Lab2

<https://www.geeksforgeeks.org/saving-operated-video-from-a-webcam-using-opencv/>

Lab1

Questions what is colour space

How colour is conveerted to black and white

Use of gaussian noise

mosaic

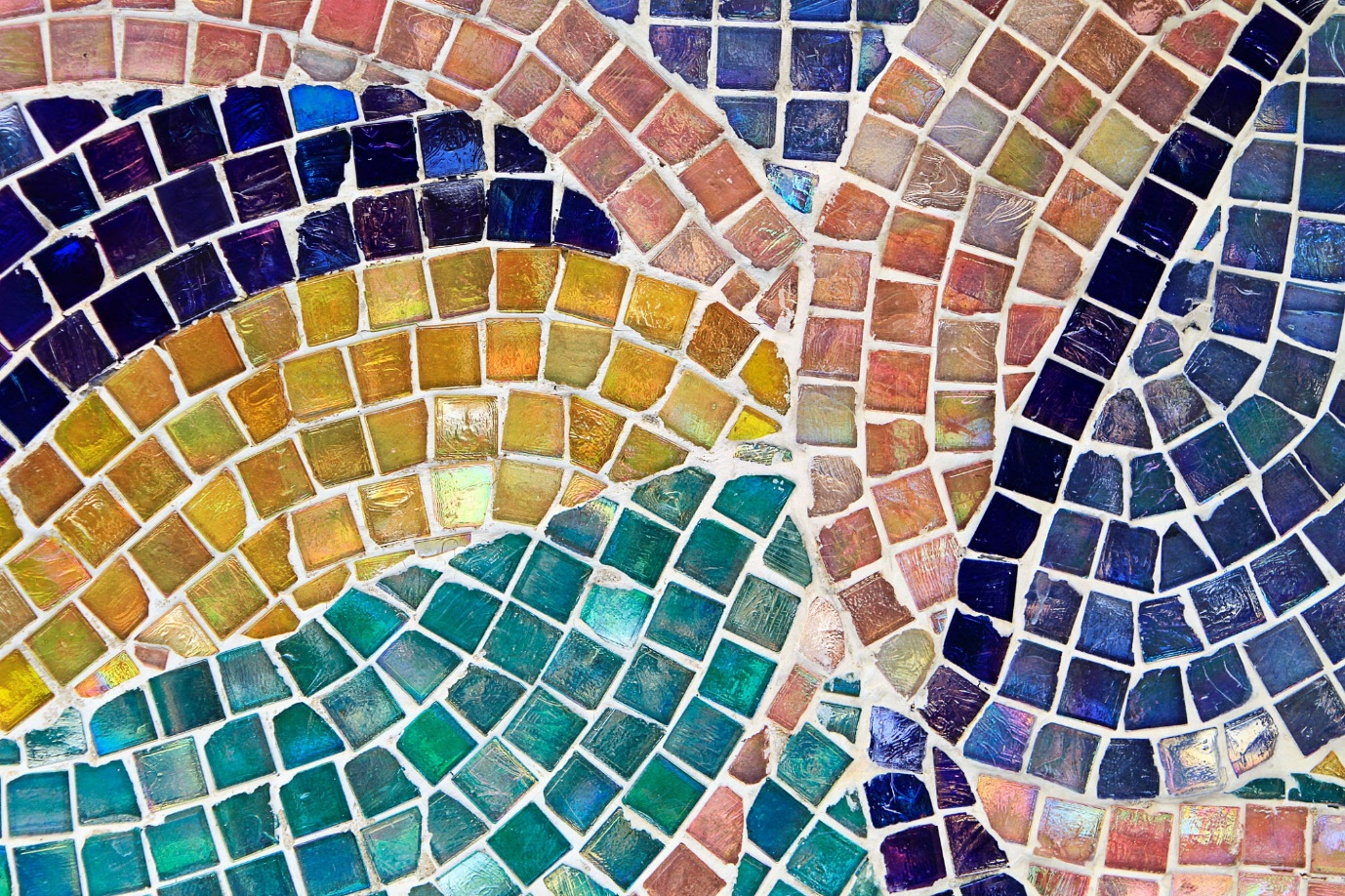
/mə(ʊ)ˈzeɪɪk/

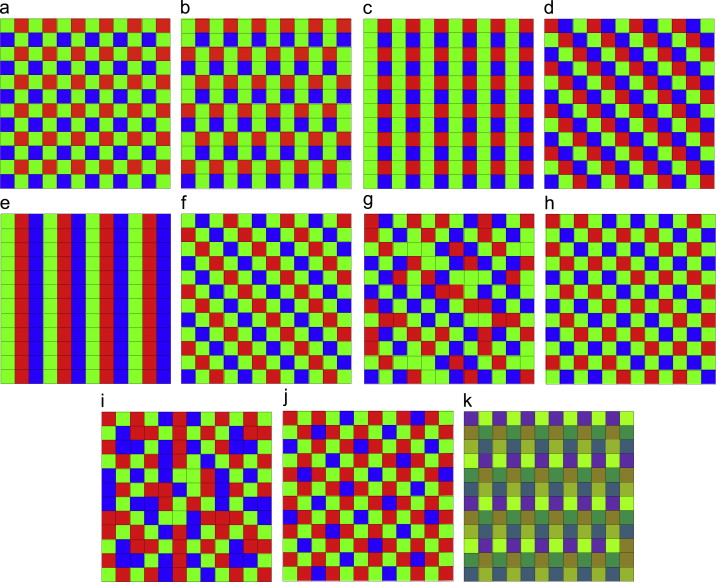
[Learn to pronounce](https://www.google.com/search?rlz=1C1ASUM_enIN808IN808&q=how+to+pronounce+mosaic&stick=H4sIAAAAAAAAAOMIfcRowS3w8sc9YSn9SWtOXmPU5OINKMrPK81LzkwsyczPExLiYglJLcoV4pLi4GLLzS9OzEy2YlFiSs3jWcQqnpFfrlCSr1AA1JEP1JKqAFEAANfm2Y5XAAAA&pron_lang=en&pron_country=gb&sa=X&ved=2ahUKEwiB89eNjYDoAhXF7XMBHZ1OAqYQ3eEDMAB6BAgEEAg)

*noun*

1. 1.

a picture or pattern produced by arranging together small pieces of stone, tile, glass, etc.





I understand that color filter is filter

A **color filter array** (CFA) is a mosaic of **color filters** (generally red, green and blue) that overlays(cover surface with coating) the pixels comprising the sensor. The **color filters** limit the intensity of light being recorded at the pixel to be associated with the wavelengths transmitted by that **color**.

A Bayer filter mosaic is a color filter array for arranging RGB color filters on a square grid of photosensors. Its particular arrangement of color filters is used in most single-chip digital image sensors used in digital cameras, camcorders, and scanners to create a color image

**Colour space –**

Color model

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components

**How gray scale image is recorded?**

In [digital photography](https://en.wikipedia.org/wiki/Digital_photography), [computer-generated imagery](https://en.wikipedia.org/wiki/Computer-generated_imagery), and [colorimetry](https://en.wikipedia.org/wiki/Colorimetry), a grayscale or greyscale [image](https://en.wikipedia.org/wiki/Image) is one in which the value of **each**[**pixel**](https://en.wikipedia.org/wiki/Pixel)**is a single**[**sample**](https://en.wikipedia.org/wiki/Sample_(signal))**representing only an *amount* of**[**light**](https://en.wikipedia.org/wiki/Light)**, that is, it carries only**[**intensity**](https://en.wikipedia.org/wiki/Luminous_intensity)**information.** Grayscale images, a kind of [black-and-white](https://en.wikipedia.org/wiki/Black-and-white) or gray [monochrome](https://en.wikipedia.org/wiki/Monochrome), are composed exclusively of [shades of gray](https://en.wikipedia.org/wiki/Shades_of_gray). The [contrast](https://en.wikipedia.org/wiki/Contrast_(vision)) ranges from [black](https://en.wikipedia.org/wiki/Black) at the weakest [intensity](https://en.wikipedia.org/wiki/Lightness) to [white](https://en.wikipedia.org/wiki/White) at the strongest.[[1]](https://en.wikipedia.org/wiki/Grayscale#cite_note-1)

**What is colour image**?

**Luminance** indicates the degree of brightness over an area

For visually acceptable results, it is necessary (and almost sufficient) to provide three [samples](https://en.wikipedia.org/wiki/Sample_(signal)) (*color channels*) for each pixels, which are interpreted as coordinates in some [color space](https://en.wikipedia.org/wiki/Color_space). The [RGB color space](https://en.wikipedia.org/wiki/RGB_color_space) is commonly used in [computer displays](https://en.wikipedia.org/wiki/Computer_displays), but other spaces such as [YCbCr](https://en.wikipedia.org/wiki/YCbCr), [HSV](https://en.wikipedia.org/wiki/HSV_color_space), and are often used in other contexts. A color image has three values (or [channels](https://en.wikipedia.org/wiki/Image_channels)) per pixel and they measure the intensity and chrominance(the colorimetric difference between a given colour in a television picture and a standard colour of equal luminance) of light. The actual information stored in the digital image data is the brightness information in each spectral band.

**RGB is a colour model.**

**Difference between colour model and colour space?**

A **color model** is a method of describing a **color**. For example with Red, Green and Blue (RGB) elements or with Cyan, Magenta, Yellow and Black (CMYK). A **color space** is the set of **colors** which can be displayed or reproduced **in a** medium (whether stored, printed or displayed) – stack overflow

**Lab3:**

## Resizing and resampling images

When you resize and resample an image, you change the amount of data in that file. To resample your image, ensure that Resample is selected at the bottom of the Image Size dialog box. Resample is on by default.

Resampling changes the total number of pixels in the image, which are displayed as Width and Height in pixels in the Image Size dialog box. When you increase the number of pixels in this part of the dialog box (upsampling), the application adds data to the image. When you decrease the number of pixels (downsampling), the application removes data. Whenever data is removed from or added to the image, the image quality degrades to some extent. Removal of data from an image is typically preferable to the addition of data. **That's because upsampling requires that Photoshop guess which pixels to add. This procedure is more complex than guessing which pixels to remove when you downsample.** You get the best results working with images you bring into Photoshop in the proper resolution for the output you want. You could get the results you need by resizing your image without resampling. However, if you resample your images, do so only once.

codec

A **codec** is a device or software that is used to compress or decompress a digital media **file**, such as a video or song. The “**codec**” can be dividing into 2 parts: encode and decode. The encoder performs the compression (encoding) function and the decoder performs the decompression (decoding) function.

Lab4: Edge Detection and Region Of Interest Using OpenCV and Python

Many common image operations are performed using **Region of Interest** in **OpenCV**. A **ROI** allows us to operate on a rectangular subset of the image. The typical series of steps to use **ROI** is: create a **ROI** on the image, perform the operation you want on this subregion of the image, reset back the **ROI**.

Tkinter

Frame button are basically used

## What are edges

We can also say that sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.

### **Types of edges**

Generally edges are of three types:

* Horizontal edges
* Vertical Edges
* Diagonal Edges

### **Why detect edges**

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Here are some of the masks for edge detection that we will discuss in the upcoming tutorials.

* Prewitt Operator
* Sobel Operator
* Robinson Compass Masks
* Krisch Compass Masks
* Laplacian Operator.

Above mentioned all the filters are Linear filters or smoothing filters.

### **Prewitt Operator**

Prewitt operator is used for detecting edges horizontally and vertically.

### **Sobel Operator**

The sobel operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection. It also calculates edges in both horizontal and vertical direction.

### **Robinson Compass Masks**

This operator is also known as direction mask. In this operator we take one mask and rotate it in all the 8 compass major directions to calculate edges of each direction.

### **Kirsch Compass Masks**

Kirsch Compass Mask is also a derivative mask which is used for finding edges. Kirsch mask is also used for calculating edges in all the directions.

### **Laplacian Operator**

Laplacian Operator is also a derivative operator which is used to find edges in an image. Laplacian is a second order derivative mask. It can be further divided into positive laplacian and negative laplacian.

All these masks find edges. Some find horizontally and vertically, some find in one direction only and some find in all the directions. The next concept that comes after this is sharpening which can be done once the edges are extracted from the image

## Sharpening

Sharpening is opposite to the blurring. In blurring, we reduce the edge content and in Sharpening, we increase the edge content. So in order to increase the edge content in an image, we have to find edges first.

Edges can be find by one of the any method described above by using any operator. After finding edges, we will add those edges on an image and thus the image would have more edges, and it would look sharpen.

This is one way of sharpening an image.

The sharpen image is shown below.

Where xx is the image we want to infer from the given image yy, AA is the degradation operator applied on xx and ww is white additive noise.

1. Denoising  
   For denoising only noise is added hence A=IA=I and we left with estimating xx from a noisy measurements.
2. Deblurring  
   In that case AA is a matrix form of some Low Pass Filter (Circulant Square Matrix) which applies a blur on the image. We try to infer xx from a blurred and noisy version of it given by yy.
3. Super Resolution  
   In this case take AA from (2) and remove few rows of it. Namely we get only some of the data in xx and the data is both blurred and noisy and we try to infer the whole data of xx given yy.

**Lab5:**

In this code,

The function **waitKey**() waits for a key event for a "delay" (here, 30 milliseconds)

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

break

The waitKey(0) function returns **-1** when no input is made whatsoever. As soon the event occurs *i.e. a Button is pressed* it returns a **32-bit integer**.

The 0xFF in this scenario is representing binary **11111111** a **8 bit** binary, since we only require 8 bits to represent a character we AND waitKey(0) to 0xFF. As a result, an integer is obtained below 255.

ord(char) returns the ASCII value of the character which would be again maximum 255.

Hence by comparing the integer to the ord(char) value, we can check for a key pressed event and break the loop.

**Image rotation**

### Rotation//must read

Rotation of an image for an angle \theta is achieved by the transformation matrix of the form

M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta   \end{bmatrix}

But OpenCV provides scaled rotation with adjustable center of rotation so that you can rotate at any location you prefer. Modified transformation matrix is given by

\begin{bmatrix} \alpha &  \beta & (1- \alpha )  \cdot center.x -  \beta \cdot center.y \\ - \beta &  \alpha &  \beta \cdot center.x + (1- \alpha )  \cdot center.y \end{bmatrix}

where:

\begin{array}{l} \alpha =  scale \cdot \cos \theta , \\ \beta =  scale \cdot \sin \theta \end{array}

To find this transformation matrix, OpenCV provides a function, **cv2.getRotationMatrix2D**. Check below example which rotates the image by 90 degree with respect to center without any scaling.

Lab 6

<https://docs.opencv.org/3.4/d5/d69/tutorial_py_non_local_means.html>

<https://www.researchgate.net/post/How_do_we_know_what_kind_of_noise_eg_Gaussian_salt_and_pepper_etc_is_present_in_an_image>

#lab 7

**Histogram equalization** is a method in image processing of contrast adjustment using the image’s histogram.

This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark

<https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html> best link it expalins how it happens

<https://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html?highlight=equalizehist#equalizehist>

it explains maths concept



See graph

<https://www.youtube.com/watch?v=WuVyG4pg9xQ> link video

mport cv2

# import Numpy

import numpy as np

# read a image using imread

img = cv2.imread(\'F:\\do\_nawab.png\', 0)

# creating a Histograms Equalization

# of a image using cv2.equalizeHist()

equ = cv2.equalizeHist(img)

# stacking images side-by-side

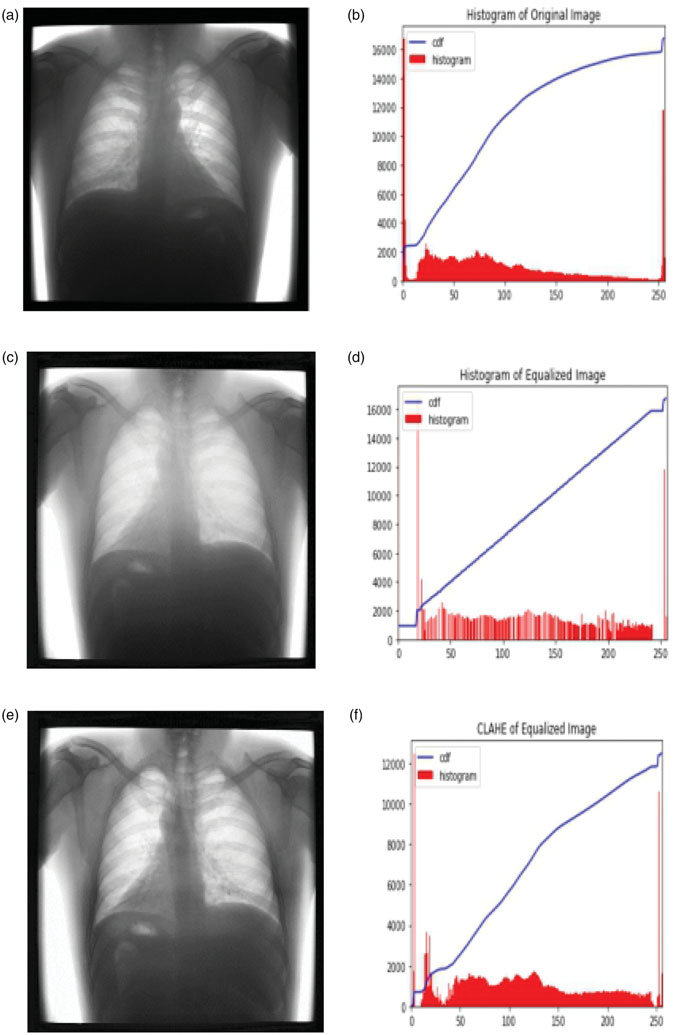
res = np.hstack((img, equ))

# show image input vs output

cv2.imshow(\'image\', res)

cv2.waitKey(0)

cv2.destroyAllWindows()



he method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of [bone](https://en.wikipedia.org/wiki/Bone) structure in [x-ray](https://en.wikipedia.org/wiki/X-ray) images, and to better detail in [photographs](https://en.wikipedia.org/wiki/Photographs) that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an [invertible](https://en.wikipedia.org/wiki/Invertible) [operator](https://en.wikipedia.org/wiki/Operator_(mathematics)). So in theory, if the histogram equalization [function](https://en.wikipedia.org/wiki/Function_(mathematics)) is known, then the original histogram can be recovered. The calculation is not [computationally](https://en.wikipedia.org/wiki/Computation) intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background [noise](https://en.wikipedia.org/wiki/Signal_noise), while decreasing the usable [signal](https://en.wikipedia.org/wiki/Signal_(information_theory)).

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histograms/py_histogram_equalization/py_histogram_equalization.html>

<https://en.wikipedia.org/wiki/Histogram_equalization>

<https://www.geeksforgeeks.org/histograms-equalization-opencv/>

Lab 10 – geeks for geeks

**It is normally performed on binary images.**

### 1. Erosion

The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). So what it does? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

Opening is just another name of **erosion followed by dilation**. It is useful in removing noise, as we explained above. Here we use the function, [**cv.morphologyEx()**](https://docs.opencv.org/trunk/d4/d86/group__imgproc__filter.html#ga67493776e3ad1a3df63883829375201f)

### 6. Top Hat

It is the difference between input image and Opening of the image. Below example is done for a 9x9 kernel.

<https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/>

<https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html>

**by me-what opening will do is it removes small dots and enhances large objects but top hat will do opposite of it so large will go**

The top-hat filter (a.k.a. white top-hat filter) is used to **enhance bright objects of interest in a dark background**. The black-hat operation (a.k.a. bottom-hat filter or black top-hat filter) is used to do the opposite, i.e., enhance dark objects of interest in a bright background. Top hat filtering will simply remains noise

See the example at [Top-hat filtering - MATLAB imtophat - MathWorks United Kingdom](https://uk.mathworks.com/help/images/ref/imtophat.html) which uses the top-hat filter to correct uneven illumination.

Essentially, the filters suppress large regions while keeping the small ones, as specified by the size of the structuring element. This could be used simply for visual purposes or as a pre-processing step in an automated image-analysis algorithm.

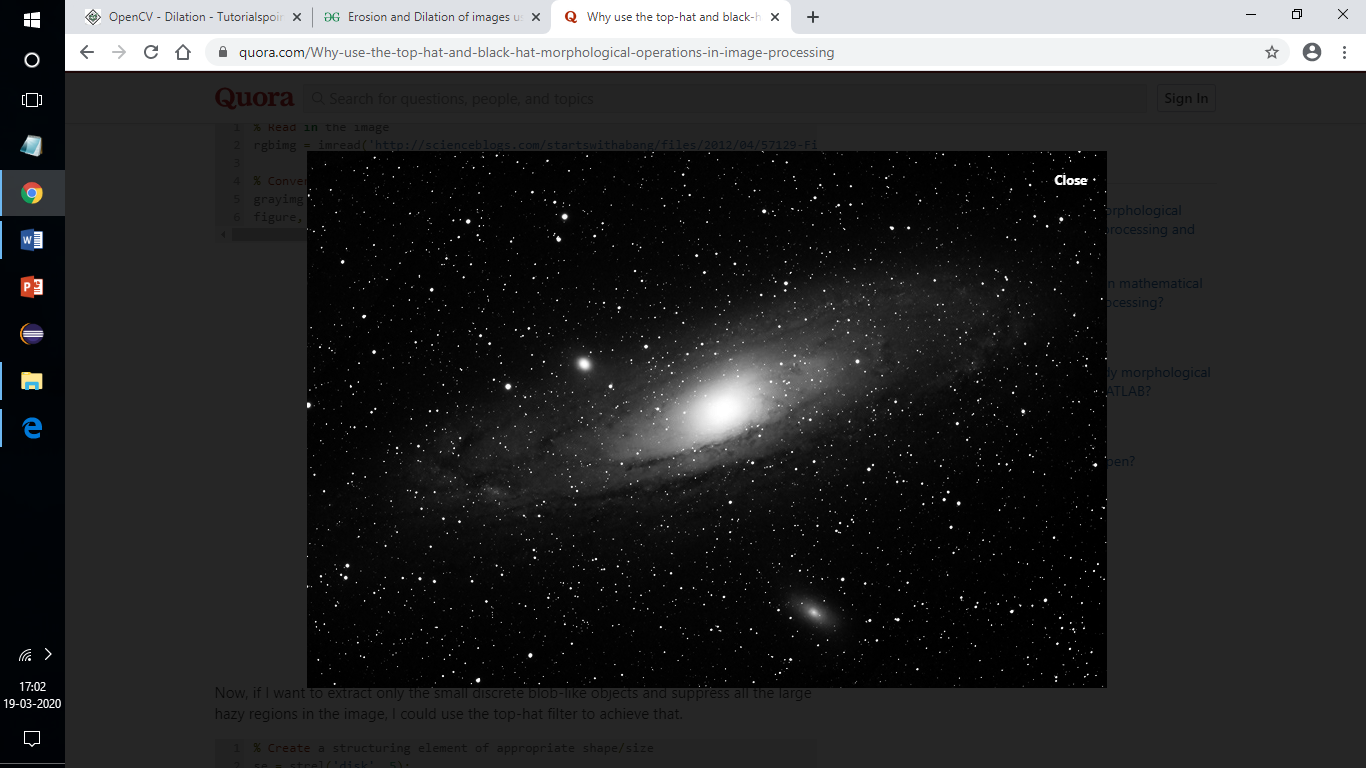
Let’s say I am working with an image of the Andromeda galaxy.

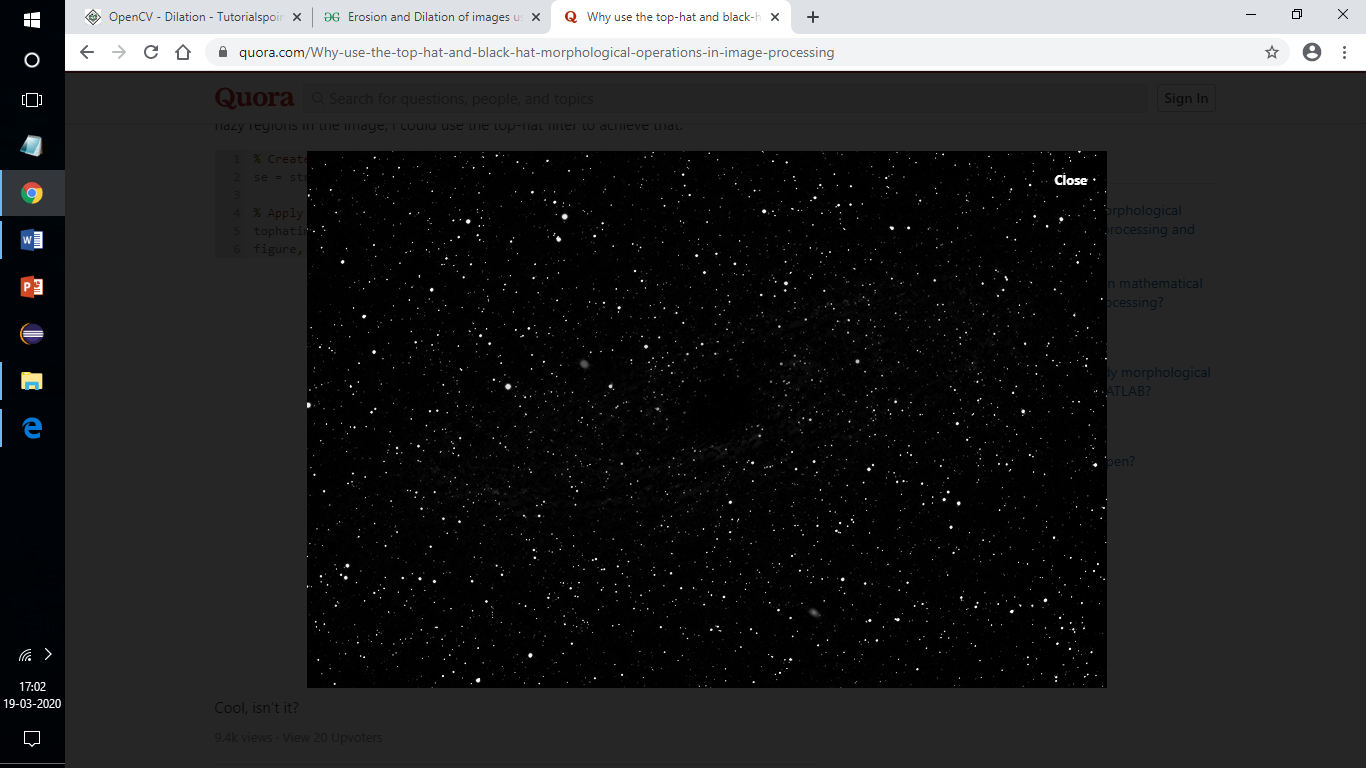
1. % Read **in** the image
2. rgbimg = imread('http://scienceblogs.com/startswithabang/files/2012/04/57129-File:M31bobo.jpeg');
4. % Convert to grayscale **and** display
5. grayimg = rgb2gray(rgbimg);
6. figure, imshow(grayimg)

Now, if I want to extract only the small discrete blob-like objects and suppress all the large hazy regions in the image, I could use the top-hat filter to achieve that.

1. % Create a structuring element of appropriate shape/size
2. se = strel('disk', 5);
4. % Apply top-hat **and** display
5. tophatimg = imtophat(grayimg, se);
6. figure, imshow(tophatimg);

Cool, isn’t it?





Uses of dialation clarifies doubt

1. Erosion:
   * It is useful for removing small white noises.
   * Used to detach two connected objects etc.
2. Dilation:
   * In cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won’t come back, but our object area increases.
   * It is also useful in joining broken parts of an object.

Lab 14:

Features circles corners,ellipses etc

Descrptor define features no matter what is orientation

Value will give you the good match

I had the same problem. It seems that SIRF and SURF are [no longer available in opencv > 3.4.2.16](https://github.com/DynaSlum/satsense/issues/13). I chose an older opencv-python and opencv-contrib-python versions and solved this problem. Here is the [history version](https://pypi.org/project/opencv-python/#history) about opencv-python, and I use the following code :

pip install opencv-python==3.4.2.16

pip install opencv-contrib-python==3.4.2.16

lab 11

<https://docs.opencv.org/master/de/d3c/tutorial_out_of_focus_deblur_filter.html>

Here is a mathematical model of the image degradation in frequency domain representation:

S=H⋅U+N

where S is a spectrum of blurred (degraded) image, U is a spectrum of original true (undegraded) image, H is a frequency response of point spread function (PSF), N is a spectrum of additive noise.

The circular PSF is a good approximation of out-of-focus distortion. Such a PSF is specified by only one parameter - radius R. Circular PSF is used in this work.

**Frequency domain** analysis refers to analysis where a signal is studied with respect to **frequency**, rather than with respect to time. The data being studied is plotted with the **frequency** on the x axis and the amplitude on the y axis; this shows how the signal's energy is **distributed** as a function of **frequency**

<https://www.youtube.com/watch?v=YhLF95crTWs>

lab 9

please make more clear your question

you want compute the euclidean distance between two RGB images for similarity measure or for change detection

for similarity measuring (matching), you will compute the euclidean distance for the whole two images as given by Gorbachev or Till above

for change detection you have to compute pixel wise euclidean distance (the euclidean distance for each same coordinate pixel in the two images) this is given by:

D(x,y)= sqrt((I1G-I2G)^2+(I1B-I2B)^2+(I1R-I2R)^2)

where I1, I2 are RGB images, I1G is the green plan of the I1 RGB image, I1B is the blue plan and I1R the red pla

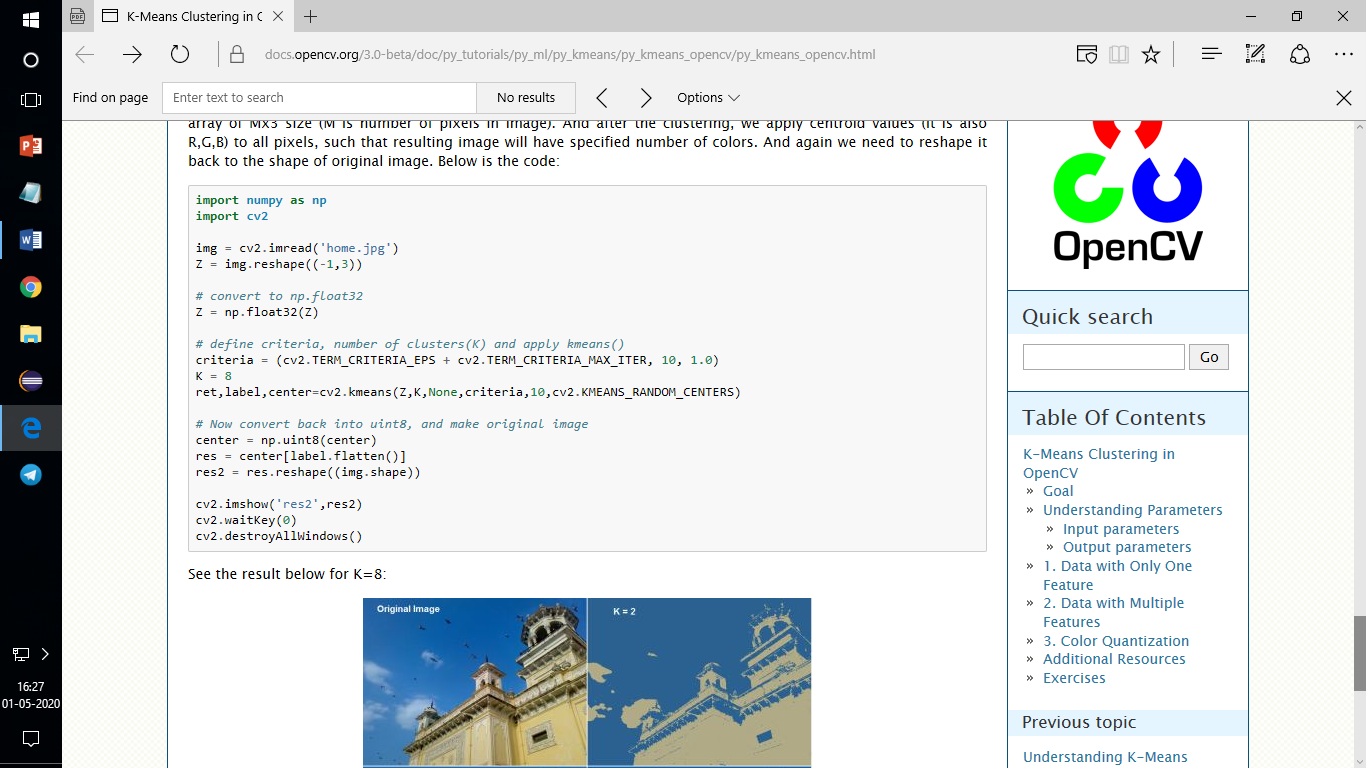
center = np.uint8(center)

res = center[label.flatten()]

res2 = res.reshape((img.shape))

labels will contain indexes for selected colours for each pixel . for example 0th index will tell that pixel at 0 should contain colour of centre[label[0]] .

<https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html>



lab 9

2 stack overflow

You can get multiple elements by index using an np.ndarray. In your case label.flatten() is an array of the indices of the the array center that you need. For example if center is an array of k-means centroids, then this operation selects some of those centers.

A minimal example:

a = np.array([-2, 1, 5, 3, 8, 5, 6])

b = [1, 2, 5]

print(list(a[b])) # Prints [1, 5, 5]

by me - centre array will contains all centres points but we need to assign these centres to our resultant image so we need to index one by one so for that we use labels labels contains information centre need to be assign for each pixel

lab 8

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze



<https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/>

for questions

1. What is Image Segmentation?
2. Why do we Need Image Segmentation?
3. The Different Types of Image Segmentation
4. Region-Based Segmentation
5. Edge Detection Segmentation
6. Segmentation based on Clustering
7. Facebook’s Mask R-CNN Framework
8. Summary of Image Segmentation Techniques

By dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works.

**Bounding box itself is not sufficient some time we need size of image then we need to go for image segmentation**

Haar cascade classifier works with a sliding window approach. If you look at the cascade files you can see a size parameter which usually a pretty small value like 20 20. This is the smallest window that cascade can detect. So by applying a sliding window approach, you slide a window through out the picture than you resize it and search again until you can not resize it further. So with every iteration haar's cascaded classifier true outputs are stored. So when this window is slided in picture resized and slided again; it actually detects many many false positives. You can check what it detects by giving **minNeighbors 0**. So an example here :



So there are a lot of face detection because of resizing the sliding window and a lot of false positives too. So to eliminate false positives and get the proper face rectangle out of detections, neighborhood approach is applied. It is like if it is in neighborhood of other rectangles than it is ok, you can pass it further. So this number determines the how much neighborhood is required to pass it as a face rectangle. In the same image when it is **1** :



So by increasing this number you can eliminate false positives but be careful, by increasing it you can also lose true positives too. When it is **3** a perfect result :



[share](https://stackoverflow.com/a/22250382)follow

<https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php>

**lab 8**

<https://www.tutorialspoint.com/detection-of-a-specific-color-blue-here-using-opencv-with-python>