Functional Convolution and Applications to Computer Graphics

Allen Chen

June 6, 2017

Contents

1 Introduction to Functional Convolution

The convolution of two functions is essential to filtering in image and signal processing, as filtering is done by convolving signals with specifically designed kernels. In fact, the advent of computer graphics allows us to visualize the effect of convolving two functions and is invaluable in education and computer vision. There are limitless varieties of graphical effects that can be achieved simply through 2D convolutions. In this paper, we will review the paper *Convolutions and Computer Graphics* by A. Burns, where the author provides the basis of applying convolution in graphics software to achieve various desired effects in image processing, such as edge detection, sharpening, and blurs. Let $f(x)$ and $k(x)$ be real valued functions of real variables. Then, we define a convolution of f with k as follows :

$$F(t) = \int_{-\infty}^{\infty} f(x)k(t-x)dx$$

Here, f represents the input signal, and we define k as the kernel of F. In the operation of filtering, a weighing kernel is convolved with an input signal to compute a weighted image as the output signal F. In particular, we are interested in convolution with k as the Gaussian Kernel :

$$\text{Gaussian Kernel, } G_\sigma(x) = \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi}\sigma}, \sigma > 0$$

Gaussian convolution is a common operation and the basis of many sophisticated algorithms in signal and image processing. It is commonly used as a means to blur 2D images, in effect removing both noise (random variation in brightness) and detail from an image. Consequently, its efficient computation is important and many approximations exist. In the following sections, I will provide an extension to 2D convolution, various properties of the Gaussian, and one of many approximations, discussing their connections with modern day computer graphics.

2 2D Convolution

Generalizing to two dimensions, we have the convolution of two real functions, $f(x,y)$ and $k(x,y)$ as a new function of two variables as follows [1]:

$$F(s,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)k(s-x,t-y)dxdy$$

Suppose the kernel has compact support as follows

$$k(x,y) = 0 \text{ if } |x| > h, |y| > r$$

Then, we make the following substitution :

$$u = s - x$$

$$du = -dx$$

$$v = t - y$$

$$dv = -dy$$

and hence we have

$$\begin{aligned}F(s,t) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(s-u,t-v)k(u,v)(-du)(-dv)\\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(s-u,t-v)k(u,v)dudv\\ &= \int_{-r}^{r}\int_{-h}^{h} f(s-u,t-v)k(u,v)dudv\end{aligned}$$

As we are concerned with graphics on a computer screen, let S be the discrete subspace of 2D Euclidean space consisting of such points that have integer coordinates. Let x be defined as the horizontal position and y be defined as the vertical position of a pixel. If we choose a kernel function that is 0 outside of the rectangle $\{(x,y)| -h \leq x \leq h, -r \leq y \leq r\}$, then for all $(x,y) \in \mathbb{Z} * \mathbb{Z}$ the convolution of $f(x,y)$ and $k(x,y)$ becomes :

$$F(s,t) = \sum_{v=-r}^{r}\sum_{u=-h}^{h} f(s-u,t-v)k(u,v)$$

$$F(x,y) = \sum_{j=-r}^{r}\sum_{i=-h}^{h} f(x-i,y-j)k(i,j)$$

Relating to a computer screen, consider the screen as a bounded subspace of S, where we have x-coordinates from $0,1,2,3,...,X$, and y-coordinates from $0,1,2,3,...,Y$. This totals $X+1$ horizontal pixels and $Y+1$ vertical pixels. Our kernel function ranges from $-h$ to h horizontally and $-r$ to r vertically. Hence, our kernel is applied to $(h+h+1)(r+r+1) = (2h+1)(2r+1)$ pixels surrounding $f(x,y)$. Now, consider a function $f(x,y)$ that is defined on our bounded subspace and takes on integer values of $0,1,2,...,N-1$, where N represents the number of colors available. One example of such a function is the RGB(Red, Green, Blue) color model, which has $N = 256$. We have functions $f_r(x,y), f_g(x,y), f_b(x,y)$ that represent respectively the values of the red, green, and blue components of a pixel at (x,y). After convolving with a kernel $k(i,j)$, we then have functions $F_r(x,y), F_g(x,y), F_b(x,y)$ which represent the color components of the pixel at (x,y) after filtering. Then, these values are rounded and ensured to be lower-bounded by 0 and upper bounded at 255. Finally, we are able to synthesize the color of our output pixel using the composition of its components.
Using our definition of the kernel function, we can visualize $F(x,y)$ by picturing an array of kernel values imposed upon the screen, centered at $f(x,y)$. We have $k(0,0)$ is applied to $f(x,y)$, $k(0,1)$ applied to $f(x,y-1)$, and so on. When we sum up the $(2h+1)(2r+1)$ pixels surrounding (x,y) after applying the kernel, the result is $F(x,y)$, which we round to an integer representing the new colored pixel shown on screen. It is noteworthy that this can also be thought of as $(2r+1)$ 1D convolutions of a row of pixels of length $2h+1$. This inevitably runs into problems as we approach the boundary due to the fact that f(x,y) is undefined outside our domain of definition.

3 Boundary Conditions

It is important that we consider the boundaries of an image when we apply convolution. As both are finite in size, some f will lie outside the domain of definition. Consider a singular row of pixels.

Let $f(x)$ be defined for the finite row length N, then $f(x)$ is defined for integers $x = 0, 1, ..., N-1$. In terms of 1D boundary handling, several possible options are

$$\text{zero-padding, } \widehat{f}(x) = \begin{cases} f(x) & 0 \leq x \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{constant, } \widehat{f}(x) = f(i), i = \min(\max(x, 0), N-1)$$

$$\text{symmetric, } \widehat{f}(x) = \begin{cases} f(x) & 0 \leq x \leq N-1 \\ \widehat{f}(-1-x) & x < 0 \\ \widehat{f}(2N-1-x) & x \geq N \end{cases}$$

Using the 1D case provided in [2], we extend this to apply for f of 2 variables. Consider a 2D image of N by M pixels. Let $f(x, y)$ be defined for $x = 0, 1, ..., N-1$ and $y = 0, 1, ..., M-1$. We have the options

$$\text{zero-padding, } \widehat{f}(x, y) = \begin{cases} f(x, y) & (0 \leq x \leq N-1), (0 \leq y < M-1) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{constant, } \widehat{f}(x, y) = f(i, j), i = \min(\max(x, 0), N-1), j = \min(\max(y, 0), M-1)$$

symmetric, $\widehat{f}(x, y)$ such that x,y follow

$$x = \begin{cases} x & 0 \leq x \leq N-1 \\ -1-x & x < 0 \\ 2N-1-x & x \geq N \end{cases}$$

$$y = \begin{cases} y & 0 \leq y \leq M-1 \\ -1-y & y < 0 \\ 2M-1-y & y \geq M \end{cases}$$

This allows us to extend f to be defined outside its domain of definition and therefore continue to perform convolution for pixels involving the boundary.

4 Gaussian Properties

Compared to many convolution kernels, there are various interesting properties of the Gaussian kernel that make it especially powerful.[2]

Separability : The Gaussian is separable.

$$e^{-(x^2+y^2)/2\sigma^2} = e^{-(x^2)/2\sigma^2} e^{-(y^2)/2\sigma^2}$$

Semigroup Property : The convolution of two Gaussians is another Gaussian. Let G_σ denote a Gaussian kernel. Then,

$$G_{\sigma 1} * G_{\sigma 2} = G_\sigma, \sigma = \sqrt{{\sigma_1}^2 + {\sigma_2}^2}$$

Discretization of the Gaussian : We consider a discretization of the Gaussian so we can apply it to a 1D signal like a row of pixels.

$$g(x) = G_\sigma(x), x \in \mathbb{Z}$$

Normalization : We often normalize values to ensure they have unit sum.

$$g_n(x) = G_\sigma(x)/S$$

$$S = \sum_{x=-\infty}^{\infty} G_\sigma(x)$$

Lemma 1 : For any $\sigma > 0$, the sum S is strictly greater than 1.
Proof : Using Poisson's summation formula ($\sum_{n=-\infty}^{\infty} f(n)e^{-2\pi nx} = \sum_{n=-\infty}^{\infty} \hat{f}(x-n)$) and $\hat{G}_\sigma(k) = e^{-2\pi^2\sigma^2k^2}$ we have

$$S = \sum_{x=-\infty}^{\infty} G_\sigma(x) = \sum_{k=-\infty}^{\infty} \hat{G}_\sigma(k) = 1 + 2\sum_{k=1}^{\infty} e^{-2\pi^2\sigma^2k^2} > 1$$

5 Finite Impulse Filtering

One of many implementations of Gaussian convolution is by approximating a finite impulse response (FIR) filter, where the Gaussian is truncated to $|x| \leq r$. In this section, I prove that the error from using such a truncated filter as an approximation during convolution is bounded for a 1D signal. Let us define the truncated filter as follows:

$$g^{trun}(x) = G_\sigma(x)/s(r), |x| \leq r$$

$$s(r) = \sum_{x=-r}^{r} G_\sigma(x)$$

We will show that when approximating $(g * \widehat{f})$ with $(g^{trunc} * \widehat{f})$, the error is bounded. Consider the distance between g and g^{trunc}.

$$||g - g^{trunc}||_1 = \sum_{|x|\leq r} \frac{G_\sigma(x)}{s(r)} - \frac{G_\sigma(x)}{s(\infty)} + \sum_{|x|>r} \frac{G_\sigma(x)}{s(\infty)}$$

We rewrite the first sum through factoring.

$$\begin{aligned}
\sum_{|x|\leq r} \frac{G_\sigma(x)}{s(r)} - \frac{G_\sigma(x)}{s(\infty)} &= s(r)(\frac{1}{s(r)} - \frac{1}{s(\infty)}) \\
&= 1 - \frac{s(r)}{s(\infty)} \\
&= \frac{s(\infty) - s(r)}{s(\infty)} \\
&= \frac{1}{s(\infty)} \sum_{|x|>r} G_\sigma(x)
\end{aligned}$$

Substituting back into the difference, we have

$$\begin{aligned}
||g - g^{trunc}||_1 &= \frac{2}{s(\infty)} \sum_{|x|>r} G_\sigma(x) \\
&= \frac{4}{s(\infty)} \sum_{x=r+1}^{\infty} G_\sigma(x) \\
&\leq \frac{4}{s(\infty)} \int_r^\infty \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}} dx \\
&= \frac{4}{s(\infty)} * 0.5 * \sqrt{\sigma^2/\sigma^2} * erfc(\frac{r}{\sqrt{2\sigma^2}}) \\
&= \frac{2}{s(\infty)} erfc(\frac{r}{\sqrt{2\sigma^2}})
\end{aligned}$$

Here, $erfc$ is the complementary error function $erfc(x) = 1 - erf(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$. Using lemma 1, we have $\frac{2}{s(\infty)} < 2$. Hence, by Young's inequality for function convolution ($||f * g|| = ||f|| * ||g||$), we have

$$||g * \widehat{f} - g^{trunc} * \widehat{f}||_\infty \leq 2erfc(\frac{r}{\sqrt{2\sigma^2}})||f||_\infty$$

The result is powerful in that it can tell us how large of a radius r we need for a desired level of accuracy. Select r as $\lceil \sqrt{2} erfc^{-1}(t/2)\sigma \rceil$, this ensures error $E(t) \leq t * ||f||_\infty$. For example, using $\sigma = 1$ and $t = 10^{-2}$, we have $r = \lceil \sqrt{2} erfc^{-1}(t/2) \rceil \approx \lceil 2.8 \rceil = 3$. In a conventional RGB system, we have $E(0.01) \leq 0.01 * 255 = 2.55$, hence the error in each color is off by a maximum of 3 color values.

The results easily extend into 2D as the Gaussian is separable. For (x, y), the 2D Gaussian is $e^{-(x^2+y^2)/2\sigma^2}$. We can fix y and do 1D convolution x, multiplying by $e^{-y^2/2\sigma^2}$ when complete. In actuality we are doing convolution per row, multiplying by a pre-defined constants, then summing the results for all rows.

6 Applications

As I mentioned earlier in the introduction, computer graphics now allow us to visualize the effects or convolving functions. I present several images to display the results discussed above. All images are generated using the CSE457 Impressionist project, which implements constants for boundary handling.

Original Image

Convolved Image, $r = 3, \sigma = 1$

Convolved Image, $r = \infty, \sigma = 1$

Convolved Image, $r = 3, \sigma = 3$

Convolved Image, $r = \infty, \sigma = 1$

It is quite evident that there aren't really any visible differences between $r = 3$ and $r = \infty$ when $\sigma = 1$. This is visual proof that using a truncated Gaussian for a small r and small σ is enough to create very accurate approximations. However, for $\sigma = 3$, we can see noticeable differences between $r = 3$ and $r = \infty$, as it would take a $t \approx 0.7$, leading to $E(0.7) \leq 0.7 * 255 = 178.5$, which means that each color component can be offset by a huge amount.

7 Concluding Remarks

Functional convolution with the Gaussian kernel is good baseline to use for elementary image filtering in computer graphics. We discussed convolution in the context of infinite integrals and explored its discretization in context of our everyday computer screens. We defined various ways to extend an colored image beyond its domain of definition that is essential to discrete convolution. Finally, we proved that the FIR filter provides a means to approximate Gaussian convolution that is bounded. Overall, FIR filtering works well when σ is small but care is needed in applying it to large σ as it becomes quite costly in terms of accuracy when σ is big.

8 References

1. Burns, Anne M. "Convolutions and Computer Graphics." Mathematics Magazine, vol. 67, no. 4, 1994, pp. 258–267. JSTOR, www.jstor.org/stable/2690844.

2. Pascal Getreuer, A Survey of Gaussian Convolution Algorithms, Image Processing On Line, 3 (2013), pp. 286–310. https://doi.org/10.5201/ipol.2013.87