Machine Learning for Signal Processing and Pattern Classification.

IMAGEINPAINTING – LEAST SQUARE APPROACH

SIMRAN

1.

AIM:

To load the inpainted and image mask and replace the text (which is to be inpainted) with 'NaN (Not a Number)'.

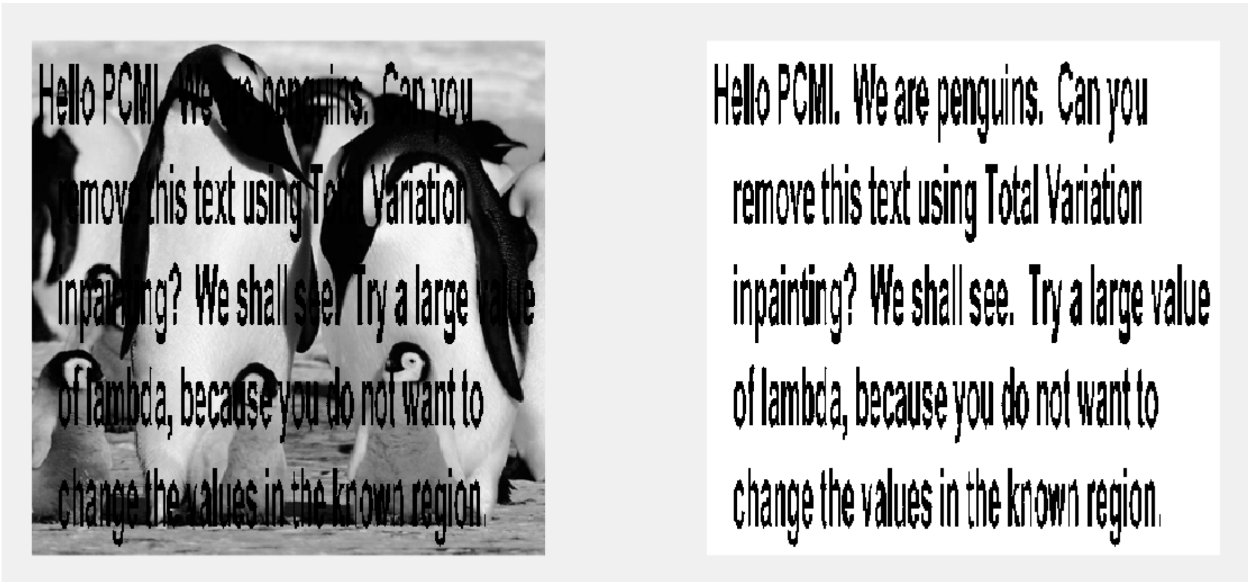
CODE:

```
clc;
clear all;
close all;
u0=double(imread('inpaint_img.tif'));  % To read the input image
to be inpainted.
D=imread('img_mask.tif');  % To read the corresponding mask
required for inpainting.
subplot(121);
imshow(uint8(u0));
subplot(122);
imshow(D); % To display the inputs in the single figure window.
u0 = mat2gray(u0);% converting to gray
u0=im2double(u0); % converting the image to double
u0=u0(:); % vectorising the image
D=D(:); % vectorising the mask
N=length(D); % N = length of the image
u0(D(:,1)==0)=NaN ; %replacing the text with NaN in the inpaint
image.
u0=reshape(u0,400,400); % reshaping the image back to the
original size.
figure
imshow(u0) % ploting the image with text.
title('image with text')
```

Grayscale image: black color - pixel value '0', white color - pixel value '1'.

Masked image we can get the position where the inpaint image is masked because there will be 0 in the masked positions.

OUTPUT:



2.

AIM:

To follow the code from above, implement the following steps and obtain the inpainted image as the output

Step 1: If a row contains NaN, estimate the values using least square approach for missing samples estimation. Repeat this for all the rows in the image.

Step 2: If a column contains NaN, estimate the values using least square approach for missing samples estimation. Repeat this for all the columns in the image.

Step 3: Compute the average of each and every pixel in the image to obtain the result.

CODE:

```
clc;
clear all;
close all;
u0=double(imread('inpaint_img.tif'));  % To read the input image
to be inpainted.
D=imread('img_mask.tif');  % To read the corresponding mask
required for inpainting.
subplot(121);
imshow(uint8(u0));
subplot(122);
imshow(D); % To display the inputs in the single figure window.
u0 = mat2gray(u0);% converting to gray
u0=im2double(u0); % converting the image to double
u0=u0(:); % vectorising the image
D=D(:); % vectorising the mask
N=length(D); % N = length of the image
u0(D(:,1)==0)=NaN ; %replacing the text with NaN in the inpaint
image.
u0=reshape(u0,400,400); % reshaping the image back to the
original size.
figure
imshow(u0) % ploting the image with text.
title('image with text')

%ROW WISE
u1=[]; % to store rowwise missing values
for j=1:400

    y=u0(j,:)'; % each row of image
    N = length(y); % length of the image

%% Define matrix D
% D represents the second-order derivative
% (2nd-order difference).
% D is defined as a sparse matrix so that Matlab
% subsequently uses fast solvers for banded systems.

e = ones(N, 1);
D = spdiags([e -2*e e], 0:2, N-2, N);

%% Define matrices S and Sc
```

```
k = isfinite(y);                    % k : logical vector,
indexes known values

S = speye(N);
S(~k, :) = [];                      % S : sampling matrix

Sc = speye(N);                      % Sc : complement of S
Sc(k, :) = [];

L = sum(~k);                         % L : number of missing
values

%% Estimate missing data
% Least square estimation of missing data.
% Note that the system matrix is banded so the system
% equations can be solved very efficiently with a fast banded
system solver.
% By defining S and D as sparse matrices, Matlab calls a fast
% banded system solver by default.

v = -(Sc * (D' * D) * Sc') \ ( Sc * D' * D * S' * y(k));    % v
: estimated samples

%% Fill in unknown values
% Place the estimated samples into the signal.

x = zeros(N,1);
x(k) = y(k);
x(~k) = v;

u1=[u1;x']; % appending each row after missing sample estimation
end
figure
imshow(u1) % plotting the image after rowwise missing sample
estimation.
title('image after row wise missing sample estimation')

%COLUMN WISE
u2=[]; % to store column wise missing values
for j=1:400
    y=u0(:,j);
    N = length(y);

%% Define matrix D
% D represents the second-order derivitive
% (2nd-order difference).
% D is defined as a sparse matrix so that Matlab
```

```
% subsequently uses fast solvers for banded systems.

e = ones(N, 1);
D = spdiags([e -2*e e], 0:2, N-2, N);

%% Define matrices S and Sc

k = isfinite(y);                    % k : logical vector,
indexes known values

S = speye(N);
S(~k, :) = [];                      % S : sampling matrix

Sc = speye(N);                      % Sc : complement of S
Sc(k, :) = [];

L = sum(~k);                         % L : number of missing
values

%% Estimate missing data
% Least square estimation of missing data.
% Note that the system matrix is banded so the system
% equations can be solved very efficiently with a fast banded
system solver.
% By defining S and D as sparse matrices, Matlab calls a fast
% banded system solver by default.

v = -(Sc * (D' * D) * Sc') \ ( Sc * D' * D * S' * y(k));    % v
: estimated samples

%% Fill in unknown values
% Place the estimated samples into the signal.

x = zeros(N,1);
x(k) = y(k);
x(~k) = v;

u2=[u2 x]; % appending each column after missing sample
estimation

end
figure
imshow(u2) % plotting the image after column wise missing sample
estimation.
title('image after column wise missing sample estimation')

u= (u1+u2)./2; %taking average
```

```
figure
imshow(u) % plotting the image after taking average
title('image after average')
```

Mask is oriented horizontally so row wise will be able to estimate more missing sample. Row wise missing sample estimation is clearer than column wise missing sample estimation.

OUTPUT:

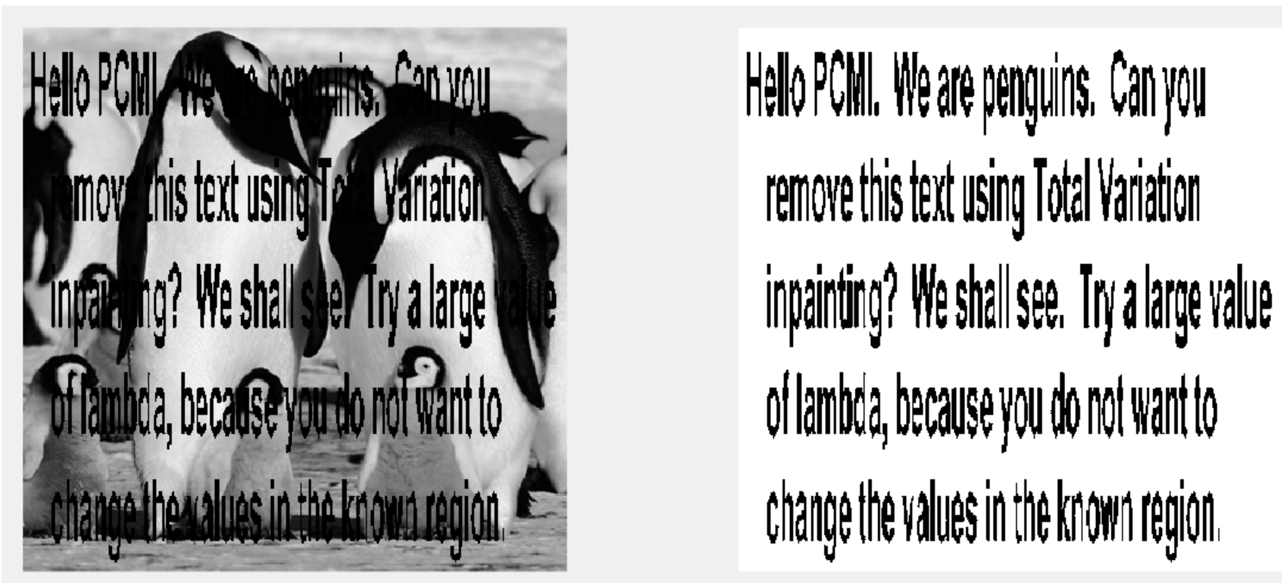


image with text

image after row wise missing sample estimation

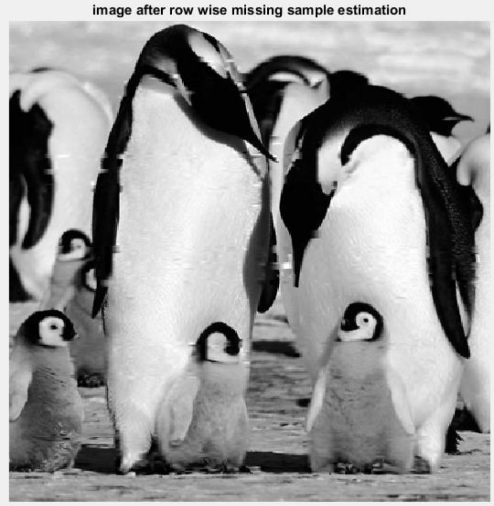



image after column wise missing sample estimation


image after average