实验：图像倍增

实验概要

图像乘法（倍增）

图像乘法与图像相加非常相似，可以使用 OpenCV 的 `cv2.multiply` 函数（推荐）或 NumPy 进行。 推荐使用 OpenCV 的功能是因为与上一节中对 cv2.add 所见的原因相同。我们可以使用 `cv2.multiply` 函数，如下所示：

```
dst = cv2.Mul(src1, src2)
```

此处， `src1` 和 `src2` 指的是我们试图相乘的两个图像，而 `dst` 指的是通过相乘获得的输出图像。

我们已经知道乘法只是重复的加法。由于我们已经看到图像相加具有增加图像亮度的作用，因此图像相乘将具有相同的作用。此处的区别在于效果将是多种多样的。当我们想对亮度进行细微修改时，通常会使用图像加法。

让我们直接通过实验来看看如何使用这些功能。

实验目的

在本实验中，我们将学习如何使用 OpenCV 和 NumPy 将一个常数值与一幅图像相乘，以及如何将两幅图像相乘。

1. 导入依赖库

In [1]:

```
# 导入模块
import cv2                          # 导入OpenCV
import numpy as np                  # 导入NumPy
import matplotlib.pyplot as plt  # 导入matplotlib

# 魔法指令，使图像直接在Notebook中显示
%matplotlib inline
```

2. 加载图像

指定需要执行仿射变换操作的目标图像路径。

您也可以上传自己的图像，需要注意的是确保加载图像路径有效，95% 以上的程序报错,除了缺少安装依赖库以外，大部分就跟数据路径不正确有关。这里使用 `cv2.imread()` 加载图像的路径，往往使用的都是相对路径，应该确保指定了正确的图片文件所在的路径。

In [2]:

```
# 设置输入输出路径
import os
base_path = os.environ.get("BASE_PATH",'../data/')
data_path = os.path.join(base_path + "lab2/")
result_path = "result/"
os.makedirs(result_path, exist_ok=True)

# 读取图像文件
img = cv2.imread("./data/puppy.jpg")
```

使用 Matplotlib 显示图像，输出信息如下。X 轴和 Y 轴分别为图像的宽度和高度:

In [3]:

```
# 显示图像
plt.imshow(img[:,:,::-1]) # 将图像从BGR转换为RGB
plt.show()                # 显示图像
```

3. 使用 OpenCV 执行图像乘法运算

使用 `cv2.multiply` 函数将图像 `img` 乘以 `2`：

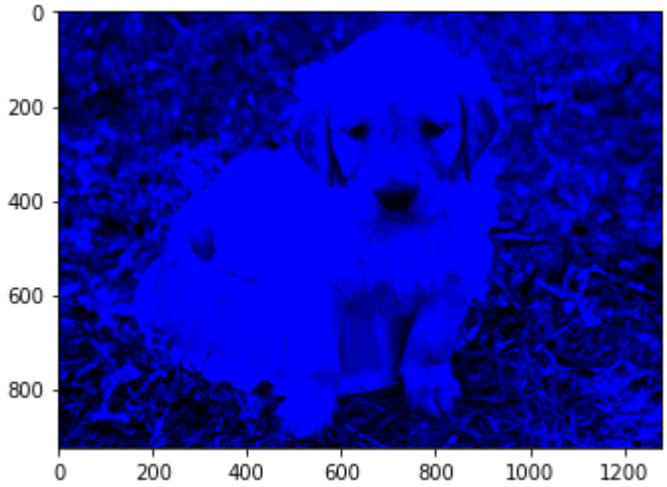
In [4]:

```
cvImg = cv2.multiply(img,2) #将图像乘以2
```

使用 Matplotlib 显示图像，输出信息如下。X 轴和 Y 轴分别为图像的宽度和高度:

In [5]:

```
plt.imshow(cvImg[:,:,::-1]) # 将图像从BGR转换为RGB
plt.show()                  # 显示图像
```



您能想到为什么在输出图像中获得了很高的蓝色调吗?

您可以使用之前实验中有关图像加法的说明作为参考。

4. 使用 NumPy 执行图像乘法运算

我们对比一下使用 NumPy 倍增图像的效果:

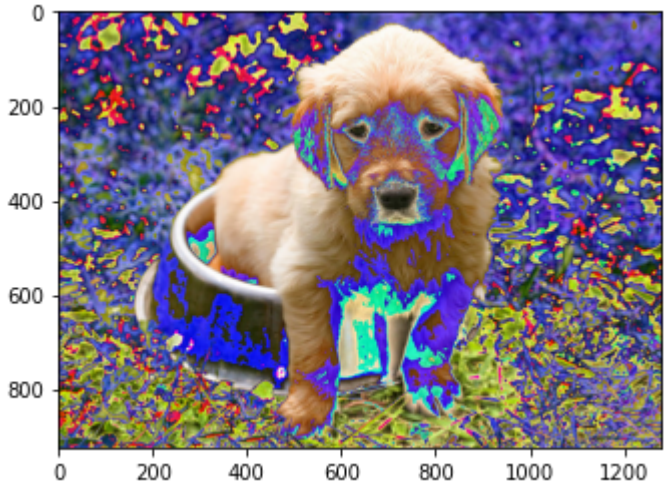
In [6]:

```
npImg = img*2 # 使用NumPy倍增图像
```

使用 Matplotlib 显示图像，输出信息如下。X 轴和 Y 轴分别为图像的宽度和高度:

In [7]:

```
plt.imshow(npImg[:,:,::-1]) # 将图像从BGR转换为RGB
plt.show()                  # 显示图像
```



由于各个色彩空间的像素值均被取模运算，图像的成像效果很混乱，因此不建议使用 NumPy 执行图像运算。

5. 多张图像相乘

In [8]:

```
img.shape # 输出图像的形状
```

Out[8]:

```
(924, 1280, 3)
```

创建一个像素值被全 2 填充的新数组（图像），该数组 `nparr` 与原始图像 `img` 的形状相同

In [9]:

```
# 使用NumPy对全1数组进行乘2，获得一个全2填充的数组（图像）
nparr = np.ones((924,1280,3),dtype=np.uint8) * 2
```

使用 `cv2.multiply` 函数，将这个新数组 `nparr` 与原始图像 `img` 相乘，并比较所获得的结果：

In [10]:

```
cvImg = cv2.multiply(img,nparr) # 使用多张图像相乘实现倍增
```

使用 Matplotlib 显示图像，输出信息如下。X 轴和 Y 轴分别为图像的宽度和高度:

In [11]:

```
plt.imshow(cvImg[:,:,::-1]) # 将图像从BGR转换为RGB
plt.show()                  # 显示图像
```

实验小结

我们知道乘法只是重复的加法，因此，使用乘法获得更亮的图像是有意义的，因为我们也使用加法获得了更亮的图像。

到目前为止，我们已经讨论了几何变换和图像运算。之后，我们将进入有关在图像上执行按位运算的更高级的主题。