Interactive Photo Editor FrontEnd using MathLink

Junzo SATO, RoboDog, JAPAN
Chikara MIYAJI, University of Tsukuba, JAPAN
Mariusz Jankowski, University of Southern Maine, USA

Last updated on 7/2/1999.

Abstract

It is not easy for users of commercial photo editing software products to add extensions, create new processing functions and customize the graphical user interface (GUI). In contrast, MATHEMATICA is an inherently extensible system for general-purpose scientific computing, since most of its features are defined by user programmable symbolic expressions. However, the existing Notebook FrontEnd lacks interactive processing such as direct manipulation of the graphic image with a mouse. Thus the authors have developed, using MathLink, an Interactive Photo Editor FrontEnd (for short, Photo Editor) with a variety of extensions. With the Photo Editor, the user can paint pictures, interactively edit and annotate images and customize the GUI by adding functions and menu items. The advantages of this software are as follows:
1) Mouse and menus are available to draw pictures, similar to a painting program.
2) Supports various image file formats.
3) Photo processing operations are defined as MATHEMATICA functions.
4) GUI and mouse actions are defined as MATHEMATICA functions.

Interactive Photo Editor FrontEnd Features

This Photo Editor is implemented as a *MathLink* program. Several advantages of this software are described below.

Interactive Photo Editing

Interactive mouse- and menu-driven interfaces are common in general drawing and photo editing application programs. Unlike the standard MATHEMATICA Notebook FrontEnd, the Photo Editor presents to the user with an interactive window which permits mouse-driven region-of-interest selection and menu-driven processing by menu item selection.

MathKernel Work Load Reduction

Generally speaking, image data requires large amounts of memory [reference, see below]. This places a heavy memory burden on the MathKernel since it needs to maintain one or more instances of an image during a processing sessions. For display purposes, the image needs to be transferred to the Notebook FrontEnd, which places additional significant time and memory loads on the system. However, photo editing is usually applied to a portion of the image only. The Photo Editor maintains a copy of the data and sends only the region-of-interest to the MathKernel, thus reducing the kernel memory loads. Because only numerical data is transferred between the Photo Editor and the MathKernel, the data transfer is much faster than between the Notebook FrontEnd and the MathKernel. For example on a Macintosh, a 200 pixels ¥ 200 pixels image was transferred about 16% faster by the Photo Editor than the Notebook FrontEnd.

Image File Formats

In MATHEMATICA, to support a new file format, writing a specific parser in the form of a MATHEMATICA package or MathLink program is needed. Writing such extensions is difficult and inconvenient. When several formats are in use, such as GIF, JPEG, and TIFF, MathLink programs under "SystemFiles/Converters/Binaries/" directory remain active even when they are no longer needed. This is unsatisfactory. The Photo Editor supports various file formats by adopting QuickTime technology invented by Apple Computer. As QuickTime components can be widely used, components developed by third-parties are available without any modification. Inversely, if we write an original component, we can share it with all software which use QuickTime. Default graphics formats supported by QuickTime4 are as follows:

For importing:
QuickDraw PICT, QuickTime Image, MacPaint, Photoshop (ver 2.5 and 3.0),
Silicon Graphics(SGI), GIF, JFIF/JPEG, QuickDraw GX Picture,
Windows Bitmap (BMP), PNG, Targa, TIFF, FlashPix.

For exporting:
QuickDraw PICT, QuickTime Image, MacPaint, Photoshop (ver 2.5 and 3.0),
Silicon Graphics(SGI), JFIF/JPEG, Windows Bitmap (BMP), PNG, Targa, TIFF.

Because the image data can be transferred freely between the Photo Editor and the Notebook FrontEnd, all these formats are available from MATHEMATICA.

Plug-ins

The Photo Editor allows the connection of many kinds of plug-ins with MATHEMATICA. The support of TWAIN format enables us to input an image directly from a scanner to MATHEMATICA. QuickTime components are useful not only for data I/O, but also for image processing effects. If the Photo Editor supports plug-ins such as those in found in Adobe Photoshop (R) and others, MATHEMATICA benefits from this property.

Advanced Image Processing

After the selection of an image area, a function defined in the MathKernel is used for image processing. Complex image processing is described in a highly abstract form. Any desired image processing operation may be expressed as a MATHEMATICA program, using MATHEMATICA's excellent programming language. Moreover, advanced image processing functionality is immediately available with a new Image Processing MATHEMATICA package. Traditional photo editing applications are not suitable for the purpose of batch processing. It is however possible with the Photo Editor because the processing can be described as a MATHEMATICA program.

Utilization As A Teaching Tool

Although image processing is simply written as a MATHEMATICA function, it is difficult to interest students by letting them confirm results only on Notebooks. The actual feeling of the application will be gotten when they can import various formats of images, like existing applications, and apply image processing functions defined by themselves. On this point, the Photo Editor is suitable for students to practice their image processing.

Implementation

Serializer

The Photo Editor is used with the Notebook FrontEnd. For example, defining functions is done in the Notebook FrontEnd and manipulation of an image area is done in the Photo Editor. In this way, the function of the Notebook FrontEnd is strengthened and its advantage is utilized. However, existing MATHEMATICA didn't have a way to use multiple frontends. With the Serializer, a new MathLink program, this is possible. The Serializer is placed between the Notebook FrontEnd and the MathKernel, and relays expressions. It can also send expressions from a frontend, with the exception of the Notebook FrontEnd, to the MathKernel. The Photo Editor is connected to the Serializer.

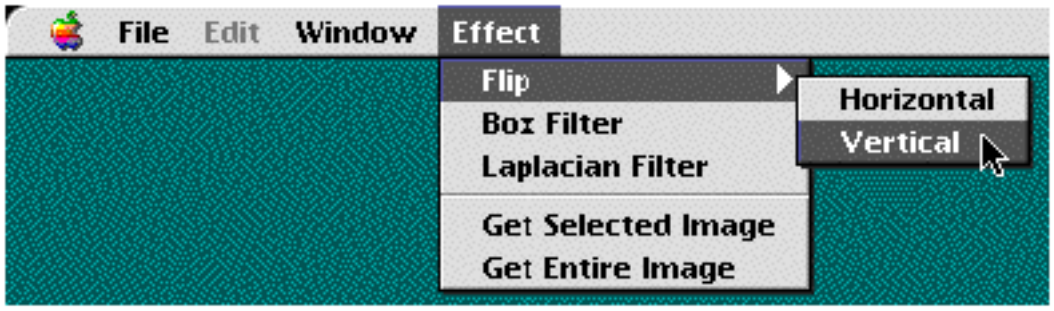
Object Oriented Programming

The method of Object Oriented Programming was adopted to implement the Photo Editor. For example, a menu class is defined in the MathKernel. Creating a menu object in MATHEMATICA corresponds to adding a menu to the Photo Editor. To add an arbitrary menu to this Photo Editor and set menu commands in it, a message is sent to this menu object. When a menu command is selected in the Photo Editor, a message to handle this command is sent from the Photo Editor to the menu object in the MathKernel via the Serializer. The method executed at this time can be freely defined.

Example Of Photo Editor

Menus

Photo Editor menu commands are added to hierarchical menus by well defined operations in the Notebook FrontEnd.

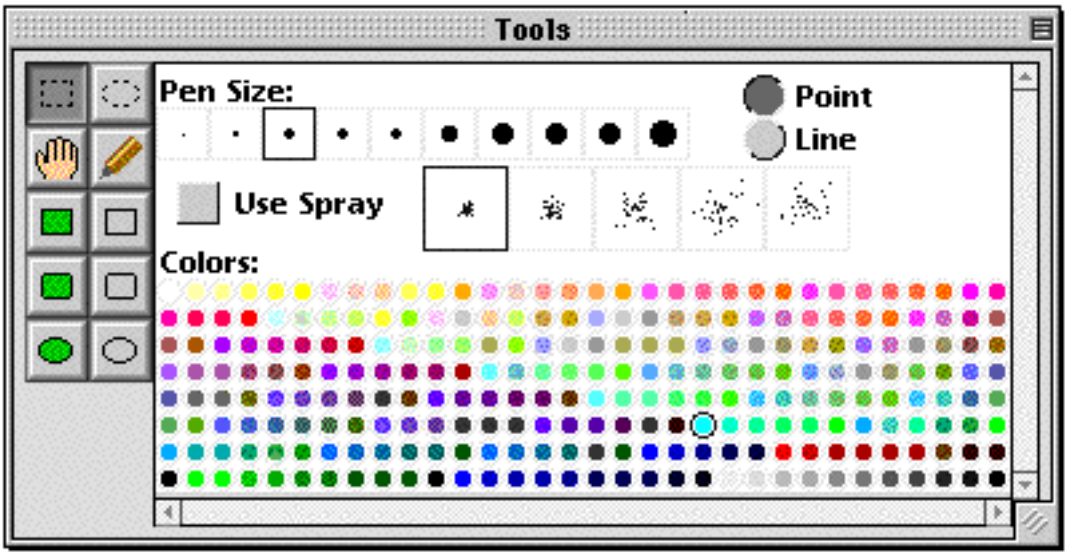


For example, an "Effect" menu is created as follows. Two objects, menuEffect and subMenuFlip, are constructed from the menu class CMenu.

```
menuEffect=New[CMenu,132,"Effect"];
menuEffect[
    SetupCommands,
    {"Flip",
      "Box Filter",
      "Laplacian Filter",
      "(-",
      "Get Selected Image",
      "Get Entire Image"},
    {-1101,-1102,-1103,0,-1105,-1106}];

subMenuFlip=New[CMenu,200,"",-1];
subMenuFlip[
    SetupCommands,
    {"Horizontal",
      "Vertical"},
    -2000];
subMenuFlip[InstallToSuperMenu,menuEffect,1];
```

Tools View

Some tools, such as a marquee, a pen, etc., are prepared so that a user can manipulate a picture using a mouse. Also custom control items like tables and radio buttons are defined on a tools view. The following sample is a palette which has controls to set a pen size, colors etc. The operation to generate a spray uses the Standard AddOn package Statistics`ContinuousDistributions`NormalDistribution[].



Because custom control items are defined as MATHEMATICA expressions, it is possible to construct an original multi-selectable group and a radio group. All items utilize the method of Object Oriented Programming. The tools view sends MATHEMATICA a message which indicates that a mouse click has occurred.

By combining several tools, defining event handlers, and calling specialized photo editing functions within them, we can establish a system which can be used as a general purpose, modern, highly customized and advanced painting and photo editing program. The next figure is a screen shot of the painting.

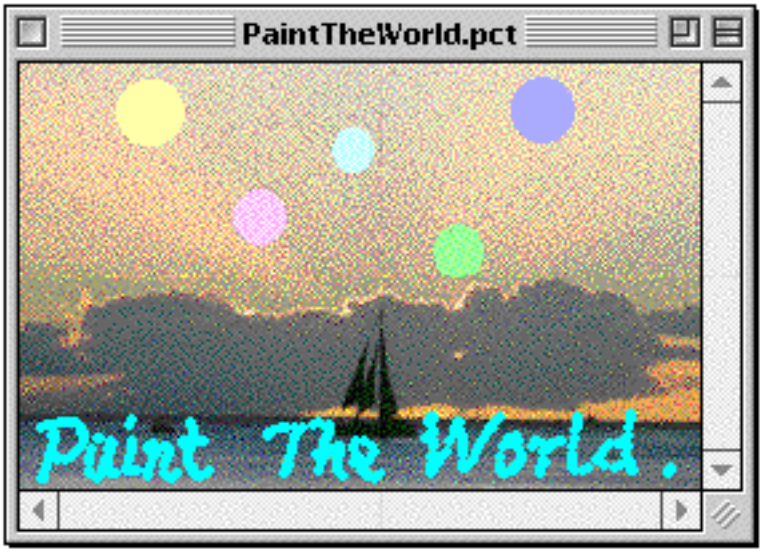


Photo Editing

To apply an effect to some area inside the marquee rect, the data should be sent to the MathKernel by the operations of the menu command handler which corresponds to the effect. The image processing defined in MATHEMATICA is then applied to this data. Finally, the resulting data is sent back to the Photo Editor. In the context of an image analysis application, where it is unnecessary to return the data, only the result of analysis needs to be outputted.

In the example "Effect" menu, a DoCommand method in the menuEffect object is defined. A menu command sent to the MathKernel is dispatched to the corresponding handler.

```
menuEffect[DoCommand,command_Integer,itemNumber_Integer]:=Module[
    {},
    Switch[
        itemNumber,
        2,doBoxFilter[],
        3,doLaplacianFilter[],
        5,doGetImage[],
        6,doGetEntireImage[]];];
```

One of authors has developed a MATHEMATICA package for image processing and its functionality is available from the Photo Editor if appropriate menu commands are prepared. The following example uses filtering operators and functions (BoxFilter, LaplacianFilter, EdgeMagnitude) defined in the package.

After loading image processing packages, handlers doBoxFilter[] and doLaplacianFilter[] are defined as follows:

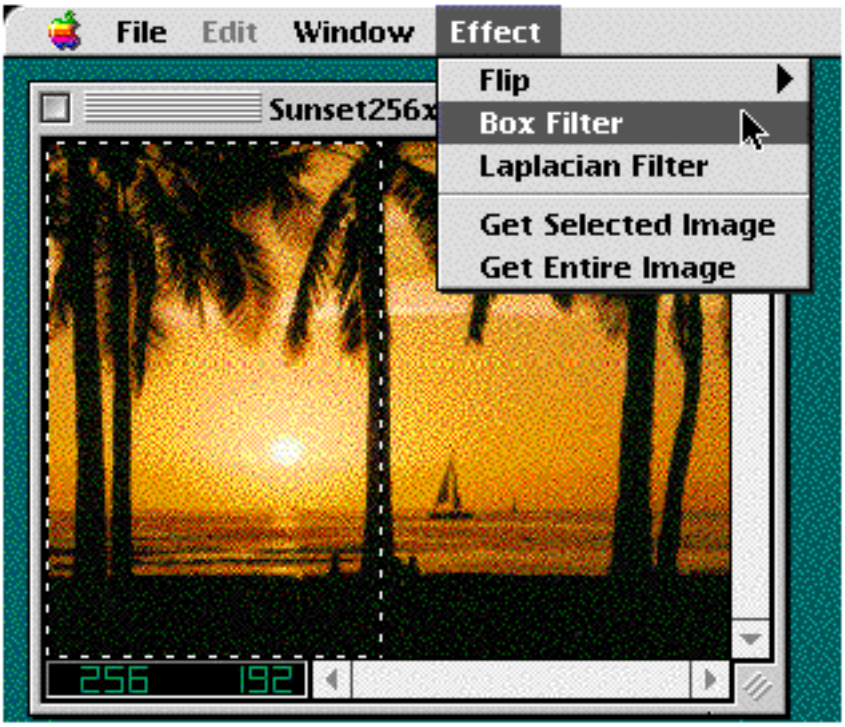
```
<<ImageProcessing`
```

```
doBoxFilter[]:=Module[
    {area,savedTitle,pixelSize,rgbs,imgData},
    {area,savedTitle,pixelSize,rgbs}=preProcess[];
    imgData=ImageData[rgbs,PixelInterleaveÆTrue,ColorFunctionÆRGBColor];
```

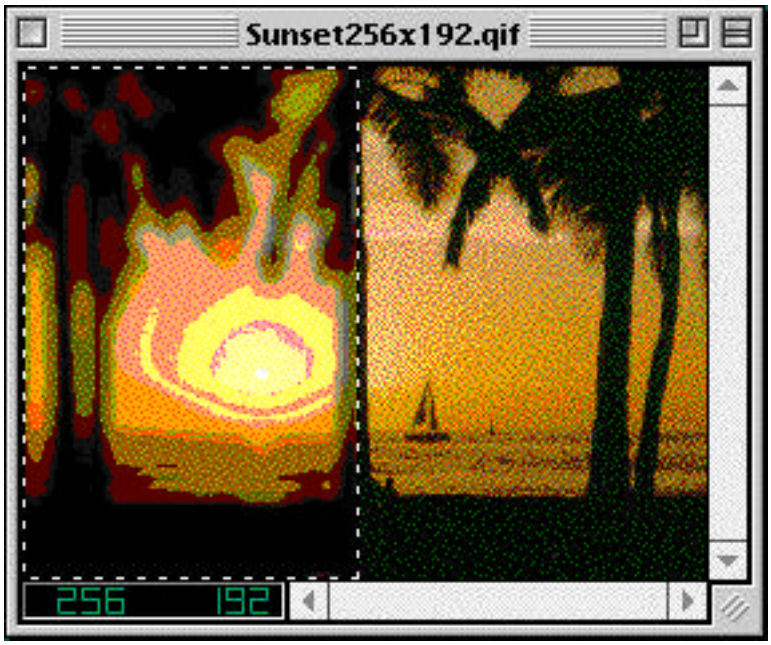
```
    imgData=DiscreteConvolve[imgData,BoxFilter[8,8],CenteredÆTrue];
    rgbs=Round[RawImageData[imgData]];
    targetWindow[TransferRasterArray,
        MacRGBColorsToRasterArray[rgbs],area,
        PixelDepth->pixelSize,Thru->48];
    postProcess[savedTitle];];

doLaplacianFilter[]:=Module[
    {area,savedTitle,pixelSize,rgbs,imgData},
    {area,savedTitle,pixelSize,rgbs}=preProcess[];
    imgData=ToGrayLevel[ImageData[rgbs,PixelInterleaveÆTrue,ColorFunctionÆ-
RGBColor]];
    Show[Graphics[imgData]];
    imgData=EdgeMagnitude[imgData,N[LaplacianFilter[]]];
    Show[Graphics[imgData]];
    postProcess[savedTitle];];
```

The preProcess[] and postProcess[] are functions to process receiving data, change window title, etc. The filter is applied to a photo after conversion of color data into ImageData type, as required by the package.

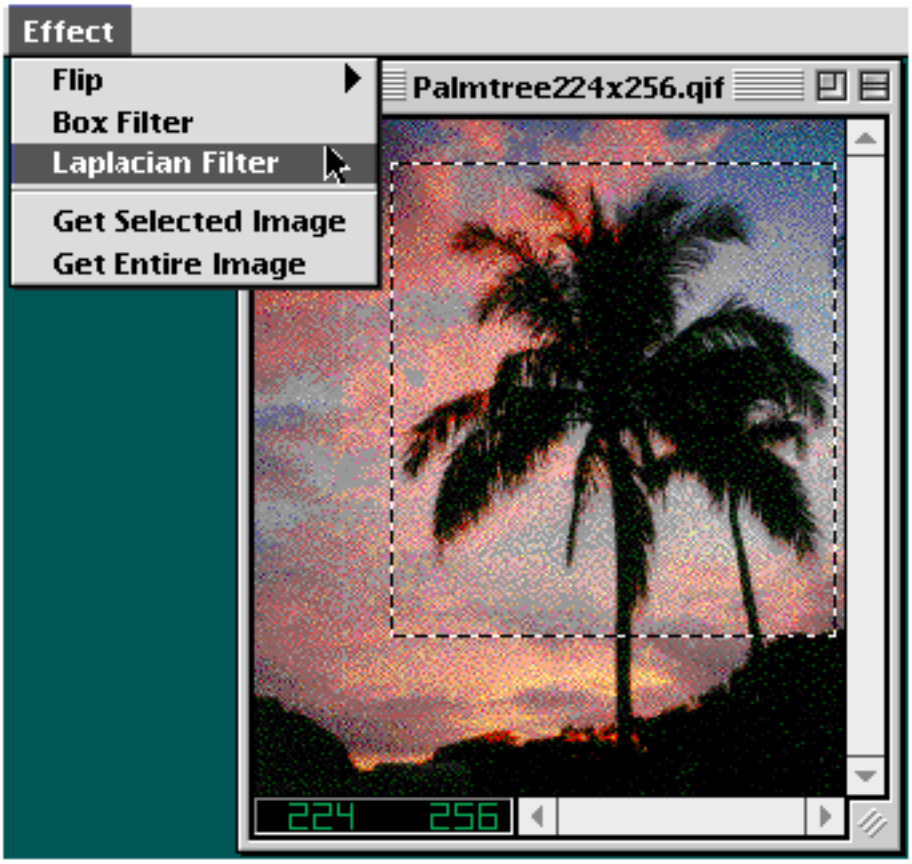
In the following example, the Box Filter is selected and the result is shown.

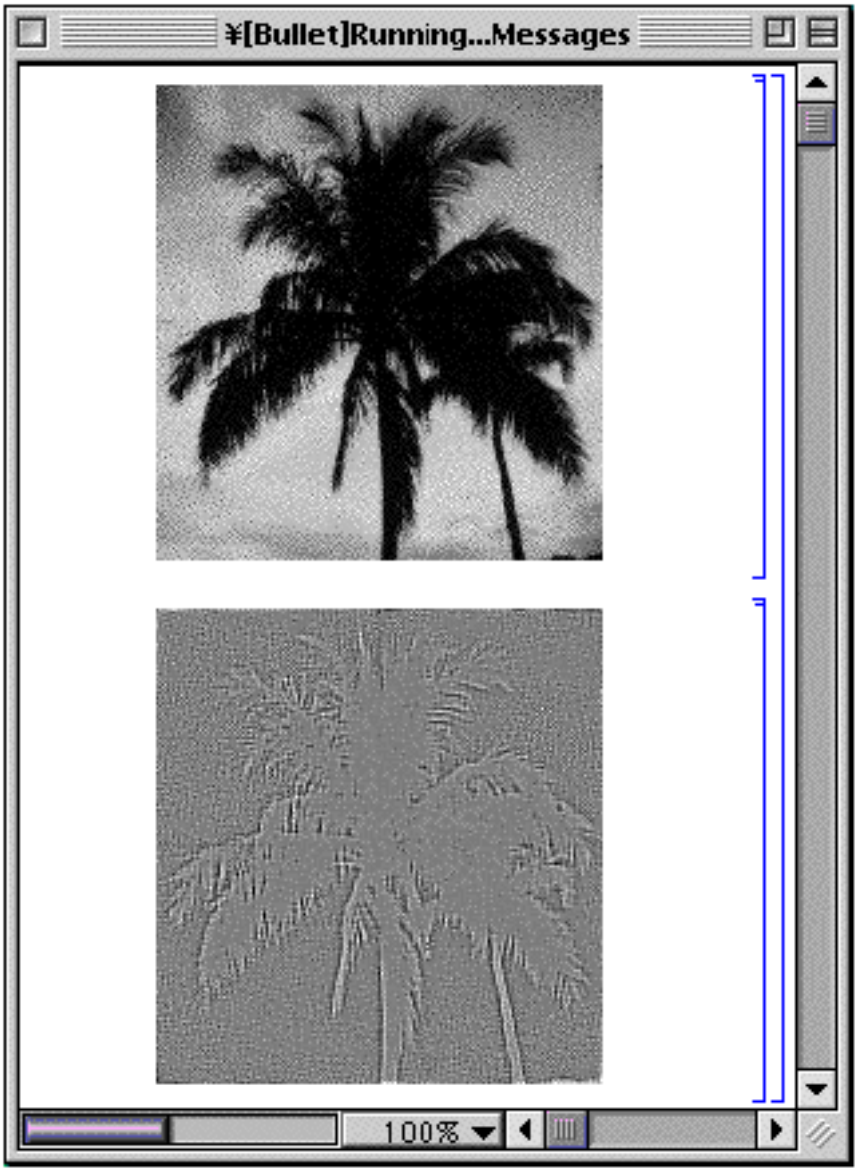


After application of the Box Filter, the image is updated.



In the following example, the Laplacian Filter is selected and the result is shown on a Notebook.





Summary

Using *MathLink*, an new photo editing interface has been created. It possesses interactivity which the existing Notebook FrontEnd doesn't have. The flexible environment has been constructed by defining responses to a mouse, menus, and painting tools as MATHEMATICA functions. Various formats of graphics image files for importing and exporting data are supported. Desired photo effects may be applied to whole images, and selected region-of-interest. In MATHEMATICA, these tools and effects may include mathematical expressions. The Photo Editor successfully combines the ease-of-use of traditional photo editing programs with the power of MATHEMATICA's language and advanced computational features. A system with a variety of extensions has been attained.

Acknowlegment

One of authors, Junzo Sato expresses his gratitude to Jessica L. Patterson.

References

[1] Power Programming with *Mathematica*, David B. Wagner, McGraw-Hill, 1996

[2] Network Programming using *Mathematica* (Japanese), Chikara Miyaji, Iwanami, 1998

[3] *MathLink*: Network Programming using *Mathematica*, Chikara Miyaji and Paul Abbott, Cambridge University Press, 1999 (in printing)

[4] Interactive Graphics using *MathLink*, Chikara Miyaji, IMS'99, 1999

[5] Digital Image Processing with *Mathematica*, Mariusz Jankowski, Worldwide *Mathematica* Conference, Chicago, 1998.