Computer Vision ICVI630E - Assignment 1

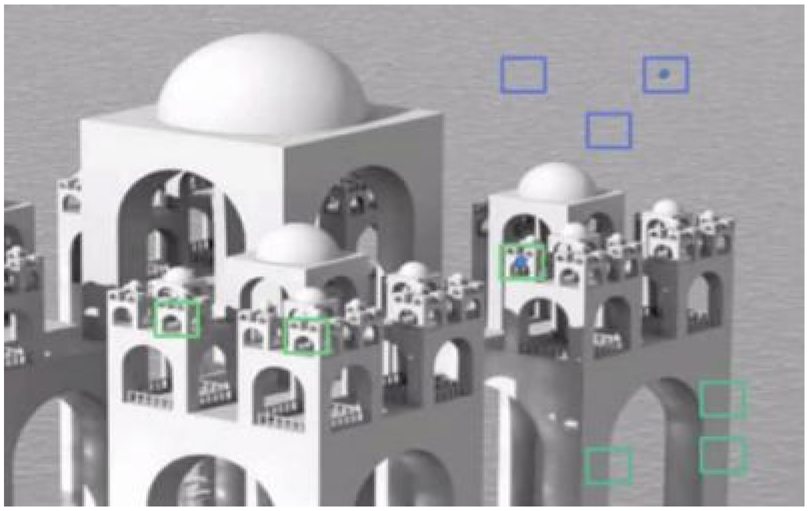
Aditya Goel IRM2016003
6th Semester

Non-local Means Denoising algorithm to remove noise in the image.

Functions in CV: **cv2.fastNlMeansDenoising()**, **cv2.fastNlMeansDenoisingColored()**

Gaussian Blurring, median Blurring are good to some extent in removing small quantities of noise. We take a small neighbourhood around a pixel and do some operations like gaussian weighted average or median of the values etc to replace the central element.
In short, noise removal at a pixel was local to its neighbourhood.

So idea is simple, we need a set of similar images to average out the noise. Consider a small window (say 5x5 window) in the image. Chance is large that the same patch may be somewhere else in the image. Sometimes in a small neighborhood around it. What about using these similar patches together and find their average? For that particular window, that is fine. See an example image below:


(Source: OpenCV Documentation)

The blue patches in the image looks the similar. Green patches looks similar. So we take a pixel, take small window around it, search for similar windows in the image, average all the windows and replace the pixel with the result we got. This method is Non-Local Means Denoising.

OpenCV offers 4 variations for this technique:

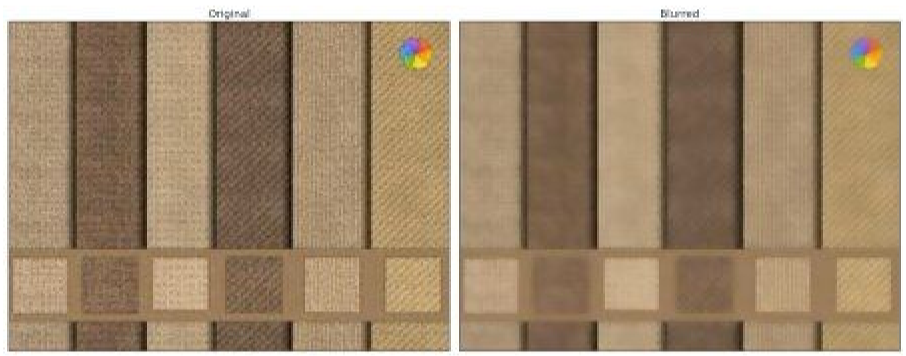
1. **cv2.fastNlMeansDenoising()** - works with a single grayscale images
2. **cv2.fastNlMeansDenoisingColored()** - works with a color image.
3. **cv2.fastNlMeansDenoisingMulti()** - works with image sequence captured in short period of time (grayscale images)
4. **cv2.fastNlMeansDenoisingColoredMulti()** - same as above, but for color images.

<u>Bilateral Filtering</u>

Function: **cv2.bilateralFilter()**

Bilateral Filtering is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters. Gaussian filter is a function of space alone, that is, nearby pixels are considered while filtering. It doesn't consider whether pixels have almost same intensity. It doesn't consider whether pixel is an edge pixel or not. So it blurs the edges also, which we don't want to do.

Bilateral filter also takes a gaussian filter in space, but *one more gaussian filter which is a function of pixel difference*. Gaussian function of space make sure only nearby pixels are considered for blurring while *gaussian function of intensity difference make sure only those pixels with similar intensity to central pixel is considered for blurring*. So it **preserves the edges** since pixels at edges will have large intensity variation.



(The texture on the surface is gone, but edges are still preserved
Source: OpenCV Documentation)

Gaussian Blurring

Function: **cv2.GaussianBlur()**

A Gaussian blur effect is typically generated by convolving an image with a kernel of Gaussian values. In practice, it is best to divide the process into two passes.

1. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction.
2. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass, but requires fewer calculations.

(Gaussian Blur applied to image. Source: Google Images)

Gaussian Kernels and filtering

In Gaussian Filtering, instead of a box filter (NxN mask), a gaussian kernel is used.
We should specify the width and height of kernel which should be positive and odd. We also should specify the *standard deviation in X and Y direction*, sigma$_X$ and sigma$_Y$ respectively. If only sigma$_X$ is specified, sigma$_Y$ is taken as same as sigma$_X$. If both are given as zeros, they are calculated from kernel size. Gaussian blurring is highly effective in removing gaussian noise from the image.

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

(Gaussian Kernels, Google Images)