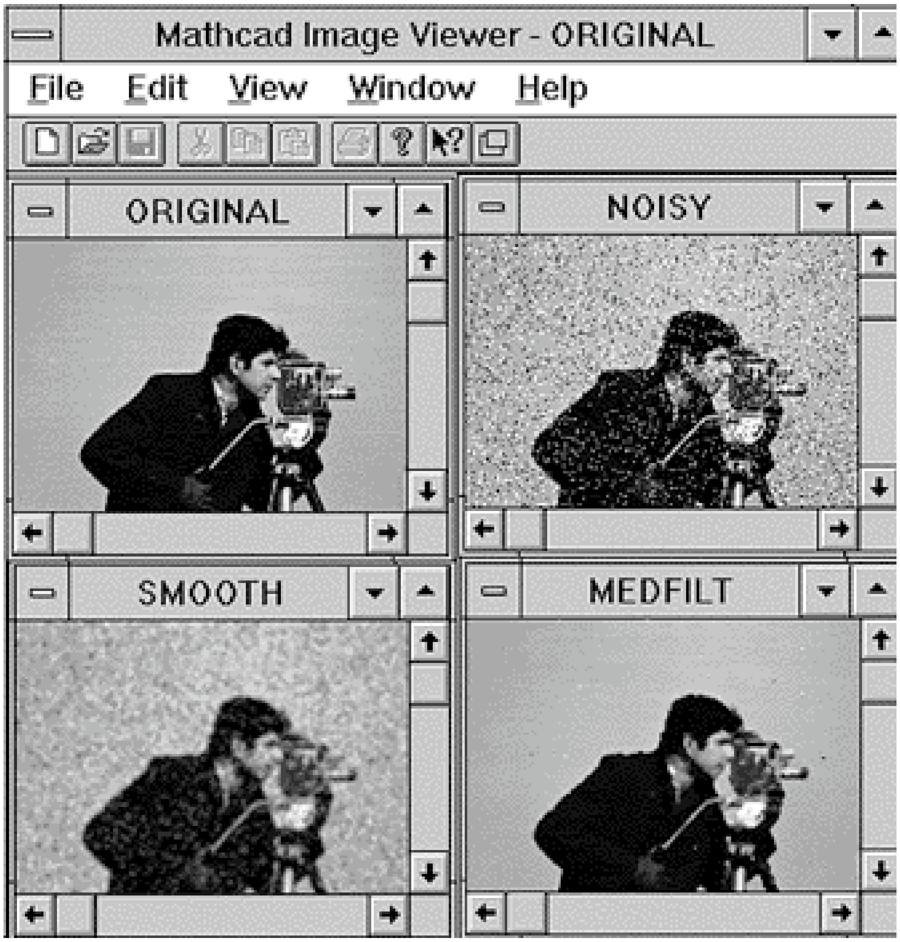
Image Processing Function Pack

Platform: Windows
Requires Mathcad PLUS 5.0 or higher, 5 MB hard disk space
Available for immediate download (size 2613968 bytes) or ground shipment

This Function Pack lets you explore image processing with Mathcad. Perform smoothing, crisping, edge detection, erosion, and dilation algorithms on color and grayscale images. Or, compare different convolution kernels and experiment with filters in the Fourier transform domain. There's even a 2-D wavelet transform function allowing you to experiment with this popular transform method. And you can read and write images in various formats, including BMP, GIF and JPEG. In addition to all the built-in processing functions, there's an image viewer which updates images automatically along with your equations.

Table of Contents | Product Sample | Back to Product List



Take a grayscale image and modify the color ranges and intensities to create different effects.

The Function Pack also comes with a built-in Electronic Book which includes examples to illustrate various image processing applications, a library of sample images, and documentation for the functions. Topic include: Binarization and Quantization, Thresholding, Histograms, False Color Imaging, Median Filtering, Boolean Image Operations, Wavelet Filtering, and much more.

Image Processing Function Pack

TABLE OF CONTENTS (page 1 of 2)

Product Sample

Back to Product List

Image Processing Function Pack

TABLE OF CONTENTS (page 2 of 2)

Product Sample

Back to Product List

Image Processing Function Pack

Convolution

Function Definitions and Syntax for convolve3(M,K), convolve5(M,K) and convol2d(M,K)

These functions perform the convolution of a square kernel of specified size (3 x 3 or 5 x 5), or arbitrary size, with the image matrix **M** . They take two arguments:

· **M**, the image matrix, and

· **K**, the square kernel for **convolve3** or **convolve5**, or an arbitrary kernel for convol2d.

The functions return a matrix containing the image convolved with the kernel. The last function, **convol2d**, convolves a kernel of arbitrary size, but takes longer than the functions specifically designed for 3 x 3 or 5 x 5 kernels.

Note that **convolve3** and **convolve5** are performed in the time domain, where edges are treated with whatever portion of the kernel overlaps them. **convol2d** is performed in the frequency domain, where the original matrices are zero-padded in the time domain before being transformed. As such, there will be slight edge and DC offset differences between the results of applying **convolve3** (or **convolve5**) and **convol2d** to the same matrices.

Examples are shown below. Convolution is an important filtering technique for smoothing, edge finding, and crisping.

Function Details

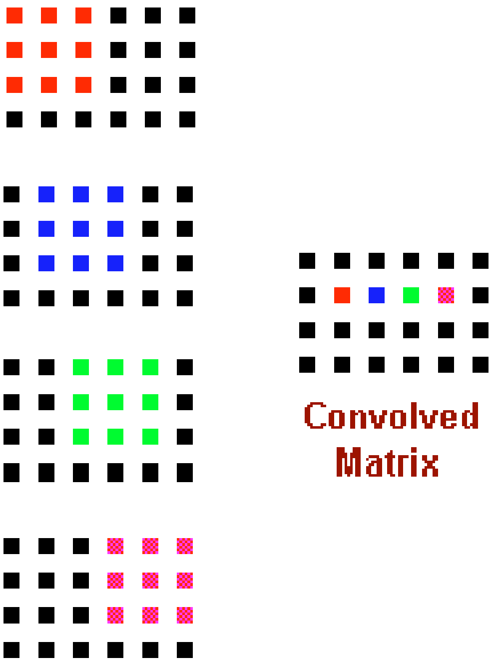
Two-dimensional convolution is one of the most common methods of mathematically filtering an image, and deserves a little explanation. Convolution is the process of sliding a filter matrix (called the kernel) across the image matrix, multiplying each element in the overlapping areas, summing these results, and inserting this sum into an element of the resulting matrix. This is illustrated in the drawing below. A 3 x 3 kernel is used in this example. The colored squares in the convolved matrix correspond to the multiplication and summation in correspondingly colored overlap regions.

Table of Contents

Back to Product List

Image Processing Function Pack

SAMPLE PAGE (page 2 of 7)



Convolution is a central concept in image processing, precisely because it corresponds to multiplication in the spatial frequency domain. In plain words, this means that convolution is equivalent to filtering out image components of some frequency. The exact type of filtering depends on the kernel used. Some kernels will filter high frequency components (image groupings which appear periodically with a small period), and some will filter low frequency components. Examples of different kernels are given here and in the next few sections.

convolve3(M,K)

This function performs convolution of **M** with **K**, where **K** is a 3 x 3 matrix.

$i := 0 .. 4$ $\quad j := 0 .. 5$ $\quad M_{i,j} := \mathrm{floor}(\mathrm{rnd}(25))$

Table of Contents

Back to Product List

Image Processing Function Pack

Define a kernel:

$$K := \begin{pmatrix} 1 & 2 & -1 \\ 1 & 3 & 1 \\ 2 & 4 & 1 \end{pmatrix} \qquad CONV := \mathrm{convolve3}(M, K)$$

$$M = \begin{bmatrix} 0 & 4 & 14 & 8 & 20 & 4 \\ 17 & 7 & 2 & 3 & 24 & 2 \\ 0 & 13 & 15 & 4 & 11 & 1 \\ 19 & 12 & 21 & 23 & 13 & 11 \\ 21 & 19 & 24 & 15 & 6 & 21 \end{bmatrix} \qquad CONV = \begin{bmatrix} 79 & 90 & 79 & 98 & 176 & 88 \\ 67 & 101 & 130 & 102 & 174 & 84 \\ 128 & 190 & 201 & 169 & 196 & 112 \\ 159 & 229 & 286 & 229 & 173 & 155 \\ 108 & 124 & 137 & 129 & 92 & 104 \end{bmatrix}$$

Notice that when part of the overlapped kernel lies outside the image **M**, the convolution sum contains fewer than nine terms. In effect we have padded the image with zeros. For example, the upper left output element is:

$$3 \cdot 0 + 4 \cdot 17 + 1 \cdot 4 + 1 \cdot 7 = 79$$

In **Mathcad**, you could define the convolution function as

$$\mathrm{conv}(i, j, n) := \sum_{k=0}^{n-1} \sum_{d=0}^{n-1} K_{k,d} \cdot M_{k+i-1,d+j-1}$$

where **i** and **j** are the row and column indices of the output matrix, and **n** is the size of the (square) kernel.

$$\mathrm{conv}(2, 3, 3) = 169 \qquad CONV_{2,3} = 169$$

Notice that if you give this function an **i** or **j** corresponding to an edge point, an error will result, since we haven't made allowances for the case where the whole kernel doesn't overlap.

Table of Contents

Back to Product List

Image Processing Function Pack

convolve5(M,K)

This function performs convolution of **M** with **K**, where **K** is a 5 x 5 matrix.

$$K := \begin{bmatrix} 2 & 1 & 0 & 1 & -3 \\ 2 & 1 & 4 & 1 & 1 \\ -1 & -1 & -2 & 2 & 0 \\ 3 & -1 & 0 & 5 & 1 \\ 0 & -1 & 1 & -3 & -1 \end{bmatrix} \qquad M = \begin{bmatrix} 0 & 4 & 14 & 8 & 20 & 4 \\ 17 & 7 & 2 & 3 & 24 & 2 \\ 0 & 13 & 15 & 4 & 11 & 1 \\ 19 & 12 & 21 & 23 & 13 & 11 \\ 21 & 19 & 24 & 15 & 6 & 21 \end{bmatrix}$$

$$\mathrm{CONV} := \mathrm{convolve5}(M,K) \qquad \mathrm{CONV} = \begin{bmatrix} -9 & -20 & 46 & 102 & -37 & -61 \\ 40 & -3 & 11 & 143 & 74 & 20 \\ 86 & 64 & 120 & 113 & 129 & 128 \\ 134 & 194 & 158 & 134 & 202 & 34 \\ 73 & 103 & 96 & 149 & 154 & 59 \end{bmatrix}$$

In this case, the outer two rows will follow a different formula, because of kernel overlapping limitations.

convol2d(M,K)

The **convol2d** function uses transforms to carry out the convolution. **Mathcad** pads the two arrays with zeros, applies the FFT to both, multiplies the transforms elementwise, and takes the inverse transform. This arithmetic gives the standard signal-processing convolution, in which the kernel is rotated 180 degrees before we slide it across the image. Thus for a 5 X 5 kernel **K**,

$$\mathrm{convol2d}(M,K)$$

is equivalent to

$$\mathrm{convolve5}(M,\mathrm{rotate90}(\mathrm{rotate90}(K)))$$

For 3 X 3 and 5 X 5 kernels, **convol2d** will generally be slower than the dedicated functions, but it can be used with kernels of arbitrary size, including nonsquare ones.

Image Processing Function Pack

SAMPLE PAGE (page 5 of 7)

One interesting kernel is the Gaussian kernel, which produces a smoothing effect. Here, a 5 X 5 kernel is generated.

$M := \mathrm{READ\_IMAGE}(\mathrm{bear}_{\mathrm{gif}})$ $\qquad$ $\mathrm{VIEW}(\mathrm{original}) := M$

$i := 0..4$ $\qquad$ $j := 0..4$

Table of Contents

Back to Product List

Image Processing Function Pack

$$K_{i,j} := \exp\left[-\frac{(i-2)^2}{1.5}\right]\cdot\exp\left[-\frac{(j-2)^2}{1.5}\right]$$

$$K = \begin{bmatrix} 4.828\cdot10^{-3} & 0.036 & 0.069 & 0.036 & 4.828\cdot10^{-3} \\ 0.036 & 0.264 & 0.513 & 0.264 & 0.036 \\ 0.069 & 0.513 & 1 & 0.513 & 0.069 \\ 0.036 & 0.264 & 0.513 & 0.264 & 0.036 \\ 4.828\cdot10^{-3} & 0.036 & 0.069 & 0.036 & 4.828\cdot10^{-3} \end{bmatrix}$$

$M := \mathrm{submatrix}(M, 58, 196, 72, 200)$

$CONV := \mathrm{convolve5}(M, K)$

$CONV := \mathrm{scale}(CONV, 0, 255)$

$\mathrm{VIEW}(\mathrm{conv5}) := CONV$

Table of Contents

Back to Product List

Image Processing Function Pack

Compare this with the results of the convolution of general size. Notice the difference in calculation time as well. Since the kernel is symmetric, the two convolutions give the same result:

CONV := convol2d(M, K)

VIEW(arb) := scale(CONV, 0, 255)

Table of Contents

Back to Product List