'Unsharp Masking')

subplot(2,2,1)

imagesc(input);

colormap (myColorScale);
```

```
colormap gray;

daspect ([1 1 1]);

axis tight;

colorbar

title('Input Image')

subplot(2,2,2)

imagesc(output);

colormap (myColorScale);

colormap gray;

daspect ([1 1 1]);

axis tight;

colorbar

title('Output Image')
```

```
subplot(2,2,3)

imagesc(myLinearContrastStretching(input));

colormap (myColorScale);

colormap gray;

daspect ([1 1 1]);

axis tight;

colorbar

title('After linear contrast stretching')

subplot(2,2,4)

imagesc(myLinearContrastStretching(output));

colormap (myColorScale);

colormap gray;

daspect ([1 1 1]);
```

```
axis tight;

colorbar

title('After Linear Contrast Stretching')

end

function [newImage, rmsd] = myBilateralFiltering(imagePath, sigmaR, sigmaS,

windowSize)

load(imagePath);

if isequal(imagePath, '../data/barbara.mat')

    originalImage = imageOrig/100;

else

    originalImage = imgCorrupt;

end

[rows, cols] = size(originalImage);
```

```
size_ = rows;

%corrupting the image with noise

sd = 0.05*(max(max(originalImage)) - min(min(originalImage)));

noisyImage = eye(size_);

for i=1:rows

    for j=1:cols

        noisyImage(i,j) = originalImage(i,j) + sd*randn;

    end

end

%parameters for the bilateral filter

global sigmar; %standard deviation for the range-based gaussian

global sigmas; %standard deviation for the spatial gaussian

sigmar = sigmaR;
```

```
sigmas = sigmaS;

%kernel for each pixel is chosen to be of size 3*3

newImage = eye(size_); %initialising the new image

for i=1:rows

    for j=1:cols

        newImage(i,j) = bilateralFilter(windowSize, noisyImage, i, j);

    end

end

rmsd = (norm(newImage - originalImage, 'fro'))/256; %'fro' stands for frobenius norm

% Displaying the input and output image

myNumOfColors = 200;

myColorScale = [ [0:1/(myNumOfColors-1):1]' , [0:1/(myNumOfColors-1):1]' ,

[0:1/(myNumOfColors-1):1]' ];
```

```
figure('name', 'Bilateral Filtering')

subplot(1,3,1)

imagesc(originalImage);

colormap (myColorScale);

colormap gray

daspect ([1 1 1]);

axis tight;

colorbar

title('Original Image');

colorbar

subplot(1,3,2)

imagesc(noisyImage);

colormap (myColorScale);
```

```
%o2=get(gca,'Position');

colormap gray

% set(gca,'Position',o2)

daspect ([1 1 1]);

axis tight;

colorbar

title('Noisy Image');

colorbar

subplot(1,3,3)

imagesc(newImage);

%o2=get(gca,'Position');

colormap (myColorScale);

colormap gray
```

```
% set(gca,'Position',o2)

daspect ([1 1 1]);

axis tight;

colorbar

title(strcat('Filtered Image, ',  "RMSD = ", string(rmsd)));

colorbar

end

function pixelValue = bilateralFilter(windowSize, image, i, j)

    global sigmas;

    global sigmar;

    window = generateWindow(windowSize, image, i, j);

    [rowsWin, colsWin] = size(window);
```

```
    spatialGaussianWeights = fspecial('gaussian', size(window), sigmas); %approximation
for edge pixels
    intensityGaussianWeights = eye(size(window)); %initialising the intensity gaussian
mask
    for io=1:rowsWin
        for jo=1:colsWin
            intensityGaussianWeights(io,jo) = gaussianFunction(window(io,jo) - image(i,j),
sigmar);
        end
    end
    kernel = times(spatialGaussianWeights, intensityGaussianWeights); %element-wise
multiplication
    numerator = sum(sum(times(window, kernel)));
```

```
    denominator = sum(sum(kernel));

    pixelValue = numerator/denominator;

end

function window = generateWindow(windowSize, image,i,j)

    w= (windowSize - 1)/2; %w = 1 for a 3*3 window, w = 2 for a 5*5 window and so
on ....

    [rows, cols] = size(image);

    x1 = max(i-w, 1);

    x2 = min(i+w, rows);

    y1 = max(j-w, 1);

    y2 = min(j+w, cols);

    window = image(x1:x2, y1:y2);

end
```

```
function gaussx = gaussianFunction(x, standardDeviation)

    gaussx = (1/(standardDeviation*sqrt(2*pi)))*exp(-
x*x/(2*standardDeviation*standardDeviation));

end

function output = myPatchBasedFiltering(image, sigma)

%input = image;

%imshow(input);

sd = 0.05*(max(max(image)) - min(min(image)));

corrupted_image = image + sd*randn(size(image));

input = corrupted_image;

M = size(input,1);

N = size(input,2);

output = zeros(size(input));
```

```
%sigma = 10.5;

patch_w = 4;

size_w = 12;

mask = zeros(25,25);

for i = 1:M

    for j = 1:N

        x1 = max(i-size_w,1);

        x2 = min(i+size_w,M);

        y1 = max(j-size_w,1);

        y2 = min(j+size_w,N);

        px1 = max(i-patch_w, 1);

        px2 = min(i+patch_w, M);

        py1 = max(j-patch_w, 1);
```

```
py2 = min(j+patch_w, N);

patch_P = input(px1:px2, py1:py2);

%fprintf('Size = %i, %i ₩n', py1,py2);

window = input(x1:x2, y1:y2);

W_P = zeros(size(window));

w1 = size(window,1);

w2 = size(window,2);

for k = 1:w1

    for l=1:w2

        wx1 = max(k-patch_w, 1);

        wx2 = min(k+patch_w, w1);

        wy1 = max(l-patch_w, 1);

        wy2 = min(l+patch_w, w2);
```

```
        patch = window(wx1:wx2, wy1:wy2);

        Xi = min(size(patch,1), size(patch_P,1));

        Yi = min(size(patch,2), size(patch_P,2));

        patch_diff_matrix = patch(1:Xi, 1:Yi) - patch_P(1:Xi, 1:Yi);

        patch_diff_norm = sum( patch_diff_matrix(:) ) / (Xi*Yi);

        %fprintf('W-Y : %i, P-Y = %i,  ₩n', size(patch_W,2),size(patch_P,2));

        W_P(k,l) = patch_diff_norm;

    end

end

gaussian_W_P = exp( -W_P.^2/(sigma*sigma) );

weighted_avg = times(gaussian_W_P,window)/(sum(gaussian_W_P(:) ));

mask = weighted_avg;

output(i,j) = sum(weighted_avg(:));
```

```
    end
end
% Displaying the input and output image
myNumOfColors = 200;
myColorScale = [ [0:1/(myNumOfColors-1):1]' , [0:1/(myNumOfColors-1):1]' ,
[0:1/(myNumOfColors-1):1]' ];
figure('name', 'Patch Based Filtering')
subplot(1,3,1)
imagesc(image);
colormap (myColorScale);
colormap gray;
daspect ([1 1 1]);
axis tight;
```

```
colorbar

title('Input Image after subsampling')

subplot(1,3,2)

imagesc(corrupted_image);

colormap (myColorScale);

colormap gray;

daspect ([1 1 1]);

axis tight;

colorbar

title('Corrupted Image')

subplot(1,3,3)

imagesc(output);

colormap (myColorScale);
```

```
colormap gray;

daspect ([1 1 1]);

axis tight;

colorbar

title('Output Image')

end
```

160010031_160010011_160010058_assignmentFiltering

ORIGINALITY REPORT

29%	5%	9%	29%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology, Bombay Student Paper	20%
2	Submitted to University of Southern California Student Paper	3%
3	Submitted to Rochester Institute of Technology Student Paper	2%
4	Submitted to University of Hertfordshire Student Paper	2%
5	Submitted to University of Sheffield Student Paper	2%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

160010031_160010011_160010058_assignmentFiltering