# Basic Intensity Transformation Functions and example with openCV

NguyenKhanhSon · Follow
7 min read · Oct 15, 2023



Basic intensity transformations are essential tools in image processing that enable the adjustment of pixel intensity values in images. These operations serve as the building blocks for enhancing and manipulating the visual appearance of various types of images, including grayscale and color ones.

In the realm of image processing, basic intensity transformations play a crucial role in tailoring images to meet specific needs and applications. They are particularly useful when images have a limited intensity range, and they help to stretch or compress this range to bring out finer details. These transformations can be used to modify overall image brightness, create binary images, and even correct the gamma of monitors or cameras. Despite their simplicity, basic intensity transformations are powerful tools for image enhancement and manipulation, making them an integral part of various fields, including computer vision, medical imaging, and digital photography.

### Linear (negative and identity transformations)

Linear intensity transformations, including negative and identity transformations, are basic operations used in image processing to alter the pixel intensities of an image. These transformations can be represented mathematically as follows:

### Negative Transformation

The negative transformation is achieved by taking the complement of the original pixel intensity values. If the original pixel intensity is denoted by “r,” the

corresponding negative intensity, denoted by “s,” can be calculated as:

```
s = L - 1 - r
```

Here, “L” represents the maximum intensity level in the image, often 255 in the case of 8-bit images. This equation subtracts the original intensity from the maximum possible intensity, effectively inverting the pixel values. For example, if the original intensity is 100 in an 8-bit image (L = 255), the negative intensity will be (255–1–100) = 154.

### Identity Transformation

The identity transformation is the simplest of all intensity transformations. It doesn’t change the pixel intensities of the image. If “r” represents the original pixel intensity and “s” is the transformed intensity, the identity transformation equation is:

```
s = r
```

In this transformation, the output intensity is equal to the input intensity, meaning that no modifications are made to the pixel values. This transformation is useful when you want to preserve the original appearance of the image without altering its intensity characteristics.

The negative transformation, with its pixel value inversion, is often used for creative or artistic purposes. The identity transformation serves as a fundamental operation in image processing, and many more complex operations are built upon this basic concept.

```
import cv2
import matplotlib.pyplot as plt

# Load the original image
original_image = cv2.imread("owl-8285565_1280.png")  # Replace with your image

# Check if the image was loaded successfully

# Perform the negative transformation
negative_image = 256 - 1 - original_image
```

```
# Display the original and negative images side by side
plt.figure(figsize=(10, 5))  # Create a figure with a specified size
plt.subplot(1, 2, 1)  # Subplot for the original image
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')  # Turn off axis labels

plt.subplot(1, 2, 2)  # Subplot for the negative image
plt.imshow(cv2.cvtColor(negative_image, cv2.COLOR_BGR2RGB))
plt.title("Negative Image")
plt.axis('off')

plt.show()  # Show the figure with both images
```


Example of negative transformation. Image source PixaBay.

### Logarithmic (log and inverse-log transformations)

Logarithmic transformations in image processing are a group of nonlinear intensity adjustments aimed at enhancing the visibility of details in images. They are particularly useful for images with a wide range of pixel values, such as those captured in low-light or high-contrast conditions. These transformations are composed of two common functions: the log transformation (logarithm function) and the inverse-log transformation (exponential function).

### Log Transformation (Logarithm Function)

The log transformation involves applying the logarithm function to each pixel value in an image. This operation spreads out the darker pixel values, making fine details in shadowed regions more visible. The transformation equation is given by:

$$S = c \cdot \log(1 + R)$$

- *S represents the transformed pixel value.*

- *R is the original pixel value.*
- *c is a constant that adjusts the degree of enhancement.*
- *The logarithm function compresses high-intensity values and stretches low-intensity values.*

### Inverse-Log Transformation (Exponential Function)

The inverse-log transformation is the reverse operation, aimed at expanding the range of brighter pixels. This is especially useful for images with overexposed regions. The transformation equation is:

$$S = e^{(R/c)} - 1$$

- *S represents the transformed pixel value.*
- *R is the original pixel value.*
- *c is a constant that controls the degree of enhancement.*
- *The exponential function stretches high-intensity values.*

Logarithmic transformations are applied to images to improve visibility in both dark and bright areas, making them a valuable tool in fields like medical imaging, astronomy, and photography. The choice of the constant ‘c’ in these transformations depends on the specific image characteristics and enhancement goals, and experimentation may be required to find the optimal value.

```
import cv2
import numpy as np

# Load an image
image = cv2.imread('owl-8285565_1280.png', cv2.IMREAD_GRAYSCALE)
# Log Transformation
c = 45
log_transformed = c * (np.log(image + 1))

# Convert to 8-bit unsigned integer format
log_transformed = np.uint8(log_transformed)

# Inverse-Log Transformation
inverse_log_transformed = c * (np.exp(log_transformed / c) - 1)

# Convert to 8-bit unsigned integer format
```

```
inverse_log_transformed = np.uint8(inverse_log_transformed)
# Display the original and negative images side by side
plt.figure(figsize=(10, 5))  # Create a figure with a specified size
plt.subplot(1, 3, 1)  # Subplot for the original image
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')  # Turn off axis labels

plt.subplot(1, 3, 2)
plt.imshow(cv2.cvtColor(log_transformed, cv2.COLOR_BGR2RGB))
plt.title("log_transformed")
plt.axis('off')
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(inverse_log_transformed, cv2.COLOR_BGR2RGB))
plt.title("inverse_log_transformed")
plt.axis('off')
plt.show()
```

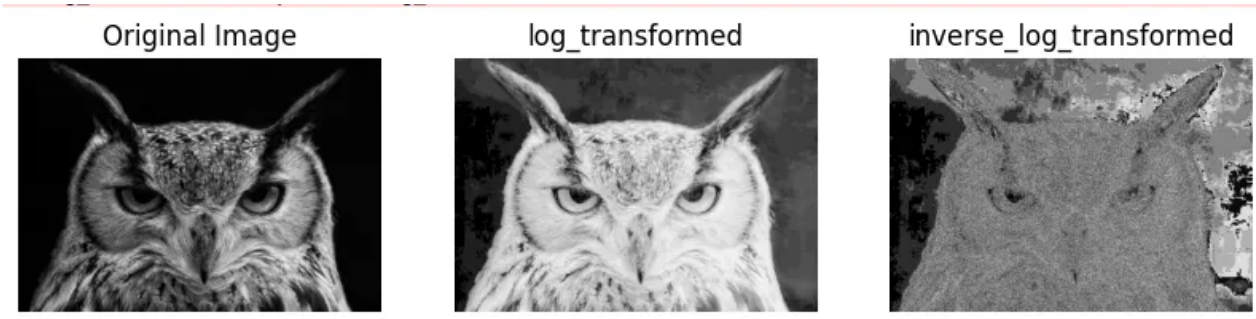



Image source PixaBay.

**Power-law transformations**

Power-law transformations, also known as gamma correction, are a class of mathematical transformations used to adjust the tonal and brightness characteristics of an image. These transformations are particularly useful in image processing and computer vision to enhance the visual quality of images or correct for issues related to illumination and contrast.

The basic idea behind power-law transformations is to raise the pixel values of an image to a certain power (exponent) in order to adjust the image's overall brightness and contrast. The general form of a power-law transformation is:

$$O = k \cdot I^{\gamma}$$

*Where:*

- *O is the output pixel value (transformed value).*
- *I the input pixel value (original value).*
- *γ is the exponent, which controls the degree of transformation.*
- *k is a constant that scales the result to fit within the desired intensity range.*

Key points about power-law transformations:

> *Gamma Correction: When γ>1, it is known as gamma correction, and it brightens the image. When γ<1, it darkens the image.*
>
> *Adjusting Contrast: Higher values of γ (greater than 1) increase contrast by making dark areas darker and bright areas brighter, while lower values (between 0 and 1) decrease contrast.*
>
> *No Change: When γ=1, the transformation has no effect on the image; the output is the same as the input.*
>
> *Nonlinear Transformation: Power-law transformations are nonlinear transformations and are especially useful for images with non-uniform illumination.*
>
> *Applications: Power-law transformations are used in various applications, including image enhancement, gamma correction for display devices, and improving the visibility of details in medical imaging.*

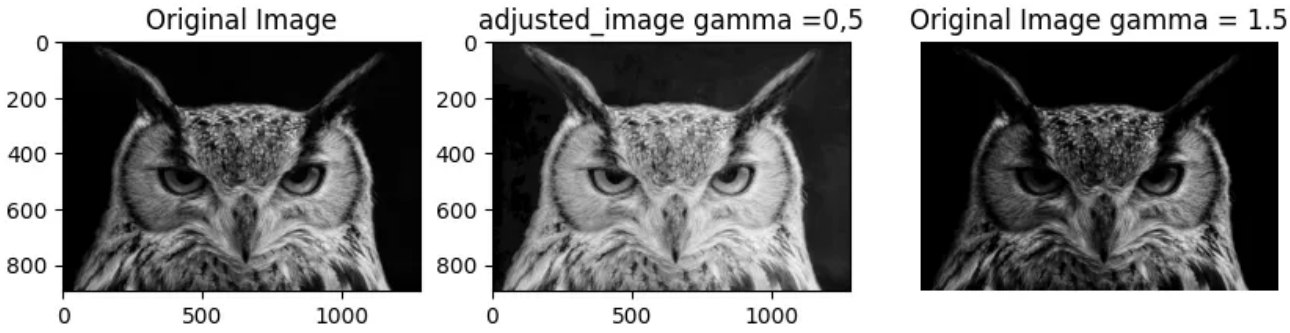
```
import cv2
import numpy as np

# Read an image
image = cv2.imread('test2.png', cv2.IMREAD_GRAYSCALE)

# Apply gamma correction (e.g., gamma = 1.5)
gamma = 2
gamma2=4
adjusted_image = np.power(image / 255.0, gamma) * 255.0
adjusted_image = adjusted_image.astype(np.uint8)
adjusted_image2 = np.power(image / 255.0, gamma2) * 255.0
adjusted_image2 = adjusted_image2.astype(np.uint8)
plt.figure(figsize=(10, 5))  # Create a figure with a specified size
plt.subplot(1, 3, 2)  # Subplot for the original image
plt.imshow(cv2.cvtColor(adjusted_image, cv2.COLOR_BGR2RGB))
plt.title("adjusted_image gamma =2")
```

```
plt.subplot(1, 3, 1)  # Subplot for the original image
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")

plt.subplot(1, 3, 3)  # Subplot for the original image
plt.imshow(cv2.cvtColor(adjusted_image2, cv2.COLOR_BGR2RGB))
plt.title("Original Image gamma = 4")
plt.axis('off')
```




**Conclusion**

These intensity transformations are valuable for a variety of image processing tasks, including image enhancement, contrast adjustment, and improving the visibility of specific image features. They find applications in fields ranging from photography and computer vision to medical imaging and scientific analysis.

Understanding these basic transformation functions provides a foundation for more advanced image processing techniques, enabling the creation of visually appealing images and the extraction of useful information from digital representations of the real world. The choice of transformation function depends on the specific goals and characteristics of the images being processed, and the appropriate selection can significantly impact the quality and interpretability of the results.

Digital Image Processing | Digital Images | Computer Vision

Follow

## Written by NguyenKhanhSon

6 Followers · 1 Following

Hi every, Thanks for vising my profile. I'm master student in AI and try to rewrite something I learned. Have a good day.

## No responses yet

What are your thoughts?

Respond

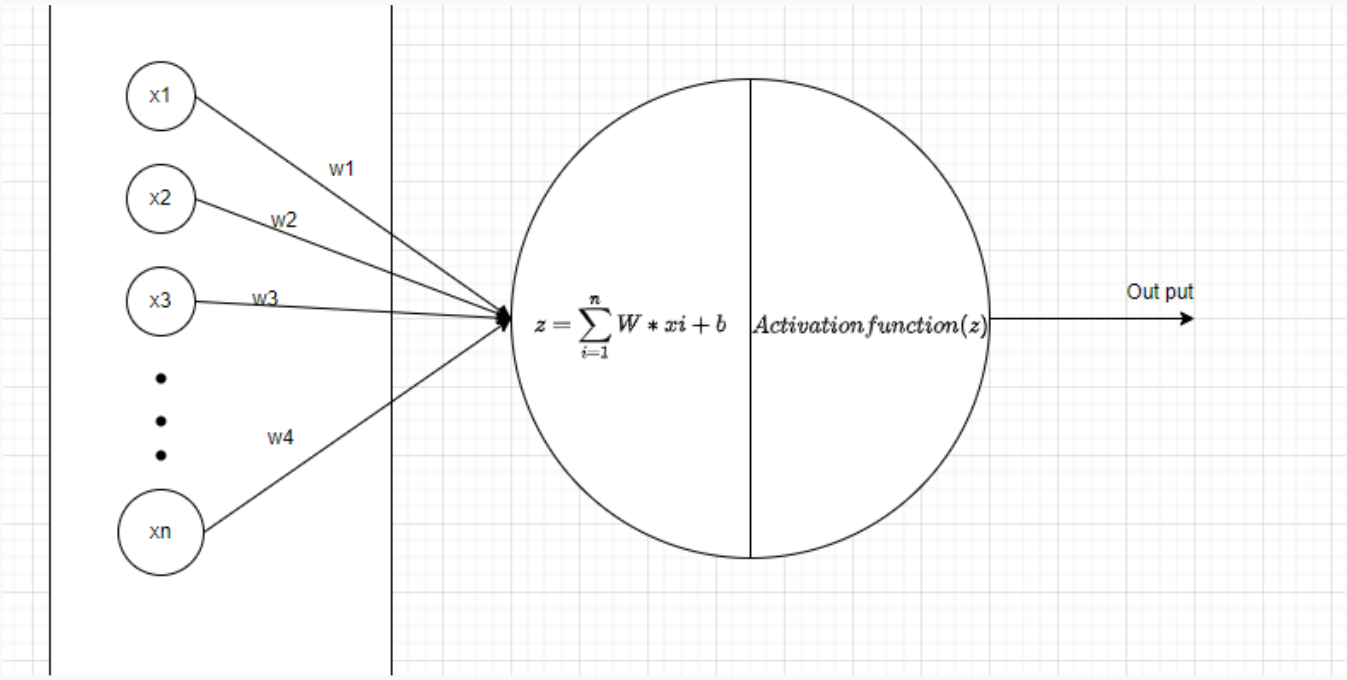
## More from NguyenKhanhSon

NguyenKhanhSon

### Unlocking Image Enhancement: A Guide to Histogram Processing and Equalization with OpenCV

Histogram processing is a fundamental technique in digital image processing that plays a crucial role in enhancing the visual quality and...

Oct 21, 2023 4 1

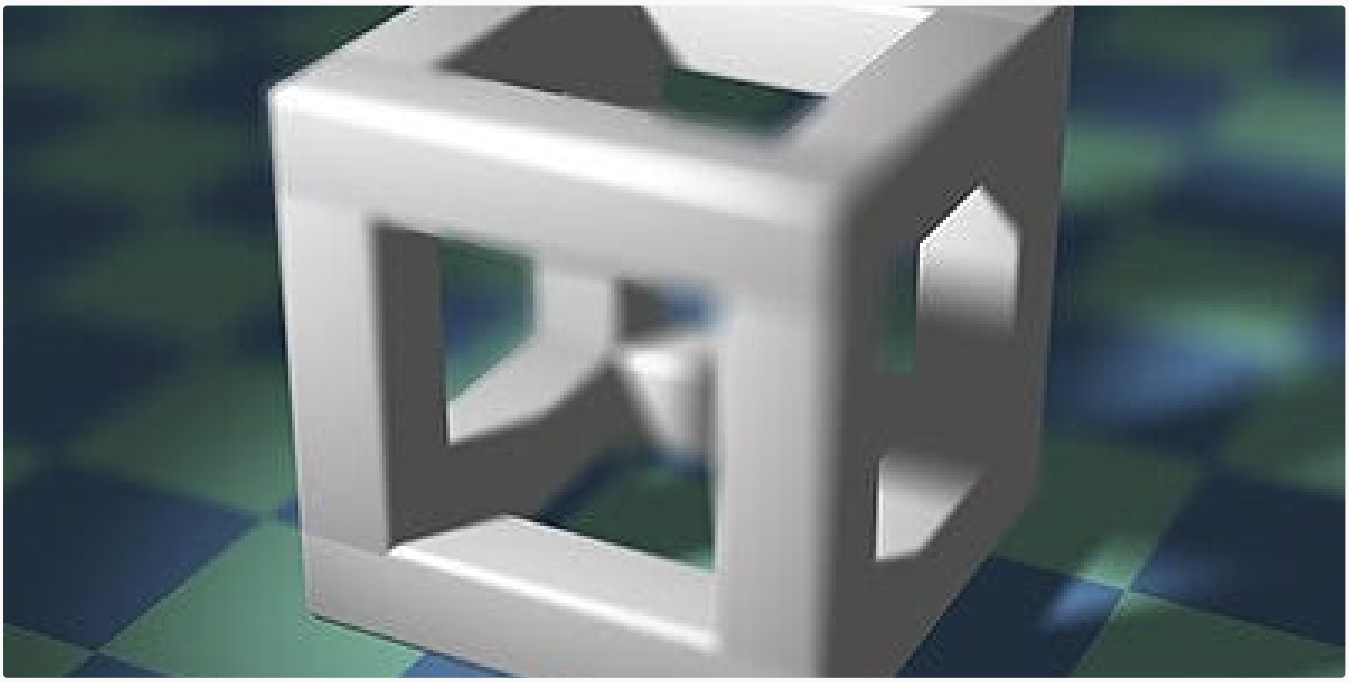



NguyenKhanhSon

## Activation function in Deep Learning

What is the Activation function?

Oct 24, 2023 4

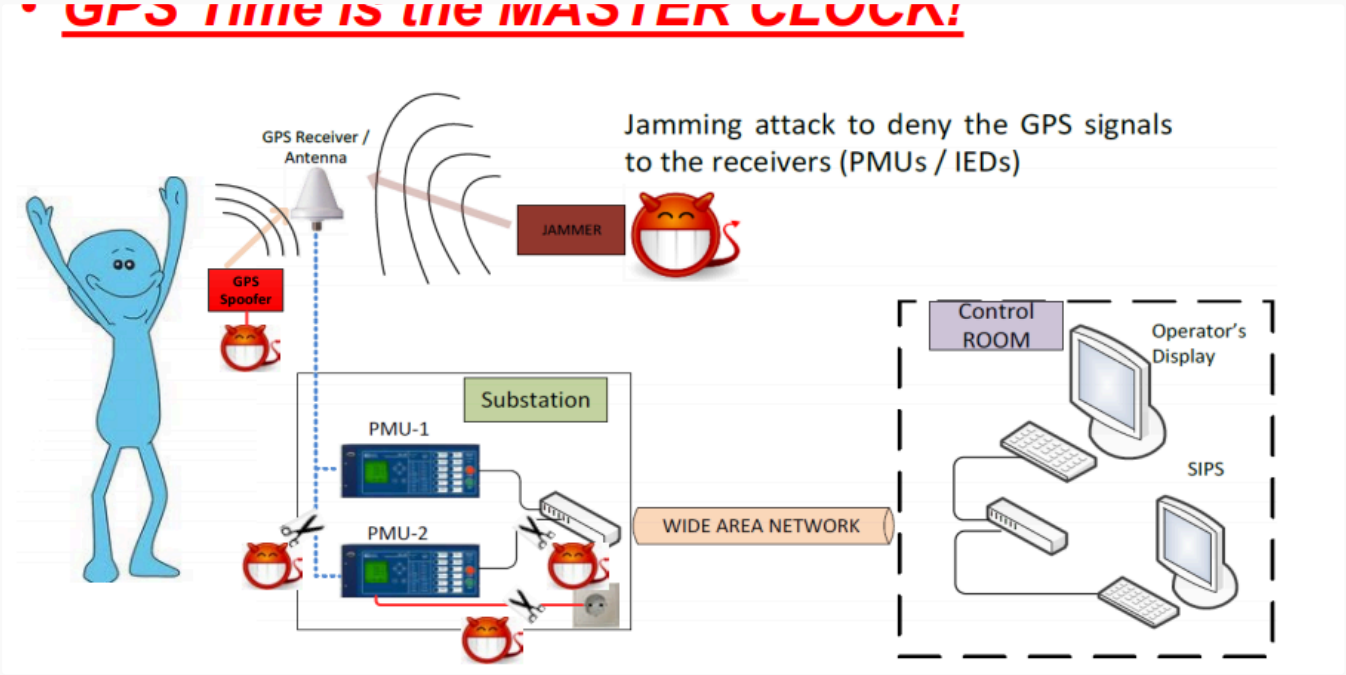


NguyenKhanhSon

## Boost Your Depth Image Edge Detection with This Modified LBP Method

This work implemented from paper “ LBP Based_Edge_Detection_Method_for_Depth_Images_With_Low_Resolutions” Link ...

Nov 4, 2023 1




NguyenKhanhSon

## Securing Smart Grids: Investigate impact of Time Synchronization Attacks on Phasor Measurement...

Time Synchronization attacks on phasor measurement units

Nov 25, 2023

See all from NguyenKhanhSon

## Recommended from Medium


Avinash Maheshwari

## Image Processing Tool with Python

In this article, we'll walk through the creation of an intuitive image processing and real time viewer tool using Python, Tkinter, and...

Jan 12 56 1


In The Deep Hub by Jorgecardete

## The Art and Science of Interpolation

Exploring the pillars of image processing

Feb 9, 2024 622 4

## Lists

**Natural Language Processing**

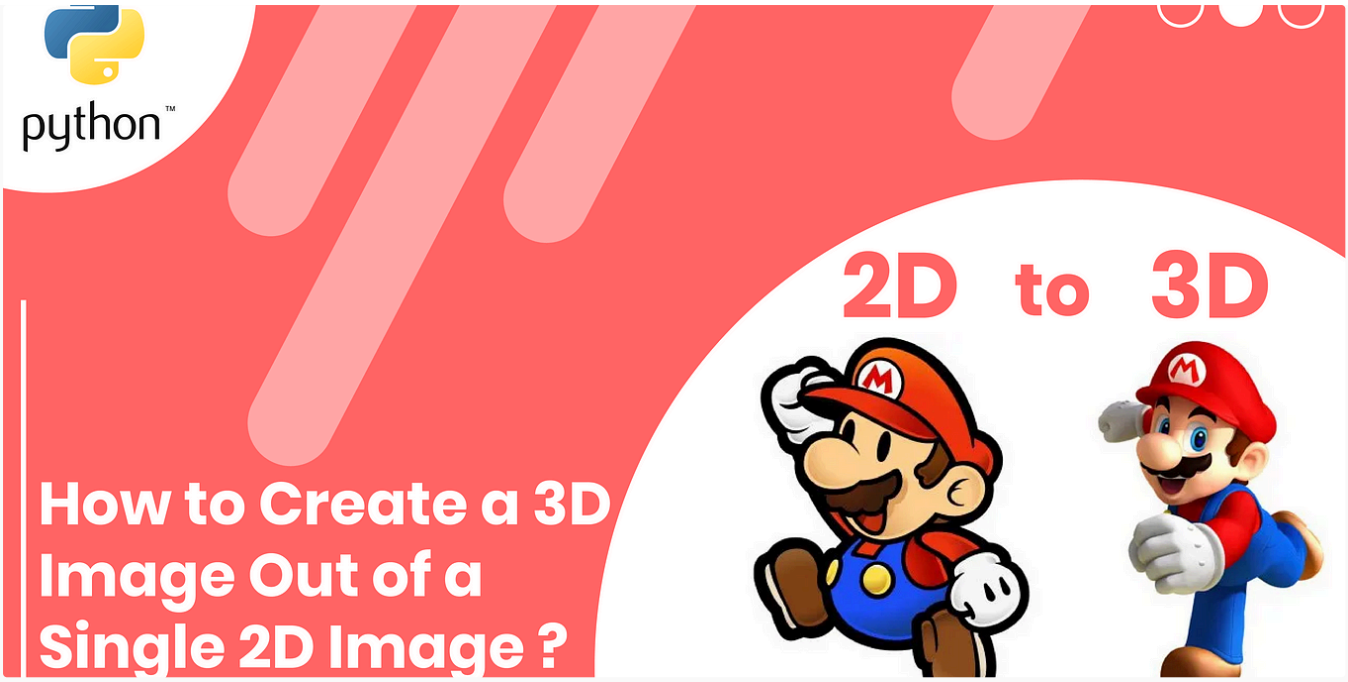
1894 stories · 1555 saves

Abisha

# Image Feature Extraction using Python - Part I

Basics of Image feature extraction techniques using python

Sep 5, 2024 623 13

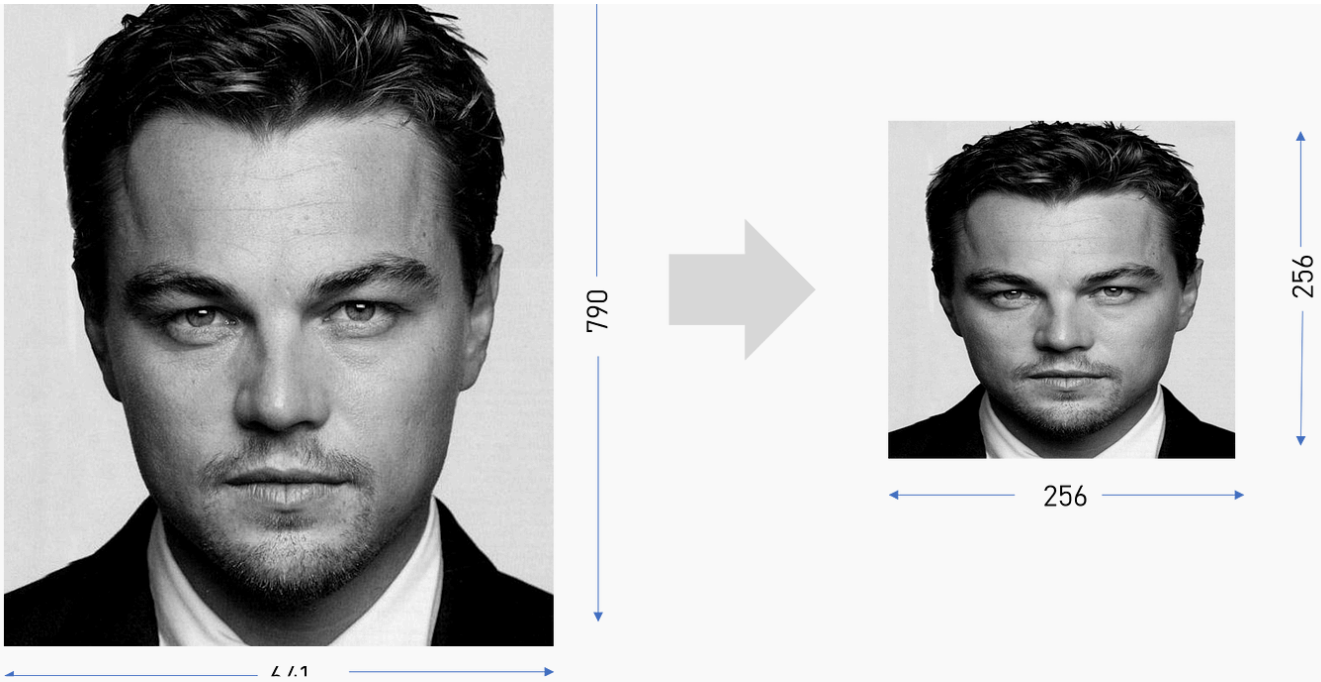



Eran Feit

## 3D Photo Magic | Convert Any Picture to 3D with Python

Hi,

Sep 11, 2024 6




Tom Tillo

## Patch based Image Color normalization and merging

How to colorize / transform Images without losing their aspect ratio and quality

Oct 16, 2024

Elmar H.

## Image Preprocessing for Computer Vision: Data Augmentation

What is Data Augmentation?

Sep 10, 2024

See more recommendations