Computational Photography: The Art and Science of Digital Images

An exploration of algorithms that restore, enhance, and create visual realities.

Denoising

Inpainting

Cloning

Stylization

A Journey from Restoration to Creation

We will explore five powerful computational photography techniques, organized by their intent. This journey shows how algorithms evolve from fixing flaws in an image to **transcending the camera's physical limits,** and finally, to creating entirely new artistic expressions.

RESTORE

Fixing the imperfect.

Inpainting
Denoising

ENHANCE

Surpassing physical limits.

HDR Imaging
Seamless Cloning

CREATE

Reimagining reality.

Stylization

Erasing Imperfections with Inpainting

Original Image with Damaged Region

Result: Navier-Stokes (`INPAINT_NS`)

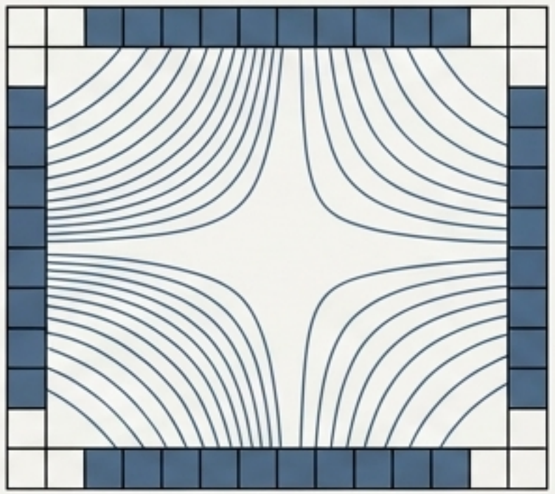
Result: Telea (`INPAINT_TELEA`)

Inpainting intelligently fills in missing or unwanted regions regions of an image by analyzing the surrounding content.

Two Approaches to Rebuilding an Image

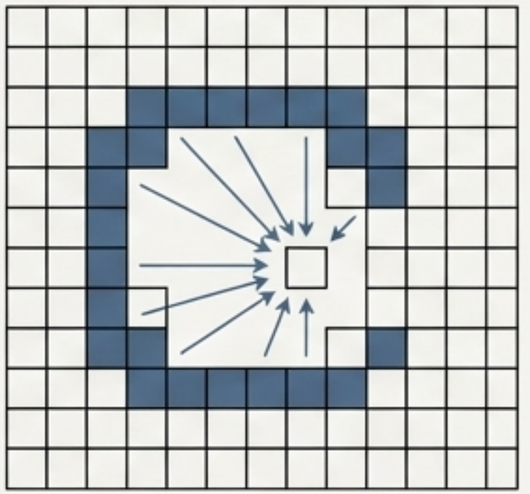
Navier-Stokes Method

Based on fluid dynamics. It propagates isophote lines (lines of equal intensity) smoothly into the missing region, like ink spreading in water.



Telea's Method

A Fast Marching Method. It works from the boundary inward, calculating each new pixel as a weighted average of its known neighbors based on distance and gradient direction.



```
# Create a mask for the region to be filled
mask = create_mask(image)

# Apply inpainting with a 3px neighborhood
result_ns = cv2.inpaint(image, mask, 3, cv2.INPAINT_NS)
result_telea = cv2.inpaint(image, mask, 3, cv2.INPAINT_TELEA)
```

Binary image defining area to repair

Flag for Navier-Stokes method

Flag for Telea's method

Finding Clarity in the Noise

Non-Local Means Denoising

Noisy Original

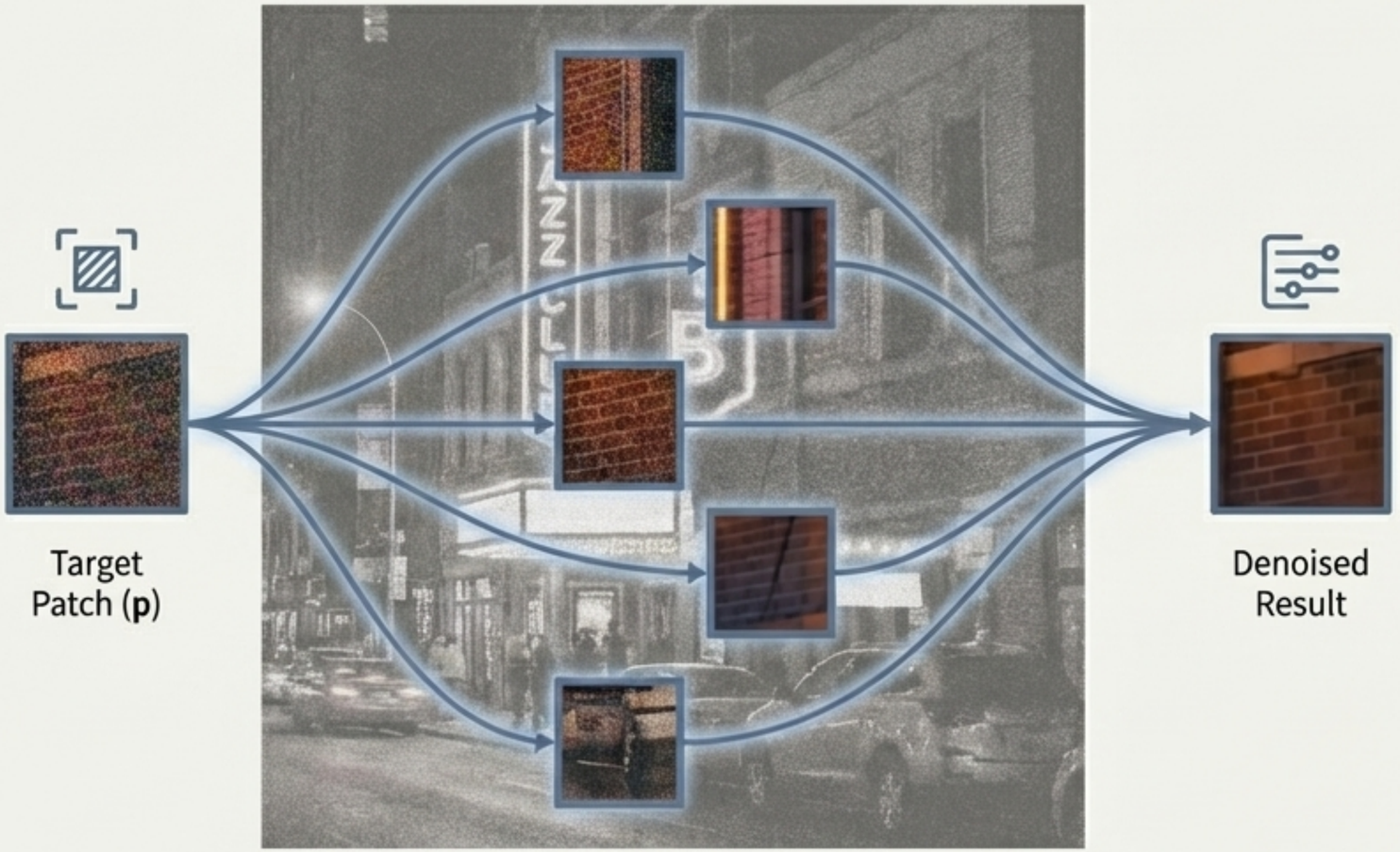
Source Serif Pro Regular

Denoised Result

Source Serif Pro Regular

Unlike traditional filters that average local pixels, Non-Local Means preserves sharp edges by finding and averaging similar patches from across the *entire* image.

The Power of Similar Patches



For every pixel, the algorithm computes a weighted average of all other pixels. The weight, w(p, q), is high if the patch around pixel q is very similar to the patch around pixel p.

$$w(p,q) = \exp\left(-\frac{\|\mathbf{P}(p) - \mathbf{P}(q)\|^2}{h^2}\right)$$

Parameter Tuning	
Parameter	**Effect of Increase**
h (Filter Strength)	More smoothing, potential detail loss
templateWindowSize	Larger patches, slower, more robust
searchWindowSize	Larger search area, slower, higher quality

Capturing Light Beyond the Sensor’s Range

High Dynamic Range (HDR) Imaging

Underexposed
Source Serif Pro Regular

Normal Exposure

Overexposed

Final HDR Result

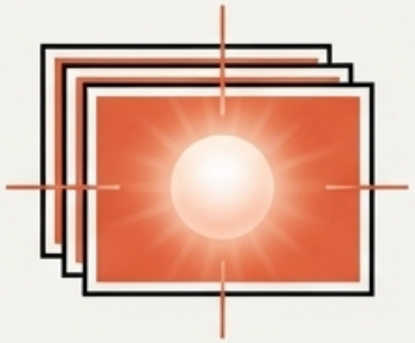
HDR combines multiple exposures of the same scene to create a single image containing the full range of light and shadow, from the darkest blacks to the brightest whites.

The HDR Pipeline: From Capture to Display

CAPTURE

Take multiple photos at different exposure times (Δt).

ALIGN & MERGE



Align images to correct for motion. Recover the true scene radiance (E) using the Camera Response Function ($g(Z) = \ln(E) + \ln(\Delta t)$) and merge into one 32-bit HDR image. Methods: Debevec, Robertson.

TONEMAP

The HDR image has a dynamic range too large for screens. Tone mapping compresses this range back into a viewable 8-bit image.

DISPLAY

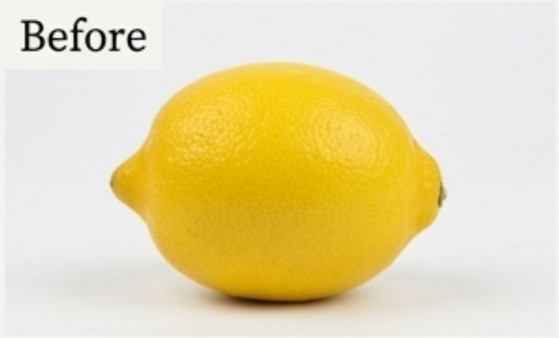
The final LDR (Low Dynamic Range) image.

```
# Merge exposures into an HDR image
merge_debevec = cv2.createMergeDebevec()
hdr = merge_debevec.process(images, times=exposure_times)

# Tonemap the HDR image
tonemap = cv2.createTonemapDrago()
ldr = tonemap.process(hdr)
```

Blending Realities with Seamless Cloning

Before



Source

Destination

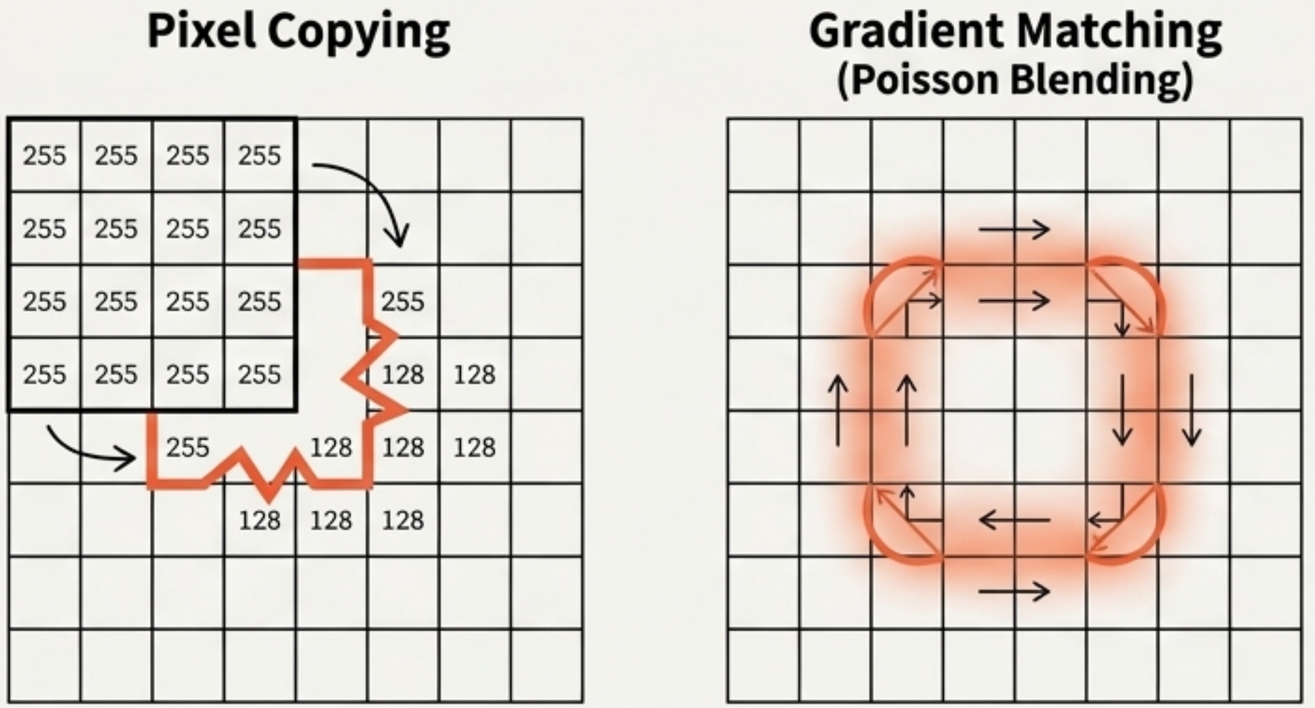
Naive Copy-Paste

Result: Seamless Clone

Result: Seamless Clone

Instead of just copying pixels, seamless cloning intelligently adjusts an object's color and texture to match its new surroundings, creating a believable composite.

The Core Idea: Paste Gradients, Not Pixels



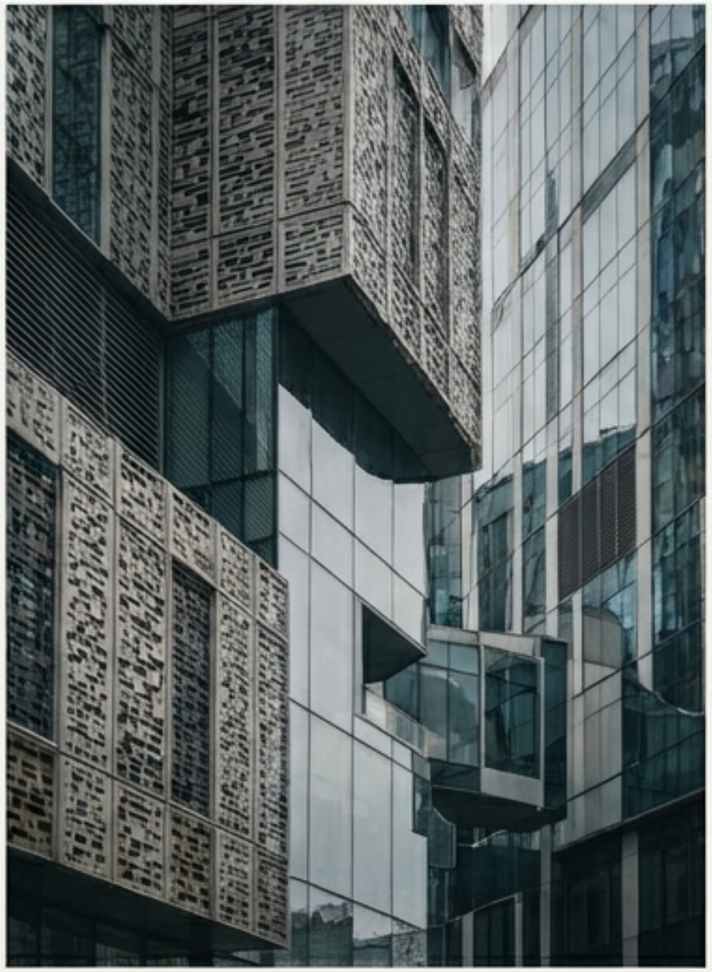
The algorithm solves the Poisson Equation ($\Delta f = \text{div}(\mathbf{v})$), which aims to find an image f whose gradients ∇f are as close as possible to the source image's gradients $\mathbf{v}$ within the cloned region Ω, while ensuring the boundary $\partial\Omega$ matches the destination image.

Cloning Modes

- NORMAL_CLONE: Standard blending.
- MIXED_CLONE: More transparent effect; uses the stronger gradient from either source or destination at each point.

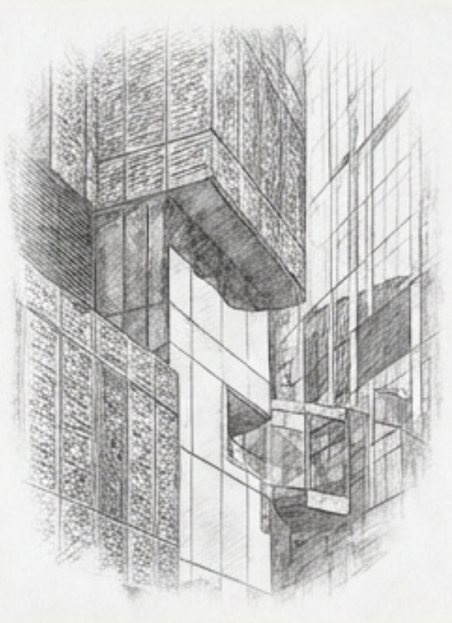
Beyond Reality: Image Stylization

Original Photograph

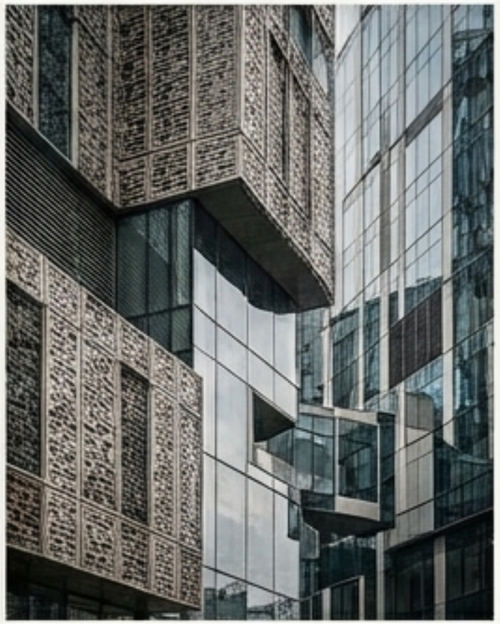


cv2.stylization

cv2.pencilSketch



cv2.detailEnhance



Stylization algorithms use edge-preserving filters to abstract and transform an image, turning photographs into artistic renderings with just a few lines of code.

The Engine of Stylization: Edge-Preserving Filters

The foundation for these effects is the ability to smooth an image while keeping important edges sharp. This is controlled by two key parameters.

`sigma_s` (Spatial Sigma)

Controls the size of the smoothing neighborhood. Larger values mean more smoothing, creating a more abstract, paint-like effect.

`sigma_r` (Range Sigma)

Controls how much color difference is considered an 'edge'. Larger values will smooth over more color variations, preserving only the strongest edges.

```
# Apply a painterly stylization effect
# sigma_s (0-200), sigma_r (0-1)
stylized_img = cv2.stylization(src, sigma_s=60, sigma_r=0.45)

# Create a pencil sketch effect
gray, color = cv2.pencilSketch(src, sigma_s=60, sigma_r=0.07, shade_factor=0.05)
```

The Computational Photography Toolbox: A Summary

Technique	Purpose	Core Idea	Key OpenCV Function
Inpainting	Restoration	Propagate information from image boundaries	`cv2.inpaint()`
NL Means Denoising	Denoising	Average similar patches across the entire image	`cv2.fastNlMeansDenoisingColored()`
HDR Imaging	Dynamic Range	Combine multiple exposures and tonemap	`cv2.createMerge...`, `cv2.createTonemap...`
Seamless Cloning	Compositing	Match gradients at boundaries (Poisson Blending)	`cv2.seamlessClone()`
Stylization	Artistic Effects	Smooth image while preserving strong edges	`cv2.stylization()`, `cv2.pencilSketch()`

Further Exploration

The concepts presented here are implemented and well-documented within the OpenCV library. To dive deeper into the code and theory, the official tutorials and underlying research papers are the best next steps.

Official OpenCV Tutorials

- Inpainting Tutorial
- Denoising Tutorial
- HDR Tutorial

Foundational Research

- ‘Poisson Image Editing’ by Pérez, Gangnet, and Blake.
- A non-local algorithm for image denoising

The power to restore, enhance, and create is a fundamental part of modern computer vision.