where $a_n(\mathbf{x})$ and $\nabla\varphi_n(\mathbf{x})$ are the amplitude and frequency modulation functions, respectively, for a given component $n$. Here, $\nabla$ denotes the gradient operator. Note that estimating $a_n(\mathbf{x})$ and $\nabla\varphi_n(\mathbf{x})$ and selecting an $N$ is an ill-posed problem. Even with $N = 1$, there are uncountably many pairs of modulation functions which will *exactly* represent the image [1].

Currently, there are three different analysis paradigms used to compute AM-FM representations of images. They are known as the Dominant Component, Channelized Components, and Tracked Components paradigms. Each of the three begins by filtering the input image with an orientation and frequency selective Gabor filterbank. They differ primarily in how they choose to combine the estimated modulation functions from each channel of the filterbank into components. For this project, we chose to implement the Channelized paradigm. It produces the most accurate representation of the image (in terms of being able to reconstruct the original), and is not currently under development.

The facts about the Channelized Components algorithm which are particularly relevant to this report are about its structure. The structure of the algorithm is shown by the block diagram in figure 1.

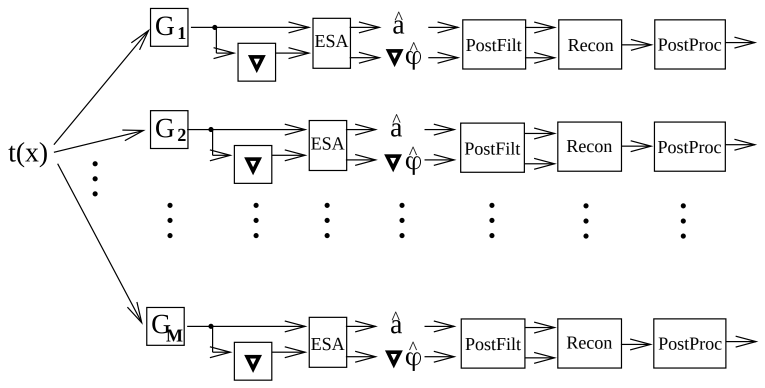

**Figure 1: Block diagram of the Channelized Components Algorithm**

The blocks labeled $G_1 \cdots G_M$ represent the Gabor filters. The estimation part of the algorithm begins at the outputs of the filters and ends at the signals labeled $\hat{a}$ and $\nabla\hat{\varphi}$. After this, the estimated modulating functions are themselves filtered to prepare them for reconstruction into components. Finally, the components are reconstructed and post-processed for viewing.

## 3. Implementation in Khoros

Khoros is a rapid prototyping and algorithm development environment that was originally designed for image processing research, and is currently in widespread use in both industry and academia. The Khoros system consists primarily of three parts: application toolboxes, the Cantata visual programming language, and a software development system designed to allow users to extend Khoros with new application toolboxes. Since the intent of the Khoros interface is to allow rapid algorithm development, we take advantage of the visual programming capabilities of Cantata and work around its limitations.

Cantata provides the benefit of allowing algorithms to be specified visually as a directed graph. However, it also has several limitations which affected the design of the Khoros interface to the AM-FM analysis code. First, Cantata is intended to model at the level of a large-grain dataflow system. In fact, each actor (known as a *glyph* in Cantata) is an executable file that is also available through the command line. Probably due to this large-grain dataflow model, the established model for communicating data between glyphs is geared towards data which can be described as one or more samples with spatial, temporal, and bit-plane indices. It does not allow passing of general data structures between blocks. It is necessary for data to fit into the model in order to use the Khoros/Cantata infrastructure. Another limitation in Cantata is the absence of any support for higher-order functions. Any repeated, parallel glyphs and connections must be explicitly laid out. Cantata does have support for conditional statements and looping constructs. However, while they are useful, they do not make up for the lack of higher-order function support. Finally, Cantata only allows scalar global variables in the visual program. Therefore glyphs cannot share common copies of any nontrivial data types such as matrices or text strings.

Attempting to fit the large-grain dataflow model that Cantata assumes and considering ease-of-use, we broke the Channelized Components algorithm into five sections and implemented each as an independent glyph. These five sections closely follow Figure 1. The first glyph performs the filtering operations in the first column of blocks. The second section estimates the modulation functions. The third glyph postfilters the estimates and prepares them for the following reconstruction glyph. This converts the AM-FM representation of the component back into a spatial domain representation. The fifth and final glyph is used to condition the reconstructed components for viewing or summing to produce an estimate of the original input image. Figure 2 shows a screen shot from Cantata with the AM-FM analysis glyphs. It is important to note that the current implementation communicates data between glyphs using disk files rather than the dataflow connections in Cantata. The user artificially creates the data dependencies using the control connections on the glyphs. The control connections simply indicate to Cantata's scheduler that a given block must terminate before the one it is connected to can be dispatched. This design greatly improves ease of use and increases the independence of the glyphs. Each glyph needs to be connected to at most two control connections rather than the many parallel data connections that would be needed otherwise.

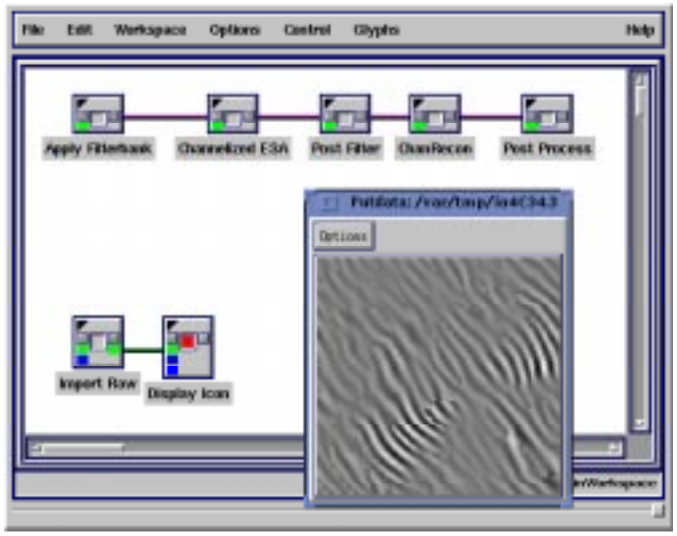

**Figure 2: Cantata's main window showing the AM-FM glyphs**

In addition, since data transfer is done via disk files, the user can always opt to use stored outputs of a glyph's predecessor as input rather than waiting for the predecessor to be run again.

## 4. Implementation in Ptolemy

When modeling in Ptolemy, we chose to use MDSDF (MultiDimensional Synchronous DataFlow) was used. A crucial difference between SDF (Synchronous DataFlow ) and MDSDF is explained using the token-flow diagrams in the figure below [3].

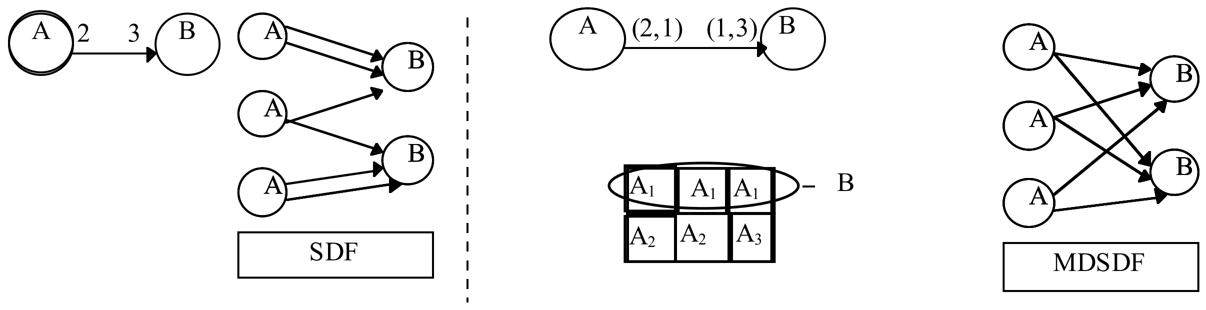

**Figure 3: Advantages of MDSDF**

If SDF is used, actor B will use the tokens as shown on the left in Figure 3. If it is desired for B to use the first token from each firing of A then an MDSDF model of computation where A will produce (2,1) size tokens and B will consume (1,3) size tokens as shown on the right. We will explain how this property is used in the implementation.

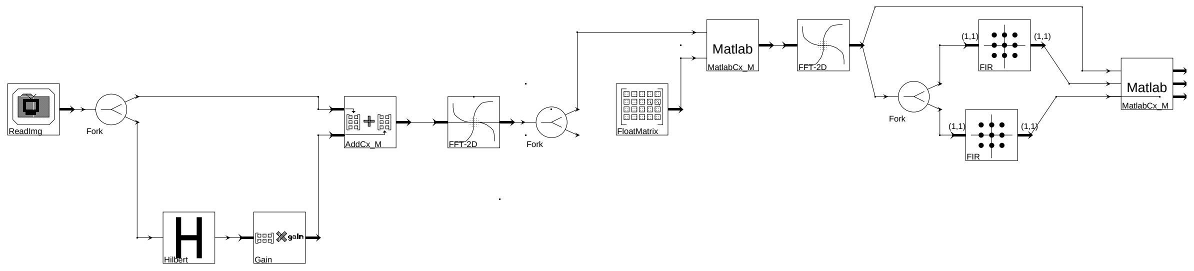

**Figure 4: Implementation in Ptolemy**

In this figure, the first actor reads the image row by row, then the actor labeled "H" takes the Hilbert Transform in the horizontal direction. Since the transform is only taken in one direction, this actor takes the image row by row as well. The 2D FFT first takes the FFT along rows and then along columns of the image. The image is processed row by row up to the column FFT, and then it accumulates. The filterbank comes next (only one channel is drawn for simplicity). The MATLAB labeled actor evaluates the filter function in the frequency domain and term by term multiplies with the FFT of the image. The FIR filters find the differences between the neighboring pixels in preparation for the final actor. This last actor implements the ESA (Energy Separation Algorithm), is where the estimates for the coefficients are found.

The same function can be implemented using different structures for the filter. There are three possible options:

1. Saving a matrix containing the frequency domain values of the filter and term by term multiplying with these values (because of symmetry, saving one fourth of the filter is enough). This is fast but uses a lot of memory.

2. A lookup table containing the Gaussian function can be used. This uses less memory but uses more time while checking the lookup table. Symmetry is also taken advantage of when storing the lookup table. Even fewer samples can be stored and interpolation can be used to find the in-between values)
3. Two FIR filters can be used because the Gabor filters are separable. In this option the FFT and the inverse FFT is not required. Since the process is finer grain, it improves parallelism.

If hardware implementations are desired, most problems will occur at the ESA where trigonometric functions are evaluated. CORDIC (COordinate Rotation Digital Computer) techniques can be used where only shift and adds are used to solve trigonometric equations of certain form [4].

## 5. Conclusion

We have implemented the Channelized Components AM-FM image modeling algorithm in both Khoros and Ptolemy. The algorithm can be implemented in parallel, but the code written in C was not. By using Ptolemy and the MDSDF domain we achieved parallelism in most parts (except column FFT and FIR filters on the last stage). Signal processing tools like MATLAB are good at computation, but by using Ptolemy, which can call MATLAB, embedded system design is made easier. The interface to Khoros allows easy, flexible access to the AM-FM analysis code both for further development of the AM-FM analysis algorithms themselves as well as their inclusion in designs for other algorithms. Further integration with Khoros resulting in a finer grain modeling of the AM-FM analysis algorithms is possible. It would be necessary to make