WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

PDE Based Image Inpainting

Sri Harini S, Sriniriksha S, Sureka S, Suruthika S, Thanuja Sibaani U S, Dr. V. Kamalaveni*

Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, India.
Email: sriharinis24ad@srishakthi.ac.in
Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, India.
Email: sriniriksha s24ad@srishakthi.ac.in
Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, India.
Email: surekas24ad@srishakthi.ac.in
Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, India.
Email: suruthikas24ad@srishakthi.ac.in
Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, India.
Email: thanujasibaanius24ad@srishakthi.ac.in
* Mentor, Professor, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, India

ABSTRACT –

PDE-Based Image Inpainting for Damaged Images is an image restoration system designed to reconstruct missing or damaged regions in digital images using partial differential equations (PDEs). The system focuses on restoring image continuity by analysing pixel intensities in the surrounding undamaged regions. It employs the Laplace equation to estimate missing pixel values through an iterative numerical process, enabling the reconstruction of damaged areas while preserving the known portions of the input image.

The proposed system provides visual comparisons of the original, damaged, masked and restored images to demonstrate the reconstruction process. It supports different damage patterns, including scratches, text overlays, blocks and rectangular missing regions. The implementation incorporates numerical solvers such as the Jacobi, Gauss–Seidel and Successive Over-Relaxation (SOR) methods to solve the discretised Laplace equation. Restoration performance can be evaluated using metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), allowing the reconstructed image quality and the effects of different damage patterns to be analysed.

Developed using Python, the system utilises numerical computation and image-processing libraries to implement the inpainting algorithm, process damage masks and generate restoration outputs. The approach offers a mathematically interpretable and computationally accessible solution for image restoration. Experimental observations demonstrate that the method can effectively reconstruct small damaged regions, while larger missing areas may result in smoothing and loss of fine image details. By combining PDE-based mathematical modelling, iterative numerical methods and quantitative evaluation, the proposed system provides a practical framework for studying digital image inpainting and restoration.

Keywords - Image Inpainting, Partial Differential Equations (PDEs), Laplace Equation, Image Restoration, Jacobi Method, Gauss–Seidel Method, Successive Over-Relaxation (SOR), Image Processing, PSNR, SSIM

I. INTRODUCTION

PDE-Based Image Inpainting for Damaged Images is an image restoration approach designed to reconstruct missing or damaged regions in digital images using Partial Differential Equations (PDEs). The system focuses on restoring image continuity by estimating pixel values in damaged areas based on information from surrounding undamaged regions. The primary objective of this project is to reduce the visual impact of image damage and recover missing information through a mathematical and computational approach.

Digital images may become damaged due to scratches, missing pixels, unwanted text, stains, or other forms of degradation. Such damage can affect image quality and make important visual information difficult to interpret. Conventional manual restoration can be time-consuming, particularly when damaged regions require detailed reconstruction. These challenges highlight the need for an efficient image restoration technique that can automatically estimate missing pixel values while preserving the available image information. PDE-based inpainting addresses this problem by using mathematical models to propagate image information into damaged regions.

The proposed system uses the Laplace equation to guide the reconstruction process within a predefined damage mask. The input image is processed to identify damaged regions, while the surrounding undamaged pixels provide boundary information for restoration. An iterative numerical solver estimates the missing pixel values until a specified convergence condition is reached. The system generates visual outputs showing the original image, damaged image, damage mask, and restored image, enabling users to examine the effectiveness of the reconstruction process.

The project is implemented using Python and image-processing libraries, providing a practical framework for experimenting with PDE-based image restoration. Different damage patterns can be introduced to evaluate the behaviour of the algorithm under varying conditions. Quantitative measures such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) can also be used to assess restoration quality when the original undamaged image is available for comparison. By combining mathematical modelling, numerical computation, and visual evaluation, the proposed system demonstrates the application of partial differential equations in digital image inpainting while highlighting the limitations of reconstructing large or complex damaged regions.

II. LITERATURE SURVEY

A. Partial Differential Equation-Based Image Inpainting

Image inpainting has become an important area of research in image processing and computer vision due to its applications in restoring damaged photographs, removing unwanted objects, and reconstructing missing image regions. Early image restoration techniques relied on manual editing and basic interpolation methods, which often produced unsatisfactory results when dealing with complex damage patterns. Research in mathematical image inpainting introduced techniques that use information from surrounding undamaged regions to estimate missing pixel values. Partial Differential Equations (PDEs) have gained attention in this field because they provide a mathematical framework for propagating image information into damaged areas while maintaining local continuity. These methods are particularly suitable for restoring small scratches, narrow gaps, and limited missing regions, although their effectiveness may decrease when large or structurally complex areas are damaged.

B. Numerical Methods for PDE-Based Image Restoration

Numerical methods play an important role in solving Partial Differential Equations for image restoration. Since analytical solutions are difficult to obtain for complex image regions, iterative techniques are commonly used to solve the discretised equations. Methods such as Jacobi, Gauss–Seidel, and Successive Over-Relaxation (SOR) estimate unknown pixel values by repeatedly updating them using neighbouring pixel information. These methods differ in their convergence behaviour and computational efficiency. The Jacobi method calculates new values using the previous iteration, whereas the Gauss–Seidel method uses updated values immediately during each iteration. The SOR method introduces a relaxation factor to accelerate convergence under suitable conditions. Applying these numerical techniques to the Laplace equation provides a systematic approach to reconstructing missing image regions and comparing restoration performance

C. Image Restoration and Performance Evaluation

Image restoration techniques aim to recover image quality by estimating missing or corrupted pixel information. The effectiveness of an inpainting method depends on the size, shape, and location of the damaged region, as well as the image's texture and structural complexity. PDE-based methods are useful for maintaining smooth transitions between damaged and undamaged areas by propagating information from neighbouring pixels. However, these methods may not accurately reconstruct complex textures or large missing objects because the Laplace equation primarily promotes smoothness rather than generating new image details. Performance evaluation is therefore important for understanding the strengths and limitations of the restoration process. Metrics such as Peak Signal-to-Noise Ratio (PSNR) measure pixel-level reconstruction quality, while Structural Similarity Index Measure (SSIM) evaluates similarities in image structure, luminance, and contrast. Together with visual comparisons, these measures help assess the effectiveness of PDE-based image inpainting under different damage conditions.

D. Applications of PDE-Based Image Inpainting

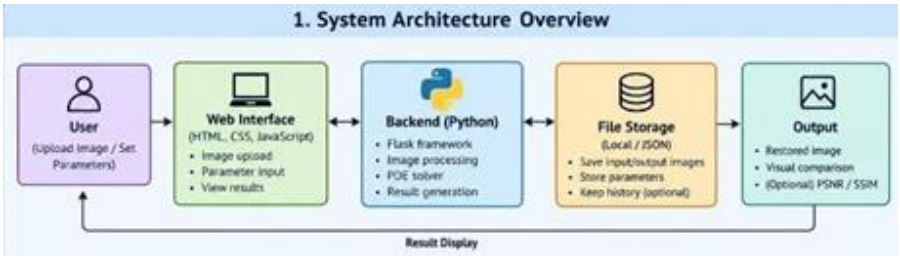
Recent developments in image processing have highlighted the importance of image inpainting in applications such as damaged photograph restoration, removal of unwanted objects, reconstruction of missing image regions, and digital image enhancement. PDE-based inpainting techniques provide a mathematical approach to these tasks by using information from surrounding pixels to estimate missing values. The Laplace equation is particularly useful for propagating intensity information smoothly into damaged regions, making it suitable for restoring small scratches, narrow gaps, and localized image defects. Numerical methods can be employed to solve the resulting equations iteratively and improve computational efficiency. Although PDE-based techniques are effective for maintaining local image continuity, their performance may be limited when reconstructing large damaged areas or complex textures. These applications and limitations motivate the development of a practical PDE-based image inpainting system for evaluating restoration quality across different damage patterns.

III. PROPOSED METHODOLOGY

A. System Architecture Overview

The proposed PDE-Based Image Inpainting system is designed as a modular image-processing application to ensure simplicity, computational efficiency, and ease of implementation. The system consists of four primary components: input image processing, damage mask generation, PDE-based restoration,

and result visualization and evaluation. Each component performs a specific role in preparing the image, identifying damaged regions, reconstructing missing pixel values, and presenting the restoration results. The architecture follows a systematic workflow in which the input image is first loaded and a damage mask is created to identify the regions requiring restoration. The Laplace equation is then solved iteratively using a numerical method to estimate pixel values within the damaged areas. Finally, the restored image is displayed alongside the original and damaged images for visual comparison. Where the original image is available, quantitative metrics such as PSNR and SSIM can be used to evaluate restoration quality. This modular structure provides a practical framework for implementing and analysing PDE-based image inpainting.

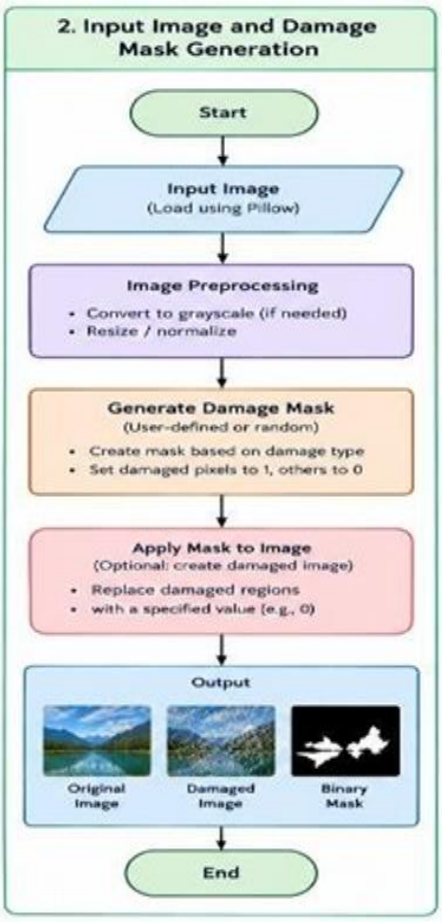


B. Input Image and Damage Mask Generation

The input image component allows the user to load or generate an image for restoration. The system prepares the image for processing and creates a damage mask that identifies the pixels requiring reconstruction. Python-based image-processing libraries are used to represent the image as a numerical array and manipulate the damaged regions

1. Image Input and Preparation: The system accepts an image as input and prepares it for the inpainting process. The image is represented using pixel intensity values, with colour images containing separate red, green, and blue channels. The image dimensions and pixel representation are maintained consistently throughout the restoration workflow.

2. Damage Mask Generation: A binary mask is created to distinguish damaged pixels from undamaged pixels. The mask identifies the regions to be restored, such as scratches, rectangular blocks, text-like marks, or other missing areas. Damaged pixels are designated for reconstruction, while the remaining pixels are preserved as known image information. This mask guides the PDE solver by defining the region in which pixel values must be estimated.



C. PDE-Based Restoration Using Python

The restoration component is implemented in Python and performs the core computation of the proposed image inpainting system. It uses the Laplace equation to estimate pixel values in damaged regions based on information from neighbouring undamaged pixels. The numerical solution is obtained through an iterative process that updates the unknown pixel values until a specified convergence criterion or iteration limit is reached.

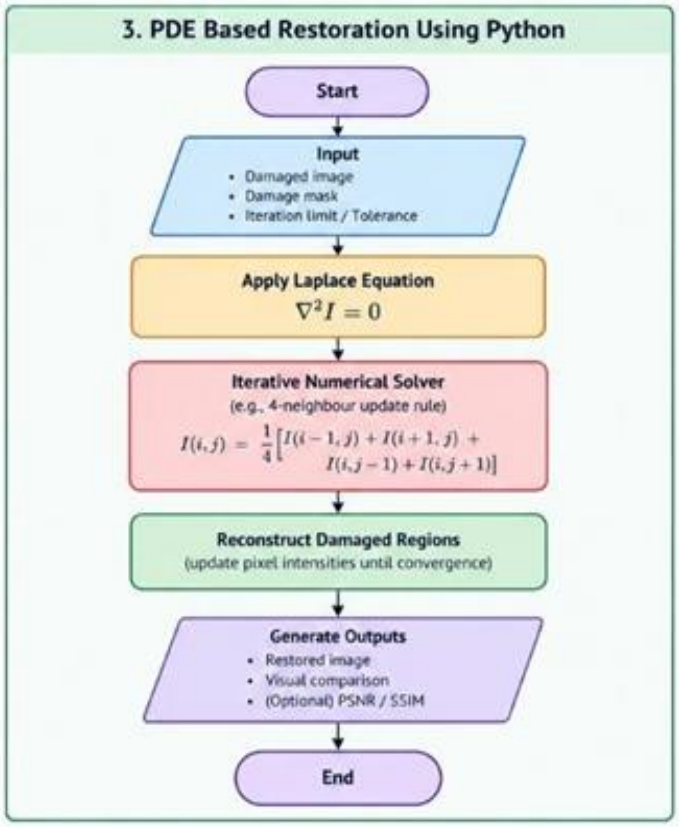
1. **Laplace Equation Formulation:** The Laplace equation is used to model the smooth propagation of image intensity into missing regions. For a grayscale image, the equation is expressed as:

$$\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = 0$$

where (I) represents the image intensity function. This formulation assumes that the intensity within the damaged region varies smoothly and is influenced by the surrounding known pixels.

2. **Numerical Solution:** The continuous equation is discretised using neighbouring pixel values, and an iterative numerical solver estimates the unknown intensities. Depending on the implementation, methods such as Jacobi, Gauss–Seidel, or Successive Over-Relaxation (SOR) can be used. The process continues until the solution satisfies the selected stopping criterion or reaches the maximum number of iterations.

3. **Image Reconstruction:** After the iterative computation, the estimated pixel values are inserted into the damaged regions to produce the restored image. The original undamaged pixels are retained, allowing the output to preserve the available image information while reconstructing the missing areas. The resulting image is then prepared for visualisation and performance evaluation.



D. Image Output and Result Management

The system manages the intermediate and final outputs generated during the image restoration process. These outputs allow the restoration procedure to be inspected and the effectiveness of the inpainting algorithm to be evaluated.

1. **Result Storage:** The system stores or displays the generated results, including the original image, damaged image, damage mask, and restored image. These outputs provide a clear visual representation of the different stages of the restoration process.

2. **Result Visualization and Retrieval:** The generated images can be accessed from the designated output directory for further analysis and documentation. The results can be used to compare the damaged and restored images and examine the performance of the PDE-based restoration method.

E. PDE-Based Inpainting Model Design

The proposed inpainting model uses the Laplace equation to reconstruct missing or damaged image regions. The model estimates unknown pixel intensities by using information from neighbouring pixels and applying an iterative numerical solution.

1. Laplace Equation Formulation:

The governing equation is expressed as:

$\nabla^2 I = 0$

where (I) represents image intensity, and the equation describes smooth intensity variation within the damaged region.

2. Iterative Pixel Estimation:

For a two-dimensional image, the discretised Laplace equation estimates the intensity of an unknown pixel using its four neighbouring pixels. The update rule is expressed as:

$I(i,j) = [I(i-1,j) + I(i+1,j) + I(i,j-1) + I(i,j+1)] / 4$

This equation calculates the average of the neighbouring pixel intensities to estimate the missing value. The iterative process continues until the specified convergence condition is satisfied or the maximum iteration limit is reached.

3. Damage Mask-Based Reconstruction:

The binary damage mask identifies the pixels that require restoration. The PDE solver updates the pixel values inside the masked region while preserving known image information outside it. Once the iterative process is completed, the estimated values are combined with the undamaged pixels to generate the final restored image.

IV. SYSTEM IMPLEMENTATION

A. Environment Setup

The proposed PDE-Based Image Inpainting system is developed using Python within the Visual Studio Code (VS Code) environment. Python is selected because of its suitability for numerical computation, image processing, and implementation of iterative mathematical algorithms. The system applies the Laplace equation to reconstruct damaged image regions through an iterative numerical procedure.

The development environment and technologies used in the implementation are as follows:

- Programming Language: Python 3.x
- Development Tool: Visual Studio Code
- Numerical Computation: NumPy
- Image Processing: Pillow (PIL)
- Visualization: Matplotlib
- Restoration Technique: Partial Differential Equation (PDE)-based image inpainting using the Laplace equation

The project is organized into Python source files and an output directory for storing generated restoration results. The main program manages the image preparation, damage mask generation, PDE-based restoration, and visualization of the final output. This structure simplifies execution, testing, and modification of the implementation. The modular workflow also allows different damage patterns and numerical configurations to be evaluated within the same development environment.

A. Input Image and Damage Mask Generation Module

The input module prepares the image for restoration and identifies the regions that require reconstruction. The system uses an input image and a corresponding binary damage mask to distinguish damaged pixels from undamaged regions. This module establishes the input data required for the PDE-based restoration process.

• Image Input: The system loads or generates an image for processing. The image is represented as a numerical array containing pixel intensity values.

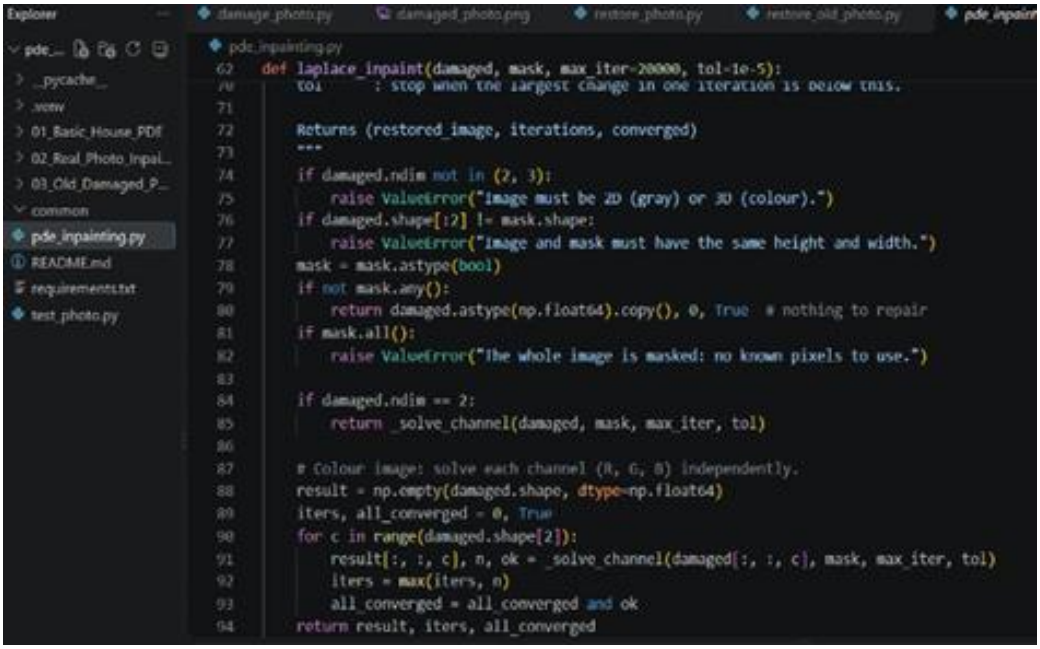
• Image Preparation: The input image is prepared in the required dimensions and format to ensure compatibility with the restoration algorithm.

• Damage Generation: Artificial damage patterns, such as scratches, rectangular regions, blocks, or text-like marks, can be introduced to simulate image degradation.

• Mask Generation: A binary mask identifies the damaged pixels that require reconstruction. The mask guides the restoration algorithm by defining the region in which pixel values must be estimated.

• Input Validation: The system checks that the image and its corresponding mask have compatible dimensions and valid representations before starting the restoration process.

The output of this module consists of the prepared input image, the damaged image, and the corresponding damage mask. These are passed to the PDE-based restoration module for further processing.



B. PDE-Based Restoration and Numerical Processing

The core functionality of the proposed system lies in its PDE-based image restoration module, which reconstructs damaged image regions using the Laplace equation and iterative numerical computation.

Image Processing:

- Loads the prepared image and its corresponding damage mask.
- Identifies the damaged and undamaged pixel regions.
- Preserves the known image information during reconstruction.

PDE-Based Restoration Algorithm:

- Applies the Laplace equation to estimate missing pixel intensities.
- Uses neighbouring pixel values to calculate updated intensities within the damaged region.
- Repeats the numerical updates until the convergence condition is satisfied or the iteration limit is reached.

Numerical Computation:

- Discretises the Laplace equation for computational implementation.
- Estimates missing pixel values using an iterative numerical solver.
- Combines the reconstructed regions with the original undamaged pixels to generate the restored image.

Output Generation:

- Produces the restored image for visual inspection.
- Displays the original image, damaged image, damage mask, and restored image for comparison.
- Supports quantitative evaluation using PSNR and SSIM when the original undamaged image is available.

This approach provides a systematic method for reconstructing missing image information by propagating intensity values from neighbouring pixels. The restoration quality depends on the size and shape of the damaged region, the image structure, and the convergence behaviour of the numerical solver.



```
01_Basic_House_PDE > main.py > ...
 89  def main():
113          if not np.all(np.isfinite(restored)):
114              raise ValueError("Restored image contains NaN or infinite values.")
115
116          restored = np.clip(restored, 0.0, 1.0)
117
118          # 4. Evaluate restoration against the original image.
119          mse = np.mean((restored - original) ** 2)
120          psnr = 10 * np.log10(1.0 / mse) if mse > 0 else float("inf")
121
122          print(
123              "Mean absolute error inside hole: "
124              f"{np.mean(np.abs(restored[mask] - original[mask])):.4f}"
125          )
126          print(f"PSNR (whole image): {psnr:.2f} dB")
127          print(
128              "Pixels outside mask unchanged: "
129              f"{np.array_equal(restored[~mask], damaged[~mask])}"
130          )
131
132          # 5. Save all output images in this task's results folder.
133          RESULTS_DIR.mkdir(parents=True, exist_ok=True)
134
135          plt.imsave(RESULTS_DIR / "original.png", original)
136          plt.imsave(RESULTS_DIR / "damaged.png", damaged)
137          plt.imsave(
138              RESULTS_DIR / "mask.png",
139              mask.astype(float),
140              cmap="gray",
141              vmin=0,
142              vmax=1,
143          )
144          plt.imsave(RESULTS_DIR / "restored.png", restored)
145
146          # 6. Create and save the four-panel comparison.
147          fig, axes = plt.subplots(1, 4, figsize=(16, 4.4))
```

C. Numerical Solver Comparison Module

The system can incorporate different iterative numerical methods for solving the discretised Laplace equation and reconstructing damaged image regions.

Numerical Methods:

- Jacobi method
- Gauss–Seidel method
- Successive Over-Relaxation (SOR) method, if implemented in the selected version

Solver Processing:

- Iteratively estimates missing pixel intensities using neighbouring pixel values.
- Applies the selected stopping criterion or maximum iteration limit.
- Monitors convergence behavior where residual calculations are implemented.

Performance Comparison:

- Compares the computational behavior of the implemented solvers.
- Examines convergence speed and restoration output.
- Helps identify differences in numerical efficiency.

This module provides a way to study how different iterative methods influence the solution of the PDE-based restoration problem.



D. Image Output and Result Management

The system generates and manages intermediate and final images produced during the restoration process.

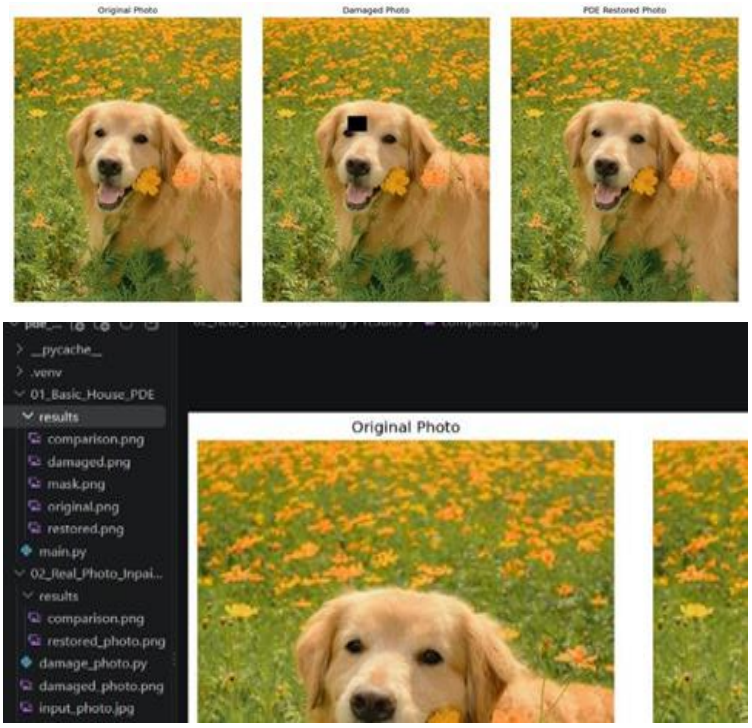
Generated Outputs:

- Original image
- Damaged image
- Binary damage mask
- Restored image

Output Management:

- Saves generated results in the designated output directory, where configured.
- Organises images for visual inspection and project documentation.
- Supports comparison between the damaged and restored images.

These outputs help demonstrate how the inpainting algorithm reconstructs damaged regions while retaining the available undamaged image information.



E. Restoration Evaluation Module

The restoration evaluation module assesses the quality of the reconstructed image using visual comparisons and quantitative image-quality metrics.

Peak Signal-to-Noise Ratio (PSNR): Measures pixel-level reconstruction quality by comparing the restored image with the original undamaged image. A higher PSNR generally indicates lower reconstruction error.

Structural Similarity Index Measure (SSIM): Evaluates similarity between images based on luminance, contrast, and structural information. Values closer to 1 generally indicate greater structural similarity.

Visual Evaluation: Examines the original, damaged, and restored images to identify remaining artefacts, smoothing, and loss of fine details.

Damage-Based Analysis: Evaluates how the size and pattern of the damaged region affect restoration quality.

Quantitative evaluation requires access to the original undamaged image as a reference. These measurements provide a basis for analysing the strengths and limitations of PDE-based image inpainting.

V. ADVANTAGES

Simple and Lightweight Implementation:

The proposed PDE-based image inpainting system is implemented using Python and commonly available image-processing and numerical computation libraries. Its straightforward architecture makes it easy to execute, understand, and modify within the Visual Studio Code environment without requiring specialized hardware.

Mathematical and Systematic Restoration:

The system uses the Laplace equation to estimate missing pixel intensities from neighbouring pixels. This mathematical approach provides a systematic method for reconstructing damaged regions and maintaining smooth transitions between restored and undamaged areas.

Support for Different Damage Patterns:

The system can be used to examine different types of image damage, including scratches, rectangular regions, blocks, and text-like marks, depending on the damage patterns implemented. This flexibility helps evaluate the behaviour of the restoration algorithm under different conditions.

Preservation of Undamaged Image Regions:

The damage mask identifies the pixels that require reconstruction, allowing the algorithm to focus on the damaged areas while retaining known image information. This reduces unnecessary modifications to unaffected regions when the implementation correctly preserves the original pixels outside the mask.

Numerical Solver Comparison:

Where multiple numerical solvers are implemented, the system allows methods such as Jacobi, Gauss–Seidel, and Successive Over-Relaxation (SOR) to be compared. This helps analyse differences in convergence behaviour, iteration requirements, and computational efficiency.

Visual and Quantitative Evaluation:

The system generates visual outputs of the original, damaged, masked, and restored images. When the original image is available, metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) can provide quantitative information about reconstruction quality.

Educational and Research Value:

The project demonstrates the practical application of partial differential equations, numerical methods, and image processing in a real computational problem. Its understandable mathematical formulation makes it useful for academic demonstrations, experimentation, and further research into image restoration techniques.

Scope for Further Development:

The modular implementation provides opportunities to extend the system with improved

boundary handling, alternative inpainting algorithms, additional image-quality metrics, and more complex test images. Such extensions could help address the limitations of Laplace-based inpainting when dealing with large damaged areas or complex textures.

VI. RESULTS AND ANALYSIS

The performance of the proposed PDE-based image inpainting system was analysed through the restoration of damaged images, visual comparison of output images, and examination of the reconstructed regions. The system uses the Laplace equation to estimate missing pixel intensities from neighbouring pixels. The restoration process was implemented using Python and executed in the Visual Studio Code environment.

Restoration Quality and Output Analysis:

The system generates the original image, damaged image, damage mask, and restored image for comparison. The damaged image contains regions that are intentionally removed or modified, while the binary mask identifies the pixels that require reconstruction. During restoration, the PDE-based method estimates the missing pixel values using information from neighbouring pixels. The resulting image demonstrates how the algorithm can reconstruct damaged regions and produce smoother transitions between restored and undamaged areas.

Impact of Different Damage Patterns:

The restoration performance depends on the size, shape, and distribution of the damaged regions. Small scratches and limited damaged areas are generally more suitable for Laplace-based inpainting because the missing pixel values can be estimated from nearby information. Rectangular blocks and larger damaged regions may be more challenging because less surrounding information is available to guide the reconstruction. Testing different damage patterns helps analyse the strengths and limitations of the proposed method.

Evaluation Using Different Input Images:

The system can be tested using different input images to examine its behaviour under varying image structures and damage conditions. The restoration process follows the same general workflow: image preparation, damage-mask generation, numerical solution of the Laplace equation, and output visualization. Since the algorithm uses local neighbouring pixel information, its performance may vary depending on the image content and the extent of the damaged region. Repeated executions with consistent settings can help verify the stability of the implementation.

Performance Interpretation:

The results demonstrate the applicability of PDE-based image inpainting for reconstructing small damaged regions while maintaining smooth intensity transitions. The Laplace equation provides a mathematically defined approach to estimating missing pixel values. However, the method primarily promotes smoothness and may not recover complex textures, sharp edges, or detailed structures that are absent from the damaged region. Consequently, the restoration quality depends on the image characteristics, damage pattern, and numerical solver used.

Output Visualization and Restoration Assessment:

The system presents the original image, damaged image, damage mask, and restored image to support visual assessment of the restoration process. Comparing these outputs helps identify the reconstructed regions and observe whether the restored areas blend naturally with the surrounding pixels. If the implementation includes quantitative evaluation, metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) can be calculated by comparing the restored image with the original undamaged image. These metrics provide additional information about reconstruction quality when a reference image is available.

The results support the use of the proposed PDE-based approach as a lightweight image-restoration method for academic experimentation and small-scale inpainting tasks. Its mathematical foundation, straightforward implementation, and visual output make it suitable for demonstrating the application of partial differential equations in image processing. Nevertheless, further improvements may be required to restore large missing regions and images containing complex textures or intricate structures.

VII. FUTURE WORK

Future versions of the proposed PDE-based image inpainting system can incorporate the following improvements:

- **Advanced Inpainting Algorithms:**

The system can be extended by integrating advanced image inpainting techniques, including diffusion-based methods, exemplar-based inpainting, and deep learning approaches. These methods may improve restoration quality in images containing complex textures, sharp edges, and large missing regions.

- **Improved Numerical Solvers:**

The implementation can be enhanced by optimising numerical methods such as Jacobi, Gauss–Seidel, and Successive Over-Relaxation (SOR), where applicable. Improved convergence criteria, boundary handling, and efficient iterative computation may reduce execution time and improve restoration performance.

- **Support for Real-World Damaged Images:**

Future development can focus on processing realistic damaged photographs containing scratches, stains, cracks, missing patches, and other forms of image degradation. Testing with diverse images will help evaluate the system's practical applicability beyond artificially generated damage patterns.

- **Automatic Damage Detection:**

An automatic damage-detection module can be developed to identify damaged regions and generate binary masks without requiring manual mask creation. Image-processing techniques or machine learning models could be incorporated to improve the identification of scratches, missing areas, and other defects.

- **Enhanced Image Quality Evaluation:**

The evaluation module can be expanded by incorporating additional image-quality metrics, including Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM). Comparing the proposed PDE-based method with other inpainting algorithms would provide a more comprehensive assessment of restoration quality.

- **Graphical User Interface Development:**

A user-friendly graphical interface can be developed to allow users to upload images, select damaged regions, configure restoration parameters, and view the restored output. This would make the system easier to use without requiring direct interaction with the Python source code.

- **Processing of High-Resolution Images:**

The system can be optimised to handle high-resolution images and larger damaged regions efficiently. Techniques such as image tiling, improved numerical computation, and parallel processing may help reduce computational costs while maintaining restoration quality.

- **Integration with Deep Learning Techniques:**

The PDE-based method can be combined with deep learning models to take advantage of both mathematical smoothness and learned image features. Such a hybrid approach may improve the reconstruction of complex structures and textures while retaining the interpretability of the PDE-based formulation.

These future enhancements could improve the accuracy, efficiency, usability, and practical applicability of the proposed image inpainting system. They also provide opportunities to extend the project towards more advanced image-restoration applications.

VIII. CONCLUSION

The proposed PDE-based image inpainting system demonstrates the application of partial differential equations in restoring damaged images. By employing the Laplace equation, the system estimates missing pixel intensities using information from neighbouring pixels to reconstruct damaged regions. The implementation uses Python and image-processing libraries to perform image preparation, damage-mask generation, restoration, and output visualization.

The use of a binary damage mask enables the system to identify regions requiring reconstruction while retaining the available information from undamaged areas. The generated outputs, including the original image, damaged image, damage mask, and restored image, support visual analysis of the restoration process. The restoration quality depends on the size and pattern of the damaged region, the surrounding image information, and the numerical solution method used.

Although the proposed method is computationally straightforward and suitable for small-scale image-restoration tasks, it has limitations when processing large missing regions, complex textures, and detailed structures. The Laplace-based approach primarily encourages smoothness and may not accurately reproduce information that cannot be inferred from neighbouring pixels.

Overall, the project provides a practical demonstration of PDE-based image inpainting and establishes a foundation for further development in image restoration. Future enhancements, including improved numerical solvers, automatic damage detection, advanced inpainting algorithms, and comprehensive performance evaluation, could increase its accuracy, efficiency, and applicability to real-world damaged images.

APPENDIX

PDE-Based Image Inpainting Algorithm

The core functionality of the proposed PDE-based image inpainting system relies on the Laplace equation to reconstruct missing or damaged image regions. The algorithm uses an input image, a binary damage mask, and an iterative numerical procedure to estimate missing pixel intensities from neighbouring pixels.

- **Input**:

Original image (I)

Binary damage mask (M), identifying pixels that require restoration

Maximum number of iterations or a convergence tolerance, depending on the implementation

- **Process:**

1. Image Loading: The system loads the input image using Python image-processing libraries.
2. Damage Mask Generation: A binary mask is created to identify the damaged regions. Masked pixels are marked for reconstruction, while the remaining pixels represent known image information.

3. Image Initialization: The damaged image is prepared for restoration. The missing pixel values are initialized using the approach implemented in the system.
4. PDE Formulation: The Laplace equation is used to estimate the missing pixel intensities:

   $\nabla^2 I = 0$
5. The equation can be expressed in two dimensions as:

$$\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = 0$$

6. Numerical Solution: The discretized equation estimates each unknown pixel using its neighbouring pixel values. For a four-neighbour finite-difference scheme, the update rule is:

$$I(i,j) = \frac{1}{4}[I(i-1,j) + I(i+1,j) + I(i,j-1) + I(i,j+1)]$$

1. The update is repeated using the numerical solver implemented in the system until the stopping condition is reached or the iteration limit is met.
2. Image Reconstruction: The estimated pixel intensities are used to reconstruct the damaged regions while preserving known pixel values where supported by the implementation.
3. Output Generation: The system generates the restored image and displays or saves the original image, damaged image, damage mask, and restored image for comparison.
4. Output:Restored image with reconstructed damaged regions. Visual comparison of the original, damaged, masked, and restored images.Quantitative restoration metrics, such as PSNR and SSIM, if implemented and if the original reference image is available. This algorithm provides a systematic approach to image restoration using partial differential equations and numerical methods. Its effectiveness depends on the damage pattern, the available neighbouring information, and the numerical solver used.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to their project guide, **Dr.V.Kamalaveni**, Professor /AIDS, for his valuable guidance, continuous support, and technical insights throughout the development of *PDE Based image inpainting.*

We also extend our heartfelt thanks to the Head of the Department, Dr. N. M. Saravana Kumar, and the faculty members of the Department of Artificial Intelligence and Data Science at Sri Shakthi Institute of Engineering and Technology for providing the necessary infrastructure and academic support to successfully complete this project.

We would like to acknowledge the technical staff for their assistance in utilizing laboratory resources and tools required for development and testing.

Finally, we express our sincere appreciation to our families, friends, and peers for their constant encouragement, motivation, and constructive feedback throughout the project development process.

REFERENCES

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image Inpainting,” in Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 417–424, 2000. doi: 10.1145/344779.344972.

[2] T. F. Chan and J. Shen, “Mathematical Models for Local Nontexture Inpaintings,” SIAM Journal on Applied Mathematics, vol. 62, no. 3, pp. 1019–1043, 2002. doi: 10.1137/S0036139900368844.

[3] A. Telea, “An Image Inpainting Technique Based on the Fast Marching Method,” Journal of Graphics Tools, vol. 9, no. 1, pp. 23–34, 2004. doi: 10.1080/10867651.2004.10487596.

[4] A. Criminisi, P. Pérez, and K. Toyama, “Region Filling and Object Removal by Exemplar-Based Image Inpainting,” IEEE Transactions on Image Processing, vol. 13, no. 9, pp. 1200–1212, 2004. doi: 10.1109/TIP.2004.833105.

[5] T. F. Chan and J. Shen, “Nontexture Inpainting by Curvature-Driven Diffusions,” Journal of Visual Communication and Image Representation, vol. 12, no. 4, pp. 436–449, 2001. doi: 10.1006/jvci.2001.0487.

[6] Python Software Foundation, “Python 3 Documentation.” [Online]. Available: https://docs.python.org/3/

[7] NumPy Developers, “NumPy Documentation.” [Online]. Available: https://numpy.org/doc/stable/

[8] Matplotlib Development Team, “Matplotlib Documentation.” [Online]. Available: https://matplotlib.org/stable/

[9] Pillow Contributors, “Pillow (PIL Fork) Documentation.” [Online]. Available: https://pillow.readthedocs.io/en/stable/

[10] Microsoft, “Visual Studio Code Documentation.” [Online]. Available: https://code.visualstudio.com/docs