CS682 Quiz 1 Solution Paper

Mithilaesh Jayakumar (G01206238)

8, September 2020

1 Show that the Convolution of two Gaussian functions gσ1 and gσ2 results in a third Gaussian function gσ3.

Let us assume that both the Gaussian functions gσ1 and gσ2 are the same for a one dimensional case. Let the Gaussian function for both these be g(x). The convolution of these two Gaussian functions can be found by substituting g(x) in the formula for Convolution as follows.

=g(x) * g(x)

$= \int_{-\infty}^{\infty} e^{-(\mu)^2/2\sigma^2} e^{-(x-\mu)^2/2\sigma^2} d\mu$

$= \int_{-\infty}^{\infty} e^{-(x/2+\mu)^2/2\sigma^2} e^{-(x/2-\mu)^2/2\sigma^2} d\mu \;\; where \;\; \mu -> \mu + x/2$

$= \int_{-\infty}^{\infty} e^{-\left(2\mu^2+x^2/2\right)/2\sigma^2} d\mu$

$= e^{-(x^2)/4\sigma^2} \int_{-\infty}^{\infty} e^{-(\mu^2)/\sigma^2} d\mu$

$= \sqrt{\Pi}\sigma e^{-\left(x^2\right)/2(\sqrt{2}\sigma)^2}$

Thus the convolution of two Gaussian function results in a Gaussian function.In our example of proof the product of the Convolution of the two Gaussian functions with a spread of σ resulted in a Gaussian function with a spread of $\sqrt{2}\sigma$. This applies for all σ values and holds for the two dimensions as well. In other words $\sigma 3$ is the $\sqrt{(\sigma 1)^2 + (\sigma 2)^2}$ during convolution.

2 Is it possible for this filter to be represented by a Gaussian function: e.g., can we find a z and σ such that we can construct the filter k using $g(d) = z\ exp\ (-d^2/2\sigma^2)$where d is the distance (in pixels) from the center of the filter? Is this result surprising? Derive a formula for a 7-element filter kernel that satisfies the criteria we discussed in class, including that it be symmetric, maintain the magnitude of a constant image and satisfy the equal contribution criterion. How many free parameters does it have?

We know that the convolution of two Gaussian filters is a Gaussian filter i.e $\sigma 3^2 = (\sigma 1)^2 + (\sigma 2)^2$.For example,

$[11] * [11] = [121]$
$[11] * [121] = [1331]$ and so on.,

which is nothing but a Pascal's triangle. The even rows in the Pascal's triangle with n values represent the nxn filters.

One way of finding the values of the Pascal's triangle is using Binomial approximation where the binomial coefficients of $(1 + X)^n$ represent the values of a nxn filter.

We can also represent this filter by using a Gaussian function with approximation. In order to do so we need to compute the weights of the mask by using the discrete Gaussian distribution. The formula is as follows.

g[d] = z $e^{-(d^2)/2\sigma^2}$
where d is the distance (i,j) in pixels from the center of the filter and z is the normalizing constant. The above equation can be re-written as follows.
g[d]/z = $e^{-(d^2)/2\sigma^2}$
We can construct any nxn filter or mask by choosing a value for σ^2 where the value at the center of the filter is 1. Now for constructing the 5x5 filter or mask let us consider the value of σ^2 as 1. Then we get the following array of values.

d	-2	-1	0	1	2
-2	0.02	0.08	0.14	0.08	0.02
-1	0.08	0.37	0.61	0.37	0.08
0	0.14	0.61	1	0.61	0.14
1	0.08	0.37	0.61	0.37	0.08
2	0.02	0.08	0.14	0.08	0.02

As the resulting filter values are not integers we need to convert these into integers. Filters with integer values are easy for computation compared to decimal values. So we take the decimal values at one of the corners and compute z such that its value becomes 1. For the above filter the z value can be calculated as follows.
z = 1.0/0.02
z = 50
Now we multiply the rest of the filter values by the value of z. The resulting filter is as follows

d	-2	-1	0	1	2
-2	1	4	7	4	1
-1	4	19	31	19	4
0	7	31	50	31	7
1	4	19	31	19	4
2	1	4	7	4	1

This is the resulting mask for the 5x5 Gaussian filter. However, as the sum of the weights of the mask do no equal to 1, we need to normalize the output pixel values by the sum of the wights of the mask which is 314 for our 5x5 filter. In order to calculate for a 7x7 filter we can use the same formula used for the 5x5 filter. Let us consider the σ^2 value as 2 for the 7x7 filter. Then

d	-3	-2	-1	0	1	2	3
-3	0.011	0.039	0.082	0.105	0.082	0.039	0.011
-2	0.039	0.135	0.287	0.368	0.287	0.135	0.039
-1	0.082	0.287	0.606	0.779	0.606	0.287	0.082
0	0.105	0.368	0.779	1	0.779	0.368	0.105
1	0.082	0.287	0.606	0.779	0.606	0.287	0.082
2	0.039	0.135	0.287	0.368	0.287	0.135	0.039
3	0.011	0.039	0.082	0.105	0.082	0.039	0.011

We follow the same procedure like before to compute z such that the value becomes 1 at the corners. For the above 7x7 filter the z value can be calculated as follows.
z = 1.0/0.011
z = 91
Now we multiply the rest of the filter values by the value of z. The resulting filter is as follows

d	-3	-2	-1	0	1	2	3
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

This is the resulting mask for the 7x7 Gaussian filter. Like before we normalize the output pixel values by the sum of the wights of the mask which is 1115 for our 7x7 filter. In both the cases we are normalizing so that the regions of uniform intensity is not affected.

3 If the output image is the same size at the input image, we would expect that the image is unchanged by "interpolation". Does the normalized sinc function satisfy this property?Why it makes sense that the two of these would be equivalent?

Every time domain waveform has a corresponding frequency domain waveform, and vice versa. Wave forms that correspond to each other in this manner are called Fourier transform pairs. The two operations - Fourier transform of the Rect() function in the frequency domain and the normalized Sinc() function in the time domain are the same. In other words, the low pass filtering of the Rect() function in the frequency domain and the normalized Sinc() interpolation in the time domain both mean the same.

Yes. The normalized Sinc() function satisfy the property that the image should be unchanged if both the input and the output images are of the same size.

The normalized Sinc() function helps us to represent the waveform of each discrete sample without adding of any higher/lower frequency content. Discrete signals are used to represent continuous signals and continuous signals do not alias. By using a normalized Sinc() function the value remains the same at the zero frequency which helps to remove aliasing.