**Opencv**

pip install opencv-python

**Color theory**

**รูปแบบสี**

RGB, RYB, CMYK, HSV and HSL

**ชนิดของภาพ**

A warning sign with binary code

Description automatically generatedBinary image

Grayscale image

ขาว-ดำ-เทา แต่ละ pixelจะมีเพียง 8 bit ( 0-255 )โดยการแปลงภาพสี RGB มาเป็นภาพ Grayscale ใช้สูตรทางคณิตศาสตร์ดังนี้



RGB image

 ภาพสีทั่วๆไป โดยแต่ละ pixel จะมี 24 bit ( R, G, B )

Read image

# *read -> rgb to gray -> export image*

import cv2

paht = './image'

filename = 'cat'

img = cv2.imread(f'{paht}/{filename}.jpg') # *array 3 dimention*

#*resize*

resize\_img = cv2.resize(img, (400, 400))

#*gray scale*

gray\_img = cv2.cvtColor(resize\_img, cv2.COLOR\_RGB2GRAY)

#*show image*

cv2.imshow('Output', gray\_img)

cv2.waitKey(0)

#*export image imwrite(paht, image)*

cv2.imwrite(f'{paht}/{filename}\_cv.jpg', gray\_img)

cv2.destroyAllWindows()

A cat wearing a hat

Description automatically generated

Read video

import cv2

paht = './video'

video\_name = 'person\_road'

cap = cv2.VideoCapture(f'{paht}/{video\_name}.mp4')

while cap.isOpened():

    # *status(boolean), piture*

    chk, frame = cap.read() #*read piture frame/frame*

    if chk:

        #*resize*

        resize\_video = cv2.resize(frame, (640, 480))

        gray\_video = cv2.cvtColor(resize\_video, cv2.COLOR\_BGR2GRAY)

        cv2.imshow('Output', gray\_video)

        if cv2.waitKey(1) & 0xFF == ord('e'):

            break

    else:

        break

cap.release()

cv2.destroyAllWindows()

Draw line

import cv2

paht = './image'

filename = 'cat'

width = 640

height = 480

size = (width, height)

img = cv2.imread(f'{paht}/{filename}.jpg')

#*resize*

size\_img = cv2.resize(img, size)

#*draw line => line( image, start(x, y), end(x, y), color(bgr), weight)*

cv2.line(size\_img, (40,400), (600,400), (33, 33, 247), 3) # *red line*

cv2.arrowedLine(size\_img, (40,350), (600,350), (255, 51, 51), 5) # *blue line with arrow*

#*show*

cv2.imshow('Output', size\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

A cat with its tongue out

Description automatically generated

Draw rectangle

import cv2

img = cv2.imread('./image/cat.jpg')

#*resize*

size\_img = cv2.resize(img, (600, 480))

#*draw rectangle => cv2.rectangle(image, start, end, color, weight)*

cv2.rectangle(size\_img, (300, 100), (500, 200), (204, 0, 102), 5) # *purple*

cv2.rectangle(size\_img, (330, 130), (470, 170), (127, 0, 255), -1) # *pink*

cv2.imshow('Output', size\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

A cat with its tongue out

Description automatically generated

Draw circle

import cv2

img = cv2.imread('./image/cat.jpg')

#*resize*

size\_img = cv2.resize(img, (600, 480))

#*draw circle => cv2.circle(image, center, redius, color, weight)*

cv2.circle(size\_img, (300, 240), 100, (0,0,255), 5)

cv2.imshow('Output', size\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

A cat with a hat on

Description automatically generated

Text in image

import cv2

img = cv2.imread('./image/cat.jpg')

#*resize*

size\_img = cv2.resize(img, (600, 480))

font = cv2.FONT\_HERSHEY\_SIMPLEX

#*put text=> cv2.putText(image, text, point, font, font size, color, weight)*

cv2.putText(size\_img, 'Hello World', (15,30), font, 1, (255, 0, 0), cv2.LINE\_4)

#*show*

cv2.imshow('Output', size\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

A cat with its tongue out

Description automatically generated

Mouse event

A road with trees and numbers on it

Description automatically generatedimport cv2

img = cv2.imread("./image/natural.jpg")

name = "output"

cv2.namedWindow(name)

# *mouse event*

size = (600, 480)

size\_img = cv2.resize(img, size)

def clickPosition(event, x, y, flags, param):

    text = '' # *put text onclick*

    color = () # *color text*

    #*left click*

    if event == cv2.EVENT\_LBUTTONDOWN:

        text = f'{x},{y}'

        color = (0, 0, 255)

    # *right click*

    elif event == cv2.EVENT\_RBUTTONDOWN:

        blue = int(size\_img[y,x,0])

        green = int(size\_img[y,x,1])

        red = int(size\_img[y,x,2])

        text = f'{blue},{green},{red}'

        color = (blue ,green, red)

    #*posiotion text (x, y)*

    if x > 460:

        x -= 100

    if y < 30:

        y += 20

    text\_location = (x,y)

    cv2.putText(size\_img, text, text\_location, cv2.FONT\_HERSHEY\_SIMPLEX, .8, color, cv2.LINE\_4)

    cv2.imshow(name, size\_img)

cv2.setMouseCallback(name, clickPosition)

cv2.imshow(name, size\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

Detect object (Detect color)

import cv2

import numpy

img = cv2.imread('./image/ball\_color.jpg')

img = cv2.resize(img, (400, 400))

while True:

    # *color range (b,g,r)*

    lower = numpy.array([7, 104, 0])

    upper = numpy.array([119, 253, 110])

    # *mask => binary*

    mask = cv2.inRange(img, lower, upper)

    result = cv2.bitwise\_and(img, img, mask=mask)

    cv2.imshow('Original', img)

    cv2.imshow('Mask', mask)

    cv2.imshow('Result', result)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cv2.destroyAllWindows()

A screenshot of a computer

Description automatically generated

Thresholding

Fundamental

* Thresholding จะทำงานร่วมกับภาพแบบ **gray scale และ Banary**

RGB

Gray

Binary

import cv2

import matplotlib.pyplot as plt

img = cv2.imread('./image/threshold.jpg')

img = cv2.resize(img, (500,500))

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

thres\_bin, result\_bin = cv2.threshold(gray, 128, 255, cv2.THRESH\_BINARY)

thres\_bin\_in, result\_bin\_in = cv2.threshold(gray, 128, 255, cv2.THRESH\_BINARY\_INV)

thres\_trunc, result\_trunc = cv2.threshold(gray, 128, 255, cv2.THRESH\_TRUNC)

thres\_2zero, result\_2zero = cv2.threshold(gray, 128, 255, cv2.THRESH\_TOZERO)

thres\_2zero\_in, result\_2zero\_in = cv2.threshold(gray, 128, 255, cv2.THRESH\_TOZERO\_INV)

r1 = cv2.cvtColor(result\_bin, cv2.COLOR\_BGR2RGB)

r2 = cv2.cvtColor(result\_bin\_in, cv2.COLOR\_BGR2RGB)

r3 = cv2.cvtColor(result\_trunc, cv2.COLOR\_BGR2RGB)

r4 = cv2.cvtColor(result\_2zero, cv2.COLOR\_BGR2RGB)

r5 = cv2.cvtColor(result\_2zero\_in, cv2.COLOR\_BGR2RGB)

# *cv2.imshow('gray', gray)*

# *cv2.imshow('result', result\_bin)*

images = [r1, r2, r3, r4, r4, r5]

titles = ["GRAY", "BINARY", "BINARY-INVERT", "TRUNC", "TOZERO", "TOZERO-INVERT"]

for i in range(len(images)):

    plt.subplot(2, 3, i+1)

    plt.imshow(images[i])

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.show()

A screenshot of a computer

Description automatically generated

Adaptive thresholding

A collage of a grid of sudoku

Description automatically generated

cv2.adaptiveThreshold(1,2,3,4,5,6)

1. A screenshot of a computer

   Description automatically generatedภาพ grayscale
2. maxValue
3. adaptive method
4. threshold type
5. box size
6. C

'''

cv2.adaptiveThreshold(1,2,3,4,5,6)

1. ภาพ grayscale

2. maxValue

3. adaptive method

4. threshold type

5. box size # ยิ่งเยอะความคมชัดของภาพยิ่งมากขึ้น

6. C

'''

import cv2

img = cv2.imread('./image/earth.jpg')

img = cv2.resize(img, (400,400))

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

#*normal threshold*

thres\_bin, result\_bin = cv2.threshold(gray, 128, 255, cv2.THRESH\_BINARY)

# *adaptive mean*

result\_mean = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 5, 1)

# *adaptive gaussian*

result\_gauss = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 5, 1)

cv2.imshow('output',gray)

cv2.imshow('THRES-NORMAL',result\_bin)

cv2.imshow('ADAP\_MEAN', result\_mean)

cv2.imshow('ADAP\_GAUSS', result\_gauss)

cv2.waitKey(0)

cv2.destroyAllWindows()

A screenshot of a computer screen

Description automatically generated

**Morphological**

เป็นเทคนิคสำหรับการวิเคราะห์และประมวลผลภาพ โดยอาศัยโครงสร้างทางเรขาคณิตบนพื้นฐานของทฤษฎีเซต

เทคนิคนี้นิยมนำมาใช้งานกับภาพดิจิตอลเพื่อ**วิเคราะห์ พื้นผิว ขนาด รูปร่าง เป็นต้น**

import cv2

import numpy as np

img = cv2.imread('./image/coin\_noise.jpg')

img = cv2.resize(img, (300,300))

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

thres, res = cv2.threshold(gray, 150, 255, cv2.THRESH\_BINARY\_INV)

#*dilation => การขนาดพื้นที่*

#*สร้าง array ประกอบไปด้วย 1 ขนาด 2x2*

kernel = np.ones((2,2), np.uint8)

dilation = cv2.dilate(res,kernel,iterations=3)

#*erosion*

erosion = cv2.erode(res,kernel,iterations=3)

#*opening*

opening = cv2.morphologyEx(dilation, cv2.MORPH\_OPEN, kernel, iterations=5)

cv2.imshow('ORIGINAL', gray)

cv2.imshow('THRES', res)

cv2.imshow('DILATION', dilation)

cv2.imshow('EROSION', erosion)

cv2.imshow('OPENING', opening)

cv2.waitKey(0)

cv2.destroyAllWindows()

Several images of different types of coins

Description automatically generated

**Convolution : กำจัด Noise ออกจากภาพ ใช้ kernel กรองรูปภาพ**

คือการปรับปรุงคุณภาพของภาพโดยใช้วิธีกรองข้อมูลภาพเพื่อกำจัดสิ่งรบกวนออกจากภาพ โดยอาศัยหลักคณิตศาสตร์ที่เรียกว่า Convolution

Method: cv2.filter2D(1,2,3)

import cv2

import numpy as np

import matplotlib.pyplot as plt

kernel3 = np.ones((3,3), np.float32) / 9

kernel5 = np.ones((5,5), np.float32) / 25

img = cv2.imread('./image/noise.jpg', 0)

img = cv2.resize(img, (400, 400))

fil1 = cv2.filter2D(img, -1, kernel3)

fil2 = cv2.filter2D(img, -1, kernel5)

ori = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

res\_fil1 = cv2.cvtColor(fil1, cv2.COLOR\_BGR2RGB)

res\_fil2 = cv2.cvtColor(fil2, cv2.COLOR\_BGR2RGB)

titles = ["ORIGINAL", "CONVOLUTION 3x3", "CONVOLUTION 5x5"]

images = [ori, res\_fil1, res\_fil2]

for i in range(len(images)):

    plt.subplot(1, 3, i+1)

    plt.imshow(images[i])

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.show()

A collage of images of a person in a warehouse

Description automatically generated

**Blur คือการเบลอภาพเพื่อลด noise อีกวิธีนึง**

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('./image/noise.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

kernel = np.ones((5,5), np.float32) / 25

ori = cv2.cvtColor(gray, cv2.COLOR\_BGR2RGB)

#*filter2D*

filter2D = cv2.filter2D(gray, -1, kernel)

filter2D = cv2.cvtColor(filter2D, cv2.COLOR\_BGR2RGB)

#*blur*

blur = cv2.blur(gray, (5,5))

blur = cv2.cvtColor(blur, cv2.COLOR\_BGR2RGB)

images = [ori, filter2D, blur]

titles = ["ORIGINAL", "FILTER 2D", "MEAN"]

for i in range(len(images)):

    plt.subplot(1,3, i+1)

    plt.imshow(images[i])

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.show()

A collage of images of a person in a warehouse

Description automatically generated

Median filtering: เหมาะสำหรับนำไปใช้กำจัด noise มากกว่า mean filter

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('./image/noise.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

kernel = np.ones((5,5), np.float32) / 25

ori = cv2.cvtColor(gray, cv2.COLOR\_BGR2RGB)

#*filter2D*

filter2D = cv2.filter2D(gray, -1, kernel)

filter2D = cv2.cvtColor(filter2D, cv2.COLOR\_BGR2RGB)

#*blur*

blur = cv2.blur(gray, (5,5))

blur = cv2.cvtColor(blur, cv2.COLOR\_BGR2RGB)

#*median*

median = cv2.medianBlur(gray, 5)

median = cv2.cvtColor(median, cv2.COLOR\_BGR2RGB)

images = [ori, filter2D, blur, median]

titles = ["ORIGINAL", "FILTER 2D", "MEAN", "MEDIAN"]

for i in range(len(images)):

    plt.subplot(2,2, i+1)

    plt.imshow(images[i])

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.show()

Several images of boxes stacked on a shelf

Description automatically generated

**Gaussian filter: เป็นการลดสัญญาณรบกวน ยอดนิยม**

**Cv2.GaussianBlur(image, ( ขนาดตัวกรอง ), sigma)**

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('./image/noise.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

kernel = np.ones((5,5), np.float32) / 25

ori = cv2.cvtColor(gray, cv2.COLOR\_BGR2RGB)

#*filter2D*

filter2D = cv2.filter2D(gray, -1, kernel)

filter2D = cv2.cvtColor(filter2D, cv2.COLOR\_BGR2RGB)

#*blur*

blur = cv2.blur(gray, (5,5))

blur = cv2.cvtColor(blur, cv2.COLOR\_BGR2RGB)

#*median*

median = cv2.medianBlur(gray, 5)

median = cv2.cvtColor(median, cv2.COLOR\_BGR2RGB)

#*gaussian blur*

gauss = cv2.GaussianBlur(gray,(5,5),5)

gauss = cv2.cvtColor(gauss, cv2.COLOR\_BGR2RGB)

images = [ori, filter2D, blur, median, gauss]

titles = ["ORIGINAL", "FILTER 2D", "MEAN", "MEDIAN", "GAUSSIAN"]

for i in range(len(images)):

    plt.subplot(2,3, i+1)

    plt.imshow(images[i])

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.show()

A collage of images of a stack of boxes

Description automatically generated

A close-up of several round objects

Description automatically generatedEdge detection: การตรวจจับขอบภาพ

ซึ่งอัลกอริทึมในการตรวจจับหาขอบภาพมีหลายวิธี เช่น

* การตรวจจับขอบภาพวิธีโซเบล ( Sobel method )
* การตรวจจับขอบภาพวิธีลาปลาเซียน ( Laplacian method )
* การตรวจจับขอบภาพวิธีแคนนี่ ( Canny method )

**การหาเส้นเค้าโครงของภาพ ( Contours )**

คำสั่ง cv2.findContours( image, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

import cv2

img = cv2.imread('./image/ant.jpg')

img = cv2.resize(img, (500,500))

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

thres, res = cv2.threshold(gray, 200, 255, cv2.THRESH\_BINARY)

contours, hiera = cv2.findContours(res, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

cv2.drawContours(img, contours, -1, (0, 255, 0), 2)

print(len(contours))

cv2.imshow('output', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

A screenshot of a phone

Description automatically generated

**Object size measurement**

Setting camera ( cap.set() )

cap = cv2.VideoCapture(0)

cap.set(10, 160)

cap.set(3, 1980)

cap.set(4, 1080)

|  |  |  |
| --- | --- | --- |
| Prop ID | Properties | value |
| 0 | CV\_CAP\_PROP\_POS\_MSEC Current position of the video file in milliseconds. |  |
| 1 | CV\_CAP\_PROP\_POS\_FRAMES 0-based index of the frame to be decoded/captured next. |  |
| 2 | CV\_CAP\_PROP\_POS\_AVI\_RATIO Relative position of the video file |  |
| 3 | CV\_CAP\_PROP\_FRAME\_WIDTH Width of the frames in the video stream. |  |
| 4 | CV\_CAP\_PROP\_FRAME\_HEIGHT Height of the frames in the video stream. |  |
| 5 | CV\_CAP\_PROP\_FPS Frame rate. |  |
| 6 | CV\_CAP\_PROP\_FOURCC 4-character code of codec. |  |
| 7 | CV\_CAP\_PROP\_FRAME\_COUNT Number of frames in the video file. |  |
| 8 | CV\_CAP\_PROP\_FORMAT Format of the Mat objects returned by retrieve() . |  |
| 9 | CV\_CAP\_PROP\_MODE Backend-specific value indicating the current capture mode. |  |
| 10 | CV\_CAP\_PROP\_BRIGHTNESS Brightness of the image (only for cameras). |  |
| 11 | CV\_CAP\_PROP\_CONTRAST Contrast of the image (only for cameras). |  |
| 12 | CV\_CAP\_PROP\_SATURATION Saturation of the image (only for cameras). |  |
| 13 | CV\_CAP\_PROP\_HUE Hue of the image (only for cameras). |  |
| 14 | CV\_CAP\_PROP\_GAIN Gain of the image (only for cameras). |  |
| 15 | CV\_CAP\_PROP\_EXPOSURE Exposure (only for cameras). |  |
| 16 | CV\_CAP\_PROP\_CONVERT\_RGB Boolean flags indicating whether images should be converted to RGB. |  |
| 17 | CV\_CAP\_PROP\_WHITE\_BALANCE Currently unsupported |  |
| 18 | CV\_CAP\_PROP\_RECTIFICATION Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently) |  |