**Pedoman Image Processing dari Youtube**

Cv2.imread(‘alamat gambar’, flags)

**Flags tuh yang menentukan gambar tuh mau dibaca kaya gimana**

Cv2.IMREAD\_COLOR --- 1 --- Loads color image

Cv2.IMREAD\_GRAYSCALE --- 0 --- jadi grayscale

Cv2.IMREAD\_UNCHANGED --- -1 --- loads image including aplha channnel

**import** cv2

img **=** cv2.imread('D:/A LEARN/Image Processing/Program Youtube/lena.jpg', 0)

print (img)

Misal imread

Terus dimasukin ke variable terus di print

Jika hasilnya **None**  maka

**Pengindeksan alamat dari gambar nya salah**

Harusnya bisa nampilin isi yaitu sebuah matriks

[[163 162 161 ... 170 154 130]

[162 162 162 ... 173 155 126]

[162 162 163 ... 170 155 128]

...

[ 43 42 51 ... 103 101 99]

[ 41 42 55 ... 103 105 106]

[ 42 44 57 ... 102 106 109]]

**Tadi kan ngeload doang, sekarang nampilin**

Cv2.waitKey()

Cv2.waitKey(0)

Dan bisa ditampilln berapa lamanya

Cv2.waitKey (5000)

Berarti 5 detik

**Kemudian dihapus semua windows yang tertampilnya nya**

Dengan fungsi

Cv2.destroyAllWindows()

Cv2.destroyWindows()

**Kalo mau ngesave gambar**

Pake imwrite

Cv2.imwrite (filename, img)

Contoh : cv2.imwrite('D:/A LEARN/Image Processing/Program Youtube/Gambar/lena\_grayscale.PNG', img)

**Bisa juga langsung convert ke JPG ke PNG**tanpa ada fungsi atau atribut lain

**Waitkey bisa digunakan untuk inputan keyboard** **terus diolah**

Misal mencet escape jadi keluar foto

Mencet save jadi ngesave fotonya.

k **=** cv2.waitKey()

**if** k **==** 27:

    cv2.destroyAllWindows()

**elif** k **==** ord('s'):

    cv2.imwrite('D:/A LEARN/Image Processing/Program Youtube/Gambar/lena\_grayscale.PNG', img)

    cv2.destroyAllWindows()

**27 tuh buat mengindikasikan Escape**

**Kalo ord**

Buat biar bisa inputan keyboard biar bisa dibandingin sama string

**Menampilkan video**

**import** cv2

cap **=** cv2.VideoCapture(0)

**while** (True):

    ret, frame **=** cap.read()

    cv2.imshow('video frame', frame)

**if** cv2.waitKey(1)  **==** ord('q'):

**break**

cap.release()

cv2.destroyAllWindows()

Cap = cv2.VideoCapture (0)

0 merupakan indeks kamera yang dipakai

0 🡪 kamera default

1 🡪 kamera eksterna

Dst.

Kadang kadang kamera default juga bisa -1

If cv2.waitKey(1) == ord(‘q’)

waitKey berguna untuk mendelay informasi

istilah nya mungkin upload speed nyaa

kalo cv2.waitKey(1000)

update video nya per 1 detik

kalo cv2.waitKey (1)

update videonya per 1 milli second

Cap.release() 🡪 gatau buat apa

**Kalo mau Grayscale gini nih**

    gray **=** cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    cv2.imshow('video frame grayscale', gray)

**CAP.isOpened**

Untuk mengecek apakah kamera yang di indeks ada atau engga

print (cap.isOpened())

*#cap.isOpened()*

*#bisa juga dipake buat kondisi while*

*#while cap.isOpened()*

*#jadi kalo False dibawahnya gaakan dieksekusi*

**Mengetahui Height dan Weight dari video**

    a **=** cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

    b **=** cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

    print ('Width : {}, Height : {}'.format(a,b))

**Kalo mau ngesave video gini nih**

**import** cv2

cap **=** cv2.VideoCapture(0)

fourcc **=** cv2.VideoWriter\_fourcc(**\***'XVID')

out **=** cv2.VideoWriter('D:/A LEARN/Image Processing/Program Youtube/Gambar/outputlagi.avi', fourcc, 20.0, (640,480))

**while** True:

    ret, frame **=** cap.read()

**if** ret **==** True :

*#output warna*

*# out.write (frame)*

        gray **=** cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

*#output grayscale*

        out.write (frame)

        cv2.imshow('video grayscale', gray)

**if** cv2.waitKey(1) **==** ord('q'):

**break**

**else** :

**break**

cap.release()

out.release()

cv2.destroyAllWindows()

pake fourcc – kode gatau buat apa

tipe nya XVID

bisa ditulis :

* (\*’XVID’) or
* (‘X’, ‘V’, ‘I’, ‘D’)

**Kalo gray gabisa**

Mungkin karena convert warna jadi gabisa

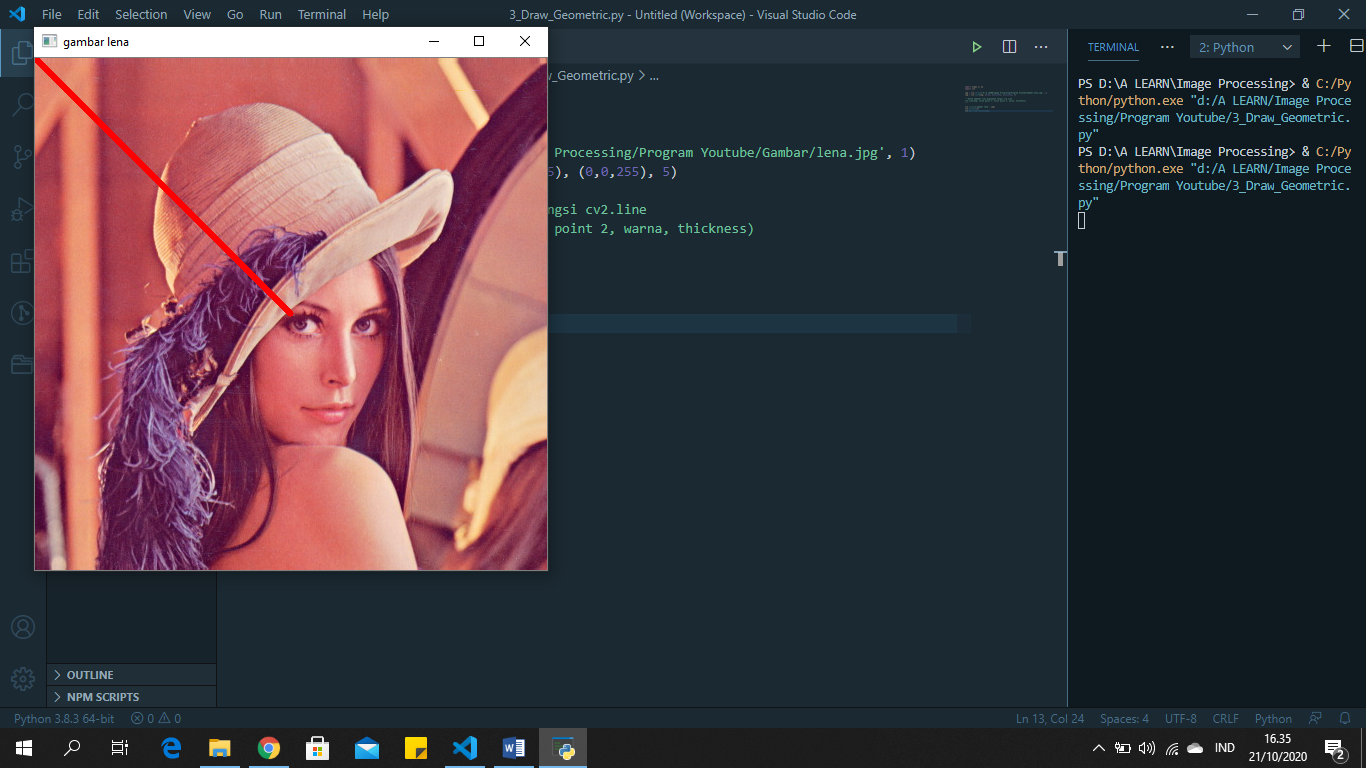
Kalo warna biasa bisa

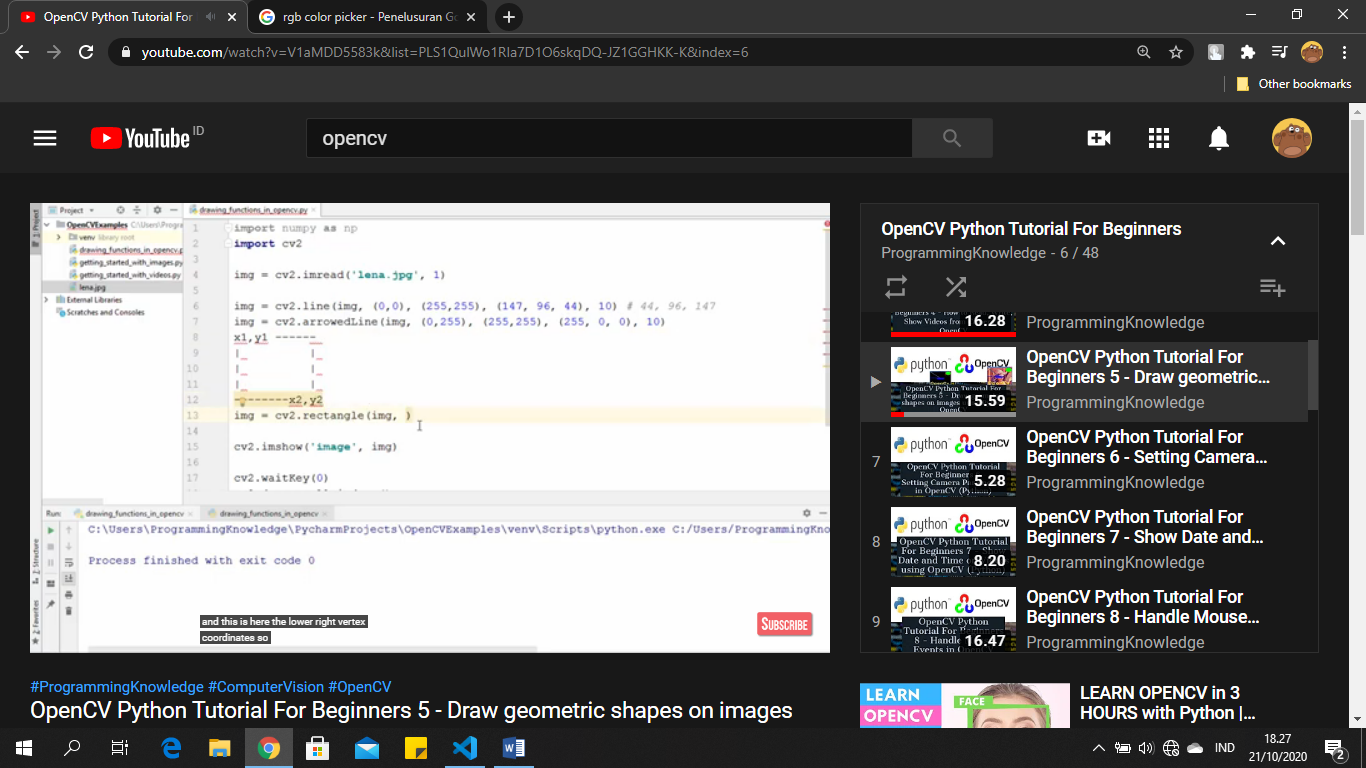
**Kalo mau gambar line di picture gini nii**

img **=** cv2.line(img, (0,0), (255,255), (0,0,255), 5)

Untuk membuat line digunakan fungsi cv2.line

cv2.line(img, titik point 1, titik point 2, warna, thickness)



**Atau rectangle**

X1 y1 dan x2 y2 nya gitu tuu

Pokonya mah kita bisa ngegambar geomatrik :

* Garis
* Garis Penunjuk
* Rectangle
* Bulet
* Ellips
* Polyline

**Dan bisa ngegenerate gambar sendiri dari numpy**

img **=** np.zeros([512,512,3]) *#blackscreen*

img **=** np.ones([512,512,3]) *#whitescreen*

zeros buat blackscreen

ones buat whitescreen

3 tuh gatau buat apaa

**Resolusi kamera tuh disettingnya kaya gini niii**

cap.set(3,300)

cap.set(4,300)

3 tuh buat width

4 tuh buat height

Dan program akan menyesuaikan sesuai resolusi kamera

Jadi gausah pake

*# print(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))*

*# print(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))*

**Cara nampilin event mouse**

**for** i **in** dir(cv2):

**if** 'EVENT' **in** i:

        print (i)

EVENT\_FLAG\_ALTKEY

EVENT\_FLAG\_CTRLKEY

EVENT\_FLAG\_LBUTTON

EVENT\_FLAG\_MBUTTON

EVENT\_FLAG\_RBUTTON

EVENT\_FLAG\_SHIFTKEY

EVENT\_LBUTTONDBLCLK

EVENT\_LBUTTONDOWN

EVENT\_LBUTTONUP

EVENT\_MBUTTONDBLCLK

EVENT\_MBUTTONDOWN

EVENT\_MBUTTONUP

EVENT\_MOUSEHWHEEL

EVENT\_MOUSEMOVE

EVENT\_MOUSEWHEEL

EVENT\_RBUTTONDBLCLK

EVENT\_RBUTTONDOWN

EVENT\_RBUTTONUP

**SetMouseCallback**

'''SetMouseCallback tuh

default nya langsung ngasih 5 parameter otomatis yaitu

event, x, y, flags, param

udah gabisa diganggu gugat itumaa

'''

**def** click\_event(**event**, **x**, **y**, **flags**, **param**):

**if** event **==** cv2.EVENT\_LBUTTONDOWN:

        print(x,', ',y)

        font **=** cv2.FONT\_HERSHEY\_SIMPLEX

        strXY **=** str(x) **+** ', ' **+** str(y)

        cv2.putText(img, strXY, (x,y), font, 1, (255,255,0), 2, cv2.LINE\_AA)

        cv2.imshow('image', img)

img **=** np.zeros((512,512,3), np.uint8)

cv2.imshow('image', img)

cv2.setMouseCallback('image', click\_event)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Kalo mau indexing warna BGR dari gambar**

        blue **=** img[x, y, 0]

        green **=** img[y, x, 1]

        red **=** img[y, x, 2]

gangerti kenapa harus begitu

mungkin

img[width, height, Indeks warna keberapa)

0 🡪 biru

1 🡪 green

2 🡪 red

**Antara comma (,) atau Concat (+)**

    '''Di string nya jan pake coma(,)

    pake concat aja (+)

    soalnya ga reliable pake , maa'''

'''Soalnya nanti kalo pake , terus ada castingan bakal

a = 1

print ('hayu', str(a))

bakal jadi: hayu, 1

kalo concat kan

print ('hayu + str(a))

bakal jadi : hayu 1

'''

Intinya kalo kalo di ada variable yang di **casting**

Terus diprint sam yang ga di casting

Terus pake ,

**Komanya bakal ngikut**

**Kalo mau ngebuat garis garis dari titik gini niiiih**

img **=** np.zeros((1024, 780, 3))

**def** click\_event(**event**, **x**, **y**, **flags**, **param**):

*global* points

**if** event **==** cv2.EVENT\_LBUTTONDOWN:

        cv2.circle(img, (x,y), 3, (0,255,0))

        points.append((x,y))

        print (points)

        cv2.imshow('image', img)

**if** len(points) **>=**  2:

            cv2.line(img, (points[0][0], points[0][1]), (points[1][0], points[1][1]),(0,255,0), 3, cv2.LINE\_AA)

            points **=** []

            cv2.imshow('image', img)

cv2.imshow('image', img)

points **=** []

cv2.setMouseCallback('image', click\_event)

**Untuk menampilkan ini sebeneranya warna apa sii, di windows baruu**

**import** cv2

**import** numpy **as** np

**def** click\_event(**event**, **x**, **y**, **flags**, **param**):

**if** event **==** cv2.EVENT\_LBUTTONDOWN:

        blue **=** img[x,y,0]

        green **=** img[x,y,1]

        red **=** img[x,y,2]

        imagecolor **=** np.zeros((512,512,3), np.uint8) *#uint kudu wajib dipake banget*

        imagecolor[:] **=** [blue, green, red]

        cv2.imshow('color', imagecolor)

img **=** cv2.imread('D:/A LEARN/Image Processing/Program Youtube/Gambar/lena.jpg', 1)

cv2.imshow('image', img)

cv2.setMouseCallback('image', click\_event)

cv2.waitKey()

cv2.destroyAllWindows()

np.uint8

HARUS wajib dipake banget supaya ngedefinisiin image yang kita buat tuh tipe data apa dan besarnya berapa

Sehingga nanti bisa di assignment misal uint8

Berarti int 8 bit

Range nya 0-255, sehingga nanti bisa di assign segitu

**Metode shape size dtype split dan merge**

print(img.shape) *#returns a tuple of rows, columns and chanels*

print(img.size) *#return total number of pixel (row\*columns\*chanell)*

print(img.dtype) *#return image datatype is obtained*

b,g,r **=** cv2.split(img)  *#ngesplit semua bgr dalam setiap pixel di gambar*

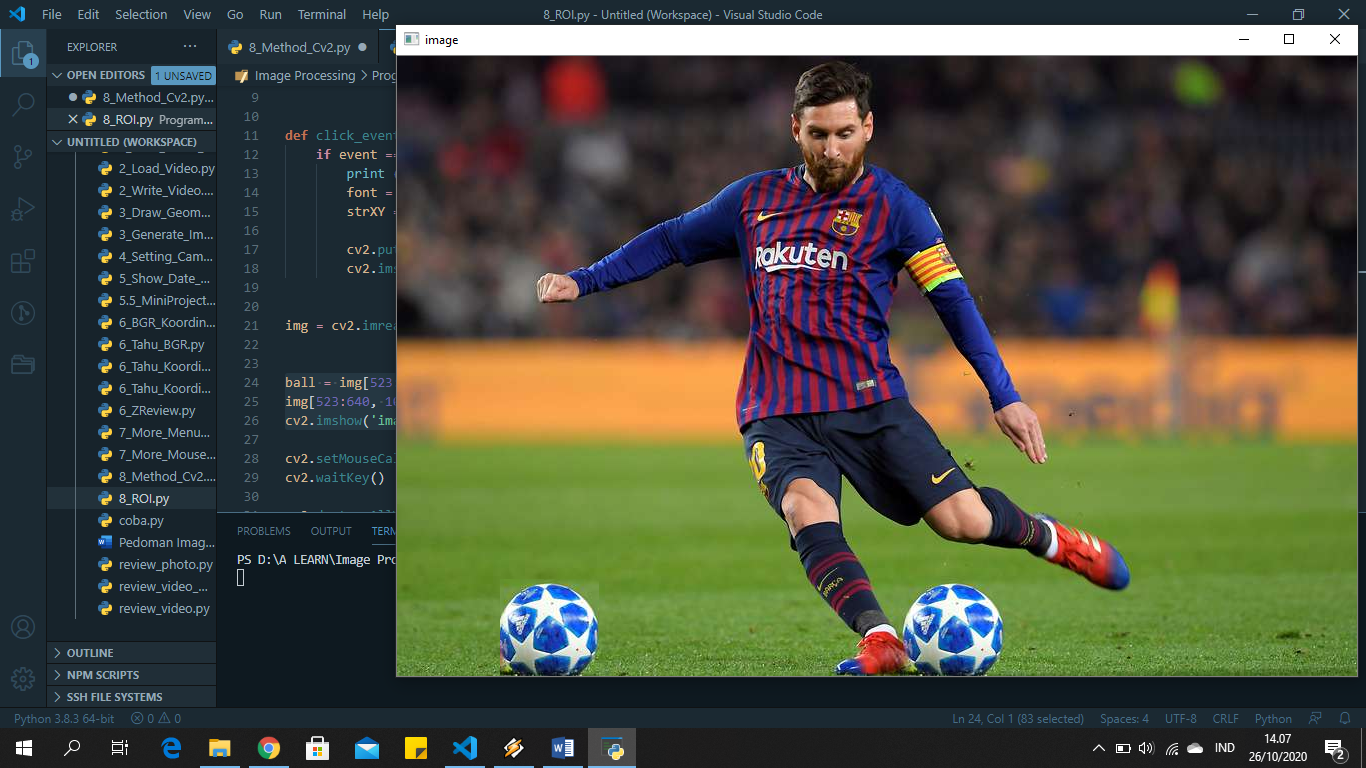
img **=** cv2.merge((b,g,r)) *#nyatuin semua bgr perpixel jadi gambar lagi*

**ROI (Region of Interest)**

**Jadi diseleksi gambarnya sesuai dengan yang diinginkan, lalu dipindahkan**

ball **=** img[523:640,508:607]

img[523:640, 103:202] **=** ball

cv2.imshow('image', img)

itu bolanya 2 mess

formatnya baris x kolom yee, kaya numpy

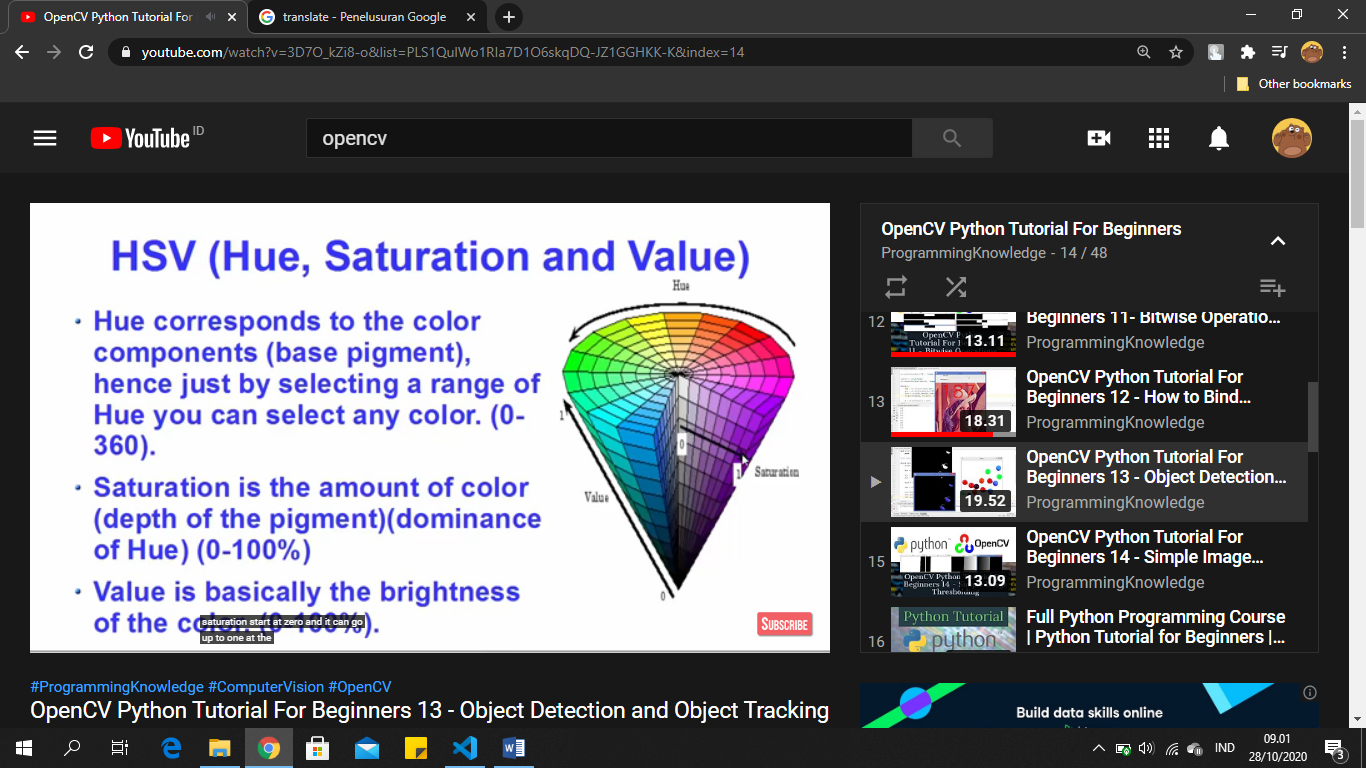
**HSV (Hue, Saturation Value)**

Adalah sebuah format nama mirip seperti RGB

H(hue) adlah pilihan warna nya, biasanya standar merah kuning ijo, biru, cyan magenta

Saturation adalah seberapa terang (range nya dari biasa ke terang)

Value adalah seberapa gelap (range nya dari biasa ke gelap)



**Resize dan Add**

Kalo mau di add harus di resize dulu biar ga error

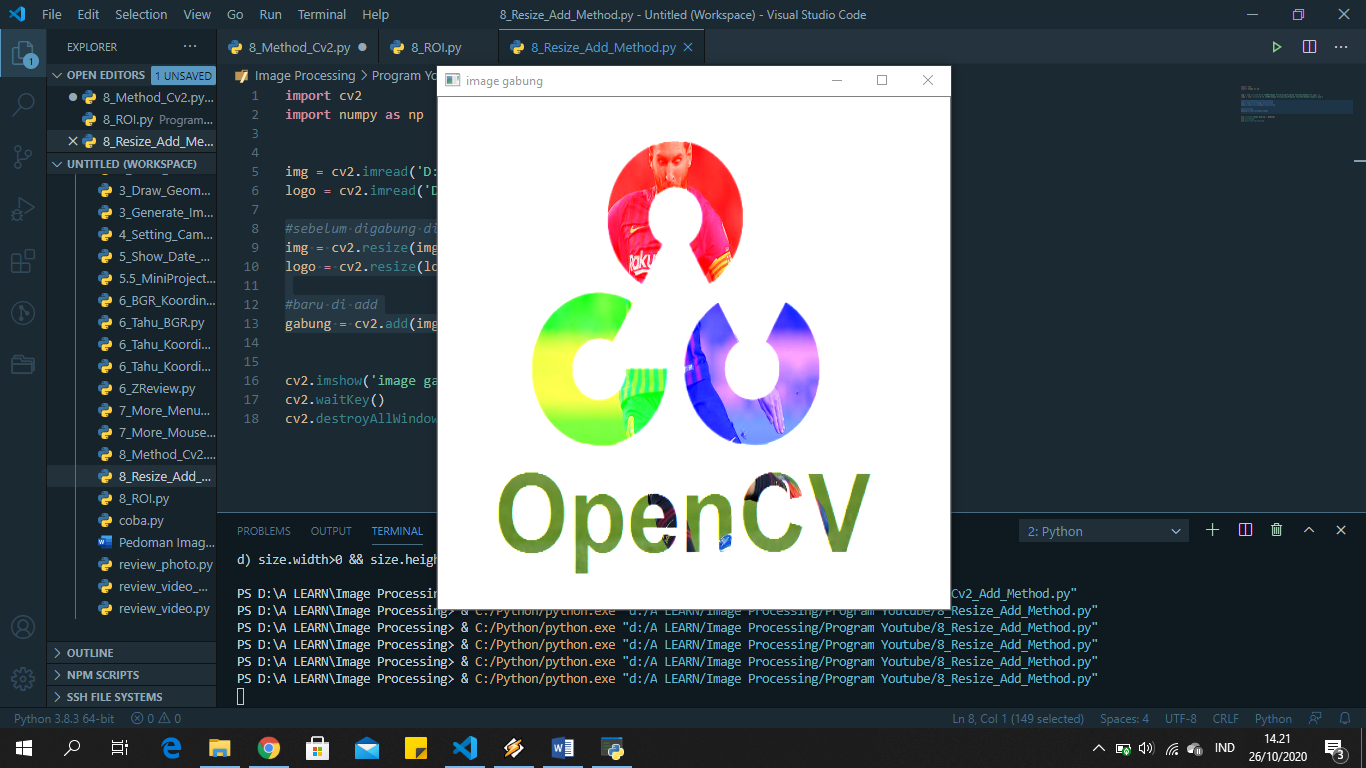
*#sebelum digabung di resize duls*

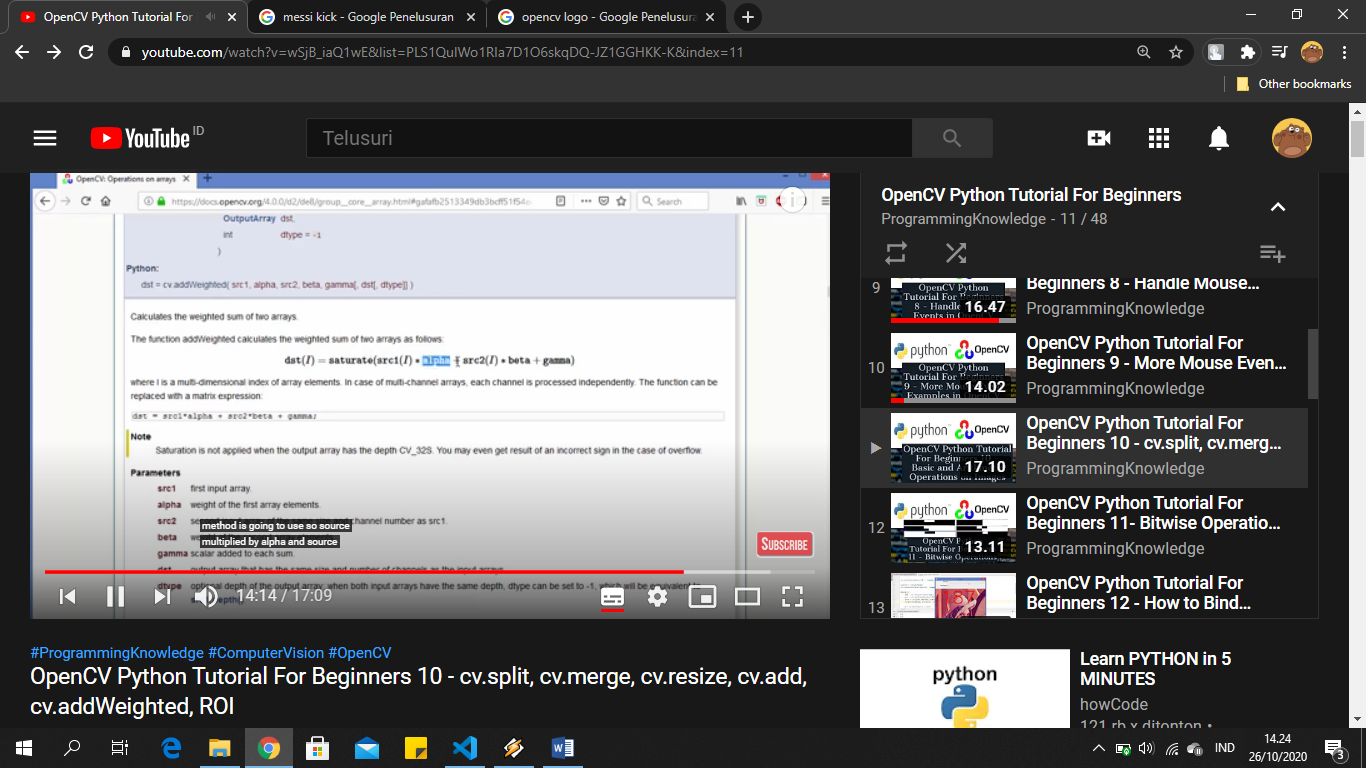
img **=** cv2.resize(img, (512,512))

logo **=** cv2.resize(logo, (512,512))

*#baru di add*

gabung **=** cv2.add(img, logo)

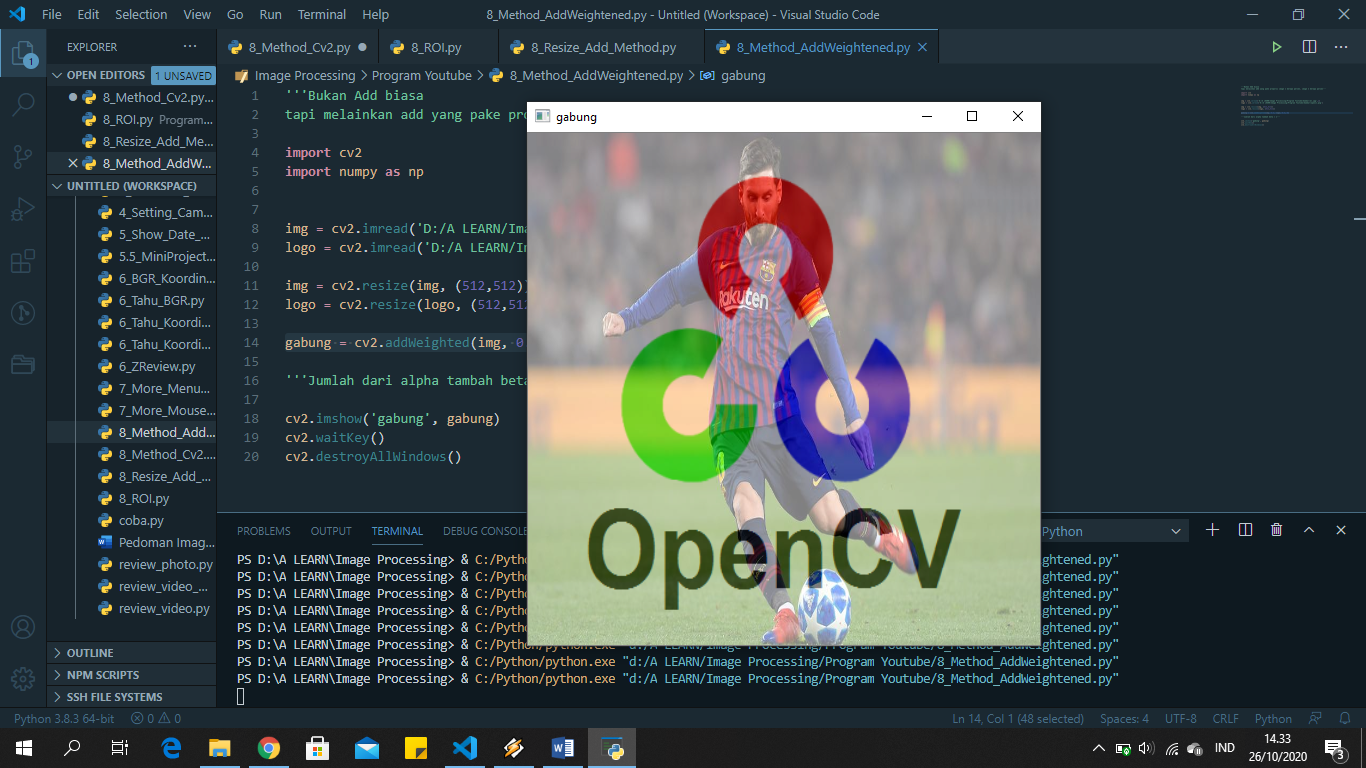


**ADDWEIGHTENed**

gabung **=** cv2.addWeighted(img, 0.5, logo, 0.5, 0)

Argumen tambahan ada alpha beta dan gamma

Alpha + beta = 1



**Bitwise Operations**

Jadi image juga bisa diolah dengan menggunakan operasi logika, ada

AND

OR

XOR

NOT dsb

EMANG GUNANYA APA SIII???

Buat mask nantii

**TERNYATA IMAGE TUH**

Isinya pixel yang direpresentasikan dalam matriks

Misal image 512x512 pixel putih semua

Berarti jumlah baris nya ada 512, jumlah kolomnya ada 512

Dan setiap barisnya memiliki 3 chanell yaitu BGR

Bayangin 512x512x3 = jumlah angka dalam image

Img[:] = ini tuh matriks semua yang setiap pixel nya ngerepresentasiin warna

**Membuat Trackbar**

**import** numpy **as** np

**import** cv2

**def** nothing(**x**):

    print (x)

img **=** np.zeros((300,512,3), np.uint8)

cv2.namedWindow('image') *#membuat windownya*

*#membuat trackbar, tapi belum bisa dipake*

cv2.createTrackbar('B', 'image', 0, 255, nothing)

cv2.createTrackbar('G', 'image', 0, 255, nothing)

cv2.createTrackbar('R', 'image', 0, 255, nothing)

**while** True :

    cv2.imshow('image', img)

    k **=** cv2.waitKey(1)

**if** k **==** 27:

**break**

*#manggil trackbar biar valuenya bisa dipake*

    b **=** cv2.getTrackbarPos('B', 'image')

    g **=** cv2.getTrackbarPos('G', 'image')

    r **=** cv2.getTrackbarPos('R', 'image')

    img[:] **=** [b,g,r]

cv2.destroyAllWindows()

ada bikin

dan ada ngambil value

waitKey juga harus (1) biar cepet proses nya

kalo waitKey(), gaakan jalan karena bingung

**CVTCOLOR tuh harus dimasukkin ke variable lalu bisa ditampilin**

Kalo gagitu gabisa

img **=** cv.cvtColor(img,cv.COLOR\_BGR2GRAY)

**Thresholding**

Untuk memisahkan sebuah objek dari background nyaa

Atau memilih warna (Range warna) yang ingin aja sisanya di itemin atau dibiarin

Konsepnya mirip kaya detector yang kemaren

Atau **nge masking** lahh

\_, th1 **=** cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)

*#nyeleksi nilai pixel misal antara 127-255, kalo bener dikasih logic 1, sisanya 0*

\_, th2 **=** cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY\_INV)

*#sama aja, tapi kalo bener dikasih logic 0*

\_, th3 **=** cv2.threshold(img, 127, 255, cv2.THRESH\_TRUNC)

*#dari nilai 127 sampe 255, nilai pixelnya disamain jadi 127*

\_, th4 **=** cv2.threshold(img, 127, 255, cv2.THRESH\_TOZERO)

*#nilai yang kurang dari diitemin (0), sisanya dibiarin*

\_, th5 **=** cv2.threshold(img, 127, 255, cv2.THRESH\_TOZERO\_INV)

*#kebalikannya*

cv2.imshow('image', img)

cv2.imshow('th1', th1)

cv2.imshow('th2', th2)

cv2.imshow('th3', th3)

cv2.imshow('th4', th4)

cv2.imshow('th5', th5)

Methode threshold akan mengekstrak 2 nilai

1. Ret
2. Threshold nya 🡪 ini yang dipake

Beberapa jeis threshold:

Thresh Binary 🡪 yang diluar range diitemin(0), dalam range diputihin (1)

Thresh Binary Inv 🡪 Sebaliknya

Thresh Trunc🡪 Nilai yang dalam range disamain jadi satu nilai misal 127-255

Semuanya disamain nilainya jadi 127

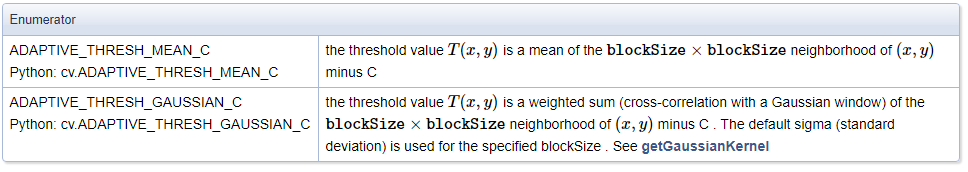
Thresh\_Tozero 🡪 nilai diluar range diitemin (0), sisanya dibiarin

Thresh\_Tozero inv 🡪 sebaliknya

**Adaptive Thresholding**

Simple Thresholding hanya akan menerapkan nilai thresholding pada global pixel, jadi semua pixel penerapannya sama

Dengan adaptive thresholding, nilai thresholding yang diterapkan hanya berlaku pada bagian bagian kecil, dan setiap bagian bagian itu pasti berbeda thresholding nya “namanya juga adaptive”



th2 **=** cv.adaptiveThreshold(img, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv.THRESH\_BINARY, 11, 2)

gini parameter yang di butuhkannya

11 tuh blocksize nya

2 tuh C nya

**Matplotlib on OpenCV**

Opencv ngebaca dan nulis pake BGR format

Sedangkan matplotlib pake RGB format

Sehingga pas ditampilin tanpa di convert dulu hasilnya warnaya beda

Enaknya pake matplotlib tuh :

* Bisa di pindah pindah
* Bisa di edit edit
* Ada koordinatnya
* Bisa di save
* Dll

**Morphological Transformation**

Memakai **Metode Kernel**  dan original image nyaa

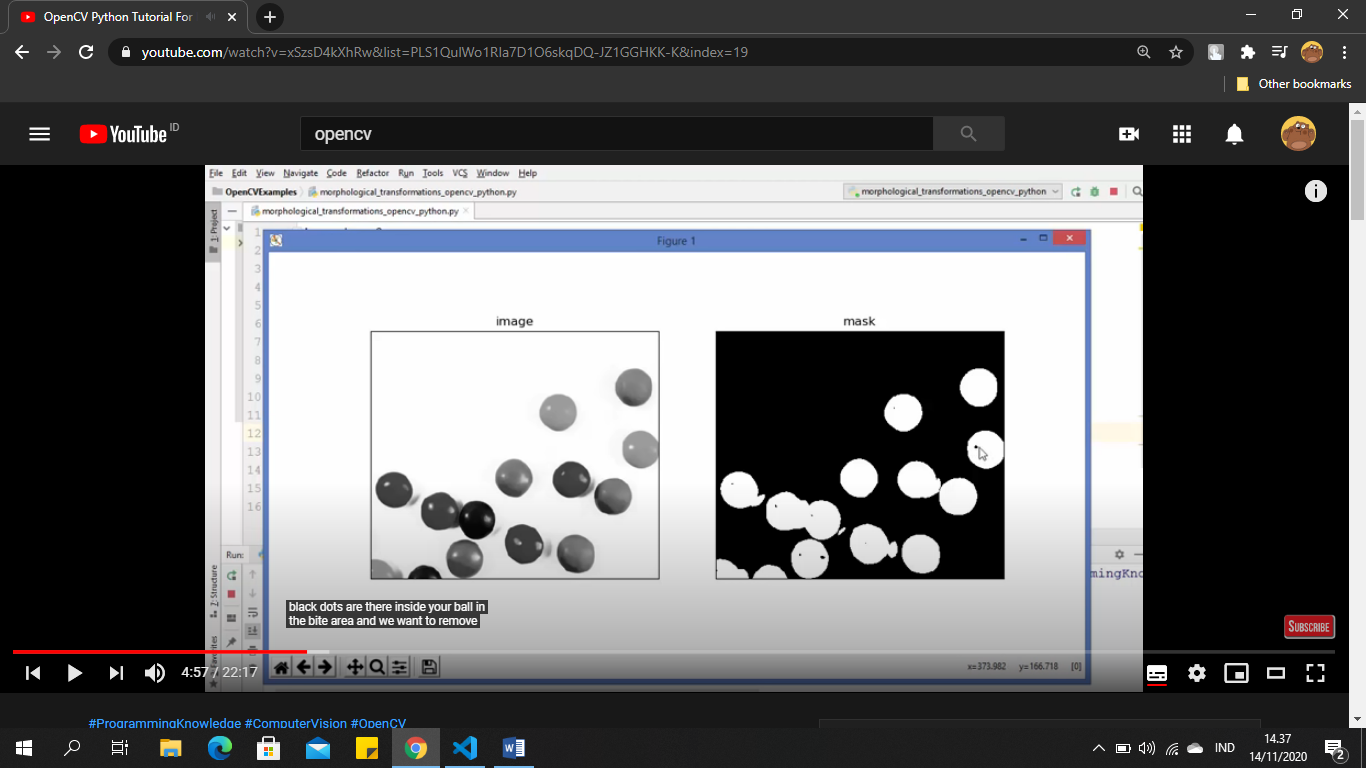
Apa sih metode kernel??

Kernel memberi tahu Anda cara mengubah nilai piksel tertentu dengan menggabungkannya dengan jumlah piksel tetangga yang berbeda

Biasanya **morphological transformation** diterapkan untuk binary images (Cuma ada nilai 1 0 di image nyaa, jadi harus kita masking dulu images nya dengan simple thresholding

Setelah di Masking

Menjadi gambar yang Cuma bernilai 1/0 ajaa



**Dilation**

Digunakan mereduksi titik titik dots hitam diantara putih

*#dilation*

kernal **=** np.ones((2,2), np.uint8)

dilation **=** cv.dilate(mask, kernal, **iterations=**5)

kernal 🡪 sebuah matriks yang akan diterapkan pada morphological

iteration 🡪 diterapkan berapa kali

nilai ukuran matriks kernal dan iteration bisa diubah ubah sesuai kebutuhan

Jadi dilation tuh ngasih matriks dengan iteration tertentu ke mask yang sesuai dengan nilai binernya (misal kalo matriks ones ngaruhnya ke mask putih, kalo matriks zeros ngaruhnya ke mask item)

Intinya:

Dilation 🡪 Pelebaran, jadi makin lebar si putihnya

Erosion 🡪 Pengikisan, semakin kecil si putihnya (dikikis jadi 0)

Tujuan nya sama untuk menyamakan

Opening 🡪 Erosion dulu baru dilation

Closing 🡪 Dilation dulu baru Erosion

**Image Gradient**

Jadi menseleksi gambar sesuai dengan kontur luarnya

Dan bekerja sangat baik ketika diterapkan di image yang **abu abu** bukan yang berwarna

Laplacian 🡪 garis gradiennya kasar, cenderung kaya scatter, tapi bagus juga, sumbu x dan y nya dapet

SobelX 🡪 garis gradiennya berdasarkan vertikal aja, horizontal diacuhkan

SobelY 🡪 garis gradiennya berdasarkan horizontal

Kebalik sih harusnya wkwk

SobelCombine 🡪 SobelX dan sobelY di or in jadi keren

**Jadi fungsi fungsi ini merupakan hasil dari operasi matematika sehingga ketemu gradient garis dari gambar.** Konsep:

Laplacian – menghitung turunan laplace

Sobel – joint Gaussion dan differentiation operations

**Laplace**

Fungsinya gampang

Cuma parameterna rada nyebelin

lap **=** cv.Laplacian(img, cv.CV\_64F, **ksize=**3)

lap **=** np.uint8(np.absolute(lap))

CV\_64F -- Adalah Cuma data type (64 bit float) yang mengsupport negatif number pada operasi laplace

Terus diambil nilai absolute dari lap

Terus diubah ke **int 8 bit lagi**

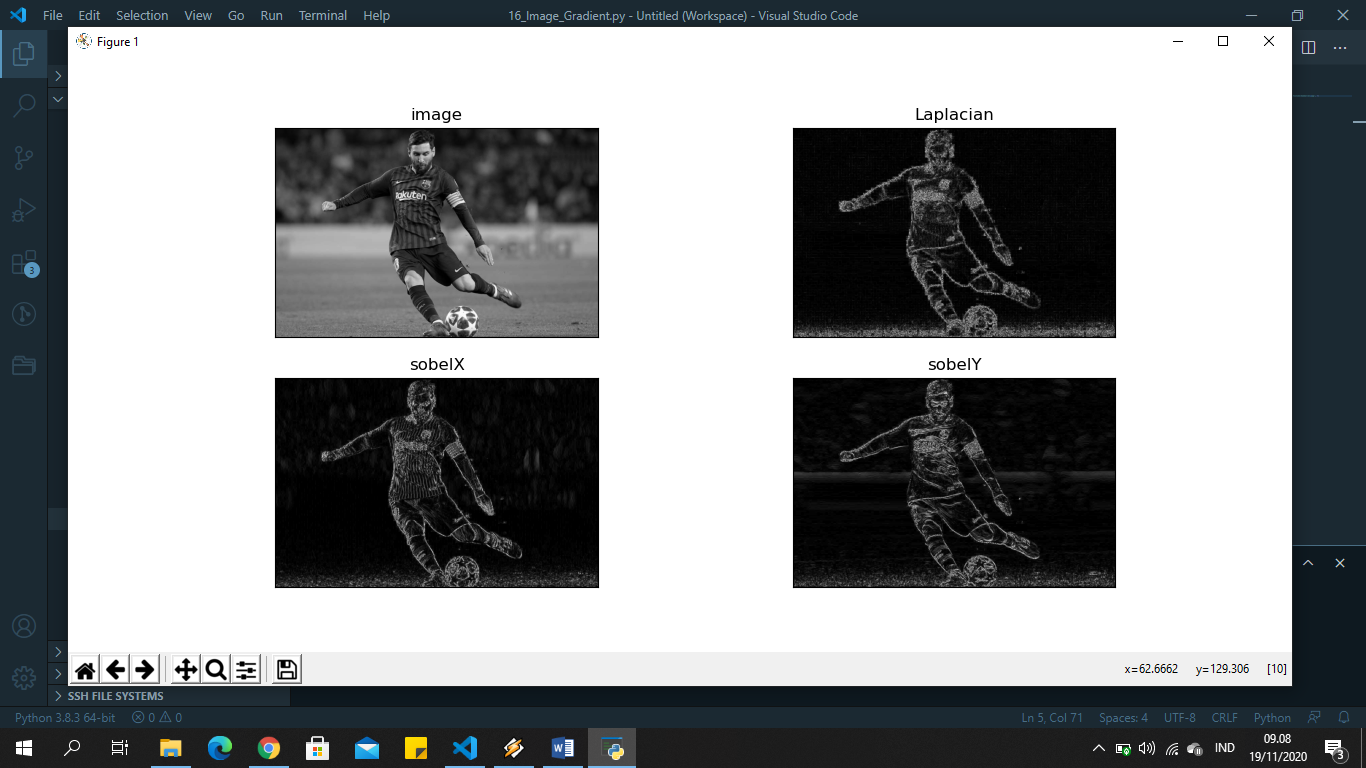
**Sobel**

sobelX **=** cv.Sobel(img, cv.CV\_64F, 1, 0, **ksize=**3)  *#dx =1, dy = 0*

sobelY **=** cv.Sobel(img, cv.CV\_64F, 0, 1, **ksize=**3) *#dx =0, dy = 1*

dx atau dy satu aja yang diisi, jangan dua duanya nanti combine

atau bisa di combine juga sama OR

Hasilnya

**CANNY (Edge Detection)**

Fungsinya sama kaya image gradient Cuma kalo dibandingin noise nya lebih kecil

Konsepnya Canny tuh gini nih

Edge detection operator dengan multistage algoritm step stepnya antara lain :

1. Noise reduction

Apply gaussian filter to smooth the image dengan tujuan untuk remove the noise

1. Gradient Calculation

Find the intensity gradient dari image

1. Non Maximum suppresion

To apply non maksimum suppresion to get rid of the spurious response to edge detection

1. Double threshold

Apply double threshold to determine the potential edges

1. Edge Tracking by Hysterisis

Track edges by hysterisis

Tapi santai makenya Cuma

Cv.Canny doang ko

**Intinya sama kaya image gradient**

**Cuma Thresholding nya bisa diatur**

**SEHINGGA NOISENYA SEDIKIT**

**Medianblur harus ganjil 3 5 7 gaboleh 1**

**Smoothing/Blurring Images**

Sering dipakai ketika image processing

Tujuannya untuk me remove **noise**  di gambar

Kita gunakan **diverse linear filters**

Karena gampang penggunaanya dan hasilnya bagus

Ada beberapa macam filter :

* Homogeneous filter
* Gaussian filter
* Median filter
* Bilateral Filter

**Homogeneous Filter**

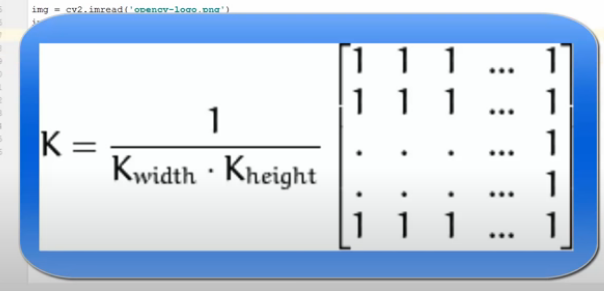
Is the most simple filter, each output pixel is the measn of its kernel neighbors

**What is kernel**??

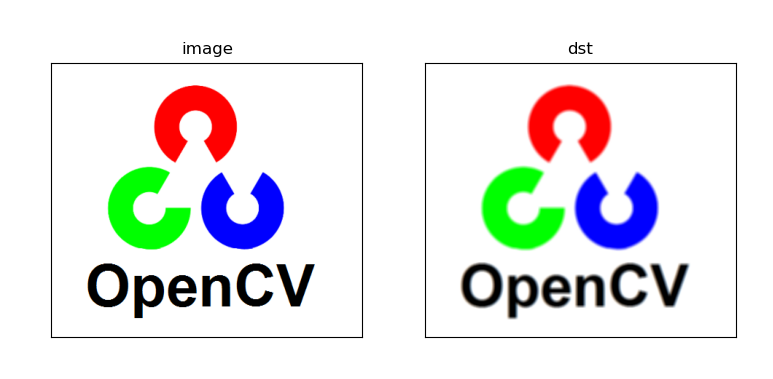
In image processing, a kernel, convolution matrix, or mask is a small matrix.

It is used fro blurring, sharpening, embossing, edge detection, and more

Di homogeneous filter, kernel yang dipakai rumusnya seperti ini



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



noise yang ada di gambar kiri di blur in dan lebih smooth

menggunakan filter 2D

As in one-dimensional signals, images also can be filtered with various *low-pass filters*(LPF), *high-pass filters* (HPF) etc.

* LPF membantu dalam meremove noise, dan blur gambar
* HPF filters membantu untuk mencari edges in the images

**Blur Function algoritm (Averaging blur)**

Method yang lain dalam opencv selain filter2d adalah blur function

blur **=** cv.blur(img, (5,5))

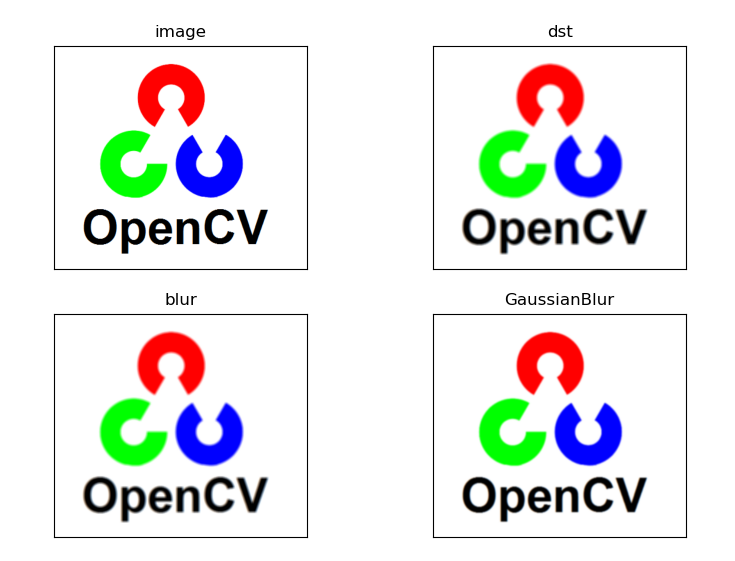
hasilnya gajauh beda sama filter2d karena kernel yang dipake juga kan hampir sama

**Gaussian Filter Argoritm**

Gaussian filter is nothing but using *different-weight-kernel*, in both x an y direction

Jadi hasilnya pertengahan kernel memiliki bobot pixel yang lebih tinggi dibanding dengan sisi sisinya



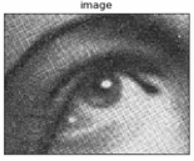


Secara awam gaussian blur terlihat lebih mantap daripada filter2d dan blur. SI gaussian teh blur tapi lebih smooth gitu

**Jadi di gaussian blur ini dirancang untuk meremove *high frequency noise* dari si gambar bisa dilihat haslinya di**



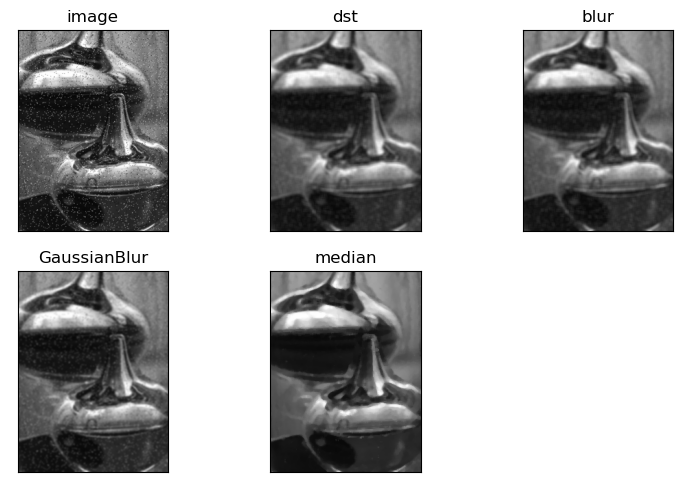
**High frequency noise** 🡪 noise nya seperti gurata guratan, jadi gambar nya gajelas karena gurata guratan itu



**Salt and pepper noise 🡪** noise nya seperti titik titik hitam dan putih



**Median Filter**

Median filter is something that replace each pixel’s value with the median of its neighboring pixels. This method is great when dealing with “**salt and pepper noise**”

Liat jadi bagus kan kalo pake median, si salt sama pepper jadi gaada

median **=** cv.medianBlur(img, 5)

kernelnya harus odd (ganjil)

**Bilateral Filter**

Jadi dengan memblur atau mensmooth gambar tuh kadang kadang edges nya jadi blur juga kan, gimana caranya si edges nya tuh biar tetep sharp, nah itu pake bilateral filter namanya.

**Jadi bilateral filter** sangat cocok dipakai untuk meremove noise sambil **menjaga edges nya ttp sharp**

Dalam artian bukan membuat edges yang blur jadi sharp yah

Bilateral filter, It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels

bilateralFilter **=** cv.bilateralFilter(img, 9, 75, 75)

9 🡪 diameter of eacg pixel neighborhood that is used during the filter

75 🡪 Sigma Color, filter sigma in the color space

75 🡪 Sigma Space, sigma space is the filter sigma in the coordinate space

**Kita bisa ngubah python script jadi .exe ataupun .app**

Jadi bisa di run gitu dengan klik 2x atau dijadiin installer juga bisa anjay

Semua itu memakai **pyinstaller**

**Pip install pyinstaller**

Membuat .exe

**Pyinstaller test.py**

Membuat .exe yang simple tanpa banyak file

**Pyinstaller --onefile test.py** atau

**Pyinstaller –F test.py**

Membuat .exe tanpa membuka jendela cmd nya

**Pyinstaller --onefile –w test.py**

Membuat .exe agar terhold, simple, dan ada iconnya

**pyinstaller -c -F -i cm\_icon.ico console\_monopoly.py**

**Jadi flagnya tuh ada apa aja :**

-- onedfile 🡪 biar simple

- F 🡪 biar simple

- w 🡪 biar gausah pake cmd

- c 🡪 biar ke hold

- i 🡪 biar tambah icon

Biar lengkap pake weh semuanya

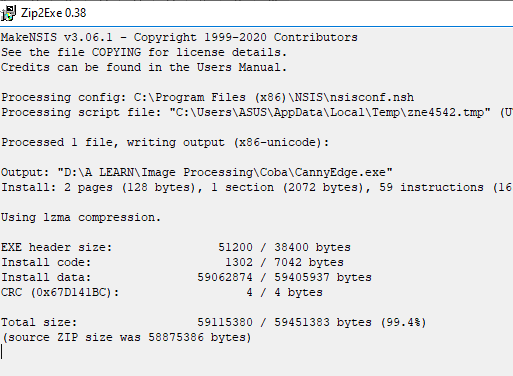
**Kita juga bisa buat .exe tadi jadi installer**

**Pake aplikasi lagi**

**Namanya NSIS**

Jadi kita buat zip dari exe nya

Terus di generate pake NSIS



Setelah itu exe nya bisa kita install di OS masing – masing, keren kannn

**Image Pyramids**

Pyrdown 🡪 membuat gambar ¼ kali menjadi lebih kecil dari gambar aslinya

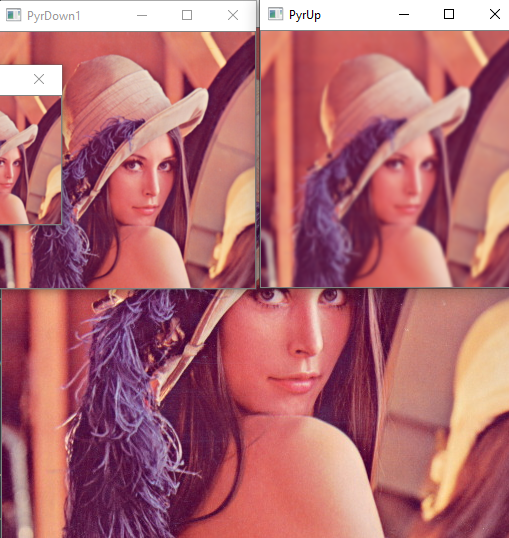


lr1 **=** cv.pyrDown(img)

lr2 **=** cv.pyrDown(lr1)

PyrUp 🡪 membuat gambar ¼ kali lebih gede

Tapi otomatis karena di gedein gitu jadi ngeblur



Cara bikin pyramids

layer **=** img.copy()

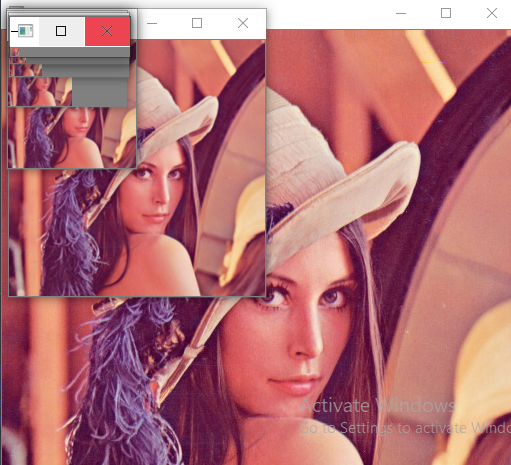
gp **=** [layer]

**for** i **in** range (6):

    layer **=** cv.pyrDown(layer)

    gp.append(layer)

    cv.imshow(str(i), layer



**Gaussian Pyramids**

Pyrup dan PyrDown itu adalah **Gaussian Pyramids**

ada juga yang namanya **Laplacian Pyramids**

Intinya

'''Jadi Laplacian Pyramids tuh

nge Substract antara Image yang udah di PyrUp (blur)

dan yang di PyrDown

Dengan resolusi frame yang sama

Jadilah kaya edge gitu

Bahasa bagusnya ada di word'''

Bahasa bagusnya :

A Level in Laplacian Pyramid is formed by the difference between that level in Gaussian Pyramid and expanded version of its upper level in Gaussian Pyramid

Hiyaa mamam, intinya sama aja

**Image Blending**

Merupakan penerapan dari image pyramids, liat gera penerapanya dimana

Intermezzo:

Ketika dipanggil apple.shape keluar

(512,512,3) 3 menurut urg adalah dimensi warna, kenapa??

Karena ketika kita panggil grayscale, 3 nya tuh ilang