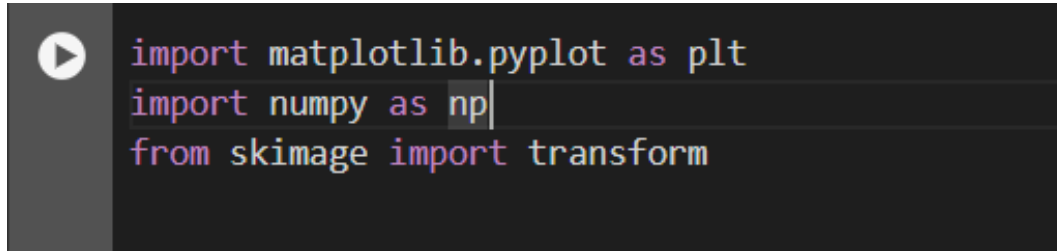


Pattern Recognition Homework 1 / 152120191023_SelimCanYazar

A screenshot of a code editor with a dark background. On the left, there is a play button icon. The code is written in a light-colored font and consists of three lines: 'import matplotlib.pyplot as plt', 'import numpy as np', and 'from skimage import transform'.

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import transform
```

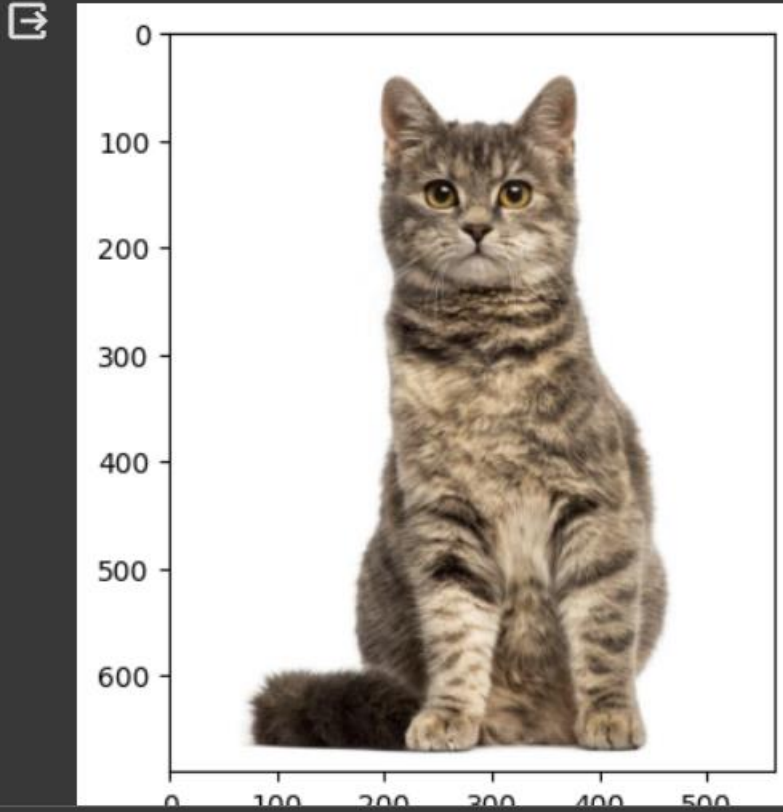
1.matplotlib.pyplot: Bu kütüphane, grafik ve görselleştirme işlemleri için kullanılır. plt olarak kısaltılmış şekilde içe aktarılmış, böylece kodun ilerleyen kısımlarında bu kütüphanenin fonksiyonlarına plt yazarak erişebiliriz. Örneğin, resimleri göstermek veya grafikler çizmek için bu kütüphaneyi kullanırız.

2.numpy: Bu, Python'da bilimsel hesaplamalar için çok kullanılan bir kütüphanedir. Diziler, matrisler ve bu tür veri yapıları üzerinde hızlı ve etkili işlemler yapmamızı sağlar. np olarak kısaltılmış şekilde içe aktarılır, böylece np kullanarak bu kütüphanenin fonksiyonlarına ve özelliklerine erişebiliriz.

3.skimage (scikit-image): Bu kütüphane, görüntü işleme ve analizi için kullanılır. Özellikle, transform modülü görüntüleri dönüştürme ve boyutlandırma gibi işlemler için gerekli araçları sağlar. Bu kodda, from skimage import transform kullanılarak, sadece transform modülünü içe aktarıyoruz. Bu şekilde, bu modülde yer alan fonksiyonlara doğrudan transform yazarak ulaşabiliriz.

```
# Resmi oku
img1 = plt.imread('cat1.jpg')

# Resmi göster
plt.imshow(img1)
plt.show()
```



1.`img1 = plt.imread('cat1.jpg')`: `plt.imread` fonksiyonu, 'cat1.jpg' adlı bir görüntü dosyasını okuyoruz ve bu görüntüyü bir numpy dizisi olarak `img1` değişkenine atıyoruz. Bu dizi, görüntünün piksel değerlerini içerir.

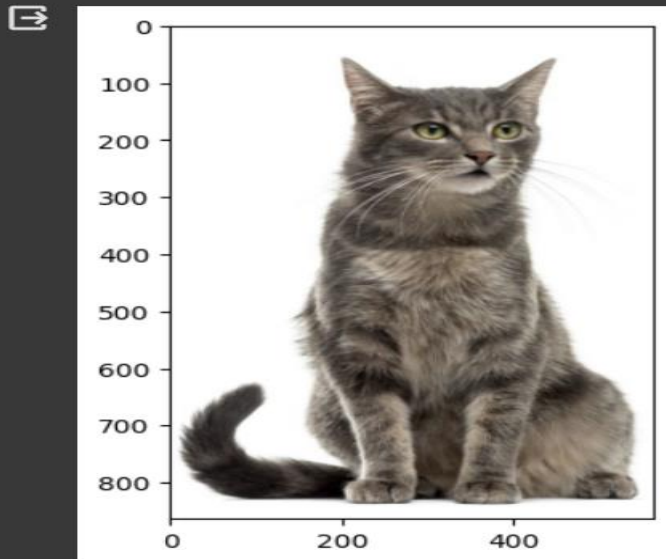
2.`plt.imshow` fonksiyonu, `img1` değişkeninde saklanan görüntüyü göstermek için kullanılır. Bir numpy dizisini alır ve bu diziye bir görüntü olarak görselleştirir. Görüntü bir pencerede gösterilir ama henüz ekrana çıkartılmaz.

3.`plt.show()`: Bu satır `plt.imshow` ile oluşturulan görüntüyü ekranda gösterir. `plt.show()` komutu çağrıldığında, öncesinde oluşturulan tüm grafikler ve görüntüler ekrana çıkarılır.

*Yukarıdaki aynı kod sadece resim yolu ve değişken adı değiştirilerek diğer kedi ve köpek resimleri içinde kullanılmıştır.

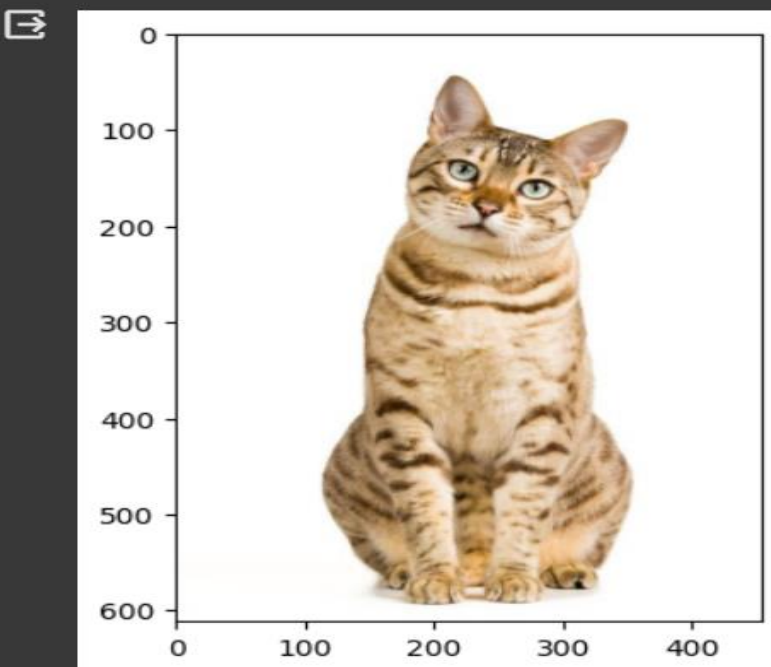
```
# Resmi oku
img2 = plt.imread('cat2.jpg')

# Resmi göster
plt.imshow(img2)
plt.show()
```



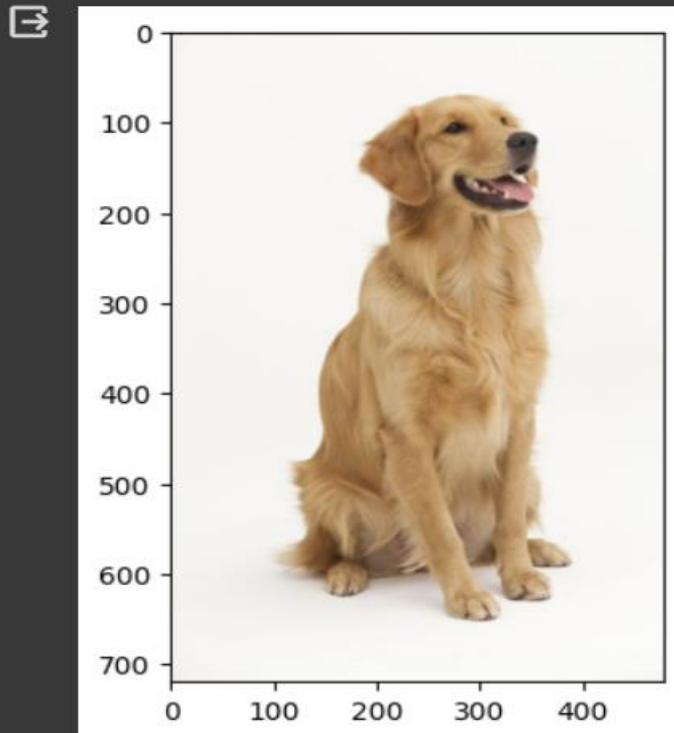
```
# Resmi oku
img3 = plt.imread('cat3.jpg')

# Resmi göster
plt.imshow(img3)
plt.show()
```



```
# Resmi oku
img4 = plt.imread('dog1.jpg')

# Resmi göster
plt.imshow(img4)
plt.show()
```



TASK 1

```
import numpy as np
import matplotlib.pyplot as plt
import imageio

# Load the image
img = imageio.imread('color_template.png')

# Display the original image
plt.imshow(img)
plt.title('Original Image')
plt.show()

# Updated transformation matrix from RGB to YIQ using the provided values
RGB_TO_YIQ = np.array([
    [0.2126, 0.7152, 0.0722],
    [-0.09991, -0.33609, 0.436],
    [0.615, -0.55861, -0.05639]
])
```

```

# Updated transformation matrix from YIQ to RGB using the provided values
YIQ_TO_RGB = np.array([
    [1, 0, 1.28033],
    [1, -0.21482, -0.38059],
    [1, 2.12798, 0]
])

# Apply the RGB to YIQ transformation
yiq_img = np.zeros_like(img, dtype=float)
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        yiq_img[i, j, :] = np.dot(RGB_TO_YIQ, img[i, j, :]/255.0)

# Display the YIQ image (note: it won't look correct as it's not in RGB space)
plt.imshow(yiq_img)
plt.title('YIQ Image')
plt.show()

# Apply the YIQ to RGB transformation
rgb_converted_img = np.zeros_like(img)
for i in range(yiq_img.shape[0]):
    for j in range(yiq_img.shape[1]):
        rgb_converted_img[i, j, :] = np.clip(np.dot(YIQ_TO_RGB, yiq_img[i, j, :]), 0, 1) * 255

rgb_converted_img = rgb_converted_img.astype(np.uint8)

# Display the converted image
plt.imshow(rgb_converted_img)
plt.title('RGB Converted Image')
plt.show()

```

1.İlk önce numpy ve matplotlib.pyplot kütüphanelerini yüklüyoruz ki matematiksel işlemler ve görselleştirme yapabilelim. imageio kütüphanesi de resimleri okumak için kullanılıyor.

2.`img = imageio.imread('color_template.png')`: Bu satırda 'color_template.png' isimli bir resim dosyasını yükleyip `img` isimli bir değişkene atıyoruz. Artık bu değişken üzerinden resme erişebilir ve işlemler yapabiliriz.

3.Sonra, orijinal resmi gösteriyoruz. `plt.imshow(img)` ile resmi ekrana getiriyoruz ve `plt.title('Original Image')` ile üstüne "Original Image" başlığını koyuyoruz. `plt.show()` ile de bu görseli ekrana basıyoruz.

4.RGB'den YIQ'ya Dönüşüm Matrisi (`RGB_TO_YIQ`): Bu, renkleri RGB (Kırmızı, Yeşil, Mavi) formatından YIQ formatına çevirmek için kullanılan bir tablodur. Her bir satır ve sütun, bu dönüşüm sırasında renk bileşenlerinin nasıl karıştırılacağını belirtir.

5.YIQ'tan RGB'ye Dönüşüm Matrisi (`YIQ_TO_RGB`): Bu da YIQ formatındaki renkleri RGB formatına çevirmek için kullanılan bir tablodur. Yani birinci matrisin yaptığıının tam tersini yapar.

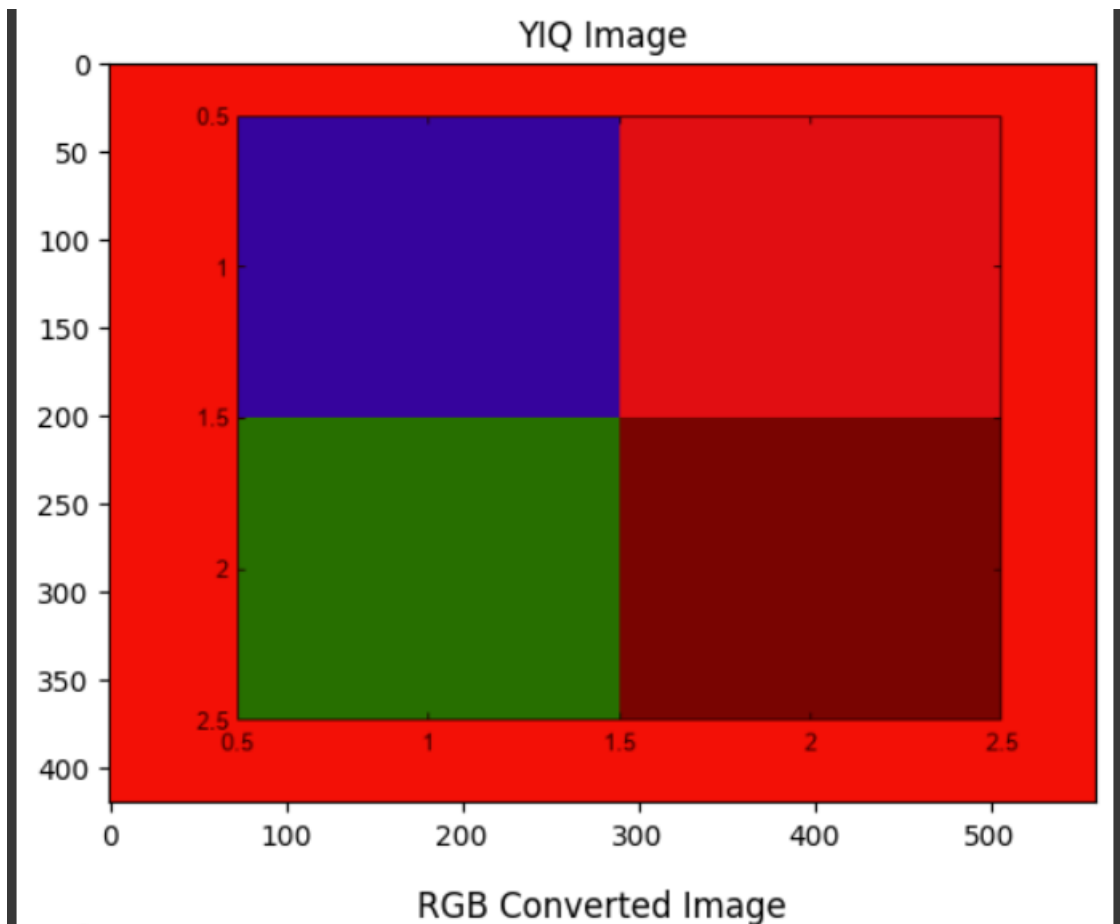
6.RGB'den YIQ'ya Dönüşüm (np.zeros_like ve np.dot Kullanımı):

`np.zeros_like(img, dtype=float)` ile `img` ile aynı boyutlarda ama içi sıfırlarla dolu bir dizi oluşturuyoruz. Bu dizi, YIQ değerlerini saklayacak. Daha sonra, her bir piksel için `np.dot(RGB_TO_YIQ, img[i, j, :]/255.0)` işlemiyle RGB'den YIQ'ya dönüşüm yapıyoruz. Burada `np.dot` fonksiyonu, matris çarpımı yapmamızı sağlıyor, yani `RGB_TO_YIQ` matrisi ile her bir pikselin RGB değerini çarpıp YIQ değerini buluyoruz.

7.YIQ Görüntüsünü Gösterme: Elde ettiğimiz YIQ değerlerini içeren görüntüyü `plt.imshow(yiq_img)` ile ekranda gösteriyoruz. Bu fonksiyon, elde ettiğimiz YIQ değerlerini içeren görüntüyü bir grafik olarak çizdirir. "YIQ Image" başlığını da `plt.title` ile ekliyoruz. `plt.show()` fonksiyonuna değinecek olursak bu fonksiyon, matplotlib kütüphanesinin grafikleri göstermek için kullandığı bir fonksiyondur.`plt.imshow()` ile oluşturulan görselleri ve `plt.title()` ile eklenen başlıkları kullanıcıya göstermek için `plt.show()` fonksiyonunu çağırmak gerekir. Eğer bu fonksiyonu çağırmazsak, grafikler yalnızca arka planda oluşturulur ve kullanıcıya gösterilmez. Dolayısıyla, görselleştirme işlemlerini tamamladıktan sonra, oluşturulan tüm grafikleri ekrana getirmek ve kullanıcının görmesini sağlamak için `plt.show()` fonksiyonunu kullanırız.

8.YIQ'tan RGB'ye Dönüşüm (np.clip Kullanımı): YIQ'tan RGB'ye dönüşüm yaparken, her bir piksel için `np.dot(YIQ_TO_RGB, yiq_img[i, j, :])` işlemiyle matris çarpımı yapıyoruz. Sonuç olarak elde edilen değerler 0 ile 1 arasında olmalı, bu yüzden `np.clip` fonksiyonu ile değerleri bu aralığa sınırlandırıyoruz. Sonrasında bu değerleri 255 ile çarparak normal RGB değerlerine dönüştürüyoruz.

9.RGB'ye Dönüştürülmüş Görüntüyü Gösterme: Son olarak, YIQ'tan RGB'ye çevirdiğimiz görüntüyü ekranda gösteriyoruz. Burada da `plt.imshow` ile görüntüyü ve `plt.title` ile "RGB Converted Image" başlığını ekliyoruz. `plt.show()` komutu ile de kullanıcının görmesini sağlıyoruz.



TASK 2

```
# Vertical Flip
img_vertical_flip = np.flipud(img4)

# Horizontal Flip
img_horizontal_flip = np.fliplr(img4)

# Rotate Left by 90 degrees
img_rotate_left = np.rot90(img4, k=1, axes=(0, 1))

# Rotate Right by 90 degrees
img_rotate_right = np.rot90(img4, k=3, axes=(0, 1))

# Resize to Half
img_resized = transform.rescale(img4, 0.5, preserve_range=False, multichannel=True)

# Display input and output images
plt.figure(figsize=(10, 10))
```

```

# Updated transformation matrix from YIQ to RGB using the provided values
YIQ_TO_RGB = np.array([
    [1, 0, 1.28033],
    [1, -0.21482, -0.38059],
    [1, 2.12798, 0]
])

# Apply the RGB to YIQ transformation
yiq_img = np.zeros_like(img, dtype=float)
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        yiq_img[i, j, :] = np.dot(RGB_TO_YIQ, img[i, j, :]/255.0)

# Display the YIQ image (note: it won't look correct as it's not in RGB space)
plt.imshow(yiq_img)
plt.title('YIQ Image')
plt.show()

# Apply the YIQ to RGB transformation
rgb_converted_img = np.zeros_like(img)
for i in range(yiq_img.shape[0]):
    for j in range(yiq_img.shape[1]):
        rgb_converted_img[i, j, :] = np.clip(np.dot(YIQ_TO_RGB, yiq_img[i, j, :]), 0, 1) * 255

rgb_converted_img = rgb_converted_img.astype(np.uint8)

# Display the converted image
plt.imshow(rgb_converted_img)
plt.title('RGB Converted Image')
plt.show()

```

1. Dikey Çevirme (Vertical Flip): `np.flipud(img4)` kullanılarak, resmi dikey olarak çeviriyoruz. Yani resmin üstü alt oluyor, altı üst oluyor bu sayede tam bir ayna etkisi gibi yaratılıyor.
2. Yatay Çevirme (Horizontal Flip): `np.fliplr(img4)` ile resmi yatay olarak çeviriyoruz. Bu da resmin sağına sol yapar, solunu sağ yapar yine bir ayna etkisi ama bu sefer yatayda.
3. Sola Döndürme (Rotate Left): `np.rot90(img4, k=1, axes= (0, 1))` ile resmi sola, yani saat yönünün tersine, 90 derece döndürüyoruz.
4. Sağa Döndürme (Rotate Right): `np.rot90(img4, k=3, axes= (0, 1))` ile resmi sağa, yani saat yönünde, 90 derece döndürüyoruz. Bu, aslında resmi 270 derece sola döndürmekle aynı şey.
5. Yeniden Boyutlandırma (Resize): `transform.rescale(img4, 0.5, preserve_range=False, multichannel=True)` ile resmin boyutunu yarı yarıya küçültüyoruz. Yani hem genişliği hem de yüksekliği orijinalinin yarısı olacak şekilde ayarlıyoruz.
6. Görselleştirme: Bu işlemlerin her birinin sonucunu görmek için bir sürü `plt.subplot` ve `plt.imshow` kullanıyoruz. Her bir alt grafik subplot farklı bir dönüşümü gösteriyor yani orijinal resim, dikey çevrilmiş, yatay çevrilmiş, sola döndürülmüş, sağa döndürülmüş ve yeniden boyutlandırılmış hallerini ekrana basıyoruz.

