FFW - Fastest Filtering in the West

The **FFW** package is an FFT-based algorithm for a fast 2D convolution using the overlap-add method. The overlap-add method is based on the fundamental technique in DSP: decompose the signal into simple components, process each of the components in some useful way, and recombine the processed components into the final signal. This is possible since the convolutional operator is linear. The **FFW** package works similarly to fftfilt function (Matlab Image Processing Toolbox) but in a deeper way: all possible lengths for vectors are considered and not only lengths which are powers of two. This is highly necessary since the FFTW package (for more details visit http://www.fftw.org) includes codelets optimized also for other fixed sizes. Codelets are produced automatically by the FFTW codelet generator: you can add your own codelets and re-calculate the execution times for each FFT. The execution times for:

- FFT of real 1D vectors
- FFT of complex 1D vectors
- IFFT of complex 1D vectors with 'symmetric' option enabled
- IFFT of complex 1D vectors with 'nonsymmetric' option enabled

have been calculated with the script papiclock.m from length N = 1 up to length N = 2048. These times have been determinated using PAPI for Matlab (available here or at http://icl.cs.utk.edu/papi). A 2D FFT (see Matlab command fft2) is decomposed into several 1D FFTs: the FFT operator for an N-dimensional array can in fact be splitted into several 1-dimensional FFTs of monodimensional arrays. The **FFW** algorithm automatically selects which is the best choice (first dimension, second dimension and best lengths for overlap-add method) and calculates the 2D convolution.

The **FFW** package can be easily used to improve speed performances of:
2D convolution (Matlab function conv2)
2D filtering (Matlab function filter2)
2D cross-correlation (Matlab function xcorr2)
Normalized cross-correlation (Matlab function normxcorr2)

How does **FFW** package work?
In order to find the best parameters for overlap-add method an exhaustive search on 2D matrices would not be possible. The computational cost of FFT2 operator is done decomposing it into the computational costs of two series of FFTs on monodimensional arrays. For example, if you want to calculate the FFT2 computational cost using a matrix N x M as input, you will perform the following sum: N*cost(FFT(M)) + M*cost(FFT(N)) where cost(FFT(X)) is the computational cost of FFT operator using as input a vector whose length is X. The computational cost of FFT for a real vector is, in general, different from the cost of FFT for a complex vector. For this reason more than one choice is possible: you can choose the first dimension along which you can apply the FFT operator. Analogous considerations can be made for IFFT operator.

After the minimum computational cost has been found, a finer tuning is possible: the **FFW** algorithm makes a quasi-exhaustive search using an optimized algorithm. The fine-tuning option requires a lot of time but it is recommended for high-performances FFT-based filtering. Of course, this option has sense only when you have to do several convolution products.

If both input images are real **FFW** algorithm uses 'symmetric' option when using IFFT operator. The optimized parameters for **FFW** algorithm depend only on sizes of input matrices and on their values (real or complex). If the same filter has to be applied to several images, its FFT2 value can be determined only one time, saving computational time. In this case (the same filter applied to several images) the determination of optimized parameter must not take into account the computational cost of such operation, since it is done only once. **FFW** algorithm can also work in time domain: this choice is necessary for small filter kernel, using standard conv2 built-in Matlab function.

Please contribute if you find this software useful.

Report bugs to luigi.rosa@tiscali.it

Luigi Rosa
Via Centrale 35
67042 Civita Di Bagno
L'Aquila – ITALY
mobile +39 3207214179
email luigi.rosa@tiscali.it
web site http://www.advancedsourcecode.com