DIP PROJECT

REPORT

Fast Digital Image Inpainting

By
Naila Fatima (201530154)
Shreya Gupta(201531123)
Vysyaraju NagaSunethra
(201530072)

Project Mentor
Saumya Rawat

Professor
Mr. Avinash Sharma

Abstract:

We present simple inpainting algorithms for reconstruction of small missing and damaged portions of images that is two to three orders of magnitude faster than current methods while producing comparable results.

Introduction:

Reconstruction of missing or damaged portions of images is an ancient practice used extensively in artwork restoration. Also known as inpainting or retouching, this activity consists of filling in the missing areas or modifying the damaged ones in a non-detectable way by an observer not familiar with the original images. We have used the algorithm mentioned in our paper. In order to compare the performance, we have inpainted the images using PDE method. We showed the superiority of these two methods over simple methods such as nearest neighbours and median diffusion.

Methods:-

1.) Main Algorithm

Let R be the region to be inpainted (this region will be user defined by providing a mask) and dR be the band (known part in the image, i.e the boundary of the region R from which we will be propagating the colour information). Region R is initialized to zero and then isotropic diffusion is used to propagate the information from dR into R. The boundary dR is a one pixel thick boundary. The diffusion is carried out by using a kernel such as one similar to the Gaussian kernel or averaging kernel with zero weight to the middle pixel. The diffusion kernel is convolved with the region to be inpainted for each iteration. Number of iterations to be performed can be controlled by checking if none of the pixels belonging to the region R had their values changed by more than a certain threshold during the previous iteration. Alternatively, number of iterations can be user defined. The paper has used the

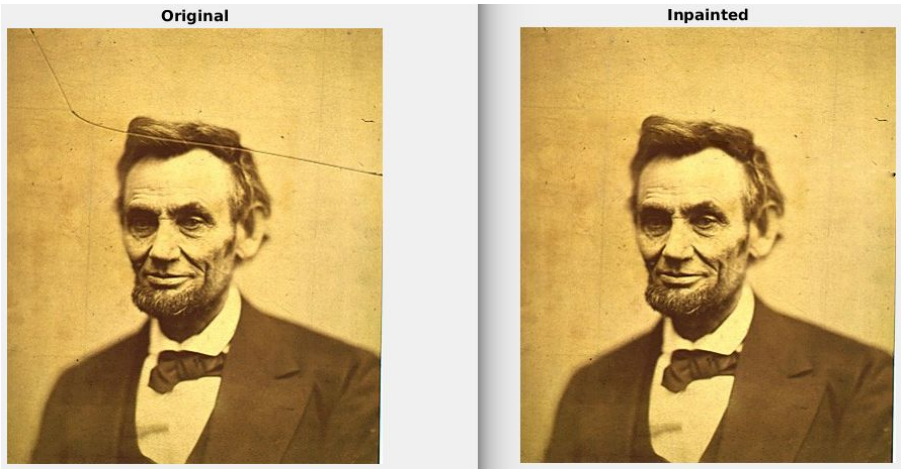
second approach by fixing the number of iterations to 100. We have done the same.

This algorithm uses a weighted average kernel that only considers contributions from the neighbor pixels (i.e., it has a zero weight at the center of the kernel). We have used two different types of kernels- one of the type [a b a;b 0 b;a b a] and the other of type [c c c;c 0 c;c c c] with a = 0.073235, b = 0.176765 and c = 0.125. The first kernel is a bit similar to Gaussian kernel whereas the second one resembles an averaging kernel. Since the kernels are of the size 3x3, we can observe that at the outermost pixels of region R, a one pixel boundary would be considered from the image which propagates the colour inwards. Since the kernel size used is 3x3 and as the convolution is carried out only within the region R, we do not need to specifically create a boundary dR.

Since we are using isotropic diffusion, blurring will be observed at the points where region R(to be inpainted) and edges in the image intersect. To eliminate this blurring, anisotropic diffusion must be carried out near the edges. Anisotropic diffusion requires additional complexity which is why we instead use diffusion barriers which eliminate blurring at the edges. Diffusion barriers were defined by user input and the previous algorithm was changed such that isotropic diffusion would be carried out only in the absence of edges in the neighbourhood (where there is no diffusion barrier) being considered. In the presence of diffusion barriers, the intensity of the neighbourhood pixel with the most similar gradient would be copied into the pixel being inpainted. Note that diffusion barriers and anisotropic diffusion give same output but anisotropic diffusion has more complexity.

Results

Without Diffusion Barriers:

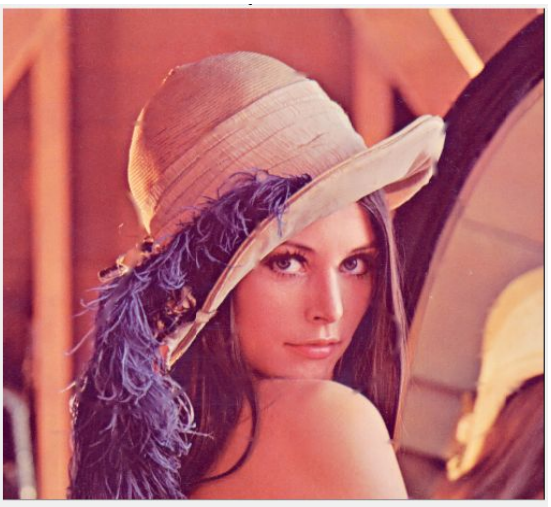


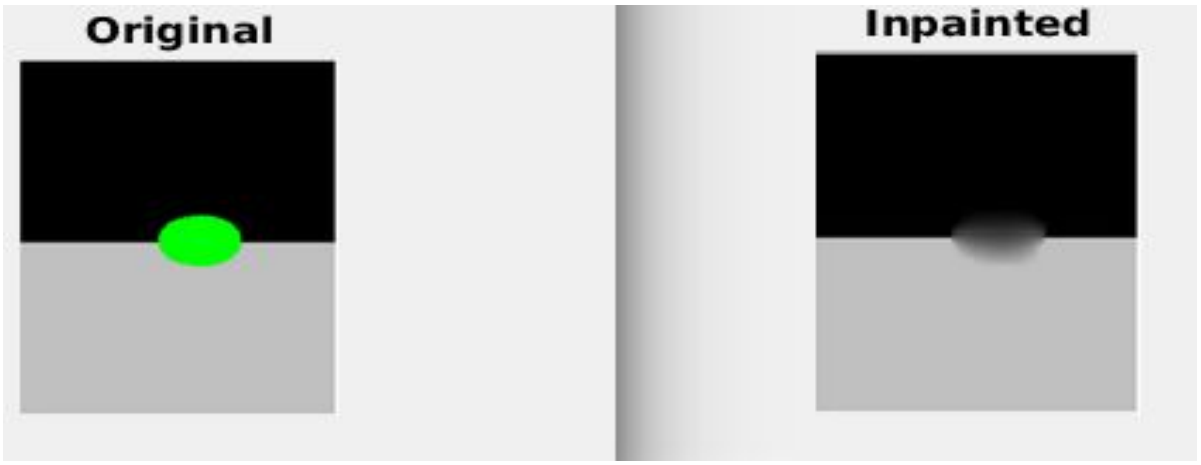
Original

Inpainted

Original

Inpainted




Original
Inpainted

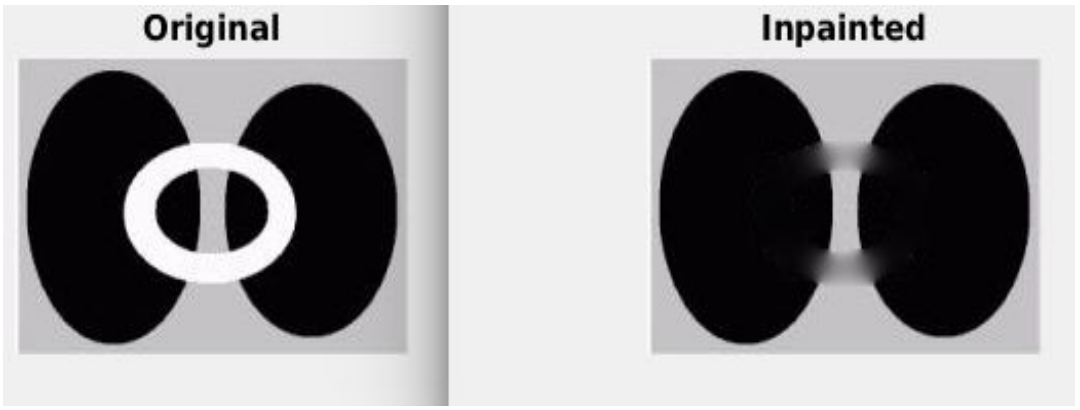
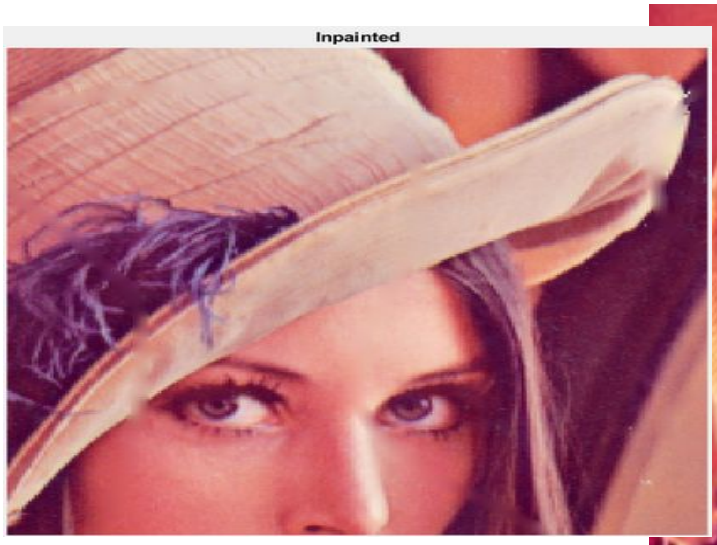
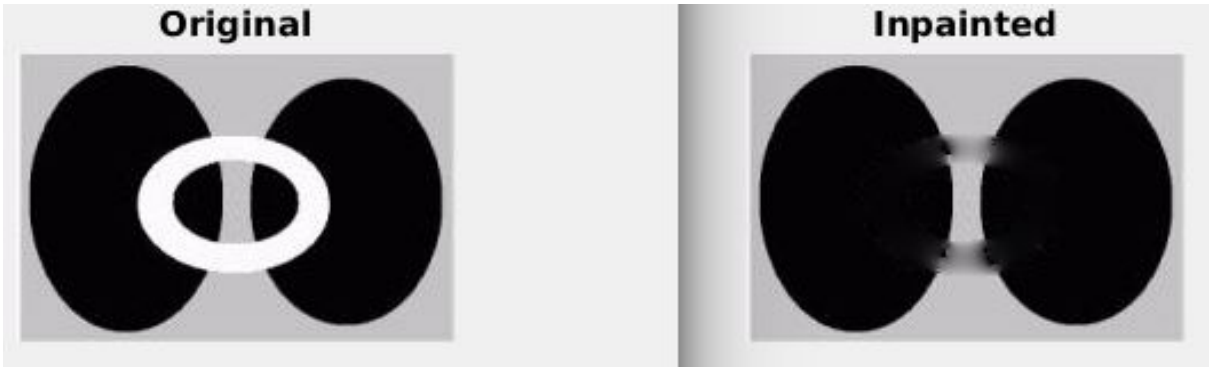

Original
Inpainted

Fig. A

With Diffusion Barriers

Isotropic diffusion causes blurring near regions where an edge must be present within R. By using diffusion barriers, we can get rid of the blurring. The simple diffusion-based inpainting algorithm produces blurred spots at the intersections between Ω and high contrast edges . By appropriately adding diffusion barriers, the user stops the diffusion process from mixing information from both sides of the mask. Diffusion barrier is 2-pixel width boundary.


Inpainted


Original
Inpainted

2.) Second Algorithm (PDE)

PDE stands for Partial Differential Equations

This algorithm also uses user-defined mask to inpaint the corrupted part of the image. This algorithm treats the input image as three separate channels (R,G and B). For each channel, it fills in the areas to be inpainted by propagating information from the outside of the masked region along level lines , i.e, isophotes. Isophote directions are obtained by computing at each pixel along the inpainting contour a discretized gradient vector and by rotating the resulting vector by 90 degrees. 2-D laplacian of each channel of the image is used to locally estimate the variation in color smoothness and then this variation is propagated along the isophote direction.

The equation that we will be using in order to fill in the region R is:

$$dI(i,j)/dt = dL(i,j) \cdot N(i,j)$$

where L is the information we want to propagate into region R, N is the direction in which we want to propagate the information and dL is the change in L.

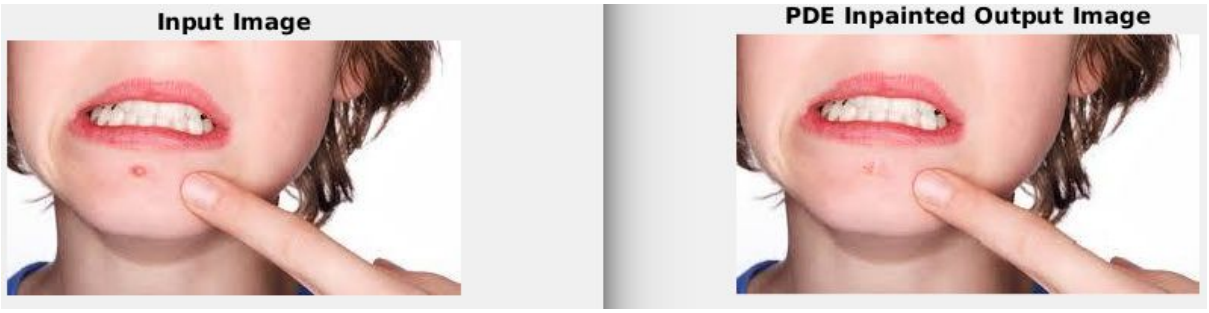
L is laplacian of the image. N will give us the isophote direction. Isophote is a curve on an illuminated surface that connects points of equal brightness.

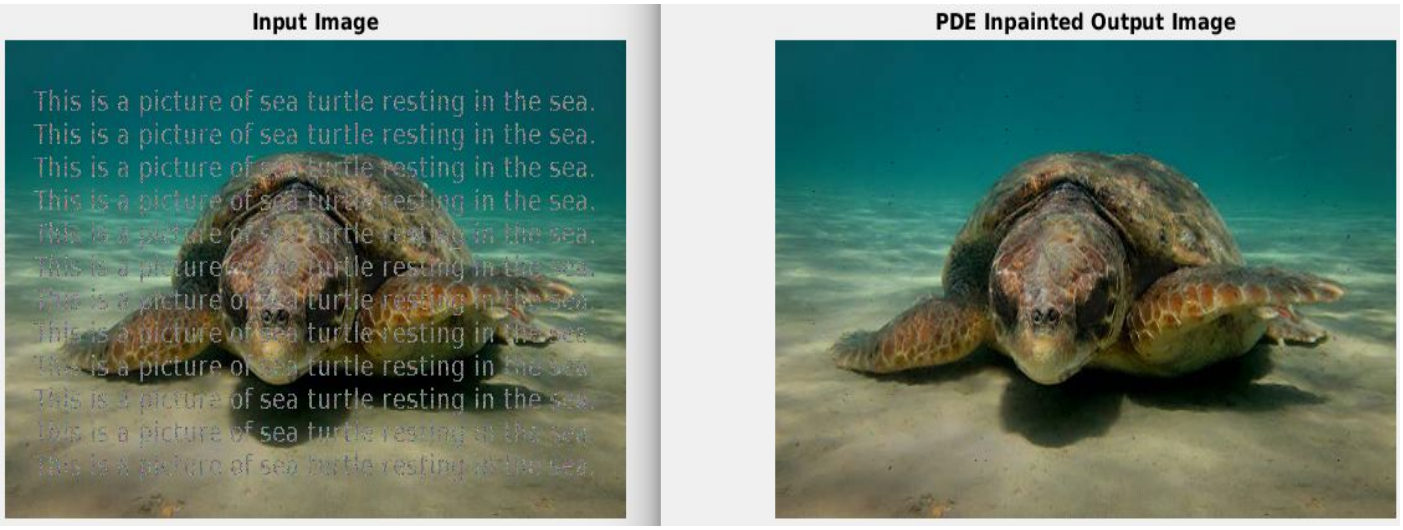
We can easily note that dL.N (dot product) will be the change in L in the direction of N. Ideally, we want this dot product to be 0 as this then means that L is constant in the direction of propagation. If L is not constant, this leads to a jump in intensities which we may be able to perceive. Since we want the dot product of dL and N to be zero, this is equivalent to saying that we want dL to be perpendicular to N.

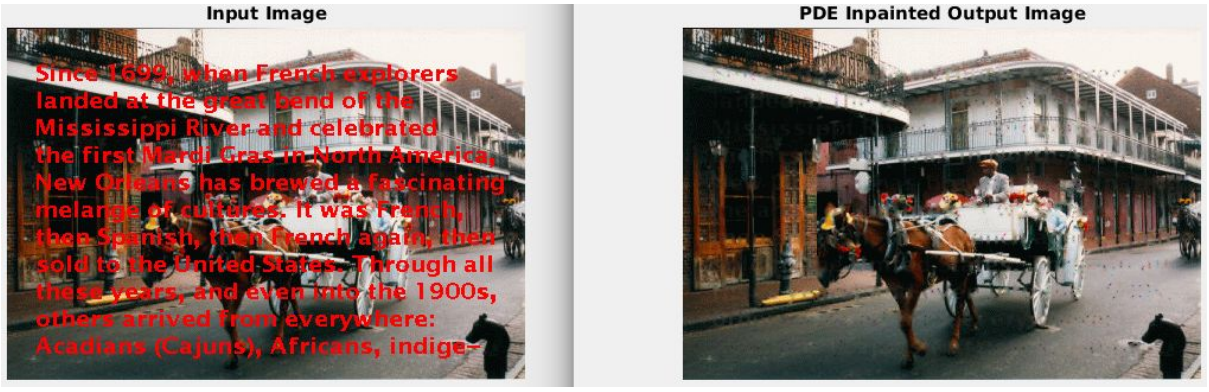
These steps will be repeated again and again until we get dI(i,j)/dt = 0 or less than some threshold value, for each pixel or the number of iterations

can be fixed beforehand.

Results Obtained:-



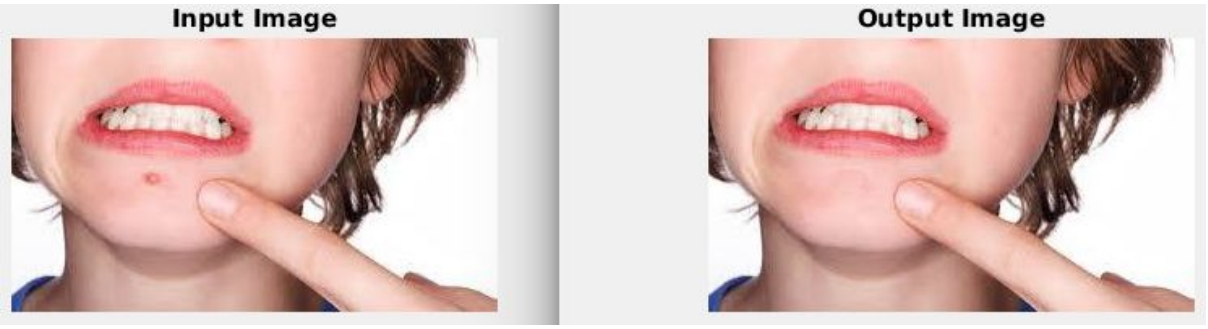




3.) Third Algorithm(Median Diffusion):

This algo is similar to the first algorithm. This algorithm uses median of the window for finding the intensity value of the corrupted or damaged pixels in the image in place of gaussian averaging used in the first algorithm.

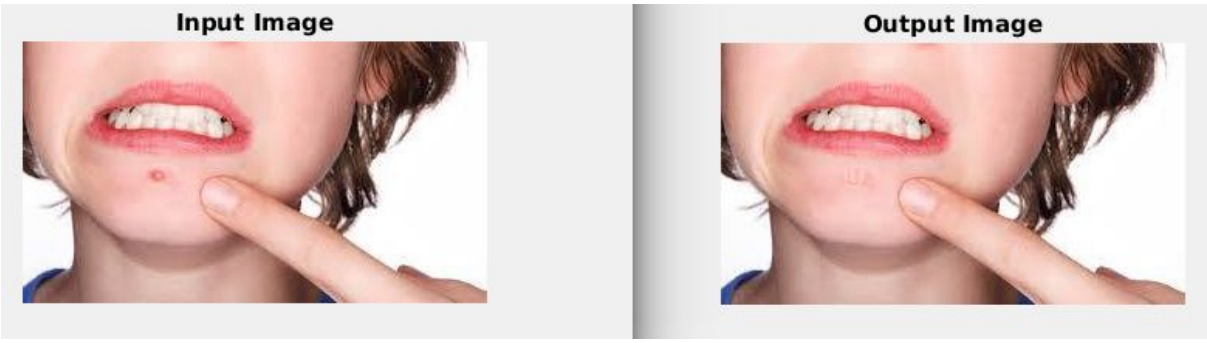
Results:

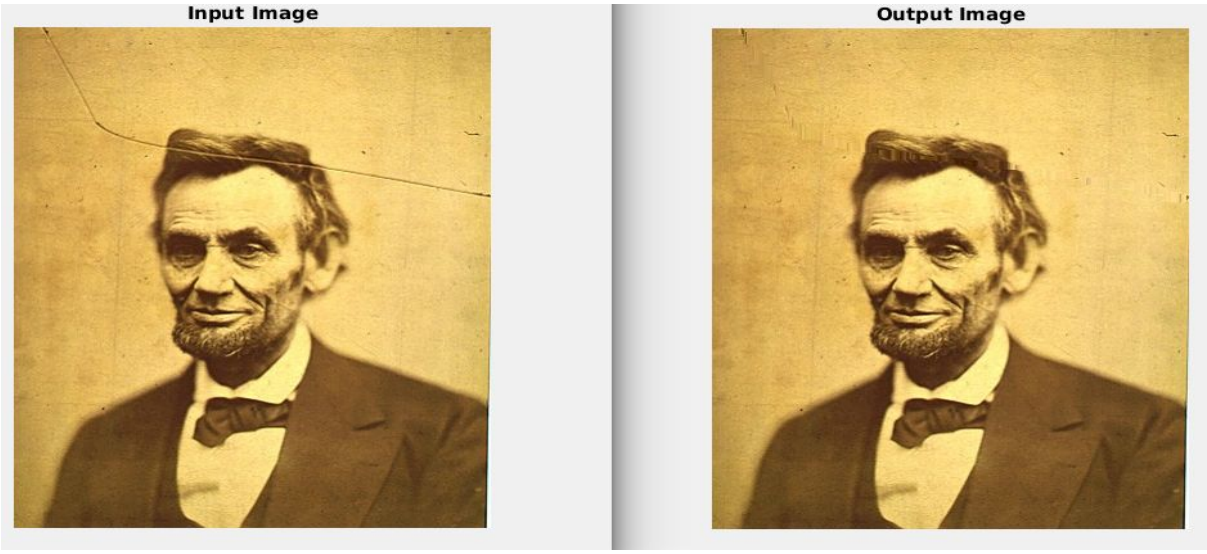


4.) Fourth Algorithm(Nearest Neighbour):

This algo is also a variation of first algorithm. This algorithm considers the intensity value of nearest neighbour of a pixel to find out intensity value of the corrupted or damaged pixels in the image in place of gaussian averaging used in the first algorithm.

Results:



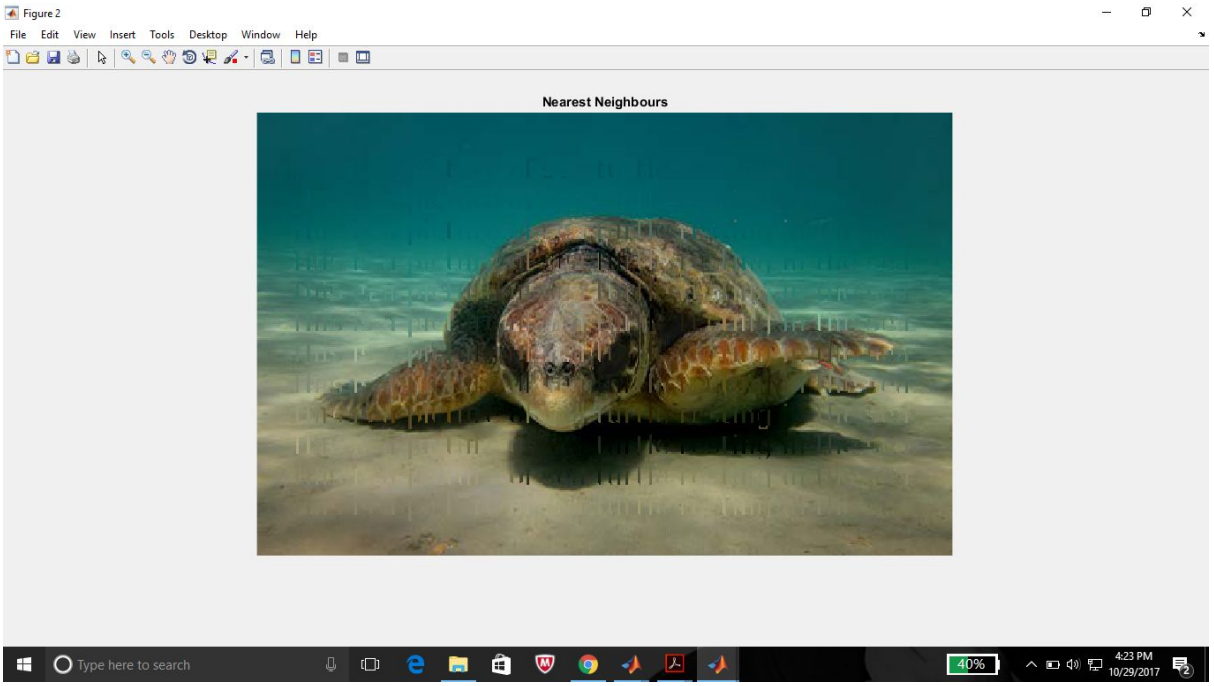


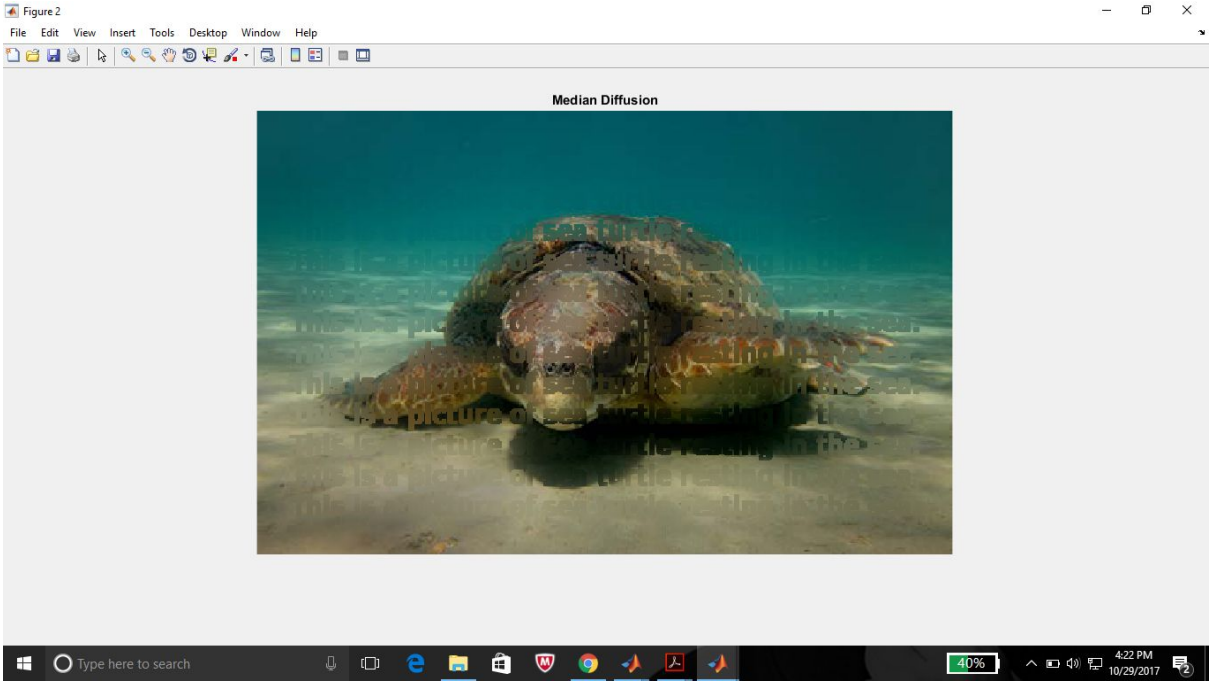
Observation:

We use four algorithms (Gaussian Average, PDE, Nearest Neighbour and Median Diffusion). We observed that our main algorithm gives better outputs compared to other algos. Median diffusion and nearest neighbour will work only if the surrounding areas are smooth. Note that generally, the outputs of median diffusion and nearest neighbour are poor compared to those of our algorithm and PDE. This can be seen in the below two images.

For main algorithm, diffusion barriers are used for getting better

edges in the restored image.





Applications:

1. Restoring damaged photographs, films and paintings.
2. Removal of occlusions, such as text, subtitles, stamps and publicity from images.

3. To produce special effects.
4. To remove some objects from the images (like in Fig. A removing insect from the image).
5. To recover pixels lost while compression and transmission of images.

Limitations

These algorithms cannot be used if the region to be inpainted is large and if it includes high contrast edges and complex background.

Related Work

The PDE algorithm is a good substitute for our algorithm as it provides results which are nearly similar. We can note however that the code for PDE is a bit complex when compared to our algorithm. The PDE algorithm by Bertalmio gives outputs similar to our algorithm. Other algorithms which may be used for inpainting include Telea's FMM (Fast Marching Method) as well as Exemplar-Based Image Inpainting by Criminisi. Note that exemplar-based inpainting is complex compared to our algorithm but it can be used to inpaint large regions- something our algorithm cannot do.