

<https://gurus.pyimagesearch.com/>



PyImageSearch Gurus Course

[\(HTTPS://GURUS.PYIMAGESEARCH.COM\)](https://gurus.pyimagesearch.com/)

1.4.6: Image arithmetic

Topic Progress: [\(https://gurus.pyimagesearch.com/topic/translation/\)](https://gurus.pyimagesearch.com/topic/translation/) [\(https://gurus.pyimagesearch.com/topic/rotation/\)](https://gurus.pyimagesearch.com/topic/rotation/) [\(https://gurus.pyimagesearch.com/topic/resizing/\)](https://gurus.pyimagesearch.com/topic/resizing/) [\(https://gurus.pyimagesearch.com/topic/flipping/\)](https://gurus.pyimagesearch.com/topic/flipping/) [\(https://gurus.pyimagesearch.com/topic/cropping/\)](https://gurus.pyimagesearch.com/topic/cropping/) [\(https://gurus.pyimagesearch.com/topic/image-arithmetic/\)](https://gurus.pyimagesearch.com/topic/image-arithmetic/) [\(https://gurus.pyimagesearch.com/topic/bitwise-operations/\)](https://gurus.pyimagesearch.com/topic/bitwise-operations/) [\(https://gurus.pyimagesearch.com/topic/masking/\)](https://gurus.pyimagesearch.com/topic/masking/) [\(https://gurus.pyimagesearch.com/topic/splitting-and-merging-channels/\)](https://gurus.pyimagesearch.com/topic/splitting-and-merging-channels/)

[← Back to Lesson \(https://gurus.pyimagesearch.com/lessons/basic-image-processing/\)](https://gurus.pyimagesearch.com/lessons/basic-image-processing/)

Remember way, way back when you studied how to add and subtract numbers in grade school?

Well, it turns out, performing arithmetic with images is quite similar — with only a few caveats of course.

In this module you'll learn how to *add* and *subtract* images, along with two important differences you need to understand regarding arithmetic operations in OpenCV and Python.

Objectives:

This topic has two primary objectives:

1. To familiarize ourselves with image addition and subtraction.
2. To understand the difference between OpenCV and NumPy image arithmetic operations.

Image Arithmetic

In reality, image arithmetic is simply matrix addition (with an added caveat on data types, which we'll explain later).

Let's take a second and review some very basic linear algebra.

Suppose we were to add the following two matrices:

$$\begin{bmatrix} 9 & 3 & 2 \\ 4 & 1 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 9 & 4 \\ 7 & 9 & 4 \end{bmatrix}$$

What would the output of the matrix addition be?

The answer is simply the *element-wise* sum of matrix entries:

$$\begin{bmatrix} 9+0 & 3+9 & 2+4 \\ 4+7 & 1+9 & 4+4 \end{bmatrix} = \begin{bmatrix} 9 & 12 & 6 \\ 11 & 10 & 8 \end{bmatrix}$$

Pretty simple, right?

So it's obvious at this point that we all know basic arithmetic operations like addition and subtraction.

But when working with images, we need to keep in mind the limits of our *color space* and *data type*.

For example, RGB images have pixels that fall within the range $[0, 255]$. What happens if we are examining a pixel with intensity 250 and we try to add 10 to it?

Under normal arithmetic rules, we would end up with a value of 260. However, since RGB images are represented as 8-bit unsigned integers, 260 is not a valid value.

So what should happen? Should we perform a check of some sorts to ensure no pixel falls outside the range of $[0, 255]$, thus clipping all pixels to have a minimum value of 0 and a maximum value of 255?

Or do we apply a modulus operation, and "wrap around?" Under modulus rules, adding 10 to 255 would simply wrap around to a value of 9.

Which way is the "correct" way to handle images additions and subtractions that fall outside the range of $[0, 255]$?

The answer is that *there is no correct way* — it simply depends on how you are manipulating your image, and what you want the desired results to be.

NumPy will perform modulus arithmetic and “wrap around.” OpenCV, on the other hand, will perform clipping and ensure pixel values never fall outside the range `[0, 255]`.

But don’t worry! These nuances will become more clear as we explore some code below.

arithmetic.py	Python
<pre>1 # import the necessary packages 2 import numpy as np 3 import argparse 4 import cv2 5 6 # construct the argument parser and parse the arguments 7 ap = argparse.ArgumentParser() 8 ap.add_argument("-i", "--image", required=True, help="Path to the image") 9 args = vars(ap.parse_args()) 10 11 # load the image and show it 12 image = cv2.imread(args["image"]) 13 cv2.imshow("Original", image) 14 15 # images are NumPy arrays, stored as unsigned 8 bit integers -- this 16 # implies that the values of our pixels will be in the range [0, 255]; when 17 # using functions like cv2.add and cv2.subtract, values will be clipped 18 # to this range, even if the added or subtracted values fall outside the 19 # range of [0, 255]. Check out an example: 20 print("max of 255: {}".format(str(cv2.add(np.uint8([200]), np.uint8([100]))))) 21 print("min of 0: {}".format(str(cv2.subtract(np.uint8([50]), np.uint8([100]))))) 22 23 # NOTE: if you use NumPy arithmetic operations on these arrays, the value 24 # will be modulo (wrap around) instead of being clipped to the [0, 255] 25 # range. This is important to keep in mind when working with images. 26 print("wrap around: {}".format(str(np.uint8([200]) + np.uint8([100])))) 27 print("wrap around: {}".format(str(np.uint8([50]) - np.uint8([100]))))</pre>	

Feedback

We are going to perform our standard procedure on **Lines 1-13** by importing our packages, setting up our argument parser, and loading our image.

Remember how I mentioned the difference between OpenCV and NumPy addition above? Well now we are going to explore it further and provide a concrete example to ensure we fully understand it.

On **Line 20** we define two NumPy arrays that are 8-bit unsigned integers. The first array has one element: a value of 200. The second array also has only one element, but a value of 100. We then use OpenCV’s `cv2.add` method to add the values together.

What do you think the output is going to be?

Well according to standard arithmetic rules, we would think the result should be 300, *but*, remember

using the `cv2.add` method, OpenCV takes care of clipping for us, and ensures that the addition produces a maximum value of 255.

When we execute this code, we can see the result on the first line of in the listing below:

arithmetic.py	Shell
1 max of 255: [[255]]	

Sure enough, the addition returned a value of 255.

Line 21 then performs subtraction using `cv2.subtract`. Again, we define two NumPy arrays, each with a single element, and of the 8-bit unsigned integer data type. The first array has a value of 50 and the second a value of 100.

According to our arithmetic rules, the subtraction should return a value of -50; however, OpenCV once again performs clipping for us. We find that the value is clipped to a value of 0. Our output below verifies this:

arithmetic.py	Shell
1 min of 0: [[0]]	

Subtracting 100 from 50 using `cv2.subtract` returns a value of 0.

But what happens if we use NumPy to perform the arithmetic instead of OpenCV?

Line 26 and 27 explore this question.

First, we define two NumPy arrays, each with a single element, and of the 8-bit unsigned integer data type. The first array has a value of 200 and the second has a value of 100. Using the `cv2.add` function, our addition would be clipped and a value of 255 returned.

However, NumPy does not perform clipping — it instead performs modulo arithmetic and “wraps around.” Once a value of 255 is reached, NumPy wraps around to zero, and then starts counting up again, until 100 steps have been reached. You can see this is true via the first line of output below:

arithmetic.py	Shell
1 wrap around: [44]	

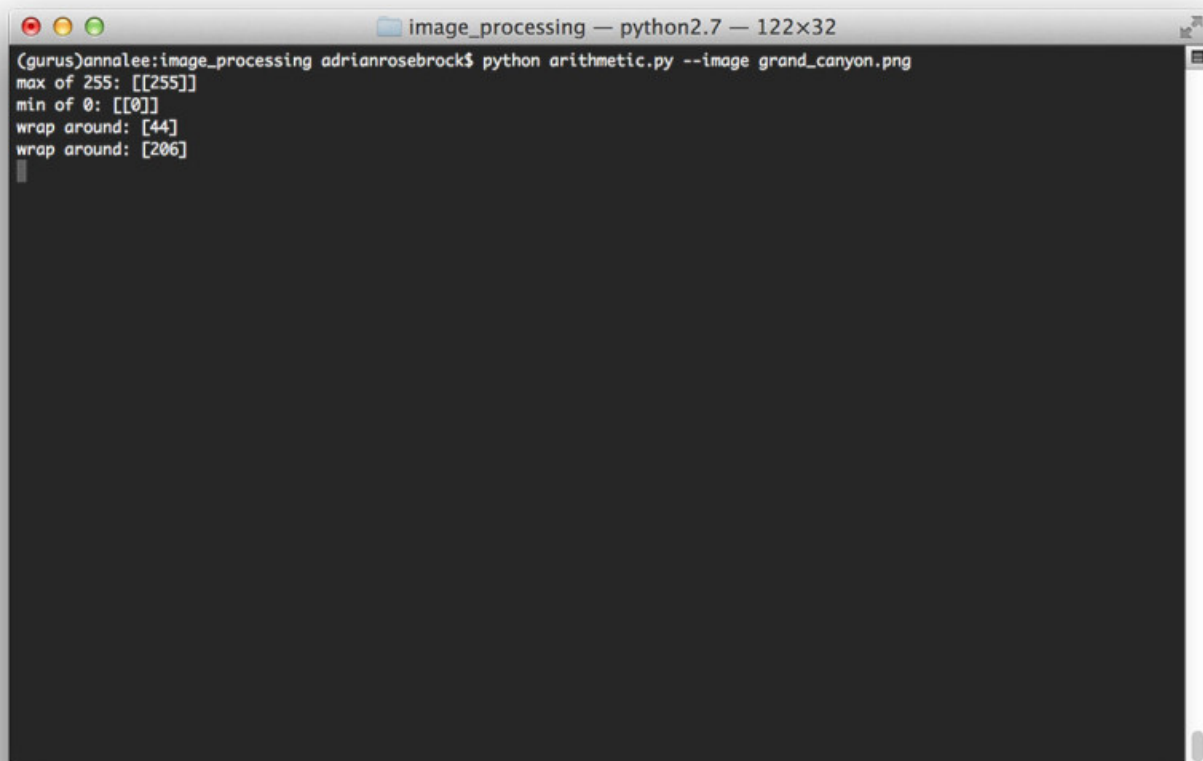
NumPy performs modulo arithmetic rather than clipping. Instead, once 0 is reached during the subtraction, the modulo operations wraps around and starts counting backwards from 255 — we can verify this from the output below:

```
arithmetic.py
```

```
Shell
```

```
1 wrap around: [206]
```

And again, we can confirm the difference between clipping and wrapping around from our terminal output:

A terminal window titled "image_processing — python2.7 — 122x32" shows the execution of a Python script. The prompt is "(gurus)annalee:image_processing adrianrosebrock\$". The command entered is "python arithmetic.py --image grand_canyon.png". The output displayed is: "max of 255: [[255]]", "min of 0: [[0]]", "wrap around: [44]", and "wrap around: [206]".

```
(gurus)annalee:image_processing adrianrosebrock$ python arithmetic.py --image grand_canyon.png
max of 255: [[255]]
min of 0: [[0]]
wrap around: [44]
wrap around: [206]
```

Feedback

(https://gurus.pyimage.com/wp-content/uploads/2015/03/image_arithmetic_terminal.jpg).

FIGURE 1: WHEN PERFORMING IMAGE ARITHMETIC OPERATIONS, OPENCV DOES CLIPPING WHERE NUMPY DOES MODULUS OPERATIONS.

When performing integer arithmetic it is important to keep in mind your desired output.

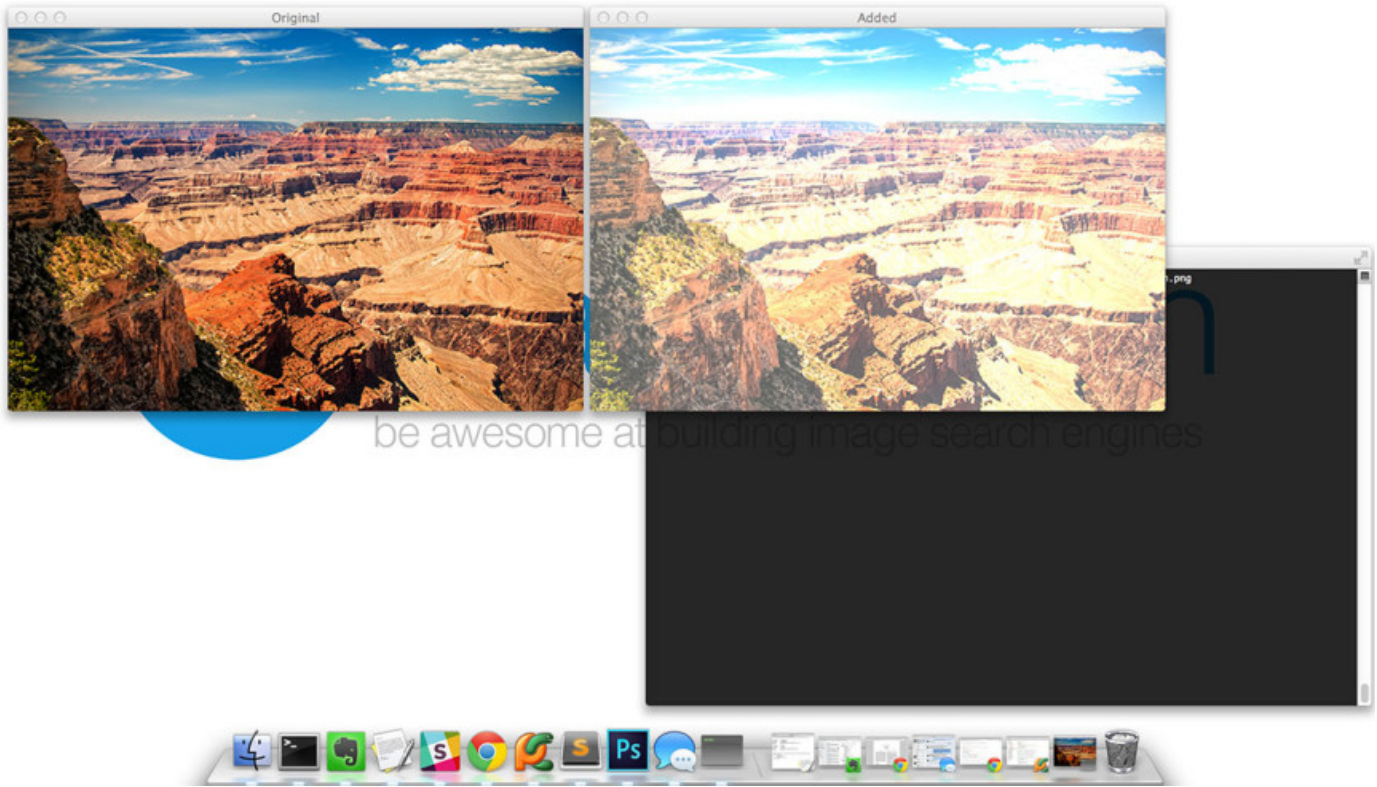
Do you want all values to be clipped if they fall outside the range $[0, 255]$? Then use OpenCV's built in methods for image arithmetic.

Now that we have explored the caveats of image arithmetic in OpenCV and NumPy, let's perform the arithmetic on actual images and view the results:

arithmetic.py	Python
<pre>29 # let's increase the intensity of all pixels in our image by 100 -- we 30 # accomplish this by constructing a NumPy array that is the same size of 31 # our matrix (filled with ones) and the multiplying it by 100 to create an 32 # array filled with 100's, then we simply add the images together; notice 33 # how the image is "brighter" 34 M = np.ones(image.shape, dtype = "uint8") * 100 35 added = cv2.add(image, M) 36 cv2.imshow("Added", added) 37 38 # similarly, we can subtract 50 from all pixels in our image and make it 39 # darker 40 M = np.ones(image.shape, dtype = "uint8") * 50 41 subtracted = cv2.subtract(image, M) 42 cv2.imshow("Subtracted", subtracted) 43 cv2.waitKey(0)</pre>	

Line 34 defines an NumPy array of ones, with the same size as our image. Again, we are sure to use 8-bit unsigned integers as our data type. In order to fill our matrix with values of 100's rather than 1's, we simply multiply our matrix of 1's by 100. Finally, we use the `cv2.add` function to add our matrix of 100's to the original image — thus increasing every pixel intensity in the image by 100, but ensuring all values are clipped to the range `[0, 255]` if they attempt to exceed 255.

The result of our operation can be seen below:



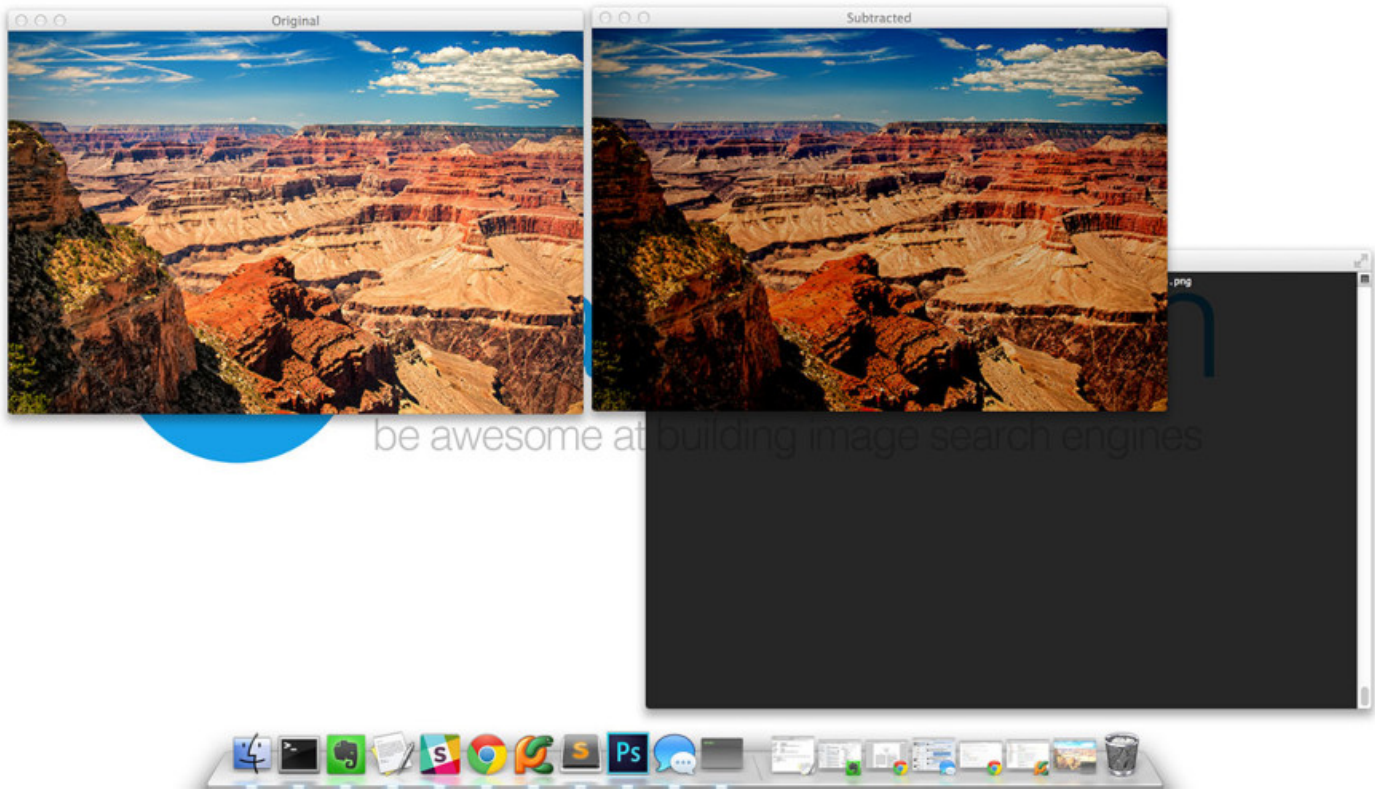
(https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/image_arithmetic_added.jpg).

FIGURE 2: (LEFT) OUR ORIGINAL IMAGE. (RIGHT) ADDING A VALUE OF 100 TO EVERY PIXEL VALUE. NOTICE HOW THE IMAGE NOW LOOKS WASHED OUT.

Notice how the image looks more “washed out” and is substantially brighter than the original. This is because we are increasing the pixel intensities by adding 100 to them and pushing them towards brighter colors.

We then create another NumPy array filled with 50's on **Line 40** and use the `cv2.subtract` function to subtract 50 from each pixel intensity of the image.

The figure below shows the results of this subtraction:



(https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/image_arithmetic_subtracted.jpg).

FIGURE 3: (LEFT) OUR ORIGINAL IMAGE. (RIGHT) SUBTRACTING A VALUE OF 50 FROM EVERY PIXEL VALUE. NOTICE HOW THE IMAGE NOW LOOKS CONSIDERABLY DARKER.

Our image now looks considerably darker than the original photo of the Grand Canyon. Pixels that were once white now look gray. This is because we are subtracting 50 from the pixels and pushing them towards the darker regions of the RGB color space.

Summary

In this section we explored addition and subtraction, two basic (but important) image arithmetic operations. As you can see, image arithmetic operations are simply no more than basic matrix addition and subtraction.

We also explored the peculiarities of image arithmetic using OpenCV and NumPy. These limitations are important to keep in mind, otherwise you may get unwanted results when performing arithmetic operations on your images.

Remember that OpenCV addition and subtraction clips values outside the range $[0, 255]$ to fit inside the range, whereas NumPy performs a modulus operations and “wraps around.” Keeping this in mind will help you avoid tracking down hard to find bugs when coding your own computer vision programs.

Downloads:

Download the Code (https://gurus.pyimagesearch.com/protected/code/computer_vision_basics/arithmetic.zip).

Quizzes		Status
1	Image Arithmetic Quiz (https://gurus.pyimagesearch.com/quizzes/image-arithmetic-quiz/)	

[← Previous Topic \(https://gurus.pyimagesearch.com/topic/cropping/\)](https://gurus.pyimagesearch.com/topic/cropping/) [Next Topic → \(https://gurus.pyimagesearch.com/topic/bitwise-operations/\)](https://gurus.pyimagesearch.com/topic/bitwise-operations/)

Course Progress

Ready to continue the course?

Click the button below to **continue your journey to computer vision guru**.

[I'm ready, let's go! \(/pyimagesearch-gurus-course/\)](/pyimagesearch-gurus-course/)

Feedback

Resources & Links

- [PyImageSearch Gurus Community \(https://community.pyimagesearch.com/\)](https://community.pyimagesearch.com/)
- [PyImageSearch Virtual Machine \(https://gurus.pyimagesearch.com/pyimagesearch-virtual-machine/\)](https://gurus.pyimagesearch.com/pyimagesearch-virtual-machine/)
- [Setting up your own Python + OpenCV environment \(https://gurus.pyimagesearch.com/setting-up-your-python-opencv-development-environment/\)](https://gurus.pyimagesearch.com/setting-up-your-python-opencv-development-environment/)
- [Course Syllabus & Content Release Schedule \(https://gurus.pyimagesearch.com/course-syllabus-content-release-schedule/\)](https://gurus.pyimagesearch.com/course-syllabus-content-release-schedule/)
- [Member Perks & Discounts \(https://gurus.pyimagesearch.com/pyimagesearch-gurus-discounts-perks/\)](https://gurus.pyimagesearch.com/pyimagesearch-gurus-discounts-perks/)
- [Your Achievements \(https://gurus.pyimagesearch.com/achievements/\)](https://gurus.pyimagesearch.com/achievements/)
- [Official OpenCV documentation \(http://docs.opencv.org/index.html\)](http://docs.opencv.org/index.html)

Your Account

9 of 10 [Account Info \(https://gurus.pyimagesearch.com/account/\)](https://gurus.pyimagesearch.com/account/)

- [Support \(https://gurus.pyimagesearch.com/contact/\)](https://gurus.pyimagesearch.com/contact/)

11/30/19, 9:32 PM

1.4.6: Image Arithmetic - PyImageSearch - Gurus <https://gurus.pyimagesearch.com/topic/image-arithmetic/>
https://gurus.pyimagesearch.com/%2Fgurus.pyimagesearch.com%2F&_wpnonce=5736b21cae

 Search

© 2018 PyImageSearch. All Rights Reserved.

Feedback