Date / /

Support of Numpy makes the task more easier. So whatever operations you can do in Numpy, you can combine it with OpenCV.

(2) So opencv-python is an appropriate tool for fast prototyping of computer vision problems.

(3) A good knowledge on Numpy is must to write optimized codes in Open CV-python.

(4) Since opencv is a open source initiative.

Introduction to OpenCV:

(1) GUI features in OpenCV:

(a) Getting started with images:

(1) Here you will learn how to read an image, how to display it and how to save it back.

(2) Functions: cv2.imread(), cv2.imshow(), cv2.imwrite()

* Read an image

(1) Use the fn cv2.imread() to read an image.

(2) Full path of image should be given.

Second argument is a flag which specifies the way image should be read:-

- cv2.IMREAD_COLOR: loads a color image. Any transparency of image will be neglected. Default flag.
- cv2.IMREAD_GRAYSCALE: loads image in grayscale mode
- cv2.IMREAD_UNCHANGED: loads image as such including alpha channel.

Instead of these flags, you can simply pass integers 1, 0, -1.

Code impl.

```
import numpy as np
Import cv2
# Load an colour image in grayscale
img = cv2.imread('messi5.jpg', 0)
```

Note: If the img path is wrong, it won't throw any error, but print img will give you None

* DISPLAY AN IMAGE :

(1) Use the fn cv2.imshow() to display an image. The window automatically fits to the image size.

(2) First argument is a window name which is a string. second argument is our image. You can create as many windows as you wish, but with diff window names.

```
cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

cv2.waitKey(): It is a keyboard binding function. Its argument is the time in milliseconds. The fn waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a key stroke. It can also be set to detect specific key strokes like if key a is pressed.

Date / /

cv2.destroyAllWindows(): It simply destroys all the windows we created. If you want to destroy any specific window, use the fn cv2.destroyWindow() where you pass the exact window name as argument.

Note: There is a special case where you can already create a window and load image to it later. In that case, you can specify whether window is resizable or not. It is done with the fn cv2.namedWindow(). By default, flag is cv2.WINDOW_AUTOSIZE. But if you specify flag to be cv2.WINDOW_NORMAL, you can resize window.

WRITE AN IMAGE:

1. Use the fn cv2.imwrite() to save an image.
2. First argument is the file name, second argument is the image you want to save.

```
cv2.imwrite('messigray.png', img)
```

→ This will save the image in PNG format in the working directory.

Q) What does waitKey return?

1. The openCV documentation says that waitKey() returns an int. This is supposed to be the ASCII value of the key pressed. This suggests that waitKey returns a char and not an int.
2. waitKey(0) will display the window infinitely until any keypress (suitable for image display). If you use waitKey(0), you see a still image until you actually press something while for waitKey(1)

the fn will show a frame for 1 ms only.

Date / /

(3) cv2.waitKey(0) & 0xff means : In case of 64-bit systems the value of cv2.waitKey(0) is bitwise AND(&) with the 0xff hexadecimal constant (which is representation of binary string 11111111) that results in the last 8 bits of it.

Program loads an image in grayscale, displays it, save the image, if you press 's' and exit, or simply exit without saving if you press ESC key

```
import numpy as np
import cv2
img = cv2.imread('Aas.jpg', 0)
cv2.imshow('image', img)
k = cv2.waitkey(0)
if k == 27:
        cv2.destroyAllWindows()
elif k == ord('s'):
        cv2.imwrite('messigray.png', img)
        cv2.destroyAllWindows()
```

Note: If you are using a 64-bit machine, you will have to modify k = cv2.waitKey(0) line as follows:

```
k = cv2.waitKey(0) & 0xFF
```

Note: Color image loaded by OpenCV is in BGR mode. But Matplotlib displays in RGB mode. So color images will not be displayed correctly in Matplotlib if image is read with OpenCV.

Basic operations on Images:

Date / /

Goal: Access pixel values and modify them.
access image properties
Setting Region of image (ROI)
splitting and Merging images.

* Accessing and Modifying pixel values:

Let's load a color image:

```
import cv2
import numpy as np
img = cv2.imread("Aas.jpg")
```

You can access a pixel value by its row and col coordinates. For BGR image, it returns an array of Blue, Green, Red values. For grayscale image just corresponding intensity is required.

```
px = img[100,100] = img[100][100]
print px
o/p: [157, 166, 200]
```

* accessing only the blue pixel.

```
blue = img[100,100,0] , img[100][100][0]
print blue
o/p: 157
```

Extra Ugaar

⇒ imread() returns a 2D or 3D matrix based on the no. of color channels present in the image. For a binary or grey scale image, 2D array is sufficient. But for a colored image, you need 3D array.

cv2.IMREAD_UNCHANGED : It reads the image as is from the source. If the source image is an RGB, it loads the image into array with Reds, Green & Blue channels. If the src image is ARGB, it loads the image

with three color components along with the alpha or transparency channel.

Date / /

eg (1)
```
import cv2
img = cv2.imread('  ')
print('Image Dimensions:', img.shape)
```
O/p: Image dimensions : (400, 640, 3)

img.shape returns tuple representing (height, width, number of channels). Height of the image is 400 pixels, width is 640 pixels and there are 3 color channels in the image. For CV2.IMREAD_COLOR, transparency channels is ignored even if present.

eg (2)
```
import cv2
img = cv2.imread('  ', cv2.IMREAD_GRAYSCALE)
print('Image Dimensions:', img.shape)
```
O/p: Image dimensions : (400, 640)

Height of the image is 400 pixels, width is 640. Each element in the array represents grey scale value at the respective pixel.

eg (3)
```
import cv2
img = cv2.imread('  ', cv2.IMREAD_UNCHANGED)
print('Image Dimension:', img.shape)
```
O/p Image dimension : (400, 640, 4)

We have read all the four channels of the image, Namely Red, Green, Blue & Transparency.

Note: imread() decodes the image into a matrix with the color channels stored in the order of Blue, Green, Red and A (Transparency) respectively.

If (400, 640, 4) is the shape of the image, then

Date / /

Extra Tigaan Khatam!

- (:, :, 0) represents Blue channel
- (:, :, 1) rep " Green channel
- (:, :, 2) rep " Red channel
- (:, :, 3) " Transparency channel.

You can modify the pixel values the same way.

```
img[100, 100] = [255, 255, 255]
print(img[100][100])
```

O/P: [255, 255, 255]

Above mentioned method is normally used for selecting a region of array. For individual pixel access, Numpy array methods, array.item() and array.itemset() is considered to be better. But it always returns a scalar. So if you want to access all B, G, R values, you need to call array.item() separately for all.

accessing RED value

```
img.item(10, 10, 2)
```

O/P 59.

Modifying Red value

```
img.itemset((10, 10, 2), 100)
img.item(10, 10, 2)
```

100

Accessing image properties:

image properties include no. of rows, cols, channels, type of image data, no. of pixels etc.

Shape of image is accessed by img.shape.

Date / /

It returns a tuple of no. of rows, cols and channels (if image is color).

```
print (img.shape)
```

O/P : (342, 548, 3)

→ If image is grayscale, tuple returned contains only number of rows and cols. So it is a good method to check if loaded image is grayscale or color image.

→ Total no. of pixels is accessed by `img.size` :

```
print (img.size)
```

O/P : 562248

→ Image datatype is obtained by `img.dtype` :

```
print (img.dtype)
```

O/P : uint8

Note: img.dtype is very important while debugging because a large no. of errors in OpenCV-python code is caused by invalid datatypes.

Image ROI :

Sometimes you will have to play with certain region of images. For eye detection in images, first perform face detection over the image until the face is found, then search within the face region for eyes. This approach improves accuracy (because eyes are always on faces) & performance (because we search for a small area).

Date / /

ROI is again obtained using Numpy indexing. Here I am selecting a ball & copying it to another region in the image.

```
import cv2
import numpy as np
img = cv2.imread("Aas.jpg")
roi = img[0:1000, 0:1000]
img[1000:2000, 1000:2000] = roi
cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.namedWindow('roi', cv2.WINDOW_NORMAL)
cv2.imshow('image', img)
cv2.imshow("roi", roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

* Splitting and Merging image Channels

The B, G, R channels of an image can be split into their individual planes when needed. Then the individual channels can be merged back together to form a BGR image again.

```
b, g, r = cv2.split(img)
img = cv2.merge((b, g, r))
```

Or

```
b = img[:, :, 0]
```

Suppose you want to make all the red pixels to zero, you need not split like and put it equal to zero. You can simply use Numpy indexing which is faster.

```
img[:, :, 2] = 0
```

Numpy indexing is much more efficient and should be used if possible.

Date

Making borders for images (Padding)

If you want to create a border around the image, something like a photo frame, you can use cv2.copyMakeBorder() function.

Arguments:

1. src – I/p image.
2. top, bottom, left, right : border width in no. of pixels in corresponding directions.
3. borderType : Flag defining what kind of border to be added. It can be following types –
   1. → CV2.BORDER_CONSTANT : Adds a constant colored bored. the value should be given as next argu.
   2. CV2.BORDER_REFLECT : Border will be mirror reflection of the border elements.
   3. CV2.BORDER_REFLECT_101 or CV2.BORDER_DEFAULT Same as above but with a slight change.
   4. CV2.BORDER_REPLICATE : Last element is replicated throughout.
   5. CV2.BORDER_WRAP : ~~Can't~~
4. value : Color of border if border type is CV2.BORDER_CONSTANT.

MATHEMATICAL TOOLS IN OPENCV.

Getting started with image Processing using Python:
(Medium).

OpenCV and Haar cascade ?

Open CV or Open Source Computer Vision Library is an open source computer vision & machine learning library. Almost most popular with Python, it interfaces quite well with C++, Java, and ~~Matplotlib~~. MatLab. OpenCV is native written in C++ and is widely used with computer vision related applications running on variety of systems like windows, Linux, Android etc. OpenCV uses HaaR cascade files for objects detection. HaaR cascade files are nothing but models trained by OpenCV to detect an object of

Date / /

Cascade Classifier:

Goal: How the Haar cascade object detection works?

Theory / Basics:

Object detection using Haar feature-based cascade classifiers is an effective objects detection method. It is a machine learning based approach where a cascade function is trained from a lots a lot of positive and negative images. It is then used to detect objects in other images.

Cascade classifier : Cascading : it is a particular case of ensemble learning, based on the concatenation of several classifiers, using all info collected from the output from a given classifier as additional info for the next classifier in the cascade.

Here we will work with face detection. Initially, the algo needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used.

Date / /

(1) A face detection algorithm, a system is designed by giving input as some faces and non faces & training a classifier or something that identifies a face. So we train something using faces & non faces and once the training is done that the data, we have got we would be able to detects any faces from an image.

(2) For example, we show some ~~faces~~ images of a face to an alien who has no previous knowledge of what the human face is, we show some two or 1000 of human faces and tell it that it is a human face & we also show some hundreds of ~~three~~ thousands of non face and tell it that these are not faces. So once an alien is trained to identify these features, whenever we show any new image later on, it will be able to classify as face or non-face.

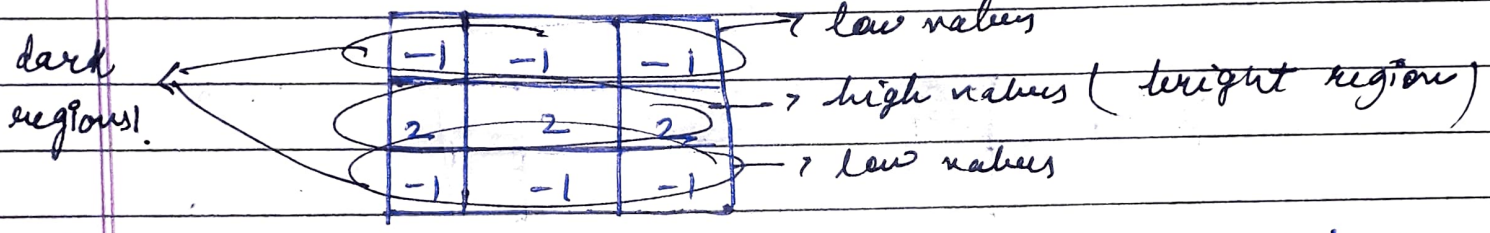
(3) Exactly, we are tying to do is train the computer to understand what a face is and what a non face is. Once the computer is trained, it will extract certain features and everything will be stored in a file. All we do is we take that file, if we get any new input image check the features from that file and apply it to the image. So if it passes through all the stages of feature comparision, you say that it is a face else it is not a face.

(4) So we have already a trained data with all the features that have been trained & now we do is in any system, we just have

Date / /

the train data and using that data we just start to classify the given name image as a face or have non face just by refering to the data that we already.

Basic intro to edge detection:

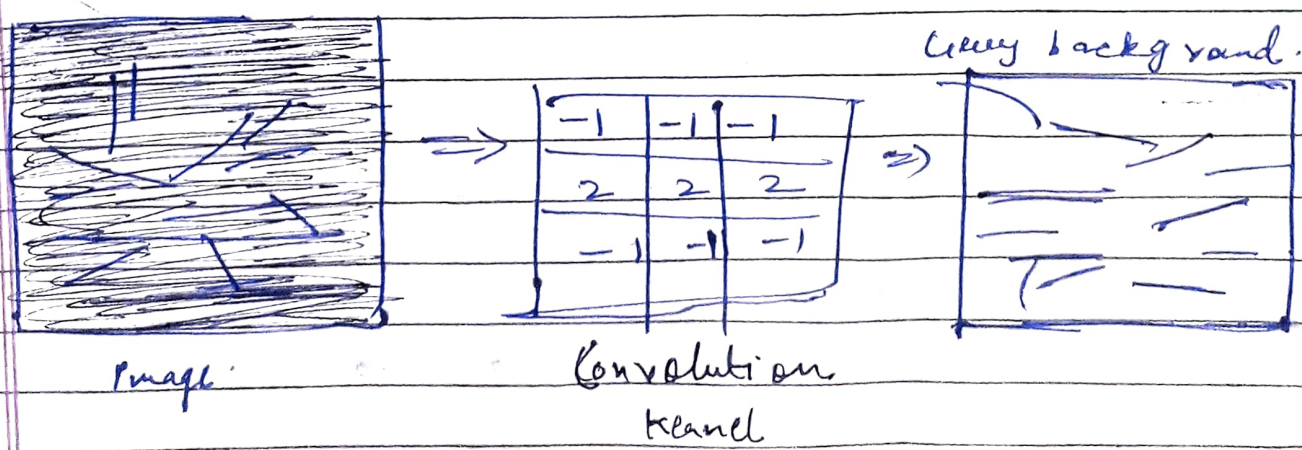


(1) we have a pattern like this. It has some low values and high values.

(2) So basically high values vaala part is the bright area and dark area regions above & below it.

(3) So I am trying to find a single horizontal high value line in this image. So I create a Kernel that is similar to what I would like to extract from the image. So what will be extracted from that o/p image will be the horizontal lines. So I apply this kernel all over the image,

(4) the o/p image only has high values only at the places wherever this pattern matches with the image.

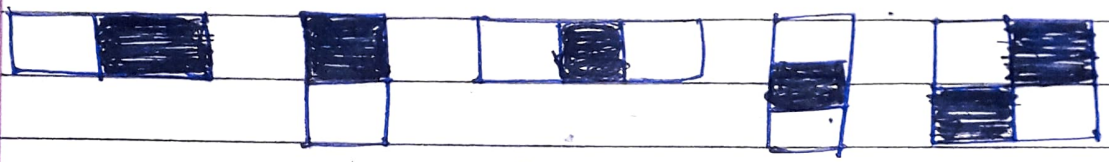


Date / /

Haar features:

(1) Haar features are similar to these convolutional kernels which are used to detect the presence of that feature in the given image.

Types of Haar features:



Type 1 Type 2 Type 3 Type 4 Type 5

Significance: A black region is replaced by -1 & white region is replaced by $+1$. Exactly like convolutional kernel with 1 row & 2 cols Right col is -1 & left col is $+1$.

(2) Each feature results in a single value which is calculated by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle.

(3) What we understand from this is all that haar features have some sort of resemblance to some facial features or to some characteristics of each face.

(4) So viola jones uses (24 × 24) sub window from an image and it calculates these features all over this image.

Date / /

(5) If we consider all possible parameters of the haar features like position, scale & type we end up calculating about 160,000+ features in this window.

(6) What we will do is eliminate the redundant features or the features which are not useful. and select only those features which are very very useful for us. This is done by Adaboost.

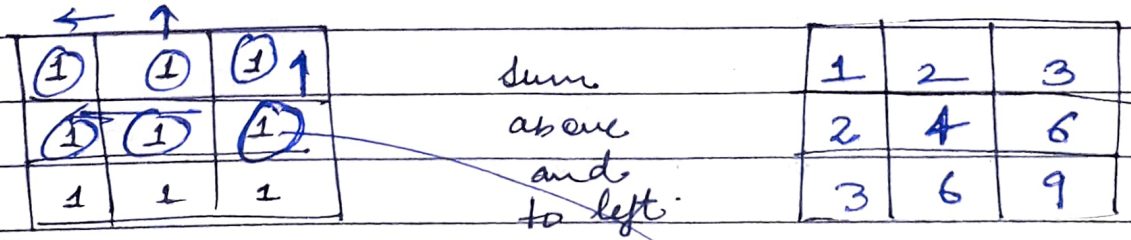
(7) Adaboost eliminates the all the redundant features, the features that we don't need,

Integral Image

: we don't need to sum up all the pixels, rather we use the corner value of this batch & do simple calculations.

What are simple calculations :?

In an Integral image the value at pixel (x, y) is the sum of pixels above & to left of (x, y)



(1) Suppose we need to calculate this value at Integral Image, so sum up everything from the top to the left of that pixel.

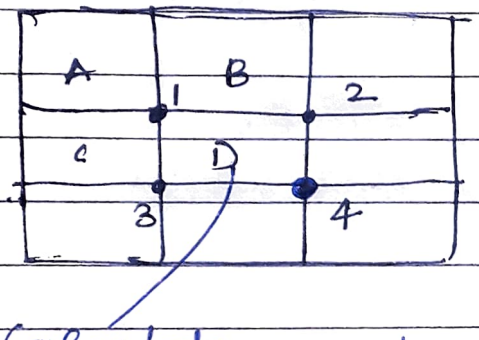
(2) So Integral Image means ~~more up~~ to get the new pixel value, just sum up all the pixel values falling in left & top region.

Advantage :

Date / /

→ Integral image allows for the calculation of sum of all pixels inside any given rectangle using only 4 values at the corners of the rectangle.

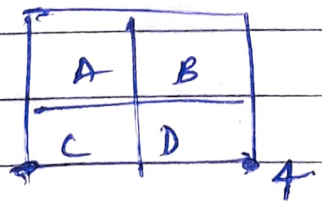
Integral Image :

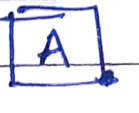


(1) Calculate no of pixels in D batch in the Integral image,

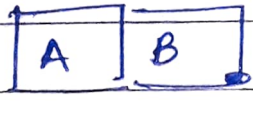
(2) Add the pixels value of 1 and 4 and then subtract the sum of pixels values of the other diagonal i.e 2 & 3.

$4 = A + B + C + D$



$1 = A =$  = Sum of pixels of region A

2 = Sum of all pixels in the region:



$= A + B$

$3 = A + C =$ 

③ So if you take the sum of its diagonals Date / /

i.e. (2+3) & subtract it from the sum of other diagonal (1+4) of in the integral image, you end up calculating the sum of all pixels in the region D.

$$D = 1+4-(2+3)$$
$$A+(A+B+C+D)-(A+C+A+B)$$
$$= D. \quad \underline{LHS = RHS}$$

Ada boost!

① Used to eliminate the redundant features.

② Relevance or irrelevance is determined by adaboost and it will select only few features which are relevant to us.

③ Adaboost is a machine learning algo which helps in finding only the best features among all these 160,000+ features. After these features are found, a weighted combination of all these features is used in evaluating & deciding any given window has a face or not. Each of the selected features are considered okay to be included if they can atleast perform better than random guessing.

Weak-Classifier: It means a good feature or a relevant feature. A weak classifier is something which atleast performs better than random guessing which means that if we give 100 faces to it, it will, atleast be able to detect more than 50 faces.

Date / /

(4) So weak classifier is just a relevant feature that is extracted by adaboost and we apply that relevant feature and find corresponding ∝ weight for that and we continually combine all those relevant features with their weights and form a strong classifier / strong detector.

(5) O/p of the weak classifier is either 1 or 0. 1 is when it has performed well & identified the feature when it is applied on that image.

$$f(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \cdots$$

↑ strong classifier

↑ weak classifier