Ahmedabad
University

ECE501- DIGITAL IMAGE PROCESSING

WEEKLY REPORT – Week 3: (18-10-2025)

Project No.: 6

Project Title: Image Inpainting for Object Removal

Group Name: 12

Group Details:		
Sr No	**Name**	**Enrollment No:**
1	Sakina Jambughodawala	AU2340114
2	Prina Patel	AU2340040
3	Aneri Maniar	AU2340080
4	Nidhee Adesara	AU2340196

Problem Definition:

The goal of this project is to fill the missing regions using the information of surrounding texture and color so that the image looks natural.

Image inpainting is the process in which an image whose parts are lost are reconstructed based on information of remaining context

Objective of the Week:

1. Understood the diffusion-based techniques and started to implement it.
2. Studied how to identify boundary pixels and calculate isophotes (gradient directions) from known regions.
3. As we are using heat diffusion equations which have three types, we are figuring how to incorporate it in the iteration to spread pixel values.
4. Implement the preprocessing of diffusion in code.

Mathematical Equations:

The equations that will be used are as follows:

Diffusion Equation

The diffusion-based inpainting process is mathematically modeled using partial differential equations (PDEs) and energy minimization principles.

1) PDE-Based Diffusion Equation:
This equation diffuses pixel values from known to missing regions, similar to heat flow, ensuring smooth filling.

$$\frac{\partial I(x,y,t)}{\partial t} = \nabla \cdot (D(x,y)\nabla I(x,y,t))$$

Where:

- **I(x, y, t)** – Image intensity at position (x, y) and time t
- **D(x, y)** – Diffusion coefficient (controls smoothness)
- ∇**I** – Image gradient (direction of intensity change)
- ∇· – Divergence operator (measures spread)

2) Variational / Energy Minimization Formulation:

$$E(I) = \int_\Omega |\nabla I|^2 \, dx \, dy$$

This minimizes the total image energy to obtain the smoothest and most natural reconstruction.

Where:

- **E(I)** – Total image energy
- ∇**I** – Gradient of image intensity
- **Ω** – Image domain (region being processed)

3) Geometry-Guided Diffusion:

$$\frac{\partial I}{\partial t} = \nabla \cdot \left(\frac{\nabla I^\perp}{|\nabla I|} \right)$$

In this formulation, diffusion occurs along image edges rather than across them, preserving important boundaries and structures. This approach is known as anisotropic diffusion.

Where:

- **$\nabla I \perp$** – Gradient perpendicular to edges
- **$|\nabla I|$** – Magnitude of gradient (edge strength)

The inpainting process is governed by the heat flow or smoothness equation, which is a diffusion partial differential equation (PDE):

$$\frac{\partial I}{\partial t} = \nabla \cdot (\nabla I)$$

Here, **I** represent the image intensity. This equation gradually updates missing pixel values based on neighboring known pixels, leading to a smooth and visually consistent reconstruction.

CODE:

The code which we implemented for now, does 3 things:

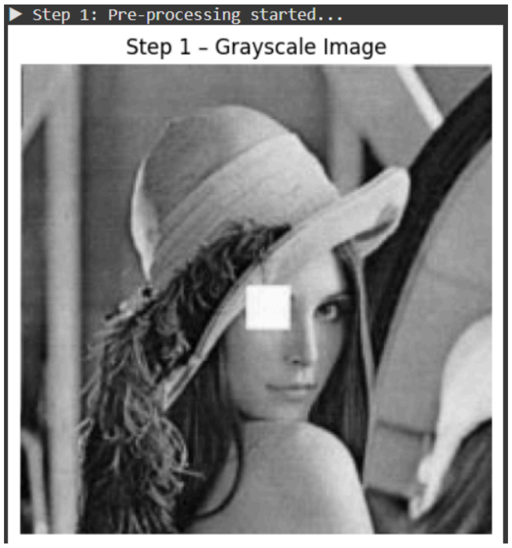
1. **Pre-Processing**
    - First it loads the input image and then converts it to grayscale for intensity-based processing.
    - Then it applies a Gaussian blur to smooth noise and prepares for mask generation. Then grayscale and blurred images are displayed.
2. **Mask Creation**
    - Automatically creates a binary mask that will identify damaged or missing regions.

- Pixels brighter than a threshold (e.g., 0.97) are considered missing (white).
- Uses morphological opening to clean up the mask and displays the mask and reports how much of the image is masked.

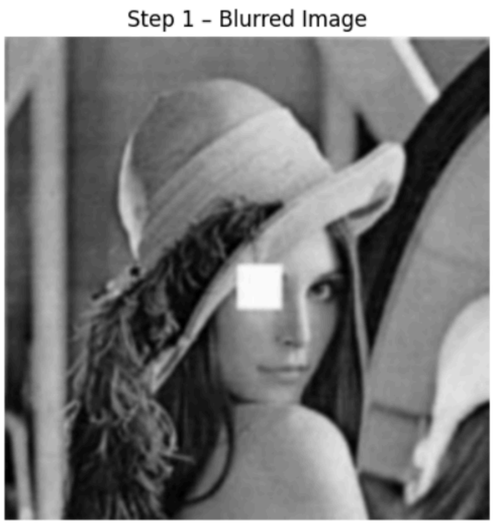
3. **Initialization**
    - Prepares an initial guess for the inpainting process.
    - Replaces masked or damaged regions with either:
        - Blurred pixel values (default)
        - Mean intensity of known areas
        - Zero intensity (black).
    - Displays the initialized image which is ready for diffusion.

Output

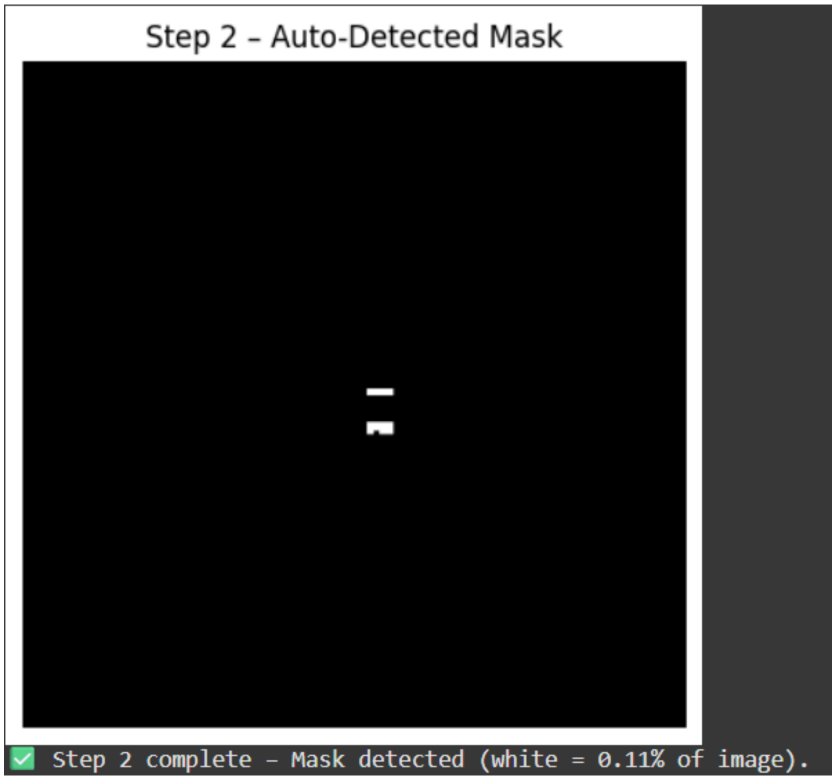
A grayscale and blurred image



```
▶ Step 1: Pre-processing started...
```

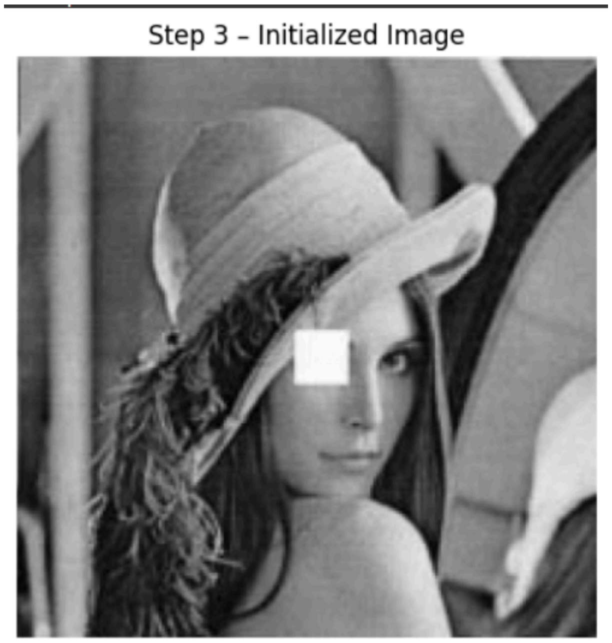


A binary mask showing damaged regions, and



```
✅ Step 2 complete – Mask detected (white = 0.11% of image).
```

An initialized image ready for the diffusion-based inpainting step that will fill in missing areas.



CODE:

https://colab.research.google.com/drive/1bVEIFaMLdEJjWPgpKfD8Lq4P5-kF62Py?usp=sharing

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# helper to display images inline
def show_img(title, img):
    plt.figure(figsize=(5,5))
    if len(img.shape) == 2:
        plt.imshow(img, cmap='gray')
    else:
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.axis('off')
    plt.show()

# ==============================
# Step 1 — Pre-Processing
# ==============================
def step1_preprocess(img_path, blur_sigma=1.0):
    print("▶ Step 1: Pre-processing started...")

    img_bgr = cv2.imread(img_path, cv2.IMREAD_COLOR)
    if img_bgr is None:
        raise FileNotFoundError(f"Image not found: {img_path}")

    gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY).astype(np.float32)
/ 255.0
    ksize = max(3, int(round(blur_sigma * 4)) | 1)
    blurred = cv2.GaussianBlur(gray, (ksize, ksize), blur_sigma)

    show_img("Step 1 – Grayscale Image", gray)
    show_img("Step 1 – Blurred Image", blurred)
    print("✅ Step 1 complete – Gray and blurred images displayed.")
    return gray, blurred


# ============================
# Step 2 — Mask Creation (auto)
# ============================
def step2_create_mask_auto(gray_img, threshold=0.97):
    print("▶ Step 2: Mask creation (auto) started...")

    mask_bin = (gray_img > threshold).astype(np.uint8)
    kernel = np.ones((3,3), np.uint8)
    mask_bin = cv2.morphologyEx(mask_bin, cv2.MORPH_OPEN, kernel)

    show_img("Step 2 – Auto-Detected Mask", mask_bin * 255)

    white_ratio = np.mean(mask_bin) * 100
    print(f"✅ Step 2 complete – Mask detected (white =
{white_ratio:.2f}% of image).")
    return mask_bin


# ============================
# Step 3 — Initialization
# ============================
def step3_initialize(gray, blurred, mask_bin, init_method="blur"):
    print("▶ Step 3: Initialization started...")

    init_img = gray.copy()

    if init_method == "blur":
        init_img[mask_bin == 1] = blurred[mask_bin == 1]
        method_used = "blurred-region fill"
    elif init_method == "mean":
        mean_val = np.mean(gray[mask_bin == 0])
        init_img[mask_bin == 1] = mean_val
        method_used = f"mean fill (mean={mean_val:.3f})"
    else:
        init_img[mask_bin == 1] = 0.0
        method_used = "zero fill"

    show_img("Step 3 – Initialized Image", init_img)

    print(f"✅ Step 3 complete – Initialization done using {method_used}.")
    return init_img


# ============================
# Driver Code
# ============================
if __name__ == "__main__":
    # 👇 path to your damaged Lena image
    img_path = "/content/Original.png"

    # Step 1
    gray, blurred = step1_preprocess(img_path, blur_sigma=1.0)

    # Step 2
    mask_bin = step2_create_mask_auto(gray)

    # Step 3
    init_img = step3_initialize(gray, blurred, mask_bin, init_method="blur")

    print("\n🎯 Up to Step 3 complete – ready for diffusion/inpainting next.")
```

Plan for Next Week:

- We will work on our python code further and implement the PDE-based diffusion process using the paper equations.
- Before that we will direct the pixel intensity from known regions to unknown regions pixel values.
- Then we will apply Anisotropic diffusion and observe the results.