



# Understanding Digital Images for Image Processing and Computer Vision (Part 1):

In this first part, we will delve into the fundamental concepts of image theory. Subsequent articles in the series will provide practical demonstrations with code, offering a comprehensive understanding of digital images, which has significant importance in the field of Image Processing and Computer Vision.



Md. Jewel · [Follow](#)

8 min read · Oct 20, 2023



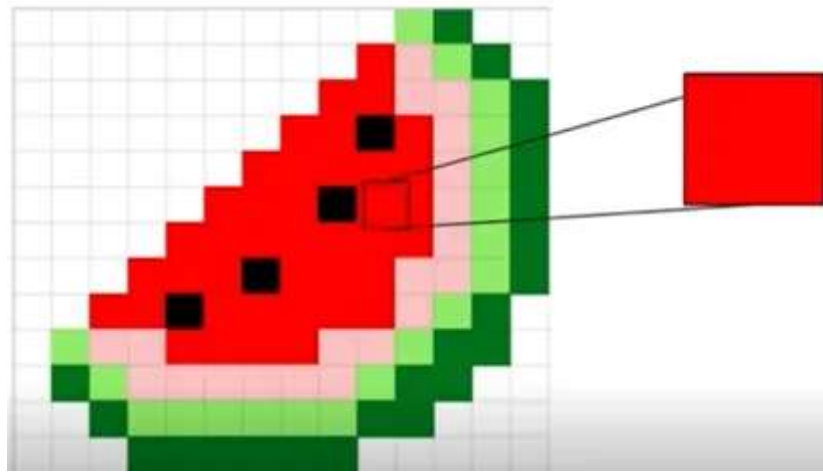
Listen



Share

... More

**Pixels:** Let's start with the 'Pixel'. Generally, images are composed of pixels, or "picture elements," which serve as the core units. Each pixel represents vital information about color and brightness, collectively forming the visual representation. In grayscale images, a pixel's intensity value denotes its brightness on a scale from 0 (black) to 255 (white) within an 8-bit system. For colored images, pixels contain information for three primary color channels: **Red (R)**, **Green (G)**, and **Blue (B)**, with each channel's intensity ranging from 0 to 255. Different combinations of these values produce a diverse spectrum of colors. For instance, full intensity in all channels (255, 255, 255) results in white, while no intensity in any channel (0, 0, 0) yields black. This is the very fundamental understanding of pixel intensity for image representation and processing in the digital domain.

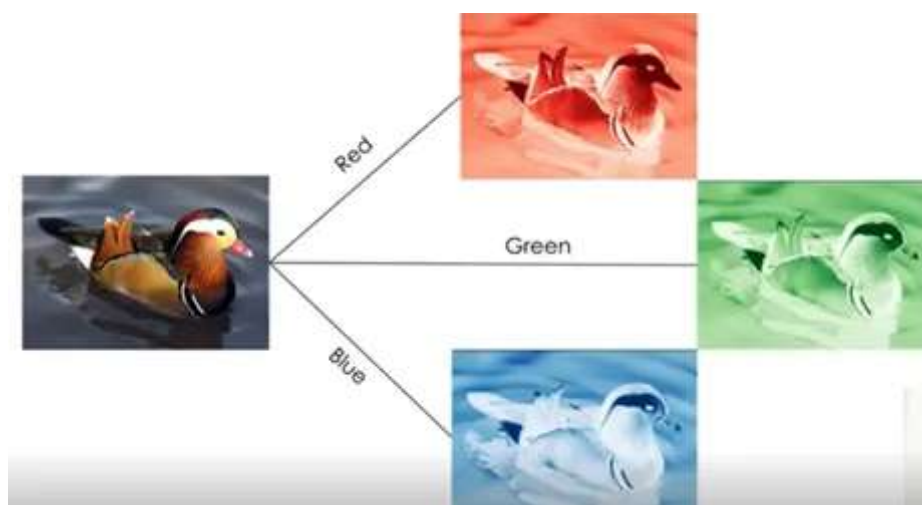


RGB Pixels

Generally, digital images have 3 major components:

- 1. **Size:** Size denotes the height and width of an digital image, which is measured by the number of pixels.*
- 2. **Color space:** It represents various conceivable color spaces, including Grayscale, RGB, and HSV. The image of the duck is depicted in the RGB color space.*
- 3. **Channels:** This explains the attributes of a color space — For example, RGB has three color channels: Red, Green, and Blue.*

You'll find the separate images of the Red (R), Green (G), and Blue (B) channels extracted from the RGB representation of the duck below:








Color space from original image

**RGB Color Scheme:** The RGB color model, widely utilized in digital devices and displays, derives its name from the primary additive colors: Red, Green, and Blue. In

this model, colors are formed through the combination of varying intensities of these three primary hues. Each channel (R, G, B) has an intensity range from 0 to 255, offering a staggering 16.8 million possible color variations ( $256^3$ ).

The higher the pixel intensity value, the more the brightness of the color.

Color	Color Name	Pixel Intensity Values - RGB Color Space		
		R (Red)	G (Green)	B (Blue)
	Red	255	0	0
	Green	0	255	0
	Blue	0	0	255
	Yellow	255	255	0
	Cyan	0	255	255

RGB Color Scheme





**Red (R):** Controls the presence of red light in the color spectrum. Maximum intensity (255, 0, 0) yields pure red, while absence (0, 0, 0) results in black.

**Green (G):** Dictates the degree of green light. Full intensity (0, 255, 0) produces pure green, while no intensity (0, 0, 0) leads to black.

**Blue (B):** Controls the amount of blue light. Maximum intensity (0, 0, 255) produces pure blue, while absence (0, 0, 0) results in black.

By mixing different amounts of these main colors, we can create a wide range of shades, from bright and lively to more subtle tones. For example, combining full red (255, 0, 0) with full green (0, 255, 0) gives us yellow (255, 255, 0). When red, green, and blue are equally intense (255, 255, 255), we get white. When there's no intensity in any channel (0, 0, 0), we get black. This model forms the basis for the rich palette of digital colors.

Likewise, the following figure is an example of grayscale colors:

Color	Color Name	Pixel Intensity Values - RGB Color Space		
		R (Red)	G (Green)	B (Blue)
	Black	0	0	0
	White	255	255	255
	Dim Gray	100	100	100
	Very Light Gray	200	200	200

Grayscale Color Scheme

**RGB to Grayscale Conversion (Dimension reduction)** : Up to this point, we've acquired an understanding of RGB and Grayscale images in terms of pixels. Now, the question arises: what purpose do Grayscale images serve? Do we even need to convert RGB images into Grayscale? If so, how do we go about it? Let's explore the reasons and methods behind this.

Converting an image to grayscale is like using a black and white filter. It's really useful when we don't need to worry about colors. Grayscale images are simpler for computers to handle because they use less memory and processing power. They work really well for tasks like improving images, recognizing faces, and Machine Learning applications. Plus, they can help us focus on the details without getting distracted by colors.

Again, transforming an image (also known as dimension reduction) from RGB to grayscale means changing it from full color to black and white. Instead of three color channels (Red, Green, Blue), grayscale has just one channel that represents brightness. This simplifies the image without losing important details. Here's a simple explanation of how this works:

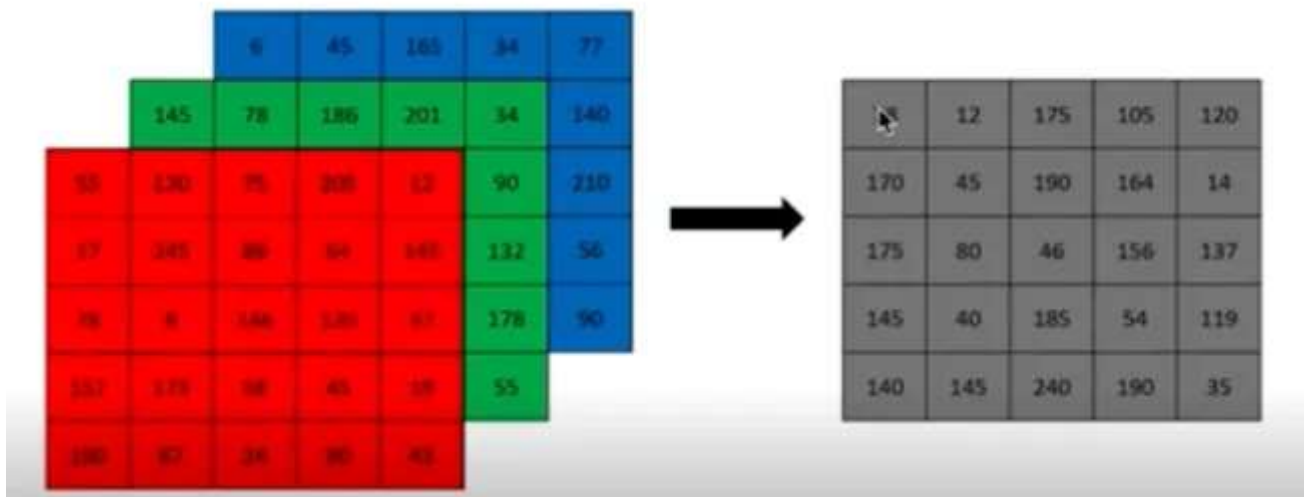
1. An easy method for converting an RGB image to a grayscale image is to use an average. The formula is:

$$\text{Grayscale} = (R+G+B)/3$$

2. As the human eye is more sensitive to green, followed by red, and less sensitive to blue, the researchers have developed another formula that is :

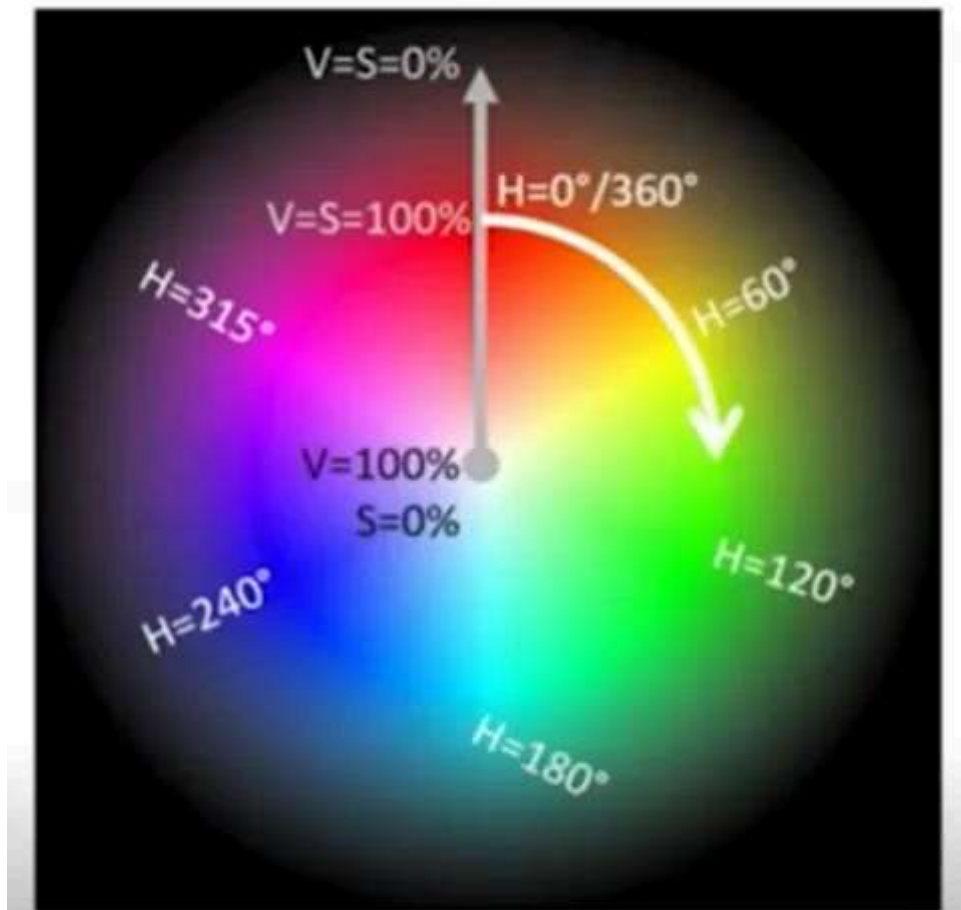
$$\text{Grayscale} = (0.299 \times R) + (0.587 \times G) + (0.114 \times B)$$

These coefficients (0.299, 0.587, 0.114) are chosen to approximate the human perception of color intensity. By applying this formula to each pixel in the image, we can obtain a single intensity value for each pixel in the grayscale image, which represents its brightness level.



RGB to Grayscale

**HSV Color Scheme :**



HSV Color Scheme

HSV is another color model that stands for **Hue, Saturation, Value**. These are the three main aspects:

**Hue:** This is the color itself in an image. It's measured in degrees and can range from 0 to 360 degrees.

**Saturation:** This indicates how much gray is mixed with the color. It's represented in percentage, so it goes from 0% (completely gray) to 100% (pure color).

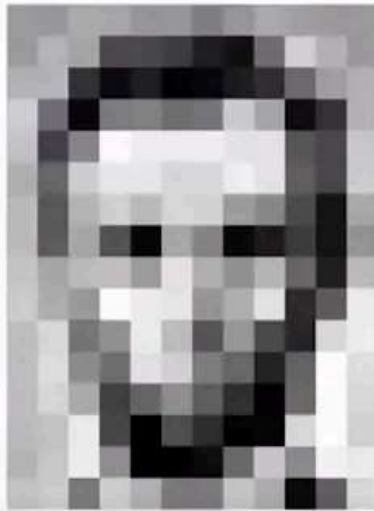
**Value:** This signifies the brightness or intensity of a color. It's also measured in percentage, ranging from 0% (completely dark) to 100% (full brightness).

**Image representation as an array:**



## DIGITAL IMAGES

BLACK: 0 - - - - - WHITE: 255



187	182	174	168	162	162	129	183	172	161	158	186
188	182	169	74	76	62	88	17	110	210	180	184
180	180	60	14	34	6	10	33	48	106	109	181
206	108	6	108	101	111	130	204	166	16	66	180
194	68	187	261	257	239	239	228	227	67	71	201
172	106	207	233	233	214	220	239	238	86	74	206
188	88	179	206	188	218	211	168	186	76	20	189
189	97	168	84	10	168	134	11	31	62	22	148
189	168	181	183	168	227	179	143	182	106	36	190
205	174	188	282	236	231	149	179	238	43	85	234
190	216	116	149	236	187	86	180	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	90	2	109	249	210
187	196	226	76	1	81	47	0	6	217	256	211
183	202	237	146	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	86	218

187	182	174	168	162	162	129	183	172	161	158	186
188	182	169	74	76	62	88	17	110	210	180	184
180	180	60	14	34	6	10	33	48	106	109	181
206	108	6	108	101	111	120	204	166	16	66	180
194	68	187	261	257	239	239	228	227	67	71	201
172	106	207	233	233	214	220	239	238	86	74	206
188	88	179	206	188	218	211	168	186	76	20	189
189	97	168	84	10	168	134	11	31	62	22	148
189	168	181	183	168	227	179	143	182	106	36	190
205	174	188	282	236	231	149	179	238	43	85	234
190	216	116	149	236	187	86	180	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	90	2	109	249	210
187	196	226	76	1	81	47	0	6	217	256	211
183	202	237	146	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	86	218

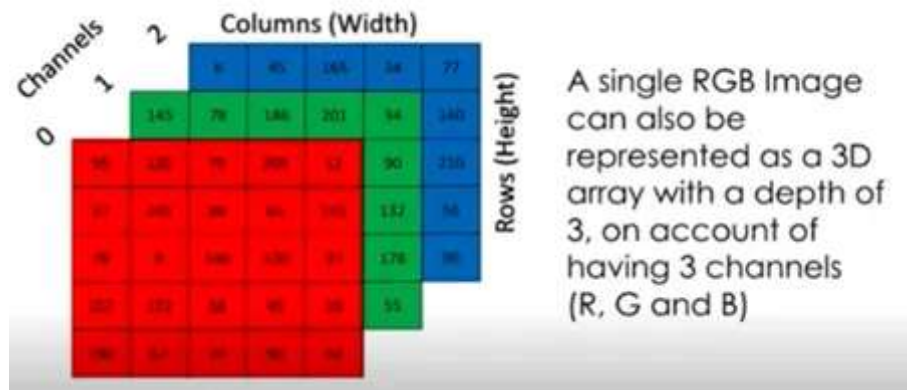
Image representation as an array

Alright, think of an image like a big grid filled with numbers. Each number, known as a pixel, is like a tiny piece of the picture. It tells us something important about the color or how bright that part of the picture is. In grayscale images, each pixel is typically represented by a single value, where higher values indicate brighter areas and lower values represent darker areas. This can be visualized as a 2D array where each element contains a single intensity value. The figure of multiple Grayscale image is given below:

**Shape: (no. of samples, height, width)**  
**Ex: (3,16,16)**

Columns (Width)					No. of Samples		
Rows (Height)	78	12	175	105	120		
	170	78	12	175	105	120	
	175	170	78	12	175	105	120
	145	175	170	45	190	164	14
	140	145	175	80	46	156	137
		140	145	40	185	54	119
			140	145	240	190	35

## Multiple Grayscale images



RGB image as an array

A single RGB image is also a 3d array with the depth dimension always having a value of 3 since each pixel of the 2d image has 3 channels (R,G,B). However representing multiples RGB images would require 4D array, because an extra dimension is required to show the number of sample images. For example, if we have three pictures, each 16 pixels tall and 16 pixels wide, with red, green, and blue colors, we'd represent it like this: (3 pictures, 16 pixels tall, 16 pixels wide, 3 colors). So, it's like saying (3, 16, 16, 3) in computer language.

For more clarity we can say, in the RGB color model, each pixel in a colored image is defined by three values (R, G, B) representing the levels of red, green, and blue respectively. This information is organized in a 3D array, with each element containing the three color channel values. In contrast, a grayscale image, which only measures intensity and not color, is represented in a simpler 2D array. For instance, a 100x100 grayscale image is a 2D array of dimensions 100x100, while a 100x100 RGB image is a 3D array of dimensions 100x100x3, with each element holding the RGB values for a particular pixel.

**Pixel normalization:** So to recap, in grayscale images, each pixel can be represented by a single number. However in RGB colored images, each pixel has to be represented by a vector of three numbers, for the three primary color channels; red, green, blue. As we also saw earlier, the pixel intensity values of the RGB digital color space can vary from 0–255. These pixel intensity values representing the image, can also be normalized/ rescaled into a range from [0, 1], as this helps reduce the storage used for each image's pixel values.



This kind of normalization or scaling is preferred for NN (Neural Networks) in computer vision, since computational cost is always an important consideration in Deep Learning. It is implemented using a rescaling ratio by which each pixel can be multiplied in order to achieve the desired range. an example of such ratio is  $1/255$  (about 0.0039).

**Common Images Characteristics:** There are certain standard images resolution and aspect ratios that are often used with images in real-world applications such as:

***The aspect ratio:** The aspect ratio of an image is a term used to describe the ratio of the width of an image to its height. It is usually denoted with two numbers separated by a colon. A few common images aspects ratios are 1:1, 3:2, 5:4 and 16:9*

***The resolution:** The resolution of an image as a term that describes how many pixels the image consists of. For example, an image with a width of 640 pixels and height of 480 pixels is said to have a resolution of 640x480, which is over 0.3MP (Megapixels).*

So, the higher the number of pixels in an image, the higher is the resolution.

**Python Imaging Library :** Alright, let's talk about the Python Imaging Library, Pillow, previously known as PIL. It covers tasks like resizing, color adjustments, and more. Its compatibility with NumPy makes it a go-to tool for image processing across various domains, including computer vision and web development.

So, finally, we have tried to cover the fundamentals of understanding digital images. In the next Part-2, we'll delve into implementing this explanation with actual code.

Pixel Theory

Digital Images



Follow

# Written by Md. Jewel


2 Followers

🤖 AI/ML Enthusiast | Data Science Explorer | Tech Storyteller 🚀 Exploring the realms of AI and ML to unravel the mysteries of modern world..

---

## More from Md. Jewel



 Md. Jewel

## A Beginner's Guide to Face Detection using OpenCV in Python

Introduction:

Oct 12, 2023



---

See all from Md. Jewel

---

## Recommended from Medium

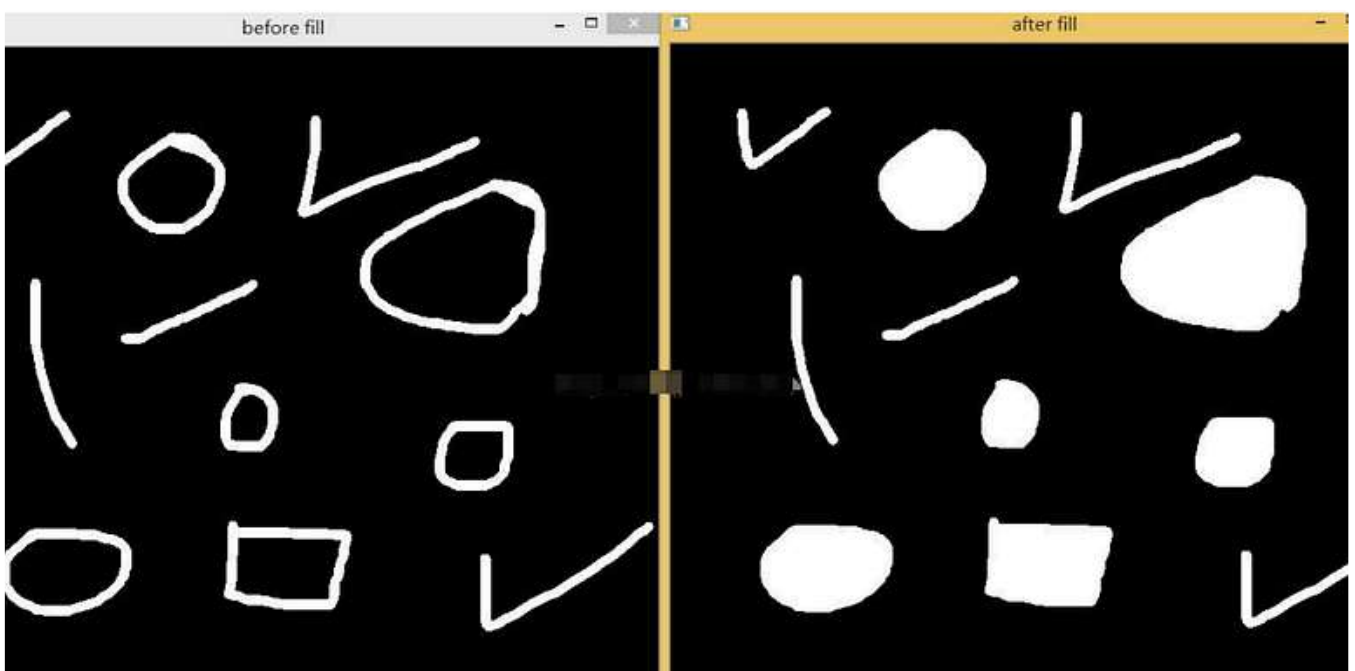
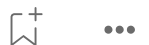


 Abhishek Jain

### Salt and pepper noise and how to remove it in machine learning

What is Salt and Pepper noise ?

Jun 16  2



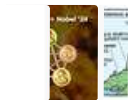
## Hole Filling of Opencv Binary Image use floodFill

Sometimes, we need to fill the image with holes, and the general idea is to use the findcontours function to find the outermost outline and...

✦ Jun 27



### Lists



#### Staff Picks

751 stories · 1398 saves



#### Stories to Help You Level-Up at Work

19 stories · 845 saves



#### Self-Improvement 101

20 stories · 2925 saves



#### Productivity 101

20 stories · 2478 saves



## Blur detection using gradient-based metric

In this blog post, I will guide you through the process of determining the level of blur in an image using OpenCV, Python, and the...


May 17  6  

$$60^\circ \times \left( \frac{G_{norm} - B_{norm}}{\text{Chroma}} \right)$$
$$60^\circ \times \left( \frac{B_{norm} - R_{norm}}{\text{Chroma}} + 2 \right)$$
$$60^\circ \times \left( \frac{R_{norm} - G_{norm}}{\text{Chroma}} + 4 \right)$$

if  $C_{max}$



if  $C_{max}$

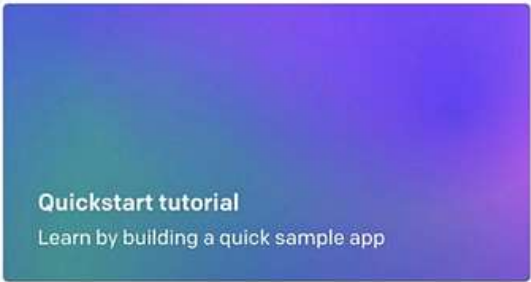
if  $C_{max}$

 Alakh Sharma

## Understanding RGB to HSV Color Space Conversion

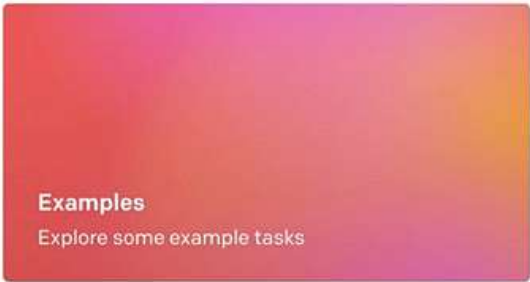
Introduction to Color Spaces

 Aug 26  



**Quickstart tutorial**


Learn by building a quick sample app



**Examples**


Explore some example tasks

### Build an application




**GPT**

Learn how to generate text and call functions




**Embeddings**



**GPT best practices**

Learn best practices for building with GPT models

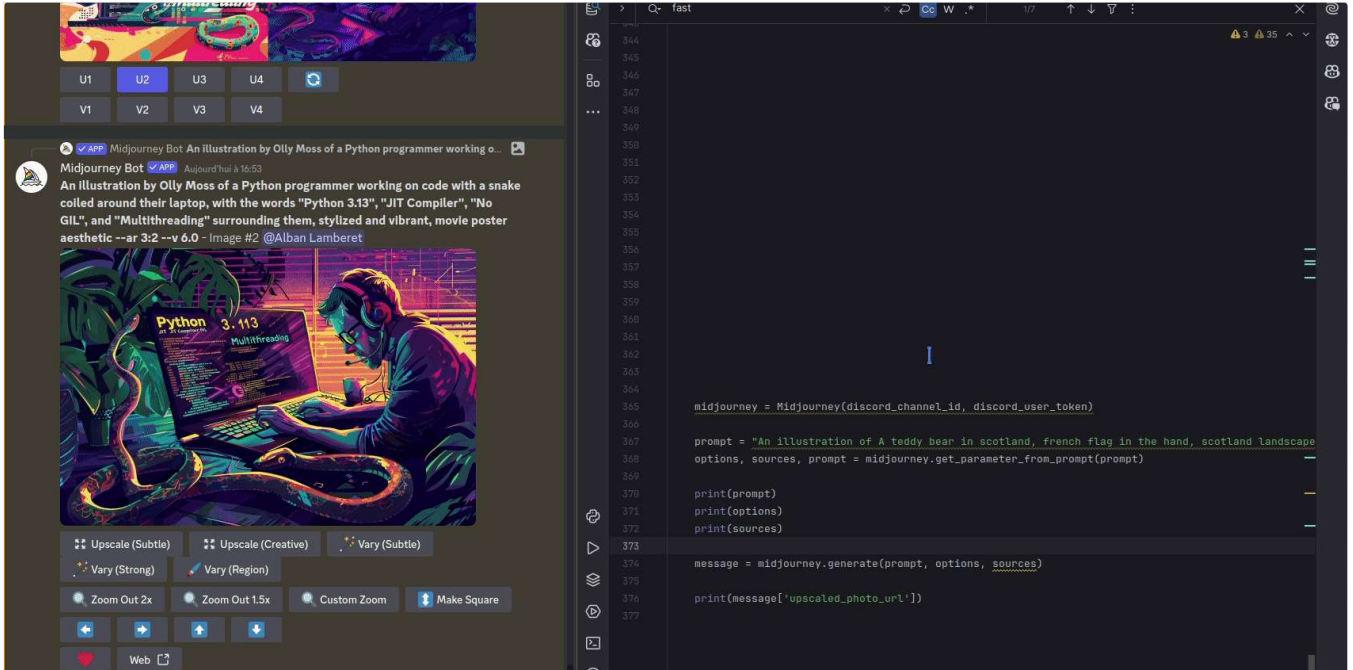


**Speech to text**

## 7 Powerful ChatGPT Alternatives

Not just any AI chatbot, but one that's smart and can carry on a conversation almost like a human.

★ Sep 17



Alban Lamberet

## How to generate a Midjourney image with Python ?

I just finished my platform that allows me to automatically publish articles on my WordPress blogs. Before I present to you how I did it...

★ May 15 🤝 1 💬 1



See more recommendations