Write a program to convert the image "fruit.jpg" to its complementary colors and display it on the screen.

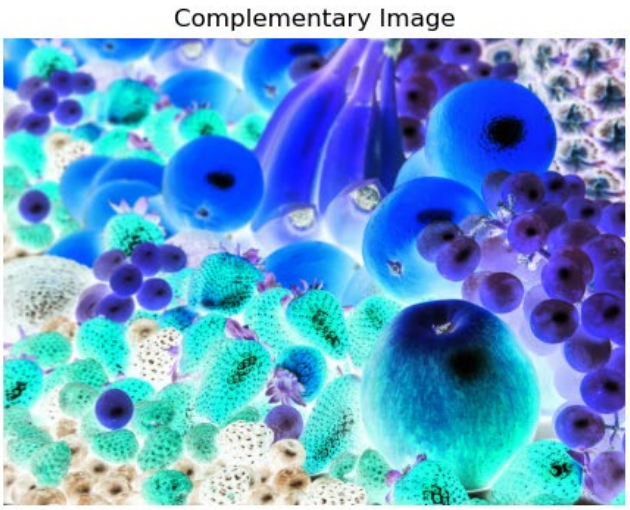
Code:

```python
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
image_array = np.array(Image.open('../original_images/fruit.jpg'))
# Calculate the complementary colors
complementary_image_array = 255 - image_array

# Convert the complementary color array back to an image
complementary_image = Image.fromarray(complementary_image_array)

# Display the original and complementary images
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(complementary_image)
plt.title('Complementary Image')
plt.axis('off')
plt.show()
complementary_image.save("./OUTPUT/fruit_complementary.jpg")
```

ได้ผลลัพธ์เป็นดังนี้



ภาพ complementary ที่ได้มีการเปลี่ยนสีจากสีแดงกลายเป็นสีฟ้าอมเขียว สีส้มเหลืองกลายเป็นสีน้ำเงิน สีเขียวกลายเป็นสีม่วง และสีดำกลายเป็นสีขาว ซึ่งถ้าดูจากวงล้อสีคร่าวก็จะพบว่าเป็นเช่นนั้นจริงๆ

การปรับแต่งภาพประเภทนี้สามารถใช้เพื่อเน้นคุณสมบัติหรือรายละเอียดบางอย่างในภาพที่อาจสังเกตเห็นได้น้อยในโทนสีดั้งเดิม คิดว่าน่าจะมักใช้ในด้านการถ่ายภาพทางดาราศาสตร์หรือการแพทย์ ส่วนในภาพผลไม้ที่ได้ค่อนข้างแปลกตามากๆ เพราะสีของผลไม้ไม่ตรงกับความเป็นจริงที่เรารับรู้ทุกวัน

Use color slicing to segment the oranges from the image "oranges.jpg" by keeping the colors of the oranges to be same but changing the colors of other fruits to blue color (0,0,0.5). Find the most suitable range of orange color using your judgement.

Code:

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
# To segment the oranges using HSI color space, we need to define a custom
function because HSI is not natively supported in common libraries.

def rgb_to_hsi(rgb_image_array):
    """
    Convert an RGB numpy array to HSI. This is an approximation since true HSI
conversion
    is complex and involves conditional transformations.

    Parameters:
    rgb_image_array: numpy array of an image with RGB values
    """

    # Normalize the RGB values to the range 0-1
    rgb_image_array = rgb_image_array / 255.0
    R, G, B = rgb_image_array[:,:,0], rgb_image_array[:,:,1],
rgb_image_array[:,:,2]

    # Calculate intensity
    I = (R + G + B) / 3.0

    # Calculate saturation
    min_RGB = np.minimum(np.minimum(R, G), B)
    S = 1 - (3 / (R + G + B + 1e-12)) * min_RGB

    # Calculate hue
    num = 0.5 * ((R - G) + (R - B))
    den = np.sqrt((R - G)**2 + (R - B) * (G - B))
    theta = np.arccos(num / (den + 1e-12))

    H = theta
    H[B > G] = 2 * np.pi - H[B > G]
    H = H / (2 * np.pi)  # Normalize to the range 0-1

    # Combine H, S, and I into one array
    hsi_image_array = np.dstack((H, S, I))
    return hsi_image_array
```

(มีcodeต่อหน้าถัดไป)

```
# Load the uploaded image
uploaded_image_path = '../original_images/oranges.jpg'
oranges_image = Image.open(uploaded_image_path)

# Convert the image to numpy array
oranges_image_array = np.array(oranges_image)

# Convert RGB to HSI
hsi_oranges_image = rgb_to_hsi(oranges_image_array)
# Define a range for the color of oranges in the HSI color space
# These values are approximations and can be tuned
# Hue for orange is around 30 degrees or 0.083 in normalized scale
hsi_orange_min = np.array([0.05, 0.1, 0.2])  # HSI minimum range for orange
color
hsi_orange_max = np.array([0.15, 1, 1])  # HSI maximum range for orange color

# Create a mask for orange color
hsi_orange_mask = np.all(hsi_oranges_image >= hsi_orange_min, axis=-1) &
np.all(hsi_oranges_image <= hsi_orange_max, axis=-1)

# Create an image with blue color where the orange is not present
hsi_segmented_image_array = oranges_image_array.copy()
hsi_segmented_image_array[~hsi_orange_mask] = [0, 0, int(0.5 * 255)]  # RGB
value for the blue color

# Convert the array back to an image
hsi_segmented_image = Image.fromarray(hsi_segmented_image_array)

# Display the original and HSI segmented images
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(oranges_image)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(hsi_segmented_image)
plt.title('HSI Segmented Image')
plt.axis('off')

plt.show()

# Save the HSI segmented image to the OUTPUT folder
hsi_segmented_image.save('./OUTPUT/segmented_image.jpg')
```

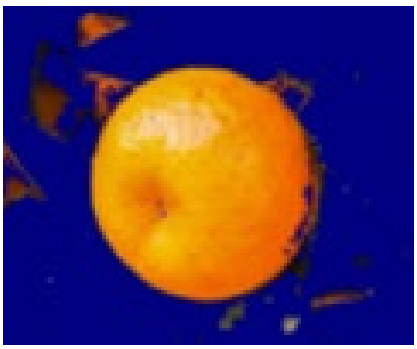
จาก code เราทำการแปลงภาพให้เป็นแบบ HSI ก่อนเพื่อความง่ายในการหาช่วงของสี ซึ่งเราได้ว่าค่าที่น่าจะดีที่สุดในการเอาส่วนของสีส้มออกมาคือ H = (0.055,0.15) S = (0,1) และ I = (0.2,1) ซึ่งได้ออกมาเป็นดังนี้

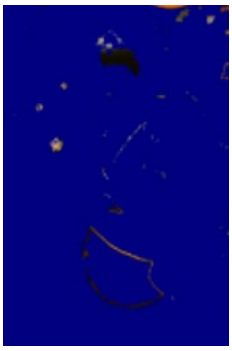
ด้านล่างจะเป็นวิธีและข้อสังเกตต่างๆที่ผมใช้ในการหาช่วงของ HSI ออกมา

อย่างแรก เรารู้ว่าสีส้มค่าของ H จะเท่ากับ 30° ซึ่งเท่ากับ 0.083 ใน normalize scale ดังนั้นในครั้งแรกเราลองกำหนดให้ H มีค่าอยู่ในช่วง 30°±10° ซึ่งจะได้ค่าใน normalize scale คือ 0.083±0.028 = (0.055,0.111)
ในส่วนของ S และ I นั้นจะกำหนดไว้ให้กว้างที่สุดคือ (0,1) จะได้ภาพออกมาเป็นดังนี้

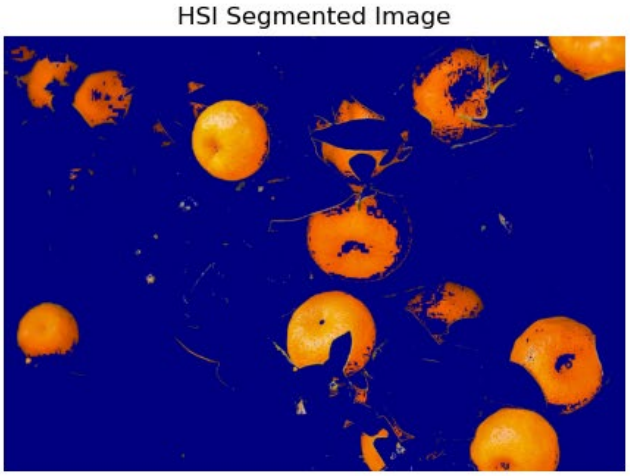
จะเห็นว่าเราได้สีส้มมาเพียงอย่างเดียว แต่เนื่องจากตัวผลของส้มอาจจะมีแสงสะท้อนทำให้มีสีส้มออกเหลืองมาด้วยนิดหน่อย ทำให้เราต้องขยายช่วงให้ครอบคลุมช่วงสีแถวๆสีเหลืองด้วย ซึ่งก็คือจะต้องขยายไปในทางขวาของscale เราจึงลองเปลี่ยนค่า 0.111 ไปเป็นค่าที่ 0.15 ซึ่งจะอยู่ที่ 54° H (เป็นมุมระหว่างสีเหลืองกับส้ม) ทำให้เราได้ค่า H ใหม่มาที่ (0.055,0.15) ซึ่งได้ออกมาเป็นดังนี้

ตอนนี้เราคาดว่าน่าจะดึงสีส้มออกมาได้พอสมควรแล้ว สังเกตว่าเราเห็นผลส้มเต็มใบแล้วในตอนนี้!



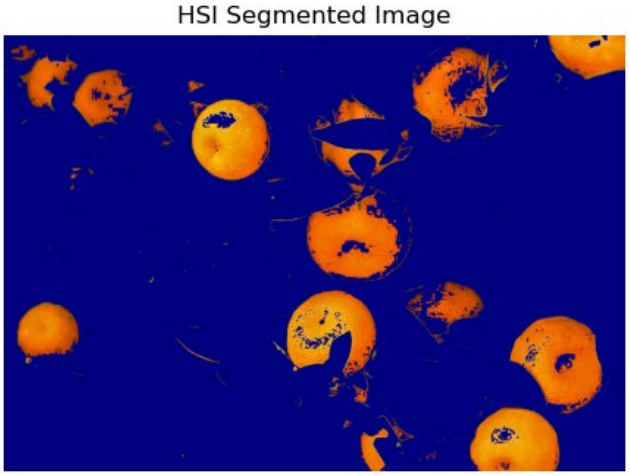


แต่ว่าเราจะเห็นว่าในภาพเกิดรอยสีดำๆขึ้น(ภาพทางขวามือ) ซึ่งเราไม่ต้องการ เราจึงทำการปรับค่า Intensity ให้ขอบเขตล่างเพิ่มขึ้นมาหน่อยจาก I = (0,1) เป็น I = (0.2,1) ในส่วนของH,S ยังคงค่าเดิมไว้ซึ่งจะได้ผลลัพธ์เป็นดังนี้



จะเห็นว่าตอนนี้รอยสีดำได้หายไปแล้ว ในส่วนของผลส้มก็ยังคงให้สีที่คงเดิมไม่เปลี่ยนแปลง (เนื่องจากเราทำการกรองในส่วนของ Intensity ซึ่งไม่เกี่ยวข้อง Hue)

ณ ตอนนี้เราได้ส่วนที่เป็นสีส้มของภาพมาแล้ว แต่ก็ยังมีรอยอยู่เล็กน้อยเราสามารถปรับขอบล่างของค่า Saturation แต่ถ้าเราปรับให้ค่าเยอะเกินไป เช่น S = (0.5,1) จะได้ผลลัพธ์เป็นดังนี้



จะเห็นว่าตอนนี้รอยที่ปรากฏในภาพได้หายไปแล้ว แต่แสงสะท้อนที่เกิดบนตัวส้มก็หายไปด้วยเช่นกัน เนื่องจากSaturation จะเป็นค่าที่บอกถึงความบริสุทธ์ของสี ทำให้บริเวณที่มีแสงตกที่สีส้มค่า Sจะน้อย(ความบริสุทธ์ต่ำ)ทำให้บริเวณนั้นหายไป

เนื่องจากโจทย์ต้องการให้เรา segment ผลส้มออกมาจากภาพดังนั้นการใช้ค่า

H = (0.055,0.15) S = (0,1) และ I = (0.2,1) จึงเป็นค่าที่คิดว่าน่าจะดีที่สุด