Reprint from proceedings of SCIA9, Uppsala, Sweden, June, 1995

A Generalized Convolver

Johan Wiklund Hans Knutsson

Computer Vision Laboratory
Linköping University
581 83 Linköping, Sweden
email: `jowi@isy.liu.se`

Abstract

A scheme for performing generalized convolutions is presented. A flexible convolver, which runs on standard workstations, has been implemented. It is designed for maximum throughput and flexibility. The implementation incorporates spatio-temporal convolutions with configurable vector combinations. It can handle general multi-linear operations, i.e. tensor operations on multidimensional data of any order. The input data and the kernel coefficients can be of arbitrary vector length. The convolver is configurable for IIR filters in the time dimension. Other features of the implemented convolver are scattered kernel data, region of interest and subsampling. The implementation is done as a C-library and a graphical user interface in AVS (Application Visualization System).

1. Introduction

A procedure to perform convolutions on multi-dimensional data with arbitrary filter kernels is a basic tool in image and signal processing [2, 3]. Typical input data are 1D signals, 2D images, 3D volumes, 3D spatio-temporal image sequences and 4D volume sequences. Each coordinate, (pixel, voxel, toxel), can contain a scalar value or a vector.

If the kernel and/or the input data has a vector length larger than one, a generalized convolution is needed. In this case the multiplication in the convolution is changed to a vector combination. This type of multilinear convolution is in this paper termed generalized convolution. Examples of vector valued input data are color (RGB), 2D vector fields, 3D vector fields, tensor fields etc.

A flexible convolver has been implemented that can perform generalized spatio-temporal convolutions with arbitrary kernel data. The kernel coefficients can be scattered, i.e they don't need to be uniformly placed inside a box. The computational cost increases lineary with the number of kernel coefficients, it does not depend on the size of the kernel bounding box. A region of interest (ROI), e.g. a spatial rectangle in the input over which the convolution should be applied, can be defined. Subsampling is user selectable and decreases the computational cost for the convolution.

There are two basic classes of filters, FIR (finite impulse response) and IIR (infinite impulse response) [1]. Both types of filters are supported by the convolver.

2. Generalized convolution

Convolution is defined as a product sum of a kernel K and input data I over a neghbourhood m:

$$\text{out}(n) = \sum_m K(m)\, I(n-m) \tag{1}$$

where m is the spatio-temporal coordinate in the kernel and n is the spatio-temporal coordinate in the input.

In a more general case the kernel and/or the input data has a vector length larger than one. This is common when working with multilinear operators, i.e. tensors. In this case the multiplication in the convolution is changed to a vector combination defined by the operation matrix M:

$$\text{out}_o(n) = \sum_{k,i} M_{o,k,i} \sum_m K_k(m)\, I_i(n-m) \tag{2}$$

where subscripts means vector components and parameters in parenthesis means spatio-temporal coordinates.

n is an index indicating the spatio-temporal coordinate in the input and output data.

m is an index indicating the spatio-temporal coordinate in the kernel.

out_o is vector component o in the output data.

K_k is vector component k of the kernel coefficients.

I_i is vector component i in the input data.

$M_{o,k,i}$ defines the operation structure, i.e. the vector combination to use between input data vectors and kernel coefficient vectors, for each output vector component.

2.1. Operation structure

The operation structure matrix M defines the type of operation to perform in the convolution. The summation over the spatio-temporal neighbourhood (equation 2) is independent of the structure matrix. It defines how the vector components in the kernel shall be combined with the vector components in the input data. Currently the only allowed values in the operation matrix are $-1, 0$ and 1. This restriction has the effect that no extra multiplications are needed in the convolution computations (equation 2).

There are a number of predefined operation matrices that can be selected, i.e complex multiplication, inner product and outer product. It is also possible to specify a user defined matrix.

In the complex multiplication mode, both the input data and the kernel coefficients consists of a two-component vector. The first component corresponds to the real part and the second component corresponds to the imaginary part of a complex number. The operation matrix that defines complex convolution looks like this:

$$\begin{aligned} M_{0,k,i} &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ M_{1,k,i} &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{aligned}$$

which corresponds to a complex multiplication of cartesian components:

$$\begin{aligned}\text{out}_{re} &= K_{re} \cdot I_{re} - K_{im} \cdot I_{im} \\ \text{out}_{im} &= K_{re} \cdot I_{im} + K_{im} \cdot I_{re}\end{aligned}$$

If the input data or the kernel data just contains one component, i.e it is a scalar value, half of the multiplications are unnecessary. The convolver detects such cases and skips those multiplications, which leads to an increase of the speed with a factor of two. If both kernel and input data are scalar, the computation collapses to an ordinary scalar convolution. In this case the speed increases with a factor of four compared to a complex convolution. It will, however, still produce complex output, with the imaginary part being zero. To perform scalar convolutions, the inner product is used.

In the inner product mode, the input data and kernel data vector lengths has to be equal. The output data becomes scalar. Equation 2 can be rewritten as

$$\text{out}(n) = \sum_i \sum_m K_i(m)\, I_i(n-m) \tag{3}$$

In the outer product mode, the output data vector length equals the product of the input data and the kernel data vector lengths. In this case, equation 2 can be rewritten as

$$\text{out}_{k,i}(n) = \sum_m K_k(m)\, I_i(n-m) \tag{4}$$

This mode gives the scalar convolution result for each component in the kernel with each conponent in the input data.

More elaborate operations can be defined by defining a suitable structure matrix and give it as input to the convolver. As example, consider a matrix multiplication where the kernel and input data has four components each, $K_k = [a\ b\ c\ d]$ and $I_i = [e\ f\ g\ h]$, producing an output vector $\text{out}_o = [p\ q\ r\ s]$. Let the elements in these three vectors constitute 2×2 matrices:

$$\begin{pmatrix} p & q \\ r & s \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

This operation is defined by the following structure matrix M:

$$M_{0,k,i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad M_{1,k,i} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M_{2,k,i} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad M_{3,k,i} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Convolution with IIR filters

The convolver is capable to deal with IIR filters in the time dimension. The reason for not implementing IIR functionality in the spatial dimensions is that there seems not to be any obvious advantages to use such filters spatially. Futhermore, IIR implementation in more than one dimension can be a problem.

An IIR filter may be implemented in direct form by expressing one output sample in terms of the input samples and previously computed output samples.

$$y(k) = \sum_{i=0}^{n} c_i\, u(k-i) - \sum_{i=1}^{n} d_i\, y(k-i) \tag{5}$$

In figure 1, a realization of an IIR filter is shown.

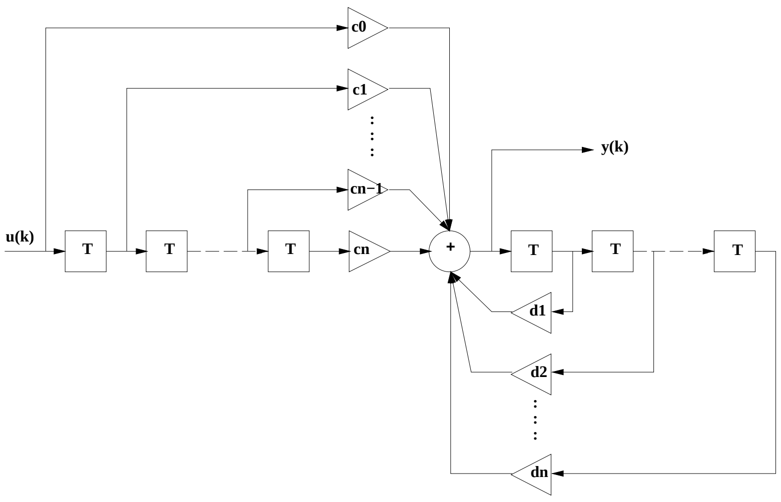


Figure 1: Realization of IIR filter, direct form I.

In figure 2, a realization of the same filter is shown that only needs half the number of delays. This realization is called direct form II and is described by the following equations:

$$\begin{aligned} w(k) &= u(k) - \sum_{i=1}^{n} d_i\, w(k-i) \\ y(k) &= \sum_{i=0}^{n} c_i\, w(k-i) \end{aligned} \tag{6}$$

The direct form II structure reduces the amount of storage required by allowing storage to be shared between the feedback and feedforward loops. This realization scheme is used in the convolver, because it minimizes the size needed for the spatio-temporal buffer. Futhermore, it makes it possible to implement IIR convolution as an add-on to the FIR convolution. Consider equation 6; if the weights d_i is zero, then the equation is similar to a FIR convolution. This means that a IIR filter is implemented as two parts; one feedback part (first row in equation 6) and one feedforward part (second row in equation 6).

4. Convolver structure

Figure 3 shows the structure of the generalized convolver. The input data, scalar or vector valued, is shifted into the spatio-temporal buffer. The feedback part of the IIR-filter (if present) operates on this buffer, and the result is added to the input at the beginning of the buffer. Then the FIR-filter is applied and the final result is stored on the output.

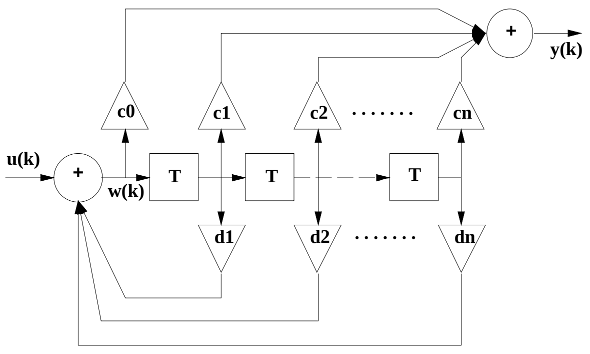


Figure 2: Realization of IIR filter, direct form II.

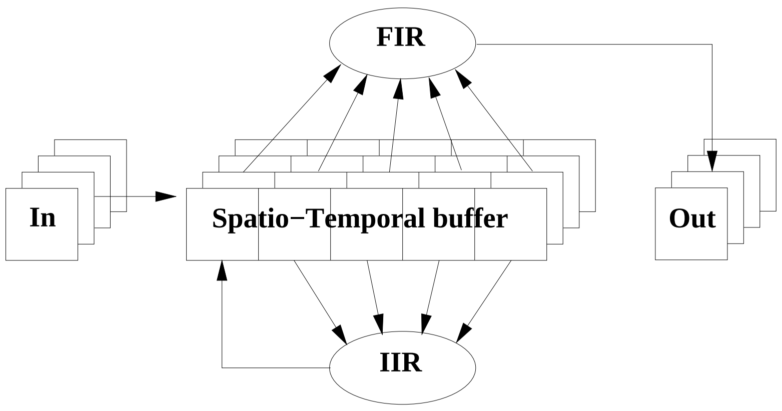


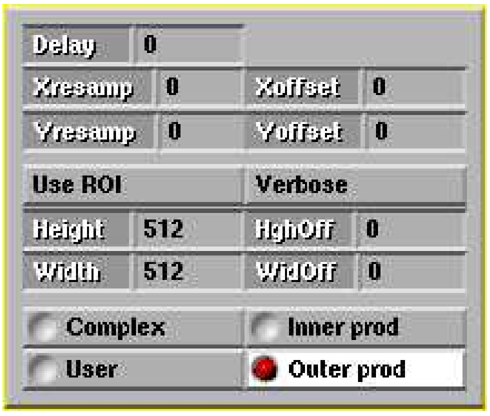
Figure 3: Convolver structure

The size of the spatio-temporal buffer is decided by the size of the kernel and the size of the input data. The spatial size equals the size of the input plus the borders needed to keep the kernel inside the buffer during the convolution. The temporal size equals the kernel span in the time dimension. The number of frames from the current frame to the origin in the kernel defines the delay in the filter. It is possible to increase the delay. This has shown to be useful when different parallel convolutions with different delays has to be synchronized. This is implemented by adding space for additional frames to the beginning of the spatio-temporal buffer.

The design decision to keep the whole frames, for the time span needed, in memory is based on efficiency and simplicity. It is efficient because the frames are never moved between different memory locations, they stay in place until they are not needed anymore. It is simple because there is no need for swapping part of frames in to and out from memory. Even concerning memory usage it is feasible, a typical application may be seven 512×512 frames in the buffer which equals 7 Mb. The spatio-temporal buffer is maintained automatically and adapts to the present kernel and input data.

5. Implementation

The convolver is implemented as a number of library functions. Some of these maintains multi-dimensional arrays in memory as efficient as possible. Others deal with the actual convolution and format conversions. Input data can be byte, short, int or float type. All computations is done in float precision. This has shown to be faster than performing the convolution in integer precision on standard platforms. Another important advantage with calculations in float precision is that normalization is straight forward and there is no problem with overflow or resolution in the product sum. The library is written in C. The convolver application is implemented in AVS (Application Visualization System) [4], in which the user interface is defined. AVS is an interactive visualization environment, in which algorithms easily can be designed and tested using visual programming, i.e. by choosing and interconnecting program modules graphically.



Delay	0		
Xresamp	0	Xoffset	0
Yresamp	0	Yoffset	0
Use ROI		Verbose	
Height	512	HghOff	0
Width	512	WidOff	0
Complex		Inner prod	
User		[✓ Outer prod]	

Figure 4: Input parameter interface for the convolver.

Figure 4 shows the user interface for the parameters that controls the behaviour of the convolver. This control panel is connected to the module “Convolve ST” in figure 5. This

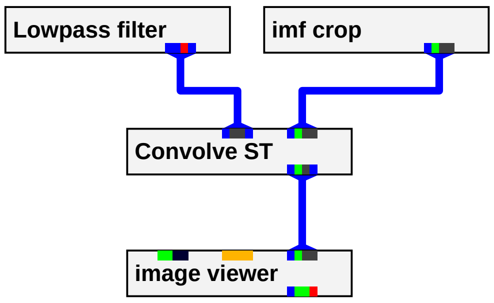


Figure 5: AVS network including the convolver.

network consists of four modules, “Lowpass filter” creates an averaging kernel, “imf crop” reads image data from file and “image viewer” displays the result in a window on the screen.

6. Performance tests

A number of tests has been carried out to get timing information for different convolution modes, image and kernel sizes. The tests has been performed on a SUN Sparc 10 with one 40Mhz cpu. To be able to compare the different timing results, a convolution rate measure has been calculated. The convolution rate (C_r) is defined as:

$$C_r = \frac{K_c \cdot I_h \cdot I_w}{t} \tag{7}$$

where

C_r is the convolution rate.

K_c is the number of kernel coefficients. The coefficients can be scalar or vector-valued.

I_h is the height of the output image.

I_w is the width of the output image.

t is the convolution time in micro-seconds.

The convolution rate is a similar measure as Mflops (Million floating point operations per second). It could be described as Million vector combinations per second.

Convolution time (seconds)			Convolution rate		
Image size	Coefficients		Image size	Coefficients	
wid×hgh	9	25	wid×hgh	9	25
512×512	0.67	1.84	512×512	3.52	3.56
512×256	0.36	0.95	512×256	3.28	3.45
512×128	0.18	0.48	512×128	3.28	3.41
256×512	0.37	0.95	256×512	3.93	3.45
128×512	0.19	0.49	128×512	3.10	3.34

Table 1: Convolution performance, scalar kernel coefficients and scalar image data.

In table 1 the performance for scalar convolution is shown. As can be seen, the convolution rate is independent of the number of coefficients in the kernel and the size of the image. This means that the speed is proportional to the number of multiplications needed.

Convolution time (seconds)				Convolution rate			
Image size	real	real	complex	Image size	real	real	complex
wid×hgh	real	complex	complex	wid×hgh	real	complex	complex
512×512	4.04	8.10	16.41	512×512	3.17	1.58	0.78
512×256	2.02	4.05	8.29	512×256	3.17	1.59	0.77
512×128	1.04	2.04	4.11	512×128	3.09	1.57	0.78
256×512	2.02	4.04	8.14	256×512	3.17	1.59	0.79
128×512	1.05	2.08	4.14	128×512	3.06	1.54	0.78

Table 2: Complex convolution performance.

Table 2 shows the performance for complex convolution. Three different cases can occur:

- Real kernel, real image. The same case as scalar convolution. This case requires one multiplication per coefficient.
- Real kernel, complex image (or vice versa). This case requires two multiplications per coefficient. One for the real part and one for the imaginary part.
- Complex kernel, complex image. This case requires four multiplications per coefficient. Two for the real part and two for the imaginary part.

As can be seen the convolution rate only depends on the type of complex operation; the ratio between the cases are 4:2:1.

Subsampling performance		
Output size	performance	
wid×hgh	time	rate
512×512	5.06	3.57
512×256	2.52	3.59
512×128	1.29	3.51
512×64	0.64	3.53
256×512	2.89	3.13
128×512	1.79	2.52
64×512	1.25	1.81

Table 3: Subsampling performance, scalar convolution.

In table 3 the performance for subsampling of a 512×512 scalar image using a scalar kernel with 69 coefficients is shown. There is a decrease of the convolution rate for subsampling along the x-axis, the reason is not clear.

7. Software availability

The software described is available via anonymous ftp at `isy.liu.se`, in the file `pub/bb/convolver.tar.gz`.

References

[1] D. E. Dudgeon and R. M. Mersereau. *Multidimensional Digital Signal Processing*. Prentice-Hall signal processing series. Prentice-Hall, 1984. ISBN 0-13-604959-1.

[2] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[3] B. Jähne. *Spatio-Temporal Image Processing: Theory and Scientific Applications*. Springer Verlag, Berlin, Heidelberg, 1993. ISBN 3-540-57418-2.

[4] C-J. Westelius, J. Wiklund, and C-F. Westin. Prototyping, visualization and simulation using the Application Visualization System. In H. I. Christensen and J.L. Crowley, editors, *Experimental Environments for Computer Vision and Image Processing*, volume 11 of *Series on Machine Perception and Artificial Intelligence*, pages 33–62. World Scientific Publisher, 1994. ISBN 981-02-1510-X.