OpenCV

04 色彩空間影像處理

Ex:04-01

```
import numpy as np
import cv2

def RGB_model( f, channel ):
    if channel == 1:     # Red
        return f[:,:,2]
    elif channel == 2:  # Green
        return f[:,:,1]
    else:                    # Blue
        return f[:,:,0]

def main( ):
    img = cv2.imread( "Rose.bmp", -1 )
    R = RGB_model( img, 1 )
    G = RGB_model( img, 2 )
    B = RGB_model( img, 3 )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Red", R )
    cv2.imshow( "Green", G )
    cv2.imshow( "Blue", B )
    cv2.waitKey()
    cv2.destroyAllWindows()

main( )
```

Ex:04-02

```
import numpy as np
import cv2

def CMY_model( f, channel ):
    if channel == 1:     # Cyan
        return 255 - f[:,:,2]
    elif channel == 2:  # Magenta
        return 255 - f[:,:,1]
    else:                    # Yellow
        return 255 - f[:,:,0]
```

```
def main( ):
    img = cv2.imread( "Rose.bmp", -1 )
    C = CMY_model( img, 1 )
    M = CMY_model( img, 2 )
    Y = CMY_model( img, 3 )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Cyan", C )
    cv2.imshow( "Magenta", M )
    cv2.imshow( "Yellow", Y )
    cv2.waitKey()
    cv2.destroyAllWindows()

main( )
```

Ex:04-03

```
import numpy as np
import cv2

def RGB_to_HSI( R, G, B ):
    r = R / 255
    g = G / 255
    b = B / 255
    if R == G and G == B:
        H = -1.0
        S =    0.0
        I = ( r + g + b ) / 3
    else:
        x = ( 0.5 * ( ( r - g ) + ( r - b ) ) ) / \
              np.sqrt( ( r - g ) ** 2 + ( r - b ) * ( g - b ) )
        if x < -1.0:    x = -1.0
        if x >    1.0:    x =    1.0
        theta = np.arccos( x ) * 180 / np.pi
        if B <= G:
            H = theta
        else:
            H = 360.0 - theta
        S = 1.0 - 3.0 / ( r + g + b ) * min( r, g, b )
```

```
        I = ( r + g + b ) / 3
    return H, S, I

def HSI_model( f, channel ):
    nr, nc = f.shape[:2]
    g = np.zeros( [nr, nc], dtype = 'uint8' )
    if channel == 1:            # Hue
        for x in range( nr ):
            for y in range( nc ):
                H, S, I = RGB_to_HSI( f[x,y,2], f[x,y,1], f[x,y,0] )
                if H == -1:
                    k = 0
                else:
                    k = round( H * 255 / 360 )
                g[x,y] = np.uint8( k )
    elif channel == 2:          # Saturation
        for x in range( nr ):
            for y in range( nc ):
                H, S, I = RGB_to_HSI( f[x,y,2], f[x,y,1], f[x,y,0] )
                k = round( S * 255 )
                g[x,y] = np.uint8( k )
    else:                       # Intensity
        for x in range( nr ):
            for y in range( nc ):
                H, S, I = RGB_to_HSI( f[x,y,2], f[x,y,1], f[x,y,0] )
                k = round( I * 255 )
                g[x,y] = np.uint8( k )
    return g

def main( ):
    img = cv2.imread( "Rose.bmp", -1 )
    H = HSI_model( img, 1 )
    S = HSI_model( img, 2 )
    I = HSI_model( img, 3 )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Hue", H )
    cv2.imshow( "Saturation", S )
    cv2.imshow( "Intensity", I )
    cv2.waitKey()
```

```
    cv2.destroyAllWindows()
main( )
```

Ex:04-04

```
import numpy as np
import cv2

def HSV_model( f, channel ):
    hsv = cv2.cvtColor( f, cv2.COLOR_BGR2HSV )
    if channel == 1:     # Hue
        return hsv[:,:,0]
    elif channel == 2:  # Saturation
        return hsv[:,:,1]
    else:                    # Value
        return hsv[:,:,2]

def main( ):
    img = cv2.imread( "Rose.bmp", -1 )
    H = HSV_model( img, 1 )
    S = HSV_model( img, 2 )
    V = HSV_model( img, 3 )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Hue", H )
    cv2.imshow( "Saturation", S )
    cv2.imshow( "Value", V )
    cv2.waitKey()
    cv2.destroyAllWindows()

main( )
```

Ex:04-05

```
import numpy as np
import cv2

def YCrCb_model( f, channel ):
    ycrcb = cv2.cvtColor( f, cv2.COLOR_BGR2YCrCb )
    if channel == 1:     # Y
        return ycrcb[:,:,0]
```

```
    elif channel == 2:  # Cr
        return ycrcb[:,:,1]
    else:               # Cb
        return ycrcb[:,:,2]

def main( ):
    img = cv2.imread( "Rose.bmp", -1 )
    Y   = YCrCb_model( img, 1 )
    Cr = YCrCb_model( img, 2 )
    Cb = YCrCb_model( img, 3 )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Y", Y )
    cv2.imshow( "Cr", Cr )
    cv2.imshow( "Cb", Cb )
    cv2.waitKey()
    cv2.destroyAllWindows()

main( )
```

Ex:04-06

```
import numpy as np
import cv2

print( "Pseudocolor Image Processing:")
print( "(0) Autumn" )
print( "(1) Bone" )
print( "(2) Jet" )
print( "(3) Winter" )
print( "(4) Rainbow" )
print( "(5) Ocean" )
print( "(6) Summer" )
print( "(7) Spring" )
print( "(8) Cool" )
print( "(9) HSV" )
print( "(10) Pink" )
print( "(11) Hot" )
print( "(12) Parula" )
print( "(13) Magma" )
```

```
print( "(14) Inferno" )
print( "(15) Plasma" )
print( "(16) Viridis" )
print( "(17) Cividis" )
print( "(18) Twilight" )
print( "(19) Twilight Shifted" )
colormap = eval( input( "Enter your choice: " ) )
img1 = cv2.imread( "Gray_Level.bmp", -1 )
img2 = cv2.applyColorMap( img1, colormap )
cv2.imshow( "Original Image", img1 )
cv2.imshow( "Pseudocolor Image", img2 )
cv2.waitKey()
cv2.destroyAllWindows()
```

Ex:04-07

```
import cv2
import matplotlib.pyplot as plt

o = cv2.imread("boat.bmp")
cv2.imshow("original", o)

plt.hist(o.ravel(), 256)
plt.show()

cv2.waitKey()
cv2.destroyAllWindows()
```

Ex:04-08

```
import cv2
import matplotlib.pyplot as plt

o = cv2.imread("boat.bmp")
cv2.imshow("original", o)

plt.hist(o.ravel(), 16, [0, 255])
plt.show()
```

```
cv2.waitKey()
cv2.destroyAllWindows()
```

Ex:04-09

```
import cv2

img = cv2.imread("boat.bmp")
cv2.imshow("original", img)
hist = cv2.calcHist([img], [0], None, [256], [0, 255])

print(type(hist))
print(hist.shape)
print(hist.size)
print(hist)
```

Ex:04-10

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6]
y = [0.3, 0.4, 2, 5, 3, 4.5, 4]

plt.plot(x, y)
plt.show()
```

Ex:04-11

```
import matplotlib.pyplot as plt

y = [0.3, 0.4, 2, 5, 3, 4.5, 4]

plt.plot(y)
plt.show()
```

Ex:04-12

```
import matplotlib.pyplot as plt
```

```
a = [0.3, 0.4, 2, 5, 3, 4.5, 4]
b = [3, 5, 1, 2, 1, 5, 3]

plt.plot(a, color='r')
plt.plot(b, color='g')

plt.show()
```

Ex:04-13 plot + cv2.calcHist()

```
import cv2
import matplotlib.pyplot as plt

o = cv2.imread("boat.bmp")
histb = cv2.calcHist([o], [0], None, [256], [0, 255])

plt.plot(histb, color='b')
plt.show()
```

Ex:04-14 plot + cv2.calcHist() + bgr

```
import cv2
import matplotlib.pyplot as plt

o = cv2.imread("lenacolor.png")

histb = cv2.calcHist([o], [0], None, [256], [0, 255])
histg = cv2.calcHist([o], [1], None, [256], [0, 255])
histr = cv2.calcHist([o], [2], None, [256], [0, 255])

plt.plot(histb, color='b')
plt.plot(histg, color='g')
plt.plot(histr, color='r')
plt.show()
```

Ex:04-15 mask + cv2.calcHist()

```
import cv2
import numpy as np
```

```
import matplotlib.pyplot as plt

image = cv2.imread("lenacolor.png", cv2.IMREAD_GRAYSCALE)
cv2.imshow('image', image)

mask = np.zeros(image.shape, np.uint8)
mask[200:400, 200:400] = 255
cv2.imshow('mask', mask)

histImage = cv2.calcHist([image], [0], None, [256], [0, 255])
histMI = cv2.calcHist([image], [0], mask, [256], [0, 255])

plt.plot(histImage)
plt.plot(histMI)
plt.show()
```

Ex:04-16 cv2.equalizeHist()

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("lenacolor.png", cv2.IMREAD_GRAYSCALE)
cv2.imshow('original', img)

equ = cv2.equalizeHist(img)
cv2.imshow('result', equ)

plt.figure("原始影像長條圖")
plt.hist(img.ravel(), 256)

plt.figure("均質化後長條圖")
plt.hist(equ.ravel(), 256)
plt.show()

cv2.waitKey()
cv2.destroyAllWindows()
```

Ex:04-17

```
import numpy as np
import cv2

def RGB_histogram_equalization( f ):
    g = f.copy( )
    for k in range( 3 ):
        g[:,:,k] = cv2.equalizeHist( f[:,:,k] )
    return g

def main( ):
    img1 = cv2.imread( "Rose.bmp", -1 )
    img2 = RGB_histogram_equalization( img1 )
    cv2.imshow( "Original Image", img1 )
    cv2.imshow( "Histogram Equalization(RGB)", img2 )
    cv2.waitKey( 0 )

main()
```

Ex:04-18

```
import numpy as np
import cv2

def HSV_histogram_equalization( f ):
    hsv = cv2.cvtColor( f, cv2.COLOR_BGR2HSV )
    hsv[:,:,2] = cv2.equalizeHist( hsv[:,:,2] )
    g = cv2.cvtColor( hsv, cv2.COLOR_HSV2BGR )
    return g

def main( ):
    img1 = cv2.imread( "Rose.bmp", -1 )
    img2 = HSV_histogram_equalization( img1 )
    cv2.imshow( "Original Image", img1 )
    cv2.imshow( "Histogram Equalization(HSV)", img2 )
    cv2.waitKey( 0 )

main( )
```

Ex:04-19 色彩校正(增強)　Gamma Correction

```
import numpy as np
import cv2

def RGB_gamma_correction( f, channel, gamma ):
    g = f.copy( )
    nr, nc = f.shape[:2]
    c = 255.0 / ( 255.0 ** gamma )
    table = np.zeros( 256 )
    for i in range( 256 ):
        table[i] = round( i ** gamma * c, 0 )
    if channel == 1:        k = 2
    elif channel == 2:    k = 1
    else:                        k = 0
    for x in range( nr ):
        for y in range( nc ):
            g[x,y,k] = table[f[x,y,k]]
    return g

def main( ):
    img    = cv2.imread( "Rose.bmp", -1 )
    gamma = eval( input( "Please enter gamma: " ) )
    img1 = RGB_gamma_correction( img, 1, gamma )
    img2 = RGB_gamma_correction( img, 2, gamma )
    img3 = RGB_gamma_correction( img, 3, gamma )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Gamma Correction(R)", img1 )
    cv2.imshow( "Gamma Correction(G)", img2 )
    cv2.imshow( "Gamma Correction(B)", img3 )
    cv2.waitKey()

main()
```

Ex:04-20 影像濾波

```
import numpy as np
import cv2

img1 = cv2.imread( "Baboon.bmp", -1 )
```

```
img2 = cv2.GaussianBlur( img1, ( 5, 5 ), 0 )
cv2.imshow( "Original Image", img1 )
cv2.imshow( "Gaussian Filtering", img2 )
cv2.waitKey()
```

Ex:04-21 HIS + 調整各參數 H S I

```
import numpy as np
import cv2

def RGB_to_HSI( R, G, B ):
    r = R / 255
    g = G / 255
    b = B / 255
    if R == G and G == B:
        H = -1.0
        S =   0.0
        I = ( r + g + b ) / 3
    else:
        x = ( 0.5 * ( ( r - g ) + ( r - b ) ) ) / \
            np.sqrt( ( r - g ) ** 2 + ( r - b ) * ( g - b ) )
        if x < -1.0:   x = -1.0
        if x >   1.0:   x =   1.0
        theta = np.arccos( x ) * 180 / np.pi
        if B <= G:
            H = theta
        else:
            H = 360.0 - theta
        S = 1.0 - 3.0 / ( r + g + b ) * min( r, g, b )
        I = ( r + g + b ) / 3
    return H, S, I

def HSI_to_RGB( H, S, I ):
    if H == -1.0:
        r = I
        g = I
        b = I
    elif H >= 0 and H < 120:
        HH = H
```

```
        b = I * ( 1 - S )
        r = I * ( 1 + ( S * np.cos( HH * np.pi / 180 ) ) /
            np.cos( ( 60 - HH ) * np.pi / 180 ) )
        g = 3.0 * I - ( r + b )
    elif H >= 120 and H < 240:
        HH = H - 120.0
        r = I * ( 1 - S )
        g = I * ( 1 + ( S * np.cos( HH * np.pi / 180 ) ) /
            np.cos( ( 60 - HH ) * np.pi / 180 ) )
        b = 3 * I - ( r + g )
    else:
        HH = H - 240
        g = I * ( 1 - S )
        b = I * ( 1 + ( S * np.cos( HH * np.pi / 180 ) ) /
            np.cos( ( 60 - HH ) * np.pi / 180 ) )
        r = 3 * I - ( g + b )
    rr = round( r * 255 )
    gg = round( g * 255 )
    bb = round( b * 255 )
    R = np.uint8( np.clip( rr, 0, 255 ) )
    G = np.uint8( np.clip( gg, 0, 255 ) )
    B = np.uint8( np.clip( bb, 0, 255 ) )
    return R, G, B

def HSI_processing( f, angle = 0, saturation = 100, intensity = 100 ):
    g = f.copy( )
    nr, nc = f.shape[:2]

    for x in range( nr ):
        for y in range( nc ):
            H, S, I = RGB_to_HSI( f[x,y,2], f[x,y,1], f[x,y,0] )
            H = H + angle
            if H > 360:    H = H - 360
            S = S * saturation / 100
            I = I * intensity / 100
            R, G, B = HSI_to_RGB( H, S, I )
            g[x,y,0] = B
            g[x,y,1] = G
            g[x,y,2] = R
```

```
    return g

def main():
    img = cv2.imread( "Rainbow_Village.bmp", -1 )
    img1 = HSI_processing( img, 180, 100, 100 )
    img2 = HSI_processing( img, 0, 50, 100 )
    img3 = HSI_processing( img, 0, 100, 50 )
    cv2.imshow( "Original Image", img )
    cv2.imshow( "Hue(Rotate 180 degrees)", img1 )
    cv2.imshow( "Saturation by 50%", img2 )
    cv2.imshow( "Intensity by 50%", img3 )
    cv2.waitKey()

main()
```

Ex:04-22 HSV + 色彩分割

```
import numpy as np
import cv2

def HSV_color_segmentation( f, H1, H2, S1, S2, V1, V2 ):
    g = f.copy( )
    nr, nc = f.shape[:2]
    hsv = cv2.cvtColor( f, cv2.COLOR_BGR2HSV )
    for x in range( nr ):
        for y in range( nc ):
            H = hsv[x,y,0] * 2
            S = hsv[x,y,1] / 255 * 100
            V = hsv[x,y,2] / 255 * 100
            if not ( H >= H1 and H <= H2 and S >= S1 and S <= S2
                        and V >= V1 and V <= V2 ):
                g[x,y,0] = g[x,y,1] = g[x,y,2] = 0
    return g

def main( ):
    img1 = cv2.imread( "Flower.bmp", -1 )
    img2 = HSV_color_segmentation( img1, 30, 70, 30, 100, 30, 100 )
    cv2.imshow( "Original Image", img1 )
    cv2.imshow( "HSV Color Segmentation", img2 )
```

```
    cv2.waitKey( 0 )

main( )
```

Ex:04-23 cv2.inRange()

```
import cv2
import numpy as np

img = np.random.randint(0, 256, size=[5, 5], dtype=np.uint8)
min_ = 100
max_ = 200
mask = cv2.inRange(img, min_, max_)
print("img=\n", img)
print("mask=\n", mask)
```

Ex:04-24 cv2.inRange() ROI

```
import cv2
import numpy as np

img = np.ones([5, 5], dtype=np.uint8)* 9
mask = np.zeros([5, 5], dtype=np.uint8)
mask[0:3, 0] = 1
mask[2:5, 2:4] = 1
roi = cv2.bitwise_and(img, img, mask=mask)
print("img=\n", img)
print("mask=\n", mask)
print("roi=\n", roi)
```

Ex:04-25 opencv Logo

```
import cv2
import numpy as np

opencv = cv2.imread("opencv.jpg")
hsv = cv2.cvtColor(opencv, cv2.COLOR_BGR2HSV)
cv2.imshow("opencv", opencv)
```

```
minBlue = np.array([110, 50, 50])
maxBlue = np.array([130, 255, 255])

mask = cv2.inRange(hsv, minBlue, maxBlue)
blue = cv2.bitwise_and(opencv, opencv, mask = mask)
cv2.imshow("blue", blue)
```

….請自行新增其他顏色

Ex:04-26　標記人臉膚色

```
import cv2

img = cv2.imread("face.jpg")
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv)

minHue = 5
maxHue = 170
hueMask = cv2.inRange(h, minHue, maxHue)

minSat = 25
maxSat = 166
satMask = cv2.inRange(s, minSat, maxSat)
mask = hueMask & satMask

roi = cv2.bitwise_and(img, img, mask = mask)
cv2.imshow("img", img)
cv2.imshow("ROI", roi)
cv2.waitKey()
cv2.destroyAllWindows()
```