

Adding (blending) two images using OpenCV

Prev Tutorial: [Operations with images](#)

Next Tutorial: [Changing the contrast and brightness of an image!](#)

Original author	Ana Huamán
Compatibility	OpenCV >= 3.0

We will learn how to blend two images!

Goal

In this tutorial you will learn:

- what is *linear blending* and why it is useful;
- how to add two images using `addWeighted()`

Theory

Note

The explanation below belongs to the book [Computer Vision: Algorithms and Applications](#) by Richard Szeliski

From our previous tutorial, we know already a bit of *Pixel operators*. An interesting dyadic (two-input) operator is the *linear blend operator*:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

By varying α from $0 \rightarrow 1$ this operator can be used to perform a temporal *cross-dissolve* between two images or videos, as seen in slide shows and film productions (cool, eh?)

Source Code

[C++](#) [Java](#) [Python](#)

Download the source code from [here](#).

```
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include <iostream>

using namespace cv;

// we're NOT "using namespace std;" here, to avoid collisions between the beta variable and std::beta in c++17
using std::cin;
using std::cout;
using std::endl;

int main( void )
{
    double alpha = 0.5; double beta; double input;

    Mat src1, src2, dst;

    cout << " Simple Linear Blender " << endl;
    cout << "-----" << endl;
    cout << "* Enter alpha [0.0-1.0]: ";
    cin >> input;

    // We use the alpha provided by the user if it is between 0 and 1
    if( input >= 0 && input <= 1 )
    { alpha = input; }

    src1 = imread( samples::findFile("LinuxLogo.jpg") );
    src2 = imread( samples::findFile("WindowsLogo.jpg") );

    if( src1.empty() ) { cout << "Error loading src1" << endl; return EXIT_FAILURE; }
    if( src2.empty() ) { cout << "Error loading src2" << endl; return EXIT_FAILURE; }

    beta = ( 1.0 - alpha );
    addWeighted( src1, alpha, src2, beta, 0.0, dst);

    imshow( "Linear Blend", dst );
    waitKey(0);

    return 0;
}
```

Explanation

C++ Java Python

Since we are going to perform:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

We need two source images ($f_0(x)$ and $f_1(x)$). So, we load them in the usual way:

```
src1 = imread( samples::findFile("LinuxLogo.jpg") );  
src2 = imread( samples::findFile("WindowsLogo.jpg") );
```

We used the following images: [LinuxLogo.jpg](#) and [WindowsLogo.jpg](#)

Warning

Since we are *adding* *src1* and *src2*, they both have to be of the same size (width and height) and type.

Now we need to generate the $g(x)$ image. For this, the function `addWeighted()` comes quite handy:

```
beta = ( 1.0 - alpha );  
addWeighted( src1, alpha, src2, beta, 0.0, dst);
```

since `addWeighted()` produces:

$$dst = \alpha \cdot src1 + \beta \cdot src2 + \gamma$$

In this case, γ is the argument 0.0 in the code above.

Create windows, show the images and wait for the user to end the program.

```
imshow( "Linear Blend", dst );  
waitKey(0);
```

Result

