NUST Balochistan Campus (NBC)

Department of Computer Science

Batch 2022

“DIP Semester Project”

“Watermark adder and Removal”

Submitted to: Sir Rozi Khan

Submitted by:

- Muhammad Haroon (CMS: 420639)
- Muhammad Anas Hassaan (CMS: 410555)
- Rania Zulfiqar (CMS: 422077)
- Muhammad Ibtesam (CMS: 416646)

Theoretical Explanation of Digital Image Watermark Adder & Remover Project

1. Project Overview

This project implements a **Digital Image Processing (DIP)-** based system for adding and removing watermarks from images. It explores two primary functionalities:

1. **Watermark Embedding**: Inserts visible watermarks into images.
2. **Watermark Removal/Extraction**: Attempts to remove or detect embedded watermarks.

The system is designed for applications such as:

- **Copyright Protection** (e.g., photographer signatures)
- **Content Authentication** (e.g., tamper detection)

2. Core Theoretical Concepts

2.1 Digital Watermarking Fundamentals

A **digital watermark** is a signal (text, logo, or noise pattern) embedded into an image such that:

- It can be **detected/extracted** later for verification.
- It minimally affects **visual quality** (for invisible watermarks).
- It resists **common distortions** (compression, cropping, noise).

3. Watermark Embedding Techniques

3.1 Visible Watermarking (Spatial Domain)

3.1.1 Text-Based Watermarking

- **Principle**: Text is overlaid on the image with adjustable:
  - **Opacity** (transparency level)
  - **Position** (top, bottom, center, custom)
  - **Font & Color** (for better visibility)

3.1.2 Image/Logo-Based Watermarking

- **Principle**: A secondary image (e.g., a logo) is blended into the original image.
- **Key Parameters**:
  - **Blending Factor** (controls visibility)
  - **Embedding Location** (affects robustness)
- **Advantages Over Text Watermarking**:
  - Harder to remove completely.
  - Better brand identification.

4.Watermark Removal Technique:

1. Overview of Watermark Removal Functionality

1. Core Watermark Removal Process Overview

The system implements a complete watermark removal pipeline using Simple LaMa (Large Mask Inpainting) with these key stages:

1. Image Acquisition (URL or file upload)
2. Automatic Watermark Detection (mask generation)
3. LaMa Inpainting (content reconstruction)
4. Result Delivery (encoded image response)

2.Simple LaMa Implementation Details:

- Uses a lightweight version of LaMa architecture
- Pre-trained weights loaded automatically
- Optimized for moderate GPU usage (works with 4GB+ VRAM)

3. Watermark Removal Pipeline

3.1 Image Preprocessing

```
image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
pil_image = Image.fromarray(image_rgb)
```

- Converts OpenCV BGR format to PIL RGB
- Maintains original resolution (with optional resizing)
- Preserves full color information

3.2 Automatic Mask Generation

```
mask = _generate_watermark_mask(img)
```

Multi-stage Detection Process:

1. **Grayscale Conversion**:
    - Reduces color variability impact
2. **Gaussian Blurring** (5×5 kernel):
    - Smooths noise while preserving edges
3. **Canny Edge Detection** (30/100 thresholds):
    - Identifies high-contrast watermark boundaries
4. **Morphological Processing**:
    - Dilation (expands detected edges)
    - Closing (fills small gaps)
5. **Gaussian Smoothing**:
    - Softens mask edges for better inpainting

3.3 Inpainting Execution

```
inpainted_pil = lama(pil_image, pil_mask)
```

Model Operations:

1. Encodes image and mask through FFC blocks
2. Identifies contextual information around masked areas
3. Progressively reconstructs missing regions
4. Blends new content with existing pixels

3.4 Post-processing

inpainted_bgr = cv2.cvtColor(inpainted_rgb, cv2.COLOR_RGB2BGR)

- Converts back to OpenCV BGR format
- Maintains original color space
- No quality-reducing compression at this stage

5. Complete Workflow Example

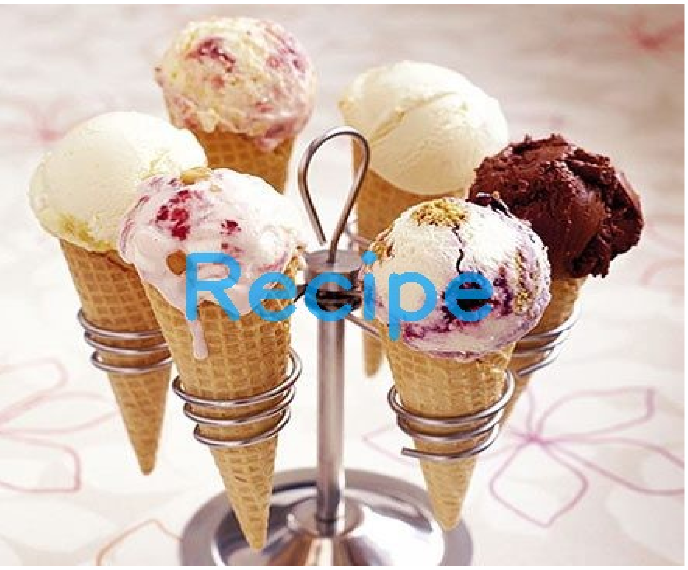
1. User uploads watermarked image (POST) or provides URL (GET)
2. **System:**
    - Validates input
    - Loads image (URL or file)
    - Generates detection mask
    - Applies Simple LaMa inpainting
3. **Returns cleaned image as JPEG:**
    - 90% quality compression
    - Original dimensions preserved
    - Color space maintained

Code Output Screenshots:

- **Watermark adder:**

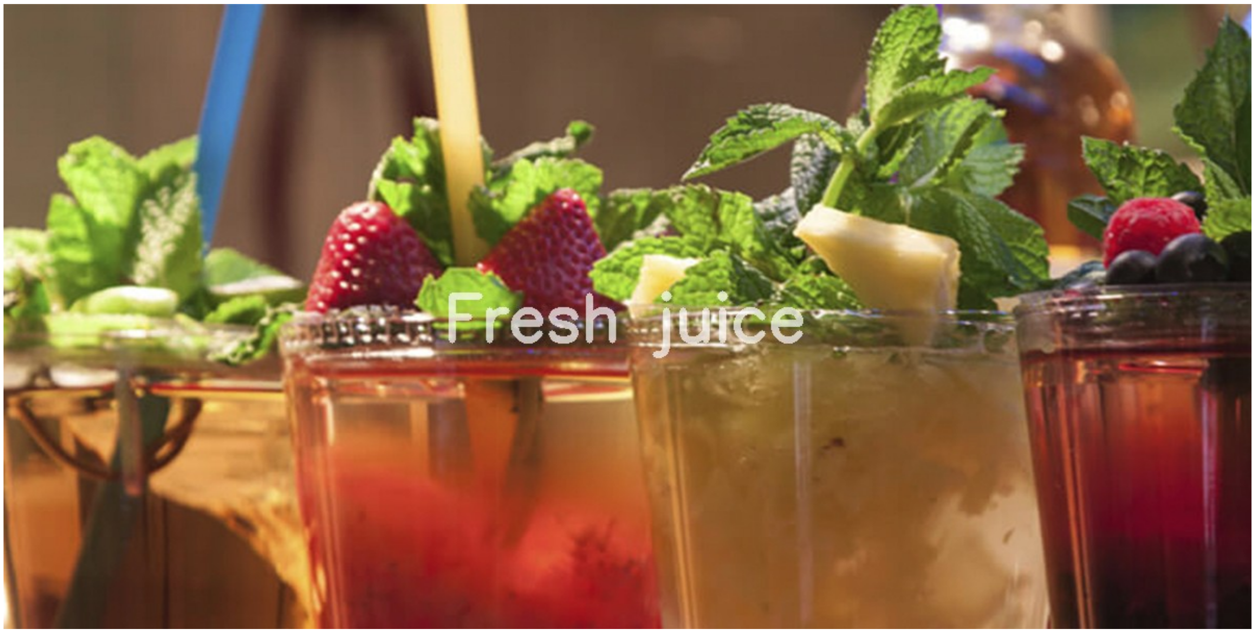
1)Before:

1)After:



2) Before:

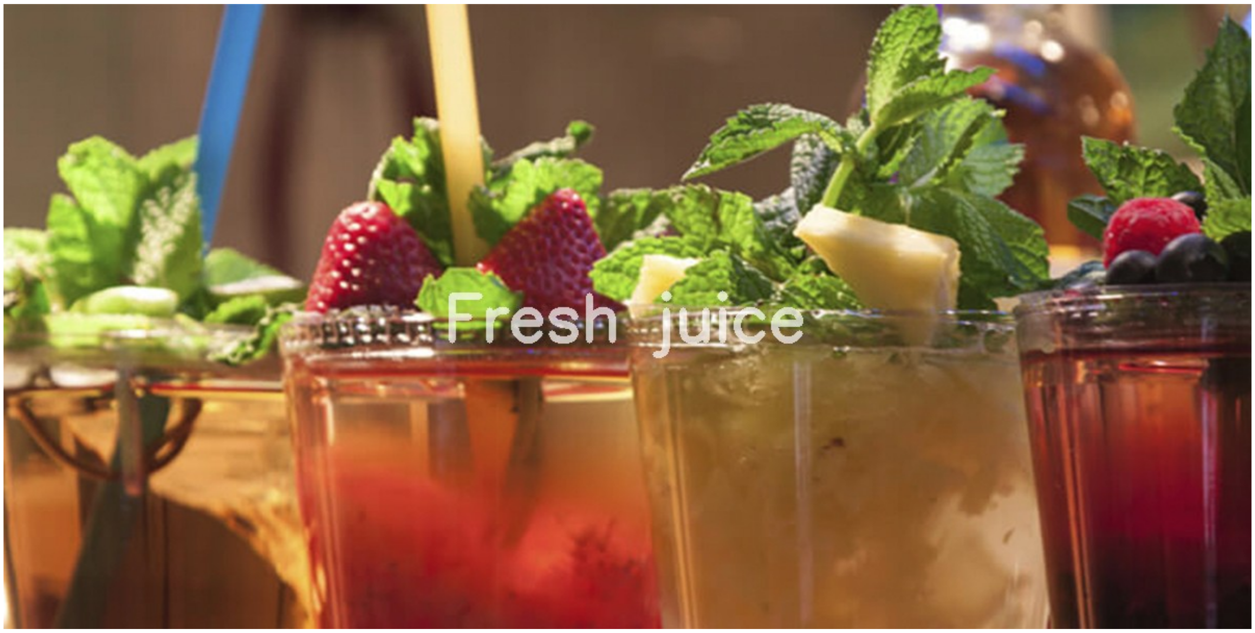
2) After:



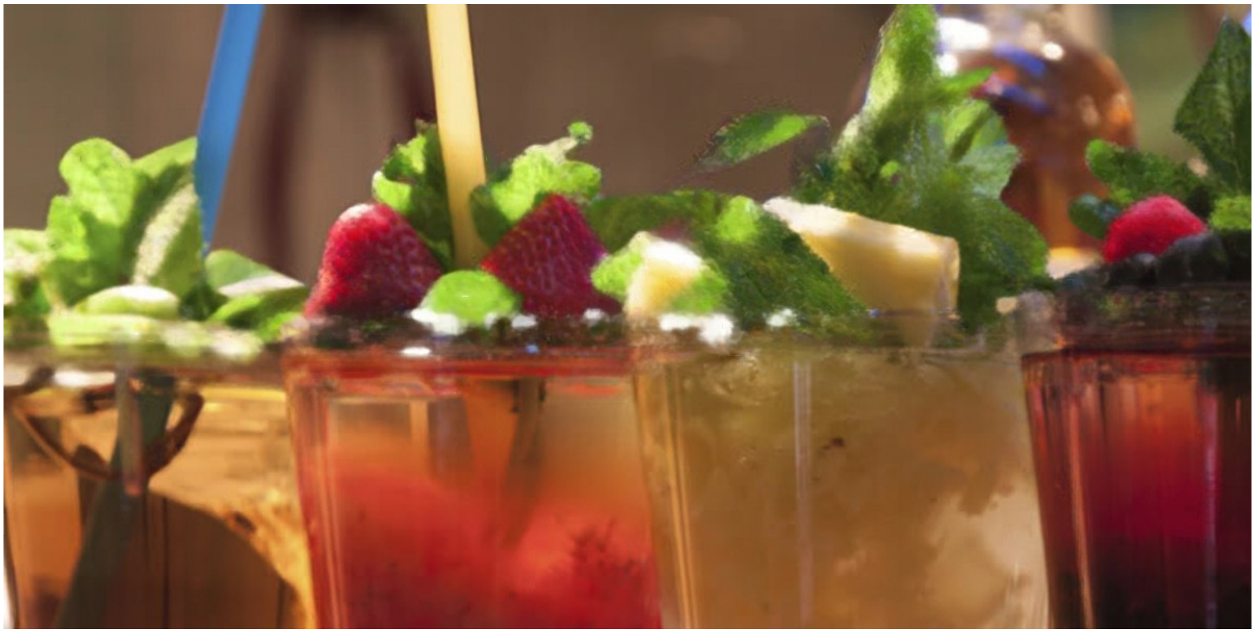
Code Output Screenshots:

- Watermark Removal:

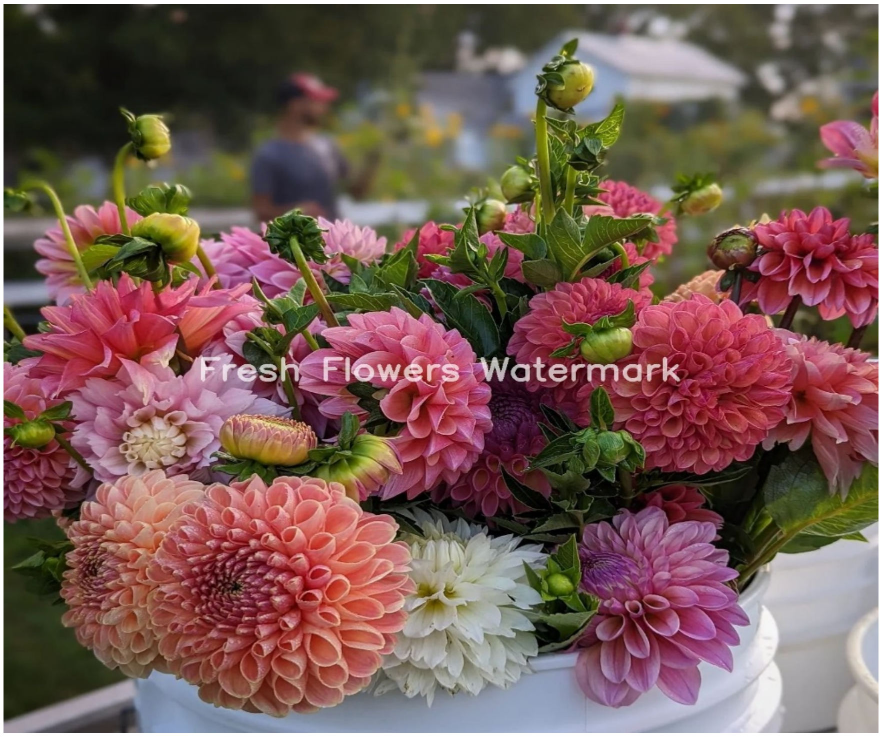
1)Before:



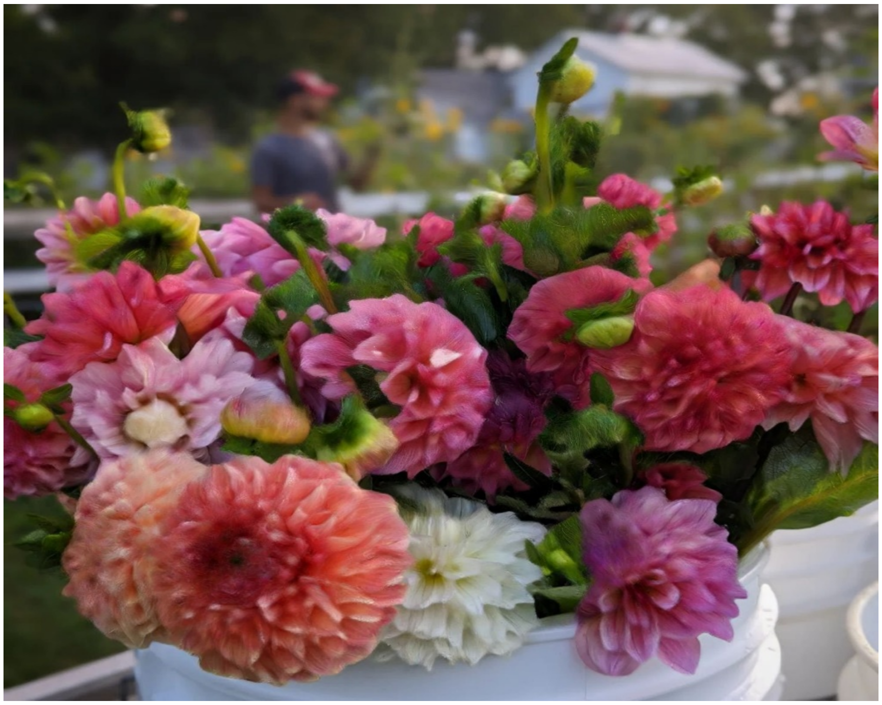
2) After:



2)Before:



2) After:



Code GitHub Repository:

https://github.com/muhammadharoon26/DIP_Image-Watermark-Adder__Remover_Semester-Project.git