**19AIE303 - Signal and Image Processing**

**Total variation-based Image Inpainting**

**End Semester Project Report**

*Submitted by*

**Shiva Rupan S** (CB.EN.U4AIE19059)

**Vijai Simmon S** (CB.EN.U4AIE19068)

**Amrita Vishwa Vidyapeetham, Ettimadai**

**Coimbatore – 641112**



**December, 2021**

**Acknowledgements**

We would like to extend our gratitude to our project guide, *Dr. V. Sowmya* for her guidance, invaluable support and dedication with which she helped us complete this project.

We also wish to thank the Centre for Excellence in Computational Engineering and Networking (CEN) department for providing us with this course thereby helping us to understand and implement studies involving Signals and Images.

We also wish to show our gratitude to our Chancellor *Mata Amritanandamayi* with whose benevolence we were able to complete this project even in such trying times of the Covid-19 pandemic.

**Abstract**

Image inpainting is the method of repairing the old and deteriorated images. It has been used a lot even before the advent of computers. However, it has gained momentum in recent owing to developments in the image processing domain. With the improvement of image processing tools and the flexibility of digital image editing, automatic image inpainting has found important applications in computer vision and has also become an important and challenging topic of research in image processing.

Our project implements two methods for image inpainting, one based on total variations and the based on Harmonic variations. The project begins with an introduction to the mathematical concept of functionals, that form the basis of optimization-based inpainting. We then derive the Euler-Lagrange equation after understanding its relation with the stationary points of functionals. Next, we proceed to formulate inpainting as a Minimization problem. The ensuing sections show the implementation of both methods in MATLAB. Finally, the report ends with a summarization of both the approaches with their pros and cons.

**Table of Contents**

**Introduction ….............................................................................................. 6**

**Functionals …............................................................................................... 7**

**The Euler-Lagrange equation …..................................................................... 9**

**Total Variation (TV) Inpainting ..................................................................... 13**

**Implementing TV Inpainting in MATLAB ............................................................ 16**

**TV Inpainting Output …................................................................................. 18**

**Implementing Harmonic Inpainting in MATLAB ............................................... 20**

**Harmonic Inpainting Output …...................................................................... 24**

**Conclusion …............................................................................................... 26**

**Future work …............................................................................................. 26**

**References .................................................................................................. 26**

**Introduction**

Image inpainting is a technique that can be used to restore missing or damaged pixels in images. Owing to its high practical value, image inpainting has been a research field for many years. For image inpainting, the Total Variation (TV) model is always a powerful and popular tool. However, when TV norm is involved, most of the conventional image inpainting methods suffer from difficulty in the numerical solution.

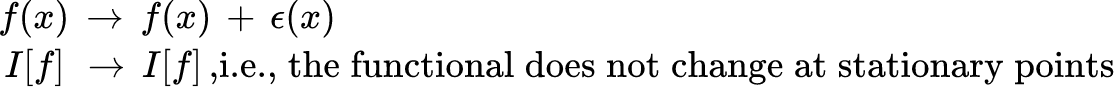
The proposed method first converts the classical TV model into an equivalent unconstrained minimization problem. The minimization problem is decomposed into several subproblems with a smaller size. In an iterative process, the optimal solution of the original problem can be efficiently obtained.

**Functionals**

The concept of functionals is key to understand how image inpainting can be formulated as an optimization problem. It is well known that real-valued functions take one number or a vector of numbers as input. For every small change dx in the input vector ‘x’, the corresponding function ‘f’ will change by df. Specifically, at points of extrema (a.k.a stationary points), there will be no change in function value for a small change in ‘x’. In, other words, df = 0 for a change dx.

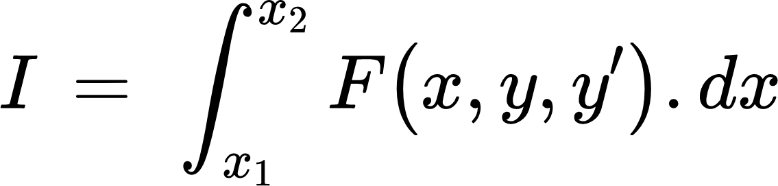


Akin to how functions take numbers as input, functionals take other functions as input. A given functional can also have stationary points while taking certain functions as input. In the below section, condition for the stationary point of a functional is derived:

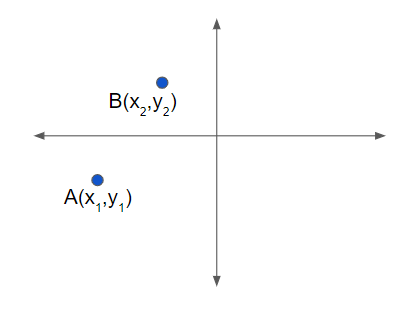


**If ‘I’ is a functional, then there can exist a function ‘f’ such that I[f] is an extrema value, i.e., if the input changes by a small amount ε(x), then there will not be any change in the function ‘I’**

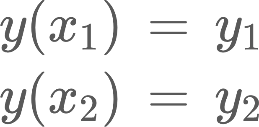
Say that the function that minimises our functional is given by y = f(x) and that our functional ‘I’ is given by the integral (y′ is yet another function):



As can be seen, our functional ‘I’ takes as input the functions y and y′. It is to be noted that while ‘I’ may have multiple stationary points, the ones we require should also pass through the existing data points we have. For sake of simplicity, assume that there are only 2 data points: A(x1, y1) and B(x2, y2).

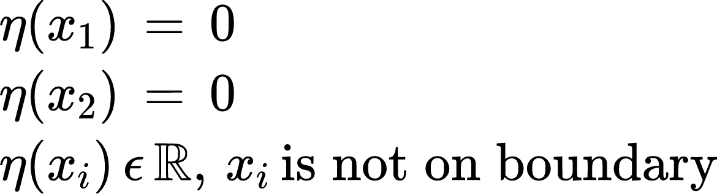


In other words, we have to ensure that the boundary conditions are ***satisfied:***



**The Euler-Lagrange Equation**

Before proceeding to find the required function ‘y’, let us define another function ‘η’ that is zero at X-coordinates of the existing data (or boundary) points, and some real value otherwise:



**Note:** The 2nd derivatives of the various functions involved (y, F, η, etc.) are ***assumed*** to not have any discontinuities.

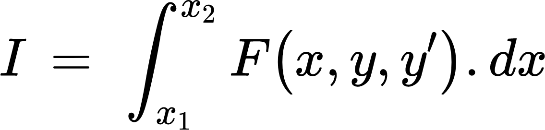
Let us define one last function ‘Ӯ’ to represent change in the input ‘y’ (similar to x → x + dx). Here, ‘ε’ is similar to ‘dx’

{"code":"$$\\overline{y}\\left(x\\right)\\,=\\,y\\left(x\\right)\\,+\\,\\epsilon\\eta\\left(x\\right)$$","font":{"size":14,"family":"Arial","color":"#000000"},"backgroundColorModified":null,"aid":null,"id":"15","type":"$$","backgroundColor":"#FFFFFF","ts":1638511347866,"cs":"e4L7XUh37qcByiFAiQzpNw==","size":{"width":198.5,"height":22}}

Since ‘y’ satisfies the boundary conditions and since ‘η’ is zero at the boundary, it can be seen that ‘Ӯ’ will satisfy the boundary conditions.

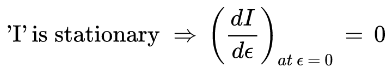
Also, ‘Ӯ’ represents a family of curves. One curve is different from the other depending on the values taken by ‘η’ and ‘ε’. We need to find specific curve(s) from this family that acts as the stationary point for the functional ‘I’.

‘y’ and ‘Ӯ’ are functions of ‘x’. After integration, all terms involving ‘x’ will be replaced by the limits x1 and x2. Finally, the only unknown will be ‘ε’. In other words, ‘I’ will become a function of ‘ε’. Therefore, in order to find the extremum point of ‘I’, we can differentiate ‘I’ w.r.t ‘ε’ and equate to zero (like in the single variable case). .......................... (1)

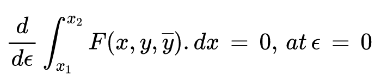


Recall our assumption that ‘y’ is the function that will make ‘I’ stationary. So, to retrieve ‘y’ from the family of curves given by ‘Ӯ’, the value of ‘ε’ has to be zero too ................... (2)

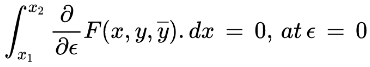
From (1) and (2):



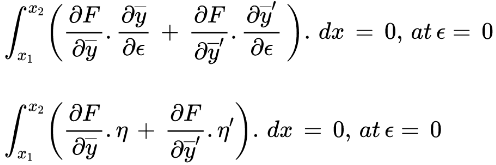
Substituting for ‘I’:



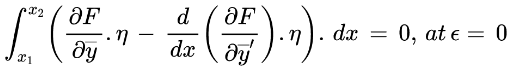
Moving the derivative inside would make it a partial derivative:



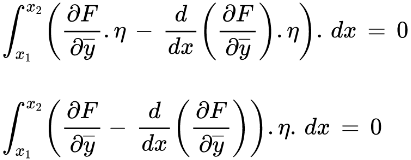
Using the chain rule:



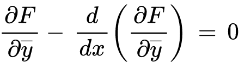
Integrating the 2nd term of the above integral by parts, it can be arrived that:



When ε = 0, Ӯ = y:

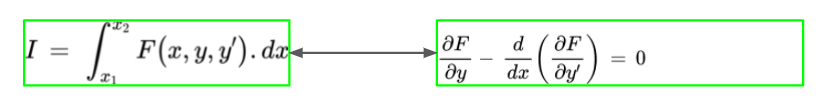


For the above product inside the integral to be zero, (‘η’ cannot be zero according to its definition):



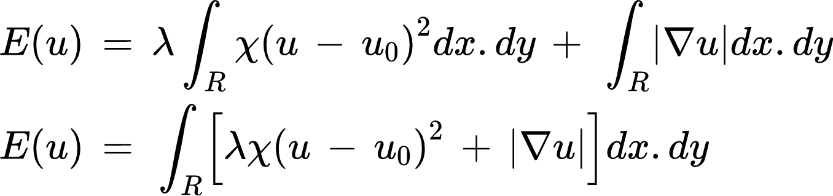
This is the Euler-Lagrange equation.

***Thus, if y = f(x) is a function that makes the aforementioned functional ‘I’ stationary, then the Euler-Lagrange equation has to be satisfied.***

............ (3)

**Total Variation Inpainting**

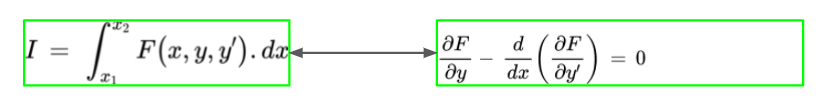
For TV Inpainting, the functional to be minimised is:



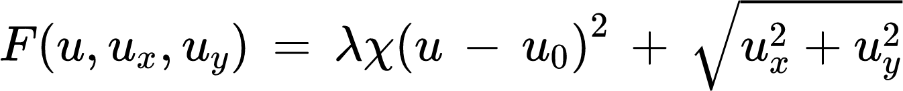
Additional parameter to be defined:



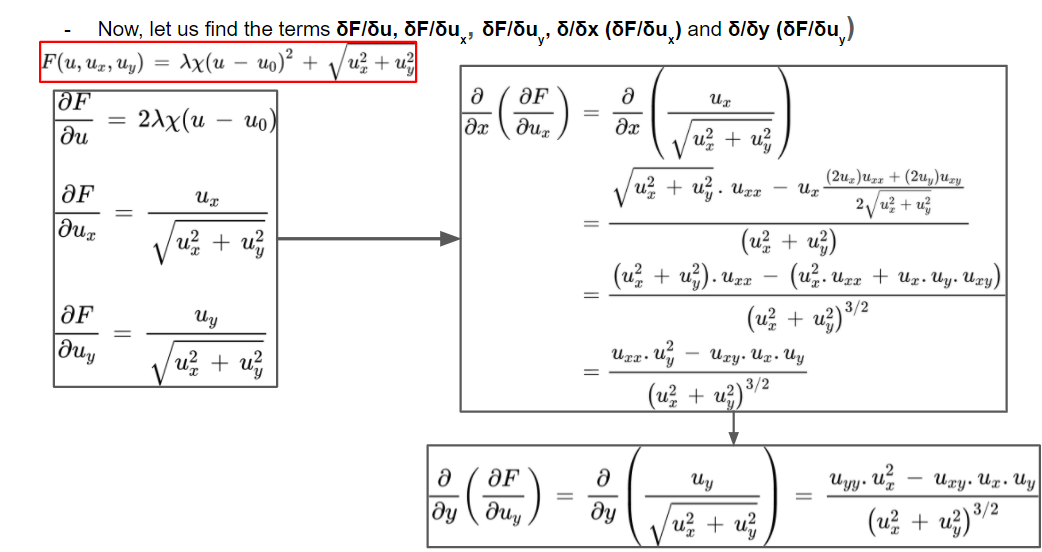
In the previous sections, we have seen that for a given functional ‘I’ in terms of the function ‘F’, the condition for extrema is given by the Euler-Lagrange equation:

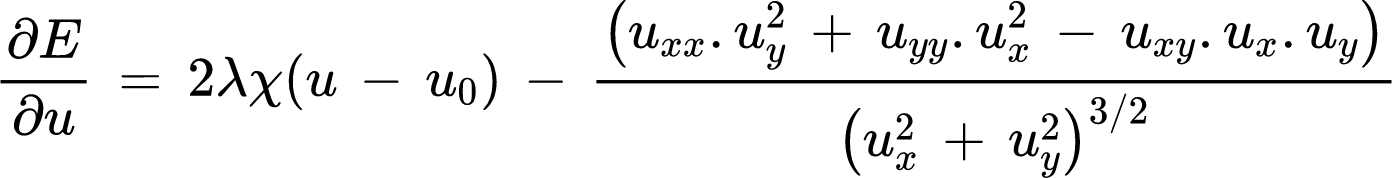


Comparing with (3), ‘E’ corresponds to ‘I’ and ‘F’ is:

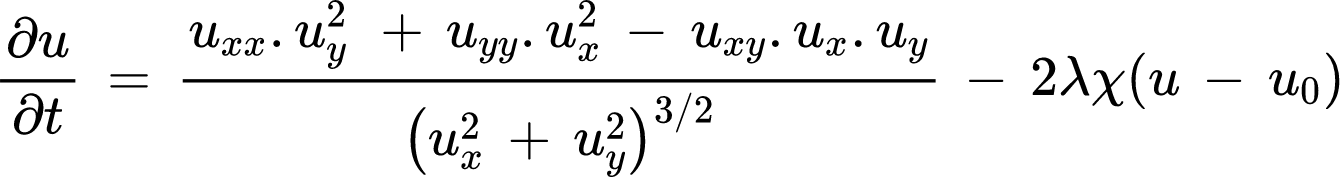


Proceeding to calculate the various derivatives involved in the Euler-Lagrange equation:

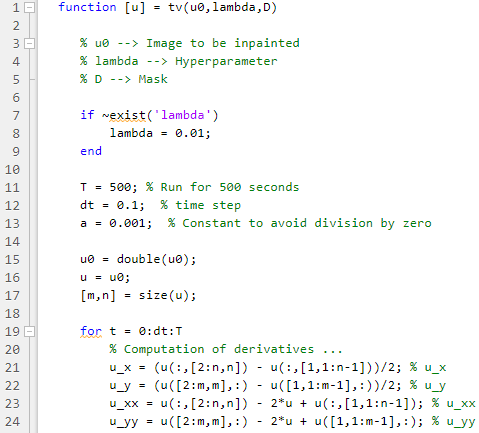


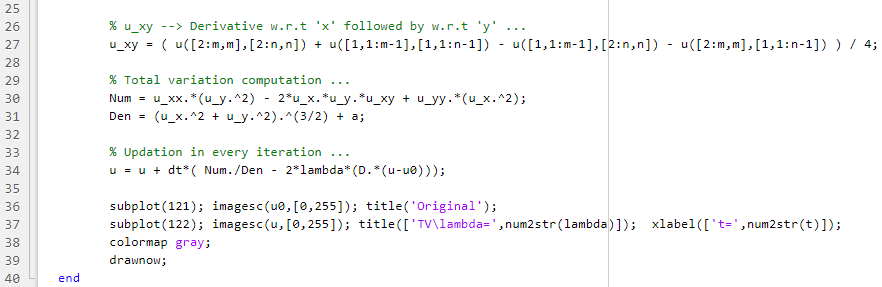


Using gradient descent, it can be arrived that:

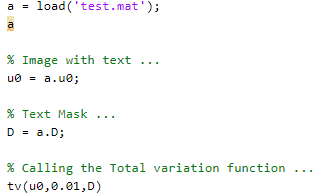


**Implementing TV Inpainting in MATLAB:**

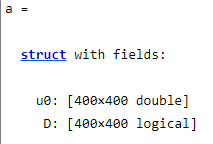




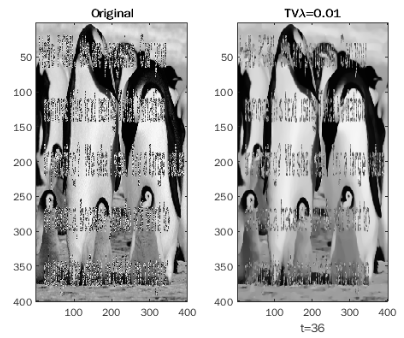
**Main function:**



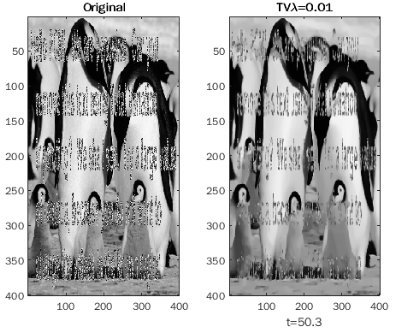
**Output**



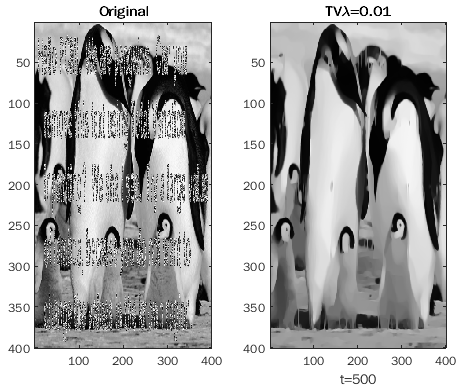
**At t = 36 seconds**



**At t = 50.3 seconds**

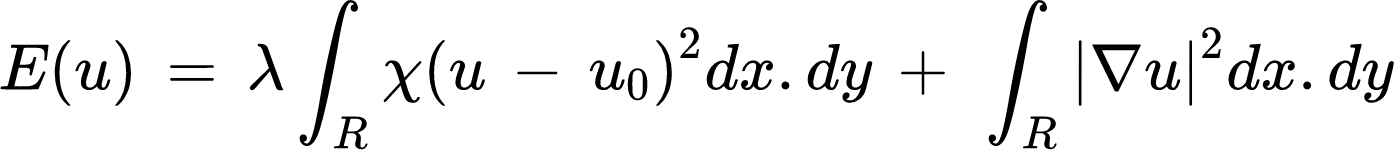


**At t = 500 seconds**

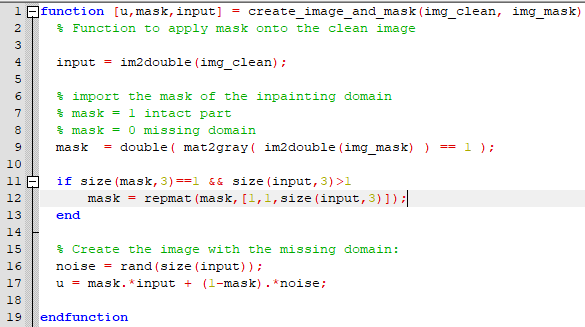


**Implementing Harmonic Inpainting in MATLAB:**

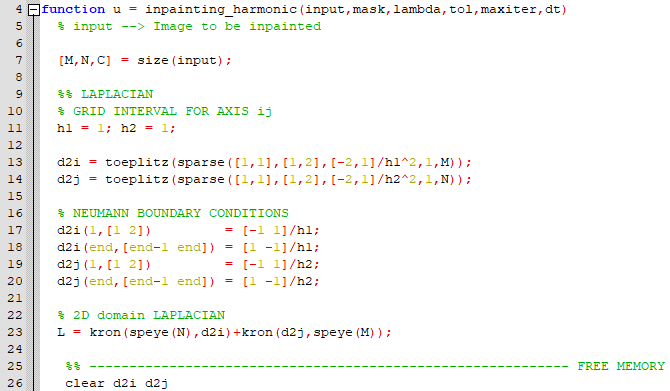
Harmonic Inpainting is similar to TV inpainting except that the second term involves a square

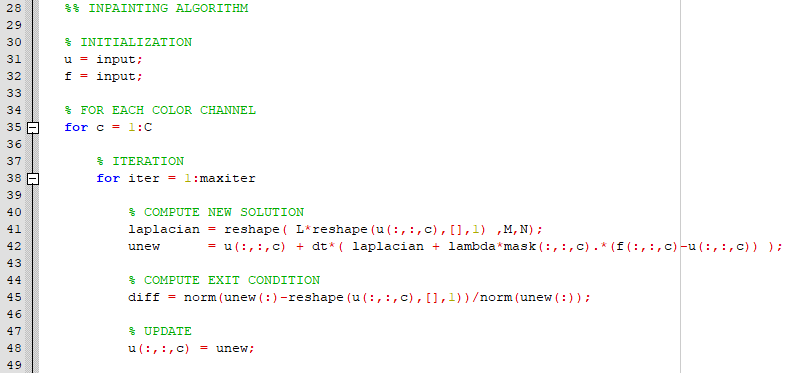


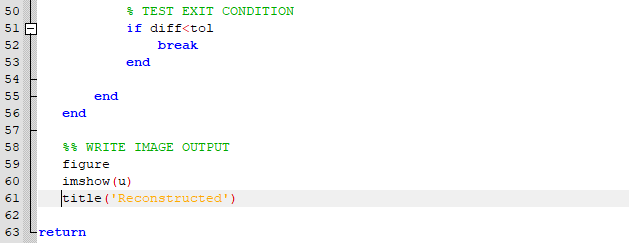
**Creat\_image\_and\_mask.m**



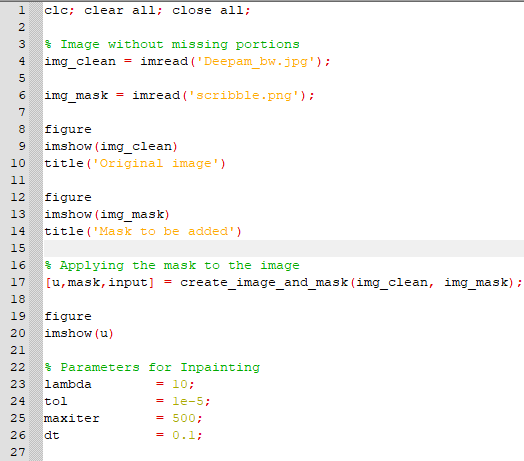
**inpainting\_harmonic.m** - Function to perform the inpainting

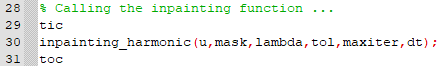






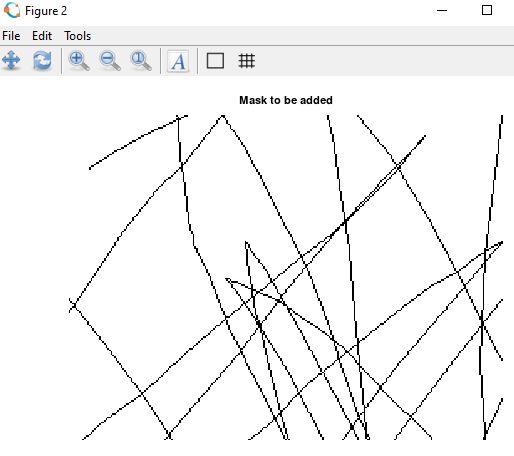
**Main function:**



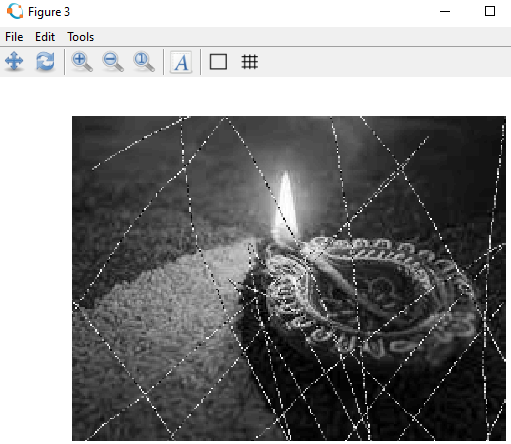


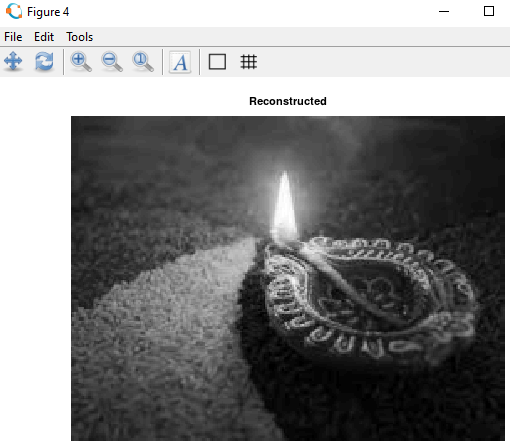
**Output figures:**





**Corrupted Image:**





**Conclusion**

Thus, we have implemented two methods for Image inpainting, one based on Total variation and the other based on Harmonic functions. While the Harmonic inpainting only takes a few seconds to complete, the total variation-based method is computationally very expensive and time consuming. Even after performing inpainting for 500 seconds, there are slight *smudges* in the inpainted areas.

Harmonic inpainting performs well over removing scratches. The quickness of the algorithm might be attributed to the Sparse matrix approach used in the code.

**Future work**

In TV inpainting, the hyperparameter *‘lambda’* can be varied to see changes in the inpainted image. Metrics like the *signal-to-noise* ratio can be used to see changes happening with varying *‘lambda’* values.

**References**

1. Digital Signal and Image Processing: The Sparse Way, K.P. Soman, R. Ramanathan

2. <https://github.com/simoneparisotto/MATLAB-Python-inpainting-codes>

**============================================================================================================================================**