Tutorial: Spatial Filter

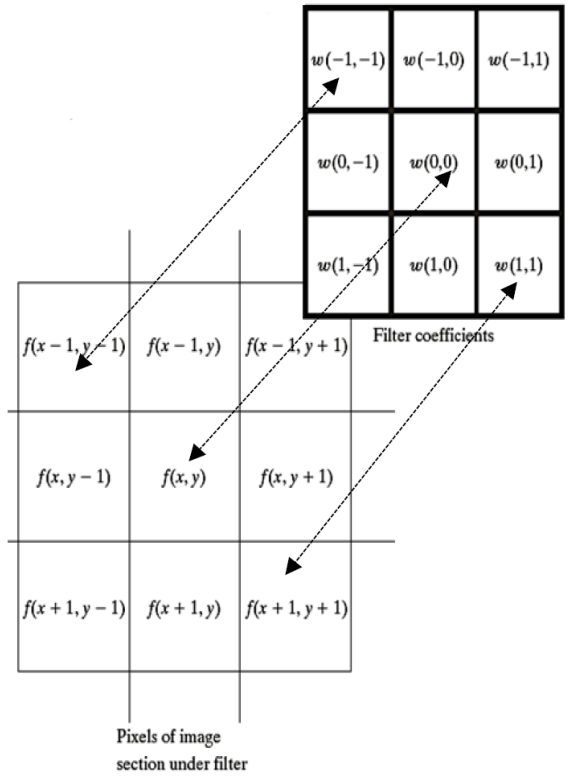
Deep Learning Image Processing.

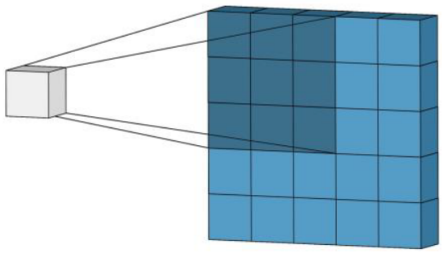
Updated. 2022.2

I. Introduction

In this tutorial, you will learn how to design various spatial filters in OpenCV. You will learn which filters to apply for gaussian and impulse noises and how to combine various filters to enhance the noisy corupted images. Also, you are required to create a camera application which allows the user select the type and size of the image filter.

The correlation of spatial filtering by kernel w(s,t) on the image f(x,y) to obtain the output image g(x,y) is represented as





$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

II. Tutorial

Convolution for Spatial Filter

We will learn how the filter convolution works by programming in MATLAB.

Then, we will learn how to use filter functions in OpenCV.

Exercise

1) Download the example code and test images.

- source code: Matlab filter tutorial
- test images

2) Write a simple program that applies a filter mask w(s,t) to image f(x,y) to result the output image g(x,y).

> Note that the index of an array in a program starts from '0' (C/C++) or '1 (MATLAB)
>
> You need to increase the size of the source image around the border by applying zero padding.

Depending on the type of the filter mask w(s,t), you can obtain smoothing or sharpening filter.

```
% u,v: image points
% s,t: kernel points
% w: kernel, w(s,t)
% f: source image, f(x,y)

clc; clear all; close all;

% image read
f = imread('Pattern_GaussNoise.jpg');
f = rgb2gray(f);
[M, N] = size(f);

% define window
w =[1 1 1 ;  1 1 1 ;  1 1 1];
[wM, wN] = size(w);
wSum = sum(w(:));
if(wSum == 0)
    wSum = 1;
end

%Padding
% e.g. 3 x 3 Filter -> pad 1 each side or 2 total
b = (wM - 1) / 2; % b: yPad
a = (wN - 1) / 2; % a: xPad

% fPad image index: [1 to M+(2*b)] x [1 to N+(2*a)]
fPad = zeros(M+wM-1,N+wN-1);
fPad(a+1:a+M,b+1:b+N) = double(f);

% apply 2D-convolution
gPad = zeros(size(fPad));
tic

for v = b+1:b+M
    for u = a+1:a+N
        % convolution of kernel at one point (u,v)
        conv = 0;
        for t = -b:b
```

```
            for s = -a:a
                % your code goes here
                % your code goes here
            end
        end
        gPad(v,u) = conv / wSum;
    end
end


g = gPad(b+1:b+M, a+1:a+N); % cropping
toc

figure, imshow(f)
figure, imshow(uint8(fPad))
figure, imshow(uint8(g))
```

OpenCV: Filtering

First, read the OpenCV documentation

https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html

- **blur()**
- **GaussianBlur()**
- **medianBlur()**

Example 1. Normalized Block Filter

In this tutorial you will learn how to apply diverse linear filters to smooth images using OpenCV functions such as: blur, GaussianBlur, medianBlur

OpenCV offers the function blur to perform smoothing with the normalized box filter.

```
int i = 3;
blur( src, dst, Size( i, i ), Point(-1,-1) );
```

We specify arguments :

- src: Source image
- dst: Destination image
- Size(w,h): Defines the size of the kernel to be used (of width w pixels and height h pixels)
- Point(-1, -1): Indicates where the anchor point (the pixel evaluated) is located with respect to the neighborhood. If there is a negative value, then the center of the kernel is considered the anchor point.

Example 2. Convolution with Filter Kernel

OpenCV offers the function `filter2D()` for Filter Kernel Convolution.

```
C++: void filter2D(InputArray src, OutputArray dst, int ddepth, InputArray
kernel, Point anchor=Point(-1,-1), double delta=0, int borderType=BORDER_DEFAULT
)

filter2D(src, dst, ddepth , kernel, anchor);
```

Example 3. Gaussian Filter

It is performed by the function GaussianBlur :

```
void cv::GaussianBlur   (
    InputArray  src,
    OutputArray     dst,
    Size    ksize,
    double  sigmaX,
    double  sigmaY = 0,
    int     borderType = BORDER_DEFAULT
)

int i = 3;
GaussianBlur( src, dst, Size( i, i ), 0, 0 );
```

```
blur = cv.GaussianBlur(img,(5,5),0)
```

- Size(w, h): The size of the kernel to be used (the neighbors to be considered). w and h have to be odd and positive numbers.
- sigmaX= The standard deviation in x. Writing 0 implies that is calculated using kernel
- sigmaY=The standard deviation in y. Writing 0 implies that is calculated using kernel

Example 4. Median Filter

This filter is provided by the medianBlur function:

```
// C++
void cv::medianBlur (   InputArray  src,
OutputArray     dst,
int     ksize
)

int i = 3;
medianBlur ( src, dst, i );
```

```
# Python
median = cv.medianBlur(img,5)
```

- i: Size of the kernel (only one because we use a square window). Must be odd.

Example 5. Bilateral Filter

Provided by OpenCV function bilateralFilter

```
// C++

void cv::bilateralFilter    (
    InputArray  src,
    OutputArray     dst,
    int     d,
    double  sigmaColor,
    double  sigmaSpace,
    int     borderType = BORDER_DEFAULT
)

int i = 3;
bilateralFilter ( src, dst, i, i*2, i/2 );
```

```
# Python
blur = cv.bilateralFilter(img,9,75,75)
```

- d: The diameter of each pixel neighborhood.
- sigmaColor : Standard deviation in the color space.
- sigmaSpace : Standard deviation in the coordinate space (in pixel terms)

Example 6: Laplacian operator (Sharpening Filter)

Calculates the Laplacian of an image using Laplacian()

```

void cv::Laplacian  (
    InputArray  src,
    OutputArray     dst,
    int     ddepth,
    int     ksize = 1,
    double  scale = 1,
    double  delta = 0,
    int     borderType = BORDER_DEFAULT
)

int kernel_size = 3;
int scale = 1;
int delta = 0;
int ddepth = CV_16S;
Laplacian( src, dst, ddepth, kernel_size, scale, delta, BORDER_DEFAULT );
src.convertTo(src, CV_16S);
```

```
result_laplcaian = src - dst;
result_laplcaian.convertTo(result_laplcaian, CV_8U);
```

- ddepth: Depth of the destination image. Since our input is CV_8U we define ddepth = CV_16S to avoid overflow
- kernel_size: The kernel size of the Sobel operator to be applied internally. We use 3 in this example.
- scale, delta and BORDER_DEFAULT: We leave them as default values.

III. Exercise

Exercise 1

Download the example code and test images.

- Example code: Filter_demo_student.cpp
- Test images

In the provided sample code, apply the following filters to all test images.

Choose appropriate filter to each image and explain why.

- blur()
- GaussianBlur()
- medianBlur()
- filter2D() ： Design a normalized box filter kernel 5 by 5
- Laplacian()

Show the result images to TA

Exercise 2

Create a camera(webcam) application that has filtering function.

Download the exercise code

- Example code: Webcam Filter Demo

You should make a keyboard input menu to let the user choose the type and size of the filter.

- Filter options: Gaussian, Laplacian(3x3), Median, others..
  - Example: 'B' for blur with Gaussian, 'L' for Laplacian, 'M' for Median
- Let the user also select the size of Gaussian Filter(bluriness)
  - Example: As 'UP' key is pressed,   filter kernel_size is increased from 3, 5, 7, 9 , ...

Show the result images to TA.

Resource

An example code of Spatial Filter with OpenCV-C++

- Example code: Spatial filter demo

An example code of Spatial Filter with OpenCV-Python

- Example code: Spatial filter Python demo