DIP PROJECT

PATCHMATCH

Team cv2
Abhinav Raundhal - 2022101089
Gaurav Behera - 2022111004

01 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)

02 IMAGE DETAILS

COHERENCE

Coherence refers to the tendency of adjacent patches in an image to have similar nearest neighbours, meaning that good matches for one patch are likely to inform the matches for nearby patches.

PEAKED DISTRIBUTION

The peaked distribution implies that the best matches for a given patch are often located spatially close to its position in the source or target image.

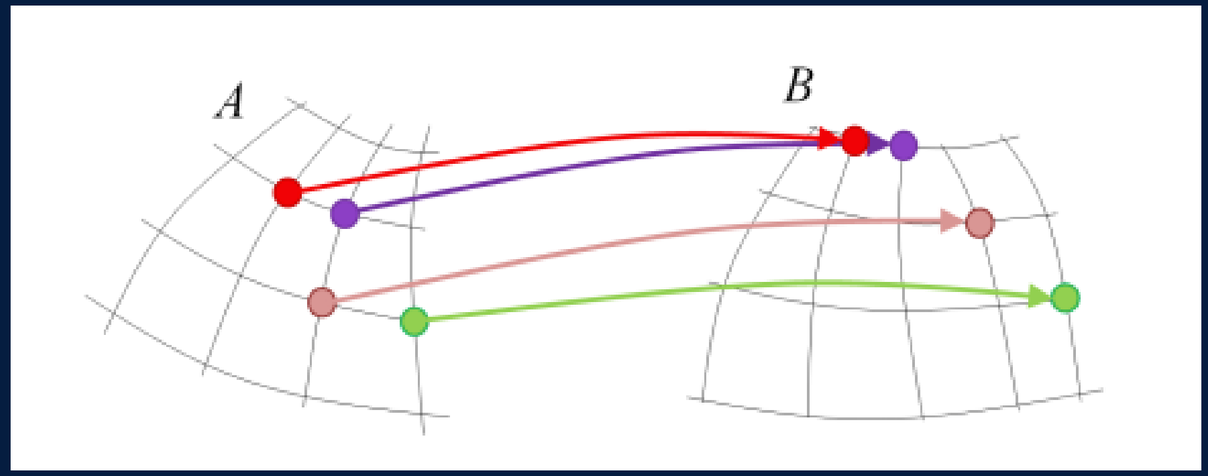
03

NEAREST NEIGHBOUR FIELD

- Nearest-Neighbour Field (NNF), which is a function that maps every patch in image A to a patch in image B, effectively specifying where the corresponding patch in B image is located.

$$f : A \rightarrow \mathbb{R}^2$$

- For a given patch at coordinate a in image A, the corresponding nearest patch in image B is located at coordinate b, and the NNF function is defined as f(a) = b.



04

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.

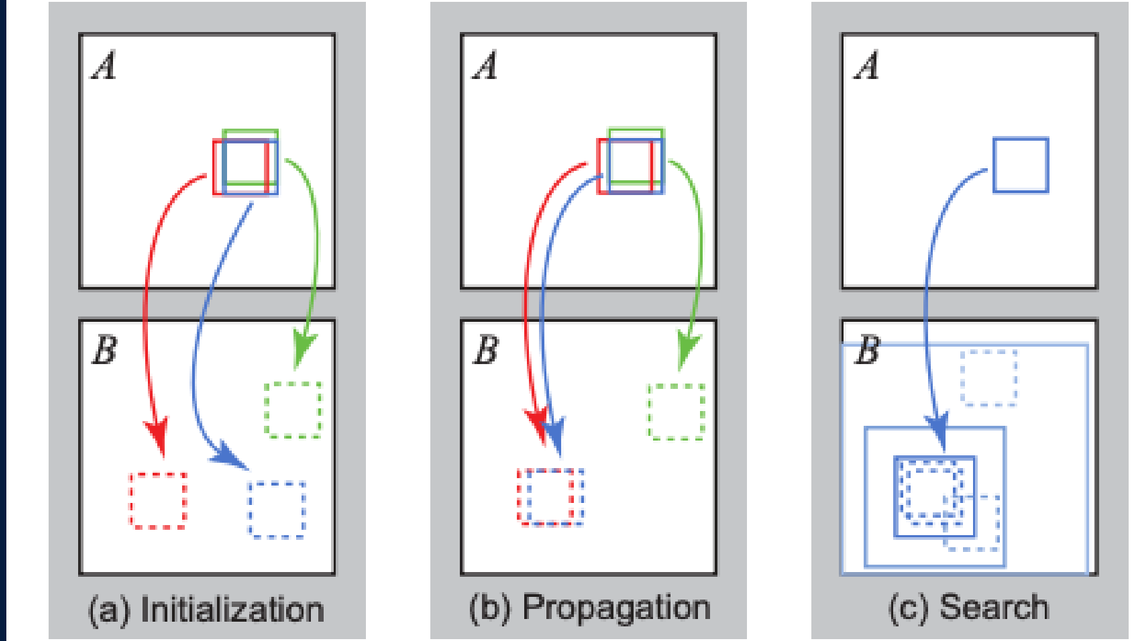
3. RANDOM SEARCH

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

04

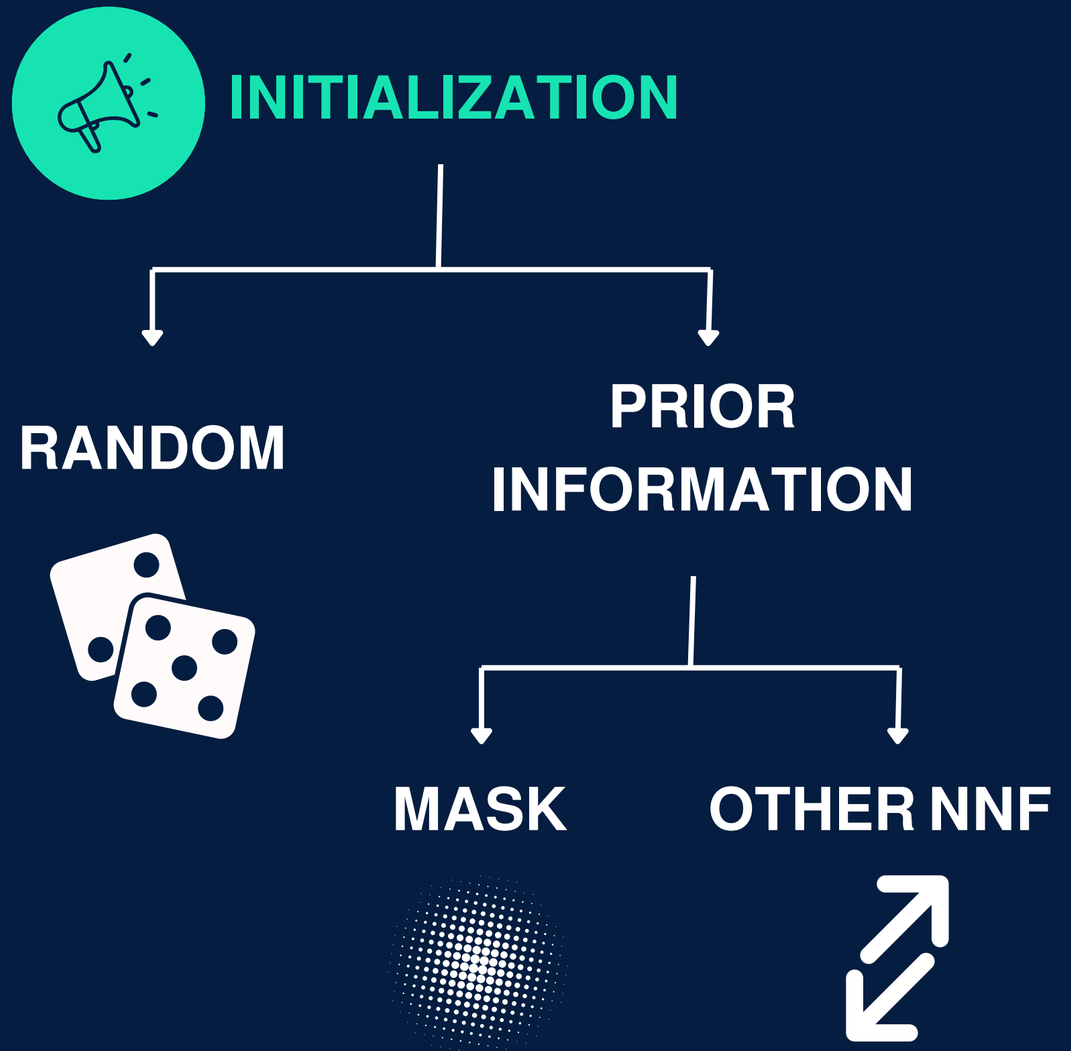
NNF ALGORITHM

ALGORITHM OVERVIEW



04

NNF ALGORITHM

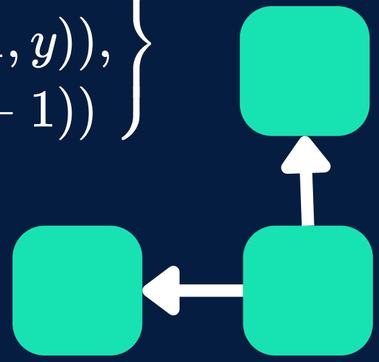


04

NNF ALGORITHM

PROPAGATION

$$f(x,y) = \arg\min_{v} \begin{Bmatrix} D(f(x,y)), \\ D(f(x-1,y)), \\ D(f(x,y-1)) \end{Bmatrix}$$



RANDOM SEARCH

$$u_i = v_0 + w\alpha^i R_i$$

04

NNF ALGORITHM

OTHER DETAILS

EVALUATION CRITERIA

Patch candidates are evaluated based on the l2-norm of RGB differences, with a penalty added if a mask is present.

HALTING CRITERIA

The process iterates 4-5 times until the nearest neighbour field (NNF) converges, with no significant improvement in assignments.

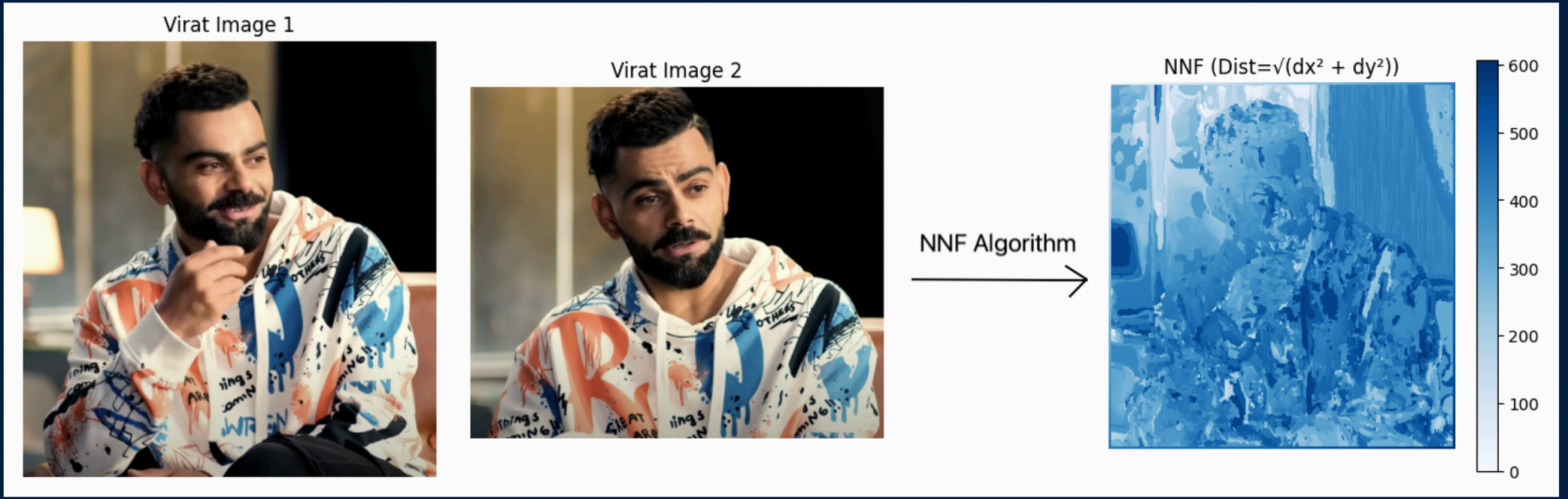
PATCH WIDTH

A patch width of 7 pixels is chosen, balancing texture capture and computational efficiency without excessive smoothing or missed details.

05 NNF RESULTS

1. NNF computation

Original input images A and B passed as inputs to the NNF algorithm. The algorithm outputs the NNF which stores the offsets from image A to B.



05 NNF RESULTS

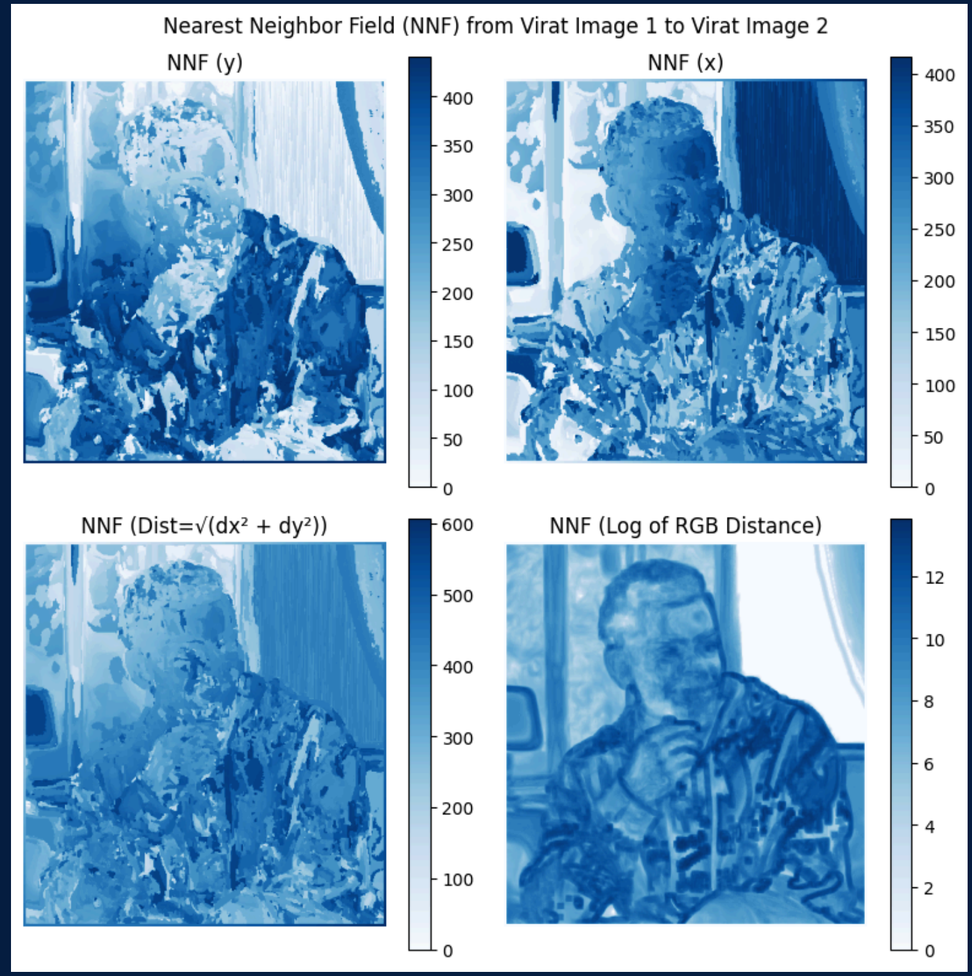
2. NNF Output

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)

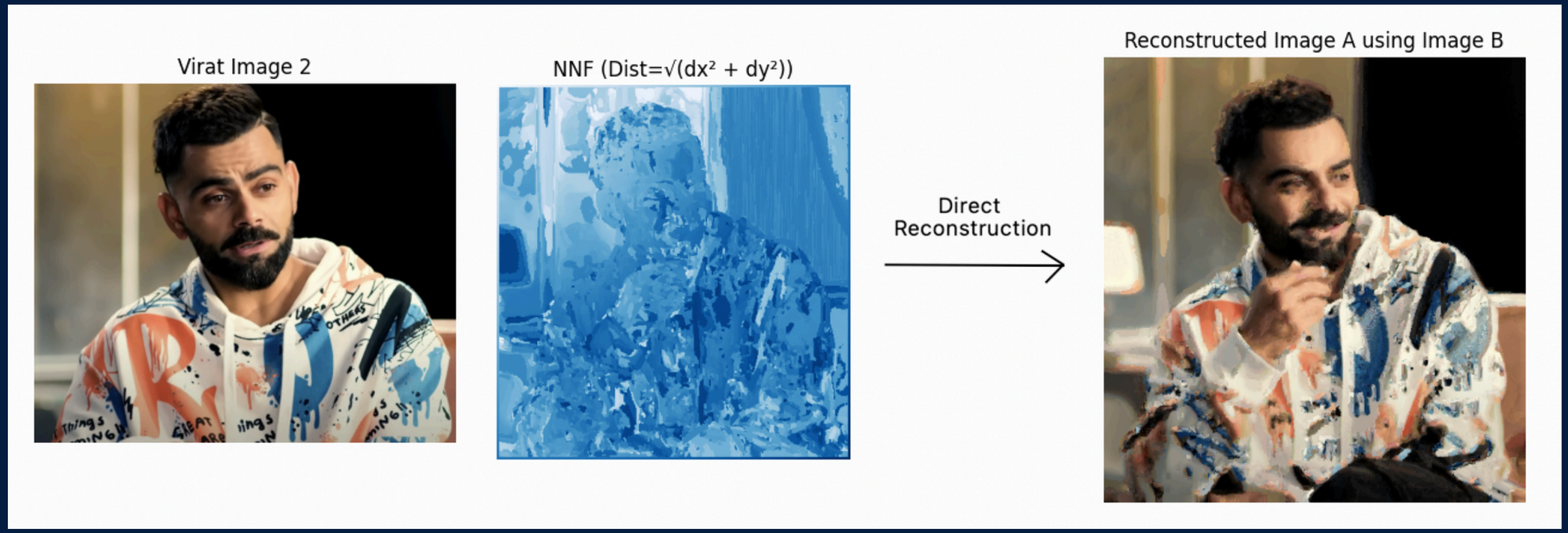
4. The NNF(RGB dist) represents the pixel intensity difference between each patch in A and B



05 NNF RESULTS

3. NNF reconstruction (Method 1)

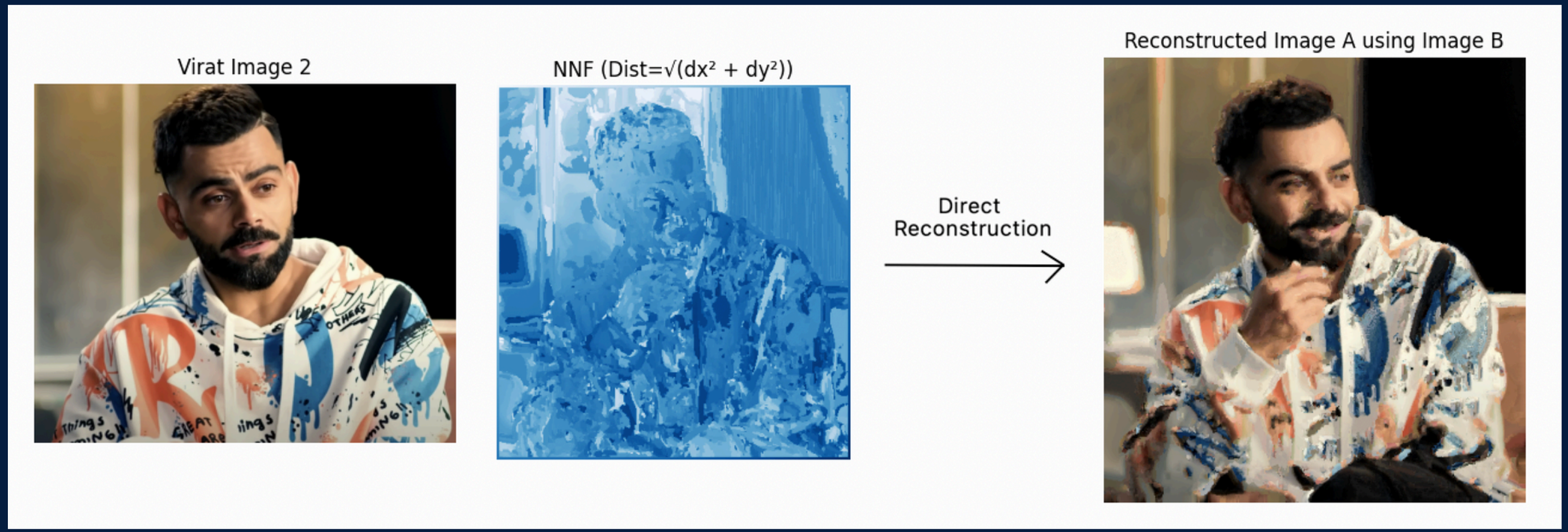
By directly replacing the pixel value with the centre of the NNF patch in B



05 NNF RESULTS

3. NNF reconstruction (Method 2)

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



06 IN-PAINTING

01 Initialisation

02 Pyramid Construction

03 EM Optimisation

04 Multi-Scale Coarse-to-Fine Refinement

06

IN-PAINTING ALGORITHM

1. INITIALISATION

- PatchMatch starts by initializing inputs: the image, mask, patch size, and maximum iterations.
- A binary mask confines in-painting to missing regions.
- Image pyramid for coarse-to-fine processing, addressing global details at lower resolutions and refining finer details at higher resolutions

06

IN-PAINTING ALGORITHM

2. PYRAMID CONSTRUCTION

- The image and mask are downsampled repeatedly until the smallest image approximates the patch size.
- Gaussian smoothing with a custom kernel reduces noise, ensuring smooth resolution transitions.
- Missing regions in the mask are propagated by weighted averaging.

06

IN-PAINTING ALGORITHM

3. EM OPTIMIZATION

- **E-Step**: Weighted contributions from nearby patches form "votes" for in-painted regions.

$$w = 1 - \frac{d(P_A, P_B)}{\text{MAX\_PATCH\_DIFF}}$$

- **M-Step**: Pixel values in missing regions are updated by averaging the weighted votes.

06

IN-PAINTING ALGORITHM

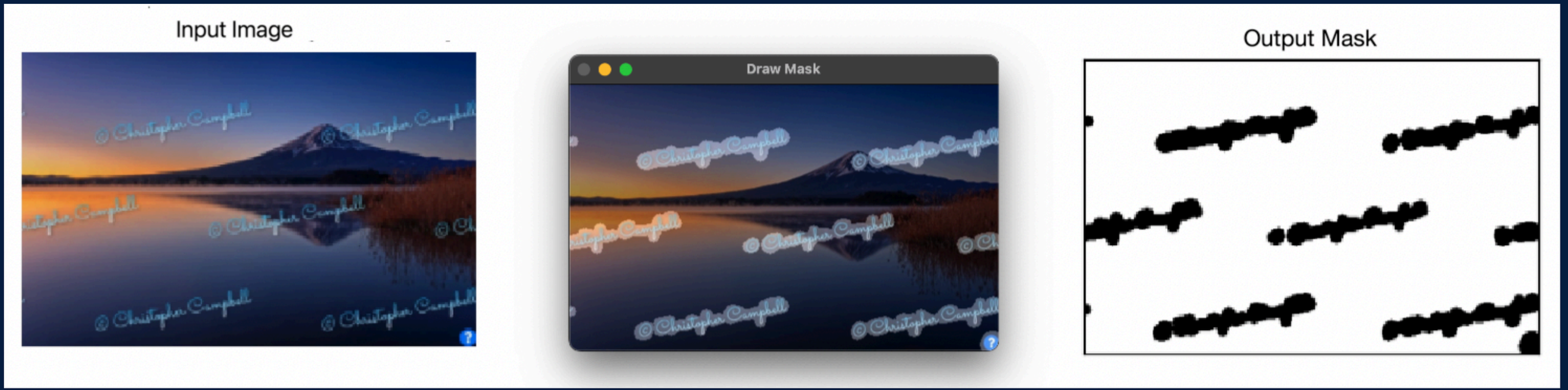
4. MULTI-SCALE COARSE-TO-FINE REFINEMENT

- **PatchMatch Initialisation**: Missing regions are initialised at the coarsest pyramid level.
- **Upsampling**: Results from lower levels are upsampled with bilinear interpolation to ensure smooth transitions.
- **NNF Propagation**: The NNF is propagated from coarser to finer levels, reducing computation and maintaining consistency.

07

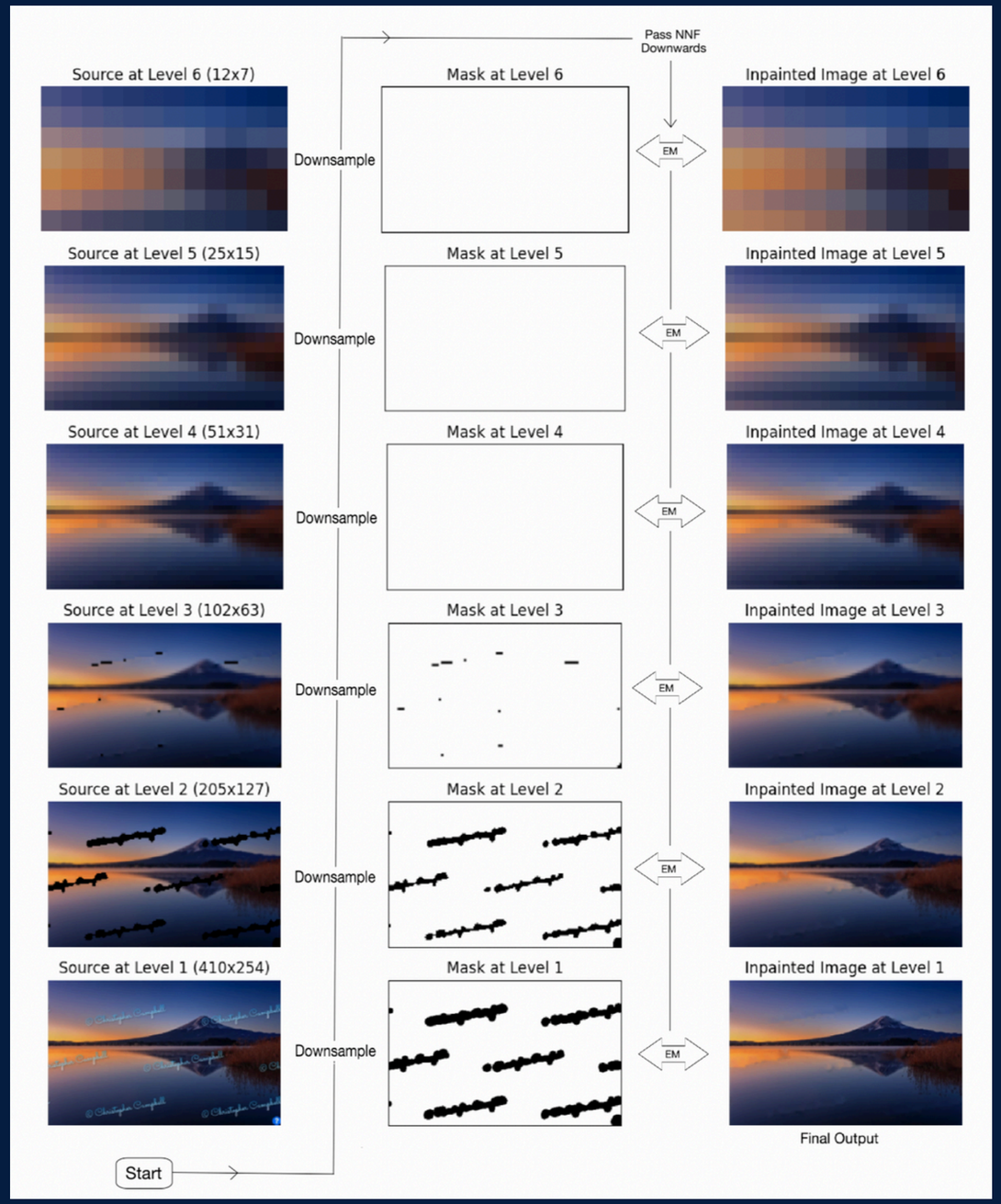
INPAINTING RESULTS

The mask of an input image is first created using our custom mask generating tool



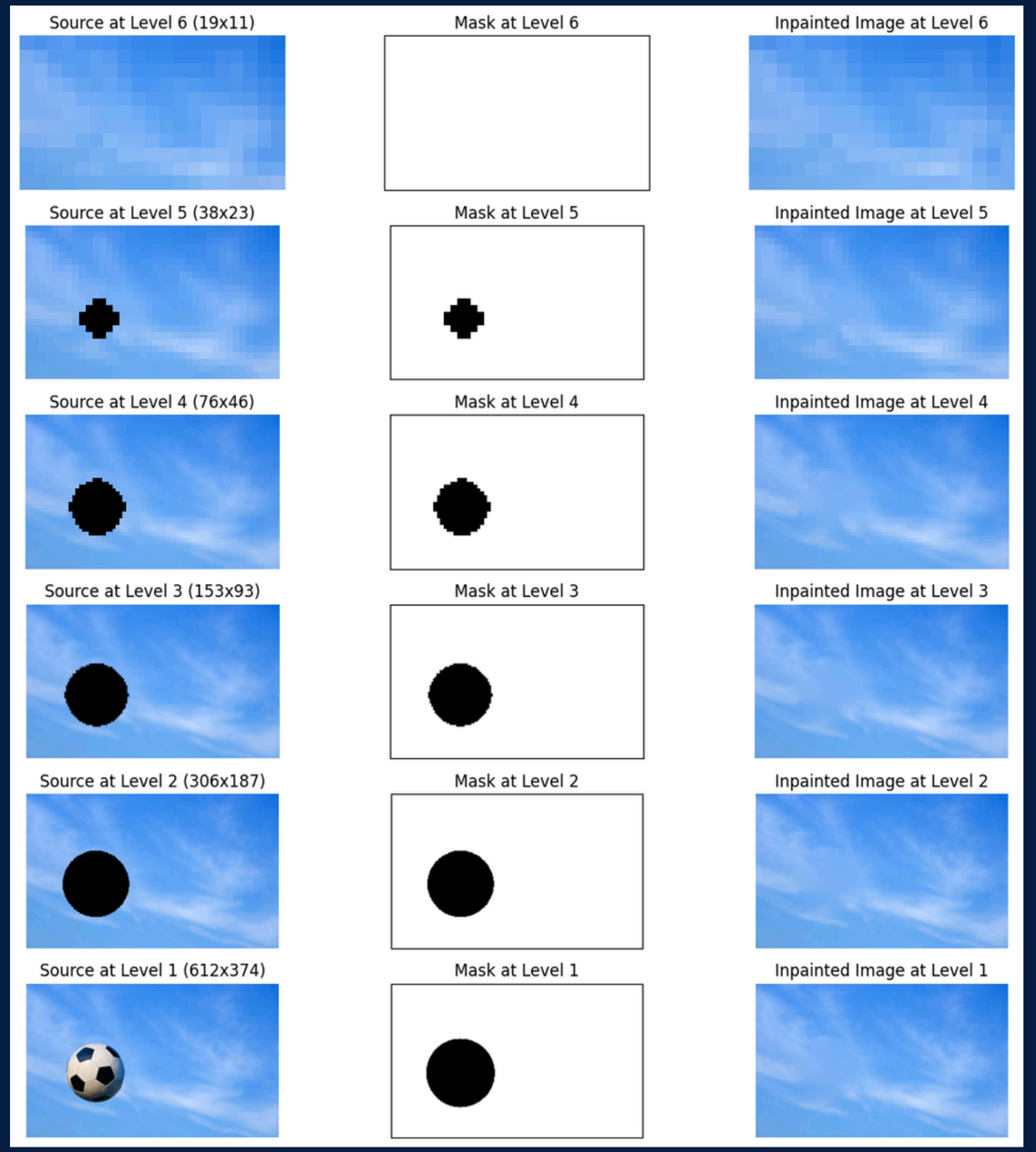
07

WATERMARK REMOVAL



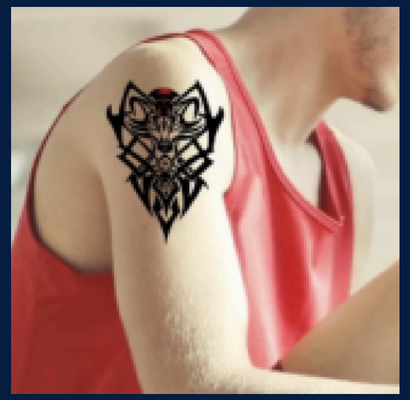
07

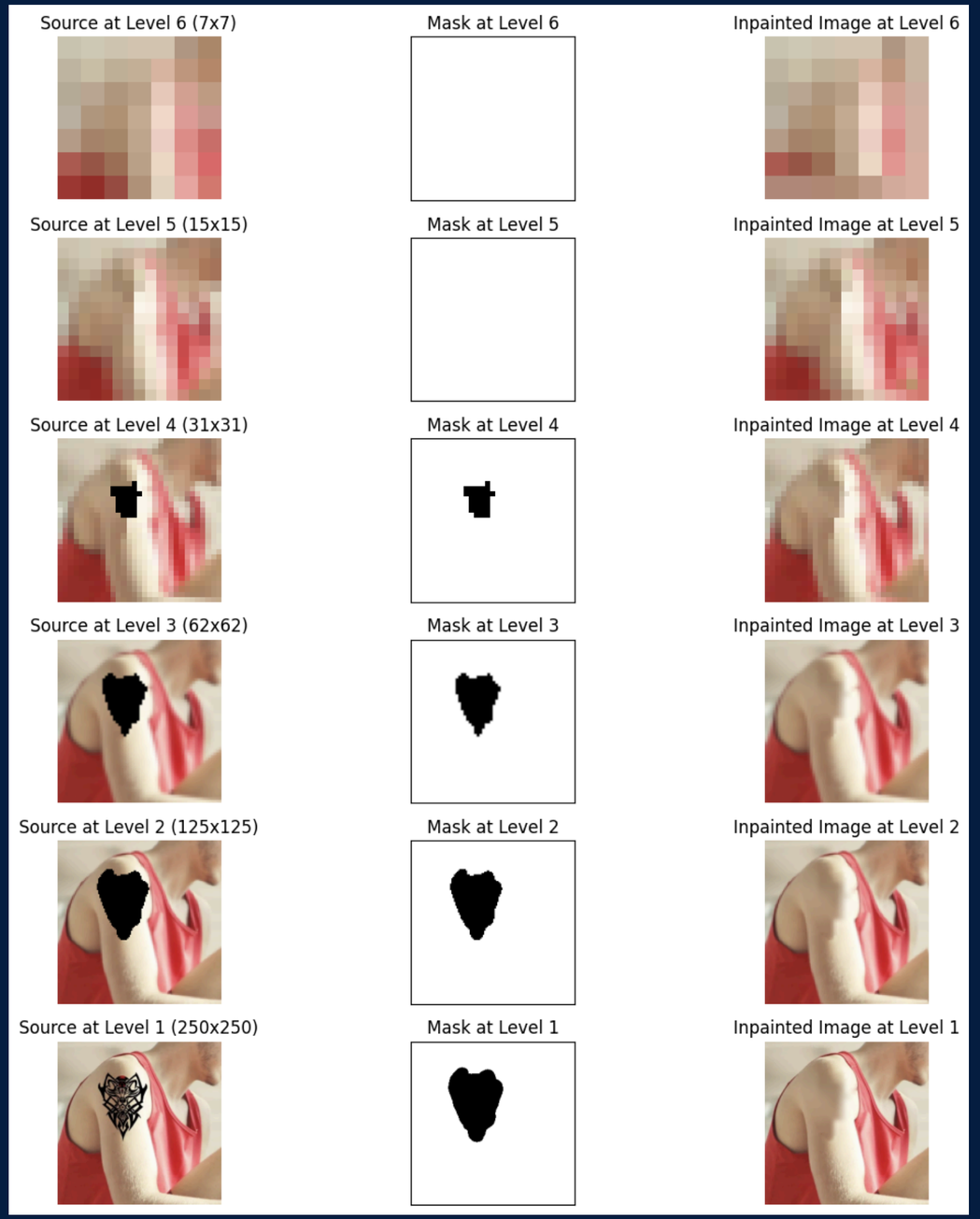
OBJECT REMOVAL



07

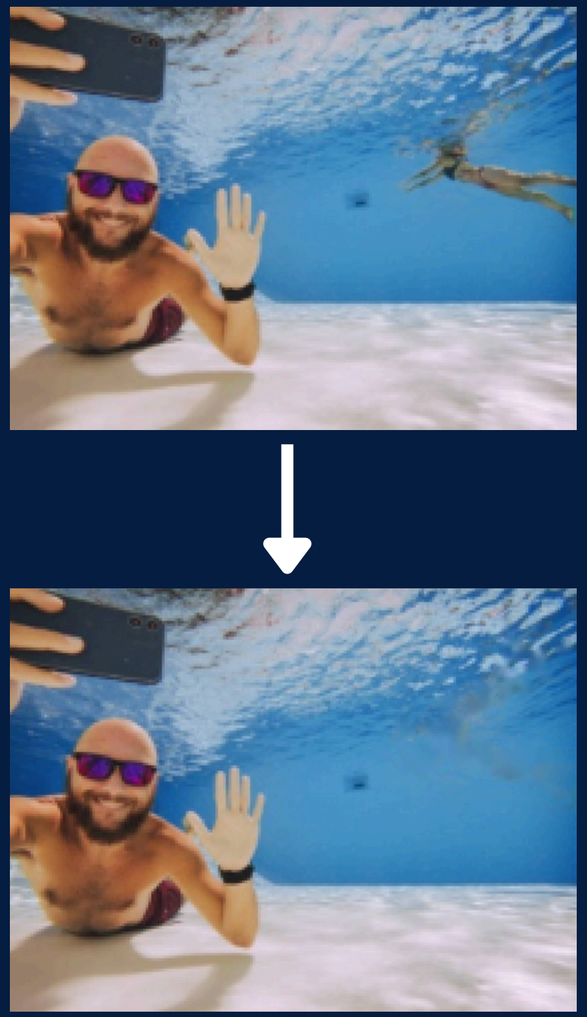
TATTOO REMOVAL

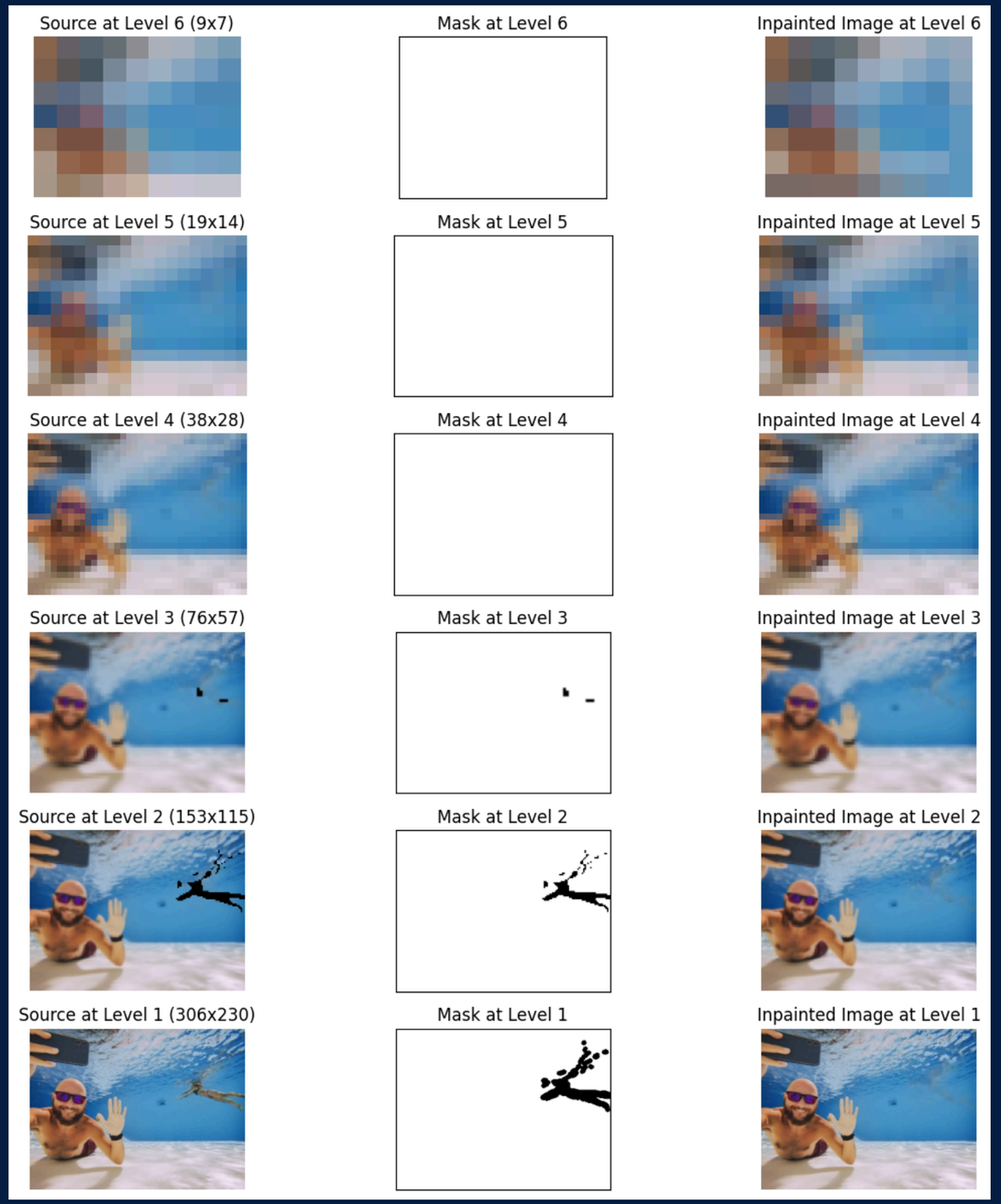




07

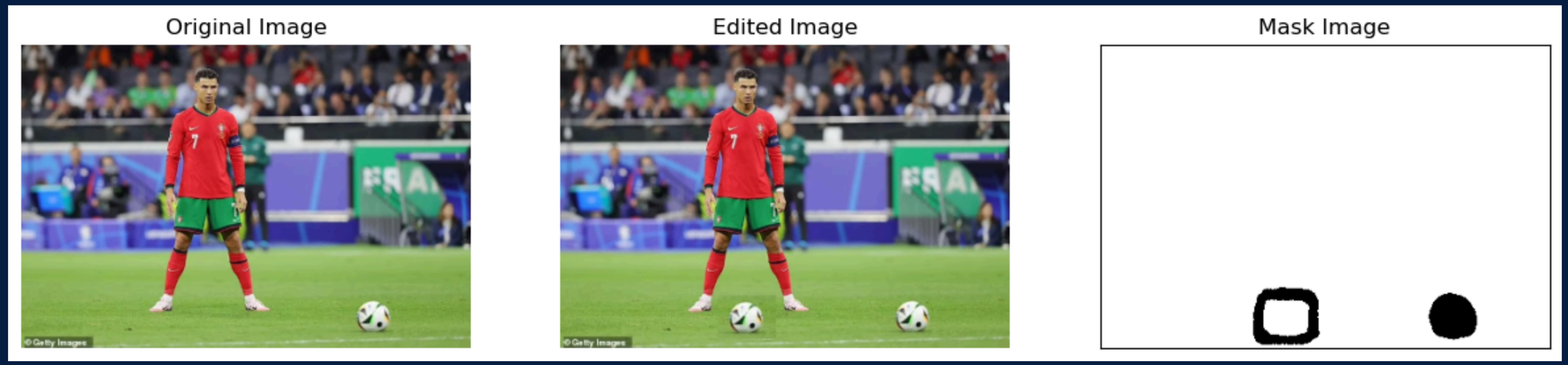
REMOVE PHOTO BOMB





08 IMAGE RESHUFFLING

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



08

RESHUFFLING

09

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)

09

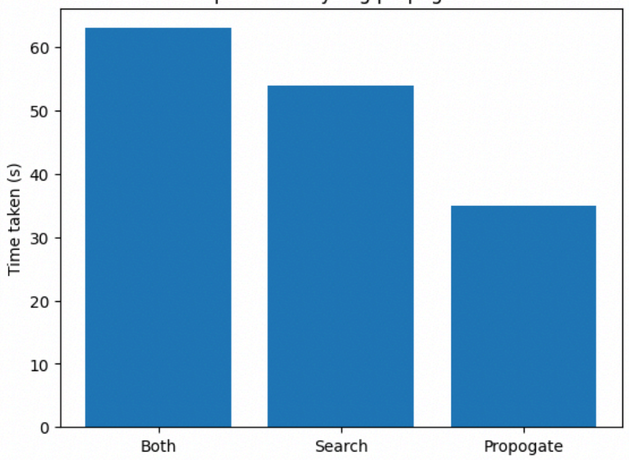
ABLATION STUDY

Inpainting result - both propogate and search

Inpainting result - only search

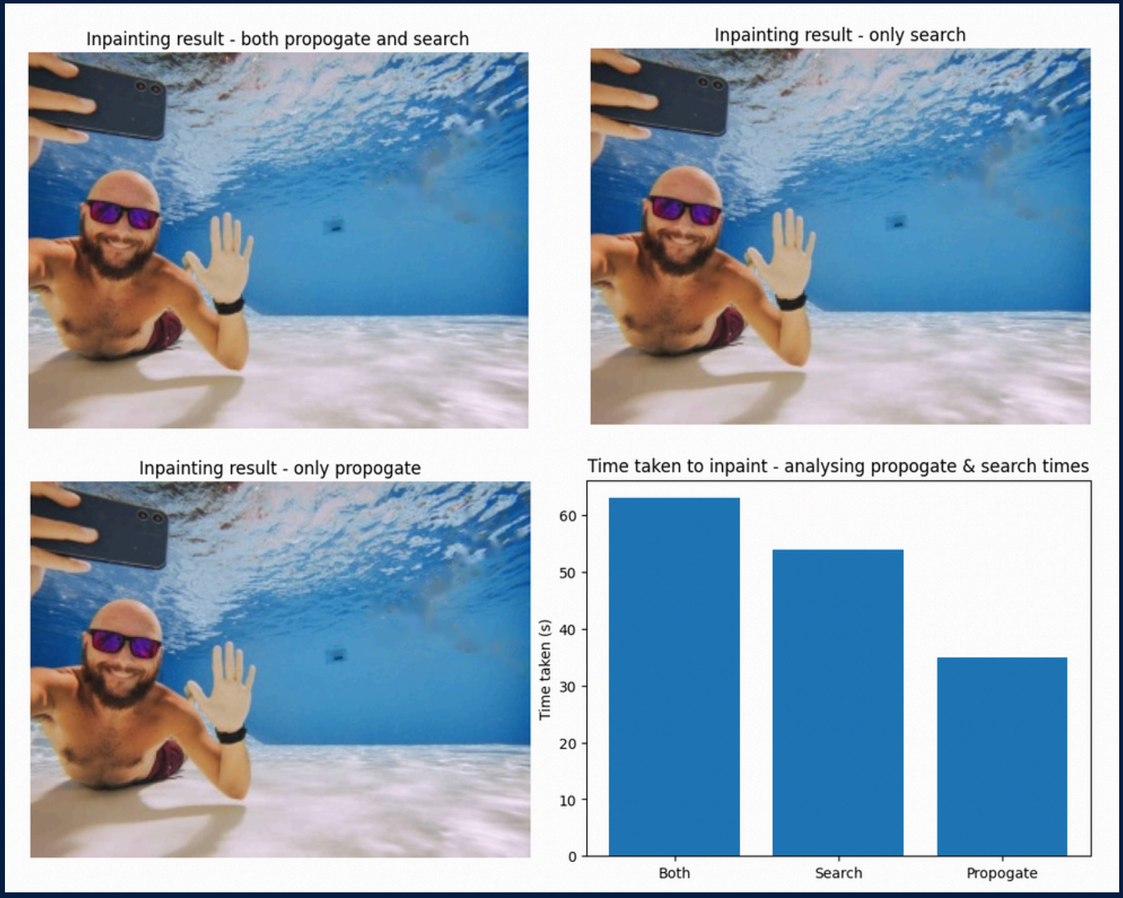
Inpainting result - only propogate

Time taken to inpaint - analysing propogate & search times



09

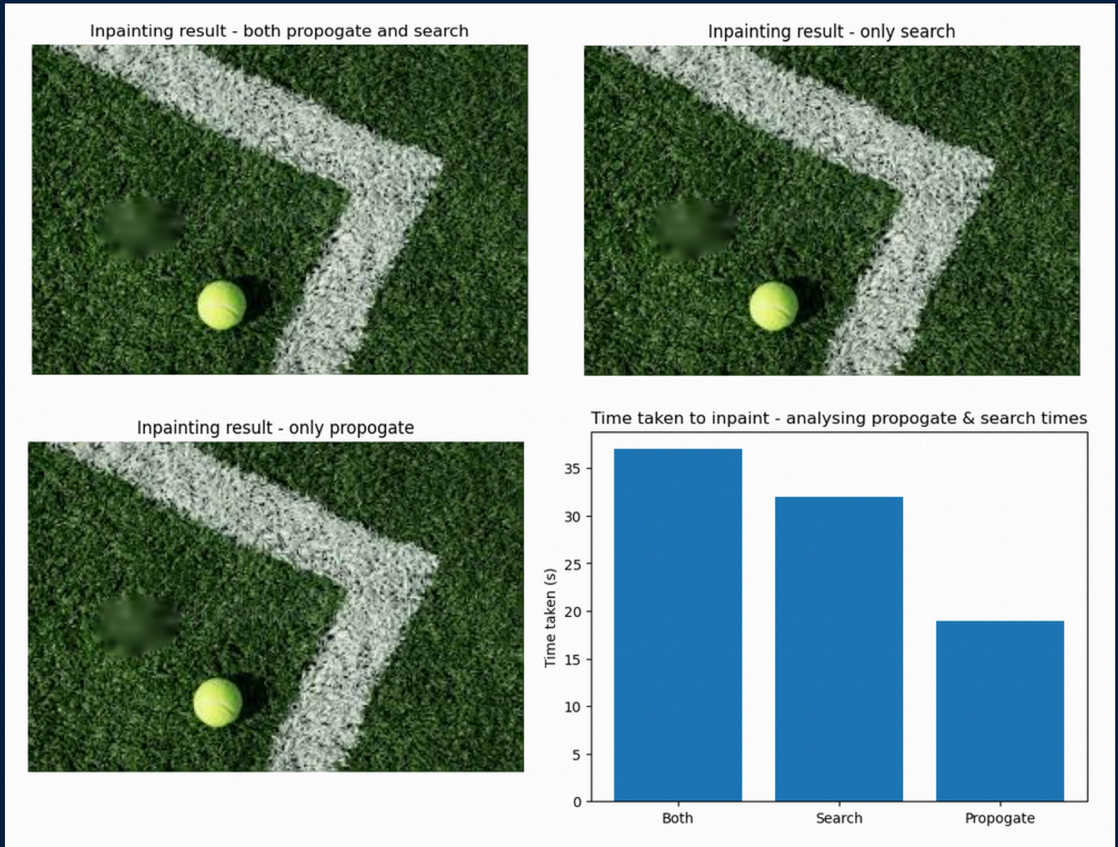
ABLATION STUDY



- The three images show minimal differences, indicating both steps independently produce good results.
- Propagation alone takes 35 seconds, outperforming Search (54 seconds) and both combined (63 seconds).
- Removing Random Search achieves efficient results, significantly reducing the algorithm’s runtime.

09

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.

THANK YOU

Team cv2