الجلسة الثانية

التحويل بين أنواع الصور

1.1 تحويل الصورة الملونة إلى صورة رمادية gray

يتم تحويل الصورة الملونة إلى صورة رمادية باستخدام التابع cvtColor. يأخذ هذا التابع بارامترين: البارامتر الأول يمثل الصورة الأصلية التي نريد تحويلها، أما البارامتر الثاني فهو النظام اللوني، وفي هذه الحالة نريد التحويل من صورة ملونة إلى صورة رمادية يكون البارامتر هو COLOR_BGR2GRAY ، مثال:

```
import cv2
image = cv2.imread('C:/Users/N/Desktop/Test.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

1.2 تحويل الصورة الرمادية إلى ثنائية 0،1

يتم ذلك باستخدام عتبة Threshold محددة، في المثال الآتي يتم تحويل صورة رمادية إلى ثنائية باستخدام عتبة 127 أي كل البكسلات ذات القيم أقل من 127 ستصبح 0 وكل البكسلات ذات القيم أعلى من 127 تصبح 255.

```
### gray and binary transformation
a=cv2.imread("rose.jpg",1)
gray = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)
cv2.imshow('image',gray)
cv2.waitKey()
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
cv2.imshow('Binary Image', binary)
```

1.3 التحويل من uint8 إلى double

الصورة من نوع double يتراوح مجال قيم بكسلاتها بين [0.0,1.0]. الصورة من النوع uint8 تتراوح مجال قيم بكسلاتها بين [0,255]. لتحويل صورة من النوع uint8 إلى النوع double يتم ذلك عن طريق تقسيم كل قيمة من قيم الصورة uint8 على أعلى قيمة فيها.

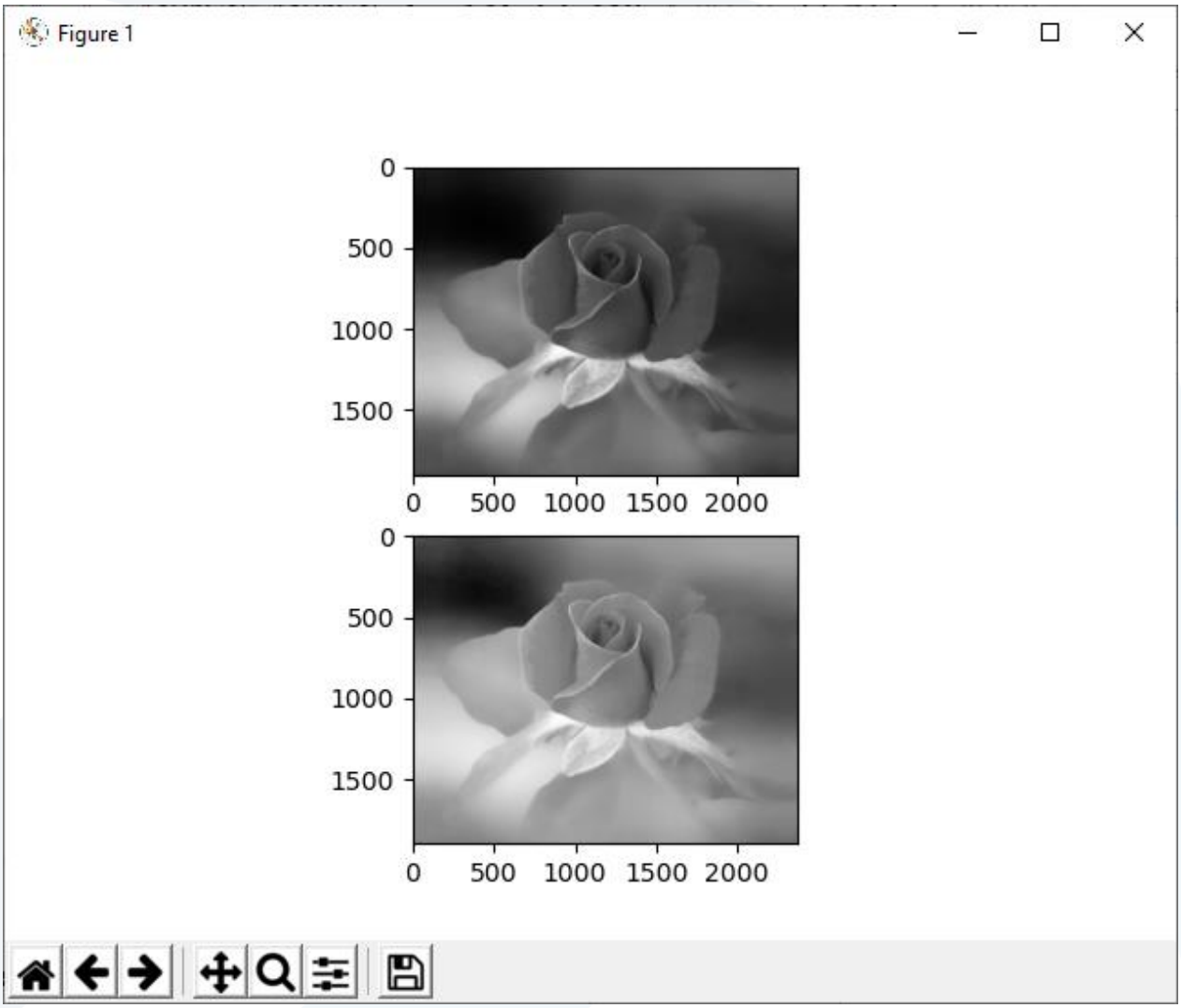
مثال برمجي: قراءة صورة كصورة ملونة ثم تحويلها للصيغة الرمادية ثم تحويلها لصيغة double ثم رفع الصورة للقيمة 0.5 للمحافظة على دقة السويات الناتجة (أما في حال تركناها بصيغة uint8 فكل السويات الناتجة عن عملية الرفع للقيمة 0.5 ستقرب لأقرب رقم صحيح).

في الكود يتم بداية اعتبار بيانات الصورة من النوع float64 ثم يتم تقسيمها على 255 والسبب حتى يتم الاحتفاظ بالدقة المضاعفة لناتج القسمة أما في حال لم نطبق عملية التحويل فإن كل قيم التقسيم سيتم تقريبها وبعد ذلك يتم تطبيق التحويل المطلوب وهو رفع جميع قيم السويات للقيمة 0.5. ثم عرض الصورة الناتجة.

```
a=cv2.imread("rose.jpg",1)
gray = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)
double_gray = gray.astype(np.float64)
double_gray=double_gray/255
double_gray = double_gray**0.5
fig, axs = plt.subplots(2, 1)
# Show the original image
axs[0].imshow(gray, cmap='gray')
axs[1].imshow(double_gray, cmap='gray')
plt.show()
```

نلاحظ خرج البرنامج هو

```
float64
0.9941002434954168
0.06262242910851495
```



وهي القيم العظمى والصغرى وكما هو ملاحظ أنها في المجال بين 0 حتى 1 وتمتلك دقة مضاعفة float64.

مخطط اللون للصورة Histogram

هو عبارة عن تابع متقطع يربط بين السويات الرمادية الموجودة في الصورة وبين عدد البكسلات في كل سوية رمادية. يفيد الهيستوغرام في دراسة تباين الصورة إذ يمكن من خلال شكل الهيستوغرام أن نحصل على معلومات جيدة عن إضاءة الصورة وذلك عند دراسة توزع البكسلات ضمن السويات الرمادية.

يتم إيجاد الهيستوغرام عن طريق التابع calcHist وفق الشكل التالي:

cv2.calcHist(*[images], [channels], mask, histSize, ranges, [hist],[accumulate]])*

images: الصورة التي نريد إيجاد الهيستوغرام لها والتي تكون من النوع uint8 أو float32. يجب أن يتم وضعها بين قوسين من الشكل [].

Channels: يمثل رقم فهرس index القناة التي نريد إنشاء هيستوغرام لها. في حالة الصورة الرمادية يكون لدينا قناة channel واحدة فقط index هو [0]. أما في حالة الصورة الرمادية يكون لدينا 3 قنوات RGB وبالتالي رقم الفهرس سيكون إما [0] للون الأحمر أو [1] للون الأخضر أو [2] للون الأزرق. يجب أن يتم وضعها بين قوسين من الشكل [].

Mask: إذا أردنا إيجاد الهيستوغرام لكامل الصورة نضع None، أما إذا أدرنا إيجاد الهيستوغرام لجزء محدد من الصورة (منطقة محددة) يجب أن نقوم بإنشاء قناع لهذه المنطقة.

histSize: عدد البتات اللازمة لتمثيل الصورة. في حال كانت الصورة من النوع uint8 في هذه الحالة يكون histsize 256.

Range: عادة يكون [0-255].

<u>مثال</u>

```
import cv2 as cv
import matplotlib.pyplot as plt
img = cv.imread('home.jpg',0)
hist = cv.calcHist([img],[0],None,[256],[0,255])
plt.plot(hist)
```

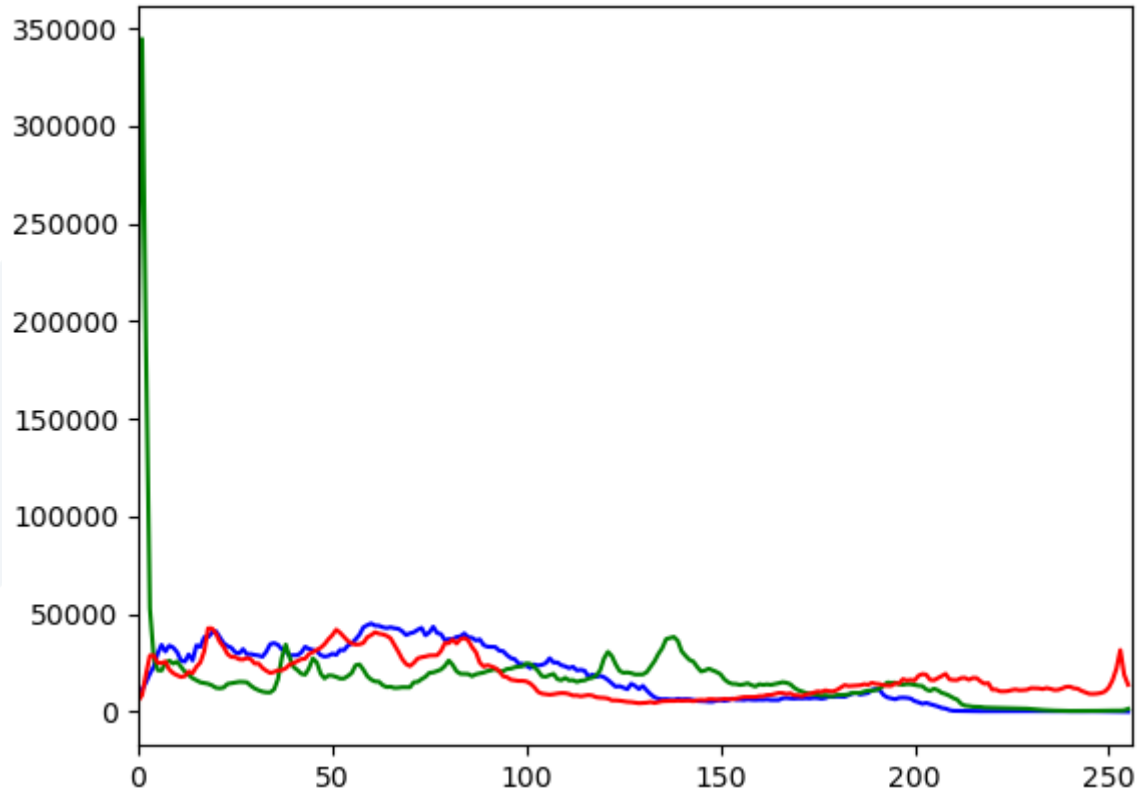
لعرض الهيستوغرام لجزء محدد من الصورة:

```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
Mask=np.zeros(img.shape[:2],np.unit8)
Mask[100:300,100:200]=255
hist = cv.calcHist([img],[0], Mask,[256],[0,255])
```

لعرض الهيستوغرام الخاص بالصورة الملونة:

```
import cv2
import matplotlib.pypplot as plt
img = cv2.imread('home.jpg',0)
```

```
color = ('b','g','r')
for i,col in enumerate(color):
  histr = cv2.calcHist([img],[i],None,[256],[0,256])
  plt.plot(histr,color = col)
  plt.xlim([0,256])
plt.show()
```



لتسوية الهيستوغرام نستخدم التابع cv2.equalizeHist(img) حيث يتم توزيع السويات الرمادية على كامل مجال السويات اللونية من 0 وحتى 255. لكن مساواة الهيستوغرام لا تنفع مع جميع الصور لذلك نلجأ لطرق أخرى مثل تسوية الهيستوغرام التكيفي adaptive histogram equalization.

لتسوية الهيستوغرام التكيفي نستخدم التابع:

```
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
```

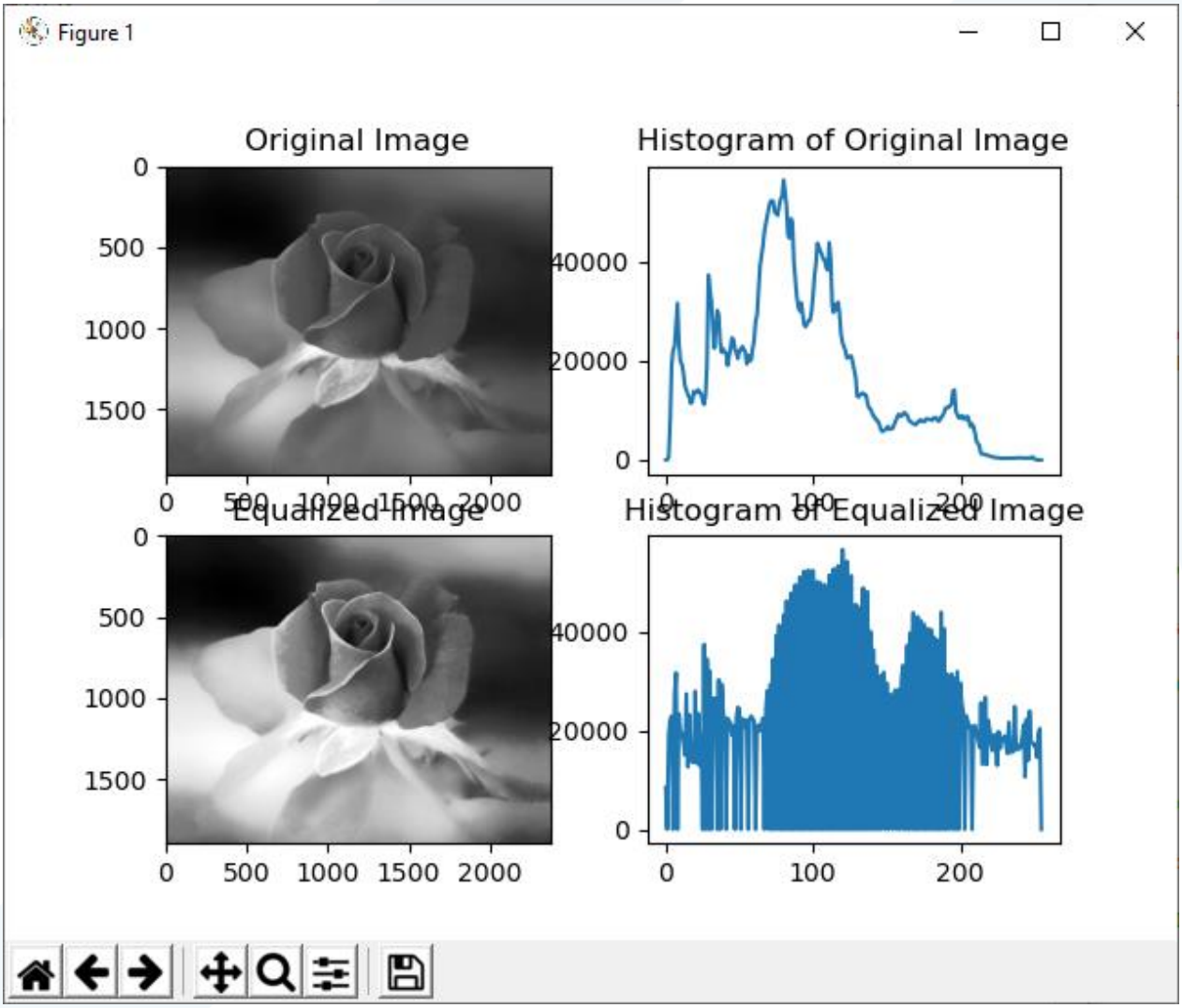
حيث clipLimit هو عتبة التسوية التكيفية فكلما كانت أعلى كلما كانت عملية التسوية أشد وكلما كانت أقل تكون عملية التسوية أقل.

tileGridSize هو حجم نافذة التسوية حيث يتم تقسيم الصورة لأجزاء ليتم ضمنها تنفيذ عملية التسوية بدلاً من تسوية كامل الصورة معاً.

لتطبيق تابع CLAHE نكتب:

```
equ = clahe.apply(img)
```

مثال: حساب هيستوغرام صورة ثم تسوية الهيستوغرام وعرض النتيجة قبل وبعد التسوية (ينفذ المثال في الجلسة العملية)

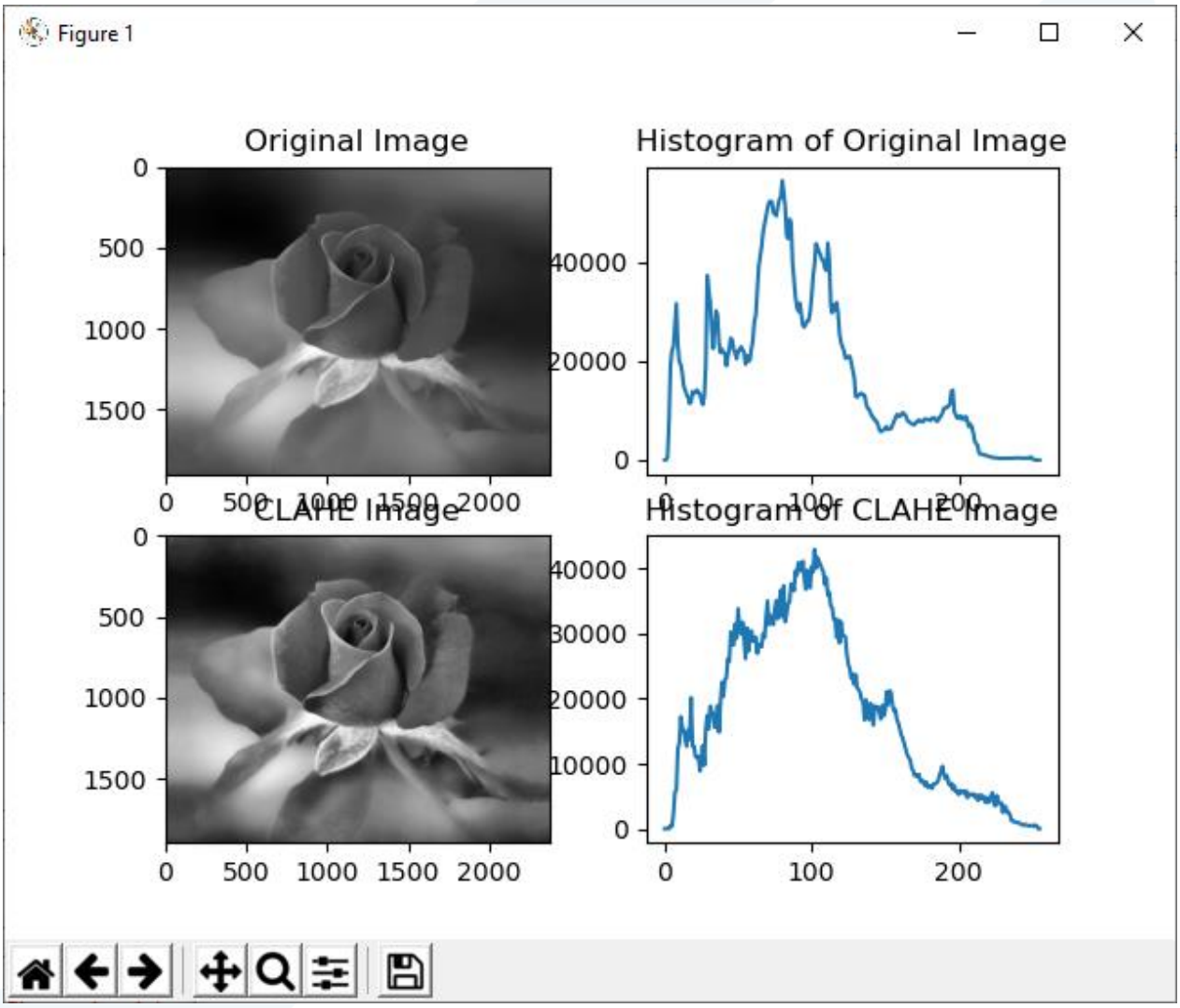


مثال: حساب هيستوغرام صورة ثم تسوية الهيستوغرام التكيفية وعرض النتيجة قبل وبعد مساواة الهيستوغرام التكيفية CLAHE

```
import numpy as np
```

```
import cv2
from matplotlib import pyplot as plt
# Load the image
img = cv2.imread('rose.jpg',0)
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
# Apply CLAHE to the image
equ = clahe.apply(img)
# Calculate the histogram of the original and CLAHE image
hist = cv2.calcHist([img],[0],None,[256],[0,255])
hist_equ = cv2.calcHist([equ],[0],None,[256],[0,255])

# Create subplots
fig, axs = plt.subplots(2, 2)

# Show the original image
axs[0, 0].imshow(img, cmap='gray')
axs[0, 0].set_title('Original Image')

# Show the histogram of the original image
axs[0, 1].plot(hist)
axs[0, 1].set_title('Histogram of Original Image')

# Show the CLAHE image
axs[1, 0].imshow(equ, cmap='gray')
axs[1, 0].set_title('CLAHE Image')

# Show the histogram of the CLAHE image
axs[1, 1].plot(hist_equ)
```

```
axs[1, 1].set_title('Histogram of CLAHE Image')

# Display the plot
plt.show()
```



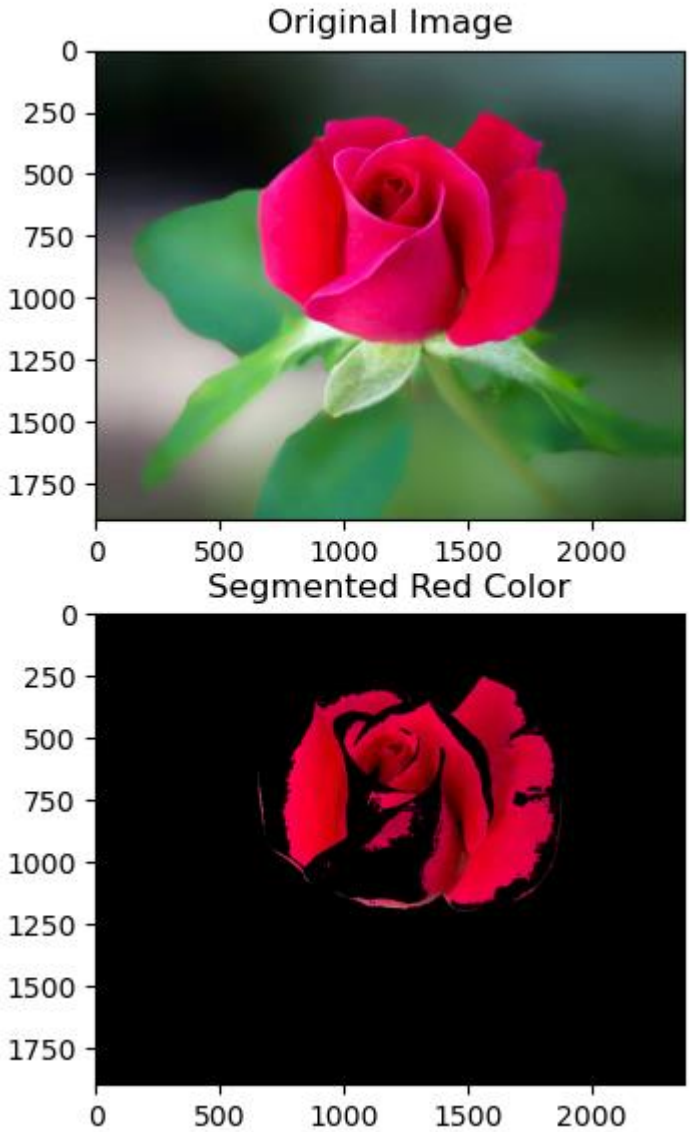
تحويلات أنظمة الألوان Color systems transformation

للتحويل من نظام الألوان BGR إلى نظام الألوان HSV وذلك لفصل مركبات الإضاءة عن اللون حيث تمثل H مركبة اللون Hue والمركبة S مركبة الإشباع S والمركبة V مركبة الإضاءة (السويات الرمادية).

يمكن استخدام مركبة الإضاءة لوحدها ومركبات الألوان لوحدها.

التجزيء اللوني بالاعتماد على فضاء الألوان HSV

سنعتمد على المركبات اللونية فقط لكشف اللون الأحمر في الصورة.



Code:

```
import cv2

import numpy as np
```

```
from matplotlib import pyplot as plt

# Load the image

img = cv2.imread('rose.jpg',1)
```

قراءة الصورة بصيغتها الملونة

```
# Convert the image from BGR to HSV

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

التحويل لنظام HSV

```
# Define range for red color in HSV

lower_red = np.array([0, 120, 70])

upper_red = np.array([10, 255, 255])
```

بناء قناع التجزيء الأول المعتمد على الجزء الأول من مجالات اللون الأحمر في نظام HSV
يمتلك اللون الأحمر مجالين في نظام HSV الأول يكون مجال H فيه بين 0-10 والثاني بين 170-180
أما الإشباع اللوني والإضاءة فهي ثابتة في القناعين

```
# Threshold the HSV image to get only red colors

mask1 = cv2.inRange(hsv, lower_red, upper_red)
```

بناء قناع التجزيء الأول mask1 بالاعتماد على المجالات اللونية السابقة

```
# Define range for red color in HSV

lower_red = np.array([170, 120, 70])

upper_red = np.array([180, 255, 255])
```

بناء قناع التجزيء الثاني المعتمد على الجزء الثاني من مجالات اللون الأحمر في نظام HSV
يمتلك اللون الأحمر مجالين في نظام HSV الأول يكون مجال H فيه بين 0-10 والثاني بين 170-180
أما الإشباع اللوني والإضاءة فهي ثابتة في القناعين

```
# Threshold the HSV image to get only red colors

mask2 = cv2.inRange(hsv, lower_red, upper_red)
```

```
# Combine the two masks

mask = mask1 + mask2
```

جمع القناعين معاً لتشكيل قناع اللون الأحمر الموحد

```
# Bitwise-AND mask and original image

res = cv2.bitwise_and(img, img, mask=mask)
```

تطبيق القناع على الصورة بطريقة bitwise_and او الضرب المنطقي في نظام unit8 حيث تتطبق عملية AND على مستويات الخانات الثمانية 8bit

```
# Convert the BGR images to RGB

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

res_rgb = cv2.cvtColor(res, cv2.COLOR_BGR2RGB)

# Create subplots

fig, axs = plt.subplots(2, 1)
```

تحويل الصور من نظام BGR الذي تعمل به cv2 وتابع imshow إلى نظام RGB الذي تعمل به Matplotlib.pyplot

```
# Show the original image

axs[0].imshow(img_rgb)

axs[0].set_title('Original Image')
```

عرض الصور بطريقة Subplot

```
# Show the result image

axs[1].imshow(res_rgb)

axs[1].set_title('Segmented Red Color')

# Display the plot

plt.show()
```