3. Median Blurring:

In this technique, median of the kernel space is used to replace the value of every pixel( which is the center of that kernel ). As we perform operation on the center of the kernel space hence the width and height of the space is always positive odd number.

In median blurring, the central pixel is always replaced by some pixel value which is present in the image. This property is not present in other techniques like Averaging and Gaussian blurring.

Median blurring supports inplace operation and is effective in smoothing salt and pepper noise.

**cv2.medianBlur( source, ksize** **)** function in OpenCV is used to smooth the image using median blurring. The function has two arguments:

1. source: it signifies the source image on which smoothing has to be done.
2. ksize: it signifies the size of kernel matrix. The value of ksize must be odd and greater than 1 always.

Python code:

# Importing cv2

import cv2

#read input image

img = cv2.imread('cat.jpg');

#increasing ksize will make the image smoother

blur1 = cv2.medianBlur(img,5);

blur2 = cv2.medianBlur(img,7);

cv2.imshow('Original',img);

cv2.imshow('blurred image 1',blur1);

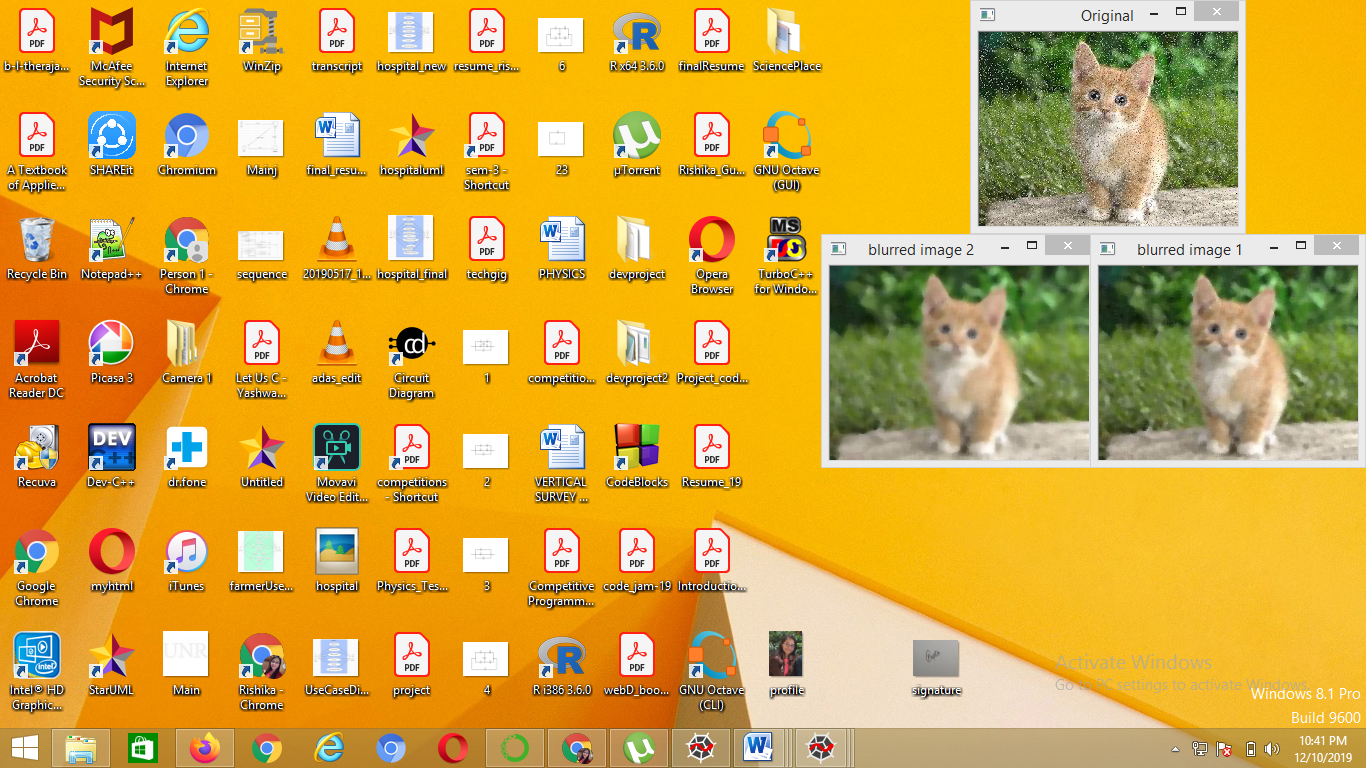
cv2.imshow('blurred image 2',blur2);

#wait for 10 sec

cv2.waitKey(10000);

cv2.destroyAllWindows();

Output:



4. Bilateral Filtering:

As we have observed in all techniques of smoothing that edges and noises are smoothed using some function. But the main purpose of smoothing is to remove noise and not to make the whole image blurred. Hence we need some technique that won’t make the edges blur.

Bilateral filtering ensures to not to affect the edges while smoothing the image by considering the intensities of neighboring pixels. It also uses Gaussian function to prepare kernel matrix but it considers only that neighboring pixels in the kernel which have similar intensity to the central pixel. More precisely bilateral filtering is based on Gaussian function with intensity difference.

As on edges the neighboring pixels will have a drastic difference of intensity as compared to central pixel, hence the edges are preserved from smoothing.

But bilateral filtering is comparatively slow than other filters and it does not support inplace operation.

cv2.bilateralFilter(source, ksize, sigmaColor, sigmaSpace) function in openCV is used to perform bilateral filtering. The function has four arguments:

1. source: it signifies the source image on which smoothing has to be done.
2. ksize: it signifies the size of kernel matrix. The value of ksize must be odd and greater than 1 always. When ksize<=0 ksize is proportional to sigmaSpace.
3. sigmaColor: it signifies the filter in color space. A larger value of the parameter means that farther colors within the pixel neighborhood will be mixed together resulting in larger areas of semi-equal color.
4. sigmaSpace: it signifies the filter in coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough.

Python code:

# Importing cv2

import cv2

#read input image

img = cv2.imread('door.jpg');

blur1 = cv2.bilateralFilter(img,5,75,100); #ksize = 5

blur2 = cv2.bilateralFilter(img,-1,100,120); #ksize<=0 hence kernel size is calculated using sigmaSace

cv2.imshow('Original',img);

cv2.imshow('blurred image 1',blur1);

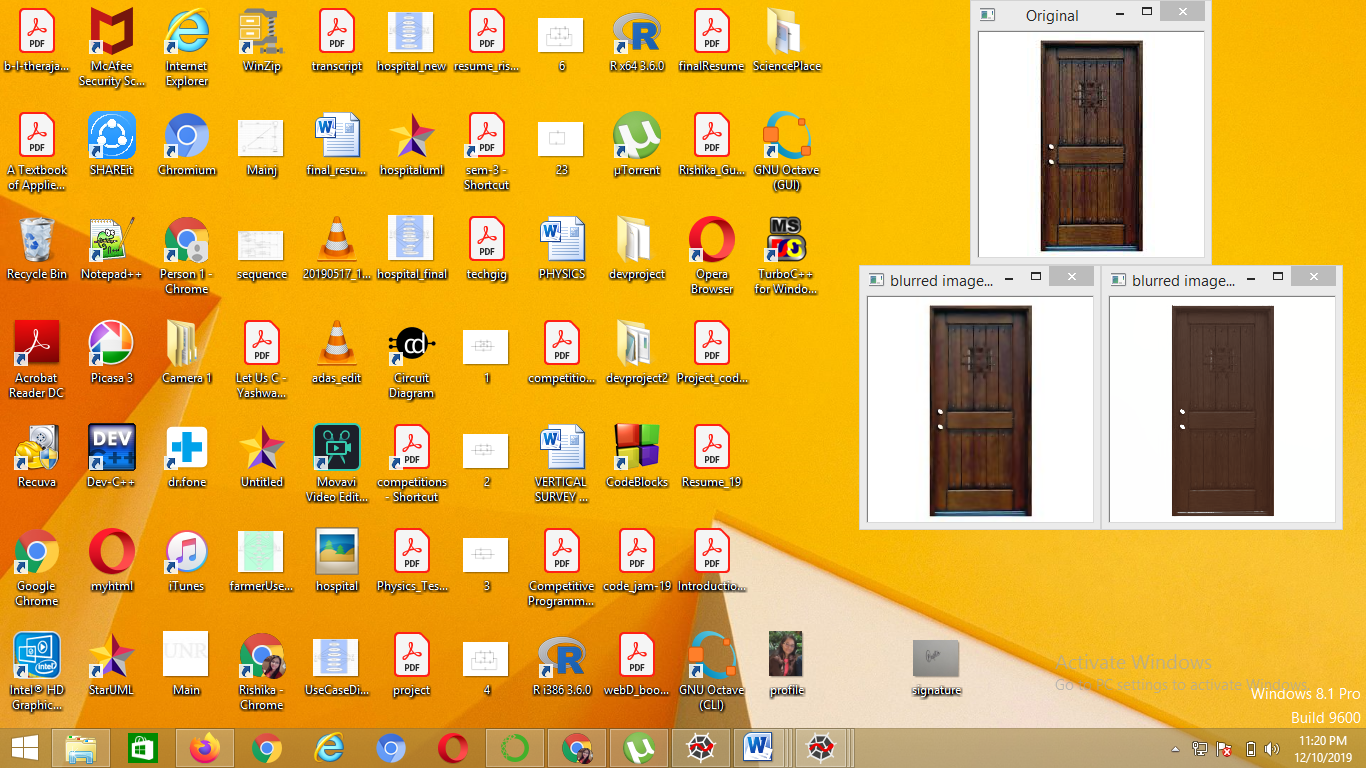
cv2.imshow('blurred image 2',blur2);

#wait for 10 sec

cv2.waitKey(10000);

cv2.destroyAllWindows();

Output:



## 5. 2D Convolution:

## It is actually an image filtering technique which can also be used to remove noise and make it smooth. An arbitrary low pass linear filter is applied on the image to create a blurring effect.

## cv2.filter2D( source, depth, kernel ) function in openCV is used to perform filtering on images. The function has three arguments:

1. source: it signifies the source image on which smoothing has to be done.
2. depth: desired depth of the output image.
3. Kernel: it specifies the kernel filter matrix.

Python code:

# Importing libraries

import cv2

import numpy as np

#read input image

img = cv2.imread('lady.jpg');

kernel = np.ones((4,4),np.float32)/16;

blur = cv2.filter2D(img,-1,kernel);

cv2.imshow('Original',img);

cv2.imshow('blurred image',blur);

#wait for 10 sec

cv2.waitKey(10000);

cv2.destroyAllWindows();

Output:

