IMAGE INPAINTING

Final Project Report

Team Members:

SHOEB SIDDIQUI (2018701017)

SOWMYA AITHA (2018702007)

TANU SHARMA (2018702012)

CONTENTS

1. INTRODUCTION

1.1 PROBLEM STATEMENT

Inpainting is the process of reconstructing lost or deteriorated parts of images and videos. In the digital world, inpainting refers to the application of sophisticated algorithms to replace lost or corrupted parts of the digital image data (small regions or to remove small defects).

This project focuses on digital image inpainting.

1.2 PROJECT MOTIVATION

Many times we observe that old photographs get deteriorated over time (crack, scratches, torn etc). In order to restore such photographs, we can take their digital images and apply image inpainting to remove the corresponding defect.

In images where stuntmen use cables or cords and actors use microphones, these objects can be made invisible by image inpainting.

Some images may have text embedded or overwritten on them. In such cases, background image (eliminating text) can be recovered using image inpainting.

1.3 OBJECTIVE

The project achieves following objectives:
1) Implementing Image Inpainting algorithm
2) Implementing Text Detection algorithm

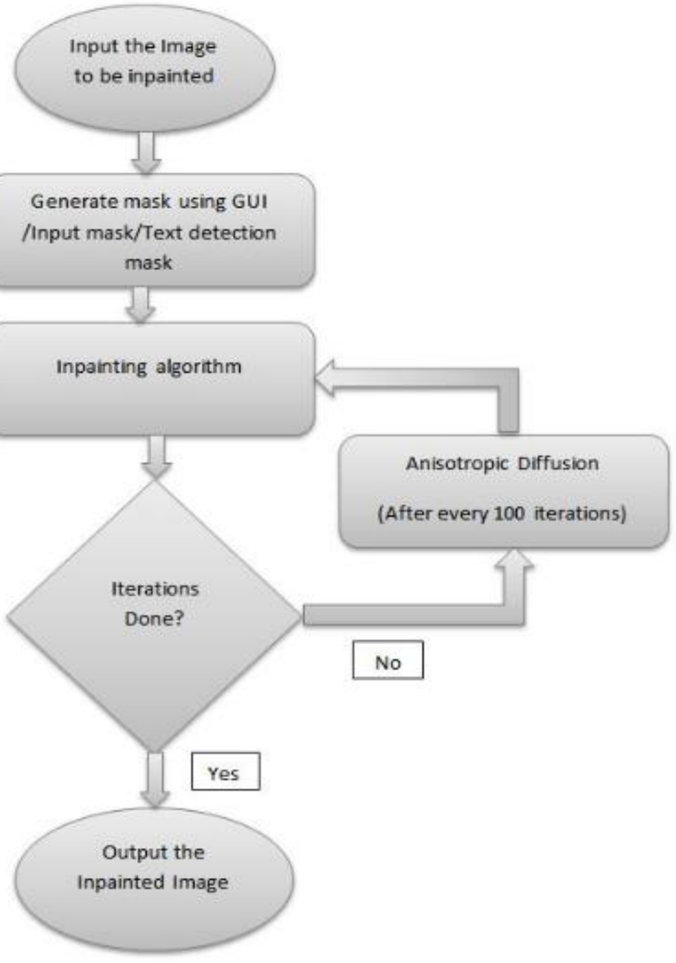
2. APPROACH CONSIDERED

Let Ω denote the area to be inpainted and $\partial\Omega$ denote its boundary. The technique we implemented will prolong the isophote lines arriving at $\partial\Omega$ while maintaining the angle of arrival.

The algorithm includes following steps:

1) Obtaining location of pixels to be inpainted using -
    a) Text Detection
    b) Generating Mask using GUI
2) Performing following steps iteratively and simultaneously -
    a) The structure of the area surrounding Ω is continued in the gap, contour lines are drawn via the prolongation of those arriving at $\partial\Omega$.
    b) The different regions inside Ω, as defined by the contour lines, are filled with color, matching those of $\partial\Omega$.
3) After every few iterations of step 2, anisotropic diffusion is performed to ensure correct evolution of direction.

Flow diagram:


Input the Image to be inpainted
Generate mask using GUI /Input mask/Text detection mask
Inpainting algorithm
Anisotropic Diffusion (After every 100 iterations)
Iterations Done?
No
Yes
Output the Inpainted Image

3. WORK PERFORMED

3.1 INFORMATION AND ISOPHOTES[1]

3.1.1 Isophote propagation using Laplacian information

This section deals with propagating the structure and contours lines inwards from the boundary of the region to be inpainted.
We do this in the following way:
1. Obtain the contours lines as isophotes
2. Obtain a curve description metric that needs to be propagated, as Laplacian.
3. Iteratively run the algorithm to gradually propagate information along the curves inwards into the region.

Basic equations used,

$$I^{n+1}(i,j) = I^n(i,j) + \Delta t I_t^n(i,j), \forall(i,j) \in \Omega \tag{3.1.1}$$

$$I_t^n(i,j) = \overrightarrow{\delta L^n}(i,j) \,.\, \overrightarrow{N^n}(i,j) \tag{3.1.2}$$

where,
$n = Inpainting\ time$
$I_t^n(i,j) = Update\ of\ the\ image\ I^n(i,j)$
$L^n(i,j) = Information\ to\ be\ propagated$
$\overrightarrow{N^n}(i,j) = Propagation\ direction$

The algorithm:
For each iteration, for each pixel in the inpainted region
1. Obtain the gradient of the Laplacian that signifies the direction of information evolution.
2. Obtain the norm (perpendicular) of gradient, signifying the direction of the isophote lines.
3. Obtain a normalised gradient, given the direction of propagation of the curve , and proportionally change the pixel value.

So why does this work and why do it in this manner?
1. We require a description for the contour lines which is aptly provided by the isophote lines, the curves containing points that have same pixel values.

2. Now that we know the initial direction that the curves need to proceed in, we try to evolve the curve. For this we use a curve descriptor i.e., information(L). We can use any metric that directionally describes the curve and attempts to iteratively propagate this information. We can't use gradient because isophote lines are perpendicular to gradients and the component of this gradient information will always be 0 along the direction we would want the information to be propagated in. Hence we use the Laplacian for this purpose which can model curves of order 2,

but will ignore higher order differentials of the curve. To propagate more complex curves accurately we can use descriptors that are of higher differential order.

3. We need to do this iteratively because the information that we require to propagate is only available at the end (the end that is being extended) of the curve. So we can't predict the interaction of curves with each other even though we can predict the evolution of just one curve individually. Therefore we extend each curve by a little in each iteration.

4. When we say that the algorithm propagates curves that could well mean that even the curves that we do not want in the region can get propagated. But this does not happen due to the fact that at the edge of the region the information metric remains constant since no change is being made outside the region. This retains and reinforces the curve and doesn't let the curve that exist at the edge of the region to vanish. This causes the unwanted curves within the region to die down while the ones on the edge prevail.

5. To maintain perceptual uniformity we perform the algorithm in the HSI planes rather than the RGB planes.

6. Anisotropic diffusion is used to assert the curves, i.e., sharpen the curves while smoothing out the smooth regions.

7. We use normalised gradient because we want the algorithm to be stable, i.e., due the nature of iterative changes in the pixel values it may happen that the process isn't ideally quasi-static as we'd like it to be. But we want the magnitude of change to be indifferent to such non-idealities and hence we ignore the gradient in certain directions in specific cases.

8. We can use padding that replicates the surrounding, if the padding is relevant.

Test runs:
The start point was as follows,

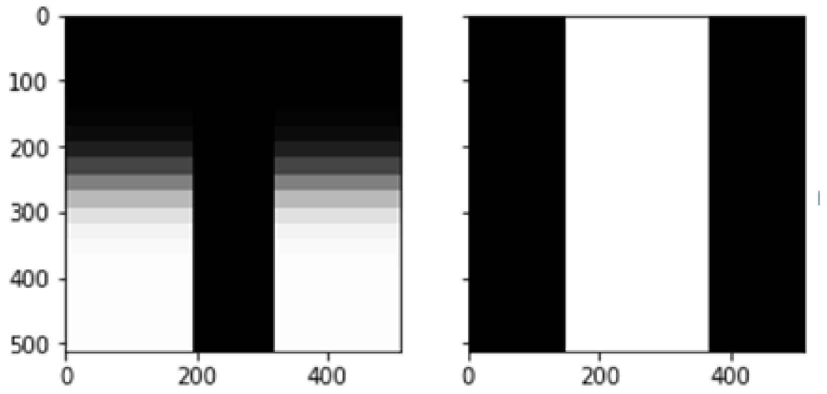


Fig 3.1.1: $Original\ image\ and\ the\ mask$

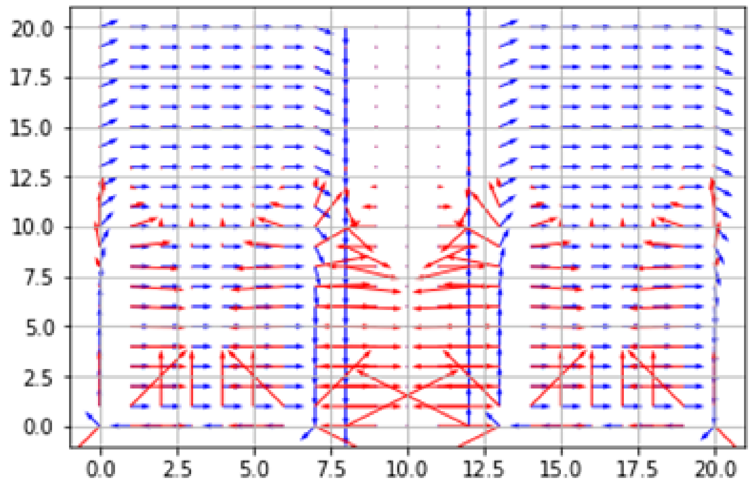


Fig 3.1.2: $Laplacian(left)$ and $Norm$ of $gradient$ in $blue, gradient$ of L in $red(right)$

After 10,000 iterations we get,

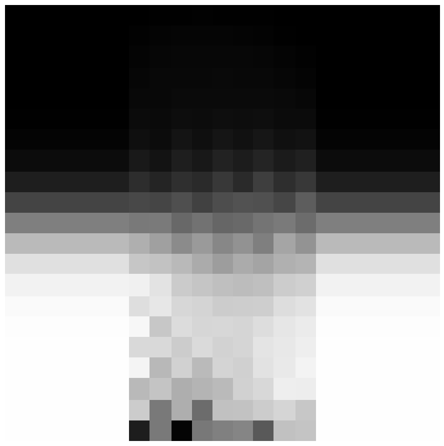
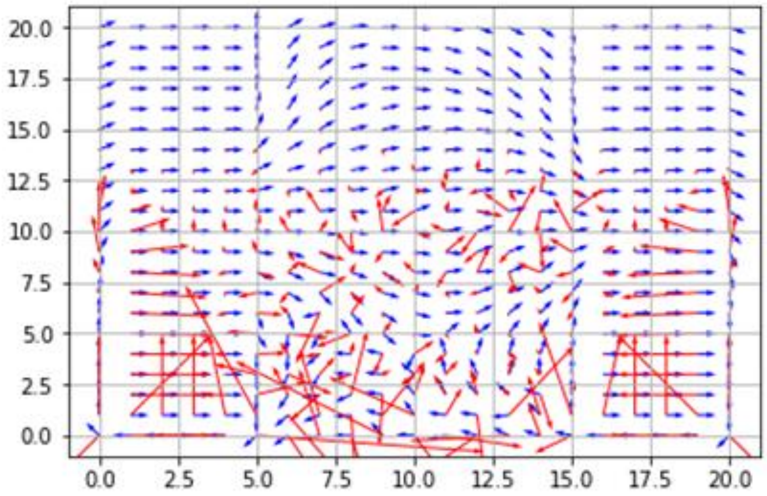



Fig 3.1.4: $Resulting$ $image(left)$ and $Norm$ of $gradient$ in $blue, gradient$ of L in $red(right)$

3.2 TEXT DETECTION

3.2.1.Text detection using Edge based method[3]

The purpose here is to find the location of pixels in which text is embedded in the image so the method which we have used produces very promising results which fulfills the requirement. The idea is to implement a simple texture based approach based on edge based algorithm. The main logic here is the strength of the text embedded in the image is more than the edges of the objects or background. This method produces better results for large sized text.

3.2.2.How does this work?

In texture based methods, text is assumed to represent a different texture than non text. Text is often considered as a periodic structure because the characters on a text line form a more or less periodic structure in the horizontal direction and the text line in turn form a periodic structure in the vertical direction.

Algorithm:

1.Convert the RGB to grayscale if input is color image.

2.Smoothen the image using Gaussian filter.

3.Find the Gradients of the image using Sobel operator in both the directions.

Magnitude of $G(i,j) = |G| = \sqrt{(G_x^2(i,j) + G_y^2(i,j))}$

Where $G_x(i,j)$ and $G_y(i,j)$ are the x and y-components of Gradient image G at pixel (i,j).

4.A threshold t1 is computed to eliminate weak edges

$$t1 = \sqrt{\frac{4 \times \sum_{i=1}^{h-1} \sum_{j=1}^{w-1} (G_x^2(i,j) + G_y^2(i,j))}{(h-1)(w-1)}} \qquad (3.2.1)$$

where h and w is the height and width of the image.

If G(i,j) > t1 then the pixel is assumed to be an edge and it is a local maxima in a 3x3 neighborhood along either horizontal or vertical direction.

5.If G(i,j)>t1, E(i,j)=255 else E(i,j)=0 thus we get an edge image of same size.

6.Now,the edge image is divided into small non-overlapping blocks of mxm pixels, where m depends on the image resolution and size. Typical values of m can vary from 6 to 12.For each block the feature F is calculated using any of the following formulae:

$$F = \sum_{i=1}^{m} \sum_{j=1}^{m} H(G(i,j) - t1) H(E(i,j) - 1) \qquad (3.2.2)$$

$$F = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} G(i,j) H(G(i,j) - t1) H(E(i,j) - 1)}{\sum_{i=1}^{m} \sum_{j=1}^{m} H(G(i,j) - t1) H(E(i,j) - 1)} \qquad (3.2.3)$$

$$F = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} G(i,j) H(G(i,j) - t1) H(E(i,j) - 1)}{m^2} \qquad (3.2.4)$$

Where E(i,j) is the edge image and H(i,j) is the step function.

$$H(x-a) = \begin{cases} 1 \; if \; x \geq a \\ 0 \; if \; x \leq a \end{cases} \qquad (3.2.5)$$

In equation(3.2.2),F means the number of edge pixels per block. In equations (3.2.3) and (3.2.4) F is the average gradient magnitude per edge pixel and average gradient magnitude per each pixel respectively.

F defined above reflects the fact that the number of edge pixels and their gradient magnitudes are usually higher for text than for non text blocks.

7.Now the block classification is performed with a predefined threshold t2.(Blocks with F2>t2 are assigned as block with text and rest with non text). However it is difficult to determine the value of t2 automatically.

8. Finally close and dilate the obtained mask image using 3x3 all ones structuring element.

Outputs of text detection algorithm:

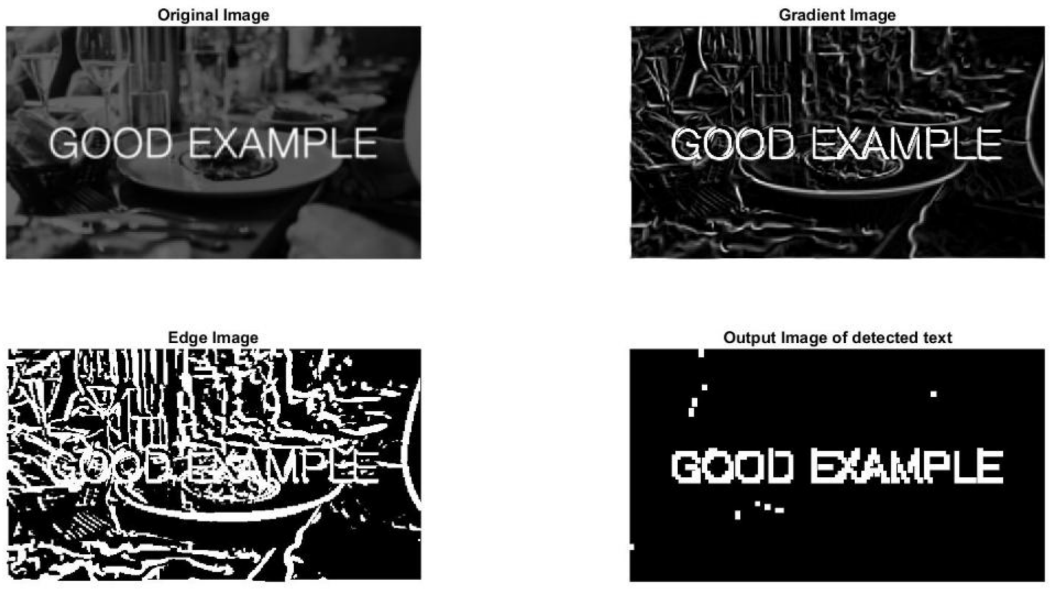
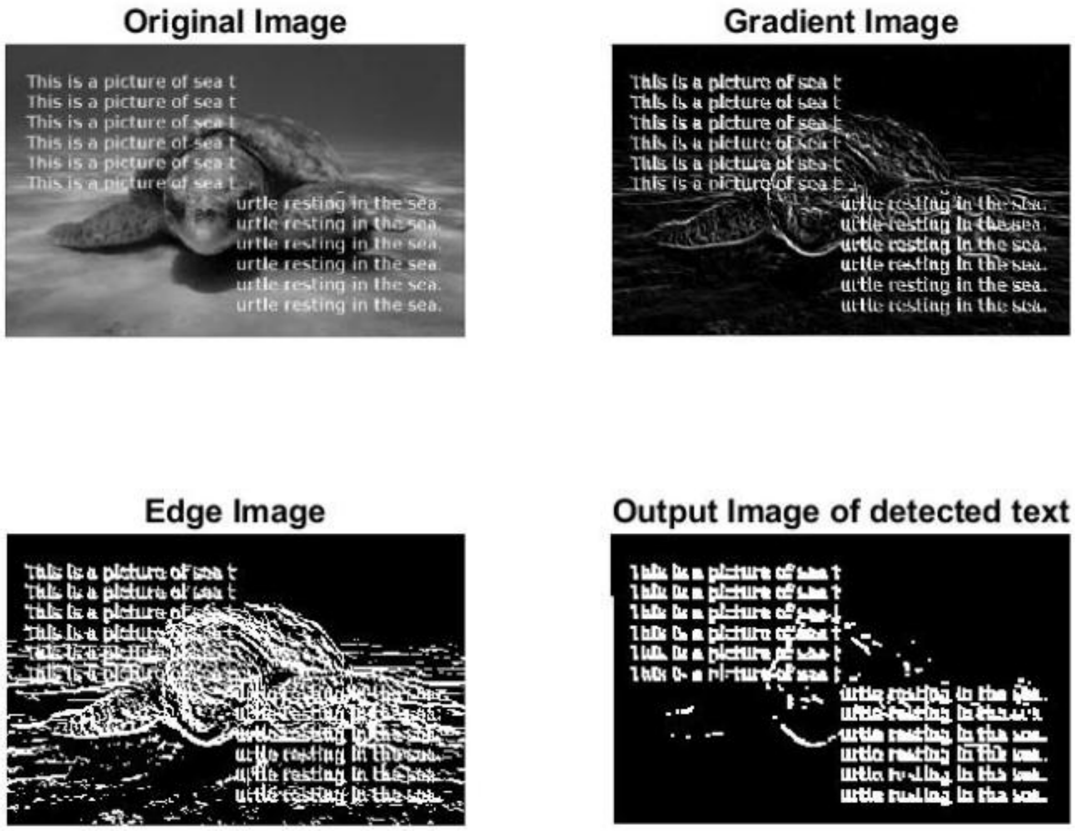


Fig 3.2.1: $Text\ detection\ done\ using\ window\ size\ =$ 2x2


Original Image
Gradient Image
Edge Image
Output Image of detected text
This is a picture of sea t
urtle resting in the sea.

$Fig\ 3.2.2: Text\ detection\ done\ using\ window\ size\ =\ 2x2$

3.3 ANISOTROPIC DIFFUSION[2]

Anisotropic diffusion is the method for nonlinear diffusion such that the following criteria are satisfied:

1) Causality: No spurious details are generated in the process

2) Immediate Localization: The region boundaries should be sharp and coincide with the semantically meaningful boundaries.

3) Piecewise Smoothing: Intraregion smoothing should occur preferentially over interregion smoothing.

So, the anisotropic diffusion equation suggested by Perona and Malik is,

$$I_t = div(c(x,y,t)\nabla I) = c(x,y,t)\Delta I + \nabla c.\nabla I \qquad (3.3.1)$$

where,

$div = divergence\ operator$

$\nabla, \Delta = \textit{gradient and laplacian operators respectively}$

$c = \textit{diffusion coefficient}$

The causality criteria requires that no new features are added in the image during the diffusion process. Addition of a new feature to the image implies creation of either a maximum or minimum. Therefore, the causality criteria can be established by showing that all maxima and minima belong to the original image. The diffusion equation (3.3.1) is a special case of a more general class of elliptic equations satisfying a maximum principle i.e., all maxima of the solution of the equation belong to the initial condition (the original image, in our case) and to the boundaries of the domain of interest provided that the conduction coefficient is positive.

The piecewise smoothing can be done by setting the conduction coefficient to 1 in the interior of each region and 0 at the boundaries. The blurring will take place separately in each region with no interactions among regions. For this we take current best estimate of the location of boundaries which is obtained by ∇I(x,y,t). The conduction coefficient,

$$c(x,y,t) = g(||\nabla I(x,y,t)||) \qquad (3.3.2)$$

where g(.) is a non-negative monotonically decreasing function such that g(0) =1. If Suitable function for g is chosen then not only the edges are preserved but also brighten. g(.) taken for the anisotropic diffusion for the project is,

$$g(u) = \frac{1}{\left(1 + \left(\frac{u}{K}\right)^2\right)} \qquad (3.3.3)$$

where, K = Threshold value

Algorithm:

1) Take the input image and split it into its individual channels if it is RGB.

2) Find the gradient of each channel in all the eight directions using respective filters like for gradient in north direction, use [0,1,0;0,-1,0;0,0,0].

3) For each of the 8 gradient images, calculate diffusion constants using equation (3.3.3) where input to the function g(.) is the gradient image.

4) Obtain sum of the dot products of each diffusion constant with the respective gradient images.

5) Update the channel by adding some fraction of the improvement obtained in step 4 to it.

6) Step 2 to 5 are repeated for user defined number of iterations.

7) Finally, the separate channels are concatenated in case of RGB image.

Outputs generated after applying anisotropic diffusion to images:

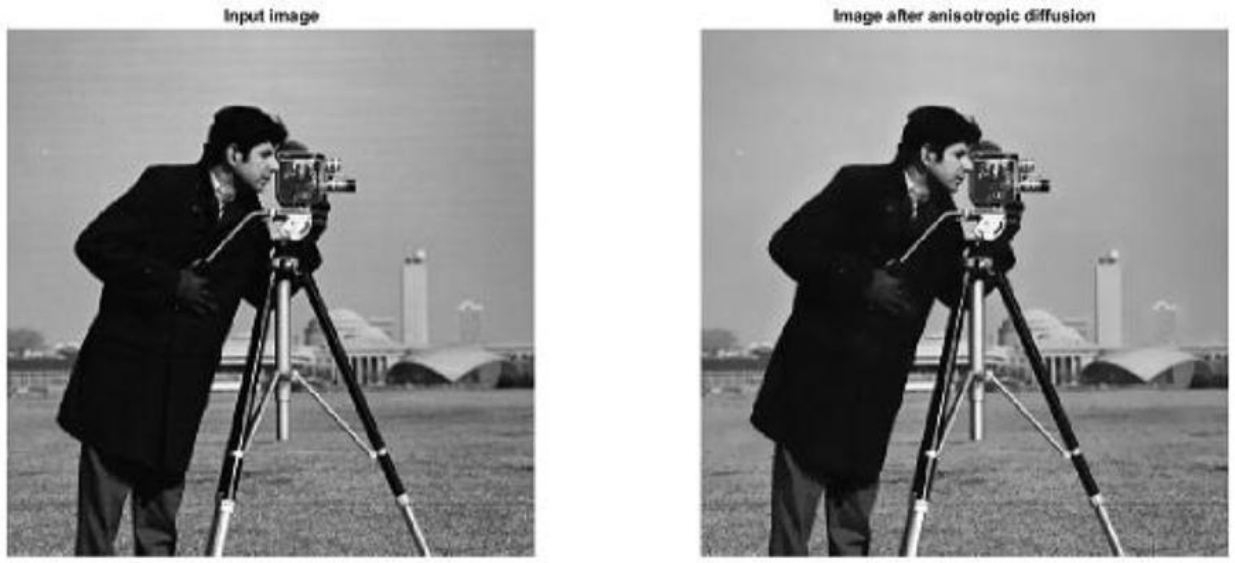


Fig 3.3.1: $Anisotropic\ diffusion\ with\ K = 5$

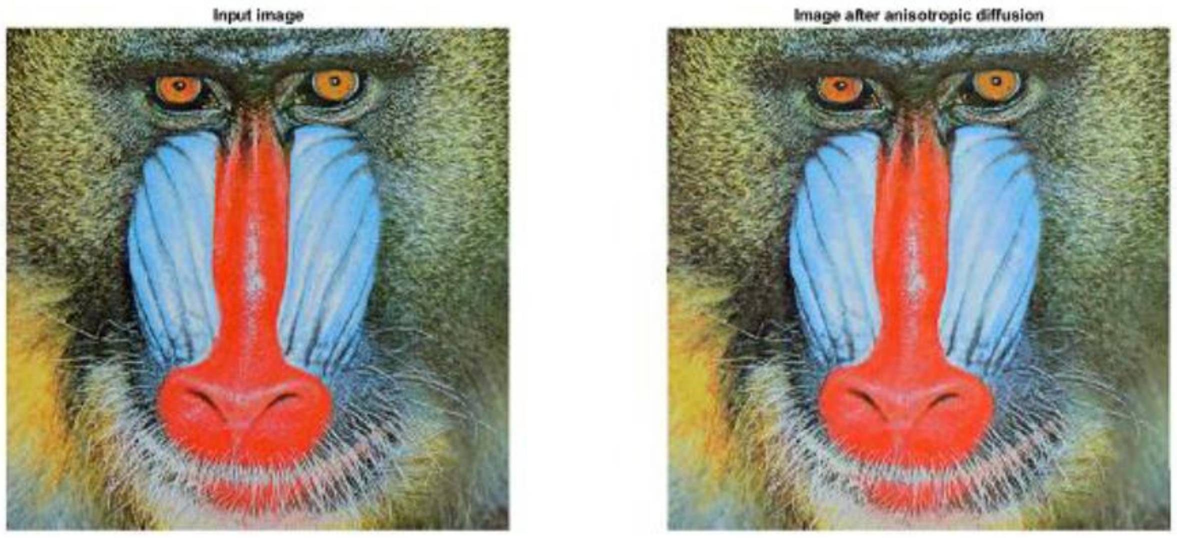


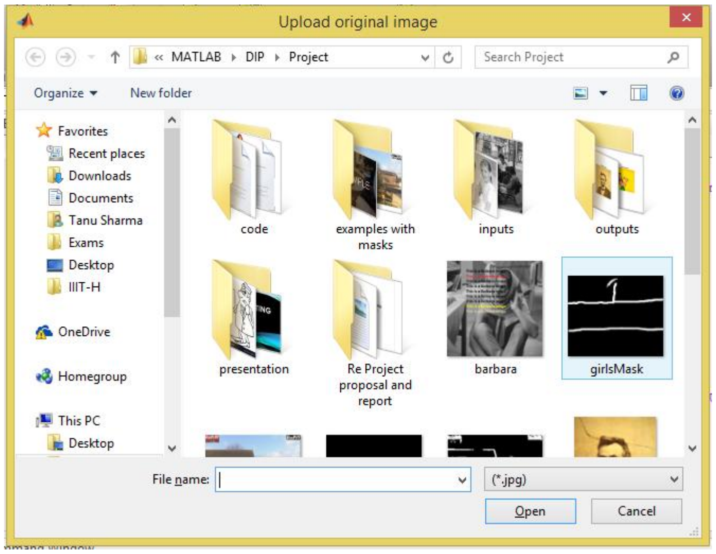
Fig 3.3.2: $Anisotropic\ diffusion\ with\ K = 5$

In Fig 3.3.1, we can see that the horizontal lines in the sky region are smoothened out . In Fig 3.3.2, the fine hair strand in the left quadrant near the edges are smoothened out. In both the images, we observe the edges are preserved even after doing anisotropic diffusion.

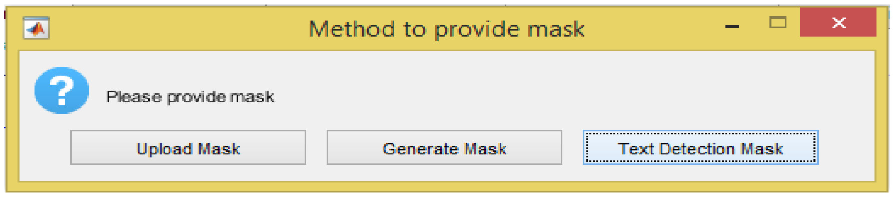
3.4 WORK FLOW OF OUR PROGRAM

Images to show how the program runs

Step 1: Upload the image to be inpainted

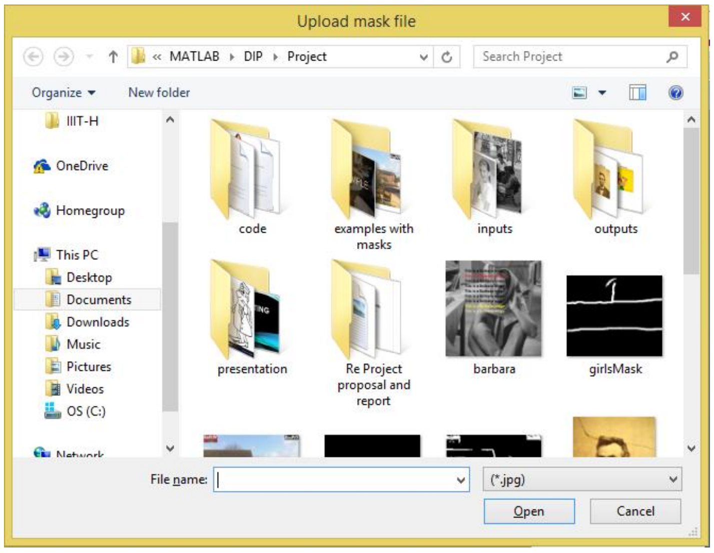


Step 2: Provide mask for the region to be inpainted in the original image.

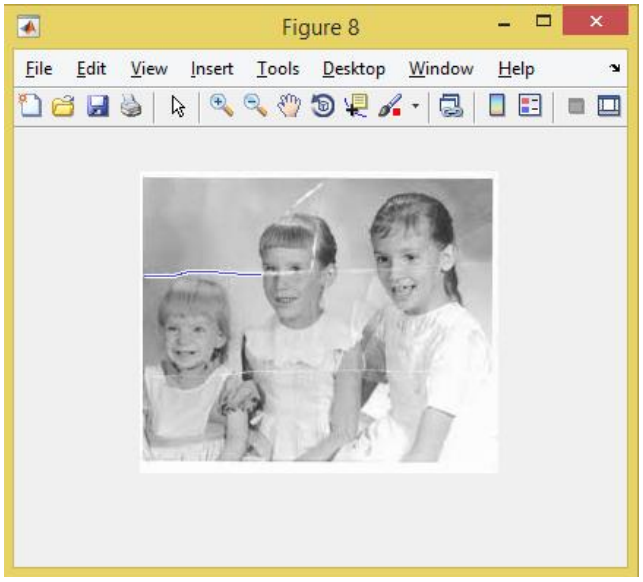

Method to provide mask

Please provide mask

Step 3: When 'Upload Mask' is chosen, user is given option to upload mask image file.



Step 4: When 'Generate Mask' is chosen, a GUI opens where user can draw mask (blue line) on the original image.

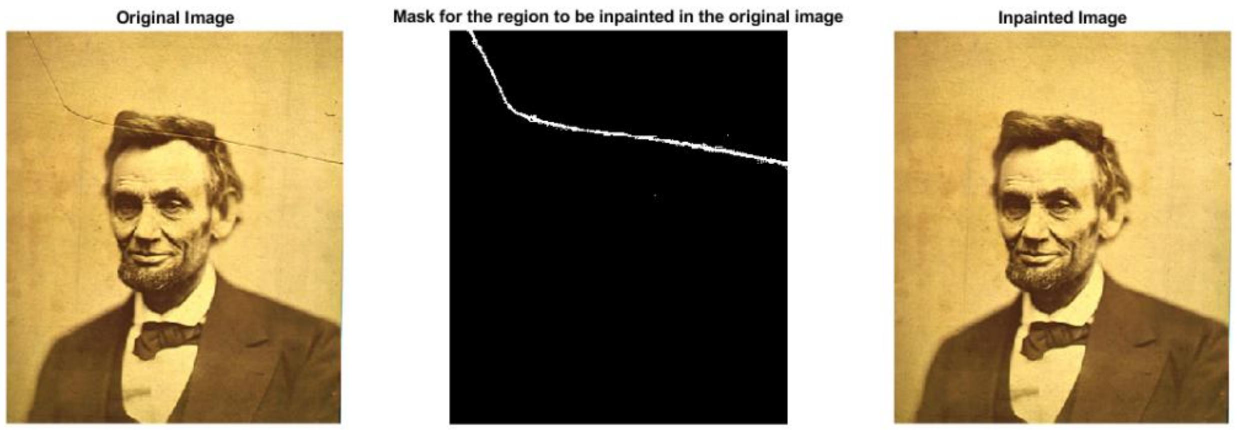


Step 5: When 'Text Detection' is chosen, the program returns a mask containing the text in the original image.

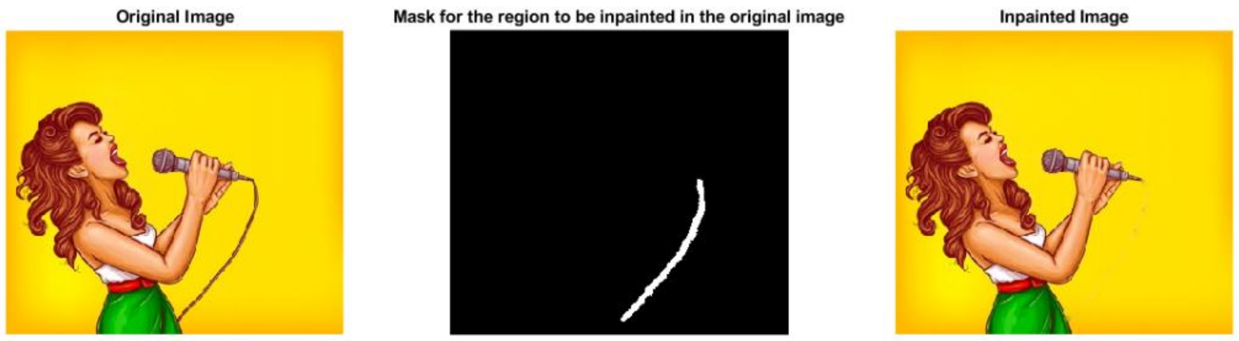
Step 6: Inpainting algorithm generates the final inpainted image using the original image and the mask.

4. RESULTS

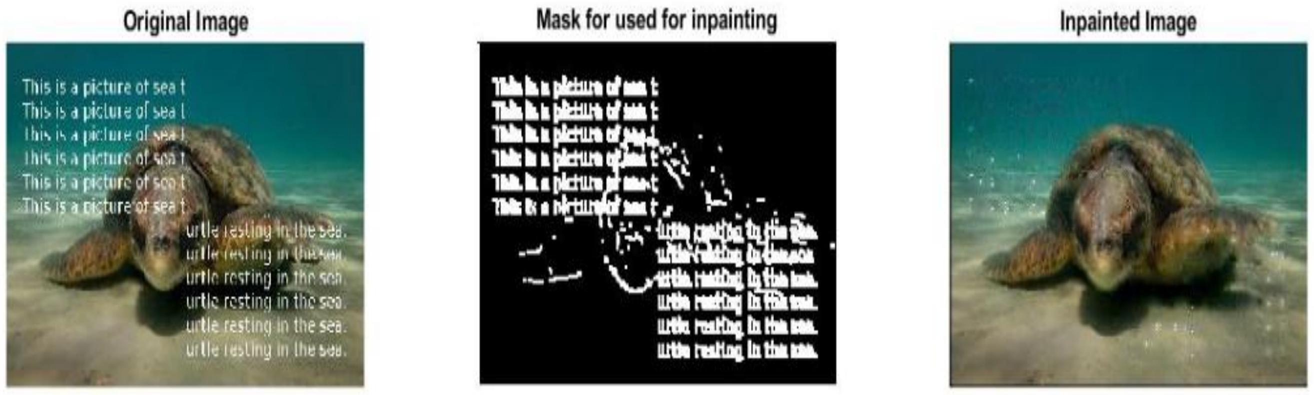
4.1 SUCCESSES



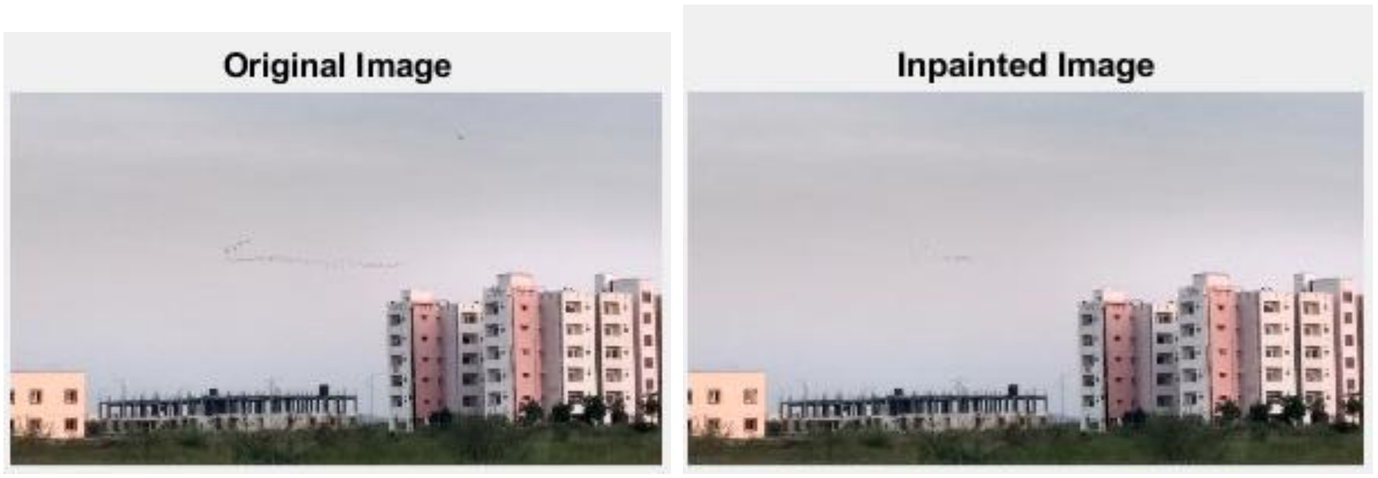
$Fig\ 4.1.1: Restoration\ of\ an\ old\ image\ of\ Lincoln$



$Fig\ 4.1.2: Removal\ of\ the\ microphone\ cord$

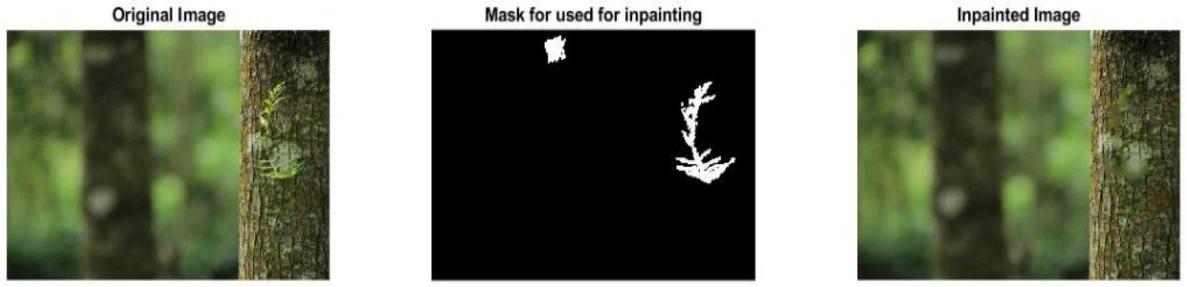


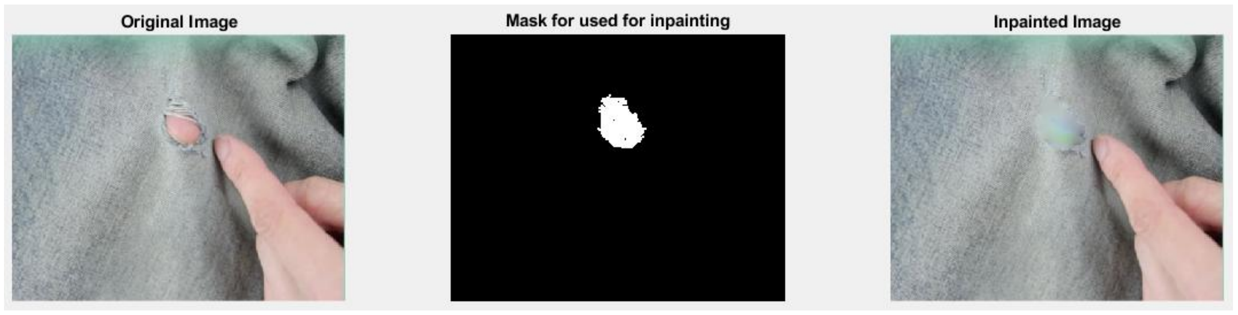
$Fig\ 4.1.3: Removal\ of\ text\ from\ image$



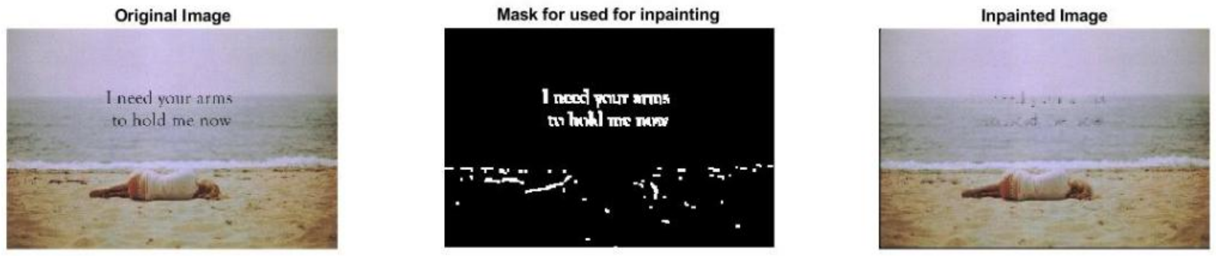
$Fig\ 4.1.4$: $Birds\ removed\ from\ the\ picture$

4.2 FAILURES





$Fig\ 4.2.1$: $Texture\ not\ properly\ inpainted\ (Top\ and\ above\ images)$



$Fig\ 4.1.4$: $Error\ due\ to\ incorrect\ generation\ of\ mask$

5. DISCUSSIONS

5.1 CONCLUSIONS

This algorithm performs quite well under ideal conditions that are:
1. region to be inpainted is smooth.
2. There are no textures that need to be filled in.
3. There is enough information at the boundary to be used.
4. The mask is accurate.

Challenges:
1. The time increases non-linearly with respect to the maximum width of the inpainting mask.
2. Non-smooth textures or noisy or any other high frequency changes in the region causes issues.
3. Mask extraction is generally inaccurate.

5.2 FUTURE DIRECTIONS

One could work on the following to improve the inpainting technique mentioned here:
1) This method can be combined with weighted Fourier Transforms to fill in surrounding textures.
2) This algorithm can be experimented with higher order information metrics.

6. TASK ASSIGNMENT AND GITHUB LINK

6.1 Task division

The whole project had following tasks to do:

1) Text detection algorithm (SOWMYA AITHA)

2) Information and Isophote directions (SHOEB SIDDIQUI)

3) Anisotropic diffusion and UI for project (TANU SHARMA)

6.2 Github Link

https://github.com/unatsharma/DipProject.git

7. REFERENCES

[1] "Image Inpainting" by Marcelo Bertalmio and Guillermo Sapiro

[2] "Scale Space and Edge Detection Using Anisotropic Diffusion" by Pierto Perona and Jitendra Malik

[3] "Edge Based Method for Text detection from Complex Document Images" by Matti Pietikainen and Oleg Okun

[4] Wikipedia

[5] Some images obtained from internet