PATCH MATCH

DIGITAL IMAGE PROCESSING

ESWAR(2023102011)
YASWANTH(2022102051)

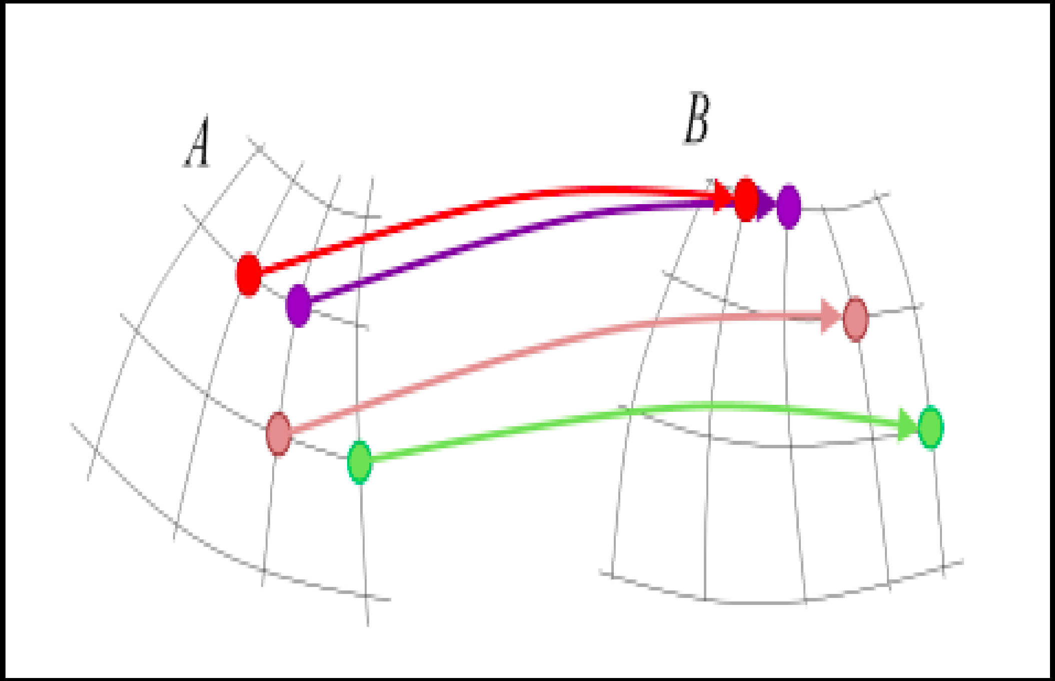
INTRODUCTION

Problem statement
Advancements in digital photography enable image manipulation using patch-based techniques like retargeting and completion, however there are computational challenges.

PatchMatch
PatchMatch accelerates nearest-neighbor search for image patches, enabling efficient in-painting, retargeting, and reshuffling, widely used in Adobe tools (Content-Aware Fill feature)

PATCHMATCH AT A GLANCE

- What PatchMatch computes: dense Nearest Neighbour Field (NNF).
- Core idea: propagation + random search → very fast nearest-patch search.
- Why useful: in-painting, retargeting, reshuffling.

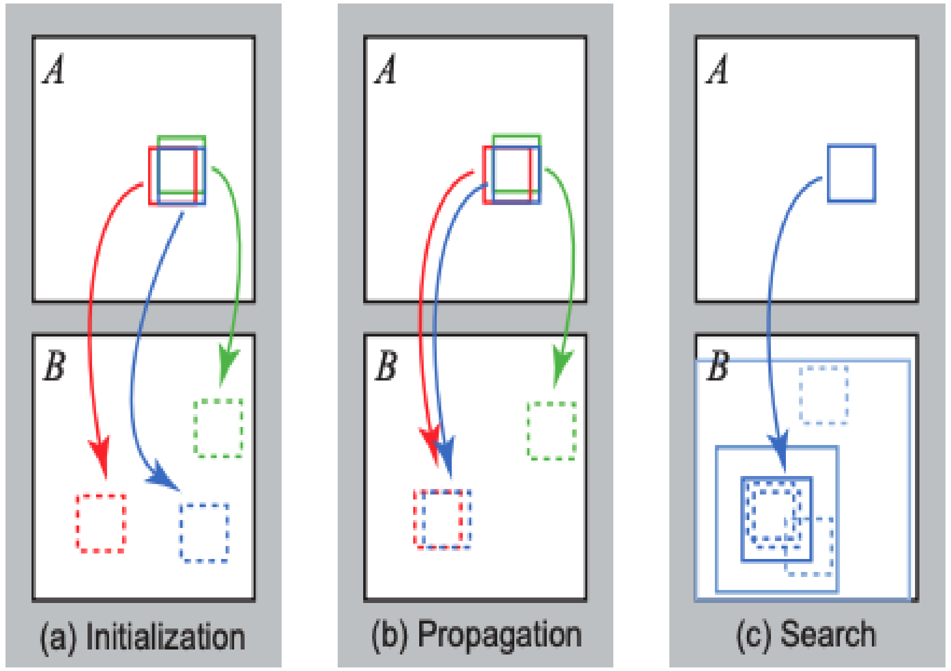


NNF ALGORITHM

1. INITIALIZATION
The algorithm initializes by randomizing coordinates or using prior information, refining correspondences progressively for in-painting and similar applications.

2. PROPAGATION
Propagation transfers good offset estimates to neighboring patches, leveraging local coherence, minimizing error, and alternating forward-reverse scans for refinement.

3. RANDOM SEARCH
Random search refines offsets by exploring nearby variations, reducing search radius exponentially until sub-pixel precision ensures optimal matching.



IN-PAINTING PIPELINE (COARSE-TO-FINE + EM)

1. INITIALISATION

- PatchMatch begins by initializing the **image, mask, patch size, and iteration parameters** needed for in-painting.
- A **binary mask** specifies the missing regions where patch synthesis must occur.
- An **image pyramid (coarse-to-fine)** enables global structure restoration at low resolution and detail refinement at higher resolutions.

IN-PAINTING PIPELINE (COARSE-TO-FINE + EM)

2. PYRAMID CONSTRUCTION

•The image and mask are **progressively downsampled** until the smallest level roughly matches the patch size.

•**Gaussian smoothing** with a custom kernel reduces noise and ensures smooth transitions between pyramid levels.

•**Missing mask regions are propagated** across scales using weighted averaging for consistent hole representation

IN-PAINTING PIPELINE (COARSE-TO-FINE + EM)

3. EM OPTIMIZATION

•E–Step
•Nearby matching patches cast weighted “votes” for filling missing pixels.
•The weight for each patch is computed as:
$w = 1 - \frac{d(P_A, P_B)}{\text{MAX\_PATCH\_DIFF}}$

•M–Step
•Pixel values in the missing region are updated by averaging the weighted votes

IN-PAINTING PIPELINE (COARSE-TO-FINE + EM)

4. MULTI-SCALE COARSE-TO-FINE REFINEMENT

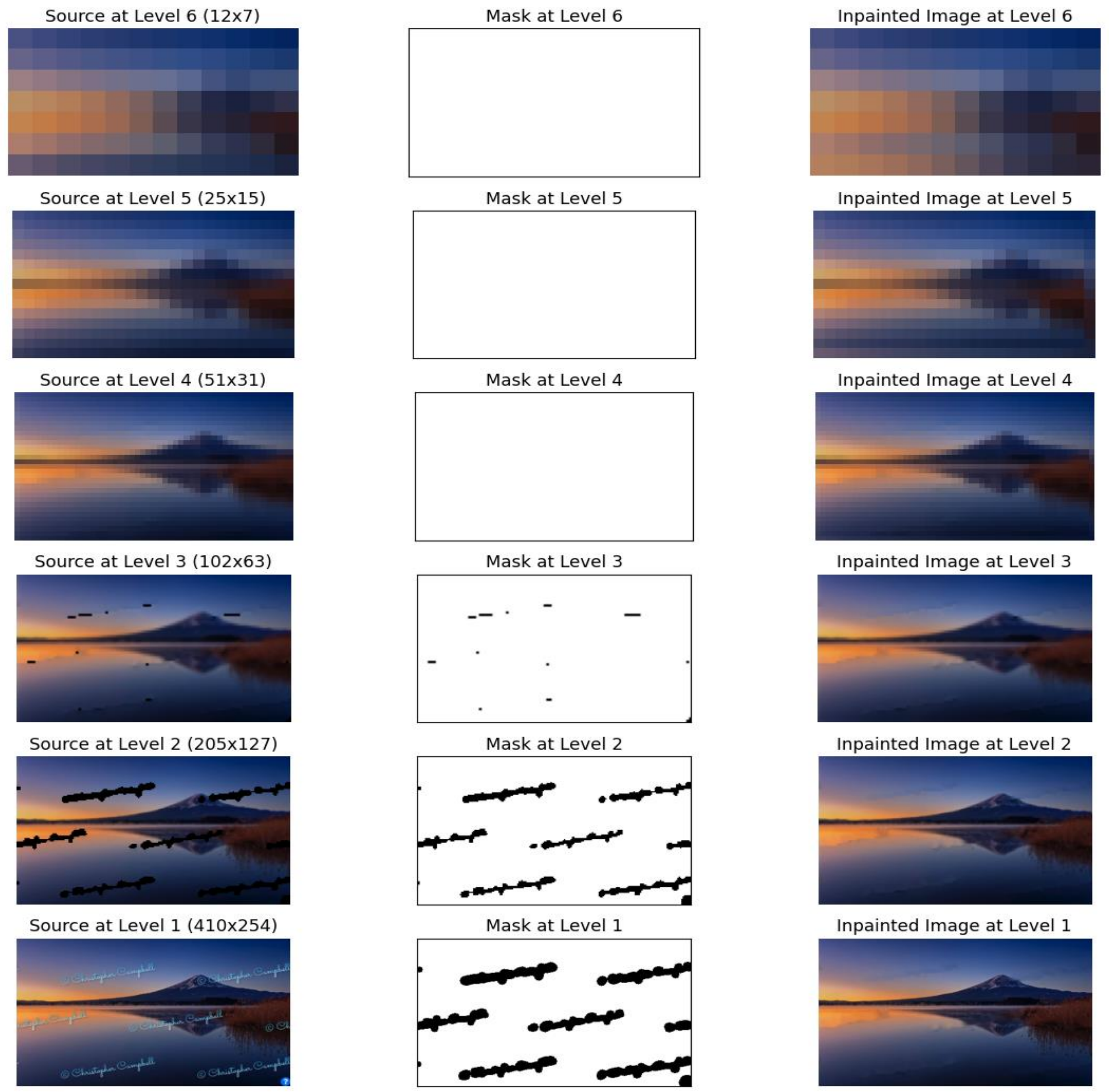
•PatchMatch initialization begins at the **coarsest pyramid level**, where missing regions are first estimated.

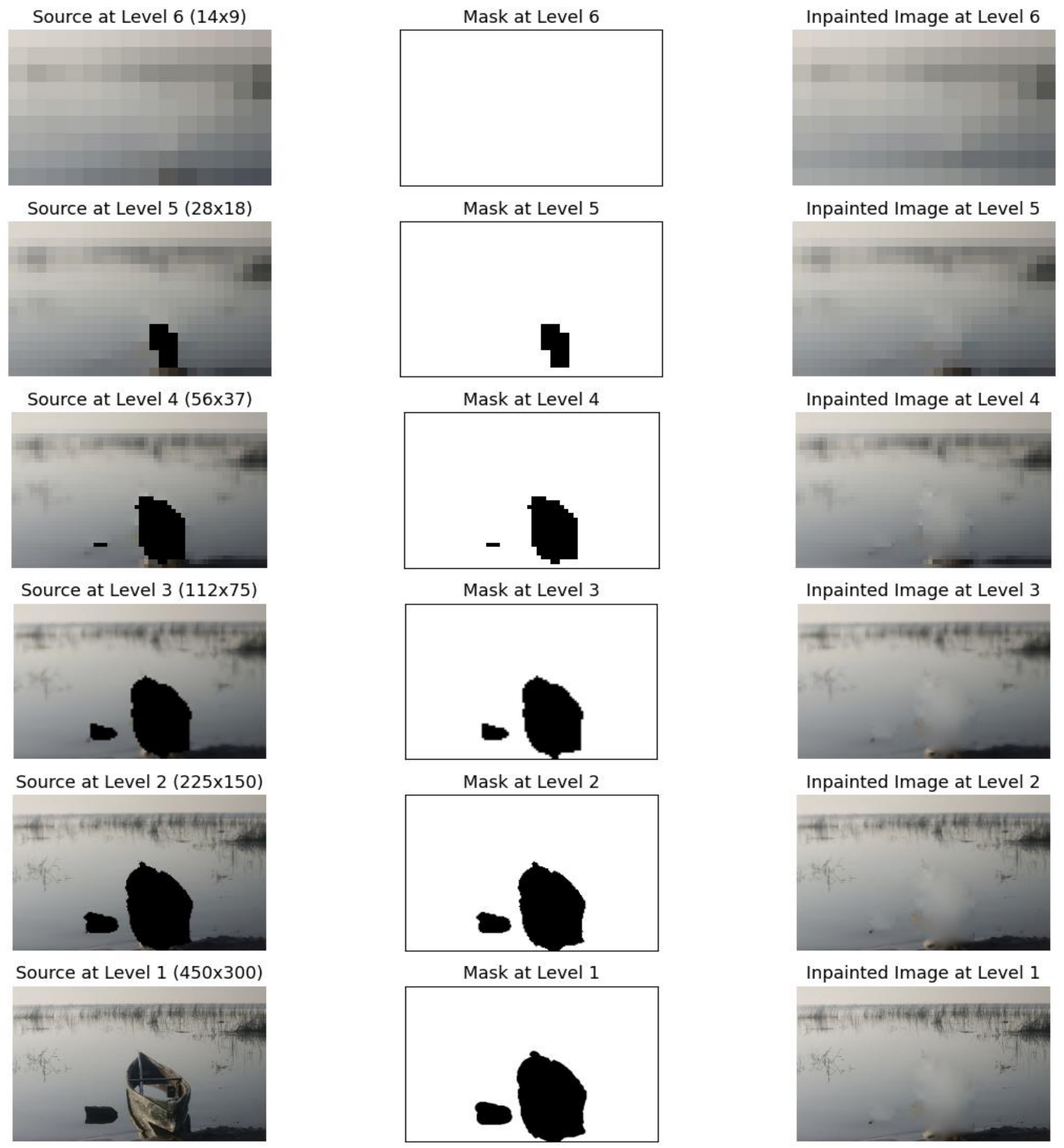
•Intermediate results are **upsampled using bilinear interpolation** to maintain smooth transitions between scales.

•The **NNF (Nearest Neighbour Field)** computed at coarse levels is propagated upward to finer resolutions.

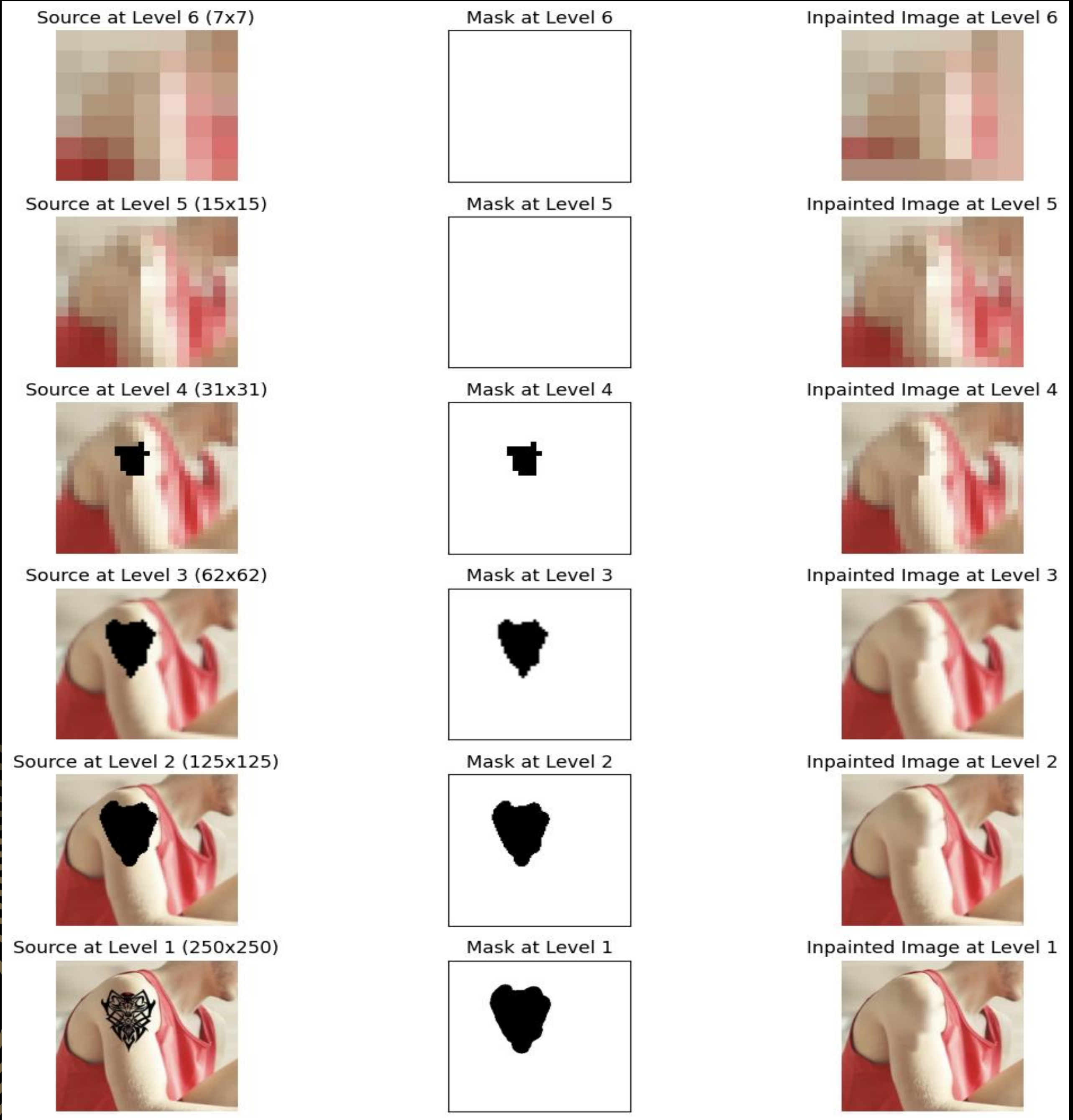
•This coarse-to-fine propagation **reduces computation** while preserving structural and visual consistency across levels.

IN-PAINTING BOAT REMOVAL AND WATERMARK REMOVAL

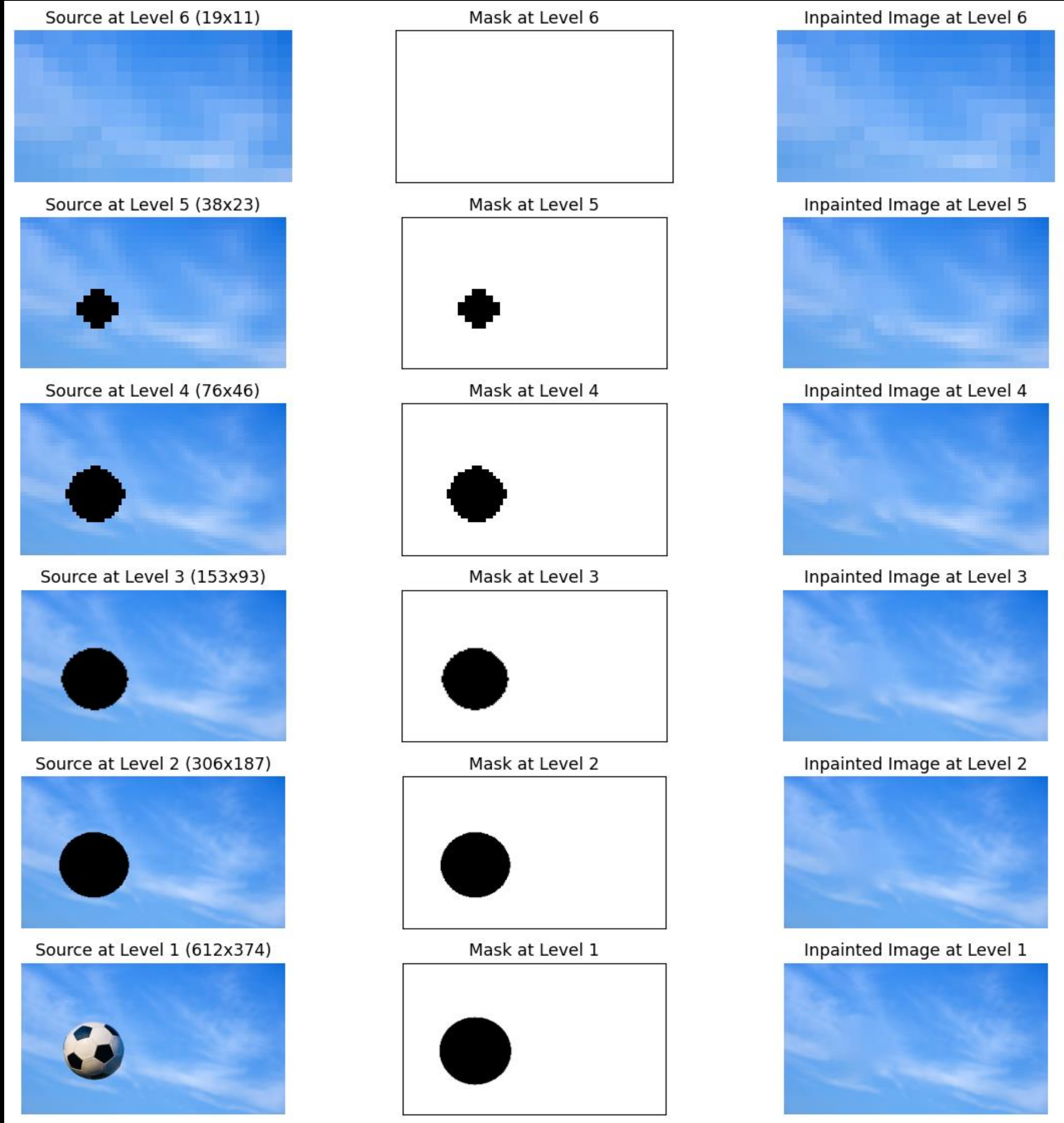

Source at Level 6 (12x7)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (25x15)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (51x31)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (102x63)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (205x127)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (410x254)
Mask at Level 1
Inpainted Image at Level 1


Source at Level 6 (14x9)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (28x18)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (56x37)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (112x75)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (225x150)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (450x300)
Mask at Level 1
Inpainted Image at Level 1

IN-PAINTING TATT00 REMOVAL AND OBJECT REMOVAL

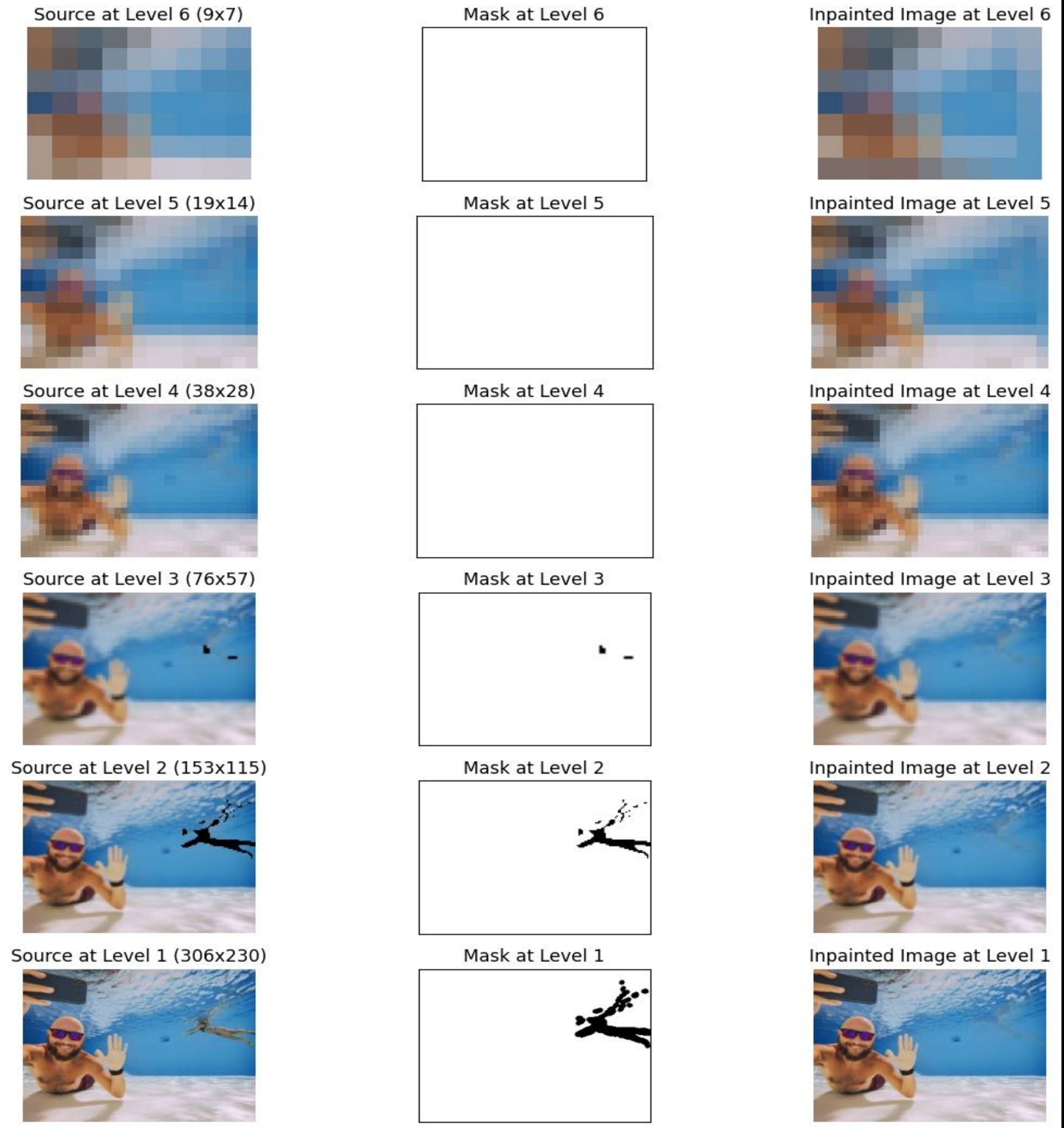


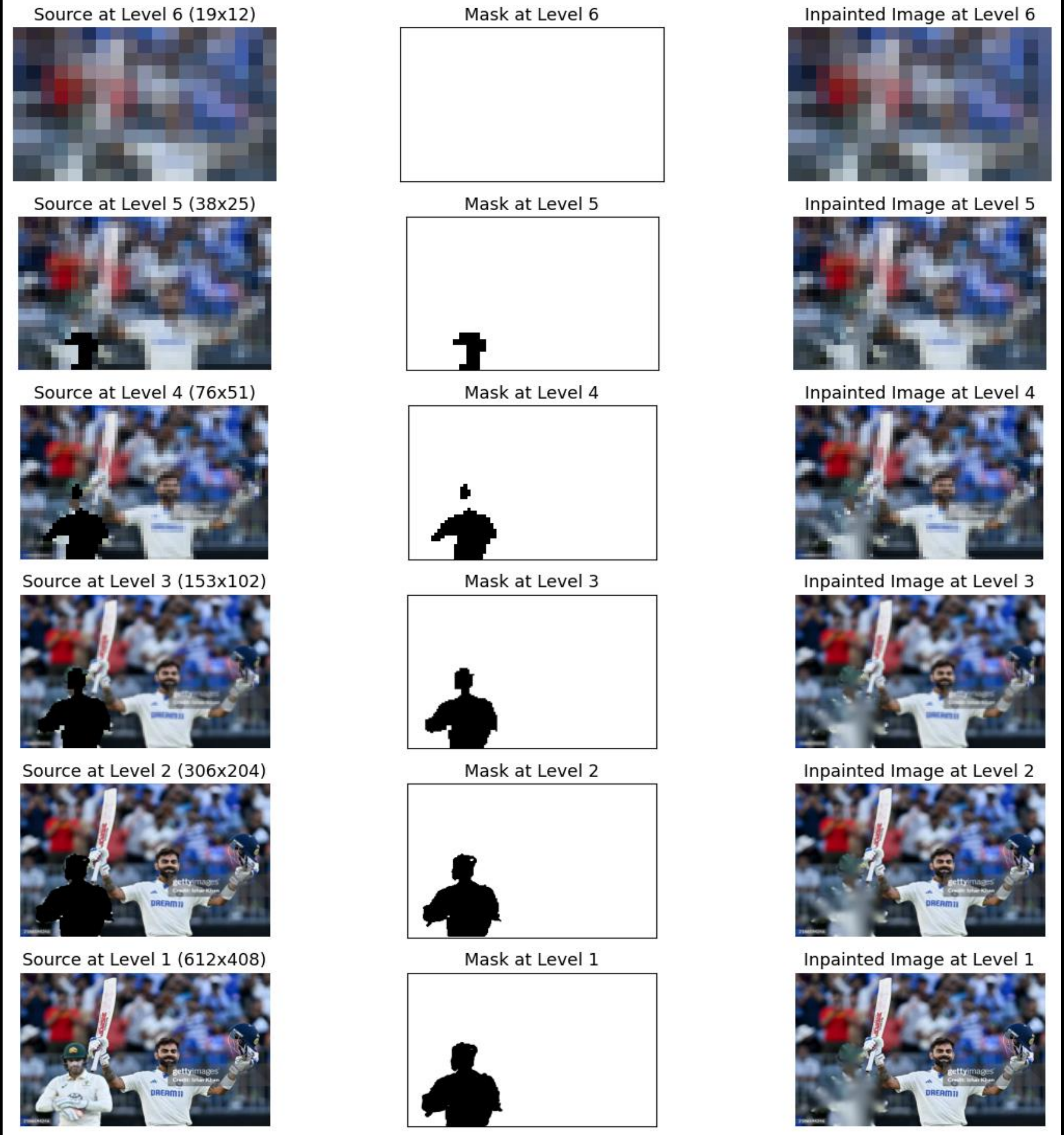
Source at Level 6 (7x7)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (15x15)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (31x31)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (62x62)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (125x125)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (250x250)
Mask at Level 1
Inpainted Image at Level 1



Source at Level 6 (19x11)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (38x23)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (76x46)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (153x93)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (306x187)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (612x374)
Mask at Level 1
Inpainted Image at Level 1

IN-PAINTING PHOTO BOMB AND MAN REMOVAL


Source at Level 6 (9x7)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (19x14)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (38x28)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (76x57)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (153x115)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (306x230)
Mask at Level 1
Inpainted Image at Level 1


Source at Level 6 (19x12)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (38x25)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (76x51)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (153x102)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (306x204)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (612x408)
Mask at Level 1
Inpainted Image at Level 1

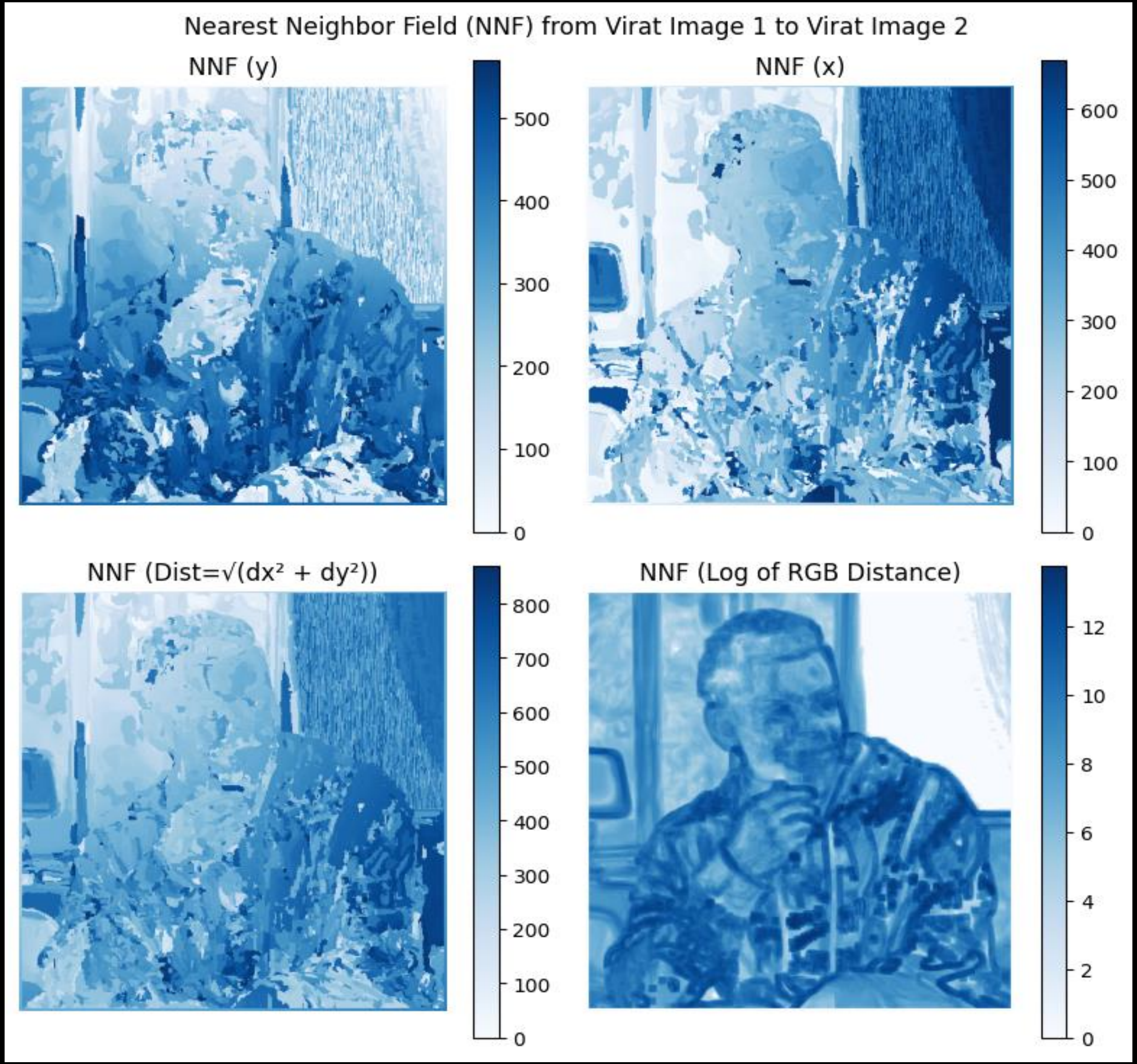
NNF RESULTS

1. NNF(y) represents the pixel distance difference along the y axis

2. NNF(x) represents the pixel distance difference along the x axis

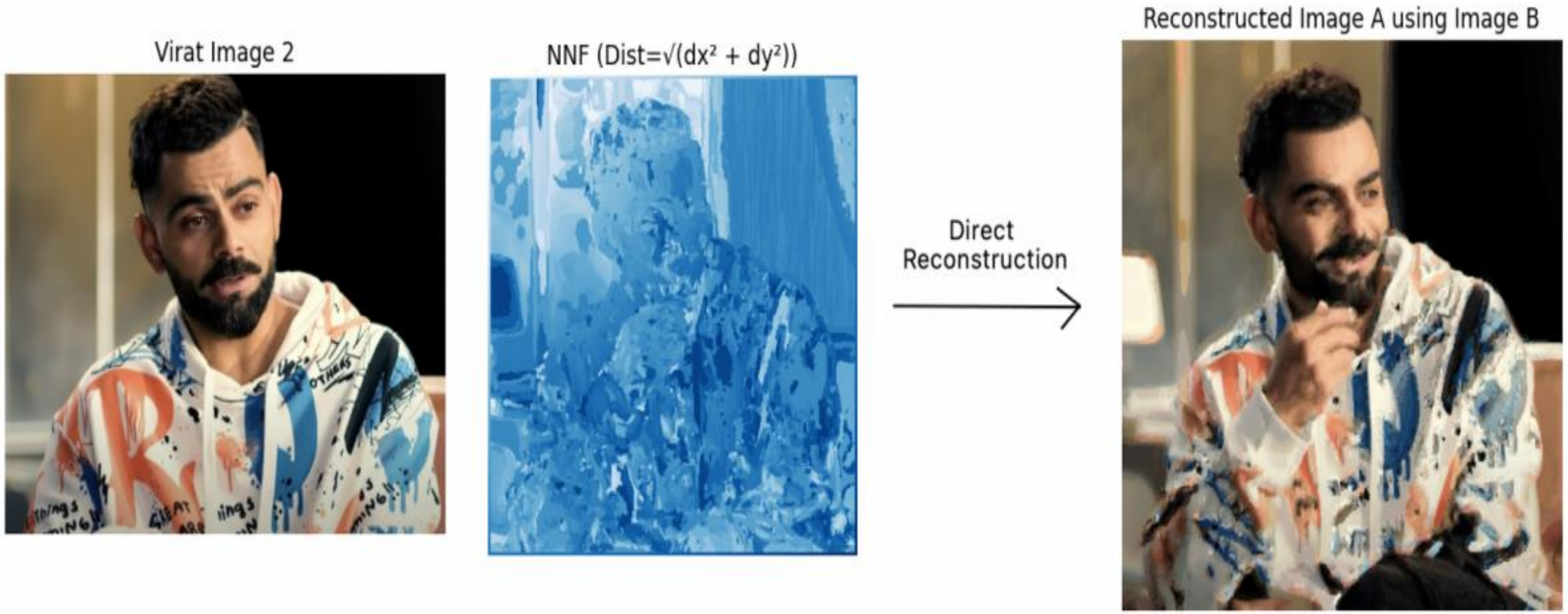
3. The NNF(Dist) represents the magnitude of distance combining NNF(x) and NNF(y)

4. The NNF(RGB dist) represents the pixel intensity difference between each patch in A and B



NNF RECONSTRUCTION(METHOD-1)

By directly replacing the pixel value with the centre of the NNF patch in B



NNF RECONSTRUCTION(METHOD-2)

By replacing patches for each pixel centre and averaging the pixel intensities over all overlapping patches

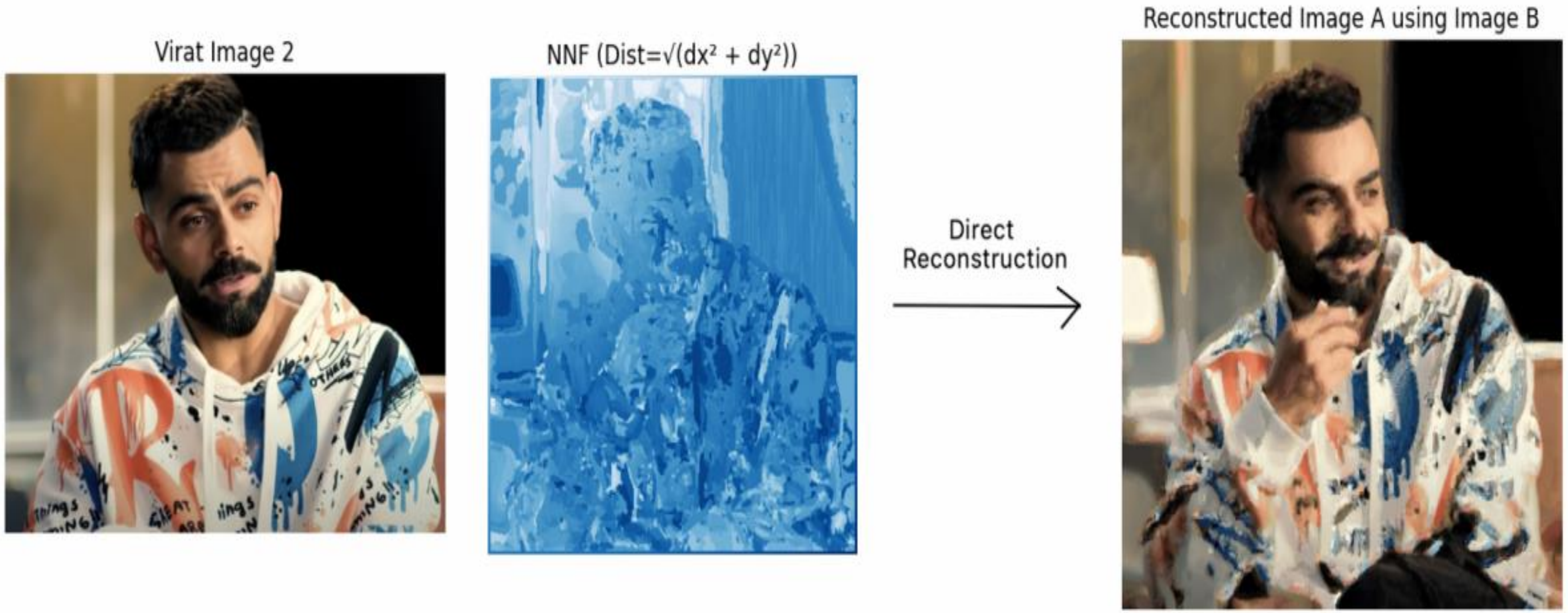


Image A

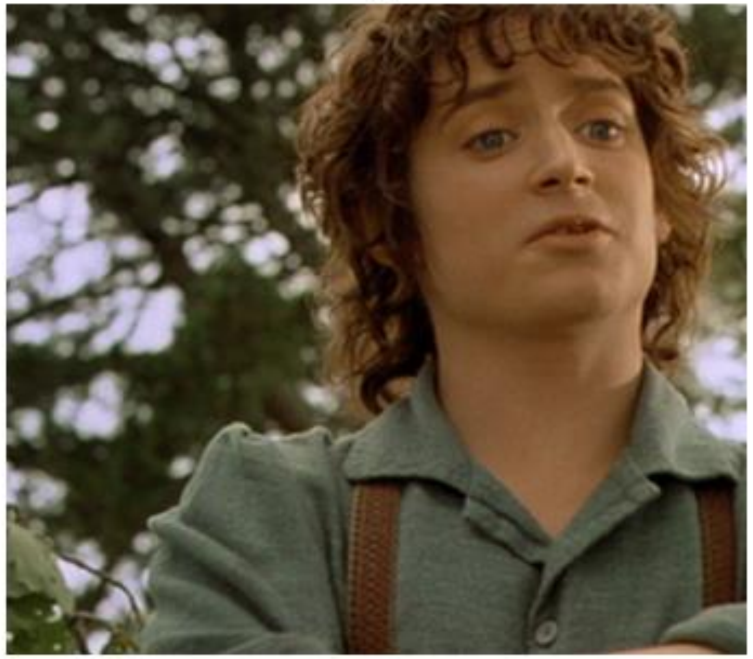
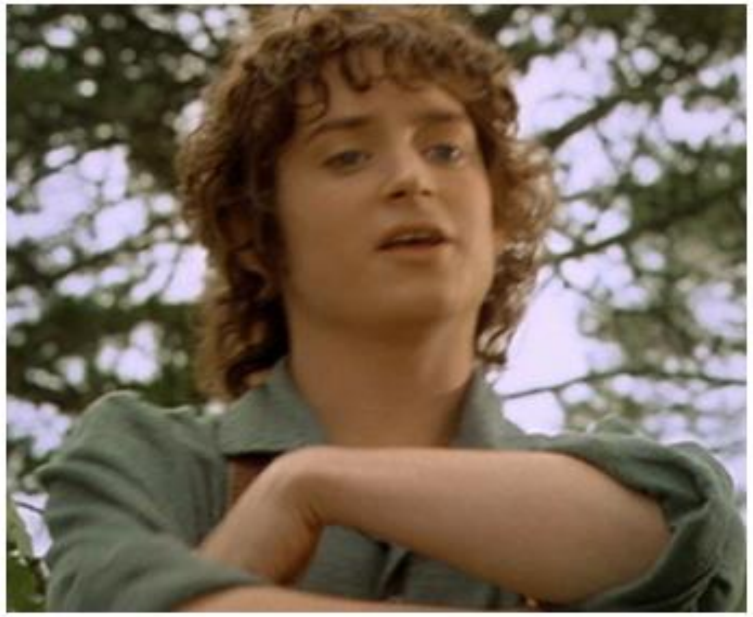
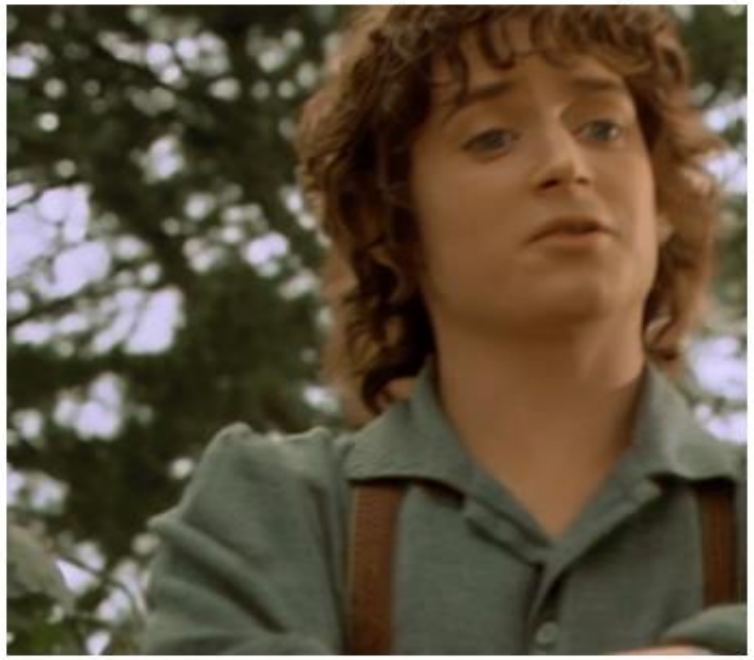


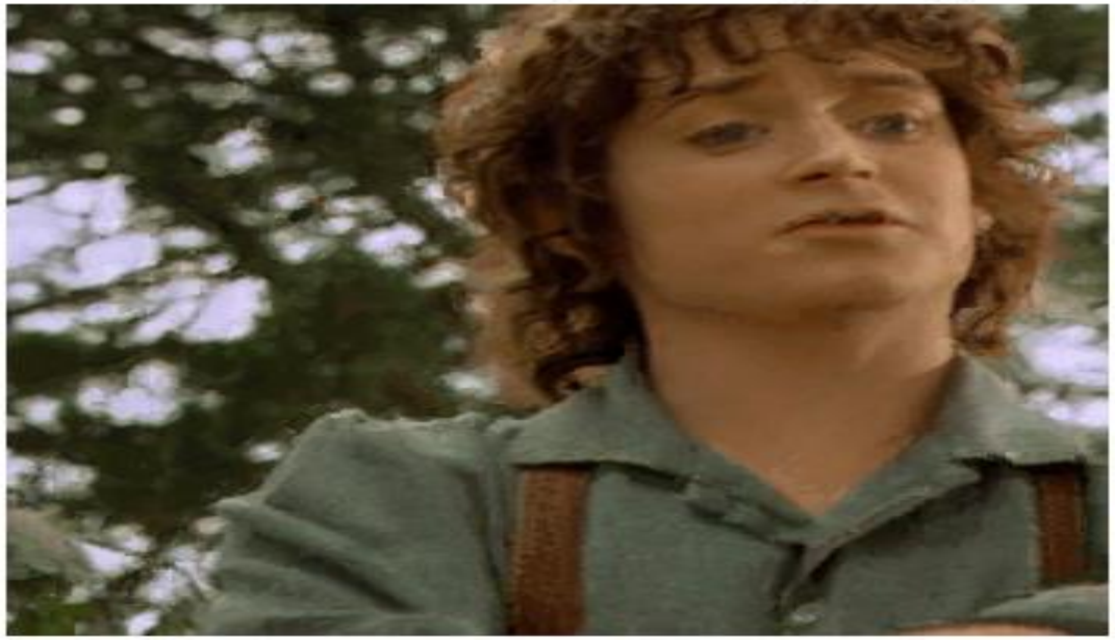
Image B



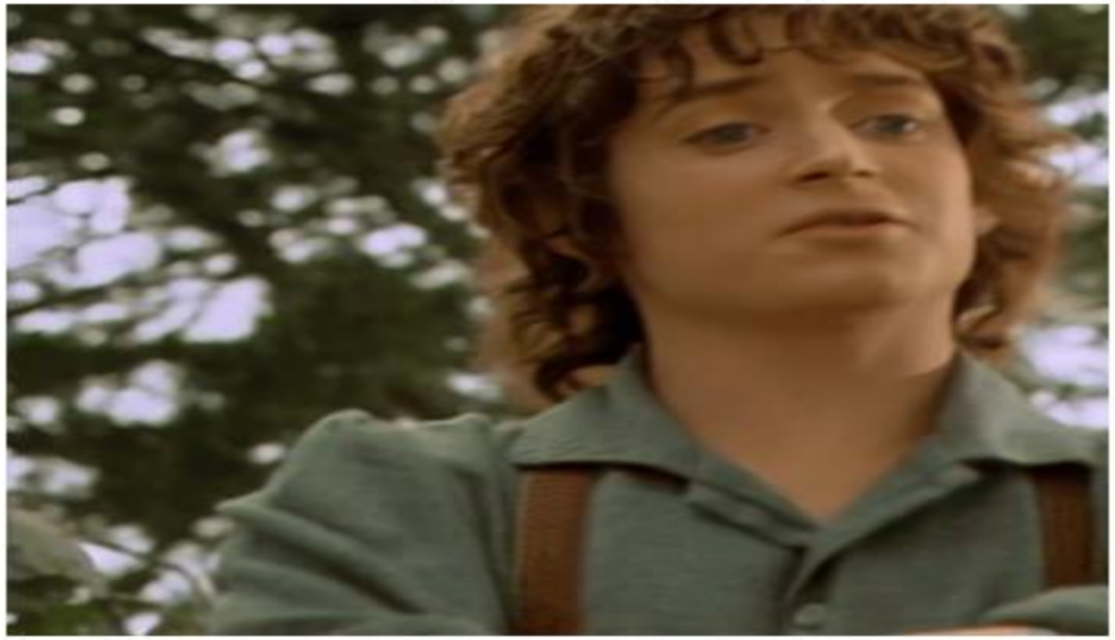
Reconstructed A using B and NNF(A to B)



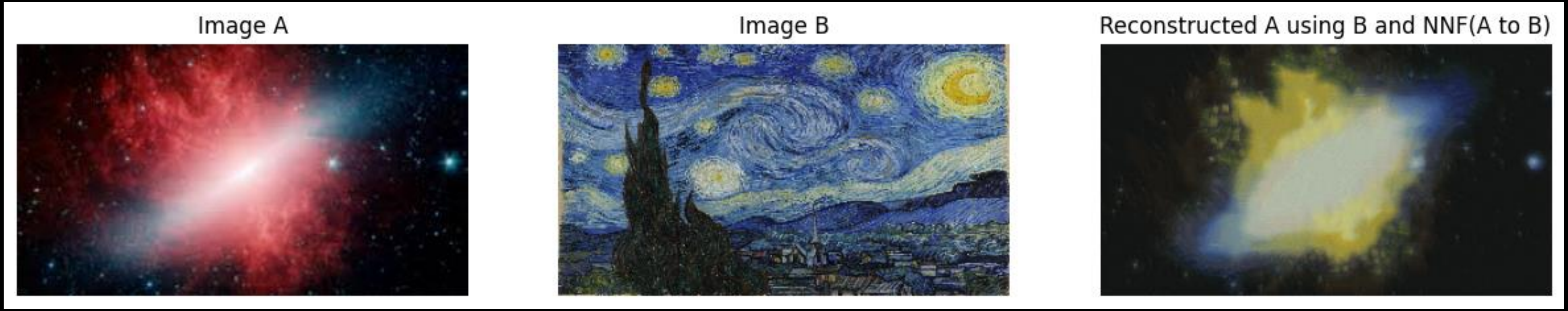
Reconstructed Image A using Image B



Reconstructed Image A using Image B (Average)



ARTISTIC STYLE TRANSFER



Texture Swapping

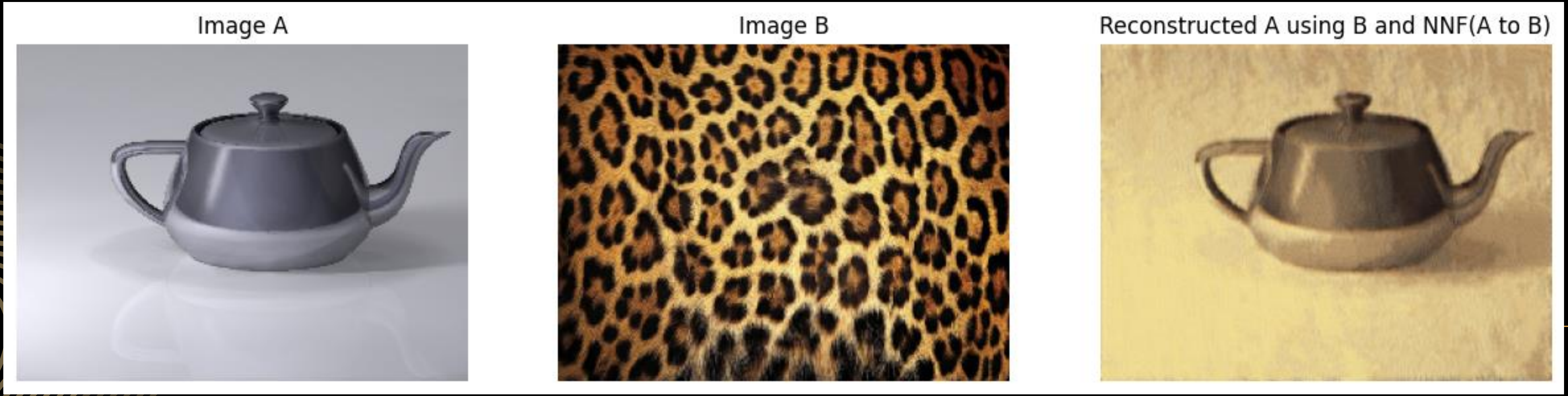


IMAGE RESHUFFLING

To reshuffle an image, copy the desired portion, paste it, apply a mask to remove the original object, and smooth edges.

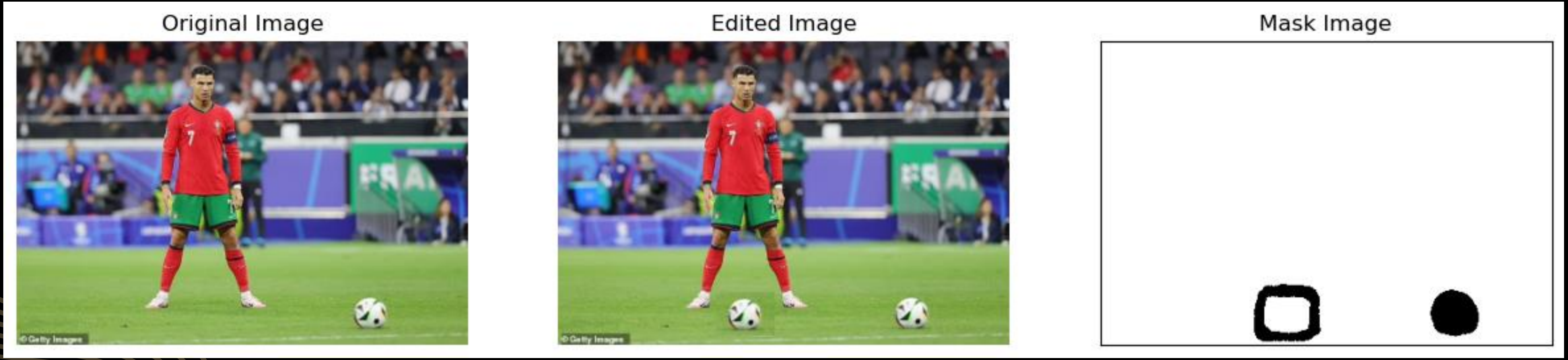
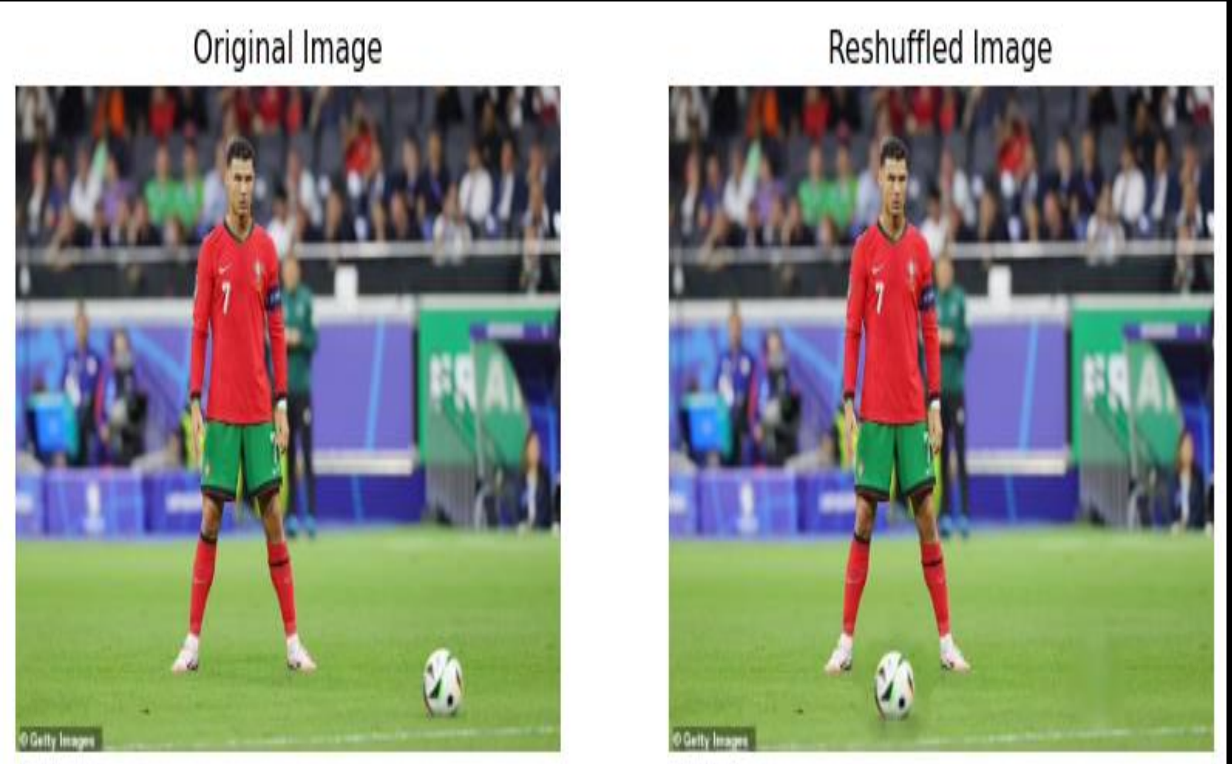
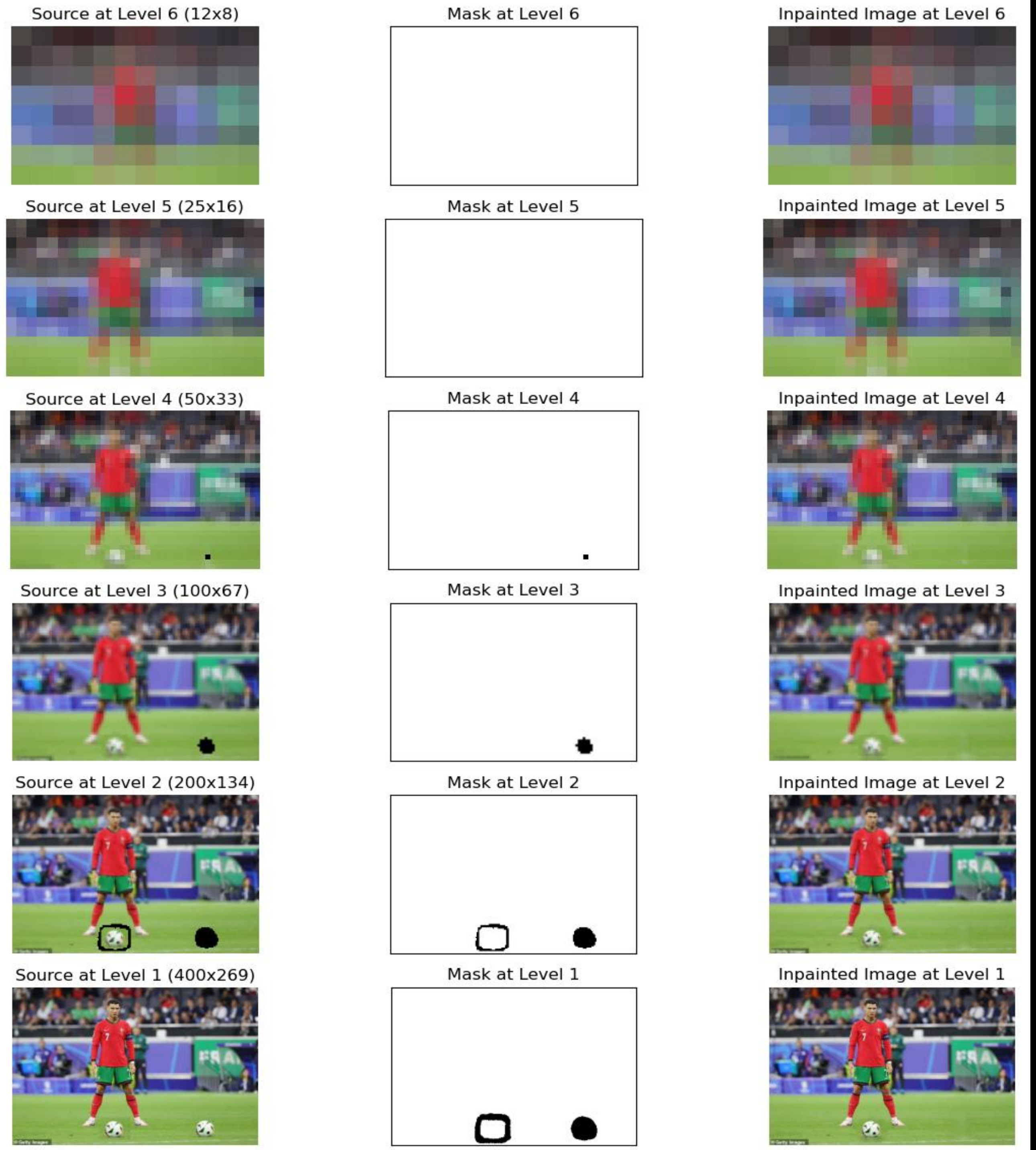


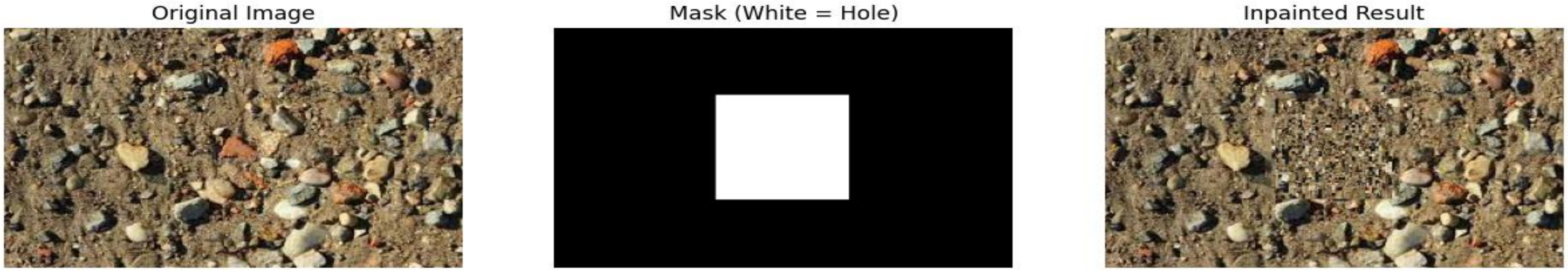
IMAGE RESHUFFLING



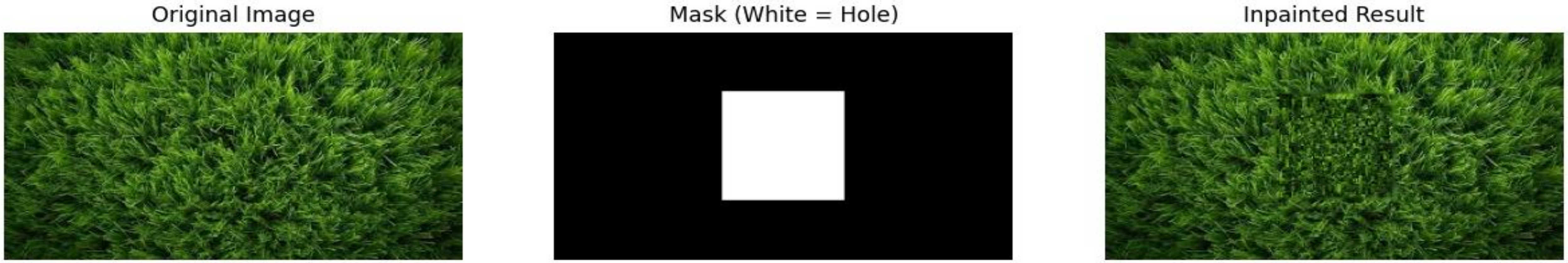

Source at Level 6 (12x8)
Mask at Level 6
Inpainted Image at Level 6
Source at Level 5 (25x16)
Mask at Level 5
Inpainted Image at Level 5
Source at Level 4 (50x33)
Mask at Level 4
Inpainted Image at Level 4
Source at Level 3 (100x67)
Mask at Level 3
Inpainted Image at Level 3
Source at Level 2 (200x134)
Mask at Level 2
Inpainted Image at Level 2
Source at Level 1 (400x269)
Mask at Level 1
Inpainted Image at Level 1

HOLE FILLING

Hole Filling for natural_sand.jpg



Hole Filling for grass.jpg



ABLATION STUDY

We compare the the effectiveness of the propagate and random search steps in the NNF computation for the task of image in-painting.

The three cases we test are:
a. Both Propagate and Random Search
b. Only Random Search (No propagate)
c. Only Propagate (No random search)

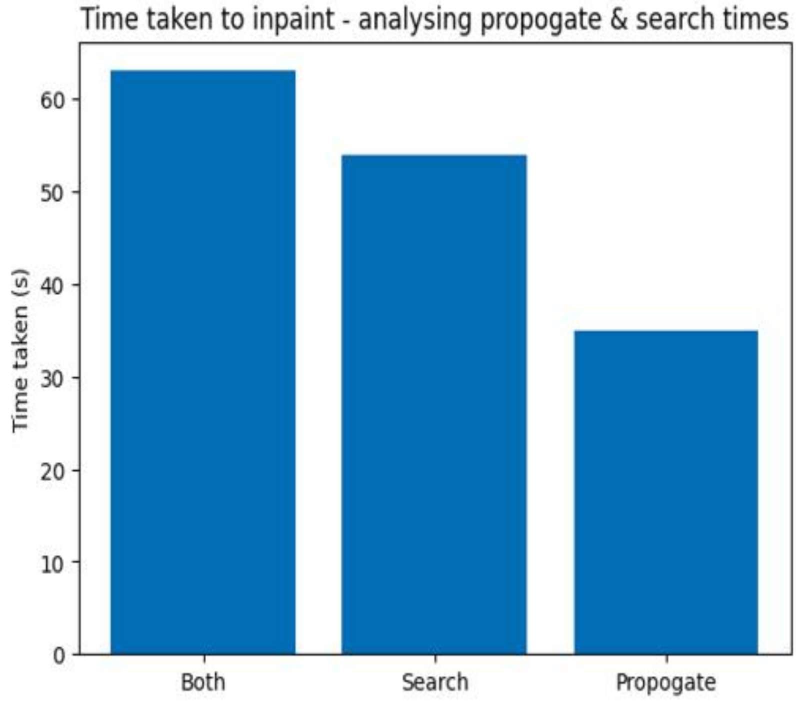
ABLATION STUDY

- The three output images show **only minor visual differences**, meaning both Propagation and Random Search work well independently.
- **Propagation alone is fastest** (35s), outperforming Random Search (54s) and the combined method (63s).
- Removing Random Search provides **efficient, high-quality results** while significantly reducing overall runtime.

Inpainting result - both propogate and search

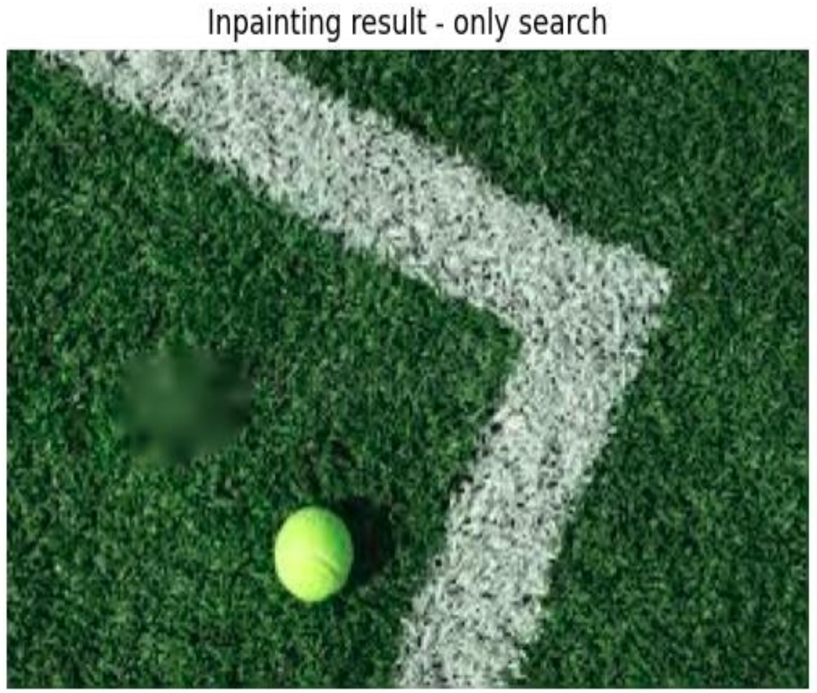
Inpainting result - only search

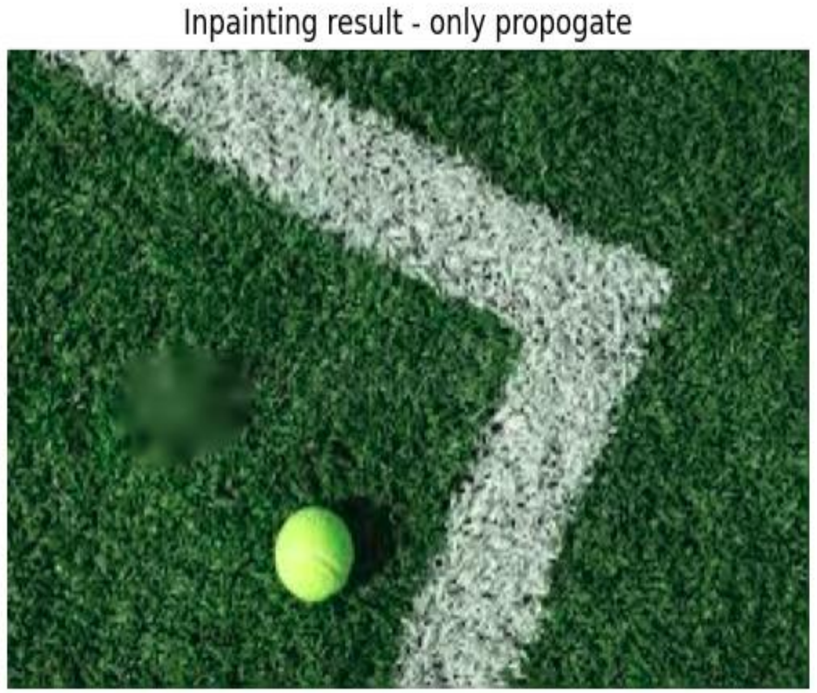
Inpainting result - only propogate

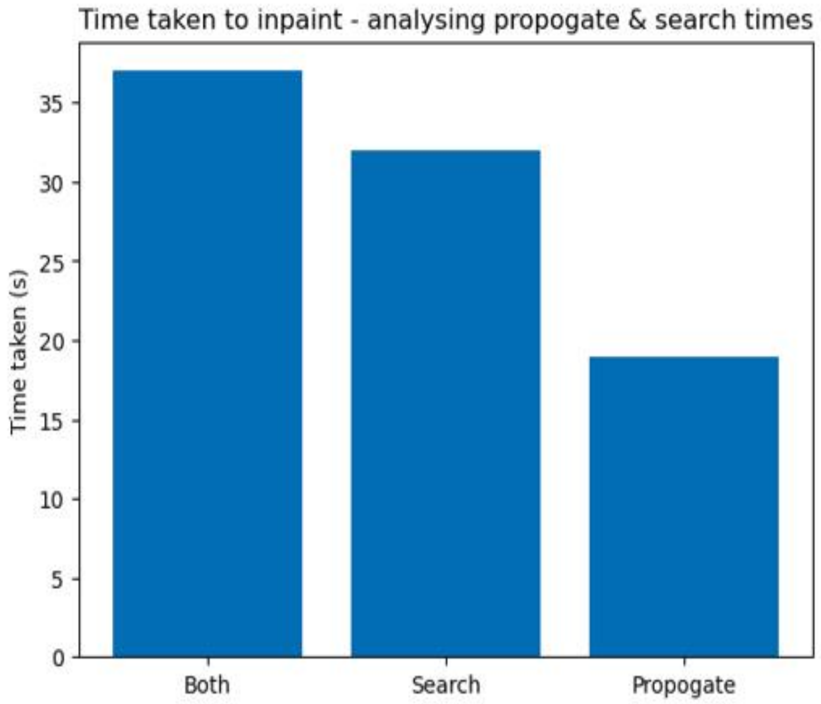


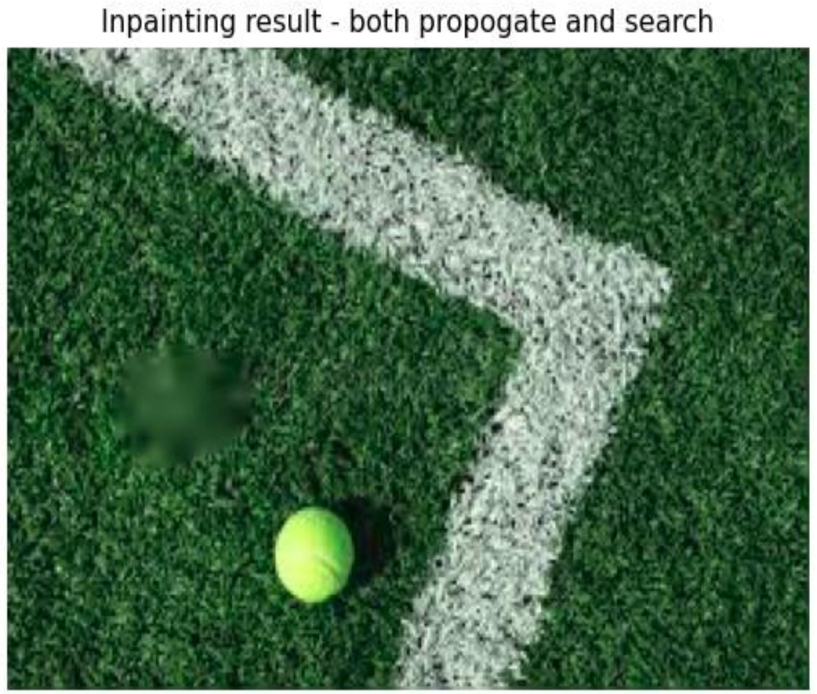
ABLATION STUDY

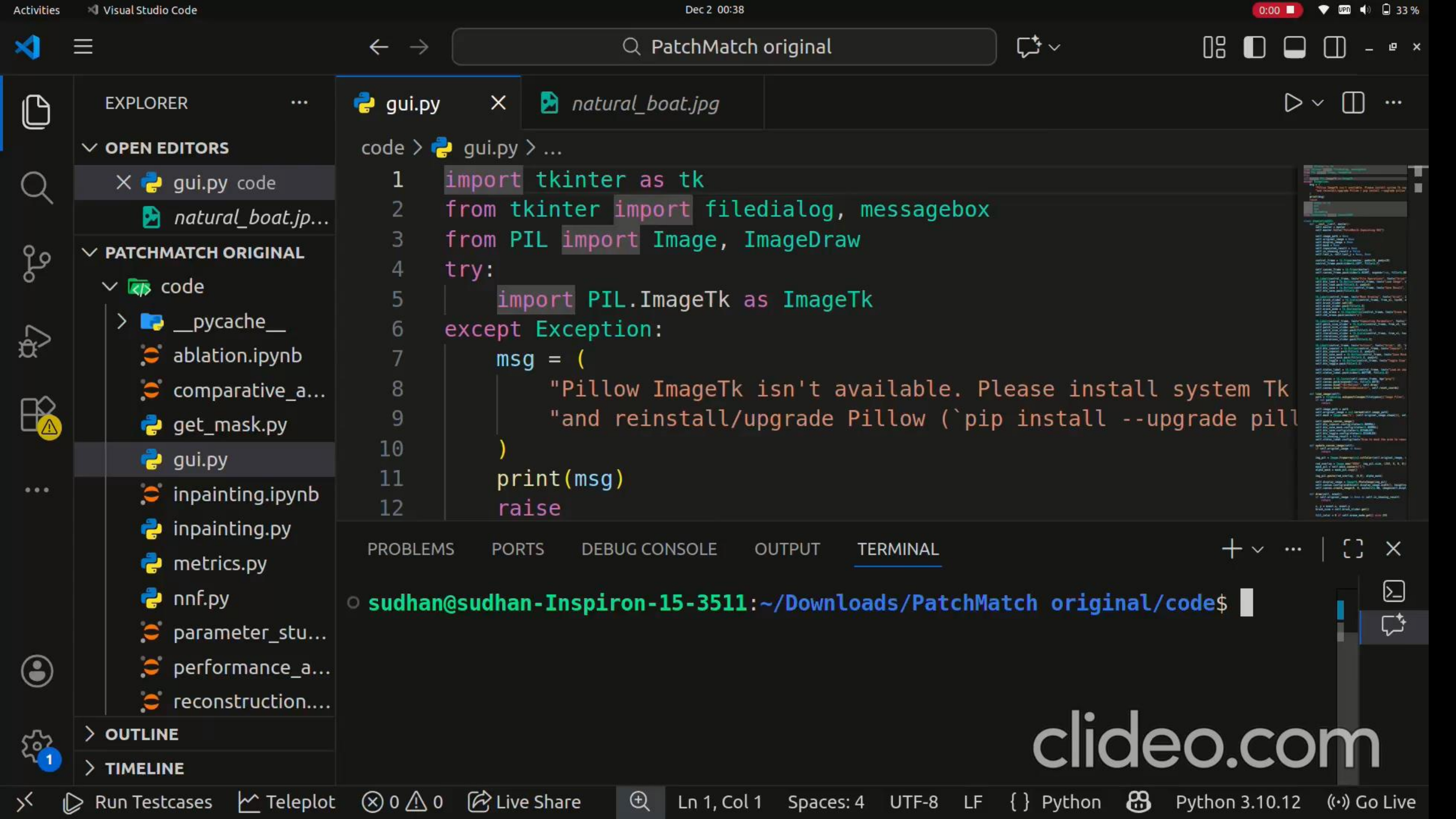
Testing another image confirms the
hypothesis - Propagate alone delivers
similar results while being the fastest
option, outperforming Random Search
and the combined approach in terms of
runtime efficiency









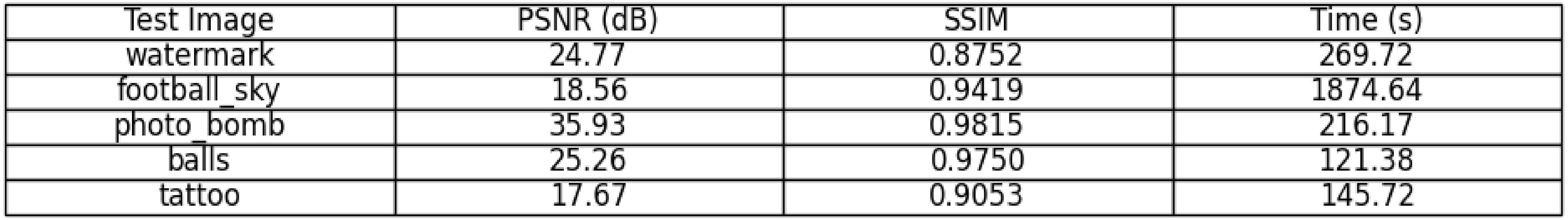


```python
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageDraw
try:
    import PIL.ImageTk as ImageTk
except Exception:
    msg = (
        "Pillow ImageTk isn't available. Please install system Tk
        "and reinstall/upgrade Pillow (`pip install --upgrade pill
    )
    print(msg)
    raise
```

```
sudhan@sudhan-Inspiron-15-3511:~/Downloads/PatchMatch original/code$
```

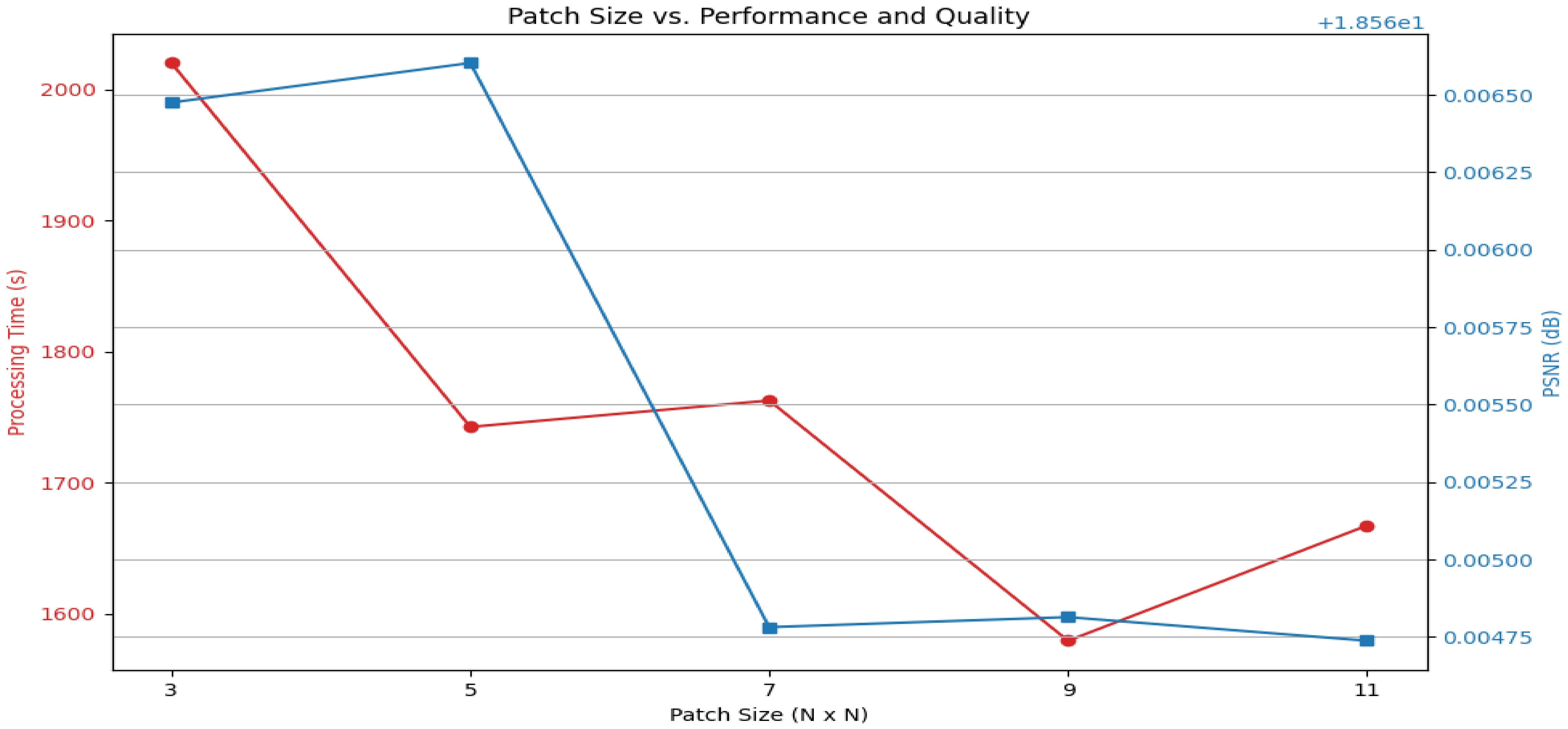
COMPARISION ANALYSIS

Quantitative Comparison Across Test Images

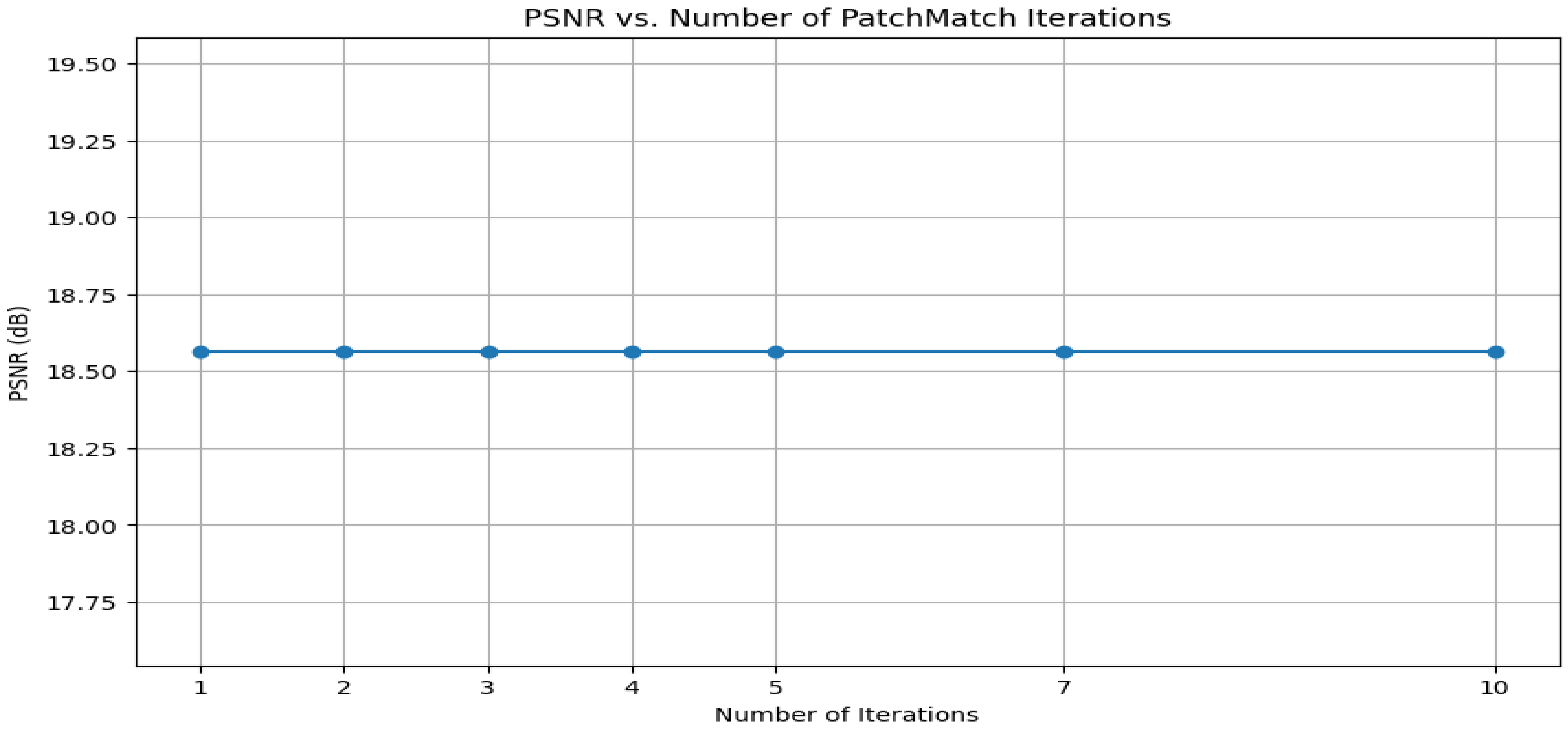


Test Image	PSNR (dB)	SSIM	Time (s)
watermark	24.77	0.8752	269.72
football_sky	18.56	0.9419	1874.64
photo_bomb	35.93	0.9815	216.17
balls	25.26	0.9750	121.38
tattoo	17.67	0.9053	145.72

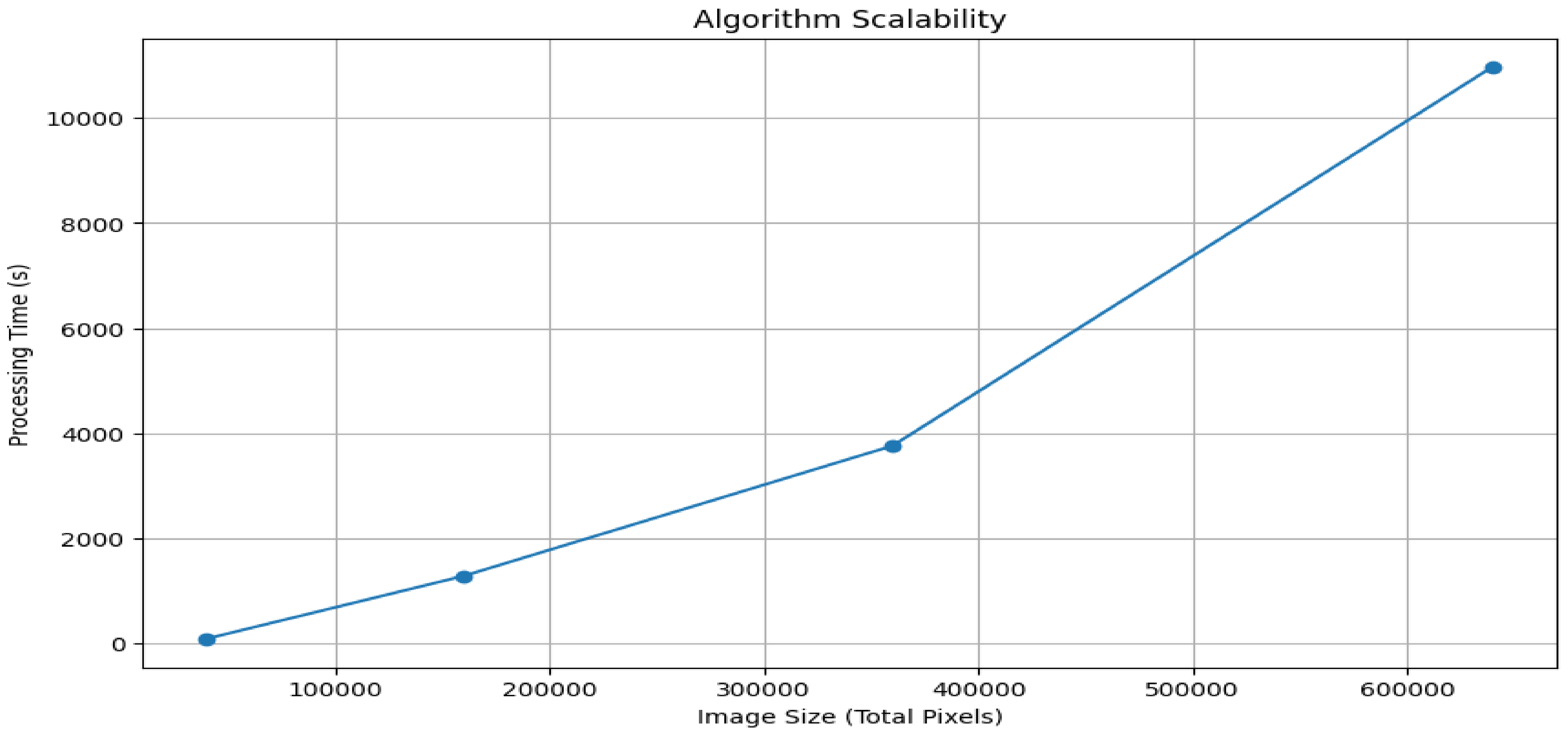
PARAMETER STUDIES



PARAMETER STUDIES



PERFORMANCE ANALYSIS



THANK YOU