案例：实现复杂的镜面效果

案例目标

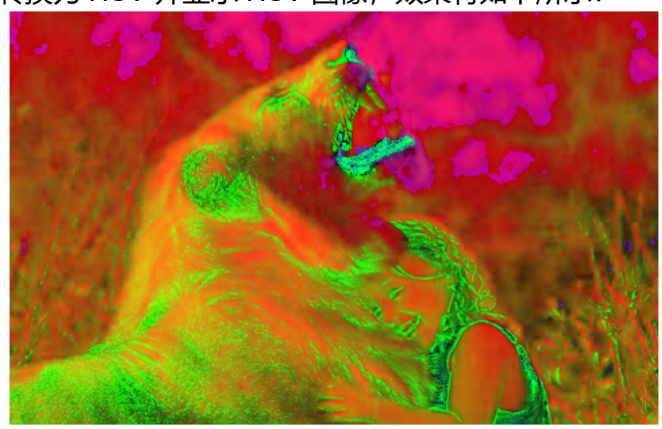
通过完成以下案例来测试到目前为止学到的知识。我们将创建镜像效果 —— 水面效果和镜面效果图像之间的区别是镜面效果将横向反转。此外，我们还将为镜像引入其他负片效果。这种负片效果的名称来自处理照片时使用的负像。通过查看下图，您可以看到镜像的效果。创建一个简单的镜像效果非常简单，根据下图所示的效果对我们的图像进行操作。这种技术在我们要创建美图秀秀，Instagram 等应用程序时，这些效果非常有用。例如，水面效果，镜面效果等非常常用的滤镜。 我们在之前的实验中介绍了水面效果，现在，我们将创建一个镜像效果：

在阅读详细说明之前，请考虑如何创建这种效果。注意图像中的对称性，即镜面效果。本案例最重要的部分是在右侧生成图像。我们将使用与先前实验中的狮子和女孩的图像。

当然，我们鼓励您上传自己的图像来进行练习！

案例操作详细说明

1. 加载所需的库：OpenCV，Matplotlib 和 NumPy。
2. 编写 magic 命令以在 Jupyter Notebook 中显示图像。
3. 加载图像，并使用 Matplotlib 显示。
4. 或取图像的形状。
5. 将图像的色彩空间从 BGR 转换为 HSV 并显示 HSV 图像，效果将如下所示：

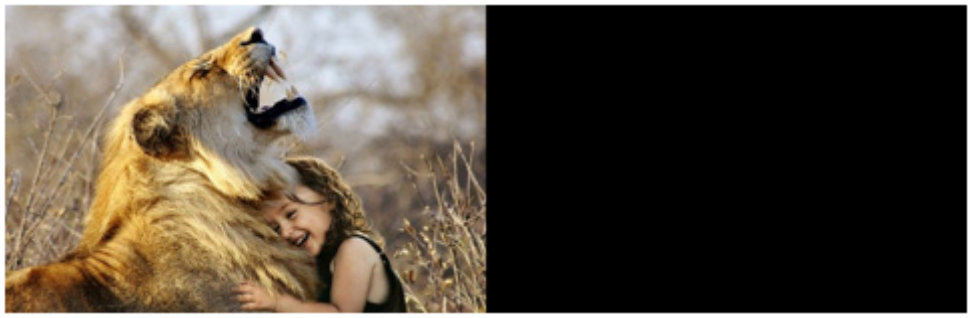


6. 从 HSV 颜色空间中提取值通道。请注意， `亮度` 通道（Value）是 HSV 图像的最后一个通道。您可以为此使用 `cv2.split` 函数显示 `明度` 通道。该图像将如下所示：

7. 我们将对图像加入负片效果。这类似于您在单击的图像的底片中看到的内容。要执行此效果，您要做的就是从 `255` 中减去 `明度` 通道。然后，显示新的 `明度` 通道。 该图像将如下所示：

8. 将 `亮度` 通道与其自身合并来创建新图像。这可以通过使用 `cv2.merge((value, value, value))` 来完成，其中 `value` 表示您在上一步中获得的 `亮度` 通道的负数。之所以这样做，是因为我们要合并两个三通道图像以创建最终效果。
9. 水平翻转之前获得的新图像。您可以参考我们前面创建水面效果中所做的翻转步骤。请注意，水平翻转等效于反转图像列的顺序。输出将如下所示：

10. 创建宽度为原始图像两倍的新图像。这意味着行数和通道数将保持不变。仅列数将加倍。
11. 复制原始图像到新图像的左半部分。图像如下所示:



12. 将水平翻转的图像复制到新图像的右半部分并显示该图像。

13. 最后，保存获取到的图像。最终的图像如下所示:

案例小结

我们从讨论计算机视觉的重要核心概念开始：什么是图像，什么像素以及它们的属性是什么。然后，我们讨论了在整个计算机视觉之旅中将使用的库：OpenCV，Matplotlib 和 NumPy。我们还学习了如何使用这些库来读取，处理，显示和保存图像。最后，我们学习了如何访问和操作像素，并使用此概念在最终的案例中生成了有趣的结果。在本案例中，您使用了在前面的实验中研究的概念，生成一个非常有趣的结果。通过完成此案例，您已经学习了如何转换图像的色彩空间以及如何使用新图像的特定通道来创建镜像效果。

案例答案

In [1]:

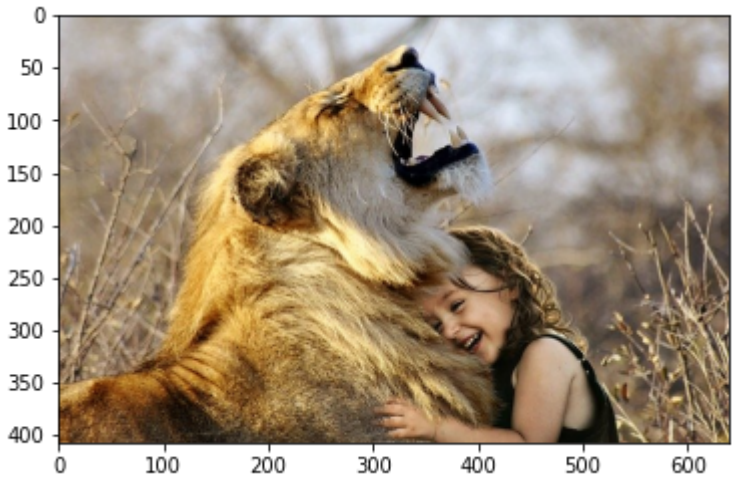
```
# 加载依赖库
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
# 编写 magic 命令以在 Jupyter Notebook 中显示图像
%matplotlib inline
```

In [3]:

```
# 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 + "lab1/")
result_path = "result/"
os.makedirs(result_path, exist_ok=True)

# 加载图像
img = cv2.imread("./data/lion.jpg")

# 显示图像
plt.imshow(img[:,:,::-1])
plt.show()
```



In [4]:

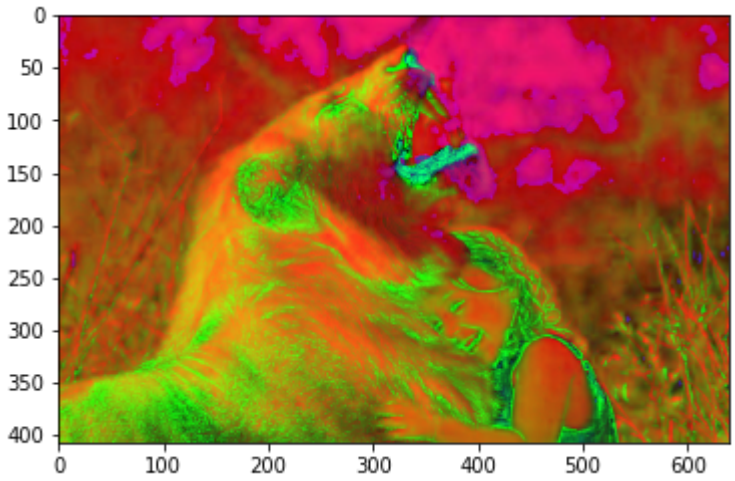
```
# 获取图像形状
img.shape
```

Out[4]:

```
(407, 640, 3)
```

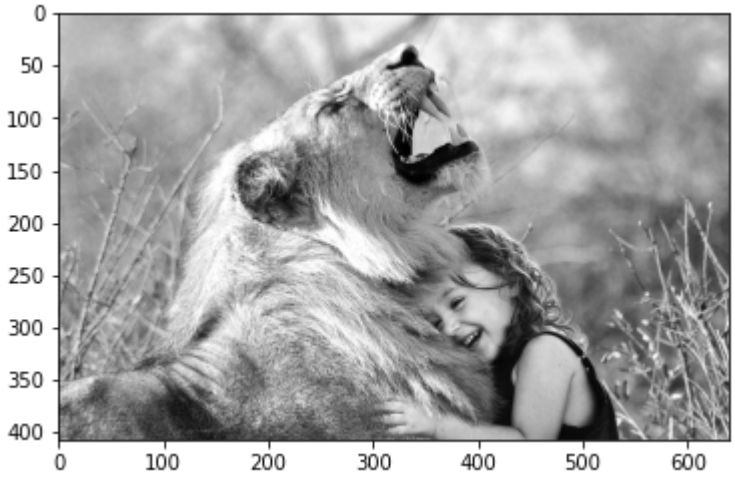
In [5]:

```
# 转换图像色彩空间为HSV
imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# 显示图像
plt.imshow(imgHSV[:,:,::-1])
plt.show()
```



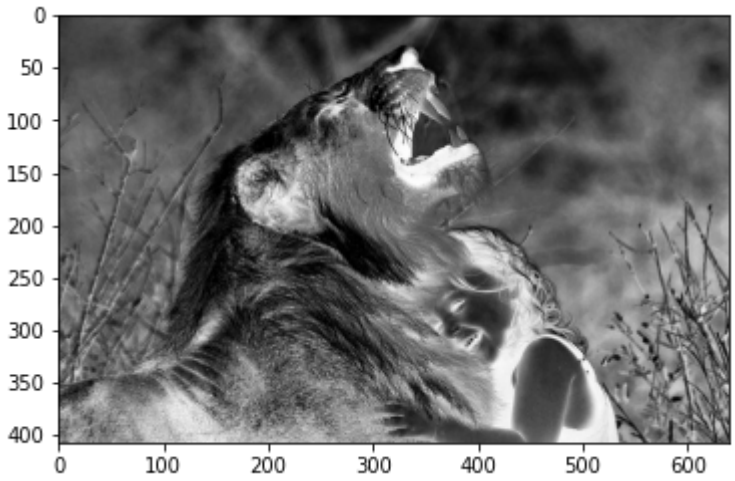
In [6]:

```
# 分离色相、饱和度、明度通道
hue, sat, val = cv2.split(imgHSV)

# 显示图像
plt.imshow(val, cmap="gray")
plt.show()
```



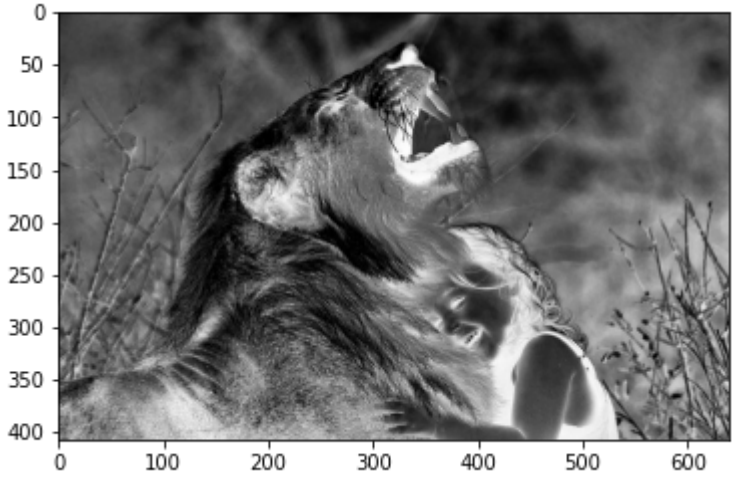
In [7]:

```
# 使用像素最大值255，减去当前明度数值，获得负片效果
val = 255-val

# 显示图像
plt.imshow(val, cmap="gray")
plt.show()
```



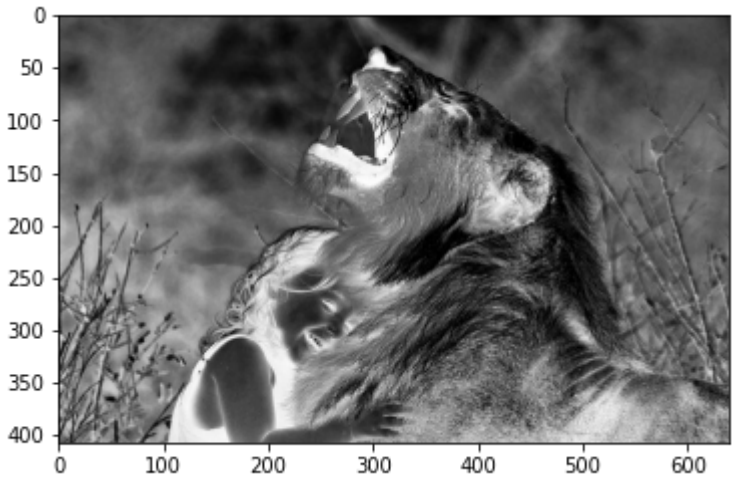
In [8]:

```
# 合并明度通道
imgHSV = cv2.merge((val,val,val))

# 显示图像
plt.imshow(imgHSV[:,:,::-1])
plt.show()
```



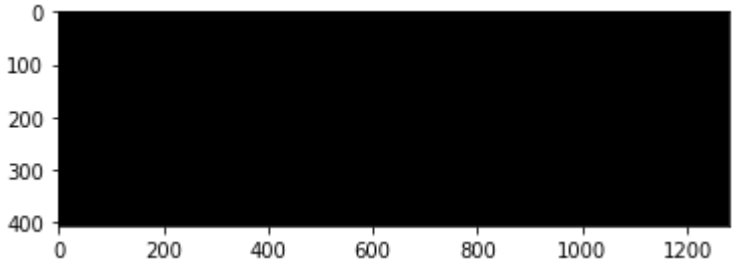
In [9]:

```
# 水平翻转图像
imgHSV = imgHSV[:,::-1,:]

# 显示图像
plt.imshow(imgHSV[:,:,::-1])
plt.show()
```



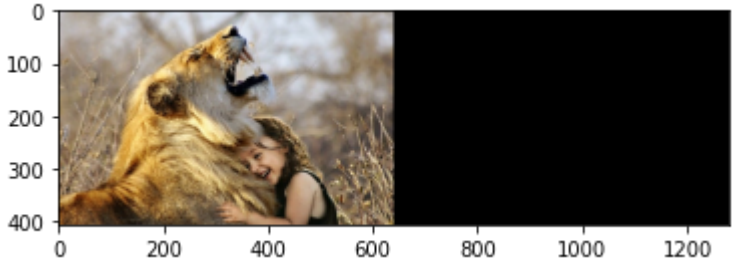
In [10]:

```
# 创建一幅新的图像
imgNew = np.zeros((407,640*2,3),dtype=np.uint8)

# 显示图像
plt.imshow(imgNew[:,:,::-1])
plt.show()
```



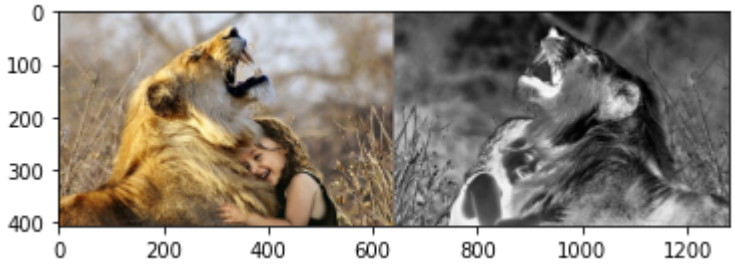
In [11]:

```
# 将img复制到新图像的左半部分
imgNew[:,:640,:] = img

# 显示图像
plt.imshow(imgNew[:,:,::-1])
plt.show()
```



In [12]:

```
# 将imgHSV复制到新图像的右半部分
imgNew[:,640:,:] = imgHSV

# 显示图像
plt.imshow(imgNew[:,:,::-1])
plt.show()
```



In [13]:

```
# 保存完成的图像
cv2.imwrite(result_path+"mirror.png",imgNew)
```

Out[13]:

True